

PART I: Theory

WHAT IS AN FPGA?

Field-Programmable Gate Arrays

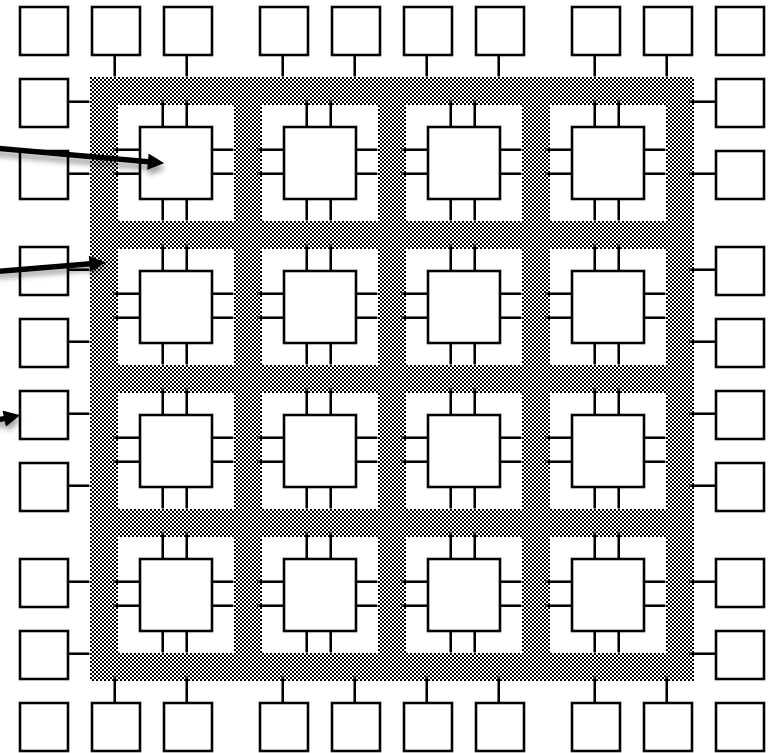
- A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing.
- In short, programmable circuit.



Image Courtesy: [https://upload.wikimedia.org/wikipedia/commons/4/4e/Xilinx_Spartan-3E_\(XC3S500E\).jpg](https://upload.wikimedia.org/wikipedia/commons/4/4e/Xilinx_Spartan-3E_(XC3S500E).jpg)

Field-Programmable Gate Arrays

- Logic blocks
 - to implement combinational and sequential logic
- Interconnect
 - wires to connect inputs and outputs to logic blocks
- I/O blocks
 - special logic blocks at periphery of device for external connections



Slide Courtesy: <http://courses.cs.washington.edu/courses/cse467/00wi/lectures/ppt/FPGAIntro/FPGAIntro.ppt>.

Applications of FPGA

- Aerospace and Defense
 - Communications, Missiles, Mars Rovers
- ASIC Prototyping
- Consumer Electronics
 - Digital displays, digital camera
- Data Centers
 - Servers, Routers
- High Performance Computing
- ...

Plan

1. The tool: Verilog HDL
2. The theory: FPGA Design and Implementation
3. The exemplary project: Lab 1 Sequencer

FPGA DESIGN AND IMPLEMENTATION FUNDAMENTALS

FPGA Design Fundamentals

- Step 1 – Design
 - Know what it is that you want to implement, e.g. an adding machine, or a traffic controller
 - Module-level diagrams and interactions between modules
 - Control logic and state machine drawings
 - Understand how your FPGA design will interact with the physical world, e.g. Ethernet, VGA, LCD.
 - Plan **everything** out before writing a single line of code! Explain the plan to someone else.

FPGA Design Fundamentals

- Step 2 – Implementation
 - Translate your plan to source code!
 - Express each module in HDL source code
 - Connect the modules in hierarchical order like building LEGO blocks. You should end up with a single top-level file.
 - Use any text editor (even Notepad or Wordpad will do) as long as the file name ends with “.v”

FPGA Design Fundamentals

- Step 3 – Simulation
 - Simulation is the single most important debugging tool you will ever use in a FPGA design
 - You will have access to real-time debugging tools (e.g. chipScope) but simulation is far easier to find and fix the bugs.

FPGA Design Fundamentals

- Step 4 – Logic Synthesis
 - Once the bugs are out, a logic synthesis tool analyzes the design and generates a netlist with common cells available to the FPGA target
 - The netlist should be functionally equivalent to the original source code.
 - We will use ISE's XST to synthesize the project

FPGA Design Fundamentals

- Step 5 – Technology Mapping
 - The synthesized netlist is mapped to the device-specific libraries.
 - The result is another netlist that's closer to the final target device.
 - On ISE this is performed by NGDBUILD

FPGA Design Fundamentals

- Step 6 – Cell Placement
 - The cells instantiated by final netlist are placed in the FPGA layout, i.e. each cell is assigned a physical location on the target device.
 - Can be a time-consuming process depending on the size of the design and complexity of timing and physical constraints.
 - On ISE this process is done by the program MAP (i.e. map to physical location)

FPGA Design Fundamentals

- Step 7 – Route
 - Often referred to as “Place-and-Route” in combination with cell placement.
 - In this process, the placement tool determines how to connect (“route”) the cells in the device to match the netlist
 - Can be a time-consuming process depending on the size of the design and complexity of timing and physical constraints.
 - Done by program PAR on ISE.

FPGA Design Fundamentals

- Step 8 – Bitstream Generation
 - A placed and routed design can be used to produce a programming file to program the FPGA.
 - The programming file is called a “bitstream.” It contains everything there is about how to configure the cells and connecting them.
 - Done by program BITGEN on ISE.
 - Now you have a “compiled” FPGA design.

