

Department of
**Computer Science
& Engineering**



计算机系统结构实验指导书-LAB01

1. OVERVIEW

1.1 实验名称

FPGA 基础实验：LED Flow Water Light

1.2 实验目的

1. 掌握 Xilinx 逻辑设计工具 Vivado 的基本操作
2. 掌握使用 VerilogHDL 进行简单的逻辑设计
3. 使用功能仿真
4. 使用 I/O Planing 添加管脚约束
5. 生成 Bitstream 文件
6. 上板验证

1.3 实验预计时间

90 分钟

1.4 实验报告与验收办法

本实验不需提交实验报告，实验完毕需验收登记

建议：为方便使用，可建立 E:\archlabs 文件夹，用于放置实验工程、给定的 IP 核或实验资料。

2. 实验步骤

2.1 新建工程

1. 启动桌面 Vivado 2018.3 开发工具或点击左下角窗口选择 Xilinx Design Tools-> Vivado 2018.3，如图所示

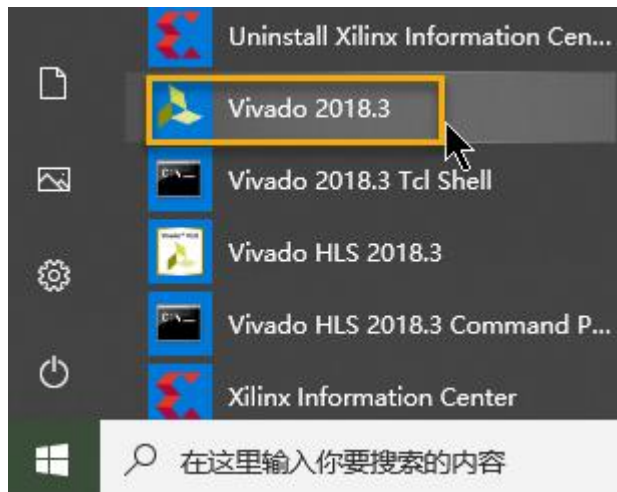


图 2-1 运行 Vivado

2. 点击 Create Project

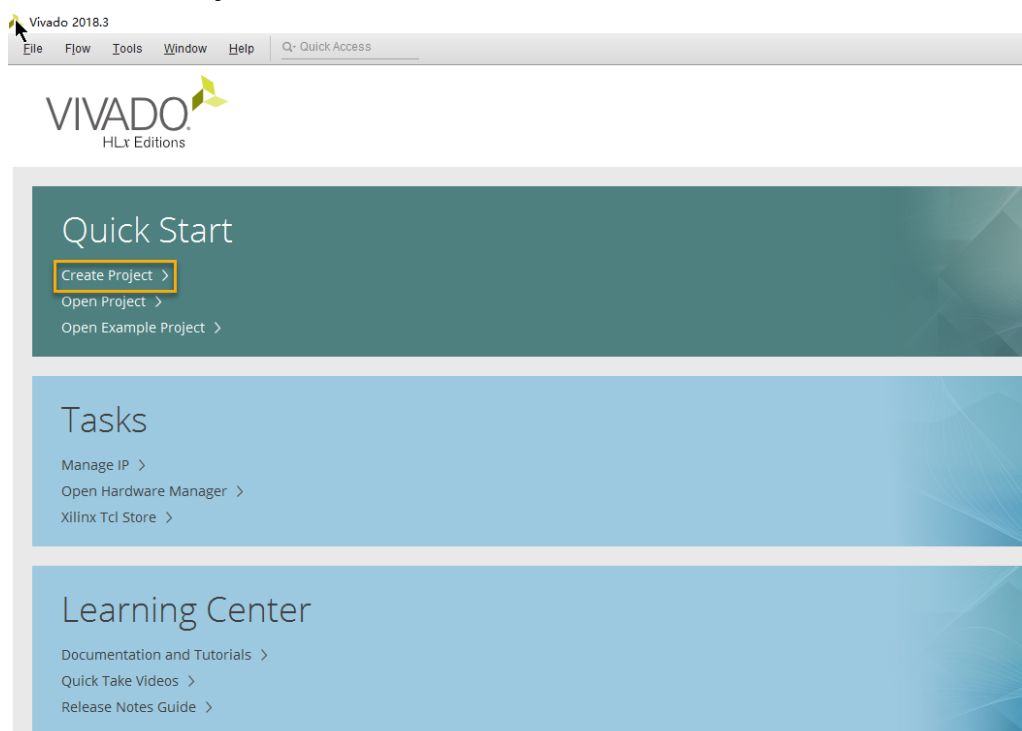


图 2-2 新建工程

3. 弹出 New Project 向导，由此建立一个新工程，点击 Next

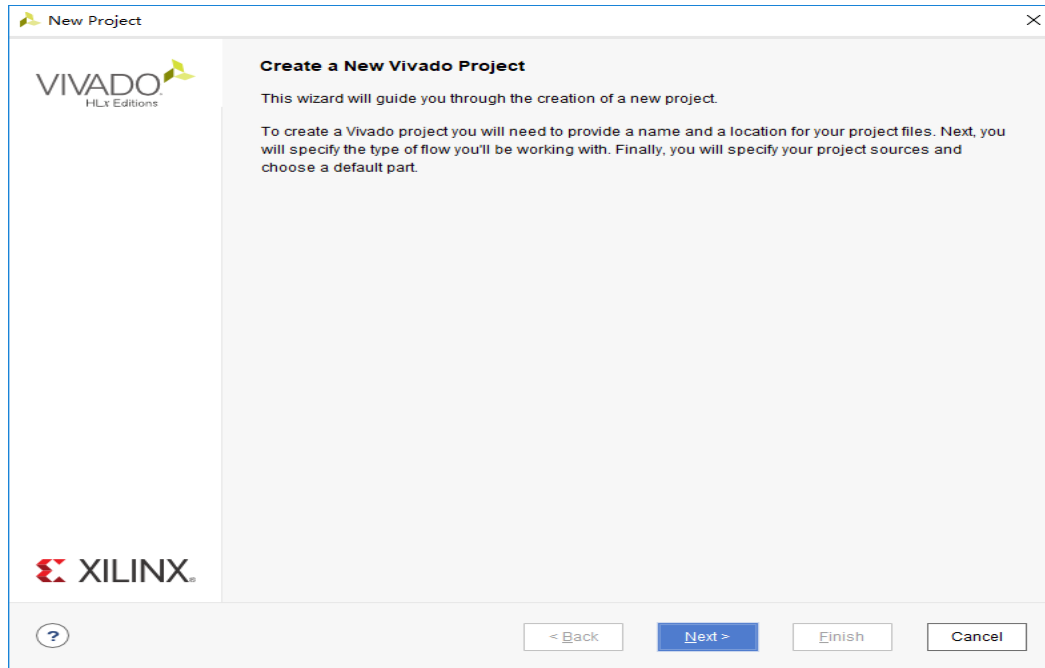


图 2-3 新建工程向导

4. 输入工程名称 **lab01**，建议工程位置 E:/archlabs，确认勾选 **Create project subdirectory** 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

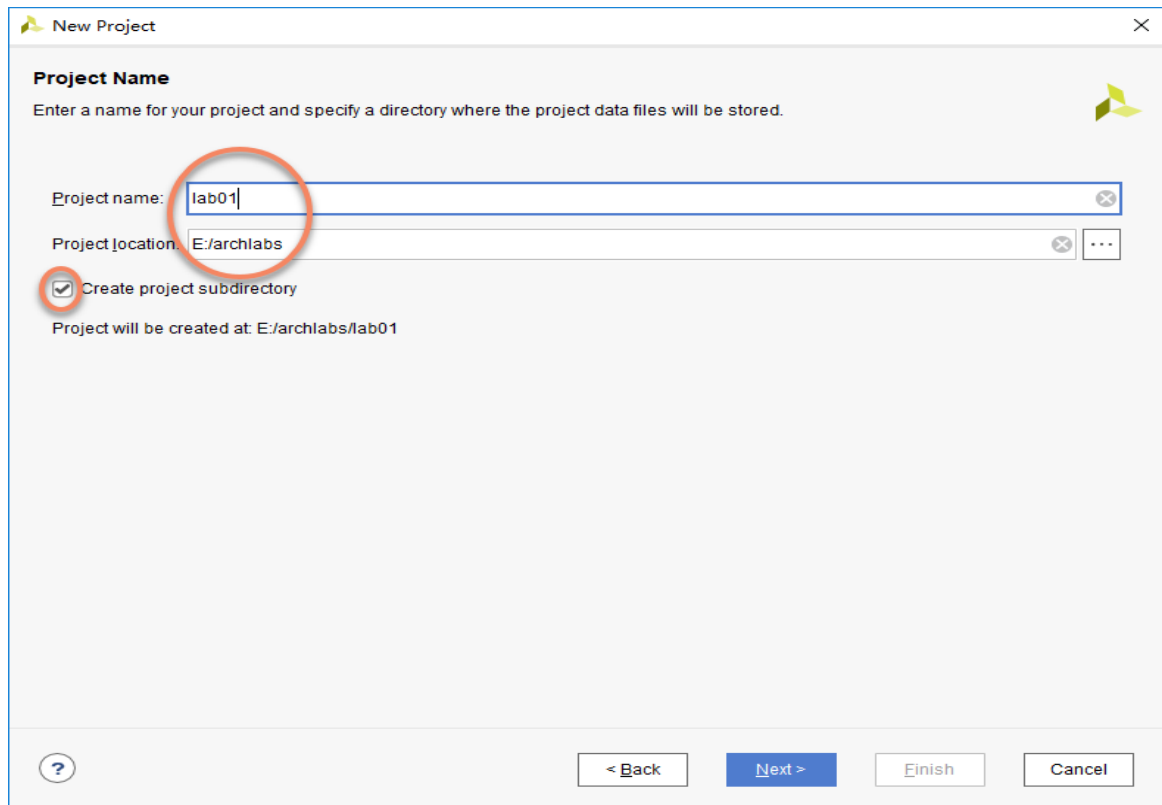


图 2-4 工程名称和路径

5. 选择 **RTL Project** 工程类型，勾选 **Do not specify sources at this time** 不在创建工程时决定 sources 文件，点击 **Next**

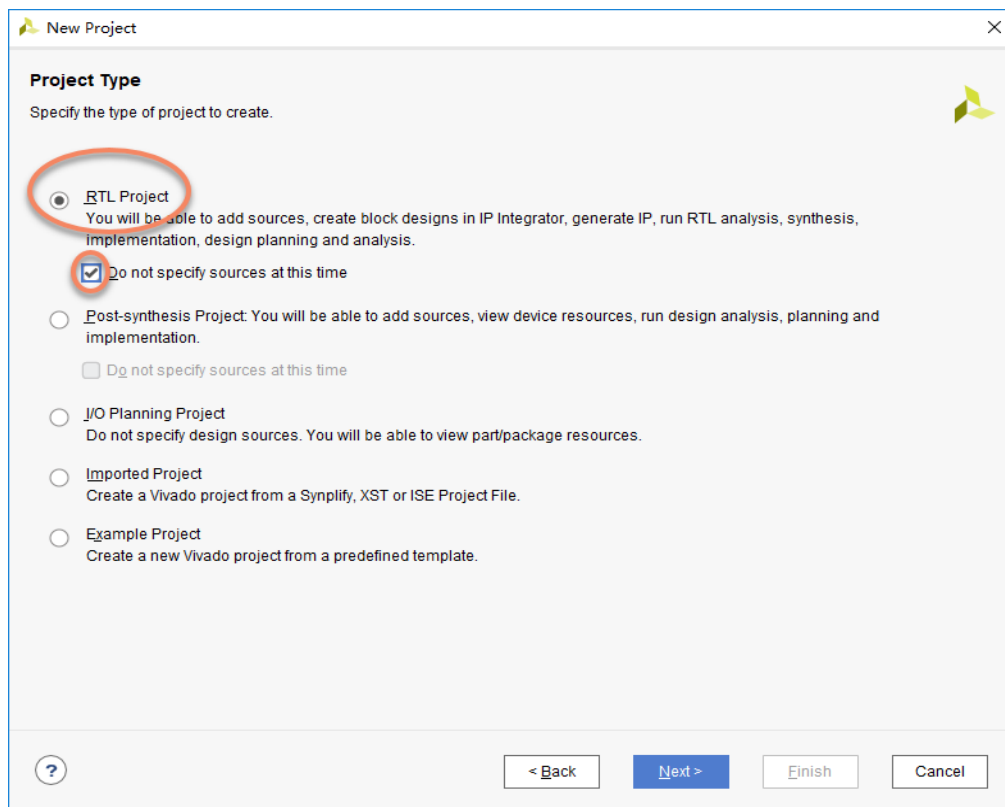


图 2-5 RTL 工程

6. 选择 SWORD4.0 的 FPGA 参数：Family 选 **Kintex-7**，Package 选 **ffg676**，Speed grade 选 **-2**；再找到的具体型号中选 **xc7k325tffg676-2**，点击 **Next**

