Practice Assignment – Single-Cycle MIPS

Senyo

January 17, 2020

The goals of this assignment are to

- To add instructions to a VHDL implmentation of a subset of the MIPS ISA.
- Familiarize yourself with speedy Simulation in GHDL and advance debugging techniques in Cocotb.
- Familiarize yourself with HDLs (in this case VHDL) and prepare you for more complex, cycle-level data path implementations, namely the multicycle and pipelined datapath implementations.

Installing the Tools

Go ahead and read through the following documents:

- `Getting_Started_With_GHDL_and_Simulation.pdf`
- `A_Deeper_Look_At_GHDL.pdf`
- `Speedy_Debugging_With_Cocotb.pdf`

Assignment: Understanding the Model

To familiarize yourself with the model, go through the following steps.

1. How large is instruction memory? What are the contents of the memory word at address 0x8?
2. How many words of data memory are there? What is the byte contents of location 0x11?
3. How many bits is the sign extended offset?
4. Can you tell from the code you are given whether the data memory operates in little endian or big endian format? How can you tell (not by the documentation)? What changes to the code would you make to have the data memory operate in big endian format?
5. What registers **(rs, rt, rd)** is the 5-bit signal **write_register_address_0 referring to**?
6. Is the offset in the beq instruction relative to PC or PC+4
7. The model has a hard-coded test program and the model as written, only supports the instructions shown in the test program (add, lw, beq). Write and encode another short test program to load the contents from different memory locations and compute their sum in a register. Since we have such a small memory, use **$0** for **rs**. You can encode instructions by using Qtspim and looking at the text segment. In this case note that branches and jumps will not have the correct encoded value of the offset or absolute address, when used in the simulation model! You need to correct this manually (this has the added benefit of making it clear how these instructions work). At the end of the program branch, back to the beginning (as shown in the test program) using **beq** to create an infinite loop. Note that you have to encode the **beq** instruction with the offset from PC+4! Edit the memory array to place data values in memory (initialization with file I/O is not implemented here). Execute your simulation,

view the trace, and make sure the trace indicates that this program works. If you have assembled the program correctly it should run with no problems.

8. Compile and execute the simulation. From the trace identify the following, demonstrating to yourself that your program executes correctly
    1. Encoded instructions
    2. The contents of the registers (click on the '+' sign next to array signals to expand them).
    3. The results of the execution of each instruction.

Assignment: Adding Instructions

This is the main part of the assignment and should be submitted. Extend the MIPS ISA simulation to add the following instructions: **sub, and, or, ori, andi, j, jr, jal, sw, slt, and lui**. **This component of the assignment must be done individually! You may verbally discuss solutions but coding must be done individually and no code can be shared or viewed**. Most of the effort will be in understanding what needs to be done (item 1 below), understanding the model, and debugging (VHDL & GHDL).

1. Keep in mind that signals are very different from variables. Each signal should have only one driver.
2. For the most part, you should be able to modify and add to the existing code segments.
3. Compile and test by creating test cases. It is highly recommended that you create unit tests – one test case for each instruction. For example, use lw followed by beq encoded to branch back to the lw instruction as a unit test to test the lw instruction.

Grading Guidelines

The purpose of this assignment is to familiarize yourself with VHDL.

However, if you are able to complete it and submit a working version, you will get **40 points** extra credit added towards your Lab grade.