

Digital design laboratory 1

Introduction

- The goal of the laboratory is to provide an opportunity to use and test the theoretical concepts you studied on lecture on a real hardware.
- For this purpose, we are going to use an FPGA board
- FPGA stands for Field-programmable gate array
- Fundamentally, the FPGA consists of a large number of logic blocks, and the way these blocks are wired together can be defined by the user
- In addition, the functionality of the logic blocks can be defined also

Introduction

- For example, a logic block can be used as a memory component, or it can be used to do mathematical operations (e.g. adding two numbers).
- FPGA-s can be configured using a hardware description language (HDL).
- These languages have their own syntax, but it's often quite similar to traditional computer programming languages.

Introduction

- However, there is one very important difference between computer programming languages and hardware description languages.
- In CPLs, the instructions are executed **sequentially**, one-by-one.
- In HDLs, the instructions are executed **parallely**, more at the same time.
- Let's demonstrate this with a simple example.

Introduction

- Take a look at the following C code:

```
int a = 0, b = 0;
while (1) {
    a = a + 1;
    b = a + 1;
}
```

- First the memory is allocated for **a** and **b**, then the value of **a** is updated with **a+1**, then **b**'s value is set to **a+1**.

Introduction

- After the first cycle the the value of **a** will be 1 and the value of **b** will be 2.
- In the n th cycle **a** will be n and **b** will be $n+1$.
- The assignment of **a** always precedes the assignment of **b**, so these two variables will never have the same value at the end of the cycles.

Introduction

- Take a look at the following Verilog code. Do not worry about the syntax (it is not important now), this is the Verilog version of the above C code.

```
reg [31:0] a; // 32 bit registers for storing
reg [31:0] b; // integer data
always @ (posedge clk) // some condition, not important
begin
    a <= a + 1;
    b <= a + 1;
end
```

- Since Verilog evaluates the assignment simultaneously, **a** and **b** registers will always have the same value.

Introduction to Verilog

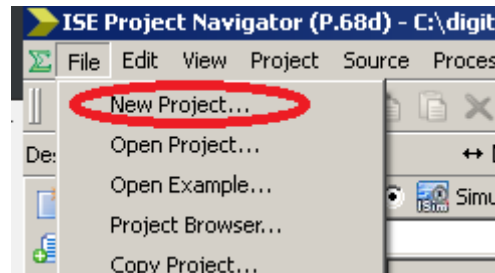
- The ISE Design Suite 14.6 software will be used on the laboratories to design and implement digital circuits.
- To launch the application, click on the icon on the desktop:



- Or launch it from the Start menu:
 - Start -> All Programs -> Xilinx Design Tools -> ISE Design Suite 14.6 -> ISE Design Tools -> Project Navigator

Introduction to Verilog

- After the navigator has started, it is time to create your first project.
- Select File, then New Project...



- The New Project Wizard appears. Next slide shows how to fill the textboxes.
- General rule: always work on the D: drive!

Introduction to Verilog

- Name: Digital_design_lab_1
- Location and Working Directory: D:\Digital_design_lab_1
- Description: this can be empty
- Top-level source type: HDL
- If everything is set, press Next.

New Project Wizard

Create New Project
Specify project location and type.

Enter a name, locations, and comment for the project

Name: Digital_design_lab_1

Location: D:\Digital_design_lab_1

Working Directory: D:\Digital_design_lab_1

Description:

Select the type of top-level source for the project

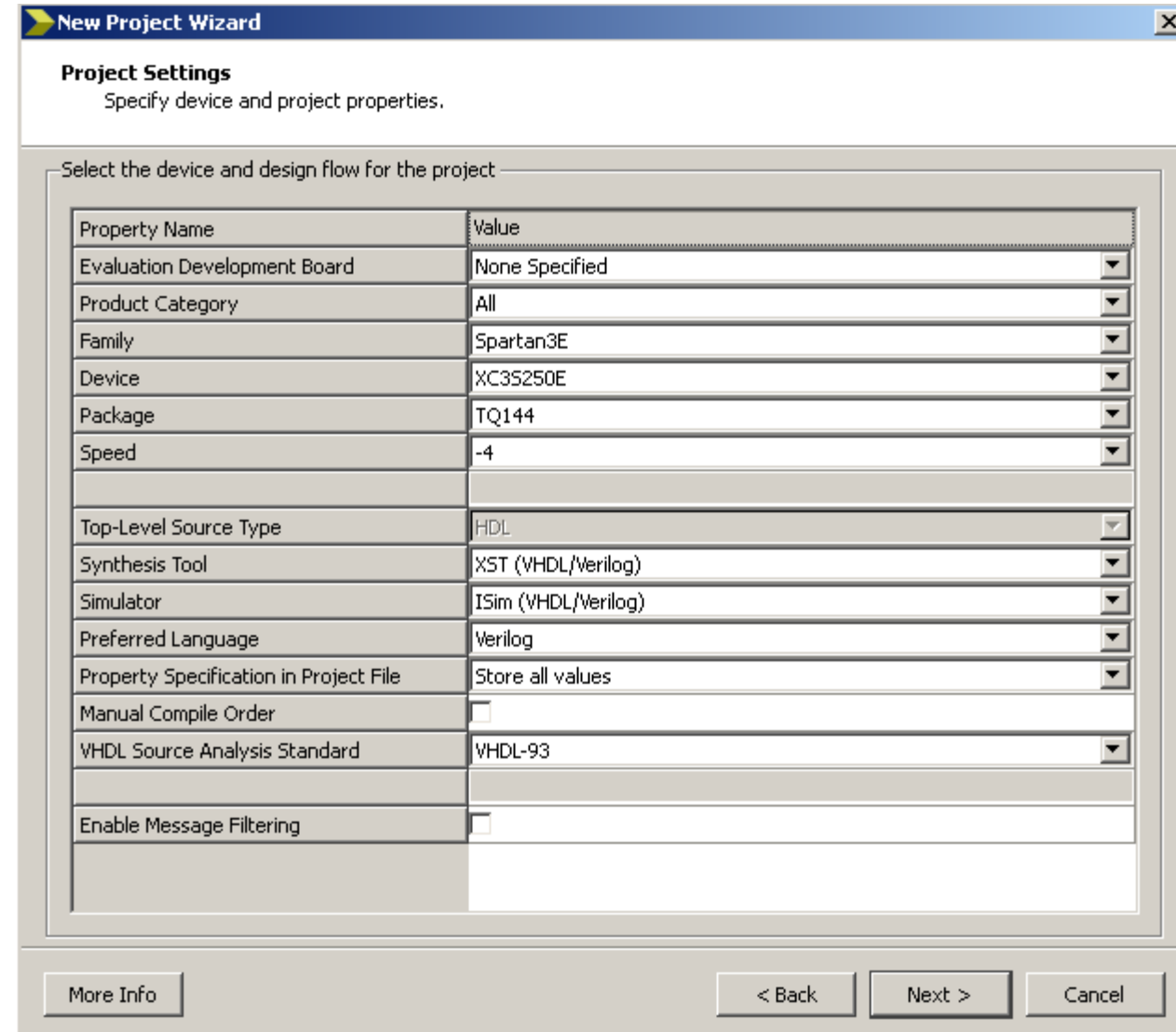
Top-level source type: HDL

More Info Next > Cancel

Introduction to Verilog

- Set the following:
- Family: Spartan3E
- Device: XC3S250E
- Package: TQ144
- Speed: -4