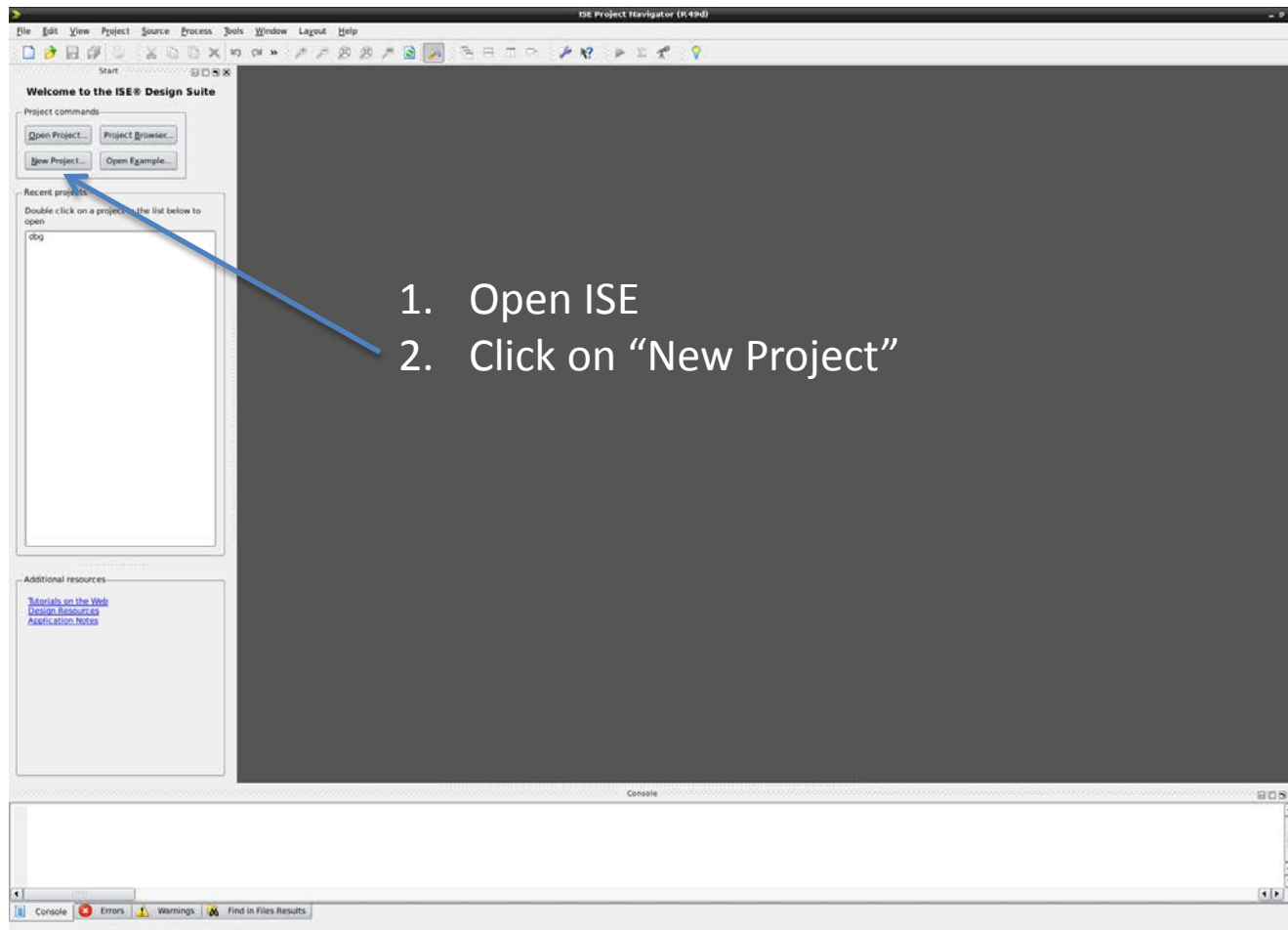
Tools of Trade

- Text editor of choice
- Simulator
 - ISE Webpack provides ISIM
 - Alternatively use free Modelsim PE
- Synthesis
 - ISE Webpack provides XST
 - Alternatively use Synplify Pro (evaluation version)
- Map, Place-and-Route
 - ISE Webpack

PART II: Practice

EXAMPLE PROJECT IMPLEMENTATION

Create an ISE Project



New Project Dialogue

Create New Project
Specify project location and type.

Enter a name, locations, and comment for the project—

Name:

Location:

Working Directory:

Description:

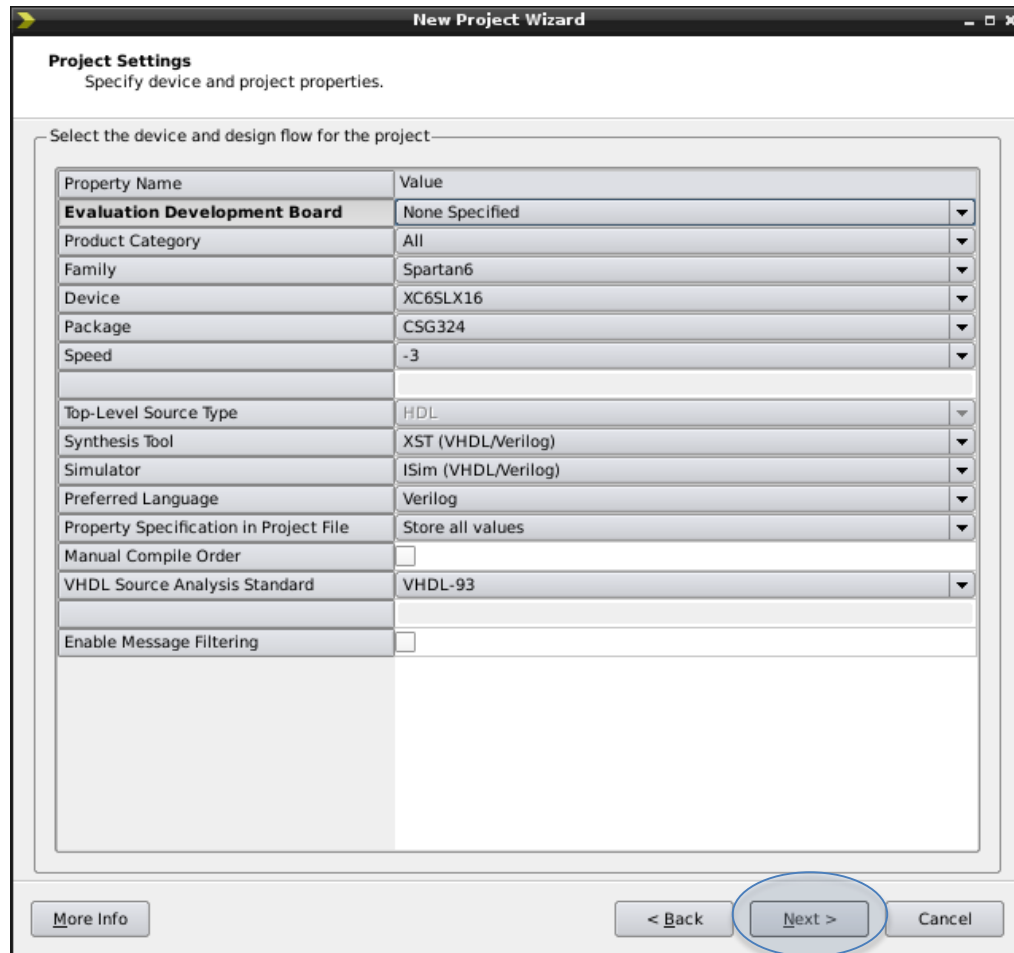
Select the type of top-level source for the project—

Top-level source type:

- No spaces in the path!
- Choose “HDL” project

Device Properties

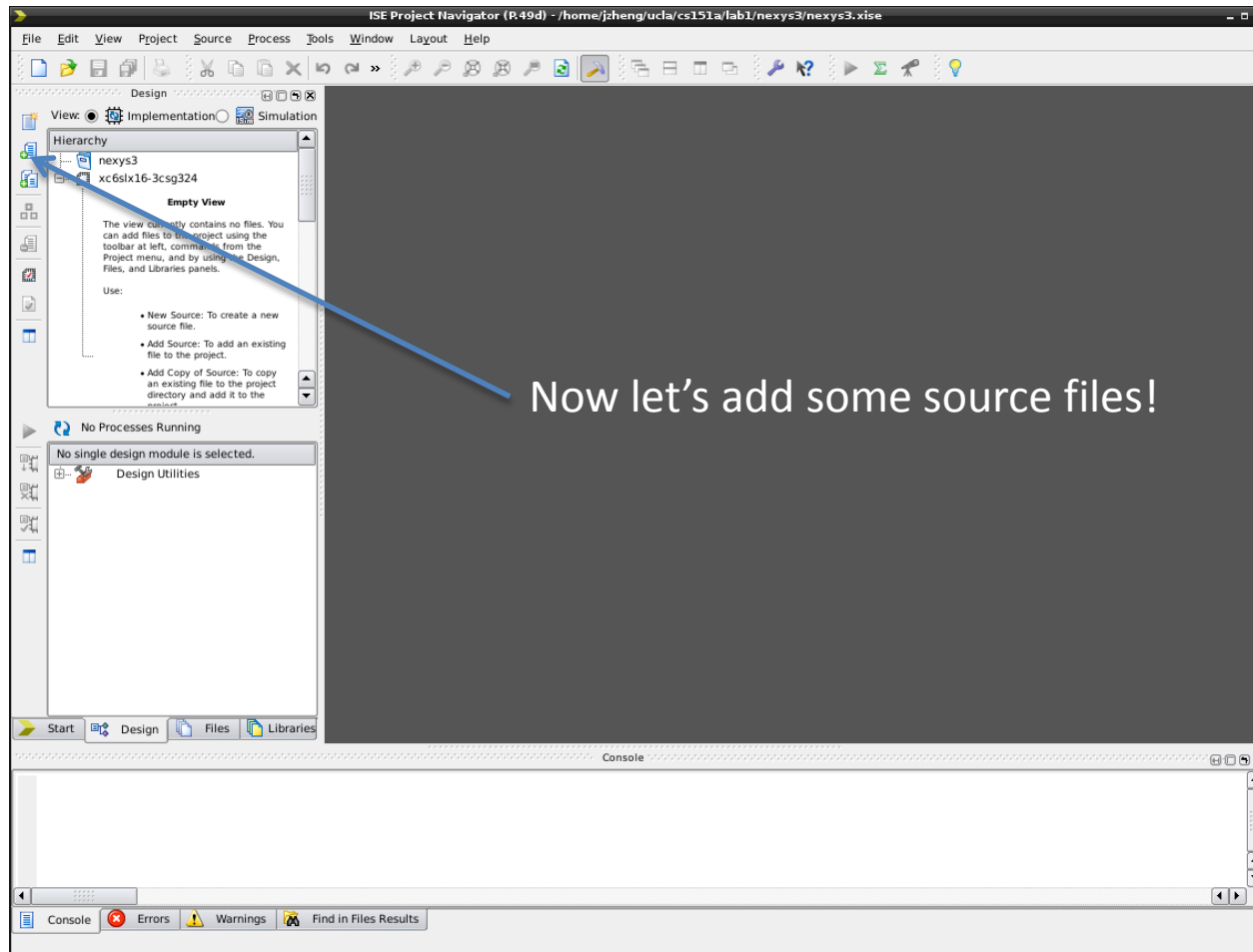
Make sure the fields match what you see here



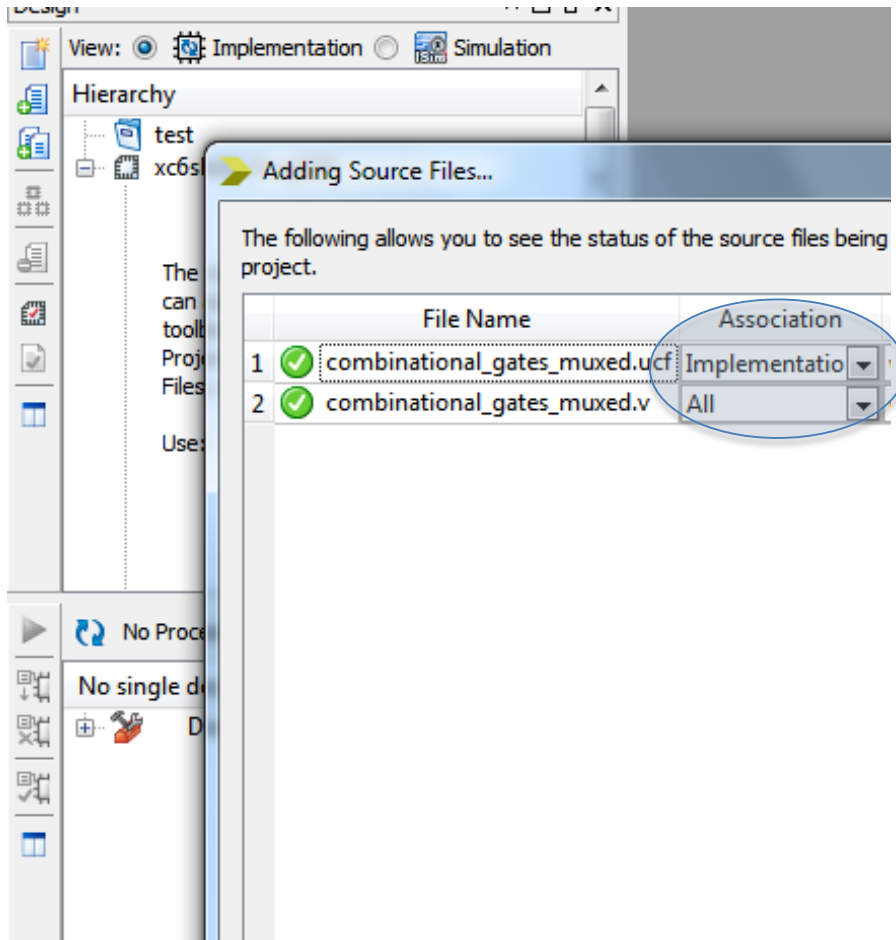
The image shows a screenshot of the 'New Project Wizard' dialog box, specifically the 'Project Settings' tab. The dialog is titled 'New Project Wizard' and has a subtitle 'Specify device and project properties.' Below the subtitle, there is a section titled 'Select the device and design flow for the project'. This section contains a table with two columns: 'Property Name' and 'Value'. The table lists various project settings, including device selection, synthesis tool, and language. At the bottom of the dialog, there are three buttons: '< Back', 'Next >', and 'Cancel'. The 'Next >' button is circled in blue.

