Design of the Register Files & Integration with the ALU Datapath

ECE/CS 3710 - Computer Design Lab
Lab 2 - Due Dates:
RTL Design and Validation demo: Tuesday, Sept. 22
Hardware Demo & Project Reports: Thursday, Sept 24
Both deadlines are hard deadlines. September 24 onwards, we will begin the development of memory interfaces and control logic.

In the previous lab, you have designed, synthesized and tested the ALU. As the next step, we have to design the register-file and integrate it with our ALU to facilitate "operand reads" and "result write-backs" to and from the ALU, respectively.

CR16 has 16 general purpose registers (each 16-bit), and a few special purpose registers. Clearly, your regfile has to have 16 16-bit registers. In general, we have to address the following issues:

- How many read and write ports should your regfile have? What control signals will be needed to control read and write operations.
- Would you prefer a MUX or a TRI-BUF interface? Keep in mind that modern FPGA synthesis tools will not generate tri-state buffers in the logic, and will convert tri-state buffers into MUXes. *I strongly recommend that you guys design a MUX-based architecture.*
- How would you organize the regfile to interface with the ALU? Think about the data-path bus interface between the regfile and the ALU.
- How will you integrate the ALU Flags with the regfile? Are the flags a separate set of (processor status) registers, or one of the registers in the regfile is dedicated to work as a flag register?
- How will you design a TestBench that will perform a sequence of reads and writes to and from the regfile, via the ALU?

The above are basic issues that all of you will have to resolve. Later on, if you have to modify your design to suit your particular application, you can build extra logic/memory around it.

I will give you some guidelines on the design of register files and the various control signals. Try to organize your design as structurally as possible. Note, also, that now we are going to be adding **clock signals, register read-enables, as well as reset signals**. My advice would be: before you begin coding, draw a block diagram of your design. Analyze the execution of every baseline instruction-type, and see how your organization supports the data-transfer to- and from- the ALU. **Keep track of control signals, document them properly in the code, as well as in your reports. These will be generated by the CPU control** (which, of course, is going to be the subject of the subsequent lab assignments). A generic block-diagram/hierarchy organization that I will discuss in class is depicted in Fig. 1.

Your first objective should be to integrate the entire datapath and simulate it extensively. Then see if it can be synthesized without any problems, and test it in hardware (download it on the FPGA). This is the most critical component of the CPU. If this doesn't work (ALU hardware operations and proper data-transfer to- and from-register files) then your computer won't be doing anything at all.

