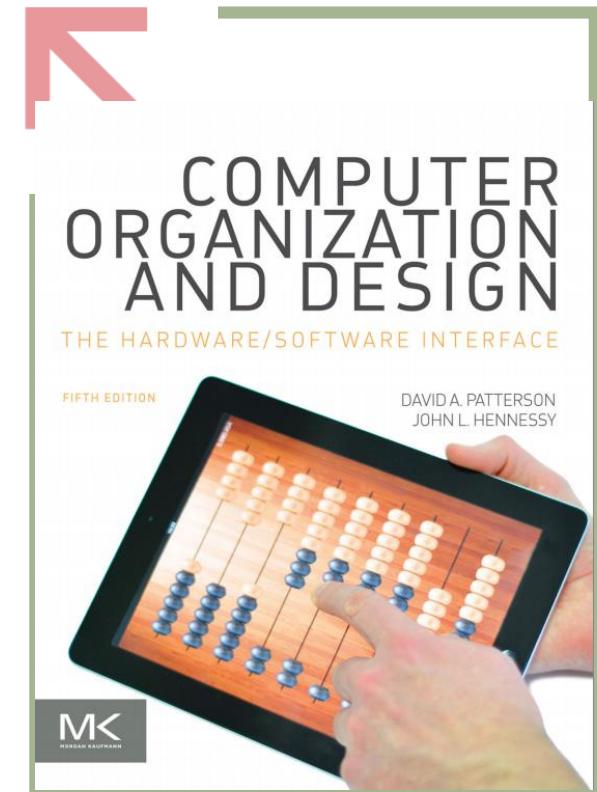


Vivadao and Minisys



硬件实验所需安装的软件：vivado 2015.4

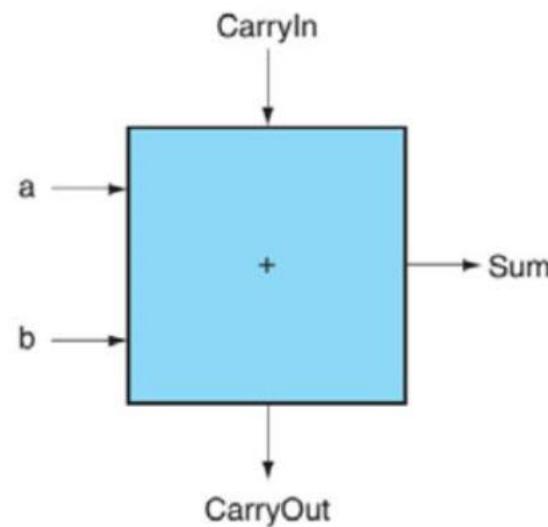
链接：

<https://pan.baidu.com/s/1MfeMCK2igcsf1WEQA49ObA>

密码： fhr4

安装时候选择第三项(System Design)，密钥文件在sakai
文件夹的Lab目录下，安装完后copy license导入密钥文件
激活

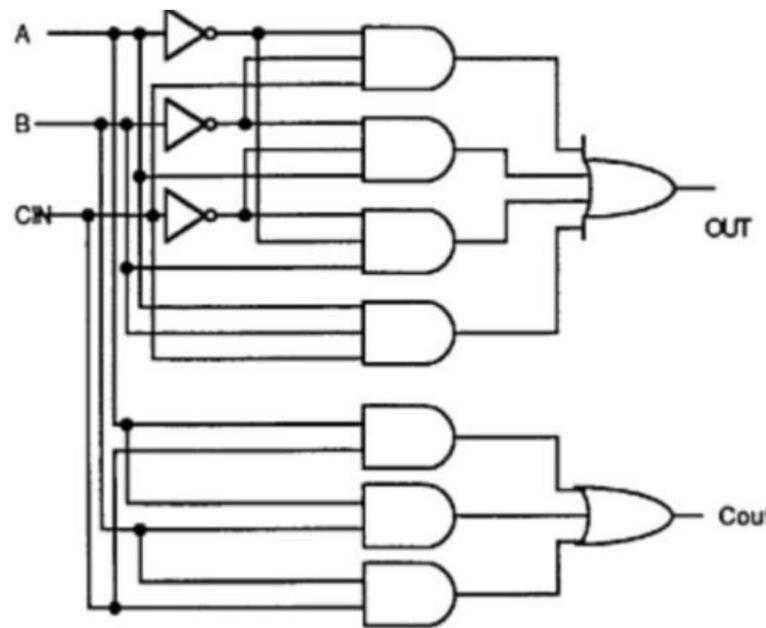
从one-bit-adder开始...



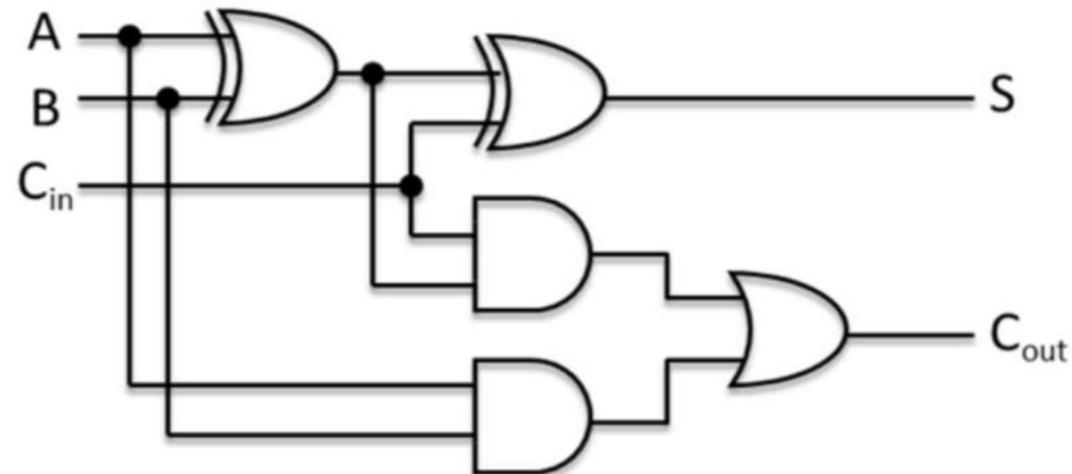
Inputs			Outputs	
a	b	CarryIn	CarryOut	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$\text{Sum} = (a \cdot \bar{b} \cdot \overline{\text{CarryIn}}) + (\bar{a} \cdot b \cdot \overline{\text{CarryIn}}) + (\bar{a} \cdot \bar{b} \cdot \text{CarryIn}) + (a \cdot b \cdot \text{CarryIn})$$

$$\text{CarryOut} = (b \cdot \text{CarryIn}) + (a \cdot \text{CarryIn}) + (a \cdot b)$$



1-bit adder – version 1



1-bit adder – version 2

Verilog 简介

硬件描述语言 (Hardware Description Language, HDL)

电子系统硬件

1. 行为描述
 2. 结构描述
 3. 数据流描述
- 的一种语言

数字电路系统的设计者通过这种语言：

1. 可以从上层到下层，从抽象到具体，逐层次地描述自己的设计思想
2. 用一系列分层次的模块来表示极其复杂的数字系统
3. 利用模块组合经由自动综合工具转换到门级电路网表
4. 用自动布局布线工具把网表转换为具体电路进行布局布线
5. 下载到专用集成电路(ASIC)或现场可编程逻辑器件

Verilog HDL与C语言的比较

C语言	Verilog HDL
sub-function	module, function, task
if-then-else	if-then-else
case	case
{,}	begin, end
for	for
while	while
int	int
printf	monitor, display

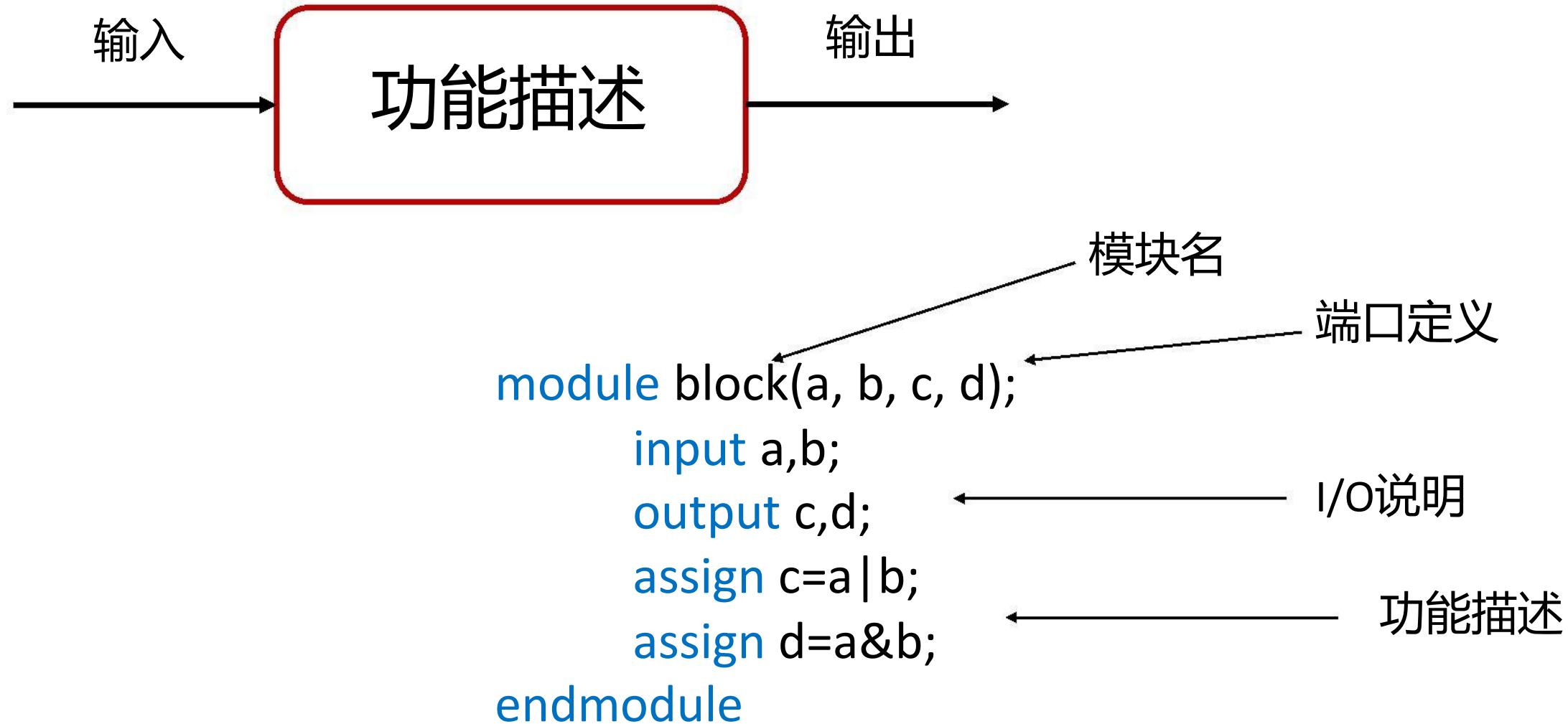
运算符比较

C语言	Verilog HDL	功能
*	*	乘
/	/	除
+	+	加
-	-	减
!	!	逻辑非
&&	&&	逻辑与
		逻辑或
>>	>>	逻辑左移
<<	<<	逻辑右移
?:	?:	条件运算符

运算符比较

C语言	Verilog HDL	功能
<code>~</code>	<code>~</code>	按位非
<code>&</code>	<code>&</code>	按位与
<code> </code>	<code> </code>	按位或
<code>^</code>	<code>^</code>	按位异或
<code>~^</code>	<code>~^</code>	按位异或非
<code>%</code>	<code>%</code>	取余
<code>></code>	<code>></code>	大于
<code><</code>	<code><</code>	小于
<code>>=</code>	<code>>=</code>	大于等于
<code><=</code>	<code><=</code>	小于等于
<code>==</code>	<code>==</code>	等于
<code>!=</code>	<code>!=</code>	不等于

Verilog 基本结构



模块的端口定义

模块端口声明了模块的端口，格式如下：

`module 模块名(端口1, 端口2, 端口3, ...);`

也可以写成：

`module 模块名(端口1,
 端口2,
 端口3,
 ...);`

端口和端口之间用**逗号**隔开，结尾处有**分号**。

模块内容



I/O说明格式 输入: `input [位宽-1:0]` 端口1, 端口2, ...;

也可以写成: 输入: `input [位宽-1:0]` 端口1;
 `input [位宽-1:0]` 端口2; **推荐分开写**

输出: `output [位宽-1:0]` 端口1, 端口2, ...;

模块内容

内部变量说明

在模块中与端口有关的变量，有reg和wire类型。其声明格式如下：

`reg [位宽-1:0] 变量1, 变量2, ...;`

`wire [位宽-1:0] 变量1, 变量2, ...;`

也可以写成：

`reg [位宽-1:0] 变量1;`

`reg [位宽-1:0] 变量2;`

推荐分开写

`wire [位宽-1:0] 变量1;`

`wire [位宽-1:0] 变量2;`

模块内容

功能定义

- assign 可以描述组合逻辑
- always 既可以描述组合逻辑也可以描述时序逻辑

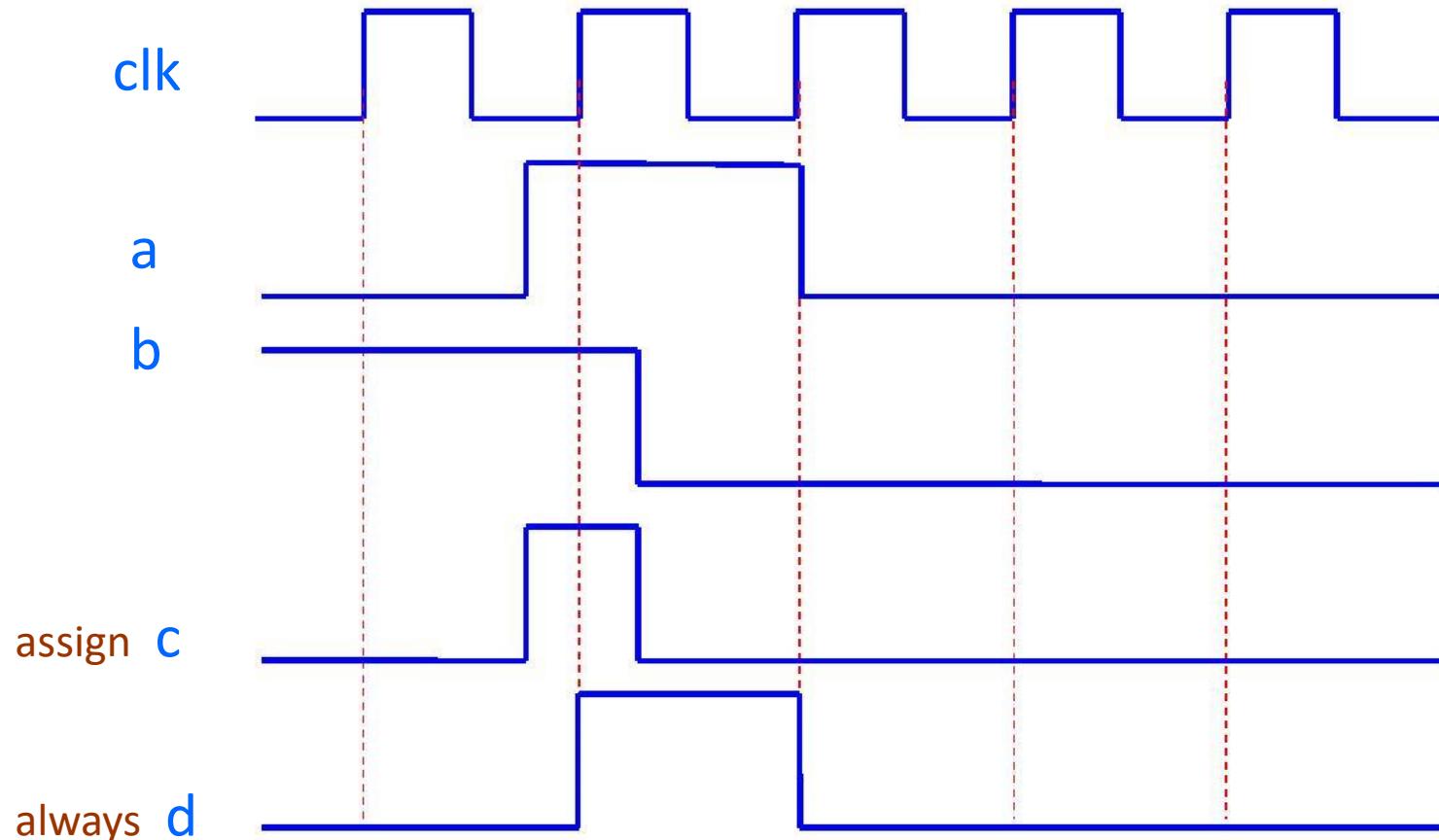
简单来说，组合逻辑就是和时间没关系，
时序逻辑与时间有关系

assign c = a & b;

always@ (a or b)
c = a&b;

always@ (posedge clk)
d <= a&b;

always和assign的区别



VHDL 数据类型

Verilog HDL中,数据主要有以下四种进制形式:

- 1)二进制整数(b或B)
- 2)十进制整数(d或D)
- 3)十六进制整数(h或H)
- 4)八进制整数(o或O)

变量

① 线型 (wire型)

wire型数据常用来表示用于以assign关键字指定的组合逻辑信号。
Verilog程序模块中输入/输出信号类型缺省时自动定义为wire型。

② 寄存器型 (reg型)

寄存器是数据储存单元的抽象， reg是寄存器数据类型的关键字。
reg型数据常用来表示用于“always”模块内的指定信号。

在“always”块内被赋值的每一个信号都必须定义成reg型。

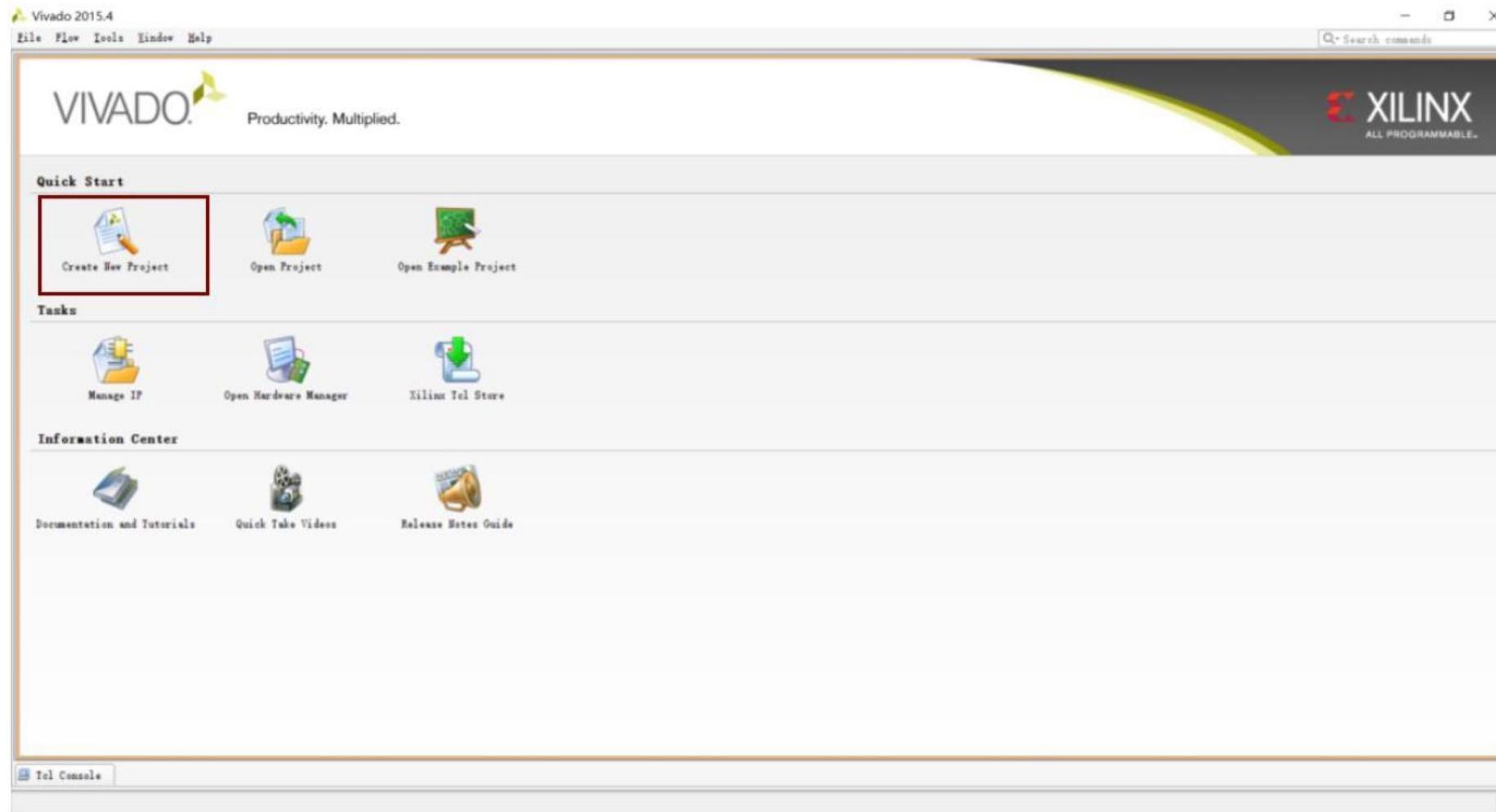
verilog 示例

1位比较器的Verilog HDL程序。通过对输入信号A，B的比较，把比较的结果反映到m,L,e端口。

具体程序如下：

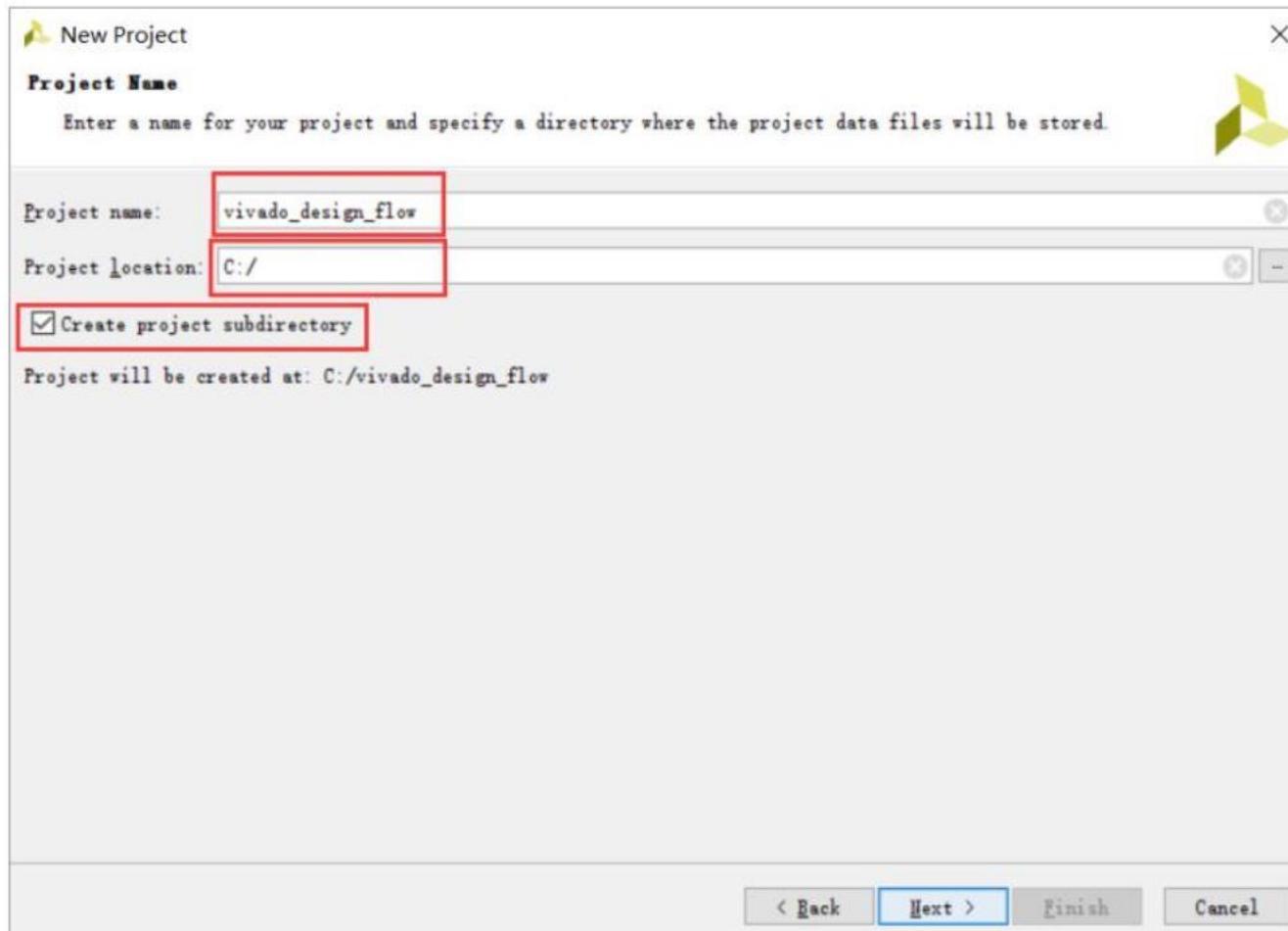
```
module comparator(A,B,m,L,e); //comparator 是模块名称， A,B,m,L,e 是端口
    input A,B; //端口类型， A、 B 为输入信号名
    output m,L,e; //端口类型， m、 L、 e 为输出信号名
    reg m,L,e; //定义内部变量 m、 L、 e
    //行为描述
    always@(A or B) //触发条件，当 A、 B 的电平发生变化时，执行以下语句
        if(A>B) //逻辑功能描述，如果 A>B 成立， m 端口输出为“1”
            {m,L,e}=3'b100;
        else if(A<B) //如果 A<B 成立， L 端口输出为“1”
            {m,L,e}=3'b010;
        else //如果 A=B 成立， e 端口输出为“1”
            {m,L,e}=3'b001;
endmodule
```

