Lab0：FPGA 设计基础

1　硬件实验平台简介

Nexys A7（以前称为 Nexys 4 DDR）是一款易于使用但功能强大的 FPGA 开发板。采用 Xilinx Artix-7 FPGA（XC7A100T-1CSG324C）芯片的数字系统开发平台，具有大规模、高容量的 FPGA, 海量的外部存储, 各种 USB、以太网、以及其它接口，能够满足从入门级组合逻辑电路到强大的嵌入式系统的设计。开发板上集成的 3 维加速度传感器、温度传感器，MEMs 数字麦克风，扬声器放大器以及大量的 I/O 设备，而且包含了 128MB DDR2 的 SDRAM 存储器，使得 Nexys A7-100T 能够用于多种多样的数字系统设计，如图 0.1 所示。

图 0.1 Nexys A7-100T 开发板

主要 I/O 端口包括：

（1）16 个拨档开关 SW0-SW15，当开关拨到上档时，表示输入引脚为高电平；当开关拨到下档时，表示输入引脚为低电平。

（2）16 个 LED 指示灯 LD0-LD15，当输出引脚为高电平时，相应的 LED 指示灯点亮，否则 LED 指示灯熄灭。

（3）5 个按钮 BTN(C/D/L/R/U)，当按钮按下时，相应的输入引脚为高电平；松开按钮时，相应的输入引脚为低电平。

（4）8 个七段数码管，开发板上使用了 2 组 4 位带小数点的 7 段共阳数码管。

Nexys A7 由 Xilinx 公司的 Vivado 设计套件支持，可使用免费的 WebPACK 版本。

需要说明的是，课程实验内容并不依赖于 Nexys A7 开发板，也可以在 DE10-standard 或其他类似的 FPGA 开发板上实现。

2 Vivado 开发流程

Vivado 设计套件是 Xilinx 公司发布的 FPGA 集成设计环境，包含设计输入、逻辑仿真、设计综合、布局布线、时序分析、下载配置、验证测试等一体化的 FPGA 开发工具。Vivado 不仅包含传统上的寄存器传输级（RTL）到比特流的 FPGA 的设计流程，而且提供了基于知识产权 IP 核的系统级设计。

基于 HDL 的设计流程一般分为前端和后端两个阶段，前端完成程序编写、语法分析和逻辑实现，在通过编译后，进行激励仿真，对设计的逻辑功能进行验证，完成功能仿真；分成模块设计、HDL 编写、激励输入、功能仿真等阶段。后端通常分成设计综合、器件实现和装配验证等阶段，涉及到具体的 FPGA 器件，在功能仿真通过后，可以进行设计综合和布局布线；将逻辑功能映射到实际物理器件上，生成网表文件和二进制位文件；针对物理器件的实际延时和物理性能，分析最终器件是否能满足设计要求；最后将二进制位文件写入 FPGA 开发板中，在 FPGA 器件上进行验证。基于 HDL 的设计流程如图 0.2 所示，每个后续阶段遇到问题都有可能需要返回前导阶段进行修改。

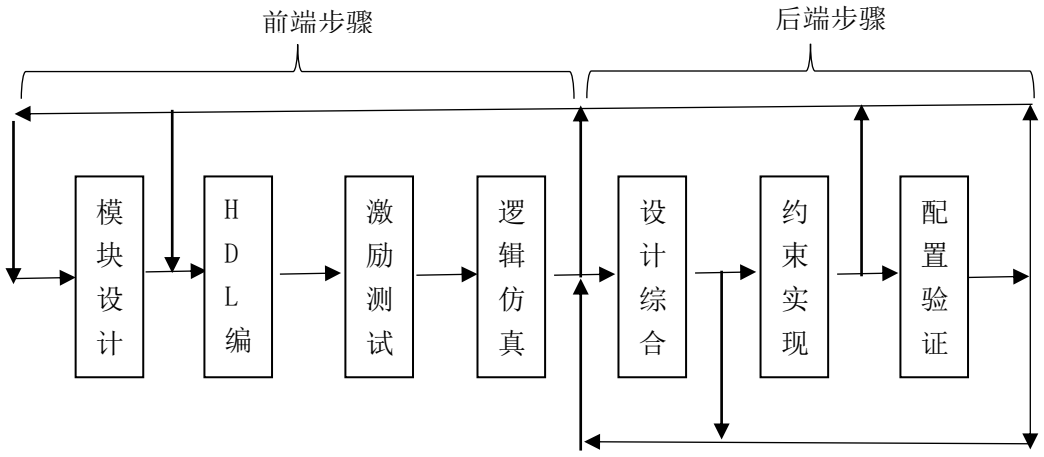

图 0.2 基于 HDL 的设计流程

Vivado 设计的过程一般包括：创建 Vivado 项目，创建模块，创建用户约束文件，插入创建的模块，声明创建的约束文件，进行功能仿真，对创建的模块进行综合，执行设计，生成编程 bit 文件，最后将文件下载到开发板上验证设计的正确性等步骤。

在开始实验之前，需要安装 Vivado 软件，选择一个稳定版本即可，如 2018.2 或 2020.2 版本，不需要安装最新版本。（提示：**实验室里计算机已经安装了 Vivado 软件，在启动时选择 Digital Design 分区，则可以使用**）

下面将通过一个实验案例来说明使用 Vivado 设计数字系统的流程，主要包括创建项目、输入代码、逻辑仿真、设计综合、器件实现、装配验证等步骤的方法。

实验要求设计一个双控开关，使用两个拨动开关控制一个 LED 灯，并在 Nexys A7 开发板上进行验证演示。假设两个开关分别用 A 和 B 来表示，灯用 F 来表示，那么双控开关的逻辑表达式是：F = A⊕B = $\bar{A} \cdot B + A \cdot \bar{B}$。

使用 Vivado 设计这个实验的流程如下：

1、创建 RTL 项目

双击桌面上的 Vivado 图标打开 Vivado IDE 页面，如图 0.3 所示。

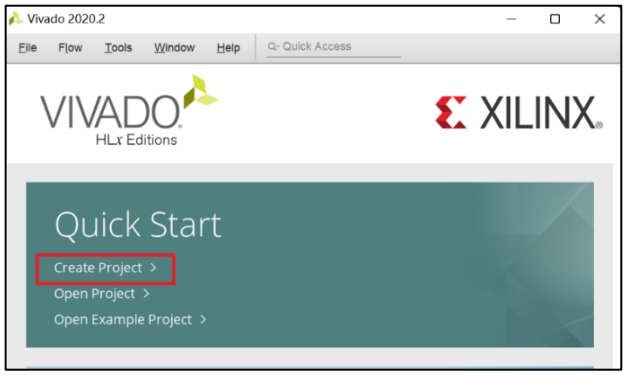

图 0.3 Vivado 初始页面

图 0.4 新建项目向导页面

点击 Create Project，弹出新建项目向导页面，如图 0.4 所示。

点击 Next 按钮，在弹出的页面中填写项目名称和项目的工作路径，如图 0.5 所示。需要为本次实验项目取一个名字比如：lab0，还要为本次实验项目生成的文件指定一个存放位置，比如：D 盘 My_design 文件夹。Vivado 则自动在 My_design 文件夹下创建一个子目录 lab0，本次实验的所有文件都将保存在 /My_design/lab0 中。（提示：文件名、项目名和路径中都不能含有空格，并不能包含中文字符。项目名和文件名可以使用字母开头，是字母、数字和下划线的组合。）

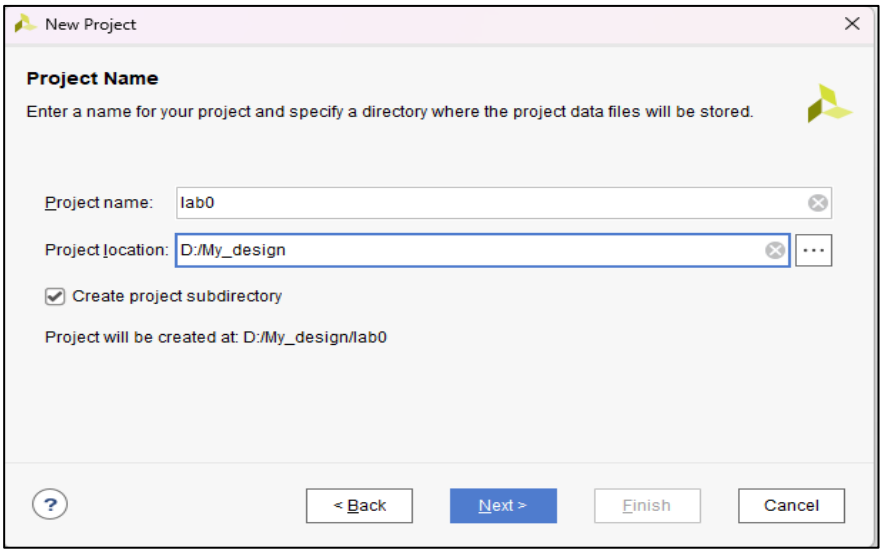

图 0.5 项目名称和存储路径设置页面

点击 Next 按钮，进入项目类型页面，如图 0.6 所示，指明项目类型，这一步定义项目源文件的类型，选中“RTL Project”。勾选“Do not specify sources at this time”选项，可以跳过在新建项目的过程中添加已有的设计源文件的过程。

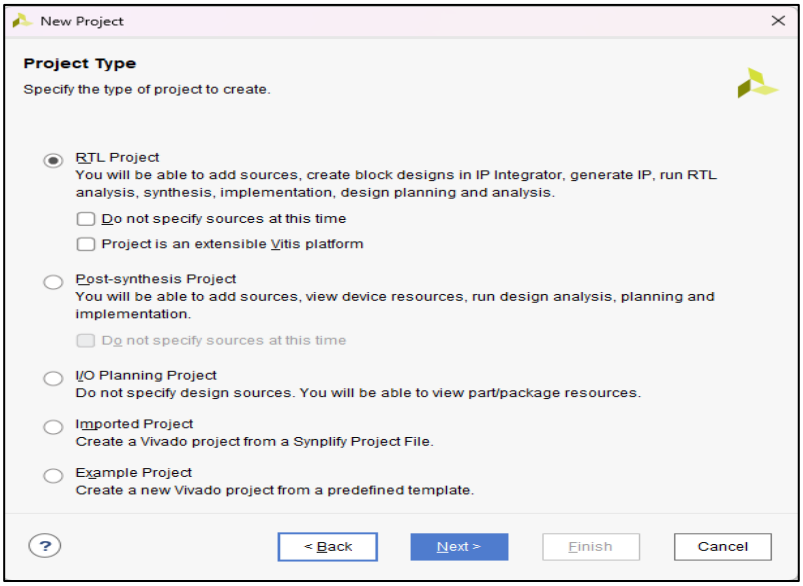

图 0.6 项目类型选择页面

点击 Next 按钮，没有可添加的 IP 核，所以不需要添加 IP 核。点击 Next，目前没有可添加的约束文件，继续点击 Next 按钮，进行下一步配置。

在目标器件“Default Part”（默认部件）页面中，根据实验平台选择相应的 FPGA 器件。本次实验使用是 Nexys A7-100T 开发板，使用的 FPGA 器件为 Artix-7 系列芯片，Family 选项为 Artix-7，封装形式 Package 选项为 csg324，速度等级 Speed grade 选项为-1。在列表显示的器件中，选择 xc7a100tcsg324-1 的器件，如图 0.7 示。

