Digital design laboratory 3

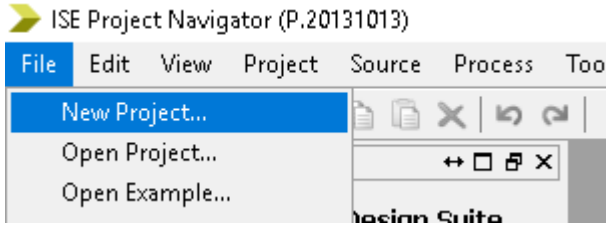
BCD code checker

- The first task is the implementation of the codechecker circuit we have designed on practice
- Valid BCD codes are 4-bit binary numbers that represent values between 0 and 9
- Thus, the circuit has 4 inputs and one error output
- The error output is set to 1 if the input is higher than 9

BCD code checker

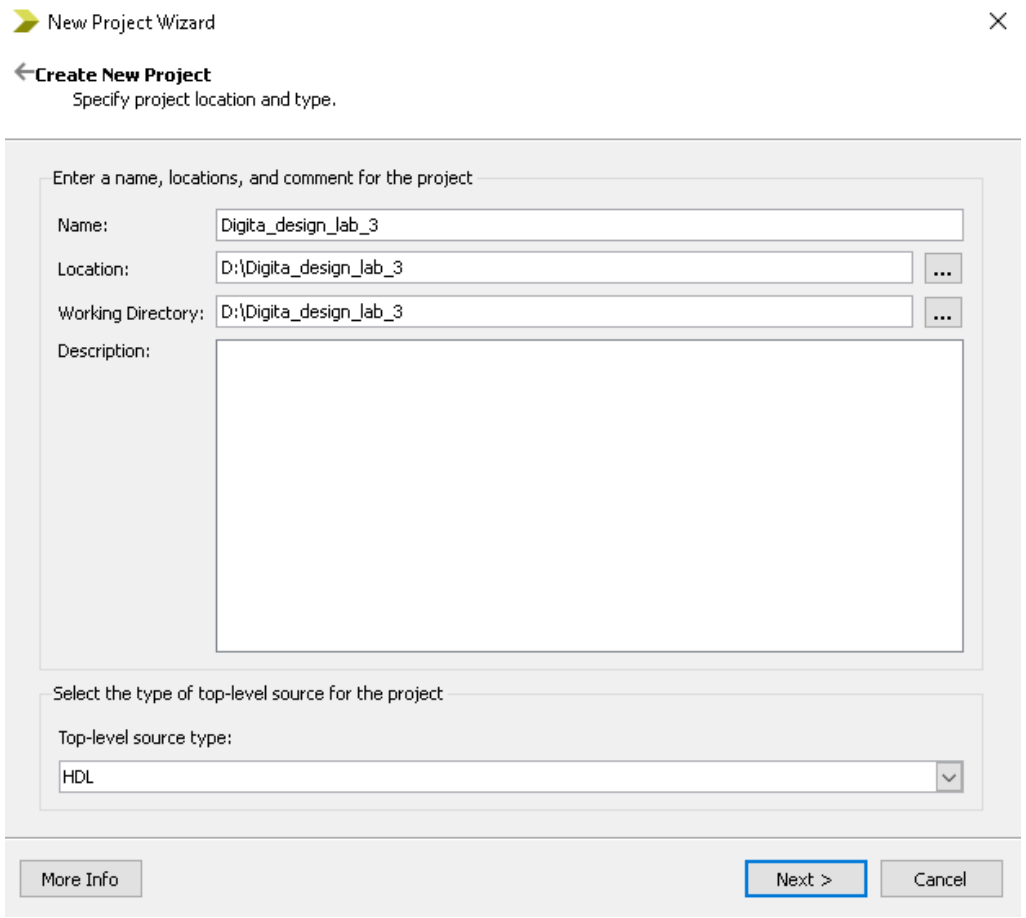
- Launch the Xilinx ISE:

- Select File -> New Project…



BCD code checker

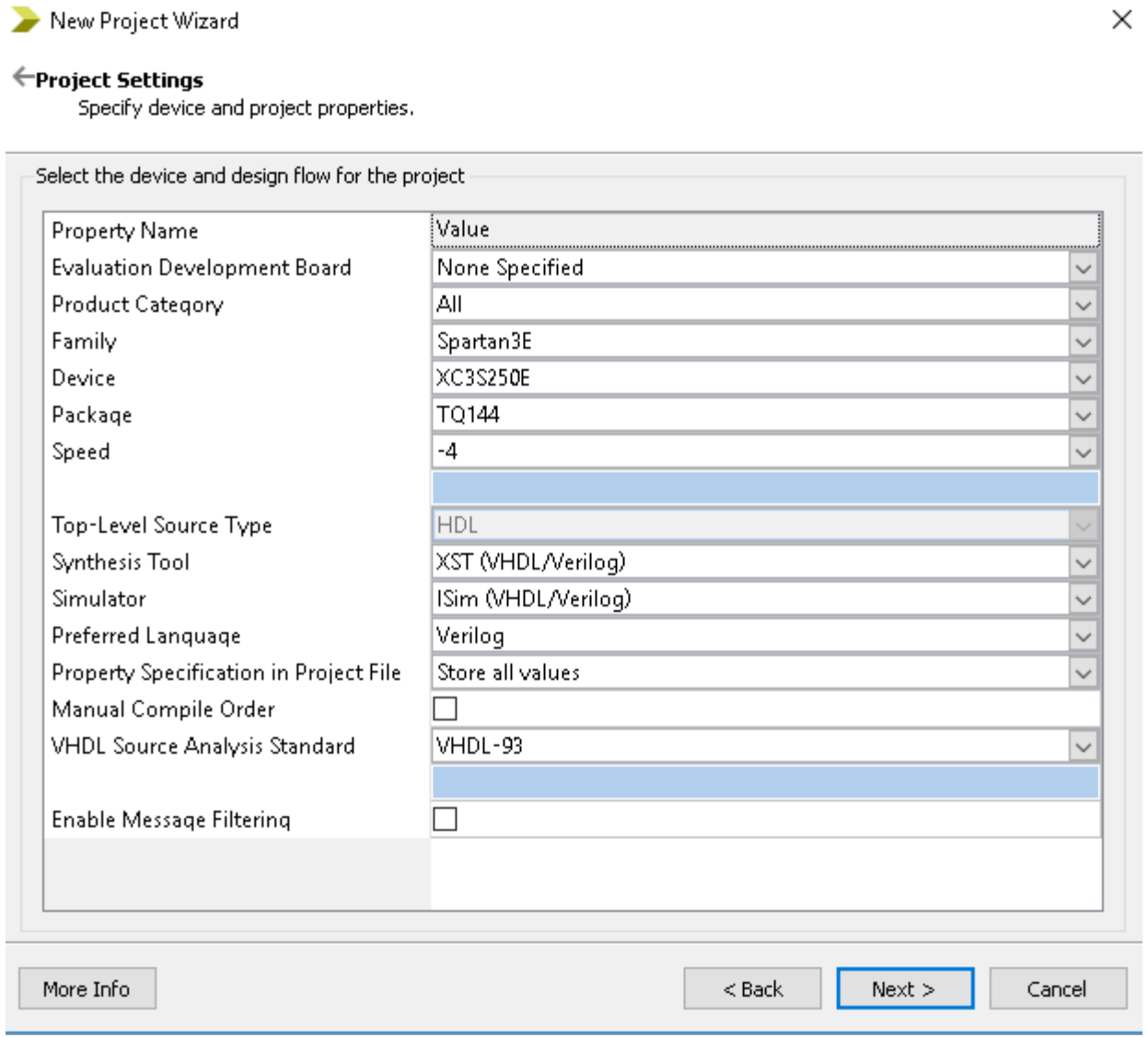
- Name: Digital_design_lab_3
- Location: work on drive D:
- Press Next



BCD code checker

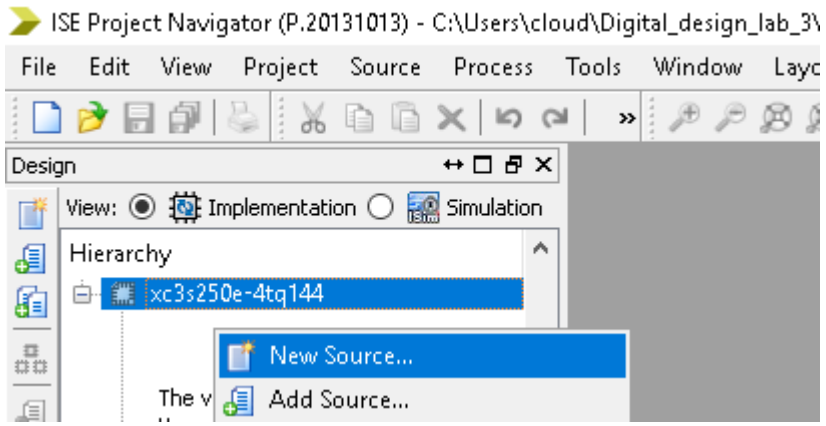
- Verify the settings
  Family: Spartan3E
  Device: XC3S250E
  Package: TQ144
  Speed: -4

