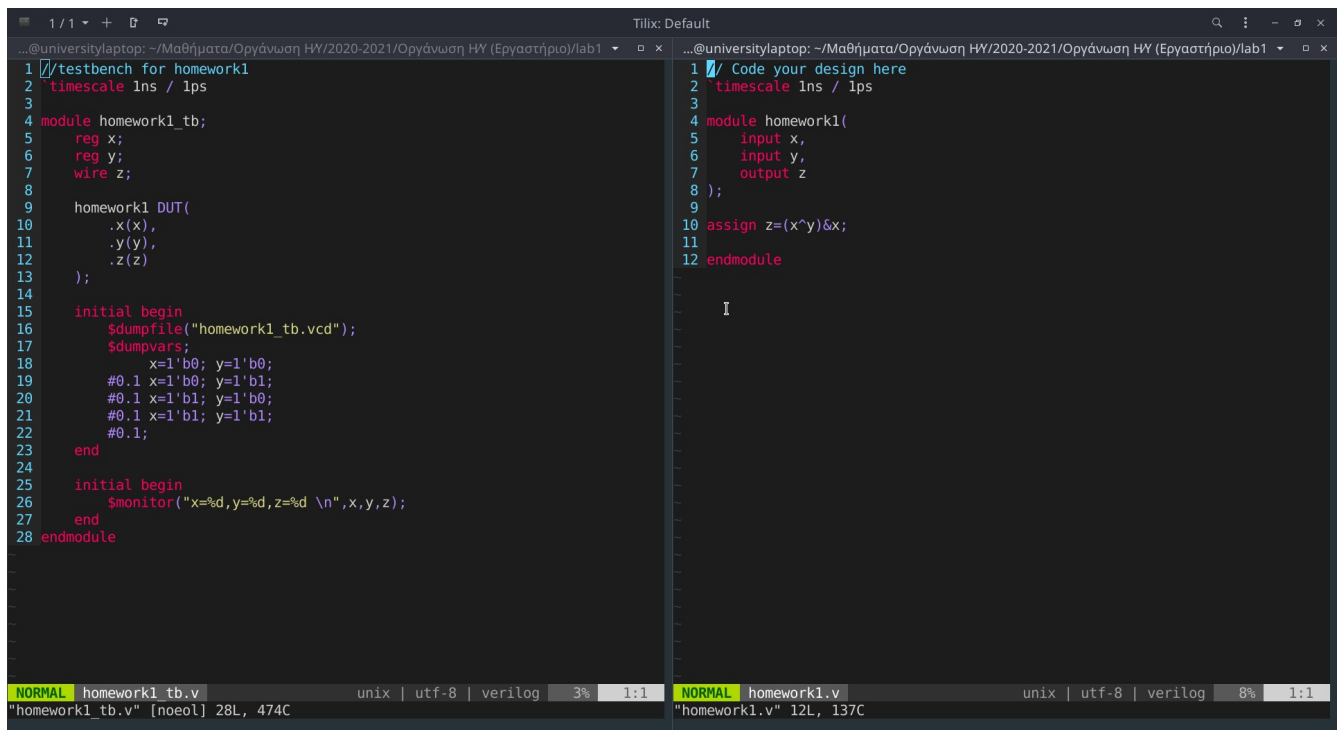


Μοσχογιάννης Πασχάλης AM 2114026

Άσκηση 1



The screenshot shows a Verilog IDE with two files open. The left file, `homework1_tb.v`, is a testbench for a module named `homework1`. It defines three registers: `x`, `y`, and `z`. The testbench initializes `x` and `y` to 1'b0 and then applies a series of test vectors. The right file, `homework1.v`, is the design under test (DUT). It defines a module `homework1` with two inputs, `x` and `y`, and one output, `z`. The output `z` is assigned the value `(x^y)&x`.

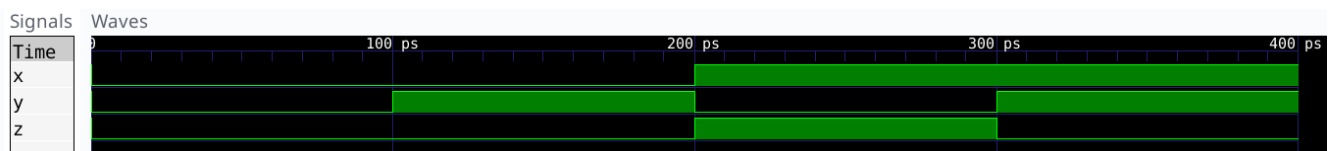
```
1 //testbench for homework1
2 timescale 1ns / 1ps
3
4 module homework1_tb;
5     reg x;
6     reg y;
7     wire z;
8
9     homework1 DUT(
10         .x(x),
11         .y(y),
12         .z(z)
13     );
14
15     initial begin
16         $dumpfile("homework1_tb.vcd");
17         $dumpvars;
18         x=1'b0; y=1'b0;
19         #0.1 x=1'b0; y=1'b1;
20         #0.1 x=1'b1; y=1'b0;
21         #0.1 x=1'b1; y=1'b1;
22         #0.1;
23     end
24
25     initial begin
26         $monitor("x=%d,y=%d,z=%d \n",x,y,z);
27     end
28 endmodule
```

```
1 // Code your design here
2 timescale 1ns / 1ps
3
4 module homework1(
5     input x,
6     input y,
7     output z
8 );
9
10 assign z=(x^y)&x;
11
12 endmodule
```

iverilog - Icarus Verilog compiler Version 10.2 (stable)
vvp - Icarus Verilog vvp runtime engine Version 10.2 (stable)

