CS 225/226 - Mini Project

Part 1

Tameesh Biswas(1601CS08) and Parth Kulkarni (1601CS20)

Aim: Designing and simulating an 8 operation ALU in LogiSim software.

Components Used:

1) For 4-bit AND: 2 4-bit inputs, 1 4-bit output, 3 splitters, 4 AND gates
2) For 4-bit OR: 2 4-bit inputs, 1 4-bit output, 3 splitters, 4 OR gates
3) For 4-bit XOR: 2 4-bit inputs, 1 4-bit output, 3 splitters, 4 XOR gates
4) For 1-bit FULL ADDER: 3 1-bit inputs, 2 1-bit outputs, 2 XOR gates, 2 AND gates, 1 OR gate
5) For 4-bit ADDER/SUBTRACTOR: 2 4-bit inputs, 1 1-bit input, 1 4-bit output, 1 1-bit output, 4 1-bit FULL ADDERs, 3 splitters, 4 XOR gates,
6) For 4-bit BIN to GRAYCODE CONVERTER: 1 4-bit input, 1 4-bit output, 2 splitters, 3 XOR gates
7) For Main ALU: 2 4-bit inputs, 1 3-bit input, 1 1-bit input, 1 4-bit output, 1 GND input, 1 HIGH input, 2 NOT gates, 2 4-bit AND gates, 2 4-bit OR gates, 2 4-bit XOR gates, 2 4-bit ADDER/SUBTRACTORs, 1 4-bit BIN to GRAYCODE CONVERTER, 1 8:1 MULTIPLEXER, 1 2:1 MULTIPLEXER.

Objectives of the Project:

The main objective of the project was to design an ALU and simulate it in LogiSim that takes in two 4 bits of input and performs the multiple operations on them, outputting the result in both binary and gray code, based on the output control bit.

The eight operations are:

1. 4 bit AND output of 2x4 bit inputs
2. 4 bit OR output of 2x4 bit inputs
3. 4 bit XOR output of 2x4 bit inputs
4. 4 bit NAND output of 2x4 bit inputs
5. 4 bit NOR output of 2x4 bit inputs
6. 4 bit XNOR output of 2x4 bit inputs
7. 4 bit ADDER output of 2x4 bit inputs
8. 4 bit SUBTRACTOR output of 2x4 bit inputs

Critical Analysis:
The designed ALU is a 4-bit ALU, taking two 4-bit inputs and is able to perform 8 operations on them namely, Bitwise AND, Bitwise OR, Bitwise XOR, Bitwise NAND, Bitwise NOR, Bitwise XNOR, ADD and SUBTRACT; leading to a 4-bit output. This output can be obtained in either Binary or Graycode, depending on the user.

A 8:1 MUX is used to decide the operation to be carried out. The detailed description of each of the operation is described as follows:
0/000 (AND): two 4-bit numbers are bitwise ANDed by in this operation. A seperate module is created for this purpose is used in the main ALU.
1/001 (OR): two 4-bit numbers are bitwise ORed by in this operation. A seperate module is created for this purpose is used in the main ALU.
2/010 (XOR): two 4-bit numbers are bitwise XORed by in this operation. A seperate module is created for this purpose is used in the main ALU.
3/011 (NAND): two 4-bit numbers are bitwise ANDed by in this operation and then output is inverted using a NOT gate. A seperate module is created for this purpose is used in the main ALU.
4/100 (NOR): two 4-bit numbers are bitwise ORed by in this operation and then output is inverted using a NOT gate. A seperate module is created for this purpose is used in the main ALU.
5/101 (XNOR): two 4-bit numbers are bitwise XORed by in this operation and then output is inverted using a NOT gate. A seperate module is created for this purpose is used in the main ALU.
6/110 (ADD): two 4-bit numbers are ADDed by in this operation. A seperate module is created for this purpose, which is a 4-bit ripple carry ADDER/SUBTRACTOR, made using 4 1-bit FULL ADDERs(a seperate module created for this) set to ADD mode using GND input is used in the main ALU.
7/111 (SUBTRACT): two 4-bit numbers are SUBTRACTed by in this operation. A separate module is created for this purpose, which is a 4-bit ripple carry ADDER/SUBTRACTOR, made using 4 1-bit FULL ADDERs(a seperate module created for this) set to SUBTRACT mode using HIGH input is used in the main ALU.