Property Name	Value
Evaluation Development Board	None Specified
Product Category	All
Family	Spartan6
Device	XC6SLX16
Package	CSG324
Speed	-3
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"/>

Project Created



Add Source Files



1. Click on “Add sources”
2. Select
combinational_gates_m
uxed.v,
combinational_gates_m
uxed.ucf
3. Make sure the file
association is correct

Source Files

- .v files are Verilog source code
- .ucf files are *User Constraint Files*

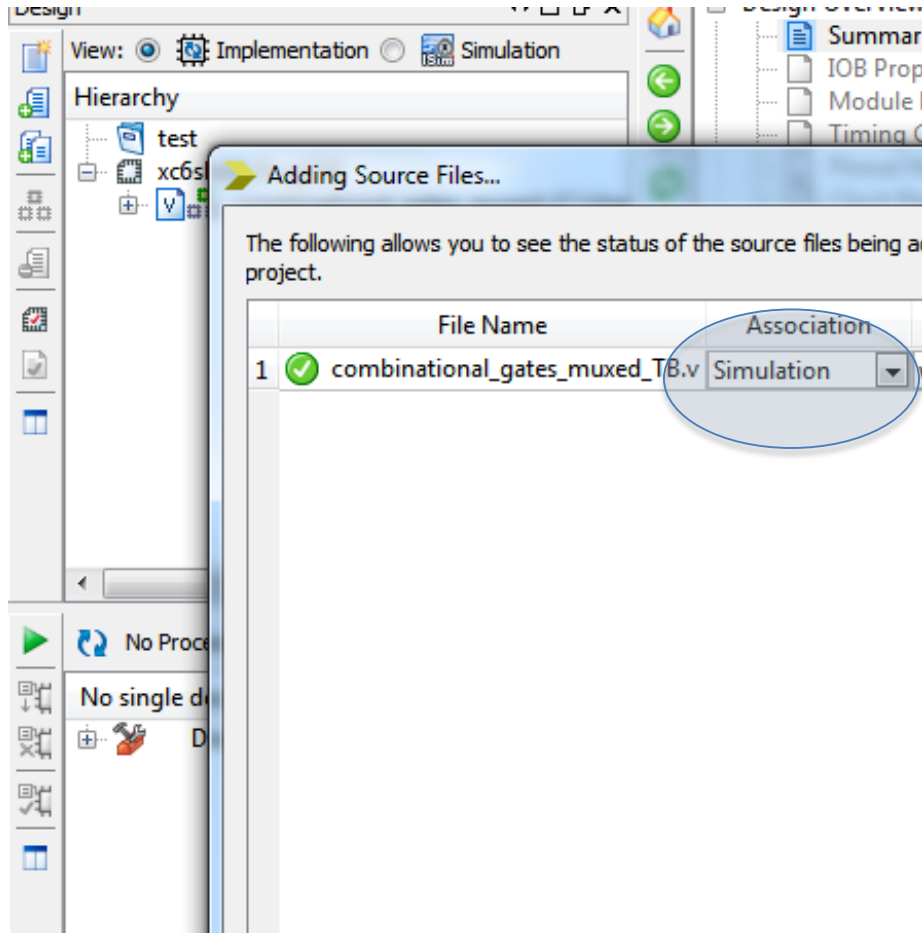
UCF lists all the available pin mappings in the FPGA in the following format:

```
Net "your_signal_name<bit_index>" LOC = XX | IOSTANDARD =  
    LVCMOS33; # More details about the pin
```

For example:

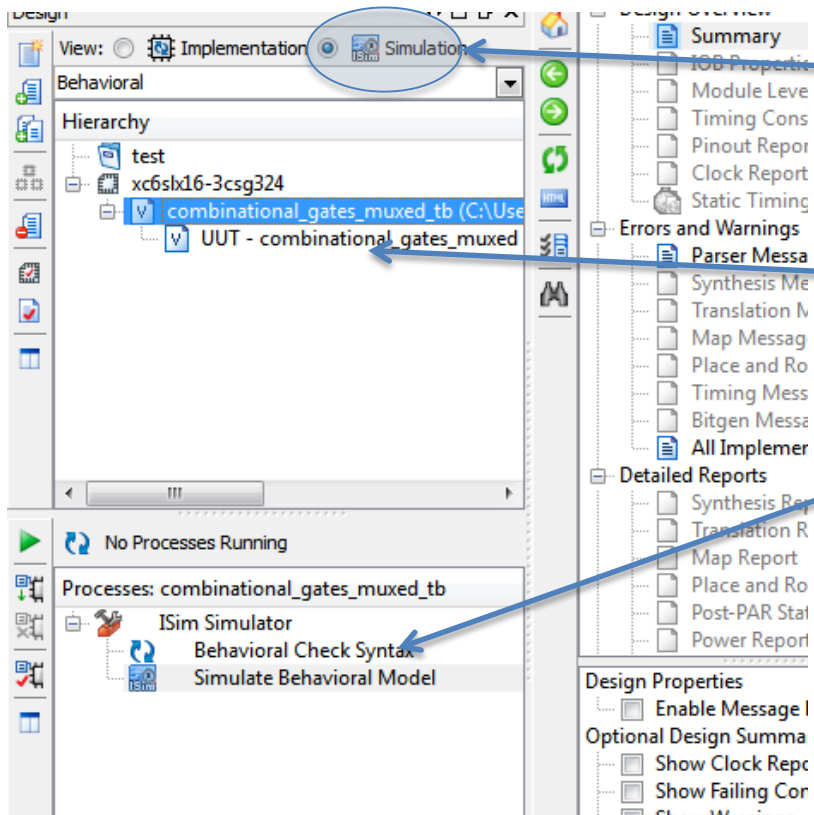
```
Net "sw<0>" LOC = T10 | IOSTANDARD = LVCMOS33; #Bank = 2,  
    pin name = IO_L29N_GCLK2, Sch name = SW0
```


Add Simulation Files



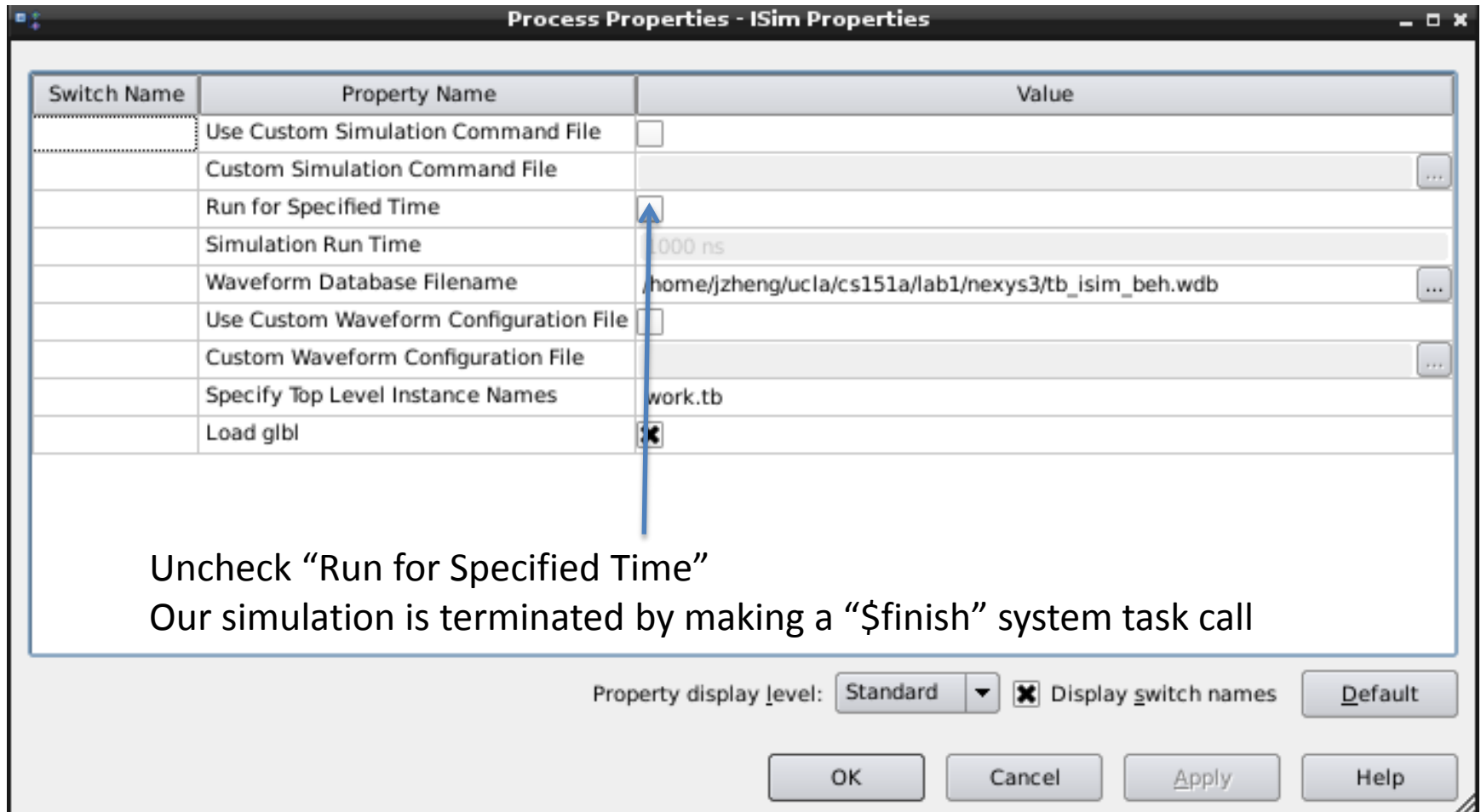
1. Click on “Add sources” again
2. Select the combinational_gates_muxed_TB.v
3. Make sure the file association is correct

