COSE222: Computer Architecture
Design Lab #2

Due: May 09, 2020 (Sunday) 11:59pm on Blackboard

Total score: 20

In Chapter 4 of this course we are exploring how to design the simple RISC-V processor. The baseline integer ISA of RISC-V is RV32I, however in this course we learn about 64-bit computers thus the instructions covered by the textbook is based on RV64I. The textbook also covers the part of integer instructions (just 7 instructions: `ld, sd, add, sub, or, and, beq`) for the processor design. The design labs provided for COSE222 will follow the processor design steps described in the textbook. For this design lab assignment, you will be requested to design datapath elements (instruction memory, register file, ALU, and data memory) of the processor.

During the class we explained about the typical RTL design process. For a single datapath element design, you will be provided the information regarding the interfaces (inputs and outputs) and functional operations of the corresponding hardware unit. You should build the RTL design based on such information and then verify your design with testbenches. If the verification is done successfully, the design of the datapath element is complete.

For this design lab, you should design the following RTL codes using *SystemVerilog*.

1. Instruction memory (imem.sv)

Instruction memory stores 32-bit RISC-V instruction. The instruction memory is accessed by the address pointing instructions (i.e. program counters). Once the program counter is given, the instruction memory provides the corresponding instructions.

In this design you are requested to design the instruction memory that includes **1024 entries** of 32-bit width. In the textbook the width of program counter is 64-bit since we are exploring 64-bit computer systems. However, the designed instruction memory will receive 10-bit address since the memory have 1024 entries.

The required input/output ports are already provided in the incomplete code of "**imem.sv**". Please complete the instruction memory design.

2. Register file (regfile.sv)

Register file means a group of registers. As a single RISC-V instruction include two source registers and one destination register at most, the register file need to provide interfaces for these requirements. The register file receives two register numbers for the source registers and one register number for the destination register. Based on the input register numbers, the data can be read from and written to the register file.

In this design lab you are requested to design the RISC-V register file. In the textbook the register file does not receive a clock signal. However if so, the register file can generate an infinite loop. Think about the case of "`add x1, x1, x1`" – the `x1` is repeatedly updated by new doubled value by `x1`. In order to avoid this undesired situation, we will make the write to the register file is triggered by the rising edge of the clock signal.

The required input/output ports and the detailed descriptions are also provided in the incomplete "**regfile.sv**". Please complete the design.

3. ALU (alu.sv)

ALU receives two operand data and generates the result data by performing the specified operations using the input operands. We will design the ALU as described in the textbook, thus the ALU will support only *add*, *subtract*, *bitwise or*, and *bitwise and* operations. The operations of ALU is selected by the ALU control signal. The interface of the ALU is already described in the incomplete ALU design file "**alu.sv**", Please complete the ALU design.

4. Data memory (dmem.sv)

Two memory reference instructions (`ld` and `sd`) accesses the data memory. The data memory design is similar to the instruction memory design, but the data memory receives two control signals: `MemRead` and `MemWrite`. You are requested to design the data memory which has **1024 entries** of 64-bit width.

In the textbook the data memory does not receive a clock signal. However, without the clock signals we need to implement the data memory using latches, which must be avoided in usual RTL designs. Latches can be exploited for special-purpose blocks. However, **if latches are generated after synthesis, it usually means your design includes critical bugs**. Hence, in this design lab you are requested to design the data memory working with the clock signal.

Another design issue of the data memory is the misaligned memory accesses supported by RISC-V ISA. Even though RISC-V supports the misaligned memory accesses, the data memory for this design lab will support only aligned accesses for double-word data in order to simplify the hardware structure. The detailed information is provided in "**dmem.sv**". Please complete the data memory design.

5. What to do

(a) Complete the design of the datapath elements (imem.sv, regfile.sv, alu.sv, and dmem.sv).

(b) Verify your design with testbenches. You should make the corresponding testbench file for each design. Set the clock frequency as 100 MHz. (Please see the provided testbench template "**tb_.sv**")

(c) Compress your completed design files (imem.sv, regfile.sv, alu.sv, and dmem.sv) and testbenches for datapath designs (tb_imem.sv, tb_regfile.sv, tb_alu.sv, and tb_dmem.sv) in **one zip file**. You must name your zip file as "`FirstName_LastName.zip`". (e.g. `Gildong_Hong.zip` for Gildong Hong) If your submission file does not meet this rule, we will reduce 1 point from your score. ☹