A 2:1 MUX is used to choose the output format of the 4-bit ALU. The following description is suffice for the same:
0 (BIN ouptut): 4-bit binary output can be obtained if this option is chosen
1 (GRAYCODE output): 4-bit Graycode output can be obtained if this option is chosen. Graycode is a format in which consecutive numbers have a difference of only 1 bit. A seperate module is used to obtain Graycode output from the binary output of the ALU.

Experimental Results and Demo:

This ALU that was designed on LogiSim was simulated using different input test cases and all the operations worked as expected.

Following are some of the test cases for each operation that our ALU performs:

1. AND operation

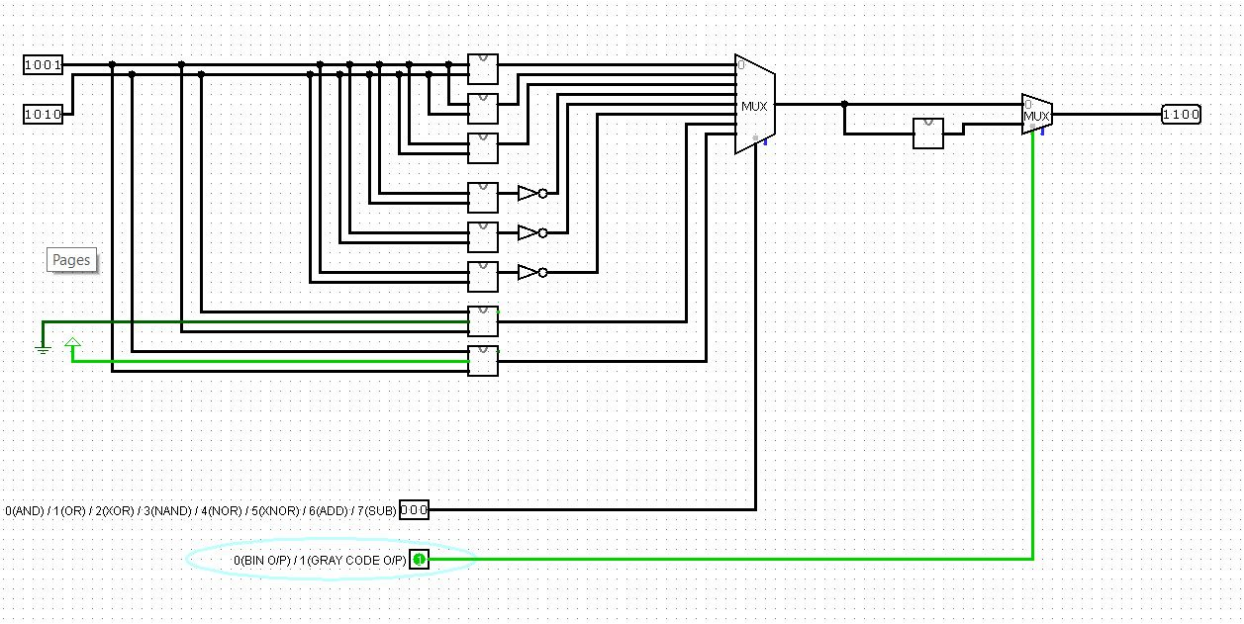
   Input: A : 1001

   B: 1010

   With select bits set to : 000

   Output : (with output-select bit 0:binary output) 1000

   (with output-select bit 1:gray code output) 1100



2. OR operation

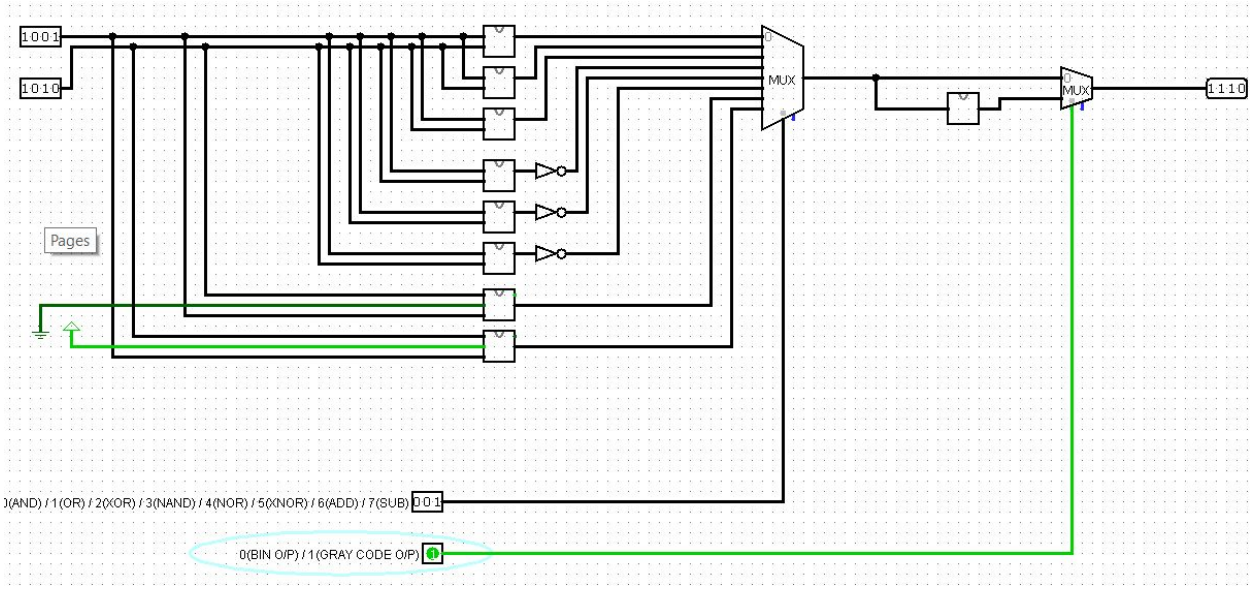
   Input: A : 1001

   B: 1010

   With select bits set to : 001

   Output : (with output-select bit 0:binary output) 1011

