

# *Lab3: 基于自定义 IP 核的 LED 显示实验* (硬件配置部分)

基于 Nexys 4 FPGA 平台

## Lab3: 基于自定义 IP 核的 LED 显示实验（硬件配置部分）

---

### 实验简介

---

本实验旨在使读者进一步熟悉 Xilinx 的 XPS 和 SDK 工具的使用，并初步掌握添加自定义 IP 核的步骤，最终完成一个通过串口输入数据来控制流水灯方向的简单程序。

---

### 实验目标

---

在完成本实验后，您将学会：

- 如何在 XPS 工具中添加自定义 IP 核

---

### 实验过程

---

本实验旨在指导读者使用 Xilinx 的 XPS 工具，调用 GPIO 与 UART 的 IP 核，并将导入到 SDK，调用它们，通过在串口输入数据并控制七段数码管的显示内容，然后在 Nexys 4 平台上进行测试验证。

实验由以下步骤组成：

1. 在 XPS 中建立工程
2. 添加 IP 核并调整相关设置
3. 进行端口的互连
4. 生成硬件工程的比特流文件

---

### 实验环境

---

◆ 硬件环境

1. PC 机
2. Nexys 4 FPGA 平台

◆ 软件环境

Xilinx ISE Design Suite 14.3（FPGA 开发工具）

## 第一步 创建工程

**1-1. 运行 Xilinx Platform Studio,创建一个空的新工程，基于 xc6slx45csg484-3 芯片和 VHDL 语言.**

**1-1-1.** 选择 开始菜单 > 所有程序 > Xilinx Design Tools > ISE Design Suite 14.3 > EDK > Xilinx Platform Studio. 点击运行 **Xilinx Platform Studio(XPS)** (Xilinx Platform Studio 是 ISE 嵌入式版本 Design Suite 的关键组件，可帮助硬件设计人员方便地构建、连接和配置嵌入式处理器系统，能充分满足从简单状态机到成熟的 32 位 RISC 微处理器系统的需求)，如图 1 所示。

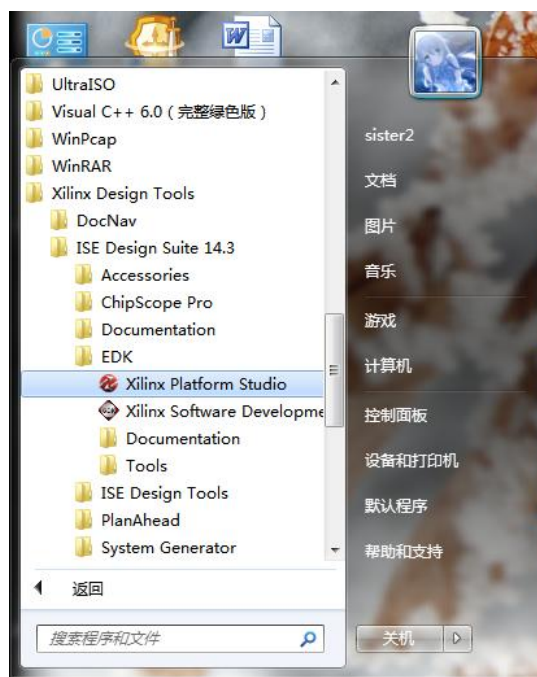


图 1: XPS 软件打开位置

**1-1-2.** 点击 **Create New Project Using Base System Builder** 来打开新工程建立向导。会出现一个 **Create New XPS Project Using BSB Wizard** 对话框，如图 2。



图 2：新工程建立界面

- 1-1-3. 如图 3，在新工程建立向导对话框的 **Project File** 栏选择工程建立后存放的路径，笔者自定义的路径是 I:\labz\liushuideng，大家可以建立任意路径，只要路径名称不包含中文及空格即可。建立的工程的名称我们使用默认的 system.xmp 即可。点击 **OK**。

