

VIVADO Quick Start Tutorial

Start the application either from the desktop or your operating system's application launcher, e.g. Windows menu / Start menu

Create a New Project

On the *Welcome screen* that greets you once you have started the Vivado application, within the **Quick Start** section you should be able to find the *Create Project* > option and click on that

Create a New Vivado Project

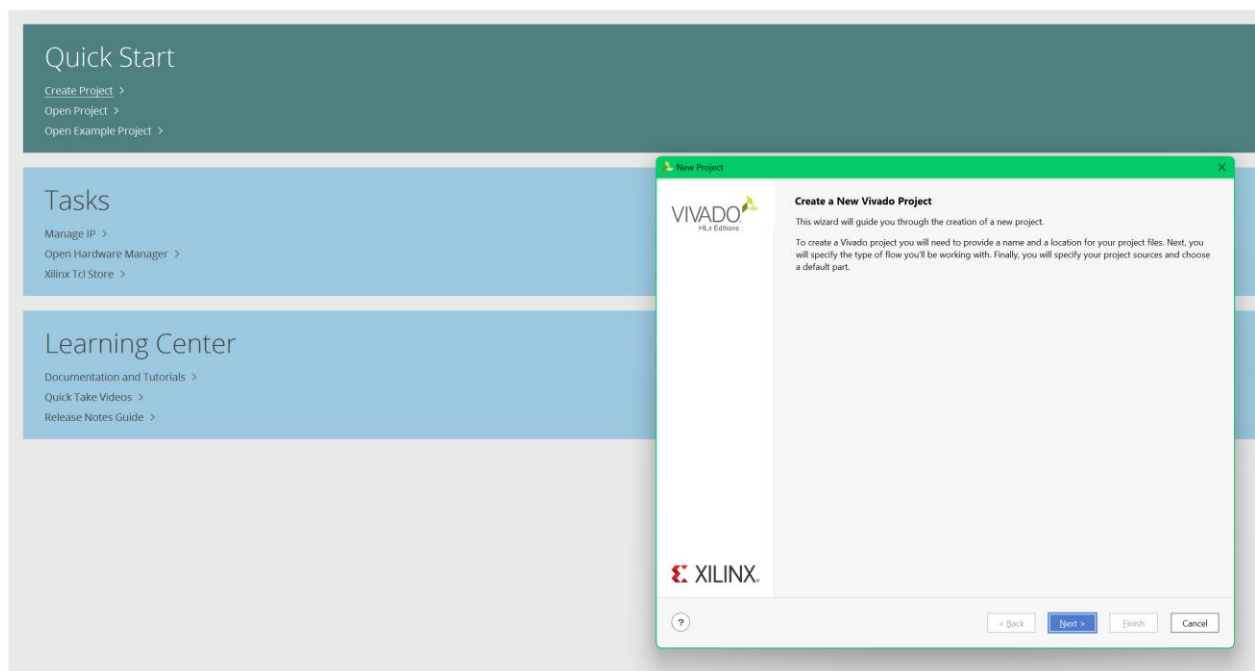
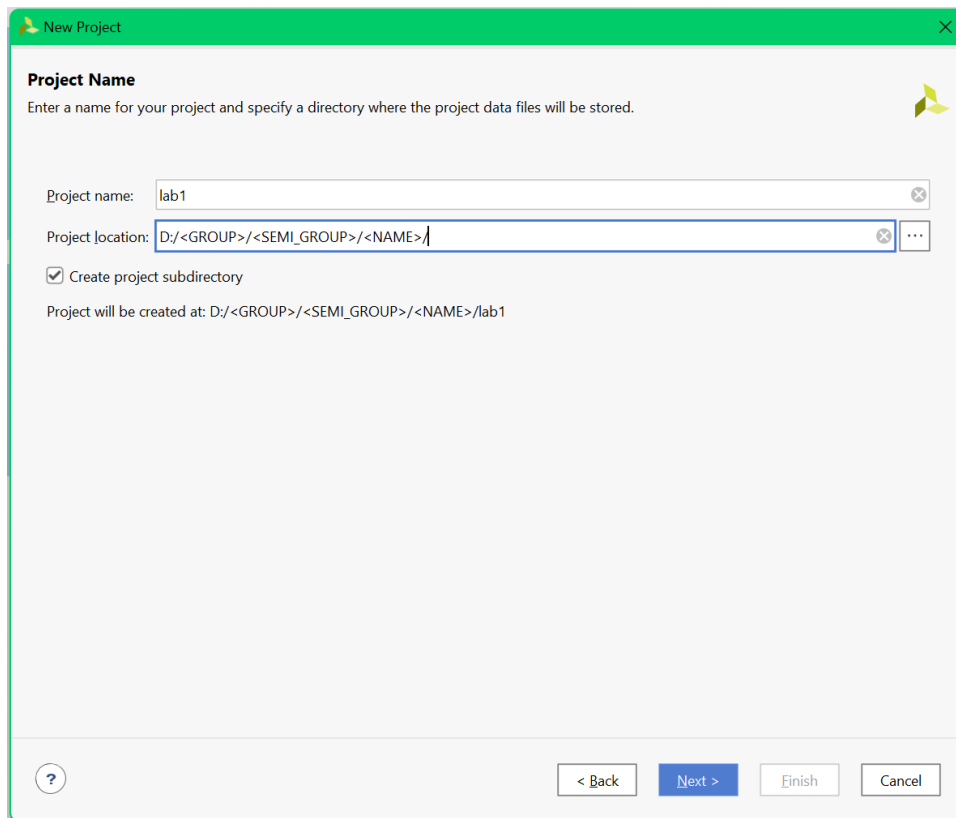


Figure 1. Create a New Vivado Project dialog

Click next, after which you will have to give your new project a *name* and *location* on the desktop's SSD.

