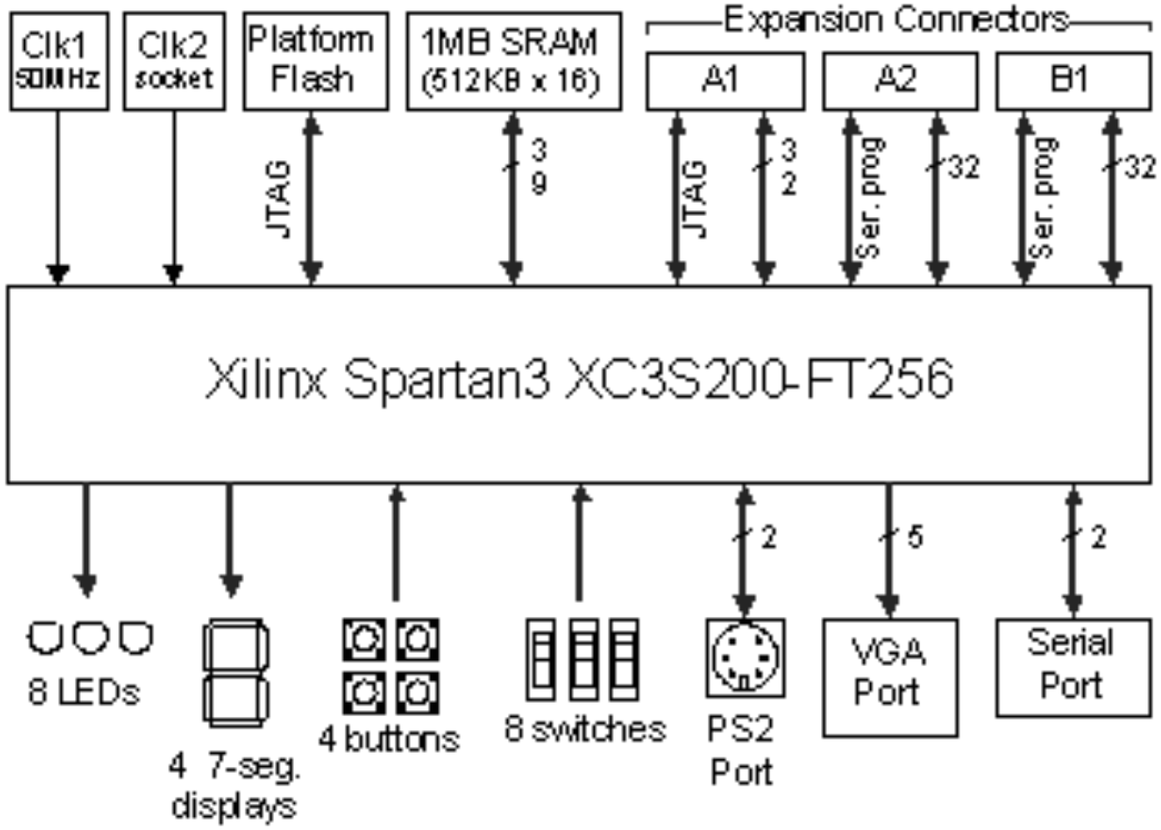
CPE 462
VHDL: Simulation and Synthesis

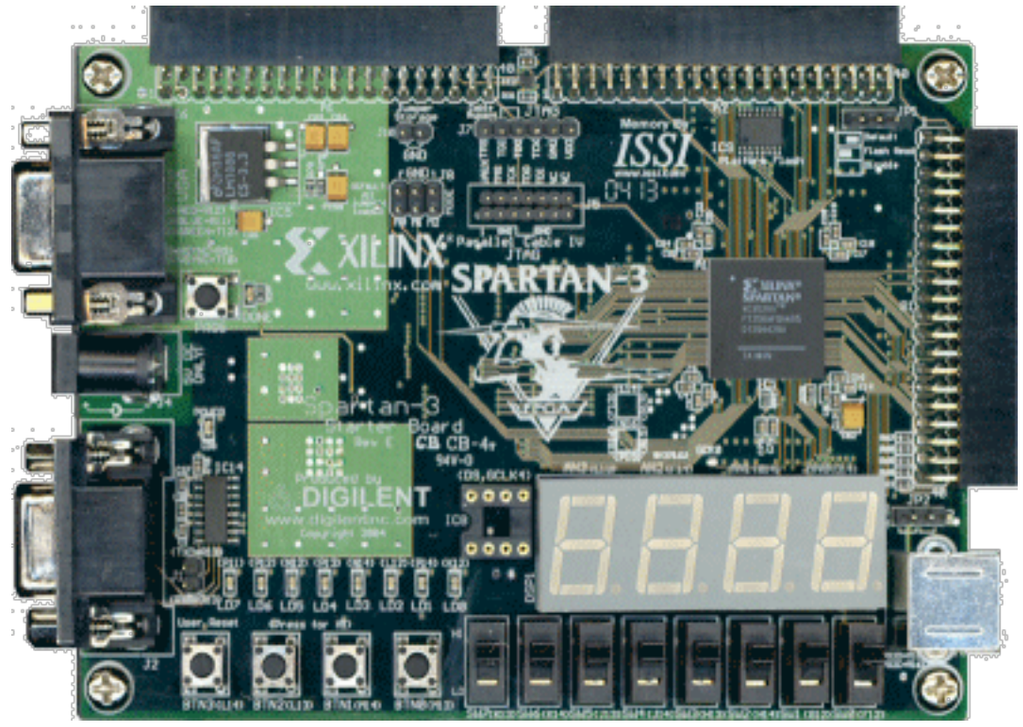
Topic #03 - b) Introduction to Xilinx development environment

Class software & hardware

- We are in the process of updating the hardware / software tools
- Our current set-up:
  - FPGA boards: Spartan 3
  - Software to load code onto FPGAs IDE: Xilinx ISE 7.1
  - Software to simulate and test VHDL: Aldec HDL
- You can install and run this software on your own computer (windows XP and up) to test and simulate your code.
- To deploy the code into FPGA boards you need to come to the lab. Our Spartan 3 boards can only be loaded with Windows XP.

Spartan-3 FPGA board

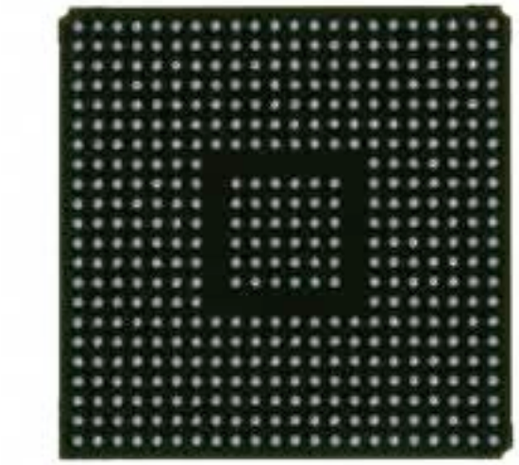


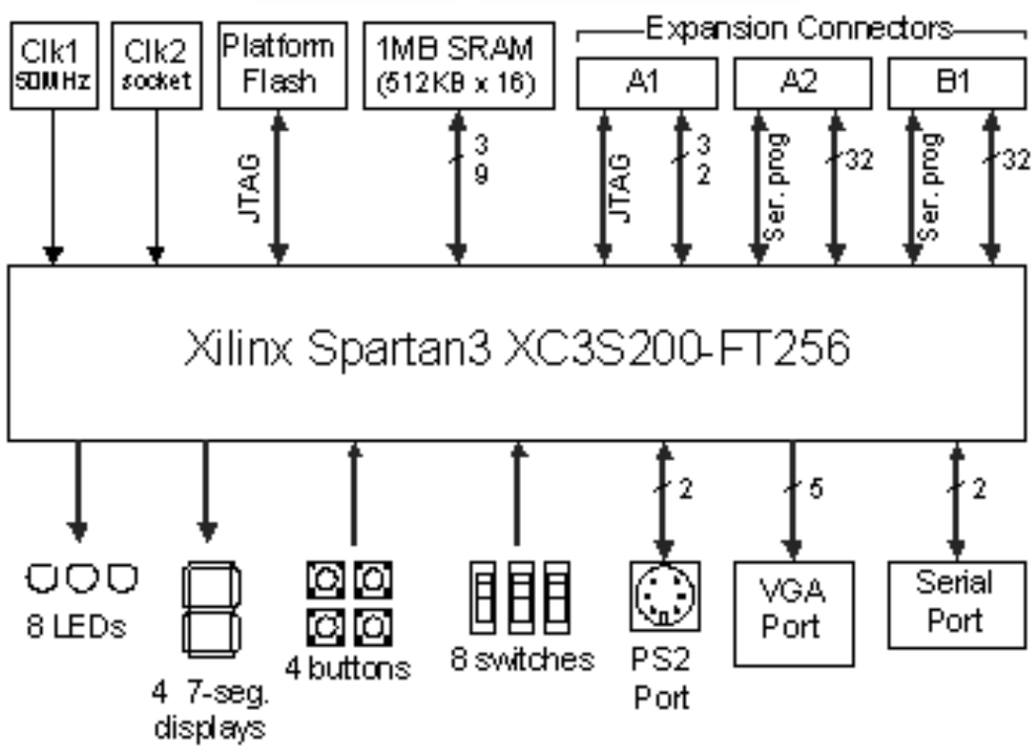


http://www.xilinx.com/support/documentation/boards_and_kits/ug130.pdf

Some useful pinouts

Back of FPGA chip is connected to a bunch of I/Os



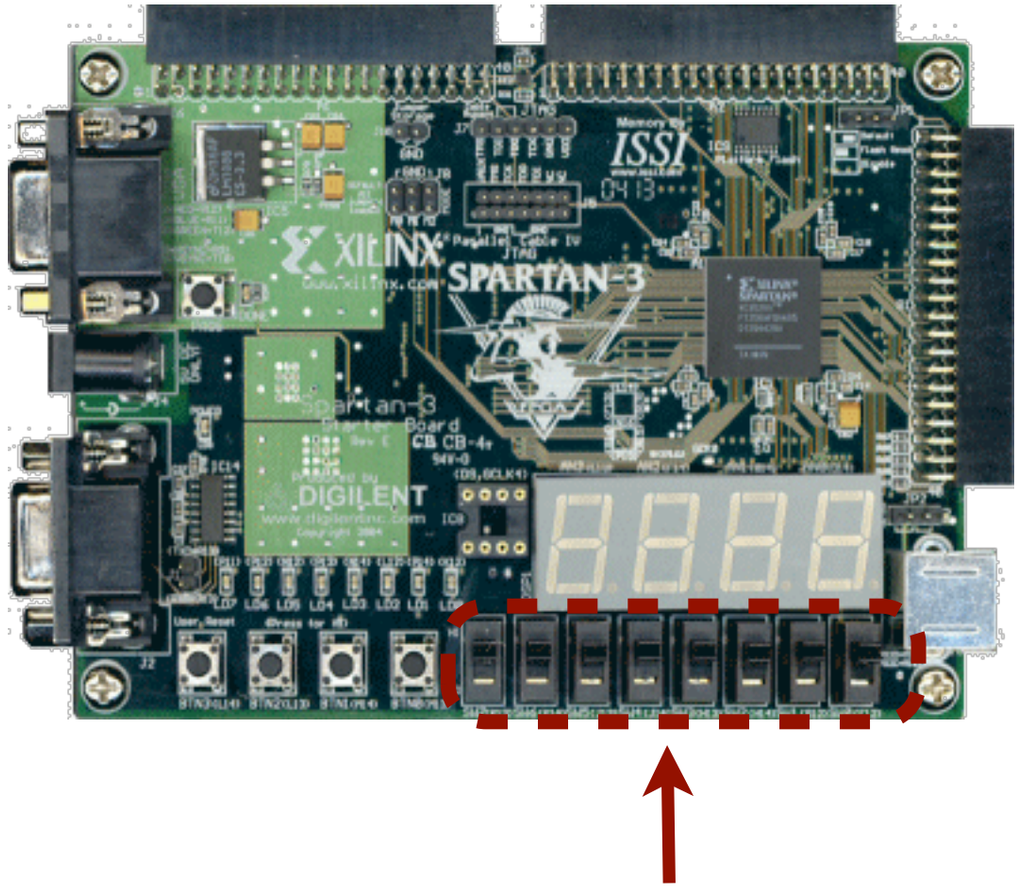


		Bank 0															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
A	GND	TDI	CPU_A2	CPU_A5	CPU_A8	VCCAUX	CPU_D1	CPU_D7	CPU_D11	CPU_D14	VCCAUX	DTACK	INIT_B	CPU_RST	TCK	GND	
B	IOB3_0	IOB3_1	CPU_A1	CPU_A4	VCCO_0	CPU_A12	CPU_D0	CPU_D6	GND	CPU_UDS	CPU_IPL2	VCCO_0	CPU_CLK	SPI_DOUT	TMS	15KHz	
C	IOB3_2	IOB3_3	CPU_A0	CPU_A3	CPU_A7	CPU_A11	CPU_A15	CPU_D5	CPU_AS	CPU_LDS	CPU_IPL1	SPI_CLK	SPI_DIN	TDO	RAM_D0	RAM_D1	
D	IOB3_4	USER_I10	PROG_B	VCCINT	CPU_A6	CPU_A10	CPU_A14	CPU_D4	CPU_D10	CPU_D13	CPU_IPL0	CPU_RW	VCCINT	RAM_D2	RAM_D3	CTS	
E	IOB3_5	VCCO_3	IOB3_6	IOB3_7	VCCINT	CPU_A9	CPU_A13	CPU_D3	CPU_D9	CPU_D12	CPU_D15	VCCINT	RAM_D4	RAM_D5	VCCO_2	RAM_D6	
F	VCCAUX	USER_I8	IOB3_8	IOB3_9	USER_I9	GND	VCCO_0	CPU_D2	CPU_D8	VCCO_0	GND	RAM_D7	RAM_D8	RAM_D9	RAM_D10	VCCAUX	
G	USER_I7	IOB3_10	IOB3_11	IOB3_12	IOB3_13	VCCO_3	GND	GND	GND	GND	VCCO_1	RXD	RAM_D11	RAM_D12	RAM_D13	RAM_D14	
H	USER_I6	GND	IOB3_14	IOB3_15	IOB3_16	IOB3_17	GND	GND	GND	GND	RAM_D15	RAM_A0	JOYB0	RAM_A1	RAM_A2	JOYB1	Bank 1
J	IOB3_18	IOB3_19	IOB3_20	IOB3_21	IOB3_22	USER_I5	GND	GND	GND	GND	JOYB2	JOYB3	RAM_A3	RAM_A4	GND	RAM_A5	
K	IOB3_23	IOB3_24	IOB3_25	USER_I4	IOB3_26	VCCO_3	GND	GND	GND	GND	VCCO_1	RAM_A6	RAM_A7	RAM_A8	RAM_A9	RAM_A10	
L	VCCAUX	IOB3_27	IOB3_28	IOB3_29	IOB3_30	GND	VCCO_2	MCLK	GREEN2	VCCO_2	GND	RAM_A11	RAM_A12	RAM_A13	RAM_A14	VCCAUX	
M	IOB3_31	VCCO_3	USER_I3	IOB3_32	VCCINT	IOB2_10	JOYA2	KBDDAT	GREEN3	RED1	JOYA4	VCCINT	JOYB4	JOYB5	VCCO_2	RAM_A15	
N	IOB3_33	USER_I1	USER_I2	VCCINT	IOB2_6	IOB2_9	IOB2_12	KBDCLK	BLUE0	RED2	JOYA3	PWRLED	VCCINT	RAMSEL0	RAMSEL1	RAMWE	
P	IOB3_34	IOB3_35	IOB2_0	IOB2_3	IOB2_5	IOB2_8	IOB2_11	MSDAT	BLUE1	RED3	HSYNC	VSYNC	TXD	RTS	RAMOE	RAMUB	
R	IOB3_36	IOB3_37	FPGASL2	IOB2_2	VCCO_2	IOB2_7	JOYA1	GND	BLUE2	GREEN0	RED0	VCCO_2	AUDIOL	CCLK	RAMLB	JOYB5	
T	GND	FPGASL0	FPGASL1	IOB2_1	IOB2_4	VCCAUX	JOYA0	MSCLK	BLUE3	GREEN1	VCCAUX	JOYA5	AUDIOR	DIN		GND	
		Bank 2															

Each I/O has a special FPGA pin name.

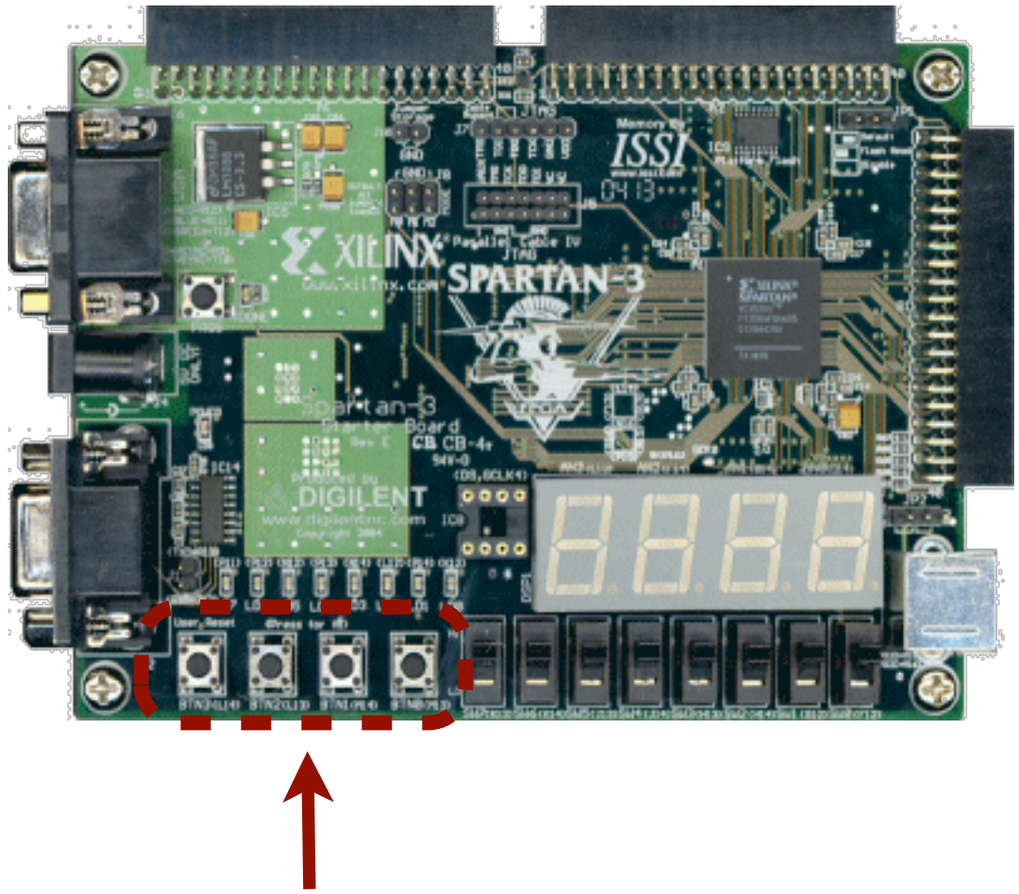
Nuno Alves (nalves@wne.edu), College of Engineering

Slider switch connections



Switch	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
FPGA Pin	K13	K14	J13	J14	H13	H14	G12	F12

