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

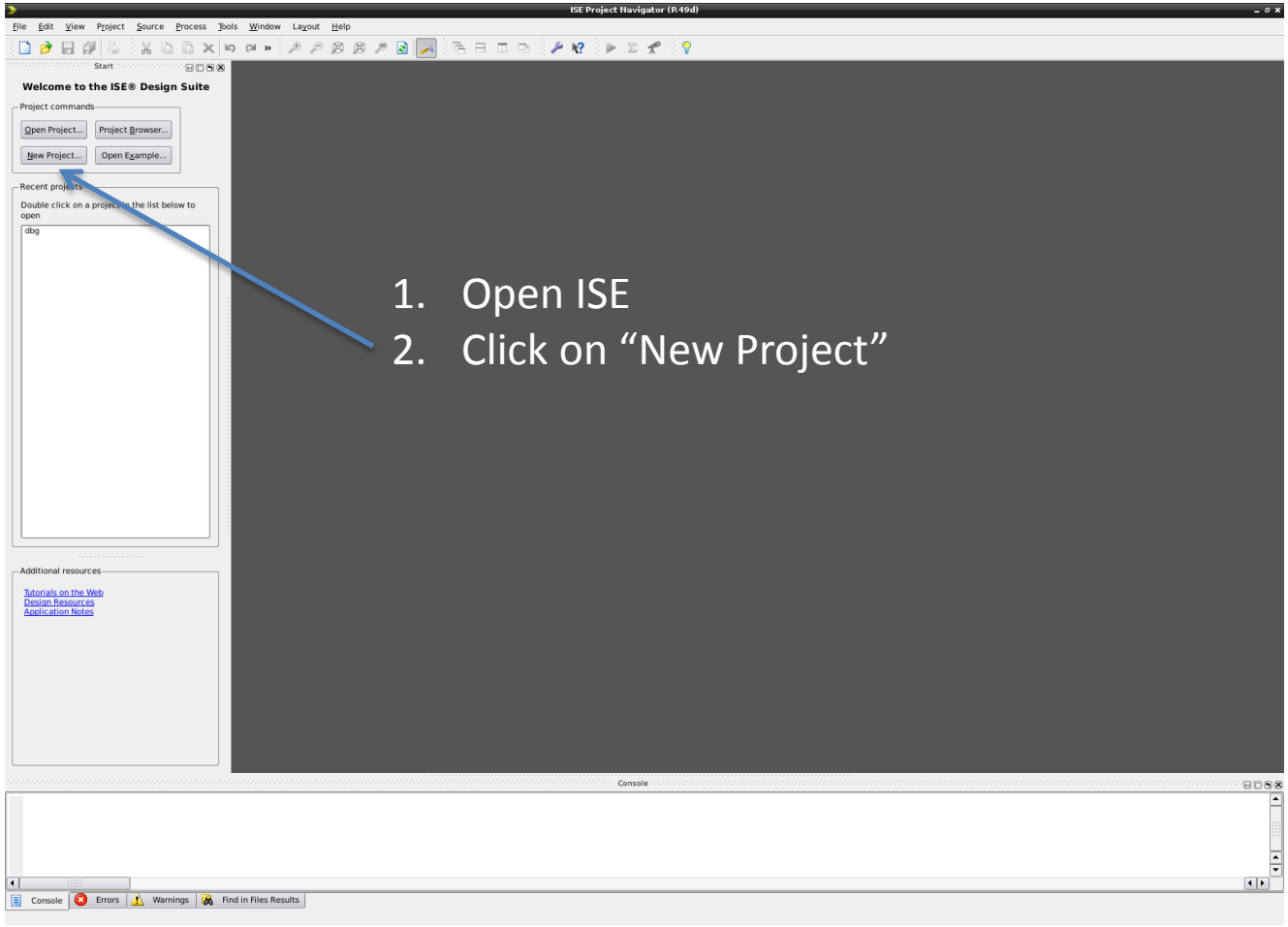
Now is the time to create the ISE project and go through steps 2-8

EXAMPLE PROJECT IMPLEMENTATION

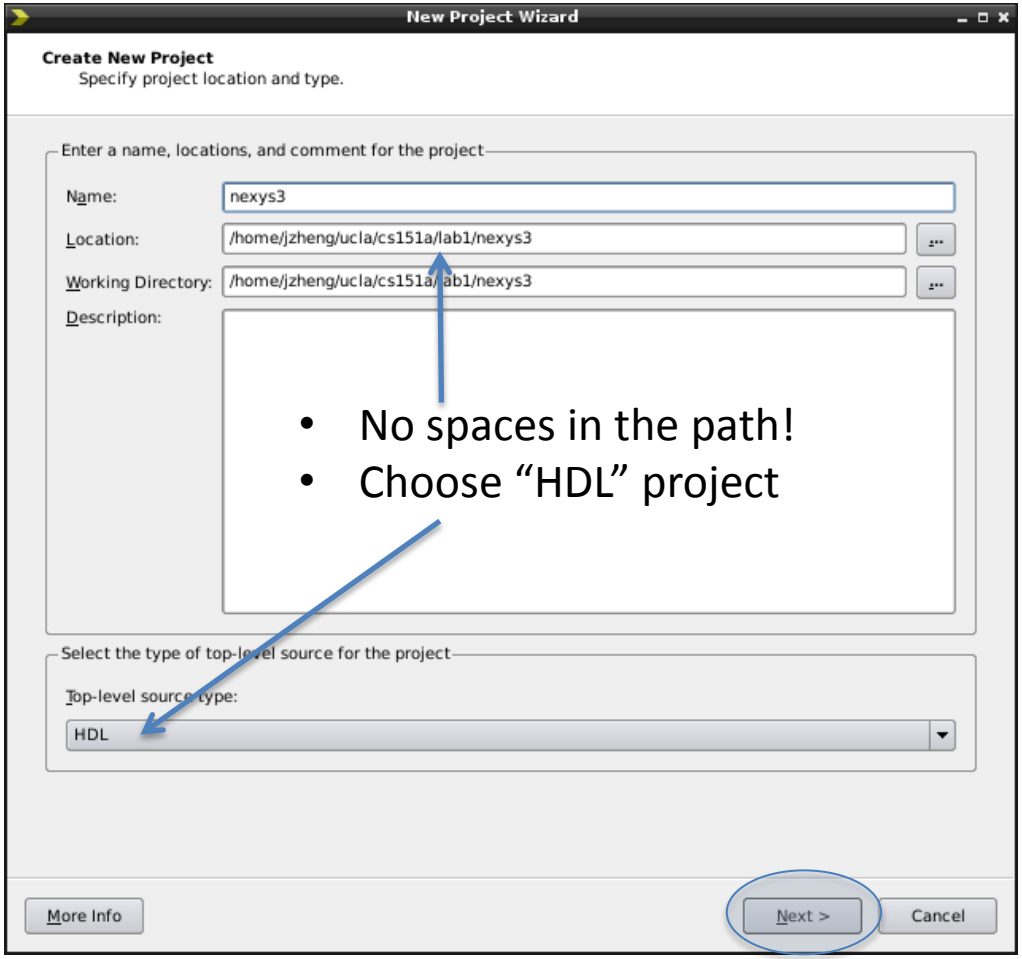
Implementation

- The previous section roughly describes the *Design* aspect of the sequencer project
- In this section, we will walk through the rest of steps (coding, simulation, etc.)
- Fortunately you are already provided with all the files ready for implementation
- Unzip the package that you are given and follow the steps in the following slides

