BIRZEIT UNIVERSITY

ENCS2340

Project Report

1st semester 22/23

Instructor: Dr. Anjad Badran

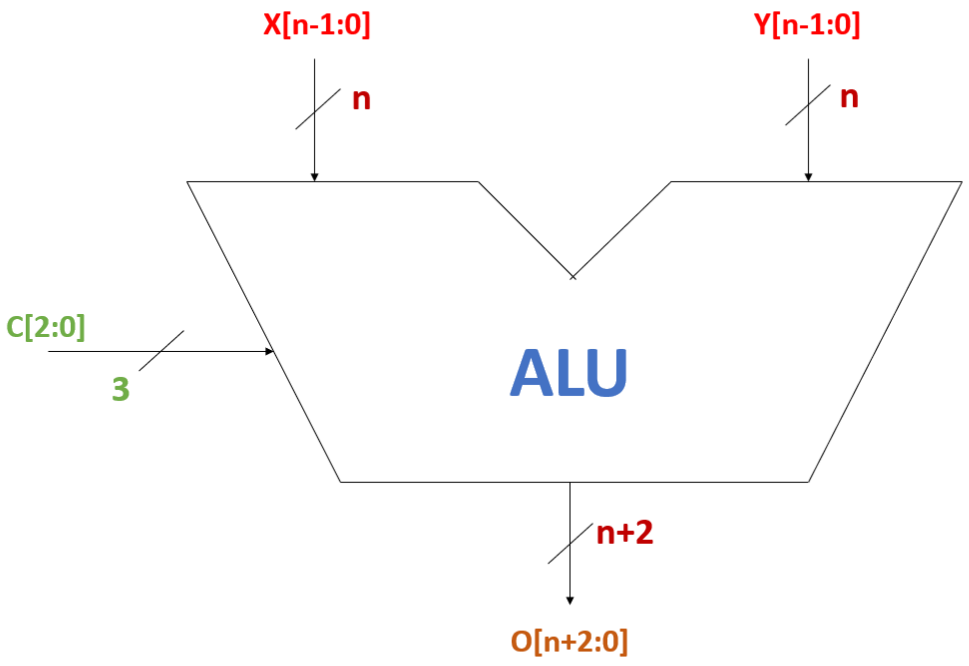
Prepared by: Obayda Sarraj

ID: 1211128

Table of contents

Introduction

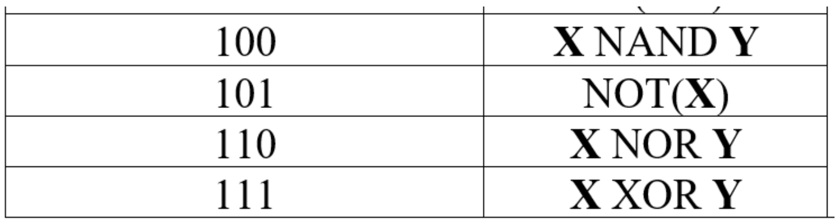
Arithmetic Logic Unit (**ALU**) **:** is a combinational digital circuit that performs arithmetic and bitwise operations on integer binary numbers. This is in contrast to a floating-point unit (FPU), which operates on floating point numbers. An ALU is a fundamental building block of many types of computing circuits, including the central processing unit (CPU) of computers, FPUs, and graphics processing units (GPUs). A single CPU, FPU or GPU may contain multiple ALUs.



a) Specify the size of the output (**O)** in bits so the overflow can never occur.

The ALU process two types of operations:

1- Logic operations the size of output stays the same of the input because there is no overflow in **all** logic operations , because every one bit from X&Y are return one bit as a result .



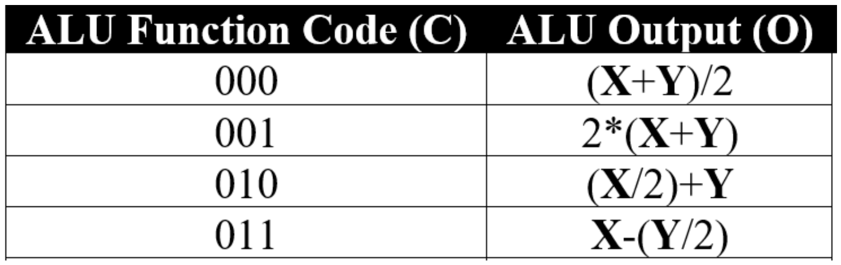
100	**X** NAND **Y**
101	NOT(**X**)
110	**X** NOR **Y**
111	**X** XOR **Y**

Example for the **NOR** gate (Assume n=4 , X=10 ,Y=8 ,O=output):

```
X :     1    0    1    0
Y :     1    0    0    0
--------------------------
    O :      0    1    0    1
```

Notice that there is no overflow.

2- Arithmetic operations , there is an overflow and the output size increase as follows:



ALU Function Code (C)	ALU Output (O)
000	(**X**+**Y**)/2
001	2*(**X**+**Y**)
010	(**X**/2)+**Y**
011	**X**-(**Y**/2)

-If we sum two inputs we need an other bit(digit) for an overflow (as in case 1,2,3)

Example (Assume n=4 , X=10 ,Y=8):

X+Y = 10+8

```
X :     1    0    1    0
Y :   1    0    0    0      +
--------------------------
   O : 1    0    0    1    0
```

-If we multiply the inputs by 2 we need an other bit(digit) for an overflow (as in case 2)

Example (Assume n=4 , X=6):

6*2 = 12

X : 0 1 1 0

$\Rightarrow$ O : 0 1 1 0 **0**

-If we subtract two inputs we need an other bit(digit) for an overflow (as in case 4)

Example (Assume n=**3** , X=-6 ,Y=-4):

X-Y = -6-4

2's comp X : 0 0 1 0

2's comp Y : 0 1 0 0 **+**

0 1 1 0

O : $\underline{\mathbf{1}}$ 0 1 0

-If we divide two inputs we do not need an other bit(digit) for an overflow (as in case 1,3,4)

With looking at the table the size of output will be :