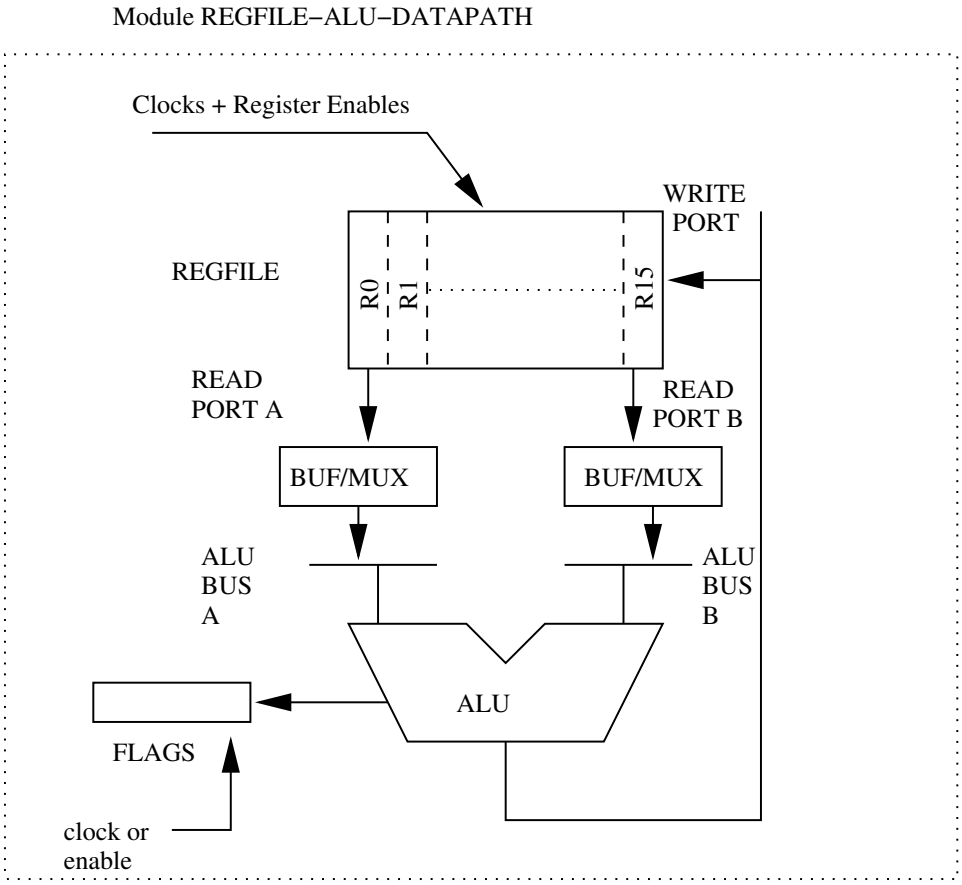

Fig. 1. Regfile + Alu Datapath Organization

Your design should be coded and validated by Sept 22. After final clean-up, by Sept 24, you should demonstrate a final working regfile+ALU combine on the FPGA.

In the previous lab, I did not ask you to write a report. So you will write a combined report on the ALU design and synthesis, along with regfile design and datapath integration. Submit a 7-8 page report that describes:

- A list of instructions/opcodes that have been implemented so far, and what has been postponed until the design of the DECODE machine (e.g., some of you may have postponed "immediate operations" for later).
- regfile design (ports, mux/buf-based, the basic block diagram, control signals, etc.)
- overall organization, data-transfer between the ALU + Regfiles
- Synthesis stats - Area (number of LUTs/ALMs, FFs, BUFs, etc. used), Delay (longest paths, etc.)

Once this lab is successfully completed, we will address: i) memory organization, ii) Block-ram and S/Dram options available on board, iii) and how they will be integrated to develop a fetch-decode framework; i.e., how to design the CPU controls.

How to test Regfile + ALU combine? Since the regfiles are sequential logic, the simulation and validation is not as simple as in the case of combinational logic. We will have to test sequences of instructions (realistically, a set of programs) to ensure that operands are read from, and results are written to, the registers correctly, irrespective of the sequence of instructions.

Perhaps a good approach is to write a sequence of ADD/SUB/CMP/SHIFT operations and check the final result. In this case, you can "preset" the regfile with certain integer values (or make use of immediates). Your testbench should then act as a simplified version of your decoder and for each instruction, give the respective OPCODE and

CNTL signals to your ALU datapath. *For example, you could preset register R0 and R1 with some constant values, and write a simple code that computes their Fibonacci sequence and stores the final result after 15 additions into R15.* This is just an example, you can come up with any interesting sequence of computations and see that the paper-and-pencil result is the same as done by your ALU, and the correct result stored in regfiles!

Try not to limit your test program to just one testbench. Prepare 3-4 testbenches. One test bench could test unsigned arithmetic algorithm (say, a Fibonacci sequence), the other could do something similar with signed arithmetic, and one testbench could be a good mix of logical and arithmetic operations. A good testbench to test shift operations too is to compute multiplication of two 8-bit numbers into a 16-bit integer. I will talk about this in class, and I will upload a description of the test-setup, along with skeleton Verilog code for regfile design, on the class website later in the week.

Another Important Note - Looking Ahead: The next step in this project is to start thinking about applications/extensions to the basic design. What application do you want to build? How do you think you will go about designing it? What time-lines do you envision? Challenges? Bottlenecks? In early October, we will have a presentation/review session - we will talk about it in the lab and start preparing for it. Your application may demand how much memory will you require, so it might be a good time to start thinking about it.