ECTE233

Project

Umm Kulsoom Emad Ul Mukhtar

5529657

Bachelors of Engineering in Electrical

Submitted to: Dr Joel Kennedy

INDEX

INTRODUCTION

OBJECTIVE:

To design a 4-bit Arithmetic Logic Unit (ALU) and test the custom made design using a simulation like Multisim or Simulink.

SPECIFICATION:

The ALU should have 10 inputs and 5 outputs. In order to choose from inputs, there should be select lines ($OP_{1:0}$). To create nibbles A and B, four input bits (for each nibble) $A_{3:0}$ and $B_{3:0}$ are used. For the output nibble S, the four bits $S_{3:0}$ are used. Also, one output bit is to indicate the overflow occurred during any operation.

OPCODE:

Opcode is the part of a machine language that signifies which operation should be performed.
The opcode used in this report:

$OP_{1:0}$	$OP_{1:0}$	$OP_{1:0}$	$OP_{1:0}$
00	01	10	11
Sn=Not(An)	S=rotate A right	S=A>B	S=-A

Conditions	Explanations			
	S_3	S_2	S_1	S_0
S=NOT(An)	$\sim A_3$	$\sim A_2$	$\sim A_1$	$\sim A_0$
S=Rotate A Right	A_0	A_3	A_2	A_1
S=A>B	A and B are 4 bit unsigned numbers. If A>B, S=XXX1, otherwise S=XXX0			
S=-A	A is a twos complement number. S is equal to A multiplied by negative 1. Leave -8 as -8 and use the overflow bit to indicate that the solution is out of range.			

ALU:

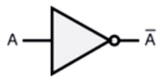
The digital electronic circuit that performs logical and arithmetic operations, bitwise, on integer numbers is called an Arithmetic Logic Unit (ALU).

LOGIC GATES:

Logic gates are building blocks in electronics which perform logical operations on one or more inputs giving only one output. Both the inputs and output are Boolean (either ON or OFF).

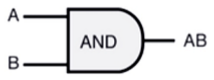
There are 7 main types of gates:-

1. NOT Gate
   There is only one input for this gate. The output is always the complementary to the input.



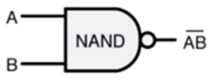
A	OUTPUT
0	1
1	0

2. AND Gate
   The output for this gate is ON only when all the inputs are ON.



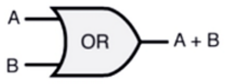
A	B	OUTPUT
0	0	0
0	1	0
1	0	0
1	1	1

3. NAND Gate
   The output for this gate is OFF only when all the inputs are ON.



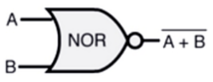
A	B	OUTPUT
0	0	1
0	1	1
1	0	1
1	1	0

4. OR Gate
   The output for this gate is ON when any input is ON.



A	B	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	1

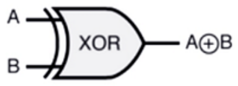
5. NOR Gate
   The output for this gate is ON only when all the inputs are OFF.



A	B	OUTPUT
0	0	1

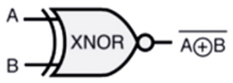
0	1	0
1	0	0
1	1	0

6. XOR Gate
   The output for this gate is ON when any odd number of inputs are same.



A	B	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	0

7. XNOR Gate
   The output for this gate is ON when any even number of inputs are same.



A	B	OUTPUT
0	0	1
0	1	0
1	0	0
1	1	1

<u>MULTIPLEXER:</u>

A multiplexer selects data from an input line and directs it to an output line. It has selection inputs ‘n’, information inputs ‘2^n’ and only one output.

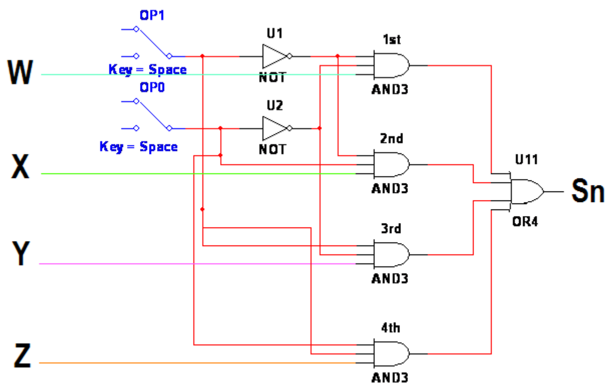
DESIGN AND EXPLANATION

CONNECTION OF INPUT AND OUTPUT BITS:

Each input bit is a switch whose one end is connected to a VCC (ON) and the other end is connected to the ground (OFF). Every output bit is connected to a probe to indicate whether it is ON or OFF by lighting up.

CONNECTION OF MULTIPLEXERS:

We used four 4-to-1 Multiplexers. Example is shown below.



