

0 | 前言

第一步（文森特·梵·高）

人们经常会忘记的是，纵使是一趟没有目的地的旅程，也还是从单纯踏出第一步开始的。



~ · ~

数字逻辑是计算机系统设计的一门基础课程。课程的终极目标是让学生掌握设计复杂数字系统的能力。生活中的一切电子产品，大到电脑，小到电子表电子秤都是一个数字系统。在这门课程的实验里，你将从简单的数码管开始，逐步掌握构建数字系统的能力，甚至最终完成一个处理器。

当你进入数字逻辑实验的课堂，就会拿到这个红色盒子装的 EGO1 开发板（图 0.1）。开发板中间的芯片，就是 FPGA。

FPGA 全称现场可编程逻辑门阵列（Field Programmable Gate Array），不同于电脑，手机处理器的芯片，在硬件层次上已经固化，不可修改，FPGA，内部存在着大量的 LUT，而 LUT 可以模拟任何逻辑门，又因为 LUT 之间的路径是可以选择的，这就意味着它可以通过编程“改变”内部结构，实现从一个加法器到一个 CPU 的跨越。

数字逻辑实验课程就是围绕着 FPGA 开发板展开，设计各式各样的电路，实现一些有趣、实用的功能。

附：FPGA 芯片价格较贵——开发板上的一颗就需要上千元，请大家爱惜实验器材！

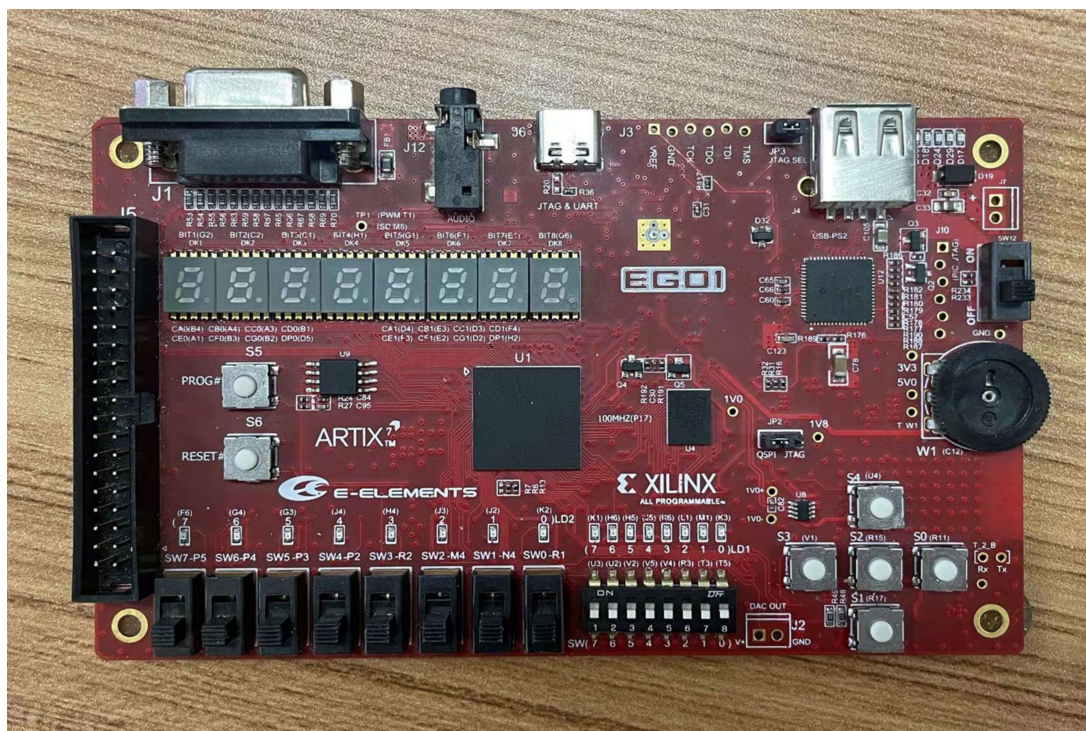


图 0.1: Ego1 开发板

以下的实验的目的是对 FPGA 有一个初步了解，**不是必做内容**，可以自行完成。

从零开始的 FPGA 生活

多年以后，面对 Vivado Error，Zakilim 将会回想起老师带他去见识 EGO1 的那个遥远的下午。

当你第一次进入数逻实验的课堂，你就会拿到这个红色小盒子装上的 FPGA 开发板——EGO1。而开发板中央的那块芯片，就是大名鼎鼎的 FPGA。你已经在数逻的第一章学到过，FPGA 是一种可编程电路，就像数电实验箱一样（事实上，从 22 级开始就没有这门课了）。但是不用担心，FPGA 开发板比数电实验箱更为健壮，很少出现实验硬件故障的问题。不过不用担心自己无法充分受到调试错误的训练，Vivado 很好的接替数电实验箱完成了这一职责，所以当你遇到 Vivado 出现问题时，请不要慌张，可以查阅https://ustb-806.github.io/DigitalLogic_Info/#中的常见问题，去各类搜索引擎上搜索，或者你也可以找到你的助教，向他们咨询问题。

当然，助教很忙，所以请稍稍体谅一下他们。在问之前思考一下你要问的问题，不要问太蠢的问题。关于 **如何提问**这个问题，可以参见https://ustb-806.github.io/DigitalLogic_Info/#/module/smart_of_asking_question

0.1 | 任务要求与实验原理

说回正题，数字逻辑实验课程围绕着 FPGA 开发板展开，你需要做的就是通过编写 Verilog 设计电路，用 Vivado 将你设计的电路变成可以下载到 FPGA 开发板上的比特流，从而操控板上的外设。为了帮助同学们更好的理解如何利用 Vivado 开发 FPGA 这一过程，我们准备了 Lab0，一个目前未得到官方承认，不计入实验分数，但是或许似乎还有那么点用的实验。这个实验的描述如下。

请你用开发板完成这样一个电路：用户可以通过板上的一个拨码开关，操控板上的一个 LED 灯。

听起来非常简单，以你自初中以来 6 年电路经验，你一定明白，最简单的方式就是——把拨码开关和 LED 灯连起来。事实上，LED 灯，拨码开关，FPGA 的关系可以通过图0.2来解释。

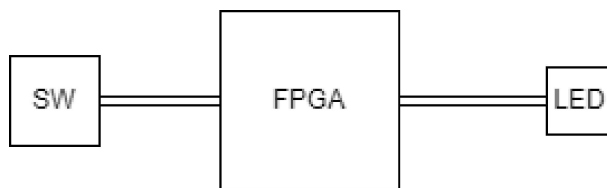


图 0.2: LED 与 SW 连接图

开关和 LED 灯之间通过一个 FPGA 这样一个黑盒连接，而 FPGA 作为一块可编程逻辑芯片，你只要可以任意配置其内部的逻辑。所以本实验需要编写的 Verilog 代码如下

```
1 module connect(  
2     input  wire sw,  
3     output wire led  
4 );  
5     assign led = sw;  
6 endmodule
```

