

## Vivado 设计流程

简介:

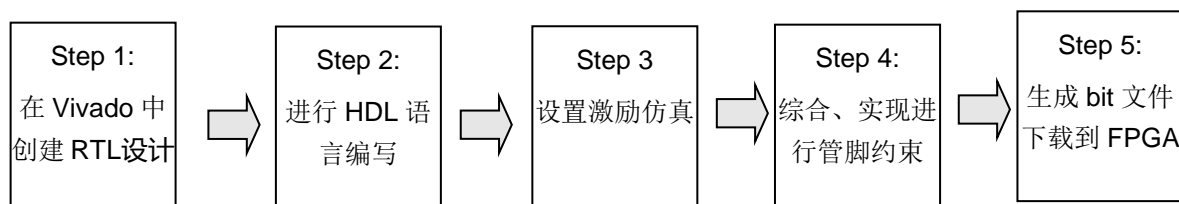
Vivado 设计分为 Project Mode 和 Non-project Mode 两种模式，一般简单设计中，我们常用的是 Project Mode。在本手册中，我们采用 Xilinx 数模混合口袋实验室，将以一个简单的实验案例，一步一步的完成 Vivado 的整个设计流程。

在本次实验中，将会学习如何使用 Xilinx Vivado 2017.1 创建、综合、实现、仿真等功能。本实验通过编写一个流水灯实验来展示使用 Xilinx Vivado 来进行基本的 FPGA 设计。

目标:

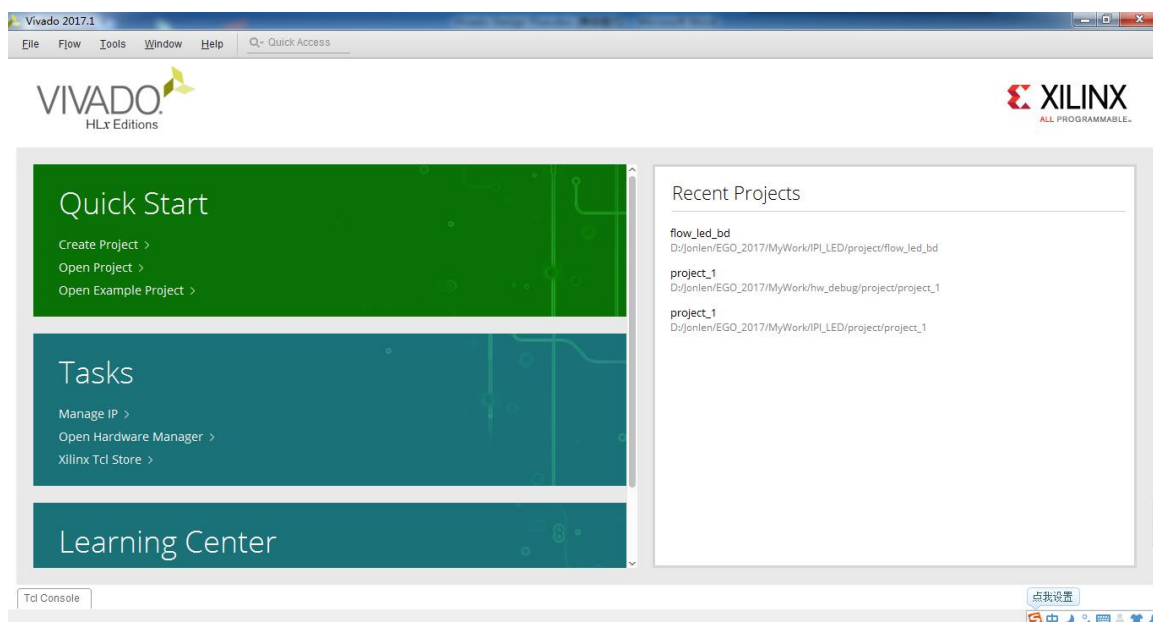
- 通过编写 HDL 文件的方式创建 Vivado 设计。
- 建立仿真
- 通过 I/O Planing 添加管脚约束
- 通过编写约束文件添加管脚约束
- 添加时序约束
- 生成 Bitstream 文件
- 将生成的 Bitstream 文件下载到 FPGA 开发板里

实验流程:

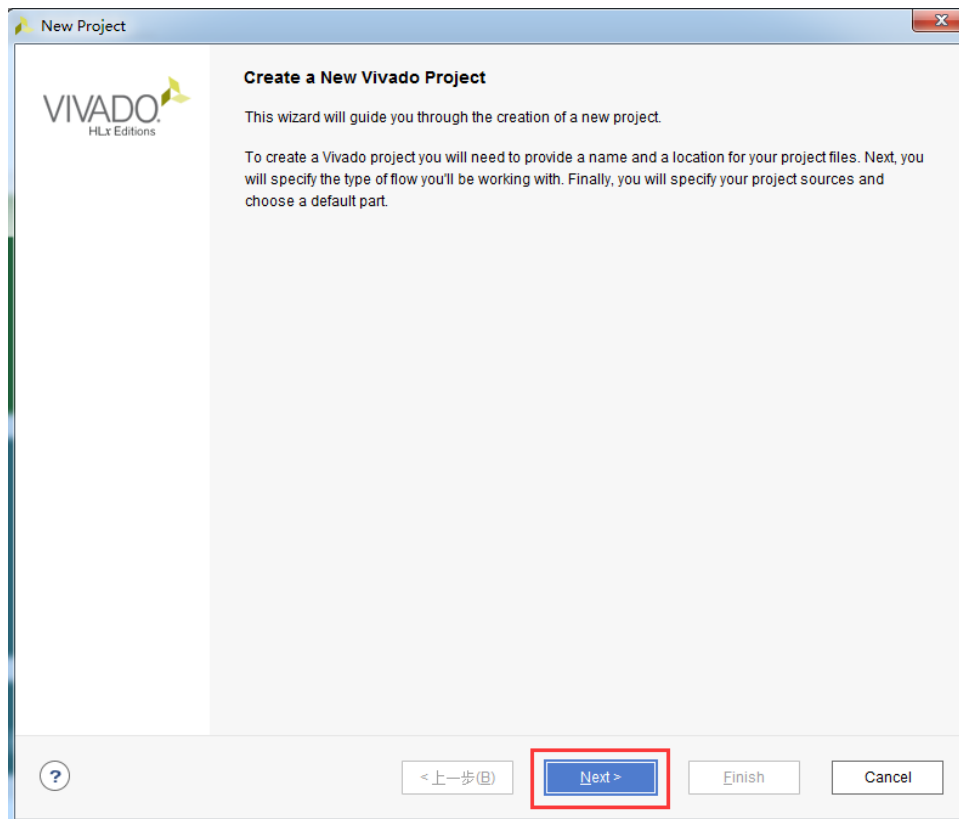


### 一、新建工程

1、打开 Vivado 2017.1 开发工具，开启后，软件如下所示:

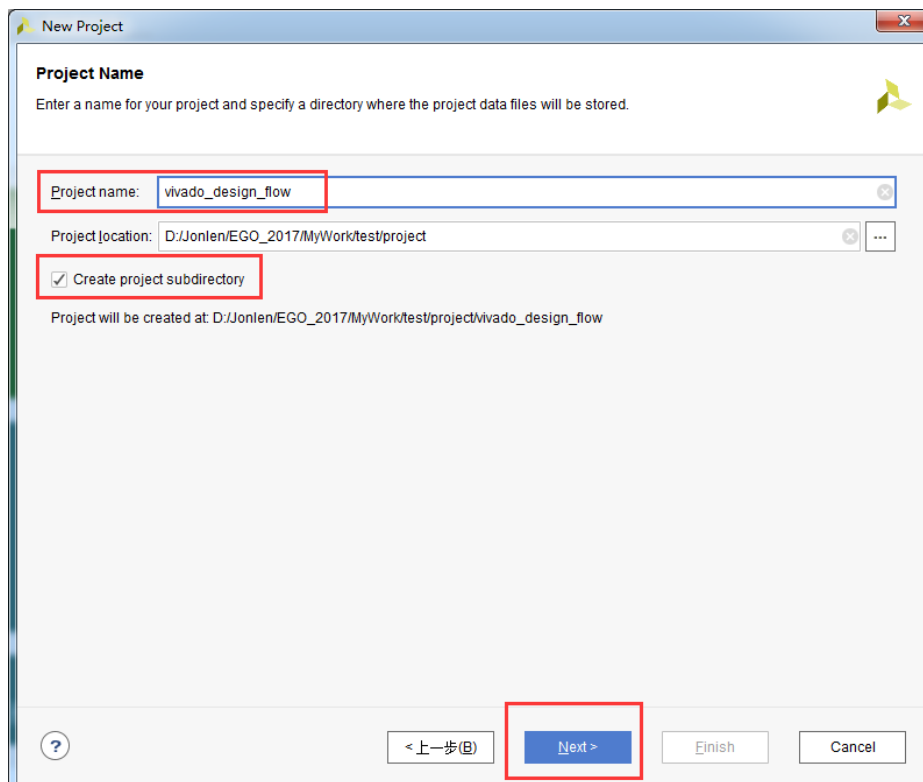


2、单击上述界面中 Create New Project 图标，弹出新建工程向导，点击 Next。

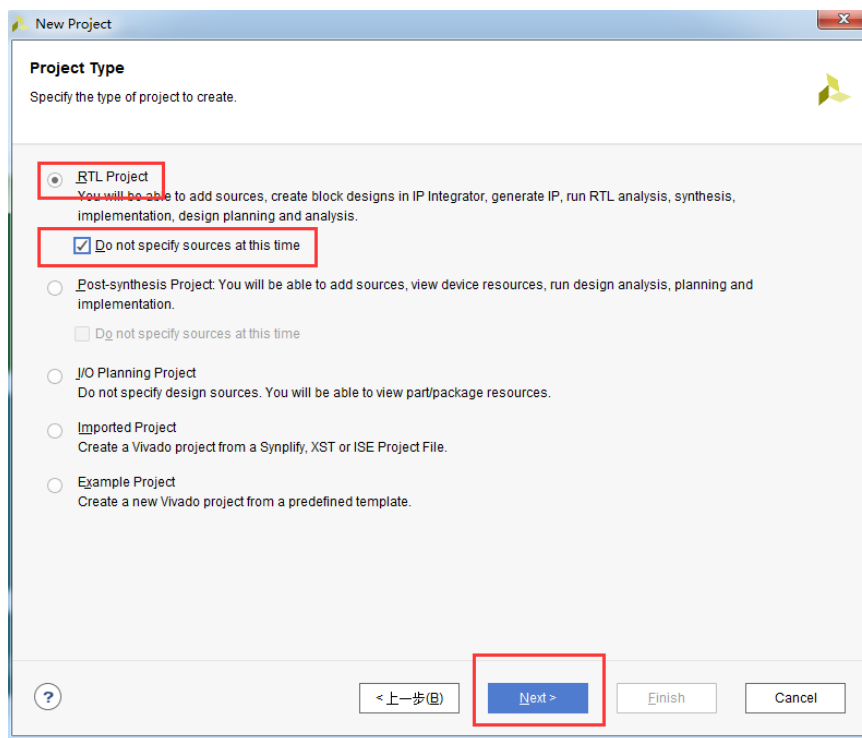


3、输入工程名称、选择工程存储路径，并勾选 **Create project subdirectory** 选项，为工程在指定存储路径下建立独立的文件夹。设置完成后，点击 **Next**。

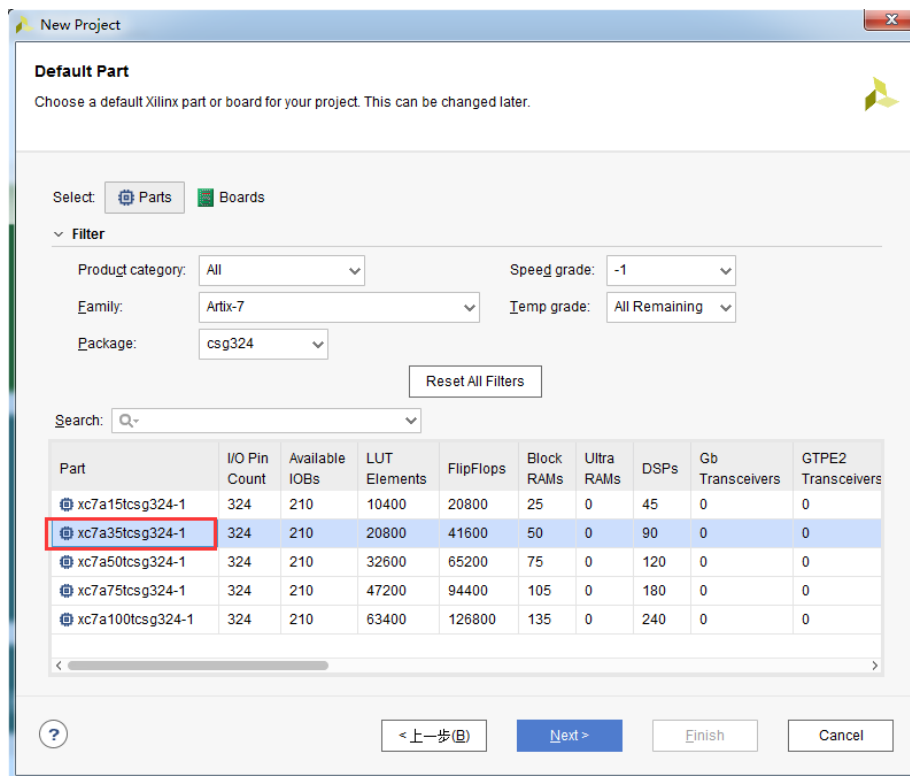
**注意：**工程名称和存储路径中不能出现中文和空格，建议工程名称以字母、数字、下划线来组成。



4、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。点击 Next。

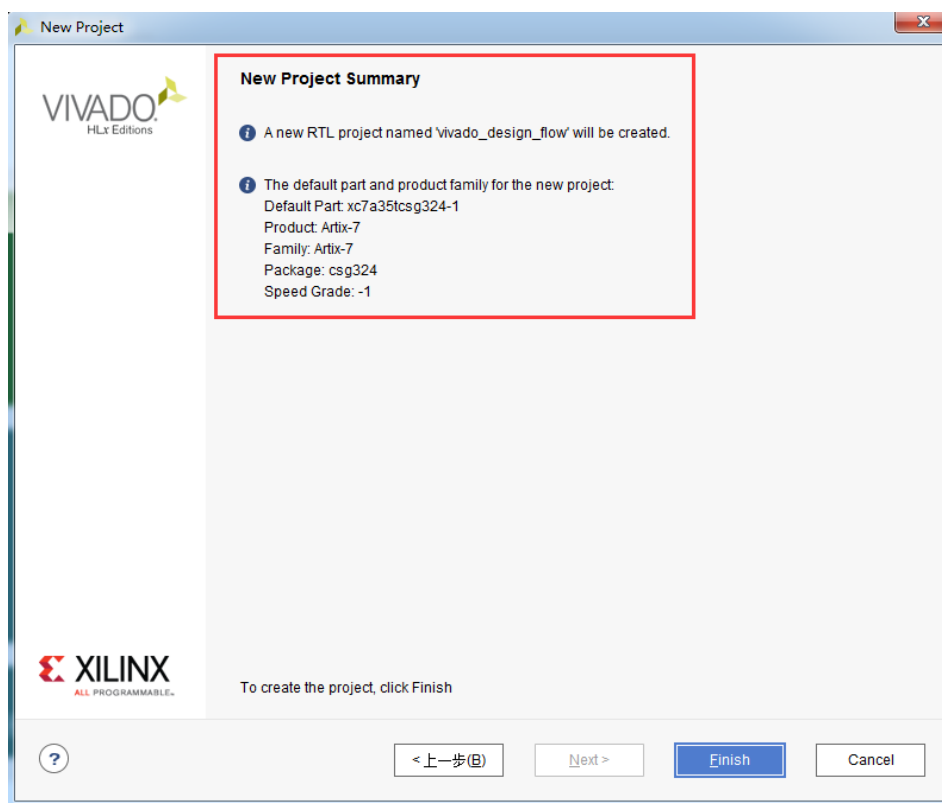


5、根据使用的 FPGA 开发平台，选择对应的 FPGA 目标器件。在本手册中，以 Xilinx 数模混合口袋实验室 为例，FPGA 采用 Artix-7 XC7A35T-1CSG324-C 的器件，即 Family 和 Subfamily 均为 Artix-7，封装形式 (Package) 为 CSG324，速度等级 (Speed grade) 为 -1，温度等级 (Temp Grade) 为 C)。点击 Next。