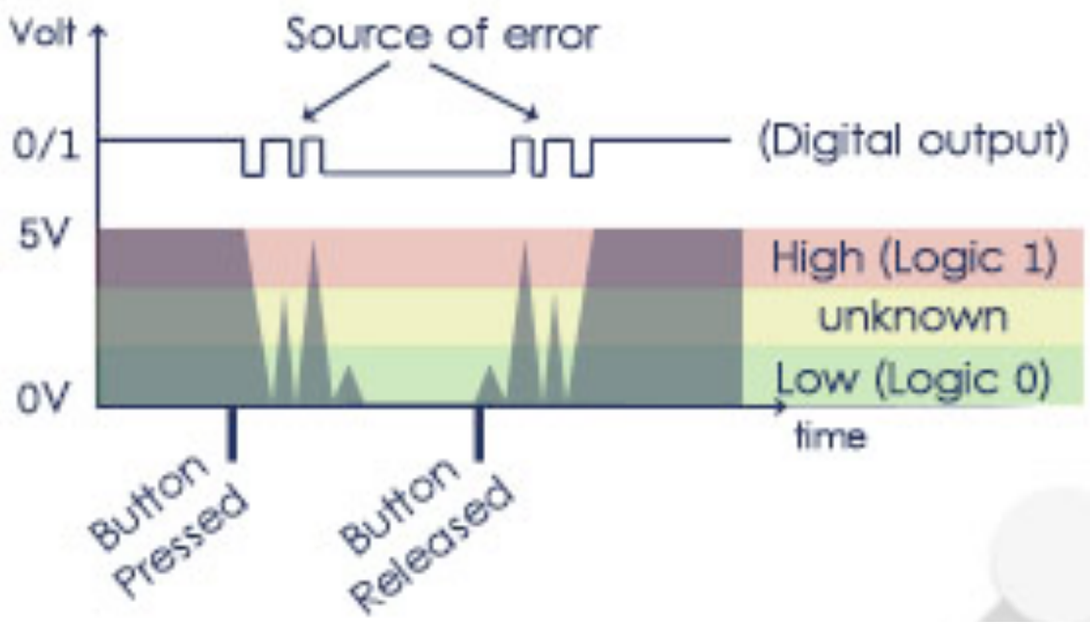
Push button connections

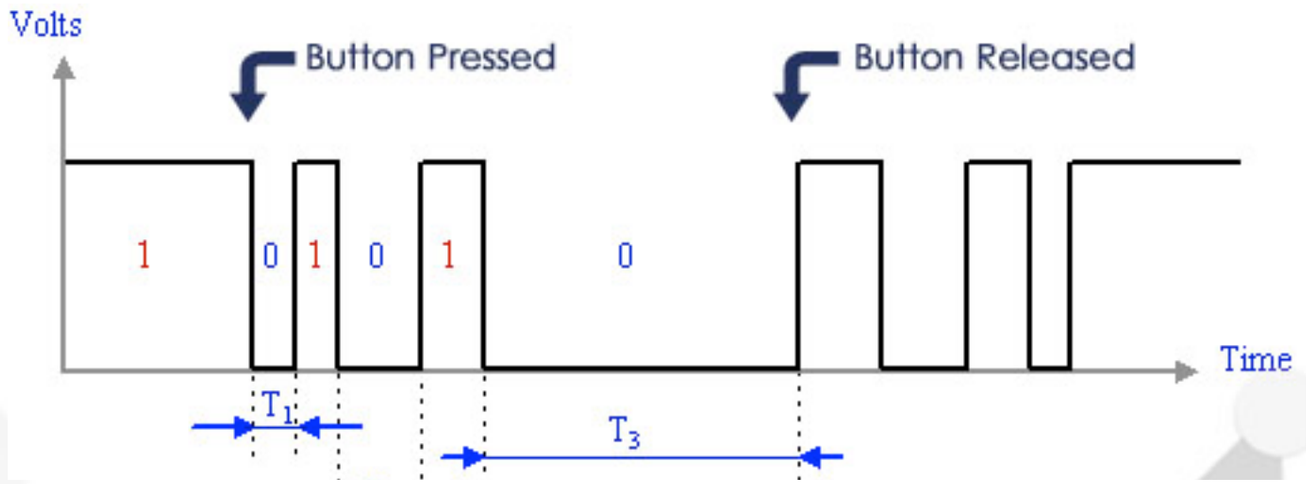


Push Button	BTN3 (User Reset)	BTN2	BTN1	BTN0
FPGA Pin	L14	L13	M14	M13

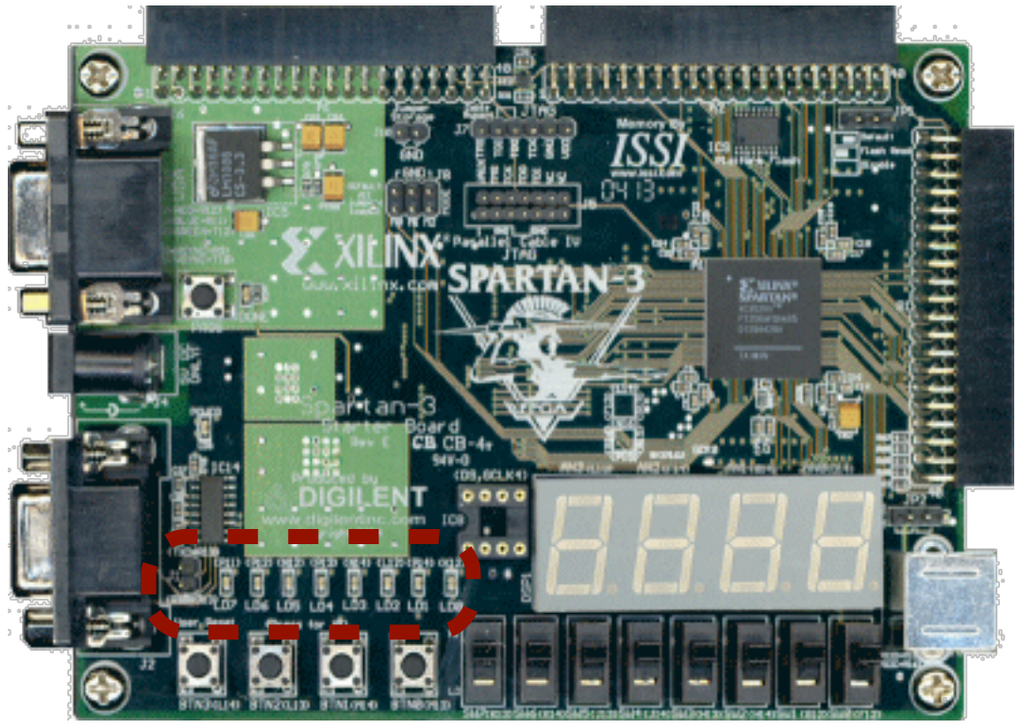
Keep in mind that these buttons are not de-bounced

- Push button switch has a spring
- When a switch is operated spikes of low and high voltages will always occur across that switch
- This is a series of High and Low signals that can by interpreted by a digital circuit as more that one pulse instead of one clean pulse or transition from a logic state to another





LEDs



LED	LD7	LD6	LD5	LD4	LD3	LD2	LD1	LD0
FPGA Pin	P11	P12	N12	P13	N14	L12	P14	K12

7-Segment Displays

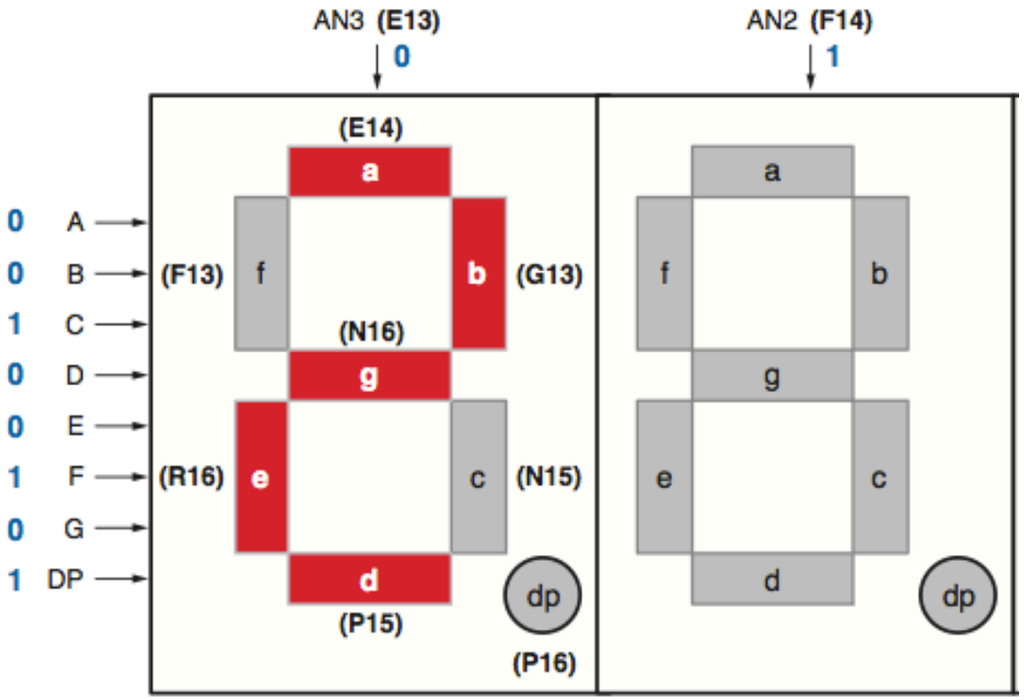


Table 3-1: **FPGA Connections to Seven-Segment Display (Active Low)**

Segment	FPGA Pin
A	E14
B	G13
C	N15
D	P15
E	R16
F	F13
G	N16
DP	P16

Table 3-2: **Digit Enable (Anode Control) Signals (Active Low)**

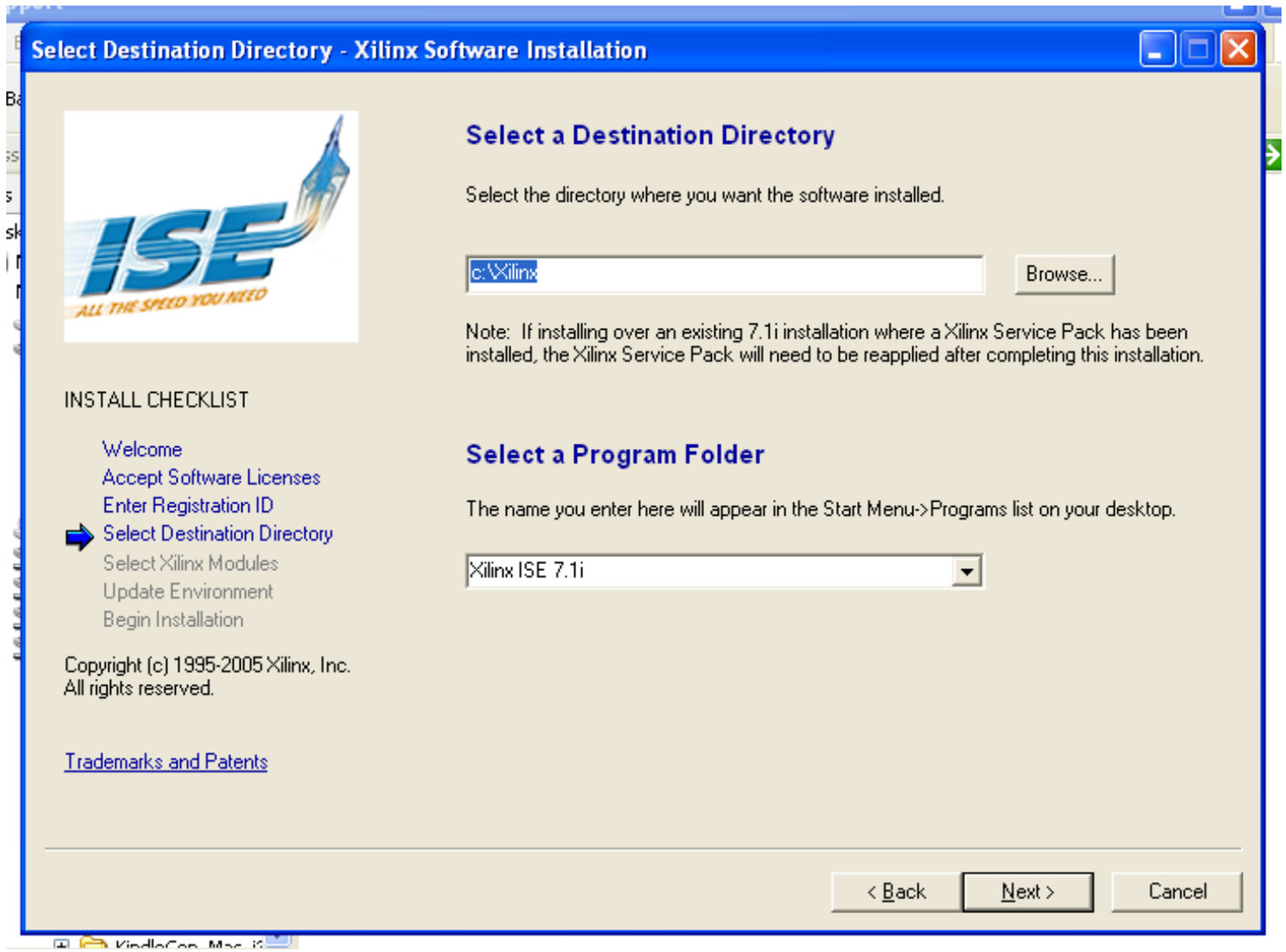
Anode Control	AN3	AN2	AN1	AN0
FPGA Pin	E13	F14	G14	D14

Installing software

Installing Xilinx ISE 7.1 in windows

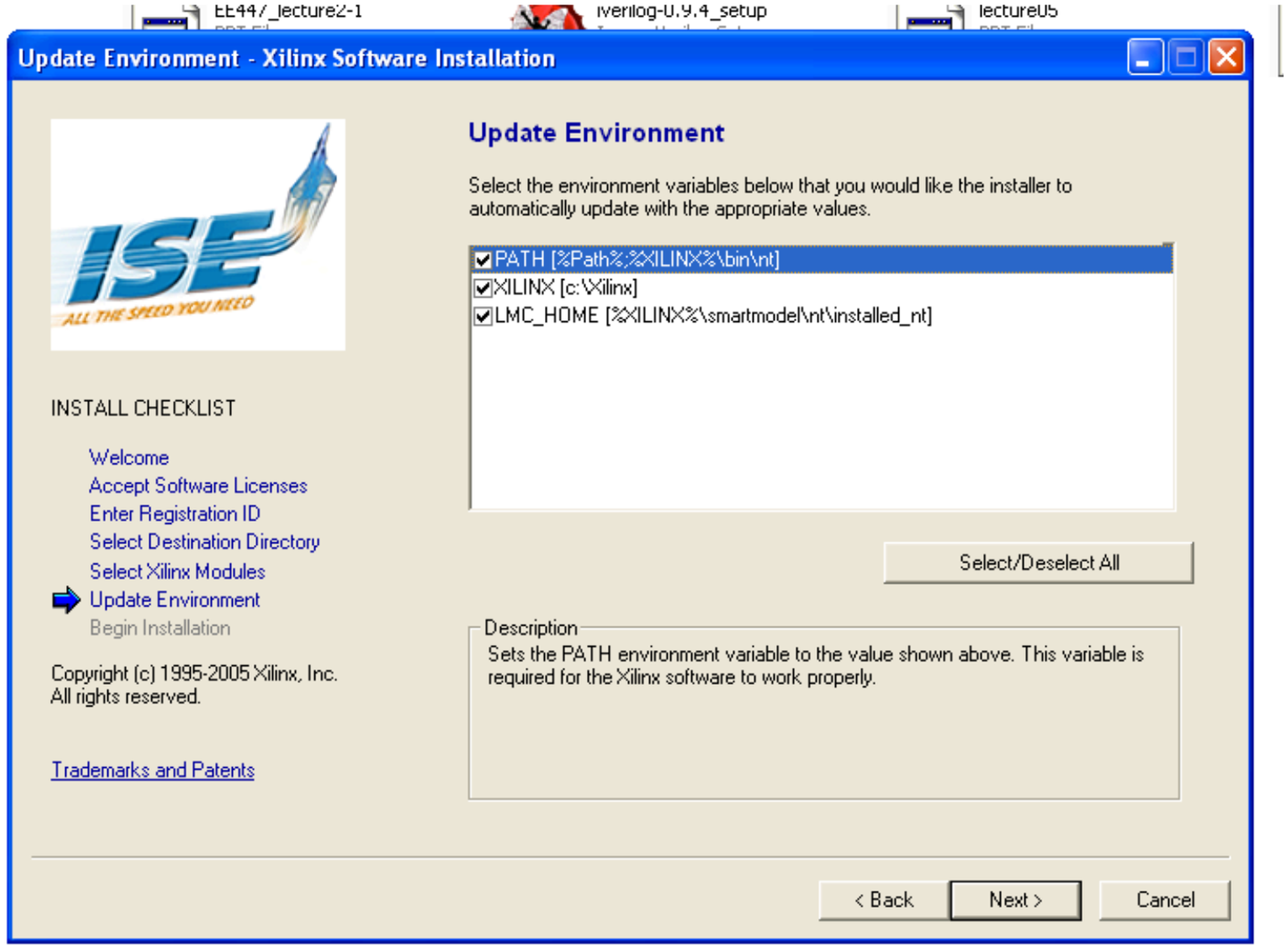
- Download all the files from the course website.
- Put them all into a **local** sub-directory **without any spaces** (e.g. c:\cpe462), and install them in order.
- Just install them all into c:\xilinx

1. WebPACK_71_fcfull_i.exe
2. 7_1_01i_pc.exe
3. EDK_7_1_02i_win.exe
4. 7_1_03i_pc.exe
5. 7_1_04i_pc.exe



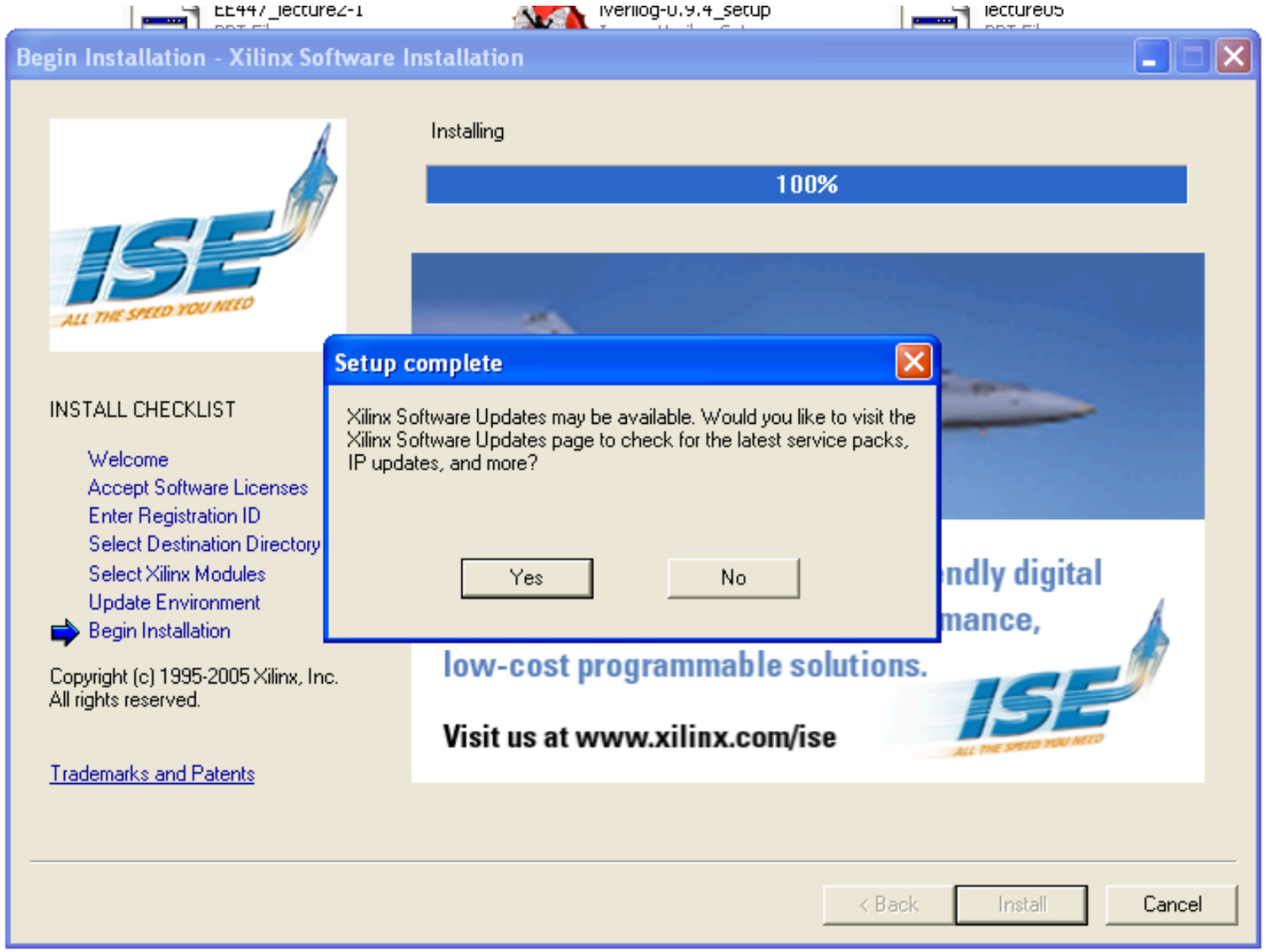
Don’t change environment

Pain will ensure if you do so.



