计算机系统结构实验指导书-LAB02

1. OVERVIEW

1.1 实验名称

FPGA 基础实验：4-bit Adder

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 VerilogHDL 进行简单的逻辑设计

3．使用功能仿真

4．约束文件的使用和直接写法

4．添加时序约束

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

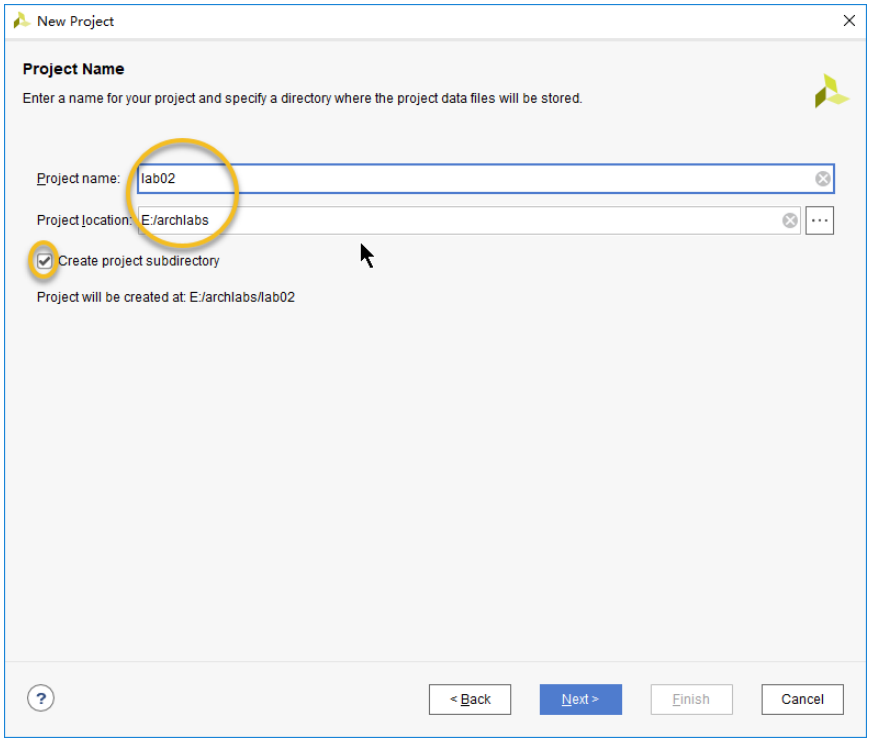
1. 启动桌面 Vivado 2018.3 开发工具

2. 点击 Create Project

3. 弹出 New Project，建立一个新工程，点击 Next

4. 输入工程名称 **lab02**，工程位置建议 E:/archlabs，确认勾选中 **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



5. 选择 **RTL Project** 工程类型，勾选 **Do not specify sources at this time**，点击 **Next**

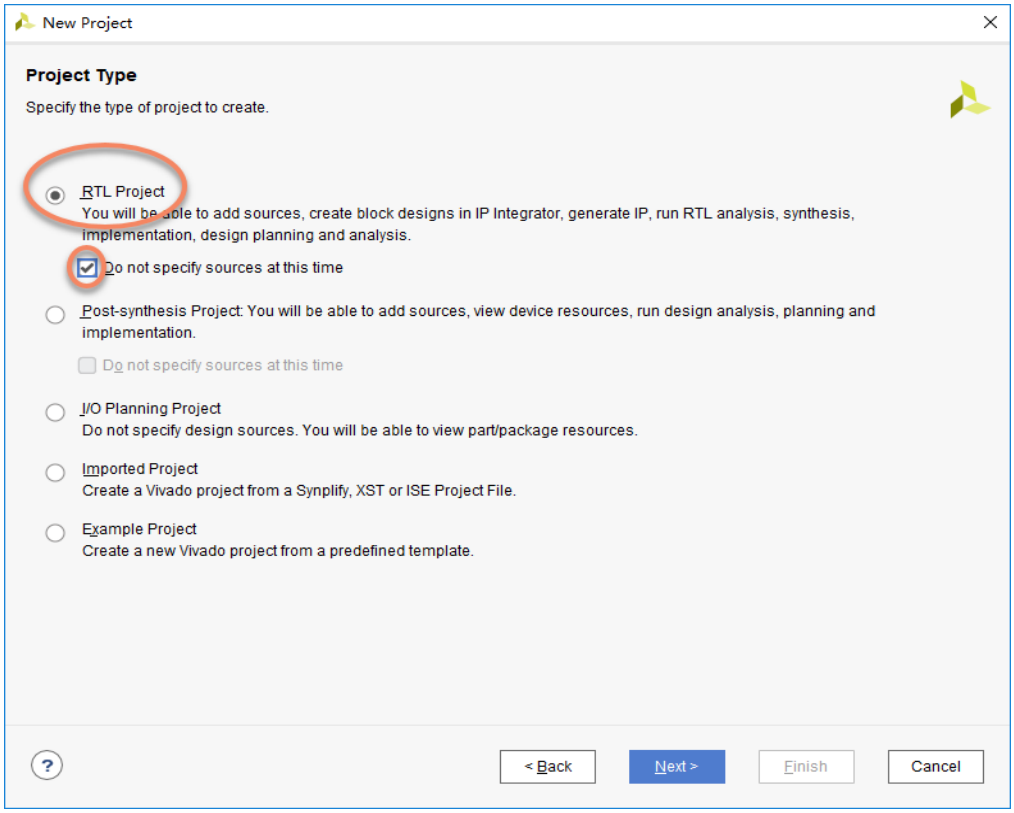
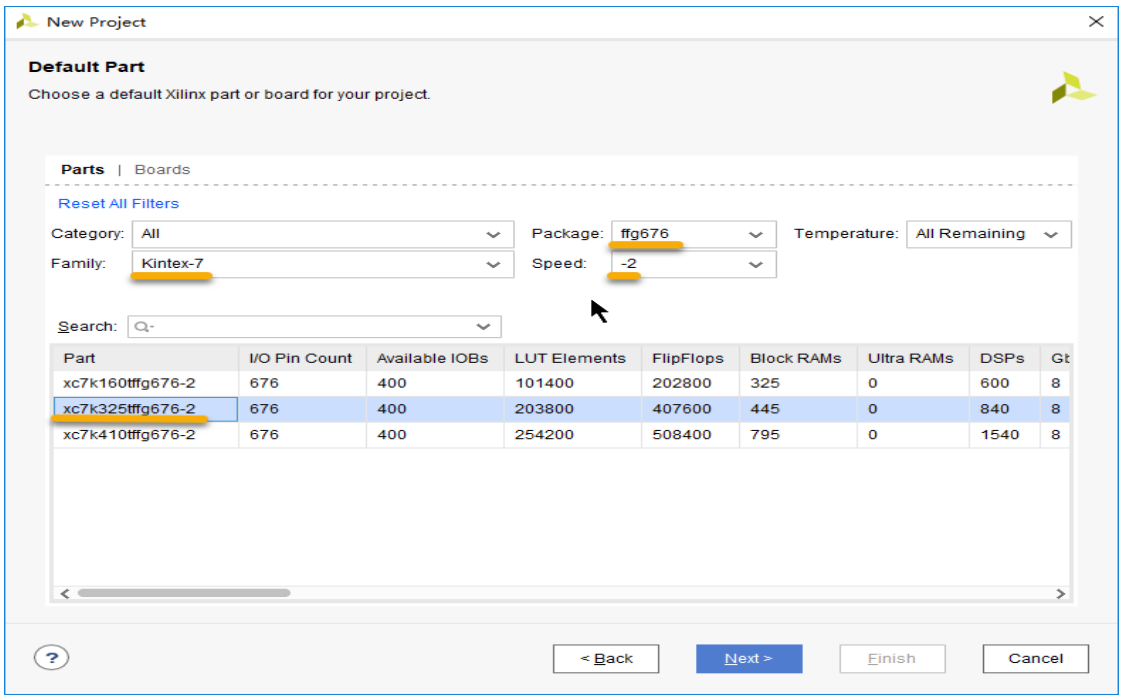