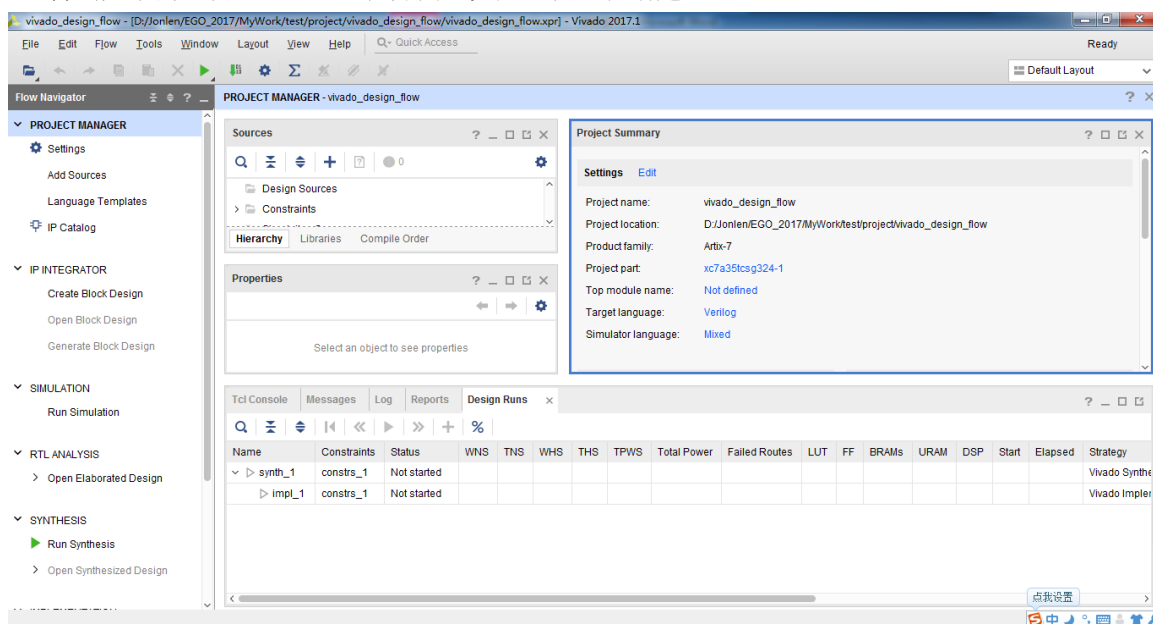


6、确认相关信息与设计所用的的 FPGA 器件信息是否一致，一致请点击 Finish，不一致，请返

回上一步修改。

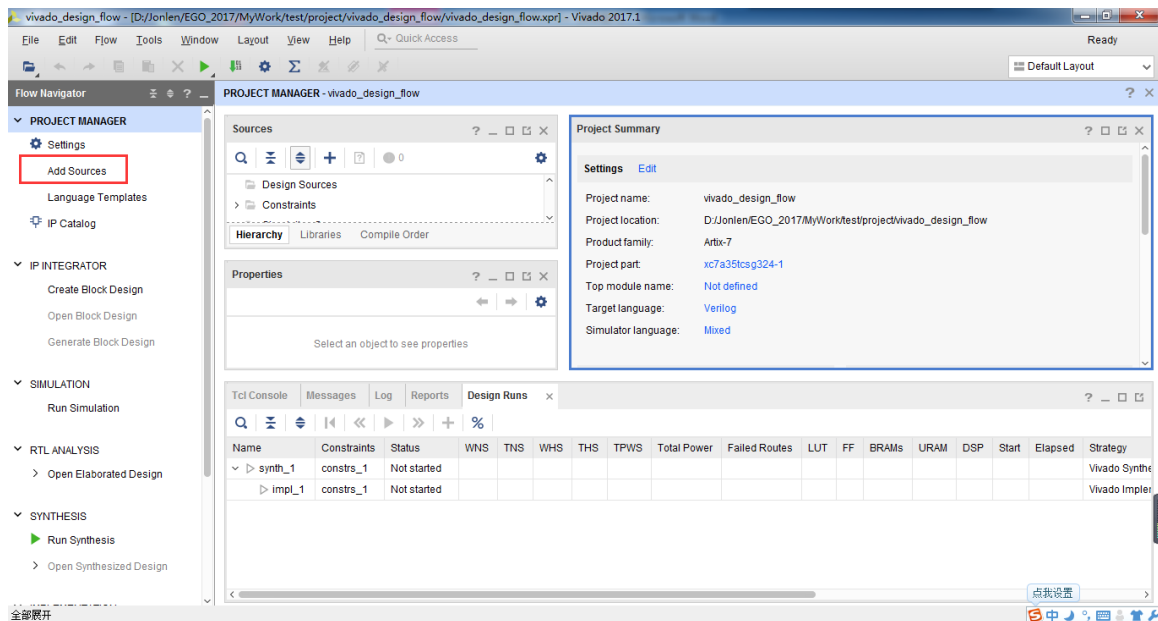


7、得到如下的空白 Vivado 工程界面，完成空白工程新建。

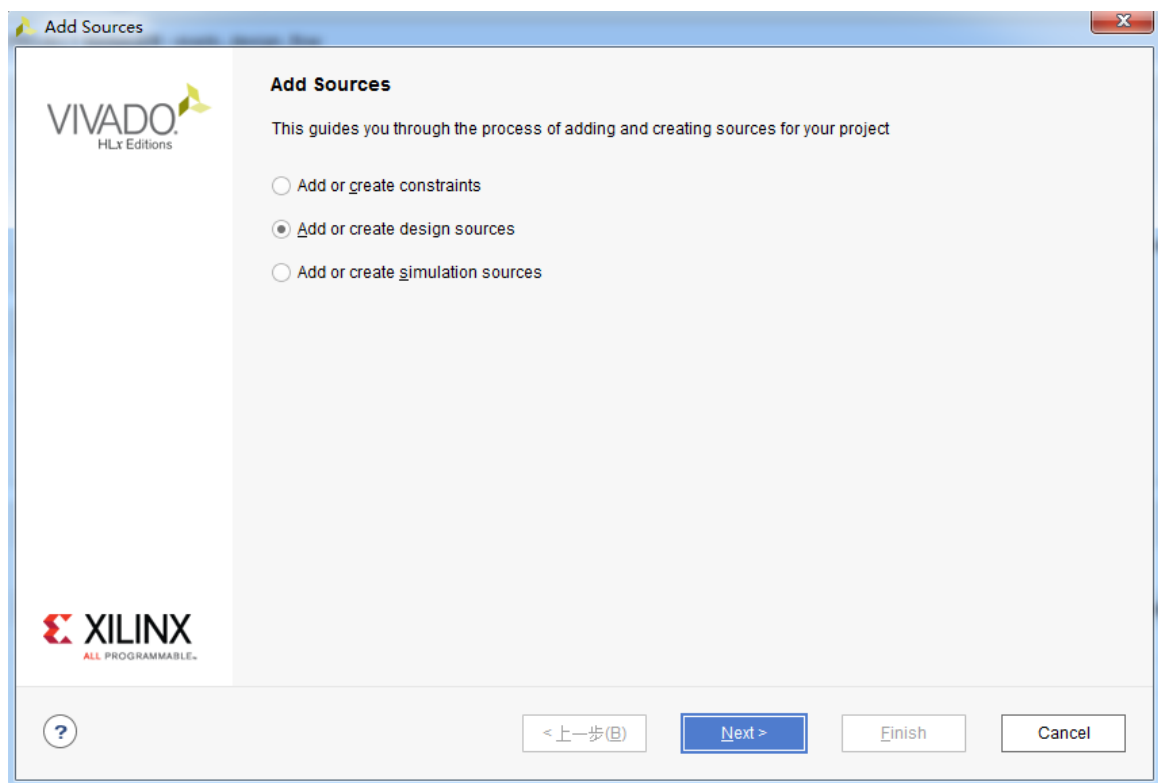


## 二、设计文件输入

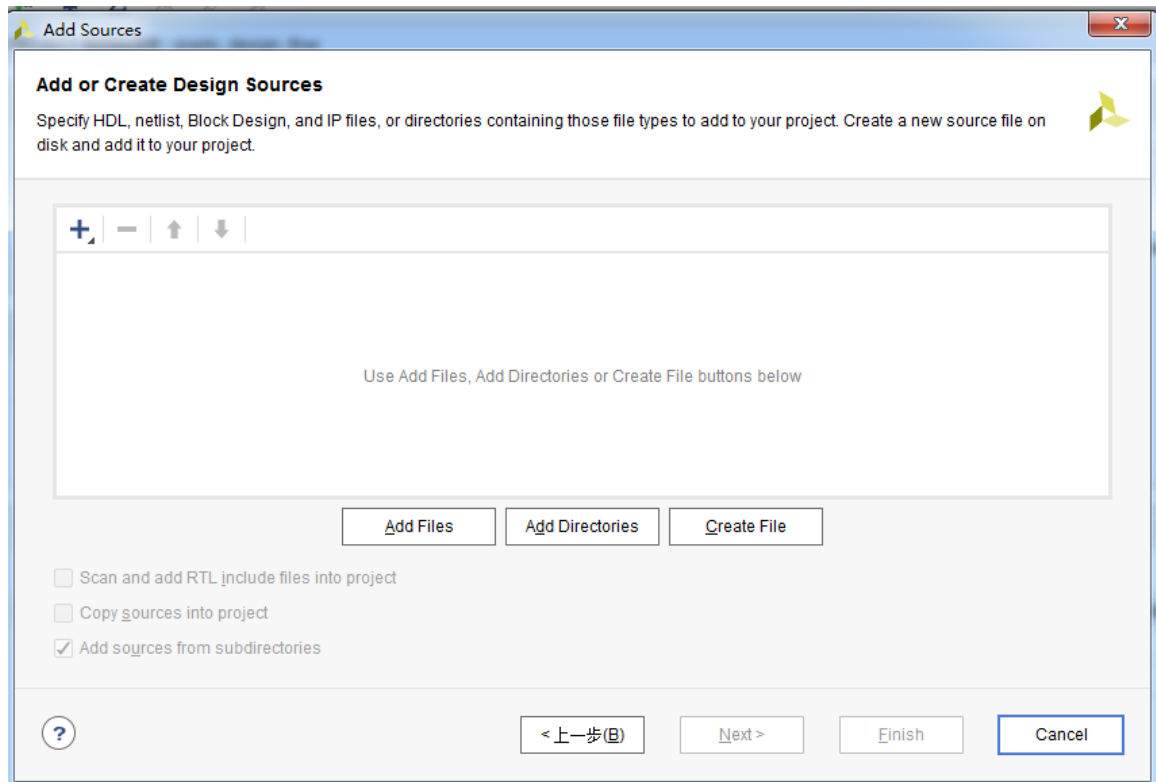
1、如下图所示，点击 Flow Navigator 下的 Project Manager->Add Sources 或中间 Sources 中的对话框打开设计文件导入添加对话框。



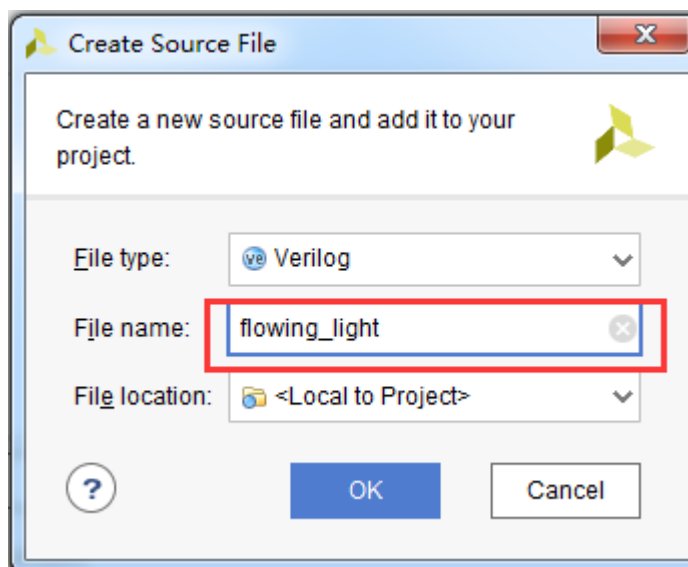
2、选择第二项 **Add or Create Design Sources**，用来添加或新建 Verilog 或 VHDL 源文件，点击 **Next**。



