



北京理工大学  
BEIJING INSTITUTE OF TECHNOLOGY

# 开发板简介与Vivado下板

主讲人：杨风帆

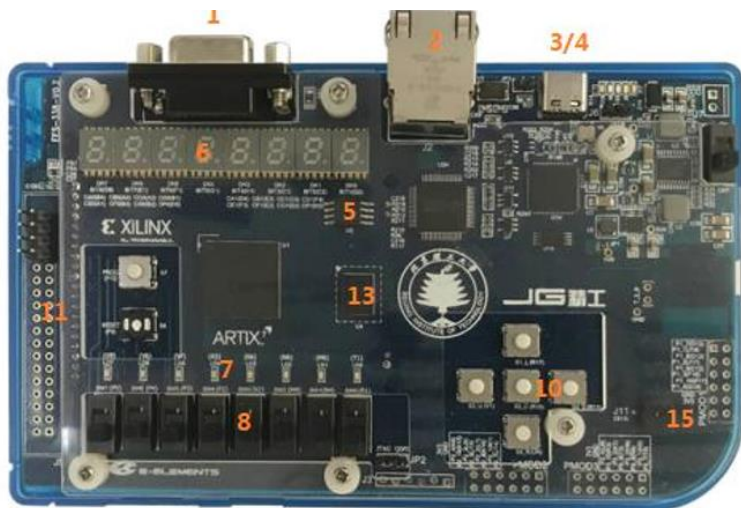


北京理工大学  
BEIJING INSTITUTE OF TECHNOLOGY



# 开发板简介

## 开发板介绍 EES-338口袋计算机



EES-338 是依元素科技基于 Xilinx Artix-7 FPGA 研发的便携式数模混合基础教学平台。

EES-338 配备的 FPGA 具有大容量高性能等特点，能实现较复杂的数字逻辑设计。该平台拥有丰富的外设，以及灵活的通用扩展接口。



编号	描述	编号	描述
1	VGA 接口	10	5 个按键
2	以太网口	11	通用拓展 IO
3	USB 转 UART 接口	12	蓝牙模块
4	JTAG 接口	13	SRAM 存储器
5	SPI FLASH 存储器	14	1 个 LCD 显示屏
6	8 个数码管	15	1 个蜂鸣器
7	8 个 LED 灯		
8	8 个拨码开关		
9	6 个游戏手柄按键		

EES-338 采用 Xilinx Artix-7 系列 XC7A35T-1CSG324C FPGA，其资源如下：

	Part Number	XC7A12T	XC7A15T	XC7A25T	XC7A35T
Logic Resources	Logic Cells	12,800	16,640	23,360	33,280
	Slices	2,000	2,600	3,650	5,200
	CLB Flip-Flops	16,000	20,800	29,200	41,600
Memory Resources	Maximum Distributed RAM (Kb)	171	200	313	400
	Block RAM/FIFO w/ ECC (36 Kb each)	20	25	45	50
	Total Block RAM (Kb)	720	900	1,620	1,800
Clock Resources	CMTs (1 MMCM + 1 PLL)	3	5	3	5
I/O Resources	Maximum Single-Ended I/O	150	250	150	250
	Maximum Differential I/O Pairs	72	120	72	120
Embedded Hard IP Resources	DSP Slices	40	45	80	90
	PCIe® Gen2 <sup>(1)</sup>	1	1	1	1
	Analog Mixed Signal (AMS) / XADC	1	1	1	1
	Configuration AES / HMAC Blocks	1	1	1	1
	GTP Transceivers (6.6 Gb/s Max Rate) <sup>(2)</sup>	2	4	4	4
Speed Grades	Commercial	-1, -2	-1, -2	-1, -2	-1, -2
	Extended	-2L, -3	-2L, -3	-2L, -3	-2L, -3
	Industrial	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L

- 两种供电方式：Type-C接口和外接直流电源。
- 提供一个 Type-C 接口，功能分别为 USB-UART 和 USB-JTAG，这个接口可以用于为板卡供电。
- 上电成功后绿色 LED 灯（D18）点亮。

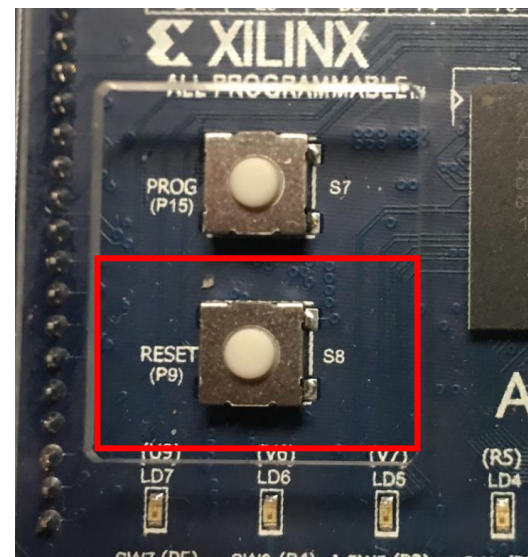
- 搭载一个 100MHz 的时钟芯片，输出的时钟信号直接与 FPGA 全局时钟输入引脚（T5）相连。

名称	原理图标号	FPGA IO PIN
时钟引脚	SYS_CLK	T5



- 两个专用按键分别用于逻辑复位 RST (S8) 和擦除 FPGA 配置 PROG (S7)，当设计中不需要外部触发复位时，RST 按键可以用作其他逻辑触发功能。没有按下时输出高电平，按下为低电平。

名称	原理图标号	FPGA IO PIN
复位引脚	FPGA_RESET	P15

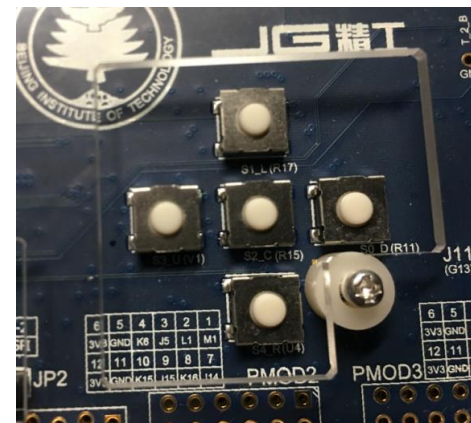




- 5个通用按键，默认为低电平，按键按下时输出高电平。

背部5个按键管脚分配如下：

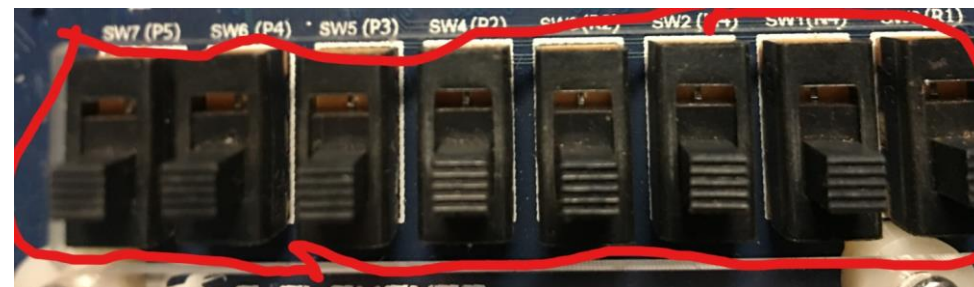
名称	原理图标号	FPGA IO PIN
S0	PB0	R11
S1	PB1	R17
S2	PB2	R15
S3	PB3	V1
S4	PB4	U4



- 开关包括 8 个拨码开关。下拨为低电平，上拨为高电平。

管脚分配如下：

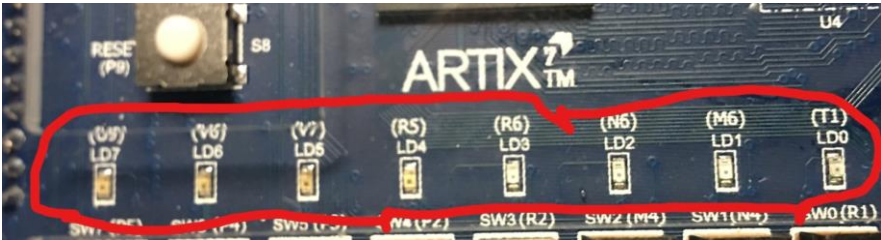
名称	原理图标号	FPGA IO PIN
SW0	SW_0	R1
SW1	SW_1	N4
SW2	SW_2	M4
SW3	SW_3	R2
SW4	SW_4	P2
SW5	SW_5	P3
SW6	SW_6	P4
SW7	SW_7	P5





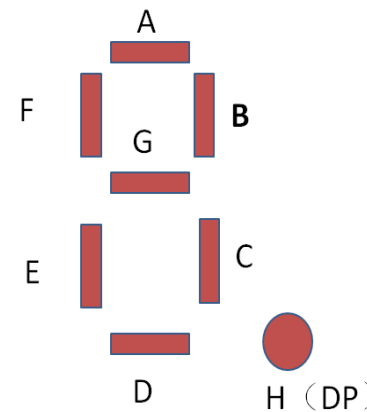
- LED 在 FPGA 输出高电平时被点亮，LED0-LED3 为绿色，LED4-LED7 为黄色。 管脚分配如下：

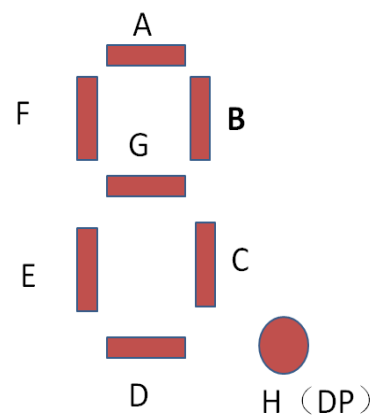
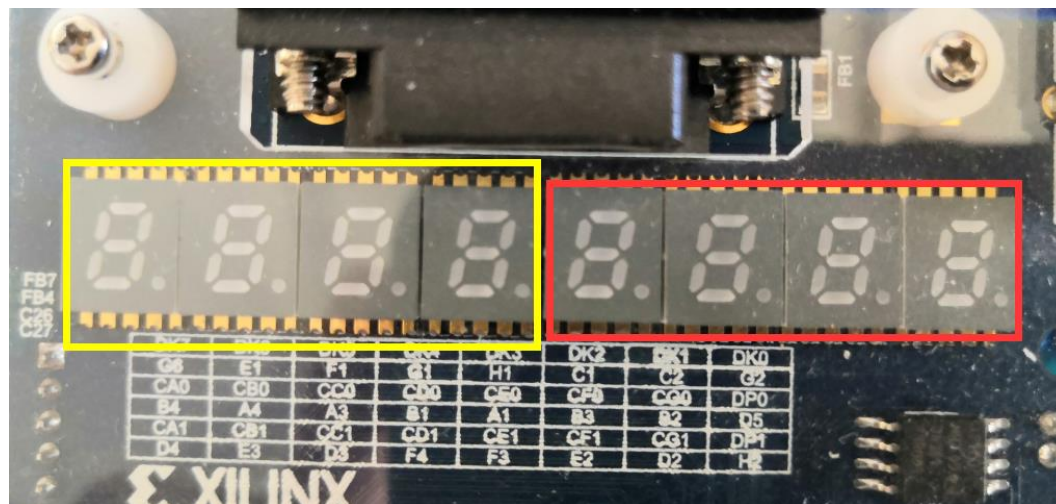
名称	原理图标号	FPGA IO PIN	颜色
D0	LED0	K2	Green
D1	LED1	J2	Green
D2	LED2	J3	Green
D3	LED3	H4	Green
D4	LED4	J4	Yellow
D5	LED5	G3	Yellow
D6	LED6	G4	Yellow
D7	LED7	F6	Yellow



- 每位数码管由7段数码管和一个小数点的LED构成
- 数码管为共阴极数码管，即公共极输入低电平。共阴极由三极管驱动，FPGA 需要提供正向信号。同时段选端连接高电平，数码管上的对应位置才可以被点亮。

- 实验所用数码管有两组
- 从左至右前4个为一组，段选端分别为DN1\_K4~DN1\_K1,LED端为A1、B1、C1、D1、E1、F1、G1、DP1
- 后4个为一组，段选端分别为DN0\_K4~DN0\_K1,LED端为A0、B0、C0、D0、E0、F0、G0、DP0
- 一个数码管中LED对应位置如右图所示





名称	原理图标号	FPGA IO PIN
A0	LED0_CA	B4
B0	LED0_CB	A4
C0	LED0_CC	A3
D0	LED0_CD	B1
E0	LED0_CE	A1
F0	LED0_CF	B3
G0	LED0_CG	B2
DP0	LED0_DP	D5
A1	LED1_CA	D4
B1	LED1_CB	E3
C1	LED1_CC	D3
D1	LED1_CD	F4
E1	LED1_CE	F3
F1	LED1_CF	E2
G1	LED1_CG	D2
DP1	LED1_DP	H2
DN0_K1	LED_BIT1	G2
DN0_K2	LED_BIT2	C2
DN0_K3	LED_BIT3	C1
DN0_K4	LED_BIT4	H1
DN1_K1	LED_BIT5	G1
DN1_K2	LED_BIT6	F1
DN1_K3	LED_BIT7	E1
DN1_K4	LED_BIT8	G6



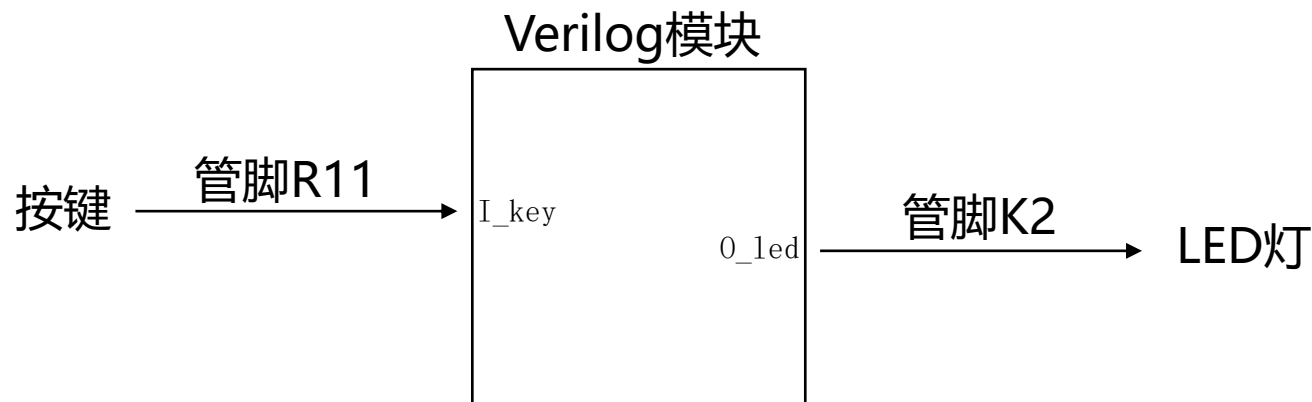
北京理工大学  
BEIJING INSTITUTE OF TECHNOLOGY

2

## 外设使用



```
1  `timescale 1ns / 1ps
2  module light_show(
3      input  I_key,  //按键输入信号
4      output O_led   //led输出信号
5  );
6      assign O_led = I_key;
7  endmodule
```



- Verilog模块中的输入输出信号与外设管脚连接，使得Verilog模块与外设之间可以进行信号的交互，实现上板效果。



北京理工大学  
BEIJING INSTITUTE OF TECHNOLOGY

3

按键去抖



- 抖动原因：按键按下或放开时，存在机械震动
- 抖动时间一般在10ms ~ 20ms



- 按键去抖动方法：延时，以避开机械震动
- 抖动检测——稳定计数——确认稳定
- 按键信号不稳定时（稳定时钟周期数不足），输出的按键信号不会发生变化

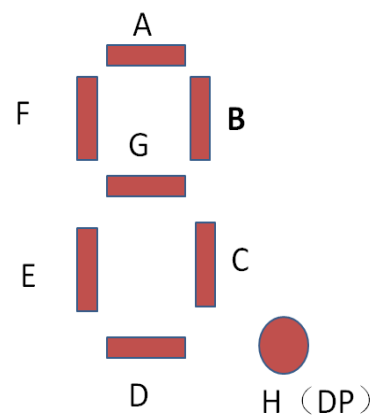
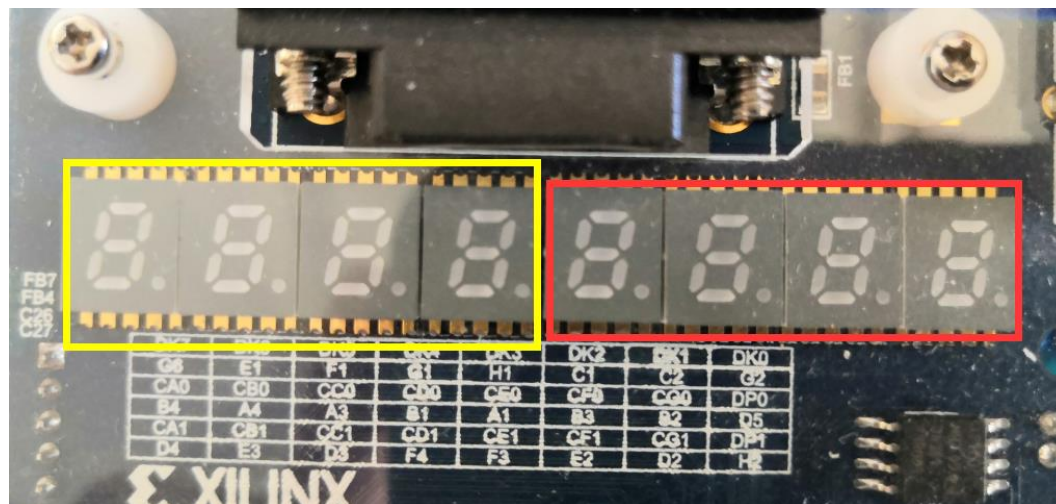