Do NOT search for updates

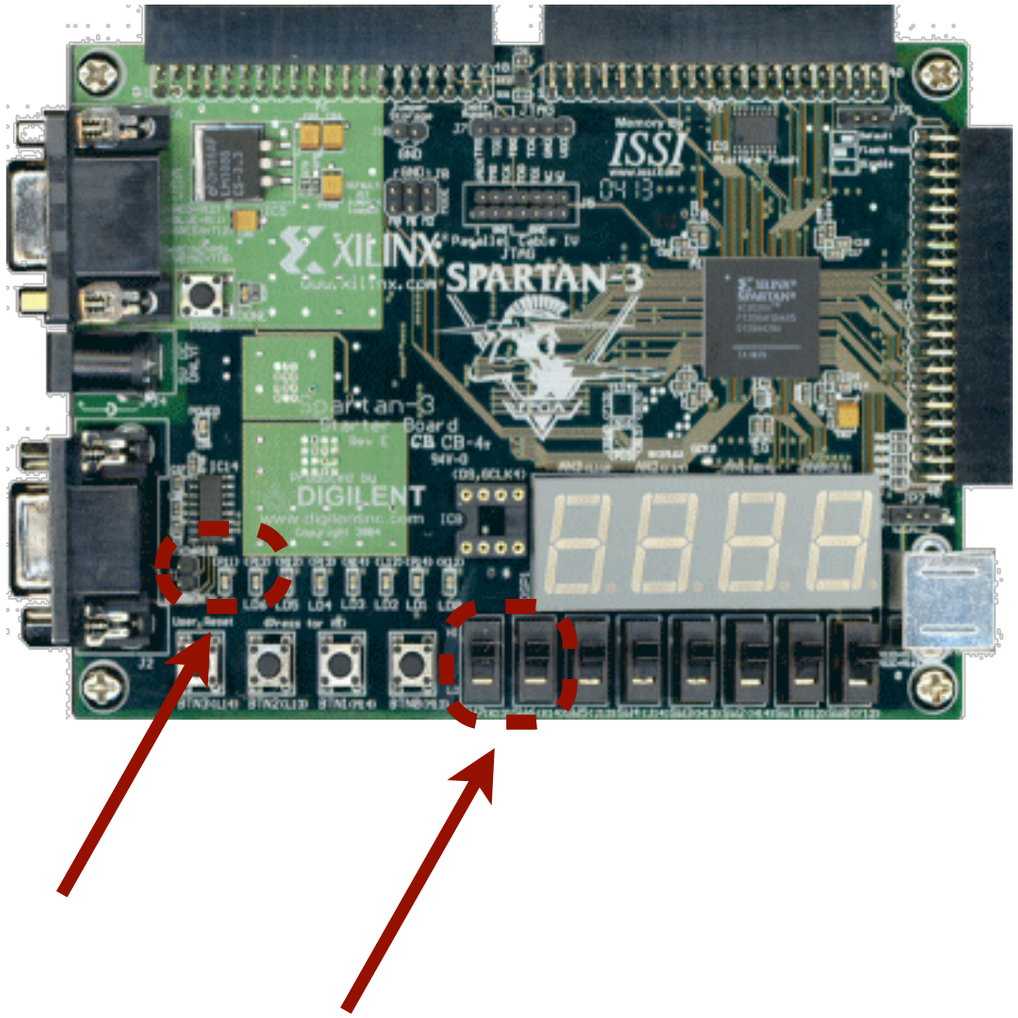
- Click on NO
- We know that we are using obsolete software



Loading NAND program into the FPGA board

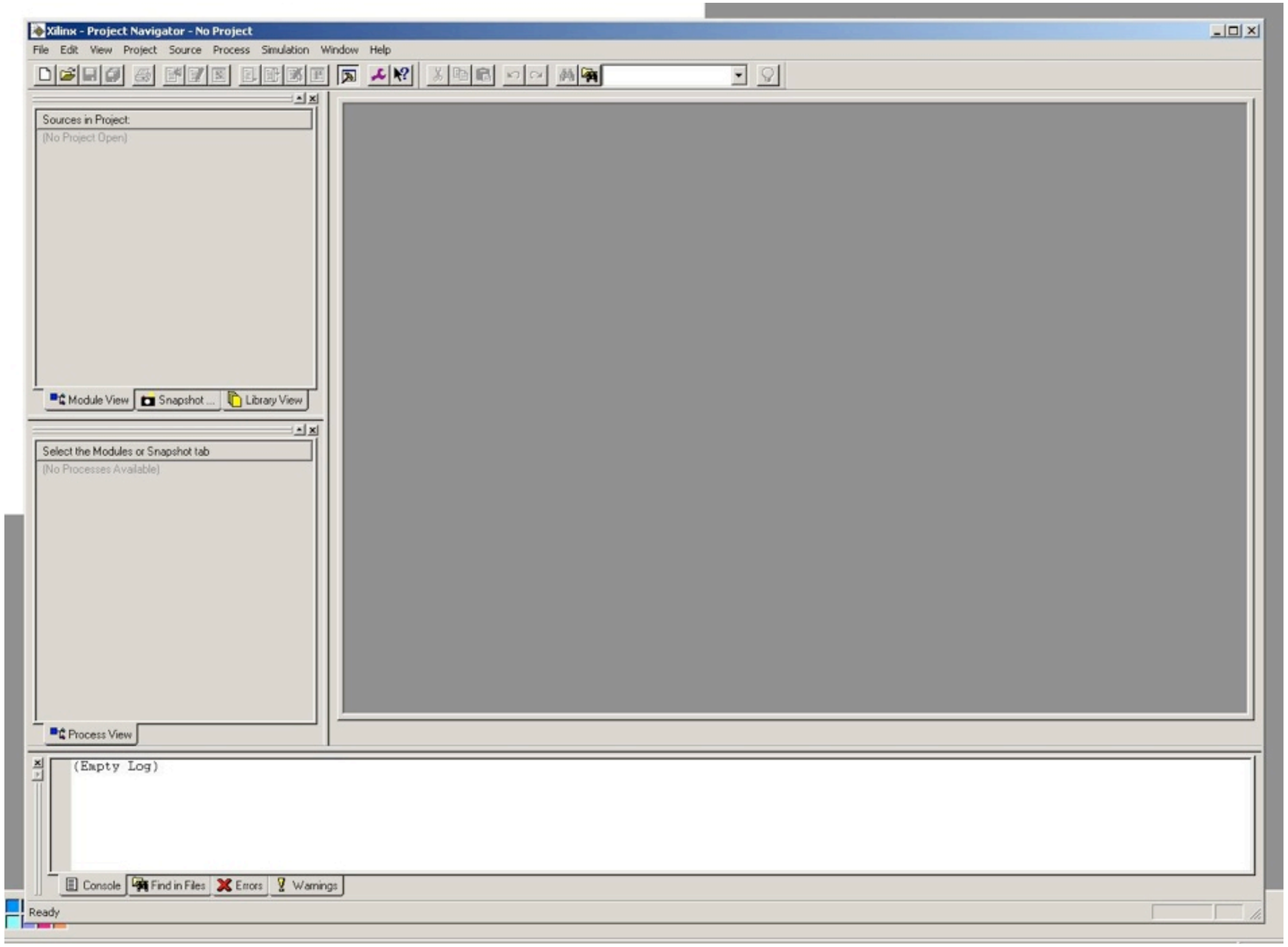
Goal

- We want to use switches #7 and #8 as the input to our NAND gate
- The result of our NAND computation, will be displayed through LED #7.

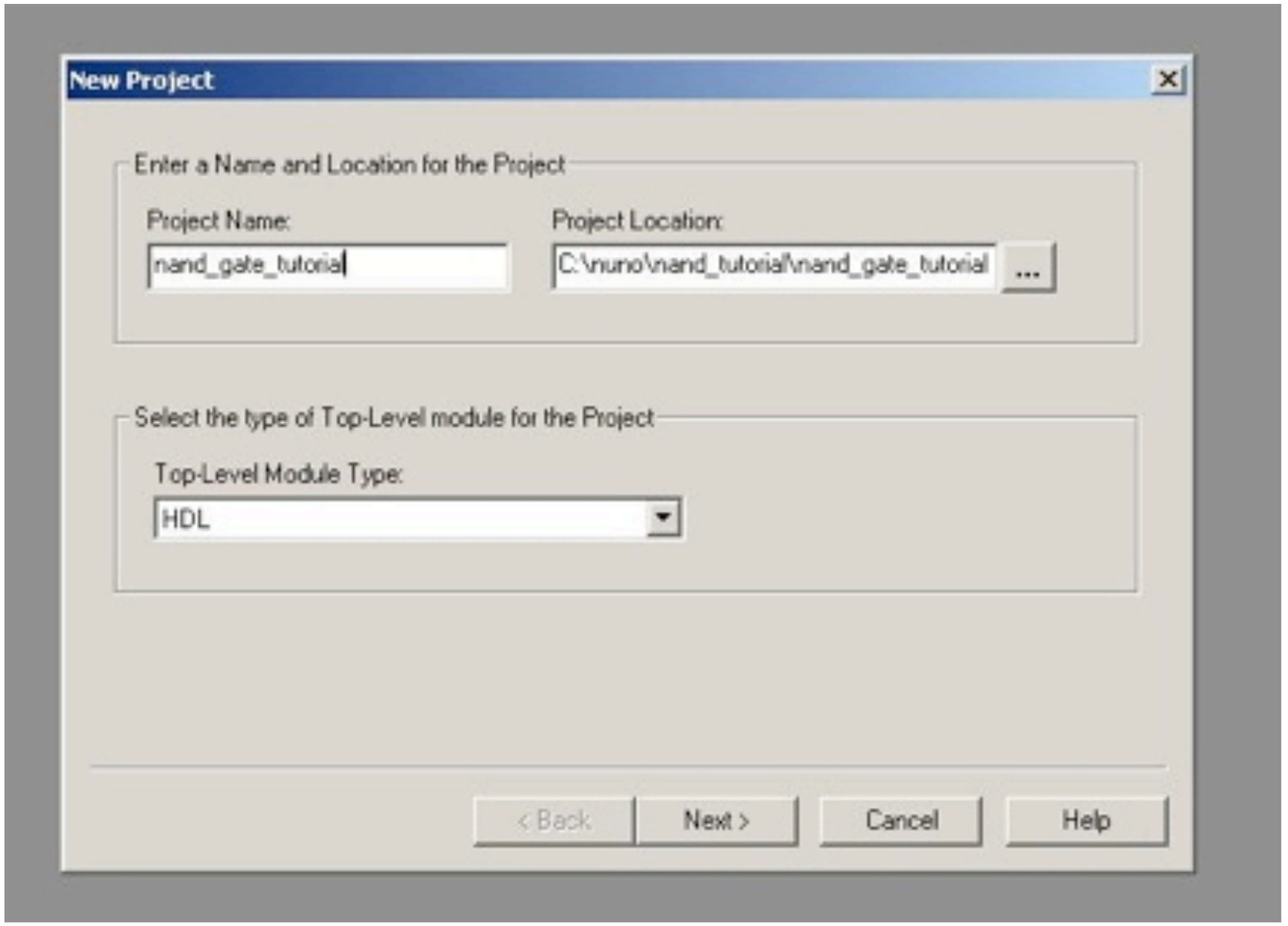


Launch Xilinx ISE 7.1 navigator

Remember, you can do all these steps at home... until the very end, for which you must come to the lab.