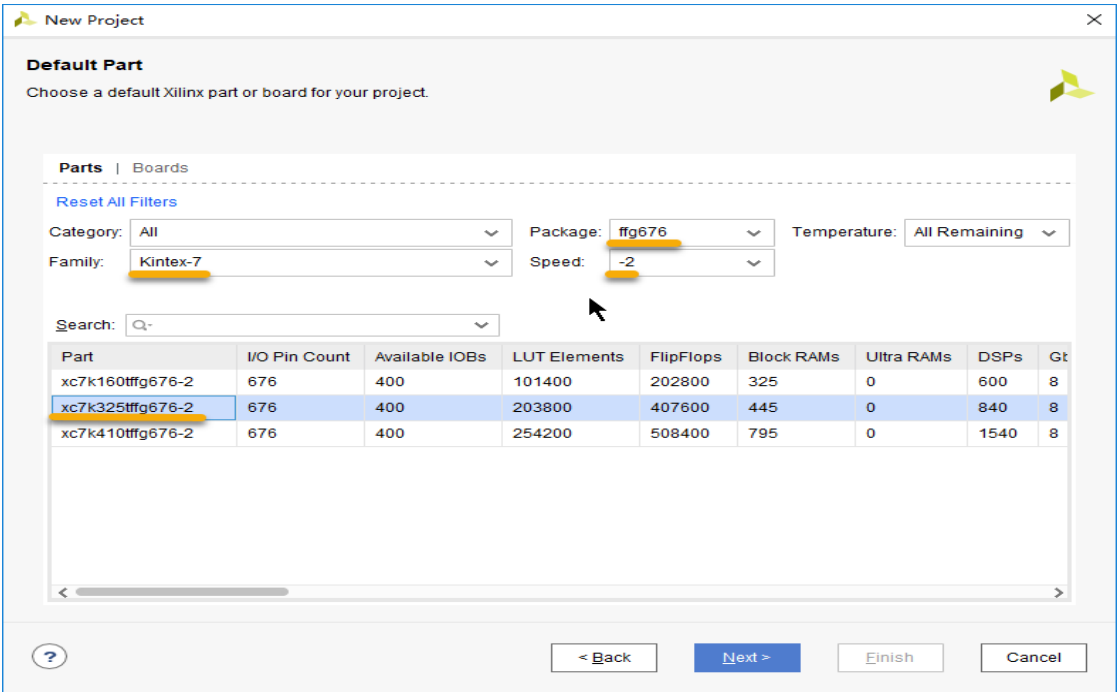


图 2-6 FPGA 型号参数

7. 弹出新工程信息综述，点击 **Finish** 结束工程的创建。

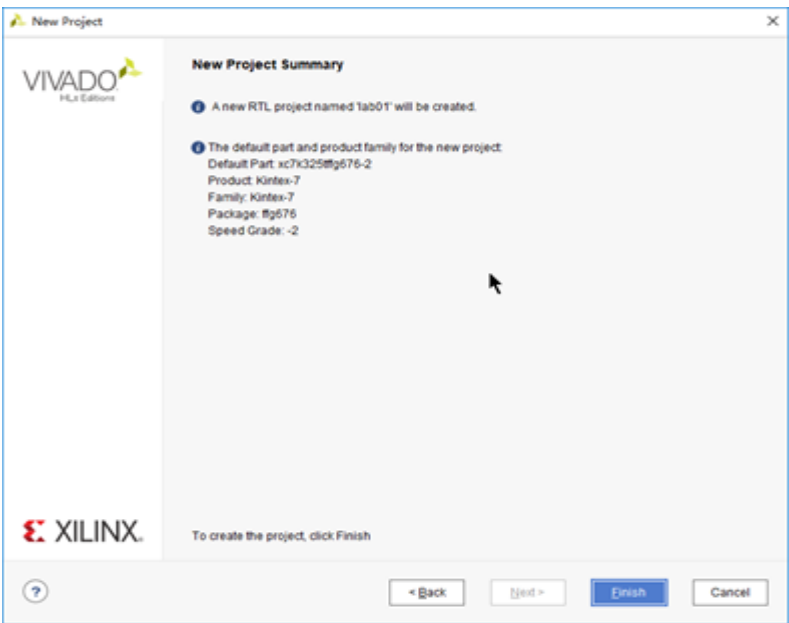


图 2-7 新建工程完毕

2.2 Vivado 整体界面

如图 2-8 是 Vivado2018.3 的整体界面，大致分为四个区：

1. 左侧区 Flow Navigator 包含整个开发流程，像 Project Settings、Run Simulation、Run Synthesized 和 Generate Bitstream 等；
2. 中间区通常显示当前工程包含的文件树结构，提供工程文件的管理；

3. 右侧区会显示工程信息、打开的编辑文件等；
4. 下部区显示各种信息状态。

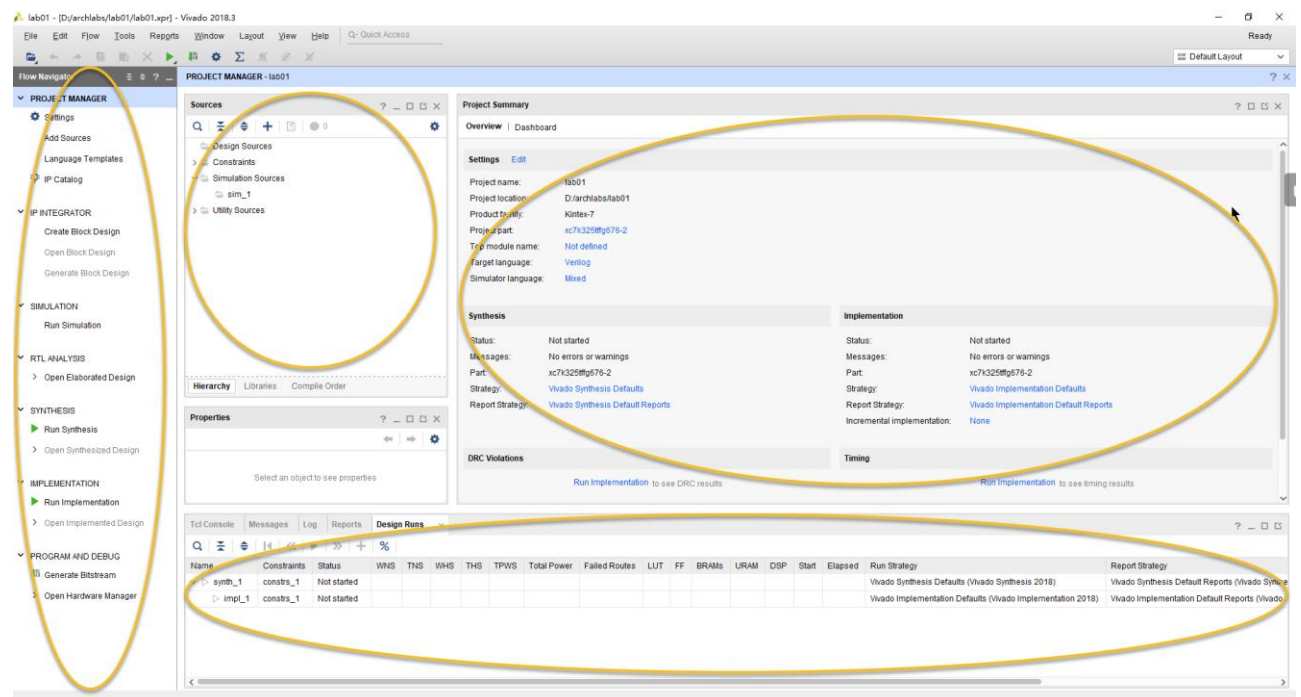


图 2-8 是 Vivado2018.3 的整体界面

注意，上图部分功能区域会随当前进行的操作而显示不同的内容

2.3 添加文件

1. 如下图，点击 左侧区 Flow Navigator 下的 Project Manager->Add Sources 或中间 区 Sources 的 “+” 号，打开 Add Sources 对话框

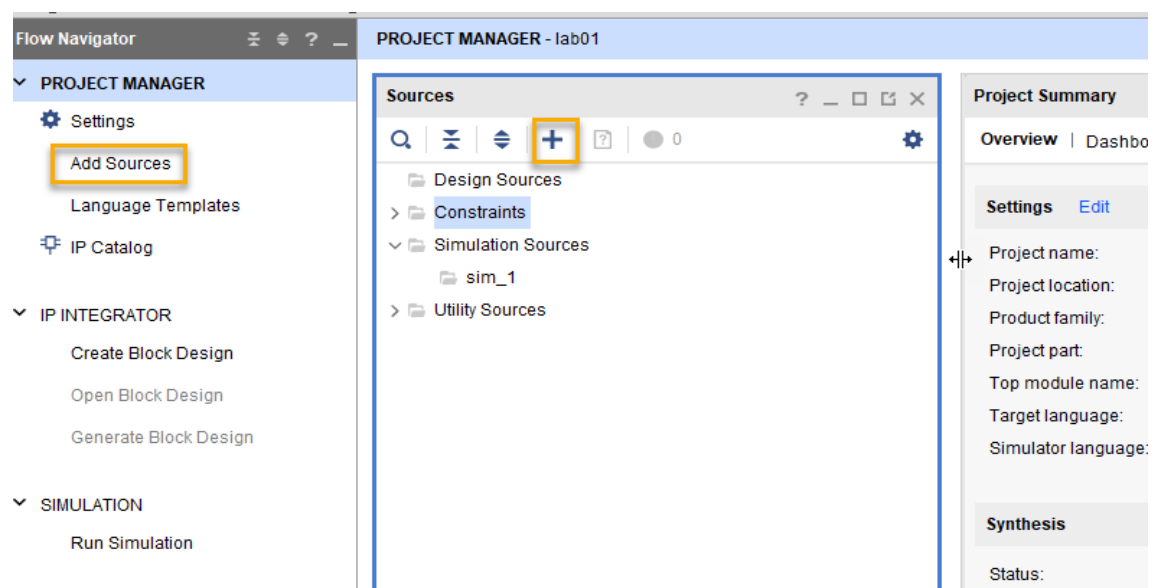
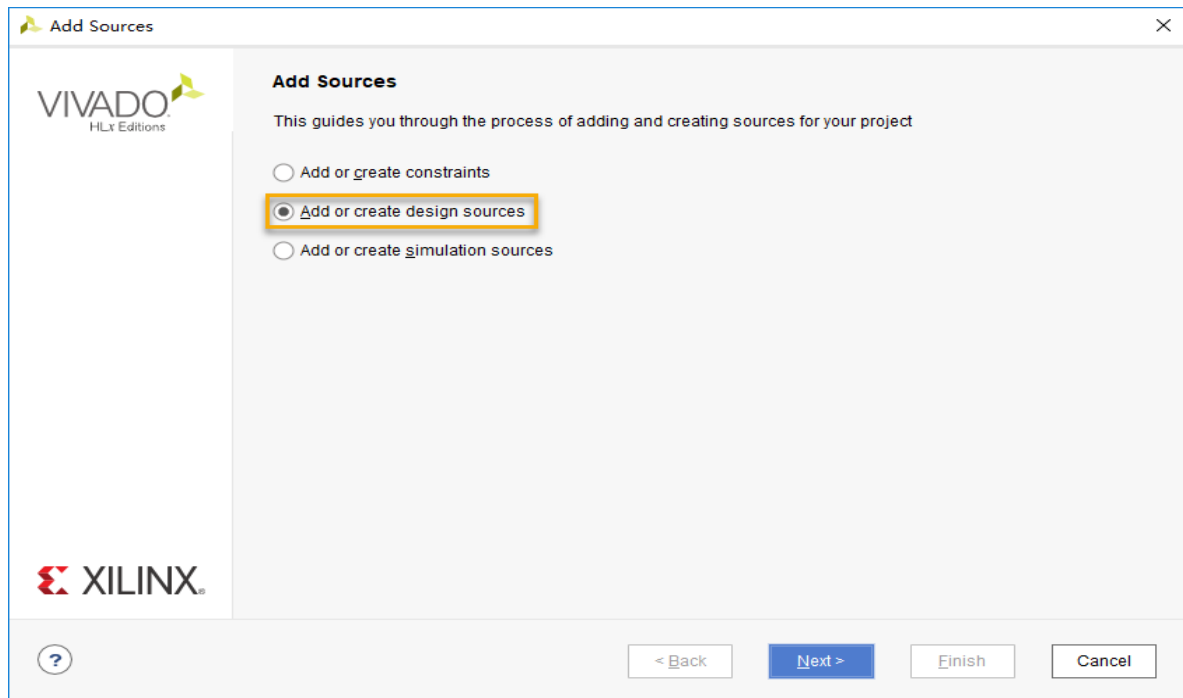
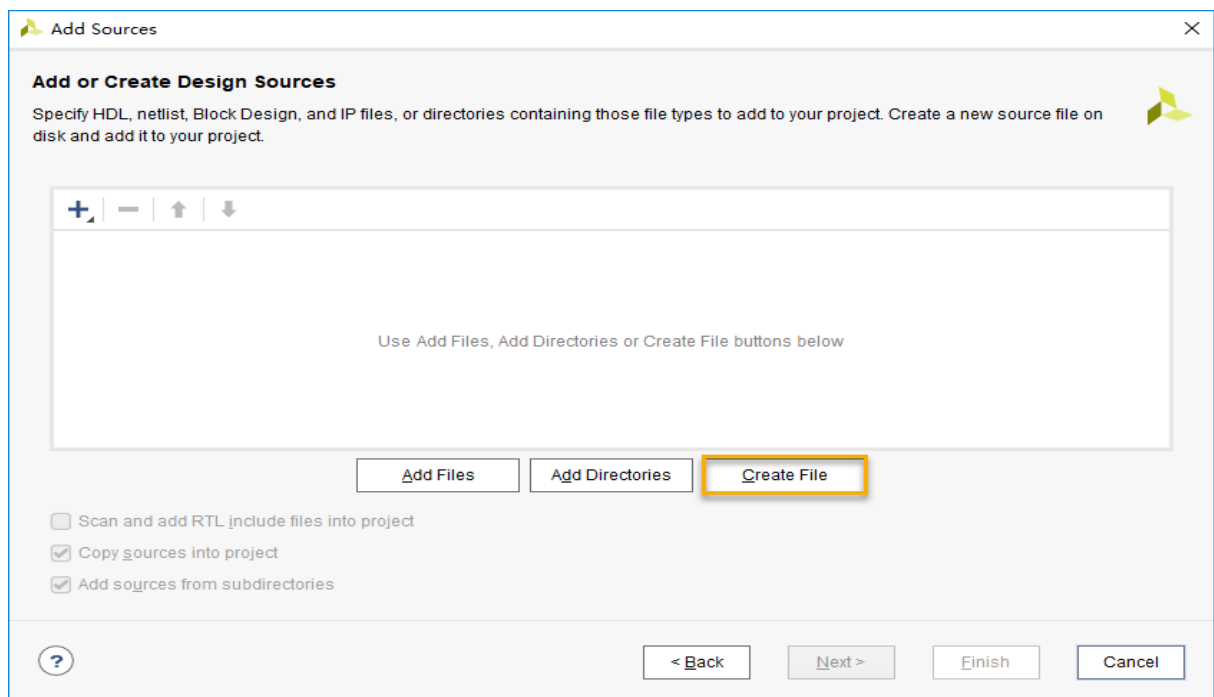