- Press Next and Finish



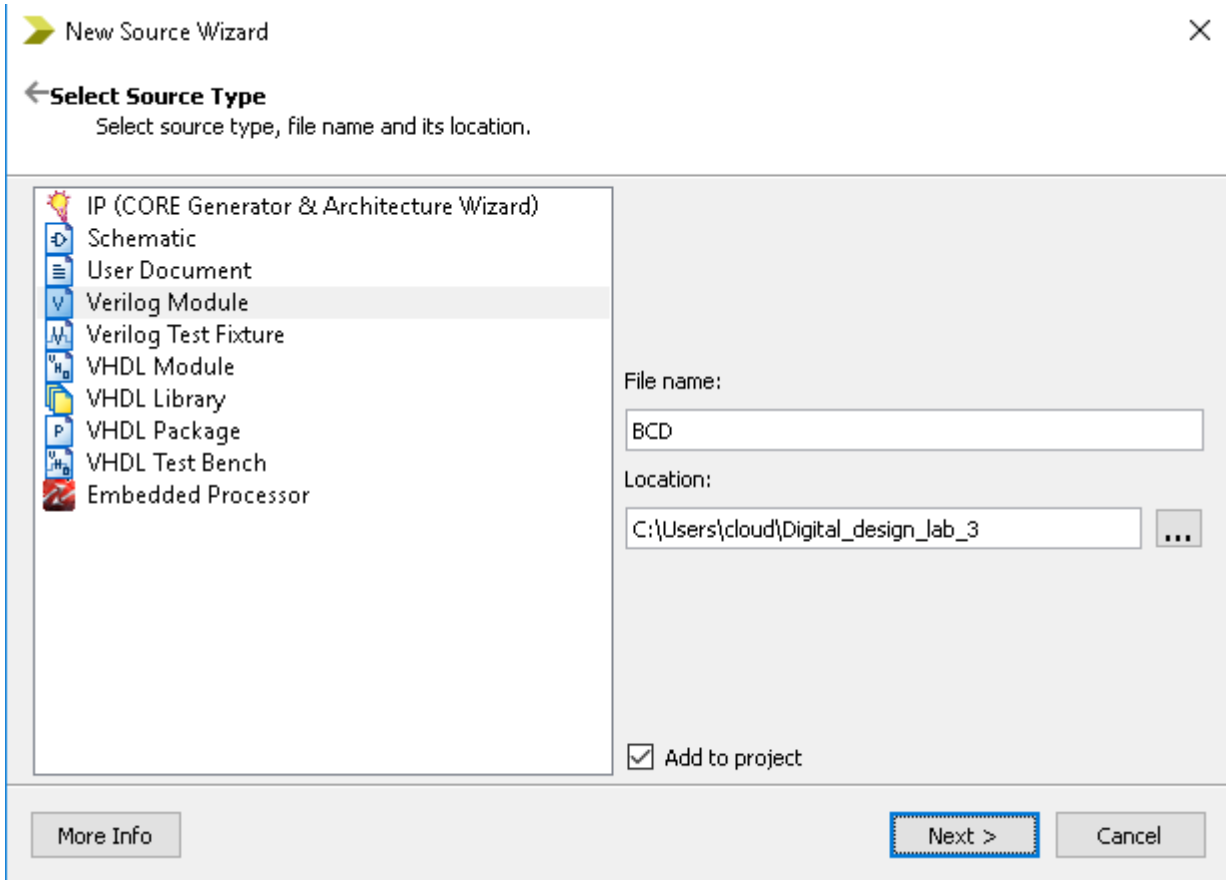
BCD code checker

- Once the project is created, right click on the label „xc3s250e-4tq144”, and select New Source…



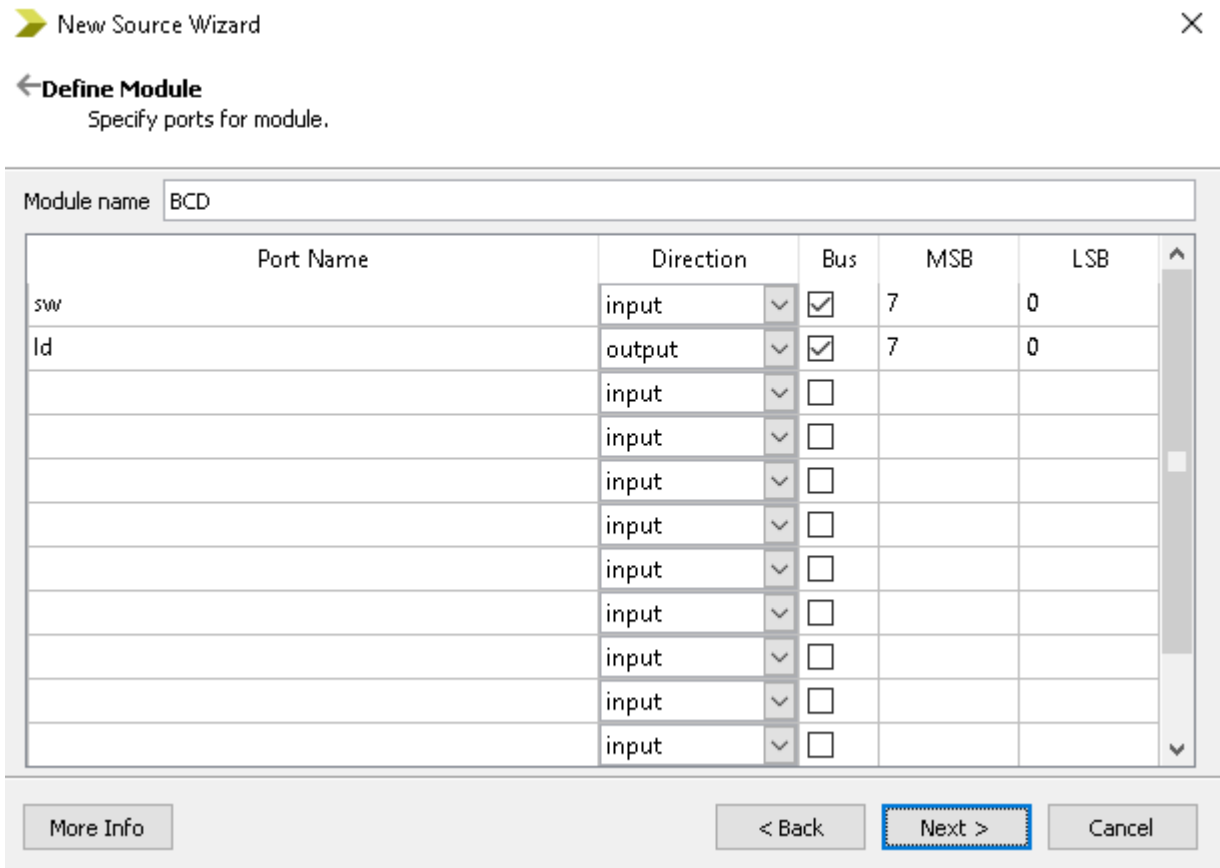
BCD code checker

- Select Verilog Module
- Name: BCD
- Do not modify the working directory
- Make sure the „Add to project” checkbox is checked
- Press Next



BCD code checker

- Add „sw” as input, Bus is checked, MSB is 7 and LSB is 0
- Add „ld” as output, Bus is checked, MSB is 7 and LSB is 0
- Press Next, then press Finish



BCD code checker

- The code checker has 6 invalid inputs: 10, 11, 12, 13, 14, 15
- So the error function consists of 6 minterms
- This could be transformed directly into a gate level circuit, but it require many gates (6 4-input AND gates, 1 6-input OR gate)
- Next slide summarizes the minimization process we did on practice
- The simplified circuit consists of 2 2-input AND gates and 1 2-input OR gate

BCD code checker

Input (ABCD)	Error	Input (ABCD)	Error
0000	0	1000	0
0001	0	1001	0
0010	0	1010	1
0011	0	1011	1
0100	0	1100	1
0101	0	1101	1
0110	0	1110	1
0111	0	1111	1

- F = AB'CD'+AB'CD+ABC'D'+ABC'D+ABCD'+ABCD = AB'C+ABC'+ABC = AB'C+ABC+ABC'+ABC = **AB+AC**

BCD code checker

- Add the following lines to the BCD module
- sw[3], sw[2], sw[1] and sw[0] will represent A, B, C, D
- ld[7]…ld[0] is also driven to avoid warning messages
- {N{EXPRESSION}} syntax: use curly braces! Description on the next slide

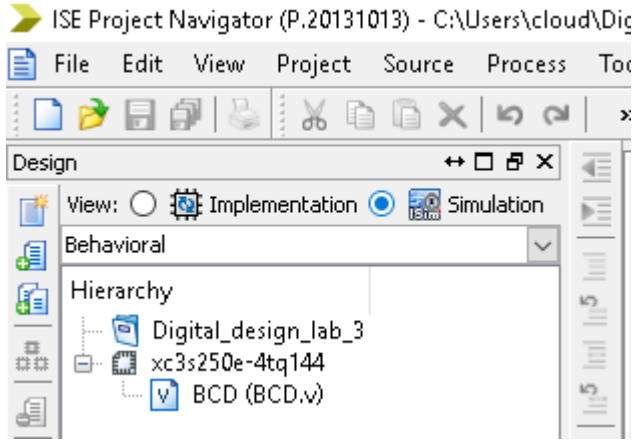
```
21 module BCD(
22     input [7:0] sw,
23     output [7:0] ld
24     );
25
26    assign ld[0] = sw[3]&sw[2]|sw[3]&sw[1];
27    // dummy line to avoid warnings caused by unused inputs and unset outputs:
28    assign ld[7:1] = {7{sw[0]&sw[4]&sw[5]&sw[6]&sw[7]}};
29
30 endmodule
```

