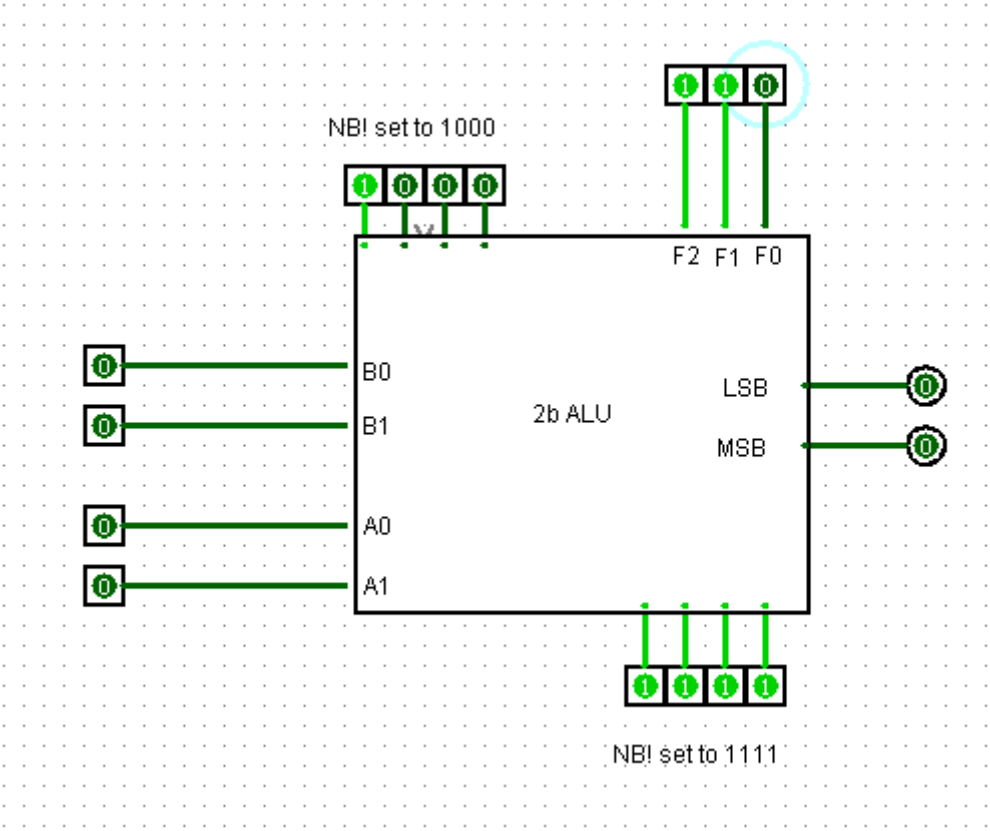
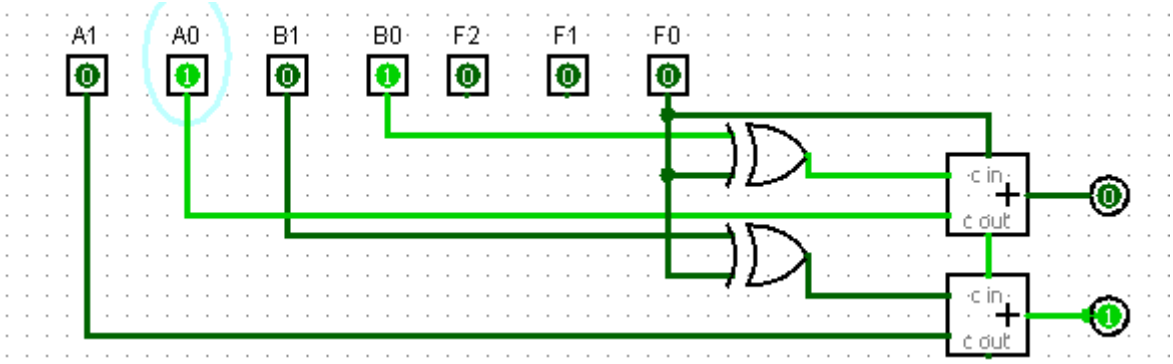
PRAC_5_LPHTUM003_JHRSHA002



ADDER SUBTRACTOR COMPONENT

Addition and subtraction operations are combined into a single circuit. The use of two 1-bit XOR gates allow for the circuit to change from simply computing $A + B$ to $A + \bar{B} + 1$. These operations are activated by the opcodes: $F_2 \rightarrow 0 : F_1 \rightarrow 0 : F_1 \rightarrow 0$ and $F_2 \rightarrow 0 : F_1 \rightarrow 0 : F_1 \rightarrow 1$ respectively.

