Computer Systems, Spring 2019

Week 3 Lab

Digital Circuits

Aims

This lab exercise is unassessed; there is nothing to hand in. Its aims are

Problems to solve on paper

1. Draw the diagram of a combinational circuit (constructed from logic gates) called and3 that takes three inputs a, b, c and outputs 1 if and only if all inputs are 1. (Hint: use a couple of and2 gates.)

2. Simulate a 4-bit ripple carry adder as it calculates the sum of 6+5, with carry input of 0. To keep this easier, do the simulation showing the inputs and outputs of each fullAdd circuit; don't expand each fullAdd into logic gates. (Hint: Convert the numbers to 4-bit binary words, and use the truth table of the fullAdd circuit to work out the sum and carry bits.)

3. Estimate the gate delay of a 4-bit ripple carry adder. Assume that at time 0 all the inputs to the adder become valid. How many gate delays later will all the outputs be valid?

4. Describe the behaviour of a dff (delay flip flop).

5. Design a 4-bit register circuit. It takes two inputs: load is a bit, and x is a 4-bit word. The register contains a 4-bit state. The output should be the current value of the state. At each clock tick, the register loads x into its state if load=1, but retains the previous state if load=0. (Use four copies of the reg1 circuit as a building block.)

6. Explain why the clock speed in a synchronous circuit depends on the maximum gate delay in the circuit. Discuss what would happen if the clock runs too fast.

7. Recall that a 4-bit word adder takes three inputs: a carry input cin and two 4-bit binary words x and y. It produces two outputs: a carry output bit $cout$ and a 4-bit sum word s, where $s = x + y + cin$. Design a 4-bit word adder/subtracter circuit called *addsub*. It takes another input *sub*. Its other inputs, and its outputs, are the same as for the adder, except that all the words are two's complement integers. The *addsub* circuit can do either addition or subtraction, and the *sub* input tells it which calculation to do: if *sub*=0 then $s = x + y$, but if *sub*=1 then $s = x - y$. *Hint:* it's a good idea to do this in stages

   (a) You already have an adder

(b) Design a subtracter; it doesn't take a *sub* input, but always calculates $x - y$. To do this, invert each bit of y (you'll need four inverters) and set the carry input to 1.

(c) Now introduce the *sub* input.

(d) Set the carry input to *sub* because if you're adding, the carry input should be 0 and if you're subtracting the carry input should be 1. In either case the carry input should be *sub*.

(e) Let's name the second input to the adder z. If $sub = 0$ then $z_i = y_i$, and if $sub = 1$ then $z_i = inv\ y_i$. So $z_i =$ (if sub=0 then y_i else inv y_i). You can do that with a multiplexer.

(f) (Optional.) It's fine to keep the multiplexer, but using either Boolean algebra or truth tables, we can see that $z_i = xor2\ sub\ y_i$, so it's possible just to use one logic gate on each y input.

8. Simulate your *addsub* circuit in order to calculate the following: $2 + 3$, $5 - 3$, $3 - 5$, $4 + (-2)$, $4 - 2$, $-3 - 4$.

Problems to try on the computer

1. Create a register circuit using a delay flip flop (dff) and a multiplexer (mux1). The lecture slides show how to do this. Simulate it for several clock cycles with suitable input, with the same input data used in the lecture. Notice that the clock input of the flip flop is connected to a binary switch. This switch should normally be left at 0. When you want to create a clock tick, you should *pulse* the clock switch: that means, click it to make it 1, then click again to make it 0.

2. CircuitVerse has a full adder component. Connect four of these in order to make a 4-bit binary word adder. Connect it to input and output components, and test it (for example, try adding 2+3 and see if you get 5).

3. Implement your add/subtract circuit (described above in the paper exercises) and test it with the simulator.