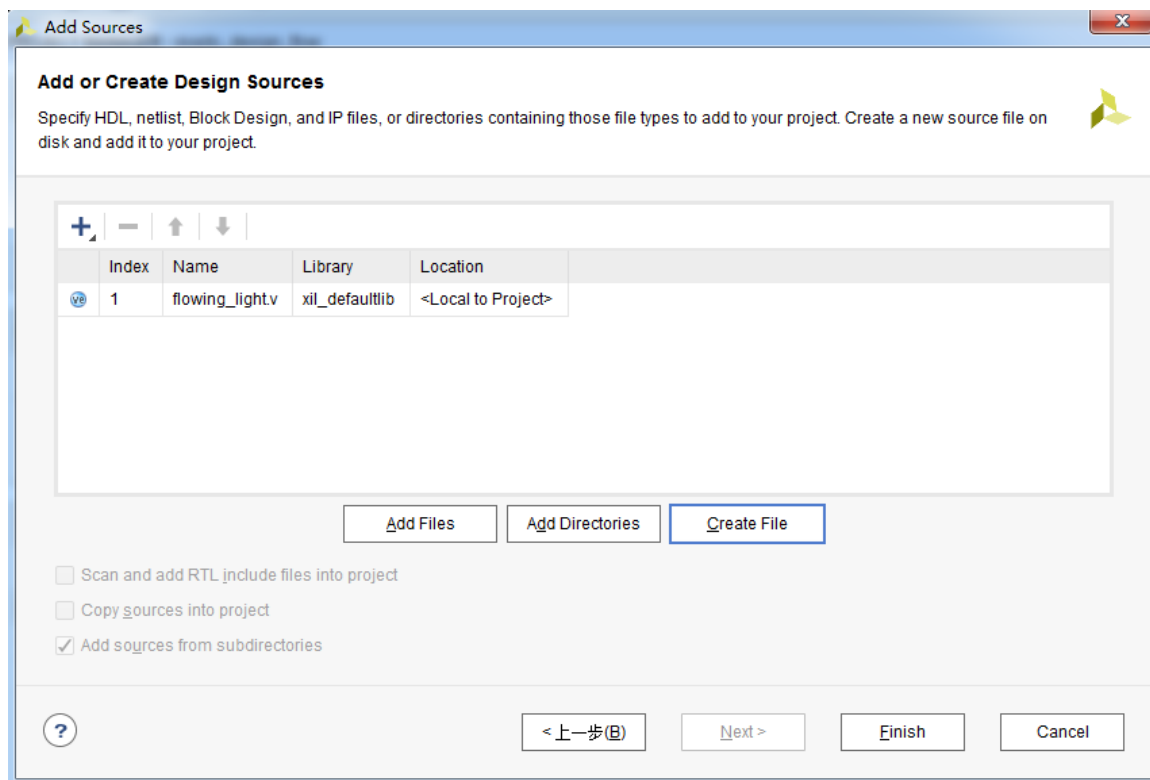
3、如果有现有的 V/VHD 文件，可以通过 **Add Files** 一项添加。在这里，我们要新建文件，所以选择 **Create File** 一项。



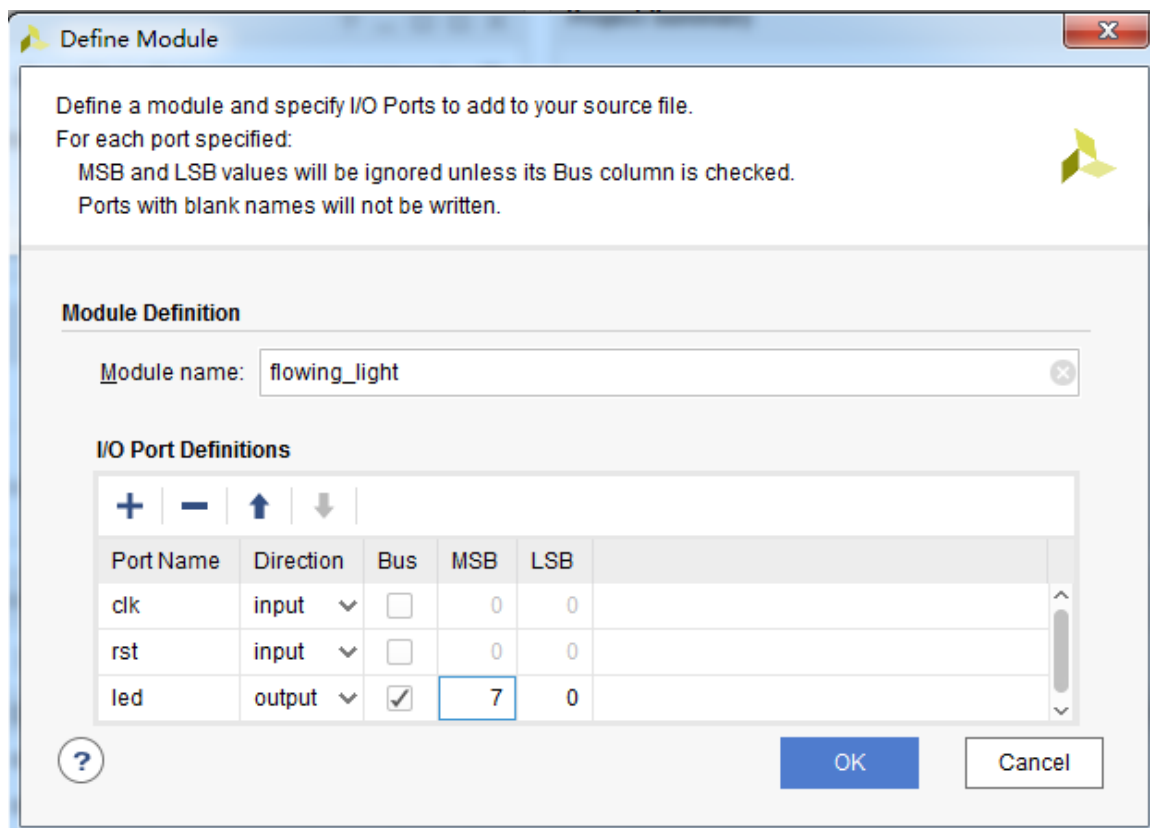
4、在 Create Source File 中输入 File Name，点击 OK。**注：名称中不可出现中文和空格。**



5、点击 Finish。



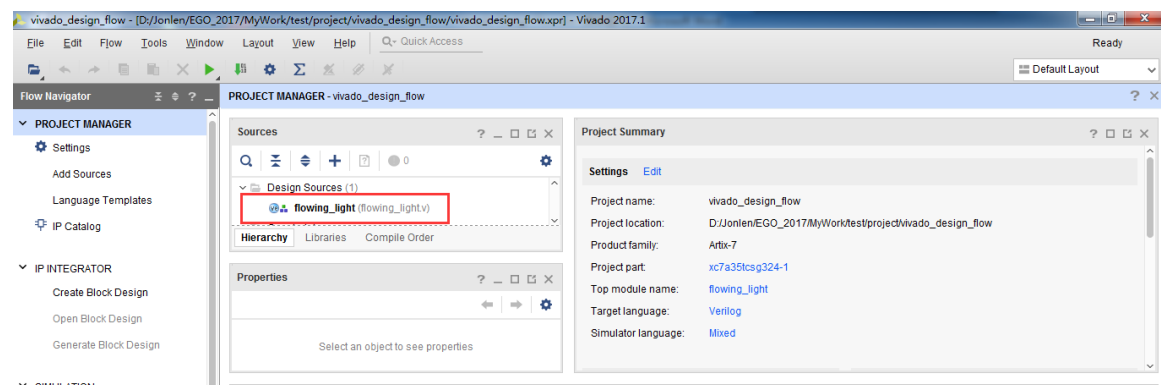
6、在弹出的 Define Module 中的 I/O Port Definition，输入设计模块所需的端口，并设置端口方向，如果端口为总线型，勾选 Bus 选项，并通过 MSB 和 LSB 确定总线宽度。完成后点 OK。



注：led 实际宽度与代码中一致，也可在代码中修改。

7、新建的设计文件（此处为 flowing\_light.v）即存在于 Sources 中的 Design Sources 中。双击

打开该文件，输入相应的设计代码。



```
`timescale 1ns / 1ps
```

```
module flowing_light(
    input clk,
    input rst,
    output [15:0] led
);
```

```
    reg [23 : 0] cnt_reg;
    reg [15 : 0] light_reg;
```

```
    always @ (posedge clk)
    begin
        if (!rst)
            cnt_reg <= 0;
        else
            cnt_reg <= cnt_reg + 1;
    end
```

```
    always @ (posedge clk)
    begin
        if (!rst)
            light_reg <= 16'h0001;
        else if (cnt_reg == 24'hffffff)
            begin
                if (light_reg == 16'h8000)
                    light_reg <= 16'h0001;
```

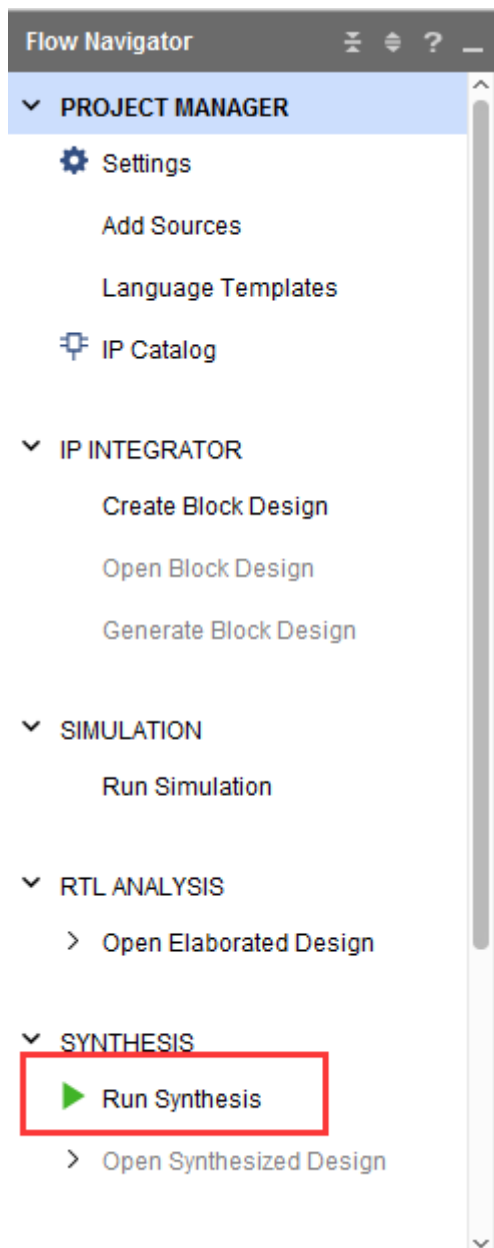


```
else
    light_reg <= light_reg << 1;
end
end
assign led = light_reg;
endmodule
```

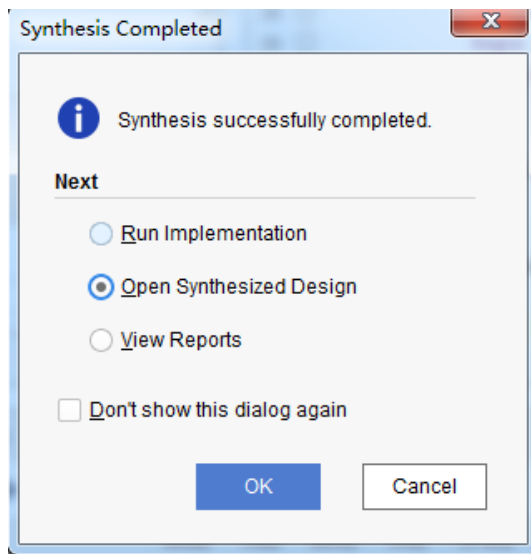
8、添加约束文件，有两种方法可以添加约束文件，一是可利用 Vivado 中 IO planning 功能，二是可以直接新建 XDC 的约束文件，手动输入约束命令。

a、先来看第一种方法，利用 IO planning。

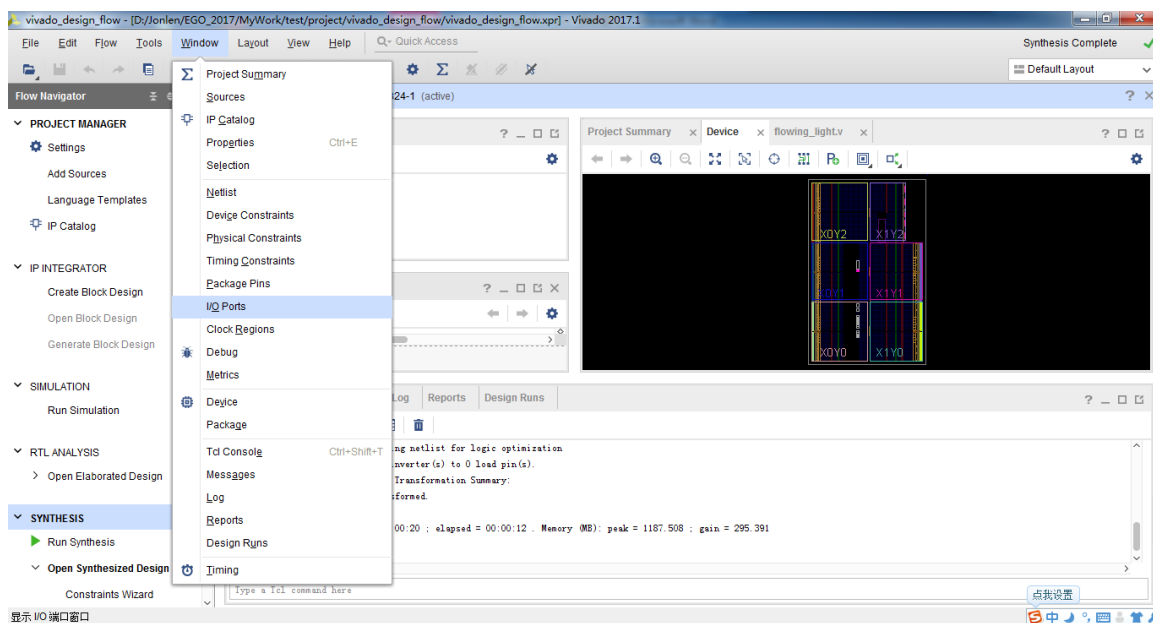
a.8.1、点击 Flow Navigator 中 Synthesis 中的 Run Synthesis，先对工程进行综合。



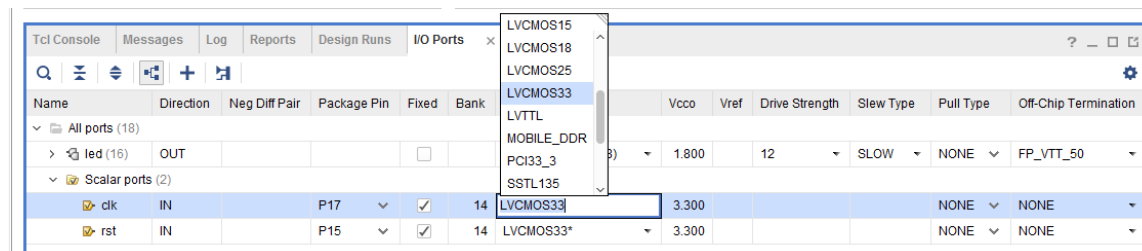
a.8.2、综合完成之后，选择 Open Synthesized Design，打开综合结果。



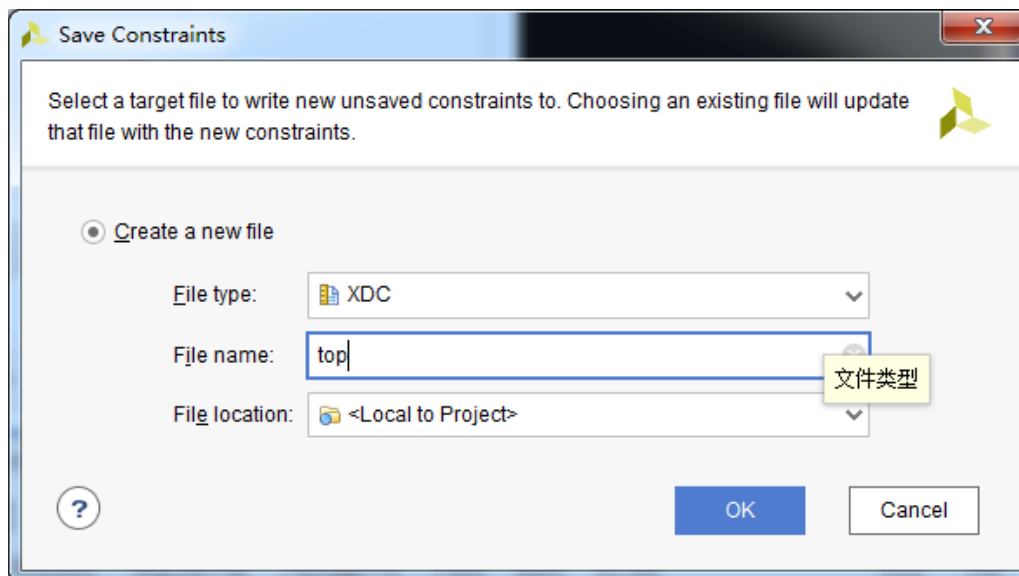
a.8.3、此时应看到如下界面，如果没出现如下界面，在图示位置的 layout 中选择 IO Ports 一项。



