See other files at:
lab2/lab2.srcs/sources_1/new/light_controller.sv
lab2/lab2.srcs/sources_1/new/register_file.sv
lab2/lab2.srcs/sources_1/new/control_unit.sv
lab2/lab2.srcs/sim_1/new/light_controller_tb.sv
lab2/lab2.srcs/sim_1/new/register_file_tb.sv
lab2/lab2.srcs/sim_1/new/control_unit_tb.sv
Design Exercises submitted separately

Github Repo:
https://github.com/Nanat1/cs391r1

Lab 2

Problem 2.1 & 2.2

I started with the given definition of the in/out wires, and then filled in actions with different scenarios as mentioned in the PDF. With every rise of clk from low to high, I separated the scene into three different conditions: if the button is not pressed, when the button starts being pressed (where I initialize values), and when the button continues to be pressed.

With the testbench, I just initialize the button to false and press the button for 10 clock cycles, unpress, and repeat again so as to see the light_state start from the beginning, not from where it left off.

I have recreated files several times, as Vivado is not very flexible with renaming files and moving them. I have accidentally created a Verilog file instead of System Verilog, and that copy may still remain in my simulation sources; I only found that out when some syntax wouldn't render.

I found that in design sources, if I put "wire <variable>", I will fail assigning anything to the variable in an "always…" statement. I ended up using reg before each variable with the professor's permission, yet I suppose next time I will just assign a local variable to them outside of always, hope that works.

Problem 2.3

I referred back to the lecture PPTs on regfiles, and it is very useful. I ended up using some variable names from there, so that things may be more standardized and clearer.

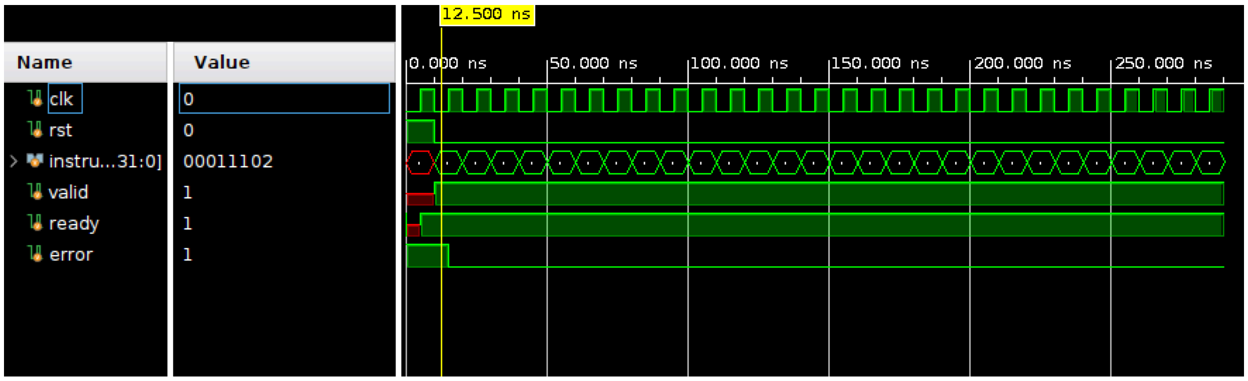
I wrote the testbench to try to write with we set to false and print the result of the 16 registers with each read channel in charge of 8, and then write with we set to true and print all, and then write to another register and print all. All went as expected.

I also find it extra helpful to use the Message tab to find error messages and figure out debugging. The messages I got from pop-up windows are not helpful at all, but the Message tab gave much more specific instructions. Also, remember to save the file often. Vivado does not update and save automatically.

Problem 2.4

I connected the control unit with the other modules, alu I copied from lab1 and register_files before I did the logic (I previously did some logic without connecting the modules, and they conflicted and overlapped the functionalities of the modules).

I test with first zeroing out reg4, then assigning b…00010101010 to reg0, b…00001010101 to reg1, then run every Type A instruction on the two registers and storing the result to reg2; I have then set imm as 5 (b…0000101) in the instructions and run every Type B instruction on reg0 and immediate value 5.



There is some confusion I have over the control unit; if I am to be picky, the words are not the clearest to describe what is wanted. It is especially hard to figure out in what order my orders will be completed, because we are not referencing another module by calls, but by connecting wires.