CDA 3201L Thursday Section
Lab number 1 -- Combinational Logic Circuits
May 24th 2014, John Gangemi, Kyle Peck

PURPOSE AND OBJECTIVES

The lab assignment is an introduction to combinational logic both in theory and practice. Its purpose is to teach the basic concepts of logic gates and Boolean algebra.

Part A of the lab asks to verify that a NAND gate can be used to implement the functionality of an AND, OR, and NOT gate using the Laws of Boolean algebra.

Part B of the lab asks to simplify a given Boolean expression, Z = W'X + W'(UY + U'Y), and implement the simplified expression using only NOT, AND, and OR gates.

COMPONENTS USED

- Integrated Circuits
    . 74LS04 HEX Inverter
    . 74LS32 2-Input OR gates
    . 74LS08 2-Input AND gates
- 5mm Red LED
    . 2.6V Forward Voltage
    . 20mA Forward Current
- 150 ohm 1/8 watt Resistor
- 5 Volt Regulated DC Power Supply
- 6 x 22 AWG Wire
- 2250 point Breadboard

DESIGN DESCRIPTION

Part A

To prove that the NAND gate is a universal gate and can implement the functionality of the AND, OR, and NOT gates one must first understand the Boolean expression and corresponding truth table for the NAND gate.

NAND Truth Table:

X	Y	X NAND Y
0	0	1
0	1	0
1	0	0
1	1	0

NAND Boolean Expression:

X’Y’

NOT Using NAND

To implement the functionality of the NOT gate, first define the truth table for the NOT gate and its corresponding Boolean expression.

NOT Truth Table:

X	X NOT
0	1
1	0

NOT Boolean Expression:
X'

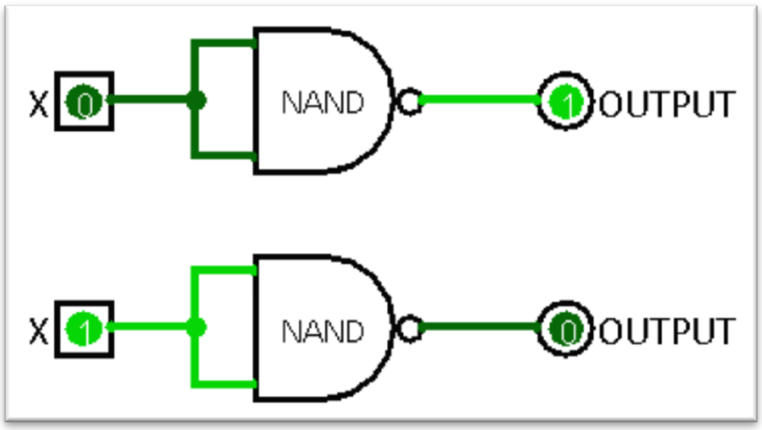
Show that the NOT Boolean expression equals the NAND Boolean expression.

$$\boxed{X' = X'Y'}$$

Since the NOT gate only accepts one input, let X and Y for the NAND gate's input be a single equivalent input.

$$\boxed{\begin{aligned} X' &= X'X' \\ &= X' \qquad \text{(Idempotent Law)} \end{aligned}}$$

Therefore the NOT gate can be modeled as follows...



AND Using NAND
To implement the functionality of the AND gate, first define the truth table for the AND gate and its corresponding Boolean expression.

AND Truth Table:

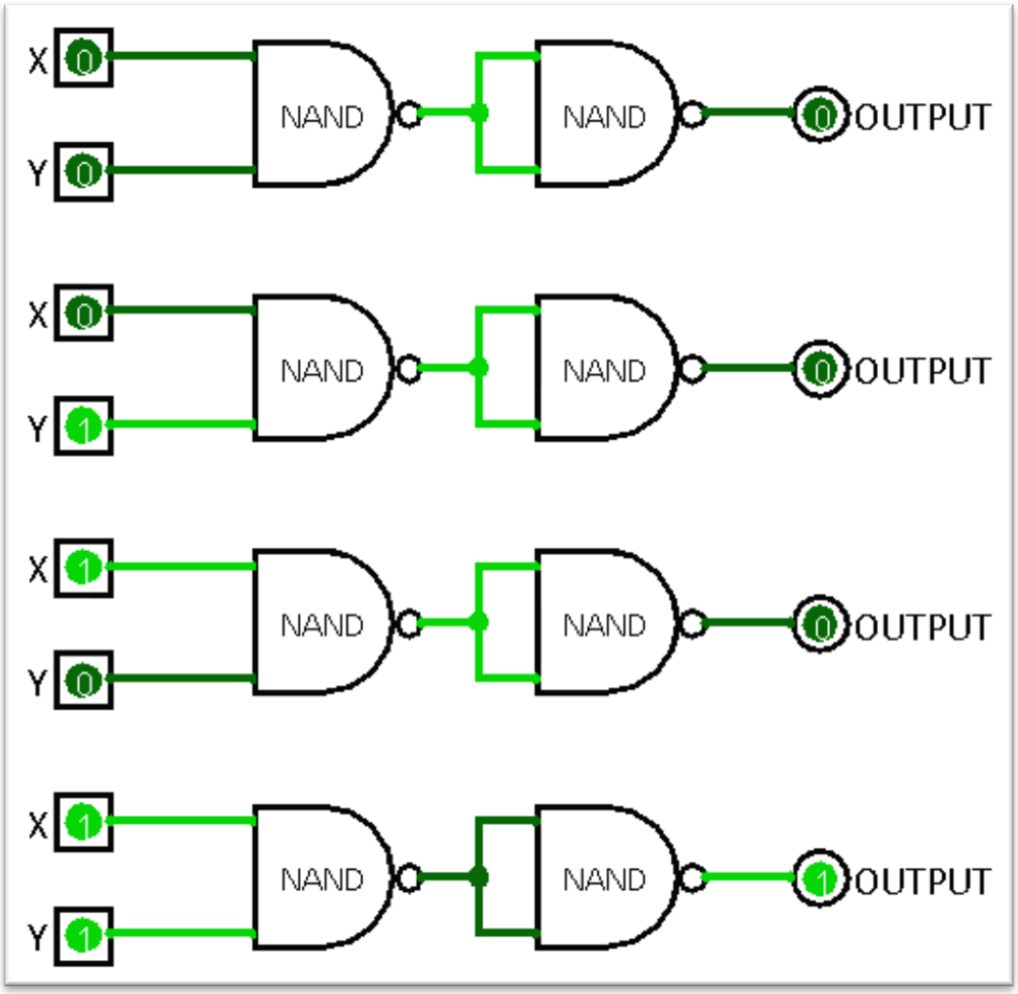
X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

AND Boolean Expression:
XY

Show that the AND Boolean expression equals the NAND Boolean expression.

XY = X'Y'
= (X'Y')' (Involution Law)
= XY

Therefore the AND gate can be modeled as follows...



OR Using NAND
To implement the functionality of the OR gate, first define the truth table for the OR gate and its corresponding Boolean expression.

OR Truth Table:

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

OR Boolean Expression:
X’Y + XY’ + XY

First, it is necessary to simplify the Boolean expression for the OR gate using a Karnaugh map.