这或许会是你在数字逻辑课程里见过的最简单的代码，他只是简单将两个管脚连接起来。但是 sw 和 led 着两个信号又是怎么来的呢？connect 这个模块不被任何其他模块所实例化，我们称之为顶层模块。在 FPGA 开发中，顶层模块的接口一般连接到 FPGA 芯片的管脚上，而这些管脚又连接着不同的外设，所以 sw 和 led 事实上对应着 FPGA 芯片的某个管脚。我们把这块 FPGA 芯片从这块价值 1000 元的开发板“抠”下来。



图 0.3: FPGA 芯片

可以看到，FPGA 看起来就是一个平平无奇的小方片，上面有着很多很多的管脚，每个管脚本来都连接着开发板上的一个触点，通过触点连接到各种各样的外设，比如说拨码开关，LED 灯。板上的按键。FPGA 的管脚如此之多，几乎每个管脚都有其用武之地，那么哪个管脚才对应着 LED 灯和拨码开关呢？关于这点，我们可以打开 ego1 开发板的手册。其中有这样两个表格。

名称	原理图标号	FPGA IO PIN
SW0	SW_0	R1
SW1	SW_1	N4
SW2	SW_2	M4
SW3	SW_3	R2
SW4	SW_4	P2
SW5	SW_5	P3
SW6	SW_6	P4
SW7	SW_7	P5
SW8	SW_DIP0	T5
	SW_DIP1	T3
	SW_DIP2	R3
	SW_DIP3	V4
	SW_DIP4	V5
	SW_DIP5	V2
	SW_DIP6	U2
	SW_DIP7	U3

名称	原理图标号	FPGA IO PIN	颜色
D1_0	LED1_0	K3	Green
D1_1	LED1_1	M1	Green
D1_2	LED1_2	L1	Green
D1_3	LED1_3	K6	Green
D1_4	LED1_4	J5	Green
D1_5	LED1_5	H5	Green
D1_6	LED1_6	H6	Green
D1_7	LED1_7	K1	Green
D2_0	LED2_0	K2	Green
D2_1	LED2_1	J2	Green
D2_2	LED2_2	J3	Green
D2_3	LED2_3	H4	Green
D2_4	LED2_4	J4	Green
D2_5	LED2_5	G3	Green
D2_6	LED2_6	G4	Green
D2_7	LED2_7	F6	Green

图 0.4: 手册中的表格

其中第一列‘名称’，这是手册给开关或 LED 起的名字，第二列‘原理图标号’，这是电路原理图给开关或 LED 起的名字。电路原理图描述了电路元件如何与其他部件相连，比如下图就是拨码开关的原理图:

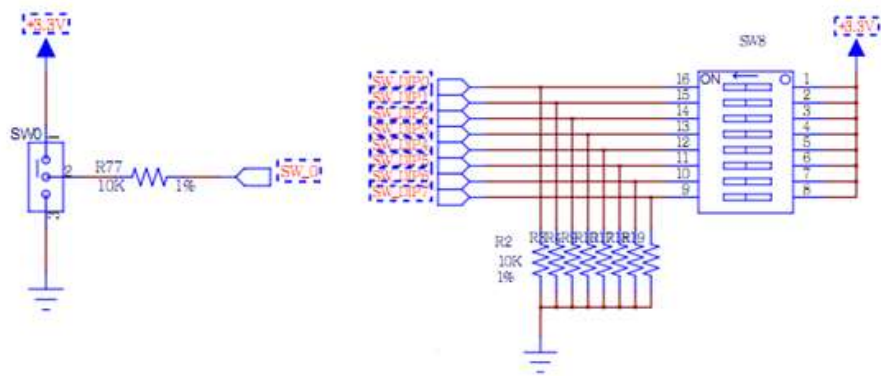


图 0.5: 拨码开关的原理图

而第三列‘FPGA IO PIN’ 则是 FPGA 厂商为 FPGA 上的针脚起的名字，FPGA 上的管脚采用类似棋盘的命名方式，比如第 K 行的 3 列就对应了 FPGA 的 K3 管脚，他连接着一个板上的一个 LED 灯。

