

## 1.2 PRELIMINARY WORK

### 1.2.2 GATES

#### 1. i.

→ NOT:	<b>A</b>	<b>F</b>	Boolean exp: $F=A'$	and	<b>B</b>	<b>F</b>	Boolean exp: $F=B'$
	0	1			1	0	
	1	0			0	1	

→ AND:	<b>A</b>	<b>B</b>	<b>F</b>	Boolean exp: $F=AB$
	0	0	0	
	0	1	0	
	1	0	0	
	1	1	1	

→ OR:	<b>A</b>	<b>B</b>	<b>F</b>	Boolean exp: $F=A+B$
	0	0	0	
	0	1	1	
	1	0	1	
	1	1	1	

→ NAND:	<b>A</b>	<b>B</b>	<b>F</b>	Boolean exp: $F=(AB)'=A'+B'$
	0	0	1	
	0	1	1	
	1	0	1	
	1	1	0	

→ NOR:	<b>A</b>	<b>B</b>	<b>F</b>	Boolean exp: $F=(A+B)'=A'B'$
	0	0	1	
	0	1	0	
	1	0	0	
	1	1	0	


→ XOR:	<b>A</b>	<b>B</b>	<b>F</b>	Boolean exp: $F=A'B+AB'$
	0	0	0	
	0	1	1	
	1	0	1	
	1	1	0	


→ XNOR:	<b>A</b>	<b>B</b>	<b>F</b>	Boolean exp: $F=A'B+AB$
	0	0	1	
	0	1	0	
	1	0	0	
	1	1	1	

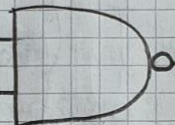
ii.


ii. - NOT  $\Rightarrow$    $A \rightarrow A'$

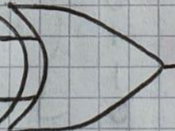
$B \rightarrow B'$


- AND  $\Rightarrow$    $A \cdot B$

- OR  $\Rightarrow$    $A + B$

- NAND  $\Rightarrow$    $(A \cdot B)'$

- NOR  $\Rightarrow$    $(A + B)'$

- XOR  $\Rightarrow$    $A \oplus B = A'B + AB'$

- XNOR  $\Rightarrow$    $A \odot B = A'B' + AB$

## 2.i.

a) **NOT:**      module lab1gates (A, B);      (For A to be input)

input A;

output B;

not G1(B, A);

endmodule

module lab1gates (B, A);    (For B to be input)

input B;

output A;

not G2(A, B);

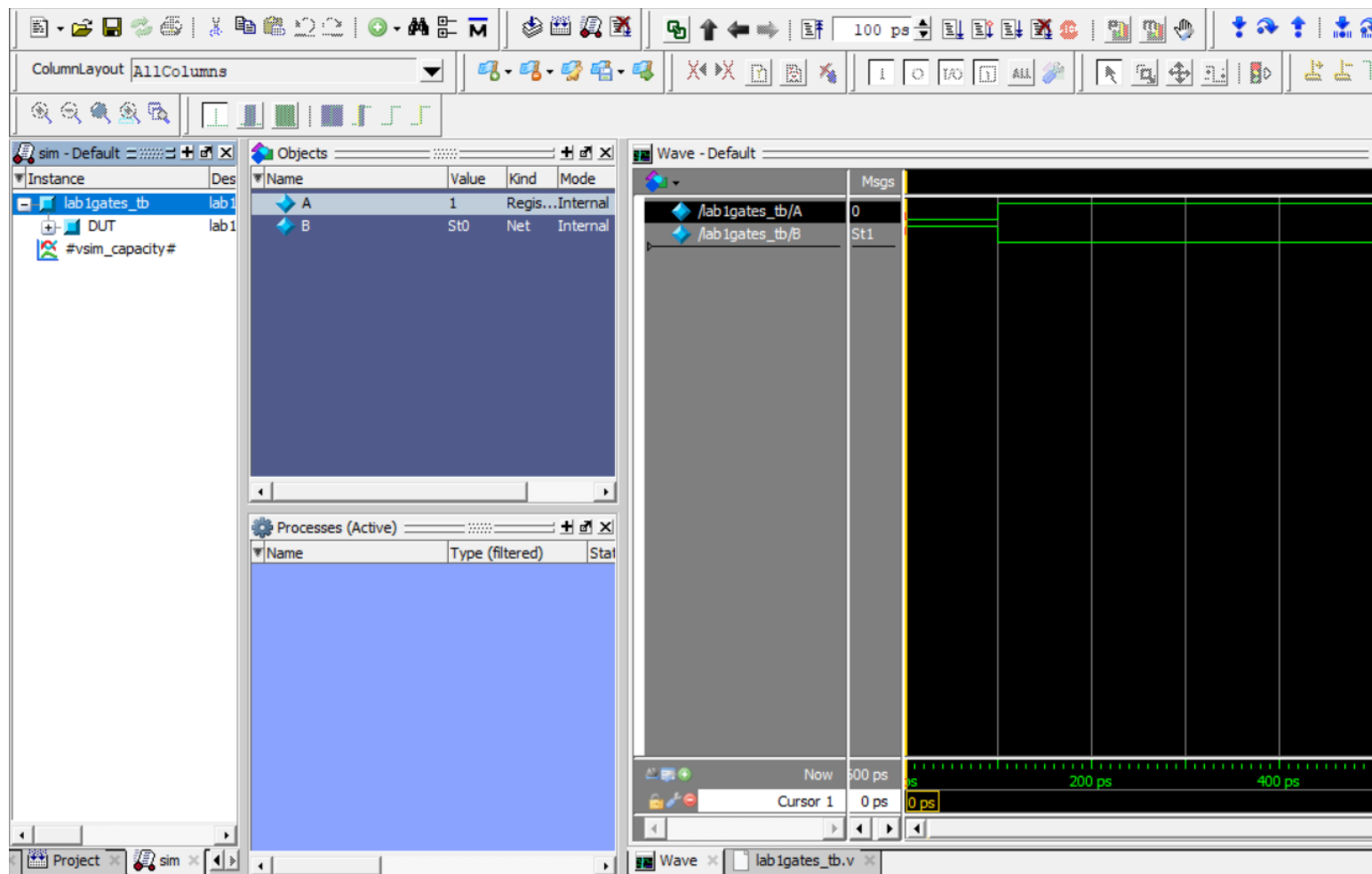
endmodule

**For A as an input**

Ln#	
1	module lablgates_tb ();
2	
3	reg A;
4	wire B;
5	
6	lablgates DUT (A, B);
7	
8	initial begin
9	A=0; #100;
10	A=1; #100;
11	
12	
13	end
14	endmodule
15	

**For B as an input**

Ln#	
1	module lablgates_tb ();
2	
3	reg B;
4	wire A;
5	
6	lablgates DUT (A, B);
7	
8	initial begin
9	B=0; #100;
10	B=1; #100;
11	
12	
13	end
14	endmodule
15	



Modelsim Simulation for the NOT gate (Same for both A and B inputs)

**b) AND:**      module lab1gates (A, B, C);

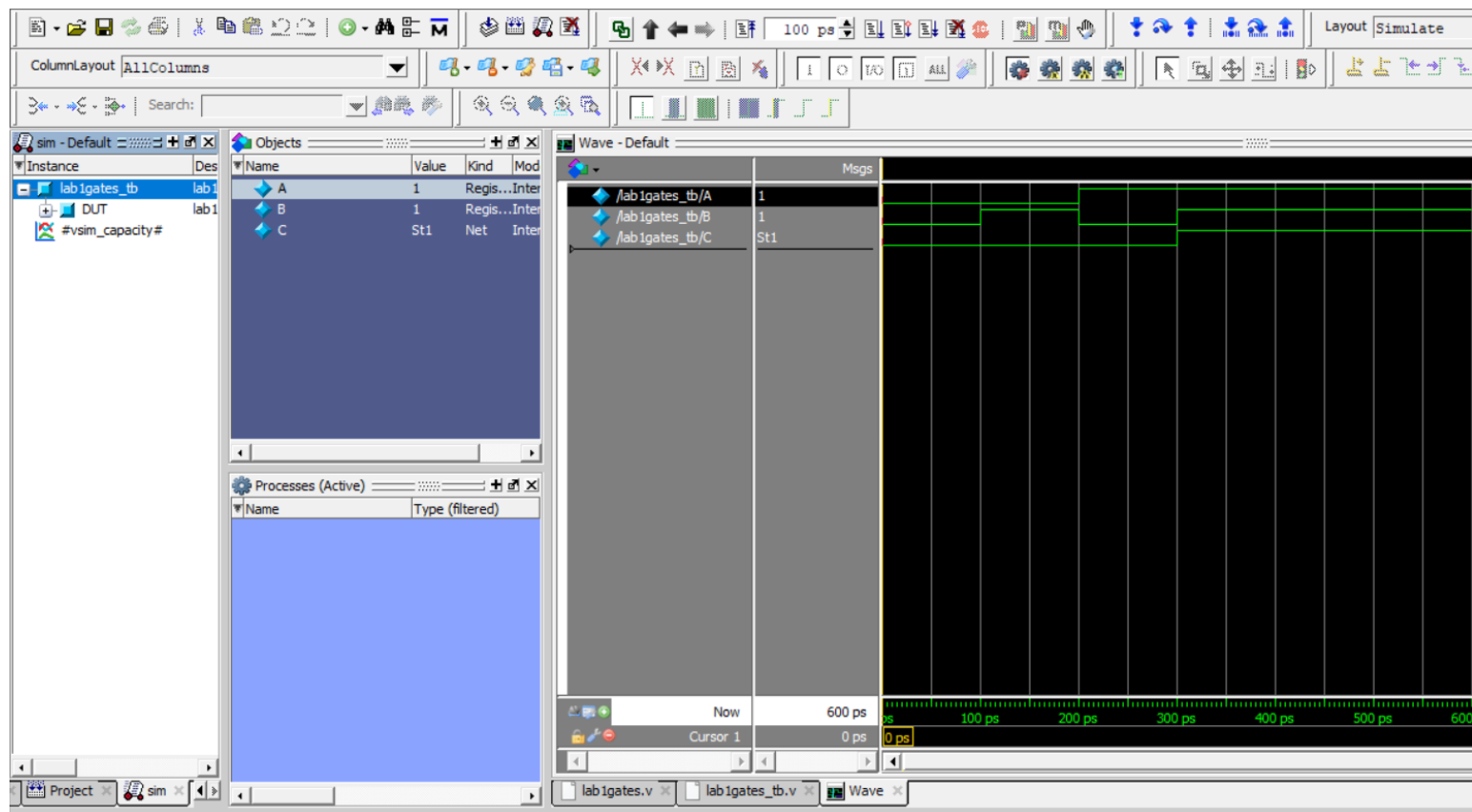
                 input A, B;

                 output C;

                 and G3(C, A, B);

         endmodule

Ln#	
1	module lab1gates_tb ();
2	
3	reg A, B;
4	wire C;
5	
6	lab1gates DUT (A, B, C);
7	
8	initial begin
9	A=0;B=0; #100;
10	A=0;B=1; #100;
11	A=1;B=0; #100;
12	A=1;B=1; #100;
13	
14	end
15	endmodule
16	



