

# FPGA

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

# **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

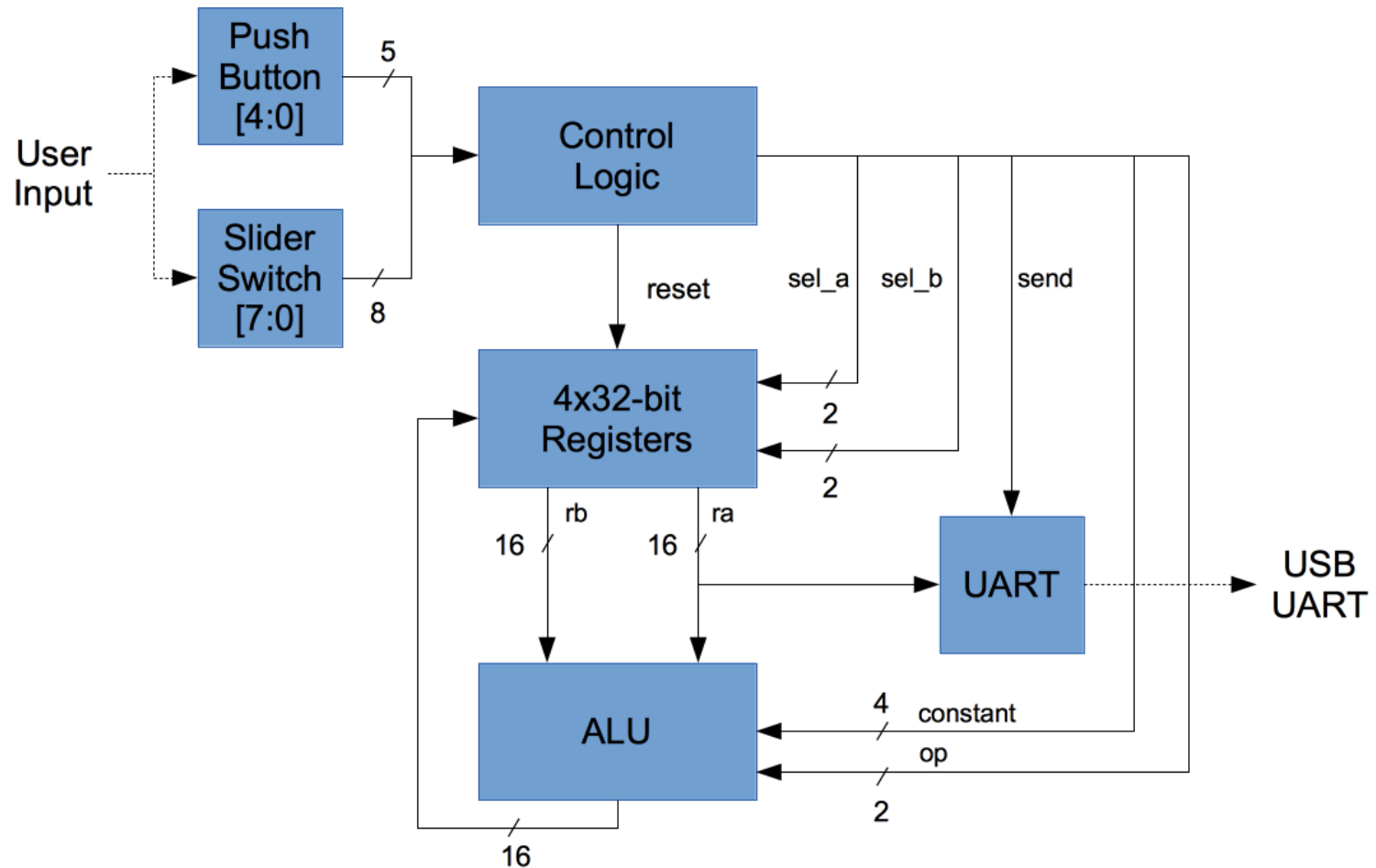
- Text editor of choice
- Simulator
  - ISE Webpack provides ISIM
  - Alternatively use free Modelsim PE
- Synthesis
  - ISE Webpack provides XST
  - Alternatively use Synplify Pro (evaluation version)
- Map, Place-and-Route
  - ISE Webpack

# **EXAMPLE PROJECT - SEQUENCER**

# Example Project

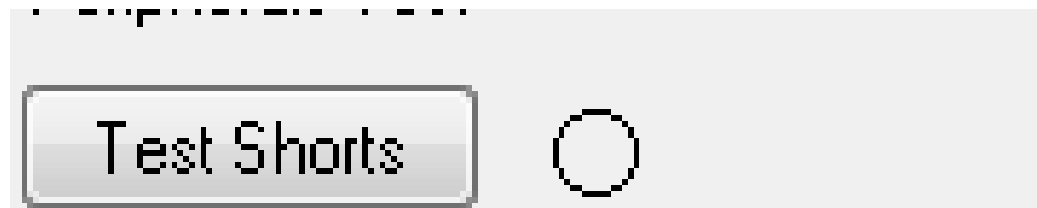
- The example project is a adder/multiplier sequencer
- The sequencer has four 16-bit general purpose registers
- The sequencer can perform add or multiply instructions using registers as operands
- The results are stored into the register file

# Project Diagram

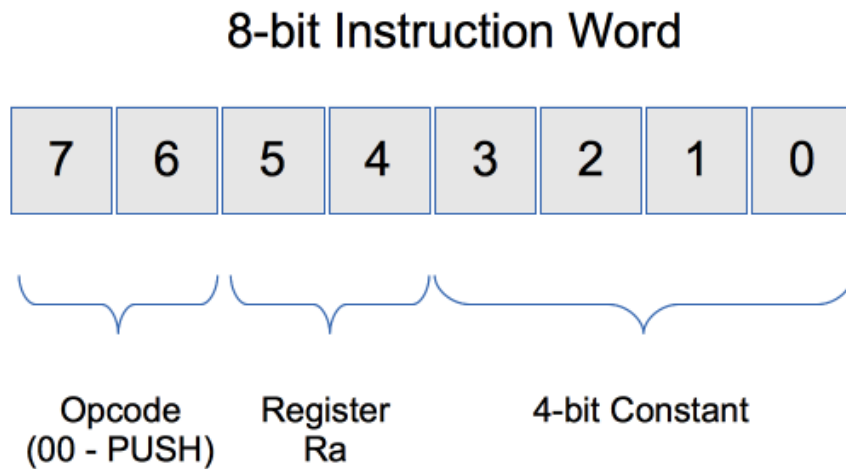


# Sequencer Operation

- To operate the sequencer, the user enters instructions using the switches and push buttons:
  - 8 switch positions represent a single instruction
  - Push center button to enter and execute the instruction
- The switch combination shown above represents an instruction of “11001100”



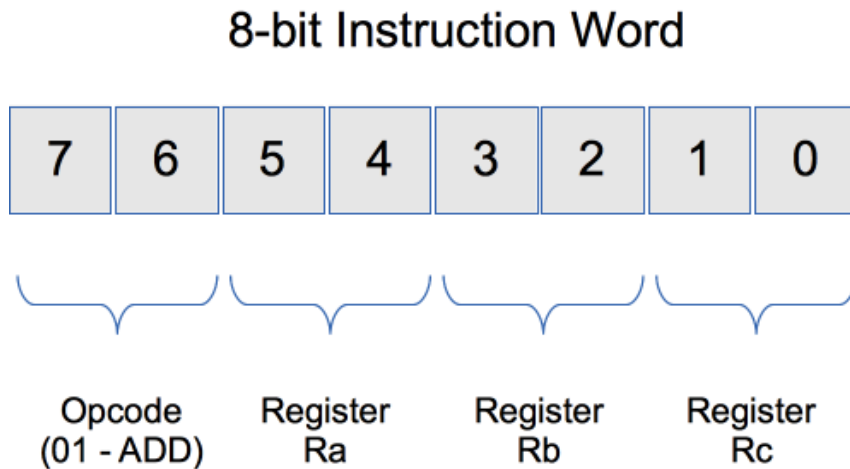
# Sequencer Instructions



- PUSH (00) instruction left-shifts the target register by 4 bits and “pushes” the new constant in.
- Example:
  - R0 starts with value 0x55AA
  - PUSH R0 0xB
  - Now R0 is 0x5AAB
- Exercise: how to set R0 to all zeros?