(with output-select bit 1:gray code output) 1110



3. XOR operation

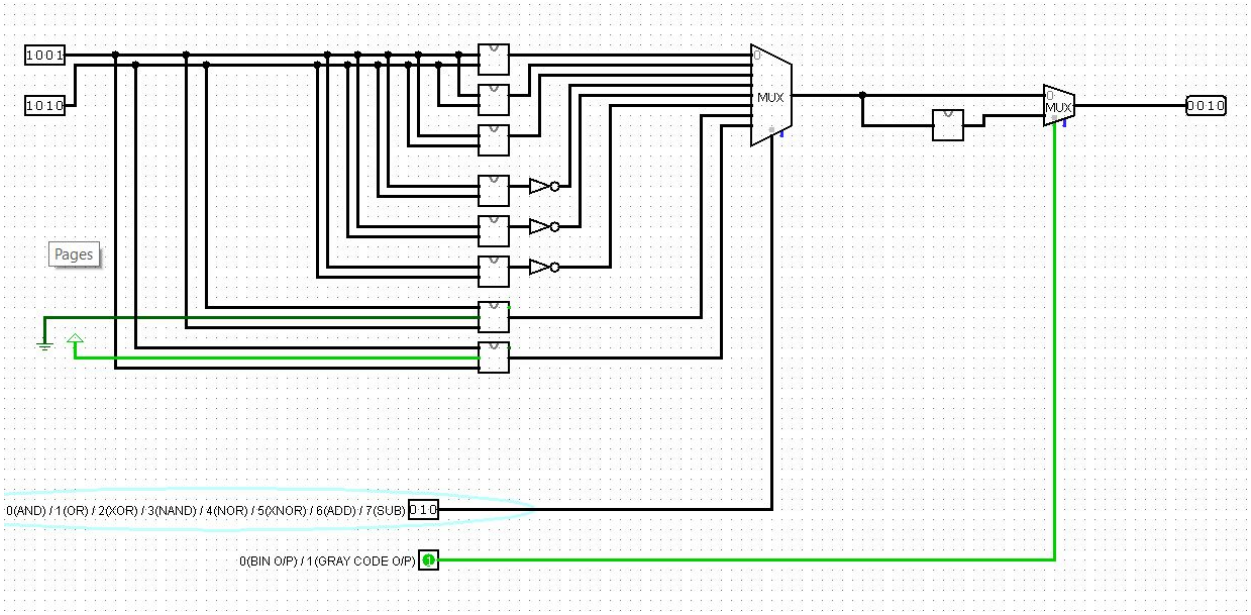
Input: A : 1001

B: 1010

With select bits set to : 010

Output : (with output-select bit 0:binary output) 0011

(with output-select bit 1:gray code output) 0010



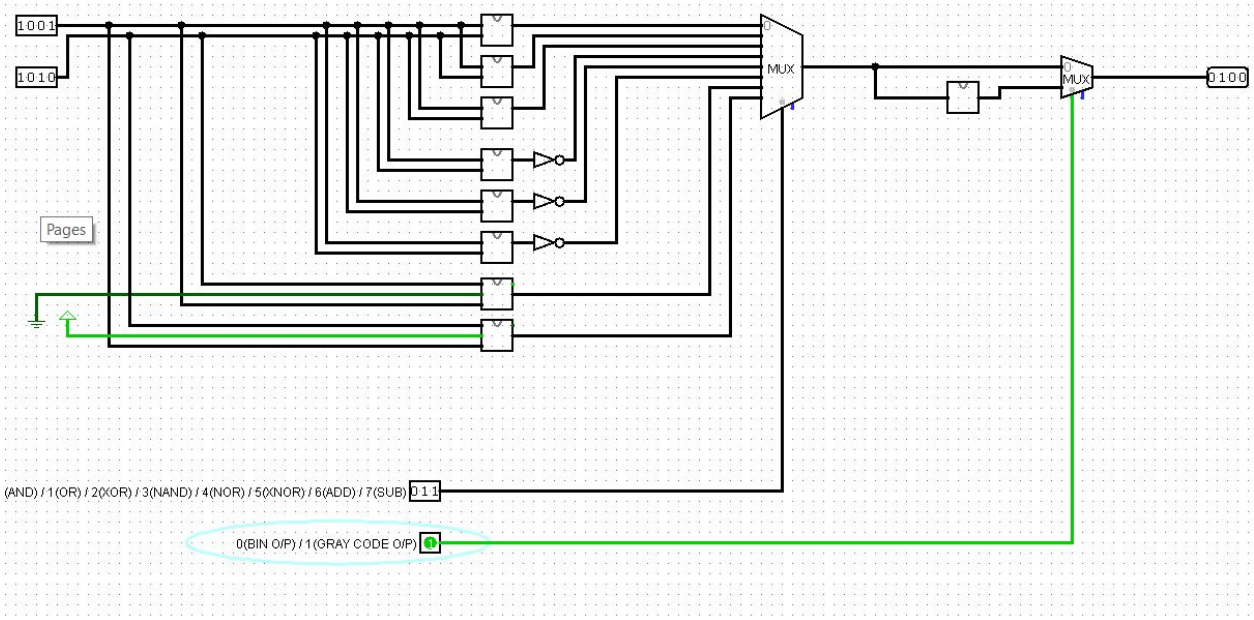
4. NAND operation

Input: A : 1001

B: 1010

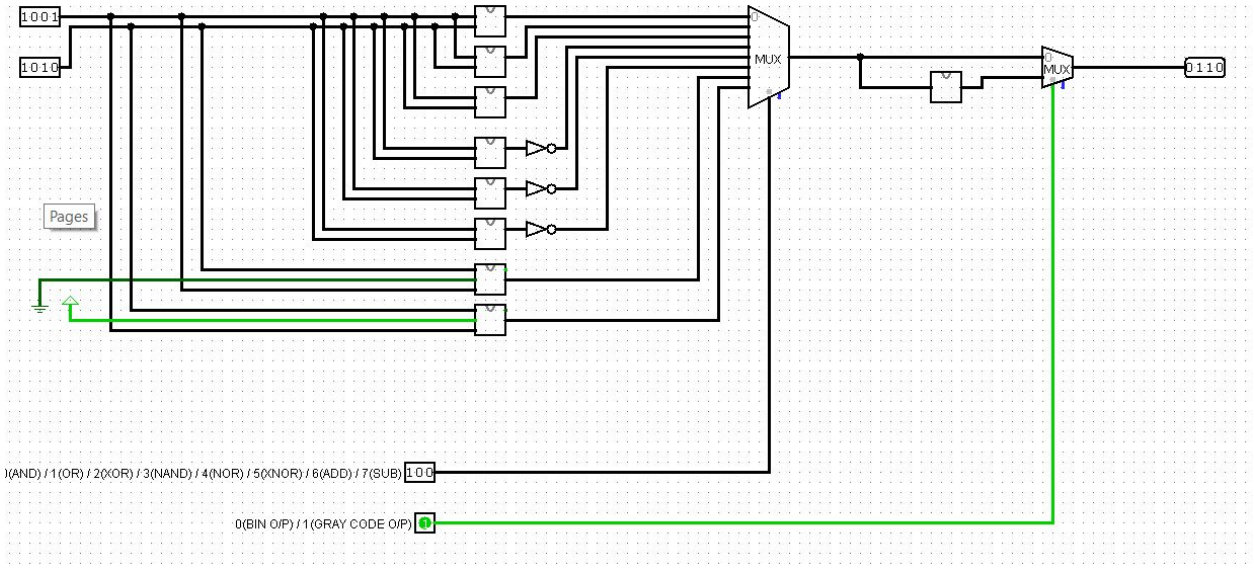
With select bits set to : 011

Output : (with output-select bit 0:binary output) 0111
(with output-select bit 1:gray code output) 0100