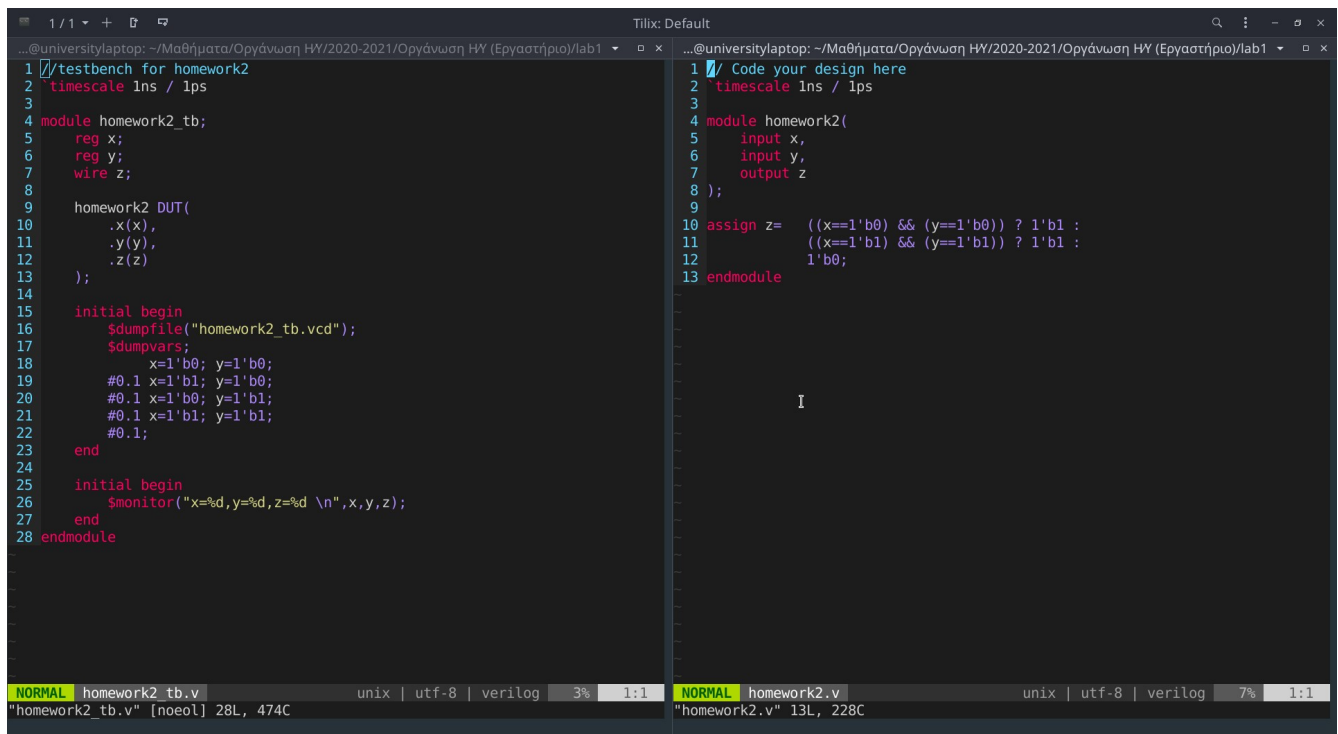
```
$ iverilog -Wall -o homework1_tb.vvp homework1_tb.v homework1.v
$ vvp homework1_tb.vvp
$ ls
homework1_tb.v
homework1_tb.vcd
homework1_tb.vvp
homework1.v
```

```
$ gtkwave homework1_tb.vcd
```



Άσκηση 2

X	Y	Z
0	0	1
1	0	0
0	1	0
1	1	1



```
1 //testbench for homework2
2 timescale 1ns / 1ps
3
4 module homework2_tb;
5     reg x;
6     reg y;
7     wire z;
8
9     homework2 DUT(
10         .x(x),
11         .y(y),
12         .z(z)
13     );
14
15     initial begin
16         $dumpfile("homework2_tb.vcd");
17         $dumpvars;
18         x=1'b0; y=1'b0;
19         #0.1 x=1'b1; y=1'b0;
20         #0.1 x=1'b0; y=1'b1;
21         #0.1 x=1'b1; y=1'b1;
22         #0.1;
23     end
24
25     initial begin
26         $monitor("x=%d,y=%d,z=%d \n",x,y,z);
27     end
28 endmodule
```

```
1 // Code your design here
2 timescale 1ns / 1ps
3
4 module homework2(
5     input x,
6     input y,
7     output z
8 );
9
10 assign z= ((x==1'b0) && (y==1'b0)) ? 1'b1 :
11           ((x==1'b1) && (y==1'b1)) ? 1'b1 :
12           1'b0;
13 endmodule
```

```
$ iverilog -Wall -o homework2_tb.vvp homework2_tb.v homework2.v
```

```
$ vvp homework2_tb.vvp
```

```
$ ls
```

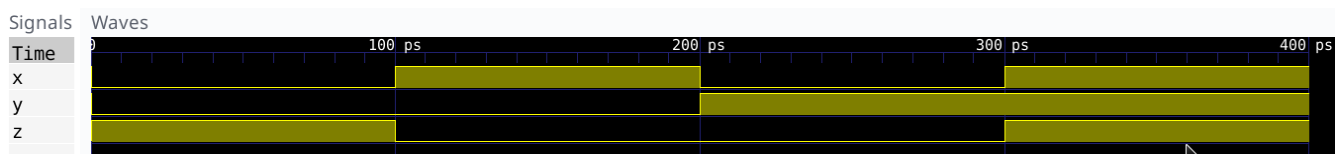
```
homework2_tb.v
```

```
homework2_tb.vcd
```

```
homework2_tb.vvp
```

```
homework2.v
```

```
$ gtkwave homework2_tb.vcd
```



Άσκηση 3

x(1)	x(0)	y(1)	y(0)	z
0	0	0	0	1
1	0	1	0	1
0	1	0	1	1
1	1	1	1	1

```

1 //testbench for homework3
2 `timescale 1ns / 1ps
3
4 module homework3_tb;
5     reg [1:0] x, y;
6     wire z;
7
8     homework3 DUT(
9         .x(x),
10        .y(y),
11        .z(z)
12    );
13
14    initial begin
15        $dumpfile("homework3_tb.vcd");
16        $dumpvars;
17        x=2'b00; y=2'b00;
18        #0.1 x=2'b00; y=2'b01;
19        #0.1 x=2'b00; y=2'b10;
20        #0.1 x=2'b00; y=2'b11;
21        #0.1 x=2'b01; y=2'b00;
22        #0.1 x=2'b01; y=2'b01;
23        #0.1 x=2'b01; y=2'b10;
24        #0.1 x=2'b01; y=2'b11;
25        #0.1 x=2'b10; y=2'b00;
26        #0.1 x=2'b10; y=2'b01;
27        #0.1 x=2'b10; y=2'b10;
28        #0.1 x=2'b10; y=2'b11;
29        #0.1 x=2'b11; y=2'b00;
30        #0.1 x=2'b11; y=2'b01;
31        #0.1 x=2'b11; y=2'b10;
32        #0.1 x=2'b11; y=2'b11;
33        #0.1;
34    end
35
36    initial begin
37        $monitor($time, x, y, z);
38    end
39 endmodule

```

```

1 // Code your design here
2 `timescale 1ns / 1ps
3
4 module homework3(
5     input [1:0] x,
6     input [1:0] y,
7     output z
8 );
9     assign z = ((x==2'b00) && (y==2'b00)) ? 1'b1 :
10                ((x==2'b10) && (y==2'b10)) ? 1'b1 :
11                ((x==2'b01) && (y==2'b01)) ? 1'b1 :
12                ((x==2'b11) && (y==2'b11)) ? 1'b1 :
13                2'b0;
14 endmodule