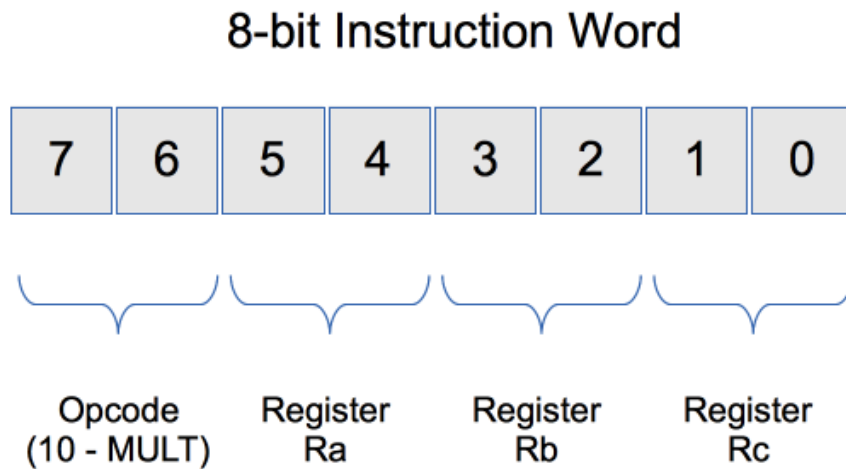


# Sequencer Instructions



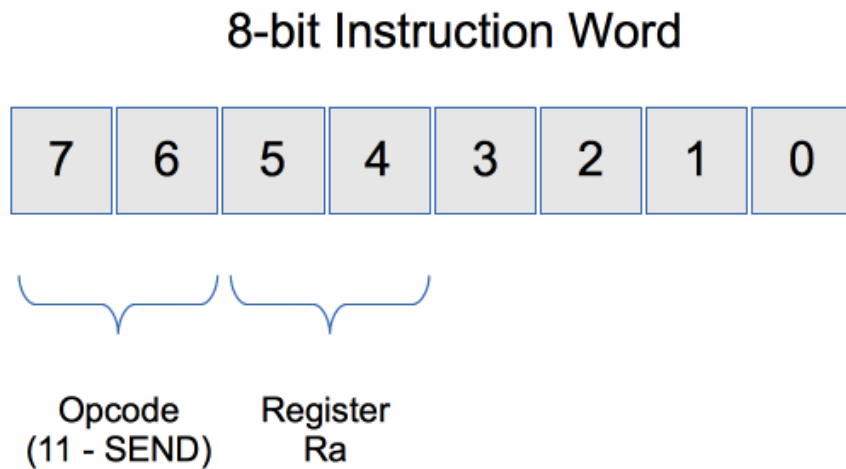
- ADD (01) instruction adds Ra and Rb and stores the result to Rc
- Addition is performed in unsigned integer arithmetic
- Example:
  - ADD R0 R1 R3
  - $R0 + R1 \rightarrow R3$

# Sequencer Instructions



- MULT (10) instruction multiplies Ra and Rb and stores the result to Rc
- Inputs are assumed to be unsigned integers
- Only the lower 16-bit results are retained
- Example:
  - MULT R3 R0 R2
  - $R3 * R0 \rightarrow R2$

# Sequencer Instructions



- SEND (11) instruction sends the content of Rt to the UART for display
- The 16-bit value is converted to ASCIIHEX and appended with a newline character
- Example:
  - R1 has a value of 0x1234
  - SEND R1 causes the UART to send “1234\n”

# Other Operations

- The push button BTNR (right button) resets values of all registers to 0
- The 8 LED indicators shows the number of instructions executed since the last reset, in binary

# Exercise

- What's the output of the following instructions (assuming the reset is pushed just now)?
  - PUSH R0 0x4
  - PUSH R0 0x0
  - PUSH R1 0x3
  - MULT R0 R1 R2
  - ADD R2 R0 R3
  - SEND R0
  - SEND R1
  - SEND R2
  - SEND R3

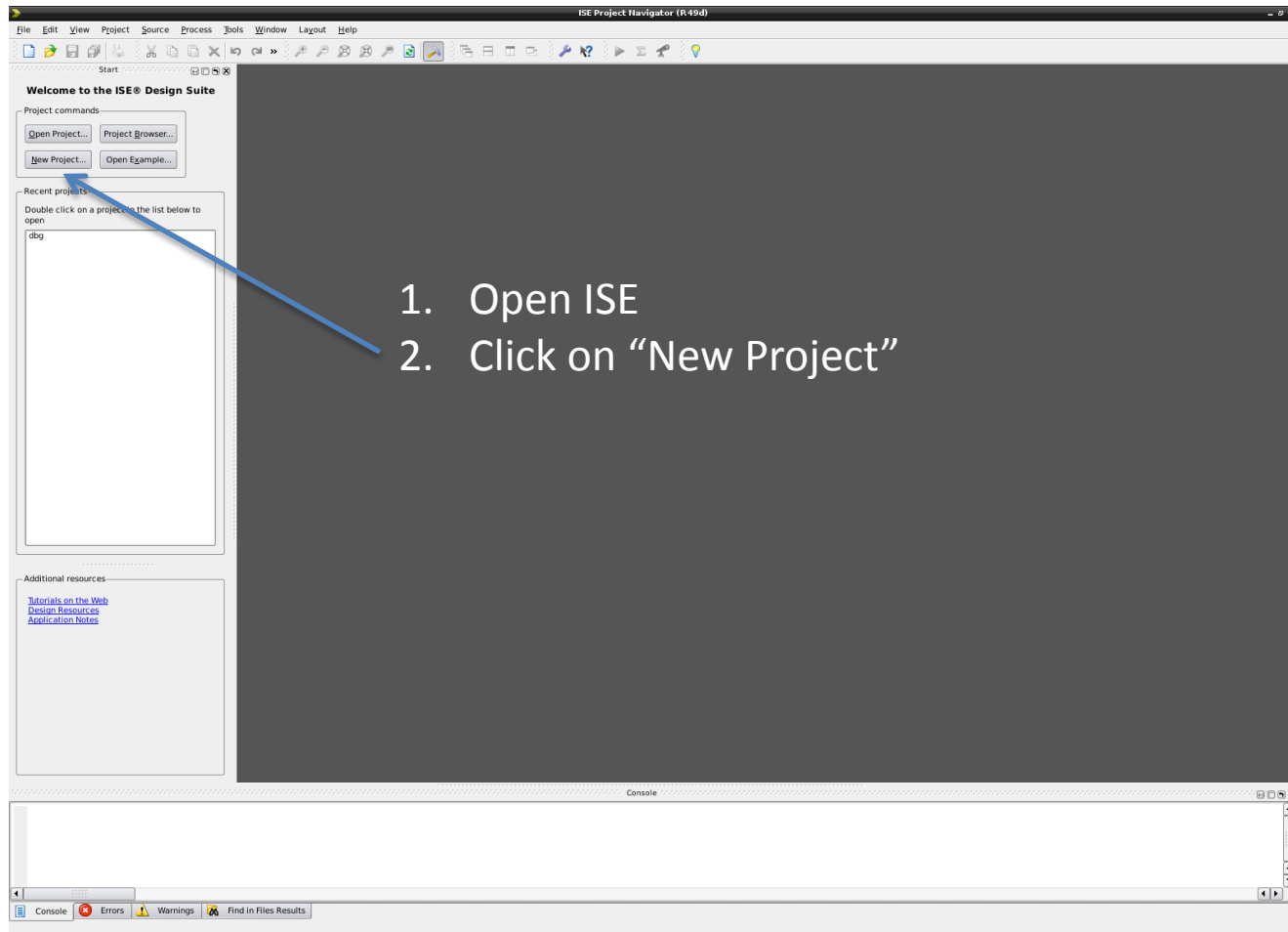
Now is the time to create the ISE project and go through steps 2-8

