PART I: Theory

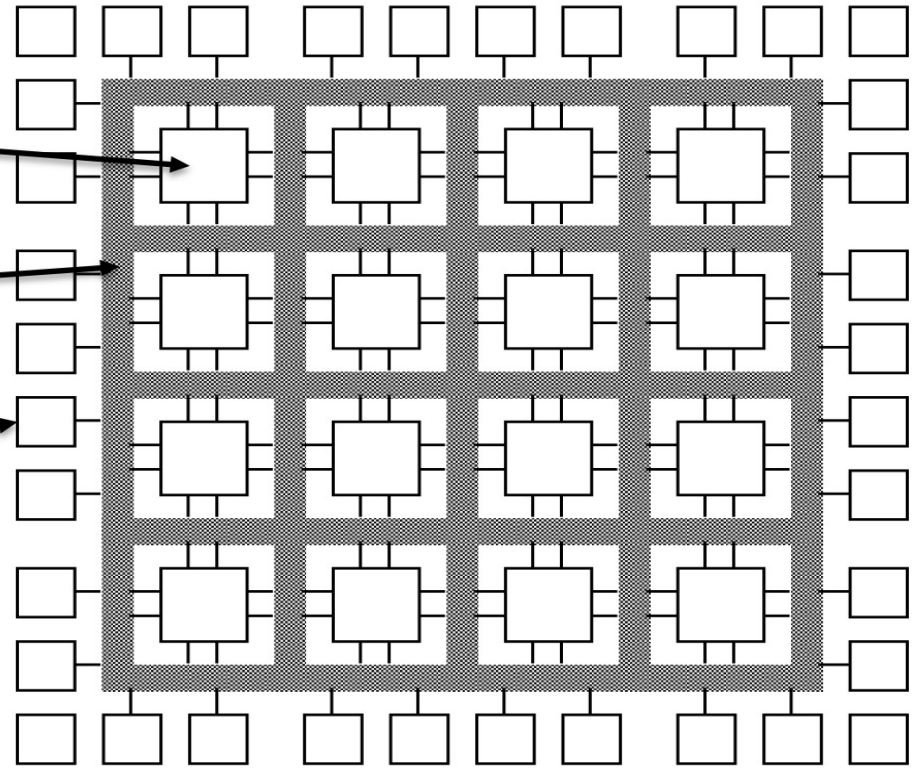
WHAT IS AN FPGA?

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.

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 resolved, a logic synthesis tool, such as Vivado, analyzes the design and produces a netlist that includes commonly used cells compatible with the FPGA target.

– The netlist should be functionally equivalent to the original source code.

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.

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.

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.

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.

– Now you have a “compiled” FPGA design.

PART II: Practice

Installation

- For the installation Guide please visit this link :

https://digilent.com/reference/programmable-logic/guides/installing-vivado-and-vitis

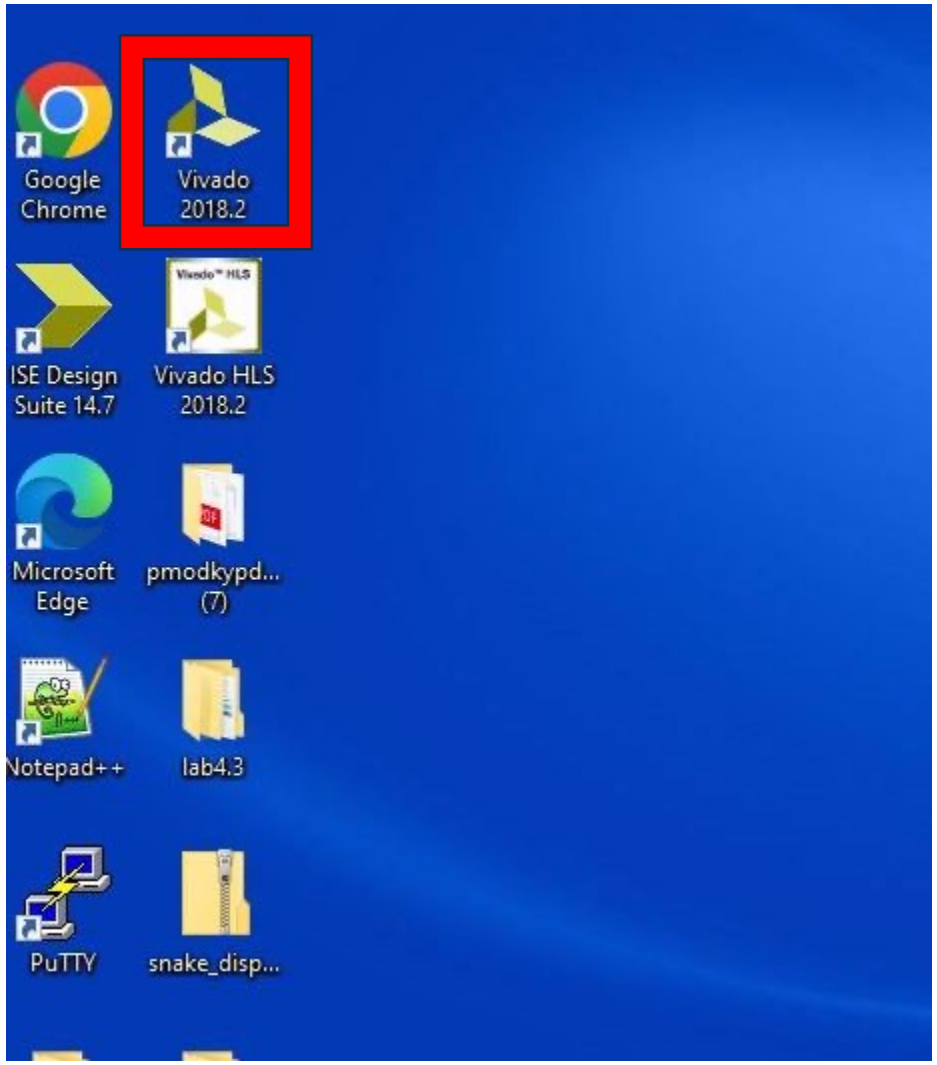
- For mac, you need a Virtual machine

- Please Download 2018.2 version :

https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/archive.html

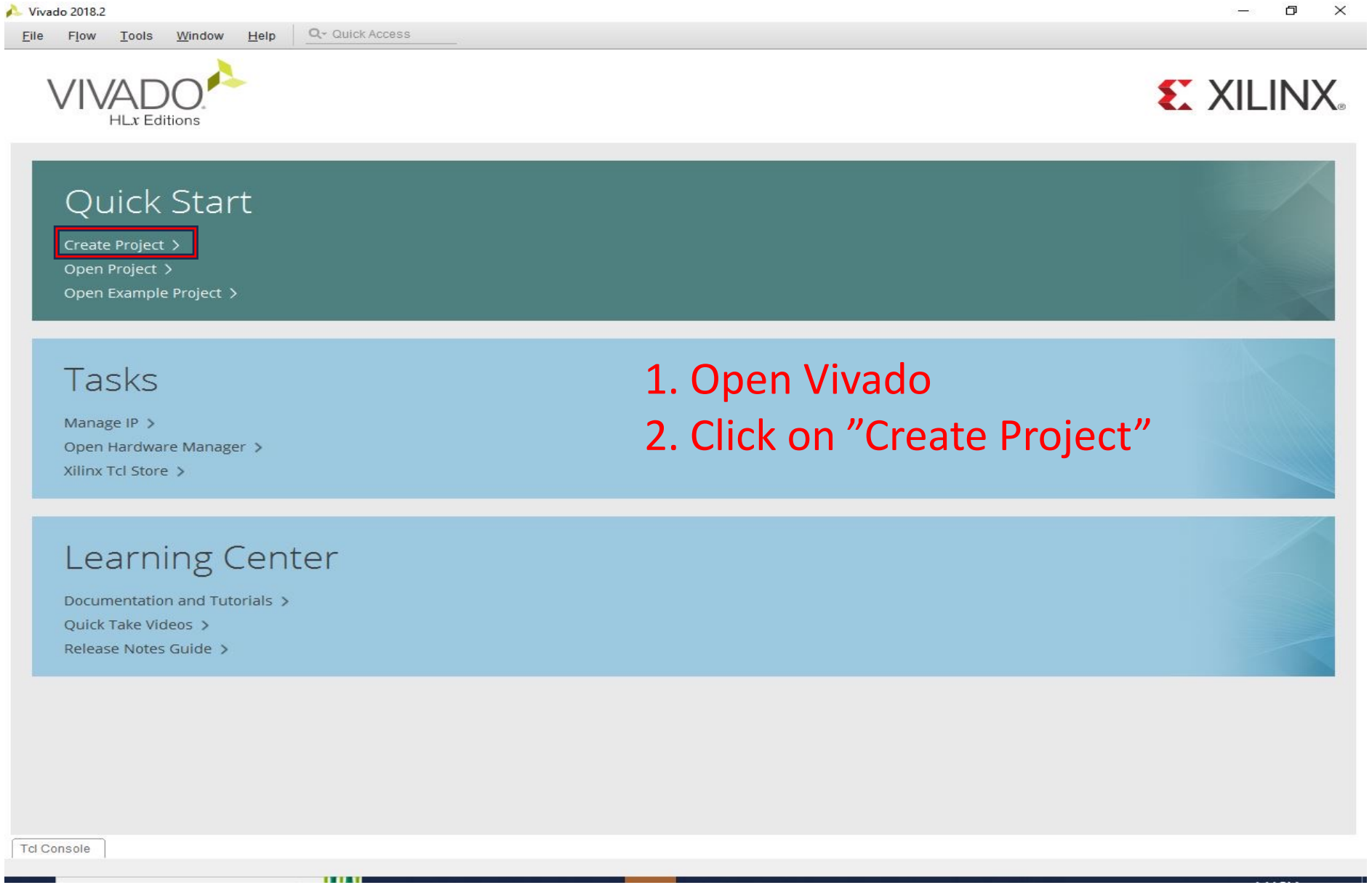
EXAMPLE PROJECT IMPLEMENTATION :

