Credit Hour System

CMPN101

Cairo University

Faculty of Engineering

Logic Project: Team 13

Team Members:

Name	ID
Hossam Eldeen Khaled AbdelGawad	1190182
Mohamed Hassan Jassim	1180614
Abdallah Gamal Abdelhady Ghita	4200380
Amr Mohmmed Shams El_Dieen	

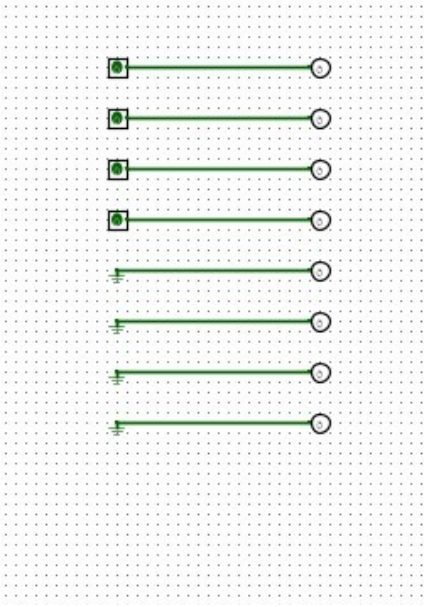
Work Distribution

Name	Operations

Hossam Eldeen Khaled AbdelGawad	Multiply (4-Bit Adder,HA,FA)
Mohamed Hassan Jassim	And, Or, Xor, Shift Right, Shift Left
Abdallah Gamal Abdelhady Ghita	Move, Not, Main circuit
Amr Mohmmed Shams El_Dieen	Add, Subtract , Negative

Modules

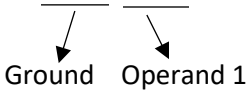
Move:



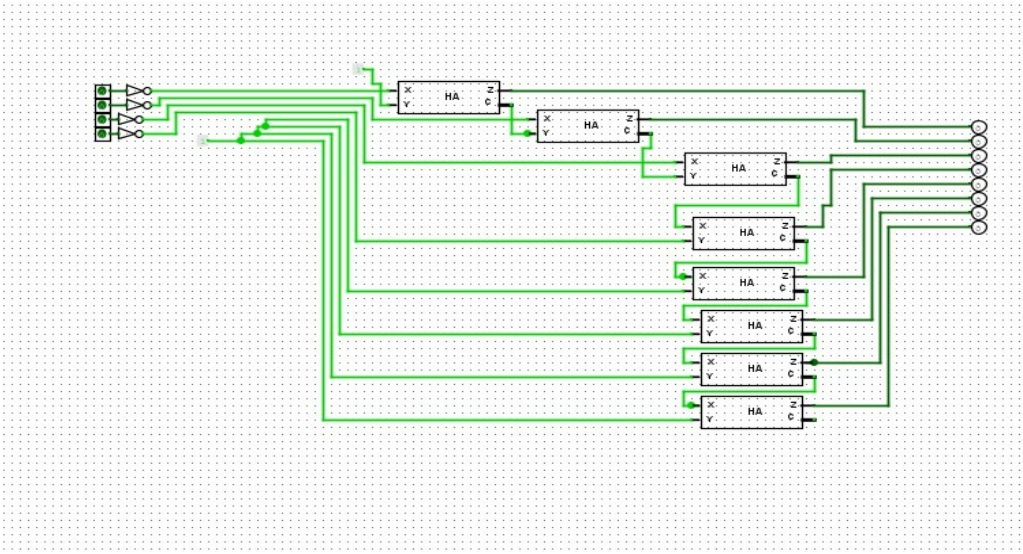
Number of gates:

Total: There are Zero gates in this Operation

How it was made: We simply put the input operand 1 in the latter half of the output preceded by the Four ground cables. As in 0000XXXX



Negative:



Number of gates:

Total: 20

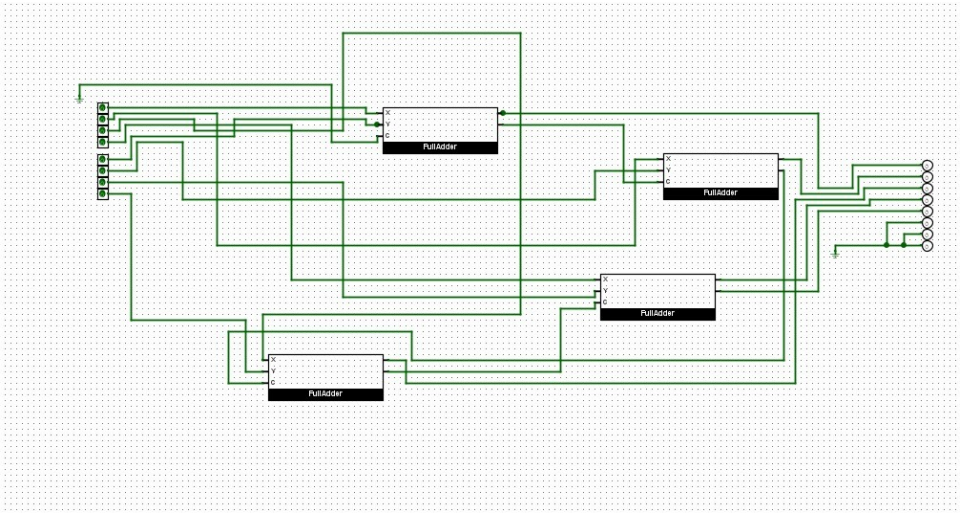
Detailed: 4 not + 8 xor + 8 and

How it was made:

I used the 2nd compliment principle. As I converted the number to the 1st compliment using Not gates then

I added one to the output number using half adder

Add:



Number of gates:

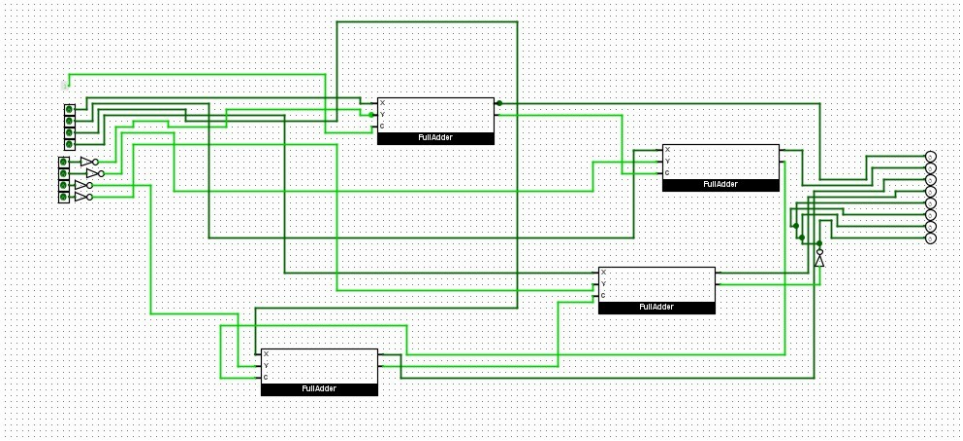
Total: 20

Detailed: 8 xor + 8 and + 4 or

How it was made:

I used the full adder to add each bit in the two operand in a full adder and the carry goes into the next full adder

Subtract:



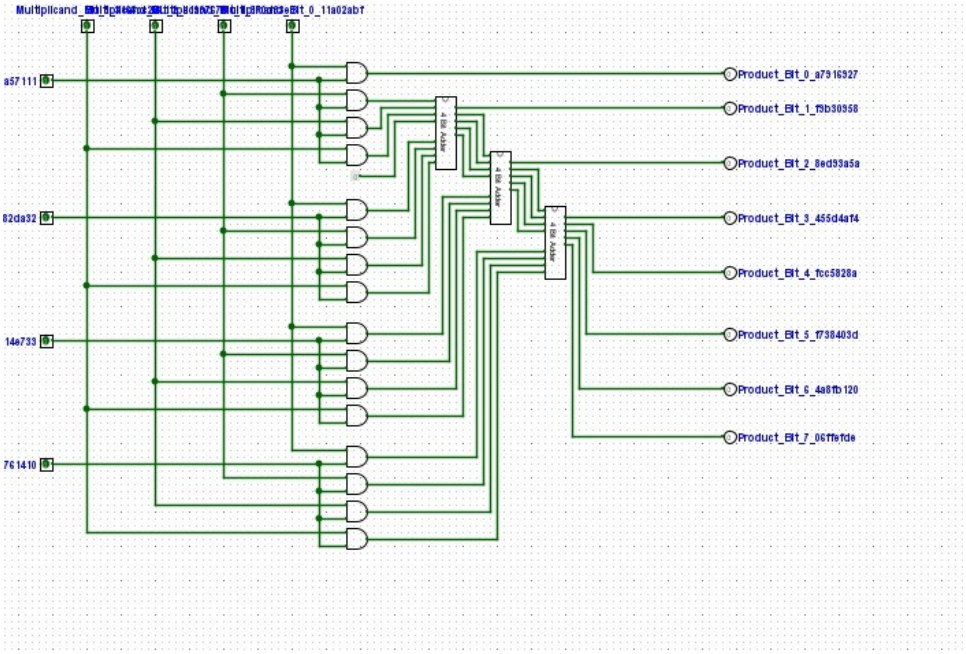
Number of gates:

Total: 25

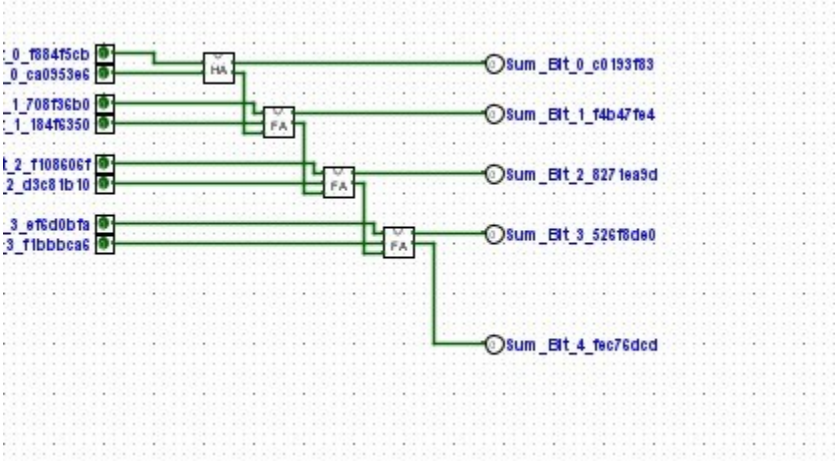
Detailed: 5 not + 8 xor + 8 and + 4 or

How it was made:

Multiply:



4 Bit Adder:



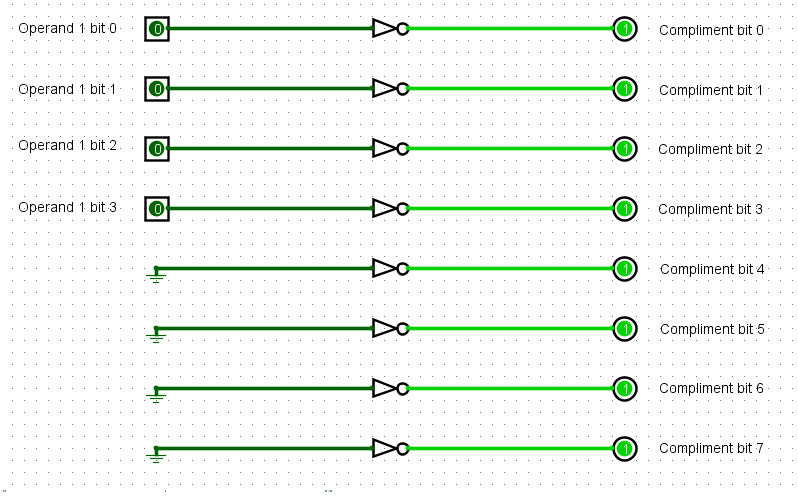
Number of gates:

Total=61

37 And + 21 Xor + 3 Or

How it was made: I multiplied each bit with the bit corresponding to it using and gates and a 4 bit adder that I made.

Not:



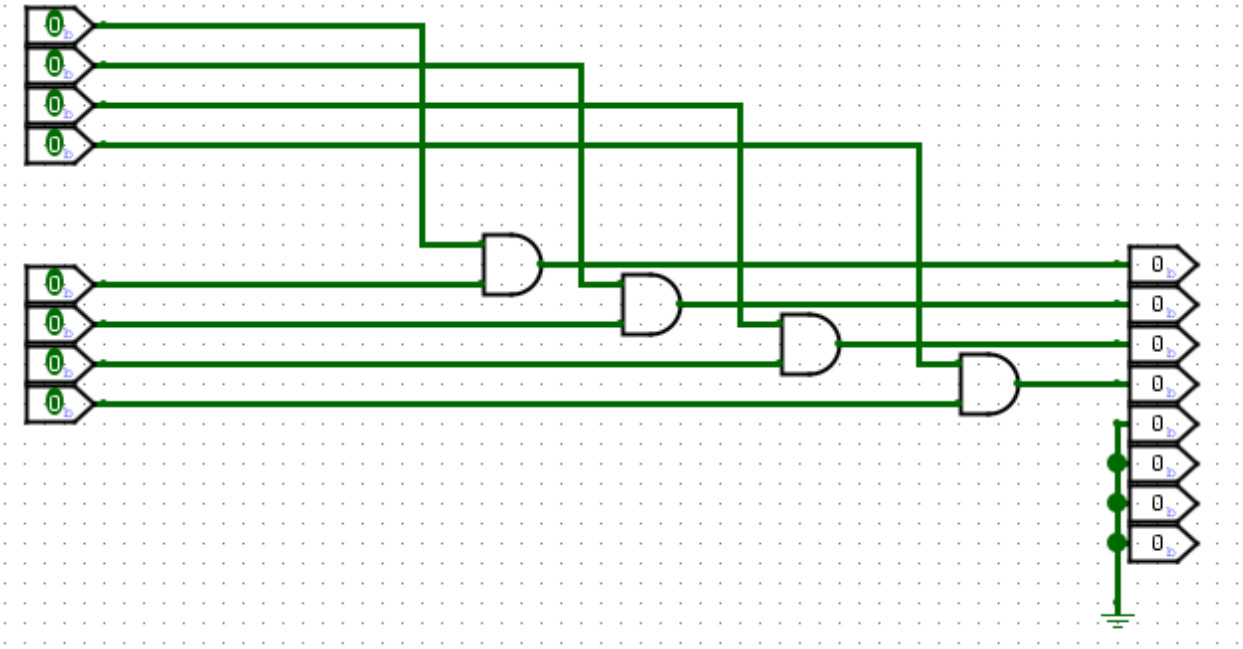
Number of gates:
There are only 8 gates.

Detailed: 8 Not gates

How it was made:
The concept is pretty much simple already, just like the Move operation we have four inputs for operand 1 with four outputs being operand1 ‘s complement, and four outputs are ground state’s complements. E.g. 1111XXXX

Ground Operand1 ‘s complement

And:



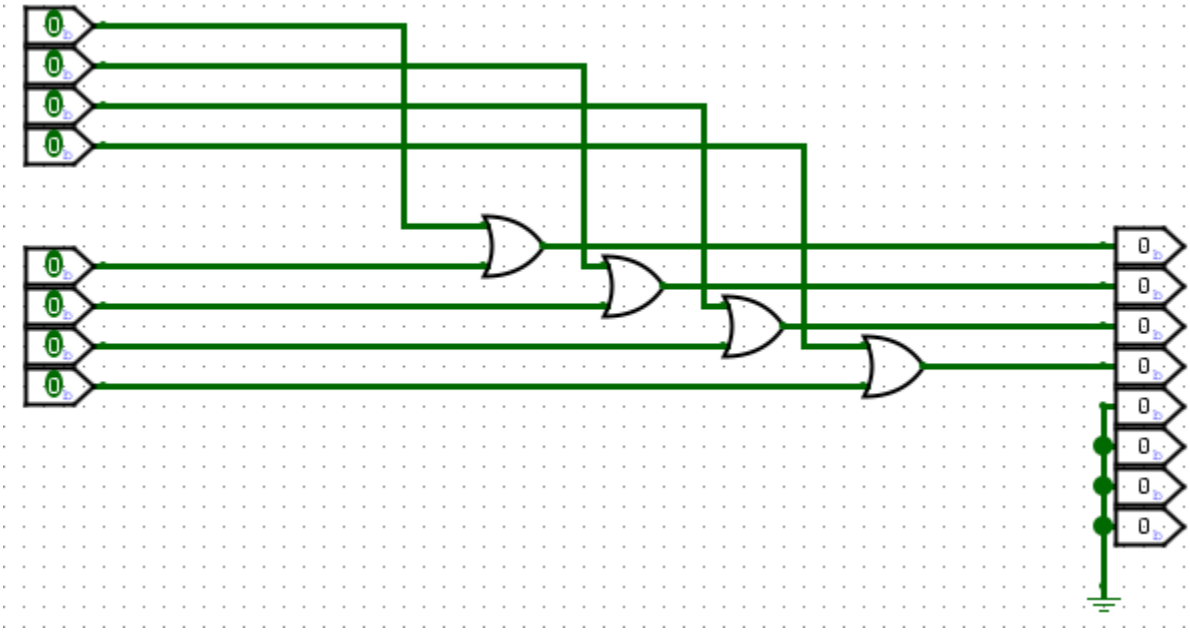
Number of gates:

Total: 4 Gates

Detailed: 4 And Gates

How it was made: every bit of each operand is compared to its opposite bit of the second operand by an And Gate

Or:



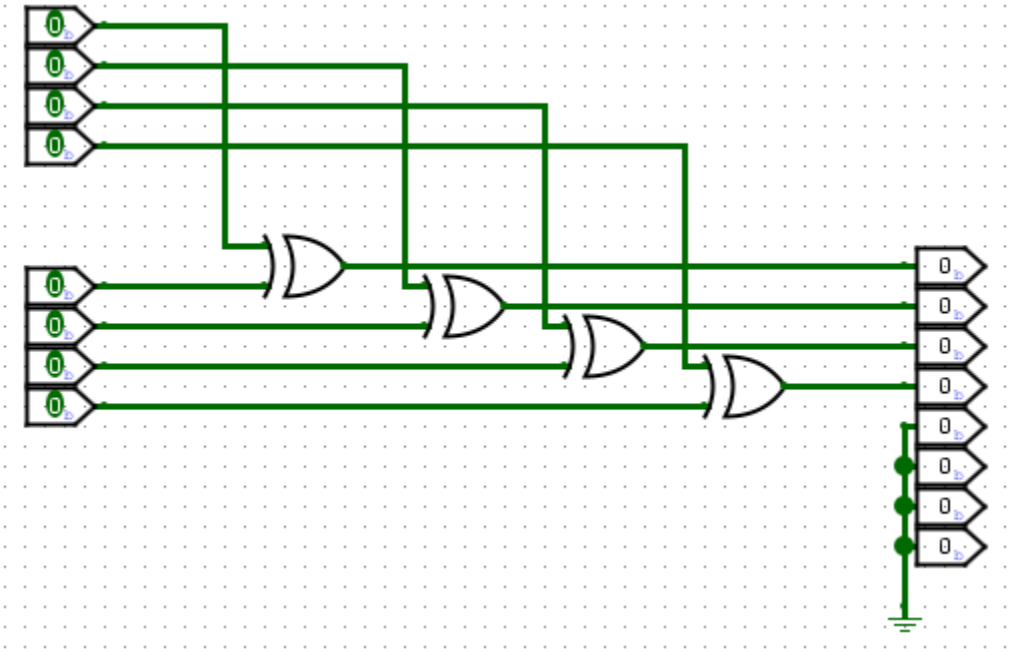
Number of gates:

Total: 4 Gates

Detailed: 4 Or Gates

How it was made: every bit of each operand is compared to its opposite bit of the second operand by an Or Gate

Xor:



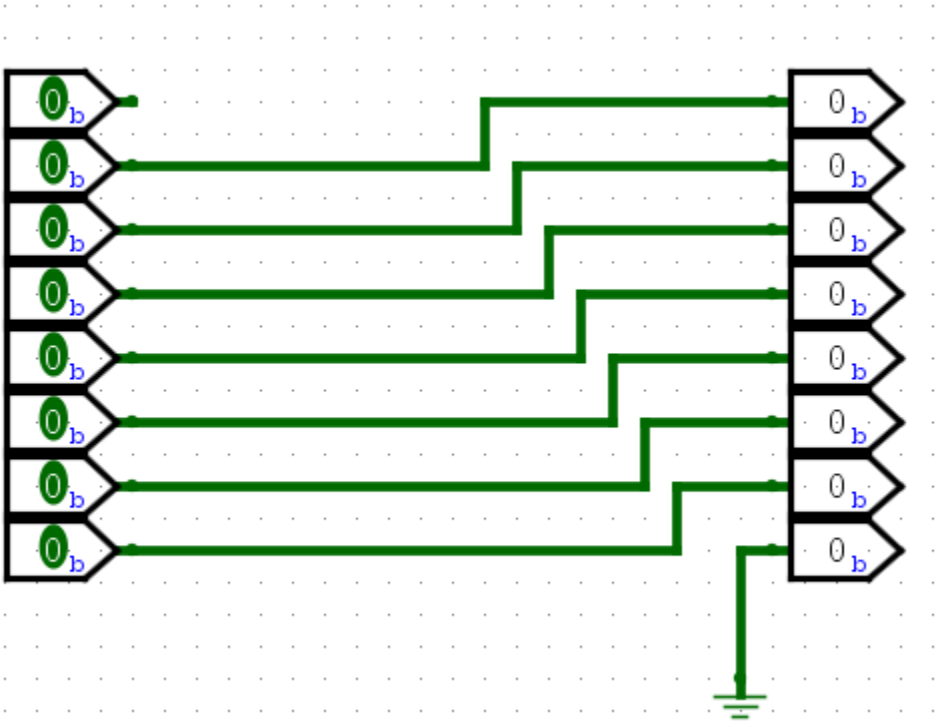
Number of gates:

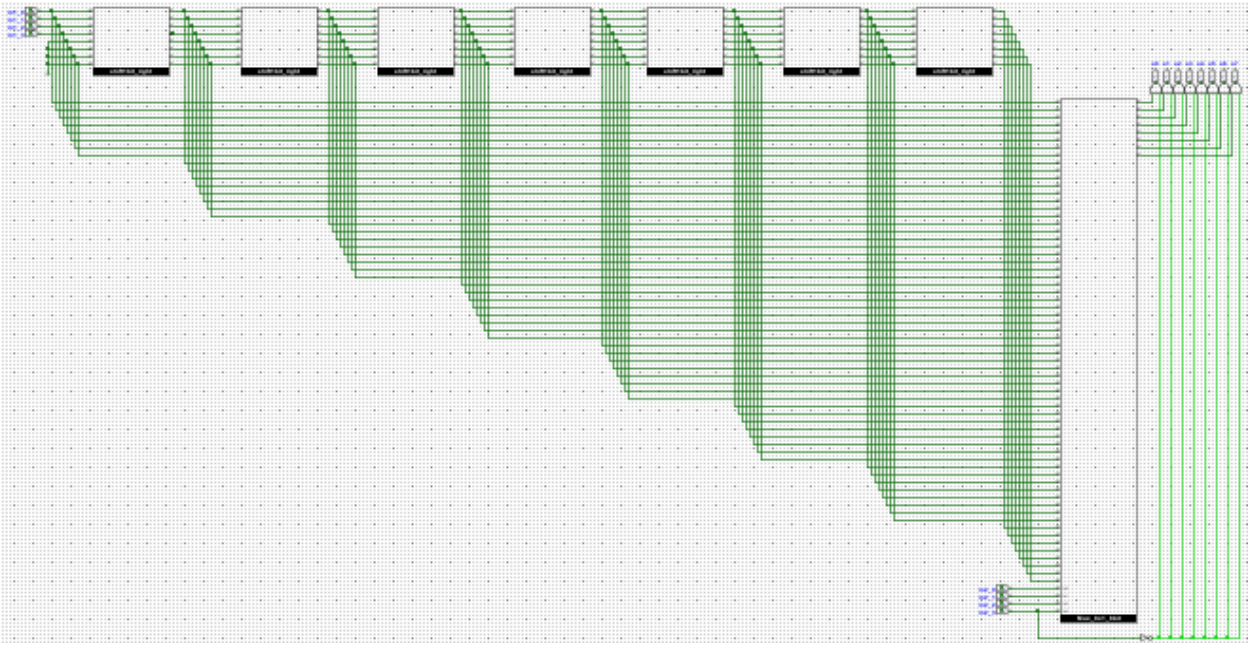
Total: 4 Gates

Detailed: 4 Xor Gates

How it was made: every bit of each operand is compared to its opposite bit of the second operand by Xor Gate

Shift Right:





Number of gates:

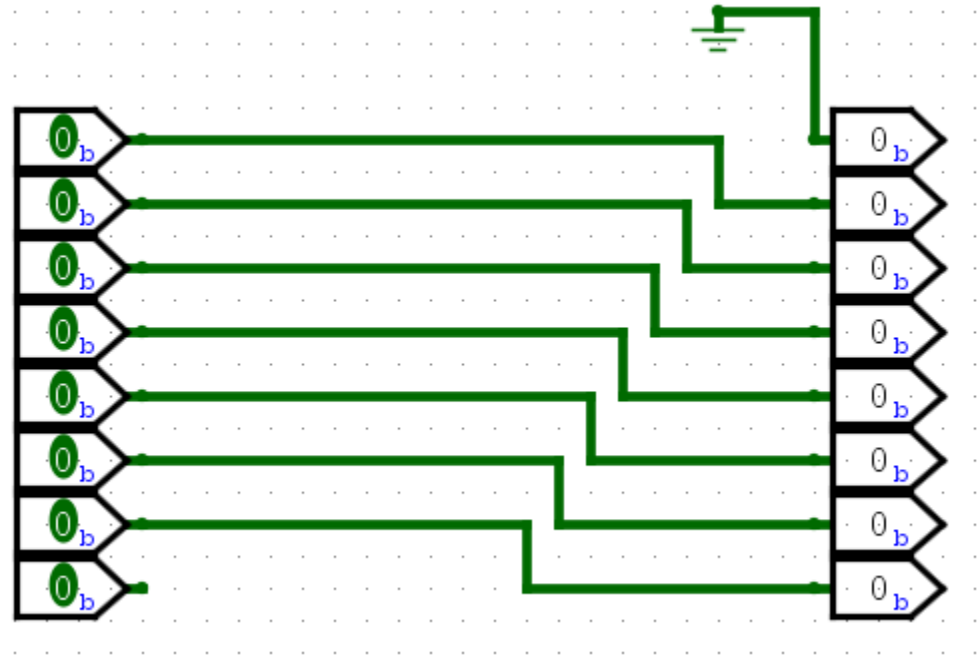
Total: 184 Gates

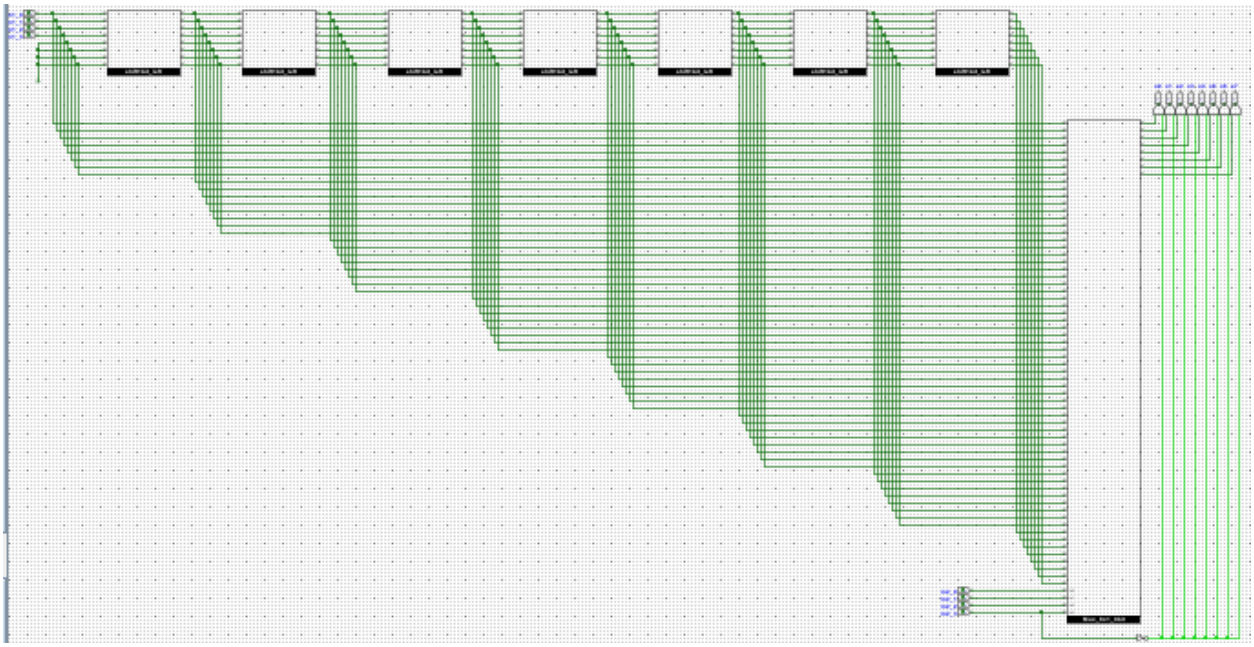
Detailed: 120 And Gates, 56 Or Gates, 8 Not Gates

How it was made: first I made a module that shifts the 8 bits of the input by only one bit to the right no gates were used in this module then I designed 2x1

multiplexer and then 8x1 multiplexer the first 4bit operand entered the module that shift 1bit and the output was entered to the shifting module 6 more times a row the output of each module was taken to be an input to the 8x1 multiplexer so it can be determined which shifting result should be taken according to the second operand readings

Shift Left:





Number of gates:

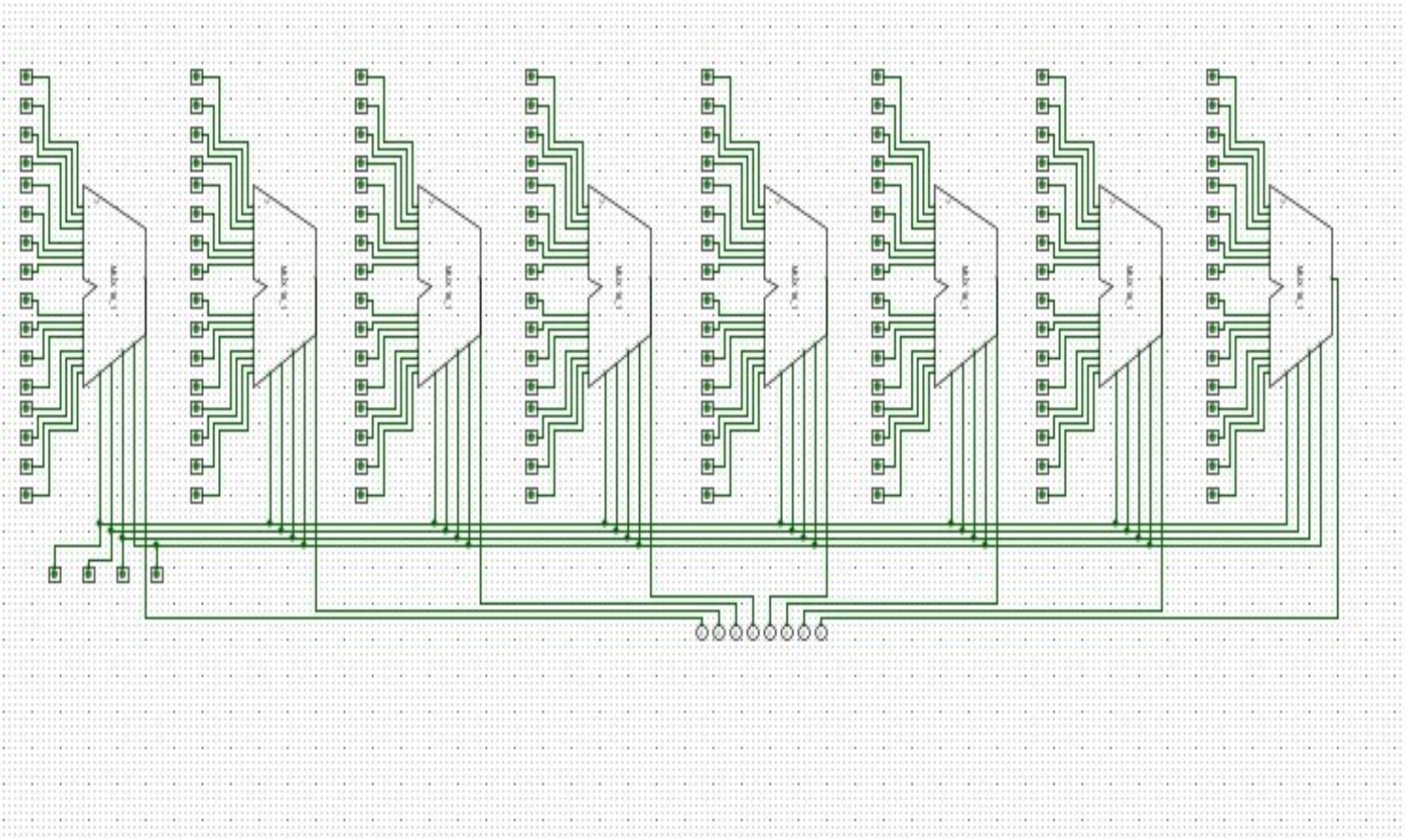
Total: 184 Gates

Detailed: 120 And gates, 56 Or Gates, 8 Not Gates

How it was made: the same thing in the right shifting operation first I made a module that shifts the 8 bits of the input by only one bit to the left no gates were used in this module then I designed 2x1 multiplexer and then 8x1 multiplexer the first 4bit operand entered the module that shift 1bit and the output was entered to the shifting module 6 more times a row the output of each module was taken to be an input to the 8x1 multiplexer so it can be determined which shifting result should be taken according to the second operand readings

Main:

L_8_bit_16_1_MUX

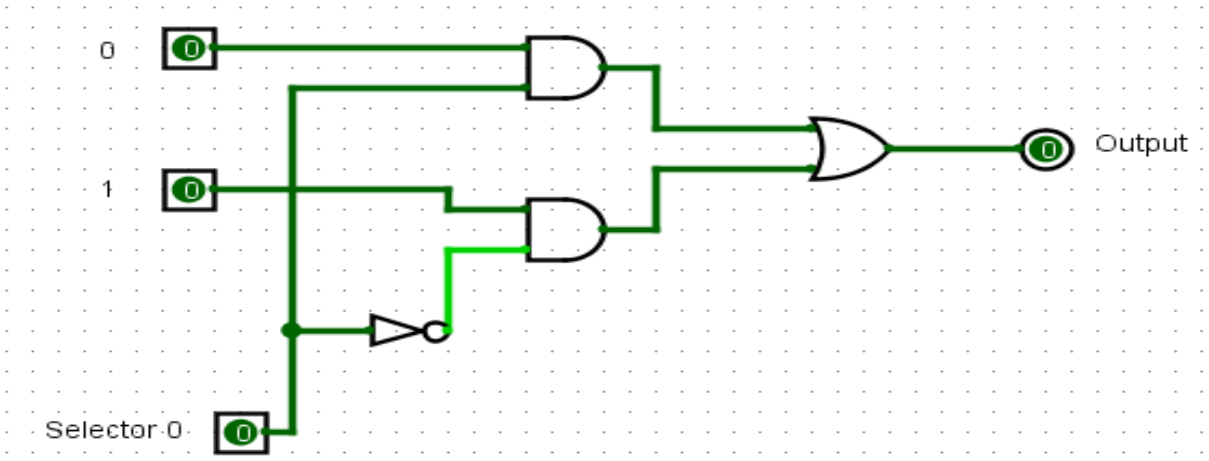


Number of gates:

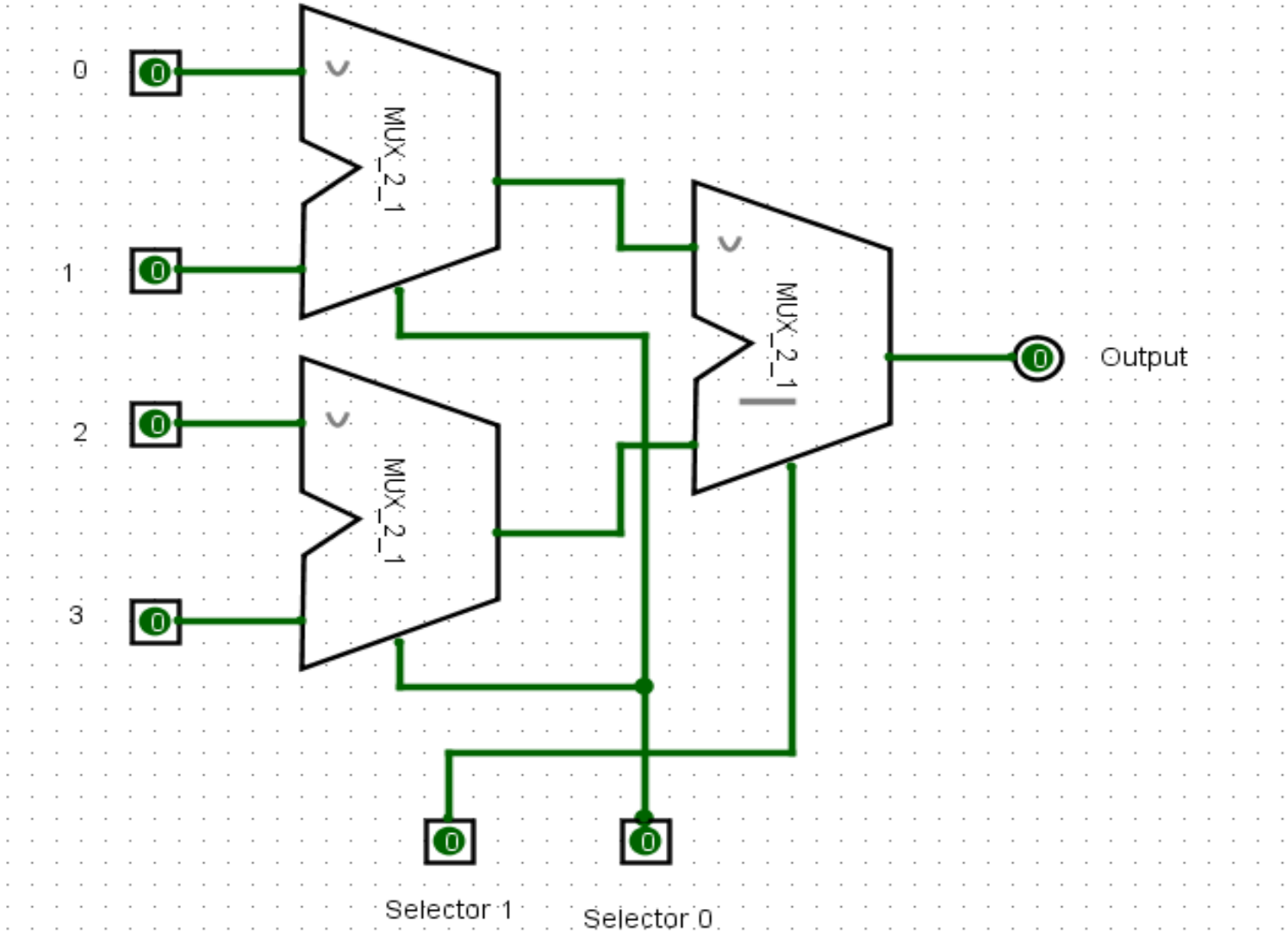
Total: There are a lot of gates inside this Giant, they sum up to a grand total of 480 gates, and that is not including the operation circuits.

