CSCB58 Lab 6: Register Files

1 Introduction

This week, we're working with larger-scale devices composed of multiple individual components. The goal of the lab, similar to Lab 3, is to become more comfortable with high-level design – *programming* with hardware devices, rather than mathematical specification using gates. In this lab, you'll also explore the use of control bits in devices, and the use of buses. A bus is simply a wire containing multiple bits.

PRELAB REPORT: This lab requires the submission of a prelab report to Quercus to **by the start of your lab section**. The deadline shown on Quercus may not apply to you. Your TA will access your prelab submission during the lab and ask you about it.

2 Multi-Bit (Parallel) Registers

A register is a device that stores a value. In reality, registers (and register files) are often created out of SRAM cells (we have not covered these), but for simplicity, you can imagine it being built out of D flip-flops. A 4-bit register stores 4 bits in "parallel" (you can imagine each D flip-flop handling one bit of the input and output). We will not be creating this component, and instead will be using the built-in `Register`, but if you feel motivated you can try building it from scratch for fun.

In addition to the clock input, registers take in a `write enable` (WE) input. It is an example of a "control" input – it determines whether or not the register is to accept input. When `write enable` is high, and the `clock` goes high, the register stores the input. If either is low, the register maintains its current state. (Recall that the flip-flop is an edge-triggered device, so the input value is stored on the positive edge of the clock.)

3 Design and Implement a 4-Register 4-Bit Register File

A single register can only hold a single value, and a computer will need more storage. (Most processors have 32 or more.) The *register file* that you will

build for the lab is an array of registers that allows one register to be read from (at any time) and one register to be written to (on the positive edge of the clock). It will be built using Logisim-Evolution's built in `Multiplexer`, `Demultiplexer` and `Register` blocks.

Your register file will contain 4 registers (referenced by the numbers 0 to 3). Your register file should accept a 4-bit data input and produce a 4-bit data output. It also requires a *clock*, *enable*, and *two* 2-bit *select* inputs. All four of the latter inputs are "control" inputs. The first *select* is a *read_select*: the value of the input indicates which of the four registers is the source of the register file's output. The second *select* is a *write_select*: the value of the input determines which of the four registers will accept input. Like a register, the *clock* and *enable* inputs determine if and when a value is to be stored.

Here are two questions to consider as you build your register file:

- All four registers will be providing output at the same time. How do you *choose one* to be the output for the whole register file? What bit-width must the devices that chooses have?
- Only one register should store the input value on the positive edge of the clock. Here are two options, **only one of which is correct**: you could use a 1-bit demux to send the enable bit to only one register, or you could use a 4-bit demux to send the input to only one register. *Hint: Remember that a demux sends the input value to the selected output line. The other output lines will have the value* 0.

PRELAB REPORT: For the prelab report, show the schematics for the register file using high level blocks (registers, muxes, etc). Be sure to indicate which wires and inputs are buses and their "width" (number of bits).

For the Logisim implementation of your circuit, be to use the multi-bit components instead of having 4 data wires for input / output. Refer to the Logisim-Evolution reference section 3 for more help, but the relevant properties are `Data Bits` and `Select Bits` to customize the components.

In your lab session, your TA will ask you questions on how you have designed your register files, and then you need to demonstrate the functionality of your circuit.

As always, if you have any questions on the lab not addressed by the video overview released at the end of the week, feel free to ask us on Piazza and we'll do out best to clarify any ambiguities.

4 Summary of TODOs

1. Before the lab, read through the lab handout. Complete and submit the **PRELAB REPORT** with the following information:
   - Your name and student number.
   - The circuit diagram of the register file.
2. In the lab, show the TA your design and convince them that it will work.
3. Once convinced the register file works, demonstrate it to your TA.

Evaluation (5 marks in total):

- 3 mark for the prelab report including the register design.
- 2 mark for working register file in Logisim-Evolution.

For the next and final Logisim-Evolution lab, we'll combine a register file with a simple instruction decoder and ALU, to build the datapath and control unit for a simple processor that you can program yourself!