Project Name

- For the project's name is *recommended* to give a unique name that you would remember
- For the project's location (as opposed to the name) it is *required* to follow the format outlined in the laboratory rules, meaning that it must be of the `D : / <GROUP> / <SEMI_GROUP> / <STUDENT_NAME>` format. Please make sure that there aren't any spaces and accents / diacritics in the given path. Also, please *do not* have the `< >` characters as part of the name, it's just meant to denote that there's supposed to be a value there (e.g. `D : / 30421 / II / a l i c e _ b o b`)



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 location:

Create project subdirectory

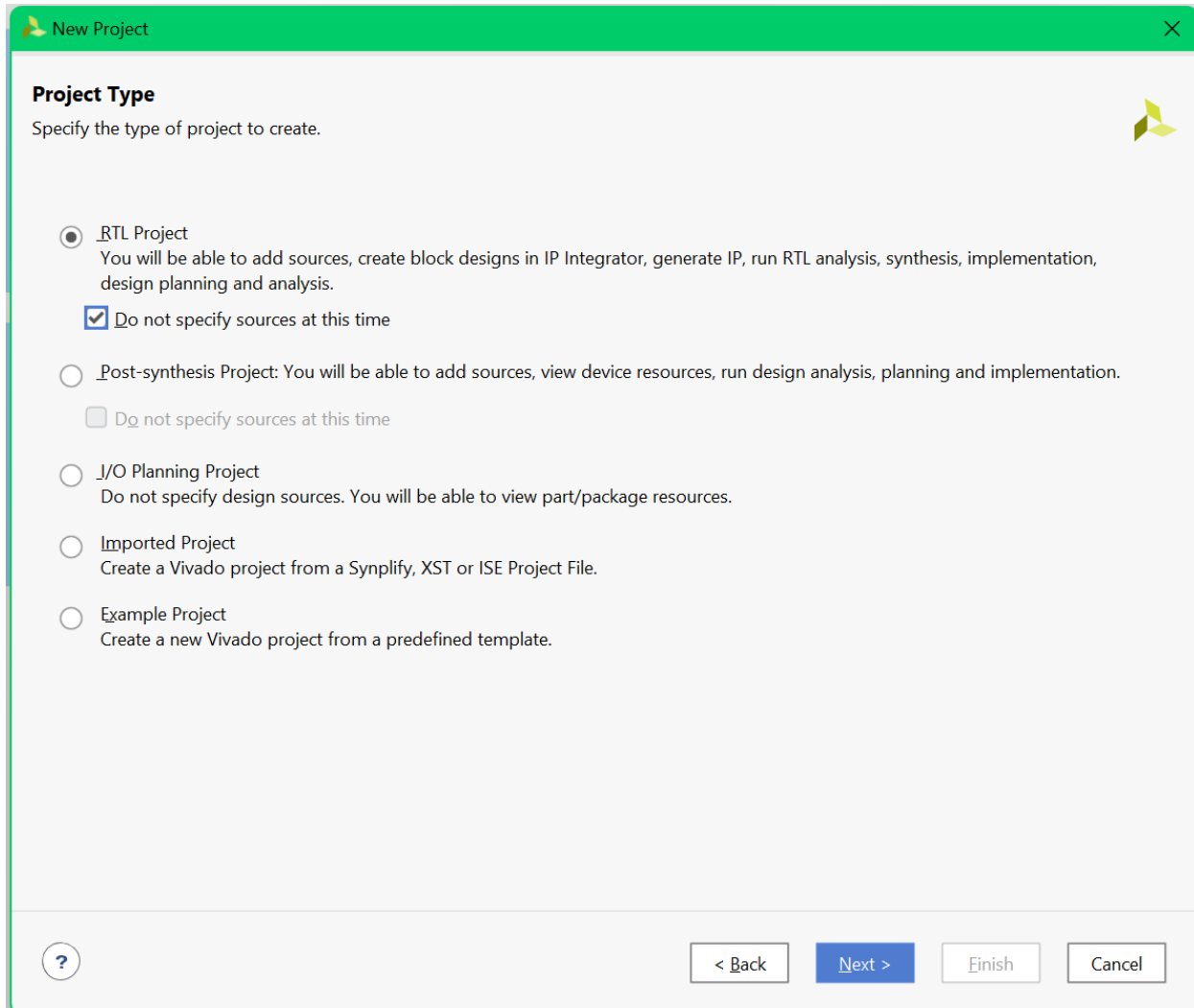
Project will be created at: D:/<GROUP>/<SEMI_GROUP>/<NAME>/lab1

Figure 2. Project Name and Location

After clicking next, you will be taken to the *Project Type* dialog

Project Type

You will select the first (selected by default) option, in other words **RTL Project** and you will want to make sure that the *Do not specify sources at this time* is checked as well



The screenshot shows a 'New Project' dialog box with a green title bar. The main content area is titled 'Project Type' and contains the instruction 'Specify the type of project to create.' There are five radio button options: 'RTL Project' (selected), 'Post-synthesis Project', 'I/O Planning Project', 'Imported Project', and 'Example Project'. Under the 'RTL Project' option, a checkbox labeled 'Do not specify sources at this time' is checked. At the bottom of the dialog, there is a help icon (question mark in a circle) on the left and four buttons: '< Back', 'Next >' (highlighted in blue), 'Finish', and 'Cancel'.

