ISE 14.7 下 LED 流水灯实验

黑金动力社区 2017-07-26

1 实验简介

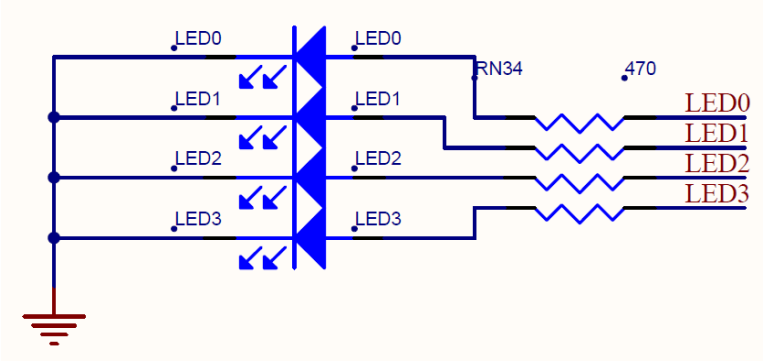
通过 LED 流水灯实验，介绍使用 ISE14.7 软件开发 FPGA 的基本流程，器件选择、设置、代码编写、编译、分配管脚、下载、程序 FLASH 固化、擦除等；同时也检验板上 LED 灯是否正常。

2 实验环境

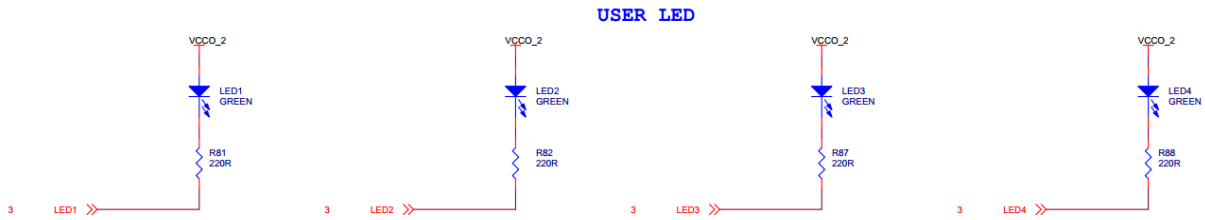
- Windows 7 SP1 64 位
- ISE Design Suite 14.7
- 黑金 FPGA 开发板（AX309 开发板、AX516 开发板、AX545 开发板）

3 实验原理

3.1 LED 硬件电路



AX309 开发板 LED 部分原理图



AX516/ AX545 开发板 LED 部分原理图

从上面的 LED 部分原理图可以看出，LED 电路有两个方式，AX309 开发板将 IO 经过一个电阻和 LED 串联接地，IO 输出高电平点亮 LED。AX516/AX545 开发板 通过串联电阻接入电源，IO 输出低电平点亮 LED。两种方式都是常见的方式，其中的串联电阻都是为了限制电流。

3.2 程序设计

FPGA 的设计中通常使用计数器来计时，对于 50Mhz 的系统时钟，一个时钟周期是 20ns，那么表示一秒需要 50000000 个时钟周期，如果一个时钟周期计数器累加一次，那么计数器从 0 到 49999999 正好是 50000000 个周期，就是 1 秒的时钟。

程序中定义了一个 32 位的计数器：

```
//Define the time counter
reg [31:0]      timer;
```

最大可以表示 4294967295，十六进制就是 FFFFFFFF，如果计数器到最大值，可以表示 85.89934592 秒。程序设计中是每隔 1 秒 LED 变化一次，一共消耗 4 秒做一个循环。

```
always@(posedge clk or negedge rst_n)
begin
    if (rst_n == 1'b0)
          timer <= 32'd0;
    else if (timer == 32'd199_999_999)
          timer <= 32'd0;
    else
          timer <= timer + 32'd1;
end
```

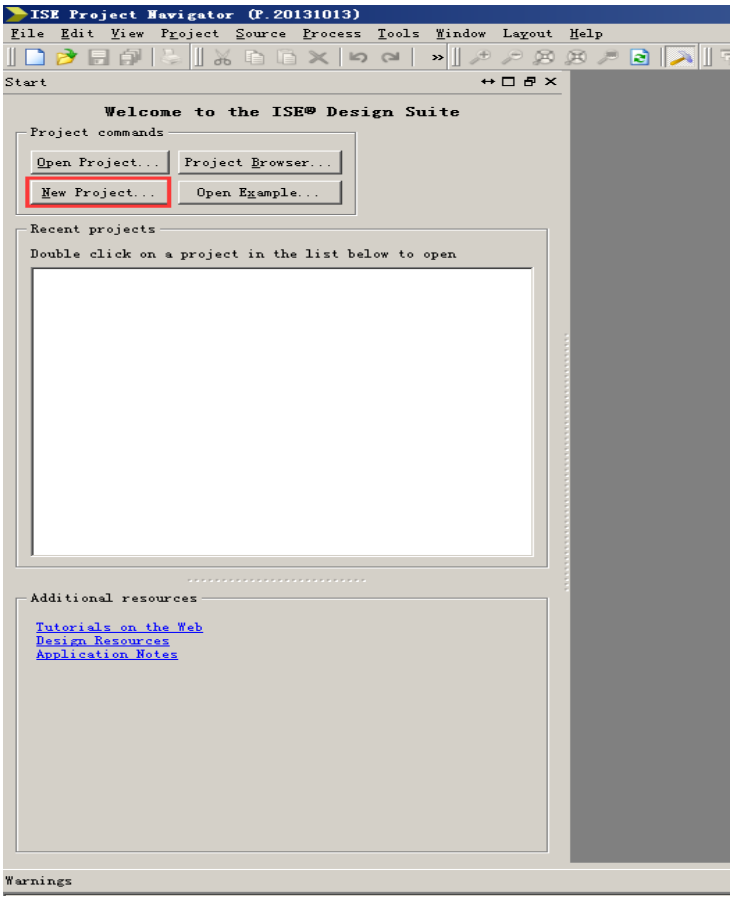
在第一秒、第二秒、第三秒、第四秒到来的时候分别改变 LED 的状态，其他时候都保持原来的值不变。

```
// LED control
always@(posedge clk or negedge rst_n)
```

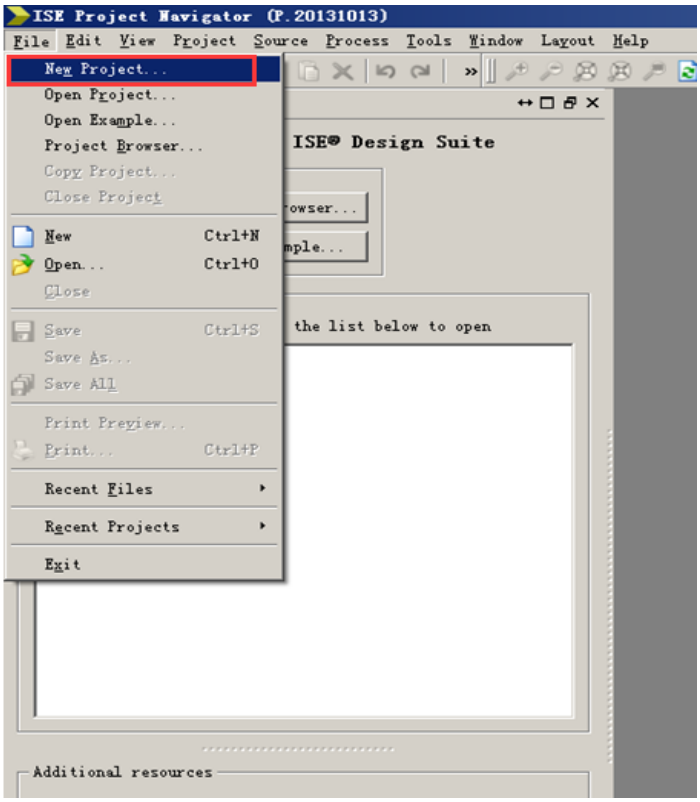
```
begin
    if (rst_n == 1'b0)
           led <= 4'b0000;
    else if (timer == 32'd49_999_999)
           led <= 4'b0001;
    else if (timer == 32'd99_999_999)
           led <= 4'b0010;
    else if (timer == 32'd149_999_999)
           led <= 4'b0100;
    else if (timer == 32'd199_999_999)
           led <= 4'b1000;
end
```

4 建立工程

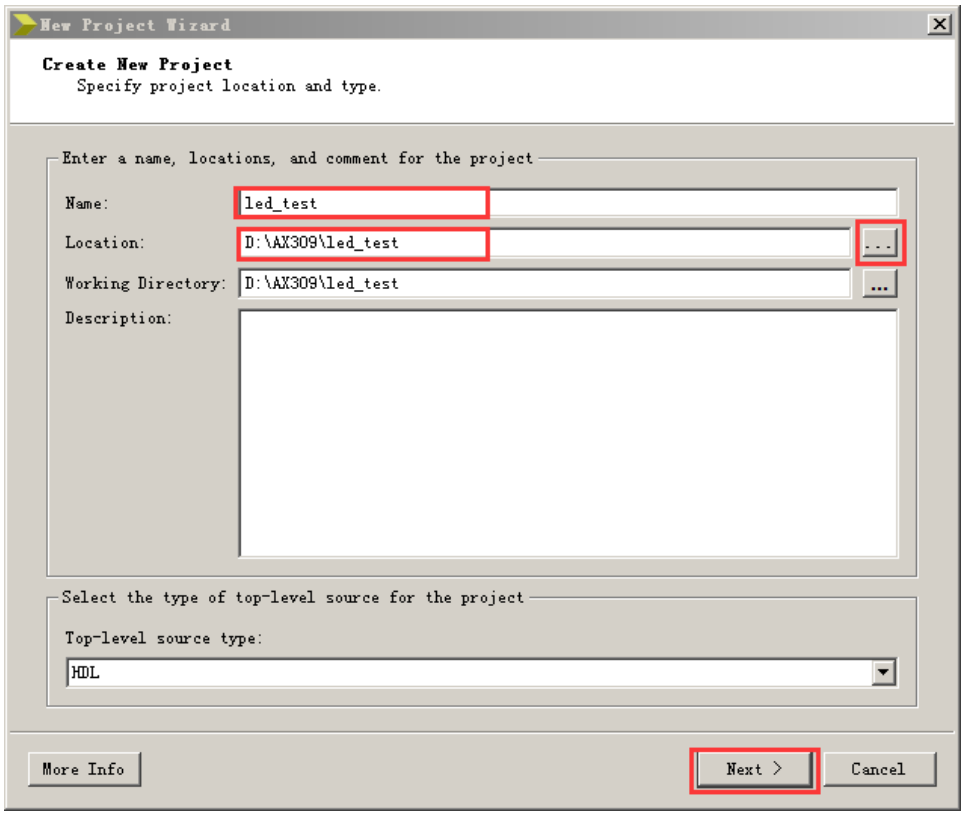
（1） 启动 ISE Design Suite 14.7 开发环境，选择 New Project 按钮选项



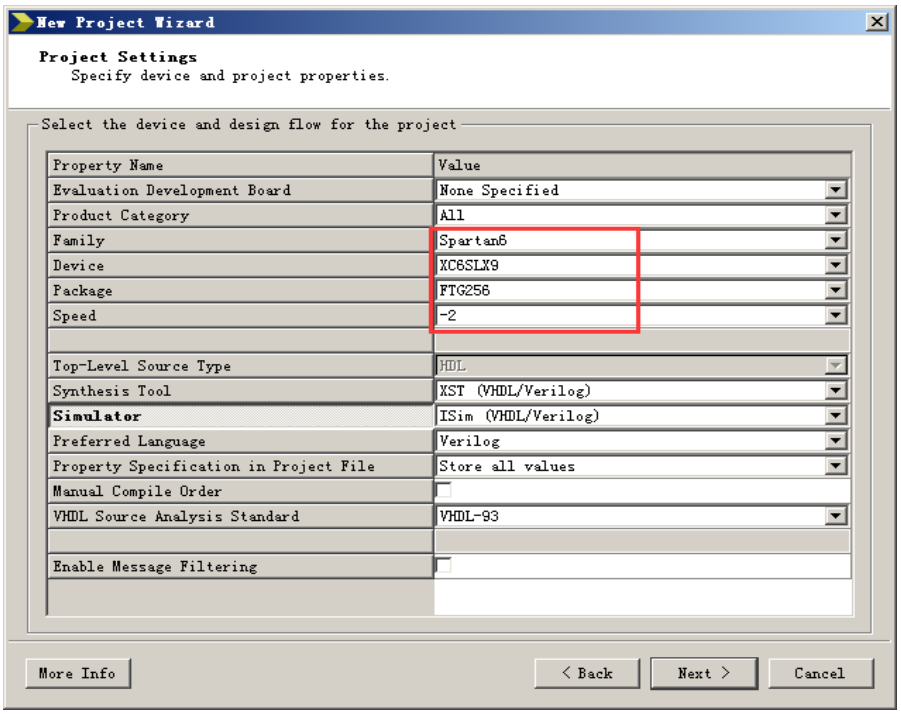
或者是选择菜单 File->New project



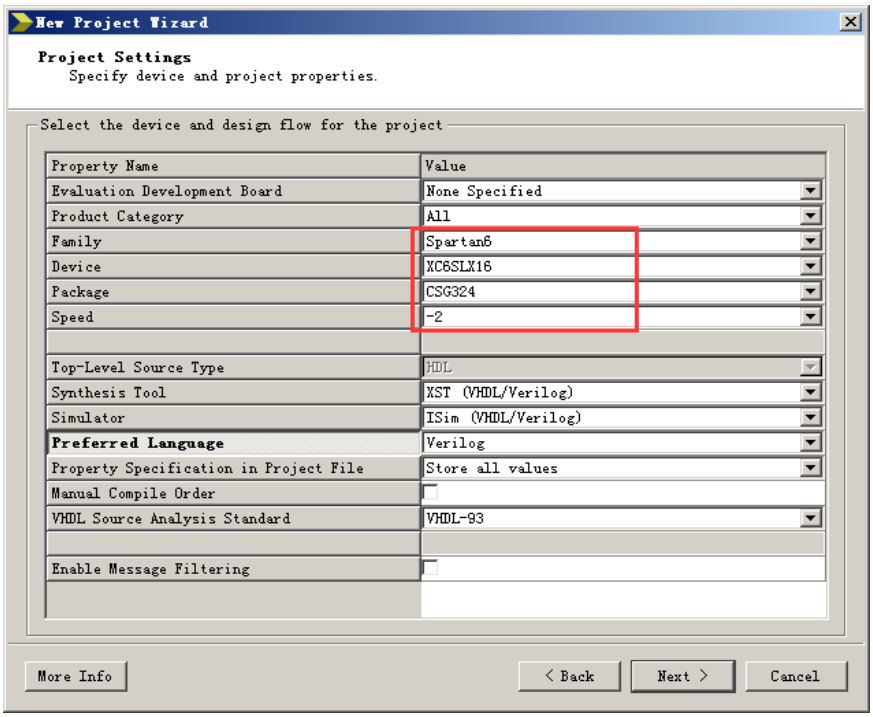
（2） 填写工程名称，添加工程保存位置，点击右边的按钮以选择工程保存位置，工作目录一般和保存位置一致，只需将工作位置添加后编译器会自动添加工作目录，无需手动添加，点击 Next（注意：在所有工程文件的建立和保存过程中，保存路径和文件名都不能有中文，空格或者其他特殊字符）；