图 2-1　RTL 工程

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



7. 点击 **Finish** 结束工程的创建。

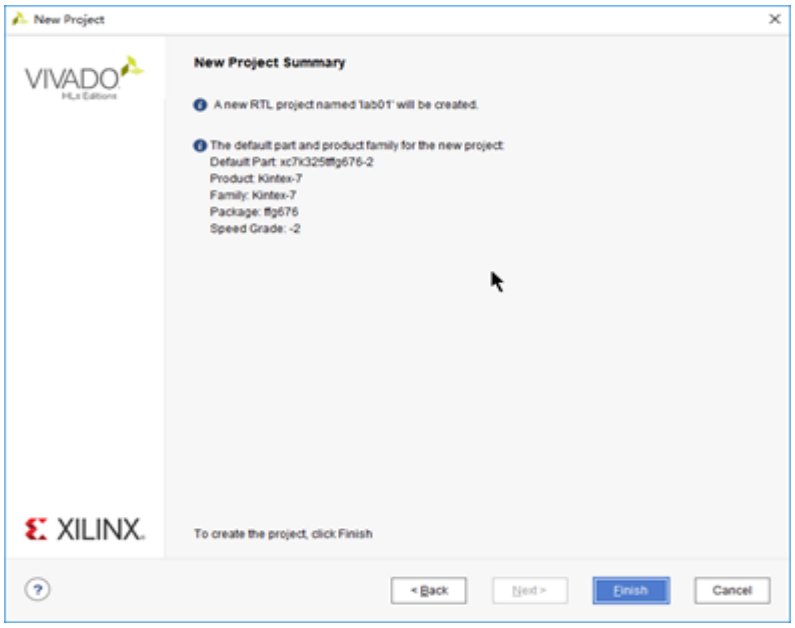


图 2-2　新建工程完毕

2.2 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

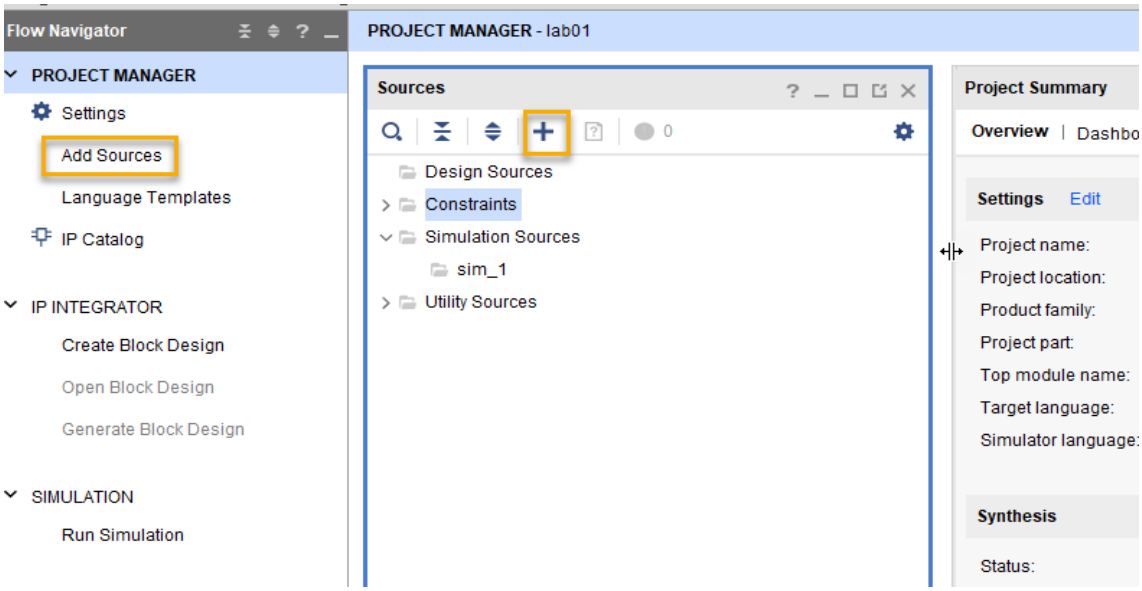
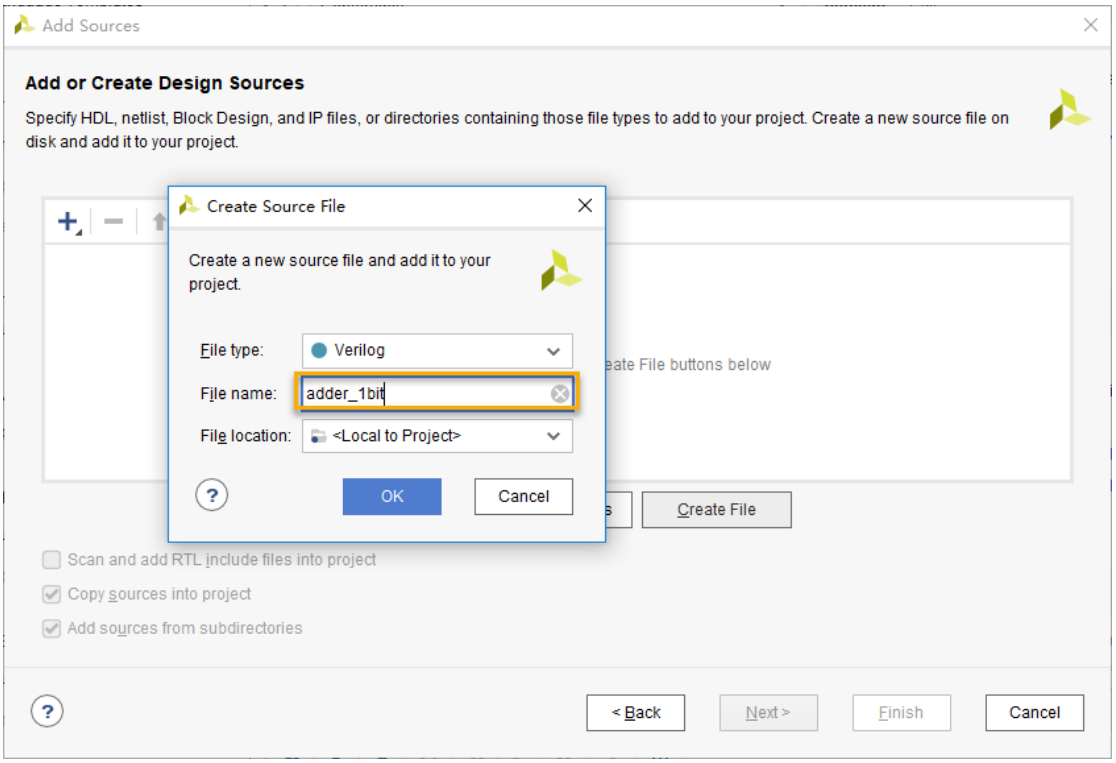