- Do not modify the other settings!
- Press Next



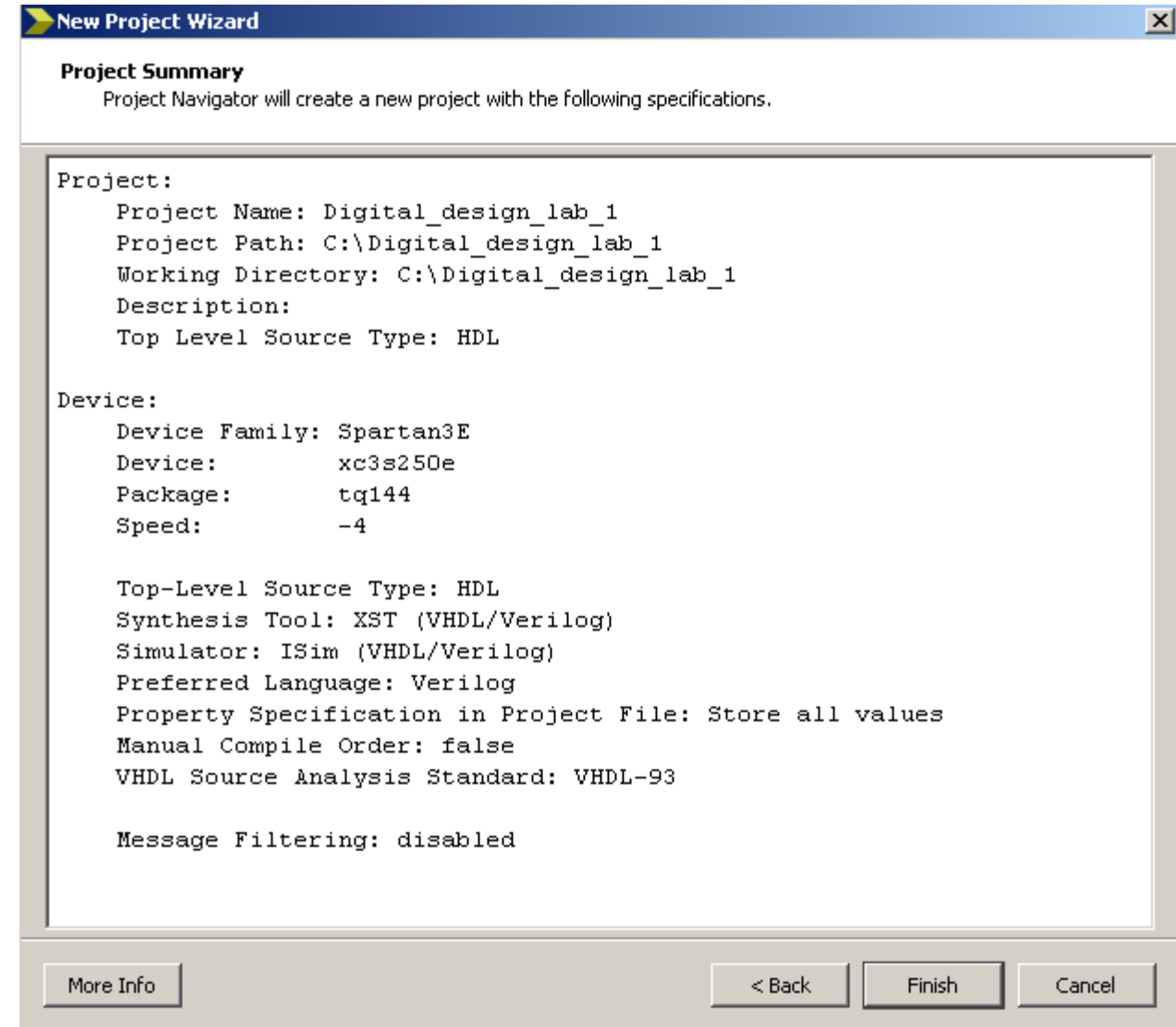
The screenshot shows the 'New Project Wizard' dialog box, specifically the 'Project Settings' step. The title bar reads 'New Project Wizard' with a close button. Below the title bar, the text 'Project Settings' is followed by the instruction 'Specify device and project properties.' The main area contains a table with the following settings:

Property Name	Value
Evaluation Development Board	None Specified
Product Category	All
Family	Spartan3E
Device	XC3S250E
Package	TQ144
Speed	-4
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	Verilog
Property Specification in Project File	Store all values
Manual Compile Order	<input type="checkbox"/>
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	<input type="checkbox"/>

At the bottom of the dialog, there are three buttons: 'More Info', '< Back', and 'Next >', and a 'Cancel' button on the far right.

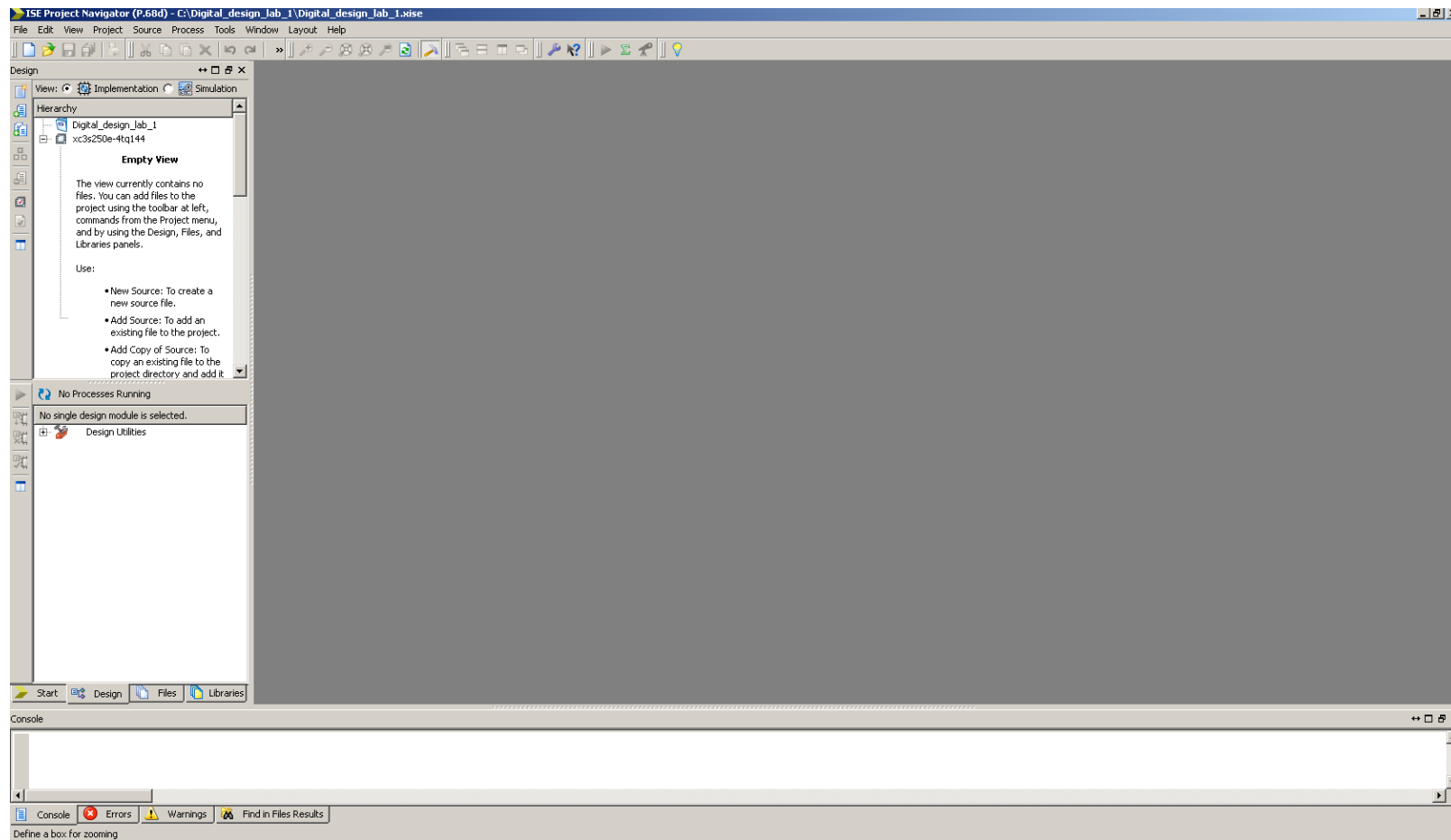
Introduction to Verilog

- The following window appears:
- Just press Finish



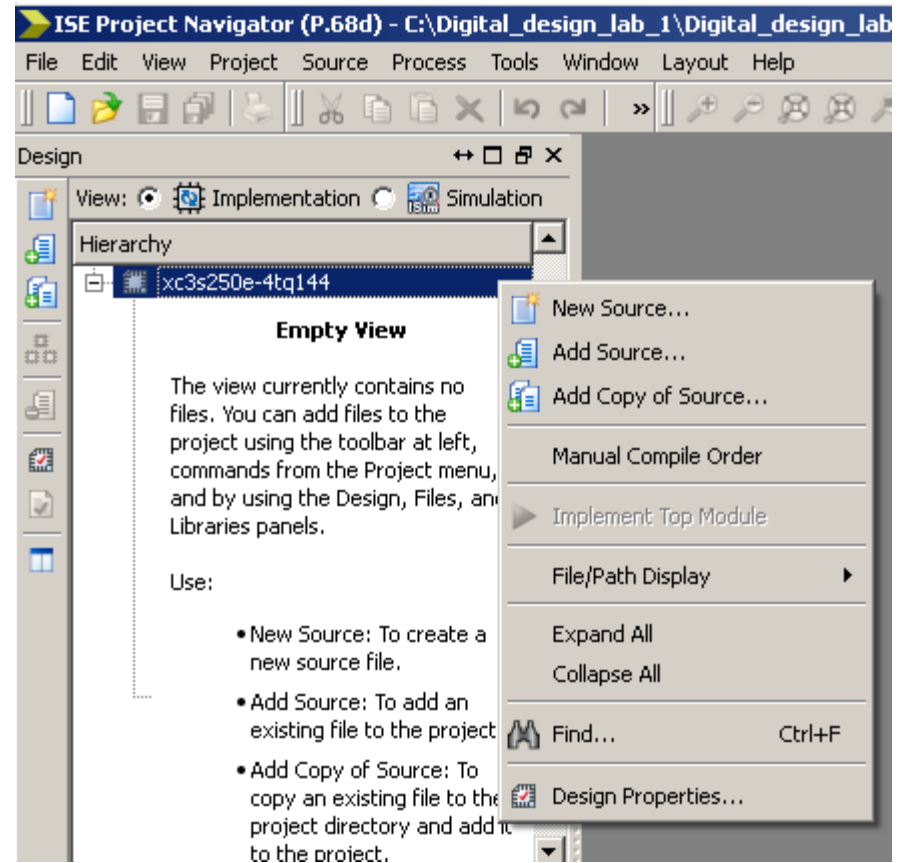
Introduction to Verilog

- The following screen appears:



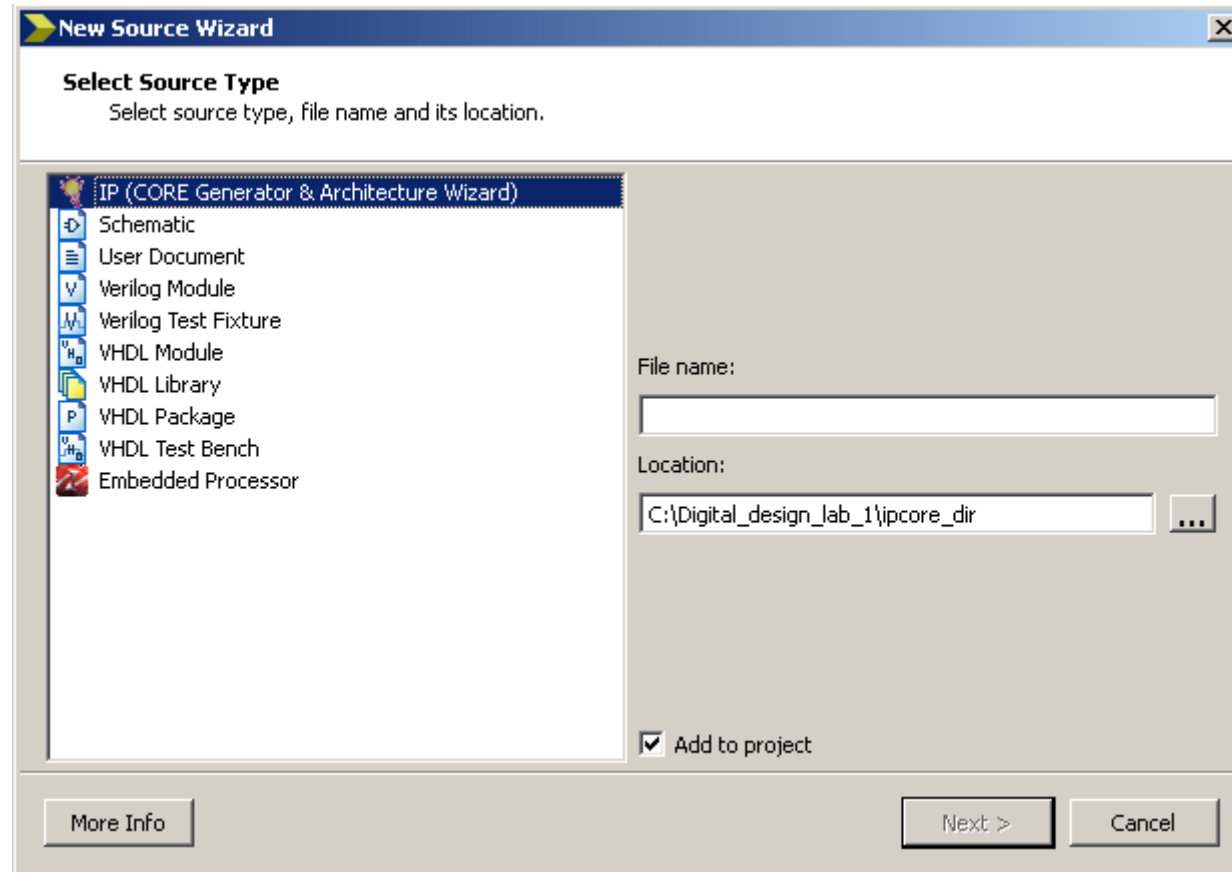
Introduction to Verilog

- To get started, right click on the xc3s250e-4tq144 label, and select New source... from the menu.



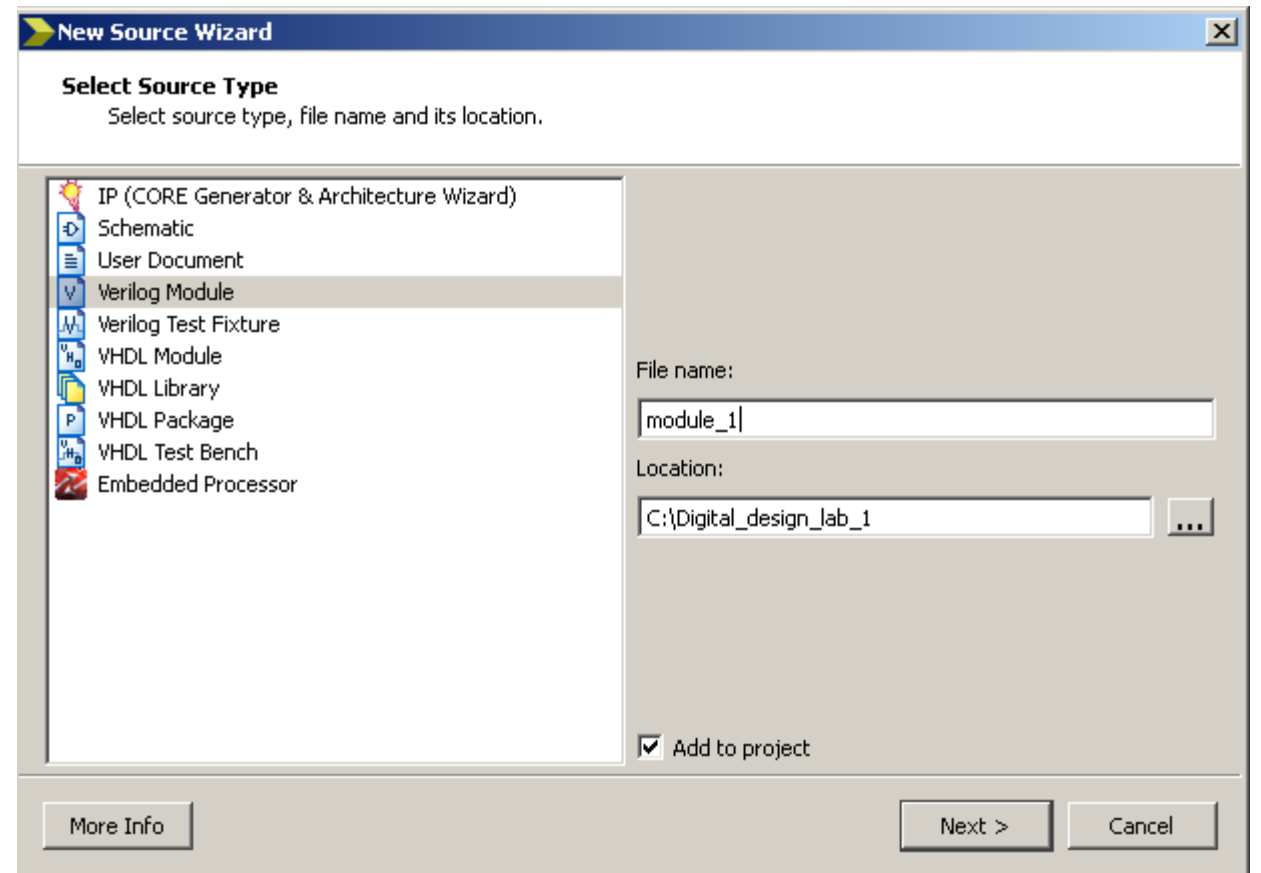
Introduction to Verilog

- The following window appears:



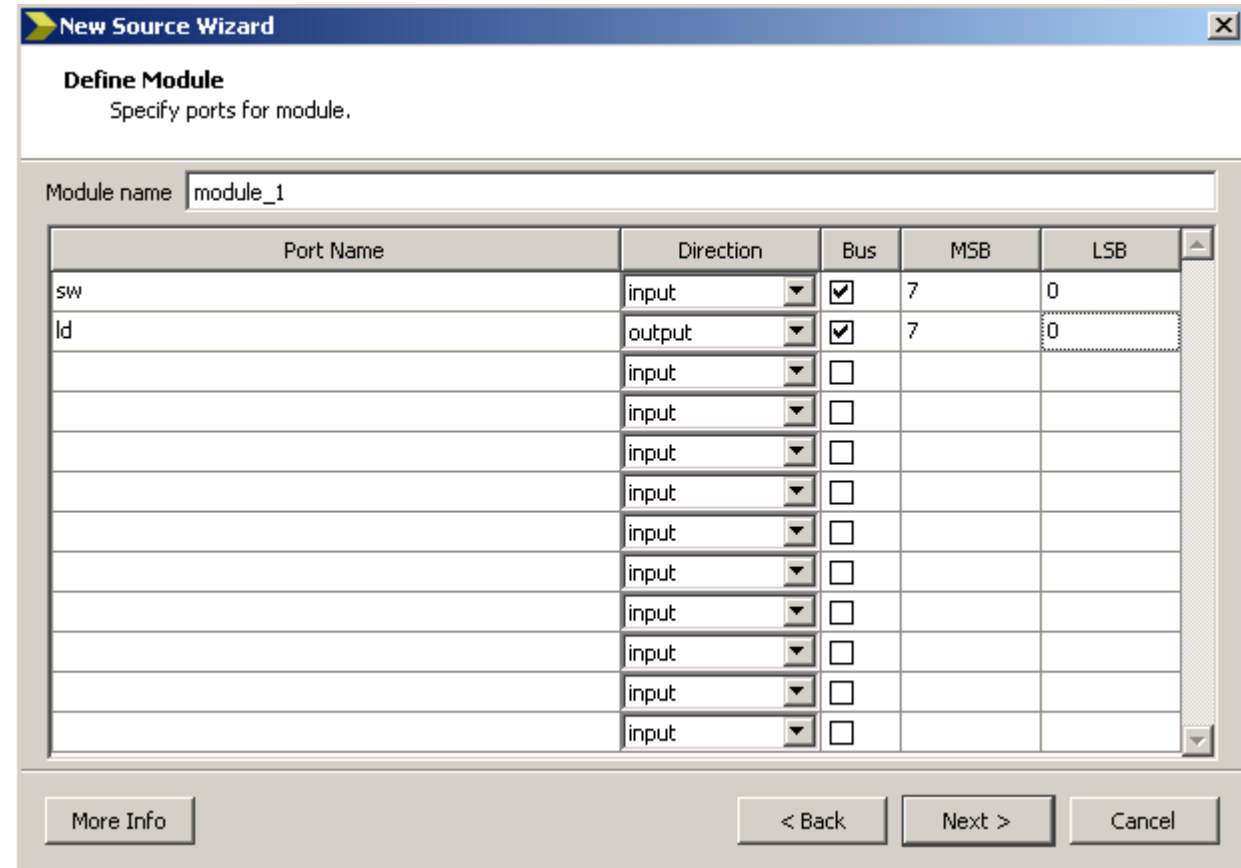
Introduction to Verilog

- Select Verilog Module
- File name: module_1
- Make sure the 'Add to project' checkbox is selected
- Do not modify the Location
- Press Next



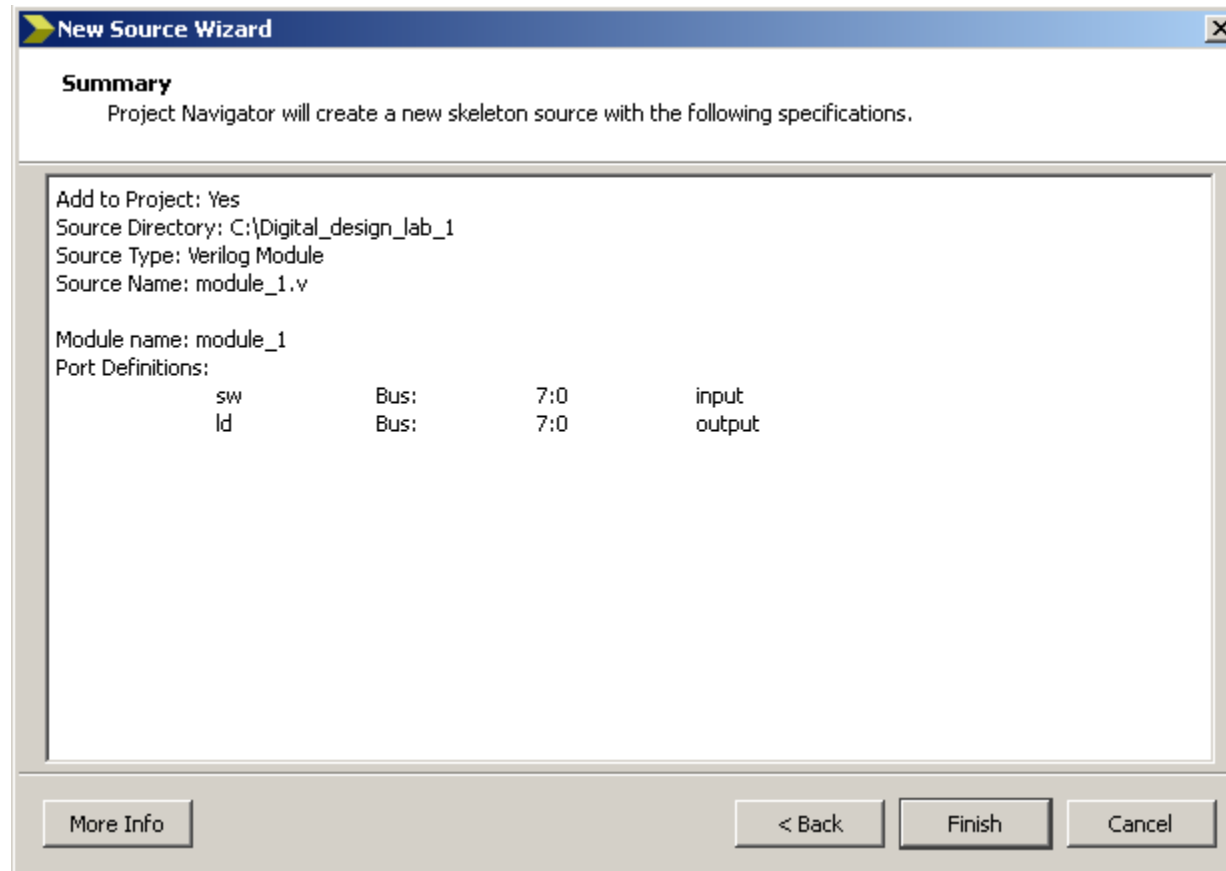
Introduction to Verilog

- The New Source Wizard appears:
- Add to new ports: sw, ld
- Set the direction of sw to input
- Set the direction of ld to output
- Check the Bus checkbox for both ports
- Set the value of the MSB to 7, and the value of the LSB to 0 for both ports
- Press the Next button