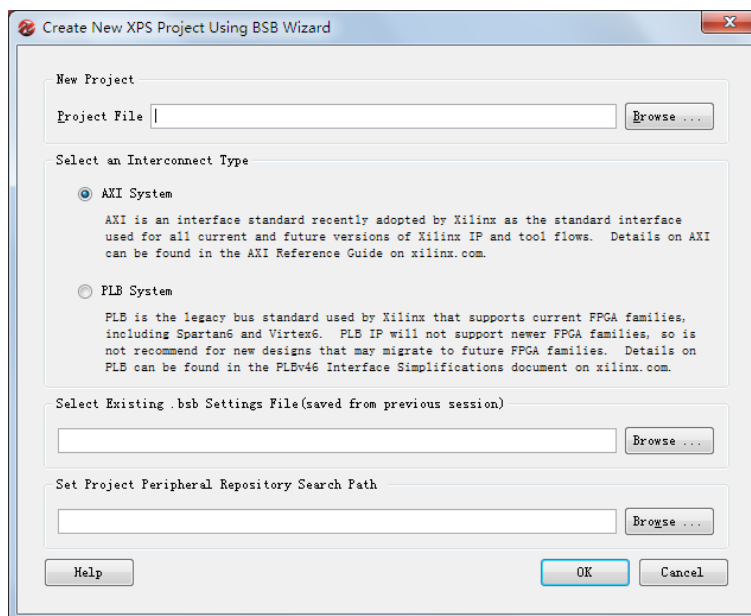


图 3：新工程建立向导

- 1-1-4. 新出现的是关于工程的一些参数设置的对话框，设置如下的参数后，点击 **Next**，如图 4。
- architecture:** artix 7
  - Device:** XC7a1007
  - Package:** CSG324
  - Speed:** -3

