BIRZEIT UNIVERSITY

Department of computer and electrical engineering

Digital Systems ENCS234

Verilog Project

Student name: Ody Shbayeh

Student number: 1201462

Instructor: Dr Mohammad Hussein

Date: 7/2/2023

Discussion and procedure

In this session it contain the answers for all parts with detailed explanation and supported by snapshots/figures, commands, designs, runs, tables, etc.

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

The size of the output = n+2 bit, in the following table it describe the size of the output of each operation.

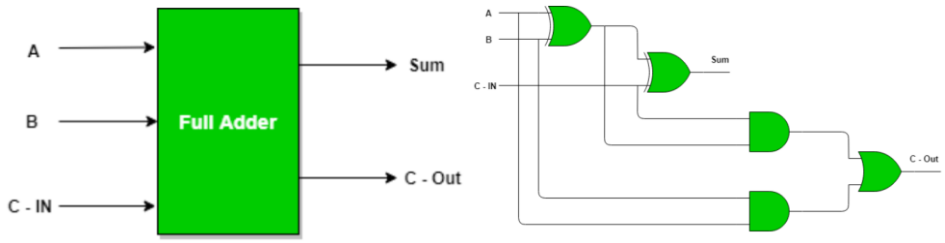
ALU Output (O)	Output size in bit
(X+Y)/2	N+1
2*(X+Y)	N+2
(X/2)+Y	N+1
X-(Y/2)	N+1
X NAND Y	N
NOT(X)	N
X NOR Y	N
X XOR Y	N

Note that the summation and the subtraction change the output size to n+1 bit due to the over flow bit, the division operation don't change the size and the multiplication by 2 will shift left and add 1 bit so it will change to n+1, as a result we find that the second operation 2*(X+Y) is the maximum size of output bits which is multiplication by 2 will increase the number of bit by 1 bit and the addition will add another bit so it will change it to n+2 bits and this is the maximum size of bits.

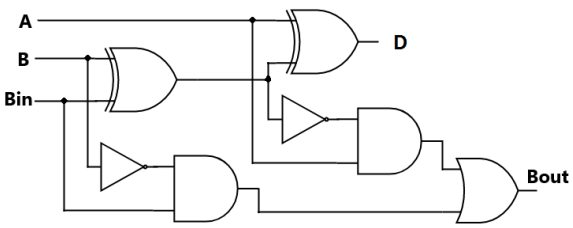
b. Show the ALU implementation using medium-scale integration (MSI) components and minimum number of gates (i.e. in blocks with their sizes). Note that, you might use some kind of extension (sign- or zero-extension).

1. Full adder

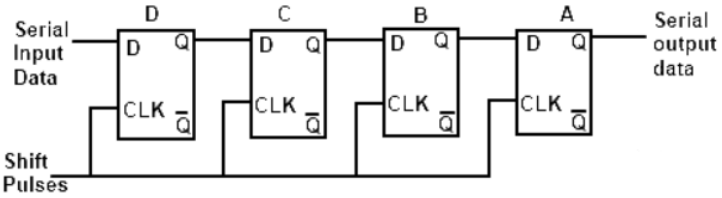
   To apply the addition and subtraction operation we will use full adder component.



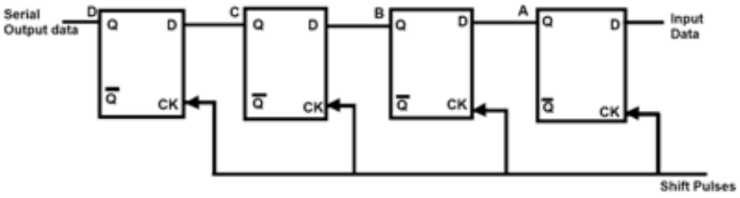
2. Full subtraction



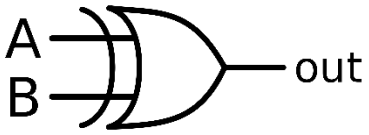
3. Right shift register
   To apply the division operation we will use the right shift register component.



