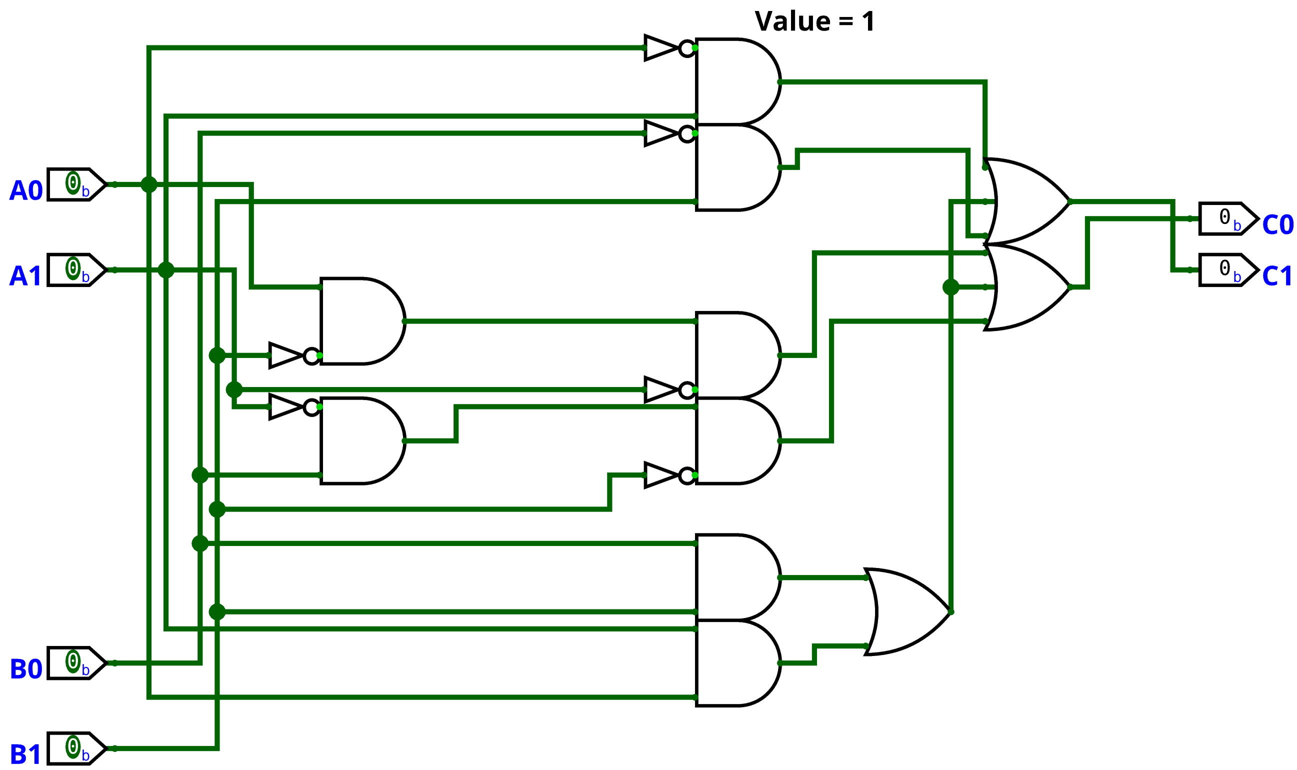
Lab 3 - Combinational Design

- Primary objectives:
  - Design and implement a multi-level AND-OR-NOT circuit for a simple 2-bit priority selector
    - Start with a natural language description
    - Then design, simulate, test and implement
- Device description and function:
  - Uses two 2-bit inputs representing two values ranging from 0 to 3
    - Value A is represented by two input variables A1 & A0
    - Value B is represented by two input variables B1 & B0
  - Uses a 2-bit output C will output the greater of the two input values
    - Value C is represented by C1 & C0
- Table #1: I/O Bit to Base-10 Conversion

A1	**A0**	**Base-10 Value**	**B1**	**B0**	**Base-10 Value**	**C1**	**C0**	**Base-10 Value**
0	0	0	0	0	0	0	0	0
0	1	1	0	1	1	0	1	1
1	0	2	1	0	2	1	0	2
1	1	3	1	1	3	1	1	3

