Computer Science Department

Digital System

COMP232

Student's name, ID:
Mohammad AbuThaher1202833

Instructor:
Dr. Anjad Badran

Section: 5

Date: 7/02/2023

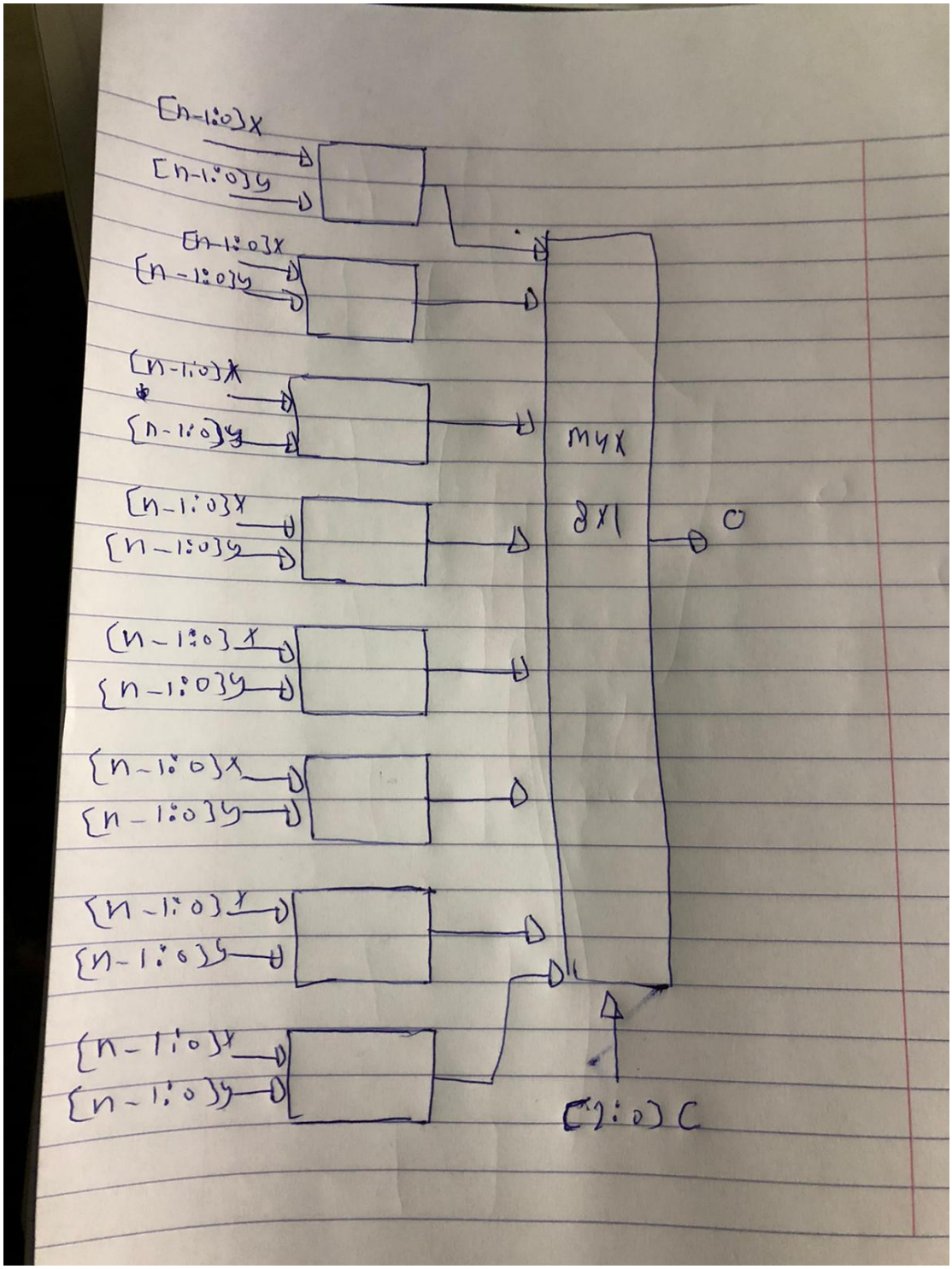
First semester 2022/2023

A)

The size of the output (O) in bits so the overflow can never occur is n + 1 bits. To prevent overflow, the output must be able to represent the largest possible result of

any of the operations, which can be represented as the largest possible n-bit signed number plus the largest possible n-bit signed number, which requires n + 1 bits.