```

\$ iverilog -Wall -o homework3_tb.vvp homework3_tb.v homework3.v

\$ vvp homework3_tb.vvp

\$ ls

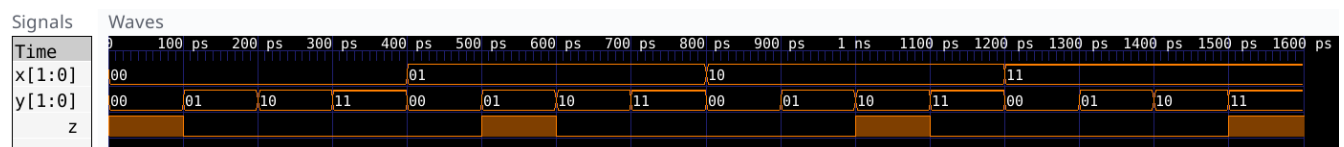
homework3_tb.v

homework3_tb.vcd

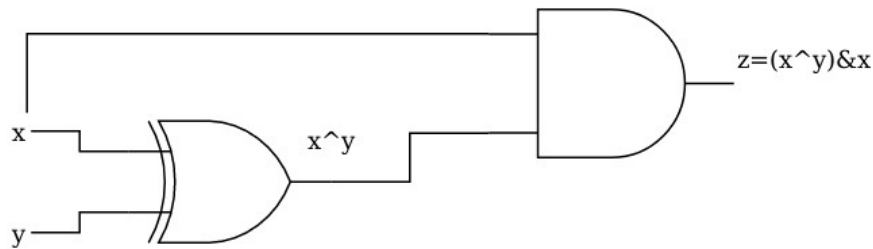
homework3_tb.vvp

homework3.v

\$ gtkwave homework3_tb.vcd



Άσκηση 4



```
1 //testbench for homework1
2 timescale 1ns / 1ps
3
4 module homework4_tb;
5     reg x;
6     reg y;
7     wire z;
8
9     homework4 DUT(
10         .x(x),
11         .y(y),
12         .z(z)
13     );
14
15     initial begin
16         $dumpfile("homework4_tb.vcd");
17         $dumpvars;
18         x=1'b0; y=1'b0;
19         #0.1 x=1'b0; y=1'b1;
20         #0.1 x=1'b1; y=1'b0;
21         #0.1 x=1'b1; y=1'b1;
22         #0.1;
23     end
24
25     initial begin
26         $monitor("x=%d,y=%d,z=%d \n",x,y,z);
27     end
28 endmodule
```

```
1 // Code your design here
2 timescale 1ns/1ps
3
4 module homework4(
5     output wire z,
6     input wire x,y
7 );
8
9     wire xy_xor;
10    xor U0 (xy_xor, x, y);
11    and U1 (z, xy_xor, x);
12
13 endmodule
```

```
$ iverilog -Wall -o homework3_tb.vvp homework3_tb.v homework3.v
```

```
$ vvp homework3_tb.vvp
```

```
$ ls
```

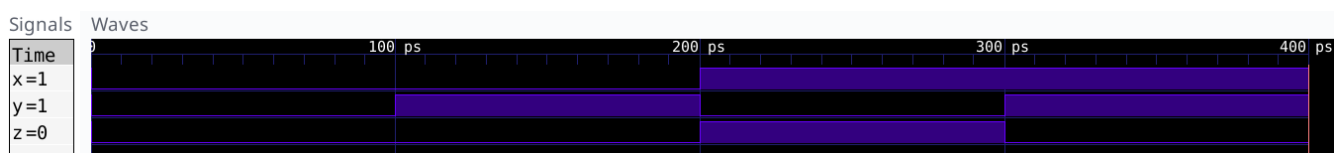
```
homework3_tb.v
```

```
homework3_tb.vcd
```

```
homework3_tb.vvp
```

```
homework3.v
```

```
$ gtkwave homework3_tb.vcd
```



Άσκηση 5

```
1 timescale 1ns/1ps
2 include "homework5.v"
3
4 module homework5_tb;
5 reg [7:0] A, B;
6 wire [7:0] Sum;
7 wire Cout;
8
9 ripple_carry DUT(
10 .Sum(Sum),
11 .Cout(Cout),
12 .A(A),
13 .B(B));
14
15 initial begin
16 $dumpfile("homework5_tb.vcd");
17 $dumpvars;
18 A=8'b00000000; B=8'b00001010;
19 #0.1 A=8'b01110001; B=8'b11001110;
20 #0.1 A=8'b01101100; B=8'b00100011;
21 #0.1 A=8'b00011100; B=8'b01001011;
22 #0.1;
23 end
24
25 initial begin
26 $monitor ("Value of A is %d, B is %d, and the calculated sum is %d,
27 as well as the Cout is %d at time %0t",A,B,Sum,Cout,$time);
28 end
29 endmodule
30
```

```
1 include "full_adder.v"
2 module ripple_carry(
3 output wire [7:0] Sum,
4 output wire Cout,
5 input wire [7:0] A, B);
6
7 wire c1, c2, c3, c4, c5, c6, c7;
8
9 full_adder U0(.Sum(Sum[0]), .Cout(c1), .A(A[0]), .B(B[0]), .Cin(1'b0));
10 full_adder U1(.Sum(Sum[1]), .Cout(c2), .A(A[1]), .B(B[1]), .Cin(c1));
11 full_adder U2(.Sum(Sum[2]), .Cout(c3), .A(A[2]), .B(B[2]), .Cin(c2));
12 full_adder U3(.Sum(Sum[3]), .Cout(c4), .A(A[3]), .B(B[3]), .Cin(c3));
13 full_adder U4(.Sum(Sum[4]), .Cout(c5), .A(A[4]), .B(B[4]), .Cin(c4));
14 full_adder U5(.Sum(Sum[5]), .Cout(c6), .A(A[5]), .B(B[5]), .Cin(c5));
15 full_adder U6(.Sum(Sum[6]), .Cout(c7), .A(A[6]), .B(B[6]), .Cin(c6));
16 full_adder U7(.Sum(Sum[7]), .Cout(Cout), .A(A[7]), .B(B[7]), .Cin(c7));
17 endmodule
NORMAL homework5.v unix | utf-8 | verilog 5% 1:1
"homework5.v" 17L, 770C written
...aptop: ~/Μαθήματα/Οργάνωση ΗΥ/2020-2021/Οργάνωση ΗΥ (Εργαστήριο)/lab1/homework5
1 include "half_adder.v"
2 module full_adder(output wire Sum, Cout, input wire A, B, Cin);
3 wire ha_1_sum, ha_1_cout, ha_2_cout;
4
5 half_adder U0(.Sum(ha_1_sum), .Cout(ha_1_cout), .A(A), .B(B));
6 half_adder U1(.Sum(Sum), .Cout(ha_2_cout), .A(ha_1_sum), .B(Cin));
7 or U2(Cout, ha_2_cout, ha_1_cout);
8 endmodule
NORMAL full_adder.v unix | utf-8 | verilog 50% 4:0
"full_adder.v" 8L, 321C written
...aptop: ~/Μαθήματα/Οργάνωση ΗΥ/2020-2021/Οργάνωση ΗΥ (Εργαστήριο)/lab1/homework5
1 module half_adder(output wire Sum, Cout, input wire A,B);
2 xor U0 (Sum, A, B);
3 and U1 (Cout, A, B);
4 endmodule
NORMAL half_adder.v unix | utf-8 | verilog 25% 1:1
```

```
$ iverilog -Wall -o homework5_tb.vvp homework5_tb.v
```

```
$ vvp homework5_tb.vvp
```

```
$ ls
```

```
full_adder.v
```

```
half_adder.v
```

