目录

第一章 硬件实验平台及 FPGA 设计流程

这一讲我们的学习任务主要有 2 个：

1. 了解本学期实验所使用的硬件实验平台。
2. 熟练掌握基于 Xilinx Vivado 集成设计环境，以图形界面下操作的 Project 方式，完成 RTL 到比特流文件的 FPGA 设计流程。

1 体系结构课程硬件实验平台

在本学期的课程中，我们统一使用“龙芯体系结构教学实验箱”（以下简称“实验箱”）作为硬件实验平台。整个实验平台采用本地使用的方式，即你的设计通过 JTAG 线缆直接下载到实验箱的 FPGA 中，同时你可以直接操作实验箱内 FPGA 开发板上的外设。

1.1 实验箱简介

打开实验箱（贴有号码牌的是上面），其内部包含一块 FPGA 开发板（图 1-1 中 A）和一系列配件和电缆，包括 1 个电源适配器（图 1-1 中 B）、1 根连接 FPGA 下载适配器的 USB 线缆（图 1-1 中 C）、1 根串口线（图 1-1 中 D）、1 个 USB 转串口接头（图 1-1 中 E），部分实验箱中还有 1 根 USB 延长线（图 1-1 中 F）和 1 根网线（图 1-1 中 G）。

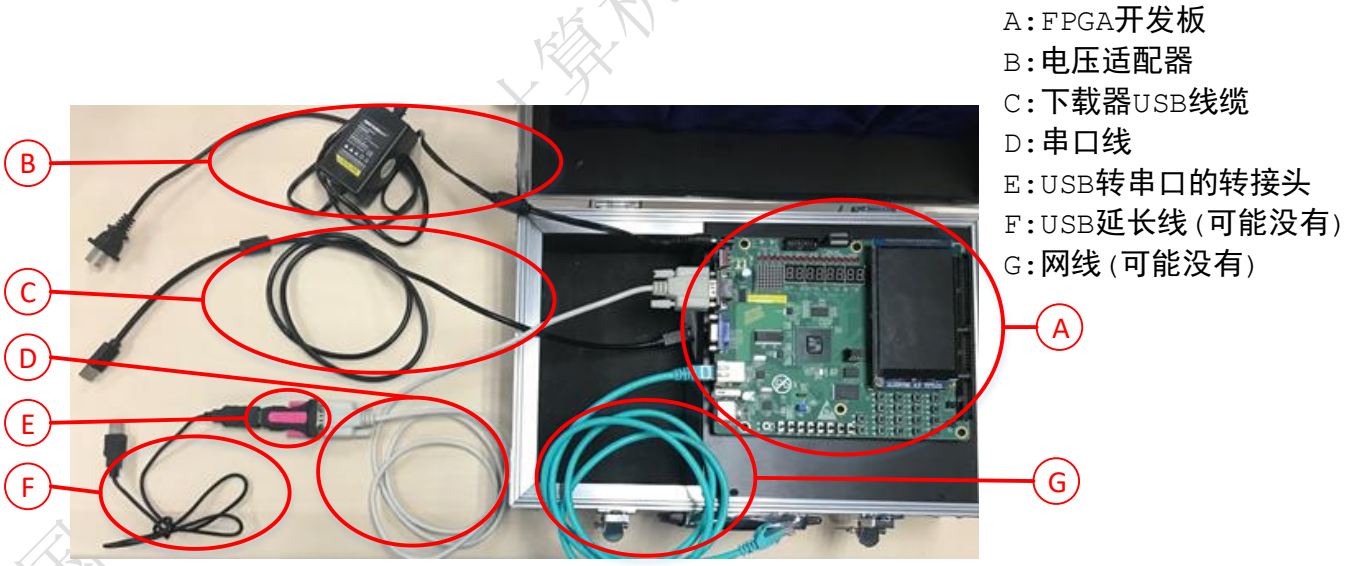


图 1-1 教学实验箱总体视图

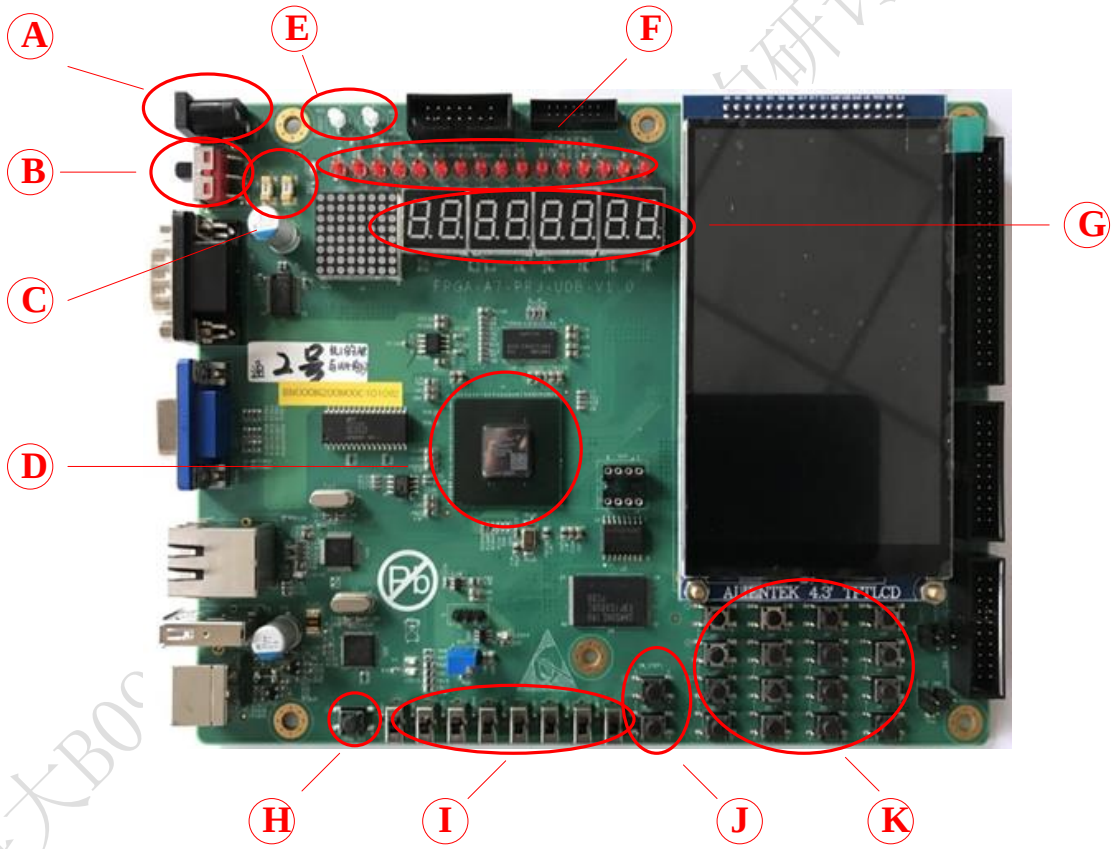
实验箱中的各种设备、线缆，本学期本课程的实验中仅需要各位使用电源适配器（图 1-1 中 B）和 FPGA 下载适配器的 USB 线缆（图 1-1 中 C）。其余线缆建议各位在平时保持在学期初大家看到的收纳状态，以防丢失。

当需要将综合好的比特流文件下载到 FPGA 实验板进行调试时，请先将电源适配器的直流接口

插入 FPGA 开发板上的电源插口（图 1-2 中 A），并将 FPGA 下载适配器的 USB 线缆的方口插入 FPGA 开发板左侧下方的下载适配器接口（图 1-1 中 G）中，将该线缆的 USB 口连接到调试主机电脑上，随后拨动 FPGA 开发板上的电源开关（图 1-2 中 B），正常情况下可见 FPGA 开发板上电源指示灯（图 1-2 中 C）亮起，表明 FPGA 开发板已经上电可以进行后续操作。

FPGA 开发板上的核心器件是中央偏左的 FPGA 芯片（图 1-2 中 D），这里选用的是一款 Xilinx 公司最新的 Airtx-7 系列 FPGA 芯片，具体型号为 XC7A200T-FBG676，其内部逻辑单元数目多，芯片引脚数目多，属于 Airtx-7 系列中的高端产品。

不同于多数公版的 Xilinx FPGA 开发板，实验箱中所集成的 FPGA 开发板针对数字电路、组成原理、体系结构、操作系统等课程的实验教学需求，集成了相当丰富的外设，这些占据了开发板的大部分面积。这里仅介绍与本课程实验相关的外设接口，具体有：双色 LED 灯（图 1-2 中 E）、单



色 LED 灯（图 1-2 中 F）、数码管（图 1-2 中 G）、FPGA 复位按键（图 1-2 中 H）、拨码开关（图 1-2 中 I）、脉冲开关（图 1-2 中 J）和 4×4 键盘（图 1-2 中 K）。同学们若是对其它接口感兴趣，可以参考讲义的附录一“龙芯体系结构教学实验箱介绍”。

图 1-2 FPGA 开发板顶视图

1.2 实验箱使用小贴士

根据往届同学们的使用情况，总结一些实验箱使用的建议如下：

1. 每次用完实验箱之后，请将配件线缆收纳整齐后，再缓慢合上实验箱盖，若感觉阻力较大，请打开箱盖理顺配件线缆再行尝试，强行关闭可能会损坏实验设备。
2. 实验箱仅在你功能仿真验证通过且顺利生成出比特流文件之后，进行上板调试的时候才需要。在此之前，不需要通电并连接到你的电脑上。
3. 使用实验箱时，请将其置于一个平整、稳定且有足够接触面积的地方，大腿上、书包上、桌子角等地方均不大合适。
4. 使用时，确保 FPGA 板已上电、确保 FPGA 板已上电、确保 FPGA 板已上电，重要的事情说三遍。
5. 绝对不要用导体（导线、潮湿的手、茶或咖啡……）连接 FPGA 上任何裸露的引脚、插针。

2 FPGA 设计流程

FPGA（Field Programmable Gate Array，现场可编程门阵列）是一种特殊的集成电路。这种特殊性体现在它的电路功能在芯片制造出来以后，可以通过编程配置进行调整，而传统的专用集成电路（Application Spedific Integrated Circuits，ASIC）则不具备这种特性。打个比方来说，传统的专用集成电路芯片设计就像是在一张纸上面画画，画成什么样就没办法再修改了，而 FPGA 芯片设计就像是在黑板上画画，画了以后觉得不合适可以擦掉重新画。FPGA 的这种可编程的特点，为我们开展数字电路、组成原理、体系结构这样的实验课提供了绝佳的硬件平台。首先，它是一个实实在在的集成电路芯片，不是仿真软件下电路行为的模拟，让硬件实验课能够真正“硬”起来，让同学们的实践经历更加贴合工业界的实际生产研制过程。其次，FPGA 上调整设计不需要重新流片重新设计焊接 PCB，只需要调整设计代码后重新运行一遍 FPGA 的综合实现流程就可以，短到几分钟长不过数天就可以获得一个功能调整后的芯片进行调试，省时又省钱。

各位同学在数字电路和组成原理的实验课上已经多次使用 FPGA，想必对 FPGA 设计流程其实已不陌生。不过请稍安勿躁，咱们温故知新，给大家再复习一遍 FPGA 的设计流程，希望全员都能熟练掌握。

2.1 FPGA 的一般设计流程

FPGA 是一种特殊的集成电路，意味着它首先是一种集成电路。现在的集成电路绝大多数都是晶体管集成电路，其中大家日常接触最多的是 CMOS 晶体管集成电路。晶体管集成电路是什么？通俗一点来说，就是用金属导线把许许多多由晶体管构成的逻辑门、存储单元连接成一个电路，具备

一定的逻辑功能。不过，各位同学从数字电路实验课开始，有进行用导线连晶体管的实验吗？显然没有。各位都是用一种叫 VerilogHDL 的硬件描述语言，写写代码，然后 Run 一下 Vivado 这个软件，电路就出来了。大家已知的这种流程，并不是课程教学实验中所独有的，它其实与现在工业界常见的 ASIC 设计流程是一致的。针对咱们实验课的具体情况，FPGA 的一般设计流程有“五步骤”，即

1. 电路设计
2. 代码编写
3. 功能仿真
4. 综合实现
5. 上板调试

2.1.1 电路设计

首先需要根据需求规格制定电路设计方案。例如，需求是设计一个 MIPS CPU，你需要把这个需求一步步分解细化，成为一个能够满足需求的电路设计。你决定了分成几个流水级，这里放几个触发器，那里放几个运算器，它们之间怎么连接，整个电路的状态转换行为是怎样的，等等。通常，我们将电路设计细化到寄存器传输级（Register Transfer Level，RTL）就可以了，无需精确到逻辑门级别或是晶体管级别。

2.1.2 代码编写

把第 1 步中完成的电路设计用硬件描述语言（Hardware Description Language，HDL）表述出来，成为一种 EDA 工具能够看得懂的形式。我们使用 VerilogHDL 语言。

2.1.3 功能仿真

对第 2 步中用 HDL 语言描述出的设计进行逻辑功能验证。所谓逻辑功能验证，就是看电路的逻辑功能行为是否符合最初的设计需求，通常我们给电路输入指定的激励，观察电路输出是否符合预期，如果不符合则表明电路逻辑功能有错误。这种错误要门是因为第 1 步的电路设计就有错误，要么是第 2 步编写的代码不符合电路设计，需要返回相应的步骤进行修改后再依流程顺序重新来一遍。如此不断迭代直到不再发现错误，就可以进入下一阶段了。

需要指出的是，由于我们对于电路是在 RTL 级建模，所以功能仿真阶段不考虑电路的延迟。

2.1.4 综合实现

综合实现这个阶段完成从 HDL 代码到真实芯片电路的转换过程。这个过程有点类似于编译器把高级语言转换成目标机器的二进制代码的过程。具体分为综合和实现两个子阶段。综合阶段将 HDL 描述的设计编译为由基本逻辑单元连接而成的逻辑网表，不过此时的网表还并不真实的门级电路。

实现阶段是将综合出的逻辑网表映射为 FPGA 中的具体电路，即将逻辑网表中的基本逻辑单元映射到 FPGA 芯片内部固有的硬件逻辑模块上（称为“布局”）；随后基于布局的拓扑，利用 FPGA 芯片内部的连线资源，将各个映射后的逻辑模块连接起来（称为“布线”）。

如果整个综合实现过程没有发生异常，EDA 工具将生成一个比特流（bitstream）文件。通俗来说，这个比特流文件描述的就是最终的电路，只不过这个文件 FPGA 芯片能读得懂。