BCD code checker

- {N{EXPRESSION}} syntax:
  - EXPRESSION is a logical expression, it can be true or false
  - The above syntax means that „take the value of expression, and repeat it N times
  - For example, if EXPRESSION=1 (true) and N=6, the result is the following 6 bit binary number: 111111
- If you have added the code, save all changes before the next step

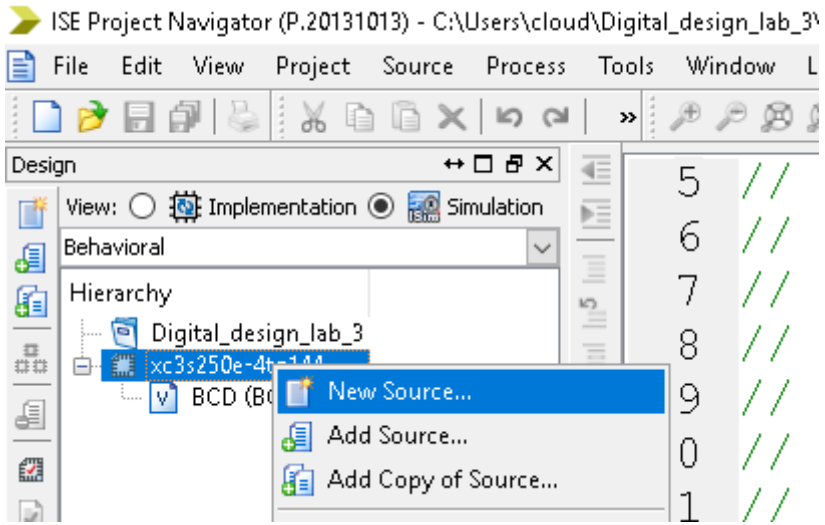
BCD code checker

- In the top left corner, swith to simulation mode



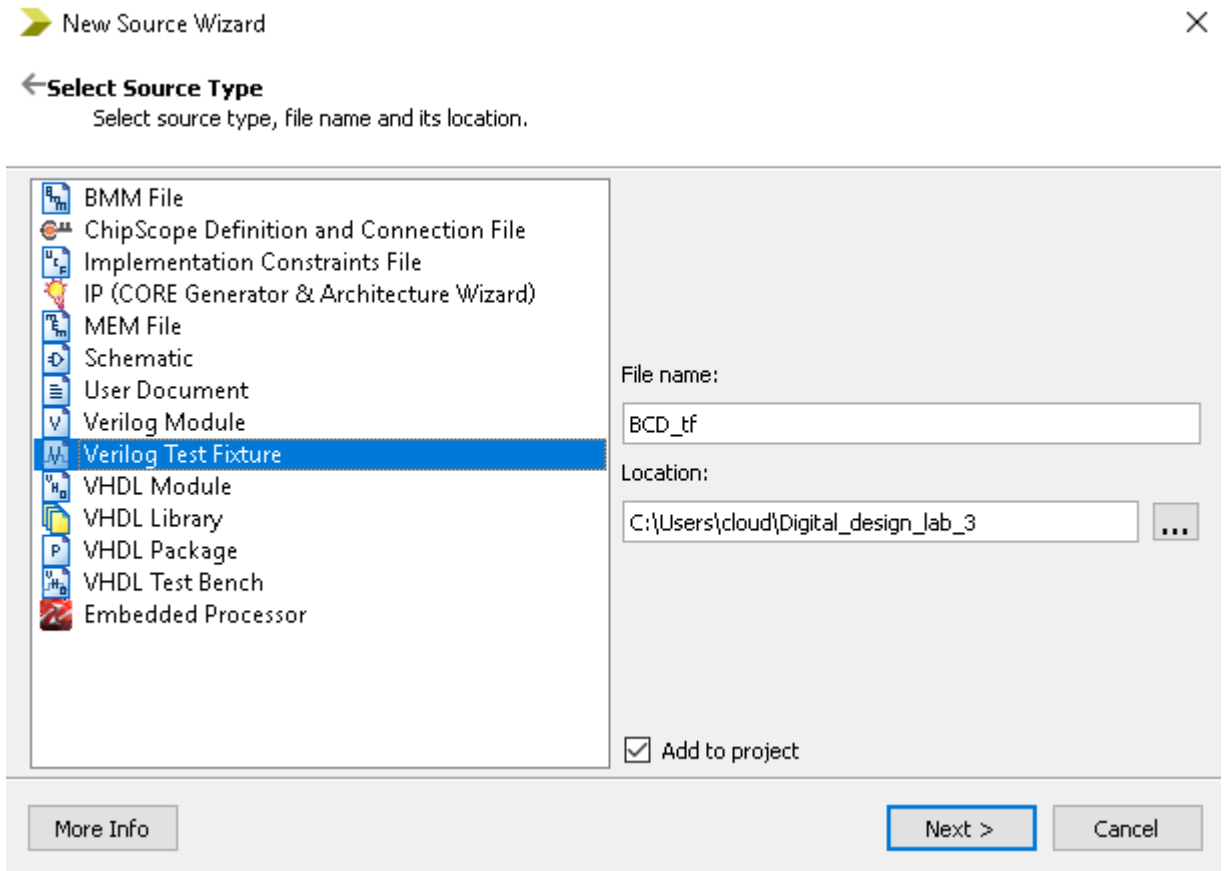
BCD code checker

- Right click on the „xc3s250e-4tq144” label, and select New Source…



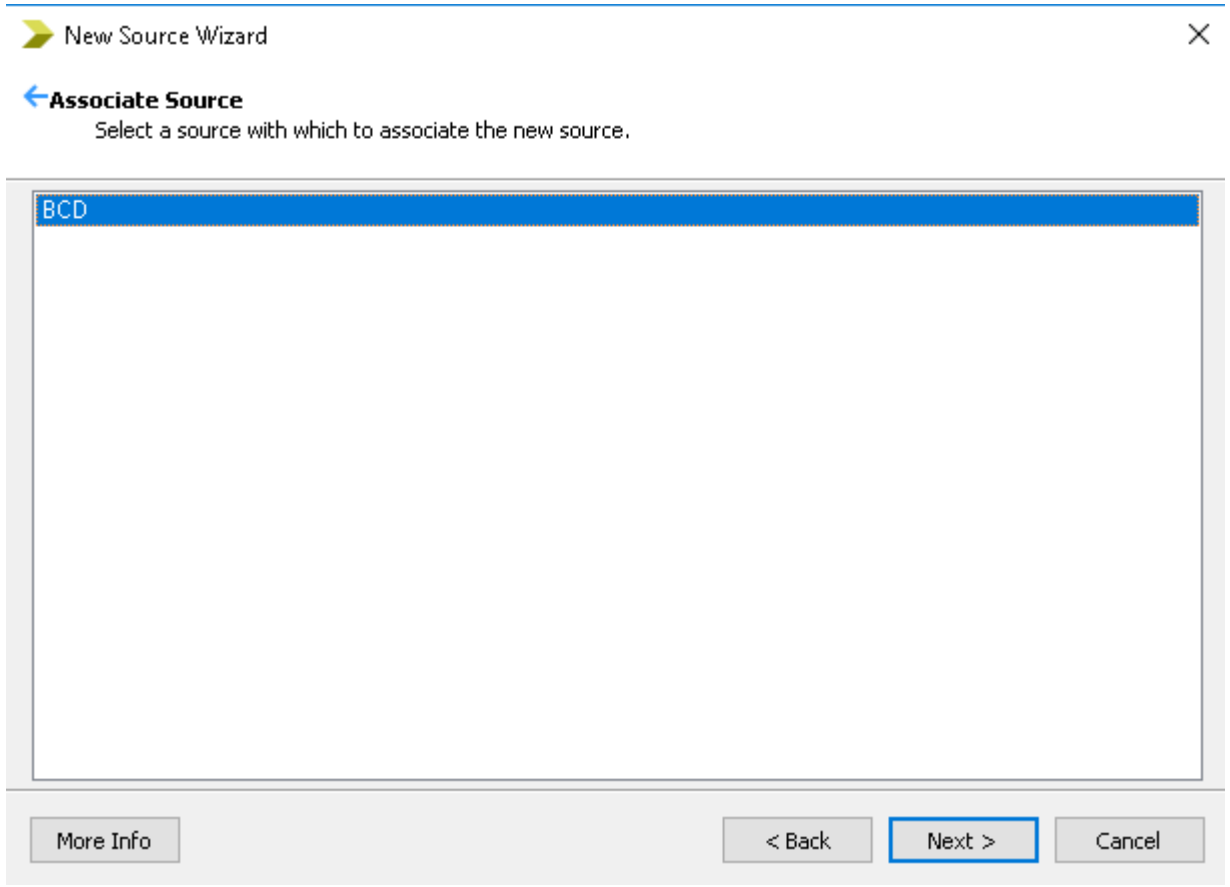
BCD code checker

- Select Verilog Text Fixture
- Name: BCD_tf
- Do not modify the location
- Check „Add to project”
- Press Next



BCD code checker

- The „Associate source” window appers
- Select BCD
- Press Next



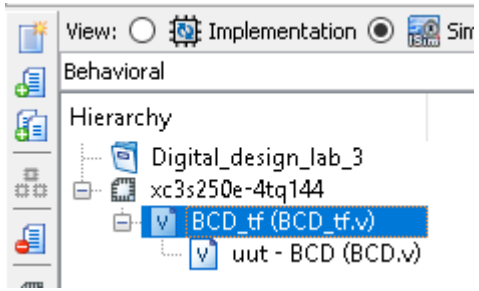
BCD code checker

- The circuit will be tested using a for loop
- Add the following lines to your code
  - integer i; **BEFORE** the „initial begin” part
  - Copy the code after the „Add stimulus here” comment
  - Note: i++ is not accepted here
- Save all changes