Detailed:
The circuit consists of 240 And gates & 120 Or gates & 120 Not gates

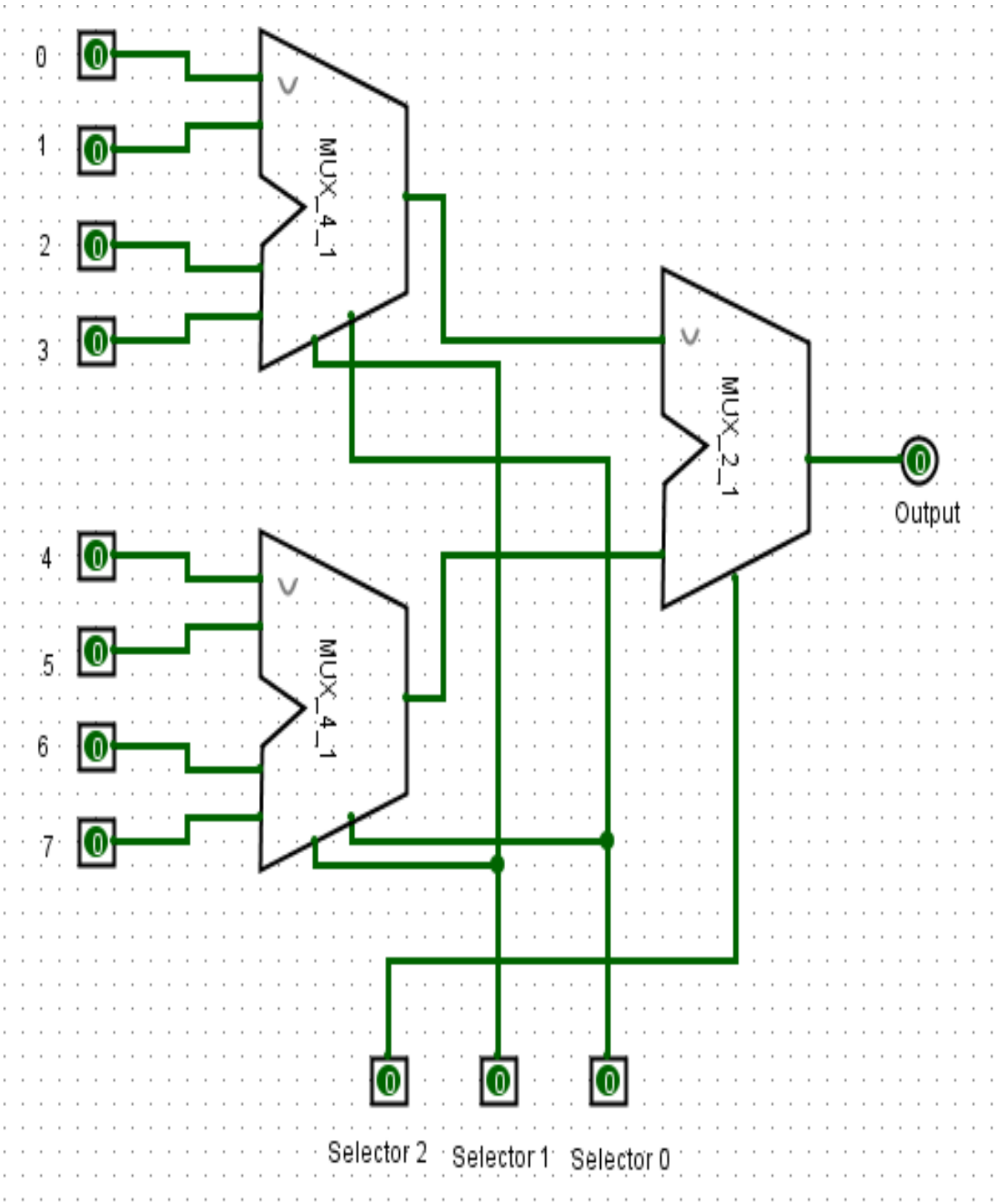
How it was made:
Because we are not allowed to use any gates other than the (Basic Gates: NOT, AND, OR, NAND, NOR, XOR, XNOR), We had to build the Multiplexers (MUX) from scratch starting with the 2_1 MUX



