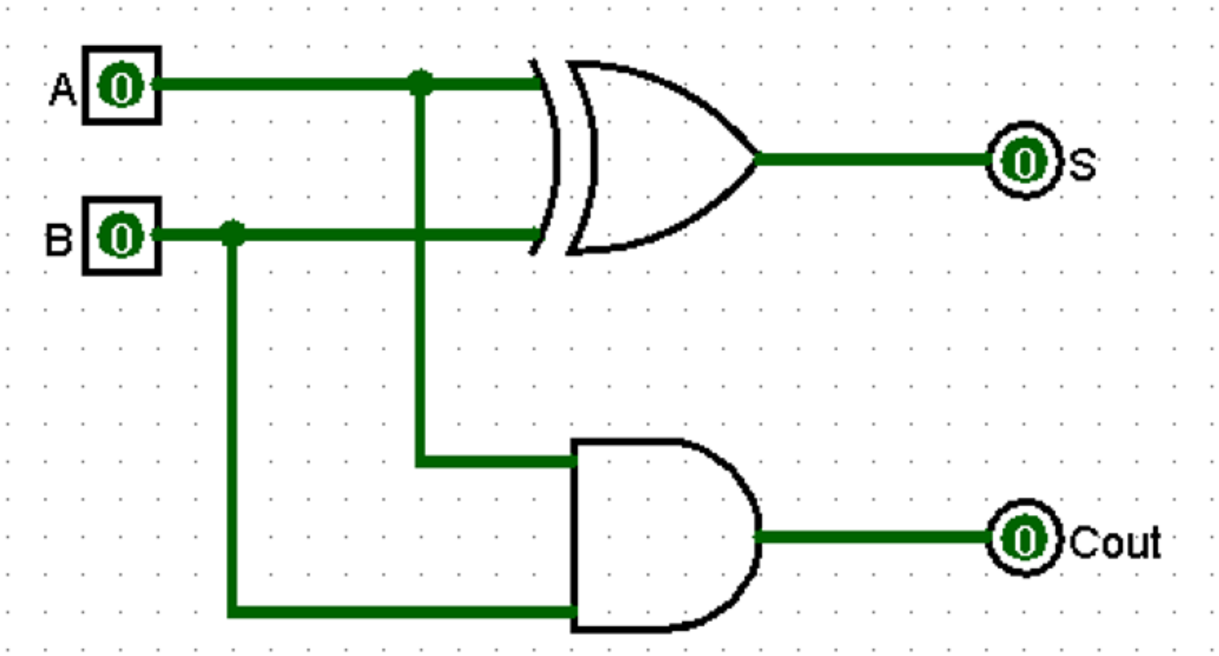
LAB 2

Half Adder

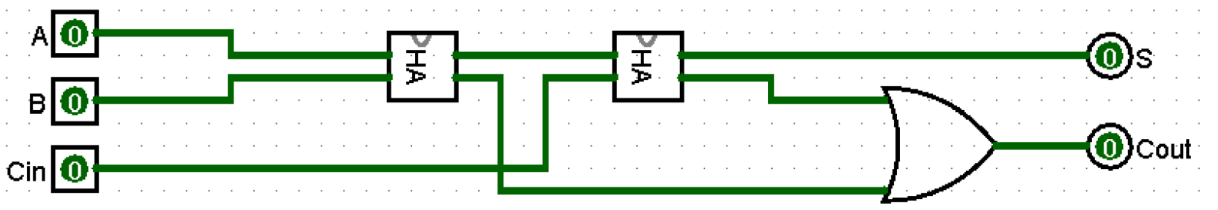


Testing the Half Adder via truth tables:

A	B	Cout	S
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	0

The above circuit was manually tested using the truth table above.