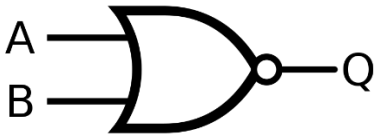
4. Left shift register
   To apply the multiplication operation we will use the left shift register component.



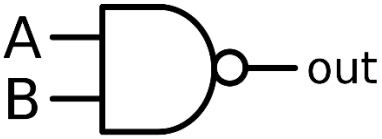
5. Xor



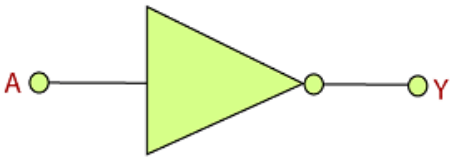
6. Nor



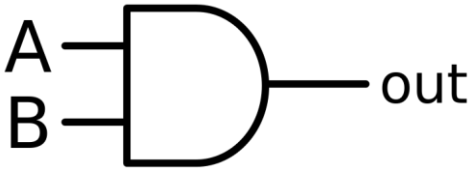
7. Nand



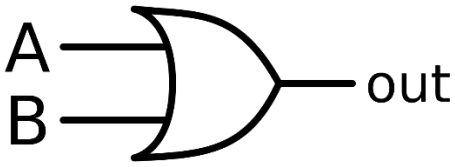
8. Not



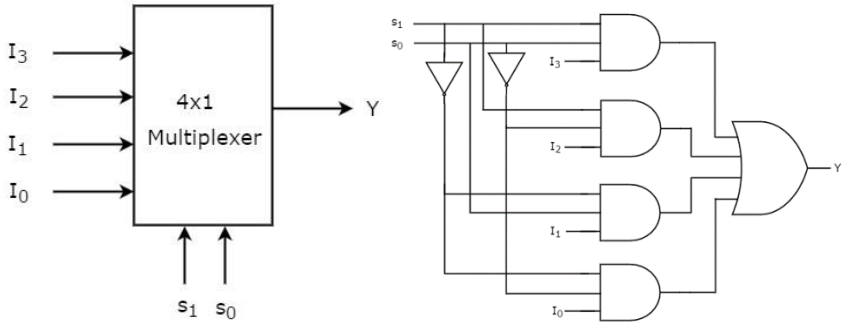
9. And



10. Or

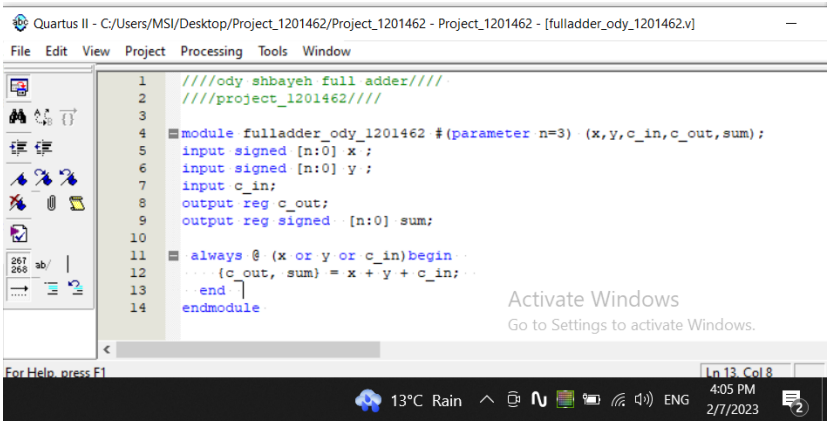


11. Mux 4X1

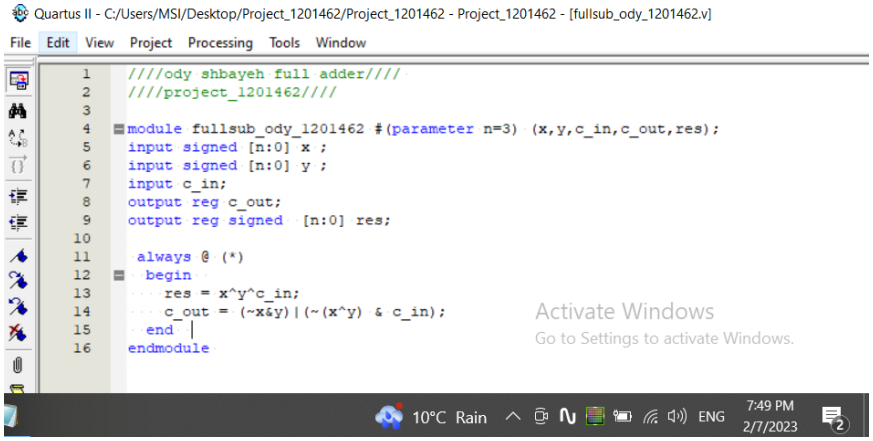


c. Write behavioral Verilog modules for your elements you defined in Part (b). Be noted that the size of every element you define should be parameterized, so that you can vary the design during the testing phase.

1. Full adder



2. Full subtraction



3. Left shift register

Quartus II - C:/Users/MSI/Desktop/Project_1201462/Project_1201462 - Project_1201462 - [leftshift_ody_1201462.v]

File Edit View Project Processing Tools Window

