PART I: Theory

WHAT IS AN FPGA?

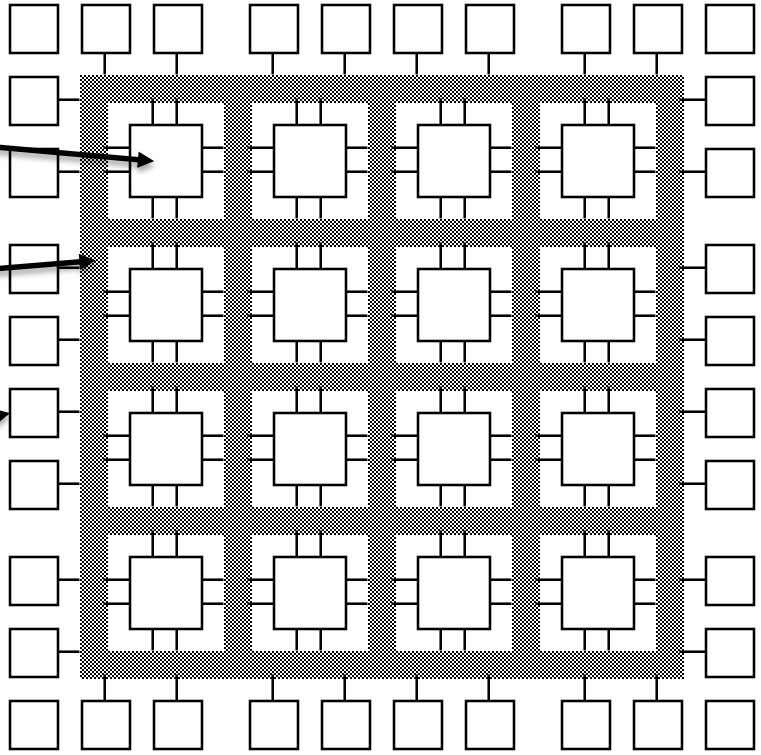
Field-Programmable Gate Arrays

- 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

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
- …

http://www.xilinx.com/training/fpga/fpga-field-programmable-gate-array.htm

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.

