Lab 6: RTL and Memory Systems

Review Questions (optional, not to be handed in)

- How many 1-bit flip-flops are needed in a RAM with 8 data lines and 16 address lines?
- How many data lines are needed on a ROM with 8 address lines and $2^{10}$ bits of storage?
- How many address lines are needed on a ROM with 8 data lines and $2^{10}$ bits of storage?
- A memory system that is to store $2^{16}$ eight-bit words could be made in many ways, including with eight chips of $2^{16}$ one-bit values, or eight chips of $2^{13}$ eight-bit values — show diagrams of each way, clearly indicating the number of bits on each multi-bit line.
- Build a RAM with 8 data lines and 5 address lines out of 4-bit registers.
- Write one RTL program to compute the average of the values in R1, R2, R3, and R4; and another to find the average of four values in memory cells M[0]...M[3] of a memory chip M.

Circuits (Lab Work)

Obtain the (empty) files for cs240 Lab #6, and create yet another vending machine, as follows: Your system will receive a sequence of 5-bit unsigned integer values between 0 and 30 (inclusive) as input, representing money coming into the machine; in each step produce a 1-bit “vend” output and a 5-bit numeric “change” output. You may restrict inputs to be a multiple of 5 if you wish.

a) Write an RTL program in the file `lab6.py` to describe a solution to this problem, and build the corresponding circuit in `lab6.v`. For this version, the 5-bit integer values should come from a 5-bit “dip switch”, the “vend” should be an LED, and “change” should be a 7-segment display. In your circuit, give a “users comment” explaining how it is to be used.

b) Now imagine that we wish to embody a standard test suite for a vending machine in a ROM memory. Add to your RTL program and then your circuit, to

i. include a “done” output which is triggered once the (now legal) input of 31 is read;

ii. include input from consecutive addresses of memory system constructed from two ROM chips, each of which has some of the bits of each 5-bit number.

You circuit should have everything from Part a intact, and an “input_or_memory” switch controlling whether you read the 5-bit inputs from the dip switch or the ROM. You should provide a `lab6rom1.mem` file giving a good test suite and ending with 0, and your circuit should not do any more memory operations once the “done” output is on.

c) Update your RTL program and then your circuit, to keep track of a log of change and vending done for each input, in a second memory system comprised of one or more RAM chips; create one or more .mem files showing what should be in RAM after your tests.

Your circuit must have only one clock and either (a) *nothing except optionally a single negation* between your clock and the circuit elements it drives, or (b) a detailed explanation of why your logic on the clock wire can’t lead to a problem when your circuit is used correctly.