```
1  ////ody shbayeh left shift reg////
2  ////project_1201462////
3
4  module leftshift_ody_1201462 #(parameter n=3) (x,y);
5  input signed [n:0] x ;
6  output reg signed [n:0] y ;
7
8  always @(x)
9  begin
10
11 y = x*2;
12 end
13
14 endmodule
```


13°C Rain ENG 4:05 PM 2/7/2023

4. Right shift register

Quartus II - C:/Users/MSI/Desktop/Project_1201462/Project_1201462 - Project_1201462 - [rightshift_ody_12

File Edit View Project Processing Tools Window

```
1  ////ody shbayeh right shift reg////
2  ////project_1201462////
3
4  module rightshift_ody_1201462 #(parameter n=3) (x,y);
5  input signed [n:0] x ;
6  output reg signed [n:0] y ;
7
8  always @(x)
9  begin
10
11 y = x/2;
12 end
13
14 endmodule
```


13°C Rain ENG 4:07 PM 2/7/2023

5. Mux 8 to 1

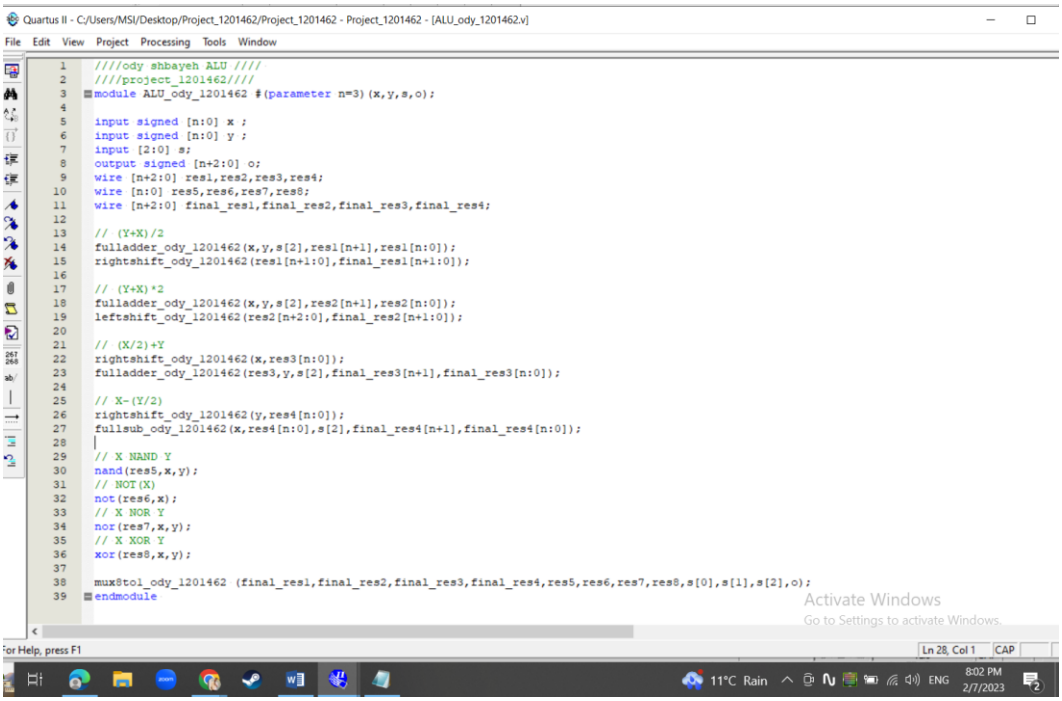
Quartus II - C:/Users/MSI/Desktop/Project_1201462/Project_1201462 - Project_1201462 - [mux8to1_ody_1201462.v]