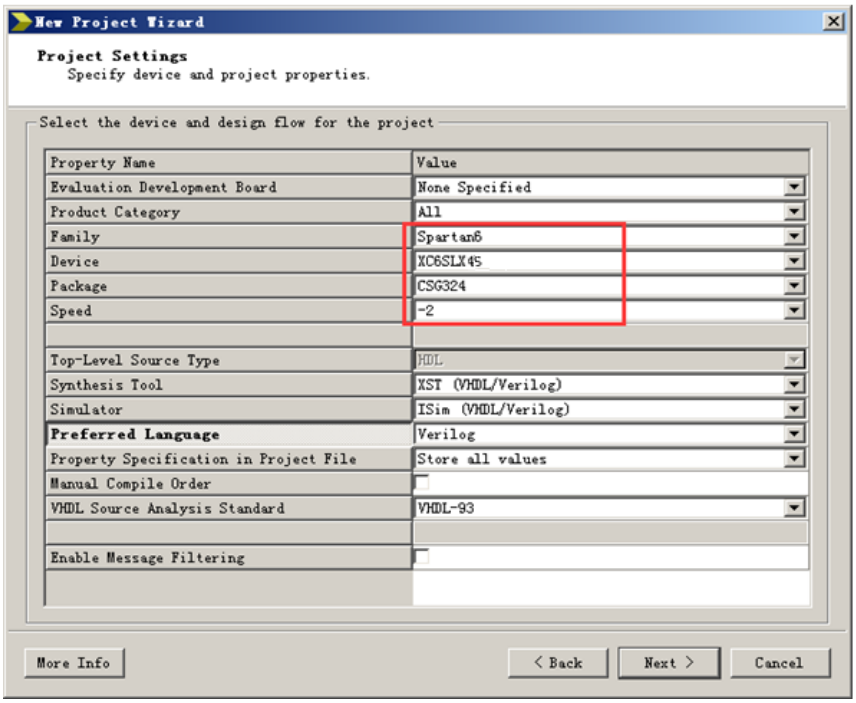
（3） 根据开发板上使用的芯片设置参数，在 AX309 开发板上设置如下：



AX516 开发板选择 XC6SLX16，AX545 开发板选择 XC6SLX45, 设置如下：



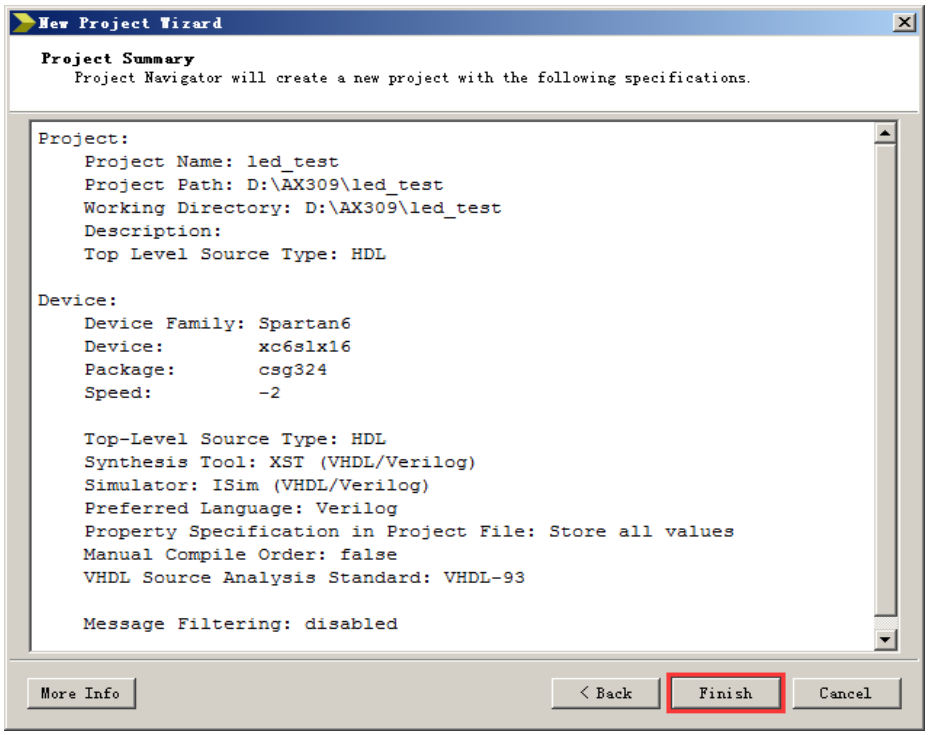
AX516 开发板



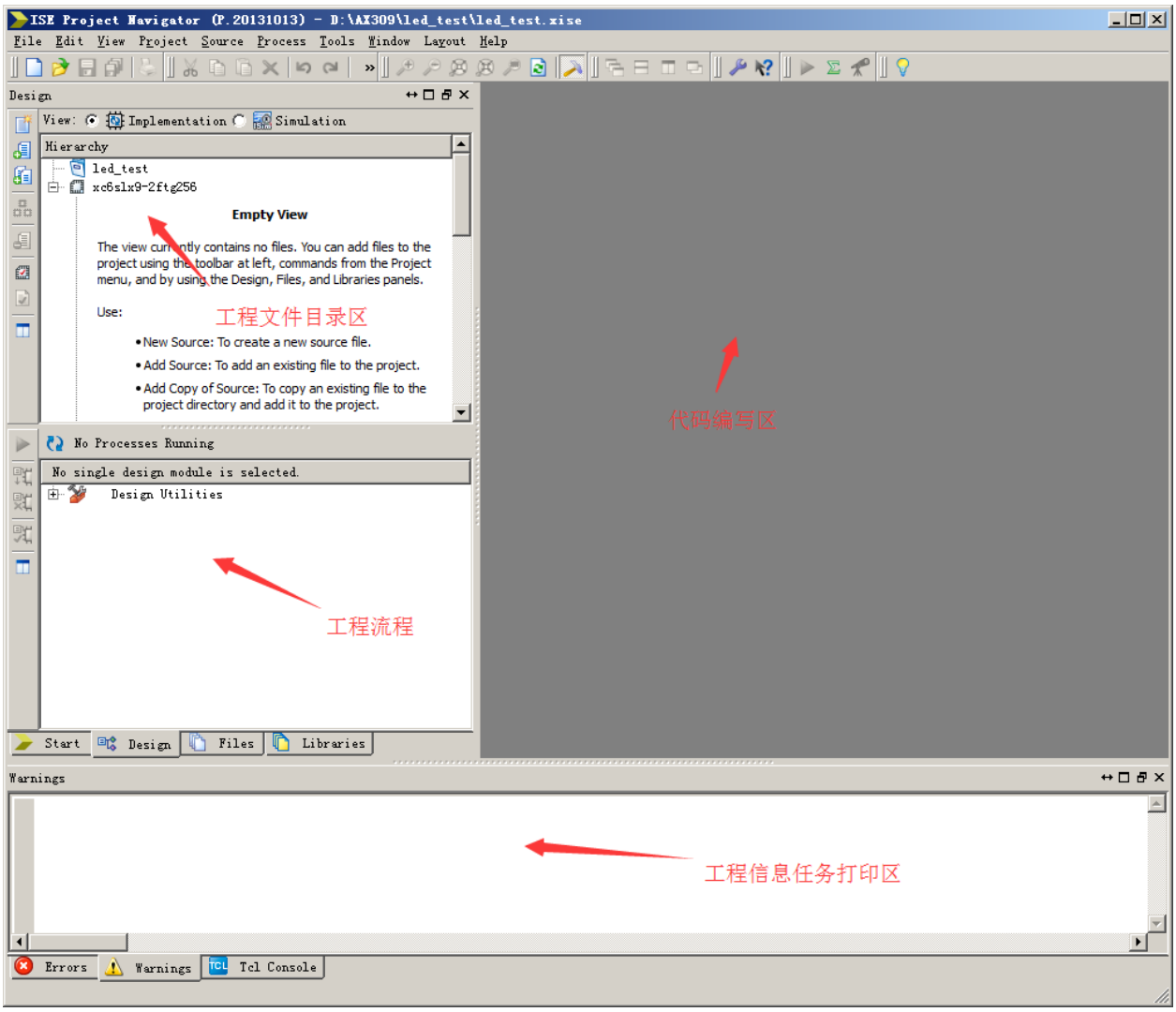
AX545 开发板

设置完成后点击 NEXT

（4） 完成工程导向，在确认信息与我们进行的设置无误后直接点击 Finish

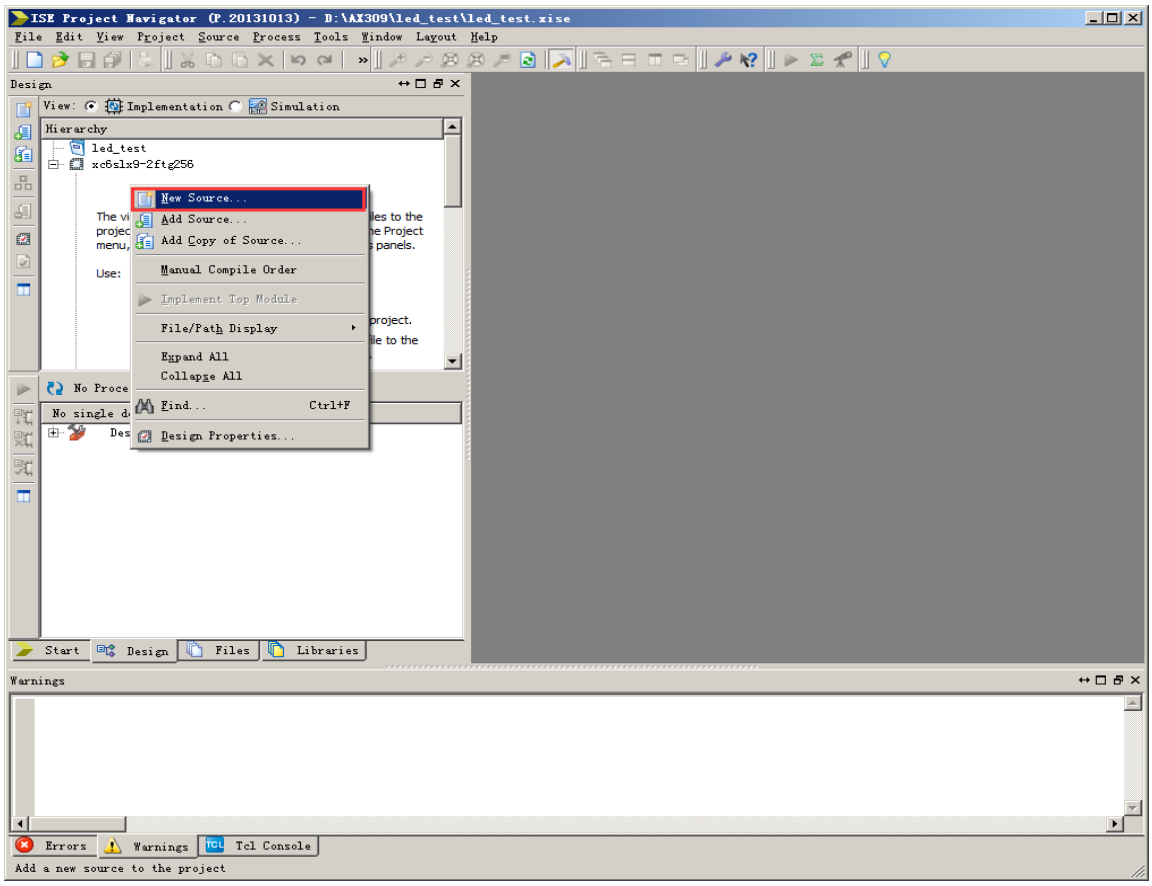


（5） 之后出现如下界面：

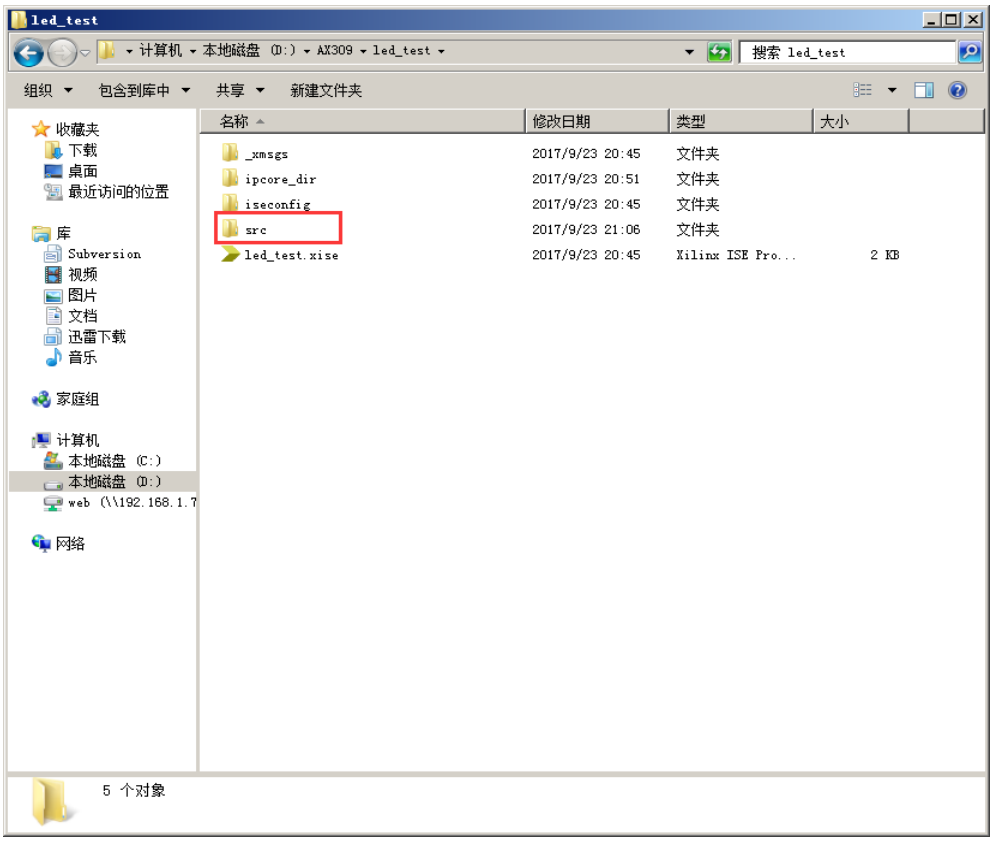


5 编写代码

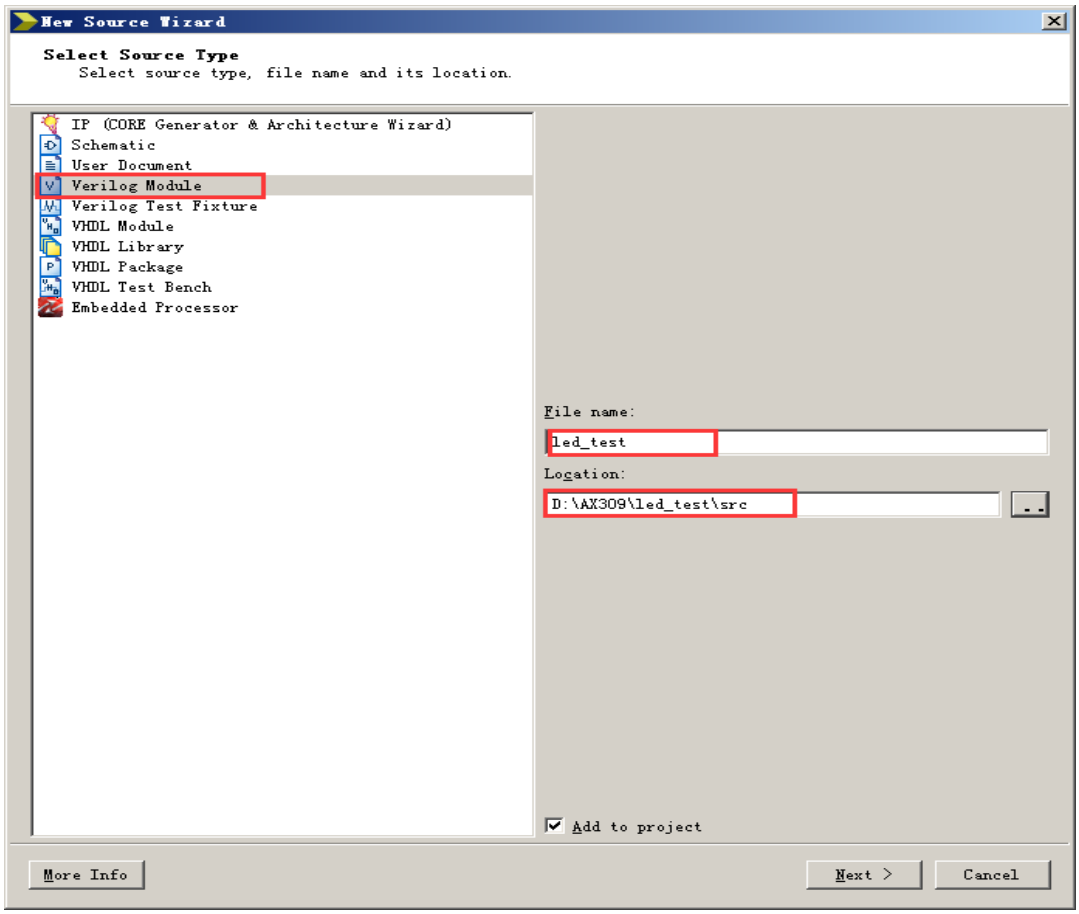
（1） 新建 Verilog HDL 文件，在工程文件目录区单击右键，选择 New Source



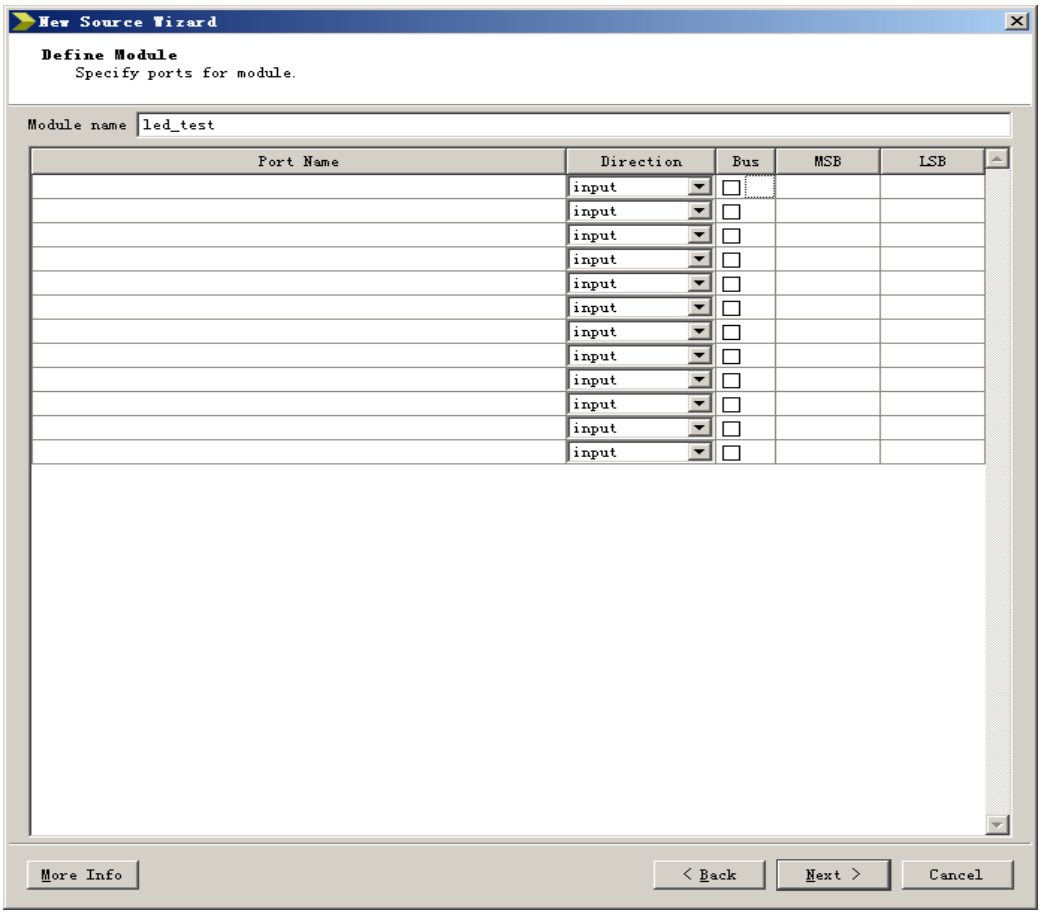
（2） 我们先找到工程的保存位置新建一个名为 src 的源文件，我们将所有的工程源文件模块都保存在这个文件夹里面：



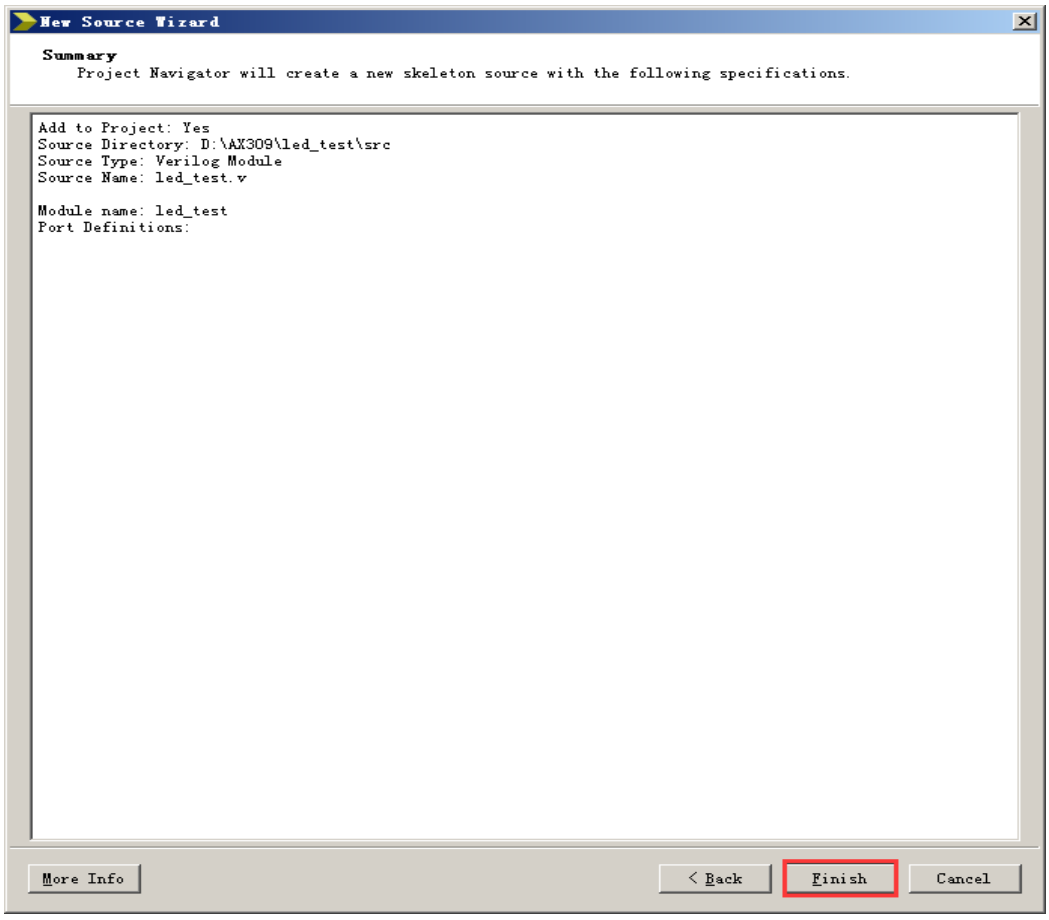
（3） 选择 Verilog module 文件，填写模块文件的名称，将模块的保存路径切换为工程目录下的 src 文件夹，点击 Next。



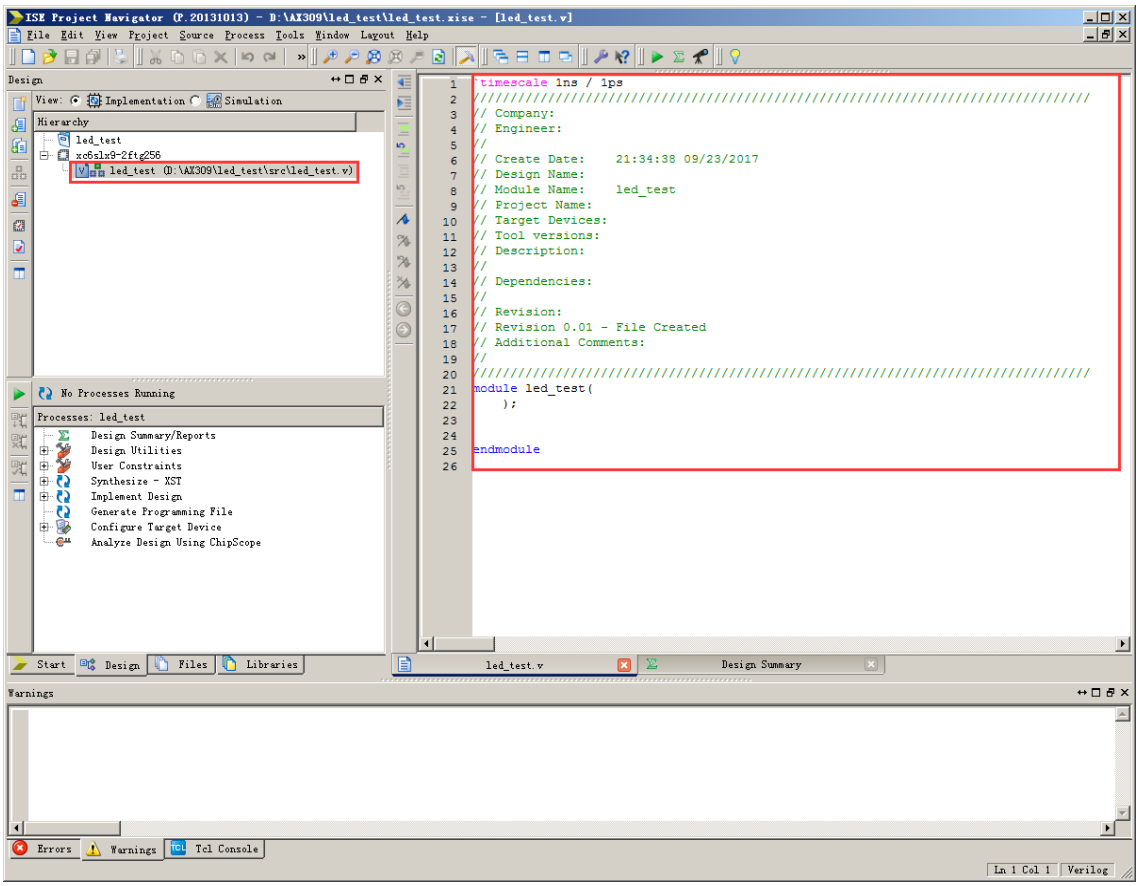
（4） 保持默认直接点击 Next



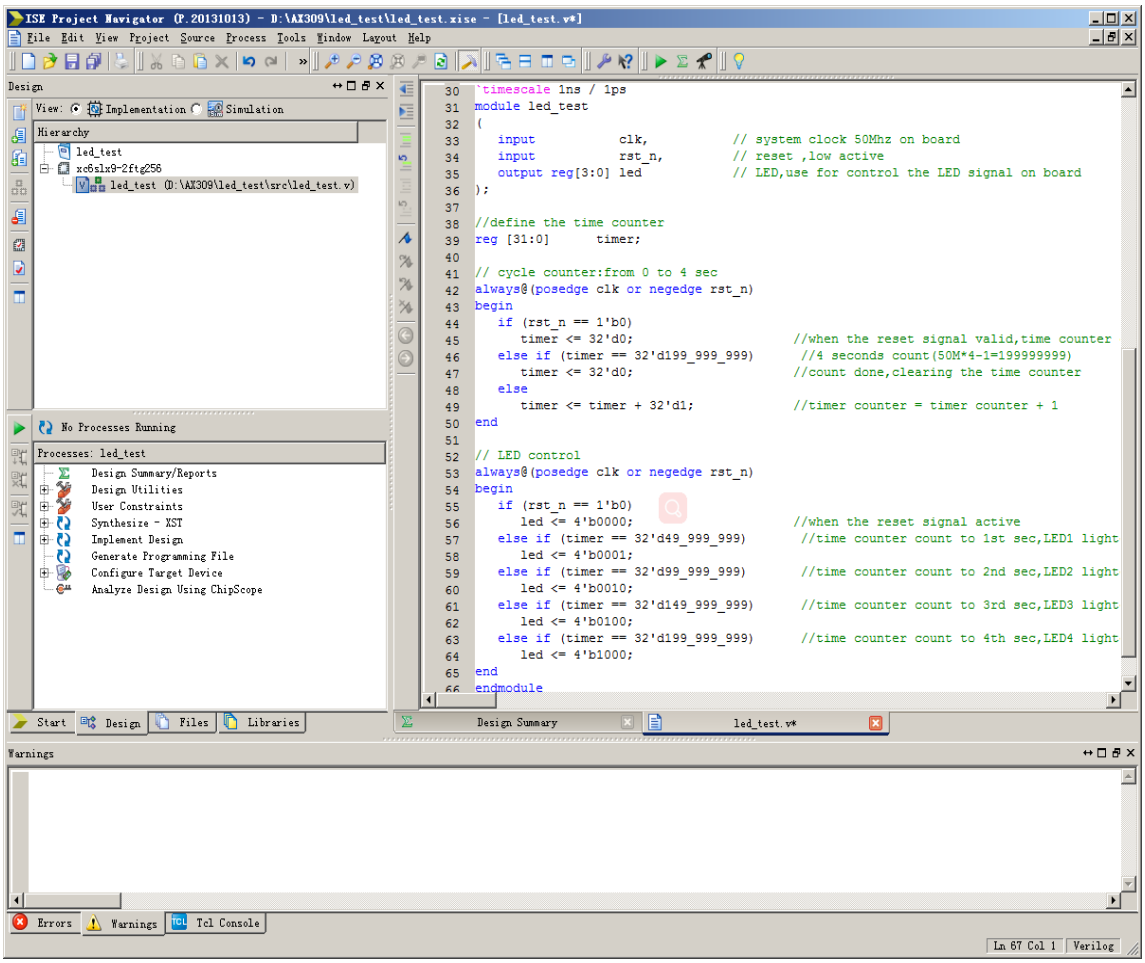
（5） 点击 Finish 完成模块建立。



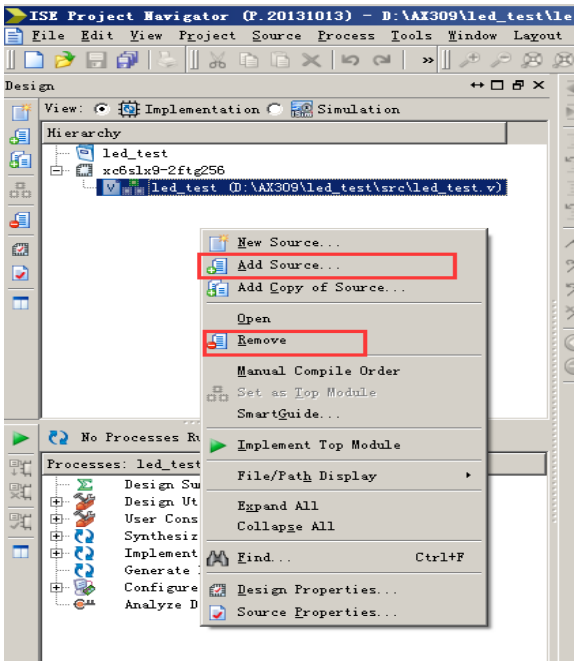
（6）我们可以看到在工程文件目录区已经有了我们建立的 led_test 的源文件，右边的代码编写区也有了 Xilinx 官方提供的代码编写模板