2.1.5 上板调试

俗话说的好，“是骡子是马拉出来溜溜”。甭管功能仿真跑得多正确，最终还是要看实际电路能否正常工作。上板调试，首先要将综合实现生成的比特流文件下载到 FPGA 芯片中，随后就是运行电路观察其工作是否正常，如果发生问题就要调试定位出错的原因。

小结一下，上面总结的 FPGA 一般设计流程五大步骤，把握的是个总的脉络，为的是让同学们先建立一个正确的整体概念，否则实验过程中连自己处于哪个阶段都搞不清楚，怎么搞的清楚自己该干什么。FPGA 设计流程中还包含很多细节，这些细节在本课程实验中涉及到的，我们会在后续的课程中陆续介绍，以避免大家一时间难以全部消化吸收。FPGA 设计流程中其实还有一些步骤，因为在本课程实验中不涉及，就没有列举在上面，同学们在今后的工作中根据实际需要再行学习吧。

2.2 基于 Vivado 的 FPGA 设计流程

前面提到的 FPGA 的一般设计流程中，“功能仿真”、“综合实现”和“上板调试”这三个步骤都要使用 EDA 工具。我们的硬件实验平台选用的是 Xilinx 公司的 FPGA 芯片，也就很自然地会使用 Xilinx 公司提供的 Vivado 集成设计环境。尽管说 Vivado 这个软件中的功能仿真和波形调试也就是个将将能用的水平，但考虑到同学们为了完成各门实验课电脑上装的软件已经够多了，能少装一个软件就少装一个吧。咱们设计的 CPU 也小，Vivado 凑合着也能用。

Vivado 针对 FPGA 设计提供了两种工作方式：Project 方式和 Non-Project 方式。其中 Project 方式可以在 Vivado 的图形界面下操作或以 Tcl 脚本方式在 Vivado Tcl Shell 中运行；而 Non-Project 方式只能以 Tcl 脚本方式运行，而且该脚本和 Project 方式下的脚本命令是不同的。本课程中采用 Project 方式，且以图形界面进行操作。相较于 Tcl 脚本这种适合于大规模工程开发的进阶运行方式，图形界面更加适合于初学者。

下面以一个通过拨码开关控制 LED 灯的电路设计为例子，介绍基于 Vivado 的 FPGA 设计流程。

2.2.1 创建工程

通过双击桌面快捷方式或开始菜单的“Xilinx Design Tools→Vivado 2017.1”打开 Vivado 2017.1，在界面上“Quick Start”下选择“Create Project”。

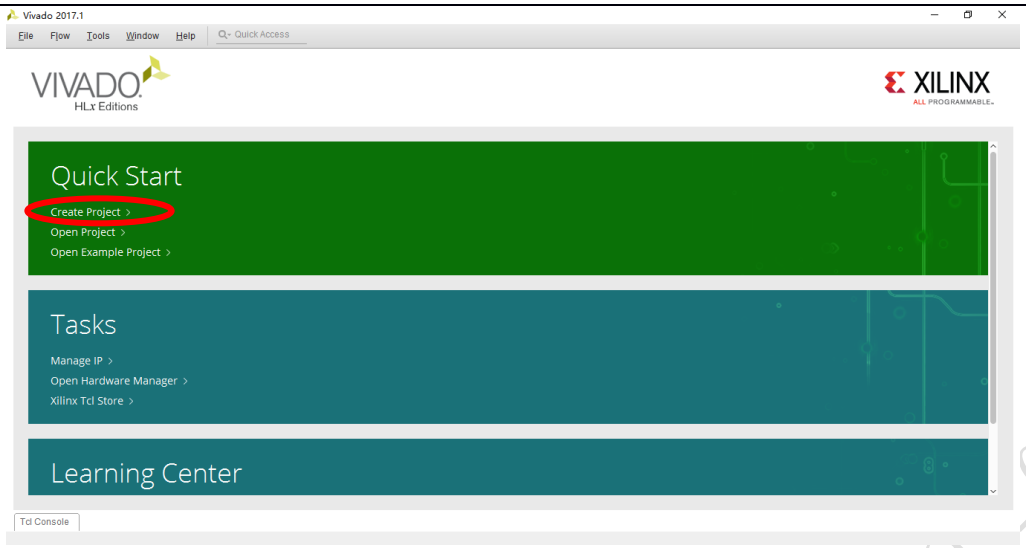


图 1-3 Vivado 启动界面

新建工程向导，点击“Next”。

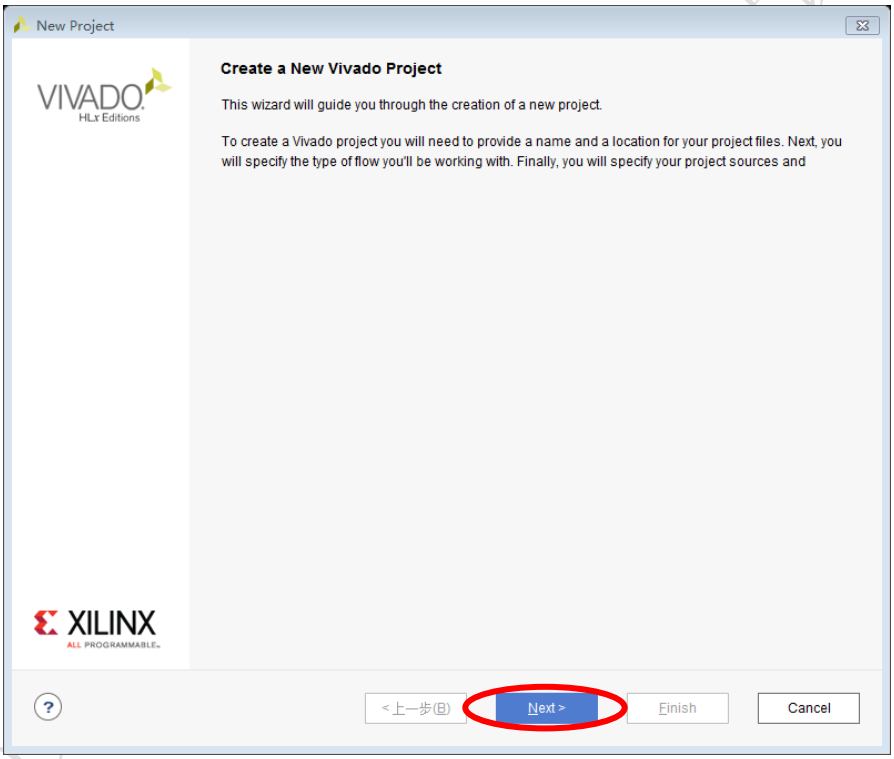


图 1-4 新建工程向导

输入工程名称并选择工程的文件位置，并勾选“Create project subdirectory”选项，为工程在指定存储路径下建立独立的文件夹。设置完成后，点击“Next”。

注意：工程名称和存储路径中不能出现中文和空格，建议工程名称以字母、数字、下划线来组成。

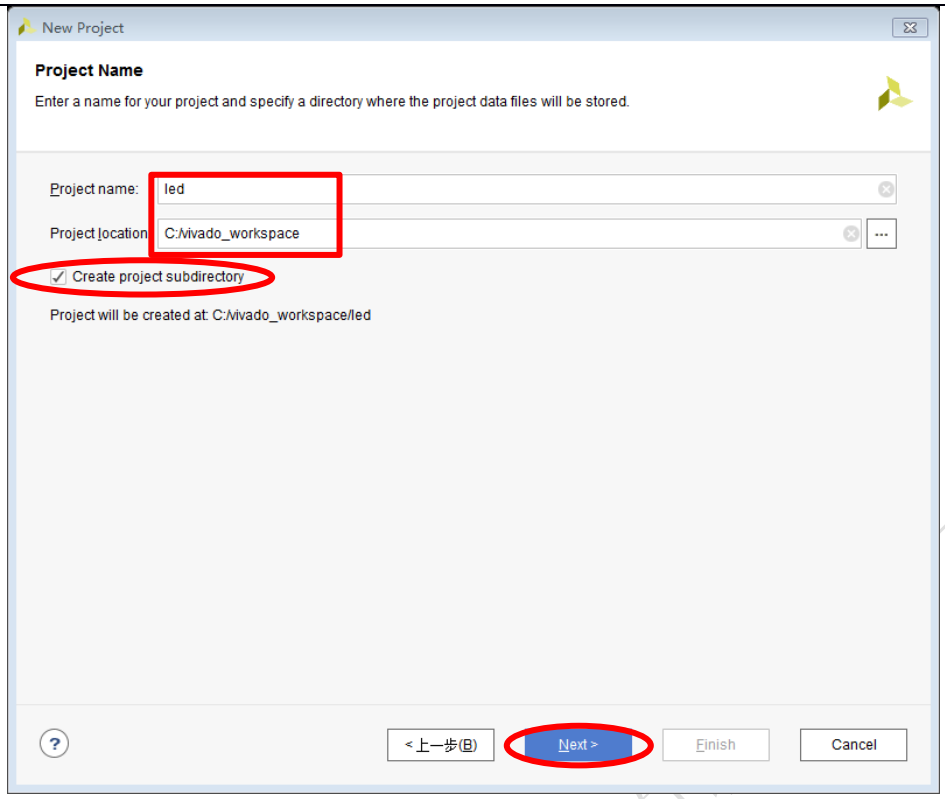


图 1-5 设置工程名称和位置

选择“RTL Project”一项，并勾选“Do not specify sources at this time”，勾选该选项是为了跳过在新建工程的过程中添加设计源文件，如果要在新建工程时添加源文件则不勾选。点击“Next”。

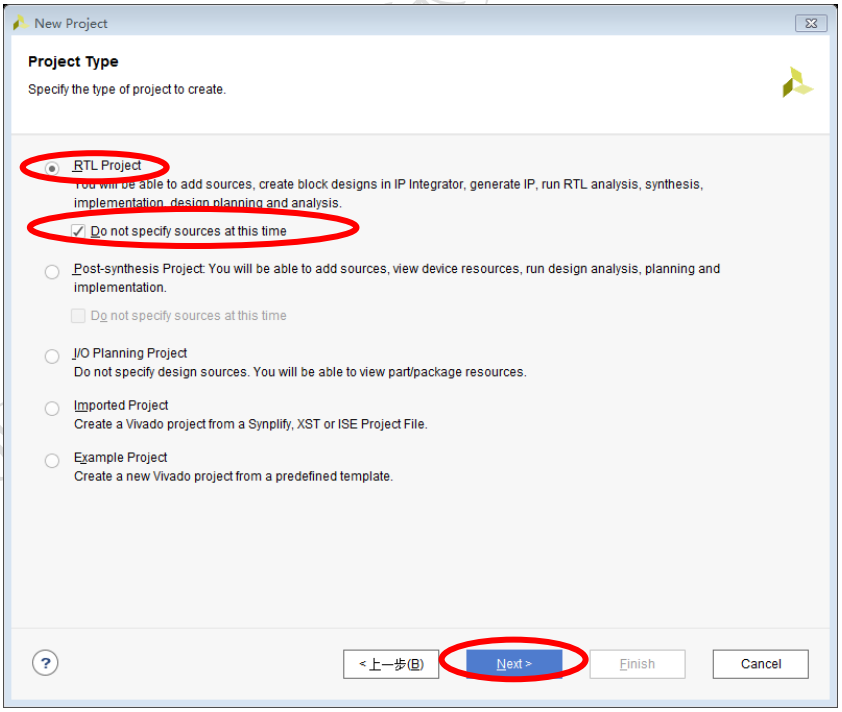


图 1-6 设置工程类型

根据使用的 FPGA 开发平台，选择对应的 FPGA 目标器件。根据实验平台搭载的 FPGA，在筛选器的“Family”选择“Artix 7”，“Package”选择“fbg676”，在筛选得到的型号里面选择“xc7a200tfbg676-1”。点击“Next”。

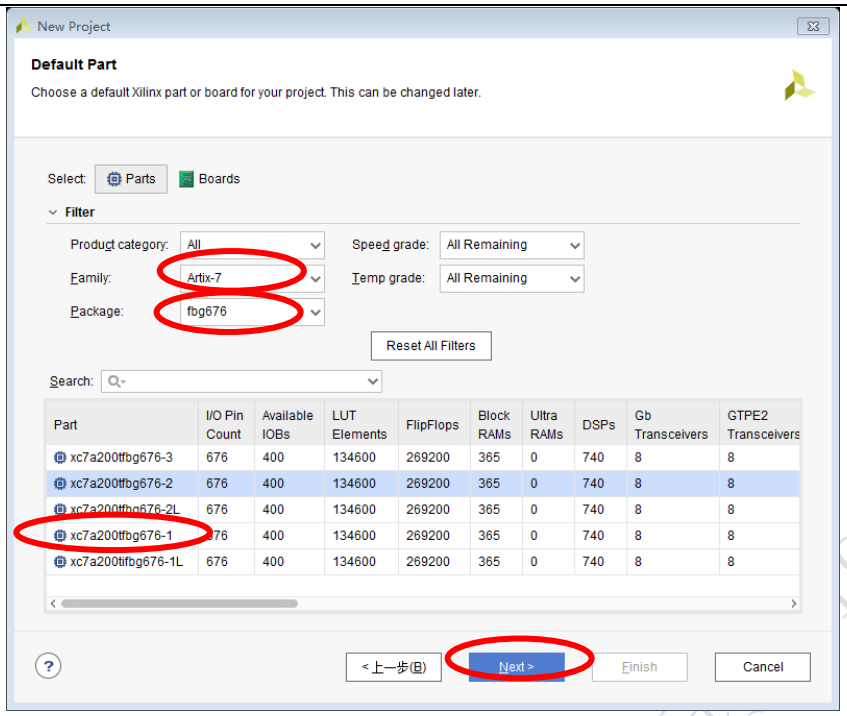


图 1-7 选择目标器件

确认工程设置信息是否正确。正确点击“Finish”，不正确则点击“上一步”返回相应步骤修改。

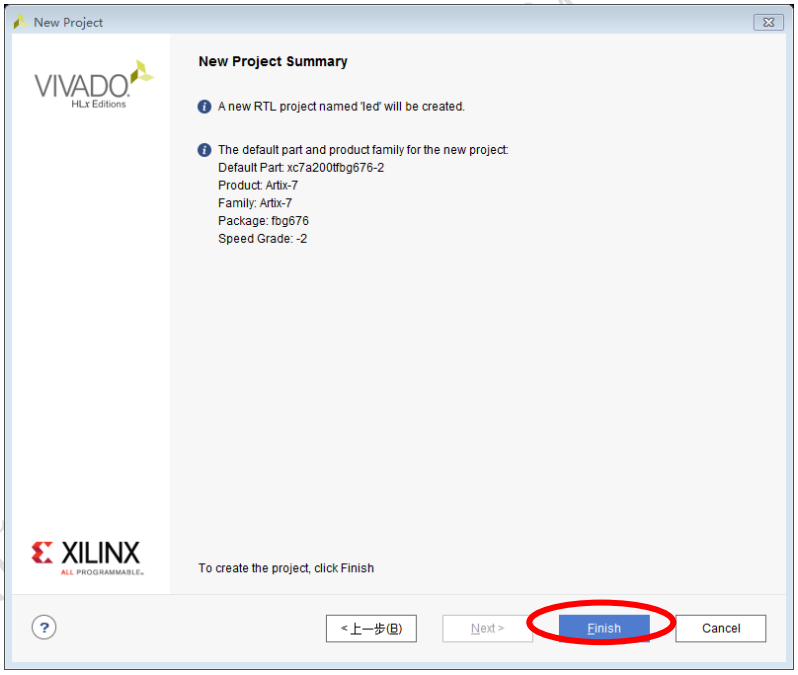


图 1-8 工程信息

完成工程新建。

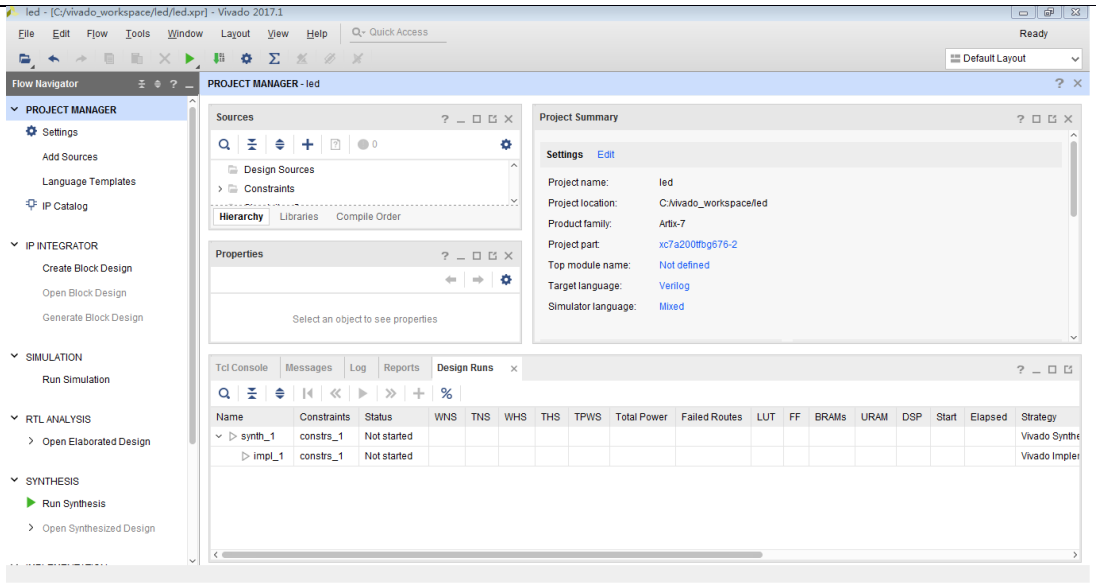


图 1-9 工程新建完成显示界面

2.2.2 添加设计文件

以使用 Verilog 完成 RTL 设计为例。Verilog 代码都是以“.v”为后缀名的文件，可以在其他文件编辑器里写好，再添加到新建的工程中，也可以在工程中新建一个再编辑。

添加源文件。在“Flow Navigator”窗口下的“Project Manager”下点击“Add sources”，或者点击“Source”窗口下“Add Sources”按钮，或者使用快捷键“Alt + A”。

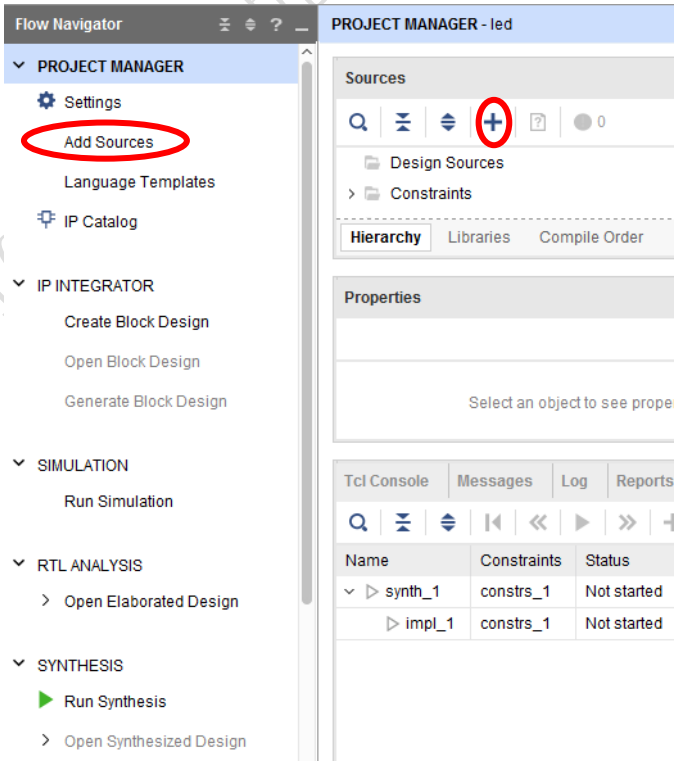


图 1-10 添加源文件

添加设计文件。选择“Add or create design sources”来添加或新建 Verilog 或 VHDL 源文件，点

击“Next”。

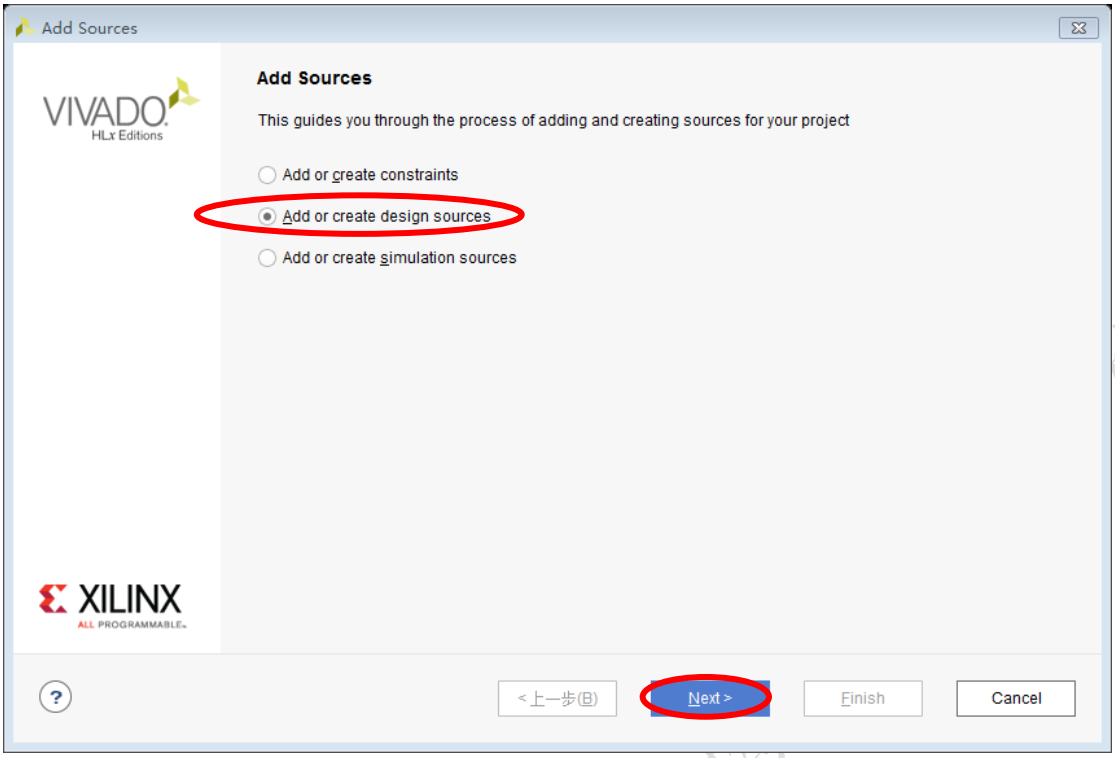


图 1-11 添加设计文件

添加或者新建设计文件。如添加已有设计文件或者添加包含已有设计文件的文件夹，选择“Add Files”或者“Add Directories”，然后在文件浏览窗口选择已有的设计文件完成添加。如创建新的设计文件，则选择“Create File”。这里新建文件。

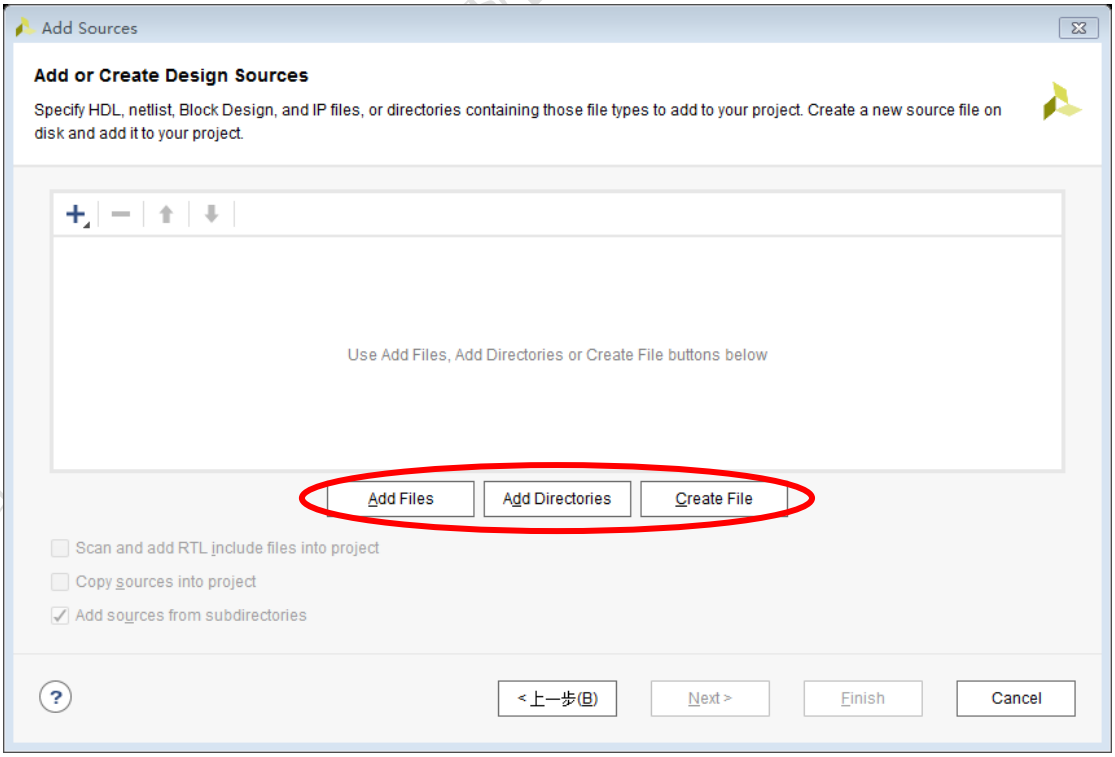


图 1-12 添加或者新建设计文件

设置新创建文件的类型、名称和文件位置。注意：文件名称和位置路径中不能出现中文和空格。

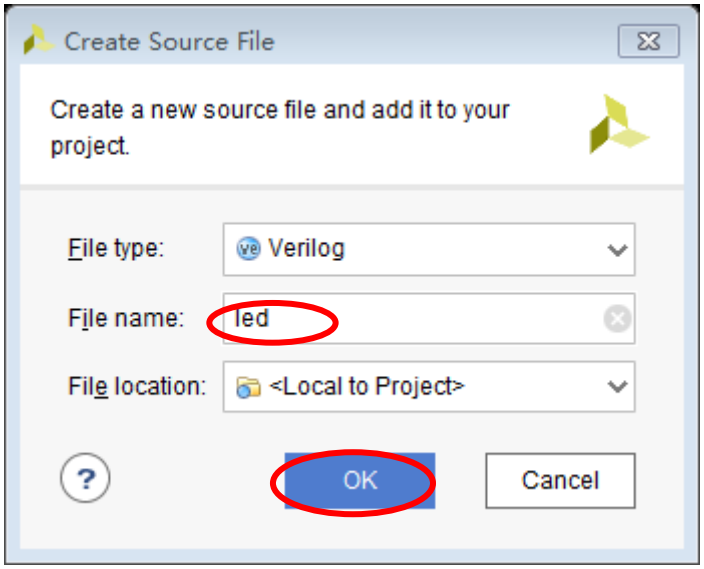


图 1-13 设置新的设计文件

继续添加设计文件或者修改已添加设计文件设置。点击“Finish”。

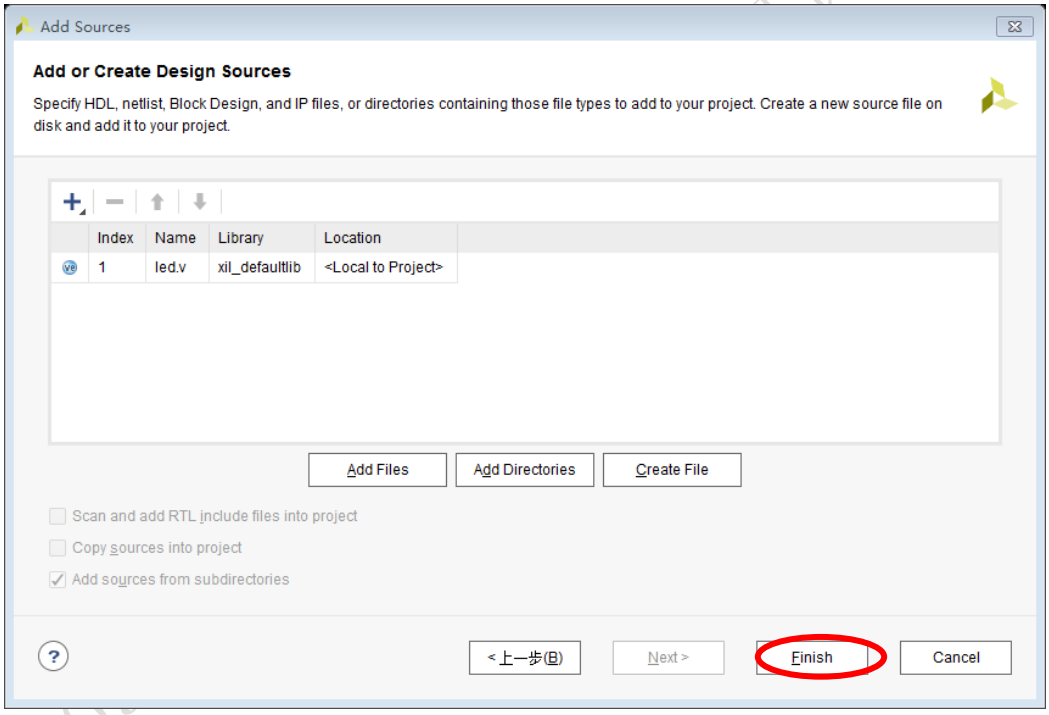


图 1-14 完成设计文件添加

模块端口设置。在“Module Definition”中的“I/O Port Definitions”，输入设计模块所需的端口，并设置端口方向，如果端口为总线型，勾选“Bus”选项，并通过“MSB”和“LSB”确定总线宽度。完成后点击“OK”。端口设置也可以在编辑源文件时完成，即可以在这一步直接点“OK”跳过。

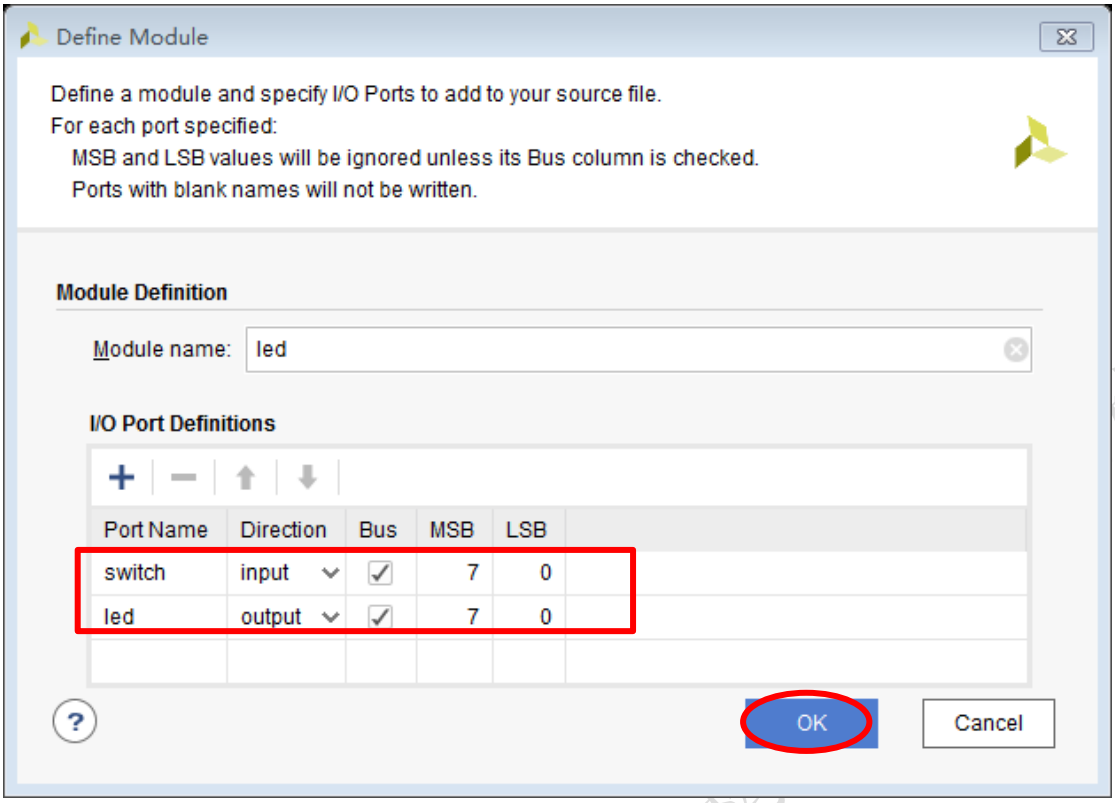


图 1-15 模块端口设置

双击“Sources”中的“Design Sources”下的“led.v”中打开该文件，输入相应的设计代码。如果设置时文件位置按默认的<Local to Project>，则设计文件位于工程目录下的“\led.srcs\sources_1\new”中。完成的设计文件如下图所示。

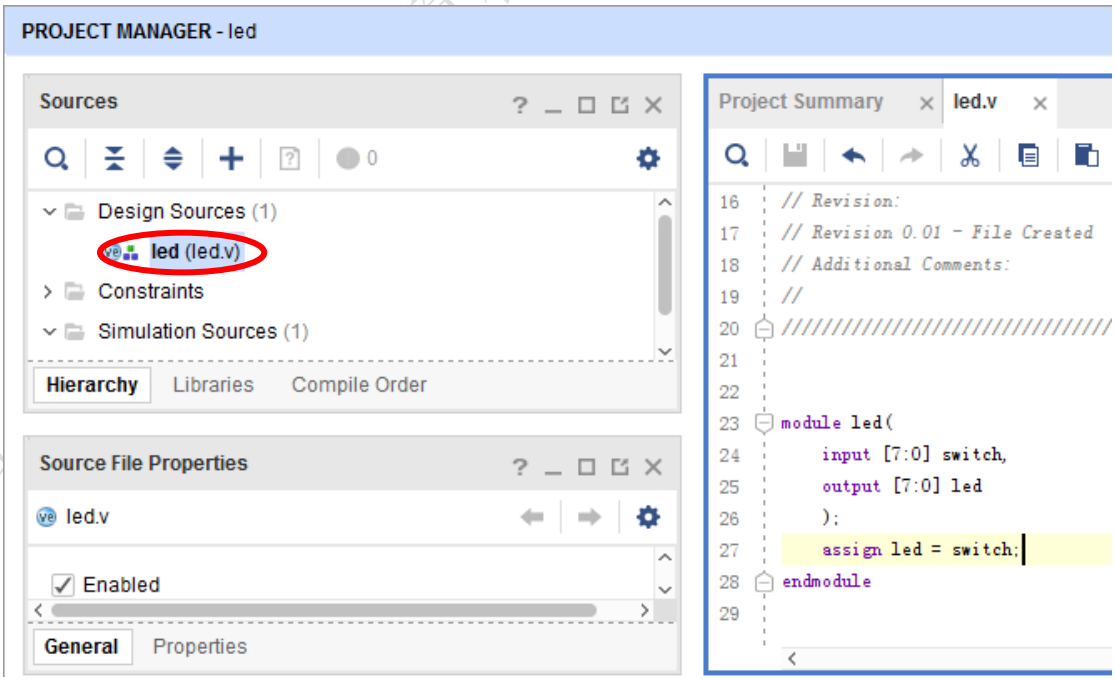


图 1-16 编辑设计文件

2.2.3 功能仿真

Vivado 集成了仿真器 Vivado Simulator，可以用它来进行功能仿真。

首先添加测试激励文件。在“Source”中“Simulation Sources”右击选择“Add source”。

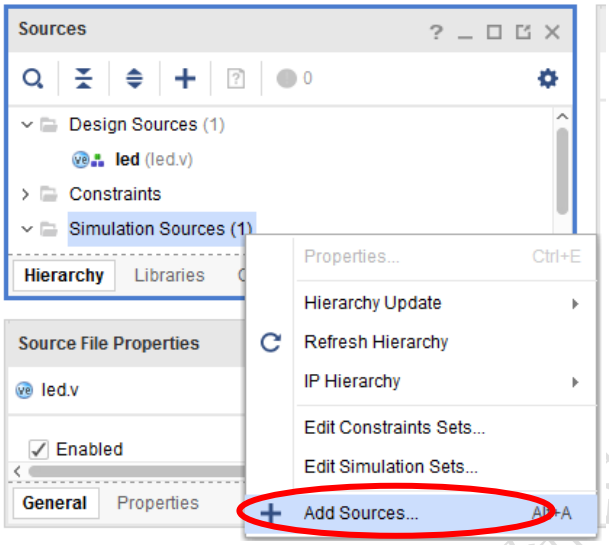


图 1-17 添加测试激励文件

在“Add Source”界面中选择“Add or Create Simulation Sources”，点击“Next”。

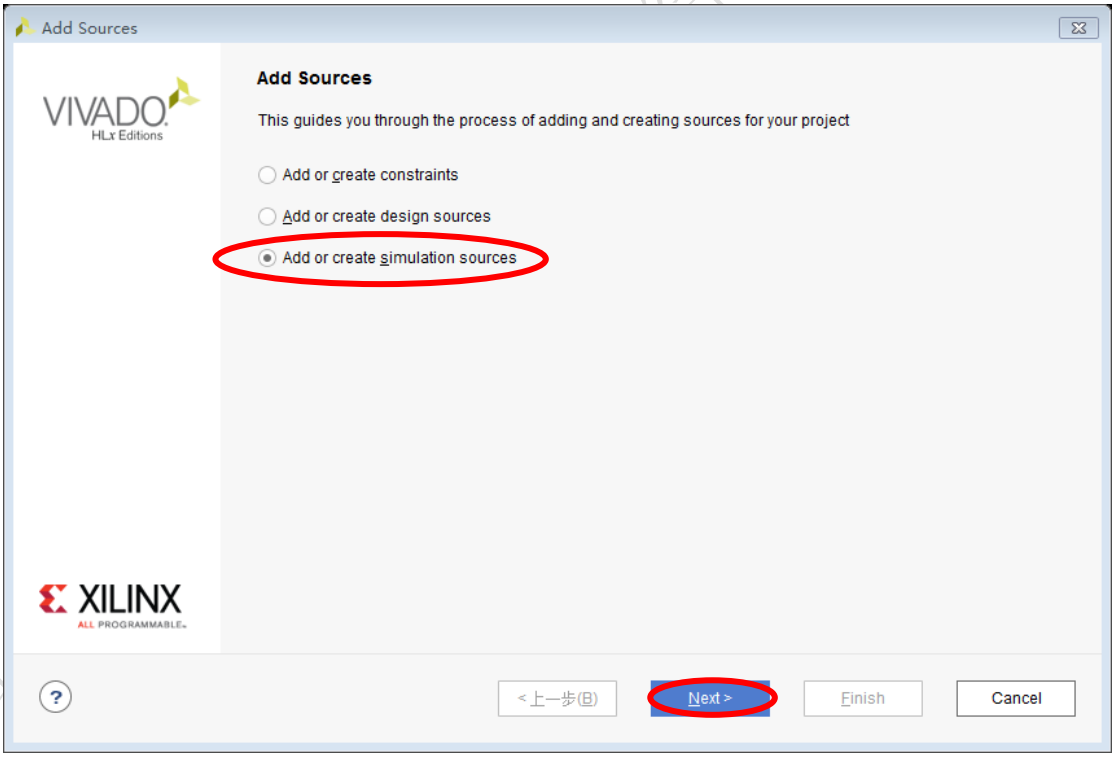


图 1-18 添加和新建测试文件

选择“Create File”，新建一个激励测试文件。

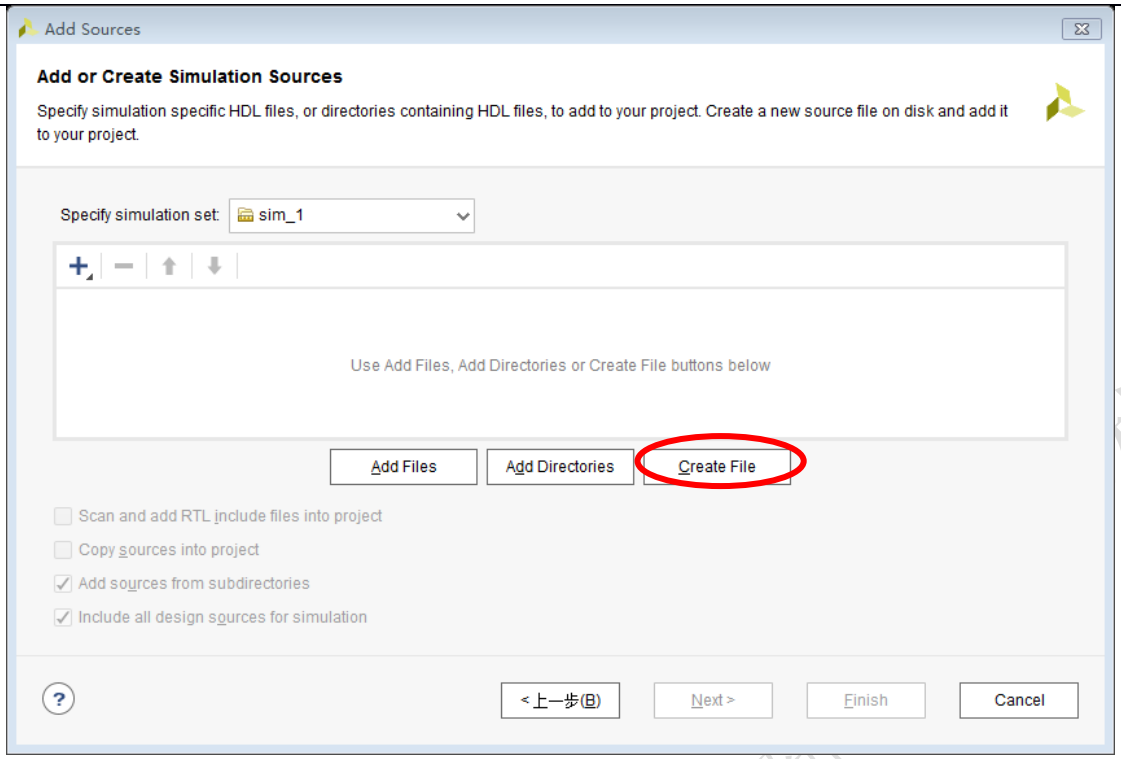


图 1-19 新建测试文件

输入激励测试文件名，点击“OK”。

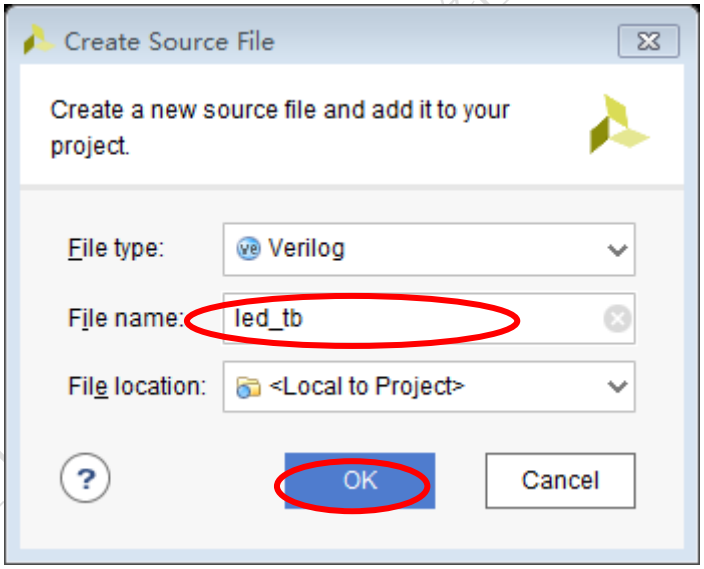


图 1-20 新建测试文件设置

完成新建测试文件，点击“Finish”。

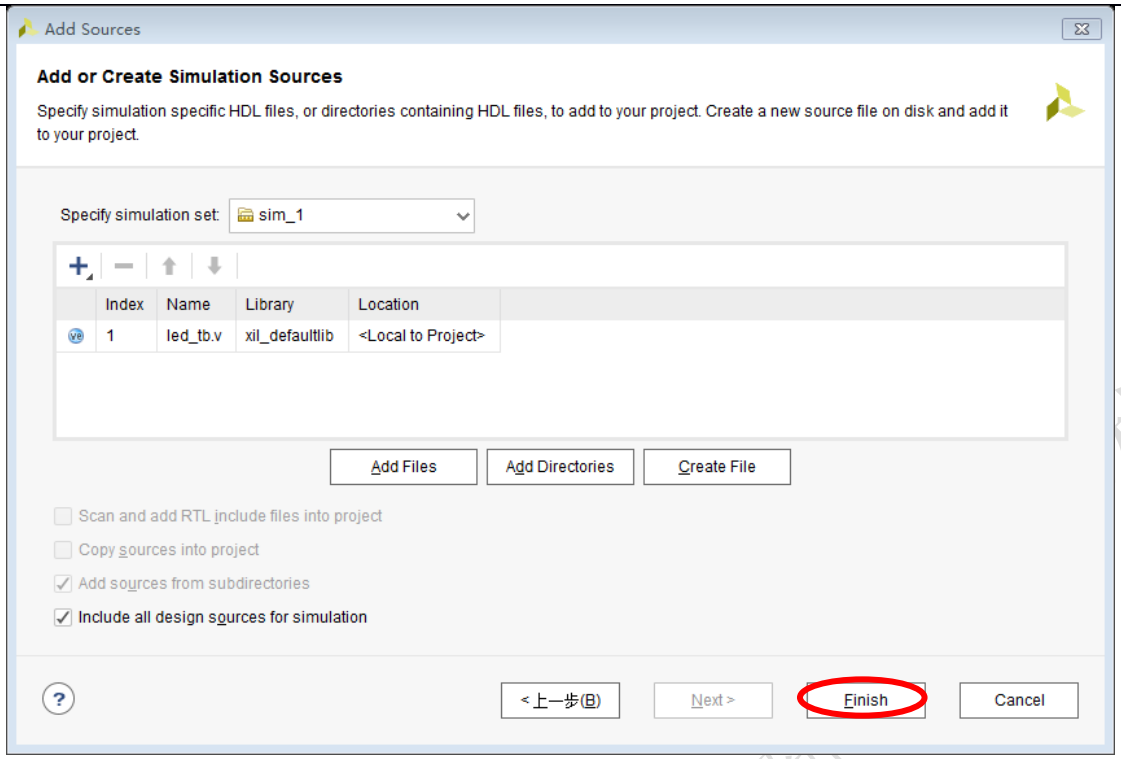


图 1-21 完成新建测试文件

对测试激励文件进行 module 端口定义，由于激励测试文件不需要有对外的接口，所以不进行 I/O 端口设置直接点击“OK”，完成空白的激励测试文件创建。

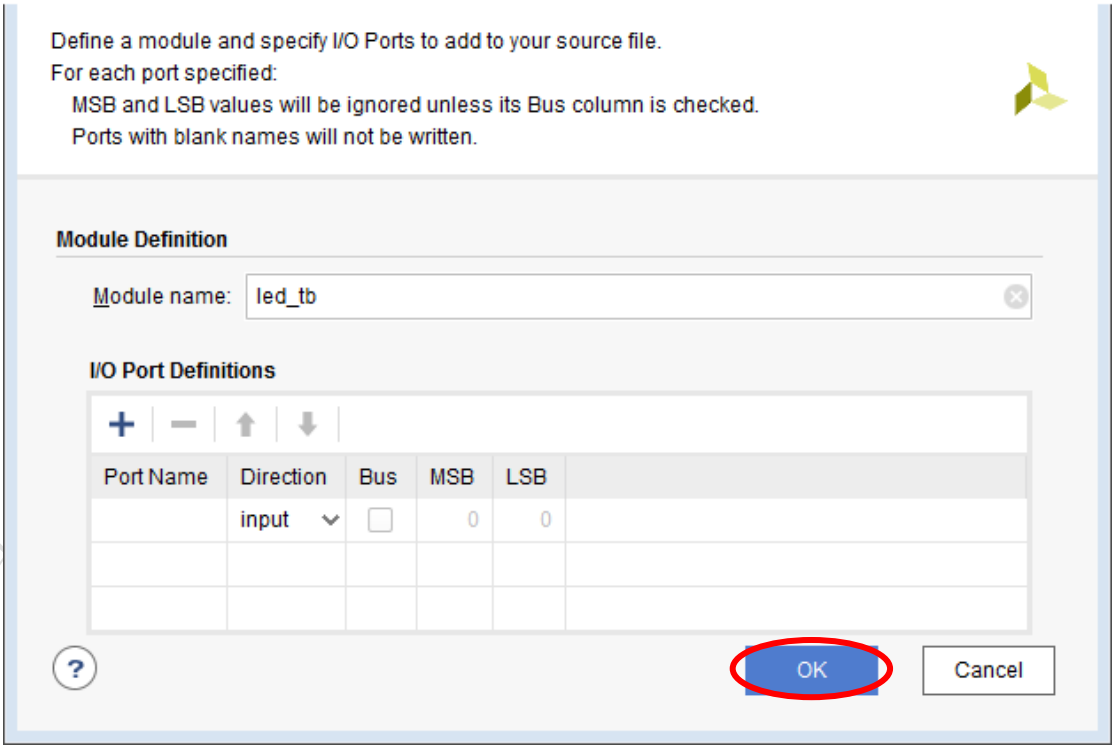


图 1-22 不需要进行端口设置

在“Source”窗口下双击打开空白的激励测试文件。测试文件 led_tb.v 位于工程目录下“\led.srcs\sim_1\new”文件夹下。

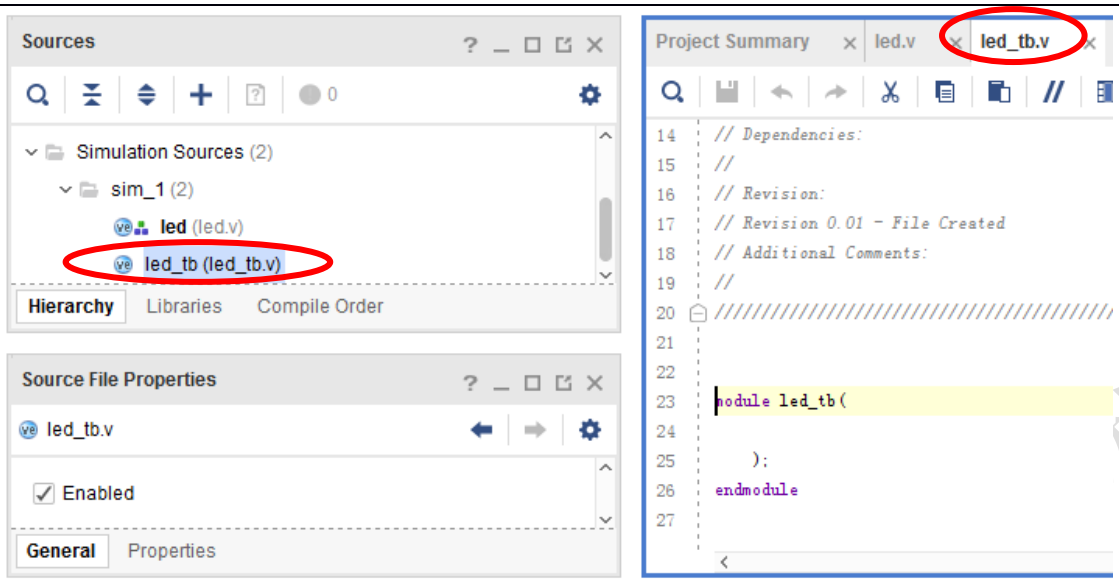


图 1-23 空白测试激励文件

完成对将要仿真的 module 的实例化和激励代码的编写，如下述代码所示。

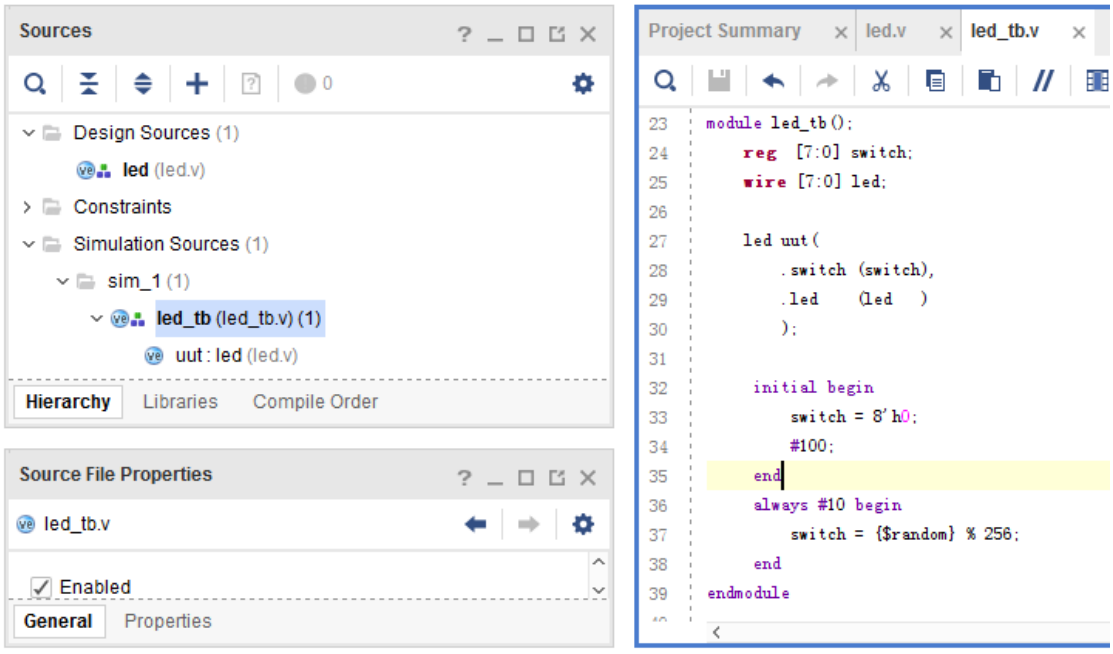


图 1-24 完成测试激励文件

进入仿真。在左侧“Flow Navigator”窗口中点击“Simulation”下的“Run Simulation”选项，选择“Run Behavioral Simulation”，进入仿真界面。

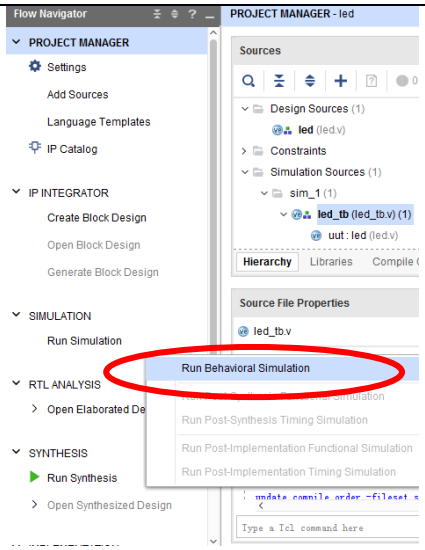


图 1-25 运行行为级仿真

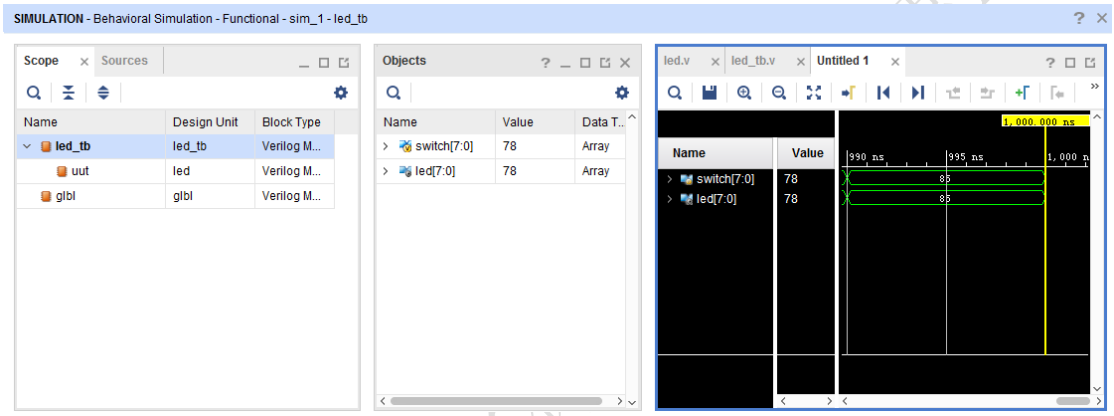


图 1-26 行为级仿真界面

可通过左侧“Scope”一栏中的目录结构定位到想要查看的 module 内部信号，在“Objects”对应的信号名称上右击选择“Add To Wave Window”，将信号加入波形图中。仿真器默认显示 I/O 信号，由于这个示例不存在内部信号，因此不需要添加观察信号。