Figure 3. Project Type

Click next to progress to the next part, where you will specify the corresponding AMD / Xilinx FPGA model that you will be developing with.

Default Part

- For the first option, which is the *Category*, it should be on **All** by default and you will leave it as it is
- Below that, you will specify the FPGA *Family* which should be **Artix-7**
- After that, for the *package* type, you will select **csg324**
- The *speed* will be **-1**
- Lastly, the *Temperature* should be **All Remaining**

After setting the appropriate filtering values from above, you should be left with a limited number of choices, from which you will select the **xc7a100tcsg324-1**

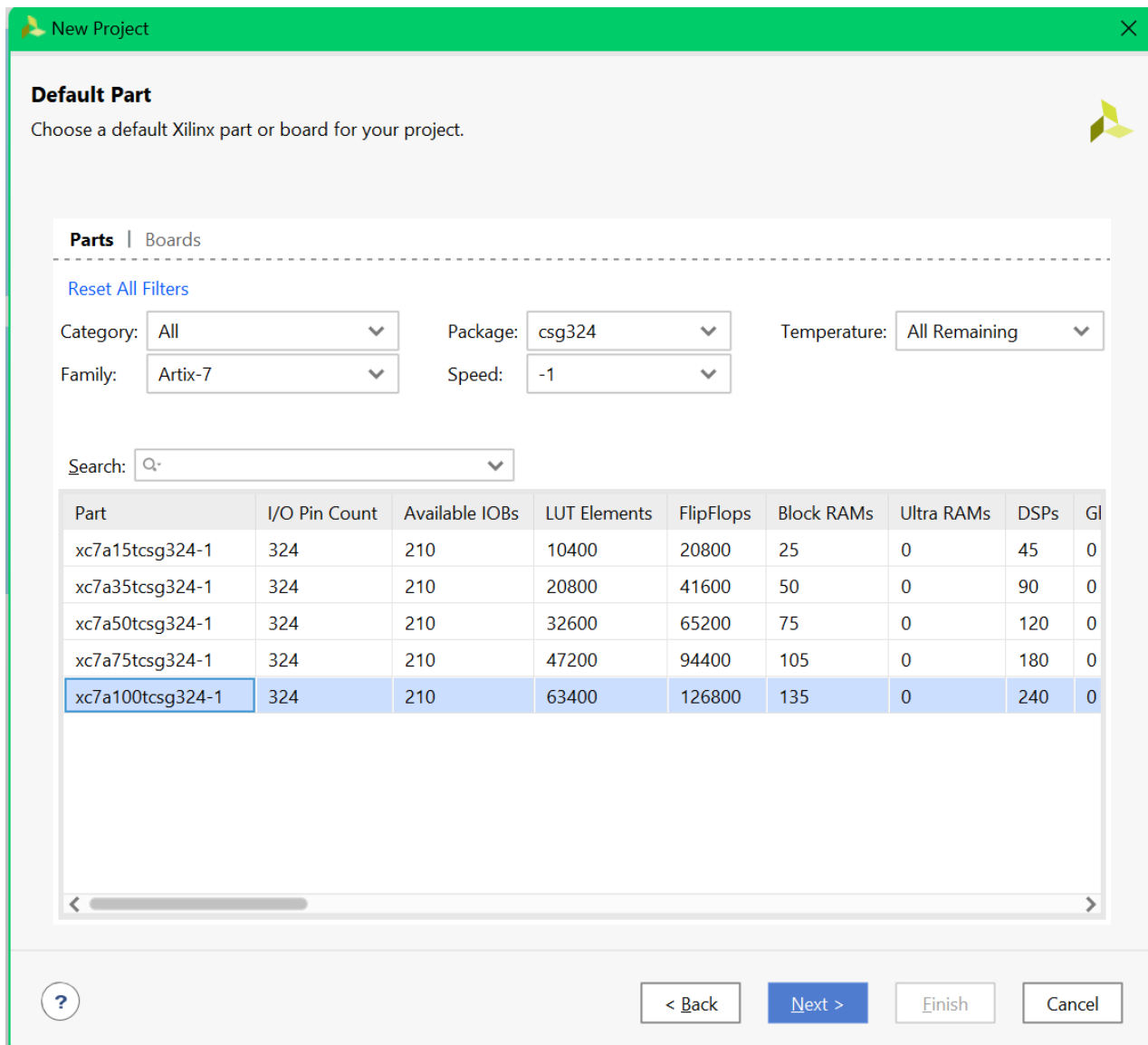


Figure 4. Default Part

Once you've verified the correct components, click on next to proceed to the *New Project Summary* dialog and then click on next again to progress.

Main Vivado Window

Project Language

Next your project will initialize, and you will be shown the core Vivado window, with the *Project Summary* again.

Pay close attention to the **Target Language** and **Simulator Language** values, since by default those are usually *Verilog* and *Mixed*, but they should both be set to *VHDL*.

Click on the corresponding values, which should bring up the settings window and you should be able to change them.