# **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

The screenshot shows the 'New Project Wizard' dialog box with the following fields and annotations:

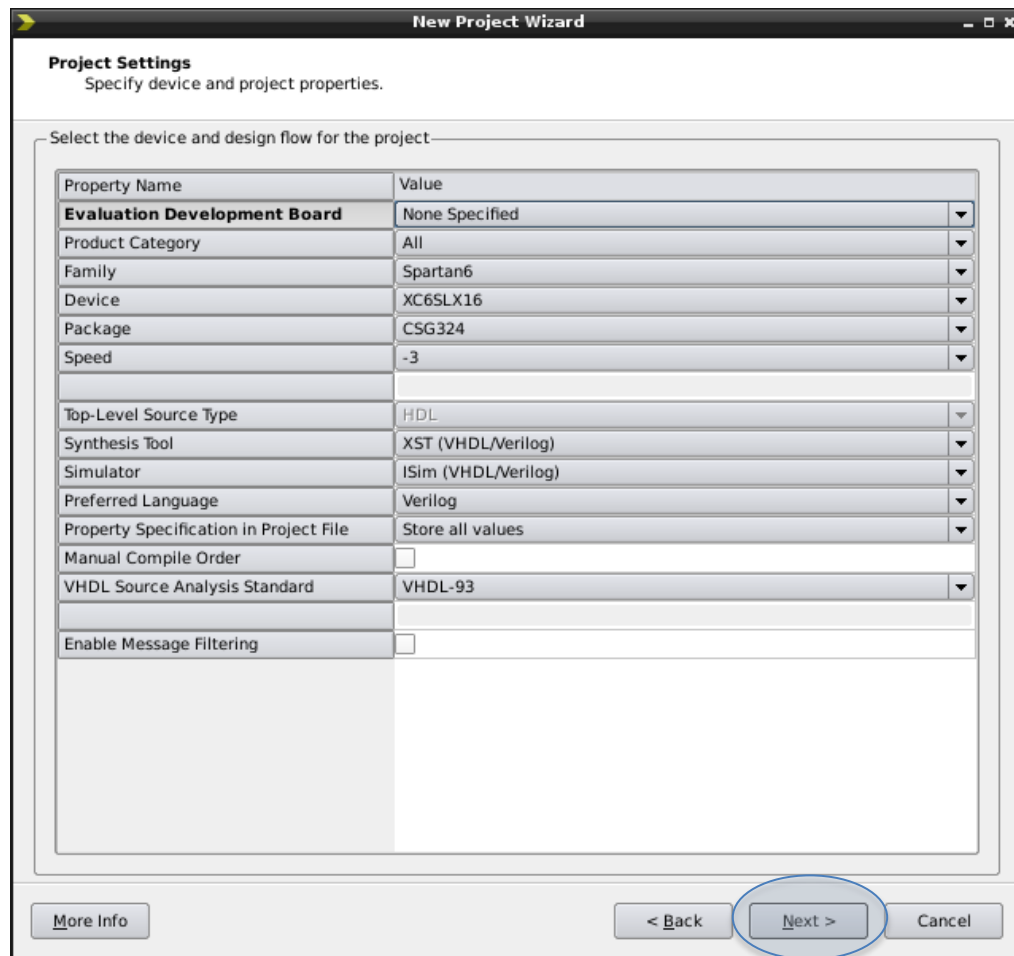
- Name:** nexys3
- Location:** /home/jzheng/ucla/cs151a/lab1/nexys3 (An arrow points to this field with the text: 'No spaces in the path!')
- Working Directory:** /home/jzheng/ucla/cs151a/lab1/nexys3 (An arrow points to this field with the text: 'Choose "HDL" project')
- Description:** (Empty text area)
- Top-level source type:** HDL (An arrow points to this dropdown menu)
- Buttons:** More Info, Next > (circled), Cancel

Annotations:

- No spaces in the path!
- Choose "HDL" project

# Device Properties

Make sure the fields match what you see here

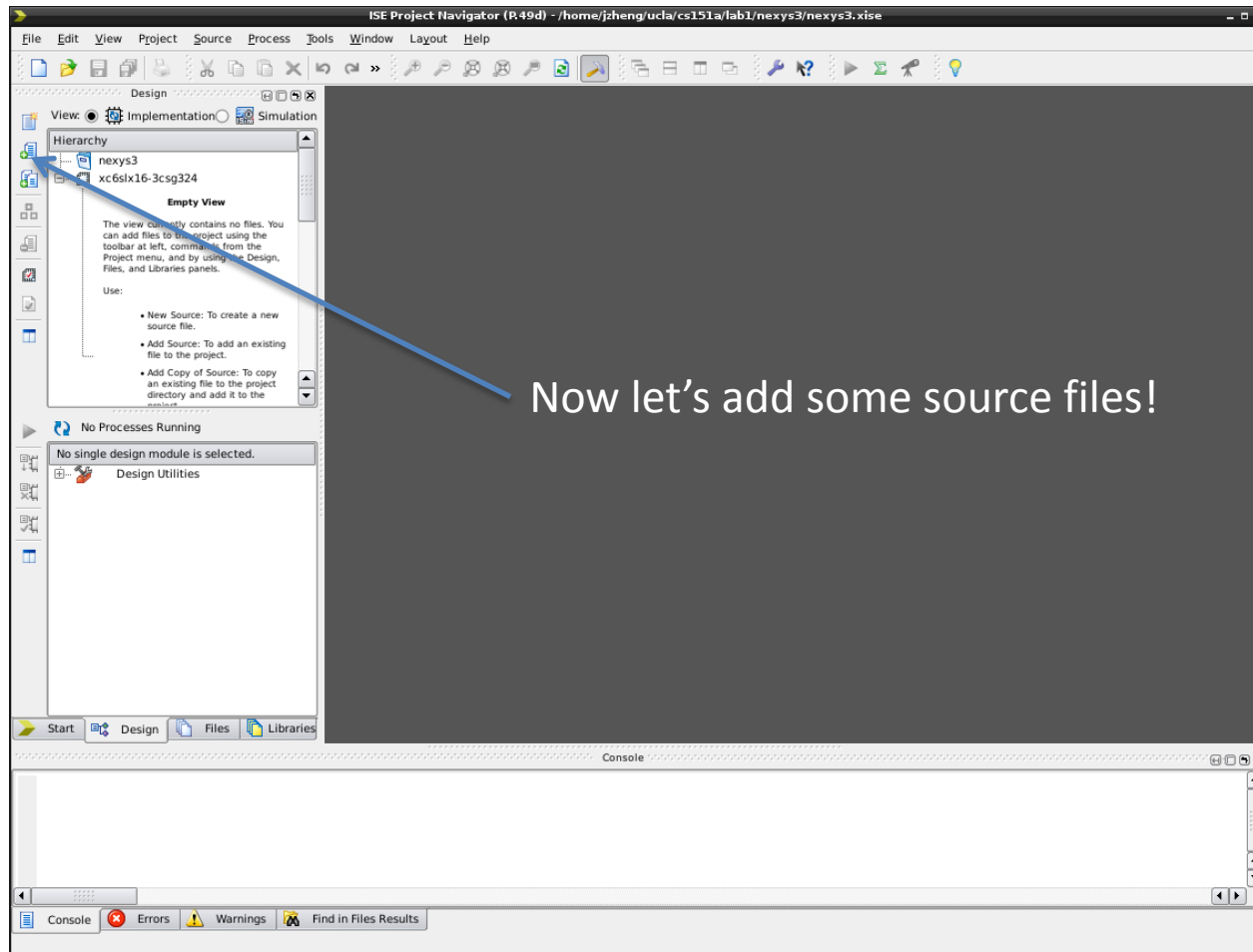


The image shows the 'New Project Wizard' dialog box, specifically the 'Project Settings' tab. The title bar reads 'New Project Wizard'. Below the title bar, the text 'Project Settings' is followed by 'Specify device and project properties.' The main area is titled 'Select the device and design flow for the project'. It contains a table with two columns: 'Property Name' and 'Value'. The table lists various project settings, most of which are dropdown menus. The 'Evaluation Development Board' property is highlighted. At the bottom of the dialog, there are three buttons: 'More Info', '< Back', and 'Next >'. The 'Next >' button is circled in blue.