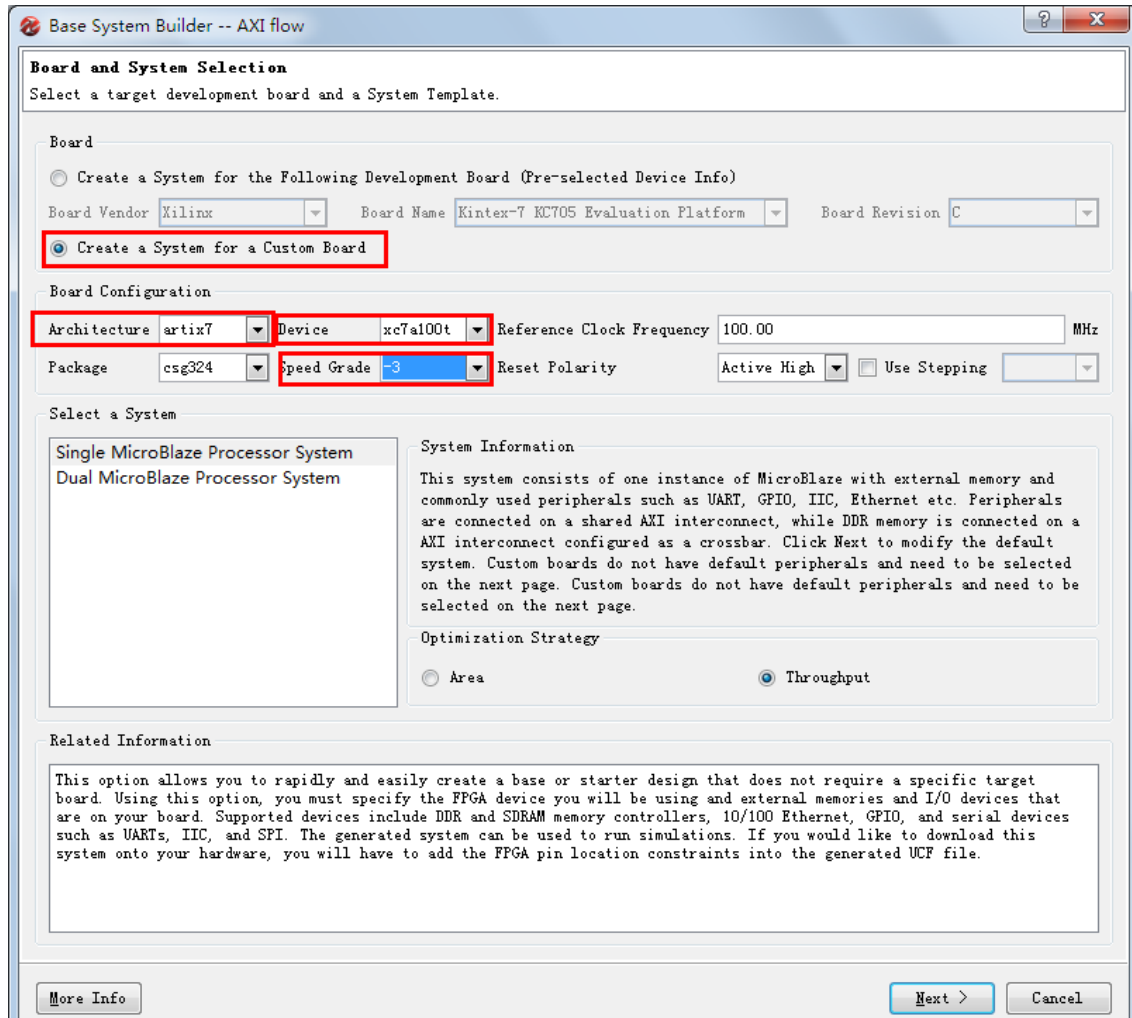


图 4：新工程参数设置

1-1-5. 在接下来出现的页面中选择要添加的 IP 核，并设置 IP 核的参数：

单击 **Select and configure Peripherals** 下的 **Add Device...**

出现图 5 中的蓝色对话框。

在 **IO Interface Type** 中的下拉菜单中选择 **UART**。

在 **Device** 的下拉菜单中选择 **RS232**。