北京理工大学  
BEIJING INSTITUTE OF TECHNOLOGY

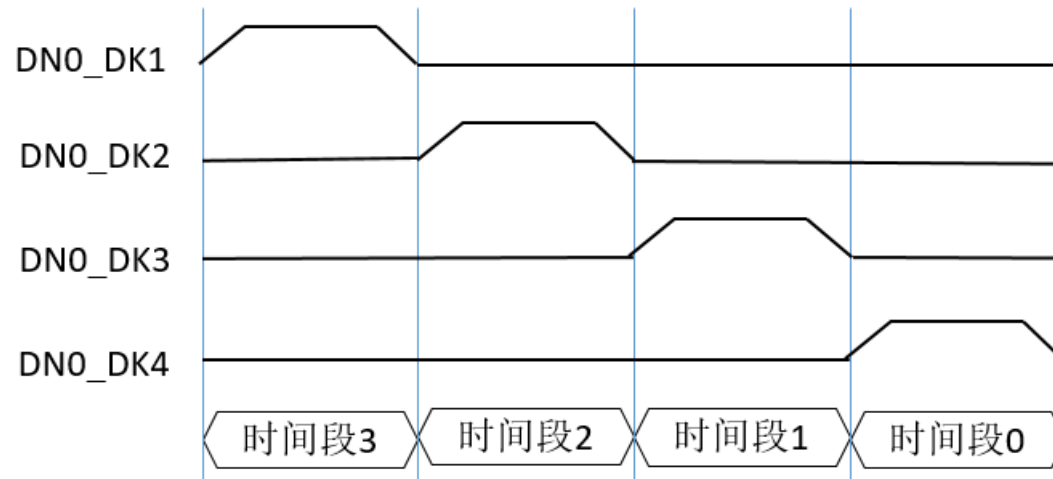
4

## 数码管显示

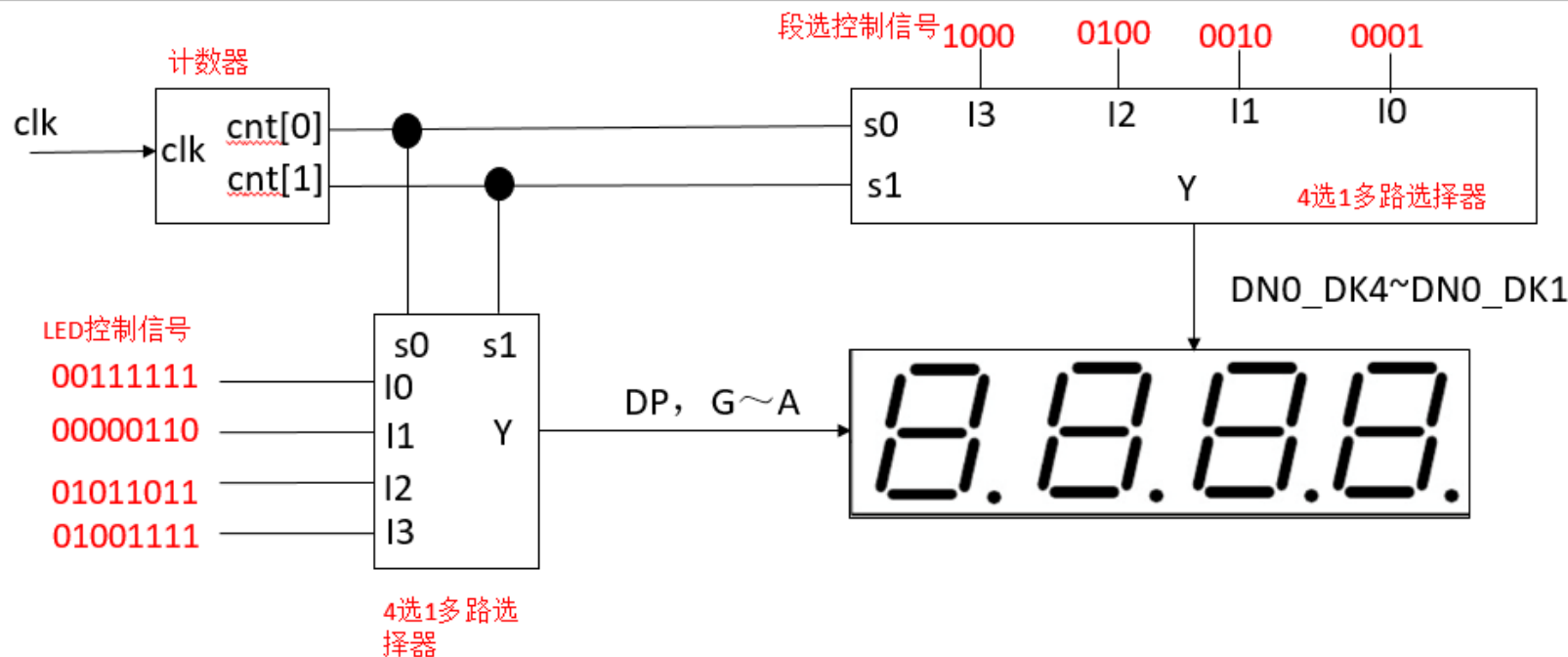


名称	原理图标号	FPGA IO PIN
A0	LED0_CA	B4
B0	LED0_CB	A4
C0	LED0_CC	A3
D0	LED0_CD	B1
E0	LED0_CE	A1
F0	LED0_CF	B3
G0	LED0_CG	B2
DP0	LED0_DP	D5
A1	LED1_CA	D4
B1	LED1_CB	E3
C1	LED1_CC	D3
D1	LED1_CD	F4
E1	LED1_CE	F3
F1	LED1_CF	E2
G1	LED1_CG	D2
DP1	LED1_DP	H2
DN0_K1	LED_BIT1	G2
DN0_K2	LED_BIT2	C2
DN0_K3	LED_BIT3	C1
DN0_K4	LED_BIT4	H1
DN1_K1	LED_BIT5	G1
DN1_K2	LED_BIT6	F1
DN1_K3	LED_BIT7	E1
DN1_K4	LED_BIT8	G6

- 显示多位数码时，利用视觉残留效果，分时扫描每位数码管对应的段选端管脚
- 以DN0\_DK4~DN0\_DK1为例







- 计数器的clk来自板载时钟（100MHz），计数器作用是对其分频后输入到数据选择器，并作为数码管扫描显示时钟
- 计数器的分频系数不能太大也不能太小



北京理工大学  
BEIJING INSTITUTE OF TECHNOLOGY

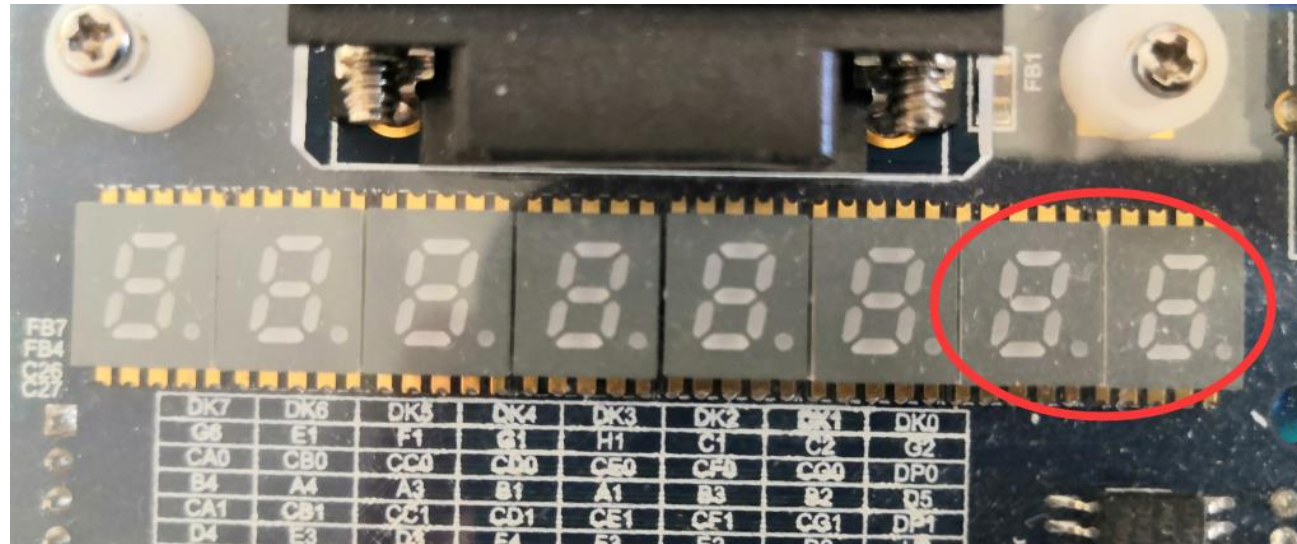
5

## Vivado 下板

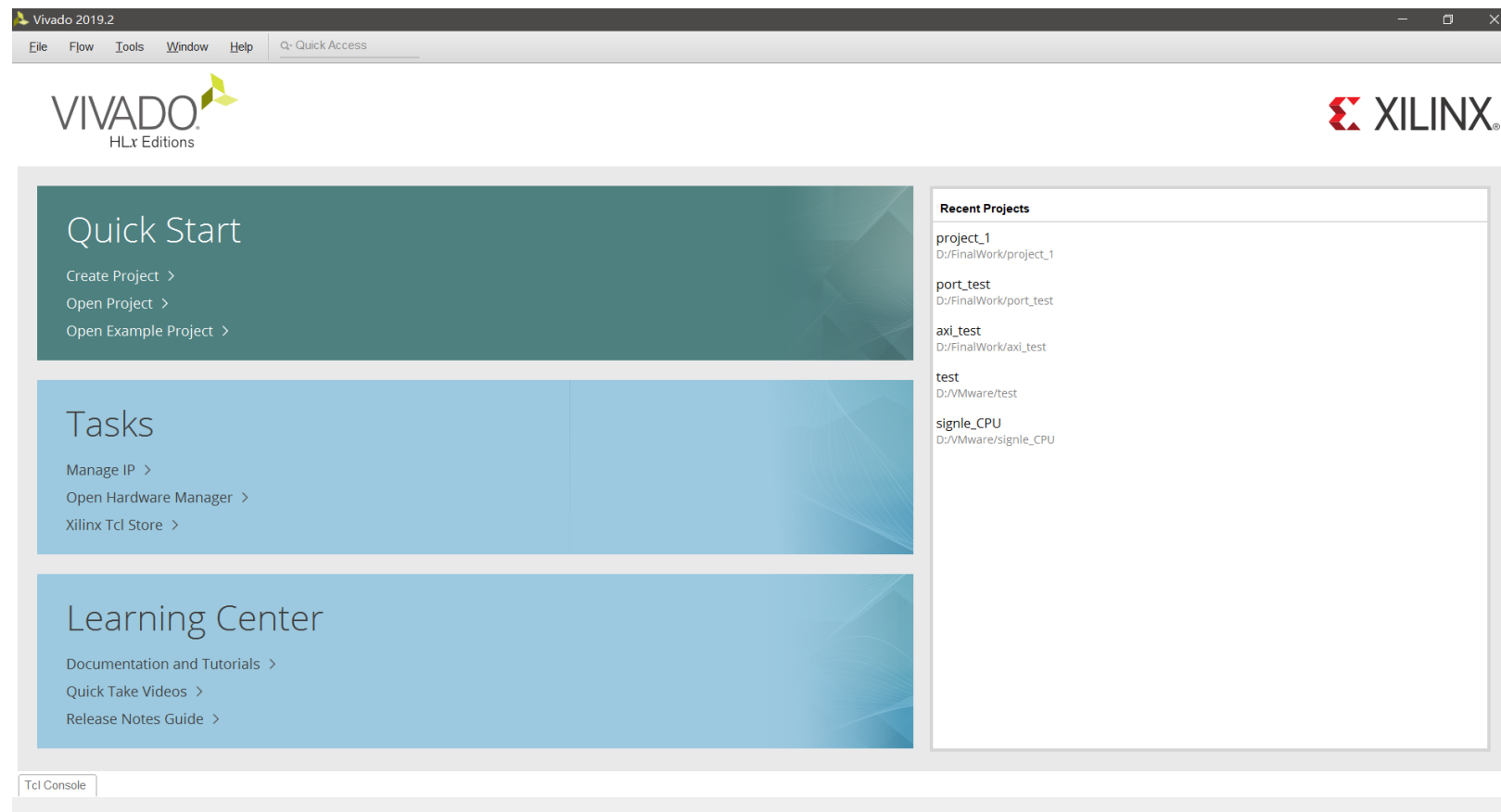
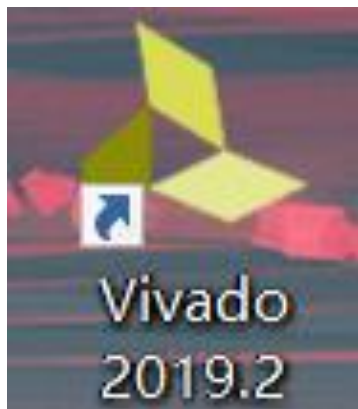


- 一. 选择器件建立工程（或者是打开之前的工程）。
- 二. 添加电路设计描述Verilog文件。
- 三. 添加TB文件，进行功能仿真。
- 四. Synthesis(综合)，添加约束文件。
- 五. 生成bit文件，下载到板子上。

- 编写Verilog代码，实现对七段数码管的控制，以16进制形式在最右侧的两个七段数码管处显示两位数据，每按下指定按钮一次就使得数字自增一。

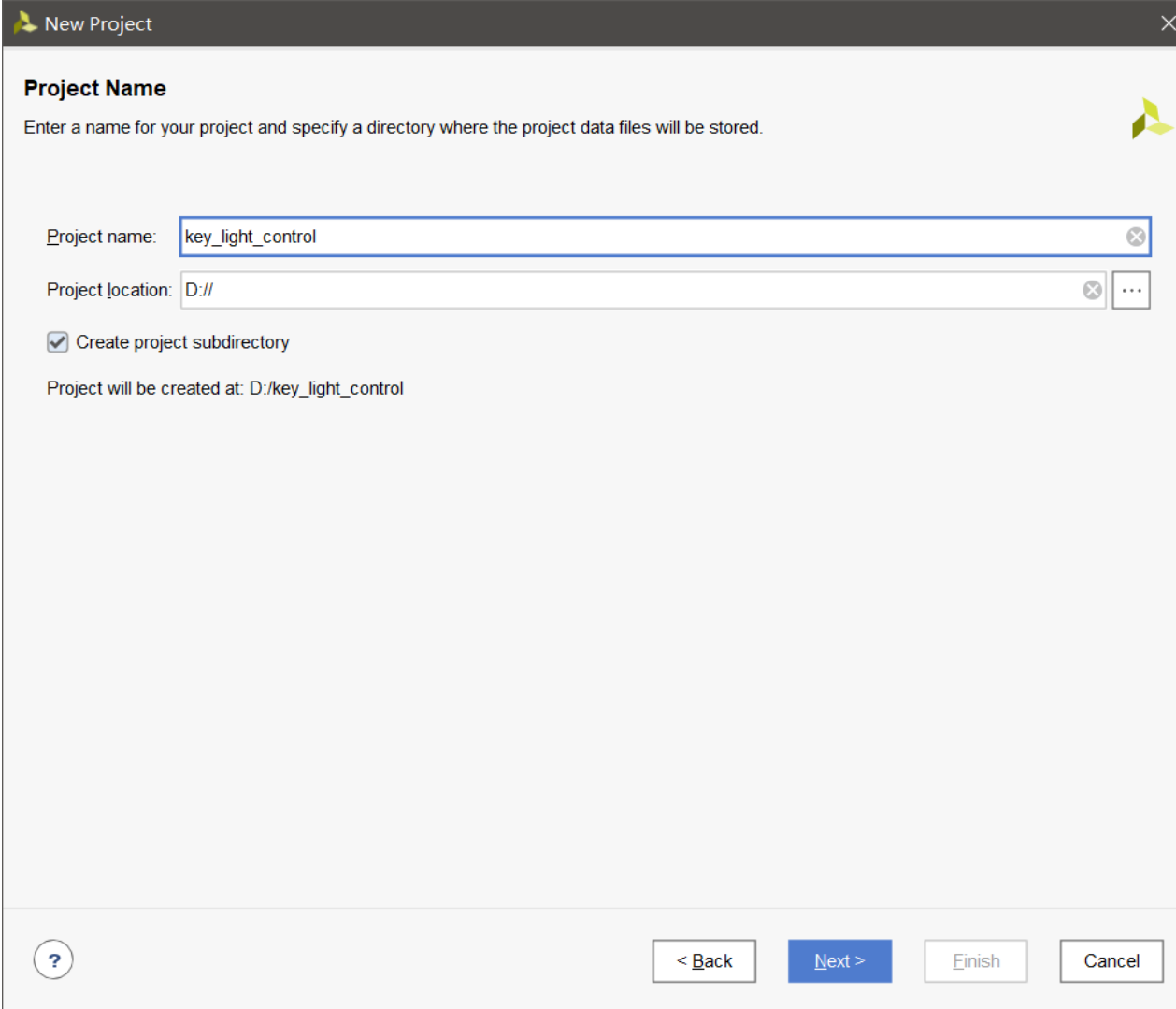


# 1、运行Vivado



## 2、创建项目

- 点击Create Project
- 设置项目路径
- 路径不能出现空格、中文



The image shows a 'New Project' dialog box with the following fields and options:

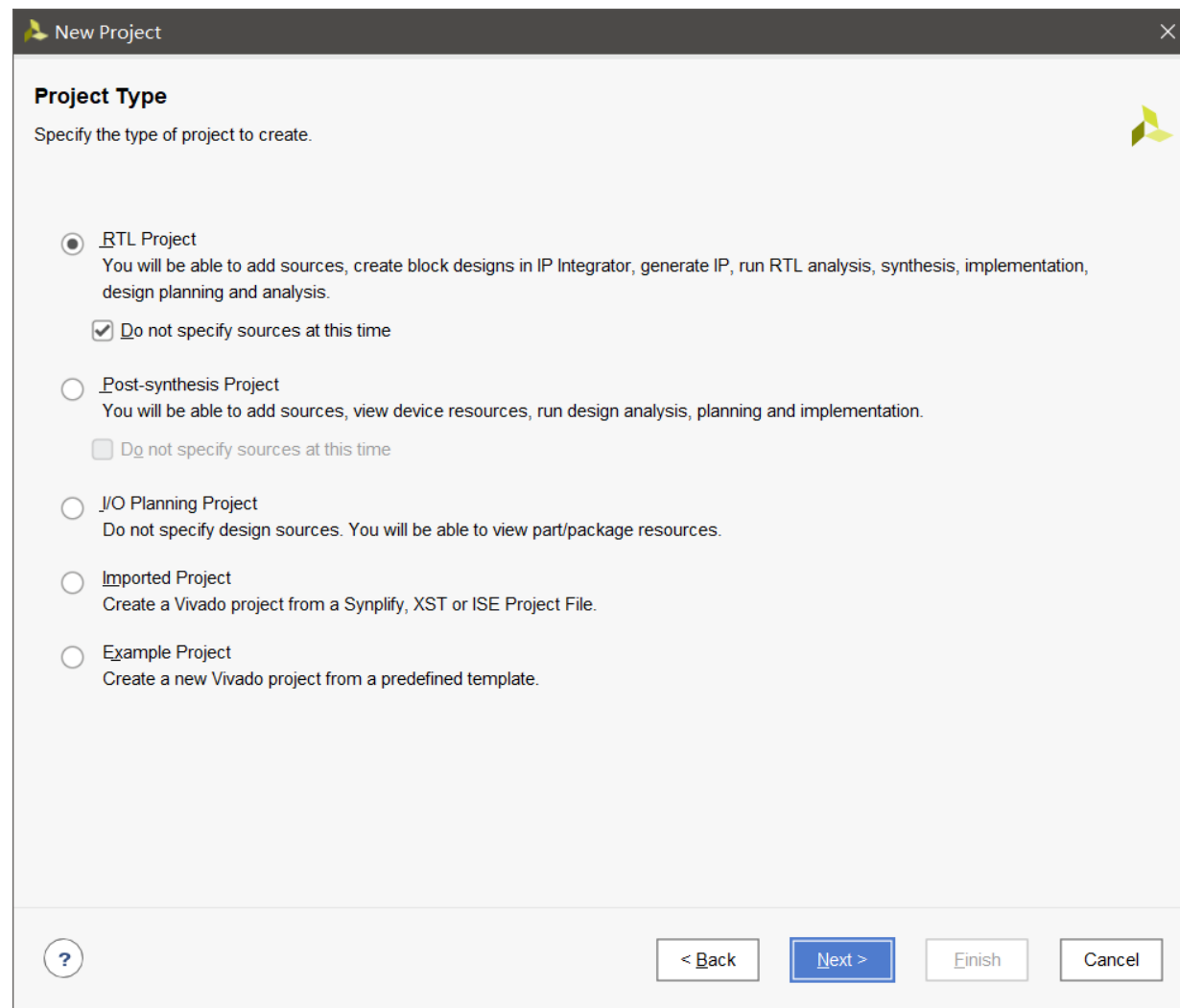
- Project Name:** A text input field containing 'key\_light\_control'.
- Project location:** A text input field containing 'D://', with a browse button (three dots) to its right.
- ☒ **Create project subdirectory**
- Project will be created at:** D:/key\_light\_control

At the bottom, there are four buttons: a help button (question mark), '< Back', 'Next >' (highlighted in blue), 'Finish', and 'Cancel'.

### 3、设置项目类型



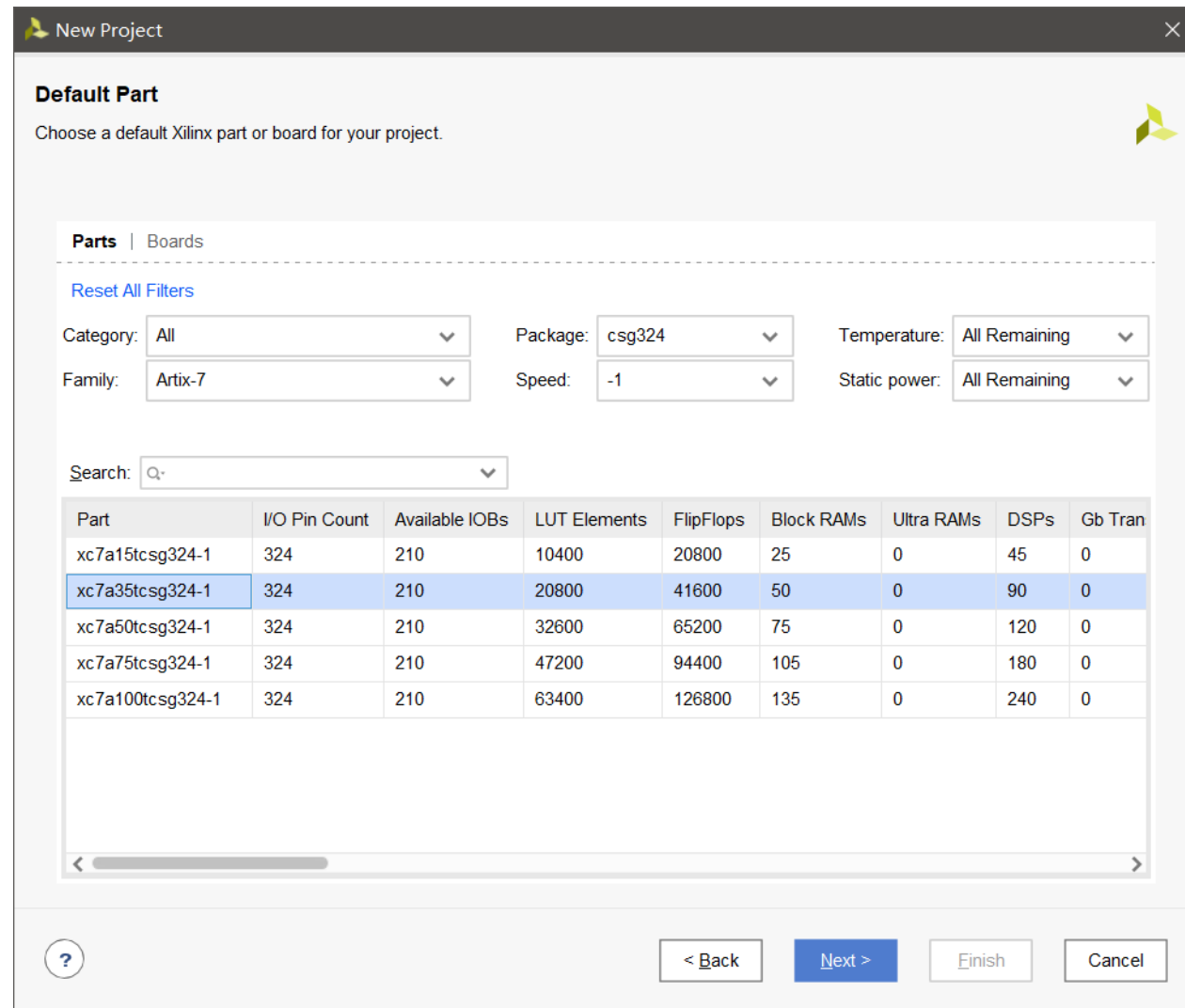
- 选择如图设置
- 点击Next





## 4、设置器件目标型号

- 选择Parts选项卡
- 设置Family为Artix-7
- 设置Package为csg324
- 设置Speed grade为-1
- 在列表中选择xc7a35tcsg324-1
- 点击Next



New Project

**Default Part**

Choose a default Xilinx part or board for your project.

Parts | Boards

[Reset All Filters](#)

Category: All Package: csg324 Temperature: All Remaining

Family: Artix-7 Speed: -1 Static power: All Remaining

Search: Q-

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gb Tran
xc7a15tcsg324-1	324	210	10400	20800	25	0	45	0
xc7a35tcsg324-1	324	210	20800	41600	50	0	90	0
xc7a50tcsg324-1	324	210	32600	65200	75	0	120	0
xc7a75tcsg324-1	324	210	47200	94400	105	0	180	0
xc7a100tcsg324-1	324	210	63400	126800	135	0	240	0

