

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 -

- Create a new project
- Add a new "schematic" source
- Draw the schematic presented on this slide.

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



- Add and adapt the constraints file Nexysx.UCF

– Inputs: **sw[2:0]**

– sw0 -> A; sw1 -> B; sw2->C

– Outputs: **led[3:0]**

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



- 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

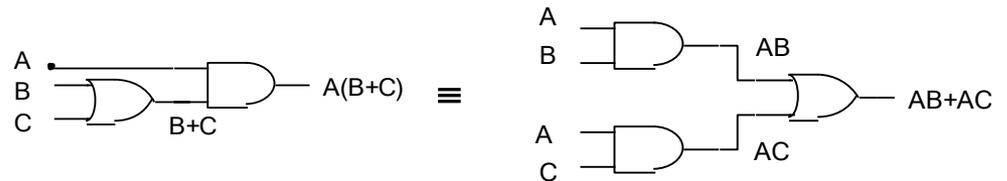
sw0	sw1	sw2	led1 $A(BC)$	led2 $(AB)C$	led3 $A+(B+C)$	led4 $(A+B)+C$
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Lab2_1b assignment:

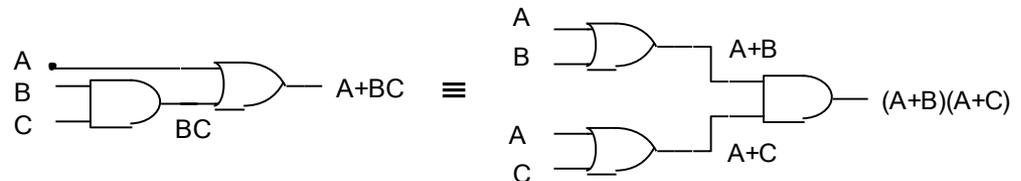
3 variable logic function implementation - Distributive rules -

- 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				

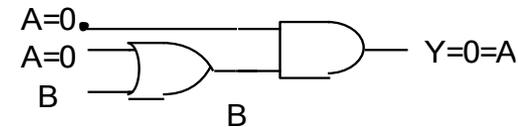
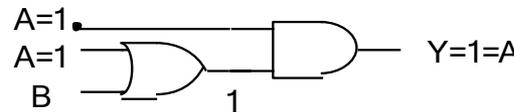
Lab2_1c assignment:

- Absorption rules -

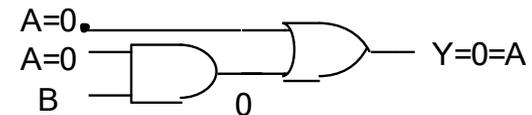
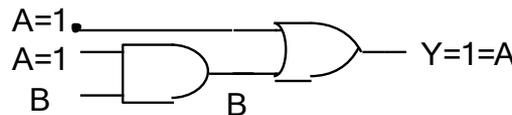
- Create a new project
- Add a new "schematic" source
- Draw the schematic presented on this slide.
- Add and adapt the Nexysx.UCF file

- 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

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

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

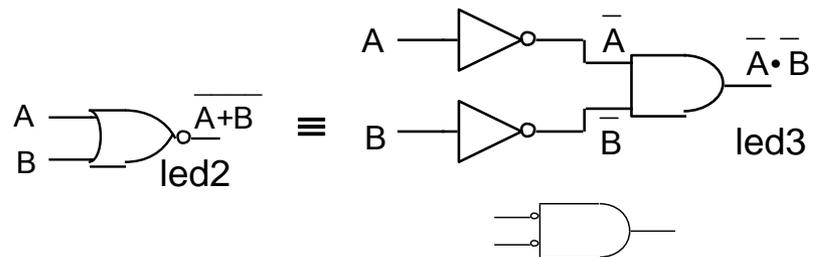
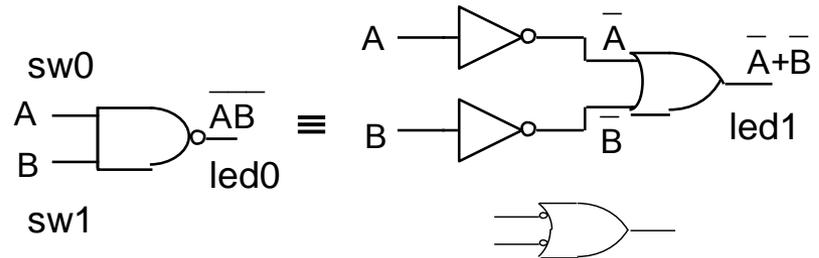
- Create a new project
- Add a new "schematic" source
- Draw the schematic presented on this slide.
- Add and adapt the Nexysx.UCF file