ADDITION

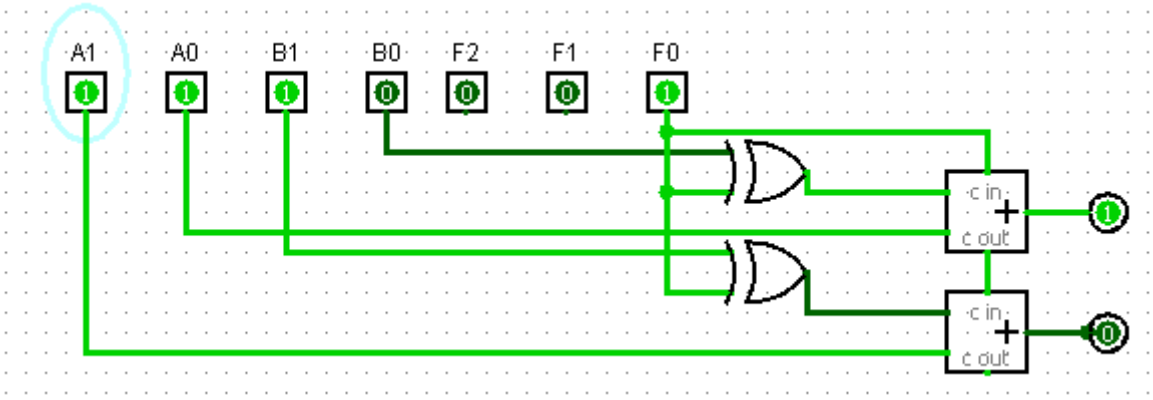


Addition was calculated using two 1-bit full adders the truth table below represents how a single adder operates.

X_1	X_0	C_{IN}	C_{OUT}	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**

SUBTRACTION



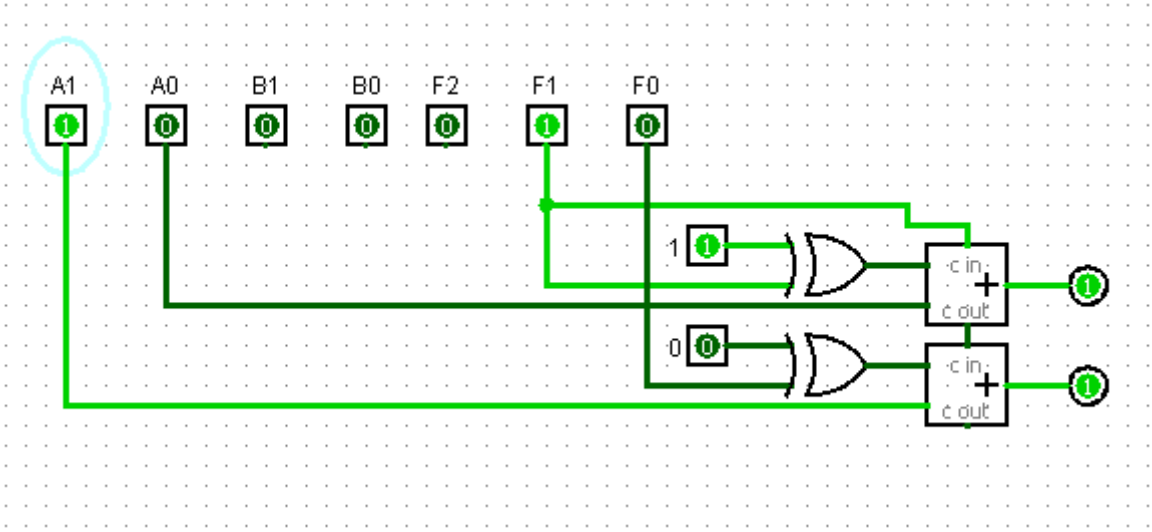
Subtraction was calculated using two 1-bit full adders and the use of XOR gates to add a constant 1 from the opcode as well as the complement of B. The truth table below represents how a single adder operates when it is used as a subtractor.

