第3章 Vivado使用指南

Vivado设计套件是Xilinx公司于2012年发布的集成设计环境，是一个基于AMBA AXI4互联规范、IP-XACT IP封装元数据、工具命令语言(TCL)、Synopsys系统约束(SDC)符合业界标准的开放式环境，能够支持多达1亿个等效ASIC门的设计。

基于Vivado的FPGA设计开发流程如图3.1所示，主要包括以下步骤。

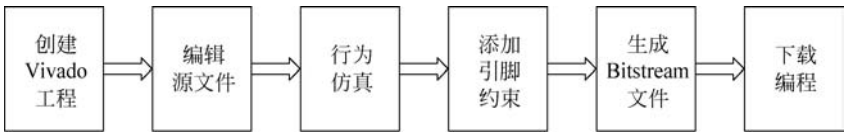

图3.1 Vivado设计的流程

(1) 创建工程。

(2) 编辑源设计文件，包括HDL文本、IP核、模块文件、网表输入等方式。

(3) 行为仿真(Behavioral Simulation)，在别的软件中也被称为功能仿真、前仿真，即不包含延时信息的仿真；Vivado自带仿真器，也可以选择采用第三方仿真工具ModelSim等工具进行仿真。

(4) 添加引脚约束。

① 综合(Synthesis)：根据设定的编译策略，对工程进行综合，生成网表文件。

② 引脚约束：通过I/O Planing或者直接编辑.xdc文件添加引脚约束信息。

③ 实现(Implimentation)：指针对某一具体的目标器件经布局布线(Place & Route)，或者称为适配(Fitting)，产生延时信息文件、报告文件(.rpt)，以供时序分析、时序仿真使用。

(5) 生成Bitstream文件，产生.bit和.bin等编程文件。

(6) 将生成的Bitstream文件下载至FPGA芯片。

设计步骤的次序并非一成不变，可根据个人习惯及实际情况进行调整和修改；同时，在设计过程中，如果出现错误(Error)，需改正错误或调整电路后重复相应的步骤；如果出现严重警告信息(Critical Warning)，也需引起注意，要不断调整和优化，直至达成设计目标。

3.1 Vivado流水灯设计

本节以使用Verilog语言设计流水灯为例，介绍在Vivado环境下运行Verilog程序的流程，包括源程序的编写、编译、仿真及下载。本例基于Vivado 18.2版本，其他不同版本的Vivado使用方法与此类似。

3.1.1 流水灯设计输入

1. 创建新工程

首先建立一个工作目录，本例的工作目录为D:/exam。

(1) 双击启动 Vivado 2018.2,出现如图 3.2 所示的 Vivado 启动界面,单击 Quick Start 栏中的 Create Project(或者选择菜单 File→New Project...命令),启动工程向导,创建一个新工程。

图 3.2 Vivado 启动界面

(2) 在工程向导(如图 3.3 所示)界面中单击 Next 按钮。

(3) 在图 3.4 所示的界面中命名工程名和存储路径,此处项目命名为 led,其存放位置为 D:/exam,勾选 Create project subdirectory 复选框,可为此工程在指定路径下建立独立的文件夹,最终整个项目存在 D:/exam/led 文件夹中。设置完成后,单击 Next 按钮。

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、下画线来组成。

(4) 选择项目类型(如图 3.5 所示)界面,选择 RTL Project 类型,单击 Next 按钮。

注意:如果在图 3.5 中勾选 Do not specify sources at this time 复选框,则跳过后面的(5)和(6),表示当前工程无须添加源文件和约束文件。

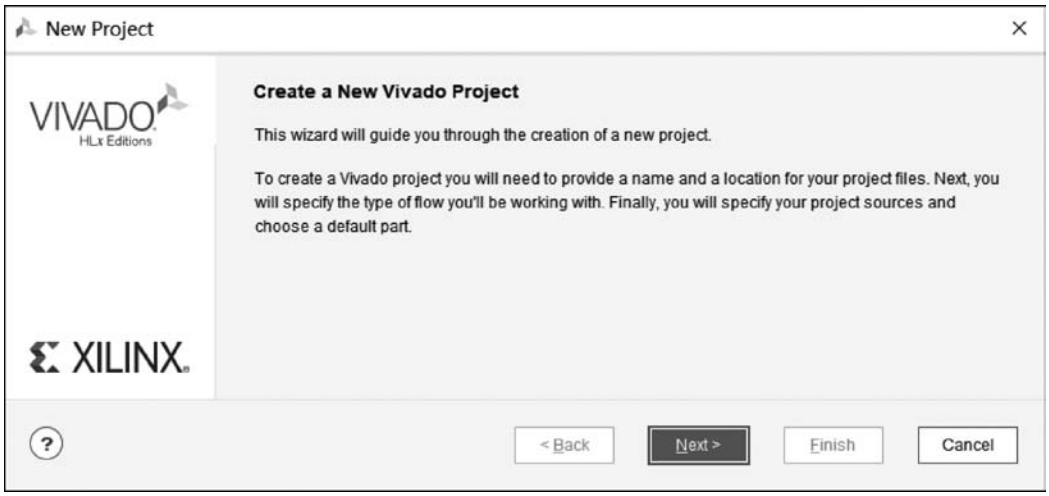

图 3.3　启动工程向导

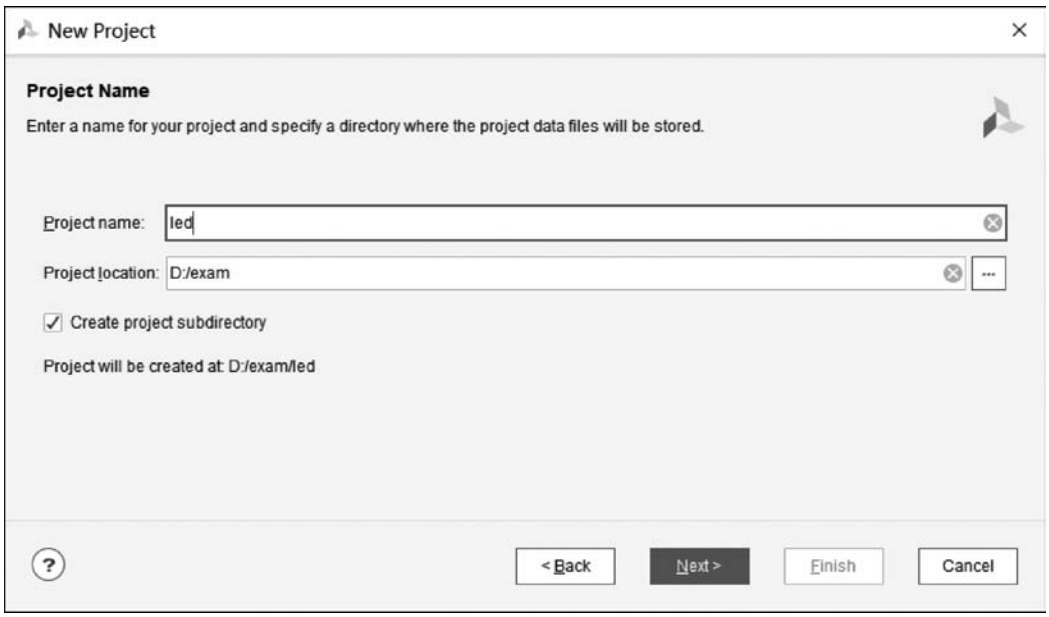

图 3.4　工程名称、路径设定界面

(5) 在图 3.6 所示的 Add Sources 界面中添加源文件并选择设计语言,其中 Target language 和 Simulator language 均选择 Verilog,单击 Next 按钮。

(6) 不添加约束文件,所以在 Add Constraints 界面直接单击 Next 按钮。

(7) 在图 3.7 所示的器件选择界面中,根据使用的 FPGA 开发板,选择相应的 FPGA 目标器件。本例中,以 Xilinx EGO1 为目标板,故 FPGA 选择 xc7a35tcsg324-1,即 Family 选择 Artix-7,封装形式(Package)为 csg324,单击 Next 按钮。

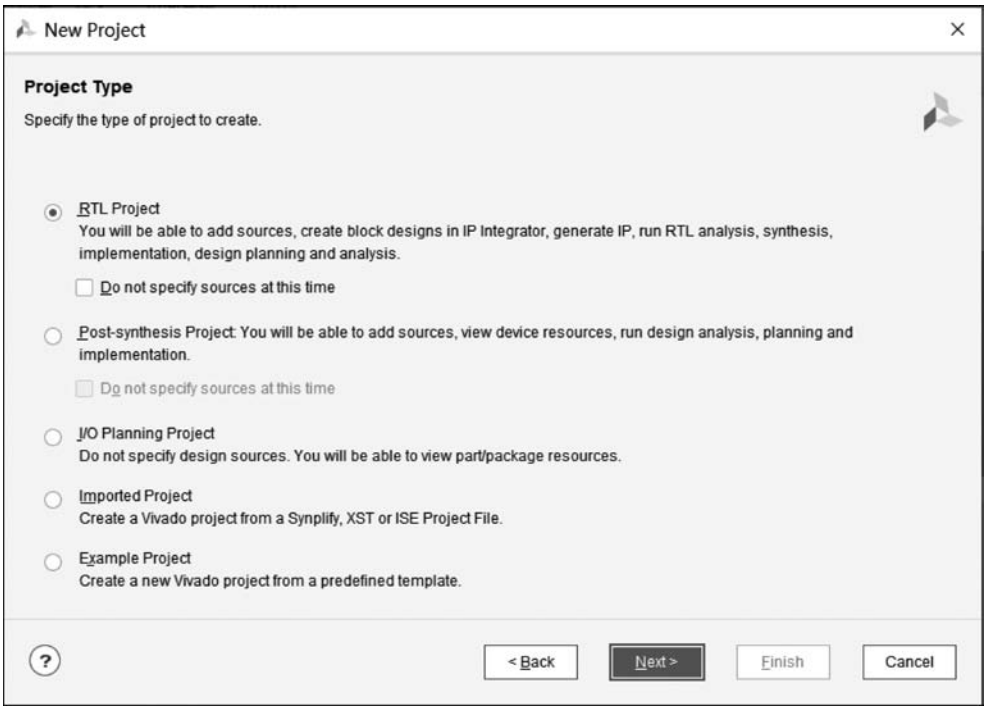

图 3.5　选择工程类型

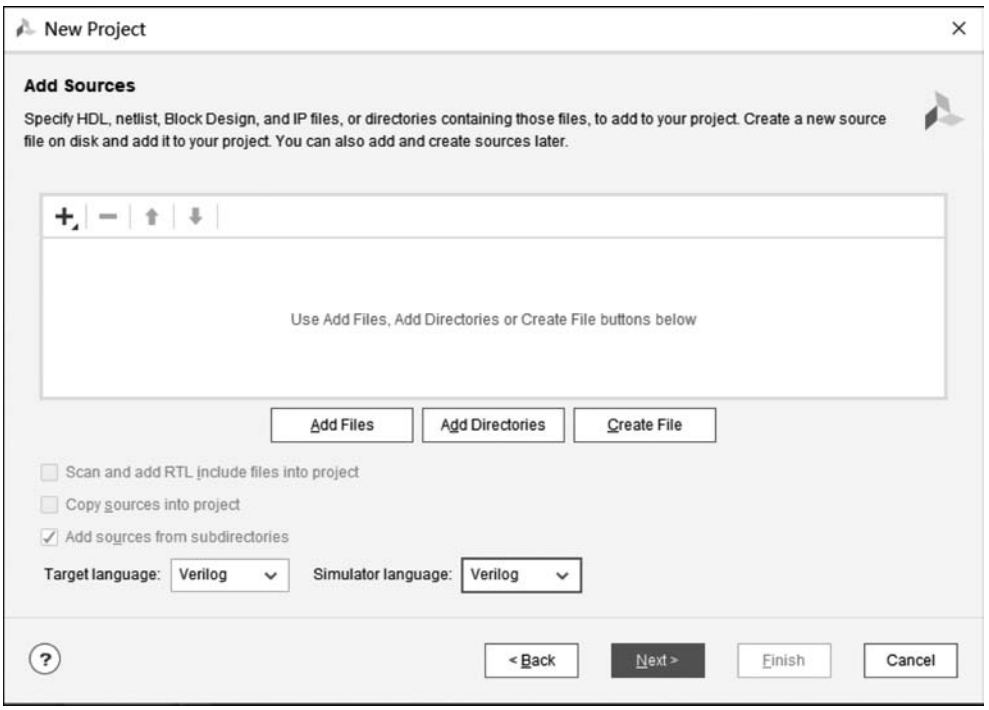

图 3.6　添加源文件并选择设计语言

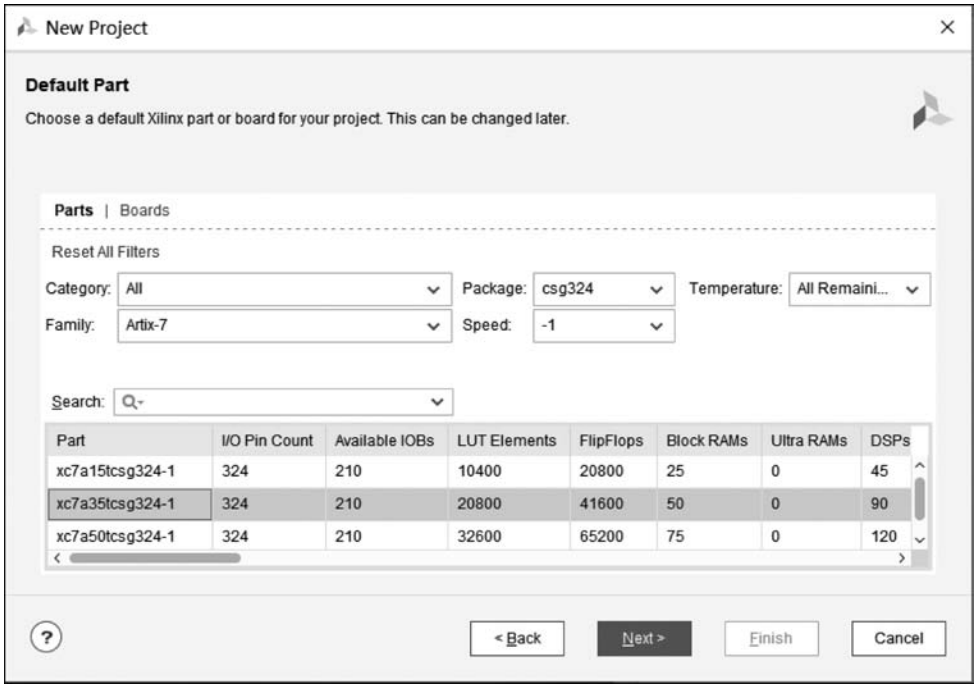

图 3.7　器件选择界面

(8) 最终出现图 3.8 所示的界面,对工程信息进行汇总,确认相关信息正确与否,包括工程类别、源文件、所用的 FPGA 器件等。如果没有问题,则单击 Finish 按钮完成工程的创建;如果有问题,则返回前面界面进行修改。

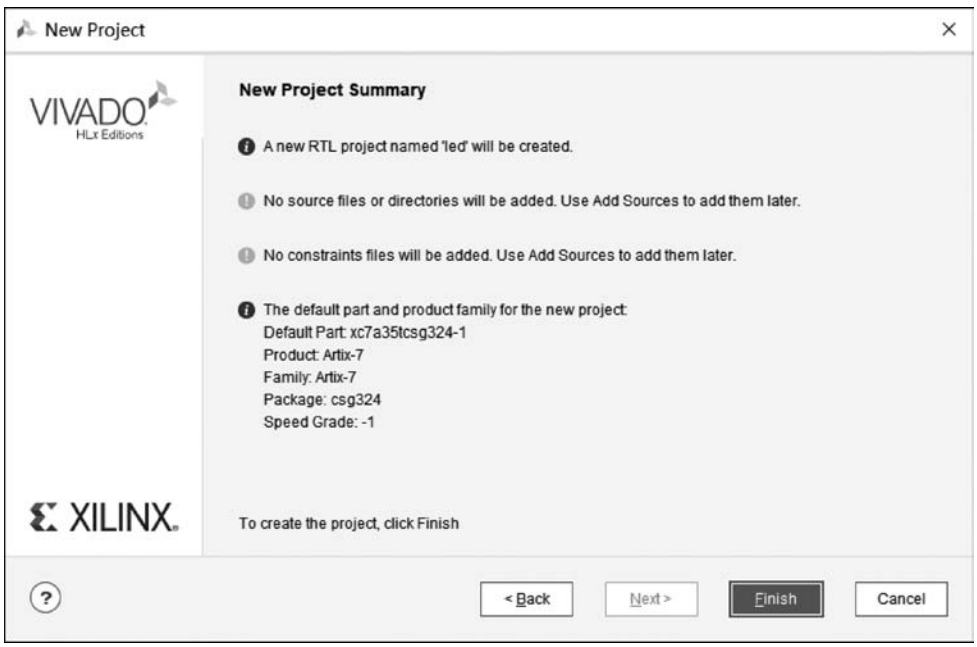

图 3.8　工程信息汇总

2. 输入源设计文件

(1) 如图 3.9 所示,选择 Flow Navigator 下 PROJECT MANAGER 中的 Add Sources 选项,打开设计文件导入窗口。

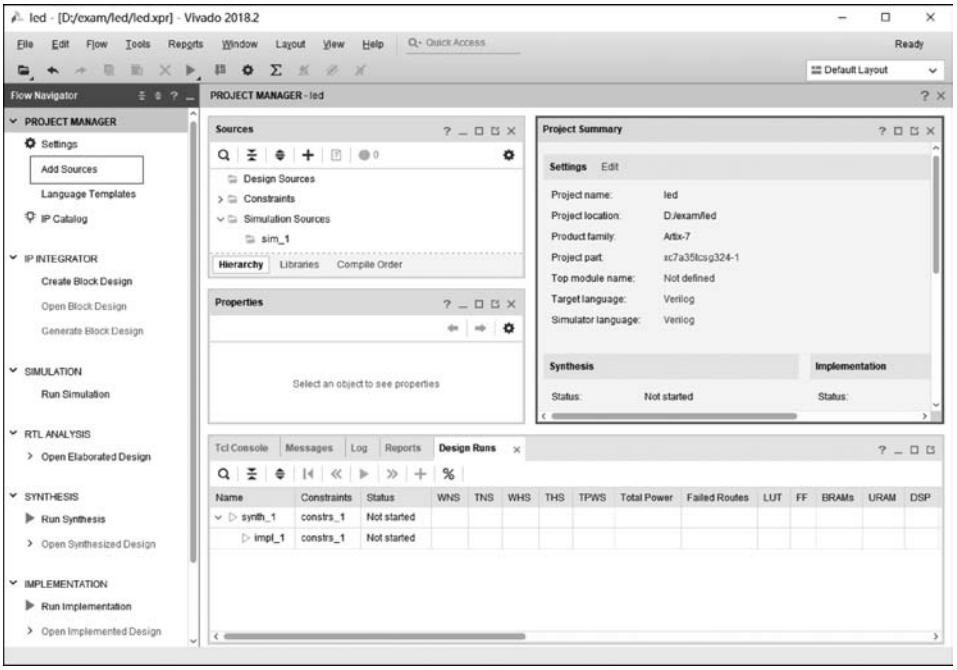

图 3.9 工程管理窗口

(2) 在 Add Sources 界面(如图 3.10 所示)中选中 Add or create design sources 单选按钮,表示添加或创建 Verilog(或 VHDL)源文件,单击 Next 按钮。

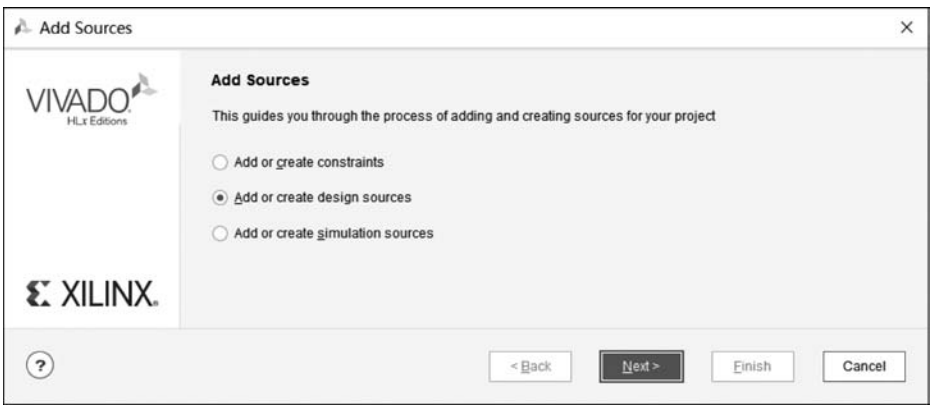

图 3.10 添加或创建源文件

(3) 在图 3.11 中单击 Create File 按钮,在弹出的 Create Source File 对话框中输入 File name 为 flow_led,单击 OK 按钮。