Create a new project



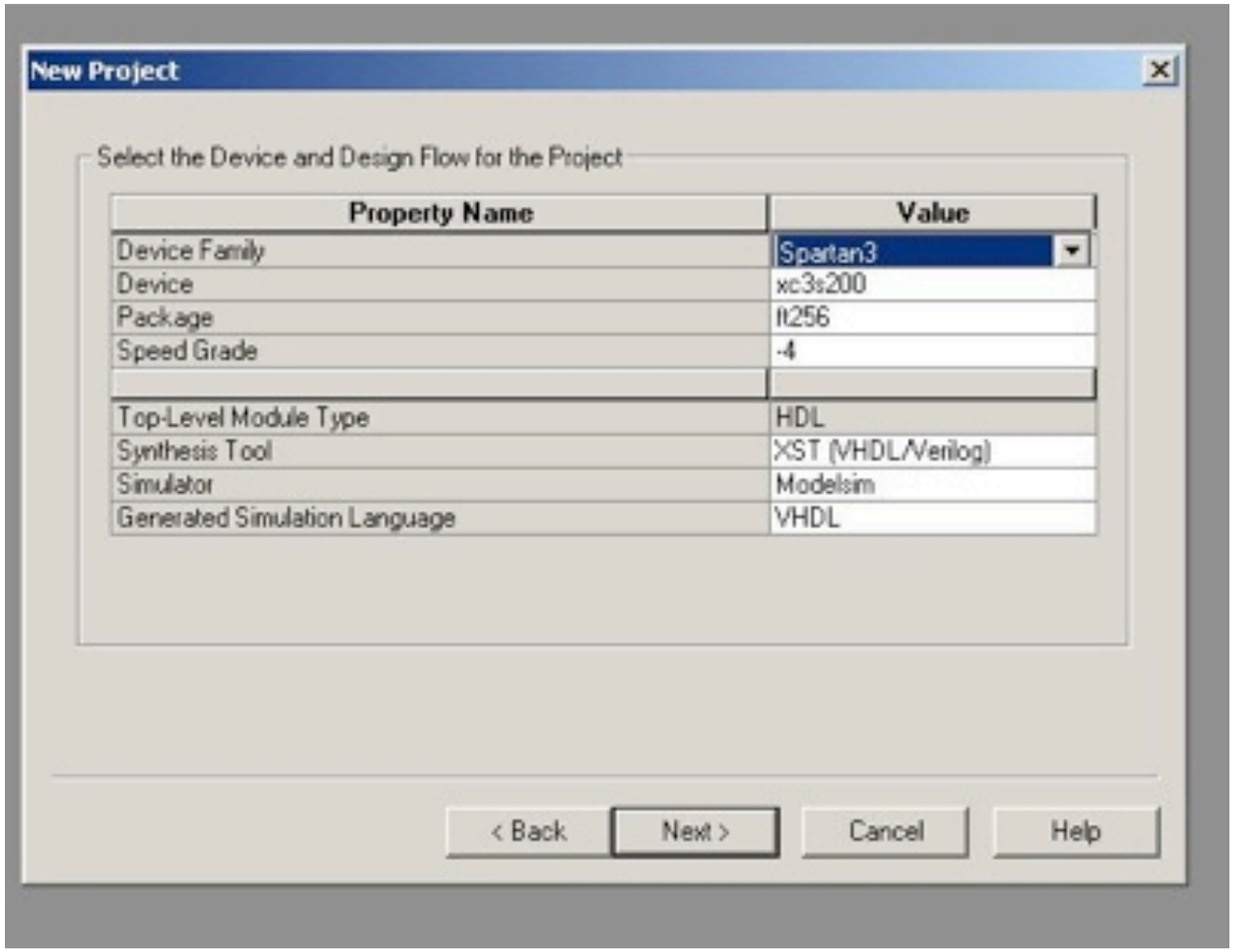
Ensure FPGA settings are correct

Family: Spartan-3

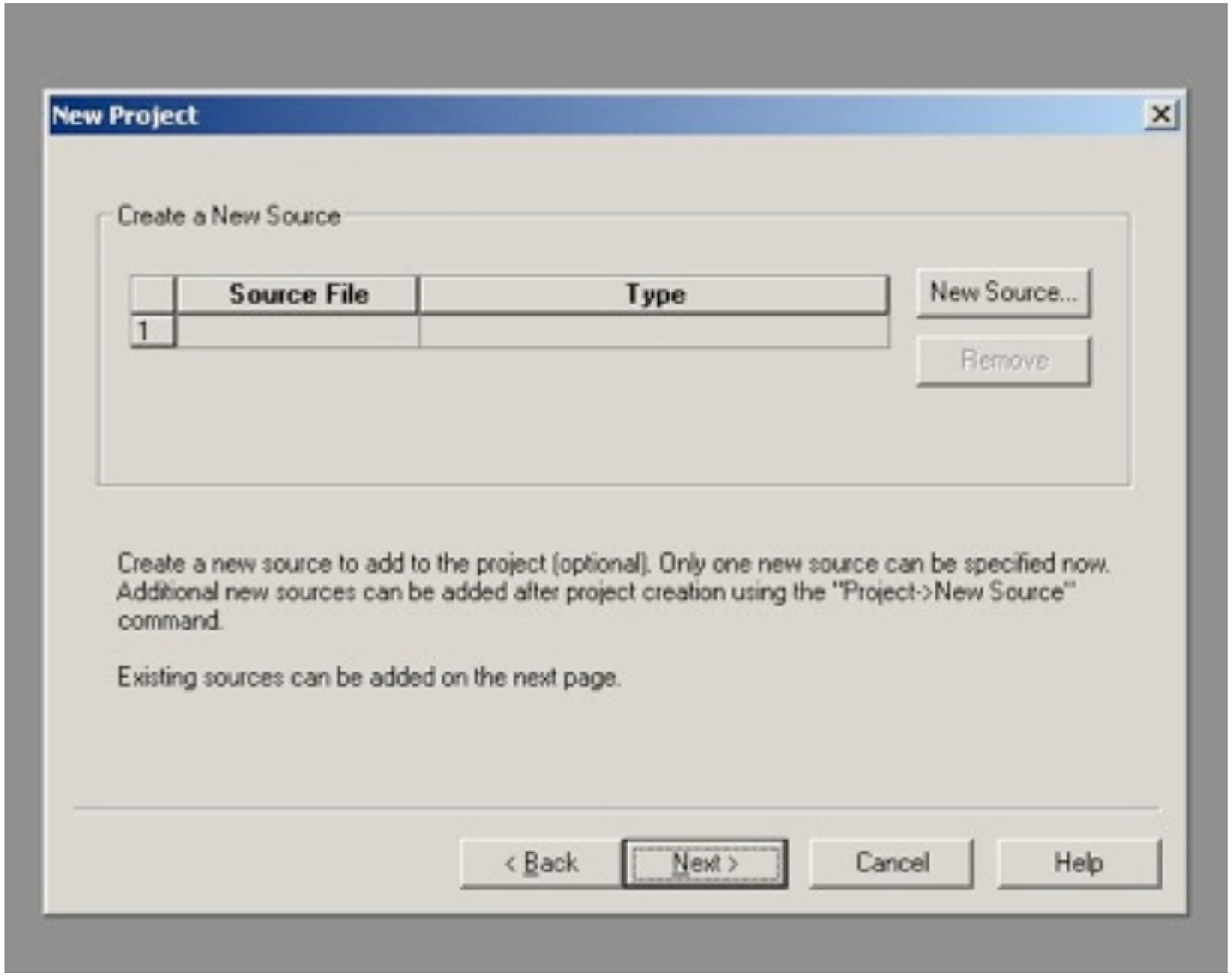
Device: xc3s200

Package: ft256

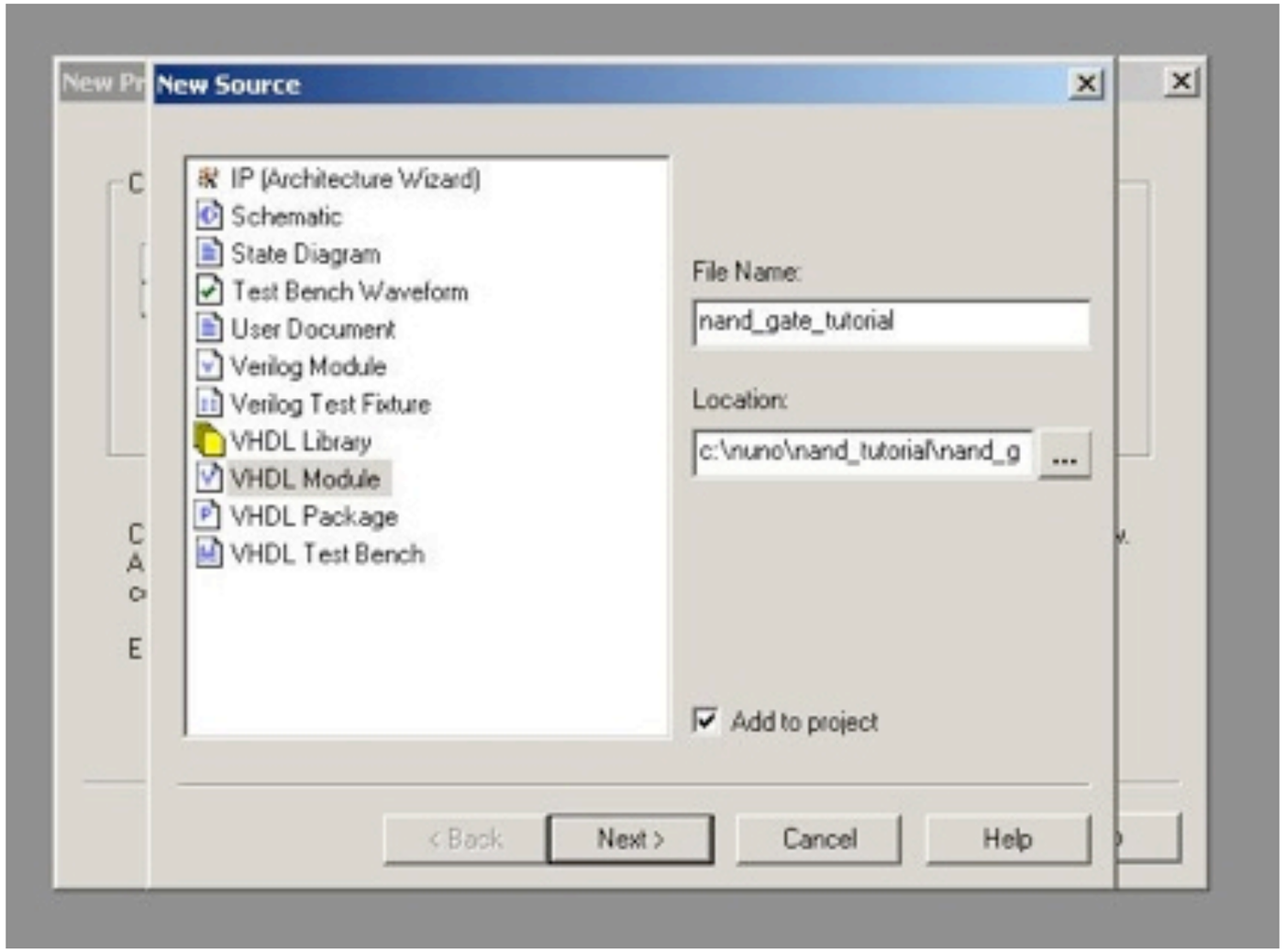
Speed Grade: -4



Add a new source



Select a VHDL Module

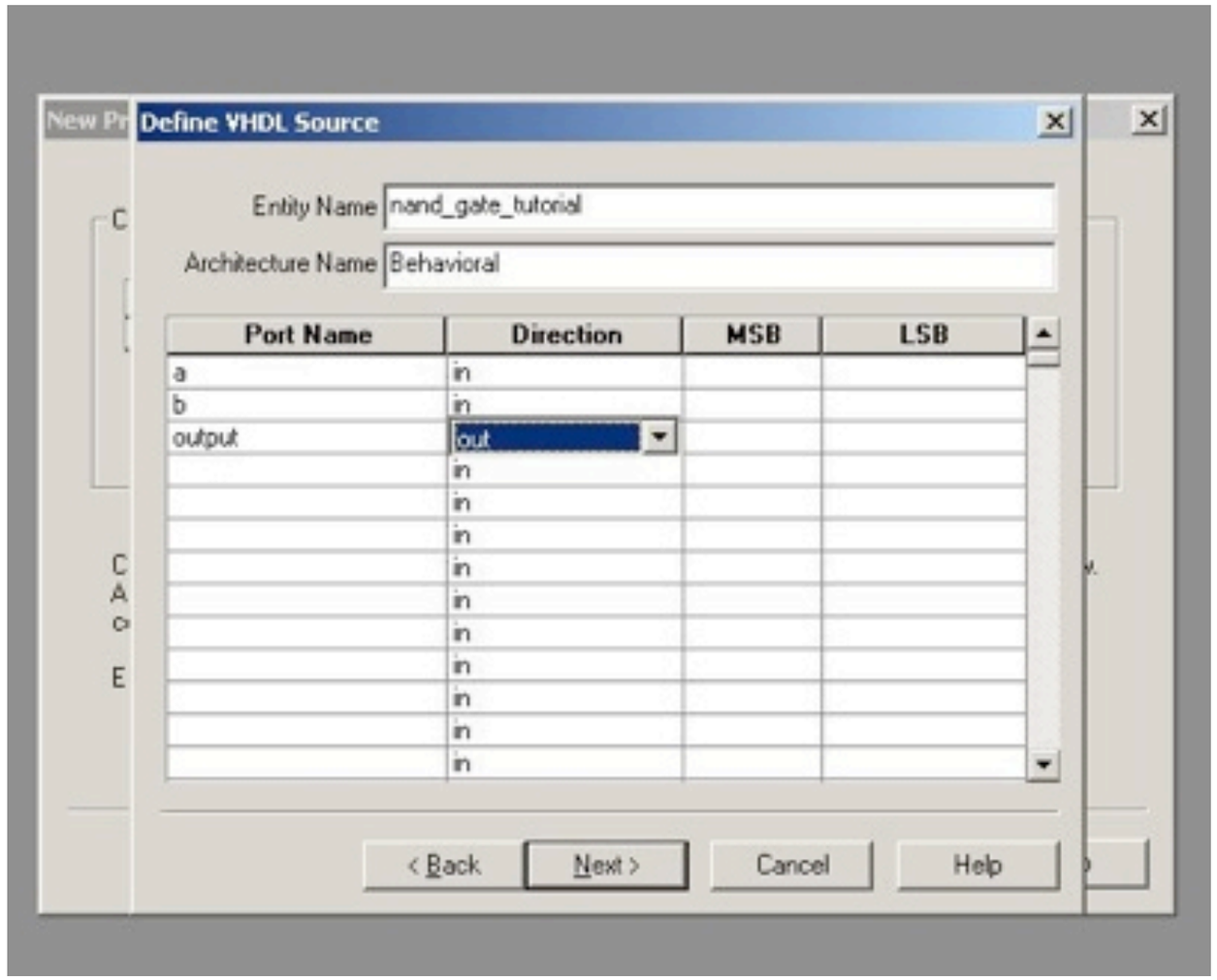


Adding the appropriate ports

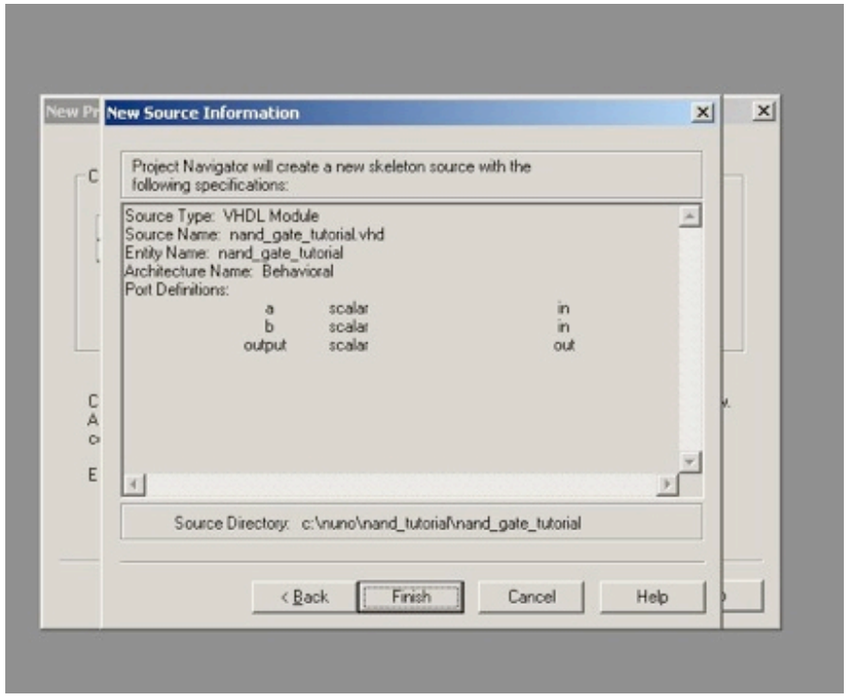
For a 2-bit NAND we just need three ports:

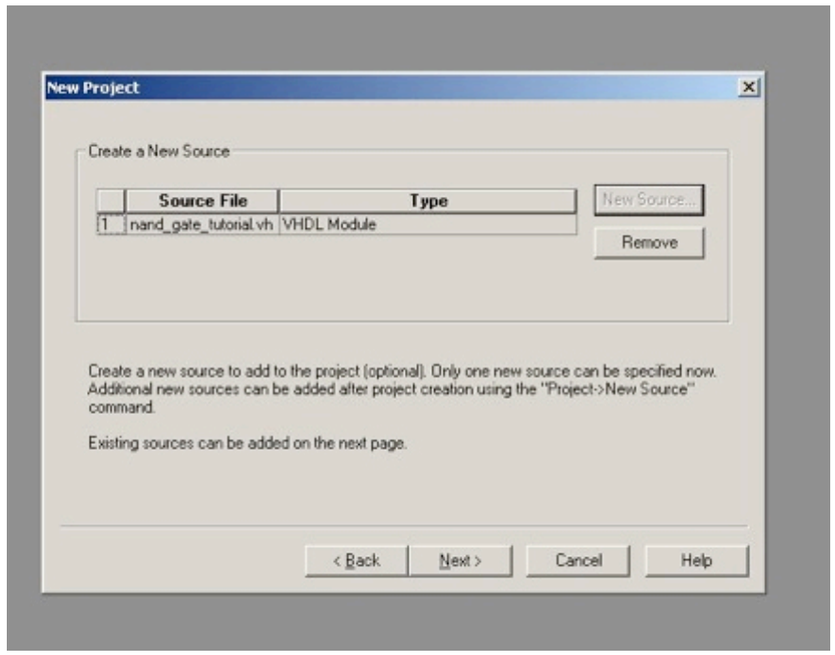
a and **b** are in

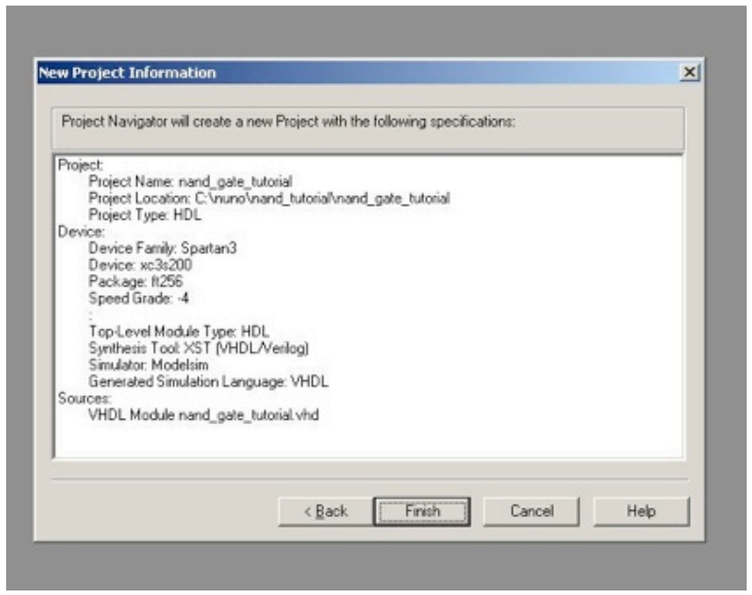
output are out



Finishing the project

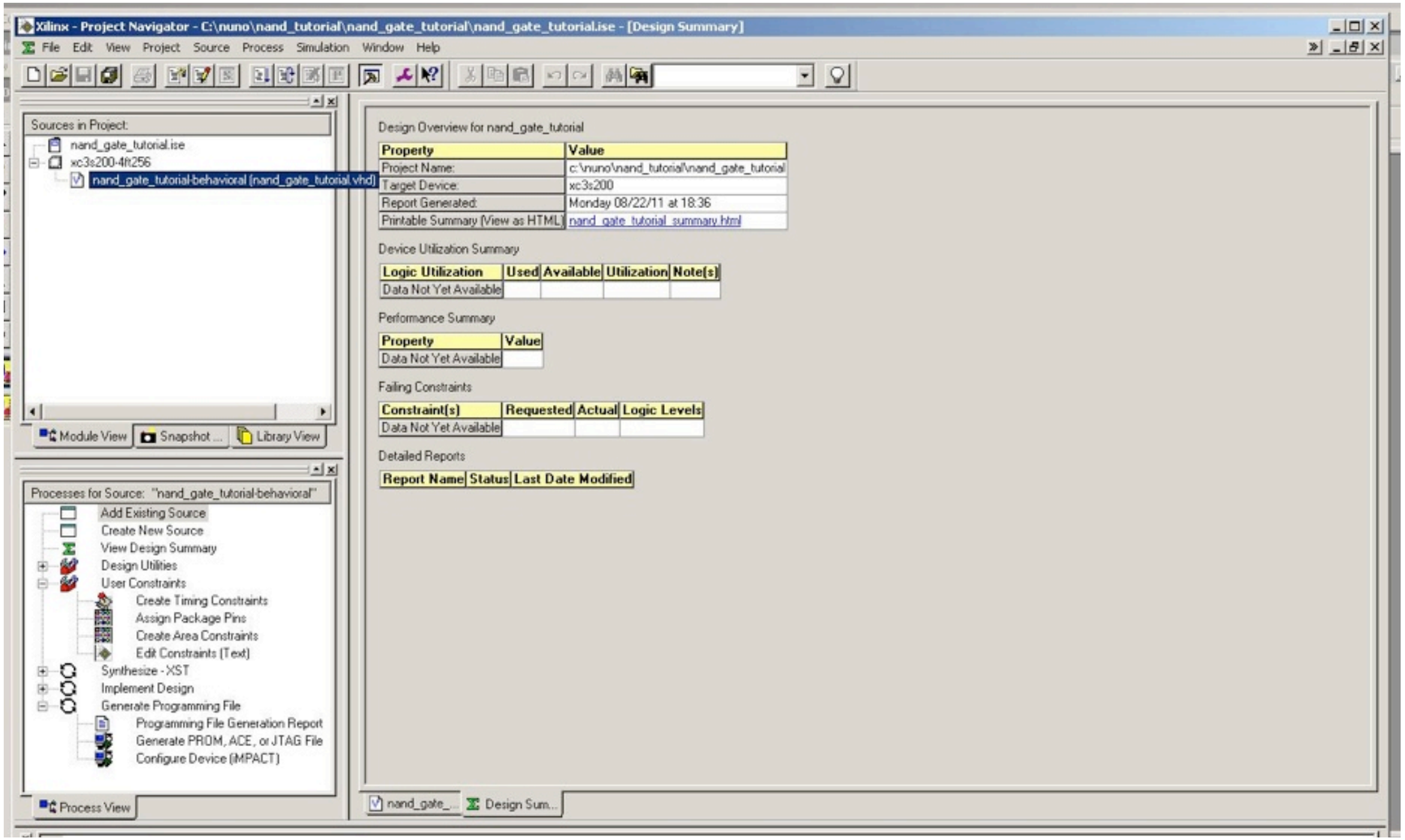




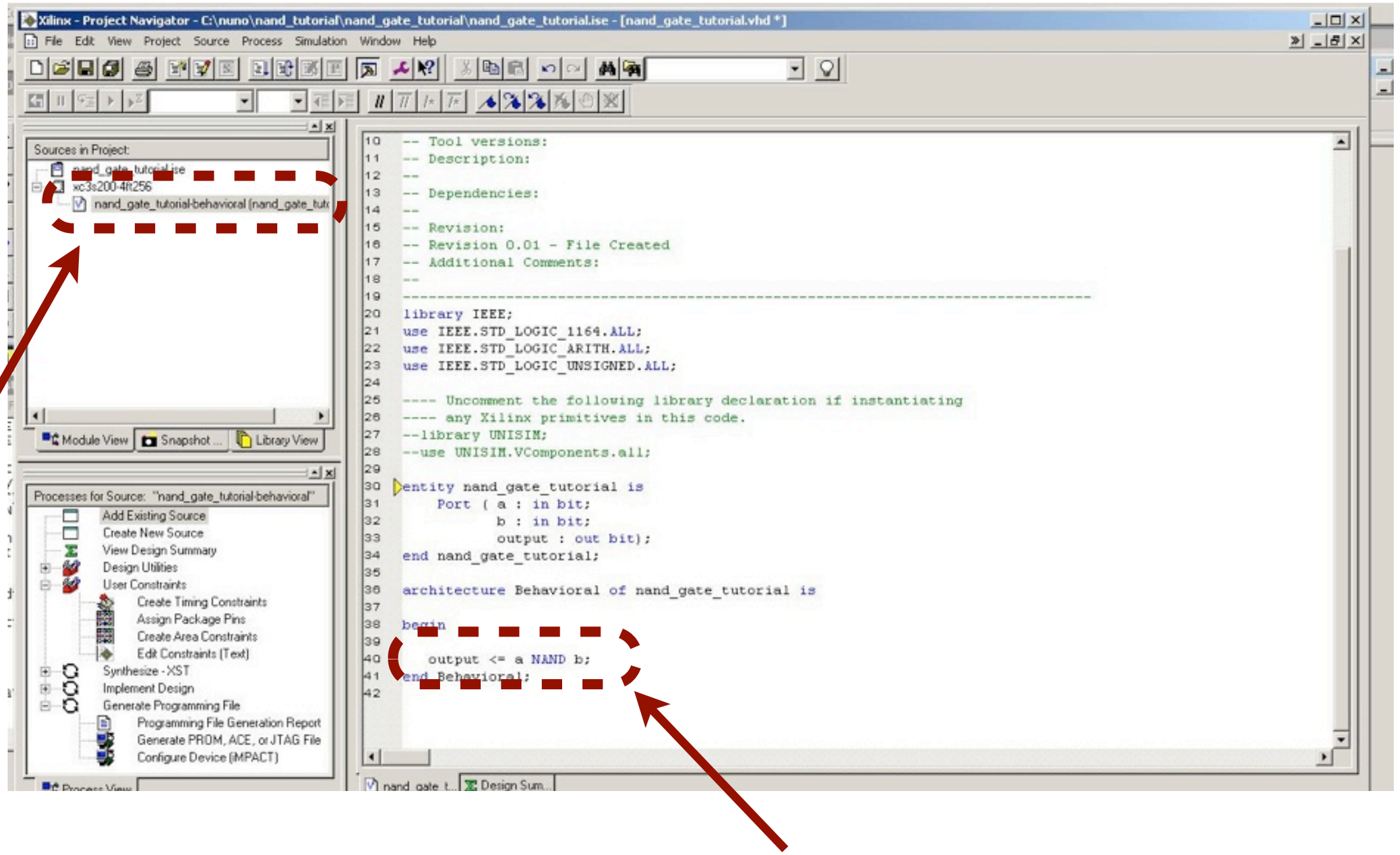


Just keep pressing *next*...

