

Spring, 2020

1090140071

FPGA设计及应用

Xilinx Vivado软件入门

大连理工大学 电信学部
夏书峰

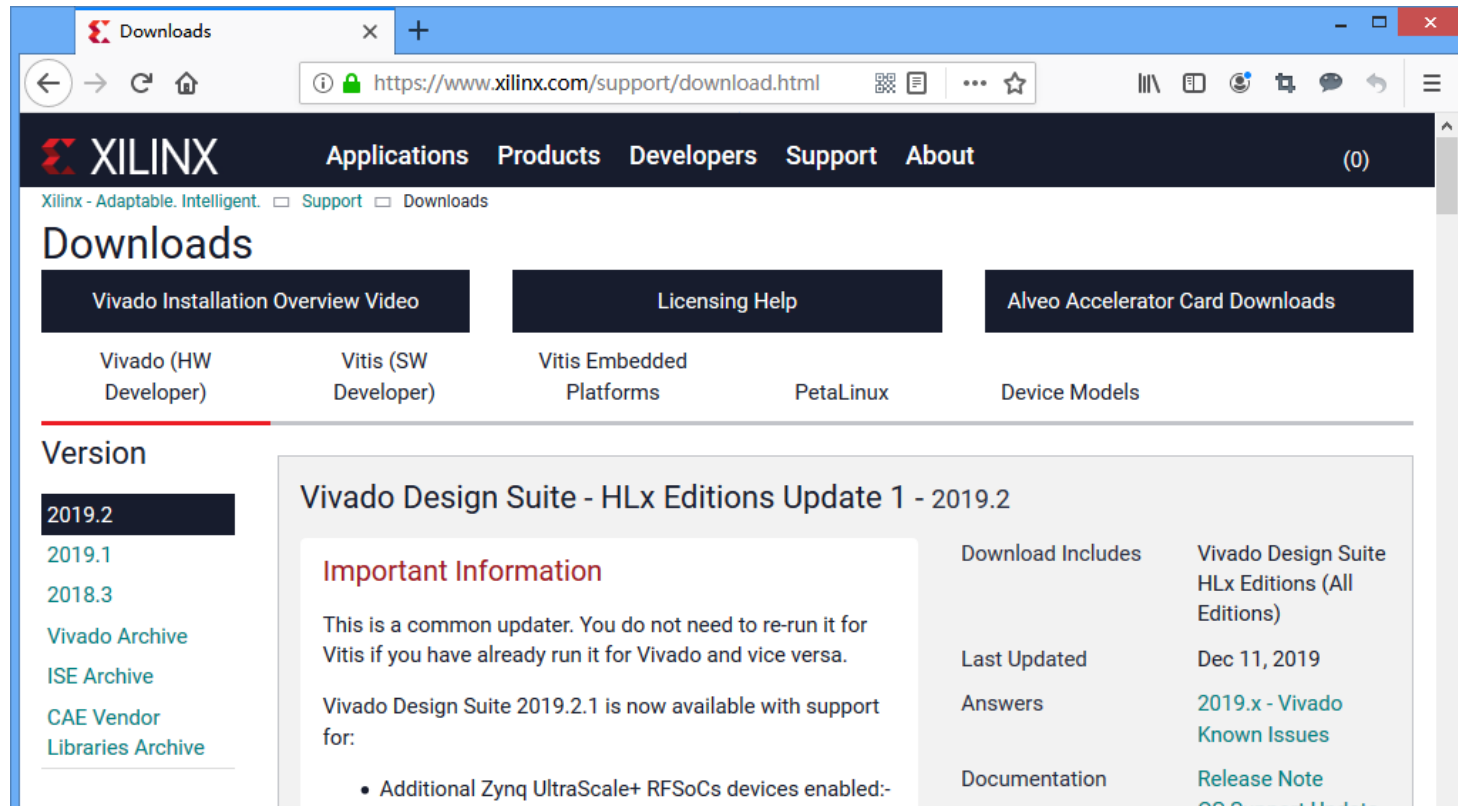
ISE/Vivado不同配置的功能区别

| Features | WebPACK | Logic Edition | Embedded Edition | DSP Edition | System Edition |
|-------------------------------------------------------------------------------|-------------------------|---------------|------------------|-------------|----------------|
| Device Support | Limited | All | All | All | All |
| ChipScope/Pro Serial I/O Toolkit | X | √ | √ | √ | √ |
| CORE Generator | √ | √ | √ | √ | √ |
| Design Preservation | √ | √ | √ | √ | √ |
| Embedded IP Peripherals | 3 smallest Zynq devices | √ | √ | √ | √ |
| ISE Simulator (ISim) | Limited | √ | √ | √ | √ |
| MicroBlaze Soft Processor | 3 smallest Zynq devices | √ | √ | √ | √ |
| Partial Reconfiguration* | Option | Option | Option | Option | Option |
| PlanAhead™ | √ | √ | √ | √ | √ |
| Platform Studio | 3 smallest Zynq devices | √ | √ | √ | √ |
| Power Optimization | √ | √ | √ | √ | √ |
| Software Dev. Kit (SDK) | √ | √ | √ | √ | √ |
| System Generator for DSP | X | X | X | √ | √ |
| Timing Driven Place & Route, SmartGuide, and SmartXplorer | √ | √ | √ | √ | √ |
| XST Synthesis | √ | √ | √ | √ | √ |

更多信息见网上[Software_matrix.pdf](#)

Xilinx软件下载

- 访问Xilinx官网主菜单Support->Download&Licensing
<https://www.xilinx.com/support/download.html>



- Vivado软件用于硬件开发，包括逻辑、DSP、嵌入式处理器，安装包26.5GB。可以只下载Installer，约65MB，之后根据所选安装项目在线下载安装；
- Vitis软件是2019年末的新平台，已包含Vivado，并整合原来嵌入式处理器开发的SDK、SDSoC、SDAccel等工具，完整安装包约31GB。可以只下载Installer，在线安装。

学校网盘里的共享资源



- <http://pan.dlut.edu.cn/share?id=hqh7mzsuhtdy>

下载完用WinMD5校验一下文件是否正确下载（与txt文件里的MD5比对），若学校网盘无法下载，请去Xilinx官网下载原始的tar.gz格式压缩包。

- Xilinx_Vitis_2019.2_1106_2127.iso是基础安装
- Xilinx_Vivado_Vitis_Update_2019.2.1_1205_0436.iso是升级补丁
- 用虚拟光驱软件加载iso文件，运行里面的xsetup.exe安装



VITIS
IDE

2019.2

Copyright 1986-2019, Xilinx, Inc.
All Rights Reserved

安装Vitis软件

- Vivado: 硬件逻辑设计
- Vitis: Vivado + SDK + SDSoc + SDAccel
逻辑设计+嵌入式软件开发等

Vitis 2019.2 Installer - Welcome



Welcome

We are glad you've chosen Xilinx as your development target. This program can install the Vitis Unified Software Platform.

Supported operating systems for Vitis 2019.2 are:

- Windows 7.1: 64-bit
- Windows 10 Professional versions 1809 and 1903: 64-bit
- Red Hat Enterprise Linux 7.4-7.6: 64-bit
- CentOS Linux 7.4-7.6: 64-bit
- SUSE Enterprise Linux 12.4: 64-bit
- Amazon Linux 2 AL2 LTS: 64-bit
- Ubuntu Linux 16.04.5, 16.04.6, 18.04.1 and 18.04.2 LTS: 64-bit - Additional library installation required

Note: This release requires upgrading your license server tools to the Flex 11.14.1 versions. Please confirm with your license admin that the correct version of the license server tools are installed and available, before running the tools.

To reduce installation time, we recommend that you disable any anti-virus software before continuing.

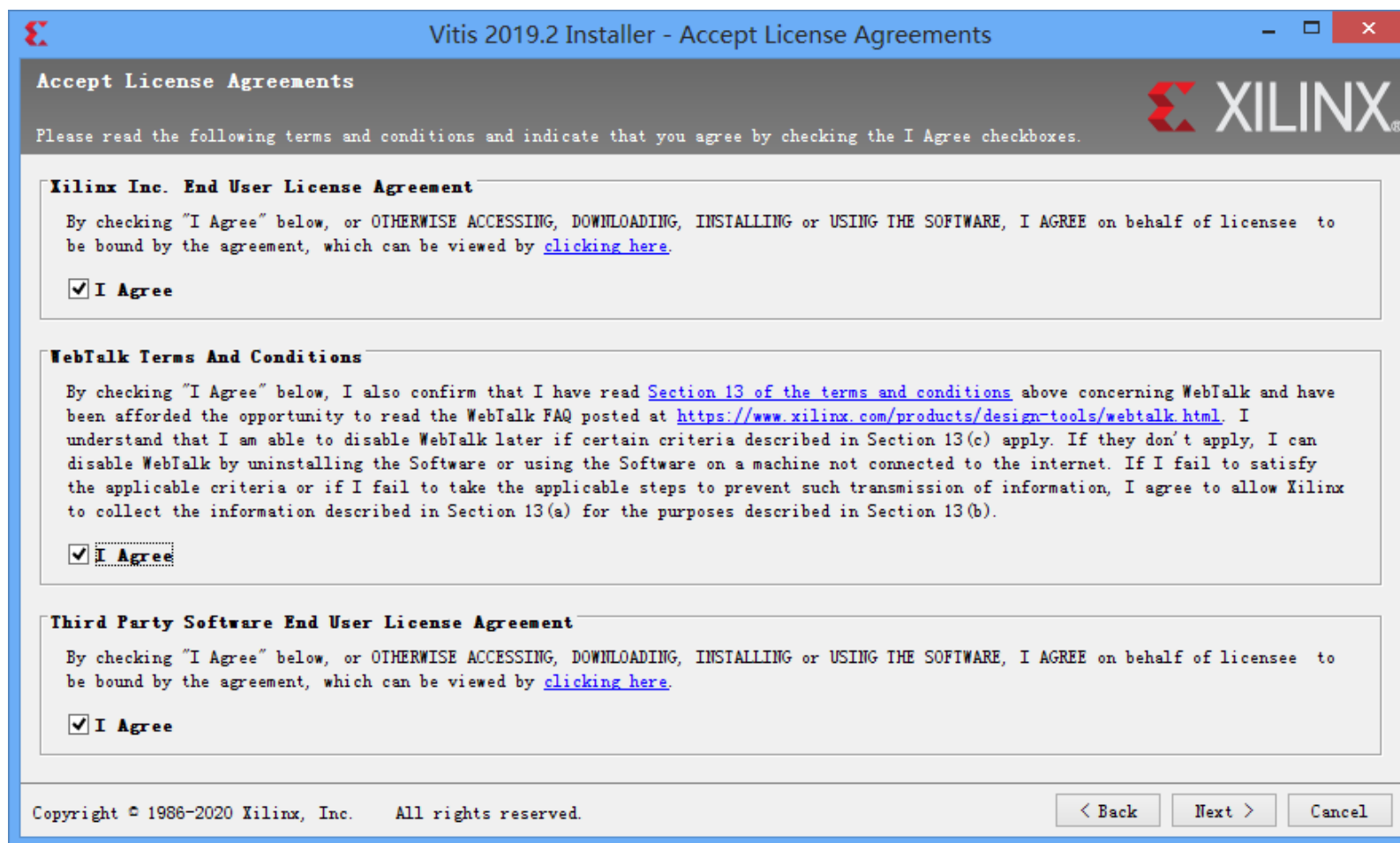
Copyright © 1986-2020 Xilinx, Inc. All rights reserved.

Preferences

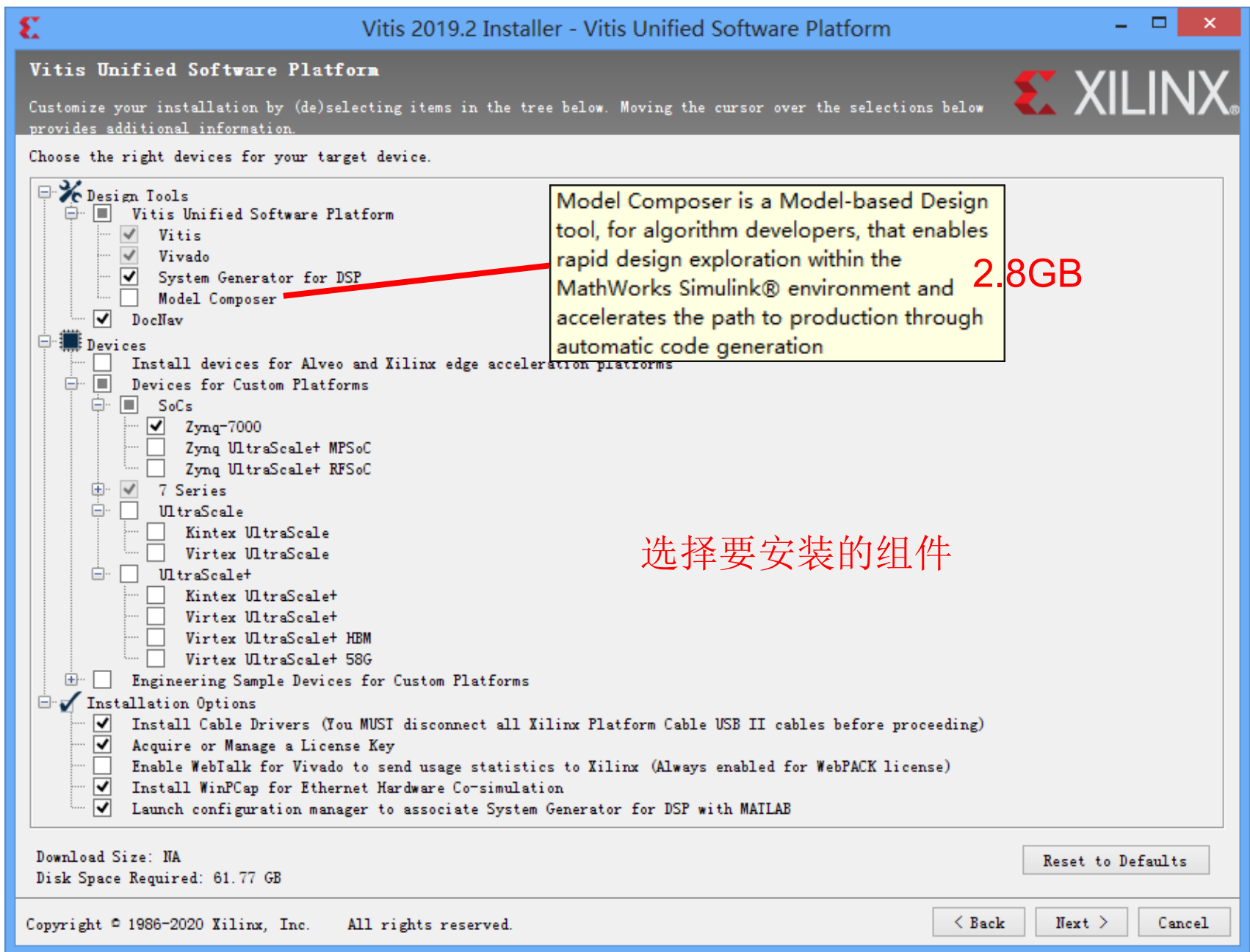
< Back

Next >

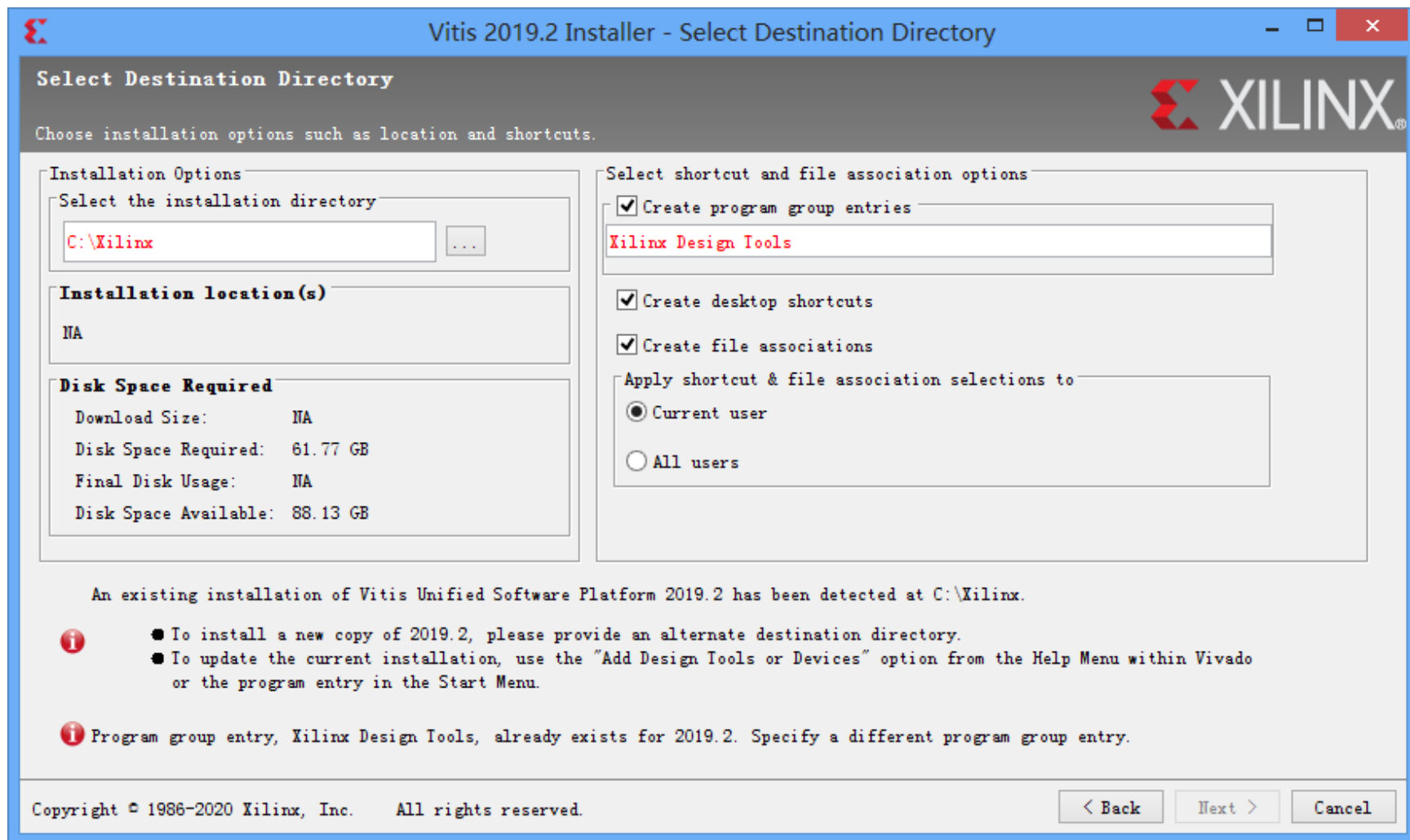
Cancel



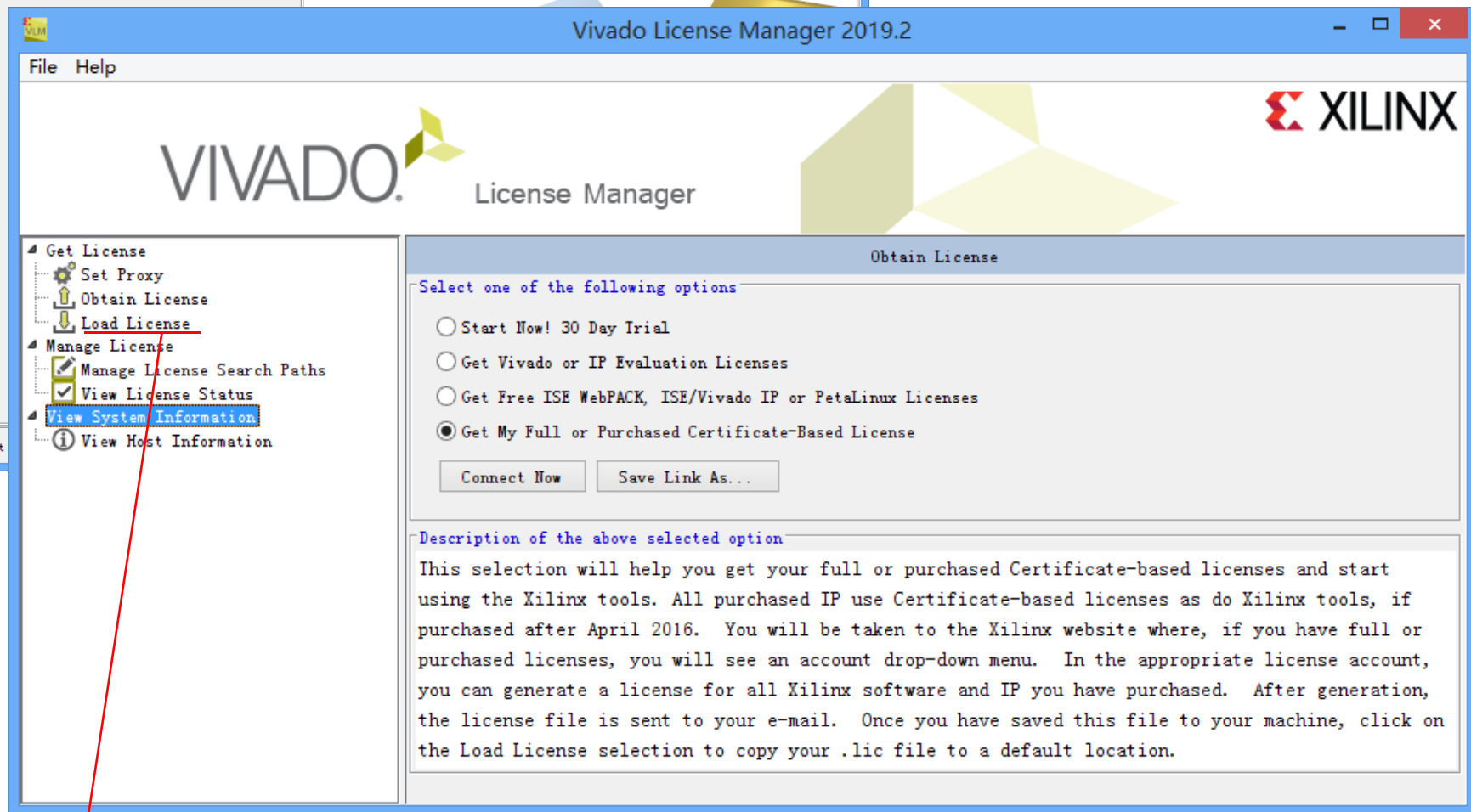
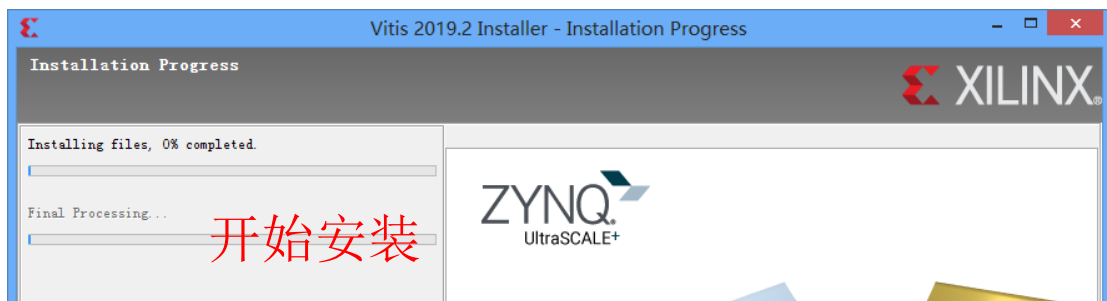
- 接受所有的License协议