单击 OK，即可添加 UART 的 IP 核。

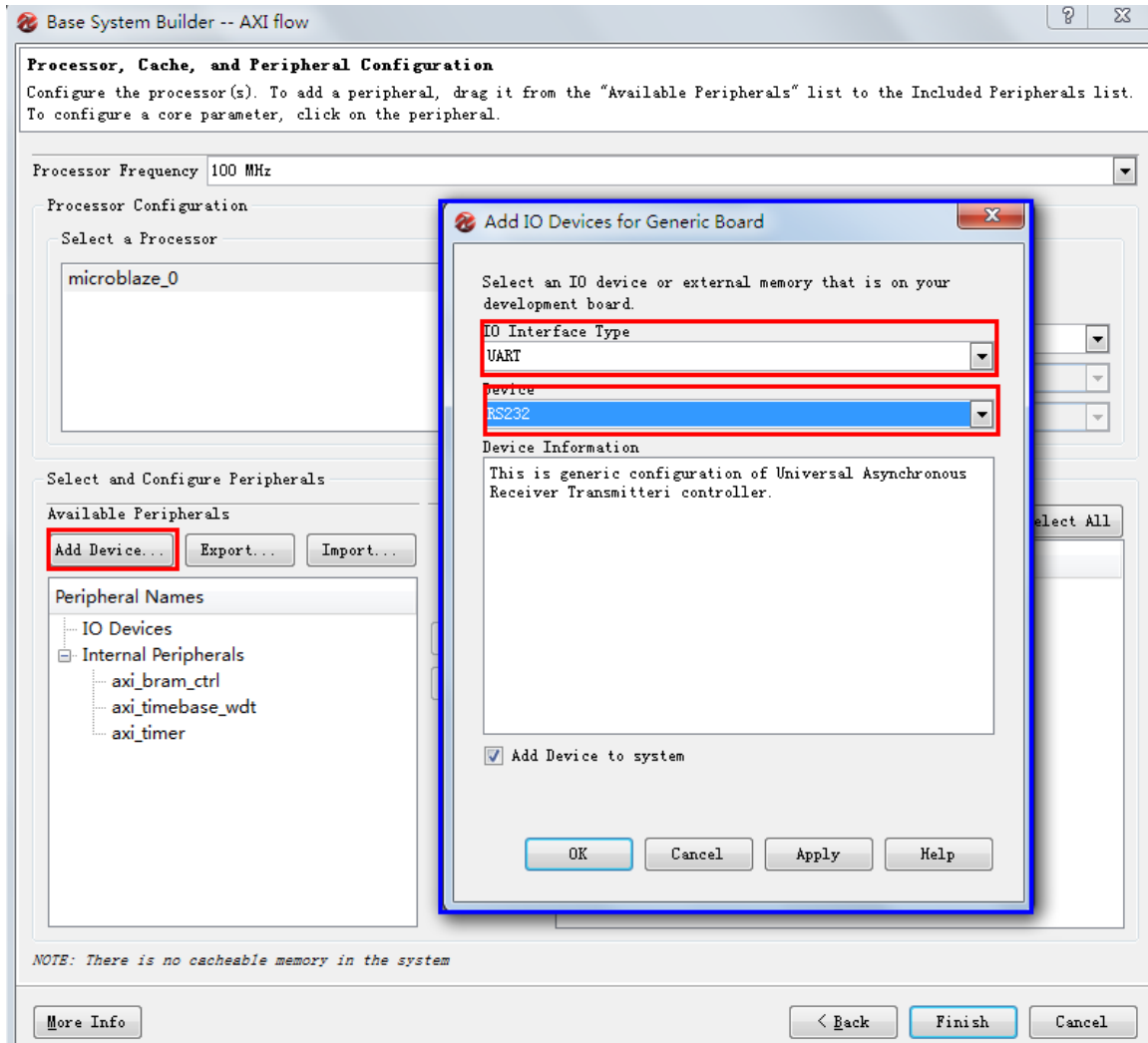


图 5：添加串口的 IP

在 **IO Interface Type** 中的下拉菜单中选择 **GPIO**。

在本实验中，我们要添加两个 **GPIO** 设备。

所以，在 **Device** 的下拉菜单中我们需要先后选择 **DIP\_Switches** 和 **Push\_Buttons**。

因为我们在后续实验中可能要用到这些控制键，如图 6 所示。单击 **OK**，即可添加这些 **GPIO** 的 IP 核。