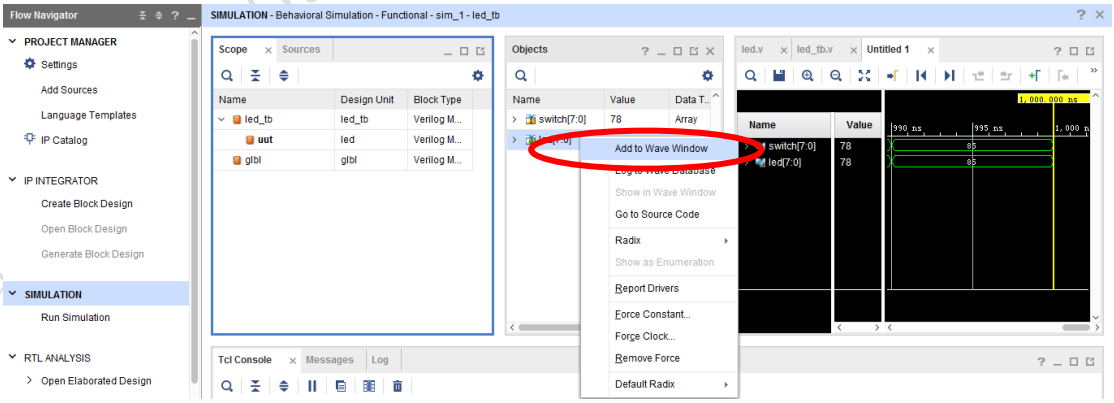


图 1-27 将内部信号添加到波形图

可通过选择工具栏中的选项来进行波形的仿真时间控制。如下图工具条，分别是复位波形（即清空现有波形）、运行仿真、运行特定时长的仿真、仿真时长设置、仿真时长单位、 单步运行、暂停、重启动。

图 1-28 仿真控制工具

观察仿真波形是否符合预期功能。在波形显示窗口上侧是波形图控制工具，由左到右分别是：查找、保存波形配置、放大、缩小、缩放到全显示、缩放到光标、转到时间 0、转到时间的最后、前一个跳变、下一次跳变、添加标记、前标记、下一个标记、交换光标。

图 1-29 波形图控制工具

可通过右键选中信号来改变信号的显示形态。如下图将信号改为二进制显示。

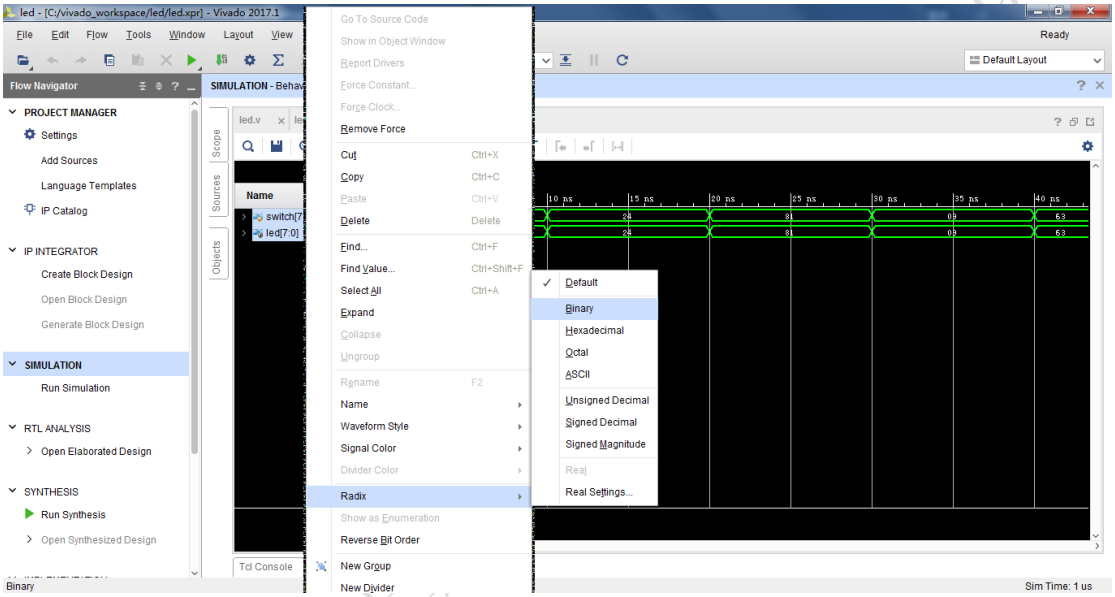


图 1-30 仿真波形窗口

查看波形检查设计功能正确性。

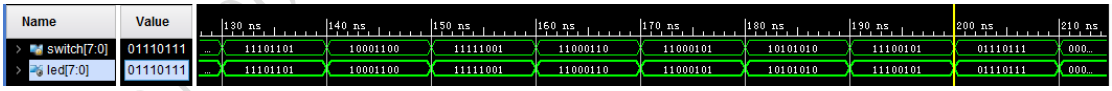


图 1-31 仿真波形结果

2.2.4 添加约束文件

添加约束文件有两种方法，一是利用 Vivado 中 I/O Planning 功能，二是直接新建 XDC 的约束文件，手动输入约束命令。下面主要介绍第二种方法：

新建约束文件。点击“Add Sources”，选择“Add or Create Constraints”，点击“Next”。

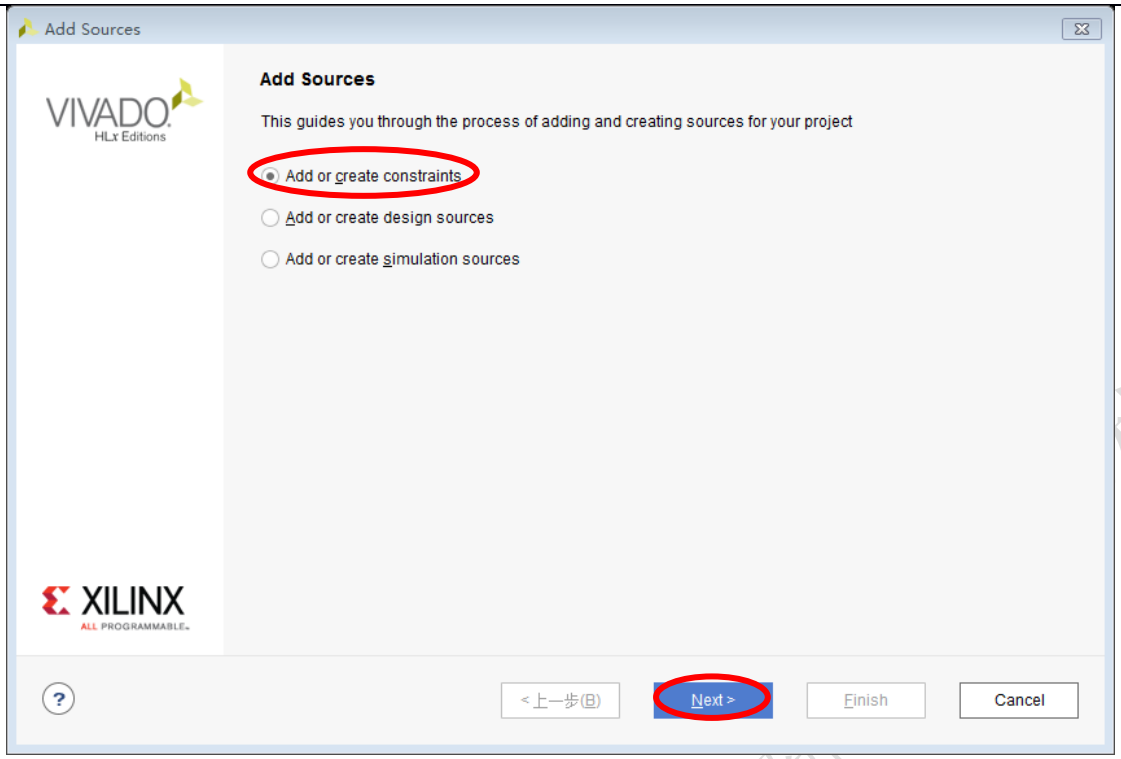


图 1-32 新建约束文件

添加或创建约束文件。点击“Create Files”添加约束文件。

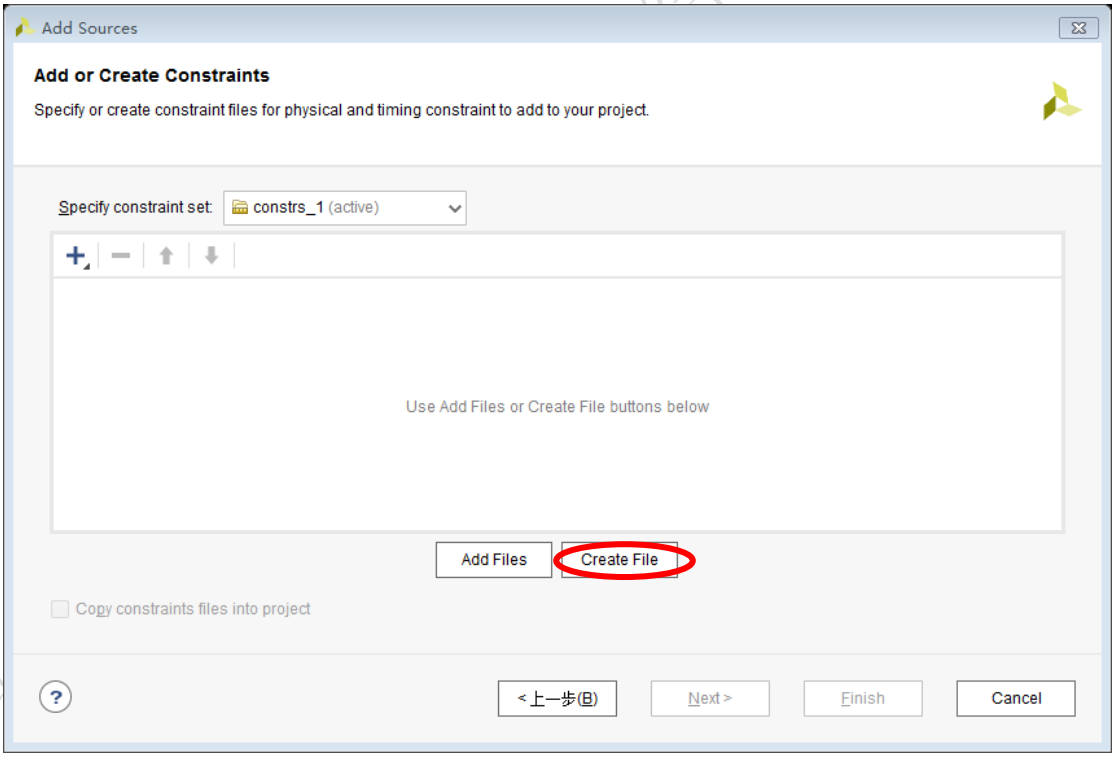


图 1-33 添加或创建约束文件

设置新建的 XDC 文件，输入 XDC 文件名，点击“OK”。默认文件在工程目录下的“\led.srcs\constrs_1\new”中。

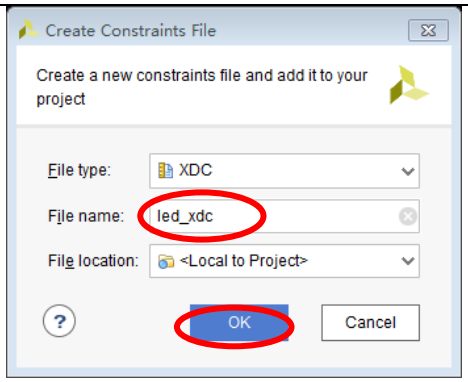


图 1-34 新建 XDC 文件设置

完成新建 XDC 文件，点击“Finish”。

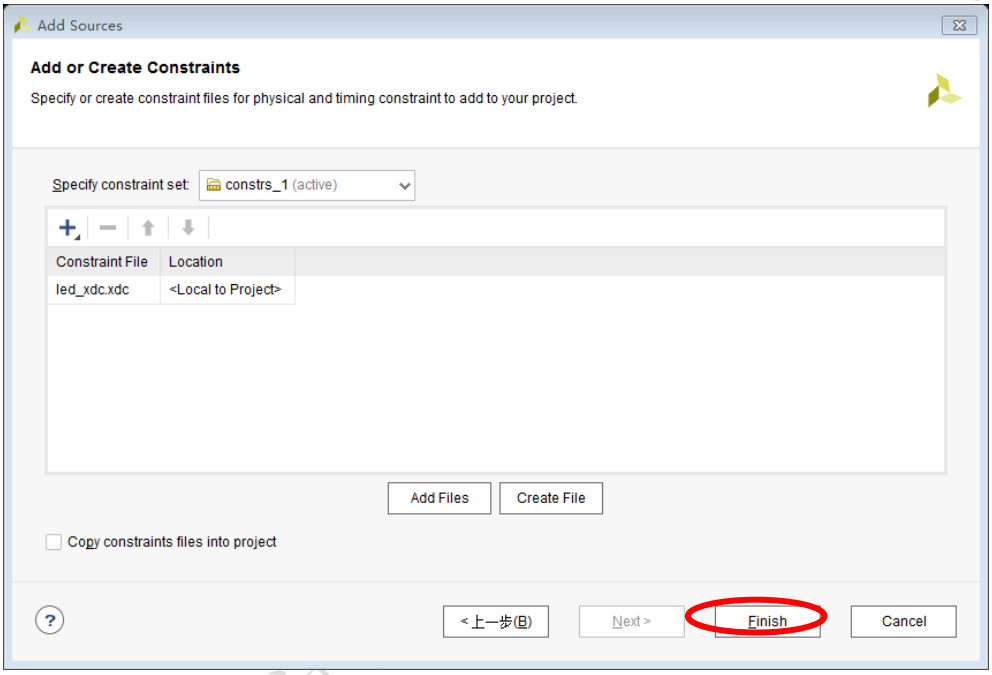


图 1-35 完成新建 XDC 文件

在“Sources”窗口“Constraints”下双击打开新建好的 XDC 文件“led_xdc.xdc”，并参照下面的约束文件格式，输入相应的 FPGA 管脚约束信息和电平标准。