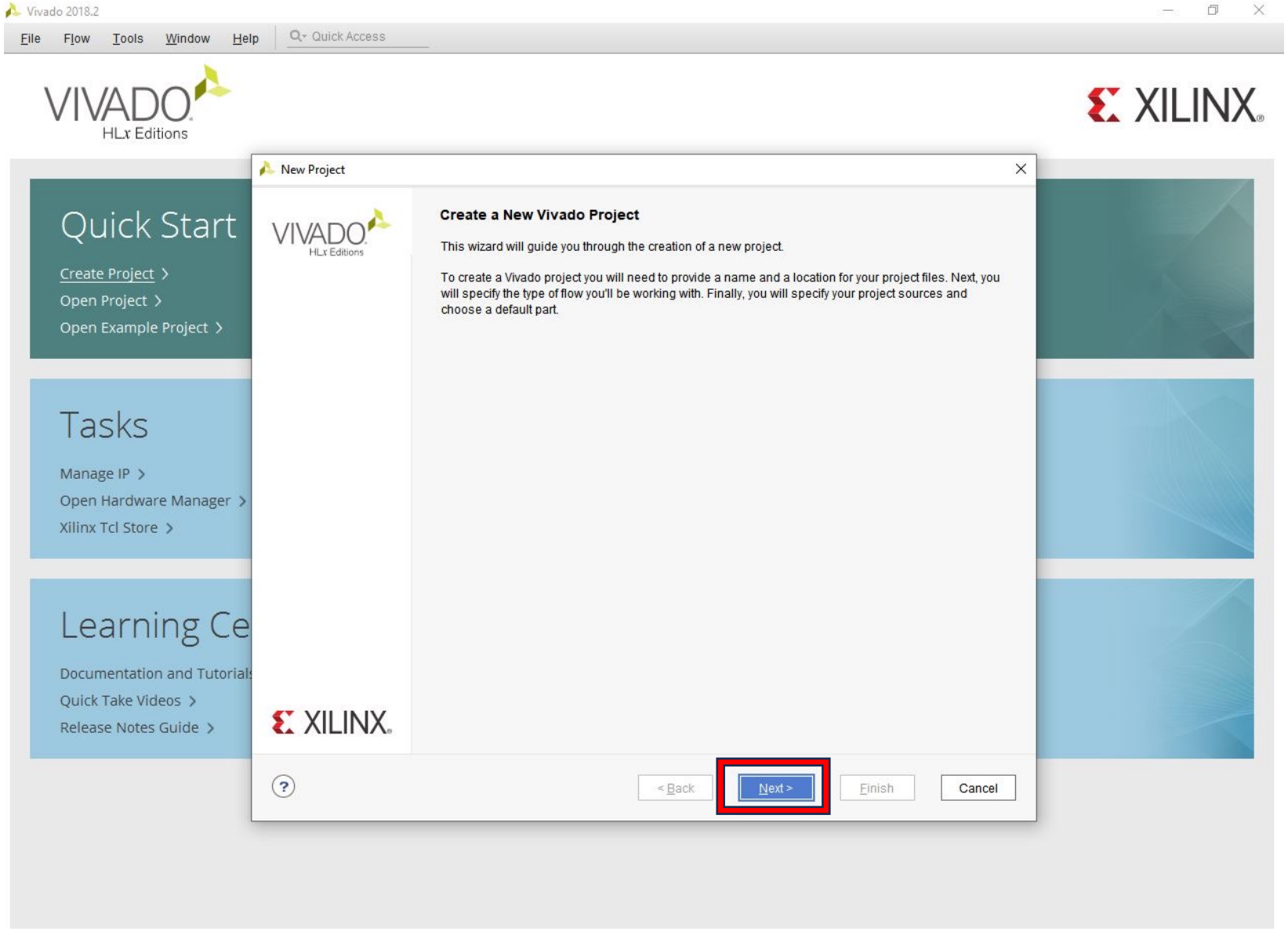
Open the Xilinx Vivado



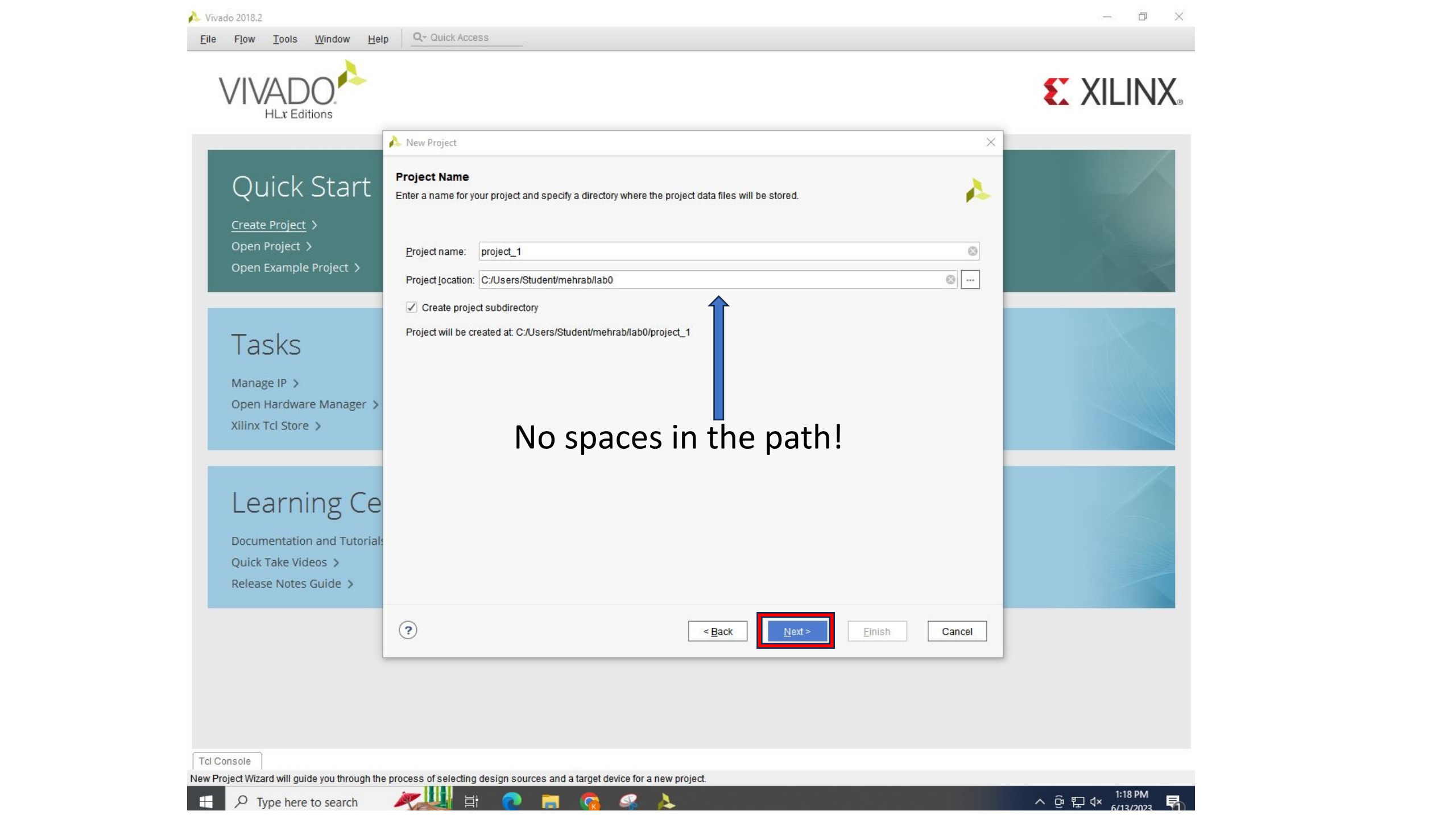
Click the Vivado 2018.2 icon

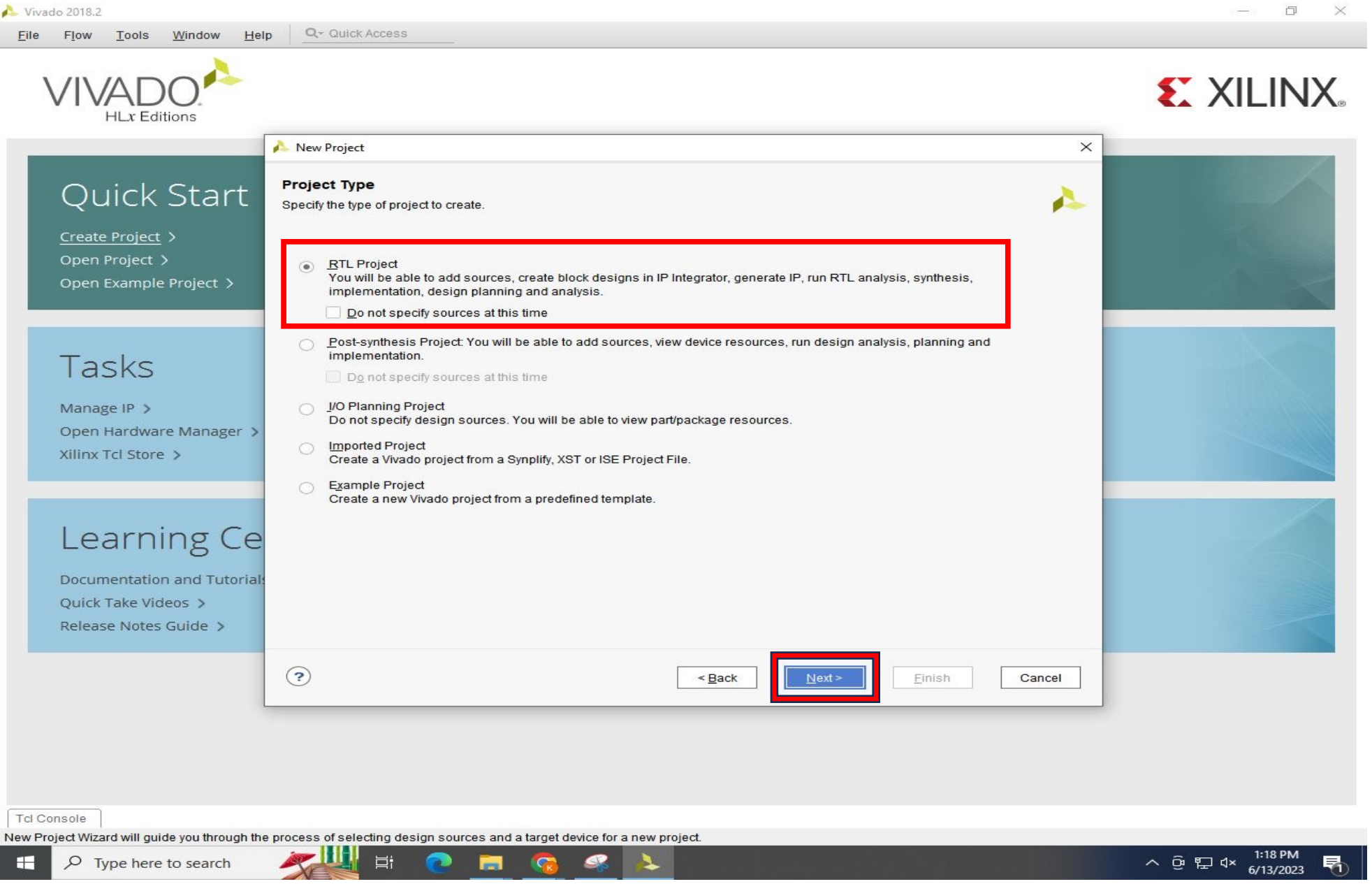
Create a VIVADO Project



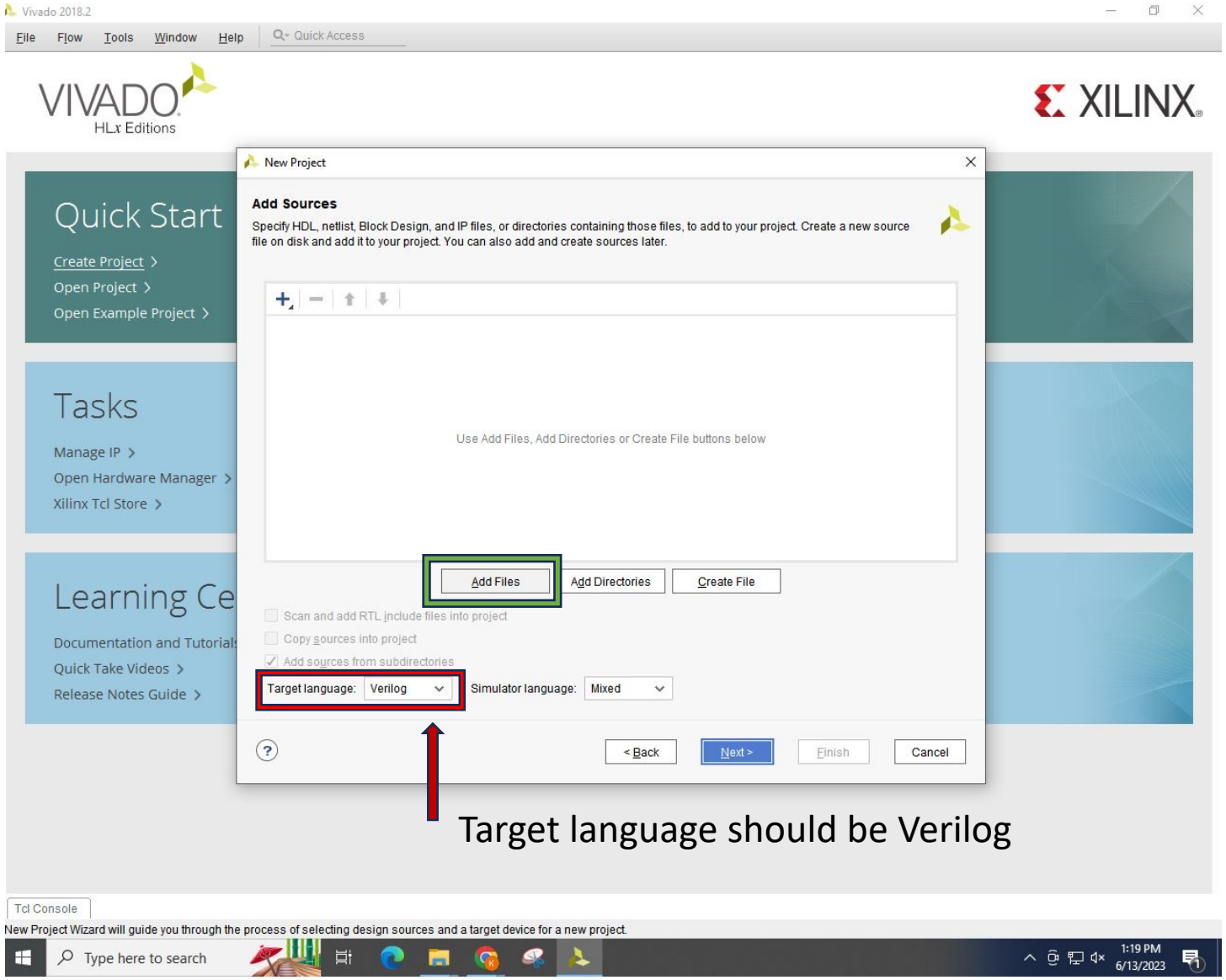

Vivado 2018.2
File
Flow
Tools
Window
Help
Quick Access
VIVADO
HLx Editions
XILINX
Quick Start
Create Project
Open Project
Open Example Project
Tasks
Manage IP
Open Hardware Manager
Xilinx Tcl Store
Learning Ce
Documentation and Tutorials
Quick Take Videos
Release Notes Guide
New Project
VIVADO
HLx Editions
Create a New Vivado Project
This wizard will guide you through the creation of a new project.
To create a Vivado project you will need to provide a name and a location for your project files. Next, you will specify the type of flow you'll be working with. Finally, you will specify your project sources and choose a default part.
XILINX
< Back
Next >
Finish
Cancel

Click Next


Vivado 2018.2
File Flow Tools Window Help Quick Access
VIVADO HLx Editions
XILINX
Quick Start
Create Project
Open Project
Open Example Project
Tasks
Manage IP
Open Hardware Manager
Xilinx Tcl Store
Learning Ce
Documentation and Tutorials
Quick Take Videos
Release Notes Guide
New Project
Project Name
Enter a name for your project and specify a directory where the project data files will be stored.
Project name: project_1
Project location: C:/Users/Student/mehrab/lab0
Create project subdirectory
Project will be created at: C:/Users/Student/mehrab/lab0/project_1
No spaces in the path!
< Back
Next >
Finish
Cancel
Tcl Console
New Project Wizard will guide you through the process of selecting design sources and a target device for a new project.
Type here to search
1:18 PM
6/13/2023



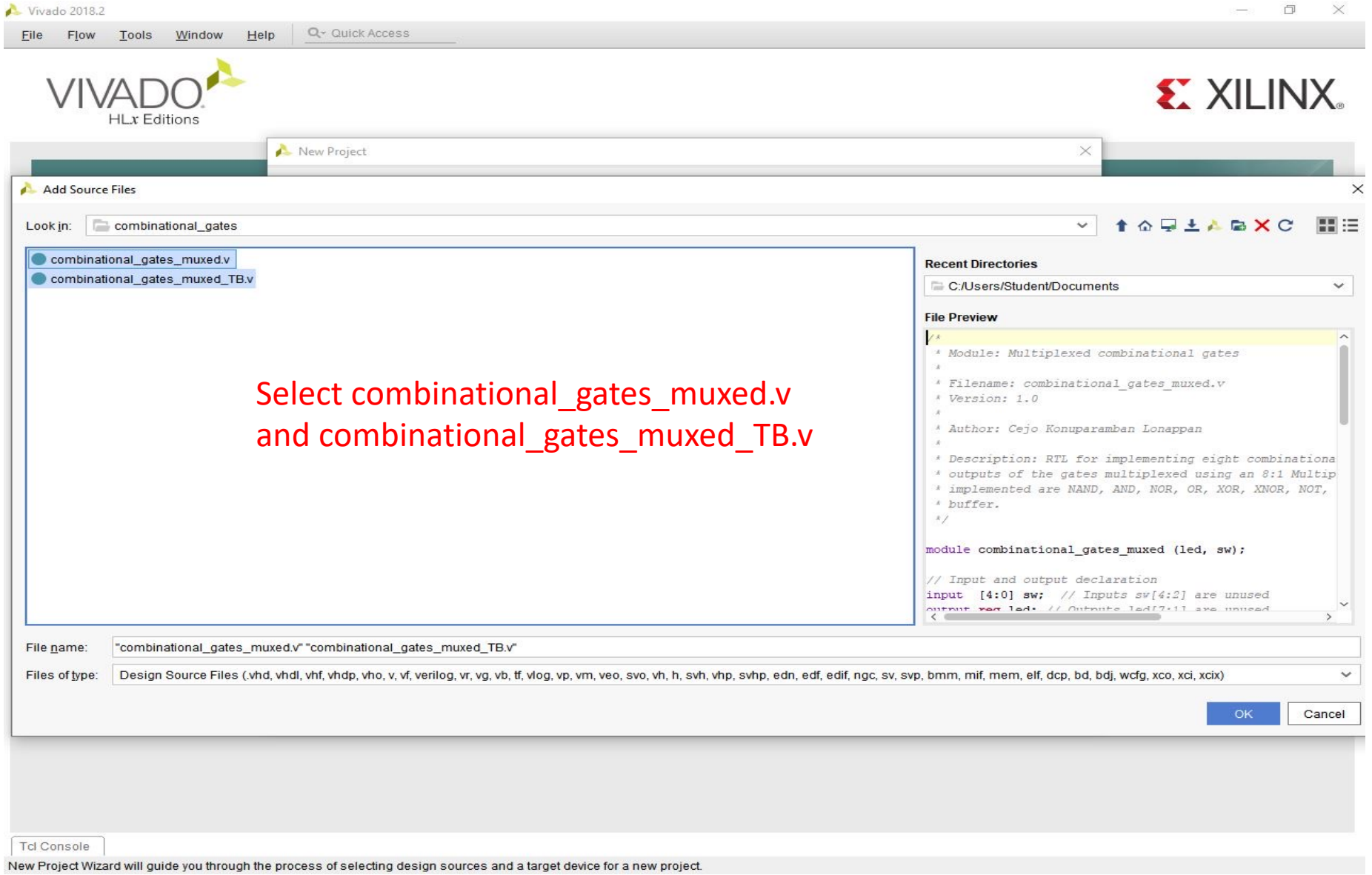
Select RTL Project and click **Next**



In this window, you can select source files or directories that you'll want to use in your projects. We can also select which language we'll be programming in.