图 2-3　增加文件

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

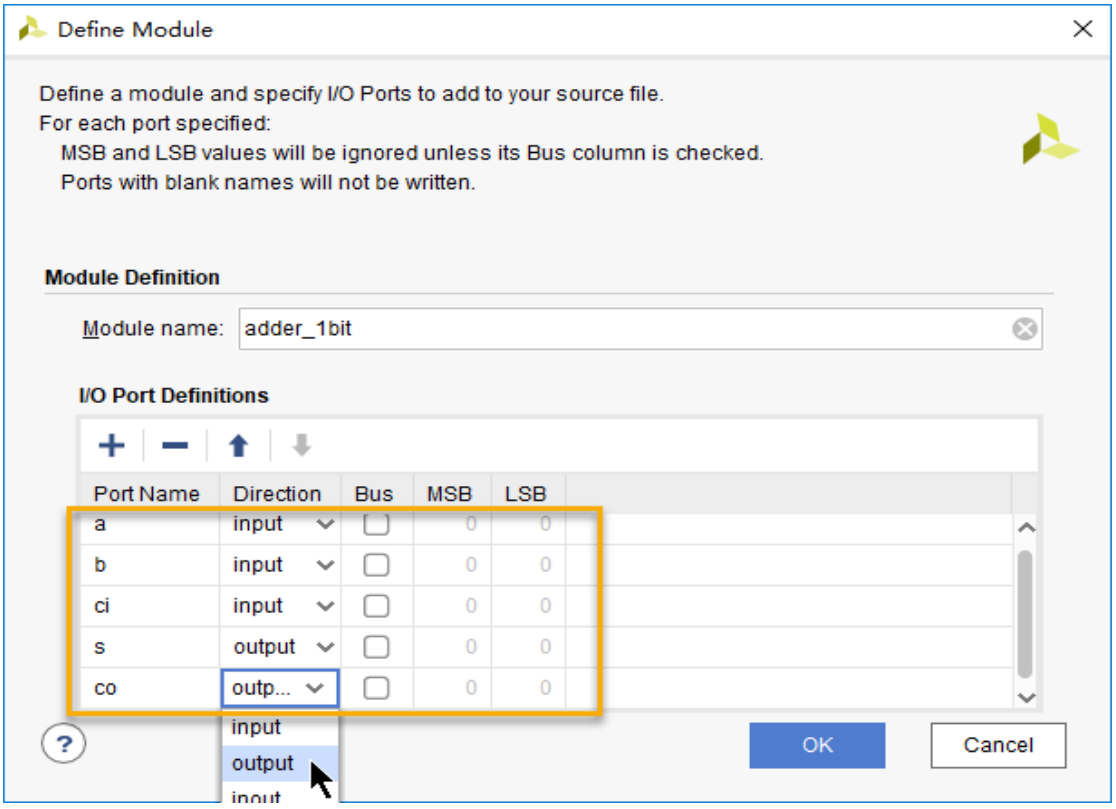
3. 若已有源文件或内核文件，可选 Add Files 项以添加文件。这里是要新建 1 位全加器模块文件，选择 Create File

4. 弹框 Create Source File 中输入 adder_1bit 文件名，点击 OK



5. 点击 Finish

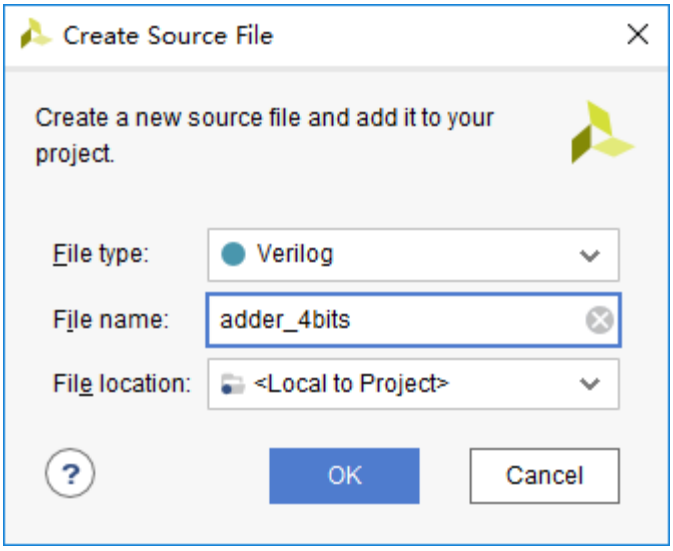
6. 在弹出的 Define Module 中输入设计模块所需的端口，并设置端口方向；完成后点击 OK



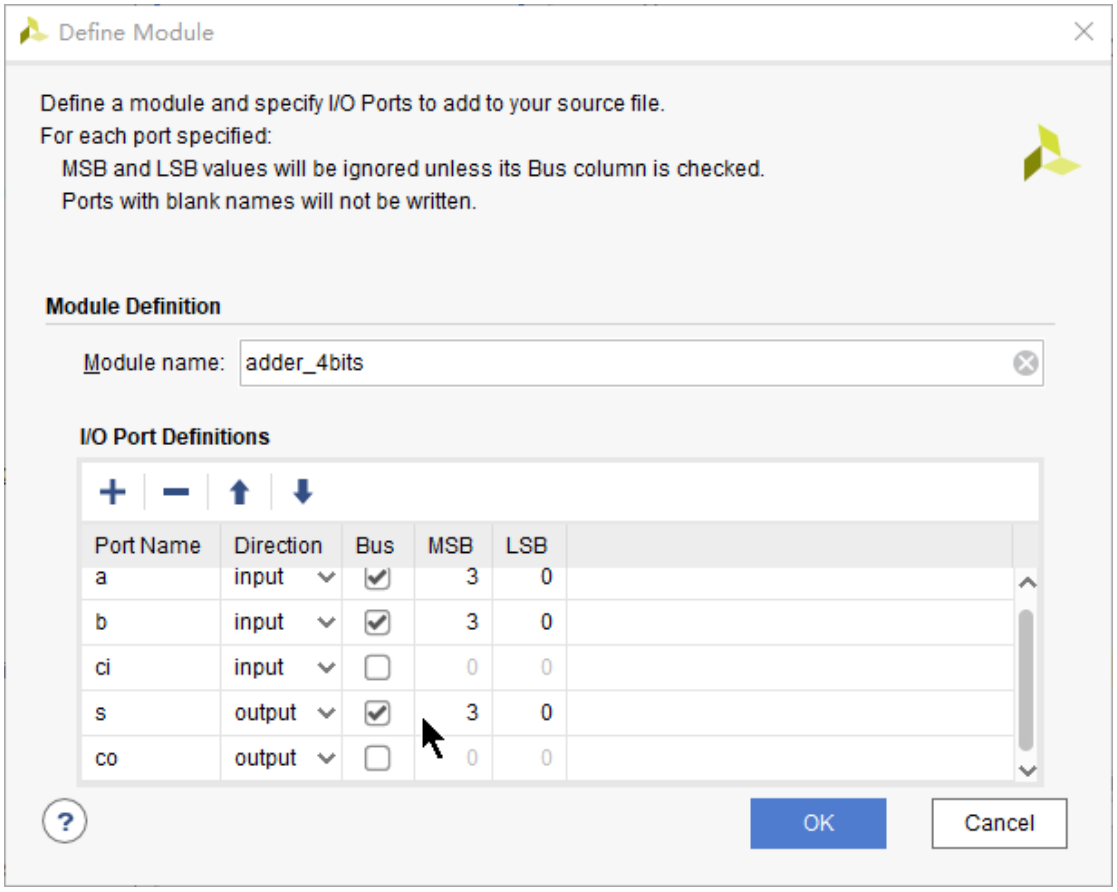
7. 在源代码编辑区双击 adder_1bit，添加如下方框中的代码

```
module adder_1bit(
    input a,
    input b,
    input ci,
    output s,
    output co
    );
    wire s1, c1, c2, c3;
    and  (c1, a, b),
         (c2, b, ci),
         (c3, a, ci);

    xor  (s1, a, b),
         (s, s1, ci);

    or   (co, c1, c2, c3);

endmodule
```

