

COMP3211/COMP9211 Computer Architecture

Lab 2 (Week 3)

Single Cycle Processor Core

Goals:

1. Study how to model a single cycle processor core using VHDL;
2. Study how to simulate and verify the processor core.

Note:

- This lab is designed to prepare you for the project work started in week 3.
- Please refer to lab 1 on how to use ISE and Modelsim. If you have no prior experience with using ISE, ModelSim and VHDL, please refer to related tutorials available on the course website.

Task1: Build and Simulate A Single Cycle Processor Core Model

Please follow the instructions for this task.

1. Copy the ~cs3211/public_html/refs/models/single_cycle_core.zip file to a suitable Project directory in your CSE home directory. This is the model you will be studying in this lab and may be used as the basis for your Project work.
2. Start a vmware session as explained in lab1. In order to obtain reasonable performance, copy the single cycle model from your CSE home to your My Documents folder. (Please note that any changes made to these files that you wish to keep must be copied back to your CSE home before exiting vmware. Failure to do so will result in loss of work because all user files are deleted from the vmware image when you exit.)
3. Unpack the zip file by extracting all files to My Documents\single_cycle_core, say. Start the ISE Project Navigator by following the appropriate links from the Start menu.
4. Open the single_cycle_core.ise Project file in My Documents\single_cycle_core folder. This will load all the files into the Xilinx development environment. You can navigate the code by clicking on the VHDL files listed in the Sources in Project pane. Get a feel for the structure of the code by taking a look around. The code should be fairly easy to understand. If you have trouble understanding the code, please refer to the VHDL references.
5. Next, simulate the design. A testbench has already been prepared for you. You can select the prepared testbench signals by clicking on the file called single_cycle_core_testbench. The Processes for Source pane contents change with the type of source file selected. When the testbench is selected, one of the processes that becomes available is to Simulate Behavioural Model. Clicking on this entry will start a simulation session in ModelSim. The result of the simulation (the waveform seen when you click on the wave tab in the ModelSim window) doesn't appear to be interesting because no interesting (internal processor) signals have been selected for display. The Quick Start Tutorial in the ModelSim tool explains how you can select signals that you want to observe. However,

we have prepared a macro that loads all signals of interest. Enter the command “do single_cycle_core_waveform.do” at the VSIM> prompt. This loads the macro. Next, you will need to restart the simulation. Enter the command “restart” and confirm your request. Finally, rerun the simulation by entering the command “run -all”. The Quick Start Tutorial describes shortcuts for these commands. If you now reselect the waveform pane, you will be able to observe a complete simulation of the program stored in instruction memory.

Task 2: Study and Verify the Core Model

Have a look through the code in the My Documents\single_cycle_core folder and answer the following questions:

1. How is the register file modeled? What is the size of the register file?
2. How is the instruction memory described? What are the differences between the instruction memory model and the data memory model?
3. How is the function of the control unit described? How many control signals are there? And how are the control signals used?
4. Given the instruction format specified at the beginning of the single_cycle_core.vhdl file, please verify the instructions stored in the instruction memory. What does the program do? Are you able to confirm the processor executes this program correctly?
5. How would you change the data the program operates on?
6. Tougher: With reference to the code and the block diagram in single_cycle_core.pdf, can you figure out how to modify the processor and machine code to implement the following program code?

```
if (i < j)
    i++;
```