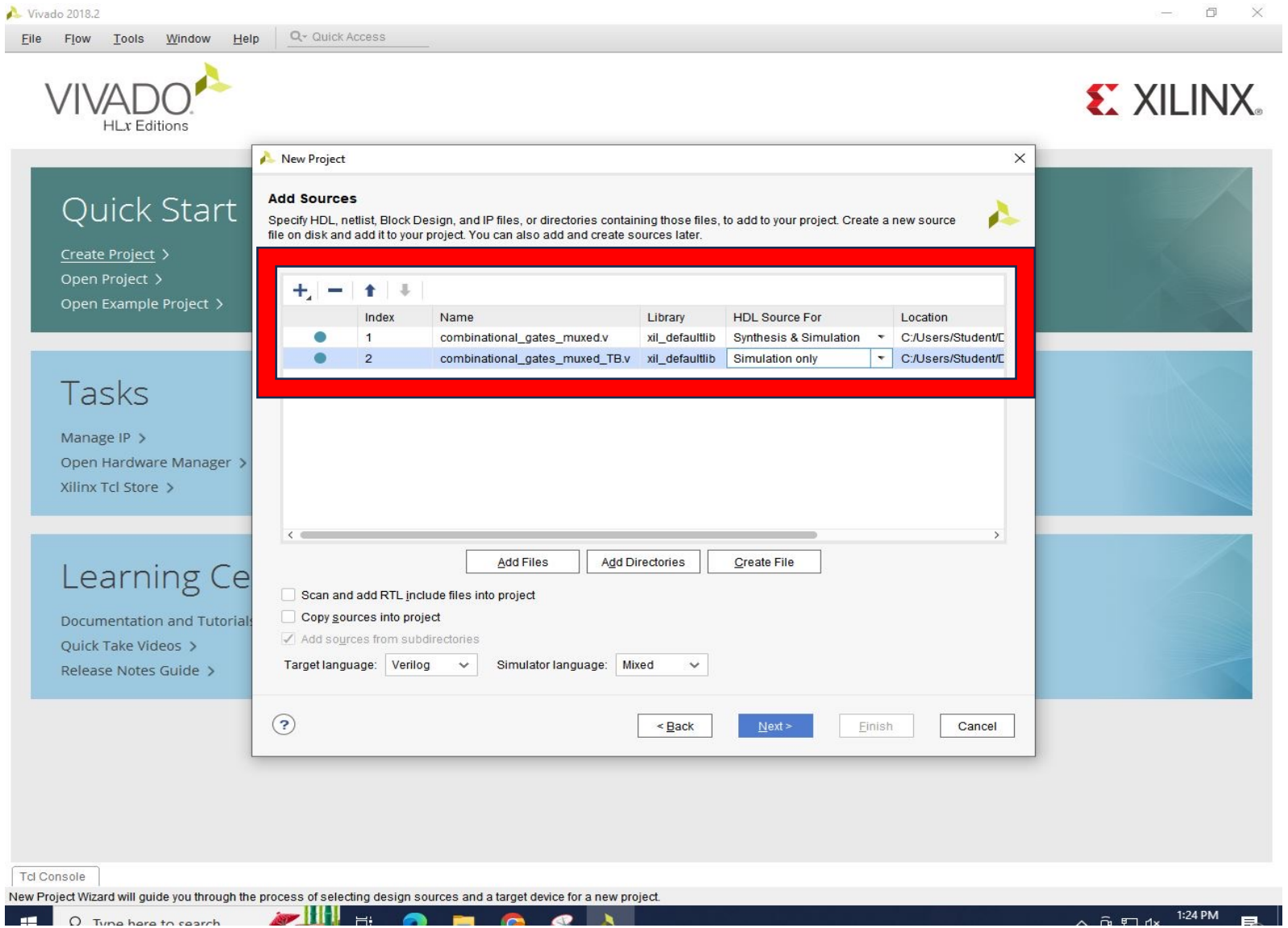
Click on Add Files

Add Source Files



Here you can add your structural code and your Testbench Code

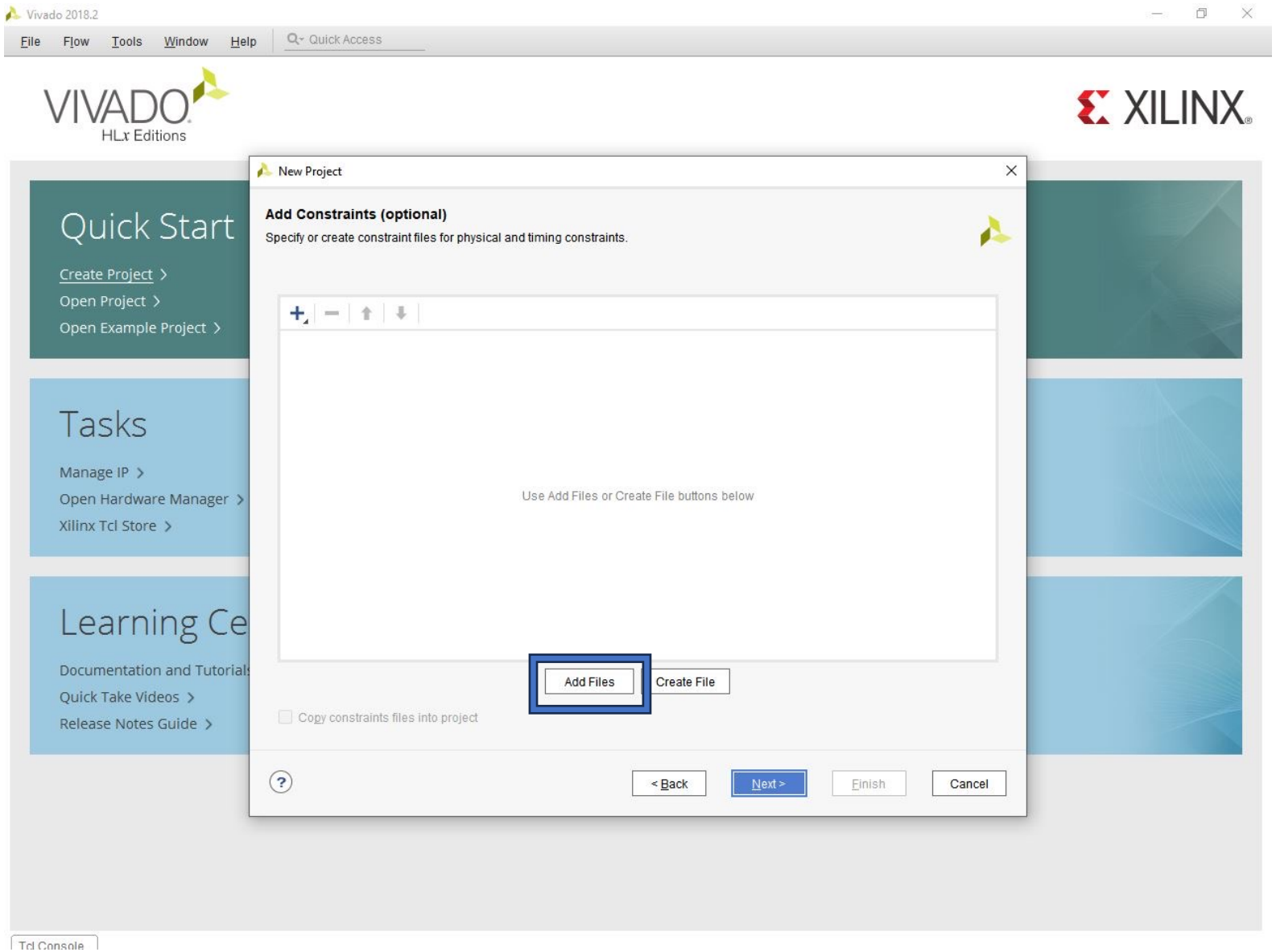
Files Association



Make sure the file association is correct

Then click Next

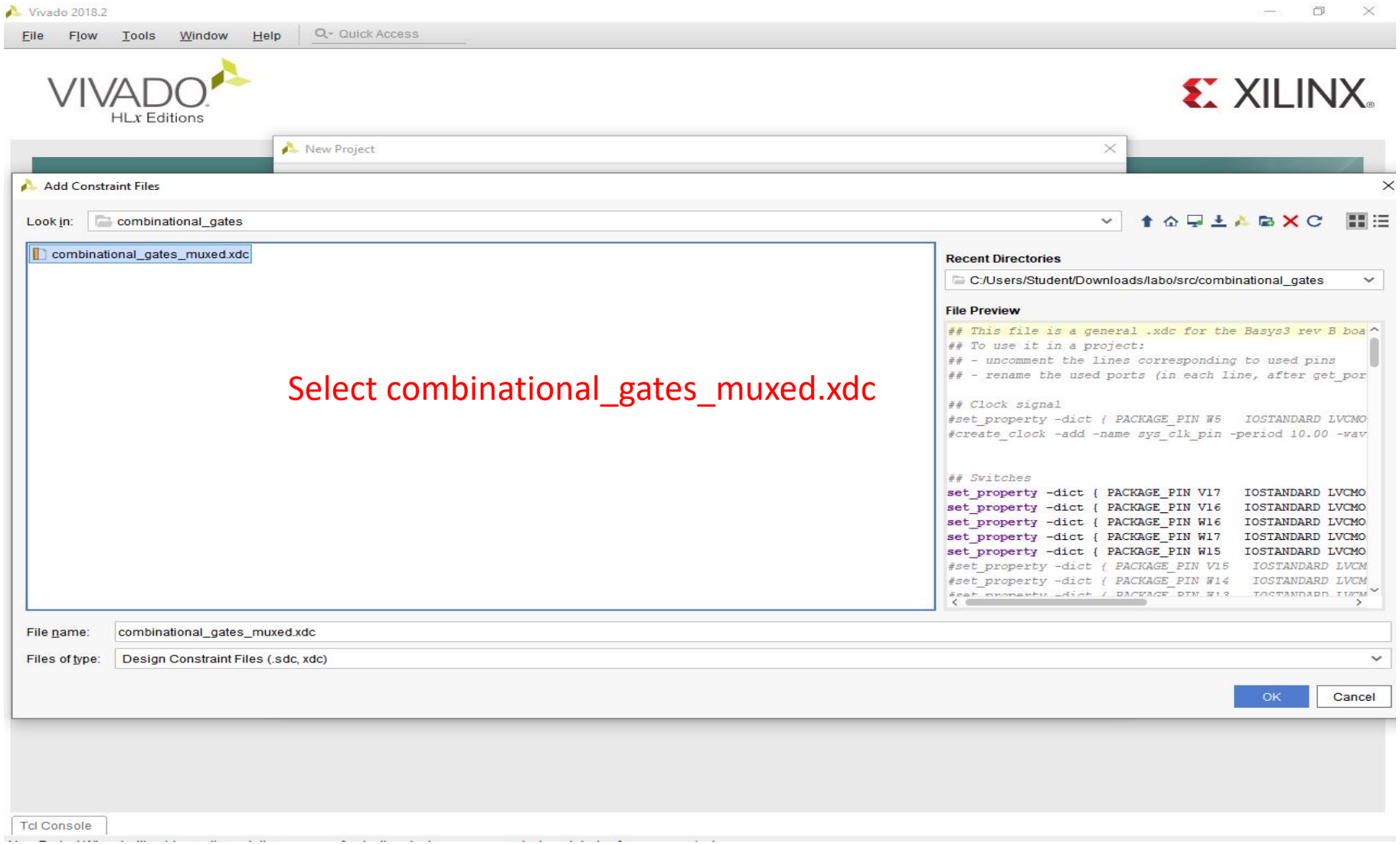
Add Constraints



Here you can add Xillinx Design Constraint file

Click on Add files to add constraint files

Constraint file(.xdc)