Case: (X+Y)/2 → n+1

Case: 2*(X+Y) → n+2

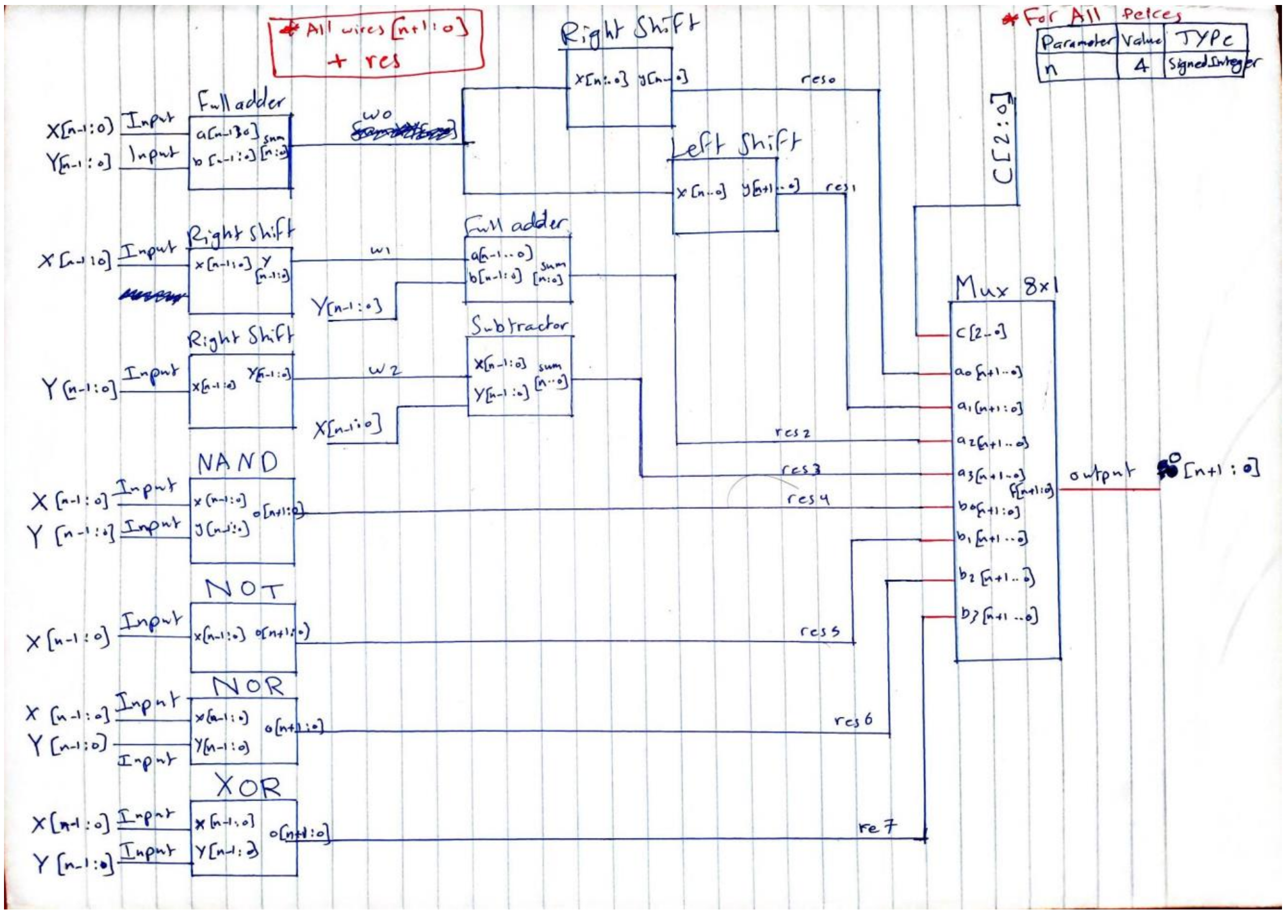
Case: (X/2)+Y → n+1

Case: X-(Y/2) → n+1

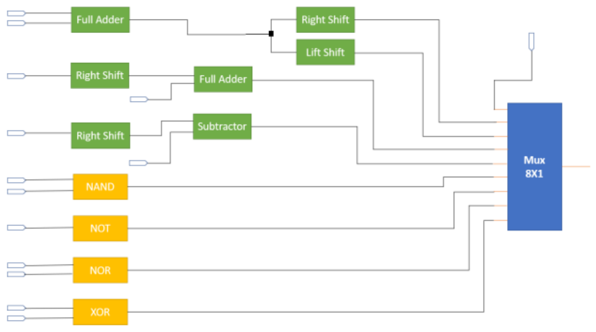
Hence , the answer is : ***n+2***

b) Show the ALU implementation using medium-scale integration (MSI) components and minimum number of gates (i.e. in blocks with their sizes). Note that, you might use some kind of extension (sign or zero-extension).

With details



Without details

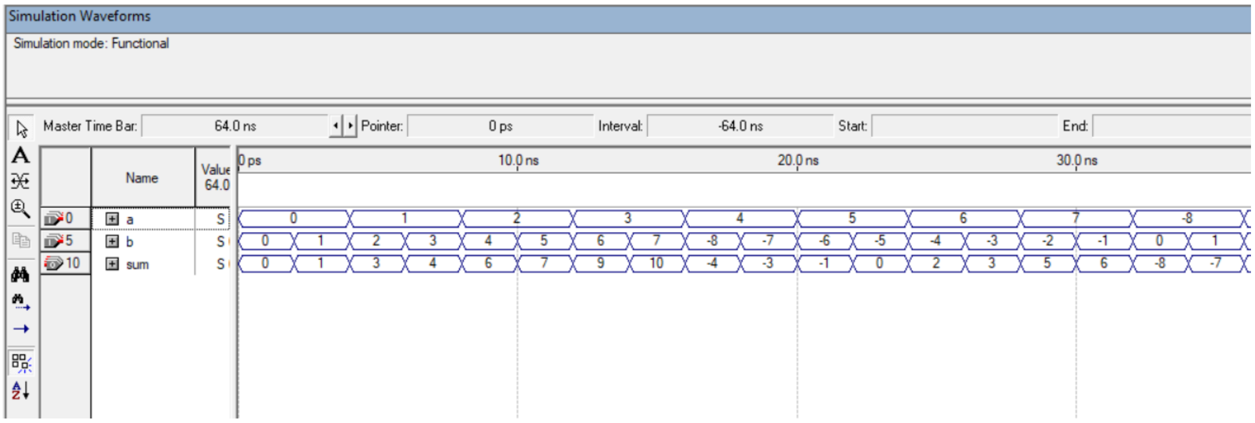


c) Write behavioral Verilog modules for your elements you defined in Part (b). Be noted that the size of every element you define should be **parameterized**, so that you can vary the design during the testing phase.

Full Adder :

```
module FullAdder_1211128 #(parameter n = 4)/*parameterization*/ (a, b, sum);

input signed [n-1:0] a, b;        //declaring inputs
output reg signed [n:0] sum;      //declaring outputs

always @ (a, b)
begin
     sum = a + b;                 //addition operation
end

endmodule
```

Simulation :