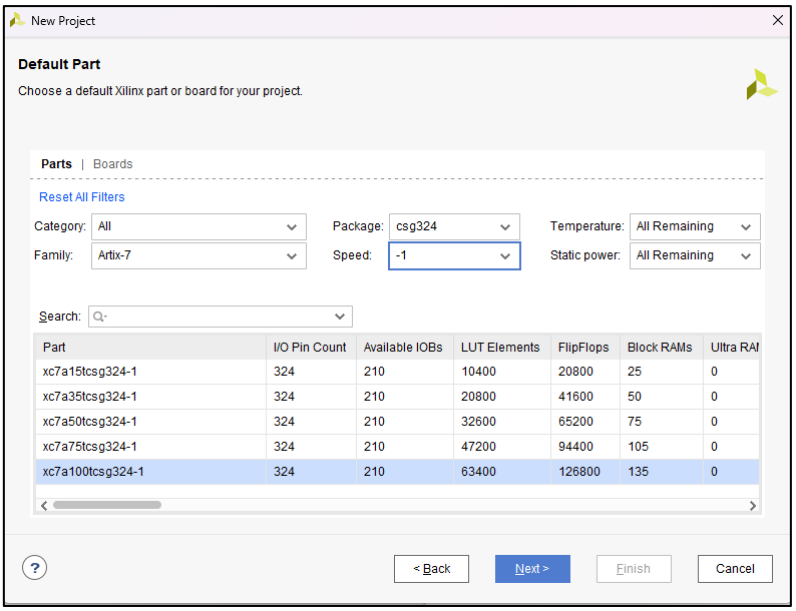

图 0.7　FPGA 目标器件的选择页面

如果 Vivado 中已经安装 Digilent 公司的开发板文件(https://github.com/Digilent/vivado-boards/tree/master/new/board_files)，就可以在 Boards 对话框的 Vendor 中选择：digilentin.com，在 Name 中选择 Nexys A7-100T，然后在列表中选中 Nexys A7-100T 开发板。如果没有安装 Digilent 公司的开发板，则可点击 Refresh 按钮更新最新的板卡信息。

点击 Next 按钮，进入新建项目汇总页面，核验相关信息与设计所用的 FPGA 器件信息一致性，如图 0.8 所示。

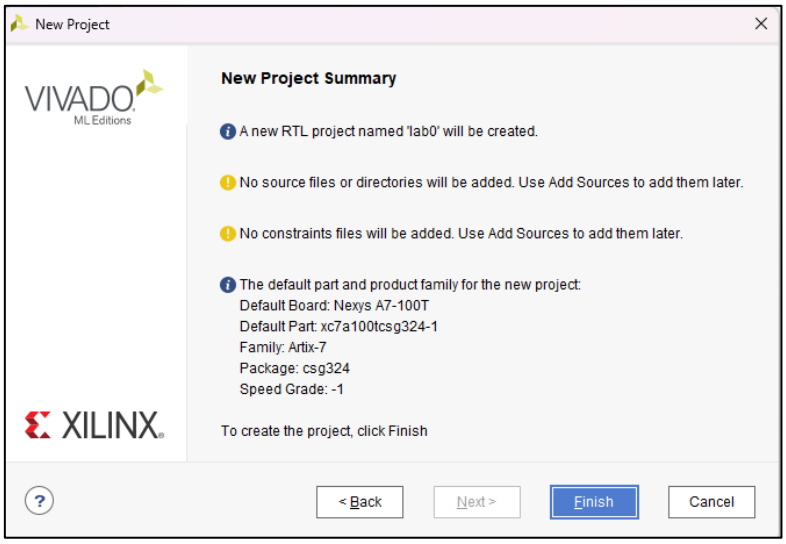

图 0.8 FPGA 目标器件信息确认页面

点击 Finish 按钮后，进入 Vivado 新建项目页面，如图 0.9 所示，可以输入或添加项目文件。

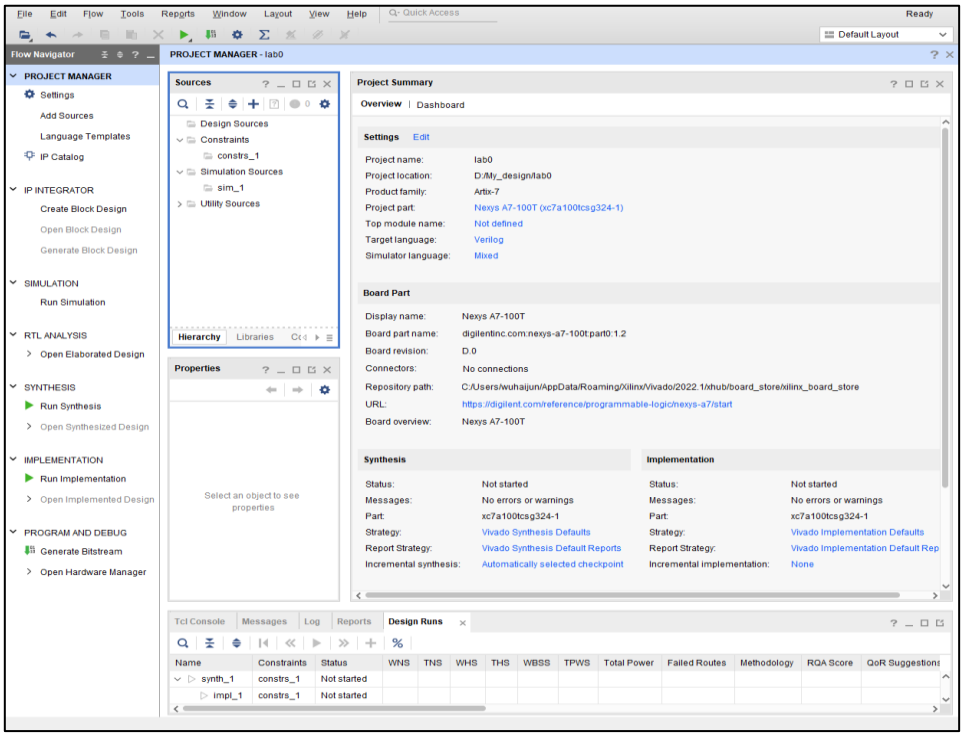

图 0.9 空白项目页面

2、输入设计文件

在 Vivado 项目页面中，点击 Flow Navigator 下的 Project Manager->Add Sources 对话框或中间数据窗口区 Sources 中的“+”（Add Sources）图标，打开添加源文件类型选择页面，如图 0.10 所示。

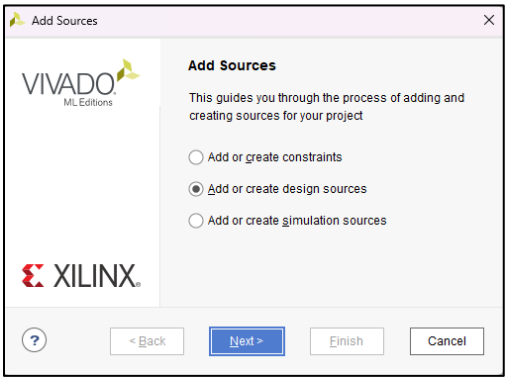

图.10 源文件类型选择页面

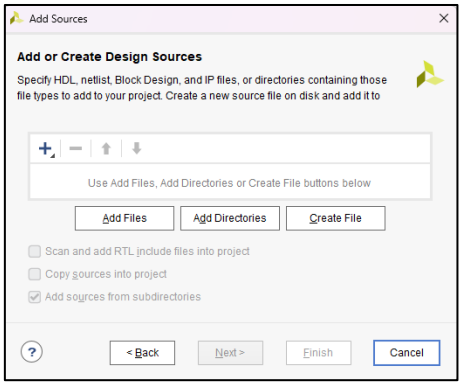

图 0.11 添加或新建设计源文件

选中“Add or Create Design Sources”，用来添加或新建数字电路设计源文件，如图 0.11 所示。在 Add Sources 对话框，选择点击“+”或“Add Files”按钮，则添加已经存在的电路设计源文件。

选择“Create File”则创建电路设计源文件，弹出新建源文件名对话框，如图 0.12 所示。在“File Type”中选择 Verilog，在“File Name”栏输入该项目的顶层实体文件名，例如“my_xor”。点击 OK 按钮。新添加的 Verilog 文件“my_xor.v”出现在对话框中，如图 0.13 所示。（提示：文件名称以字母开头，由字母、数字、下划线来组成，不能出现中文和空格，不能以数字开头。Verilog HDL 对字母的大小写敏感，编写代码时请注意。）

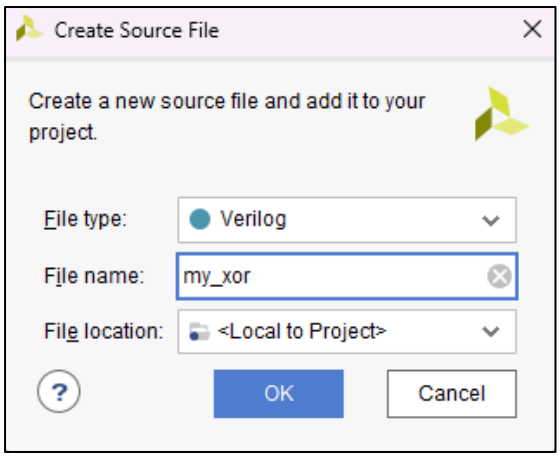

图 0.12 新建源文件名对话框

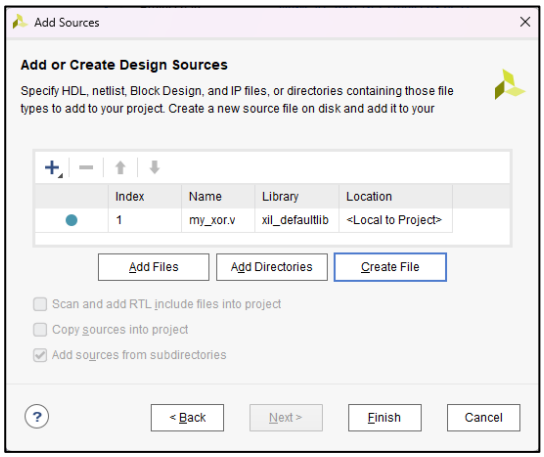

图 0.13 新建源文件确认页面

新建源文件名输入后，点击 Finish 按钮，在弹出的 Define Module 中的 I/O Port Definition，输入此次设计模块中所需要的端口，在 Direction 选项中，需要确定端口输入输出属性；如果端口类型为总线型，需勾选 Bus 选项，并通过 MSB 和 LSB 确定总线宽度，如图 0.14 所示，完成后点击 OK 按钮。（提示：这一

步也可以不设置端口，后续在 Verilog 文件中直接声明就可以。如果这一步设置有误，也可以在 Verilog 文件中修正。）

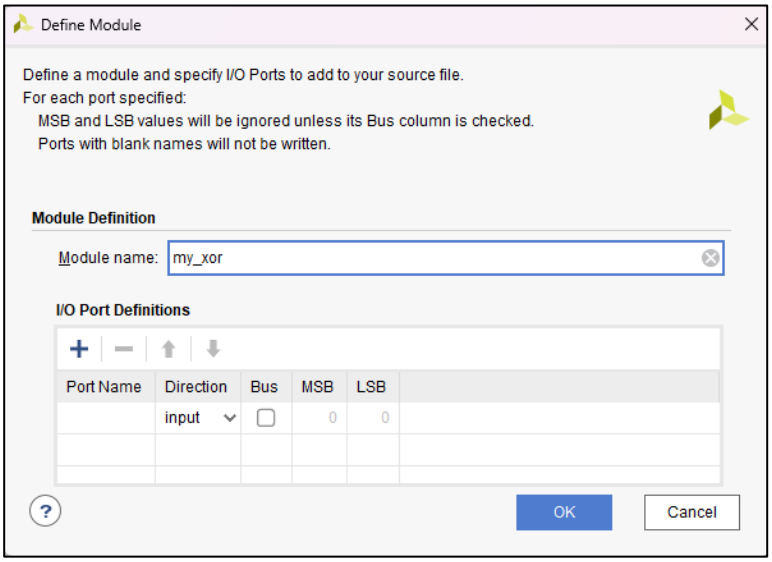

图0.14 模块IO端口定义

添加完毕之后点击 OK 按钮，进入 Vivado 主页面。创建的设计代码源文件 my_xor.v 保存在 Sources 的 Design Sources 中。双击“my_xor.v”打开该文件，可以看到 Vivado 已经自动生成了代码框架。如果在上一步中设置了端口属性，则此时的代码中已经对端口进行了声明，如没有设置端口，则需要在源文件中声明，如图 0.15 所示。

