Slide 1

Administrivia

- Reminder: Homework 4 due today. Next homework to be on the Web soon.
- Quiz 5 next Friday. Topic(s) TBA next week, but from Appendix C.
- In reading/skimming Appendix C, okay to skim/skip sections involving hardware description language (Verilog).

Slide 2

Minute Essay From Last Lecture

- Many people guessed right. Figures C.6.1 and C.6.2 provide another analogy that may be helpful in understanding the overall idea of data flowing through logic blocks.

Slide 3

Memory Elements

- So now we (sort of) know how to design logic blocks that use switches/gates to compute output bits from input bits.
- But where do those input bits come from, and where do the output bits go? "state elements" — things that can save values.
- (Keep in mind that the goal here is to get a sense of how you can build something that stores a value out of gates/switches. Details are interesting but can to some extent be skimmed.)

Slide 4

A Very Little Bit About Clocking

- Many (most, currently?) hardware designs are based on the idea of a "clock" — something that generates regular signal changes and can be used to control when updates to state elements happen.
- As sketched in section C.7 — inputs/outputs to combinational logic block are connected to state elements. Input values are "sampled" at one point in the clock cycle and written out at a different point in the cycle — "synchronous" circuit. (So does that mean "asynchronous" circuits are also possible? yes, but well beyond the scope of this course.)
- Why do this? as a way to avoid race conditions.
- One implication, though, is that the clock cycle has to be long enough for the slowest combinational logic block!

Slide 5

Memory Elements, Continued

- Idea here is to come up with a logic block that can hold a value:
  - Inputs are old value, "set" (to 1), "reset" (to 0).
  - Outputs are value, negation of value.
- An unclocked logic block that can do this — Figure C.8.1.

Slide 6

Memory Elements, Continued

- Can then extend this to something that only samples (data) input when clock input is 1 ("D latch", Figure C.8.2) and further to something whose output only changes when clock input is 0 ("D flip-flop", Figure C.8.4).
- Notice how we're starting with simple things and using them to construct more complicated things — much as you do in writing software. "Hm!" ?

Slide 7

Register Files

- So now we have something that can read/write/save one bit. But what we want is a bunch of "registers" that can each read/write/save 32 bits. What to do?
- Usual approach — "register file", a logic block that holds a bunch of values and allows us to read and write them. Figures in section C.9 give more details (next slide) — and this should look like something that would be useful in implementing MIPS instructions with three register operands, no?

Slide 8

Register Files, Continued

- Inputs:
  - Two (multi-bit) register numbers saying which registers we want to "read" (use as input to some operation).
  - One (multi-bit) register number saying which register we (might) want to "write" (change the value of).
  - One (32-bit) value to (maybe) save in a register.
  - A "yes do a write" bit.
- Outputs:
  - Two (32-bit) values representing the contents of the two registers selected by the "read register" numbers used as input

Slide 9

Minute Essay

- None — quiz.