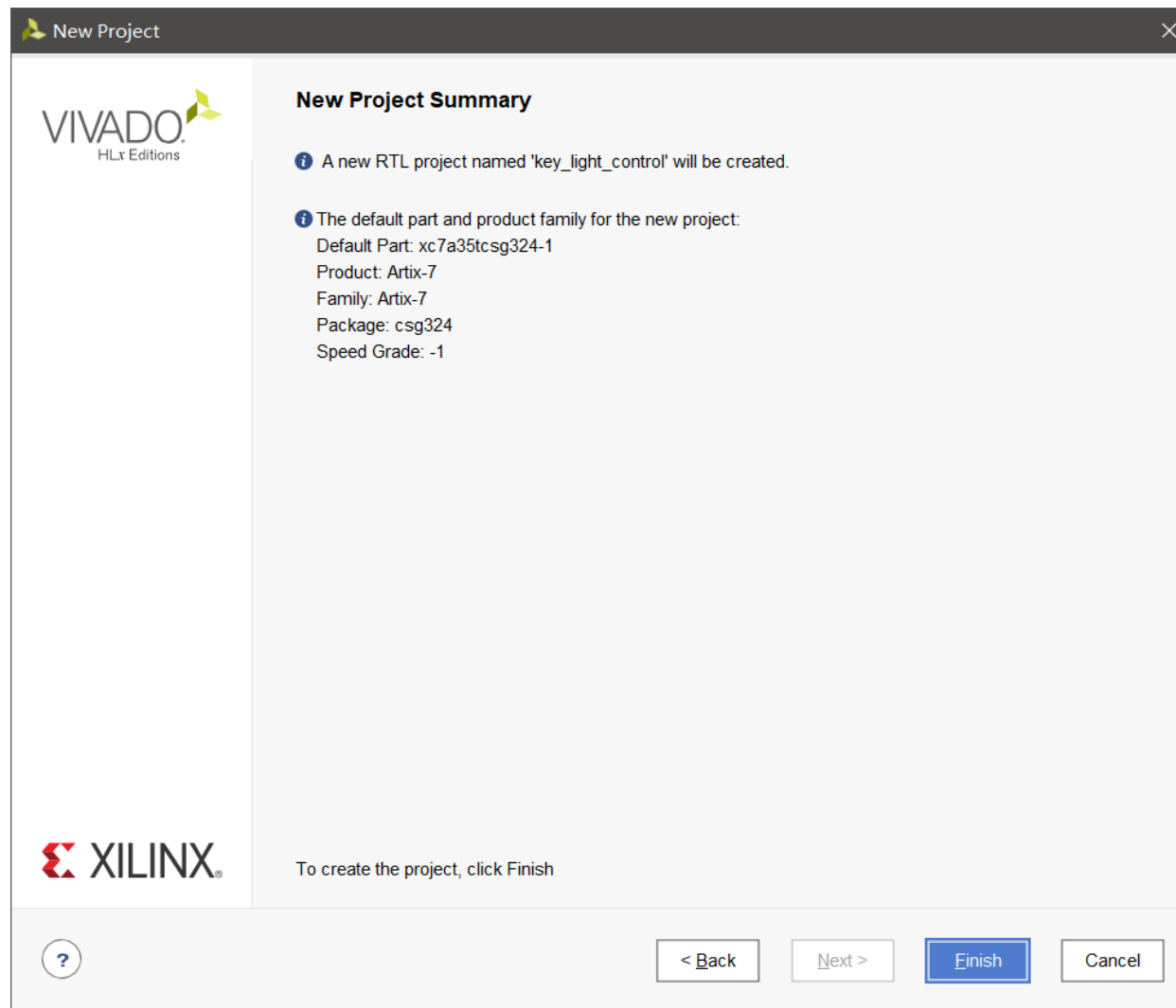
< >

? < Back Next > Finish Cancel

## 5、完成创建



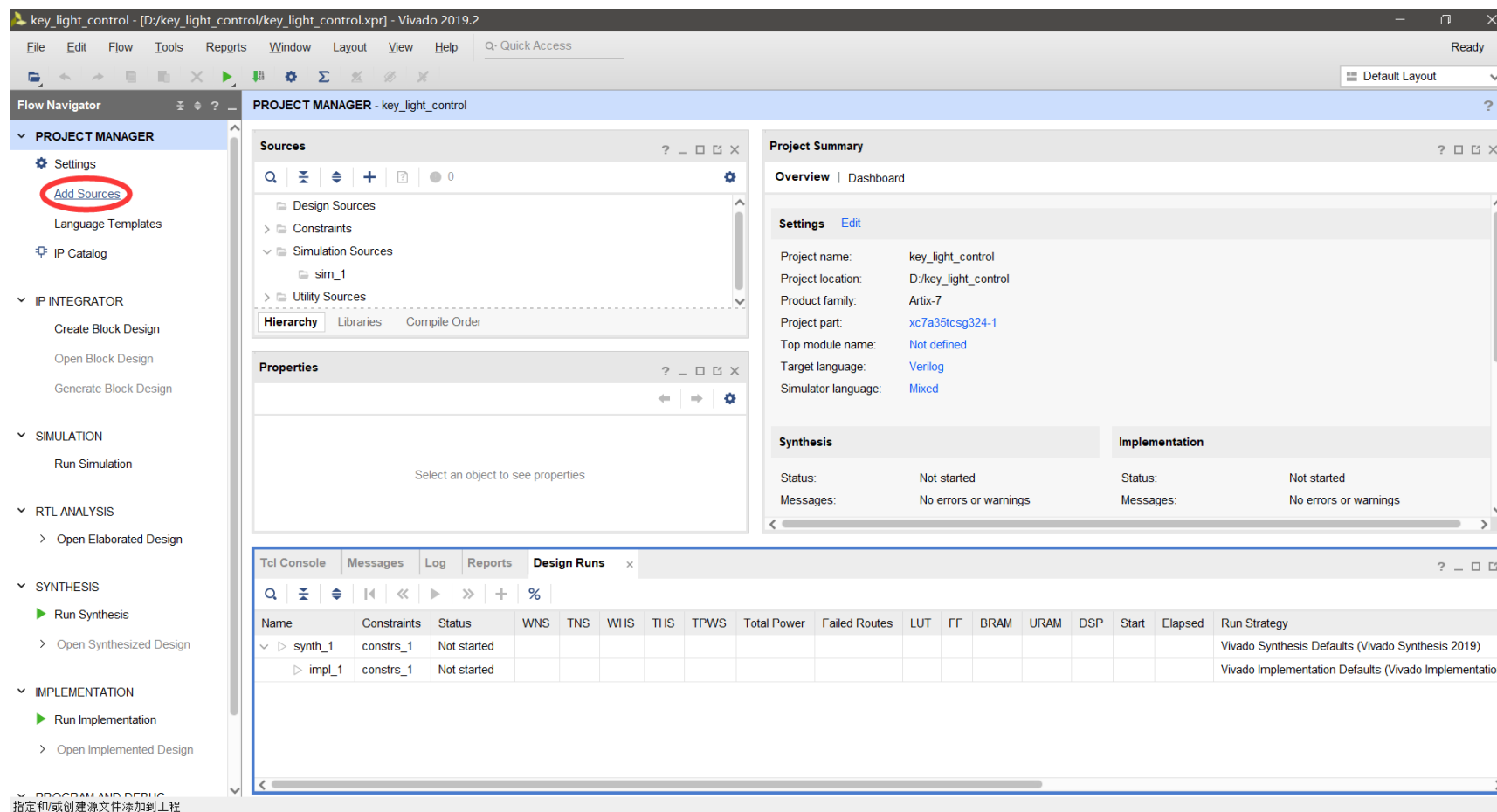
- 在该界面显示已设置好的配置，点击Finish完成创建。



## 6、添加源文件

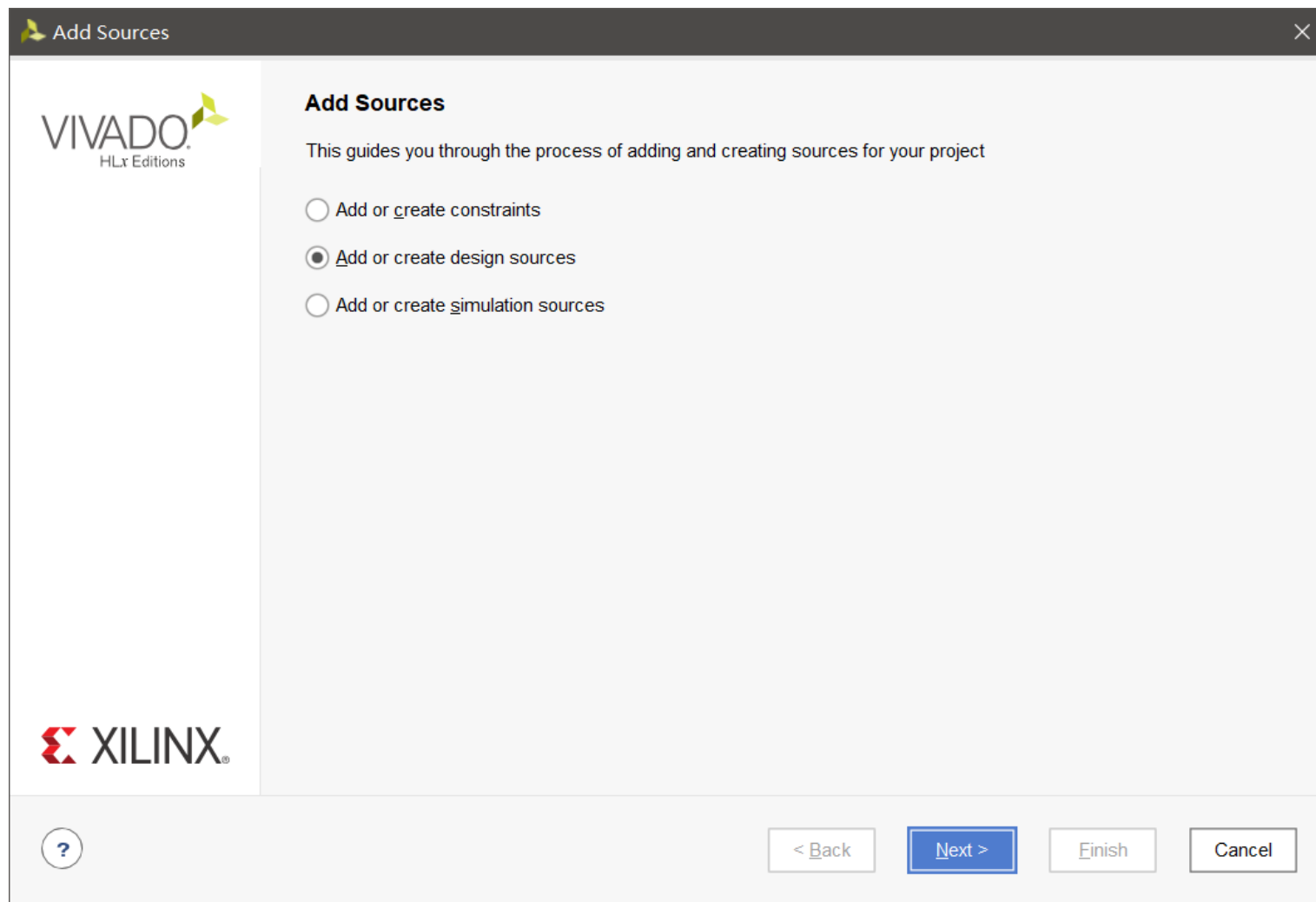


- 点击左侧Flow Navigator 栏中Project Manager选项中的Add Sources



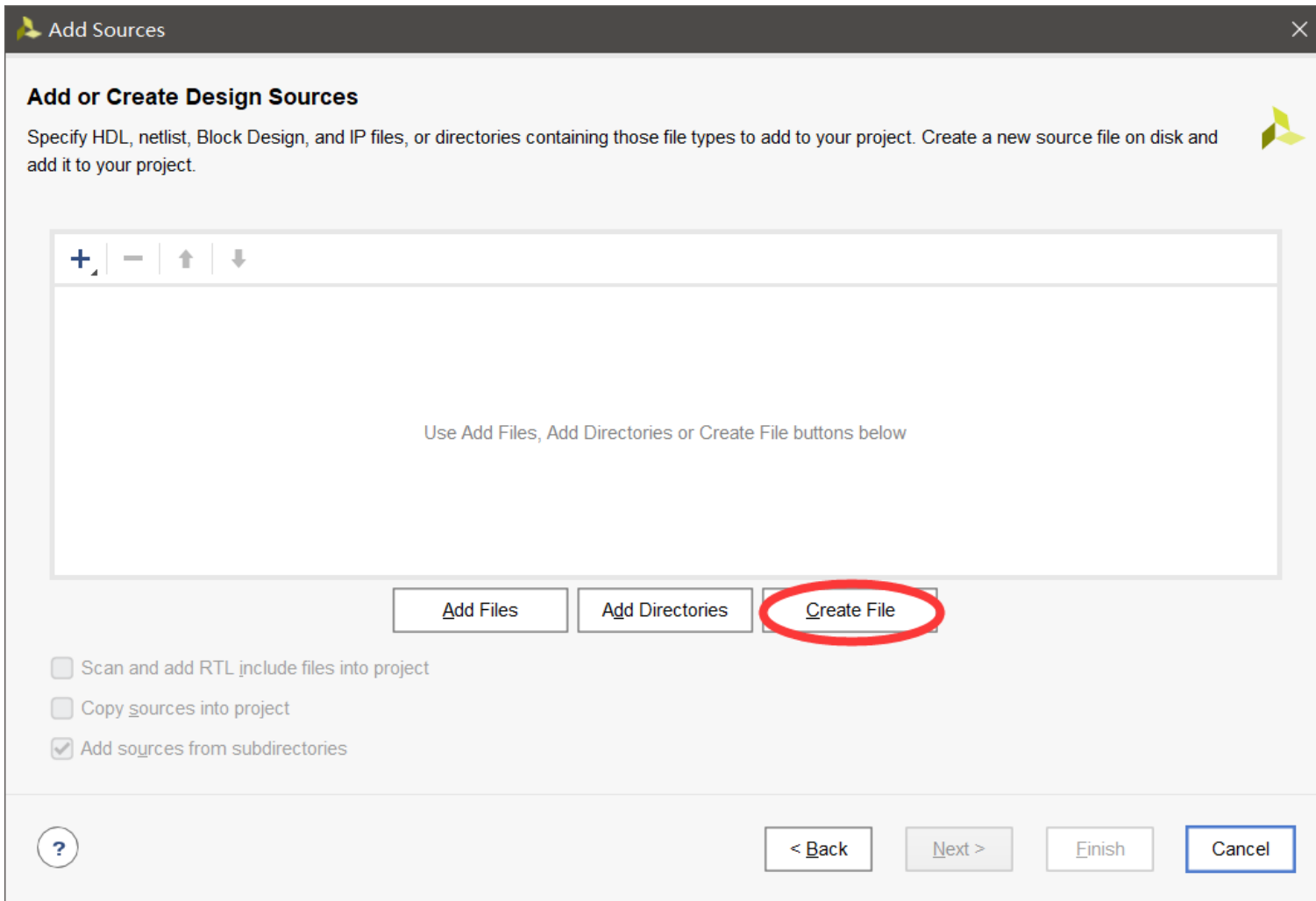
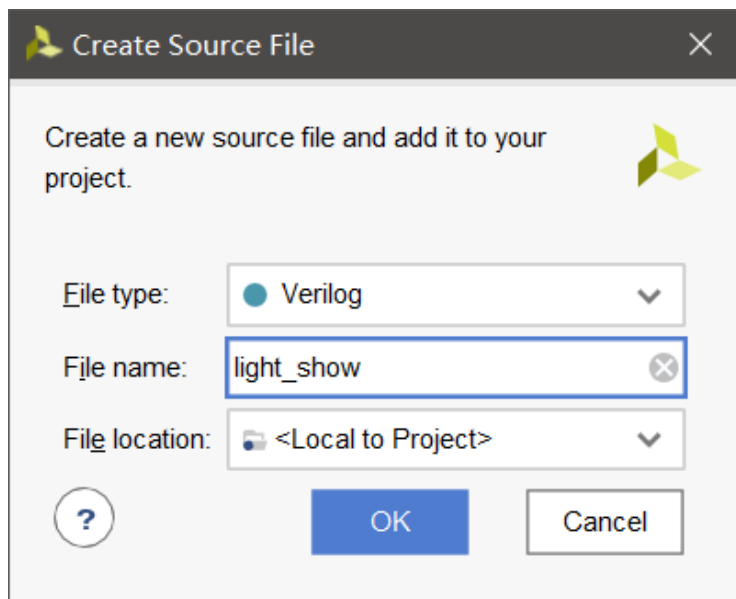
## 6、添加源文件