Main project window

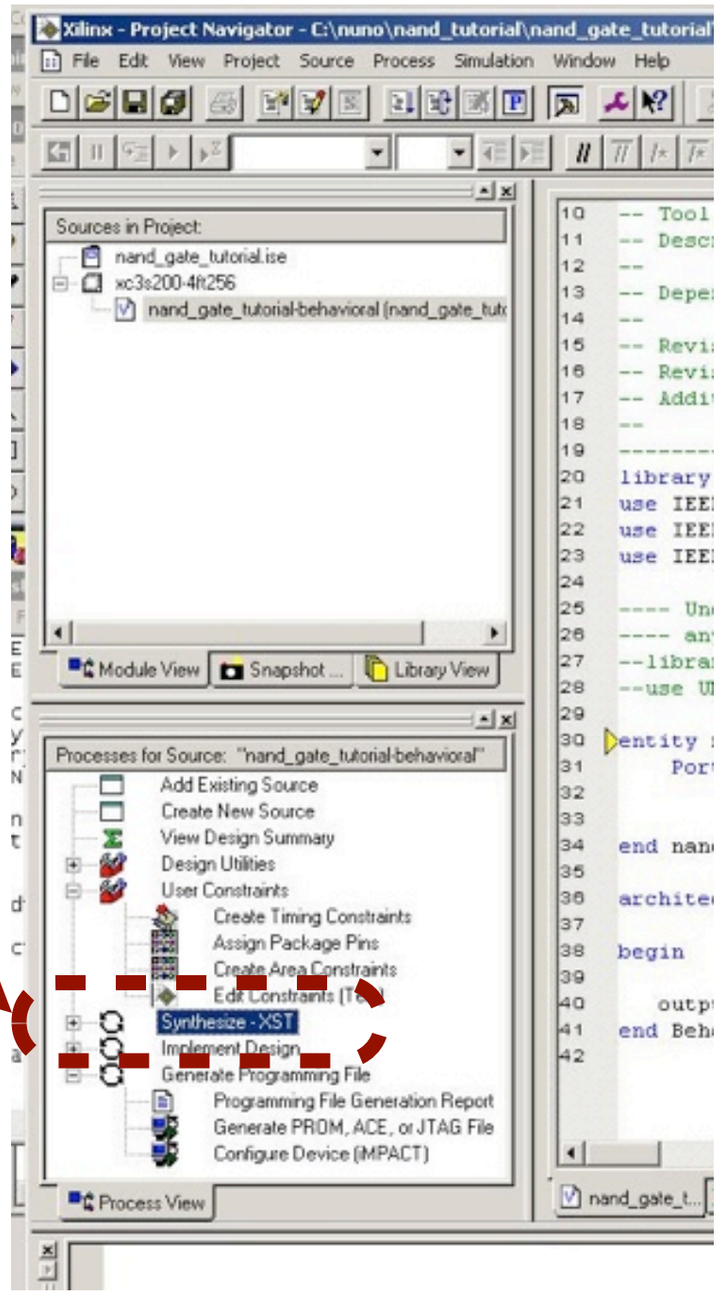


Select VHDL and type code



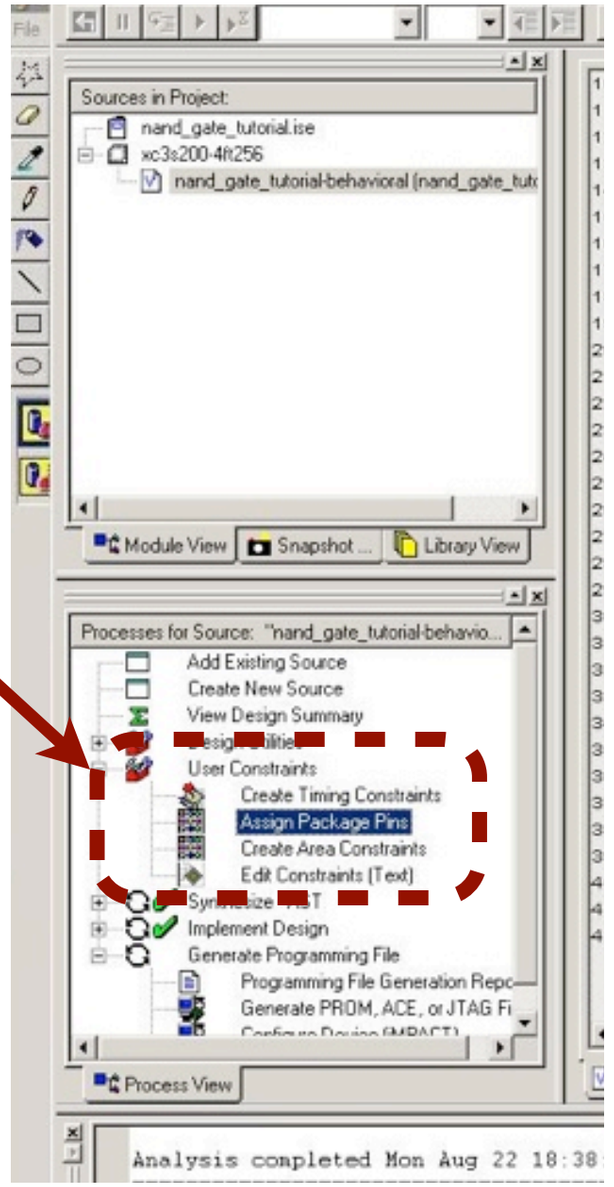
Synthesize

- Synthesize the VHDL code while ensuring no syntax errors
- Double click in Synthesize XST



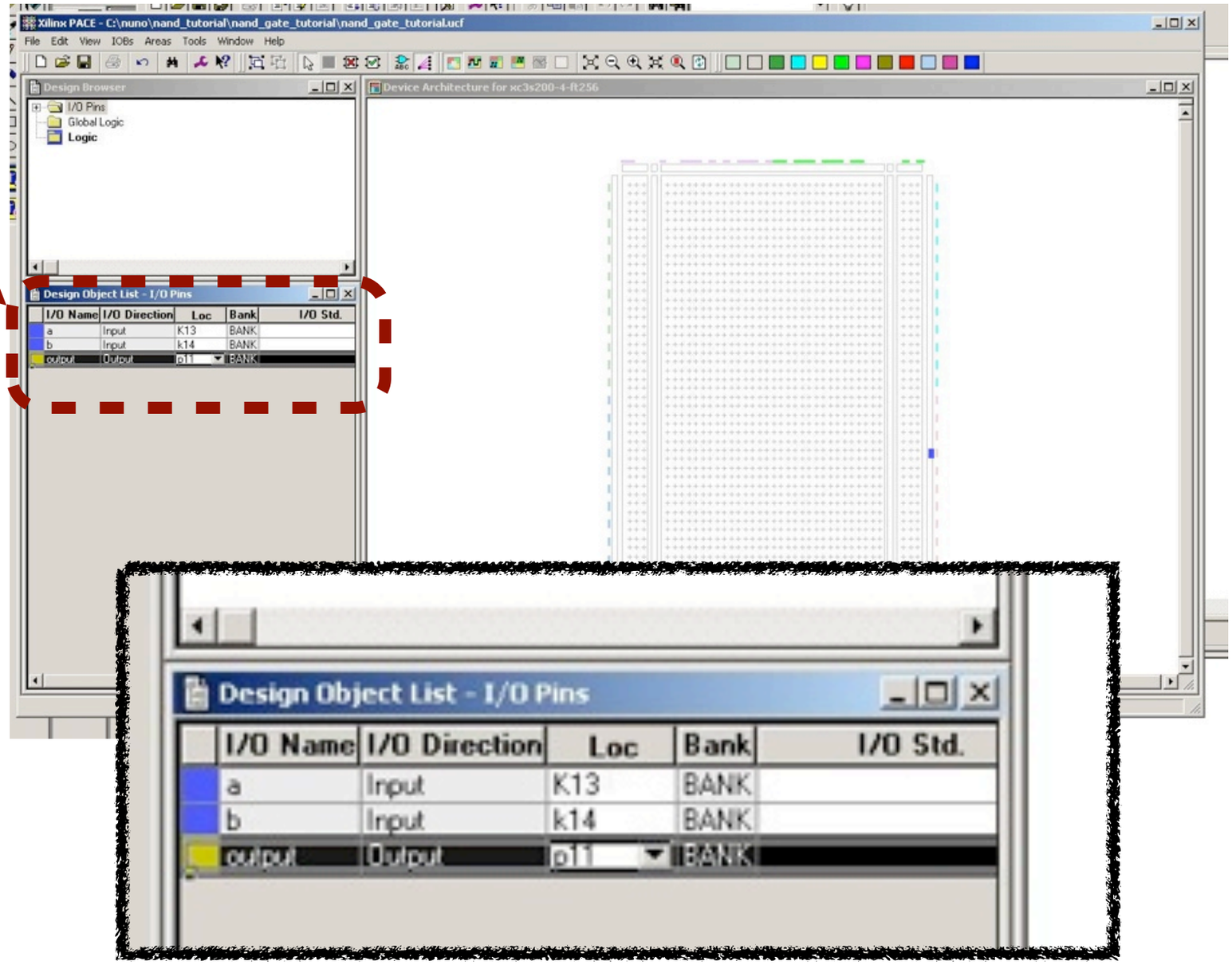
Assign package pins

- We need to assign the input / output ports to the buttons and LEDs
- Click on *Assign Package Pins*, which is under *User Constraints*
- And say "*YES*" to the creation of an UCF file



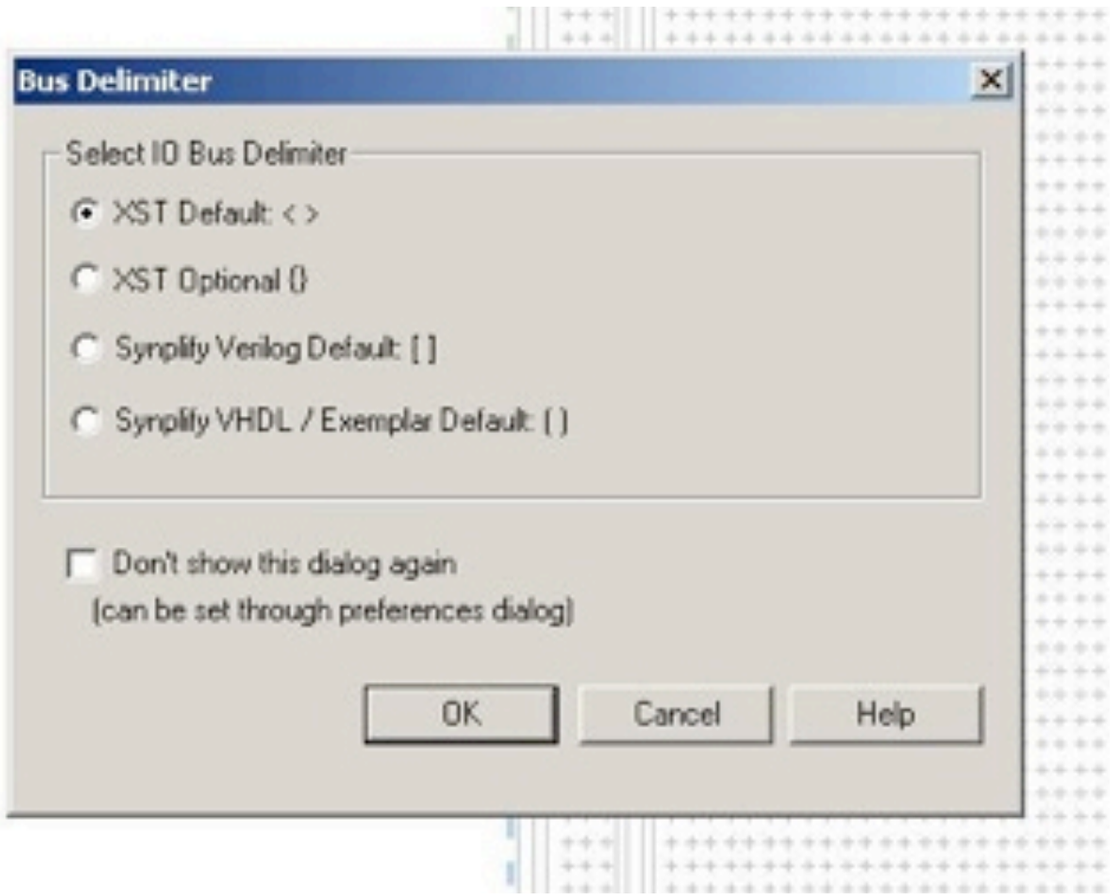
Mapping pins to design

- We want to use switches #7 and #8 as the input
- The physical FPGA BGA pins for these are K13 and K14
- The output should be displayed through LED #7
- This is pin P11

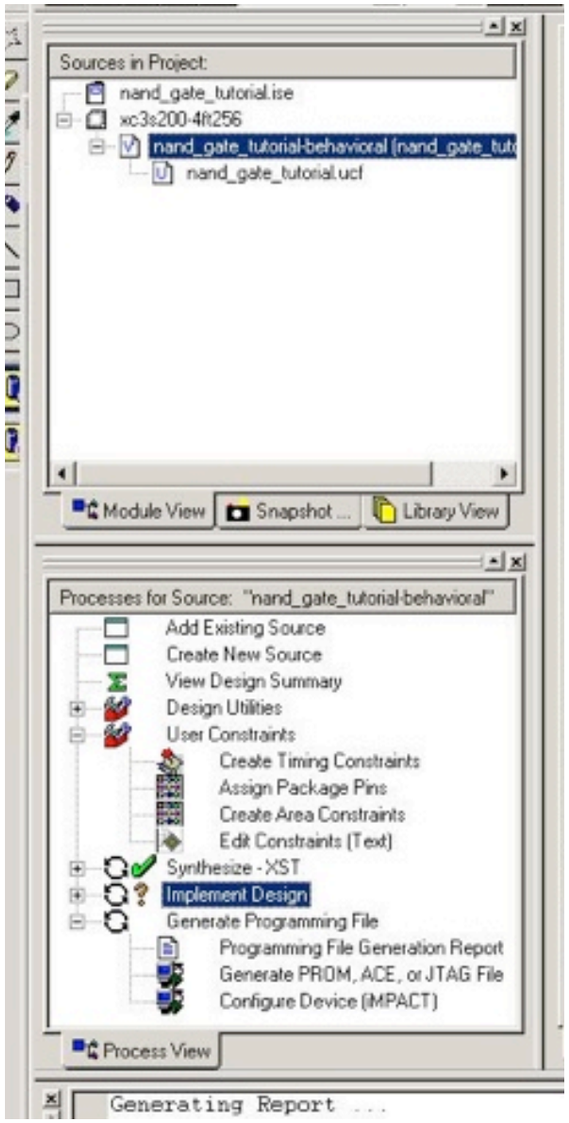


Specifying default bus delimiter

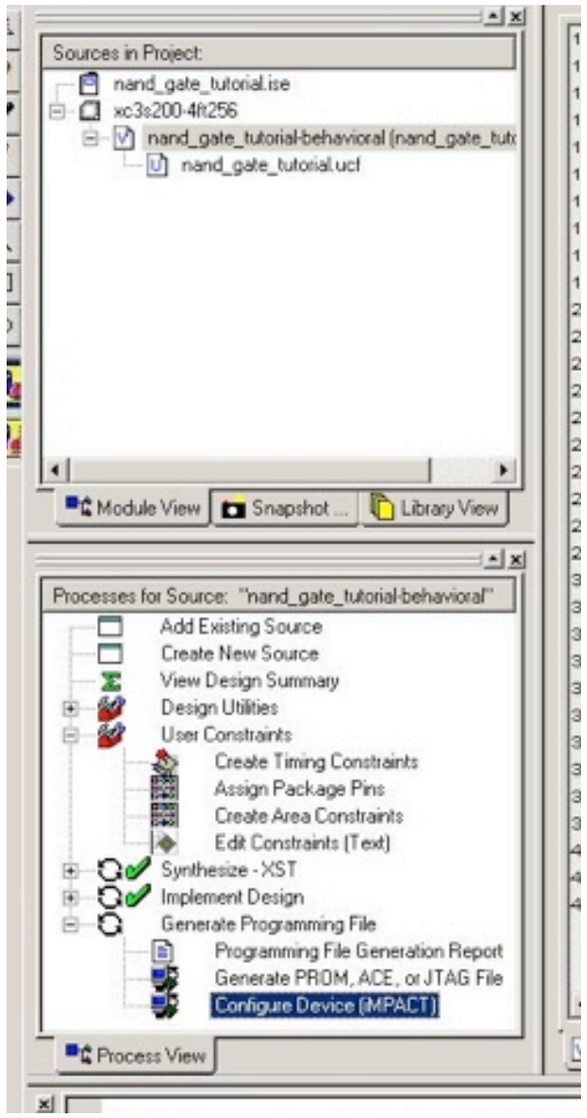
- After saving, ensure that you are using the default bus delimiter.



