

Digital Systems ENCS2340

Verilog HDL Project

Dr. Ismail Khater

Section (2)

Yaqen Nouman Hamouda (1211168)

Contents:

- >> Definition for N-bit ALU
- >> Description for ALU we want to design
- >> A) Specify the size of the output (O)
- >> B) ALU components & implementation
 - >> The components use to design this ALU
 - >> Implementation for each component
 - >> Implementation for whole system
- >> C) Behavioral Verilog modules
- >> D) Structural Verilog module for ALU
- >> E) Waveform of the Structural ALU
- >> F) Behavioral Verilog module for ALU
- >> G) Waveform of the Behavioral ALU



Definition for N-bit ALU:

(Arithmetic Logic Unit) ALU is a digital function that implements the microoperation on the information stored in registers, ALU is a fundamental building block of the many varieties of computing circuits, including the central processing unit (CPU) of computers and graphics processing units (GPUs). one CPU or GPU may contain multiple ALUs.

Description for ALU we want to design:

ALU Function Code (C)	ALU Output (O)	ALU Symbol
000	(X + Y)/2	X[n-1:0] Y[n-1:0]
001	2*(X + Y)	
010	(X/2)+Y	<u>√</u> n <u>√</u> n
011	X-(Y /2)	C[2:0]
100	X NAND Y	ALU
101	NOT(X)	3
110	X NOR Y	¥
111	X XOR Y	0

In this project, we need to design ALU that execute the above operations. Such that:

1. X and Y are the inputs of the unit and they are N-bit signed numbers represented by 2's complement.

2. C is a 3-bit unsigned number and used to select the operation of the unit (i.e. arithmetic or logical operation).

3. O is the signed ALU output and represented in 2's complement and The size of the O is shown below.

A) Specify the size of the output (O):

The size of the output (O) in bits should be two bit larger than the size of the inputs (X and Y) in order for overflow to never occur.

Since X and Y are N-bit signed numbers represented in 2's complement, the size of O should be N+2 bits.

The first extra bit because the 2's complement representation allows for the detection of overflow by checking the sign bit of the result and having an extra bit in the output allows for the representation of the extra bit in the result that would occur in the case of overflow.

The second extra bit because the second operation in this ALU $(2^{*}(X+Y))$, (X+Y) are represented in N+1 bits and we need extra bit to Multiplication by two then the size of O should be N+2 bits.

Example:

Suppose N=3;

Then the maximum value for X and Y equal 3 (011);

Execute the second operation (001) (2*(X+Y));

The value for (X+Y) equal 6 and need 4 bits to represented it in 2's complement (0110) ;

The value for (2*(X+Y)) equal 12 and need 5 bits to represented it in 2's complement (01100) ;

Then we need 5 bits to represented the output without occur overflow.

B) ALU components & implementation:

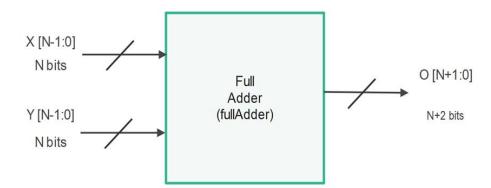
>> ALU implementation using all the following:

- 1. A adder is used for addition and subtraction operations.
- 2. A right shift register for N+2 is used for division by 2.
- 3. A right shift register N is used for division by 2
- 4. A left shift register is used for multiplication by2.
- 5. A NOT module is used for NOT operation.

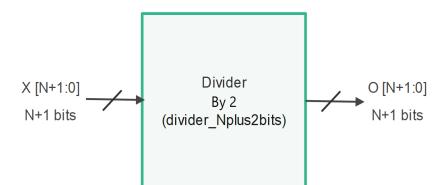
- 6. A NAND module is used for NAND operation.
- 7. A NOR module is used for NOR operation.
- 8. A XOR module is used for XOR operation.
- 9. A multiplexer(8to1) is used to select the output of the operation.