Almost Ready for Simulation!



1. Switch to simulation view
2. Select ...TB.v from Hierarchy view
3. Right click on “Simulate Behavioral Model” in process view
4. Click on “Process Properties”

ISIM Process Properties



Switch Name	Property Name	Value
	Use Custom Simulation Command File	<input type="checkbox"/>
	Custom Simulation Command File	...
	Run for Specified Time	<input checked="" type="checkbox"/>
	Simulation Run Time	1000 ns
	Waveform Database Filename	/home/jzheng/ucla/cs151a/lab1/nexys3/tb_isim_beh.wdb
	Use Custom Waveform Configuration File	<input type="checkbox"/>
	Custom Waveform Configuration File	...
	Specify Top Level Instance Names	work.tb
	Load glbl	<input checked="" type="checkbox"/>

Uncheck "Run for Specified Time"
Our simulation is terminated by making a "\$finish" system task call

Property display level: Standard ☐ Display switch names

Launch ISIM

- Right click on “Simulate Behavioral Model” again, this time choose “run all”
- ISIM will be launched
- ISIM is the simulation environment where you can dynamically debug the circuit, much like a software debugger
- Your main focus should be on the console window and the waveform window

ISIM Main Window

Hierarchical
View

Signal
View

Waveform

The screenshot displays the Xilinx ISIM Main Window with three primary views visible:

- Hierarchical View:** Located on the left, it shows a tree structure of the design hierarchy. The selected object is 'Initial_49_3' under the 'combinational_gates_muxed_tb' instance.
- Signal View:** Located in the middle, it shows a table of simulation objects for 'Initial_49_3'. The selected object is 'sw_T[4:0]' with a value of '11111'.
- Waveform View:** Located on the right, it shows the Verilog source code for the testbench. The code includes a register 'sw_T' and a wire 'led_T', and it instantiates the 'combinational_gates_muxed' module.

The bottom of the window shows the Console, which displays the following text:

```
ISim P.68d (signature 0x7708f090)
WARNING:Security:42 - Your software subscription period has lapsed. Your current version of Xilinx tools will continue to function, but you no longer qualify for Xilinx software updates or new releases.

This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
Stopped at time : 160 ns : File "C:/Users/152/Desktop/test/test/combinational_gates_muxed_TB.v" Line 50
ISim>
```

Console

Post Simulation Examination

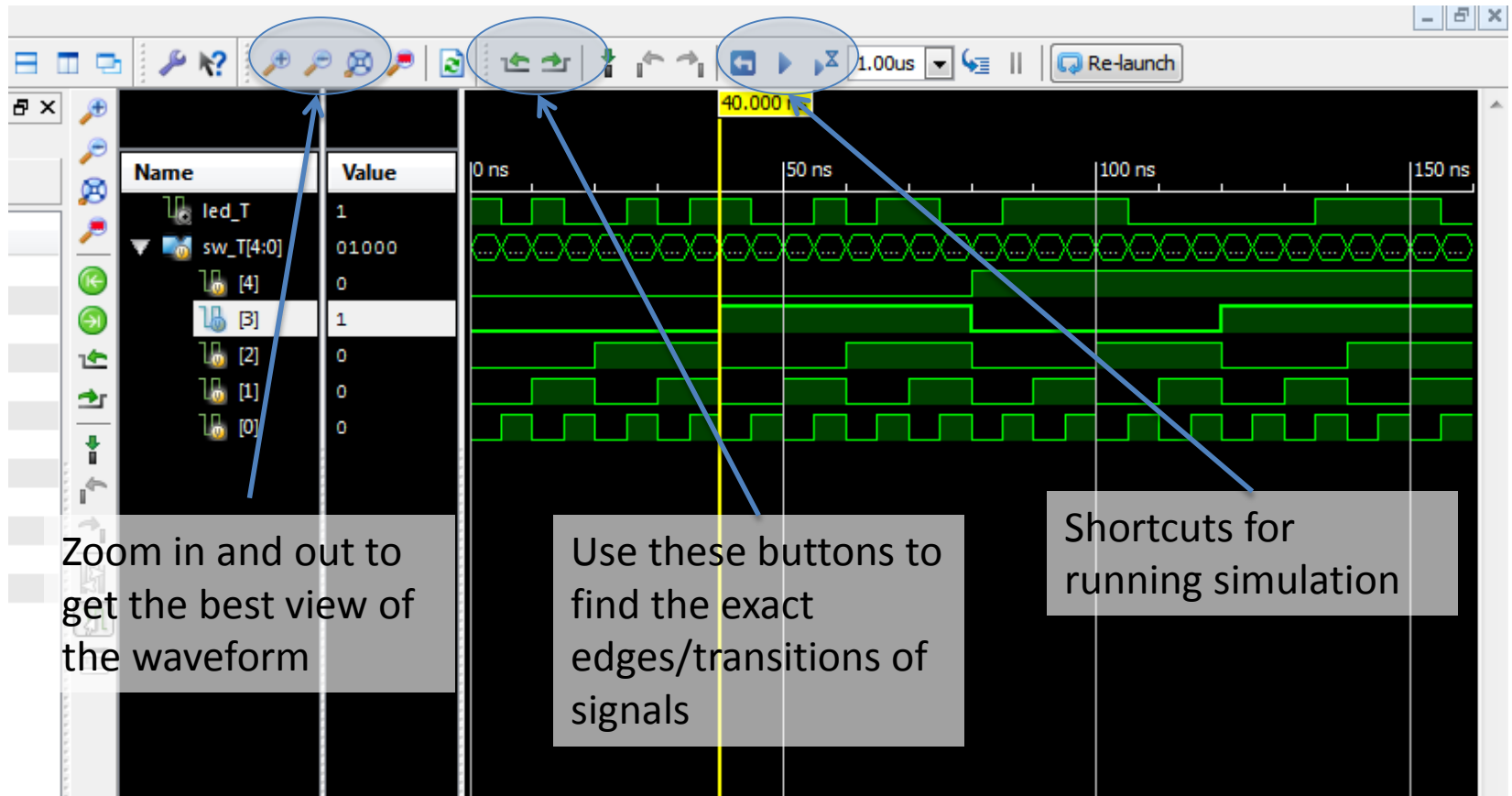
The screenshot displays a digital logic simulation tool interface. The 'Objects' panel on the left lists simulation objects for 'UUT'. The 'Value' column shows the current state of these objects. A blue arrow points from the 'sw_T[4:0]' signal in the waveform view to the 'sw[4:0]' object in the 'Objects' panel.

