Kyle Loyka

Lab 2 Report

ECEN 449-503

Due: 8 February 2016

Introduction
The purpose of this lab was to familiarize ourselves with the XPS environment and to learn how to run C code on the FPGA.

Procedure
The first part of this lab involved generating netlists and bitsreams which allowed the XPS software to compile and run C code on the FPGA. The provided C code was built and uploaded to the FPGA successfully. This code incremented the LED counter at approximately 1Hz.

The second part of the lab involved implementing a program that would increment a counter, or indicate which DIP switches were activated using the built in LEDs on the FPGA board. A new GPIO IP block was created for the 5 push buttons and 4 DIP switches. IF statements were used to influence program behavior based on what buttons were pressed. The conditions of the IF statements were evaluated by ANDing different bits with the GPIO input. This implementation would allow the program to detect if a button or switch was activated.

Results / Q&A

(A) All both parts of the lab were working correctly. In Lab 1, the hardware clock operated at 100MHz. Using Verilog, a clock divider was designed to output a new clock signal at 1Hz. Using this implementation, the count value was 100,000,000. In Lab 2, the delay() function acted as a clock divider. The added delay function made the program loop at approximately 1Hz. In this implementation the count value was 0x1000000 = $(16777216)_{10}$. The differences in count values between Lab 1 and Lab 2 could be caused by differences in execution speed. When using Verilog, the code will run faster as Verilog is a lower level hardware language. When using C, the code takes longer to execute due to the higher level of abstraction that C provides. The C program needs to be translated into a hardware language in order for the FPGA to understand it. This process creates more complex program environment and results in slower execution.

Since the clock on the FPGA board operates at 100MHz and the delay function needs 16777216 loops in order to exit, approximately 6 clock cycles are needed to run the delay function.

(B) The *volatile* attribute prevents the delay function from entering an infinite loop. The *volatile* attribute prevents compiler optimization and is also used when dealing with physical hardware. This attribute forces the program to reload the data from memory, instead of relying on a temporary register. This temporary register could have an incorrect value which results in undesired program execution.

(C) The while(1) expression makes the code execute in an infinite loop. The program should never stop running (unless some unaccounted error occurs).

(D) Lab 1 relied on lower level/ hardware programming. Lab 2 relied on upper level constructs by using the C language. The software implementation (C language) provides the benefit of higher level abstraction, but unlike Verilog, does not provide a means for

parallel processing. Verilog does not have all of the abstraction of C, but does provide a means for parallel processing.

Conclusion

In conclusion, this lab provided our first introduction to writing C for a FPGA board. This was also our first time using the XPS environment. We saw how the higher level of abstraction provided by C can aid in code design and readability.