使用vivado仿真

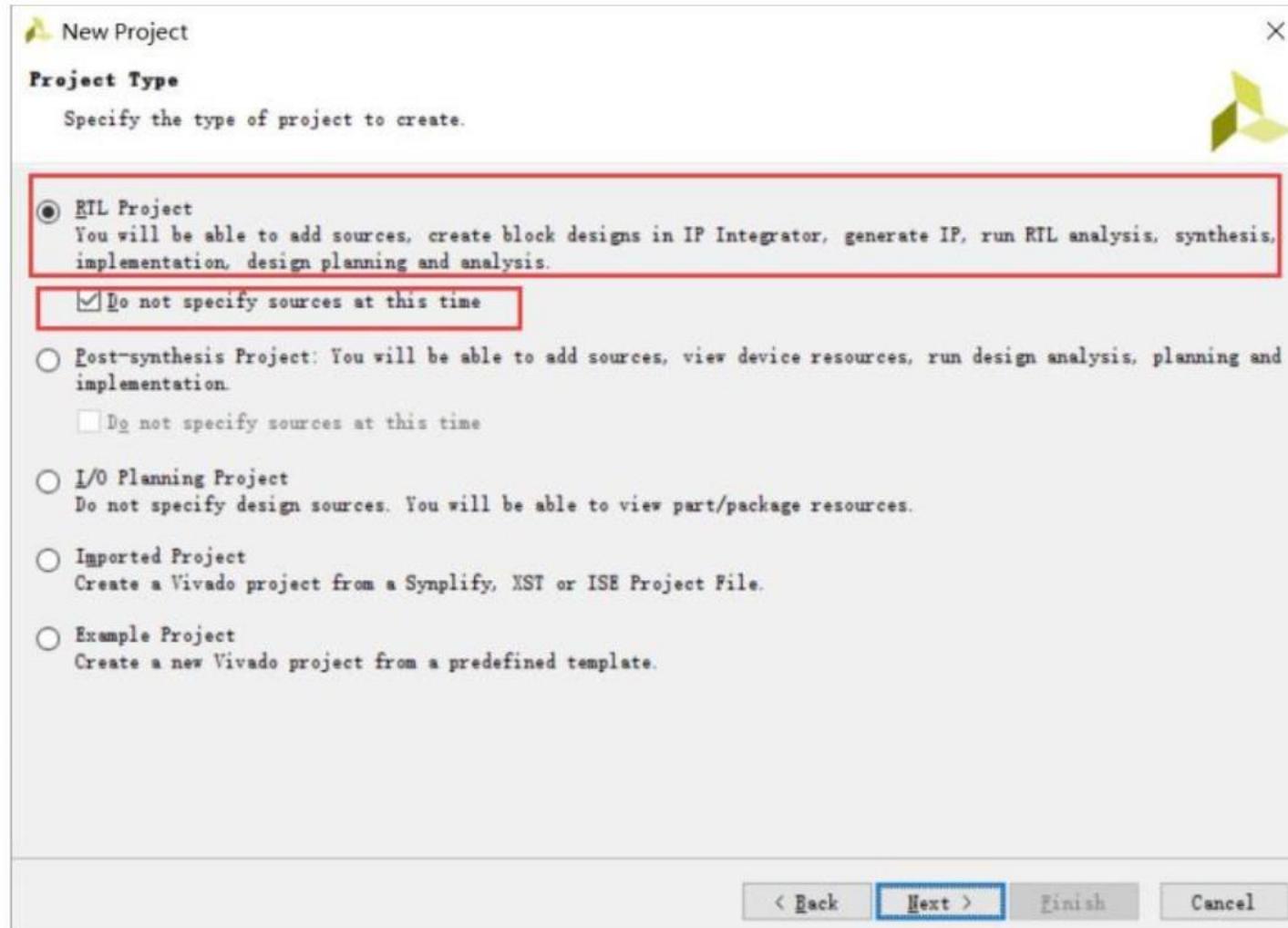


使用vivado仿真

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。



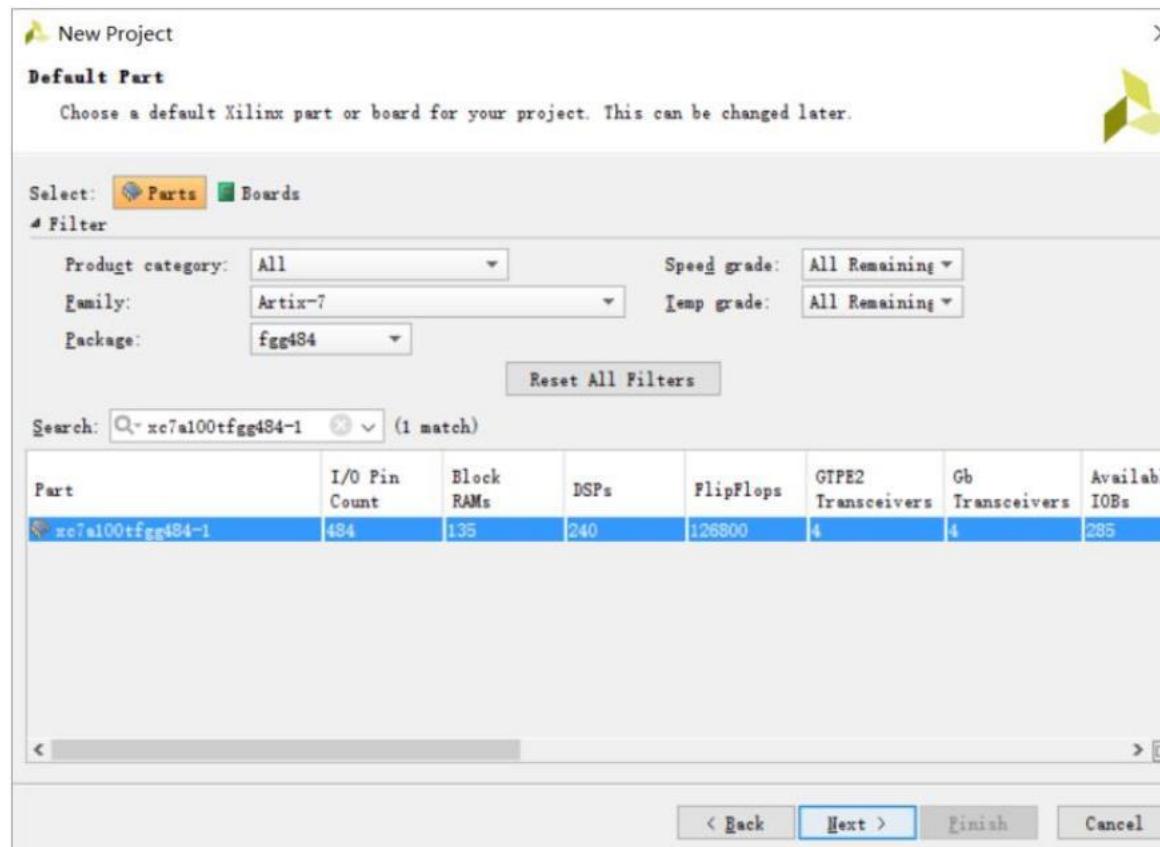
使用vivado仿真



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。

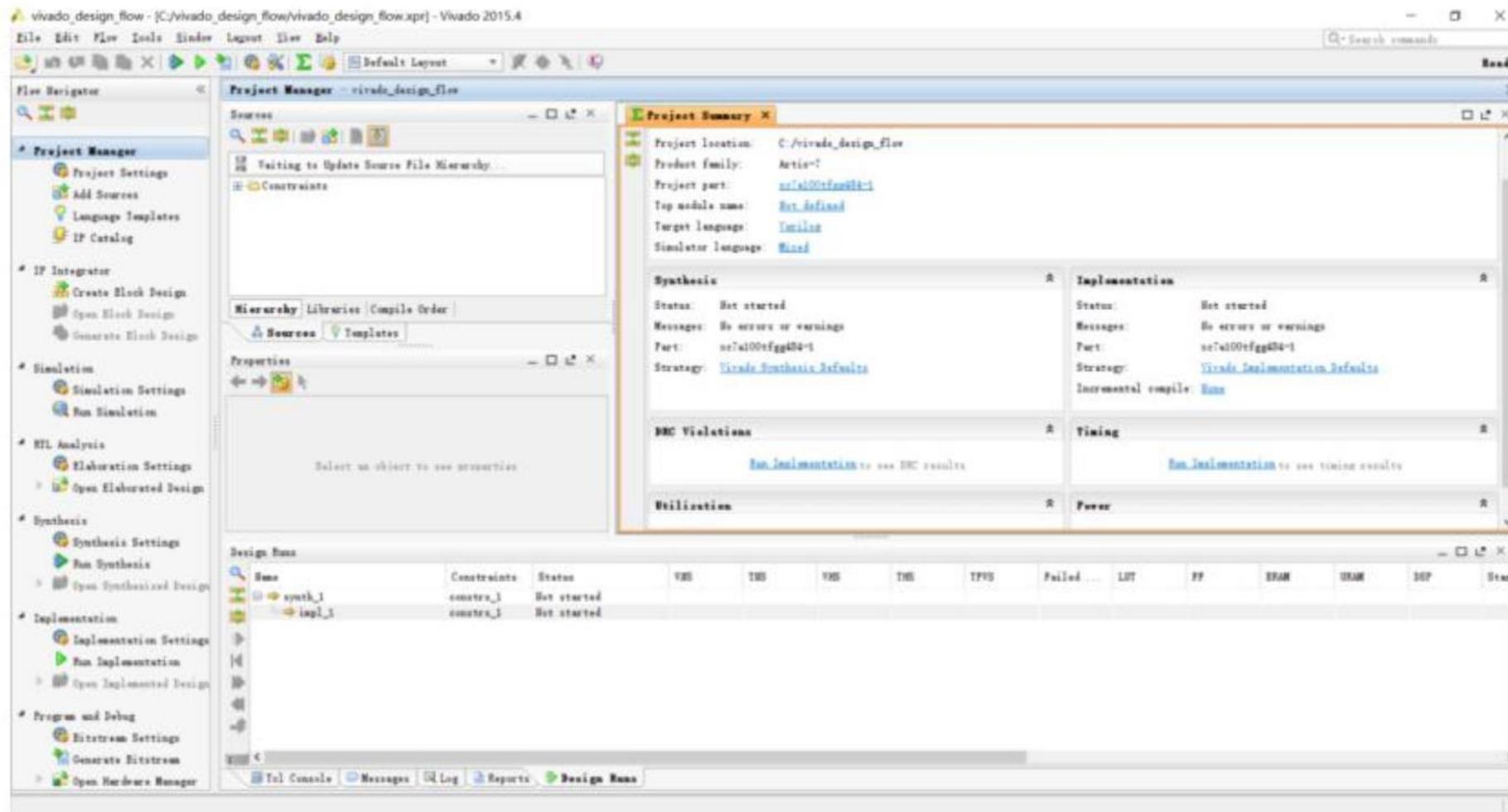
使用vivado仿真

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 XC7A100T-1FGG484-C 的器件，即 Family 和 Subfamily 均为 Artix-7，封装形式(Package)为 FGG484，速度等级(Speed grade)为-1，温度等级(Temp Grade)为 C)。点击 Next。

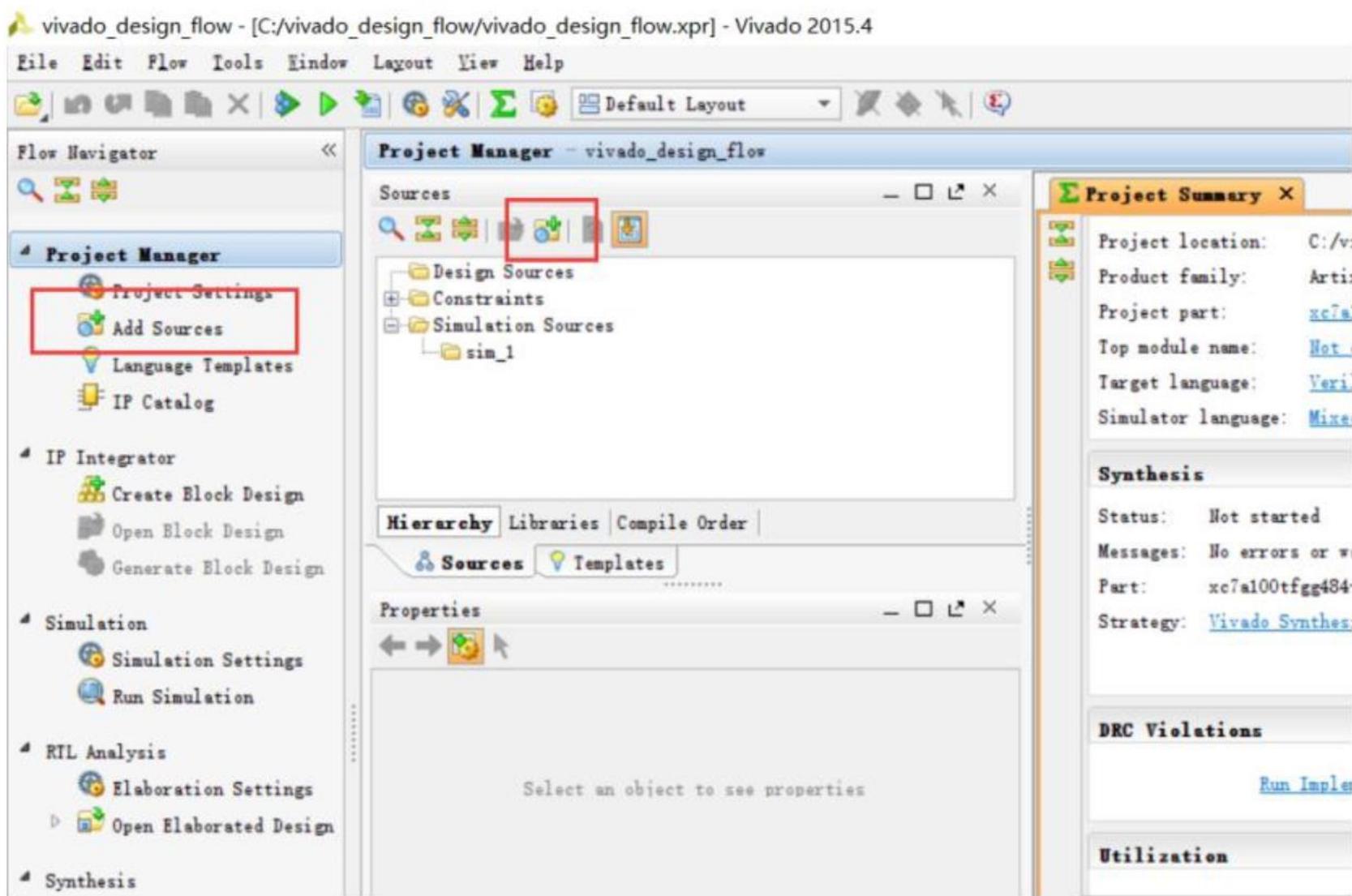


使用vivado仿真

得到如下的空白 Vivado 工程界面，完成空白工程新建。



使用vivado仿真

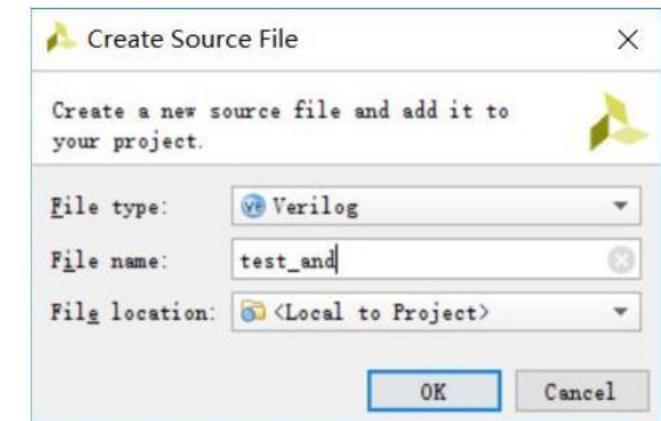
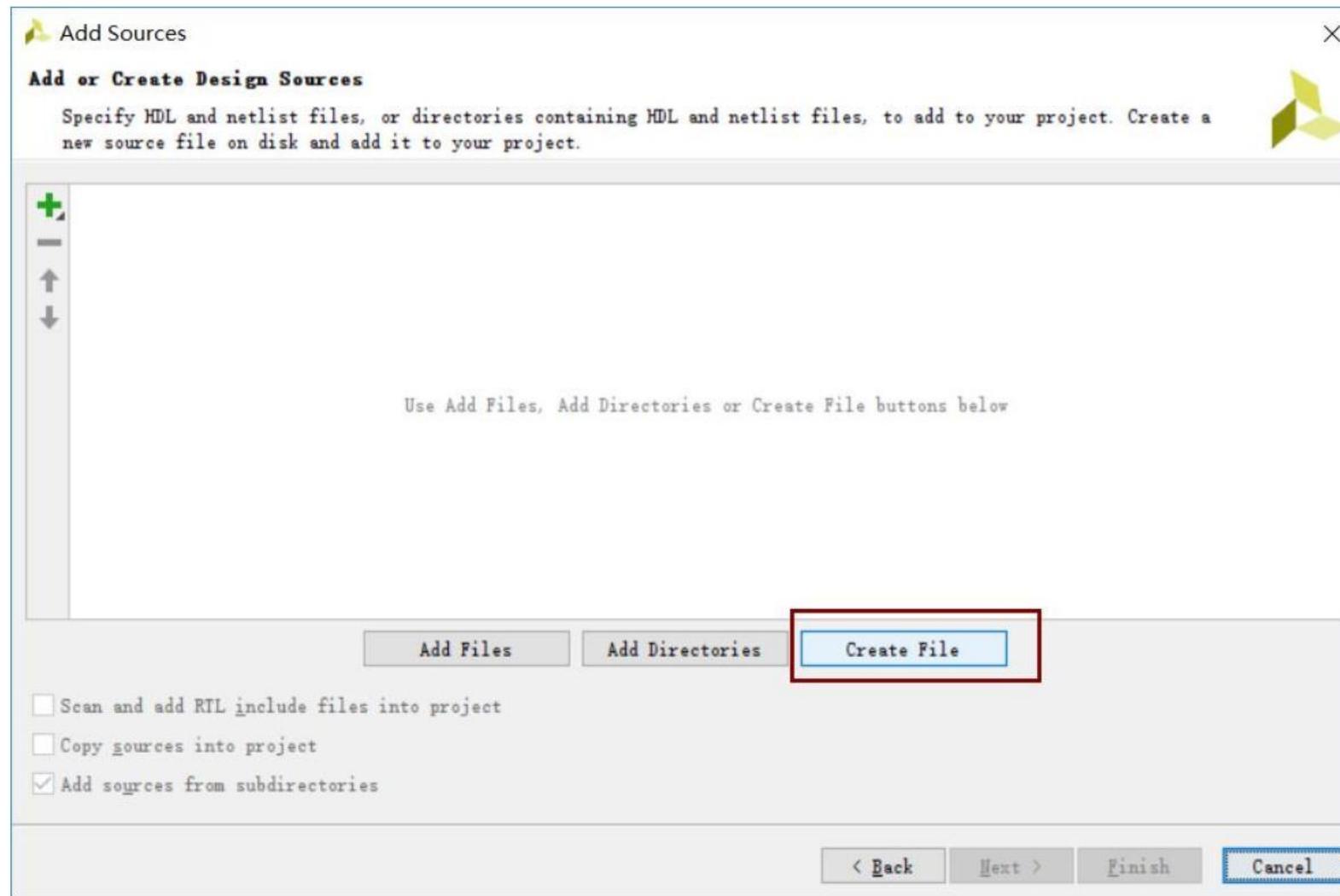