- 选择第二个选项  
点击Next



## 6、添加源文件

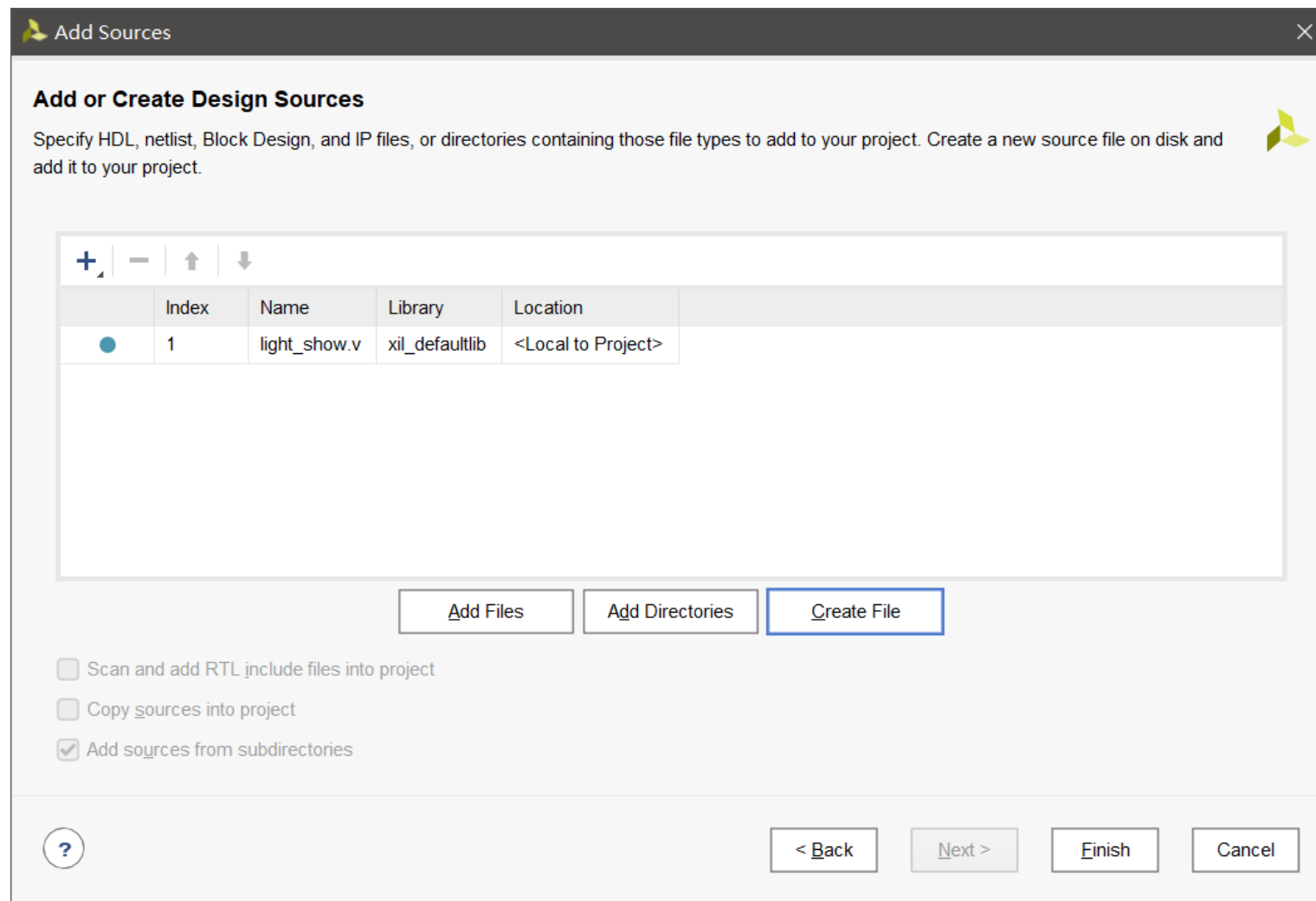
- 点击Create File
- 设置源文件名称
- 点击OK



## 6、添加源文件

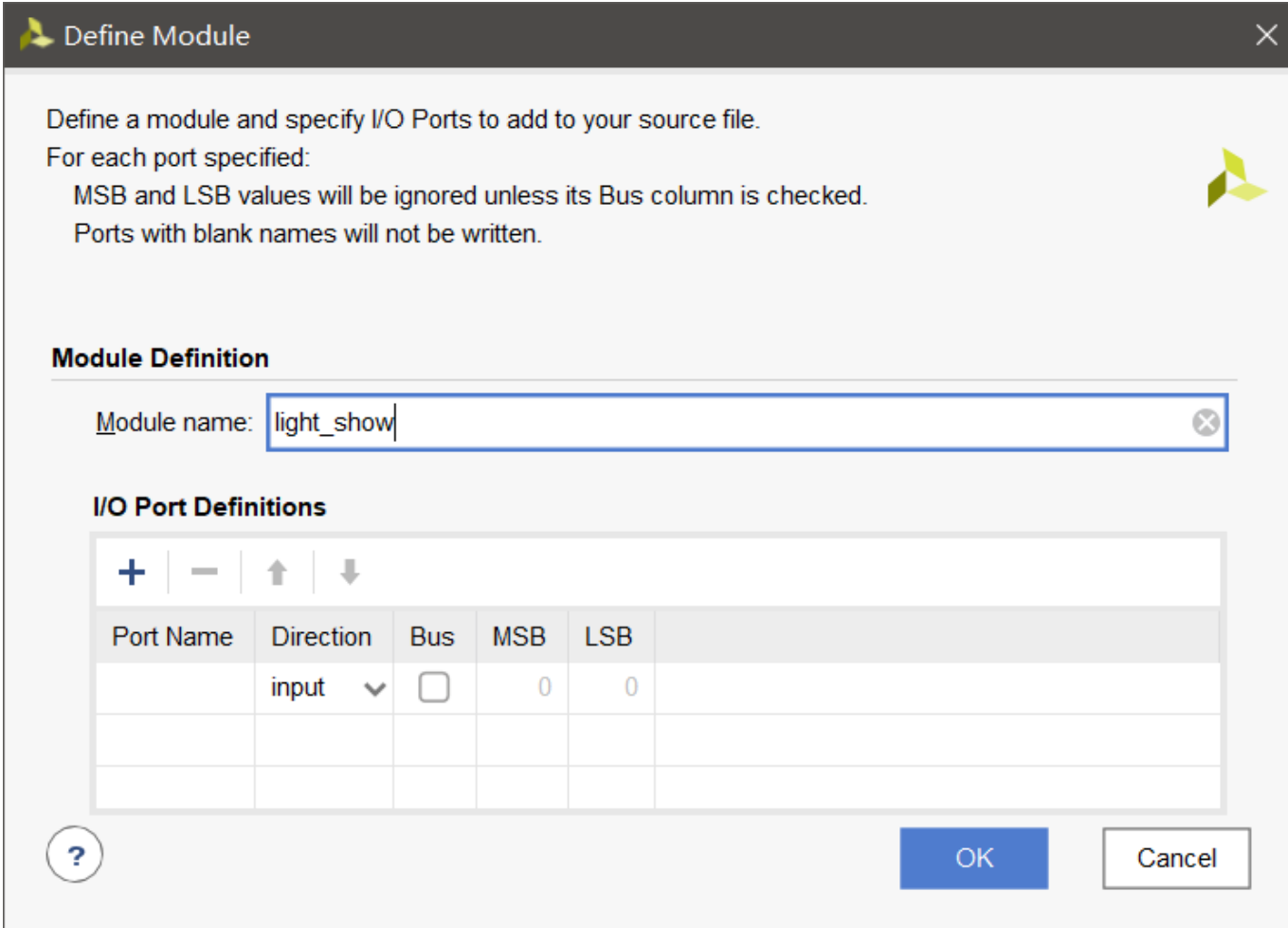


- 源文件已经显示在列表中了
- 点击Finish



## 6、添加源文件

- 弹出设置IO端口的窗口  
先无视掉，在之后代码  
中手动设置端口即可
- 点击OK，然后点击Yes



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**

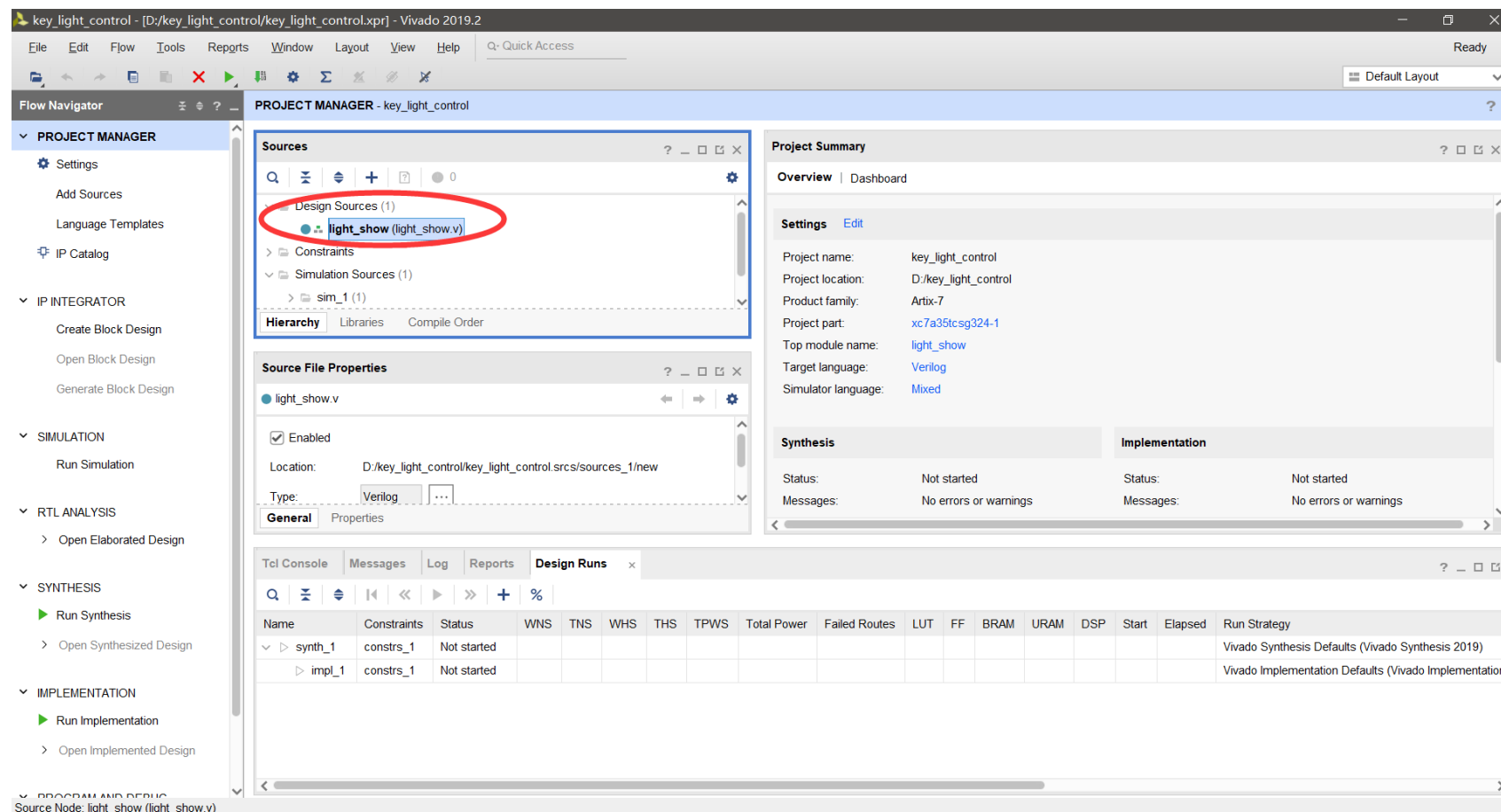
Port Name	Direction	Bus	MSB	LSB
	input	<input type="checkbox"/>	0	0

Buttons: ? OK Cancel



## 7、输入Verilog源代码

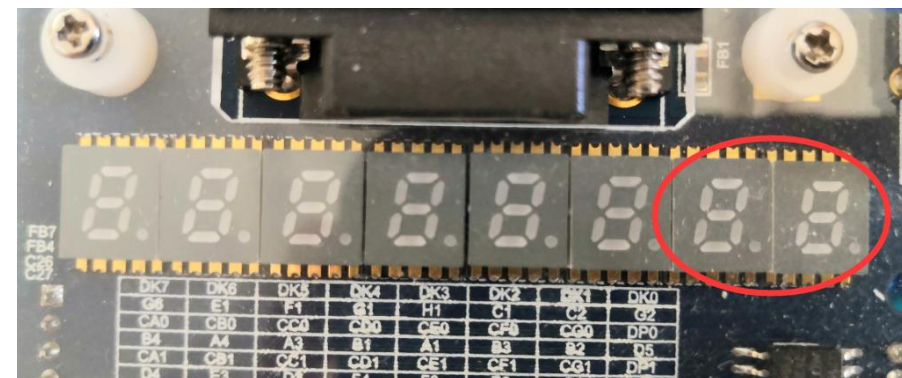
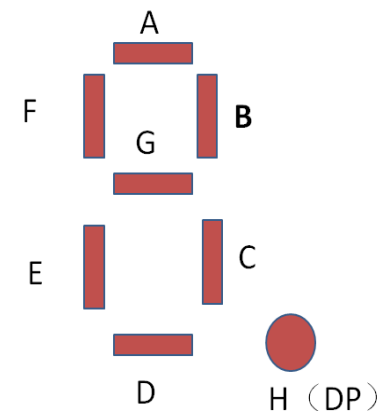
- 在source窗口已经看到刚刚添加的源文件了，双击打开开始编写代码



## 7、编写Verilog代码（七段数码管显示模块）



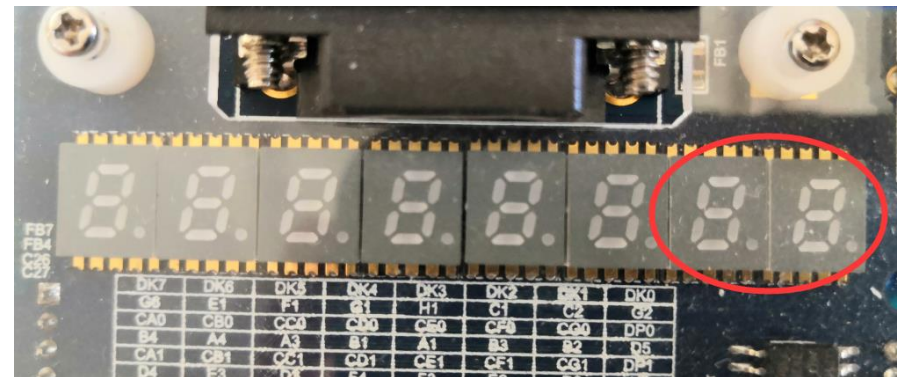
```
1  `timescale 1ns / 1ps                //定义时间单位和仿真精度
2
3  module light_show(                  //定义输入输出端口
4      input      I_clk,               //系统时钟
5      input      I_rst_n,             //复位信号
6      input      [7:0] I_show_num,    //输入8-bit数据
7      output     [6:0] O_led,          //七段数码管LED信号
8      output     [1:0] O_dx           //七段数码管段选信号
9  );
10
11      //parameter C_COUNTER_NUM = 1000000;
12      parameter C_COUNTER_NUM = 10; //计数器峰值
13
14      reg [3:0] R_temp;                //当前显示的4-bit数据寄存器
15      reg [1:0] R_dx_temp;             //段选信号寄存器
16      reg [32:0] R_counter;            //计数器寄存器
```



## 7、编写Verilog代码（七段数码管显示模块）



```
18  always@(posedge I_clk or negedge I_rst_n)
19  begin
20      if(!I_rst_n)
21          begin //复位处理
22              R_dx_temp <= 2'b01;
23              R_temp <= I_show_num[3:0];
24              R_counter <= 0;
25          end
26      else if(R_dx_temp == 2'b01 && R_counter >= C_COUNTER_NUM)
27          begin //显示高4-bit数据
28              R_temp <= I_show_num[7:4];
29              R_dx_temp <= 2'b10;
30              R_counter <= 0;
31          end
32      else if(R_dx_temp == 2'b10 && R_counter >= C_COUNTER_NUM)
33          begin //显示低4-bit数据
34              R_temp <= I_show_num[3:0];
35              R_dx_temp <= 2'b01;
36              R_counter <= 0;
37          end
38      else
39          begin //计数器自增
40              R_counter <= R_counter + 1;
41          end
42  end
```

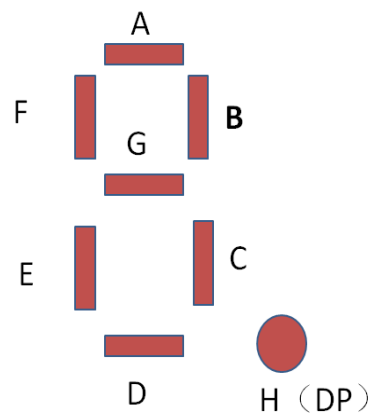


每次计数器到达峰值，改变七段数码管的段选信号及要显示的数字，实现每位数码管交替显示。

## 7、编写Verilog代码（七段数码管显示模块）



```
44 //led端信号
45 assign O_led[0] = (R_temp == 4'b0000 || R_temp == 4'b0001 || R_temp == 4'b0111
46                  || R_temp == 4'b1100) ? 0 : 1;
47 assign O_led[1] = (R_temp == 4'b0001 || R_temp == 4'b0010 || R_temp == 4'b0011
48                  || R_temp == 4'b0111 || R_temp == 4'b1101) ? 0 : 1;
49 assign O_led[2] = (R_temp == 4'b0001 || R_temp == 4'b0011 || R_temp == 4'b0100
50                  || R_temp == 4'b0101 || R_temp == 4'b0111 || R_temp == 4'b1001) ? 0 : 1;
51 assign O_led[3] = (R_temp == 4'b0001 || R_temp == 4'b0100 || R_temp == 4'b0111
52                  || R_temp == 4'b1010 || R_temp == 4'b1111) ? 0 : 1;
53 assign O_led[4] = (R_temp == 4'b0010 || R_temp == 4'b1100 || R_temp == 4'b1110
54                  || R_temp == 4'b1111) ? 0 : 1;
55 assign O_led[5] = (R_temp == 4'b0101 || R_temp == 4'b0110 || R_temp == 4'b1011
56                  || R_temp == 4'b1100 || R_temp == 4'b1110 || R_temp == 4'b1111) ? 0 : 1;
57 assign O_led[6] = (R_temp == 4'b0001 || R_temp == 4'b0100 || R_temp == 4'b1011
58                  || R_temp == 4'b1101) ? 0 : 1;
59 //段选信号
60 assign O_dx = R_dx_temp;
61 endmodule
```



## 7、编写Verilog代码（按键去抖模块）



```
1  `timescale 1ns / 1ps                //定义时间单位和仿真精度
2
3  module key_rebounce(                 //定义输入输出端口
4      input      I_clk,                //系统时钟
5      input      I_rst_n,              //复位信号
6      input      I_key_in,             //按键输入信号
7      output reg  O_key_out             //按键去抖输出信号
8  );
9
10     reg R_key_in0;                    //记录上个时钟周期的按键输入信号
11     reg [19:0] R_count;                //计数寄存器
12
13     wire W_change;                    //按键输入改变信号
14
15     parameter C_COUNTER_NUM = 5;
16     //parameter C_COUNTER_NUM = 180000;
17
18     always@(posedge I_clk or negedge I_rst_n)
19         if(!I_rst_n) //复位处理
20             R_key_in0 <= 0;
21         else //记录按键输入
22             R_key_in0 <= I_key_in;
23     //如果前后两个时钟按键输入数据不同，将此信号置为1
24     assign W_change=(I_key_in & !R_key_in0) | (!I_key_in & R_key_in0);
```



抖动检测：  
判断按键信号是否发生改变，若发生变化则 W\_change 为1。

## 7、编写Verilog代码（按键去抖模块）



```
26  always@(posedge I_clk or negedge I_rst_n)
27      if(!I_rst_n)    //复位处理
28          R_count <= 0;
29      else if(W_change) //按键输入发生改变，重新开始计数
30          R_count <= 0;
31      else
32          R_count <= R_count + 1;
33
34  always@(posedge I_clk or negedge I_rst_n)
35      if(!I_rst_n)    //复位处理
36          O_key_out <= 0;
37      else if(R_count >= C_COUNTER_NUM - 1) //更改输出信号
38          O_key_out <= I_key_in;
39
40  endmodule
```

稳定计数：

若按键信号发生变化，重新开始计数。

确认稳定：

计数器到达峰值，则将按键输出信号与输入信号同步。



## 7、编写Verilog代码（顶层控制模块）



```
1  `timescale 1ns / 1ps                //定义时间单位和仿真精度
2
3  module key_light_control(            //定义输入输出端口
4      input      I_clk,                //系统时钟
5      input      I_rst_n,              //复位信号
6      input      I_key,                //按键输入信号
7      output     [6:0] O_led,           //七段数码管LED信号
8      output     [1:0] O_dx             //七段数码管段选信号
9  );
10
11  wire          W_add_able;             //按键去抖后信号
12  key_rebounce U_key_rebounce           //调用按键去抖模块
13  (
14      .I_clk      (I_clk),
15      .I_rst_n    (I_rst_n),
16      .I_key_in   (I_key),
17      .O_key_out  (W_add_able)
18  );
```



## 7、编写Verilog代码（顶层控制模块）



```
20  reg [7:0]  R_num;           //显示数字寄存器
21  reg        R_added;        //确保一次按键响应一次自增操作
22
23  always @(posedge I_clk or negedge I_rst_n)
24  begin
25      if(!I_rst_n)
26      begin//复位处理
27          R_num <= 0;
28          R_added <= 0;
29      end
30      else if(W_add_able == 1)
31      begin//按键信号为高电平
32          if(R_added == 0)
33          begin//此次按键没有执行自增操作，自增并将信号置为1，暂停自增行为
34              R_num <= R_num + 1;
35              R_added <= 1;
36          end
37      end
38      else
39      begin//按键信号为低电平，将信号置为0，可以继续自增
40          R_added <= 0;
41      end
42  end
```

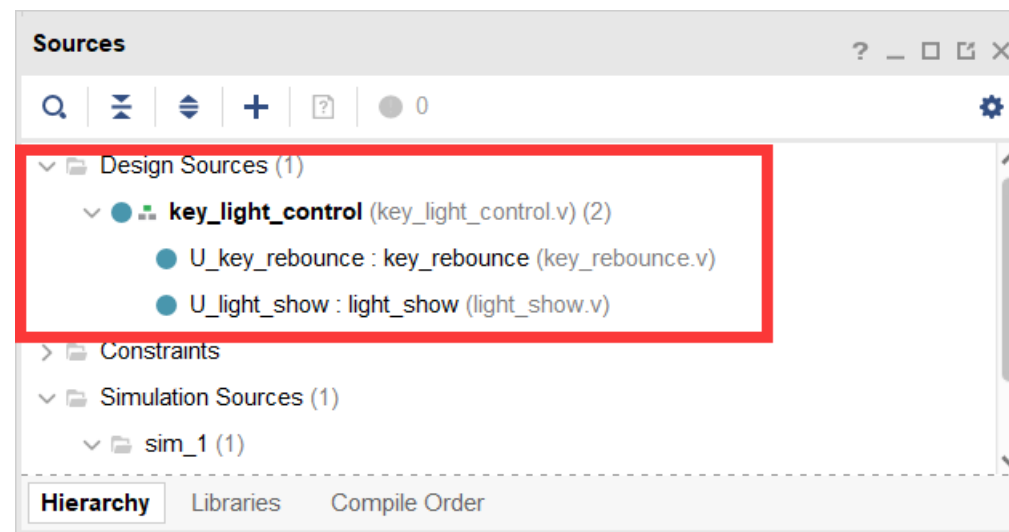
自增一次后将R\_added置为1，使得每次按下按键之会发生一次自增。

## 7、编写Verilog代码（顶层控制模块）



```
44     light_show U_light_show    //调用数码管显示模块
45     (
46         .I_clk          (I_clk),
47         .I_rst_n         (I_rst_n),
48         .I_show_num      (R_num),
49         .O_led           (O_led),
50         .O_dx            (O_dx)
51     );
52
53 endmodule
```