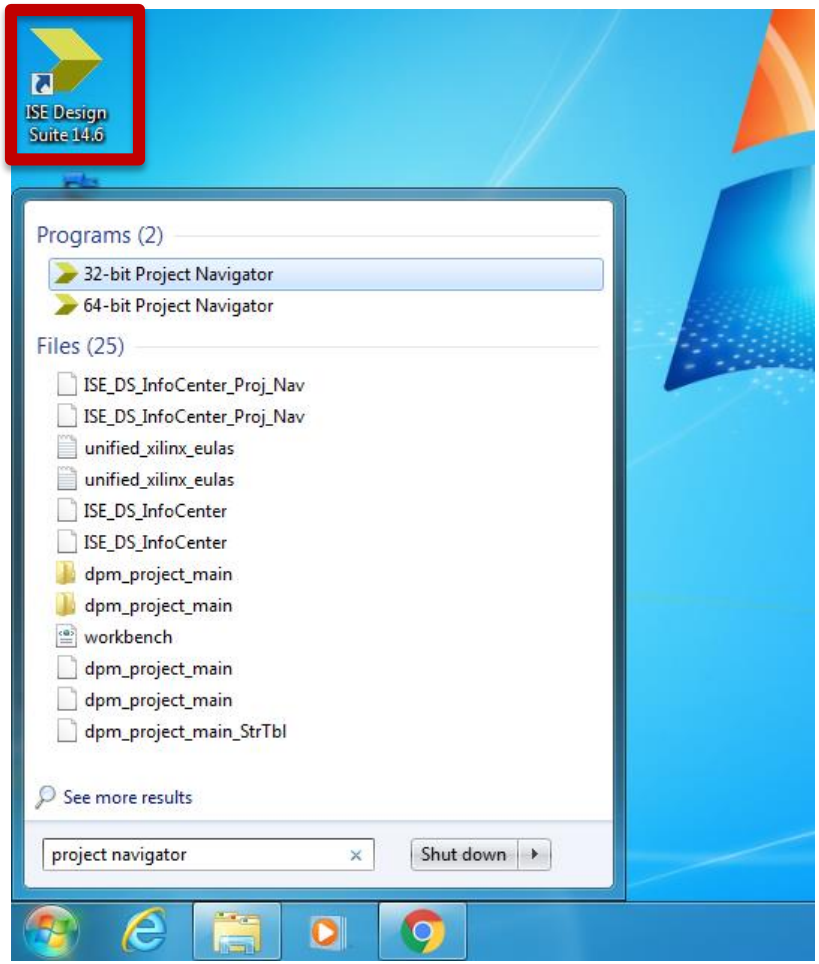
Tools of Trade

- 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

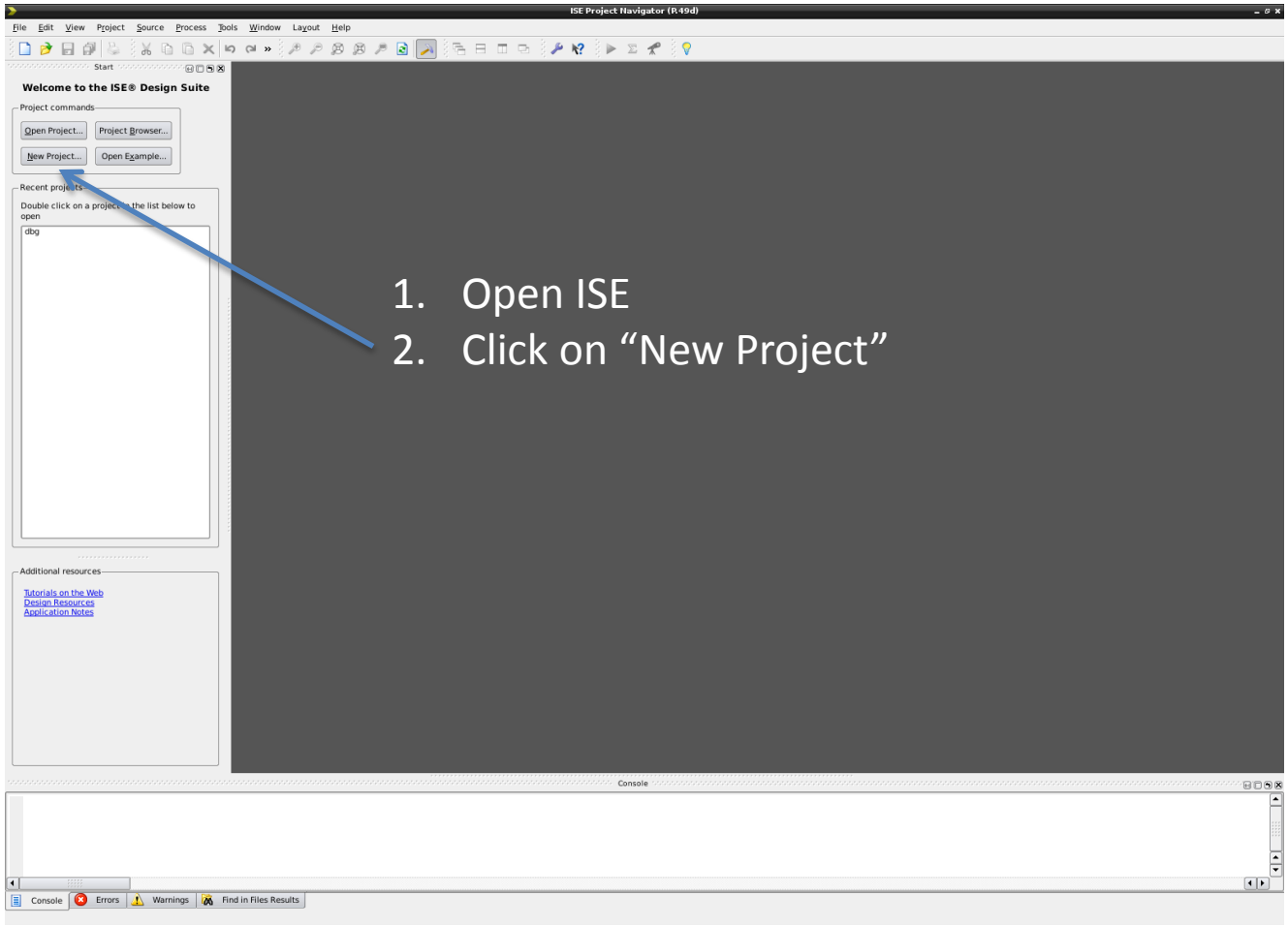
EXAMPLE PROJECT IMPLEMENTATION

Open the Xilinx ISE

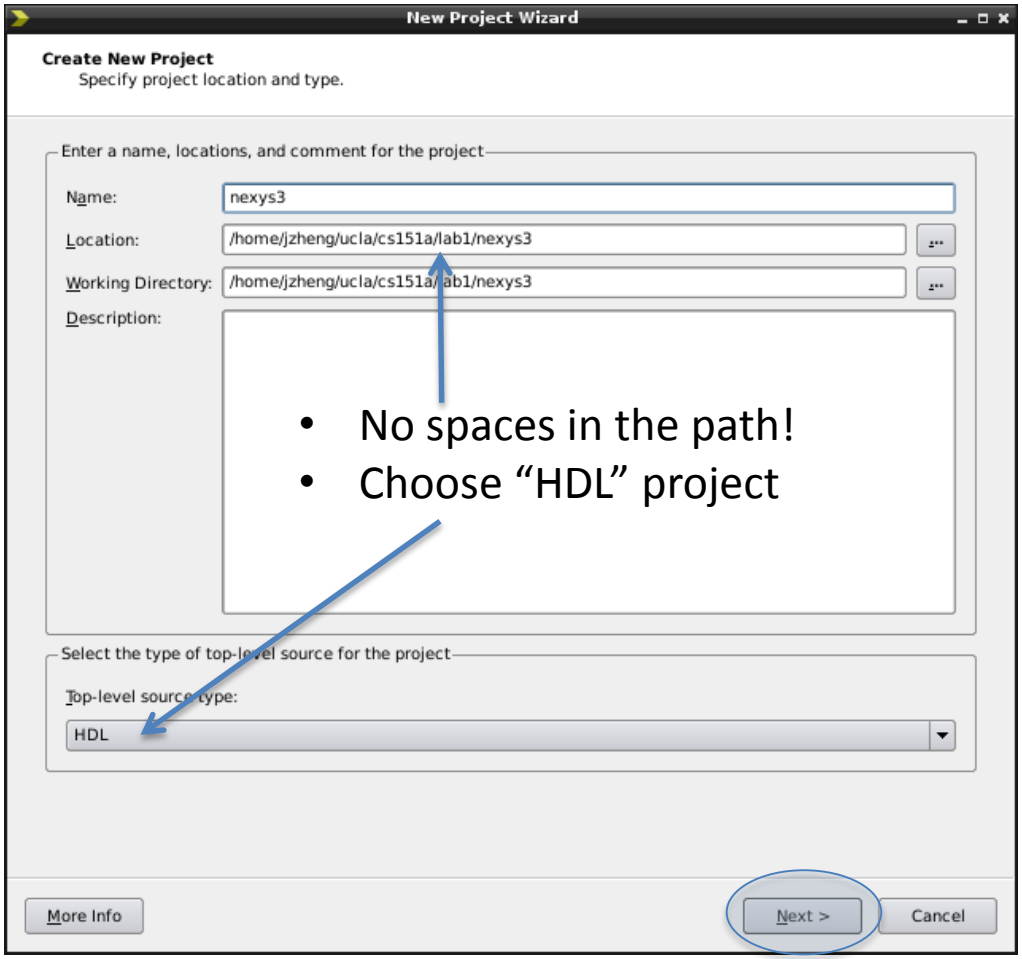


- Click the ISE Design Suite 14.6 icon
- Or search for “project navigator” in the start menu

Create an ISE Project



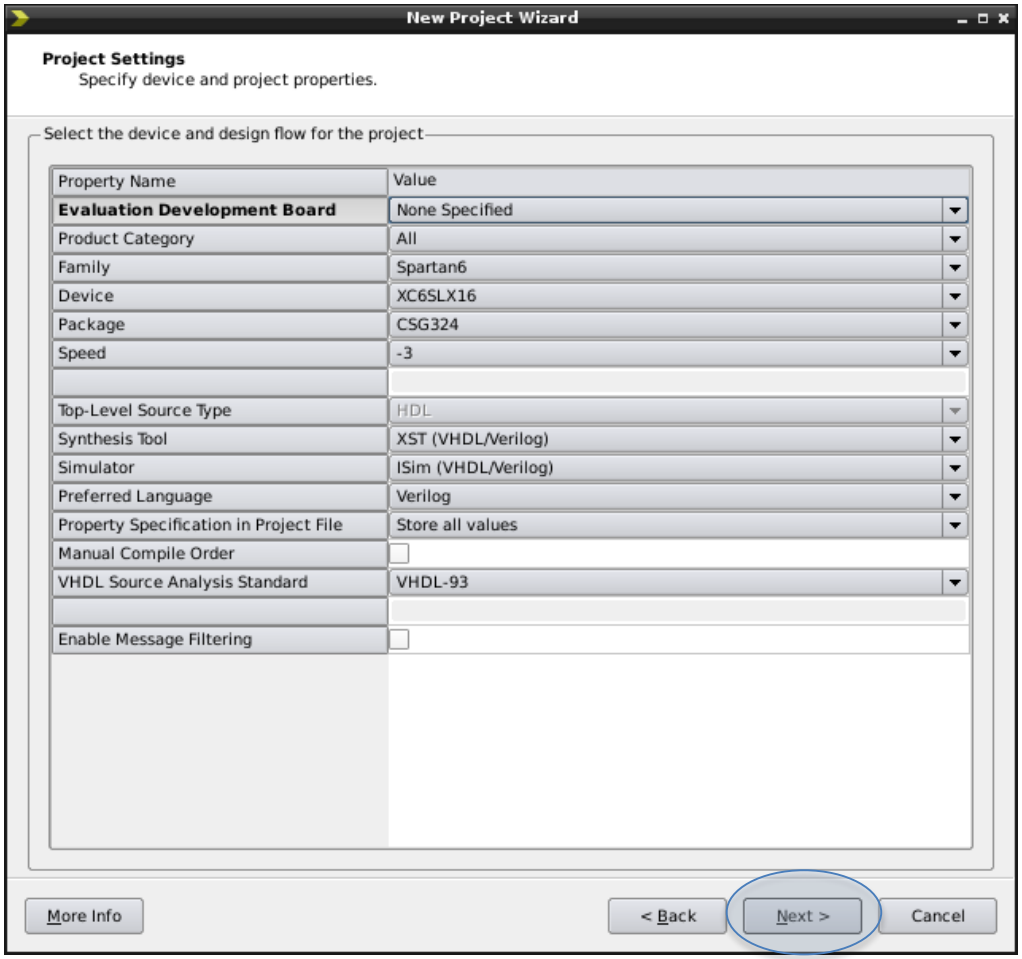
New Project Dialogue



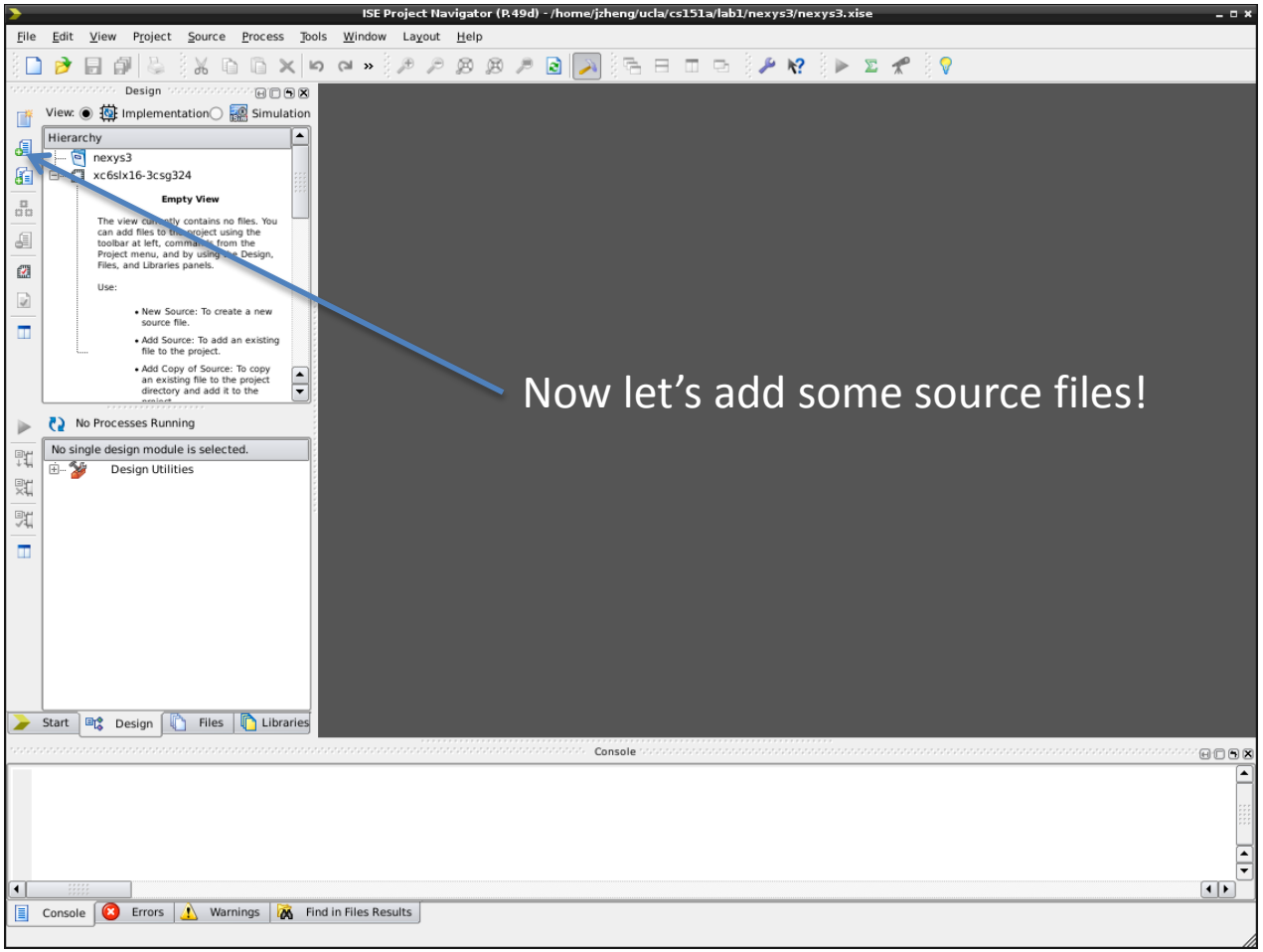
- No spaces in the path!
- Choose “HDL” project