Modelsim Simulation for an AND gate

**c) OR:**

```

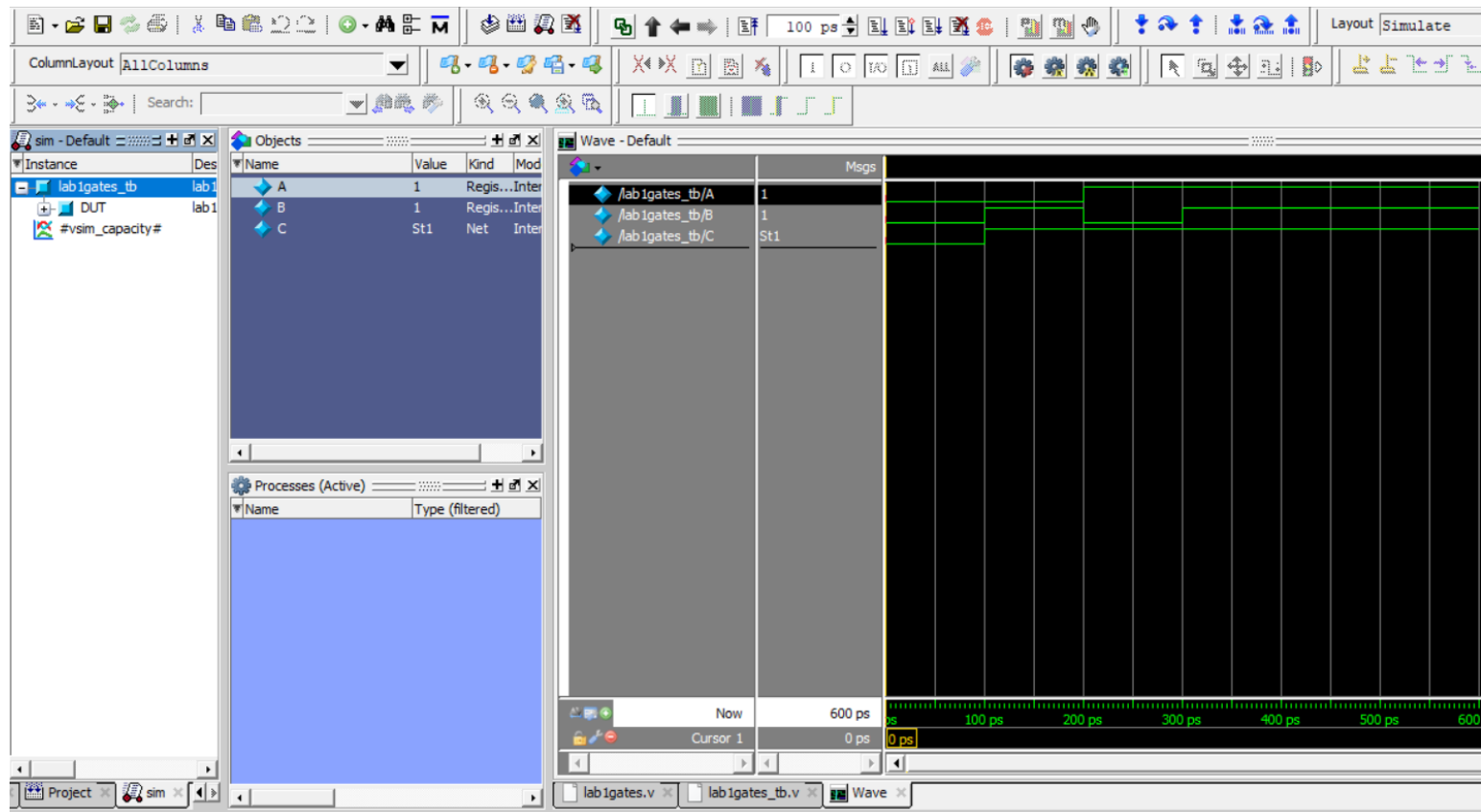
module lab1gates (A, B, C);

    input A, B;
    output C;
    or G4(C, A, B);

endmodule

```

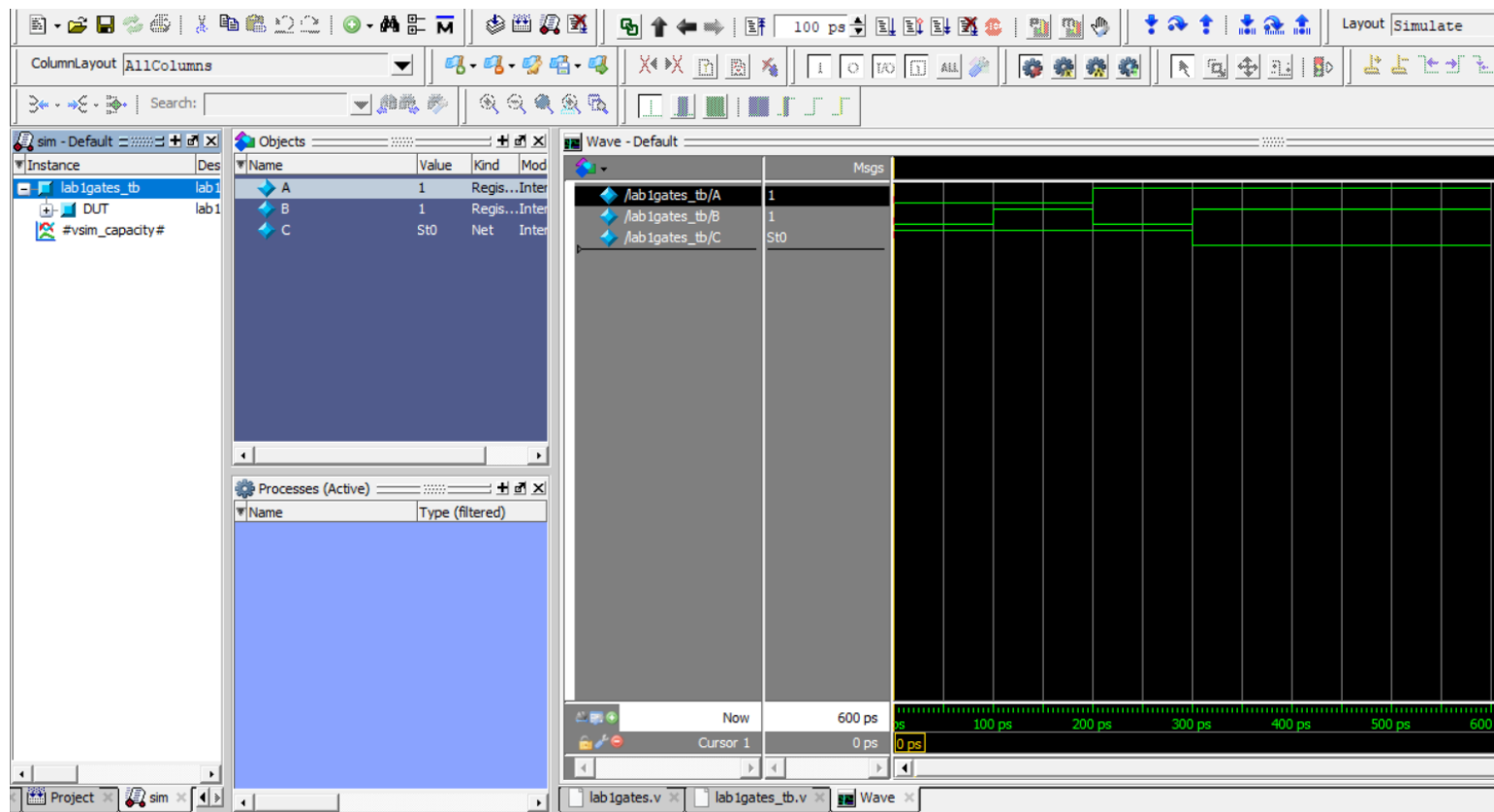
Ln#	
1	module lab1gates_tb ();
2	
3	reg A, B;
4	wire C;
5	
6	lab1gates DUT (A, B, C);
7	
8	initial begin
9	A=0;B=0; #100;
10	A=0;B=1; #100;
11	A=1;B=0; #100;
12	A=1;B=1; #100;
13	
14	end
15	endmodule
16	