X	**0**	**1**
Y		
0	0	1
1	1	1

Group One: XY' + XY
Group Two: X'Y + XY

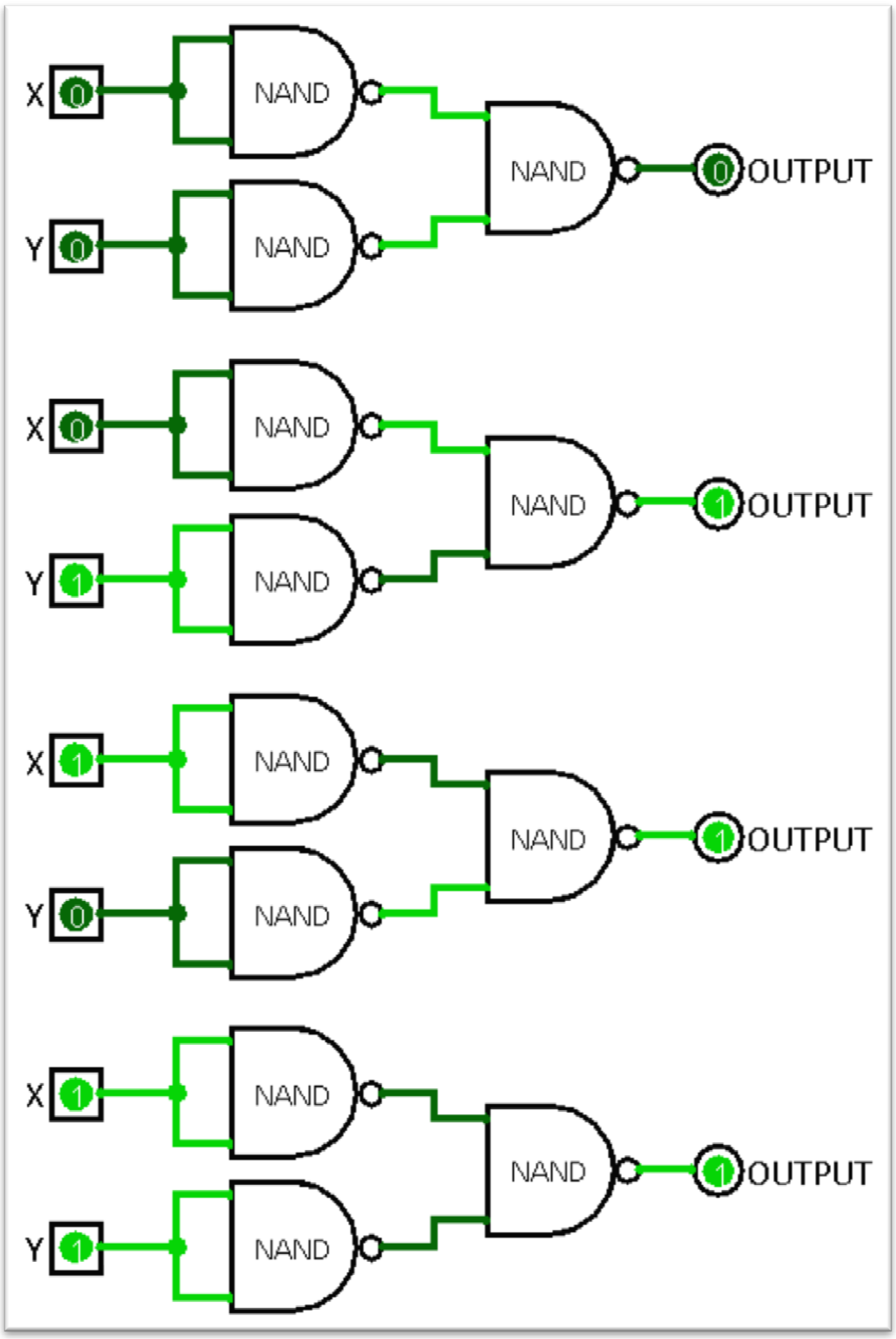
Thus eliminating non-constants from both groups and summing the products results in the following expression...

X + Y

Show that the OR Boolean expression equals the NAND Boolean expression.

X + Y = X'Y'
= ((X')(Y'))' (Involution Law)
= ((X + Y)')' (DeMorgan's Law)
= X + Y

Therefore the OR gate can be modeled as follows...



Part B

To design a circuit using the given Boolean expression it is necessary to simplify the expression first using Boolean algebra.

Z = W' X + W' (U Y + U' Y)
= W' X + W' Y (U + U') (Factoring Y)
= W' X + W' Y (1) (Complementarity Law)
= W' (X + Y) (Distributive Law)

Checking the simplified expression using a truth table proves the algebraic simplification.

X	Y	W	U	W’X	W’(UY + U’Y)	Z
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	1	1
0	1	0	1	0	1	1
0	1	1	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	1	0	1
1	0	0	1	1	0	1
1	0	1	0	0	0	0
1	0	1	1	0	0	0
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	0	0	0
1	1	1	1	0	0	0

Reducing the expression for Z from the truth table with a Karnaugh map gives a reduced form.