Full Adder



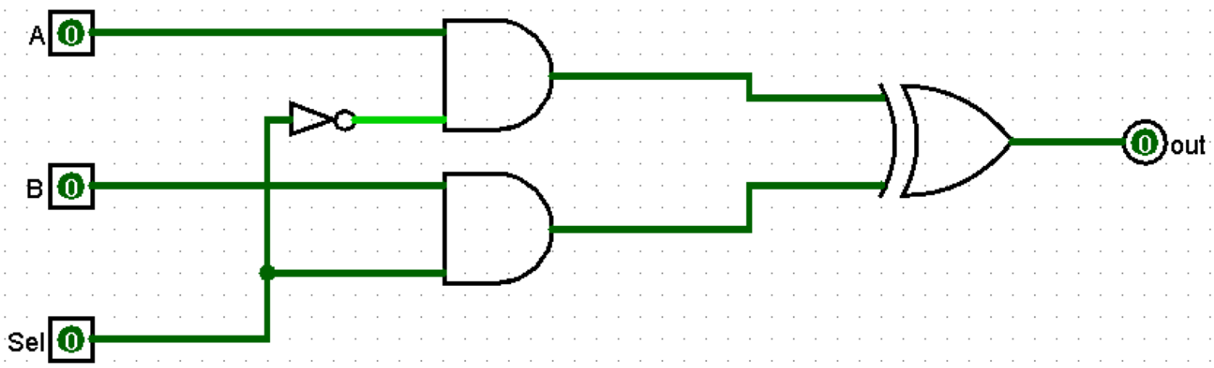
Testing the Full Adder

Cin	A	B	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

The Full Adder was built using 2 half adders instead of doing an entirely new circuit.

The circuit was tested using the above truth table.

2-MUX

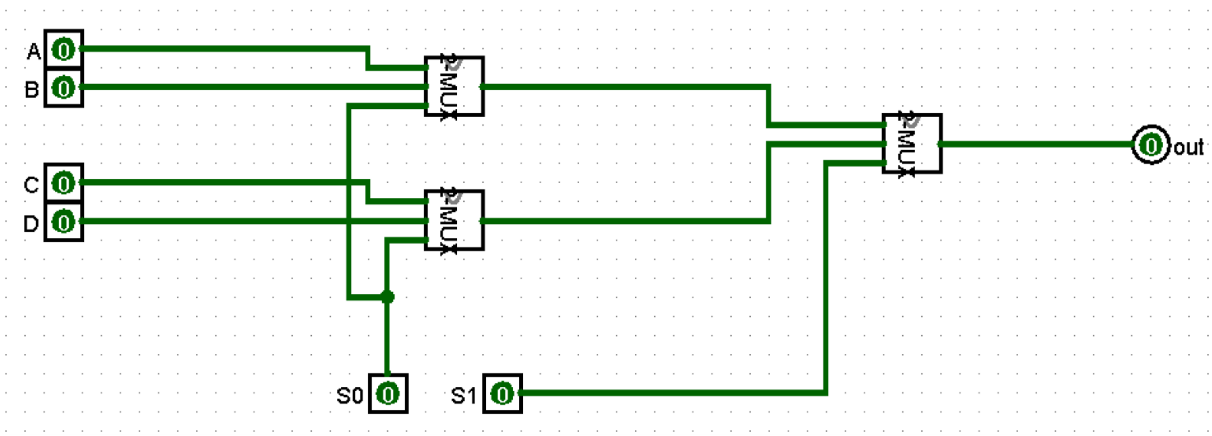


Testing the 2-MUX

A	B	Sel	out
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

The circuit was tested using the above truth table.

4-mux



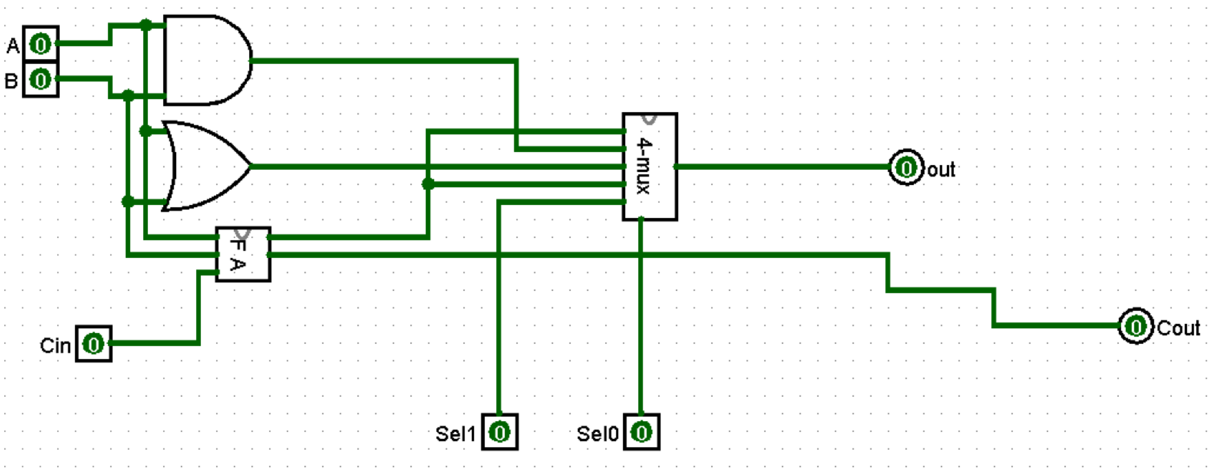
Testing the 4-mux

A	B	C	D	S0	S1	out
1	0 or 1	0 or 1	0 or 1	0	0	1
0 or 1	1	0 or 1	0 or 1	1	0	1
0 or 1	0 or 1	1	0 or 1	0	1	1
0 or 1	0 or 1	0 or 1	1	1	1	1

The 4-mux was built using the three 2-mux switches.