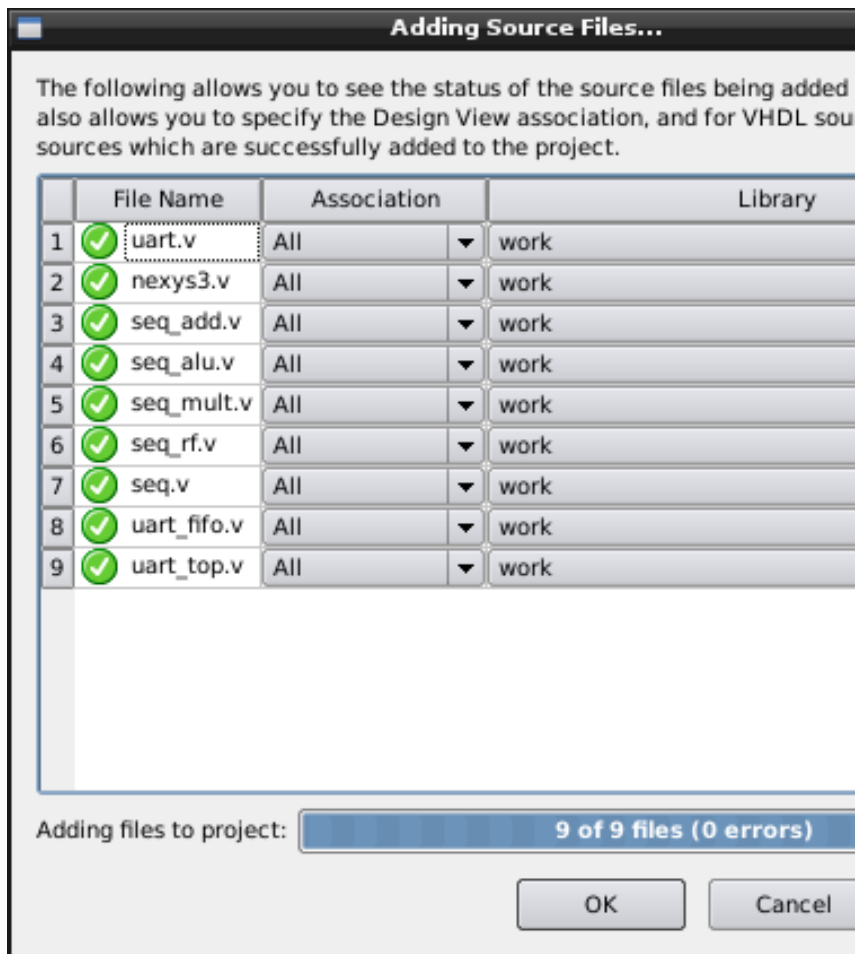
Property Name	Value
<b>Evaluation Development Board</b>	None Specified
Product Category	All
Family	Spartan6
Device	XC6SLX16
Package	CSG324
Speed	-3
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	Verilog
Property Specification in Project File	Store all values
Manual Compile Order	<input type="checkbox"/>
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	<input type="checkbox"/>

More Info    < Back    **Next >**    Cancel

# Project Created

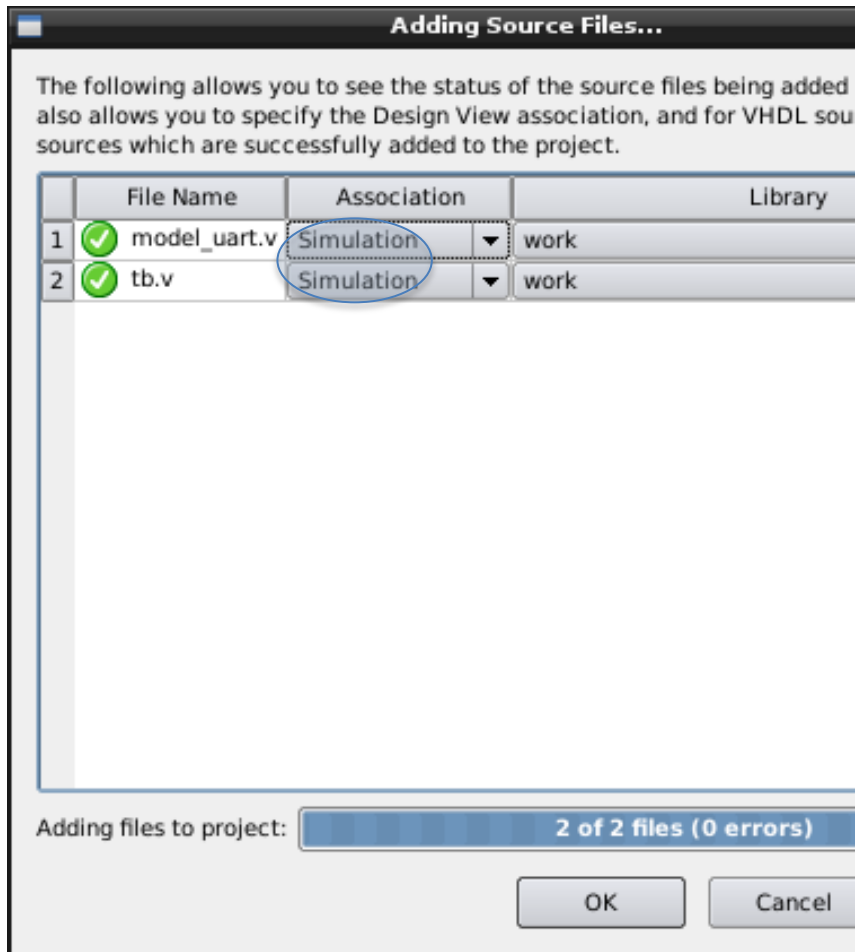


# Add RTL Sources



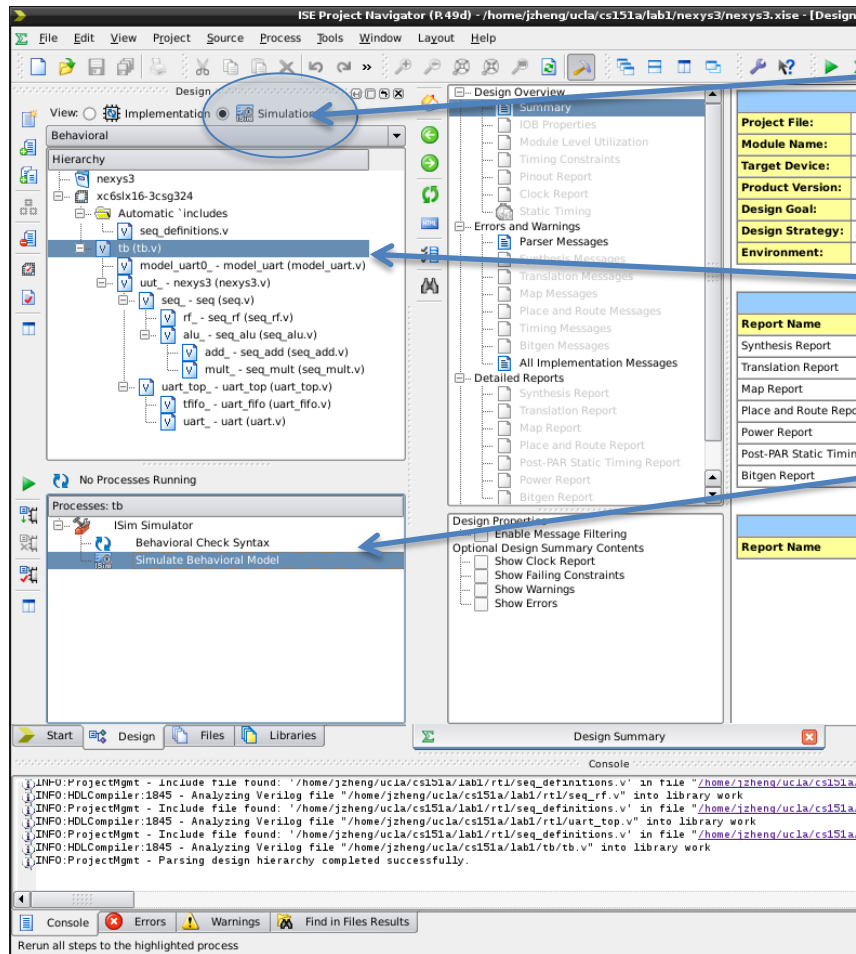
1. Click on “Add sources”
2. Select all files under the rtl subdirectory, except for seq\_defininitions.v