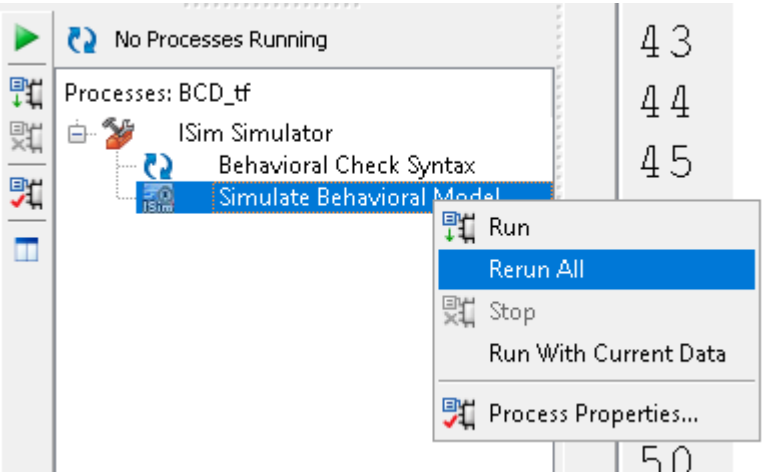
```
33    // Instantiate the Unit Under Test (UUT)
34    BCD uut (
35       .sw(sw),
36       .ld(ld)
37    );
38    integer i;
39    initial begin
40       // Initialize Inputs
41       sw = 0;
42
43       // Wait 100 ns for global reset to finish
44       #100;
45
46       // Add stimulus here
47       for (i=0; i<16; i=i+1) // i++ is not accepted here!
48       begin
49          #25; sw = i;
50       end
51    end
52
53 endmodule
```

BCD code checker

- In the top left corner, left click on the BCF_tf module:

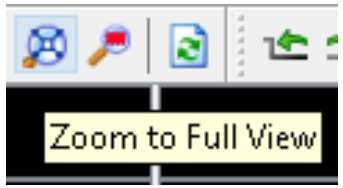


- Press the + sign next to the Isim simulator, right click on „Simulate Behavioral Model” and select Rerun All

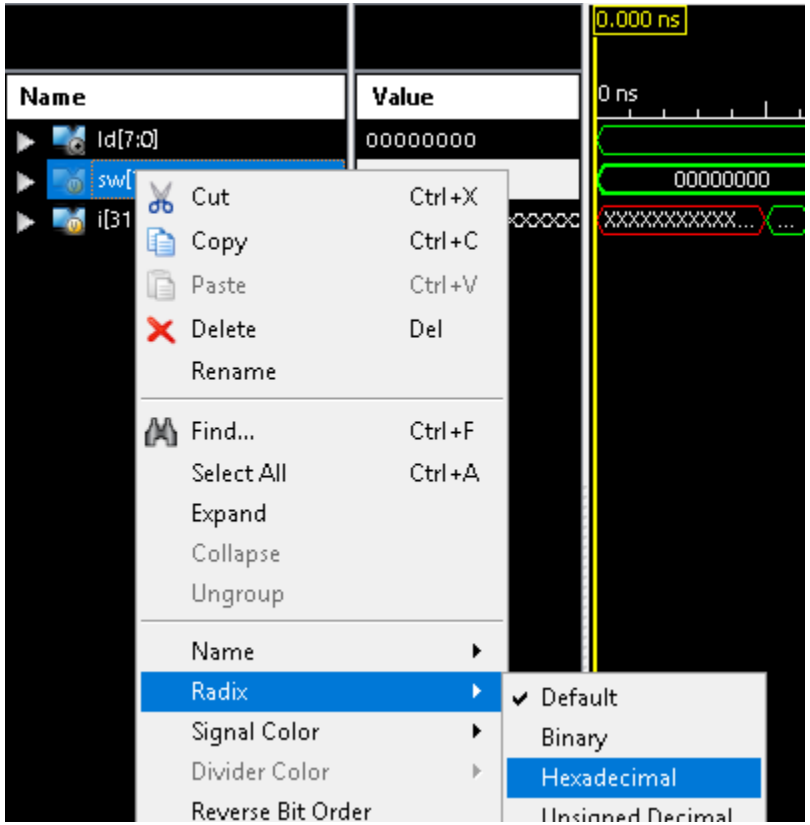


BCO code checker

- The simulator starts and the waveforms appear, press the Zoom to Full View button:

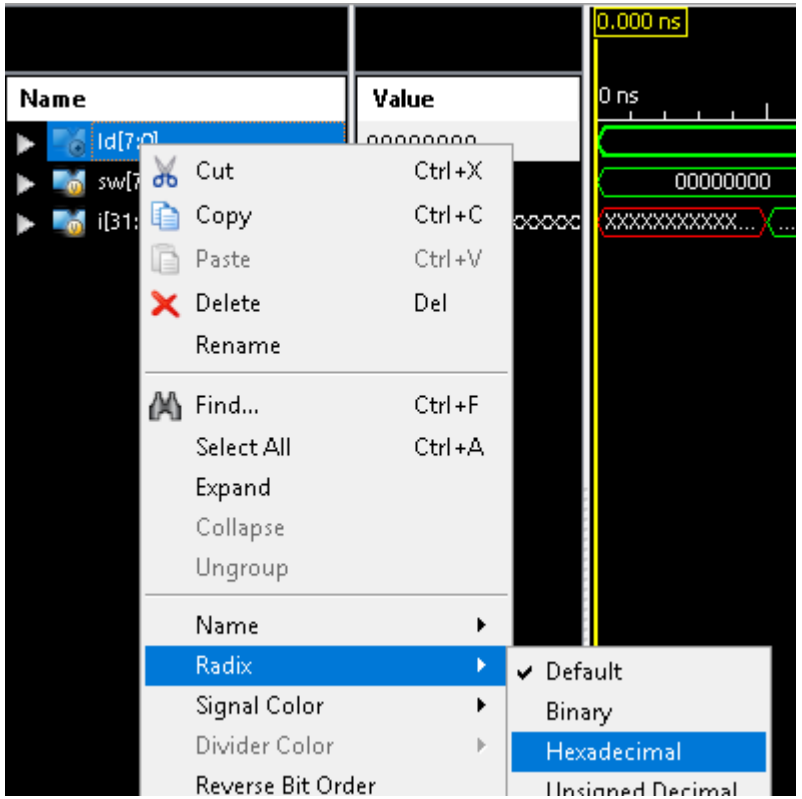


- Right click on the sw[7:0] signal, select Radix->Hexadecimal



BCD code checker

- Do the same for the ld signal

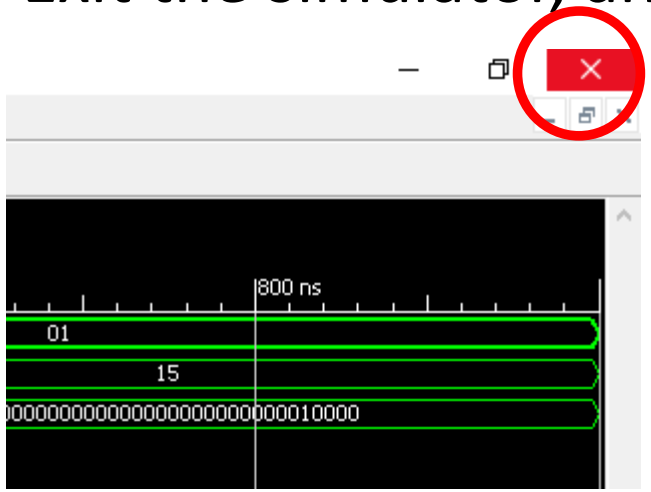


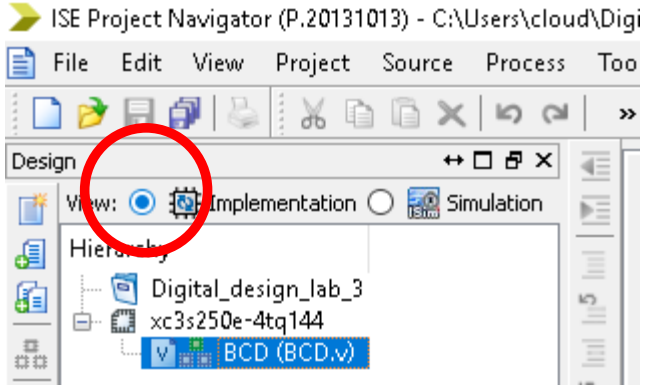
- Now you can read the inputs and outputs as hexadecimal values:

- Try to switch to decimal representation

BCD code checker

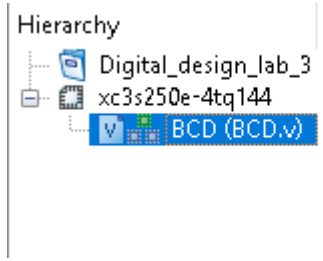
- Exit the simulator, and switch back to implementation mode:



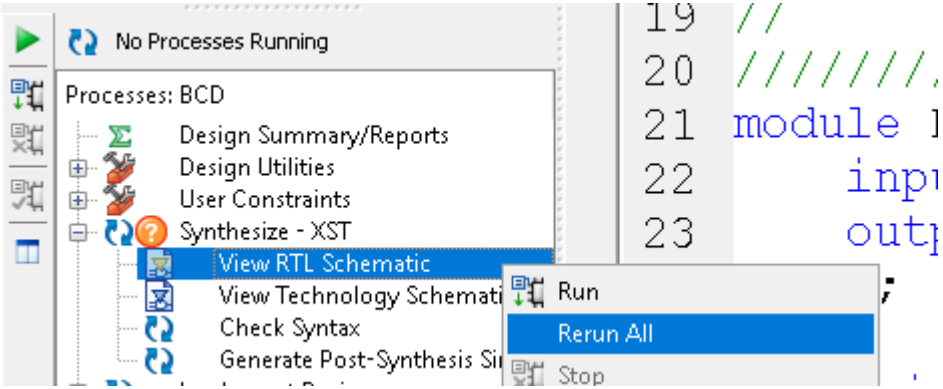


BCD code checker

- Now we will check the design verilog generates from the module code you have entered.
- Left click on the BCD module in the top left corner

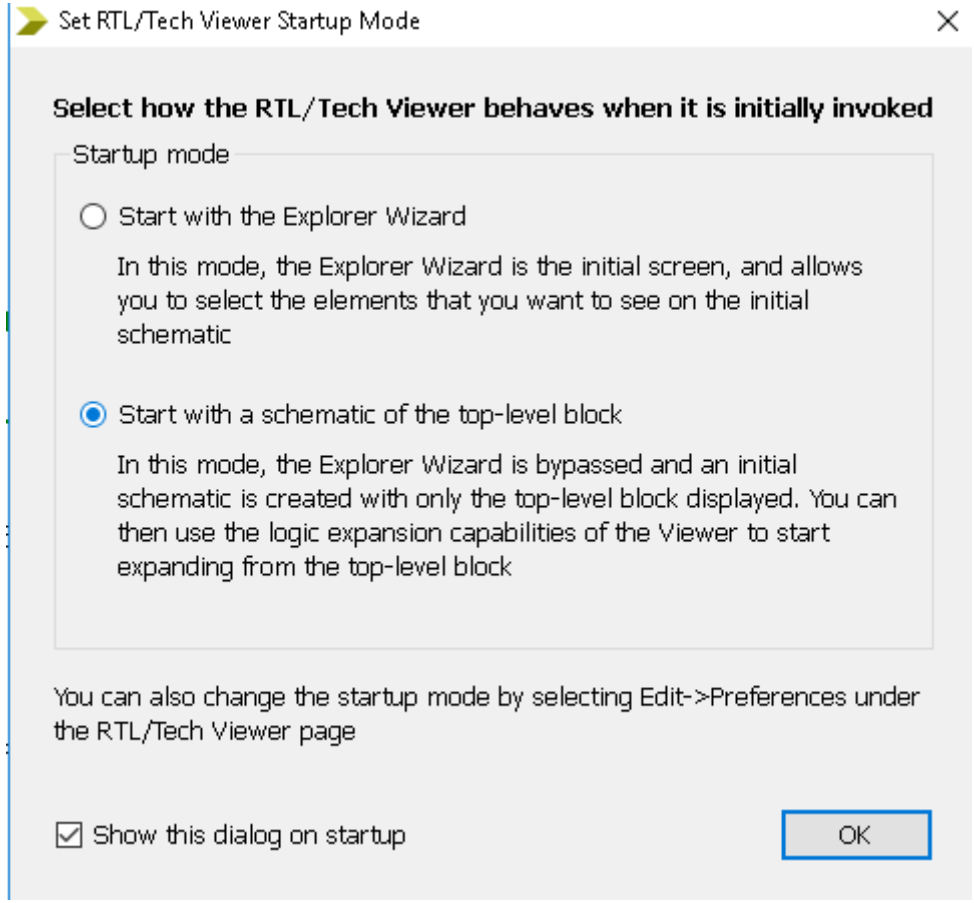


- Press the + sign next to Synthesiye-XST, right click on „View RTL schematic”, and select „Rerun All”



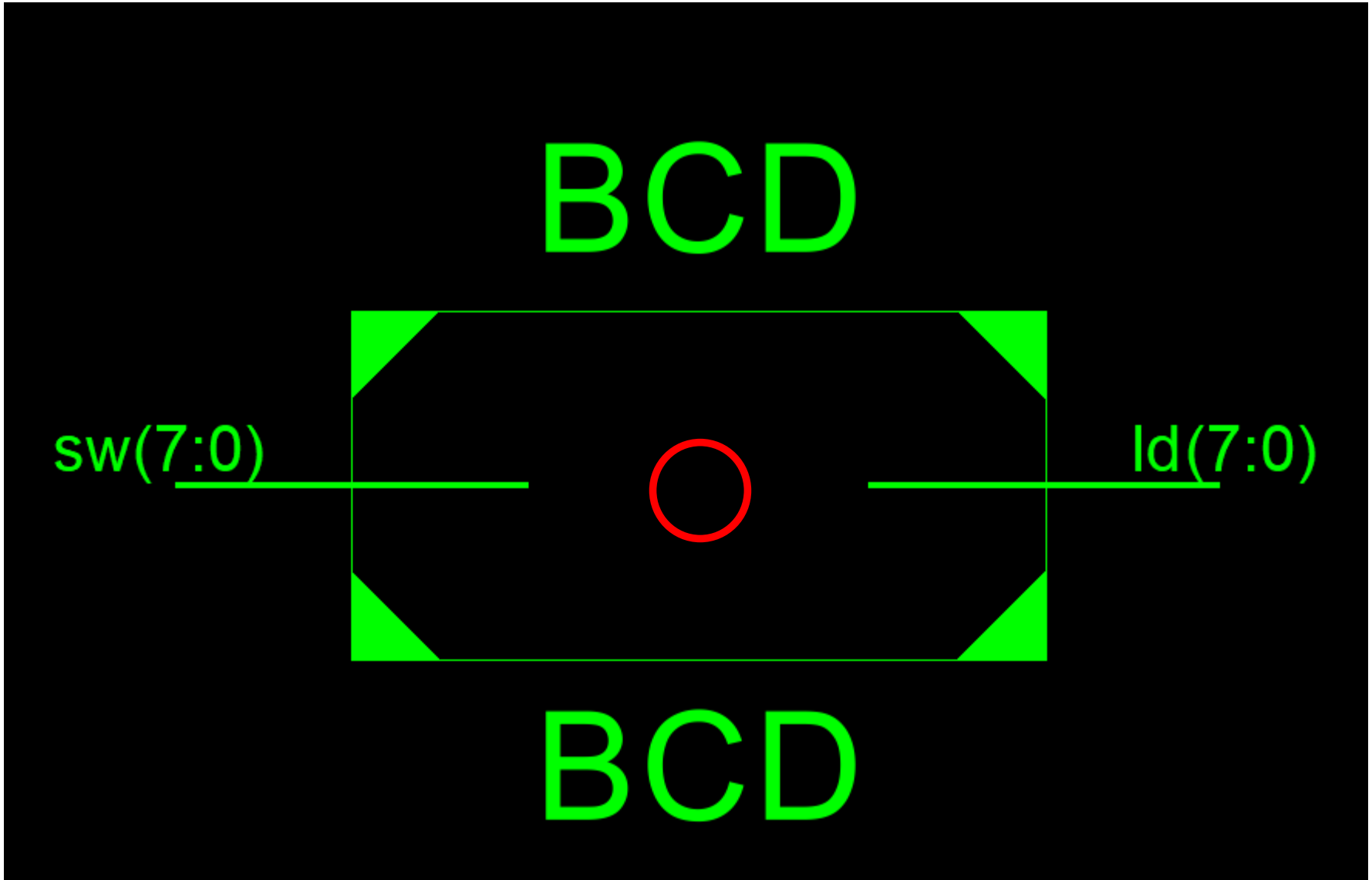
BCD code checker

- Select „Start with a schematic of the top level block” in the popup window
- Press OK