Object Name	Value
sw[4:0]	11111
nand_out	0
and_out	1
nor_out	0
or_out	1
xor_out	0
xnor_out	1
buff_out	1
not_out	0
Muxn[7:0]	01010110
SelectIn[2:0]	111
led	0

To debug, you can add any signals to the waveform view by right click and select add

Name	Value
led_T	0
sw_T[4:0]	11111

Post Simulation Examination



Ready for Synthesis

The screenshot shows the Xilinx ISE software interface. The 'Design' tab is active, and the 'Implementation' view is selected in the 'View' dropdown. The 'Design Overview' panel on the right shows the 'Summary' section, which includes IOB Properties, Module Level Utilization, Timing Constraints, Pinout Report, Clock Report, Static Timing, Errors and Warnings, Parser Messages, Synthesis Messages, Translation Messages, Map Messages, Place and Route Messages, Timing Messages, Bitgen Messages, and All Implementation Messages. The 'Design Properties' panel at the bottom shows the 'Optional Design Summary Contents' section, which includes checkboxes for 'Show Clock Report', 'Show Failing Constraints', 'Show Warnings', and 'Show Errors'. The 'Design Summary/Reports' panel on the left shows the 'Synthesize - XST' option selected. The 'Errors' panel at the bottom is empty.

Switch back to implementation view if you haven't already

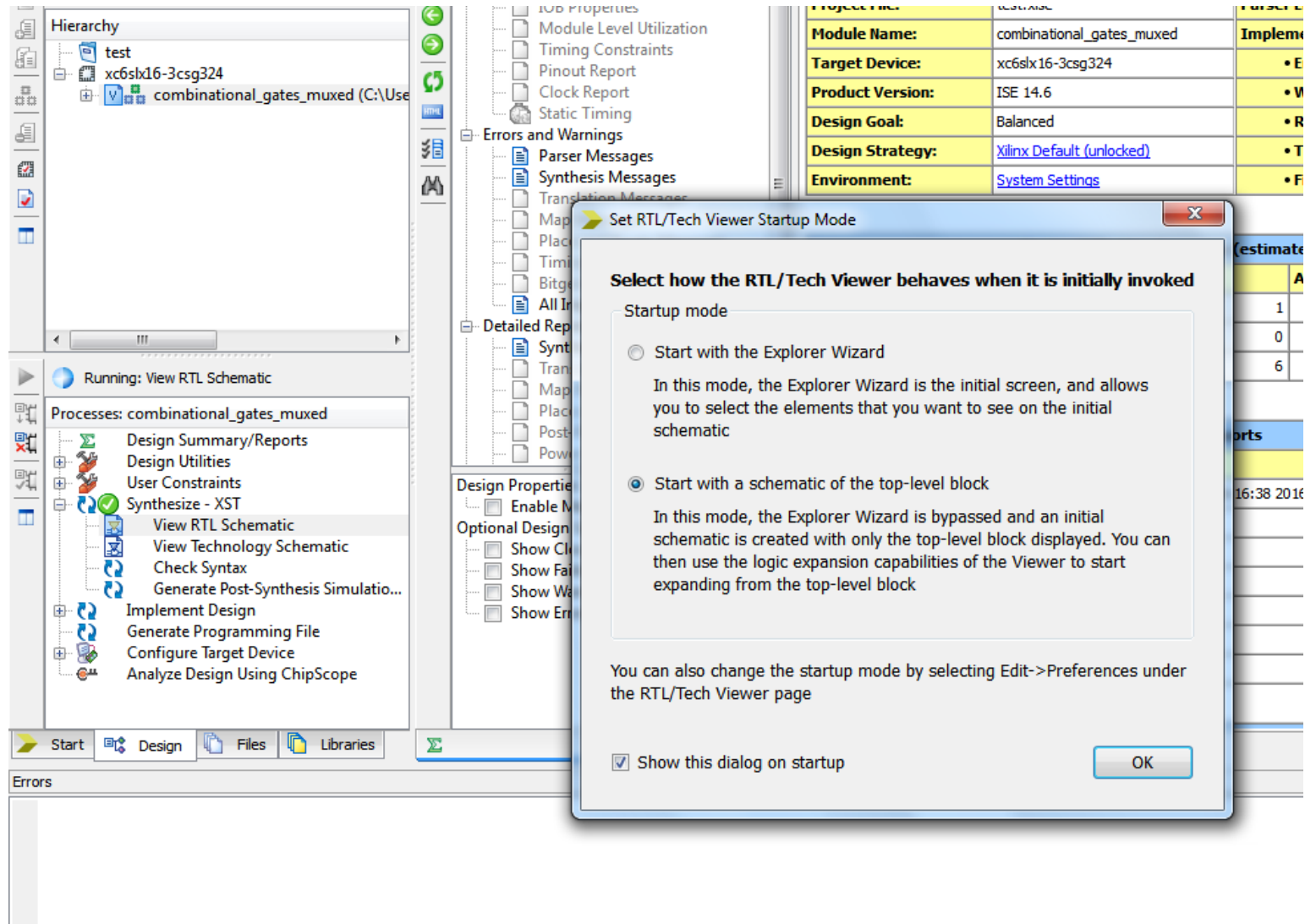
Double click on synthesis to start. You can also view the auto generated schematics here

combinational_gates_muxed Project Status				
Project File:	test.xise	Parser Errors:		
Module Name:	combinational_gates_muxed	Implementation State:		
Target Device:	xc6slx16-3csg324	• Errors:		
Product Version:	ISE 14.6	• Warnings:		
Design Goal:	Balanced	• Routing Results:		
Design Strategy:	Xilinx Default (unlocked)	• Timing Constrai		
Environment:		• Final Timing Sco		

Detailed Reports				
Report Name	Status	Generated	Errors	Warn
Synthesis Report				
Translation Report				
Map Report				
Place and Route Report				
Power Report				
Post-PAR Static Timing Report				
Bitgen Report				

Secondary Reports		
Report Name	Status	Generated
Date Generated: 09/28/2016 - 16:10:44		