图 2-8 增加文件

2. 选择第二项 **Add or Create Design Sources**，用来添加或新建 Verilog HDL 源文件，点击 **Next**

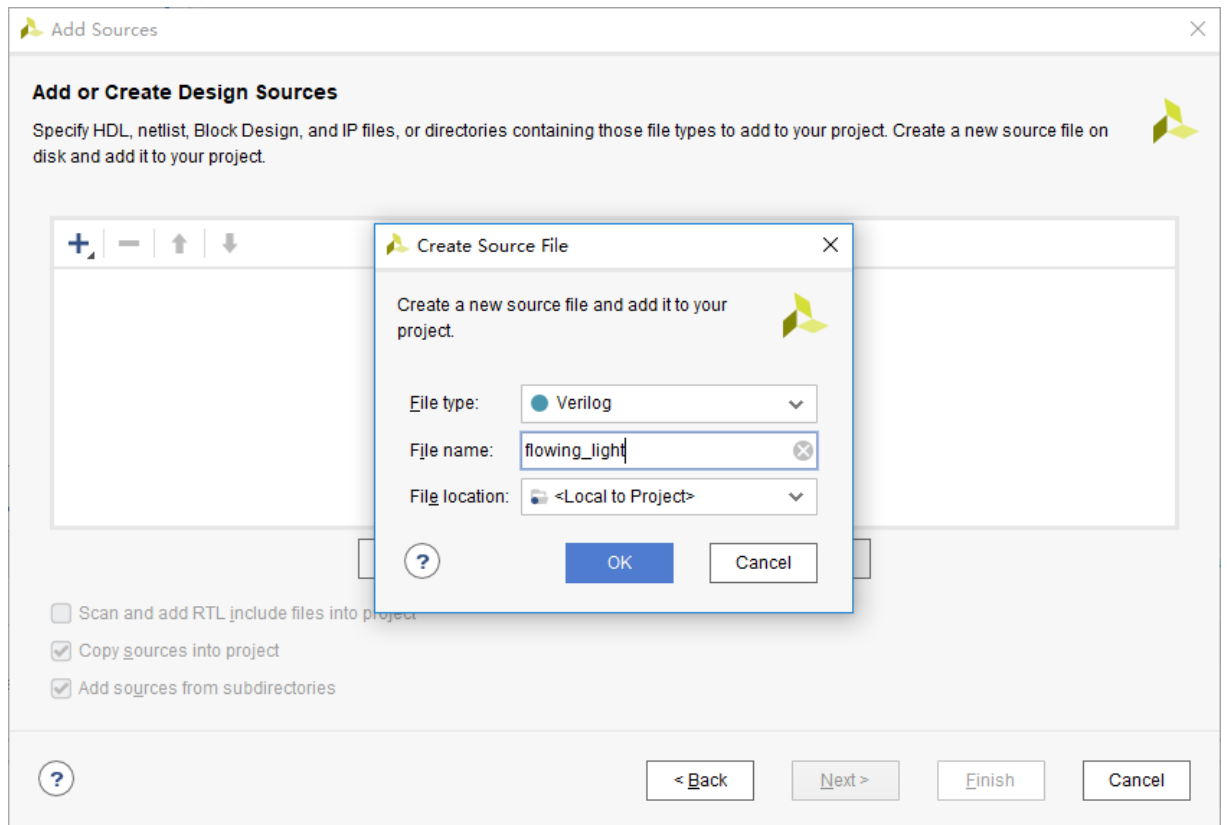


3. 若已有代码模块文件或 IP 核文件，可选 **Add Files** 以添加所需文件。本实验这里要新建代码文件，故选择 **Create File** 项。

