ECS 154B Lab 1, Spring 2018
Due by 11:59 PM on April 11, 2018
Via Canvas

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 on Dr. Carl Burch's website.
- You can download Logisim on SourceForge for Windows and OS X. If you are using Linux, your package manager for your distribution may have Logisim.
- You can find a brief tutorial for Logisim on Professor Farrens' class website.
- 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. Here 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	00 (`0x0`)	0
0	1	01 (`0x1`)	1
1	0	10 (`0x2`)	1
1	1	11 (`0x3`)	0

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

ROMs and Sequential Circuits

You can also 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 current 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, we get the following table:

Current State	X	Address	Next State	Output	Value in ROM
0	0	00 (`0x0`)	0	01	001 (`0x1`)
0	1	01 (`0x1`)	1	01	101 (`0x5`)
1	0	10 (`0x2`)	1	11	111 (`0x7`)
1	1	11 (`0x3`)	0	11	011 (`0x3`)

Because of the way that we decided to implement the machine, the most significant bit of the ROM feeds back into the top most bit of its input. The order of the next state and output bits can be reversed, as long as the bits fed back to the input are the correct ones.

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 combinational logic.
2. Implement a combinational circuit using a ROM.
3. Implement a sequential circuit using only a ROM.

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

1. Combinational Logic

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

- When creating the combinational circuit using combinational logic, you may only use splitters and these gates: AND, OR, and NOT.
- The logic must be the minimal amount to express the truth table.
- **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	0	0	0
0	0	0	0	0	0	0	1	1	1	1
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	1	1	1	0	0
0	0	0	0	0	1	0	0	1	1	0
0	0	0	0	0	1	0	1	0	1	1
0	0	0	0	0	1	1	0	0	1	0
0	0	0	0	0	1	1	1	1	0	0
0	0	0	0	1	0	1	0	1	0	1
0	0	1	1	D	D	D	D	0	0	0
0	1	0	0	D	D	D	D	0	0	1
1	0	0	0	D	D	D	D	0	0	0
1	0	1	1	D	D	D	D	0	0	0
1	1	0	0	D	D	D	D	0	1	0
1	1	0	1	D	D	D	D	0	1	1

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

Inputs

- *Input*: the concatenation of the input bits.

Outputs

- *CombinationalOutput*: The concatenation of *Out2-0*, with *Out2* as the top-most (most significant) bit and *Out0* as the bottom-most (least significant) bit.

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. ROM

Implement the same circuit from Part 1 using a **ROM**. The truth table is identical for both circuits. You may only use a ROM and splitters for this part. Your sub-circuit should have the following inputs and outputs:

Inputs

- *Input*: the concatenation of the input bits.

Outputs

- *ROMCombinationalOutput*: The concatenation of *Out2-0*, with *Out2* as the top-most (most significant) bit and *Out0* as the bottom-most (least significant) bit.

3. Sequential Circuit

For this section, you may only use the ROM, Register, Splitter, and Clock modules in Logisim. You may not use any logic gates in this part. The sequential circuit example above discusses in detail how you should solve this part.

Implement a Moore Model sequential circuit that takes as input a stream of bits. If at least 4 of the last 6 bits inputted (not including the current input bit) to the machine are 1, the circuit outputs a 1. Otherwise, the circuit outputs a 0. It also outputs a 0 if the 6 bits have not been input yet. The circuit then resets to check the next set of bits.

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

```
Input:   000000 100100 011111 111100 010010
Output:  000000 000000 000000 100000 100000 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.
- *Not Done*: 1 when the test is still ongoing. Connect this to the *sel* input on your ROM.

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**:
    - *Input*: The concatenation of the input signals *In7-In0* 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.
    - *Not Done*: 1 when the test is still ongoing.
    - *Logging*: Alternates between 1 and 0 while the test is still ongoing. Used as a workaround for the lazy logging feature of Logisim. A probe has already been attached to this output for you.

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

- **part1correct.txt**
  - 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.
- **part2correct.txt**
  - 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.
- **part3correct.txt**
  - 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:

part1.txt		
Signal Name	Radix	Description
Input	2	The concatenation of In7-0.
CombinatonalOutput	2	The concatenation of Out2-0 from the combinational circuit.

part2.txt		
Signal Name	Radix	Description
Input	2	The concatenation of In7-0.
ROMCombinationalOutput	2	The concatenation of Out2-0 from the ROM combinational circuit.

part3.txt		
Signal Name	Radix	Description
Logging	2	Used to defeat Logisim's lazy logging. The values are meaningless.
X	2	The input bit stream X.
SequentialOutput	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.txt your.txt
```

where `correct.txt` is the file that contains the correct signals, and `your.txt` 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 part1correct.txt part1.txt
```

if your log file was named `part1.txt`.

The tester was written for Python 2.7. If you want to use Python 3, you will need to encapsulate each print statement's argument with parentheses. If you are using Windows, you may want to add Python to your system path to make testing easier, if you have not already.

Resetting the Log Files

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

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.

If the first line of your log file has a line in which the last number is missing, you may safely delete it. Additionally, if you reset your circuit while you are still logging, you will notice dashes in the log file indicating when the circuit was reset. You may delete everything from those dashes up to the headers to clear your log file.

Grading

- 50% Implementation
  - 12.5% for correct Part 1.
  - 12.5% for correct Part 2.

  - 25% for correct Part 3.
  - Partial credit at the grader's discretion.

- 50% Interactive Grading
- It is possible to receive a lower grade than what you earned, 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 close to when the assignment is due.

Submission

Read the submission instructions carefully. Failure to adhere to the instructions will result in a loss of points.

- Upload a signle .circ file to Canvas.
- In the text submission box, include the following:
  - The names of you and your partner.
  - Any difficulties you had.
  - Anything that doesn't work correctly and why.
  - Anything you feel that the grader should know.
- Only one partner should submit the assignment.
- You may submit your assignment as many times as you want.

Late Policy

Note: the following is subject to change.

You may submit up to 48 hours late for partial credit. The partial credit that you receive is determined by the following equation:

$$\text{final grade} = \text{original grade} \cdot \left(1 - \frac{\text{hours late}^2}{48^2}\right)$$

This system penalizes you less if you submit closer to the original deadline.

Hints

- When filling in the values for the ROM in the combinational circuit, it may be worthwhile 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.
- It is recommended to create an Excel spreadsheet for your sequential circuit's state transition table in order to save time and cut down on errors. In particular, the functions `DEC2BIN()`, `BIN2HEX()`, the `&` concatenation operator, and this Stack Overflow post may come in handy.
- If you need help, come to the office hours for the TA, or post your questions on Piazza.