The circuit was tested using the truth table above.

1-Bit ALU



Testing the 1-bit ALU

OP code: S1 = 0 S0 = 0 : addition

A	B	C-in (should always be 0)	out	Cout
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1

OP code: S1 = 0 S0 = 1 : AND

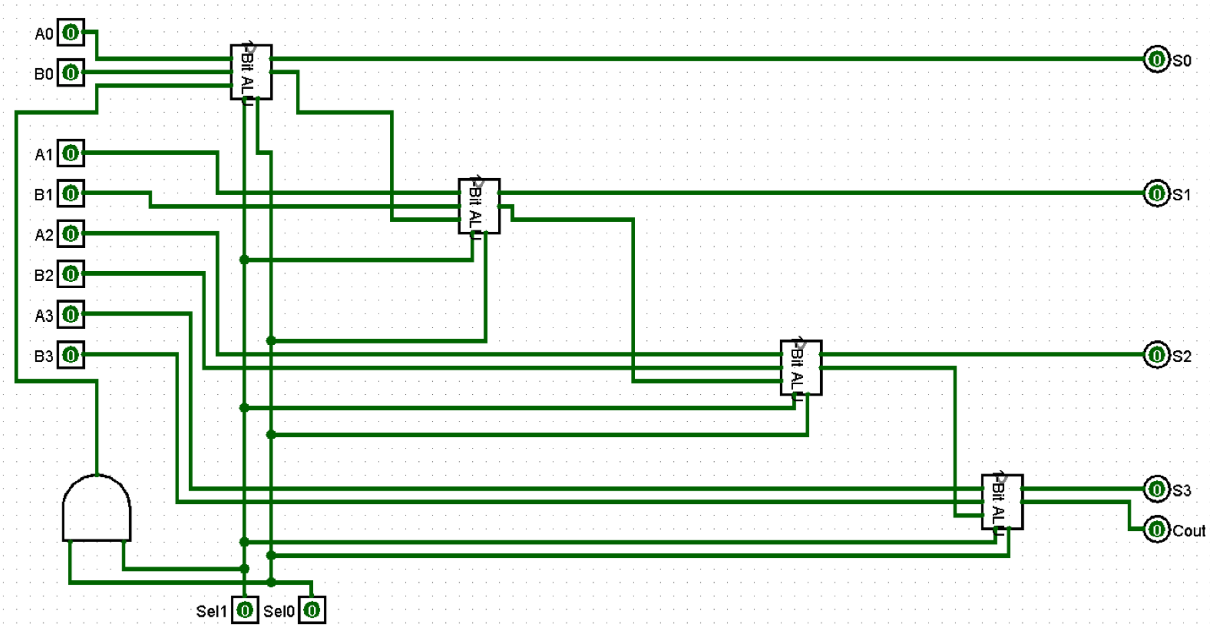
A	B	out
0	0	0
0	1	0
1	0	0
1	1	1

OP code: S1 = 1 S0 = 0 : OR

A	B	out
0	0	0
0	1	1
1	0	1
1	1	1

The 1-bit ALU was made using the Full adder and a 4-way mux switch and tested using the above truth tables. The tables above are of inconsistent size as inputs and outputs that are not important for a particular function are not shown. For example, the Cout is not an important output for AND and OR functions hence is not shown.

4-bit Ripple ALU



Here, the circuit has been constructed by using four 1-bit ALUs. The Cin input is removed and now directly determined by the OP code such that Cin =1 iff S1 = S0 = 1.

Testing the 4-bit Ripple ALU

The above circuit was tested using the values given in the truth tables below. The tables above are of inconsistent size as inputs and outputs that are not important for a particular function are not shown. For example, the Cout is not an important output for AND and OR functions hence is not shown.