- 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 $\neg(A \cdot B)$	led2 $\neg A + \neg B$	led3 $\neg(A + B)$	led4 $\neg A * \neg B$
0	0				
0	1				
1	0				
0	1				

Lab2_2b assignment:

De Morgan rules for 3 variables

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

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

- 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

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

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

- 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]**

- 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

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				

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

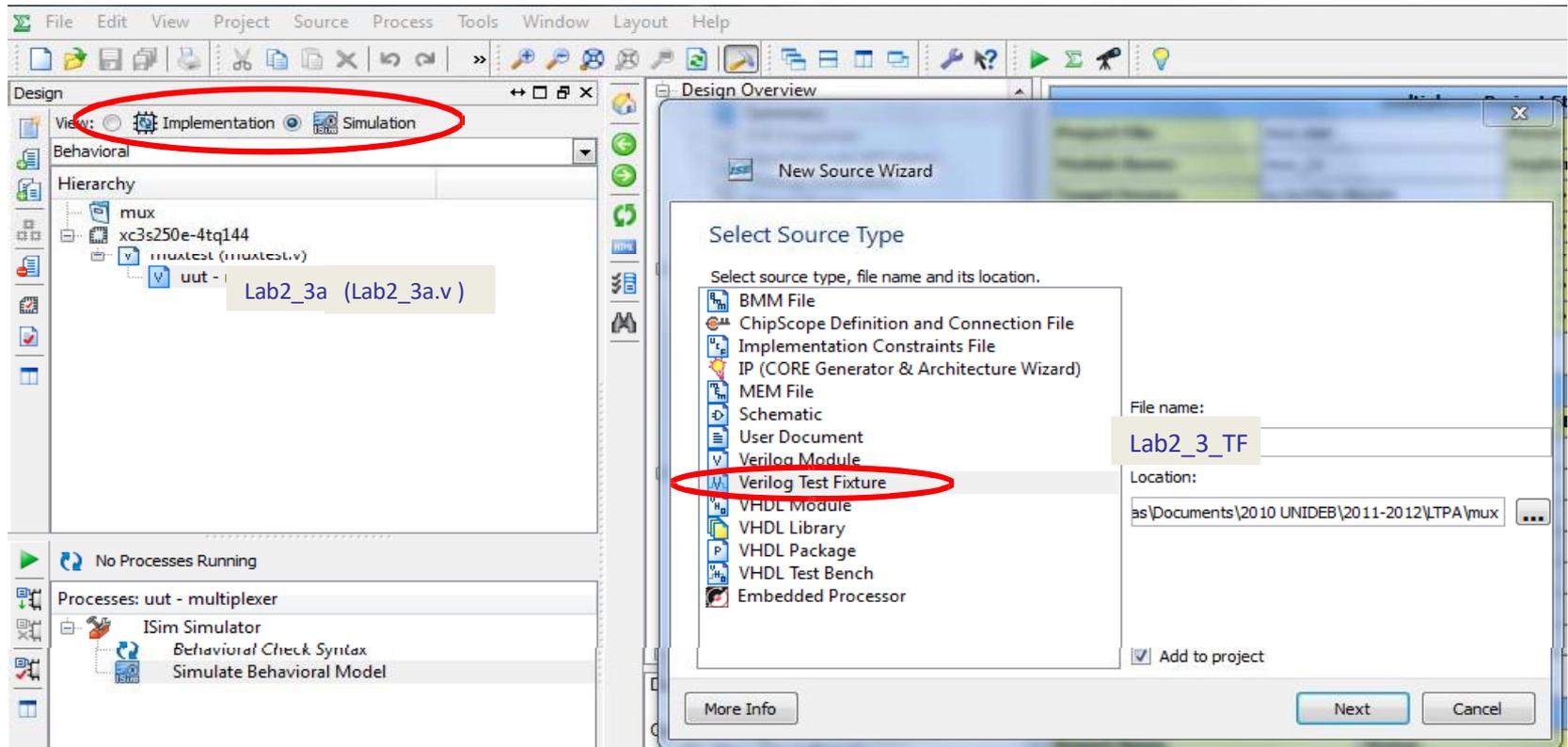
```
21 module Lab2_3a(  
22     input [3:0] sw,  
23     output [3:0] ld  
24 );  
25  
26 assign ld[0] = sw[3] & sw[2] & sw[1] & sw[0] ; // 4 változó ÉS függvénye  
27 assign ld[1] = sw[3] | sw[2] | sw[1] | sw[0] ; // 4 változó VAGY függvénye  
28 assign ld[2] = sw[3] ^ sw[2] ^ sw[1] ^ sw[0] ; // 4 változó XOR függvénye  
29 assign ld[3] = ~sw[3] & ~sw[2] & ~sw[1] & ~sw[0] ; // 4 változó NOR függvénye  
30  
31 endmodule
```