Device Properties

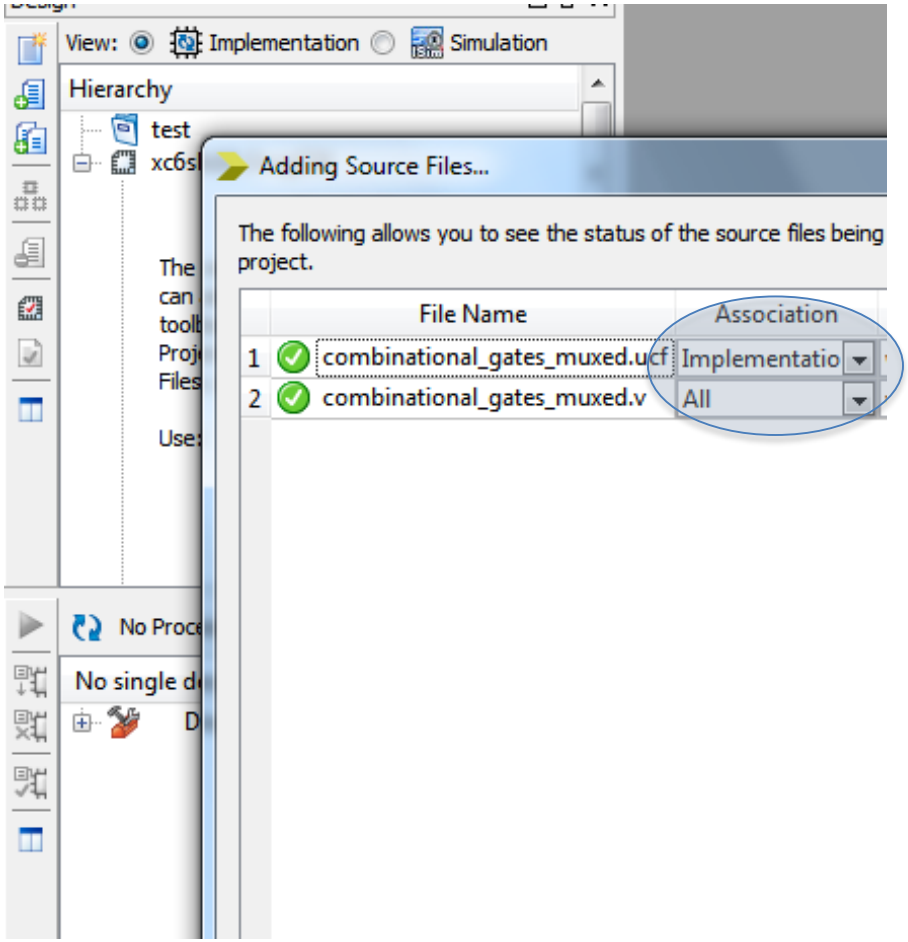
Make sure the fields match what you see here



Project Created

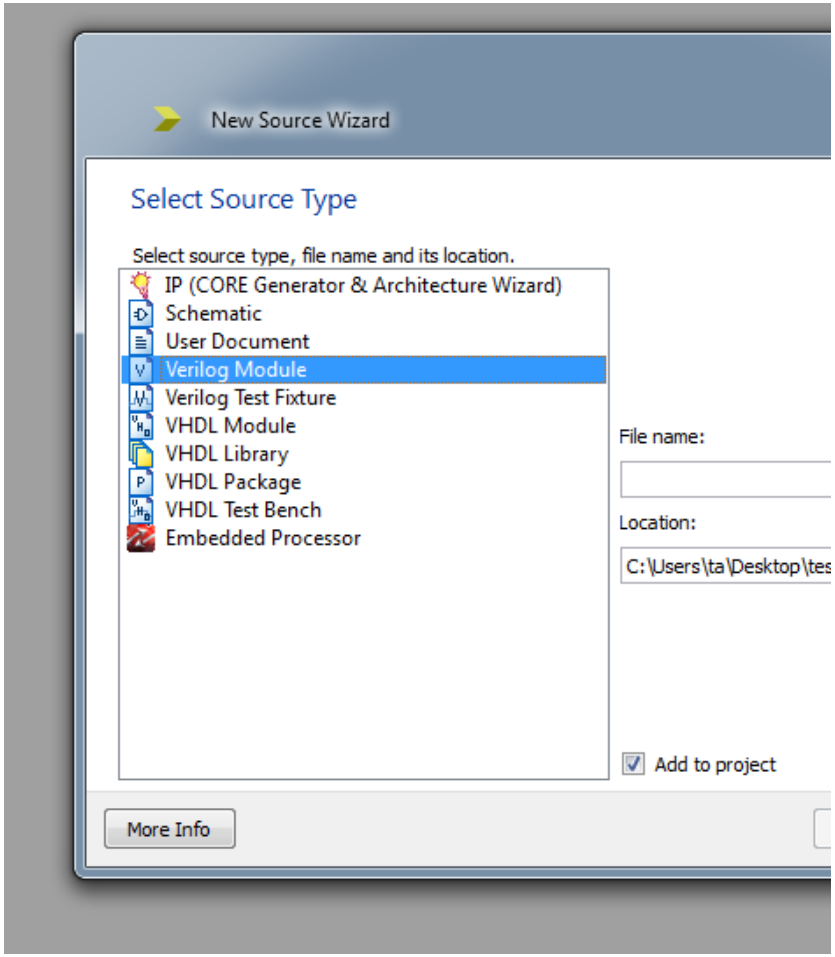


Add Source Files



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

New Source Files



- Alternatively if you want to write your own code, click on “New Source”
- Select “Verilog Module” then follow the instructions

Source Files

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

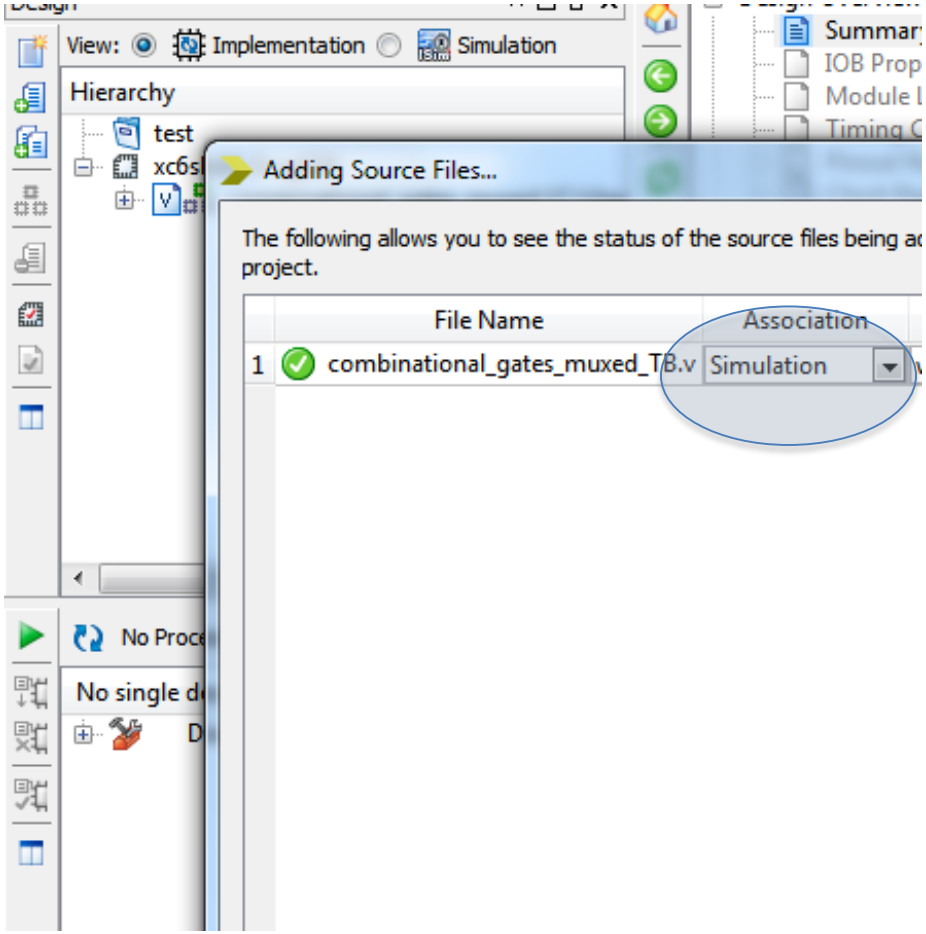
UCF lists all the available pin mappings in the FPGA in the following format:

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

For example:

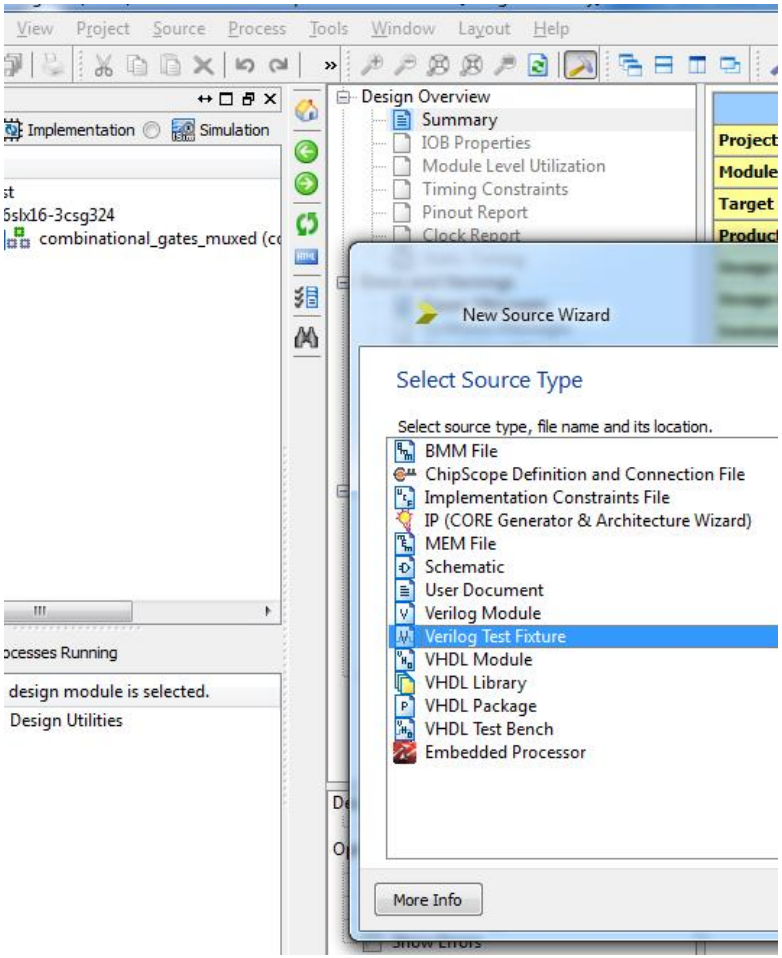
```
Net "sw<0>" LOC = T10 | IOSTANDARD = LVCMOS33; #Bank = 2,
        pin name = IO_L29N_GCLK2, Sch name = SW0
```