# Add Simulation Files



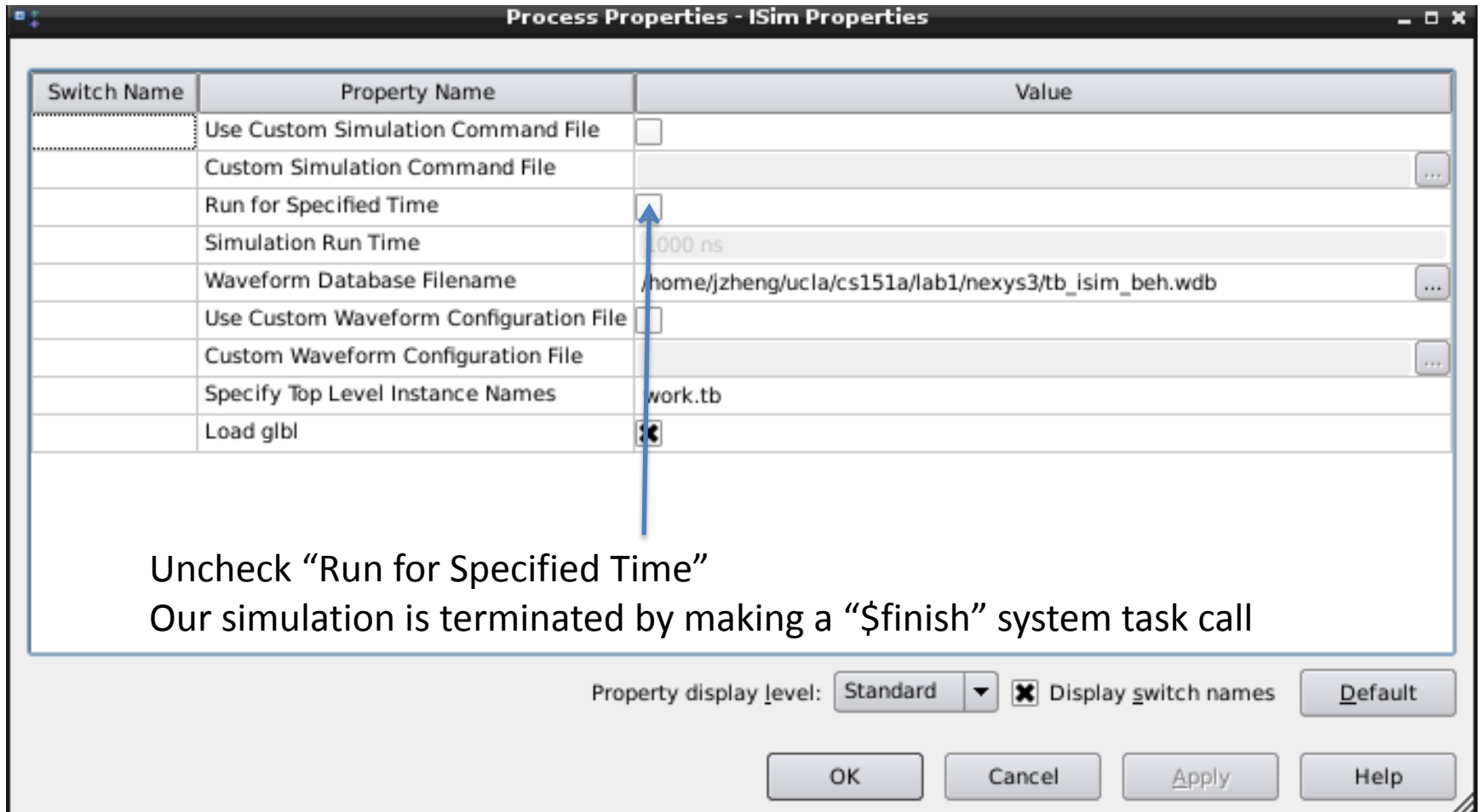
1. Click on “Add sources” again
2. Select all files under the tbench subdirectory.
3. Make sure their association is “Simulation”

# 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



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 Default

OK Cancel Apply Help

# Launch ISIM

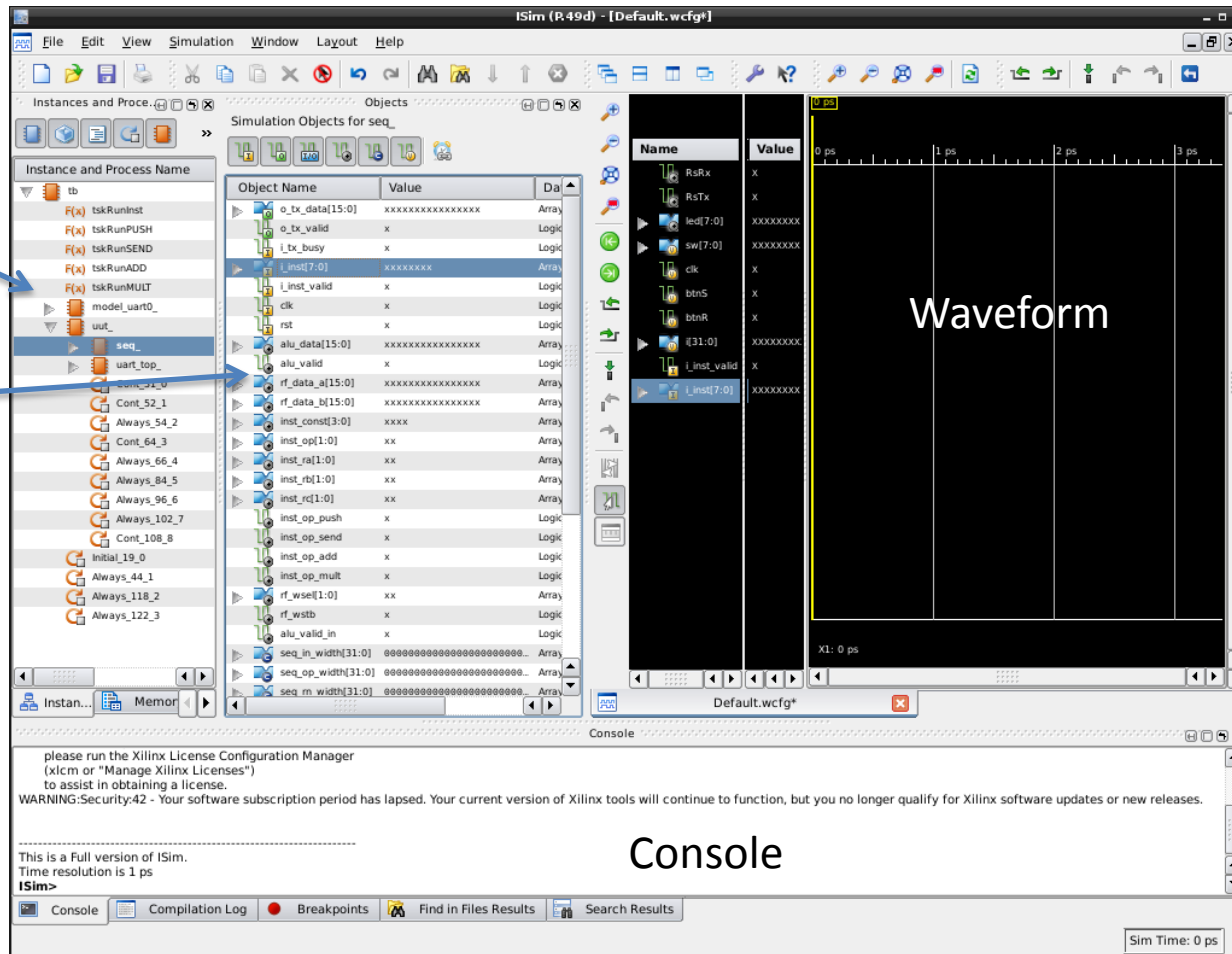
- Right click on “Simulate Behavioral Model” again, this time choose “run all”
- ISIM will be launched
- ISIM is the simulation environment where you can dynamically debug the circuit, much like a software debugger
- Your main focus should be on the console window and the waveform window



# 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

The screenshot displays a digital logic simulation interface. On the left, a tree view shows the 'Instance and Process Name' hierarchy, with 'Initial\_19\_0' selected. The central pane, titled 'Simulation Objects for Initial\_19\_0', lists various objects and their current values. A blue arrow points from the text 'Switch to waveform view' to the 'Waveform' icon in the toolbar. The right pane shows a list of variables and their values, with 'i\_inst[7:0]' highlighted. The bottom pane shows a console window with simulation logs.