Simulation Waveforms

Simulation mode: Functional

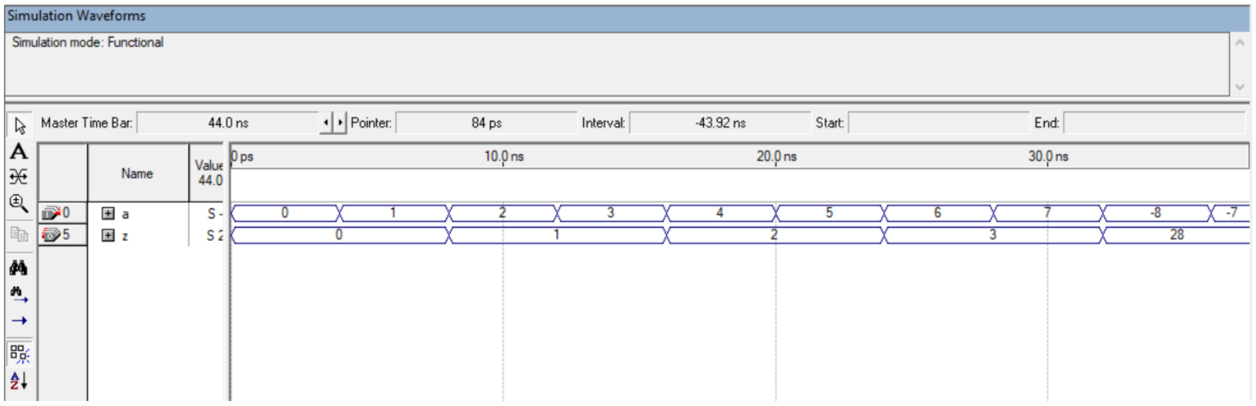
Master Time Bar: 64.0 ns   Pointer: 0 ps   Interval: -64.0 ns   Start: End:

```
        Name   Value 64.0   0 ps              10.0 ns             20.0 ns             30.0 ns
0       a      S    0    1    2    3    4    5    6    7    -8
5       b      S    0  1  2  3  4  5  6  7  -8  -7  -6  -5  -4  -3  -2  -1  0  1
10      sum    S    0  1  3  4  6  7  9  10  -4  -3  -1  0  2  3  5  6  -8  -7
```

Right shifter :

```
module R_Shifter_1211128 #(parameter n = 4)/*parameterization*/ (a, z);

input signed [n-1:0] a ;          //declaring inputs
output reg signed [n+1:0]z;       //declaring outputs

always@ (a)
begin
    z = a >> 1 ;                  //dividing(shifting) operation
end

endmodule
```

Simulation :


Simulation Waveforms

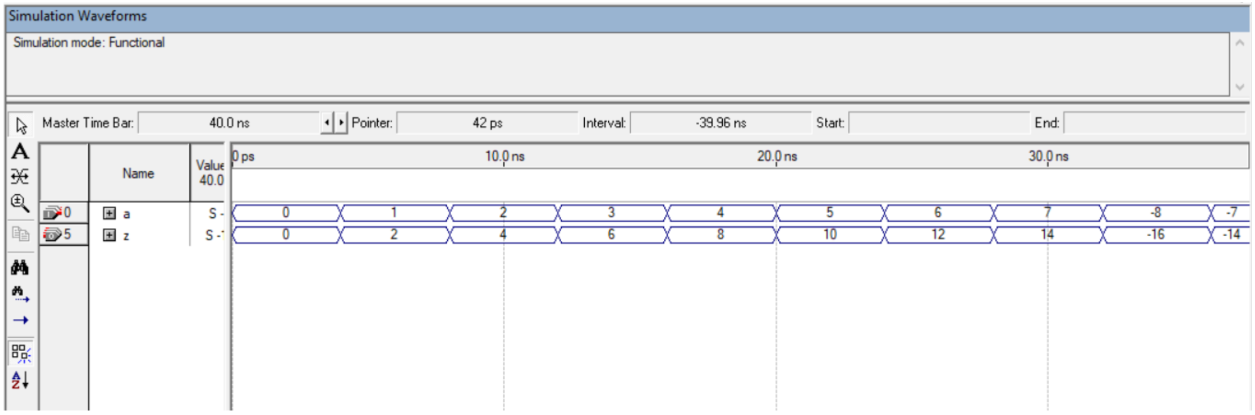
Simulation mode: Functional

Master Time Bar: 44.0 ns   Pointer: 84 ps   Interval: -43.92 ns   Start: End:

Left shifter :

```
module L_Shifter_1211128 #(parameter n = 4)/*parameterization*/ (a, z);

input signed [n-1:0] a ;          //declaring inputs
output reg signed [n+1:0]z;       //declaring outputs

always@ (a)
begin
    z = a << 1 ;                  //mutiplying(shifting) operation
end

endmodule
```

Simulation :

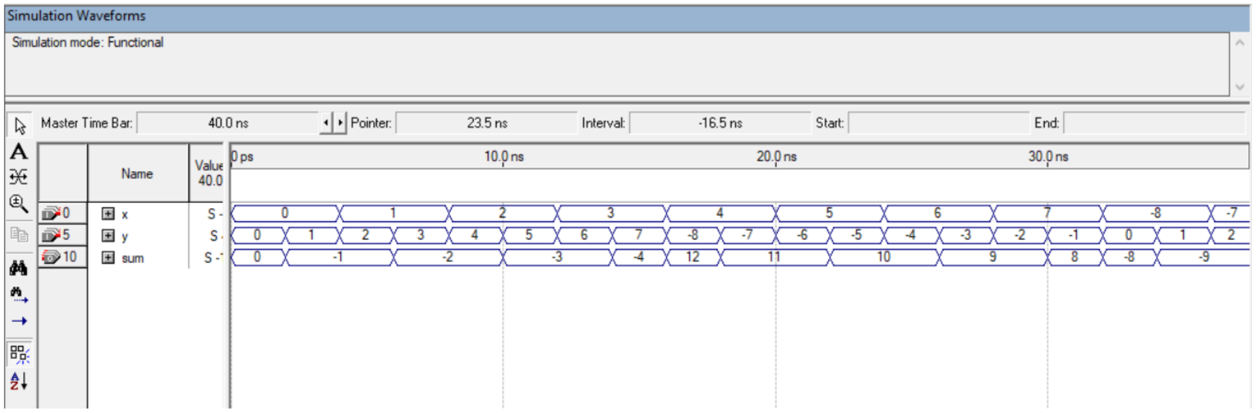

Simulation Waveforms
Simulation mode: Functional

```
Master Time Bar: 40.0 ns   Pointer: 42 ps   Interval: -39.96 ns   Start:   End:
Name   Value 40.0   0 ps   10.0 ns   20.0 ns   30.0 ns
a      S -   0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | -8 | -7
z      S -   0 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | -16 | -14
```

Subtractor :

```
module Subtractor_1211128 #(parameter n = 4)/*parameterization*/ (x,y, sum);

input signed [n-1:0] x, y;          //declaring inputs
output reg signed [n:0] sum;        //declaring outputs

always @ (x,y)
begin
     sum = x - y ;                  //subtraction operation
end

endmodule
```

Simulation :