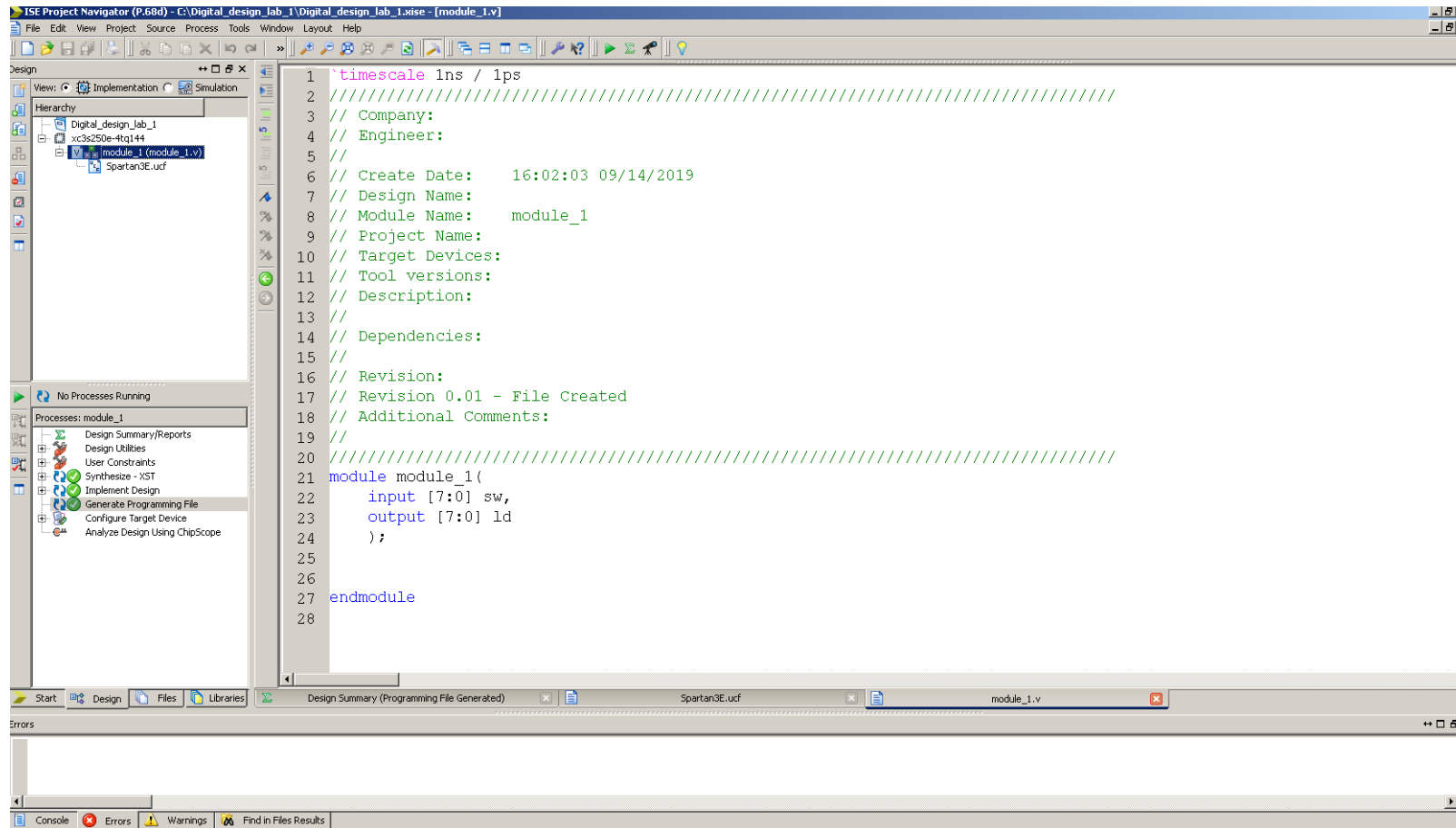
Introduction to Verilog

- The following window appears:
- Press Finish



Introduction to Verilog

- The ISE software generates the editable source file for the project:

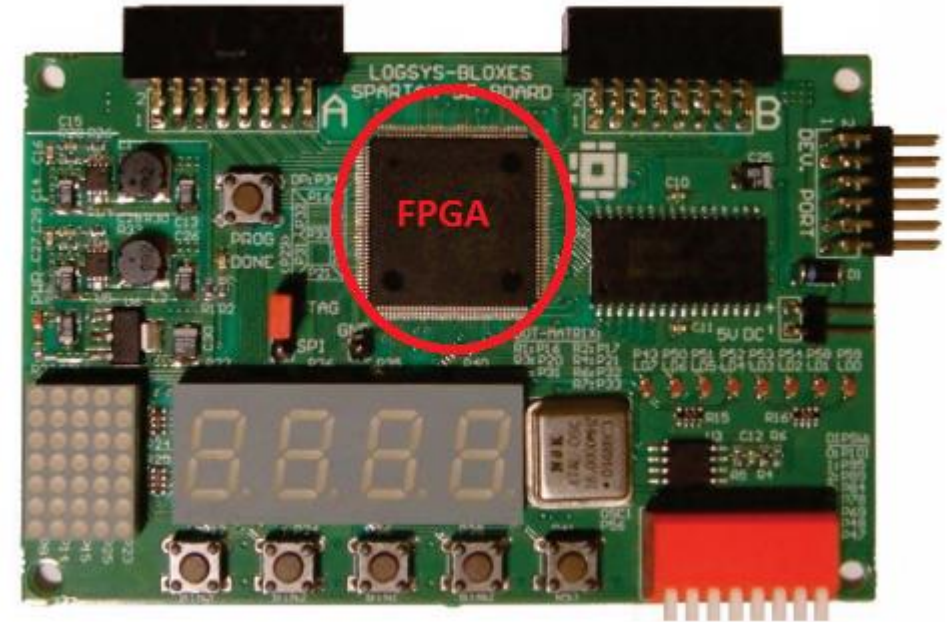


The screenshot displays the ISE Project Navigator software interface. The main window shows a Verilog source file for a module named 'module_1'. The code includes a timescale, a header with project information, and a module definition with two ports: 'sw' (input) and 'ld' (output).

```
1 `timescale 1ns / 1ps
2 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////
3 // Company:
4 // Engineer:
5 //
6 // Create Date:    16:02:03 09/14/2019
7 // Design Name:
8 // Module Name:    module_1
9 // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////
21 module module_1(
22     input [7:0] sw,
23     output [7:0] ld
24 );
25
26
27 endmodule
28
```

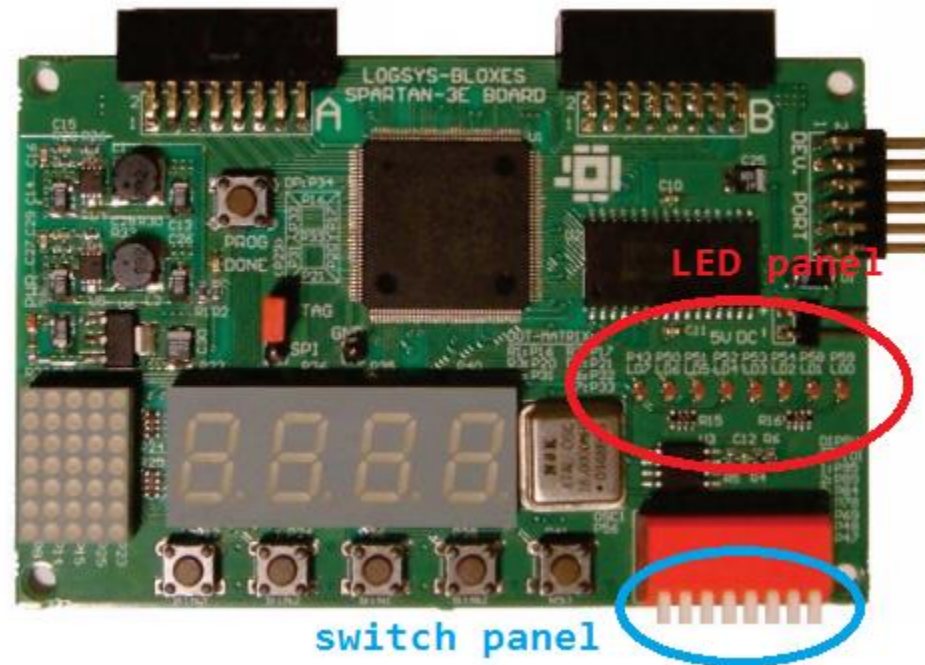
Introduction to Verilog

- Now you can create your first “Hello World” module.
- After you have finished with the code, you can test your module on an FPGA board:
- The FPGA itself is the black, square shaped device in the middle of the board



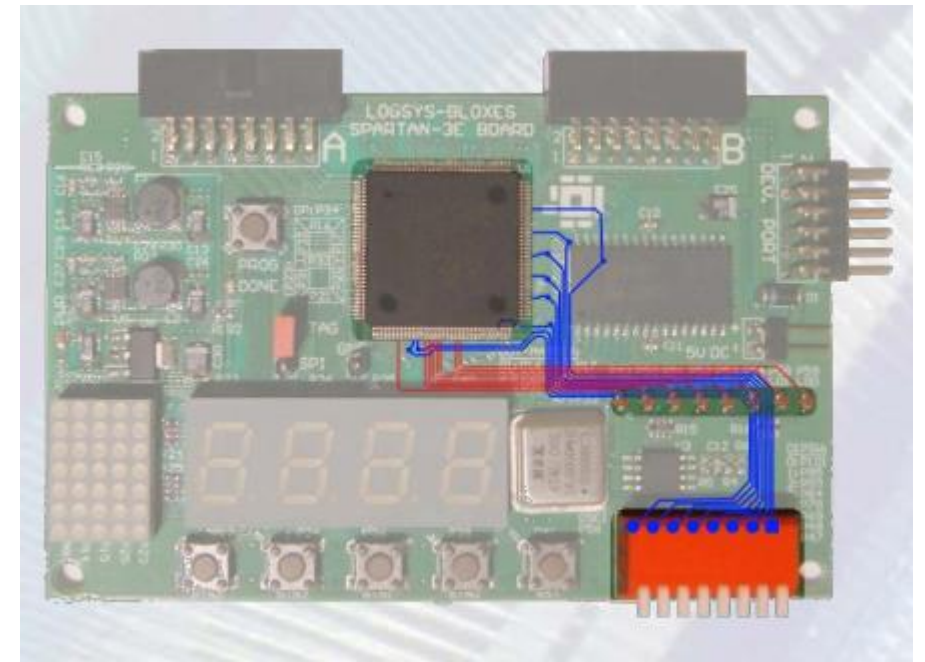
Introduction to Verilog

- In this first design, you are going to control the LED panel of the FPGA using the switches: if a switch is turned on, the corresponding LED will light up.