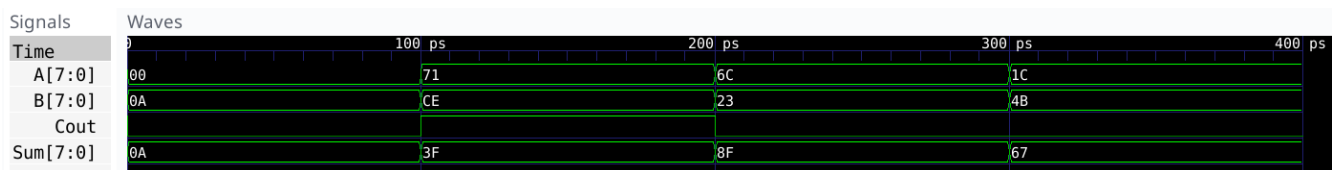
```
homework5_tb.v
```

```
homework5_tb.vcd
```

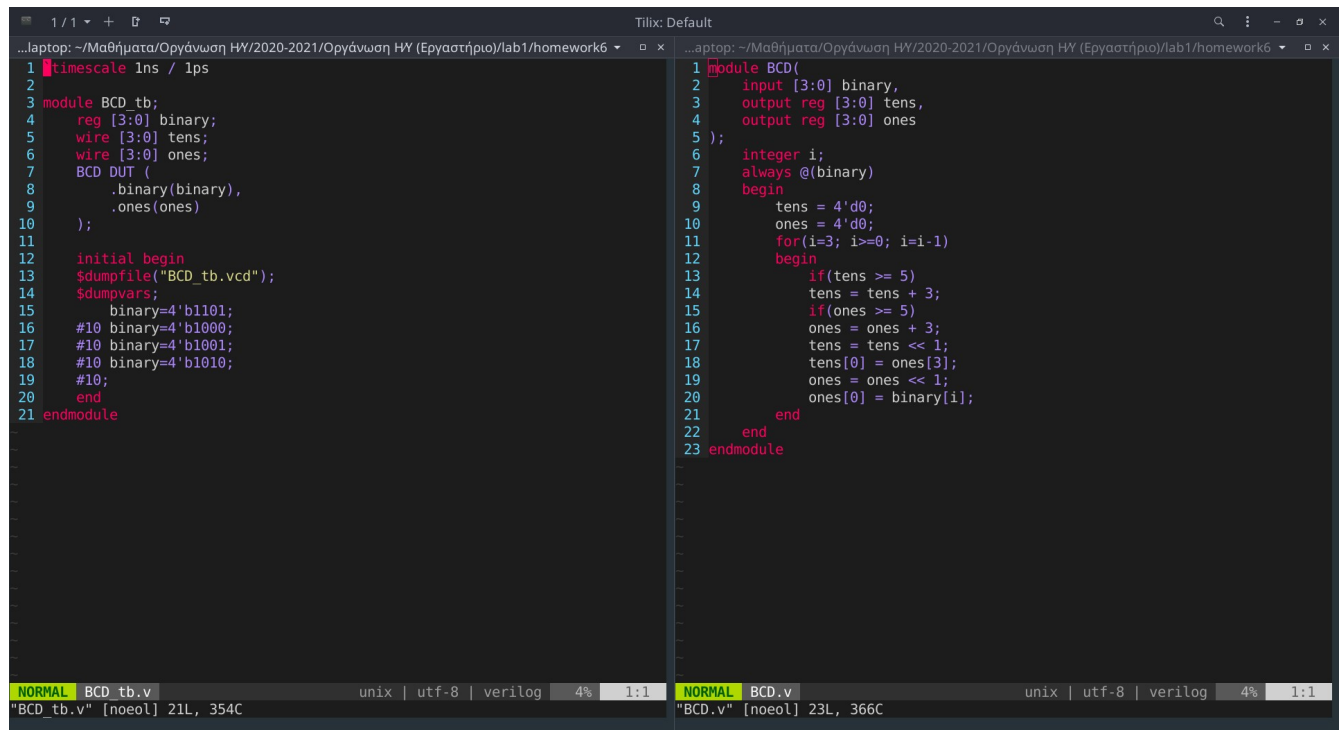
```
homework5_tb.vvp
```

```
homework5.v
```

```
$ gtkwave homework5_tb.vcd
```



Άσκηση 6

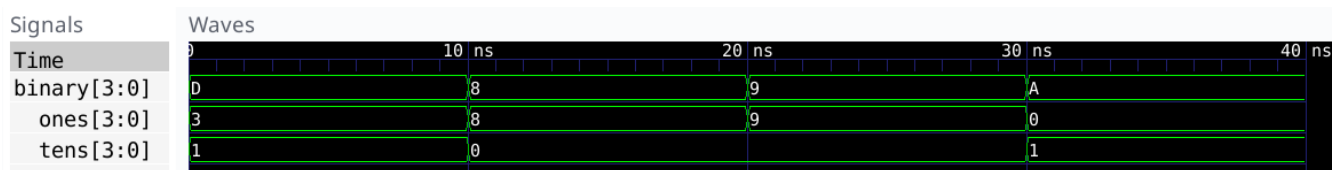


```
1 timescale 1ns / 1ps
2
3 module BCD_tb;
4     reg [3:0] binary;
5     wire [3:0] tens;
6     wire [3:0] ones;
7     BCD DUT (
8         .binary(binary),
9         .ones(ones)
10    );
11
12    initial begin
13        $dumpfile("BCD_tb.vcd");
14        $dumpvars;
15        binary=4'b1101;
16        #10 binary=4'b1000;
17        #10 binary=4'b1001;
18        #10 binary=4'b1010;
19        #10;
20    end
21 endmodule

1 module BCD(
2     input [3:0] binary,
3     output reg [3:0] tens,
4     output reg [3:0] ones
5 );
6     integer i;
7     always @(binary)
8     begin
9         tens = 4'd0;
10        ones = 4'd0;
11        for(i=3; i>=0; i=i-1)
12        begin
13            if(tens >= 5)
14                tens = tens + 3;
15            if(ones >= 5)
16                ones = ones + 3;
17            tens = tens << 1;
18            tens[0] = ones[3];
19            ones = ones << 1;
20            ones[0] = binary[i];
21        end
22    end
23 endmodule
```

```
$ iverilog -Wall -o BCD_tb.vvp BCD_tb.v BCD.v
$ vvp BCD_tb.vvp
$ ls
BCD_tb.v BCD_tb.vcd BCD_tb.vvp BCD.v
```

```
$ gtkwave BCD_tb.vcd
```