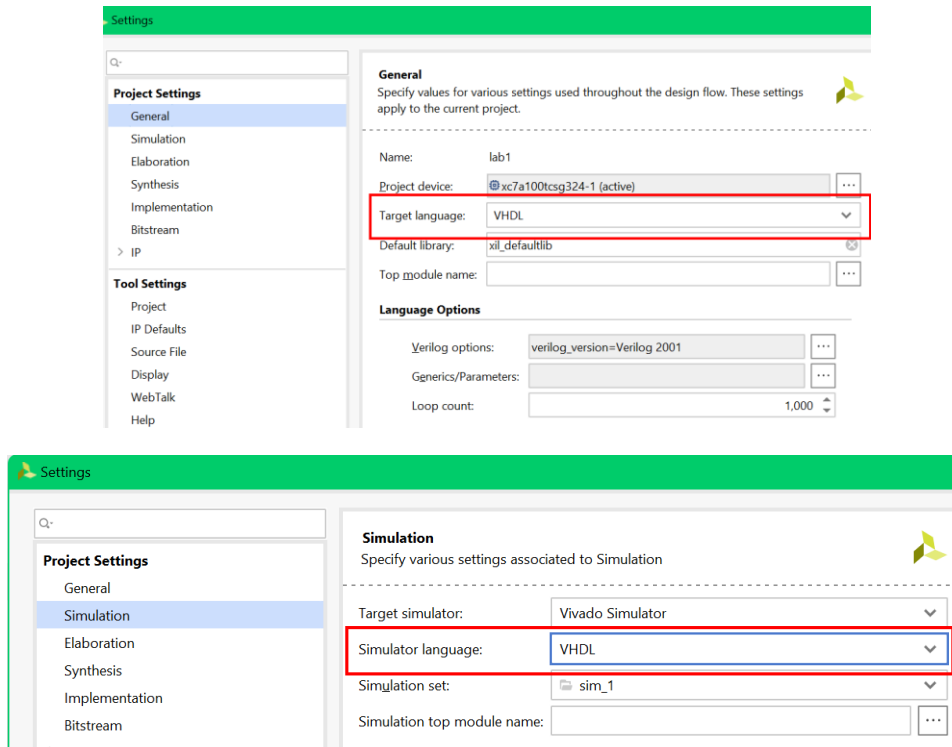
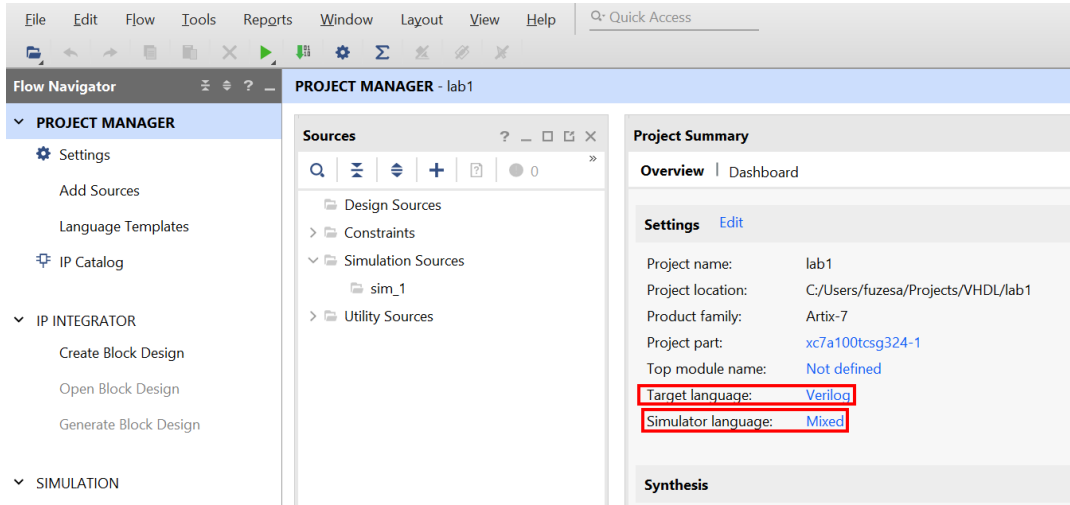


Figure 5. Verify Target & Simulator languages

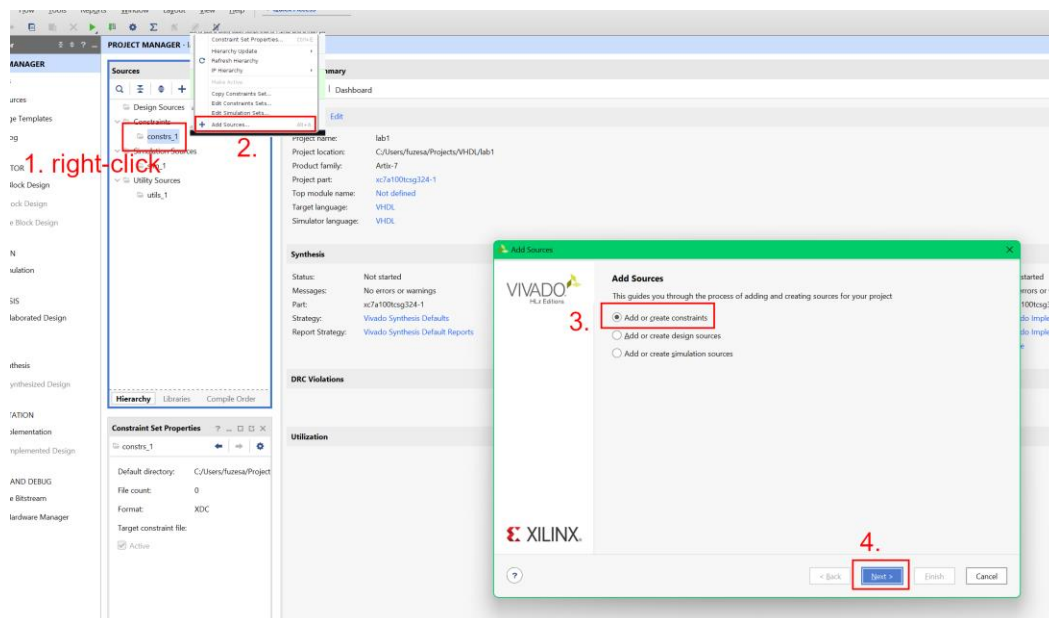
Add a constraints file

The constraints file that in the AMD / Xilinx environment usually has a .xdc (Xilinx Design Constraints) extension contains information as to how logically within your code / design, you will interface with the actual physical ports that are on the development board.