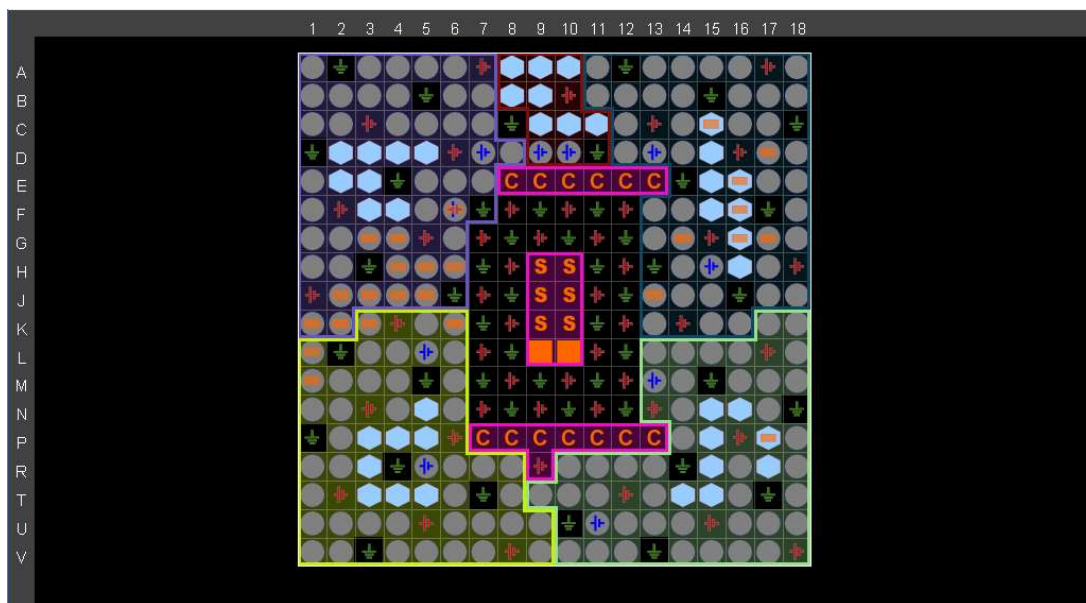


图 0.6: 管脚图

实验手册中的这些表格描述了 FPGA 芯片管脚与外设管脚的连接关系，所以本实验的主要内容事实上是根据实验手册建立这样的映射关系。

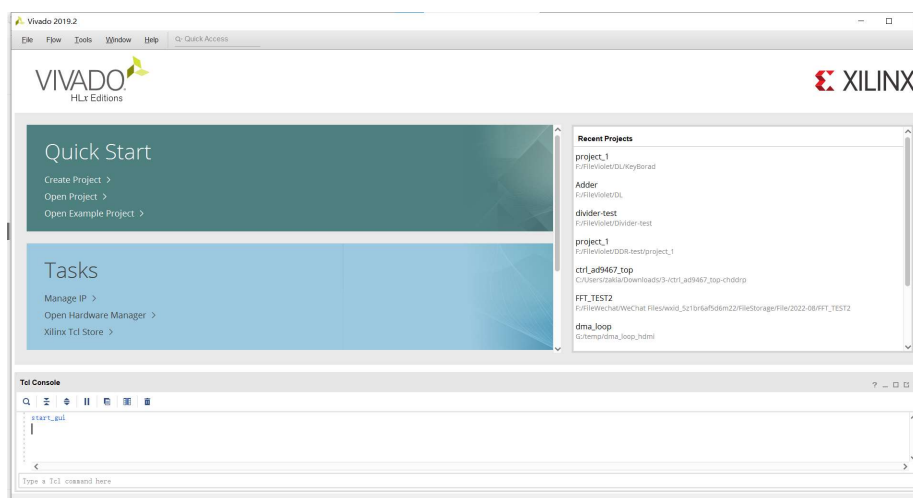
手册之中还有更多的外设与 FPGA 管脚的映射关系，接下来的实验当中，你还会用到各种各样的外设，这本简短的手册还将发挥更大的作用。

0.2 | 实验步骤

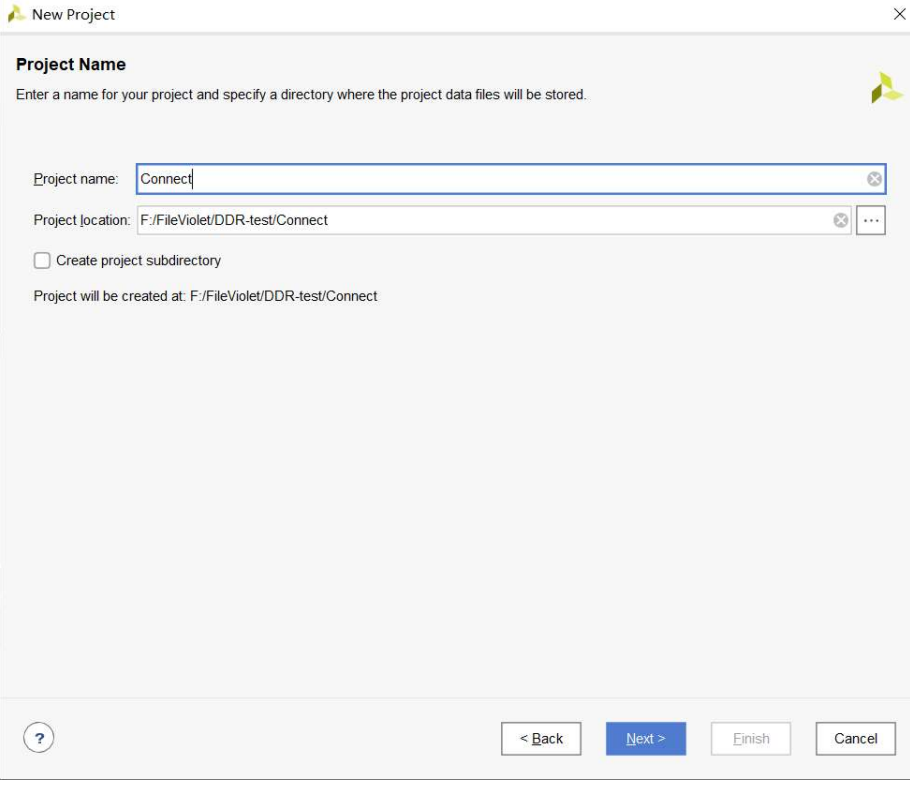
1. 安装 Vivado

请保证你的计算机名称、用户名、用户文件夹下的名称 **没有除英文字母、下划线以外的任何字符**，否则你将会遇到各种问题

2. 首先，打开 Vivado, 在 Quick Start 部分点击 Create Project



3. 点击 Next 进入选择路径的页面，选择项目名称和任意你喜欢的路径，但是路径 **不要有除英文字母、下划线以外的任何字符** 也不要过长。



New Project

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

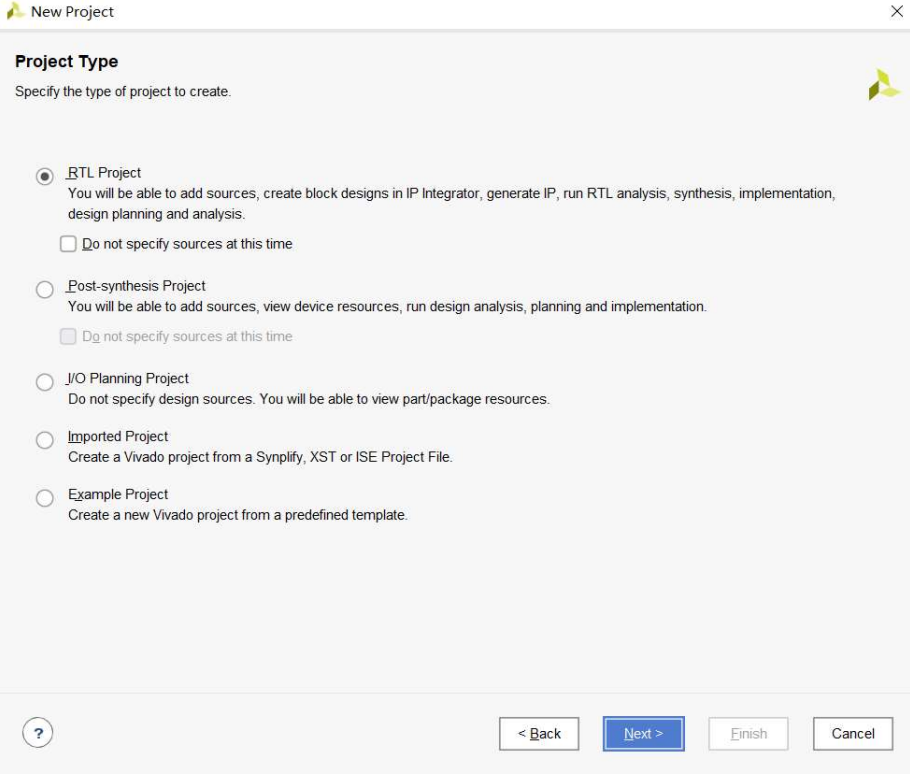
Project name:

Project location:

☐ Create project subdirectory

Project will be created at: F:/FileViolet/DDR-test/Connect

4. 选择项目类型 ‘RTL Project’



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**
You will be able to add sources, view device resources, run design analysis, planning and implementation.
☐ Do not specify sources at this time

☐ **I/O Planning Project**
Do not specify design sources. You will be able to view part/package resources.

☐ **Imported Project**
Create a Vivado project from a Synplify, XST or ISE Project File.

☐ **Example Project**
Create a new Vivado project from a predefined template.