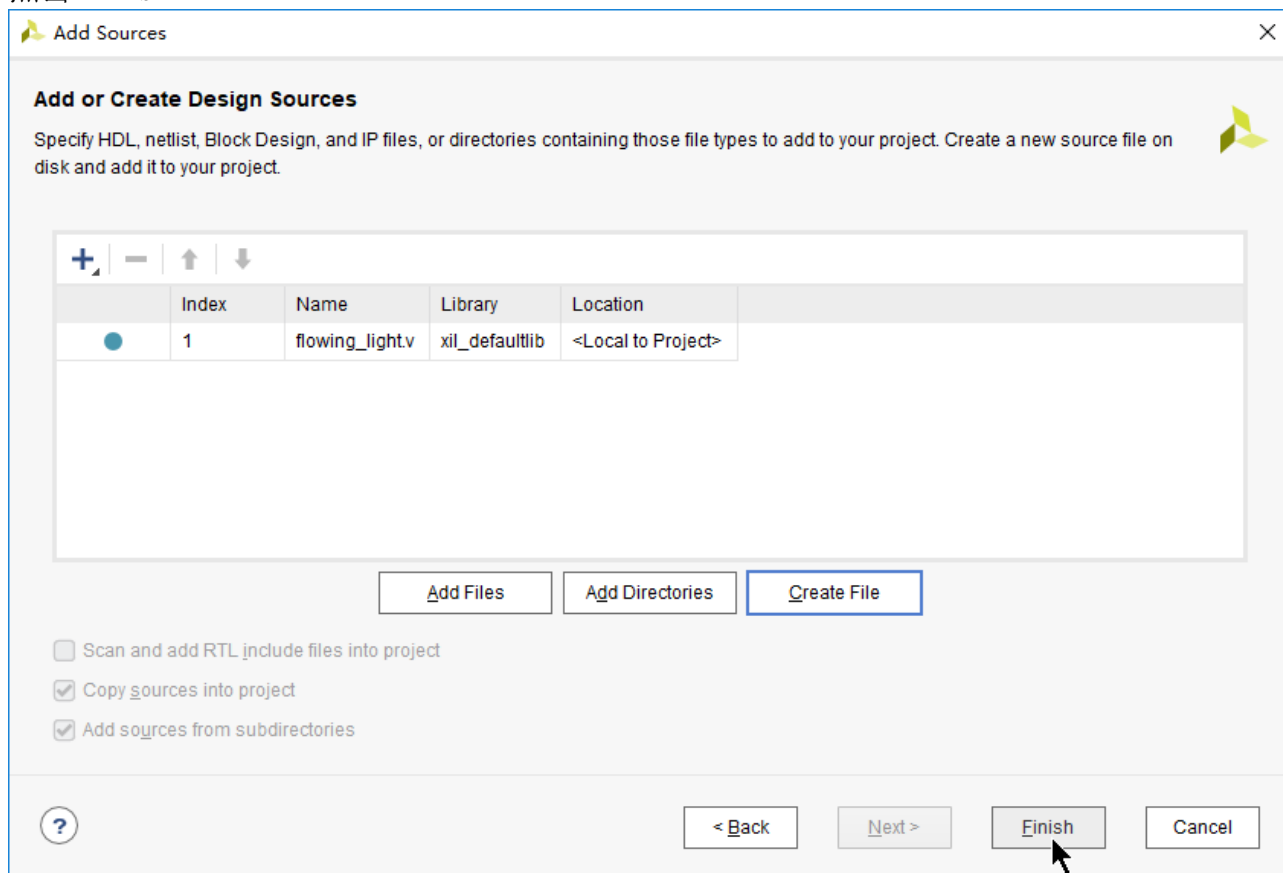


4. 弹框 **Create Source File** 中文件名输入 `flowing_light`，点击 **OK**

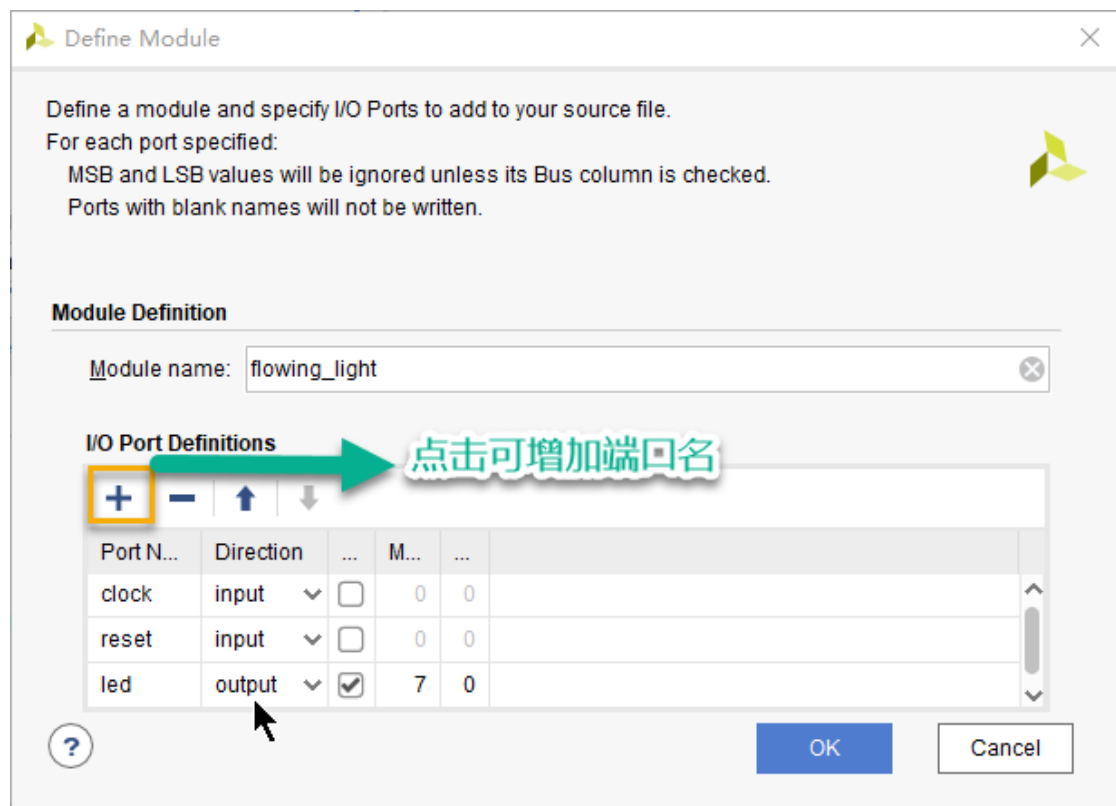
注：名称中不可出现中文和空格



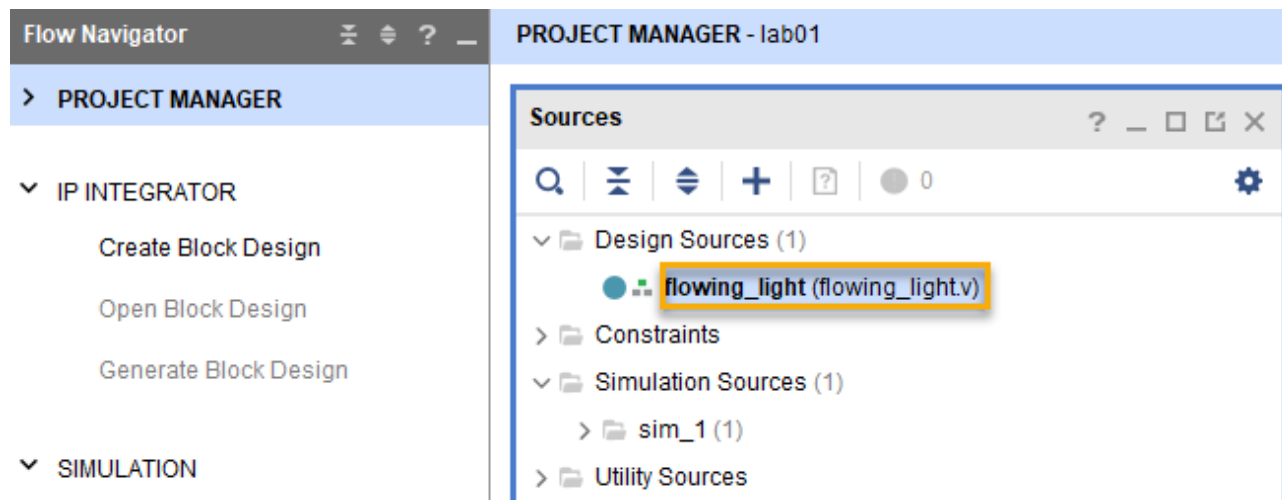
5. 点击 Finish



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, 若端口为总线型, 勾选 Bus 选项, 并由 MSB 和 LSB 确定总线宽度; 完成后点 OK



7. 新建的 flowing_light 存在于 Sources 中的 Design Sources 下;



8. 双击 flowing_ligh, 添加如下设计代码, 并点击保存按钮

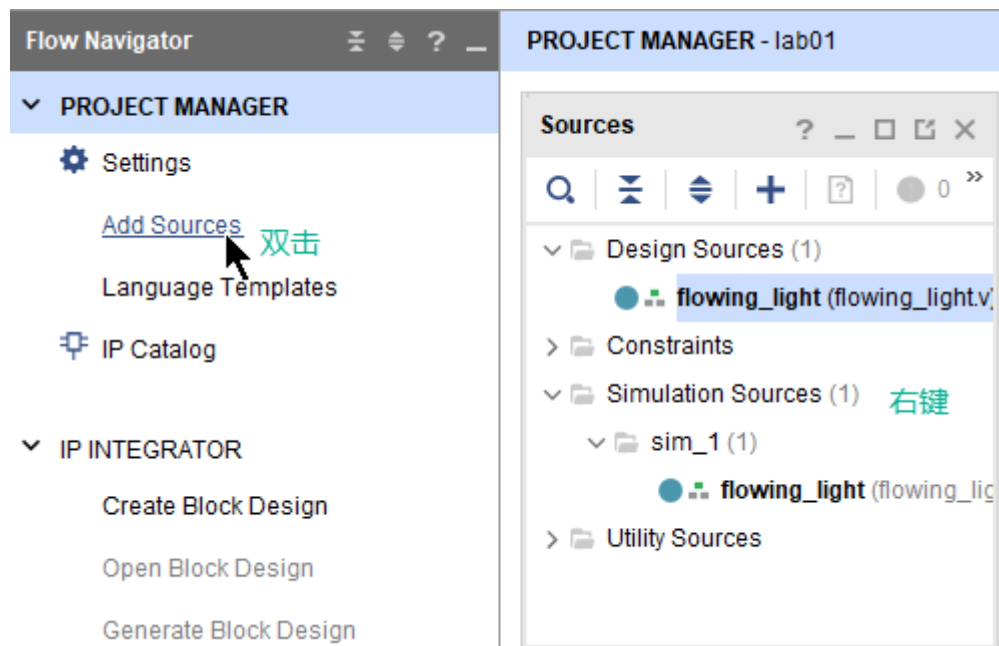
```

1  reg [23 : 0] cnt_reg;
2  reg [7 : 0] light_reg;
3  always @ (posedge clock)
4      begin
5          if (reset)
6              cnt_reg <= 0;
7          else
8              cnt_reg <= cnt_reg + 1;
9          end
10     always @ (posedge clock )
11         begin
12             if (reset)
13                 light_reg <= 8'h01;
14             else if (cnt_reg == 24'hffffff)
15                 begin
16                     if (light_reg == 8'h80)
17                         light_reg <= 8'h01;
18                     else
19                         light_reg <= light_reg << 1;
20                 end
21             end
22     end
23     assign led = light_reg;

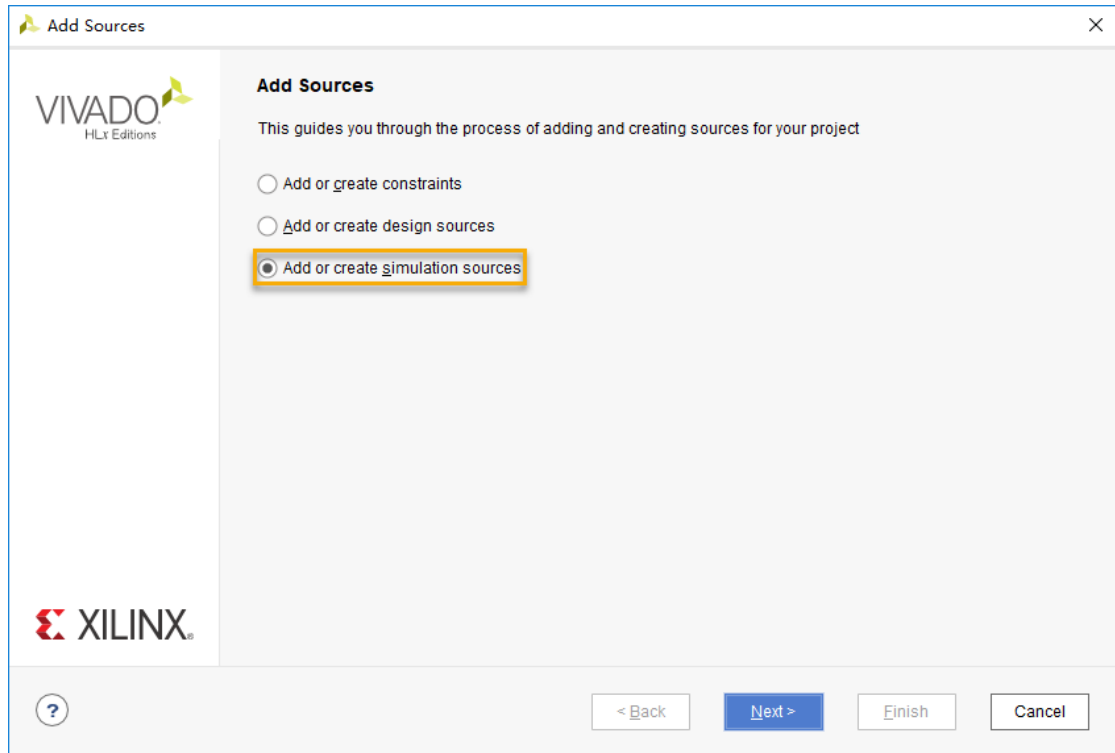
```

2.4 功能仿真

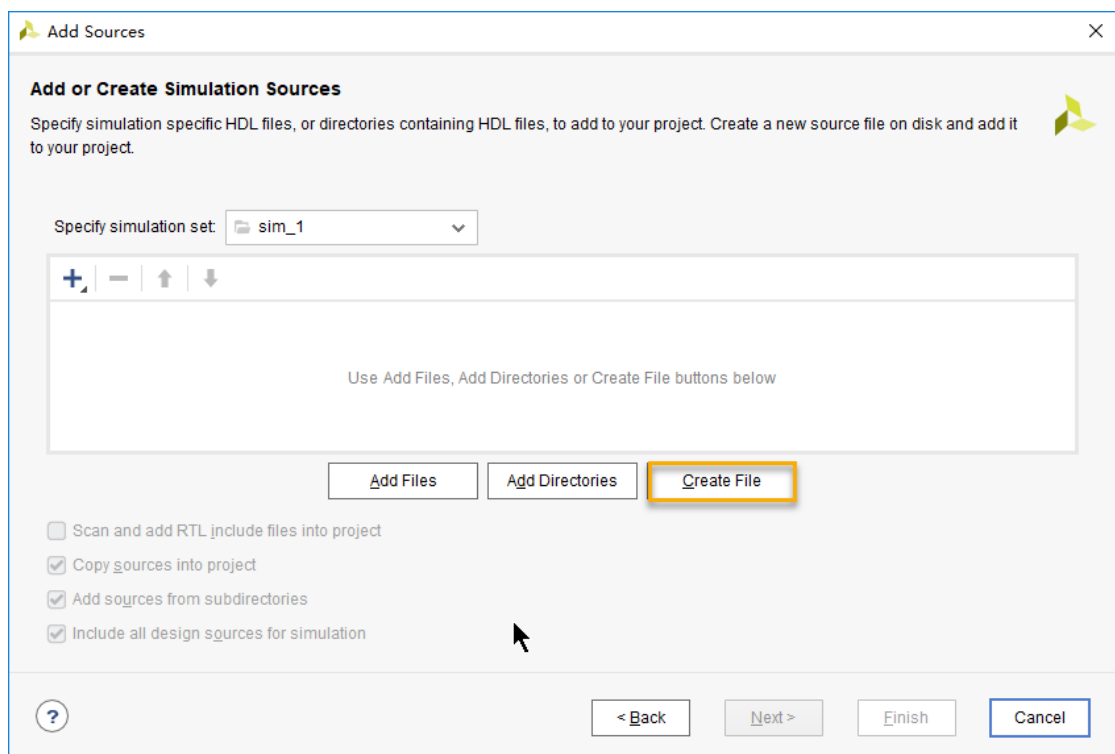
1. 创建激励测试文件。在中间区 Source 中右击选择 Add Source 或于左侧区 PROJECT MANAGER 下选择 Add Source



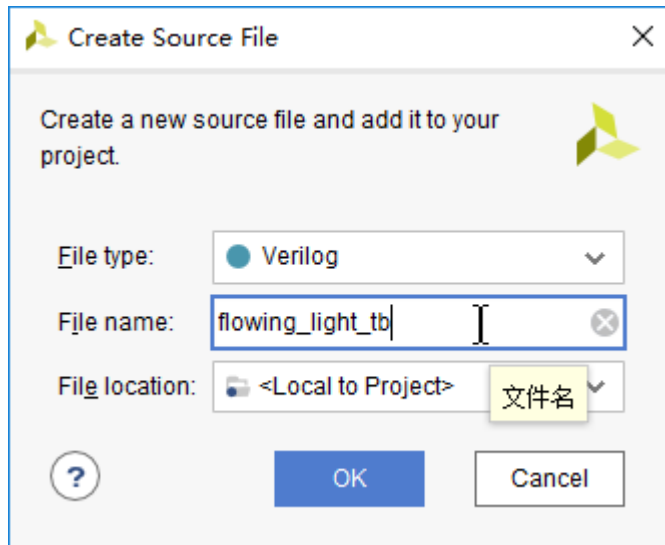
2. 在 Add Source 中选择第三项 Add or Create Simulation Source, 点击 Next