X_1	X_0	B_{IN}	D	B_{OUT}
0	**0**	**0**	**0**	**0**
0	**0**	**1**	**1**	**1**
0	**1**	**0**	**1**	**1**
0	**1**	**1**	**0**	**1**
1	**0**	**0**	**1**	**0**
1	**0**	**1**	**0**	**0**
1	**1**	**0**	**0**	**0**
1	**1**	**1**	**1**	**1**

INCREMENT AND DECREMENT

Increment and decrement operations are combined into a single circuit. The use of two XOR gates allow for the circuit to change from simply computing $A + 1$ to $A + \bar{1} + 1$. These operations are activated by the opcodes: $F_2 \rightarrow 0 : F_1 \rightarrow 1 : F_1 \rightarrow 0$ and $F_2 \rightarrow 0 : F_1 \rightarrow 1 : F_0 \rightarrow 1$ respectively.

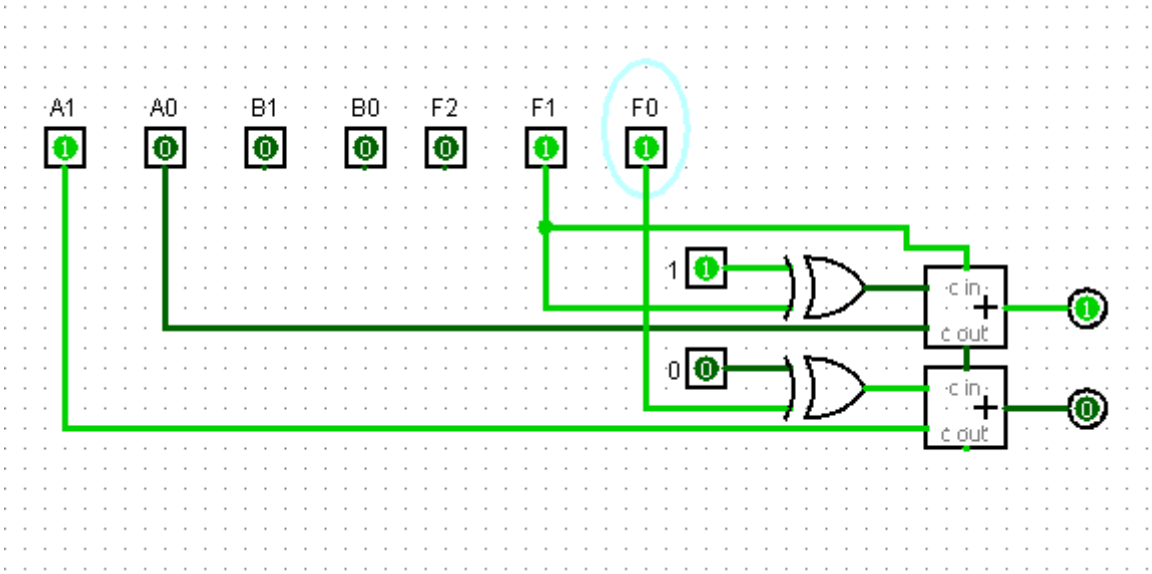
INCREMENT



Increment is achieved by using 2 1-bit full adders and having a constant 1 added to each A bit the truth table below represents how a single adder operates in these conditions.

A_1	A_0	1	C_{OUT}	S
0	**0**	**1**	**0**	**1**
0	**1**	**1**	**1**	**0**
1	**1**	**1**	**0**	**0**
1	**0**	**1**	**1**	**1**

DECREMENT



Decrement is achieved by using 2 1-bit full adders and the use of XOR gates to add a constant 1 from the opcode as well as the complement of 1. The truth table below represents how a single adder operates when it is used as a decrementor.