Create an ISE Project



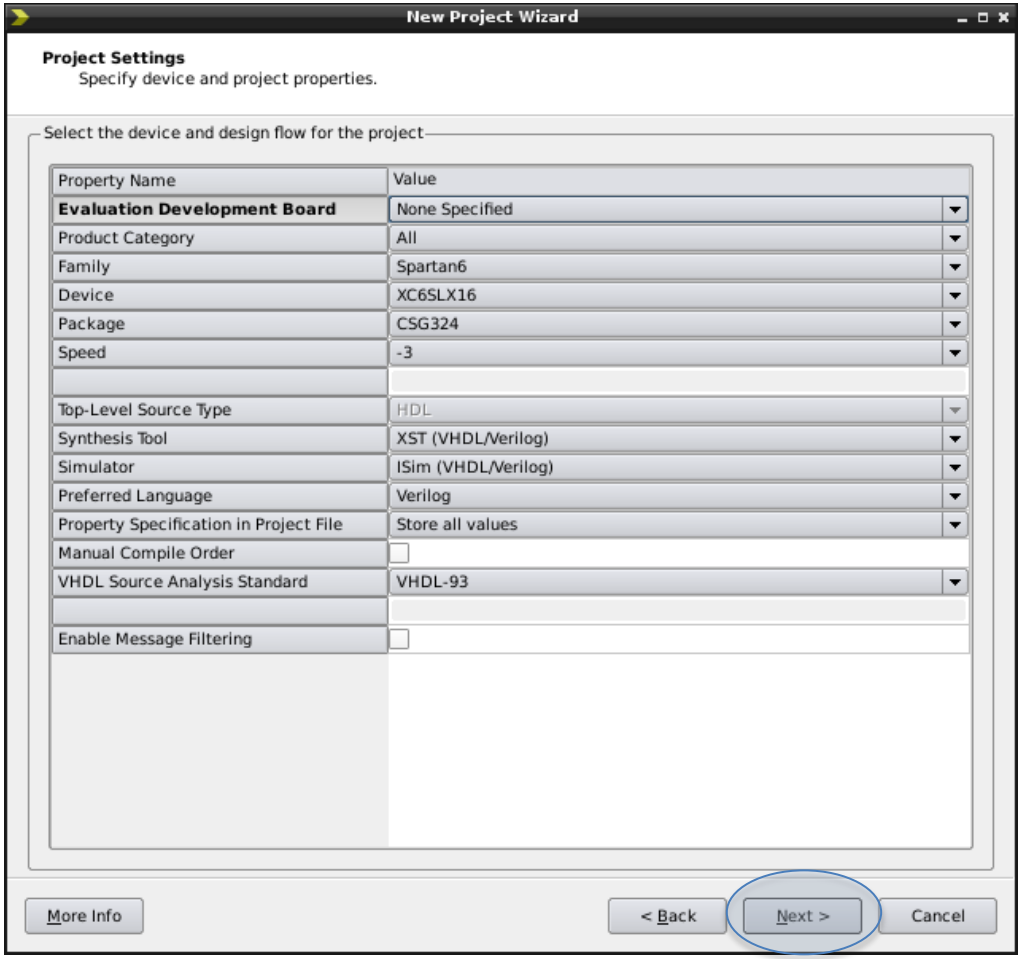
New Project Dialogue



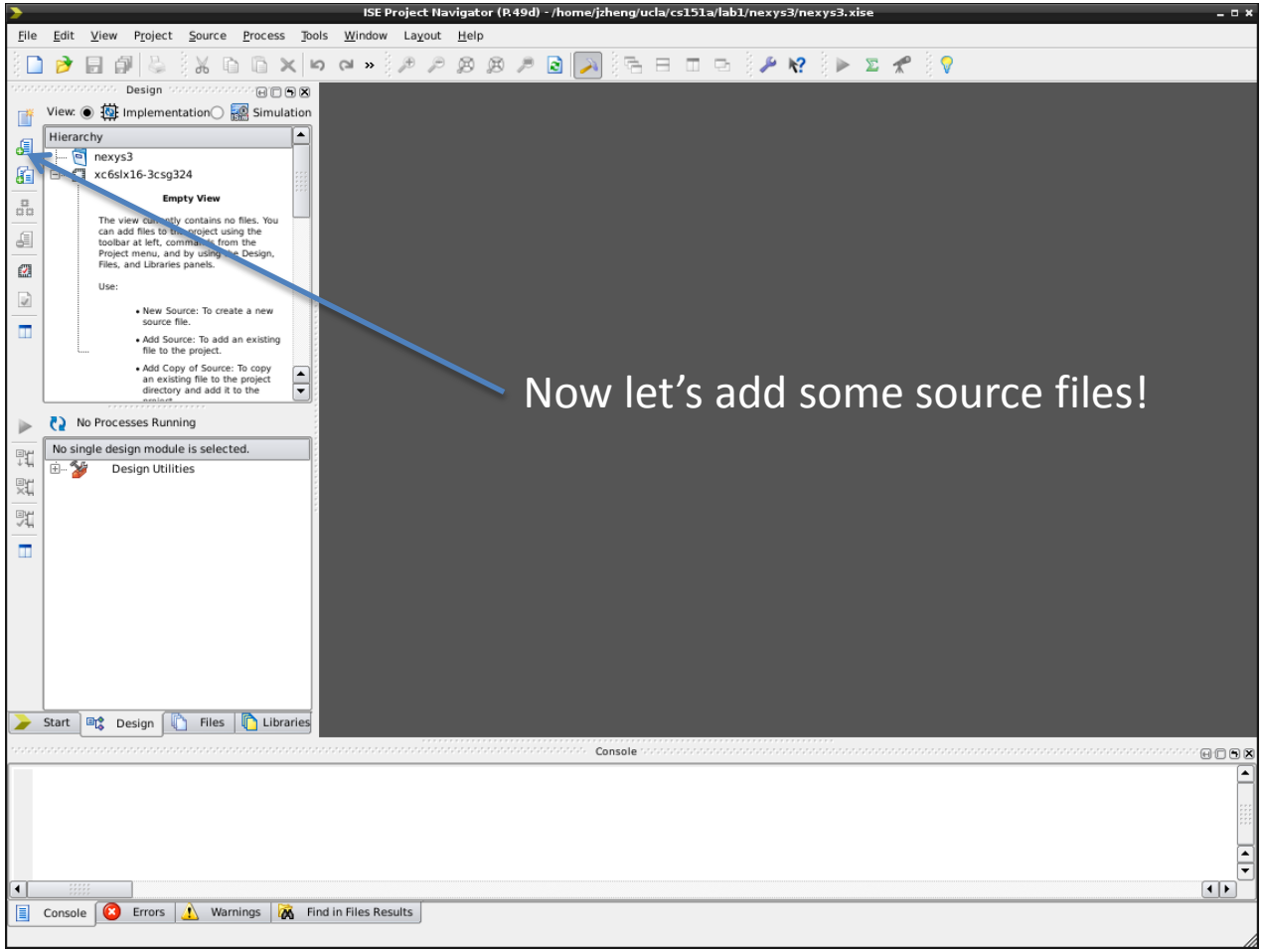
- No spaces in the path!
- Choose “HDL” project

Device Properties

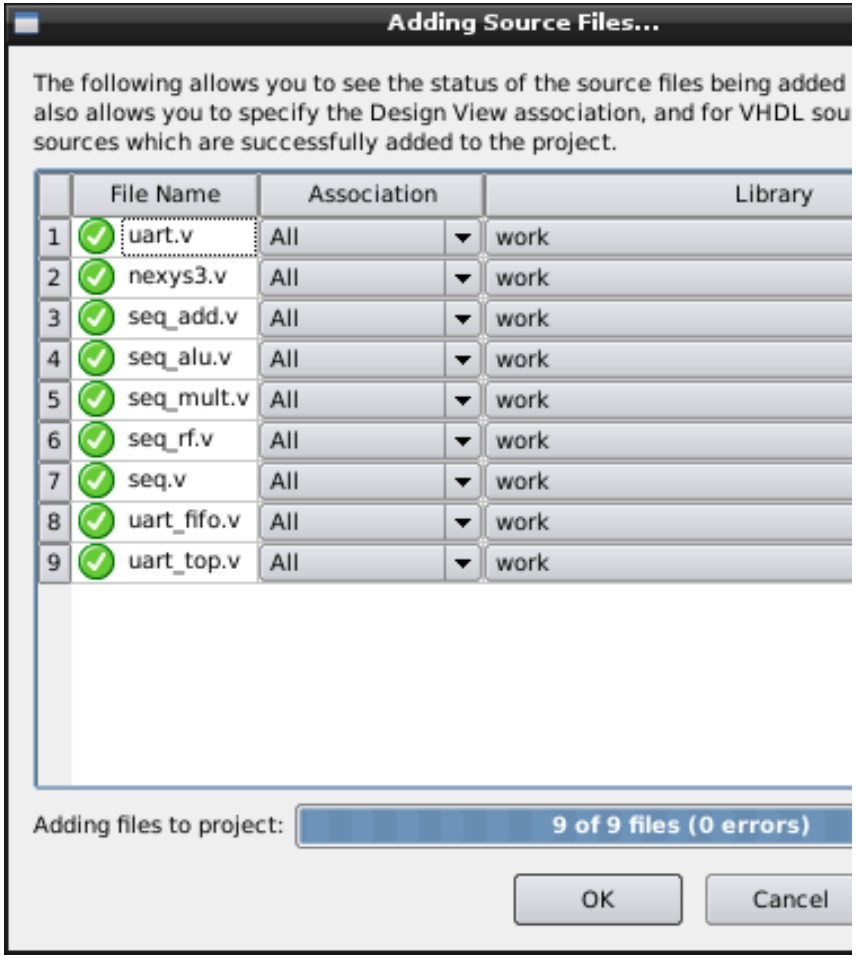
Make sure the fields match what you see here