• UltraScale系列器件可以不装，开发这个系列对计算机内存容量要求很高。



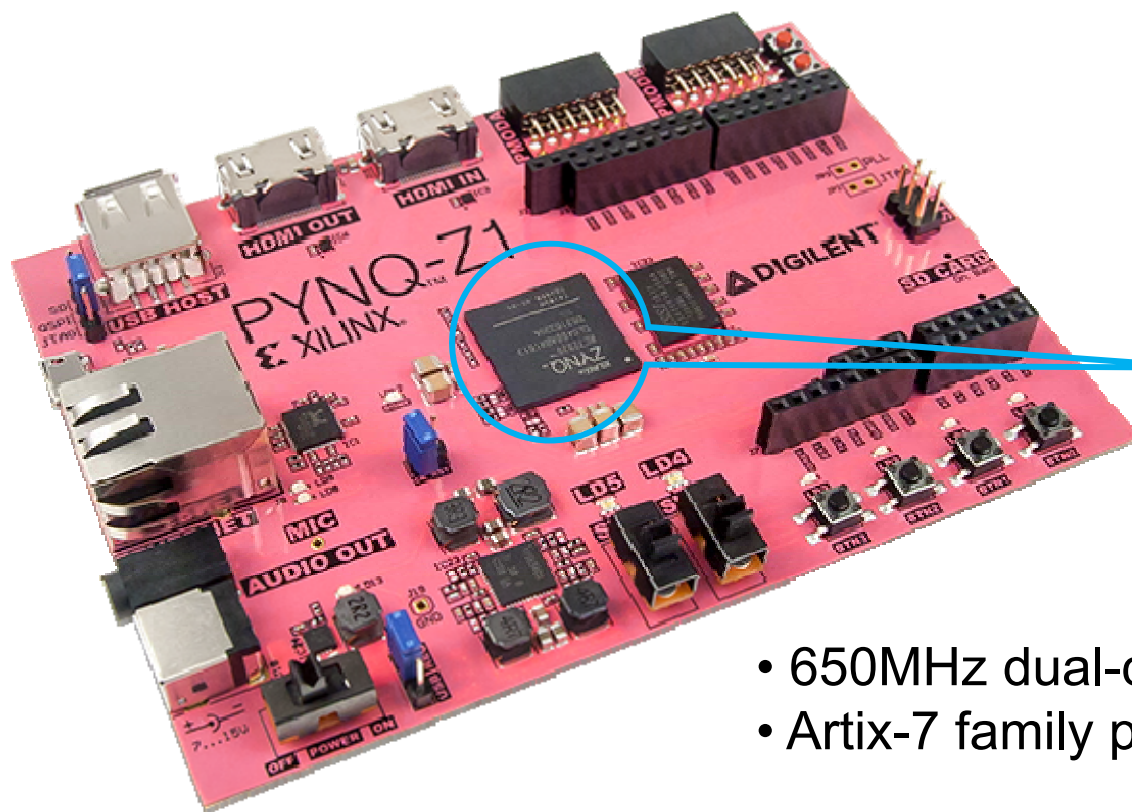
根据安装选项不同，安装占用磁盘空间62~76GB以上，加上安装包占用32GB，空闲硬盘应该有100GB以上。最好用SSD硬盘，以提高软件安装和启动速度。



安装完成后，选择Load License加载已有的License文件

硬件平台

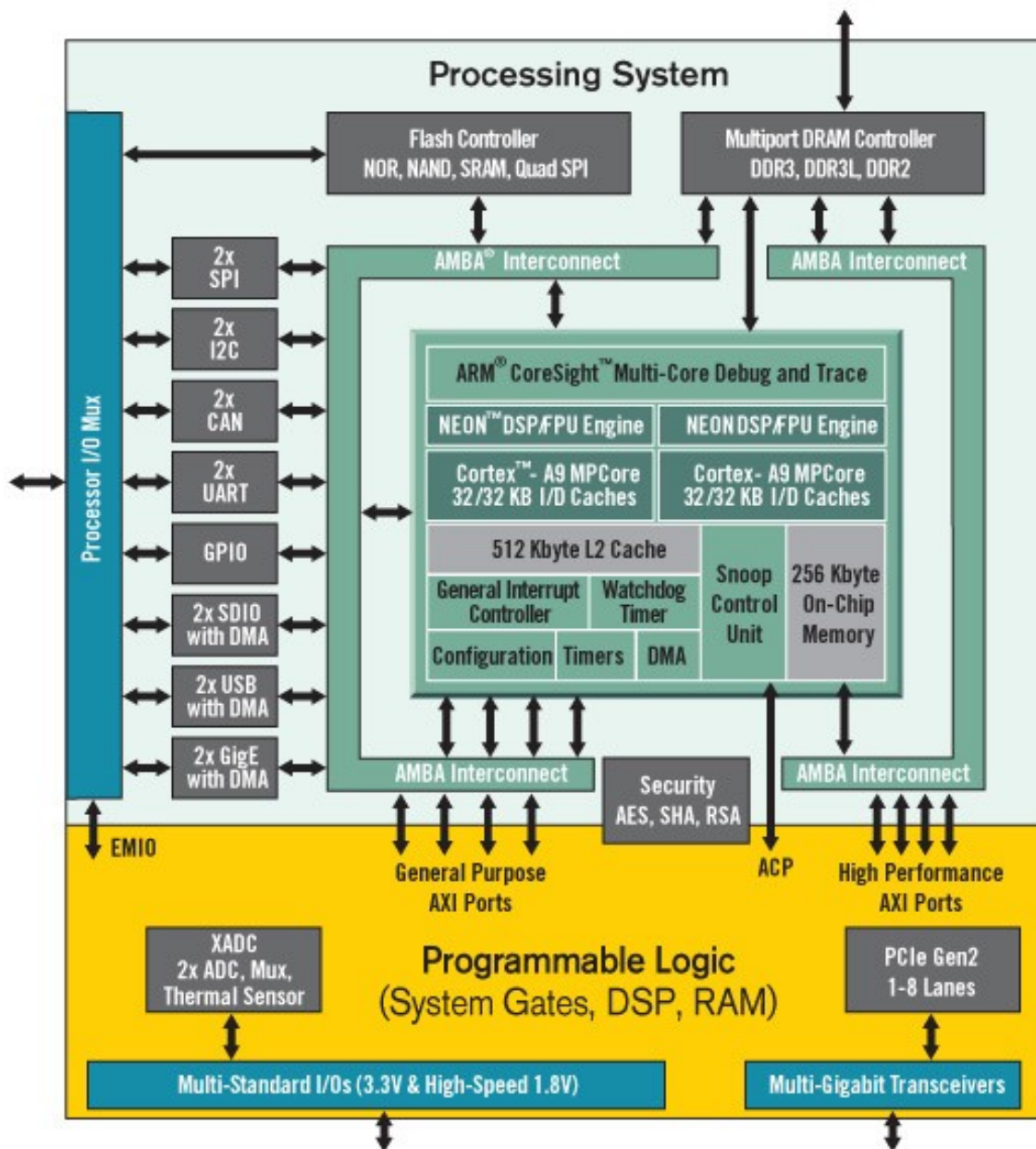
<http://www.pynq.io/>



- 650MHz dual-core Cortex-A9 processor
- Artix-7 family programmable logic

实验硬件平台是Digilent的“PYNQ-Z1”或TUL的“PYNQ-Z2”板，
采用的FPGA是Xilinx ZYNQ-7000系列的XC7Z020-1CLG400C。

ZYNQ-7000结构

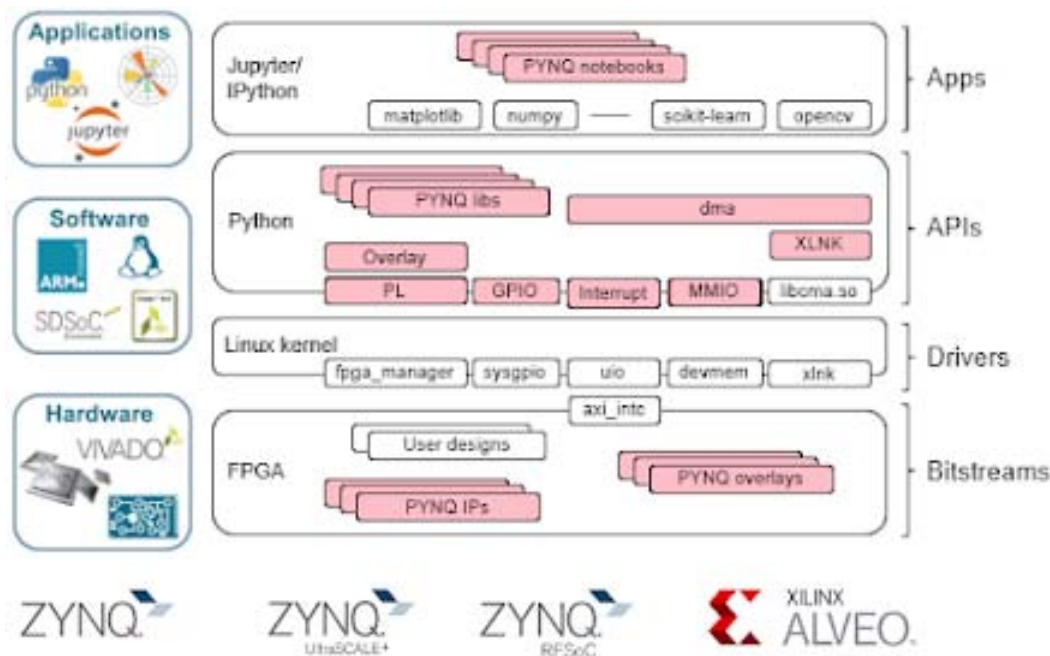


• PS: **P**rocessing **S**ystem
硬核嵌入式处理器及外设

• PL: **P**rogrammable **L**ogic
FPGA通用可编程逻辑资源

以下设计仅用其中的**PL**部分

PYNQ项目架构



- 硬件层是ZYNQ系列等嵌入式处理器为中心的FPGA
- 软件底层是Linux内核及硬件外设的驱动程序
- 应用编程接口通过Overlay机制支持Python语言控制板上硬件资源
- 用户应用程序APP采用Python语言调用API函数来操作底层硬件

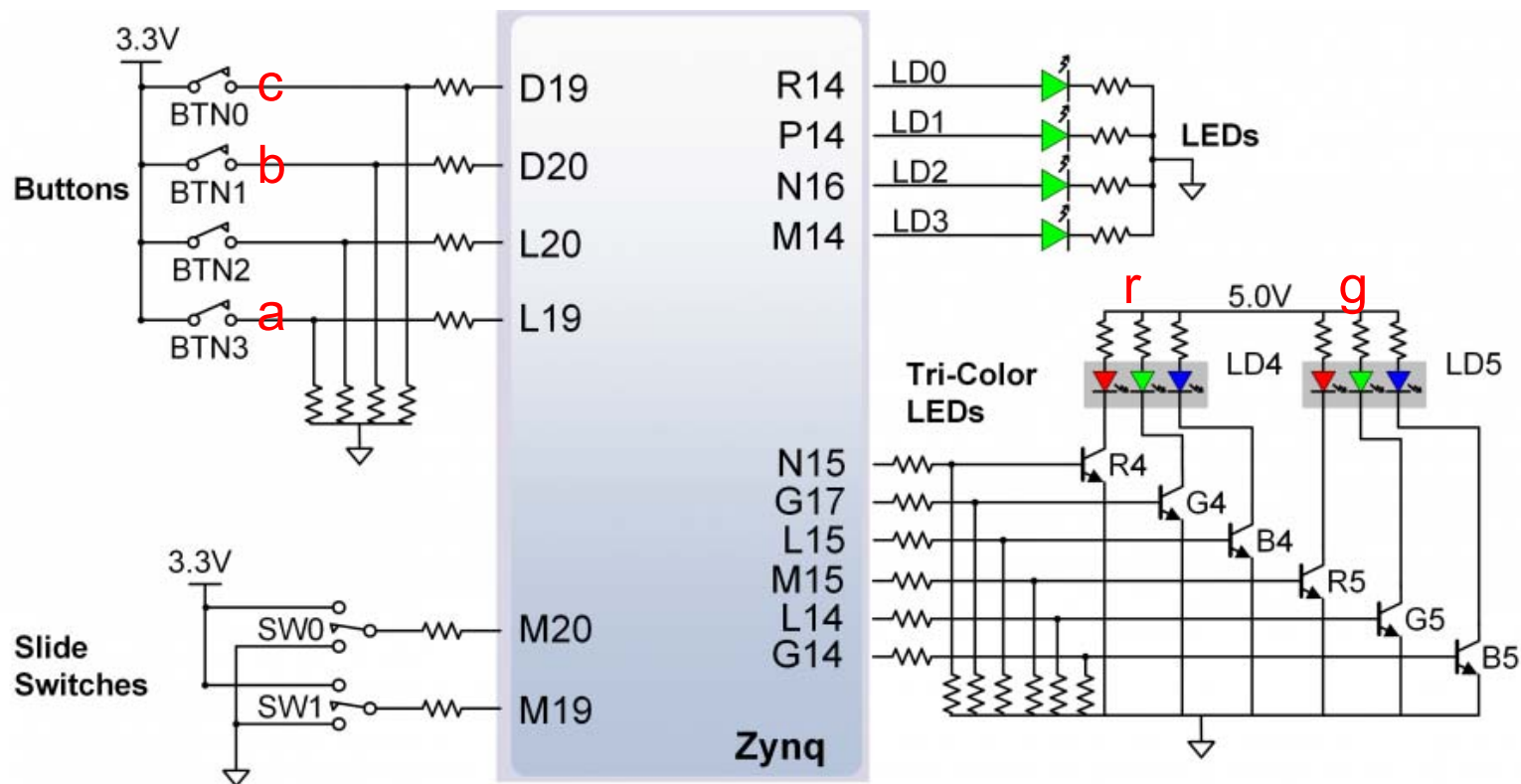
设计一个简单的举重结果裁判电路，要求：

- 假设举重比赛有三名裁判，其中一名是主裁判。若某裁判认为运动员举起就按自己的按键，否则不按键。
- 裁判结果用红、绿两灯表示：红绿灯全亮表示举起；只红灯亮表示需要再研究；两灯都不亮表示未举起。
- 灯亮的判决条件：
 1. 三名裁判都按键 ——红、绿灯全亮；
 2. 两名裁判按键，一名是主裁判 ——红、绿灯全亮；
 3. 一名主裁判或两名副裁判按键—— 只红灯亮；
 4. 其余情况（只一名副裁判按键或三名裁判都不按键）—— 红、绿灯全不亮。

设计步骤

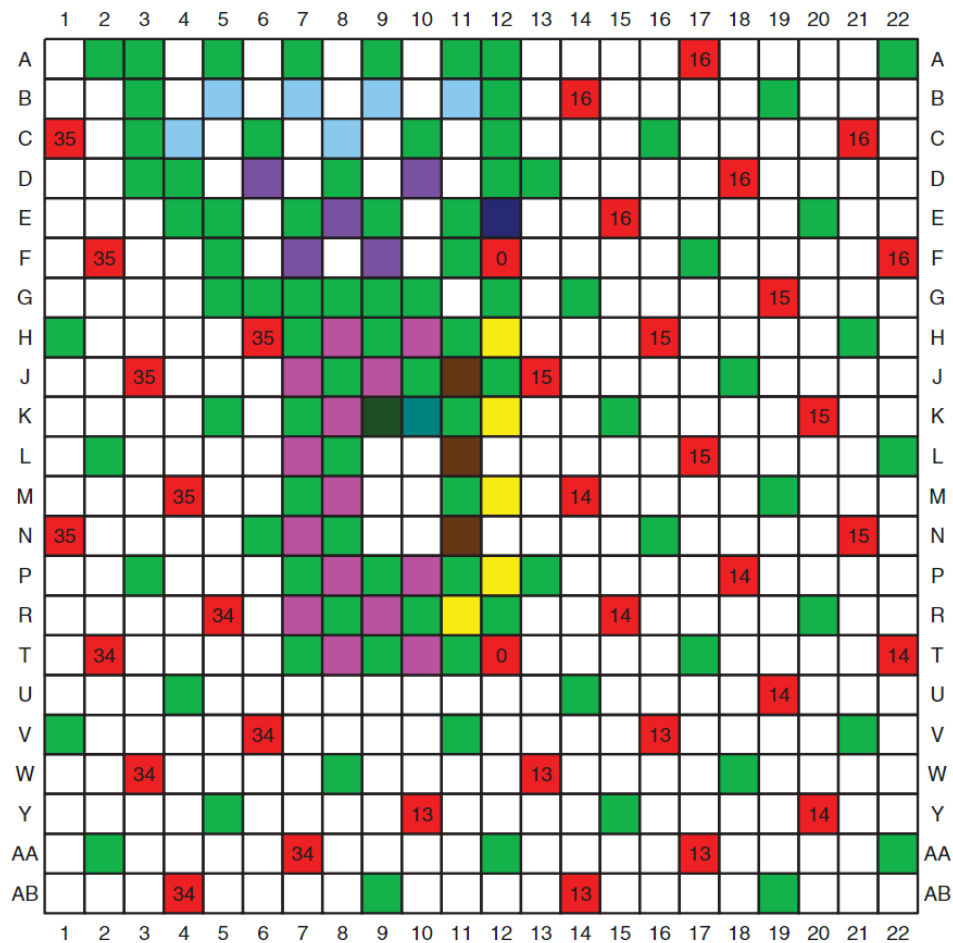
1. 设计FPGA外围应用电路
2. 设计的输入
使用Verilog HDL语言输入
3. 设计的综合和实现
4. 设计的仿真测试
5. FPGA配置（下载）
6. 在板验证实际功能

设计FPGA外部应用电路



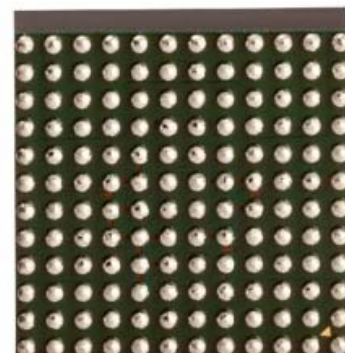
- 先确定FPGA外部的应用电路，外电路连线不同影响内部逻辑实现，图中是PYNQ-Z1板上按键和LED相关的GPIO（通用IO口）接线。

关于BGA/LGA封装的管脚编号

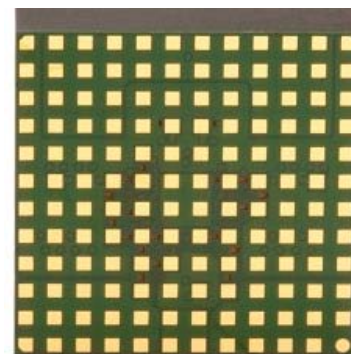


字母共20个，不含I/O/Q/S/X/Z

BGA: ball grid array
/球状栅格阵列



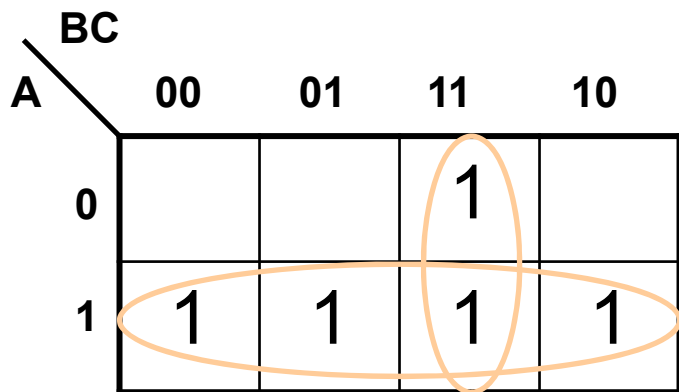
LGA: land grid array
/平面网格阵列



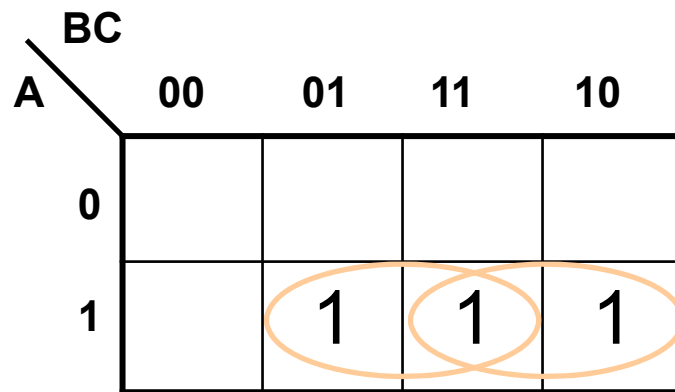
真值表 (Truth Table)

| A(BTN3) | B(BTN1) | C(BTN0) | R(LD5) | G(LD4) |
|---------|---------|---------|--------|--------|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |

卡诺图(karnaugh map)逻辑化简



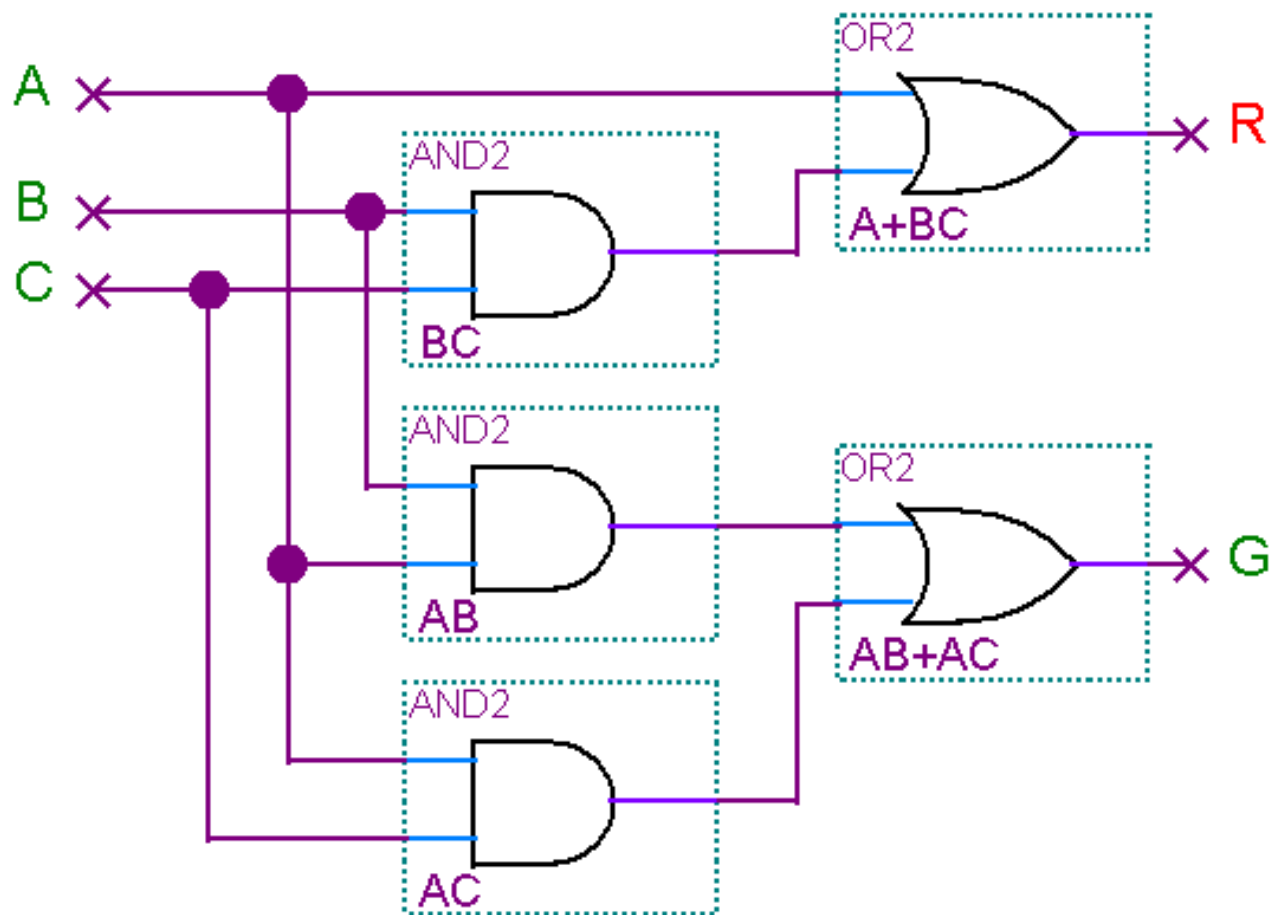
$$\text{RED} = A + BC$$



$$\text{GREEN} = AC + AB$$

- 方格取值、相邻、最小项（乘积之和/SOP）
- 画圈包围相邻的 2^n （2、4、8、16...）个方格，逻辑化简

逻辑电路实现



2. 设计的输入

- Xilinx Vivado 2019.2软件界面



Vivado仅支持7系列、ZYNQ等新器件，老器件要用ISE Design Suite 14.7。

- 原理图方式输入

ISE里定义了一些基本的门级元器件库，可以用来画电路图。Vivado已不支持原理图输入。

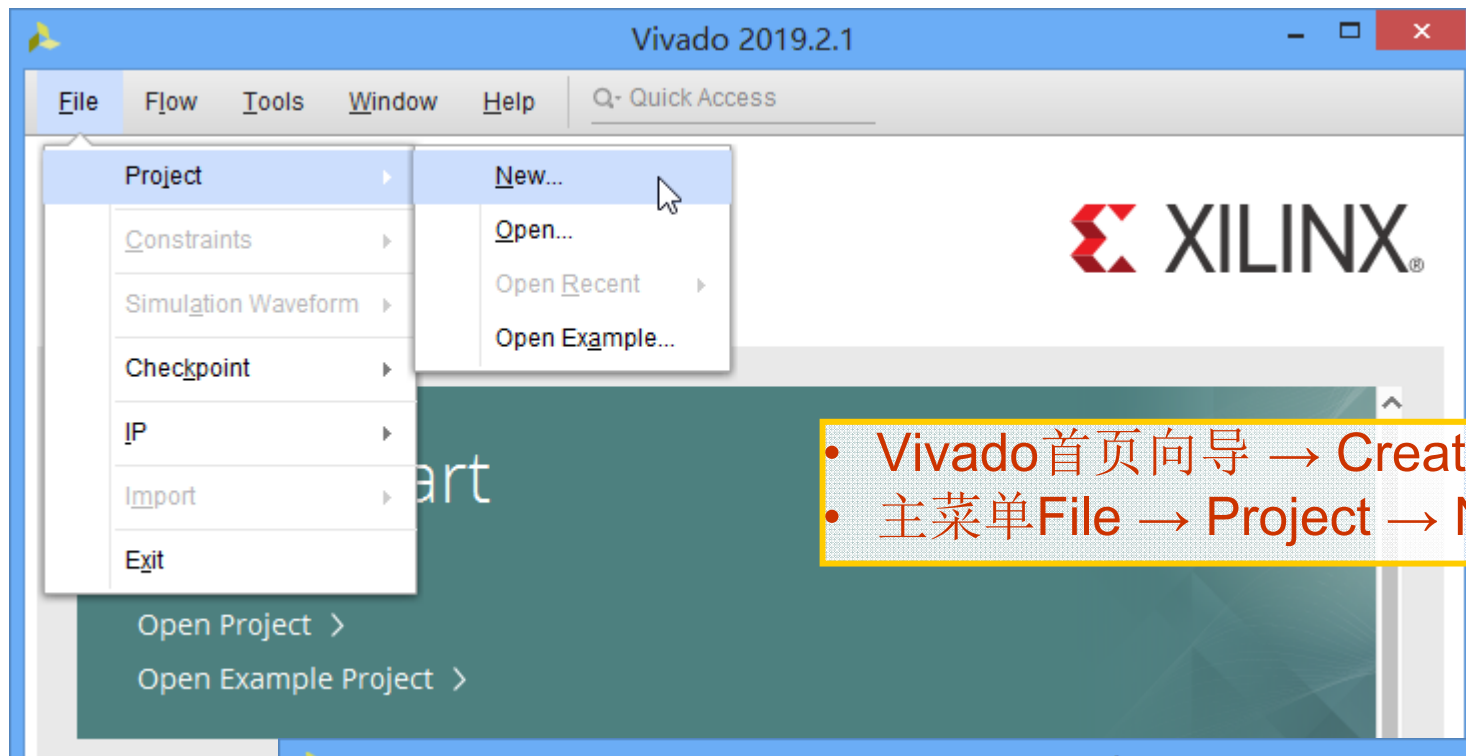
- HDL语言输入

Verilog HDL、VHDL、SystemVerilog等。

- 网表文件Netlist

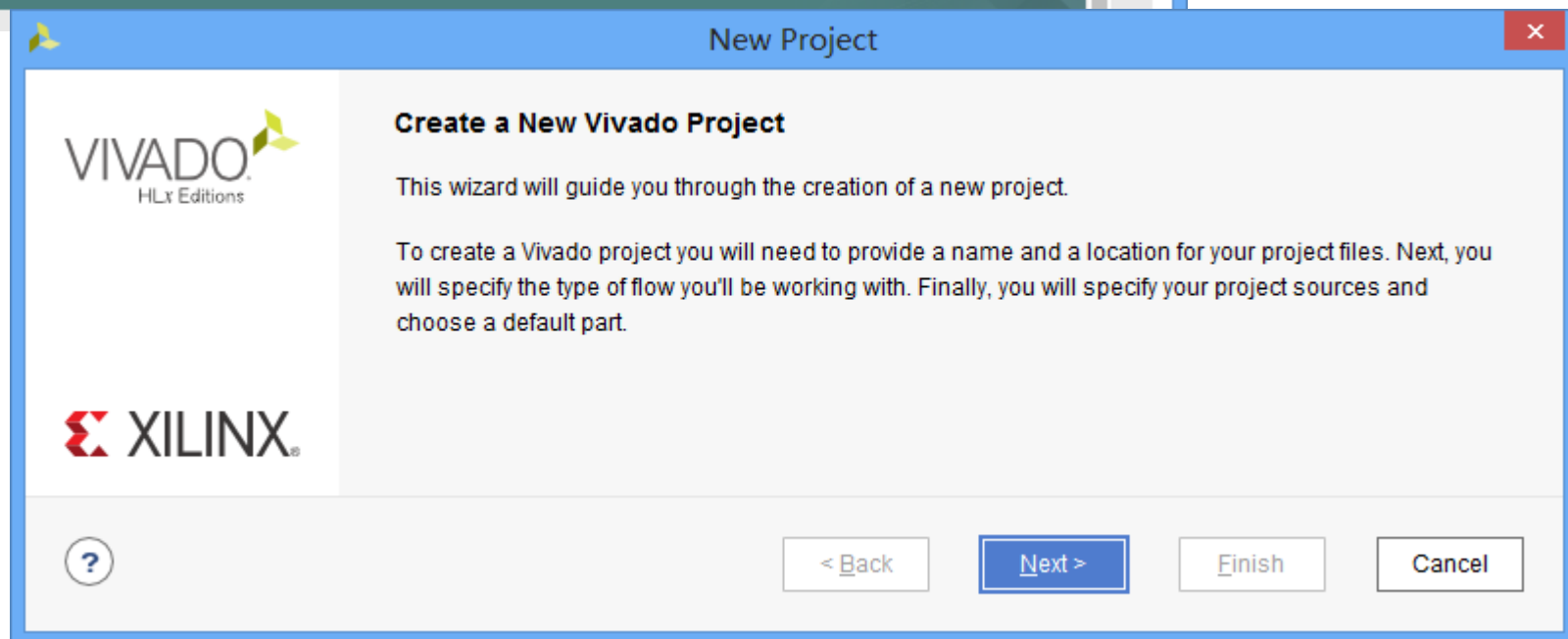
来自其它工程的电子设计互换文件edif等。

以下使用Verilog HDL语言输入



- Vivado 首页向导 → Create Project
- 主菜单 File → Project → New ...

建立新工程



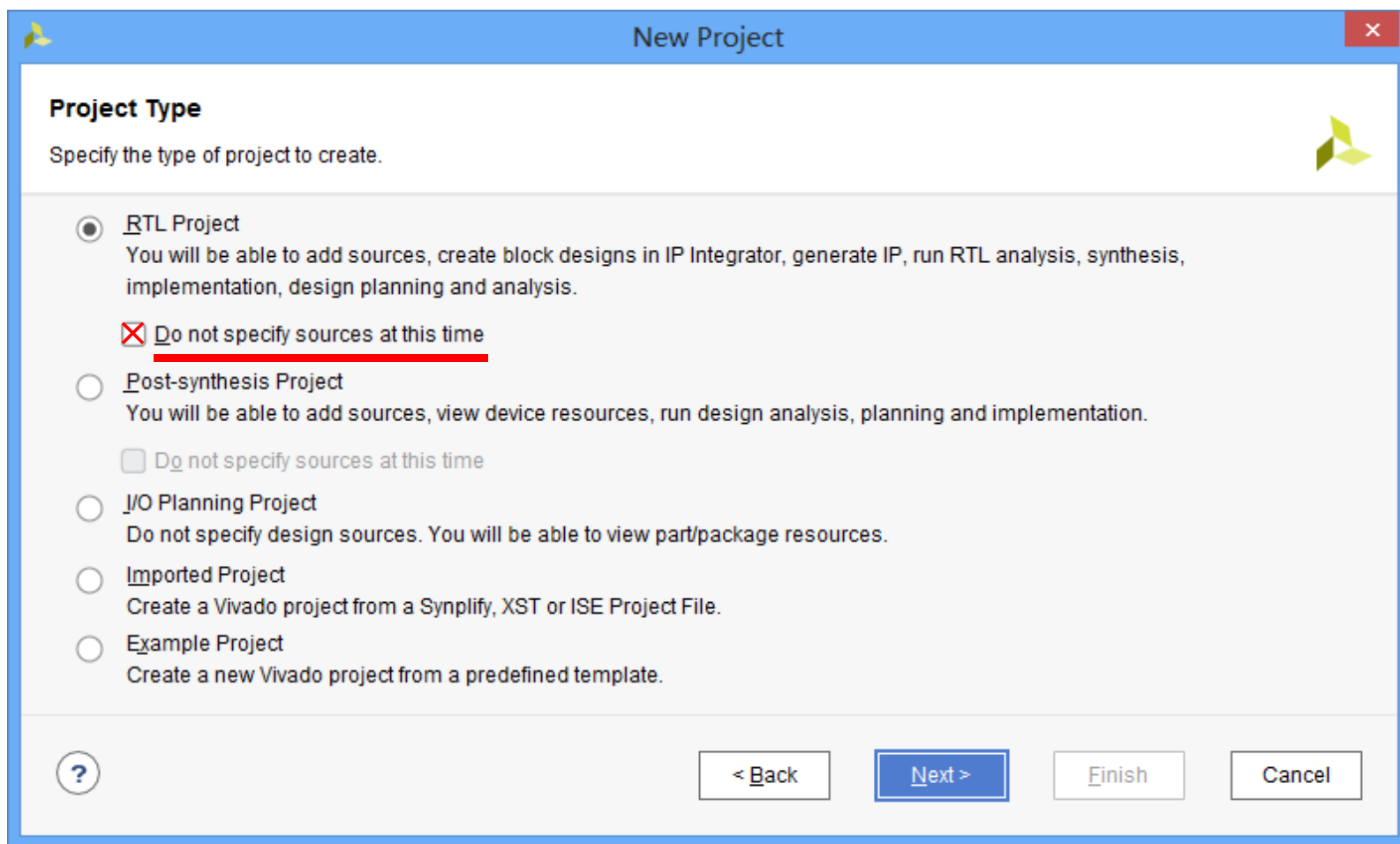
New Project Wizard/新工程生成向导

1. 浏览到存放工程目录的路径

2. 给新工程起名



3. 向导自动用工程名给新工程单独新建一个目录，设计过程将生成数十个新文件


选择工程类型



新建RTL工程，选择暂不添加源代码。

选择器件: XC7Z020CLG400-1

New Project

Default Part

Choose a default Xilinx part or board for your project.

Parts | Boards

[Reset All Filters](#)

Category: General Purpose

Package: clg400

Temperature: All Remaining




Family: Zynq-7000


Speed: -1

Static power: All Remaining

Search:

| Part | I/O Pin Count | Available IOBs | LUT Elements | FlipFlops | Block RAMs | Ultra RAMs | DSPs | Gt |
|------------------|---------------|----------------|--------------|-----------|------------|------------|------|----|
| xc7z007sclg400-1 | 400 | 100 | 14400 | 28800 | 50 | 0 | 66 | 0 |
| xc7z010clg400-1 | 400 | 100 | 17600 | 35200 | 60 | 0 | 80 | 0 |
| xc7z014sclg400-1 | 400 | 125 | 40600 | 81200 | 107 | 0 | 170 | 0 |
| xc7z020clg400-1 | 400 | 125 | 53200 | 106400 | 140 | 0 | 220 | 0 |





< Back

Next >

Finish

Cancel

添加源文件

TestSch - [D:/FPGA/Vivado/TestSch/TestSch.xpr] - Vivado 2019.2.1

File Edit Flow Tools Reports Window Layout View Help Q- Quick Access Ready

Flow Navigator PROJECT MANAGER - TestSch

PROJECT MANAGER

- Settings
- Add Sources**
- Language Templates
- IP Catalog

IP INTEGRATOR

- Create Block Design
- Open Block Design
- Generate Block Design

SIMULATION

- Run Simulation

RTL ANALYSIS

- Open Elaborated Design

SYNTHESIS

- Run Synthesis
- Open Synthesized Design

IMPLEMENTATION

- Run Implementation

Sources

- Design Sources
 - Constraints
 - constrs_1
 - Simulation Sources
 - sim_1
 - Utility Sources
 - utils_1

Hierarchy Libraries Compile Order

Properties

Select an object to see properties

Project Summary

Overview | Dashboard

Settings Edit

Project name: TestSch
Project location: D:/FPGA/Vivado/TestSch
Product family: Zynq-7000
Project part: xc7z020clg400-1
Top module name: Not defined
Target language: Verilog
Simulator language: Mixed

Synthesis

Status: Not started
Messages: No errors or warnings
Part: xc7z020clg400-1
Strategy: Vivado Synthesis Defaults

Implementation

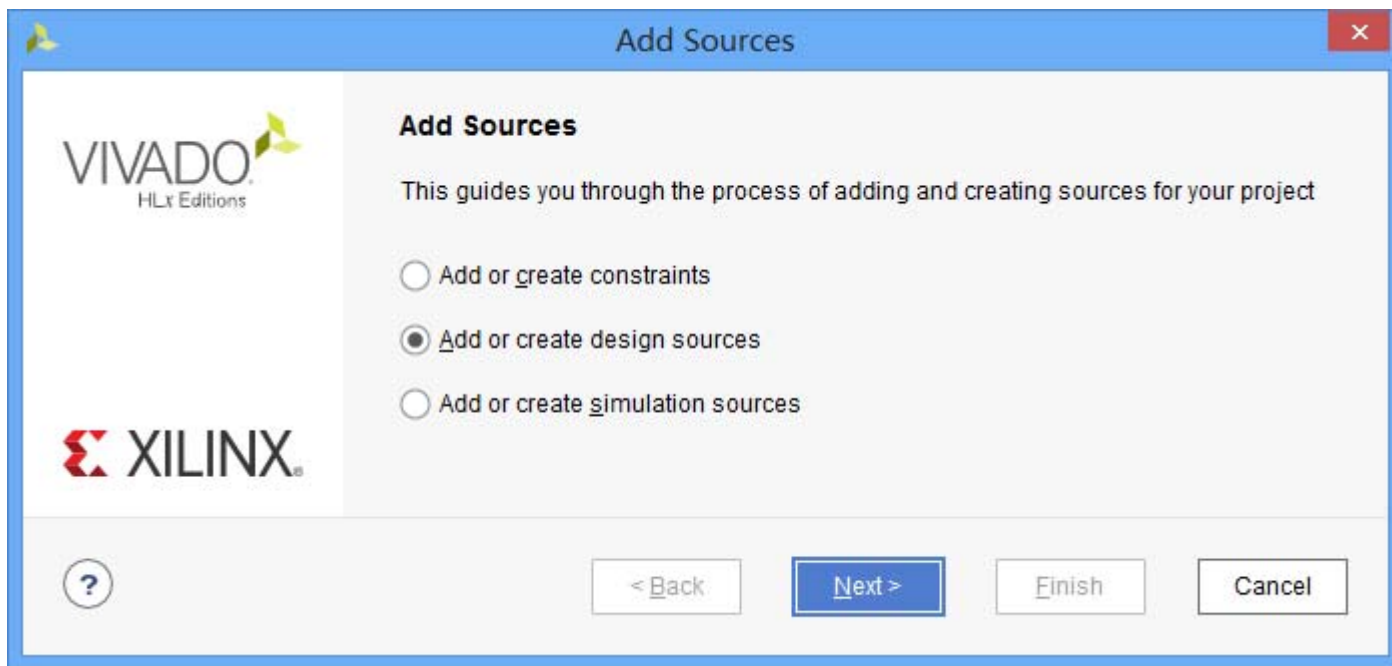
Status:
Messages:
Part:
Strategy:

Tcl Console Messages Log Reports **Design Runs**

| Name | Constraints | Status | WNS | TNS | WHS | THS | TPWS | Total Power | Failed Routes | LUT | FF | BRAM | URAM |
|---------|-------------|-------------|-----|-----|-----|-----|------|-------------|---------------|-----|----|------|------|
| synth_1 | constrs_1 | Not started | | | | | | | | | | | |
| impl_1 | constrs_1 | Not started | | | | | | | | | | | |

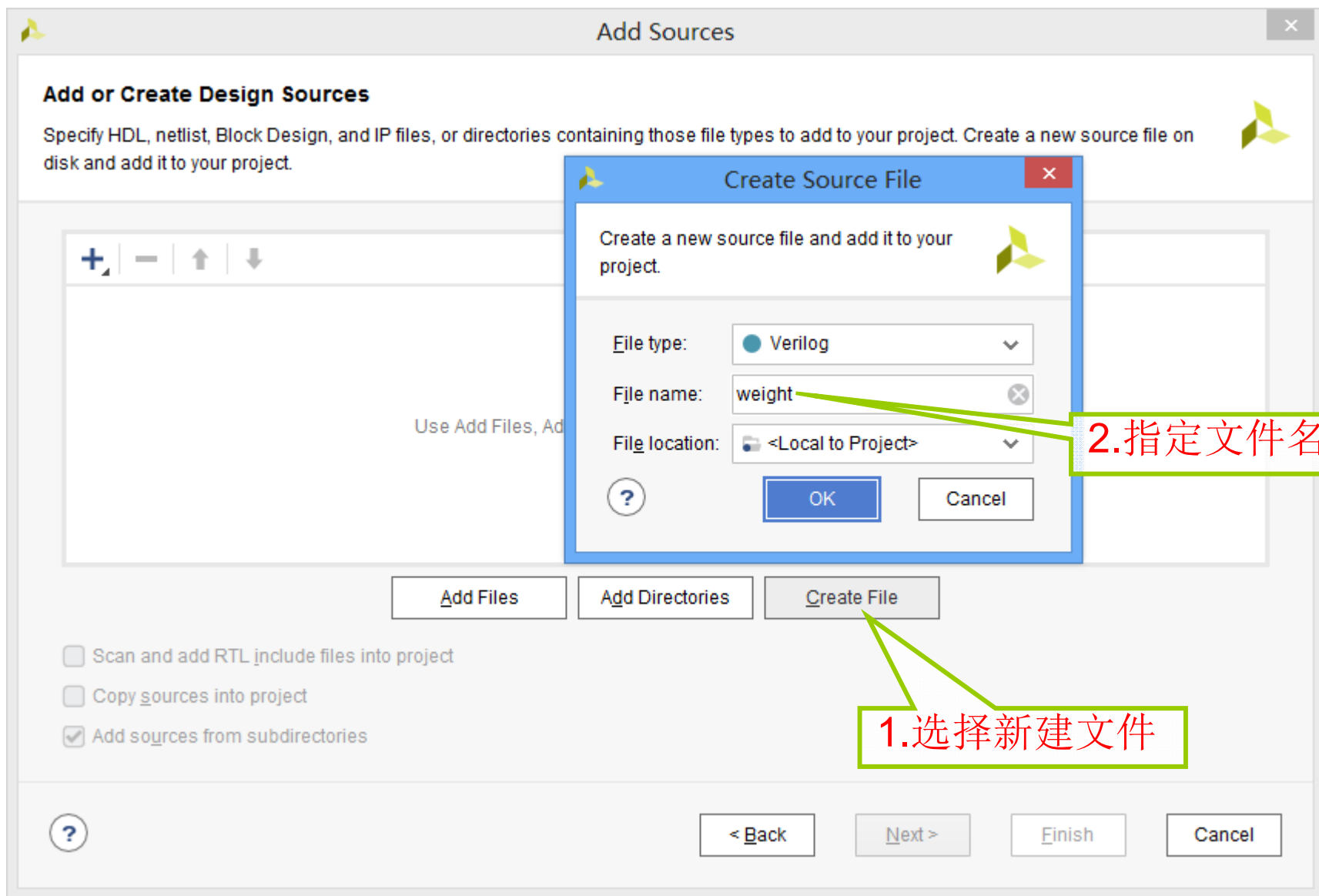
指定和/或创建源文件添加到工程

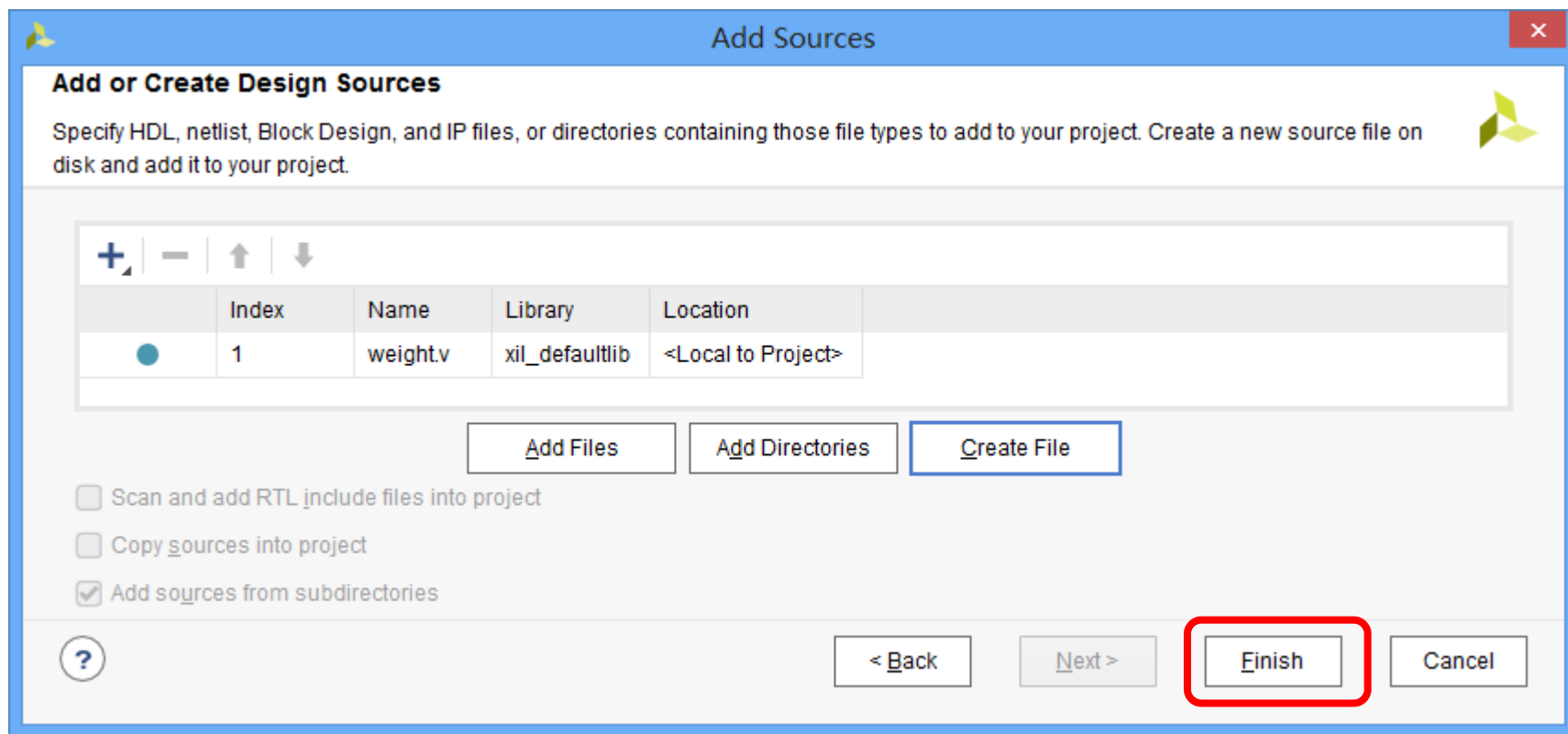
用新代码向导向工程中添加新代码



添加/新建源文件有三种类型可选：

- **Constraints:** 约束文件，指定管脚位置，面积、时钟等约束
- **Design source:** 逻辑设计的源文件
- **Simulation Source:** 测试向量文件





按Finish键，弹出模块定义向导，定义模块Module的框架和端口。

Define Module

Define a module and specify I/O Ports to add to your source file.
For each port specified:
MSB and LSB values will be ignored unless its Bus column is checked.
Ports with blank names will not be written.