>> I will use this component to design the ALU, and the implementation for each component will be as shown below:

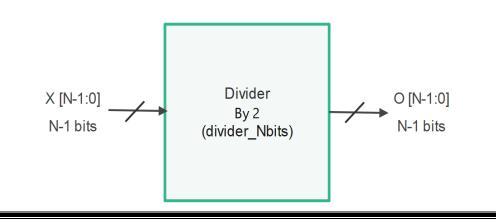
1. implementation for Adder:



2. implementation for divider N+2 bits

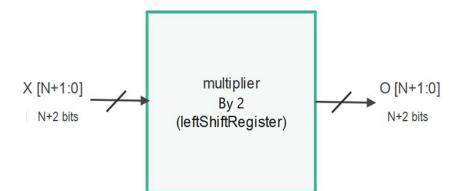


3. implementation for divider N bits

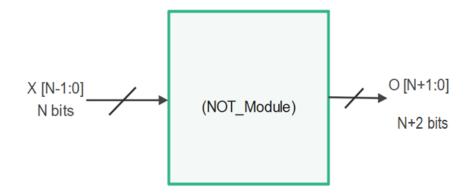


Two modules were created to divide by two the first module has input with size N and the second module has input with size N+2, The reason for the creation of these two because it will become easier to put in the extra bits (sign- or zero-extension).

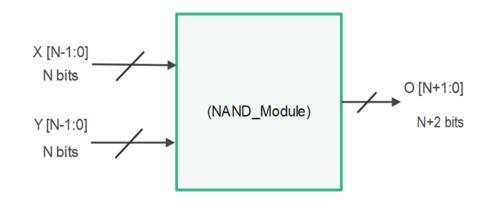
4. implementation for multiplier N+2 bits



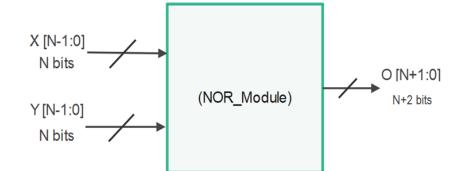
5. implementation for NOT module:



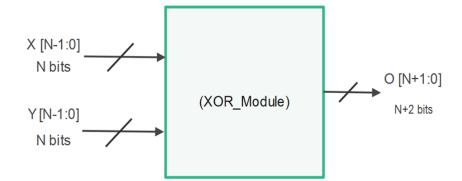
6. implementation for NAND module:



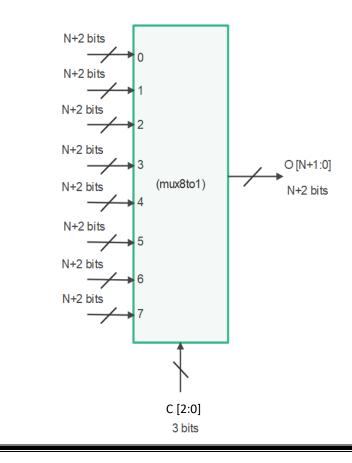
7. implementation for NOR module:



8. implementation for XOR module:



9. implementation for NOR module:



>> Implementation for whole system: N+2 bits N bits (Divider By 2) N+2 bits (Full Adder) 0 N bits N+2 bits nultiplier By 2 N bits N+2 bits (Full Adder) (Divider By 2) 2 N+2 bits N bits N+2 bits N bits X [N-1:0] O [N+1:0] 3 N bits (Full Adder) (-)Nbits N+2 bits 4 (Divider By/2) (mux8to1) N+2 bits N bits N+2 bits Y[N-1:0] Nbits N bits 4 N bits NAND Modul N+2 bits N+2 bits N bits ▶5 N bits (NOT Module) N+2 bits N+2 bits 6 N bits N+2 bits (NOR Module) N+2 bits N hits N bits (XOR Module) N+2 bits N bits

C [2:0] 3 bits

C) Behavioral Verilog modules

Behavioral Verilog code for each component:

1. fullAdder_1211168

```
module fullAdder_1211168 #(parameter N = 4) (X,Y,Out);
 1
 2
       // Define input and output
 3
       input [N-1:0] X, Y;
 4
       output reg [N+1:0] Out;
 5
       reg [N-1:0] s, c;
 6
 7
       always @(*) begin
   // calculute the result
8
9
         s = X + Y;
10
         // calculate the N+1 bit,
11
         //by check the value of caary and the value of overFlow
12
         c = (X[N-1] \& Y[N-1]) | (X[N-1] \& s[N-1]) | (Y[N-1] \& s[N-1]);
         // the final resalt with number bits N+1
13
14
         Out = \{c, s\};
15
         // make sign-extension for the last bit
16
         Out ={Out[N],Out[N:0]};
17
       end
18
     endmodule
```

2. divider_Nplus2bits_1211168

```
1
  module divider Nplus2bits 1211168 #(parameter N = 4)
      (input signed [N+1:0] X, output reg signed[N+1:0] O);
2
3
      // Define input and output as signed
4
     //because the output will represented in 2's complement
5
  always @* begin
      // calculate the result
6
7
        0 = X / 2;
8
      end
9
    endmodule
```

3. divider_Nplus2bits_1211168

```
1
   module divider Nplus2bits 1211168 #(parameter N = 4)
2
      (input signed [N+1:0] X, output reg signed[N+1:0] O);
3
      // Define input and output as signed
4
      //because the output will represented in 2's complement
5
  always @* begin
6
      // calculate the result
7
        0 = X / 2;
8
      end
9
    endmodule
```

4. multiplier_1211168

```
1
  module multiplier 1211168 #(parameter N = 4)
2
      (input signed [N+1:0] X, output reg signed [N+1:0] O);
3
      // Define input and output as signed
4
      // because the output will represented in 2's complement
5
      always @* begin
  6
      // calculate the result
7
        0 \le x * 2;
8
      end
9
    endmodule
```

5. NAND_Module_1211168

```
module NAND Module 1211168 #(parameter N = 4)
1
2
         (input [N-1:0] X, input [N-1:0] Y, output reg [N+1:0] O);
3
      // Define input and output
4
  always @* begin
5
      // calculate the result
6
        O = \sim (X \& Y);
7
      end
  endmodule
8
```

6. XOR_Module_1211168

```
module XOR Module 1211168 #(parameter N = 4)
1
        (input [N-1:0] X, input [N-1:0] Y, output reg [N+1:0] O);
2
      // Define input and output
3
      always @* begin
4
  5
       // calculate the result
6
        O = (X ^ Y);
7
      end
8
    endmodule
```

7. NOR_Module_1211168

```
module NOR Module 1211168 #(parameter N = 4)
1
2
             (input [N-1:0] X, input [N-1:0] Y, output reg [N+1:0] O);
3
      // Define input and output
  always @* begin
4
5
       // calculate the result
6
        O = ~(X | Y);
7
      end
8
    endmodule
```

8. NOT_Module_1211168

```
module NOT Module 1211168 #(parameter N = 4)
1
2
        (input [N-1:0] X, output reg[N+1:0] O);
3
      // Define input and output
      always @* begin
4
   5
       // calculate the result
6
        0 = (X);
7
      end
8
    endmodule
```

9. mux8to1_1211168

```
1 ■module mux8to1 1211168 #(parameter N = 4) ( input [N+1:0]
 2
                     result0, result1, result2, result3, result4, result5, result6, result7,
 3
                      input [2:0] sel, output reg [N+1:0] O);
 4
          // Define input and output for the mux
 5
         always @* begin
   6
          // by the selction concate between the input and output
 7
   case(sel)
 8
                  3'b000: 0 = result0;
 9
                 3'b001: 0 = result1;
10
                 3'b010: 0 = result2;
                 3'b011: 0 = result3;
11
                 3'b100: 0 = result4;
12
                 3'b101: 0 = result5;
13
14
                 3'b110: 0 = result6;
15
                 3'b111: 0 = result7;
16
             endcase
17
         end
18
     endmodule
```