- 编写完成后，可在Sources窗口中看到各模块间的引用关系

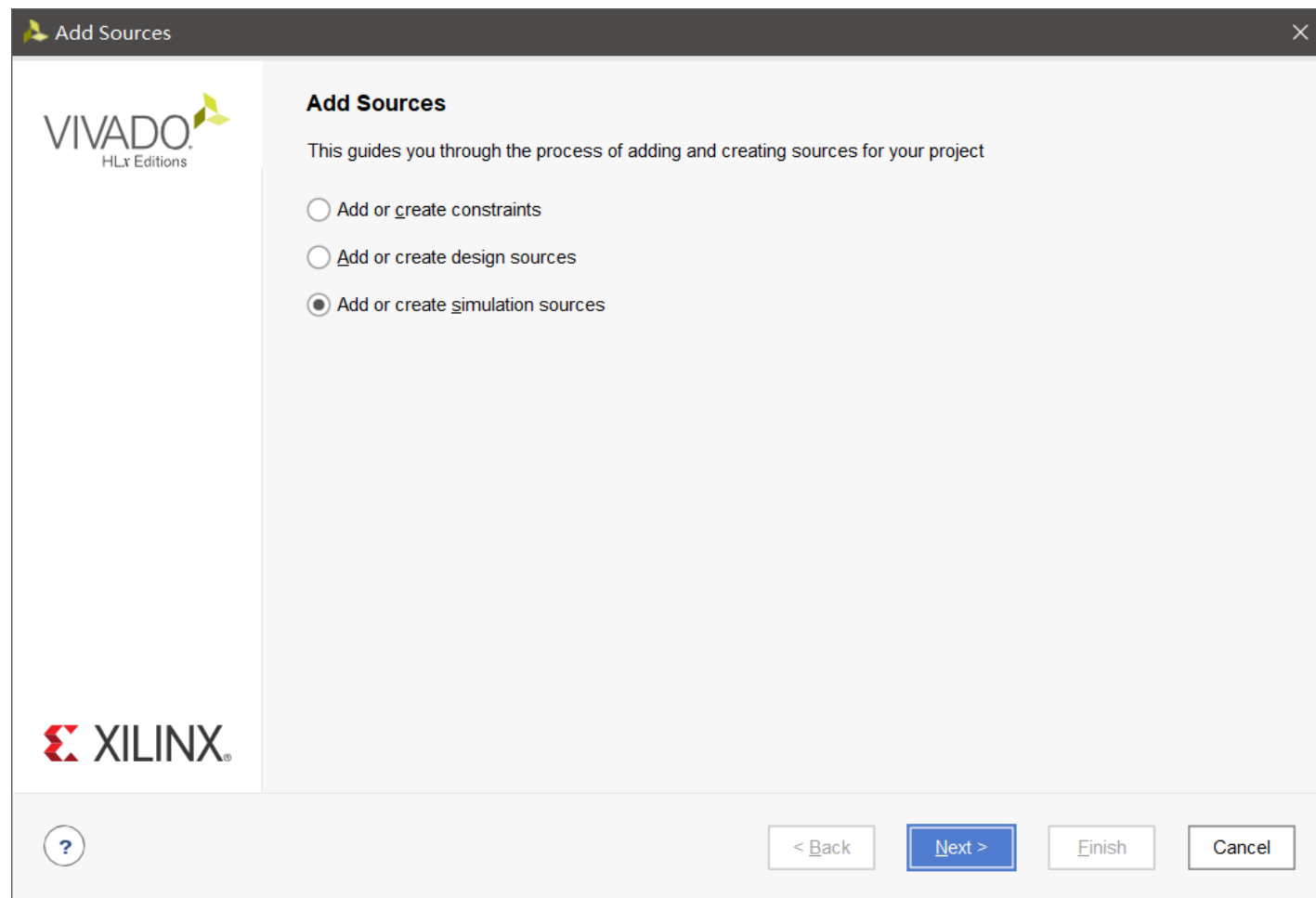


- Testbench:

Testbench是一种验证的手段,可以看做模拟实际环境的输入激励和输出校验的一种“虚拟平台”。在这个平台上可以对设计从软件层面上进行分析和校验,类似于一个激励的产生器。

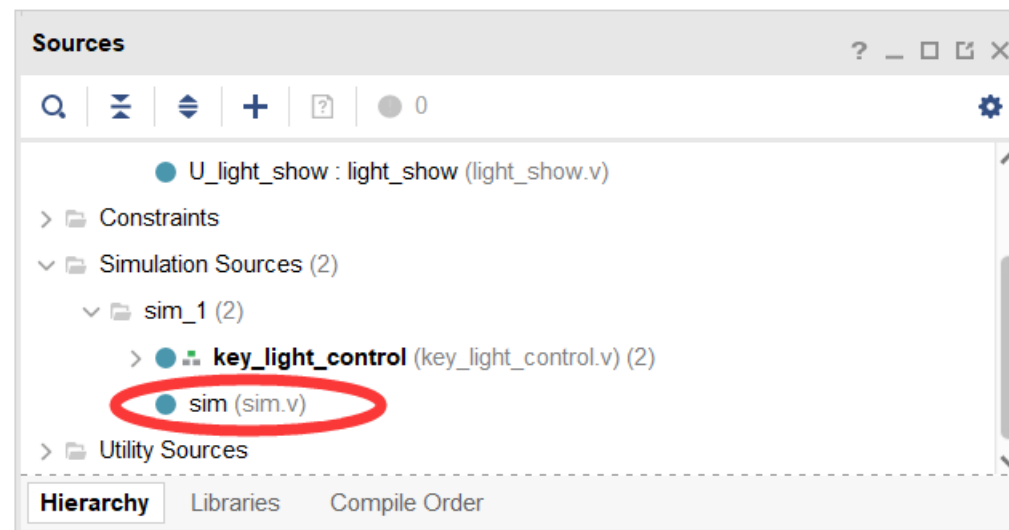
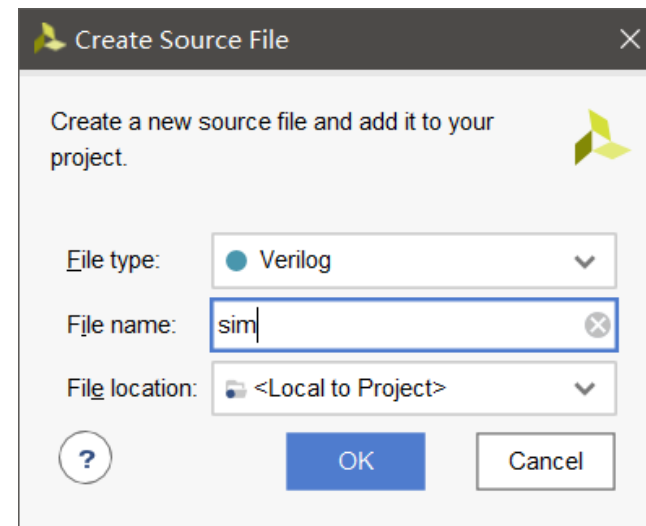
## 8、行为仿真

- 添加仿真激励文件，点击Add Sources，选中simulation sources



## 8、行为仿真

- 点击Create File，设置文件名为sim
- 点击下一步，直至创建完成
- 可以在右图的位置看到创建的文件



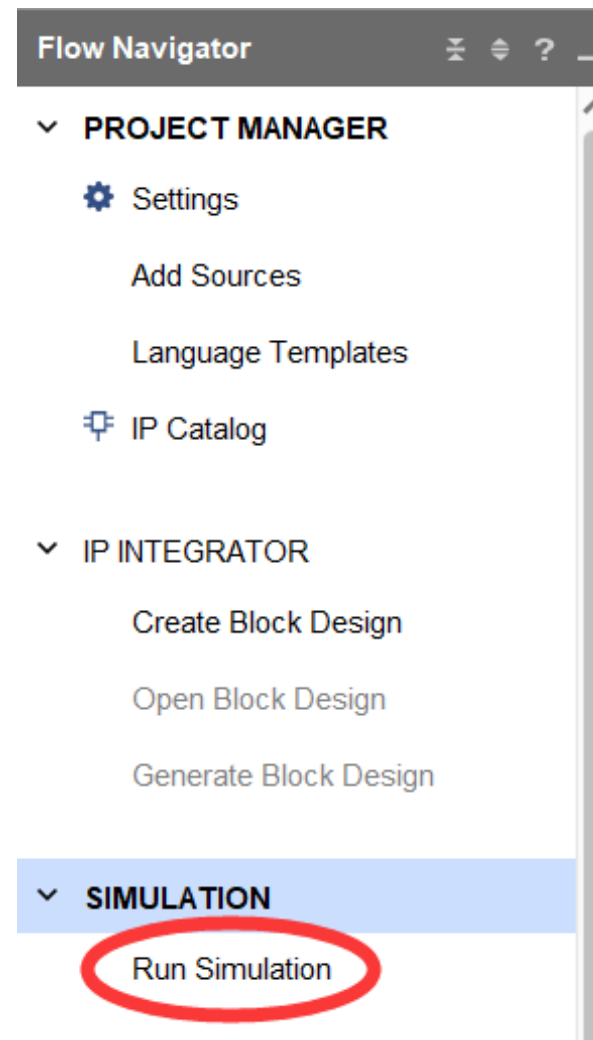
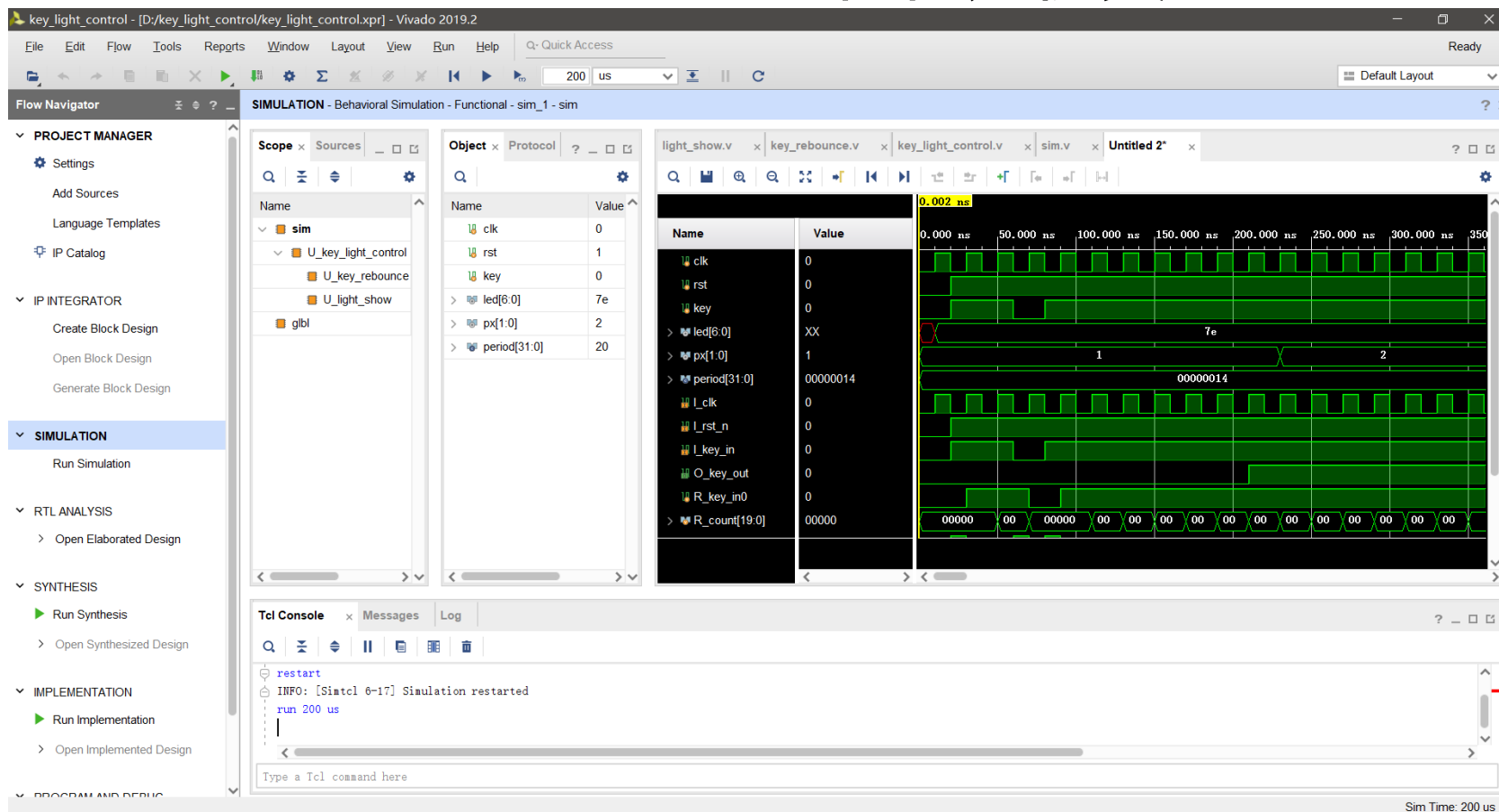
```
1  `timescale 1ns / 1ps
2
3  module sim();
4      reg clk, rst_n, key;    //模块输入信号设置为reg型
5
6      wire [6:0] led;         //模块输出信号设置为wire型
7      wire [1:0] dx;
8
9      parameter period = 20; //一时钟周期时间
10     //调用想要进行仿真的模块
11     key_light_control U_key_light_control
12     (
13         .I_clk          (clk),
14         .I_rst_n         (rst_n),
15         .I_key           (key),
16         .O_led           (led),
17         .O_dx            (dx)
18     );
19     //模拟时钟信号
20     always begin
21         clk = 1'b0;
22         #(period/2); //延时半个period
23         clk = 1'b1;
24         #(period/2);
25     end
```

```
27     always begin
28         rst_n = 1'b0;
29         key = 1'b0;
30         #(period/2);
31
32         //按键去抖测试
33         rst_n = 1'b1;
34         key = 1'b1;
35         #(4*period);
36         key = 1'b0;
37         #period;
38
39         //按键控制测试
40         key = 1'b1;
41         #(50*period);
42         key = 1'b0;
43         #(50*period);
44         key = 1'b1;
45         #(50*period);
46         key = 1'b0;
47         #(1000*period);
48     end
49
50 endmodule
```

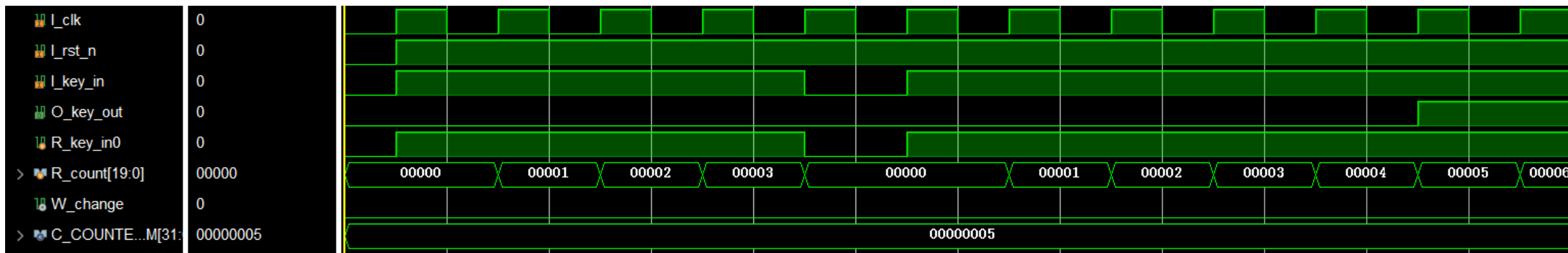
## 8、行为仿真



- 点击左侧菜单栏的Run Simulation，并选择Run Behavioral Simulation进行行为仿真



## 8、行为仿真（按键去抖效果展示）



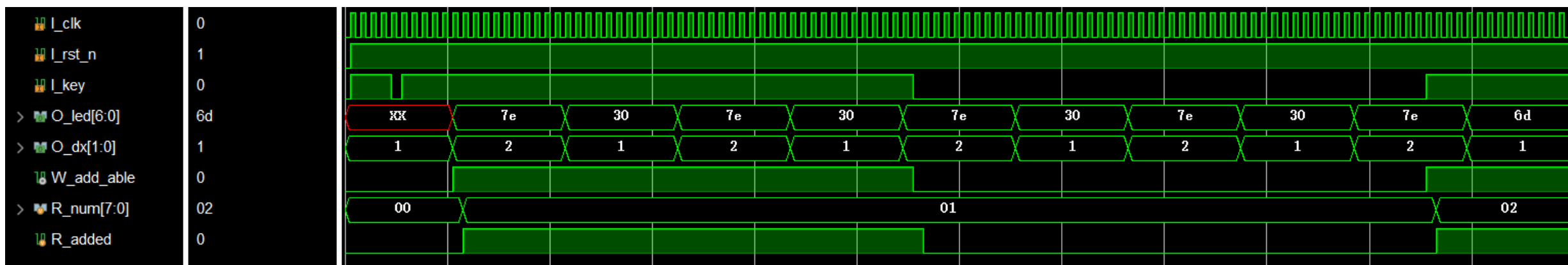
- 按键信号不稳定时（稳定时钟周期数不足），输出的按键信号不会发生变化

```
//按键去抖测试
key = 1'b1;
#(50*period);
key = 1'b0;
#(50*period);
key = 1'b1;
#(50*period);
key = 1'b0;
#(1000*period);

//按键去抖测试
rst = 1'b1;
key = 1'b1;
#(4*period);
key = 1'b0;
#period;
```