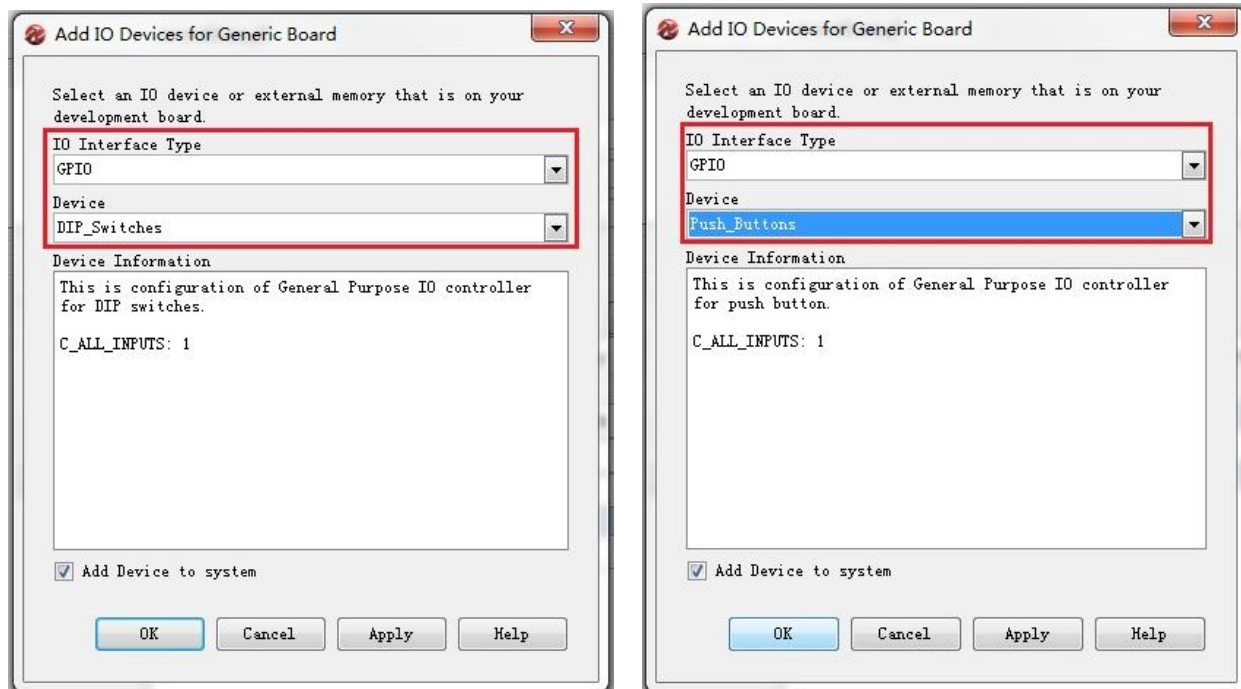


图 6：添加 GPIO 的 IP

1-1-6. 注意,串口的默认波特率设置为 9600。在 Lab2 中, 我们需要统一修改到 115200, 以便提高数据传输速度, **SDK 工程中的 Terminal 的波特率以及串口的其他设置必须与之保持一致。**

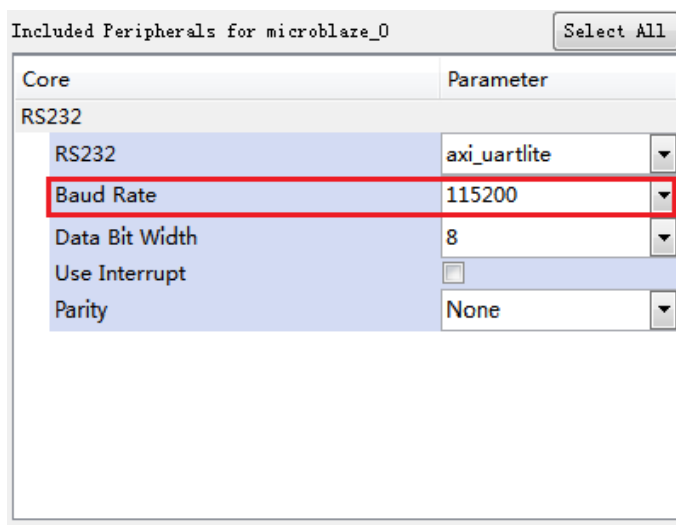
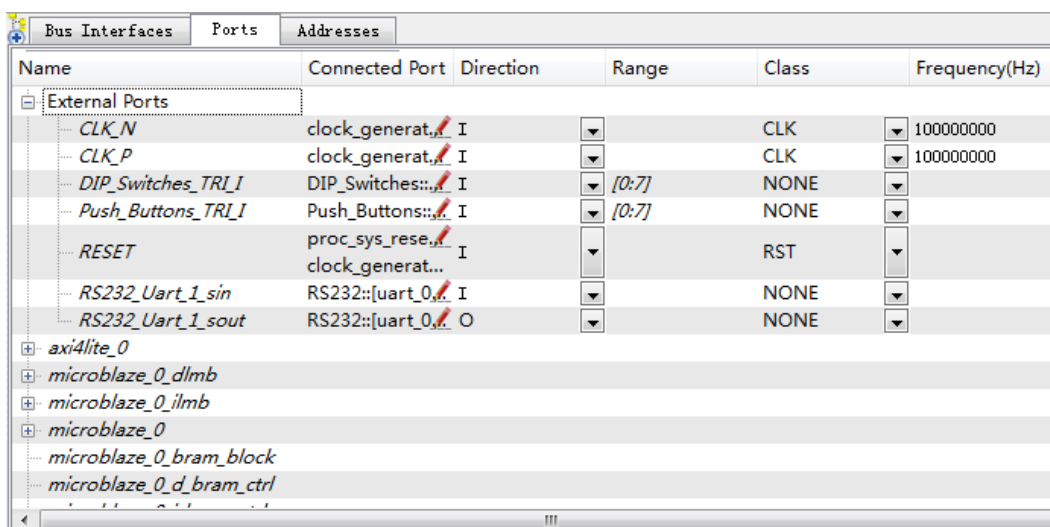


图 7：串口的设置

## 第二步 添加自定义 IP 核&进行端口的互连

### 2-1. 在 PORT 选项卡中修改时钟的相关设置

#### 2-1-1. Port 选项卡（展开 External Port），如图 8。

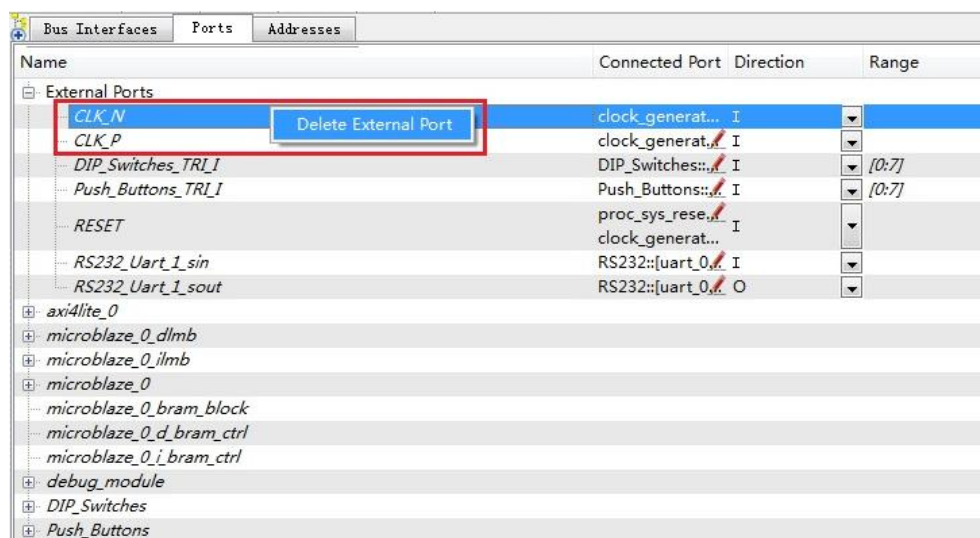


Name	Connected Port	Direction	Range	Class	Frequency(Hz)
CLK_N	clock_generat...	I		CLK	100000000
CLK_P	clock_generat...	I		CLK	100000000
DIP_Switches_TRI_I	DIP_Switches::...	I	[0:7]	NONE	
Push_Buttons_TRI_I	Push_Buttons::...	I	[0:7]	NONE	
RESET	proc_sys_rese...	I		RST	
RS232_Uart_1_sin	RS232::[uart_0...	I		NONE	
RS232_Uart_1_sout	RS232::[uart_0...	O		NONE	
axi4lite_0					
microblaze_0_dlm					
microblaze_0_ilmb					
microblaze_0					
microblaze_0_bram_block					
microblaze_0_d_bram_ctrl					

图 8: PORT 选项卡的初始状况

#### 2-1-2. 将 External Port 中的 CLK\_N 和 CLK\_P 都去掉。

右键选中该端口，然后点击 Delete External Port，如图 9。



Name	Connected Port	Direction	Range
CLK_N	clock_generat...	I	
CLK_P	clock_generat...	I	
DIP_Switches_TRI_I	DIP_Switches::...	I	[0:7]
Push_Buttons_TRI_I	Push_Buttons::...	I	[0:7]
RESET	proc_sys_rese...	I	
RS232_Uart_1_sin	RS232::[uart_0...	I	
RS232_Uart_1_sout	RS232::[uart_0...	O	
axi4lite_0			
microblaze_0_dlm			
microblaze_0_ilmb			
microblaze_0			
microblaze_0_bram_block			
microblaze_0_d_bram_ctrl			
microblaze_0_i_bram_ctrl			
debug_module			
DIP_Switches			
Push_Buttons			