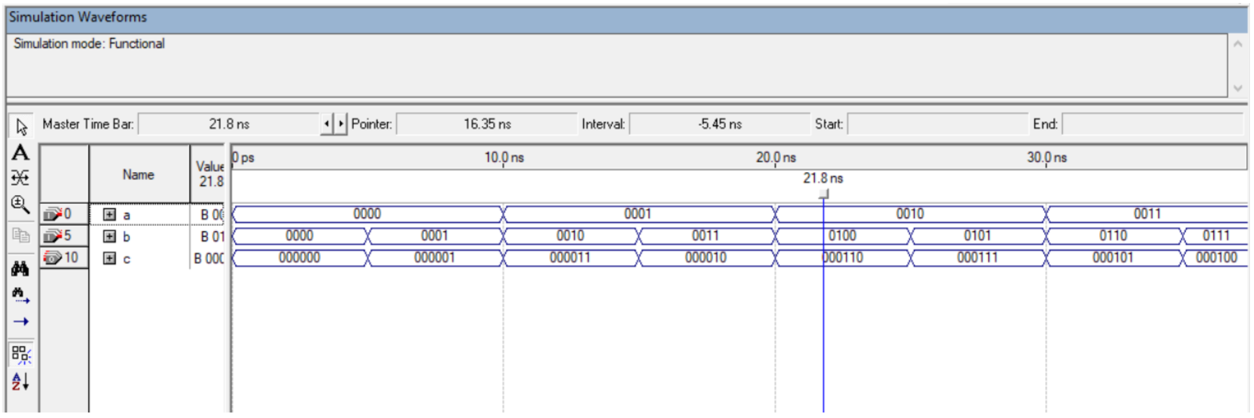


```
Simulation Waveforms
Simulation mode: Functional

Master Time Bar: 40.0 ns   Pointer: 23.5 ns   Interval: -16.5 ns   Start:   End:

      Name   Value 40.0   0 ps        10.0 ns        20.0 ns        30.0 ns
0     x      S -    0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | -8 | -7
5     y      S -    0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 | 0 | 1 | 2
10    sum    S -    0 | -1 | -2 | -3 | -4 | 12 | 11 | 10 | 9 | 8 | -8 | -9
```

Bitwise XOR :

```
module XOR_1211128 #(parameter n = 4) /*parameterization*/ (a,b,c) ;

input [n-1:0] a,b ;          //declaring inputs
output reg [n+1:0]c ;        //declaring outputs

always@ (a,b)
    begin
        c = a ^ b ;          //XOR operation

    end

endmodule
```

Simulation :



Simulation Waveforms

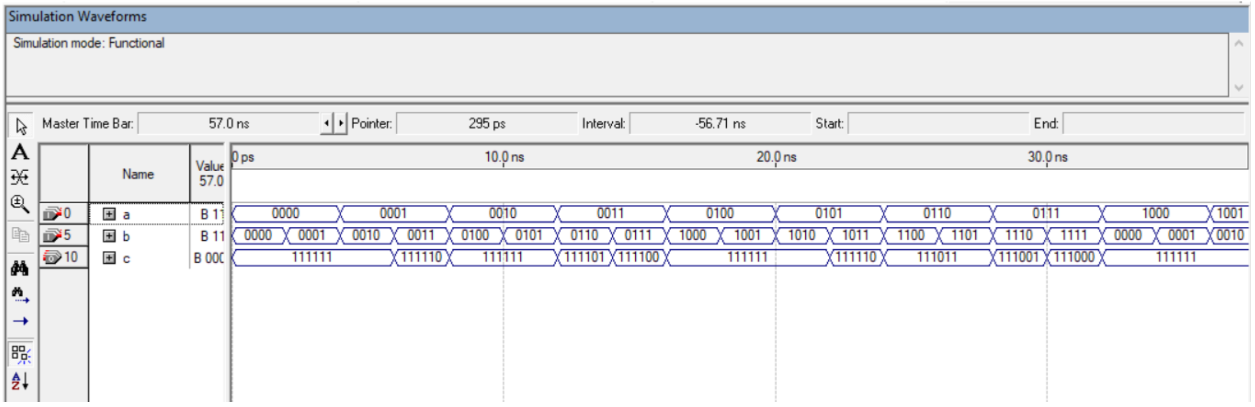
Simulation mode: Functional

Master Time Bar: 21.8 ns   Pointer: 16.35 ns   Interval: -5.45 ns   Start: End:

Bitwise NAND :

```
module NAND_1211128 #(parameter n = 4) /*parameterization*/ (a,b,c) ;

input signed [n-1:0] a,b ;          //declaring inputs
output  reg signed[n+1:0]c ;        //declaring outputs

always@ (a,b)
    begin
        c = ~(a & b) ;              //NOT AND operation
    end

endmodule
```

Simulation:



Simulation Waveforms

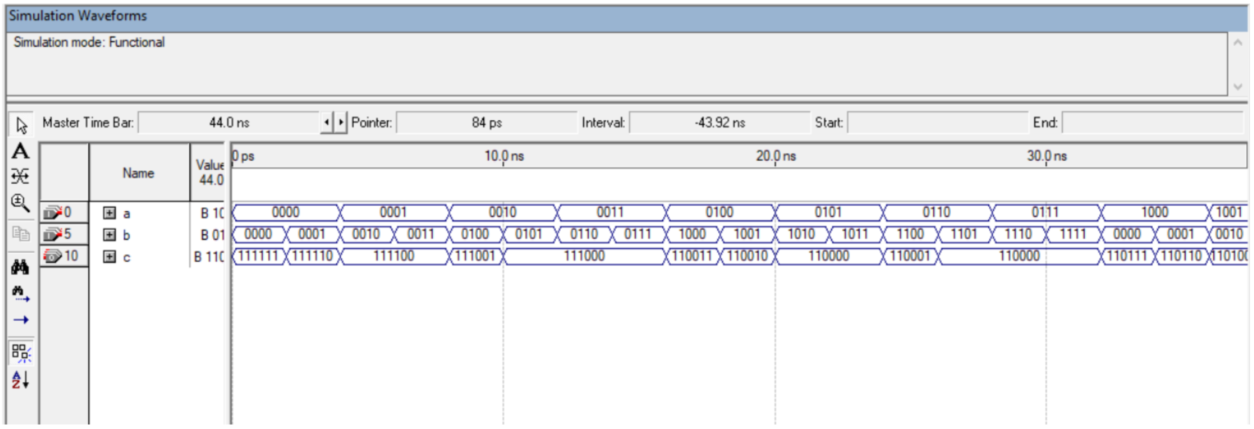
Simulation mode: Functional

Master Time Bar: 57.0 ns   Pointer: 295 ps   Interval: -56.71 ns   Start: End:

Bitwise NOR :

```
module NOR_1211128 #(parameter n = 4) /*parameterization*/ (a,b,c) ;

input [n-1:0] a,b ;          //declaring inputs
output reg [n+1:0]c ;        //declaring outputs

always@ (a,b)
    begin
        c = ~(a | b) ;       //NOT OR operation
    end

endmodule
```

Simulation :