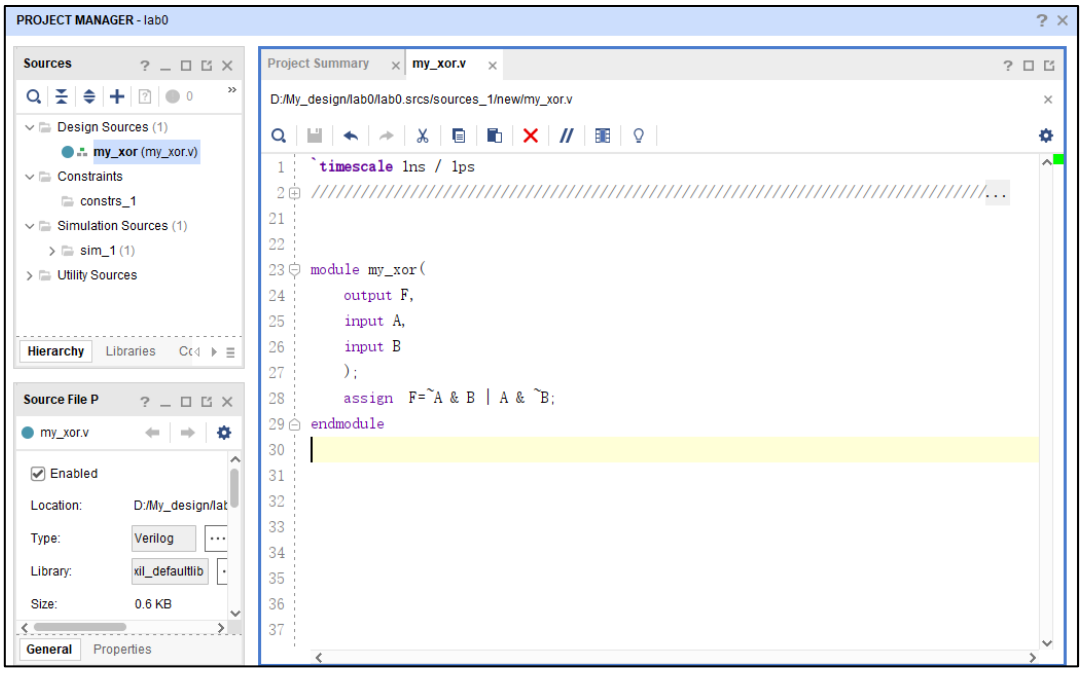

图0.15 设计文件选择及编辑窗口

在源文件 my_xor.v 编辑窗口的模块中插入相应的设计代码：“assign F=~A&B | A&~B; ”，输入后，保存源文件。

编译器会自动检查语法、拼写错误，并将此文件加入到项目库中。如果编译器没有发现语法或拼写等错误，则在源程序编辑窗口的右上角显示绿色小方块。如果存在语法错误，则在编辑窗口的右上角显示为红色方块，并在发生语法错误的那一行右侧边框上显示为红色，Source 窗口的快捷工具栏上用橙色圆圈提示错误信息记录数，同时在结果窗口区的 Message 栏中提示语法错误信息。检查代码中是否有书写、语法等错误，消除错误后，源文件编辑恢复正常状态，进入逻辑仿真阶段。

3、逻辑仿真 Simulation

逻辑仿真是验证代码功能实现的基本方式。设计文件完成之后，要检查设计的逻辑是否符合要求，可以通过观察电路的输入输出时序图，来验证设计功能的正确性。

在 Vivado 中要通过添加设计文件的仿真测试程序来编辑电路的输入时序，验证输入和输出的逻辑关系。

创建仿真测试文件步骤如下：

（1）、在项目管理器中，点击“Add source”，在创建源文件选择页面中选择第三项 Add or Create Simulation Sources，如图 0.16 所示。

图 0.16 创建仿真测试文件页面

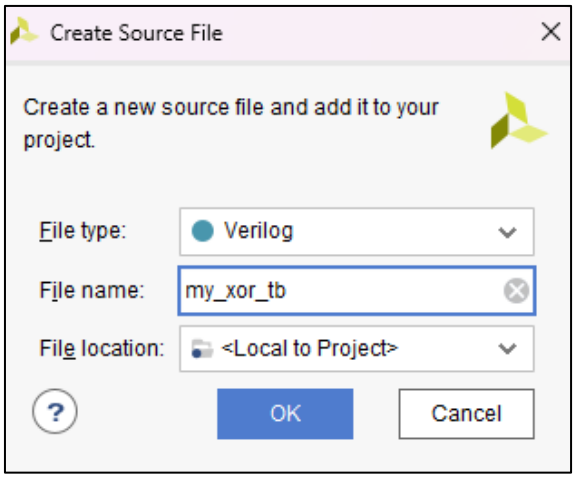

图 0.17 仿真测试文件名确认对话框

点击 Next 按钮，进入仿真测试文件创建页面，点击 Create File 按钮，进入创建仿真测试文件确认对话框，如图 0.17 所示。输入仿真测试文件名，如：my_xor_tb，创建一个新的激励测试文件，点击 OK 按钮完成创建。

仿真测试文件创建页面中，点击 finish 按钮。则弹出模块 IO 定义对话框，由于仿真测试文件不需要对外端口，所以 Port 定义部分空着即可，不要设置端口。单击 OK 按钮，完成仿真测试文件的创建。

（2）、在 Sources 窗口的 Simulation Sources 菜单的 sim_1 下，将显示新建的仿真测试文件 my_xor_tb.v。双击 my_xor_tb.v，则在工作区输入仿真测试代码，如图 0.18 所示。

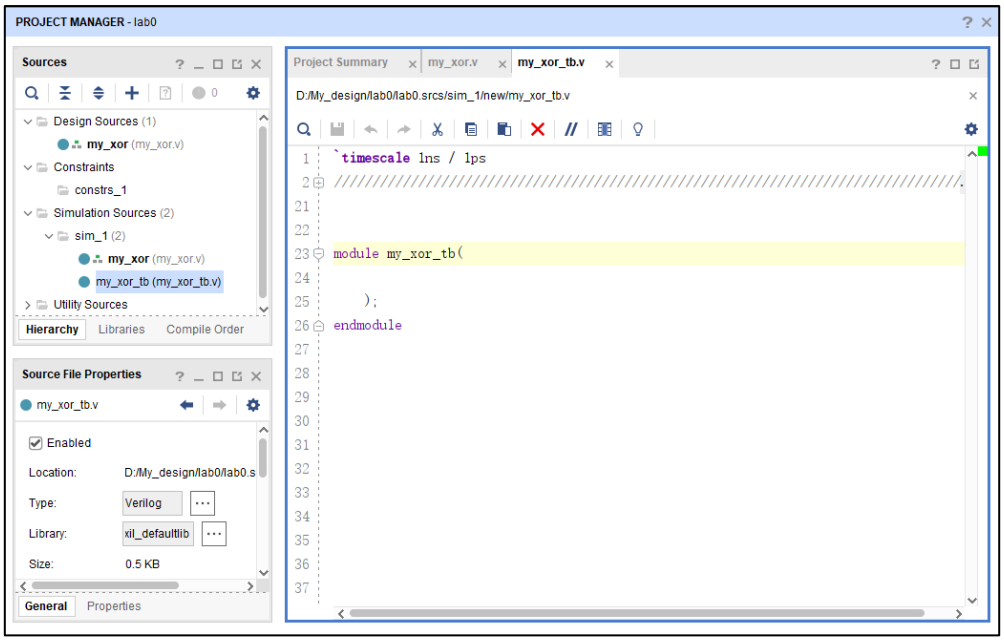

图 0.18 仿真测试文件输入输出端口定义页面

测试代码需要完成对被仿真测试设计文件的模块 Module 进行实例化，给模块中输入信号赋予不同的初值，然后在执行后检查输出的状态变化和时序是否符合设计要求。本实验中的仿真测试程序如下：

```
`timescale 1ns / 1ps
module my_xor_tb();
    wire f;        //输出变量要声明为 wire 类型的，这里 wire 不可缺省
    reg a,b;       //输入变量声明为 reg 类型的
   my_xor s0(.A(a),.B(b),.F(f));
   initial begin
    begin     a = 1'b0;     b = 1'b0;    end
    #200 begin     a = 1'b0;     b = 1'b1;    end
    #200 begin     a = 1'b1;     b = 1'b0;    end
    #200 begin     a = 1'b1;     b = 1'b1;    end
    #200 begin     a = 1'b0;     b = 1'b0;    end
   end
endmodule
```

仿真测试程序首行：

```
`timescale 1ns / 1ps
```

此语句说明测试程序的时间单位为 1ns 且时间精度为 1ps。如“#200”代表延时时间 200ns。在生成测试文件的时候，编译器会在测试文件的第一行自动产生这个语句，如果自动产生了，则不需要重复输入了，但用户可以修改延时时间和时间精度。

```
my_xor s0(.A(a),.B(b),.f(F));
```

这条语句是将 my_xor.v 文件中的 my_xor 模块实例化，并取名为“s0”。因为需要对 my_xor 模块进行仿真，因此要实例化 my_xor 模块。将 my_xor_tb 模块中的 a，b，f 变量值分别和 my_xor 中的 A，B 和 F 变

量值一一对应起来。（提示：测试代码的端口参数列表要保持为空；在变量声明中，输入变量声明“reg”类型的，输出变量要声明为“wire”类型的。）

保存仿真测试文件，如果没有语法错误，这时在 Sources 栏的 Simulation Sources 栏下会出现实例“s0”。因为测试代码中将“my_xor”模块实例化，并取名为“s0”，如图 0.19 所示。如果没有出现“s0”，则检查测试代码是否有逻辑错误，修改错误，保存，直至测试代码被正确解析。

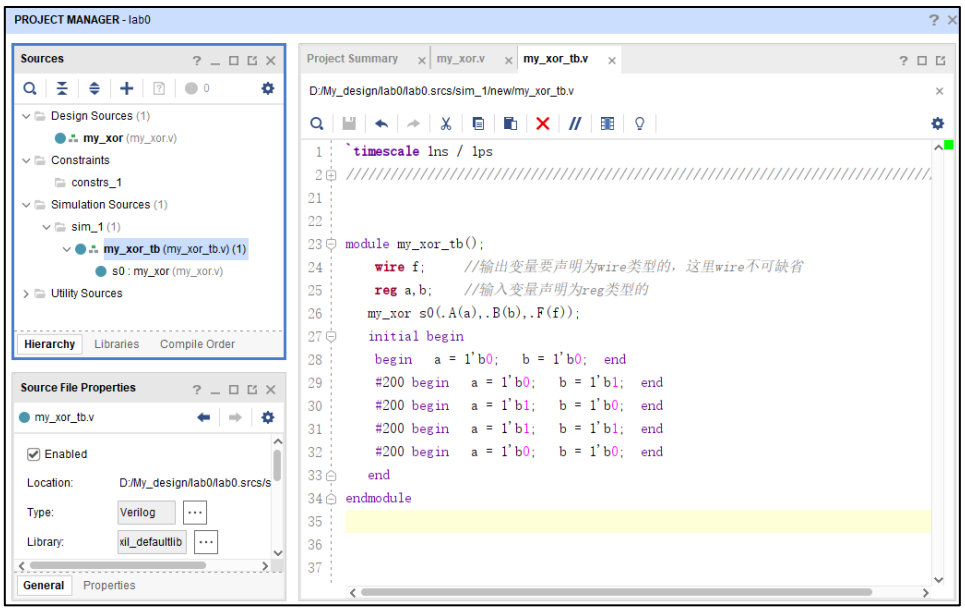

图 0.19 创建完成仿真测试文件

（3）点击流程导航栏 Simulation 下的 Run Simulation 菜单，选择 Run Behavioral Simulation，如图 0.20 所示。