View Schematics



Place-n-Route and Bitstream

Design Overview

- Summary
 - IOB Properties
 - Module Level Utilization
 - Timing Constraints
 - Pinout Report
 - Clock Report
 - Static Timing
- Errors and Warnings
 - Parser Messages
 - Synthesis Messages
 - Translation Messages
 - Map Messages
 - Place and Route Messages
 - Timing Messages
 - Bitgen Messages
 - All Implementation Messages
- Detailed Reports
 - Synthesis Report
 - Translation Report
 - Map Report
 - Place and Route Report
 - Post-PAR Static Timing Report
 - Power Report

Design Properties

- ☐ Enable Message Filtering

Optional Design Summary Contents

- ☐ Show Clock Report
- ☐ Show Failing Constraints
- ☐ Show Warnings
- ☐ Show Errors

combinational_gates_muxed

Project File:	test.xise
Module Name:	combinational_gates_muxed
Target Device:	xc6slx16-3csg324
Product Version:	ISE 14.6
Design Goal:	Balanced
Design Strategy:	Xilinx Default (unlocked)
Environment:	System Settings

Device Utilization Summary

Logic Utilization	Used
Number of Slice LUTs	
Number of fully used LUT-FF pairs	
Number of bonded IOBs	

Detailed Reports

Report Name	Status	Generated
Synthesis Report	Current	Wed Sep 28
Translation Report		
Map Report		
Place and Route Report		
Power Report		
Post-PAR Static Timing Report		
Bitgen Report		

Processes: combinational_gates_muxed

- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
- Implement Design**
- Generate Programming File
- Configure Target Device
- Analyze Design Using ChipScope

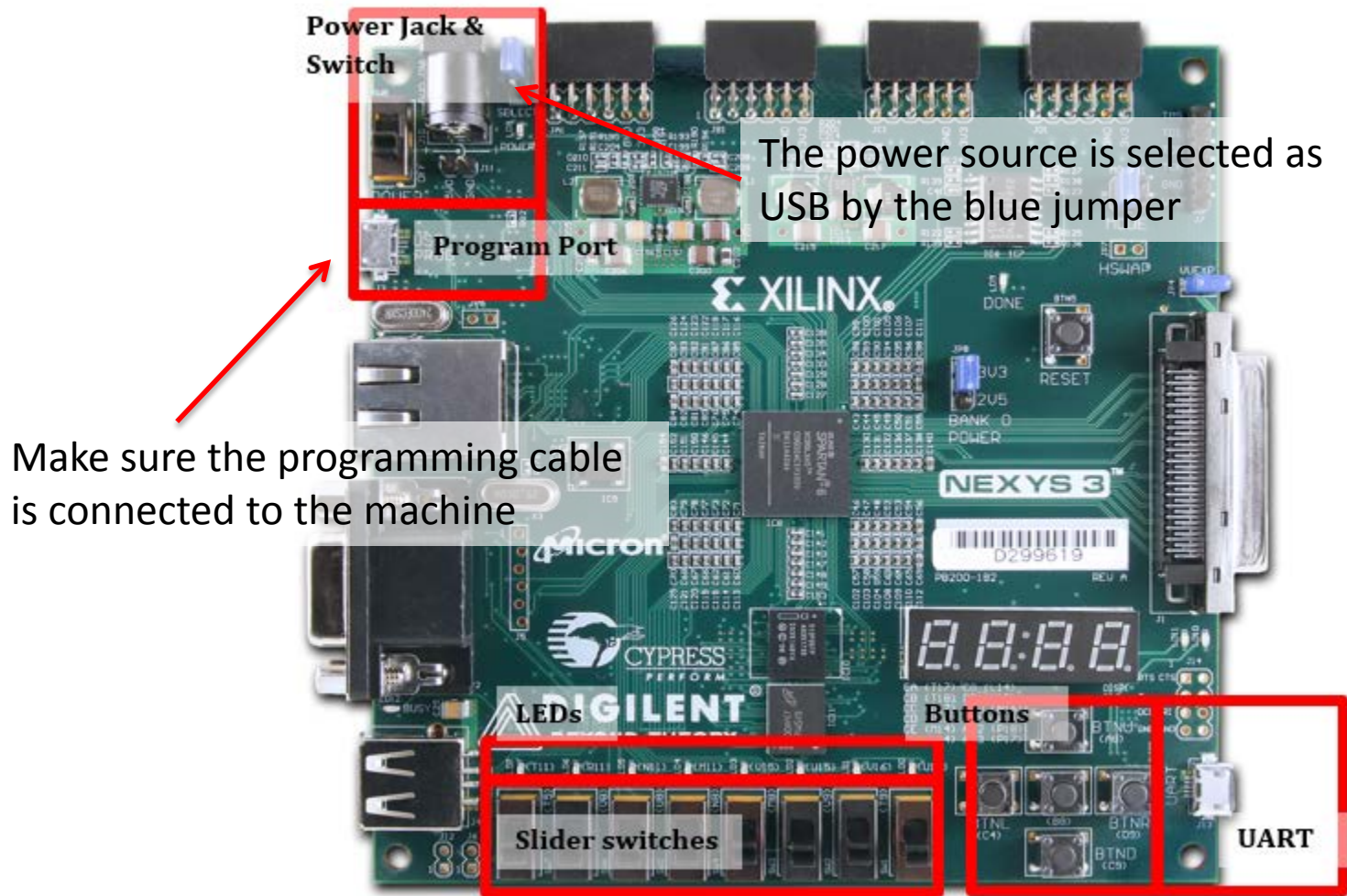
Start Design Files

Double click on "Implement Design" and then "Generate Programming File"