使用vivado仿真

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。

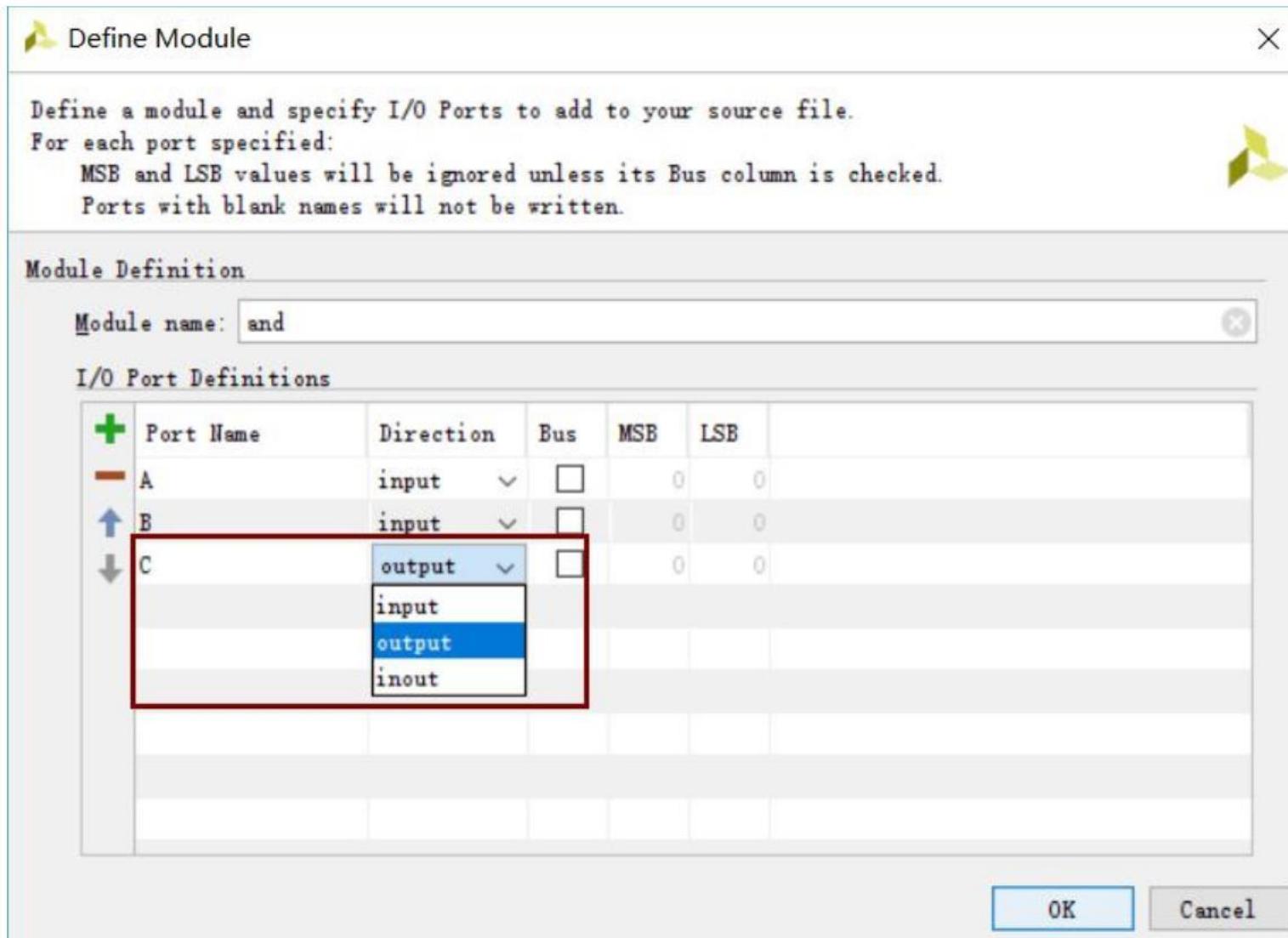


使用vivado仿真

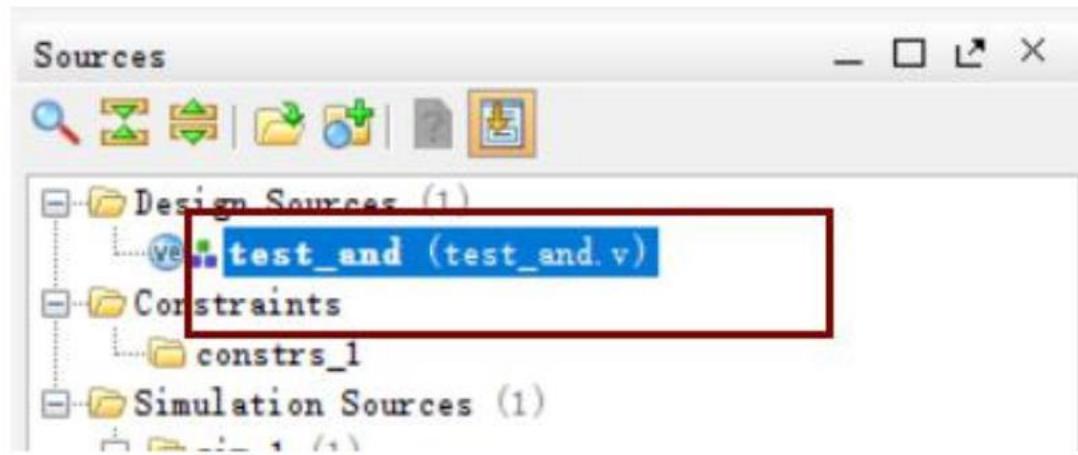


点击Finish完成

使用vivado仿真

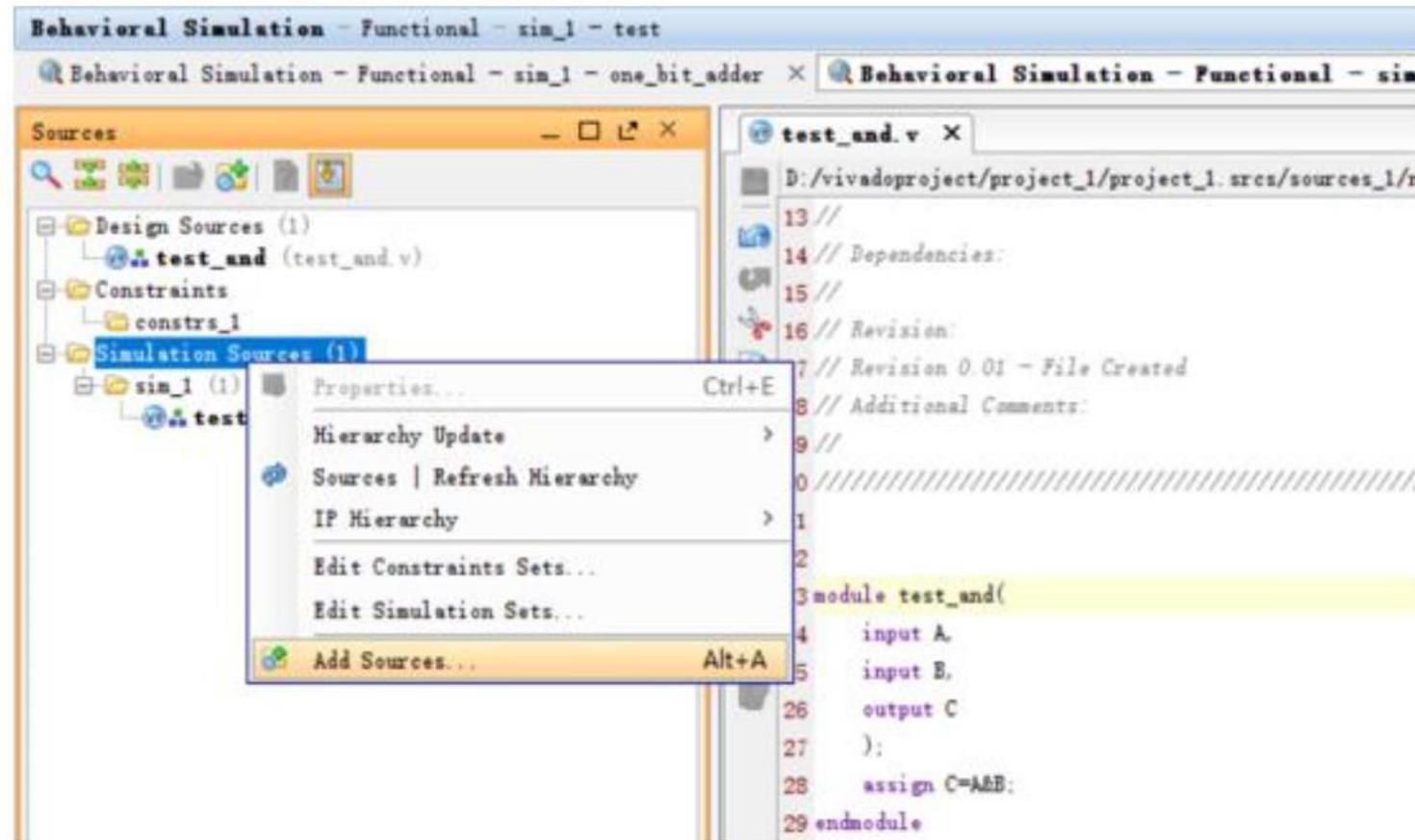


使用vivado仿真

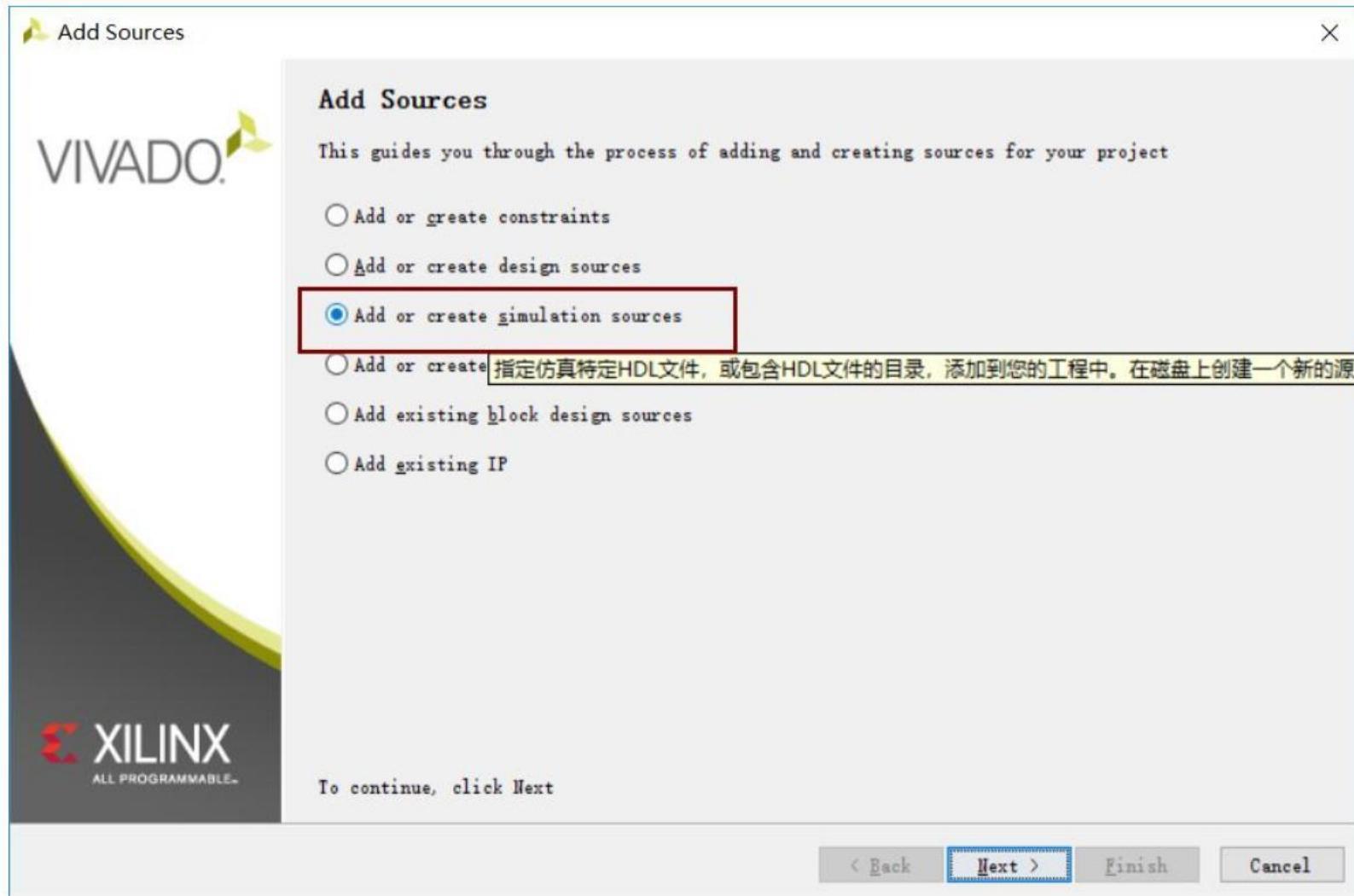


```
module test_and(
    input A,
    input B,
    output C
);
    assign C=A&B;
endmodule
```

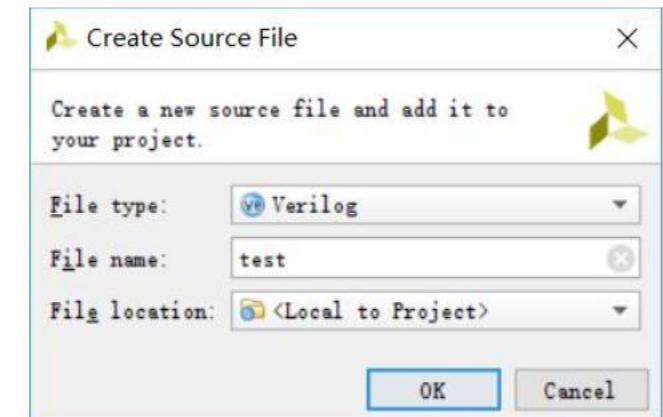
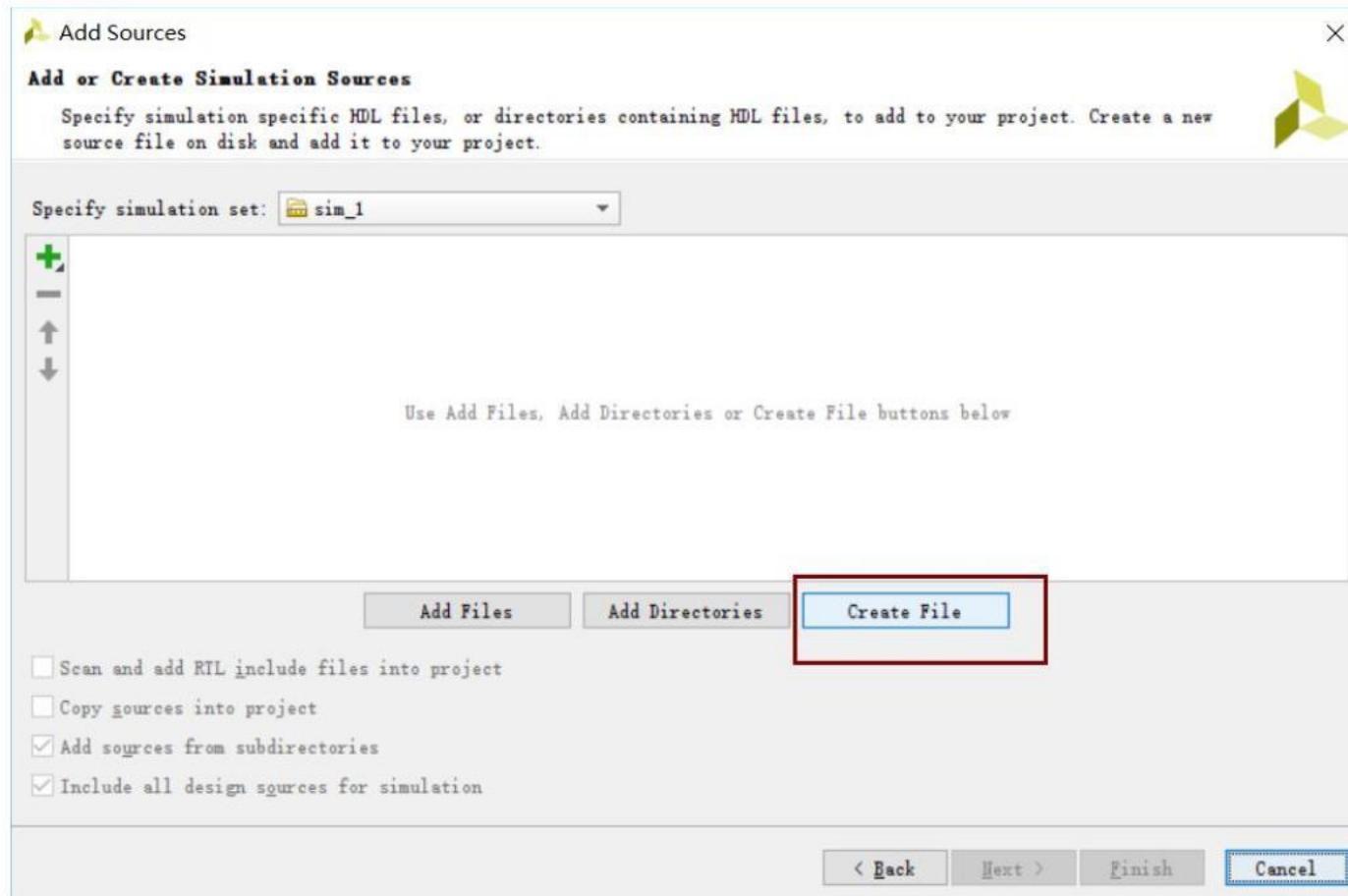
使用vivado仿真



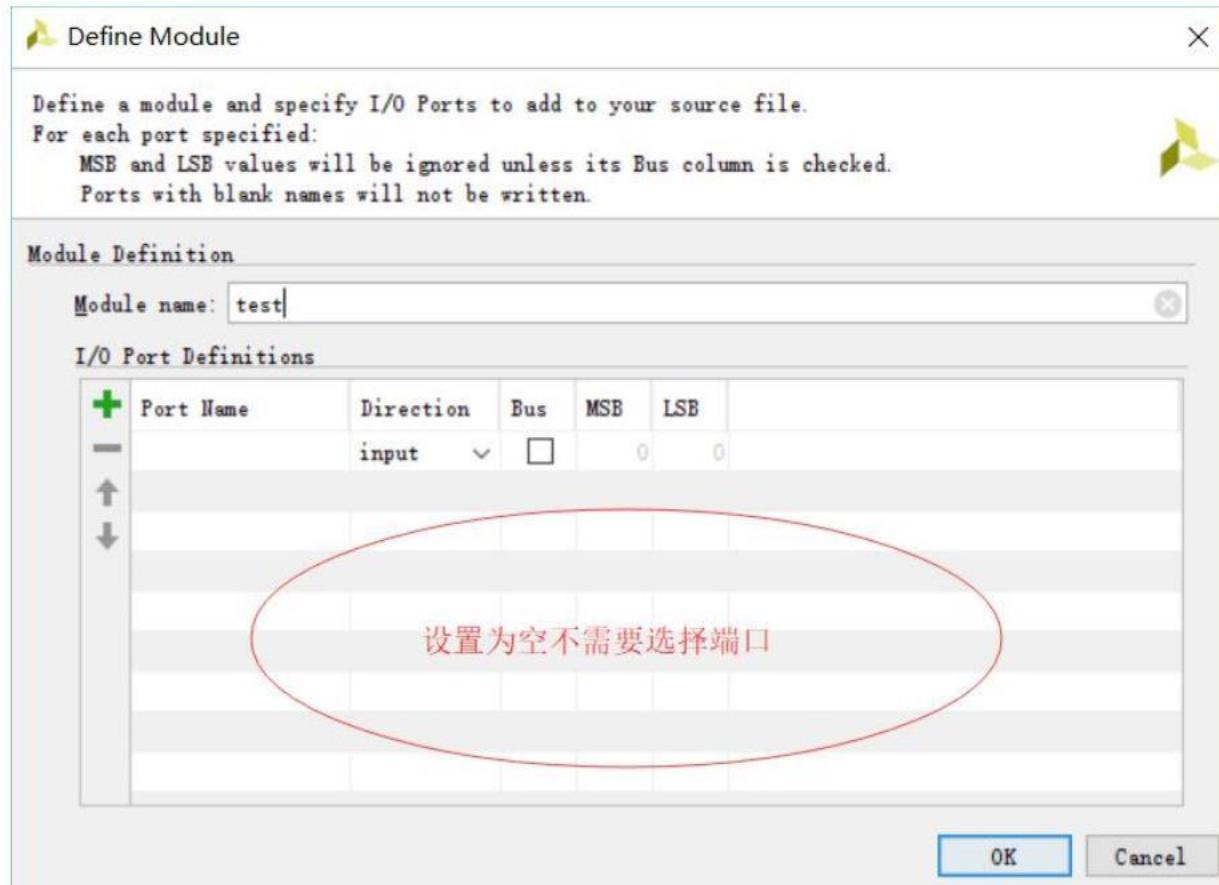
使用vivado仿真



使用vivado仿真

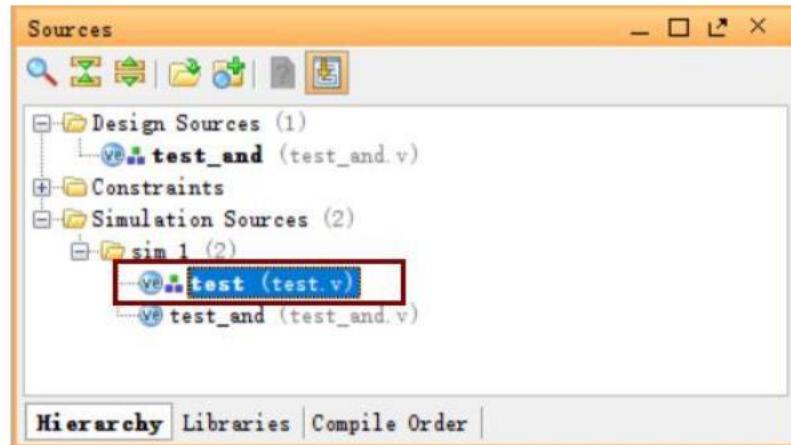


使用vivado仿真



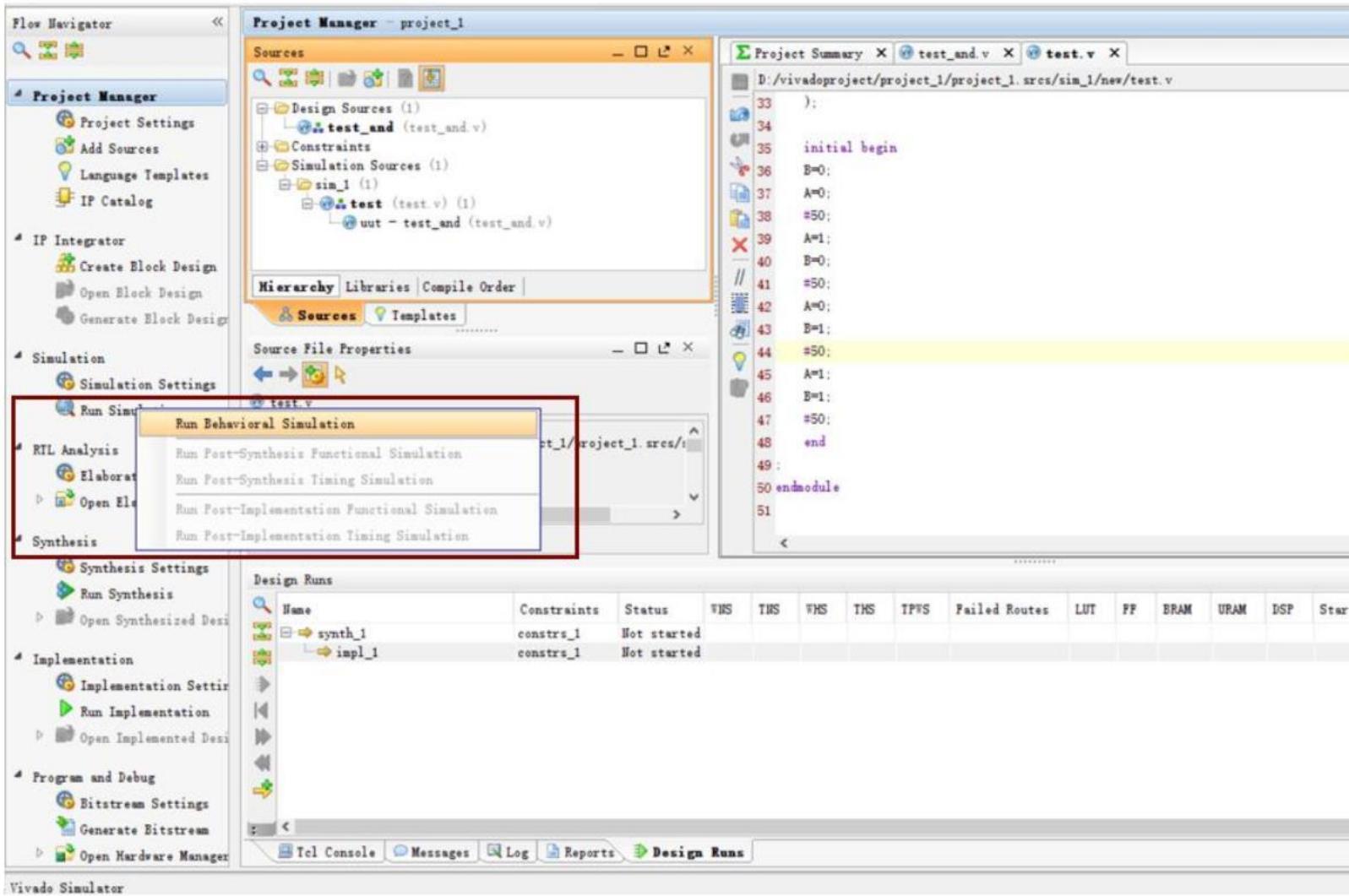
确认添加完成之后点击 Finish,
因为是激励文件不需要对外端口，
所以直接 Port 部分直接空着，点
击 ok。

使用vivado仿真

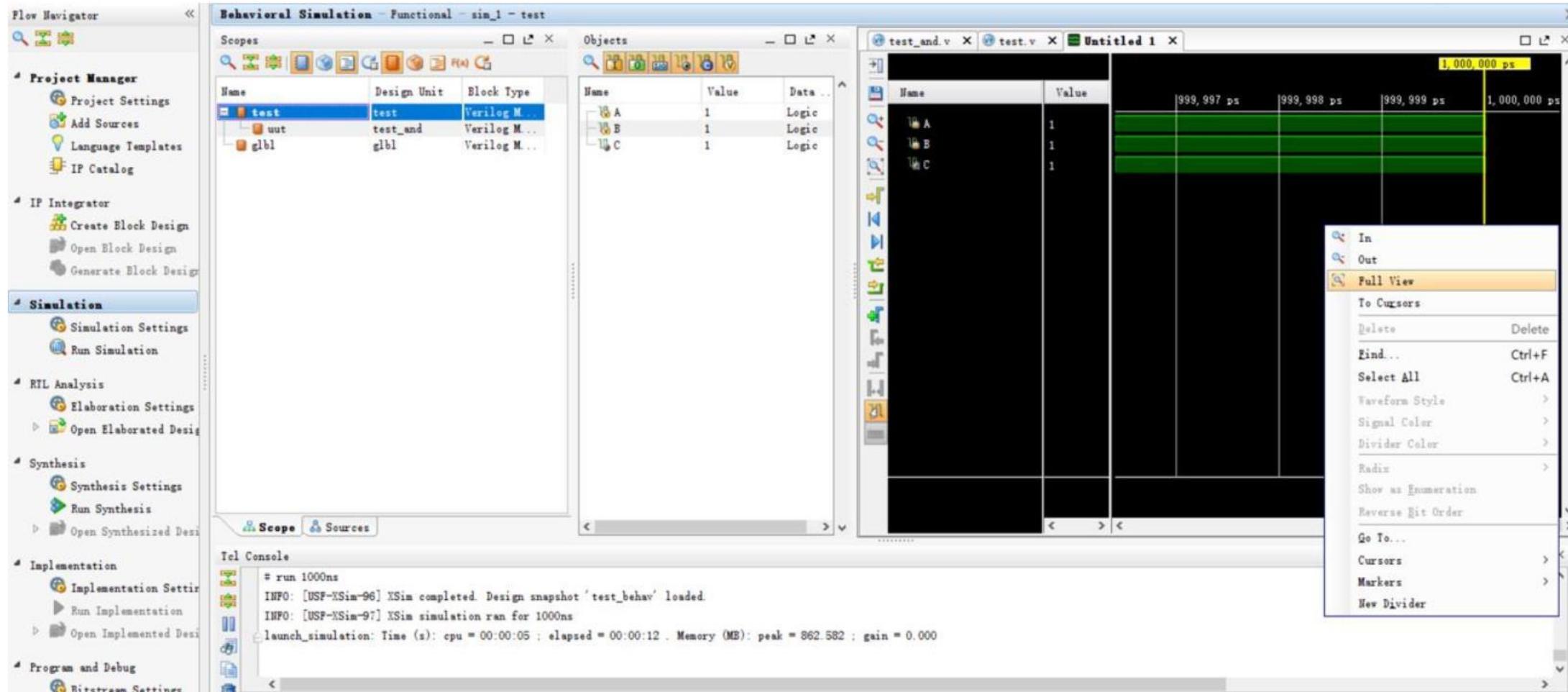


```
23 module test;
24     reg A;
25     reg B;
26
27     wire C;
28
29     test_and uut(
30         .A(A),
31         .B(B),
32         .C(C)
33     );
34     initial begin
35         B=0;
36         A=0;
37         #50;
38         A=1;
39         B=0;
40         #50;
41         A=0;
42         B=1;
43         #50;
44         A=1;
45         B=1;
46         #50;
47     end
48
49 endmodule
```

