ECEN 240 Lab 4 – Small Scale Integrated Circuits (SSI)

Name: Brodric Young

Purpose:

1. Successfully reduce a Boolean logic equation to its simplest form.
2. Simulate the simplified equation using *Logisim Evolution*.
3. Learn how to use Small Scale Integrated circuit (SSI) logic devices (7400 family of ICs).
4. Verify that the circuit correctly implements the truth table of the original equation.
5. Explore the merits of using only NAND gates to implement a truth table (NOR gates are also functionally complete, but we will use NAND gates).

Procedure:

There will be two parts to this lab. Both parts of the lab will use the same truth table shown below. With this Truth table, you will:

- Simplify the Boolean expression two different ways (Using Boolean theorems and K-maps)
- Implement the simplified equation in Logisim 4 different ways
- Learn the merits of each type of simplification and implementation

A	B	C	D		X
0	0	0	0		1
0	0	0	1		1
0	0	1	0		1
0	0	1	1		0
0	1	0	0		1
0	1	0	1		1
0	1	1	0		0
0	1	1	1		0
1	0	0	0		1
1	0	0	1		1
1	0	1	0		1
1	0	1	1		0
1	1	0	0		1
1	1	0	1		1
1	1	1	0		0
1	1	1	1		0

Lab 4 Part 1

This portion of the lab is intended to be completed after **Lesson 4 part 1**.

Write the non-reduced (10 product) Boolean equation for the previous truth table in SOP form:

X = A'B'C'D' + A'B'C'D + A'B'CD' + A'BC'D' + A'BC'D + AB'C'D' + AB'C'D + AB'CD' + ABC'D' + ABC'D

Simplify the equation using Boolean theorems, and write the reduced equation:

X = C' + B'D'

K-map-Based Simplification

Verify that your fully reduced equation is correct using a Karnaugh map:

- Fill in the K-map from the truth table.
- Identify the prime implicants.
- Highlight each prime implicant loop with a different color or pattern.
    - Select a group of cells in the table that you want to highlight.
    - Right click and select the "Borders and Shading" menu.
    - Under the "Shading" tab, select a color and/or a pattern.

CD \ AB	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	0	0	0
10	1	0	0	1

CD \ AB	00	01	11	10
00	m0	m4	m12	m8
01	m1	m5	m13	m9
11	m3	m7	m15	m11
10	m2	m6	m14	m10

Make sure you understand how the minterms, m0-m15, are mapped in the above K-map.

List the minterms that are present in the **largest** prime implicant loop (m0, m1, …):

Loop1 = Σm(m0, m1, m4, m5, m8, m9, m12, m13)

List the minterms that are present in the **smallest** prime implicant loop:

Loop2 = Σm(m0, m2, m8, m10)

Verify that the K-map based simplification matches your Boolean theorem-based simplification.

*****Take Lab 4: Quiz 1*****

(10-point quiz)

Build Version 1 of the Circuit: Use AND, OR and NOT Gates

Implement the reduced Boolean equation in *Logisim* using AND, OR, and NOT gates. Use the following pin names:

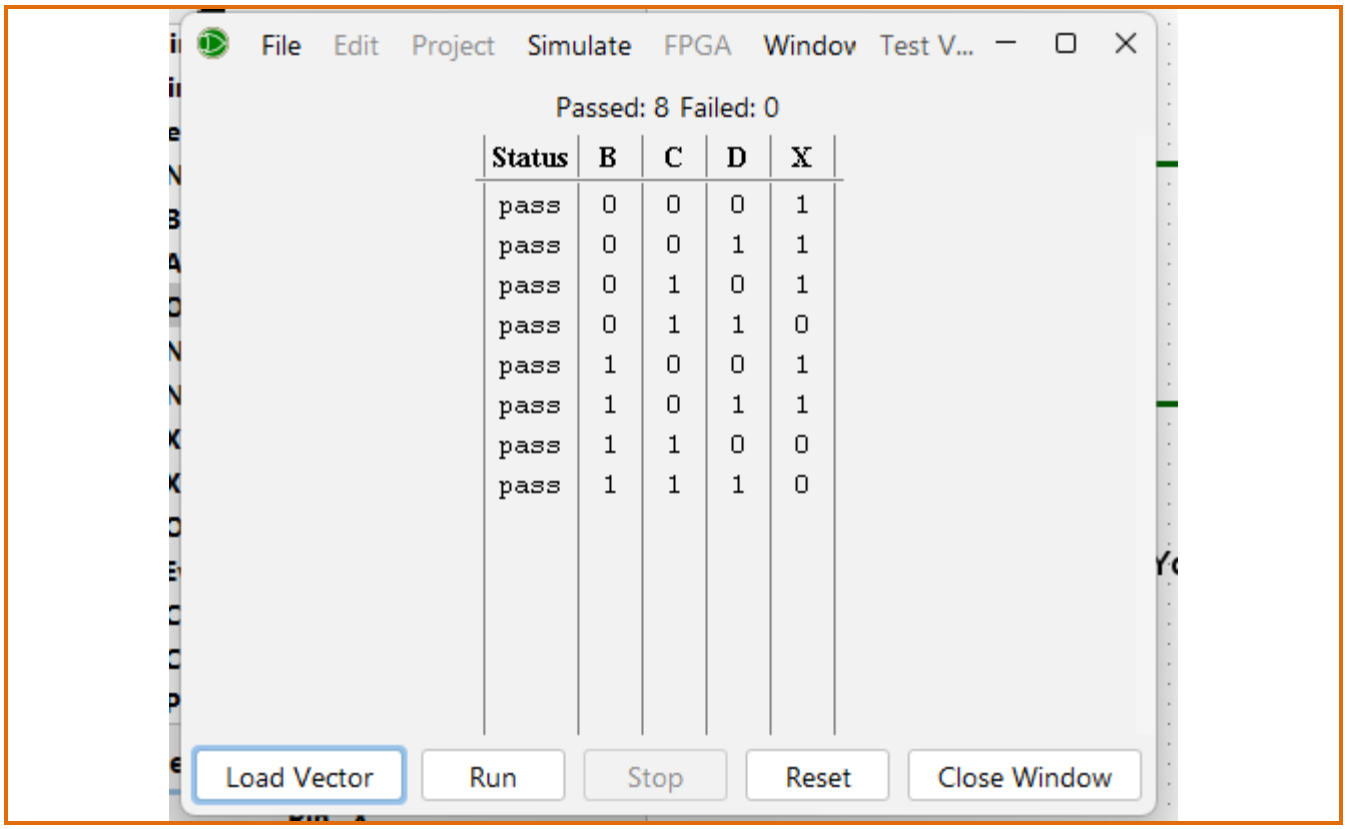
Input Names (not all will be needed)	Output Names
A	X
B	
C	
D	

Make your schematic nice and neat (construction quality will affect the score you receive). You can verify its functionality by manually testing the input combinations shown in the original truth table.

Once you are convinced your *Logisim* implementation is correct, verify its functionality using the "test vector" tool available in *Logisim Evolution*.

- Download the test file called "SSI_test.txt" found in the Lab 4 module in *ILearn*.
- Place the file in your *Logisim Evolution* folder.
- Run the "Test Vector" tool from the "Simulate" menu of *Logisim Evolution*.
- Select "Load Vector" and navigate to the "SSI_test.txt" file.
- The tool will display a truth table showing the tests that passed and the tests that failed. Keep working on your circuit until there are no failures.
- Take a "snapshot" of the window showing your test results, and paste the snapshot in the submission box below (the "snipping tool" may be used in Windows, or "cmd-shift-4" in Mac OSx).

(The "test vector" border box will expand to fit a screen-shot of your test results)

