Digital Design Laboratory

Dr. Oniga István
University of Debrecen, Faculty of Informatics

This work was supported by the construction EFOP-3.4.3-16-2016-00021.
The project was supported by the European Union, co-financed by the European Social Fund.

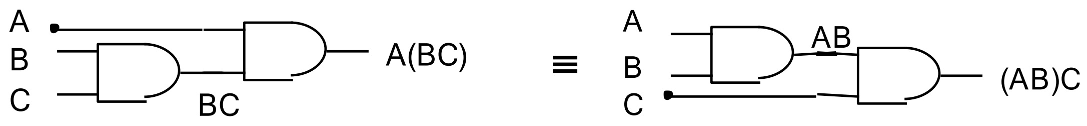
2. Laboratory assignments

- Boolean algebra
  - Associative rules
  - Distributive rules
  - Absorption rules
  - De Morgan rules
- Implementation of for variable functions AND, OR, XOR and NOR
- Simulation using test vectors

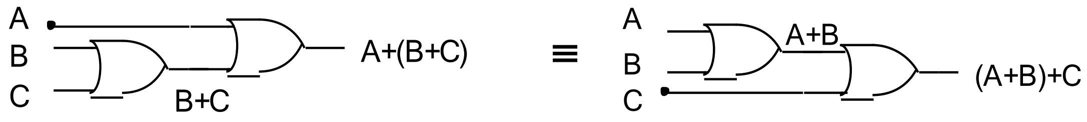
Lab2_1a assignment:

3 variable logic function implementation - Associative rules -

$$A \bullet (B \bullet C) = (A \bullet B) \bullet C$$



$$A + (B + C) = (A + B) + C$$



- Create a new project
- Add a new "schematic" source
- Draw the schematic presented on this slide.
- Add and adapt the constraints file Nexysx.UCF
  - Inputs: ***sw[2:0]***
    - sw0 -> A; sw1 -> B; sw2->C
  - Outputs: ***led[3:0]***
- Generate the configuration file, download to board, test the project
- Note your experience in the Laboratory's Report Questionnaire

Lab2_1a Results

- Using switches sw0, sw1 and sw2 make all 8 possible combinations and note the corresponding state of the LEDs on the following table

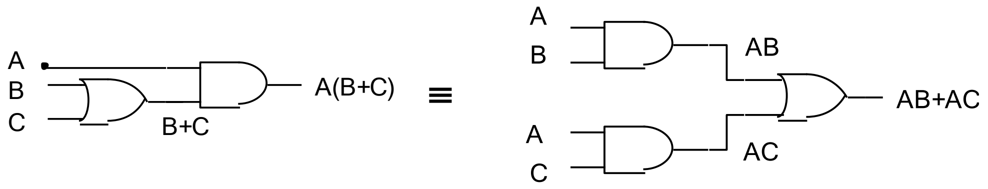
sw0	sw1	sw2	led1 A(BC)	led2 (AB)C	led3 A+(B+C)	led4 (A+B)+C
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Lab2_1b assignment:

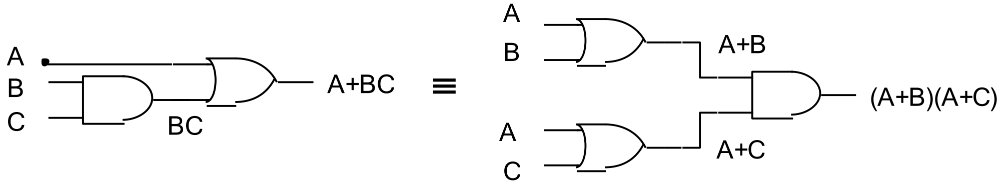
3 variable logic function implementation - Distributive rules -

- Create a new project
- Add a new "schematic" source
- Draw the schematic presented on this slide.
- Add and adapt the Nexysx.UCF file
  - Inputs: ***sw[2:0]***
    - sw0 -> A; sw1 -> B; sw2->C
  - Outputs: ***led[3:0]***

$$A\bullet(B+C)=A\bullet B+A\bullet C$$



$$A+B\bullet C=(A+B)\bullet(A+C)$$



- Generate the configuration file, download to board, test the project
- Note your experience in the Laboratory's Report Questionnaire