- Using bit reduction operators on vectors

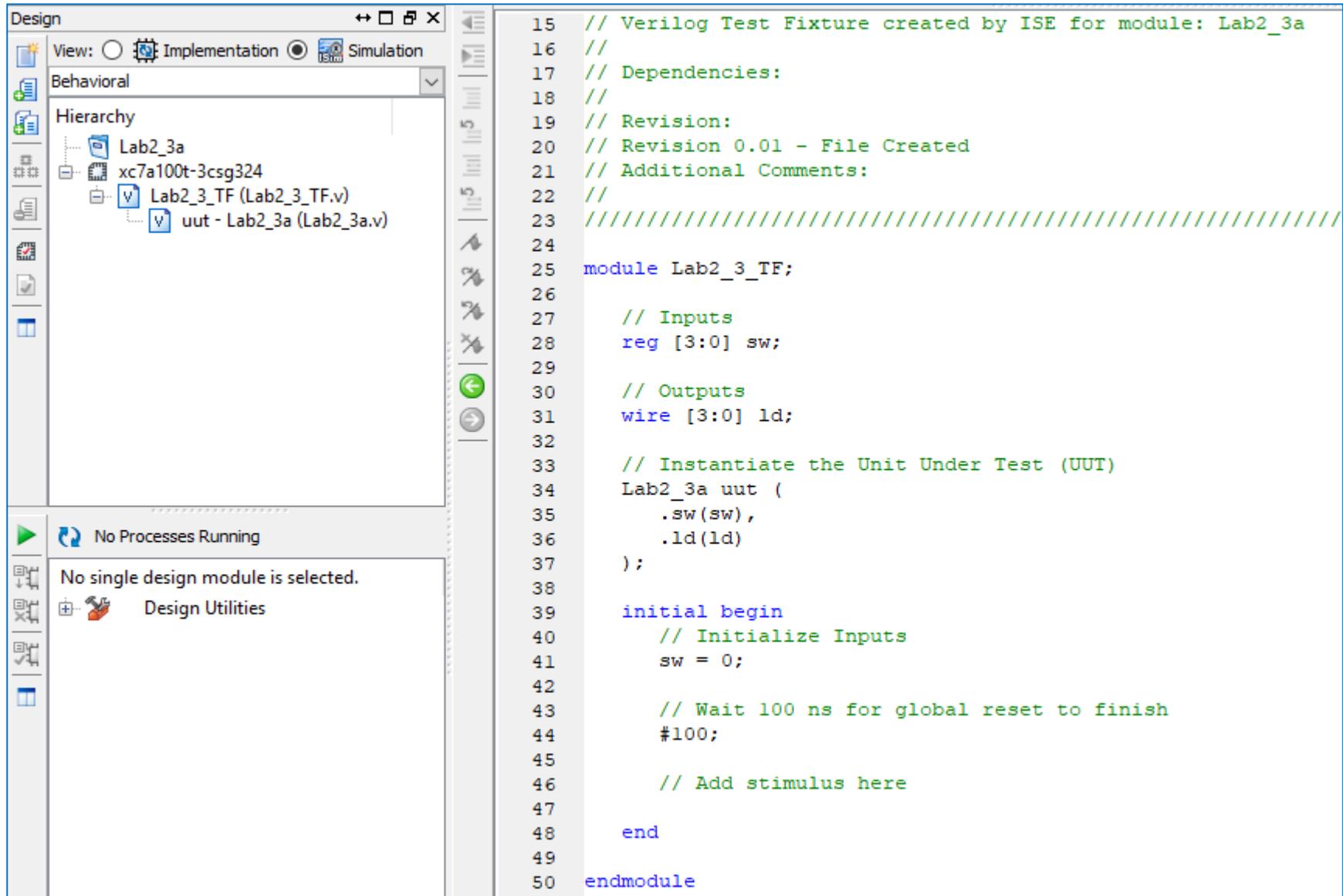
```
34 module Lab2_3a(  
35     input [3:0] sw,  
36     output [3:0] ld  
37 );  
38  
39 assign ld[0] = &sw[3:0]; // ÉS kapcsolat a 4 bites változó bitjeire // 1111?  
40 assign ld[1] = |sw[3:0]; // VAGY kapcsolat a 4 bites változó bitjeire  
41 assign ld[2] = ^sw[3:0]; // XOR kapcsolat a 4 bites változó bitjeire  
42 assign ld[3] = ~|sw[3:0]; // NOR kapcsolat a 4 bites változó bitjeire // 0000?  
43  
44 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



The screenshot displays the ISE IDE interface for a Verilog simulation. The left pane shows the design hierarchy for 'Lab2_3a', which includes the device 'xc7a100t-3csg324' and the test fixture 'Lab2_3_TF (Lab2_3_TF.v)'. The right pane shows the Verilog code for the test fixture, which is a Verilog Test Fixture created by ISE for the module 'Lab2_3a'. The code includes comments for dependencies, revision, and additional comments, followed by the module definition for 'Lab2_3_TF'. The module has a 4-bit input 'sw' and a 4-bit output 'ld'. It instantiates the Unit Under Test (UUT) 'Lab2_3a' with the input 'sw' and output 'ld'. The testbench includes an initial block to initialize the inputs to 0 and a delay of 100 ns for global reset to finish. The code ends with 'end' and 'endmodule'.