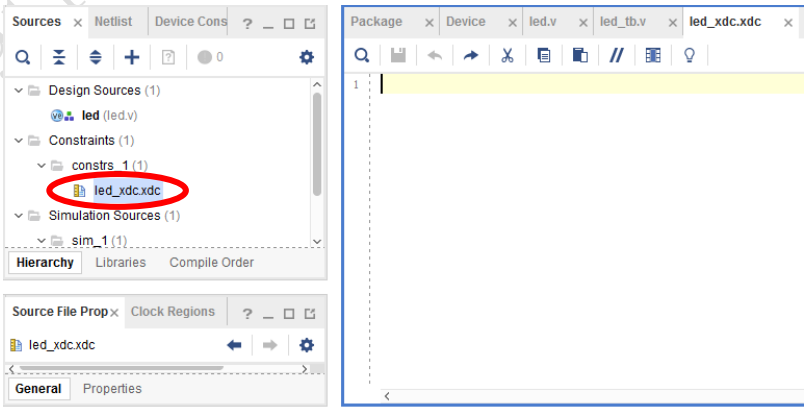


图 1-36 打开空白 XDC 文件

约束文件参考：

```
set_property PACKAGE_PIN H8 [get_ports {led[7]}]
set_property PACKAGE_PIN G8 [get_ports {led[6]}]
```

```
3   set_property PACKAGE_PIN F7 [get_ports {led[5]}]
4   set_property PACKAGE_PIN A4 [get_ports {led[4]}]
5   set_property PACKAGE_PIN A5 [get_ports {led[3]}]
6   set_property PACKAGE_PIN A3 [get_ports {led[2]}]
7   set_property PACKAGE_PIN D5 [get_ports {led[1]}]
8   set_property PACKAGE_PIN H7 [get_ports {led[0]}]
9   set_property PACKAGE_PIN Y6 [get_ports {switch[7]}]
10  set_property PACKAGE_PIN AA7 [get_ports {switch[6]}]
11  set_property PACKAGE_PIN W6 [get_ports {switch[5]}]
12  set_property PACKAGE_PIN AB6 [get_ports {switch[4]}]
13  set_property PACKAGE_PIN AC23 [get_ports {switch[3]}]
14  set_property PACKAGE_PIN AC22 [get_ports {switch[2]}]
15  set_property PACKAGE_PIN AD24 [get_ports {switch[1]}]
16  set_property PACKAGE_PIN AC21 [get_ports {switch[0]}]
17  set_property IOSTANDARD LVCMOS33 [get_ports {led[7]}]
18  set_property IOSTANDARD LVCMOS33 [get_ports {led[6]}]
19  set_property IOSTANDARD LVCMOS33 [get_ports {led[5]}]
20  set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
21  set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
22  set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
23  set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
24  set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]
25  set_property IOSTANDARD LVCMOS33 [get_ports {switch[7]}]
26  set_property IOSTANDARD LVCMOS33 [get_ports {switch[6]}]
27  set_property IOSTANDARD LVCMOS33 [get_ports {switch[5]}]
28  set_property IOSTANDARD LVCMOS33 [get_ports {switch[4]}]
29  set_property IOSTANDARD LVCMOS33 [get_ports {switch[3]}]
30  set_property IOSTANDARD LVCMOS33 [get_ports {switch[2]}]
31  set_property IOSTANDARD LVCMOS33 [get_ports {switch[1]}]
32  set_property IOSTANDARD LVCMOS33 [get_ports {switch[0]}]
```

2.2.5 综合实现和生成比特流文件

在“Flow Navigator”中点击“Program and Debug”下的“Generate Bitstream”选项，工程会自动完成综合、布局布线、Bit 文件生成过程，完成之后，可点击“Open Implemented Design”来查看工程实现结果。

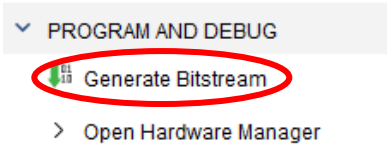


图 1-37 生成 bit 流文件

可能会出现以下提示：综合过时，需重新运行综合和实现。点击“Yes”，在弹出的“Launch Runs”窗口点“OK”。

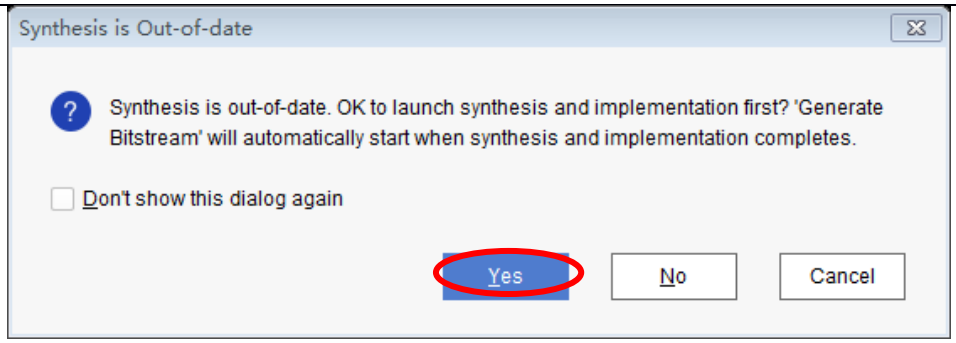


图 1-38 提示重新运行综合和实现

2.2.1 FPGA 烧写配置

在比特流文件生成完成的窗口选择“Open Hardware Manager”，进入硬件管理界面。连接 FPGA 开发板的电源线和与电脑的下载线，打开 FPGA 电源。

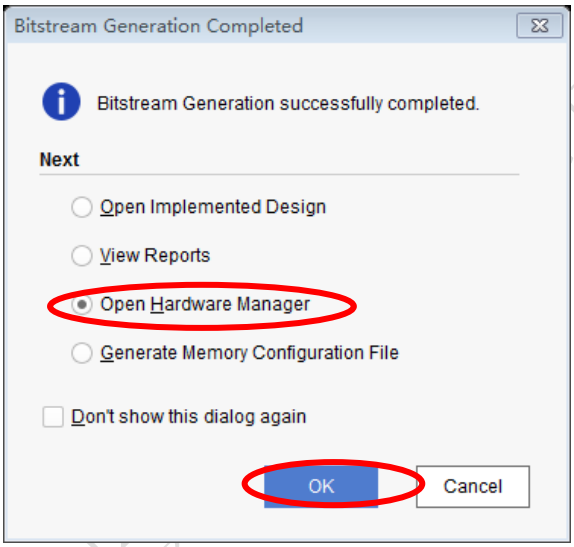


图 1-39 打开硬件程序和调试管理

在“Hardware Manager”窗口的提示信息中，点击“Open target”的下拉菜单的“Open New Target”（或在“Flow Navigator”下“Program and Debug”中展开“Open Hardware Manager”，点击“Open Target”->“Open New Target”）。也可以选择“Auto Connect”自动连接器件。