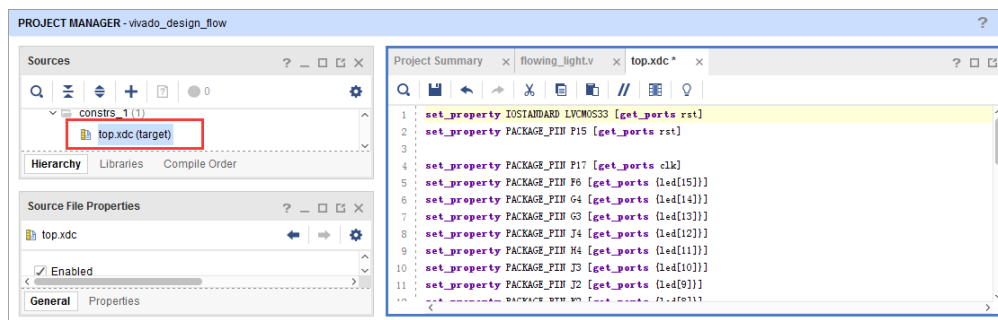
a.8.4、在右下方的选项卡中切换到 I/O ports 一栏，并在对应的信号后，输入对应的 FPGA 管脚标号（或将信号拖拽到右上方 Package 图中对应的管脚上），并指定 I/O std。具体的 FPGA 约束管脚和 IO 电平标准，可参考对应板卡的用户手册或原理图。



a.8.5、完成之后，点击左上方工具栏中的保存按钮，工程提示新建 XDC 文件或选择工程中已有的 XDC 文件。在这里，我们要 Create a new file，输入 File name，点击 OK 完成约束过程。

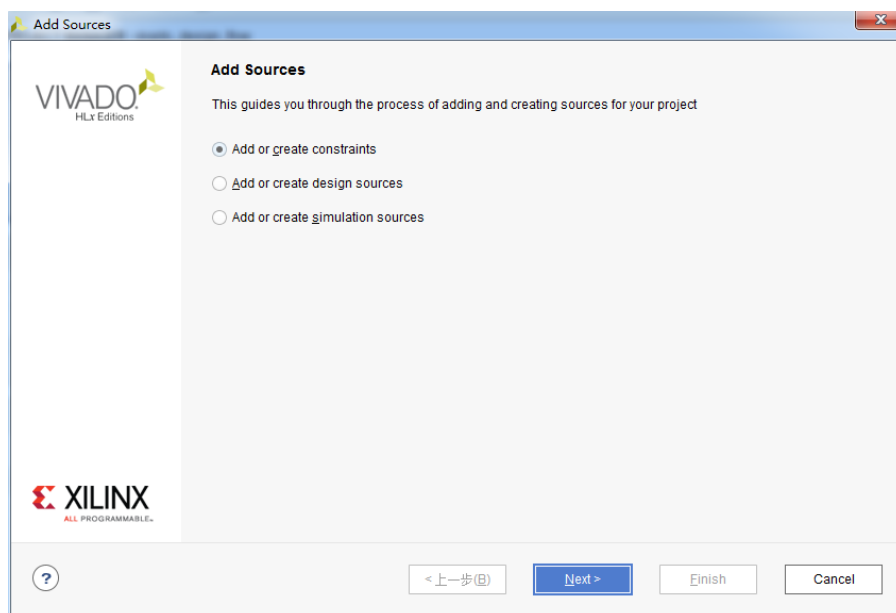


a.8.6、此时，在 Sources 下 Constraints 中会找到新建的 XDC 文件。

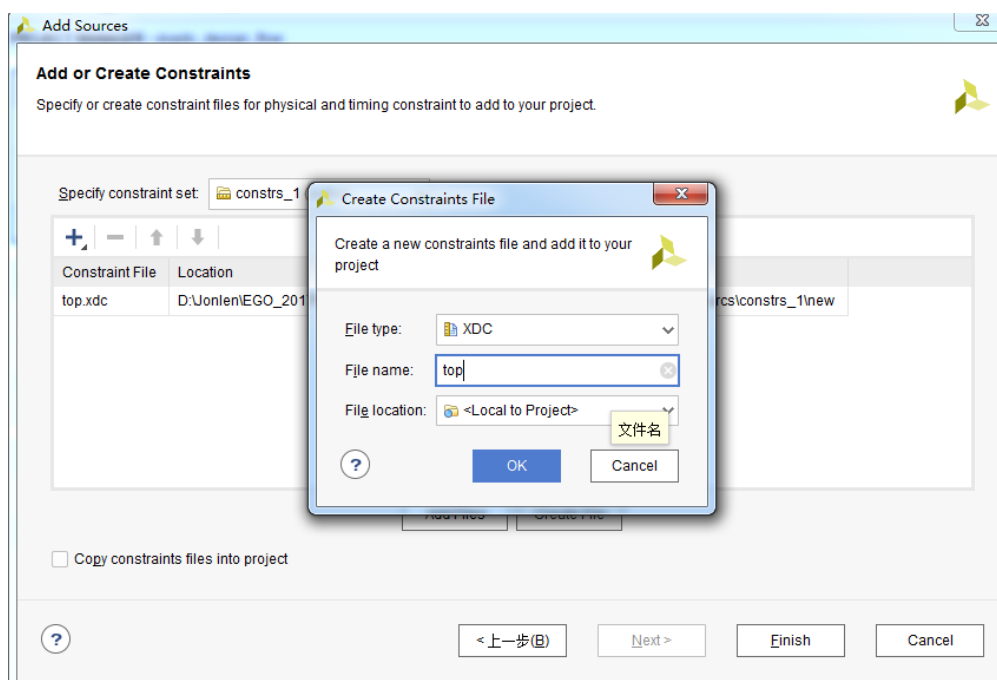
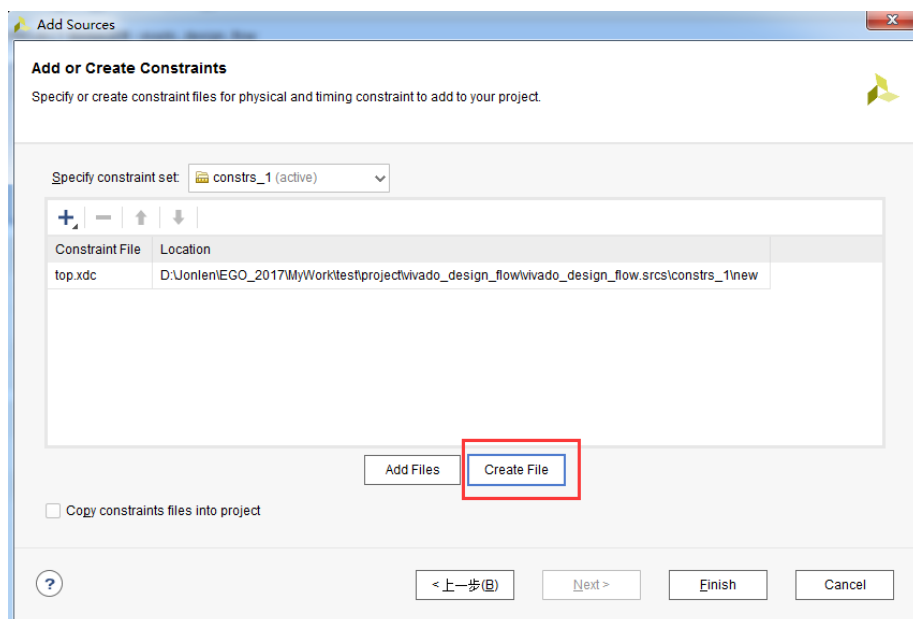


b、如何利用第二种方法添加约束文件。

b.8.1、点击 Add Sources，选择第一项 Add or Create Constraints 一项，点击 Next。

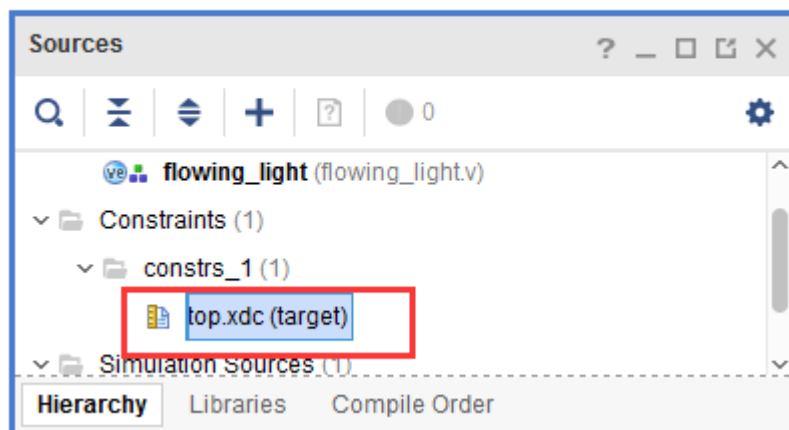


b.8.2、点击 Create File，新建一个 XDC 文件，输入 XDC 文件名，点击 OK。点击 Finish。



b.8.3、双击打开新建好的 XDC 文件，并按照如下规则，输入相应的 FPGA 管脚约束信息和电平标准。

# PROJECT MANAGER - vivado\_design\_flow



```

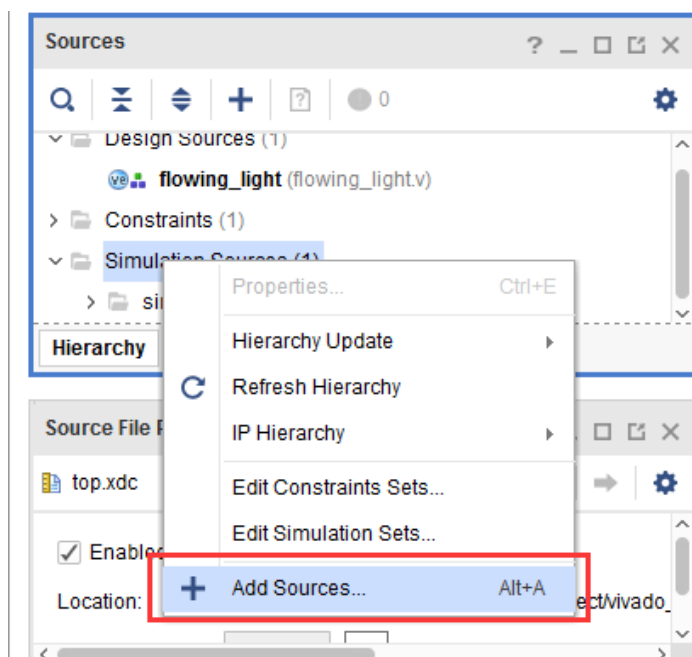
set_property PACKAGE_PIN K3 [get_ports {led[0]]}
set_property PACKAGE_PIN M1 [get_ports {led[1]]}
set_property PACKAGE_PIN L1 [get_ports {led[2]]}
set_property PACKAGE_PIN K6 [get_ports {led[3]]}
set_property PACKAGE_PIN J5 [get_ports {led[4]]}
set_property PACKAGE_PIN H5 [get_ports {led[5]]}
set_property PACKAGE_PIN H6 [get_ports {led[6]]}
set_property PACKAGE_PIN K1 [get_ports {led[7]]}
set_property PACKAGE_PIN K2 [get_ports {led[8]]}
set_property PACKAGE_PIN J2 [get_ports {led[9]]}
set_property PACKAGE_PIN J3 [get_ports {led[10]]}
set_property PACKAGE_PIN H4 [get_ports {led[11]]}
set_property PACKAGE_PIN J4 [get_ports {led[12]]}
set_property PACKAGE_PIN G3 [get_ports {led[13]]}
set_property PACKAGE_PIN G4 [get_ports {led[14]]}
set_property PACKAGE_PIN F6 [get_ports {led[15]]}
set_property PACKAGE_PIN R15 [get_ports rst]
set_property PACKAGE_PIN P17 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports {led[15]]}
set_property IOSTANDARD LVCMOS33 [get_ports {led[14]]}
set_property IOSTANDARD LVCMOS33 [get_ports {led[13]]}
set_property IOSTANDARD LVCMOS33 [get_ports {led[12]]}
set_property IOSTANDARD LVCMOS33 [get_ports {led[11]]}
set_property IOSTANDARD LVCMOS33 [get_ports {led[10]]}
set_property IOSTANDARD LVCMOS33 [get_ports {led[9]]}
set_property IOSTANDARD LVCMOS33 [get_ports {led[8]]}
set_property IOSTANDARD LVCMOS33 [get_ports {led[7]]}
set_property IOSTANDARD LVCMOS33 [get_ports {led[6]]}
set_property IOSTANDARD LVCMOS33 [get_ports {led[5]]}
set_property IOSTANDARD LVCMOS33 [get_ports {led[4]]}
set_property IOSTANDARD LVCMOS33 [get_ports {led[3]]}