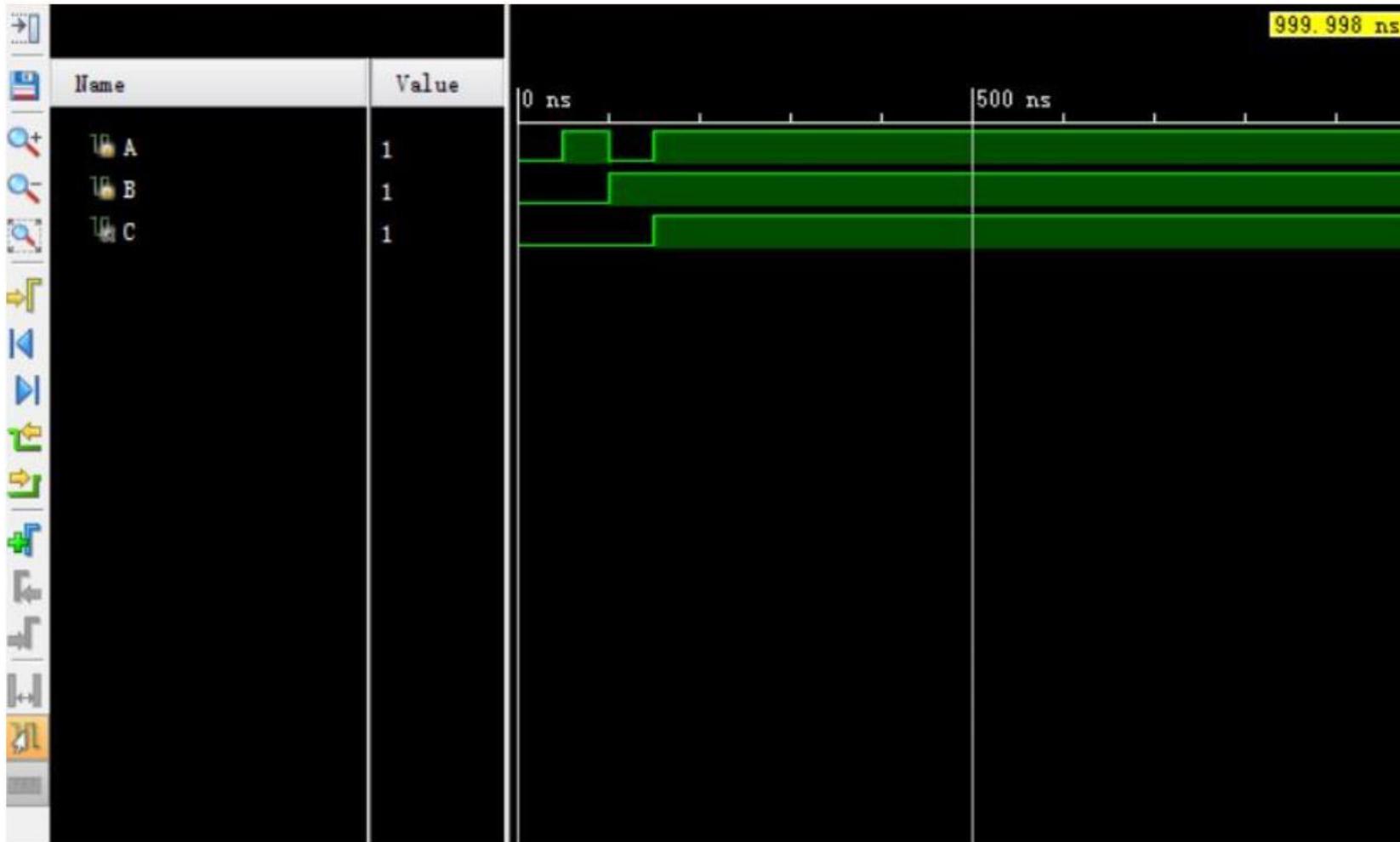
使用vivado仿真



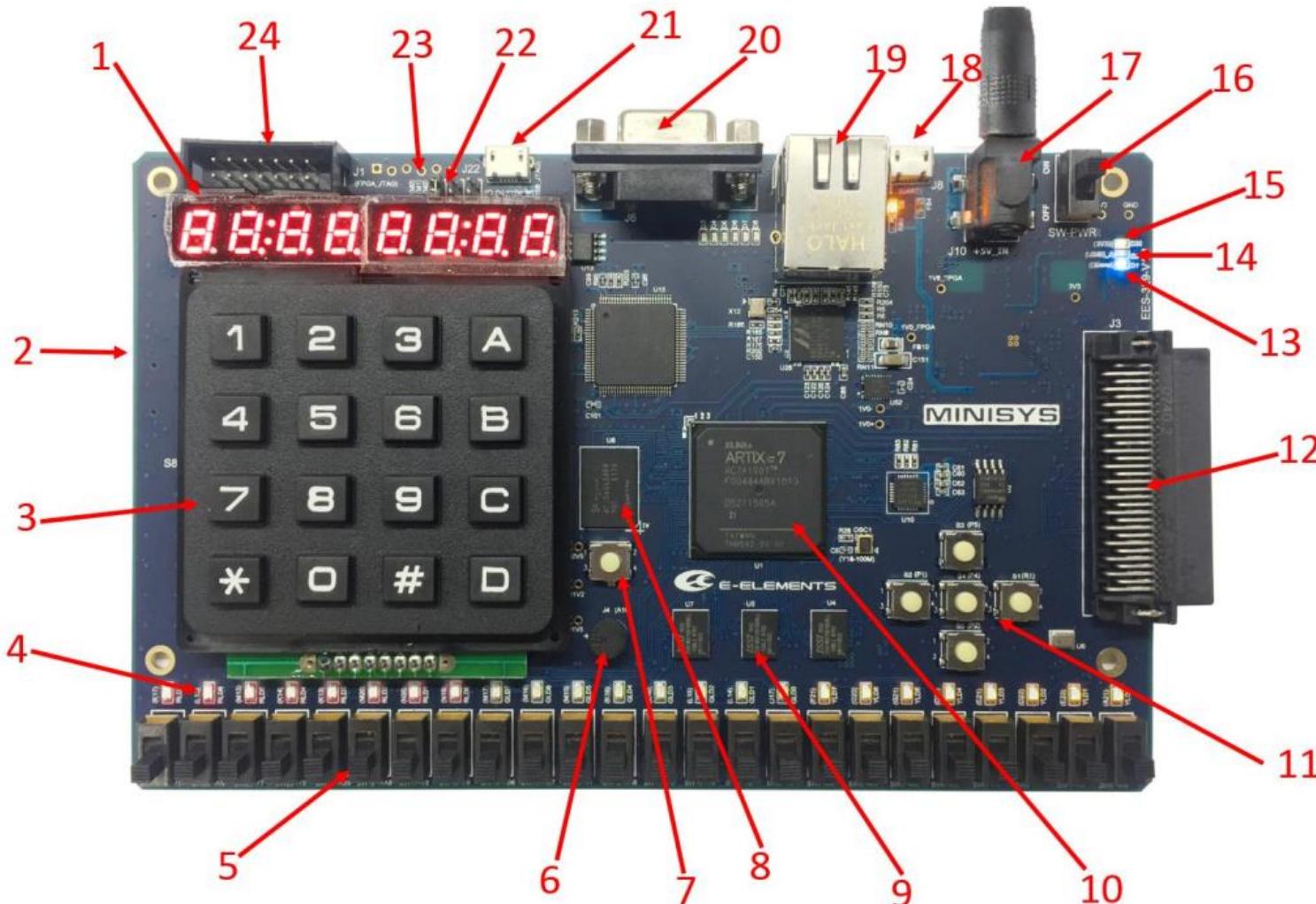
使用vivado仿真



使用vivado仿真

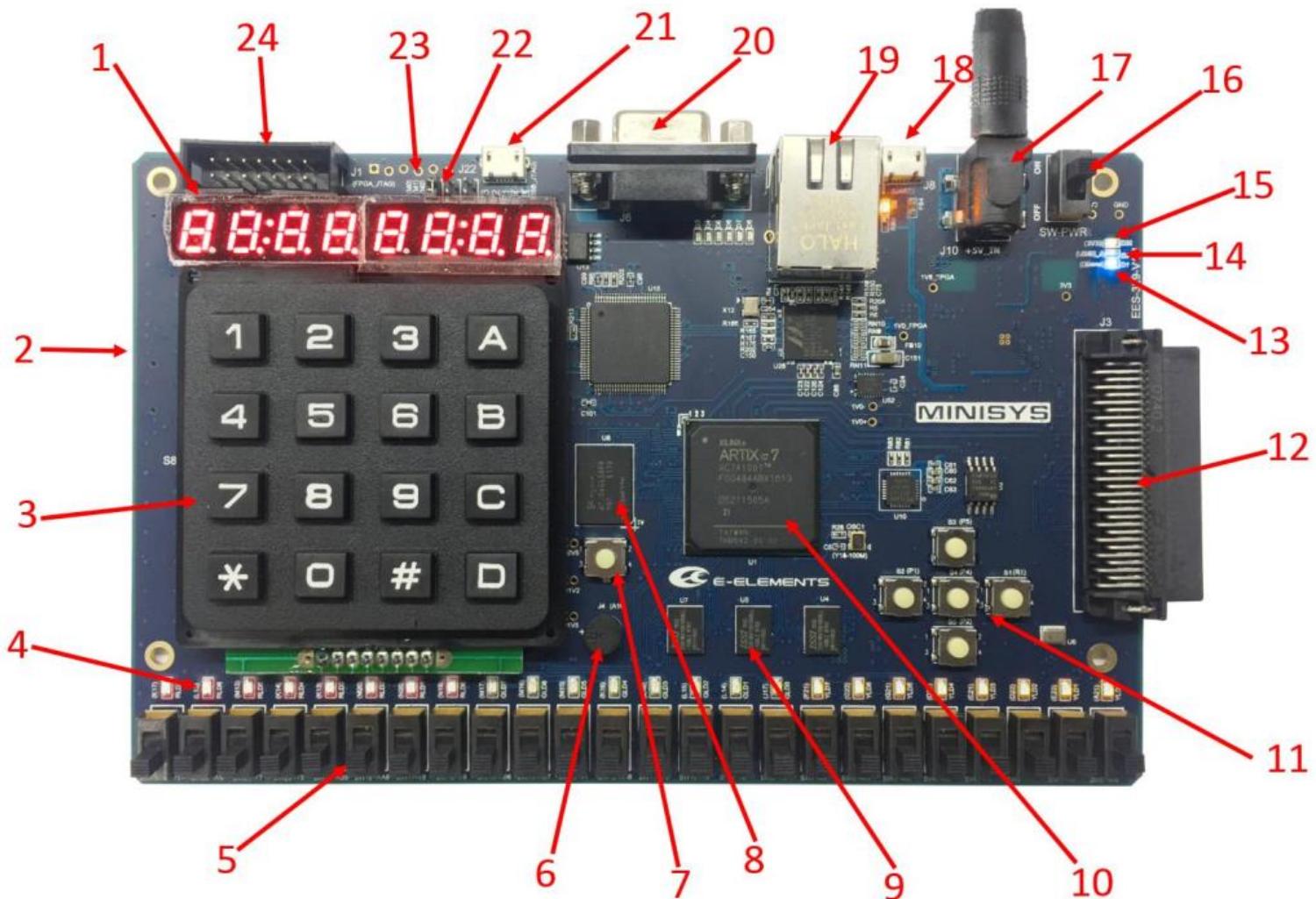


Minisys板卡图



标注	描述
1	8 个 7 段数码管
2	Micro SD 卡槽(板卡背面)
3	4*4 小键盘
4	LEDs (红、黄、绿各 8 个)
5	拨码开关
6	蜂鸣器
7	FPGA 复位按键
8	DDR3 SDRAM
9	SRAM
10	XC7A100T 主芯片
11	5 个按键开关
12	接口板连接器

Minisys板卡图



标注	描述
13	FPGA 烧写完成指示灯
14	USB_JTAG 指示灯
15	电源指示灯
16	电源开关
17	电源连接口
18	USB 转 UART 接口（供电用）
19	以太网接口
20	VGA 接口
21	USB_JTAG 接口（编程用）
22	编程跳线
23	用户扩展 IO
24	JTAG 接口

基本I/O设备

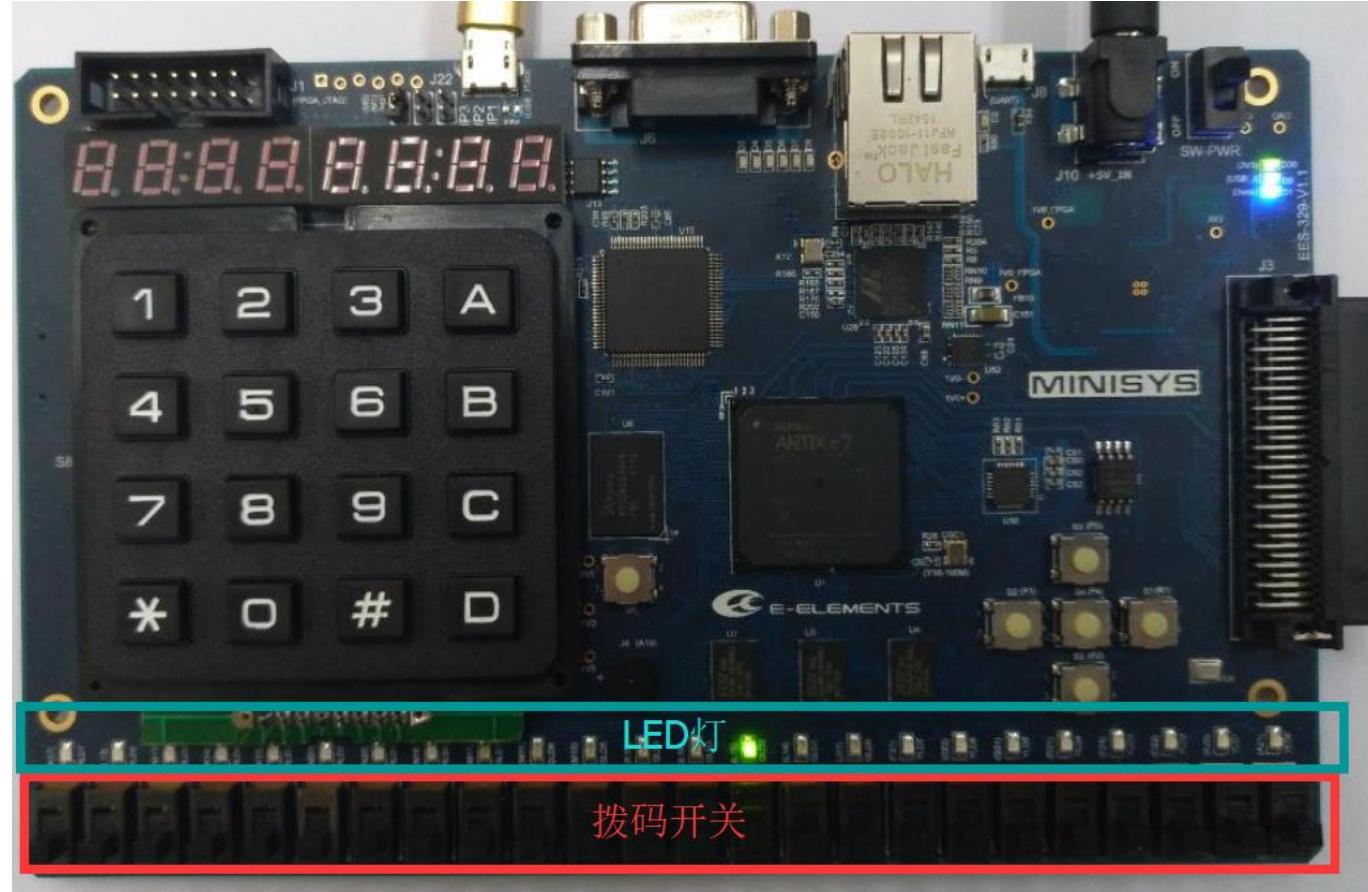
1. 拨码开关与LED灯

Minisys 实验板上有 24 个拨码开关，其板上标注为 SW23~SW0。在实验中，常将拨码开关作为数据输入，当开关拨到下档时，表示输入为 0，否则为 1。

另外，实验板上还有 24 个 LED 灯（红、绿、黄分别 8 个），板上标号为 RLD7~0、GLD7~0 和 YLD7~0。当 FPGA 相应管脚的输出为高电平时，所连接的 LED 灯被点亮，否则灯熄灭。

基本I/O设备

1. 拨码开关与LED灯



基本I/O设备

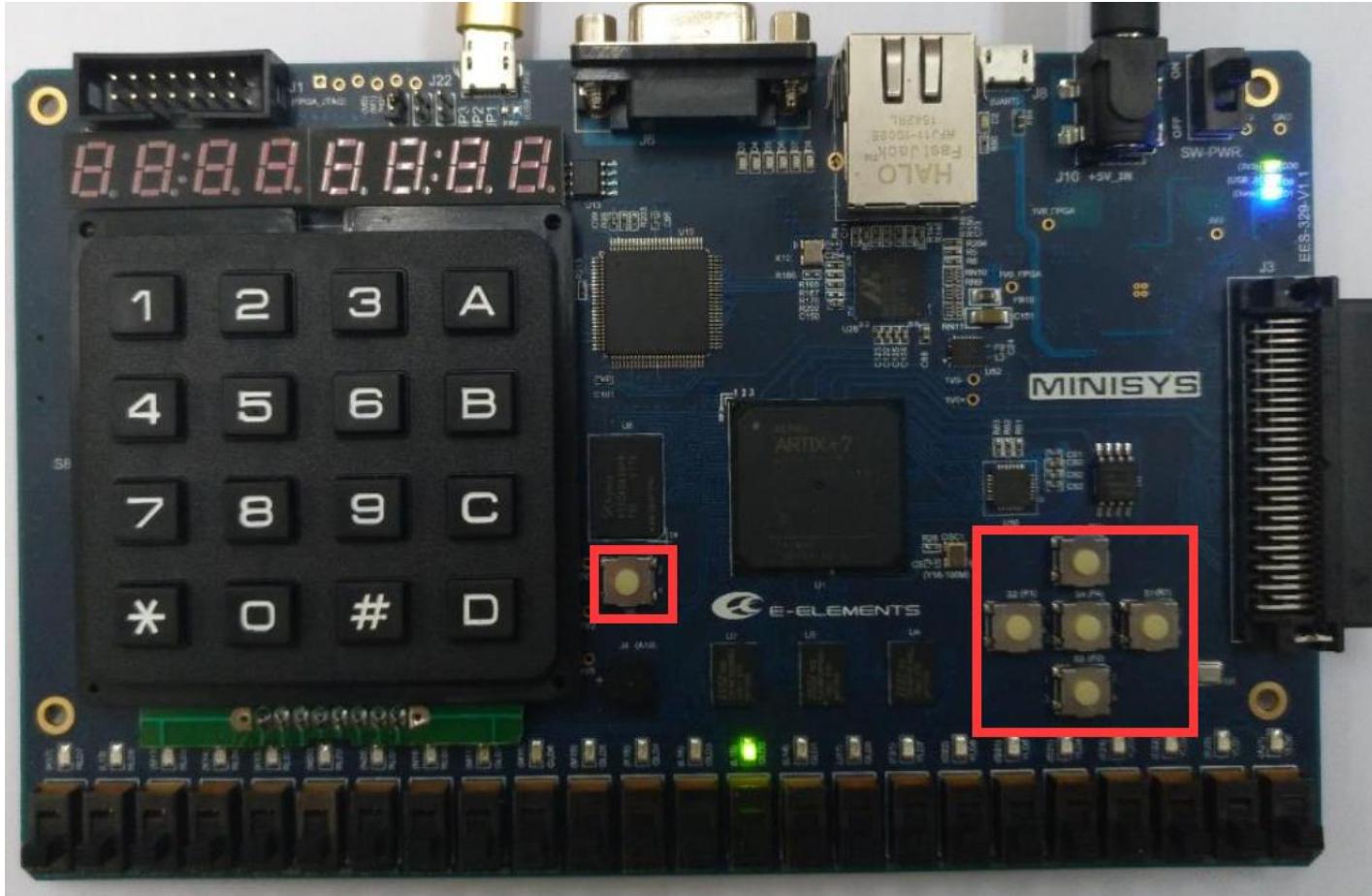
2. 按键开关

当某一按键按下时，其对应的 FPGA 输入为 1，否则为 0。

板上共有 6 个按键开关 (S1~S6) , 其中的 S6 按键被选作 FPGA 的复位按键。在开发学习过程中，建议在需要时设置一个复位输入，这不仅是所开发系统的功能需求，还有利于代码调试。

基本I/O设备

2. 按键开关



基本I/O设备

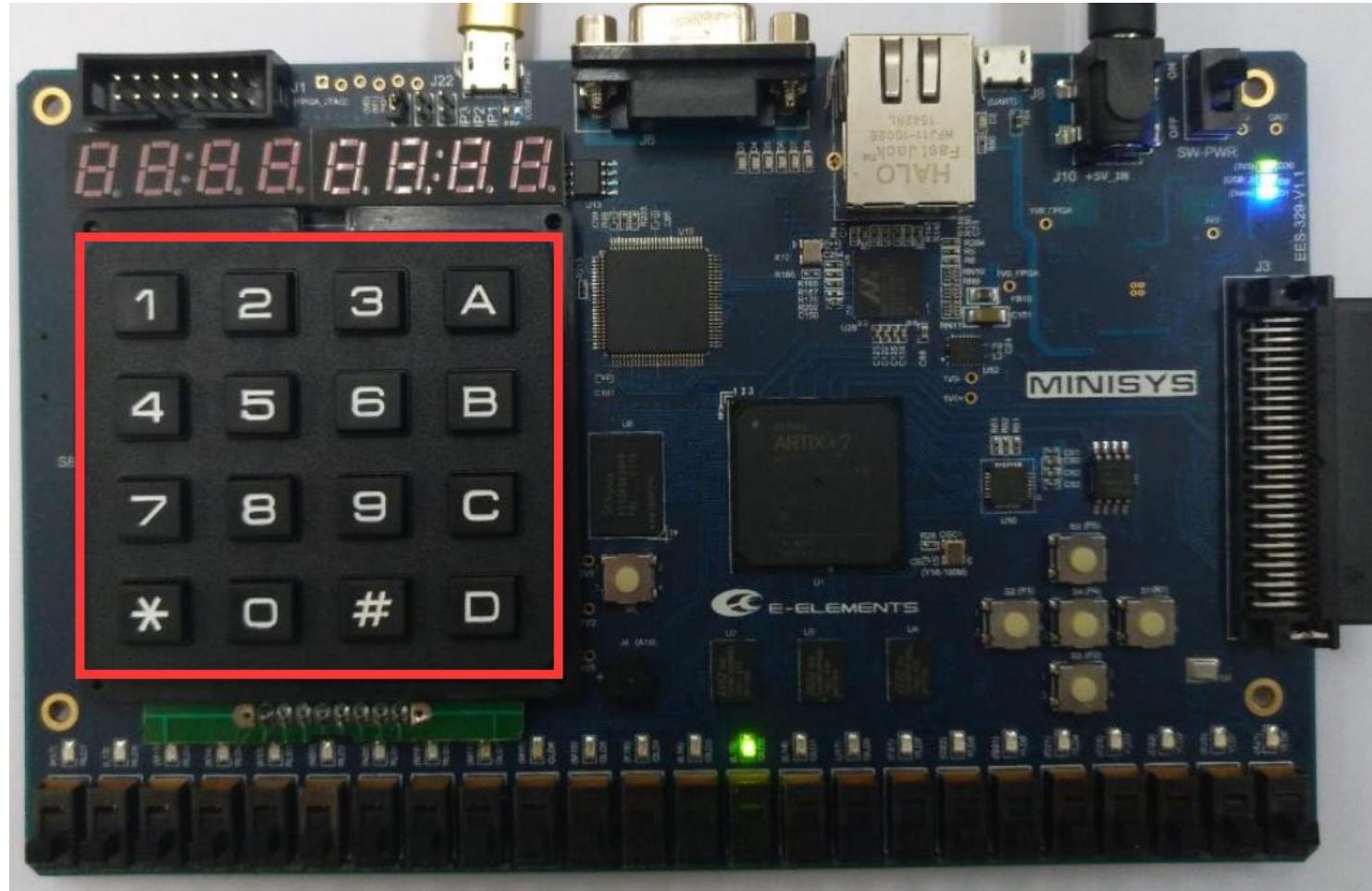
3. 4×4 矩阵键盘

Minisys 实验板上还连接了一个 4×4 的矩阵键盘，用以方便快速的进行数值输入。

主芯片接受的是按键的“坐标”，而不是其所对应的键值。要想获得需要的键值，需要在程序中对行、列信号的每一种组合方式进行翻译。

基本I/O设备

3. 4×4矩阵键盘



基本I/O设备

4. 七段数码管

Minisys 实验板上有两个 4 位带小数点的七段数码管。

其中 A7~A0 是数码管 8 个位的使能信号，而 CA~CG/DP 则对应各个位上七个段以及小数点的触发信号。需要注意的是，使能信号和触发信号都是低电平触发的。

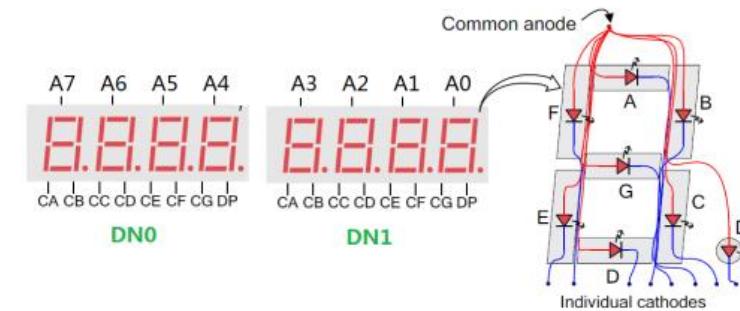


图 1-7 共阳极数码管电路结构

基本I/O设备

4. 七段数码管

8个位中的各个相应的段及小数点分别连接到一组低电平触发的引脚上，他们被称为CA、CB、CC、...、CG、DP，其中，CA接到这8个数码管中每一个数码管A段的负极,CB接到这8个数码管中每一个数码管B段的负极，以此类推。