Modelsim Simulation for the OR gate

**d) NAND:** module lab1gates (A, B, C);  
input A, B;  
output C;  
nand G5(C, A, B);  
endmodule

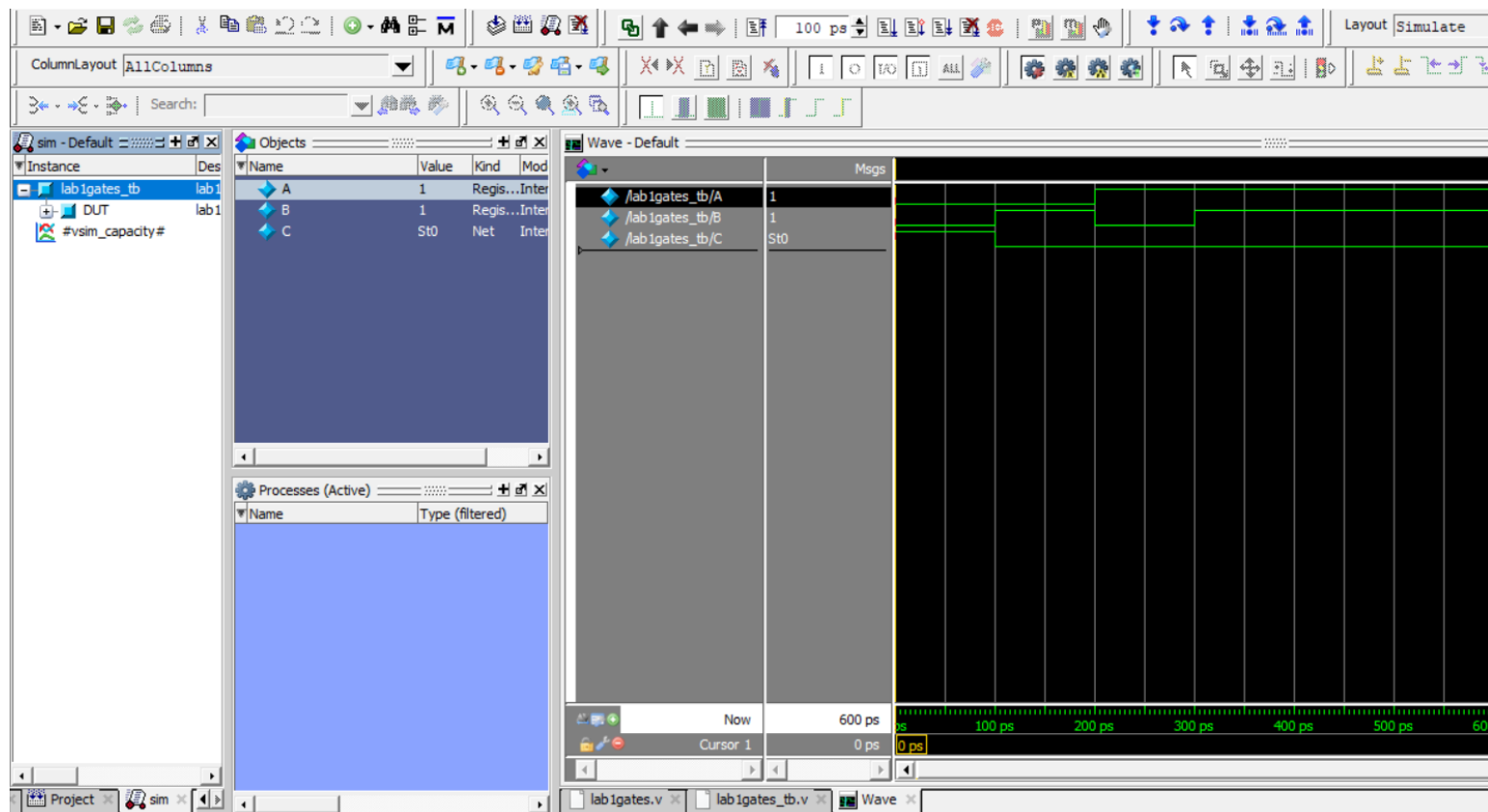
Ln#	
1	module lab1gates_tb ();
2	
3	reg A, B;
4	wire C;
5	
6	lab1gates DUT (A, B, C);
7	
8	initial begin
9	A=0;B=0; #100;
10	A=0;B=1; #100;
11	A=1;B=0; #100;
12	A=1;B=1; #100;
13	
14	end
15	endmodule
16	



Modelsim Simulation for the NAND gate

**e) NOR:**    module lab1gates (A, B, C);  
                   input A, B;  
                   output C;  
                   nor G6(C, A, B);  
                   endmodule

Ln#	
1	module lab1gates_tb ();
2	
3	reg A, B;
4	wire C;
5	
6	lab1gates DUT (A, B, C);
7	
8	initial begin
9	A=0;B=0; #100;
10	A=0;B=1; #100;
11	A=1;B=0; #100;
12	A=1;B=1; #100;
13	
14	end
15	endmodule
16	

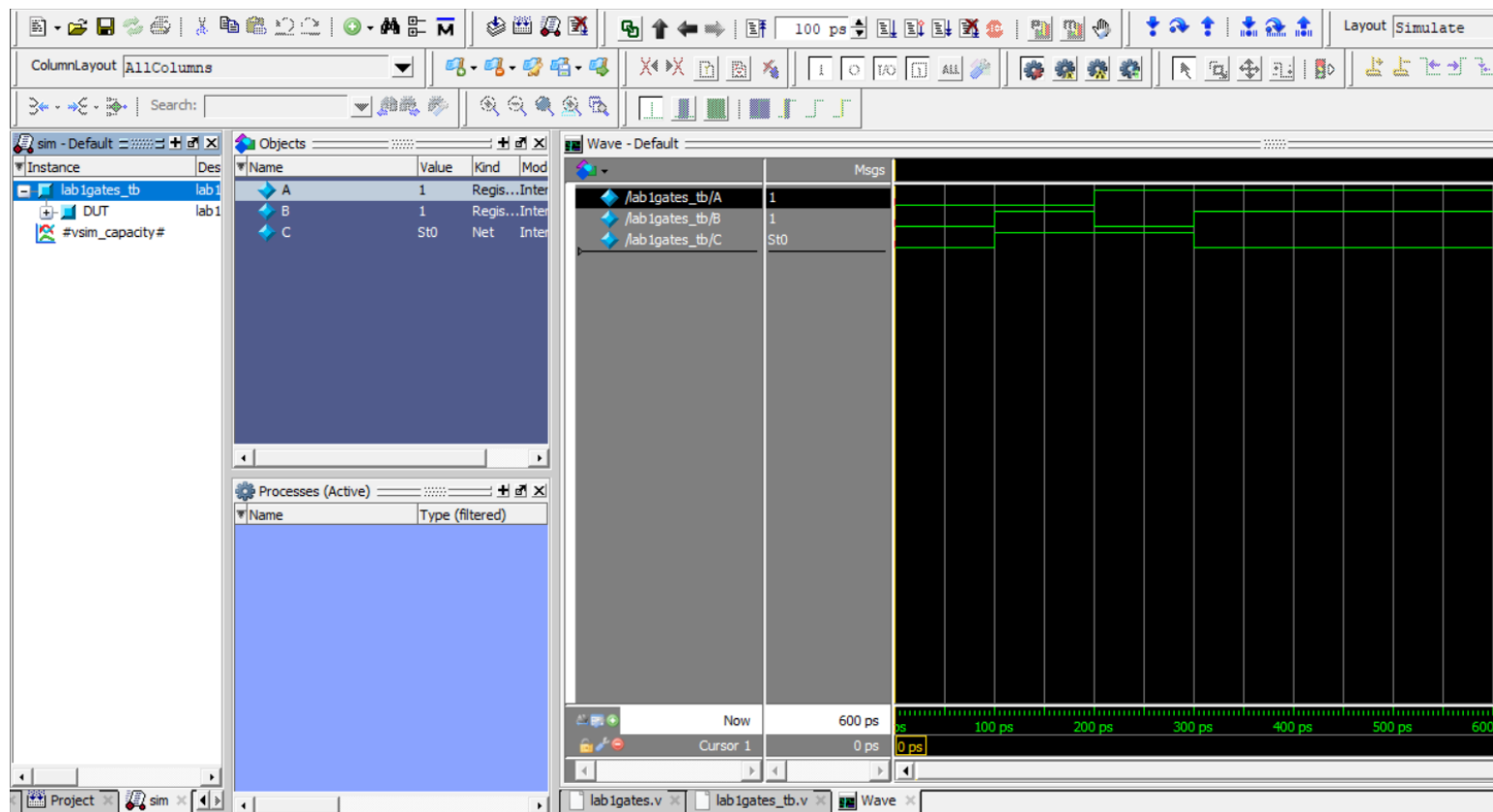


Modelsim Simulation for the NOR gate

**f) XOR:**     module lab1gates (A, B, C);  
                   input A, B;  
                   output C;  
                   xor G7(C, A, B);  
                   endmodule

Ln#	
1	module lab1gates_tb ();
2	
3	reg A, B;
4	wire C;
5	
6	lab1gates DUT (A, B, C);
7	
8	initial begin
9	A=0;B=0; #100;
10	A=0;B=1; #100;
11	A=1;B=0; #100;
12	A=1;B=1; #100;
13	
14	end
15	endmodule
16	