5. 添加源文件和约束（可以跳过）

The image displays two sequential screenshots of a 'New Project' wizard interface.

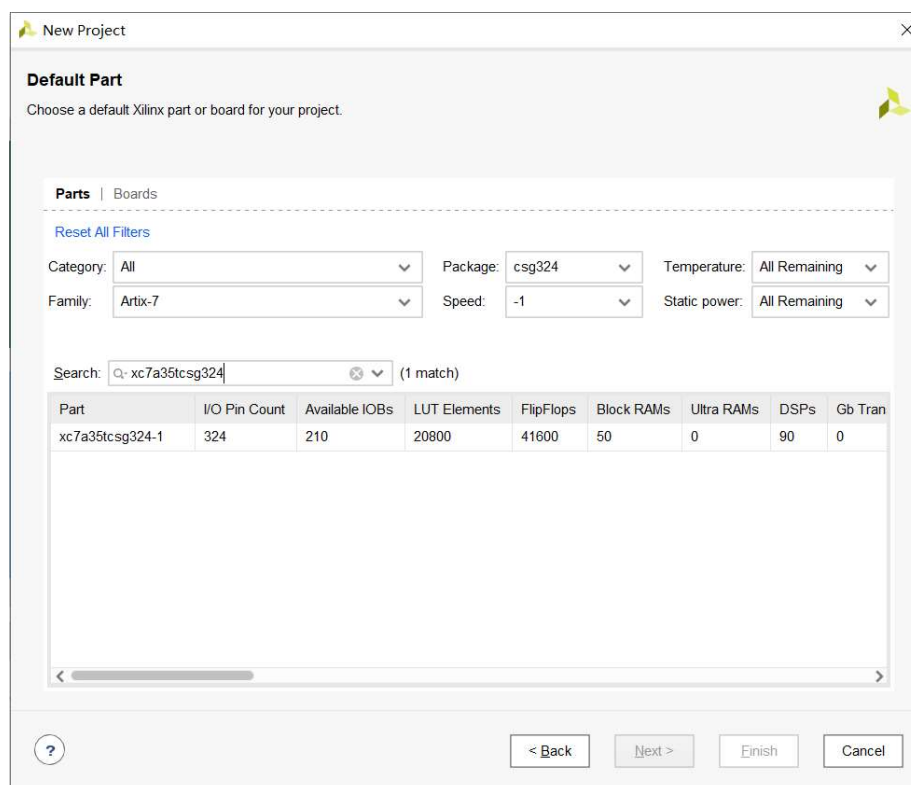
Top Screenshot: Add Sources

- Title:** New Project
- Section:** Add Sources
- Description:** 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.
- Buttons:** Add Files, Add Directories, Create File
- Options:**
 - ☒ Scan and add RTL include files into project
 - ☐ Copy sources into project
 - ☒ Add sources from subdirectories
- Target language:** Verilog
- Simulator language:** Mixed
- Navigation:** < Back, Next >, Finish, Cancel

Bottom Screenshot: Add Constraints (optional)

- Title:** New Project
- Section:** Add Constraints (optional)
- Description:** Specify or create constraint files for physical and timing constraints.
- Buttons:** Add Files, Create File
- Options:**
 - ☐ Copy constraints files into project
- Navigation:** < Back, Next >, Finish, Cancel

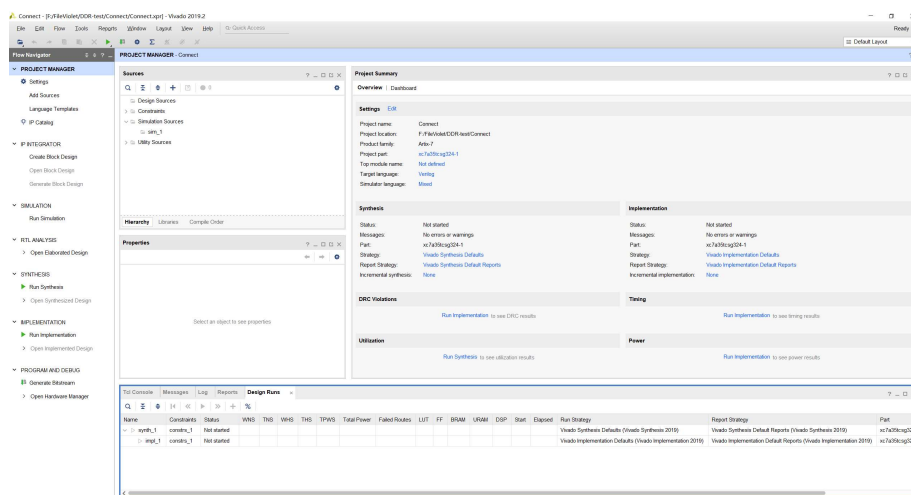
6. 接下来是选择板子的型号，不同的 FPGA 的资源数量，速度，管脚功能都是不同的，**一定要选择本课程使用的 FPGA 芯片 ‘xc7a35tcsg324-1’。**



7. 点击 Finish，完成工程创建。

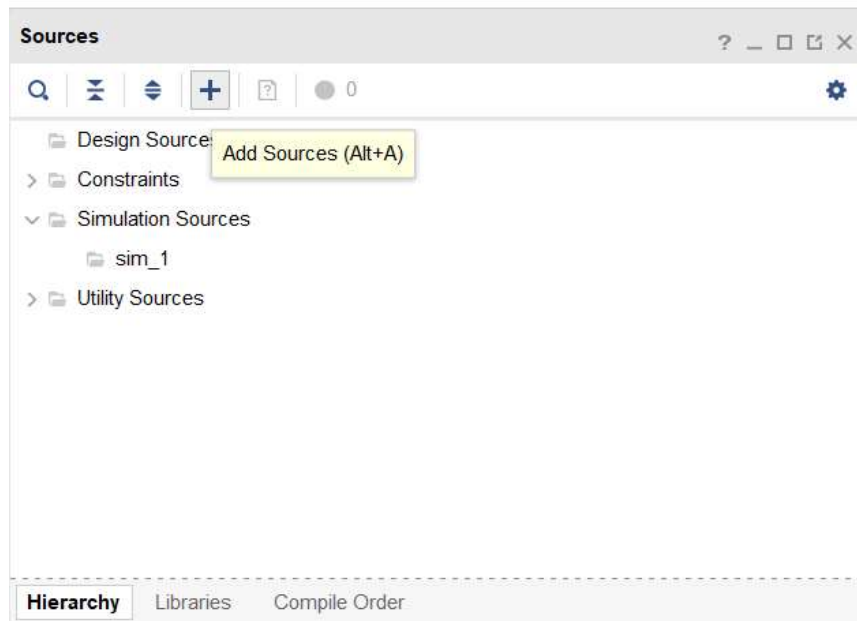
恭喜你完成第一阶段——创建工程

8. 进入 vivado，界面如下所示，



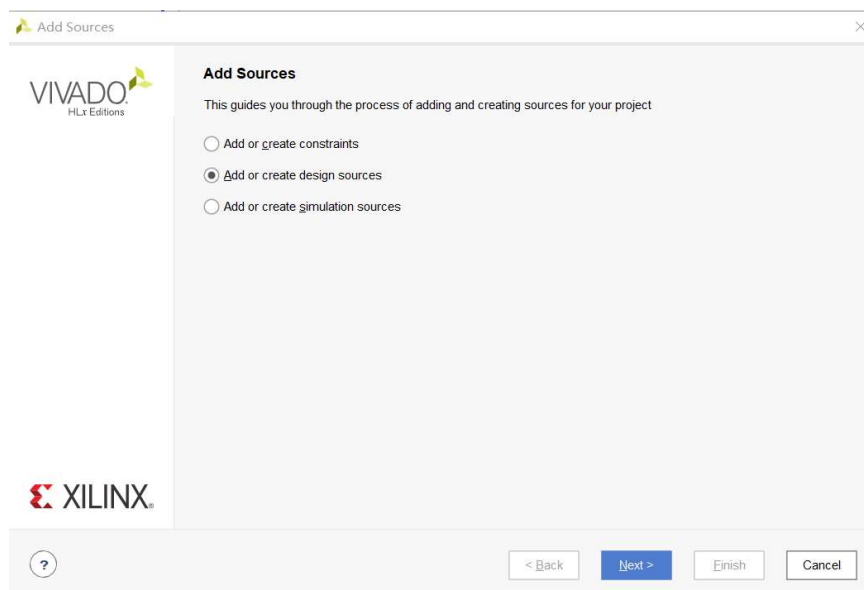
9. 创建设计文件

[a] 点击下面的加号

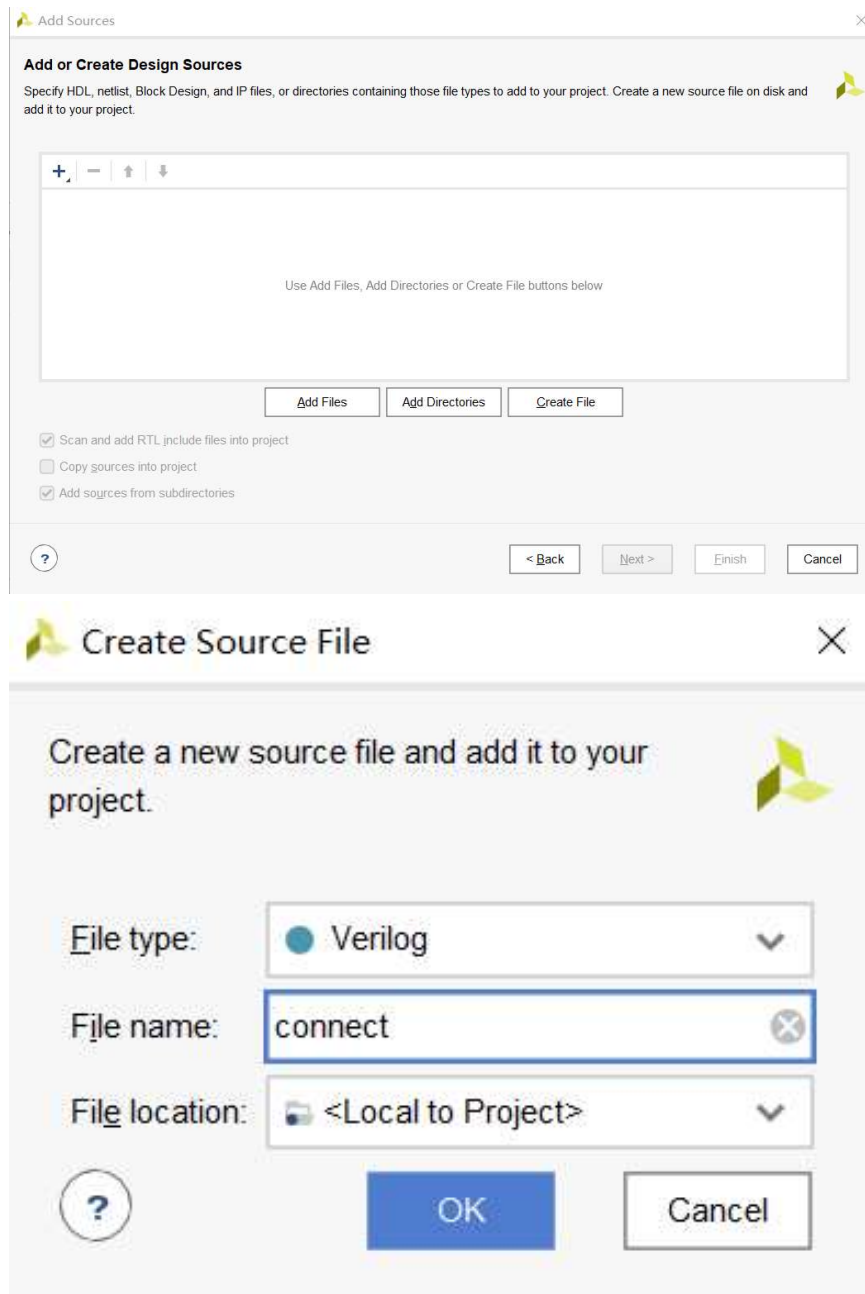


[b] 点开，我们会看到下面的界面，其中包括三个部分，分别是

- 创建或添加约束（Constraints），添加 IO 管脚约束和时序约束，这些我们会在后面谈到。
- 创建或增加设计文件（design sources），设计电路的 Verilog 文件都是设计文件
- 创建或增加仿真文件（simulation sources），主要是电路的仿真激励。