Lab2_1b Results

- Using switches sw0, sw1 and sw2 make all 8 possible combinations and note the corresponding state of the LEDs on the following table

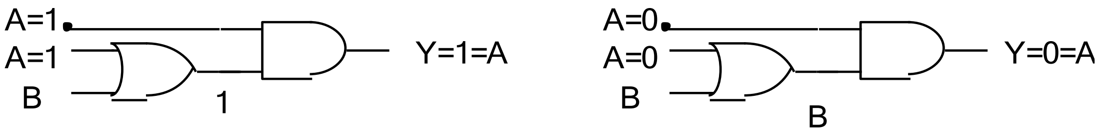
sw0	sw1	sw2	led1 A(B+C)	led2 AB+AC	led3 A+BC	led4 (A+B)*(A+C)
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Lab2_1c assignment:

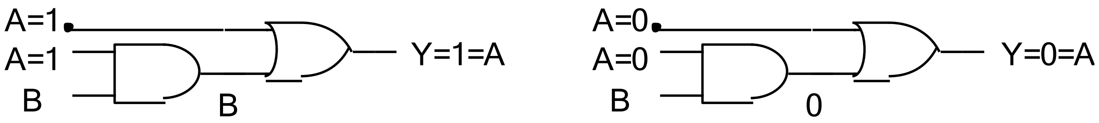
- Absorption rules -

- Create a new project
- Add a new "schematic" source
- Draw the schematic presented on this slide.
- Add and adapt the Nexysx.UCF file
  - Inputs: ***sw[1:0]***
    - sw0 -> A; sw1 -> B
  - Outputs: ***led[1:0]***

$$A \bullet (A + B) = A$$



$$A + A \bullet B = A$$



- Generate the configuration file, download to board, test the project
- Note your experience in the Laboratory's Report Questionnaire

Lab2_1c Results

- Using switches sw0 and sw1 make all 4 possible combinations and note the corresponding state of the leds on the following table

sw0	sw1	led1 A(A+B)	led2 A+AB
0	0		
0	1		
1	0		
1	1		

Lab2_2a assignment:

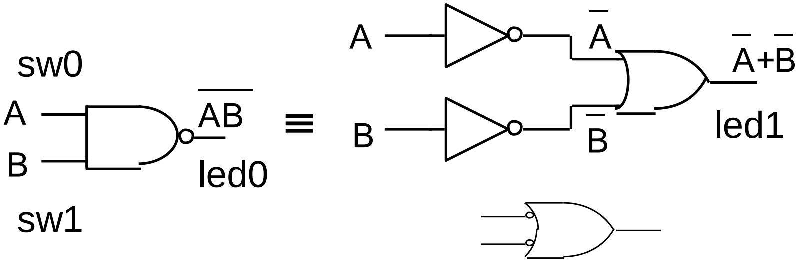
De Morgan rules for 2 variables

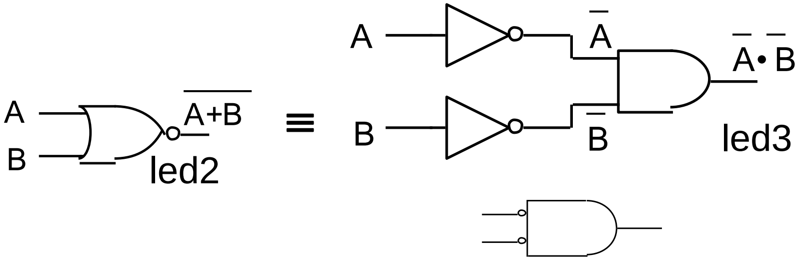
$$\overline{A \bullet B} = \overline{A} + \overline{B}$$

$$\overline{A + B} = \overline{A} \bullet \overline{B}$$

- Create a new project
- Add a new "schematic" source
- Draw the schematic presented on this slide.
- Add and adapt the Nexysx.UCF file
  - Inputs: sw[1:0]

    sw0 -> A
    sw1 -> B
  - Outputs: led[3:0]





- Generate the configuration file, download to board, test the project
- Note your experience in the Laboratory's Report Questionnaire

Lab2_2a Results

- Using switches sw0 and sw1 make all 4 possible combinations and note the corresponding state of the LEDs on the following table

sw0	sw1	led1 /(AB)	led2 /A+/B	led3 /(A+B)	led4 /A*/B
0	0				
0	1				
1	0				
0	1				

Lab2_2b assignment:

De Morgan rules for 3 variables

$$\overline{A\bullet B\bullet C} = \overline{A}+\overline{B}+\overline{C}$$

$$\overline{A+B+C} = \overline{A}\bullet\overline{B}\bullet\overline{C}$$

- Create a new project
- Add a new "schematic” source
- Draw the schematic.
- Add and adapt the Nexysx.UCF file
  - Inputs: ***sw[2:0]***

    sw0 -> A; sw1 -> B; sw2 -> C
  - Outputs: ***led[3:0]***

sw0	sw1	sw2	led1 /(ABC)	led2 /A+/B+/C	led3 /(A+B+C)	led4 /A*/B*/C
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

- Generate the configuration file, download to board, test the project
- Using switches sw0, sw1 and sw2 make all 8 possible combinations and note the corresponding state of the LEDs on the following table
- Note your experience in the Laboratory's Report Questionnaire

Lab2_2c (optional assignment):

De Morgan generalization

$$X = \overline{A \bullet B + A \bullet \overline{C} + ABC} = \overline{A \bullet B} \bullet \overline{A \bullet \overline{C}} \bullet \overline{ABC}$$

$$Y = \overline{(A \bullet B + A \bullet \overline{C}) \bullet (ABC + \overline{B}C)} = \overline{A \bullet B + A \bullet \overline{C}} + \overline{ABC + \overline{B}C}$$

- Create a new project
- Add a new "schematic” source
- Draw the schematic.
- Add and adapt the Nexysx.UCF file
  - inputs: ***sw[2:0]***

    sw0 -> A; sw1 -> B; sw2 -> C
  - Outputs: ***led[3:0]***

sw0	sw1	sw2	led1 X1	led2 X2	led3 Y1	led4 Y2
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

- Generate the configuration file, download to board, test the project
- Using switches sw0, sw1 and sw2 make all 8 possible combinations and note the corresponding state of the LEDs on the following table
- Note your experience in the Laboratory's Report Questionnaire

Lab2_3a assignment:

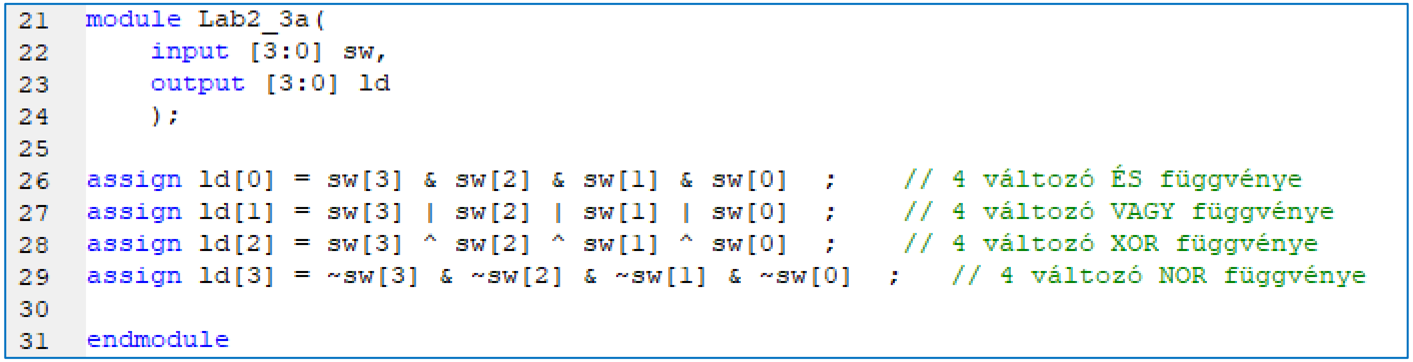
- **Implementation of AND, OR, XOR and NOR functions**
  - Inputs: DIP switches lower 4 bits
  - Outputs: lower 4 LEDs
- **Simulation generating the test vectors using “for” loop**

Follow the flow on the previous week flow presented in „DDL_1.pdf”

- Start ISE, create a new project
- Add a new Verilog file Lab2_3a.v
- Add copy of source: Nexys4.UCF file, adapt to actual inputs and outputs
- Edit Lab2_3a.v adding the needed functionality
- Functional simulation
- Generate configuration file, download to board, test.