Project Created

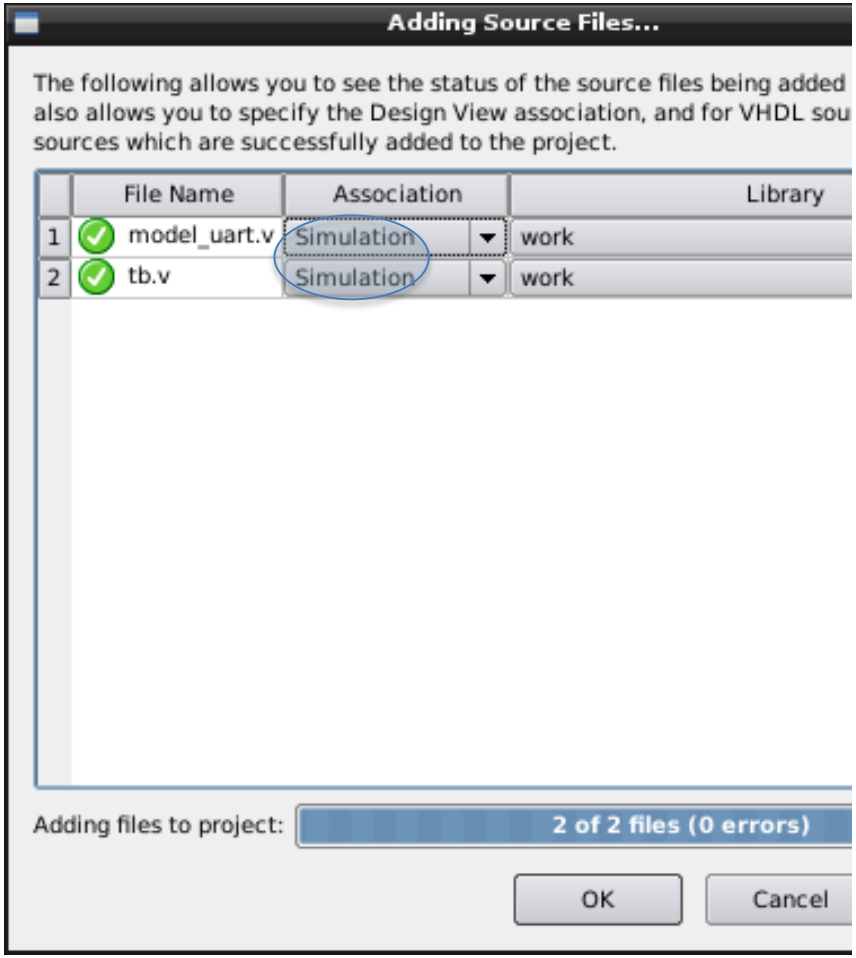


Add RTL Sources



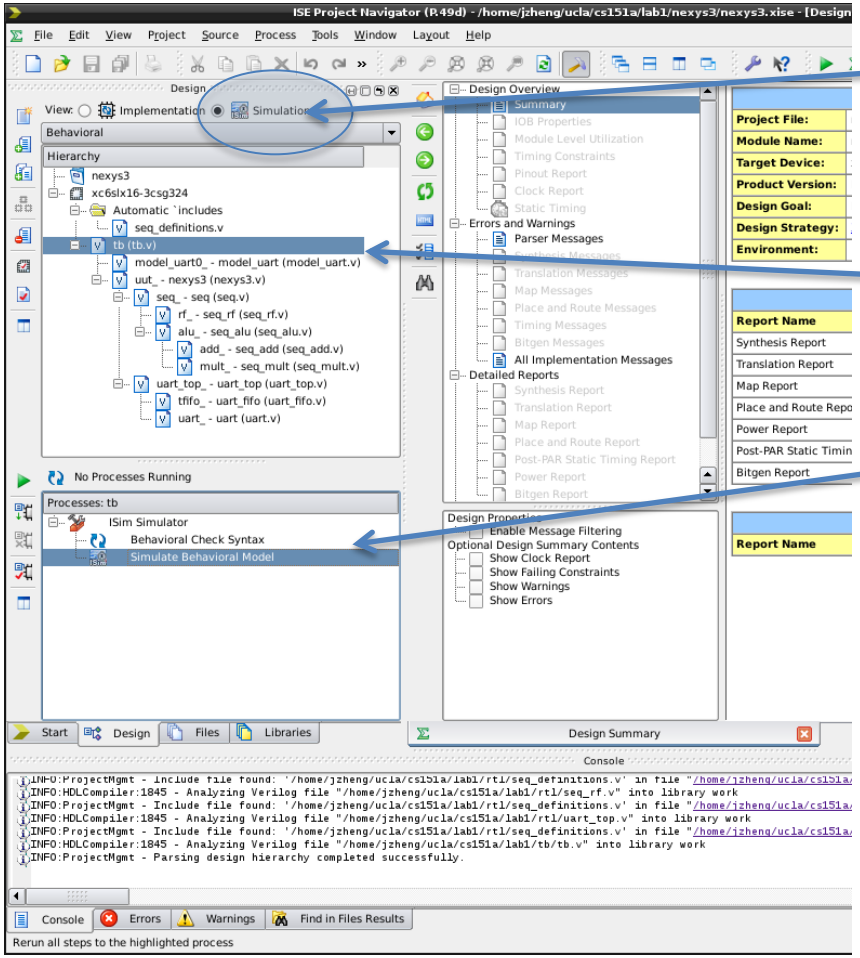
1. Click on “Add sources”
2. Select all files under the rtl subdirectory, including seq_defininitions.v

Add Simulation Files



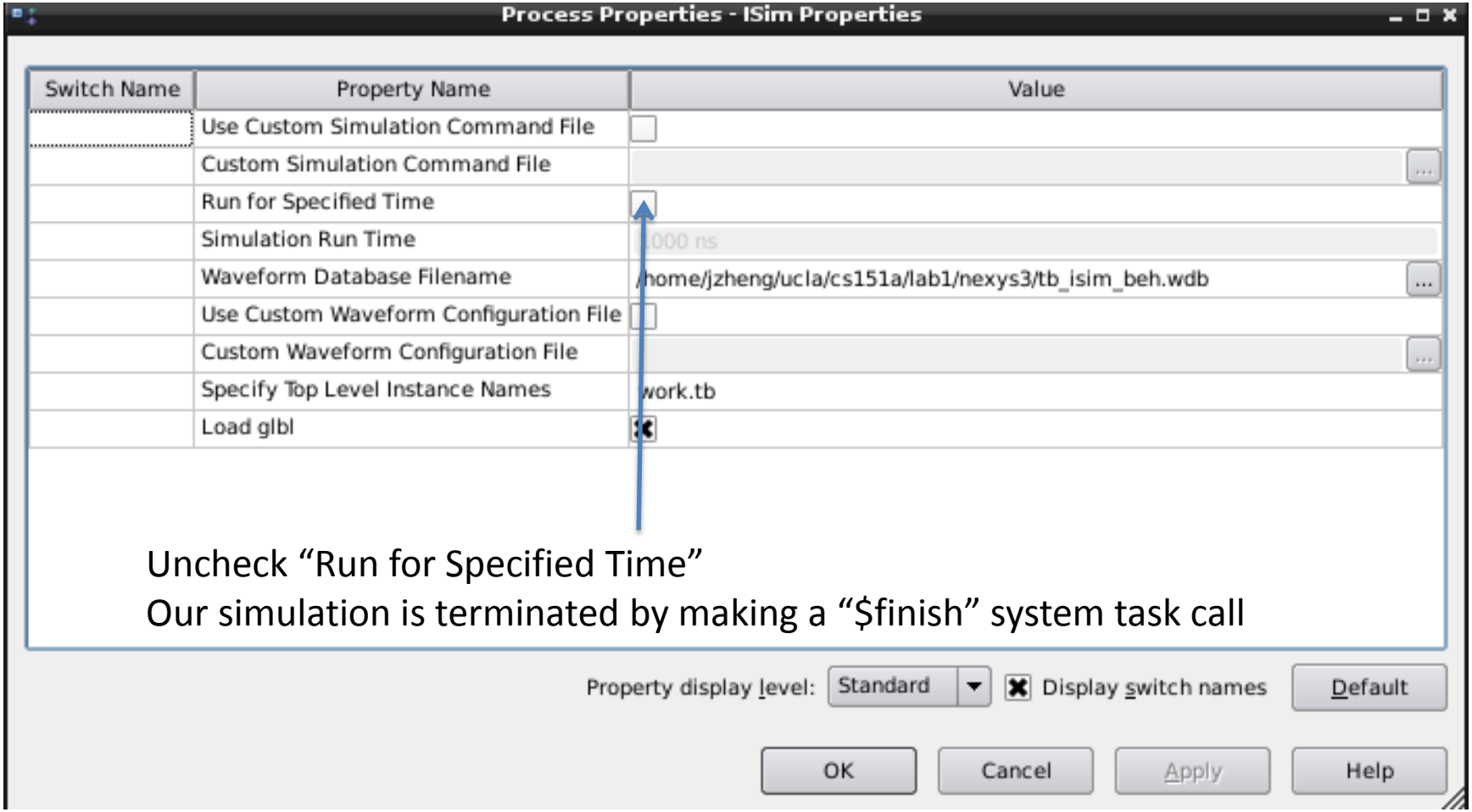
1. Click on “Add sources” again
2. Select all files under the tbench subdirectory.
3. Make sure their association is “Simulation”

