CPRE 381 Lab 4
Sequential Logic – Register File

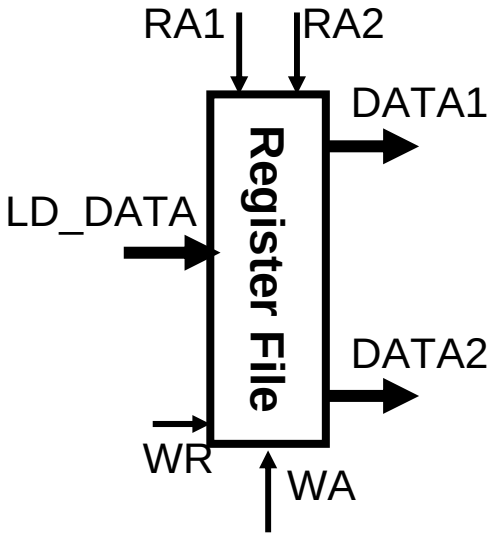
In this lab you will construct a design for a register file. Registers are sequential logic devices that store values for later retrieval. The register file will require both sequential logic as well as combinational logic devices implemented in Lab 1.

Part 1 – A simple register

1. Implement a verilog module for a single bit register that is positive-edge-triggered. There should be a data input (d), a clock input (clk), a write-enable input (we) and a data output (o).
2. Implement a verilog module for a single bit register that is negative-edge-triggered.
3. Construct a module that has one instance of each register with the inputs tied together and separate outputs.
4. Simulate the combined module showing the difference between the two types of edge-triggering.

Part 2 – A full 16-bit register

1. Implement a design for a positive-edge-triggered 16-bit register. Please make sure your 16-bit inputs and outputs are 16-bit buses and not individual 1-bit signals.
2. Simulate the 16-bit register to verify it is working correctly.



Part 3 – The full register file

1. Implement a register file in verilog with 16 16-bit registers.
    a. There should be one 16-bit data input (write or destination register).
    b. There should be two 16-bit data outputs (read or operand registers).
    c. There should be three address inputs (how many bits each?), one for a write register, and two for read registers.
    d. There should be a clock input and a write enable input.
2. Perform a simulation that shows all 16 registers can be written to and subsequently read on either operand output (don't set both operand addresses to the same register at the same time).

Final check: Can all of the registers in the log file be accounted for in the register file design?