Add Testbench Code



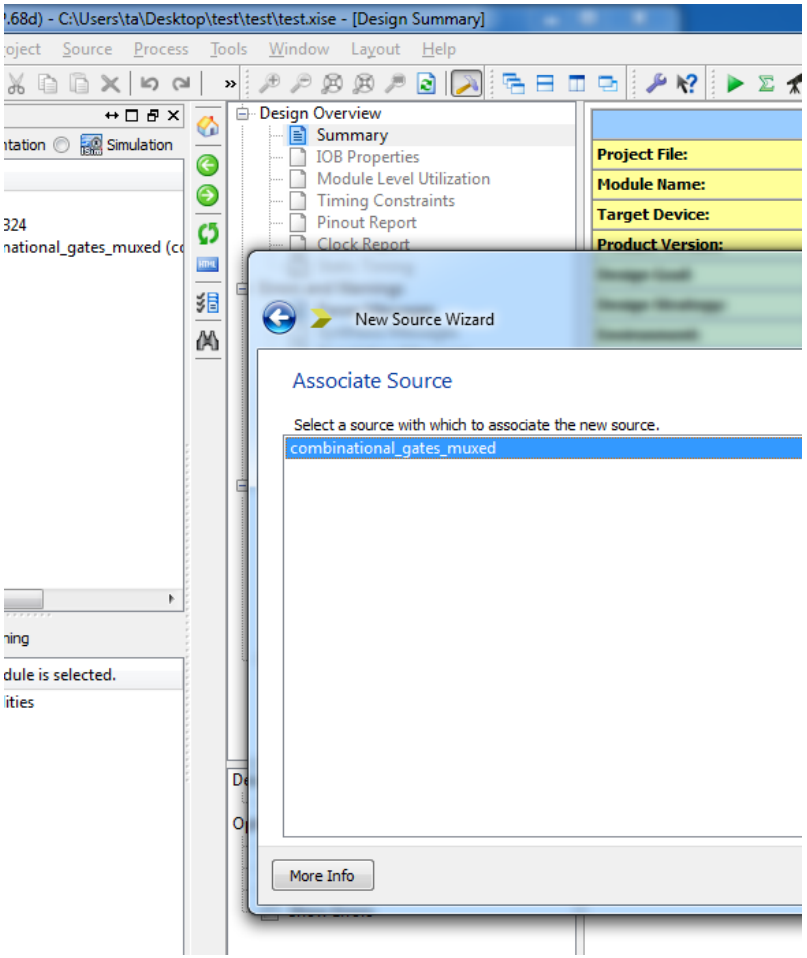
1. Click on “Add sources” again
2. Select the combinational_gates_muxed_TB.v
3. Make sure the file association is correct

Create Testbench Code



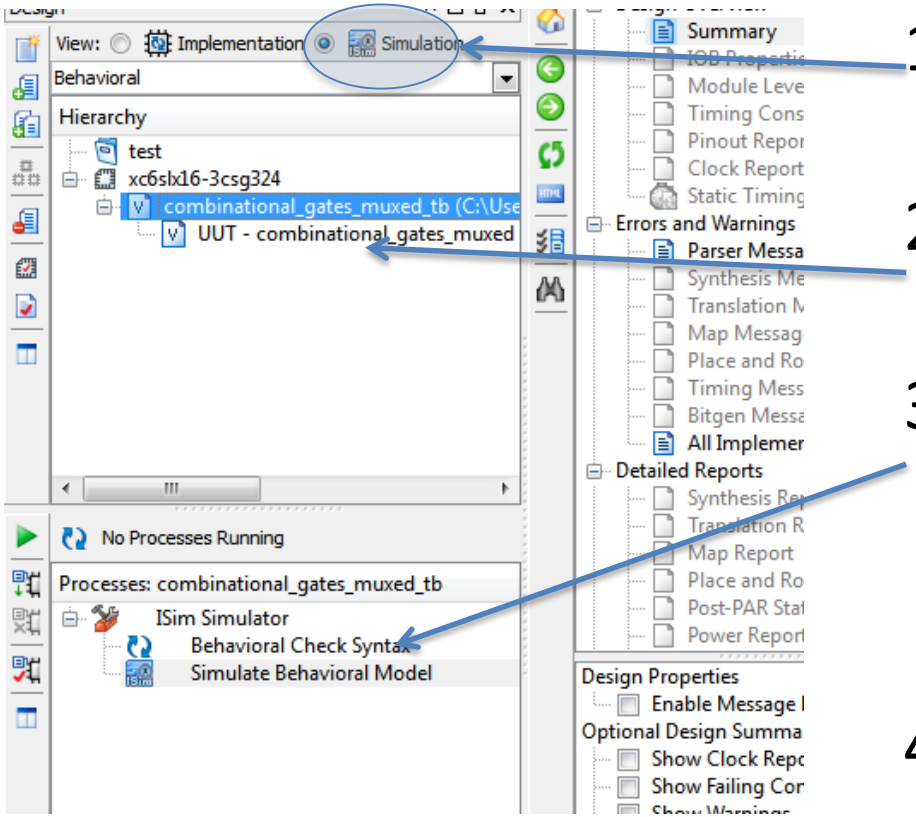
- Alternatively if you want to write your own testbench, click on "New Source"
- Select "Verilog Test Fixture"

Create Testbench Code



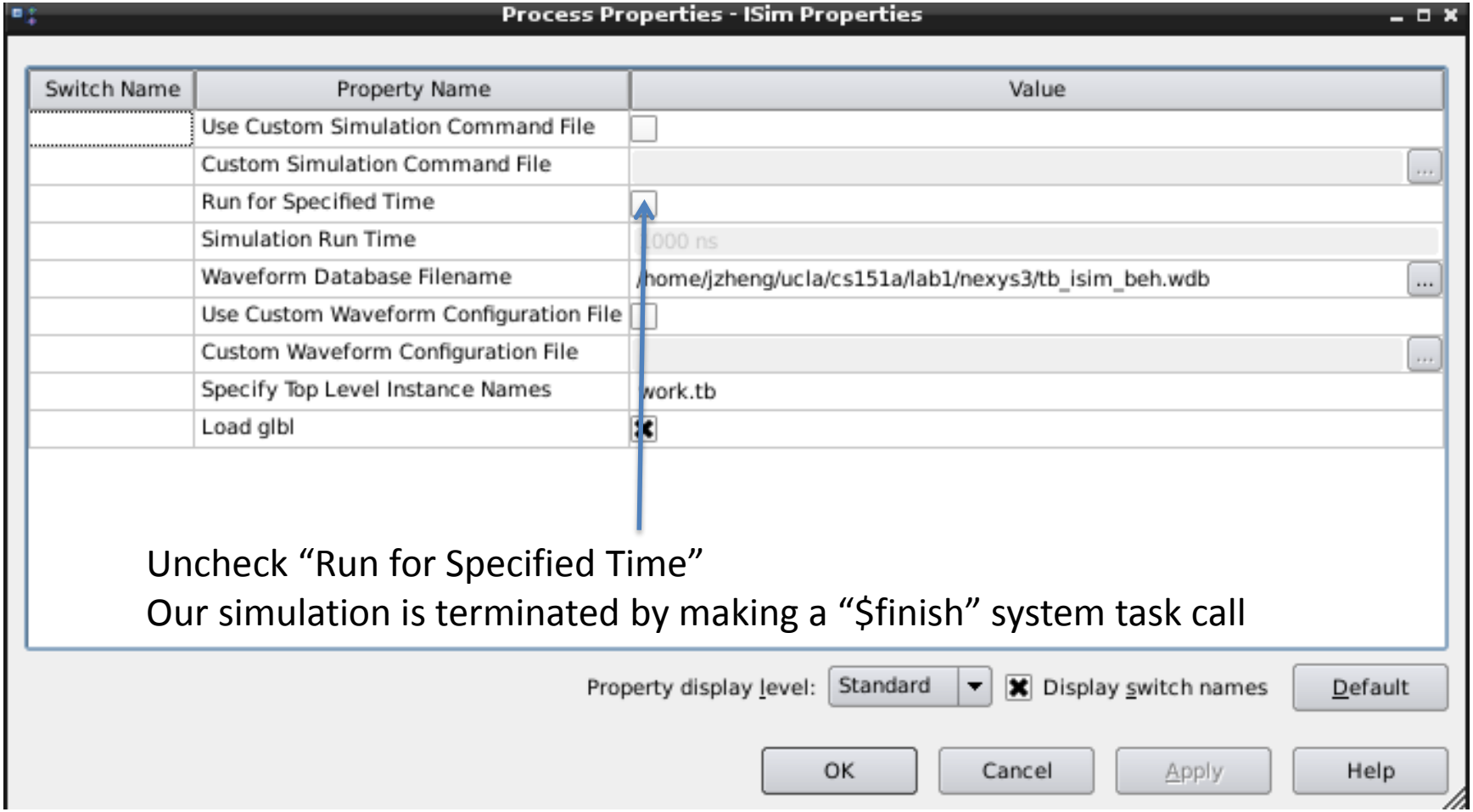
- Select the module to test and then follow the instructions