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

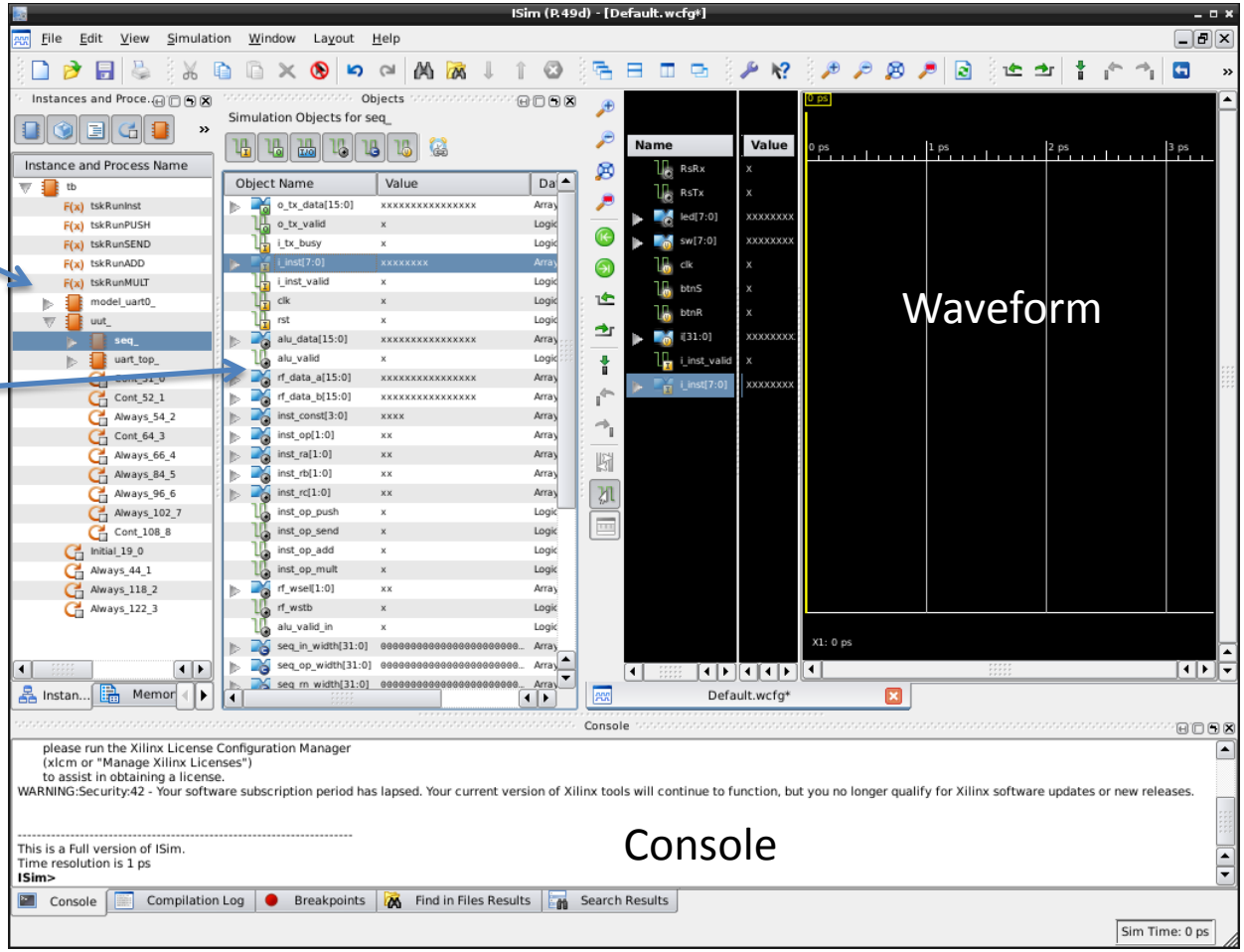
Launch ISIM

- 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

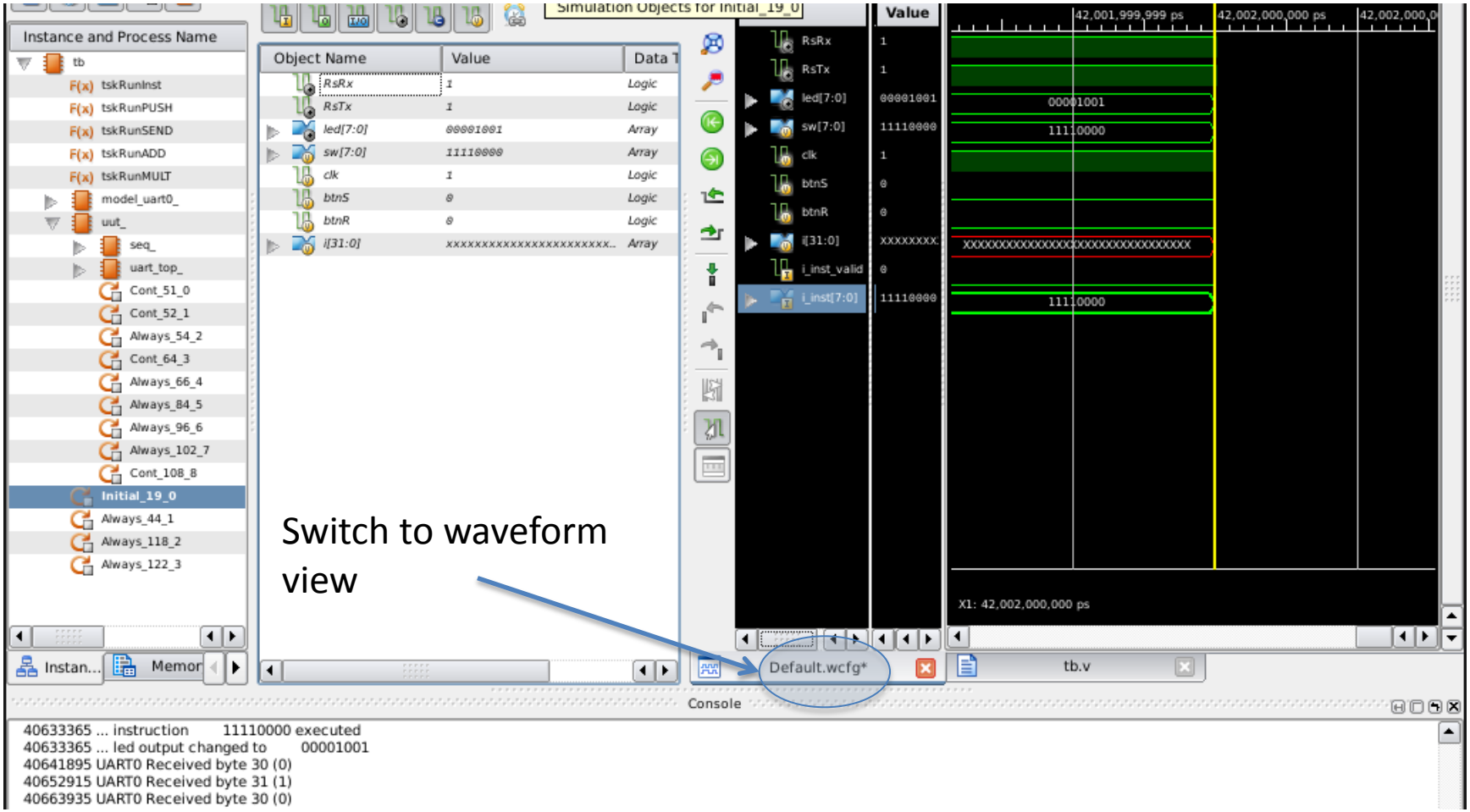
Signal View



Simulation

- Take a look at the provided test bench file tb.v, it run a small “program” that evaluates the exercise from slide 24.
- Spend some time in ISIM to add some signals to the waveform display
  - Find uut_ in hierarchical view
  - Add inst_wd and inst_vld to the waveform
  - These two signals indicate when the instruction is captured by the sequencer from the slider switches
- When ready, choose “Simulation” from the top menu, and “run all”
- Watch the output in the console and the waveform viewer

Post Simulation Examination

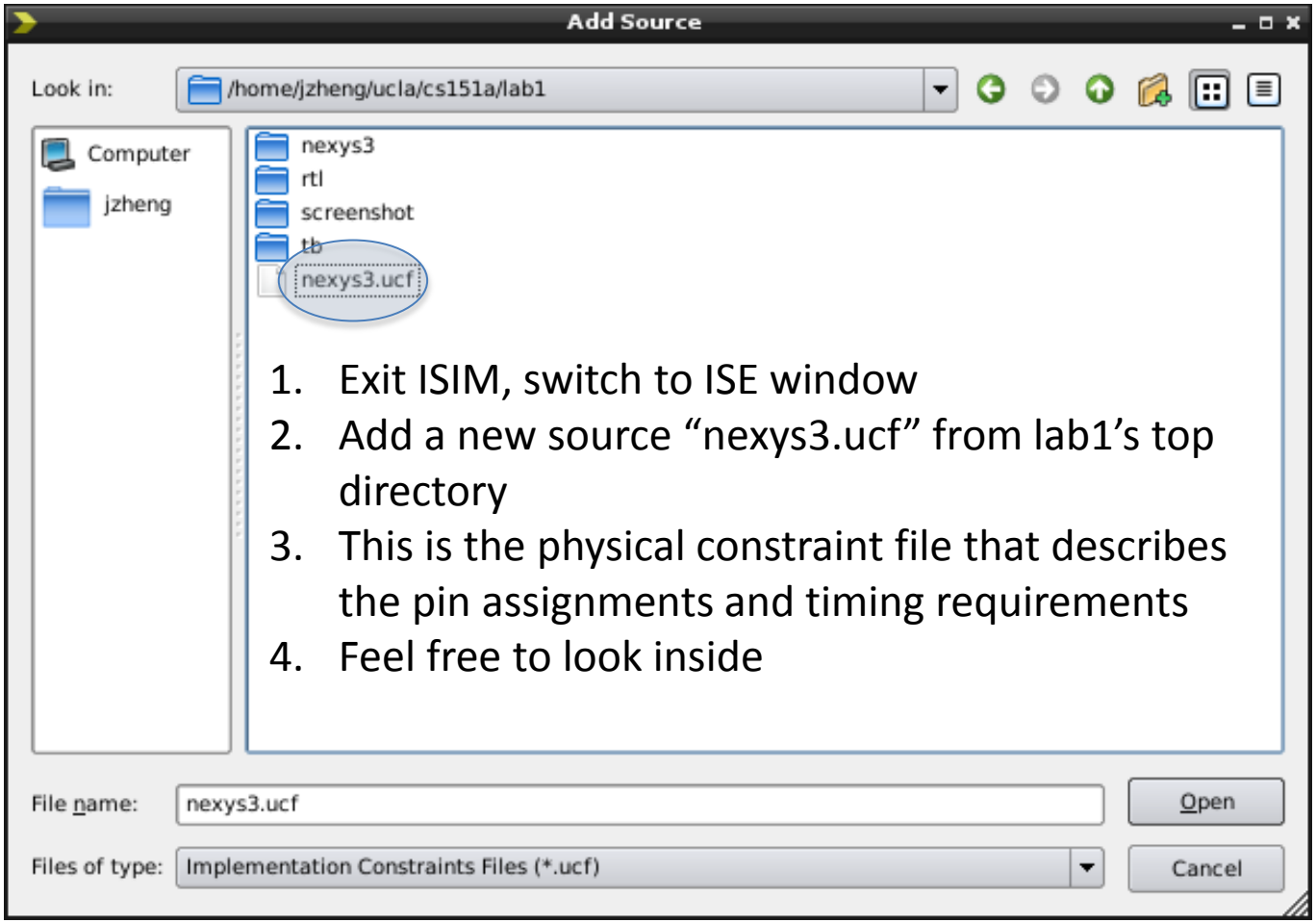


Simulation Console Output

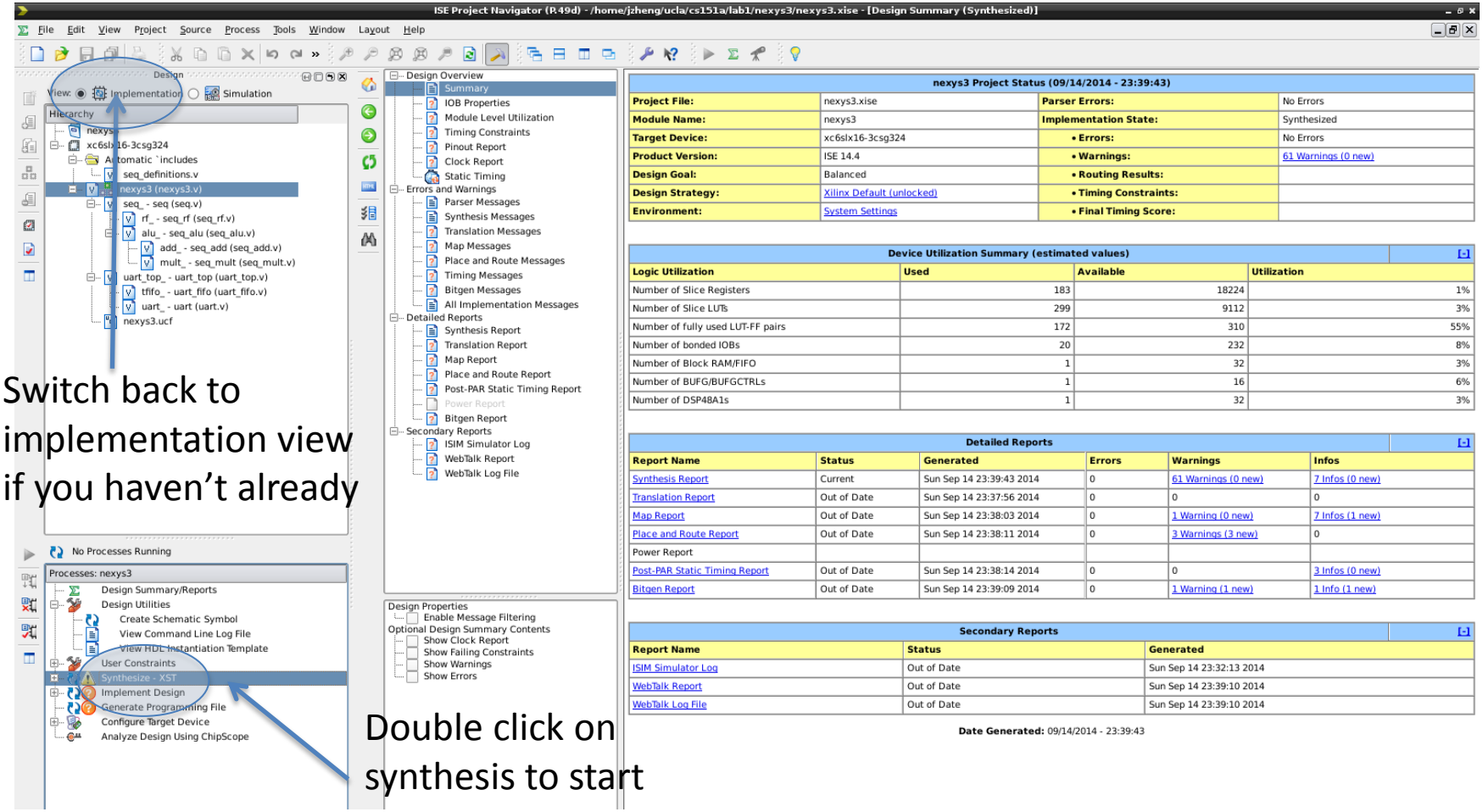
```
ISim>
# run all
Simulator is doing circuit initialization process.
Finished circuit initialization process.
        5 ... led output changed to       00000000
  1501000 ... Running instruction       00000100
  5243925 ... instruction       00000100 executed
  5243925 ... led output changed to       00000001
  6001000 ... Running instruction       00000000
  9176085 ... instruction       00000000 executed
  9176085 ... led output changed to       00000010
 10501000 ... Running instruction       00010011