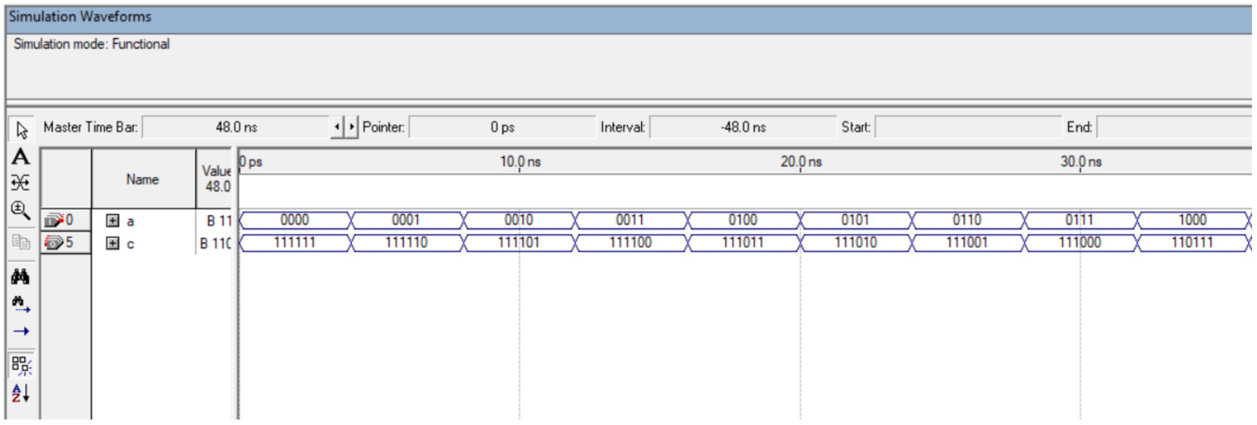


```
Simulation Waveforms
Simulation mode: Functional

Master Time Bar: 44.0 ns    Pointer: 84 ps    Interval: -43.92 ns    Start:    End:

      Name   Value 44.0   0 ps          10.0 ns          20.0 ns          30.0 ns
0     a      B 10   0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001
5     b      B 01   0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 | 0000 | 0001 | 0010
10    c      B 11   111111 | 111110 | 111100 | 111001 | 111000 | 110011 | 110010 | 110000 | 110001 | 110000 | 110111 | 110110 | 11010
```

Bitwise NOT :

```
module NOT_1211128 #(parameter n = 4) /*parameterization*/ (a,c) ;

input [n-1:0] a ;              //declaring inputs
output reg [n+1:0]c ;          //declaring outputs

always@ (a)
    begin
        c = ~(a) ;             //INVERTOR operation
    end

endmodule
```

Simulation :

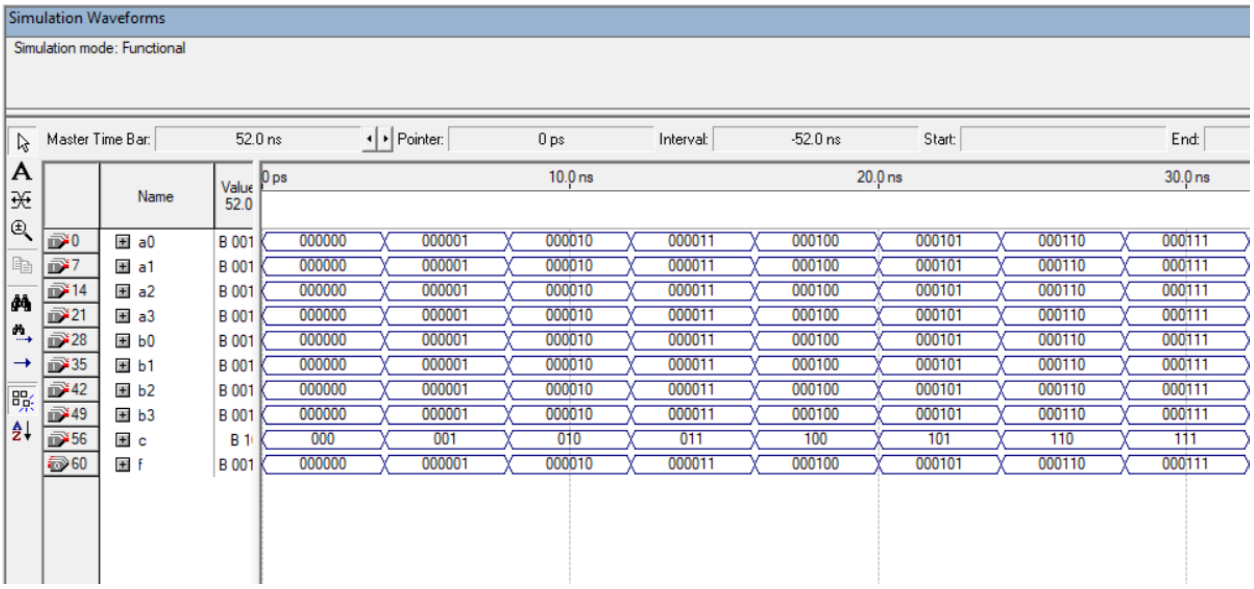


```
Simulation Waveforms
Simulation mode: Functional

Master Time Bar: 48.0 ns    Pointer: 0 ps    Interval: -48.0 ns    Start:    End:

     Name   Value 48.0   0 ps      10.0 ns           20.0 ns           30.0 ns
0    a      B 11     0000   0001   0010   0011   0100   0101   0110   0111   1000
5    c      B 110    111111 111110 111101 111100 111011 111010 111001 111000 110111
```

MUX 8x1 :

```
module MUX8x1_1211128 #(parameter n = 4) /*parameterization*/ (c,a0,a1,a2,a3,b0,b1,b2,b3,f) ;

input signed [n+1:0] a0,a1,a2,a3,b0,b1,b2,b3 ;       //declaring inputs
input [2:0] c ;
output reg signed [n+1:0] f ;                        //declaring outpus

always@ (c,a0,a1,a2,a3,b0,b1,b2,b3)
    begin

        if (c == 'b000)          //Setting values of the output depending on selection
        f = a0 ;
        else if (c == 'b001)
        f = a1 ;
        else if (c == 'b010)
        f = a2 ;
        else if (c == 'b011)
        f = a3 ;
        else if (c == 'b100)
        f = b0 ;
        else if (c == 'b101)
        f = b1 ;
        else if (c == 'b110)
        f = b2 ;
        else if (c == 'b111)
        f = b3 ;
    end

endmodule
```

Simulation :

