CDA 3201L - Thursday (3:30 - 4:45PM) Section 005
Lab #04 - Combinational Logic Circuits (4)
10/15/2016 - Steven Romeiro, Jennyfer Munoz

Purpose & Objectives:

Lab 4 required us to design wired circuits that implemented Boolean expressions as inputs to a multiplexer. An 8:1 multiplexer was utilized in Part A to select 1 input from 8 possible selections. The objective was to simply implement the multiplexer and understand the schematic for it. Part ‘B’ of this lab required us to implement the same 8 inputs using a 4:1 multiplexer. The objective was not only to implement the same 8 input through a 4 input mux, but to do so in a way where only 1 mux was being used. Part ‘C’ of the lab shifted away from working with Multiplexers and towards designing a comparator logic system. The system required a design that accepted two 3-bit numbers and generated a 2-bit output in the form of mathematical relational operators (i.e. < = > operators).

Components Used:

Name	Type	Quantity
74LS08	AND IC (2-inputs)	4
74LS11	AND IC (3-inputs)	4
74LS04	Inverter IC	1
74LS32	OR IC	4
74LS151	Multiplexer (8 to 1)	1
74LS153	Multiplexer (4 to 1)	1
470 Ω Resistor	Resistor	4
LED	Red LED	4
Power Supply	5v	1
Wire Kit	Assorted	1

Description:

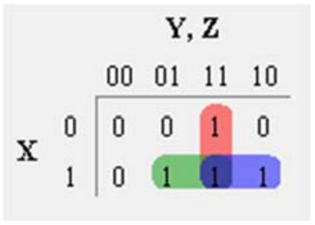
The multiplexer works that it is a 2^n multiplexer it will have n lines and each of those are the inputs that can be send to the output. Depending on the one selected is that one can determine the output that will be shown. We also have the select lines which govern which of the input lines will be selected.

Part A:

Implement an 8:1 multiplexer using expression Σm(3, 5, 6, 7).

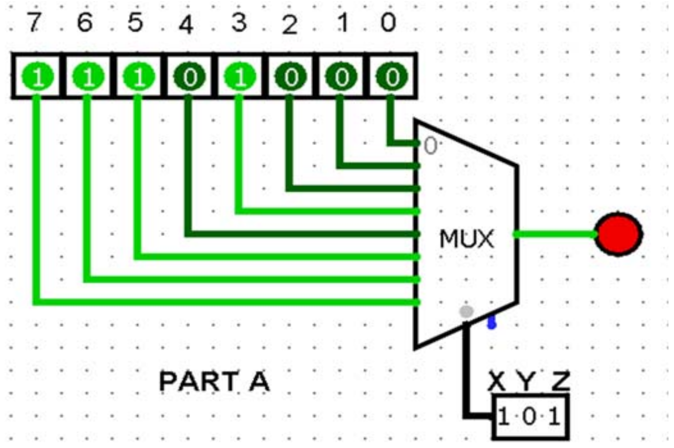
Truth Table:

X	Y	Z	Out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Boolean Expression: Y Z + X Z + X Y