Depending upon the combination of the select lines ($OP_{1:0}$), one of the four input lines is directed to the output.

OP1	OP0	Sn	What does it represent
0	0	W	1st condition
0	1	X	2nd condition
1	0	Y	3rd condition
1	1	Z	4th condition

1. Opcode's 1st Condition: Sn=Not(An)
   We connected NOT gate to each bit of An ($A_{3:0}$).
   The output from the NOT gate of the A_3 was connected to the 'W' of mux having the output S_3.
   The output from the NOT gate of the A_2 was connected to the 'W' of mux having the output $S_{2.}$
   The output from the NOT gate of the A_1 was connected to the 'W' of mux having the output $S_{1.}$
   The output from the NOT gate of the A_0 was connected to the 'W' of mux having the output $S_{0.}$

2. Opcode's 2nd Condition: Rotate A right
   A wire from A_3 was connected to the 'X' of mux having the output $S_{2.}$
   A wire from A_2 was connected to the X' of mux having the output $S_{1.}$
   A wire from A_1 was connected to the X' of mux having the output $S_{0.}$
   A wire from A_0 was connected to the X' of mux having the output $S_{3.}$

3. Opcode's 3rd Condition: S= A>B
   A and B are 4-bit unsigned numbers, when input A is greater than input B the output S_0 would be ON (S=XXX1).
   We started by comparing the most significant bit of both numbers and forming a truth table:

A_3	B_3	$A_3>B_3$ (OUTPUT)
0	0	0
0	1	0
1	0	1
1	1	0

Output equation for the truth table is $A_3 \cdot \overline{B_3}$.

For $A_3 > B_3$, the equation is, $E_1 = \boldsymbol{A_3 \cdot \overline{B_3}}$.

Then we compared the second most significant bit by presuming that $A_3 == B_3$

A_3	B_3	$A_3==B_3$ (OUTPUT)
0	0	1
0	1	0
1	0	0
1	1	1

Output equation is $\overline{A_3 \oplus B_3}$

Hence, when any same bits of A is greater than B the equation is $A_n \cdot \overline{B_n}$ and when they are equal to each other, the equation is $\overline{A_n \oplus B_n}$

For $A_3 == B_3$ and $A_2 > B_2$, the equation is, $E_2 = \boldsymbol{\overline{A_3 \oplus B_3} \cdot (A_1 \cdot \overline{B_1})}$

Then comparing the 3rd bit of both numbers, we presumed that first two bits were equal and the third bit of A is greater than that of B.

For $A_3 == B_3$, $A_2 == B_2$ and $A_1 > B_1$, the equation is, $E_3 = \boldsymbol{\overline{A_3 \oplus B_3} \cdot \overline{A_2 \oplus B_2} \cdot (A_1 \cdot \overline{B_1})}$

Lastly, comparing the least significant bit of A and B, we presumed the previous three bits were equal and the last bit of A is greater than that of B.

For $A_3 == B_3$, $A_2 == B_2$, $A_1 == B_1$ and $A_0 > B_0$, the equation is, $E_4 = \boldsymbol{\overline{A_3 \oplus B_3} \cdot \overline{A_2 \oplus B_2} \cdot \overline{A_1 \oplus B_1} \cdot (A_0 \cdot \overline{B_0})}$

If the output from any of the above four mentioned equations was ON that is when number would be greater than number B. For that reason, four separate logic circuits were created from above obtained equations ($E_{1:4}$). The output signal from each of them was combined with 4-input OR gate. The output of the OR gate was connected to the 'Y' of the multiplexer having the output S_0. The 'Y' of the other three mux with the bits S_3, S_2 and S_1 were grounded because they don't affect as per the condition's explanation.

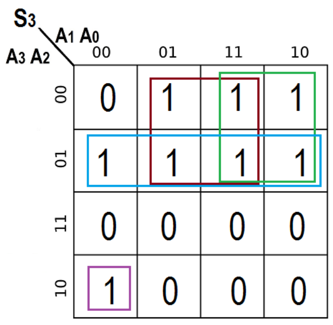
4. <u>Opcode's 4th Condition: S= -A</u>
This meant that whatever number is input as A, the output S of it would be negative A. The range of two's complement numbers that can be represented in 4-bits is from -8 to 7. Thus, any other output nibble would cause an overflow. We input 16 combinations starting from 0 to 15 and formed a truth table. Also, according to condition's explanation, the output for -8 remains -8.

