

Homework 3

Due: 7 Oct 2009 1:55pm

All homeworks are due at 1:55pm in the **CS 31 bin** on the second floor. No late homeworks are accepted.

Please include your *login name* on each piece of paper you hand in, and please staple your pages together before handing in.

Some of the problems on this homework must be handed in electronically. Please do each problem in a separate file. Run `cs031_handin hw03` in the directory where your files are stored to hand in. You will be prompted for the name of each file.

A note about clocks. Gating the clock signal can introduce some delay, which in the extreme is undesirable. However, there are design considerations in real life for which gating the clock makes sense and a little delay is acceptable. Thus you *may* gate the clock for this and all assignments in this class. That said, please keep it reasonable. We reserve the right to take off points for egregious and/or unnecessary clock-gating.

Problem 3.1

Fred and George want to hack the traffic light in Hogsmeade, so they can walk to The Three Broomsticks without fear and grab their butterbeers. They want the light to meet the following specifications:

Assume there is one set of lights, visible from either direction, and two pedestrian buttons, one on either side of the road. If no button is pushed, the light will remain green indefinitely. Pushing either or both buttons while the light is green causes the light to turn yellow for 10 seconds, then red for 20 seconds, then back to green. Pushing either or both of the buttons when the light is yellow or has been red for less than 10 seconds has no effect. However, pushing *both* buttons with 10 seconds or less remaining on the red light (indicating many pedestrians trying to cross) will cause the light to be red for an additional 10 seconds. There is no limit to the number of times the red light can be extended by pushing both buttons.

Note: To simplify the problem, assume that the buttons stay pressed until their status is examined. In Diglog or Logisim, buttons can be pressed whenever, but their states will only be recorded on the clock rise. When

you are testing your circuit, be sure to leave buttons pressed long enough so that the clock rises at least once.

Help the Weasley twins by doing the following: Design a sequential circuit for this controller.

Parts a, b, and c should be physically handed in to a hand-in bin, so show your work! Part d should be electronically handed in.

- a. Draw a state transition diagram for the traffic light as a finite state machine.

Use whatever you like for the states. One possibility is 00 for green, 01 for yellow, and 10 for red, but it may be desirable to use a different number of states.

The inputs to your transition function are the values of the two buttons (1 for pressed, 0 for released).

A clock cycle occurs every 10 seconds; and your circuit will essentially evaluate your state transition function on the rising edge of each clock cycle.

Write this up by enumerating the states and the state transitions.

- b. Using the state transition diagram, write truth tables for the new state bits.
- c. Write an optimized boolean expression for the truth table. You should do this using a K-map.
- d. Using Diglog or Logisim, draw a circuit for this problem and verify that it operates according to the specification.

Since a 10-second clock cycle would probably get annoying, you may set the clock to a convenient speed for testing.

To change the clock cycle in Diglog, click on the lower right hand corner of the window, which should say ROT or MIR, until it says CNFG. Click on the clock and set the rate to whatever you are most comfortable with. Another valuable option is to use a switch so that you can manually trigger change.

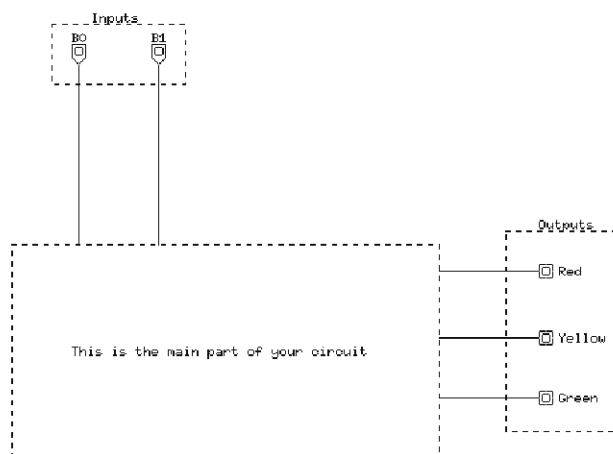
To change the clock cycle in Logisim go to the Simulate menu and click on Tick Frequency. To enable the clock make sure the Ticks Enabled box is checked (also located in the Simulate menu).

Be sure to include two switches (not counting the switch you can use in place of your clock) and three LED's/Output Pins (labeled red, yellow, and green).

You will be using the D-flipflop (available in the memory dropdown) for this problem.

Hand in a separate LGF diglog file or CIRC logisim file for this. Do not hand in LGO files.

Your inputs/outputs should look something like this:



Problem 3.2

Build a RAM with a 2-bit address space, and 2 bits of data at each address.

You may use the following components, as necessary (component names in parentheses are diglog specific):

- decoders (DECODER4)

- Controlled Buffer/tri-states (BC8)
- multiplexers (MUX4 and MUX8)
- D flip-flops (DNEG or DPOS)
- a clock (CLOCK), though this is not part of the RAM itself
- logic gates

Your RAM should expose the following wires to the outside world; they should be wired to appropriate structures for testing. For testing, you may find it useful to construct a “mini-bus”, that is, attach tri-state-guarded LED’s and switches to the wires which may be input or output.

- 2 address input wires
- 2 wires for data input/output (yes, input and output will be on the same 2 wires), to be attached directly to a bus
- a clock wire
- two control wires: **select** and **load data**

On the rising edge of the clock . . .

. . . when the select control wire is **high**, and the load data wire is **high** (meaning output is enabled), the data wires are filled with the data at the address specified.

. . . when the select control wire is **high**, and the load data wire is **low** (meaning output is disabled), the address specified is filled with the contents of the data wires.

. . . when the select control wire is **low**, regardless of the voltage of the load data wire, the data wires are disconnected (given high impedance)

Note: Logisim Users: Output pins attached to wires with high impedance values display all ”X”s.

Diglog Users: Wires with only high impedance values (no high or low voltages) show up as green in diglog; wires with high impedance attached to LED’s come out black (not gray or red).

Note: Since the idea here is to mimic the functionality of the RAM in logisim, there is no need to have an intermediate storage of output; the output of the RAM should change instantaneously.

If the specs of this RAM seem wiggly to you, please don't hate us. We want you to learn this because it matches the control for logisim's RAM, which you will need for RISC (rejoice because the diglog RAM is very different and much wiggier).

Note that the Moon-1 computer includes a RAM4X8, which uses a different control. When designing your own circuits, always use the RAM in Logisim and SRAM8K in Diglog.

Hand in only an LGF diglog file or logisim CIRC file for this problem. Do not hand in LGO files.