Object Name	Value	Data Type
RsRx	1	Logic
RsTx	1	Logic
led[7:0]	00001001	Array
sw[7:0]	11110000	Array
clk	1	Logic
btnS	0	Logic
btnR	0	Logic
i[31:0]	xxxxxxxxxxxxxxxxxxxxxx...	Array

Switch to waveform view

Default.wcfg\*

tb.v

Console

40633365 ... instruction 11110000 executed  
40633365 ... led output changed to 00001001  
40641895 UART0 Received byte 30 (0)  
40652915 UART0 Received byte 31 (1)  
40663935 UART0 Received byte 30 (0)

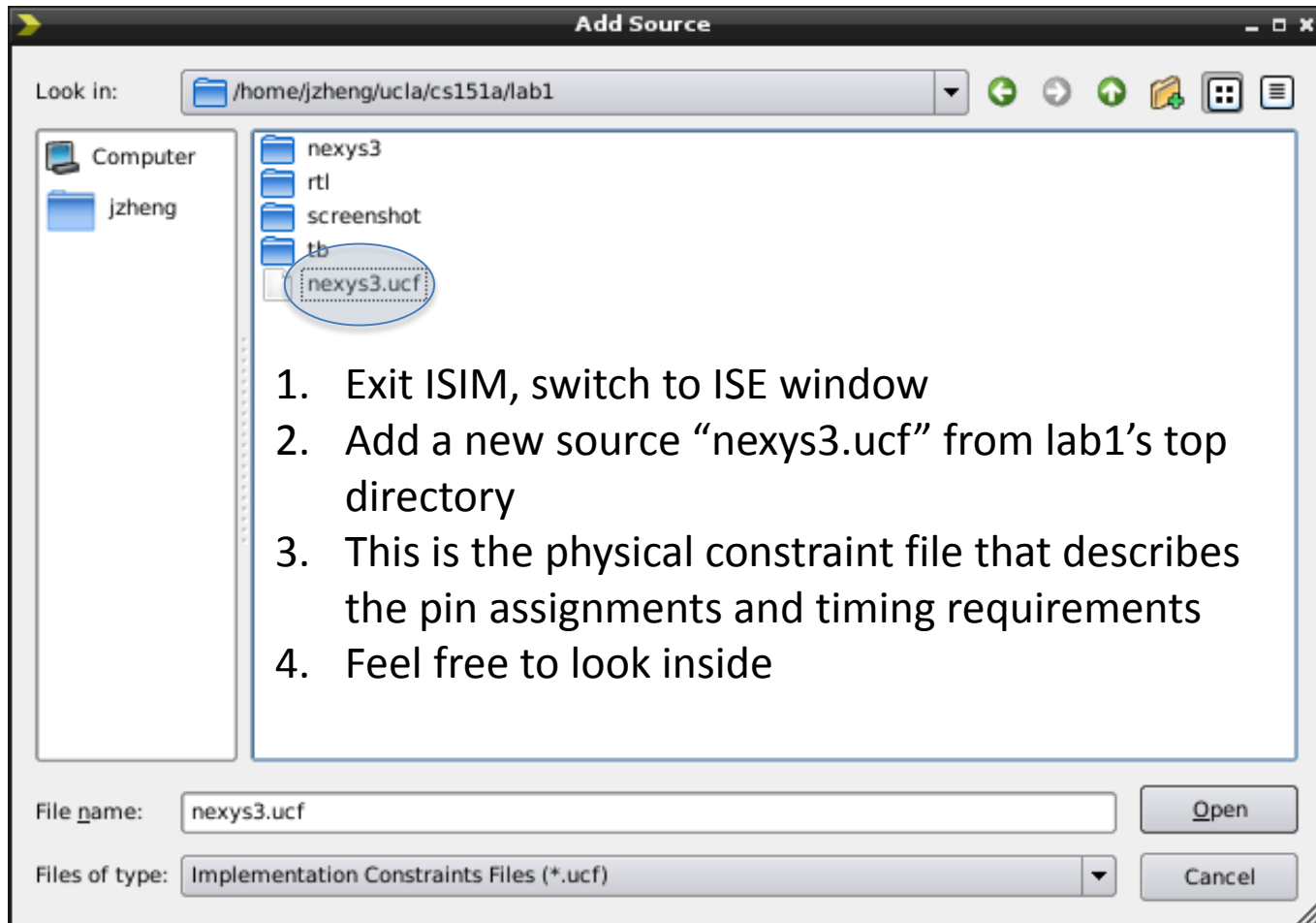
# Simulation Console Output

```
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

ISE Project Navigator (P.49d) - /home/jzheng/ucla/cs151a/lab1/nexys3/nexys3.xise - [Design Summary (Synthesized)]

View: **Design** Implementation Simulation

**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
  - Bitgen Report
- Secondary Reports
  - ISIM Simulator Log
  - WebTalk Report
  - WebTalk Log File

**nexys3 Project Status (09/14/2014 - 23:39:43)**

<b>Project File:</b>	nexys3.xise	<b>Parser Errors:</b>	No Errors
<b>Module Name:</b>	nexys3	<b>Implementation State:</b>	Synthesized
<b>Target Device:</b>	xc6slx16-3csg324	<b>Errors:</b>	No Errors
<b>Product Version:</b>	ISE 14.4	<b>Warnings:</b>	61 Warnings (0 new)
<b>Design Goal:</b>	Balanced	<b>Routing Results:</b>	
<b>Design Strategy:</b>	Xilinx Default (unlocked)	<b>Timing Constraints:</b>	
<b>Environment:</b>	System Settings	<b>Final Timing Score:</b>	

**Device Utilization Summary (estimated values)**

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	183	18224	1%
Number of Slice LUTs	299	9112	3%
Number of fully used LUT-FF pairs	172	310	55%
Number of bonded IOBs	20	232	8%
Number of Block RAM/FIFO	1	32	3%
Number of BUFG/BUFGCTRLs	1	16	6%
Number of DSP48A1s	1	32	3%

**Detailed Reports**

Report Name	Status	Generated	Errors	Warnings	Infos
<a href="#">Synthesis Report</a>	Current	Sun Sep 14 23:39:43 2014	0	61 Warnings (0 new)	7 Infos (0 new)
<a href="#">Translation Report</a>	Out of Date	Sun Sep 14 23:37:56 2014	0	0	0
<a href="#">Map Report</a>	Out of Date	Sun Sep 14 23:38:03 2014	0	1 Warning (0 new)	7 Infos (1 new)
<a href="#">Place and Route Report</a>	Out of Date	Sun Sep 14 23:38:11 2014	0	3 Warnings (3 new)	0
<a href="#">Power Report</a>					
<a href="#">Post-PAR Static Timing Report</a>	Out of Date	Sun Sep 14 23:38:14 2014	0	0	3 Infos (0 new)