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



Uncheck “Run for Specified Time”

Our simulation is terminated by making a “$finish” system task call

Launch ISIM

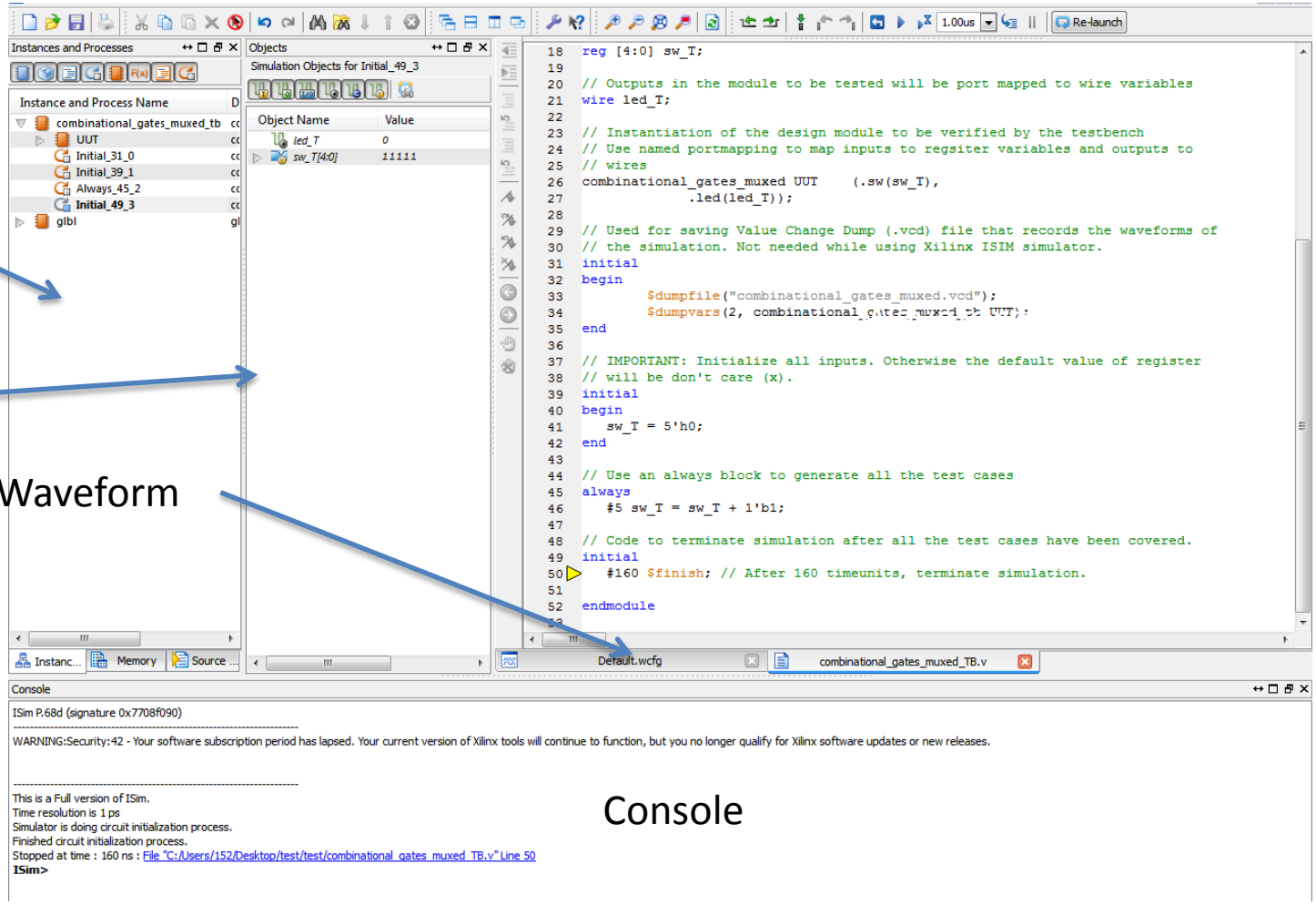
- Right click on “Simulate Behavioral Model” again, this time choose “run all”
- ISIM will be launched
- ISIM is the simulation environment where you can dynamically debug the circuit, much like a software debugger
- Your main focus should be on the console window and the waveform window