Inputs				**Outputs**				
A_3	*A_2*	*A_1*	*A_0*	*S_3*	*S_2*	*S_1*	*S_0*	*Overflow*
0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	0
0	0	1	0	1	1	1	0	0
0	0	1	1	1	1	0	1	0
0	1	0	0	1	1	0	0	0
0	1	0	1	1	0	1	1	0
0	1	1	0	1	0	1	0	0
0	1	1	1	1	0	0	1	0
1	0	0	0	1	0	0	0	0
1	0	0	1	0	1	1	1	1
1	0	1	0	0	1	1	0	1
1	0	1	1	0	1	0	1	1
1	1	0	0	0	1	0	0	1
1	1	0	1	0	0	1	1	1
1	1	1	0	0	0	1	0	1
1	1	1	1	0	0	0	1	1

Then we created k-maps for each output and obtained equations. For each equation, a circuit was made and it was connected to the corresponding outputs.

- ❖ K-maps and their Equations for each Output

→S_3

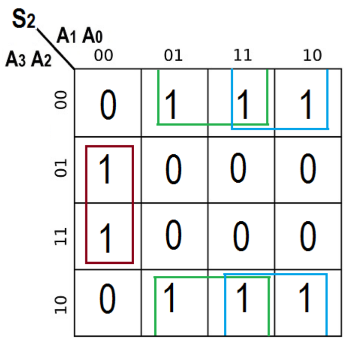


$$S_3 = A_3\overline{A_2}\,\overline{A_1}\,\overline{A_0} + \overline{A_3}A_2 + \overline{A_3}A_1 + \overline{A_3}\,A_0$$

$$S_3 = A_3 \oplus (A_2A_1A_0)$$

A logic circuit for the S_3 equation was made, and connected to the 'Z' of its multiplexer.

→S_2

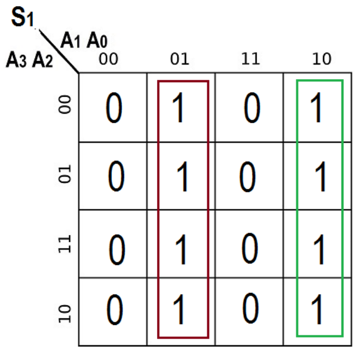


$$S_2 = A_2\overline{A_1}\,\overline{A_0} + \overline{A_2}\,A_1 + \overline{A_2}\,A_0$$

$$S_2 = A_2 \oplus (A_1A_0)$$

A logic circuit for the S_2 equation was made, and connected to the 'Z' of its multiplexer.

→S_1



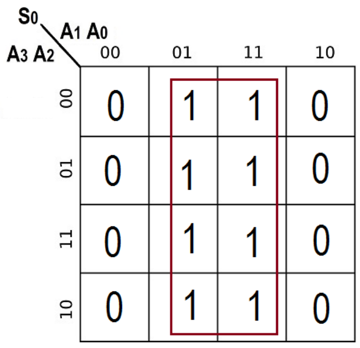
S1 A3 A2 \ A1 A0	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$$S_1 = \overline{A_1}A_0 + A_1\overline{A_0}$$

$$S_1 = A_1 \oplus A_0$$

A logic circuit for the S_1 equation was made and connected to the 'Z' of its multiplexer.

→S_0

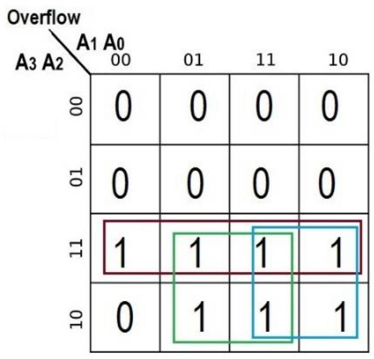


S0 A3 A2 \ A1 A0	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	1	1	0
10	0	1	1	0

$$S_0 = A_0$$

For equation S_0, a wire from A_0 is connected to the 'Z' of its multiplexer.

→Overflow



Overflow A3 A2 \ A1 A0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	1	1	1