 14418965 ... instruction       00010011 executed
 14418965 ... led output changed to       00000011
 15001000 ... Running instruction       10000110
 18351125 ... instruction       10000110 executed
 18351125 ... led output changed to       00000100
 19501000 ... Running instruction       01100011
 23594005 ... instruction       01100011 executed
 23594005 ... led output changed to       00000101
 24001000 ... Running instruction       11000000
 27526165 ... instruction       11000000 executed
 27526165 ... led output changed to       00000110
 27534695 UART0 Received byte 30 (0)
 27545715 UART0 Received byte 30 (0)
 27556735 UART0 Received byte 34 (4)
 27567755 UART0 Received byte 30 (0)
 27578775 UART0 Received byte 0a (
)
 28501000 ... Running instruction       11010000
 31458325 ... instruction       11010000 executed
 31458325 ... led output changed to       00000111
 31466855 UART0 Received byte 30 (0)
 31477875 UART0 Received byte 30 (0)
 31488895 UART0 Received byte 30 (0)
 31499915 UART0 Received byte 33 (3)
 31510935 UART0 Received byte 0a (
)
 33001000 ... Running instruction       11100000
 36701205 ... instruction       11100000 executed
 36701205 ... led output changed to       00001000
 36709735 UART0 Received byte 30 (0)
 36720755 UART0 Received byte 30 (0)
 36731775 UART0 Received byte 43 (C)
 36742795 UART0 Received byte 30 (0)
 36753815 UART0 Received byte 0a (
)
```