Άσκηση 7

a

```
1 //testbench for homework7
2 `timescale 1ns / 1ps
3
4 module homework7_tb;
5     reg A;
6     reg B;
7     reg C;
8     reg D;
9     wire Y;
10    reg [1:0] sel;
11
12    homework7 DUT(
13        .A(A),
14        .B(B),
15        .C(C),
16        .D(D),
17        .sel(sel),
18        .Y(Y)
19    );
20
21    initial begin
22        $dumpfile("homework7_tb.vcd");
23        $dumpvars;
24        A=1'b1; B=1'b0; C=1'b1; D=1'b0; sel=2'b00;
25        #0.1 A=1'b1; B=1'b0; C=1'b1; D=1'b0; sel=2'b10;
26        #0.1 A=1'b1; B=1'b0; C=1'b1; D=1'b0; sel=2'b01;
27        #0.1 A=1'b1; B=1'b0; C=1'b1; D=1'b0; sel=2'b11;
28        #0.1;
29    end
30
31    initial begin
32        $monitor("SELECT=%d,A=%d,B=%d,C=%d,D=%d,Y=%d \n",sel,A,B,C,D,Y);
33    end
34 endmodule
```

```
1 // Code your design here
2 `timescale 1ns / 1ps
3
4 module homework7(
5     output Y,
6     input A,B,C,D,
7     input [1:0] sel
8 );
9
10 assign Y = (sel == 2'b00) ? A :
11            (sel == 2'b10) ? B :
12            (sel == 2'b01) ? C :
13            (sel == 2'b11) ? D :
14            1'bX;
15 endmodule
```

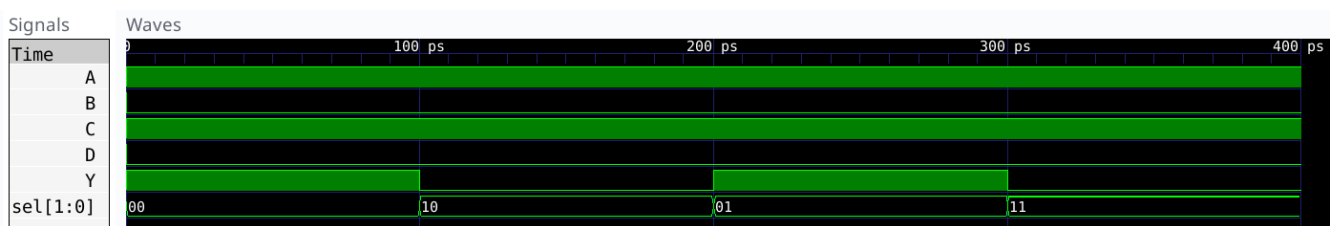
```
$ iverilog -Wall -o homework7_tb.vvp homework7_tb.v homework7.v
$ ls
homework7_tb.v homework7_tb.vcd homework7_tb.vvp homework7.v
$ vvp homework7_tb.vvp
VCD info: dumpfile homework7_tb.vcd opened for output.
SELECT=0,A=1,B=0,C=1,D=0,Y=1
```