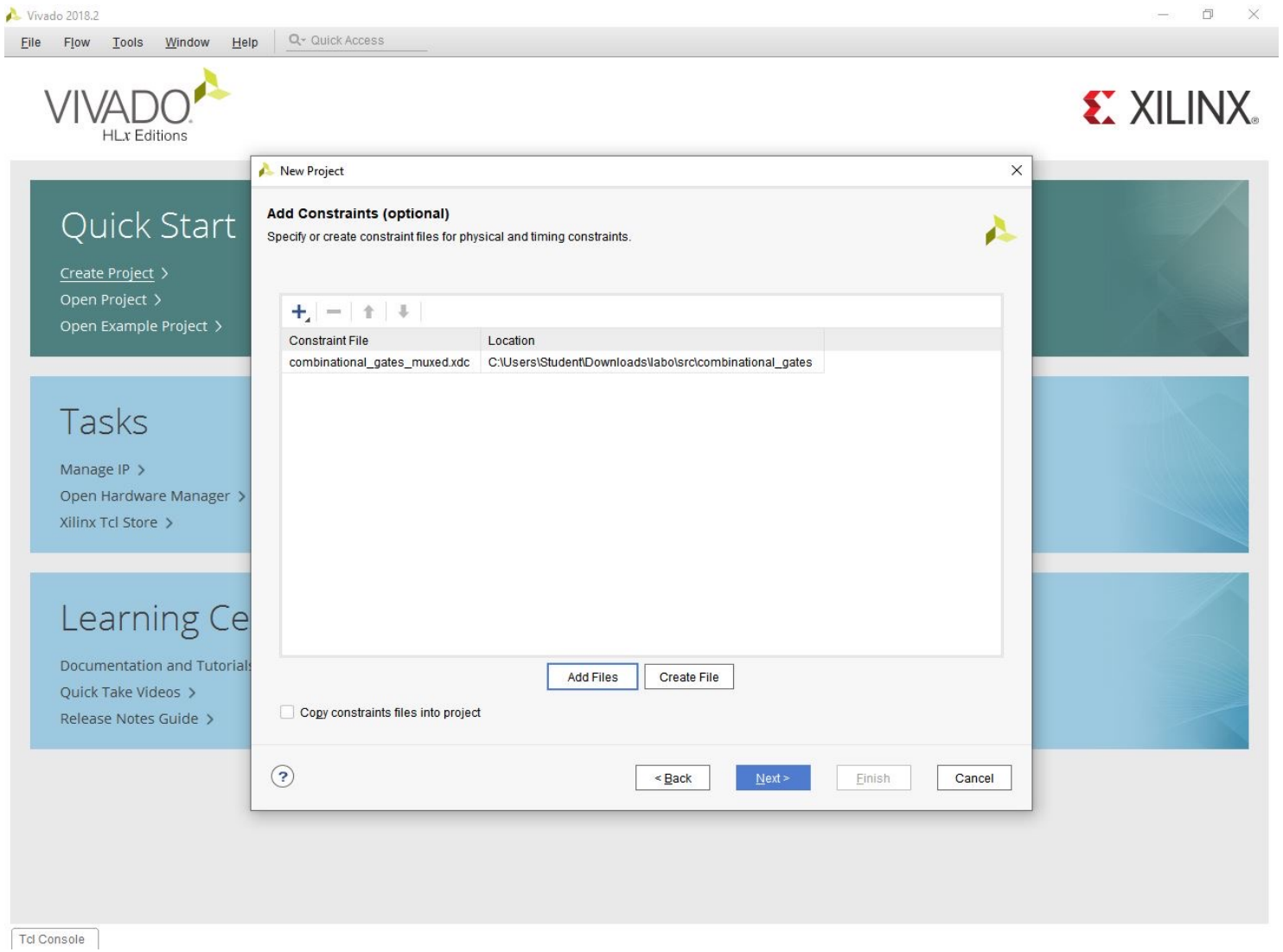


XDC lists all the available pin mappings in the FPGA in the following format:

#set_property -dict { PIN Location IOSTANDARD LVCMOS33 } [get_ports {“your_signal_name”}]

For example:

#set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports {sw[0]}]

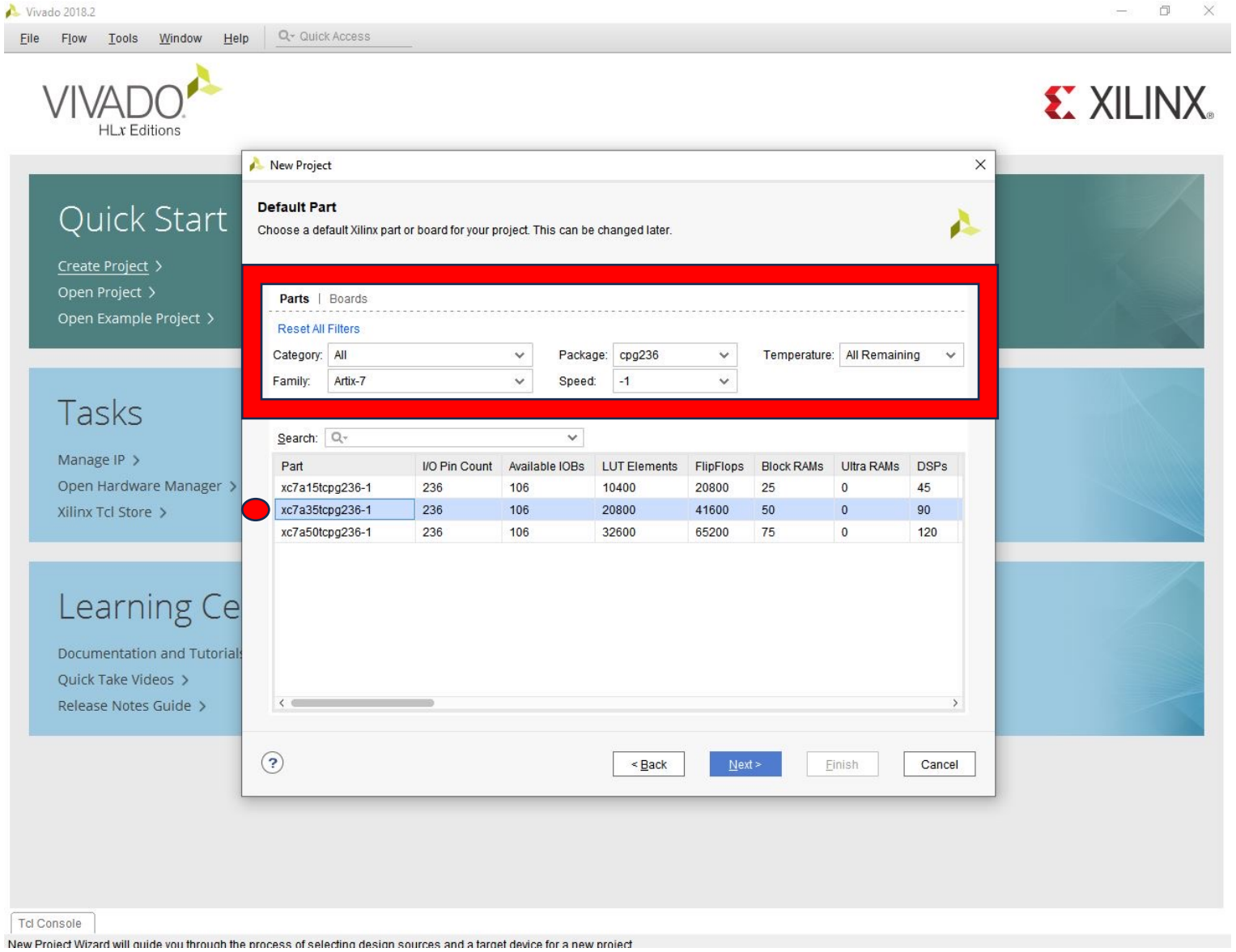

Vivado 2018.2
File
Flow
Tools
Window
Help
Quick Access
VIVADO
HLx Editions
XILINX
Quick Start
Create Project
Open Project
Open Example Project
Tasks
Manage IP
Open Hardware Manager
Xilinx Tcl Store
Learning Ce
Documentation and Tutorials
Quick Take Videos
Release Notes Guide
New Project
Add Constraints (optional)
Specify or create constraint files for physical and timing constraints.
Constraint File
Location
combinational_gates_muxed.xdc
C:\Users\Student\Downloads\labo\src\combinational_gates
Add Files
Create File
Copy constraints files into project
< Back
Next >
Finish
Cancel
Tcl Console

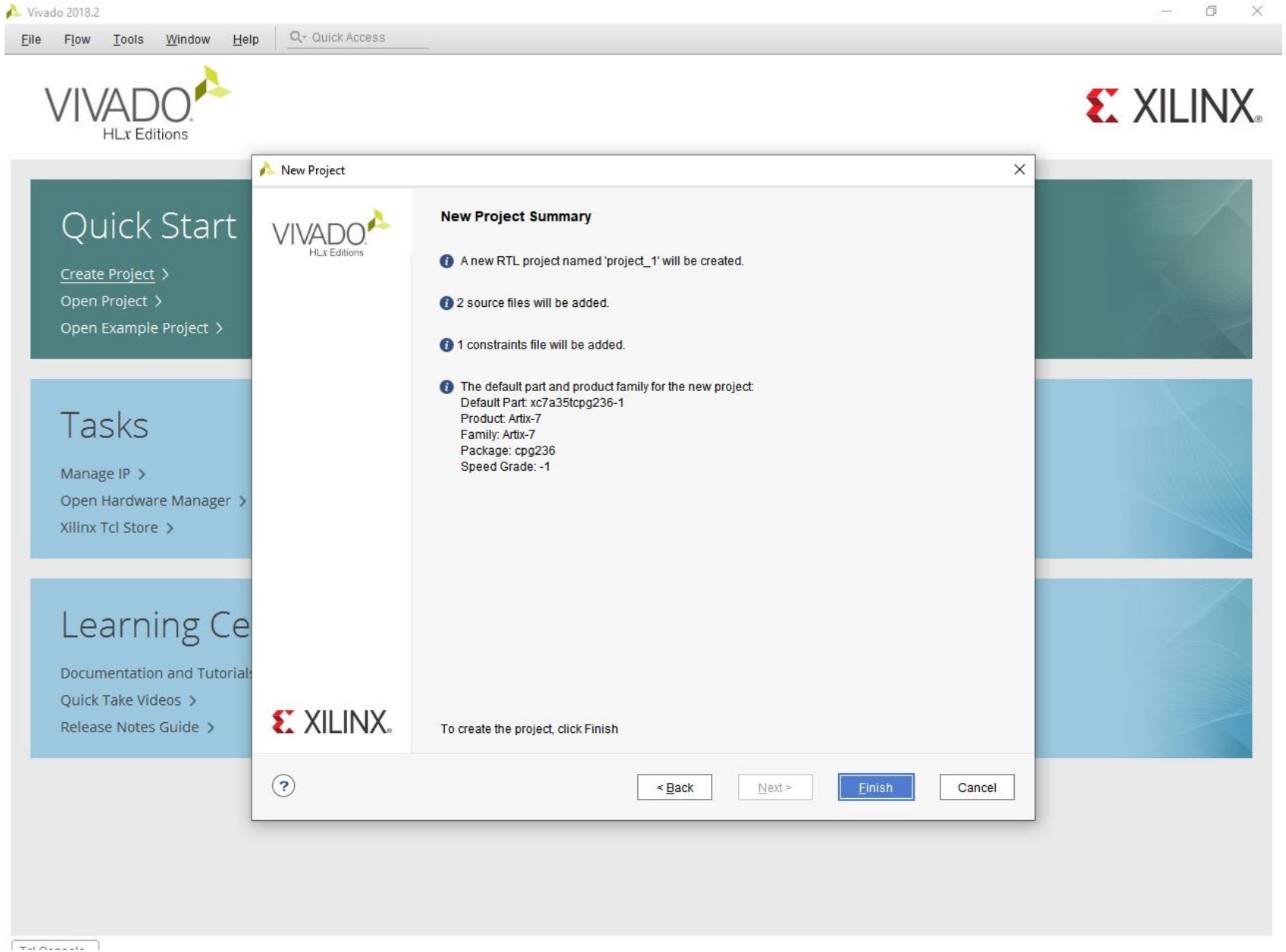
Click Next

Device Properties

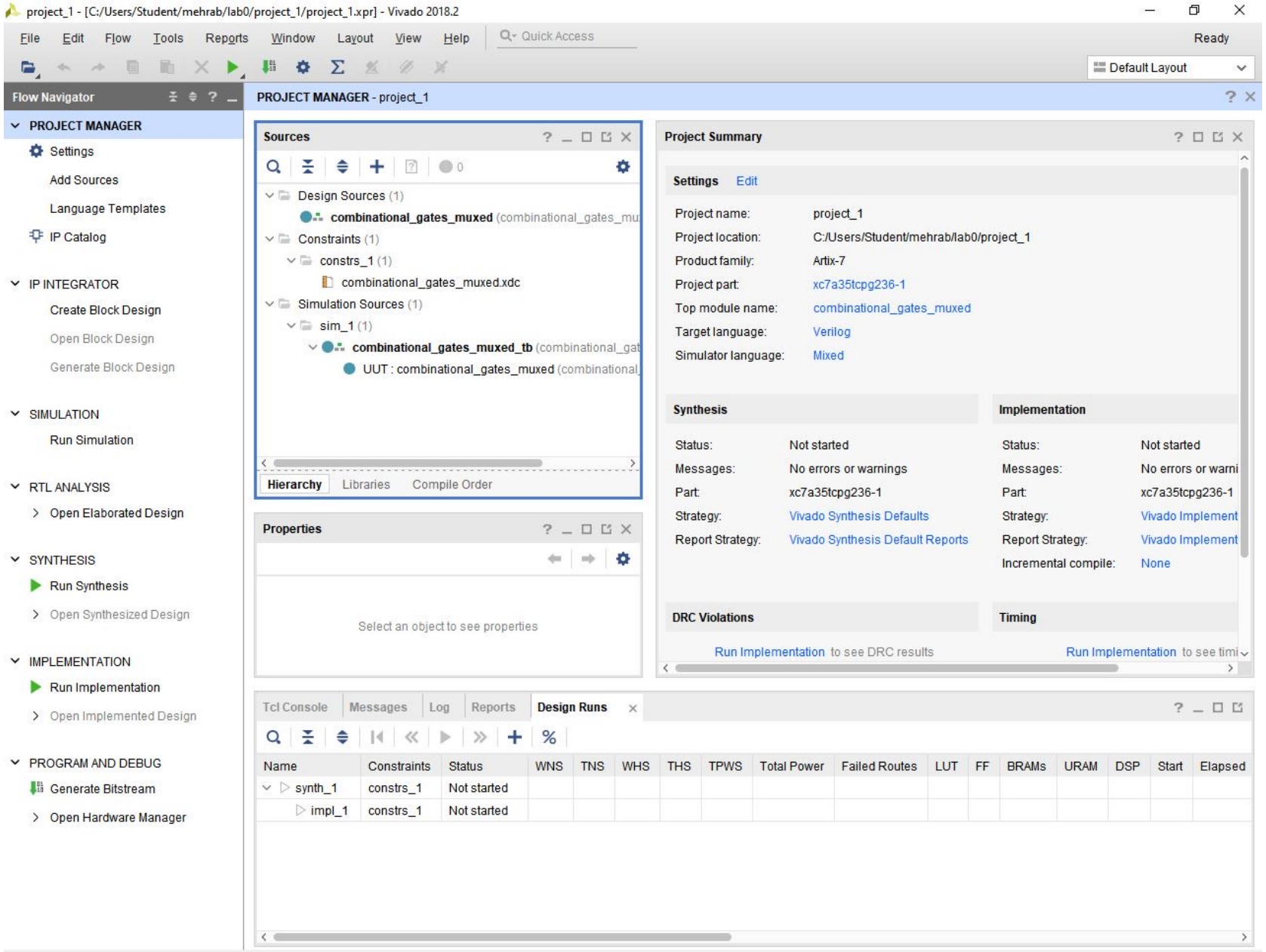
Make sure the fields match what you see here