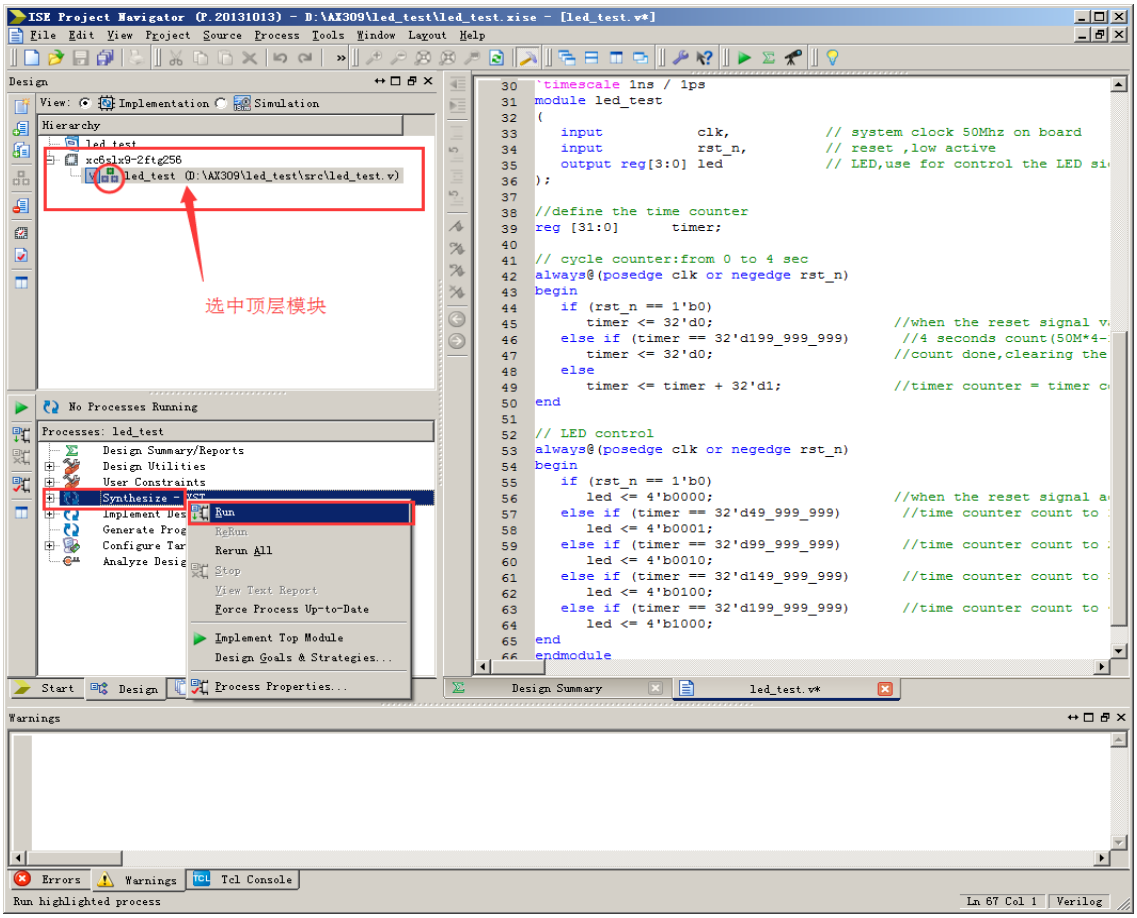
在这里代码不是我们本实验的重点，直接给出代码，不做重点介绍。



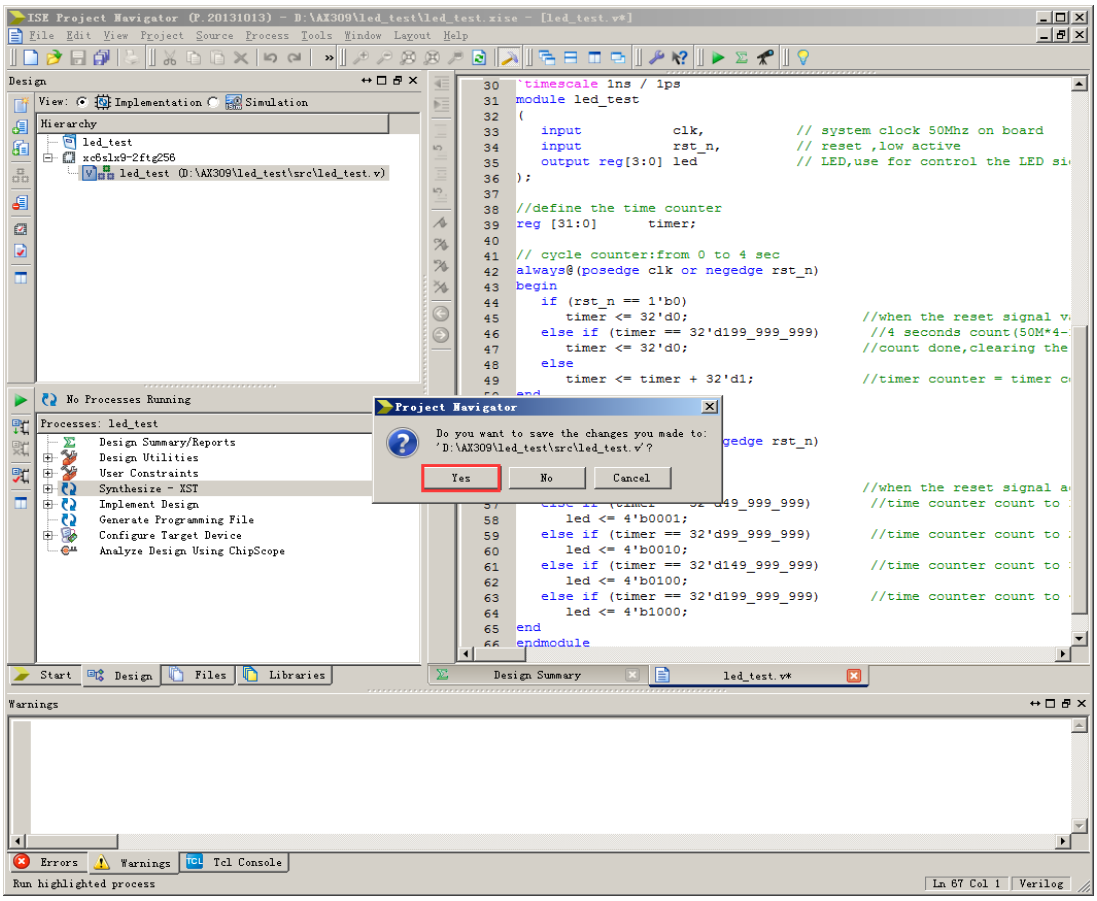
（7）文件的操作。我们可以在工程文件目录区选中某个已经新建的模块，右键单击进行删除，或者添加新的模块的操作：



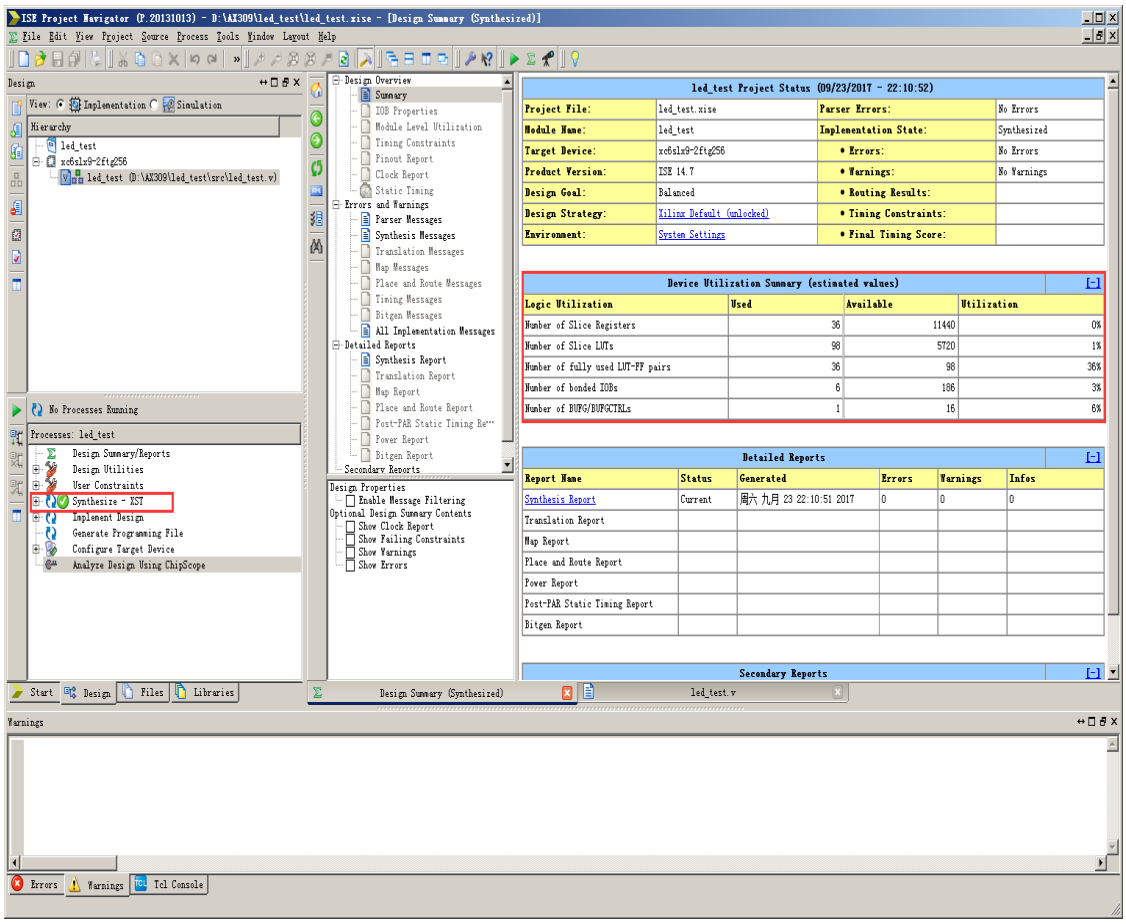
（8）编译工程。先在工程文件目录区选中顶层模块，然后在工程流程区右键选中 Synthesize - XST,单击 run。



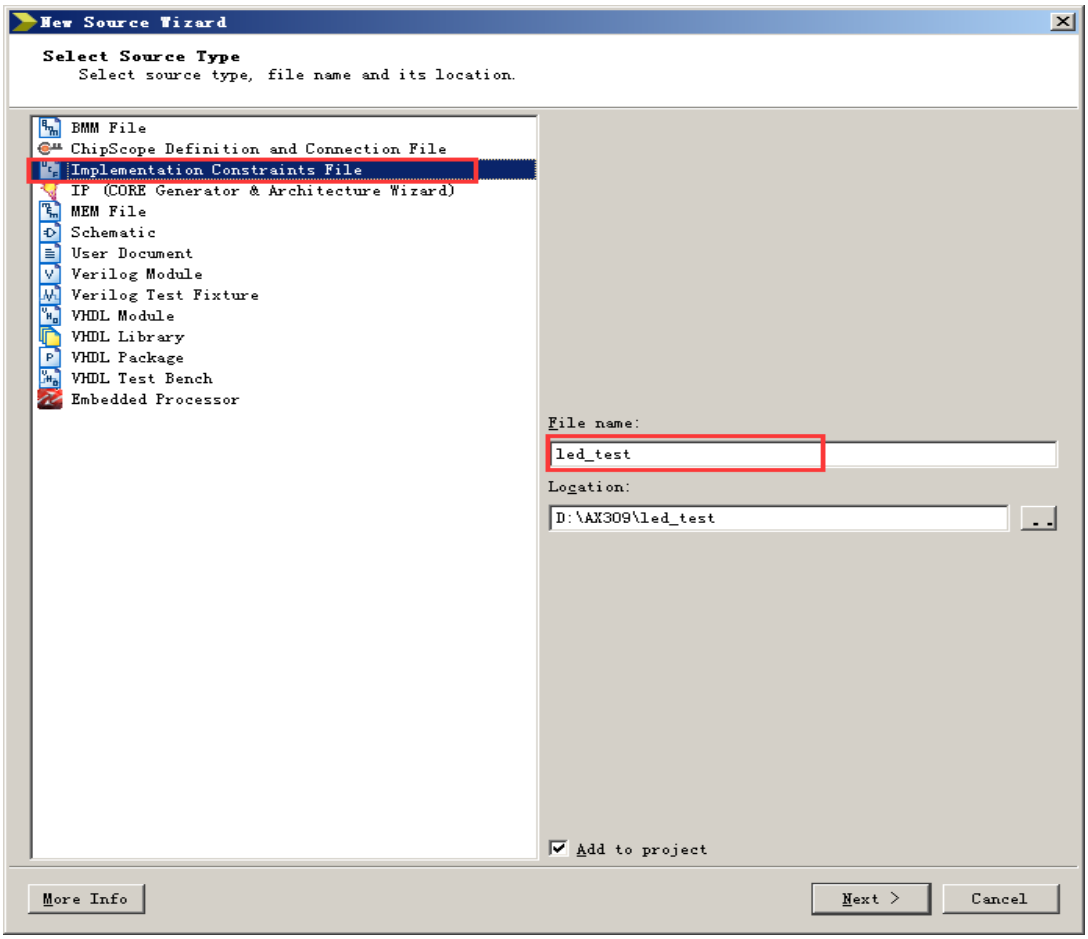
（9）选择 Yes 保存



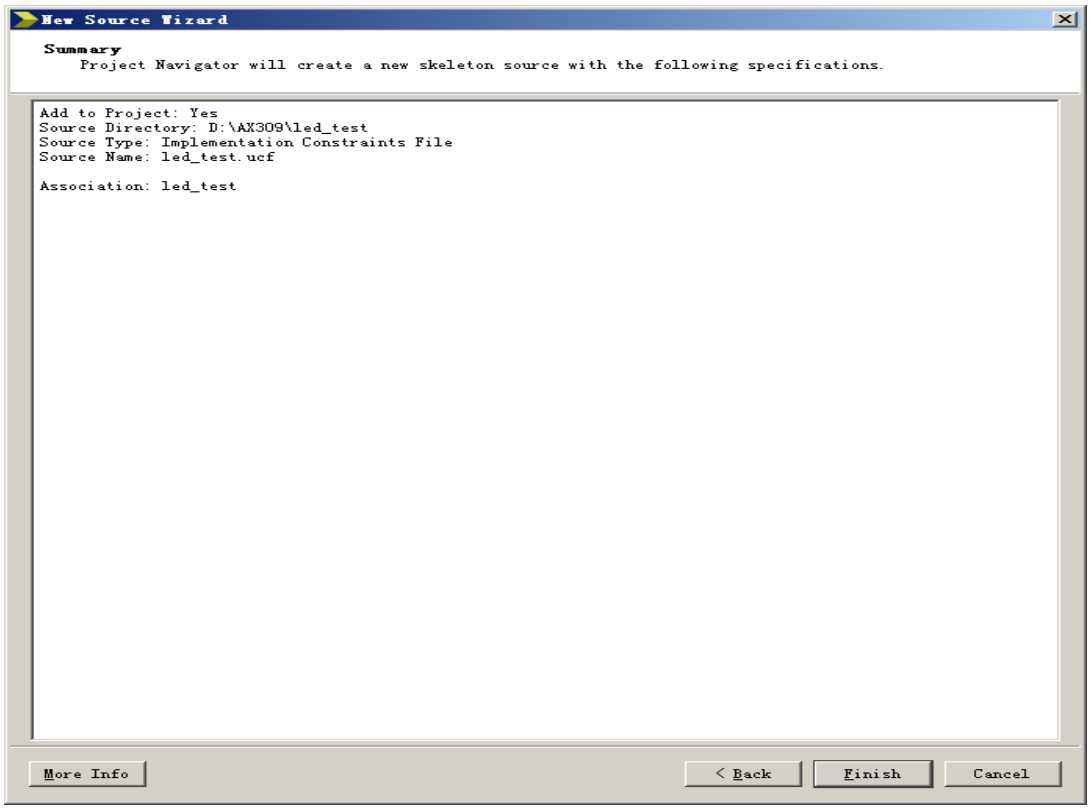
（10） 如果代码没有语法错误则在 Synthesize - XST 菜单前会出现运行成功标志，同时右边的代码编辑区会出现代码的编译信息，包括寄存器和查找表，Buffer 等资源的消耗情况，如下所示：



（11） 管脚信息分配。在工程文件目录区单击鼠标右键选择 New Source->Implementation Constraints File，文件名保存于工程名一致为“led_test”。


New Source Wizard
Select Source Type
Select source type, file name and its location.
BMM File
ChipScope Definition and Connection File
Implementation Constraints File
IP (CORE Generator & Architecture Wizard)
MEM File
Schematic
User Document
Verilog Module
Verilog Test Fixture
VHDL Module
VHDL Library
VHDL Package
VHDL Test Bench
Embedded Processor
File name:
led_test
Location:
D:\AX309\led_test
Add to project
More Info
Next >
Cancel

（12） 点击 Next。

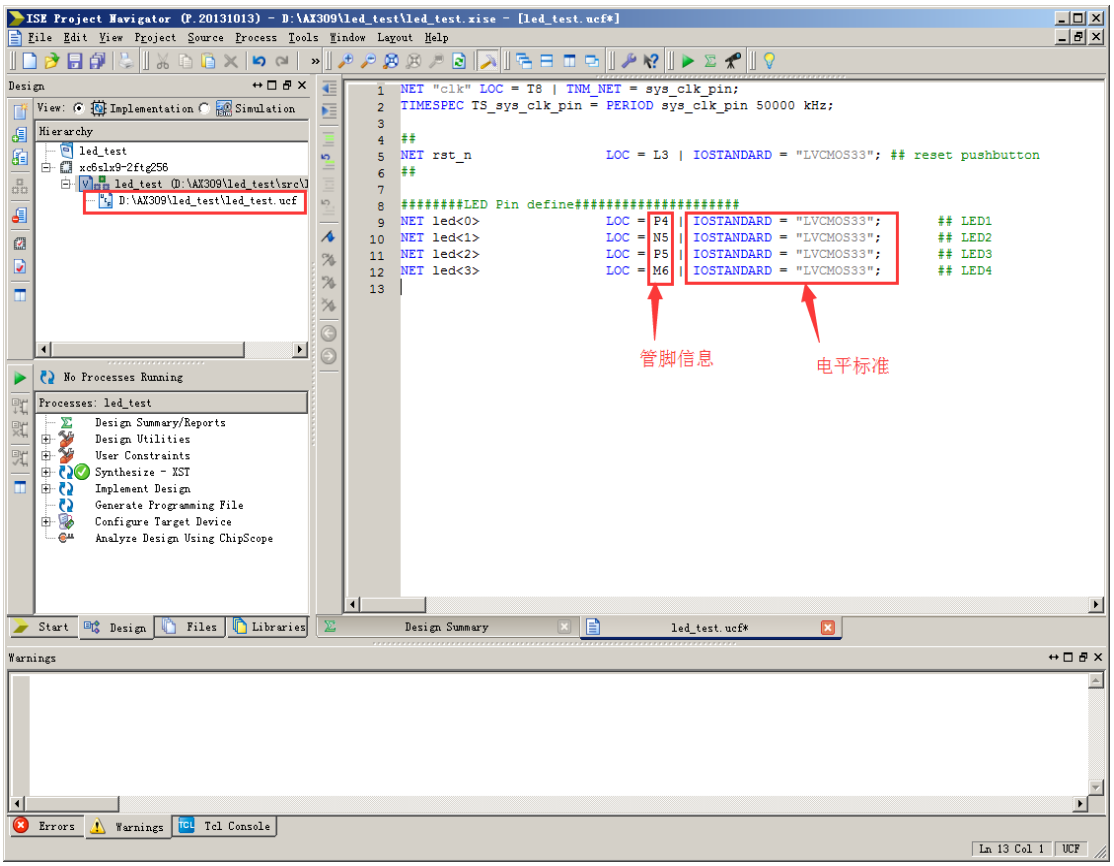


（13） 点击 Finish 进入管脚分配界面。可以看到在左边的工程文件目录区已经有了后缀名为“.ucf” 管脚分配和约束文件。我们已经在配套的资料里附带了开发板上所有管脚的分配信息，文件位置在一个名为“UCF”的文件夹下，在使用时可以直接复制。