Do these UART0 outputs match your expectation (see slide 24)?

Prepare Project for Synthesis



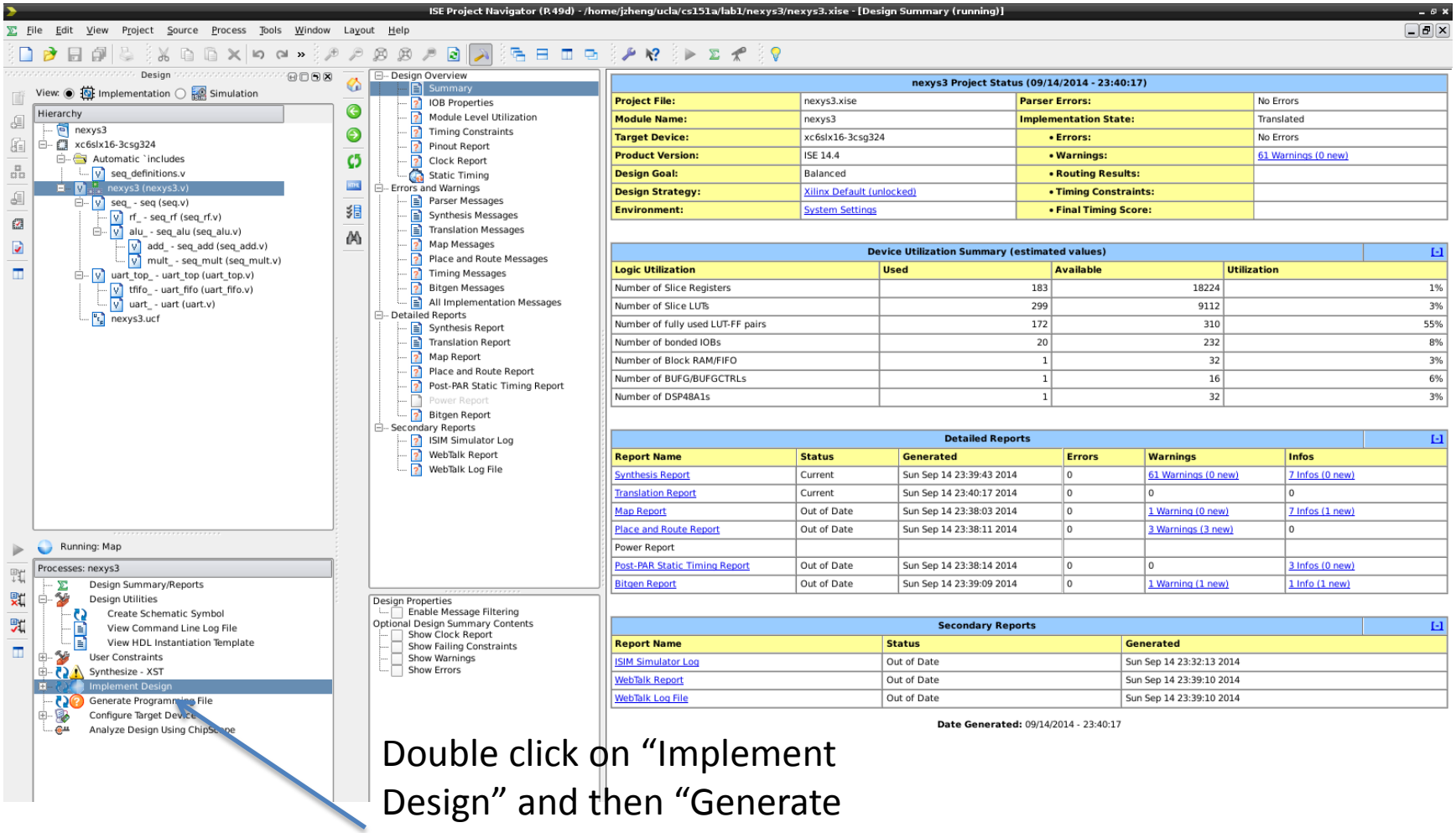
Ready for Synthesis



Switch back to implementation view if you haven't already

Double click on synthesis to start

Place-n-Route and Bitstream

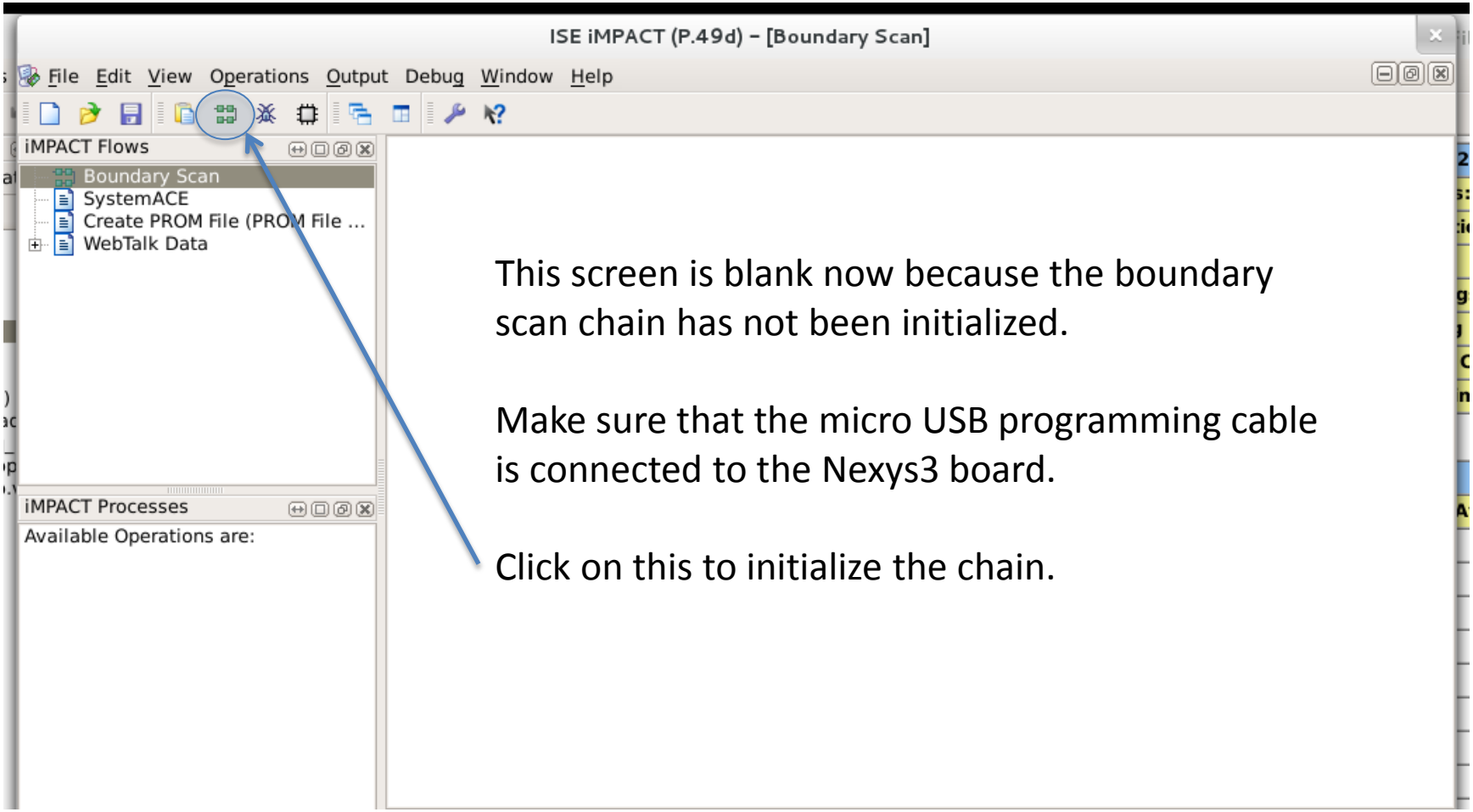