Select xc7a35tcp236-1 and click Next



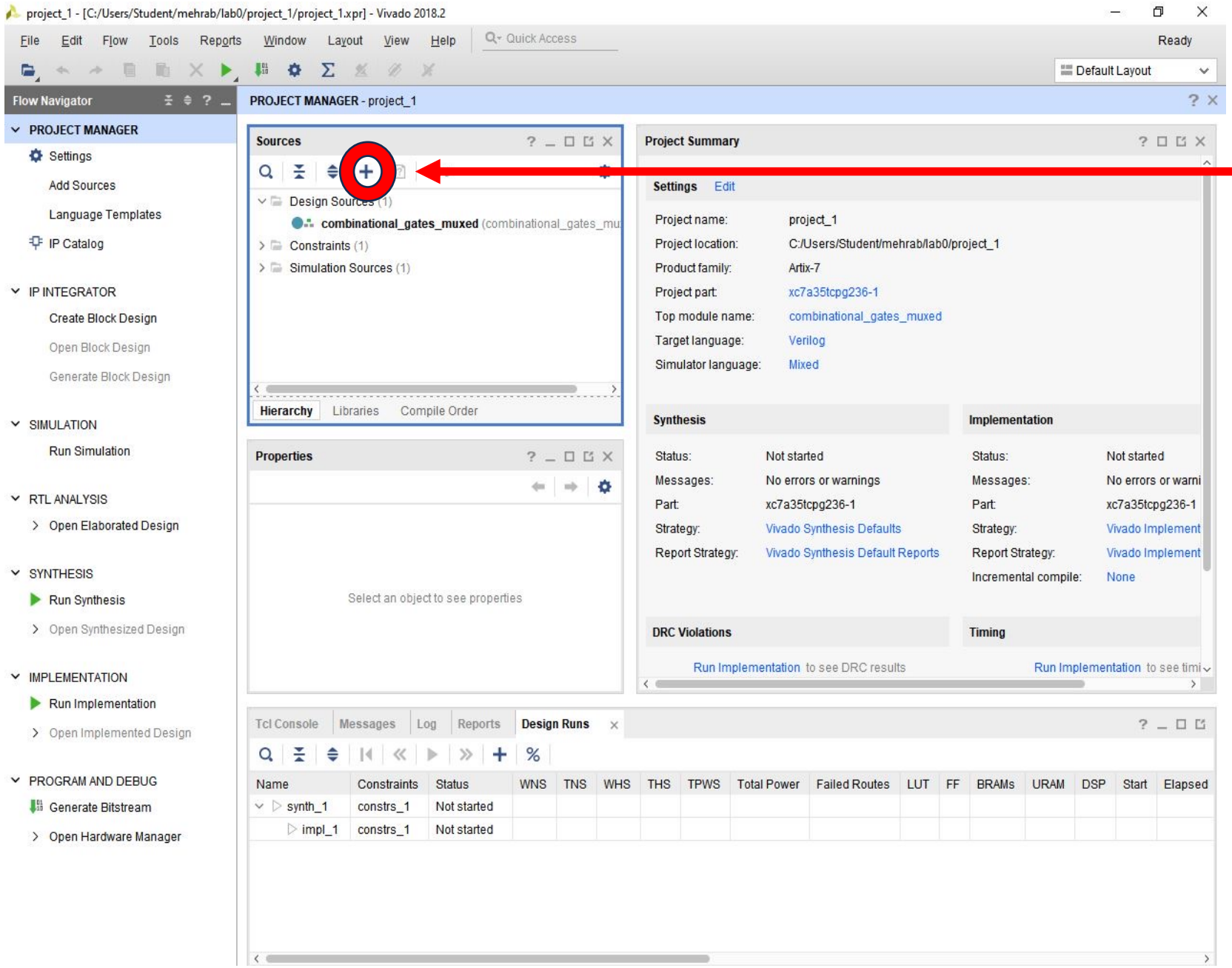


Click Finish to build the project



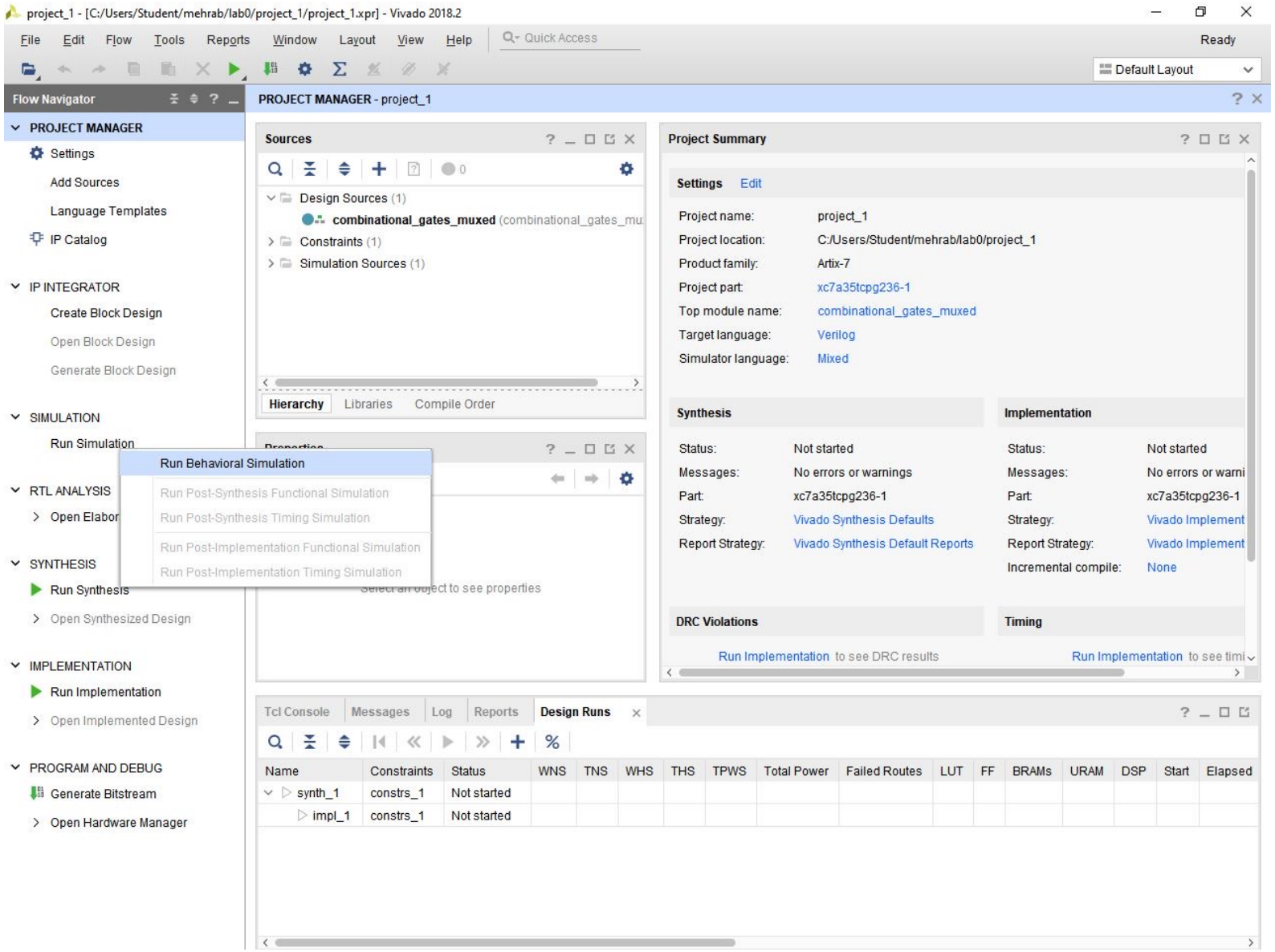
You should be able to see this view.

Make sure the Hierarchy of source files are match what you see here



You could add source files using "+"

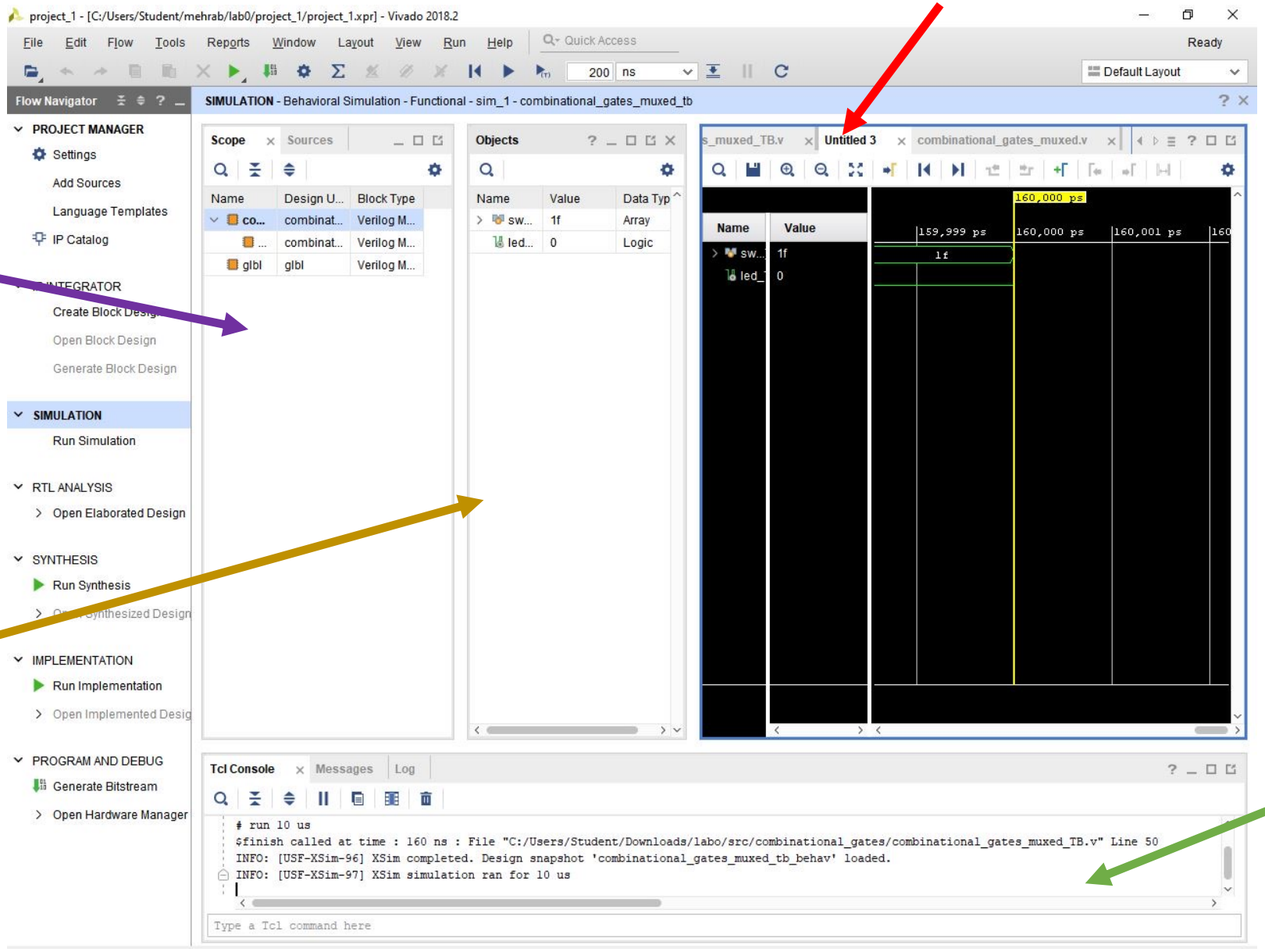
Ready for simulation



Click on Run Simulation
Then click on Run Behavioral Simulation

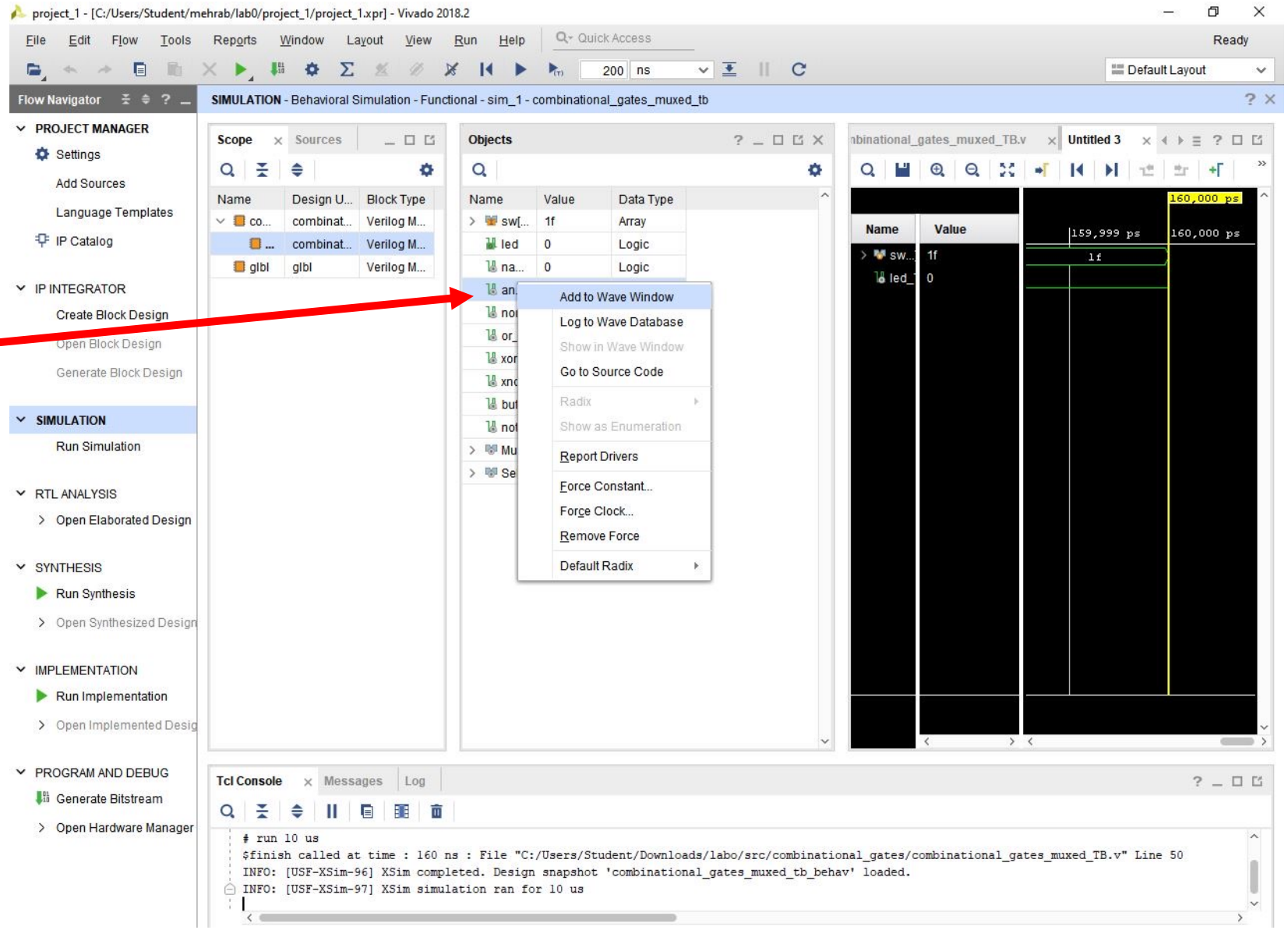
- Simulation will be launched
- In the simulation environment you can dynamically debug the circuit, much like a software debugger
- Your main focus should be on the console window and the waveform window