此外，每一个数码管都有一个使能信号 A[7:0]。A[7:0]通过一个反相器接到对应数码管的每一个段的正极上。比如说，只有到 A[0]为 0 的时候，最右侧数码管的显示才会受到CA...CG 这几个信号的驱动。

基本I/O设备

4. 七段数码管

要想让每个数码管显示不同的数字，使能信号（A[7:0]）和段信号（CA...CG）必须依次地被持续驱动，数码管之间的刷新速度应该足够快这样就看不出来数码管之间在闪烁。举个例子，如果想在数码管 0 上显示数字 3 而数码管 1 上显示数字 9，可以把 CA...CG 设置为显示数字 3，并拉低 A[1]信号，然后再把 CA...CG 设置为显示数字 9 并拉高 A[1]拉低 A[2]。刷新频率可以设置为 2ms 刷新一次，这样人眼就看不出闪烁了。

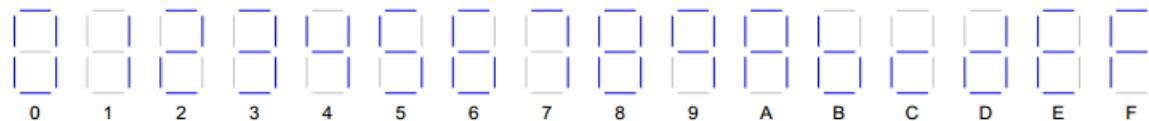


图 1-8 7-段数码管显示功能

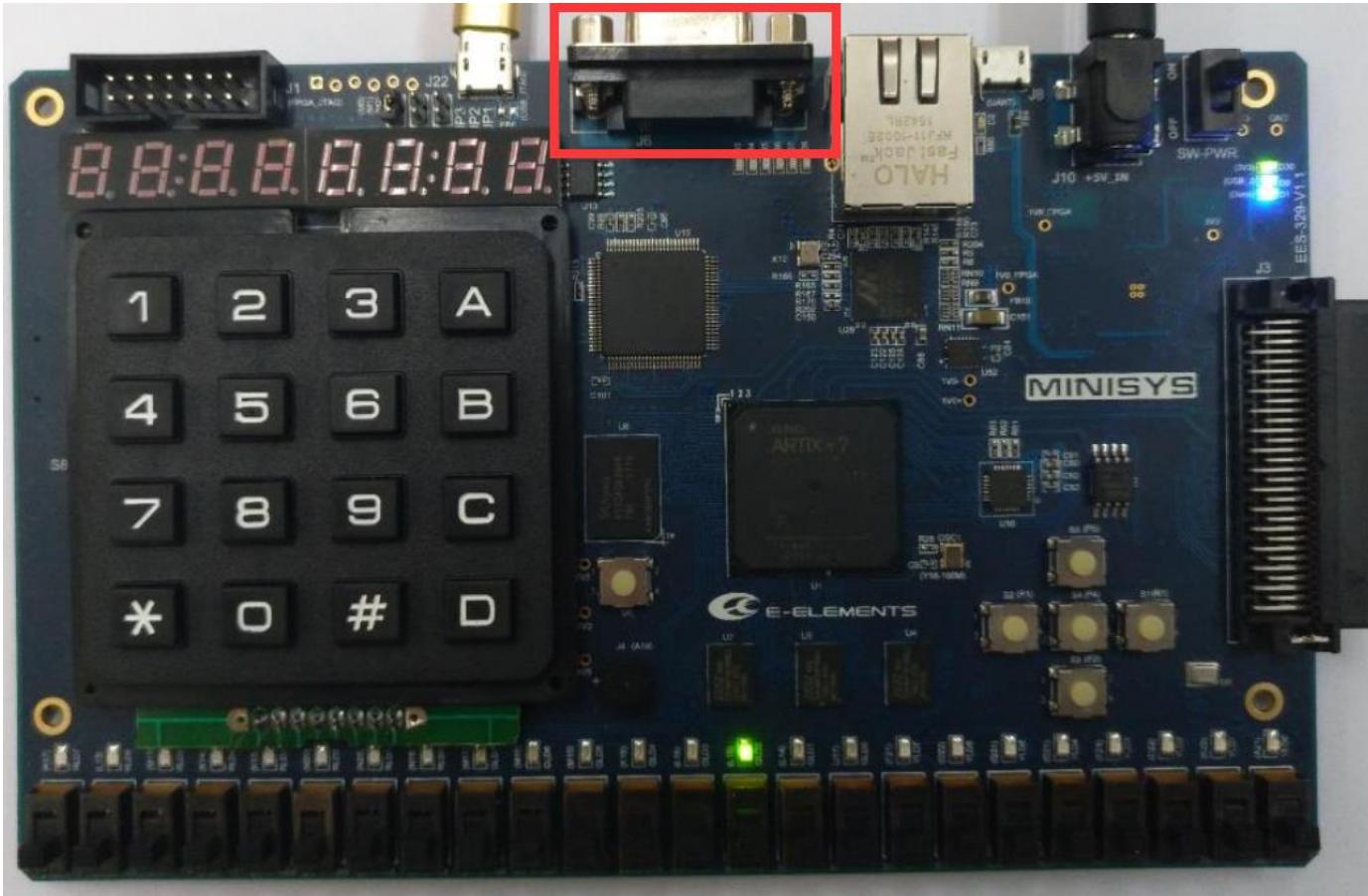
基本I/O设备

5. VGA模块

Minisys 使用 14 路 FPGA 信号生成一个 VGA 端口，该端口包括 4 位的红、绿、蓝三基色信号和标准行、列同步信号。色彩信号由电阻分压电路产生，支持 12 位的 VGA 彩色显示，共有 4096 种不同的颜色配置。

基本I/O设备

5. VGA模块



基本I/O设备

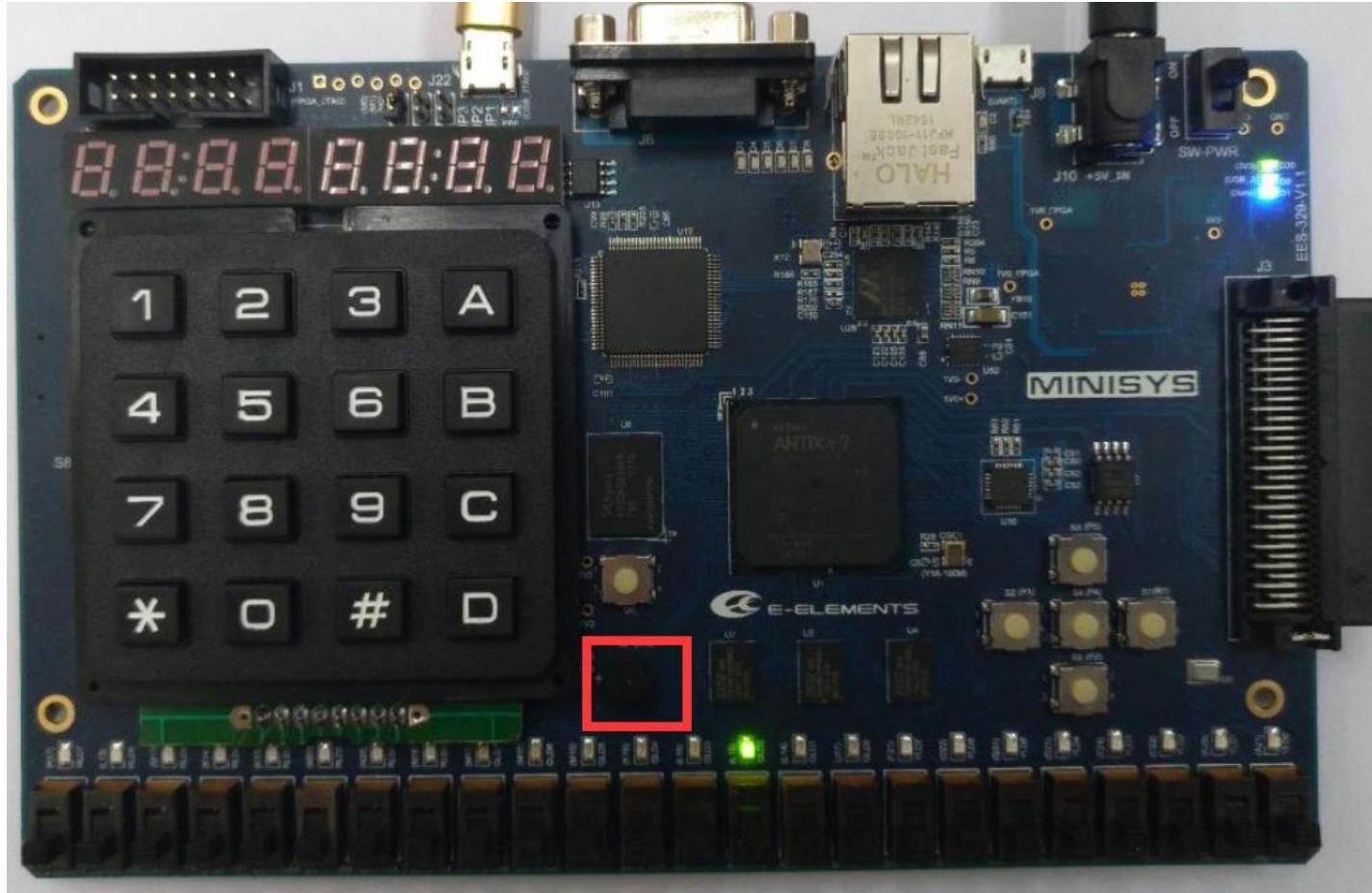
6. 蜂鸣器

除了上述的各种视觉输出部件，minisys 实验板上还配置了一个蜂鸣器用作声音输出部件。

主芯片通过 A19 管脚向蜂鸣器输出一个电信号，该信号的频率由用户决定。在该信号驱动下，蜂鸣器内部发生机械振动，发出相应频率的声音。

基本I/O设备

6. 蜂鸣器



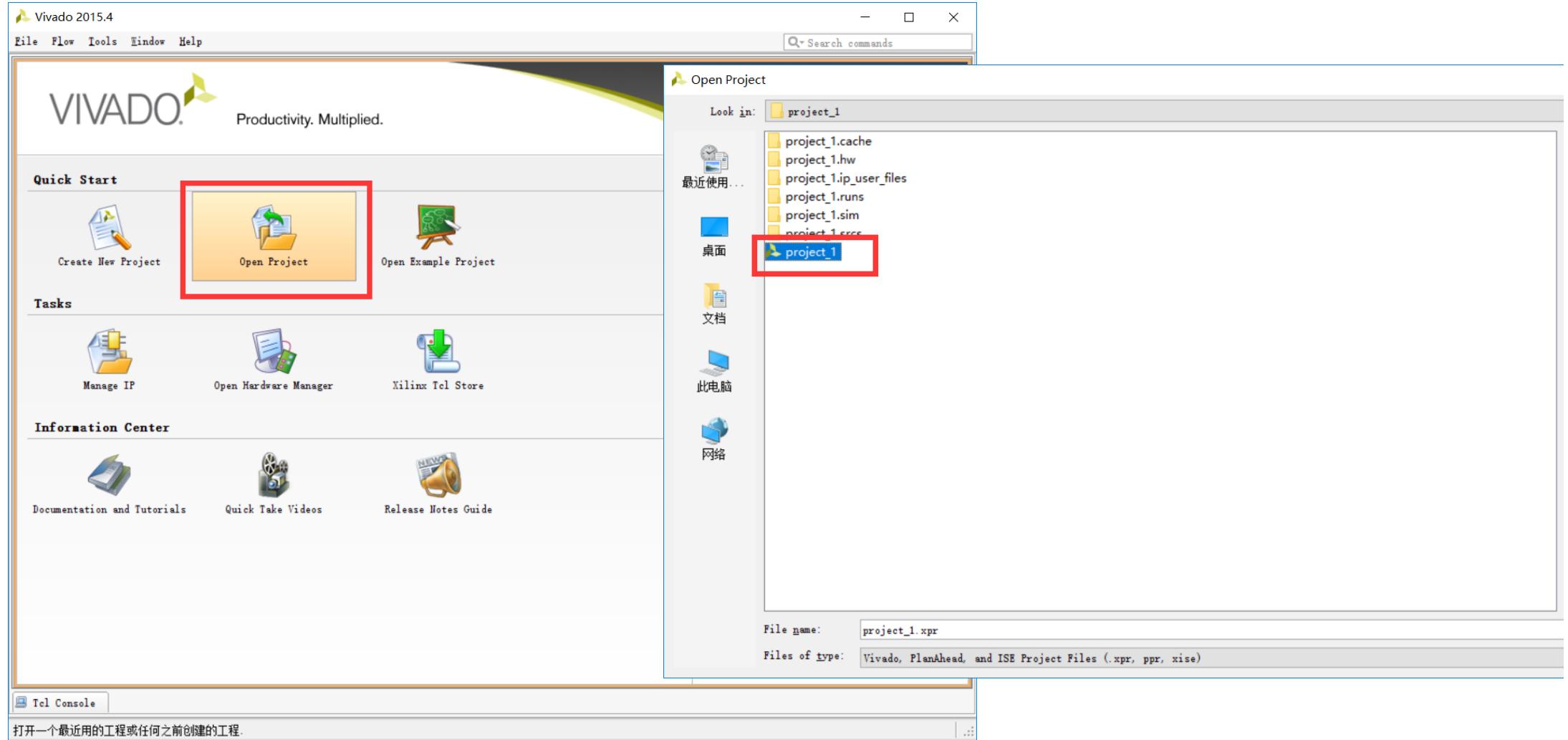
Vivado工程

以**test_and**模块为例，将其烧写进**Minisys**实验板。

打开已有的**project**工程文件，其中应包含设计源文件**test_and.v**和测试激励文件**test.v**，在完成行为级仿真，验证了模块功能正确后即可进行后续的综合、实现，生成bit文件及烧写至**Minisys**实验板的操作。



Vivado 工程



约束文件

输入、输出 → 管脚

需要**约束文件**来约束输入输出与管脚的关系！

两种添加约束文件的方式：

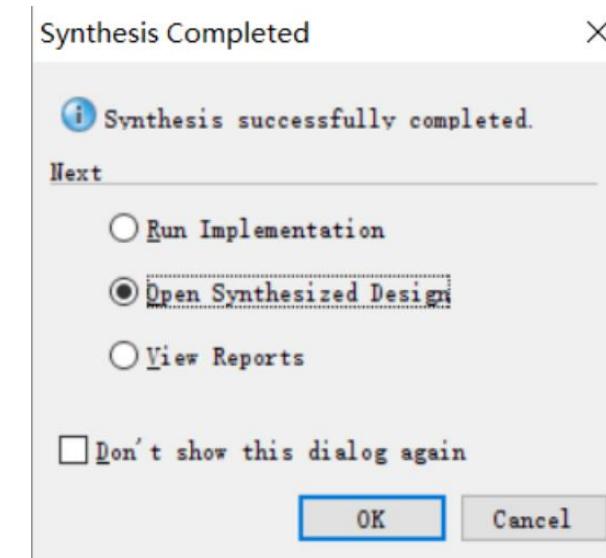
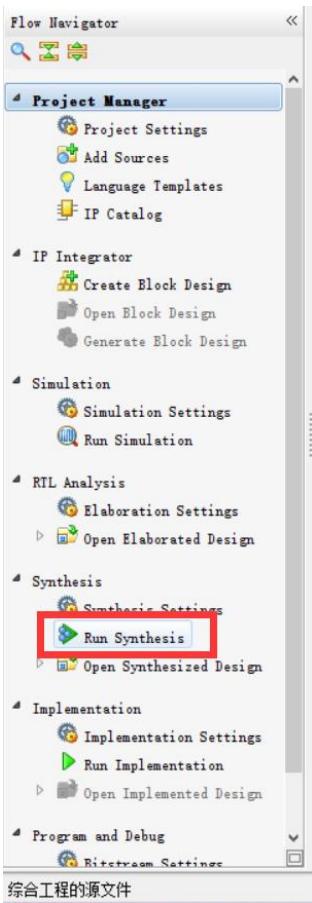
1. 利用Vivado中IO planning 功能
2. 可以直接新建XDC的约束文件，手动输入约束命令

约束文件

1. 利用Vivado中IO planning 功能

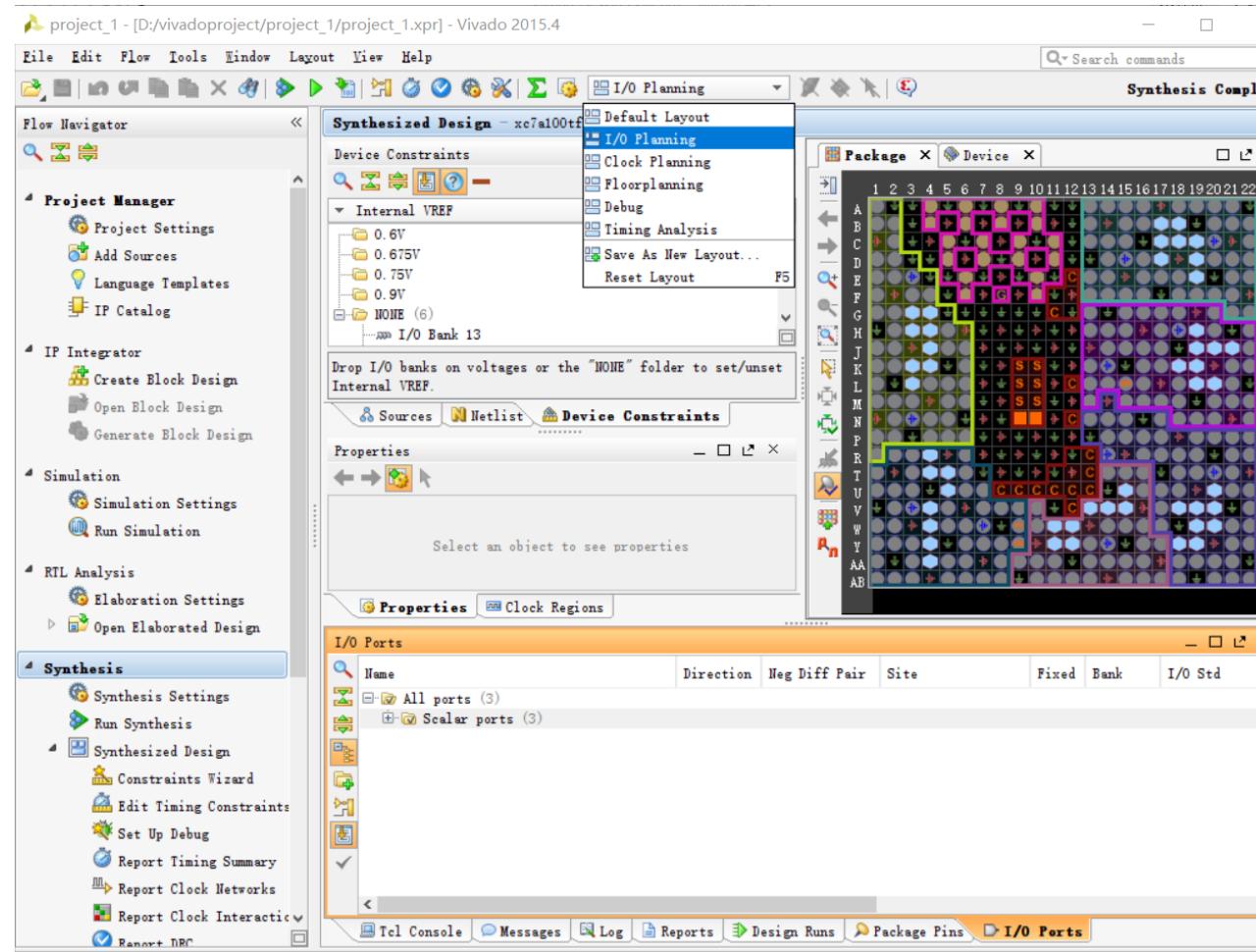
点击 Flow Navigator 中 Synthesis 中的 Run Synthesis，先对工程进行综合。

综合完成之后，选择 Open Synthesized Design，打开综合结果。



约束文件

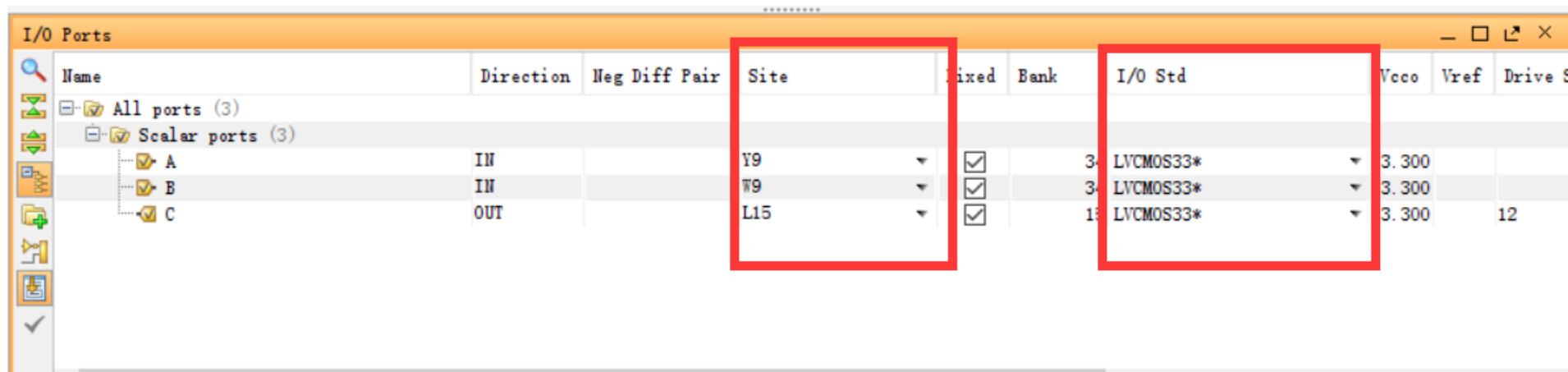
此时应看到如下界面，如果没出现如下界面，在图示位置的 layout 中选择 IO planning 一项。



约束文件

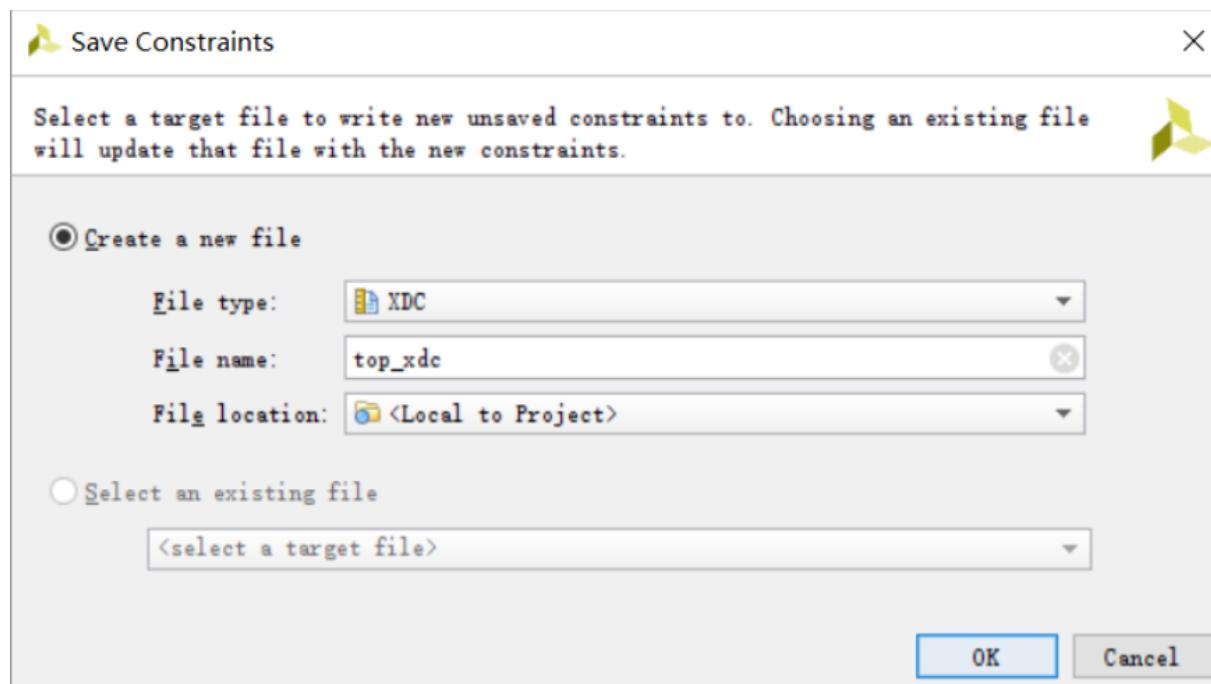
在右下方的选项卡中切换到 I/O ports 一栏，并在对应的信号后，输入对应的 FPGA 管脚标号（或将信号拖拽到右上方 Package 图中对应的管脚上），并指定 I/O std。具体的 FPGA 约束管脚和 IO 电平标准，可参考对应板卡的用户手册或原理图）。

在此，我们选择 SW23 和 SW22 作为输入，YLD2 作为输出，从板子上可以看到 SW23 和 SW22 的引脚分别是 Y9 和 W9，YLD2 的引脚为 L15



