THE IMAGINATION UNIVERSITY PROGRAMME

RVfpga Lab 1
Creating a Vivado Project

1. INTRODUCTION

In order to work with and modify RVfpga, you will need to build a project that includes all of the Verilog, SystemVerilog, header, configuration, and text files that define the system. In this lab, we show how to create a Vivado project that targets RVfpga to Digilent's Nexys A7 FPGA board, -100T version. (Remember that if you have a Nexys4 DDR board already, you can also use that as well.) By following these same steps, you will be able to modify the RVfpga core or system and resynthesize it.

IMPORTANT: Before starting RVfpga Labs, you must have already completed the RVfpga Getting Started Guide provided by the Imagination University Programme (https://university.imgtec.com/).

For example, if you have not already, install Xilinx's Vivado and Verilator following the instructions in the RVfpga Getting Started Guide. Also, make sure that you have copied the ***RVfpga*** folder that you downloaded from Imagination's University Programme to your machine. We will refer to the absolute path of the directory where you place folder RVfpga as [*RVfpgaPath*]. The RVfpga/src folder contains the Verilog and SystemVerilog sources for RVfpga, the RISC-V SoC that we will use and modify throughout the labs. The RVfpga/Labs folder contains some programs that you will use during Labs 1 to 10.

2. Creating a Vivado Project for RVfpga

You will use Xilinx's Vivado Design Suite[1] to build the RVfpga system using the RTL, the Verilog files that define the system. Follow these steps, detailed below, to build the RVfpga system and target it to a Nexys A7 FPGA board.

Step 1. Open Vivado
Step 2. Create a new RTL project
Step 3. Add the RTL source files and the constraint files
Step 4. Select Nexys A7 as target board
Step 5. Set rvfpga as Top Module and common_defines.vh as global
Step 6. Generate Bitstream

Step 1. Open Vivado

If you did not install Vivado on your machine as described in the RVfpga Getting Started Guide, do so now. Be sure to install the board files as well.

Now, run Vivado (in **Linux**, open a terminal and type: vivado; in **Windows**, open Vivado from the Start menu). The Vivado welcome screen will open. Click on Create Project (see Figure 1).

[1] In these materials we use Vivado 2019.2. Although most things should also work in more recent Vivado distributions, we highly recommend the use of this version.

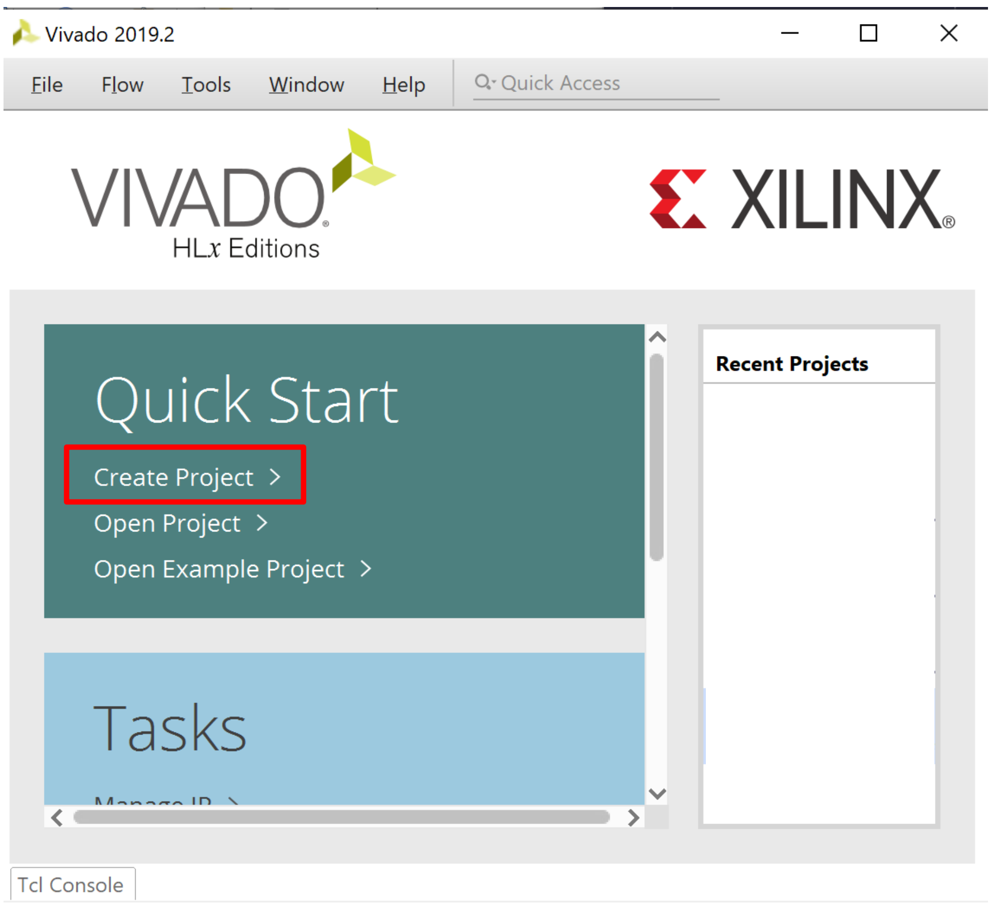
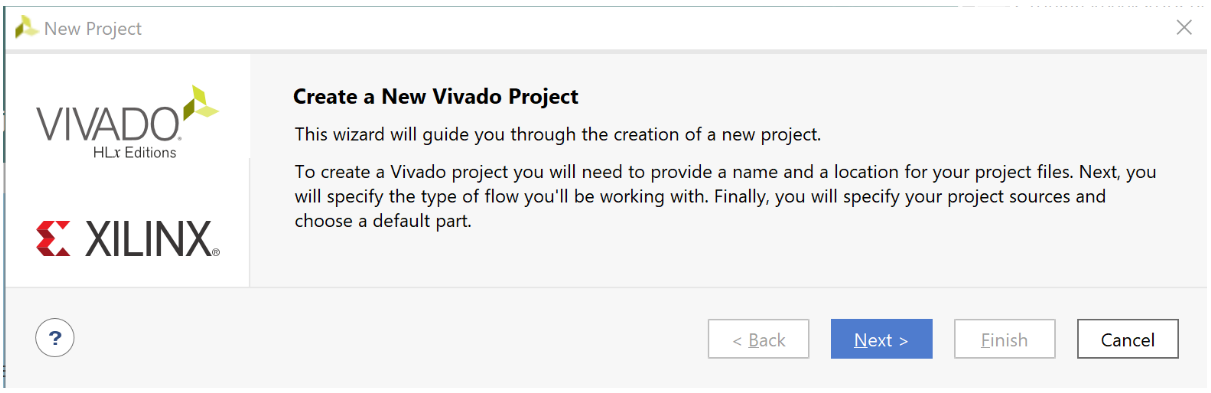


