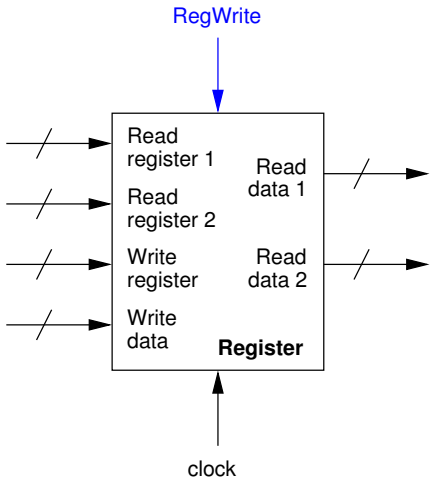
Lab 9: Building a Register File

Submission timestamps will be checked and enforced strictly by the CourseWeb; **late submissions will not be accepted**. Check the due date of this lab on the CourseWeb. Remember that, per the course syllabus, if you are not marked by your recitation instructor as having attended a recitation, your score will be cut in half.

A Register File

A register file is a subcircuit that lets the surrounding circuitry read from one or two registers, which are chosen dynamically. A register file also lets the surrounding circuitry optionally write to one register, which is also chosen dynamically. In this lab, you will build a register file that has two read ports and one write port. That means that your register file circuit should let outside circuitry read two registers at a time while also supporting writing to one register at a time. **You should support eight registers total. Each register must hold 32 bits of data**. Use Logisim's built-in register component.

The Figure below shows a symbolic representation of what your register file subcircuit should looks like. The slash indicates that an input/output contains more than one bit of data.



- **Read register 1** specifies which register should be read from. That register's data should be presented on the **Read data 1** output.
- **Read register 2** specifies which register should be read from. That register's data should be presented on the **Read data 2** output.
- **Write register** specifies which register should be written to.
- **Write data** is the data that will be written to the register specified by **Write register**. On a **falling** clock edge, the data will be written to the specified register.
- **RegWrite**, if true (1), will cause the **Write data** to be written to the register specified by **Write register**, otherwise, **Write data** will be ignored and no register will be written to.
- **Clock** conveys the clock signal.

Before starting, ask yourself the following questions:

1. How can we tell how many bits will be outputted on **Read data 1** and on **Read data 2**?
2. How can we tell how many bits specify **Read register 1** and **Read register 2**?
3. How can we tell how many bits specify **Write register**?
4. How can we tell how many bits specify **Write data**?
5. How can we tell how many bits specify **RegWrite**?
6. Why is **RegWrite** necessary?
7. Why does the circuit not have **RegRead** input?
8. Why does the register file have a clock input? Why doesn't it use its own clock instead of requireing a clock from outside circuitry?

To assist you, here are some tips about sizing different bit widths:

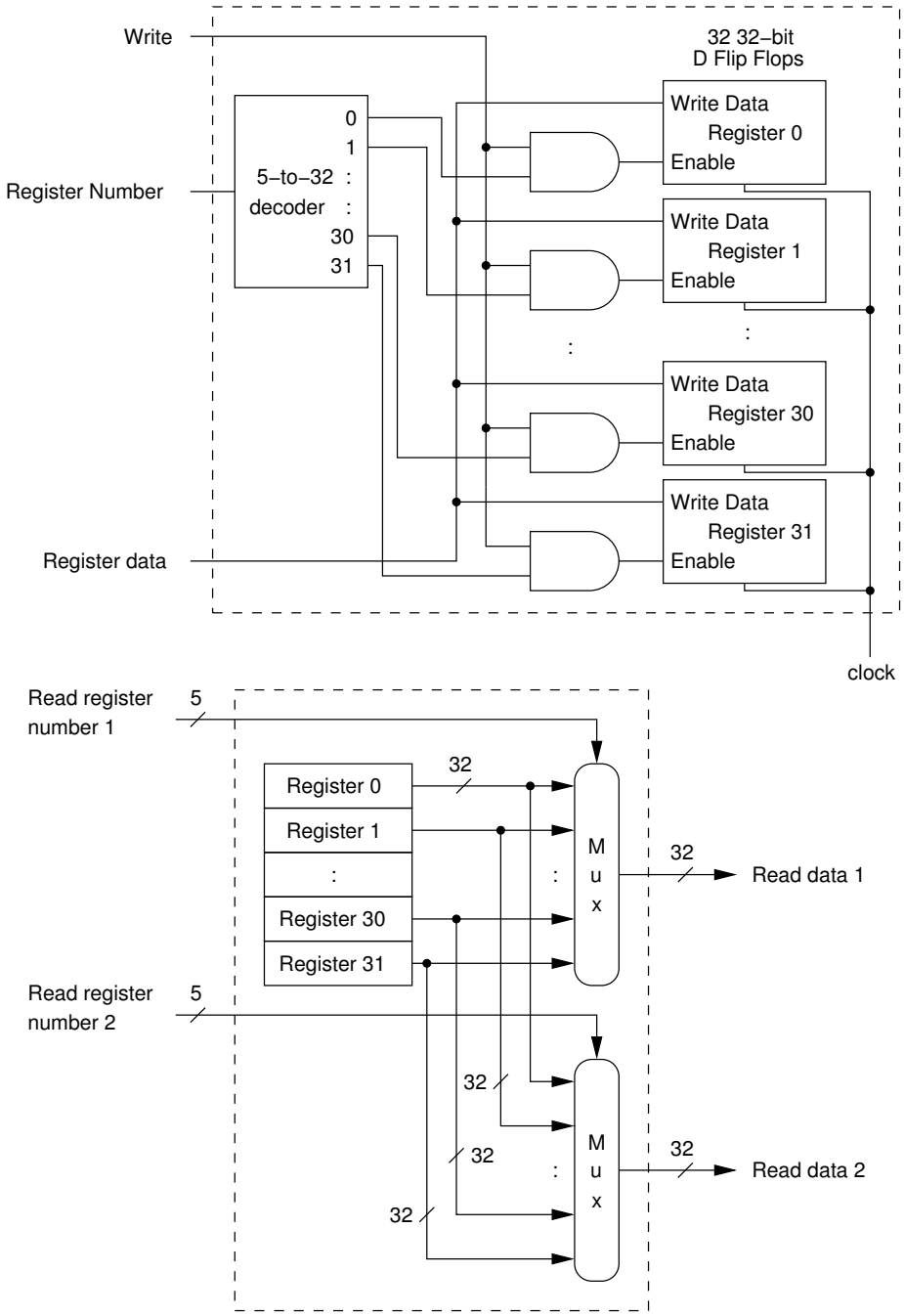
1. Since the registers are 32 bits wide, and since **Read data 1** will output the entire contents of a register, **Read data 1** will be a 32 bits wide. By the same reasoning, **Read data 2** will also be a 32 bits wide.
2. Since you need 8 registers, then 3 bits are needed to "name" each register. The register having the name 000 is the register 0, 001 is the register 1, 010 is the register 2, and so on. **Read register 1** and **Read register 2** are therefore each 3 bits wide.
3. **Write register** describes which register of the eight registers is being written to. Therefore, **Write register** is 3 bits wide.
4. **Write data** specifies the data that will be written into a register. Since the registers are 32 bits, **Write data** will be 32 bits.
5. **RegWrite** says whether or not a register write should actually be performed (yes or no). Therefore, **RegWrite** is 1 bit wide.

Here are some general tips to help you implement the register file:

- The data that should be written to a register must be able to be "given" to **any** of the 8 registers in the register file (i.e., your circuit must be able to write to any register).
- Registers can be made read-only (disabled) by setting one of their pins appropriately. This can let you control which register, if any, is written to. Take a look at the Library Reference for more information.
- Consider where you might need multiplexers.
- Recall that, given multiple inputs, a **multiplexer** lets you select from exactly one of them. Remember that the number of inputs from which a multiplexer chooses can be configured by changing its "select bits" property, and that the "data bits" property controls how many bits are on each of its inputs.
- A **decoder** has multliple output wires, but only one of those output wires is true at any given time. It is easy to control which output is true. Decoders are found under the plexers category.

You can poke a register and give it a value. This will let you give each register a different value to make sure that the circuitry reads from the correct register (according to **Read register 1** and **Read register 2**).

Diagrams below show circuit diagram of a register file discussed in class (32 32-bit registers file). Use these diagram to help you build your register file.



Submission

Submit the file `lab09.circ` via CourseWeb before the due date stated on the CourseWeb.