These are usually supplied by the manufacturer of the board – in our case Digilent / National Instruments – and they have the mapping as to which pin has what variable name.

The default Master XDC Constraints file is available on Digilent's GitHub repo, but we have modified it a bit for the purposes of this laboratory, so that it would make some things simpler, and you should be able to download it from the course's website.

It is possible that others before you have already downloaded it, so you will not have to download it again.



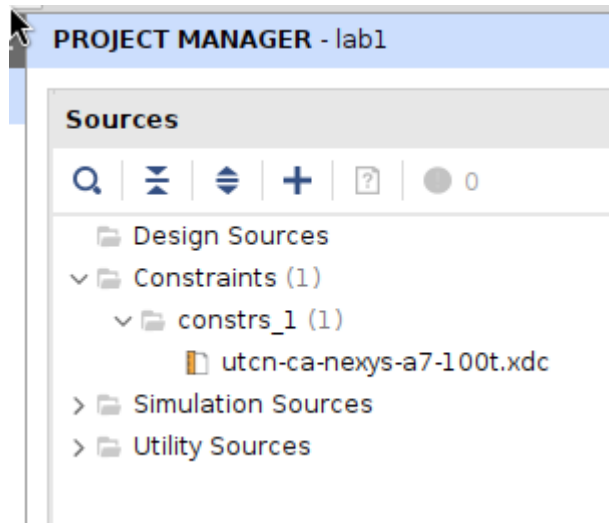
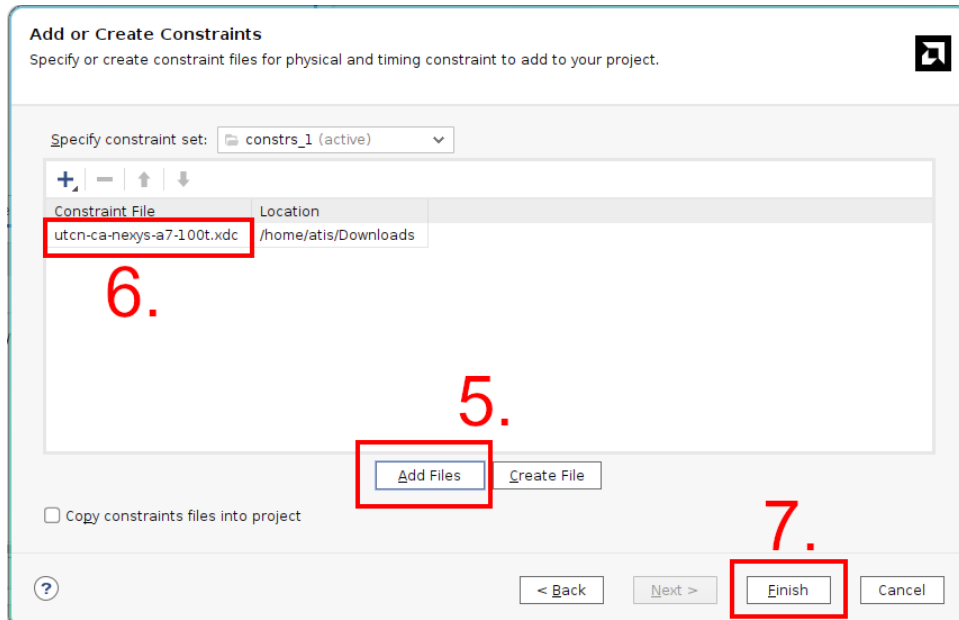