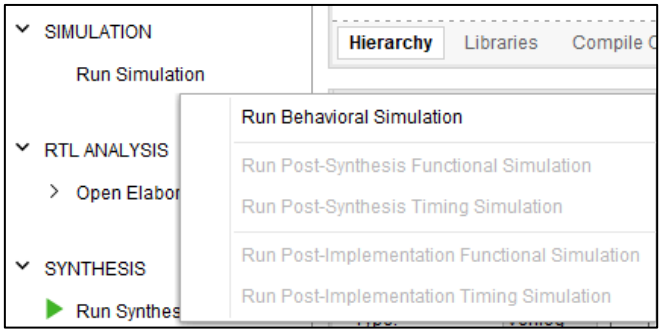

图 0.20 启动逻辑仿真测试

执行仿真测试程序后，进入输入输出时序仿真页面，如图 0.21 所示。

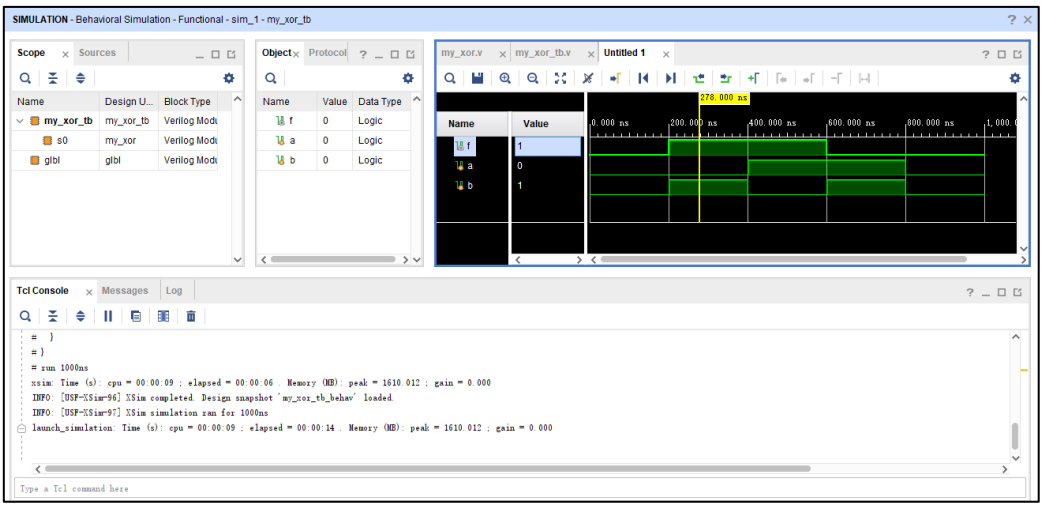

图 0.21 仿真页面

还可以通过左侧 Scope 一栏中的目录结构定位到设计者想要查看的 module 内部寄存器，点击 Scopes 栏中的“s0”，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。

在 Untitled 窗口中，如果没有出现上述仿真图，可能是因为比例因子不合适，可以对仿真图进行放大、缩小，通过点击图标和向左拖动仿真图窗口下方滚动条来选择观察。

可通过选择工具栏中选项来进行波形的仿真时间控制，如：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，如图 0.22 所示。

图 0.22 仿真波形设置工具栏

对仿真图像进行多次“缩小视图”并移动滚动条，最终得到的仿真效果图如图 0-27 所示。验证输入输出波形与预设的逻辑功能是否一致。从仿真图中可以看到，在运行到 278ns 时，a=0，b=1，f=1，可见输入与输出信号之间的逻辑关系满足设计要求，仿真测试功能通过。

4、设计综合（Synthesis）

设计综合是把项目设计文件进行逻辑优化，并将 RTL 代码映射到 FPGA 器件的原语，生成网表文件。如果 RTL 代码中有语法问题，这一步就会报错；同时，综合后生成的报告需要仔细查看。

在流程导航窗口中的 Synthesis 下点击 Run Synthesis 菜单，在弹出的对话框中选择在本地化运行，然后点击 OK，如图 0.23 所示。

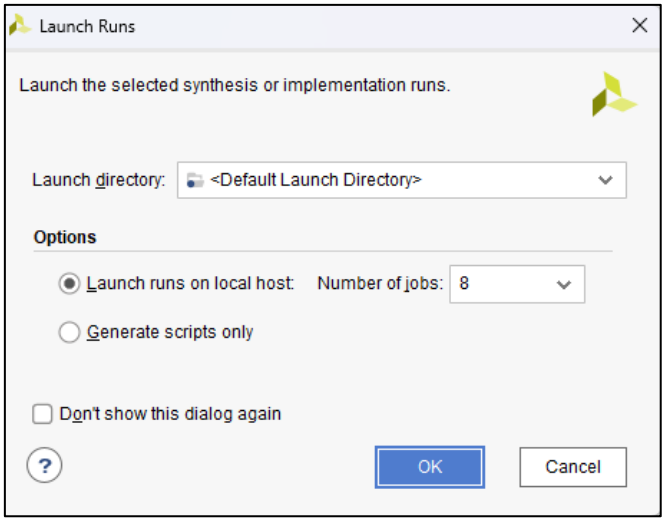

图 0.23 启动设计综合和实现

在综合过程中将对设计进行检查，检查设计中的逻辑错误等，如果报错，需认真检查“error”项，修改每一个 error 后重新选择设计综合，直至所有错误全部修改通过。

完成综合后，将弹出综合成功完成的对话框，如图 0.24 所示。有 3 个选项：

（1） Run Implementation 运行实现过程

（2） Open Synthesized Design 打开综合后的设计

（3） View Reports 查看报告

由于第一次执行综合时且尚未添加约束文件到项目中，因而需要选择 Open Synthesized Design 选项，点击 Ok 按钮后。如果已经添加了约束文件到项目中，不再需要打开综合后的设计进行查看和编辑，则选择 Run Implementation 选项，直接进入器件实现步骤。

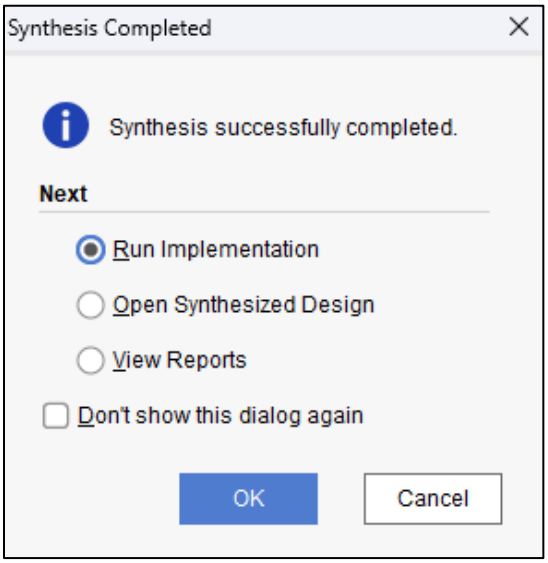

图 0.24 综合完成对话框

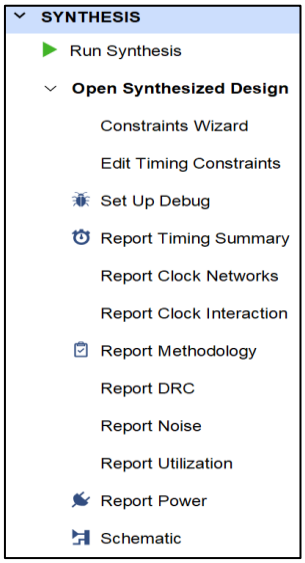

图 0.25 Synthesized Design 菜单

执行Open Synthesized Design选项后，在流程导航窗口中的Open Synthesized Design有如下选项，如图0.25所示。

选择Schematic，打开综合后的电路原理图如图0.26所示。

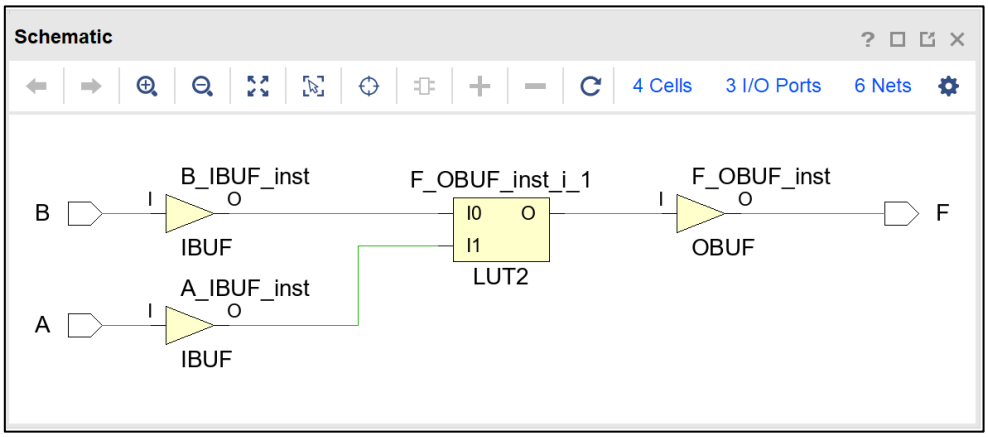

图 0.26 项目原理图

如果在综合完成对话框中，选择执行Run Implementation选项时，如果尚未没有建立约束文件，则执行过程中报错，需要点击Constraints Wizard菜单来建立约束文件。

5、创建约束文件 Contraints

设计的电路需要通过下载到FPGA上来展示验证的，FPGA芯片没有输入输出设备，必须使用外部的输入输出设备。在Nexys A7-100T开发平台上，所有的输入输出信号都被绑定到FPGA的某个引脚上了。比如开发板上的拨动开关SW0，SW1以及发光二极管LED0，是分别和FPGA的J15，L16和H17连接在一起的。设计的电路需要在FPGA中实现的，程序中的输入/输出端口分别取名为A，B和F。如果我们将在FPGA内部的F端口和FPGA的H17引脚连接起来，而H17在开发板上又是和指示灯LED0连接起来的，那么在硬件上我们就将F端口和LED0连接起来了。如果电路的F端输出“1”值，则开发板上的LED0就会亮。这样就可以使用开发板上的输入输出设备来验证我们的电路。

如何将程序中定义的输入/输出端口映射到实验开发板上的输入输出接口信号上来验证设计电路的功能呢？这就需要对程序中输入输出端口进行引脚约束，将程序中的输入输出端口和FPGA的引脚进行配置连接，从而使其连接到开发板的输入输出器件上。

有两种方法可以添加引脚约束文件：一是可以直接新建XDC的约束文件，手动输入约束语句；二是可利用Vivado中IO planning功能。

方法一：通过约束文件来实现

与添加设计源文件步骤类似，在流程导航窗口中的Project Manager 区域，点击Add Sources菜单，在弹出的添加文件对话框中，选中第一项Add or Create Constraints，点击Next按钮，如图0.27所示。

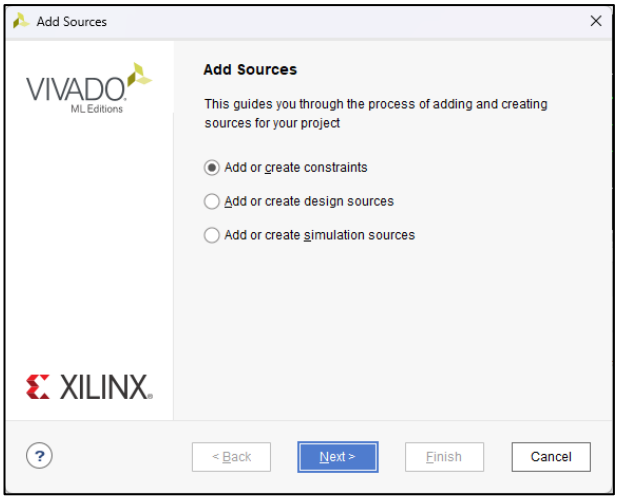

图0.27 添加约束文件选择页面

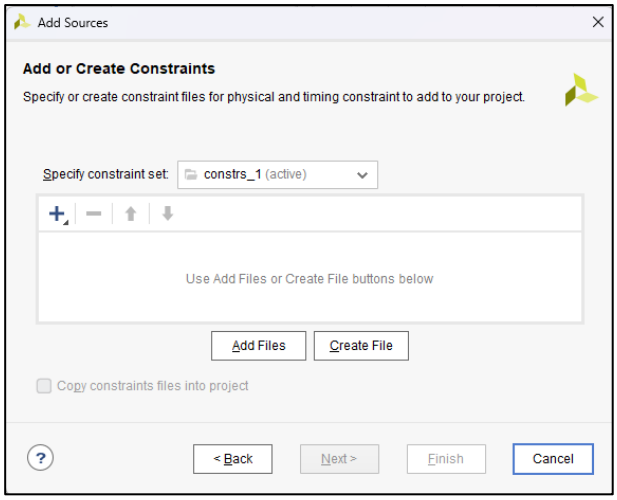

图0.28 创建约束文件页面

在弹出的Add Sources对话框中，点击“+”，选择“Create File…”，如图0.28所示。

点击Create File按钮，新建一个约束文件XDC(Xilinx Design Constraint)，输入约束文件名，例如：my_xor，如图0.29所示，点击OK按钮。