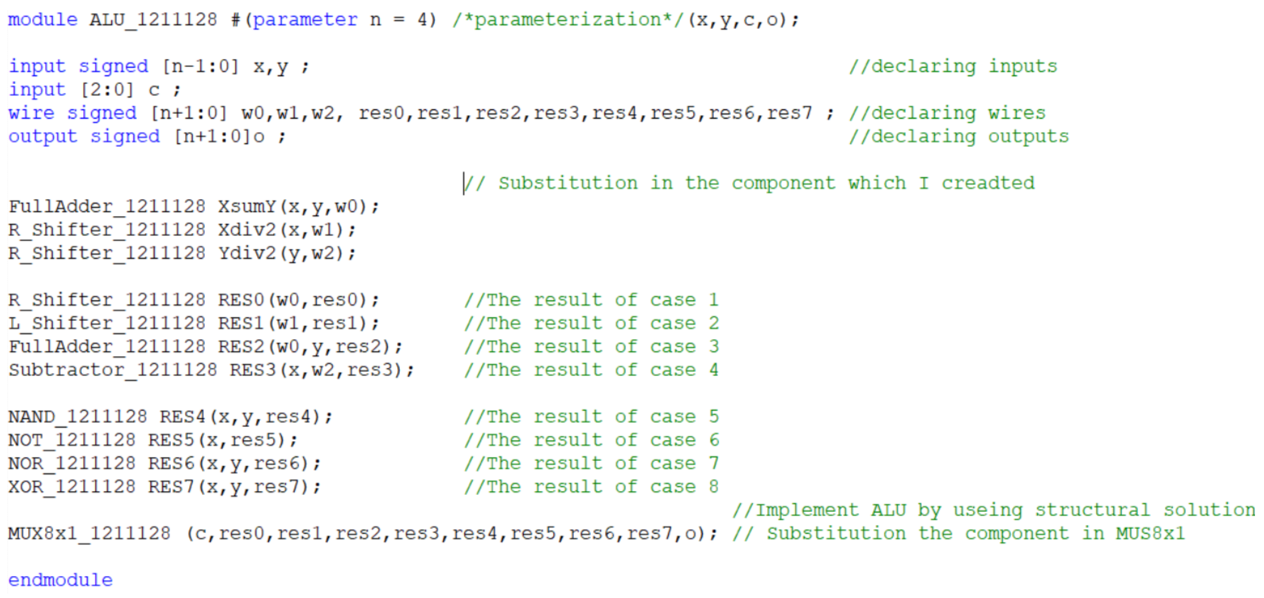

Simulation Waveforms
Simulation mode: Functional

Master Time Bar: 52.0 ns   Pointer: 0 ps   Interval: -52.0 ns   Start: End:

```
     Name  Value  0 ps                      10.0 ns                     20.0 ns                     30.0 ns
0    a0    B 001  000000  000001  000010  000011  000100  000101  000110  000111
7    a1    B 001  000000  000001  000010  000011  000100  000101  000110  000111
14   a2    B 001  000000  000001  000010  000011  000100  000101  000110  000111
21   a3    B 001  000000  000001  000010  000011  000100  000101  000110  000111
28   b0    B 001  000000  000001  000010  000011  000100  000101  000110  000111
35   b1    B 001  000000  000001  000010  000011  000100  000101  000110  000111
42   b2    B 001  000000  000001  000010  000011  000100  000101  000110  000111
49   b3    B 001  000000  000001  000010  000011  000100  000101  000110  000111
56   c     B 1    000     001     010     011     100     101     110     111
60   f     B 001  000000  000001  000010  000011  000100  000101  000110  000111
```

d) Write a structural Verilog model for your ALU designed in Part (b) using the elements you defined in Part (c).

ALU structural code



```
module ALU_1211128 #(parameter n = 4) /*parameterization*/(x,y,c,o);

input signed [n-1:0] x,y ;                                        //declaring inputs
input [2:0] c ;
wire signed [n+1:0] w0,w1,w2, res0,res1,res2,res3,res4,res5,res6,res7 ; //declaring wires
output signed [n+1:0]o ;                                          //declaring outputs

                                       // Substitution in the component which I creadted
FullAdder_1211128 XsumY(x,y,w0);
R_Shifter_1211128 Xdiv2(x,w1);
R_Shifter_1211128 Ydiv2(y,w2);

R_Shifter_1211128 RES0(w0,res0);       //The result of case 1
L_Shifter_1211128 RES1(w1,res1);       //The result of case 2
FullAdder_1211128 RES2(w0,y,res2);     //The result of case 3
Subtractor_1211128 RES3(x,w2,res3);    //The result of case 4

NAND_1211128 RES4(x,y,res4);           //The result of case 5
NOT_1211128 RES5(x,res5);              //The result of case 6
NOR_1211128 RES6(x,y,res6);            //The result of case 7
XOR_1211128 RES7(x,y,res7);            //The result of case 8
                                                              //Implement ALU by useing structural solution
MUX8x1_1211128 (c,res0,res1,res2,res3,res4,res5,res6,res7,o); // Substitution the component in MUS8x1

endmodule
```

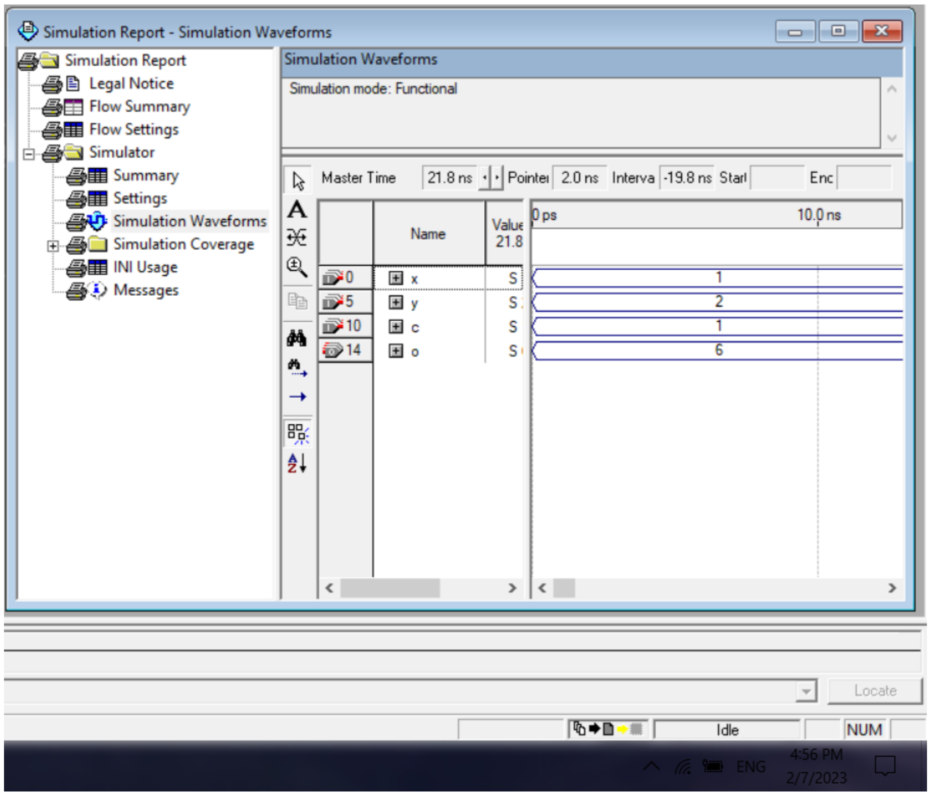
e) Generate the waveforms of the ALU defined in Part (d), assumes that X and Y are 4-bits and their values based on your student ID should be set as follows:

Note: If any value from the set {$C_2, Y_2, X_2, C_1, Y_1, X_1$} is 8 or 9, you need to replace it by 1

By note, my ID is 1211128 → 1211121 (**1$C_2Y_2X_2C_1Y_1X_1$**)

Test	X	Y	C	Expected **O**
1	$X_1 = 1$	$Y_1 = 2$	$C_1 = 1$	6
2	$X_2 = 1$	$Y_2 = 1$	$C_2 = 2$	1
3	$X_3 = -X_1$	$Y_3 = -Y_1$	$C_3 = C_2$	-2

First test :