```

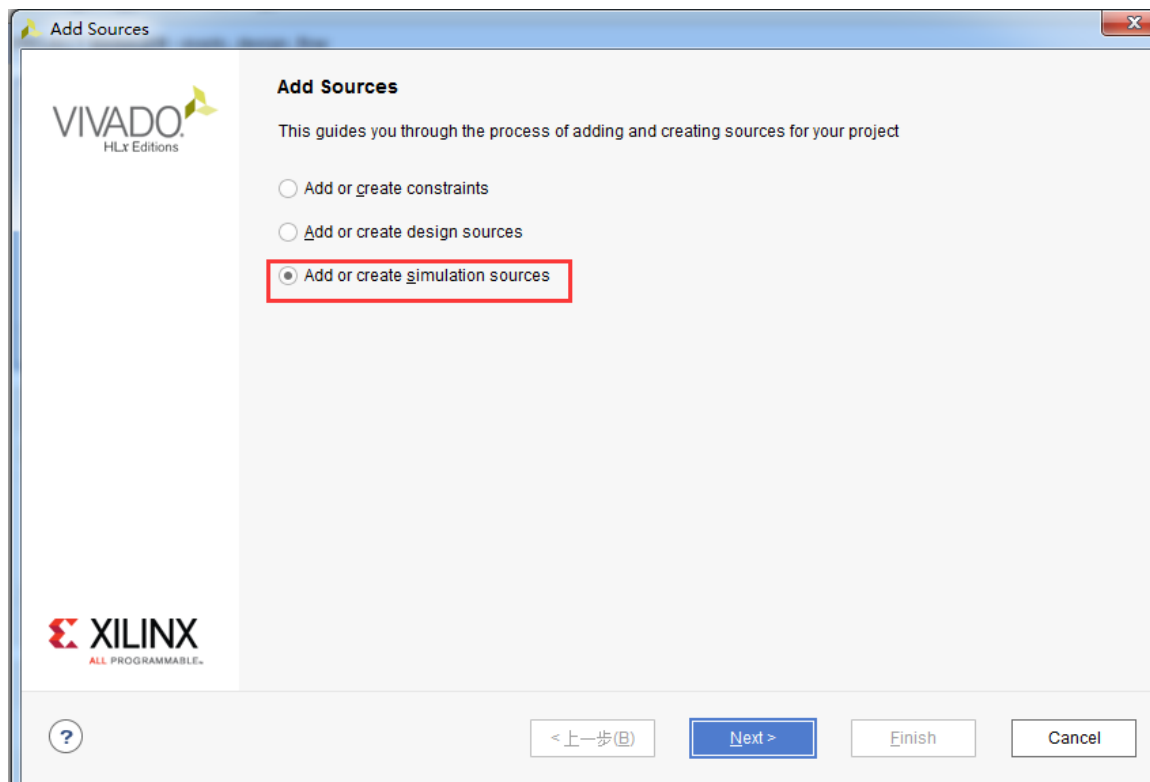
```
set_property IOSTANDARD LVCMOS33 [get_ports {led[2]]}
set_property IOSTANDARD LVCMOS33 [get_ports {led[1]]}
set_property IOSTANDARD LVCMOS33 [get_ports {led[0]]}
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports rst]
```

### 三、利用 Vivado 进行功能仿真

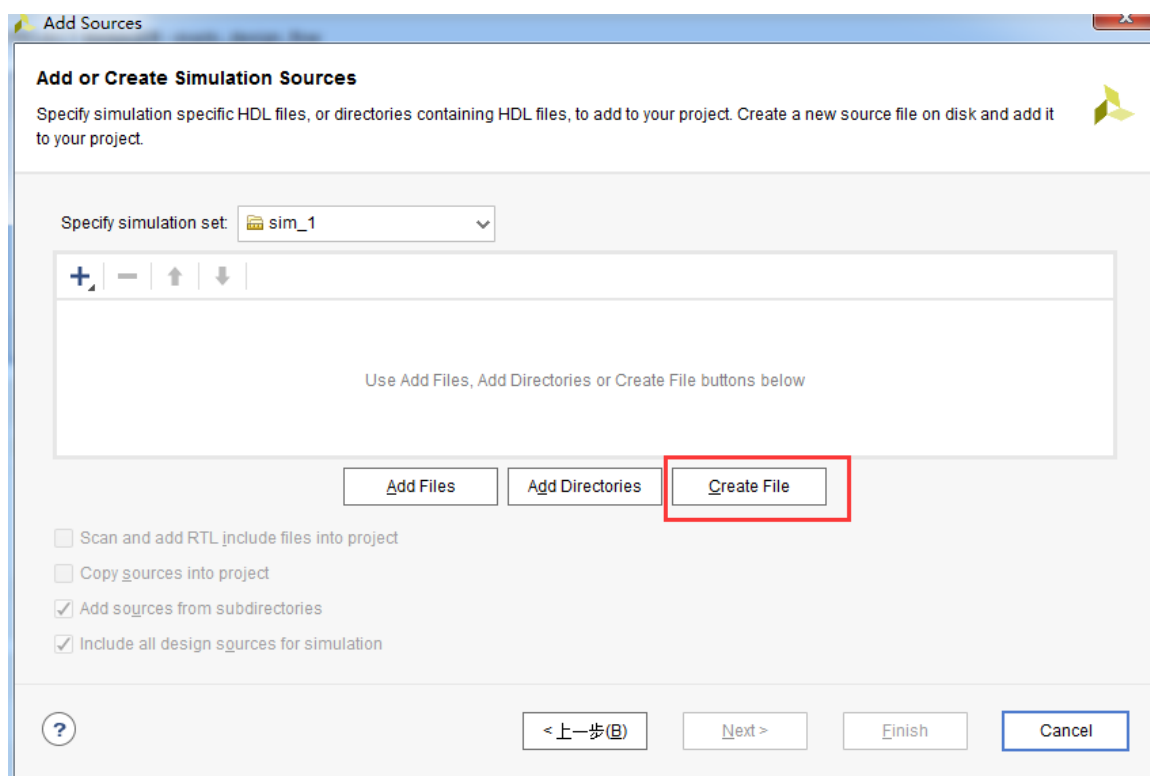
#### 1. 创建激励测试文件，在 Source 中右击选择 Add Source



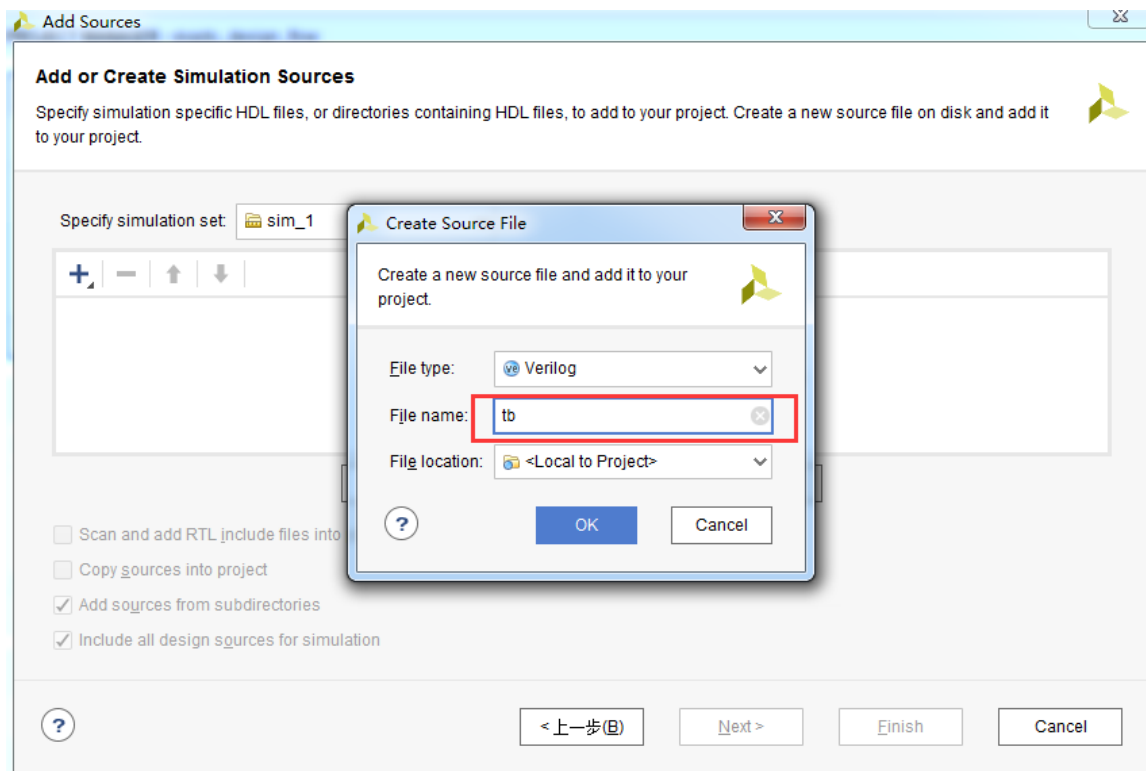
#### 2、在 Add Source 界面中选择第三项 Add or Create Simulation Source, 点击 Next。



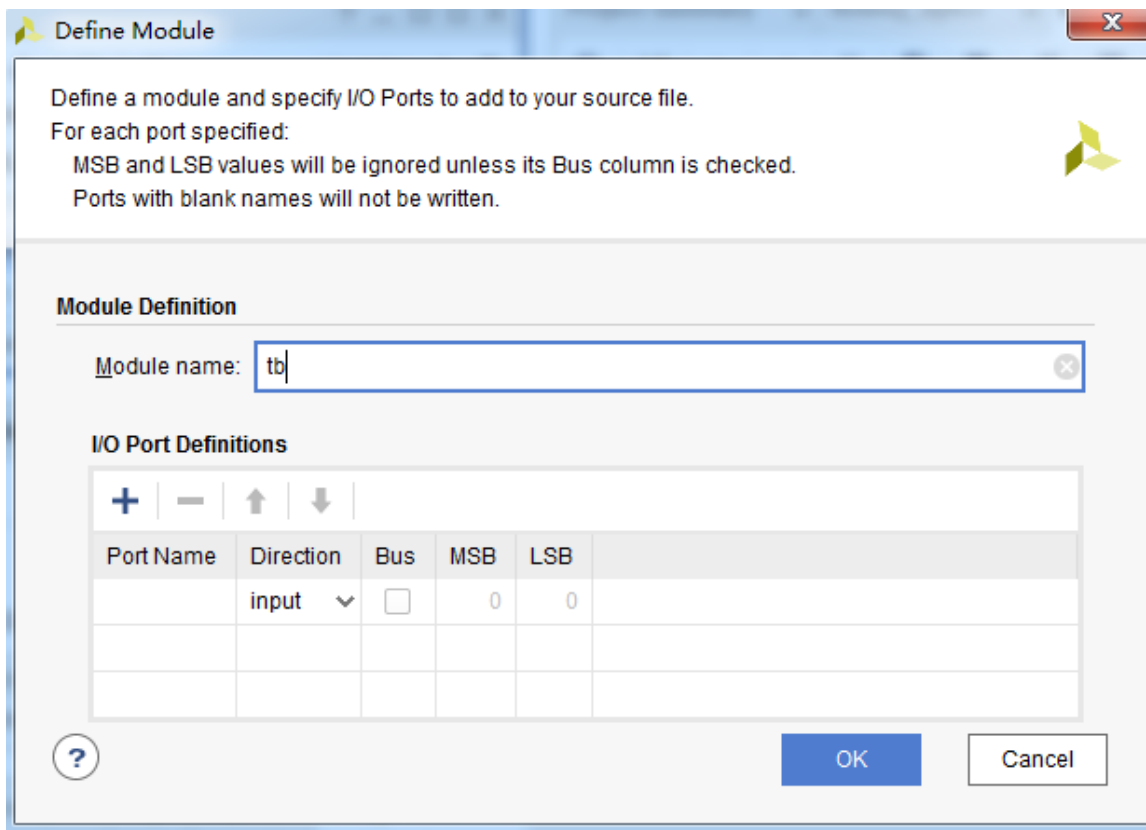
### 3、选择 Create File 创建一个仿真激励文件



### 4、输入激励文件名称，点击 OK。



5、确认添加完成之后点击 Finish，因为是激励文件不需要对外端口，所以直接 Port 部分直接空着，点击 ok。





5、在 Source 下双击打开空白的激励测试文件，完成对将要仿真的 module 的实例化和激励代码的编写，如下图和下述代码所示。

```
`timescale 1ns / 1ps
module test( );

reg clk;

reg rst;
wire [3 : 0] led;

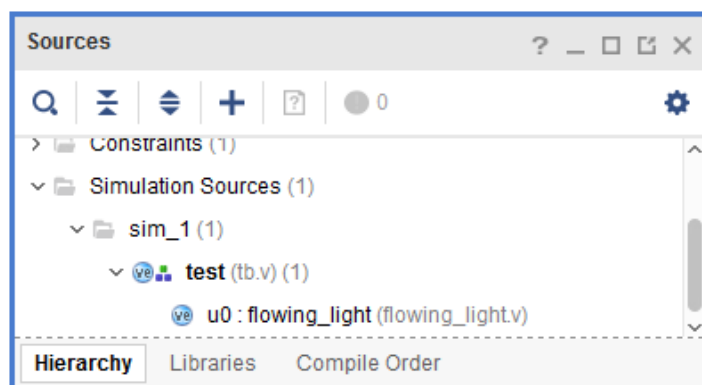
flowing_light u0(
.clk(clk),
.rst(rst),
.led(led) );

parameter PERIOD = 10;

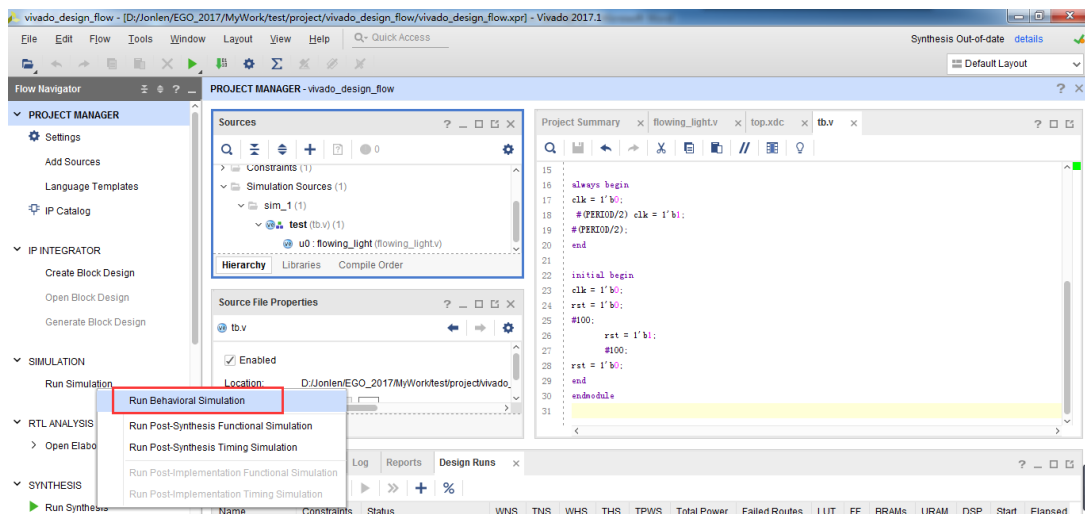
always begin
clk = 1'b0;
#(PERIOD/2) clk = 1'b1;
#(PERIOD/2);
end

initial begin
clk = 1'b0;
rst = 1'b0;
#100;
rst = 1'b1;
#100;
rst = 1'b0; #100; rst = 1'b1;
end
endmodule
```