ISIM Main Window

Hierarchical View

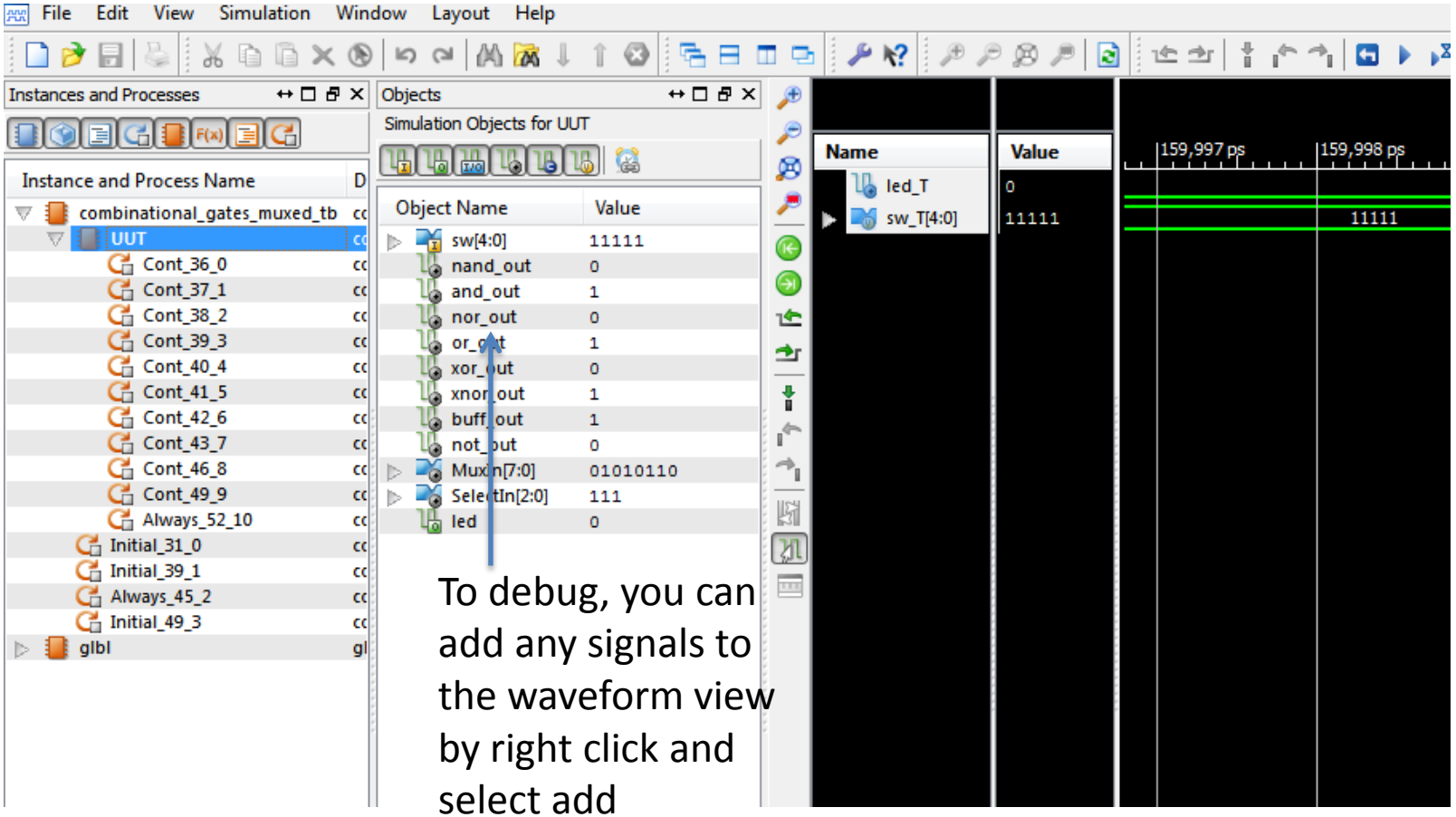
Signal View

Waveform

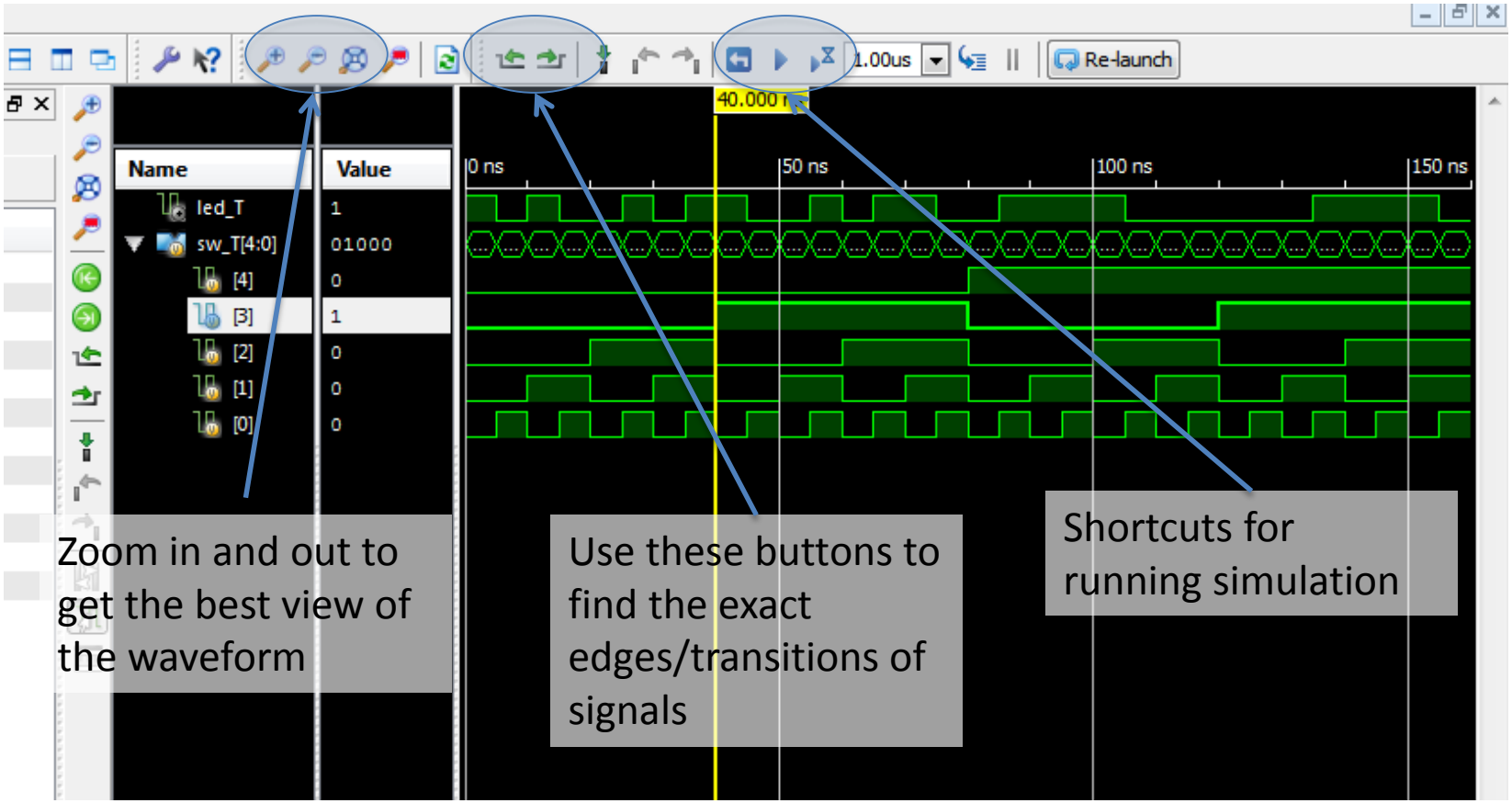


Console

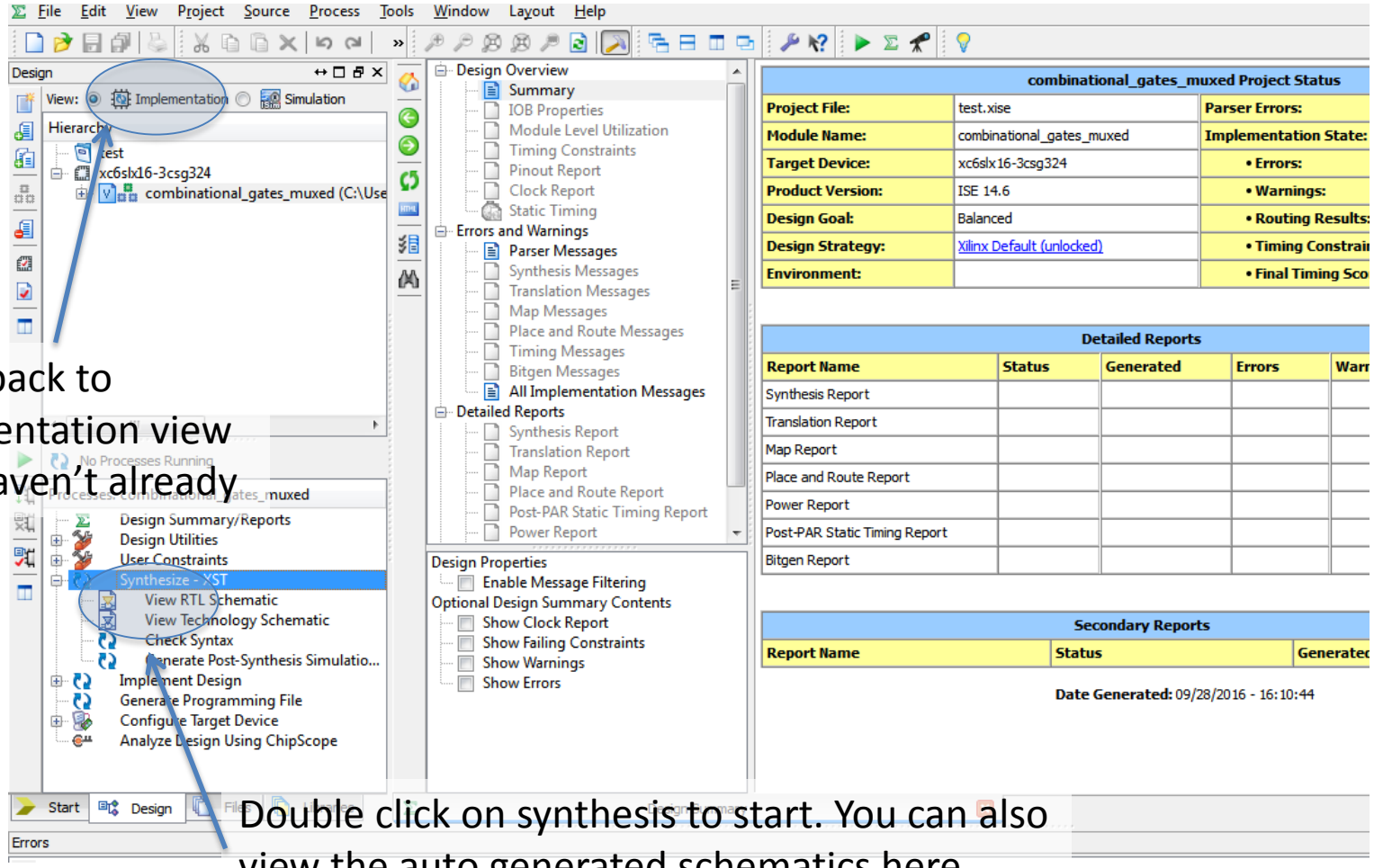
Post Simulation Examination



Post Simulation Examination



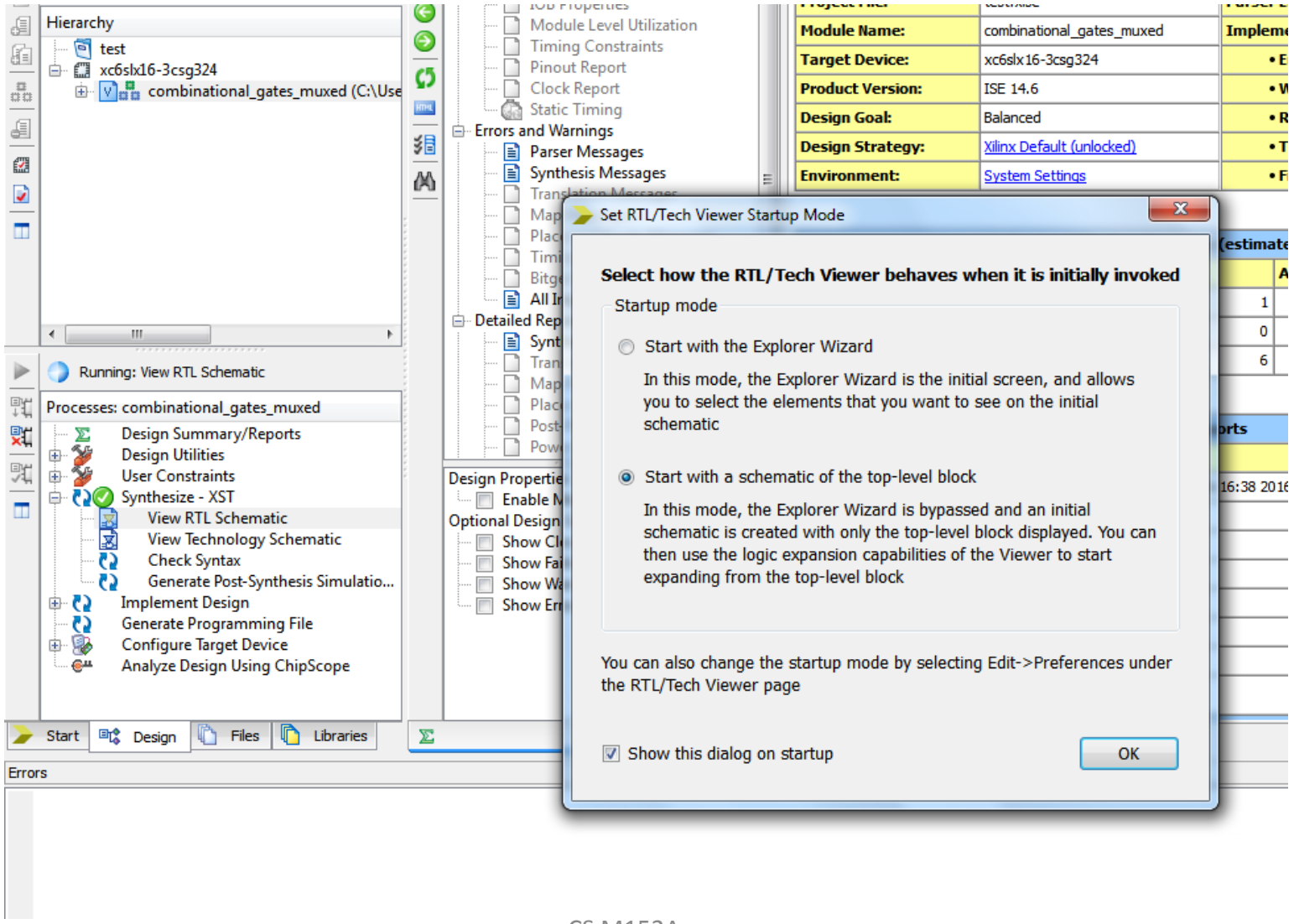
Ready for Synthesis