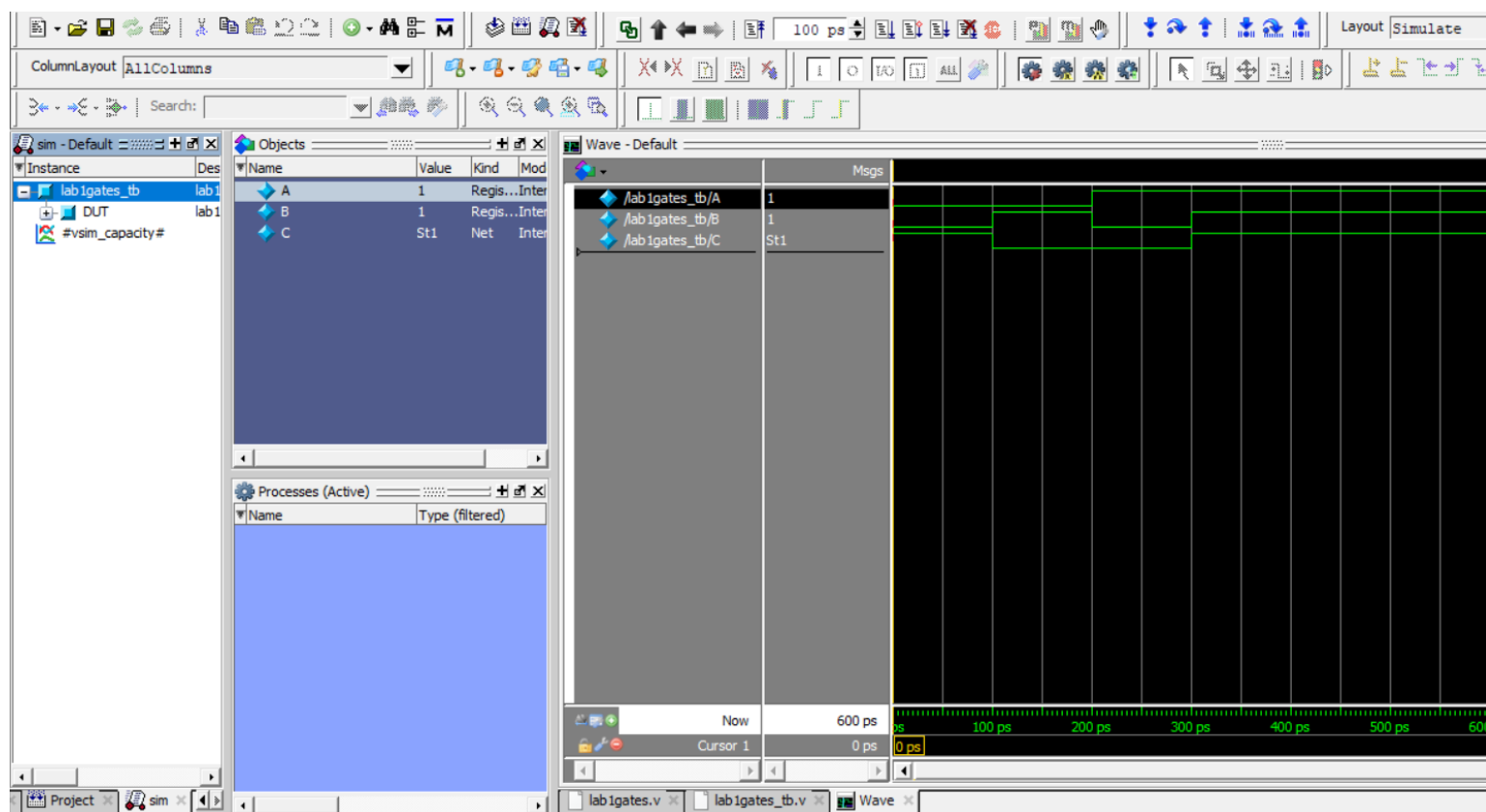




Modelsim Simulation for the XOR gate

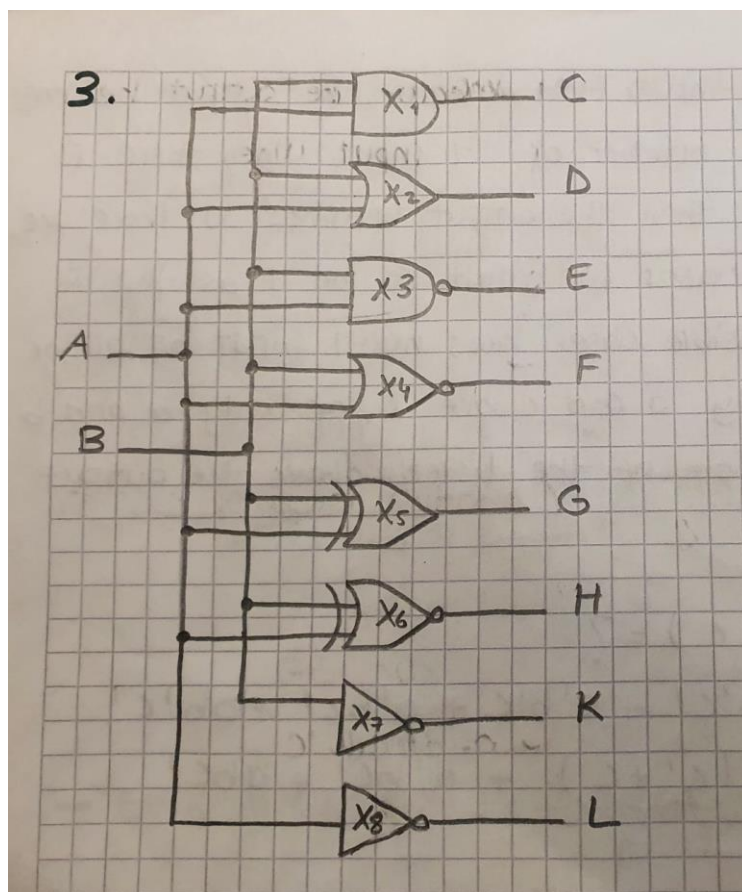
**g) XNOR:**    module lab1gates (A, B, C);  
                   input A, B;  
                   output C;  
                   xnor G8(C, A, B);  
                   endmodule

Ln#	
1	module lab1gates_tb ();
2	
3	reg A, B;
4	wire C;
5	
6	lab1gates DUT (A, B, C);
7	
8	initial begin
9	A=0;B=0; #100;
10	A=0;B=1; #100;
11	A=1;B=0; #100;
12	A=1;B=1; #100;
13	
14	end
15	endmodule
16	



Modelsim Simulation for the XNOR gate

3. After naming the outputs as shown below:

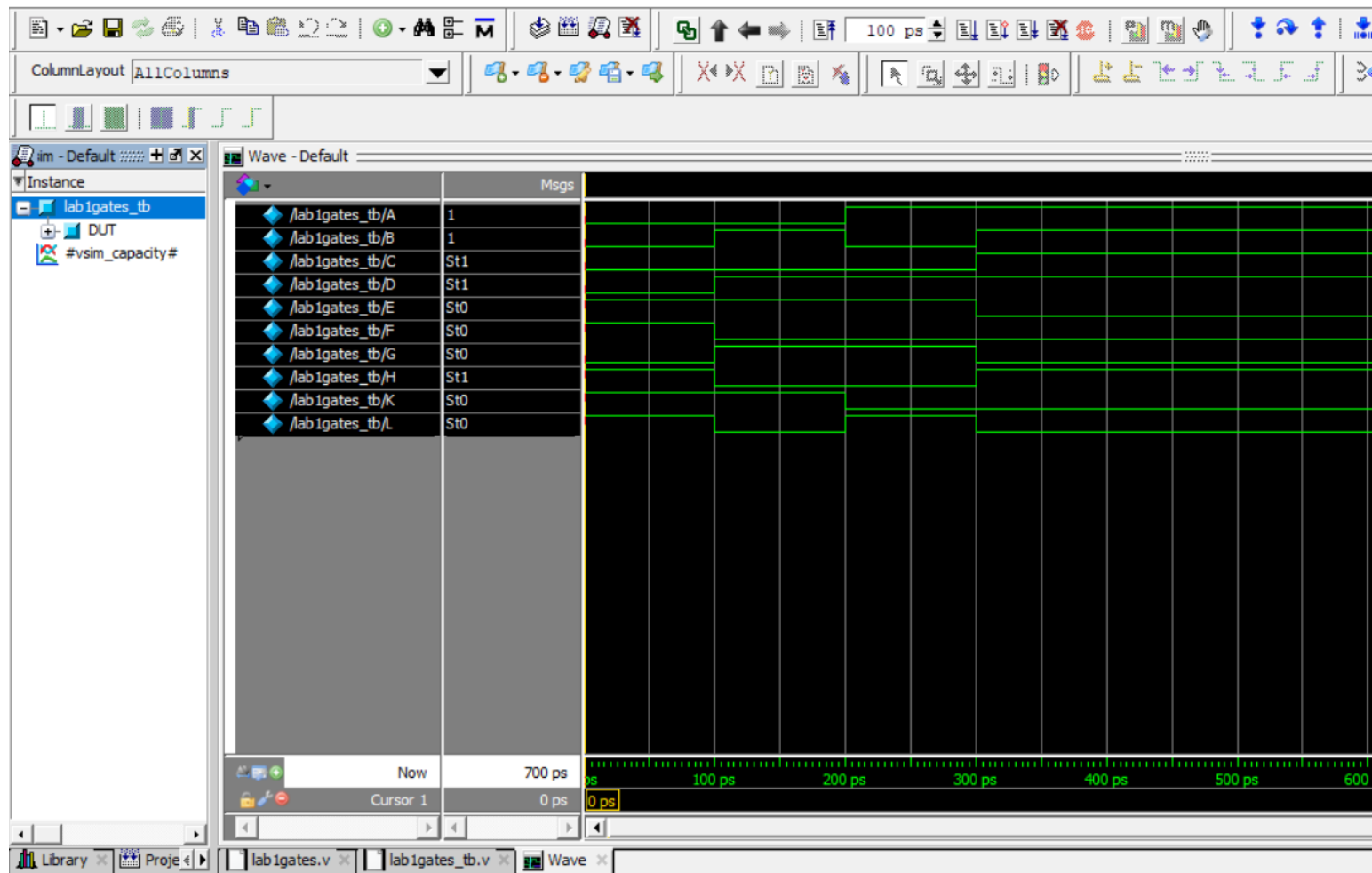


Verilog code for the circuit in Figure 1:

```
module lab1gates (A, B, C, D, E, F, G, H, K, L);  
    input A, B;  
    output C,D, E, F, G, H, K, L;  
    and X1(C, A, B);  
    or X2(D, A, B);  
    nand X3(E, A, B);  
    nor X4(F, A, B);  
    xor X5(G, A, B);  
    xnor X6(H, A, B);  
    not X7(K, A, B);  
    not X8(L, A, B);  
endmodule
```

Testbench:

Ln#	
1	<code>module lab1gates_tb ();</code>
2	
3	<code>reg A, B;</code>
4	<code>wire C, D, E, F, G, H, K, L;</code>
5	
6	<code>lab1gates DUT (A, B, C, D, E, F, G, H, K, L);</code>
7	
8	<code>initial begin</code>
9	<code>    A=0;B=0; #100;</code>
10	<code>    A=0;B=1; #100;</code>
11	<code>    A=1;B=0; #100;</code>
12	<code>    A=1;B=1; #100;</code>
13	
14	<code>end</code>
15	<code>endmodule</code>
16	



Modelsim Simulation for the circuit in Figure 1

### 1.2.3 Complex Gates

1. In table (a) which is 3-input Minority gate, output becomes '1' when there is at most one '1'. When there are more than one '1' input, then the output becomes '0'. Therefore, there are four possible cases for output to be '1'. Either all inputs are '0', or only A is '1', or only B is '1', or only C is '1'. So, it is obvious that output becomes '1' where there is at most one '1' input.