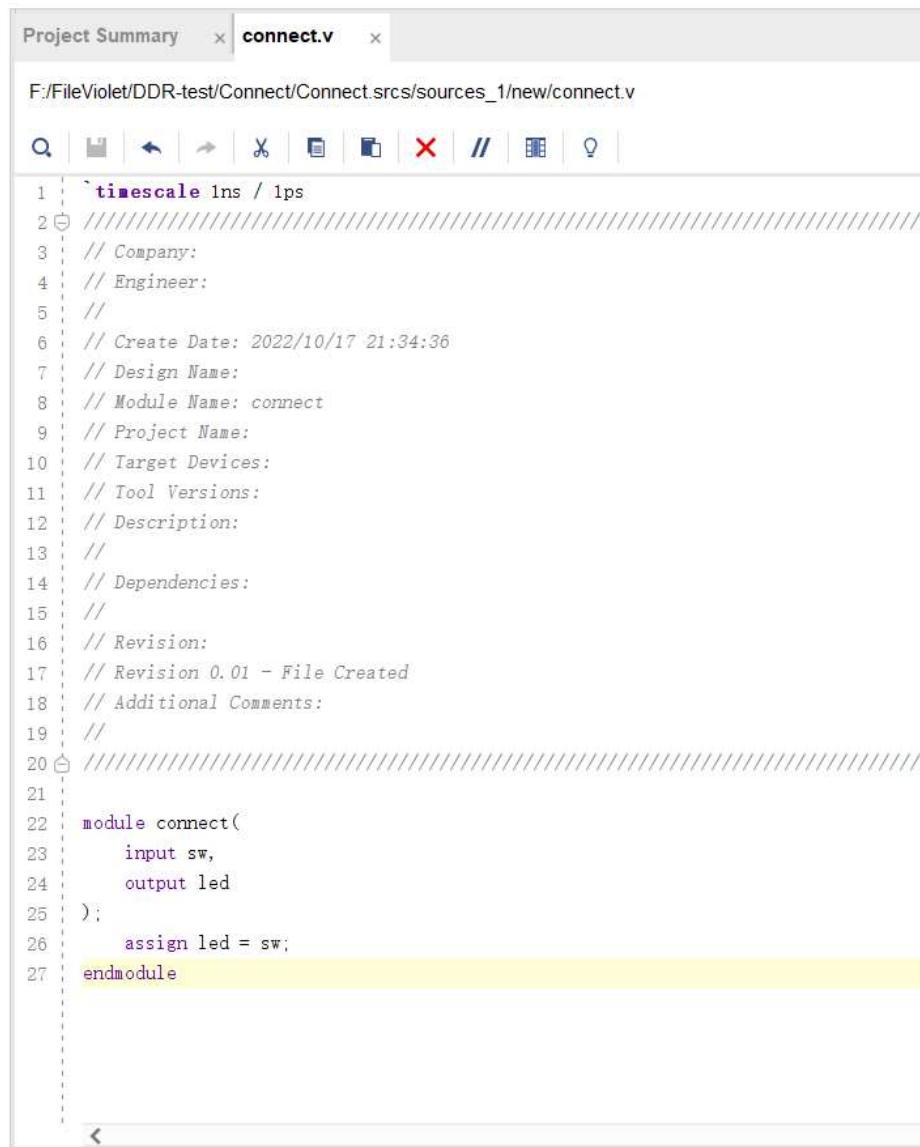


[c] 你可以选择添加现有文件或目录，或者你也可以创建新文件。这里我们创建一个新文件



[d] 添加完文件后选择 Finish

10. 编写文件代码，把上面的代码放到 Vivado 中。



```
Project Summary x connect.v x
F:/FileViolet/DDR-test/Connect/srcs/sources_1/new/connect.v

1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: 2022/10/17 21:34:36
7  // Design Name:
8  // Module Name: connect
9  // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////
21
22 module connect(
23     input sw,
24     output led
25 );
26     assign led = sw;
27 endmodule
```

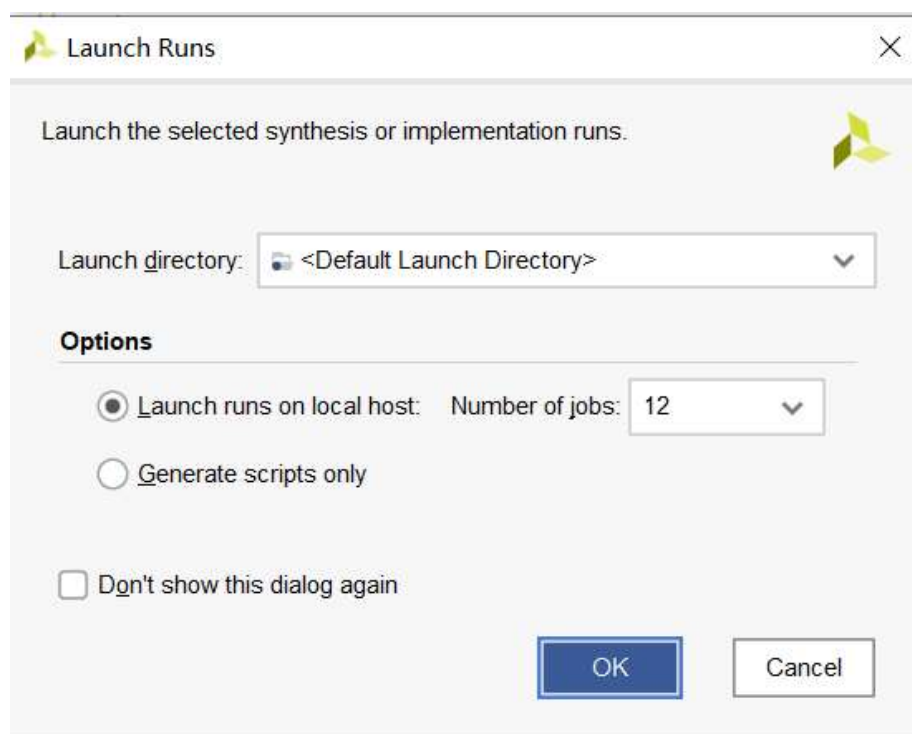
11. 接下来点击综合，将这个 Verilog 文件变成网表文件。这里通过在“Flow Navigator”栏中的“Synthesis”下点击“Run Synthesis”。右上角的进度条“Running synth_design”指示正在对工程进行综合。