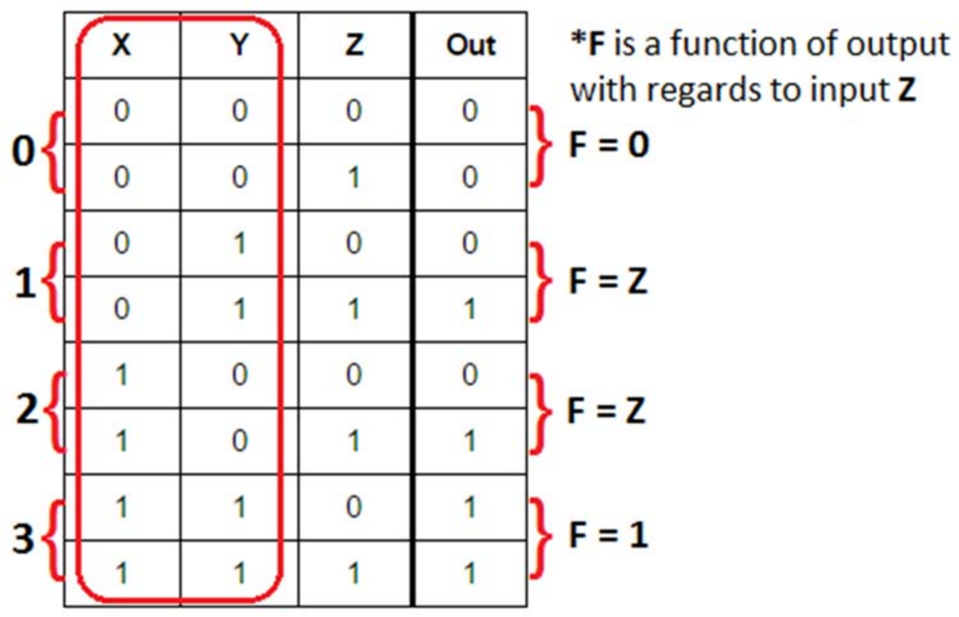
Diagram



:

Part B:

Circuit that implements the functions A+1, A-B, A+B, A-1 using the TTL 7483 4-bit adder.



*F is a function of output with regards to input Z

	X	Y	Z	Out	
0	0	0	0	0	F = 0
	0	0	1	0	
1	0	1	0	0	F = Z
	0	1	1	1	
2	1	0	0	0	F = Z
	1	0	1	1	
3	1	1	0	1	F = 1
	1	1	1	1	

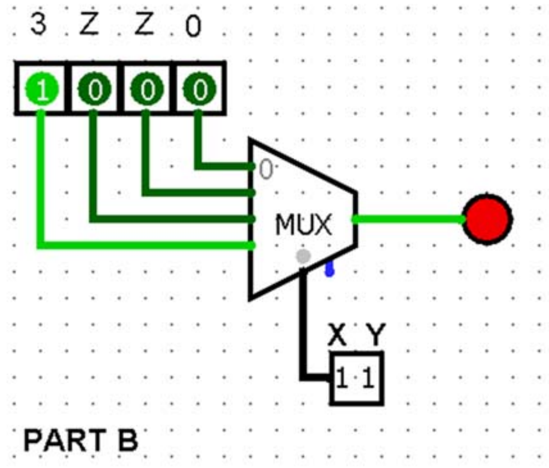
Truth Table:

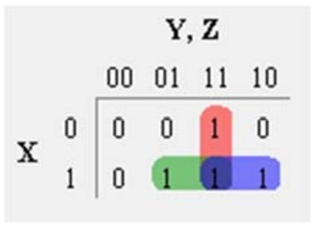
- Numbers on the left correspond to the inputs on the 4:1 multiplexer

Boolean Expression:

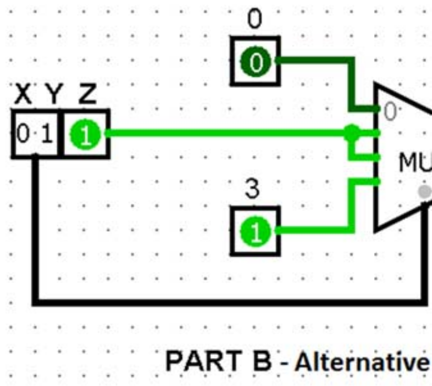
Y Z + X Z + X Y

Diagrams:





X \ Y, Z	00	01	11	10
0	0	0	1	0
1	0	1	1	1



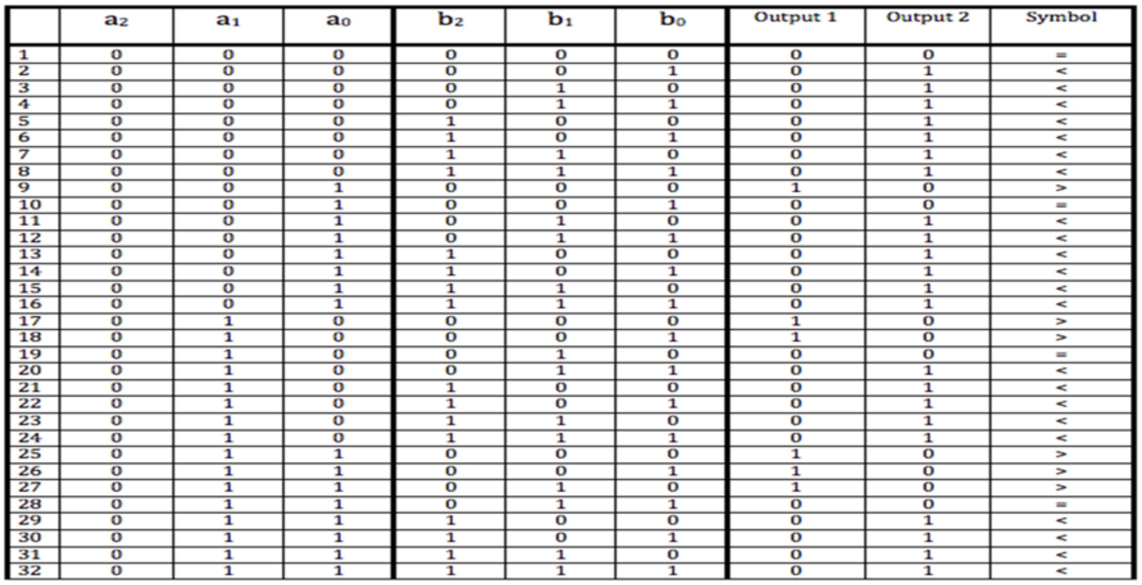
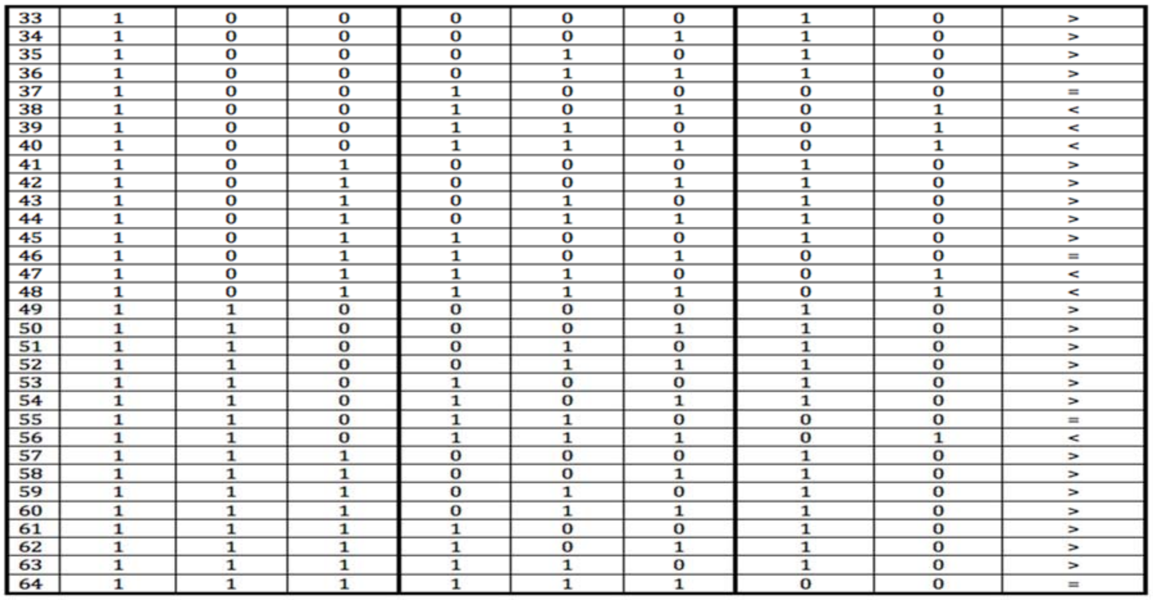
Part C:

Design a logic circuit that accepts two 3-bit numbers and generates different 2-bit outputs based on the scenarios below:

1. The 3-bit binary number a2a1a0 is greater than b2b1b0
2. The 3-bit binary number a2a1a0 is less than b2b1b0
3. The 3-bit binary number a2a1a0 is equal to b2b1b0

Truth Table:

< → 01 = → 00 **>** → 10

	a_2	a_1	a_0	b_2	b_1	b_0	Output 1	Output 2	Symbol
1	0	0	0	0	0	0	0	0	=
2	0	0	0	0	0	1	0	1	<
3	0	0	0	0	1	0	0	1	<
4	0	0	0	0	1	1	0	1	<
5	0	0	0	1	0	0	0	1	<
6	0	0	0	1	0	1	0	1	<
7	0	0	0	1	1	0	0	1	<
8	0	0	0	1	1	1	0	1	<
9	0	0	1	0	0	0	1	0	>
10	0	0	1	0	0	1	0	0	=
11	0	0	1	0	1	0	0	1	<
12	0	0	1	0	1	1	0	1	<
13	0	0	1	1	0	0	0	1	<
14	0	0	1	1	0	1	0	1	<
15	0	0	1	1	1	0	0	1	<
16	0	0	1	1	1	1	0	1	<
17	0	1	0	0	0	0	1	0	>
18	0	1	0	0	0	1	1	0	>
19	0	1	0	0	1	0	0	0	=
20	0	1	0	0	1	1	0	1	<
21	0	1	0	1	0	0	0	1	<
22	0	1	0	1	0	1	0	1	<
23	0	1	0	1	1	0	0	1	<
24	0	1	0	1	1	1	0	1	<
25	0	1	1	0	0	0	1	0	>
26	0	1	1	0	0	1	1	0	>
27	0	1	1	0	1	0	1	0	>
28	0	1	1	0	1	1	0	0	=
29	0	1	1	1	0	0	0	1	<
30	0	1	1	1	0	1	0	1	<
31	0	1	1	1	1	0	0	1	<
32	0	1	1	1	1	1	0	1	<
33	1	0	0	0	0	0	1	0	>
34	1	0	0	0	0	1	1	0	>
35	1	0	0	0	1	0	1	0	>
36	1	0	0	0	1	1	1	0	>
37	1	0	0	1	0	0	0	0	=
38	1	0	0	1	0	1	0	1	<
39	1	0	0	1	1	0	0	1	<
40	1	0	0	1	1	1	0	1	<
41	1	0	1	0	0	0	1	0	>
42	1	0	1	0	0	1	1	0	>
43	1	0	1	0	1	0	1	0	>
44	1	0	1	0	1	1	1	0	>
45	1	0	1	1	0	0	1	0	>
46	1	0	1	1	0	1	0	0	=
47	1	0	1	1	1	0	0	1	<
48	1	0	1	1	1	1	0	1	<
49	1	1	0	0	0	0	1	0	>
50	1	1	0	0	0	1	1	0	>
51	1	1	0	0	1	0	1	0	>
52	1	1	0	0	1	1	1	0	>
53	1	1	0	1	0	0	1	0	>
54	1	1	0	1	0	1	1	0	>
55	1	1	0	1	1	0	0	0	=
56	1	1	0	1	1	1	0	1	<
57	1	1	1	0	0	0	1	0	>
58	1	1	1	0	0	1	1	0	>
59	1	1	1	0	1	0	1	0	>
60	1	1	1	0	1	1	1	0	>
61	1	1	1	1	0	0	1	0	>
62	1	1	1	1	0	1	1	0	>
63	1	1	1	1	1	0	1	0	>
64	1	1	1	1	1	1	0	0	=

Kmaps:

Output _ 1

$b_2b_1b_0$ / $a_2a_1a_0$	000	001	011	010	100	101	111	110
000	0	0	0	0	0	0	0	0
001	1	0	0	0	0	0	0	0
011	1	1	0	1	0	0	0	0
010	1	1	0	0	0	0	0	0
100	1	1	1	1	0	0	0	0
101	1	1	1	1	1	0	0	0
111	1	1	1	1	1	1	0	1
110	1	1	1	1	1	1	0	0

Output _ 2

$b_2b_1b_0$ / $a_2a_1a_0$	000	001	011	010	100	101	111	110
000	0	1	1	1	1	1	1	1
001	0	0	1	1	1	1	1	1
011	0	0	0	0	1	1	1	1
010	0	0	1	0	1	1	1	1
100	0	0	0	0	0	1	1	1
101	0	0	0	0	0	0	1	1
111	0	0	0	0	0	0	0	0
110	0	0	0	0	0	0	1	0

Boolean Expressions:

Output bit 1:

(A0 ~B2 ~B1 ~B0) + (A1 ~B2 ~B1) + (A1 A0 ~B2 ~B0) + (A2 ~B2) + (A2 A0 ~B1 ~B0) + (A2 A1 ~B1) + (A2 A1 A0 ~B0)

Output bit 2:

(~A2 ~A1 ~A0 B0) + (~A2 ~A1 B1) + (~A2 ~A0 B1 B0) + (~A2 B2) + (~A1 ~A0 B2 B0) + (~A1 B2 B1) + (~A0 B2 B1 B0)