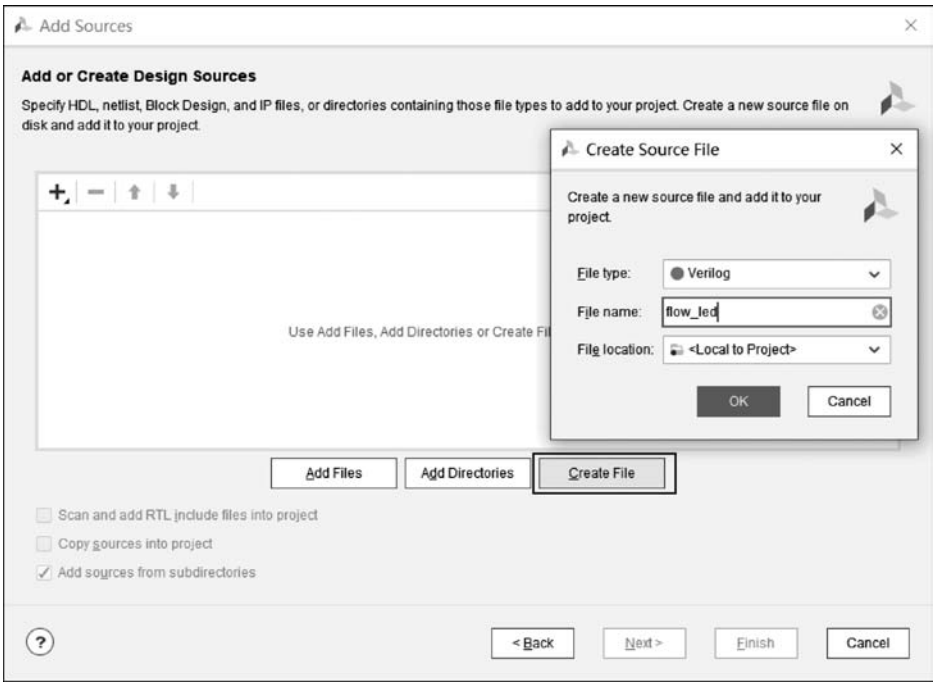

图 3.11　创建源文件

注意：文件名中不可出现中文和空格；如果有现成的.v 或.VHD 文件，可单击 Add Files 或者 Add Directories 按钮进行添加。

(4) 单击图 3.12 中的 Finish 按钮，完成源文件的创建。

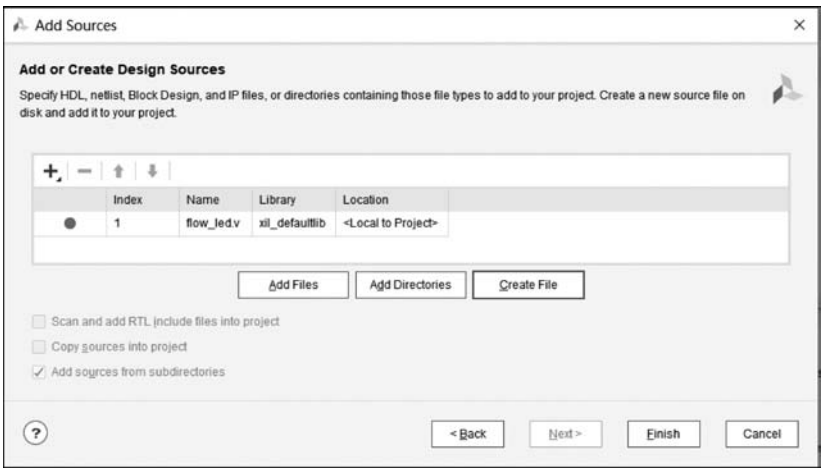

图 3.12　完成源文件创建

(5) 在弹出的 Define Module 对话框中填写模块名称，此处模块命名为 flow_led，如图 3.13 所示。还可以在 I/O Port Definitions 栏中填写模块的端口并设置端口方向，如果端口为总线型，勾选 Bus 选项，并通过 MSB 和 LSB 确定总线宽度。完成后单击 OK 按钮。

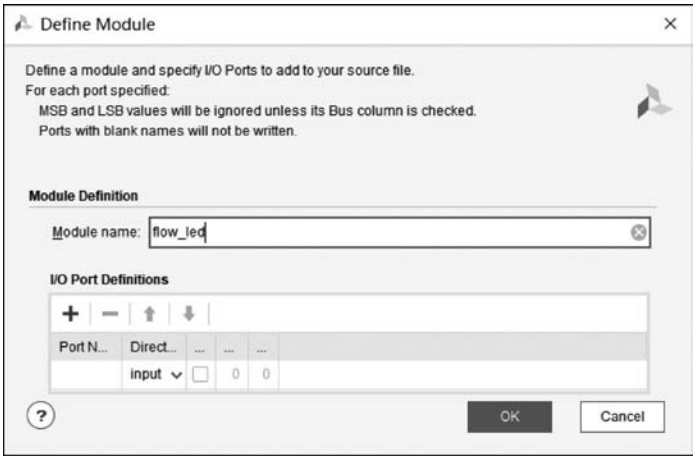

图 3.13　Define Module 对话框

(6) 当前 Vivado 界面如图 3.14 所示，在中间的 Sources 窗格的 Design Sources 中出现新建的设计文件 flow_led.v，双击打开该文件，利用 Vivado 自带的文本编辑器(Text Editor)输入设计代码。本例 LED 流水灯的代码如例 3.1 所示。

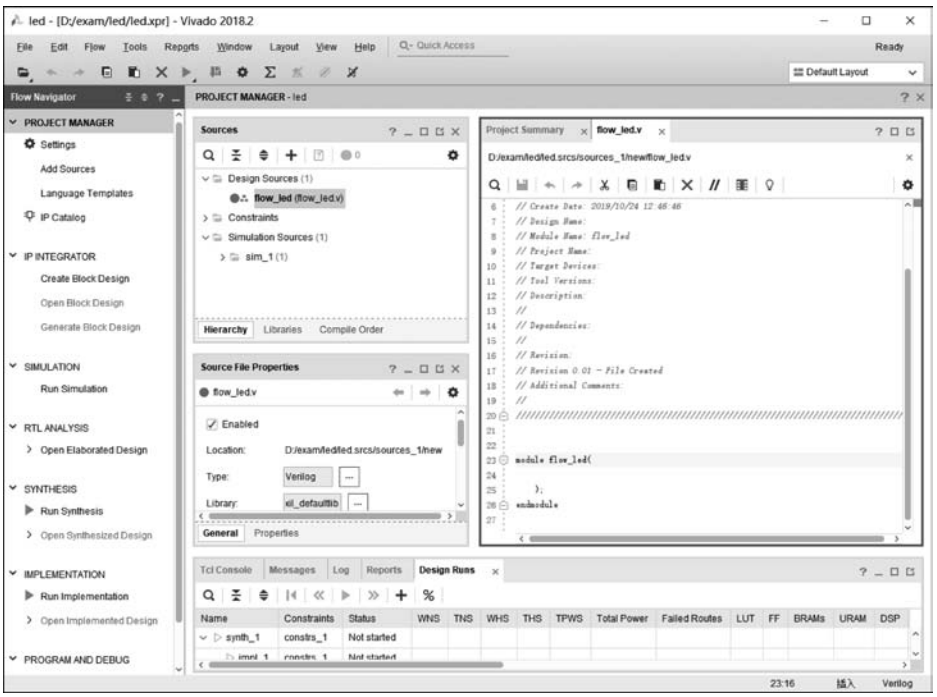

图 3.14　Verilog 代码编辑窗口

【例 3.1】 8 位流水灯源代码。

```
module flow_led(clk,clr,led);
  input clk,clr;
```

```
    output reg [7:0] led;
    reg [28:0] counter;

always @(posedge clk)
begin
if(!clr) begin counter <= 0;led <= 8'h01; end
    else
    if( counter < 50000000)          //2Hz
       counter <= counter + 1;
    else
       begin
           counter <= 0;
           led <= {led[6:0],led[7]};
       end
  end
endmodule
```

3.1.2 行为仿真

至此，已完成源文件输入，此时可对源文件进行行为（功能）仿真，以测试其功能。

（1）创建激励测试文件，在 Sources 中右击选择 Add Sources，在出现的 Add Sources 界面中（见图 3.10）选择第三项 Add or create simulation sources 单选按钮，单击 Next 按钮。

（2）在如图 3.15 所示的界面中单击 Create File 按钮，创建一个仿真激励文件，在弹出的 Create Source File 对话框中输入激励文件名称为 tb_led，文件类型为 Verilog，单击 OK 按钮，确认添加完成后单击 Finish 按钮。

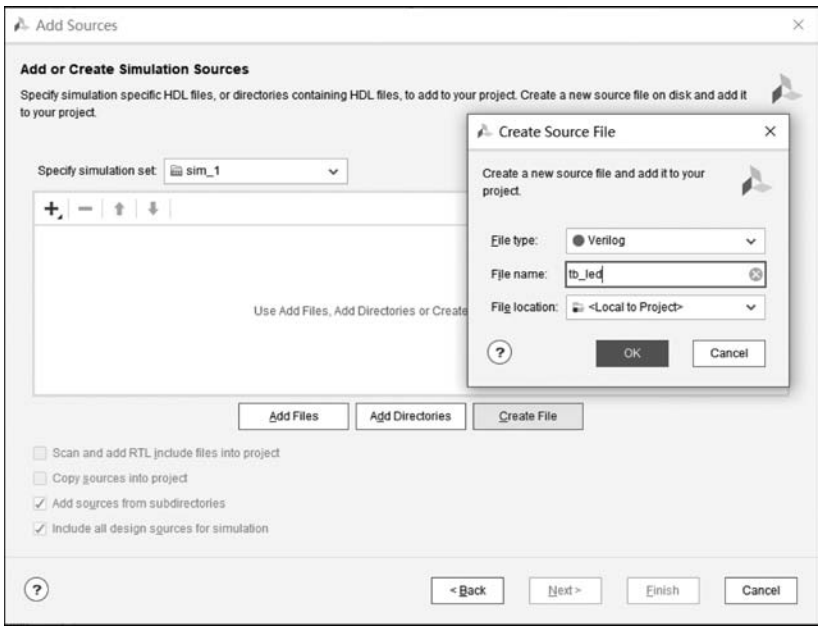

图 3.15　创建仿真激励文件

(3) 在如图 3.16 所示的仿真模块定义界面中填写仿真模块的名字为 tb_led,因为是激励文件不需要对外端口,所以 I/O Port Definitions 部分无须填写,单击 OK 按钮。

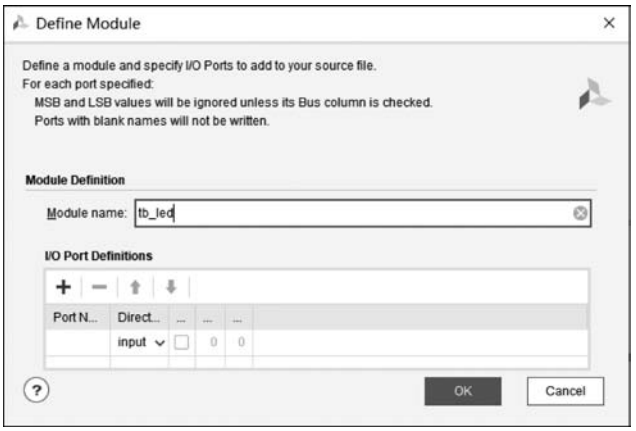

图 3.16　仿真模块定义界面

(4) Vivado 界面如图 3.17 所示,在 Sources 窗格的 Simulation Sources 中出现新建的仿真文件 tb_led.v,双击打开该文件,利用 Vivado 的文本编辑器输入激励代码。本例 LED 流水灯的 Test Bench 激励代码如例 3.2 所示。

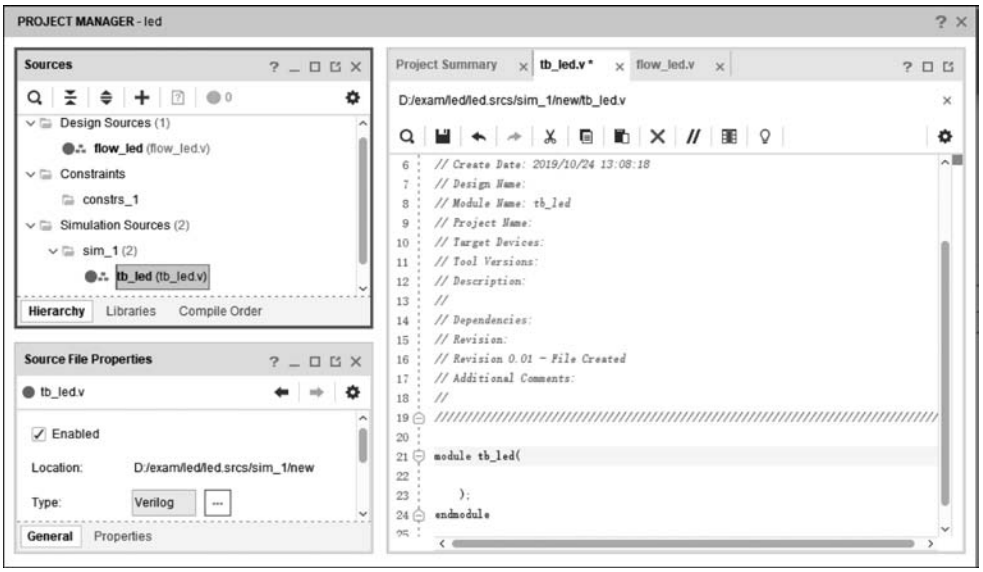

图 3.17　Vivado 工程管理界面

【例 3.2】 LED 流水灯的 Test Bench 激励代码。

```
`timescale 1ns/1ns
module tb_led( );
parameter DELY = 20;
```

```
reg clk;
reg clr;
wire [7:0] led;
flow_led i1(
    .clk (clk),
    .clr (clr),
    .led (led));
initial begin
clk = 1'b0; clr = 1'b0;
#(DELY * 2) clr = 1'b1;
end
always
begin
#(DELY/2)   clk = ~clk;
end
endmodule
```

(5) 在 Flow Navigator 中选择 SIMULATION 下的 Run Simulation 选项,并选择 Run Behavioral Simulation,启动仿真界面,如图 3.18 所示。

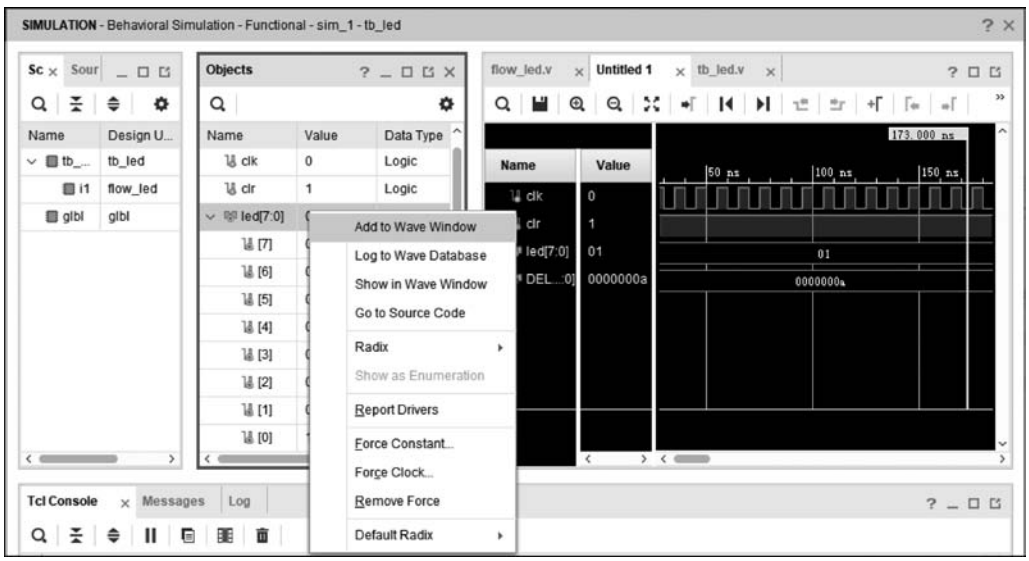

图 3.18　仿真界面

端口信号自动出现在波形图中,此外,可通过左侧 Scope 一栏中的目录结构定位到想要查看的 Module 内部寄存器,在 Objects 对应的信号名称上右击选择 Add To Wave Window,将信号加入波形图中查看。

(6) 可通过仿真工具条来对仿真进行设置和操作。仿真工具条如图 3.19 所示,包括复位波形(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、暂停等操作。本例中仿真时长设置为 500ms。

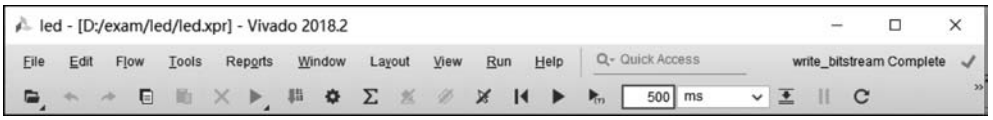

图 3.19　仿真工具条

(7) 最终得到的仿真波形如图 3.20 所示，检查此波形是否与预想的功能一致，以验证源设计文件的正确性。

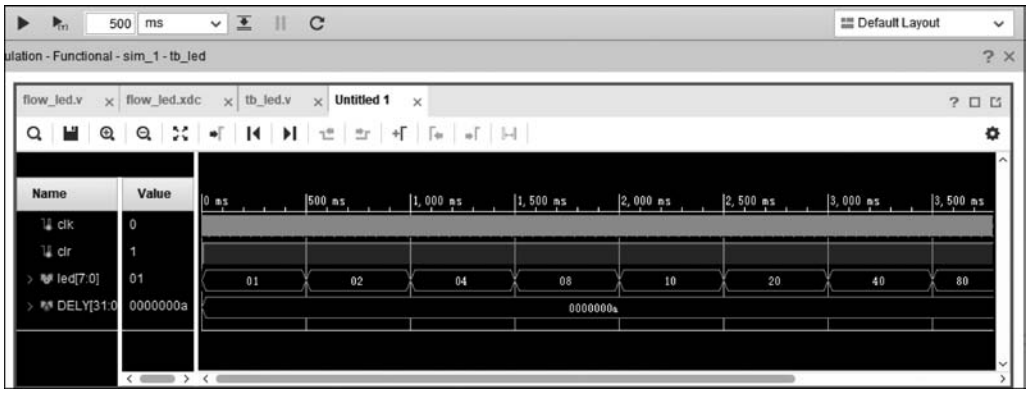

图 3.20　行为仿真波形图

3.1.3　综合与引脚的约束

1. 综合编译

(1) 如图 3.21 所示，在 Flow Navigator 中选择 SYNTHESIS 下的 Run Synthesis 选项，对当前工程进行综合，弹出 Launch Runs 对话框，在 Options 中选中 Launch runs on local host 单选按钮，在 Number of jobs 下拉列表中选择最大值，以缩短编译时间，此处选择 8。

(2) 编译成功后，在 Flow Navigator 中选择 Synthesis 下的 Schematic 选项，可以查看综合后的电路图。本例的综合后的电路如图 3.22 所示。

2. 添加引脚约束文件

有两种方法可以添加引脚约束，第一种是利用 Vivado 中的 I/O Planning 功能(需先对工程进行综合，在综合后选择打开 Open Synthesis Design，然后在右下方的选项卡中切换到 I/O Ports 栏，在对应的信号后输入对应的 FPGA 引脚号)；第二种是直接新建 XDC 约束文件。本例采用方法二。XDC(Xilinx Design Constraints)是 Vivado 采用的约束文件格式，它是在业界广泛采用的 SDC(Synopsys Design Constraints)格式的基础上，加入 Xilinx 自身的一些物理约束来实现的。