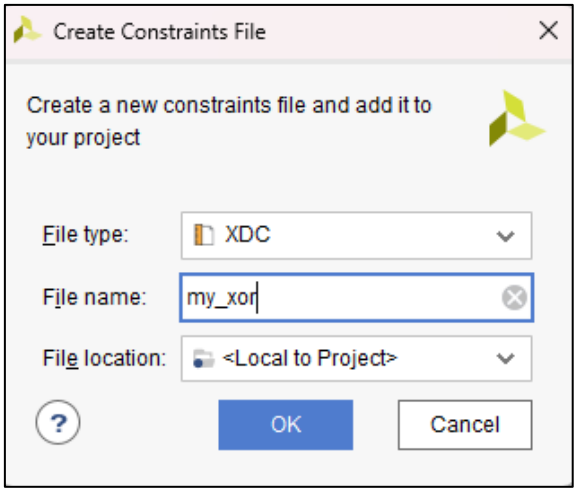

图0.29 创建约束文件名对话框

弹出建立约束文件对话框，可以看到已经添加my_xor.xdc的约束文件，在点击finish按钮，返回编辑页面。

在Source窗口中展开Constrains 选项，直到看见创建的my_xor.xdc 文件，双击创建约束文件名进入编辑模式，就可以将约束语句添加到该文件中。Vivado中的约束类型分成两类，一类是物理约束，一类是时序约束。

物理约束是指FPGA引脚分配和引脚电平的大小，假设把输入端口A、B分别连接到Nexys A7-100T实验板上的拨动开关SW0和SW1上，输出端口F连接到LED指示灯LD0上。输入和输出引脚的工作电压都为3.3V。

Nexys A7-100T实验板输入输出引脚和FPGA芯片引脚的对应关系如表0.1所示。

表 0.1 Nexys A7-100T 输入/输出信号和 FPGA 引脚分配表

拨档开关	FPGA 引脚	LED 指示灯	FPGA 引脚	数码管信号	FPGA 引脚	按钮及时钟	FPGA 引脚
SW0	J15	LD0	H17	CA	T10	BTNC	N17
SW1	L16	LD1	K15	CB	R10	BTNU	M18
SW2	M13	LD2	J13	CC	K16	BTNL	P17
SW3	R15	LD3	N14	CD	K13	BTNR	M17
SW4	R17	LD4	R18	CE	P15	BTND	P18
SW5	T18	LD5	V17	CF	T11	CLK100MHz	E3
SW6	U18	LD6	U17	CG	L18		
SW7	R13	LD7	U16	DP	H15		
SW8	T8	LD8	V16	AN0	J17		
SW9	U8	LD9	T15	AN1	J18		
SW10	R16	LD10	U14	AN2	T9		
SW11	T13	LD11	T16	AN3	J14		
SW12	H6	LD12	V15	AN4	P14		
SW13	U12	LD13	V14	AN5	T14		
SW14	U11	LD14	V12	AN6	K2		
SW15	V10	LD15	V11	AN7	U13		

从表中可以得到SW0、SW1和LD0分别对应FPGA的引脚J15、L16和H17，则在提供的nexysa7.xdc约束文件中修改sw[0]、sw[1]和led[0]引脚定义，去掉注释符号“#”，修改端口名称，然后把修改后的内容粘贴到my_xor.xdc文件中，如图0.30所示：

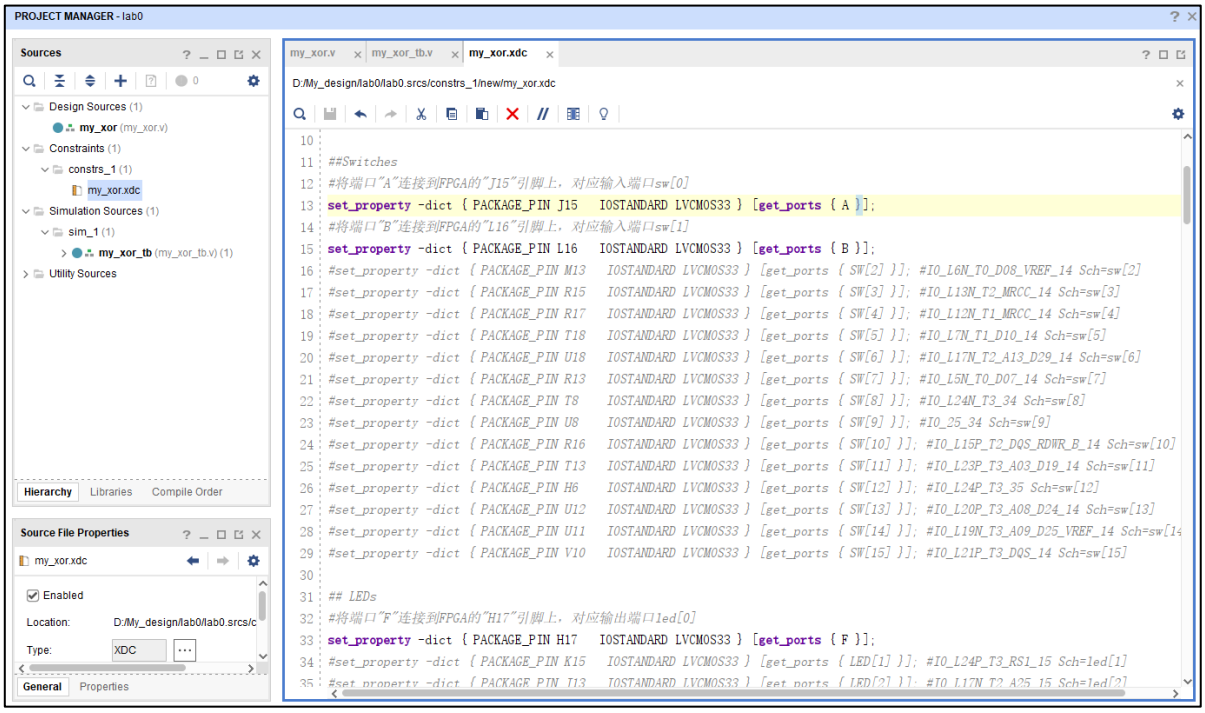

图0.30 约束文件内容

添加后点击保存按钮，检查输入输出端口设置是否有遗漏，如果有管脚设置遗漏，则产生二进制文件

时会报错。如果希望使用其它引脚，则需修改对应的封装引脚。

方法二：利用IO Planning

在流程导航窗口中，单击Run Synthesis菜单，完成对项目的综合后，点击Open Synthesized Design，打开综合设计页面，在右上角的窗口布局中选择“I/O Planning”,则在右下方的结果窗口的选项卡中选中I/O Ports栏，可以看到有2个输入IN端口和1个输出OUT端口，当鼠标悬停在IO端口行上面时，显示该行对应的输入输出变量名称，在对应的输入输出信号后输入对应的FPGA引脚标号或将信号拖曳到右上方FPGA封装引脚Package图中对应的引脚上，并设定I/O Std为：LVCMOS33，输入输出引脚约束配置完成后，如图0.31所示。详细的FPGA引脚约束和I/O电平标准，可参看开发板的用户手册。

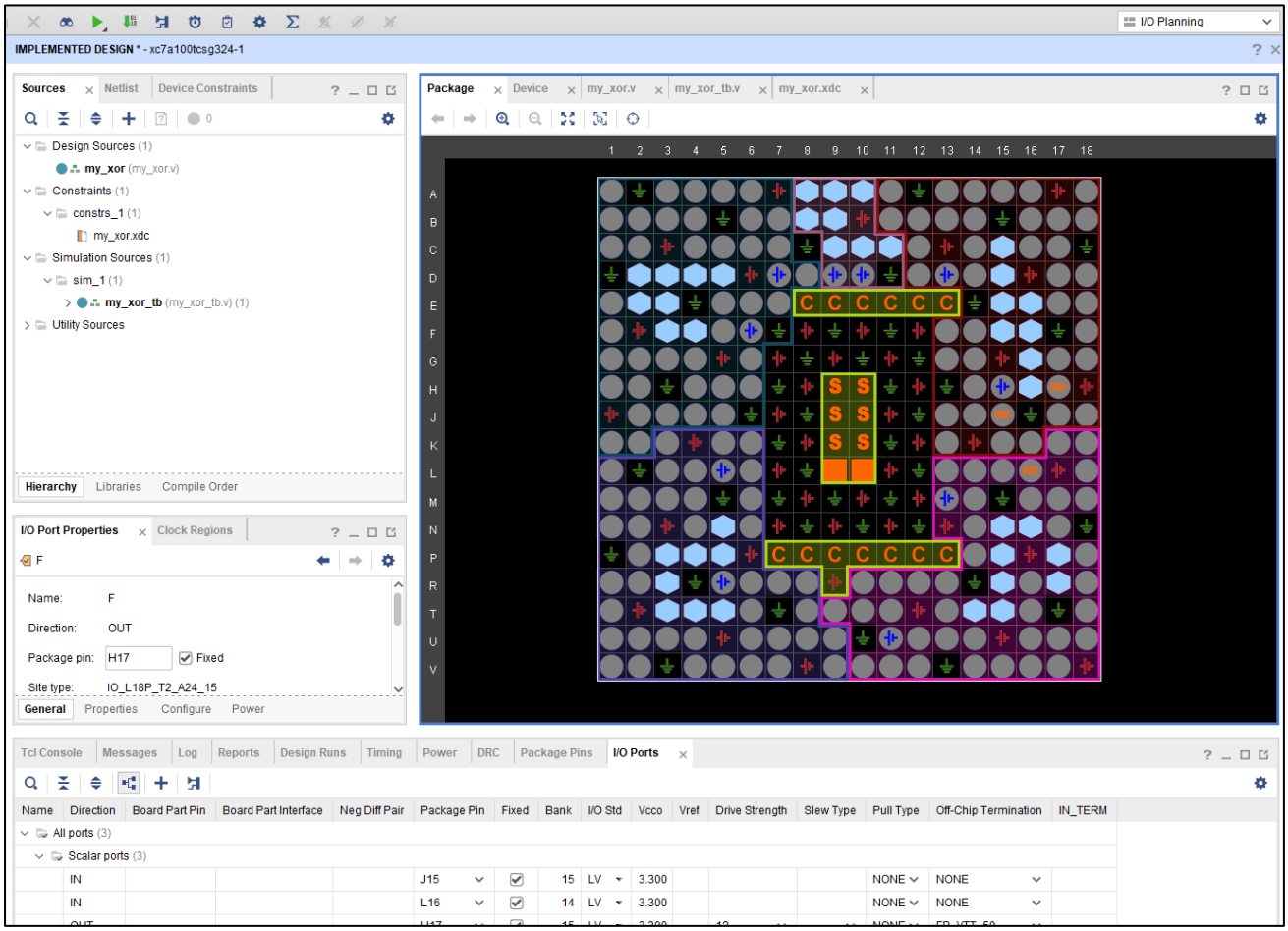

图0.31 完成I/O Ports引脚约束

输入输出信号引脚约束配置结束后，点击页面左上方的“保存”按钮，则弹出输入新建XDC文件名或选择已有的XDC文件的对话框。点击Create a new file按钮，输入文件名为：my_xor，单击OK按钮生成约束文件。在Sources窗口中的Constraints中，可以看到新建的XDC文件及其内容，如图0.32所示。