Module Definition

Module name:

I/O Port Definitions

+ - ↑ ↓

| Port Name | Direction | Bus | MSB | LSB |
|-----------|-----------|--------------------------|-----|-----|
| a | input | <input type="checkbox"/> | 0 | 0 |
| b | input | <input type="checkbox"/> | 0 | 0 |
| c | input | <input type="checkbox"/> | 0 | 0 |
| r | output | <input type="checkbox"/> | 0 | 0 |
| g | output | <input type="checkbox"/> | 0 | 0 |
| | input | <input type="checkbox"/> | 0 | 0 |

? OK Cancel

- 此处生成Module框架的端口声明部分
- 如果点击Cancel，可以自己在源文件里录入源码

向导自动生成空的module框架代码

The screenshot displays the Vivado 2019.2.1 IDE interface. The top menu bar includes File, Edit, Flow, Tools, Reports, Window, Layout, View, and Help. The main workspace is divided into several panes:

- Flow Navigator:** Shows the project hierarchy with sections for PROJECT MANAGER, IP INTEGRATOR, SIMULATION, RTL ANALYSIS, SYNTHESIS, and IMPLEMENTATION.
- PROJECT MANAGER - TestSch:** Contains the Sources pane, Source File Properties, and Design Runs.
- Sources:** Lists Design Sources (1), Constraints, Simulation Sources (1), and Utility Sources. The 'weight (weight.v)' source is highlighted.
- Source File Properties:** Shows the 'weight.v' file is enabled.
- Design Runs:** A table showing the status of synthesis and implementation runs.
- weight.v:** The main editor window showing the Verilog code for the 'weight' module.

The Verilog code in 'weight.v' is as follows:

```
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21
22
23 module weight (
24     input a,
25     input b,
26     input c,
27     output r,
28     output g
29 );
30
31 endmodule
32
```

A red arrow points to the 'endmodule' line (line 31) with the text: 此处继续修改或添加代码 (Continue modifying or adding code here).

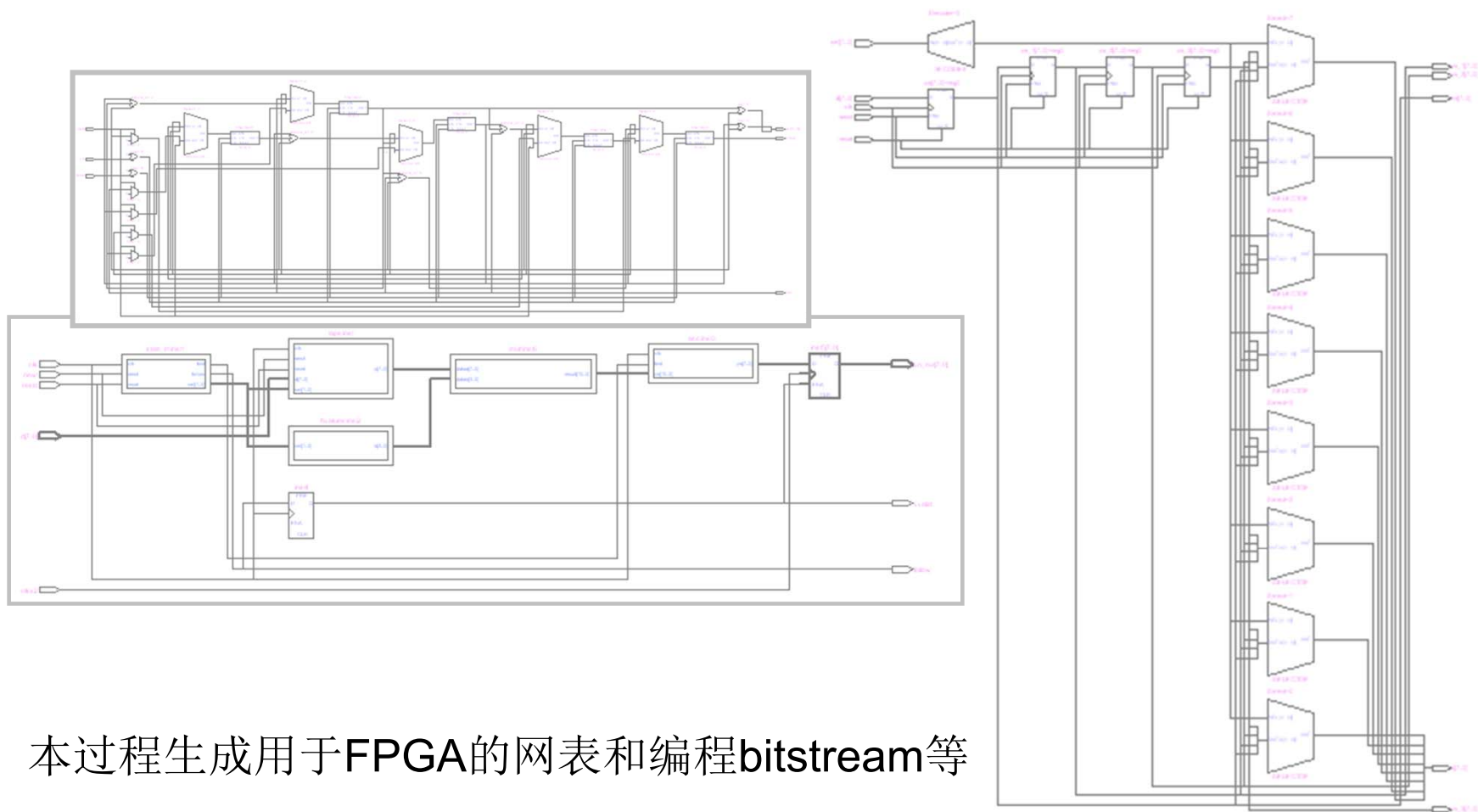
| Name | Constraints | Status | WNS | TNS | WHS | THS | TPWS | Total Power | Failed Routes | LUT | FF | BRAM | URAM |
|---------|-------------|-------------|-----|-----|-----|-----|------|-------------|---------------|-----|----|------|------|
| synth_1 | constrs_1 | Not started | | | | | | | | | | | |
| impl_1 | constrs_1 | Not started | | | | | | | | | | | |

Verilog HDL代码编辑器

```
22 module weight(a, b, c, r, g);
23     input a, b, c;
24     output r, g;
25     reg r, g;
26     always @(a or b or c)
27     begin
28         case({a, b, c})
29             3'b000,
30             3'b001,
31             3'b010: {r, g}=2'b00;
32             3'b011,
33             3'b100: {r, g}=2'b10;
34             3'b101,
35             3'b110,
36             3'b111: {r, g}=2'b11;
37             default: {r, g}=2'b00;
38         endcase
39     end
40 endmodule
```

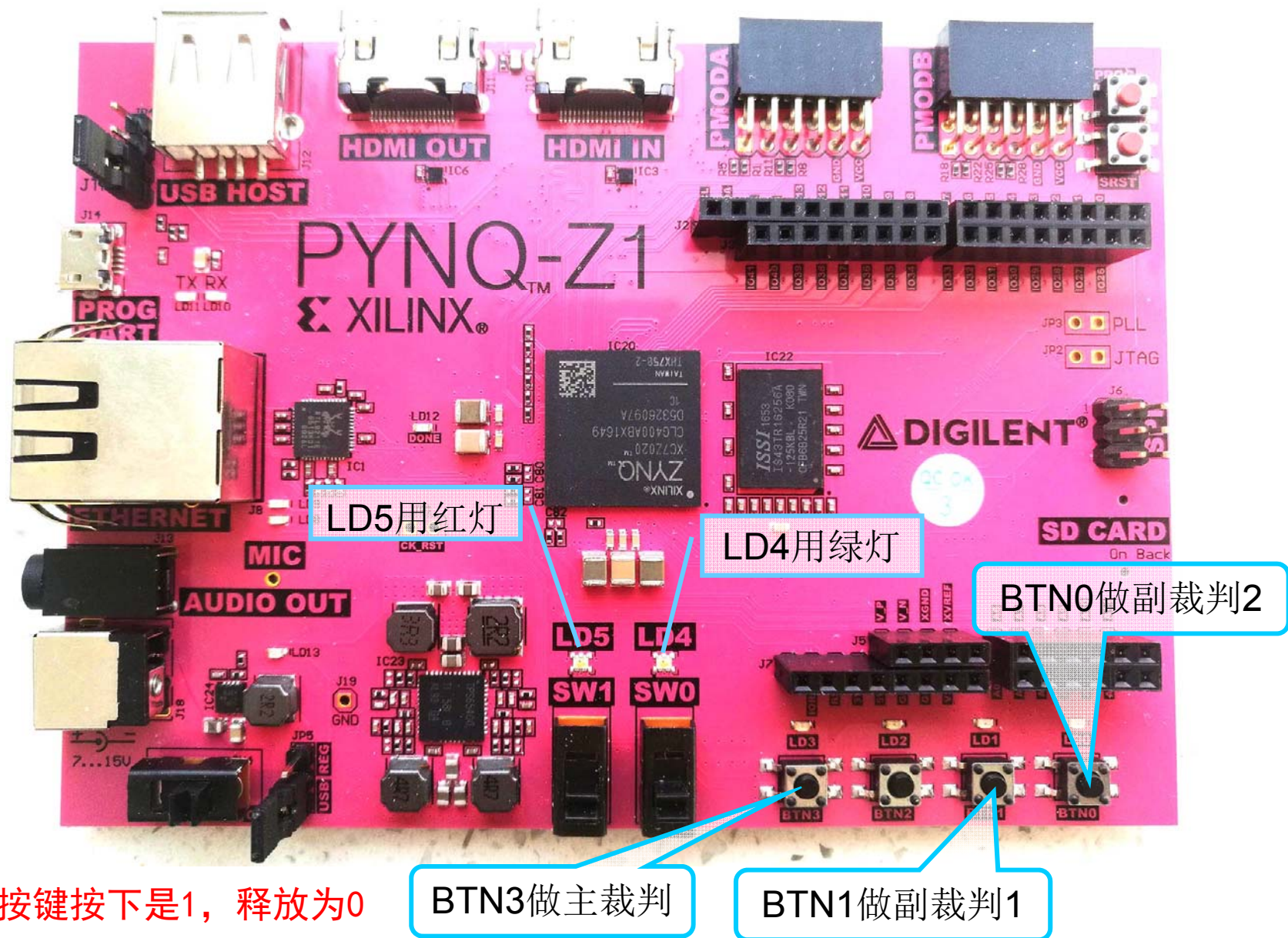
用Verilog HDL的case语句根据前面真值表列输入输出对应关系，之后的综合和实现工作交给软件去做。保存时会自动进行语法检查。

3.设计的综合和实现



本过程生成用于FPGA的网表和编程bitstream等

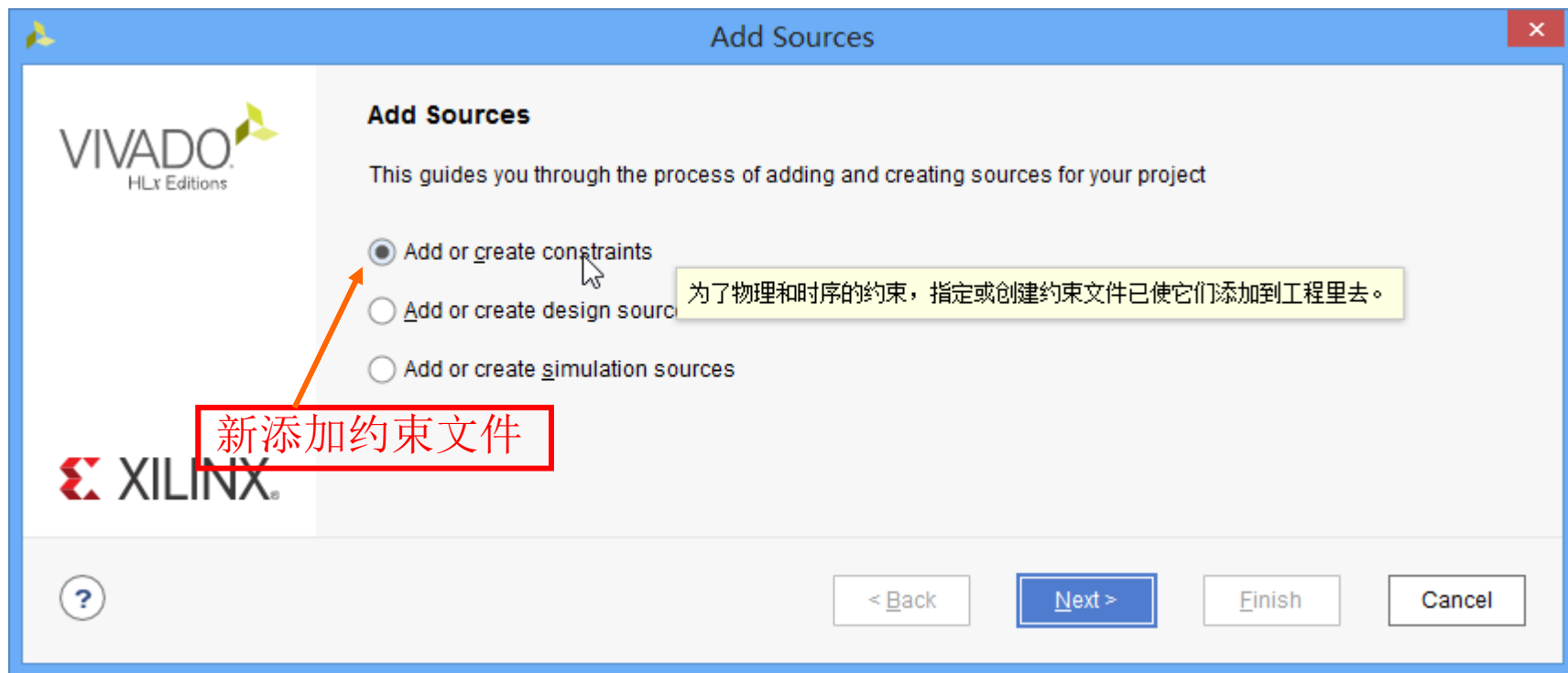
按键和指示灯设定

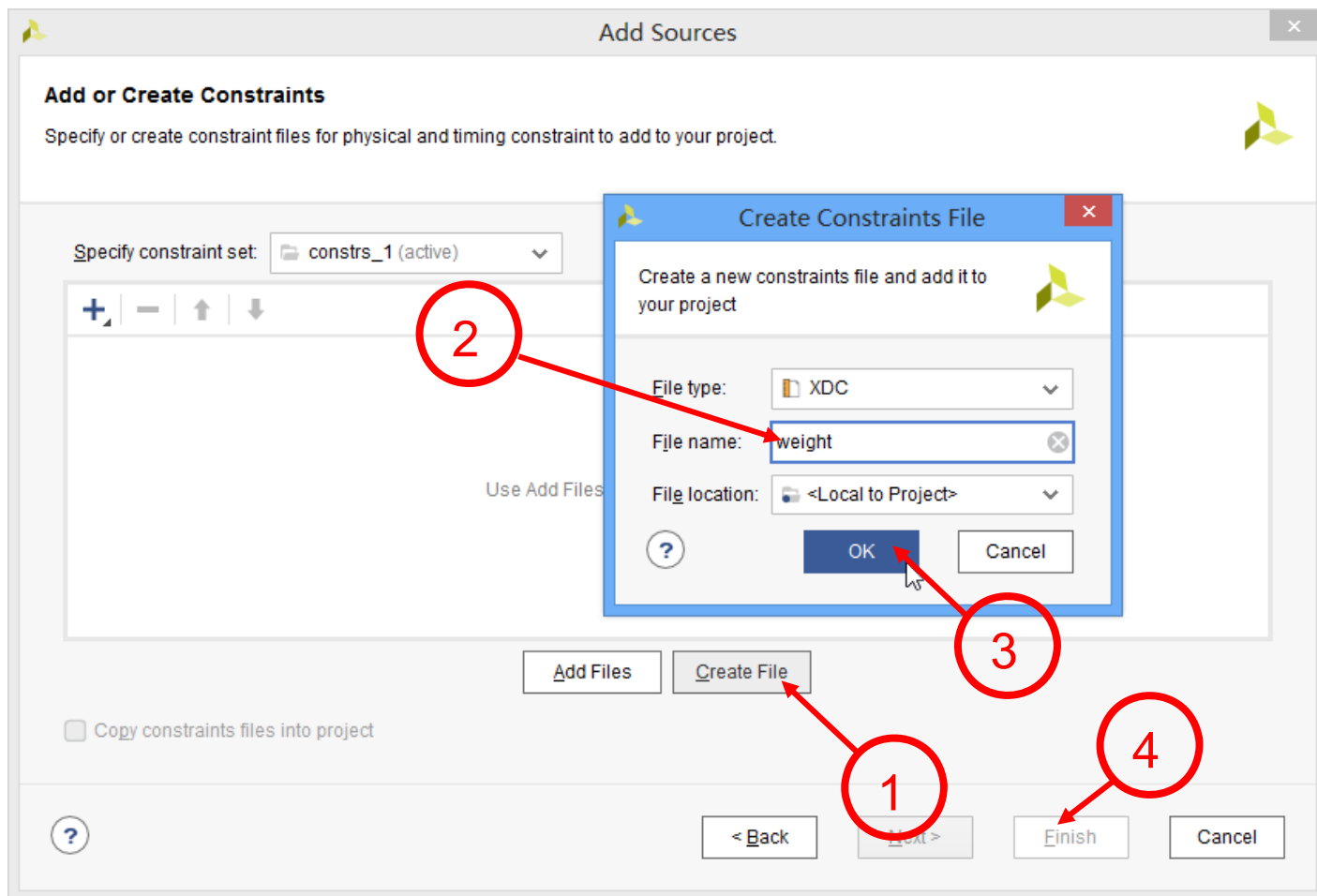


按键按下是1，释放为0

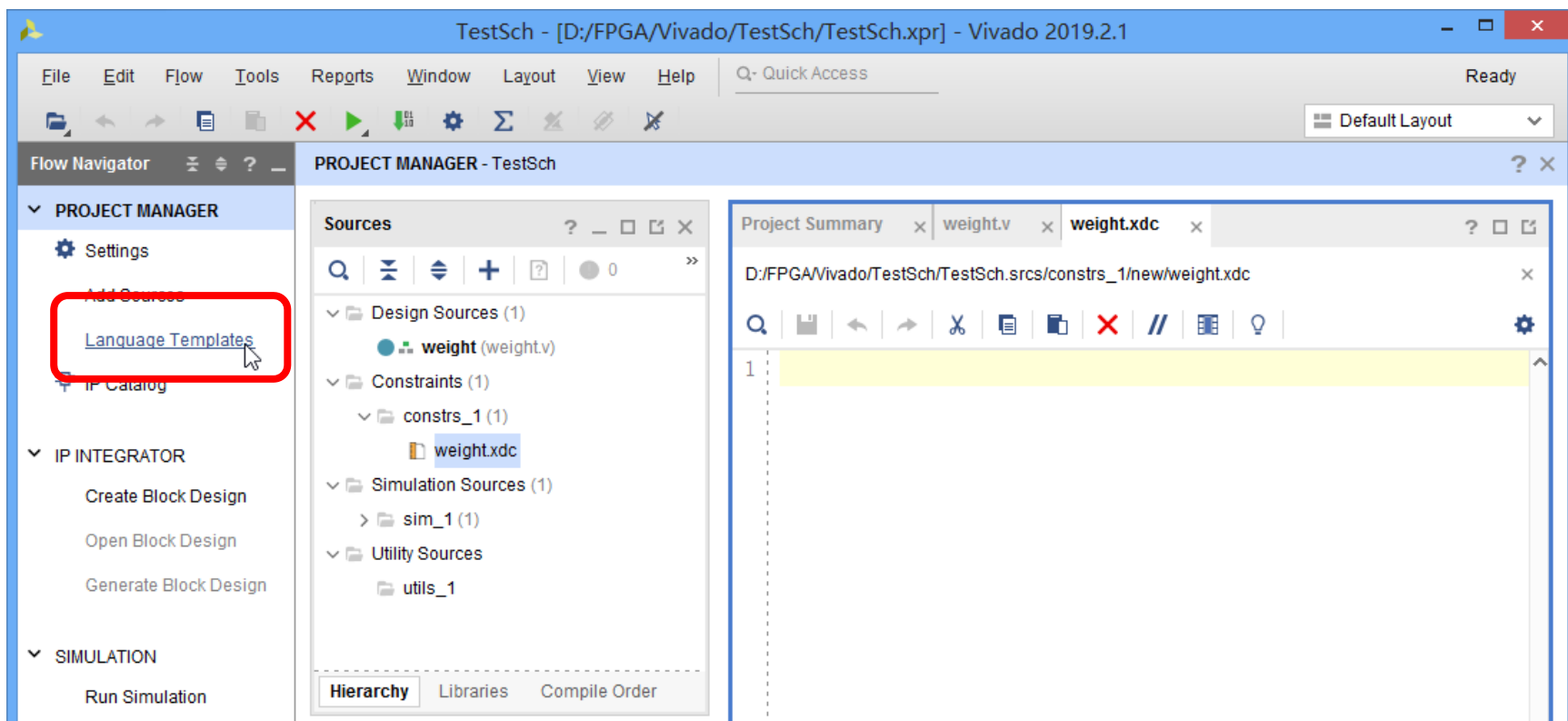
映射IO口位置

为了最终的物理实现，需要指定各外部端口所用的管脚。Xilinx工具采用“约束” constraint的方式指定IO管脚，向工程添加用户约束文件：
ISE的用户约束文件是*.ucf，并提供了易用的管脚约束编辑器—PACE；
Vivado的用户约束文件是*.xdc，格式与ucf不同，未找到PACE工具。