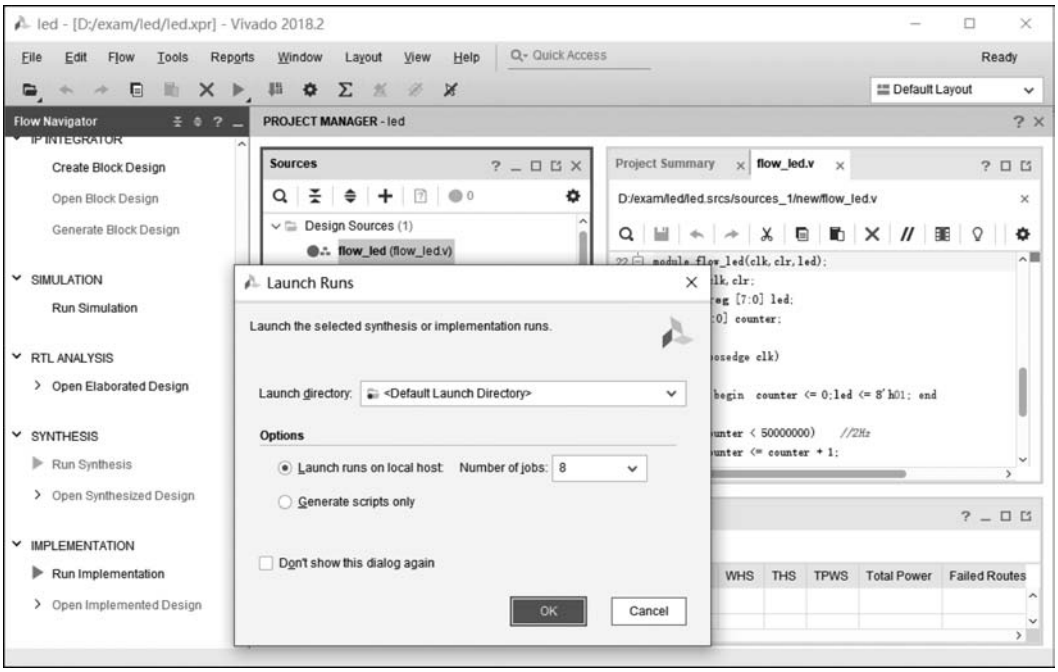

图 3.21　Synthesis 综合编译

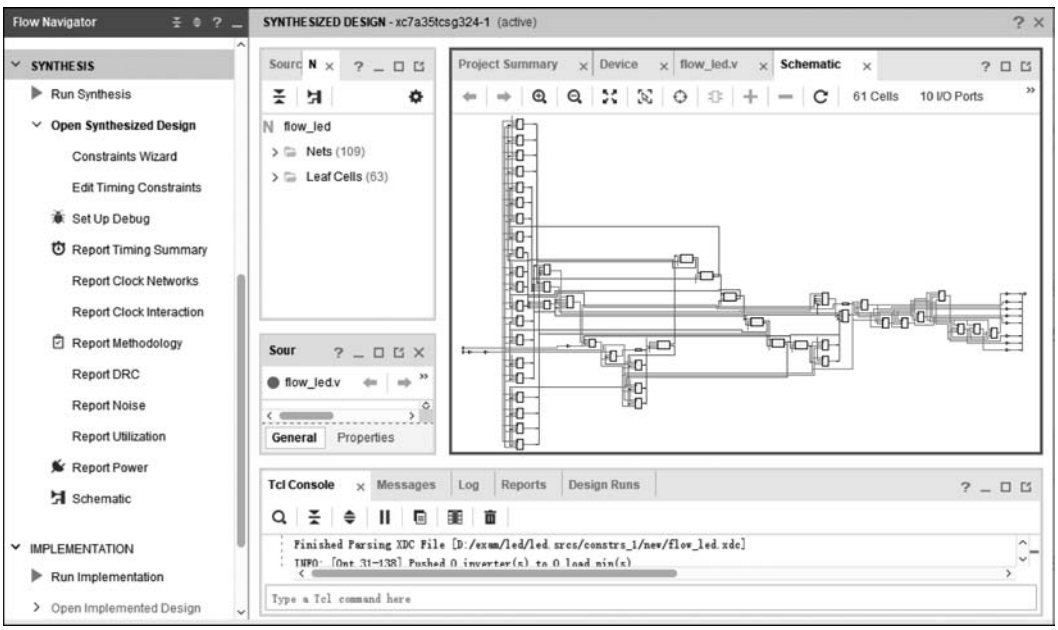

图 3.22　综合后的电路图

(1) 在 Flow Navigator 中选择 PROJECT MANAGER 下的 Add Sources 选项(或右击约束子目录下的文件夹,选择 Add Sourses...),打开如图 3.23 所示的 Add Sources 界面,选中第一项 Add or create constraints 单选按钮,单击 Next 按钮。

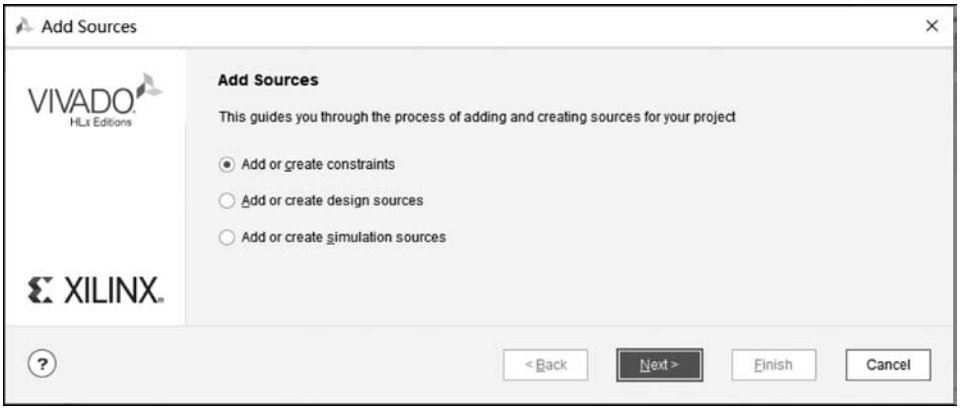

图 3.23 创建约束文件

(2) 在图 3.24 所示的界面中,单击 Create File,在弹出的 Create Constraints File 对话框中输入 XDC 文件名,本例填写 flow_led,单击 OK 按钮,再单击 Finish 按钮。

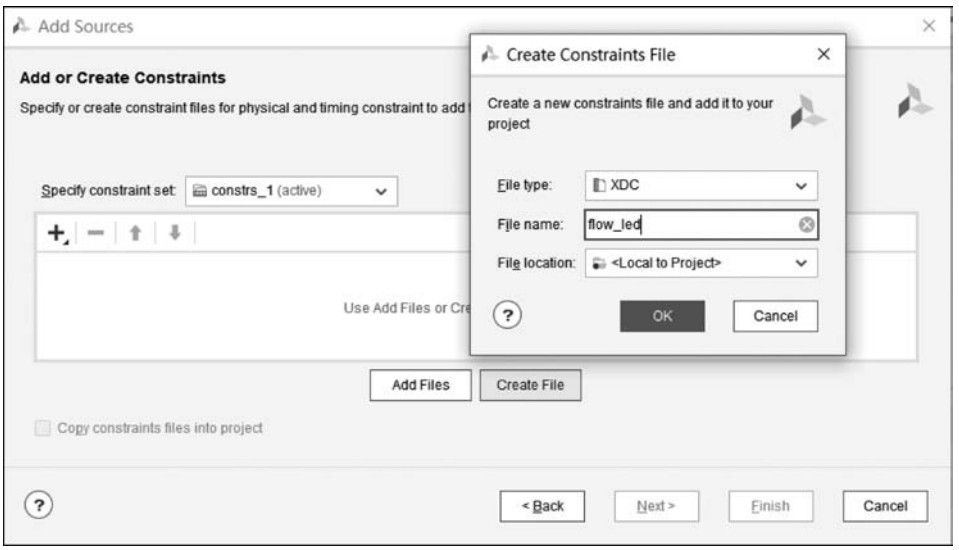

图 3.24 输入约束文件名

(3) 如图 3.25 所示,在 Sources 窗格中双击 flow_led.xdc 文件名,打开该文件,填写引脚约束文件的内容。本例的引脚约束文件内容如例 3.3 所示。

注意:具体的 FPGA 约束引脚号和 I/O 电平标准,应参考目标板卡的用户手册或原理图。

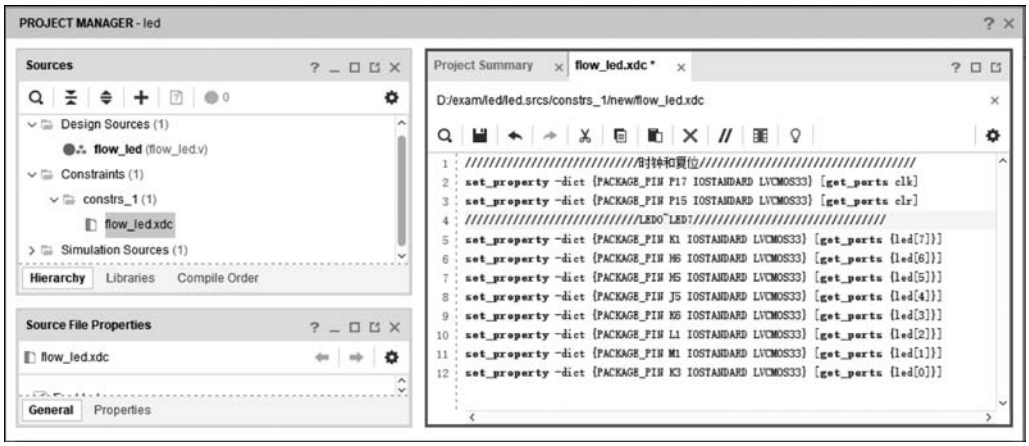

图 3.25　编辑引脚约束文件

【例 3.3】 LED 流水灯的.XDC 引脚约束文件。

```
#////////////////////////////时钟和复位////////////////////////////////////
set_property -dict {PACKAGE_PIN P17 IOSTANDARD LVCMOS33} [get_ports clk]
set_property -dict {PACKAGE_PIN P15 IOSTANDARD LVCMOS33} [get_ports clr]
#////////////////////////////LED0～LED7////////////////////////////////////
set_property -dict {PACKAGE_PIN K1 IOSTANDARD LVCMOS33} [get_ports {led[7]}]
set_property -dict {PACKAGE_PIN H6 IOSTANDARD LVCMOS33} [get_ports {led[6]}]
set_property -dict {PACKAGE_PIN H5 IOSTANDARD LVCMOS33} [get_ports {led[5]}]
set_property -dict {PACKAGE_PIN J5 IOSTANDARD LVCMOS33} [get_ports {led[4]}]
set_property -dict {PACKAGE_PIN K6 IOSTANDARD LVCMOS33} [get_ports {led[3]}]
set_property -dict {PACKAGE_PIN L1 IOSTANDARD LVCMOS33} [get_ports {led[2]}]
set_property -dict {PACKAGE_PIN M1 IOSTANDARD LVCMOS33} [get_ports {led[1]}]
set_property -dict {PACKAGE_PIN K3 IOSTANDARD LVCMOS33} [get_ports {led[0]}]
```

3.1.4　生成比特流文件并下载

(1) 如图 3.26 所示，在 Flow Navigator 中选择 PROGRAM AND DEBUG 下的 Generate Bitstream 选项，工程会自动完成综合、实现比特流文件的生成过程。完成后，选中 Open Hardware Manager 单选按钮，进入硬件编程管理界面。

(2) 进入如图 3.27 所示的 HARDWARE MANAGER 对话框，将目标板通过 USB 连接至计算机，打开电源开关，单击图 3.27 中的 Open target，选择 Auto Connect 选项，使软件连接到目标板。

(3) 软件和目标板连接成功后，软件界面如图 3.28 所示。

在目标芯片上右击，选择 Program device，在弹出的 Program Device 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 flow_led.bit，单击 Program 按钮对 FPGA 芯片进行编程下载。

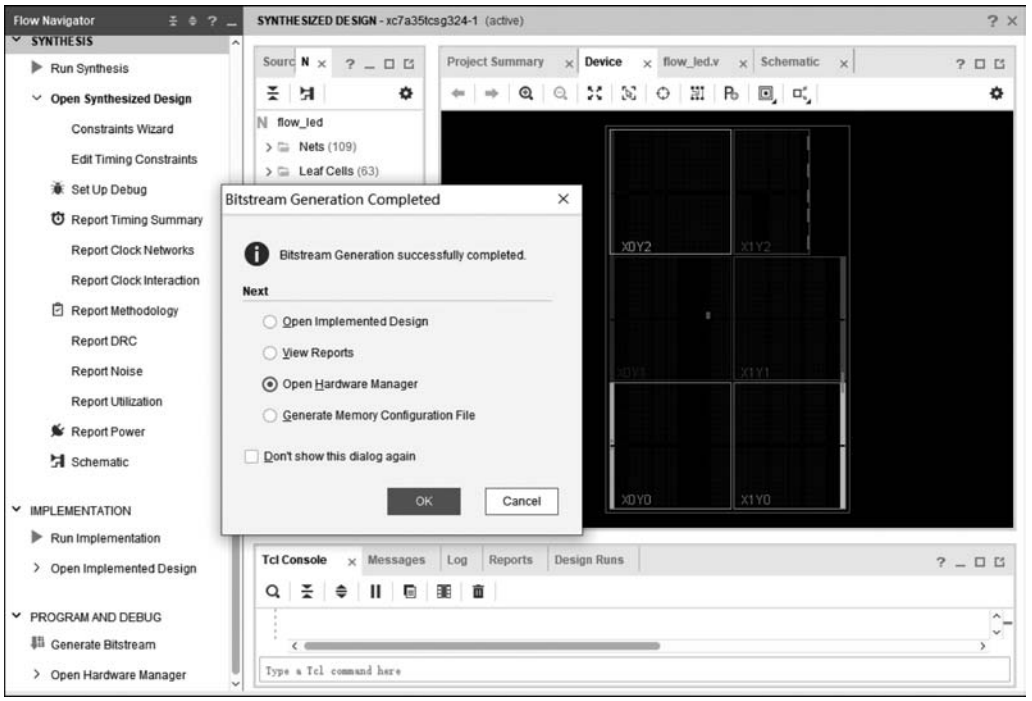

图 3.26　生成比特流文件

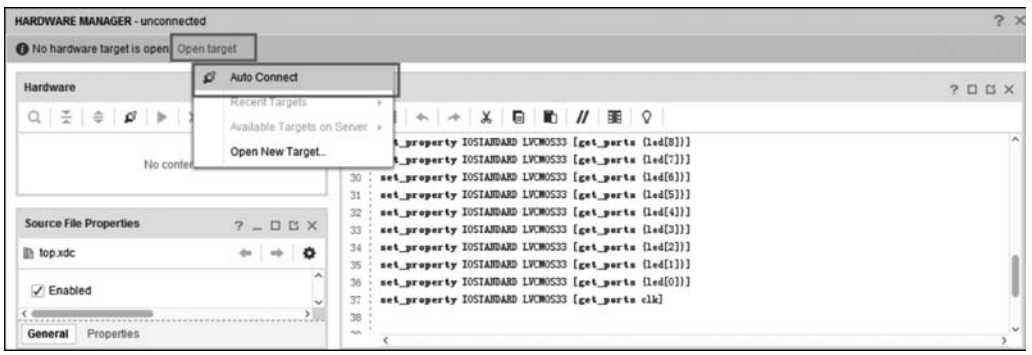

图 3.27　连接到目标板

(4) 下载完成后,在目标板上观察实际运行效果。

3.1.5　将配置数据烧写至 Flash 中

如果将程序烧写到 Flash(ROM)中,则程序会固化到板卡中,可脱机独立运行且掉电不丢失。

(1) 生成烧录至 Flash 中的.bin 文件,选择菜单 Tools 中的 Settings 命令,在弹出的 Settings 对话框(如图 3.29 所示)中选择 Bitstream,在右面勾选-bin_file,单击 OK 按钮。

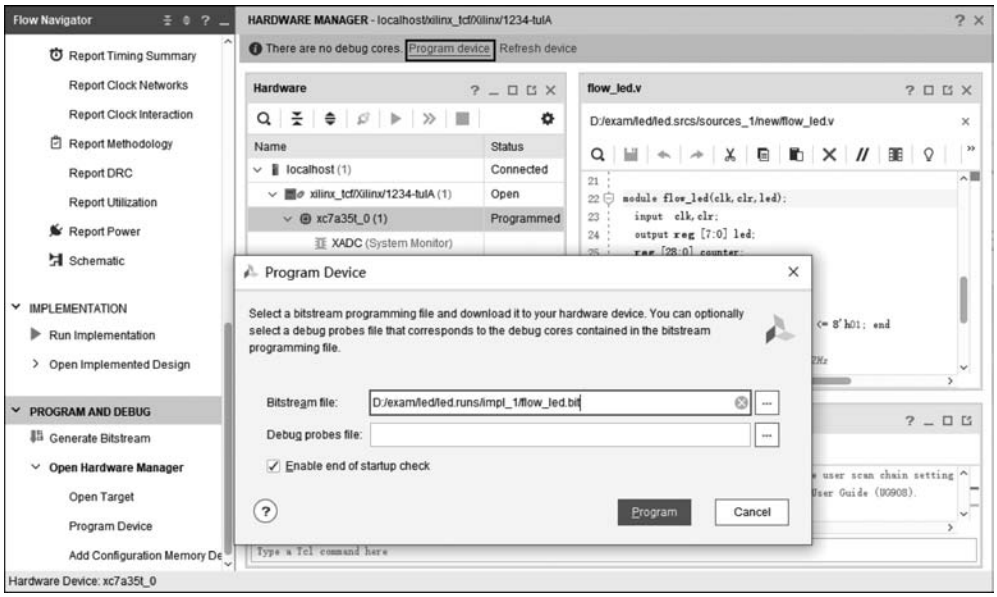

图 3.28　芯片编程下载

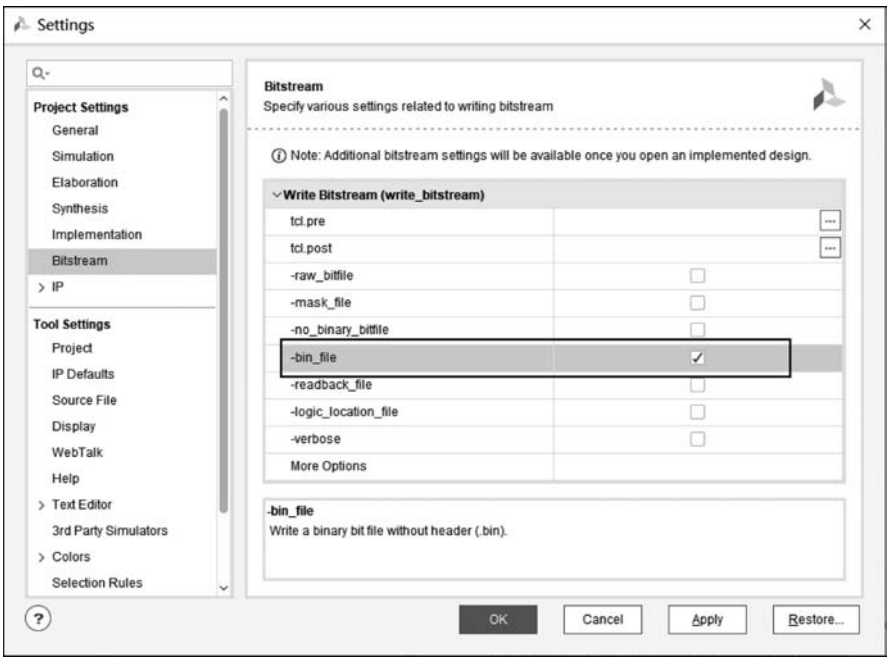

图 3.29　勾选_bin_file

(2) 在 Flow Navigator 中选择 PROGRAM AND DEBUG 下的 Generate Bitstream 选项(见图 3.26),启动编译并自动生成.bit 文件和用于固化的.bin 文件。

(3) 将目标板连接至计算机,打开电源,进入 HARDWARE MANAGER 对话框,如图 3.30 所示,选中芯片 xc7a35t,右击选择 Add Configuration Memory Device。

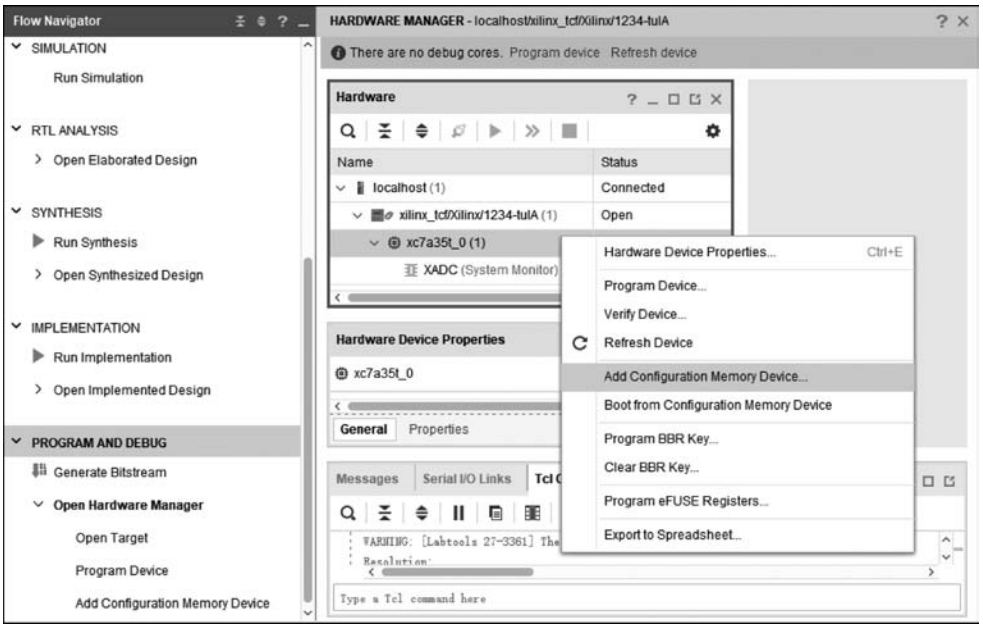

图 3.30　添加 Flash 芯片

(4) 在 Add Configuration Memory Device 对话框(如图 3.31 所示)的搜索框中输入 n25q64,选择 n25q64-3.3v(根据所用的目标板,选择相应的 Flash 芯片型号),单击 OK 按钮。

