Computer Systems 2

Assessed Exercise 1

Hand out: Friday, 3 October 2014
Hand in: Monday 27 October 2014 at 16:30

This exercise will be carried out in the laboratory sessions in Weeks 3 and 5. The aims of this exercise are to learn more about basic circuits, to practice using LogicWorks, and to understand how a register transfer machine circuit works. The exercise consists of two phases:

- Phase 1 covers material from lectures in Weeks 1—2, and you should complete it in the Week 3 lab. The aim is to familiarise you with some basic circuits.

- Phase 2 covers material from lectures in Weeks 3—4, and you should start work on it in the Week 3 lab and complete it in the Week 5 lab. Phase 2 explores a circuit called the Register Transfer Machine (rtm42.cct).

Phase 1 (Week 3 lab, week of 6 October)

Report

Obtain the file **Report.docx**. You can find it on the Moodle page for CS2. Copy Report.doc into your workspace, and edit it (put in your own name, etc.) As you work through the exercise, document what you do in Report.doc, **explaining what you did in text**, and **including screenshots showing your work in the LogicWorks** window. To take a screenshot, make sure the LogicWorks window is selected, and press Alt-PrintScreen. Then go to the window with Report open in Word, and paste the screenshot.

For each section, your description should be in plain English, but it should be clear and precise enough to enable somebody to use the component in a circuit design:

- it should be clear which wire is connected to which port
- marks are awarded for showing an understanding of what each component does.
- marks are awarded if the explanation is **clear**
- remember to include the screen shots of your circuit diagrams
- describe in your own words what each component words

Launch LogicWorks. As you work on a circuit, save it frequently in your workspace, because LogicWorks is prone to crash. Often it will show a dialogue box with an error message; often you can just click OK and proceed without problems. It is

recommended that you close the Timing window that appears at the bottom of the main LogicWorks window; this reduces the likelihood that LogicWorks will crash.

If you're looking for a component in LogicWorks, type in its name (or a prefix of its name) in the Filter text box that appears above the list of components in the right hand window.

Exercises

1. **Inverter (NOT), Binary Switch** and **Binary Probe.** Construct a circuit that contains an inverter (called NOT in LogicWorks). The input should be connected to a Binary Switch, and the output should be connected to a Binary Probe. You can change the value of the input by clicking on the binary switch component. Check that the output is correct. Make a screenshot of the circuit and include it in your report.

2. **Mux1 implemented with logic gates.** Implement the mux1 circuit (Lecture 2, Slide 11) using logic gates. Use binary switches to provide the inputs, and a binary probe to show the output. Make a screenshot of LogicWorks showing the output when the inputs are set to c=0, x=1, y=0, and another screenshot when c=1, x=1, y=0.

3. **Hex keyboard wo/STB** and **Hex Display.** Make a circuit that inputs a 4-bit word, inverts each bit, and outputs the results. Use a Hex keyboard wo/STB component to provide the inputs, and use a Hex Display component to show the output. Make a screenshot showing the result when the input is set to hex 5.

4. **Mux-2x4 T.S.** This is a multiplexor black box circuit provided in the LogicWorks library. It takes one control input bit named S0, and two 4-bit word inputs, named [1D0, 2D0, 3D0, 4D0] and [1D1, 2D1, 3D1, 4D1], and an inverted enable input EN. It produces a 4-bit output word. Build a small circuit that contains one Mux-2x4 T.S. component, and that uses hex keyboards for its inputs, and that uses a hex display to show the output value. Describe in your own words what the component does. Your description should be plain English, but it should be clear and precise enough to enable somebody to use the component in a circuit design.

5. **Adder-4.** Design a circuit that inputs two 8-bit binary numbers, adds them, and outputs the carry output and 8-bit sum. Use two Adder-4 circuits to perform the arithmetic, and use hex keyboards and hex displays for input and output. Demonstrate the circuit by adding two integers, and include a screenshot in your report. Make sure that your input data tests the circuit adequately: the input numbers should be reasonably large, and they should demonstrate that carry propagation between the two Adder-4 circuits works correctly.