Figure 1. Vivado welcome screen: Create Project

Step 2. Create a new RTL project
The Create a New Vivado Project Wizard will now open (see Figure 2). Click Next.



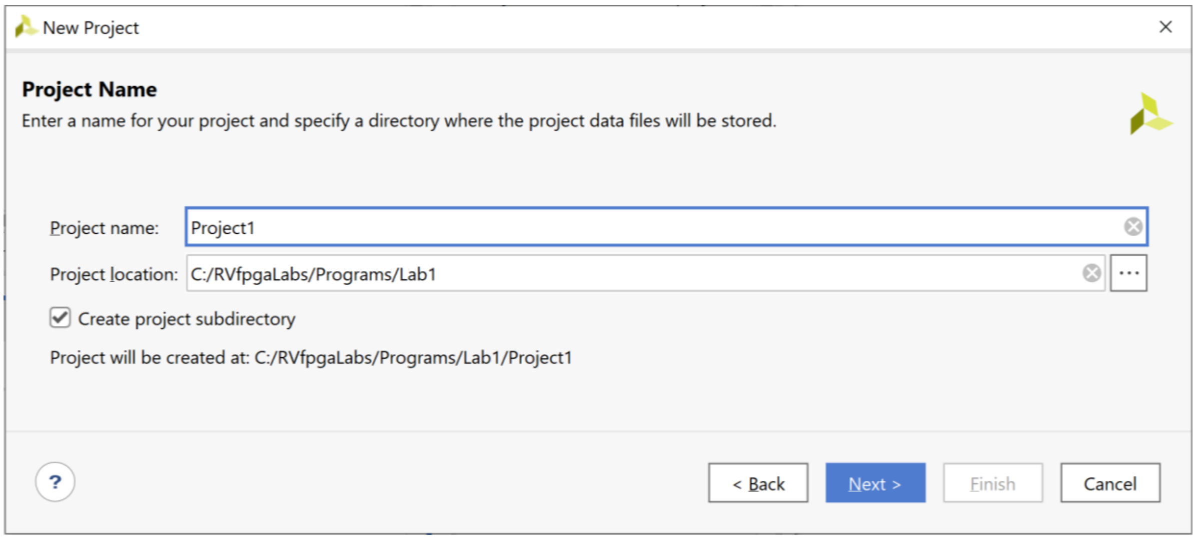
Create a New Vivado Project

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.

Figure 2. Create a New Vivado Project Wizard

Call the Project name Project1 and place it in the *[RVfpgaPath]/RVfpga/Labs/Lab1* folder. Then click Next (see Figure 3).


New Project

Project Name

Enter a name for your project and specify a directory where the project data files will be stored.

Project name: Project1

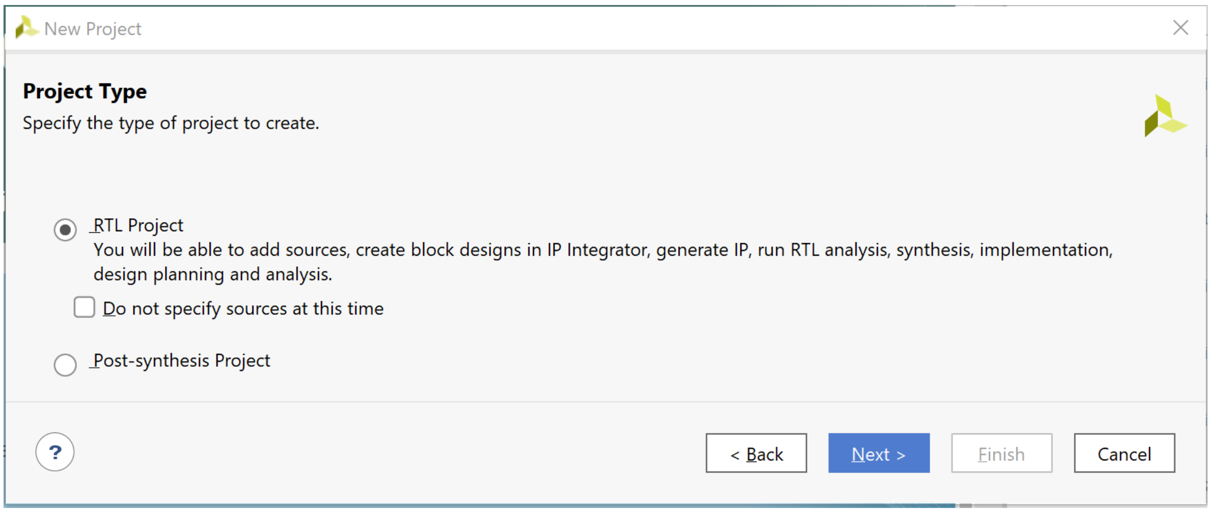
Project location: C:/RVfpgaLabs/Programs/Lab1

[✓] Create project subdirectory

Project will be created at: C:/RVfpgaLabs/Programs/Lab1/Project1

Figure 3. Project Name

Select the project type as RTL Project, and click Next (see Figure 4).


New Project

Project Type

Specify the type of project to create.

● RTL Project
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.

☐ Do not specify sources at this time

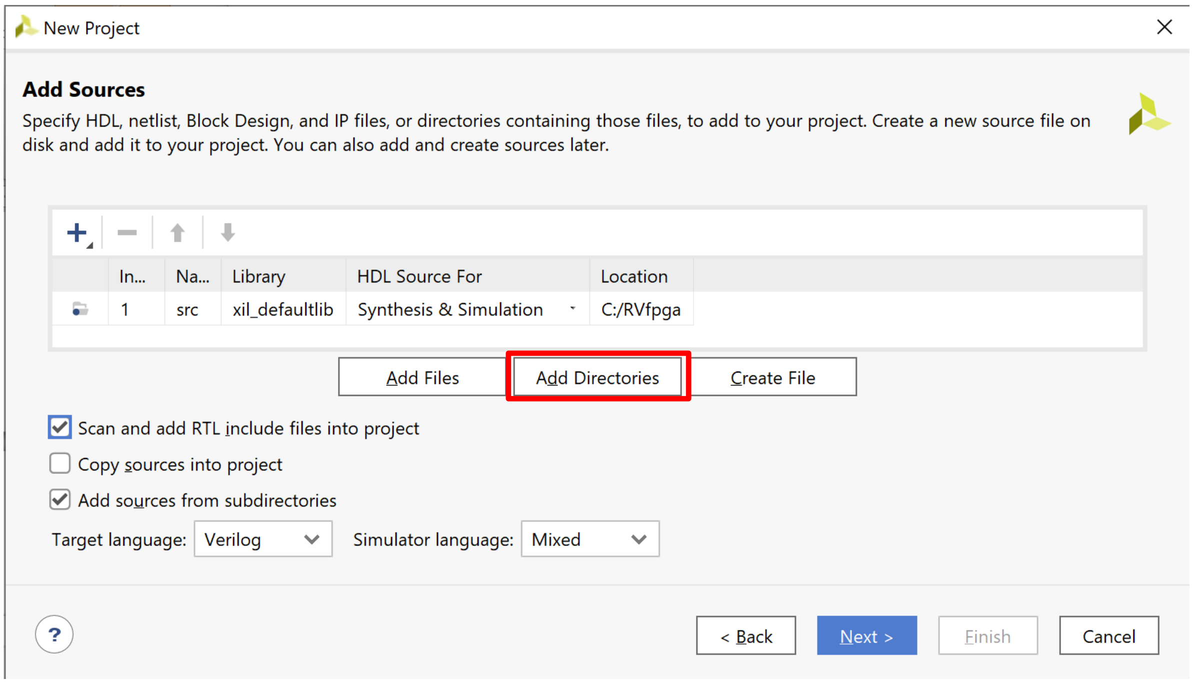
○ Post-synthesis Project

Figure 4. RTL Project

Step 3. Add the RTL source files and the constraint files
In the Add Sources window, click on Add Directories, and select *[RVfpgaPath]/RVfpga/src* (see Figure 5). Make sure both of the following options are selected (as shown in Figure 5):

- Scan and add RTL include files into project
- Add sources from subdirectories

Then click Next.


New Project

Add Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

In...	Na...	Library	HDL Source For	Location
1	src	xil_defaultlib	Synthesis & Simulation	C:/RVfpga

[✓] Scan and add RTL include files into project
[ ] Copy sources into project
[✓] Add sources from subdirectories

Target language: Verilog
Simulator language: Mixed

Figure 5. Add Sources

You will now add the constraints for the system. These files map the signal names to the pins on the board. For example, the LEDs on the Nexys A7 FPGA board are connected to FPGA pins on the board through traces in the PCB. Vivado must know this so that it maps the correct signal name in the RTL to the correct FPGA pin. For example, the following line in the *[RVfpgaPath]/RVfpga/src/rvfpga.xdc* file, a Xilinx design constraints file, indicates that FPGA pin H17 maps to the least significant LED (`o_led[0]`) and that it uses LVCMOS 3.3V signalling:

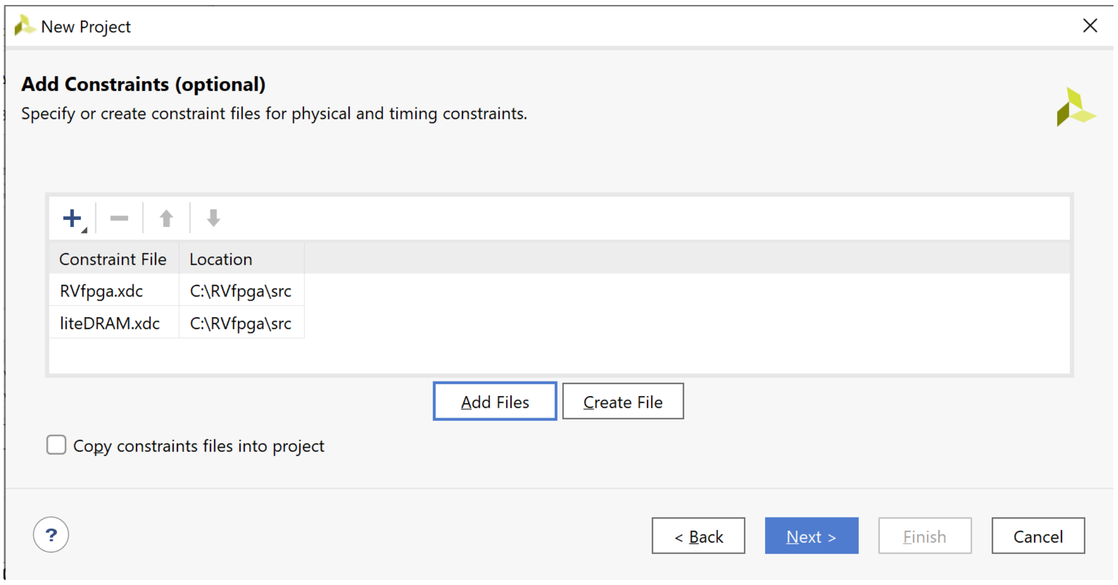
```
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { o_led[0] }]
```

Note that the signal name `o_led` is the name used in RVfpga's Verilog code to drive the Nexys A7 board's LEDs.

In the Add Constraints window, click on Add Files and select the following two files (see Figure 6):

[RVfpgaPath]/RVfpga/src/rvfpga.xdc
[RVfpgaPath]/RVfpga/src/litedram.xdc

Then click Next.


New Project

Add Constraints (optional)

Specify or create constraint files for physical and timing constraints.

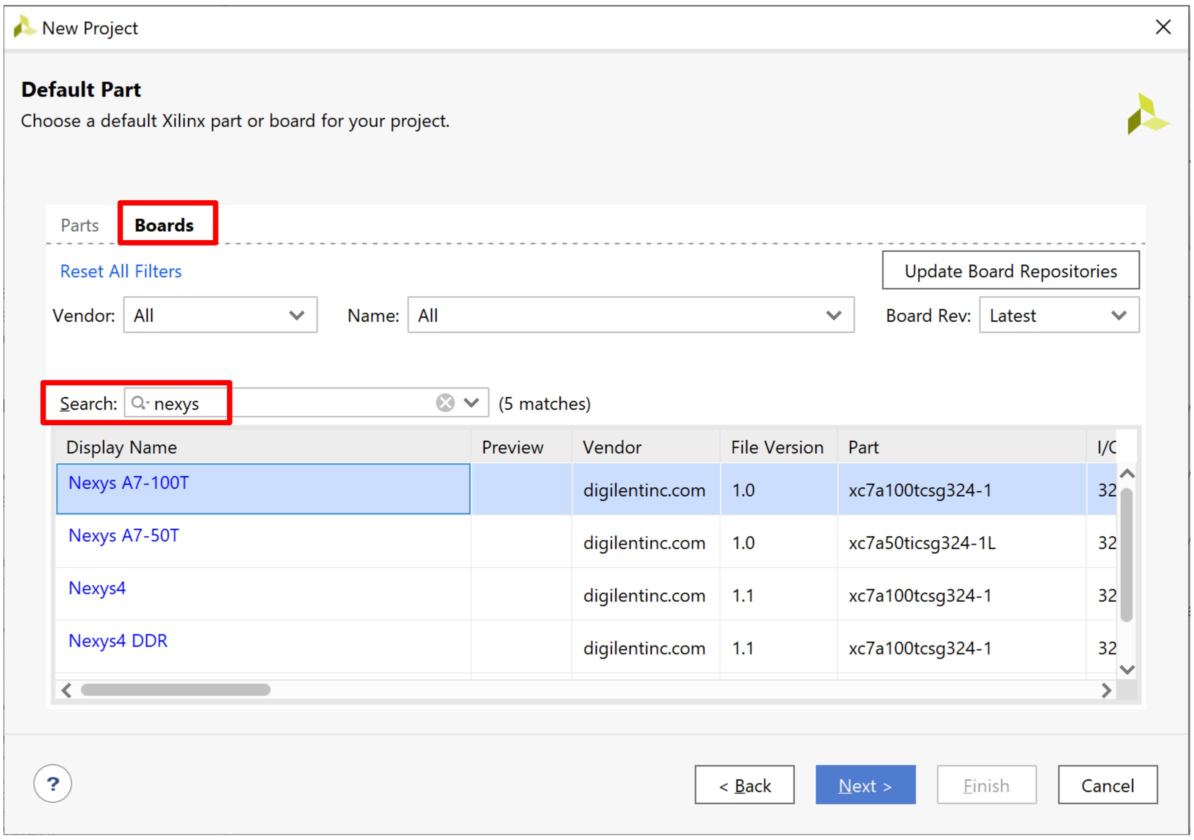
Constraint File	Location
RVfpga.xdc	C:\RVfpga\src
liteDRAM.xdc	C:\RVfpga\src

Figure 6. Add Constraints

Step 4. Select Nexys A7 as target board

In the Default Part window, click on Boards and then select Nexys A7-100T (see Figure 7). You may use the Search box to narrow down the results. You will also notice that the name of the actual target FPGA is listed in the Part column: xc7a100tcsg324-1. This indicates that it is a Xilinx Artix-7 FPGA with 100k equivalent gates with a CSG (chip-scale grid) package and 324 pins.

Click Next.


New Project

Default Part

Choose a default Xilinx part or board for your project.

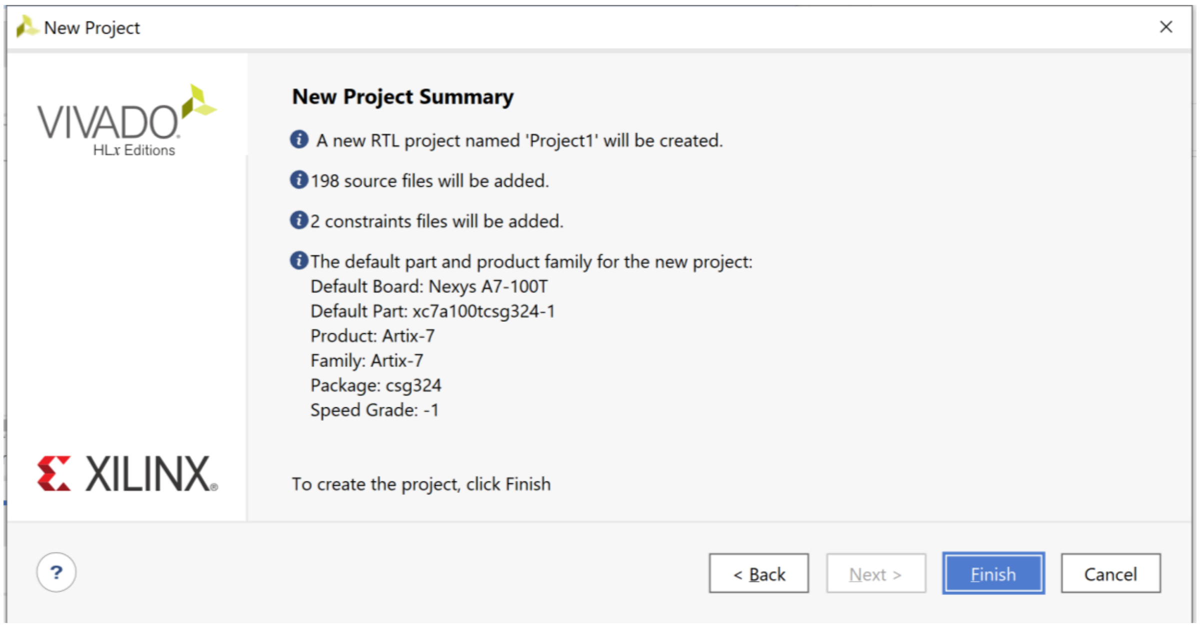
Vendor: All Name: All Board Rev: Latest

Search: nexys (5 matches)

Display Name	Preview	Vendor	File Version	Part	I/C
Nexys A7-100T		digilentinc.com	1.0	xc7a100tcsg324-1	32
Nexys A7-50T		digilentinc.com	1.0	xc7a50ticsg324-1L	32
Nexys4		digilentinc.com	1.1	xc7a100tcsg324-1	32
Nexys4 DDR		digilentinc.com	1.1	xc7a100tcsg324-1	32

Figure 7. Select target board: Nexys A7-100T

In the New Project Summary window, click Finish (see Figure 8).


New Project

New Project Summary

A new RTL project named 'Project1' will be created.

198 source files will be added.

2 constraints files will be added.

The default part and product family for the new project:
Default Board: Nexys A7-100T
Default Part: xc7a100tcsg324-1
Product: Artix-7
Family: Artix-7
Package: csg324
Speed Grade: -1

To create the project, click Finish

Figure 8. New Project Summary Window

Note that once the project completes being set up, it will indicate that files exist with Syntax Errors – this will be fixed in the next step.

Step 5. Set rvfpga as Top Module and common_defines.vh as global
The project will initialize. You will now set the rvfpga module as the top module. In the Sources pane, scroll down under Design Sources, right-click on the rvfpga module, and select Set as Top (see Figure 9). You can also find the rvfpga module by typing this name in the search box, as shown. This sets rvfpga as the highest-level module in the hierarchy and the target to be synthesized and implemented onto the FPGA. After setting rvfpga as the top module, the hierarchy will update.

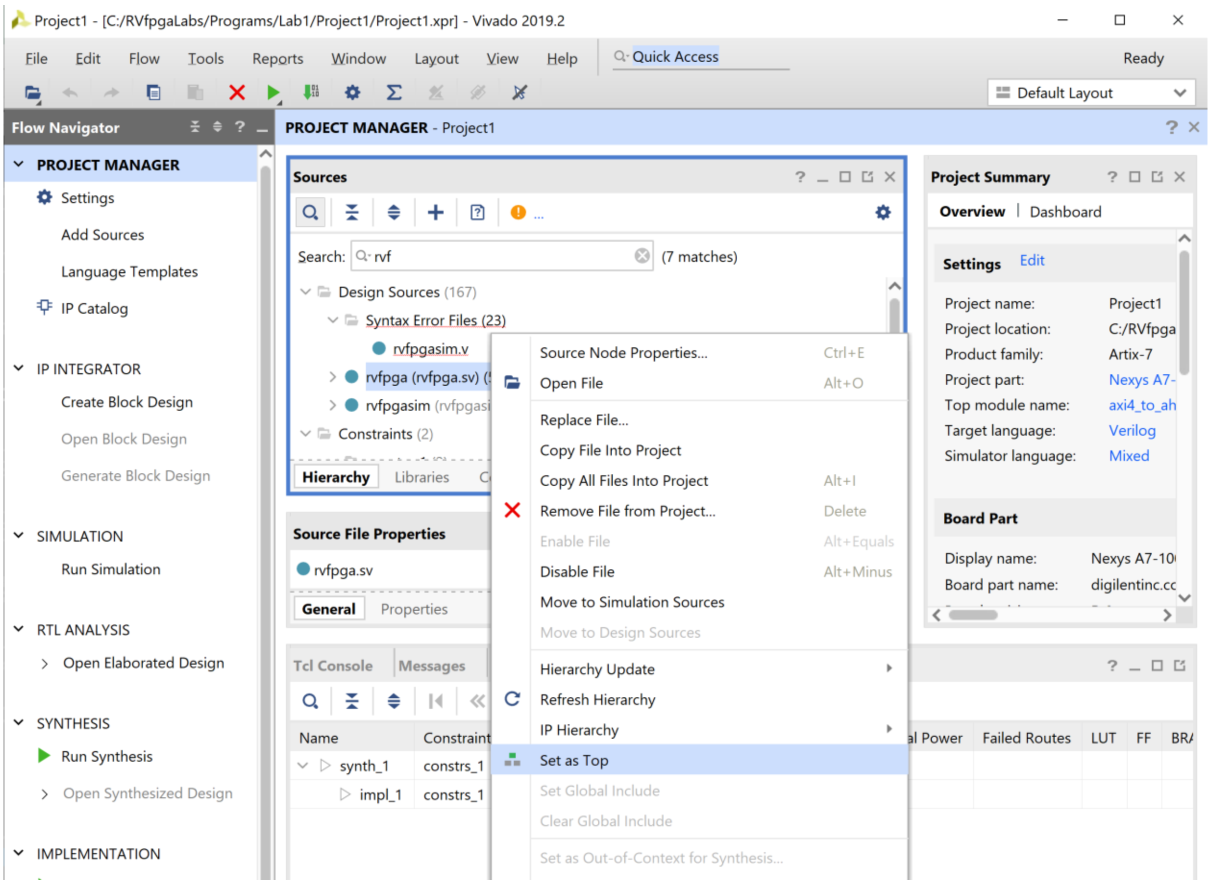
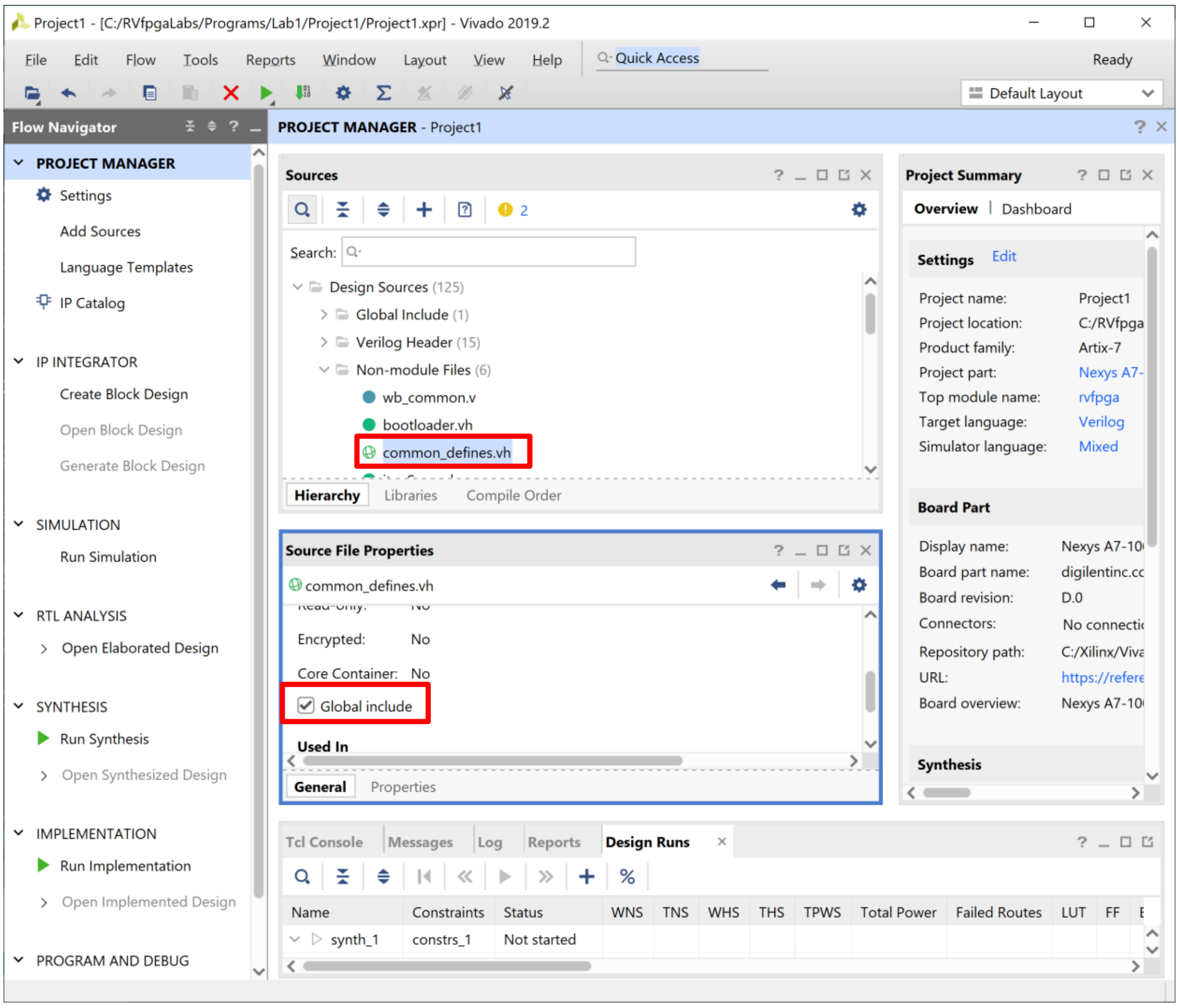


Figure 9. Set rvfpga as top module

Now, still in the Sources pane under Design Sources, expand the Non-modules file group and click on common_defines.vh. The properties of the file will then open in the Source File Properties pane, just below the Sources pane. Click on Global Include to tick that box (see Figure 10). The hierarchy will now update and include that file in Design Sources/Global Include. Note that the Syntax Error Files will disappear in a future step.



Encrypted: No

Core Container: No

☑ Global include

Used In

Project name:	Project1
Project location:	C:/RVfpga
Product family:	Artix-7
Project part:	Nexys A7-
Top module name:	rvfpga
Target language:	Verilog
Simulator language:	Mixed

Display name:	Nexys A7-10
Board part name:	digilentinc.cc
Board revision:	D.0
Connectors:	No connecti
Repository path:	C:/Xilinx/Viva
URL:	https://refere
Board overview:	Nexys A7-10

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF
synth_1	constrs_1	Not started									

Figure 10. Set common_defines.vh as a Global include file

Step 6. Generate Bitstream

Now Click on Flow → Generate Bitstream as shown in Figure 11. A window might pop up that says there are no implementation results available and ask to launch synthesis and implementation (see Figure 12). Click Yes. Then click OK in the Launch Runs window (see Figure 13). This step synthesizes RVfpga (as defined by the Verilog and SystemVerilog files in the project), maps it onto the FPGA, and creates the bitstream. This process typically takes 20-50 minutes, depending on the speed of your computer.

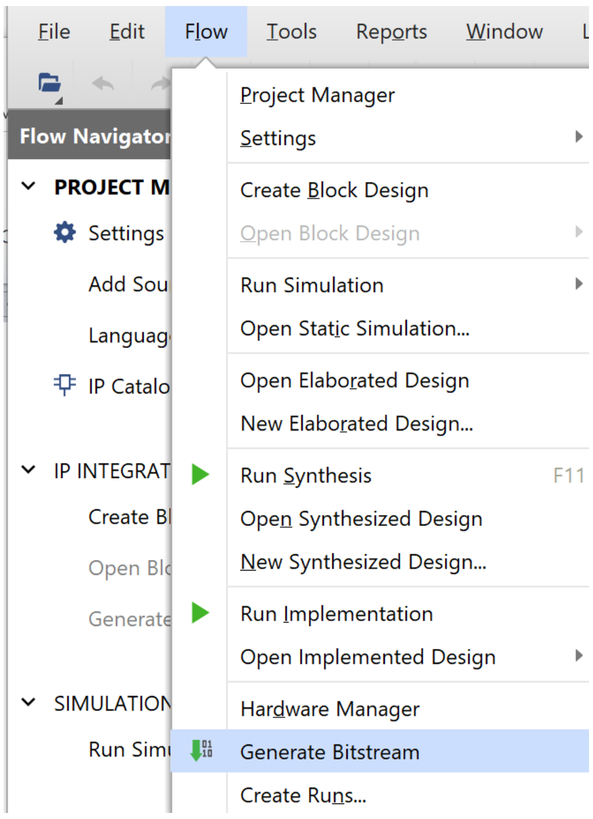
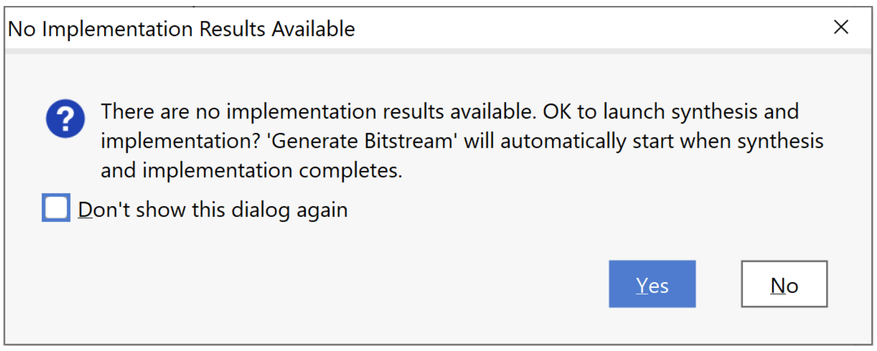


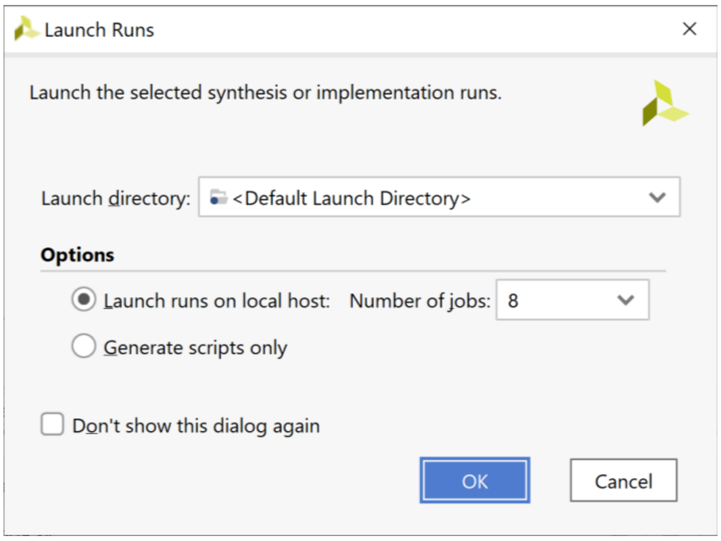
Figure 11. Generate Bitstream


No Implementation Results Available

There are no implementation results available. OK to launch synthesis and implementation? 'Generate Bitstream' will automatically start when synthesis and implementation completes.

☐ Don't show this dialog again

Figure 12. Launch synthesis and implementation window


Launch Runs

Launch the selected synthesis or implementation runs.

Launch directory: <Default Launch Directory>

Options

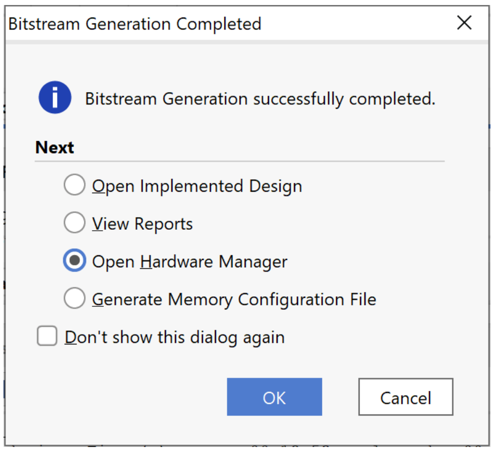
Launch runs on local host: [✓] Number of jobs: 8

Generate scripts only

Don't show this dialog again

Figure 13. Launch Runs

After the bitstream has been generated, a window will pop up as shown in Figure 14. Click on the ✕ button in the top-right corner to close the window.


Bitstream Generation Completed

Bitstream Generation successfully completed.

Next

Open Implemented Design

View Reports

Open Hardware Manager [✓]

Generate Memory Configuration File

Don't show this dialog again

Figure 14. Bitstream Generation Completed

Now that you have built the RVfpga system yourself, you will be able to rebuild RVfpga after you make modifications to it in Labs 6-10. For now, you will begin using the RVfpga system you just built to download and run programs on it using PlatformIO.

It is recommended that you use PlatformIO to download RVfpga onto the Nexys A7 board. This method was described in Section 6.A of the RVfpga Getting Started Guide (GSG) in detail. As also described in the GSG for the different examples (6.B to 6.H), after downloading the RVfpga system onto the FPGA on the Nexys A7 board, you will use PlatformIO to download and run/debug programs on RVfpga.

You can also use Verilator, an HDL simulator, to simulate programs running on RVfpga, as described in Section 7 of the RVfpga Getting Started Guide. These RTL-level simulations allow you to view low-level hardware signals as the software program runs. We will rely on Verilator heavily in Labs 6-10 as you extend RVfpga and test and debug your changes.