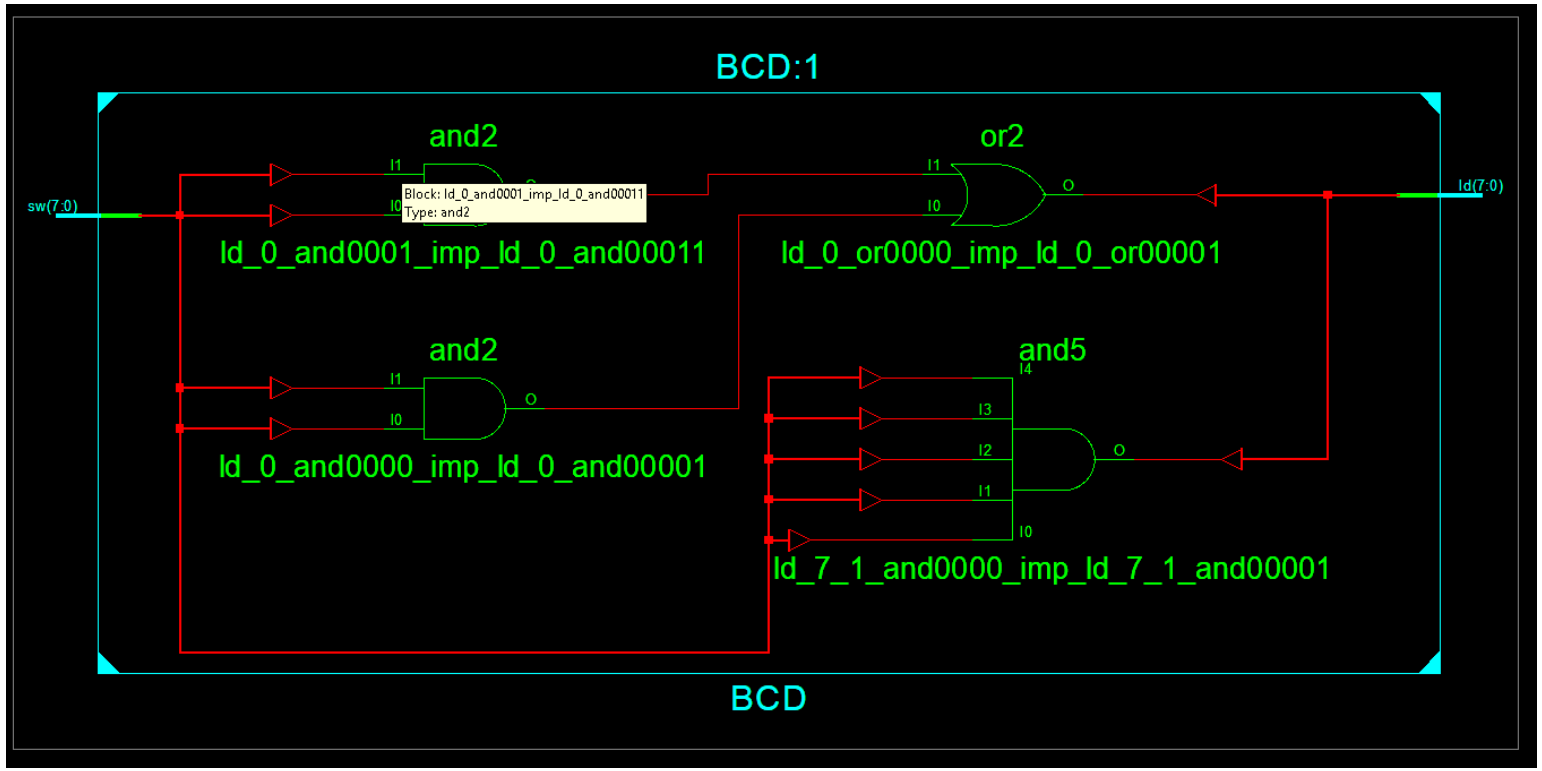
BCD code checker

- Double click inside the BCD module (red circle)



BCD code checker

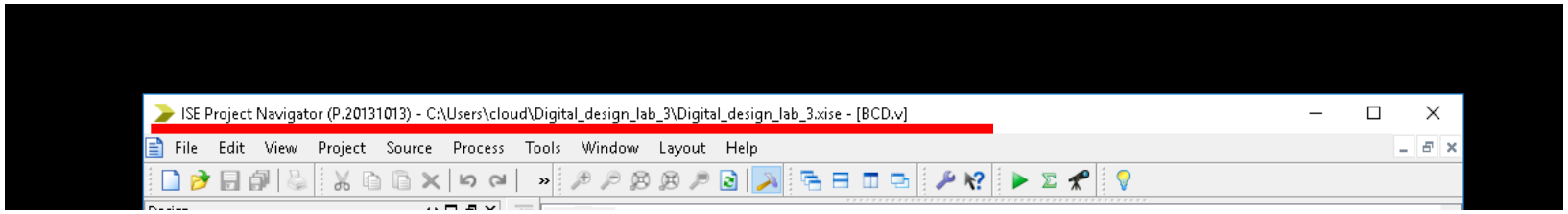
- The gate level design of the module appears:



- Study the design, focusing on the two and2 gates and the or2 gate (the and5 gate belongs to our dummy line to avoid warnings)

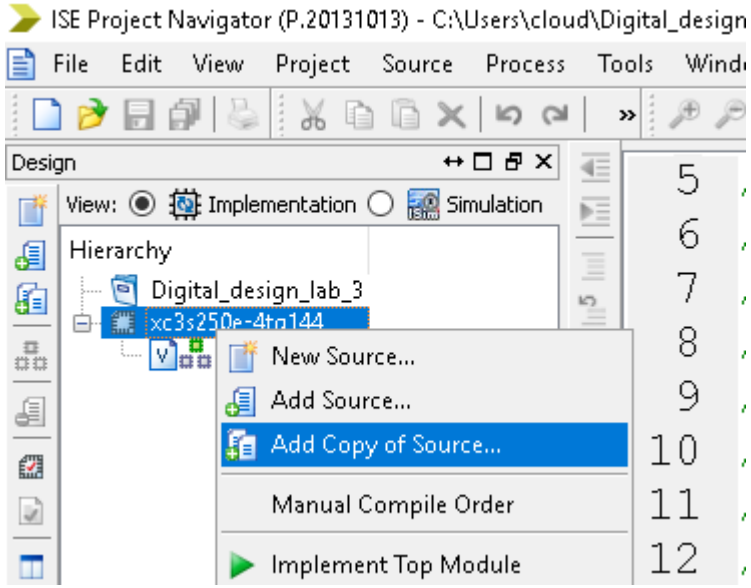
BCD code checker

- Now we will implement and download the module to the FPGA board. A file has been prepared for this purpose, you can download it using the following link: download
- If the browser opens the file instead of displaying the download dialog window: right click, and select “Save as…”
- Download the file, and save it to the **current working directory** of your project (D:\Digital_design_lab_2).
- If you are not sure, you can check it in the title bar of the ISE (top of the ISE window)



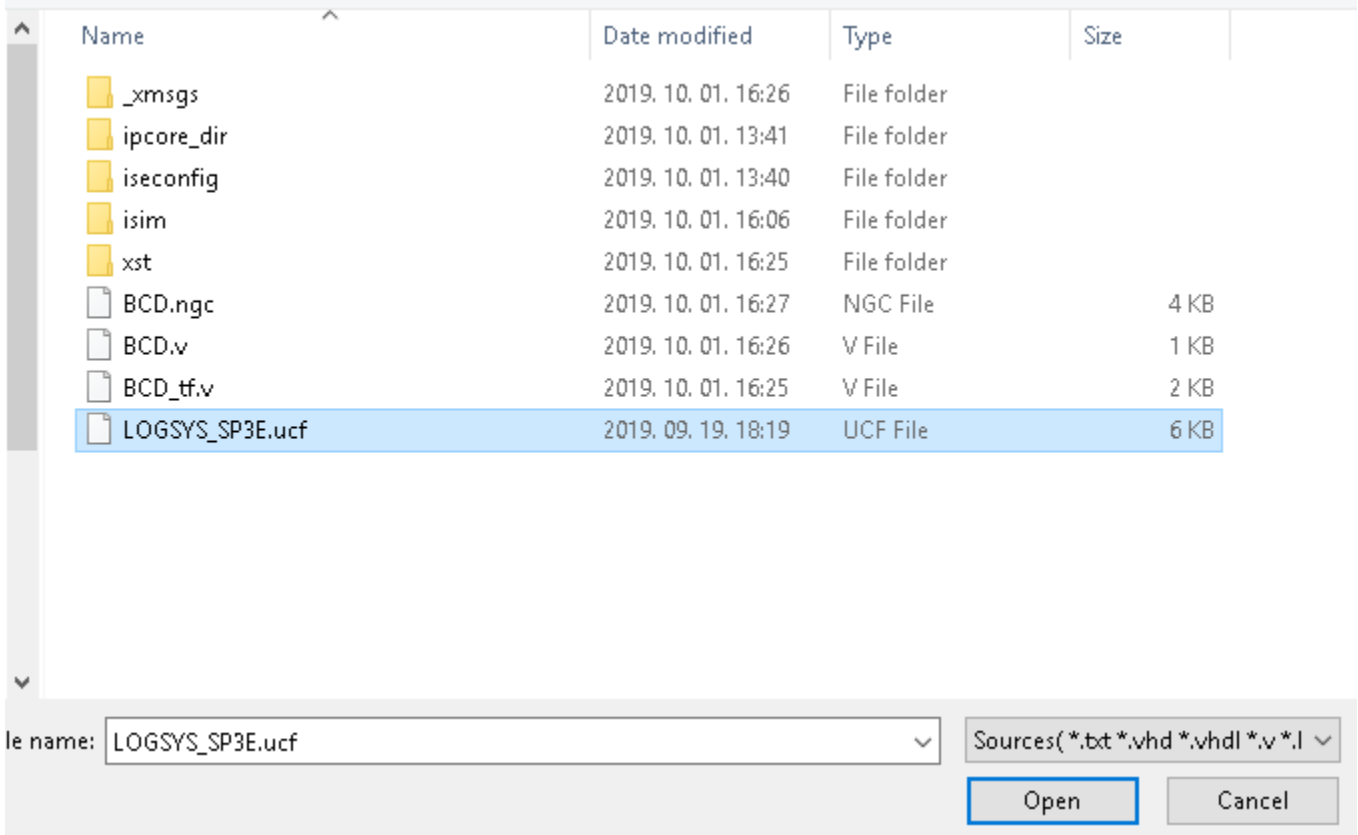
BCD code checker

- Go back to the ISE, right click on the „xc3s250e-4tq144” label and select „Add copy of source”