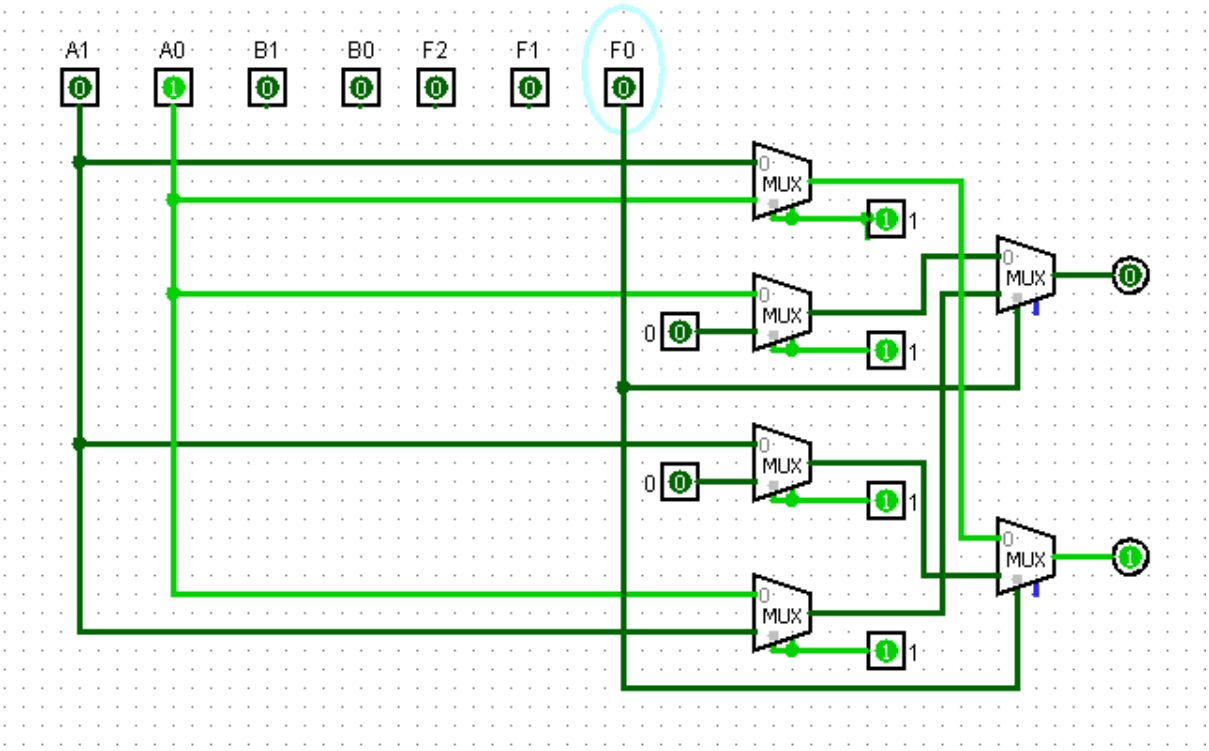
A_1	A_0	1	D	B_{OUT}
0	**0**	**1**	**1**	**1**
0	**1**	**1**	**0**	**0**
1	**1**	**1**	**0**	**0**

1	**0**	**1**	**1**	**0**

MULTIPLY AND DIVIDE

Multiplication and division operations are combined into a single circuit. A 2-bit logical shifter can be constructed using two 1-bit multiplexers. Six 1-bit multiplexers were used to choose between $A \ll 1$ and $A \gg 1$. These operations are activated by the opcodes: $F_2 \to 1 : F_1 \to 0 : F_1 \to 0$ and $F_2 \to 1 : F_1 \to 0 : F_1 \to 1$ respectively.

