Laboratory Exercise 4

Due March 4, 2016.

The starting point for this lab is the MIPS CPU example we have been covering in class. The workspace includes the CPU, memory and simple test program.

In this lab, you are augmenting the MIPS model to include the instructions **mult**, **multu**, **div**, **divu**. The instruction encodings are shown in Table 9.7. Visit the following link to get a description on how these instructions operate http://en.wikibooks.org/wiki/MIPS_Assembly/Arithmetic_Instructions. In the MIPS, the result of multiplications and divisions are stored in a special 64-bit register called the result register. To retrieve the values from the result register, you will need to also implement the **mfhi** and **mflo** instructions.

Part 1.
Add the instructions and the associated registers. In order to test the instructions, create a simple routine that takes the weighted average of a collection of values. Assume the values are stored in memory beginning at location X"00000100", the weights are stored in memory beginning at location X"00000200", and the size of the collection is stored in location X"000000FF". Use an Aldec simulation to confirm the operation of the new instructions.

Part 2.
Modify the MIPS model so that it can be synthesized in Quartus. In your report describe the how the original model needed to be modeled in order to be synthesizable. Structure your model so that it can be tested by either a push-button clock or a 1 MHz clock generated by a PLL.

Part 3.
Use TimeQuest to determine the fastest allowable clock and modify the phased-locked loop component to assure correct operation. Compile and test your model on the DE2 board. Use the SignalTap logic analyzer to confirm the correct operation of your test program. Your Aldec simulation should include the PC, CPU state, memory address/data, the RAM and register files. For the SignalTap analysis, your encodings should match the encodings for objects in the Aldec simulation (e.g. opcode and alu function2).

Part 4.
Create a new architecture for the entity from Part 3. Modify the phase-locked loop to make the clock period 15 ns. Modify the state machine to include wait states for the multiplication and division operations. Using the same program in Part 1, confirm the MIPS CPU operates properly. In Quartus, apply the Time Quest multicycle operation to appropriate paths to remove Time Quest setup and hold time violation warnings. As with Part 2, use the SignalTap Logic analyzer to confirm the operation.

In your lab report, please include the following:

- summarize any design that you had to do to complete each part.
- include a commented listing of the assembly language program that includes the location of each instruction in memory and the machine code.
- include Aldec simulations showing the verification of your models.
- provide a comparison of the simulation results from Aldec with the results from the SignalTap

logic analyzer.

In your workspaces, make sure your SOF files are easy to find and that you have a different SOF file for each part.