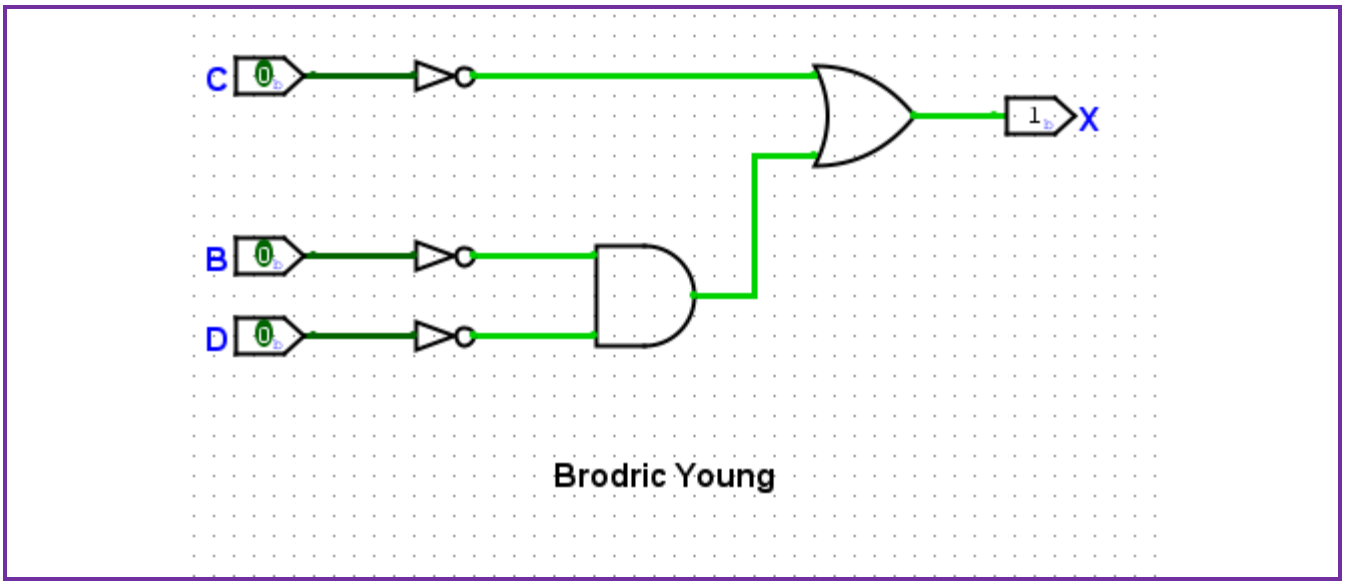


Test Vector Results with "AND, OR, NOT" Gates (7 points)

Take a "snapshot" of the circuit (including your name), and paste the snapshot in the submission box below (the "snipping tool" may be used in Windows, or "cmd-shift-4" in Mac OSx). You can zoom in on the circuit using the up and down arrows found at the bottom-left corner.

Paste the snapshot in the border box below.

(The circuit box should expand to fit the size of your screen-shot)



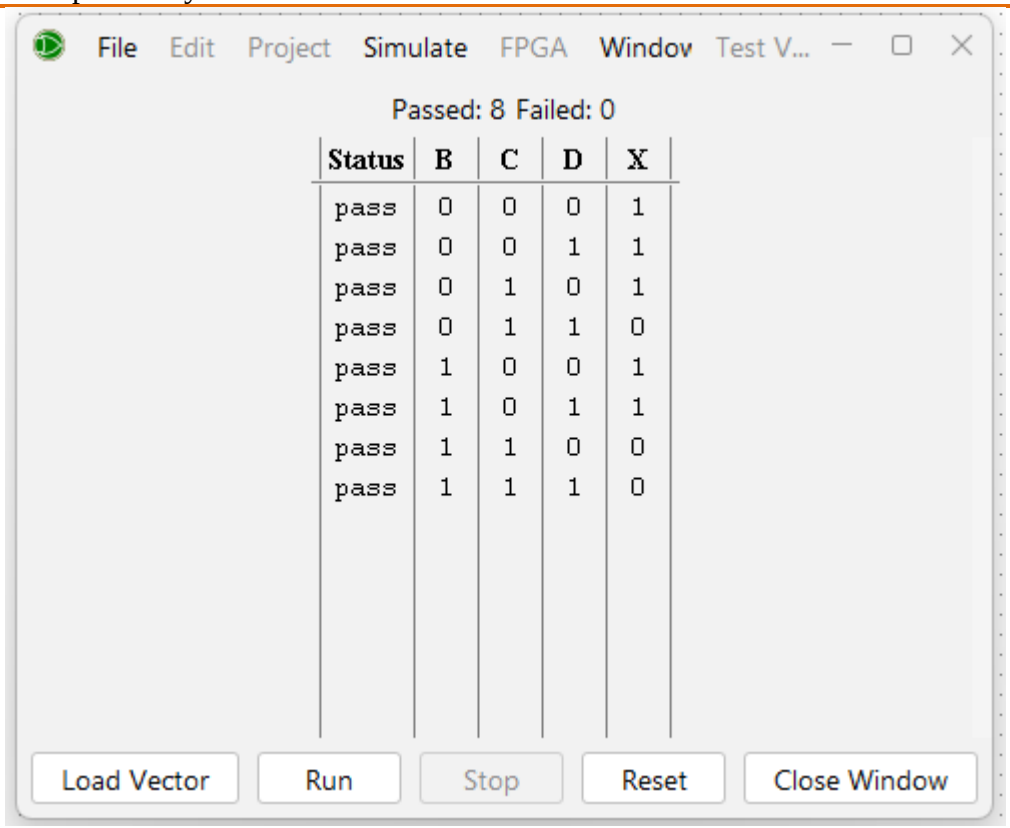
Logisim Evolution Circuit with "AND, OR, NOT" Gates (8 points)

Version 2 of the Circuit: NAND Gates Only

Re-design the circuit using only NAND gates in *Logisim* and test it against the original truth table (use the same pin names).

Once you are convinced your *Logisim* implementation is correct, test its functionality using the same test vector file used earlier (SSI_test.txt).