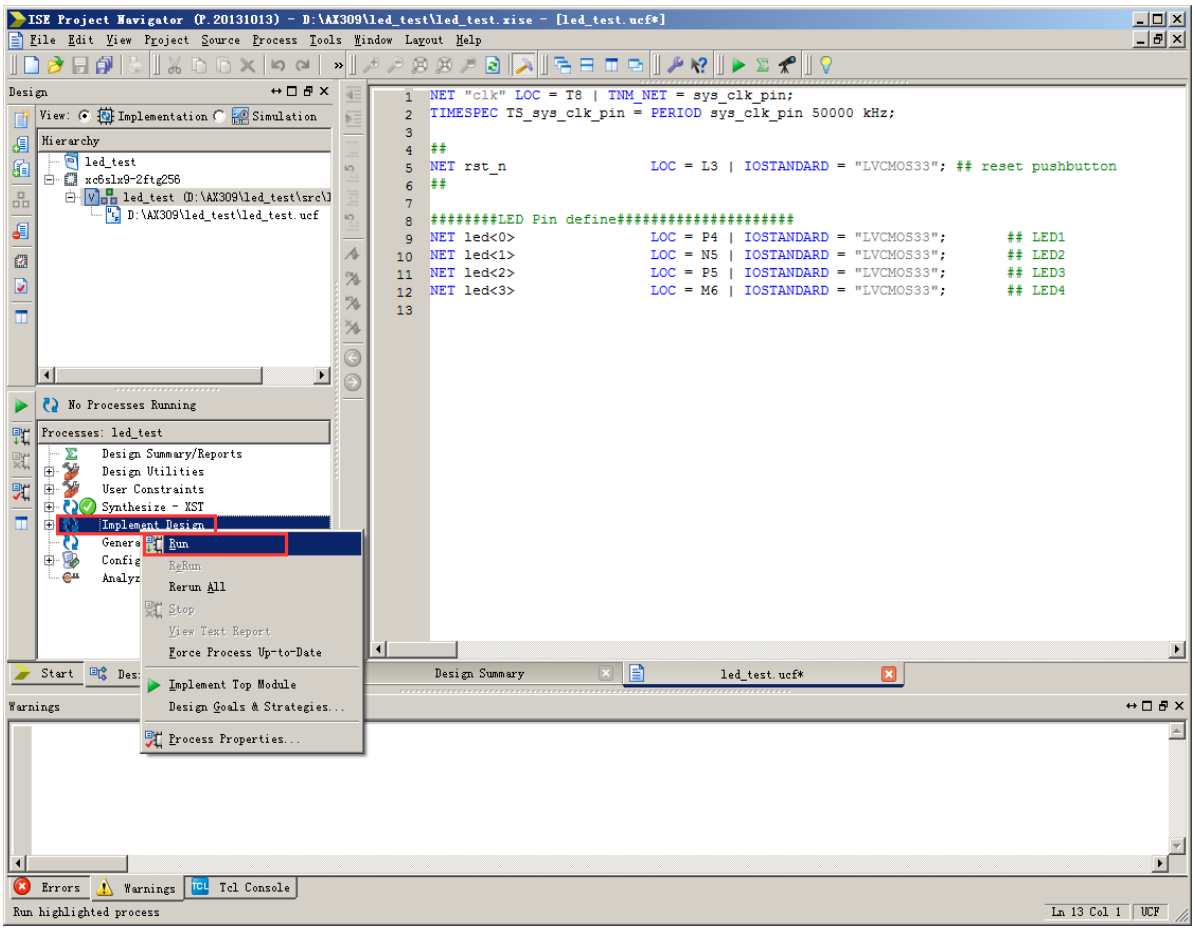
（14） 下面是一种最基本最简单的 UCF 文件编写方法：

NET “端口名称” LOC = 引脚编号 | IOSTANDARD = “电压” ;

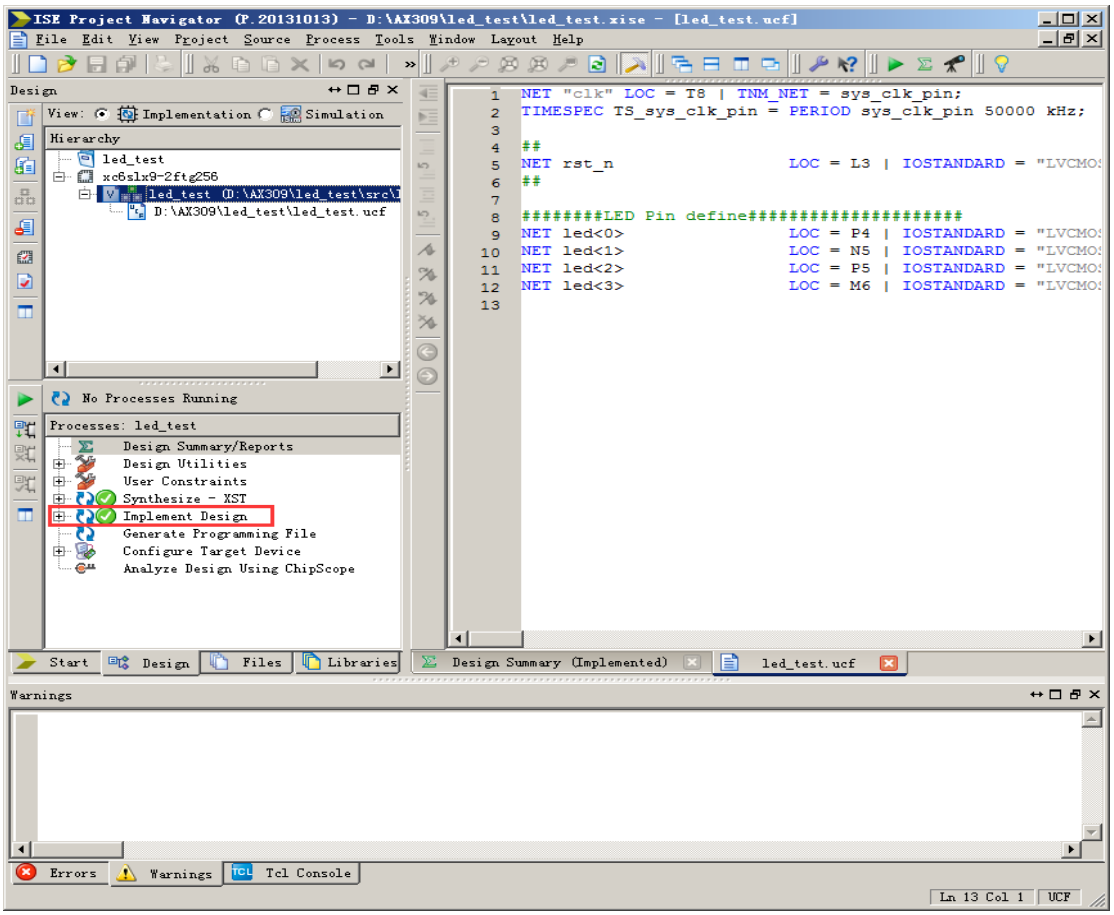
本例程中我们使用到的管脚有 50Mhz 的系统全局时钟：“clk” 全局复位信号“rst_n”和四个 led 灯“led<0~3>”，所以约束文件也只需要写出这几个管脚的分配信息就可以了，另外将除系统时钟之外的所有管脚的电平标准约束为 3.3V 以满足要求，这个设置是和 FPGA 硬件设计的 bank 电压有关，黑金的大部分开发板的 IO BNAK 电压为 3.3V，所以设置为“3.3-V LVTTL”。输出电压和设置没有关系，如果 BANK 电压是 3.3V，你这里设置 2.5V，也不会改变 IO 输出的电压幅度。当然在其他应用场合我们也可以根据不同的应用要求来分配不同的电平标准：



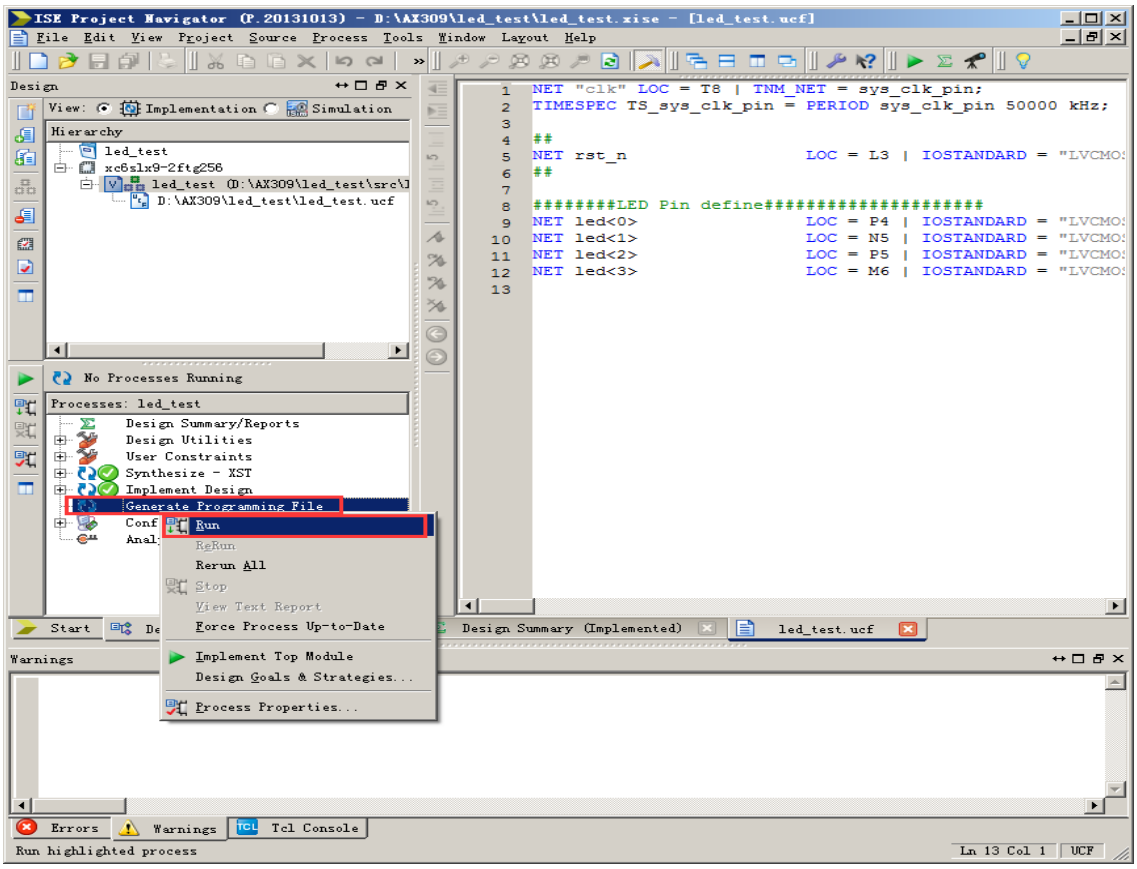
（15） 布局布线。回到工程流程区，右键单击选项“Implement Design”,点击 run 后编译器会自动根据管脚信息布局布线：



（16） 如果管脚分配和约束信息没有错误的话运行结束会出现如下界面：

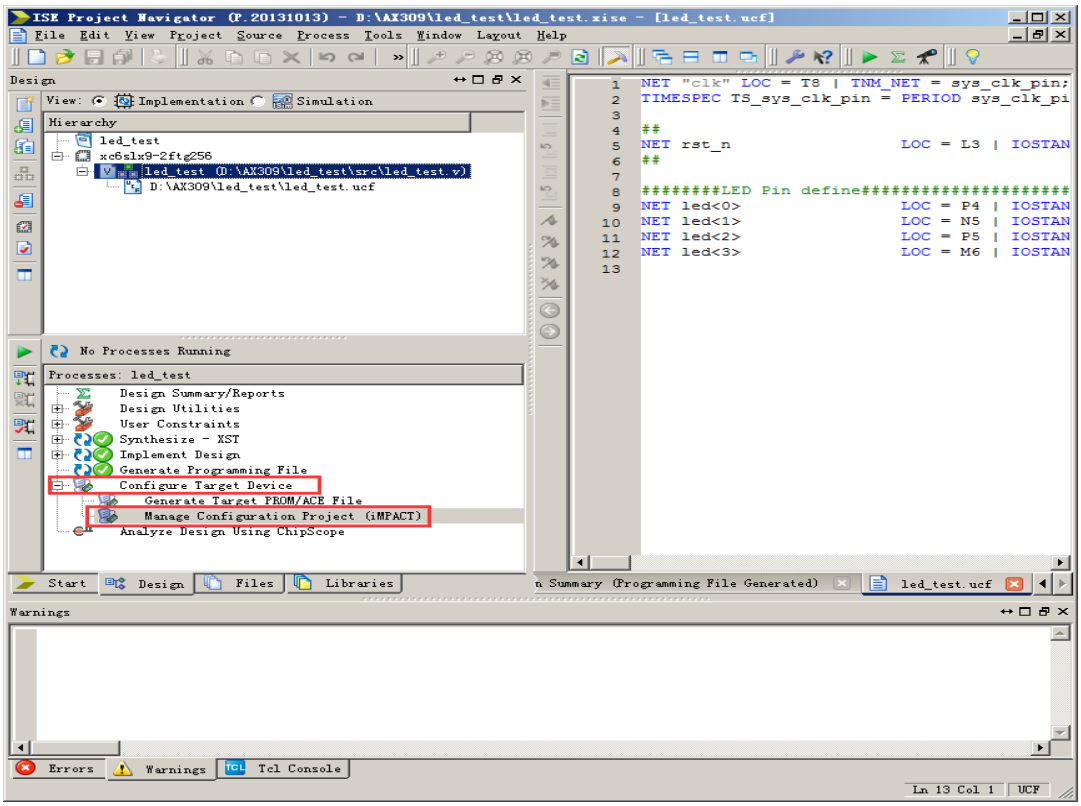


（17） 生成可执行文件，可执行文件是编译器根据编写的程序和管脚信息还有约束信息所生成的后缀名为“.bit”的文件。这个文件可以通过 JTAG 方式下载到 FPGA 运行，但不能直接固化到 Flash。在工程流程区，右键选中“Generate Programming File”，选择 run，如果前面操作都没有问题，那这一步也不会有问题，待生成成功后在“Generate Programming File”选项前也会出现一个绿色的勾表示生成 成功。

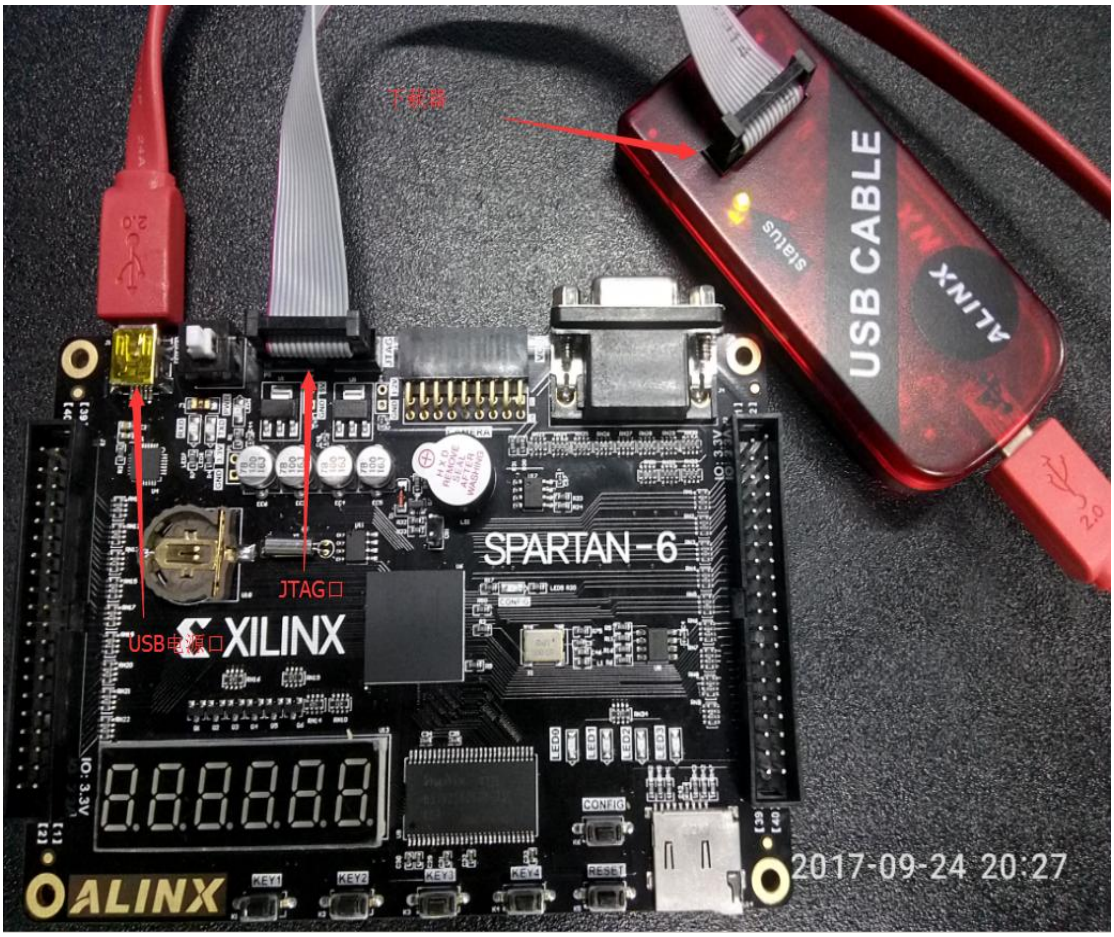


6 下载

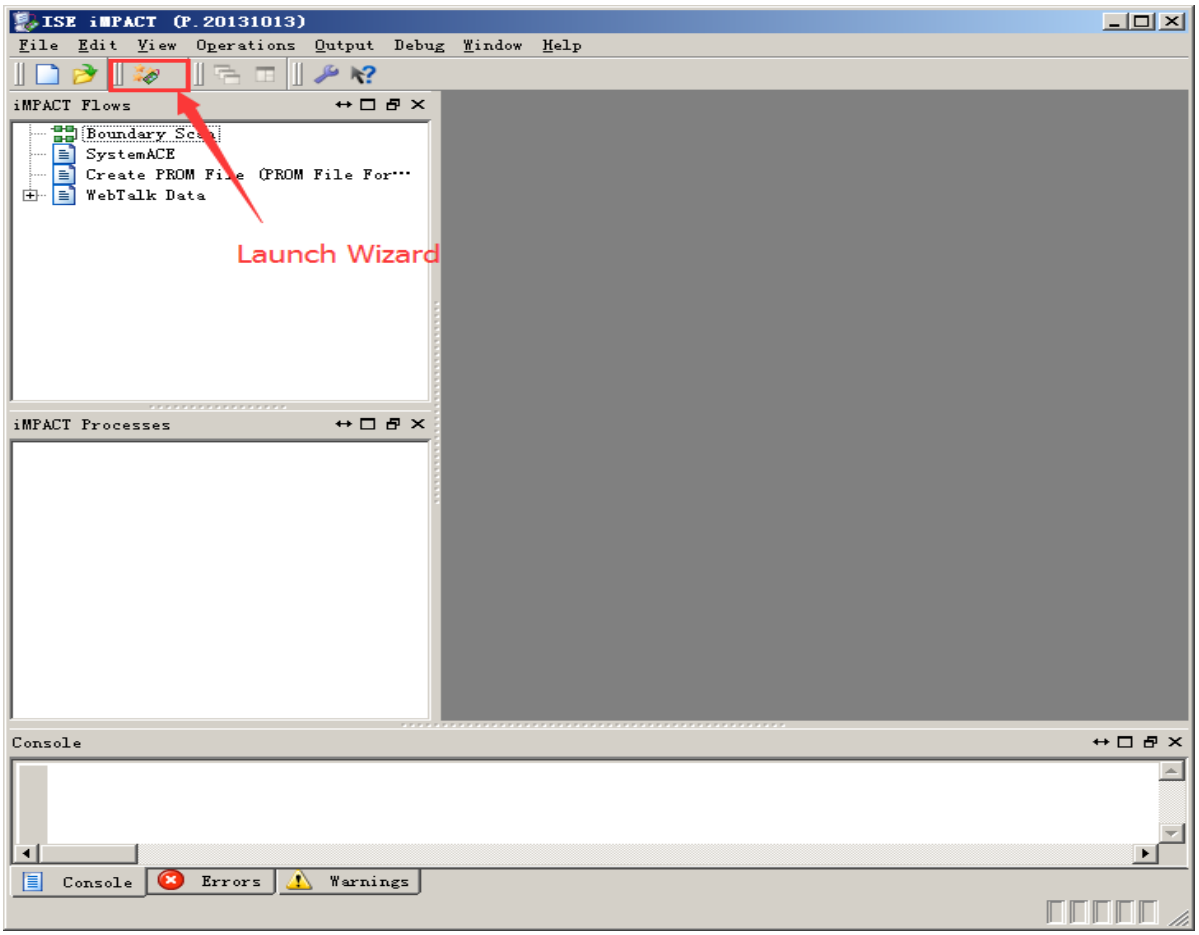
（1）在这里我们选择用 iMPACT 的方式下载 bit 文件到 FPGA。在工程流程区点开“Configure Target Device”选项（点击该选项前的加号按钮），选中"Manage Configuration Project(iMPACT)",双击或是右键单击选择 run：