Simulation Main Window

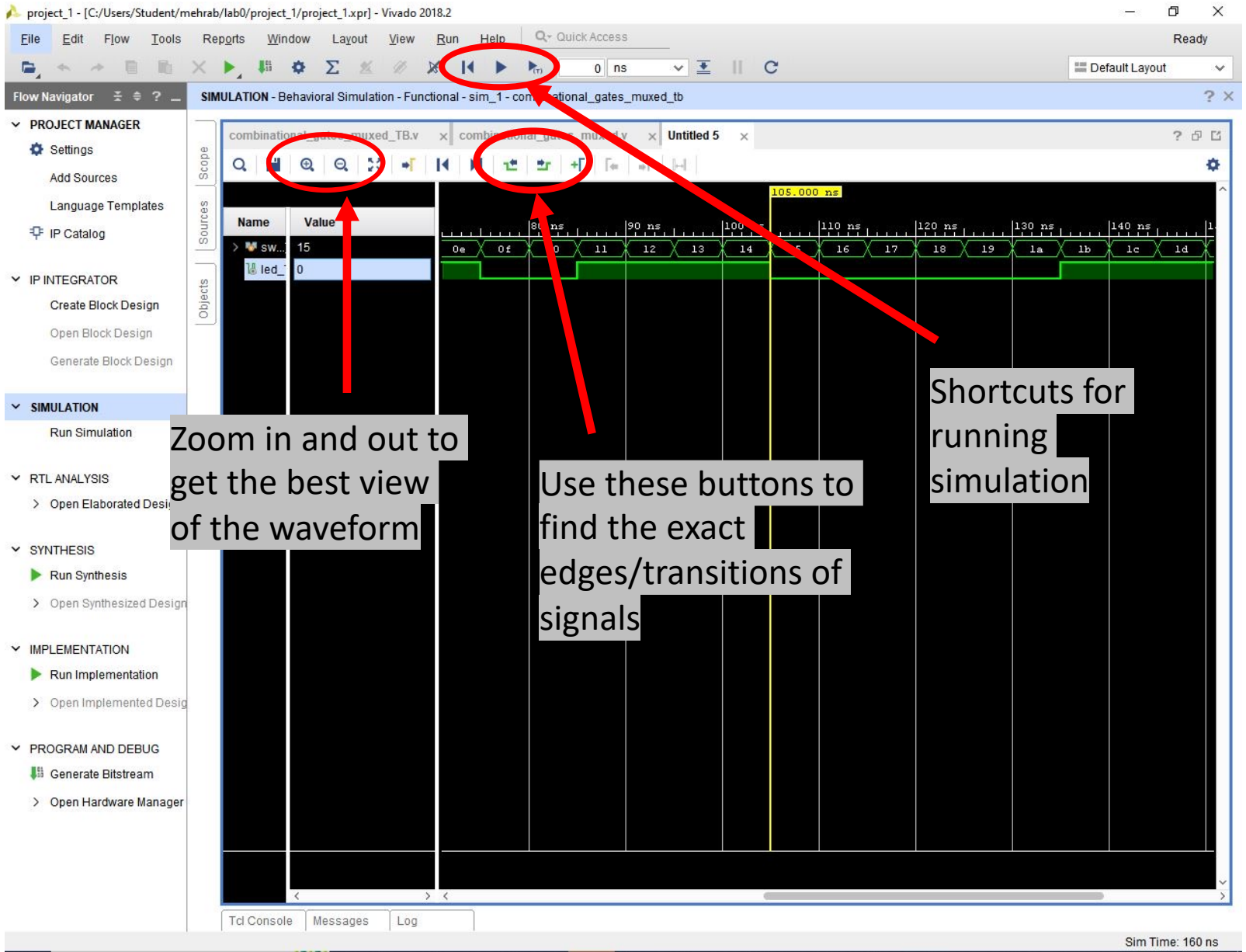


Post Simulation Examination

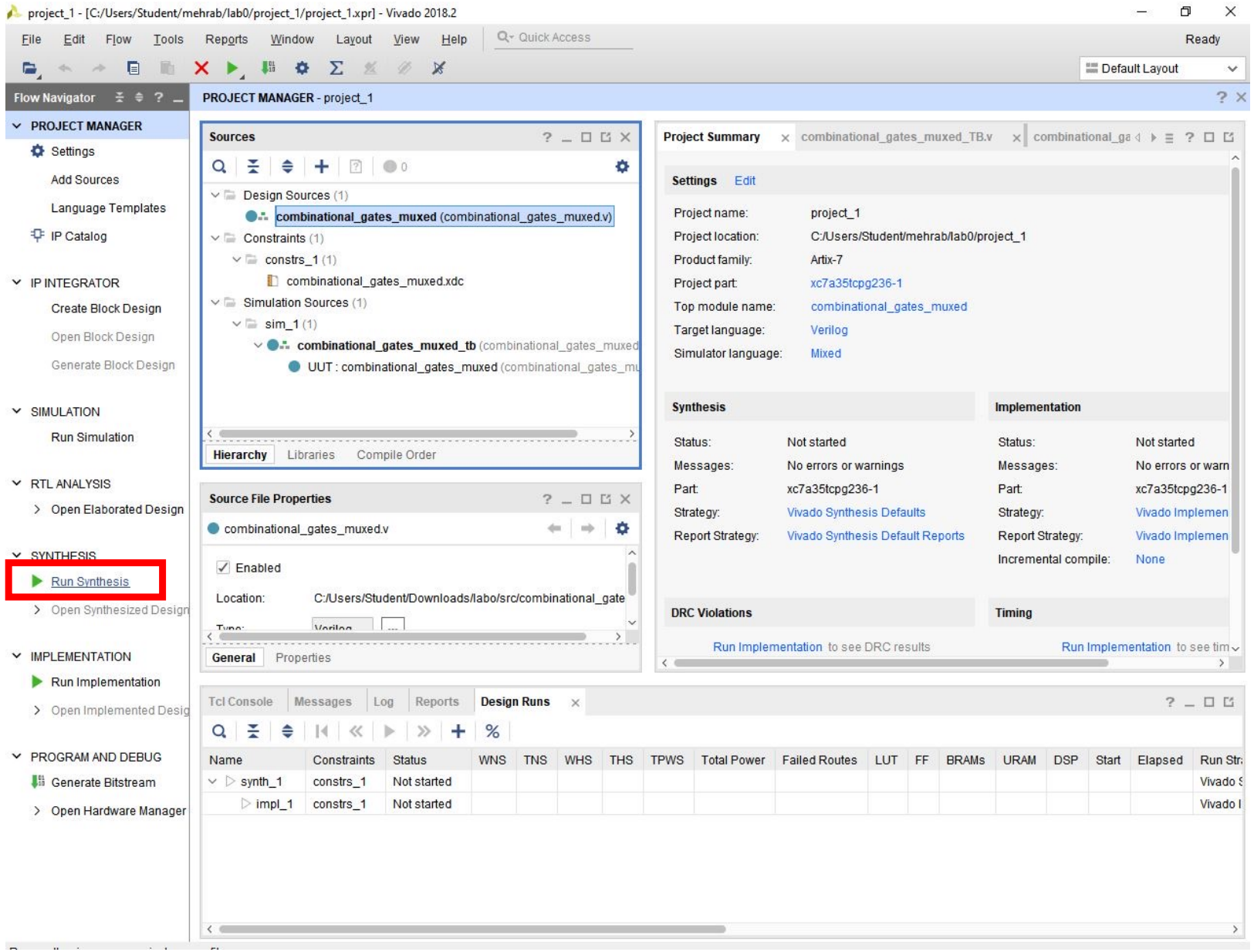
To debug, you can add any signals to the waveform view by right click and select add