File Edit View Project Processing Tools Window

```
1
2  ////ody shbayeh mux 8to1////
3  ////project_1201462////
4
5  module mux8to1_ody_1201462 #(parameter n=3)( a,b,c,d,e,f,g,h,s0,s1,s2, out);
6
7  input [n:0] a, b, c, d,e,f,g,h;
8  input wire s0, s1, s2;
9  output reg [n:0]out;
10
11 always @ (a or b or c or d or e or f or g or h ,s0, s1, s2)
12 begin
13
14 case (s0 | s1 | s2)
15 3'b000 : out <= a;
16 3'b001 : out <= b;
17 3'b010 : out <= c;
18 3'b011 : out <= d;
19 3'b100 : out <= e;
20 3'b101 : out <= f;
21 3'b110 : out <= g;
22 3'b111 : out <= h;
23
24
25 endcase
26
27 end
28 endmodule
29
```


10°C Rain ENG 7:50 PM 2/7/2023

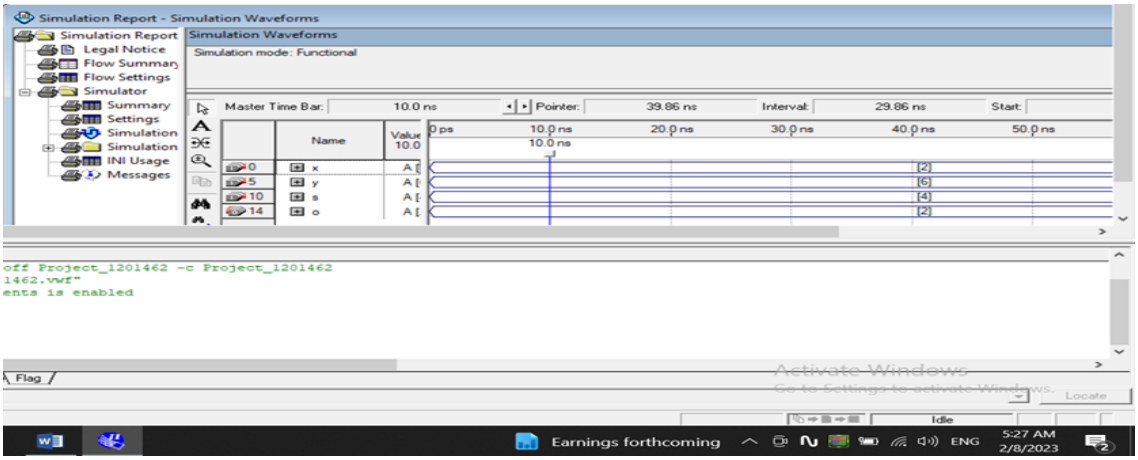
d. Write a structural Verilog model for your ALU designed in Part (b) using the elements you defined in Part (c).



