Lab 3: A Single-Cycle Calculator in VHDL

This lab is due Monday April 30th. Your lab report and source code must be submitted by **11:59PM before the midnight**. You are also required to do a 5-minutes presentation of your calculator design in the class of **Apr. 16. All presentation slides are due by Apr. 16 before class.** The late policy applies to this lab project.

This lab is to be done in teams. Get started early! The required format for lab reports can be found on the resource page.

Objective: The objective of this laboratory is to implement a calculator based on your calculator ISA design. Moreover, you should develop a VHDL testbench to test the functionality and verify the correctness of your implementation with VHDL simulation.

Specification of the calculator: The calculator should have 9 input ports (think the inputs as "buttons" or "switches") in total. You should implement the 8-bit ISA you designed in the first lab on the board. The 8-bit instruction words should be mapped to 8 of the 9 inputs, and you should use one of the inputs for clock signal. Remember that your ISA is required to have a display instruction. For that instruction, the content should be displayed on screen. **Note: the display instruction is the ONLY operation that outputs anything on the screen. No hack printout from your calculator internal is allowed.**

All numbers are displayed **in decimal format** with 4-digit, even though the operands are read from the input ports as binary numbers. For example, if the content to display is "0101", the 4-digit display should read as "[empty] [empty] [empty] 5".

You should adapt your implementation of the adder/subtractor from the previous lab in this work. You can make necessary and reasonable changes to your code, but the base line is that you cannot directly use the "+" and "-" operators in VHDL to do the adding or the subtracting. The subtract operation might produce negative results, and you should display the result as negative. For example, if the two inputs are "0010" and "1111", when the "add" function is invoked, the display should read "[empty] [empty]17". For the same inputs, when the "sub" function is invoked, the display should read "[empty] – 1 3".

Circuit Design: You should first design a single-cycle datapath+controller circuit for your calculator. **IMPORTANT: The diagram should illustrate the circuit at RTL level, and contain sufficient details about main gate-level components such as registers, mux and demux.** In particular, the diagram should clearly label signal connections and datapath width.

You design, as well as implementation, should have a clear distinction between the datapath and the controller. In other words, the controller should be represented as a separate component in your design, and in its VHDL implementation, as a separate *entity*.

You should have the diagram ready by Apr. 16 for the presentation. On the presentation day, all teams are expected to showcase their calculator design in a 5-minute presentation. You are expected to briefly describe your circuit schema, on blackboard or with slides. If you prefer, you can also present your design earlier (making an appointment with the instructor for that).

The design presentation and the final circuit design submitted in your lab report will together account for 40% of your grades in this lab project.

Testbench: You should write a VHDL testbench that test ***ALL*** instructions that your calculator implements. The testbench should drive the input ports, and we should be able tell whether your implementation is correct or not by examining what the display instructions display on screen. Remember that you ***CANNOT*** use hack printout to prove correctness.

Use synthesizable features of VHDL (IMPORTANT)

You should only use the synthesizable features of VHDL. If your code doesn't conform to the following requirements, the score for that part of code will be zero. Note that the following requirements only apply to the implementation part. You can use ANY features in your test benches.

VHDL feature requirements:

(1) No variables are allowed;
(2) Up to one "wait" statement OR a sensitivity list is allowed in a process.
(3) A process should wait on (using a "wait" statement or the sensitivity list) either (1) ALL signals appearing on the right-hand-side in the process body; or (2) the clock signal or with the optional reset signal.
(4) No "wait for" statements or time expressions in signal assignment statements.

What to Turn In

For this lab project, submit the gate-level RTL circuit schematic design that shows the design of your single cycle datapath. The diagram can be a drawing from software or scanned image of your drawing on paper or a photo of your drawing.

Also turn in all of your source code and the code that tests your implementations. For each problem, all the source files and the project configuration files must be packed as a single zip file. The goal is that I can unpack your zip file and type a single command, e.g., "make" to compile your whole project. In your project report, please also describe your testing methodology. In particular, describe the instruction sequence that you use to test, why the sequence will prove the correctness, and what is the expected printout if everything runs correctly.

Peer evaluation: Please email your peer evaluation (guideline is posted online) of your teammates, including scores and justification, directly to the TA.