```
15 // Verilog Test Fixture created by ISE for module: Lab2_3a
16 //
17 // Dependencies:
18 //
19 // Revision:
20 // Revision 0.01 - File Created
21 // Additional Comments:
22 //
23 ///////////////////////////////////////////////////////////////////
24
25 module Lab2_3_TF;
26
27 // Inputs
28 reg [3:0] sw;
29
30 // Outputs
31 wire [3:0] ld;
32
33 // Instantiate the Unit Under Test (UUT)
34 Lab2_3a uut (
35     .sw(sw),
36     .ld(ld)
37 );
38
39 initial begin
40     // Initialize Inputs
41     sw = 0;
42
43     // Wait 100 ns for global reset to finish
44     #100;
45
46     // Add stimulus here
47
48 end
49
50 endmodule
```

Test vectors generation

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

Test vector generation using *for loop*

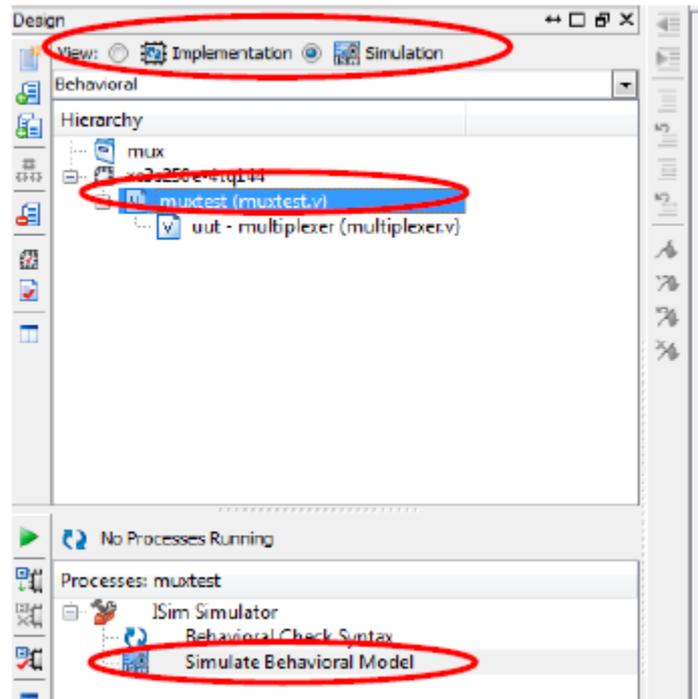
```
25 module Lab2_3_TF;
26     // Inputs
27     reg [3:0] sw;
28     // Outputs
29     wire [3:0] ld;
30     // Instantiate the Unit Under Test (UUT)
31     Lab2_3a uut (
32         .sw(sw),
33         .ld(ld)
34     );
35
36     integer i ;
37     initial begin
38         // Initialize Inputs
39         sw = 0;
40         // Wait 100 ns for global reset to finish
41         #100;
42         // Add stimulus here
43
44     // Teljes tesztvektorkészlet ciklussal generálva
45     for (i = 0 ; i<=15; i = i+1)
46     begin
47         #100 sw = i;
48     end
49
50     end
51 endmodule
```

Test vector generation using linear code