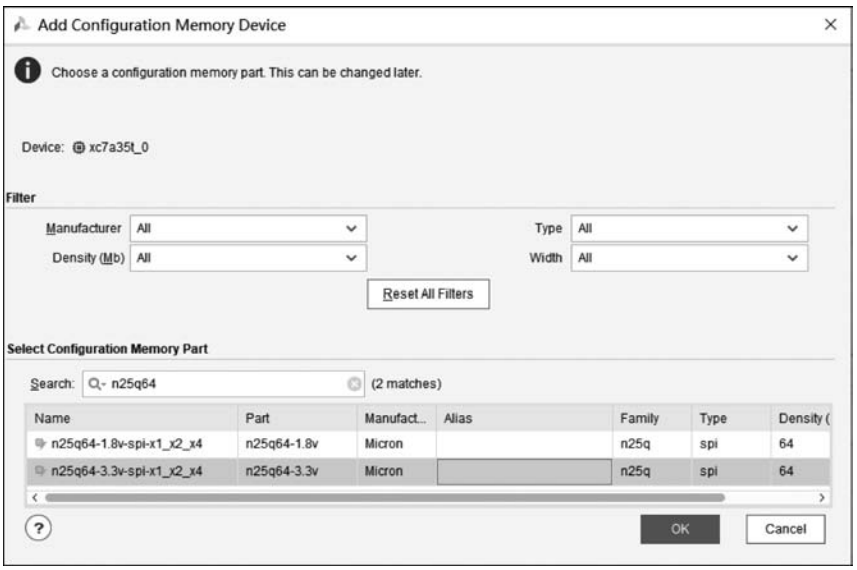

图 3.31　选择 Flash 芯片型号

(5) 在 HARDWARE MANAGER 对话框中(如图 3.32 所示),选中 Flash 芯片 n25q64-3.3v,右击选择 Program Configuration Memory Device,弹出如图 3.33 所示的对话框,确认配置文件为 flow_led.bin,单击 OK 按钮,完成对 Flash 芯片的编程。

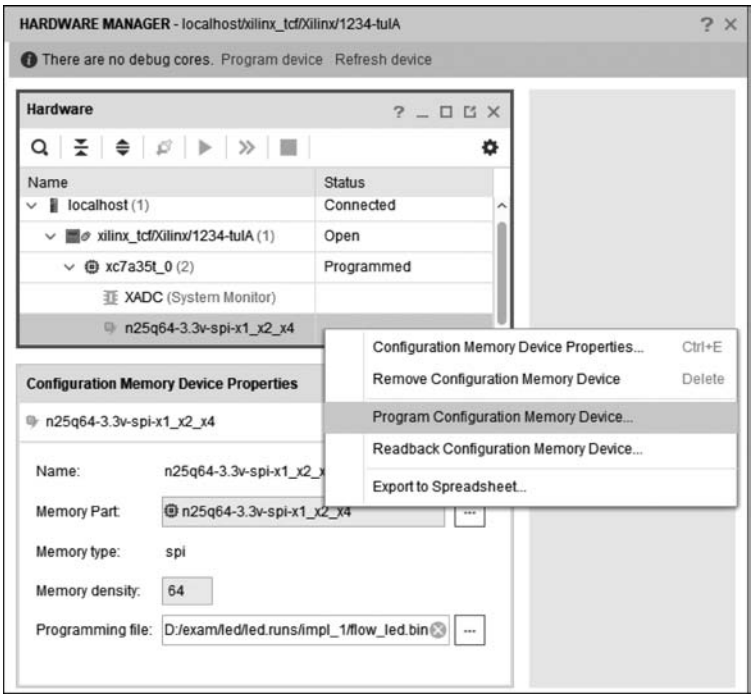

图 3.32　选中 Flash 芯片

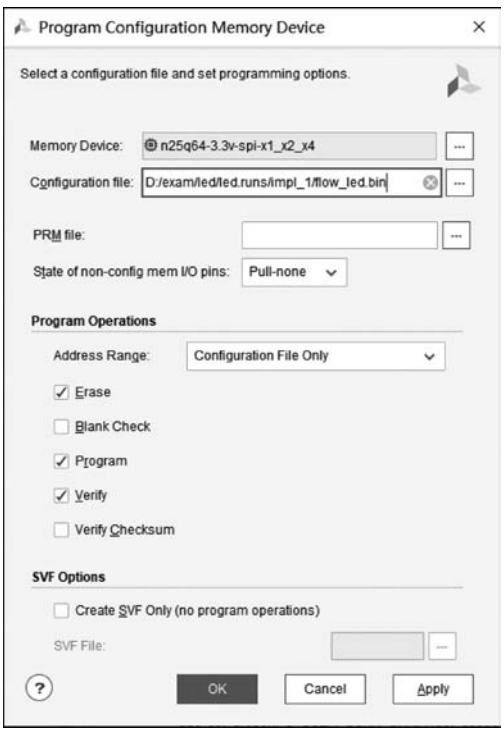

图 3.33　对 Flash 芯片编程

(6) 编程完成后,将开发板断电再重新上电,开发板会从 Flash 中启动,观察开发板的实际运行效果。

3.2 IP 核的创建和封装

基于 IP 核的设计对提高设计的复用具有优越性。Vivado 本身自带了丰富的 IP 核,还允许设计者自己定义和封装 IP 核。本节以设计和封装功能类似 74LS161 和 74LS00 的 IP 核为例,介绍基于 Vivado 的 IP 核封装流程。

1. 创建工程

启动 Vivado 2018.2,单击 Quick Start 栏中的 Create Project,启动工程向导,创建一个新工程,将其命名为 ip_161,存于 D:/exam/ip_161 文件夹中,如图 3.34 和图 3.35 所示。工程创建的过程可参考 3.1 节,此处不再赘述。

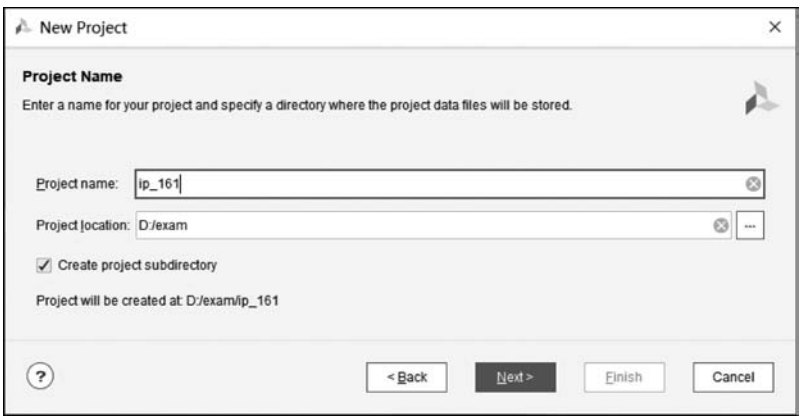

图 3.34 工程名称、路径设定

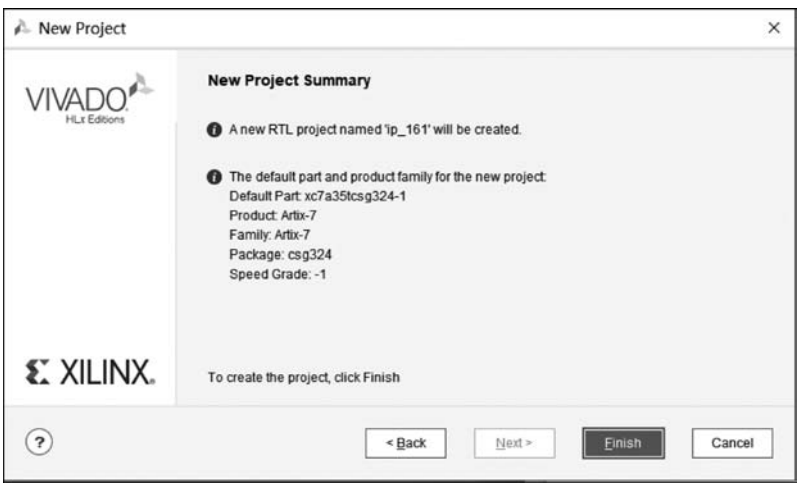

图 3.35 工程信息汇总

2. 输入源设计文件

在 Flow Navigator 中选择 PROJECT MANAGER 下的 Add Sources 选项，在弹出的界面中选中 Add or Create Design Sources 单选按钮(参见图 3.10)，创建一个名为 ls161.v 的源文件，其代码如例 3.4 所示，输入源文件后的 Vivado 界面如图 3.36 所示。

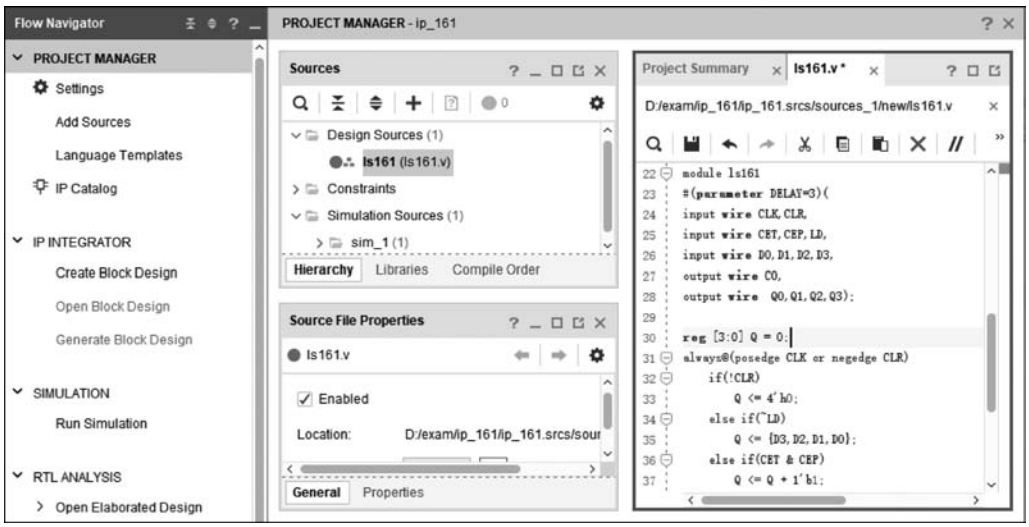

图 3.36 输入源设计文件

【例 3.4】 ls161 核的源代码。

```
module ls161
#(parameter DELAY = 3)(
         input wire CLK,CLR,
         input wire CET,CEP,LD,
         input wire D0,D1,D2,D3,
         output wire CO,
         output wire  Q0,Q1,Q2,Q3);

reg [3:0] Q = 0;
always@(posedge CLK or negedge CLR)
   if(!CLR)
      Q <= 4'h0;
   else if(~LD)
      Q <= {D3,D2,D1,D0};
   else if(CET & CEP)
      Q <= Q + 1'b1;
   else Q <= Q;

assign #DELAY Q0 = Q[0];
assign #DELAY Q1 = Q[1];
assign #DELAY Q2 = Q[2];
```

```
    assign #DELAY Q3 = Q[3];
    assign CO = ((Q == 4'b1111)&&(CET == 1'b1))? 1 : 0;

    endmodule
```

在 Flow Navigator 中选择 SYNTHESIS 下的 Run Synthesis 选项，对当前工程进行综合。综合完成后在弹出的 Synthesis Completed 对话框中单击 Cancel 按钮，表示不再继续进行后续操作。

3. 创建 IP 核

(1) 在 Flow Navigator 中选择 PROJECT MANAGER 下的 Settings 选项，弹出 Settings 对话框，如图 3.37 所示，在左侧选中 IP 下面的 Packager，在右侧的 Packager 标签页中定制 IP 核的库名和目录。

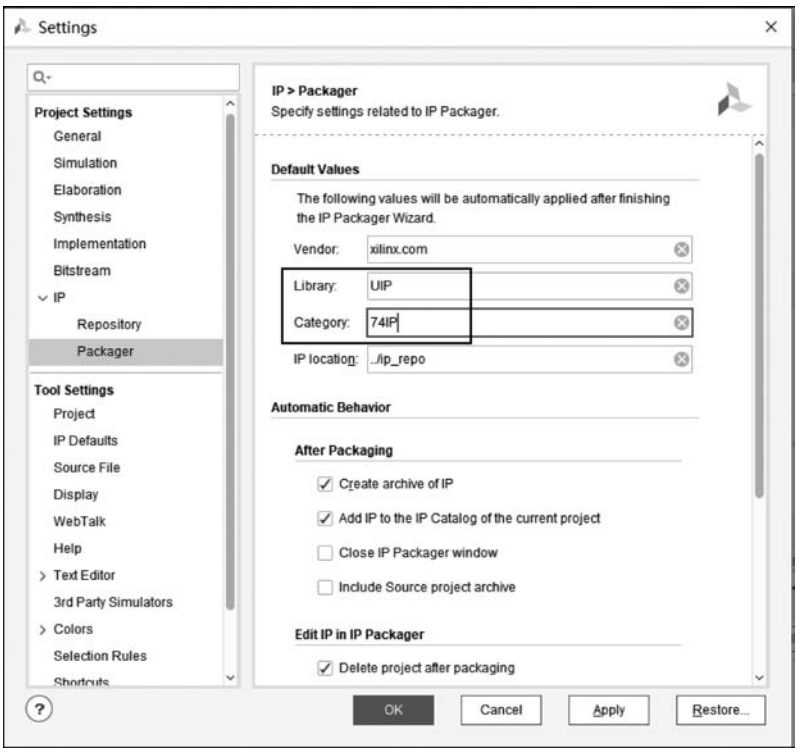

图 3.37　定制 IP 核属性

在 Library(库名)处填写 UIP，Category 处填写 74IP，勾选 After Packaging 下的 Create archive of IP、Add IP to the IP Catalog of the current project 复选框，其他按默认设置。

设置完成后单击 Apply 按钮，再单击 OK 按钮。

(2) 在 Vivado 主界面中，选择菜单 Tools 中的 Create and Package New IP 命令，如图 3.38 所示，启动创建和封装新 IP 的过程。此过程的启动界面如图 3.39 所示。

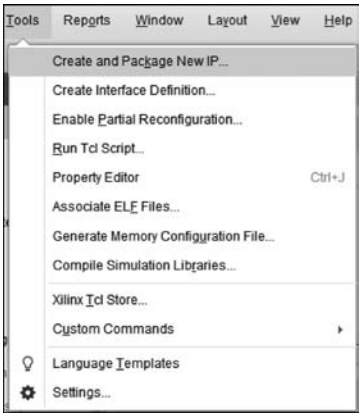

图 3.38　创建和封装新的 IP

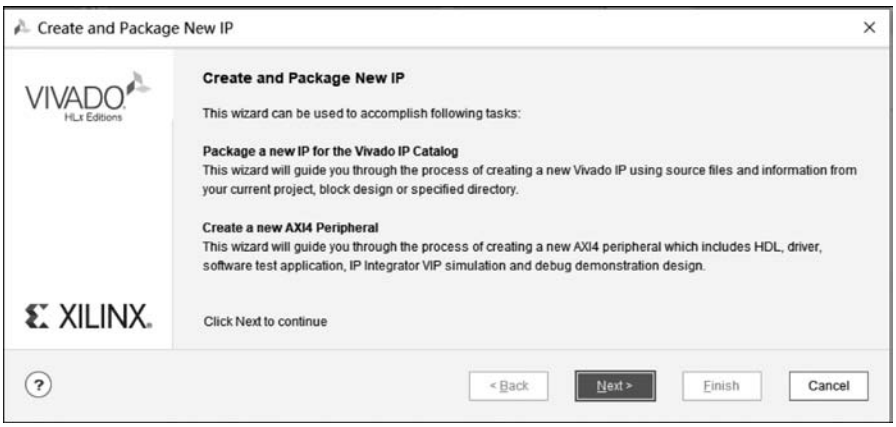

图 3.39　创建和封装新 IP 的启动界面

（3）单击 Next 按钮，弹出如图 3.40 所示的封装选项界面，选中 Packaging Options 下的 Package your current project 单选按钮，表示将当前的工程封装为 IP 核，单击 Next 按钮。

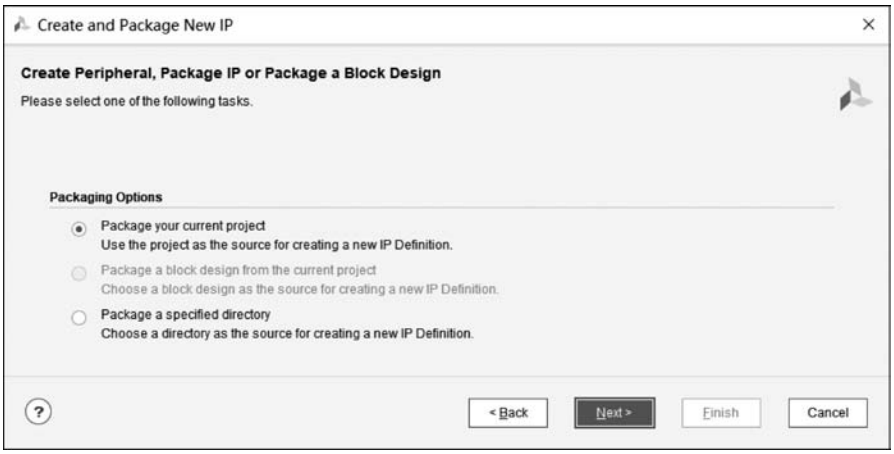

图 3.40　封装选项界面

(4) 如图 3.41 所示,此界面中的 IP location 指示 IP 核的路径,以便设计者到此路径下将 IP 核导入别的工程中,也可通过单击右侧带省略号的按钮来给 IP 核指定新的位置,单击 Next 按钮。

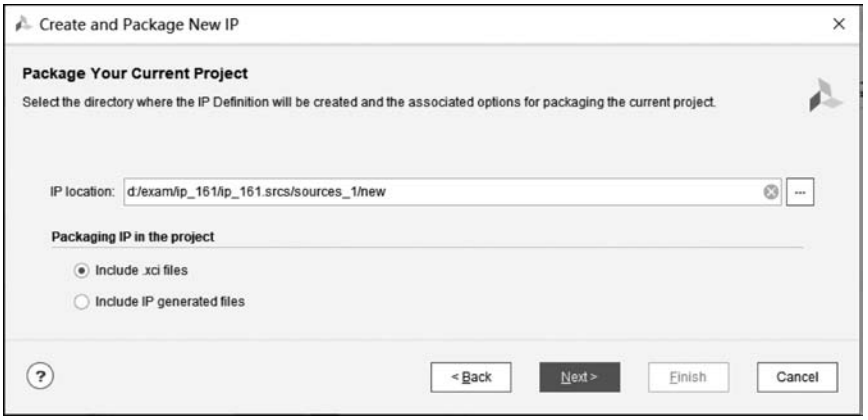

图 3.41 IP 核的路径

(5) 单击 Finish 按钮,完成 IP 核的创建,如图 3.42 所示。

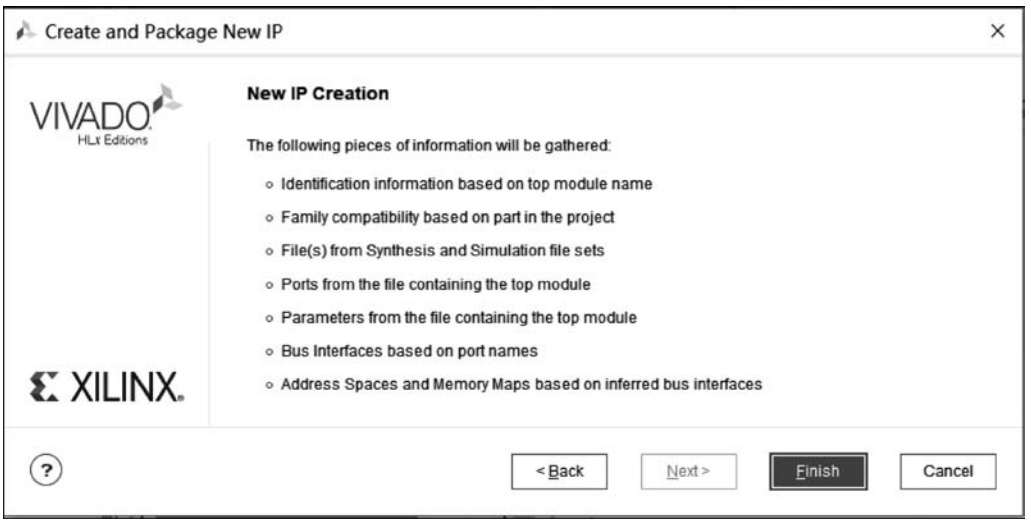

图 3.42 IP 核创建完成

4. 封装 IP 核

(1) 完成 IP 核的创建后,在 Vivado 主界面中,选择 Sources 窗格下的 Hierarchy 标签页,此时在 Design Sources 下方出现一个名为 IP-XACT 的图标,其下有一个 component.xml 的文件,其中保存了封装 IP 核的信息,如图 3.43 所示。