In table (b) which is 3-input Even detector, the output becomes 1 when there is even number of '1' input. When there is an odd number of '1', then the output becomes '0'. Therefore, there are four possible cases for output to be '1'. So, the possible cases are: no '1' input at all, or only b and c are '1', or only a and c are '1', or only a and b are '1'. So, if the case is one of the listed above, the output becomes '1'.

#### 2.i. MN-out (a, b, c)=?

$$\begin{aligned}
 F(a, b, c) &= a'b'c' + a'b'c + a'bc' + ab'c' \\
 &= a'b'(c' + c) + a'bc' + abc' \\
 &= a'b' + a'bc' + abc'
 \end{aligned}$$

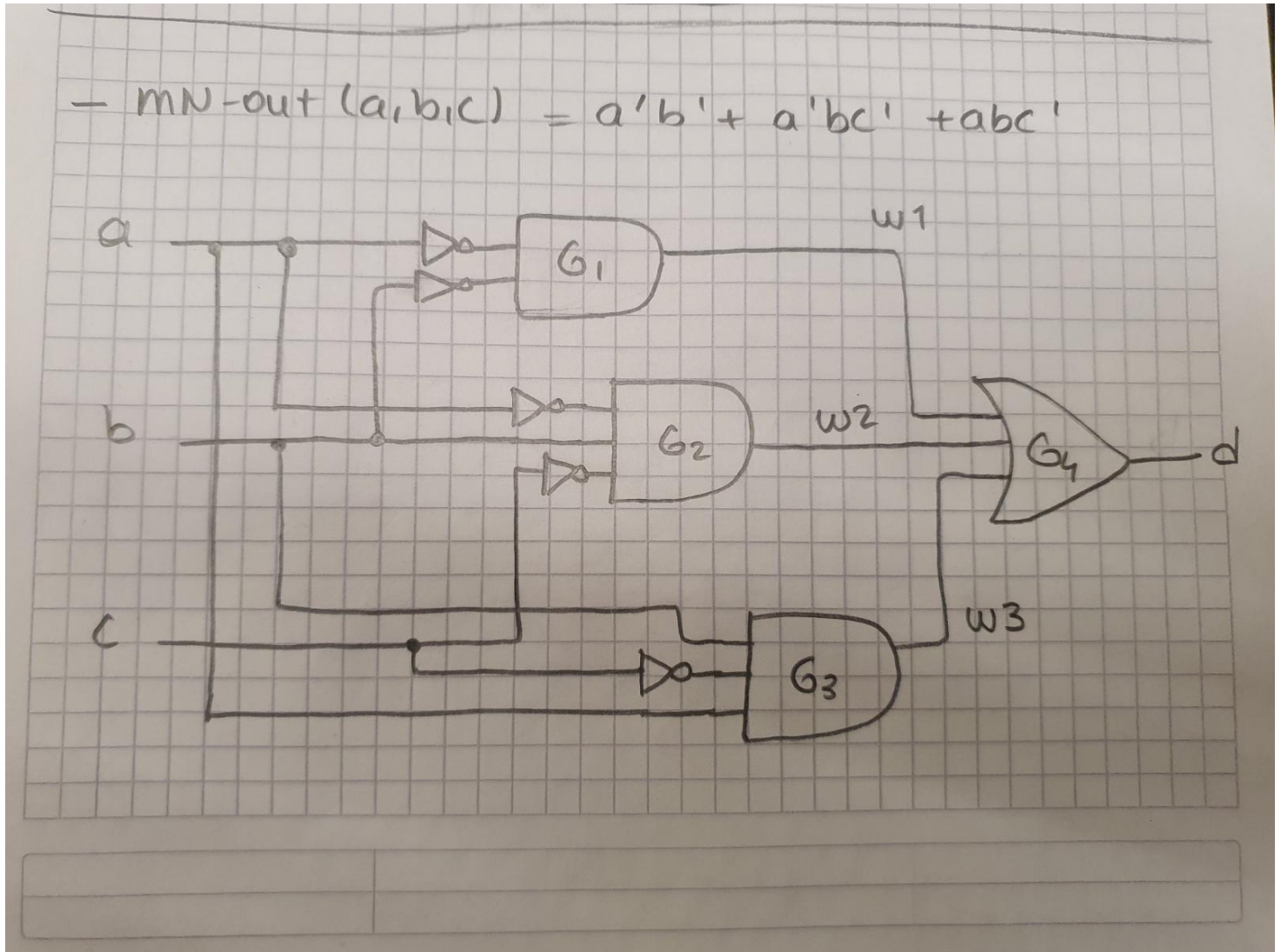
$$\text{MN-out (a, b, c)} = a'b' + a'bc' + abc'$$

ii. Even-out (a, b, c)=?

$$F(a, b, c) = a'b'c' + a'bc + ab'c + abc'$$

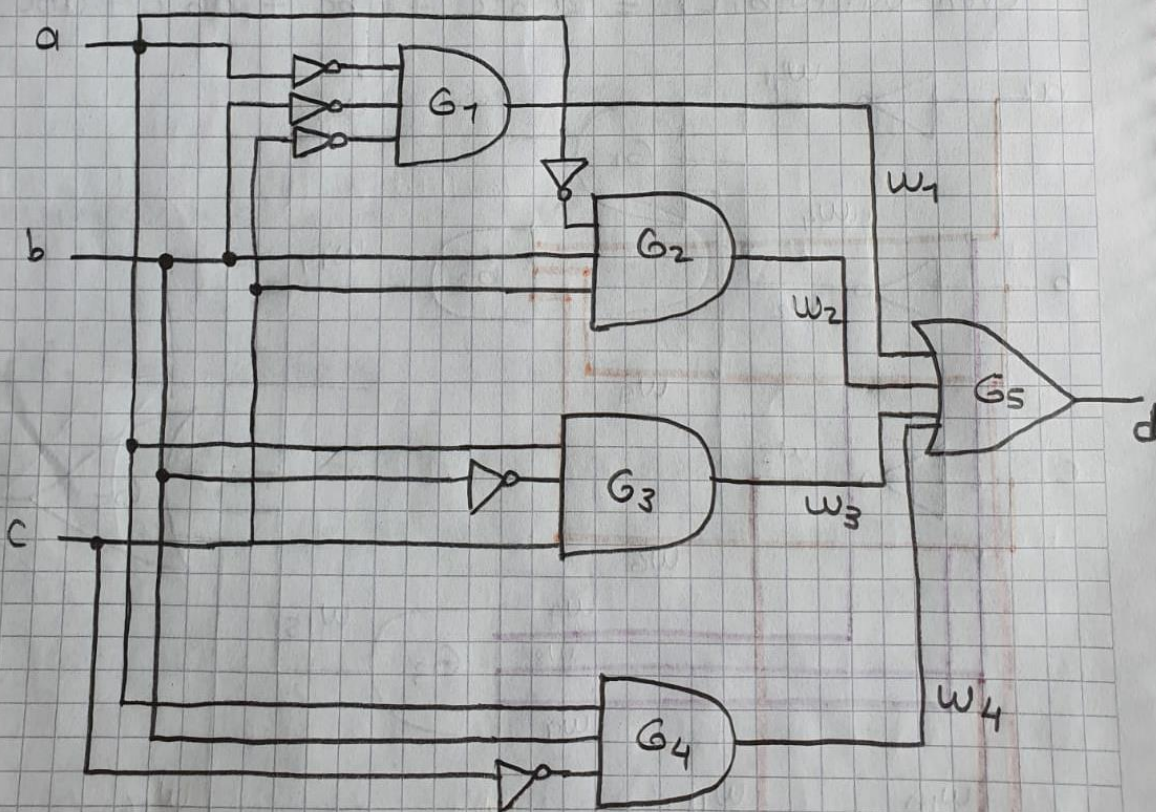
Even-out (a, b, c) =  $a'b'c' + a'bc + ab'c + abc'$  (There is no simplification)

3. Schematic implementations for both MN-out and Even-out functions:





— Even-out ( $a, b, c$ ) =  $a'b'c' + a'bc + ab'c + abc'$



Verilog codes:

For MN-out:

```
module lab1complex (a, b, c, d);
    input a, b, c;
    output d;
    wire w1, w2, w3;
    and G1(w1, !a, !b);
    and G2(w2, !a, b, !c);
    and G3(w3, a, b, !c);
    or G4(d, w1, w2, w3);
endmodule
```

For Even-out:

```
module lab1complex(a, b, c, d);
    input a, b, c;
    output d;
    wire w1, w2, w3, w4;
    and G1(w1, !a, !b, !c);
    and G2(w2, !a, b, c);
    and G3(w3, a, !b, c);
    and G4(w4, a, b, !c);
    or G5(d, w1, w2, w3, w4);
endmodule
```

Testbench:

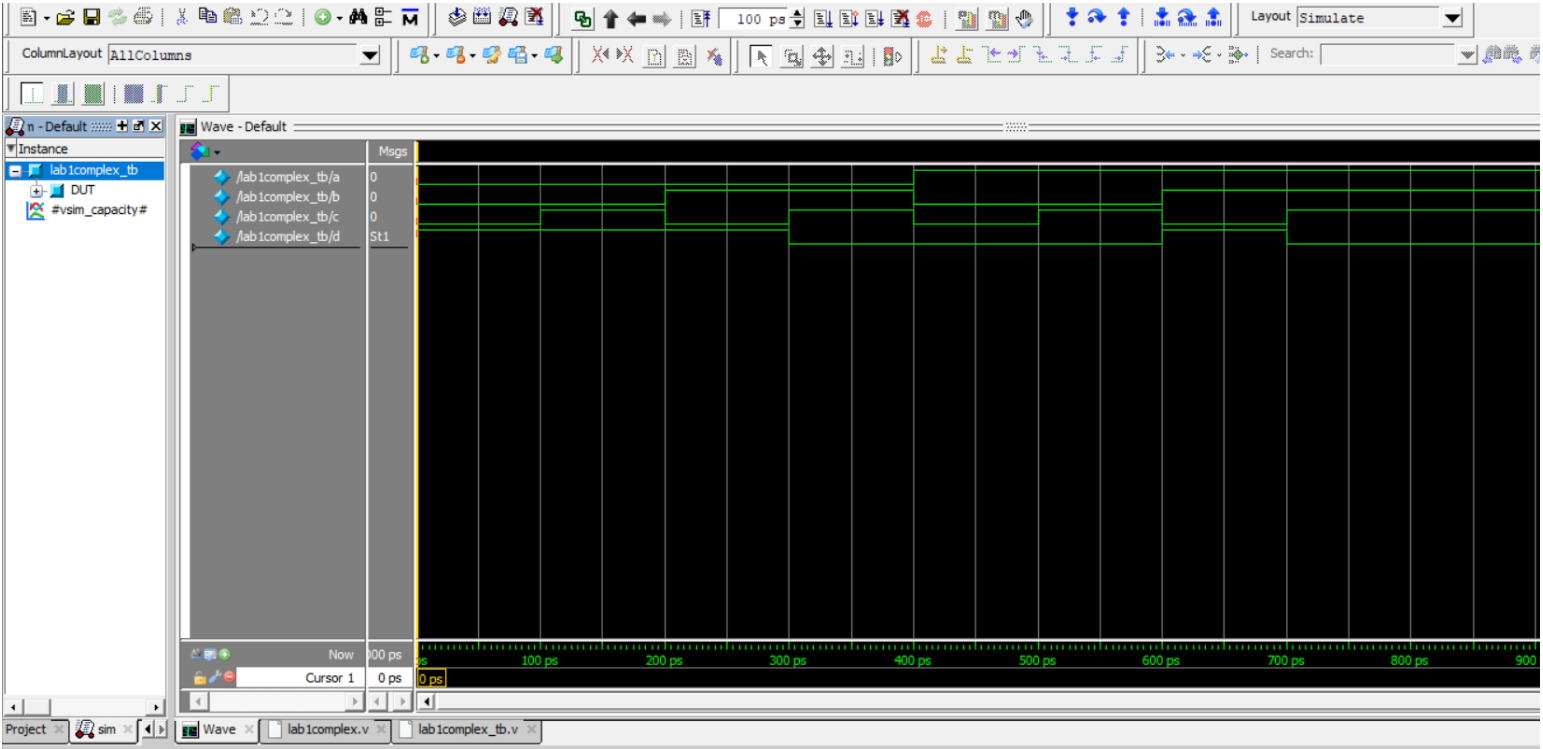
For MN-out

```
Ln#
1  module lab1complex_tb();
2      reg a, b, c;
3      wire d;
4
5      lab1complex DUT(a, b, c, d);
6
7      initial begin
8          a=0; b=0; c=0; #100;
9          a=0; b=0; c=1; #100;
10         a=0; b=1; c=0; #100;
11         a=0; b=1; c=1; #100;
12         a=1; b=0; c=0; #100;
13         a=1; b=0; c=1; #100;
14         a=1; b=1; c=0; #100;
15         a=1; b=1; c=1; #100;
16
17     end
18 endmodule
19
20
```

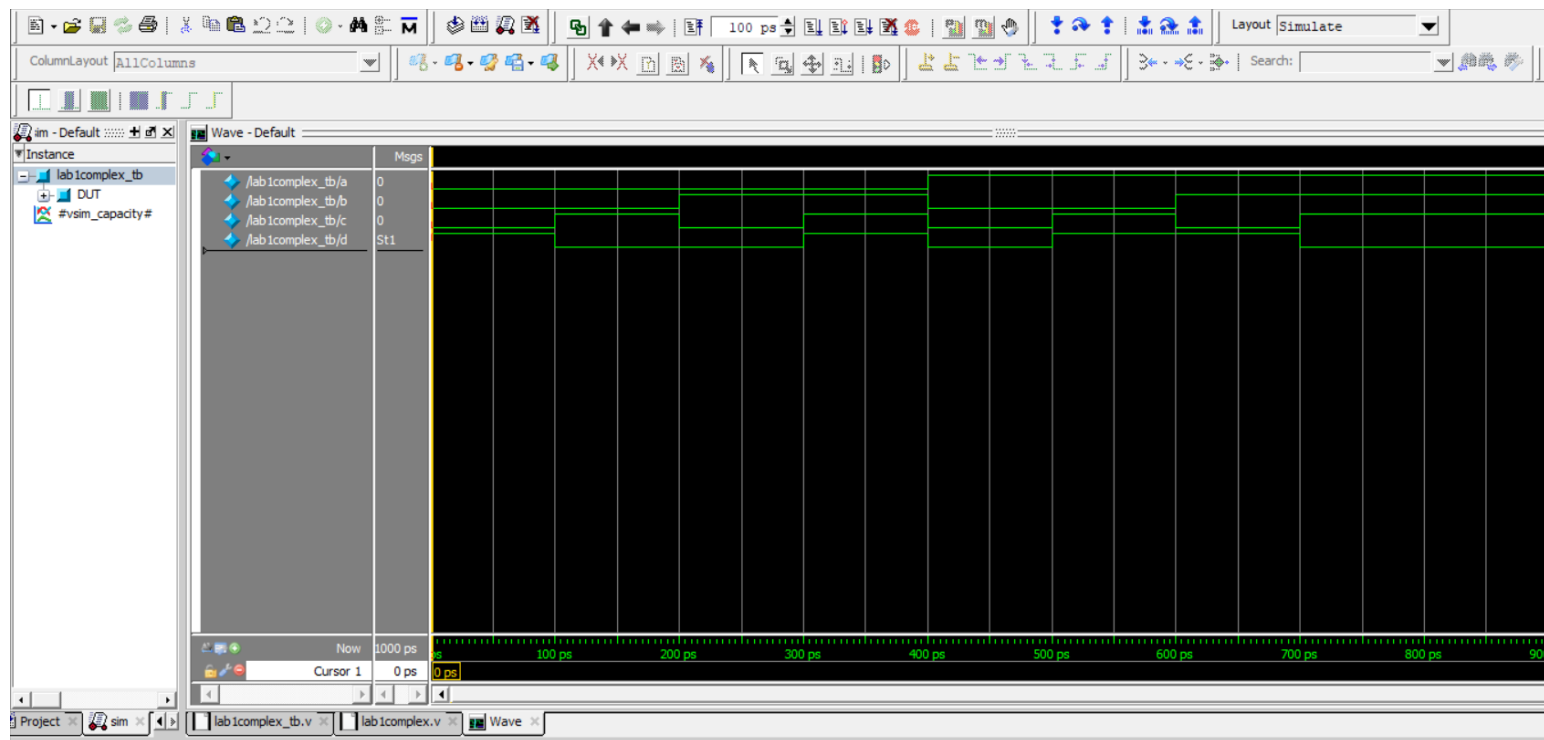
For Even-out

```
Ln#
1  module lab1complex_tb();
2      reg a, b, c;
3      wire d;
4
5      lab1complex DUT(a, b, c, d);
6
7      initial begin
8          a=0; b=0; c=0; #100;
9          a=0; b=0; c=1; #100;
10         a=0; b=1; c=0; #100;
11         a=0; b=1; c=1; #100;
12         a=1; b=0; c=0; #100;
13         a=1; b=0; c=1; #100;
14         a=1; b=1; c=0; #100;
15         a=1; b=1; c=1; #100;
16
17     end
18 endmodule
19
20
```

Simulations:



For MN-out



For Even-out

#### 4. XNOR truth table with 3-inputs:

a	b	c	d
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

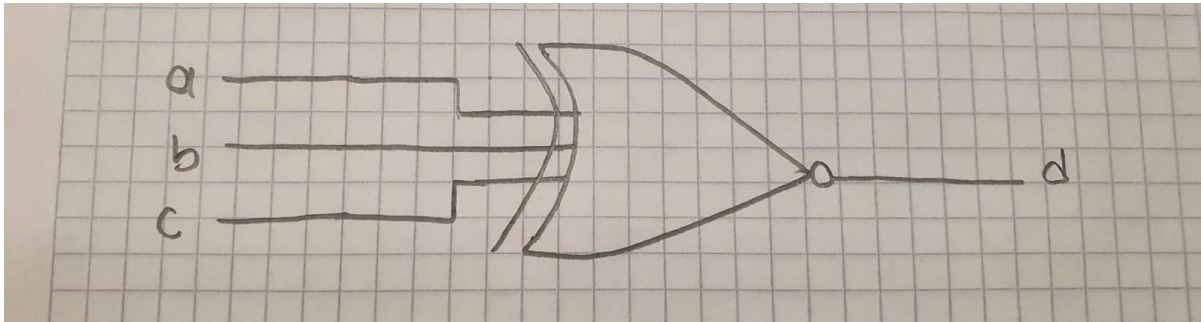
So,  $d(a, b, c) = a'b'c' + a'bc + ab'c + abc'$ .