Figure 6. Add constraints file

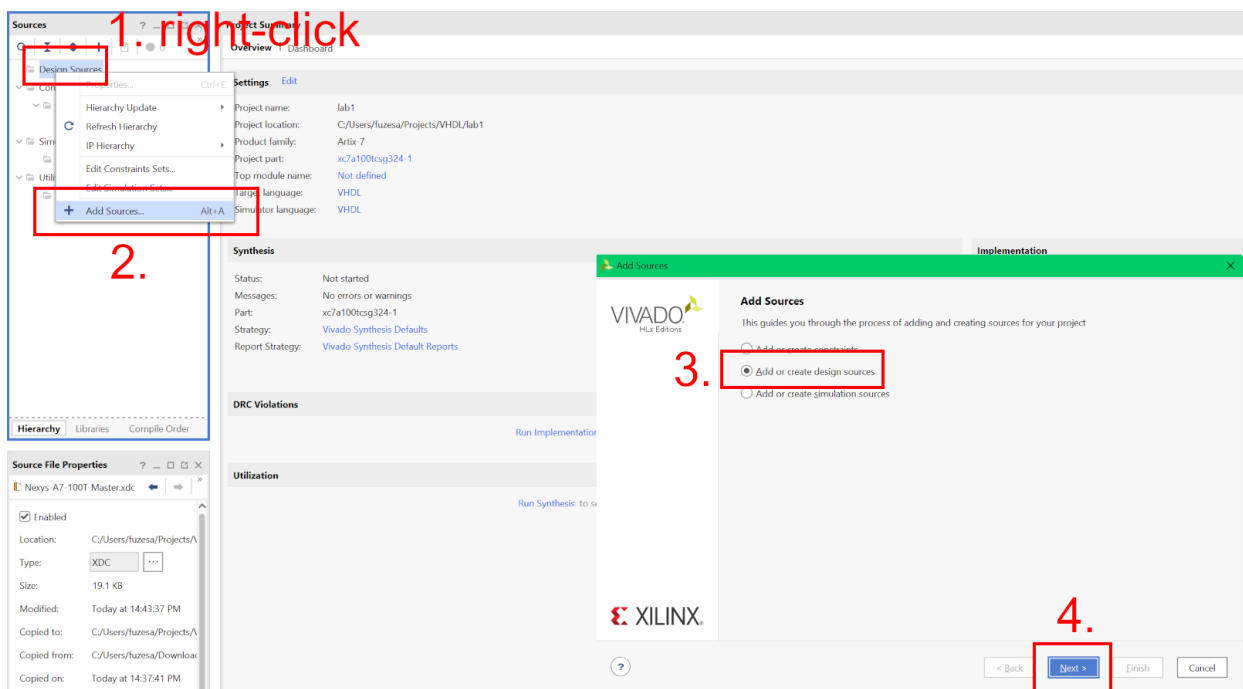
Once you have successfully added the constraints file, then you can proceed to adding a new *Design Source* in similar fashion.

Adding a new Design Source

Next, you will create your first source file for your hardware design.

Similarly to the constraints file above, you will right-click on the *Design Sources* folder within the sources panel, then on *Add Sources...*

Follow the dialogs outlined by the pictures below in order to create your first empty VHDL source file.



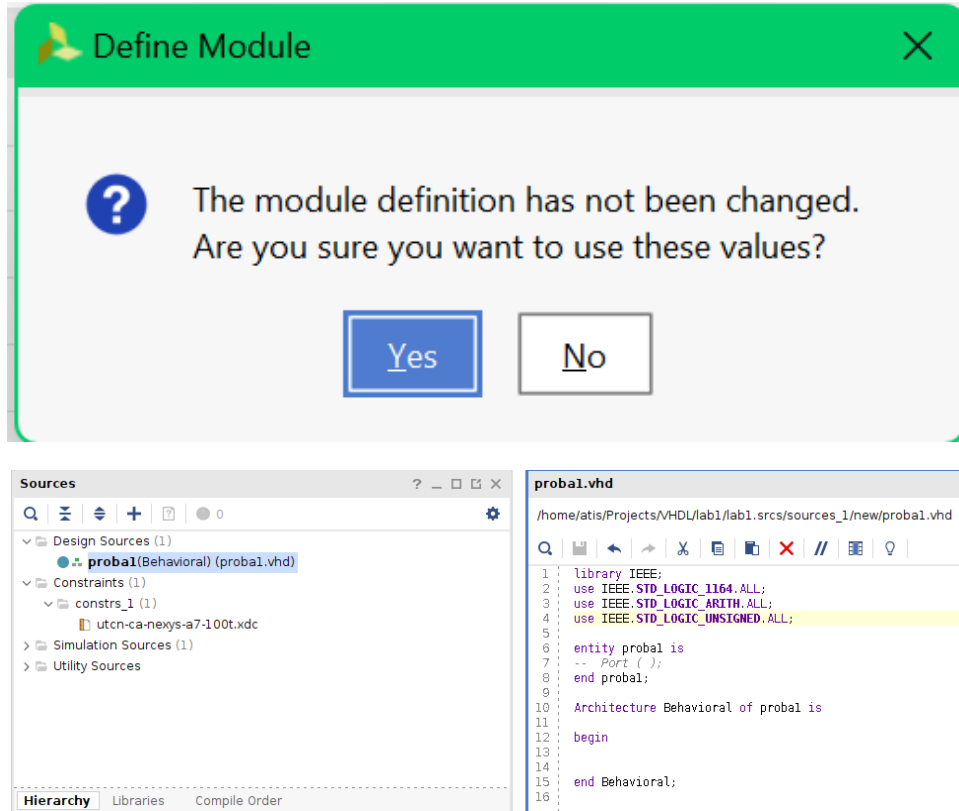


Figure 7. Add new design source

Add your code to the design source

You will notice that the newly created design source file contains a ton of different comments. For the moment, we will disregard that, and remove so that it wouldn't clutter our screen and we can focus on our code.

Copy-paste the code from below and go over the individual parts that might not be clear for you with your laboratory teacher.

NOTE: The code is also available at <https://github.com/UTCN-AC-CS-CA/nexys-a7-supplement>

Since there could be issues with copy-pasting the code from a PDF document, it might be better to copy it from the code repo

```
library ieee;
  use ieee.std_logic_1164.all;
  use ieee.std_logic_arith.all;
  use ieee.std_logic_unsigned.all;
```

```
entity proba1 is
  port (
    clk : in    std_logic;
    btn : in    std_logic_vector(4 downto 0);
    sw  : in    std_logic_vector(15 downto 0);
    led : out   std_logic_vector(15 downto 0);
    an  : out   std_logic_vector(7 downto 0);
    cat : out   std_logic_vector(6 downto 0)
  );
end entity proba1;

architecture behavioral of proba1 is

begin

  led <= sw;
  an  <= "000" & btn(4 downto 0);
  cat <= (others => '0');

end architecture behavioral;
```

Verifying & Uploading

Next – similarly as to how you go about building / compiling a software project – you will perform the three key steps in order to make sure that your design doesn't contain any major issues and upload it to the development board. The steps are outlined in the following sections. This section will go through some of the different components of the **Flow Navigator**, that can be found on the left side of your Vivado IDE.