## 8、行为仿真（按键控制效果展示）



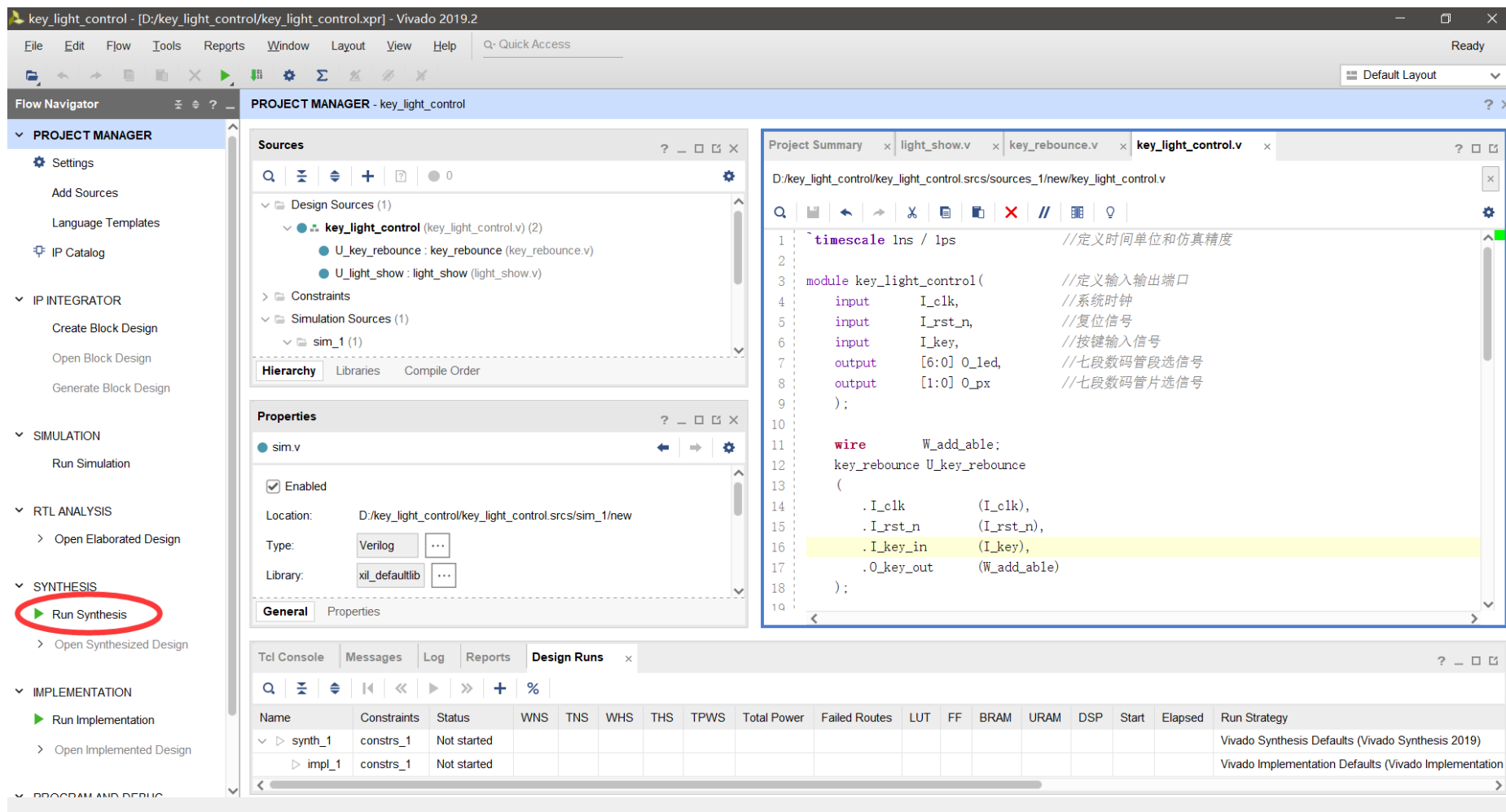
- 一次按键只会触发一次自增效果
- 七段数码管LED信号、段选信号正常切换

//按键控制测试

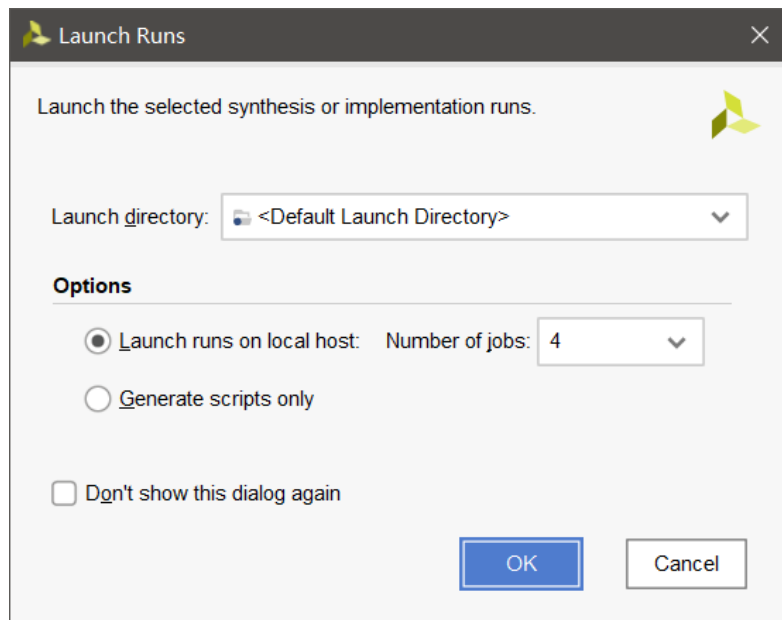
```
key = 1'b1;  
#(50*period);  
key = 1'b0;  
#(50*period);  
key = 1'b1;  
#(50*period);  
key = 1'b0;  
#(1000*period);
```

## 9、项目综合

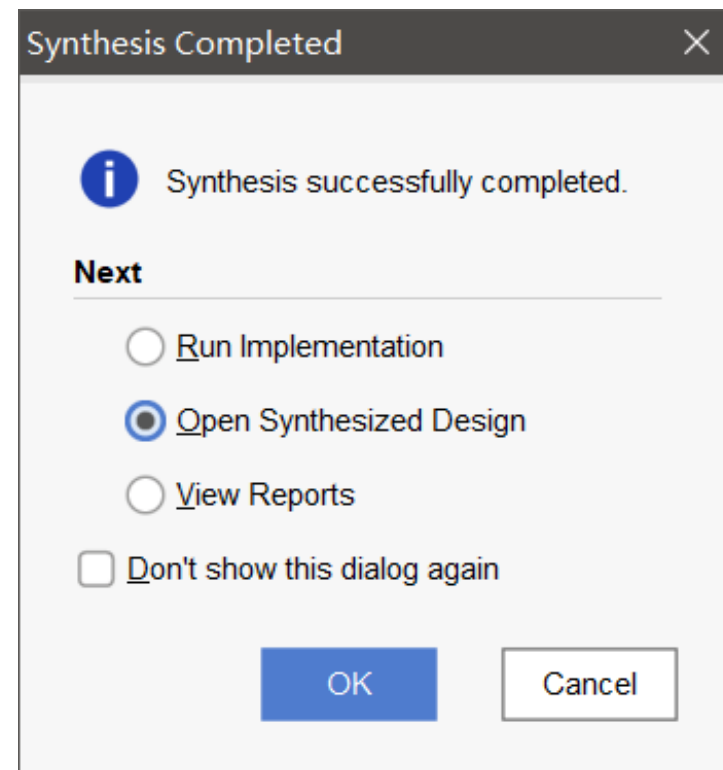
- 点击Run Synthesis进行项目综合
- (若弹出保存提示窗口 点击Save)



- 点击OK

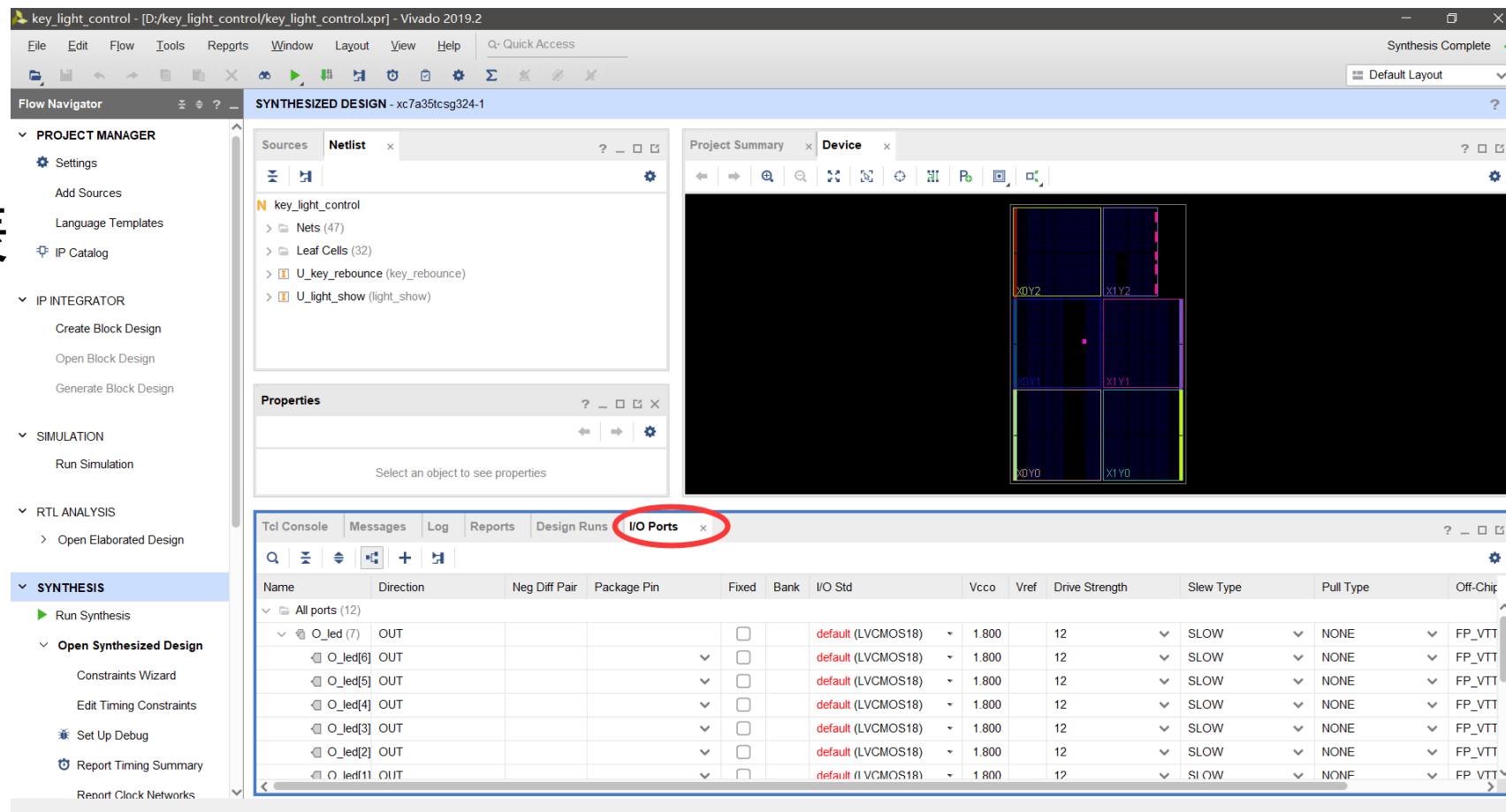


- 综合成功后弹出右图窗口，选择第二个选项，点击OK



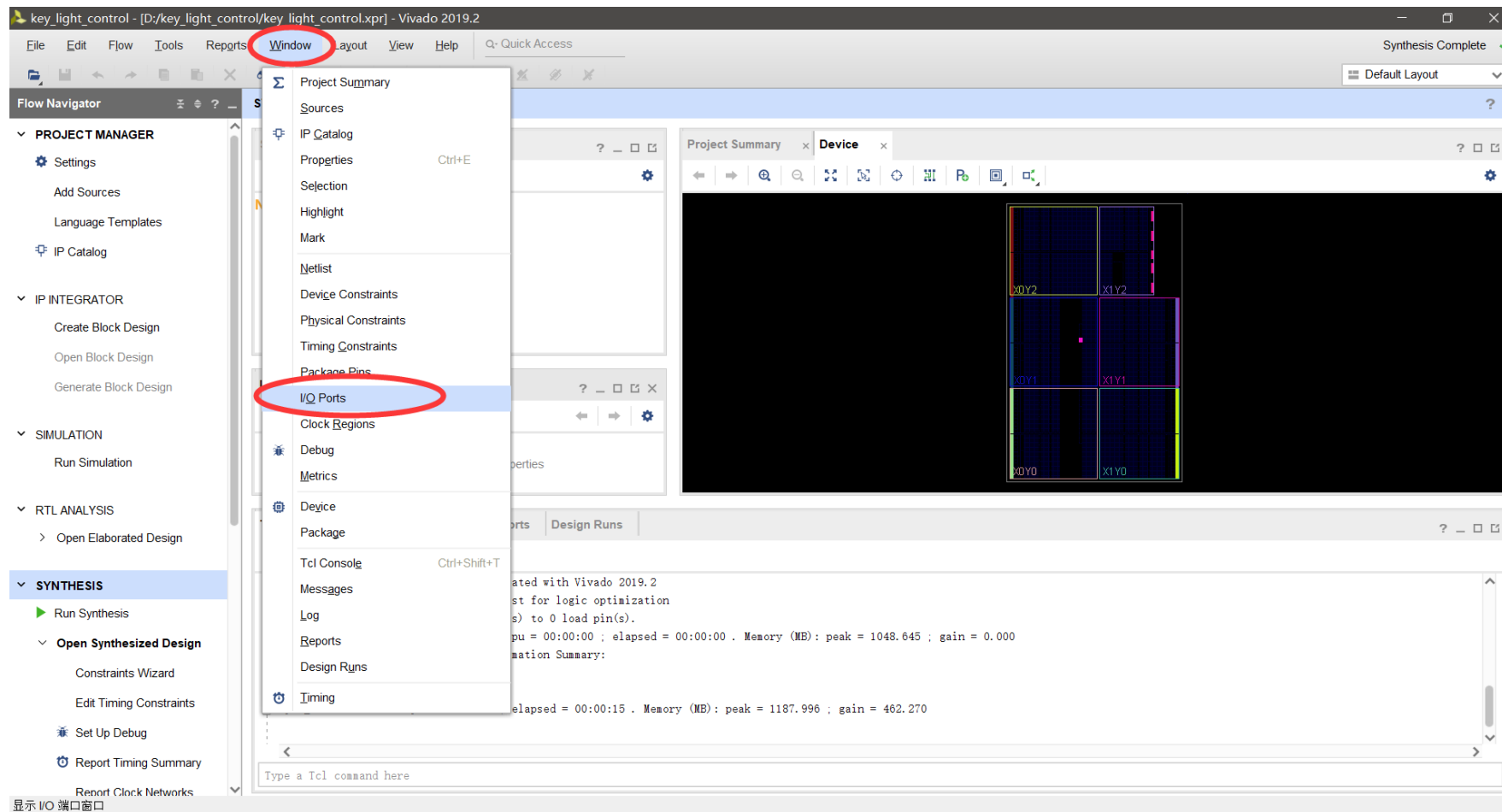
# 10、配置管脚

- 显示如图界面
- 点击I/O Ports  
可看到管脚列表



## 10、配置管脚

- 若界面中没有 I/O Ports
- 点击Windows，打开I/O Ports 窗口



## 10、配置管脚



按照下图配置管脚（此处O\_led对应七段数码管的LED信号，O\_dx为七段数码管的段选信号，I\_clk为时钟，I\_rst\_n为板子上的复位按钮S8，I\_key为板子上的通用按钮S1）

Tcl Console

Messages

Log

Reports

Design Runs












I/O Ports

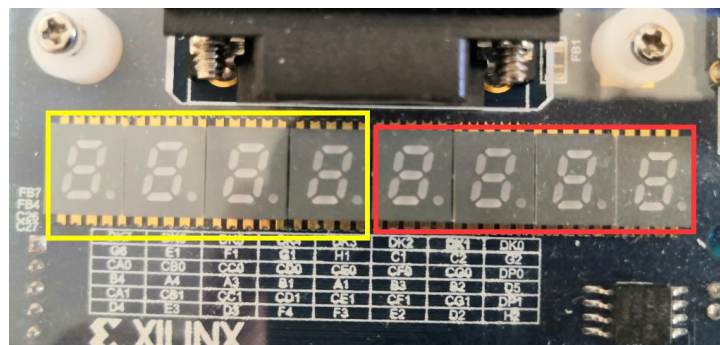
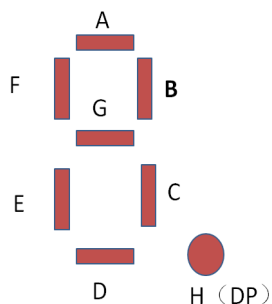
×

Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco
All ports (12)							
O_dx (2)	OUT			<input checked="" type="checkbox"/>	35	LVC MOS33*	3.300
O_dx[1]	OUT		C2	<input checked="" type="checkbox"/>	35	LVC MOS33*	3.300
O_dx[0]	OUT		G2	<input checked="" type="checkbox"/>	35	LVC MOS33*	3.300
O_led (7)	OUT			<input checked="" type="checkbox"/>	35	LVC MOS33*	3.300
O_led[6]	OUT		B4	<input checked="" type="checkbox"/>	35	LVC MOS33*	3.300
O_led[5]	OUT		A4	<input checked="" type="checkbox"/>	35	LVC MOS33*	3.300
O_led[4]	OUT		A3	<input checked="" type="checkbox"/>	35	LVC MOS33*	3.300
O_led[3]	OUT		B1	<input checked="" type="checkbox"/>	35	LVC MOS33*	3.300
O_led[2]	OUT		A1	<input checked="" type="checkbox"/>	35	LVC MOS33*	3.300
O_led[1]	OUT		B3	<input checked="" type="checkbox"/>	35	LVC MOS33*	3.300
O_led[0]	OUT		B2	<input checked="" type="checkbox"/>	35	LVC MOS33*	3.300
Scalar ports (3)							
I_clk	IN		T5	<input checked="" type="checkbox"/>	34	LVC MOS33*	3.300
I_key	IN		R17	<input checked="" type="checkbox"/>	14	LVC MOS33*	3.300
I_rst_n	IN		P15	<input checked="" type="checkbox"/>	14	LVC MOS33*	3.300

## 10、配置管脚（七段数码管）

O\_led对应七段数码管的LED信号，O\_dx为七段数码管的段选信号

✓  O_dx (2)	OUT			
✓  O_dx[1]	OUT		C2	▼
✓  O_dx[0]	OUT		G2	▼
✓  O_led (7)	OUT			
✓  O_led[6]	OUT		B4	▼
✓  O_led[5]	OUT		A4	▼
✓  O_led[4]	OUT		A3	▼
✓  O_led[3]	OUT		B1	▼
✓  O_led[2]	OUT		A1	▼
✓  O_led[1]	OUT		B3	▼
✓  O_led[0]	OUT		B2	▼