8. 接下来要实现 4 位加法器。继续创建源文件，命名为 adder_4bits，点击 OK



9. 点击 Finish，输入模块所需的端口，并设置端口方向，若端口属总线型，勾选 Bus，并由 MSB 和 LSB 确定宽度（当然这一步也可忽略模块的端口定义和宽度确定，直接点击 OK 之后点击 Yes，再点击 Finish）点击 OK

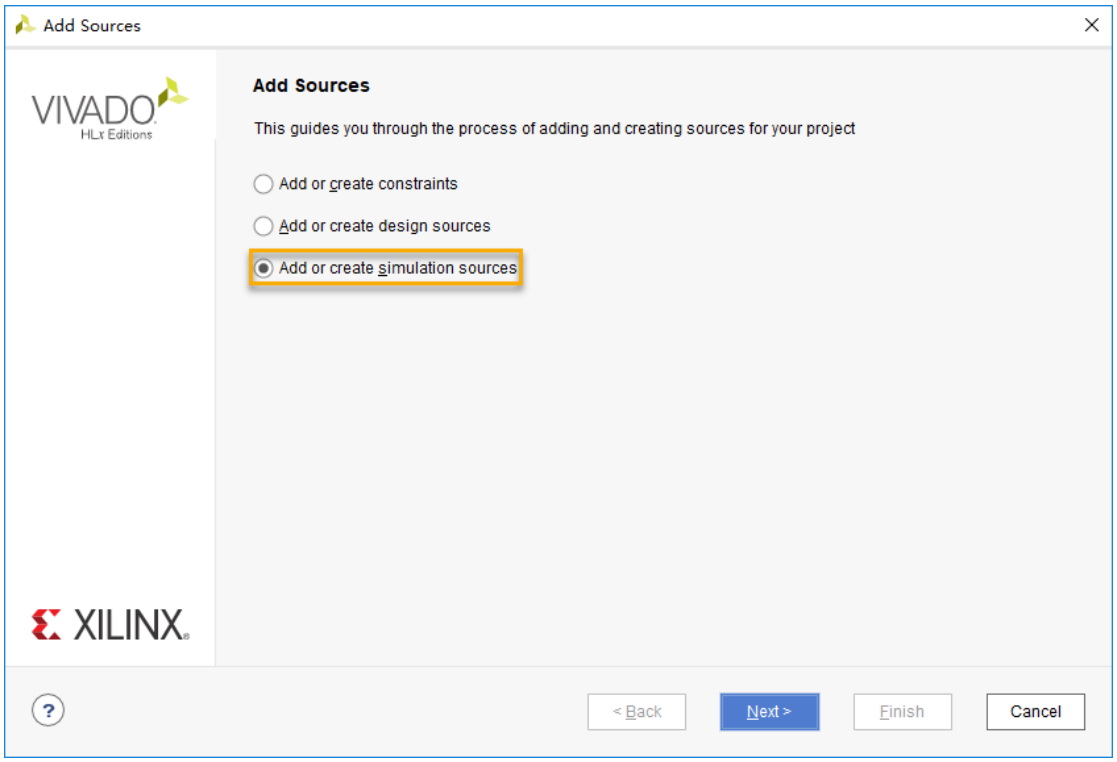


10. 双击 adder_4bits，添加如下代码

```
module adder_4bits(
        input [3:0] a,
        input [3:0] b,
        input ci,
        output [3: 0] s,
        output co
        );

        wire [2:0] ct;

        adder_1bit a1(.a(a[0]), .b(b[0]), .ci(ci), .s(s[0]),.co(ct[0])),
                   a2(.a(a[1]), .b(b[1]), .ci(ct[0]), .s(s[1]),.co(ct[1])),
                   a3(.a(a[2]), .b(b[2]), .ci(ct[1]), .s(s[2]),.co(ct[2])),
                   a4(.a(a[3]), .b(b[3]), .ci(ct[2]), .s(s[3]),.co(co));

endmodule
```

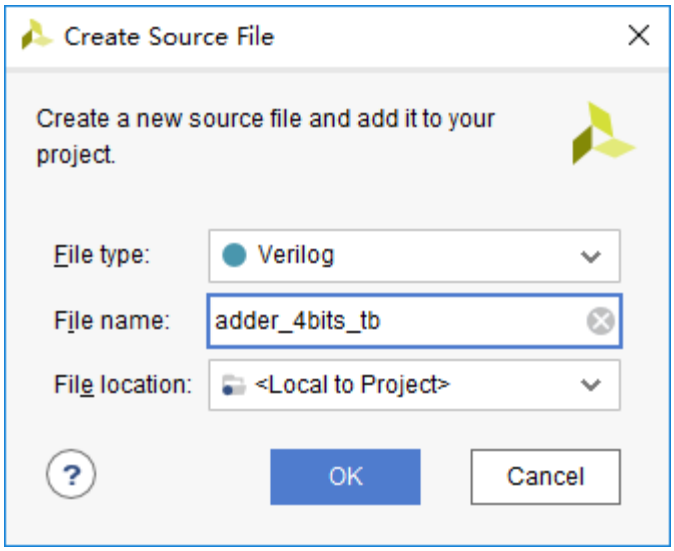
2.3 功能仿真

1. 创建激励测试文件。在中间区 Sources 具栏点击“+”号或于左侧区 PROJEU MANAGER 下选择 Add Source
2. 在 Add Sources 选择 Add or Create Simulation Source，点击 Next



3. 选择 Create File 创建一个仿真激励文件

4. 激励文件测试名可输入 adder_4bits_tb，点击 OK



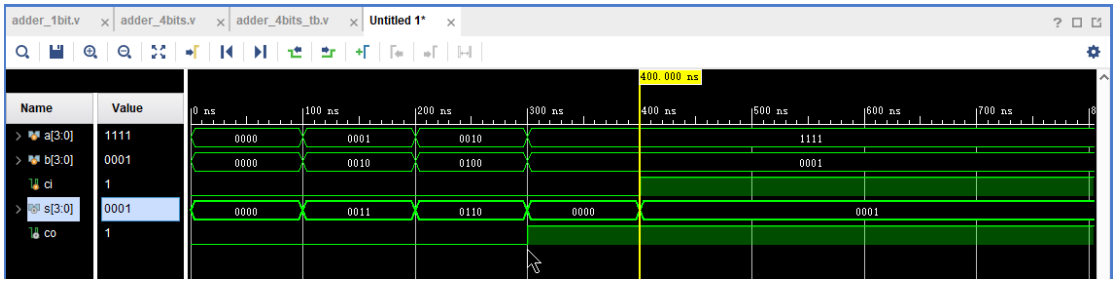
5. 点击 Finish，创建激励测试文件不需要对外端口，点击 OK

6. 在弹出的对话框点击 YES

7. 在 Source 区 Simulation Sources 下，打开测试文件 adder_4bit_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

```
module adder_4bits_tb(    );

    reg [3:0] a;
    reg [3:0] b;
    reg ci;

    wire [3:0] s;
    wire co;

    adder_4bits u0 (
        .a(a),
        .b(b),
        .ci(ci),
        .s(s),
        .co(co)
        );

    initial begin
        a = 0;
        b = 0;
        ci = 0;

        #100;
        a = 4'b0001;
        b = 4'b0010;
        #100;
        a = 4'b0010;
        b = 4'b0100;

        #100;
        a = 4'b1111;
        b = 4'b0001;
        #100;
        ci = 1'b1;

    end

endmodule
```