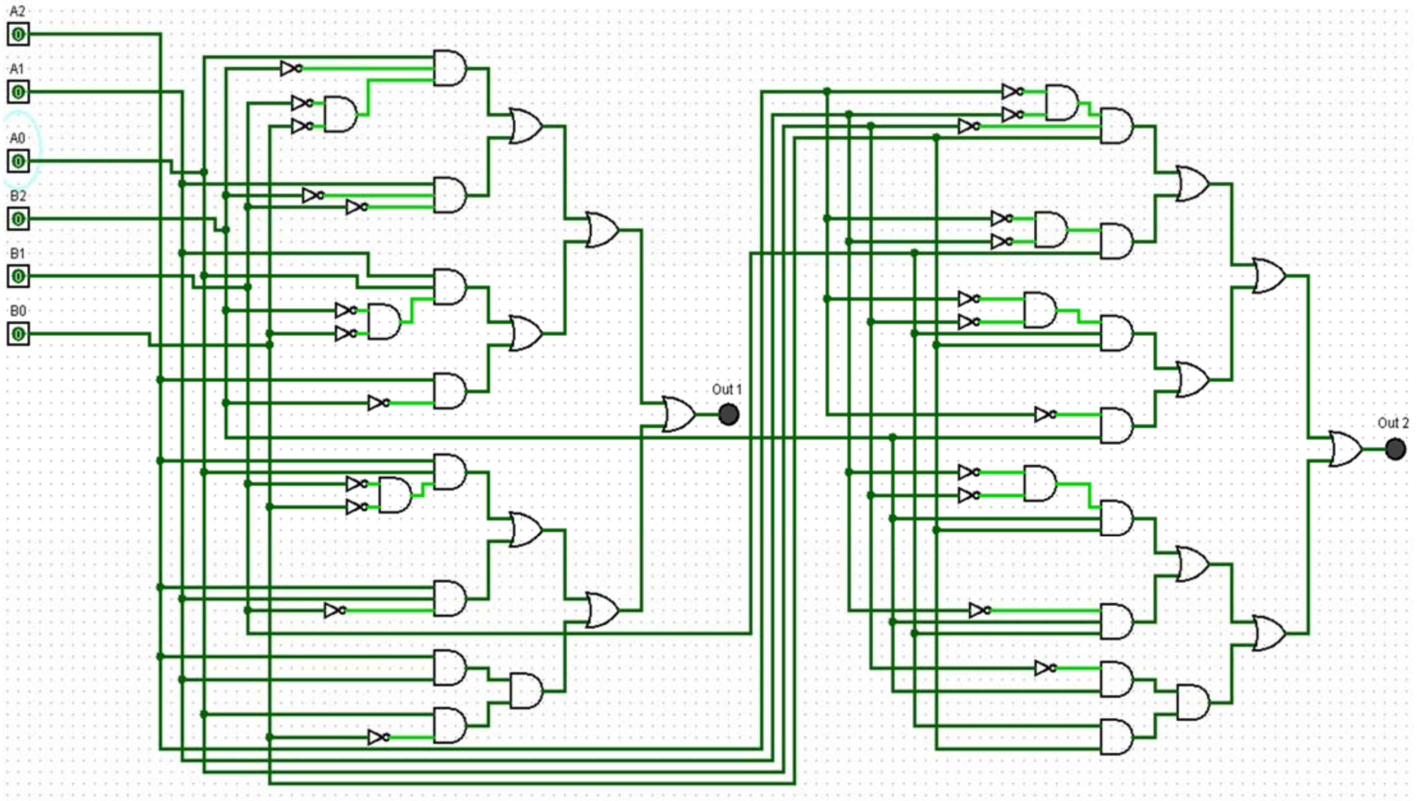
Test Cases:

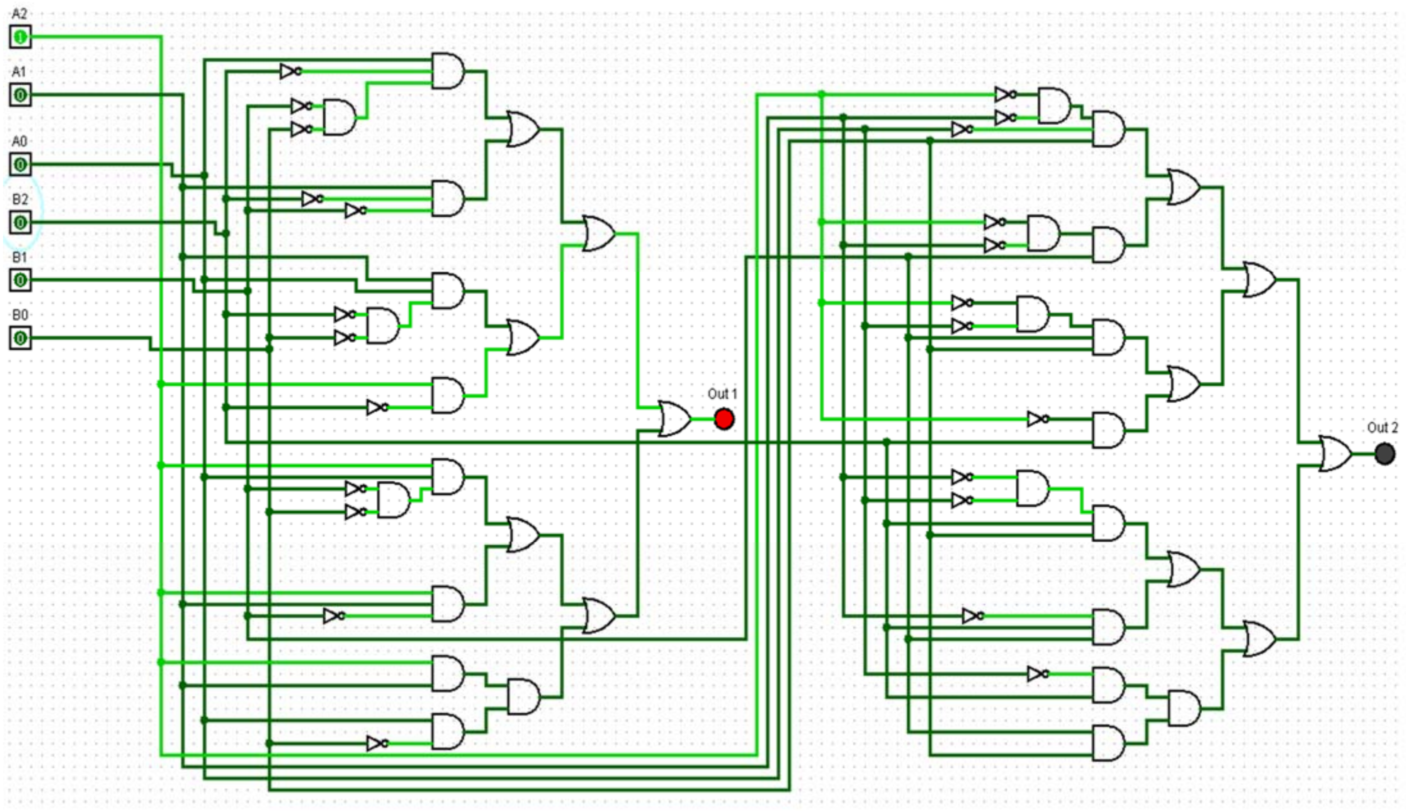
a2a1a0 = 001 ----- & ------- b2b1b0 = 010 ; a2a1a0 < b2b1b0 → Out_1 = 0, Out_2= 1