$$Overflow = A_3A_2 + A_3A_1 + A_3A_1$$

$$A_3(A_2 + A_1 + A_0)$$

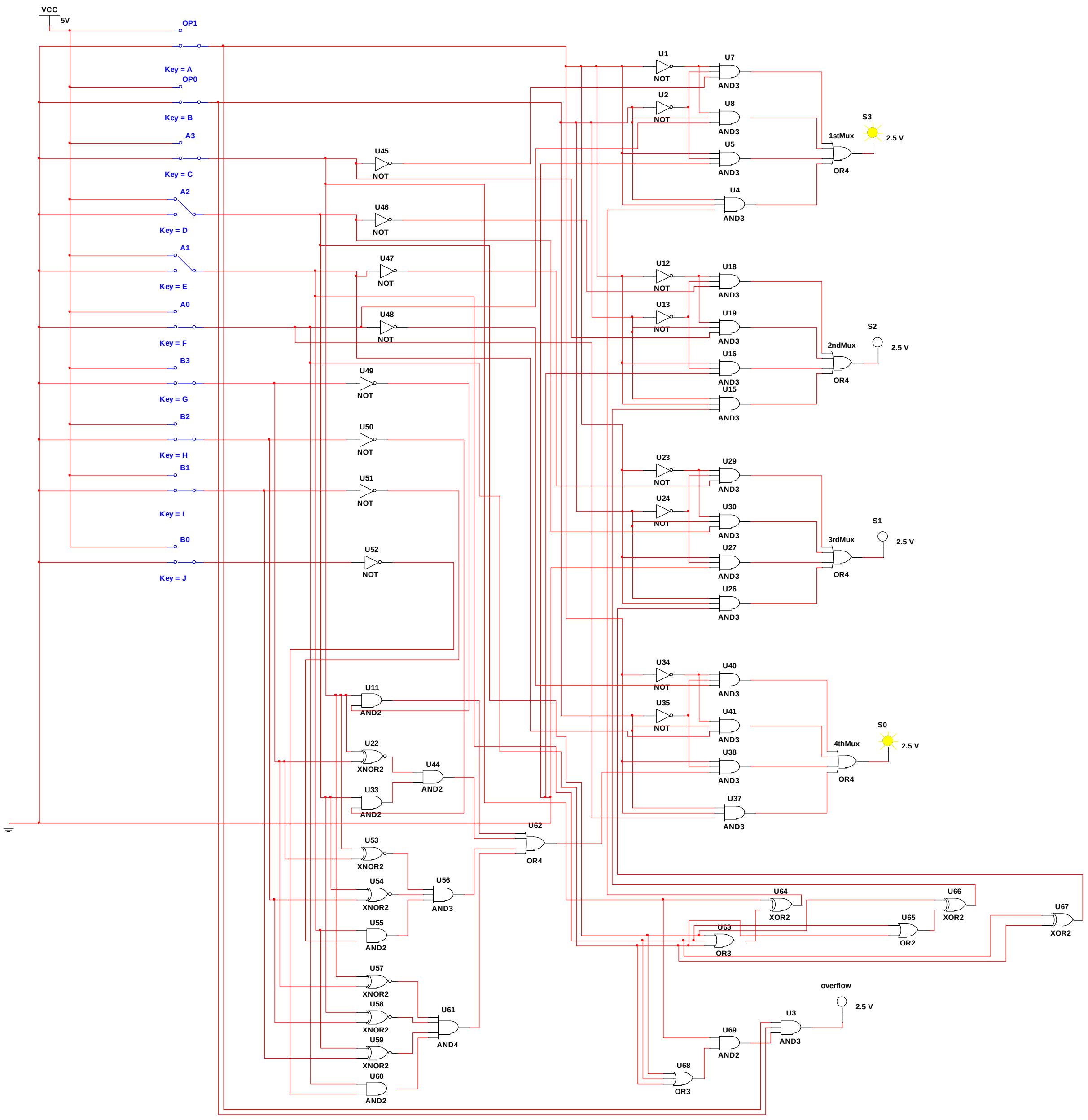
The output from the above overflow equation is connected to a 3-input AND gate along with both selectors ($OP_{1:0}$) and then has a separate output probe.

TESTING PROCEDURE

We tested our arithmetic logic unit by providing an input, predicting the expected output and comparing that with our output in Multisim (a figure is attached for each input).
→Whenever an output bit is ON (1) that's when the probe of that output bit is lit.