8. 点击左侧区的 Run Simulation 并选择 Run Behavioral Simulation。下图为仿真运行后得到的仿真波形图



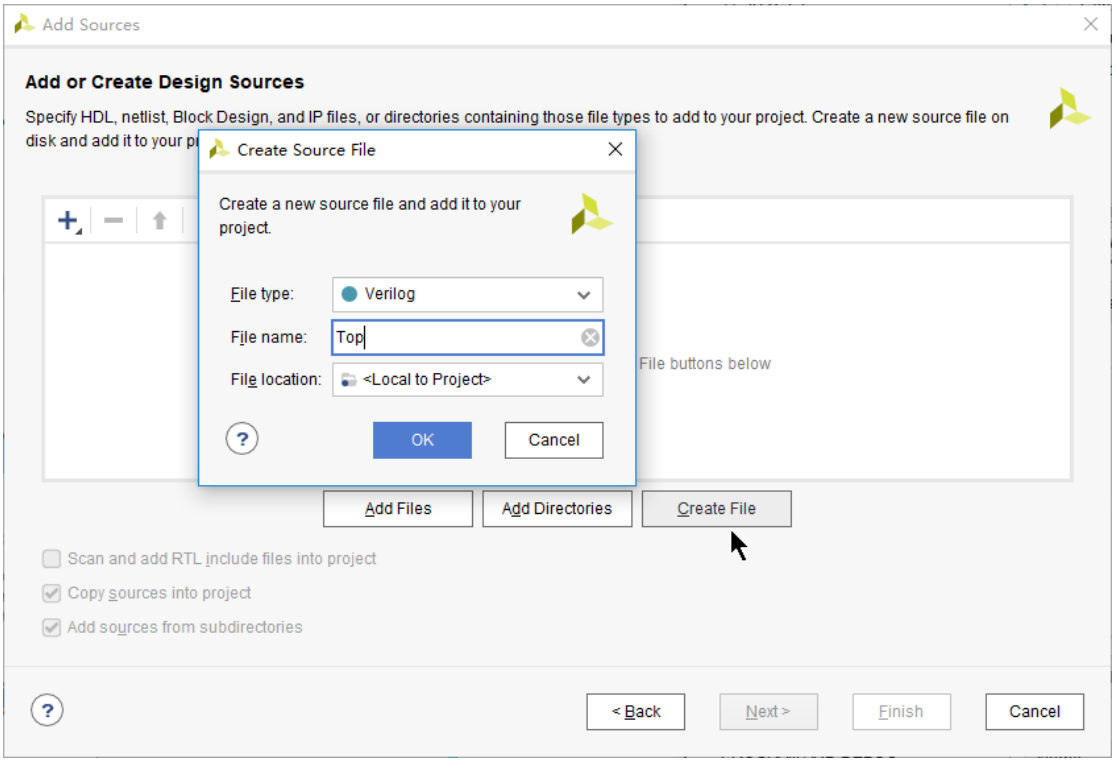
观察可知，仿真结果与逻辑功能一致，电路能正常工作。如果不一致可以放大或缩小、可以滑动黄色的时间运行竖线来查看输入与输出间的变化是否吻合，以便修改逻辑代码

2.5 工程实现

在本实验中要用 Sword 平台的 8 个 Switch 对应二组 4 位二进制输入，用 4 个 LED 发光二极管对应输出并用 2 个七段数码管显示运行结果。故本实验需要用到 display.v 这个七段数码管 SEGMENT 和 LED 显示模块（实验室提供该核，以网表文件形式给出）

1. 将4位加法器的输出赋予LED和七段数码管显示，需要创建一个顶层源文

件，可命名Top，如下图



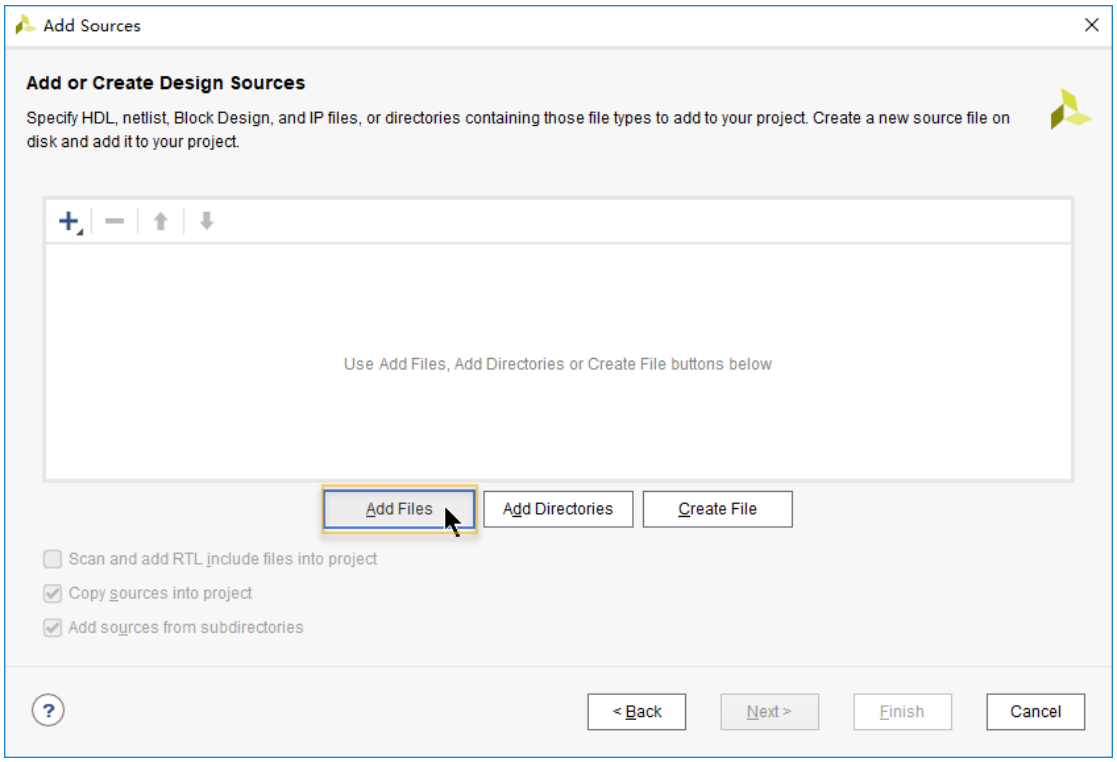
进入Define Module，这里我们略过端口定义，将在源程序中自行添加；
双击Top模块，添加以下端口变量和代码语句：

```
module Top(
    input clk_p,
    input clk_n,
    input [3:0] a,
    input [3:0] b,
    input reset,

    output led_clk,
    output led_do,
    output led_en,

   output wire seg_clk,
   output wire seg_en,
   output wire seg_do
);
```

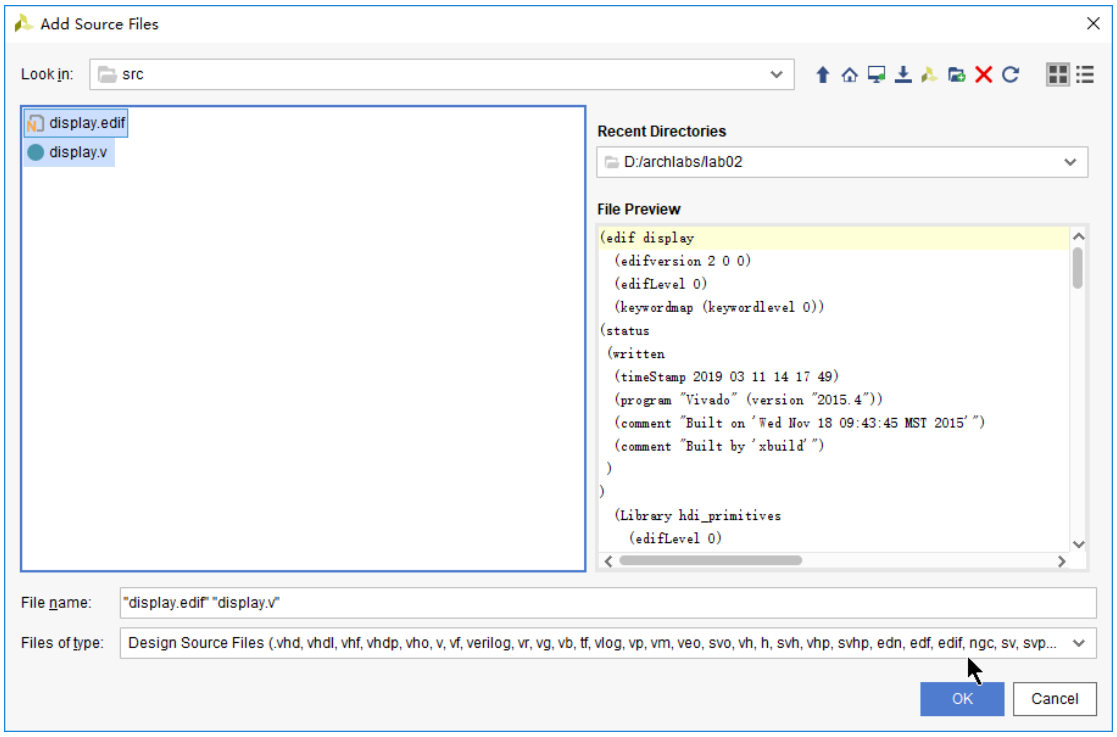
```
wire CLK_i;
wire Clk_25M;

IBUFGDS IBUFGDS_inst (
      .O(CLK_i),
      .I(clk_p),
      .IB(clk_n)
);

wire [3:0] s;
wire co;
wire [4:0] sum;
assign sum = {co, s};

adder_4bits U1 (
.a(a),
.b(b),
.ci(1'b0),
.s(s),
.co(co)
);

reg [1:0] clkdiv;
always@(posedge CLK_i)
    clkdiv<=clkdiv+1;
assign Clk_25M=clkdiv[1];

display DISPLAY (
   .clk(Clk_25M),
   .rst(1'b0),
   .en(8'b00000011),
   .data({27'b0, sum}),
   .dot(8'b00000000),
   .led(~{11'b0, sum}),
   .led_clk(led_clk),
   .led_en(led_en),
   .led_do(led_do),
   .seg_clk(seg_clk),
   .seg_en(seg_en),
   .seg_do(seg_do)
);
```

Endmodule

2. Top.v中用到了一个display核（已转换为网表），接下来添加该核。在Add Sources中，如下图点击

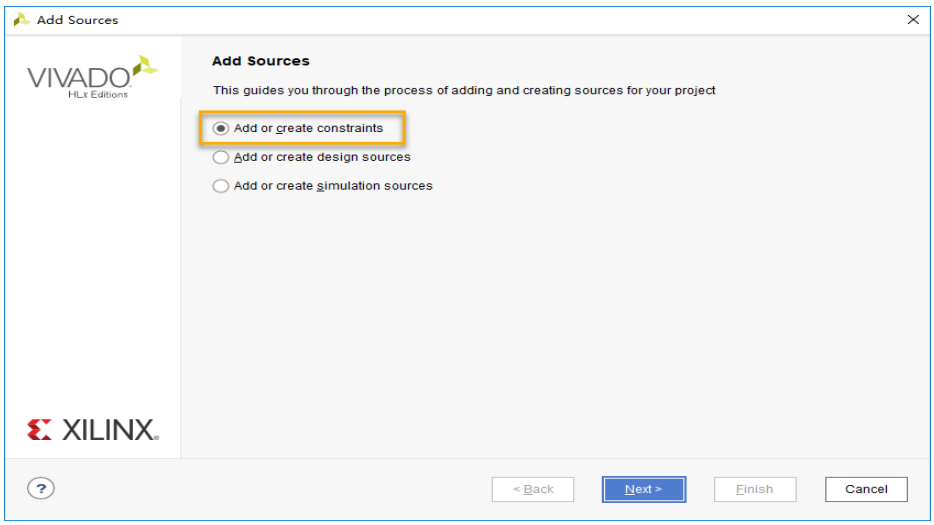


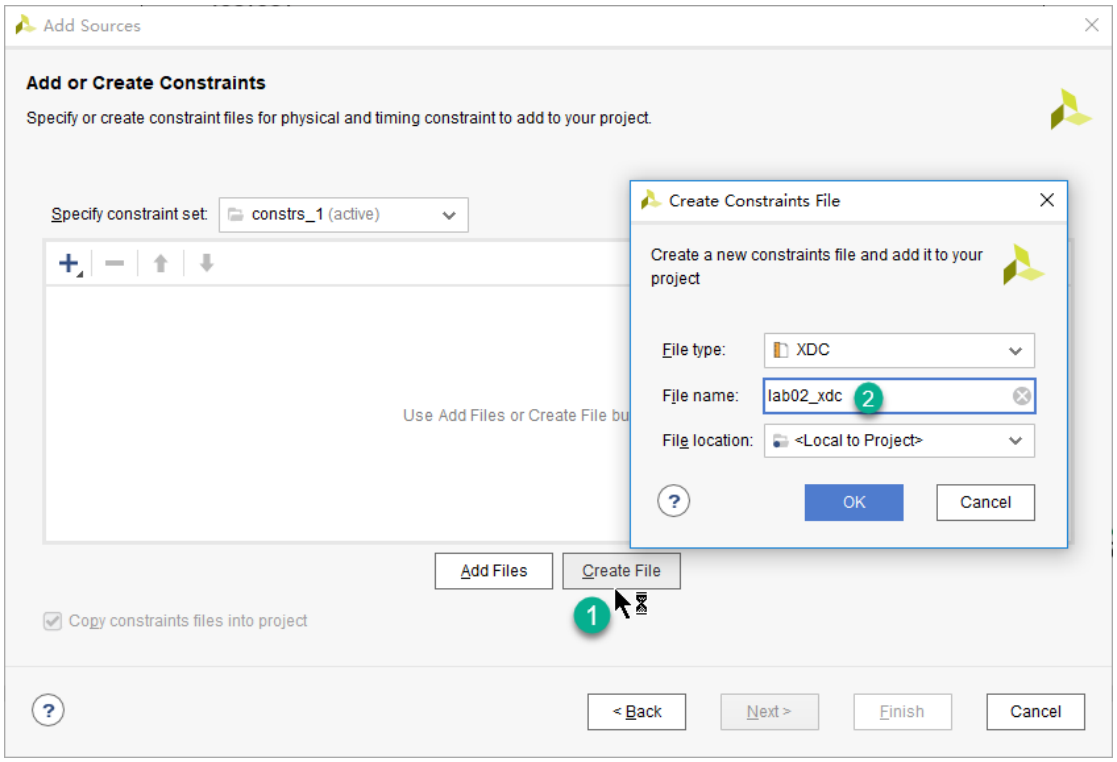
如下图，在Add Source Files中同时选择该核一对同前缀名的端口和网表文件



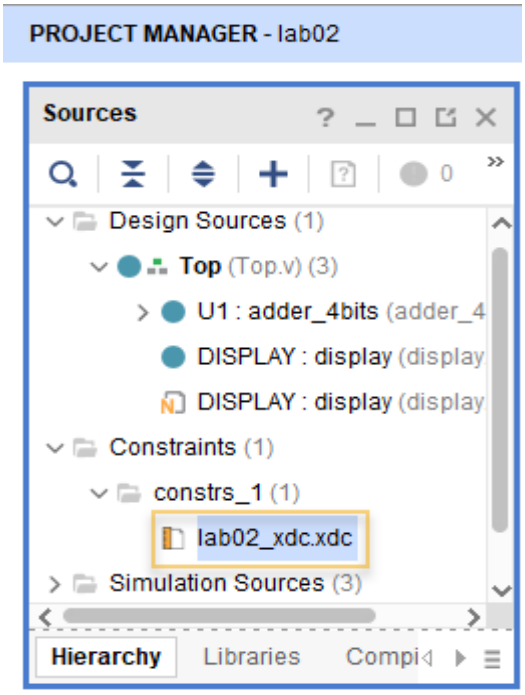
点击Finish，则在Sources区的顶层源文件下关联了这一对源文件

3. 添加约束文件。打开Add Sources对话框，如图选择第一项





如下图位置，打开这新建的空白的约束文件，



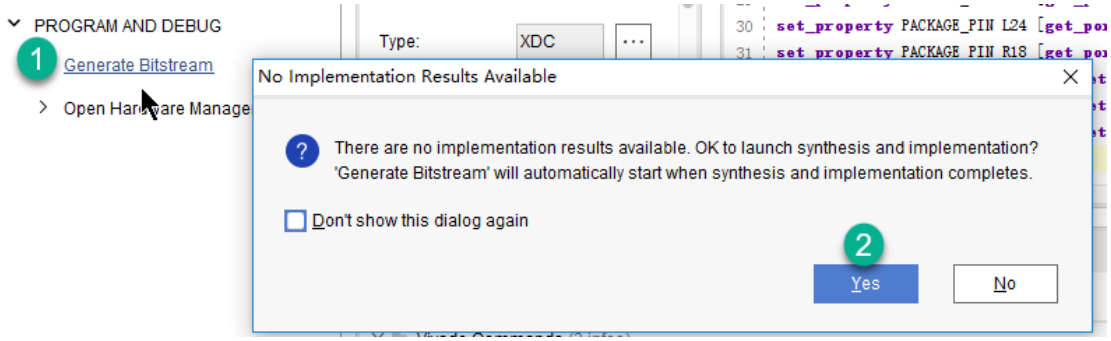
本实验添加如下约束代码：

```
set_property PACKAGE_PIN AC18 [get_ports clk_p]
set_property IOSTANDARD LVDS [get_ports clk_p]

set_property PACKAGE_PIN AA12 [get_ports {a[3]}]
set_property PACKAGE_PIN AA13 [get_ports {a[2]}]
set_property PACKAGE_PIN AB10 [get_ports {a[1]}]
set_property PACKAGE_PIN AA10 [get_ports {a[0]}]
set_property IOSTANDARD LVCMOS15 [get_ports {a[0]}]
set_property IOSTANDARD LVCMOS15 [get_ports {a[1]}]
set_property IOSTANDARD LVCMOS15 [get_ports {a[2]}]
set_property IOSTANDARD LVCMOS15 [get_ports {a[3]}]

set_property PACKAGE_PIN AD10 [get_ports {b[3]}]
set_property PACKAGE_PIN AD11 [get_ports {b[2]}]
set_property PACKAGE_PIN Y12 [get_ports {b[1]}]
set_property PACKAGE_PIN Y13 [get_ports {b[0]}]
set_property IOSTANDARD LVCMOS15 [get_ports {b[0]}]
set_property IOSTANDARD LVCMOS15 [get_ports {b[1]}]
set_property IOSTANDARD LVCMOS15 [get_ports {b[2]}]
set_property IOSTANDARD LVCMOS15 [get_ports {b[3]}]

set_property PACKAGE_PIN N26 [get_ports led_clk]
set_property PACKAGE_PIN M26 [get_ports led_do]
set_property PACKAGE_PIN P18 [get_ports led_en]
set_property IOSTANDARD LVCMOS33 [get_ports led_clk]
set_property IOSTANDARD LVCMOS33 [get_ports led_do]
set_property IOSTANDARD LVCMOS33 [get_ports led_en]
```

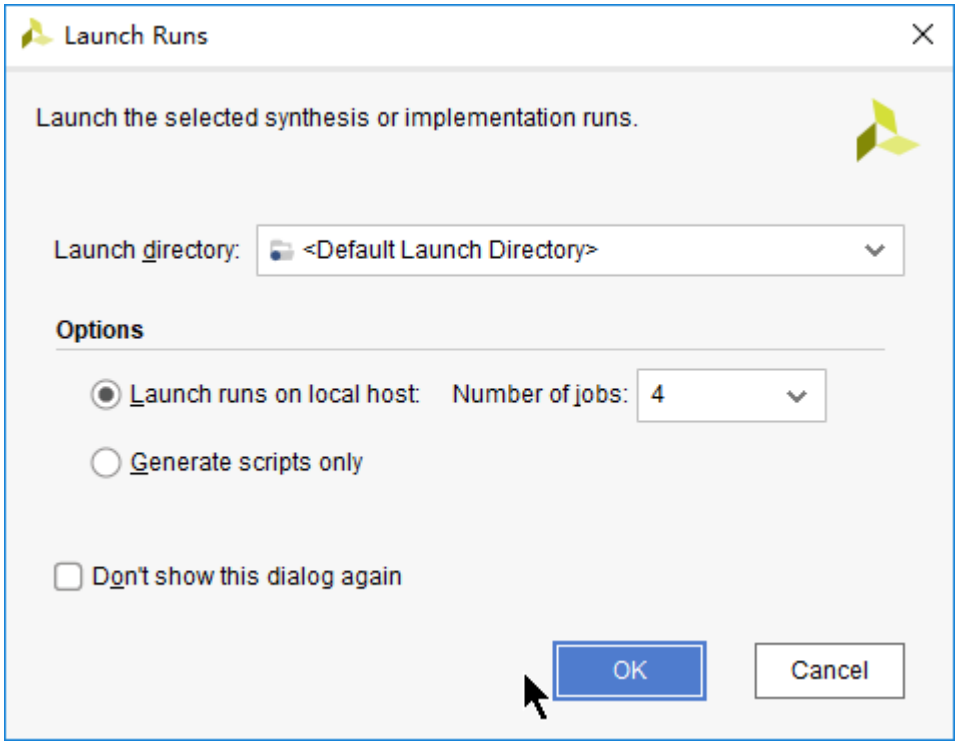
```
set_property PACKAGE_PIN M24 [get_ports seg_clk]

set_property PACKAGE_PIN L24 [get_ports seg_do]
set_property PACKAGE_PIN R18 [get_ports seg_en]
set_property IOSTANDARD LVCMOS33 [get_ports seg_clk]
set_property IOSTANDARD LVCMOS33 [get_ports seg_do]

set_property IOSTANDARD LVCMOS33 [get_ports seg_en]
```