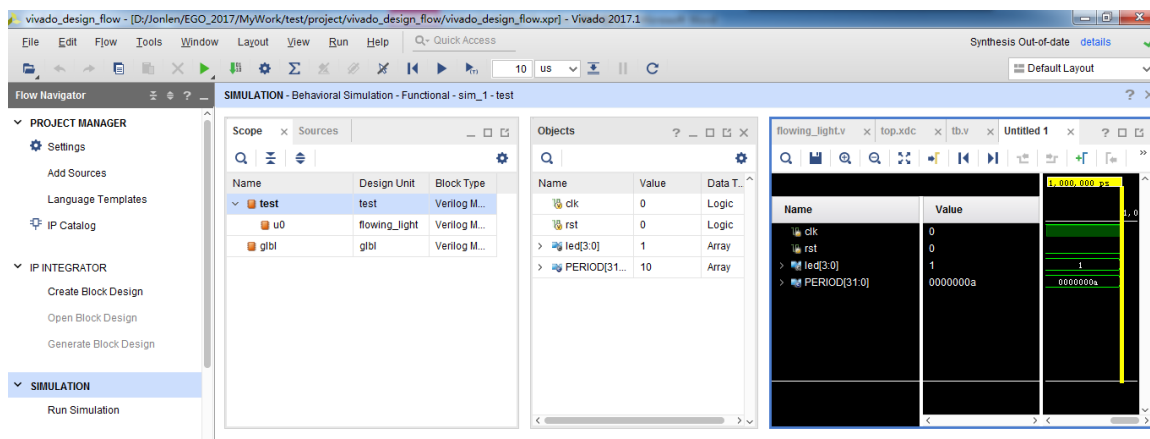
激励文件完成之后，工程目录如下图所示。



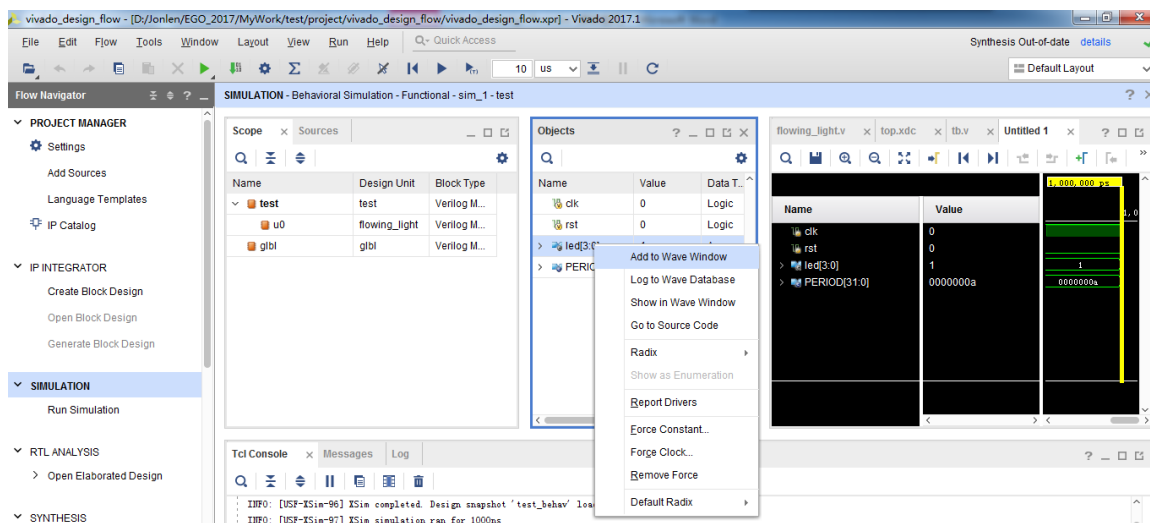
6、此时，进入仿真。在左侧 Flow Navigator 中点击 Simulation 下的 Run Simulation 选项，并选择 Run Behavioral Simulation 一项，进入仿真界面。



7、下图所示为仿真界面。

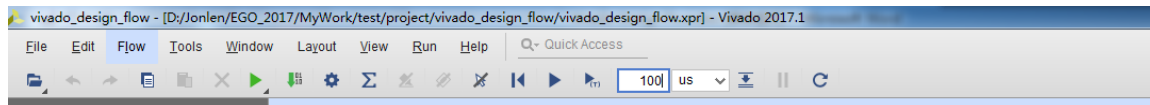


可通过左侧 Scope 一栏中的目录结构定位到设计者想要查看的 module 内部寄存器，在 Objects 对应的信号名称上右击选择 Add To Wave Window，将信号加入波形图中。因为窗口已有信号，此操作不需要进行。

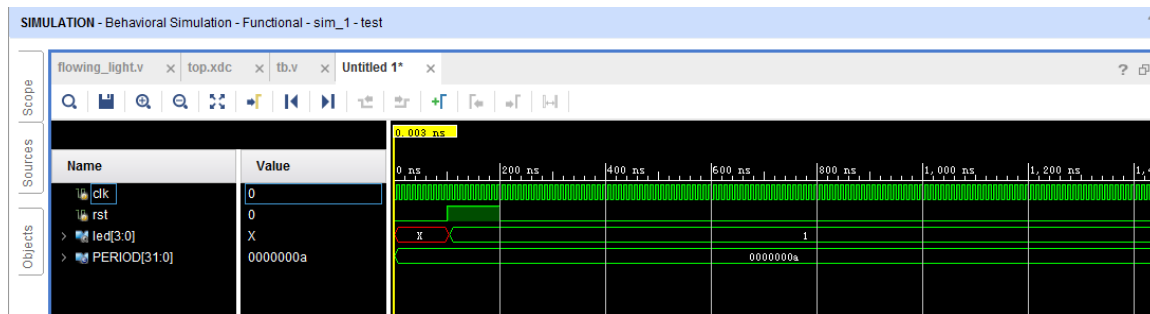


可通过选择工具栏中的如下选项来进行波形的仿真时间控制。如下工具条，分别是复位波形（即

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.....

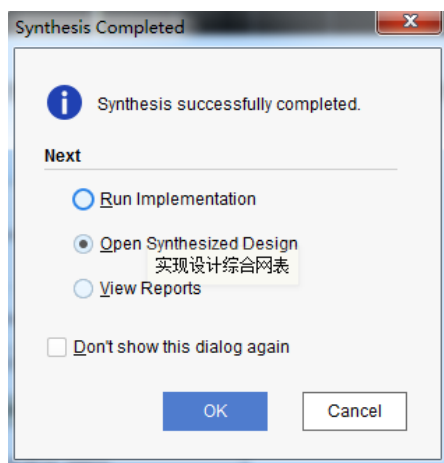


8、最终得到的仿真效果图如下。核对波形与预设的逻辑功能是否一致。仿真完成。

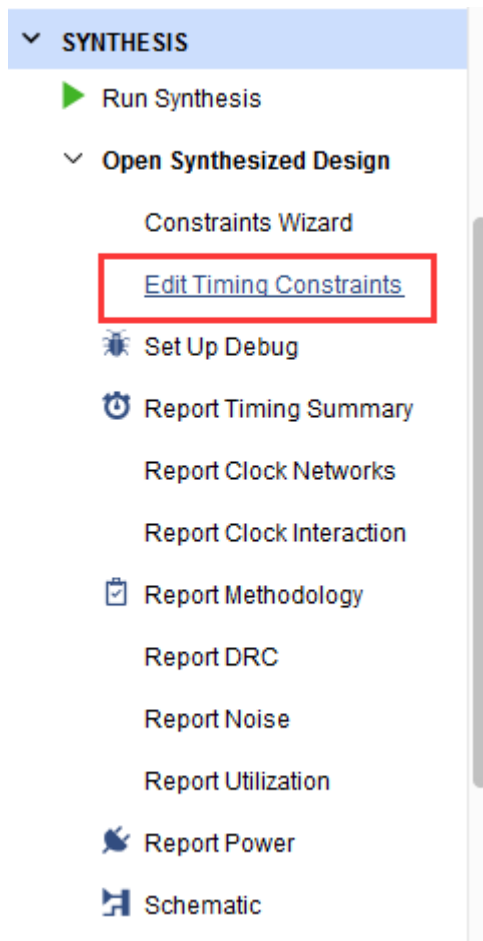


#### 四、添加时序约束

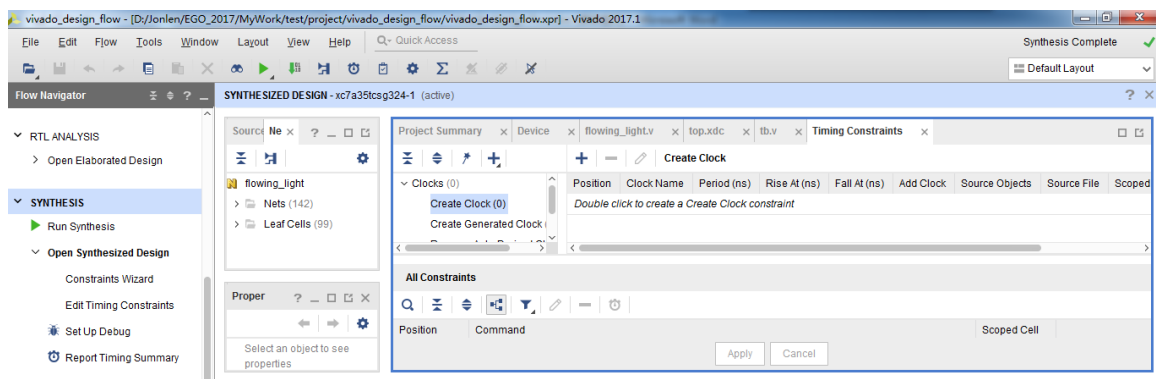
1、在综合完成后打开选择 Open Synthesis Design,或者从 Flow Navigator 中选择 Open Synthesis Design



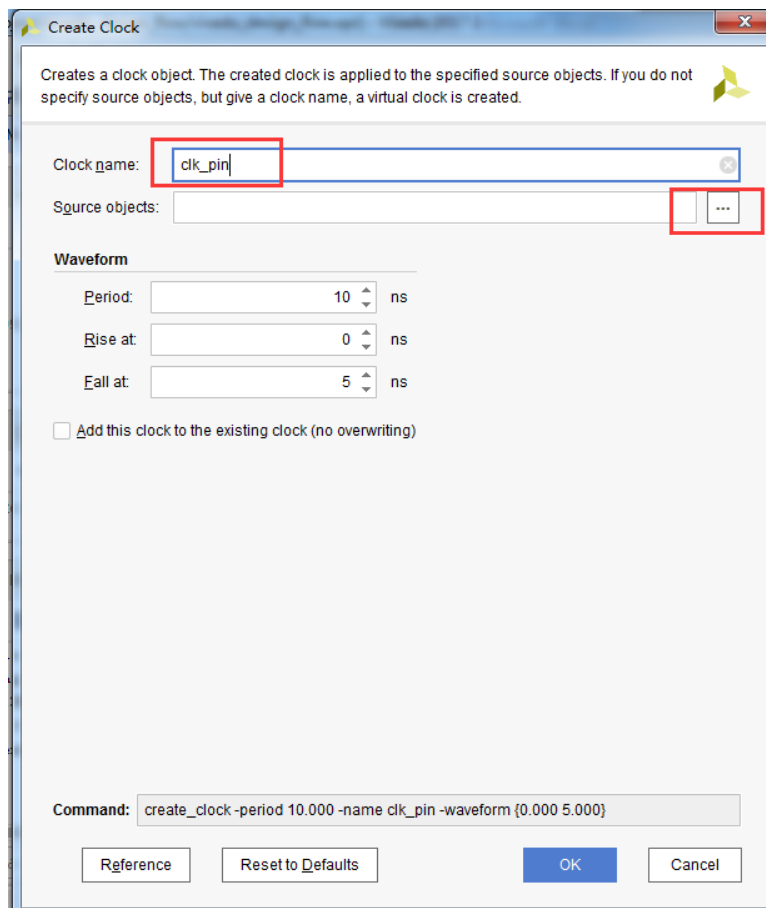
2、选择在 Flow Navigator 中选择 Synthesis > Synthesized Design > Edit Timing Constraints。



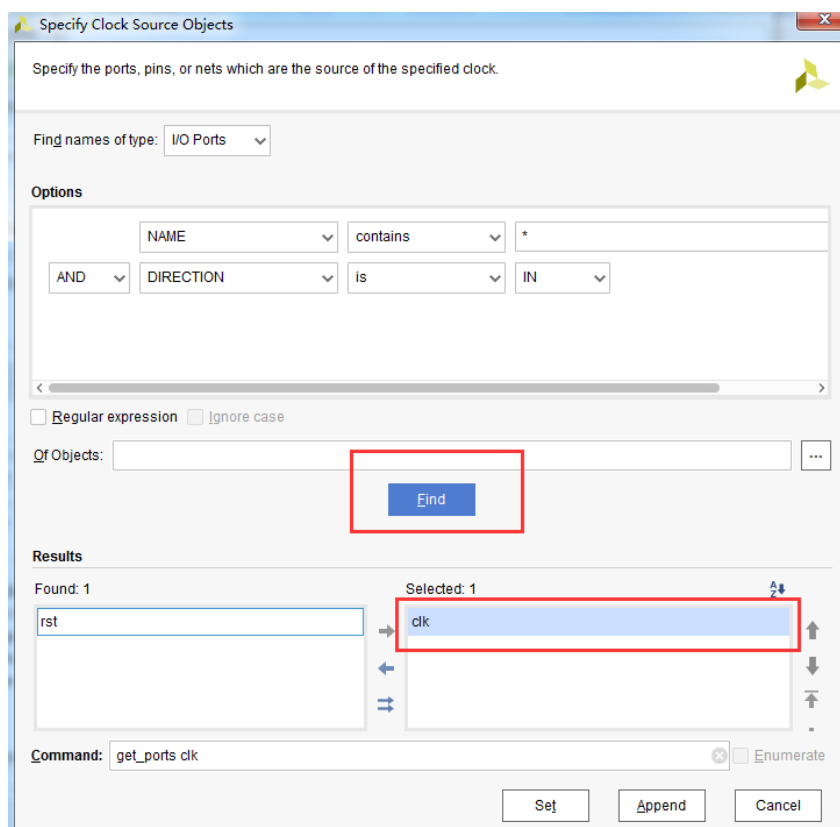
3、打开时序约束界面，开始进行时序约束。



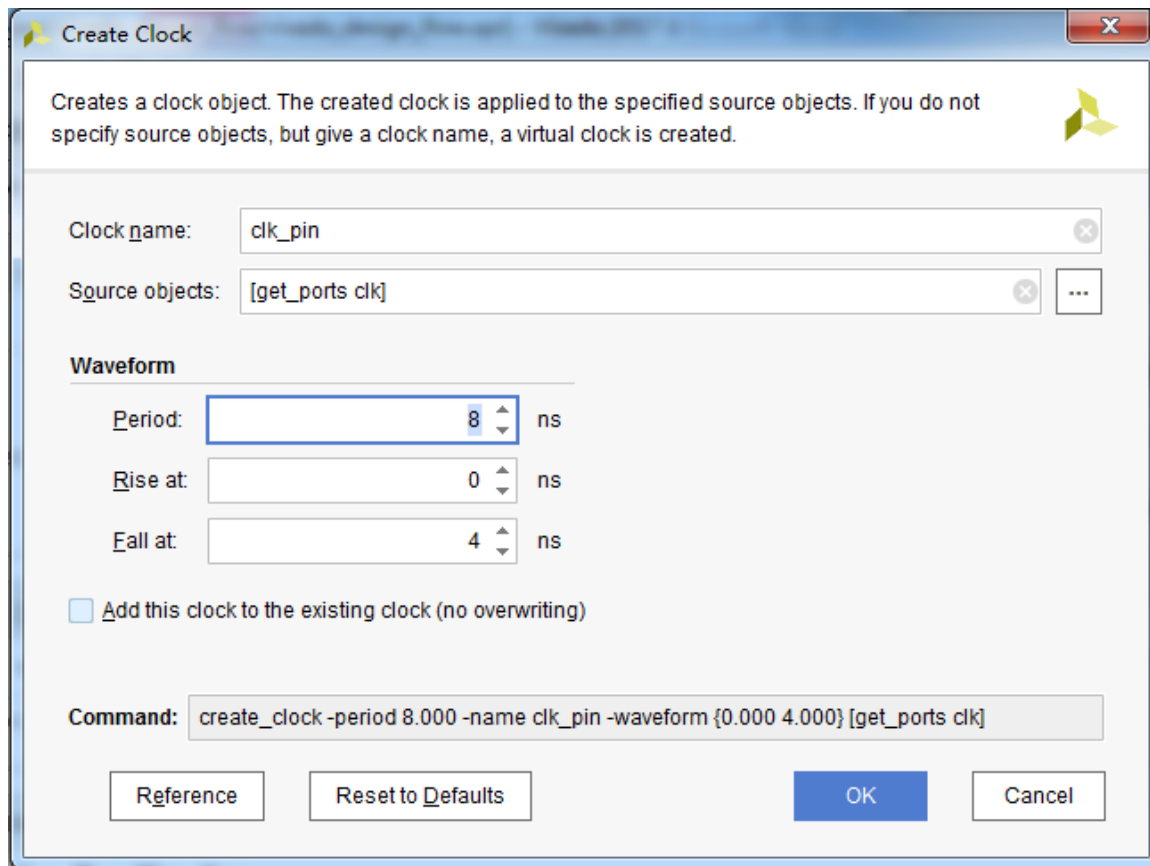
4、双击左边 Clock->Create Clock,进入 Create Clock 界面，在 Clock name 中输入 clk\_pin。在 Source objects 中选择右边的按钮。