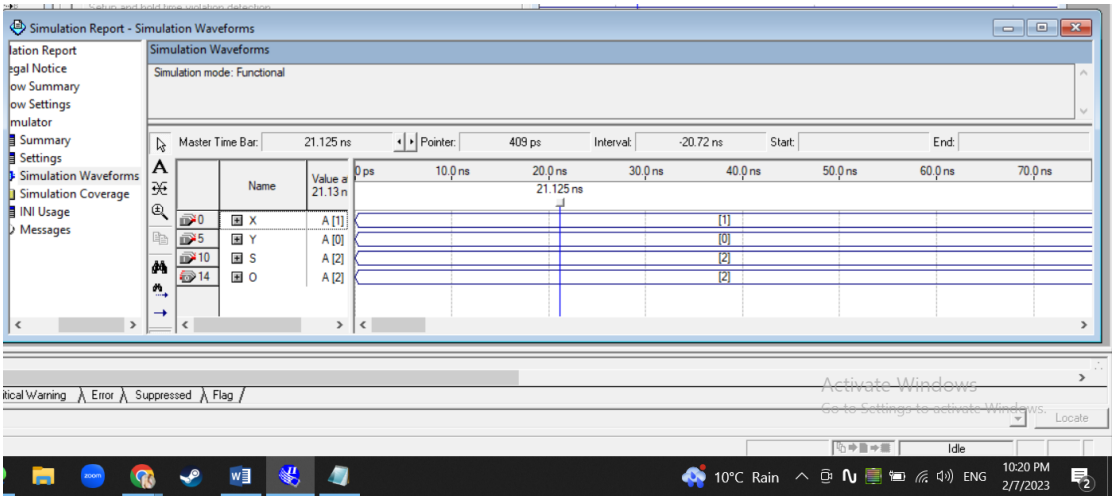
e. Generate the waveforms of the ALU defined in Part (d), assumes that X and Y are 4-bits and their values based on your student ID should be set as follows:

f. For case 1 which my university card is 1201462—x1=2,y1=6,c1=4

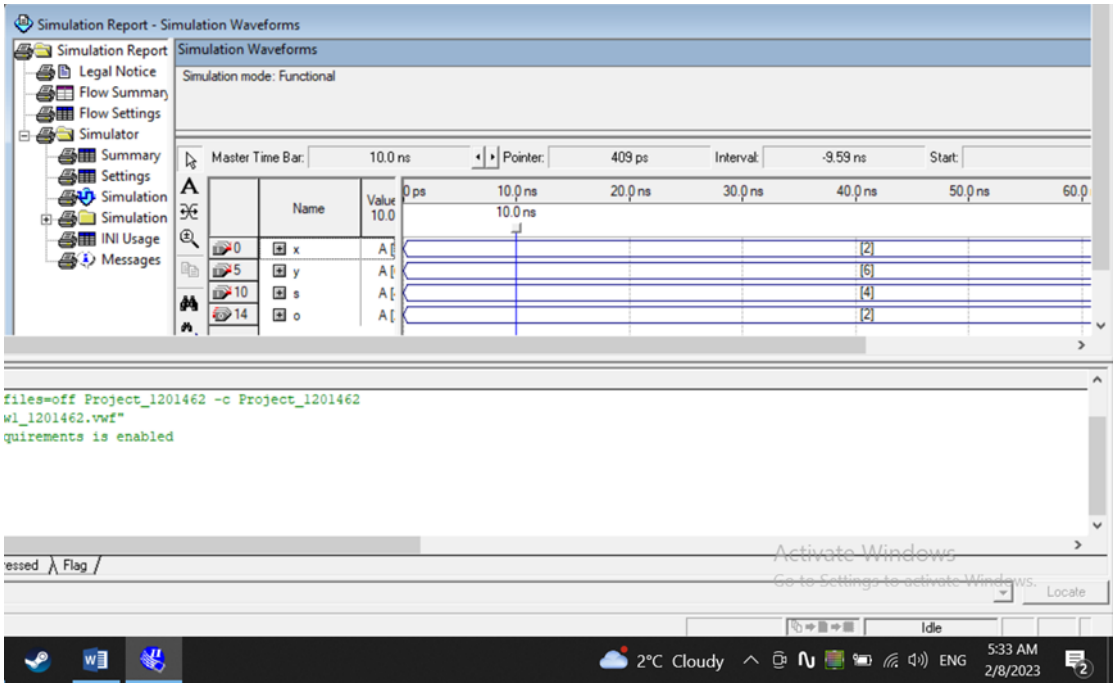
Results are



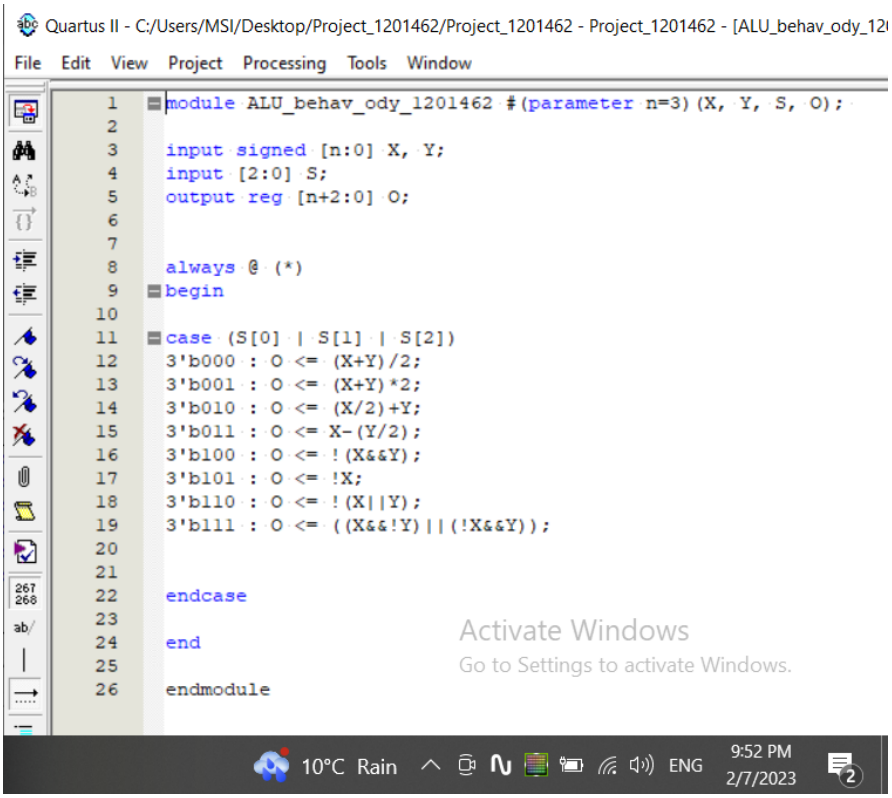
For case 2 which my university card is 1201462—x2=1,y2=0,c2=2
Results are



For case 3 which my university card is 1201462—x1=-2,y1=-6,c1=2
Results are

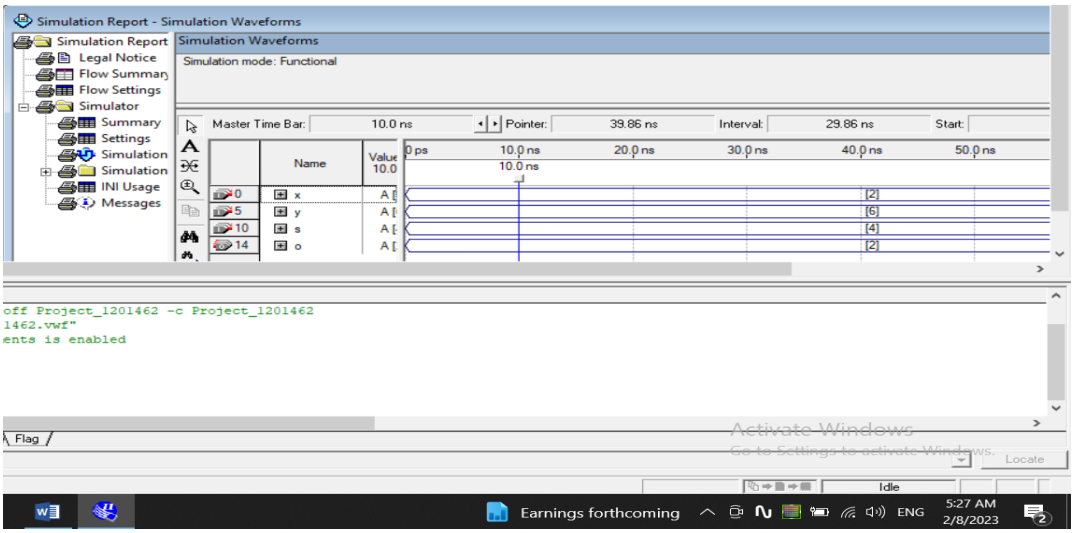


g. Write a single behavioral Verilog module that models the designed ALU.