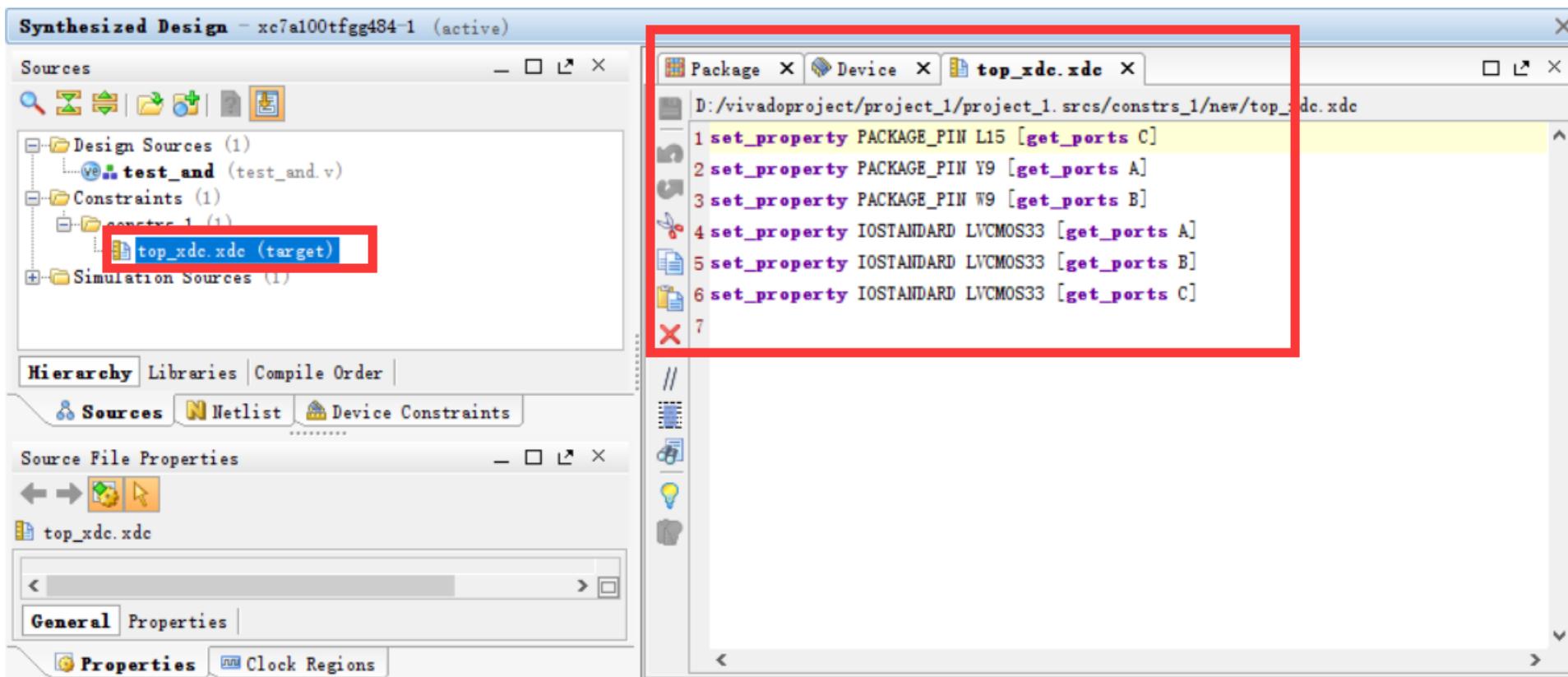
约束文件

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。



约束文件

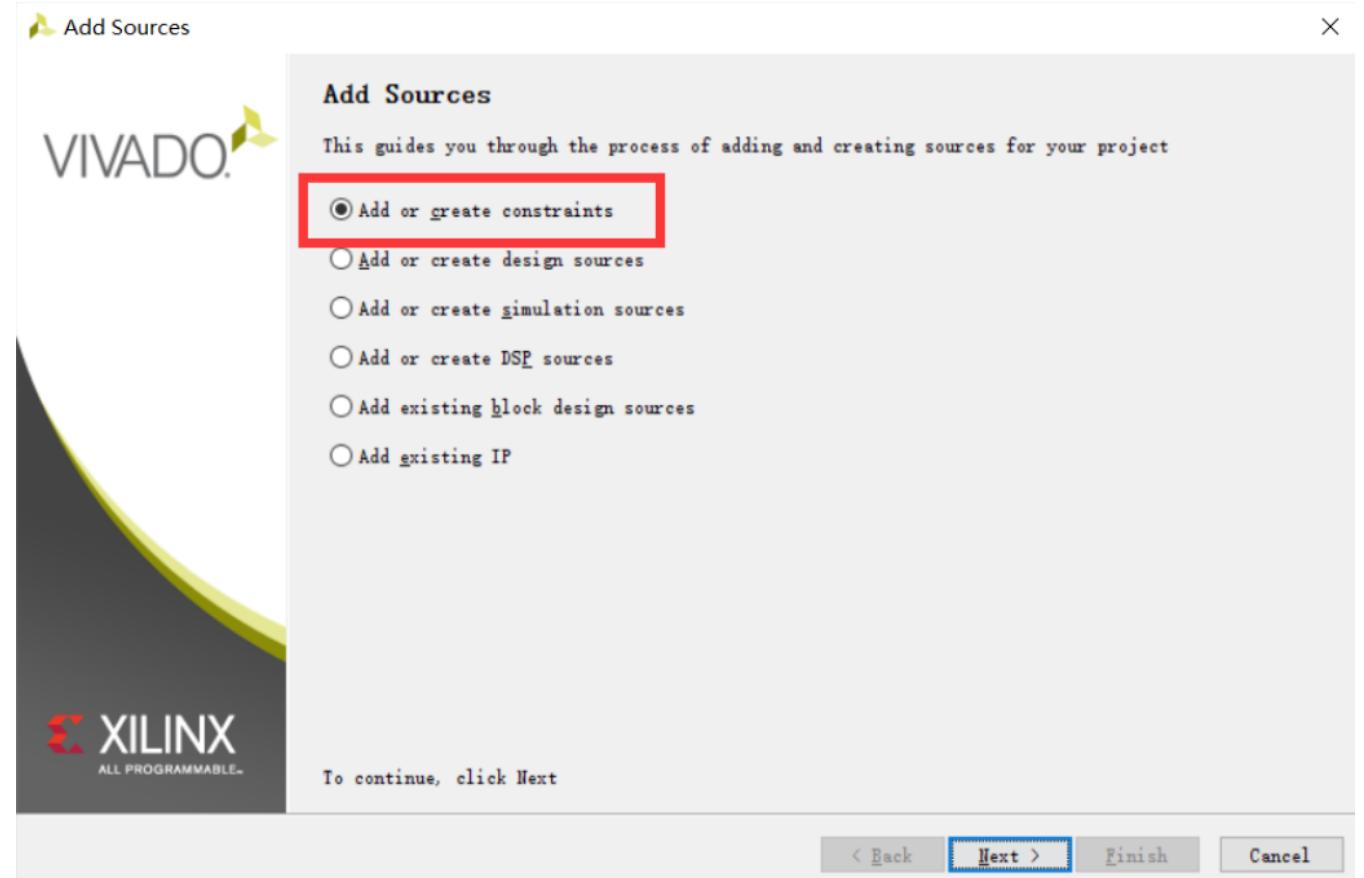
此时，在 Sources 下 Constraints 中会找到新建的 XDC 文件。



约束文件

2. 可以直接新建XDC的约束文件，手动输入约束命令

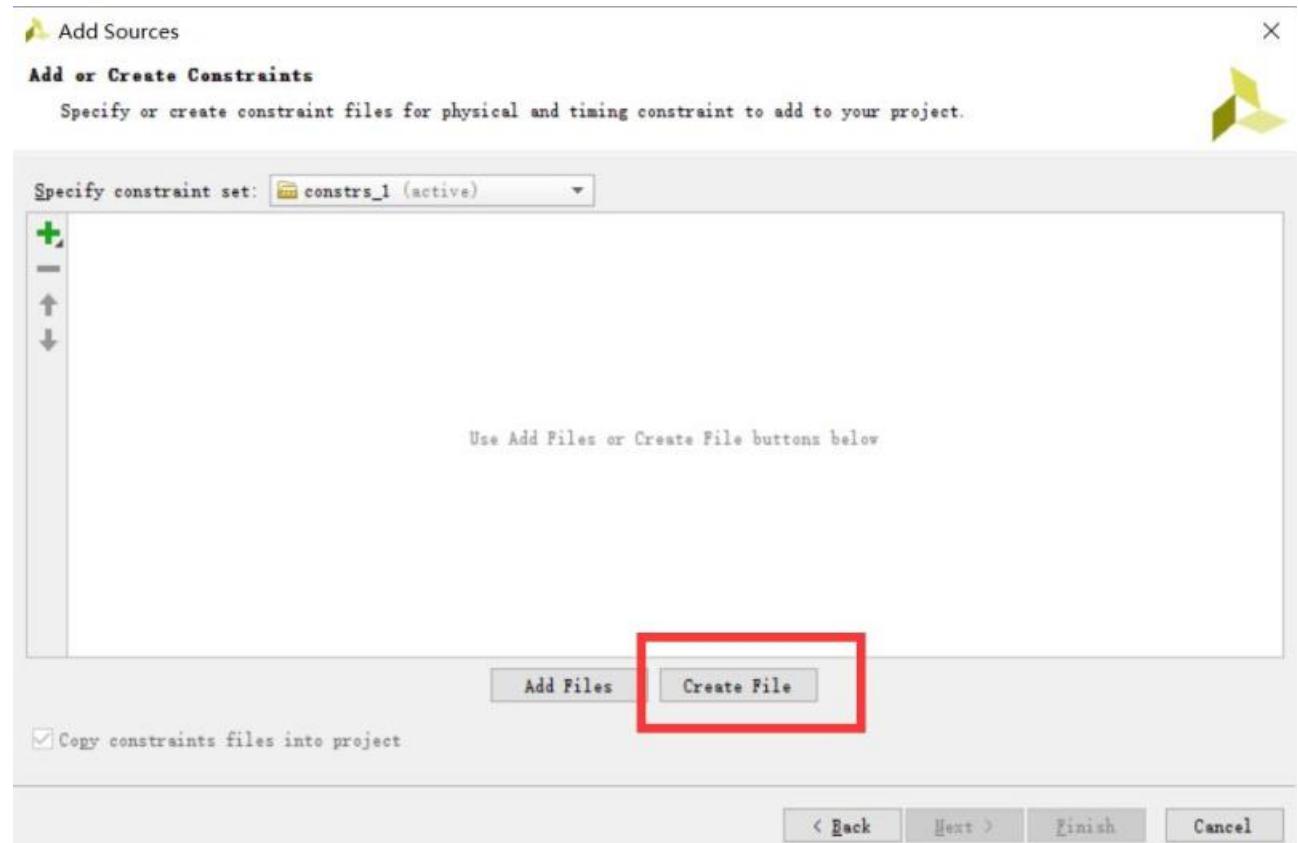
点击 Add Sources，选择第一项 Add or Create Constraints 一项，点击 Next。



约束文件

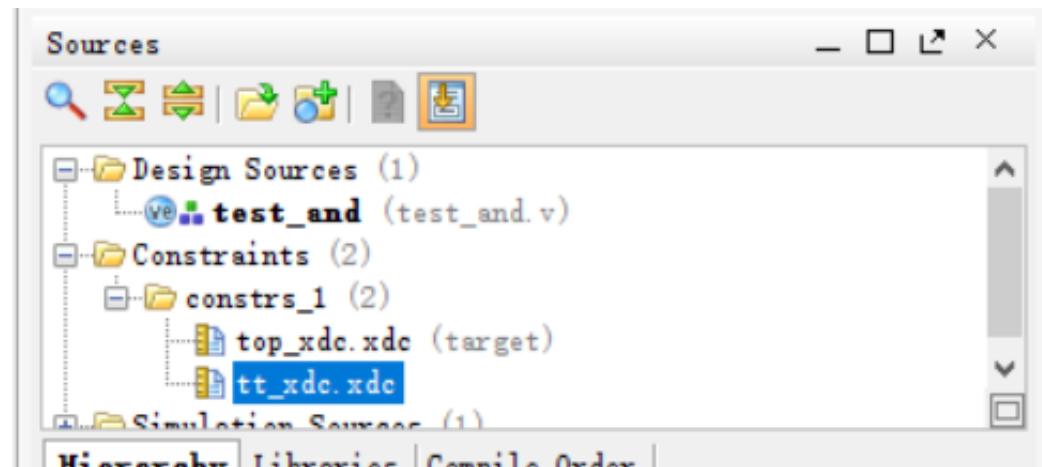
2. 可以直接新建XDC的约束文件，手动输入约束命令

点击 Create File，新建一个 XDC 文件，输入 XDC 文件名，点击 OK。
点击 Finish。



约束文件

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。



```
set_property PACKAGE_PIN L15 [get_ports C]
set_property PACKAGE_PIN Y9 [get_ports A]
set_property PACKAGE_PIN W9 [get_ports B]
set_property IOSTANDARD LVCMOS33 [get_ports A]
set_property IOSTANDARD LVCMOS33 [get_ports B]
set_property IOSTANDARD LVCMOS33 [get_ports C]
```

使用vivado烧写程序

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。



使用vivado烧写程序

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

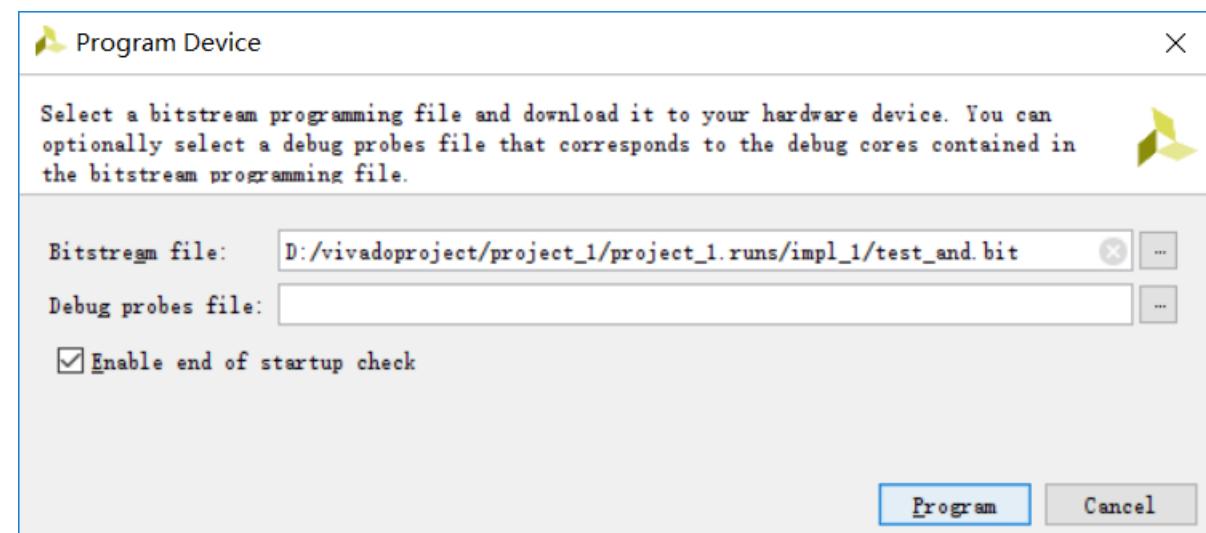
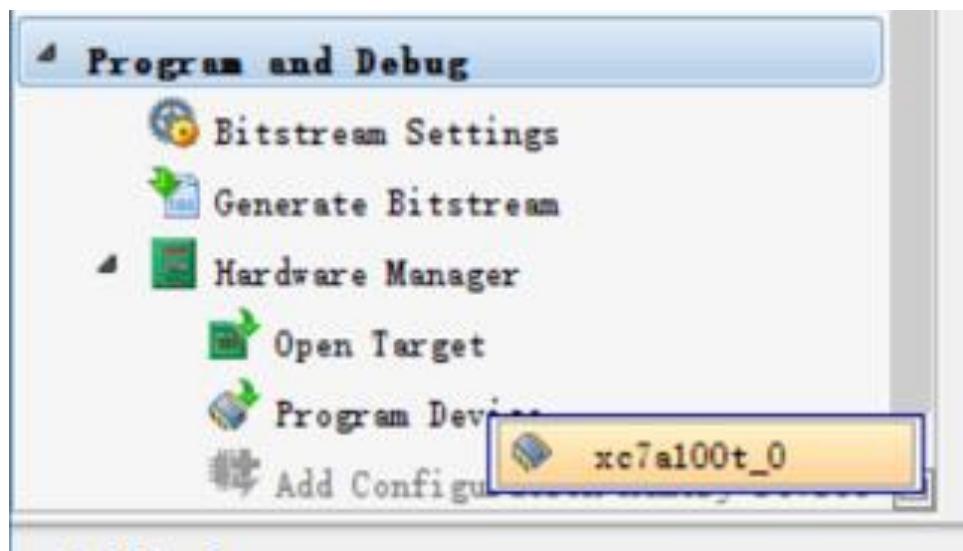


在提示的信息中，选择 Open Hardware Manager（或在 Flow Navigator 中展开 Hardware Manager，点击 Open Target）。选择Auto Connect连接到板卡。



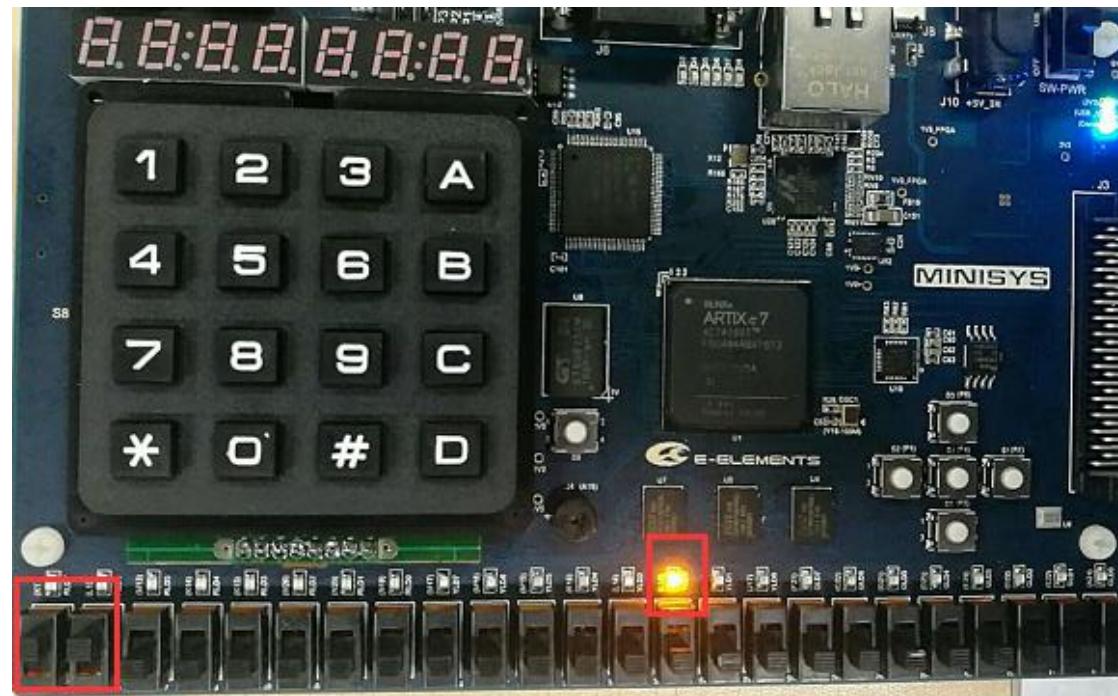
使用vivado烧写程序

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。



使用vivado烧写程序

烧入完成后，可以拨动SW23和SW22，来看YLD2上的值，SW23和SW22作为两个输入，已经构成一个与门，将结果展示在YLD2上。



练习

- 用门电路表示一个两位二进制无符号数相乘的逻辑电路，列出真值表
- 写出逻辑表达式，使用verilog写出函数，利用vivado进行仿真
- 输出这两个二进制数分别为00, 01, 10, 11时的波形图结果
- 功能验证正确后，将该乘法器烧入Minisys板子中，并验证结果是否正确
(以上为课堂练习,不需要提交)

IP核

在集成电路的可重用设计方法学中，IP核，全称知识产权核（英语：intellectual property core），是指某一方提供的、形式为逻辑单元、芯片设计的可重用模块。

后面的示例以使用Verilog HDL语言的数据流描述方法设计一个数据宽度可在1~32之间变化，输入端口可在2~8之间变化的与门andgate，输入端8个，分别是a,b,c,d,e,f,g,h，输出端为q。将该与门封装成可配置输入端口数和数据宽度的“与门”IP核。

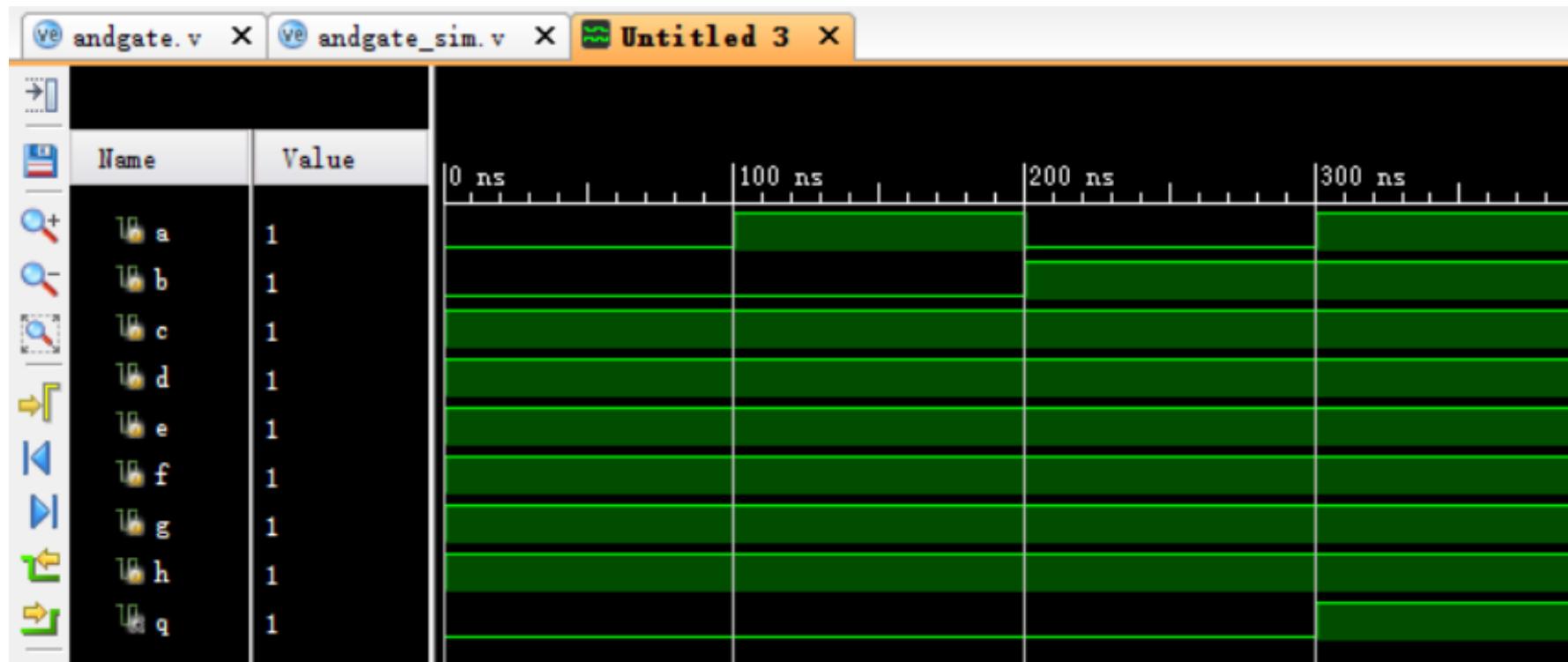
与门IP核(andgate.v)

```
module andgate
#(parameter Port_Num = 2, // 指定缺省的输入是 2 个输入端口
parameter WIDTH=8) // 指定数据宽度参数， 缺省值是 8
(
    input [(WIDTH-1):0] a,
    input [(WIDTH-1):0] b,
    input [(WIDTH-1):0] c,
    input [(WIDTH-1):0] d,
    input [(WIDTH-1):0] e,
    input [(WIDTH-1):0] f,
    input [(WIDTH-1):0] g,
    input [(WIDTH-1):0] h,
    output [(WIDTH-1):0] q
);
    assign q = (a & b & c & d & e & f & g & h);
endmodule
```

仿真1位8输入的结果

```
`timescale 1ns / 1ps
module
andgate_sim();
// input
reg a=0;
reg b=0;
reg c=1;
reg d=1;
reg e=1;
reg f=1;
reg g=1;
reg h=1;
//output
wire q;
// 实例化与门的时候，设定宽度为 1
andgate #(8,1) u(.a(a),.b(b),.c(c),.d(d),
.e(e),.f(f),.g(g),.h(h),.q(q));
initial begin
#100 a=1;
#100 begin a=0;b=1;end
#100 a=1;
end
endmodule
```

