Nicholas Tann

Lab 2

ECEN 449-503

2/2/2020

Introduction:

In Lab 2 we went through the process of setting up Xilinx xps, our software suite for using synthesizing a processor and programming the synthesized computer system. We set up a new project and system to repeat the functionality of lab 1 but with software.

Procedure:

This lab was divided into two parts. The first part of the manual demonstrated how to set up the MicroBlaze processor and instructed how to write and import C code so the hardware can function properly. The instructions for the first part as to how to set up a GPIO block and set up the peripherals and load it onto the hardware was self explanatory in the lab manual. It really helped see the differences between designing and exporting something in C using a pure hardware method vs. designing it in Verilog. The second part instructed us to have the FPGA perform different functions, such as incrementing and decrementing using two buttons, displaying the current value, and displaying the status of the switches on the board. Setting up the MicroBlaze processor was similar to the first part, except I needed to add an 8-input GPIO block for buttons and switches on the board in addition to the 4-output one already present for the LEDs in part one.

Results:

source files Included in lab2.zip

Conclusion:

The project was much more difficult to set up and had a lot more overhead. After the initial setup however, the ease of programming the desired functionality was much greater than programming in Verilog. Adding complex logic is a breeze compared to writing always blocks in verilog.

Questions:

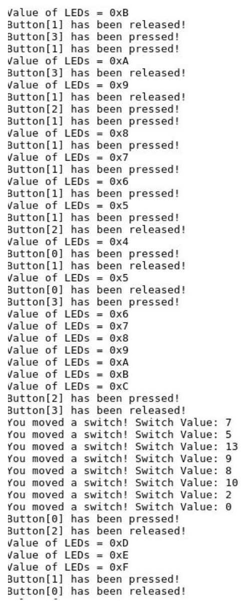
1. In the first part of the lab, we created a delay function by implementing a counter. The goal was to update the LEDs approximately every second as we did in the previous lab. Compare the count value in this lab to the count value you used as a delay in the previous lab. If they are different, explain why? Can you determine approximately how many clock cycles are required to execute one iteration of the delay for-loop? If so, how many?
    - The count for this lab is much smaller because every while iteration takes multiple cycles. In general we have a comparison (branch instruction) increment (addition) loop (jump), depending on how these instructions are implemented on the processor will determine the number of cycles the while loop takes. In our case about 6 cycles.

2. Why is the count variable in our software delay declared as volatile?
    - Volatile lets the compiler know that the variable can be changed by something other than code (ie hardware). It prevents the compiler from performing optimizations that can mess up the while loop.

3. What does the while(1) expression in our code do?

- Continuously runs the code within the while block. Everything before is initialization.

4. Compare and contrast this lab with the previous lab. Which implementation do you feel is easier? What are the advantages and disadvantages associated with a purely software implementation such as this when compared to a purely hardware implementation such as the previous lab?
   - The implementation of the software is much easier than the hardware and the code much more closely resembles the logic of what we want to do, but the setup of the software is much harder as well as the overhead costs of simulating a processor being much higher.



```
Value of LEDs = 0xB
Button[1] has been released!
Button[3] has been pressed!
Button[1] has been pressed!
Value of LEDs = 0xA
Button[3] has been released!
Value of LEDs = 0x9
Button[1] has been released!
Button[2] has been pressed!
Button[1] has been pressed!
Value of LEDs = 0x8
Button[1] has been pressed!
Value of LEDs = 0x7
Button[1] has been pressed!
Value of LEDs = 0x6
Button[1] has been pressed!
Value of LEDs = 0x5
Button[1] has been pressed!
Button[2] has been released!
Value of LEDs = 0x4
Button[0] has been pressed!
Button[1] has been released!
Value of LEDs = 0x5
Button[0] has been released!
Button[3] has been pressed!
Value of LEDs = 0x6
Value of LEDs = 0x7
Value of LEDs = 0x8
Value of LEDs = 0x9
Value of LEDs = 0xA
Value of LEDs = 0xB
Value of LEDs = 0xC
Button[2] has been pressed!
Button[3] has been released!
You moved a switch! Switch Value: 7
You moved a switch! Switch Value: 5
You moved a switch! Switch Value: 13
You moved a switch! Switch Value: 9
You moved a switch! Switch Value: 8
You moved a switch! Switch Value: 10
You moved a switch! Switch Value: 2
You moved a switch! Switch Value: 0
Button[0] has been pressed!
Button[2] has been released!
Value of LEDs = 0xD
Value of LEDs = 0xE
Value of LEDs = 0xF
Button[1] has been pressed!
Button[0] has been released!
```