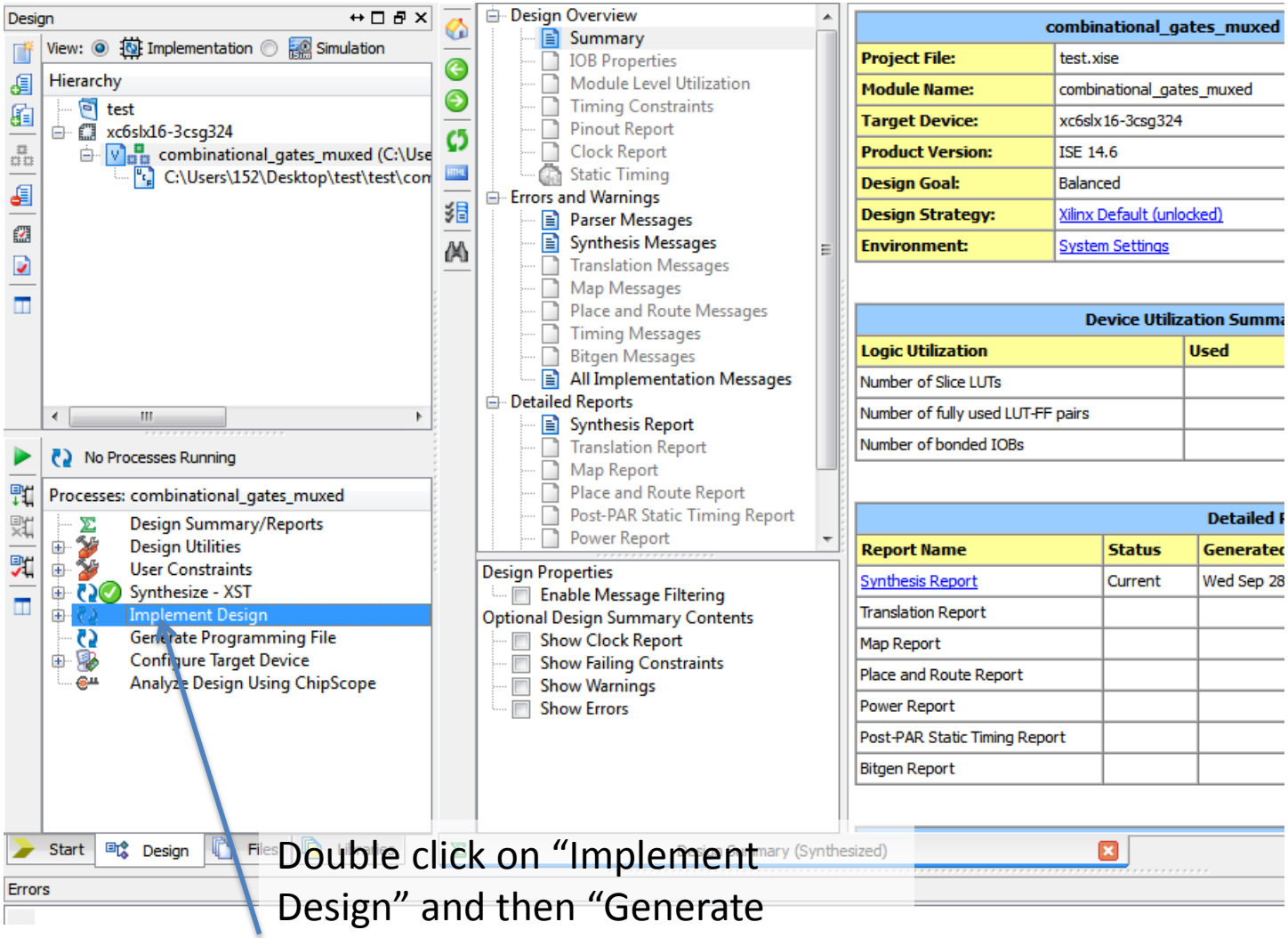
Switch back to implementation view if you haven't already

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

View Schematics



Place-n-Route and Bitstream

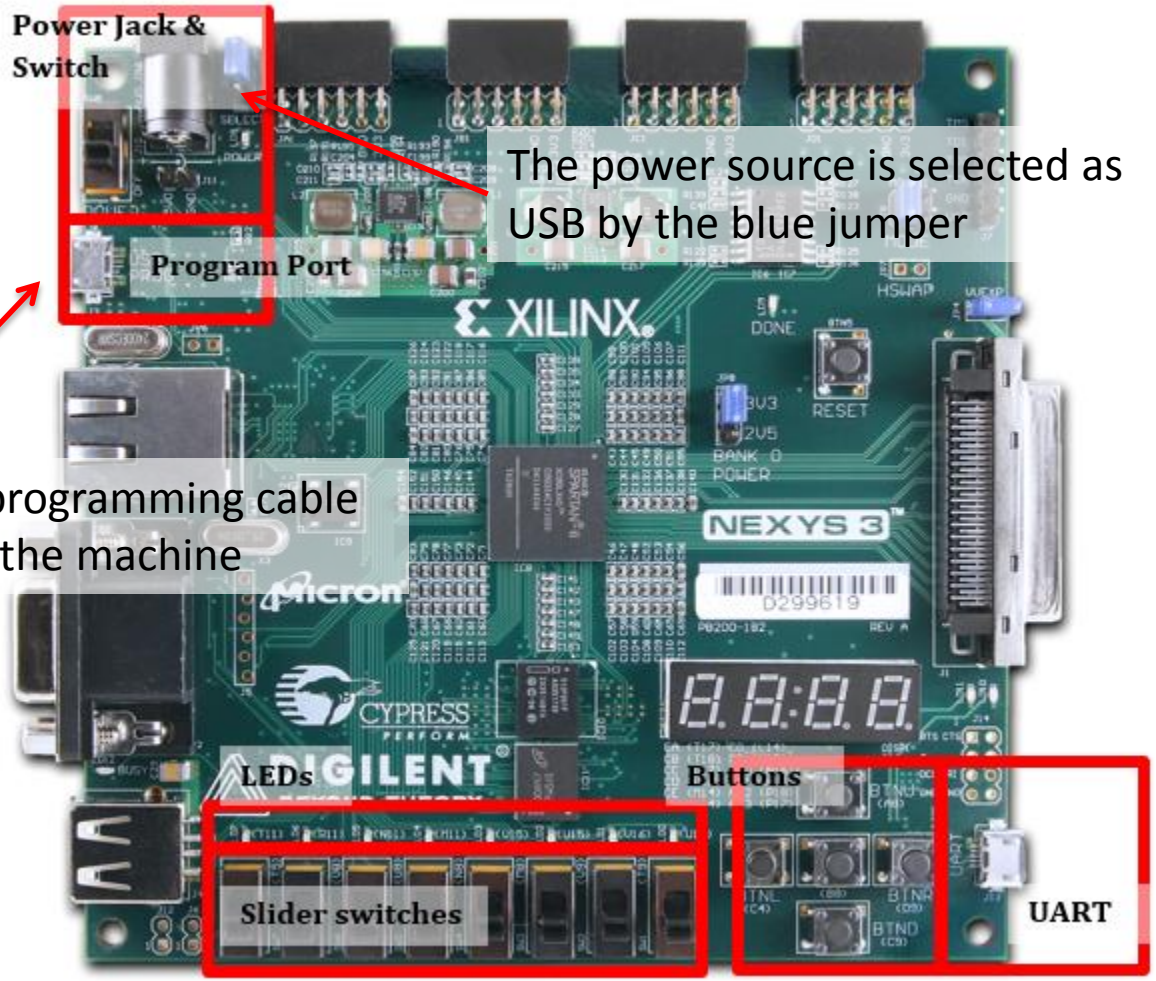


Double click on “Implement Design” and then “Generate Programming File”