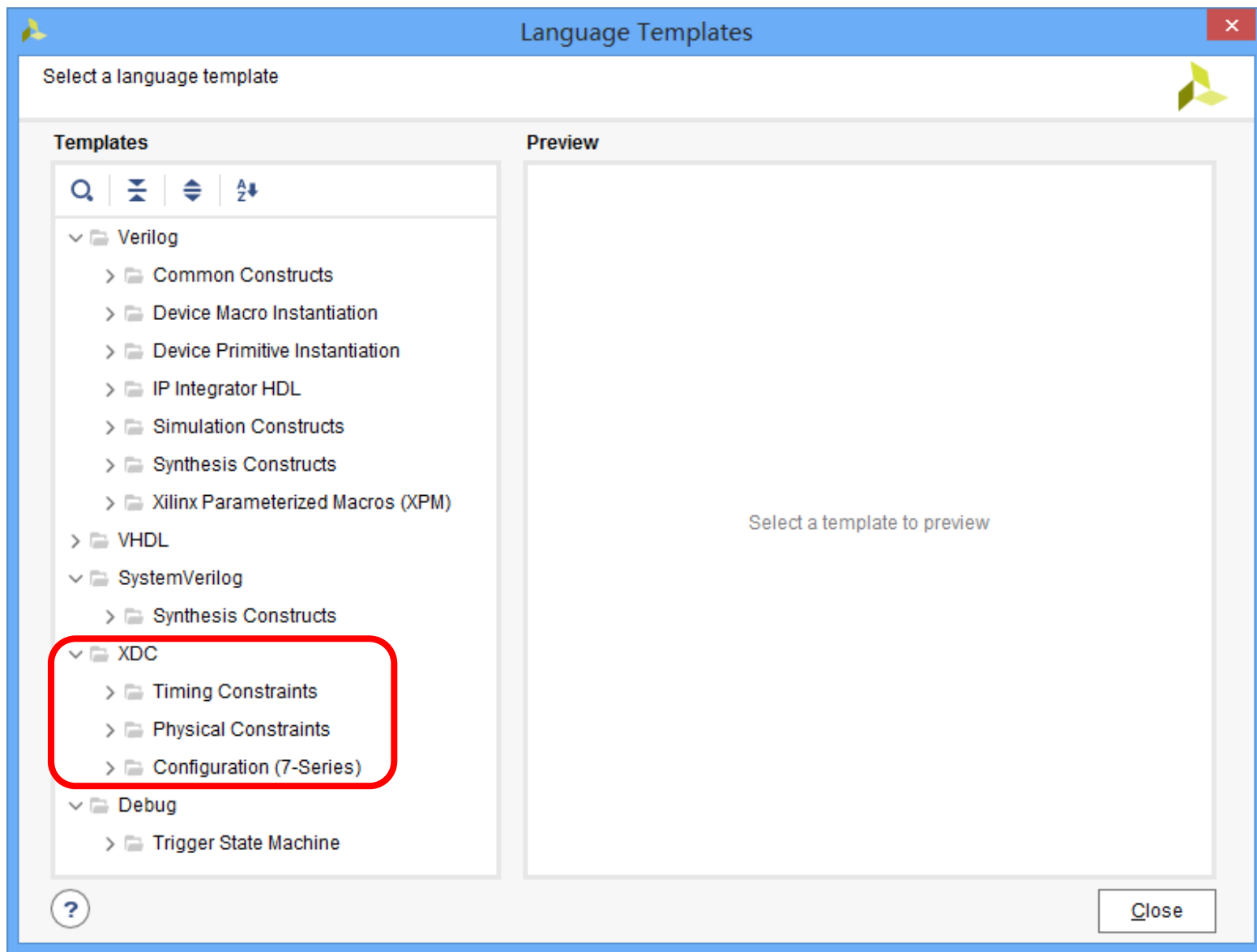




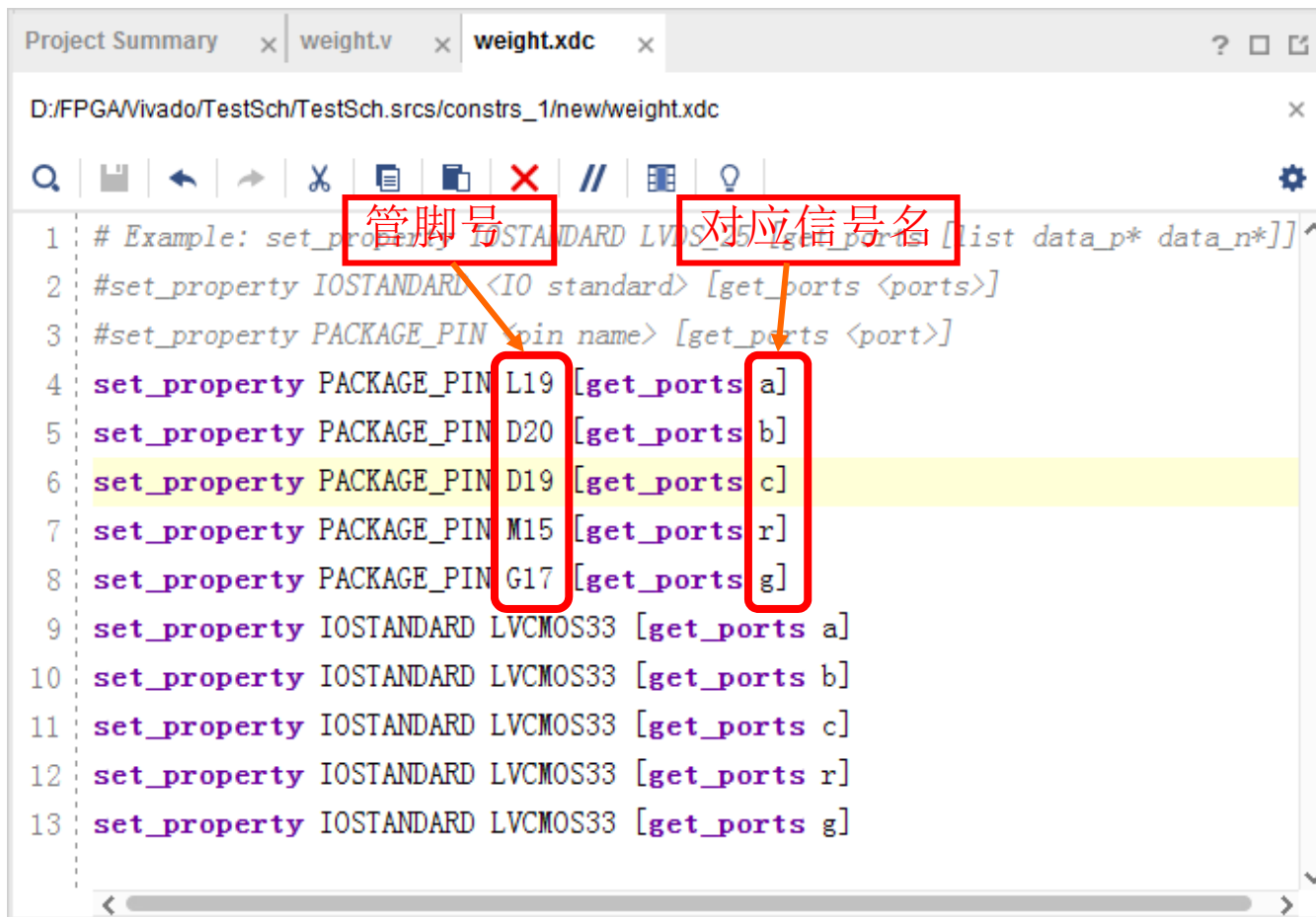
新建并指定约束文件名称，一个空白约束文件自动加入工程内。



空白weight.xdc文件需要手工填写内容，Vivado的xdc文件与ISE的ucf语法格式大有不同，虽未找到PACE工具，但提供了语言模板，如图单击左侧Flow Navigator里的“Language Templates”。



XDC文件语句的模板在XDC分组下。

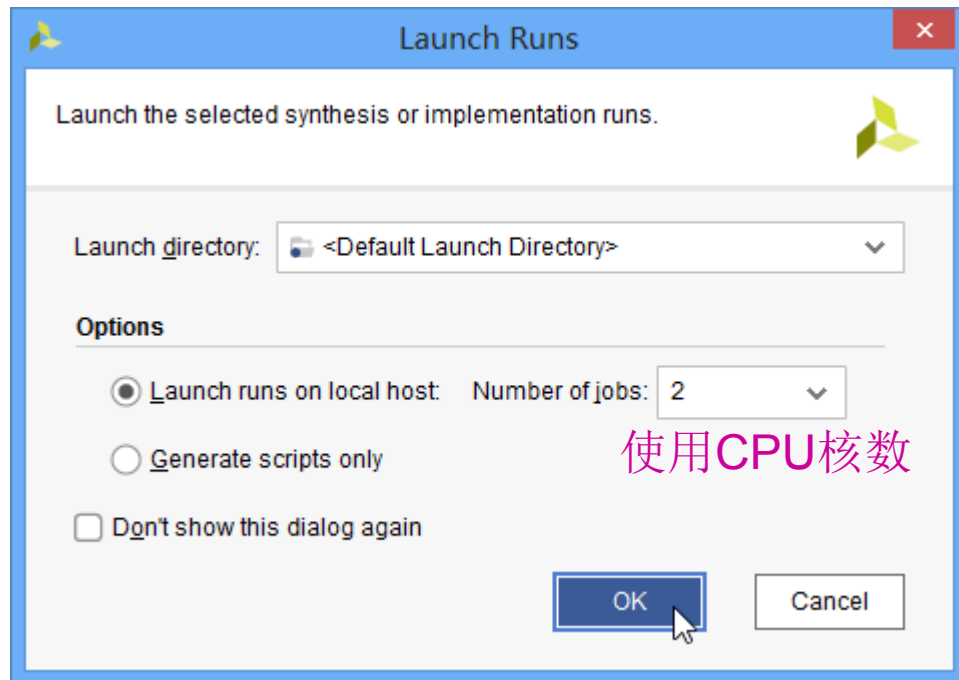
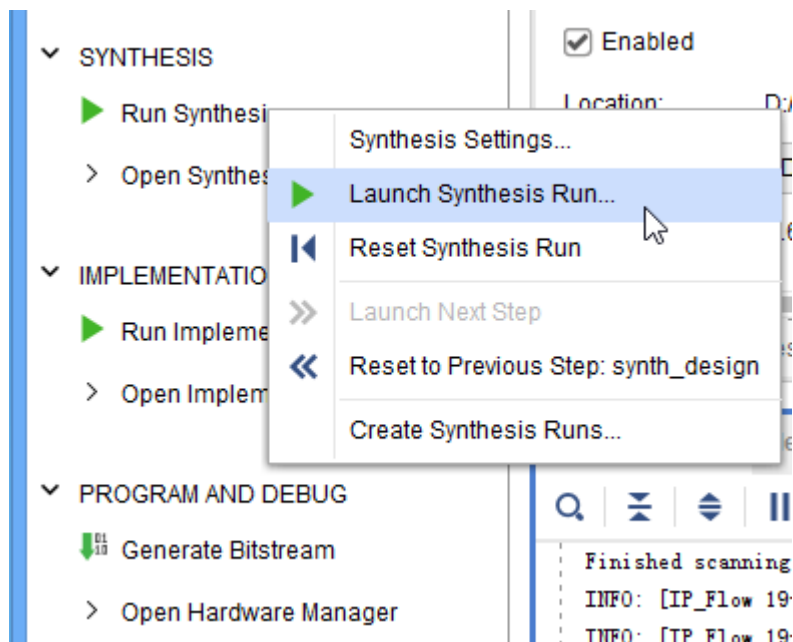


复制两条约束模板到xdc文件，如图格式修改管脚号和电平标准：

XDC -> Physical Constraints -> Placement -> Specific location
-> IO Pin Assignment

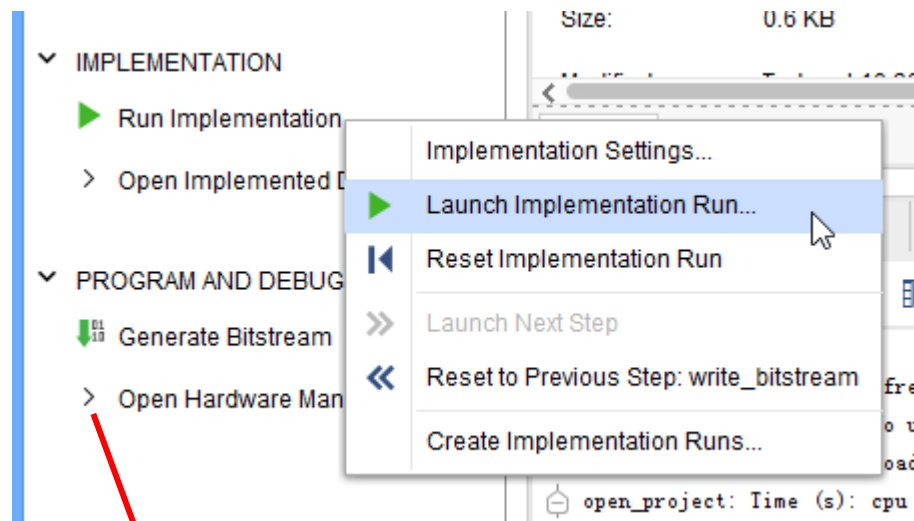
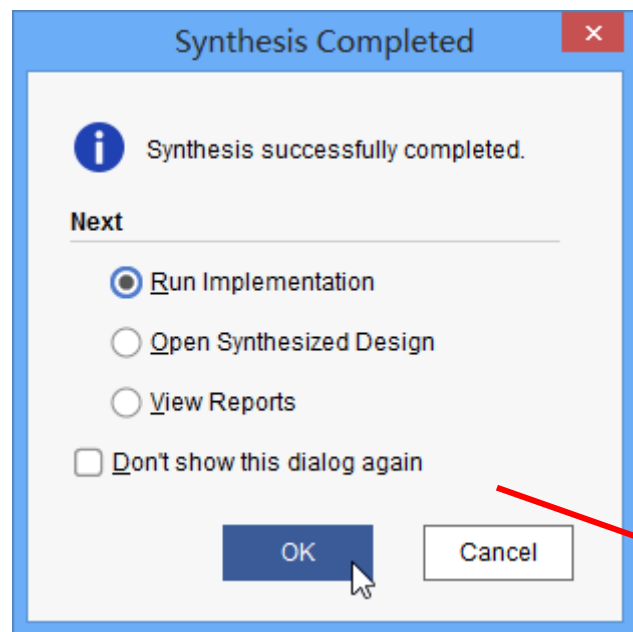
XDC -> Physical Constraints -> IO Constraints -> IOSTANDARD

综合的过程生成网表文件Netlist

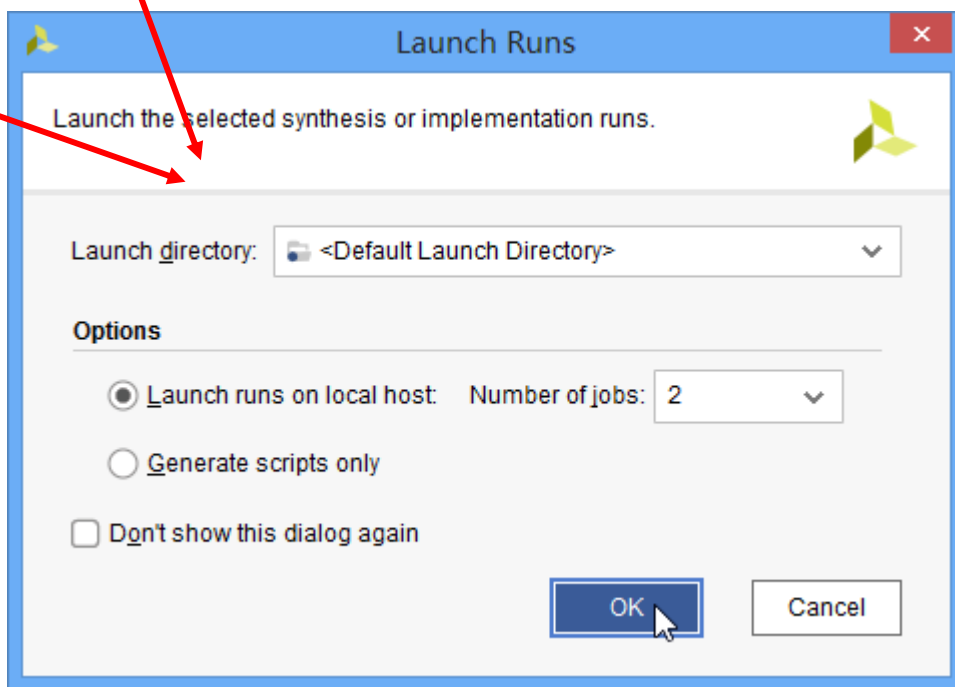


在左侧Flow Navigator的SYNTHESIS里点击Run Synthesis或鼠标右键里的Launch Synthesis Run，弹出右图的对话框，选择在本机运行综合。

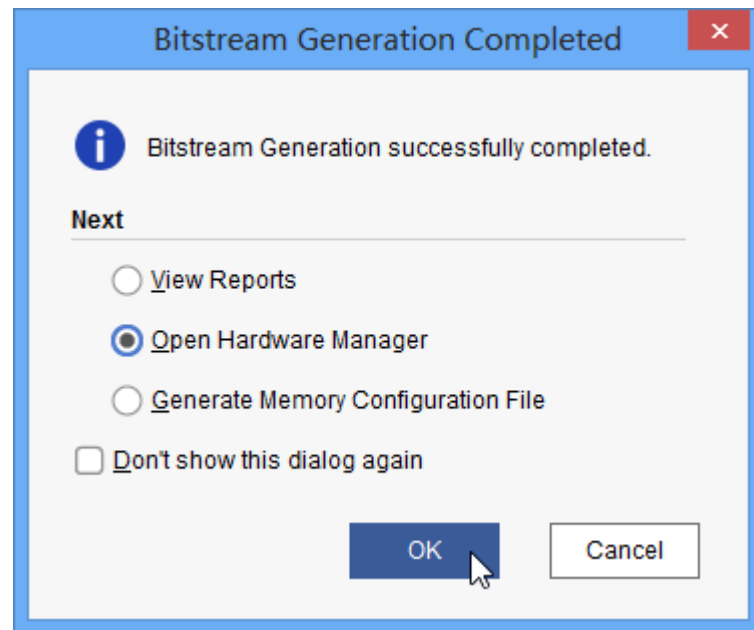
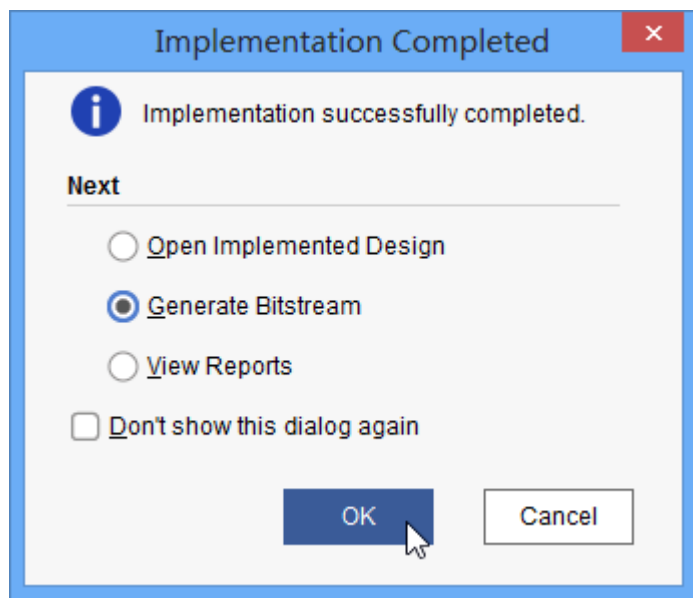
设计的实现Implementation: 将网表转化为FPGA内部的资源, 并进行布局和布线。



Synthesis之后弹出上图菜单, 也可在IMPLEMENTATION里, 左键点击Run Implementation或右键菜单的Launch Implementation Run, 弹出右图的对话框, 选择Launch runs on local host运行。



实现之后可以继续生成Bitstream，用于对FPGA进行编程



实现之后可选择“Open Implemented Design”或者Generate Bitstream之后的对话框里选择“Open Hardware Manager”，都会打开“IMPLEMENTED DESIGN”查看窗口，查看资源使用和布局布线结果。

File Edit Flow Tools Reports Window Layout View Help Q- Quick Access write_bitstream Complete ✓

Flow Navigator Generate Block Design

- SIMULATION
 - Run Simulation
- RTL ANALYSIS
 - Open Elaborated Design
- SYNTHESIS
 - Run Synthesis
 - Open Synthesized Design
- IMPLEMENTATION**
 - Run Implementation
 - Open Implemented Design
 - Constraints Wizard
 - Edit Timing Constraints
 - Report Timing Summary
 - Report Clock Networks
 - Report Clock Interaction
 - Report Methodology
 - Report DRC
 - Report Noise
 - Report Utilization
 - Report Power
 - Schematic
- PROGRAM AND DEBUG
 - Generate Bitstream
 - Open Hardware Manager

IMPLEMENTED DESIGN * - xc7z020clg400-1

Sources Netlist x ? _ □ □

weight

- Nets (10)
- Leaf Cells (7)

I/O Bank Properties ? _ □ □ x

I/O Bank 35

| Name | Available | Prohibit | Ports | I/O Std |
|------|-----------|--------------------------|-------|------------|
| D18 | 1 | <input type="checkbox"/> | ▼ | |
| D19 | 0 | <input type="checkbox"/> | c ▼ | LVC MOS33* |
| D20 | 0 | <input type="checkbox"/> | b ▼ | LVC MOS33* |
| E18 | 1 | <input type="checkbox"/> | ▼ | |
| E19 | 1 | <input type="checkbox"/> | ▼ | |
| F16 | 1 | <input type="checkbox"/> | ▼ | |
| F17 | 1 | <input type="checkbox"/> | ▼ | |
| M19 | 1 | <input type="checkbox"/> | ▼ | |

general Properties Port Summary Package Pins

Project Summary x Device x weight.v x weight.xdc x ? _ □ □

使用的逻辑资源 →

使用的IOB资源

Tcl Console Messages Log Reports Design Runs Power DRC Timing x ? _ □ □

Design Timing Summary

General Information

Timer Settings

Design Timing Summary

Check Timing (0)

User Ignored Paths

Unconstrained Paths

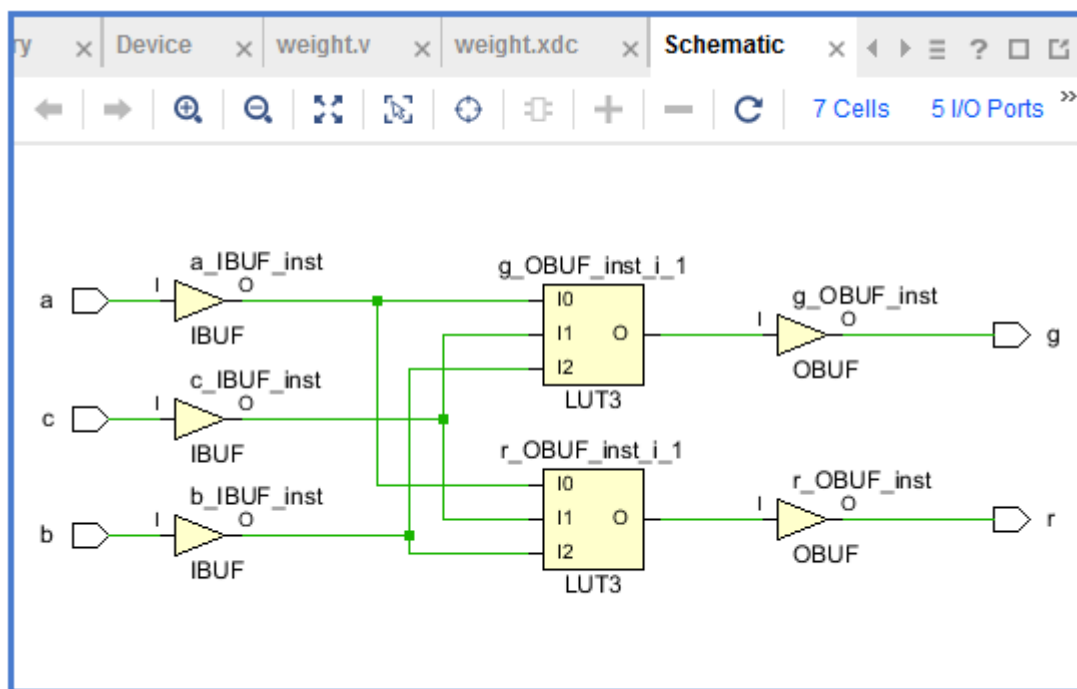
| Setup | Hold | Pulse Width |
|---------------------------------|---------------------------------|-----------------------------------------|
| Worst Negative Slack (WNS): NA | Worst Hold Slack (WHS): NA | Worst Pulse Width Slack (WPWS): |
| Total Negative Slack (TNS): NA | Total Hold Slack (THS): NA | Total Pulse Width Negative Slack (TPWS) |
| Number of Failing Endpoints: NA | Number of Failing Endpoints: NA | Number of Failing Endpoints: |
| Total Number of Endpoints: NA | Total Number of Endpoints: NA | Total Number of Endpoints: |

There are no user specified timing constraints.

