Homework 3

Due Apr 20 by 11:59am **Points** 100 **Submitting** a file upload **File Types** circ, txt, pdf, zip, tar, and gz
Available after Apr 10 at 6pm

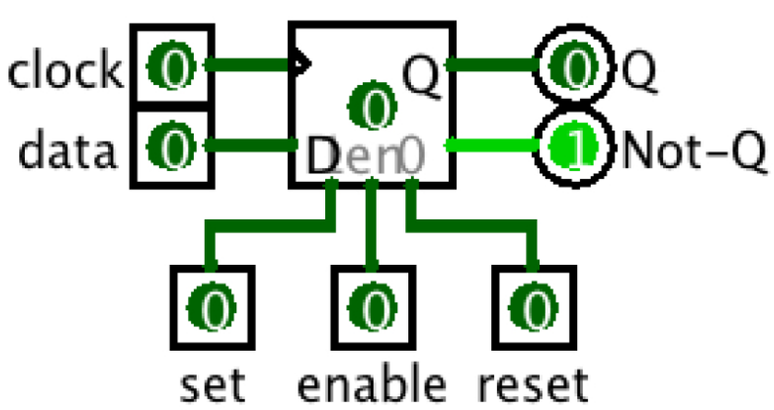
Submit online via Canvas.

We accept only the following file formats:

- .circ
- .pdf
- .txt
- or tar, zip, or gz of the above

Q0: FF Prologue (10pts)

Consider LogiSim's "D flip/flop" (under "Memory"):



This is basically one bit of memory. When "enable" is asserted and "clock" does the right thing, the value of "data" gets stored in the device (and is readable via "Q").

LogiSim lets you configure what the clock needs to do in order to trigger the storage. We're interested in two:

- When the clock has a **rising edge** (from 0 to 1)
- When the clock is at the **high level** (that is, 1)

This simulator also lets you also inspect the stored bit and play with it directly, via clicking on the number inside the flip-flop box; real circuits don't do this.

Your assignment:

- Build this circuit and play with it.
- What happens when "set" is asserted?
- What happens when "reset" is asserted?
- If the FF is configured to be triggered on **high-level**, what happens when the data input changes while the clock is high? While the clock is low?
- If the FF is configured to be triggered on a **rising edge**, what happens when the data input changes while the clock is high? While the clock is low?

Q1: Getting Edgy (30pts)

(This problem would have been challenging if I hadn't let you use the baked-in high-level component)

Next week, we'll show you how to build a level-triggered flip-flop from gates (and also note that some people insist such beasts are properly called "latches").

For this problem.... using basic gates and level-triggered flip-flops (and, if necessary, other devices whose internals we've already built), build a rising-edge triggered flip-

flop (with set, reset, and enable).

You might want to try LogiSim's somewhat clunky baked-in delays (e.g., to see that, try the inverter-AND trick into a rising edge FF)

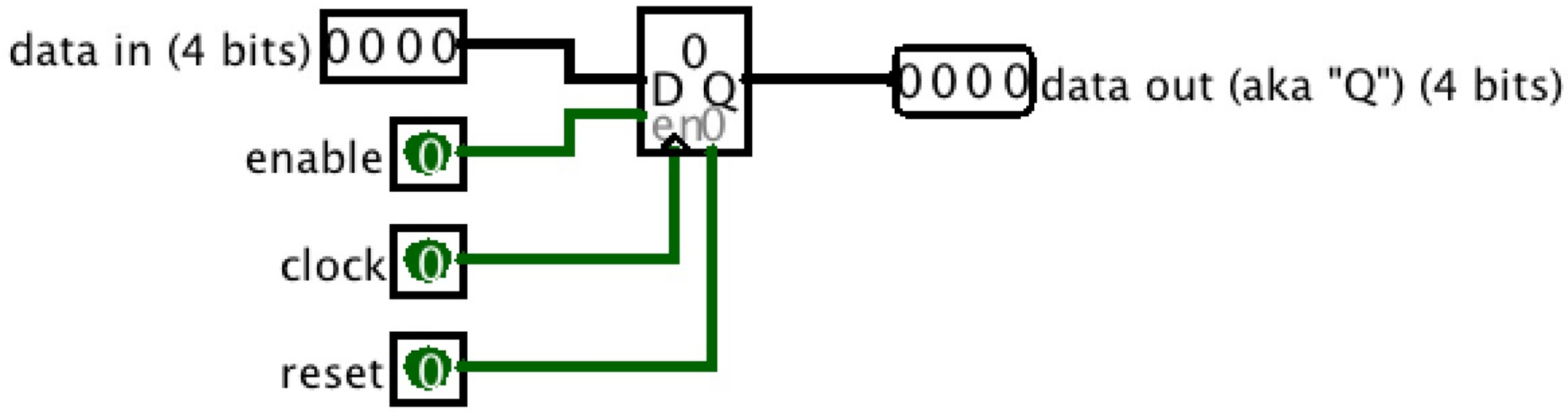
Or you may use my "delayed NOT" gate, for a 5 point penalty.

If you can do it using neither my delayed NOT nor LogiSim's baked-delays... you get a 5-point bonus!

Q2: Register Prologue (5pts)

LogiSim lets you store multi-bit quantities via a device called a "Register".

Build this simple circuit (with a 4-bit rising-edge register).



Play with it. Get used to what it does.