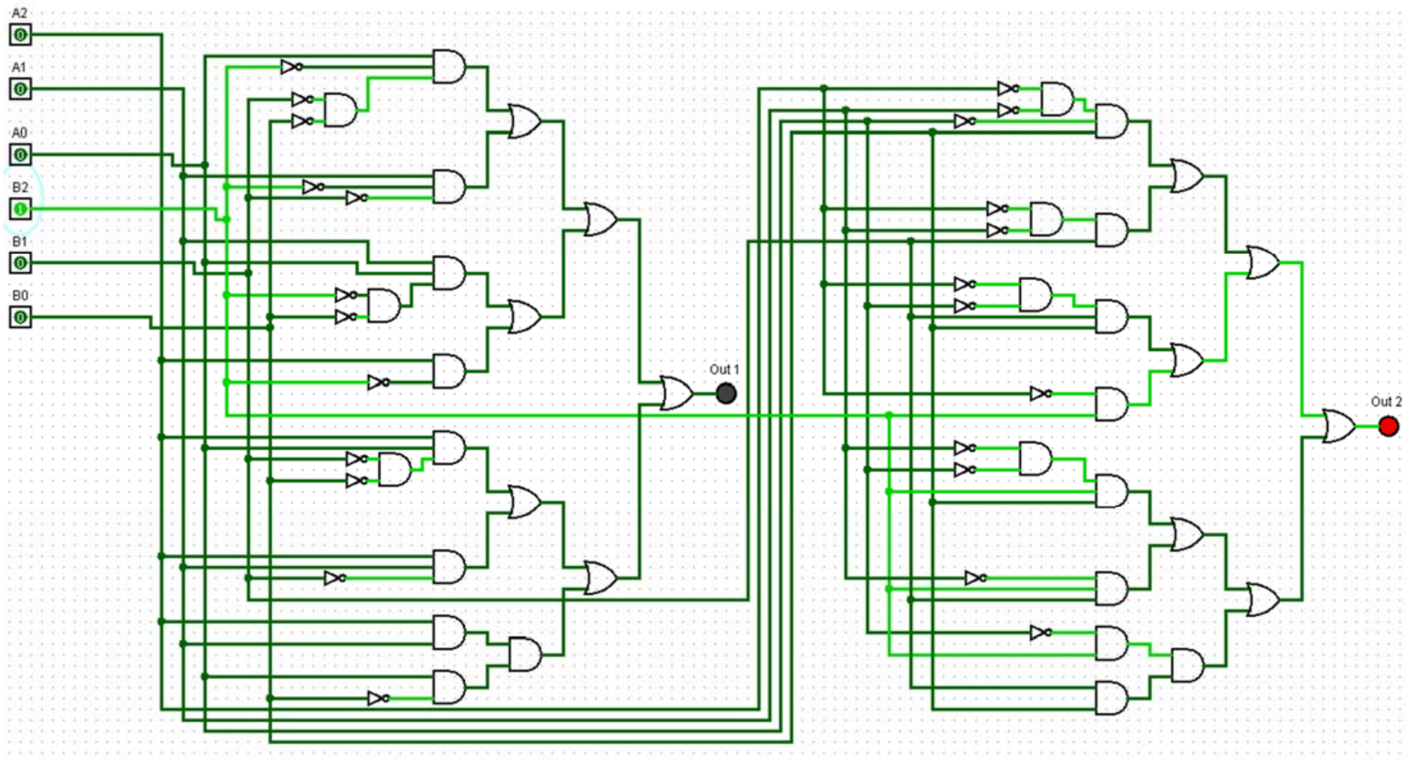
a2a1a0 = 010 ----- & ------- b2b1b0 = 010 ; a2a1a0 = b2b1b0 → Out_1 = 0, Out_2= 0

a2a1a0 = 011 ----- & ------- b2b1b0 = 010 ; a2a1a0 > b2b1b0 → Out_1 = 1, Out_2= 0

Diagrams: Cases **equal**, **greater than** and **less than** examples shown below







Discussion & Conclusion:

Part A was a simple exercise of understanding how the 8:1 multiplexer is laid out. Once we understood how how it worked we implemented the Boolean expression using the 8:1 mux. We ran into issues with the multiplexer not outputting the correct result even though all lines were wired correctly. The issue came down to Pin# 7 labeled as the Enabler input. Once we properly connected that input to low, the mux worked as intended.

Part B required us to implement the same Boolean expression using a single 4:1 multiplexer. The technique behind making this work required examining two inputs and how it related to a function of the output with respect to the remaining 3rd input (Z) that was lost when moving from an 8:1 mux to a 4:1. The function of said output with respect to Z showed that 2 of the 4 inputs on the mux were going to be variable inputs. As can be seen in our diagrams, inputs I_1 and I_2 were connected to variable input Z while input I_0 was hardwired to low and input I_3 was hardwired to high. This allowed us to represent the same expression using only a single 4:1 multiplexer.

Part C involved the designing of a comparator system. The logic design was based around performing relational comparisons between 2 3-bit numbers. Partway through implementing our design we came to the realization that another approach could be utilized. The alternative approach would entail comparing each input bit, from most significant to least. Once a comparison is found to be equal the next inputs would be evaluated until the least significant bit is compared. The alternative logic would require the use a few XOR or XNOR gates for the initial equality test and could have simplified our design. Our design began with a 64 row truth table and 2-bit output that allowed us to represent the relational operators. After the truth table was filled we proceeded to extract two Boolean expressions, one for each output bit. The two expressions were simplified and then implemented as a logic design on our breadboard. As no restrictions were made on how many gates could be used, we chose to implement it with AND, OR and Inverter gates.