Introduction to Verilog

- We need to define a connection between the LEDs and the switches using Verilog. The aim is to connect them one-by-one.
- The LEDs are connected to the outputs (red wires) of the FPGA, while the switches are connected to the inputs (blue wires).
- In the implementation, we will connect these inputs and outputs “inside” the FPGA



Introduction to Verilog

- To do so, add the following line to your module:

```
19 //  
20 //////////////////////////////////////  
21 module module_1(  
22     input [7:0] sw,  
23     output [7:0] ld  
24 );  
25  
26     assign ld = sw;  
27  
28 endmodule  
29
```

- Don't forget to press the save button 

Introduction to Verilog

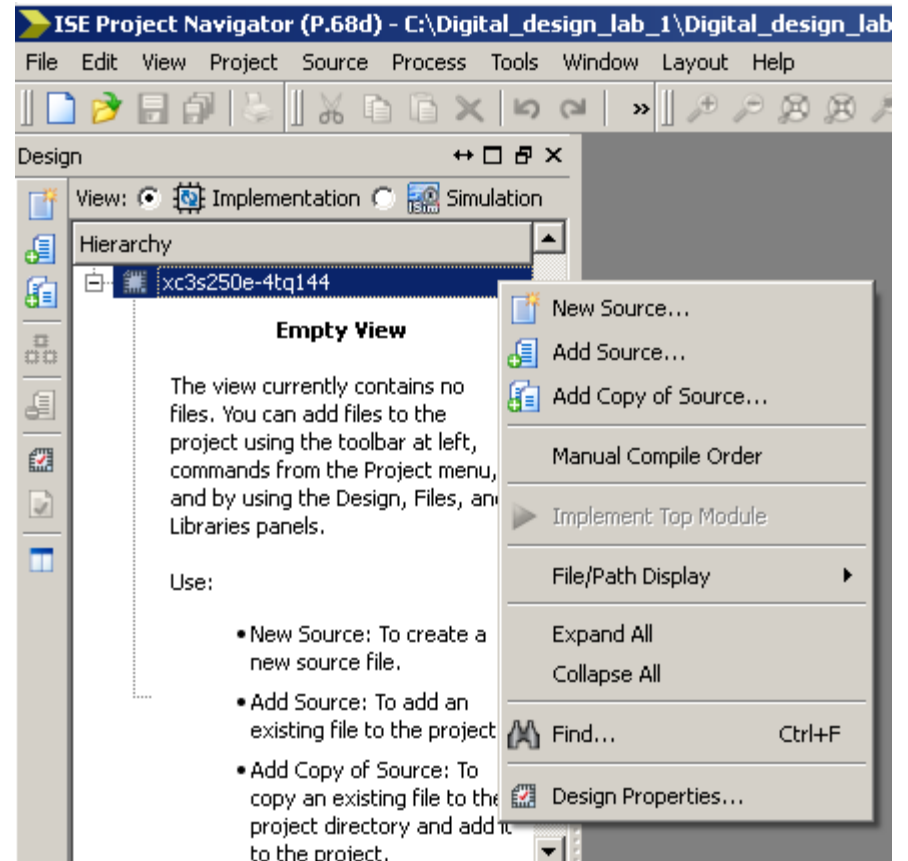
- As a last step, you have to define the connection between your own port names (ld and sw), and the port pins of the FPGA.
- Take a close look at the LEDs of the board. You can see the matching of the LEDs and the corresponding ports of the FPGA. For example, LD0 is connected to P59, LD1 is connected to P58, and so on.
- The name of the leds in our module is ld, which is an 8 bit output. So we have to define that ld[0] is connected to port P59, ld[1] is connected to port P58, and the same for the rest of the leds (up to ld[7], the 8th led).

Introduction to Verilog

- The same definition has to be done also for the switches. Check the lower right corner of the board.
- You can see the matchings between the switches and the input ports: sw[0] is connected to port P101, sw[1] connects to P95, and so on.
- We need to provide this information somewhere in our project before we generate the programming file and download it to the board.
- For this purpose, we have to add a .UCF file (User Constraints File) to the project.

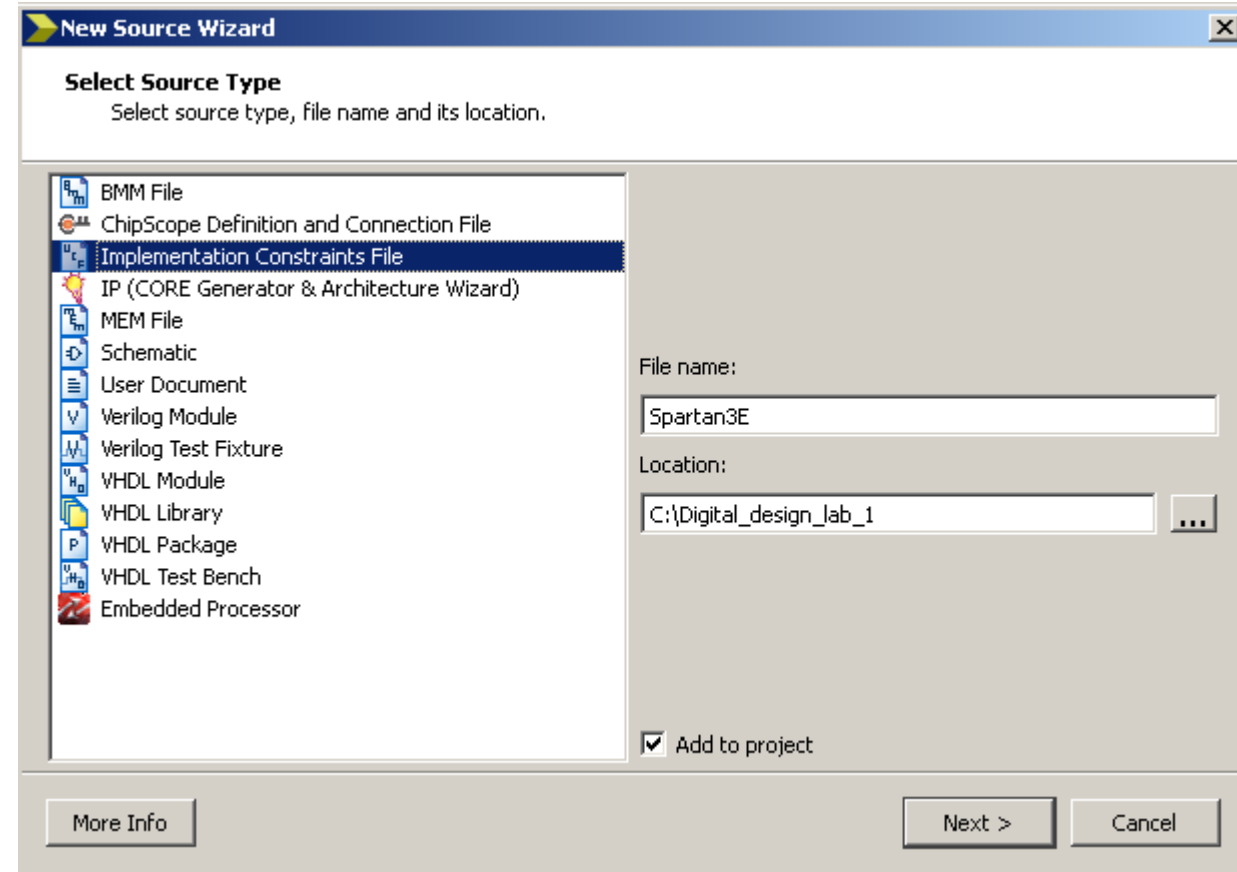
Introduction to Verilog

- Right click on “xc3s250e-4tq144” in the top left corner of the ISE window, and select New Source.