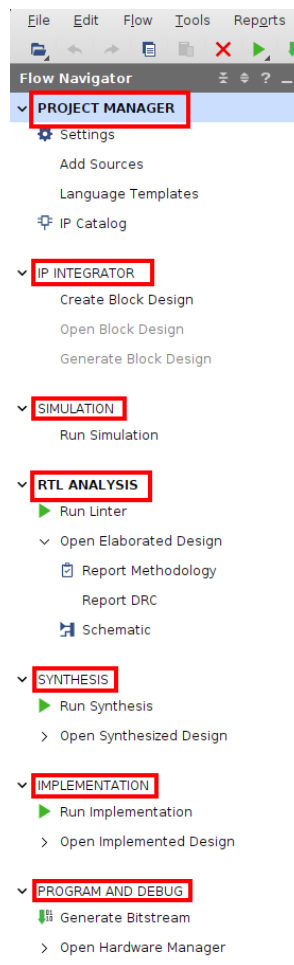


Figure 8. Flow Navigator

Elaborate Design (Optional)

For basic, incremental designs, it might be worth it to sometime visually verify that the code you have written is correctly interpreted by the synthesizing tool.

In order to do this, Vivado has the **RTL Analysis** section within the *Flow Navigator*.

First, you will have to initiate linting by clicking on the Run Linter option, which will check for any syntax errors. After that, expand the Open Elaborated Design menu item, to reveal several further options, and one of them should be the Schematic.

After running that, you should be presented with a schematic of your design, like the name implies.

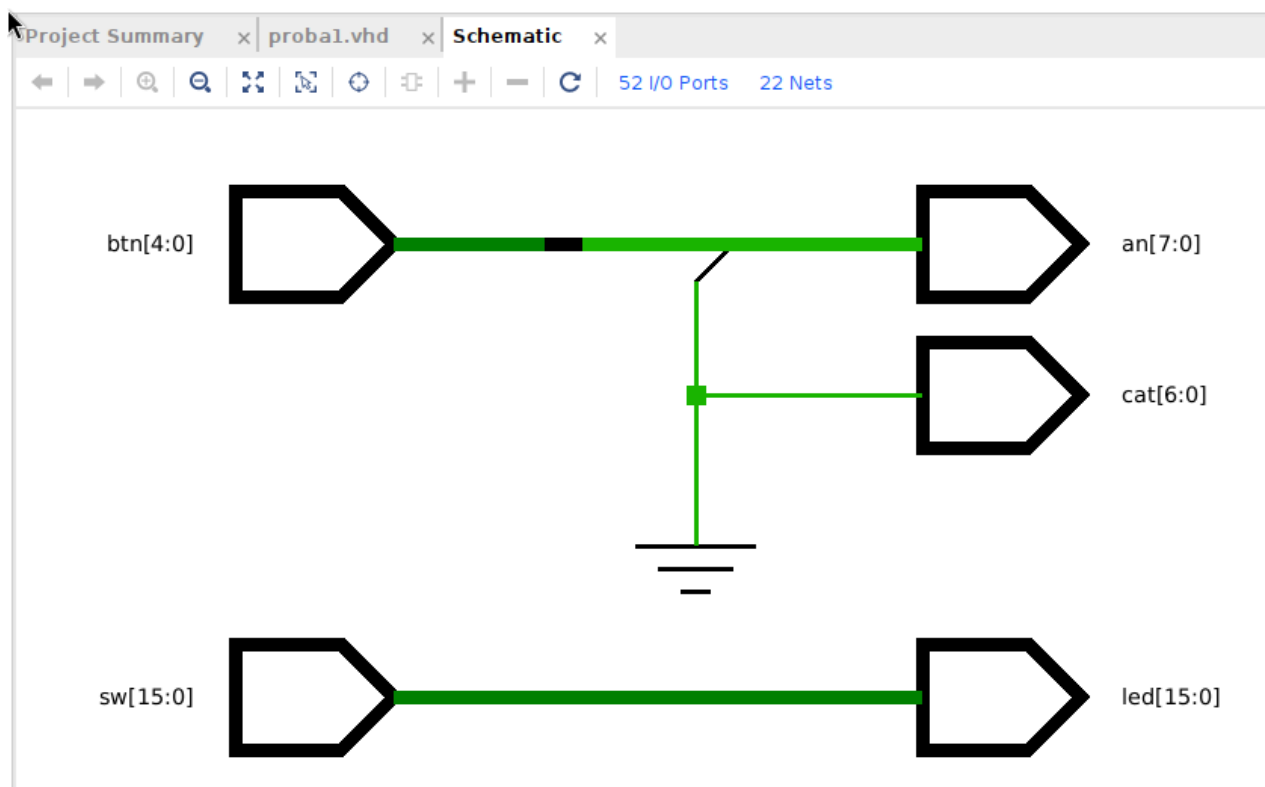


Figure 9. Elaborated Design / Generate Schematic

Synthesis

The first required step that you will be running is the **synthesis**. This is the process that will take your RTL design code and create a **netlist**, in other words a certain representation of the circuit with the different logic gates / electronic components, *without the placement*.