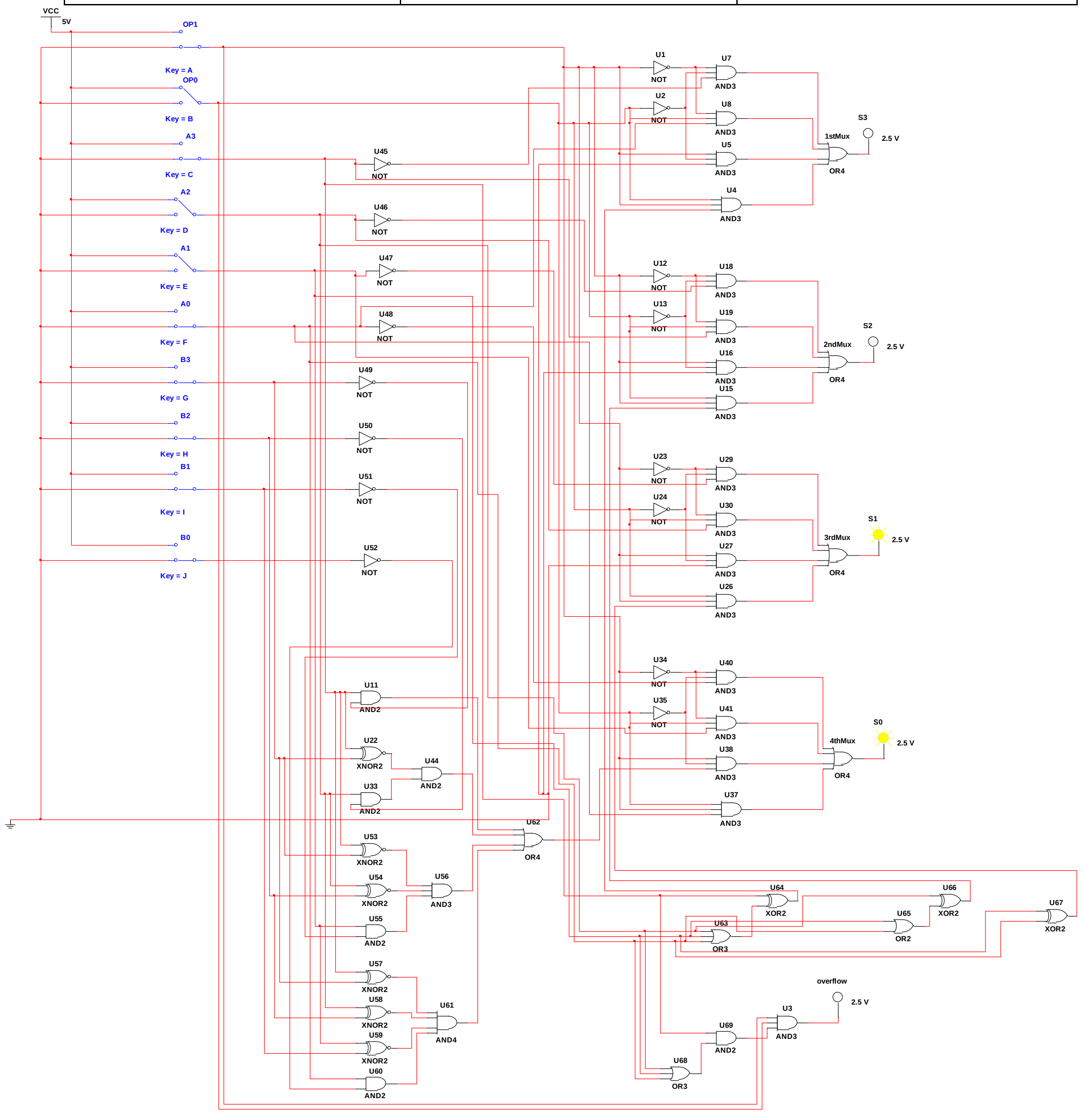
1. Opcode’s 1st Condition: Sn=Not(An)

$OP_{1:0}$	Input	Expected Output
00	0110	1001