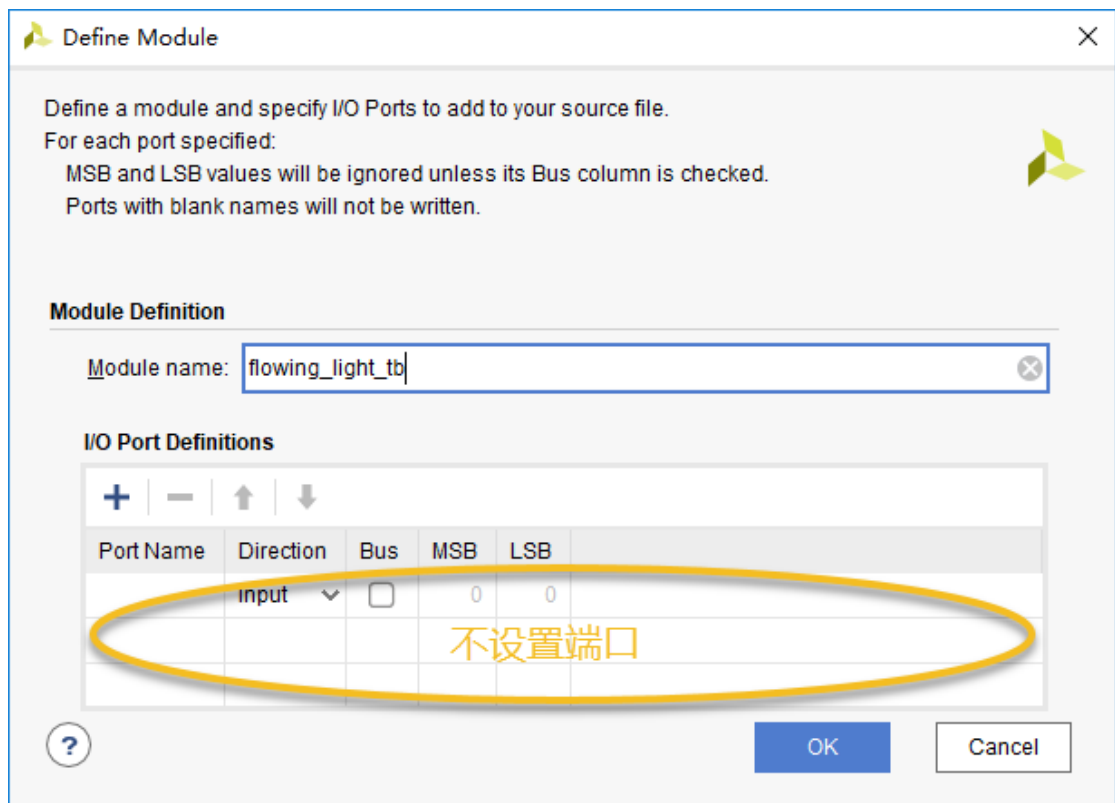
3. 选择 Create File 创建一个仿真激励文件



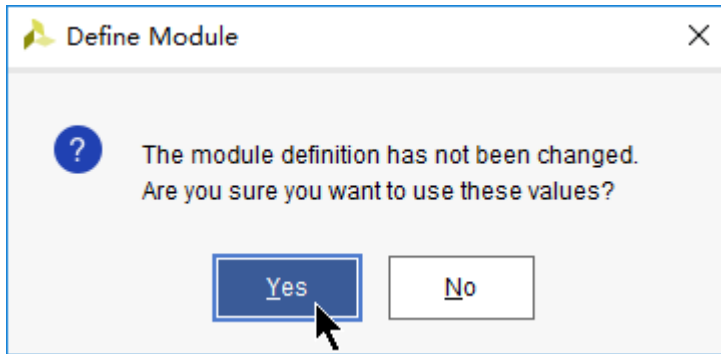
4. 输入激励文件名称可以是 flowing_light_tb, 点击 OK



5. 完成之后点击 **Finish**，创建激励文件是不需要对外端口，再点击 **OK**



6. 在弹出的对话框中点击 **Yes**



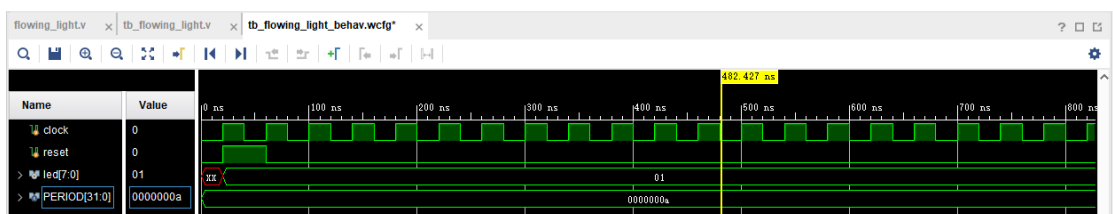
7. 在 Source 区 Simulation Sources 下，打开仿真测试文件 `flowing_light_tb`，在其中对要仿真的模块进行实例化和激励代码的编写（并点击保存），如下图所示

```

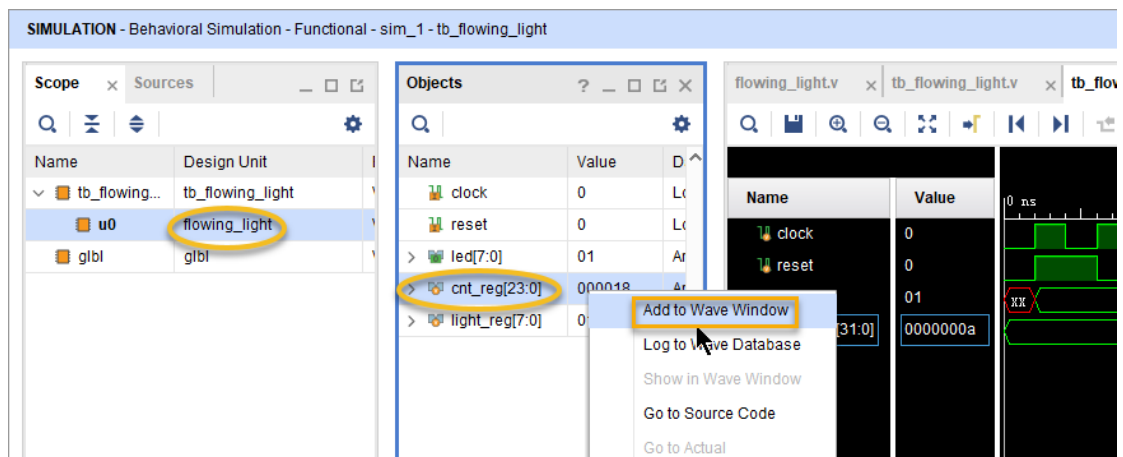
1
2  reg  clock;
3  reg  reset ;
4  wire [7:0] led;
5
6  flowing_light u0(
7      .clock(clock),
8      .reset(reset),
9      .led(led));
10
11  parameter PERIOD= 10;
12
13  always #(PERIOD*2) clock = !clock;
14
15  initial begin
16      clock = 1'b0;
17      reset = 1'b0;
18      #(PERIOD*2) reset = 1'b1;
19      #(PERIOD*4) reset = 1'b0;
20
21      // #580; reset = 1'b1;
22  end

```

8. 在左侧 Flow Navigator 中点击 Simulation 下的 Run Simulation 选项，并选择 Run Behavioral Simulation。下图为仿真运行后所得仿真波形图



注意 1) Scopes 窗口里可选中需要查看信号的模块，选中感兴趣的信号，点击右键并选 Add To Wave Window，便可增加信号到仿真波形图中；



2) 对长信号而言，有时十六进制看起来更顺。选择该信号并点击右键，通过 Radix 菜单可以改变信号的显示方式；

3) 增加信号后需要使用 Run 菜单下的 Restart，重新开始。

4) 使用 Run for...仿真指定时间长的波形。

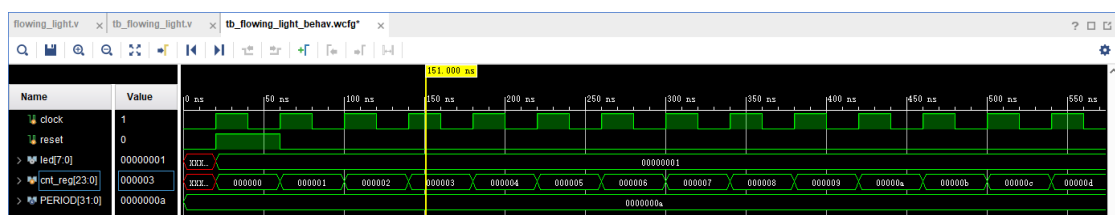
5) 通过键盘 Ctrl+ “-” 和 Ctrl+ “+” 可以对波形图进行缩放

6) 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



通过查看仿真波形和逻辑设计应有的波形对比，可知设计的电路是否正常工作。

9. 最终得到的仿真效果图如下

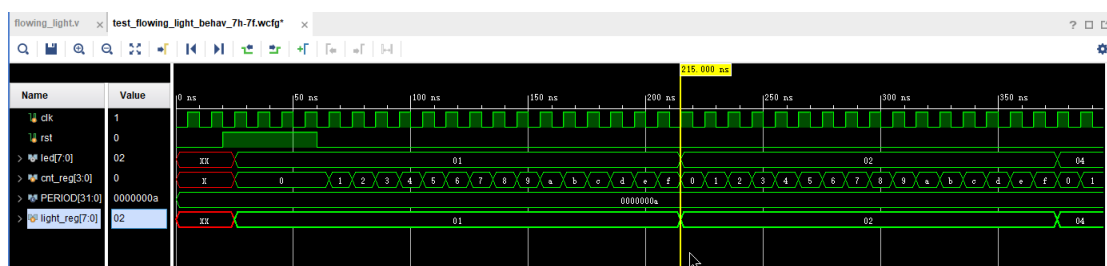


从图中可以看到：

当 reset 信号为 1 时，计数器 cnt_reg 被初始化为零，输出信号 led 被初始化为 00000001；

当 reset 信号为 0 时，计数器在每个时钟信号上升沿加 1 计数，直至加到 24 位值全为 1 时，输出信号左移 1 位。

本仿真运行周期不够，计数器并没加到 24 位全是 1 而波形显示早已结束。我们可以将计数器的位数改变或者改变计数器计数值，以便较快速达到左移条件，如下图仿真结果



2.5 工程实现

由于 SWORD4.0 平台板载 200MHz 时钟振荡器，属高频时钟，下载验证时需用了差分时钟。原 flowing_light 模块需做时钟方面的修改

1. flowing_light.v 修改如下图方框中：

```

1
2 module flowing_light(
3     input clock_p,
4     input clock_n,
5     input reset,
6     output [7:0] led
7 );
8     reg [23:0] cnt_reg;
9     reg [7:0] light_reg;
10
11     IBUFGDS IBUFGDS_inst (
12         .O(CLK_i),
13         .I(clock_p),
14         .IB(clock_n)
15     );
16
17     always @ (posedge CLK_i)
18     begin
19         if (!reset) //板子上的按钮实际上按下去是低电平，故要取反
20             cnt_reg <= 0;
21         else
22             cnt_reg <= cnt_reg + 1;
23     end
24
25     always @ (posedge CLK_i )
26     begin
27         if (!reset)
28             light_reg <= 8'h01;
29         else if (cnt_reg == 24'hffffff)
30             begin
31                 if (light_reg == 8'h80)
32                     light_reg <= 8'h01;
33                 else
34                     light_reg <= light_reg << 1;
35             end
36     end
37
38     assign led = light_reg;
39 endmodule

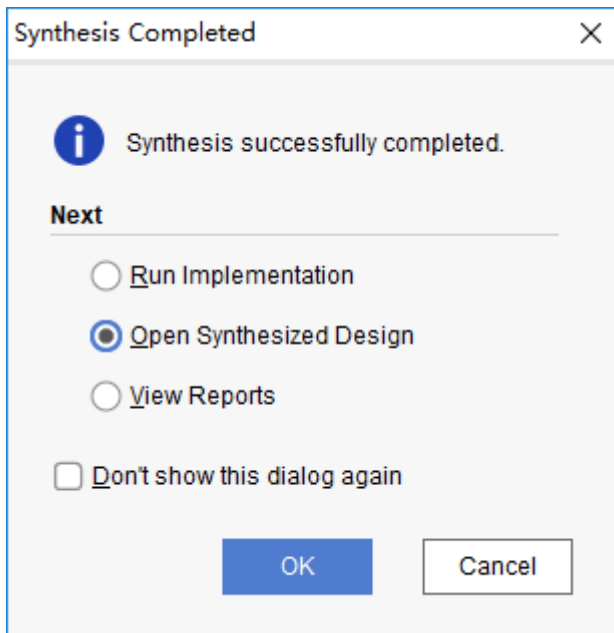
```

2. 添加约束文件

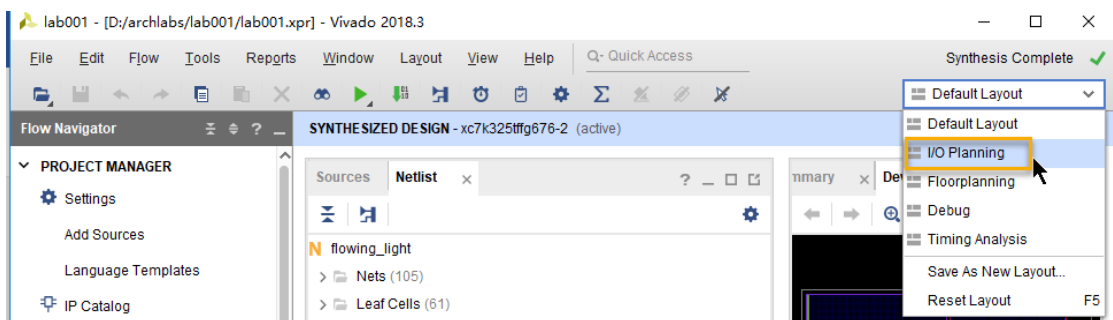
有两种方法可以添加约束文件，一是利用 Vivado 中 IO planning 功能，二是直接新建xdc的约束文件，手动输入约束命令。

- 1) 利用 IO planning: 点击 Flow Navigator 中 Synthesis 中的 Run

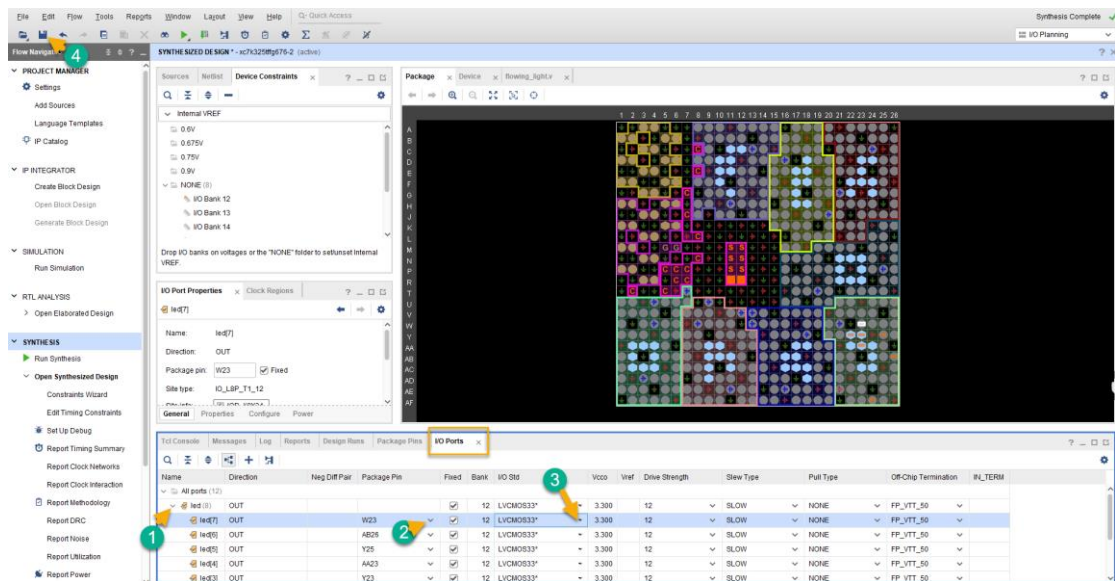
Synthesis，先对工程进行综合。综合完成之后，弹出以下对话框，选择 Open Synthesized Design，点击OK



应看到如下界面。若无，在图示方框位置选择 IO planning



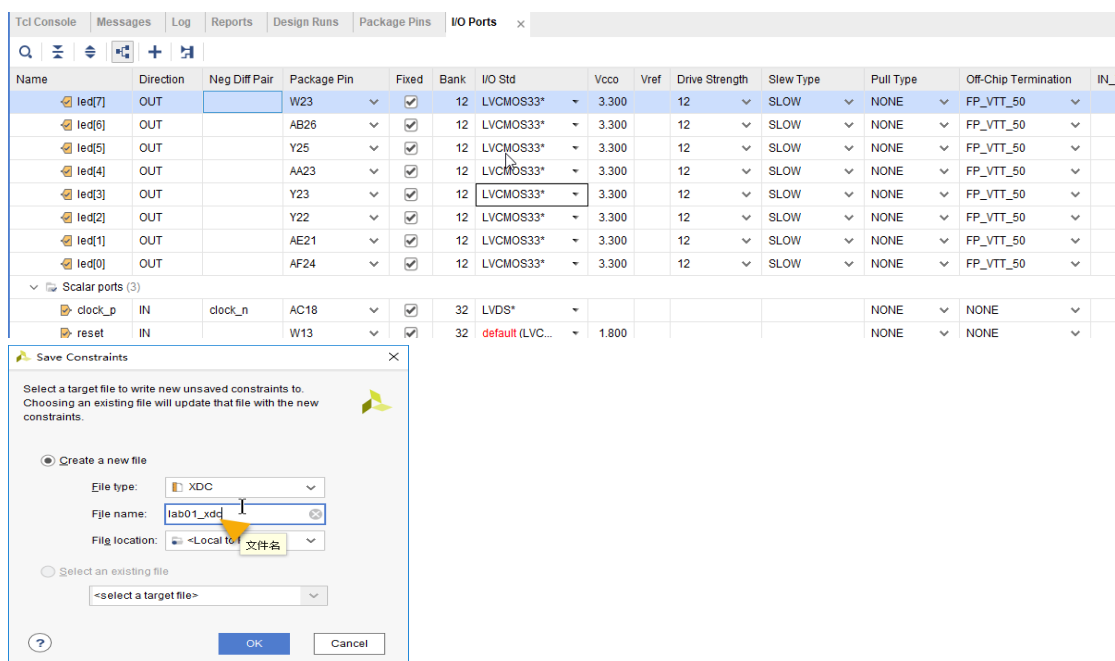
在下方区的选项卡中选中 I/O ports，并在对应的信号后，输入对应的FPGA 管脚标号（或将信号拖拽到右上方 Package 图中对应的管脚上），并指定I/O Std。具体的 FPGA 约束管脚和 IO 电平标准，可参考对应板卡的硬件手册。



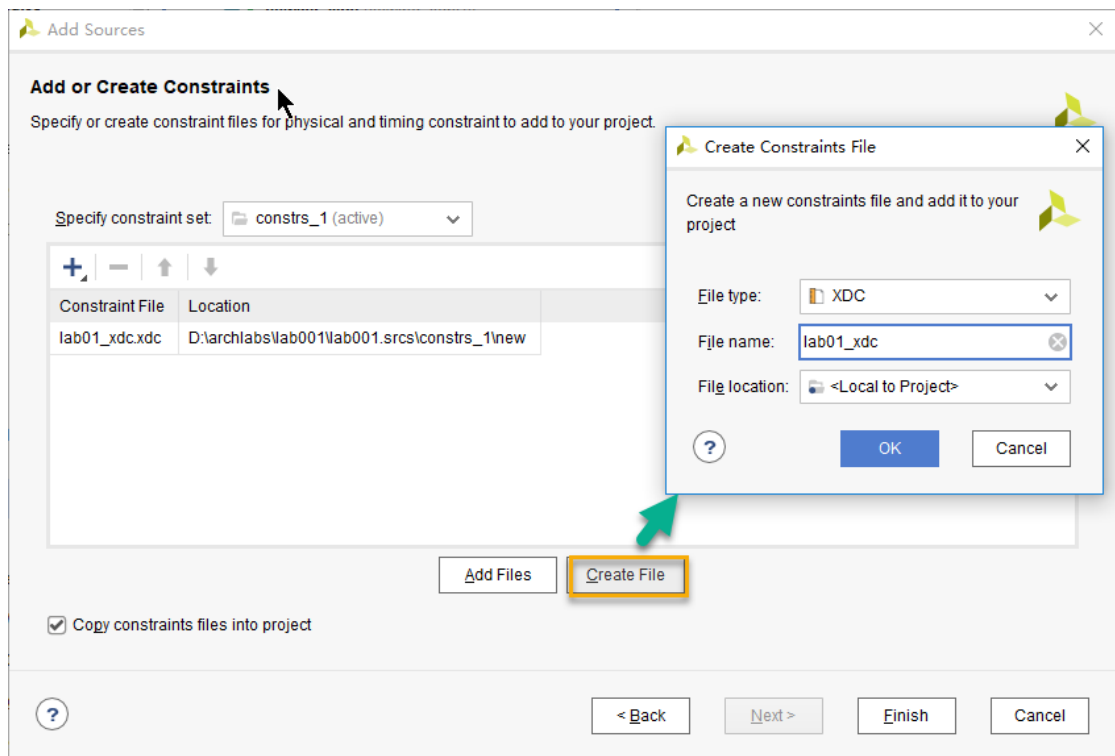
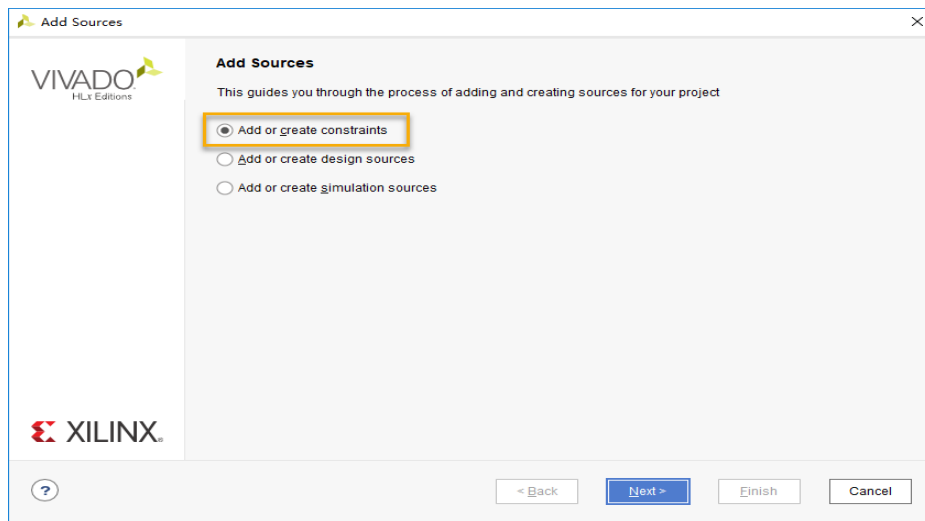
本次实验用到的对应管脚及IO电平标准如下：

| | | |
|----|---------|----------------------------------|
| 2 | led[7] | 对应的管脚是 W23 ; I/O Std为 LVC MOS33 |
| 3 | led[6] | 对应的管脚是 AB26 ; I/O Std为 LVC MOS33 |
| 4 | led[5] | 对应的管脚是 Y25 ; I/O Std为 LVC MOS33 |
| 5 | led[4] | 对应的管脚是 AA23 ; I/O Std为 LVC MOS33 |
| 6 | led[3] | 对应的管脚是 Y23 ; I/O Std为 LVC MOS33 |
| 7 | led[2] | 对应的管脚是 Y22 ; I/O Std为 LVC MOS33 |
| 8 | led[1] | 对应的管脚是 AE21 ; I/O Std为 LVC MOS33 |
| 9 | led[0] | 对应的管脚是 AF24 ; I/O Std为 LVC MOS33 |
| 10 | | |
| 11 | clock_p | 对应的管脚是 AC18 ; I/O Std为 LVDS |
| 12 | reset | 对应的管脚是 W13 ; I/O Std为 LVC MOS18 |

全部管脚锁定及I/O Std指定后，点击左上方工具栏中的保存按钮，弹出对话框提示新建约束文件，可输入lab01_xdc，并点击OK



2) 像创建模块代码文件一样新建约束文件。打开Add Sources对话框, 如图选择第一项



在Sources区双击打开新建好的空的约束文件，并按照规则输入符合的相应的 FPGA 管脚约束信息和电平标准的约束语句：

```
set_property PACKAGE_PIN W23 [get_ports {led[7]}]
set_property PACKAGE_PIN AB26 [get_ports {led[6]}]
set_property PACKAGE_PIN Y25 [get_ports {led[5]}]
set_property PACKAGE_PIN AA23 [get_ports {led[4]}]
set_property PACKAGE_PIN Y23 [get_ports {led[3]}]
set_property PACKAGE_PIN Y22 [get_ports {led[2]}]
```