D) structural Verilog model for ALU

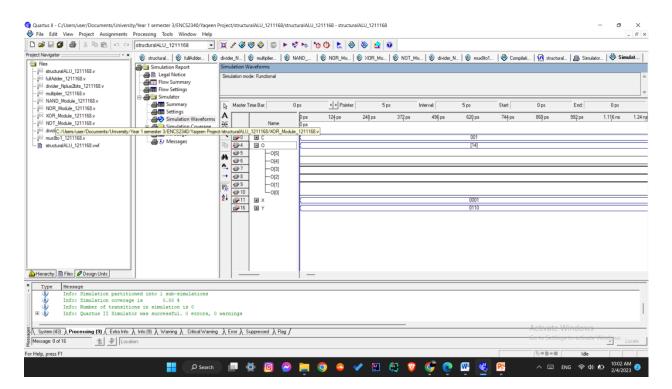
```
module structuralALU 1211168 #(parameter N = 4) (
 1
       input [N-1:0] X,
 2
 3
       input [N-1:0] Y,
 4
       input [2:0] C,
 5
       output [N+1:0] 0 );
 6
       // Define input and output for this ALU
 7
 8
       // Define wires to save the result for each operation
 9
       wire [N+1:0] result0, result1, result2, result3, result4, result5, result6, result7;
10
11
       // Define wires to save temp res
12
       wire [N+1:0] op1;
13
       wire [N-1:0] op2, op3;
14
15
       // operation (000)
16
       fullAdder_1211168 #(N) FA_0 (.X(X), .Y(Y), .Out(op1));
17
       divider_Nplus2bits_1211168 #(N) RS_0 (.X(op1), .O(result0));
18
19
       // operation (001)
20
       multiplier_1211168 #(N) M_0 (.X(op1), .O(result1));
21
22
       // operation (010)
       divider_Nbits_1211168 #(N) RS_1 (.X(X), .O(op2));
23
       fullAdder_1211168 #(N) FA_1 (.X(op2), .Y(Y), .Out(result2));
24
25
26
        // operation (011)
27
        divider Nbits 1211168 #(N) RS 2 (.X(Y), .O(op3));
28
        fullAdder_1211168 #(N) FA_2 (.X(X), .Y(-op3), .Out(result3));
29
30
        // operation (100)
31
       NAND_Module_1211168 #(N) NAND_0 (.X(X), .Y(Y), .O(result4));
32
33
       // operation (101)
34
       NOT_Module_1211168 #(N) NOT_0 (.X(X), .O(result5));
35
36
       // operation (110)
37
       NOR_Module_1211168 #(N) NOR_0 (.X(X), .Y(Y), .O(result6));
38
39
       // operation (111)
       XOR_Module_1211168 #(N) XOR_0 (.X(X), .Y(Y), .O(result7));
40
41
42
       // send all result with selection to Mux8to1 to define the final Output
43
   mux8to1_1211168 #(N) MUX_0 (.result0(result0),.result1(result1),
44
          .result2(result2), .result3(result3),.result4(result4),
45
          .result5(result5),.result6(result6),.result7(result7),
46
          .sel(C),.O(O)
47
        );
48
   endmodule
```

E) Waveform of the Structural ALU

My number 1211168 ; The Form: $1 C_2 Y_2 X_2 C_1 Y_1 X_1$ Then $C_2 = 2 Y_2 = 1 X_2 = 1$ $C_1 = 1 Y_1 = 6 X_1 = 1$

Test	Х	Y	С	Operation
1	$X_1 = 1$	$Y_1 = 6$	$C_1 = 1$	2*(1+6)
2	$X_2 = 1$	$Y_2 = 1$	$C_2 = 2$	(1/2)+(1)
3	$X_3 = -1$	$Y_3 = -6$	$C_3 = 2$	(-1/2)+(-6)

