KTH Information and Communication Technology

IS1500 (not part of IS1200)

Logic Design Lab (LD-Lab)
2017-10-26

Introduction

The purpose of this lab is to give a "hands-on" experience of using gates and digital building blocks. These build blocks are then used in the next lab where we will construct a subset of a simple processor.

The intended learning outcomes of this laboratory exercise are as follows. That is, after finishing the lab, the student shall be able to

1. construct basic digital circuits using a graphical design environment.
2. analyze the behavior of both combinational and sequential logic.

Lab Environment

The lab is done in **Logisim**, a graphical tool for simulation and design of logic circuits. The tool is open source and is based on Java. It should work on Windows, Mac OS, and various versions of Linux. Please download and install the tool from http://www.cburch.com/logisim/

Resources and Reading Guidelines

This laboratory exercise contains the assignments, but not the theory or the background. Please make sure that you are familiar with the following material:

- The course book by Harris & Harris (H&H), second edition. Chapters 1,2,3, and 5 are relevant for this lab. In particular, you should be familiar with the content in 1.4 (Number systems), 1.5 (Logic Gates), 2.8 (Combinational Building blocks), 3.2 (Latches and Flip-Flops), 3.3 (Synchronous Logic Design), and 5.2 (Arithmetic Circuits).
- The two lectures for Module 3 (Logic Design):
  Lecture 7 "Combinational Logic" and Lecture 8 "Sequential Logic".
- The documentation of Logisim, which is available from http://www.cburch.com/logisim/.

Examination

This laboratory exercise is examined individually. Each student should submit their own report using the KTH social assignment submission system. Please see the course webpage for more information. Rules for submission of the report:

- The report **must** be written by each **student individually**. This means that it is explicitly **forbidden** to copy, cut, and paste any text from the Internet or from other students. Each report will be automatically checked for plagiarism.

- However, it is allowed to discuss solutions and how to address the problems with other students, as long as you personally perform all the exercises in Logisim and you write all your answers individually.
- All assignments must be solved and all the bullet points for the different tasks completely answered. Note also that the report must clearly show the task number for each solution. The solution must be **submitted as a PDF-file**. There is no page limit for the report or other requirements of the formatting.

Warming up: Getting familiar with Logisim

Go through Logisim's "Beginner's tutorial". You can find the tutorial in the documentation section on Logisim's webpages. Note that you need to perform each of the steps by using the software, so that you learn how to use the tool.

Assignment 1: Decoder

Task 1.1

Construct a 2:4 decoder from the following components: 2 inputs, 4 outputs, 2 NOT−gates, and 4 AND−gates. Mark the two inputs as A0 and A1 and the four outputs as Y0, Y1, Y2, and Y3.

For task 1.1, include the following in your report:

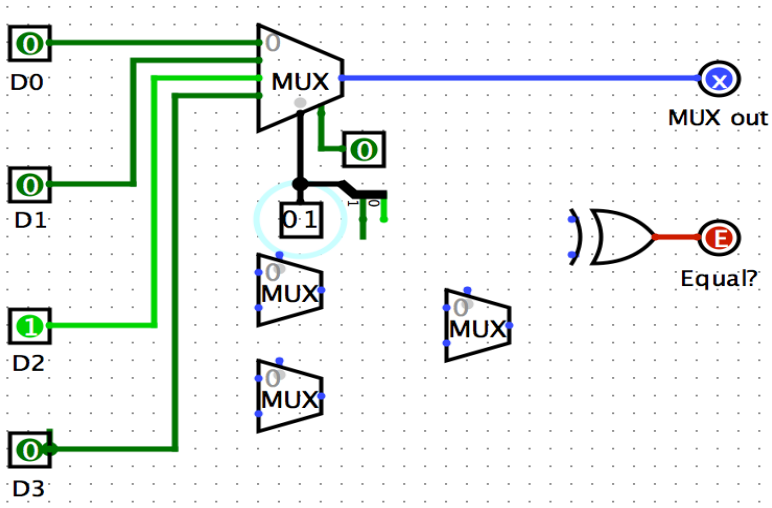
1. A screenshot of your complete design.
2. Give a short explanation for how you have verified that your design is completely correct.

Assignment 2: Multiplexer

Before you try to solve the following assignment, carefully read through the Logisim documentation about *"Libraries and attributes"* and *"Wire bundles"*.

Task 2.1

Construct a circuit that is similar to the one below. Note that the multiplexer at the upper part of the figure is a 4:1 multiplexer and that the three multiplexers at the bottom are 2:1 multiplexers. Note also that the 2:1 multiplexers have their select signals at the north side of the component. The component that looks like a fork below the 4:1 multiplexer is a splitter. Note also that the select port is a 2-bit port (black lines to the MUX below). The green port below the MUX is the enable signal.



For task 2.1, include the following in your report:

1. Explain why output “MUX out” is blue, why output signal “Equal?” is red, and the select signal of the 4:1 multiplexer is black.

Task 2.2

A 4:1 multiplexer can be constructed from 3 different 2:1 multiplexers (see H&H or the lecture slides). Your task is now to connect the three 2:1 multiplexers in such a way that their output gives the same behavior as a 4:1 multiplexer. You should also connect both the output from this new multiplexer (the combination of the three 2:1 multiplexers) and the old 4:1 multiplexer to the XOR gate.