Final steps inside project navigator



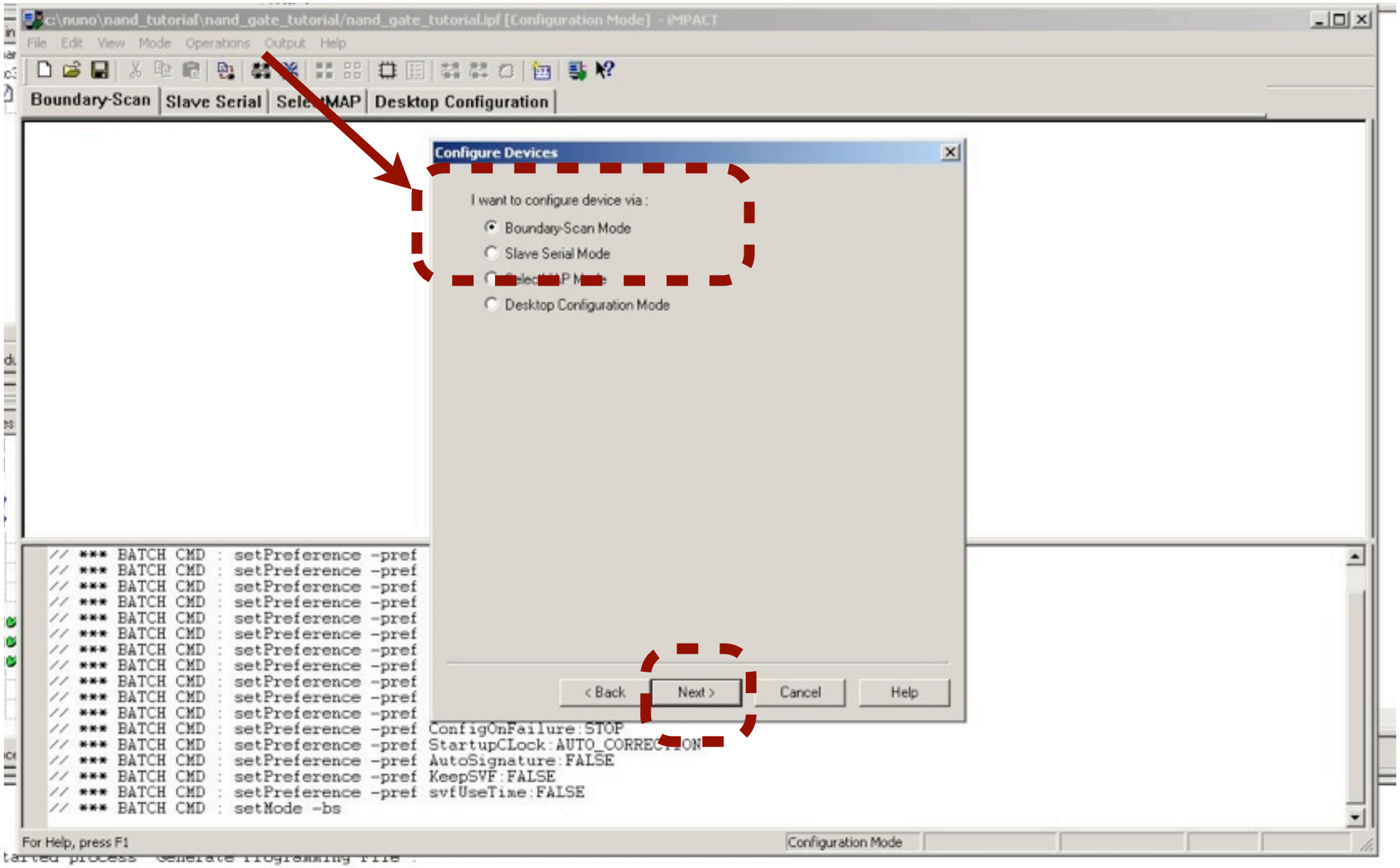
then



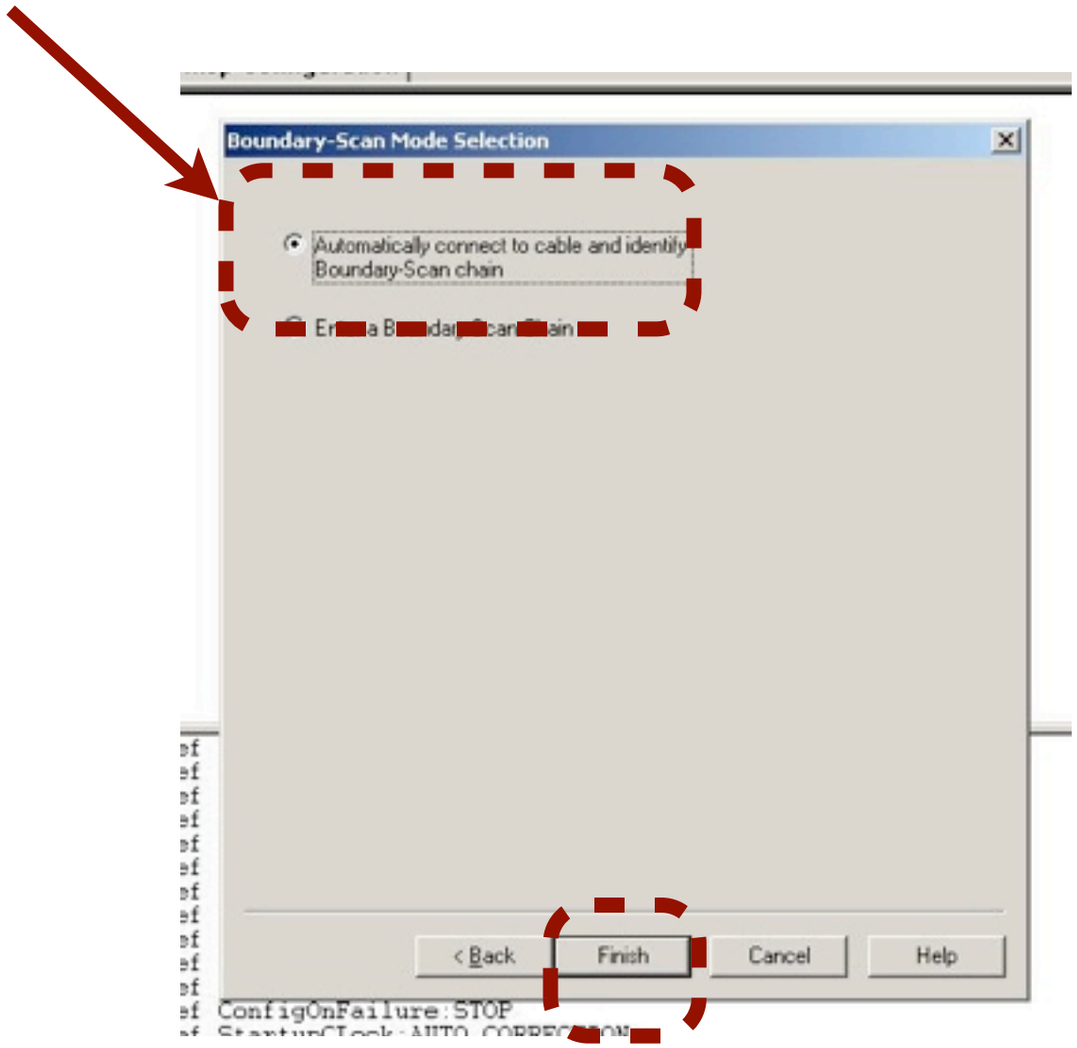
Double click on *Implement Design*

Double click on *Configure Device*

Inside IMPACT 1/2

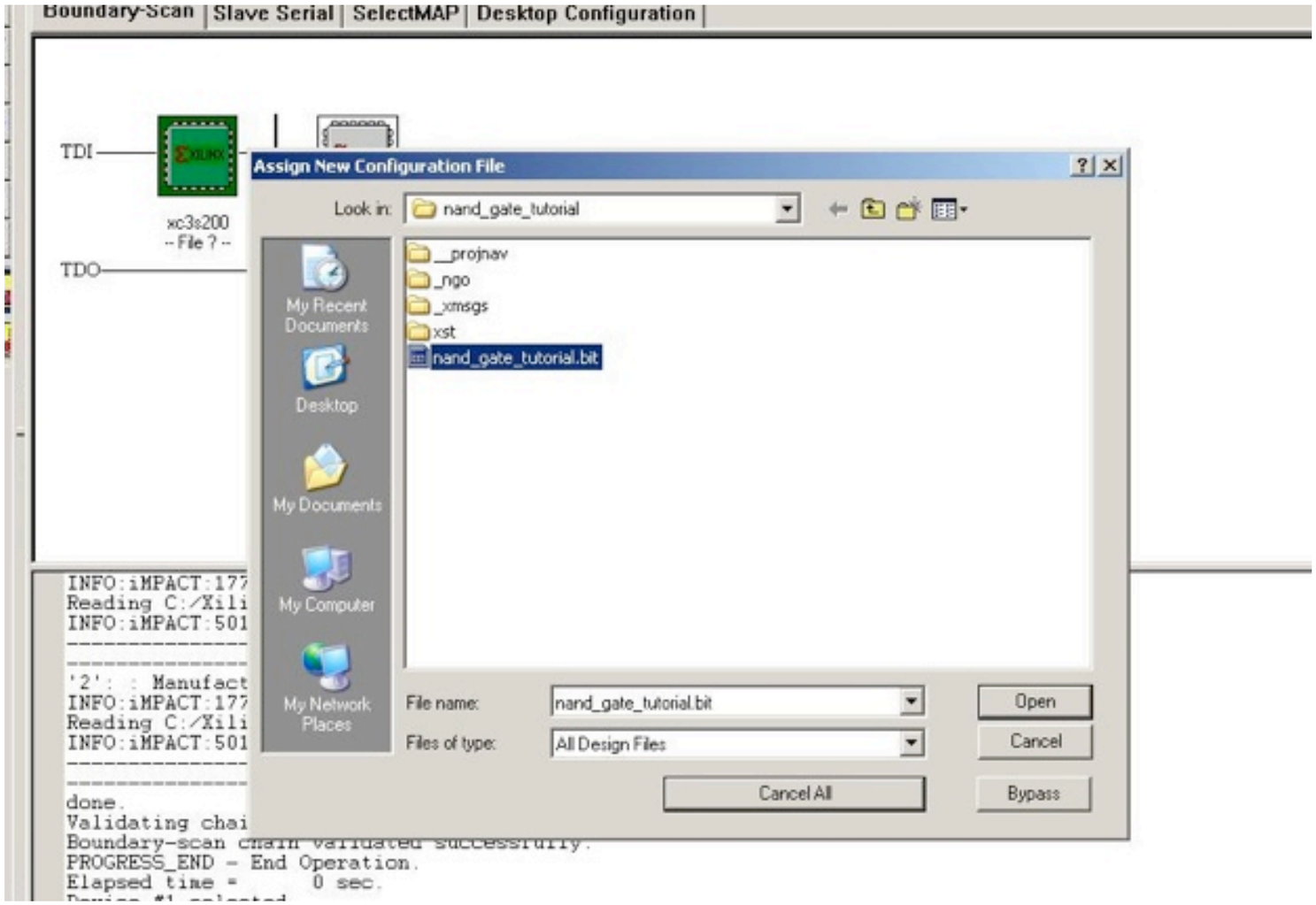


Inside IMPACT 2/2

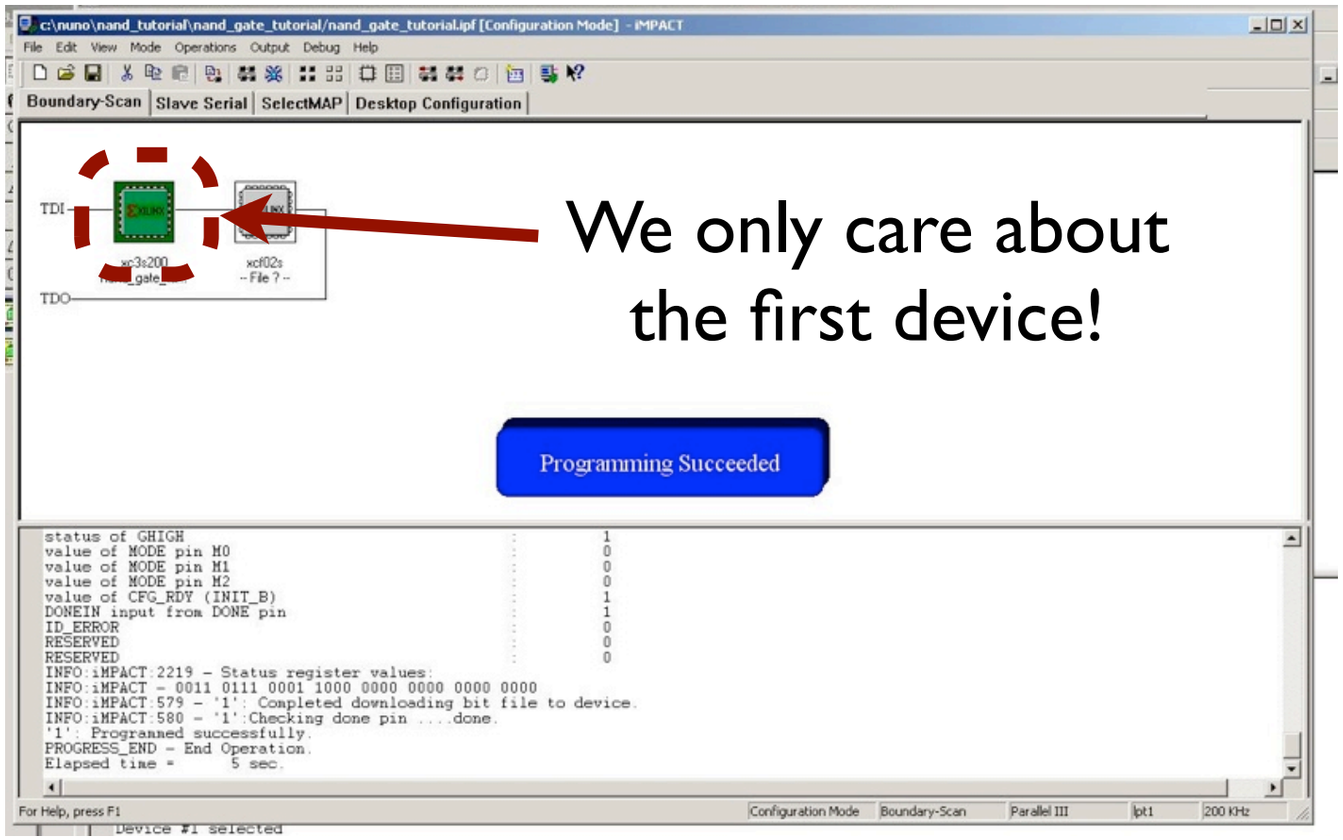


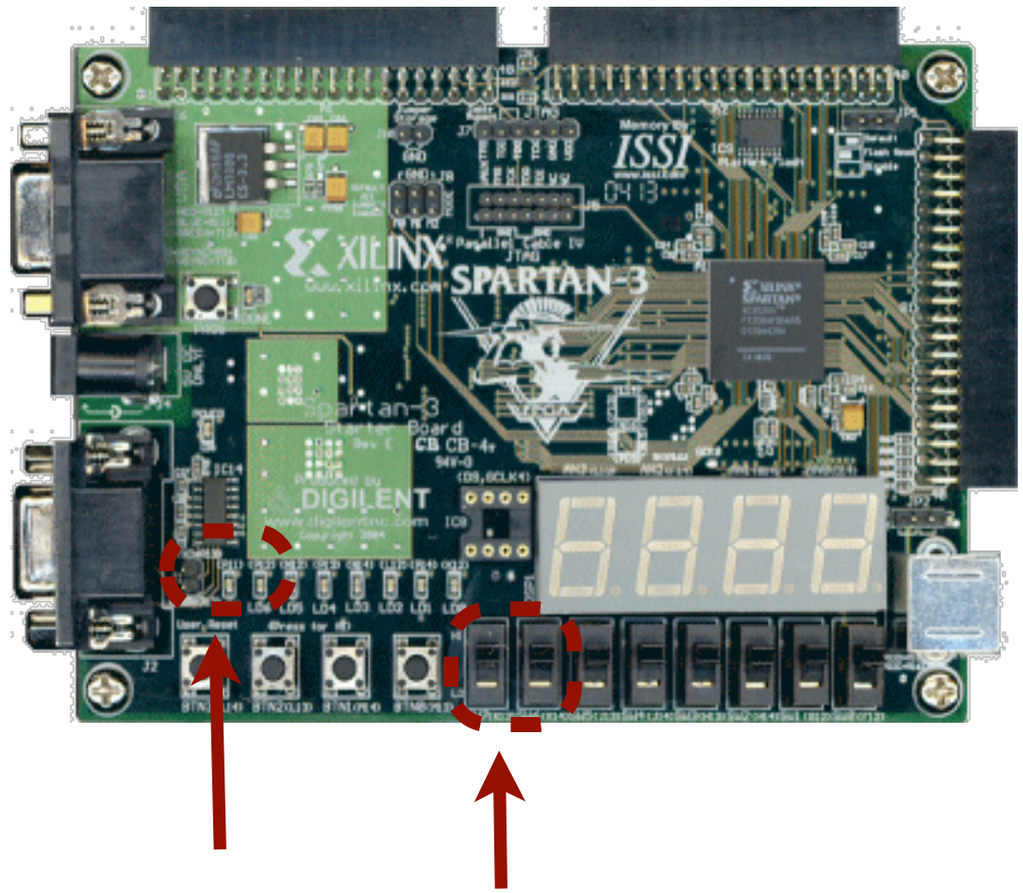
Specifying the bitstream

- Select the .bit file
- This file, generated after the "*implement design*" step is the bitstream that needs to be sent into the FPGA



Playing with the board





Programming Succeeded is what you want to see!

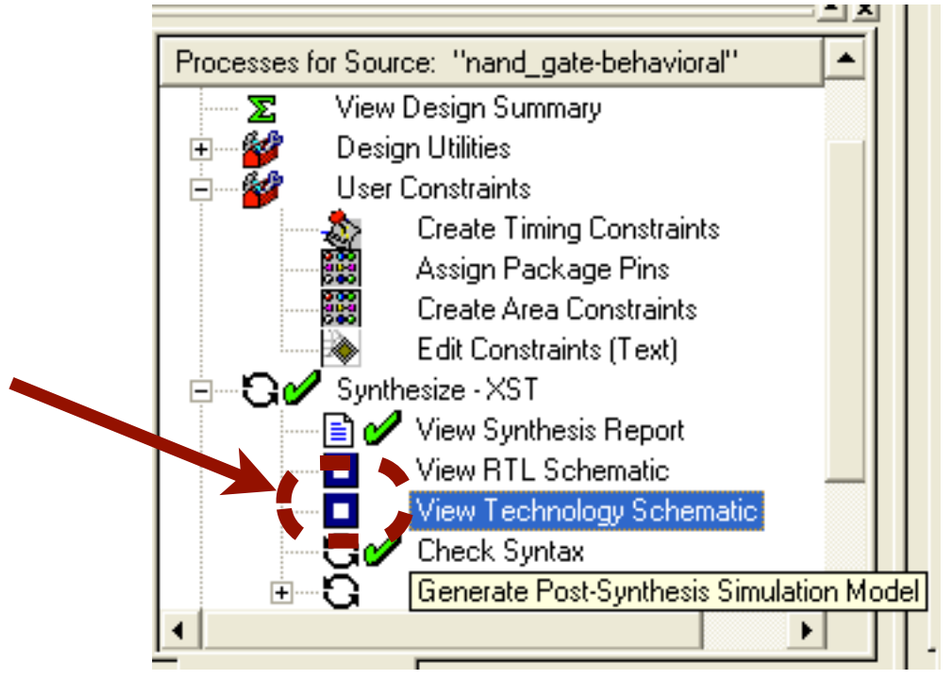
Move switches and see LED change through a NAND operation!

Verification of Synthesis

Nuno Alves (nalves@wne.edu), College of Engineering

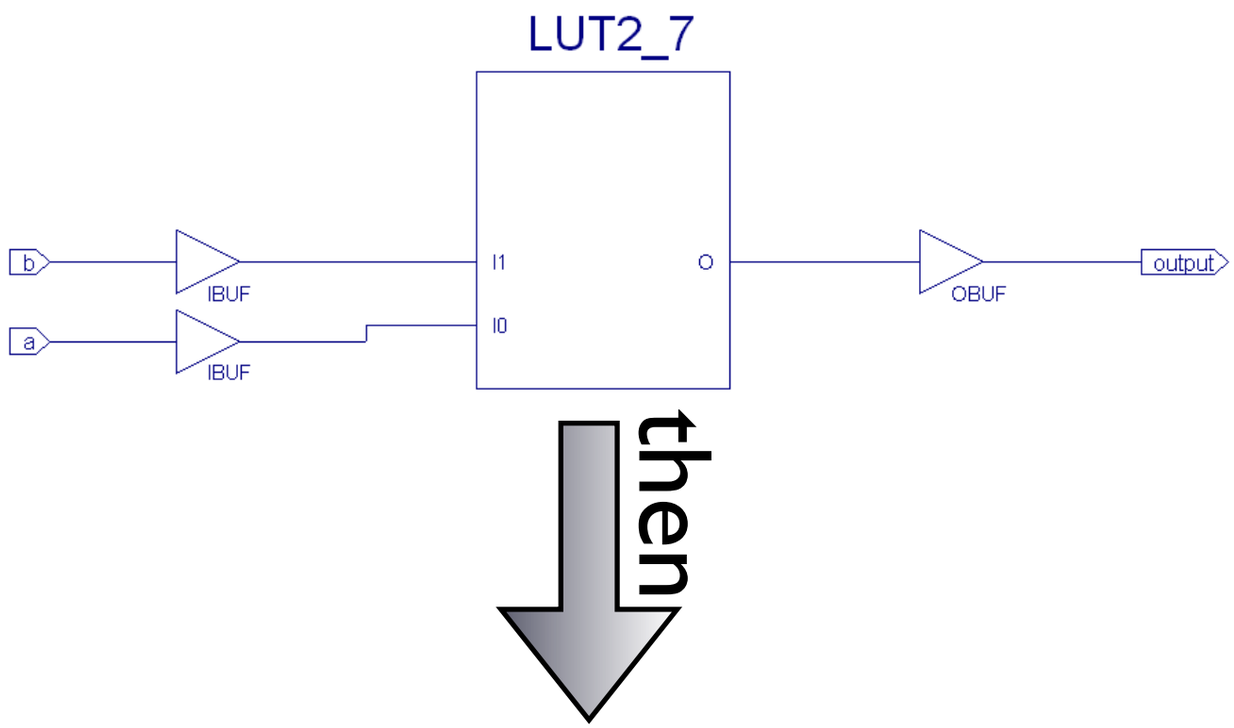
Register Transfer Level (RTL)

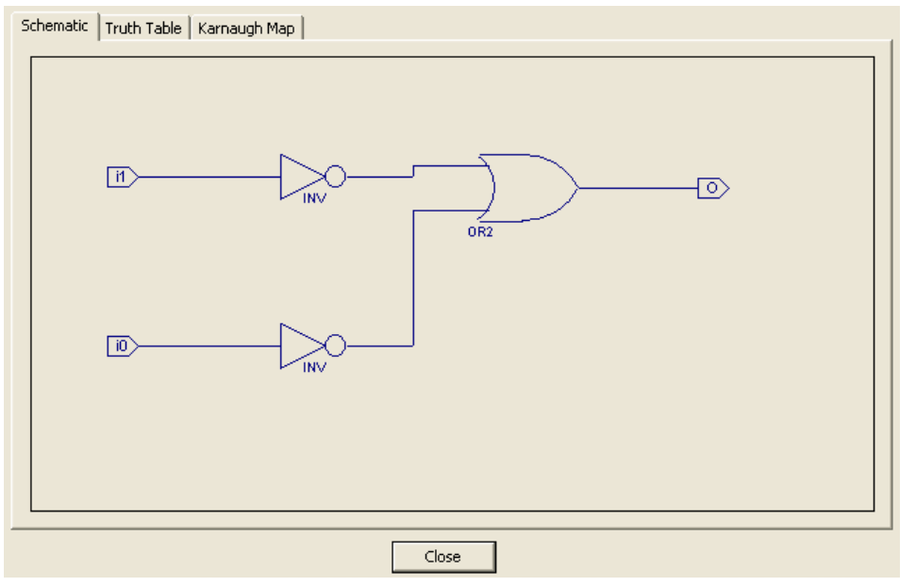
- Your synthesized design can be viewed as a schematic in the Register Transfer Level (RTL) Viewer.



