X	Y	x
0	0	0
0	1	1
1	0	1
1	1	0

Figure 1: XOR gate

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Figure 2: HalfAdder

A	B	C2	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Figure 3: FullAdder

A	B	Sel	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Figure 4: 2-Mux

A	B	C	D	Sel0	Sel1	S
0	0	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
0	0	0	1	0	1	0
0	0	0	1	1	0	0
0	0	0	1	1	1	1
0	0	1	0	0	0	0
0	0	1	0	0	1	1
0	0	1	0	1	0	0
0	0	1	0	1	1	0
0	0	1	1	0	0	0
0	0	1	1	0	1	1
0	0	1	1	1	0	0
0	0	1	1	1	1	1
0	1	0	0	0	0	0
0	1	0	0	0	1	0
0	1	0	0	1	0	1
0	1	0	0	1	1	0
0	1	0	1	0	0	0
0	1	0	1	0	1	0
0	1	0	1	1	0	1
0	1	0	1	1	1	1
0	1	1	0	0	0	0
0	1	1	0	0	1	1
0	1	1	0	1	0	1
0	1	1	0	1	1	0
0	1	1	1	0	0	0
0	1	1	1	0	1	1
0	1	1	1	1	0	1
0	1	1	1	1	1	1
1	0	0	0	0	0	1
1	0	0	0	0	1	0
1	0	0	0	1	0	0
1	0	0	0	1	1	0
1	0	0	1	0	0	1
1	0	0	1	0	1	0
1	0	0	1	1	0	0
1	0	0	1	1	1	1
1	0	1	0	0	0	1

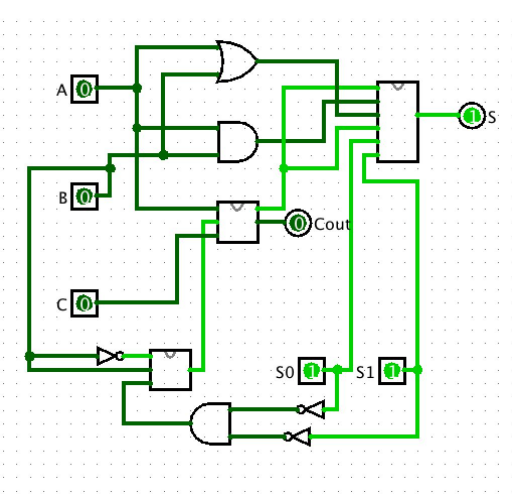
Figure 5: 4-mux (some got cut off)

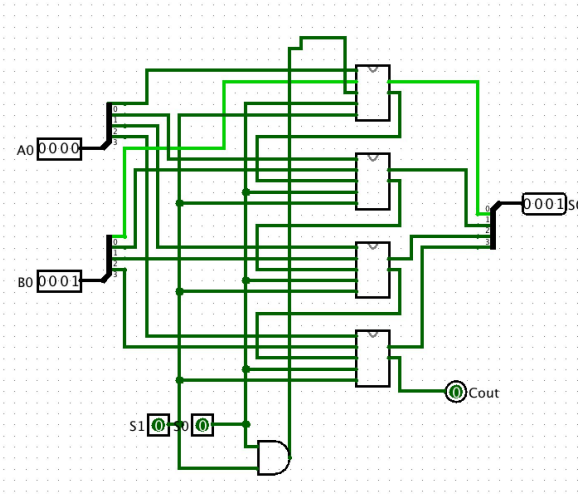
A	B	C	S0	S1	S	Cout
0	0	0	0	0	0	0
0	0	0	0	1	0	0
0	0	0	1	0	0	0
0	0	0	1	1	1	0
0	0	1	0	0	1	0
0	0	1	0	1	0	1
0	0	1	1	0	0	1
0	0	1	1	1	0	1
0	1	0	0	0	1	0
0	1	0	0	1	1	0
0	1	0	1	0	0	0
0	1	0	1	1	0	0
0	1	1	0	0	0	1
0	1	1	0	1	1	0
0	1	1	1	0	0	0
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	0	0	1	1	1
1	0	0	1	0	0	1
1	0	0	1	1	0	1
1	0	1	0	0	0	1
1	0	1	0	1	1	1
1	0	1	1	0	0	1
1	0	1	1	1	1	1
1	1	0	0	0	0	1
1	1	0	0	1	1	0
1	1	0	1	0	1	0
1	1	0	1	1	1	0
1	1	1	0	0	1	1
1	1	1	0	1	1	1
1	1	1	1	0	1	1
1	1	1	1	1	0	1

Figure 6: 1-Bit ALU

In order to make sure my Full ALU properly worked, I inputted several different 4-bit numbers to both A and B. I then went through every possible Select combination to not only check to make sure addition worked properly, but also subtraction. Once I was satisfied with each Select options output, I was sure my circuit was working properly.

In order to prove the results of my Full 4-bit ALU, here are diagrams of the 1 and 4 bit respectively, including how to use s1 and s2 to perform subtraction.


A
B
C
Cout
S
S0
S1


A0 0000
B0 0001
0001
Cout
S1
S0