For task 2.2, include the following in your report:

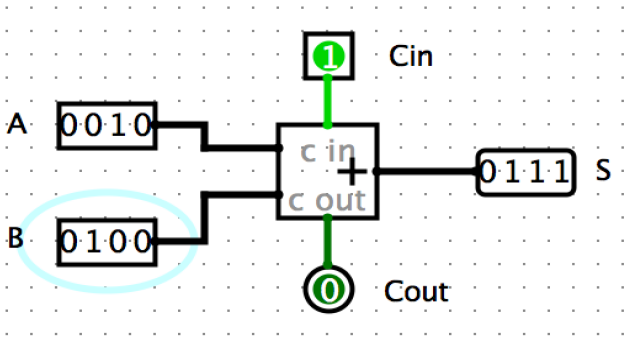
1. Include a screenshot of your complete design. Note that it should look similar to the figure in task 2.1, but with additional wires. You are not allowed to add any extra components.
2. Explain how you can test the correctness of the component using the output signal “Equals?”.

Assignment 3: Adder

Task 3.1

The following figure depicts a 4-bit adder. The figure uses the adder component from the *Arithmetic* library. Construct the circuit as in the figure below, but with the following input and output signals:

B = 0110, S = 0000, Cin = 1, Cout = 1



For task 3.1, include the following in your report:

1. Include a screenshot of your design. Note that you need to find out what input signal A should have, to fulfill the requirements in task 3.1. The figure should contain the correct input and output signal values.

Task 3.2

Construct a 4-bit ripple-carry adder by using four 1-bit full adders, two 4-bit input signals (A and B), one carry in signal, one carry out signal, and a 4-bit output signal (S). The design needs to use three 4-bit splitters.

For task 3.2, include the following in your report:

1. Include a screenshot of your design. Your design should use the input and output values that were requested in task 3.1.

Assignment 4: Latches

The three first assignments did not contain any memory, they were all examples of combinational logic design. The rest of the assignments will be examples of sequential logic design, that is, circuits that have states.

Task 4.1

Construct an SR latch by using two NOR-gates. Clearly mark S and R inputs, and the Q and Q' outputs.

For task 4.1, include the following in your report:

1. Include a screenshot where both S and R are 0. Explain what the possible values for Q and Q' can be.
2. Include a screenshot where S = 1 and R = 0. Explain the possible values for Q and Q' in this case. If you toggle S (switch between 0 and 1), do then Q or Q' change? Why or why not?
3. What happens if S is equal to 0 and R is toggled between 1 and 0?

Task 4.2

Create a D latch by extending the SR latch that you created in task 4.1.

For task 4.2, include the following in your report:

1. Include a screenshot of your complete design of the D latch, when both the D and the CLK signals are 0. Explain what happens if D is equal to 0 and you toggle CLK? What happens if CLK is 0 and you toggle D? Are Q and Q' changed? Why or why not?
2. What happens if CLK is 1 and you toggle D?
3. Explain the benefits with the D latch compared to the SR latch.
4. Explain the problem with D latches and why they are not used in synchronous sequential logic.

Assignment 5: D Flip-Flops and Registers

Before you try to solve the following assignment, read through the Logisim documentation about *"Subcircuits"* carefully.

Task 5.1

Create a sub-circuit of the D latch. Instantiate two D latches and connect them together to form a D Flip-Flop (see H&H or the lecture slides). Your circuit should have two 1-bit input signals: (i) Din (data in), and (ii) CLK (clock signal). Moreover, it should have one output signal Q.

For task 5.1, include the following in your report:

1. Include a screenshot of your design of the D Flip-Flop, where the two D latches are sub-circuits.
2. Explain in what circumstances the value of Q changes. Explain by giving concrete examples.

Task 5.2

Construct a 4-bit register by reusing the D Flip-Flop as sub-circuits. Your design should have a 4-bit input signal called X, a 4-bit output signal called Y, and a clock signal.

For task 5.2, include the following in your report:

1. Include a screenshot of your design.

Assignment 6: Design of a Synchronous Sequential Circuit

Task 6.1

Design a synchronous sequential circuit with the following components:

- An 8-bit register. Use standard register component available in Logisim (located in the memory library).
- A clock input signal (located under the Wiring library)
- One 8-bit input signal, called I.
- One 8-bit output signal, called O.
- An 2:1 multiplexer with 8-bit width.
- An 8-bit adder. Use the adder from the standard library.
- A NOT gate with 8-bit data width.
- A button, located in the Input/Output library.

The circuit should increment the output signal O with the value I each time the clock ticks (rising edge) as long as the button is not pressed. If the button is pressed, the output should be decremented with value I. Note that you can start the simulation mode by checking "Ticks Enabled" under the "Simulation" menu in Logisim.

For task 6.1, include the following in your report:

1. Include a screenshot of your design. Briefly explain how the circuit works.
2. Explain the meaning of a synchronous sequential circuit. Which of the tasks in this lab can be classified as synchronous sequential circuits?

Revision History

2016-10-25: Fixed two minor typos.

2017-10-26: Minor fixes to make the questions clearer.