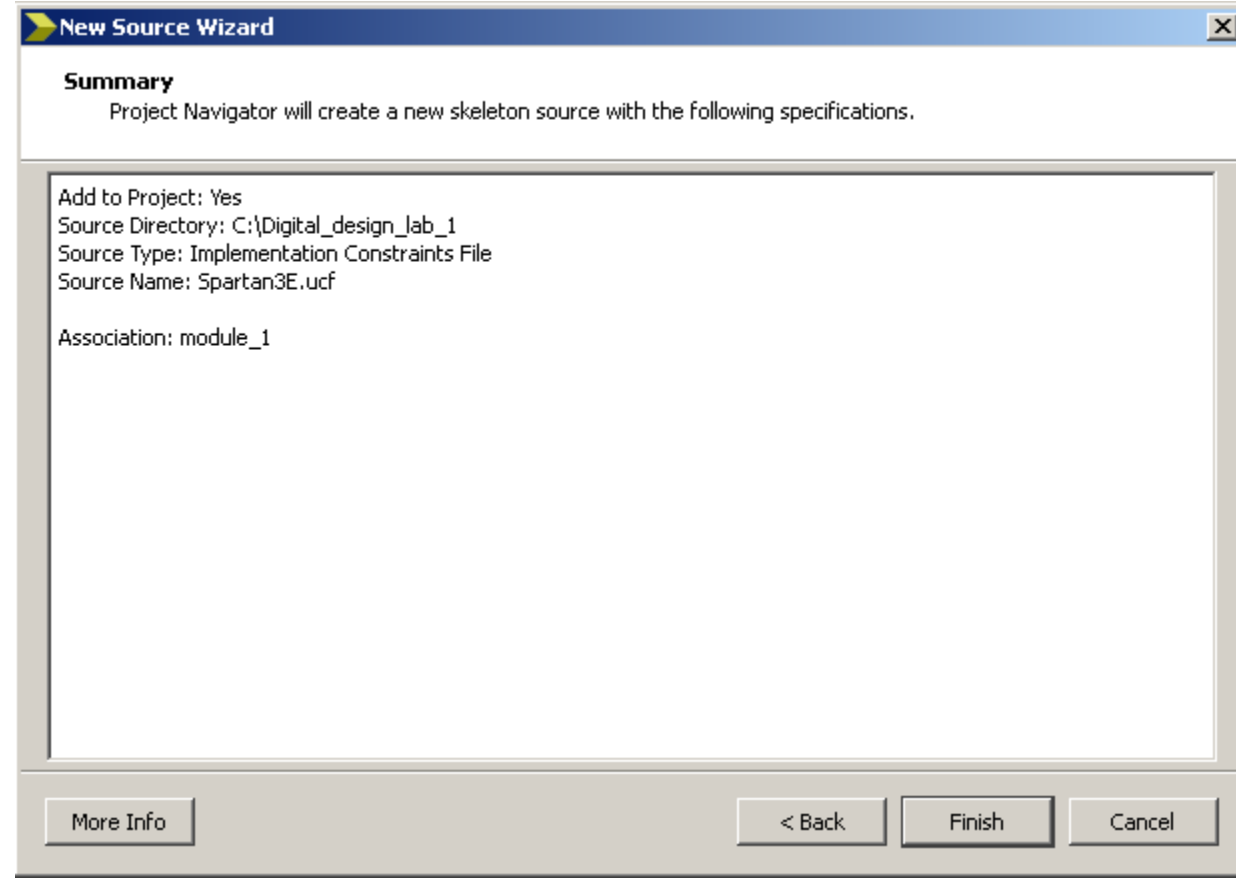
Introduction to Verilog

- In the next window, select Implementation Constraints File
- File name: Spartan3E
- Do not edit the Location
- Make sure the Add to project checkbox is set
- Press Next




Introduction to Verilog

- On the next window, press Finish



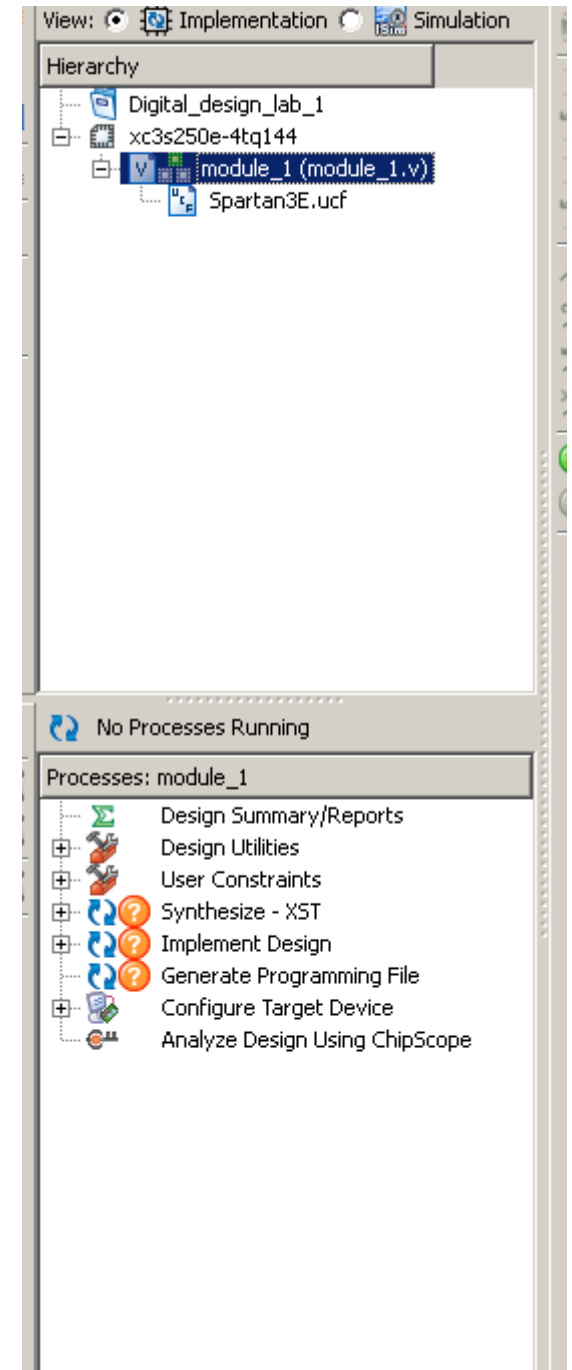
Introduction to Verilog

- Add the following lines to the UCF file:
- Note: some of the port names after LOC were replaced with question marks.
- Check the board and define the correct missing ports!
- Don't forget to press the save button. 

```
b_1\Digital_design_lab_1.xise - [Spartan3E.ucf]
Window Layout Help
>>
1 # 8 switches, from the left to the right
2 NET "sw<7>" LOC = "P47";
3 NET "sw<6>" LOC = "P48";
4 NET "sw<5>" LOC = "P69";
5 NET "sw<4>" LOC = "???" ;
6 NET "sw<3>" LOC = "P84";
7 NET "sw<2>" LOC = "P89";
8 NET "sw<1>" LOC = "???" ;
9 NET "sw<0>" LOC = "P101";
10
11 # 8 LEDs, from the left to the right
12 NET "ld<7>" LOC = "P43";
13 NET "ld<6>" LOC = "P50";
14 NET "ld<5>" LOC = "???" ;
15 NET "ld<4>" LOC = "P52";
16 NET "ld<3>" LOC = "P53";
17 NET "ld<2>" LOC = "P54";
18 NET "ld<1>" LOC = "P58";
19 NET "ld<0>" LOC = "???" ; |
```

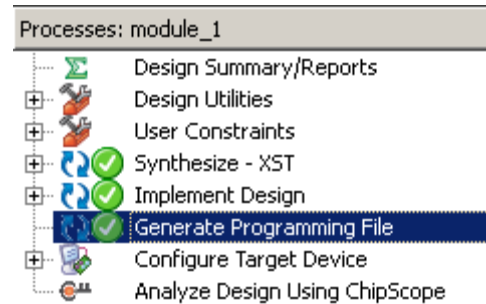
Introduction to Verilog

- Now you are ready to generate the so-called programming file
- To do so, left click on module_1 in the top left corner of the ISE
- On the lower side, the following options appear:
- Right click on Generate programming file, and select Run in the pop up menu.



Introduction to Verilog

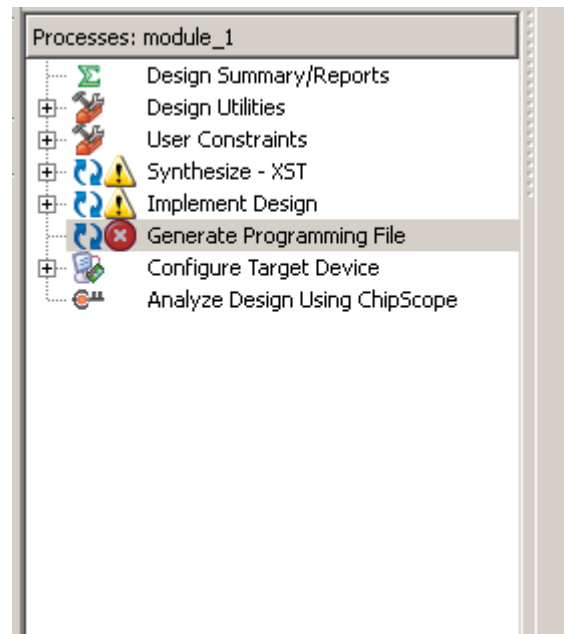
- If everything was OK, you should see 3 check marks:



- Check your working directory, you should see a new file named "module_1.bit".