Post Simulation Examination

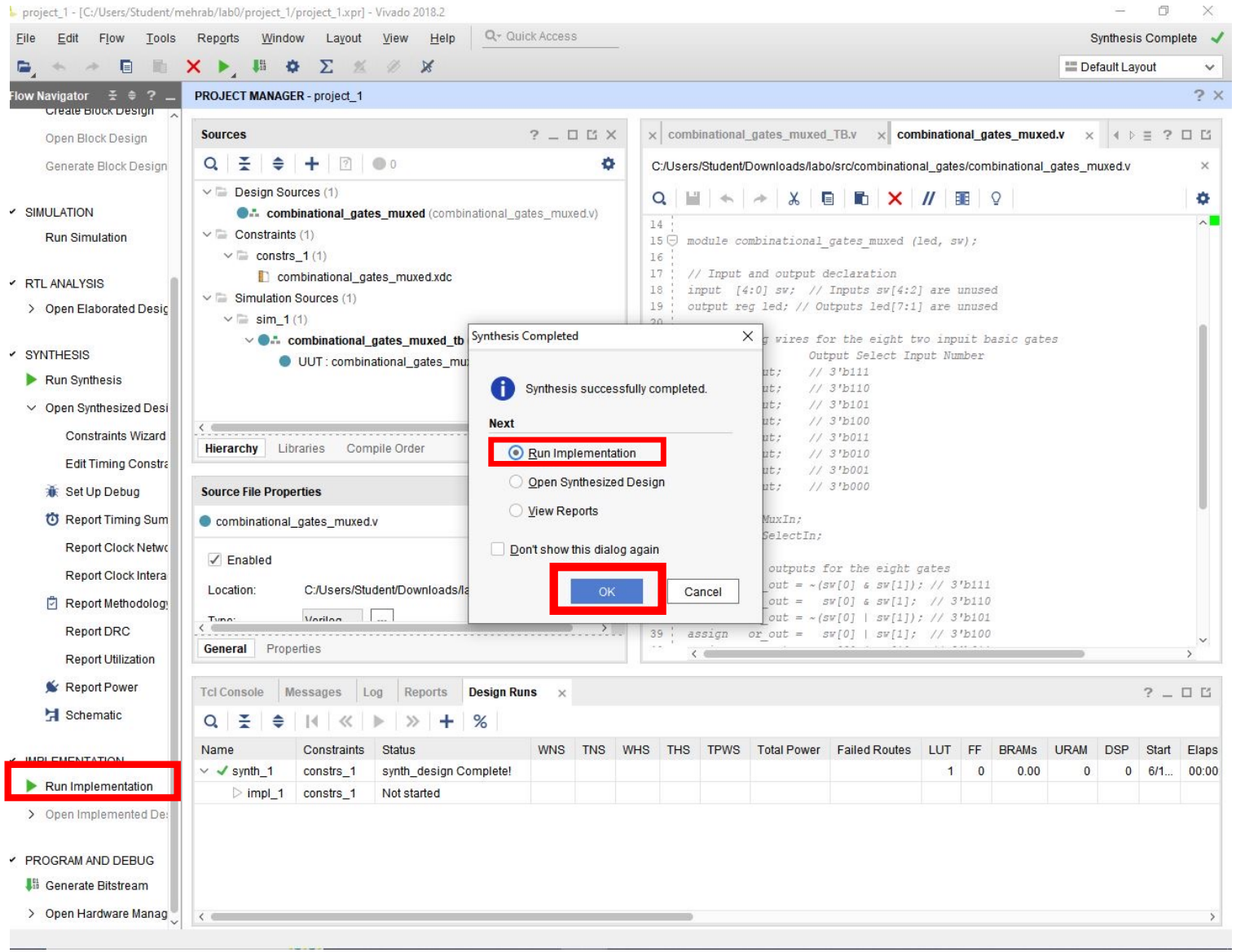


Ready for Synthesis



Click on Run Synthesis

Implementation

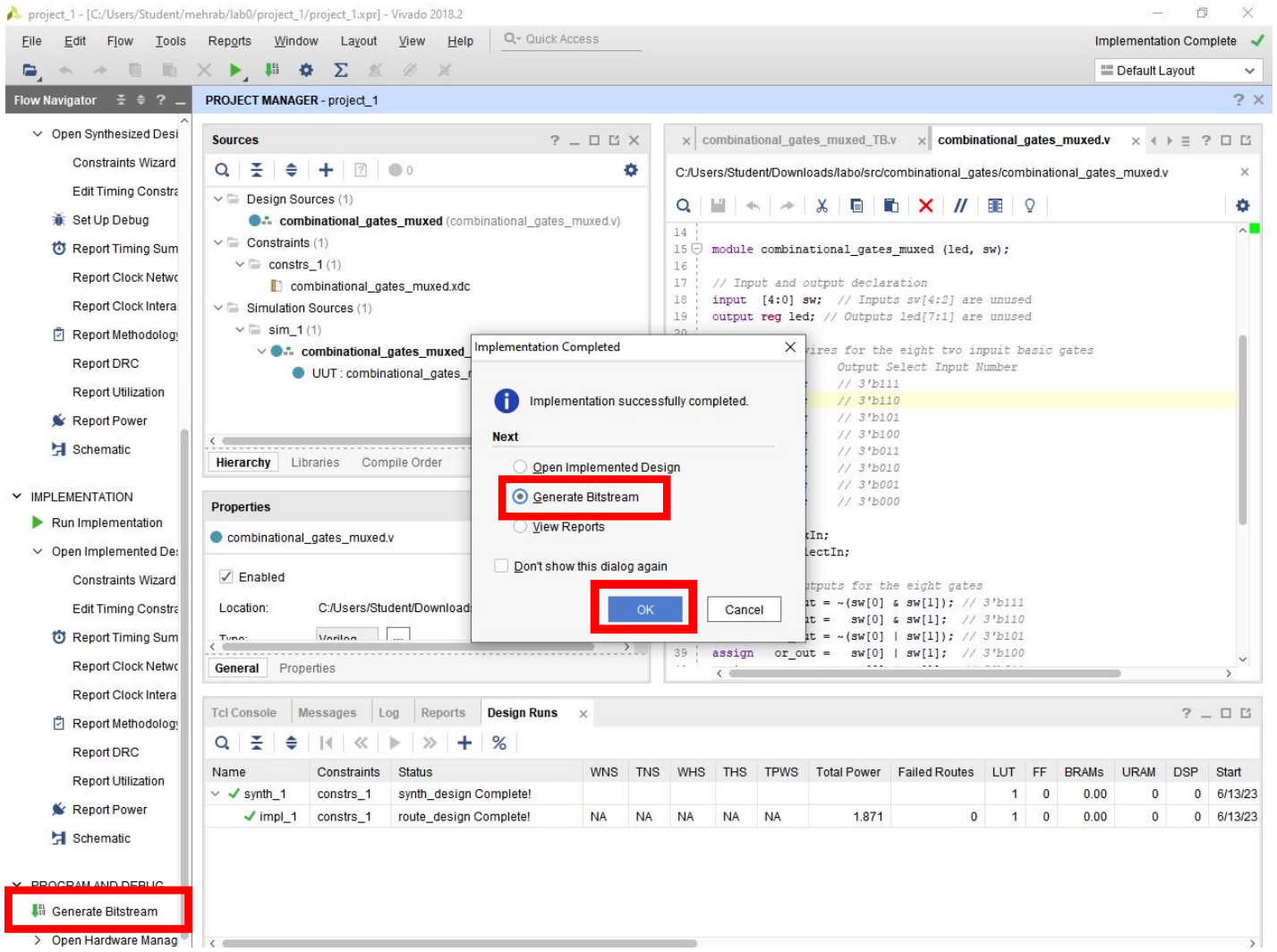


When VIVADO is done synthesizing your project, you will see the Synthesis Completed window.

Select ”Run Implementation” And Click Ok

Or alternatively you can click on “Run Implementation”

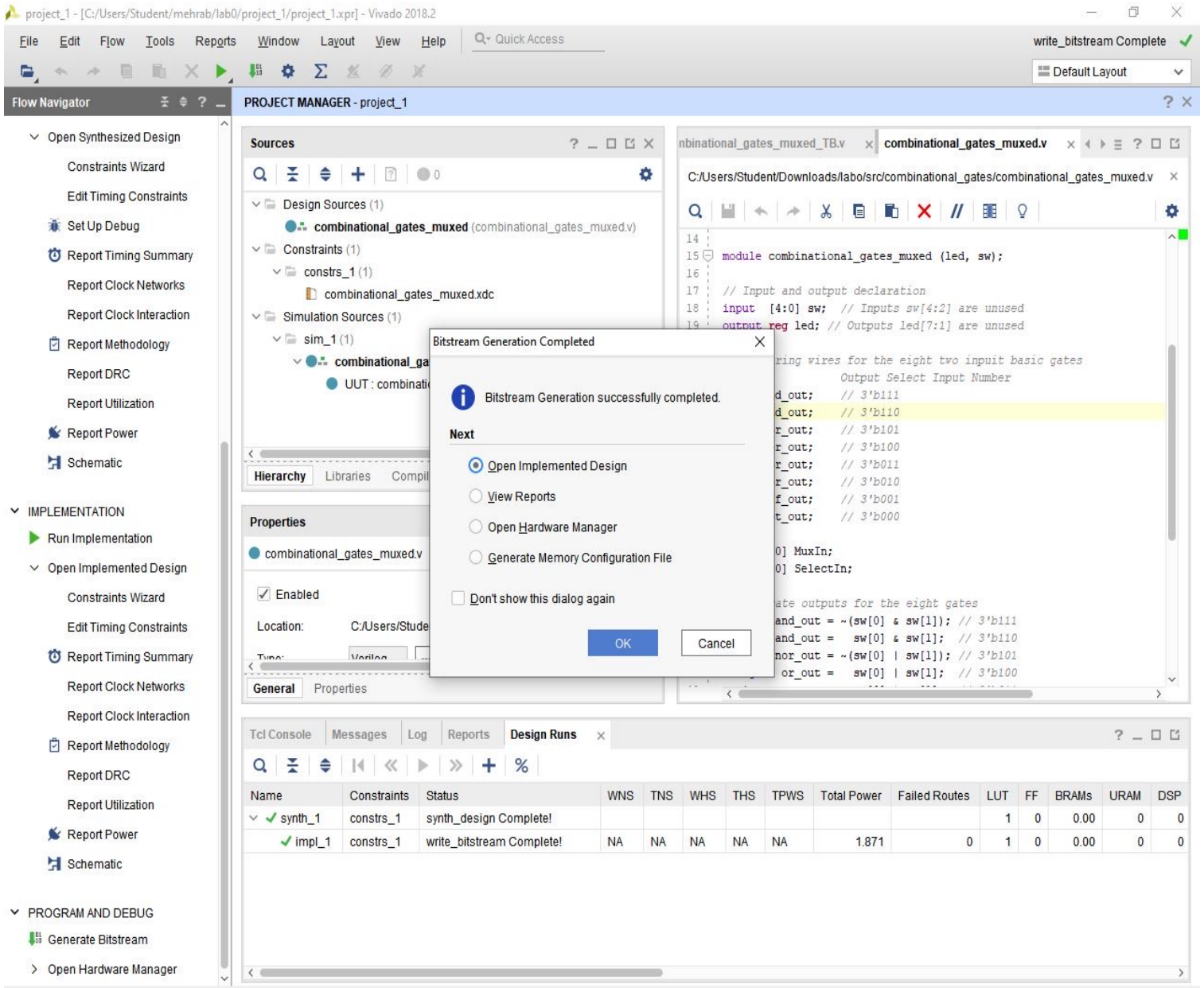
Generate Bitstream



When VIVADO is done implementing your project, you will see the Implementation Completed window.

Select ”Generate Bitstream” And Click Ok

Or alternatively you can click on “Generate Bitstream”



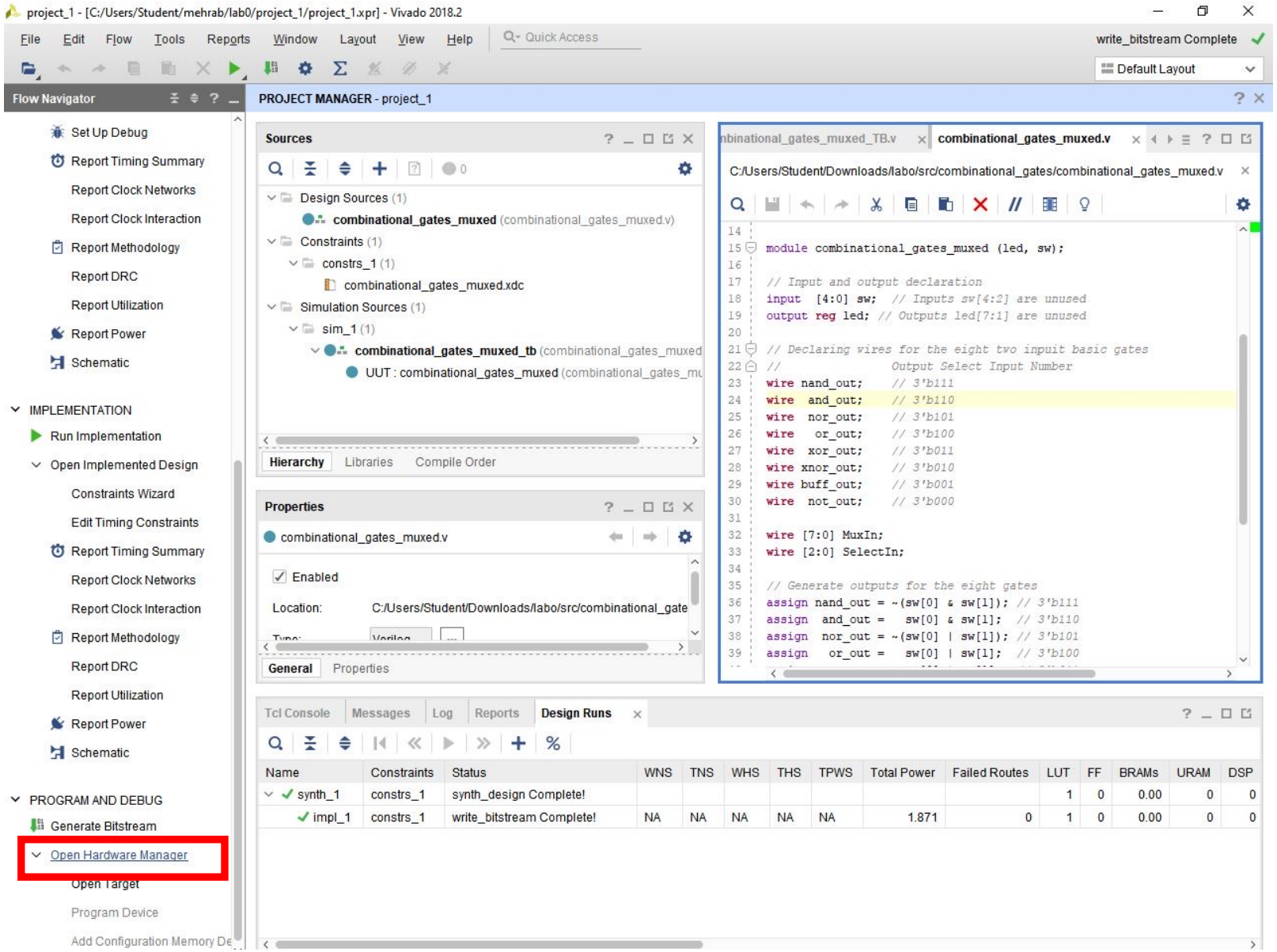
When VIVADO is done generating Bitstream, you will see the Bitstream Generation Completed window.

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.

Programming the FPGA Board



Click on
“Open Hardware Manager”

Connect the board

Make sure the programming cable is connected to the machine and the Power Switch is ON

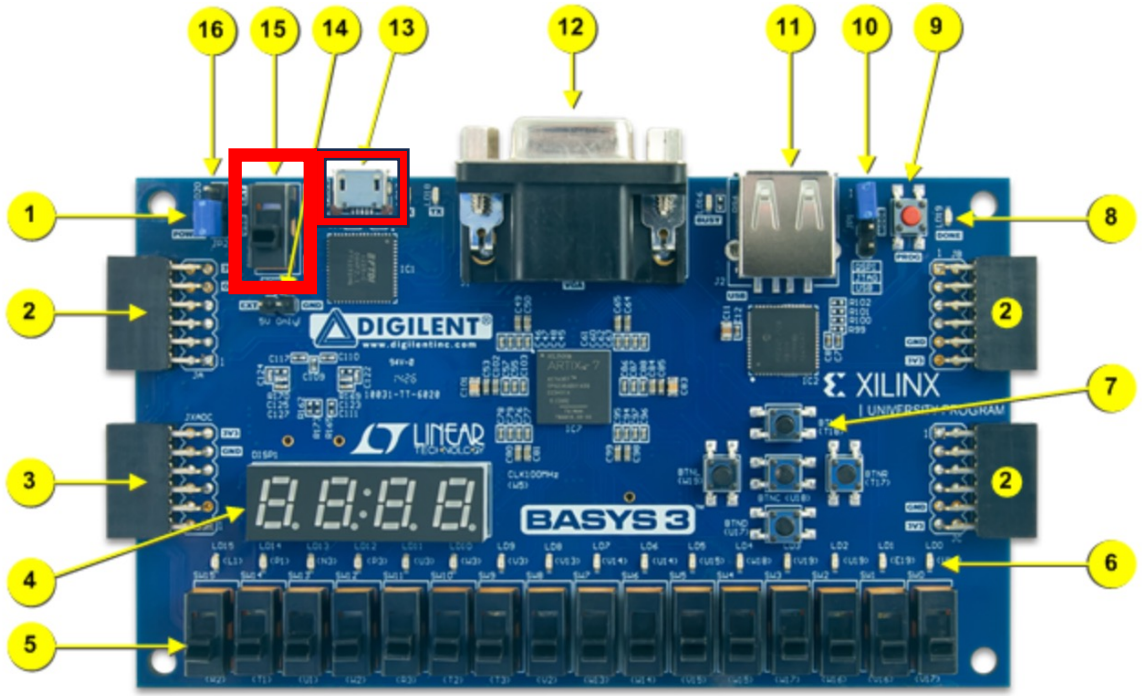
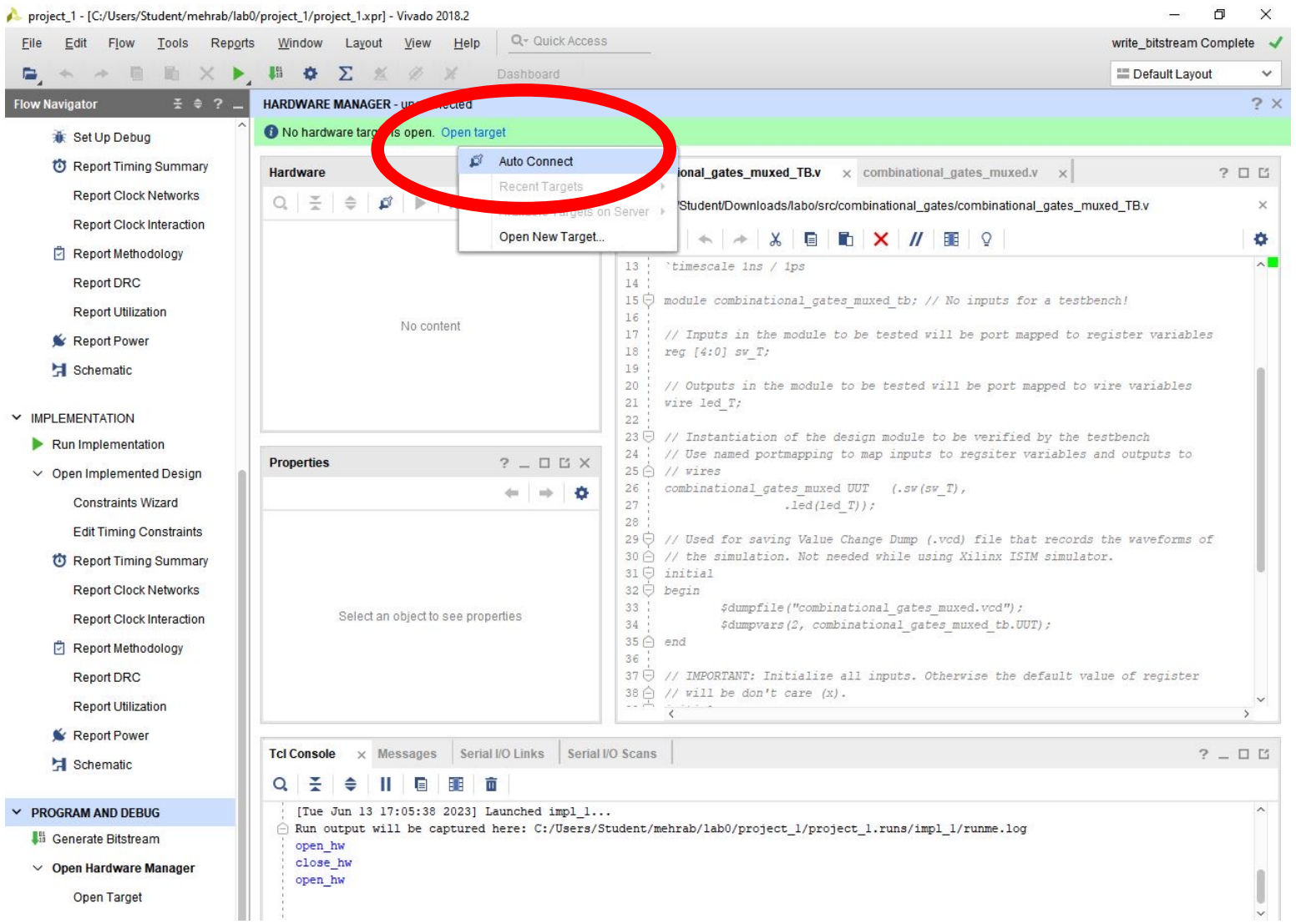