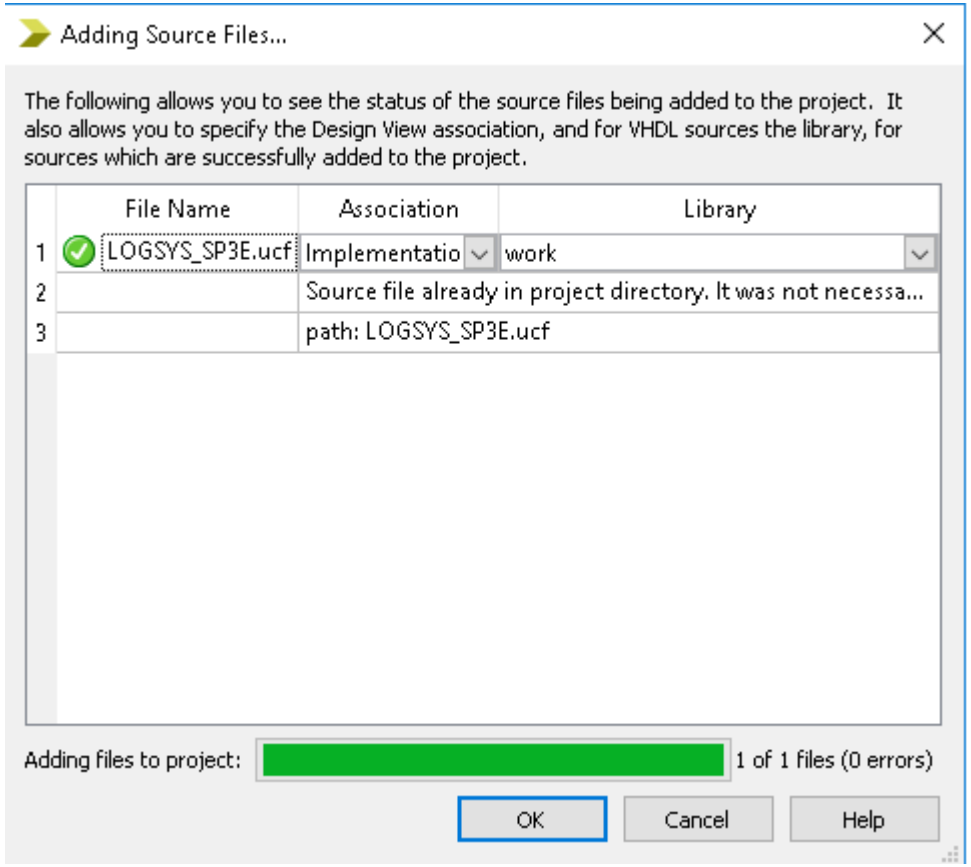
BCD code checker

- Browse the file you have downloaded, select it and press Open:



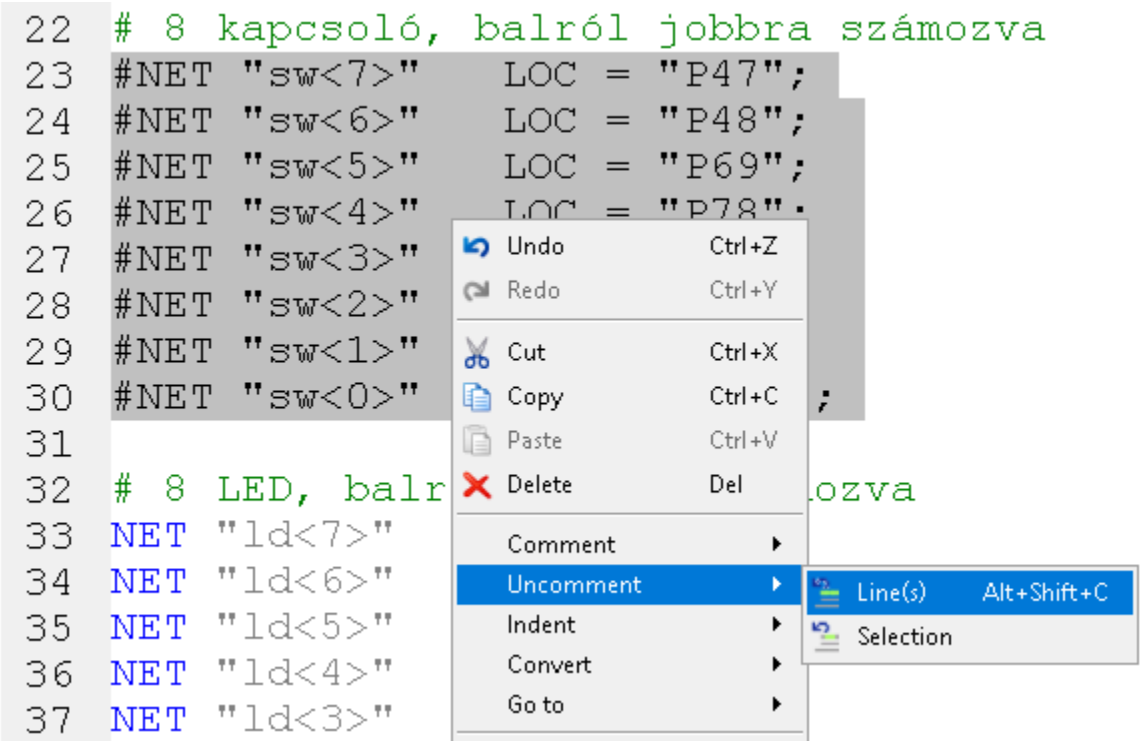
BCD code checker

- The following window appears:
- Press OK
- If you have difficulties, ask for assistance



BCD code checker

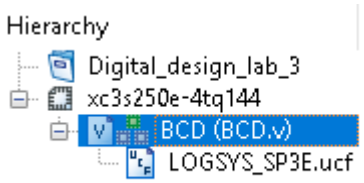
- Open the downloaded file inside the ISE, and uncomment the lines belongig to „sw”and „ld”



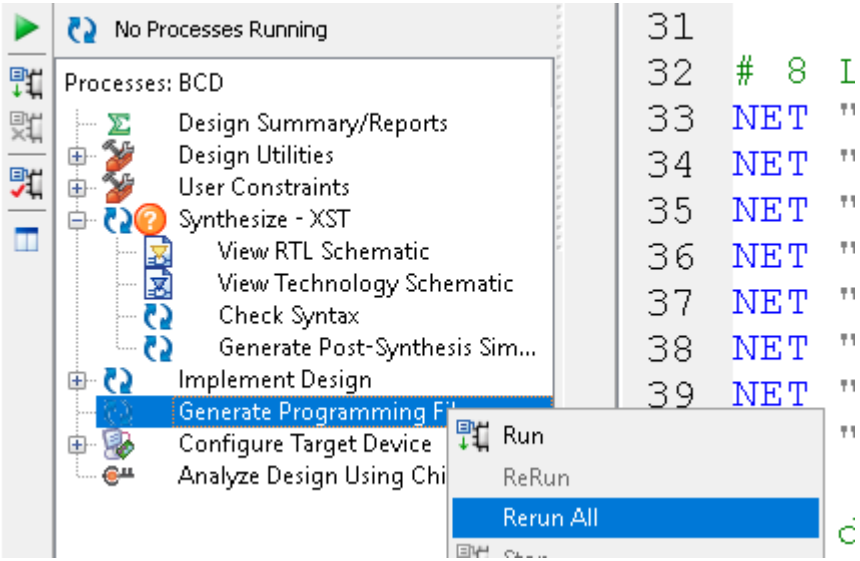
- Press save

BCD code checker

- Select the BCD module in the top left corner (left click)

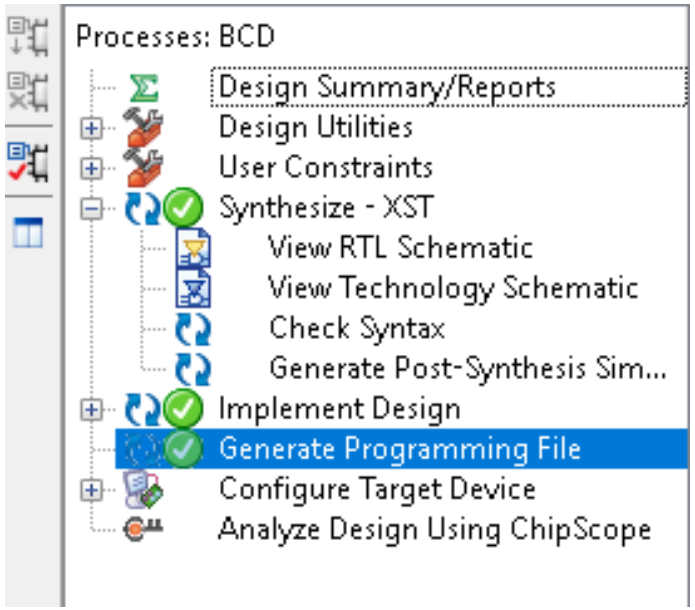


- Then right click on „Generate Programming File", and select „Rerun All"