(2) 在 Vivado 主界面右侧窗格中的 Package IP 标签页下,单击 Identification 可查看并修改 IP 核的相关信息,如图 3.44 所示。

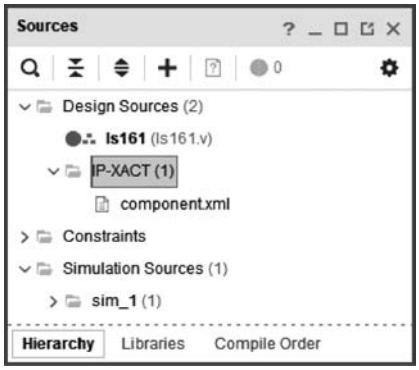

图 3.43　IP 核封装信息文件

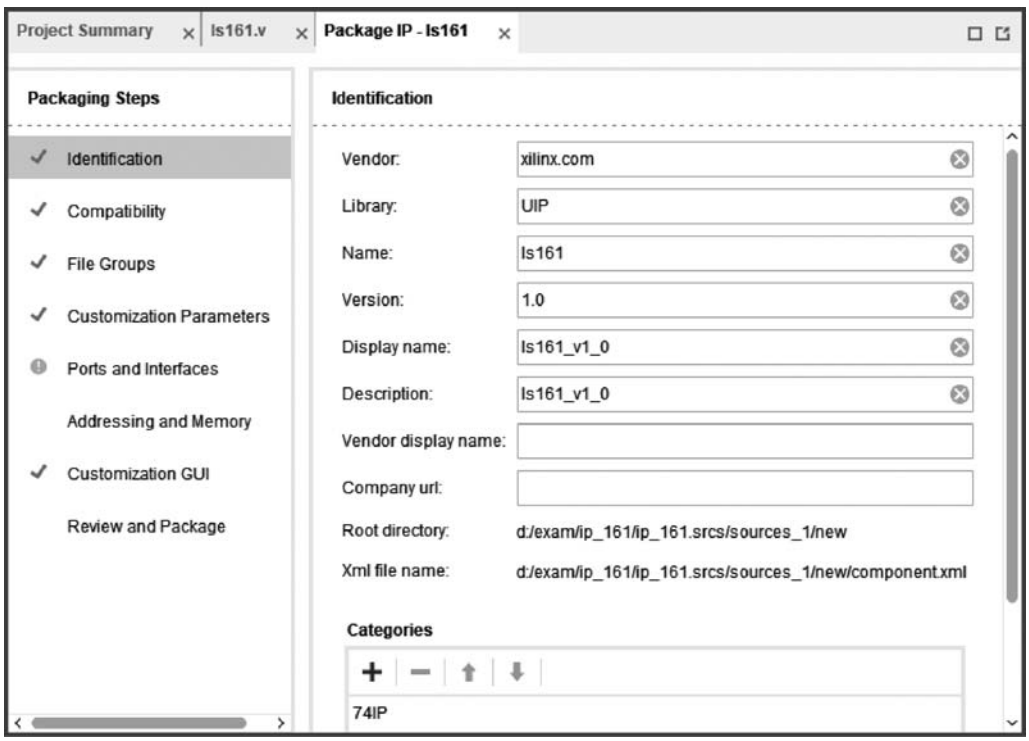

图 3.44　封装 IP 核的 Identification 界面

(3) Compatibility 界面显示 IP 核支持的 FPGA 系列，可以继续添加 IP 核支持的 FPGA 器件，单击右侧的加号，选择第一项 Add Family Explicitly...，如图 3.45 所示。

(4) 在弹出的 Add Family 对话框中可添加除已支持的 artix7(Artix-7)外的其他器件系列，如图 3.46 所示，勾选完毕单击 OK 按钮。

(5) 单击 Customization GUI 界面，在右侧可以预览 IP 核的信号接口，同时可以在 Component Name 文本框中修改 IP 核的名称，如图 3.47 所示。

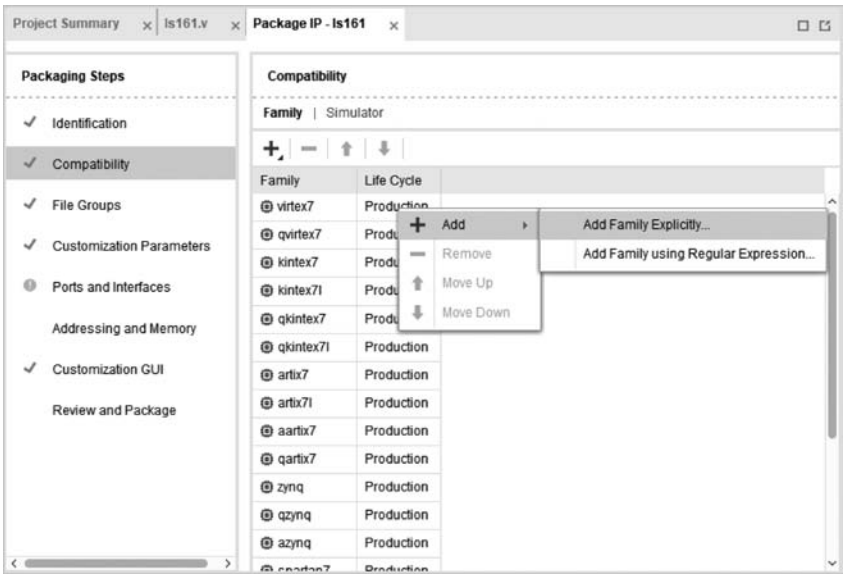

图 3.45　IP 核的 Compatibility 界面

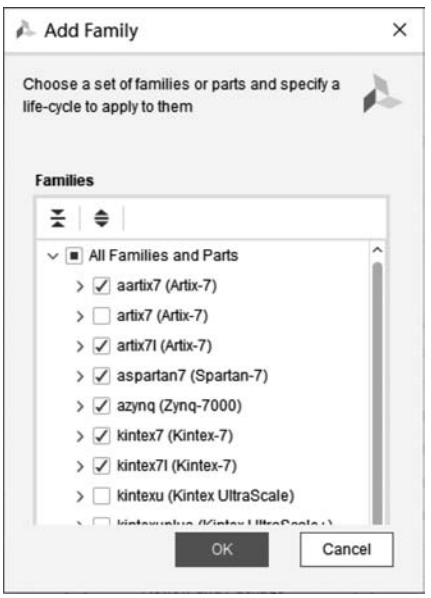

图 3.46　添加 IP 核支持的器件系列

（6）单击 Review and Package 界面，可查看 IP 核的最终信息，其中，Root directory 表示 IP 核的存储目录，信息确认无误后单击下方的 Package IP 按钮，完成 ls161 核的封装，如图 3.48 所示。

（7）回到 Vivado 主界面，选择 PROJECT MANAGER 中的 IP Catalog 选项，出现 IP Catalog 窗格，在其中的 User Repository 下可找到刚创建的 ls 161_v1_0，说明该 IP 核

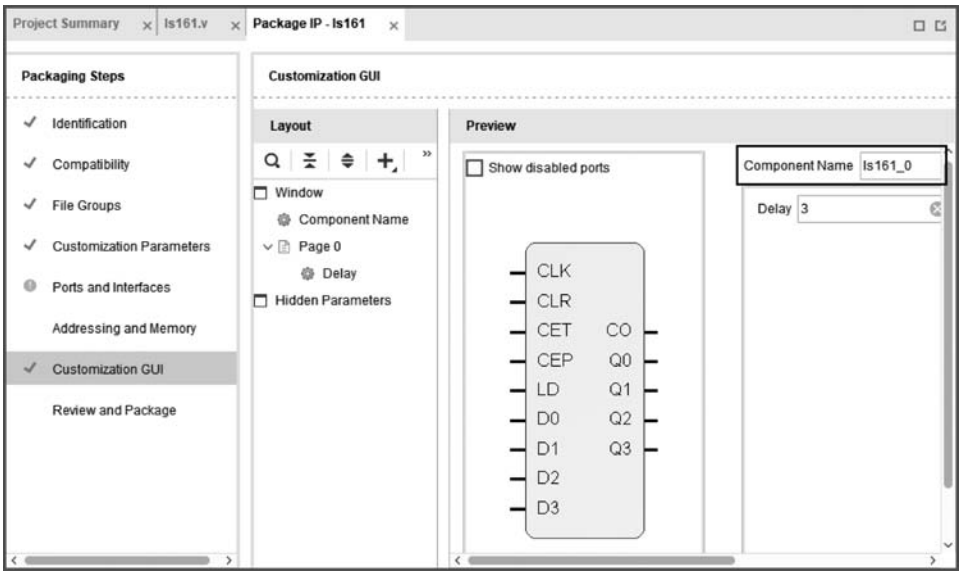

图 3.47　Customization GUI 界面

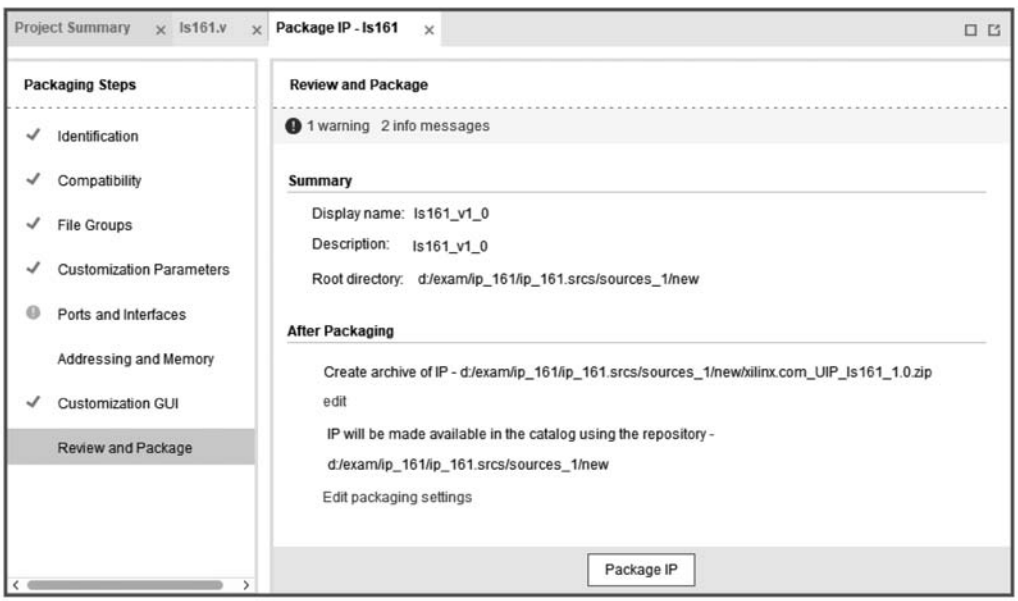

图 3.48　Review and Package 界面

已创建和封装成功,可以调用了,如图 3.49 所示。

5. 创建和封装另一 IP 核 74ls00

采用与上面 74ls161 核相同的步骤,创建和封装功能类似 74ls00(2 输入与非门)的 IP 核,以供调用。ls00 核的源代码如例 3.5 所示。

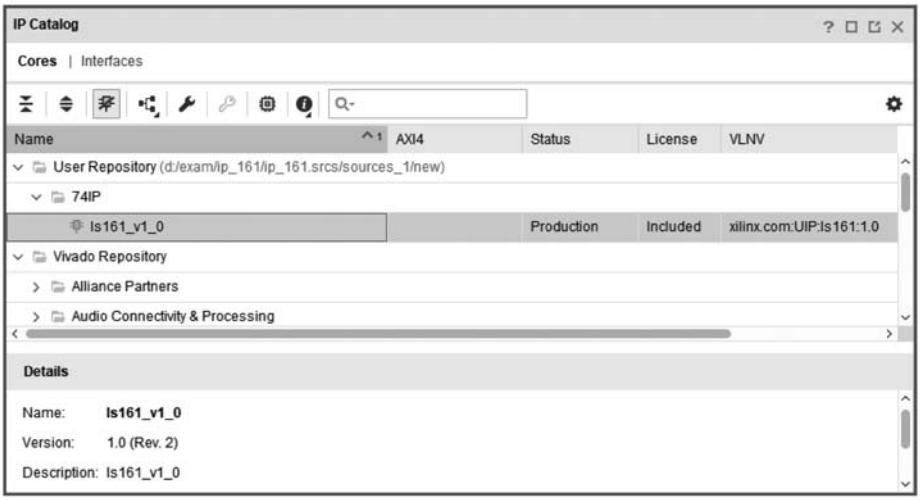

图 3.49　查看 IP 核

【例 3.5】 ls00 核(2 输入与非门)的源代码。

```
module ls00
#(parameter DELAY = 3)(
    input a,b,
    output y
    );

nand #DELAY (y,a,b);
endmodule
```

ls00 核创建后,对其进行封装,其中 Identification 界面信息如图 3.50 所示。

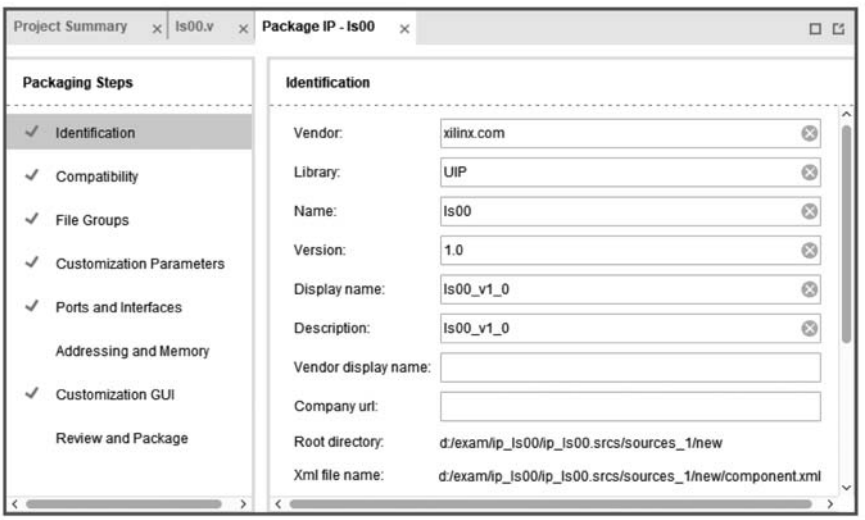

图 3.50　ls00 核的 Identification 界面

单击 Review and Package 界面，可查看 ls00 核的最终信息，信息确认无误后单击下方的 Package IP 按钮，完成 ls00 核的封装，如图 3.51 所示。

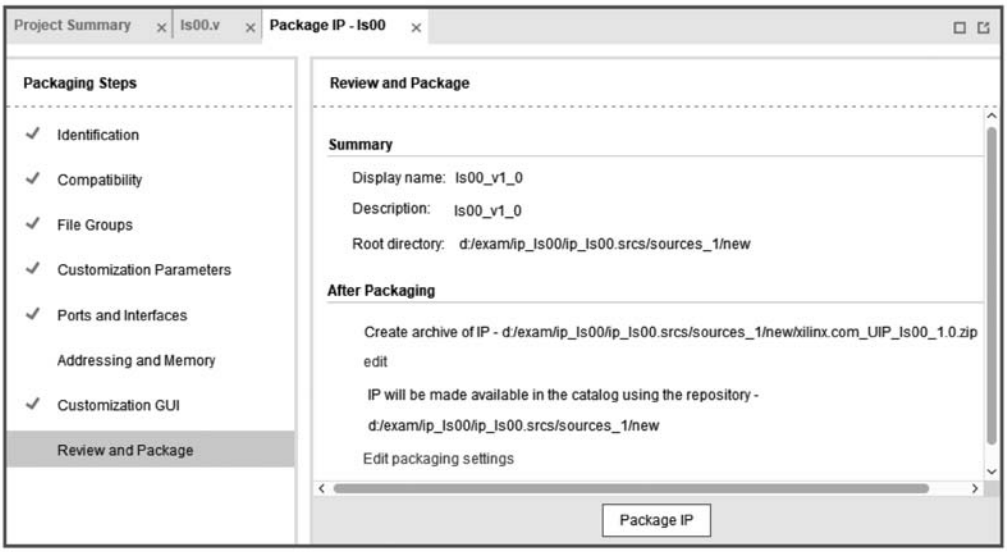

图 3.51　ls00 核的 Review and Package 界面

3.3　基于 IP 集成的计数器设计

本节利用 3.2 节创建和封装的 ls161 和 ls00 两个 IP 核，采用原理图设计的方式实现一个模 9 计数器，以说明基于 IP 集成的 Vivado 设计的流程。

1. 创建工程

启动 Vivado 2018.2，单击 Quick Start 栏中的 Create Project，启动工程向导，创建一个新工程，将其命名为 count_bd，存于 D:/exam/count_bd 文件夹中。此过程不再详述。

2. 添加 IP 核

(1) 将 3.2 节中生成的 IP 封装目录中的压缩包“xilinx.com_UIP_ls161_1.0.zip”和“xilinx. com_UIP_ls00_1.0.zip”复制到当前工程目录中，并解压到新建的 UIP 目录下，解压后的文件目录如图 3.52 所示。

(2) 在 Flow Navigator 中选择 PROJECT MANAGER 下的 Settings 选项，在弹出的 Settings 对话框的左侧选中 IP，单击 Repository，出现 Repository 标签页，单击＋，进入当前工程目录，选中 UIP 文件夹(其中放置 ls161、ls00 两个 IP 核封装文件)，单击 Select 按钮，在弹出的窗口中单击 OK 按钮，上述过程如图 3.53 所示。

(3) 如图 3.54 所示，D:/exam/count_bd/UIP 文件夹已出现在 IP Repositories 下，单击 Apply 按钮，再单击 OK 按钮。

图 3.52　将 IP 核文件夹放至 UIP 目录下

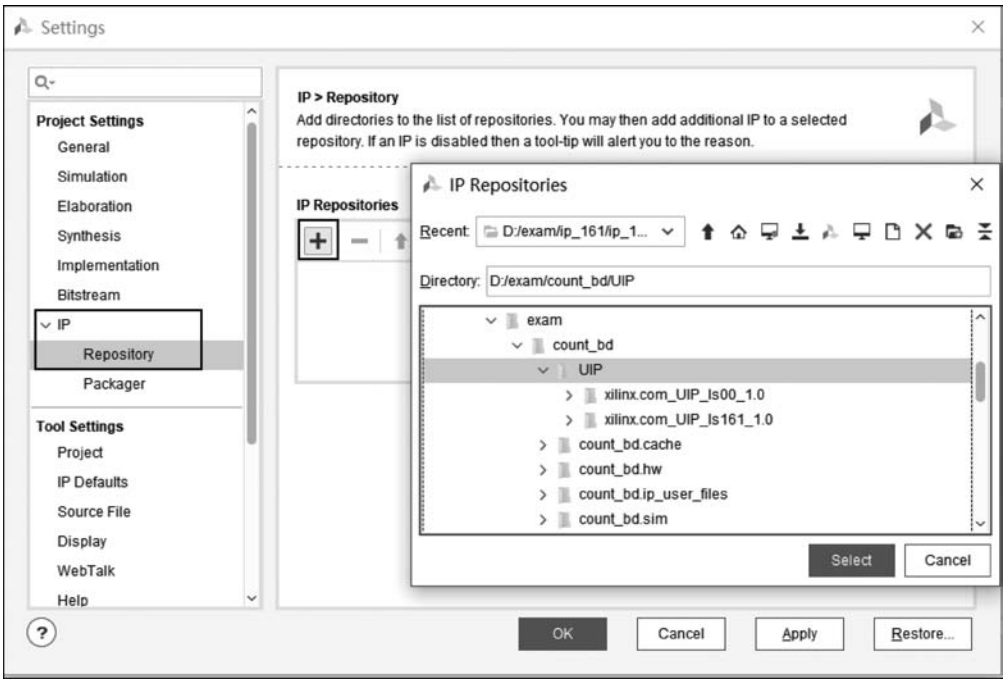

图 3.53　将 IP 核封装文件加入 IP 库

(4) 在 PROJECT MANAGER 下选择 IP Catalog 选项,在右侧的 IP Catalog 标签页中,展开 User Repository,可以看到用户自定义的 IP 核 ls161_v1_0 和 ls00_v1_0 已经出现在 IP 库中,可以调用了,如图 3.55 所示。

3. 基于 IP 集成的原理图设计

(1) 进入 Vivado 主界面,在左侧的 Flow Navigator 中选择 IP INTEGRATOR 下的 Create Block Design 选项,在弹出的 Create Block Design 对话框的 Design name 栏中输入设计名 count_bd,表示新建一个名为 count_bd 的原理图文件,如图 3.56 所示。

(2) 单击 OK 按钮,进入 BLOCK DESIGN 设计界面。在原理图中添加 IP 核,可采用如下方式:

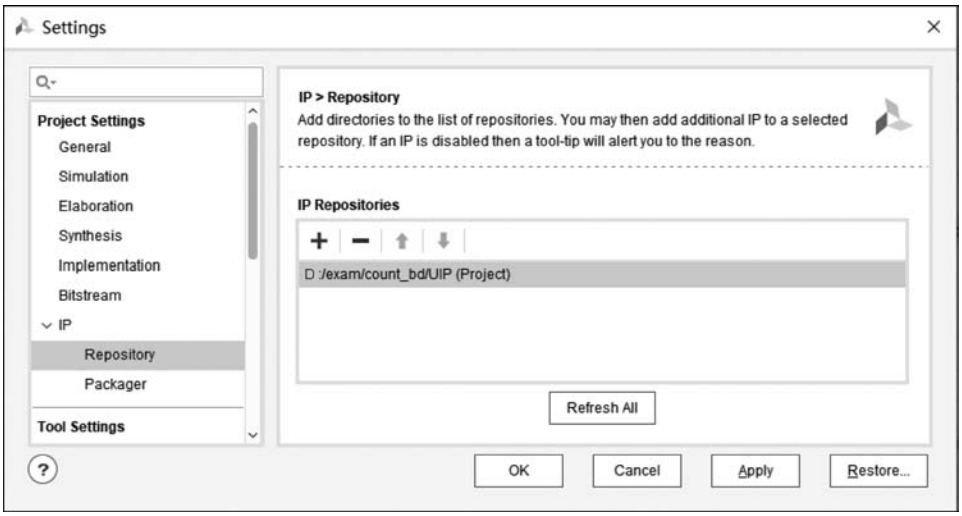

图 3.54　指定 IP 库

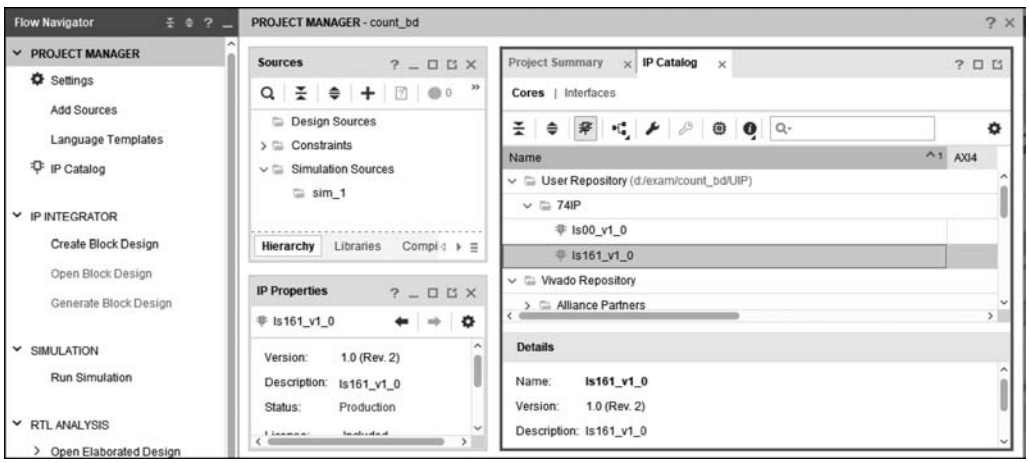

图 3.55　将 ls161 和 ls00 核添加到 IP 库

① 单击原理图中间区域的＋按钮。

② 在 Diagram 图形界面上侧工具栏中单击＋按钮。

③ 在原理图空白区域右击，在弹出的快捷菜单中选择 Add IP 命令。

在弹出窗口的 Search 搜索栏中输入 ls，在列表中选择 ls161_v1_0，如图 3.57 所示。

（3）双击 ls161_v1_0，或者按 Enter 键，将其添加到原理图中。采用同样的方法将 IP 核 ls00_v1_0 也调入原理图中，左击选中 ls00_v1_0 模块，右击，选择菜单 Orientation 中的 Rotate Clockwise 命令，连续执行两次，使其旋转 180°，并将其移动到原理图上合适的位置，如图 3.58 所示。

（4）连线：将鼠标指针移至 ls161 模块的 Q0 接口处，待其变成铅笔形状后，按下鼠标左键并拖曳到 ls00 模块的 a 接口处，释放鼠标左键后可看到两个接口信号已被连接起

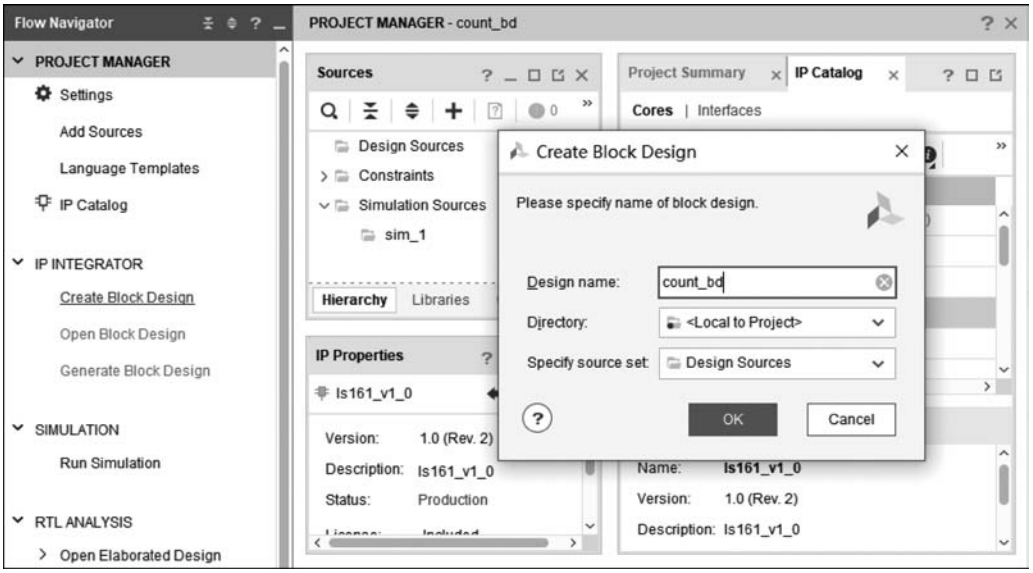

图 3.56　新建原理图并输入文件名

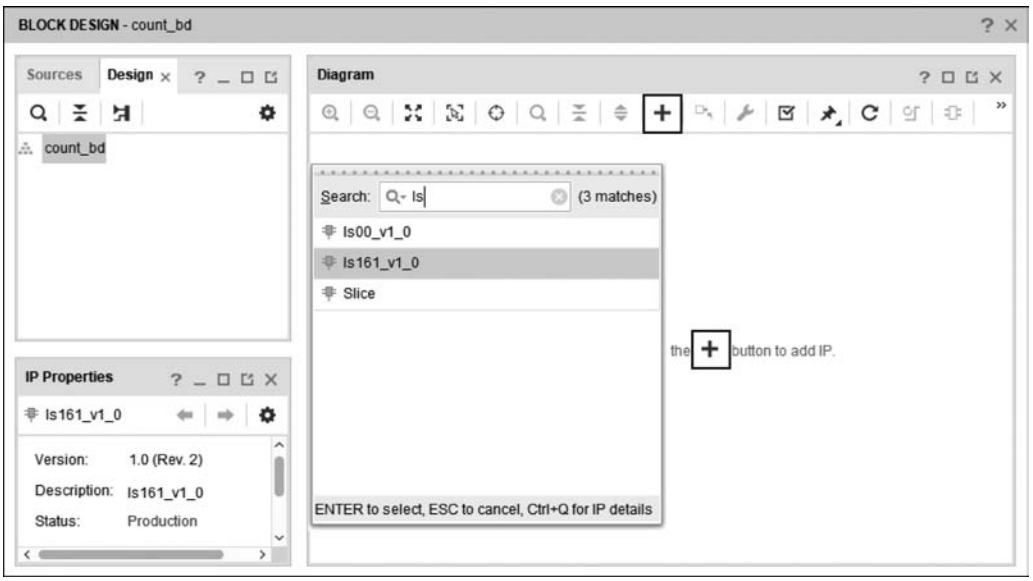

图 3.57　在原理图中添加 IP 核

来。采用同样的方式进行其他连线。

(5) 创建端口。创建端口有以下两种方式。

① 在原理图空白处右击,在弹出的快捷菜单中选择 Create Port...命令,在弹出的 Create Port 对话框中设置端口的名称、方向和类型。图 3.59 所示是创建了一个名为 PT 的输入端口;图 3.60 所示是创建了一个名为 Q0 的输出端口。

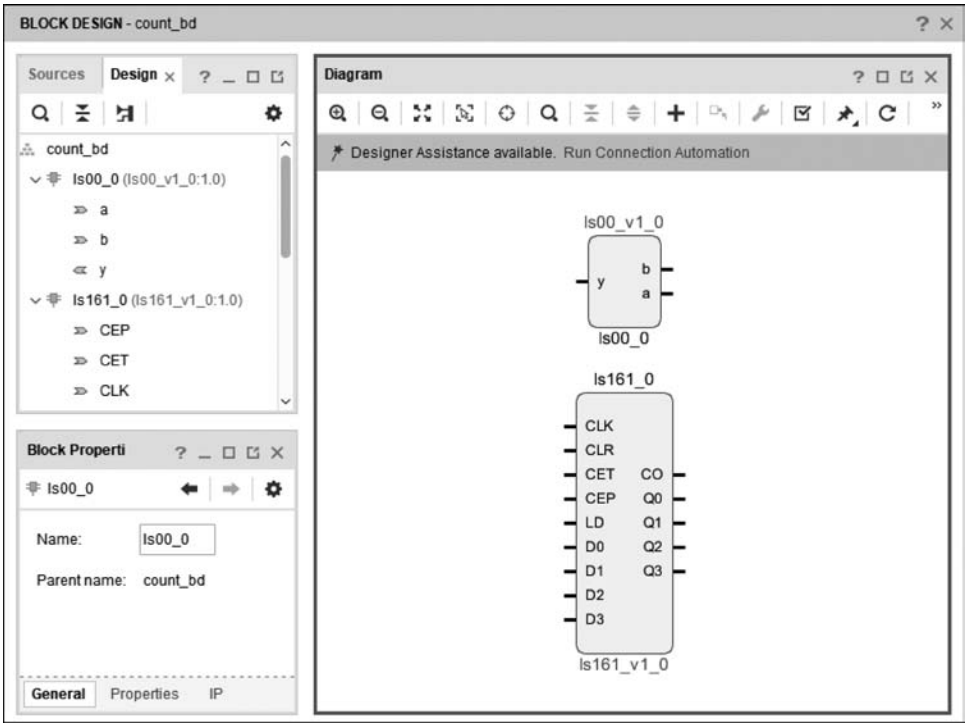

图 3.58　添加 IP 核并合理布局

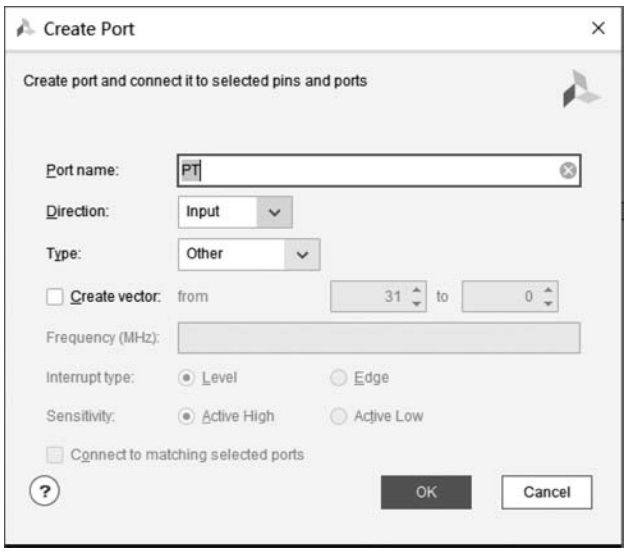

图 3.59　创建一个名为 PT 的输入端口

② 单击选中模块的某一引脚，右击，在弹出的快捷菜单中选择 Make External 命令，可自动创建与引脚同名、同方向的端口。

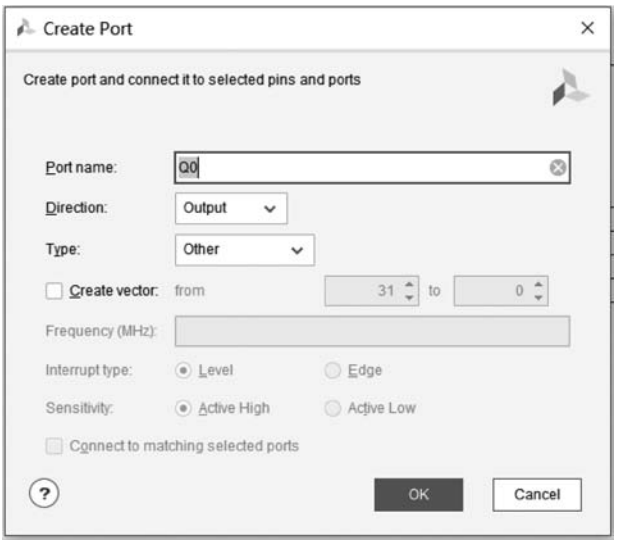

图 3.60　创建一个名为 Q0 的输出端口

(6) 连线完成后的原理图如图 3.61 所示，单击原理图工具栏中的 Regenerate Layout，自动对模块和连线进行优化布局。执行 Regenerate Layout 后的原理图如图 3.62 所示。在完成后将对原理图存盘。

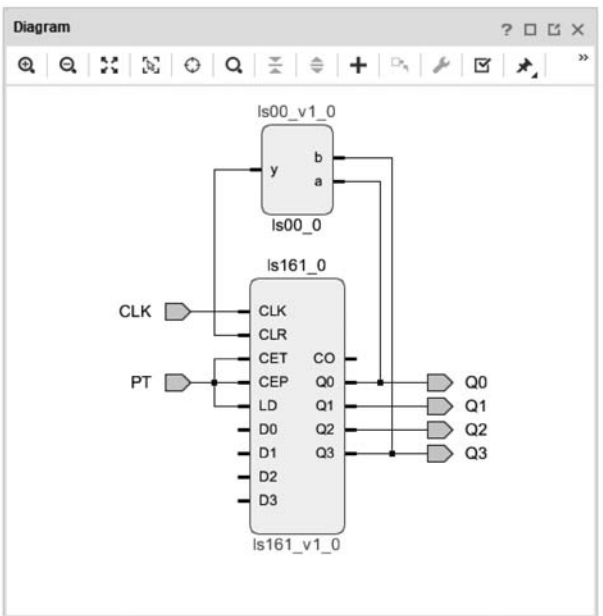

图 3.61　完成后的原理图

(7) 完成原理图后，生成顶层文件。

① 在 BLOCK DESIGN 对话框的 Hierarchy 标签页中，在 Design Sources 下的

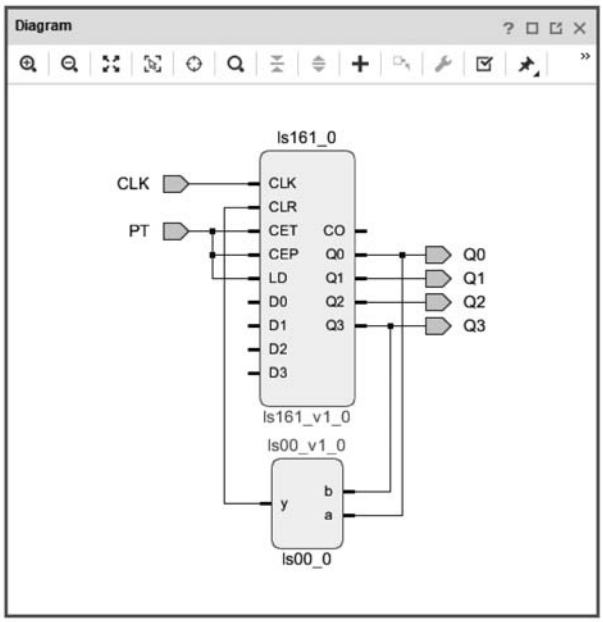

图 3.62　执行 Regenerate Layout 后的原理图

count_bd.bd 图标上右击，在弹出的快捷菜单中选择 Generate Output Products 命令，如图 3.63 所示。

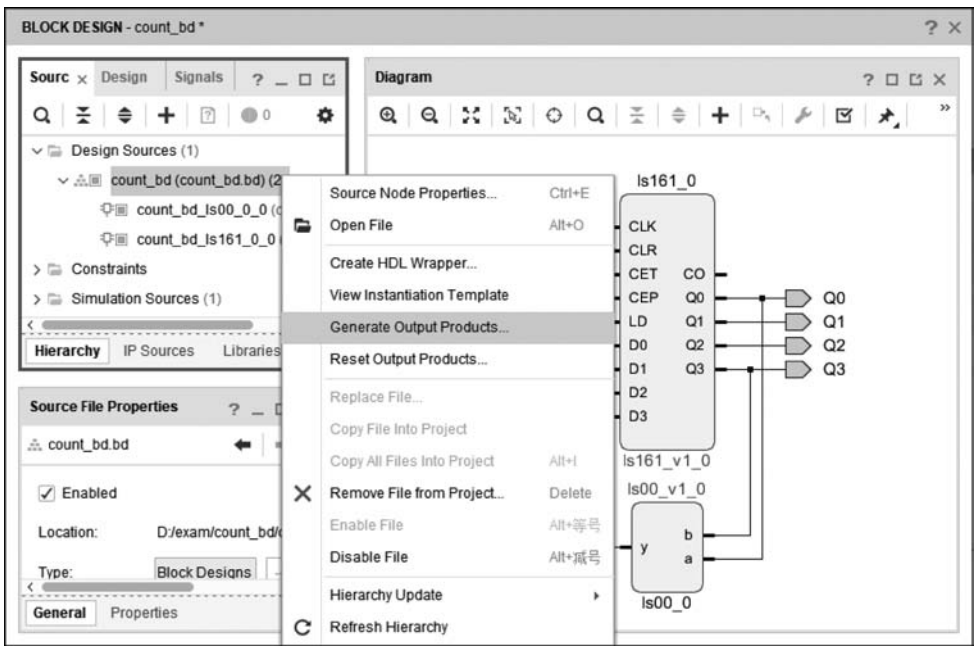

图 3.63　选择 Generate Output Products 命令

② 弹出 Generate Output Products 对话框，如图 3.64 所示。Synthesis Options 选项组中有如下选项。

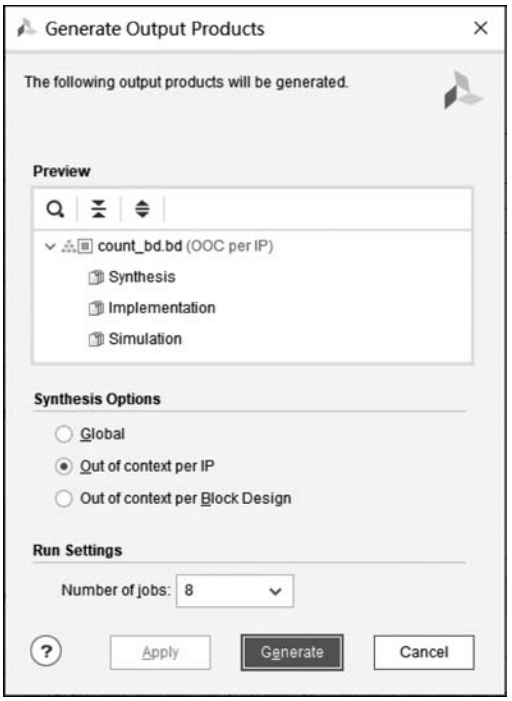

图 3.64　Generate Output Products 对话框

Global：表示全局综合，选择此选项，则 IP 核生成的文件将会和其他文件一起进行综合，即每一次设计文件被修改后，IP 核文件都会跟着一起综合一遍。

Out of context per IP：即 OOC 选项，此选项是 Vivado 的默认选项，选择此选项，Vivado 将会把生成的 IP 核当作一个单独的模块来综合，生成.dcp(design checkpoint)文件；在运行实现(Implementation)时，Vivado 会将 OOC 模块的综合网表插入到顶层网表中，从而完成设计。此选项还会生成一个以 stub 为扩展名的存根文件，类似黑盒子，即文件中只有输入输出端口。

Out of context per Block Design(此处略)。

本例选择 Out for context per IP 单选按钮，然后单击 Generate 按钮，如图 3.64 所示，完成后单击 OK 按钮。

③ 输出文件生成后，再次在 BLOCK DESIGN 对话框中的 count_bd.bd 图标上右击，在弹出的快捷菜单中选择 Create HDL Wrapper 命令，如图 3.65 所示。

④ 在弹出的 Create HDL Wrapper 对话框中选中 Let Vivado manage wrapper and auto-update 单选按钮，单击 OK 按钮，如图 3.66 所示。

至此，已完成原理图设计。从图 3.67 可看到原理图源文件层次结构图，在 Design Sources 的 count_bd.bd 图标之上已生成 count_bd_wrapper.v 顶层文件。