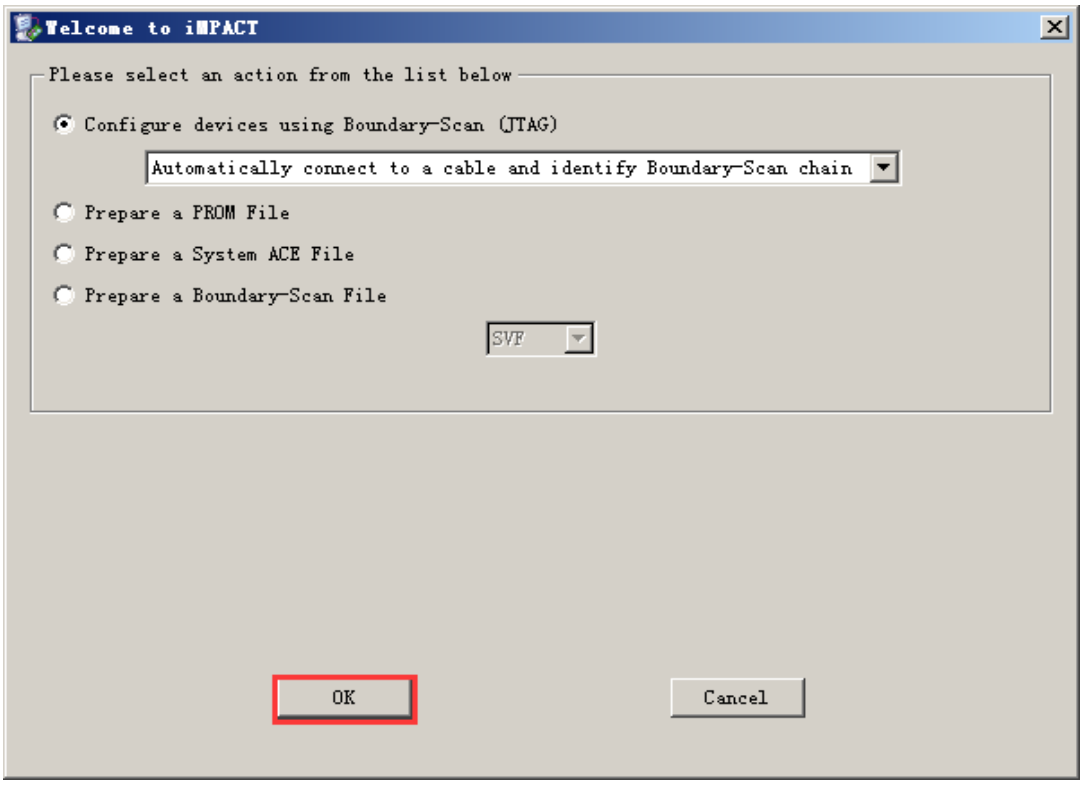
（2）之后出现如下界面，此时将开发板的电源线一端连接开发板的 PWR 口，另一端与电源连接。将下载器与连接线正确连接，一端接开发板的 JTAG 口，一端接 PC 端的 USB 口，一定要确保连接正确且连接成功。



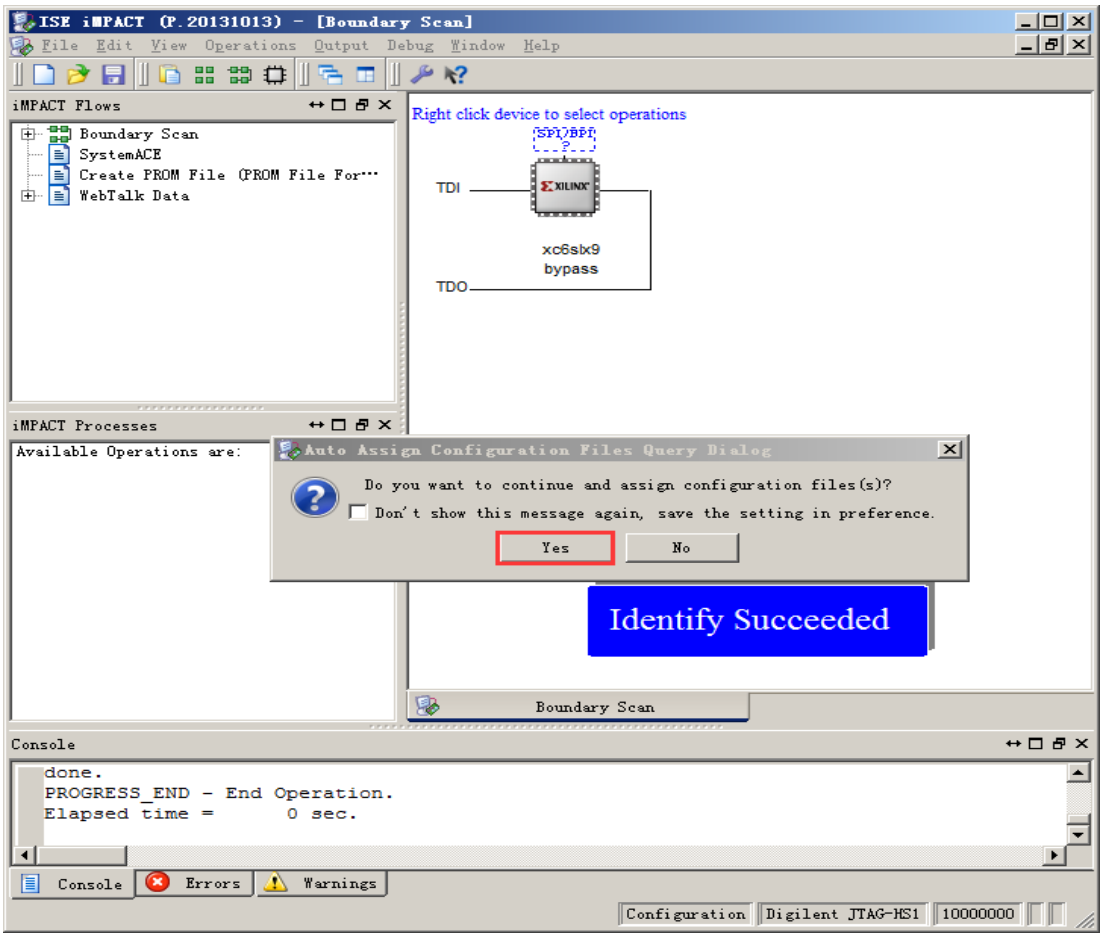
（3）将开发板上电，点击菜单上的“Launch Wizard”。



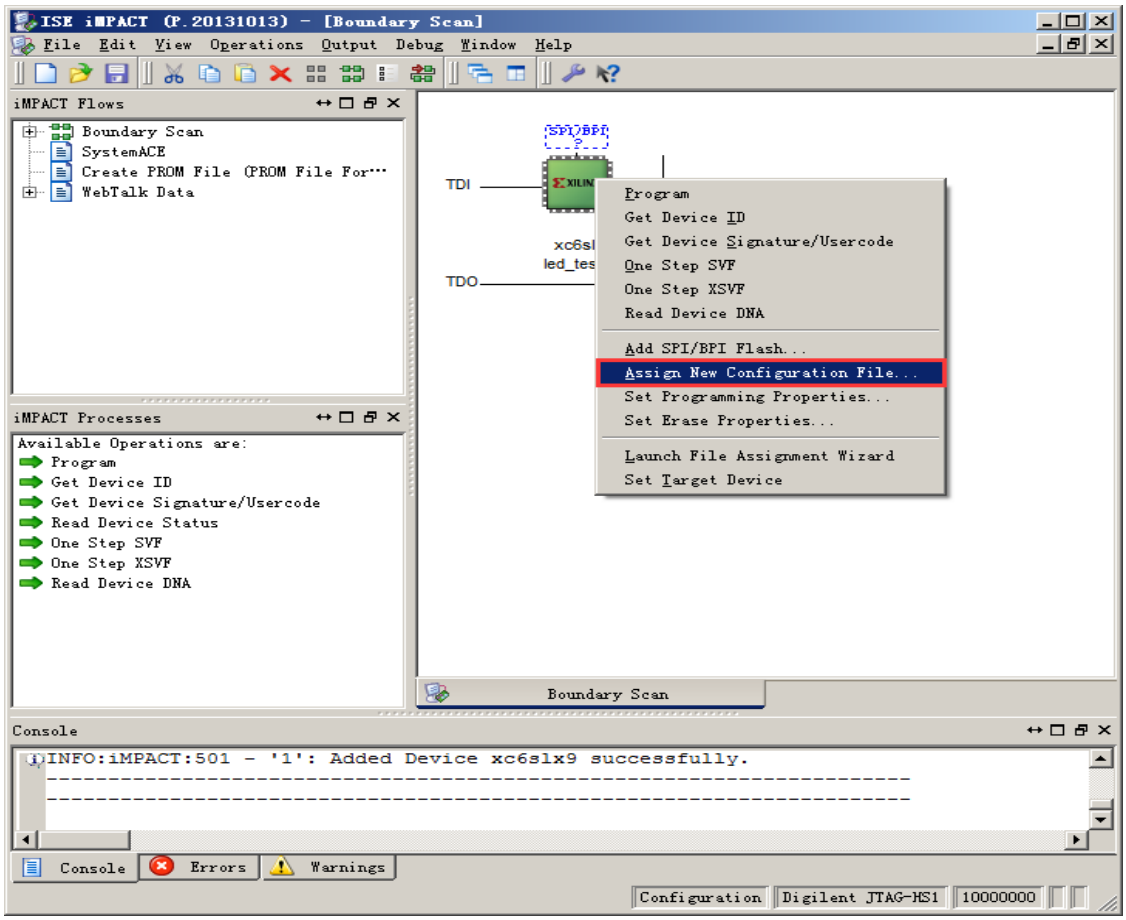
（4）出现如下界面，我们默认选择 JTAG 下载，直接点击 OK 进入下一步。



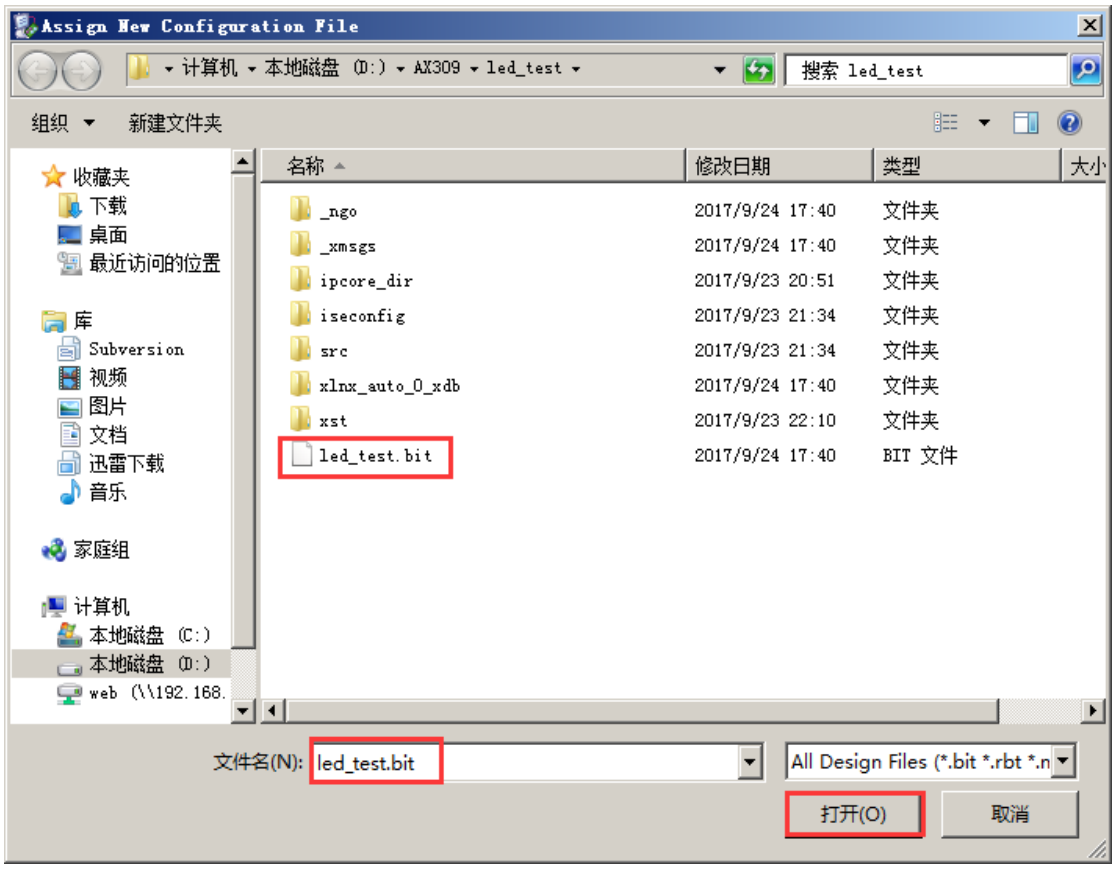
（5）选择 Yes 添加 bit 文件。



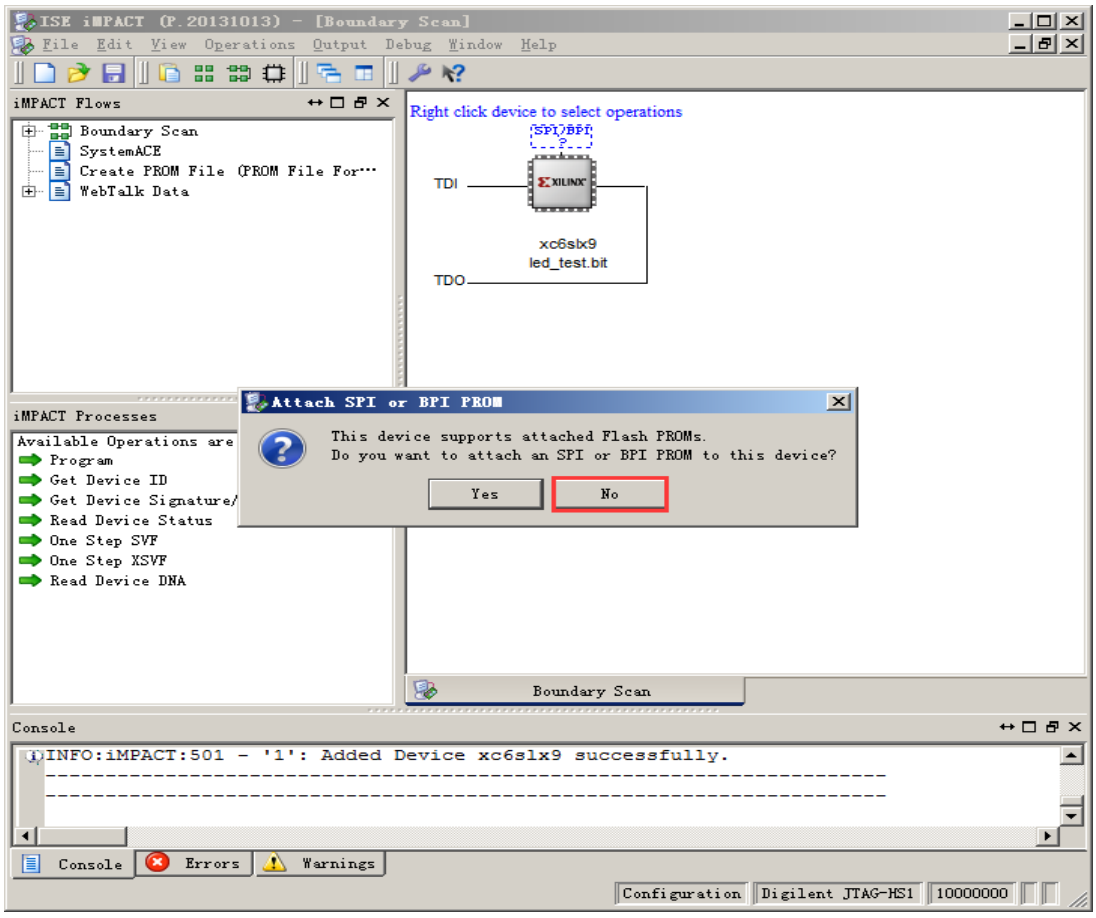
（6）如果在上一步中勾选了“Don't show this message again”的选项框，那在以后的操作中都不会再出现，这种情况，我们就需要换种操作方式，选中在界面中出现的芯片图标，右键单击，选择“Assign New Configuration Flie”



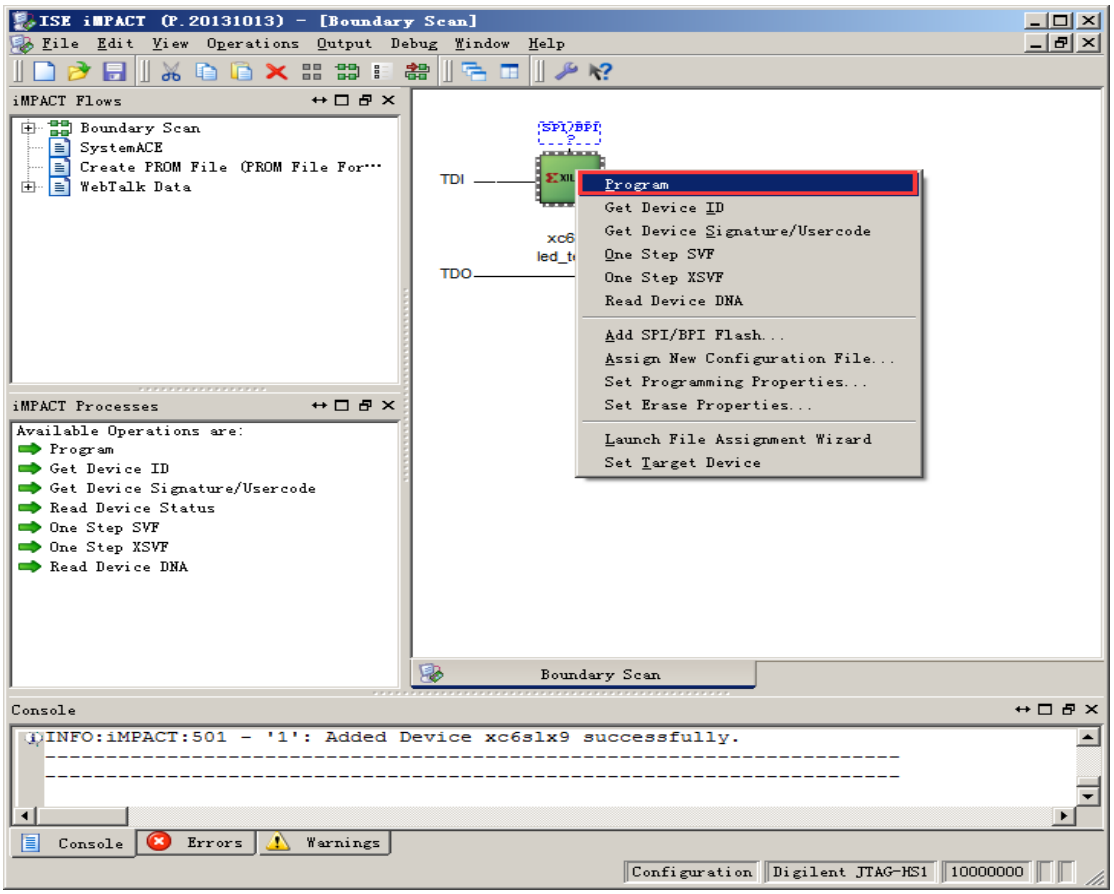
（7）在弹出的选项框里找到工程保存的位置，选中 bit 文件，在此例中为“led_test.bit”并打开


Assign New Configuration File
计算机 ▸ 本地磁盘 (D:) ▸ AX309 ▸ led_test ▸
搜索 led_test
组织
新建文件夹
收藏夹
下载
桌面
最近访问的位置
库
Subversion
视频
图片
文档
迅雷下载
音乐
家庭组
计算机
本地磁盘 (C:)
本地磁盘 (D:)
web (\\192.168.
名称
修改日期
类型
大小
_ngo 2017/9/24 17:40 文件夹
_xmsgs 2017/9/24 17:40 文件夹
ipcore_dir 2017/9/23 20:51 文件夹
iseconfig 2017/9/23 21:34 文件夹
src 2017/9/23 21:34 文件夹
xlnx_auto_0_xdb 2017/9/24 17:40 文件夹
xst 2017/9/23 22:10 文件夹
led_test.bit 2017/9/24 17:40 BIT 文件
文件名(N): led_test.bit
All Design Files (*.bit *.rbt *.n
打开(O)
取消