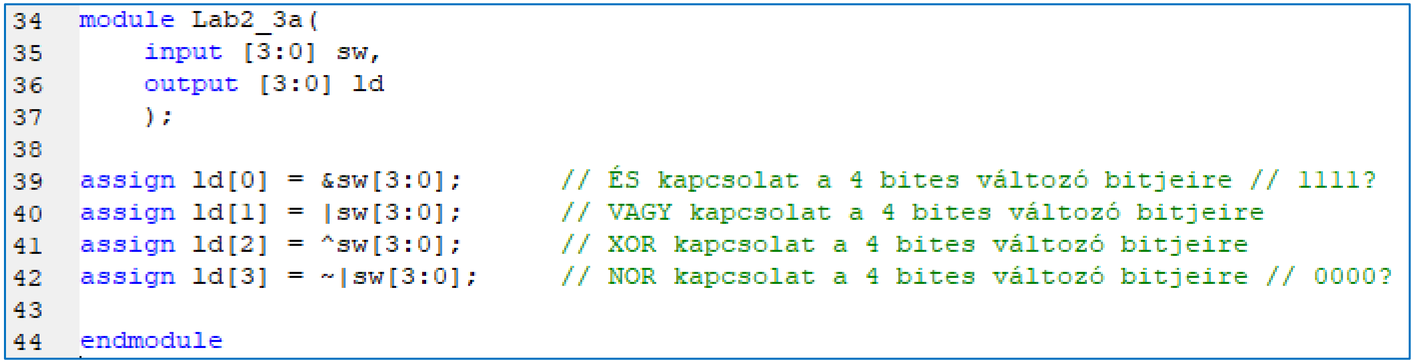
Lab2_3a assignment:

- **Input signals specification as individual bits**



```verilog
module Lab2_3a(
    input [3:0] sw,
    output [3:0] ld
    );

assign ld[0] = sw[3] & sw[2] & sw[1] & sw[0]  ;    // 4 változó ÉS függvénye
assign ld[1] = sw[3] | sw[2] | sw[1] | sw[0]  ;    // 4 változó VAGY függvénye
assign ld[2] = sw[3] ^ sw[2] ^ sw[1] ^ sw[0]  ;    // 4 változó XOR függvénye
assign ld[3] = ~sw[3] & ~sw[2] & ~sw[1] & ~sw[0]  ;   // 4 változó NOR függvénye

endmodule
```

