Alan Achtenberg

Lab 2 Software

ECEN 449 Sec:505

Due:9/26/2014

Introduction:

In Lab 1 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:

First we set up a new project in xps. After initializing a new project, we went through the process of configuring the microblaze computer system that would be simulated on the FPGA.

Once setup of hardware is complete, we went ahead and set up a software project, configuring the settings to program our C code into the BRAMs. In part 1 we ran code that implemented the functionality of lab1 part a. Our leds was a 4 bit counter with a 1 hz clock, that continually counts up until it rolls over.

For part 2 we implemented the functionality of both lab 1 part 2 and lab 1 part 3, all of the push buttons and the first 4 switches were used. If you pressed the left button the mode of the leds was the up/down counter with reset being the center button. If you press the right button, the mode is set to display the status of the DIP switches. Every time there is a detected change the event is printed to the console along with the value of the LEDS.

Results:

Verilog Source Files

Conclusion:

For the software version of the lab, the project was much more difficult to set up and had a lot more overhead. Thad being said, the ease of programming the desired functionality was much greater than programming in Verilog. Adding very complex logic is a breeze compared to writing an always block.

Questions:

(a) In the first part of 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 for our while every 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 it is approximately 6 cycles.

(b) 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) and it prevents the compiler from performing optimizations that can mess up the while loop.

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

Continuously runs, everything inside the while block is the logic of the application we programmed. Everything before is just initialization of the system.

(d) 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 set up of the software is much harder as well as the overhead costs of simulating a processor being much higher. Lastly you do not have a fine grained cycle by cycle control of the of the hardware like you do with Verilog, because of this you al