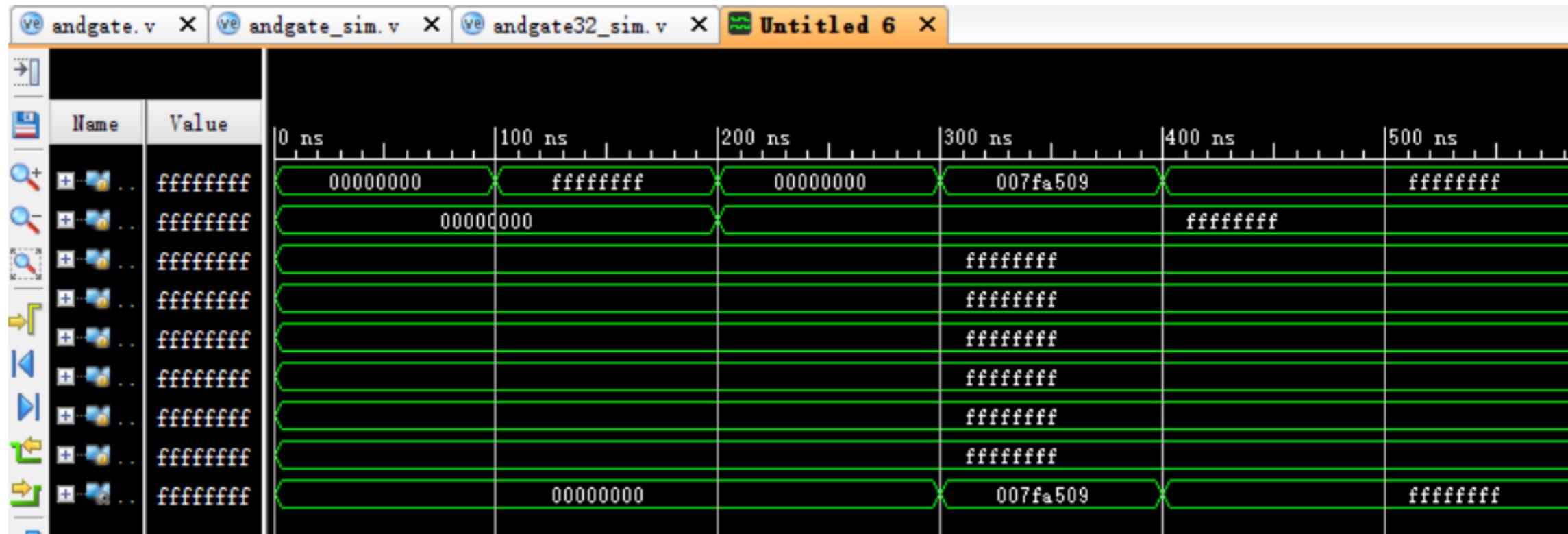
验证仿真结果



仿真32位二输入与门的结果

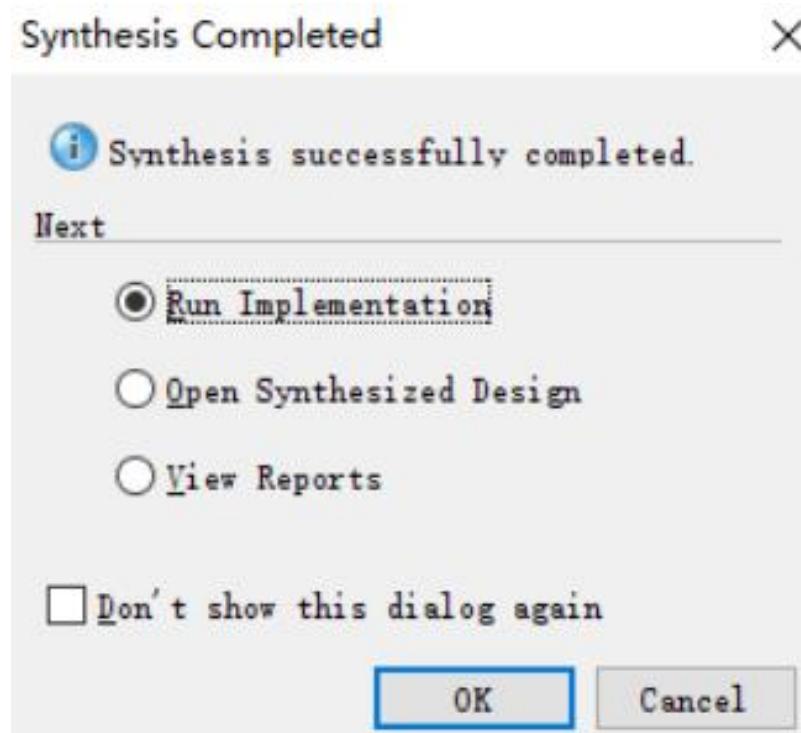
```
module andgate32_sim();
// input
reg [31:0] a=32'h00000000;
reg [31:0] b=32'h00000000;
reg [31:0] c=32'hffffffff;
reg [31:0] d=32'hffffffff;
reg [31:0] e=32'hffffffff;
reg [31:0] f=32'hffffffff;
reg [31:0] g=32'hffffffff;
reg [31:0] h=32'hffffffff;
//outbut
wire [31:0] q;
// 实例化与门的时候，设定宽度为 32
andgate #(8,32) u(.a(a),.b(b),.c(c),.d(d),
.e(e),.f(f),.g(g),.h(h),.q(q));
initial begin
#100 a=32'hffffffff;
#100 begin a=32'h00000000;b=32'hffffffff;end
#100 a = 32'h007fa509;
#100 a=32'hffffffff;
end
endmodule
```

验证仿真结果



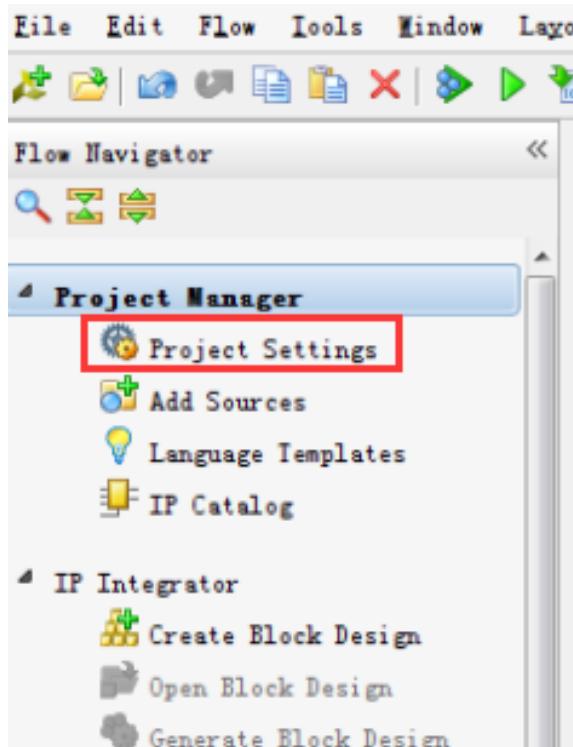
综合并封装IP核

仿真正确的 andgate 模块进行综合（参考前面章节的步骤），综合结束后会出现下图的对话框，选择 Cancel。



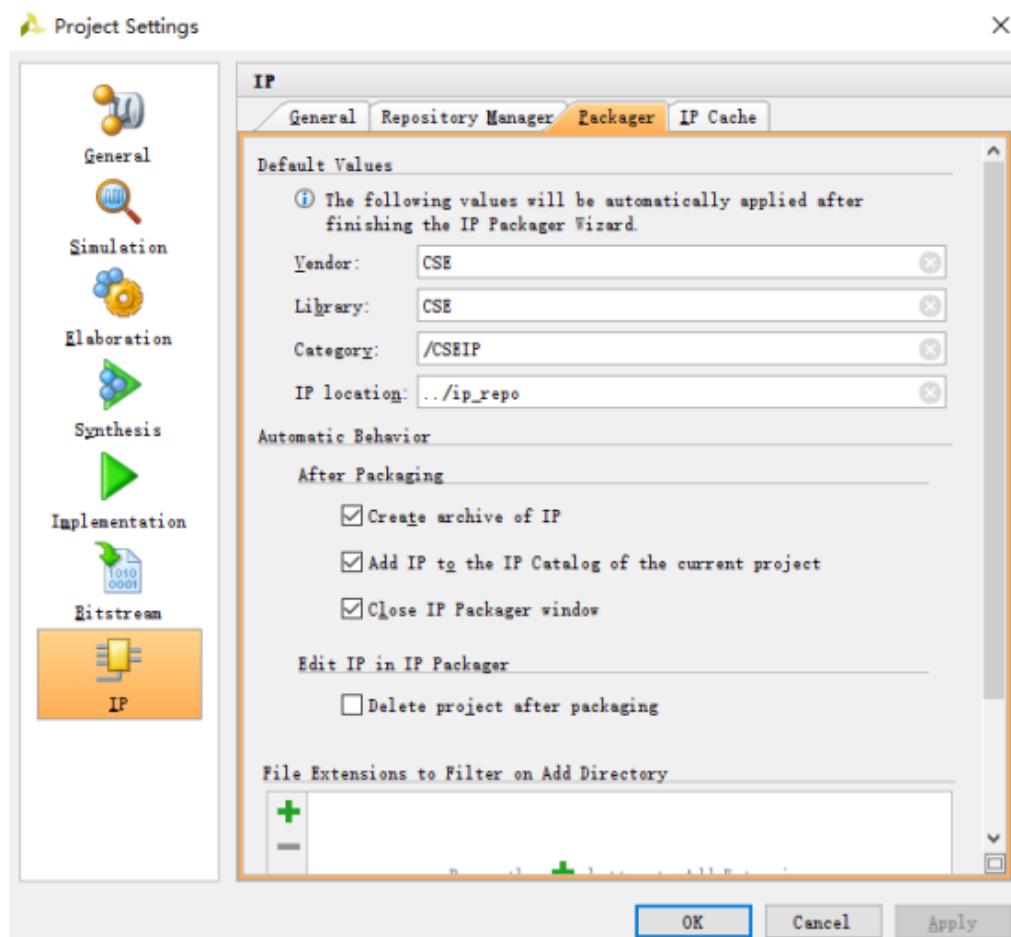
综合并封装IP核

在下图所示的的界面中点击 Project Settings。



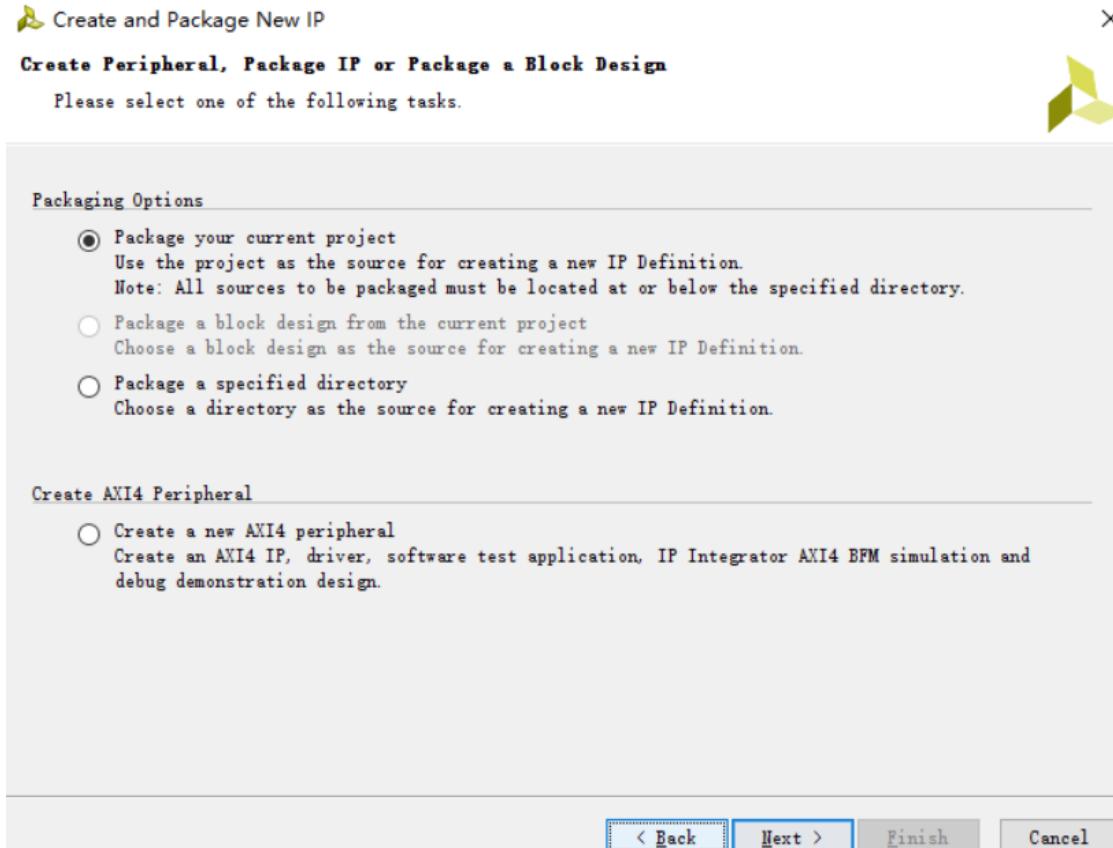
综合并封装IP核

在 Project Settings 对话框中选择 IP，并进入 Packager 选项卡，如下图所示进行设置。设置好后，点击 Apply，然后点击 OK。记住这里设置的各个属性。



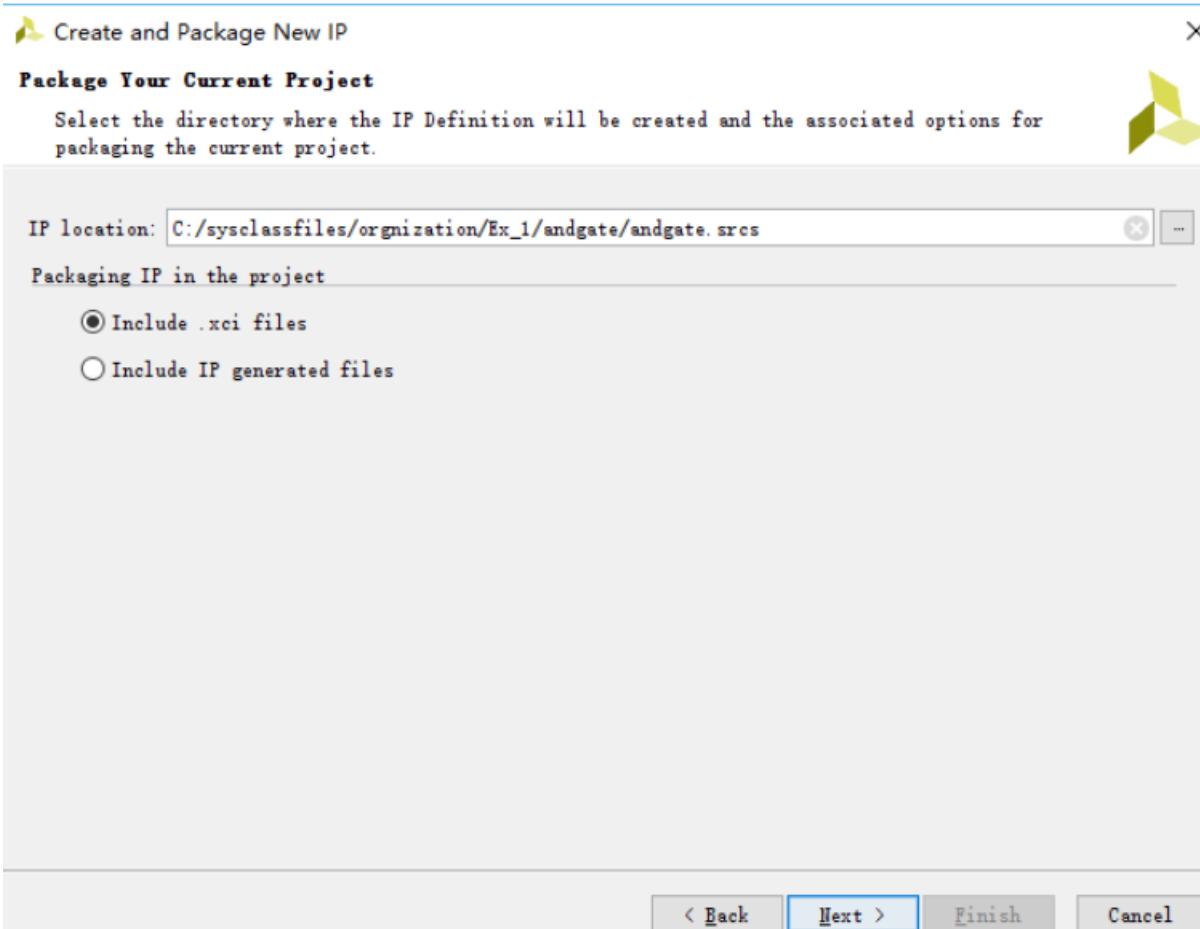
综合并封装IP核

在 Vivado 的菜单栏中选择 Tools->Create and Package IP。在弹出的窗口中点击 Next。在之后弹出的窗口中如下图所示设置封装选项。点击 Next。



综合并封装IP核

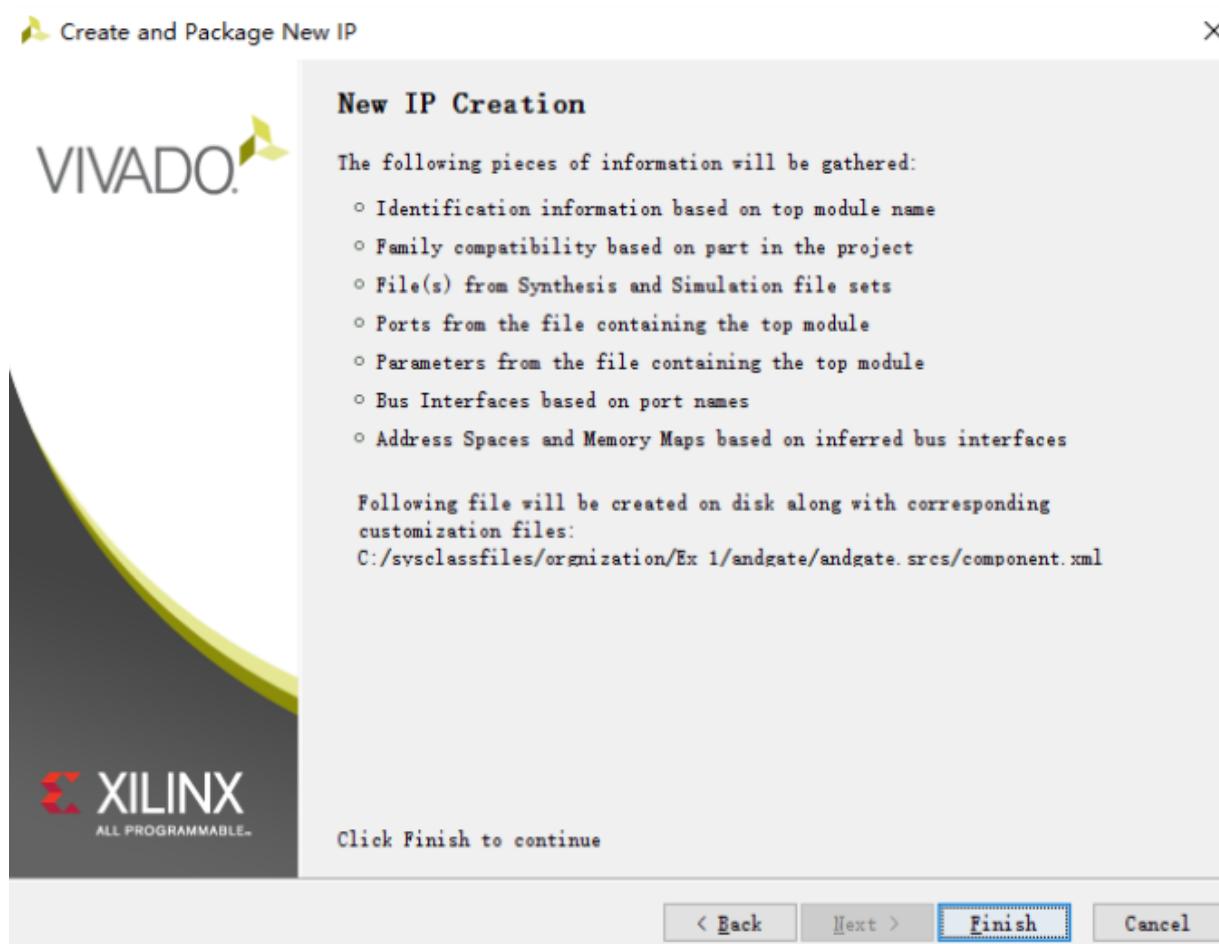
在 IP Location 中我们不做修改，点击 Next。



不过我们知道了封装后的 IP 放在了
C:/sysclassfiles/organization/Ex_1/andgate/andgate.srcs
这个文件夹中。