The result is same with Even-out (a,b,c) function, so this function can be written as:

**Even-out**  $(a, b, c) = (a \oplus b \oplus c)'$  which is an exact expression for XNOR.



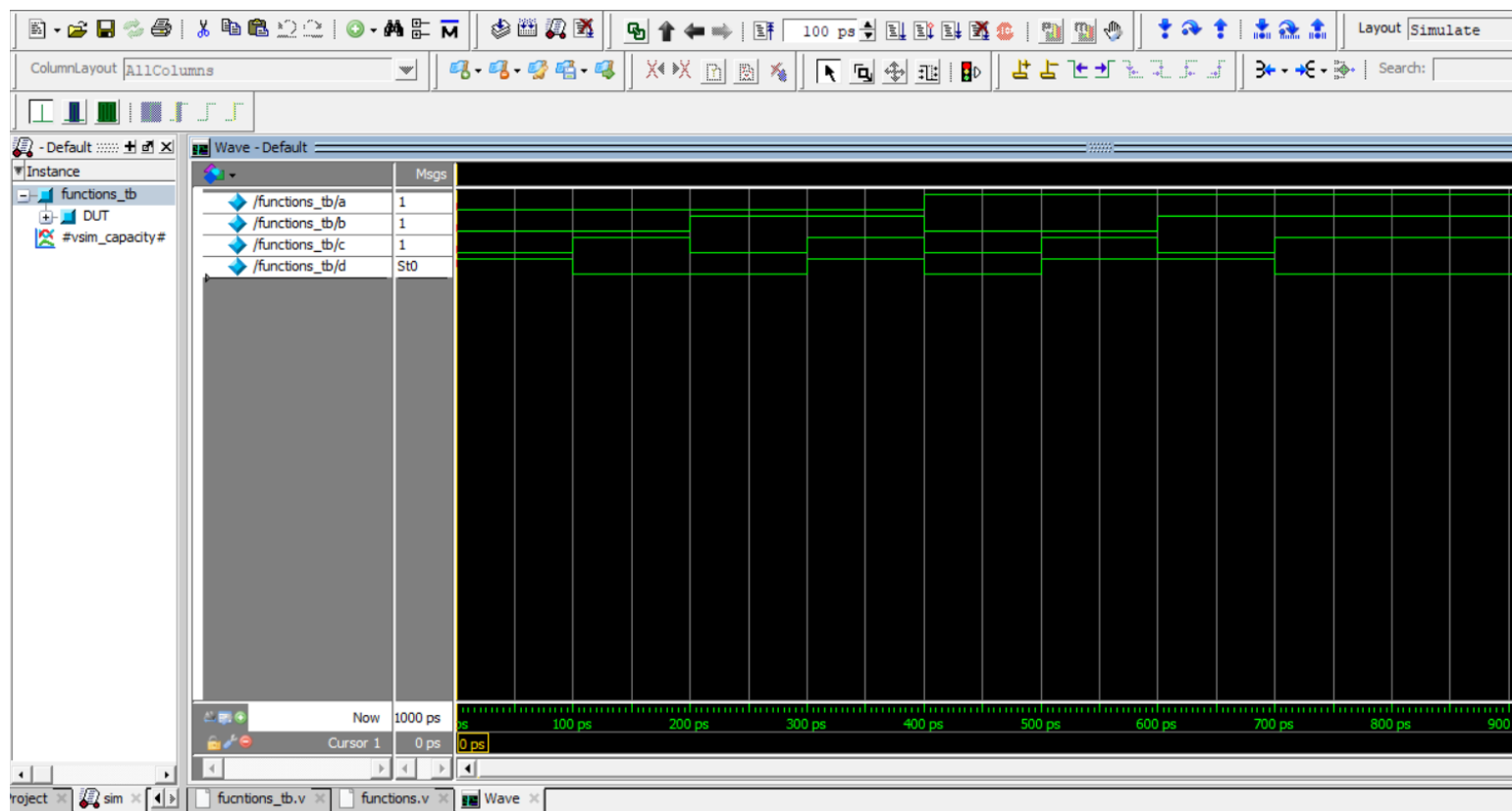
And its schematic would look like this:



After converting, the Verilog code looks like this:

```
module lab1complex (a, b, c, d);  
    input a, b, c;  
    output d;  
    xnor (d, a, b, c);  
endmodule
```

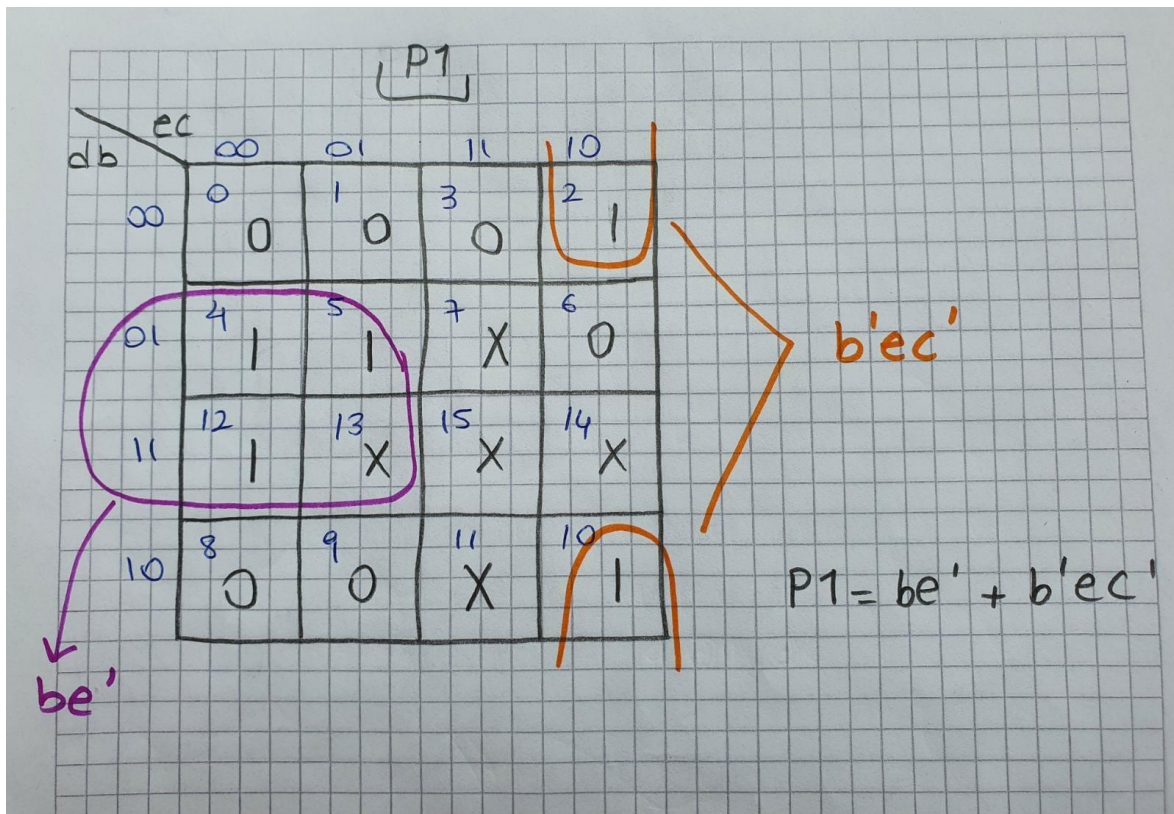
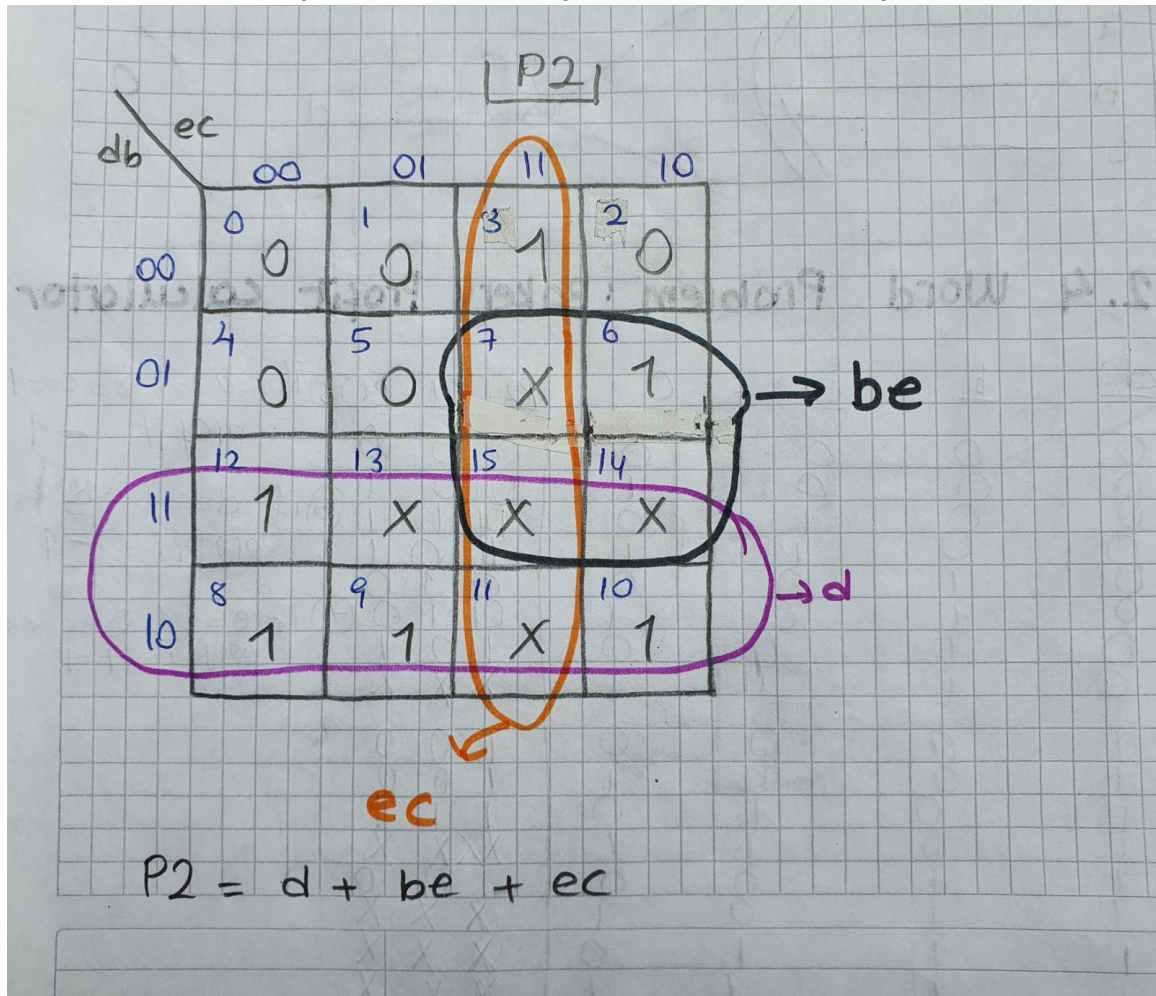
And the simulation:

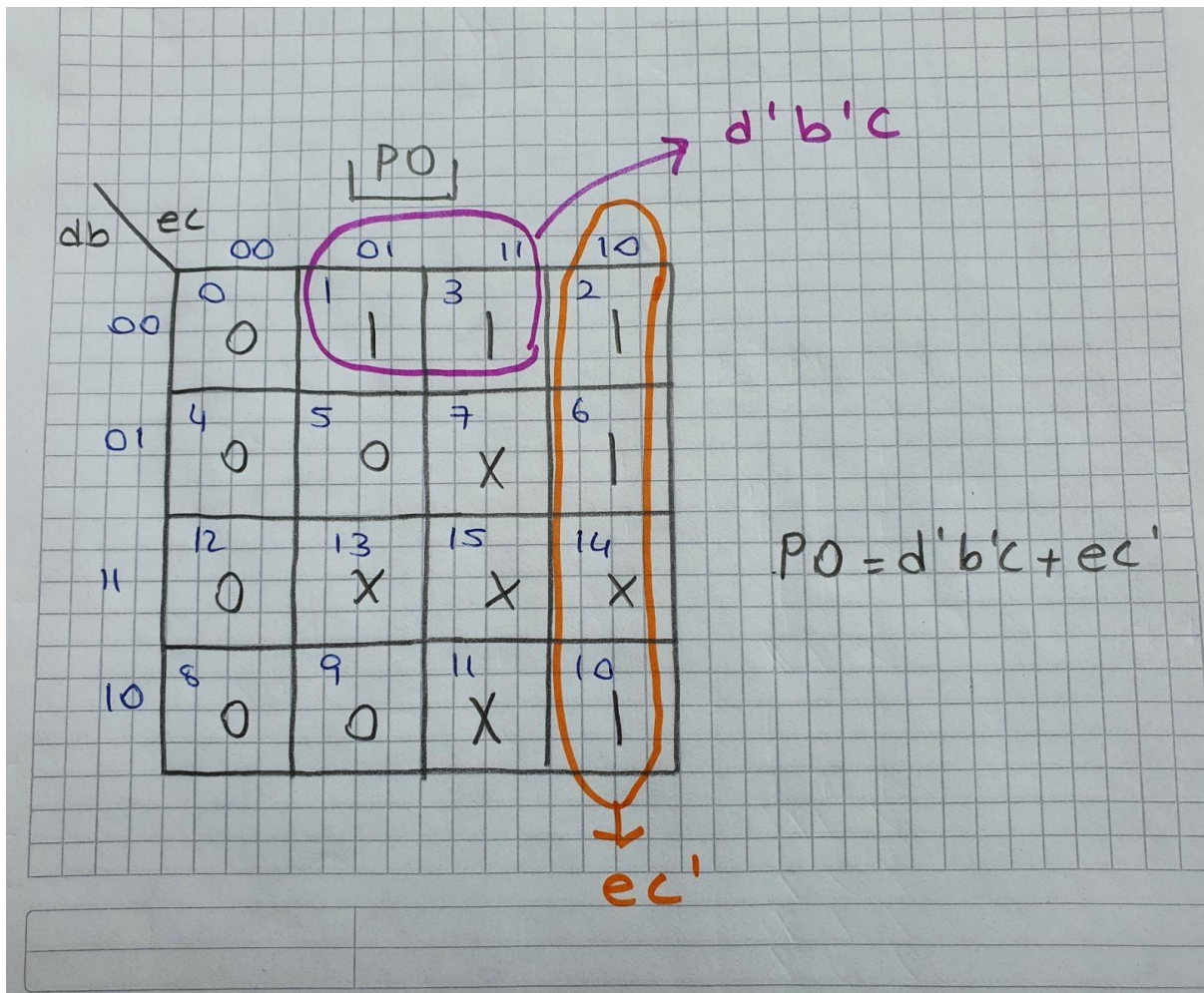


1.2.4 Word Problem: Bakery Profit Calculator

D	B	E	C	P2	P1	P0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	1
0	0	1	1	1	0	1
0	1	0	0	0	1	0
0	1	0	1	0	1	0
0	1	1	0	1	0	1
0	1	1	1	X	X	X
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	1	1	1
1	0	1	1	X	X	X
1	1	0	0	1	1	0
1	1	0	1	X	X	X
1	1	1	0	X	X	X
1	1	1	1	X	X	X

# K-Maps and Boolean Expressions for each output:





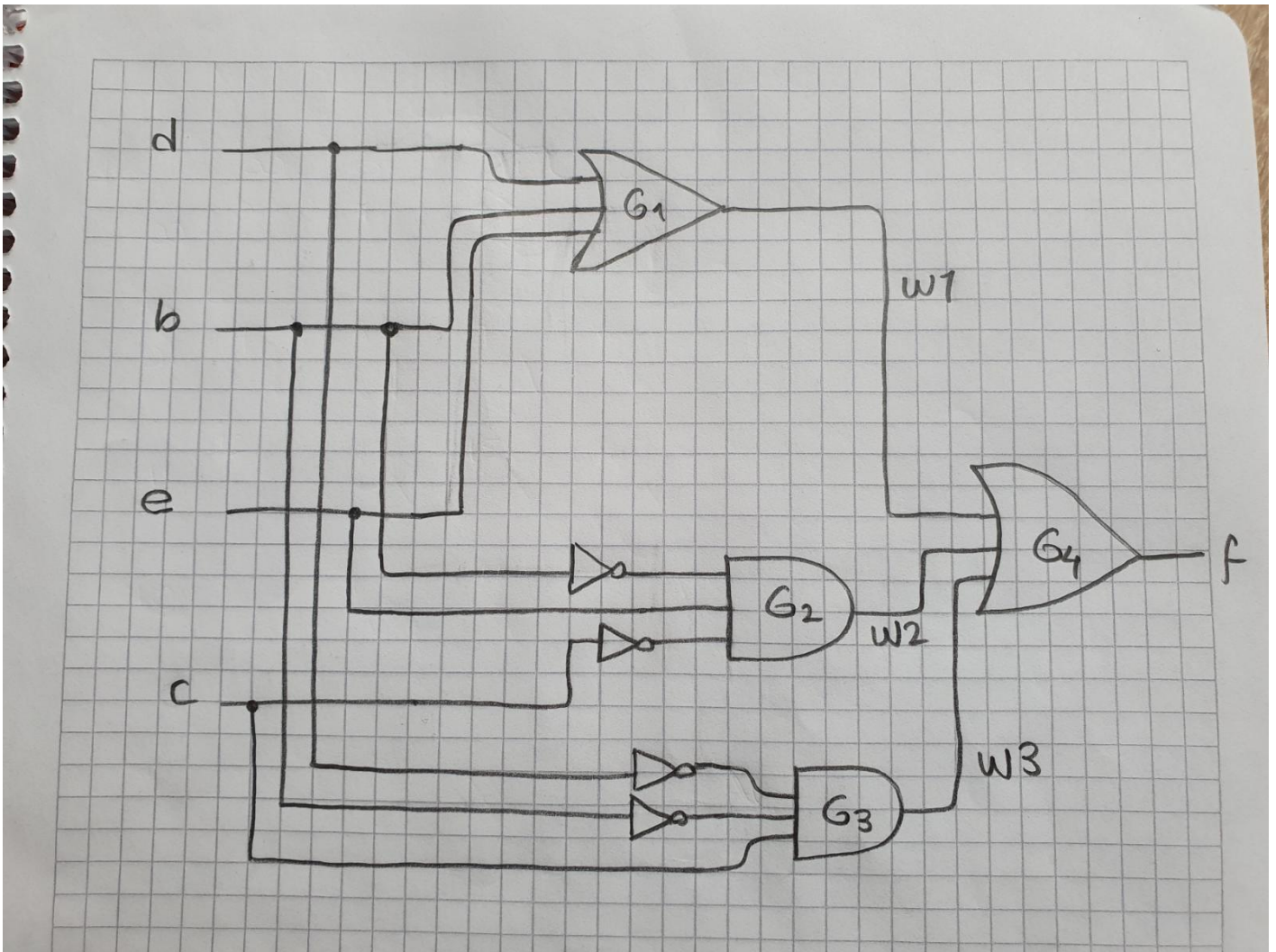
So, the final output for the circuit is:

$$\begin{aligned}
 f(d, b, e, c) &= P_2 + P_1 + P_0 = d + be + ec + be' + b'ec' + d'b'c + ec' \\
 &= d + b(e+e') + e(c+c') + b'ec' + d'b'c \\
 &= d + b + e + b'ec' + d'b'c
 \end{aligned}$$

$$f(d, b, e, c) = d + b + e + b'ec' + d'b'c$$



The schematic of the circuit:



Verilog code for the circuit:

```
module lab1bakery(d, b, e, c, f);
    input d, b, e, c;
    output f;
    wire w1, w2, w3;
    or G1(w1, d, b, e);
    and G2(w2, !b, e, !c);
    and G3(w3, !d, !b, c);
    or G4(f, w1, w2, w3);
endmodule
```

## Testbench:

Ln#	
1	module lablbakery_tb();
2	reg d, b, e, c;
3	wire f;
4	
5	lablbakery DUT(d, b, e, c, f);
6	
7	initial begin
8	d=0; b=0; e=0; c=0; #100;
9	d=0; b=0; e=0; c=1; #100;
10	d=0; b=0; e=1; c=0; #100;
11	d=0; b=0; e=1; c=1; #100;
12	d=0; b=1; e=0; c=0; #100;
13	d=0; b=1; e=0; c=1; #100;
14	d=0; b=1; e=1; c=0; #100;
15	d=0; b=1; e=1; c=1; #100;
16	d=1; b=0; e=0; c=0; #100;
17	d=1; b=0; e=0; c=1; #100;
18	d=1; b=0; e=1; c=0; #100;
19	d=1; b=0; e=1; c=1; #100;
20	d=1; b=1; e=0; c=0; #100;
21	d=1; b=1; e=0; c=1; #100;
22	d=1; b=1; e=1; c=0; #100;
23	d=1; b=1; e=1; c=1; #100;
24	
25	end
26	endmodule
27	
28	
29	

☐ lablbakery\_tb.v x ☐ lablbakery.v x ☒ Wave x

The simulation:

