



Imagination大学计划

RVfpga实验1

创建Vivado项目

1. 简介

要使用和修改RVfpga系统，需要先构建一个项目，其中包括定义系统的所有Verilog文件、SystemVerilog文件、头文件、配置文件和文本文件。本实验将展示如何创建一个以Digilent的Nexys A7 FPGA开发板（100T版本，即RVfpgaNexys）作为本课程中所用SoC的目标的Vivado项目。（请记住，如果已有Nexys4 DDR开发板，也可以使用该开发板。）通过执行这些相同的步骤，即可修改RVfpgaNexys并重新加以综合。

重要信息：在开始RVfpga实验之前，必须完成Imagination大学计划（<https://university.imgtec.com/>）提供的“RVfpga入门指南”中所述的准备工作。

例如，必须依据“RVfpga入门指南”中的说明安装Xilinx的Vivado和Verilator（如果尚未安装）。此外，请确保已将从Imagination大学计划下载的RVfpga文件夹复制到您的计算机上。我们将RVfpga文件夹所在目录的绝对路径称为[RVfpgaPath]。[RVfpgaPath]/RVfpga/src文件夹包含RVfpga系统（即，我们将在所有实验中使用和修改的RISC-V SoC）的Verilog和SystemVerilog源文件。[RVfpgaPath]/RVfpga/Labs文件夹包含实验1至20中将会使用的一些程序。

2. 为RVfpgaNexys创建Vivado项目

您将使用Xilinx的Vivado设计套件¹通过RTL（定义系统的Verilog文件）构建RVfpgaNexys系统。请按照以下的详细步骤来构建RVfpgaNexys系统，并指定Nexys A7 FPGA作为目标板。

步骤1. 打开Vivado

步骤2. 新建RTL项目

步骤3. 添加RTL源文件和约束文件

步骤4. 选择Nexys A7作为目标板

步骤5. 将rvfpganexys设为顶层模块，common_defines.vh设为全局文件

步骤6. 生成比特流

步骤1. 打开Vivado

如果尚未按照“RVfpga入门指南”的说明在计算机上安装Vivado，请立即进行安装。务必还要安装开发板文件。

然后，运行Vivado（在Linux中，打开一个终端并输入：vivado；在Windows中，通过“Start”（开始）菜单打开Vivado）。Vivado欢迎界面随即打开。单击“Create Project”（创建项目）（参见图1）。

¹ 在这些材料中，我们均使用 Vivado 2019.2。尽管大多数内容在版本更高的 Vivado 中也适用，但我们强烈建议使用此版本。

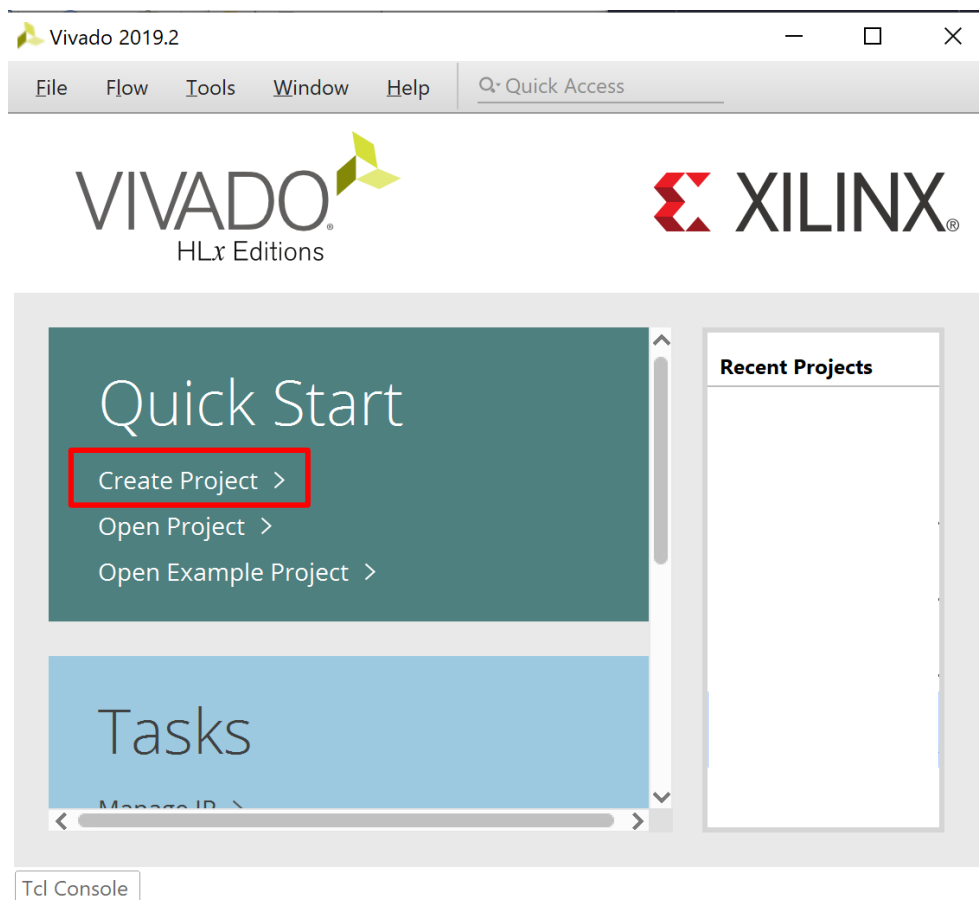


图1. Vivado欢迎界面：“Create Project”（创建项目）

步骤2. 新建RTL项目

“Create a New Vivado Project”（创建一个新的Vivado项目）向导随即打开（参见图2）。单击“Next”（下一步）。

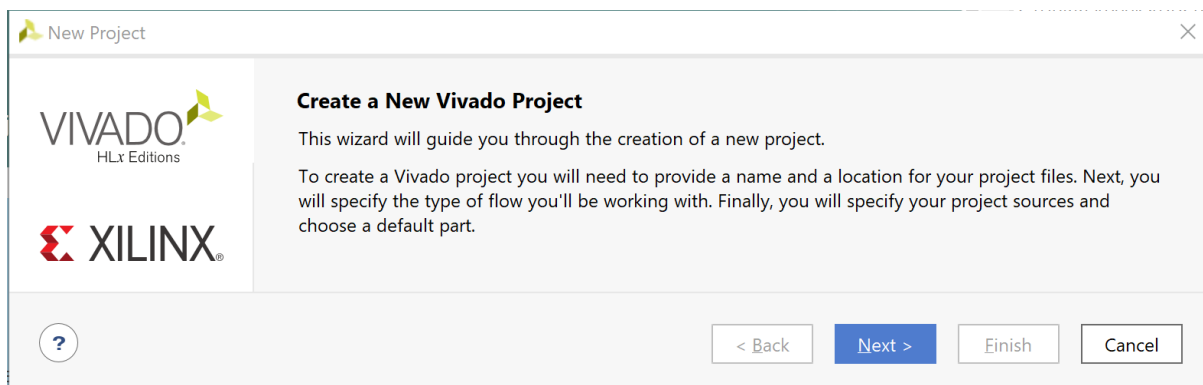


图2. “Create a New Vivado Project”（新建Vivado项目）向导

调用项目名称“Project1”并将其放到`[RVfpgaPath]/RVfpga/Labs/Lab1`文件夹中。选择“*Create project subdirectory*”（创建项目子目录）选项。然后单击“Next”（下一步）（参见图3）。

Project Name

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

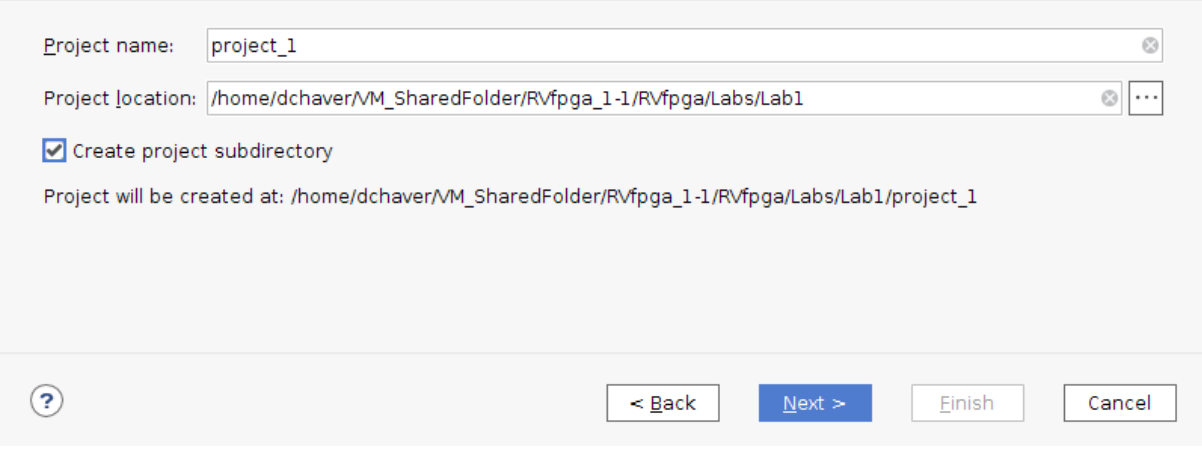



图3. 项目名称

选择项目类型“RTL Project”（RTL项目），然后单击“Next”（下一步）（参见图4）。

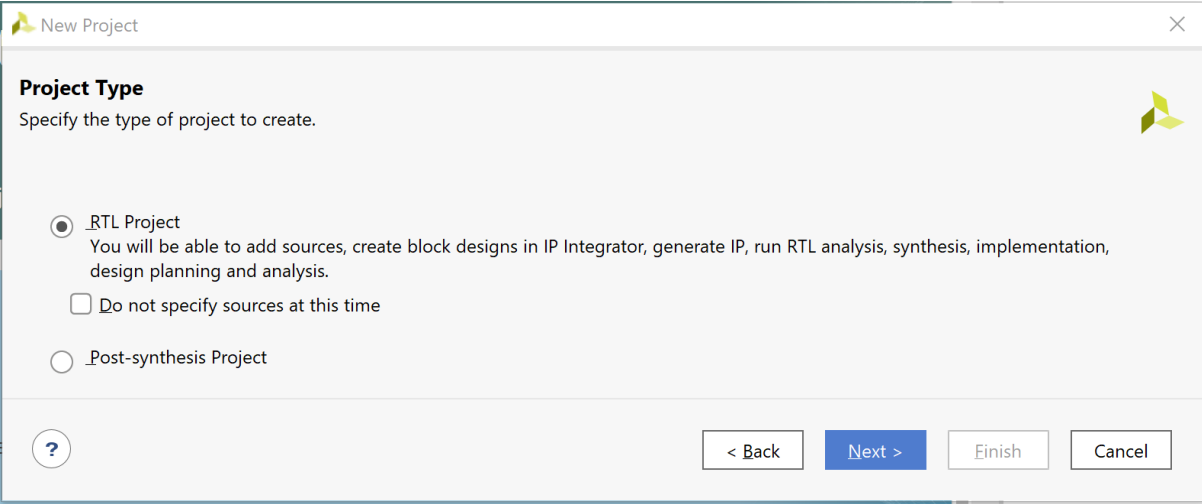


图4. RTL项目

步骤3. 添加RTL源文件和约束文件

在“Add Sources”（添加源文件）窗口中，单击“Add Directories”（添加目录），然后选择`[RVfpgaPath]/RVfpga/src`（参见图5）。确保同时选中以下两个选项（如图5所示）：

- “Scan and add RTL include files into project”（扫描RTL包含文件并将其添加到项目中）
- “Add sources from subdirectories”（从子目录添加源文件）

然后单击“Next”（下一步）。

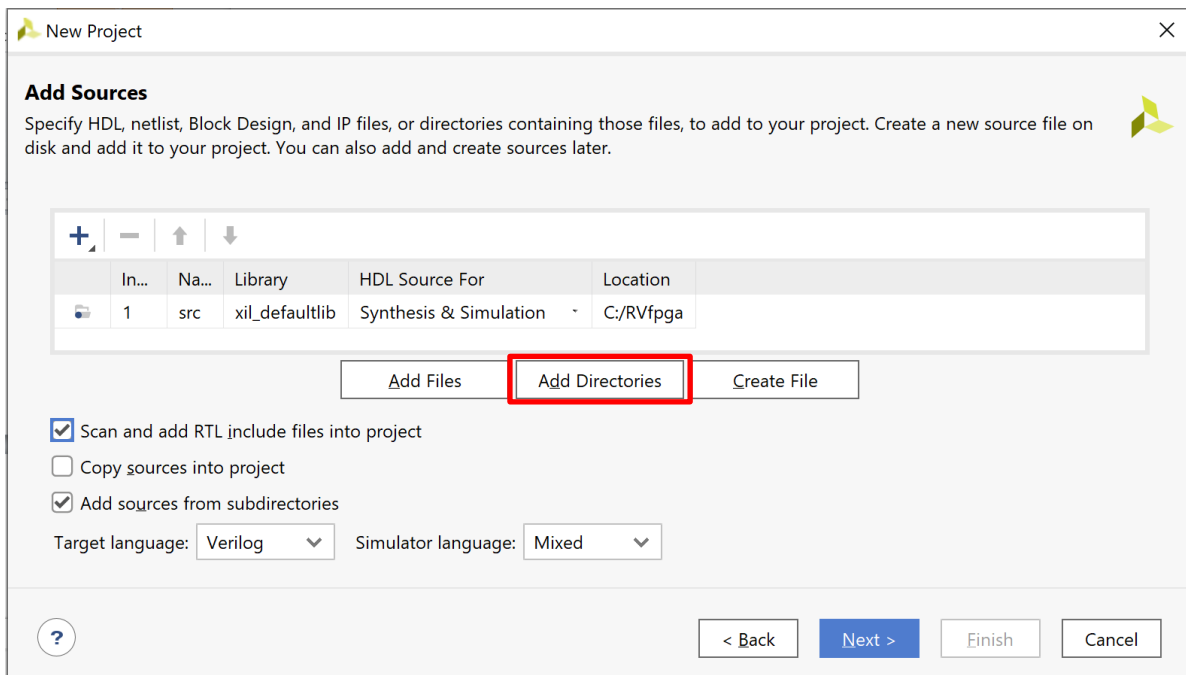


图5. 添加源文件

现在将为系统添加约束文件。这些文件将信号名称映射到开发板上的引脚。例如，Nexys A7 FPGA开发板上的LED通过PCB中的走线与开发板上的FPGA引脚相连。Vivado必须知道这些信息，以便将RTL中的信号名称正确地映射到相应的FPGA引脚。例如，`[RVfpgaPath]/RVfpga/src/rvfpganexys.xdc`文件（一种Xilinx的设计约束文件）中的以下行表示FPGA引脚H17映射到LED的最低有效位（`o_led[0]`），并使用LVCMOS 3.3V信号：

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

请注意，信号名称`o_led`是Verilog代码中用于驱动Nexys A7开发板LED的名称。

在“Add Constraints”（添加约束文件）窗口中，单击“Add Files”（添加文件），然后选择以下两个文件（参见图6）：

```
[RVfpgaPath]/RVfpga/src/rvfpganexys.xdc
[RVfpgaPath]/RVfpga/src/LiteDRAM/liteDRAM.xdc
```

然后单击“Next”（下一步）。

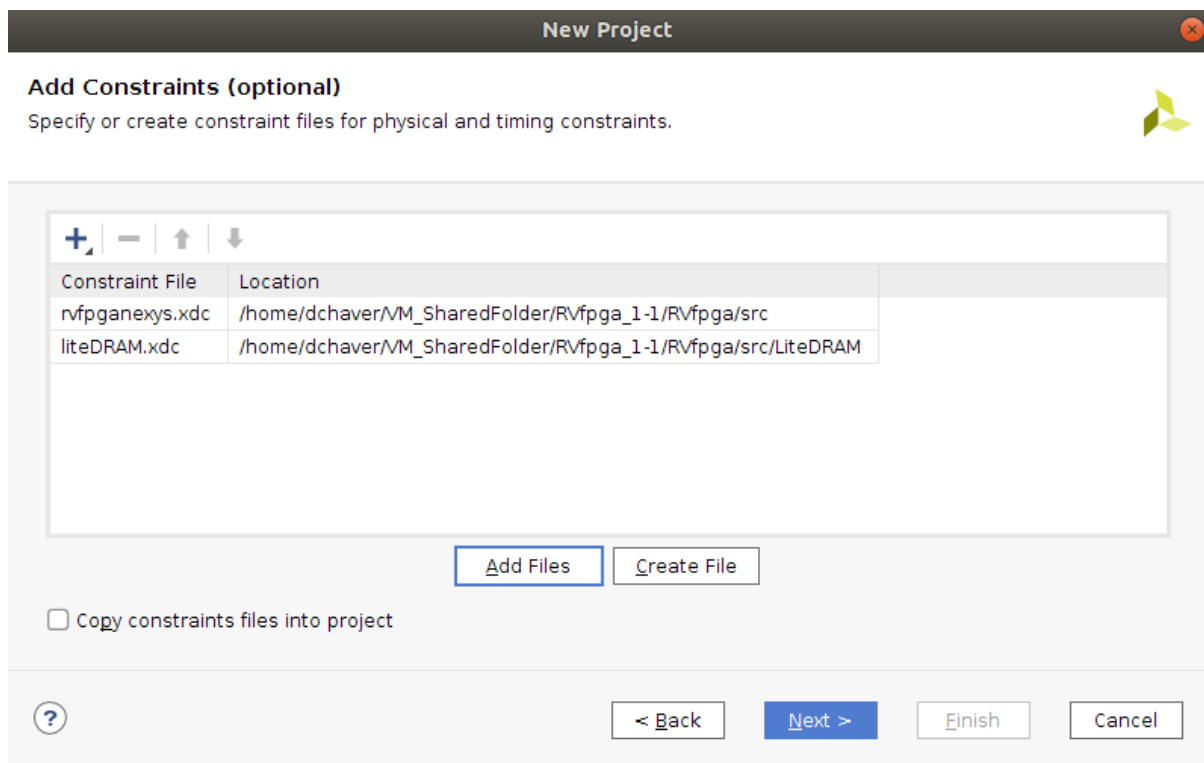


图6. 添加约束文件

步骤4. 选择Nexys A7作为目标板

在“Default Part”（默认部件）窗口中，单击“Boards”（开发板），然后选择“Nexys A7-100T”（参见图7）。可以使用“Search”（搜索）框缩小结果范围。您还会注意到实际目标FPGA的名称在“Part”（部件）列中列出：xc7a100tcsg324-1。这表明它是一个Xilinx Artix-7 FPGA，具有100k等效门并采用324引脚CSG（芯片级网格）封装。

单击“Next”（下一步）。

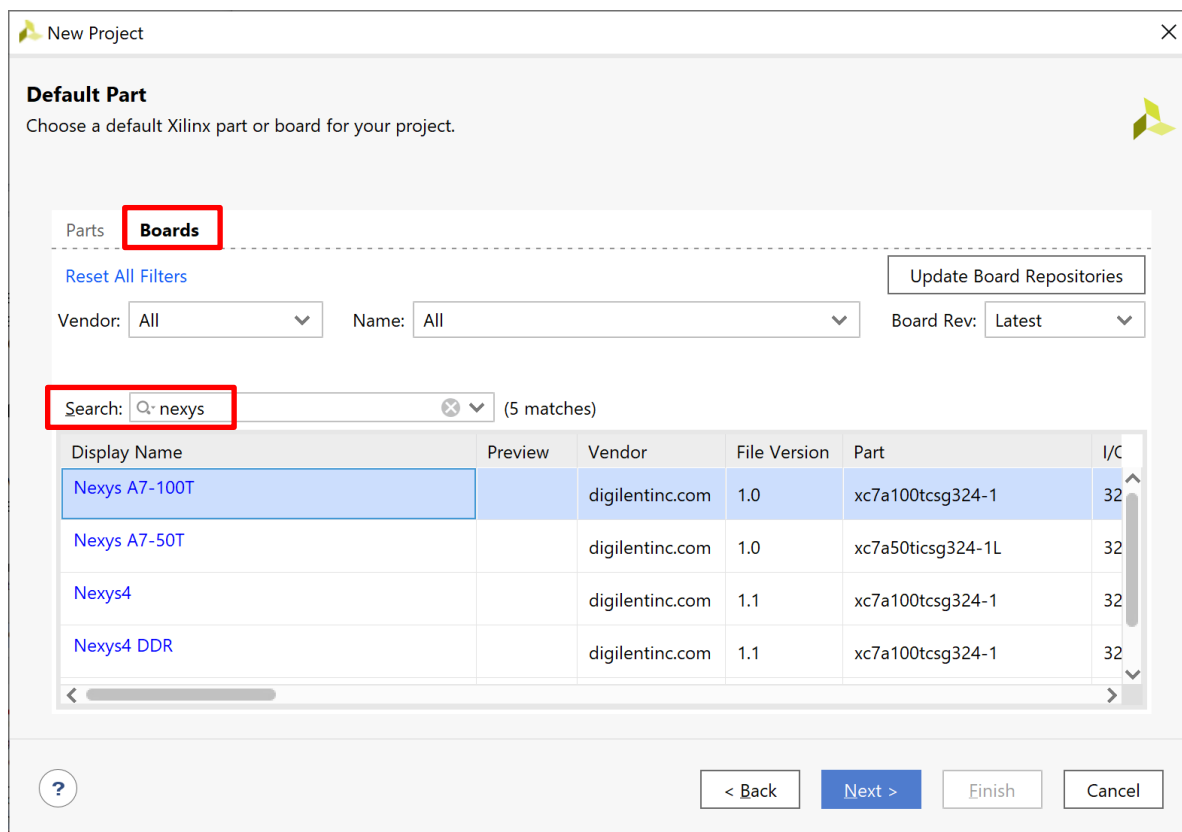


图7. 选择开发板：Nexys A7-100T

在“New Project Summary”（新项目摘要）窗口中，单击“Finish”（完成）（参见图8）。

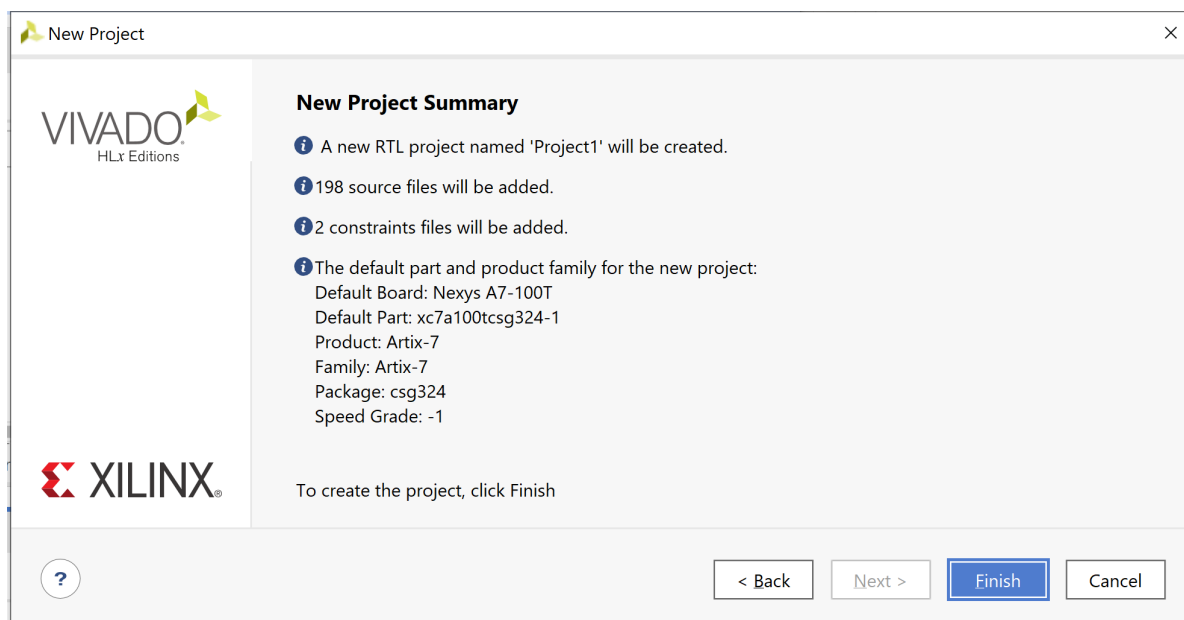


图8. “New Project Summary”（新建项目摘要）窗口

请注意，一旦项目完成设置，将会指示存在语法错误的文件 - 这将在下一步中修复。

步骤5. 项目配置：将rvfpganexys设为顶层模块，将文件**common_defines.vh**设为全局文件，将**boot_main.mem**添加到项目中，并包含Pulp平台文件夹

将rvfpganexys设为顶层模块：接下来将rvfpganexys模块设为顶层模块。在“Sources”（源文件）窗格中的“Design Sources”（设计源文件）下向下滚动，右键单击rvfpganexys模块，然后选择“Set as Top”（设为顶层模块）（参见图9）。也可以通过在搜索框中输入相应名称来查找rvfpganexys模块，如下图所示。这会将rvfpganexys设为层级中级别最高的模块，并将其作为要在FPGA上合成并实现的目标。将rvfpganexys设为顶层模块后，层级将更新。

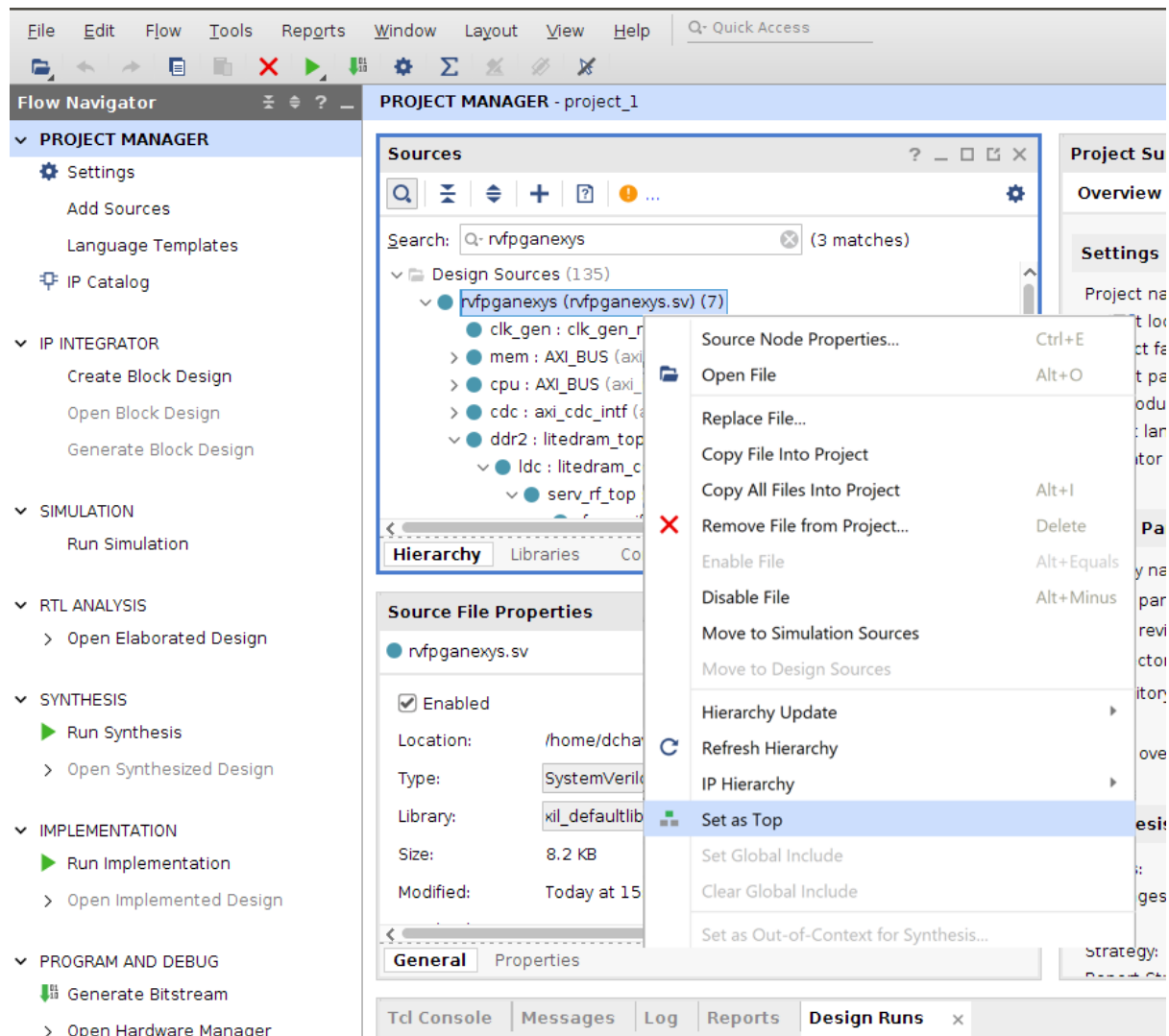


图9. 将rvfpganexys设为顶层模块

将文件common_defines.vh设为全局包含文件：然后，还是在“Design Sources”（设计源文件）下的“Sources”（源文件）窗格中，展开“Non-modules”（非模块）文件组，然后单击common_defines.vh。随后，该文件的属性将在“Sources”（源文件）窗格正下方的“Source File Properties”（源文件属性）窗格中打开。单击选中“Global Include”（全局包含文件）复选框（参见图10）。层级现在将更新并将该文件包含在“Design Sources”（设计源文件）/“Global Include”（全局包含）中。请注意，语法错误文件将不会出现在后续步骤中。

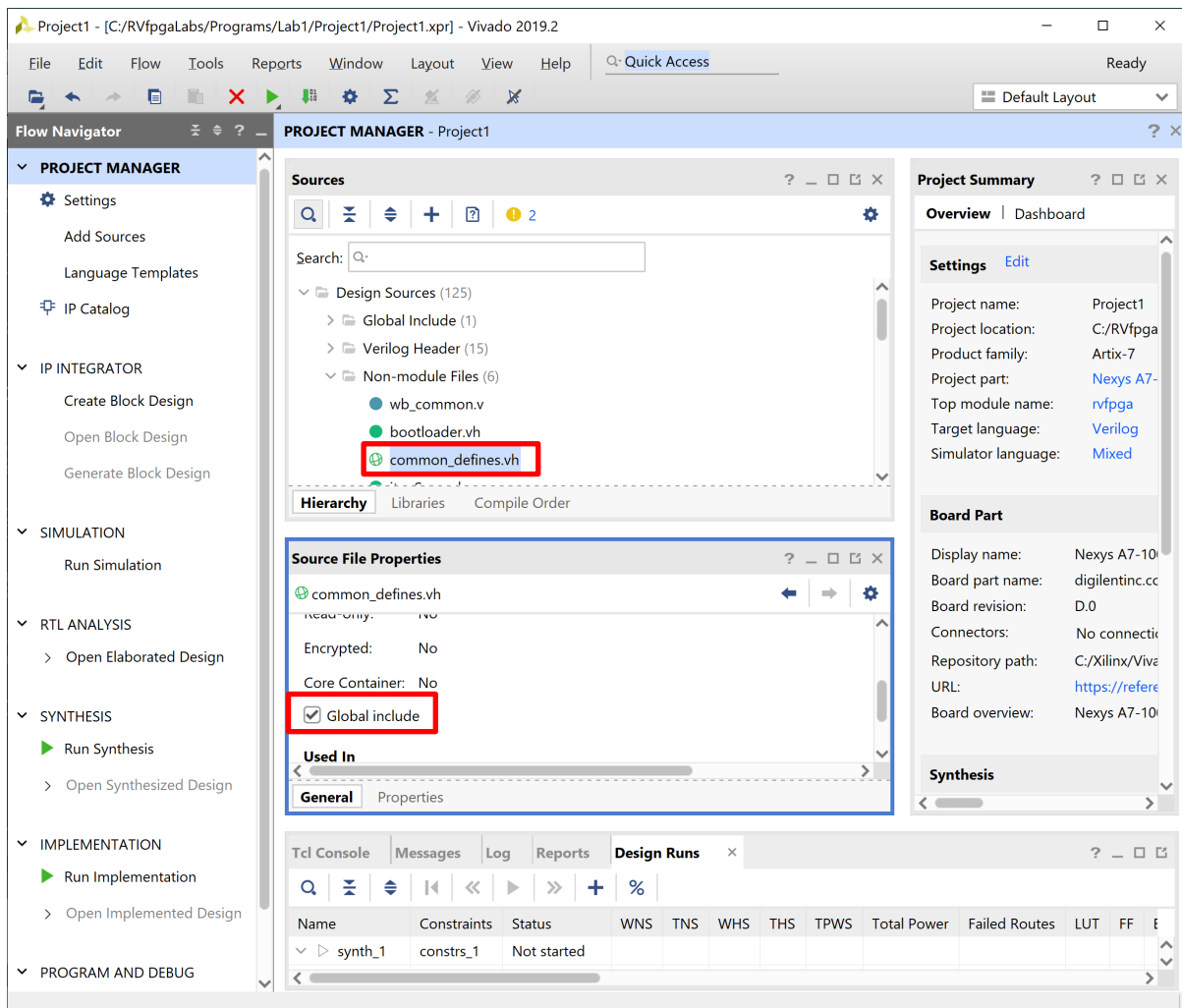


图10. 将common_defines.vh设置为全局包含文件

将boot_main.mem添加到项目中：在“Flow Navigator”（流程导航器）窗格中，单击“Add Sources”（添加源文件），保持默认选项（“Add or create design sources”（添加或创建设计源文件）），然后单击“Add Files”（添加文件）（参见图11）。导航到[RVfpgaPath]/RVfpga/src/SweRVolfSoC/BootROM/sw 并选择boot_main.mem（如图11所示）。层级随即更新，并将该文件包含在“Design Sources”（设计源文件）/“Memory File”（存储器文件）中。

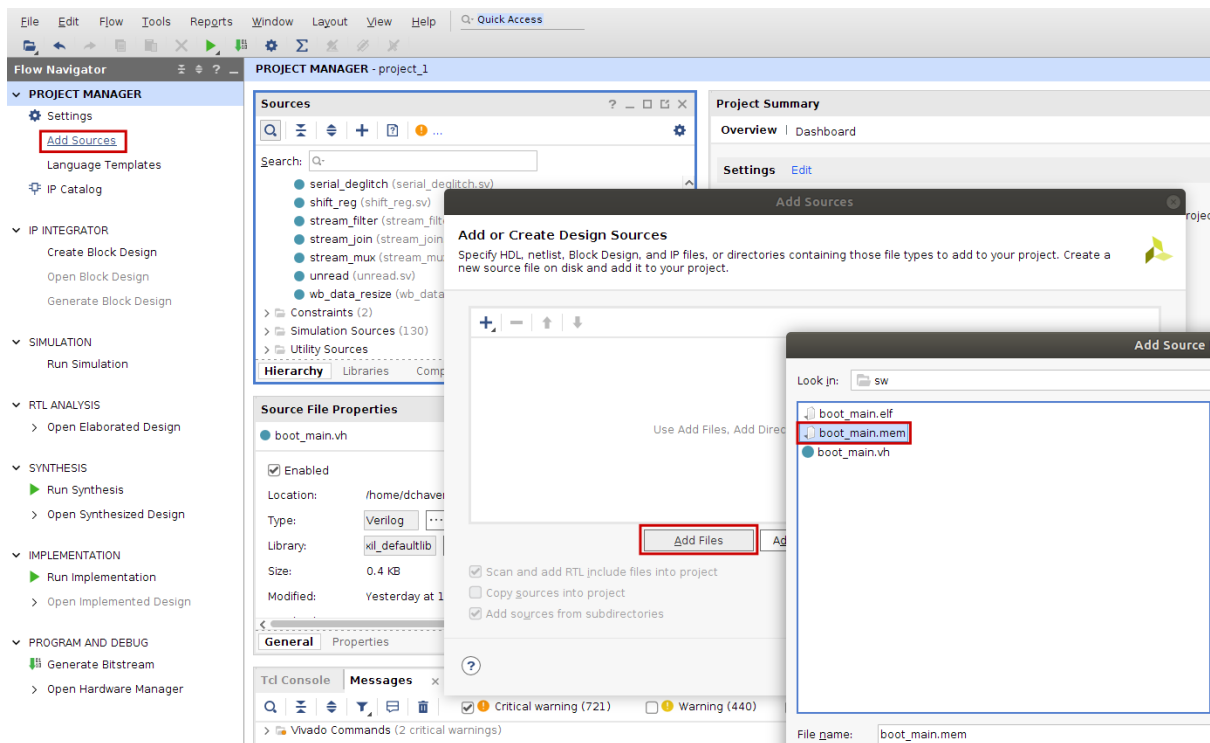




图11. 添加存储器文件boot_main.mem

该文件（*boot_main.mem*）用于初始化我们的SoC的引导ROM，方法是在文件[RVfpgaPath]/RVfpga/src/rvfpganexys.rv中将其作为参数调用：

```
25 module rvfpga
26   #(parameter bootrom_file = "boot_main.mem")
```

入门指南中的第6.A节包含该文件的更多信息。

包含文件夹：最后，包含Pulp平台的两个文件夹（参见图12）。在“Flow Navigator”（流程导航器）窗格中单击“**Settings**”（设置），然后在打开的窗口中依次单击“**General**”（常规）和“**Verilog options**”（Verilog选项）（）。在新窗口中，单击  并浏览到相应目录，添加以下两个包含目录：

```
[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Interconnect/AxiInterconnect/pulp-platform.org__axi_0.25.0/include
[RVfpgaPath]/RVfpga/src/OtherSources/pulp-platform.org__common_cells_1.20.0/include
```

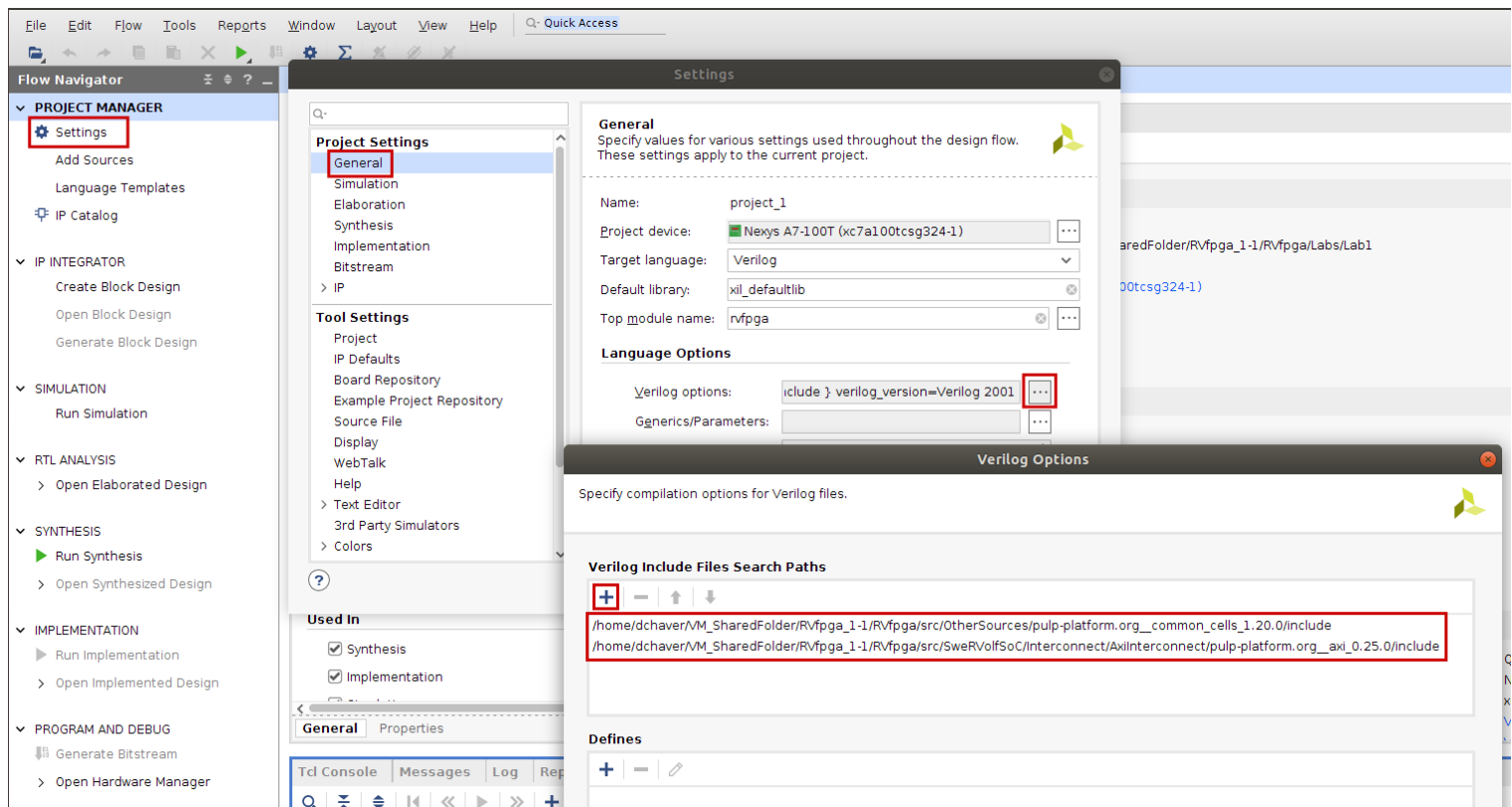


图12. 包含Pulp平台的文件夹

步骤6. 生成比特流

接下来单击“Flow”（流）→“Generate Bitstream”（生成比特流），如图13所示。此时可能会弹出一个窗口，提示没有可用的实现结果，并要求启动综合和实现（参见图14）。单击“**Yes**”（是）。然后在“Launch Runs”（启动运行）窗口中单击“**OK**”（确定）（参见图15）。此步骤会对RVfpgaNexys进行综合（根据项目中Verilog和SystemVerilog文件的定义）、将其映射到FPGA并创建比特流。此过程通常需要20-50分钟，具体取决于计算机的速度。

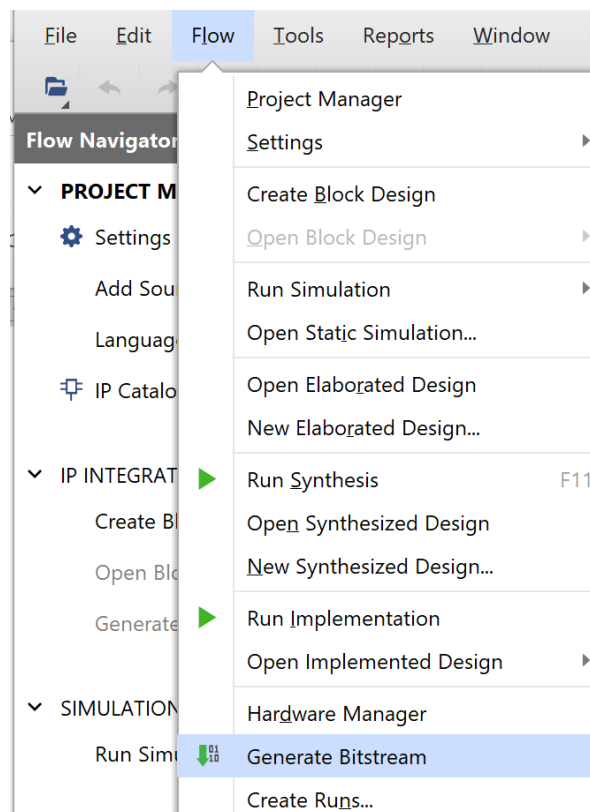


图13. 生成比特流

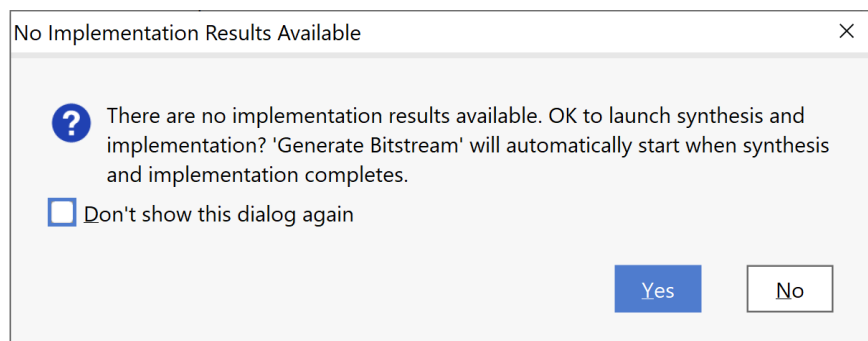


图14. 启动综合和实现窗口

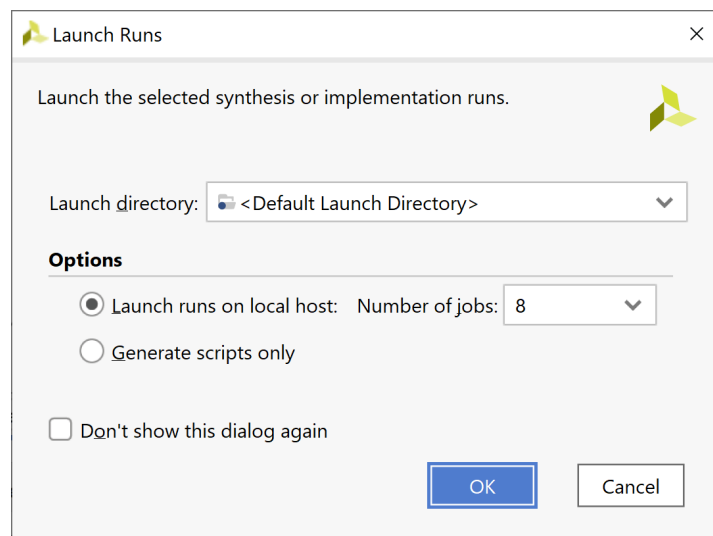


图15. 启动运行

生成比特流后，将弹出一个窗口，如图16所示。单击右上角的 **X** 按钮关闭该窗口。

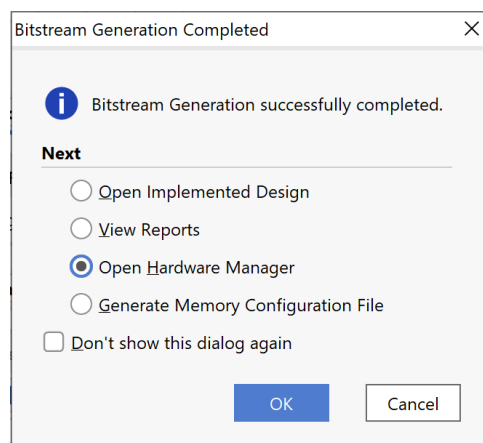


图16. 已生成比特流

现在您已自行编译了RVfpgaNexys系统，在实验6-10中对其进行修改之后，可以重新编译RVfpgaNexys。随后需要使用刚刚编译的RVfpgaNexys系统，利用PlatformIO在系统中执行下载并运行程序。

建议使用PlatformIO将RVfpgaNexys下载到Nexys A7开发板。有关此方法的详细说明，请参见“RVfpga入门指南（GSG）”的第6.A节。GSG中的其他示例（6.B至6.H）也显示，将RVfpgaNexys系统下载到Nexys A7开发板上的FPGA之后，将使用PlatformIO在RVfpgaNexys上下载和运行/调试程序。

也可使用Verilator（一种HDL仿真器）来仿真RVfpgaSim上运行的程序，如“RVfpga入门指南”第7节所述。借助这些RTL级仿真，可以在软件程序运行时查看底层硬件信号。对于实验6-10，在扩展RVfpga系统以及测试和调试更改时，我们将很大程度依赖Verilator。