如果你是在学校的电脑上进行综合、实现，那么你将会度过一段漫长的时光... 当然如果时间太长了可能是你的某些地方出了问题，你可以尝试重启或者删除相关文件重新开始

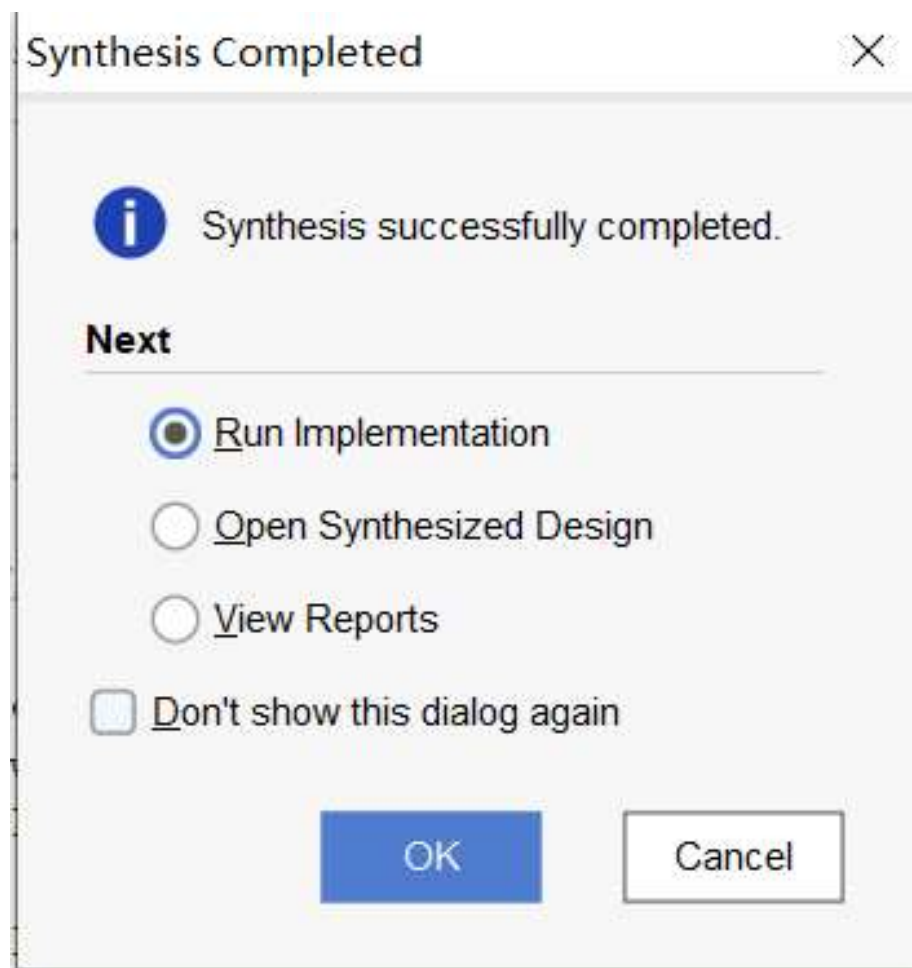
▼ SYNTHESIS

▶ [Run Synthesis](#)

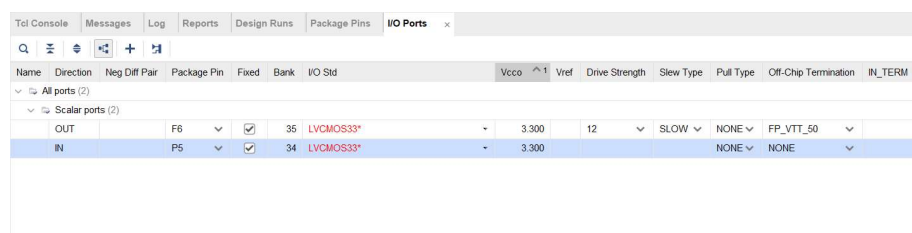
> [Open Synthesized Design](#)



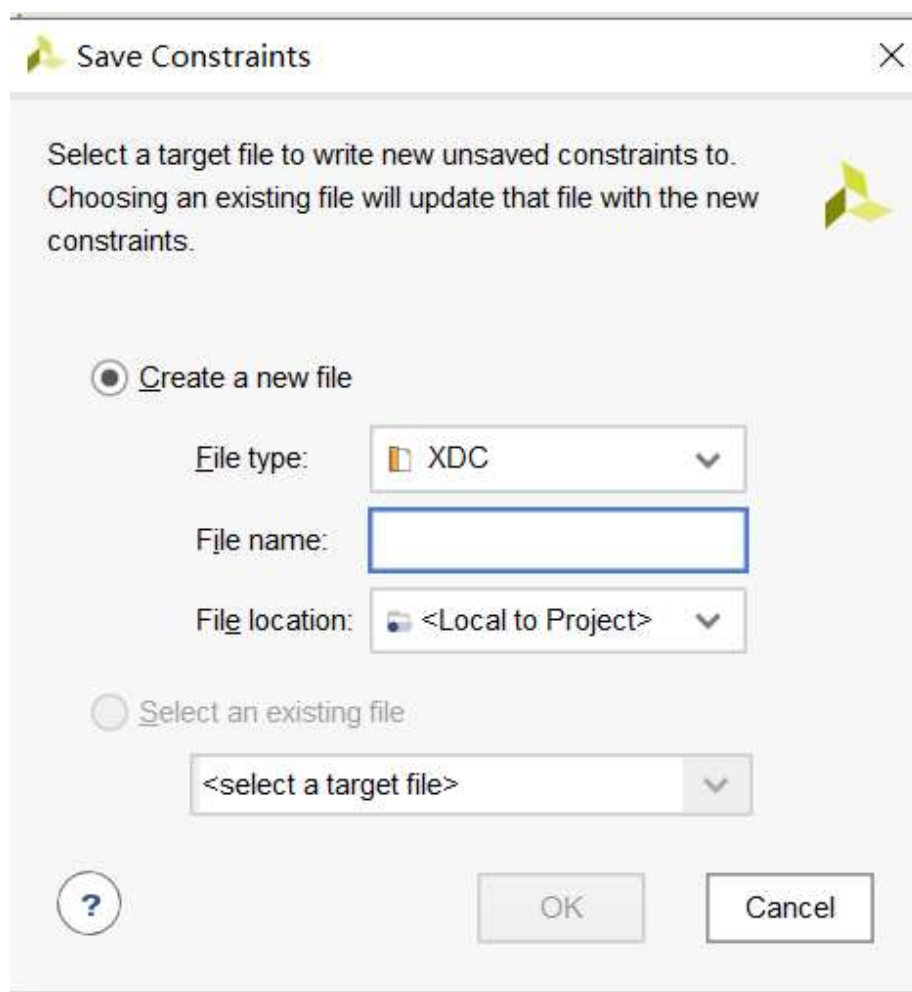
12. 等待综合结束，下面的界面弹出，在弹出的对话框中选择“Open Synthesized Design”，并点击“OK”，打开综合后的工程。



13. 我们要对 sw 和 led 进行绑定，将其绑定到对应的实际管脚上。对管脚的物理属性进行约束，在“I/O Ports”窗口中对输入输出信号添加管脚约束。首先在“I/O Std”一栏通过下拉按钮选择“LVCMOS33”，将所有信号的电平标准设置 3.3V。在“Package Pin”一栏分配各个信号在 FPGA 芯片上引脚的位置，各信号的具体位置可查看板卡的原理图。