<a href="#">Bitgen Report</a>	Out of Date	Sun Sep 14 23:39:09 2014	0	1 Warning (1 new)	1 Info (1 new)

**Secondary Reports**

Report Name	Status	Generated
<a href="#">ISIM Simulator Log</a>	Out of Date	Sun Sep 14 23:32:13 2014
<a href="#">WebTalk Report</a>	Out of Date	Sun Sep 14 23:39:10 2014
<a href="#">WebTalk Log File</a>	Out of Date	Sun Sep 14 23:39:10 2014

Date Generated: 09/14/2014 - 23:39:43

Processes: nexys3

- Design Summary/Reports
- Design Utilities
  - Create Schematic Symbol
  - View Command Line Log File
  - View HDL Instantiation Template
- User Constraints
- Synthesize - XST**
- Implement Design
- Generate Programming File
- Configure Target Device
- Analyze Design Using ChipScope

# Place-n-Route and Bitstream

The screenshot displays the ISE Project Navigator (R49d) interface for the project 'nexys3.xise'. The Design Summary (running) is shown, providing a comprehensive overview of the project's status and resources.

**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
  - Bitgen Report
- Secondary Reports
  - ISIM Simulator Log
  - WebTalk Report
  - WebTalk Log File

**Design Summary (running)**

nexys3 Project Status (09/14/2014 - 23:40:17)			
<b>Project File:</b>	nexys3.xise	<b>Parser Errors:</b>	No Errors
<b>Module Name:</b>	nexys3	<b>Implementation State:</b>	Translated
<b>Target Device:</b>	xc6slx16-3csg324	<b>Errors:</b>	No Errors
<b>Product Version:</b>	ISE 14.4	<b>Warnings:</b>	61 Warnings (0 new)
<b>Design Goal:</b>	Balanced	<b>Routing Results:</b>	
<b>Design Strategy:</b>	Xilinx Default (unlocked)	<b>Timing Constraints:</b>	
<b>Environment:</b>	System Settings	<b>Final Timing Score:</b>	

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	183	18224	1%
Number of Slice LUTs	299	9112	3%
Number of fully used LUT-FF pairs	172	310	55%
Number of bonded IOBs	20	232	8%
Number of Block RAM/FIFO	1	32	3%
Number of BUFGBUFCTRLs	1	16	6%
Number of DSP48A1s	1	32	3%

Detailed Reports					
Report Name	Status	Generated	Errors	Warnings	Infos
<a href="#">Synthesis Report</a>	Current	Sun Sep 14 23:39:43 2014	0	61 Warnings (0 new)	7 Infos (0 new)
<a href="#">Translation Report</a>	Current	Sun Sep 14 23:40:17 2014	0	0	0
<a href="#">Map Report</a>	Out of Date	Sun Sep 14 23:38:03 2014	0	1 Warning (0 new)	7 Infos (1 new)
<a href="#">Place and Route Report</a>	Out of Date	Sun Sep 14 23:38:11 2014	0	3 Warnings (3 new)	0
<a href="#">Power Report</a>					