XY	00	01	11	10
WU				
00	0	1	1	1
01	0	1	1	1
11	0	0	0	0
10	0	0	0	0

Group One: X’YW’U’ + XYW’U’ + X’YW’U + XYW’U
Group Two: XY’W’U’ + XY’W’U + XYW’U’ + XYW’U

Thus eliminating non-constants from both groups and summing the products results in the following expression...

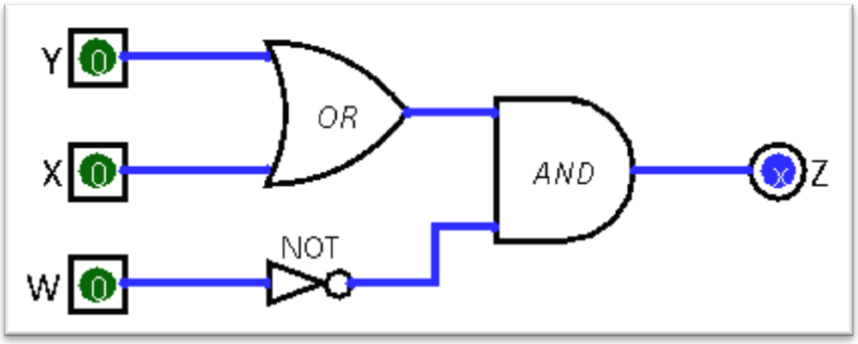
W’Y + W’X
= W’(X + Y)

OBSERVATIONS AND DATA ANALYSIS

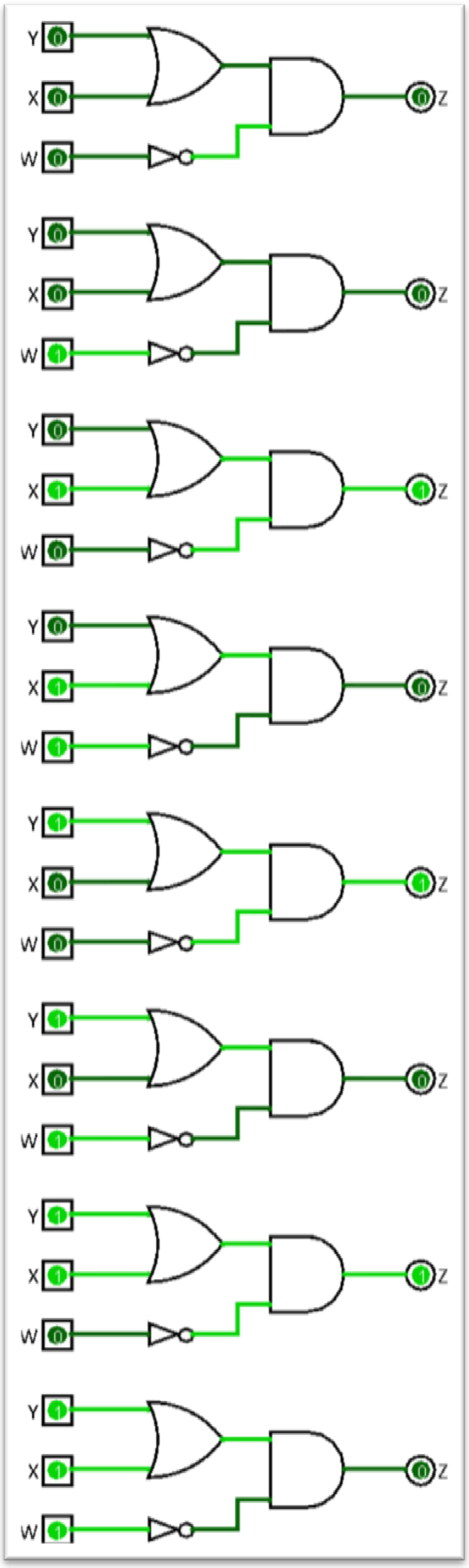
Part B

It can be noted that in the simplification of the original Boolean expression, Z = W'X + W' (UY + U'Y), that the variable "U" is eliminated thus reducing a possible 16 configurations to 8 configurations of inputs/outputs for Z.

Analyzing the simplified Boolean expression gives the following logic design schematic.



Running a simulation of the circuit pictured above using the program Logisim results in the following inputs and their corresponding outputs.


Y 0
X 0
W 0
Z 0
Y 0
X 0
W 1
Z 0
Y 0
X 1
W 0
Z 1
Y 0
X 1
W 1
Z 0
Y 1
X 0
W 0
Z 1
Y 1
X 0
W 1
Z 0
Y 1
X 1
W 0
Z 1
Y 1
X 1
W 1
Z 0

According to the Logisim simulation of the simplified expression, Z = W’(X + Y), the outputs are “Hi” when the corresponding inputs are configured in the following manner…

XY’W’ + X’YW’ + XYW’

Checking the simplified Boolean expression, Z = W’(X +Y), with a truth table verifies that the simulation does indeed display the correct configurations of inputs to outputs.

X	Y	W	X + Y	Z
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	1	1
1	1	1	1	0

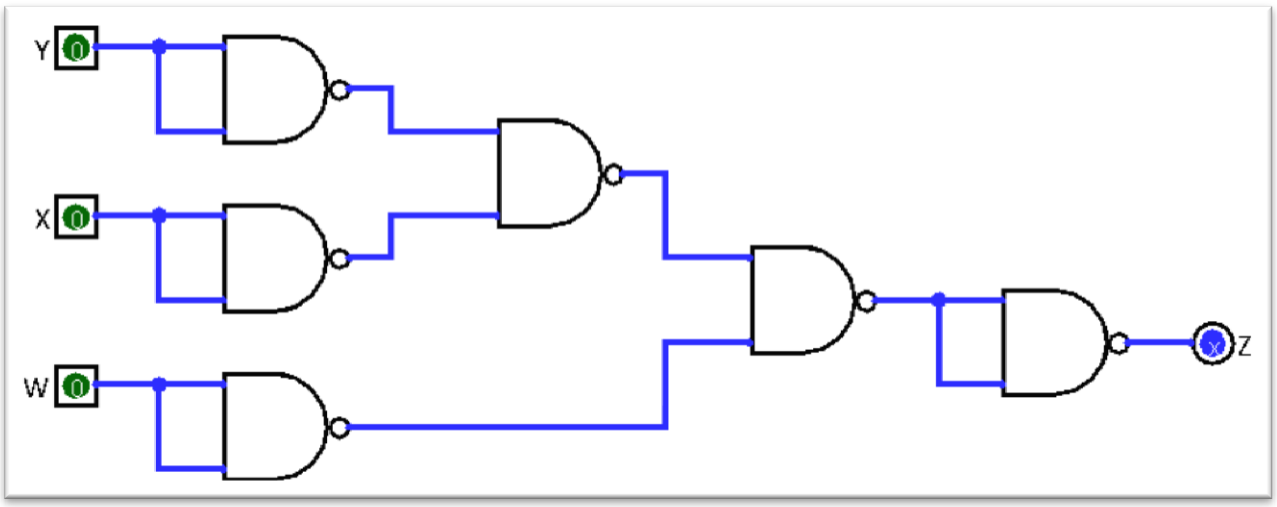
Inputs for Z: X’YW’ + XY’W’ + XYW’

Discussion and Conclusions

In this lab the group learned how to prove that every gate can be modeled using NAND gates, however it will result in a more complicated circuit where it would be more prudent to use a variety of logic gates than to restrict the implementation with a single gate.

Being that none of the group members had experience working with integrated circuits the group found the requirements of Part B somewhat daunting. Simplifying the Boolean expression and testing our logic diagram using Logisim it became apparent the procedure that needed to be followed in order to design functioning logic circuits.

Applying the knowledge gained from Part A of the lab to Part B the logic diagram could be arranged to use only NAND gates and still provide equivalent functionality. For the Boolean expression Z = W’(X + Y), there needs to be a NOT, AND, and OR logical function used to implement the circuit. Applying the logic diagrams for such logic functions using NAND gates resulted in the following design schematic using Logisim.



Running a simulation of the circuit provided an exact correlation between inputs and outputs as shown previously in the “Observations and Analysis” section of the report.