B)


[n-1:0]x
[n-1:0]y
[n-1:0]x
[n-1:0]y
[n-1:0]x
[n-1:0]y
[n-1:0]x
[n-1:0]y
[n-1:0]x
[n-1:0]y
[n-1:0]x
[n-1:0]y
[n-1:0]x
[n-1:0]y
mux
8x1
O
[2:0] C

C)

1)

```
module optione0(X, Y, O);
	parameter n = 4;
	input [n - 1 : 0] X, Y;
	output reg [n + 1 : 0] O;

	always @(X or Y)
		begin
			O <= (X + Y) / 2;
		end

endmodule
```

2)

```
module optione1(X, Y, O);
	parameter n = 4;
	input [n - 1 : 0] X, Y;
	output reg [n + 1 : 0] O;

	always @(X or Y)
		begin
			O <= 2 * (X + Y);
		end

endmodule
```

3)

```
module optione2(X, Y, O);
	parameter n = 4;
	input [n - 1 : 0] X, Y;
	output reg [n + 1 : 0] O;

	always @(X or Y)
		begin
			O <= (X / 2) + Y;
		end

endmodule

module optione3(X, Y, O);
	parameter n = 4;
	input [n - 1 : 0] X, Y;
	output reg [n + 1 : 0] O;

	always @(X or Y)
		begin
			O <= X - (Y / 2);
		end

endmodule
```

5)

```
module MUX8X1(c0, c1, c2, c3, c4, c5, c6, c7, sel, out, zero);
        parameter n = 4;
        input [n + 1 : 0] c0, c1, c2, c3, c4, c5, c6, c7;
        input [2 : 0] sel;
        output reg signed [n + 1 : 0] out;
        output reg zero;
        always @* begin
                        out = 0;
                        zero = 1;
                case (sel)
                        3'b000: out = c0;
                        3'b001: out = c1;
                        3'b010: out = c2;
                        3'b011: out = c3;
```

```
                3'b100: out = c4;
                3'b101: out = c5;
                3'b110: out = c6;
                3'b111: out = c7;
                default: ;
            endcase
                if(out == 0)
                        zero = 1;
                else
                        zero = 0;
        end

endmodule
```

d)

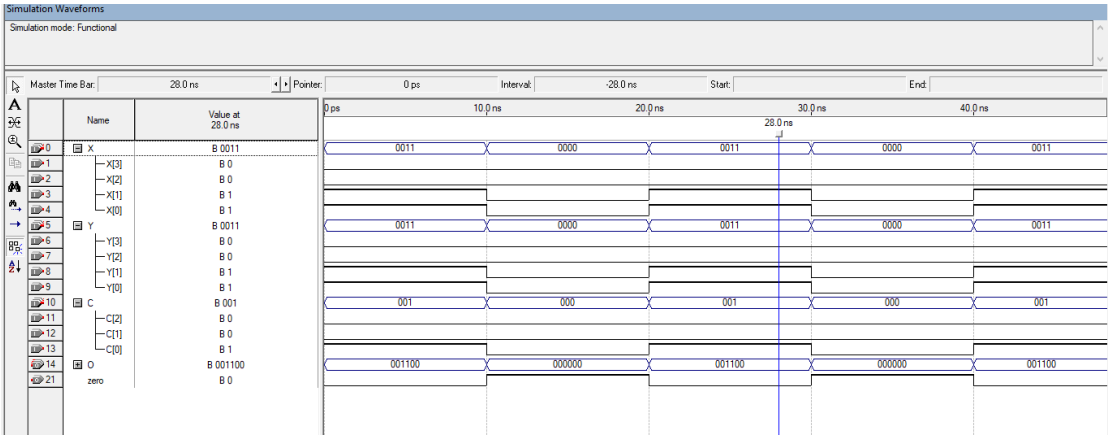
```
module ALUStruct(X, Y, C, O, zero);
    parameter n = 4;
    input [n - 1 : 0] X, Y;
    input [2 : 0] C;
    output  signed [n + 1 : 0] O;
    output  zero;
    wire [n - 1 : 0] optione0_Ans, optione1_Ans, optione2_Ans, optione3_Ans,
optione4_Ans, optione5_Ans, optione6_Ans, optione7_Ans;
    optione0 caout0(X, Y, optione0_Ans);
    defparam caout0.n = n;
    optione1 caout1(X, Y, optione1_Ans);
    defparam caout1.n = n;
    optione2 caout2(X, Y, optione2_Ans);
```

```
        defparam caout2.n = n;

        optione3 caout3(X, Y, optione3_Ans);

        defparam caout3.n = n;

        nand caout4(optione4_Ans, X, Y);

        not caout5(optione5_Ans, X);

        nor caout6(optione6_Ans, X, Y);

        xor caout7(optione7_Ans, X, Y);

        MUX8X1 mux2(optione0_Ans, optione1_Ans, optione2_Ans, optione3_Ans,
optione4_Ans, optione5_Ans, optione6_Ans, optione7_Ans, C, O, zero);

        defparam mux2.n = n;


endmodule
```