2. Opcode’s 2st Condition: Rotate A right

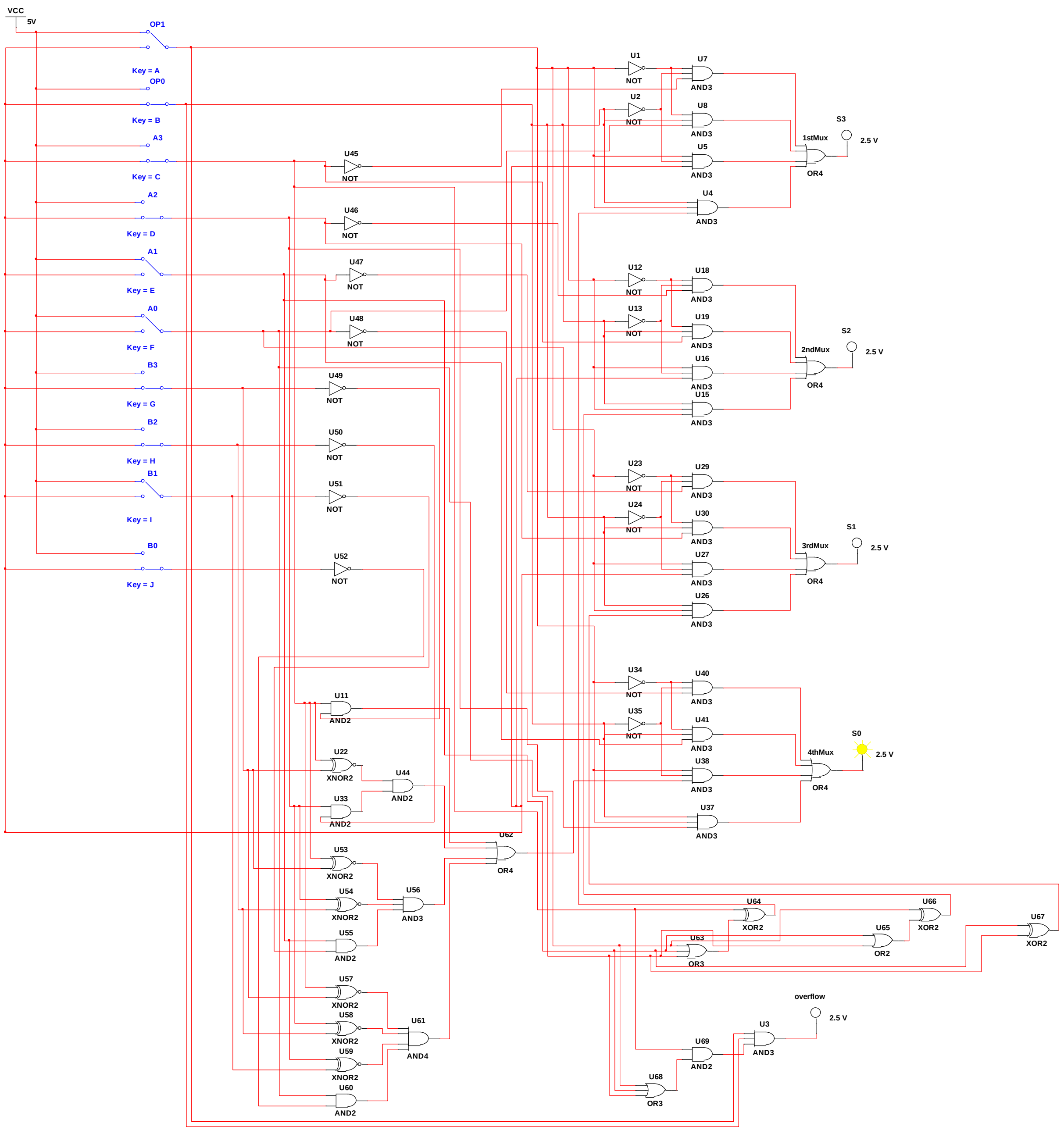
$OP_{1:0}$	Input	Expected Output
01	0110	0011