SELECT=2,A=1,B=0,C=1,D=0,Y=0

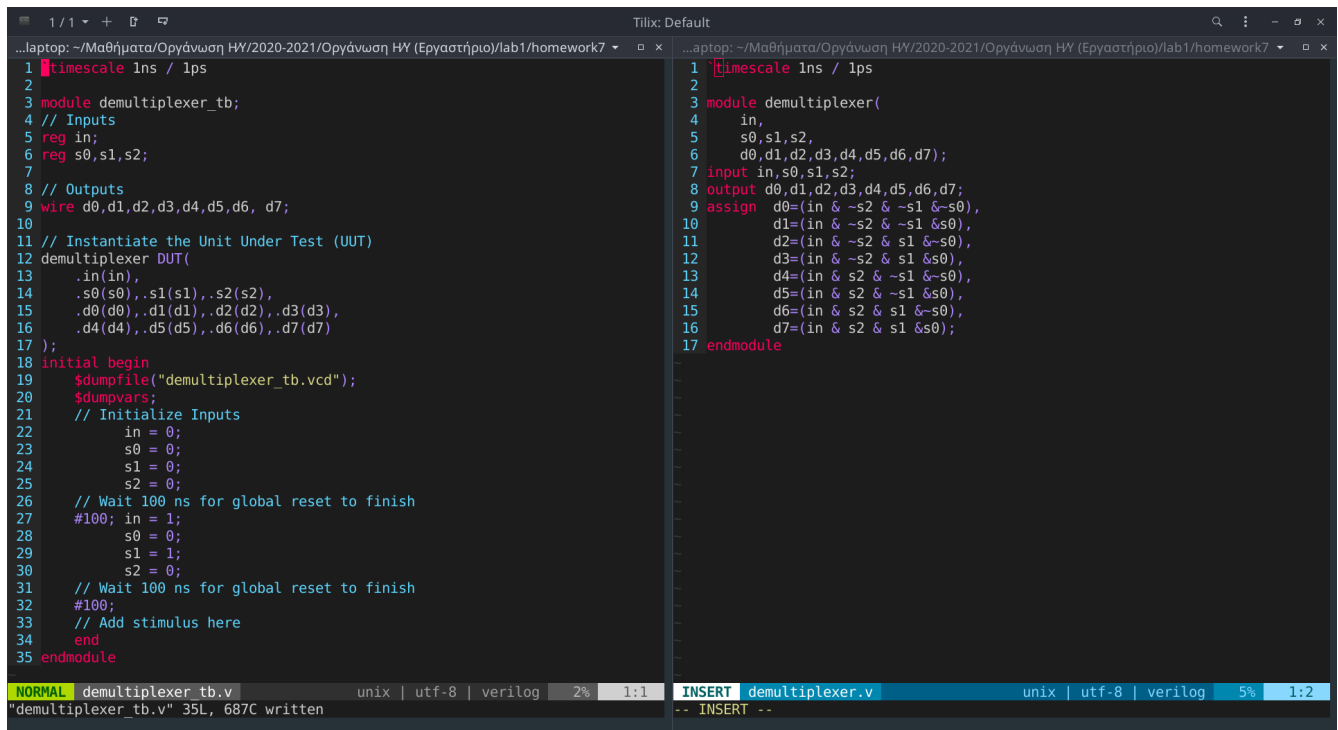
SELECT=1,A=1,B=0,C=1,D=0,Y=1

SELECT=3,A=1,B=0,C=1,D=0,Y=0

```
$ gtkwave homework7_tb.vcd
```



b



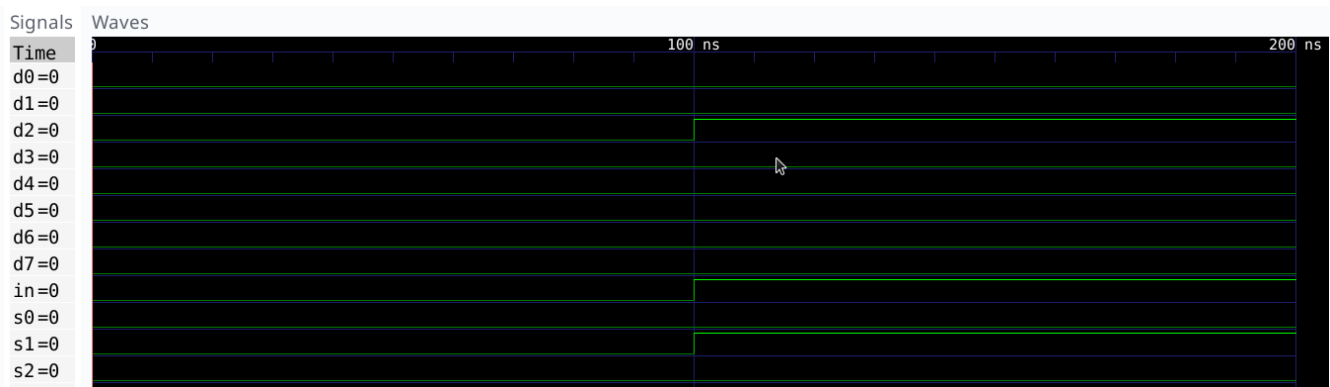
The screenshot shows a Verilog code editor with two files open. The left file, `demultiplexer_tb.v`, is a testbench. It includes a `timescale` of 1ns / 1ps, defines inputs `in`, `s0`, `s1`, and `s2`, and outputs `d0` through `d7`. It instantiates a `demultiplexer` module and initializes inputs to 0. It then sets `in` to 1 and `s1` to 1 for 100ns, followed by adding a stimulus. The right file, `demultiplexer.v`, is the module definition. It takes `in`, `s0`, `s1`, and `s2` as inputs and outputs `d0` through `d7`. The module uses `assign` statements to implement a 3-to-8 demultiplexer logic.

```
1 timescale 1ns / 1ps
2
3 module demultiplexer_tb;
4 // Inputs
5 reg in;
6 reg s0,s1,s2;
7
8 // Outputs
9 wire d0,d1,d2,d3,d4,d5,d6, d7;
10
11 // Instantiate the Unit Under Test (UUT)
12 demultiplexer DUT(
13     .in(in),
14     .s0(s0),.s1(s1),.s2(s2),
15     .d0(d0),.d1(d1),.d2(d2),.d3(d3),
16     .d4(d4),.d5(d5),.d6(d6),.d7(d7)
17 );
18 initial begin
19     $dumpfile("demultiplexer_tb.vcd");
20     $dumpvars;
21     // Initialize Inputs
22     in = 0;
23     s0 = 0;
24     s1 = 0;
25     s2 = 0;
26     // Wait 100 ns for global reset to finish
27     #100; in = 1;
28     s0 = 0;
29     s1 = 1;
30     s2 = 0;
31     // Wait 100 ns for global reset to finish
32     #100;
33     // Add stimulus here
34     end
35 endmodule
```

```
1 timescale 1ns / 1ps
2
3 module demultiplexer(
4     in,
5     s0,s1,s2,
6     d0,d1,d2,d3,d4,d5,d6,d7);
7 input in,s0,s1,s2;
8 output d0,d1,d2,d3,d4,d5,d6,d7;
9 assign d0=(in & ~s2 & ~s1 & ~s0),
10        d1=(in & ~s2 & ~s1 & s0),
11        d2=(in & ~s2 & s1 & ~s0),
12        d3=(in & ~s2 & s1 & s0),
13        d4=(in & s2 & ~s1 & ~s0),
14        d5=(in & s2 & ~s1 & s0),
15        d6=(in & s2 & s1 & ~s0),
16        d7=(in & s2 & s1 & s0);
17 endmodule
```

```
$ iverilog -Wall -o demultiplexer_tb.vvp demultiplexer_tb.v demultiplexer.v
$ vvp demultiplexer_tb.vvp
$ ls
demultiplexer_tb.v demultiplexer_tb.vcd demultiplexer_tb.vvp demultiplexer.v
```

```
$ gtkwave demultiplexer_tb.vcd
```