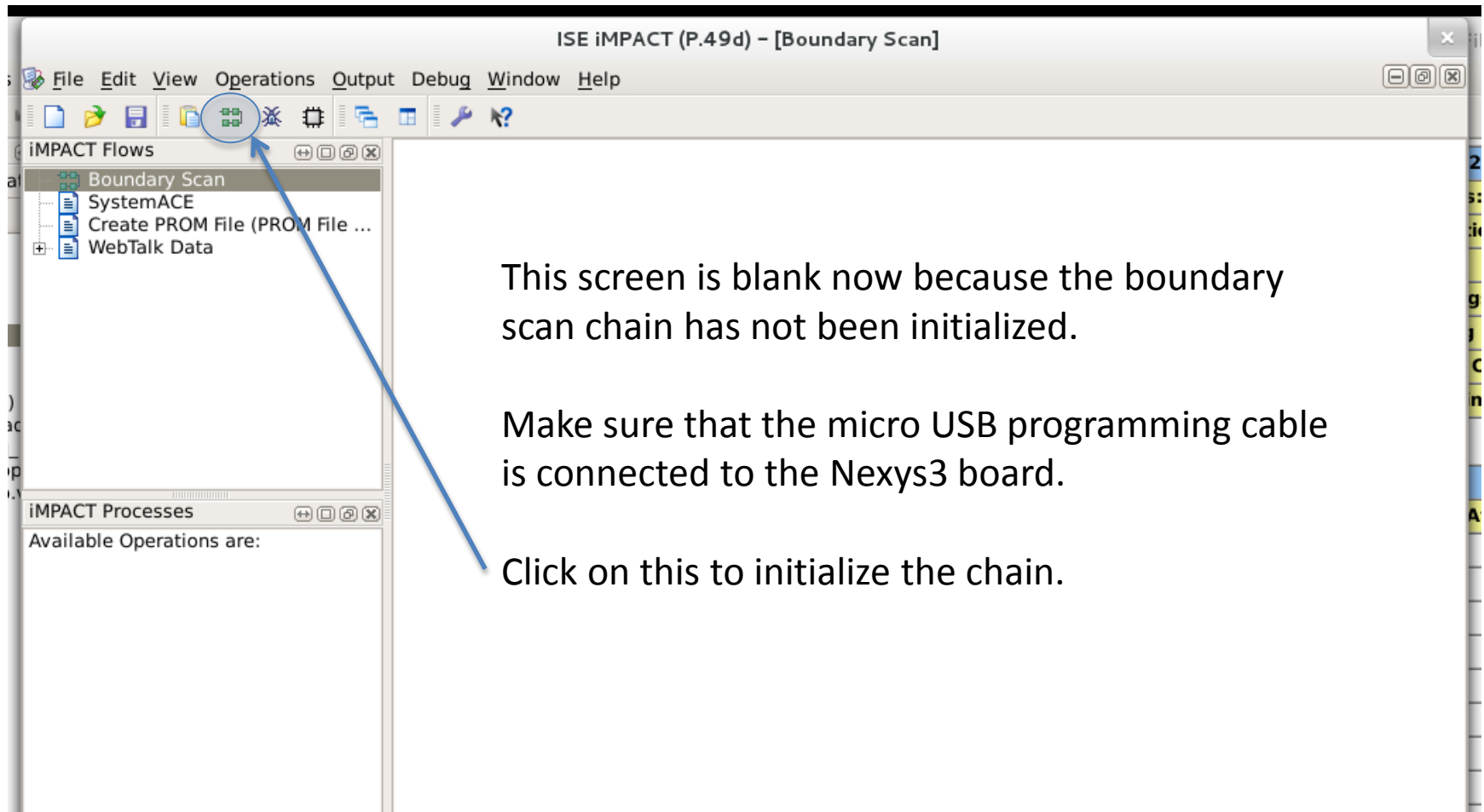
Download Bitstream to FPGA

- By now you should have a `top_module_name.bit(combinationa_gate_muxed.bit)` file generated in the project folder
- You will now program the FPGA using this file.
- Click on “Configure Target Device” to open the Impact program.

Connect the board



ISE Impact



Scan Chain Initialization

ISE iMPACT (P.49d) - [Boundary Scan]

File Edit View Operations Output Debug Window Help

iMPACT Flows

- Boundary Scan
 - SystemACE
 - Create PROM File (PROM File ...)
 - WebTalk Data

iMPACT Processes

Available Operations are:

Right click device to select operations

TDI

TDO

xc6slx16
nexys3.bit

SPI/BPI?

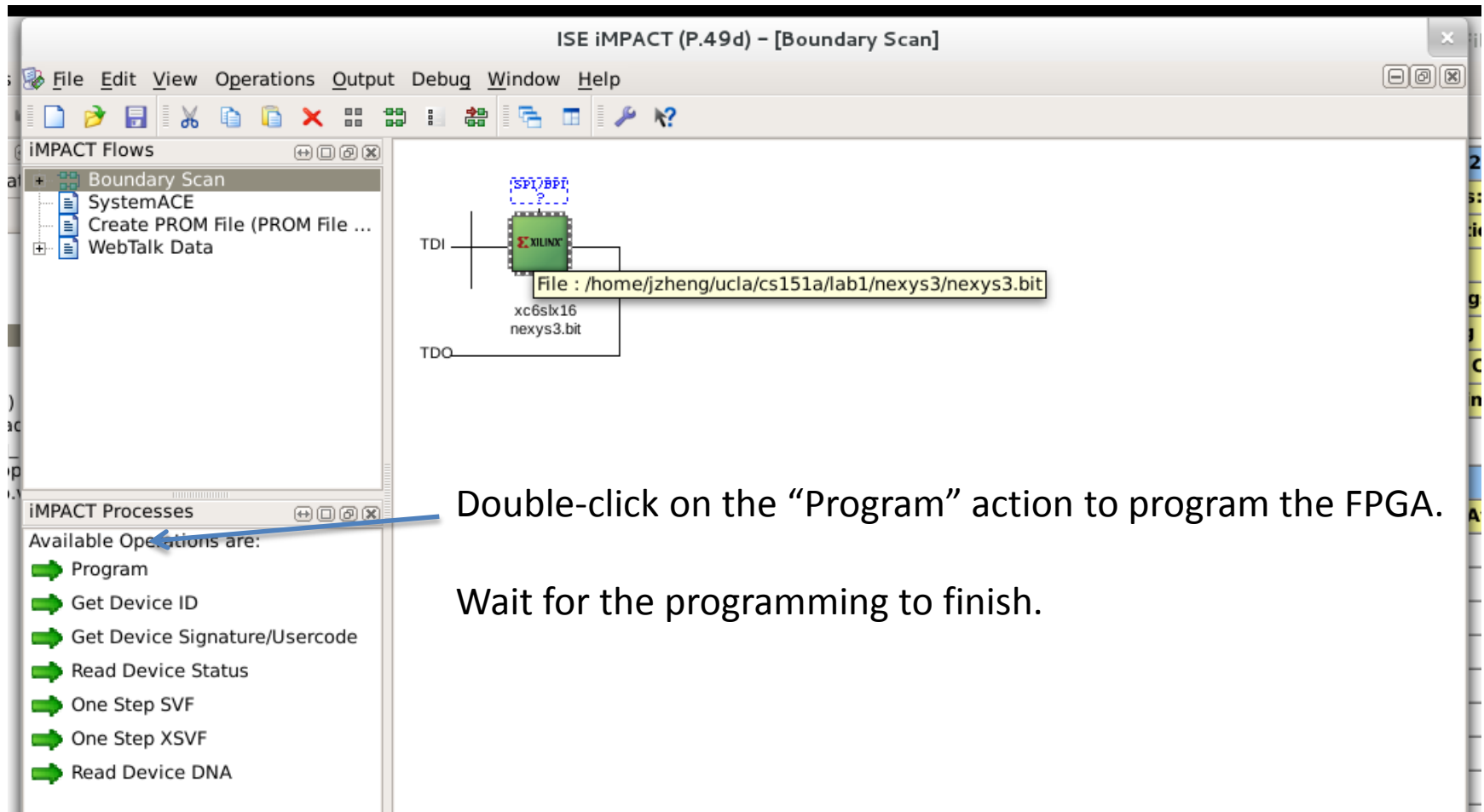
A properly initialized chain should contain a single FPGA (xc6slx16).

Assign the bit file to this FPGA

Do not assign any SPI flash programming files.

Click on the FPGA symbol to show the available options.

Program FPGA



Play Time

- Did you see the rightmost LED light up?
 - If yes, the board is programmed!
- Can you use the switches to control the LED?
 - Study code to understand how this is done