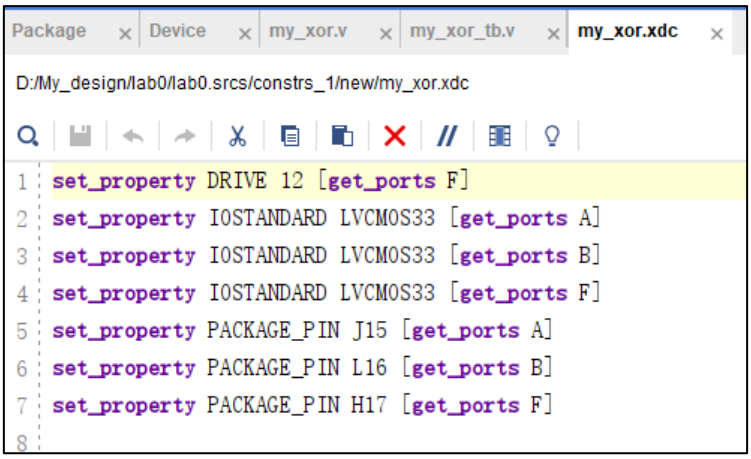

图0.32 查看约束文件内容

完成约束文件建立后，在导航窗口中选择执行Implementation。

Implementation主要包含布局布线两个过程，布局主要将综合后的基本单元放到FPGA中合适的位置，而布线则是将这些基本单元连接起来。完成项目实现后，将弹出对话框，选择Open Implemented Design，单击OK按钮，执行后，在右侧的Schematic窗口中显示电路原理图，如图0.33所示。左侧的Open Implemented Design选项菜单，可以发现该菜单选项和Open Synthesized Design选项完全相同。选择Generate Bitstream，单击OK，则生成二进制流文件；选择View Reports，单击OK，则查看综合后的各类报告。选择Cancel，则直接回到主页面。

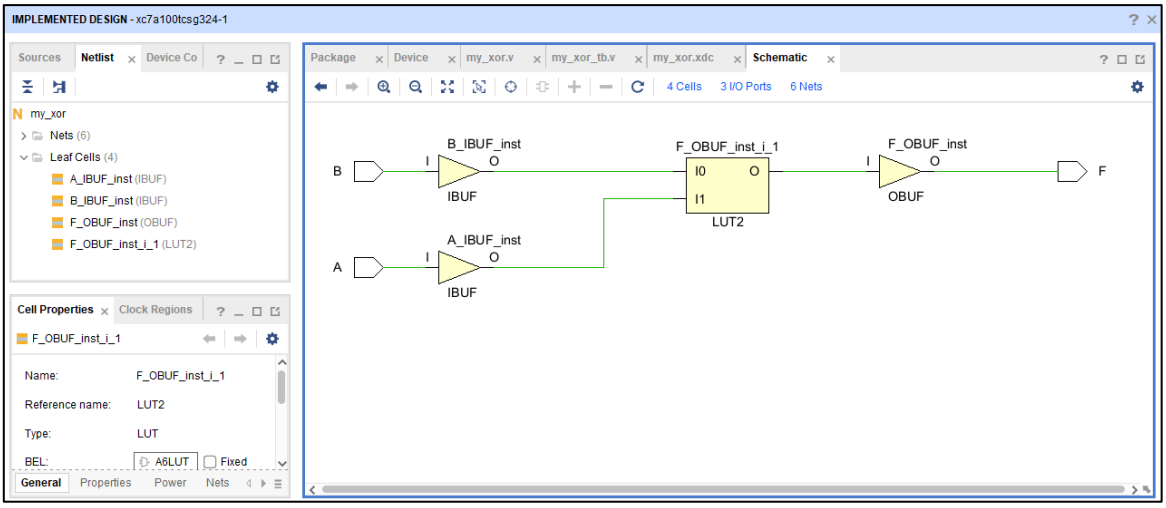

图0.33 电路原理图

项目实现通过引脚约束和时序约束，把网表文件映射到可用FPGA器件资源上，在物理上将逻辑电路互连起来。

6、配置验证

（1）生成FPGA二进制流文件。在项目实现完成的对话框中选择在Generate Bitstream选项或者在流程

导航窗口中点击Program and Debug下的执行Generate Bitstream菜单命令，项目会自动完成综合、实现、Bit文件生成过程（过程缓慢，时间比较长）！完成之后弹出对话框，可点击Open Implemented Design来查看项目实现结果，如图0.34所示。

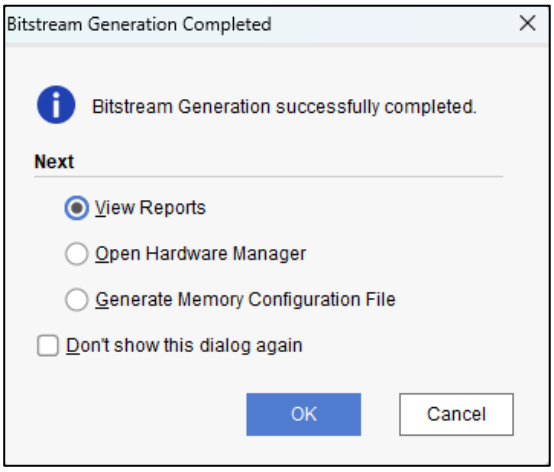

图0.34 bit文件生成完成页面

（2）建立硬件连接。

使用提供的USB连接线将Nexys A7-100T和电脑连接起来，上电前需确认开发板上的JP3 跳线配置为USB电源输入方式或外接电源；JP1 跳线配置为JTAG下载方式；然后打开开发板上的电源开关，等待全部驱动程序安装完成。

在流程导航窗口中执行Open Hardware Manager菜单命令或者在bit文件生成完成页面中选择Open Hardware Manager选项，点击OK按钮，进入硬件编程管理页面。单击Open Target，选择Auto Connect，建立和实验开发板之间的连接，如图0.35所示。

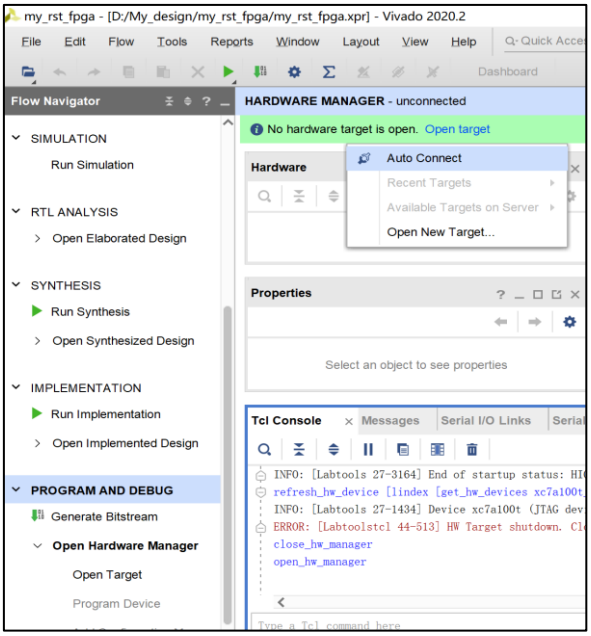

图0.35 硬件编程管理页面

接着能在Hardware 窗口看到Vivado自动开始检测硬件设备，检测到JTAG 扫描链上的Nexys A7 100T FPGA，如图0.36所示。

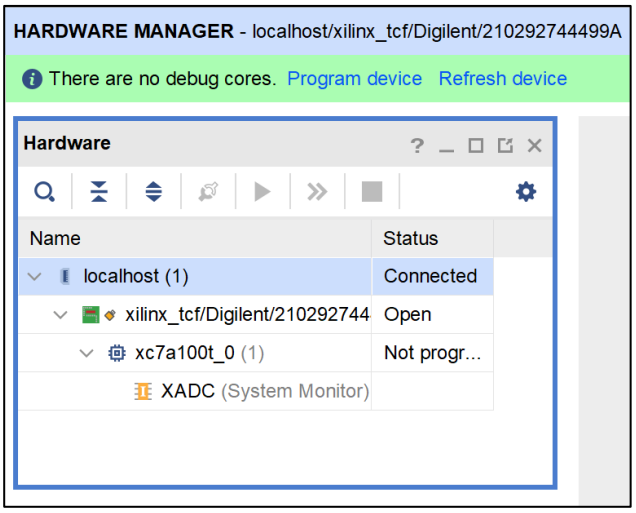

图0.36 检测到Nexys A7开发板

（3）下载bit文件。在目标的FPGA器件xc7a100t上用鼠标右击，选择Program Device…菜单或者在流程导航窗口中的Open Hardware Manager中点击Program Device菜单，将弹出芯片编程文件确认对话框，指定所需的bit文件路径，如图0.37所示。