Schematic for a NAND gate

then

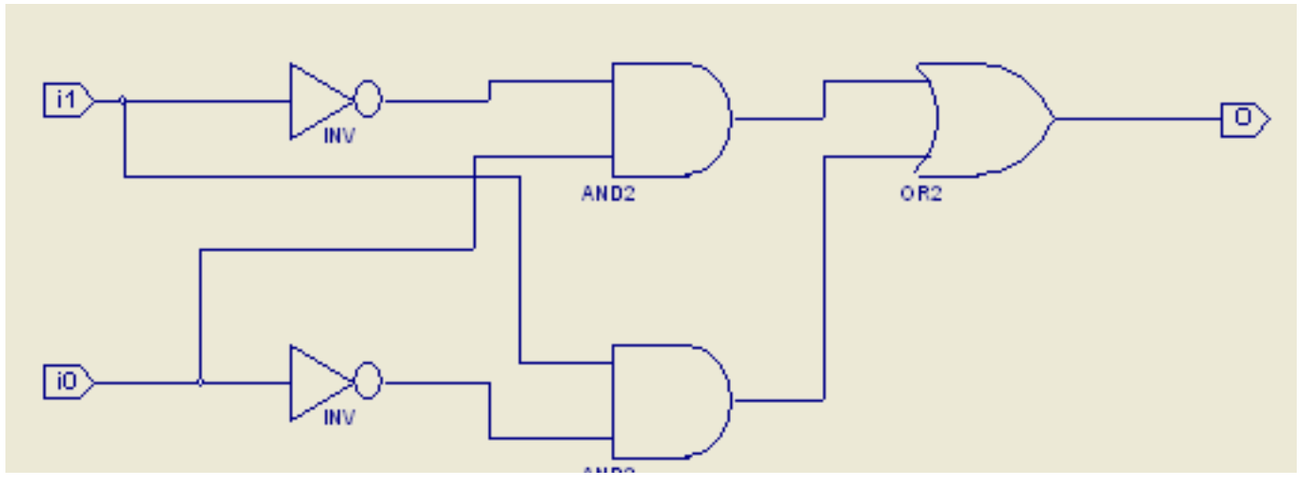




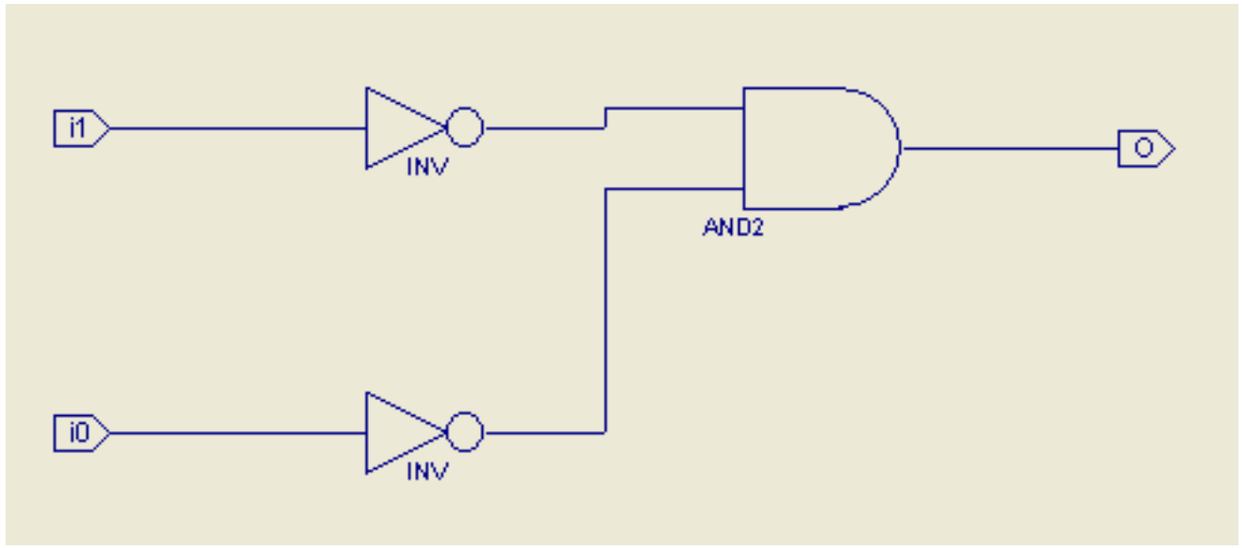
- Keep double clicking for more detail
- You can even see Karnaugh Maps

Some small design modifications

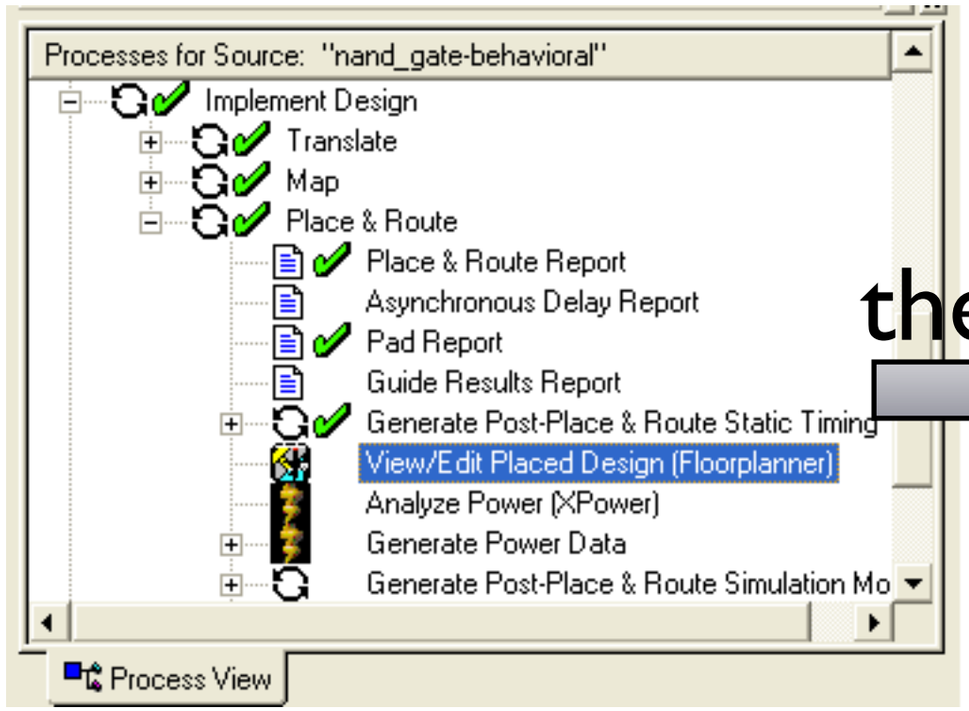
A XOR B



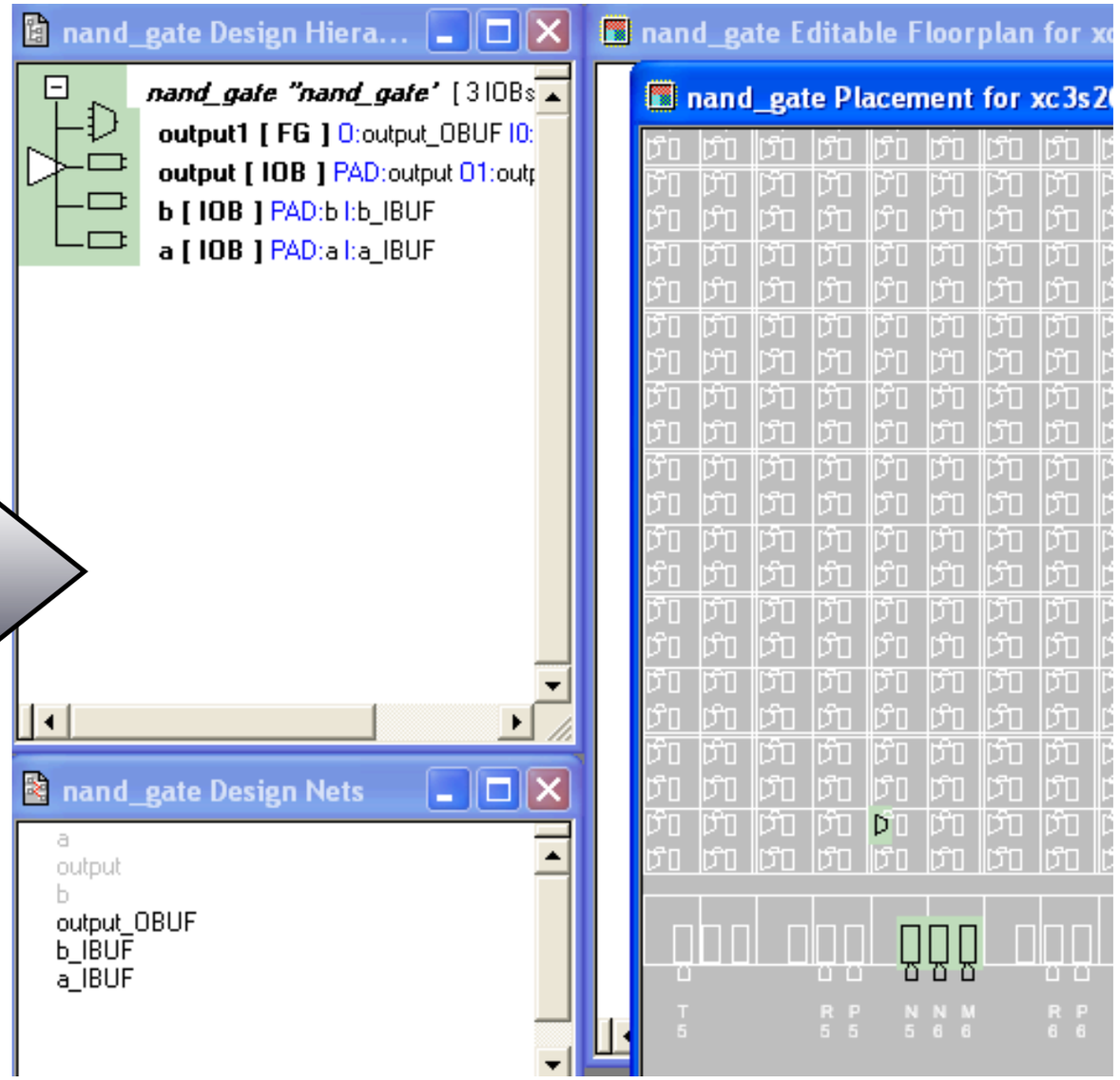
A NOR B



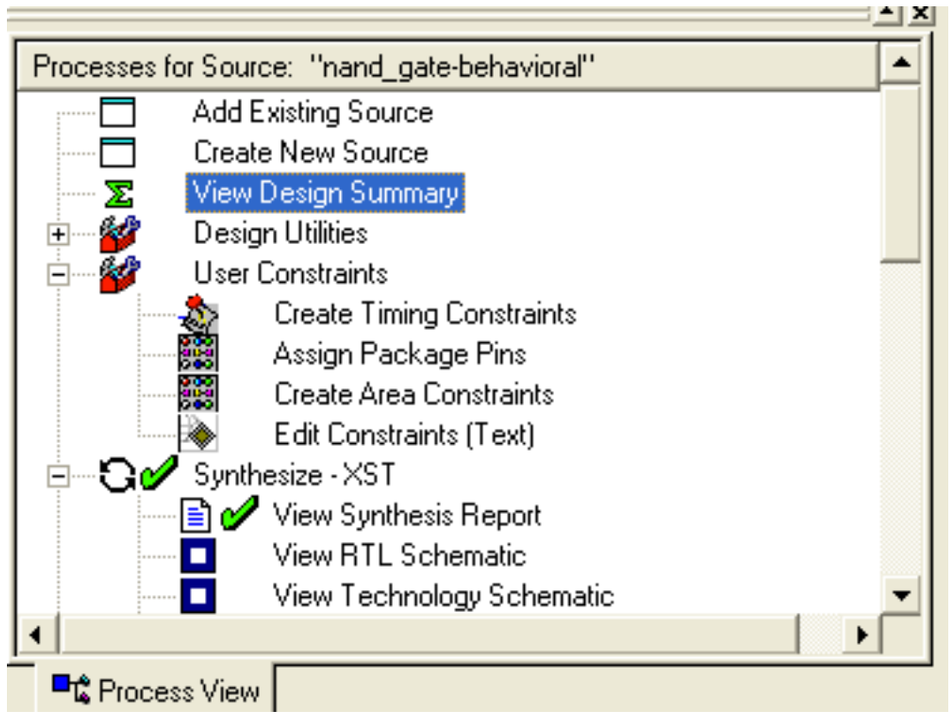
Viewing placement



then



Viewing resource utilization



then

Device Utilization Summary

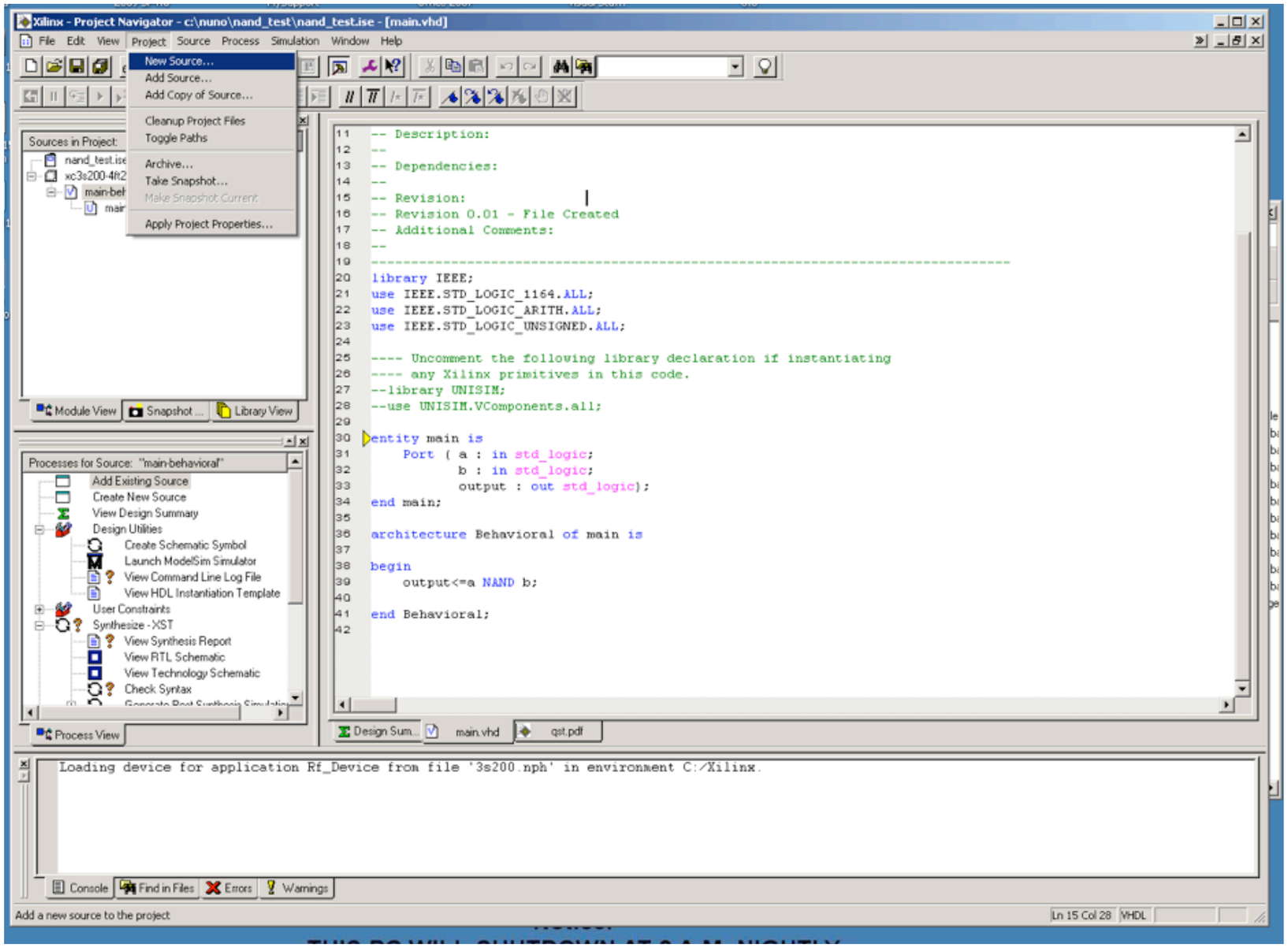
Logic Utilization	Used	Available	Utilization	Note(s)
Number of 4 input LUTs:	1	3,840	1%	
Logic Distribution:				
Number of occupied Slices:	1	1,920	1%	
Number of Slices containing only related logic:	1	1	100%	
Number of Slices containing unrelated logic:	0	1	0%	
Total Number of 4 input LUTs:	1	3,840	1%	
Number of bonded IOBs:	3	173	1%	

Waveform Simulation

Important note:
ISE 7.1 is an obsolete software, so you **CAN'T** request software licenses to run this feature on your own computer.