名称	原理图标号	FPGA IO PIN
A0	LED0_CA	B4
B0	LED0_CB	A4
C0	LED0_CC	A3
D0	LED0_CD	B1
E0	LED0_CE	A1
F0	LED0_CF	B3
G0	LED0_CG	B2
DP0	LED0_DP	D5
A1	LED1_CA	D4
B1	LED1_CB	E3
C1	LED1_CC	D3
D1	LED1_CD	F4
E1	LED1_CE	F3
F1	LED1_CF	E2
G1	LED1_CG	D2
DP1	LED1_DP	H2
DN0_K1	LED_BIT1	G2
DN0_K2	LED_BIT2	C2
DN0_K3	LED_BIT3	C1
DN0_K4	LED_BIT4	H1
DN1_K1	LED_BIT5	G1
DN1_K2	LED_BIT6	F1
DN1_K3	LED_BIT7	E1
DN1_K4	LED_BIT8	G6

## 10、配置管脚（时钟及按钮）



I\_clk为时钟，I\_rst\_n为板子上的复位按钮S8，I\_key为板子上的通用按钮S1

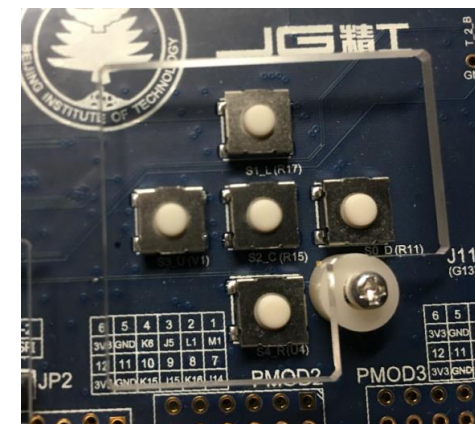
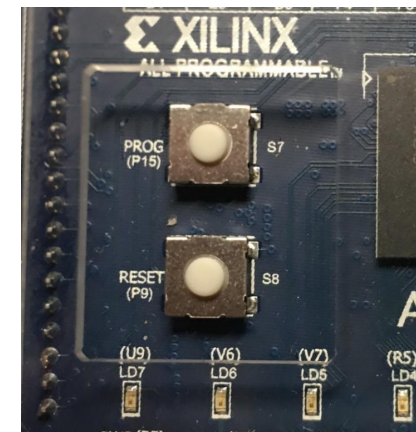
Scalar ports (3)

<input checked="" type="checkbox"/> I_clk	IN	T5	▼
<input checked="" type="checkbox"/> I_key	IN	R17	▼
<input checked="" type="checkbox"/> I_rst_n	IN	P15	▼

名称	原理图标号	FPGA IO PIN
时钟引脚	SYS_CLK	T5

名称	原理图标号	FPGA IO PIN
复位引脚	FPGA_RESET	P15

名称	原理图标号	FPGA IO PIN
S0	PB0	R11
S1	PB1	R17
S2	PB2	R15
S3	PB3	V1
S4	PB4	U4

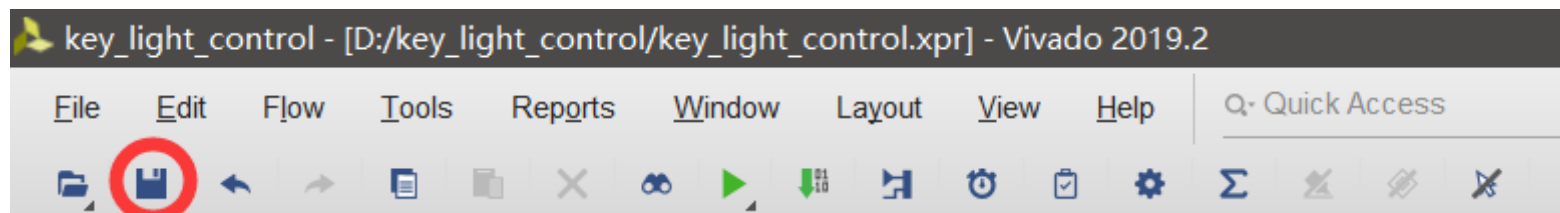




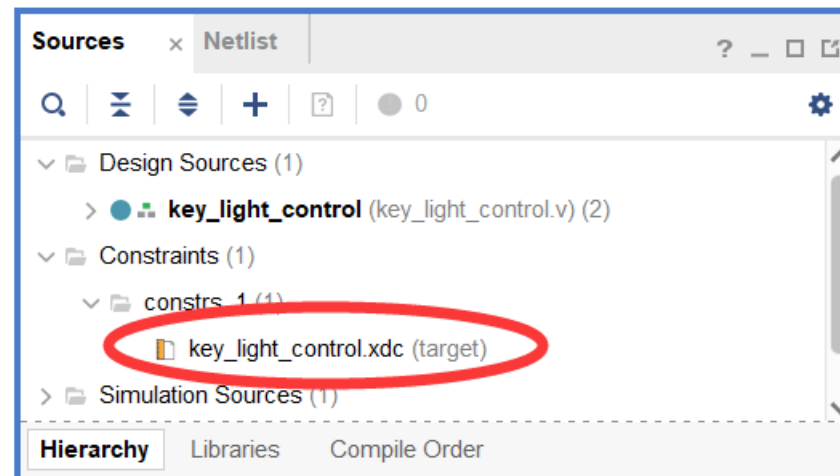
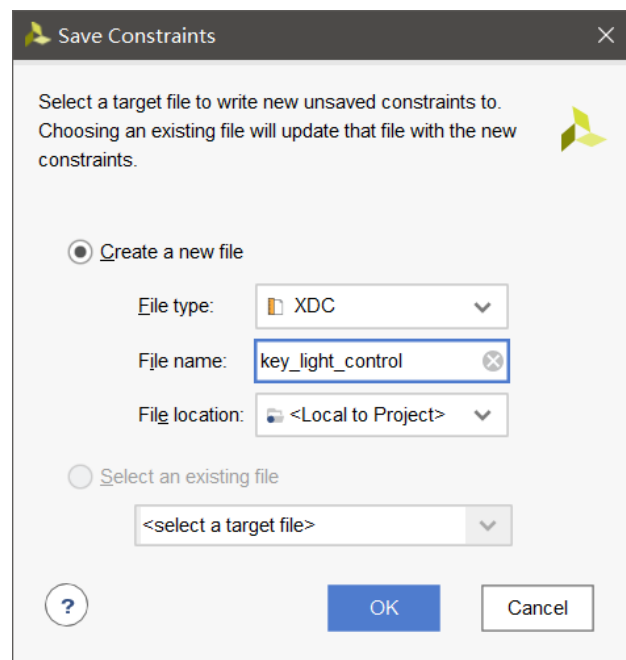
## 10、配置管脚



点击工具栏中的保存（或ctrl+S），保存约束文件



设置约束文件名称，点击OK按钮后，可以在Sources窗口中看到新的约束文件已经生成



## 11、生成比特流文件



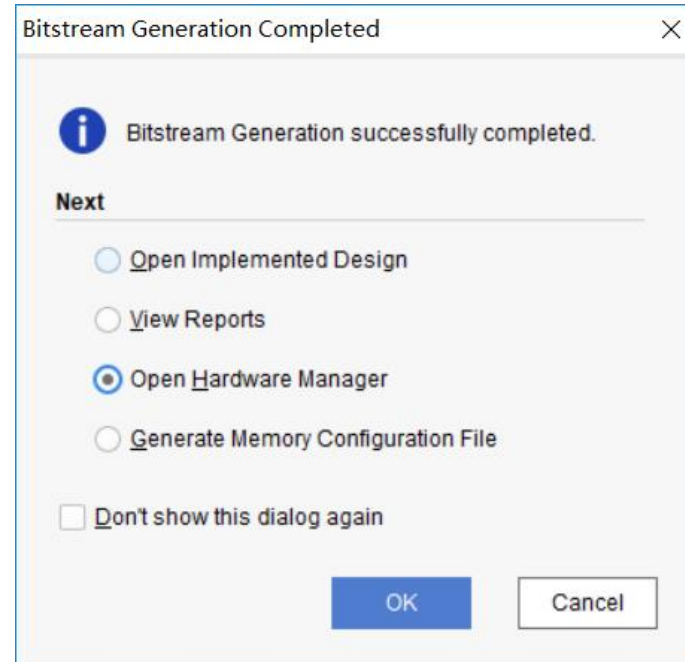
- 点击左侧菜单栏Generate Bitstream选项一直点击OK直到开始生成比特流文件
- 生成成功后弹出如图界面，选择Open Hardware Manager

PROGRAM AND DEBUG



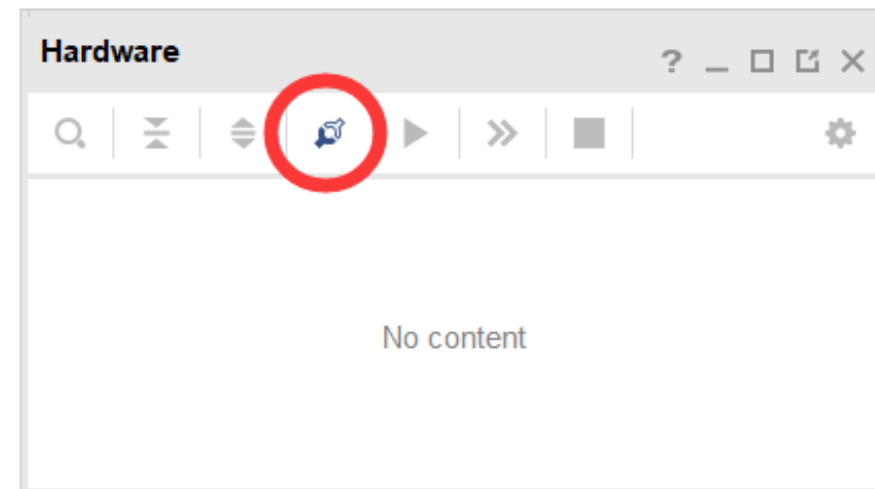
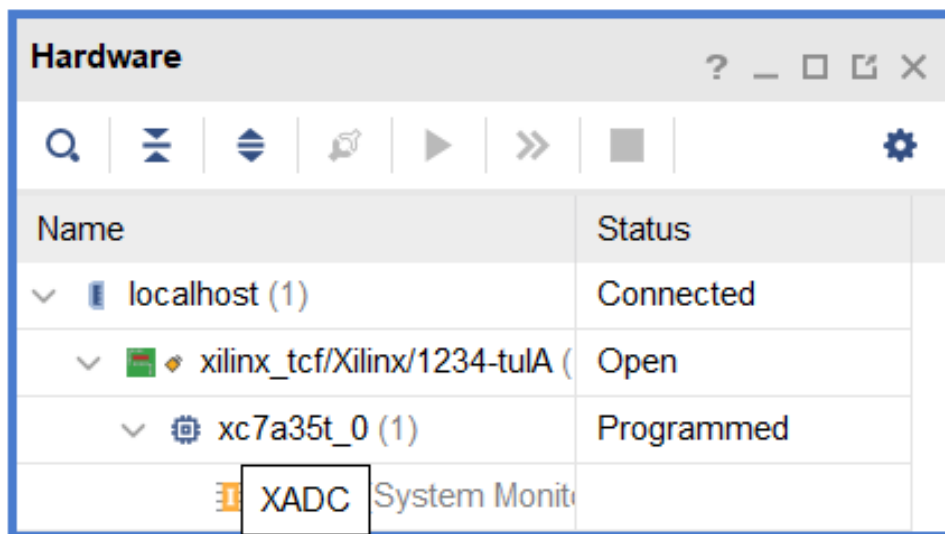
Generate Bitstream

> Open Hardware Manager



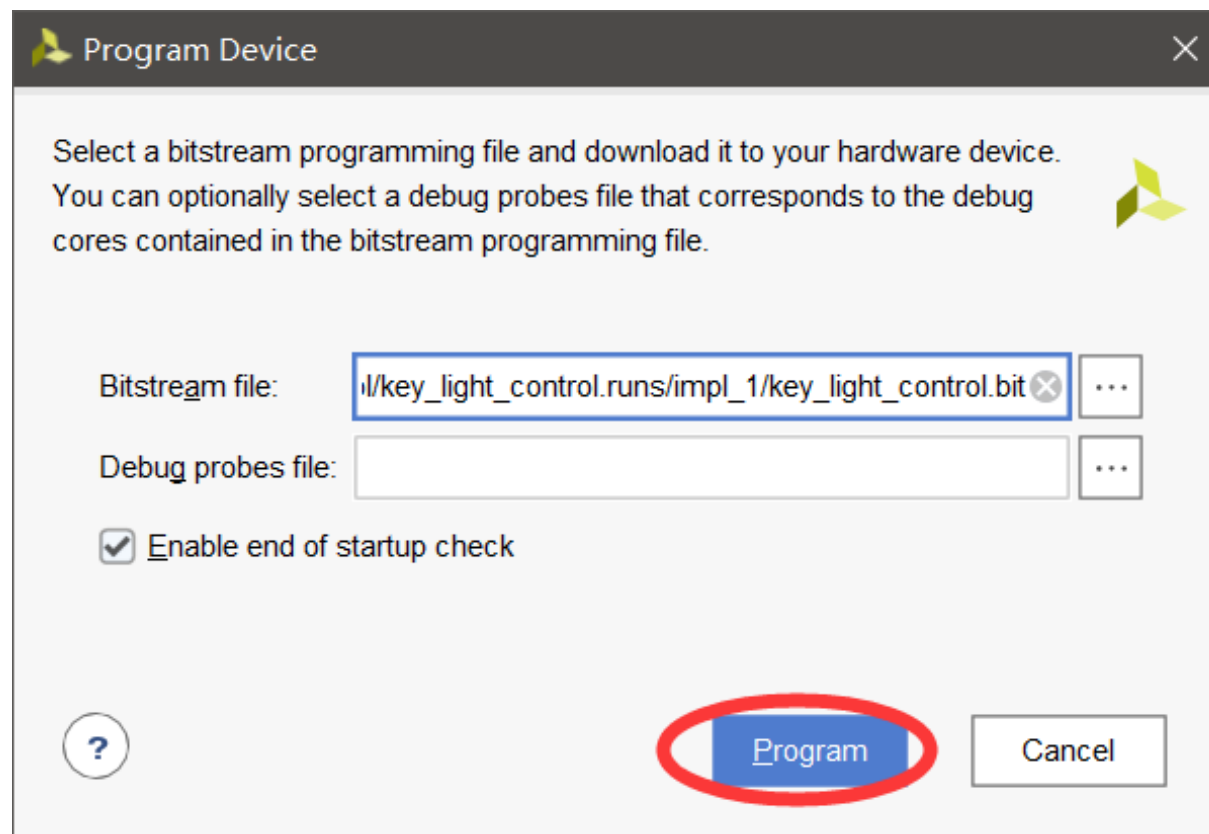
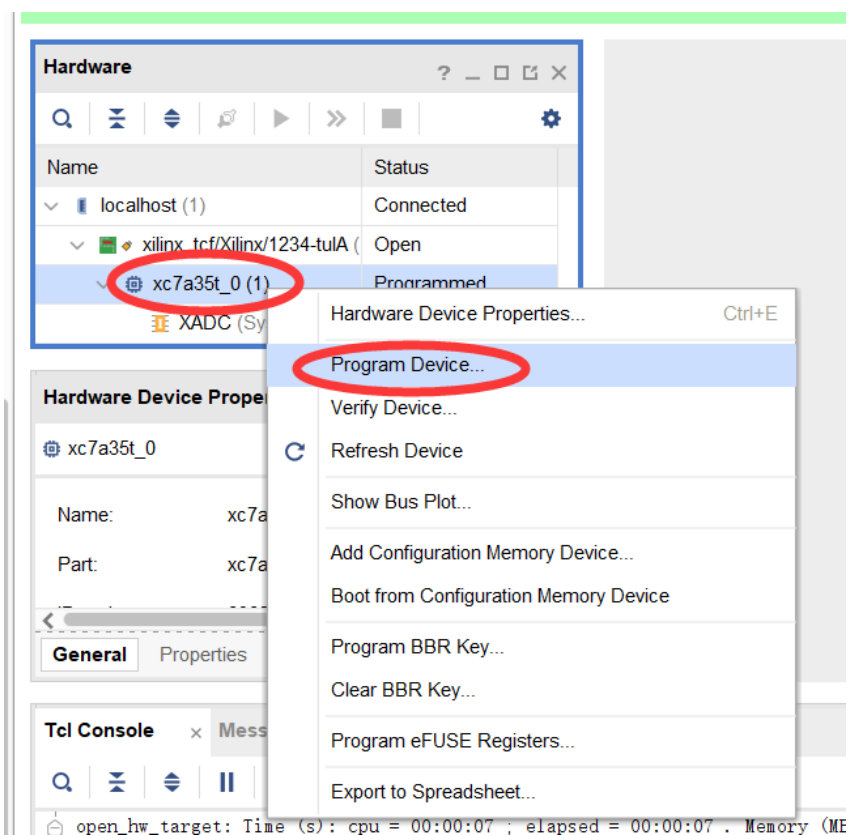
## 12、连接开发板

- 将EES-338通过USB，连接到电脑，打开开关后等待连接
- 点击auto connect 自动连接开发板
- 连接完成显示如下图



## 13、下载代码并验证

- 右键点击xc7a35t\_0(1)，选择Program Device点击Program



## 14、效果展示

- 可以看到最右侧数码管显示数字，每按下一次S1按钮，数字就会自增一。