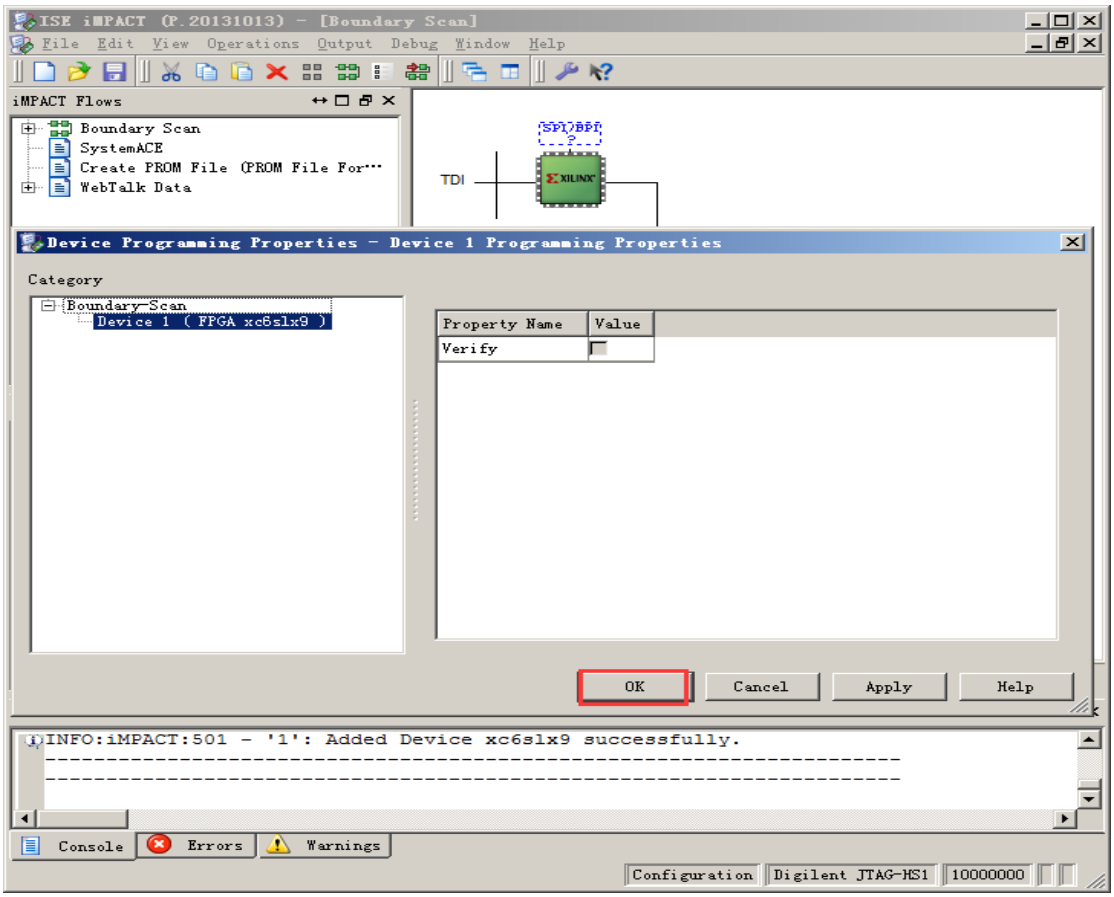
（8）选择 No



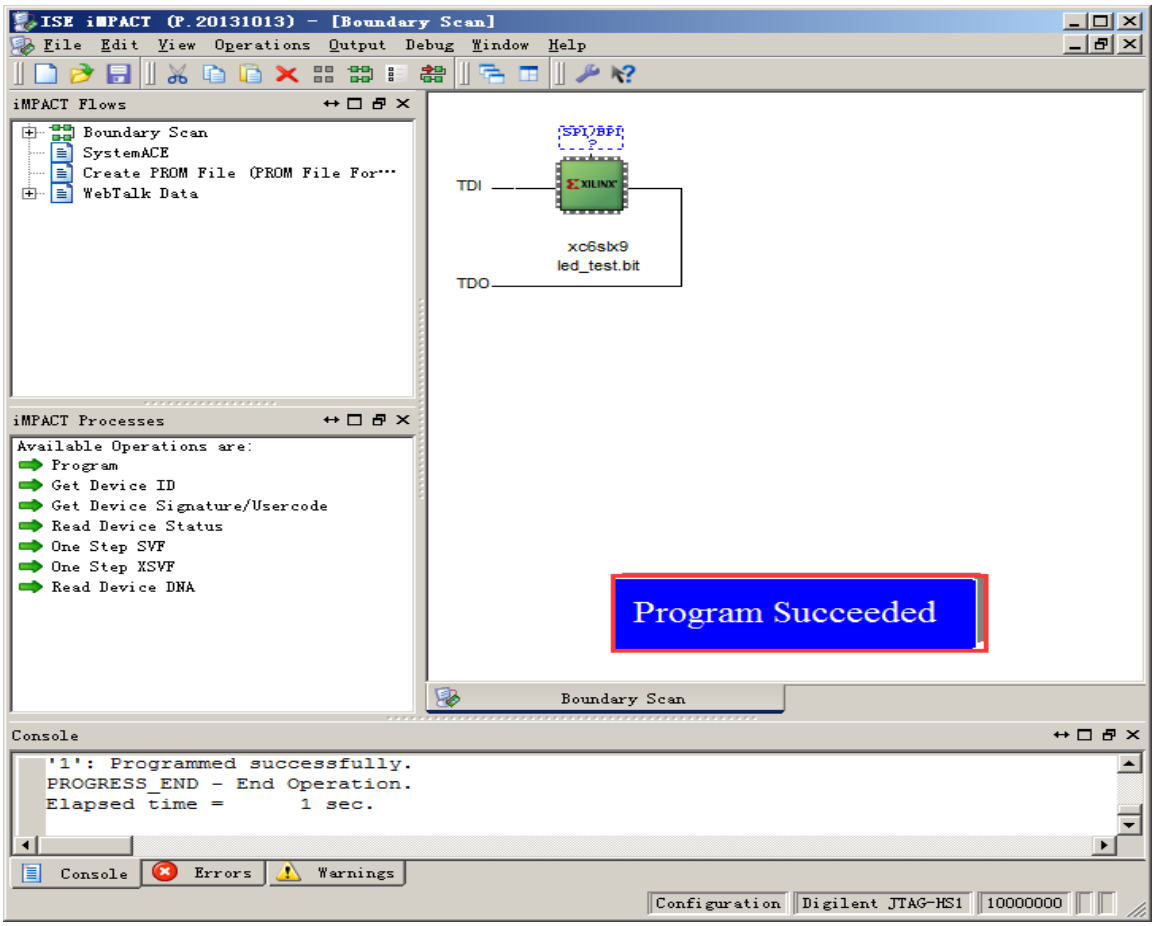
（9）回到 ISE iMPACT 的界面再次选中芯片的图标。单击右键选择 Program:



（10） 点击 OK 进行下一步

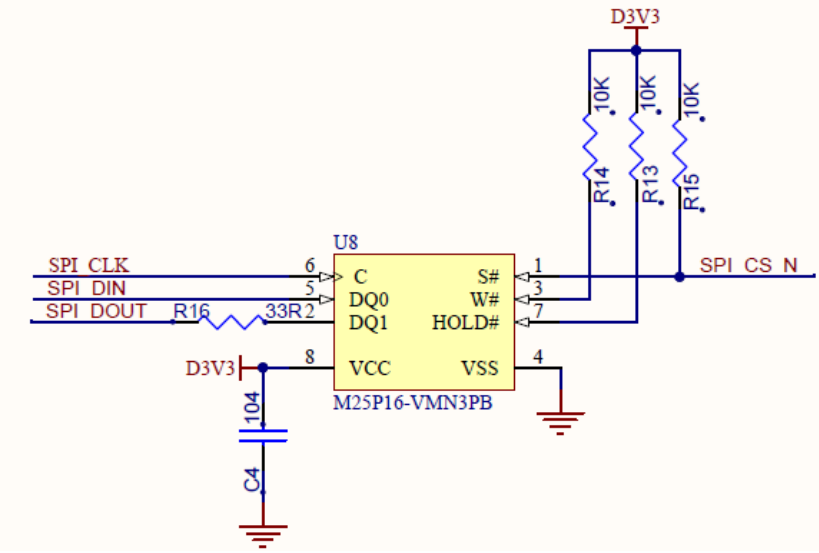


（11） 出现蓝色的 Program Succeeded 就说明程序已经工程下载到 FPGA 上了，可以观察到板上的 led 灯是否像程序中描述的那样在运行。

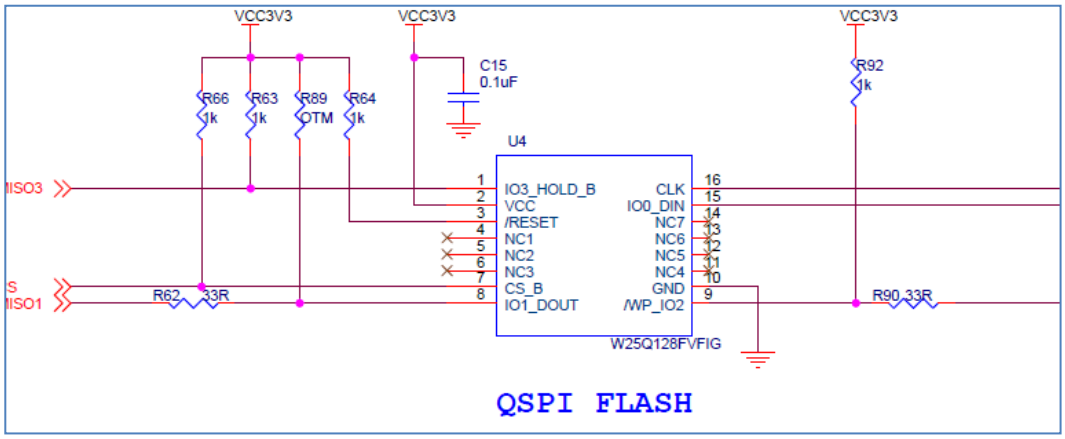


7 固化程序到 Flash

在黑金 FPGA 开发板中使用 SPI Flash 固化 FPGA 程序，Flash 部分原理图如下。



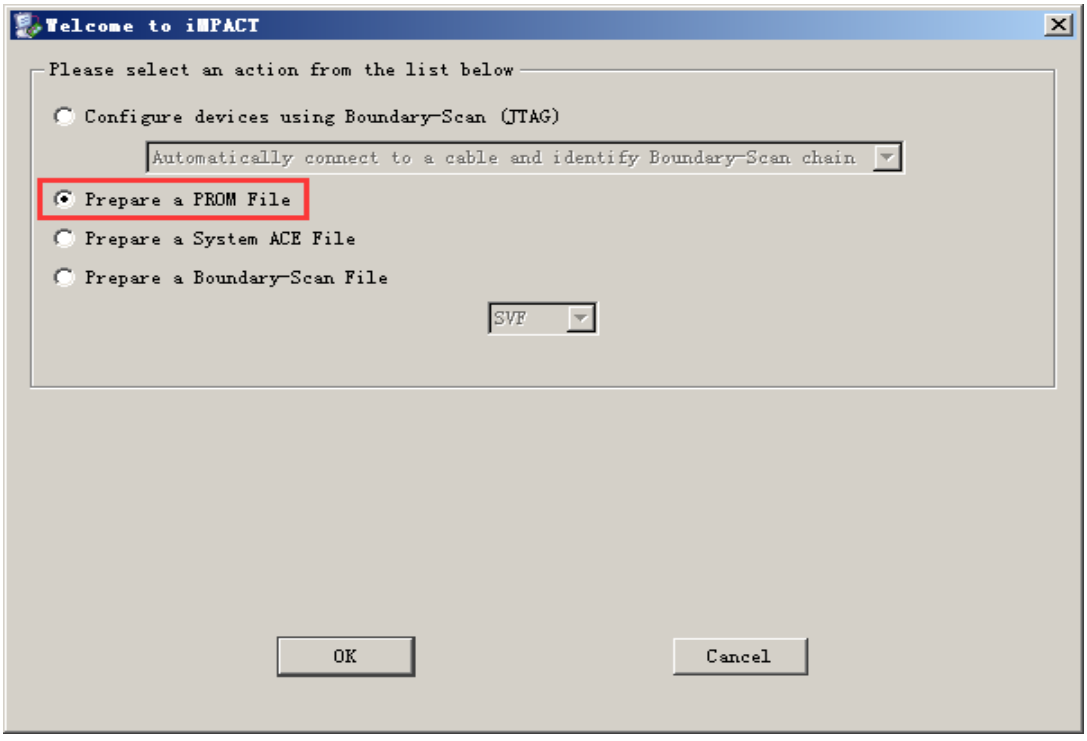
3091 开发板 Flash



AX516 开发板 AX545 开发板中 Flash

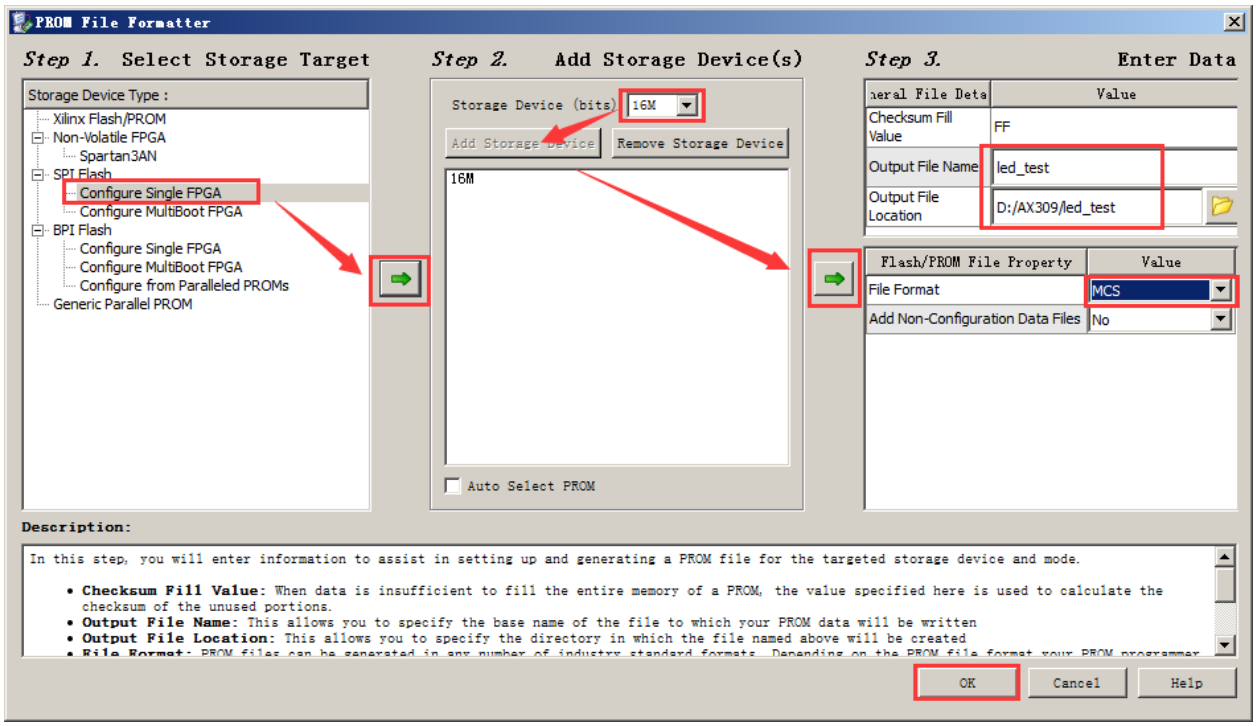
7.1 生成 MCS 文件

（1） 在 ISE 的 iMPACT 界面里同样先选择“Launch Wizard”，选择“Prepare a PROM File”

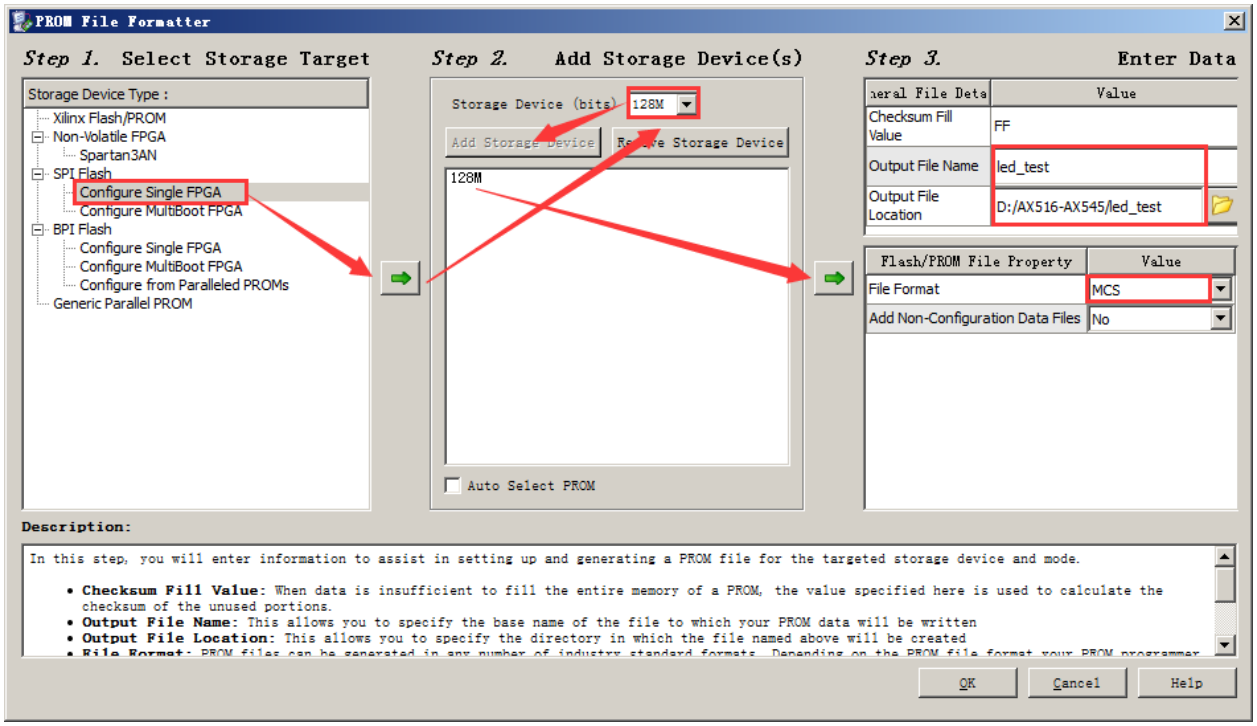


（2） 在生成 mcs 文件之前要确定 SPI FLASH 的型号以及容量和 mcs 文件名和放置的位置。

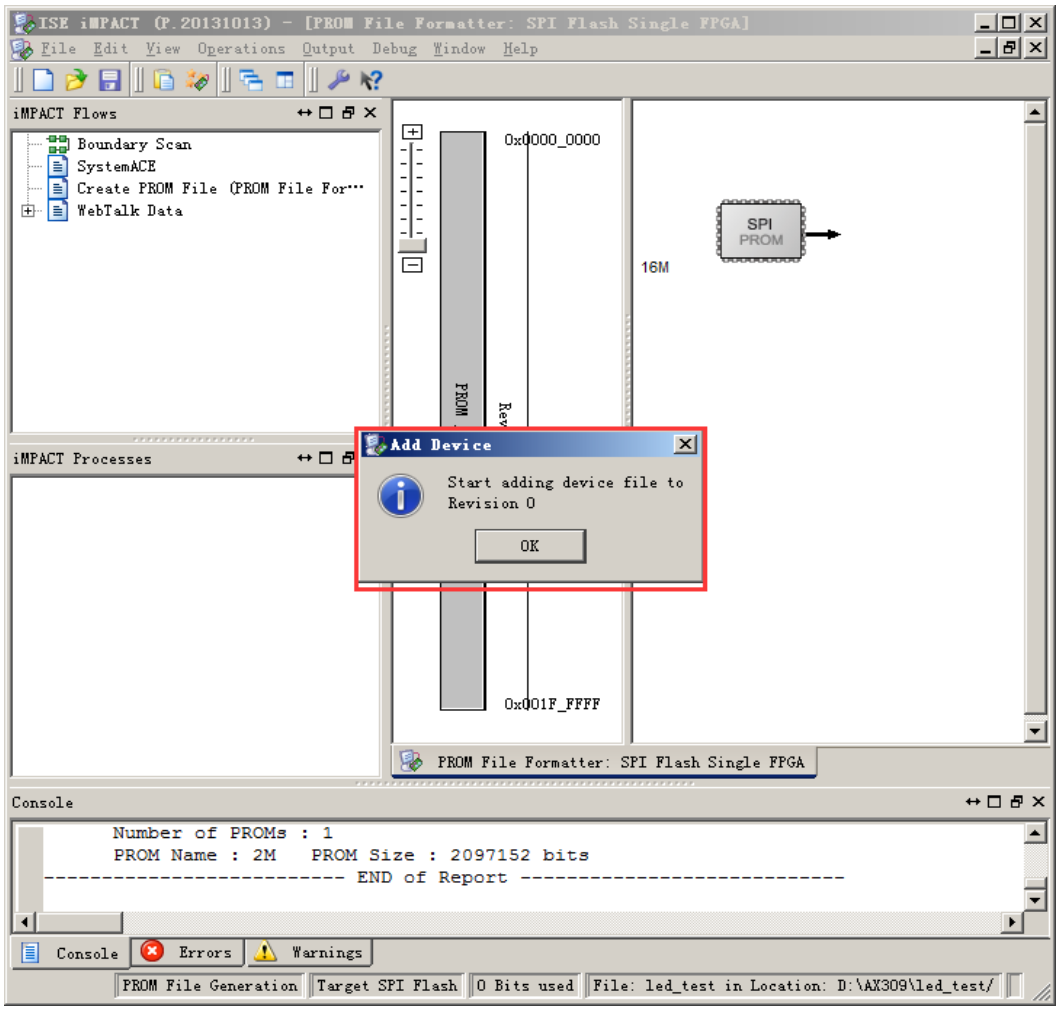
在弹出的“PROM File Formatter”界面中 AX309 开发板依次选择如下：