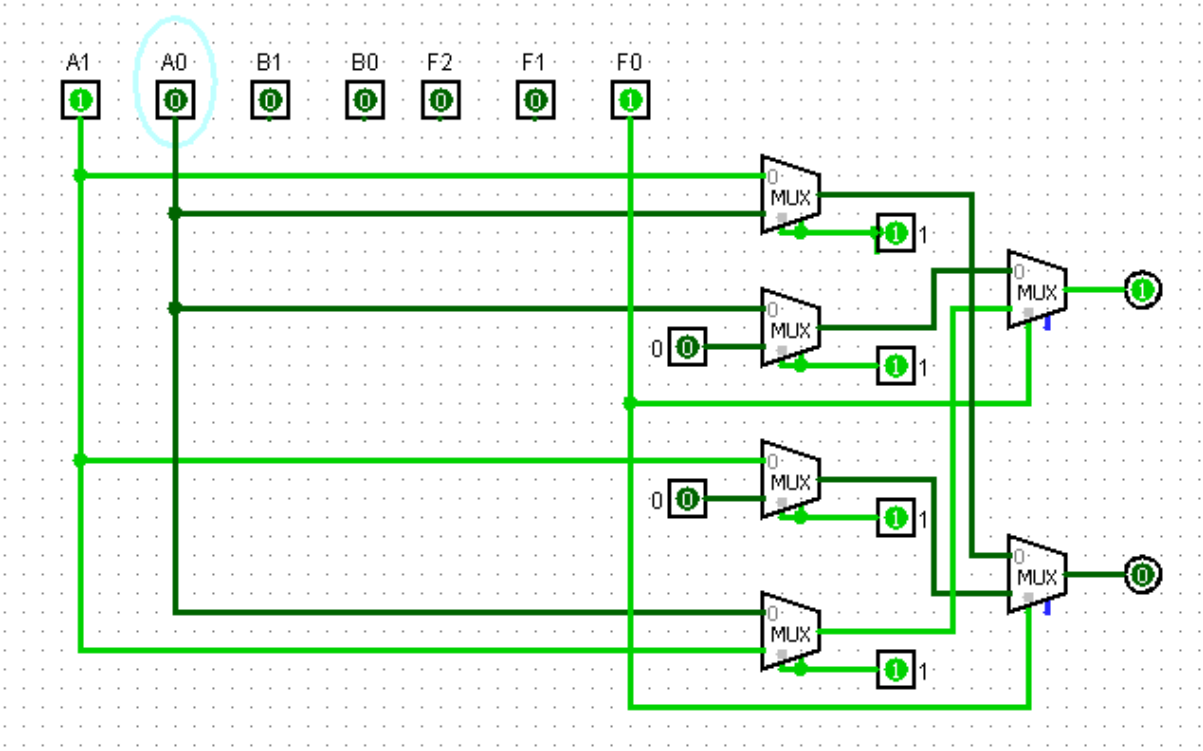
MULTIPLICATION



Multiplying by 2 required the combination of six 1-bit multiplexers whereby the last pair of multiplexers have a select of 0. The truth table below represents the results of multiplying by 2

A_1	A_0	Y_1	Y_0
0	**0**	**0**	**0**
0	**1**	**1**	**0**
1	**0**	**0**	**0**
1	**1**	**1**	**0**

DIVISION



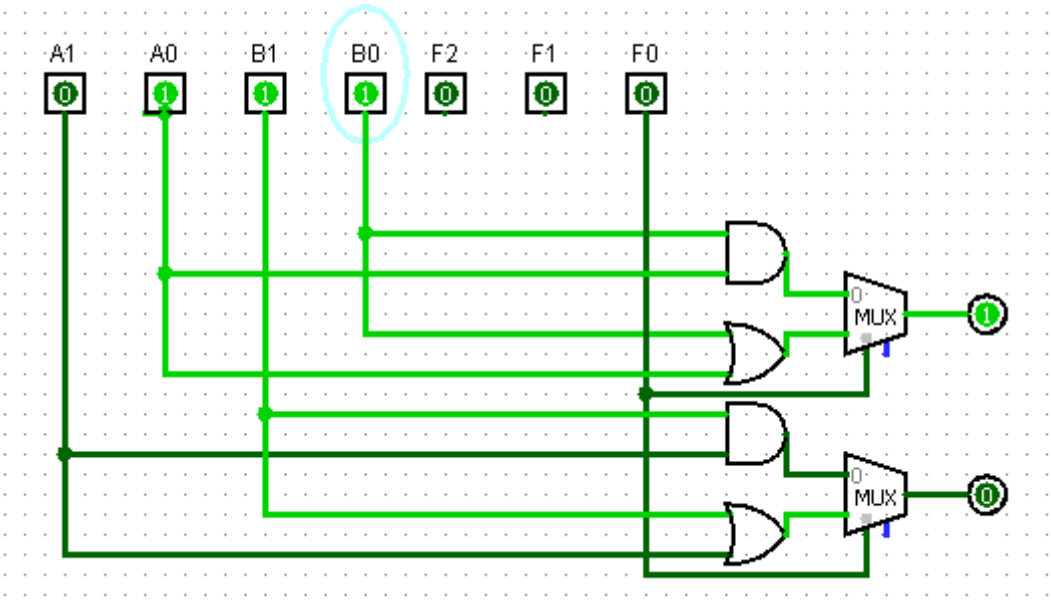
Dividing by 2 required the combination of 6 1-bit multiplexers whereby the last pair of multiplexers have a select of 1. The truth table below represents the results of dividing by 2