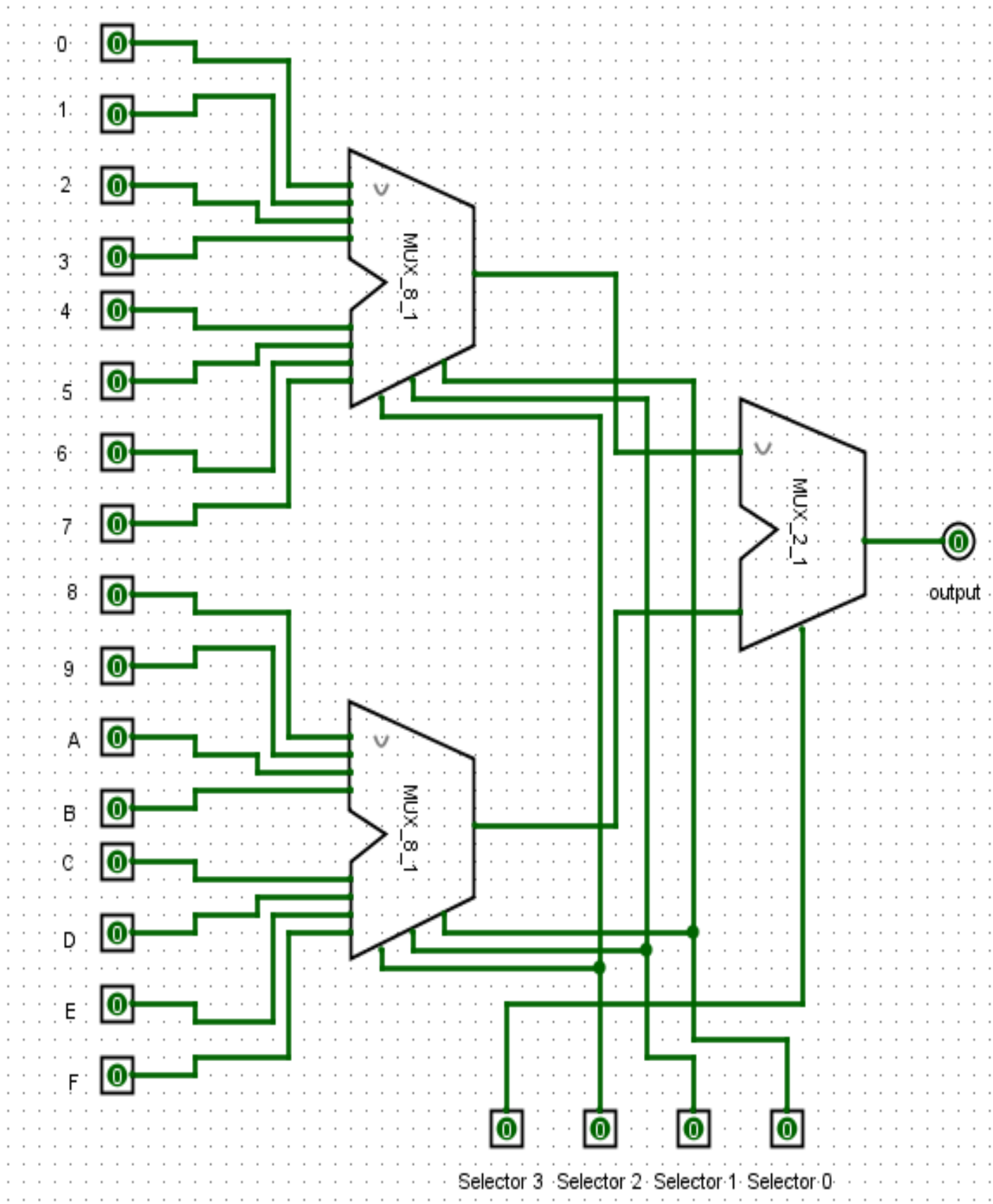
Then the 4_1 MUX



And the 8_1 MUX

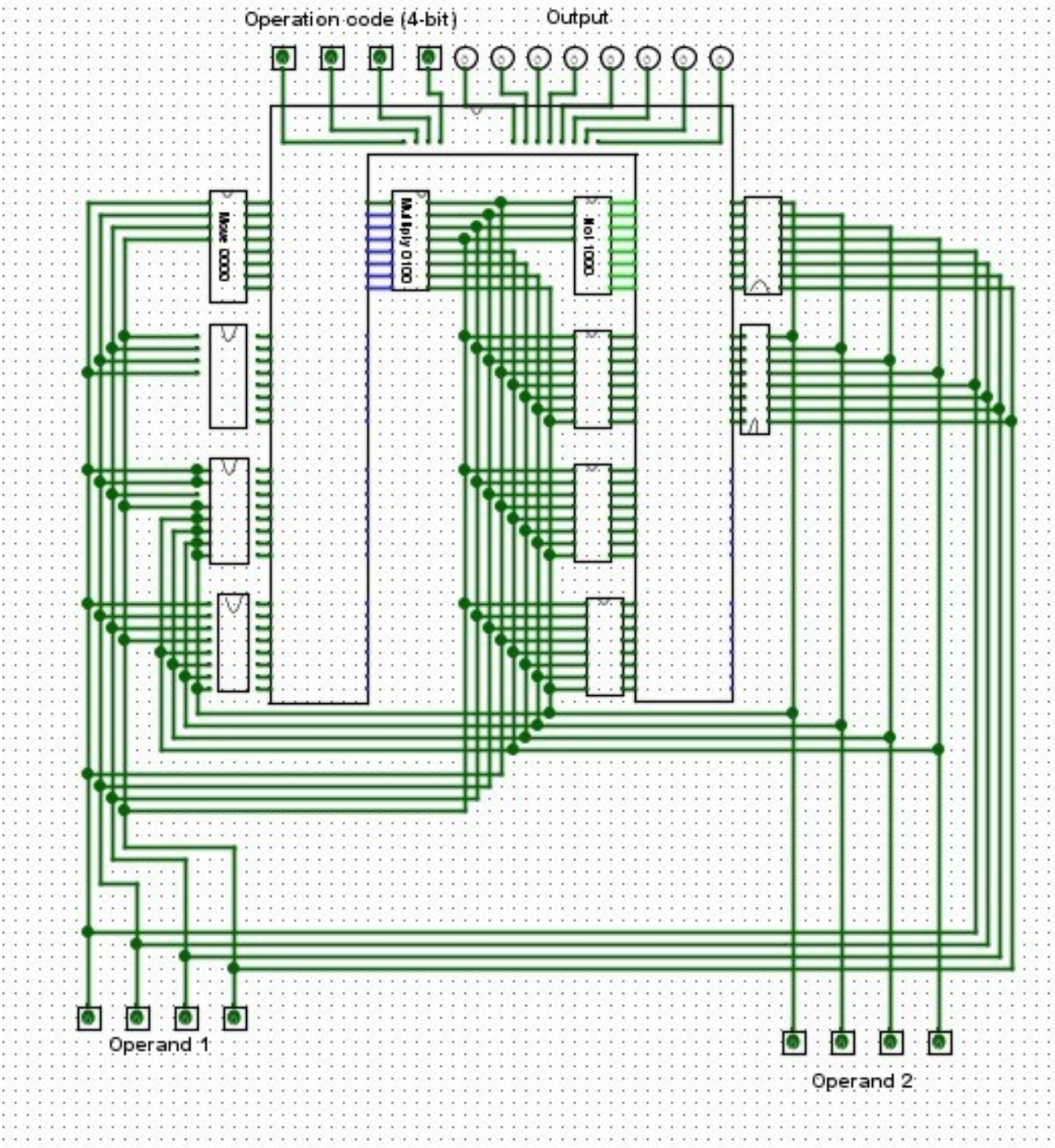


Then Finally we have our 16_1 MUX unit Used in the Main circuit of the ALU


0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
MUX_8_1
MUX_8_1
MUX_2_1
output
Selector 3
Selector 2
Selector 1
Selector 0

After that we connected 8 (16_1 MUX) to form the main circuit with a width of 8 bits.

By separating each element with the corresponding ones in the 8-bit 16_1MUX circuit we connected each operation in place, and we are now done with our ALU unit.


Operation code (4-bit)
Output
Operand 1
Operand 2