Timing Summary - impl_1 (saved)

查看RTL Schematic

选择IMPLEMENTATION -> Open Implemented Design -> Schematic, 可以查看原理图。可见用了3个输入缓冲器、2个输出缓冲器, 以及用于实现组合逻辑的2个查找表 (LUT: Lookup Table)。



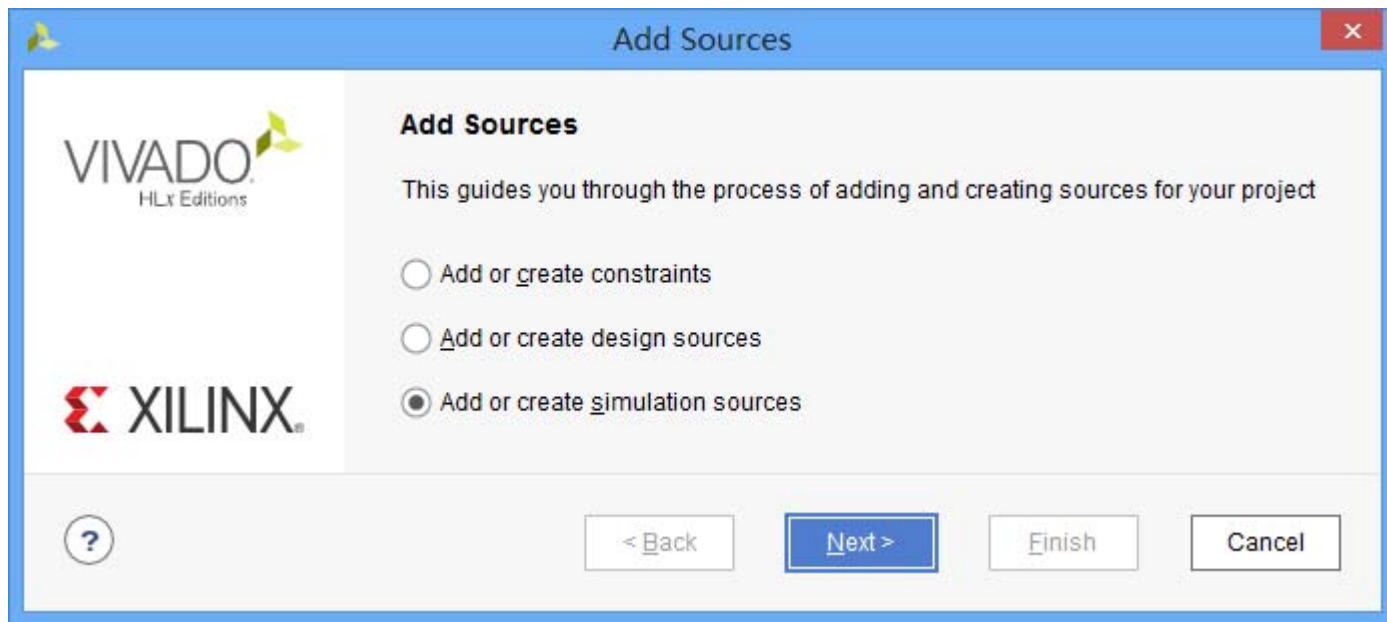
鼠标左键点击选中某个单元后, 再右键点击单元, 可以通过右键菜单里查看其关联的源码, 查看关联的单元、导出原理图等。

4.设计的仿真

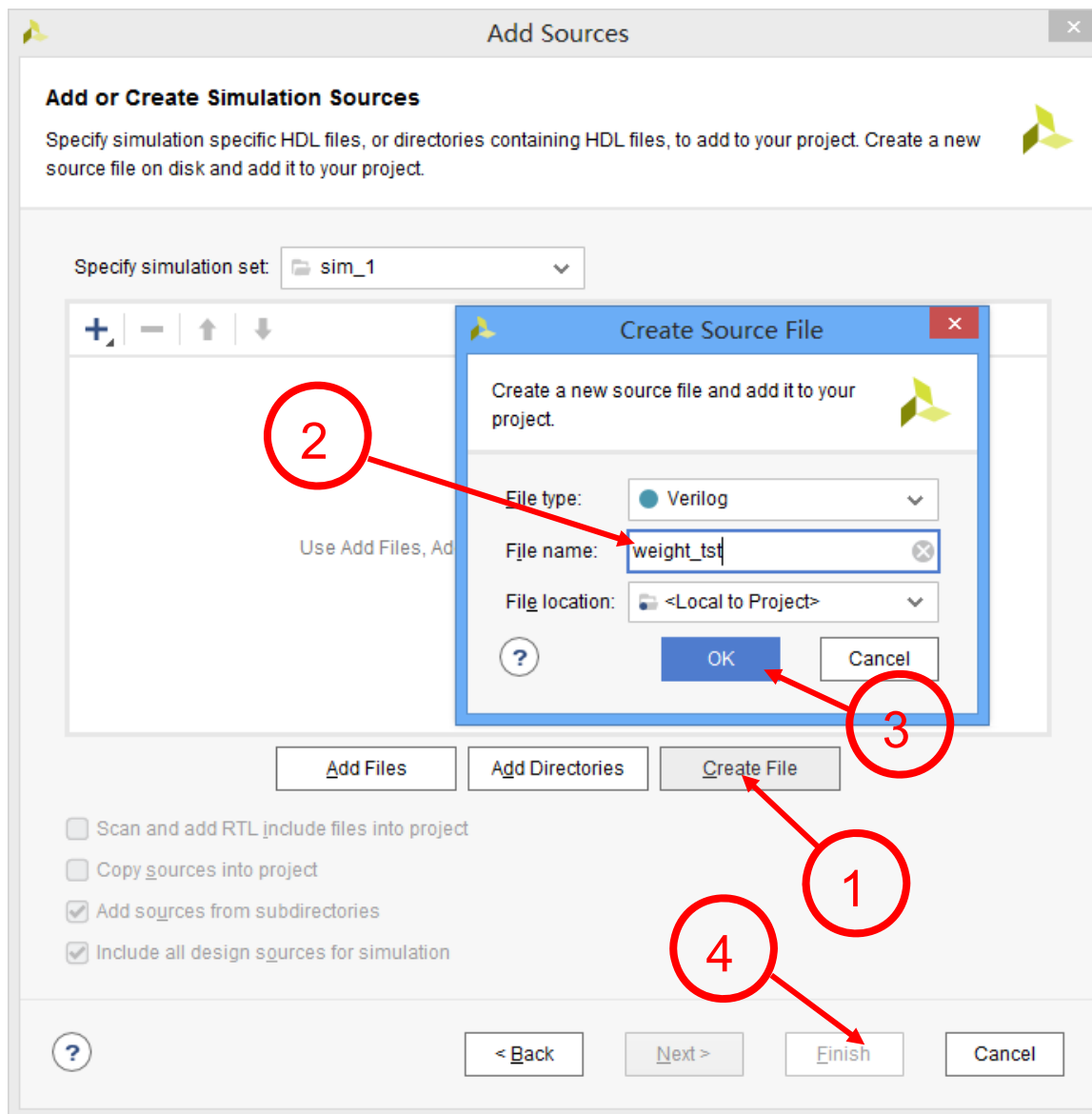
- 逻辑电路的仿真
 - 使用波形输入作为激励信号
低版本**ISE**支持绘制波形作为输入，
高版本**ISE Design Suite**已废除波形输入功能，
Vivado也没有波形输入功能；
 - 使用**HDL**语言编写测试向量文件进行仿真。
测试向量：**Test bench /test fixture**

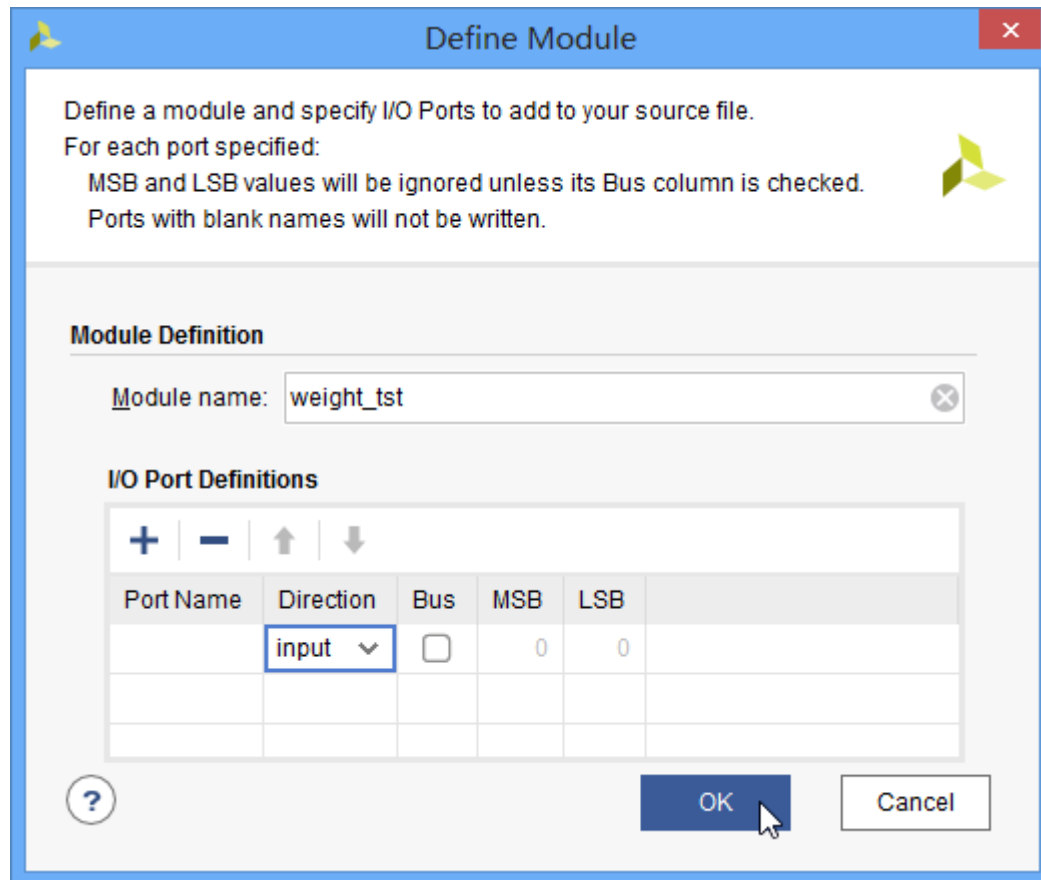
添加仿真用的测试向量文件

Vivado左侧Flow Navigator栏里，点击Add Source，选择添加仿真源码的选项：Add or create simulation sources