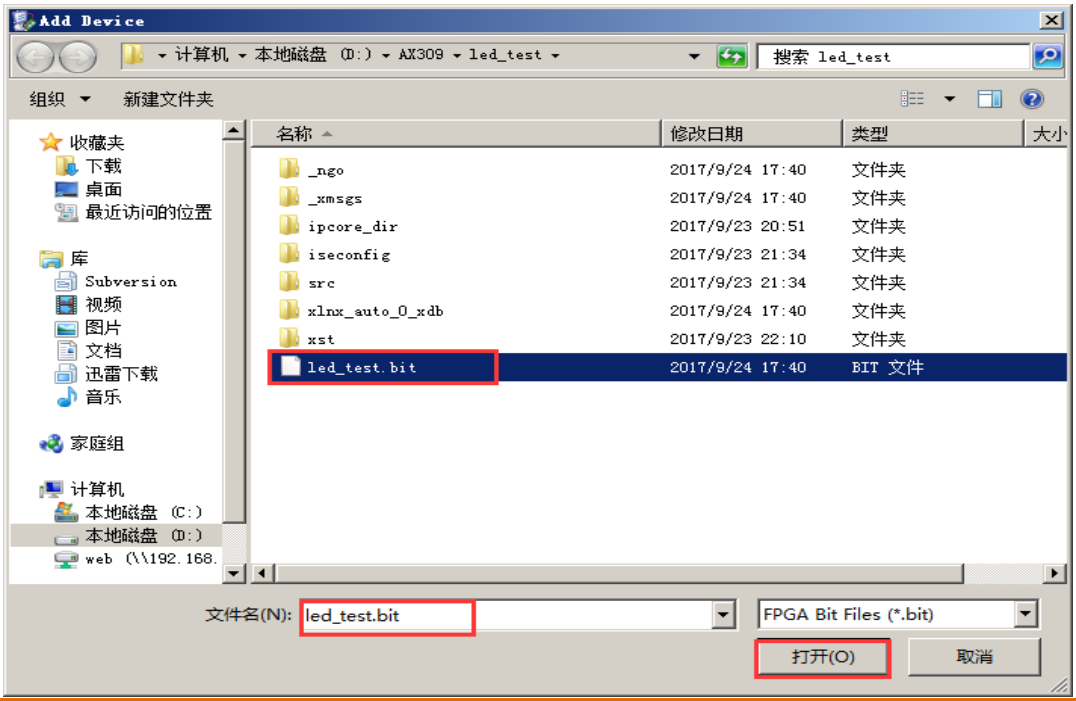
在 AX516 和 AX545 开发板中选择如下



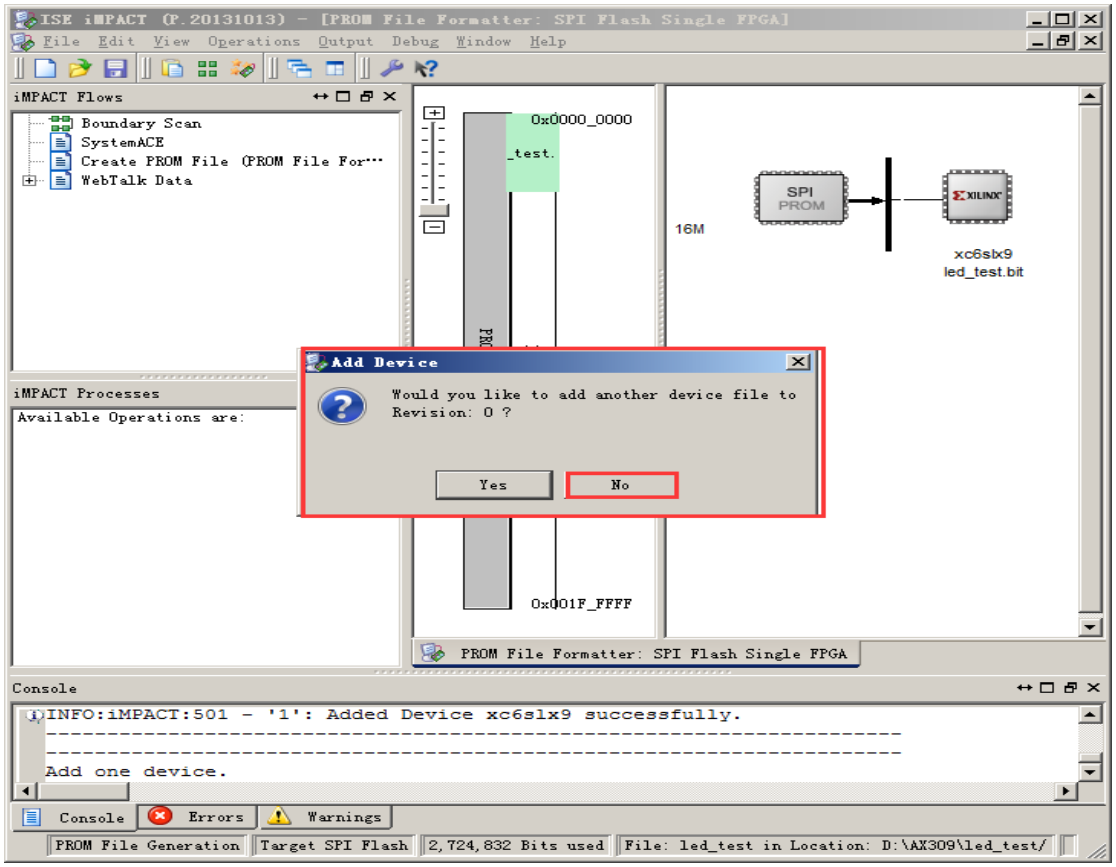
（3）点击 OK 后弹出的 Add Device 窗口中也选择 OK



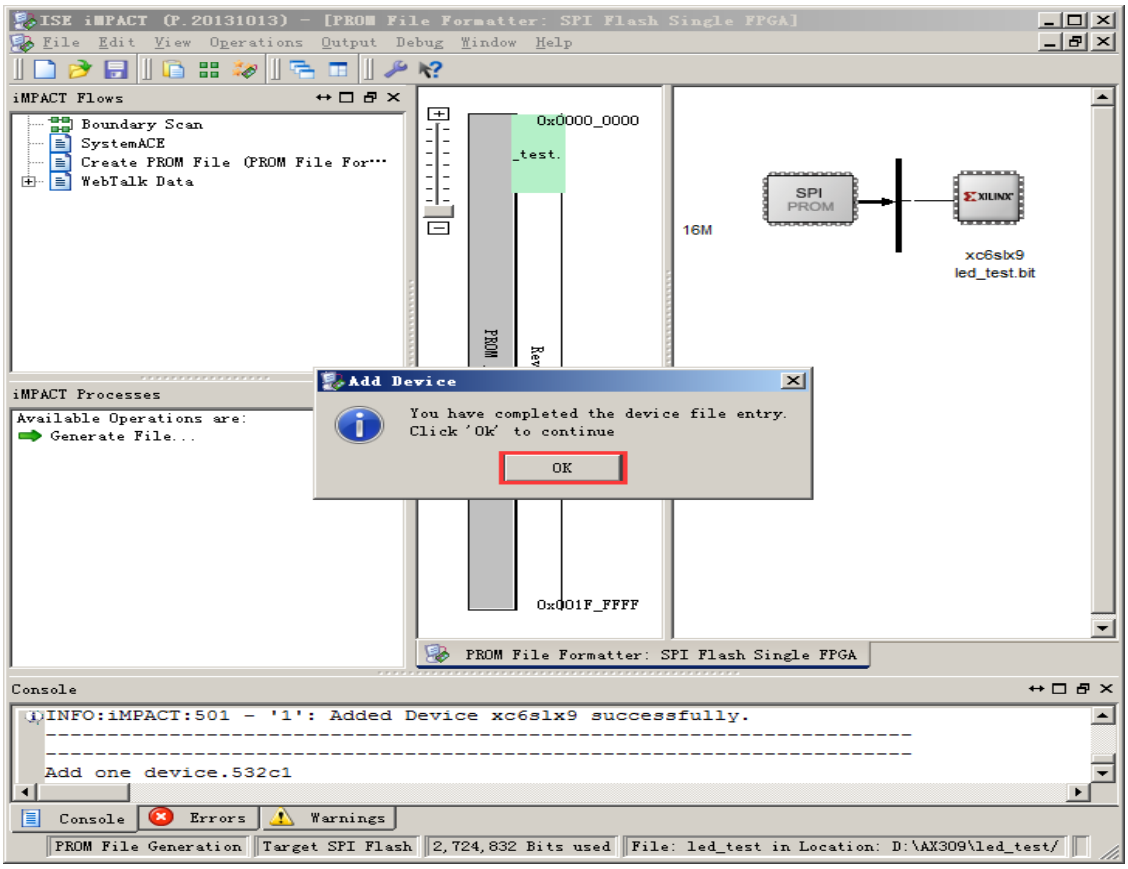
（4）之后弹出如下界面，选择在上面操作中生成的“led_test.bit”文件并打开



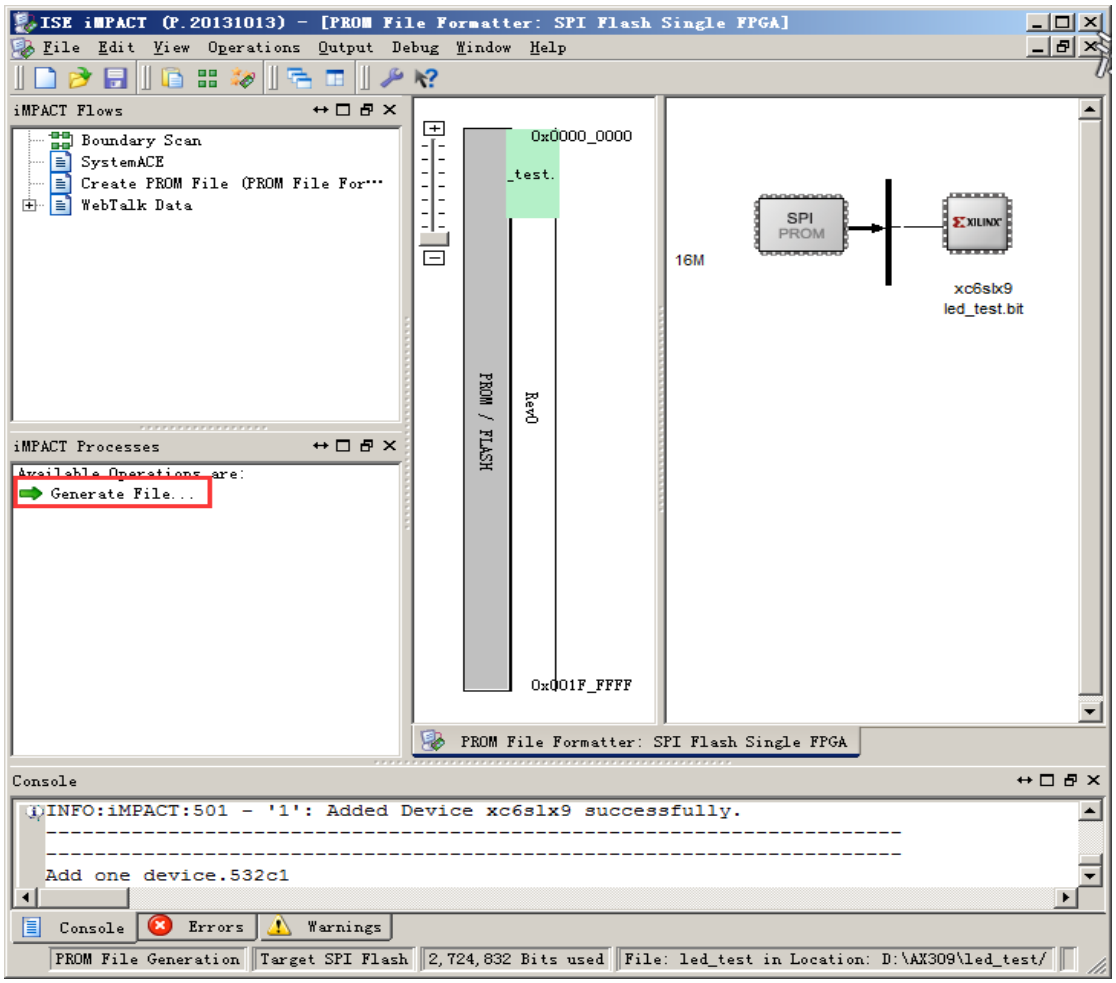
（5）完成上面操作后会弹出“Add Device”的界面，我们选择 NO



（6）直接点击 OK



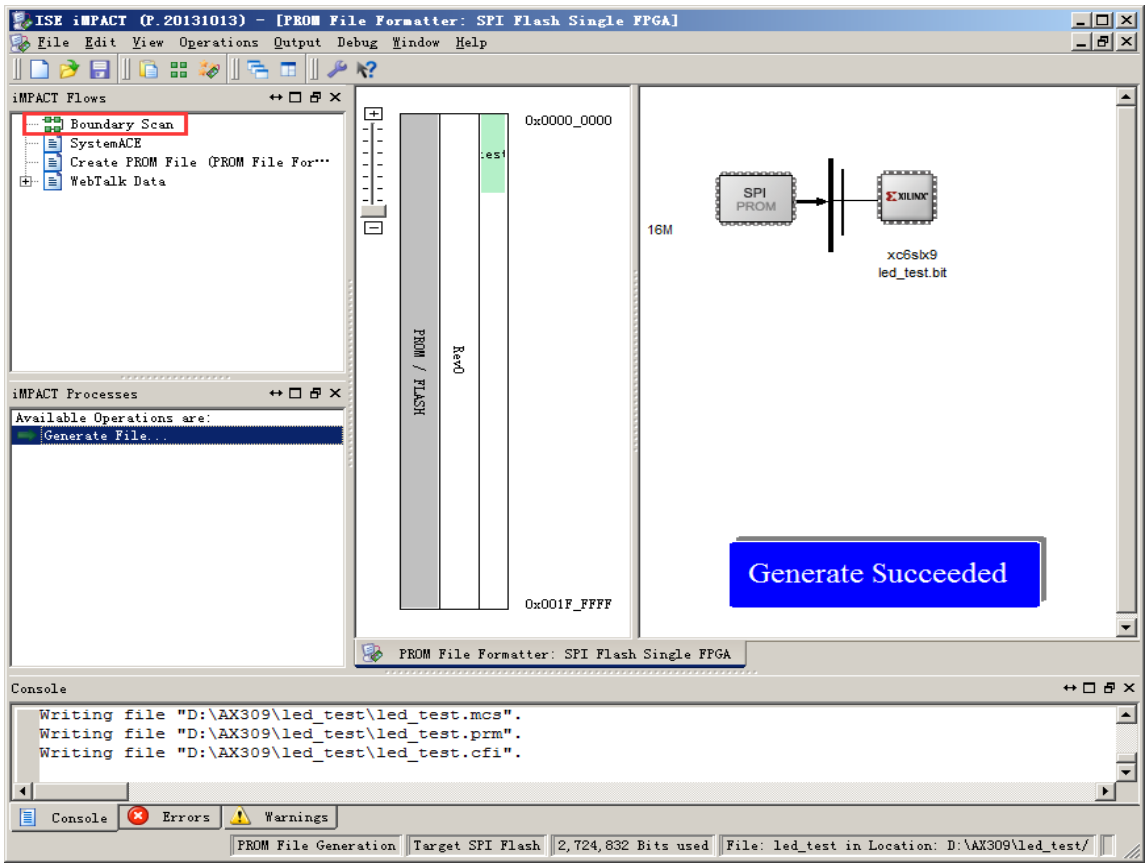
（7）左键双击 ISE iMPACT 界面左下方 iMPACT Processes 框里的"Generate File..."生成 MCS 文件。



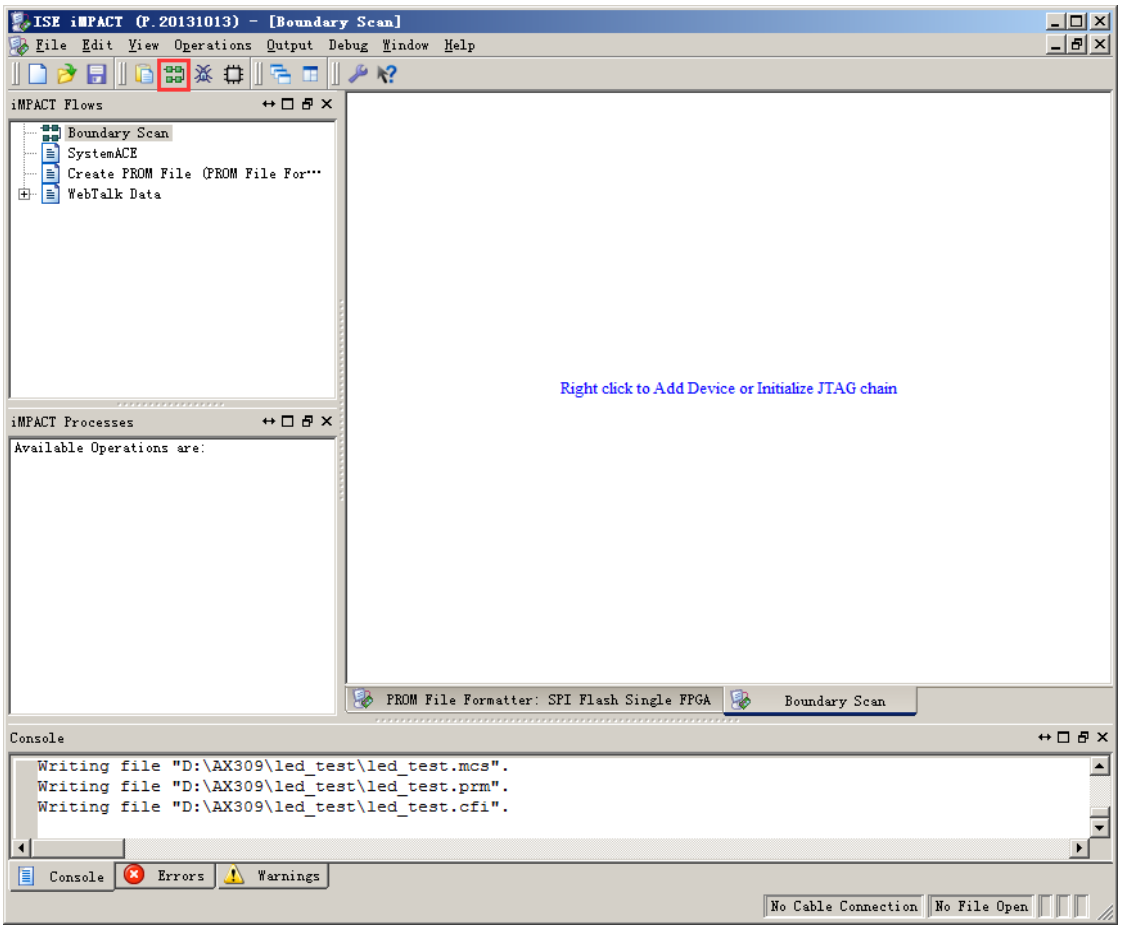
（8）生成成功界面中会出现如下标志：

7.2 固化到 Flash

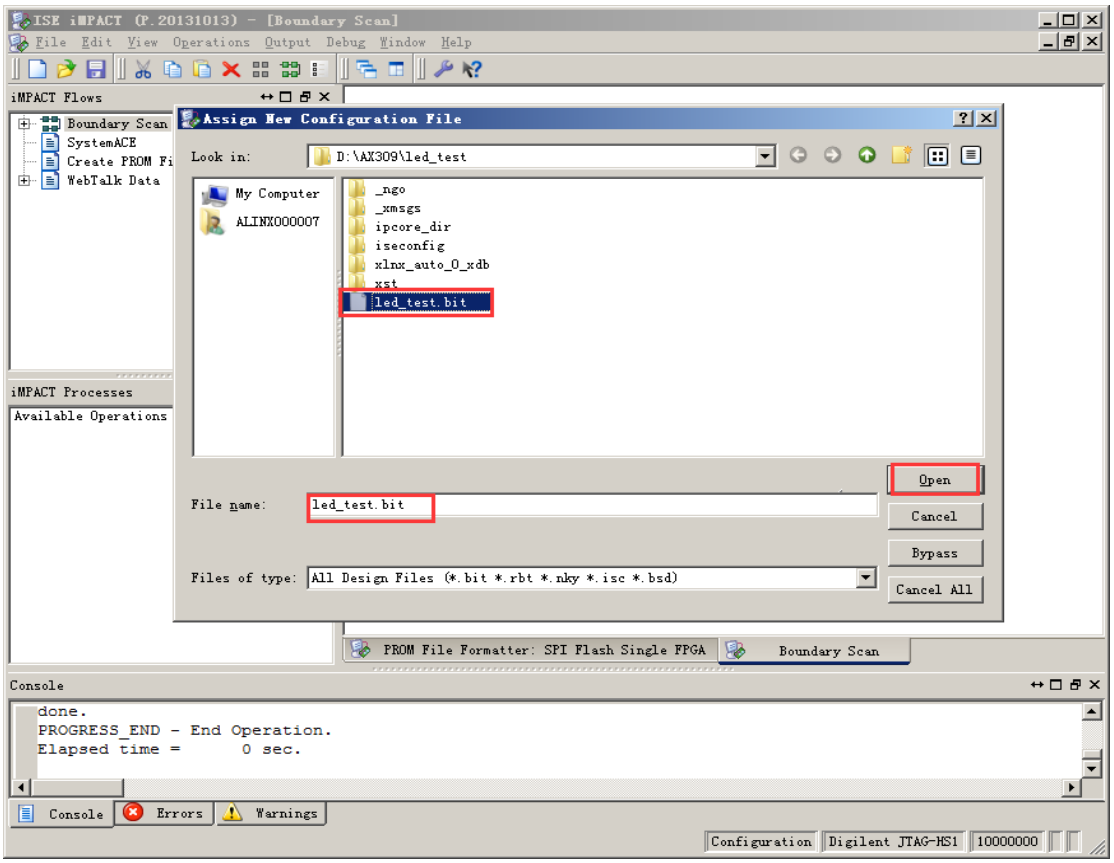
（1） 在 ISE iMPACT 界面中左上方的 iMPACT Flows 框中，双击 Boundary Scan 回到 Boundary Scan 界面：



