ECS 154B Lab 1, Spring 2016
Due by 11:55 PM on April 8, 2016
Via SmartSite

Goals

- Learn how to use Logisim.
- Lean how to use the circuit analyzer tool.
- Learn an alternative to combinational logic.

Logisim

- The project page for Logisim is at `http://www.cburch.com/logisim`.
- You can download Logisim on SourceForge for Windows and OS X at `http://sourceforge.net/projects/circuit/`. If you are using Linux, your package manager for your distribution may have Logisim as well.
- You can find a brief tutorial for Logisim on Professor Farrens' class website at `http://american.cs.ucdavis.edu/academic/ecs154a/postscript/logisim-tutorial.pdf`.
- The **User's Guide** and **Library Reference** in the Help menu in Logisim are also very helpful.

Introduction

When designing a circuit to implement a function, there are two different design routes that you can take: use pure combinational logic, or use a Read Only Memory (ROM). Combinational logic involves combining AND, OR, and NOT gates to implement the Boolean equation that you derive from the truth table. On the other hand, by using a ROM, you simply implement the truth table in hardware.

ROM Implementation

A memory unit can be viewed as simply a truth table in hardware. Here are the steps to creating a ROM implementation of a truth table.

1. Create a ROM, where the number of addressing bits is equal to the number of inputs, and the data bit width is equal to the number of output bits.
2. Using the input bits as the address, fill in the entries of the ROM with the correct outputs for that combination of inputs.
3. To get the output, address the ROM using the concatenation of the input signals.

Example

The following is an example of implementing the XOR function using microcode. On the next page is the truth table for XOR ($\oplus$).

XOR		
X	Y	X $\oplus$ Y
0	0	0
0	1	1
1	0	1
1	1	0

Applying step 1, we first create a ROM with 2 address bits, because we have 2 inputs. Each entry is 1 bit wide, because there is only output. Next, we fill in the ROM using X and Y as the address bits.

X	Y	Address	Value
0	0	0	0
0	1	1	1
1	0	2	1
1	1	3	0

The Logisim implementation is included with the given files for the assignment, in the subcircuit **ROM XOR**.

ROMs and Sequential Circuits

Here is how you can use a ROM to implement Moore sequential circuits. Instead of implementing the truth table, the ROM implements the next state transition table and the outputs in hardware. Here is how to do it:

1. Create a ROM where the number of address bits is equal to (number of state bits) + (number of input bits). The data bit width is equal to (number of state bits) + (number of output bits).
2. Fill in the ROM using the state transition table. Each entry in the ROM contains the next state bits along with the output at that current state.
3. To address the ROM, concatenate the next state bits from the ROM with the input signals. The next state portion of the output of the ROM will feed back into the input of the ROM.

Example

Suppose we want to create a sequential circuit that while it is in state 0, it outputs 01, when in state 1 it outputs 11, and changes between states whenever the input, X, is 1. The machine starts in state 0. The transition table for this machine would be as follows.

State Transition Table			
Current State	X	Next State	Output
0	0	0	01
0	1	1	01
1	0	1	11
1	1	0	11

By applying step 1, we would first create a ROM that has 2 address bits, 1 for the next state and 1 for the input X. For each address, the ROM has data entries that are 3 bits wide, 1 for the next state and 2 for the output. Applying step 2 and using the top bit in each entry in the ROM to store the next state and the bottom 2 for the output:

Current State	X	Address	Next State	Output	Value in ROM
0	0	0	0	01	001 (`0x1`)
0	1	1	1	01	101 (`0x5`)
1	0	2	1	11	111 (`0x7`)
1	1	3	0	11	011 (`0x3`)

Because of the way that we decided to implement the machine, the top most output of the ROM feeds back into the top most bit of its input.

A Logisim implementation of this circuit is included with the assignment, in the subcircuit **ROM Sequential**. Note that the hex editor in Logisim expects hexadecimal values, so it is necessary to convert the concatenation of the bits from binary to hex, as shown.

Sequential Circuits using ROMs in Logisism

In order to implement sequential circuits with using ROM, the ROM must be synchronous. Logisim does not come with a built in synchronous ROM, but you can make your own by connecting a register to the output of the built-in ROM module.

Assignment

1. Implement a combinational circuit using both combinational logic and a ROM.
2. Implement a sequential circuit using only a ROM.

For each circuit, please create a sub-circuit with appropriately named inputs and outputs.

1. Combinatorial Circuit

Implement the circuit that has the following truth table using both **combinational logic** and **microcode**.

- When creating the combinational circuit using combinational logic, you may only use splitters and these gates: AND, OR, NOT, XOR.
  - In addition, the logic must be the minimal amount to express the truth table.
- When creating the combinational circuit using a ROM, you may only use the ROM module.
- **In** are the inputs and **Out** are the outputs.
- All unspecified input and output combinations are don't cares, as are the **D**s in the table.

In7	In6	In5	In4	In3	In2	In1	In0	Out2	Out1	Out0
0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	1	0	0	0	1	1
0	0	0	0	0	1	0	1	1	0	1
0	0	0	0	0	1	1	0	1	1	1
0	0	0	0	0	1	1	1	0	1	0
0	0	0	0	1	0	1	0	0	0	1
0	0	0	0	1	1	0	1	1	1	0
0	0	0	0	1	1	1	0	0	0	1
0	0	1	1	D	D	D	D	1	0	0
0	1	1	1	D	D	D	D	0	1	0
1	0	0	0	D	D	D	D	1	1	1
1	0	0	1	D	D	D	D	0	1	1
1	1	0	0	D	D	D	D	1	0	1
1	1	1	0	D	D	D	D	0	0	0
1	1	1	1	D	D	D	D	0	0	1

Your sub-circuit should have the following inputs and outputs:

Inputs

- In7-0

Outputs

- Out: The concatenation of Out2-0, with Out2 as the top most bit and Out0 as the bottom most bit.