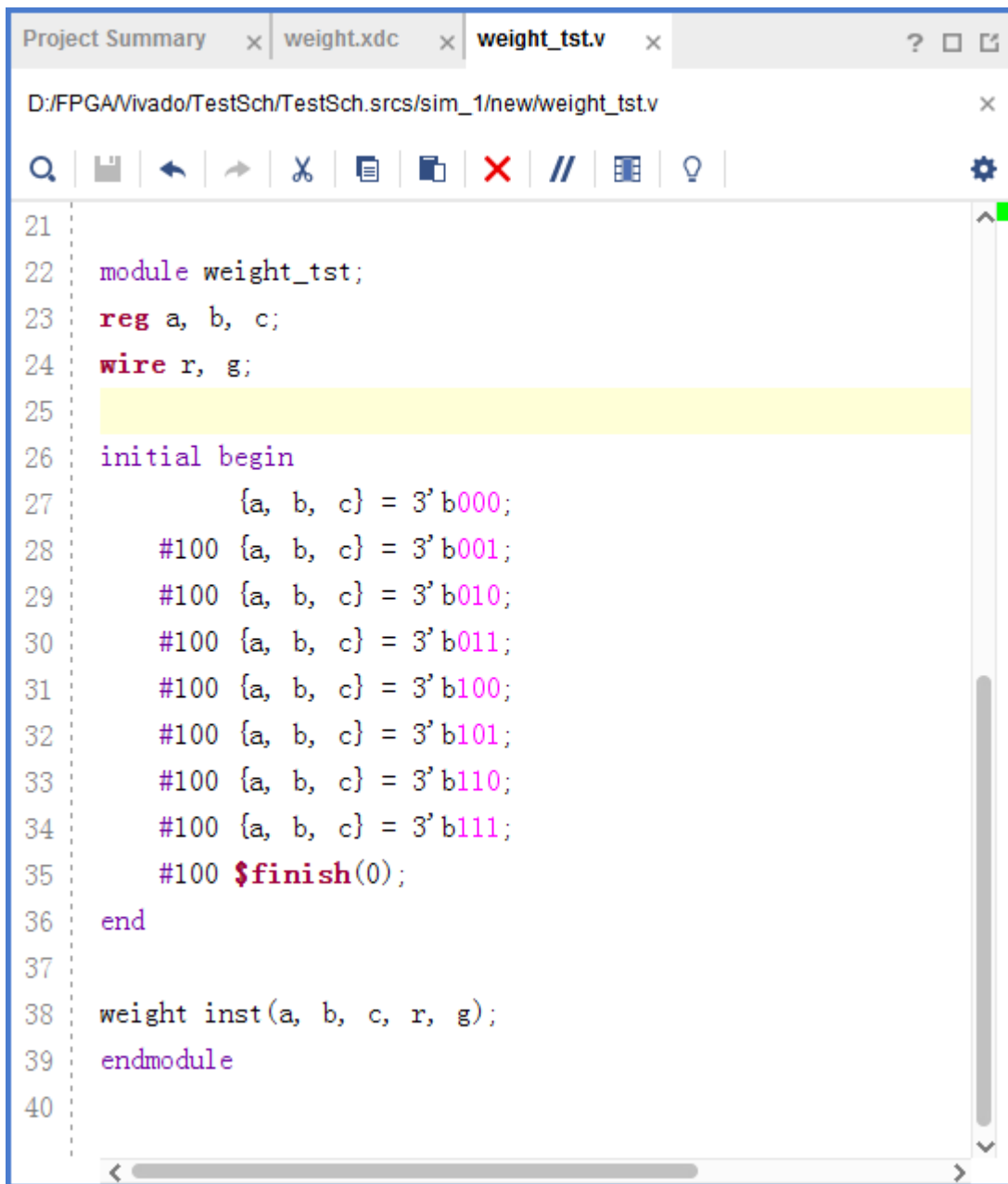
新建测试向量文件



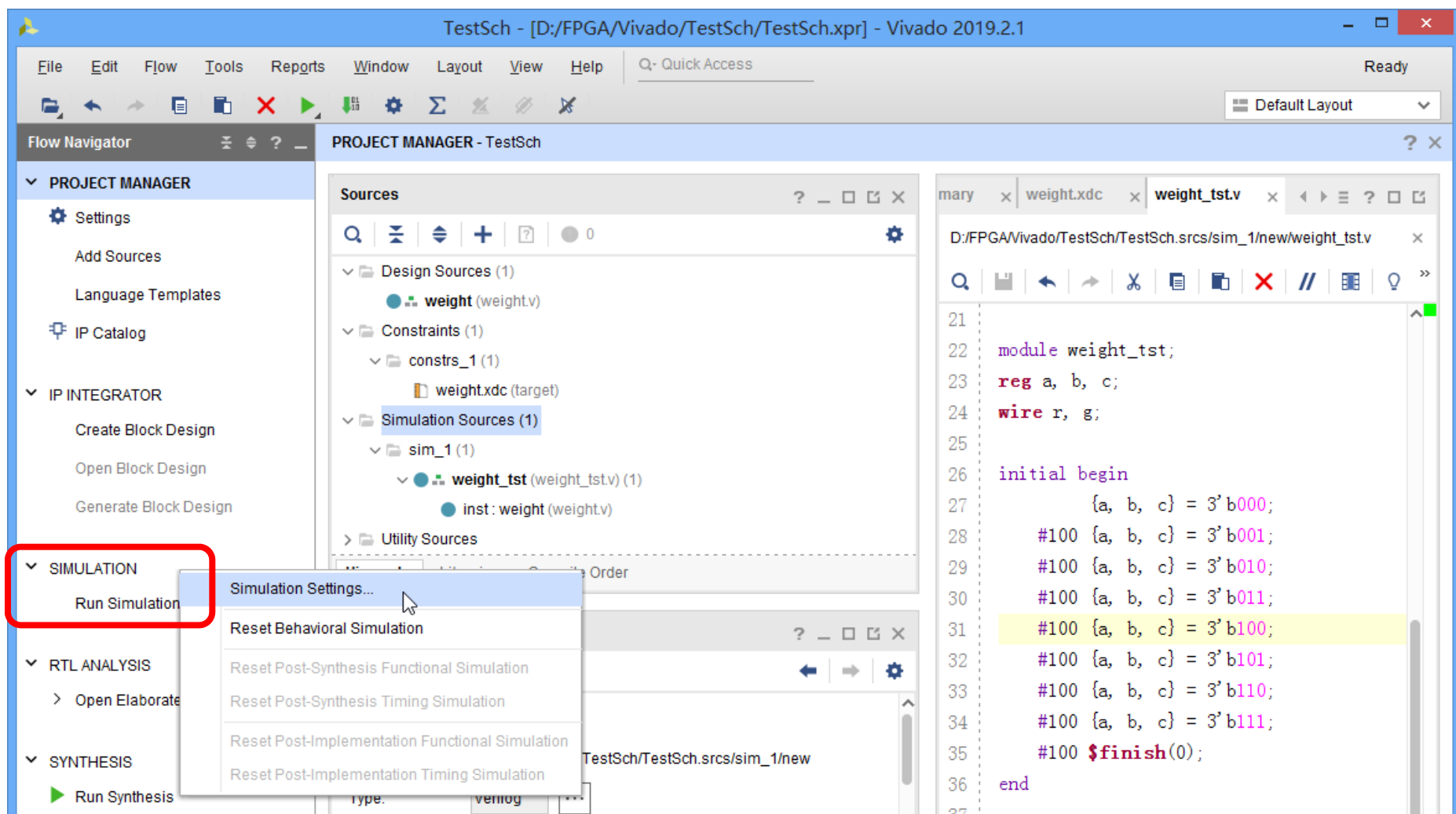


顶层测试向量无外部端口，此对话框不添加IO端口。

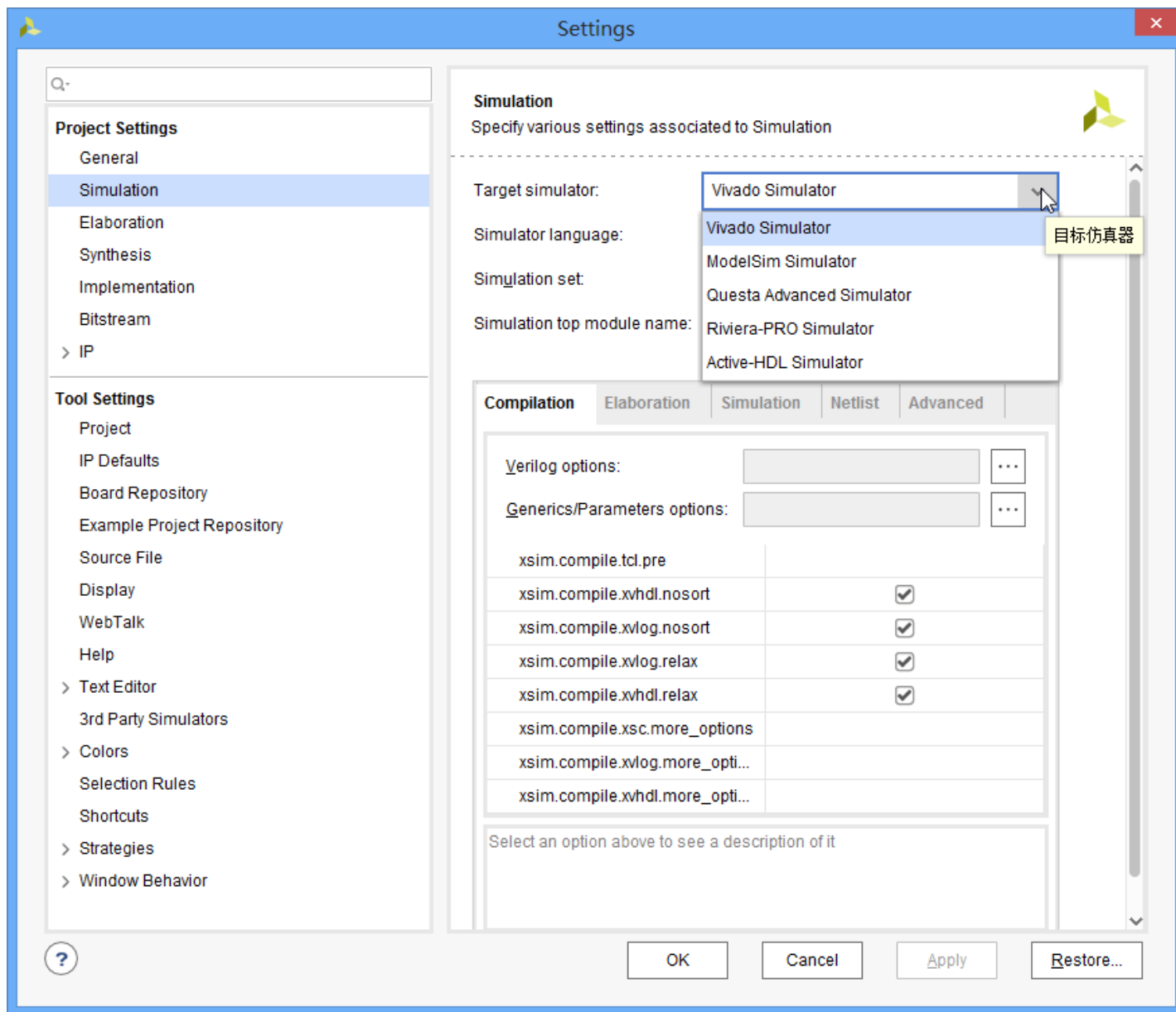
在测试向量文件里添加如图所示的代码



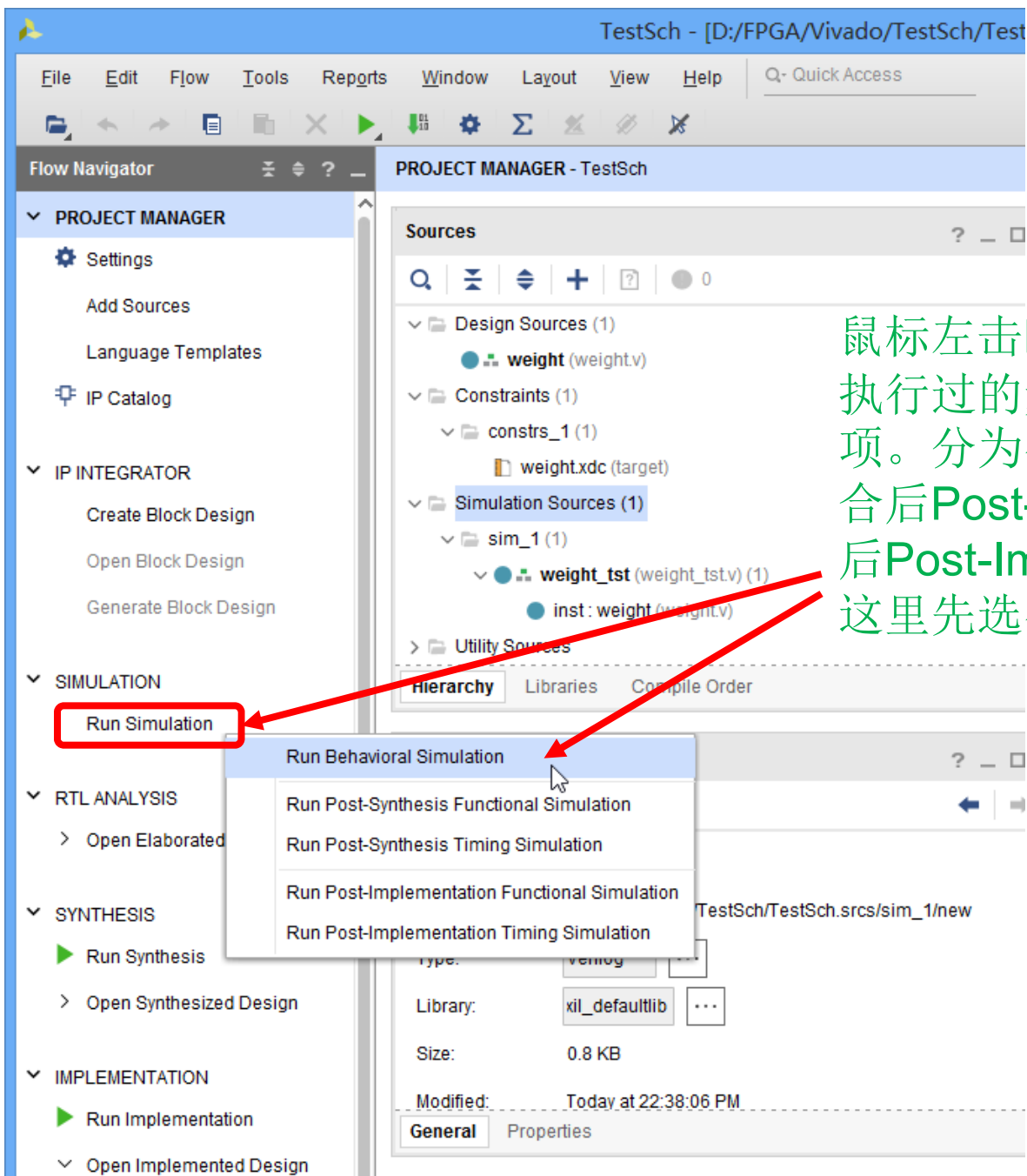
```
Project Summary x weight.xdc x weight_tst.v x ? □ ▢
D:/FPGA/Vivado/TestSch/TestSch.srscs/sim_1/new/weight_tst.v x
🔍 📁 ⬅ ➡ ✂ 📄 📋 ✖ // 📊 💡 ⚙
21
22 module weight_tst;
23 reg a, b, c;
24 wire r, g;
25
26 initial begin
27     {a, b, c} = 3'b000;
28     #100 {a, b, c} = 3'b001;
29     #100 {a, b, c} = 3'b010;
30     #100 {a, b, c} = 3'b011;
31     #100 {a, b, c} = 3'b100;
32     #100 {a, b, c} = 3'b101;
33     #100 {a, b, c} = 3'b110;
34     #100 {a, b, c} = 3'b111;
35     #100 $finish(0);
36 end
37
38 weight inst(a, b, c, r, g);
39 endmodule
40
```



鼠标右键单击SIMULATION或Run Simulation，点选Simulation Settings...



Target Simulator里可选Xilinx的Vivado Simulator; 如果安装了Mentor公司（被Simense收购）的ModelSim仿真器，也可以选。这里选第一项。



鼠标左击Run Simulation，根据执行过的步骤，会出现不同的选项。分为行为级Behavioral、综合后Post-Synthesis、布局布线后Post-Implementation三个级别。这里先选行为级仿真。

Vivado仿真器界面（行为仿真）

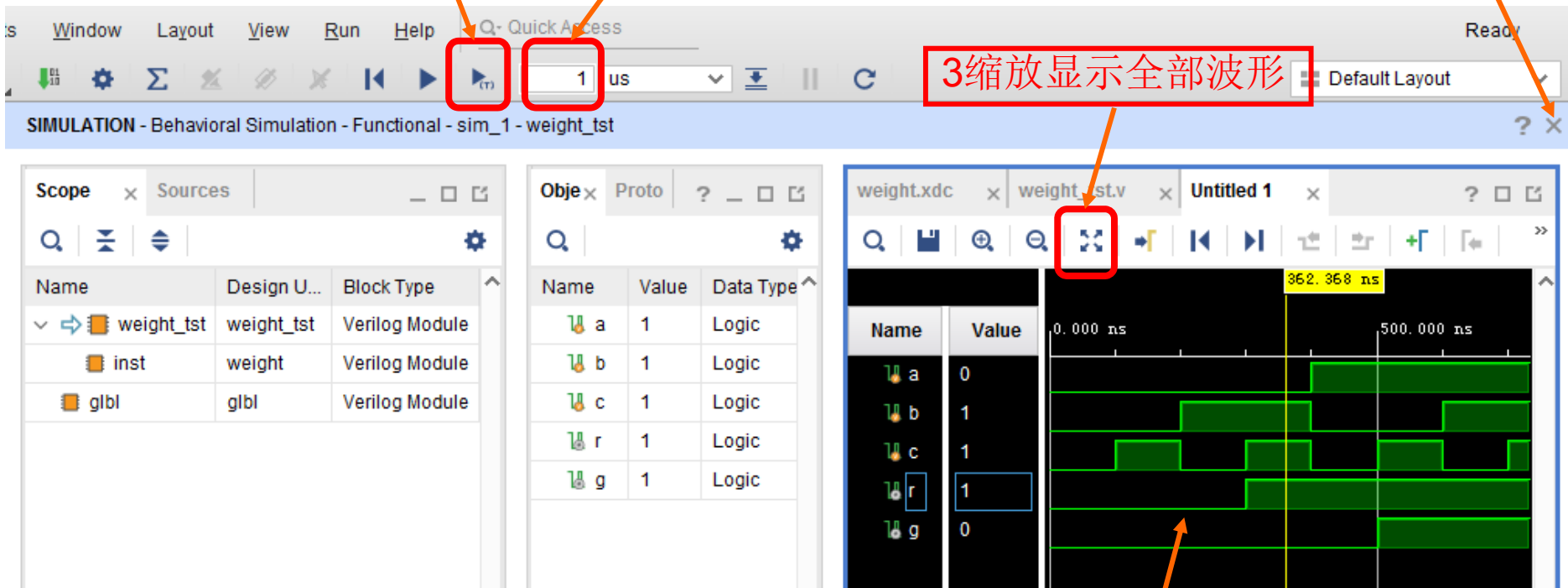
2运行指定时长

1设定运行时长

5退出仿真状态

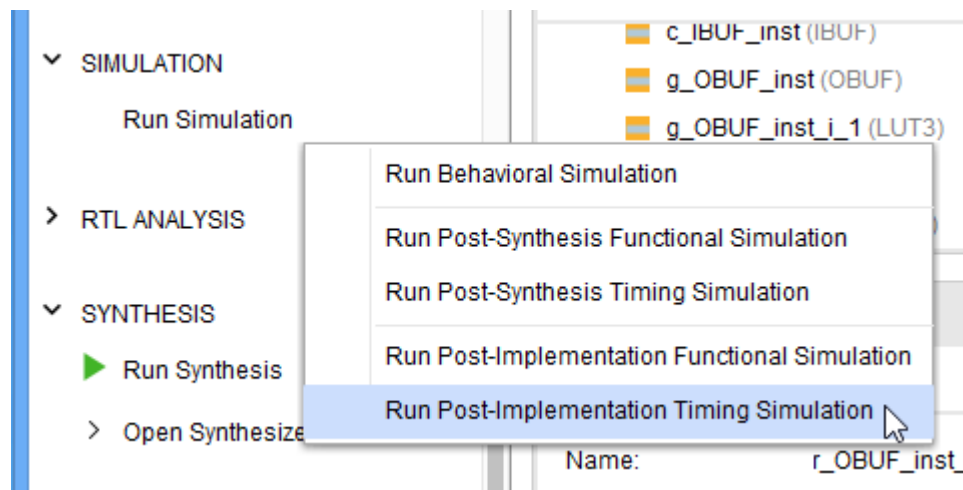
3缩放显示全部波形

4查看仿真结果

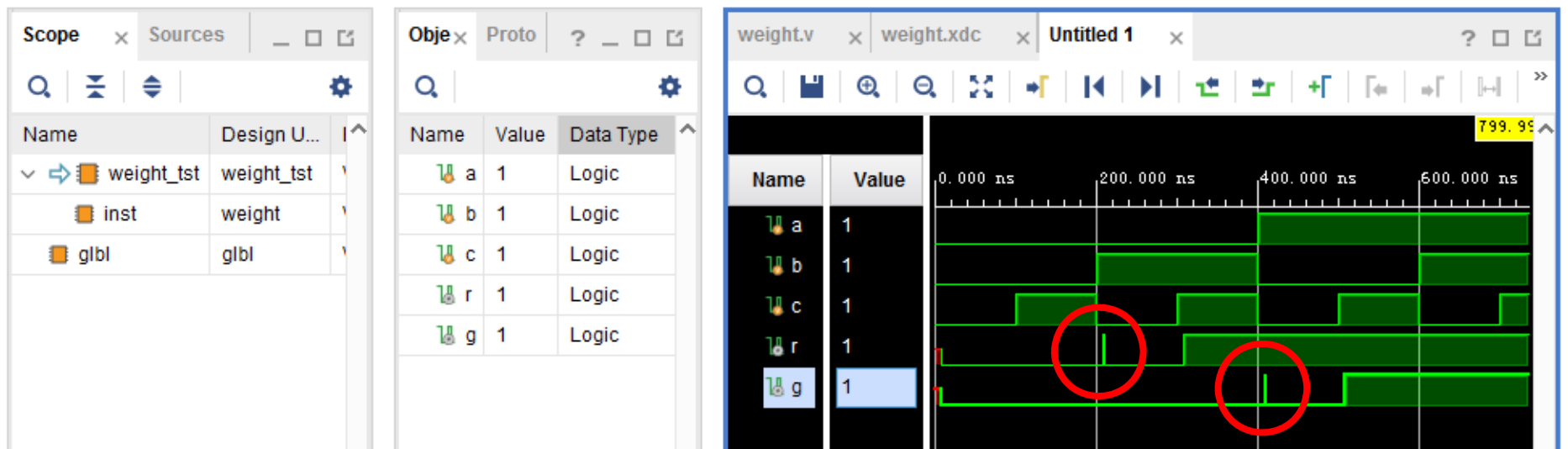


可见逻辑正确。Behavior仿真，输入信号改变到结果输出，没有逻辑电路延迟

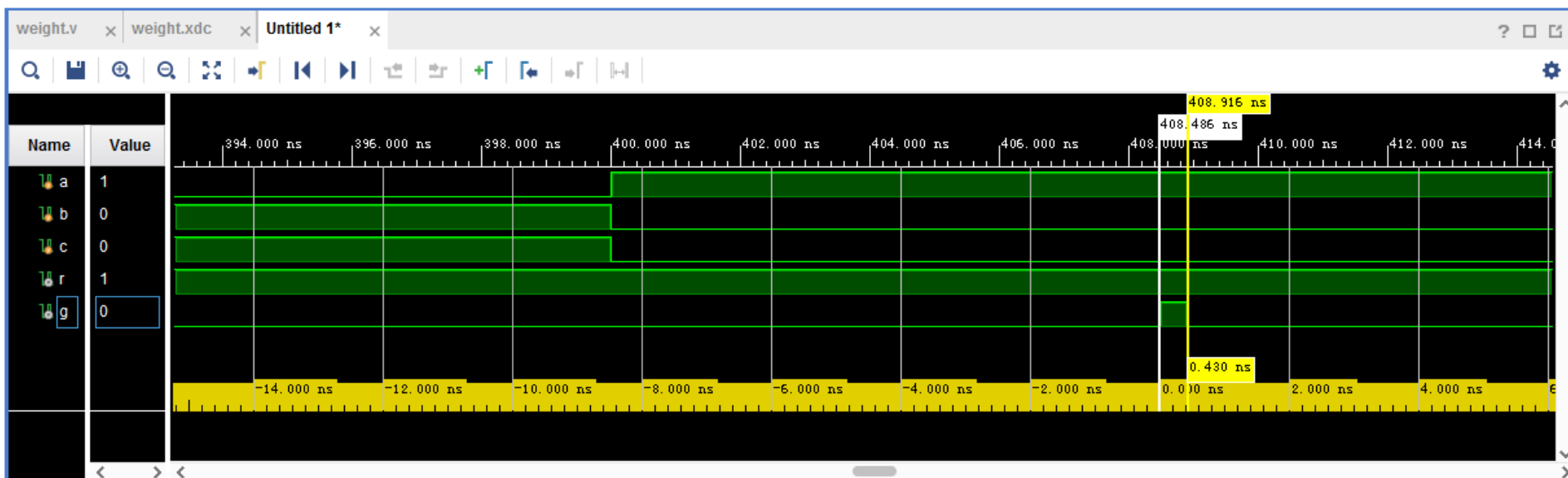
Vivado进行时序仿真



前面已经完成了布局布线，此处可以用鼠标左键点击Flow Navigator里SIMULATION下的Run Simulation，选择Run Post-Implementation Timing Simulation进行布局布线后时序仿真。

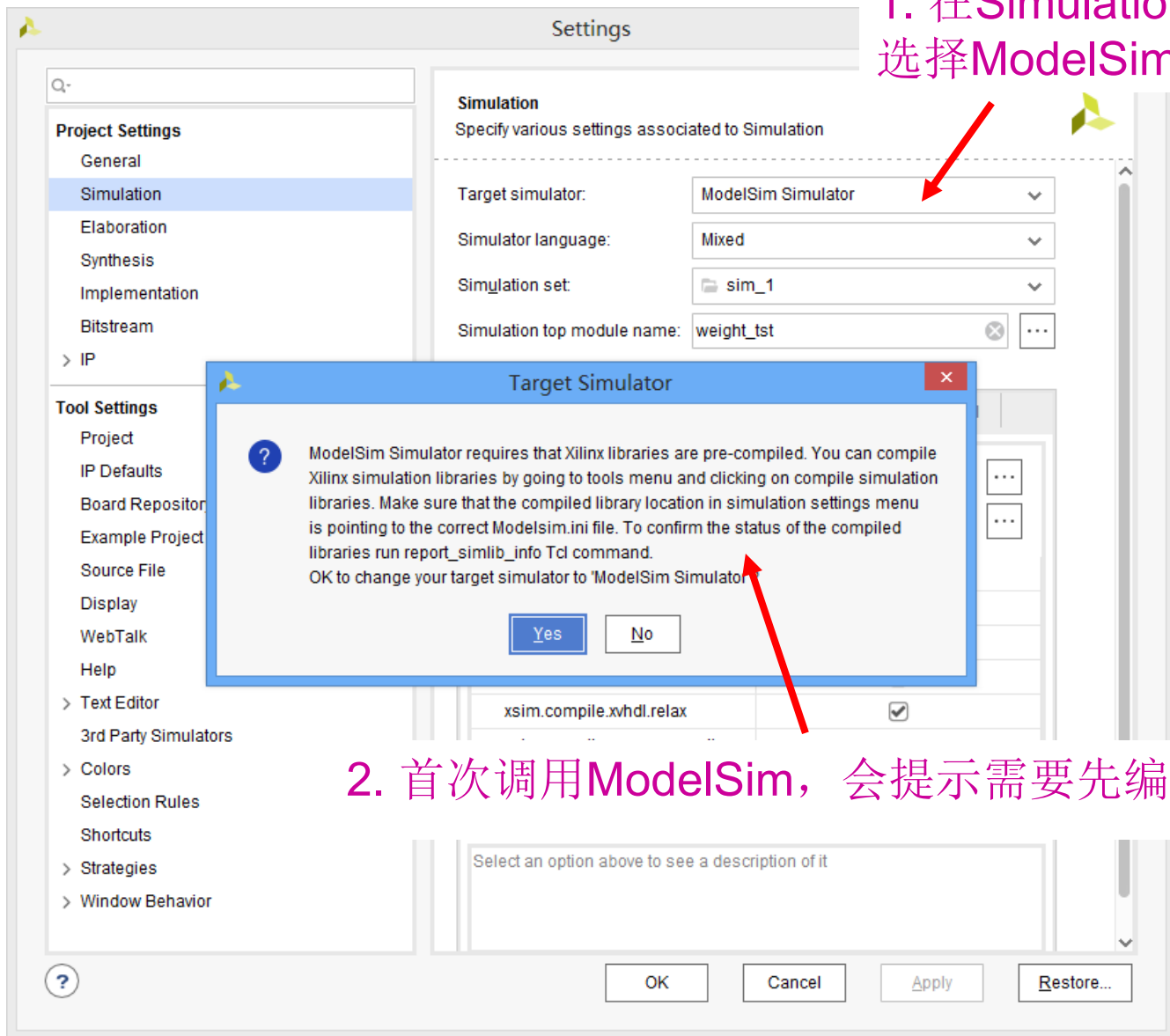


可见逻辑输出与输入变化存在延迟，并且存在门延迟造成的毛刺。



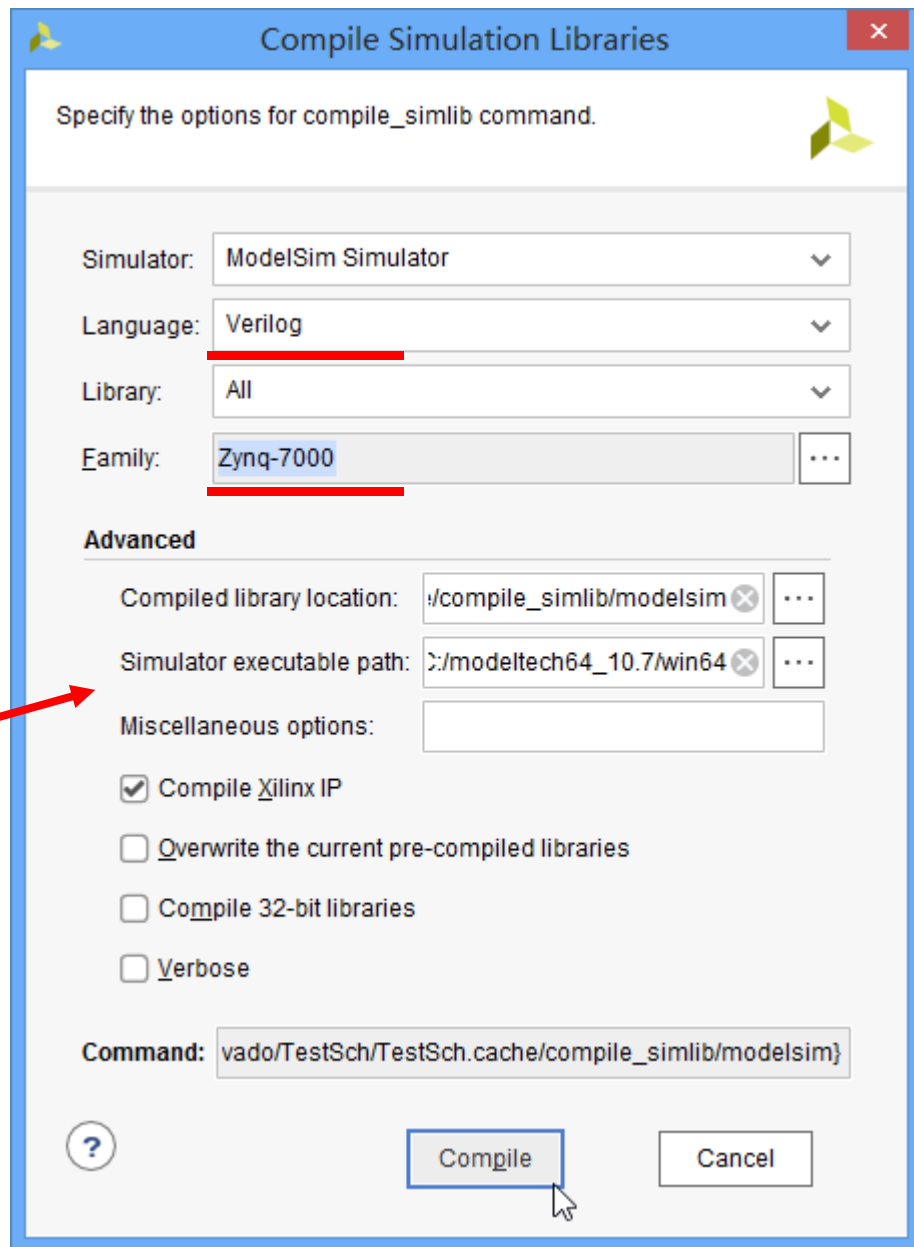
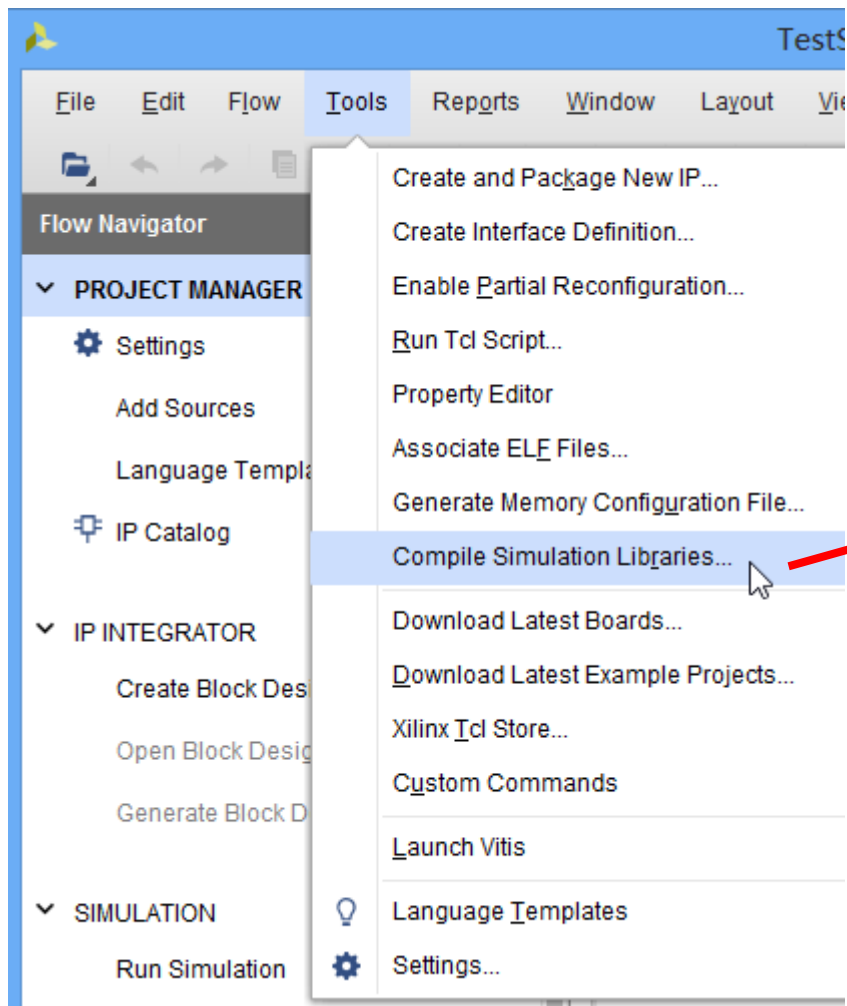
Vivado下调用ModelSim仿真