Waveform for test 1:



Zoom:

4	Master T	ime Bar:	0 ps	Pointer:	141	l ps	Interval:	141 ps	Start:	0 ps	End:	0 ps
1			0 ps	124 ps	248 ps	372 ps	496 ps	620 ps	744 ps	868 ps	992 ps	1.116 ns
£		Name	0 ps					r				
	i ₽0		— i—					001				
5	6 24	0						[14]				
	-@>5	-O[5]										
,		—O[4]										
		-0[3]							_			
_	- <u>-</u>	—O[2] —O[1]										
£	10											
ţ	P 11	Ξ×						0001				
	16	ΞY						0110	_			
								,				

The result for the first test equal (001110)

2*(1+6) = 14

Waveform for test 2:

File Edit View Project Assignments		×	/ 🧭 🦁 🤇) 🗊 🕨	1	▶	ነው 🕐 📘 🖣	ی 🐌 🛓	0								- 8
Pies > * * Pies Internative ULU_1211168 v Pies INAND, Modale, 1211168 v Pies Internative ULU_1211168 v		divide Simu Simu Simu Simu Simu Simu Simu		ubiplier 1 nrms nnctional ar.	NA 0 p	ND		� xor_n		Inte	S divider_N .	😵 multo1 14 ps 620 ps 010 (1) 0001 0001	Stat:	0 ps 865 ps	End	0 ps 1.116 ms	1.24
 Info: Simulation coverag Info: Number of transiti 	ons in simulation is 0 or was successful. 0 errors, 0 <u>\lambda</u> Info(3) <u>\lambda</u> Warning <u>\lambda</u> Critical Warning			ssed) Flag	7		-						;	Activate W So to Settings	findows to activate W		Locate

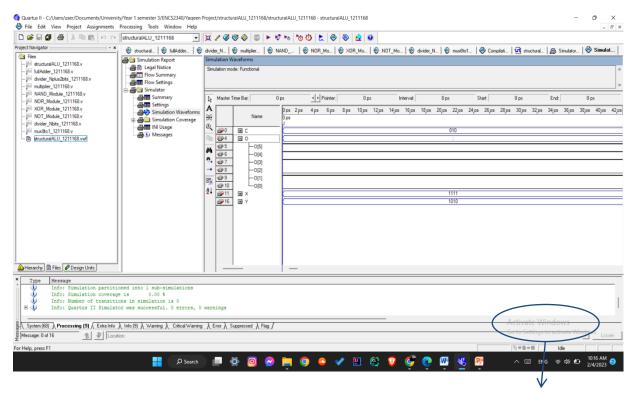
Zoom:

ß	Master T	'ime Bar	:	0 ps	Image: Image	14	ps	Interval:	14 ps	Start:	0 ps	End:	0 ps
A				0 ps	124 ps	248 ps	372 ps	496 ps	620 ps	744 ps	868 ps	992 ps	1.116 ns
Æ			Name	0 ps								· · · ·	
	i ₽0	⊡ C	:	-6					010				
B	@ 4	Ξ 0							[1]				
å	₫5		-O[5]										
ĝ	@ 6		-O[4]										
•	@7		-O[3]										
	@ 8		-O[2]										
Ŗ	@ 9		-0[1]										
	il 🚳		L_O[0]										
ţ	11	Ξ×							0001				
	16								0001				
		I							1				
								✓					

The result for the first test equal (000001)

(1/2)+(1) = 1

Waveform for test 3:



I Know I need to active windows

Zoom:

Q.	Master T	ime Bar:	0 ps	• • Pointer		0 p:	s	Ir	nterval:		0 ps	:	S	tart		0 ps		End:		0 p	s
A			0 ps 2 ps	4 ps 6 ps	8 ps	10 _. ps	12.ps	14 ps	16 ps	18 ps	20 ps	22 _{,ps}	24.ps	26 ps	28 ps	30,ps	32 ps	34 ps	36 ps	38 _. ps	40.ps
Æ		Name	0 ps				•														
±,	₽ 0		- F									010									
Þ	<u>−</u> 4											:									
<i>i</i> ż	@ 5	—O[5]																			
	 ●6 	—O[4]																			
	@7	-0[3]																			
	@8 @9	-0[2]											_								
咒	@ 10	-O[1] -O[0]																			
₽↓	₽ 11											1111	_								
	■ 16	±Υ										1010									
		I																			
										J	,										

The result for the first test equal (111010)

(-1/2)+(-6) = -6

F) Behavioral Verilog module for ALU

In the previous part we designed Structural ALU with his component but in this part we designed a behavioral ALU as we shown.

The behavioral code in Verilog for this ALU:

```
module behavioralALU 1211168 #(parameter N = 4)
 1
 2
    (input signed [N-1:0] X, Y, input signed [2:0] C, output reg signed[N+1:0] O);
 3
    // define input & output for ALU
 4
 5
    // define a zero-extension for logic operation
 6
    wire ex = 2'b00;
 7
 8 Ealways @* begin
9
   // check the selection and calculate result
10 
case (C)
         3'b000: 0 = (X + Y)/2;
11
        3'b001: O = 2*(X + Y);
12
13
         3'b010: 0 = (X/2) + Y;
14
         3'b011: O = X - (Y/2);
         3'b100: O = {ex,~(X & Y)};
15
16
         3'b101: O = {ex,~X};
         3'b110: O = {ex,~(X | Y)};
17
18
         3'b111: O = {ex,X ^ Y};
19
      endcase
20
   end
21
    endmodule
22
```

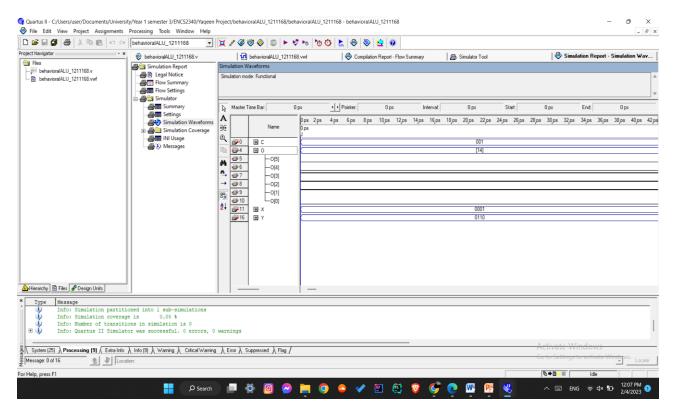
G) Waveform of the Behavioral ALU

My number 1211168 ; The Form: $1 C_2 Y_2 X_2 C_1 Y_1 X_1$

Then $C_2 = 2 \ Y_2 = 1 \ X_2 = 1$ $C_1 = 1 \ Y_1 = 6 \ X_1 = 1$

Test	Х	Y	С	Operation
1	$X_1 = 1$	$Y_1 = 6$	$C_1 = 1$	2*(1+6)
2	$X_2 = 1$	$Y_2 = 1$	$C_2 = 2$	(1/2)+(1)
3	$X_3 = -1$	$Y_3 = -6$	$C_3 = 2$	(-1/2)+(-6)

Waveform for test 1:



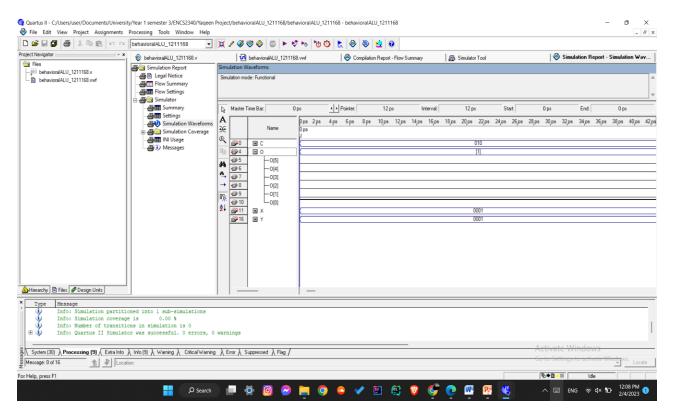
Zoom:

	Master T	ime Bar	:	0 ps		• •	Pointer:		0 p:	\$	In	terval:		0 ps		St	art:		0 ps		End:	
-			Name		ps 2ps ps	4 ps	6 ps	8 ps	10 ps	12.ps	14 ps	16 _. ps	18 ps	20 ps	22.ps	24 ps	26 ps	28 ps	30 _. ps	32 _, ps	34 ps	36 _, p
	i	⊡ C	:	Ē											001							
à	@≥4)	K											[14]							
			-O[5]											L								
	@6		-O[4]	– L																		
	@7		-O[3]	- F																		
	• 8		-O[2]	- F																		
7	@ 9		-0[1]	- 1																		
1	🐵 10		L_O[0]	_ L																		
	11	Ξ×	((0001							
	16	±Υ	,											(0110							
														- 7								
														K								

The result for the first test equal (001110)

2*(1+6) = 14

Waveform for test 2:



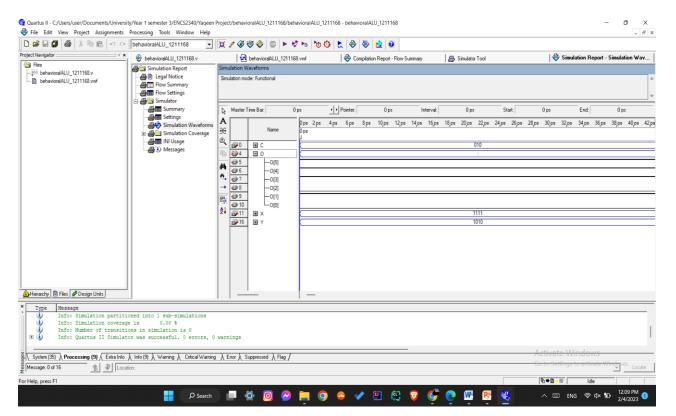
Zoom:

ß	Master T	ime Bar:	0 ps	• •	Pointer:		12 p	8	In	iterval:		12 p:	8	St	art:		0 ps		End:	
A ⊛		Name	0 ps 2 ps 0 ps	4 ps	6 ps	8 ps	10 ps	12 ps	14 ps	16 _, ps	18 _, ps	20 ps	22.ps	24 ps	26 ps	28 ps	30 _, ps	32 _, ps	34 ps	36 ps
€ ⊡	P 0	EC										_	010							
40 #4		□ 0 -0[5]											[1]							
899 M	 	—O[4] —O[3]										-		_						
-	@ 8	-O[2]																		
暻	 	-O[1] -O[0]										+		⊨						
₽↓	₽11 ₽16	± Χ € Υ											0001							
												/								
												K								

The result for the first test equal (000001)

(1/2)+(1) = 1

Waveform for test 3:



Zoom:

ß	Master T	ime Bar:	0 ps		• • F	Pointer:		0 p:	s	Ir	nterval:		0 ps		St	art:		0 ps		End:		
4 €		Name	0 ps 0 ps	2 ps	4 ps	6 ps	8 ps	10 _. ps	12 _{ps}	14 ps	16 ps	18 _. ps	20 ps	22.ps	24 ps	26 _. ps	28 ps	30 _. ps	32 _, ps	34 ps	36 _, ps	
L	0	± C	-¢											010								-
à	@ ¥	0												:								_
ŝ,	 	—O[5] —O[4]	F										\vdash									_
↓ +		—0[3] —0[2]																				
×	 																					-
ŧ	P 11	€ ×												111								=
	₩ 16	≇ Υ												010								_

The result for the first test equal (111010)

(-1/2)+(-6) = -6