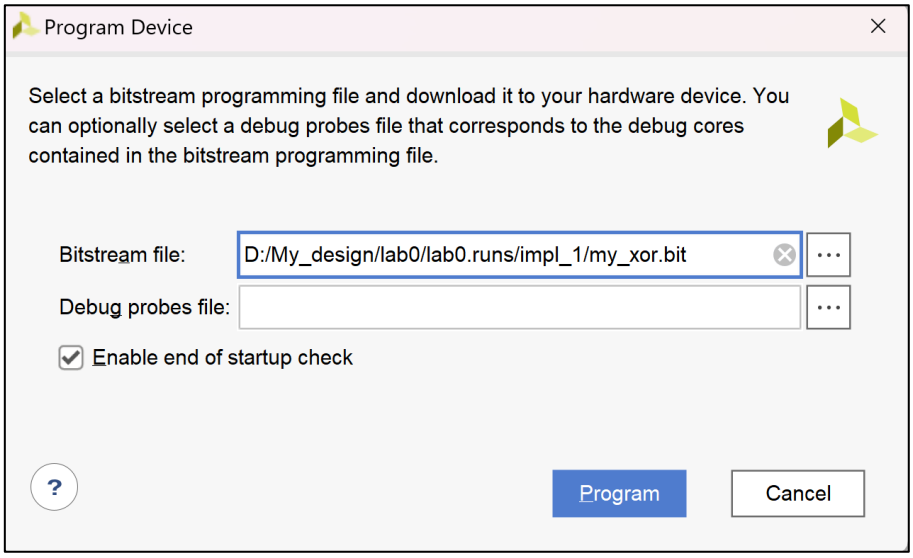

图0.37 芯片编程文件确认对话框

点击Program按钮，将bit文件下载到开发板卡上的FPGA中，下载结束后，在FPGA芯片名称后面显示“Programmed”，如图0.38所示。

请注意上图中的“.bit”文件地址路径，确保是该路径下的“.bit”文件进行Program。

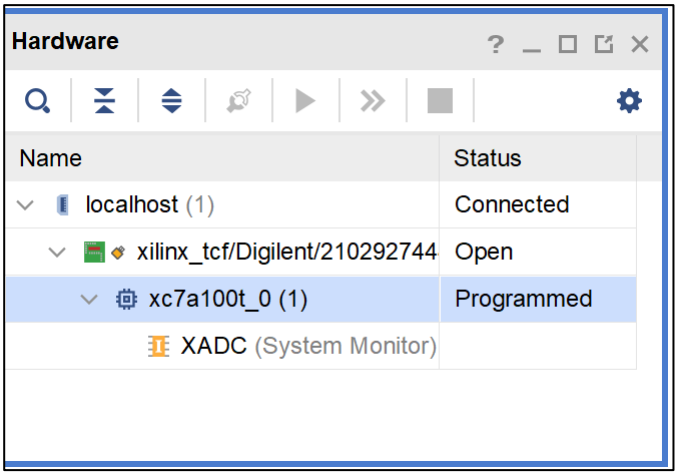

图0.38 bit文件下载到开发板

等待Hardware Manager 将bit文件烧写进Nexys A7-100T后，将首先会看到开发板上表示烧写完成的Done指示会点亮（LD21）。

（4）验证。流文件下载安装结束后，拨动开关SW0和SW1，观察开发板的指示灯LD0亮灭情况，验证电路是否满足设计要求，如图0.39所示。

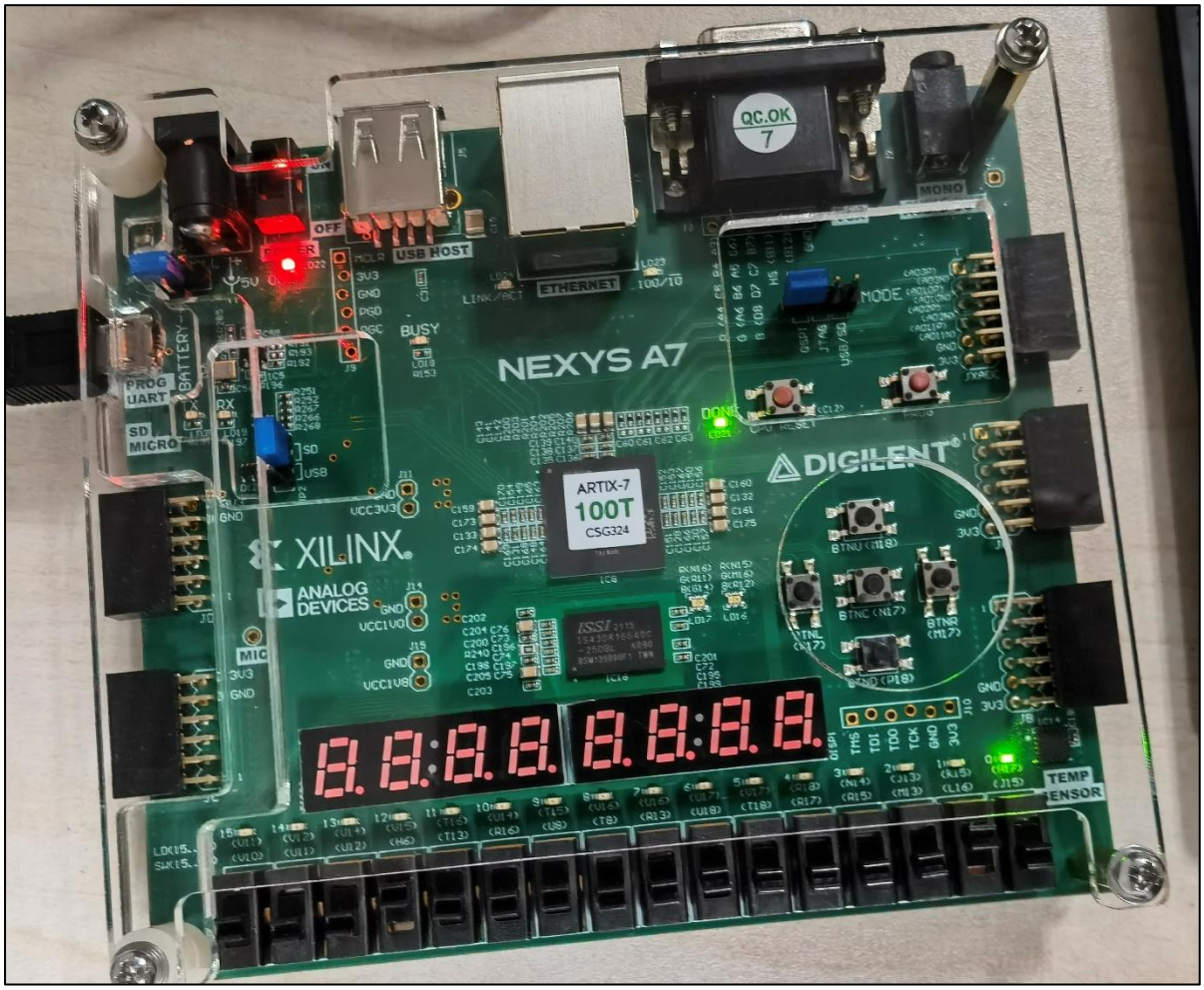

图0.39 Nexys A7-100T开发板验证实验结果

7、封装 IP 核

IP核(Intellectual property core)用于ASIC或FPGA中的预先设计好的电路功能模块，类似编程语言中的函数库，设计人员可以直接调用。IP核可重用，有助于减少重复劳动，缩短产品研发时间。IP核有三种不同的存在形式：HDL语言形式（软核），网表形式（硬核）、版图形式（固核）。

Vivado中有很多IP核可以直接调用，例如数学运算（乘法器、除法器、浮点运算器等）、信号处理（FFT、DFT、DDS等）。

本次实验设计的异或门电路也可以封装IP核，可供后续电路设计中调用。

进入打开lab0项目管理状态，点击在流程导航窗口下的Settings菜单，弹出对话框中，选择IP下Packager菜单，如图0。40所示，修改相关IP封装所有者信息，点击OK按钮进行保存。

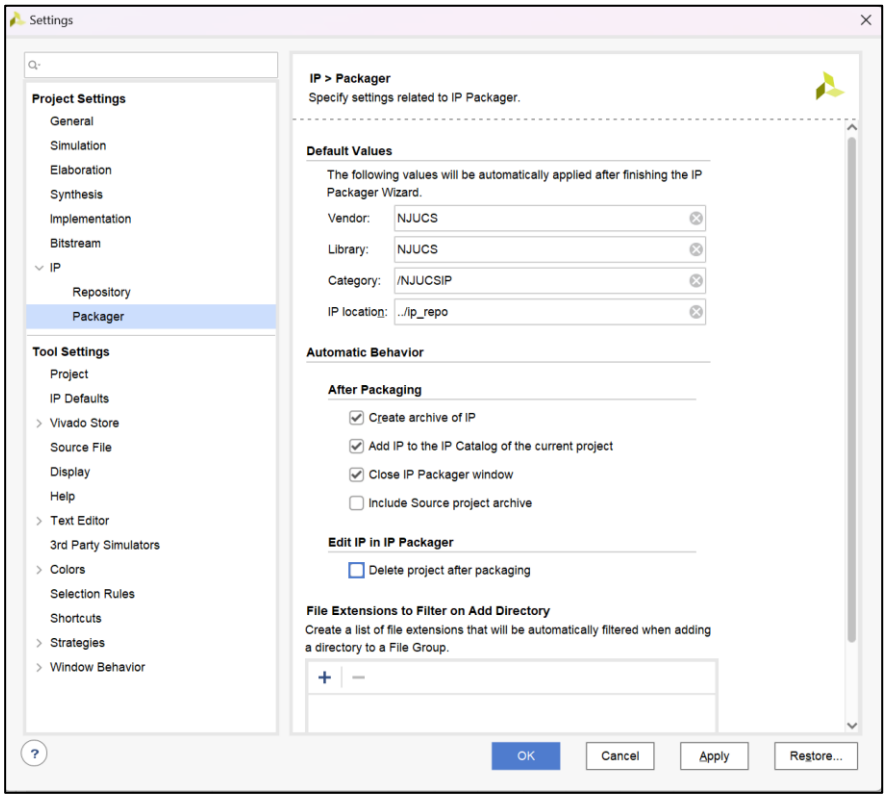

图 0.40 项目设置中设置 IP 封装所有者属性

在Vivado的菜单栏中点击Tools->Create and Package New IP菜单在弹出的窗口中单击Next按钮，如图0.41所示。在弹出的窗口中设置封装参数，选择封装当前项目，如图0.42所示，再单击Next按钮。然后设置封装文件保存路径。在弹出的窗口中，保留IP Location缺省设置，不要修改，如图0.43所示，单击Next按钮。弹出Finish对话框，点击Finish按钮完成封装。

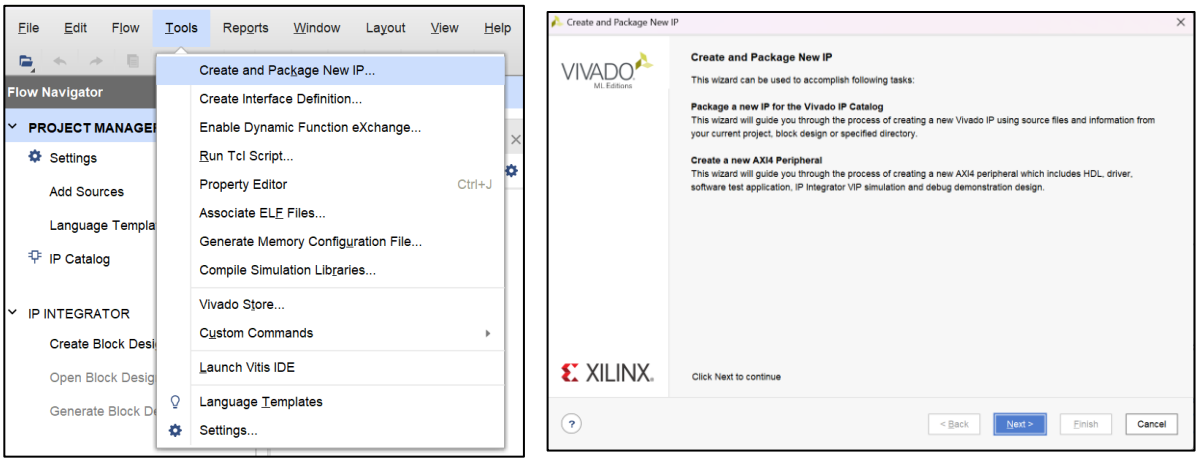

图0.41 封装IP菜单

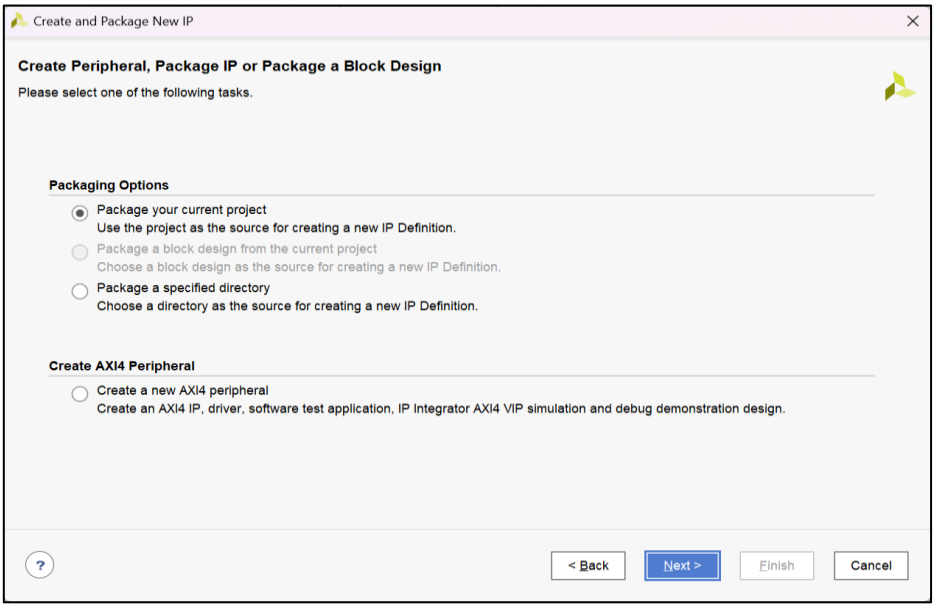

图0.42 封装选项

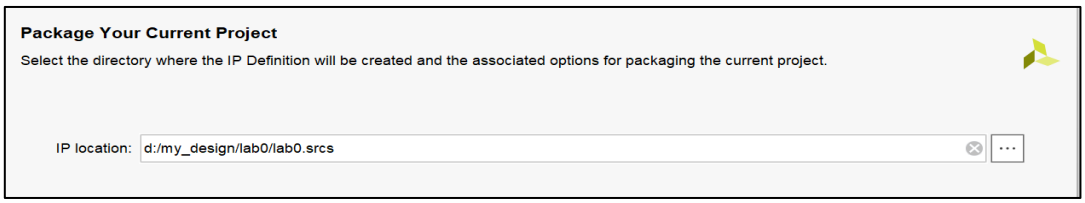

图0.43 IP Location窗口

Vivado的工作区展示了Package IP-my_xor的个性化设置对话框，可以看到封装后的IP核放在d:/my_design/lab0/lab0.srcs文件夹中，设置参数存在文件夹下的component.xml文件中，如图0.44所示，根据需要，可以修改IP核的名称、显示名称和描述信息。