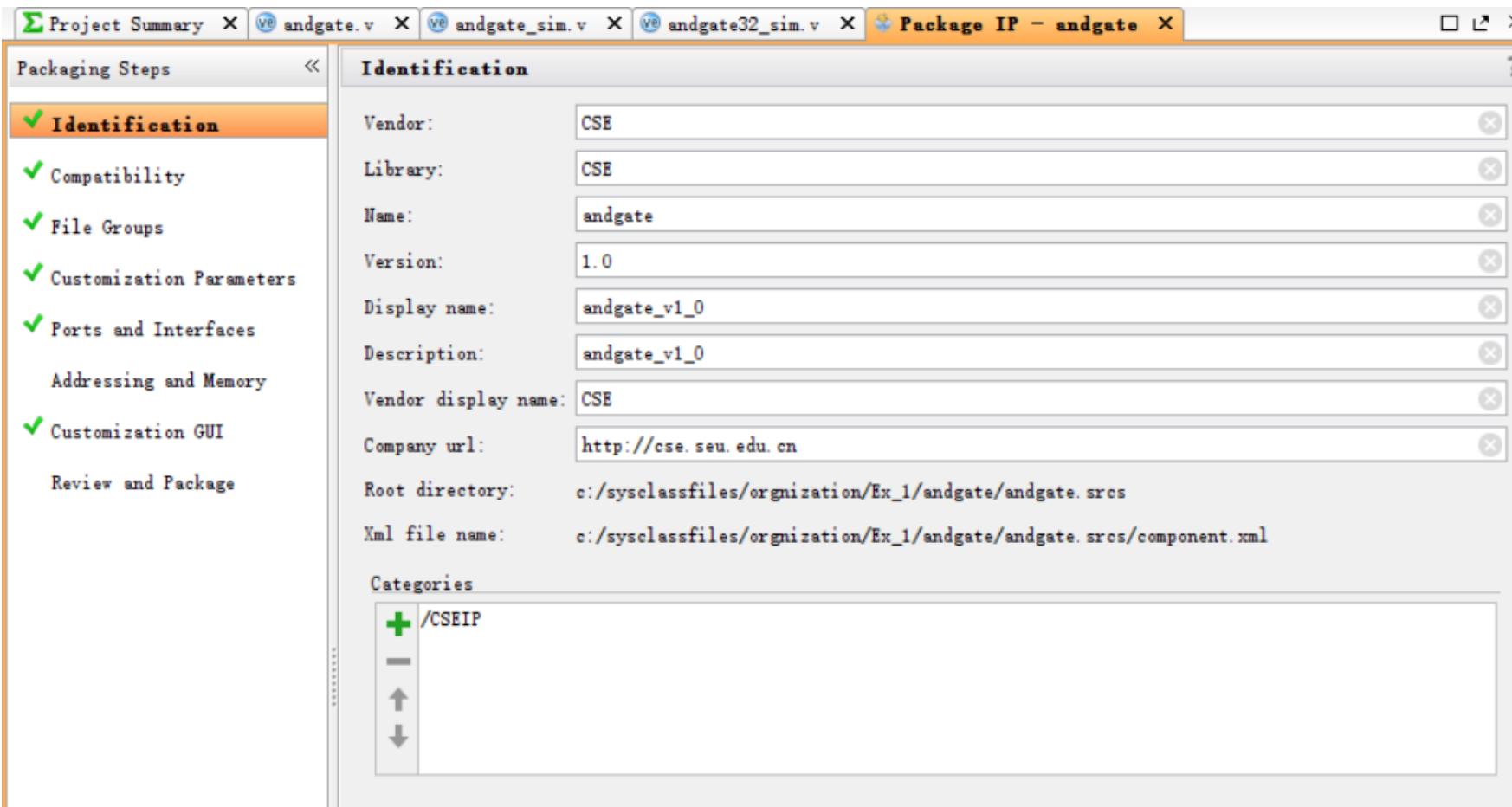
综合并封装IP核

在下图所示的对话框中点击 Finish。



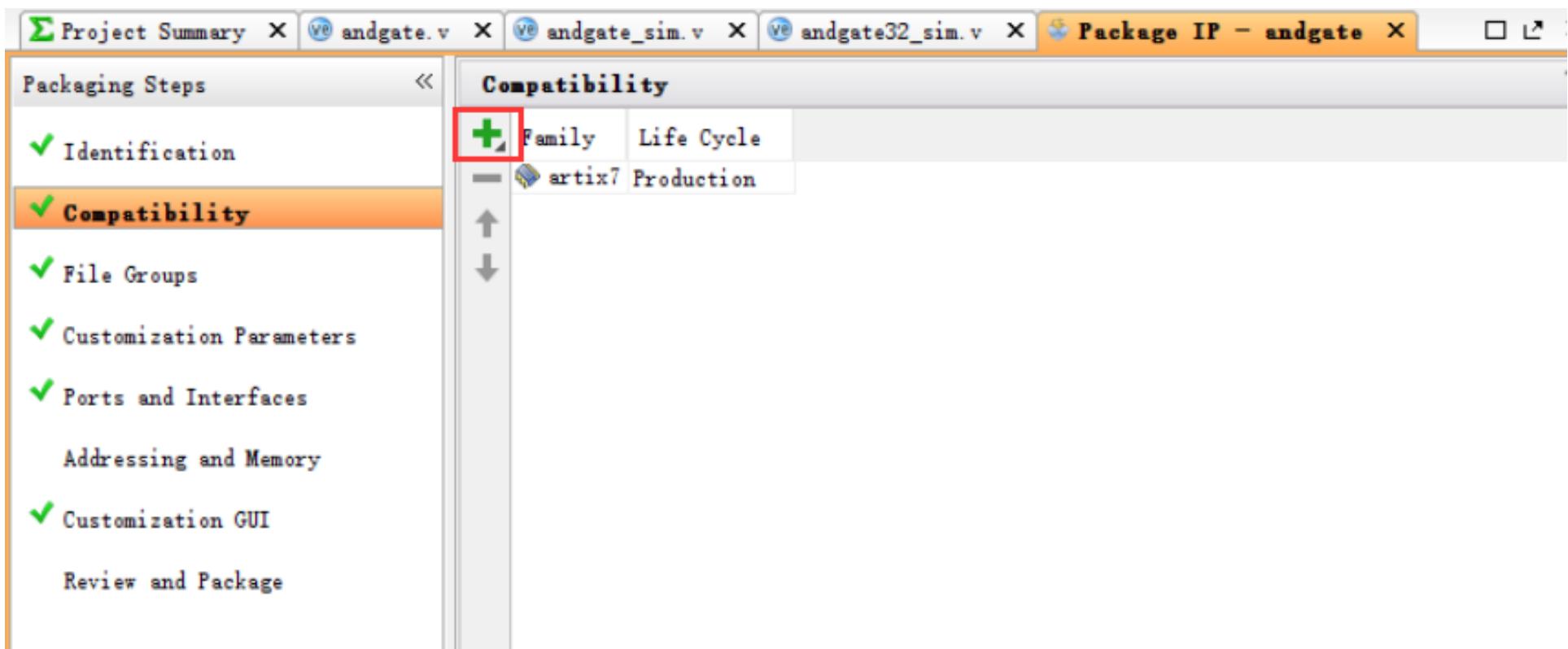
综合并封装IP核

这时可以看到如下图所示的 IP 封装设置。按下图那样设置好各个项目。



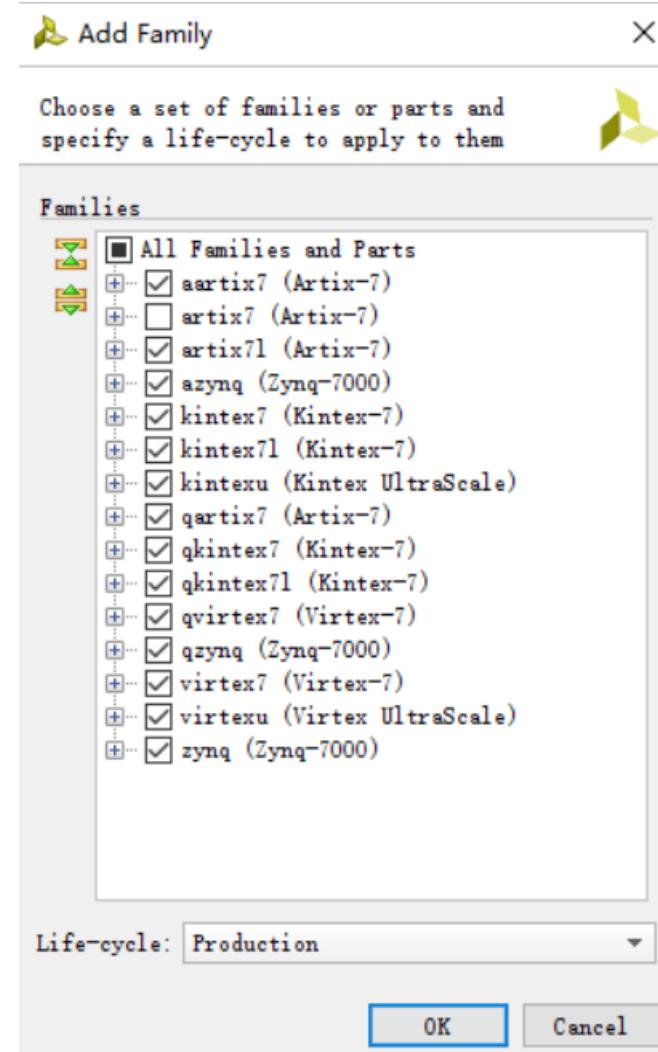
综合并封装IP核

在下图中我们可以添加 IP 核所支持的芯片家族。点击+并选择 Add Family Explicitly



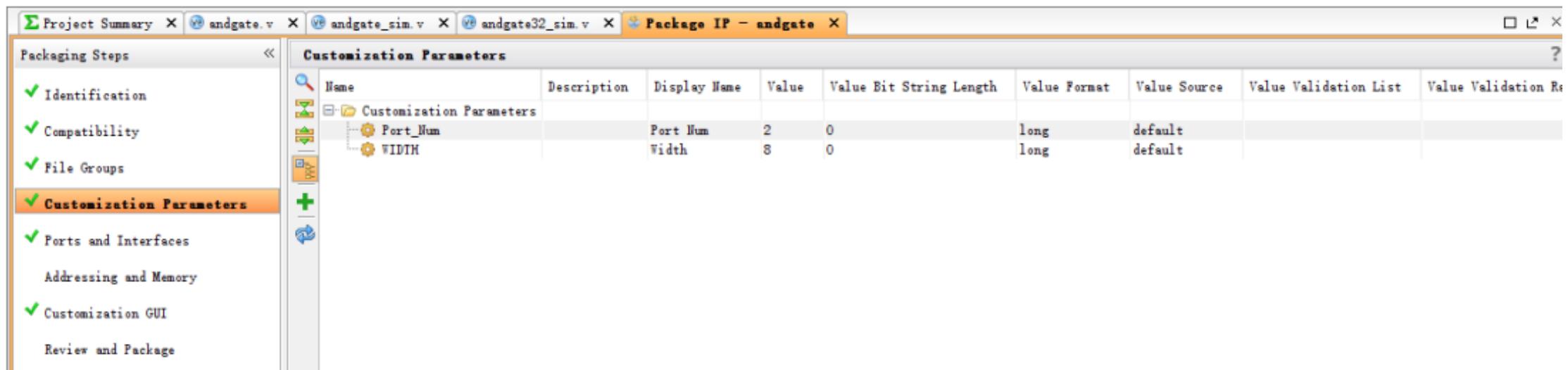
综合并封装IP核

在 Add Family 对话框中选中除 artix7 之外的所有芯片家族（因为 artix7 系列已经有了），Life-cycle 选择 Production，然后点击 OK。这样就设置完了 compatibility。



综合并封装IP核

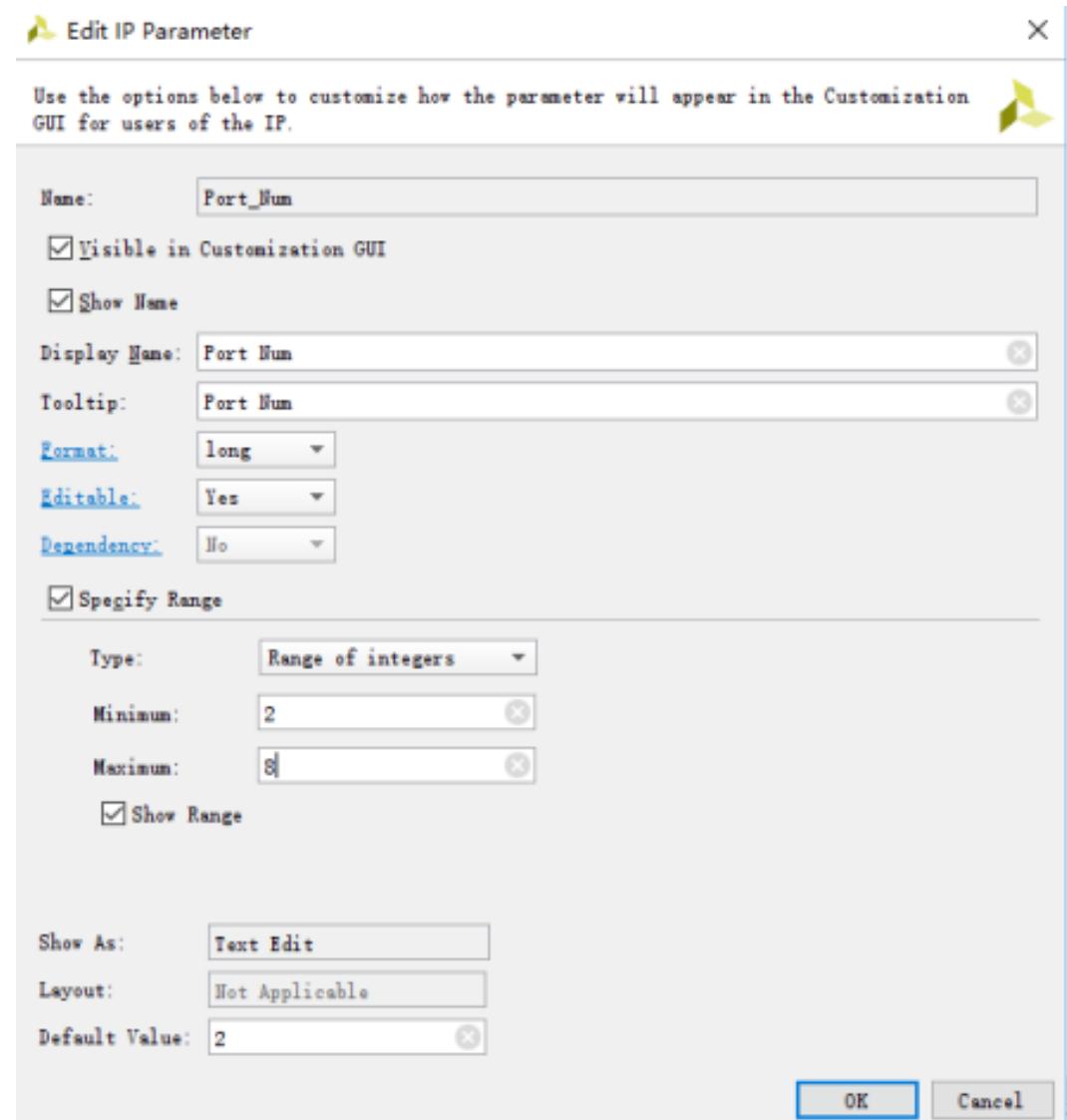
接下来我们到 Customization Parameters



双击上图中的Port_Num得到下一页PPT所示的 IP和参数的对话框，按照下一页PPT那样设置 Port_Num 参数厚点 OK。

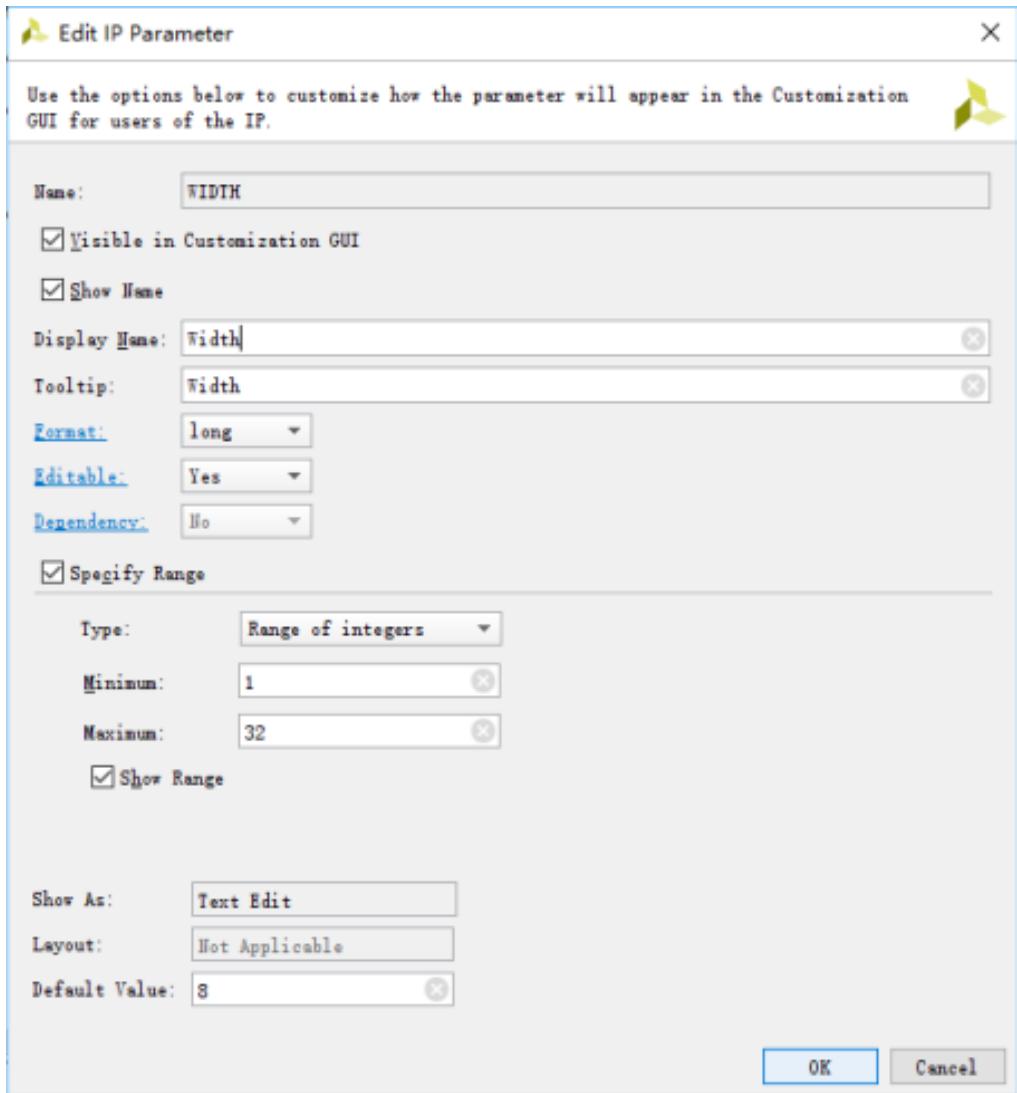
综合并封装IP核

从图中我们可以看到，IP 核至少有两个输入端，最多可以有 8 个输入端。



综合并封装IP核

从图中可以看到，数据位宽 WIDTH 最小是 1 位，最大是 32 位。



综合并封装IP核

接下来到 Ports and Interfaces。由于我们设计的数据输入端最小是 2 个，因此，a,b 两个输入端时钟都是 Enable 的。而 c~h 输入端则要根据 Port_Num 的值决定是否 Enable。

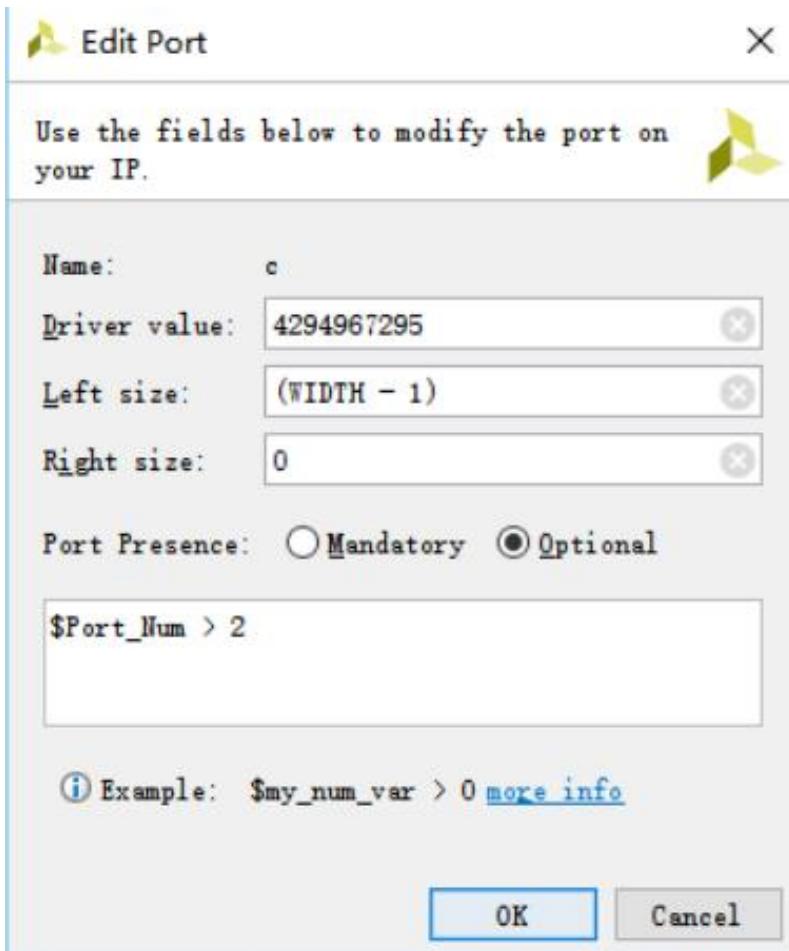
The screenshot shows the Quartus II software interface with the following details:

- Project Summary**: Contains files: andgate.v, andgate_sim.v, andgate32_sim.v, andgate.
- Packaging Steps**:
 - ✓ Identification
 - ✓ Compatibility
 - ✓ File Groups
 - ✓ Customization Parameters
 - Ports and Interfaces** (highlighted)
 - Addressing and Memory
- Ports and Interfaces** table:

Name	Interface Mode	Enablement Dependency	Is Declaration	Direction	Driver Value	Size Left	Size Right	Size Left Dependency	Size Right Dependency	Type Name
a			<input type="checkbox"/>	in		7	0 (WIDTH - 1)			std_logic_vector
b			<input type="checkbox"/>	in		7	0 (WIDTH - 1)			std_logic_vector
c			<input type="checkbox"/>	in		7	0 (WIDTH - 1)			std_logic_vector
d			<input type="checkbox"/>	in		7	0 (WIDTH - 1)			std_logic_vector
e			<input type="checkbox"/>	in		7	0 (WIDTH - 1)			std_logic_vector
f			<input type="checkbox"/>	in		7	0 (WIDTH - 1)			std_logic_vector
g			<input type="checkbox"/>	in		7	0 (WIDTH - 1)			std_logic_vector
h			<input type="checkbox"/>	in		7	0 (WIDTH - 1)			std_logic_vector
q			<input type="checkbox"/>	out		7	0 (WIDTH - 1)			std_logic_vector

综合并封装IP核

请双击端口 c，在打开的对话框中，按照下图所示设置。



Driver value 是指端口 c 在 disable 时候的取值。在 andgate 模块中，共定义了 8 个输入端口，但实际应用中允许只用 2~8 个中的任意多个输入端口，不用的端口需要设置成 disable 状态。在 IP 核中，所谓 disable 的端口实际上只是不提供给外部接口输入，但在模块中该端口依然存在，所以如果必要，需要对它赋一个驱动值。由于制作的是“与门”，因此所有不接外部接口的 disable 输入端应该赋值为全 1，也就是 16 进制的 FFFFFFFF，十进制的 4294967295，因此在这里给出的赋值是 4294967295。

Port Presence（端口存在），选择的是 Optional 说明该端口的存在是有条件的，在下面的启动表达式编辑框中，输入 \$Port_Num > 2，表明当输入端口大于 2 的时候，c 端口被启用（Enable）。设置好后点击 OK。

综合并封装IP核

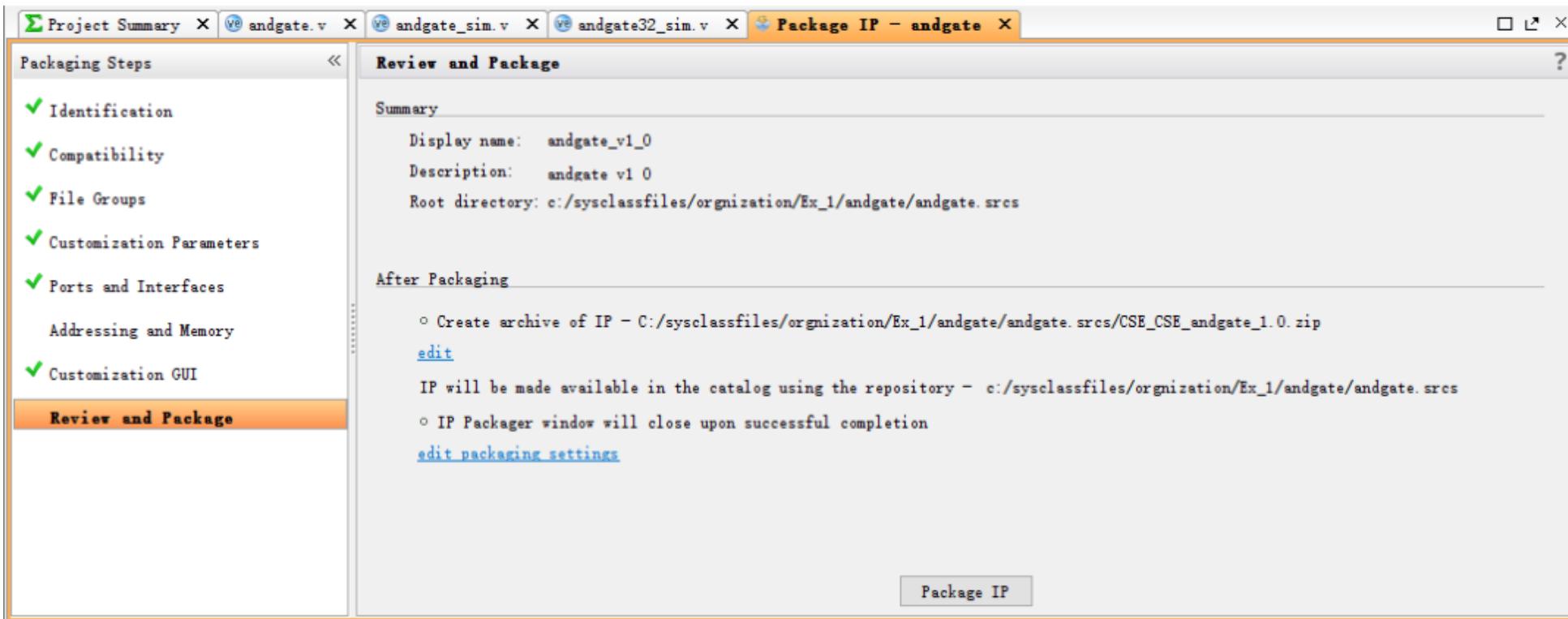
请大家按照上述步骤设置好端口 d,e,f,g,h 的参数。

The screenshot shows the Quartus Project Manager interface with the title bar "Project Summary X Package IP - andgate X". On the left, a sidebar lists packaging steps: Identification, Compatibility, File Groups, Customization Parameters, Ports and Interfaces (which is selected and highlighted in orange), Addressing and Memory, Customization GUI, and Review and Package. The main area displays a table titled "Ports and Interfaces" with the following data:

Name	Interface Mode	Enablement Dependency	Is Declaration	Direction	Driver Value	Size Left	Size Right	Size Left Dependency	Size Right Dependency	Type Name
a				in		7	0 (WIDTH - 1)			std_logic_vector
b				in		7	0 (WIDTH - 1)			std_logic_vector
c		\$Port_Num > 2		in	4294967295	7	0 (WIDTH - 1)			std_logic_vector
d		\$Port_Num > 3		in	4294967295	7	0 (WIDTH - 1)			std_logic_vector
e		\$Port_Num > 4		in	4294967295	7	0 (WIDTH - 1)			std_logic_vector
f		\$Port_Num > 5		in	4294967295	7	0 (WIDTH - 1)			std_logic_vector
g		\$Port_Num > 6		in	4294967295	7	0 (WIDTH - 1)			std_logic_vector
h		\$Port_Num > 7		in	4294967295	7	0 (WIDTH - 1)			std_logic_vector
q				out		7	0 (WIDTH - 1)			std_logic_vector

综合并封装IP核

大家可以到 Customization GUI 验证一下参数的变化对 IP 封装的影响。
接下来到 Review and Packaging，如下图所示。



综合并封装IP核

我们可以看到 IP 生成到一个称为 CSE_CSE_andgate_1.0.zip 文件中，路径是

C:/sysclassfiles/organization/Ex_1/andgate/andgate.srcs。

点击 Package IP 或 Re-Package IP。这样，andgate 的 IP 核就生成了。

为了方便今后的使用，请大家在 C:\sysclassfiles\orgnization\路径下新建一个文件夹IPCore，并将

C:\sysclassfiles\orgnization\Ex_1\andgate\andgate.srcs\CSE_CSE_andgate_1.0.zip
文件拷贝到 C:\sysclassfiles\orgnization\IPCore 中，并将其解压缩。

课后练习

参考上述步骤，完成以下IP核的封装

门电路	模块名	参数			
		Port_Num	WIDTH	输入	输出
或门	orgate	2~8	1~32	a,b,c,d,e,f,g,h	q
非门	notgate	-	1~32	a	c
与非门	nandgate	2~8	1~32	a,b,c,d,e,f,g,h	q
或非门	norgate	2~8	1~32	a,b,c,d,e,f,g,h	q
异或门	xorgate	2~8	1~32	a,b,c,d,e,f,g,h	q
异或非门	nxorgate	2~8	1~32	a,b,c,d,e,f,g,h	q

表中的各个门电路的 IP 核封装好后，统一拷贝到
C:\sysclassfiles\organization\IPCore中，并分别解压这些.zip 文件