e) ID = 1202833

X1 = 3	Y1 = 3	C1 = 1	2*(X+Y)
X2 = 2	Y2 = 2	C2 = 2	(X/2) +Y
X3 = 5	Y3 = 5	C3 = 2	(X/2) +Y

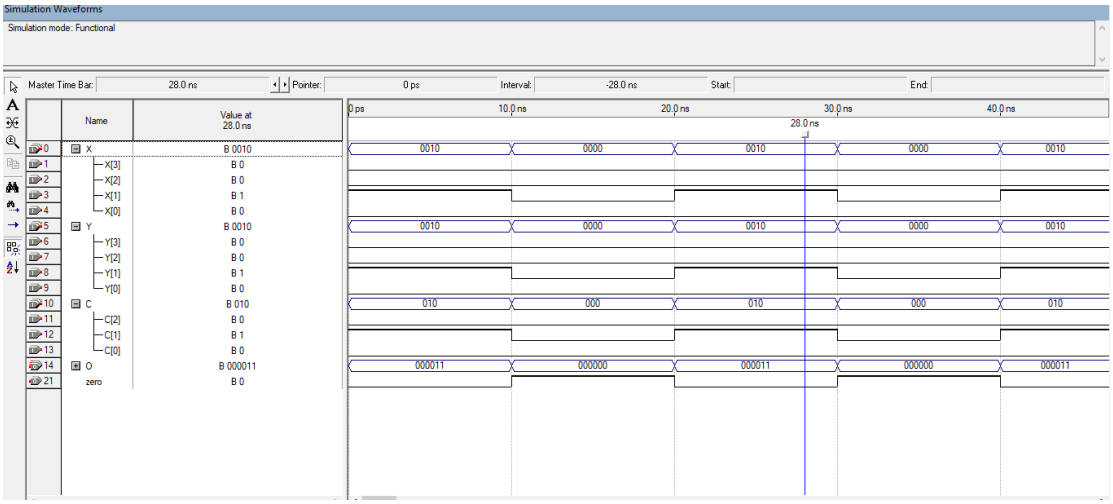
In the test case(1) Number one the waveform will be like that:



The selection in this case is 1 and the input x = 3 and y =3 from the ALU module the operation will be

2*(X+Y) in this case we add the x and y and multiplied it by 2 .

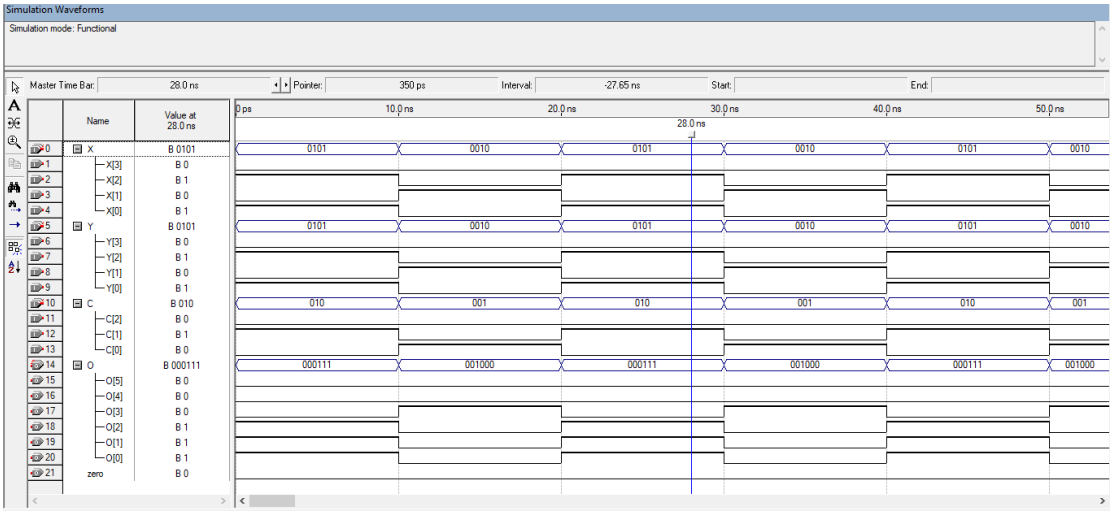
Case (2) Number one the waveform will be like that:



The selection in this case is 2 and the input x = 2 and y =2 from the ALU module the operation will be

(X/2) +Y in this case we divided the x by 2 and added the result to the y.

Case (3) Number one the waveform will be like that:



The selection in this case is 2 and the input x = 5 and y =5 from the ALU module the operation will be

The selection in this case is 1 and the input x = 3 and y =3 from the ALU module the operation will be

(X/2) +Y in this case we divided the x by 2 and added the result to the y.

f)

```
module ALU (X, Y, C, O, zero);
        parameter n = 4;
        input [n - 1 : 0] X, Y;
        input [2 : 0] C; // opcode
        output reg signed [n + 1 : 0] O; // Output can be maximum n + 2 bits which might happen when opcode = 001, (x + y) can be (n + 1) bits and 2 * (x + y) can be n + 2 bits
        output reg zero; // zero extension
always @* begin
  O = 0;
  zero = 1;
  case (C)
    3'b000: O = (X + Y) / 2;
    3'b001: O = 2 * (X + Y);
    3'b010: O = (X / 2) + Y;
    3'b011: O = X - (Y / 2);
    3'b100: O = ~(X & Y);
    3'b101: O = ~X;
    3'b110: O = ~(X | Y);
    3'b111: O = (X ^ Y);
    default: ;
  endcase
  if (O == 0)
    zero = 1;
  else
    zero = 0;
```

end

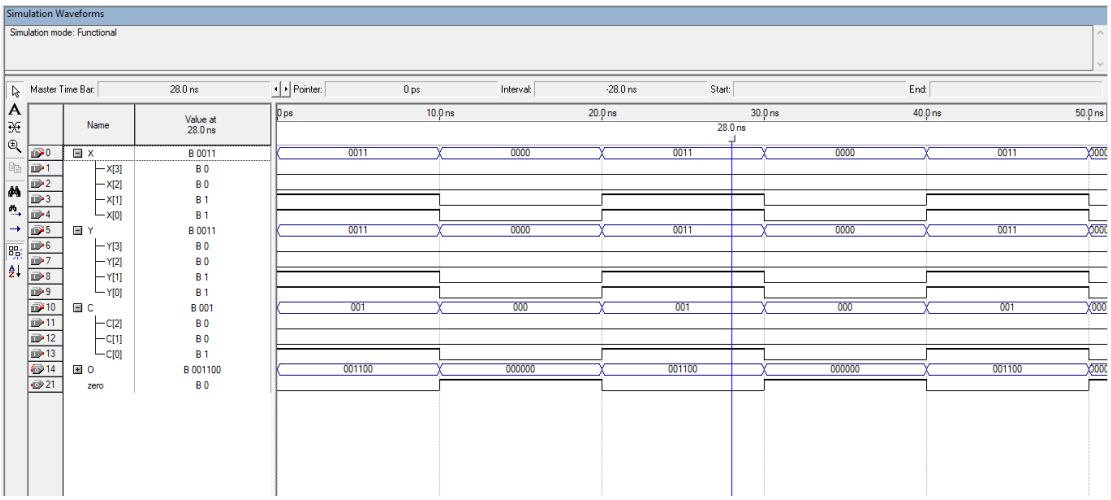
endmodule

g)