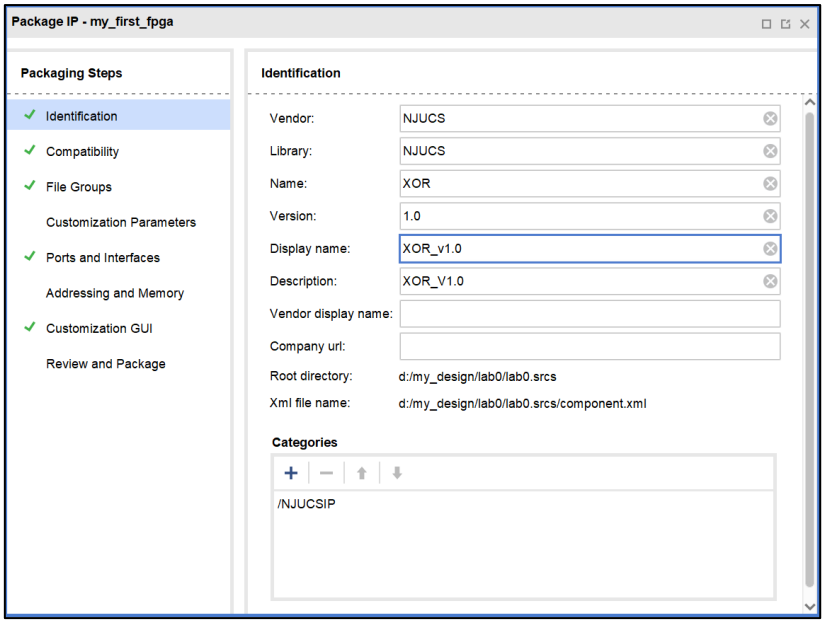

图0.44 IP核个性化设置对话框

在该对话框的Compatiblity兼容性设置可以看到本次IP核支持的芯片家族，如果还需要添加其他芯片家族，则可以点击+按钮，并选择Add Family Explicitly，弹出Add Family对话框，显示所有的Xilinx芯片家族，如图0.45所示。

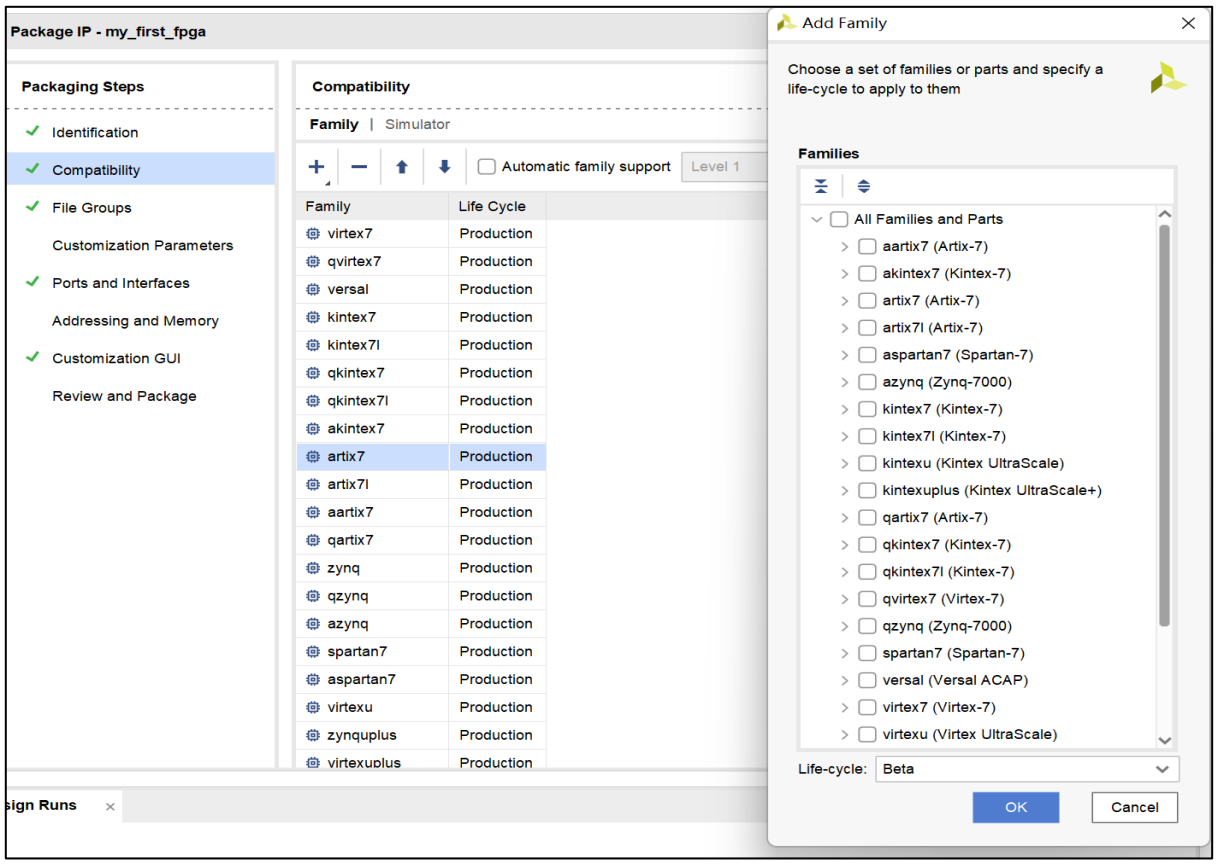

图0.45 设置IP核兼容的芯片家族

在设计电路的源程序中没有使用parameter来定义参数，因此在Customization parameter中无需修改参数的属性。

在Ports and Interfaces中保持缺省设置即可。

在Customization GUI中，可以看到封装后的模块外观图。

在Review and Package窗口中，可以看到创建IP核保存在：d:/my_design/lab0/lab0.srcs/NJUCS_NJUCS_XOR_1.0.zip文件中，可以修改保存路径核文件名，如图0.46所示。单击Package IP按钮，如成功创建IP，则弹出如图0.47所示的信息框。

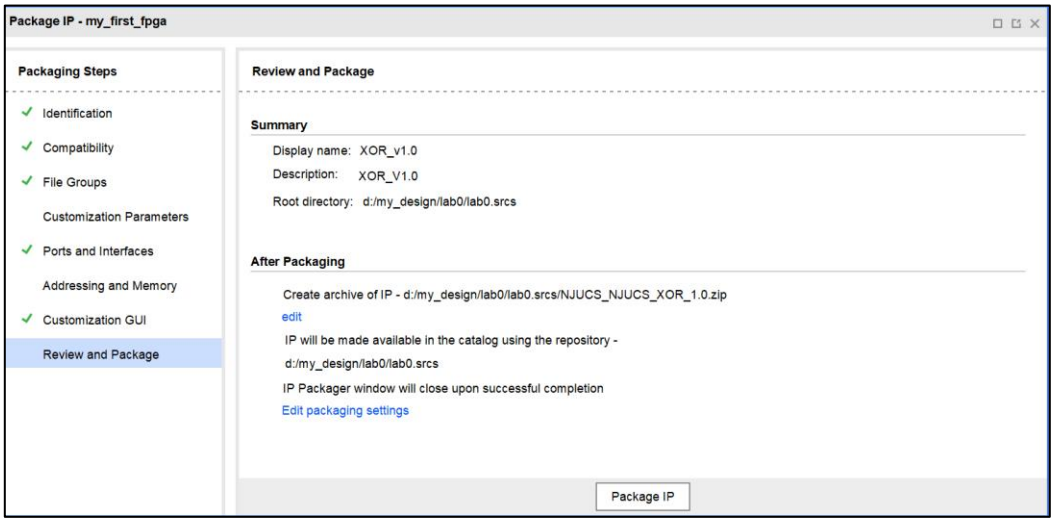

图0.46 Package对话框

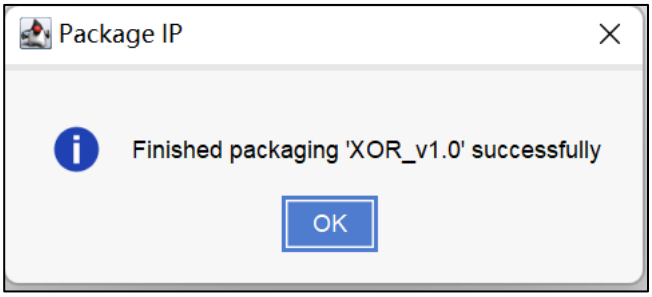

图0.47 封装成功提示框

为了以后使用方面，可以建立一个新文件夹，如MyIPCore，将封装后的压缩文件都拷贝到该文件夹下，并将其解压。

如果需要修改当前项目已经封装的IP核，可以项目流程导航窗口下，点击Edit Package IP菜单，在显示Package IP-XOR对话框中，可以点击执行Re-Package IP按钮完成IP核的封装修改，如图0.48所示。

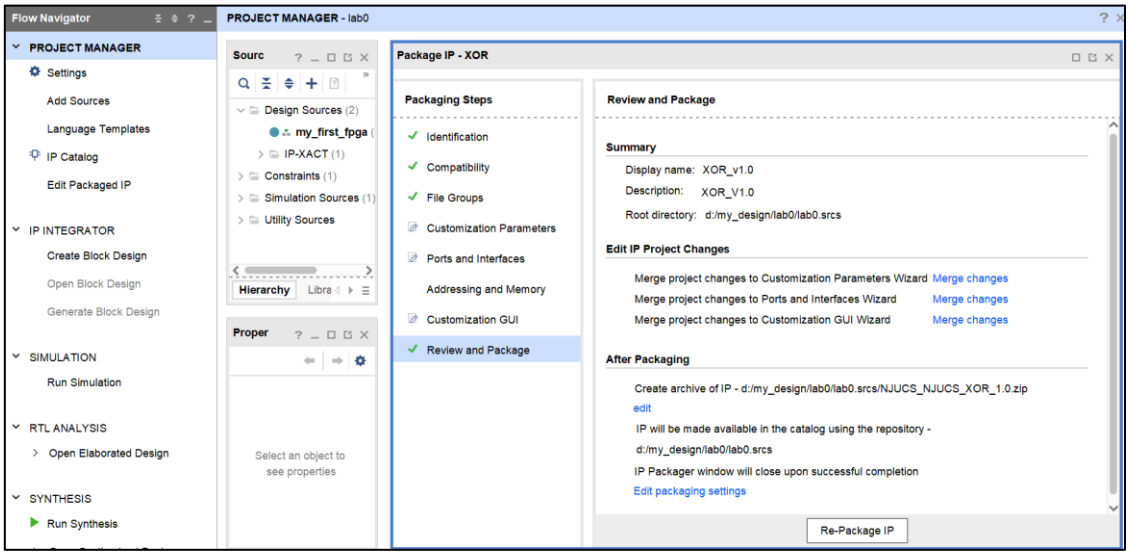

图0.48 编辑已封装的IP核

思考题

1. 如果将模块中数据位的宽度分别设置为4位、8位、16位和32位，则源代码需要做哪些改变？为什么？并对比不同数据位宽下的芯片资源占用情况。
2. 如果在源代码设计中使用参数定义了端口数量Port_Num和数据位的宽度WIDTH，则需要如何修改源代码？封装IP核时，如果定义端口数量为2-8，数据位的宽度为1-32，则在封装属性中如何设置？