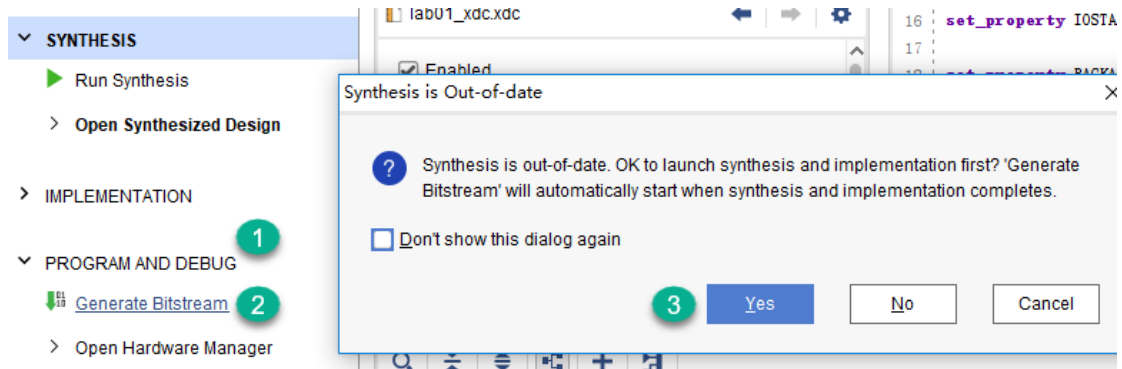
```

set_property PACKAGE_PIN AE21 [get_ports {led[1]]}
set_property PACKAGE_PIN AF24 [get_ports {led[0]]}
set_property PACKAGE_PIN AC18 [get_ports clock_p]
set_property PACKAGE_PIN W13 [get_ports reset]
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 LVDS [get_ports clock_p]
set_property IOSTANDARD LVDS [get_ports clock_n]
set_property IOSTANDARD LVCMOS18 [get_ports reset]

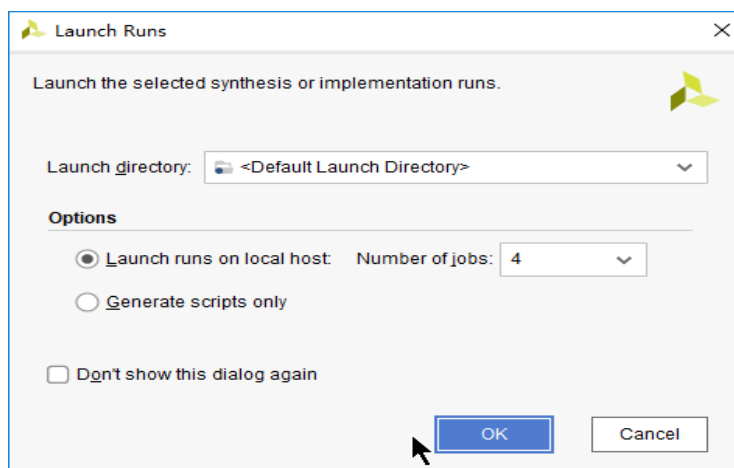
```

2.6 下载验证

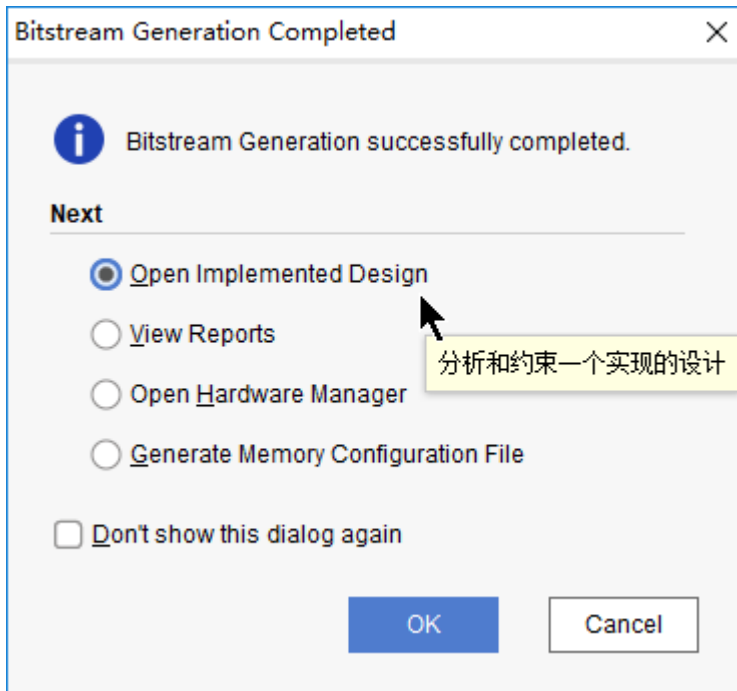
1. 在 Flow Navigator 中点击 Program and Debug 下的 Generate Bitstream 选项，系统会自动完成综合、实现、生成 FPGA 配置文件（bit 文件）。如出现以下框，点 Yes



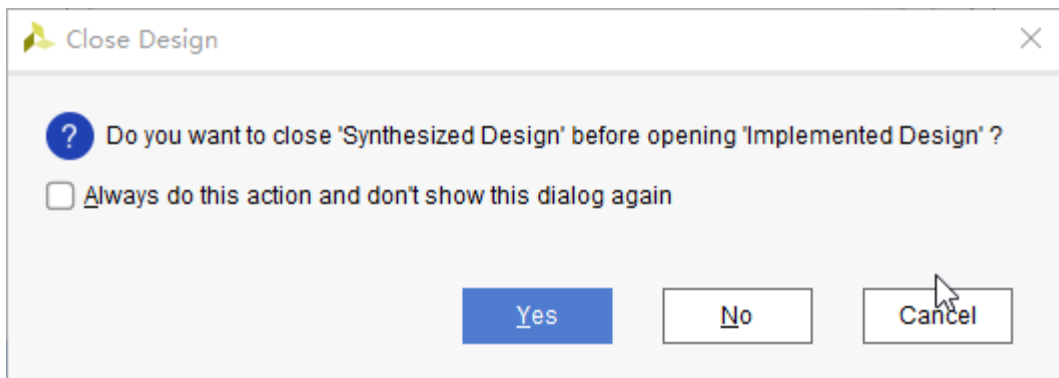
接下来的框，点击OK



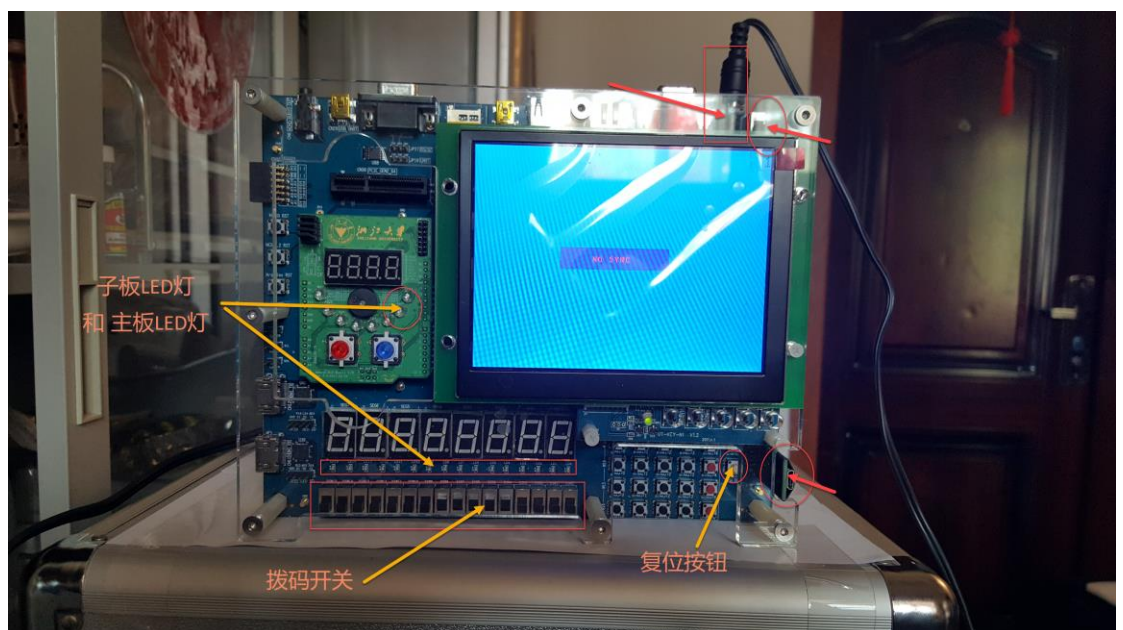
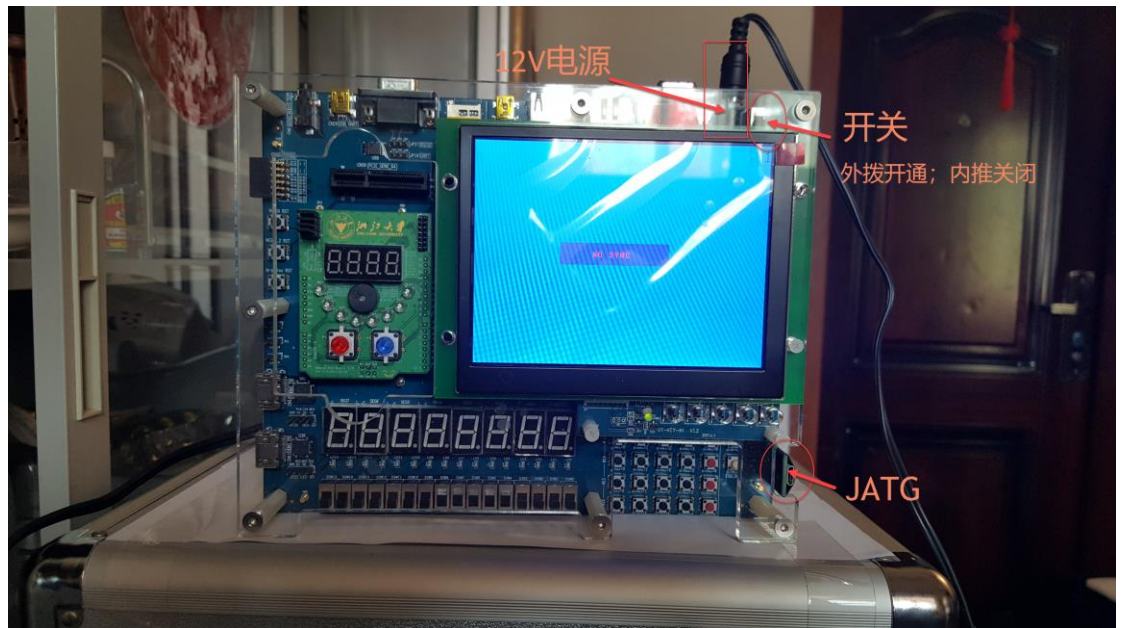
Bitstream生成后，可点击 **Open Implemented Design** 来查看实现的结果，也可点击Cancel退出，也可选第三项直接去烧写



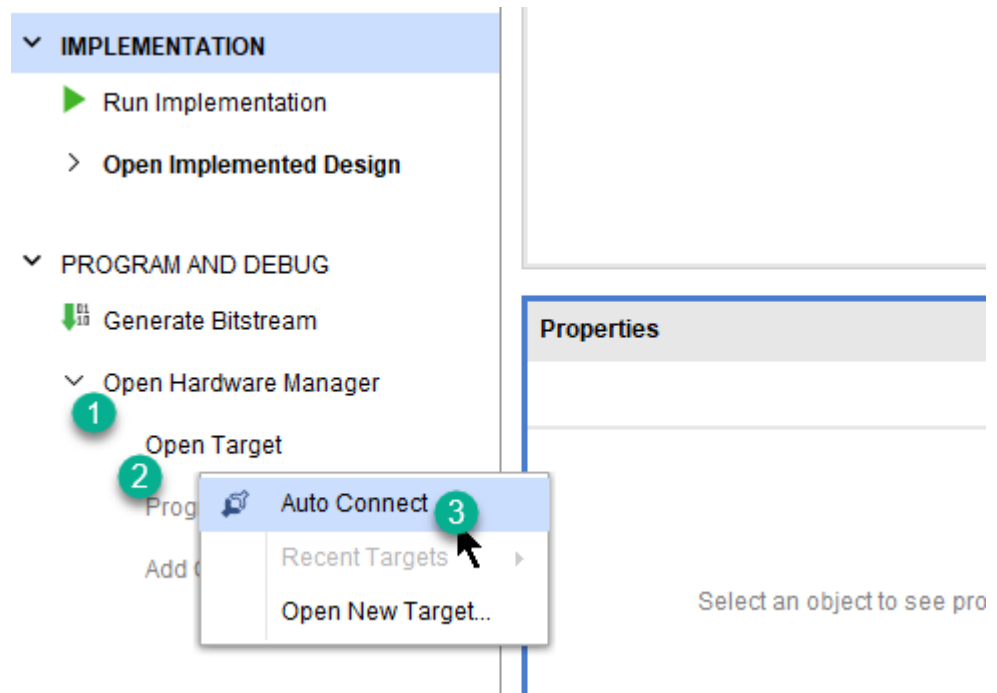
点击Yes 或No



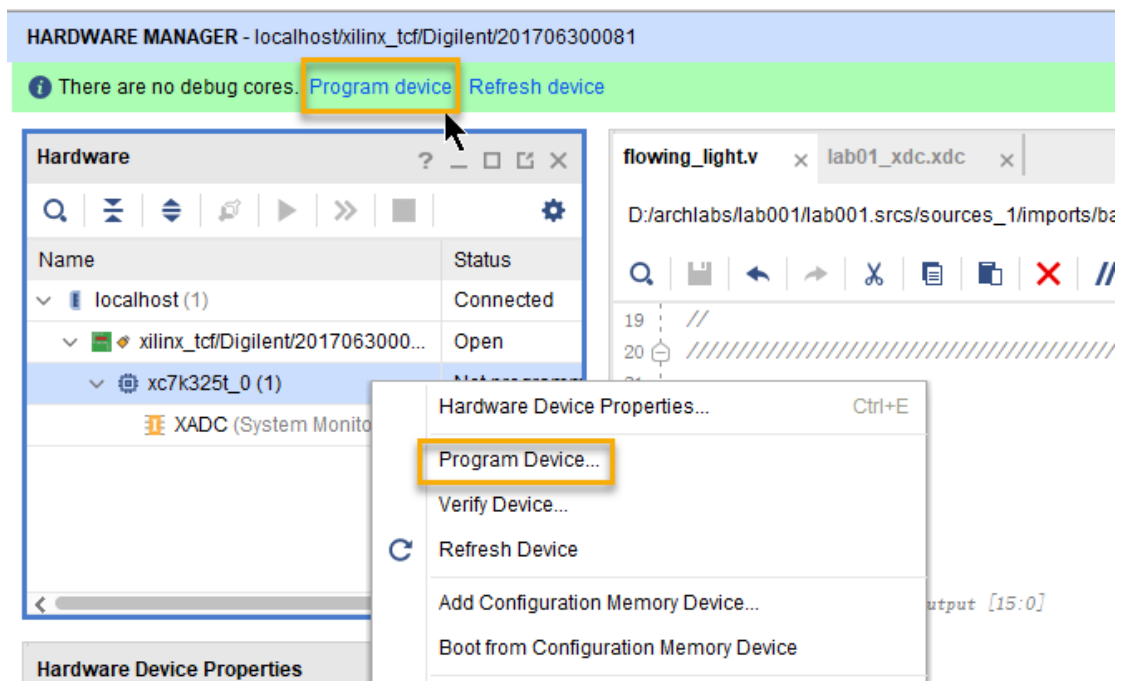
2. 按下图连接 **SWORD4.0** 的 12V 电源和通过JTAG下载器连到计算机，然后打开开关，并令防火墙允许访问。