Newer versions of Vivado support the more recent VHDL-2008 and VHDL-2019 standards of the language, whereas previous ones the core version from 1993.

To begin the process, click on the Run Synthesis item from the **Synthesis** menu

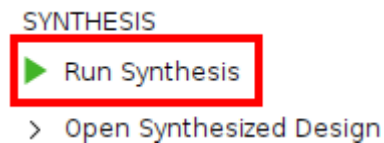


Figure 10. Run Synthesis

In the top right corner of the IDE, it will tell you as to which phase of the current process it's working on momentarily.

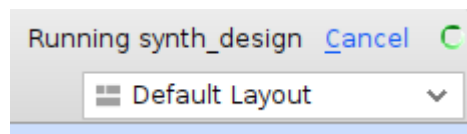


Figure 11. Current phase of the current process in the top-right corner

Once the process is complete, click on the Cancel button, since for the moment, we don't immediately wish to start with the implementation.

Implementation

The next crucial step will be the **implementation**, which is responsible for taking the netlist that was generated during the synthesis and based on that perform additional optimizations in order to create a placement of the necessary components and route them, in other words connect them with each other.

Just like with the previous step, click the Run Implementation item in the **implementation** section.



Figure 12. Start the implementation process

Similarly to the Synthesis, the status of the process will be displayed in the top right corner.

Once it has successfully completed, click on cancel again in order to not start another process or open a new dialog box.

Generate Bitstream

The last crucial step will be the fastest one as will, since the bulk of the necessary work has already been completed in the previous two processes.

This process will take your optimized implementation and create a binary file that will be uploaded to the development board's internal memory, thus applying your actualy logic / design of the hardware.

Just like with the other processes, you will start this process by clicking on the Generate Bitstream item from the **Program and Debug** menu.

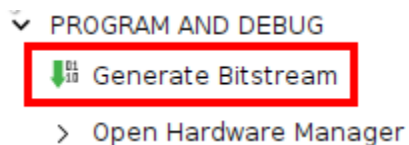


Figure 13. Generate Bitstream

Once the process completed successfully, this time around, instead of cancelling the dialog box, go ahead and select the "Open Hardware Manager" item, then click on "OK".

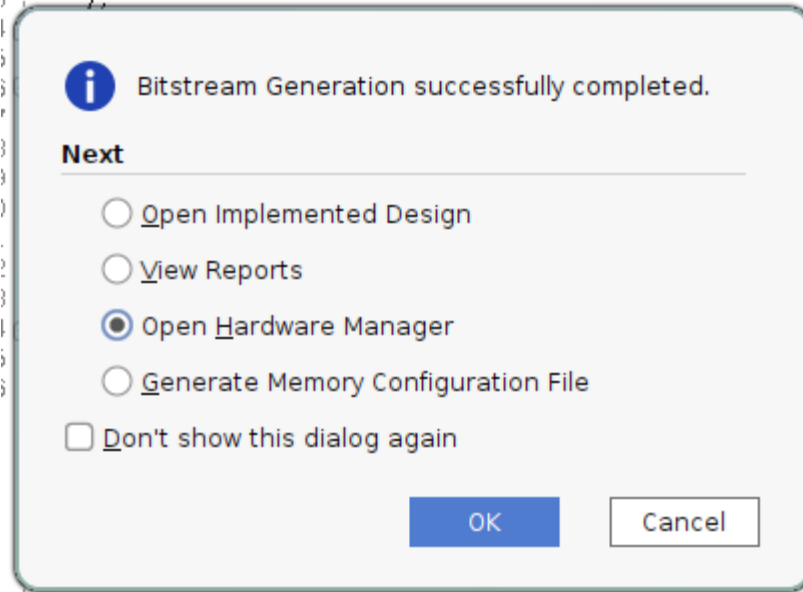


Figure 14. Open Hardware Manager

Upload the bitfile

Lastly, you will have to upload the development board. There are several different ways to achieve this, but the simplest would most likely be to click on the Open Target item within the **Open Hardware Manager** menu, then to click on “Auto Connect”.

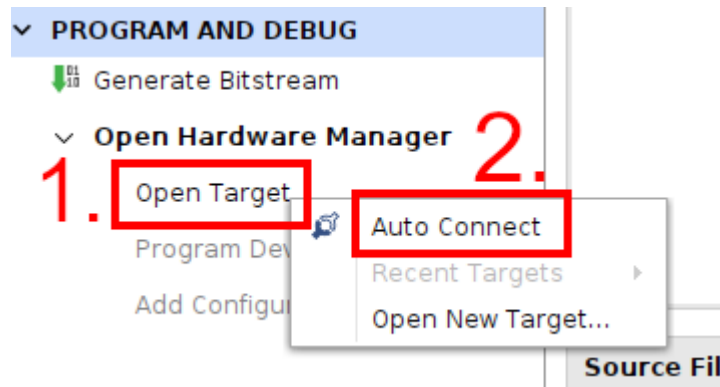


Figure 15. Open Target to Auto Connect

After this, you should click on the Program Device item that should now be available, since the IDE has detected a connection to the development board.

This will bring up an additional dialog, where the previously generated bitfile should be populated already with the full path. Confirming this dialog should begin the upload process, after which you could verify your design on the actual hardware itself.