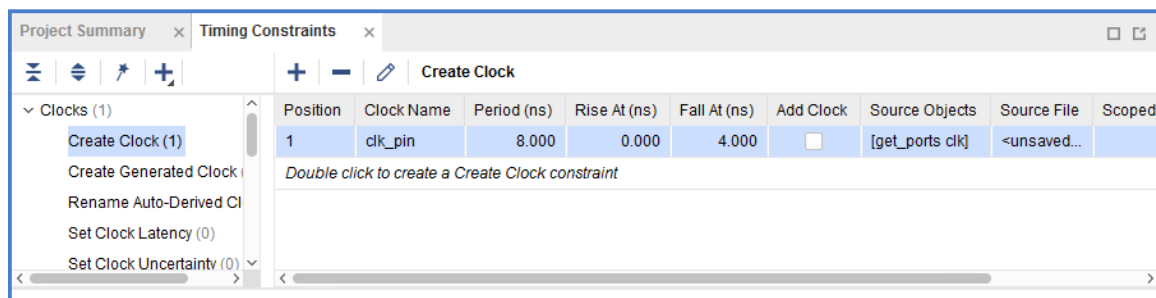
5、在 Specify Clock Source Object 中 Find names of type 选择 I/O Ports 后点击 Find，并将查找到的 clk 选中。完成选择后点击 Set。



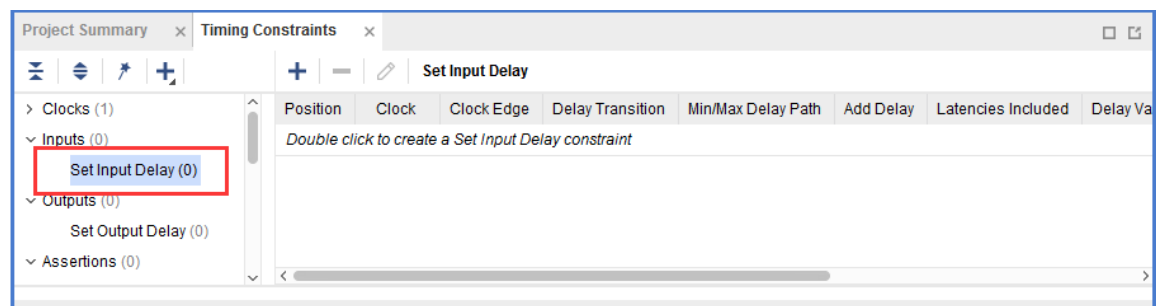
6、将 Period 设置成 8 ns、Rise 设置成 0 ns、fall 设置成 4ns，并点击 ok。



7、如下图所示，这时 Clock 已经创建完成。

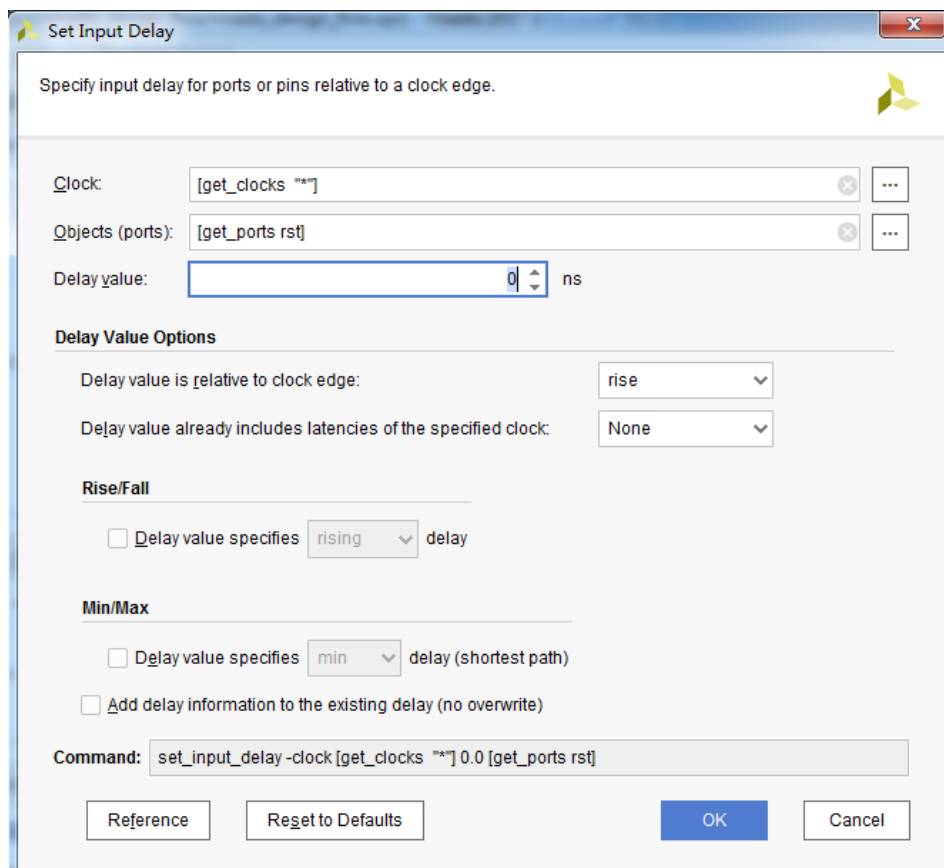


8、接下来将设置 Input Setup Delay，双击左边 Input-> Input Setup Delay。

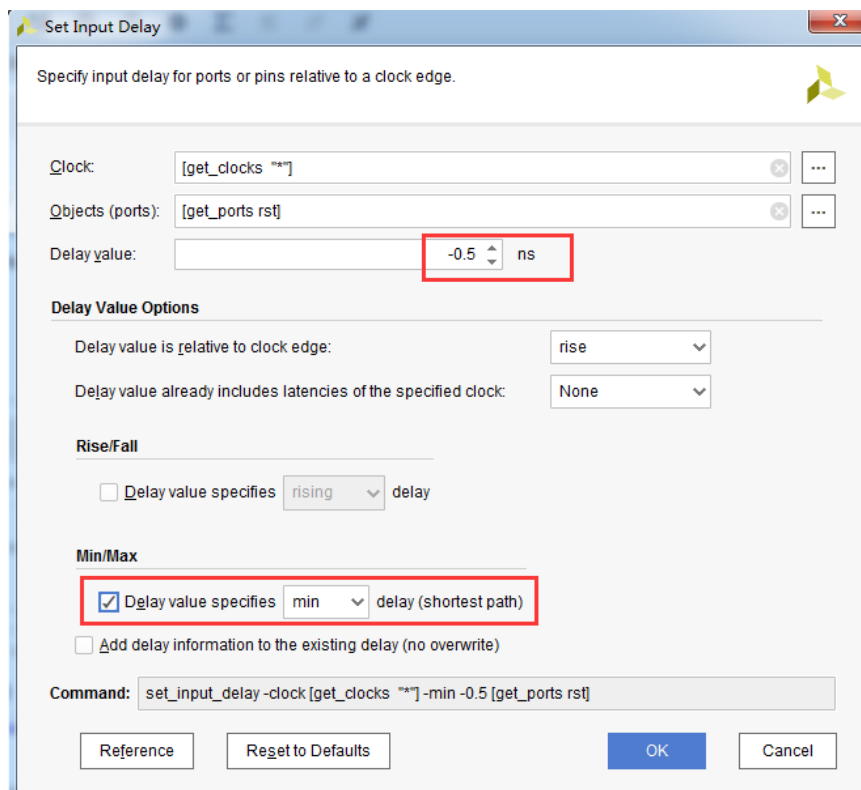


9、进入 Set Input Delay，按照下图配置，Clock 选择 clk\_pin, Objects 选择 rst, Delay 选择 0 ns。完

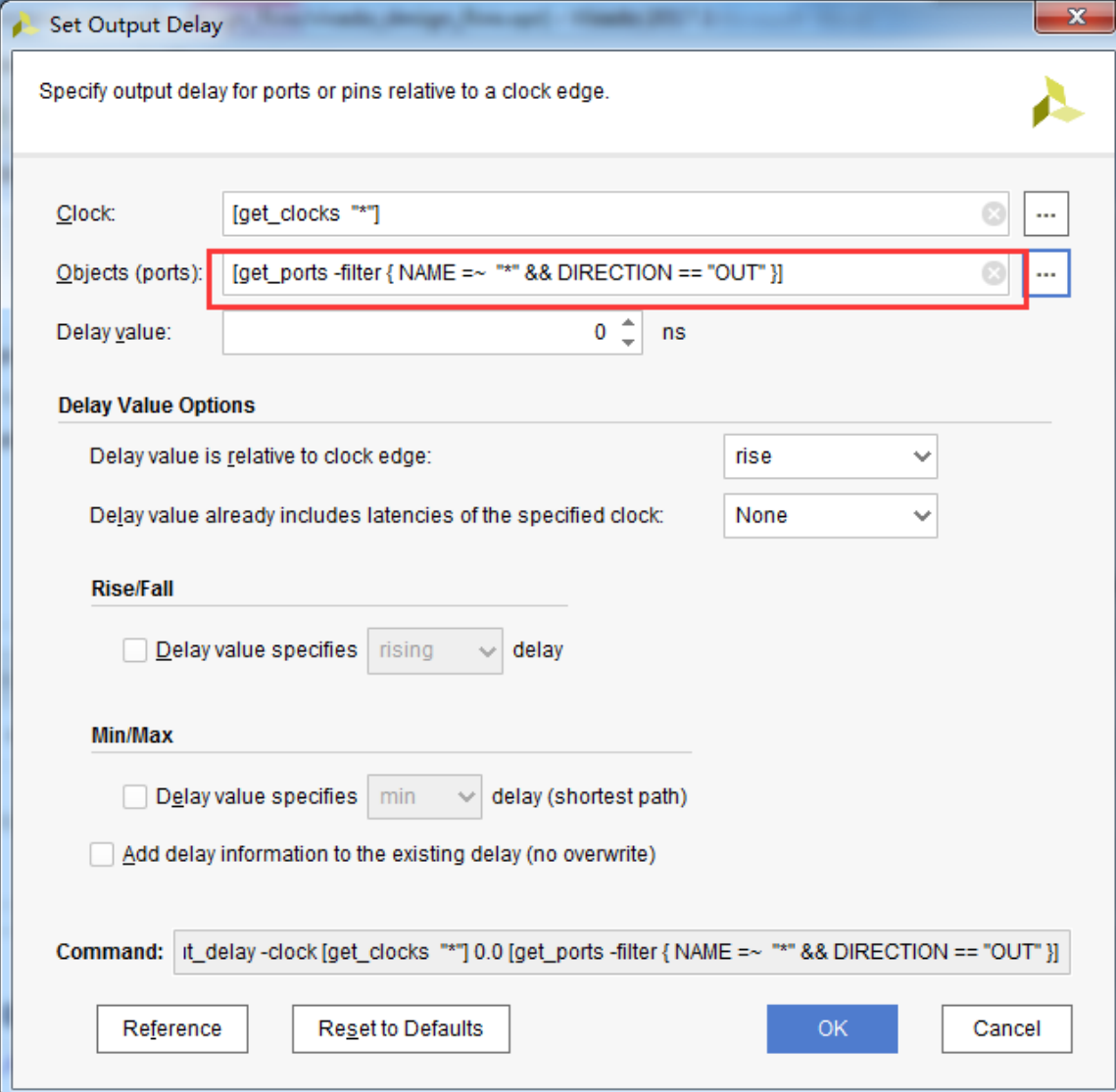
成设置后点击 OK。



10、接下来将设置 Delay value specifies <min/max> delay，双击左边 Input-> Input Setup Delay。将 Clock 选择为 clk\_pin、Objects 选择 rst、Delays value 选择-0.5 ns、并选中 Delay value specifies, Delay 设置成 min。完成设置后点击 ok。



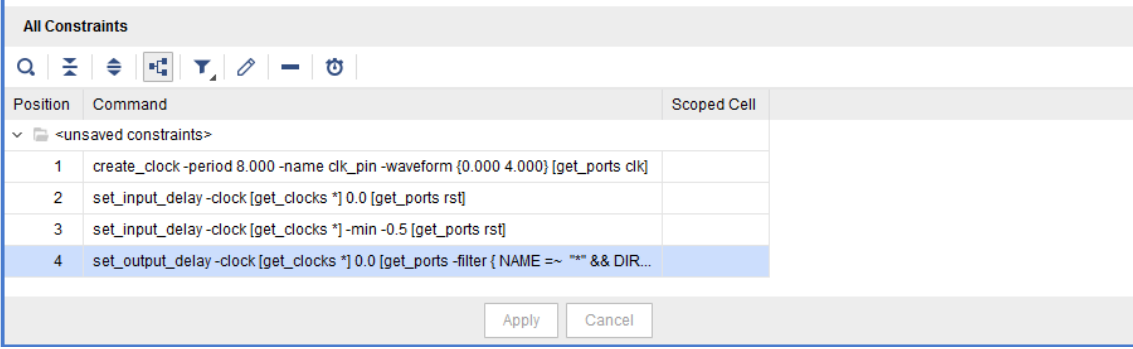
11、接下来设置 Output Delay，双击左边 Output->Set Output Delay。Clock 选择 clk\_pin、Objects 选择所有输出，Delay value 设置为 0 ns。



The image shows the 'Set Output Delay' dialog box in a software interface. The dialog has a title bar with a close button. The main area contains the following fields and options:

- Clock:** A text box containing '[get\_clocks "\*\*"]' with a search icon and an ellipsis button.
- Objects (ports):** A text box containing '[get\_ports -filter { NAME =~ "\*\*" && DIRECTION == "OUT" }]' with a search icon and an ellipsis button. This field is highlighted with a red rectangle.
- Delay value:** A text box containing '0' and a unit dropdown set to 'ns'.
- Delay Value Options:**
  - Delay value is relative to clock edge:** A dropdown menu set to 'rise'.
  - Delay value already includes latencies of the specified clock:** A dropdown menu set to 'None'.
  - Rise/Fall:**
    - ☐ Delay value specifies **rising** delay
  - Min/Max:**
    - ☐ Delay value specifies **min** delay (shortest path)
    - ☐ Add delay information to the existing delay (no overwrite)
- Command:** A text box containing 'it\_delay -clock [get\_clocks "\*\*"] 0.0 [get\_ports -filter { NAME =~ "\*\*" && DIR...'.
- Buttons:** 'Reference', 'Reset to Defaults', 'OK', and 'Cancel'.