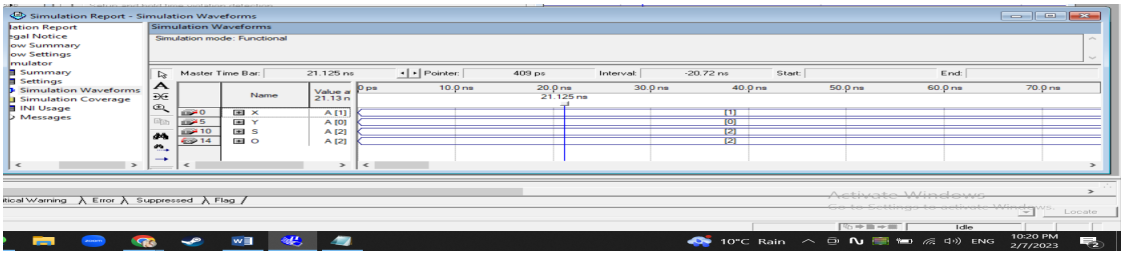


h. Generate the waveforms of the behavioral ALU defined in Part (f), assumes that X and Y are 4-bits and their values based on your student ID should be set as follows: The general representation of the student ID is 1C2Y2X2C1Y1X1, so, if your student ID is 1220520, then X, Y, and C values for the three test cases as follows:

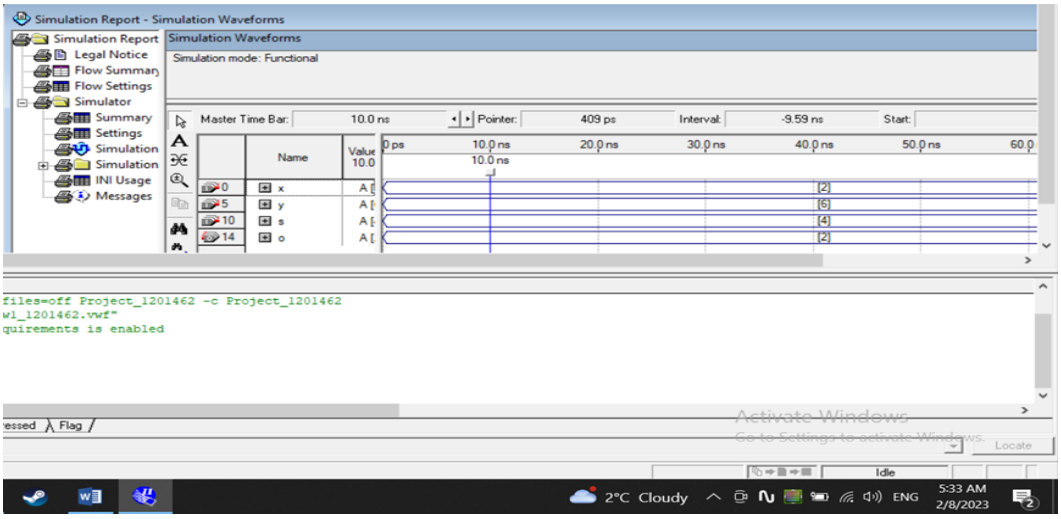
For case 1 which my university card is 1201462—x1=2,y1=6,c1=4
Results are