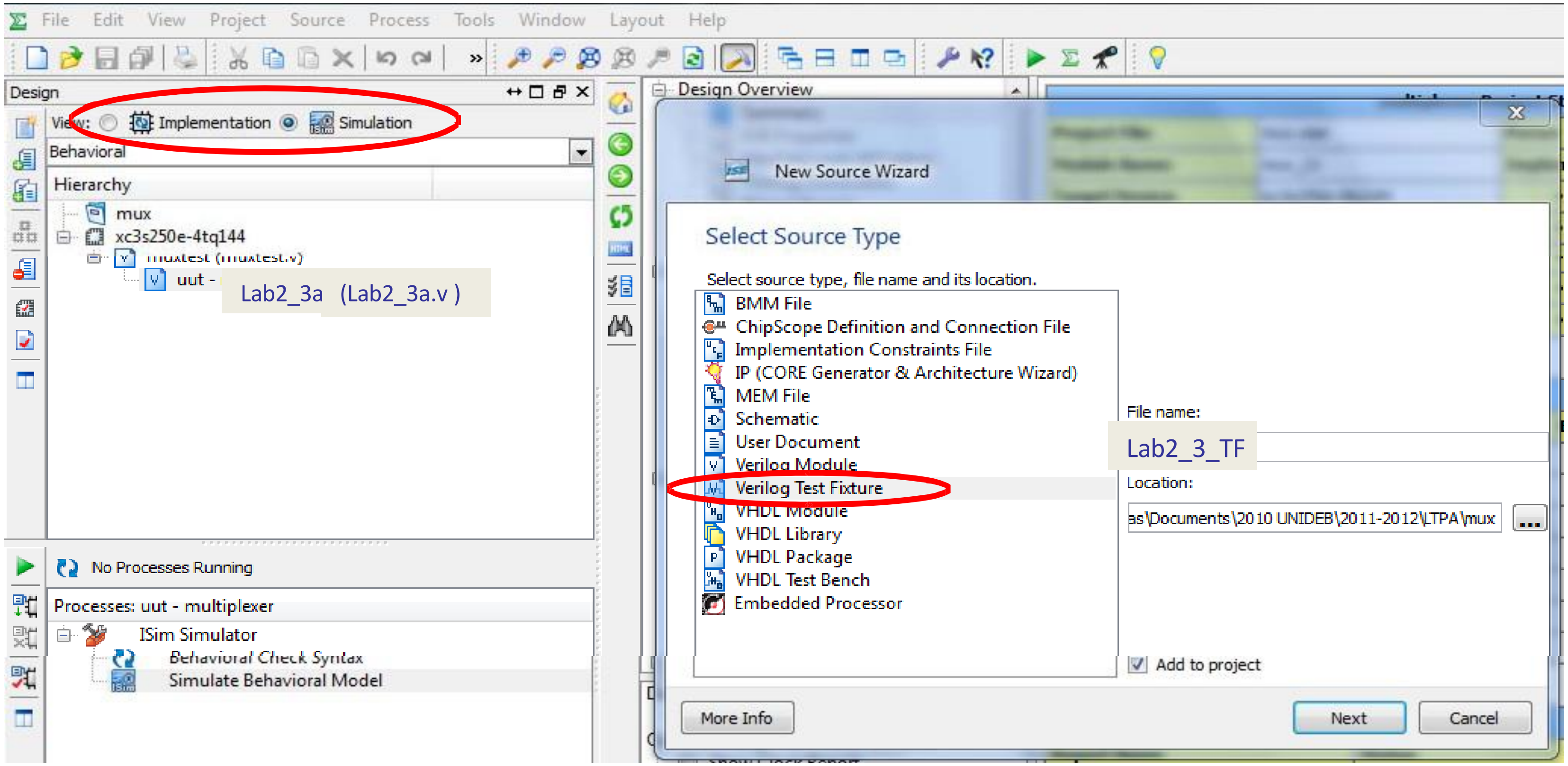
- **Using bit reduction operators on vectors**