<a href="#">Post-PAR Static Timing Report</a>	Out of Date	Sun Sep 14 23:38:14 2014	0	0	3 Infos (0 new)
<a href="#">Bitgen Report</a>	Out of Date	Sun Sep 14 23:39:09 2014	0	1 Warning (1 new)	1 Info (1 new)

Secondary Reports		
Report Name	Status	Generated
<a href="#">ISIM Simulator Log</a>	Out of Date	Sun Sep 14 23:32:13 2014
<a href="#">WebTalk Report</a>	Out of Date	Sun Sep 14 23:39:10 2014
<a href="#">WebTalk Log File</a>	Out of Date	Sun Sep 14 23:39:10 2014

**Date Generated:** 09/14/2014 - 23:40:17

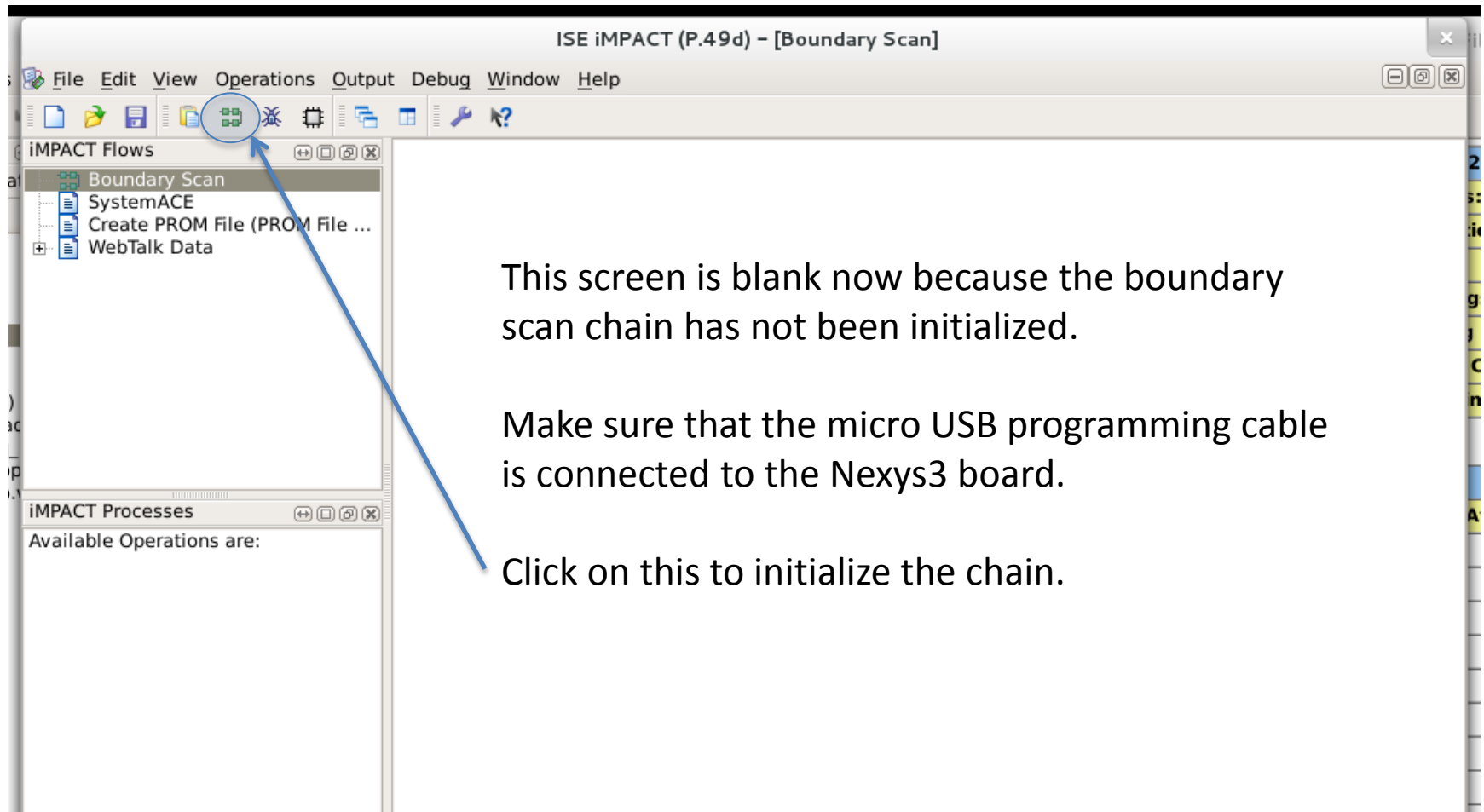
Double click on “Implement Design” and then “Generate Programming File”

# Download Bitstream to FPGA

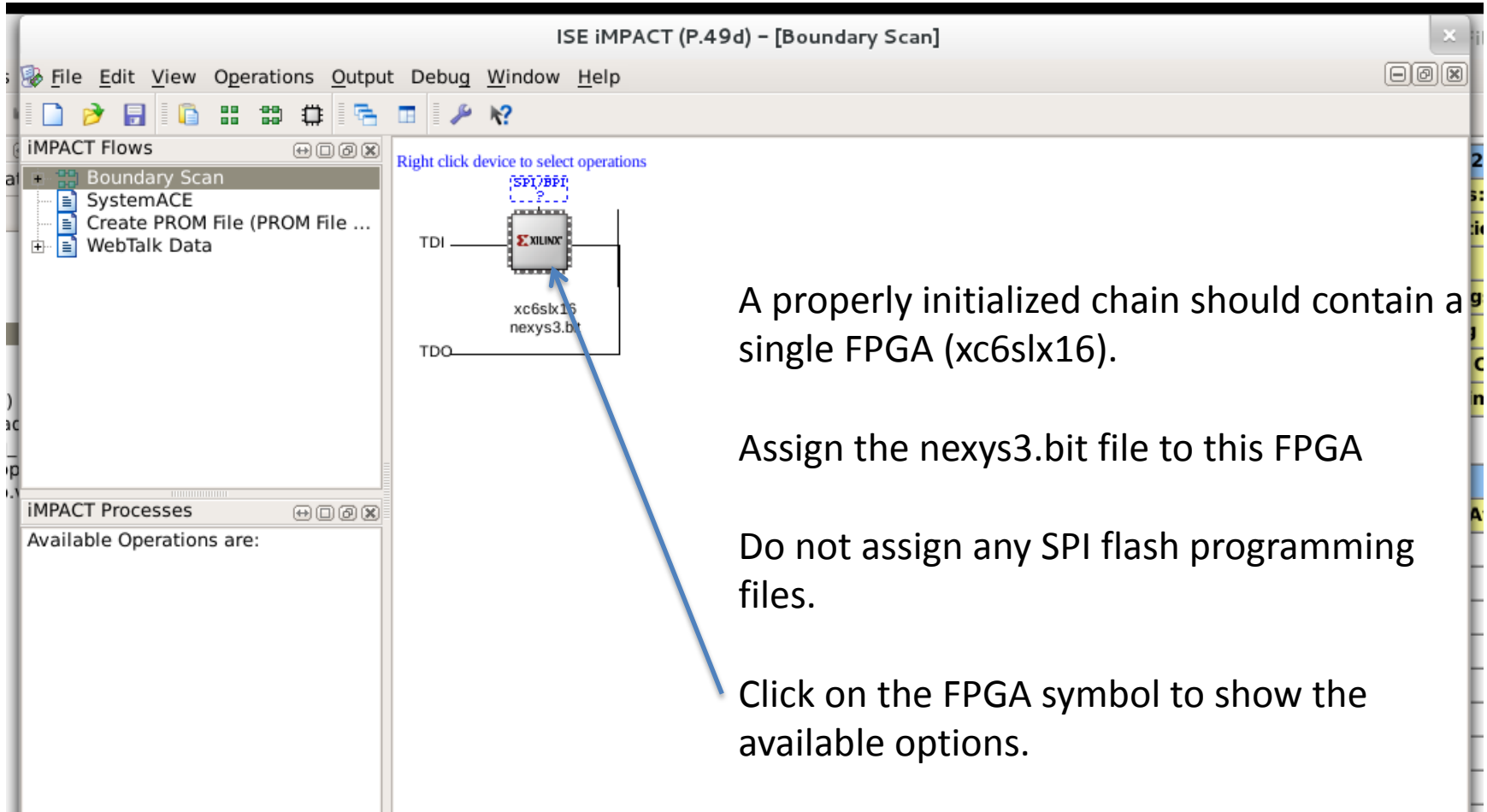
- By now you should have a nexys3.bit file generated.
- You will now program the FPGA using this file.
- Click on “Configure Target Device” to open the Impact program.



# ISE Impact



# 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

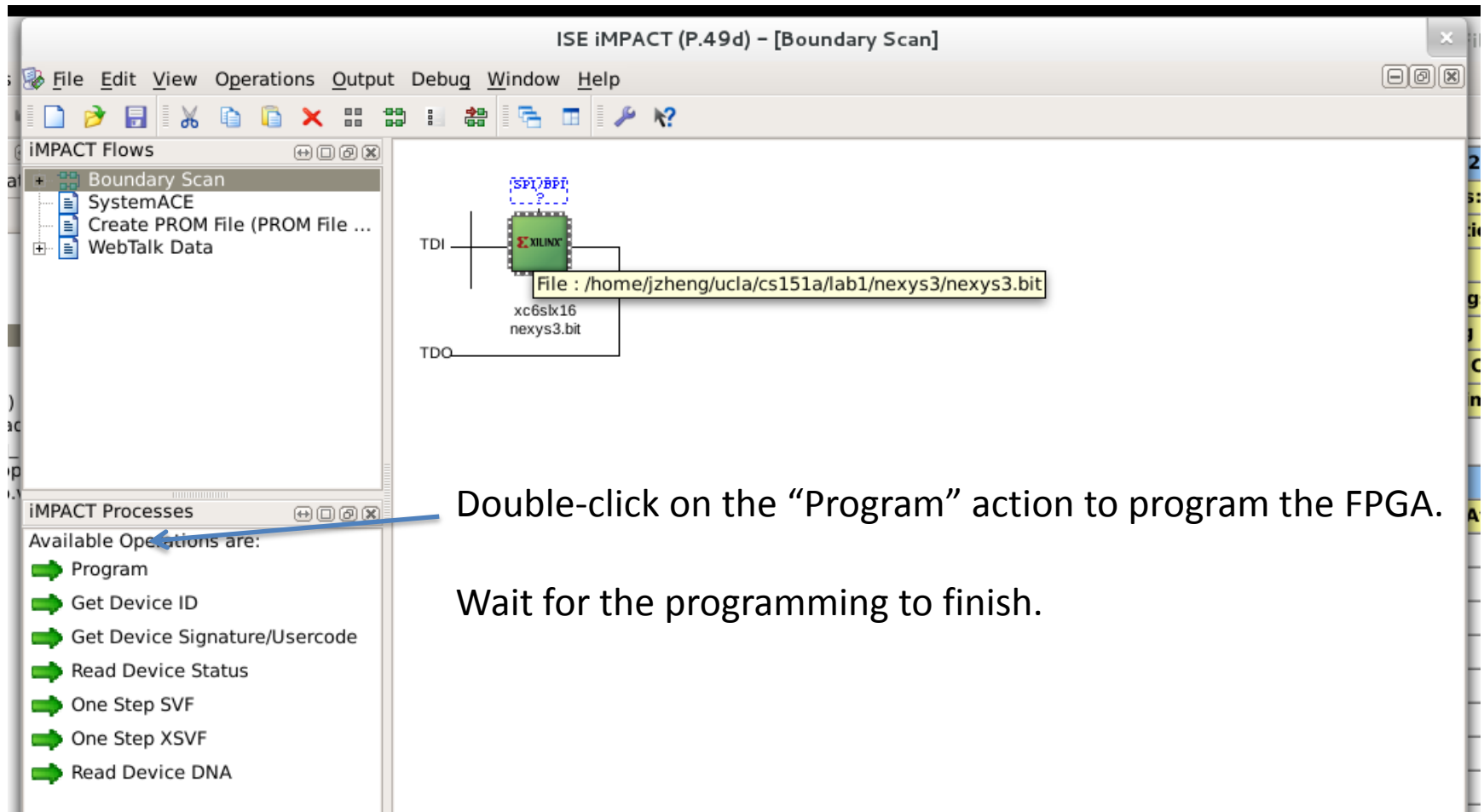
A properly initialized chain should contain a single FPGA (xc6slx16).

Assign the nexys3.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!**

# Set up a serial console

- We now need to set up a serial console to talk to the FPGA's UART
- Use BAUD RATE 1000000, 1-8-1, no parity, no hw flow control
- 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 1000000
  - If you run into permission problems, it's usually because you are not a member of the "dialout" group

# Use the Sequencer

- Translate the “program” from slide 24 into binary instructions
- Enter these instructions using the slider switches and the btnS button
- Watch the serial console window. Does the output match your expectation?
- Does the LEDs reflect the instruction count?

# Exercise

- Write a program to print out the first 10 numbers of the Fibonacci series.
- Try to understand the project's source files as much as you can, including the test benches.
- For bonus, read the Nexys3 user manual regarding the 7-segment LED display, and design a hardware controller that controls the display.

# Conclusion

- If you are having trouble following the steps, take a deep breath. Read the slides again, and make sure you understand the instructions.
- If you don't know what each of the step is for, ask your TA or classmates.
- If you are enjoying this lab so far, you are in a good shape to pass this class!