Double click on “Implement Design” and then “Generate Programming File”

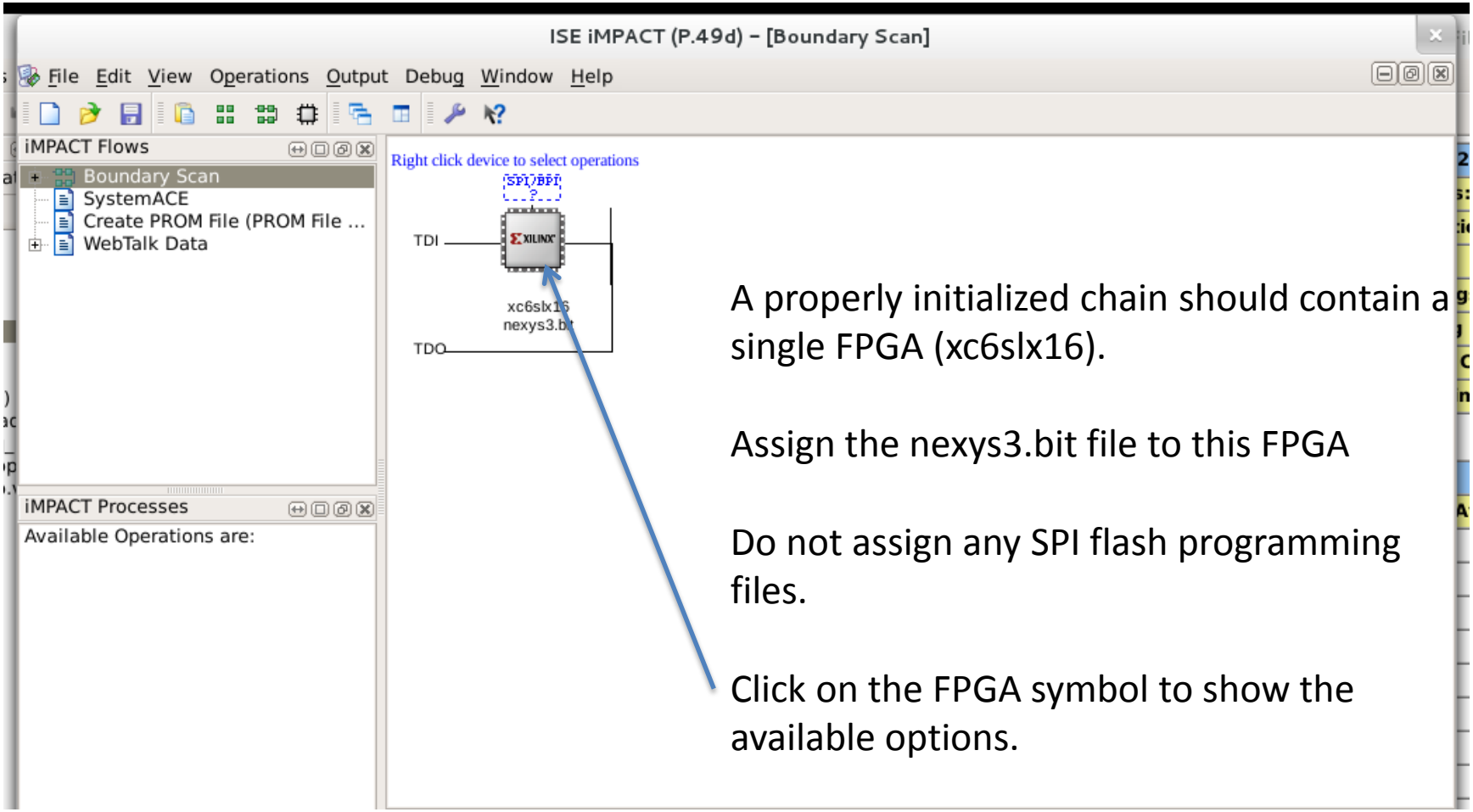
Download Bitstream to FPGA

- By now you should have a nexys3.bit file generated.
- You will now program the FPGA using this file.
- Click on “Configure Target Device” to open the Impact program.