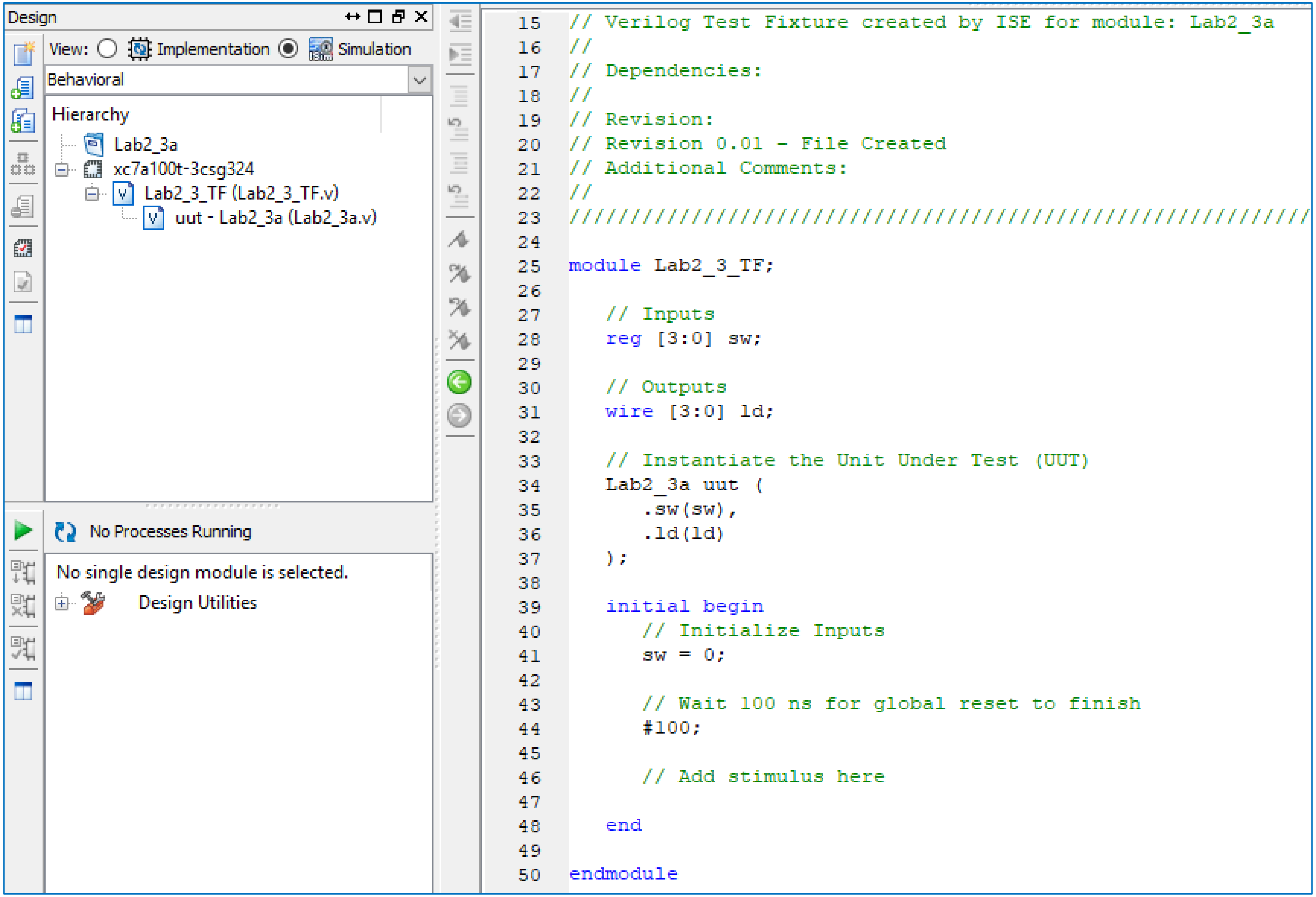
```verilog
module Lab2_3a(
    input [3:0] sw,
    output [3:0] ld
    );

assign ld[0] = &sw[3:0];      // ÉS kapcsolat a 4 bites változó bitjeire // 1111?
assign ld[1] = |sw[3:0];      // VAGY kapcsolat a 4 bites változó bitjeire
assign ld[2] = ^sw[3:0];      // XOR kapcsolat a 4 bites változó bitjeire
assign ld[3] = ~|sw[3:0];     // NOR kapcsolat a 4 bites változó bitjeire // 0000?

endmodule
```

Lab2_3a assignment: simulation

- Change to simulation Mode
- Creating the text fixture and specifying the text vectors
- Add a new source: Project / New Source - Verilog Test Fixture. The file name: Lab2_3_TF !
- Select the module to be tested.



Lab2_3a assignment: simulation

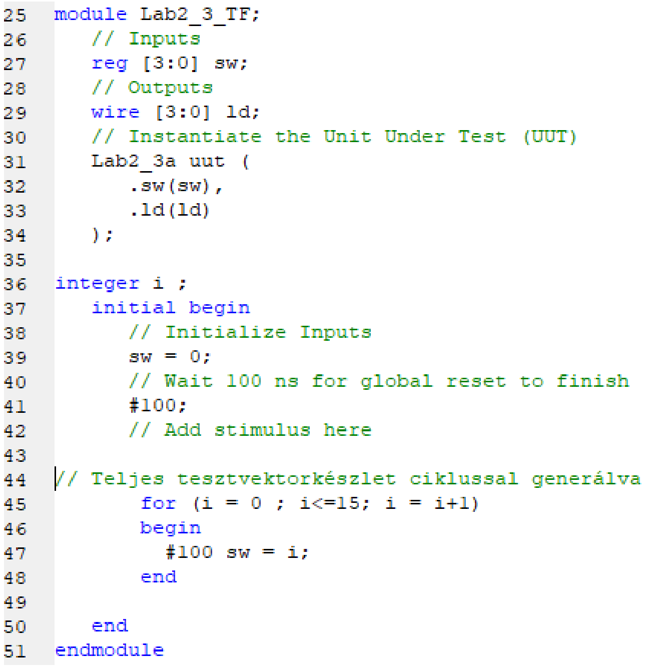


```
// Verilog Test Fixture created by ISE for module: Lab2_3a
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////

module Lab2_3_TF;

    // Inputs
    reg [3:0] sw;

    // Outputs
    wire [3:0] ld;

    // Instantiate the Unit Under Test (UUT)
    Lab2_3a uut (
        .sw(sw),
        .ld(ld)
    );

    initial begin
        // Initialize Inputs
        sw = 0;

        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here

    end

endmodule
```