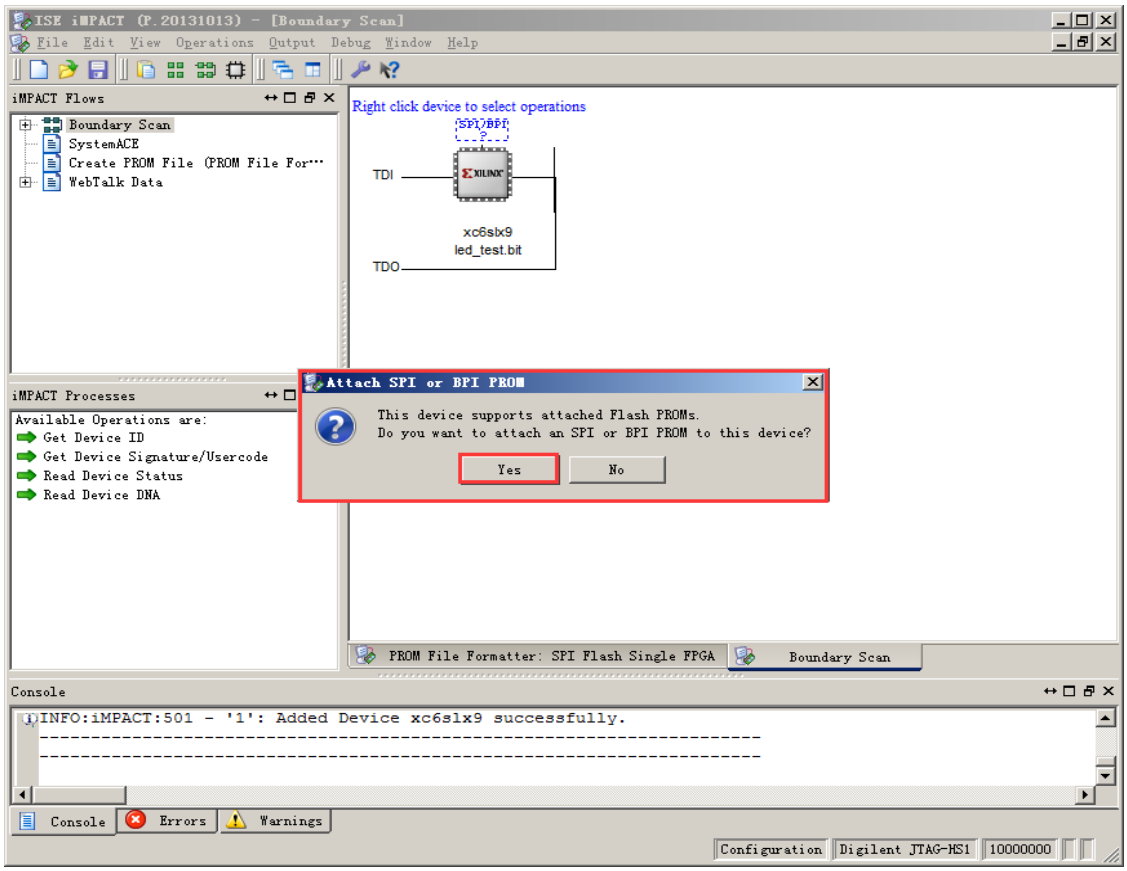
（2） 在点击菜单上的扫描链，如下图红色方框里所示：



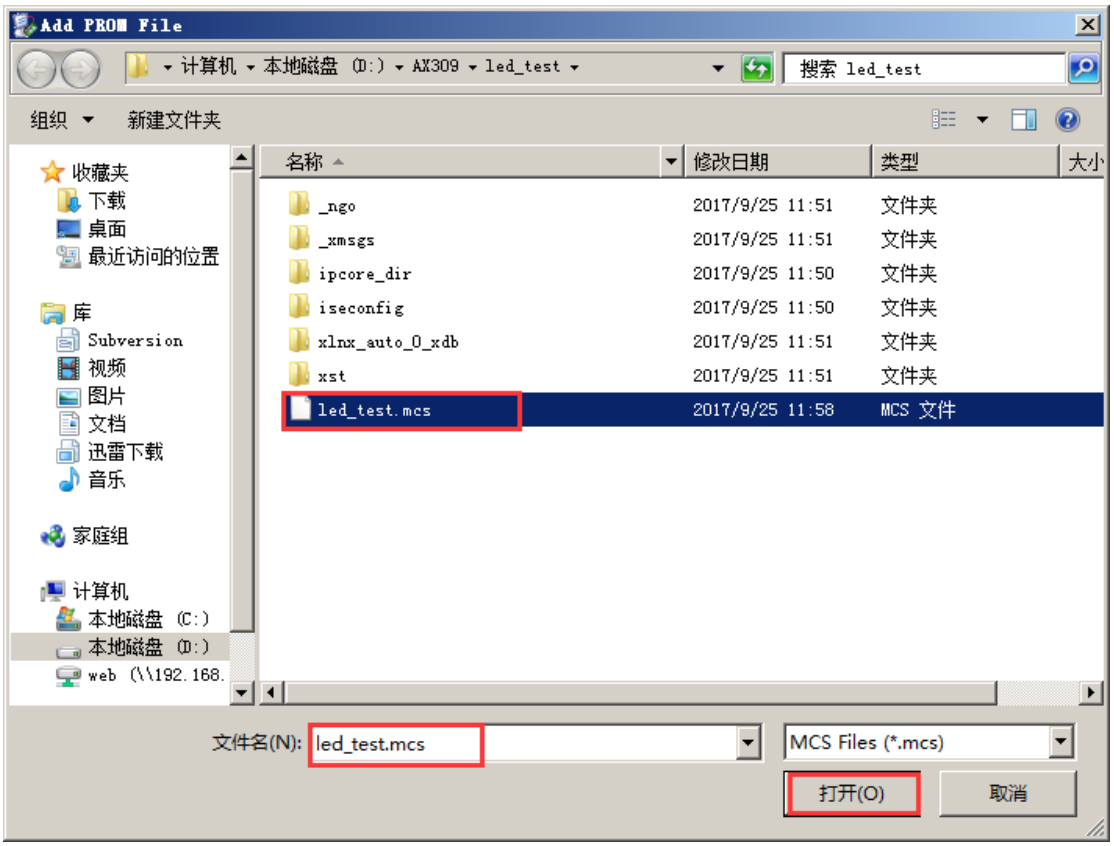
（3） 在弹出的窗口中选择工程保存位置的 bit 文件并打开，在这里是“led_test.bit”



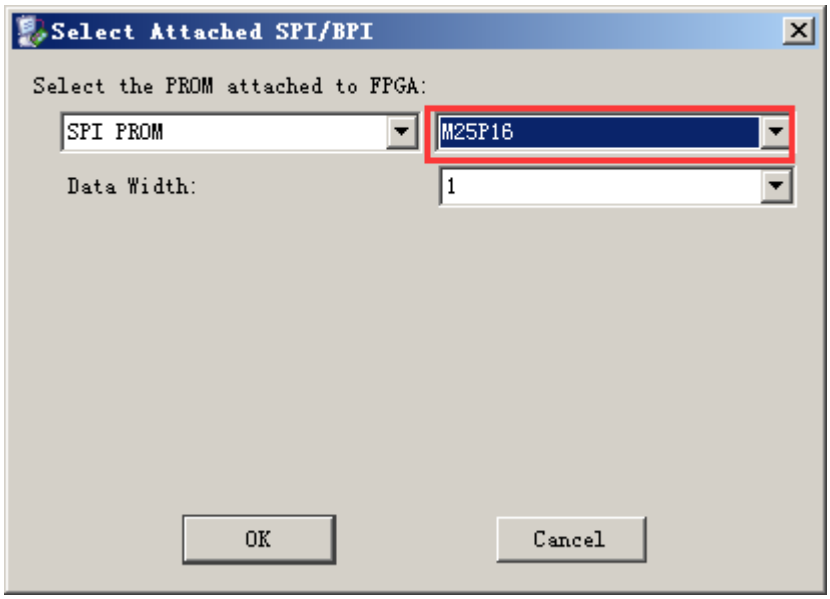
（4） 弹出的 Attach SPI or BPI PROM 窗口选择 Yes



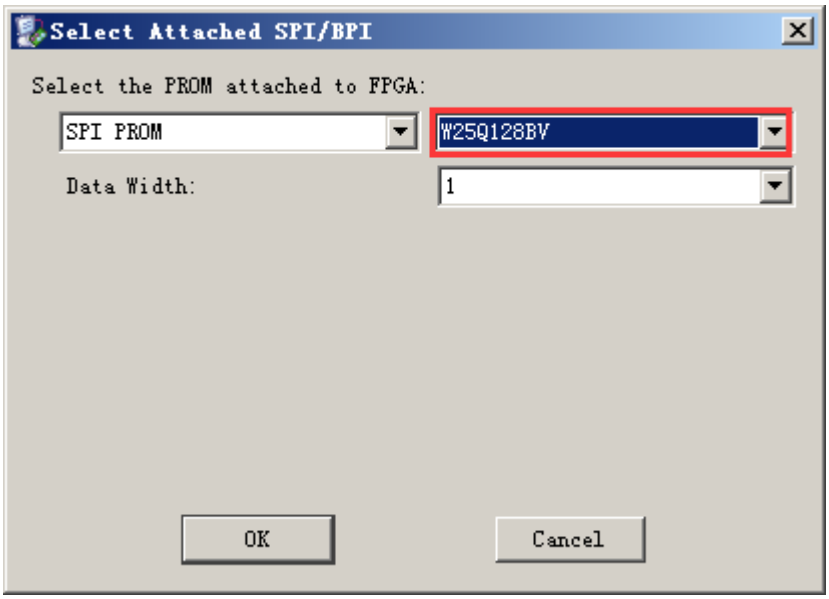
（5） 添加生成的 MCS 文件。在工程保存位置选择 MCS 文件并打开，在本例程中的是“led_test.mcs”



（6） 在弹出的 Select Attached SPI/BPI 窗口中我们根据开发板上的 flash 来选择，AX309 开发板我们选择 M25P16,AX516 和 AX545 开发板我们选择：W25Q128BV

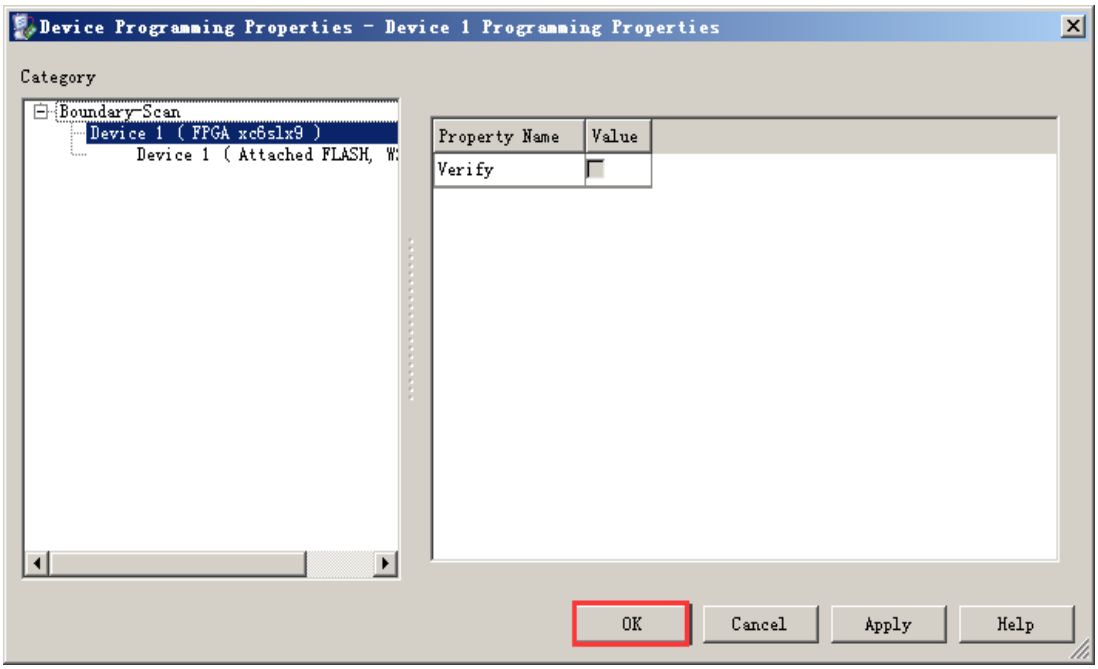


AX309 开发板 SPI Flash 选择

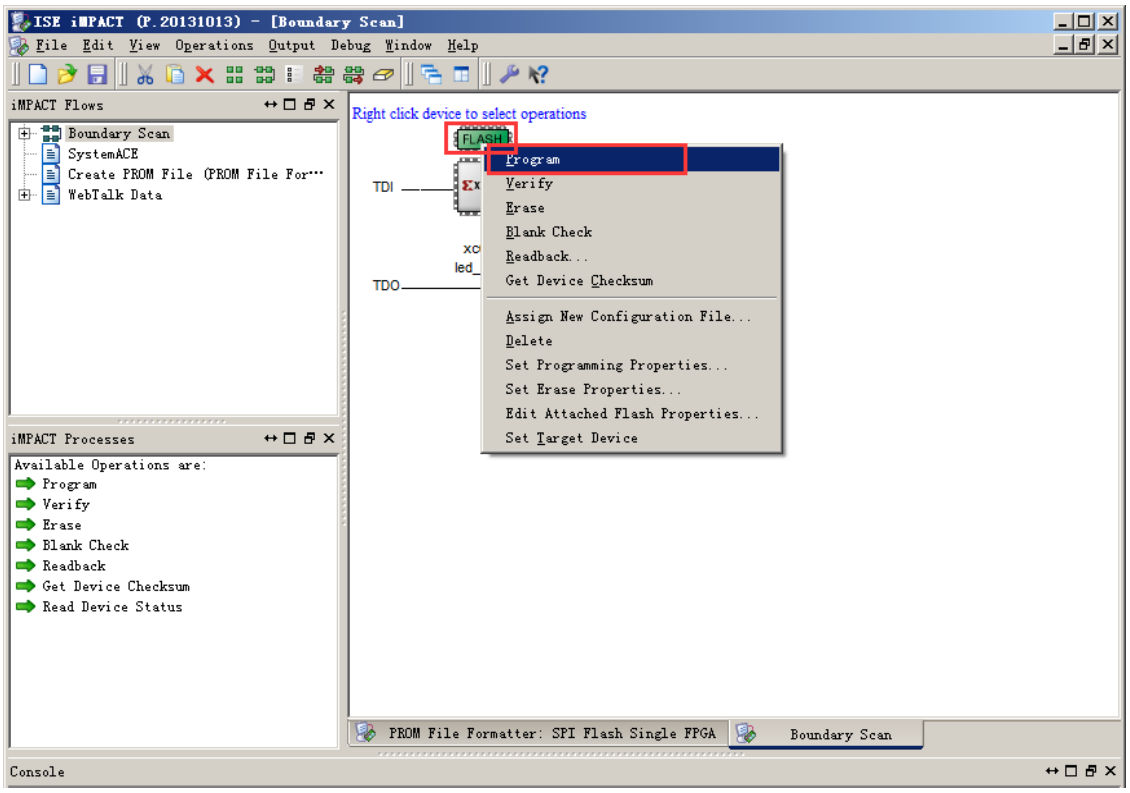


AX516，AX545 开发板 SPI Flash 型号选择

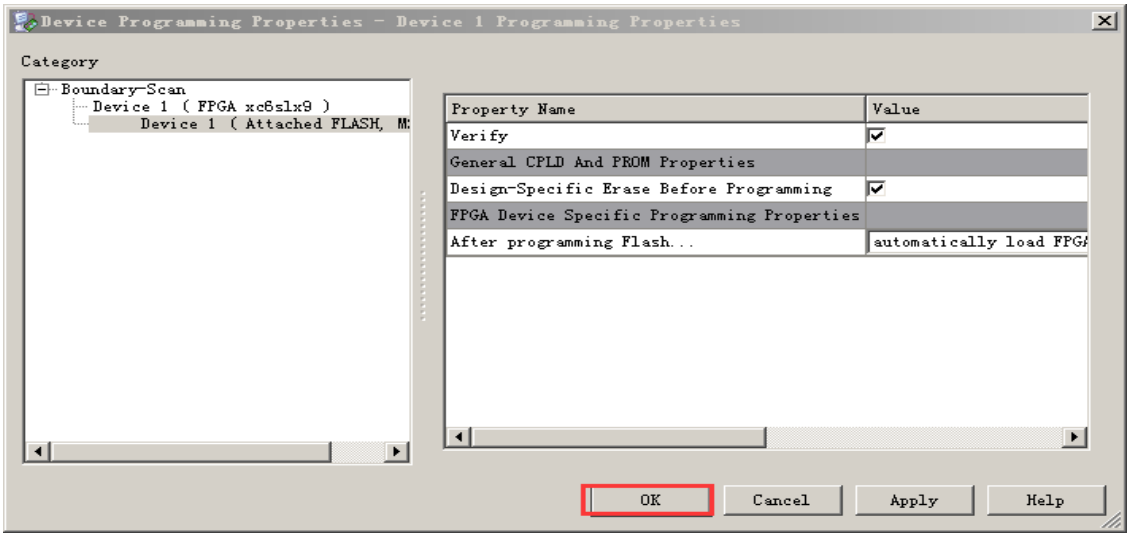
（7） 点击 OK 进入下一步



（8） 选中芯片图标的 Flash，右键单击选择 Program



（9） 点击 OK 进行下载

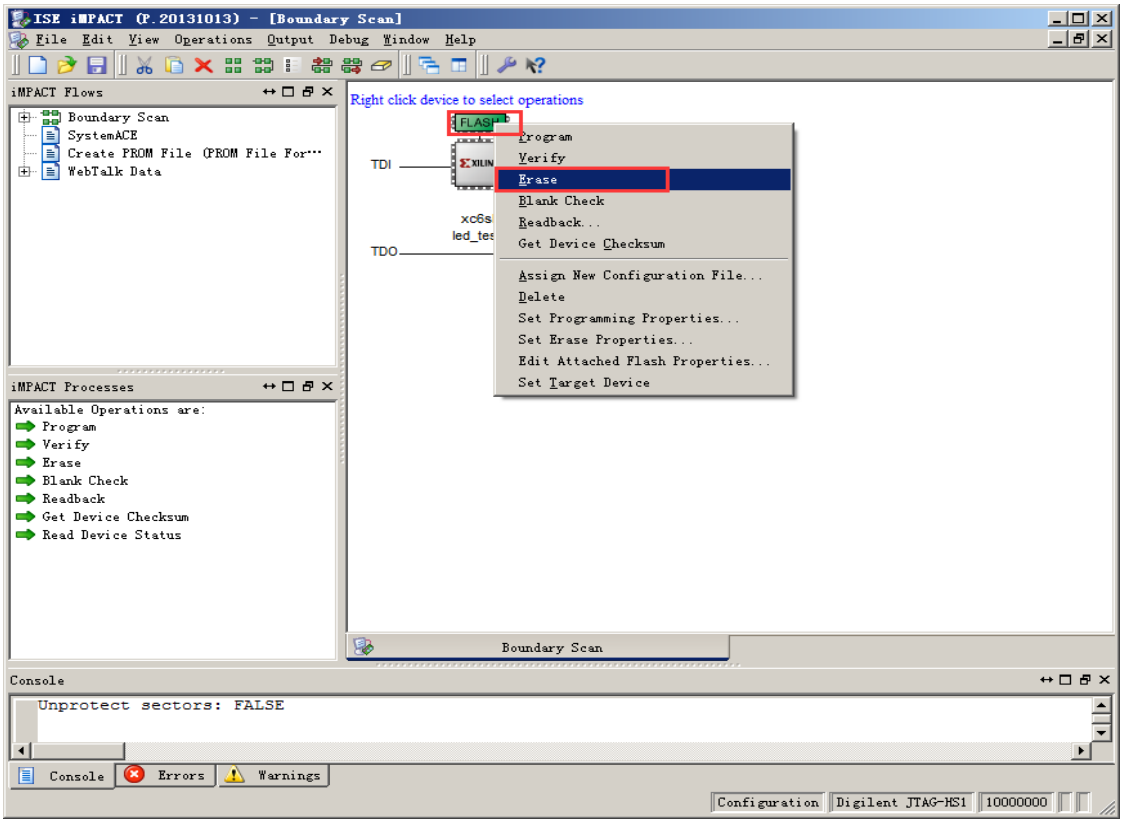


（10）当界面中出现如下标志时说明下载到 Flash 成功

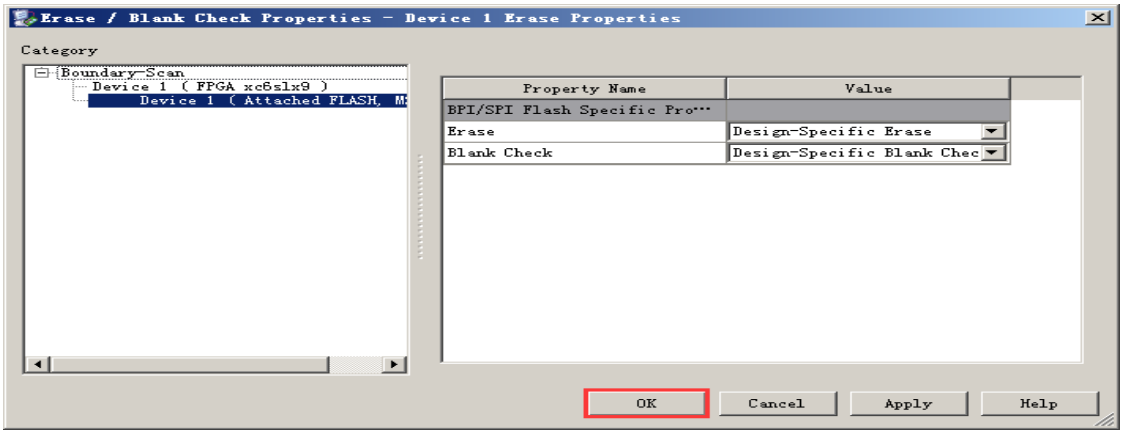
（11）下载 Flash 后，***断电重启开发板，程序就可以运行了***

8 擦除 Flash 中已有的程序

（1） 如果要擦除 Flash 中的程序，我们同样要先添加生成的 MCS 文件，然后才能擦除我们按照上面操作的步骤将生成的 MSC 文件添加并且选择开发板上的 SPI Flash 型号完成后来到如下界面，选中芯片图标的 FLASH，单击右键选择 Erase 进行擦除操作：



点击 OK 完成擦除：



（2） Flash 擦除成功界面中会出现如下标志

9 附录

led_test.v(verilog 代码)

```
    `timescale 1ns / 1ps
    module led_test
    (
        input           clk,            // system clock 50Mhz on board
        input           rst_n,          // reset ,low active
        output reg[3:0] led             // LED,use for control the LED signal on
board
    );

    //define the time counter
    reg [31:0]      timer;

    // cycle counter:from 0 to 4 sec
    always@(posedge clk or negedge rst_n)
    begin
        if (rst_n == 1'b0)
                timer <= 32'd0;                     //when the reset signal
valid,time counter clearing
        else if (timer == 32'd199_999_999)      //4 seconds count(50M*4-
1=199999999)
                timer <= 32'd0;                     //count done,clearing the time
counter
        else
                timer <= timer + 32'd1;             //timer counter = timer
counter + 1
    end

    // LED control
    always@(posedge clk or negedge rst_n)
    begin
        if (rst_n == 1'b0)
                led <= 4'b0000;                     //when the reset signal active
        else if (timer == 32'd49_999_999)       //time counter count to 1st
sec,LED1 lighten
                led <= 4'b0001;
        else if (timer == 32'd99_999_999)       //time counter count to 2nd
sec,LED2 lighten
                led <= 4'b0010;
        else if (timer == 32'd149_999_999)      //time counter count to 3rd
sec,LED3 lighten
                led <= 4'b0100;
        else if (timer == 32'd199_999_999)      //time counter count to 4th
sec,LED4 lighten
                led <= 4'b1000;
    end
    endmodule
```

注意：在定义寄存器时，如果寄存器在 always 块里使用必须定义为 reg 类型，如果仅是用于连线或是直接赋值需定义为 wire 类型，输入信号的类型不能定义为 reg 型，不管是 reg 类型信号

还是 wire 类型的信号，定义的寄存器宽度必须满足使用时的需要，但必须稍大于或等于需要使用的位宽。若定义寄存器位宽远远大于使用需求则会浪费资源，如果定义的位宽小于使用需求，则会造成数据位截断，导致程序错误。还有其他信号的类型及用法请大家参考 Verilog 语法教程。