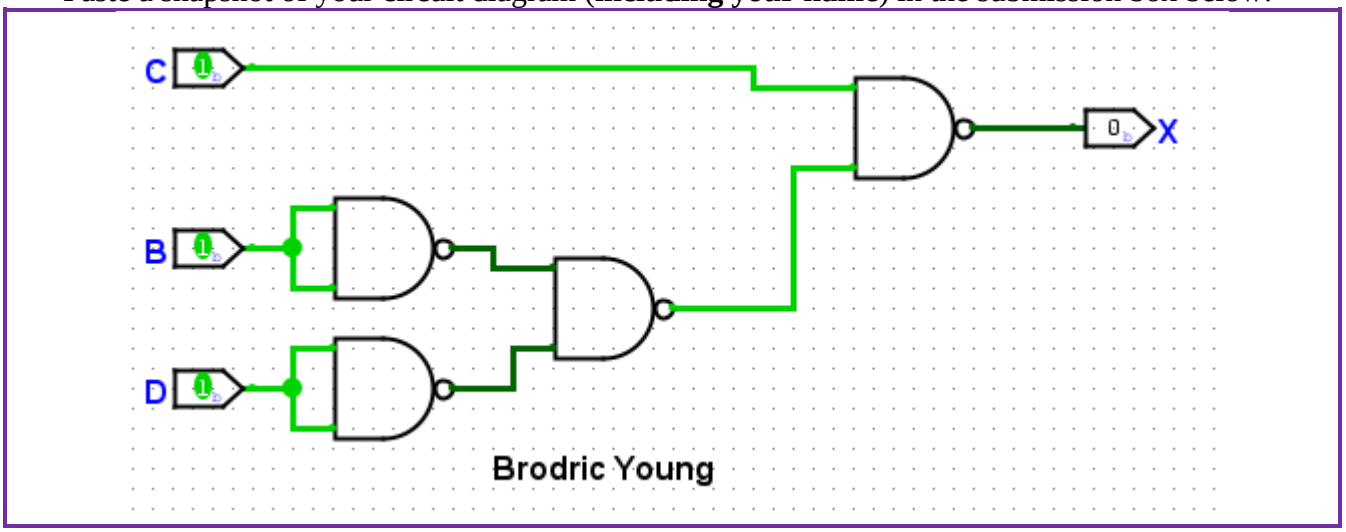
Paste a snapshot of your test results in the submission box below:



Status	B	C	D	X
pass	0	0	0	1
pass	0	0	1	1
pass	0	1	0	1
pass	0	1	1	0
pass	1	0	0	1
pass	1	0	1	1
pass	1	1	0	0
pass	1	1	1	0

Test Vector Results Using **NAND Gates** Only (7 points)

Paste a snapshot of your circuit diagram (**including your name**) in the submission box below:



Logisim Evolution Circuit Diagram Using **NAND Gates** Only (8 points)

*****Take Lab 4: Quiz 2*****
(16-point quiz)

Lab 2 Part 2

Procedure:

1. Build and debug the simplified SSI circuit using a 74HC08 (2-input AND), a 74HC32 (2-input OR), and a 74HC04 (Inverter) integrated circuit.
2. Keeping the previous circuit intact, build and debug the SSI circuit using only the 74HC00 (2-input NAND). Compare the operation of the NAND only circuit with the 3-chip solution.

*****Pass off circuit to TA or Instructor, take Lab 4: Quiz 3 *****
(24-point quiz)

Conclusions Statement

Write a brief conclusions statement that discusses the original purposes of the lab found at the beginning of this lab document.

- What are the methods of Boolean equation simplification, and which do you prefer?
- What can you learn from the several methods you used in this lab to implement the same equation?
- Are there any merits in using only NAND gates to implement a truth table?

Please use complete sentences and correct grammar to express your thoughts:

(The conclusions box will expand as you write)

In this lab, we practiced two methods of Boolean simplification. One is through using simplification theorems on the equation, and the other is through k-maps. I prefer k-maps for the really big unsimplified equations because it makes it much faster and easier. There is also several methods for implementing the same equation. We can use multiple chips for the specific AND, OR, and NOT gates, or we can turn them all into NAND gates to use a singular chip. We learned that it is possible to only use NAND gates and that we can further simplify the number of NAND gates down to a smaller number. It was fun figuring out how to only use 4 instead of the 8 or 9 that it would have taken before simplifying again. There are also some merits in only using NAND gates, one is that it takes up a lot less space only using one chip instead of three. Another is that there's less wires to implement it and NAND gates are also faster than the other three gates.

Conclusions Statement (20 points)

Congratulations, you have completed the lab!

You may now submit this document.