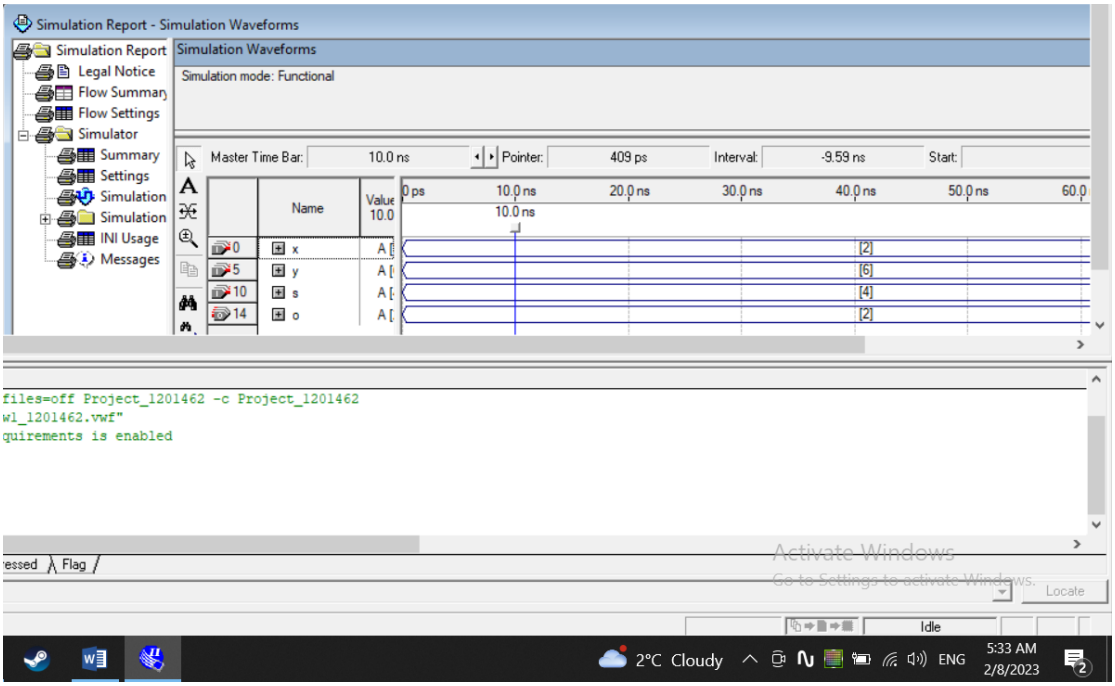


For case 2 which my university card is 1201462—x2=1,y2=0,c2=2
Results are



For case 3 which my university card is 1201462—x1=-2,y1=-6,c1=2
Results are



i. 

the results were the same but with the inverse signed number—

Appendix

Full adder Verilog code

```
module fulladder (  input [3:0] a,
                    input [3:0] b,
                    input c_in,
                    output reg c_out,
                    output reg [3:0] sum);


 always @ (a or b or c_in) begin
    {c_out, sum} = a + b + c_in;
  end
endmodule
```

Left shift register Verilog code

```
module leftshift_ody_1201462 #(parameter n=3) (x,y);///module
input signed [n:0] x ;///input
output reg signed [n+1:0] y ;//output

always @(x) ///shift left by 1 bit as a multiplication by 2
begin

;y = x*2
end


endmodule
```

Right shift register Verilog code

```
module rightshift_ody_1201462 #(parameter n=3) (x,y);///module
input signed [n:0] x ;//input
output reg signed [n:0] y ;//output

always @(x)//shift right as a division operation by 2
begin

;y = x/2
end

      endmodule
```

Mux 1 to 4 Verilog code

```
module mux4to1 ( a, b, c, d, s0, s1, out);

input wire a, b, c, d;
input wire s0, s1;
output reg out;

always @ (a or b or c or d or s0, s1)
begin

case (s0 | s1)
'2b00 : out <= a;
'2b01 : out <= b;
'2b10 : out <= c;
'2b11 : out <= d;
endcase

end

endmodule
```

FULL SUBTRACTOR

```
module fullsub_ody_1201462 #(parameter n=3) (x,y,c_in,c_out,res);

; input signed [n:0] x

; input signed [n:0] y

;input c_in

;output reg c_out

;output reg signed  [n:0] res

(*) @ always

begin

;res = x^y^c_in

;c_out = (~x&y)|(~(x^y) & c_in)

end

endmodule
```