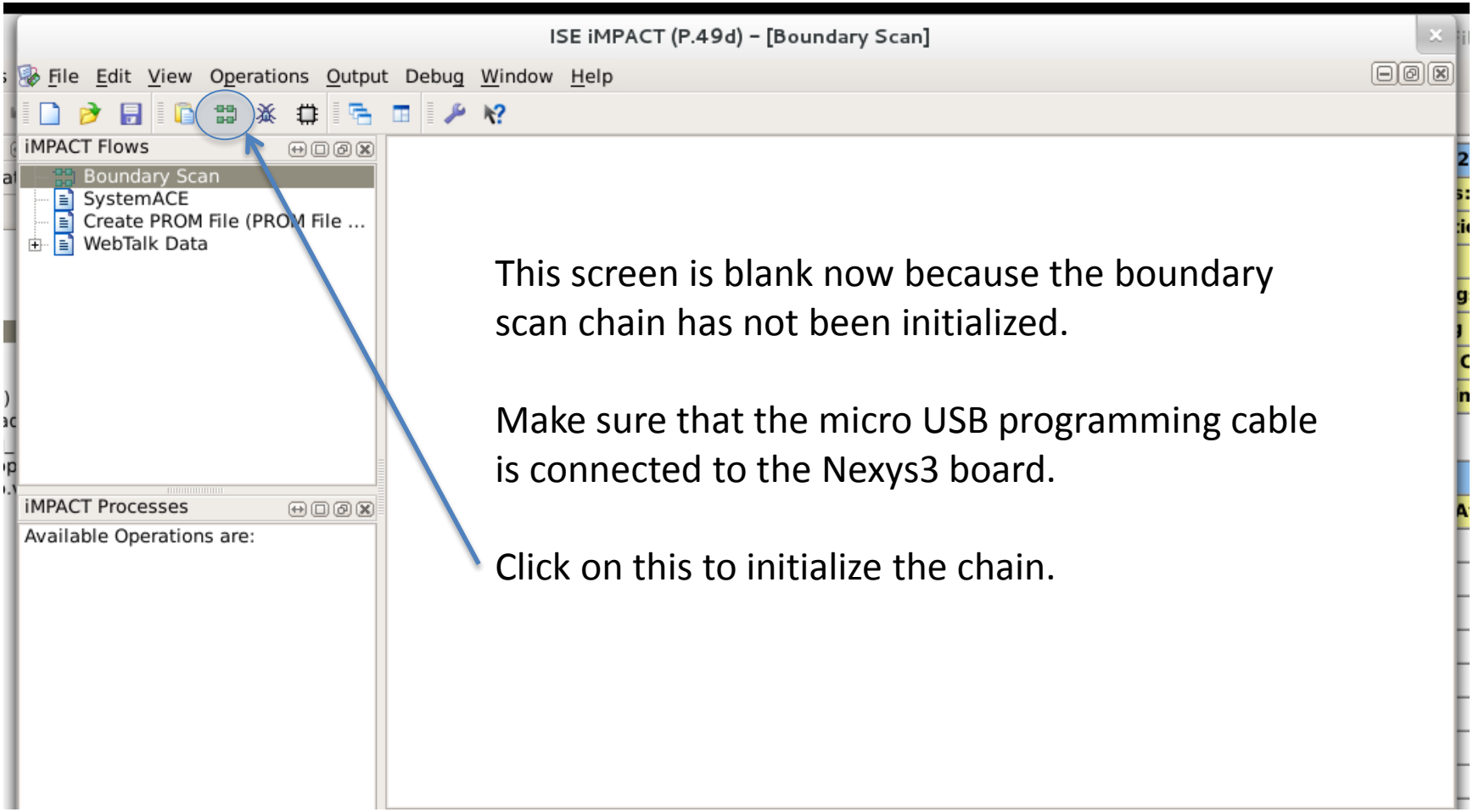
Download Bitstream to FPGA

- By now you should have a top_module_name.bit(combinational_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.

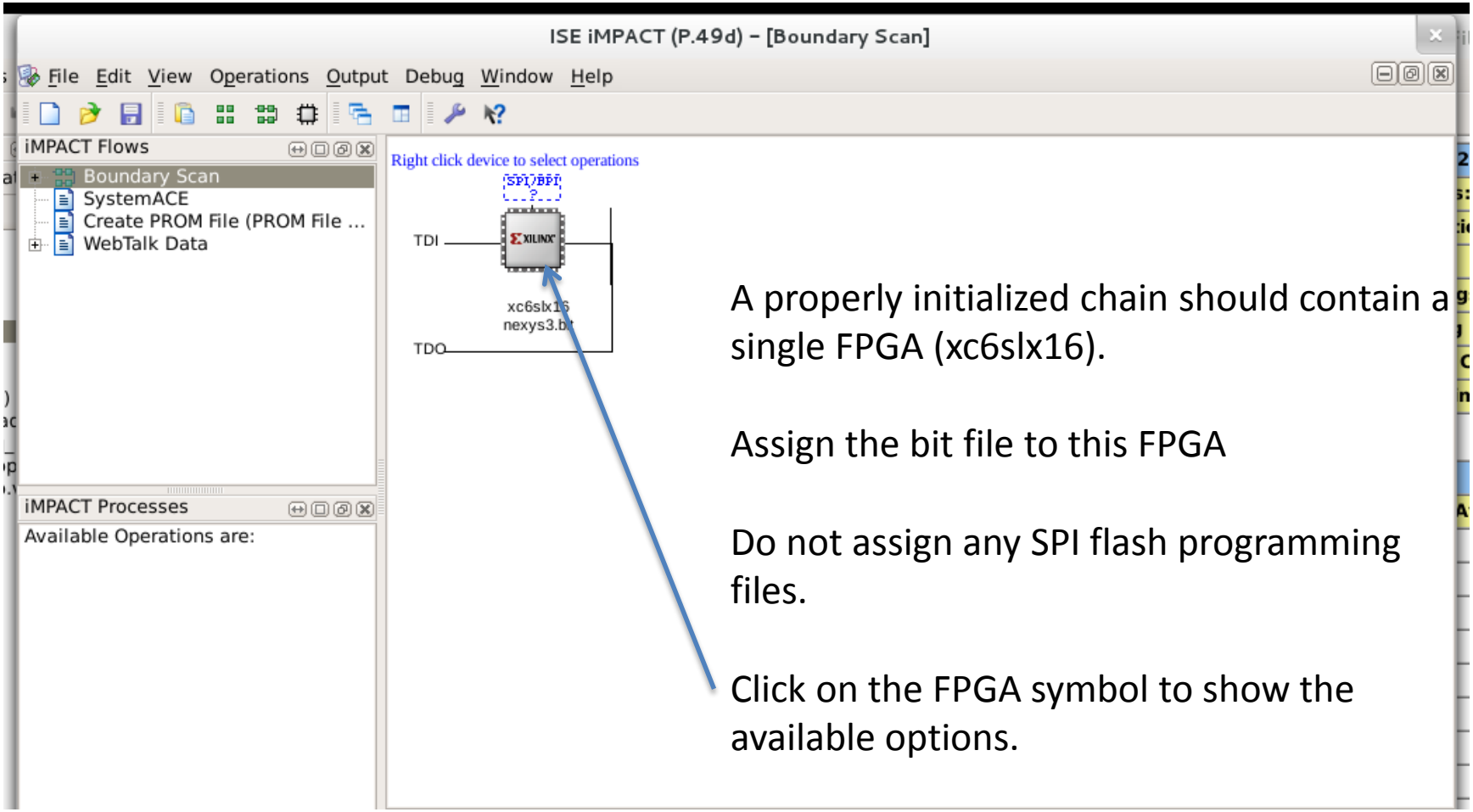
Connect the board



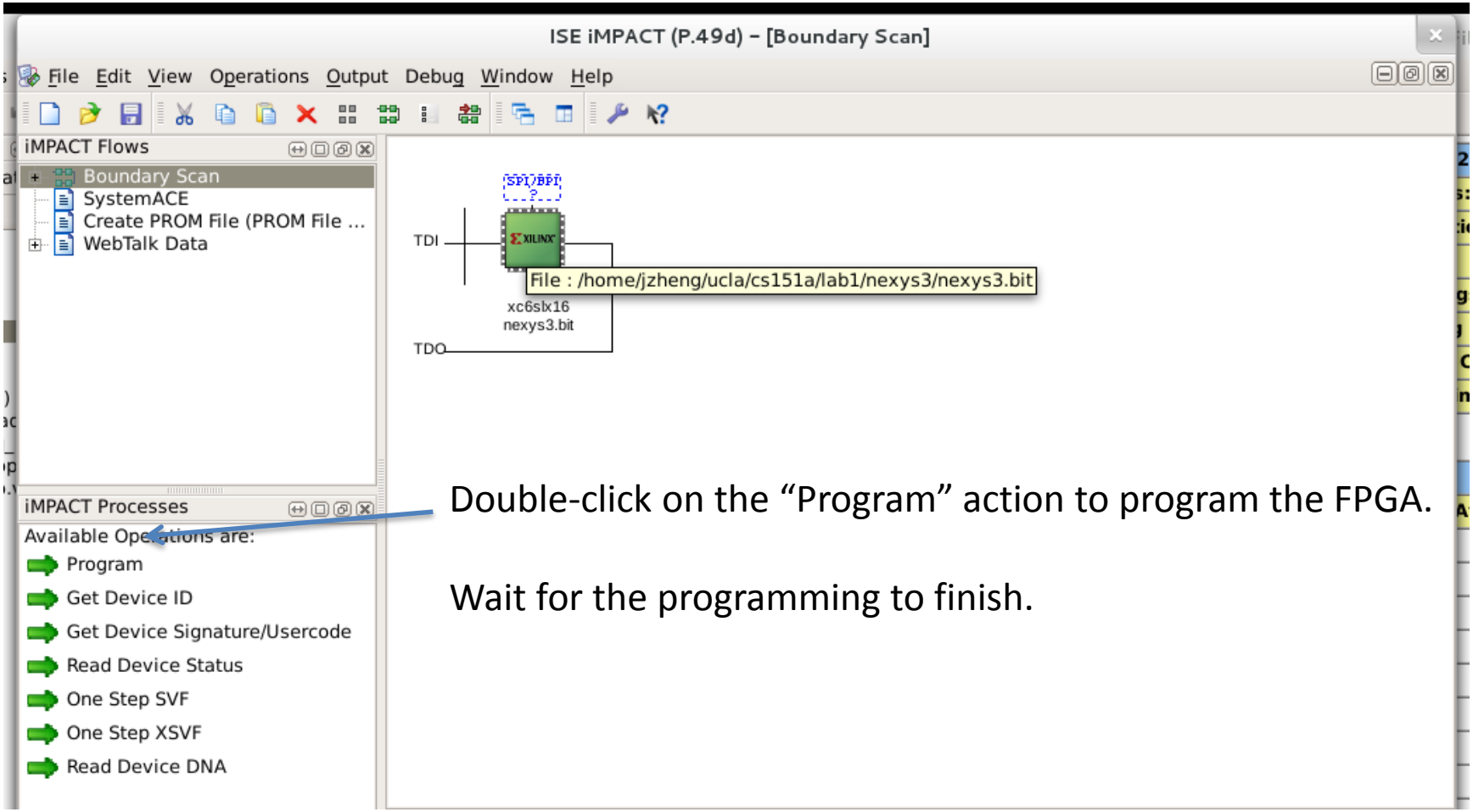
ISE Impact


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:
This screen is blank now because the boundary scan chain has not been initialized.
Make sure that the micro USB programming cable is connected to the Nexys3 board.
Click on this to initialize the chain.

Scan Chain Initialization



Program FPGA



Double-click on the “Program” action to program the FPGA.

Wait for the programming to finish.

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