Test vectors generation

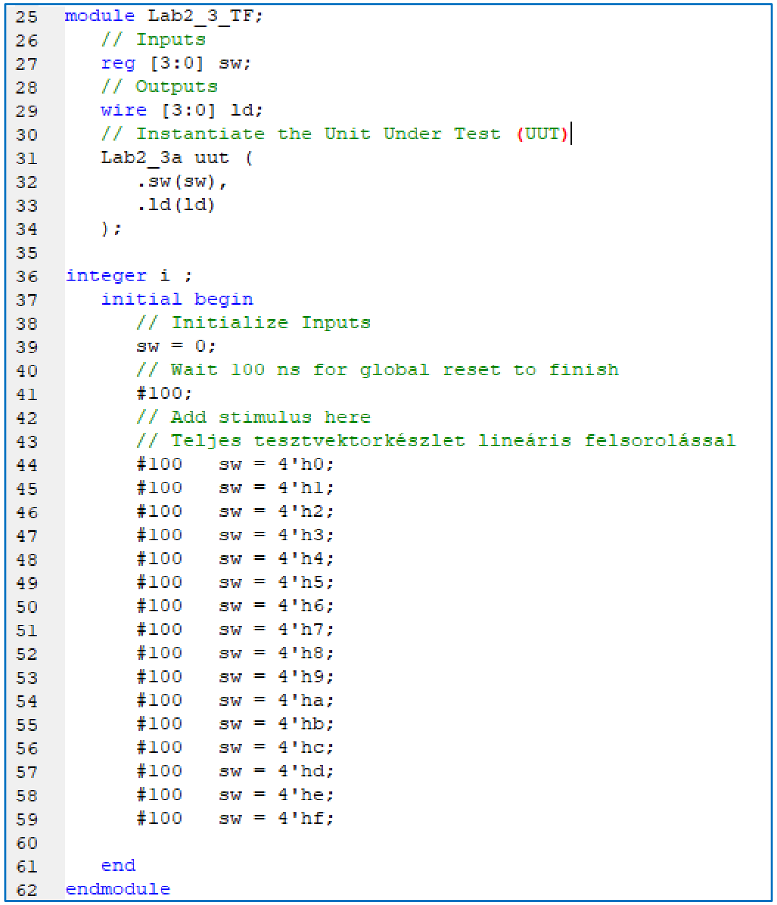
- Change the automatically generated Verilog Test Fixture file
- 4 variable function
  - Max. 16 combinations

Test vector generation using ***for** loop*



```
module Lab2_3_TF;
   // Inputs
   reg [3:0] sw;
   // Outputs
   wire [3:0] ld;
   // Instantiate the Unit Under Test (UUT)
   Lab2_3a uut (
      .sw(sw),
      .ld(ld)
   );

integer i ;
   initial begin
      // Initialize Inputs
      sw = 0;
      // Wait 100 ns for global reset to finish
      #100;
      // Add stimulus here

// Teljes tesztvektorkészlet ciklussal generálva
       for (i = 0 ; i<=15; i = i+1)
       begin
         #100 sw = i;
       end

   end
endmodule
```