Figure 1. Basys3 board features

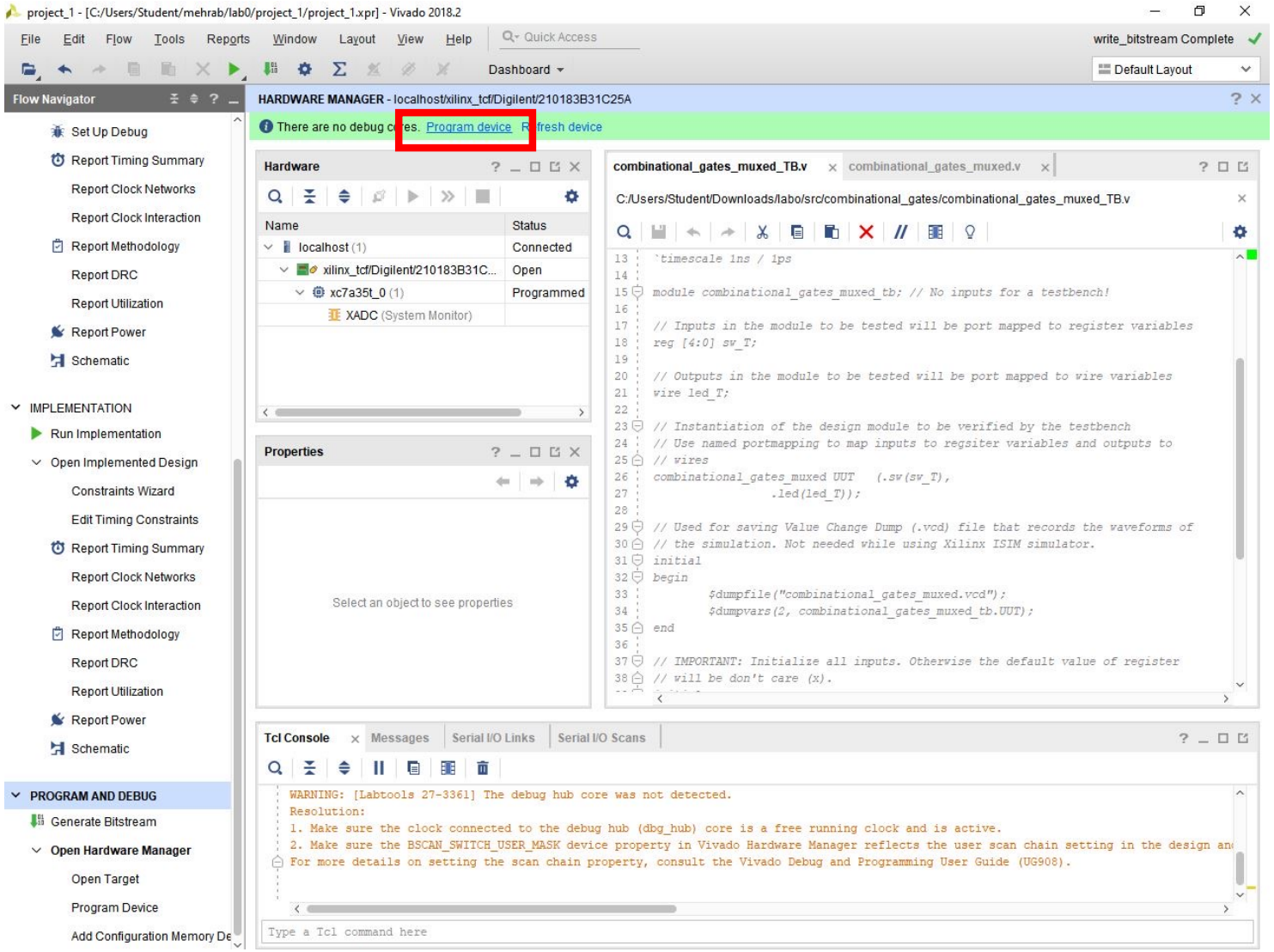
Callout	Component Description	Callout	Component Description
1	Power good LED	9	FPGA configuration reset button
2	Pmod connector(s)	10	Programming mode jumper
3	Analog signal Pmod connector (XADC)	11	USB host connector
4	Four digit 7-segment display	12	VGA connector
5	Slide switches (16)	13	Shared UART/ JTAG USB port
6	LEDs (16)	14	External power connector
7	Pushbuttons (5)	15	Power Switch
8	FPGA programming done LED	16	Power Select Jumper

Connect the board



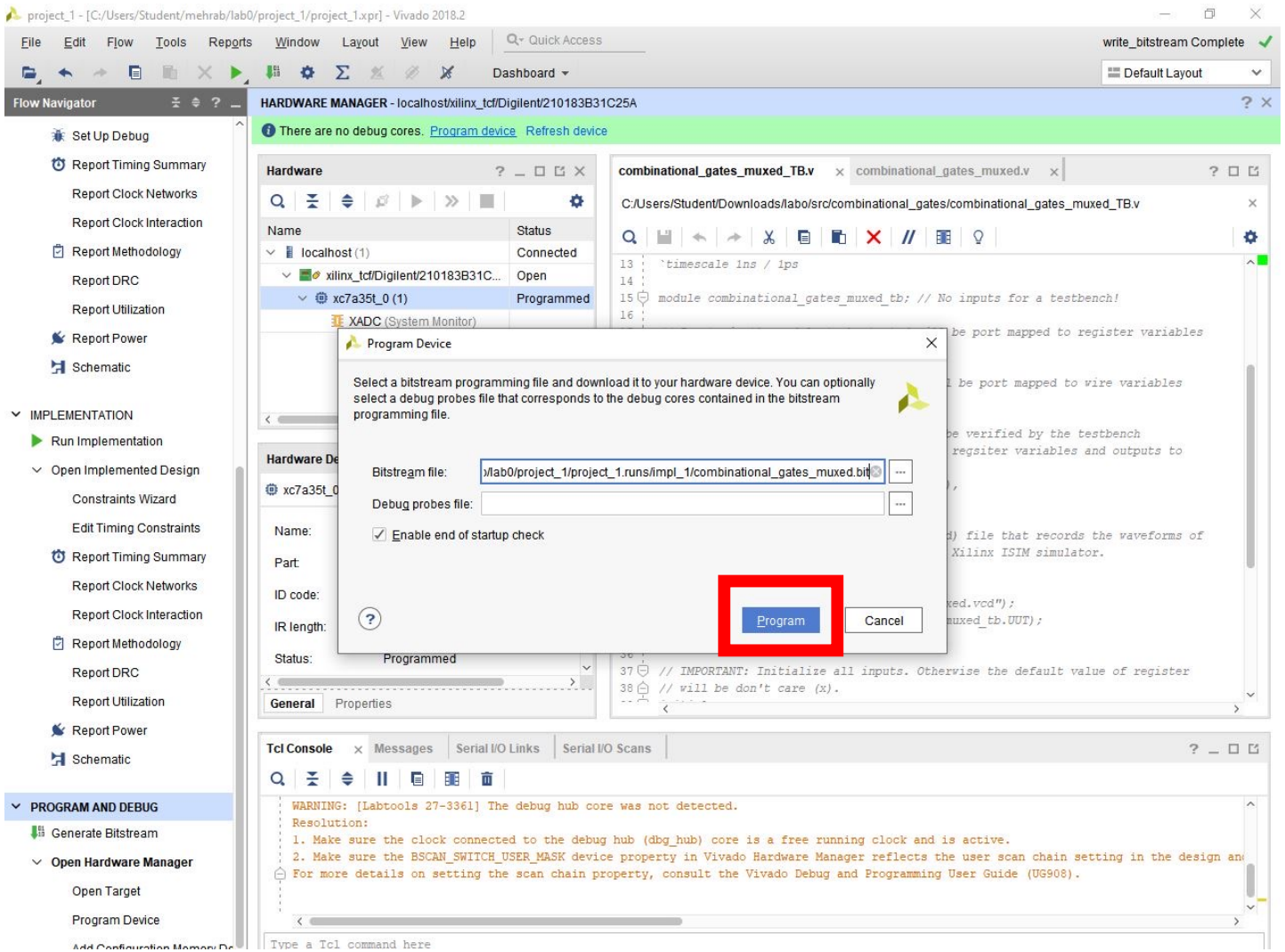
Click on “Open target” and then “Auto Connect”

Programming the Board



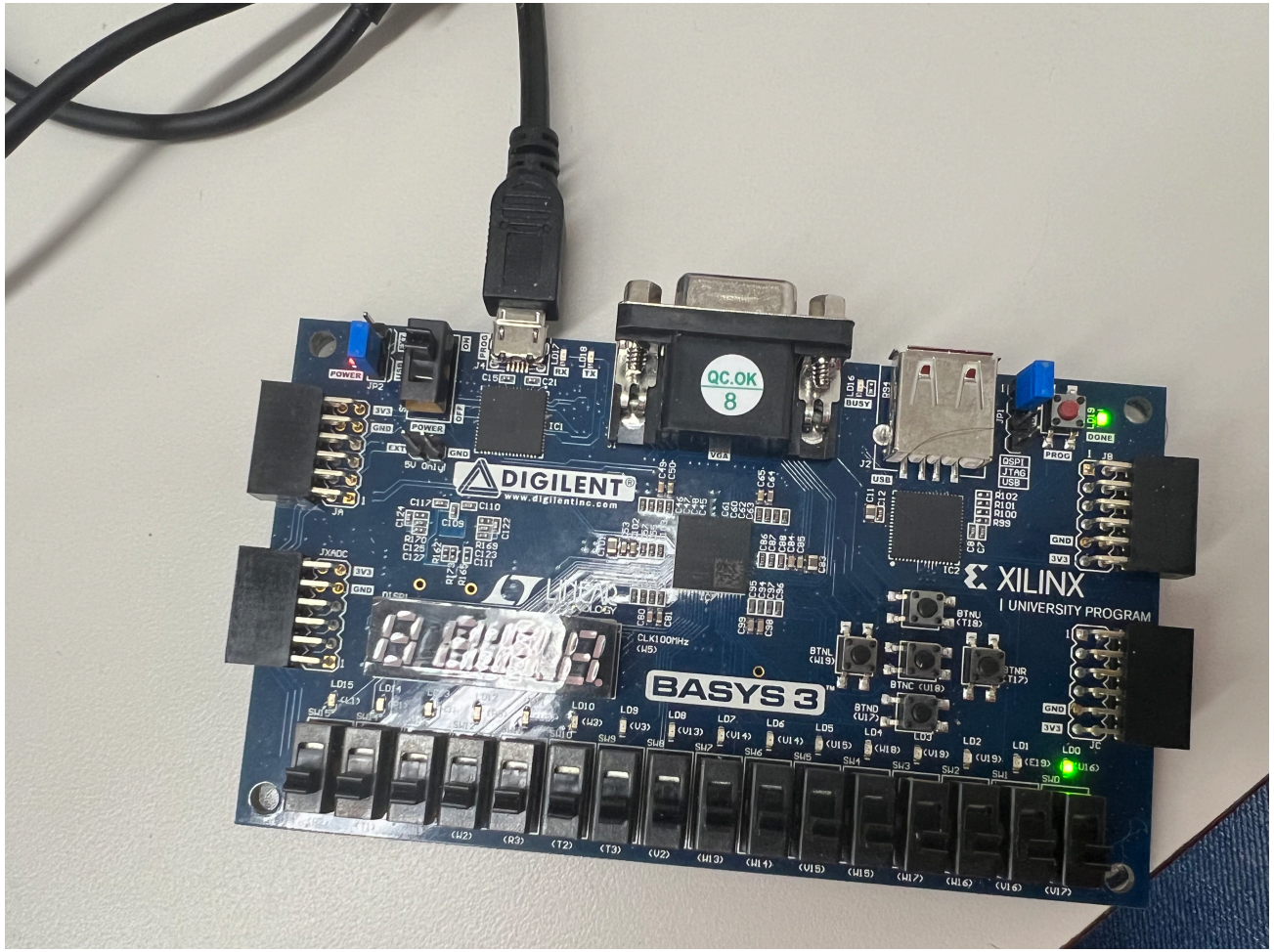
Click on “Program device”

Programming the Board



Find the bit file and click on Program

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