12、完成以上约束后可以在 All Constraints 看到如下约束结果。选择 File->Save Constraints 将设置的约束保存。



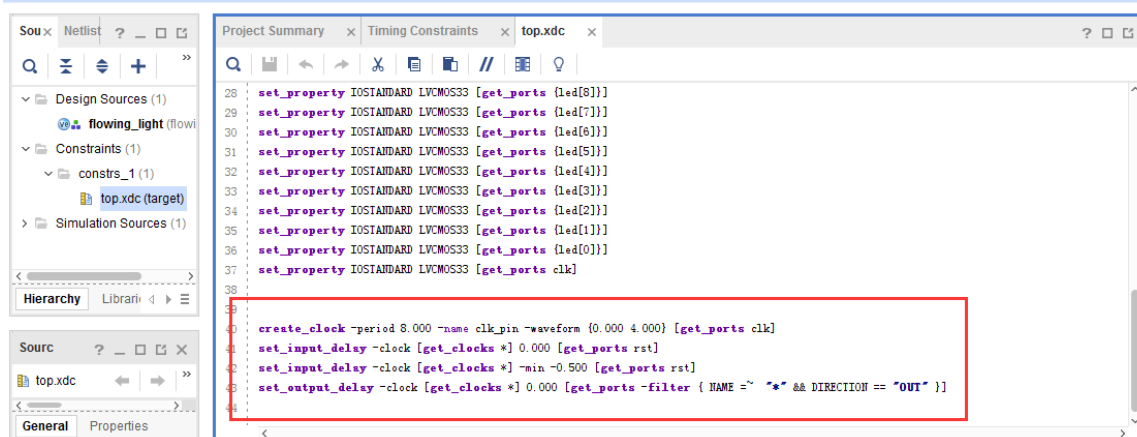
The image shows the 'All Constraints' window, which displays a list of constraints. The window has a title bar and a toolbar with icons for search, zoom, and other functions. The table below shows the constraints:

Position	Command	Scoped Cell
▼ <unsaved constraints>		
1	create_clock -period 8.000 -name clk_pin -waveform {0.000 4.000} [get_ports clk]	
2	set_input_delay -clock [get_clocks *] 0.0 [get_ports rst]	
3	set_input_delay -clock [get_clocks *] -min -0.5 [get_ports rst]	
4	set_output_delay -clock [get_clocks *] 0.0 [get_ports -filter { NAME =~ "**" && DIR...	

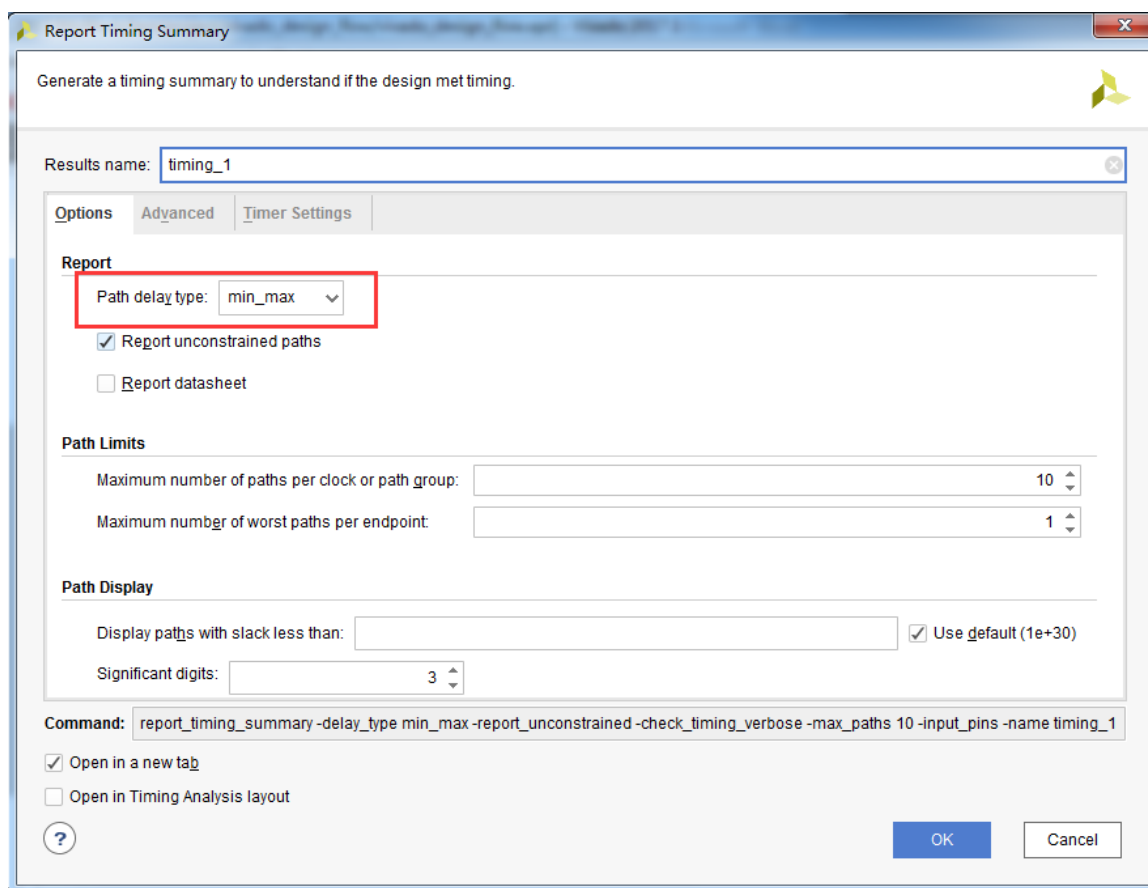
At the bottom of the window, there are 'Apply' and 'Cancel' buttons.



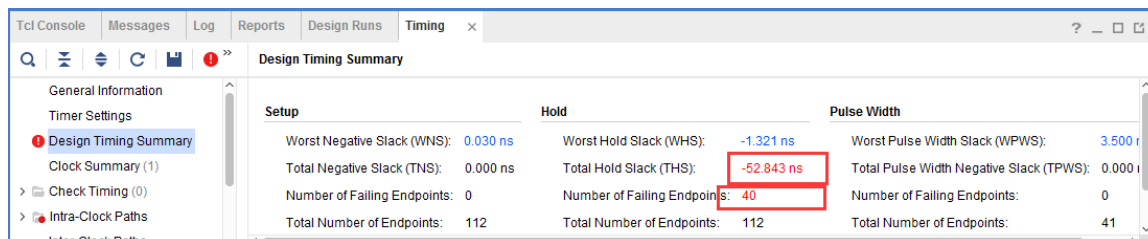
13、这时候大家打开 Sources 界面就可以看到我们的约束已经写入了 XDC 文件去了。



14、在 Flow Navigator 中选择 Synthesized Design -> Report Timing Summary。并将 Options 标签里将 Path delay type 设置成 min\_max。



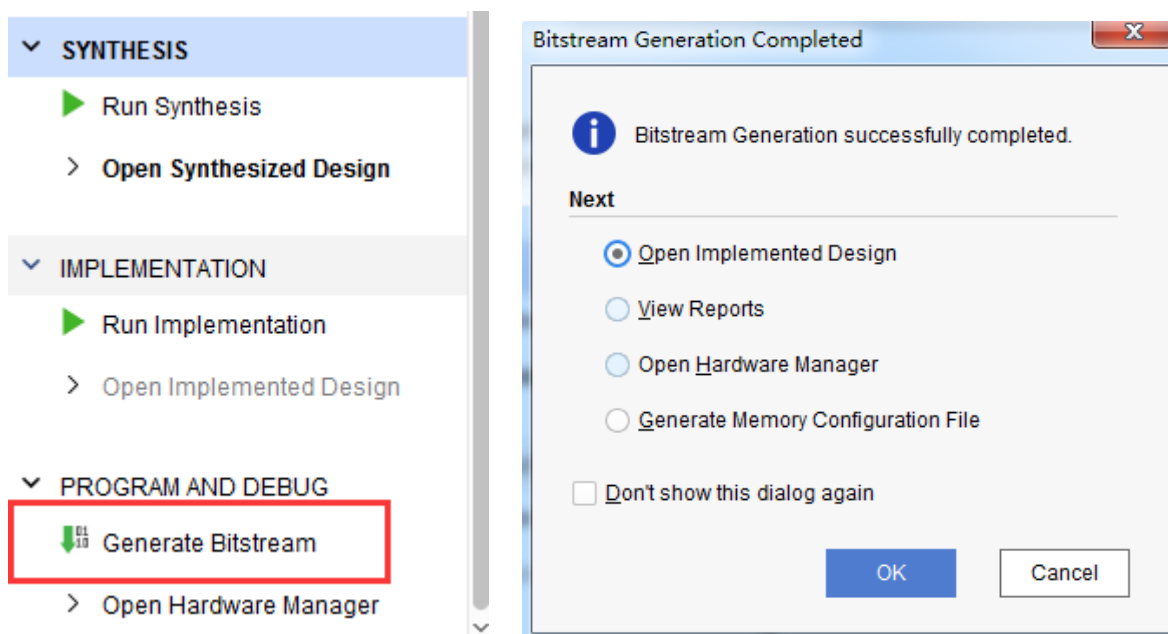
15、在完成时序报告后，大家可以在报告中看到 Setup 和 Hold 的地方显示了红色，即为时序约束后，需求没有满足。然后在进行 Implementation 的时候，Vivado 会自动优化布线路径，来满足用户设定的约束时间。如果在 Implementation 中还是显示无法满足，则需要分析电路进行进一步约束。



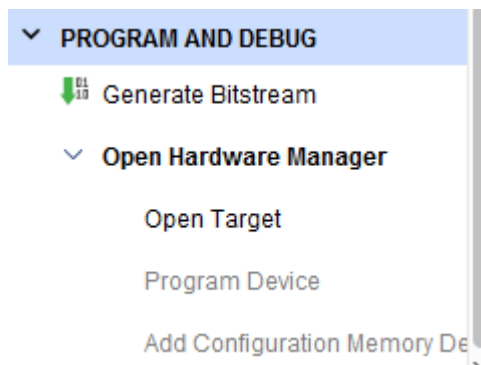
Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 0.030 ns	Worst Hold Slack (WHS): -1.321 ns	Worst Pulse Width Slack (WPWS): 3.500 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): -52.843 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 40	Number of Failing Endpoints: 0
Total Number of Endpoints: 112	Total Number of Endpoints: 112	Total Number of Endpoints: 41

## 五、工程实现

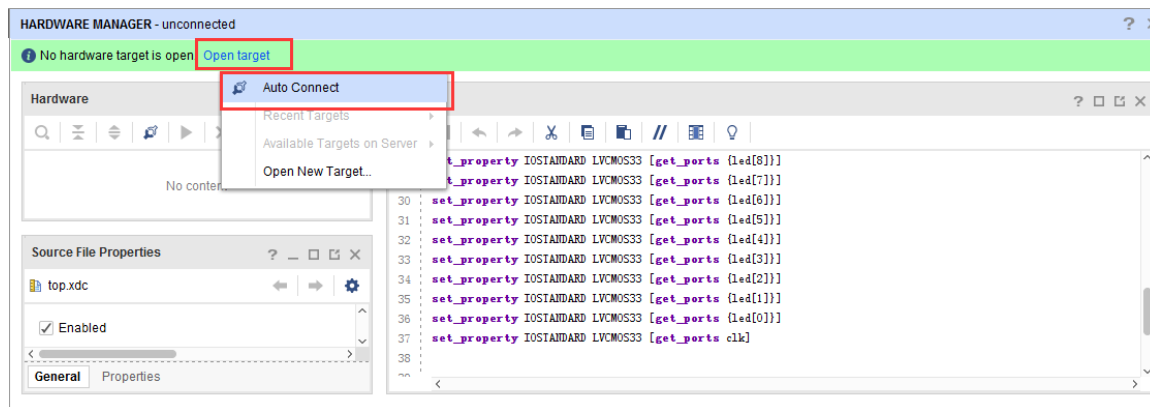
1、在 Flow Navigator 中点击 Program and Debug 下的 Generate Bitstream 选项，工程会自动完成综合、实现、Bit 文件生成过程，完成之后，可点击 Open Implemented Design 来查看工程实现结果。



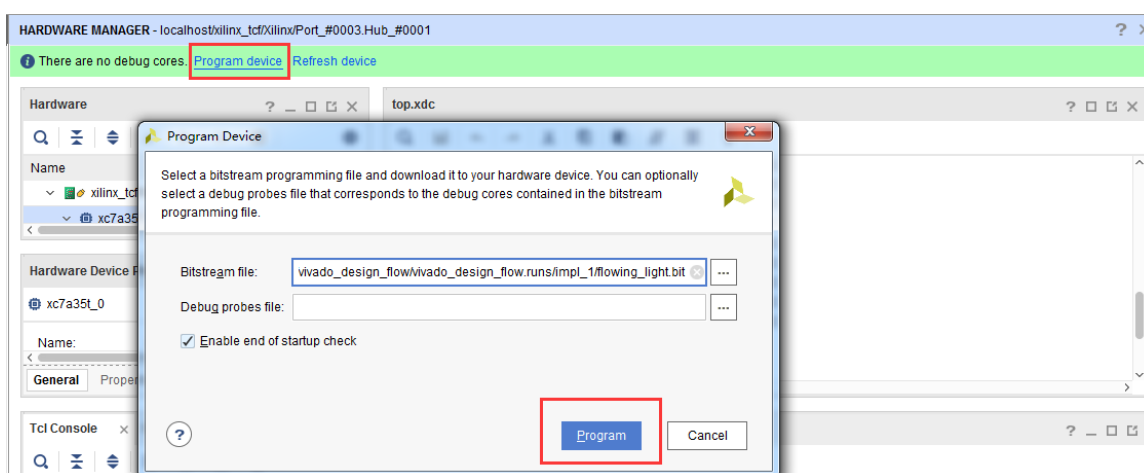
2、点击 Flow Navigator 中 Open Hardware Manager 一项，进入硬件编程管理界面。



3、在提示的信息中，选择 Open Hardware Manager（或在 Flow Navigator 中展开 Hardware Manager，点击 Open Target）。选择 Auto Connect 连接到板卡。



4、连接成功后，在目标芯片上右击，选择“Program Device”。在弹出的对话框中“Bitstream File”一栏已经自动加载本工程生成的比特流文件，点击“Program”对 FPGA 芯片进行编程。



5、下载完成后，在板上观察实验结果。