```
25 module Lab2_3_TF;
26     // Inputs
27     reg [3:0] sw;
28     // Outputs
29     wire [3:0] ld;
30     // Instantiate the Unit Under Test (UUT)
31     Lab2_3a uut (
32         .sw(sw),
33         .ld(ld)
34     );
35
36     integer i ;
37     initial begin
38         // Initialize Inputs
39         sw = 0;
40         // Wait 100 ns for global reset to finish
41         #100;
42         // Add stimulus here
43         // Teljes tesztvektorkészlet lineáris felsorolással
44         #100 sw = 4'h0;
45         #100 sw = 4'h1;
46         #100 sw = 4'h2;
47         #100 sw = 4'h3;
48         #100 sw = 4'h4;
49         #100 sw = 4'h5;
50         #100 sw = 4'h6;
51         #100 sw = 4'h7;
52         #100 sw = 4'h8;
53         #100 sw = 4'h9;
54         #100 sw = 4'ha;
55         #100 sw = 4'hb;
56         #100 sw = 4'hc;
57         #100 sw = 4'hd;
58         #100 sw = 4'he;
59         #100 sw = 4'hf;
60
61     end
62 endmodule
```

Simulation

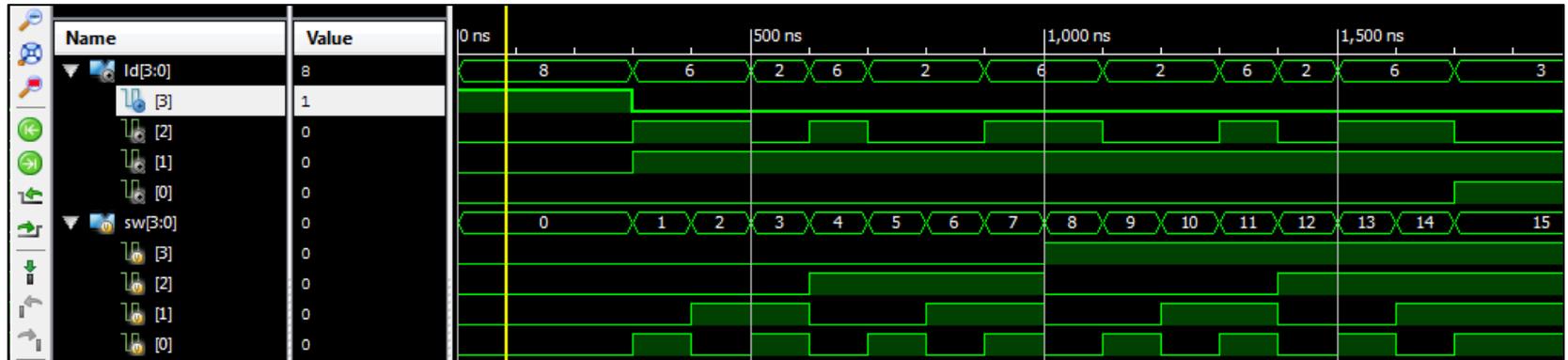
- Project Navigator program View: Simulation, in Hierarchy select the testfixture (*Lab2_3_TF*).
- *In Processes window choose ISim Simulator /Simulate Behavioral Model.*



Lab2_3a results

- **Simulation results**

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



- **Design implementation:**

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