5. NOR operation

Input: A : 1001
B: 1010
With select bits set to : 100
Output : (with output-select bit 0:binary output) 0100
(with output-select bit 1:gray code output) 0110



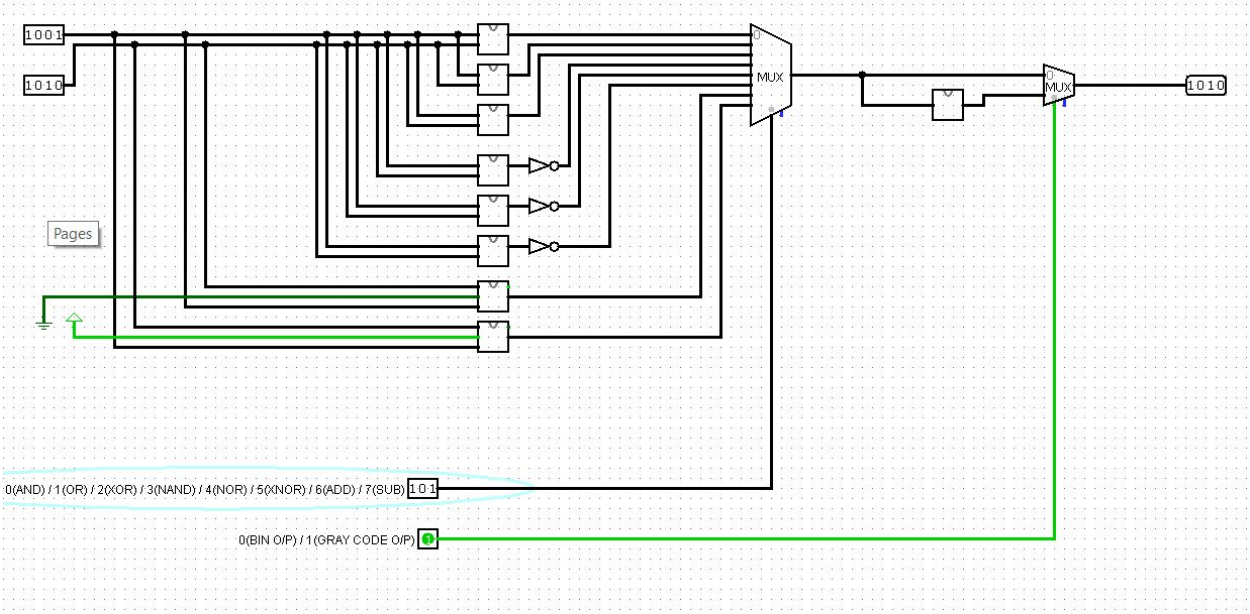
6. X NOR operation

Input: A : 1001
B: 1010

With select bits set to : 101

Output : (with output-select bit 0:binary output) 1100

(with output-select bit 1:gray code output) 1010



7. ADD operation

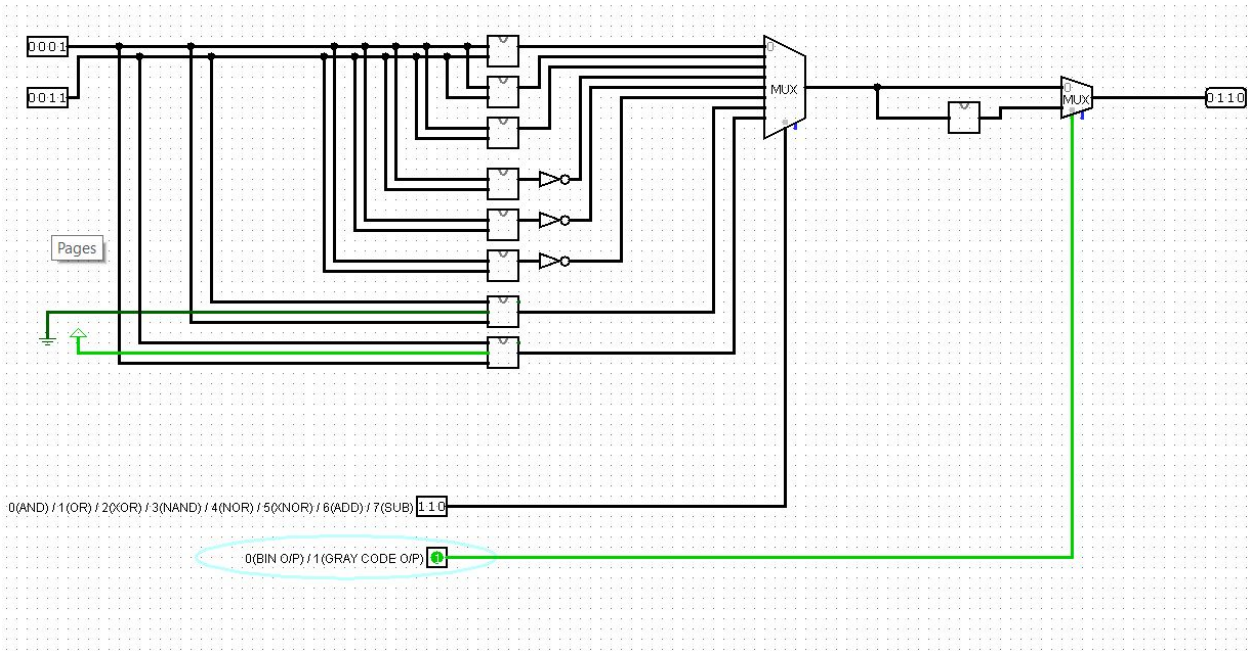
Input: A : 0001

