CMSC 411—Project—Part a

For the first phase of the term project, you will be constructing a MIPS-style register file from discrete low-level components like AND and OR gates and wires, plus some pre-built higher-level components like multiplexers, demultiplexers and flip-flops. Obviously, you should not use things like completely-built register files from any libraries; else, what would be the point of the project? The goal is to show you how straightforward the design of such seemingly complex objects are, giving you all a sense of empowerment and confidence ☺.

We gave a demo earlier in class of the Logisim application that allows you to do simple to moderately complex circuit design. You should already have downloaded the Logisim package onto your own computer (download links and other information available at:

`http://www.cburch.com/logisim/download.html`

I am assuming you have already also begun playing with it to familiarize yourself with the controls and workflow. For this project phase, you are going to use create a complete circuit from the following components (you do not have to use all of them; however, you really should not have to use anything from outside this set):

- Registers (Memory→Register in library)
  These are glorified collections-of-D flip-flops that have some additional circuitry to allow it to store, input and output multiple bits. You are free to use raw flip-flops (Memory→D Flip-Flop), but it would create a lot of extra busy work for you, so why bother? In Logisim, both flip-flops and registers have clock-in, enable and clear lines. The additional features of a flip-flop, like complemented output and preset input, are not things you should not need for this project.
- Multiplexers (Plexers→Multiplexer in library)
  These can switch multiple bits from one of N inputs, selected by the Select lines, to the output. So, with an 32-bit/8 input multiplexer, you can channel any one of the 8 32-bit-wide inputs to the one 32-bit-wide output, by feeding in the proper binary number on the Select input.
- Demultiplexers (Plexers→Demultiplexer in library)
  These can switch multiple bits from the input to one of the specific outputs, channeled by the Select lines. Inputs and outputs are analogous to multiplexers, except reversed, sort of.
- Decoders (Plexers→Decoder in library)
  These raise a single one of N output wires at any given time, specified by the “select” input. Note that by default one of the output lines is always on. This component has an “enable” line, but read the description of what the default behavior is when “enable” is off—it might ot be what you want.
- Pins (Wiring→Pin in library)
  These allow you to specify pads for manually inputting values to the circuit, later used as connection points for integrating your register file as components into larger circuits. The pins can be configured as input or output pins, in which case they allow you to manually

set input values, or see the current output state, respectively. Pins have configurable bit width.

- Clock (Wiring→Clock in library) – exactly what it says it is.
- Wiring: note that Logisim lets you create wire paths that are multiple bits wide, automatically setting the width based on the components connected. So, if you add a wire connection between a 1-bit input pin and a 32-bit register, Logisim will complain, but if you reconfigure the input pin to be 32 bits wide, Logisim will make the single wire connection a 32-bit data path (but it will still appear as a single wire line in the diagram).

The Assignment

You will be constructing a register file that functions exactly as the black box we've been using in our architecture diagrams for the last several weeks. However, it will be scaled down a bit: you only need to build an 8 x 32-bit register file, i.e., it should contain 8 individually selectable registers, each with a word size of 32 bits. The circuit should have two independent output buses, each able to read out one of the registers, possibly the same register being output on both. It should have an input write bus that is used to receive the data that should be written to some register, as well as a write enable line to indicate something should actually be written on the next clock pulse. It needs a way to specify what registers are being read out by each output bus, as well as what register should be written to on a write request. And it should be clocked.

In detail, the inputs to your circuit should be:

- A clock: used for writes; a value is written into a specified register at the next rising edge of the clock pulse, only if writing is enabled. Note that the read outputs should be combinational circuits, reacting immediately (relatively speaking) to changes in control lines.
- WrEnable: write enable line: when (and only when) this line is high, indicates data should be written to a register at the next clock pulse.
- WrData: 32-bit-wide input for the data to be written to a register
- RegW: selector: a 3-bit-wide input specifying which register the "WrData" input data should be written to, iff WrEnable is high.
- RegA: selector: a 3-bit-wide input specifying which register's content should be presented on BusA output; RegA can be the same as RegB
- RegB: selector: a 3-bit-wide input specifying which register's content should be presented on BusB output; RegB can be the same as RegA

And the outputs from your circuit should be:

- BusA: a readout of the value stored in the register specified by RegA
- BusB: a readout of the value stored in the register specified by RegB

Clocking

A word on how and when the clock is used:

Your circuit should have an input pin for connecting a clock. To test your circuit, you should add a clock and connect its output to the Clock input pin. The clock signal should only need to be fed to your registers for write operations, i.e., when the WrEnable line is high. Note that Logisim's "register" component is already clocked and has an "enable" line. However, you cannot just wire the Clock and WrEnable inputs in parallel to all of your registers, because that would cause *every* register to write a new value, and you only want the user-selected (via RegW) register to be written. Hint: demultiplexer or decoder to the rescue!

By default, Logisim starts up with the layout pane opened to the "Untitled→main" circuit. You should create your register file as new circuit at the top level (just right-click on the "Untitled" folder in the left pane, then select "new circuit…"), and call it "RegFile". That way, you can more easily integrate it as a component in other circuits in later projects.