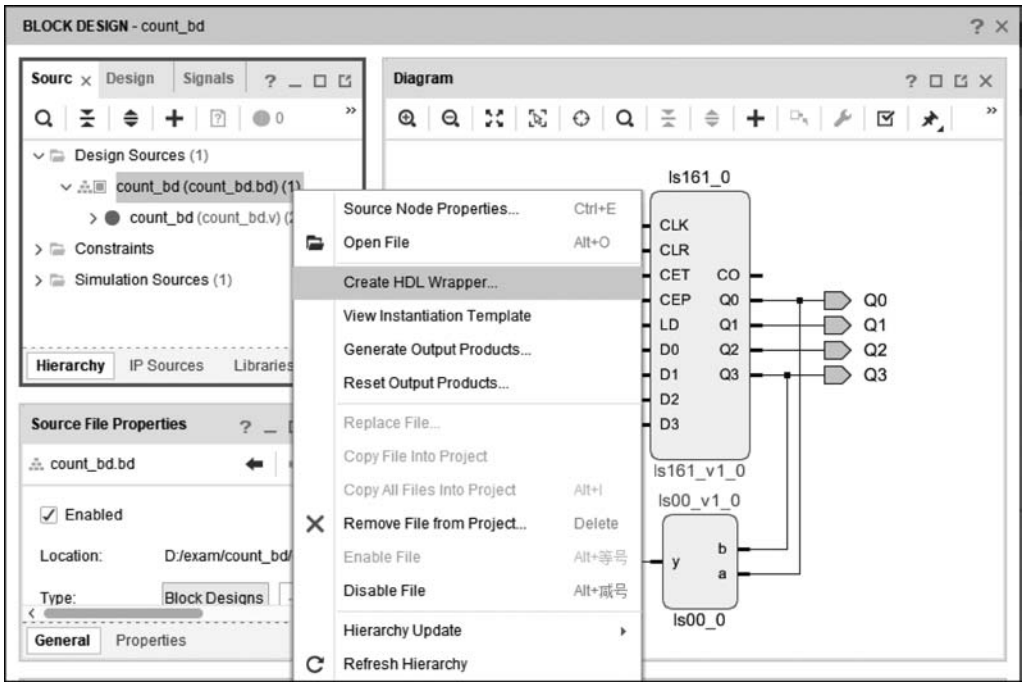

图 3.65　选择 Create HDL Wrapper 命令

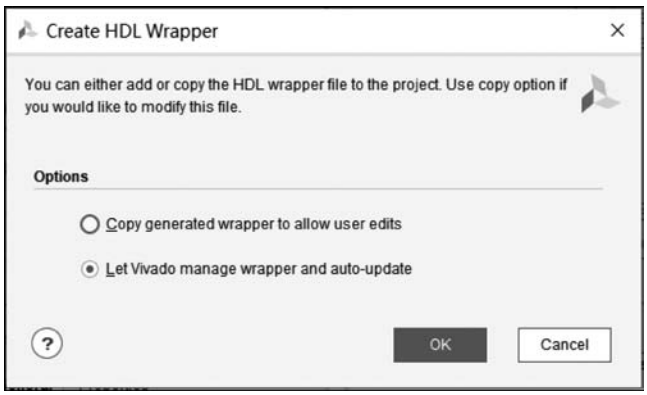

图 3.66　Create HDL Wrapper 对话框

4. 添加引脚约束文件

添加引脚约束有两种方法：第一种是利用 Vivado 中的 I/O Planning 功能；第二种是直接新建 XDC 约束文件。3.2 节中采用了方法二，本例采用方法一来完成此任务。

(1) 在 Vivado 主界面中，在 Flow Navigator 中选择 SYNTHESIS 下的 Run Synthesis 选项，单击 OK 按钮，完成后在弹出的 Synthesis Completed 对话框中选中 Open Synthesized Design 单选按钮，并单击 OK 按钮，如图 3.68 所示。

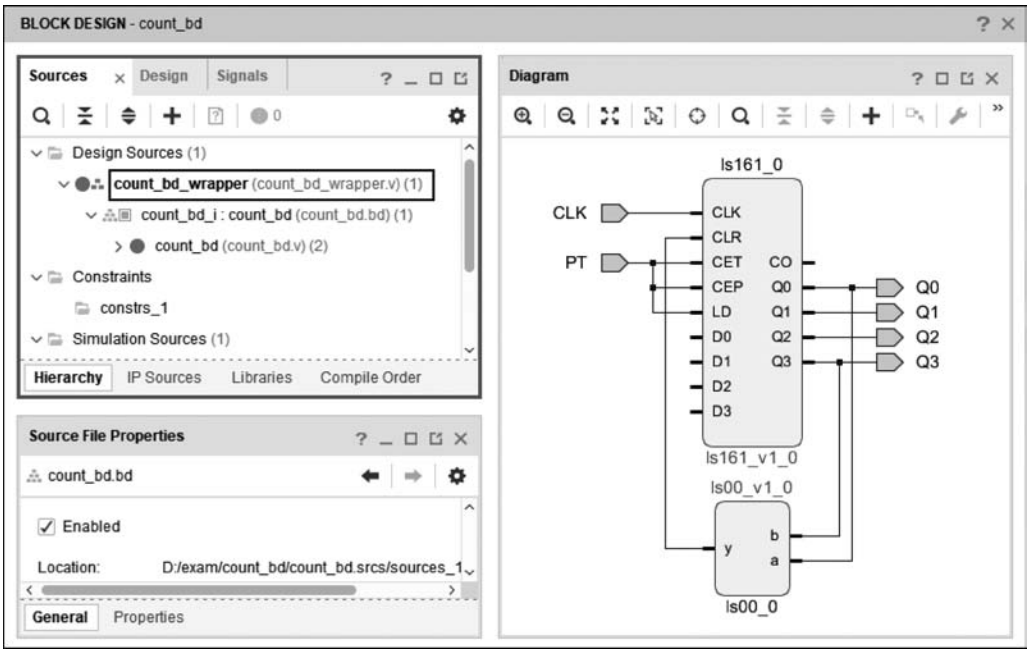

图 3.67　原理图源文件层次结构图

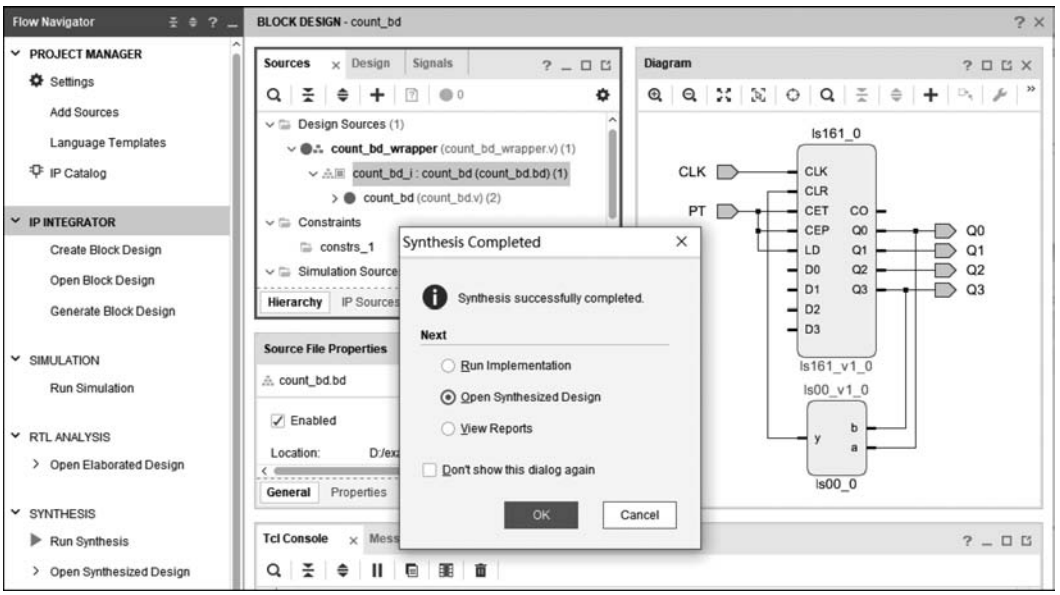

图 3.68　选中 Open Synthesized Design 单选按钮

（2）如图 3.69 所示，选择菜单 Window 中的 I/O Ports 命令，使 I/O Ports 标签页出现在主窗口下方。

（3）在 I/O Ports 标签页中对输入输出端口添加引脚约束，首先在 Package Pin 栏中

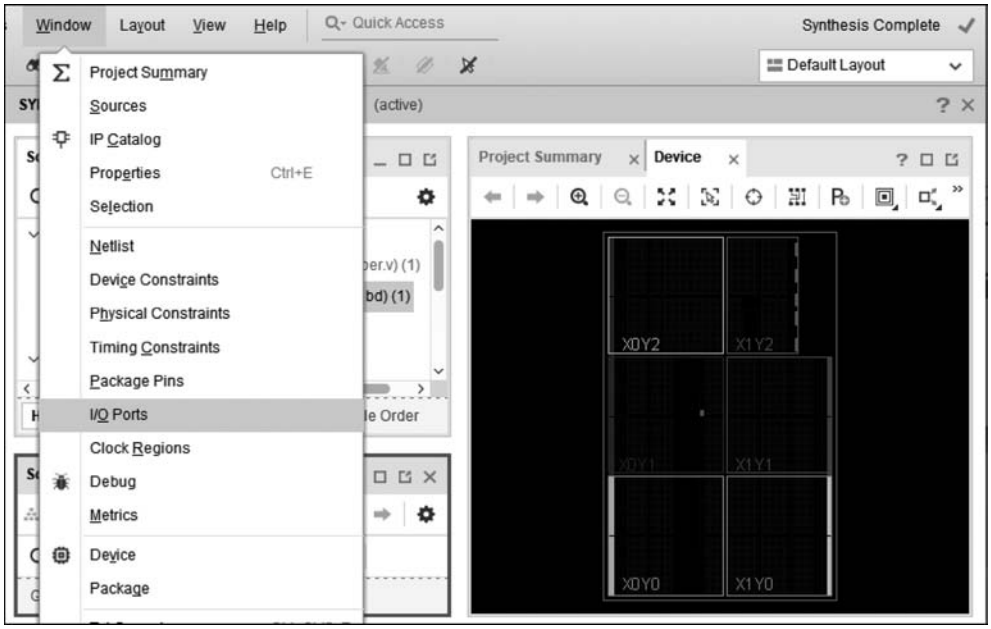

图 3.69　使能 I/O Ports 标签页

输入各端口对应的 FPGA 芯片的引脚号(对应关系可查看目标板说明文档或原理图),本例的 Q0～Q3 锁至 EGO1 开发板的 4 个 LED 灯,CLK 锁至按键 S1,PT 锁至拨码开关 SW0;然后在 I/O Std 栏中通过下拉菜单选择 LVCMOS33,将所有信号的电平标准设置为 3.3V,如图 3.70 所示。

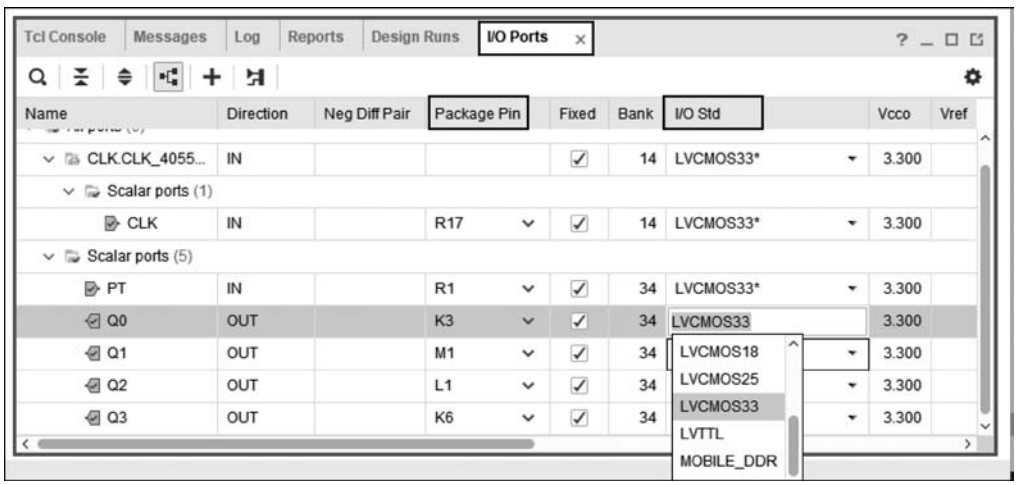

图 3.70　对输入输出端口添加引脚约束

(4) 引脚约束完成后单击保存按钮,如图 3.71 所示,可看到 Sources 标签页中已出现引脚约束文件 count_bd_wrapper.xdc,双击该文件,其内容如图中所示。

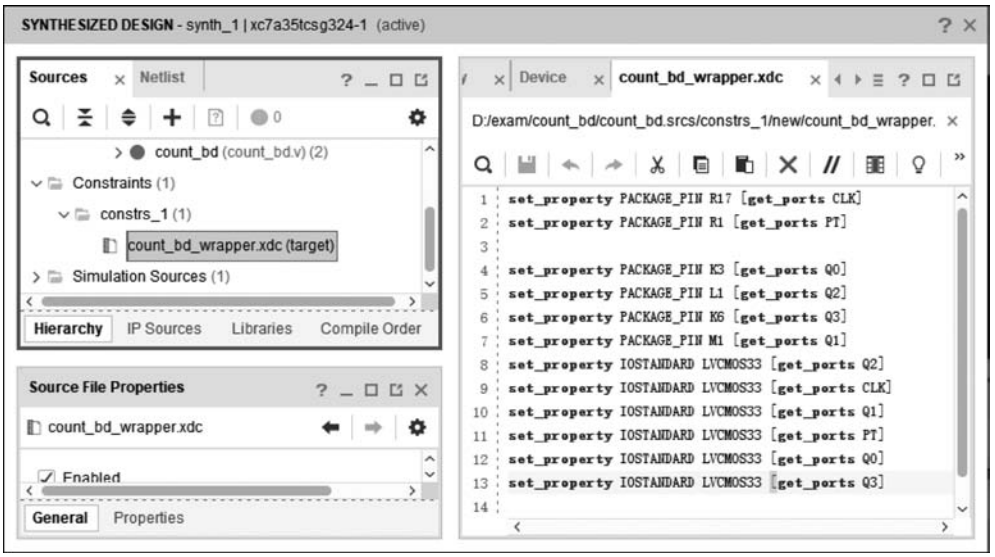

图 3.71　引脚约束文件

5. 生成比特流文件并下载

(1) 在 Vivado 主界面的 Flow Navigator 中选项 PROGRAM AND DEBUG 下选择 Generate Bitstream 选项，此时会弹出 No Implementation Results Available 对话框，提示工程还没有经过 Run Implementation 等过程，如图 3.72 所示，单击 Yes 按钮，再单击 OK 按钮，软件会自动执行 Run Implementation 并生成比特流文件。

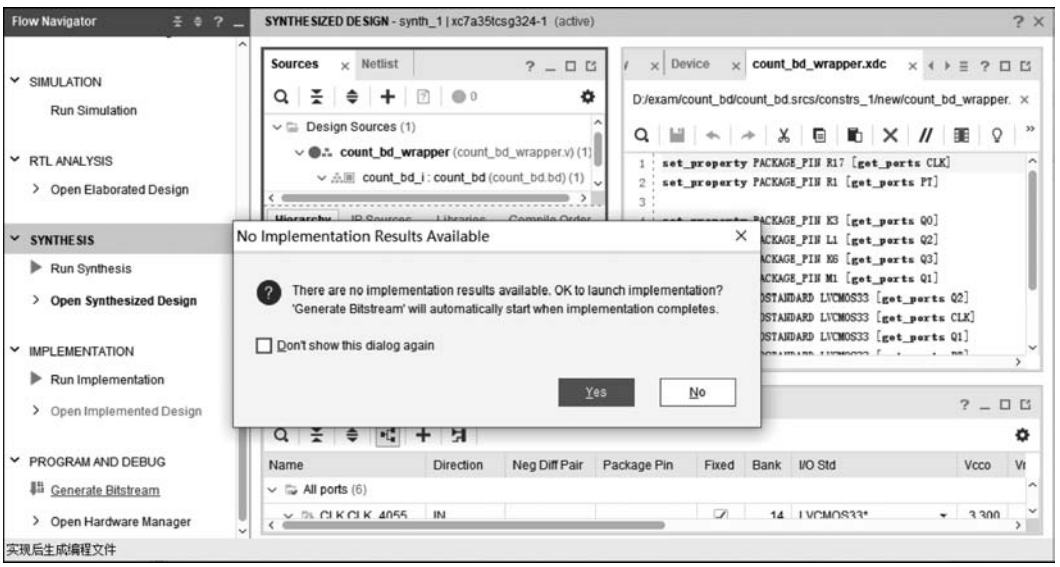

图 3.72　生成比特流文件

(2) 生成比特流文件后,选择 Open Hardware Manager 并单击 OK 按钮,用 Micro USB 线连接计算机与板卡,并打开电源开关。在 HARDWARE MANAGER 界面单击 Open Target,选择 Auto Connect,连接成功后,在目标芯片上右击,选择 Program device,在弹出的 Program Device 对话框中单击 Program 按钮,对 FPGA 芯片进行编程,上述过程如图 3.73 所示。

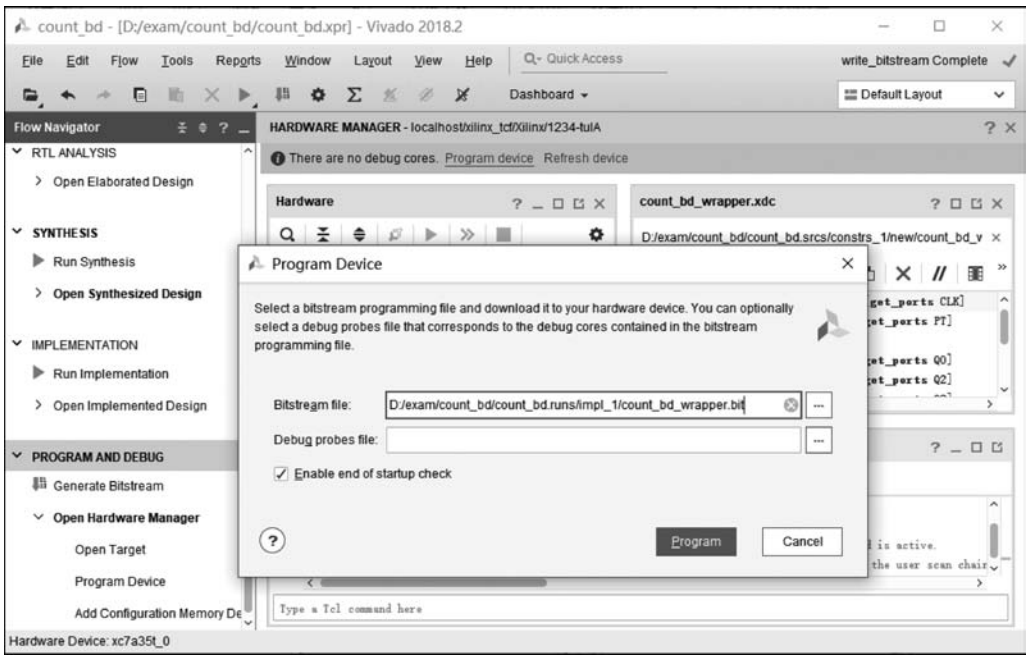

图 3.73 对 FPGA 芯片进行编程

(3) 下载完成后,观察开发板实际运行效果。

3.4 Vivado 的综合策略与优化设置

1. 设计的调试和可视化

在 Vivado 设计流程中,一般可查看三个阶段的网络表(如图 3.74 所示),实现设计的可视化:

- 详细设计(Elaborated Design)。
- 综合后的设计(Synthesized Design)。
- 实现后的设计(Implemented Design)。

所谓的网络表(Netlist),是对设计的一种描述,用部件、端口和连线等来表达设计。

(1) 选择 Open Elaborated Design(打开详细设计),该网表是在综合之前表述设计,一般由复用器、加法器、比较器、寄存器组等较大的部件来表述设计,也称为 RTL 级设计。

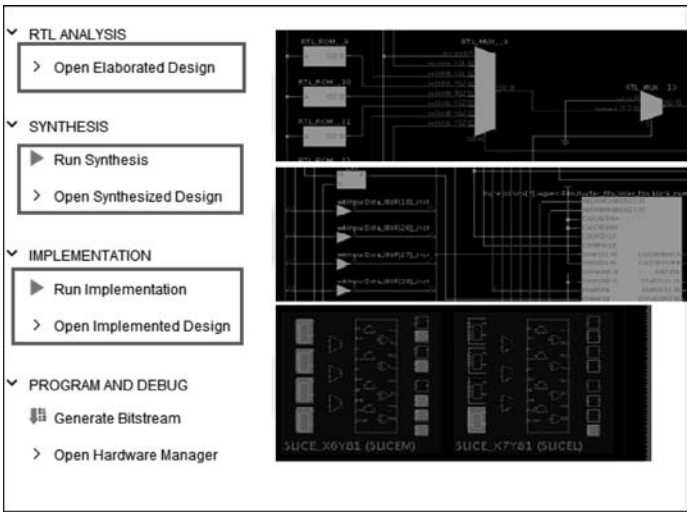

图 3.74　查看不同设计阶段的网络表

(2) 选择 Open Synthesized Design(打开综合设计),该网表是在综合之后表达设计,由 LUT、缓冲器、触发器、进位链等基础元件(BEL)构成的网络表来实现设计。