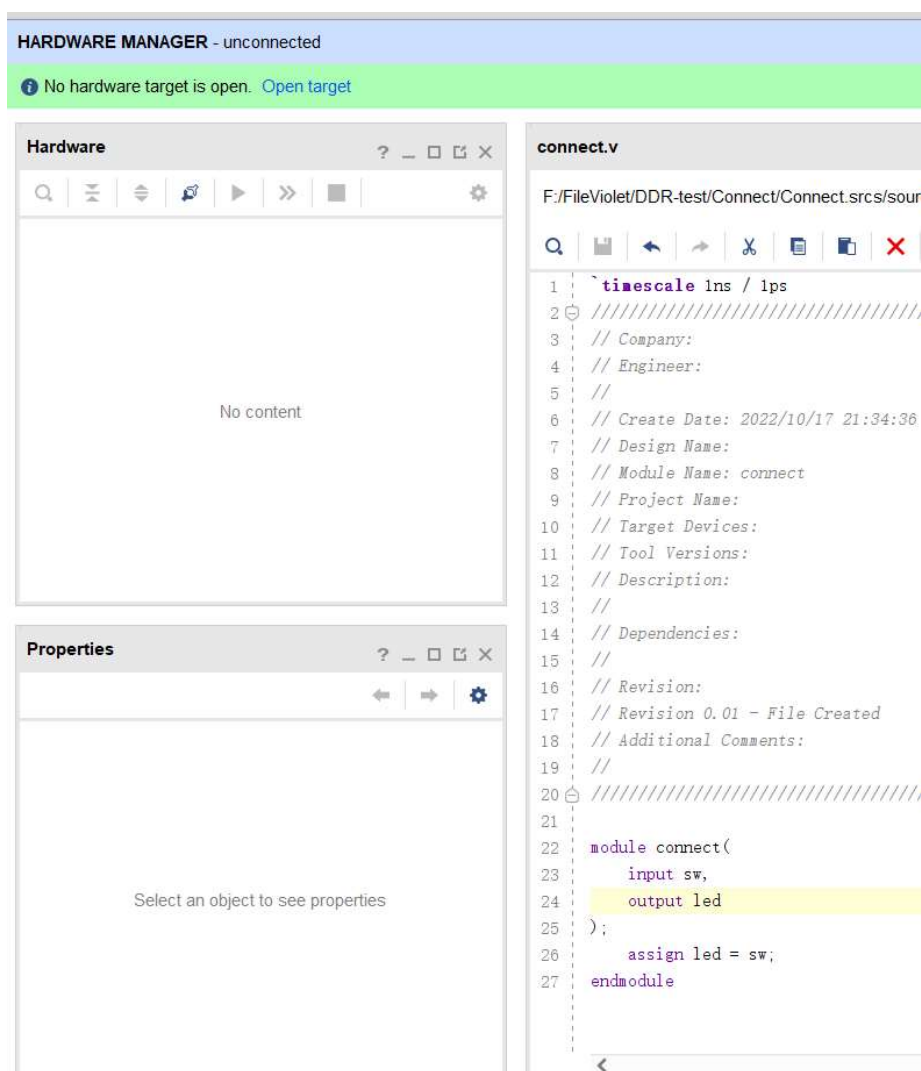
14. 点击保存，会弹出一个保存约束文件的选项。这个文件是实际上起到了对 IO 进行约束的文件。



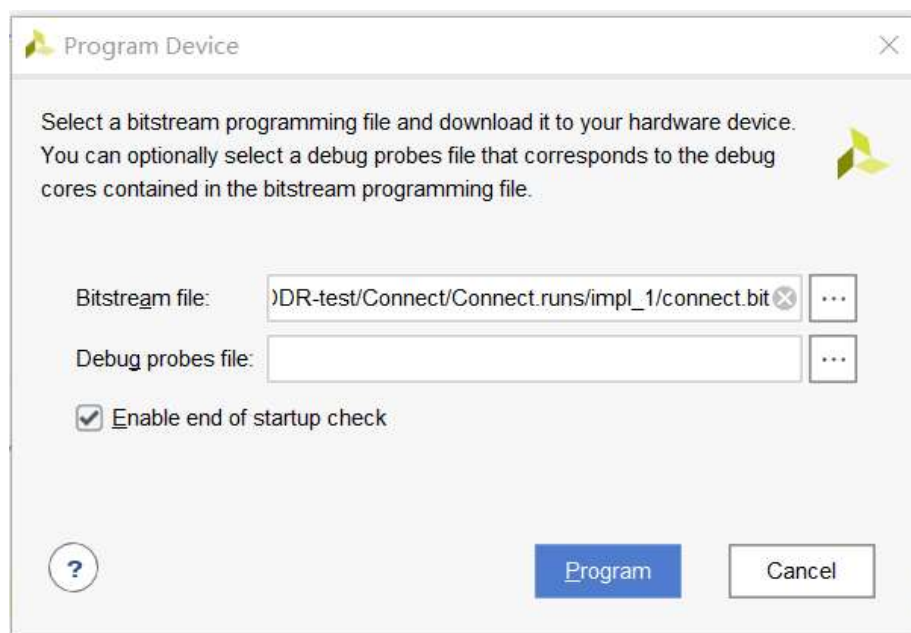
15. 完成后点击 Implementation 和 Generate Bitstream, 生成 bit 流

慢慢等吧，不要着急

16. 用 USB Type C 连接板卡和电脑，打开板卡开关。打开 Hardware Manager，点击 Open Target, Auto Connect



17. 点击 Program device



18. 拨动板卡的对应的开关，看看是不是能正常控制 LED 灯的亮灭呢

0.3 | 查看 RTL 原理图

正如我们之前说的，Verilog 代码的编写本质上是“画电路”，但是如何才能看到我们画出来的电路长什么样子呢？Vivado 为我们提供了一个查看原理图的功能，可以帮助我们了解工具将你的代码转换成了什么样子的电路。

按照上面描述的流程，创建一个新的工程，并添加一个设计文件，将下述代码粘贴进去。下述代码实现了一个 $d = (a \& b) | c$ 的功能

Listing 1: 示例模块（test）

```
1  `timescale 1ns / 1ps //时间单位和时间精度
2
3  module test(
4      input a,
5      input b,
6      input c,
7      output d
8  );
9      wire temp;
10     and(temp, a, b);
11     or (d, temp, c);
12
13 endmodule
```

在 **进行综合** 之前，点击左侧窗口中的 RTL ANALYSIS 中的 Schematic，查看原理图。

RTL ANALYSIS

Open Elaborated Design

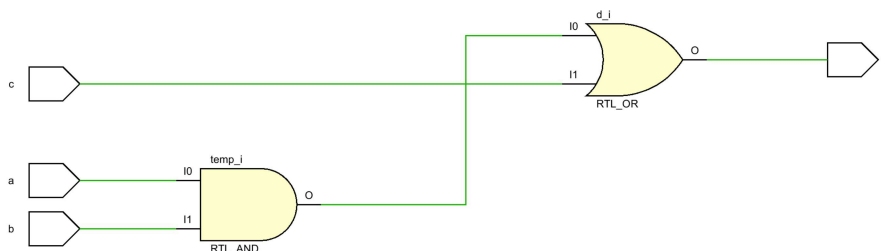
☒ Report Methodology

Report DRC

Report Noise

Schematic

在新生成的 Schematic 窗口中，可以看到生成的原理图。



可以看到，一个与门接收了信号 a 和 b，输出和 c 一起输入或门，最终赋给 output 信号 d。

0.4 | Lab0 后日谈

本实验的主要工作就是带大家熟悉一下 Vivado 的工作流程，理解一下 Vivado 设计过程中时序约束的概念。

在本实验当中呢，由于电路模块非常简单，所以我们采用直接综合实现上板的方式，但是在实际的开发中，我们的电路往往不会这么简单，综合和实现都会耗费 **巨大** 的时间，然而最终获得电路设计却往往可能存在错误，所以我们需要对电路进行仿真，通过模拟的方式，模拟输入信号，观察输出结果。后面的实验中，大家将会了解到仿真验证电路的方法。

如果遇到了错误, 请认真反复阅读讲义内容, 机器永远是对的。