Άσκηση 8

a

```
1 //testbench for homework8_a
2 `timescale 1ns / 1ps
3 `include "toplevel_hw_8_a.v"
4
5 module homework8_a_tb;
6     reg pla,p1b,p1c,p1d,p2a,p2b,p2c,p2d;
7     wire ply,p2y;
8
9     toplevel_hw_8_a DUT(
10         .pla(pla),.p1b(p1b),.p1c(p1c),.p1d(p1d),
11         .p2a(p2a),.p2b(p2b),.p2c(p2c),.p2d(p2d),
12         .ply(ply),.p2y(p2y)
13     );
14
15     initial begin
16         $dumpfile("homework8_a_tb.vcd");
17         $dumpvars;
18         pla=1'b1; p1b=1'b1; p1c=1'b1; p1d=1'b1;
19         p2a=1'b0; p2b=1'b0; p2c=1'b0; p2d=1'b1;
20         #0.1 pla=1'b1; p1b=1'b1; p1c=1'b0; p1d=1'b0;
21         p2a=1'b0; p2b=1'b1; p2c=1'b1; p2d=1'b1;
22         #0.1 pla=1'b1; p1b=1'b1; p1c=1'b0; p1d=1'b0;
23         p2a=1'b0; p2b=1'b1; p2c=1'b1; p2d=1'b0;
24         #0.1 pla=1'b1; p1b=1'b1; p1c=1'b1; p1d=1'b0;
25         p2a=1'b1; p2b=1'b1; p2c=1'b1; p2d=1'b0;
26         #0.1;
27     end
28 endmodule
```

```
1 `include "submodule_hw_8_a.v"
2
3 module toplevel_hw_8_a(
4     output wire ply,p2y,
5     input wire pla,p1b,p1c,p1d,p2a,p2b,p2c,p2d
6 );
7
8 submodule_hw_8_a U0 (ply,pla,p1b,p1c,p1d);
9 submodule_hw_8_a U1 (p2y,p2a,p2b,p2c,p2d);
10
11 endmodule
```

```
1 module submodule_hw_8_a(
2     output wire y,
3     input wire a,b,c,d
4 );
5
6 nand U0 (y, a, b, c, d);
7 endmodule
8
```

```
$ iverilog -Wall -o homework8_a_tb.vvp homework8_a_tb.v
```

```
$ vvp homework8_a_tb.vvp
```

```
$ ls
```

```
homework8_a_tb.v
```

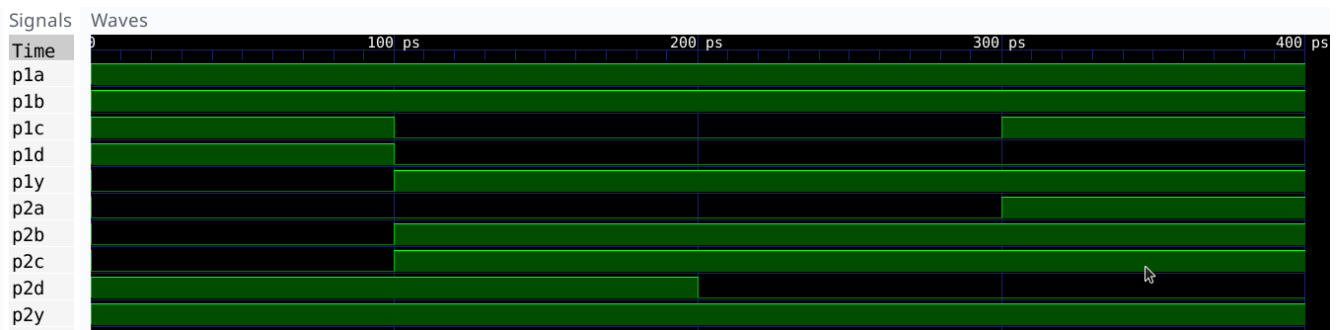
```
homework8_a_tb.vcd
```

```
homework8_a_tb.vvp
```

```
submodule_hw_8_a.v
```

```
toplevel_hw_8_a.v
```

```
$ gtkwave homework8_a_tb.vcd
```



b

```
1 timescale 1ns / 1ns
2 include "toplevel_hw_8_b.v"
3
4 module homework_b_tb;
5 reg S0, S1, S2, D0, D1, D2, D3, D4, D5, D6, D7;
6 wire 0;
7
8   toplevel_hw_8_b DUT(
9     .S0(S0), .S1(S1), .S2(S2),
10    .D0(D0), .D1(D1), .D2(D2), .D3(D3),
11    .D4(D4), .D5(D5), .D6(D6), .D7(D7),
12    .0(0)
13  );
14
15  initial begin
16    $dumpfile("homework_b_tb.vcd");
17    $dumpvars;
18    S0=1'b0; S1=1'b0; S2=1'b0;
19    D0=1'b0; D1=1'b0; D2=1'b0; D3=1'b0;
20    D4=1'b0; D5=1'b0; D6=1'b0; D7=1'b0;
21    #10 S0=1'b0; S1=1'b0; S2=1'b0;
22    D0=1'b1; D1=1'b1; D2=1'b1; D3=1'b1;
23    D4=1'b1; D5=1'b1; D6=1'b1; D7=1'b1;
24    #10;
25  end
26 endmodule
```

```
1 include "submodule_hw_8_b_or.v"
2 include "submodule_hw_8_b_not.v"
3 include "submodule_hw_8_b_and.v"
4
5 module toplevel_hw_8_b(
6   output wire 0,
7   input wire S0, S1, S2, D0, D1, D2, D3, D4, D5, D6, D7
8 );
9 wire NOTS0,NOTS1,NOTS2,AND0,AND1,AND2,AND3,AND4,AND5,AND6,AND7;
10
11 submodule_hw_8_b_not U0 (NOTS0, S0);
12 submodule_hw_8_b_not U1 (NOTS1, S1);
13 submodule_hw_8_b_not U2 (NOTS2, S2);
14 submodule_hw_8_b_and U3 (AND0, NOTS0, NOTS1, NOTS2, D0);
15 submodule_hw_8_b_and U4 (AND1, S0, NOTS1, NOTS2, D1);
16 submodule_hw_8_b_and U5 (AND2, NOTS0, S1, NOTS2, D2);
17 submodule_hw_8_b_and U6 (AND3, S0, S1, NOTS2, D3);
18 submodule_hw_8_b_and U7 (AND4, NOTS0, NOTS1, S2, D4);
19 submodule_hw_8_b_and U8 (AND5, S0, NOTS1, S2, D5);
20 submodule_hw_8_b_and U9 (AND6, NOTS0, S1, S2, D6);
21 submodule_hw_8_b_and U10 (AND7, S0, S1, S2, D7);
22 submodule_hw_8_b_or U11 (0, AND0,AND1,AND2,AND3,AND4,AND5,AND6,AND7);
23 endmodule
```

NORMAL toplevel_hw_8_b.v unix | utf-8 | verilog 4% 1:1

```
1 module submodule_hw_8_b_or(
2   output wire F1,
3   input wire A, B, C, D, E, F, G, H
4 );
5 or U0 (F1, A, B, C, D, E, F, G, H);
6 endmodule
```

NORMAL submodule_hw_8_b_or.v unix | utf-8 | verilog 16% 1:1

"homework8_b_tb.v" [noeol] 26L, 694C

"submodule_hw_8_b_or.v" [noeol] 6L, 139C

```
1 module submodule_hw_8_b_not(
2   output wire F1,
3   input wire A
4 );
5 not U0 (F1, A);
6 endmodule
```

```
1 module submodule_hw_8_b_and(
2   output wire F1,
3   input wire A, B, C, D
4 );
5 and U0 (F1, A, B, C, D);
6 endmodule
```