The Circuit Analyzer Tool

Don't be intimidated by the number of inputs when doing the combinational circuit. You can use Logisim's **Analyze Circuit** tool, in the Project drop-down menu, to have Logisim build the circuit for you. To learn how to use it, click on Help → User's Guide. In the User Guide, click on Combinational Analysis and read how to use it. You will find this tool very helpful in this and future labs.

2. Sequential Circuit

For this section, you may only use the ROM, Register, Splitter, and Clock modules in Logisim. Refrain from using any logic gates in this part. The sequential circuit example **ROM Sequential** discussed above discusses in detail how you should solve this part.

Implement a Moore Model sequential circuit that takes as input a stream of bits. **There are two separate conditions to keep track of:**

If at least 2 bits out of every 4 are 1, the circuit should output 1 after seeing the 4th bit. **Additionally**, if the circuit detects that the 4 bit sequence was 0000, then it will also output 1 after seeing the 4th bit. Otherwise, the circuit outputs 0 at all other times.

Here is an example input and output, with bits being input to the machine from **left to right**:

```
Input:   0001 0000 0011 1111 0100
Output:  0000 0000 1000 1000 1000 0
```

Remember, since we are implementing a Moore model, the output will be delayed by one clock tick.

Your sub-circuit should have the following inputs and outputs:

Inputs

- Clock: The system clock.
- X: The current value in the input bit stream.

Outputs

- Out: The output from your sequential circuit.

Testing

You will be provided with the following circuits to facilitate testing.

- **Combinational Input**: Generates the inputs for the combinational circuit.
  - Inputs:
    - Clock: The system clock.
  - Outputs: From top to bottom, In7-In0, the input signals to the combinational circuit.
- **Sequential Input**: Generates the inputs for the sequential circuit.

- Inputs:
  - Clock: The system clock.
- Outputs:
  - X: The input bit stream for your sequential circuit.
  - NotDone: 1 when the test is still ongoing. Connect this to the *sel* input on your ROM.

You will also be provided with the following log files to test if your circuits are correct:

- **CombOut.correct.log**
  - The log file containing the correct outputs for the combinational logic circuit using combinational logic.
  - The X's in the file indicate don't cares.
- **ROMCombOut.correct.log**
  - The log file containing the correct outputs for the combinational logic circuit using a ROM.
  - The X's in the file indicate don't cares.
- **ROMSeqOut.correct.log**
  - The log file containing the correct outputs for the sequential circuit using a ROM.
  - The X's in the file indicate don't cares.

We will be testing your code using Logisim's logging feature. To log the results of your program, do the following:

1. Attach a probe or pin to the wires that you want to log, and give it a name.
2. Click Simulate → Logging.
3. In the Selection tab, select the signals you want to log.
4. Click on the File tab.
5. Select a file to log the signals to.

You will need to create three separate log files, one for each sub-circuit:

CombOut.log		
Signal Name	Radix	Description
Input	2	The concatenation of In7-0.
CombOut	2	The concatenation of Out2-0 from the combinational circuit.

ROMCombOut.log		
Signal Name	Radix	Description
Input	2	The concatenation of In7-0.
ROMCombOut	2	The concatenation of Out2-0 from the ROM combinational circuit.

ROMSeqOut.log		
Signal Name	Radix	Description
X	2	The input bit stream X.
ROMSeqOut	2	The output from the sequential circuit.

To see if your circuit is correct, use the Python program, `tester.py`, included with assignment. To use it, type, in your command line, with all files in the same directory:

```
python tester.py correct_log_file your_log_file
```

where `correct_log_file` is the file that contains the correct signals, and `your_log_file` is the name of the log file you have your signals in. For example, to test if your combinational circuit is correct, you would type:

```
python tester.py CombOut.correct.log CombOut.log
```

if your log file was named `CombOut.log`.

If you are using Windows, you may want to add Python to your system path to make testing easier, if you have not already.

Note: the Logisim logger is a ”lazy” logger, in that it only logs a value if there is a change in the output value. This causes the log file's output to be different from what one would expect for a given output. Hence, you will need to test all the problems manually and finally validate by using `tester.py` as stated above to match your output to the correct output.

Resetting the Log Files

If your circuit has some errors the first time, in order to retest your file, you must do the following steps in this order:

1. Delete the contents of your log file except for the headers, the names of the signals.
2. Reset your circuit by pressing Ctrl + R, or by going to Simulate → Reset Simulation.
3. Simulate again.
4. Run `tester.py` again.

Grading and Submission

- 95% Implementation
  - 1/3 for each circuit.
- 5% Interactive Grading
  - If there are still errors in your circuit, you will have 5 minutes to fix them in interactive grading.
- It is possible to receive a lower grade than this, if you do not understand how your implementation works.
- You must attend interactive grading to receive a grade for this project.
- Times for interactive grading will be posted after the assignment is due.

Submission

- Upload to SmartSite the zip/tar of your .circ file along with a README file that contains:
  - The names of you and your partner.
  - Any difficulties you had.
  - Anything that doesn't work correctly and why.

  - Anything you feel that the graders should know.

- You may submit your assignment as many times as you want, but if you plan on submitting using slip days, ensure that you submit **something** before the initial due date. Due to a SmartSite limitation, you may only resubmit if you submitted something before the initial deadline.

- You have 2 slip days.

Hints

- When filling in the values for the ROM in the combinational circuit, it may be a good idea to write a program to fill in the values for the ROM. If you don't, you may have to fill in a large amount of numbers by hand. It is by no means required, though.

- If you need help, come to office hours for the TAs, or post your questions on Piazza.