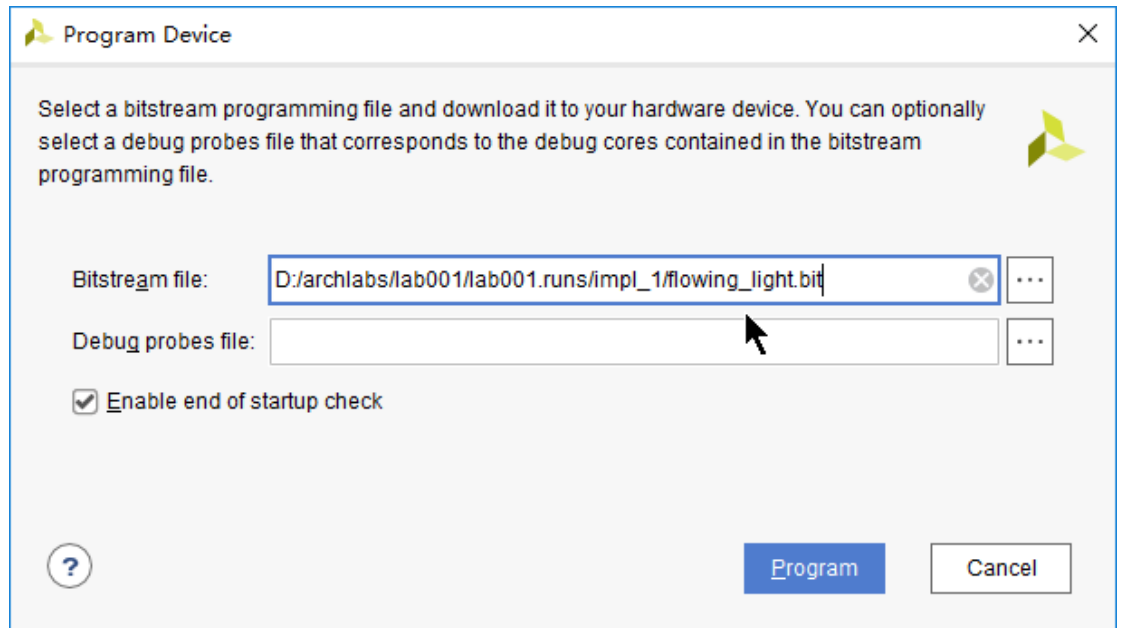
3. 点击 Flow Navigator 中 Open Hardware Manager 一项，进入硬件编程管理界面



4. 连接成功后，在目标芯片上右击，选择 Program Device或者如下图



5. 在弹出的对话框中 会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



6. 在主板的子板上观察实验结果，看8位流水灯显示是否预期。

2.7 主板16位LED流水灯的实现（可选做）

SWORD 平台提供了 4 种 GPIO 接口：4X4 按键矩阵、16 位滑动开关、16 位 LED、8 位 7 段数码管。其中为了节省 I/O，仅 16 位的滑动开关采用了和 FPGA 直连的方式，16 位 LED 和 8 位 7 段数码管采用了 SN74LV164 移位寄存器进行串行转并行的处理。

在原工程中的 flowing_light.v 和 lab01_xdc.xdc 文件加入串行转并行机制（parallel2serial 软核）并进行相应添加代码行就可实现。

1) 模块文件修改如下：

```
module flowing_light(
    input clock_p,
    input clock_n,
    input reset,
    output [7:0] led,
    output LED_CLK, //为实现主板16位led流水而新增
    output LED_D0,  //新增
    output LED_CLR  //新增
);

reg [23:0] cnt_reg;
reg [7:0] light_reg;
reg [15:0] light_reg2;//新增

IBUFGDS IBUFGDS_inst (
    .O(CLK_i),
    .I(clock_p),
    .IB(clock_n)
```



```

    );

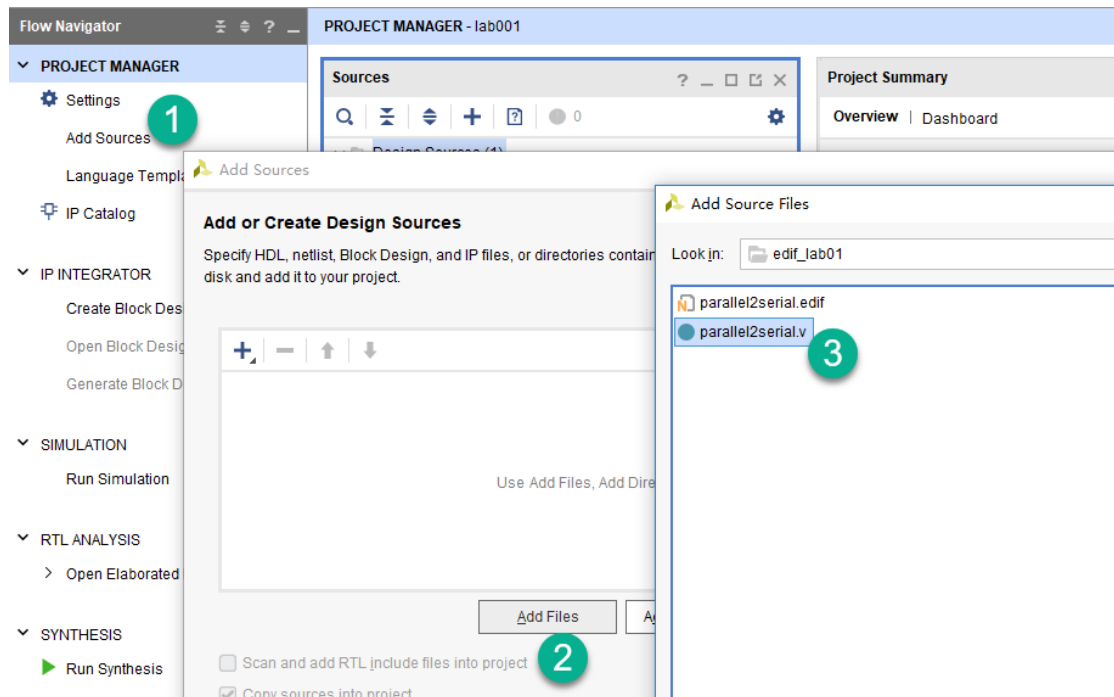
always @ (posedge CLK_i)
begin
    if (!reset) //板上复位按钮按下是底电平
        cnt_reg <= 0;
    else
        cnt_reg <= cnt_reg + 1;
end
always @ (posedge CLK_i )
begin
    if (!reset)
    begin//新增
        light_reg <= 8'h01;
        light_reg2<=16'b01;//新增
    end //新增
    else if (cnt_reg == 24'hffffff)
    begin
        light_reg2<={light_reg2[14:0],light_reg2[15]};//新增
        if (light_reg == 8'h80)
            light_reg <= 8'h01;
        else
            light_reg <= light_reg << 1;
        end
    end
end
assign led  = light_reg;

//以下所有语句为实现主板16位led流水而新增
wire led_clr;
assign LED_CLR=~led_clr;
reg [23:0] clkcnt;
always@(posedge CLK_i)
    clkcnt<=clkcnt+1;
parallel2serial #(
    .P_CLK_FREQ(200),
    .S_CLK_FREQ(20),
    .DATA_BITS(16),
    .CODE_ENDIAN(1))
P2S_LED (
    .clk(CLK_i),
    .rst(~reset),
    .data(light_reg2),
    .start((clkcnt==24'b0)?1'b1:1'b0),
    .busy(),
    .finish(),
    .s_clk(LED_CLK),
    .s_clr(led_clr),
    .s_dat(LED_DO));

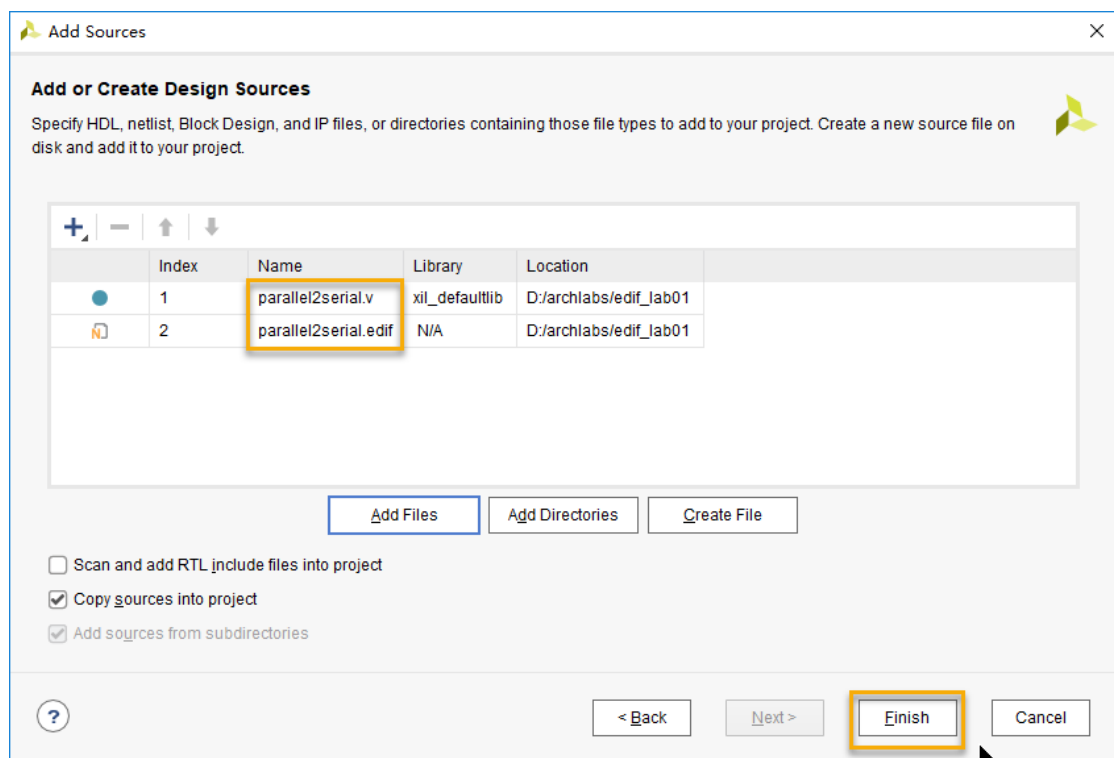
```

Endmodule

2) flowing_light.v调用了parallel2serial模块，需添加该模块（已封装成网表文件共两个文件）到工程中



两个文件添加后，点击Finish



3) 管脚约束文件添加如下

#16leds 以下为16led显示流水新增

```
set_property PACKAGE_PIN N26 [get_ports LED_CLK]
```

```
set_property PACKAGE_PIN N24 [get_ports LED_CLR]
```

```
set_property PACKAGE_PIN M26 [get_ports LED_D0]  
set_property IOSTANDARD LVCMOS33 [get_ports LED_CLK]  
set_property IOSTANDARD LVCMOS33 [get_ports LED_CLR]  
set_property IOSTANDARD LVCMOS33 [get_ports LED_D0]
```

4) 生成bit文件、下载验证，结果是否符合预期。