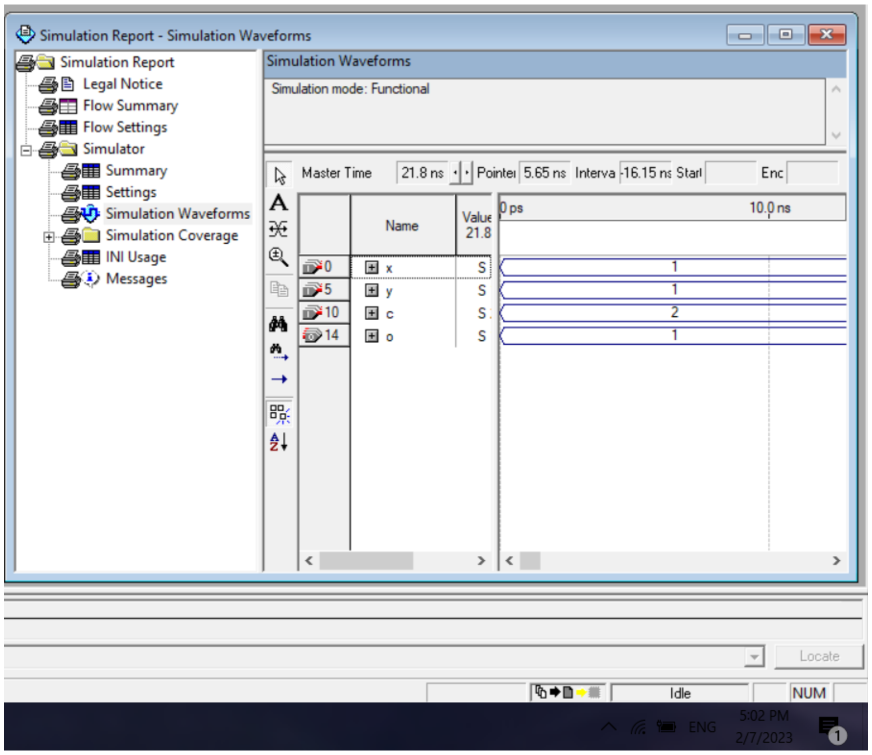


Simulation mode: Functional

Master Time 21.8 ns Pointer 2.0 ns Interval -19.8 ns

	Name	Value 21.8	0 ps … 10.0 ns
0	x	S	1
5	y	S	2
10	c	S	1
14	o	S	6

Second test :

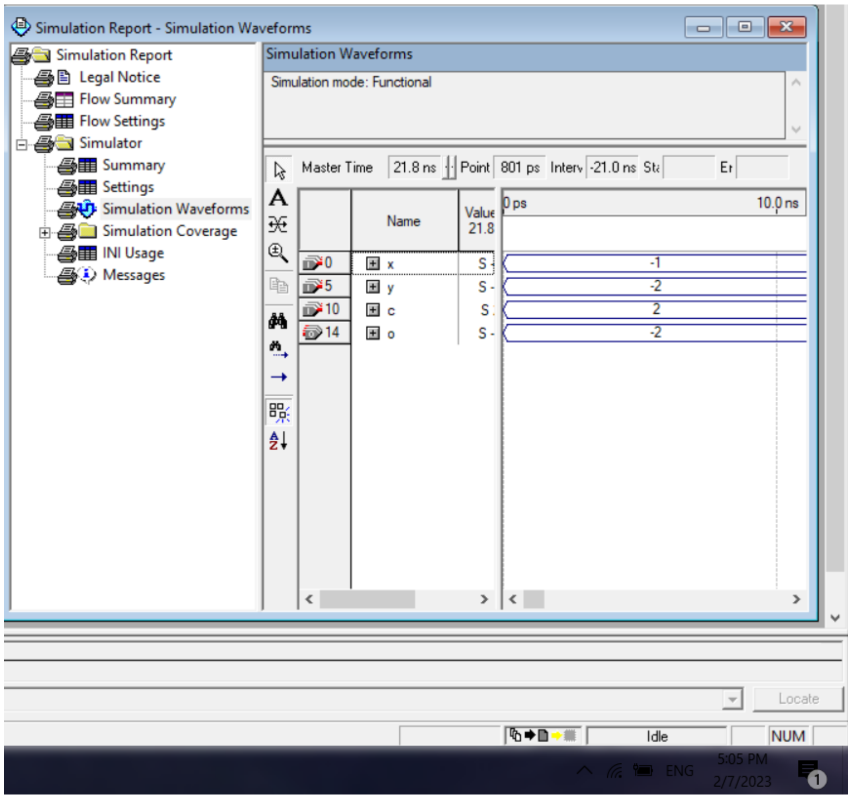

Simulation Report - Simulation Waveforms

Simulation mode: Functional

Master Time 21.8 ns   Pointer 5.65 ns   Interval -16.15 ns

	Name	Value 21.8
0	x	S 1
5	y	S 1
10	c	S 2
14	o	S 1

Third test :


Simulation Report - Simulation Waveforms

Simulation mode: Functional

Master Time 21.8 ns   Pointer 801 ps   Interval -21.0 ns

	Name	Value 21.8
0	x	S -1
5	y	S -2
10	c	S 2
14	o	S -2

f) Write a single behavioral Verilog module that models the designed ALU.

ALU behavioral code

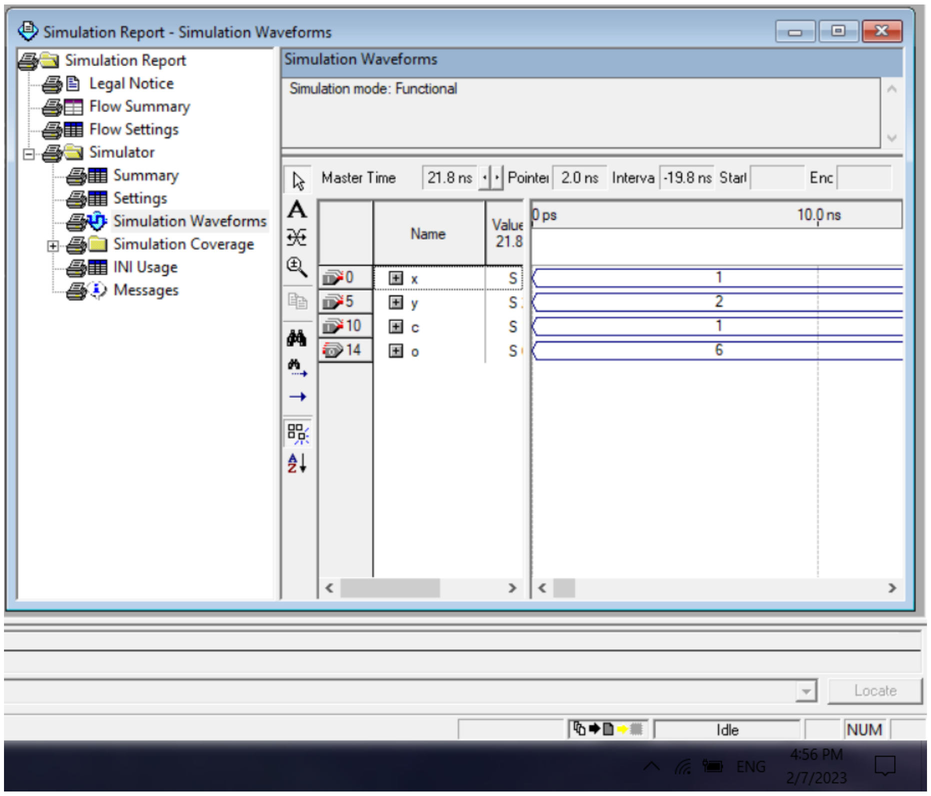
```
module ALU_behav_1211128 #(parameter n = 4) /*parameterization*/(x,y,c,o);

input signed [n-1:0] x,y ;           //declaring inputs
input [2:0] c ;
output reg signed [n+1:0] o ;        //declaring outputs

always@(*)
begin

    if(c == 3'b000)                  //Implement ALU by useing behavioral solution
        o = (x+y)/2 ;
    else if(c == 3'b001)
        o =  2*(x+y);
    else if(c == 3'b010)
        o = (x/2)+y ;
    else if(c == 3'b011)
        o = x-(y/2) ;
    else if(c == 3'b100)
        o = ~(x&y) ;
    else if(c == 3'b101)
        o = ~(x) ;
    else if(c == 3'b110)
        o = ~(x|y) ;
    else if(c == 3'b111)
        o = x ^ y ;
    else
        o = 0;

end
endmodule
```