6. **Reg-4.** This is a 4-bit register component. Experiment with it, describe the behaviour, and include a screenshot. Use a Binary Switch for the clock input; don't use a clock generator component. Normally the clock input should be 0.

To make a clock tick, pulse the clock by clicking it twice (this will make it go from 0 to 1, and then from 1 to 0). Describe the behaviour of the circuit, and include a screenshot.

7. **Decoder-4.** Experiment and describe. This is essentially the same as a demultiplexor, except the data input is called Enable and is received inverted.

Phase 2 (Week 5 lab, week of 20 October)

Obtain the file **rtm42.cct**. You can find it on the Moodle page. This file is the circuit diagram for a register transfer machine with 4-bit words and 2-bit addresses (hence its name, rtm42). Since the addresses are 2 bits wide, there are $2^2 = 4$ registers, each 4 bits wide.

Study the rtm42 circuit. The aim is to understand everything in it. To learn how it works, study the diagram, study the explanation of the Register Transfer Machine in the lectures, and experiment with the circuit through simulation.

All the inputs, including the clock, are provided by the Hex Keyboard and the Binary Switches, which are all located in a row near the top of the diagram. You won't need the clock generator or the waveform window, so you can close the waveform window to get more screen space for the circuit diagram.

The clock input should normally be kept at 0. To simulate a clock cycle, first set all the input signals using the switches (but don't touch the clock). When you have all the inputs ready, then pulse the clock: click it to bring it up to 1, then click it again to bring it back down to 0. This is a clock tick, ending the previous clock cycle and beginning the next one.

At any time, you can clear the circuit: this means to set all the registers to 0. Pulse the clear control signal by clicking it twice. Normally, the clear signal should be 0.

8. On a sequence of clock cycles, we wish to make the rtm42 circuit perform the following sequence of operations:
    1. load 3 into reg2
    2. load b into reg0
    3. load 7 into reg3
    4. set reg1 = reg2 + reg3
    5. set reg2 = reg2 + reg1

    Work out the control and data inputs required for each clock cycle, and predict the outputs you expect to see. Then simulate the circuit by operating the switches, and note down the results showing in the output devices. **Take a screenshot before you start, and a screenshot after each clock cycle. Include the screenshots in your Report.**

9. Modify the circuit so that it can perform subtraction as well as addition. Give your circuit the name **rtmSubtract.cct**. There should be a control input called

ctl_neg; if you are doing an addition leave that signal 0, but if you are subtracting set it to 1. Devise test inputs that demonstrate that your subtraction is working, and do a simulation. **Explain how your circuit works in your Report, and include a screenshot showing the circuit diagram and suitable test data.**

What to hand in

Use Moodle to submit:

- **Report.docx** which contains your results from each part of the exercise.
- Your modified circuit **rtmSubtract.cct**

The assessment will be based on the correctness and clarity of your solutions. Phase 1 and Phase 2 have equal weight.

How Exercise 1 will be marked:

Following timely submission on Moodle, the exercise will be given a numerical mark, between 0 (no submission) and 30 (perfect in every way). These numerical marks will be converted to a band (C2 etc.)

- **Phase 1**, Demonstration and documentation of components: up to 15 marks
  - 8 marks for showing an understanding of what each component does. Proportional marks (if the student is halfway there, 4 marks awarded).
  - 7 marks for clear documentation. 7 marks if the documentation is precise enough that it could be used to build an actual circuit (i.e. it's clear which wire is connected to which port). If it is not possible to tell which bit of a word is most/least significant, but the documentation is otherwise ok, 3 or 4 marks awarded.
- **Phase 2**, RTM: up to 15 marks
  - 7 marks if the student can run the circuit to perform the prescribed sequence of operations. Full marks if the student shows how they set the controls, and show the effect of each cycle. Partial marks awarded if the solution is unclear.
  - 8 marks for the subtraction circuit. 4 marks for the circuit design; if two separate adder circuits used, 1 mark off, and if it's necessary to set two separate control signals to make the circuit add or subtract, 1 mark off. 4 marks for testing: 2 marks for only one fully working test case. 4 marks awarded when the student demonstrates several examples, involving both addition and subtraction and using both positive and negative numbers.