- Figure #1 & #2: Initial Circuit Design & Truth Table



A	B		C	D		E	F	
A1	A0	Base-10 A	B1	B0	Base-10 B	C1	C0	Base-10 C
0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	1	1
0	0	0	1	0	2	1	0	2
0	0	0	1	1	3	1	1	3
0	1	1	0	0	0	0	1	1
0	1	1	0	1	1	0	1	1
0	1	1	1	0	2	1	0	2
0	1	1	1	1	3	1	1	3
1	0	2	0	0	0	1	0	2
1	0	2	0	1	1	1	0	2
1	0	2	1	0	2	1	0	2
1	0	2	1	1	3	1	1	3
1	1	3	0	0	0	1	1	3
1	1	3	0	1	1	1	1	3
1	1	3	1	0	2	1	1	3
1	1	3	1	1	3	1	1	3

0		
A0	B0	C0
0	0	0
0	0	0
0	1	1
0	1	1
0	0	0
0	0	0
0	1	1
0	1	1
1	0	1
1	0	1
1	1	1
1	1	1
1	0	1
1	0	1
1	1	1
1	1	1

Behaviour for 0 bit's:
Relies on OR logic
A0 + B0 = C0

1		
A1	B1	C1
0	0	0
0	1	1
0	0	0
0	1	1
1	0	1
1	1	1
1	0	0
1	1	1
0	0	0
0	1	0
0	0	0
0	1	1
1	0	1
1	1	1
1	0	1
1	1	1

Behaviour for 1 bit's
Appears to use same OR logic except for two cases

- Note: This circuit functioned as intended and matched the values of the truth table. Upon further evaluation of the truth tables for this circuit, I noticed that this circuit is overly complicated for the task and can be significantly simplified.

- Table #2: Circuit Behavioral Truth Table

A1	A0	B1	B0	C1	C0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	1

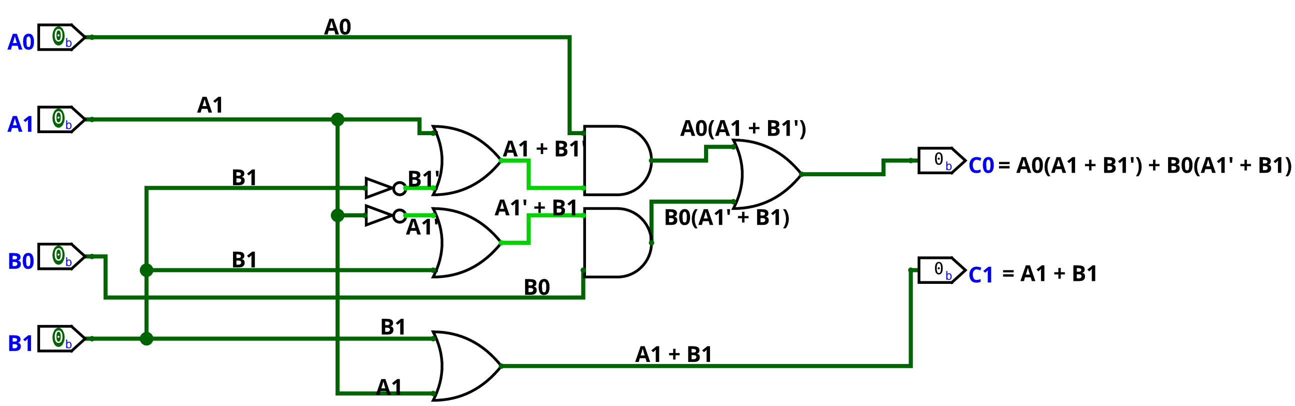
A1	A0	B1	B0	C1	C0
0	1	1	1	1	1
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	1	0
1	0	1	1	1	1
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1

- Figure #3 & #4: K-Maps for C1 & C0 (Note: A = A1, B = A0, C = B1, D = B0)
-

C1		0	1	3	2
		C'D'	C'D	CD	CD'
0	A'B'	00	00	**01**	**01**
1	A'B	00	00	01	00
3	AB	01	01	01	01
2	AB'	01	00	**01**	**01**

-

C0		0	1	3	2
		C'D'	C'D	CD	CD'
0	A'B'	00	01	01	00
1	A'B	01	01	01	01
3	AB	01	01	01	01
2	AB'	00	01	01	00

- SimplifiedBoolean Expressions for Truth Table
  - C1 = A0 + B0
  - C0 = B1B0 + (A0'B1) + A1A0 + (A1B0') = A1(A0 + B0') + B1(A0' + B0)
- Figure #5: Circuit Version 2
- 

- Conclusion & Testing:
  - This circuit turned out to be much simpler than my initial version, and it truth values were identical to the desired truth values for this circuit