B: 0011

With select bits set to : 110

Output : (with output-select bit 0:binary output) 0100

(with output-select bit 1:gray code output) 0110



8. SUB operation

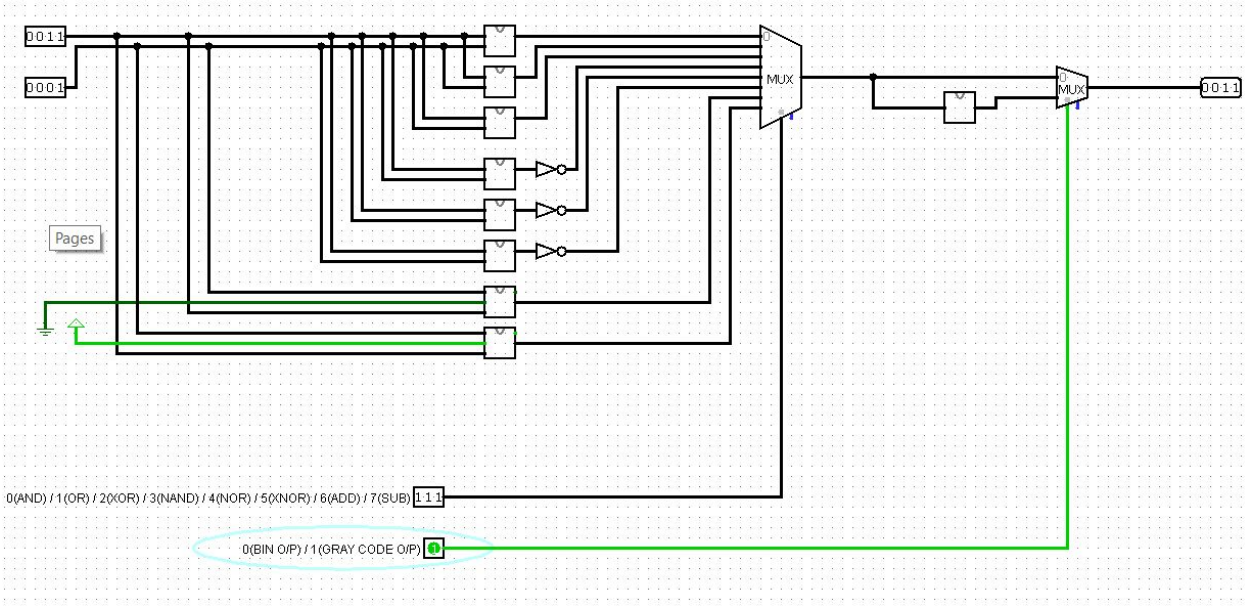
Input: A : 0011

B: 0001

With select bits set to : 111

Output : (with output-select bit 0:binary output) 0010

(with output-select bit 1:gray code output) 0011



Ultimately the expected results from the experiment were obtained and the functionality of the ALU was verified.