Alan Achtenberg

Lab 3 Hardware/Software

ECEN 449 Sec:505

Due: 10/3/2014

Introduction:

In Lab 3 we created a new hardware peripheral using the xps tools, and then we wrote a simple c program that wrote input into the peripheral before reading back the ouput, and then printing the value to the console.

Procedure:

First we set up a new project in xps just as we did in lab 2, with a SPLB communication block.

Then we went thru the process of creating a hardware peripheral. Our hardware that we created is a simple 32 bit unsigned multiplier. It has two input registers named slv\_reg0 and slv\_reg1. It has one output register slv\_reg2.

After adding the hardware peripheral to our project we then went through the process of implementing the testing logic with our c program. Our simple c program runs through a testing loop of 16 values, that writes to the first two registers some test values and then reads the output. After reading the output, the program then prints the value to the console

Results:

Verilog and C Source Files

Conclusion:

It is easy to see from this lab how you can greatly increase the speed and efficiency of a program by creating custom hardware to perform time consuming tasks. Use of the hardware can also simplify a program much in the way calling a function does. You only see the input and output in your program main instead of all of the implementation logic.

Questions:

5. [2 points.] The output of the terminal (kermit).

6. [3 points.] Answers to the following questions:

(a) What is the purpose of the tmp reg from the Verilog code provided in lab, and what happens if

this register is removed from the code?

The tmp reg serves as an intermediate storage for the output of the multiply. Since I/O operations tend to take to much time, if we did all the I/O and the multiply in the same step we would not be able to finish the multiply in one clock cycle. If it is removed we will have a delay on being able to read the result of the multiply possibly causing errors.

(b) What values of ‘slv reg0’ and ‘slv reg1’ would produce incorrect results from the multiplication

block? What is the name commonly assigned to this type of computation error, and how would

you correct this? Provide a Verilog example and explain what you would change during the

creation of the corrected peripheral.

Values greater that are greater than 32 bits when multiplied, ie. 0xFFFFF \* 0xFFFFF, this commonly known as a overflow error. To correct this error we would simply increase the size of our register if possible or we would calculate our multiply in separate chunks and output the result in multiple registers.

Verilog example: redefine [31:0]slv\_reg2 to [63:0] slv\_reg2;