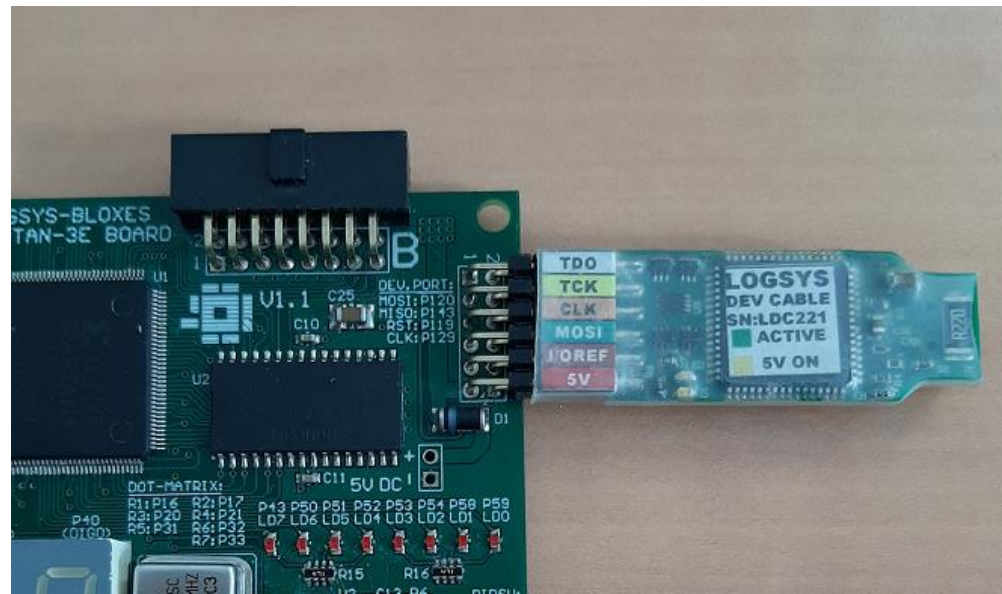
Introduction to Verilog

- However, if you see anything but the 3 check marks, ask for assistance:



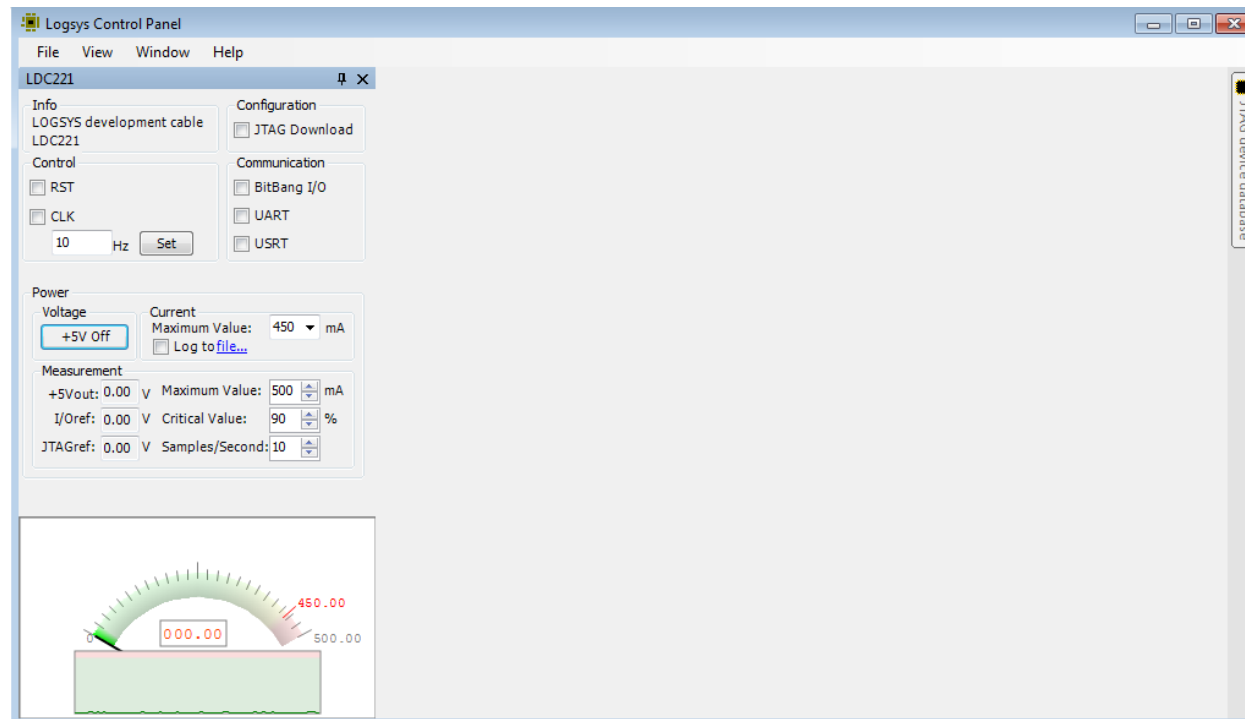
Introduction to Verilog

- Before uploading the generated programming file, the board needs some setup.
- First of all, connect the JTAG connector to the board. The labels (TDO, TCK, CLK, MOSI, I/OREF and 5V) should be oriented the same way as on the following picture, otherwise you might harm the device!



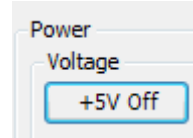
Introduction to Verilog

- Connect the USB cable to the JTAG connector.
- Launch the Logsys GUI application

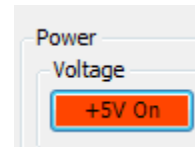


Introduction to Verilog

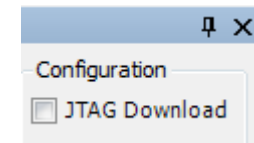
- Press the +5V Off button



- After pressing it, it turns into red:

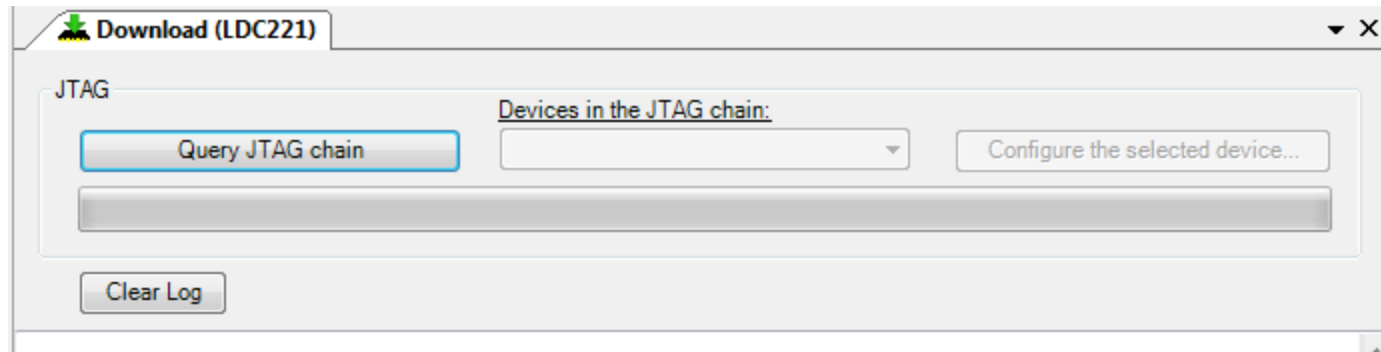


- Select the JTAG download checkbox under Configuration:



Introduction to Verilog

- The following options appear on the right side of the screen:



- Press the Query JTAG chain button, the “Configure the selected device” button becomes active on the right.
- Press it, and browse the “module_1.bit” file from your working directory.
- After selecting the file, press Open

Introduction to Verilog

- After the programming is complete (green status bar), you can try turning the LEDs on and off by flipping the switches.
- Try to set the following decimal and hexadecimal number assuming LD7, LD6, ..., LD0 represent 2^7 , 2^6 , ..., 2^0 :
 - 68_{10}
 - 178_{10}
 - 68_{16}
 - $C7_{16}$