A_1	A_0	Y_1	Y_0
0	**0**	**0**	**0**
0	**1**	**1**	**0**
1	**0**	**0**	**0**
1	**1**	**1**	**0**

BITWISE OPERANDS

Bitwise AND and OR operations are combined into a single circuit. For each bit from A and B we apply both AND and OR operations and have a select line choose which operation to apply to them using a 1-bit multiplexer. These operations are activated by the opcodes: $F_2 \rightarrow 1 : F_1 \rightarrow 1 : F_1 \rightarrow 0$ and $F_2 \rightarrow 1 : F_1 \rightarrow 1 : F_1 \rightarrow 1$ respectively.

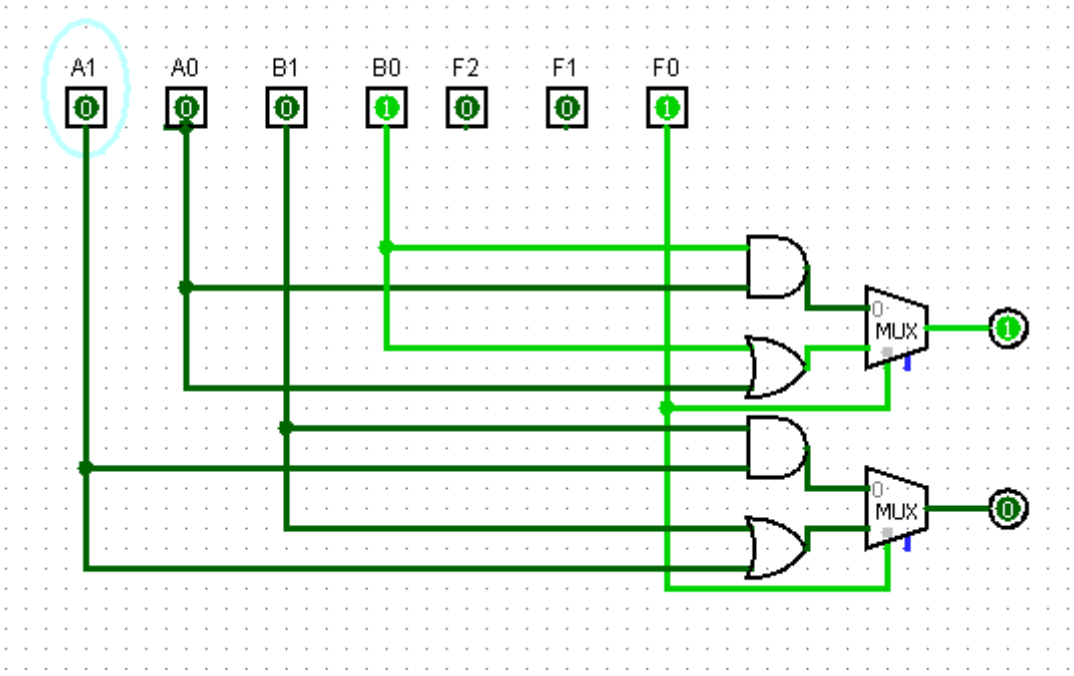
BITWISE AND



Logical AND is applied to each bit and select is 0. The truth table below represents bitwise A AND B

X_1	X_0	Y
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

BITWISE OR



Logical OR is applied to each bit and select is 1. The truth table below represents bitwise A OR B

X_1	X_0	Y
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**