2.6 下载验证

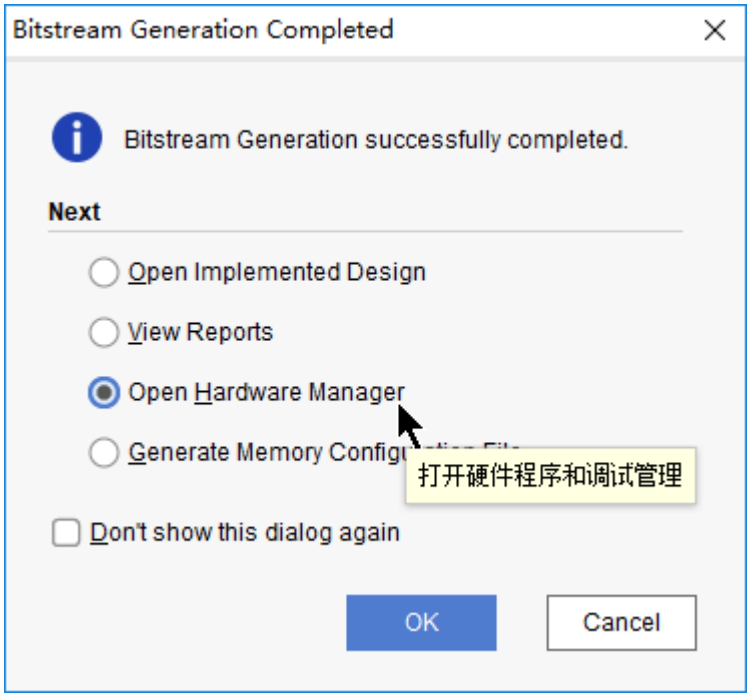
1. 在 Flow Navigator区点击 Program and Debug下Generate Bitstream选项，系统将自动完成综合、实现、并生成 FPGA 配置文件。出现以下框，点Yes



接下来的框，点OK

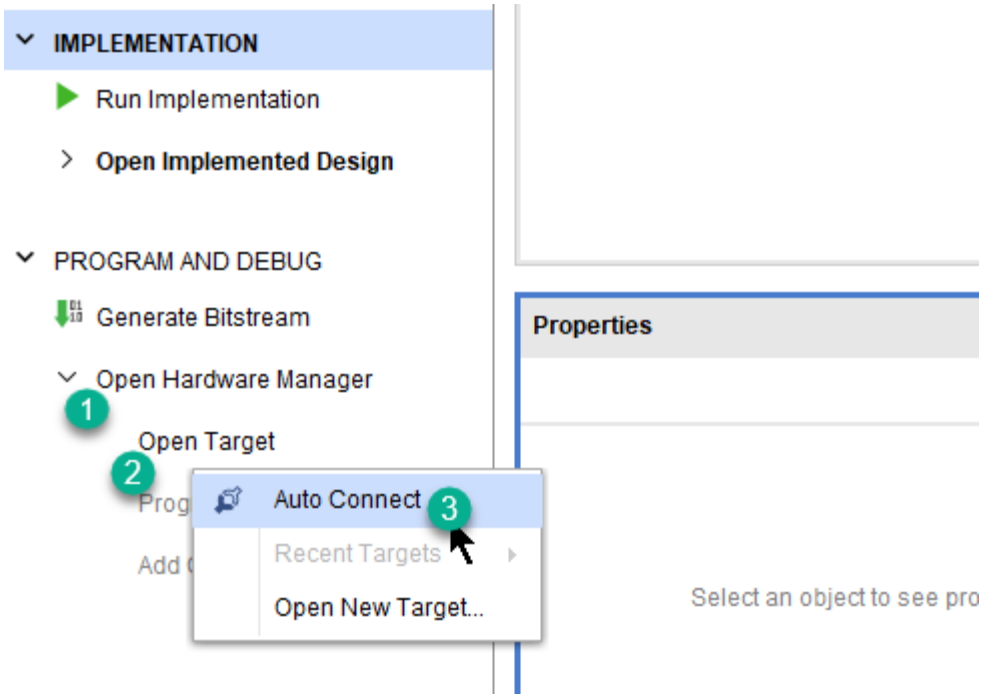


FPGA 配置文件生成后，便可将bit文件下载到开发平台以查看真实效果，此次选择第三项直接去烧写

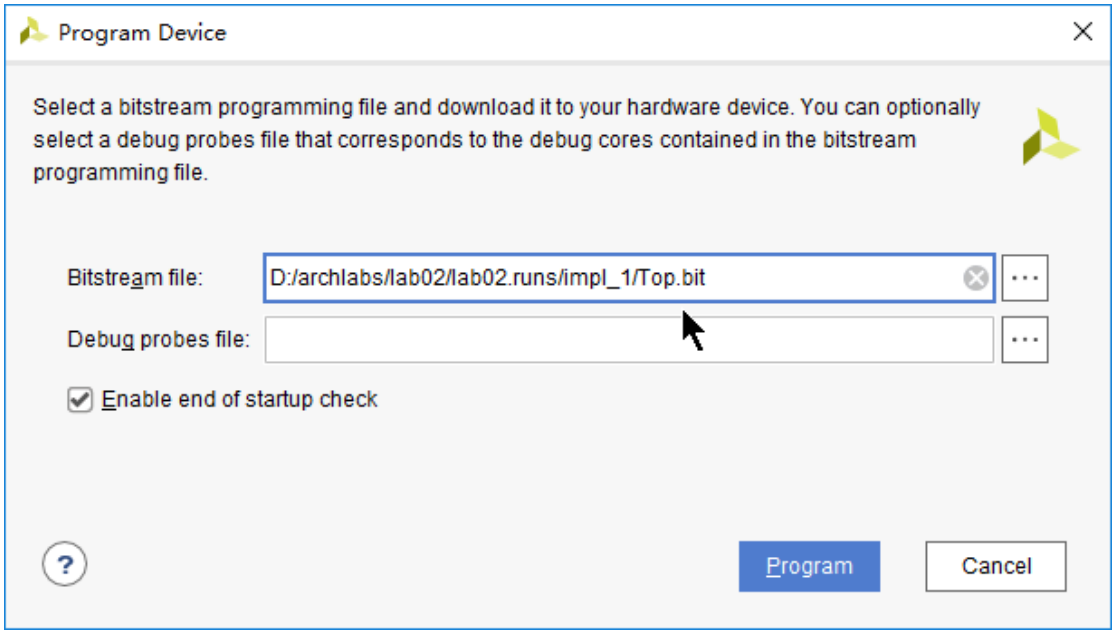


2. 连接好SWORD的电源和JTAG，然后打开开关

3. 将bit 文件下载到SWORD开发平台以查看真实效果。点击 Flow Navigator 中点击Open Hardware Manager下的Open Target再选中Auto Connect



4. 点击Open Hardware Manager下的Program Device再选择FPGA芯片xc7k325t_0，弹出的对话框中文件名处会自动加载本工程生成的bit文件，点击 Program对FPGA 芯片进行现场编程



5. 观察实验结果，显示是否预期。等待bit文件下载配置完成，尝试拨动开关，拨码开关的低4位和次高4位代表两个加数；LED的低4位代表的是和，第 5 位代表的是进位；2个七段数码管也代表和的结果。