Q3: Flip-Flops into Registers (5pts)

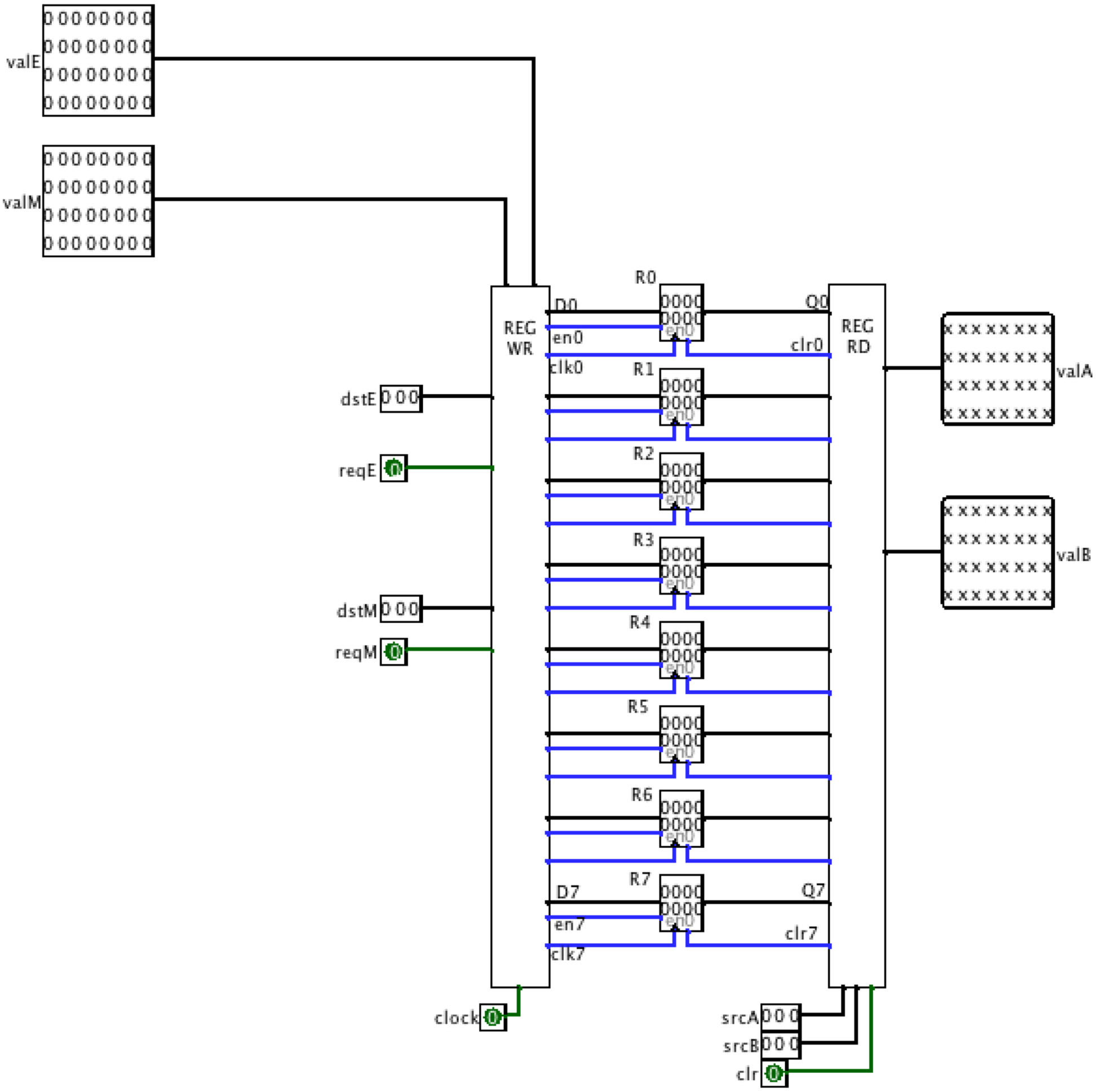
Using D flip-flops (any kind) and basic gates, buid a 4-bit register that does the same thing as LogiSim's 4-bit register.

Q4: Your Y86 Register File (50pts)

The textbook develops the Y86 processor, a simplified version of the ubiquitous X86. In this course, you will build one in LogiSim. In this problem, you build the first piece: the register file! (See pp 361-363 for more info).

More specifically:

- You'll have 8 32-bit registers, each specified by a 3-bit index.
- You want to be able to write to up to two different registers simultaneously (triggered by a rising edge on the clock)
- You want to be to read two different registers simultaneously, independent of the clock



Your Assignment

Download **hw3-skeleton.circ (https://ssl.cs.dartmouth.edu/~sws/cs51-s15/hw3/hw3-skeleton.zip)**

Build the "regwriter" and "regreader" subcircuits (I've given you skeletons there already) so that they can drive your eight 32-bit LogiSim registers!

- When "reqE" is asserted, the value at "valE" should be loaded into the register specified by "dstE" at clock rise
- When "reqM" is asserted, the value at "valM" should be loaded into the register specified by "dstM" at clock rise
- "valA" should output the value currently stored in the register specified by "srcA"
- "valB" should output the value currently stored in the register specified by "srcB"
- when "clr" is asserted, the registers should all be cleared to zero. (Yes, I sloppily used "clr" and "reset" as synonyms.)

Inside your boxes, you may use basic gates, constants, and components we've built before. If you feel that encapsulating common functionality into subcircuits would improve clarity and correctness, do so!

Homework 3

Criteria	Ratings	Pts
Q0. view longer description		10 pts
Q1. Correctness		15 pts
Clarity (and neatness)		10 pts
Simplicity		5 pts
Q2. Correctness		5 pts
Q3. Correctness, Clarity (and neatness), and Simplicity		5 pts
Q4. Correctness		35 pts
Clarity & Neatness		10 pts
Simplicity		5 pts
Q1 Bonus		0 pts
		Total Points: 100