NORMAL submodule_hw_8_b_not.v unix | utf-8 | verilog 16% 1:1

NORMAL submodule_hw_8_b_and.v unix | utf-8 | verilog 16% 1:1

```
$ iverilog -Wall -o homework8_b_tb.vvp homework8_b_tb.v
```

```
$ vvp homework8_b_tb.vvp
```

```
$ ls
```

```
homework8_b_tb.v
```

```
homework8_b_tb.vvp
```

```
homework_b_tb.vcd
```

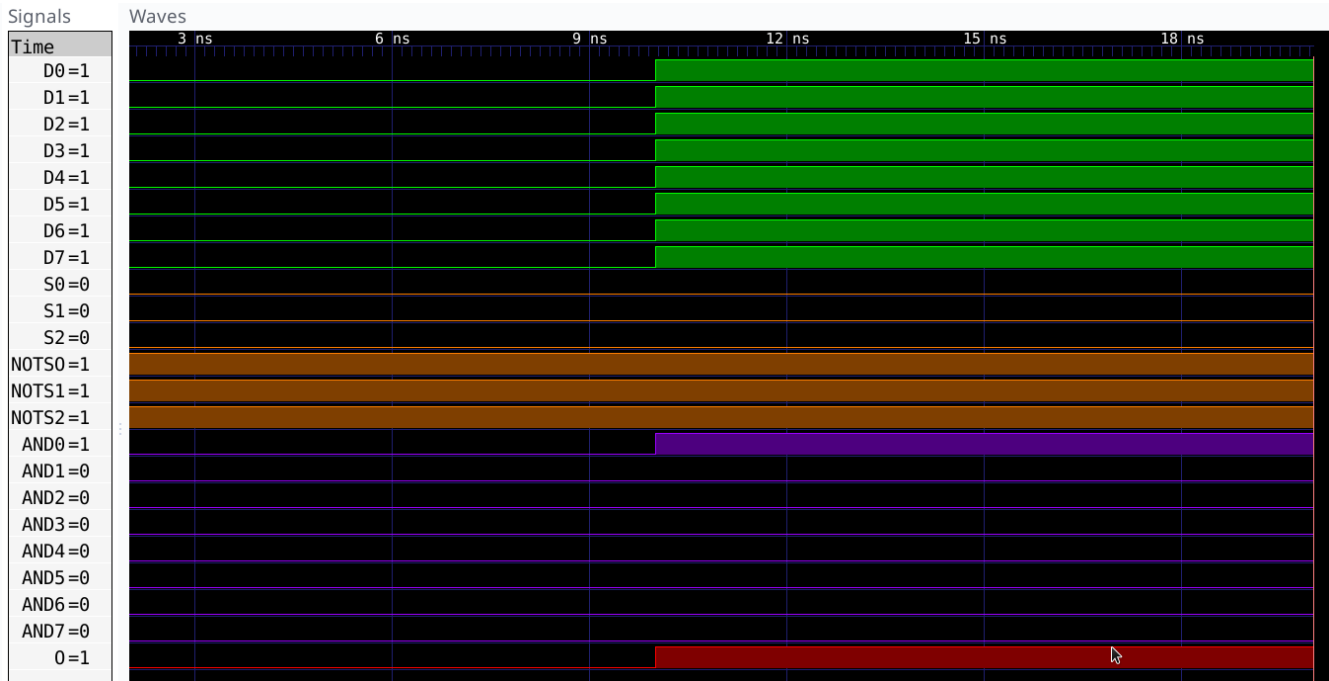
```
submodule_hw_8_b_and.v
```

```
submodule_hw_8_b_not.v
```

```
submodule_hw_8_b_or.v
```

```
toplevel_hw_8_b.v
```

```
$ gtkwave homework8_b_tb.vcd
```



C

```
1 //testbench for homework1
2 `timescale 1ns / 1ps
3 `include "toplevel_hw_8_c.v"
4
5 module homework8_c_tb;
6     reg X;
7     reg Y;
8     reg Z;
9     wire F;
10
11     toplevel_hw_8_c DUT(
12         .X(X),
13         .Y(Y),
14         .Z(Z),
15         .F(F)
16     );
17
18     initial begin
19         $dumpfile("homework8_c_tb.vcd");
20         $dumpvars;
21         X=1'b0; Y=1'b0; Z=1'b0;
22         #10 X=1'b0; Y=1'b0; Z=1'b1;
23         #10 X=1'b0; Y=1'b1; Z=1'b0;
24         #10 X=1'b0; Y=1'b1; Z=1'b1;
25         #10 X=1'b1; Y=1'b0; Z=1'b0;
26         #10 X=1'b1; Y=1'b0; Z=1'b1;
27         #10 X=1'b1; Y=1'b1; Z=1'b0;
28         #10 X=1'b1; Y=1'b1; Z=1'b1;
29         #10;
30     end
31
32     initial begin
33         $monitor("X=%d,Y=%d,Z=%d,F=%d\n",X,Y,Z,F);
34     end
35 endmodule
```

```
1 include "submodule_hw_8_c_or.v"
2 include "submodule_hw_8_c_not.v"
3 include "submodule_hw_8_c_xor.v"
4
5 module toplevel_hw_8_c(
6     output wire F,
7     input wire X,Y,Z
8 );
9
10 wire OR1,NOTX,OR2,XOR1;
11 submodule_hw_8_c_or U0 (OR1,X,Y);
12 submodule_hw_8_c_not U1 (NOTX,X);
13 submodule_hw_8_c_xor U2 (XOR1,Y,Z);
14 submodule_hw_8_c_or U3 (OR2,NOTX,XOR1);
15 submodule_hw_8_c_xor U4 (F,XOR1,OR2);
16 endmodule
```

```
1 module submodule_hw_8_c_not(
2     output wire F1,
3     input wire A
4 );
5 not U0 (F1, A);
6 endmodule
```

```
1 module submodule_hw_8_c_or(
2     output wire F1,
3     input wire A,B
4 );
5 or U0 (F1, A, B);
6 endmodule
```

```
1 module submodule_hw_8_c_xor(
2     output wire F1,
3     input wire A,B
4 );
5 xor U0 (F1, A, B);
6 endmodule
```

```
$ iverilog -Wall -o homework8_c_tb.vvp homework8_c_tb.v
$ vvp homework8_c_tb.vvp
$ ls
homework8_c_tb.v
homework8_c_tb.vcd
homework8_c_tb.vvp
submodule_hw_8_c_not.v
submodule_hw_8_c_or.v
submodule_hw_8_c_xor.v
toplevel_hw_8_c.v
```

\$ gtkwave homework8_c_tb.vcd

