Name: _____

Lab for ITSC 3181, Introduction to Computer Architecture, Spring 2023

Lab #08: Logic design using Digital (https://github.com/hneemann/Digital): 32/16/8-bit ALU that can perform add, sub, and, or, nor and **slt** operations. **Due 03/22 and count for 1% of the final accumulated percentage grade.**

From Lab 07 to Lab 12, you will build a RISC-V CPU to support an essential set of instructions including at least add, and, or, sub, lw, sw, and beq instructions. While you are encouraged to design a 32-bit CPU (each register stores a 32-bit word, and the ALU operates on 32-bit data), you are ok to design 8-bit or 16-bit CPU (each register stores a 8-/16-bit data, and the ALU operates on 8/16 bits data). Designing 8-bit/16-bit is much easier comparing with designing 32-bit, but with same design principle. But regardless of the bitwidth you choose, each instruction is still encoded as a 32-bit word and the address for accessing data from memory are also 32-bit address. For this lab, you will create an ALU of the bitwidth of your choice.

Tasks:

1. (20%) Review the lecture slides (https://passlab.github.io/ITSC3181/notes/AppendixA_LogicDesignBasics.pdf#page=74) to understand how to create a 1-bit logic unit (from slide 76), 1-bit half-adder (slide 79) and 1-bit full adder (slide 80-81), and then design a circuit for each of the three units in Digital and save them in three files. For each design, simulate the unit by giving the input based on the truth table in the slides and make sure the output from the designed unit are correct according to the truth table.

2. (30%) Review lecture slides for how to create a 1-bit ALU and how to extend it to support **add**, **sub**, **and**, **or**, **nor** and **slt** operations (from slide 82 to 86). Then review the Digital Documentation section 1.4 about how to create circuit that uses other subcircuits. Following the document instruction, design and simulate a 1-bit ALU that supports the above 6 operations based on the design we study during the class. The final circuit should be very similar as the 1-bit ALU in slide #83 except no wire or output for Overflow. After complete the design, make sure inputs are labeled exactly as the slide as a, b, CarryIn, Ainvert, Binvert, Less, and Operation, and the outputs exactly are CarryOut, Set and Result.

3. (50%) Review the lecture slides for creating 32-bit and 64-bit ALU (slide 88, 89, and 91) and then design an ALU circuit of the bitwidth (32-, 16-, or 8-bit) of your choice. This bitwidth will be the one you use for designing the register, memory and CPU in the following labs. Name it as “32-bit ALU” (or 16-bit ALU, or 8-bit ALU). Your design should correctly perform SLT operation (slide 84), and check zero (slide 89). The designed circuit should be very similar to the one in slide 91 except that no overflow circuit is needed and it is the 32-/16-/8- bitwidth you select. In this design, reuse the 1-bit ALU you designed in the last task. There are several important things for your design (using 8-bit ALU as example):
    a. Layout properly at the beginning so wiring will be easier and well organized. The organization in slide 91 is a good layout.
    b. For ALU0, Binvert needs to connect to CarryIn, which is for the sub operation
    c. ALU7's set output needs to connect to the ALU0 Less Input
    d. Constant 0 (Components→Wire→Constant Value) needs to be used for the Less input for ALU1 to ALU7
    e. Input A, B and Output Result are all 8-bits, Operation input is 2 bit.

f. Need a Splitter (Components→Wire→Spliter/Merger) to split 8-bit input (A, B) into 8 1-bit inputs for each 1-bit ALU. Also need a Merger to merge 8 1-bit Result outputs of all the ALUs into one 8-bit Result output. Check the Document about how to use splitter and merger.

Important: when designing the circuit, please make sure you follow these rules for adding, changing components, input/output and wires. These rules are applied to all the design lab tasks of the course.

1) **Each input and output of a design MUST be properly and meaningfully labeled.**
2) **Each component, input and output should be correctly configured in terms of its bitwidth and signal control width.**
3) **Keep wires and components organized and layed out according to the circuit schematics rules, a) Inputs on the left (or top), b) Outputs on right (or bottom), c) gates flow from left to right, d) Straight (not angled) wires are best, e) Wires always connect at a T junction (only 90-degree turn or connection), f) A dot where wires cross indicates a connection between the wires, and g) Wires crossing *without* a dot make no connection. While you may not need to follow all those rules for creating correct and small circuit, it is very important when for creating complex circuit. So, make sure you properly organize the components and wires, make them structured and look good. That will help reduce errors.**
4) **For drawing straight wires using mouse, make one turn per each draw. You should not use one draw to make a connection that needs to have two or more 90-degree turns, which would create angled wires. For those wires, you have to make a wire that has a 90-degree turn, and then connect it with another wire that also make 90 turn, and so on. Try to minimize the turns as much as possible. If two wires have to be crossed, but should not be connected, make sure no dot is marked on where the wire is crossed.**

Your grade will be based on both the correctness of your design and the organization of the design, 60% and 40% respectively.

Task	1 (20%)	2 (30%)	3 (50%)	Total
Correctness (components, input/output, bitwidth, connection, width of control, etc), 60%				
Organization (I/O labeled and positioned correctly, all straight wires and T junction, dot used correctly, readability etc), 40%				
Total				

After finishing your design, put your solutions in a file and submit it from canvas. For the design, you can save the design to an image and copy to file, or can screenshot your design and save to file.