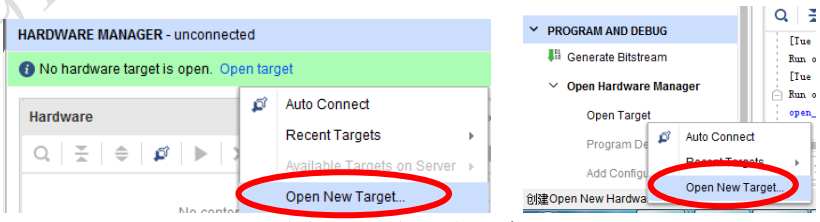


图 1-40 打开新目标

在“Open Hardware Target”向导中，先点击“Next”，进入 Server 选择向导。

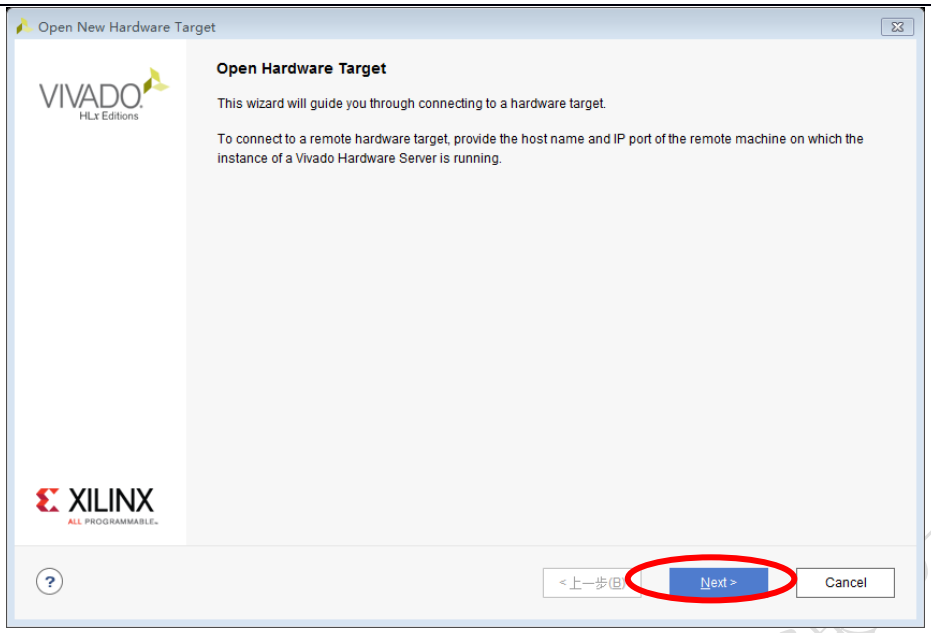


图 1-41　Open Hardware Target 向导

选择连接到“Local server”，点击“Next”。

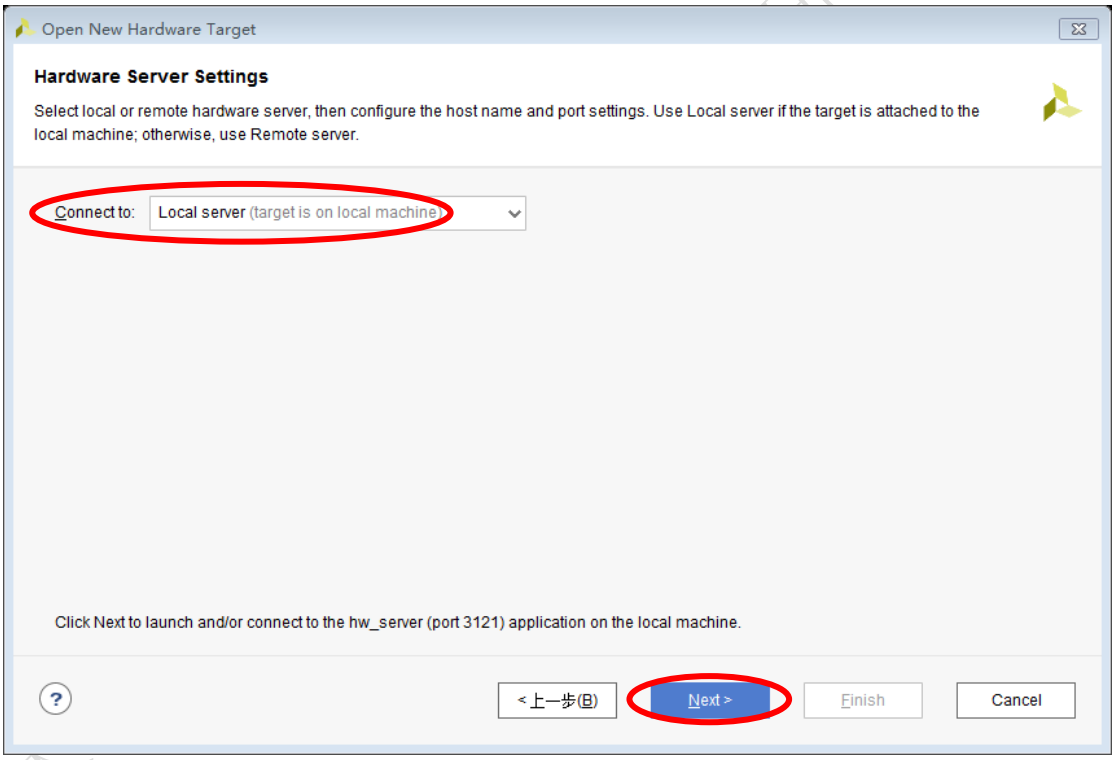


图 1-42 连接 Local server

选择目标硬件，点击“Next”。

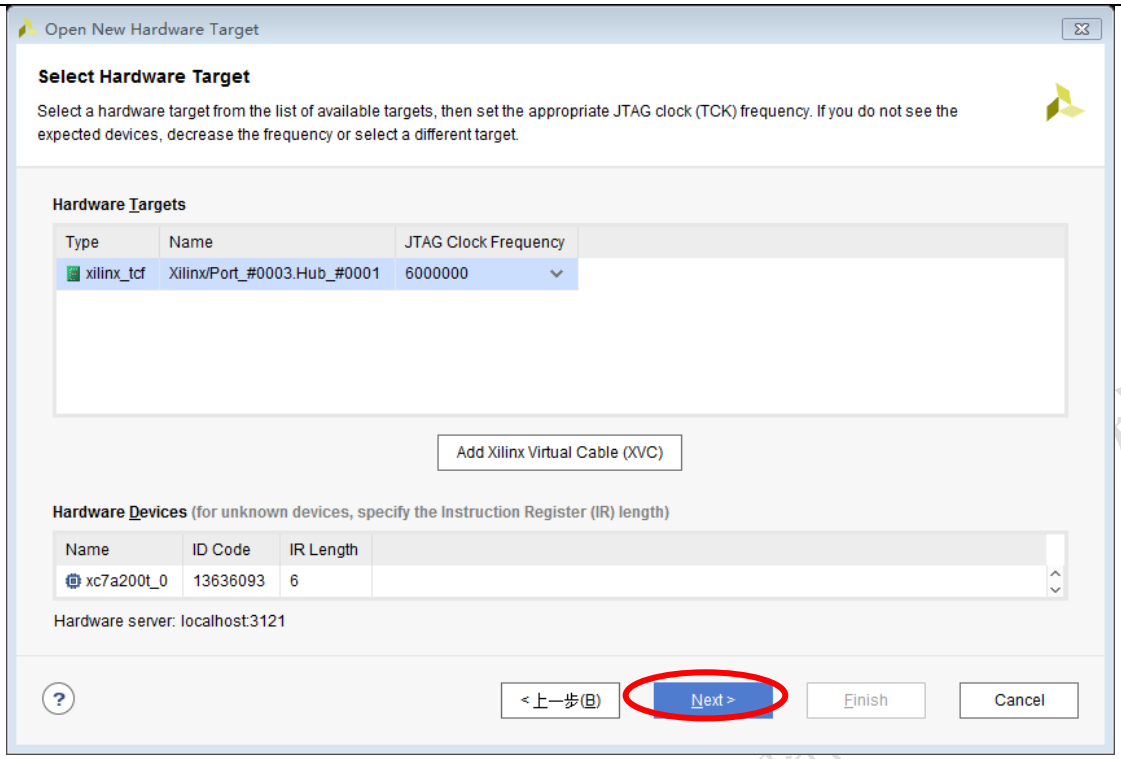


图 1-43 选择目标硬件

完成目标硬件打开。点击“Finish”。

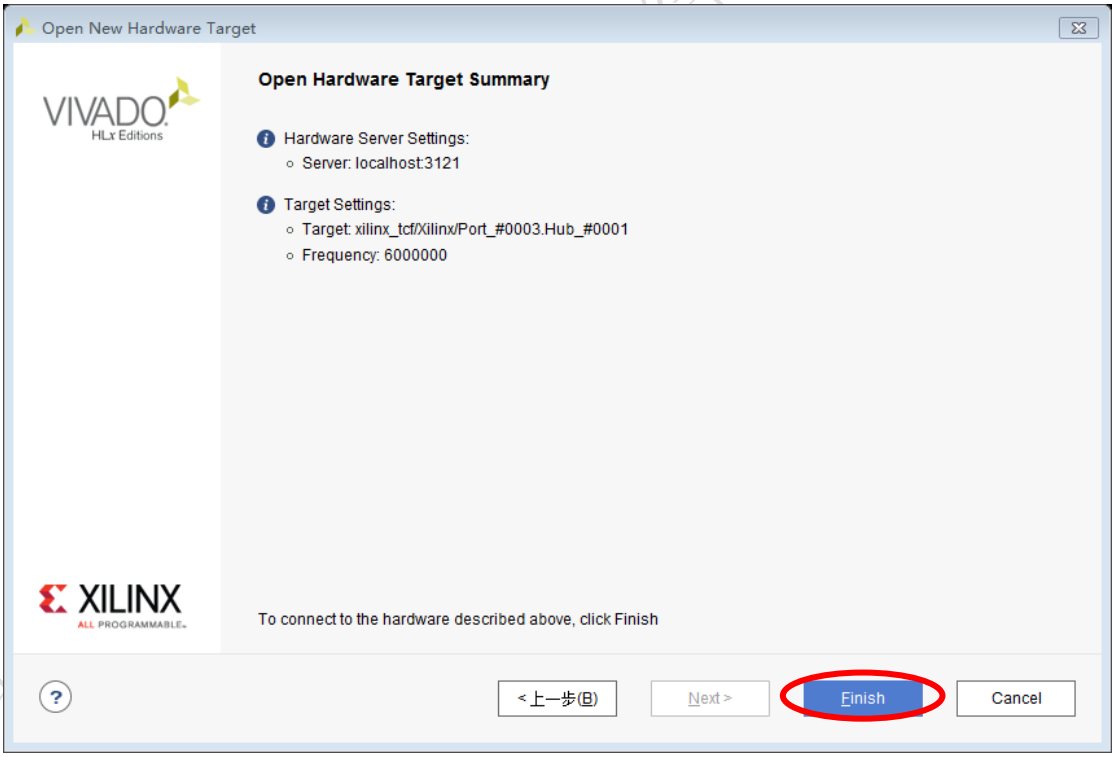


图 1-44 完成目标硬件打开

对目标硬件编程。在“Hardware”窗口右键单击目标器件“xc7a200t_0”，选择“Program Device…”。或者“Flow Navigator”窗口中“Program and Debug”->“Hardware Manager”->“Program Device”。

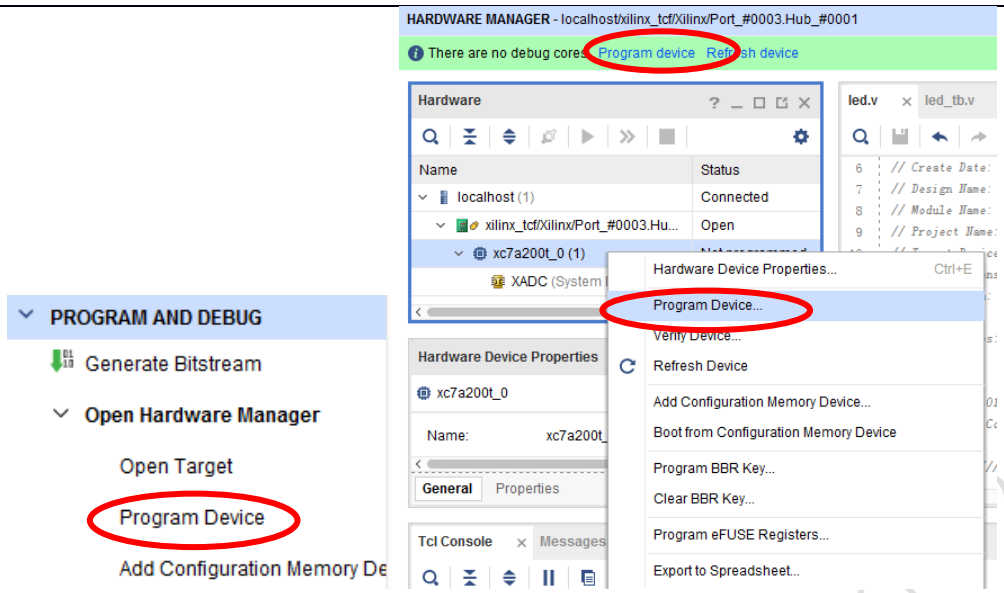


图 1-45 对目标器件编程

选择下载的 bit 流文件，点击“Program”。

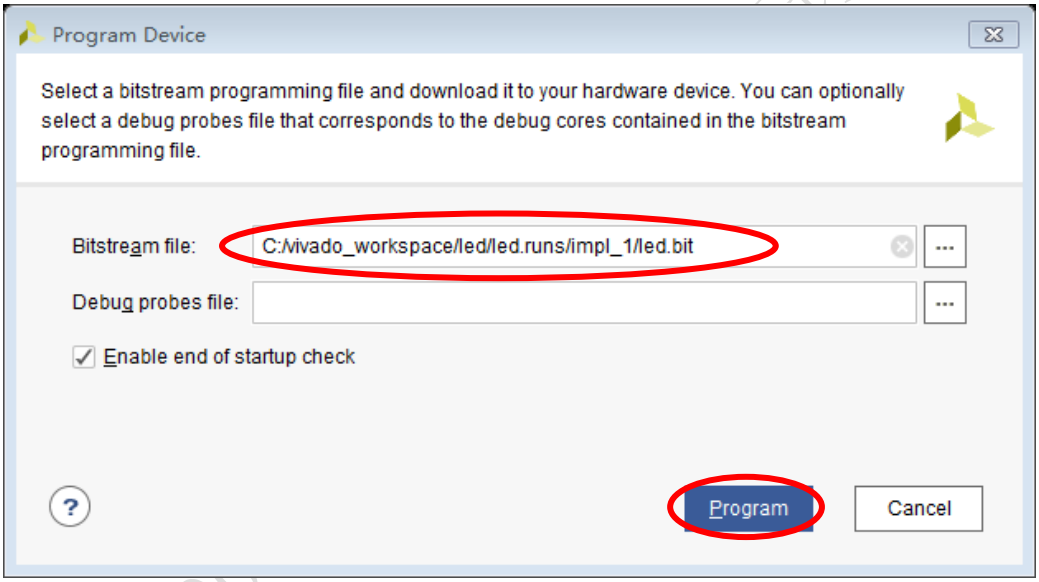


图 1-46 选择 bit 流文件

完成下载后，“Hardware”窗口下的“xc7a200t_0”的状态变成“Programmed”。

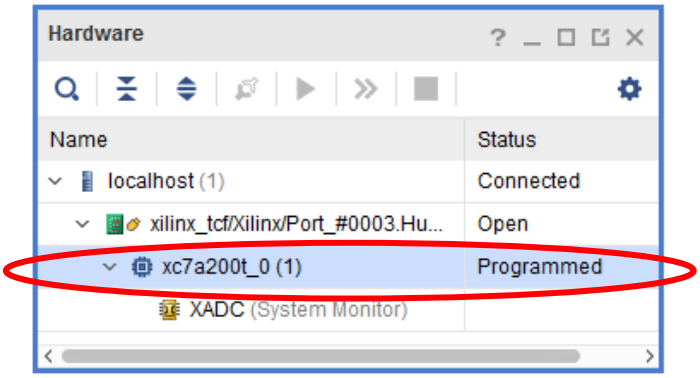


图 1-47 完成 FPGA 编程

FPGA 开发板上，8 个拨码开关 SW18~SW25 对应控制 8 个 led 灯 LED1~LED8 的亮灭。设计完成。

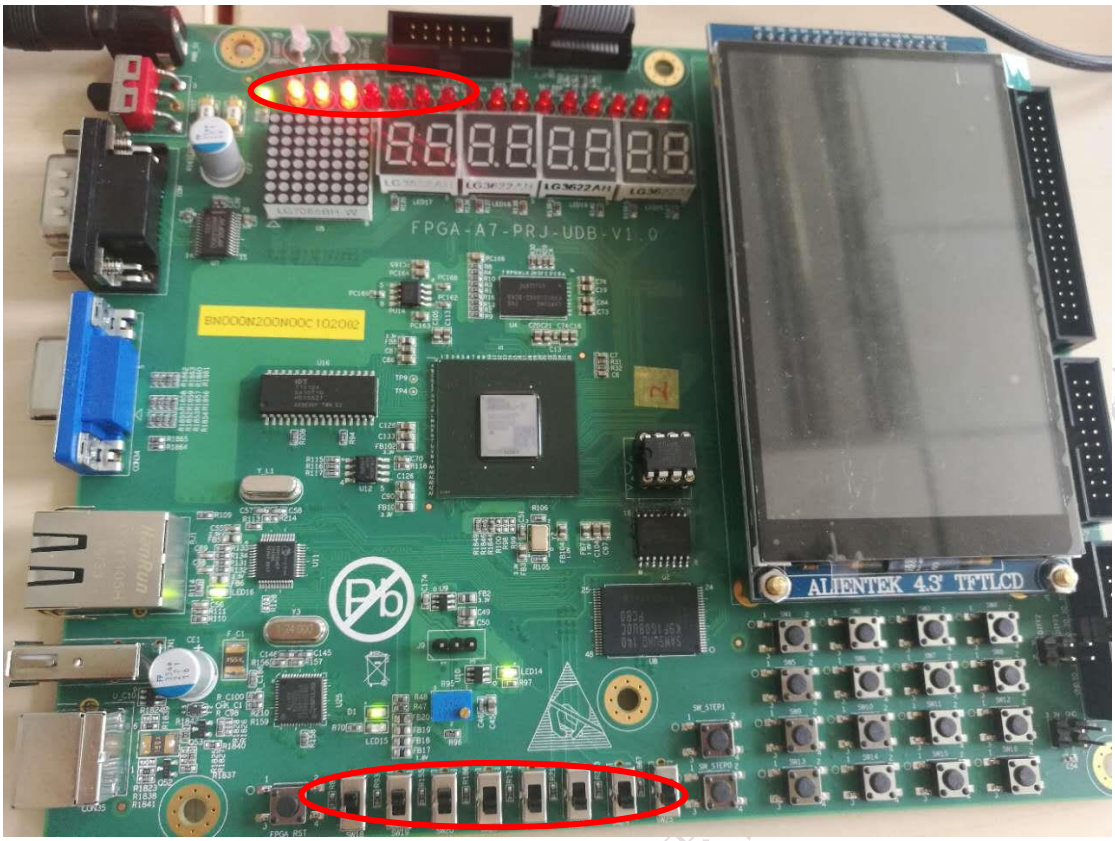


图 1-48 FPGA 运行结果

2.3 Vivado 使用小贴士

根据往届同学们的使用情况，总结一些 Vivado 使用的建议如下：

1. 强烈建议各位不要在虚拟机下安装 Vivado 并使用，Vivado 在综合实现过程中需要较大的内存，因此在虚拟机下运行 Vivado 时间会变长，而且是非线性的变长，特别是当虚拟机的内存只有区区 2GB 的时候，等待的时间有时会长得让人难以忍受。
2. 如果你的电脑上已经安装了 Vivado，不要再重新装一个了，直接升级到最新版本就好了；当然，也不要就是不升级，我们后期发布的实验环境都是用最新版本适配检查了一遍，我们不保证在旧版本上不会出现异常。
3. 如果你是在裸机上安装了 Linux 系统，直接安装 Linux 版的 Vivado 就好了，比 Windows 版的还要快一点。不过很有可能会在 FPGA 下载电缆驱动的安装上出现问题。因为这方面问题的原因五花八门，请自行上网查找解决方法。既然你已经在裸机上装 Linux 系统了，相信你早已经有了这样的觉悟……
4. 不要用中文的账户名，会出现各种诡异的问题，虽然网上提供了一些解决的方法，但并不能保证对所有问题都适用。
5. 在上板调试环节如果发现 Vivado 不能正常识别开发板时，请按照下列步骤排查：

a) 检查 FPGA 开发板是否上电？如果确实上电了，不放心的话，可以断电后再上一次电。

b) 检查 FPGA 配置下载器的 USB 电缆是否与电脑以及适配器正常连接。

c) 检查 Cable Driver 是否正常安装。

   i. 如果是在 Windows 系统下，若驱动正常安装，你可以在设备管理器的界面中查看到“Programming cables→Xilinx USB Cable”这样的条目。如果在设备管理器界面中找不到该条目，说明下载器 USB Cable 的驱动未正常安装，参考 d)进行安装。

   ii. 如果是在虚拟机上安装的系统，还需要检查虚拟机软件中与 USB 相关的配置，确保 USB 全部转发到虚拟机。对于 Virtualbox 而言，选择指定虚拟机→设置→USB 设备→添加筛选器（默认添加一个各个域的值都为空的 USB 筛选器，此种筛选器将会匹配所有连接到你电脑上的 USB 设备。）

d) 如果在 Windows 或 Linux 下发现下载器 USB Cable 的驱动未正常安装，请参考 https://china.xilinx.com/support/answers/59128.html 进行驱动的安装。

e) 在 Vivado Tcl Console 面板下输入 disconnect_hw_server 命令并在 Open target 的时候选择 Auto Connect。

f) 重启 Vivado 软件，重复步骤 d)。

g) 重启你的电脑，重复步骤 c)和 d)。

h) 找一个已经证实能被其它电脑识别出的实验箱，如果用你的电脑连上后能识别，拿着你手头不能识别的那个实验箱来找我们。

i) 如果上面的手段尝试了都没有效果，再来找我们吧，尽管我们可能也束手无策☹。