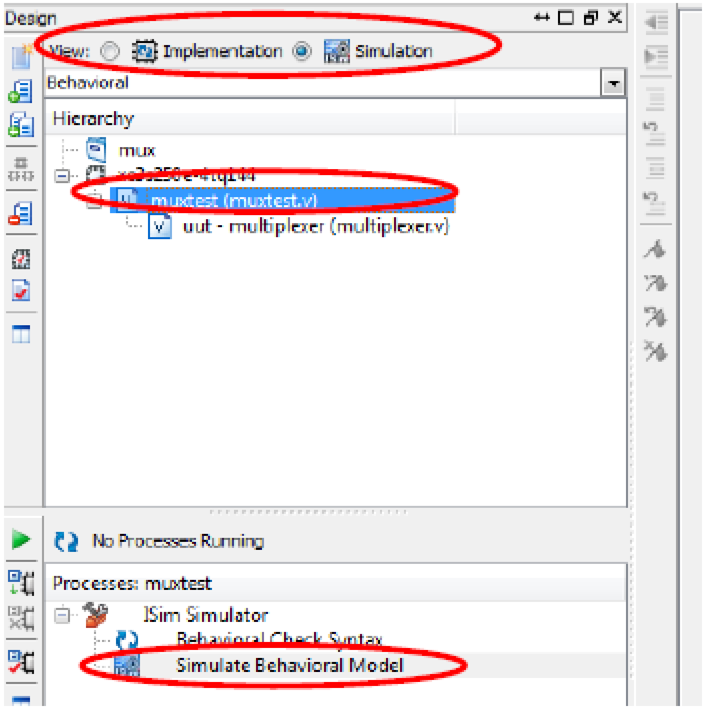
Test vector generation using linear code



```
module Lab2_3_TF;
   // Inputs
   reg [3:0] sw;
   // Outputs
   wire [3:0] ld;
   // Instantiate the Unit Under Test (UUT)
   Lab2_3a uut (
      .sw(sw),
      .ld(ld)
   );

integer i ;
   initial begin
      // Initialize Inputs
      sw = 0;
      // Wait 100 ns for global reset to finish
      #100;
      // Add stimulus here
      // Teljes tesztvektorkészlet lineáris felsorolással
      #100   sw = 4'h0;
      #100   sw = 4'h1;
      #100   sw = 4'h2;
      #100   sw = 4'h3;
      #100   sw = 4'h4;
      #100   sw = 4'h5;
      #100   sw = 4'h6;
      #100   sw = 4'h7;
      #100   sw = 4'h8;
      #100   sw = 4'h9;
      #100   sw = 4'ha;
      #100   sw = 4'hb;
      #100   sw = 4'hc;
      #100   sw = 4'hd;
      #100   sw = 4'he;
      #100   sw = 4'hf;

   end
endmodule
```

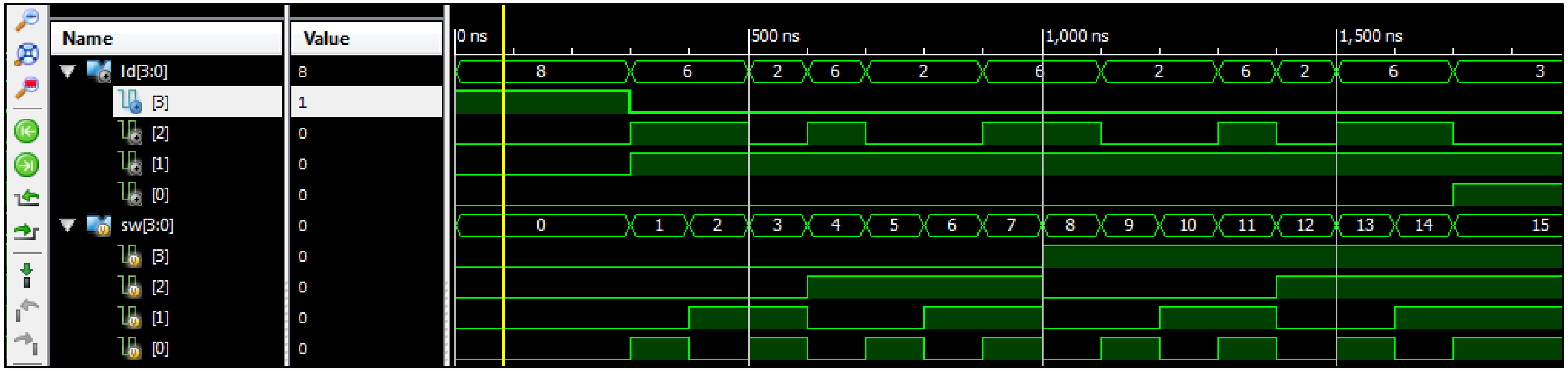
Simulation

- **Project Navigator program View: Simulation, in Hierarchy** select the testfixture (*Lab2_3_TF).*
- *In **Processes window choose ISim Simulator /*****Simulate Behavioral Model.**



Lab2_3a results

- **Simulation results**
  - LD[0] → AND, LD[1] → OR, LD[2] → XOR, LD[3] → NOR



- Design implemenation:
  - Generate .bit file
  - Download and test in board
- Note your experience in the Laboratory's Report Questionnaire