3. Opcode's 3rd Condition: A>B

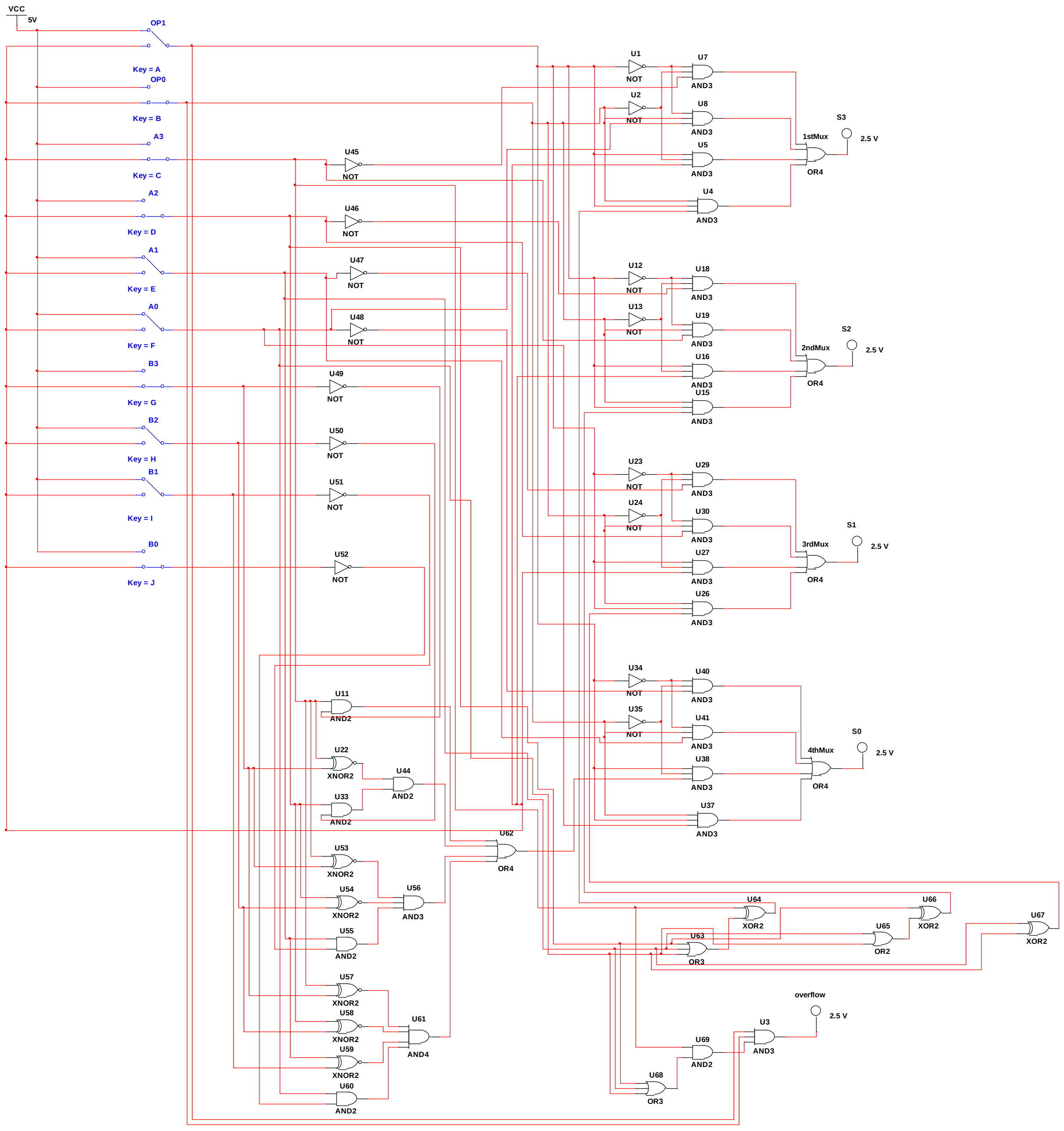
$OP_{1:0}$	1st Input	Expected Output

10	A=0011
	B=0010
		XXX1



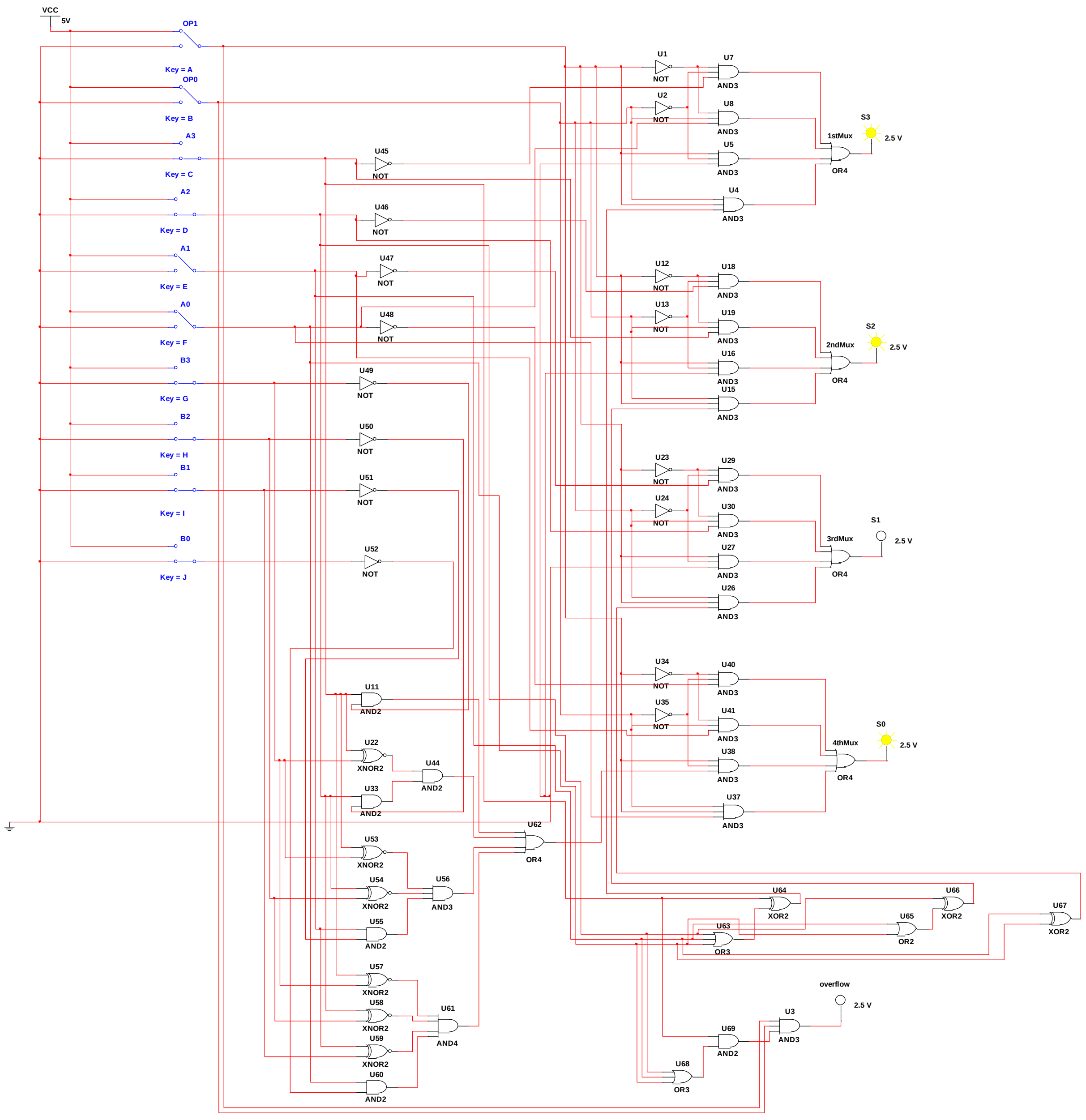
$OP_{1:0}$	2nd Input	Expected Output

10	A=0011
	B=0110
		XXX0

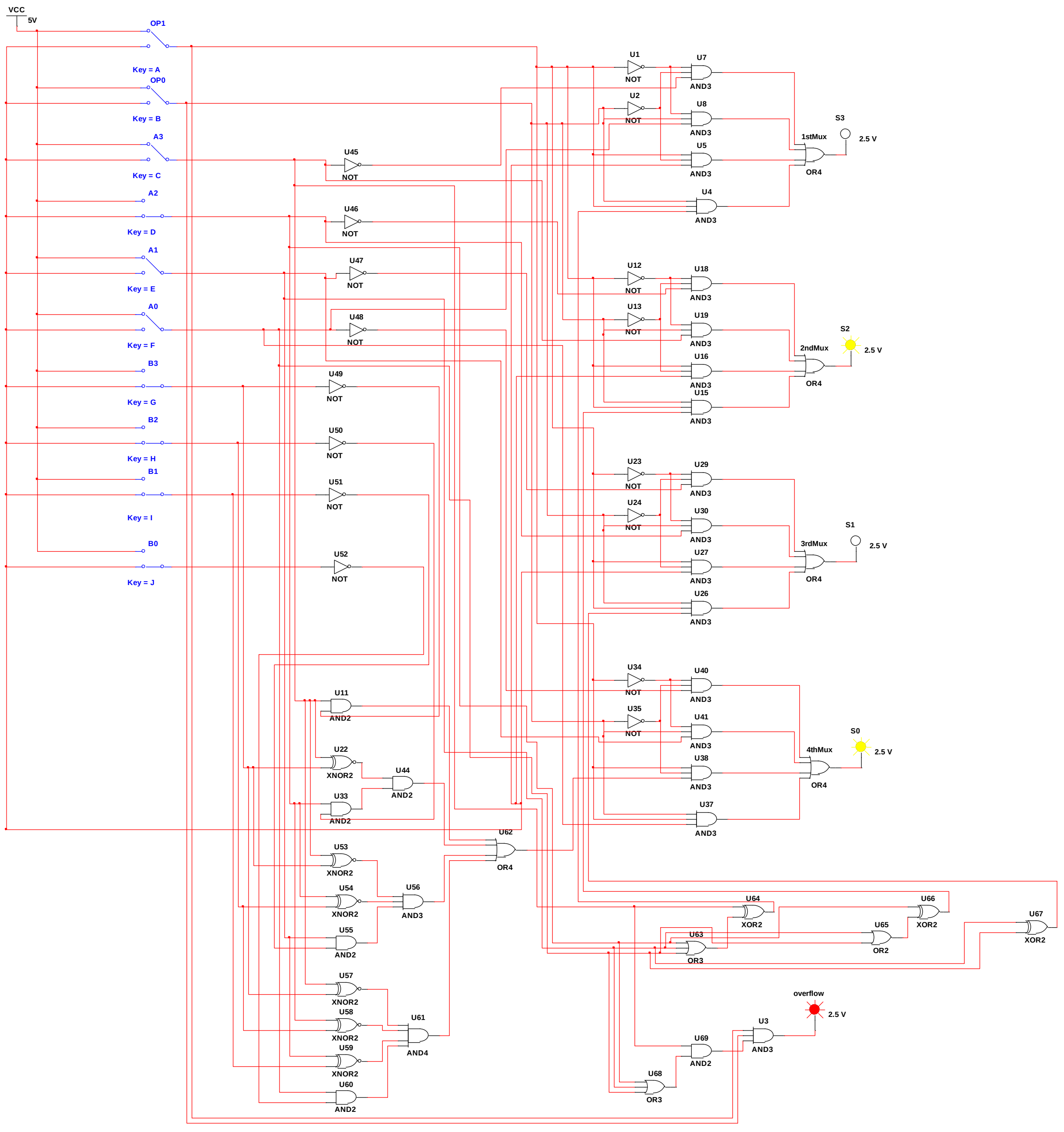


4. Opcode's 4th Condition: S=-A

$OP_{1:0}$	Input	Expected Output	Overflow
11	0011	1101	0



$OP_{1:0}$	Input	Expected Output	Overflow
11	1011	0101	1



CONCLUSION

In conclusion, the arithmetic logic unit given to us for designing, has the input and output nibbles, connections and testing outputs as required.