(3) 选择 Open Implemented Design(打开实现设计),该网表是在布局布线之后具体实现设计的网表。

2. Language Template(语言模板)

Vivado 提供多种语言模板代码供用户参考,选择菜单 Tools 中的 Language Templates 命令,便会出现 Language Templates 对话框,如图 3.75 所示。从图中可以看出,Vivado 提供 Verilog、VHDL、SystemVerilog 语言模板,还提供 XDC 约束文件模板和 Debug 调试模板,选择 Verilog 模板,可看到包括可综合模板、激励代码模板和 IP 集成器模板等。

模板代码实现的设计不仅规范,而且更优化,可有效节省 FPGA 资源。图 3.76 所示是一个固定深度的移位寄存器(SRL)LUT 的可综合模板的参考代码。

建议在设计时尽可能参考语言模板。

3. 综合设置选项

在 Vivado 主界面的 Flow Navigator 中,选择 PROJECT MANAGER 下的 Settings 选项,在弹出的 Settings 对话框中选中 Synthesis 标签页,如图 3.77 所示。以下介绍 Synthesis 标签页中的各项设置。

(1) Constraints 栏:选择用于综合的约束集。约束集是一组 XDC 约束文件,默认选择 active 约束集。约束集包括时序约束和物理约束。

- 时序约束(Timing Constraints):定义设计的时序需求。如果没有时序约束,Vivado 会根据布线长度和布局拥挤度优化设计。

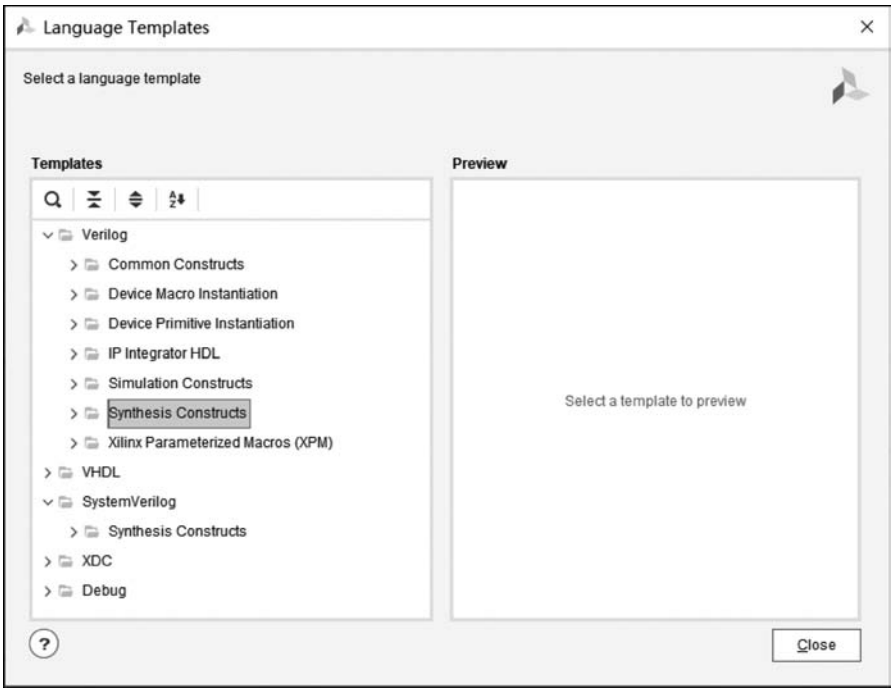

图 3.75　Language Templates 对话框

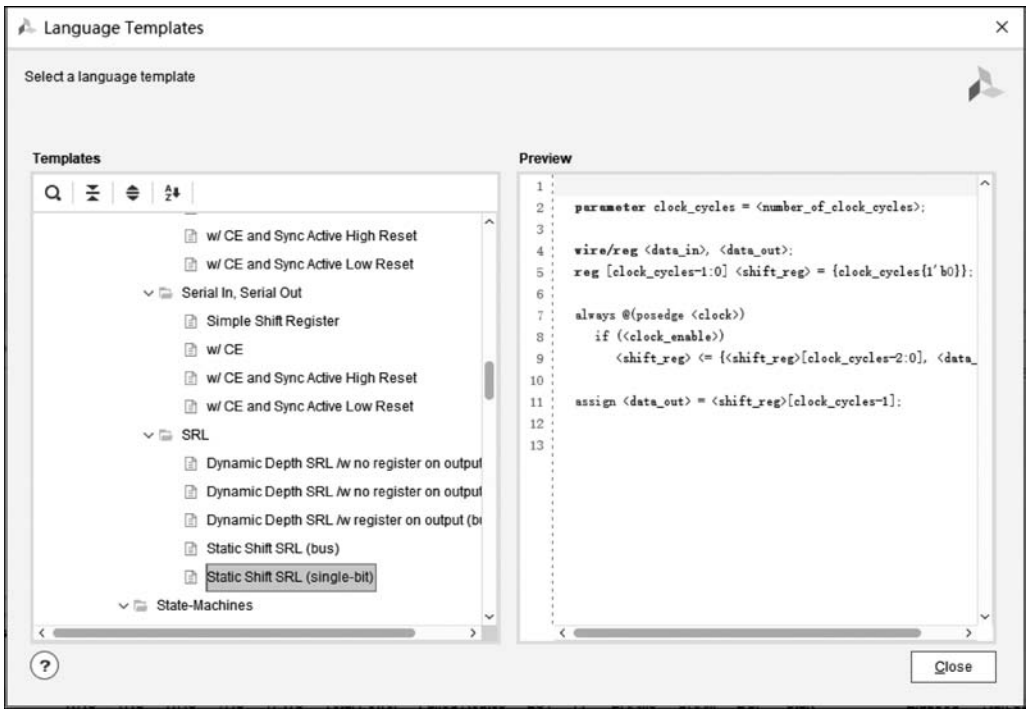

图 3.76　固定深度的 SRL 可综合模板代码

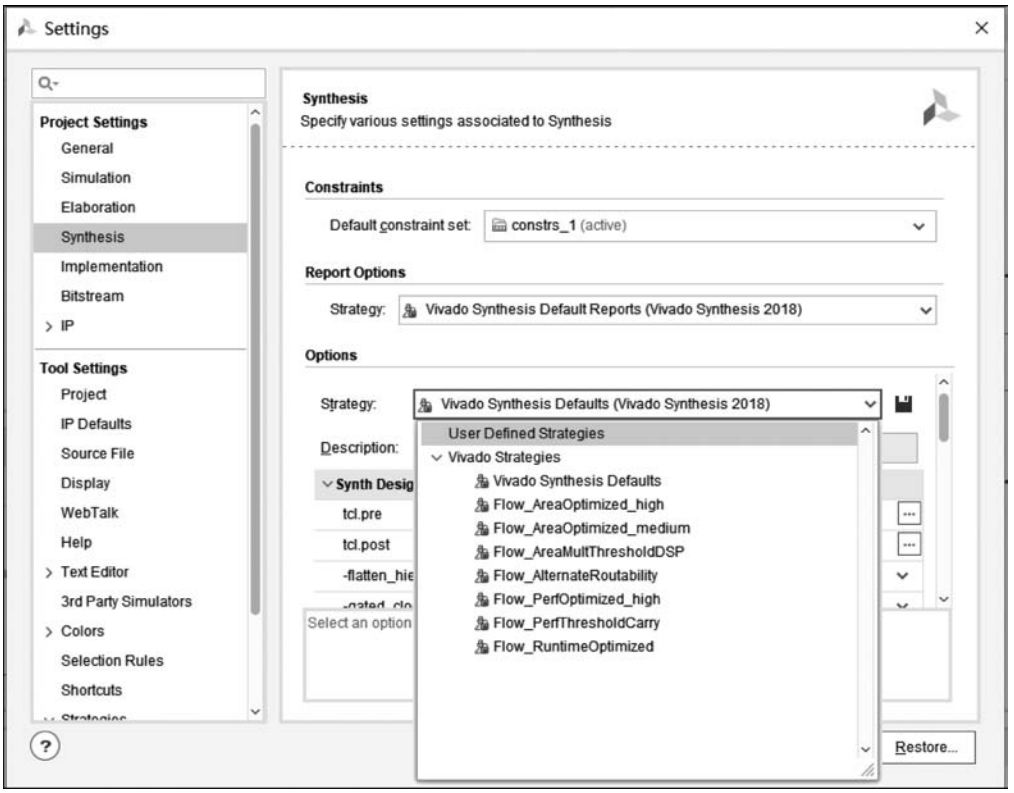

图 3.77　Synthesis 设置页面

- 物理约束(Physical Constraints)：包括引脚约束，物理单元(如块 RAM、查找表、触发器)布局的位置等。

(2) Options 栏：用于选择综合运行时使用的策略(Strategy)。Vivado 提供了几种预定义的策略，也可以自定义策略，这需要对策略中每个选项单独设置，这些选项如图 3.78 所示。以下给出各选项的含义。

-flatten_hierarchy：定义综合工具如何控制层次结构，选择将所有层次融为一体进行综合，还是独立综合再连接到一起。具体选项如下。

- none：不展开层次结构。
- full：全部展开层次结构。
- rebuilt：让综合工具展开层次结构，综合之后再重建层次结构。

-gated_clock_conversion：门控时钟转换使能。设计中应避免使用门控时钟，时钟信号应尽可能由混合模式时钟管理器(Mixed-Mode Clock Manager，MMCM)或者锁相环(Phase-Locked Loop，PLL)产生。

-fanout_limit：信号最大驱动负载数量，如果超出了该数值，会复制一个相同的信号来驱动超出的负载。

-directive：设置 Vivado 综合的优化策略，包括以下选项。

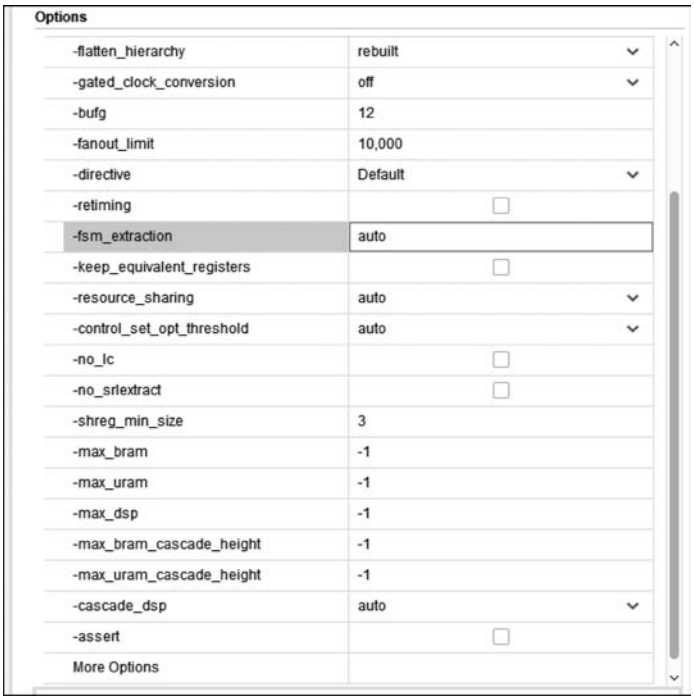

图 3.78　综合策略中的 Options 设置选项

- AreaOptimized_high：面积最省选项。
- AreaOptimized_medium：面积优化中等，在面积和速度间平衡处理。
- AreaMultThresholdDSP：更多地使用 DSP 模块资源实现设计。
- AlternateRoutability：提高布线能力，减少 MUXF 和 CARRY 的使用。
- PerfOptimized_high：性能最优，即速度最快，但耗用的资源会高一些。
- RuntimeOptimized：综合运行时间最短，会忽略一些优化以减少综合运行时间。

-retiming：启用该功能，可通过在逻辑门和 LUT 之间的移动寄存器，降低最大路径时延，以提高电路时序性能。

-fsm_extraction：设定状态机的编码方式，默认值为 auto。

- one-hot：一位热码编码方式。
- gray：格雷编码。
- johnson：约翰逊编码。
- sequential：顺序编码。
- auto：此时 Vivado 会自动推断最佳的编码方式。

注意：-fsm_extraction 设定的编码方式优先级高于 HDL 代码中自定义的编码方式。

-keep_equivalent_registers：保留或合并等效寄存器。勾选该选项，等效寄存器保留。

-resource_sharing：资源共享。

-no_lc：勾选该选项，表示不允许 LUT 整合。当两个或多个逻辑函数的输入变量总数不超过 6 时，这些函数均可放置在一个 LUT 中实现，称为 LUT 整合。LUT 整合可以降低 LUT 的资源消耗，但也可能导致布线拥塞。因此，Xilinx 建议，当整合的 LUT 超过 LUT 总量的 15%时，应考虑勾选-no_lc，关掉 LUT 整合。

-no_srlextract：勾选该选项时，SRL 会用 FPGA 内的触发器实现，而不用 LUT 资源去实现（Xilinx 的 LUT 可用于实现 SRL）。

-shreg_min_size：当 SRL 的深度小于或等于-shreg_min_size 时，其实现方式采用触发器级联的方式；当其深度大于-shreg_min_size 时，实现方式则为“触发器＋LUT＋触发器”的形式。上面的-no_srlextract 选项，如果勾选，则是阻止将移位寄存器用 LUT 实现，其优先级高于-shreg_min_size。

-max_bram：块 RAM（BRAM）的最大使用数量。默认值为－1，表示允许使用 FPGA 中所有的块 RAM。

-max_uram：设置 UltraRAM 最大使用数量（对于 UltraScale 架构 FPGA 而言）。默认值为－1，表示允许使用所有的 UltraRAM。

-max_dsp：设置 DSP 模块的最大使用数量。默认值为－1，表示允许使用该 FPGA 中所有的 DSP 模块。

-max_bram_cascade_height：设置可以将 BRAM 级联在一起的最大数量。

-max_uram_cascade_height：设置可以将 UltraRAM 级联在一起的最大数量。

-assert：将 VHDL 中的 assert 状态纳入评估。

（3）自定义综合策略：除了 Vivado 提供的配置好的综合策略，还可以自定义综合策略。在 Settings 对话框中设置好各选项后，单击 Options 栏右侧的存盘按钮，弹出 Save Strategy As 窗口，在其中填写名称和描述，即可保存为用户自定义的综合策略，单击 Apply 按钮，综合策略列表中就会出现自定义的策略，可在后面的综合中使用该策略。

在 Settings 对话框的 Tool Settings 中，选择 Strategies 下的 Run Strategies 选项，在右侧的页面中也可以设置综合策略，单击＋按钮可新建策略，如图 3.79 所示。如果想在已有策略的基础上修改，可选中一个策略，单击上方的 Copy Strategy 按钮，User Defined Strategies 中会出现该策略的备份以供修改（Vivado 提供的策略是不能修改的）。

4. 控制文件编译次序

Vivado 可以自动识别和设置顶层模块，同时自动更新编译次序。

如图 3.80 所示，在 Vivado 主界面的 Sources 窗格中右击 Design Sources，在弹出的 Hierarchy Update 级联菜单中选择 Automatic Update and Compile Order 选项，则设定当源文件发生改动时，Vivado 会自动管理层次结构和编译次序，Hierarchy 标签页中会自动调整各模块的层级，Compile Order 标签页中将显示编译顺序。

如果选择 Automatic Update 中的 Manual Compile Order 选项，则表示 Vivado 自动调整各模块层级，但允许人工设定编译顺序，在 Compile Order 标签页中拖曳文件所处位置即可完成设定。

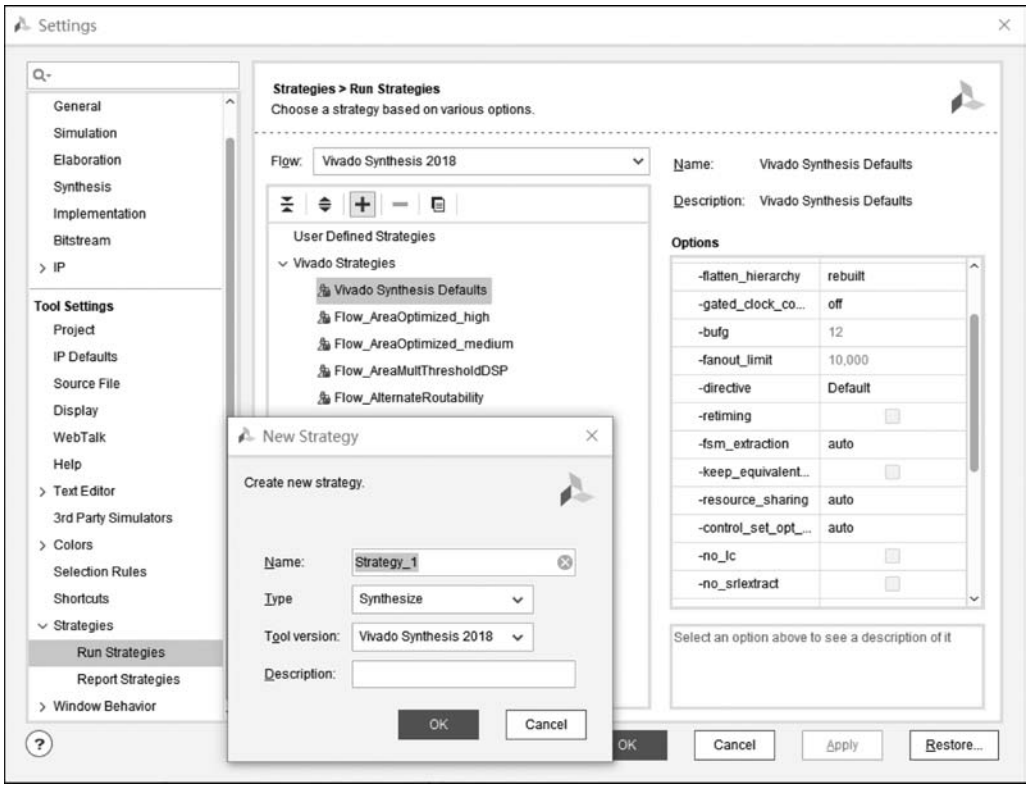

图 3.79　自定义综合策略

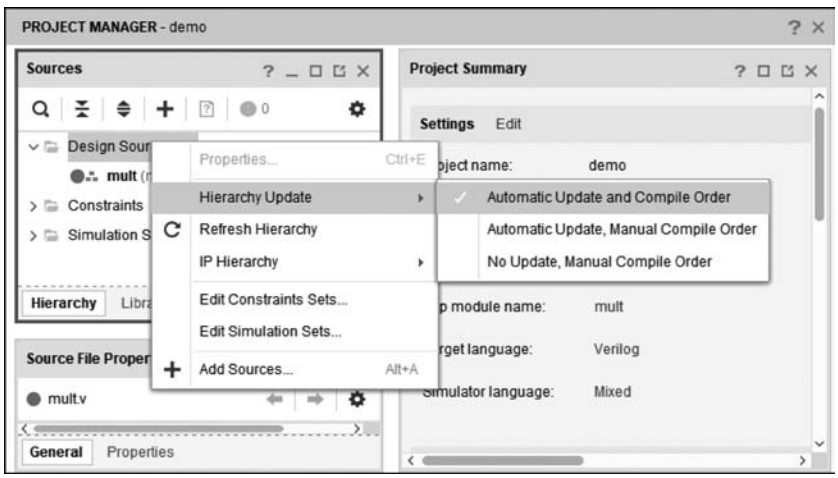

图 3.80　设置自动更新编译顺序

习题 3

3.1　用 Verilog 语言编写一个功能类似 74ls90 的程序，并用 Vivado 软件进行综合和仿真。

3.2　在 3.1 题的基础上，将 74ls90 的 Verilog 程序封装成一个 IP 核，并采用 IP 核集成的方式设计一个模 12 计数器，进行引脚锁定和下载。

3.3　在 3.2 题的基础上，调用两个 74ls90 的 IP 核，采用 IP 核集成的方式设计一个模 60 计数器，个位和十位均采用 8421BCD 码的编码方式，进行引脚锁定和下载。

3.4　基于 Vivado 软件，设计功能类似 74163 的 IP 核，并采用 IP 核集成的方式设计一个模 24 计数器，进行引脚约束和下载。

3.5　用数字锁相环(PLL)实现分频，输入时钟频率为 100MHz，用数字锁相环得到 6MHz 的时钟信号，用 Vivado 中自带的 IP 核(Clocking Wizard 核)实现该设计。