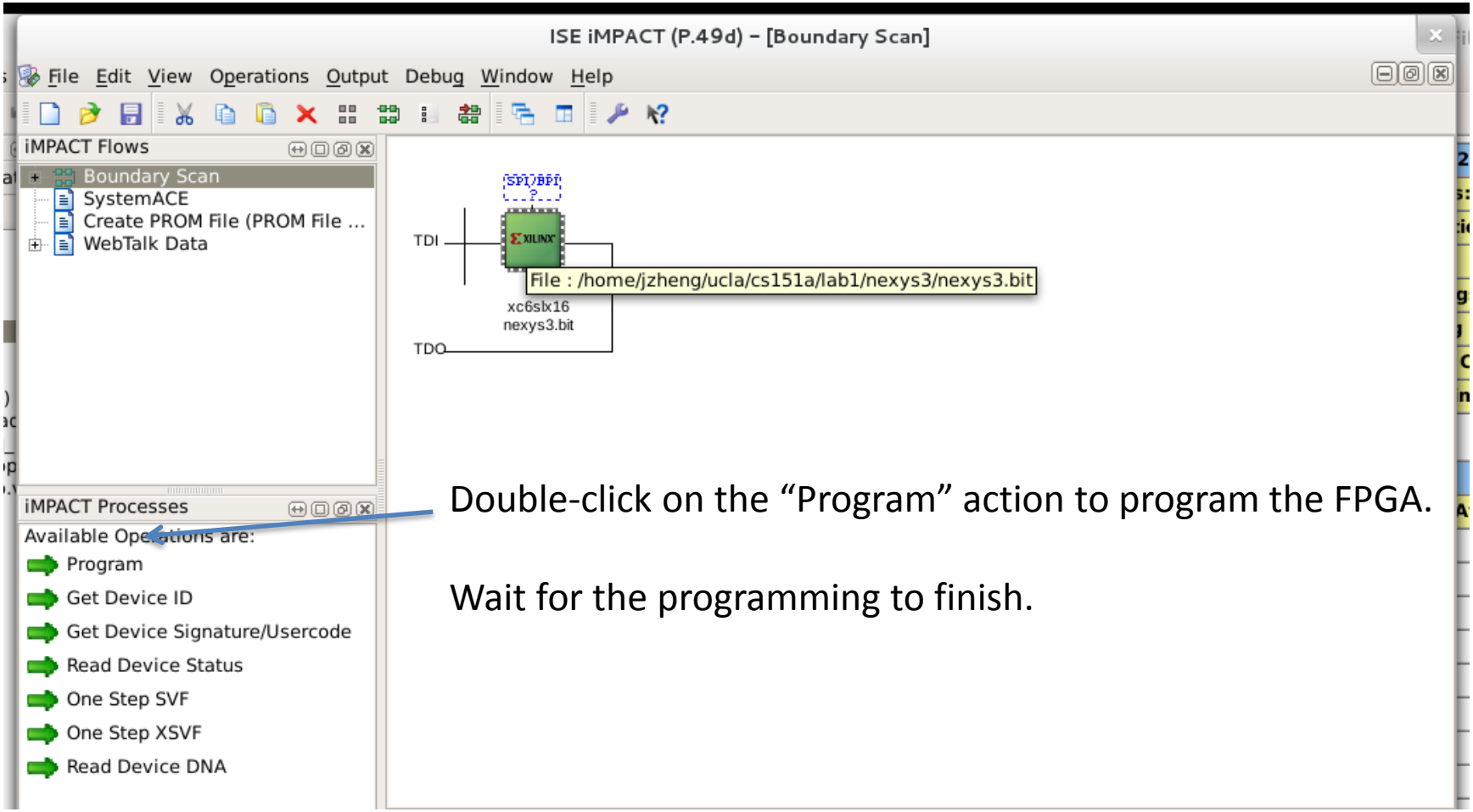
ISE Impact


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.

Before you start, please read through the lab 1 manual

PLAY TIME

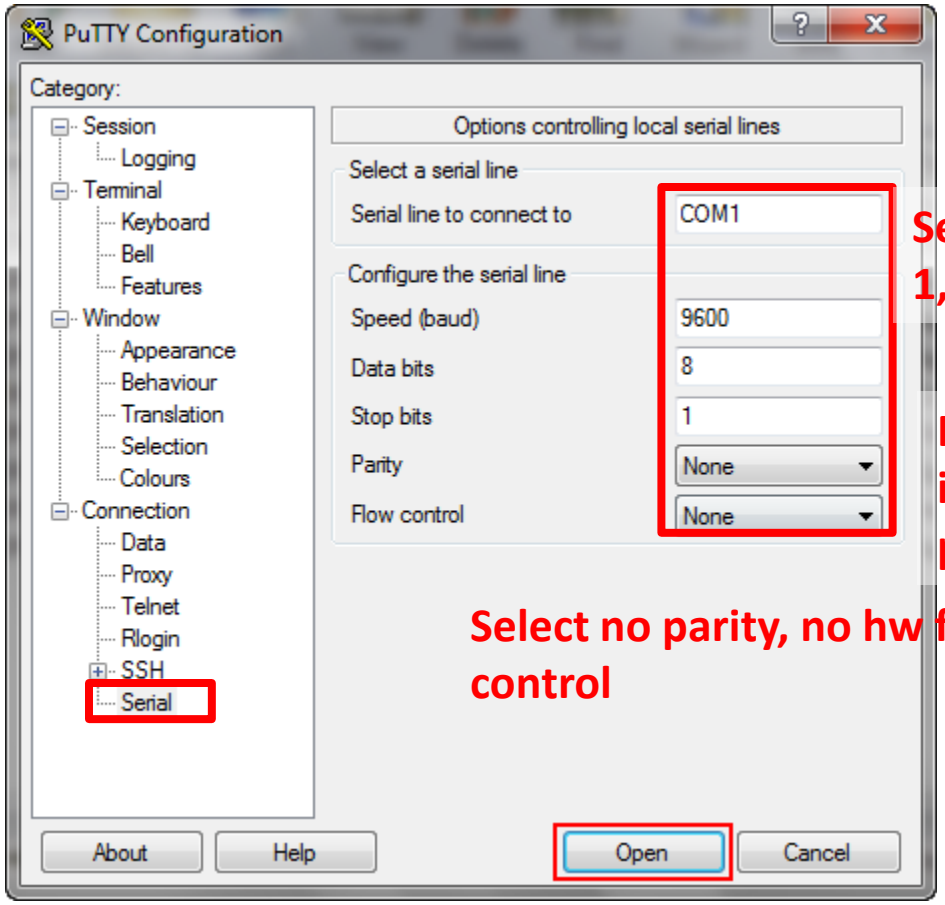
Optional: Install Driver

- This step is already done for all the PCs in the lab
- Windows:
  - May need to install FTDI serial driver
  - http://www.chipkin.com/using-putty-for-serial-com-connections-hyperterminal-replacement/
- Mac:
  - May need to install FTDI driver
  - Follow the minicom instructions from here http://pbxbook.com/other/mac-tty.html
- Linux:
  - Driver should be built-in: /dev/ttyUSB0
  - minicom –D /dev/ttyUSB0 –b 10000000
  - If you run into permission problems, it's usually because you are not a member of the "dialout" group

Setting up Console

- Open *putty*
- Set BAUD RATE 1000000, 1-8-1, no parity, no hw flow control (see the Appendix)

Appendix: Set up a Serial Console

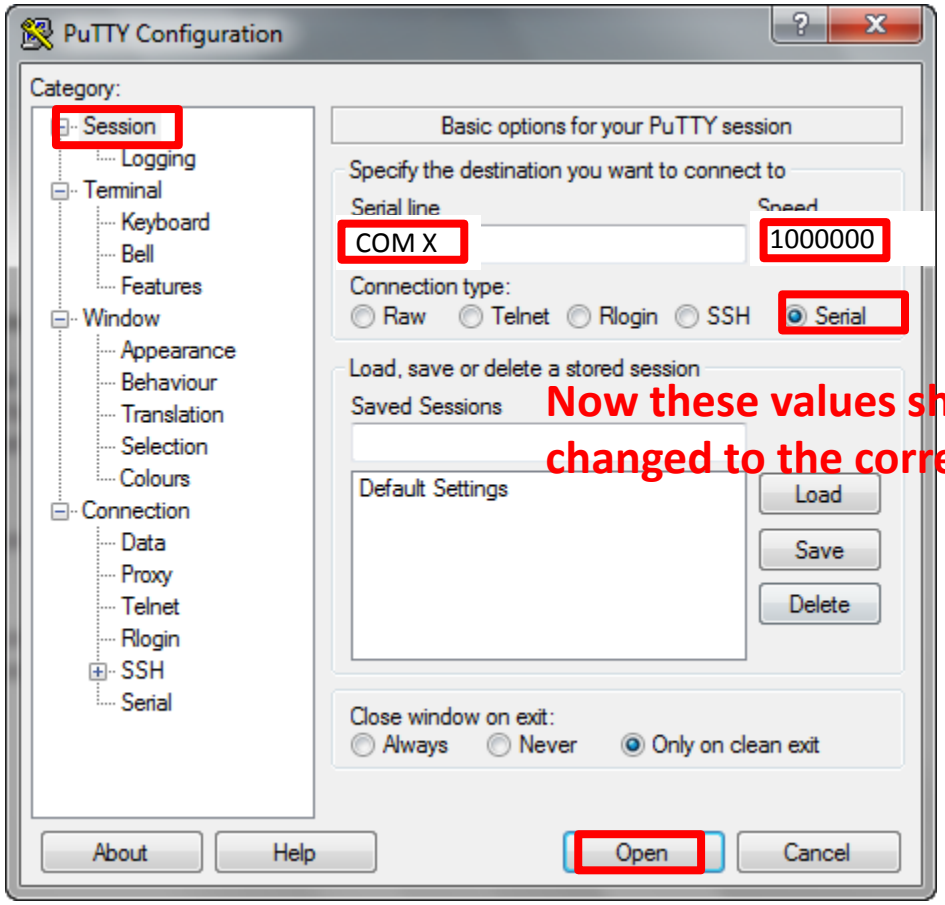


Set speed (baud rate) to 1,000,000

Find the COM port number in the device manager – ports – USB Serial Port

Select no parity, no hw flow control

Appendix: Set up a Serial Console



Now these values should have changed to the correct ones

Now you can see the output of “send” instructions.

Play with the board or work on exercises.