1. 在Simulation属性里选择ModelSim仿真器



2. 首次调用ModelSim，会提示需要先编译仿真库

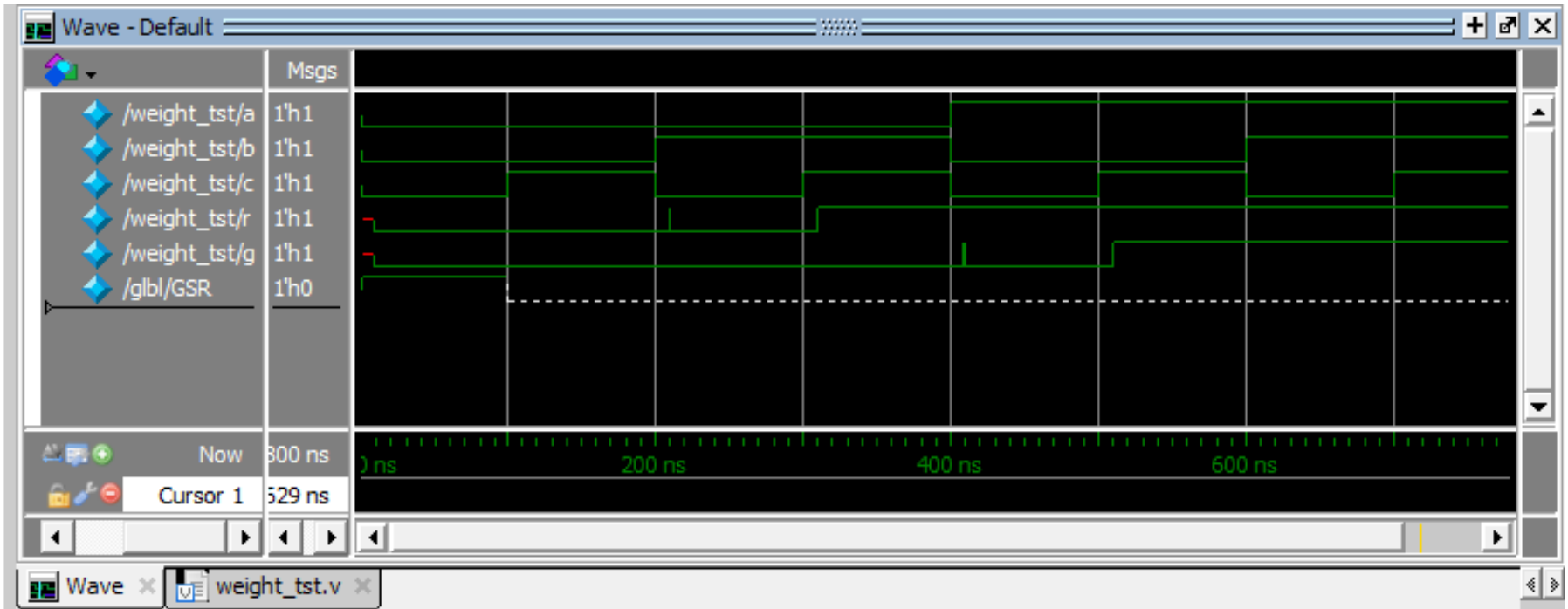
ModelSim编译仿真库



在**PROJECT MANAGER**状态，Tools -> Compile Simulation Libraries菜单。
为减少编译时间，仅选择Verilog语言、Zynq-7000系列器件，约需半小时。

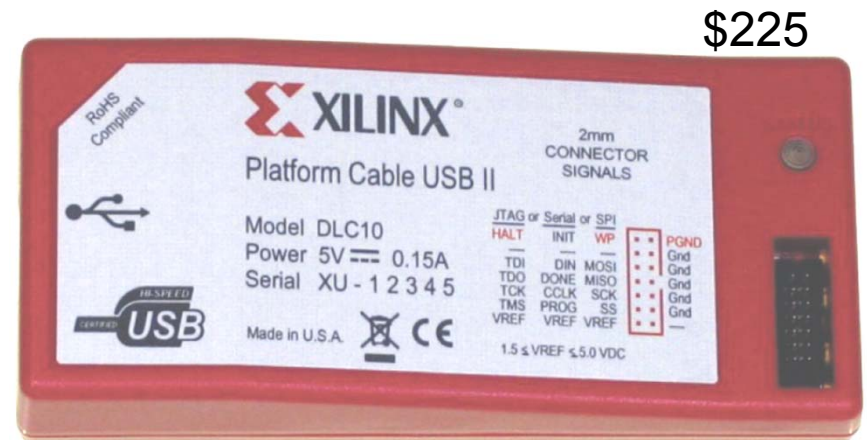
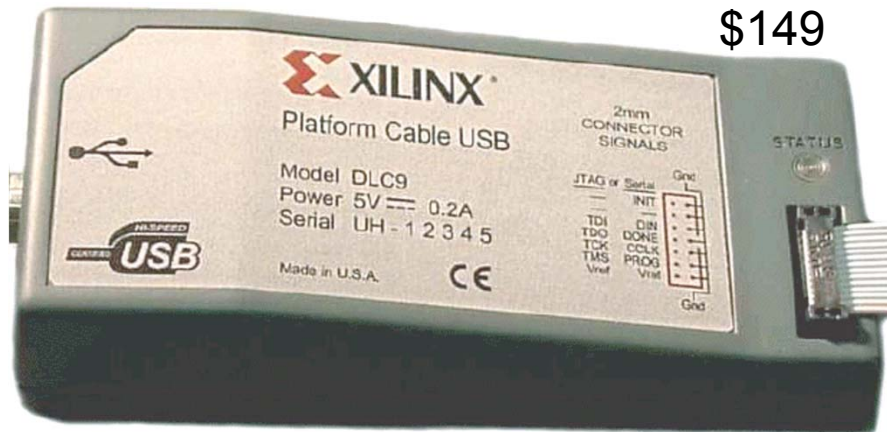
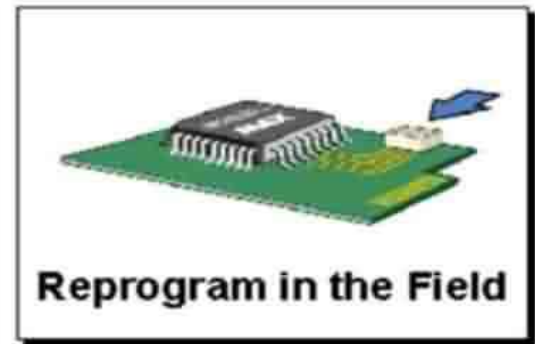
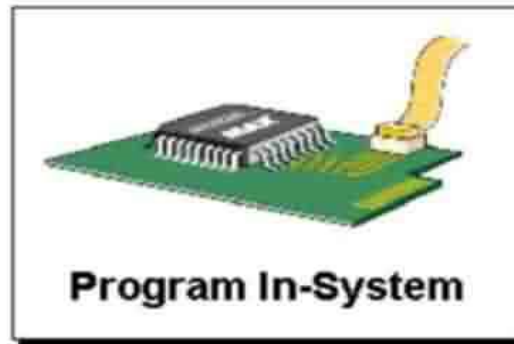
启动ModelSim仿真器

启动仿真器方法同前：鼠标点击Flow Navigator栏的SIMULATION -> Run Simulation，弹出菜单选Run Post-Implementation Timing Simulation



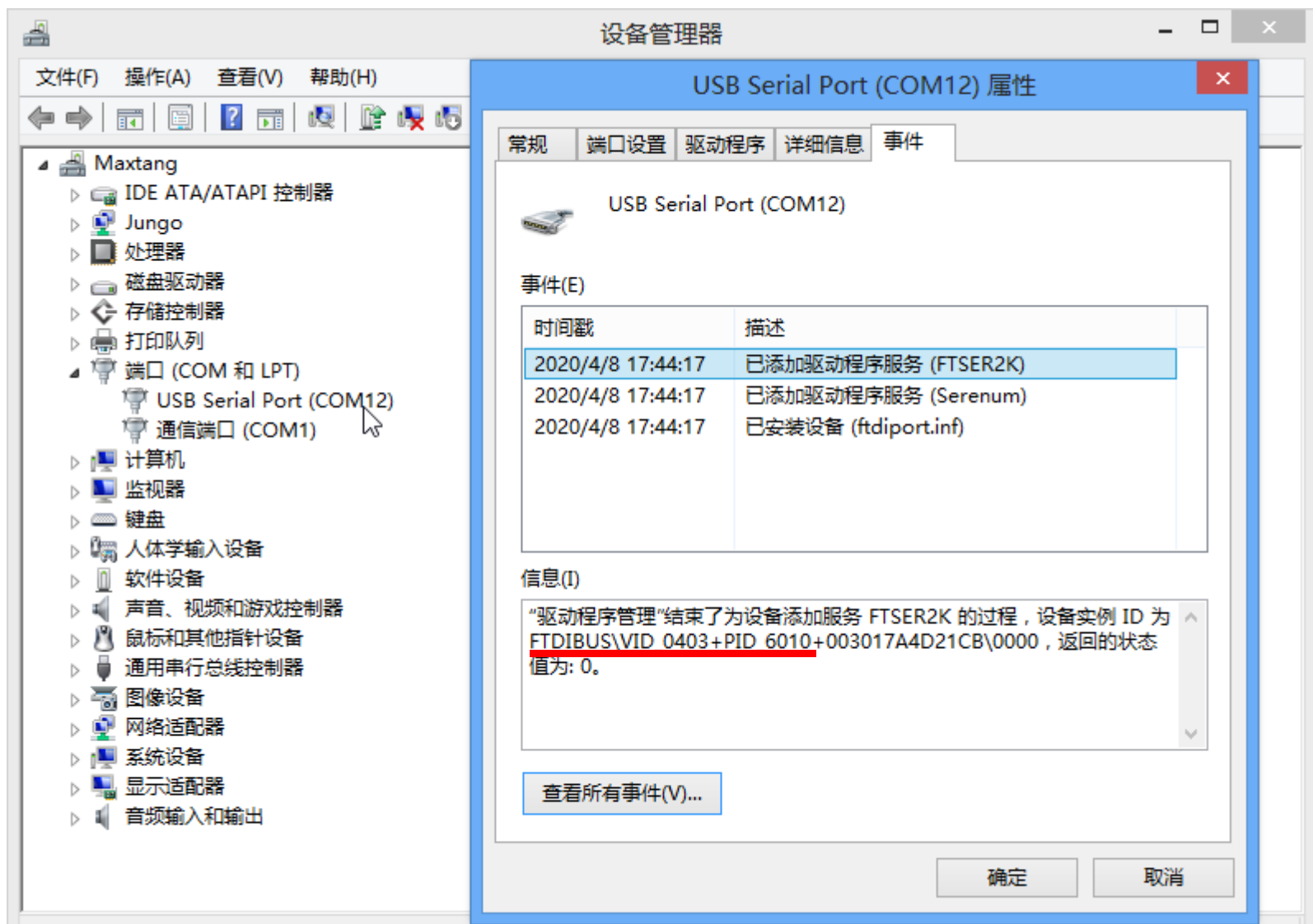
ModelSim是优选的仿真器，功能丰富强大，仿真速度远高于Xilinx软件自带仿真器，只不过该软件的价格非常昂贵。

5.配置FPGA（下载）



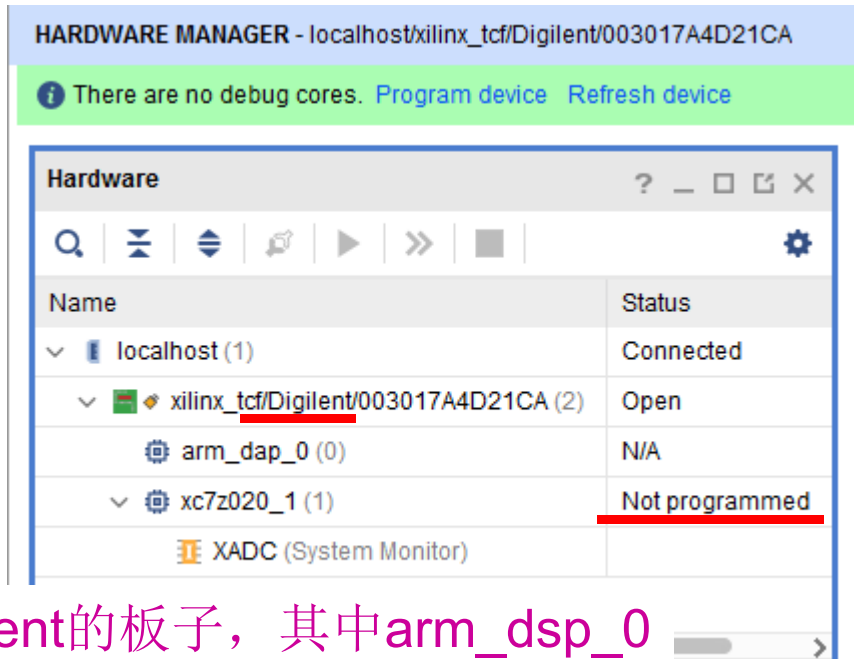
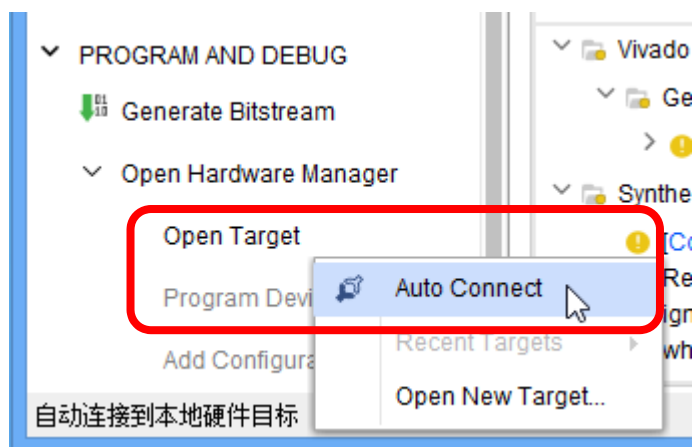
Xilinx USB下载电缆支持Xilinx全系列器件（PROM、CPLD、FPGA），经JTAG接口（Boudry Scan Interface: IEEE1149.1/IEEE1532）下载和调试程序，还支持Slave Serial和SPI模式配置器件。

Xilinx的评估板一般集成某种下载器，不用另外采购专用编程器。其中PYNQ-Z1/Z2板上集成了由FTDI公司FT2232HQ组成的USB-UART桥接器。

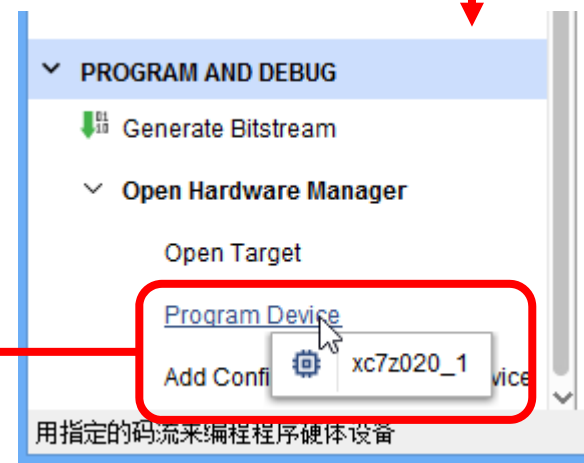
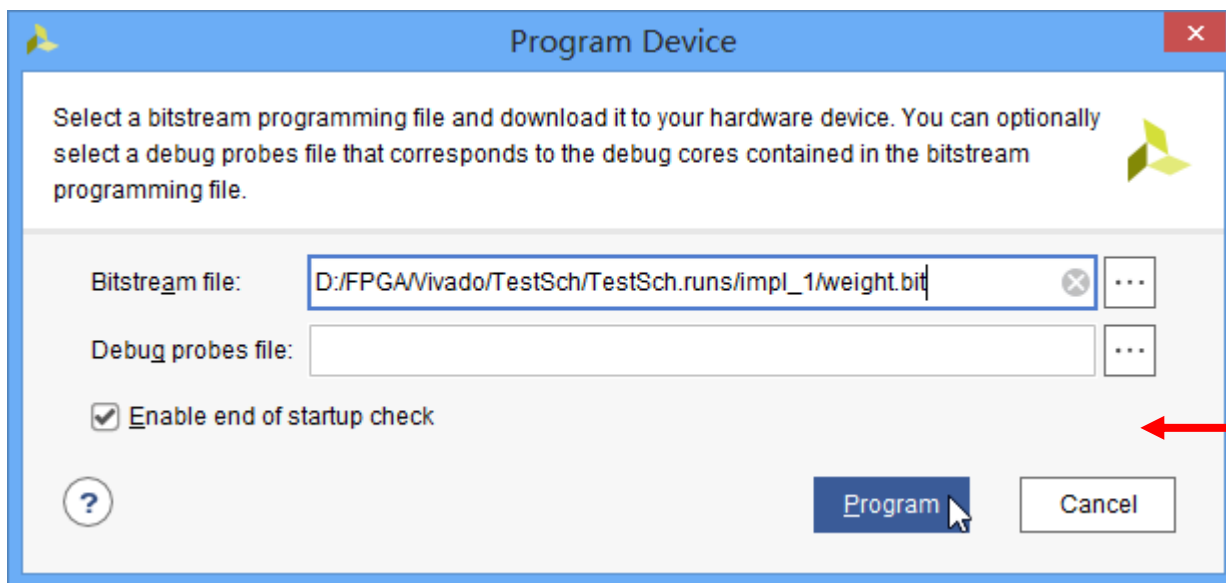


用一条USB-A转USB-MicroB的数据线（注意还有一种只能充电的线，里面没有数据线，不行）把板子J14连接在计算机上，打开板子电源开关，驱动程序会自动安装。如果打开Windows设备管理器，可以看到多出一个COM口，说明驱动程序自动安装完成。其属性的事件信息里有FTDI字样。

向FPGA下载bitstream编程数据



启动Hardware Manager，可以找到Digilent的板子，其中arm_dsp_0是PS部分，xc7z020_1是PL部分，bitstream是PL部分的配置数据。



6. 在板验证功能

HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/003017A4D21CA

There are no debug cores. Program device Refresh device

| Name | Status |
|---------------------------------------|------------|
| localhost (1) | Connected |
| xilinx_tcf/Digilent/003017A4D21CA (2) | Open |
| arm_dap_0 (0) | N/A |
| xc7z020_1 (1) | Programmed |
| XADC (System Monitor) | |

FPGA的PL配置完毕

Hardware Device Properties

xc7z020_1

Name: xc7z020_1

Part: xc7z020

ID code: 23727093

IR length: 6

Status: Programmed

Programming file: TestSch/TestSch.runs/impl_1/weight

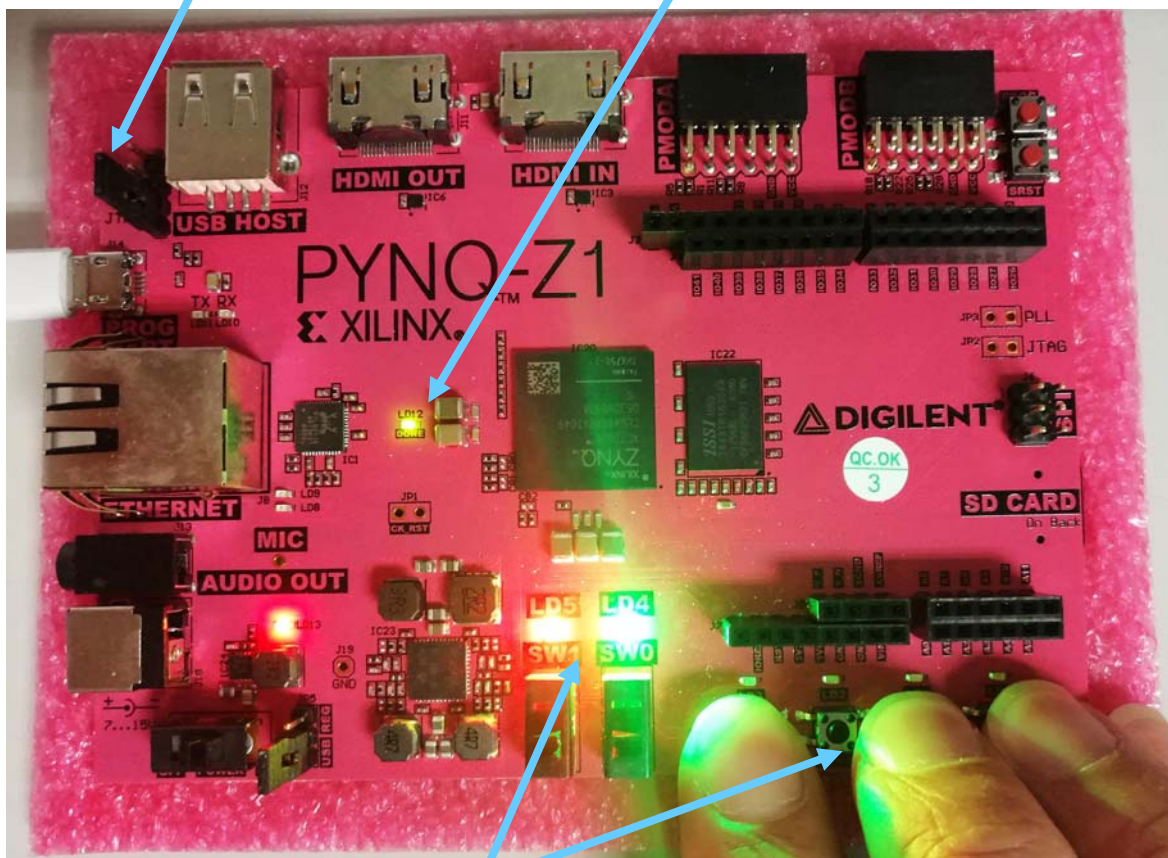
Probes file:

User chain count: 4

General Properties

上电前JP4的跳线
要插在JTAG位置

下载后“Done”灯
LD12点亮



r、g灯能根据按键动作正确亮灭

小结

- 以上简单演示了Vivado软件的设计和仿真基本流程，还有很多设计方法问题以及开发工具中影响设计效率和性能的选项需要在更深入的应用中仔细体会。
- PYNQ-Zx板上只有2个拨动开关（Switch），拨动后可以保持电平。举重裁判例程需要3个输入，因此用了3个按键（Button），按键释放恢复初始电平而不能保持。本例是一个组合逻辑电路，所以松手后r、g灯就灭了。
- **思考**：如何设计逻辑，使按下按键生效、记住按键事件，松手也能保持亮灯？
- **提示**：需要用到时序逻辑电路，三个裁判按键以及一个复位按键。