OP code: S1 = 0 S0 = 0 : addition

A3	A2	A1	A0	B3	B2	B1	B0	S3	S2	S1	S0	Cout
0	0	0	1	0	0	0	1	0	0	1	0	0
0	0	1	1	0	0	0	1	0	1	0	0	0
1	1	0	0	0	1	0	1	0	0	0	1	1
1	0	1	1	0	1	0	1	0	0	0	0	1

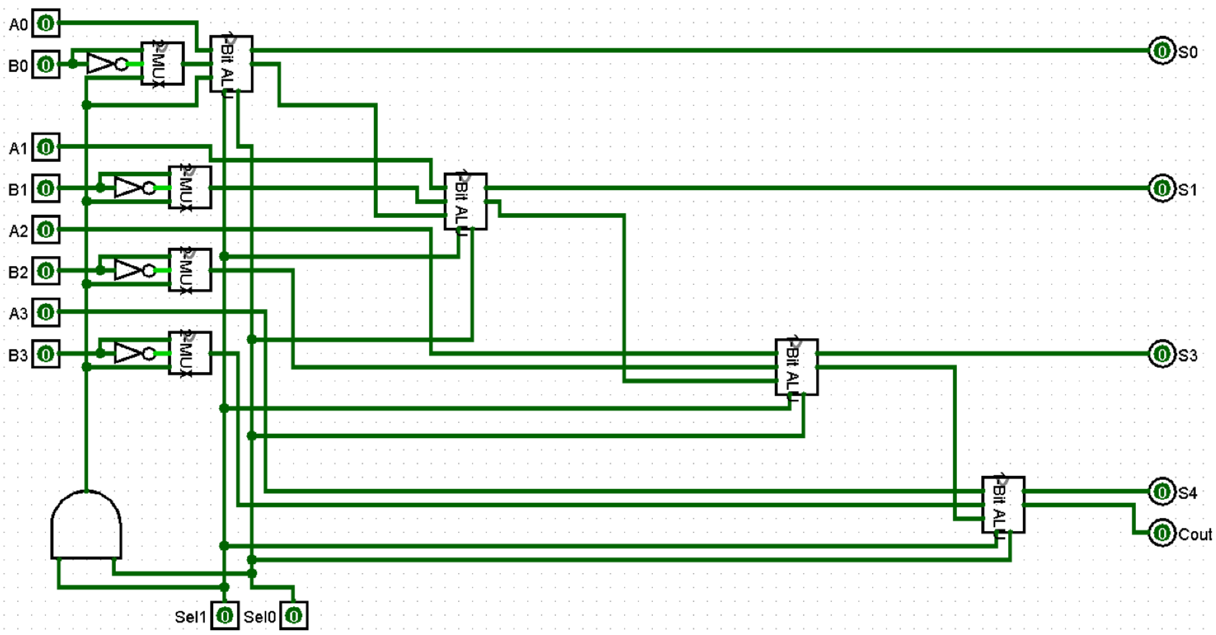
OP code: S1 = 0 S0 = 1 : AND

A3	A2	A1	A0	B3	B2	B1	B0	S3	S2	S1	S0
0	0	0	1	0	0	0	1	0	0	0	1
0	0	1	1	0	0	0	1	0	0	0	1
1	1	0	0	0	1	0	1	0	1	0	0
1	0	1	1	0	1	0	1	0	0	0	1

OP code: S1 = 1 S0 = 0 : OR

A3	A2	A1	A0	B3	B2	B1	B0	S3	S2	S1	S0
1	1	1	1	0	0	0	1	1	1	1	1
0	0	1	1	0	0	0	1	0	0	1	1
1	1	0	0	0	1	0	1	1	1	0	1
1	0	1	1	1	1	0	1	1	1	1	1

Full ALU



The only difference between the Full 4-Bit ALU and the 4-Bit Ripple ALU is that the functionality of subtraction is added. Everything else remains the same as the 4-Bit Ripple ALU.

Testing the 4-bit Full ALU

Since everything except subtraction was same as the 4-bit ripple ALU, please refer to the truth tables in the testing section of the 4-bit Ripple ALU for tests pertaining to functions Addition, AND and OR

OP code: S1 = 1 S0 = 1 : subtraction

A3	A2	A1	A0	B3	B2	B1	B0	S3	S2	S1	S0
0	0	0	1	0	0	0	1	0	0	0	0
1	1	1	1	0	0	0	1	1	1	1	0
1	1	0	0	0	1	0	1	0	1	1	1
1	0	1	1	0	1	0	1	0	1	1	0

The way I tested my 4-bit Full ALU and Ripple ALU is by going back and fourth between smaller components and testing them individually. After this, I made sure that the OP codes for each operation were correct and the correct wire were in the correct input pin by matching labels.

The tests that I used are mentioned above. These output values for all the circuits were calculated by hand first and then the values within the circuit were changed. The output of the circuit matched the values calculated by hand.