BCD code checker

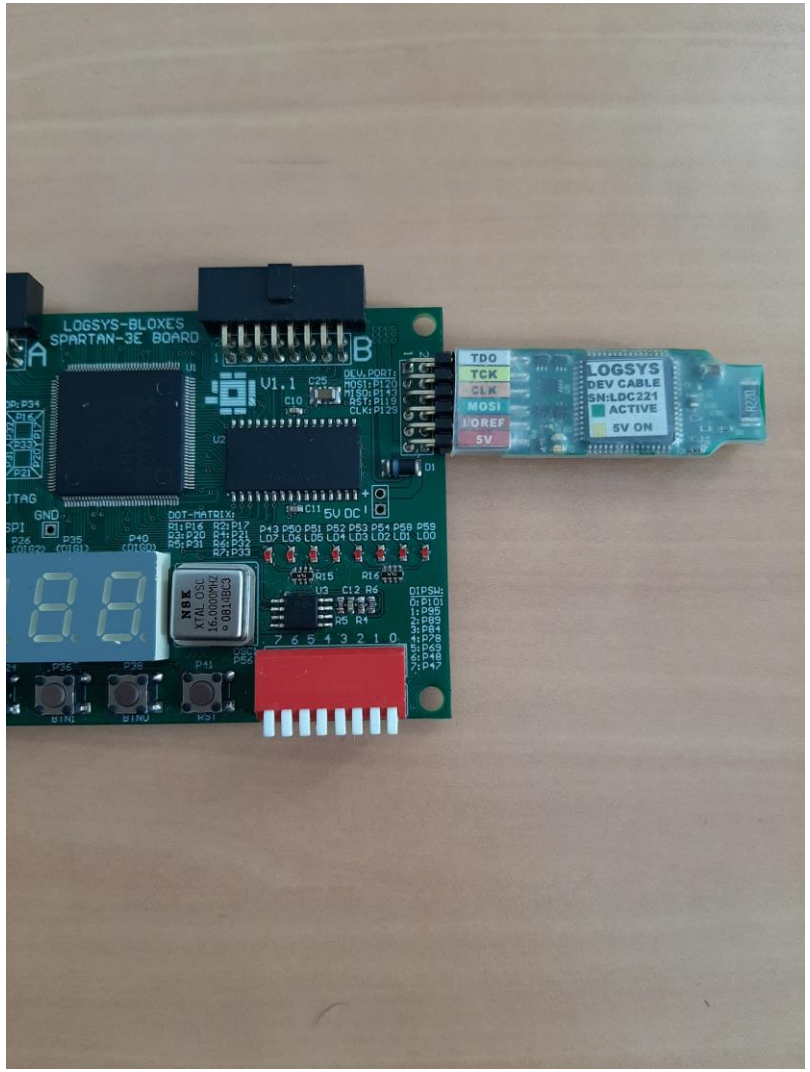
- Wait until the programming file is generated
- If everything is OK, you should see this:



- Ask for assistance if you see errors (red X) or warnings (yellow !).

BCD code checker

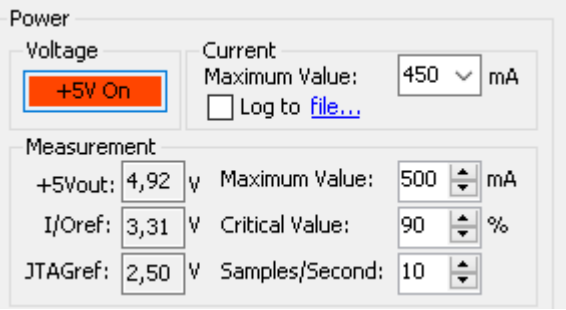
- If the program file was generated succesfully, you can connect the FPGA board to the PC
- Mind the orientation of the JTAG connector!



BCD code checker

- Launch the Logsys GUI application

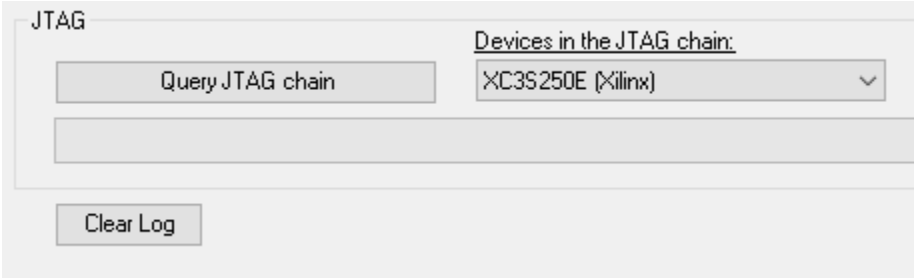
- Press the +5V button to turn the board on



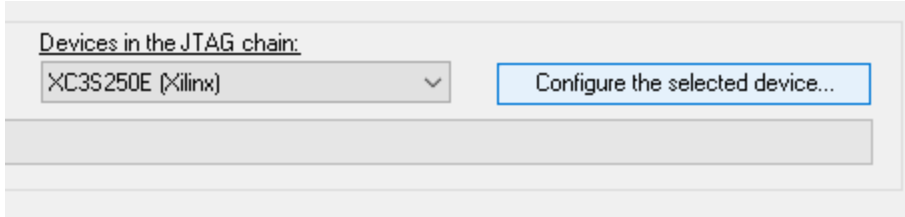
BCD code checker

- On the right side of the screen, select JTAG download:

- Press the „Query JTAG chain” button

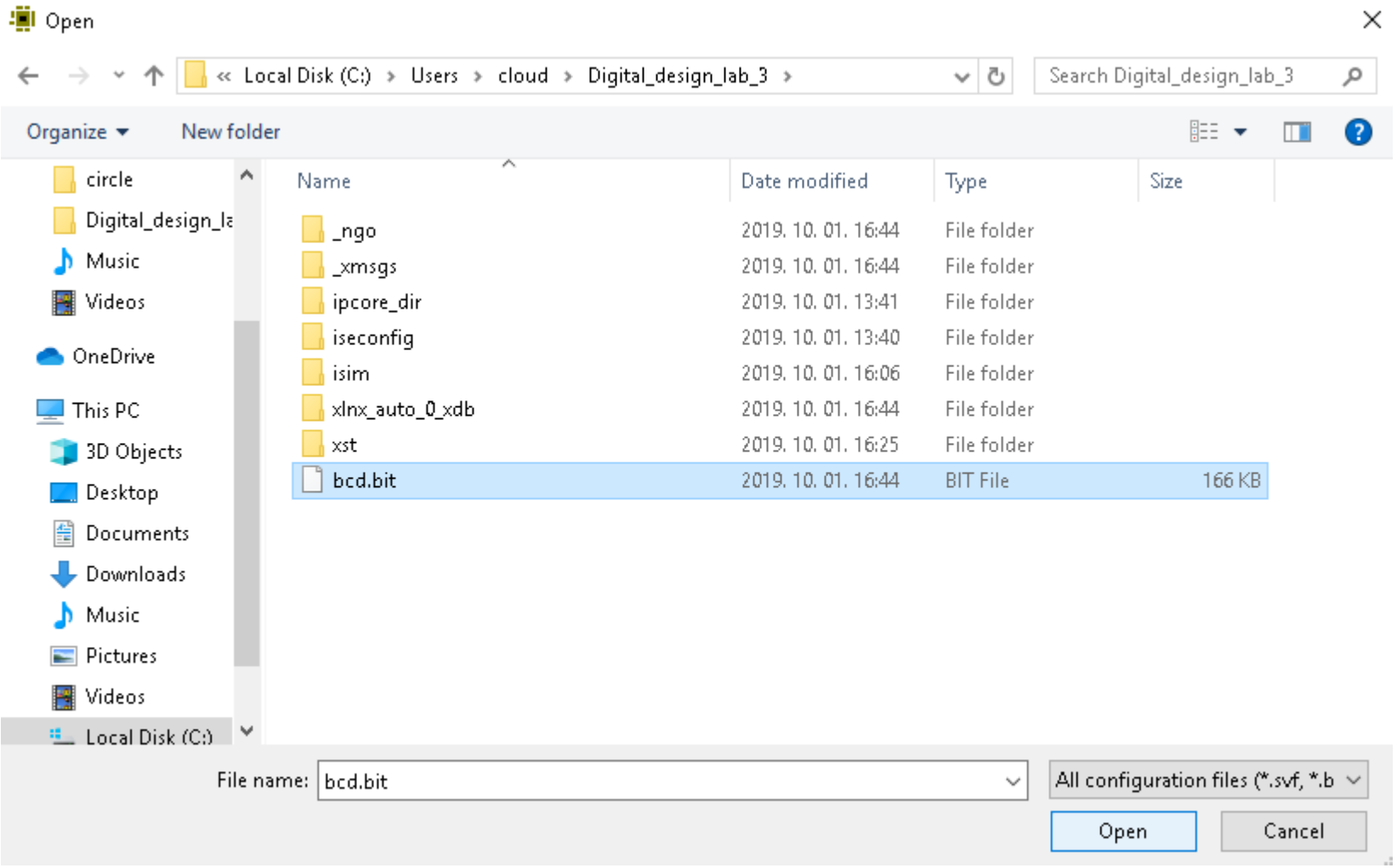


- Then press „Configure the Selected Device”



BCD code checker

- Browse the bcd.bit file
- Press Open



BCD code checker

- Try the inputs from 0 to 15 using the switches (4 on the right side). Is the output set correctly for inputs higher than 9?

BCD code checker

- Try to modify the circuit on your own: the output on ld[0] should be 1, if the 8 bit input is an invalid BCD code.
- Help: sw[7:0] can be converted to a 2 digit hexadecimal number:
  - sw[3] sw[2] sw[1] sw[0]: ABCD
  - sw[7] sw[6] sw[5] sw[4]: EFGH
- So the error output is 1 if ABCD is greater than 9 OR EFGH is greater than 9

Fibonacci number detector

- Try to modify the circuit on your own to implement a Fibonacci number detector.
- You can find additional information on Fibonacci numbers here: https://www.mathsisfun.com/numbers/fibonacci-sequence.html
- Since the board has an 8-bit input, the following Fibonacci numbers can be detected: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233
- Design a circuit that outputs 1 on ld[0] if the input on the switches is a Fibonacci number
- Help: implement a minterm for each Fibonacci number
- sw[0] represents 2^0, sw[1] is 2^1, …, and sw[7] represents 2^7