g) Generate the waveforms of the behavioral ALU defined in Part (e), assumes that X and Y are 4-bits and their values based on your student ID should be set as follows:

Note: If any value from the set {$C_2, Y_2, X_2, C_1, Y_1, X_1$} is 8 or 9, you need to replace it by 1

Test	X	Y	C	Expected **O**
1	$X_1 = 1$	$Y_1 = 2$	$C_1 = 1$	6
2	$X_2 = 1$	$Y_2 = 1$	$C_2 = 2$	1
3	$X_3 = -X_1$	$Y_3 = -Y_1$	$C_3 = C_2$	-2

First test :



Simulation Report - Simulation Waveforms

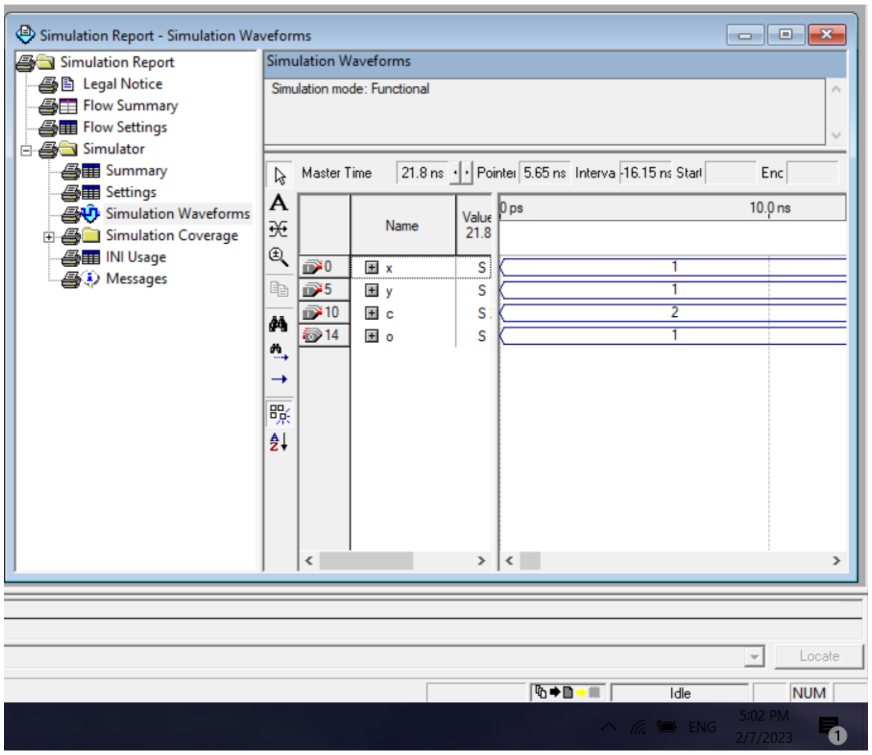
Simulation Waveforms

Simulation mode: Functional

Master Time 21.8 ns Pointer 2.0 ns Interval -19.8 ns Start End

	Name	Value 21.8	0 ps ... 10.0 ns
0	x	S	1
5	y	S	2
10	c	S	1
14	o	S	6

Second test :

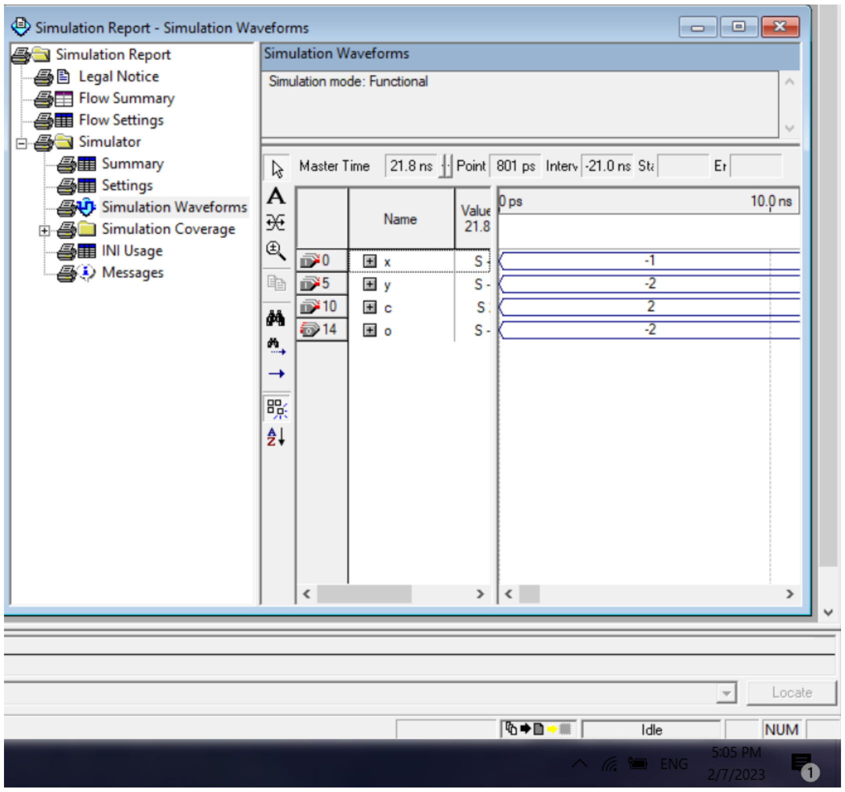

Simulation Report - Simulation Waveforms

Simulation mode: Functional

Master Time 21.8 ns   Pointer 5.65 ns   Interval -16.15 ns

	Name	Value 21.8	0 ps ... 10.0 ns
0	x	S	1
5	y	S	1
10	c	S	2
14	o	S	1

Third test :


Simulation Report - Simulation Waveforms

Simulation mode: Functional

Master Time 21.8 ns   Point 801 ps   Interv -21.0 ns

	Name	Value 21.8	0 ps ... 10.0 ns
0	x	S -	-1
5	y	S -	-2
10	c	S	2
14	o	S -	-2

Thank you …