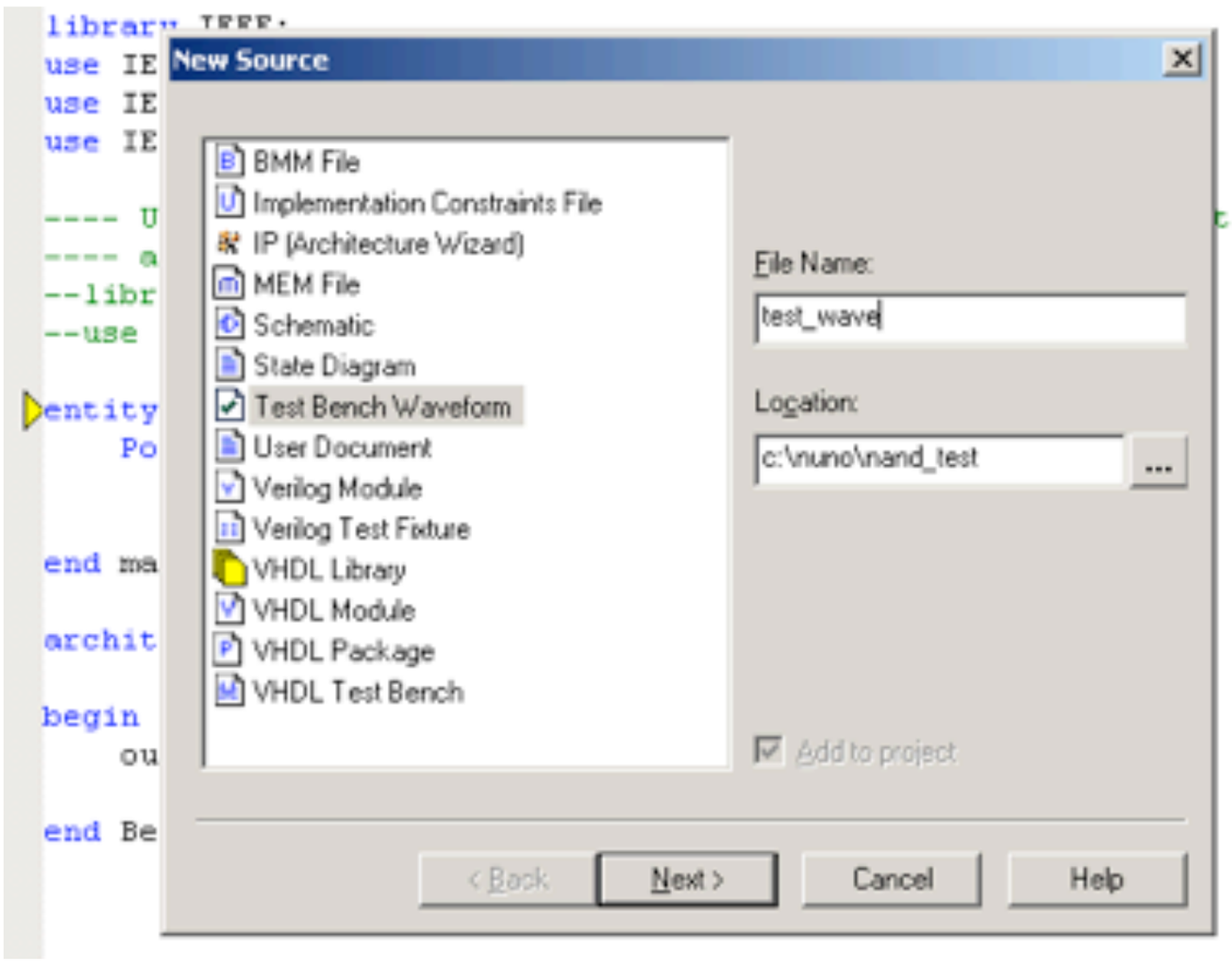
You can also simulate in the lab

- Add new source

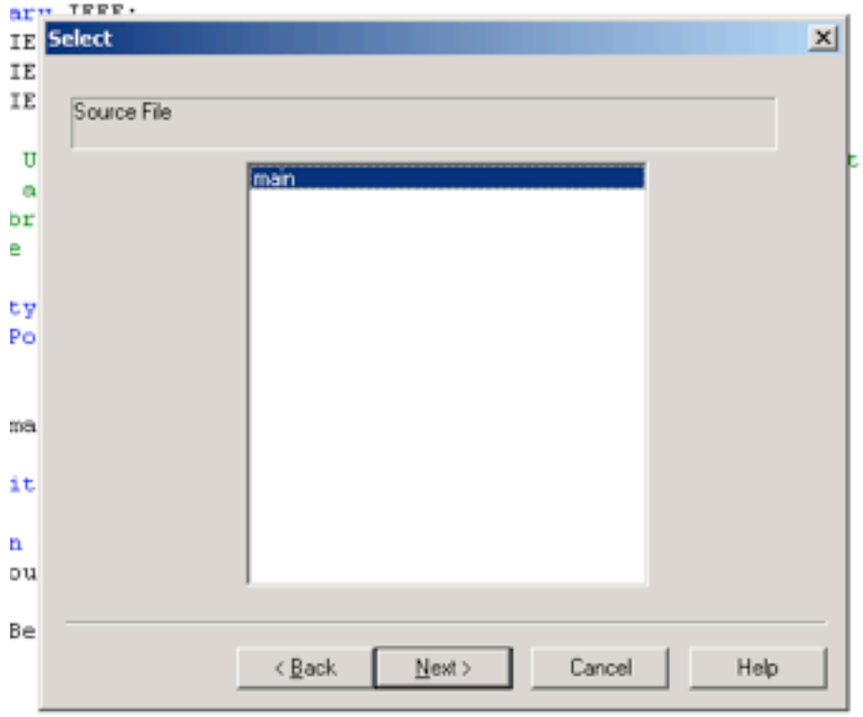


Adding test waveform

- *Test Bench Waveform*

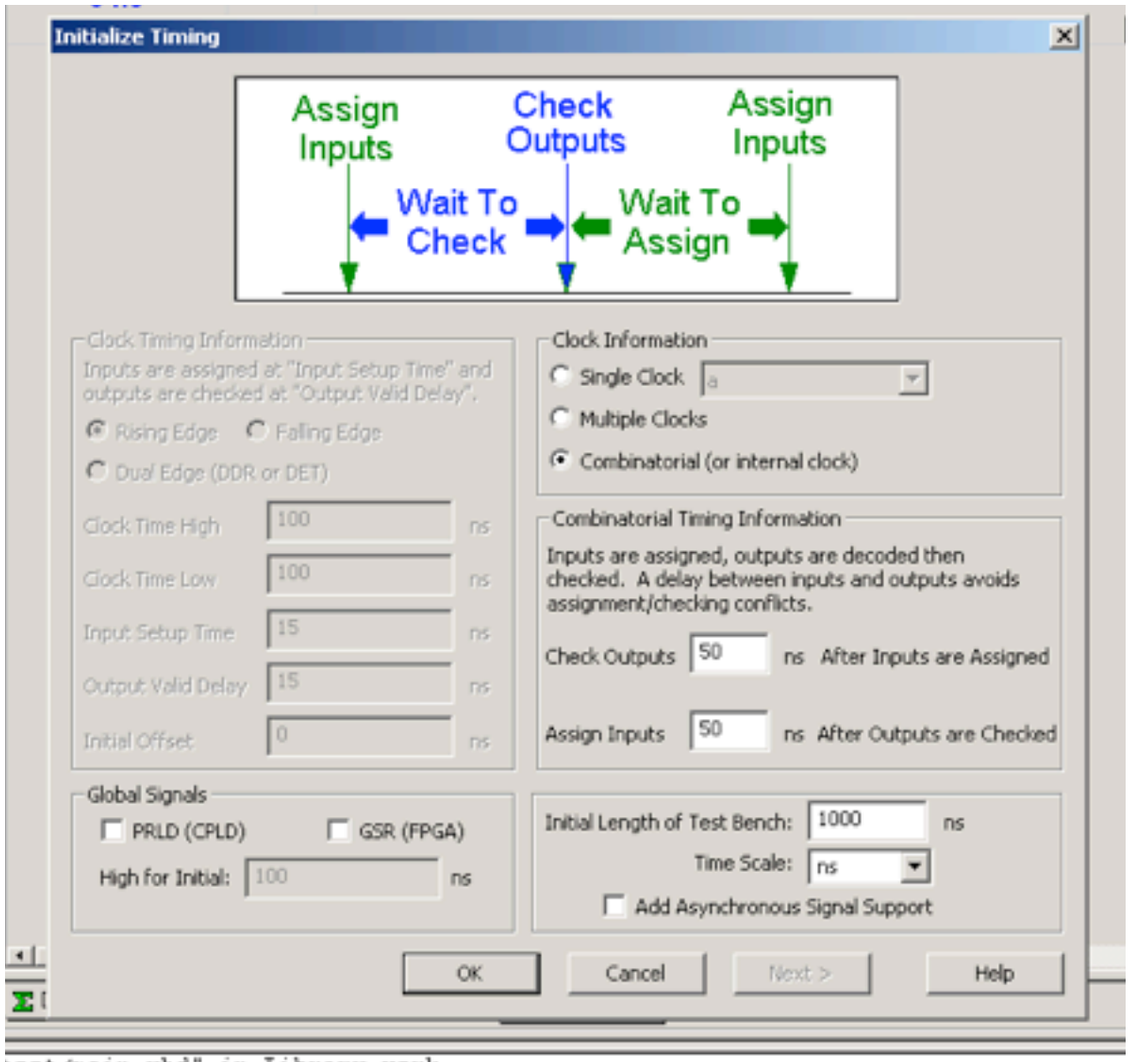


- Make sure you select the main source program.

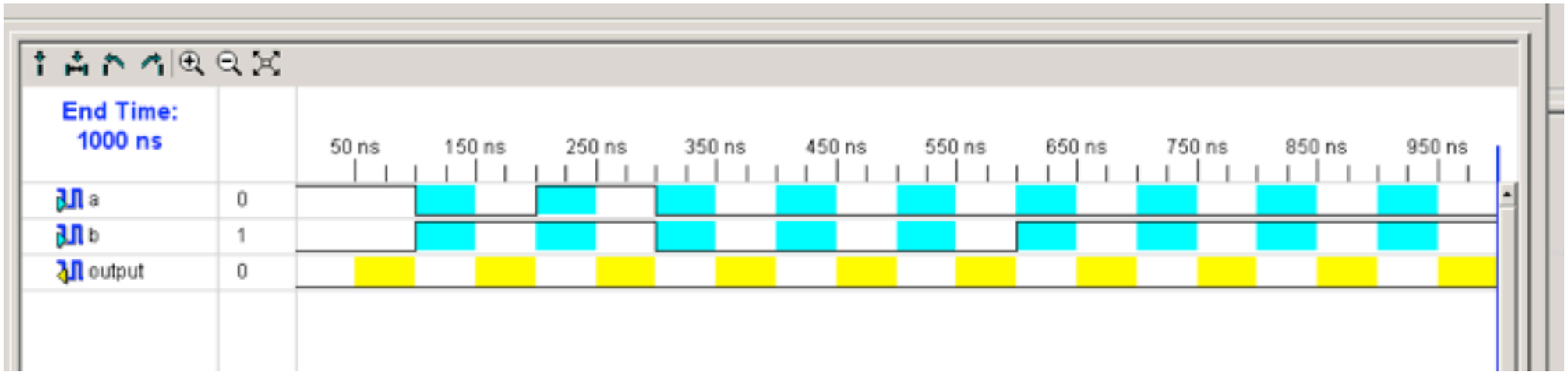


Internal clock selection

For this example we are not concerned about timings, so select *internal clock*



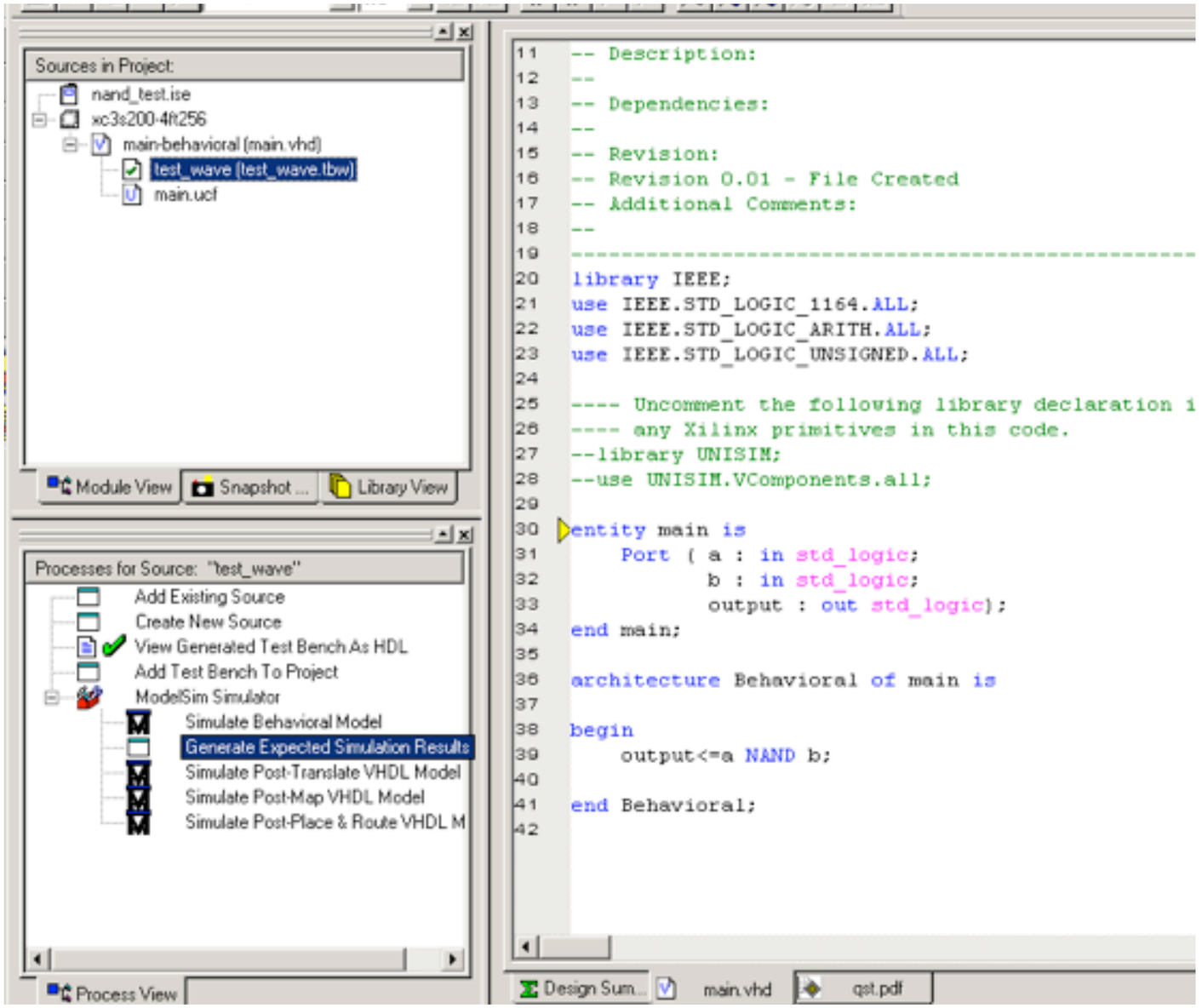
Manual input selection



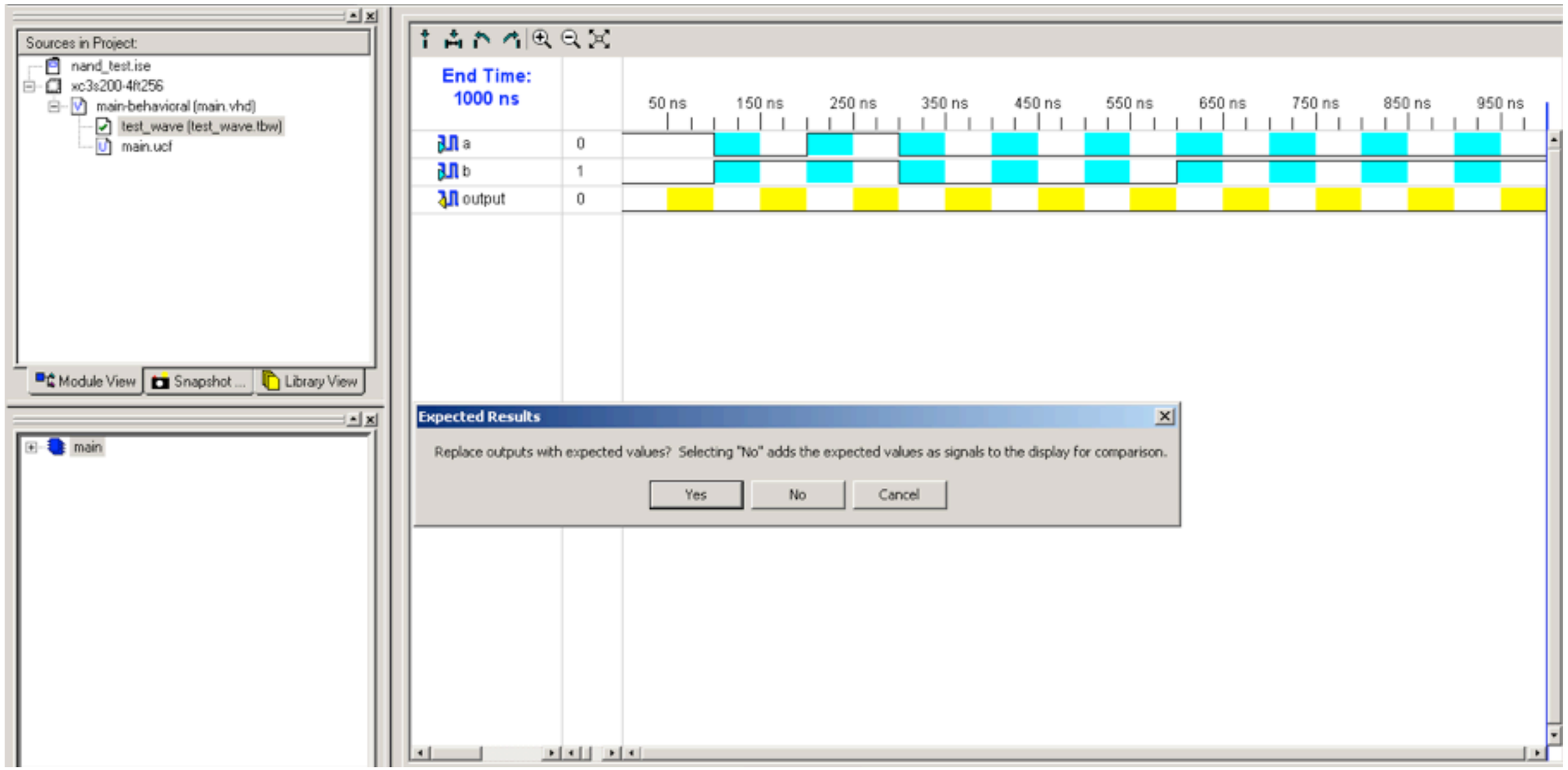
Click on the blue boxes to change the logic values for each signal.

Generating expected simulation results

- Go back to the main project window
- Select the wave file
- *Generate expected simulation results* which is under *ModelSim Simulator*



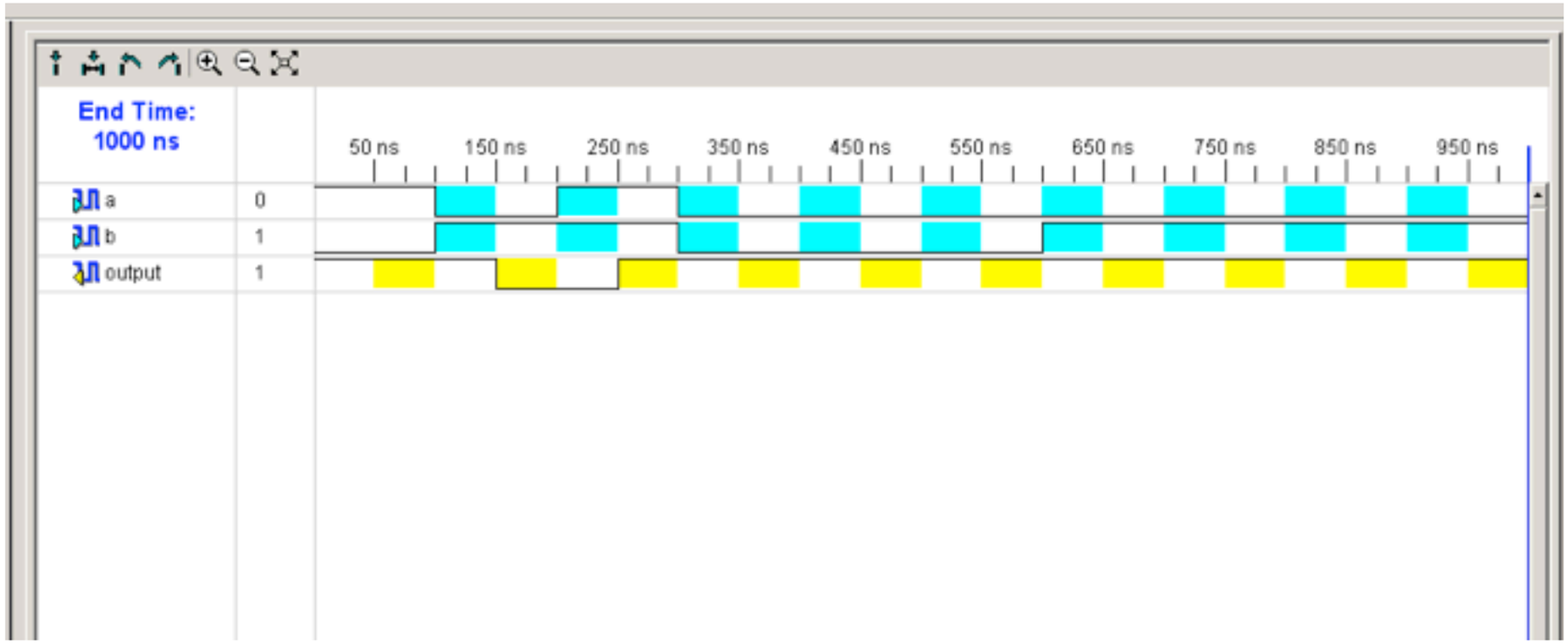
Adding the expected results to test waveform



Make sure you add the expected values to your wave (select *yes*)

Visualizing the simulation outcome

Make sure that the output wave matches your expected simulation outcome

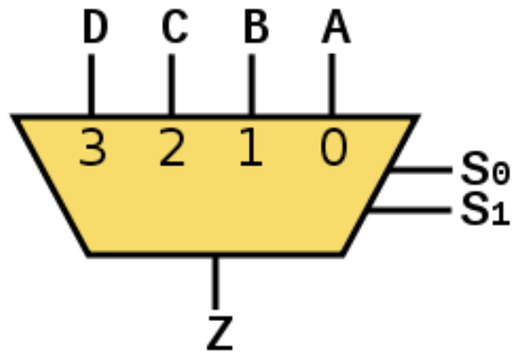


Practice Exercises

Gate vs functional level

Up to now, we described our circuits using logic gates...
for example look at the following 4-to-1 MUX

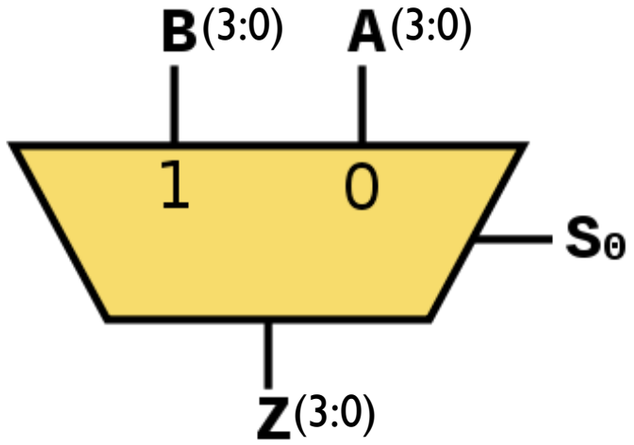
```
entity mux is
     port(a,b,c,d,s0,s1 : in bit;
     z : out bit);
end entity;

architecture myarch of mux is
begin
     z <=
     (a AND NOT(s0) AND NOT(s1))
     OR
     (b AND s0 AND NOT(s1))
     OR
     (c AND NOT(s0) AND s1)
     OR
     (d AND s0 AND s1);
end architecture ;
```



$$F = (A \cdot \overline{S_0} \cdot \overline{S_1}) + (B \cdot S_0 \cdot \overline{S_1}) + (C \cdot \overline{S_0} \cdot S_1) + (D \cdot S_0 \cdot S_1)$$

However, its is not very practical. VHDL also can describe the function of the circuit without logic gates.

Exercise #1- SPARTAN-3 FPGA implementation of a 2-to-1 MUX



$$Z = (A \cdot \overline{S}) + (B \cdot S)$$

- Inputs **A** and **B** are a 4-bit bus
- Fix the code on the left and map each 4-bit bus to different sets of switches
- Map S0 to button **BTN0**
- Map the output to 4 LEDs

```
library ieee;
use ____________________________;

entity mux is
    port (___, ___: ___ std_logic_vector(3
downto 0);
          sel: in ____________________;
          ___: out std_logic_vector(3 downto
0));
end mux;

architecture example of _____ is
begin
    process (a, b, sel)
    begin
       if (sel='0') then
           c <= a;
       elsif (___________) then
           c <= _____;
       end if;
    end ___________;
end ___________;
```