ID = 1202833

X1 = 3	Y1 = 3	C1 = 1	2*(X+Y)
X2 = 2	Y2 = 2	C2 = 2	(X/2) +Y
X3 = 5	Y3 = 5	C3 = 2	(X/2) +Y

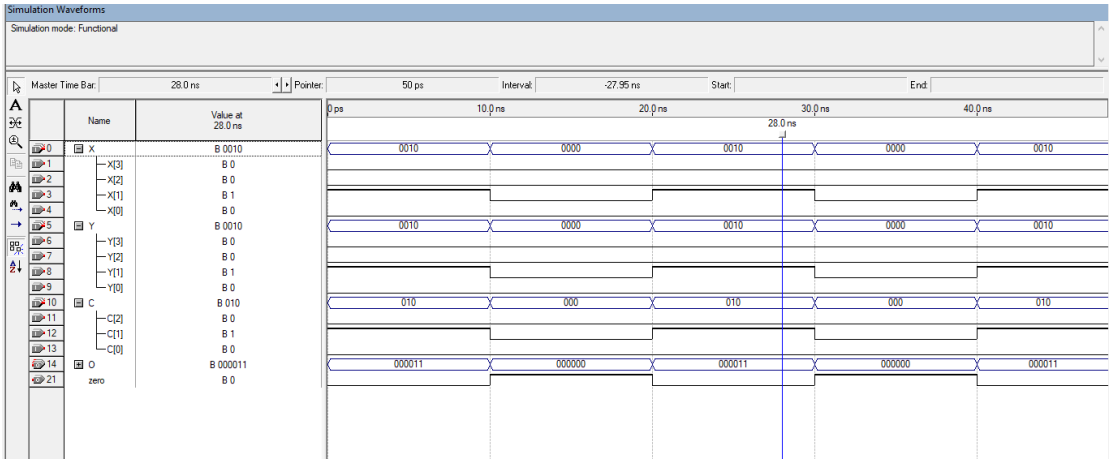
In the test case(1) Number one the waveform will be like that:



The selection in this case is 1 and the input x = 3 and y =3 from the ALU module the operation will be

2*(X+Y) in this case we add the x and y and multiplied it by 2 .

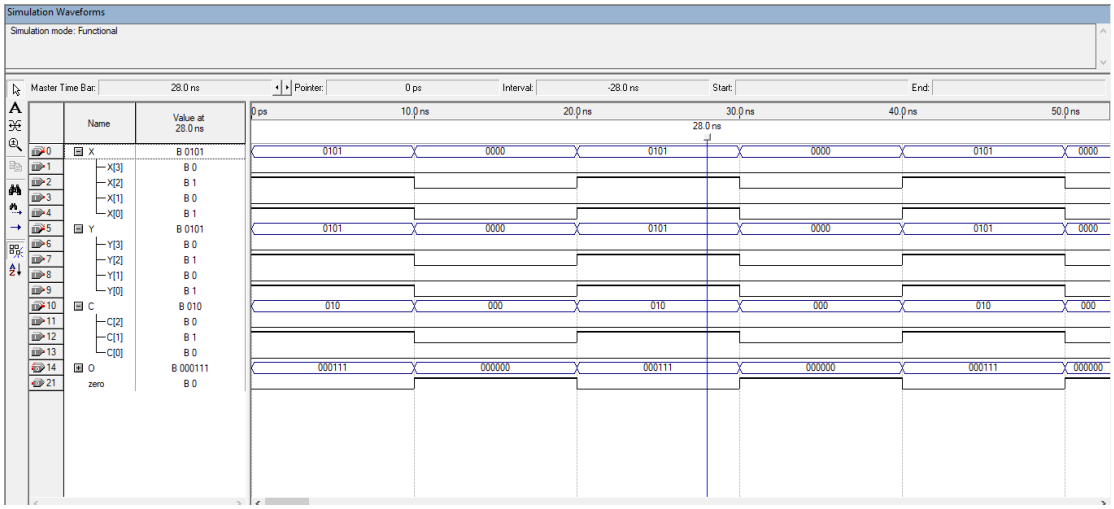
Case (2) Number one the waveform will be like that:



The selection in this case is 2 and the input x = 2 and y =2 from the ALU module the operation will be

(X/2) +Y in this case we divided the x by 2 and added the result to the y.

Case (3) Number one the waveform will be like that:



The selection in this case is 2 and the input x = 5 and y =5 from the ALU module the operation will be

(X/2) +Y in this case we divided the x by 2 and added the result to the y.