图 9: 删除多余时钟



### 2-1-3. 将 Clock\_generator\_0 作为新的时钟，加入外部端口。

找到 **Clock\_generator\_0** 中的 **CLKIN**，右键选中，在菜单中点击 **Make external**。

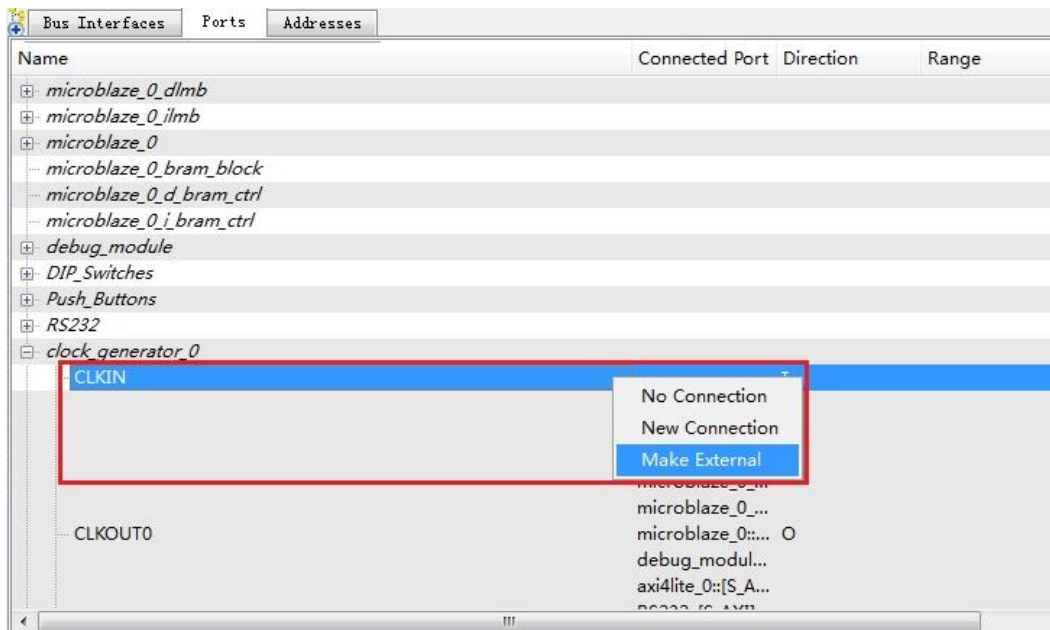


图 10: Clock\_generator\_0 中的 CLKIN

## 2-2. 添加自定义 IP 核

2-2-1. 下面我们来添加自定义 IP 核，请按照下面的图示方法一步一步进行：在页面上侧选择 **Hardware** 一项，并在下拉菜单中点击里面的 **Create or Import Peripheral** 一栏。

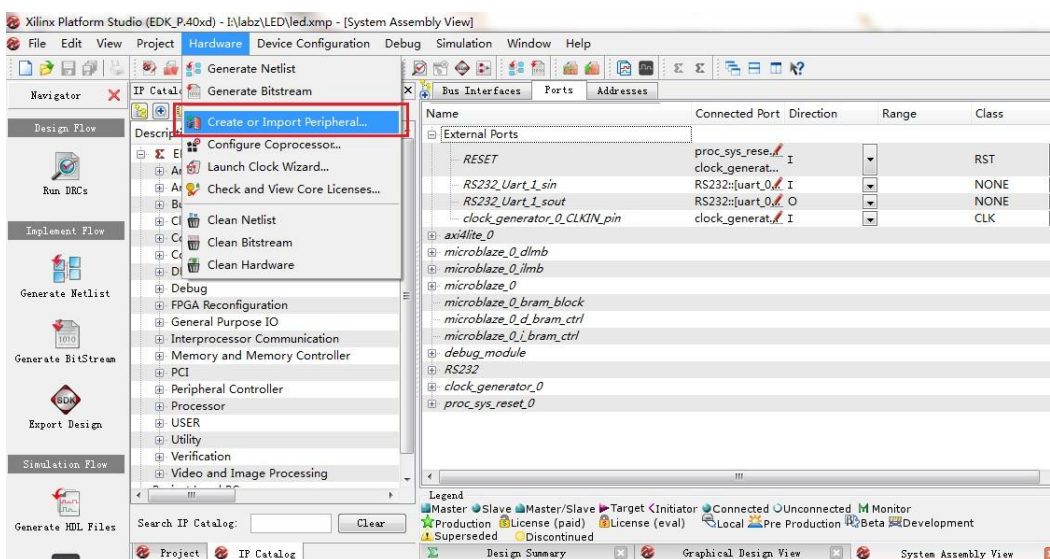


图 11: 添加自定义 IP

2-2-2. 如图所示设置自定义 IP 核的各项参数。

(1) 欢迎界面，直接点击 **next** 即可。

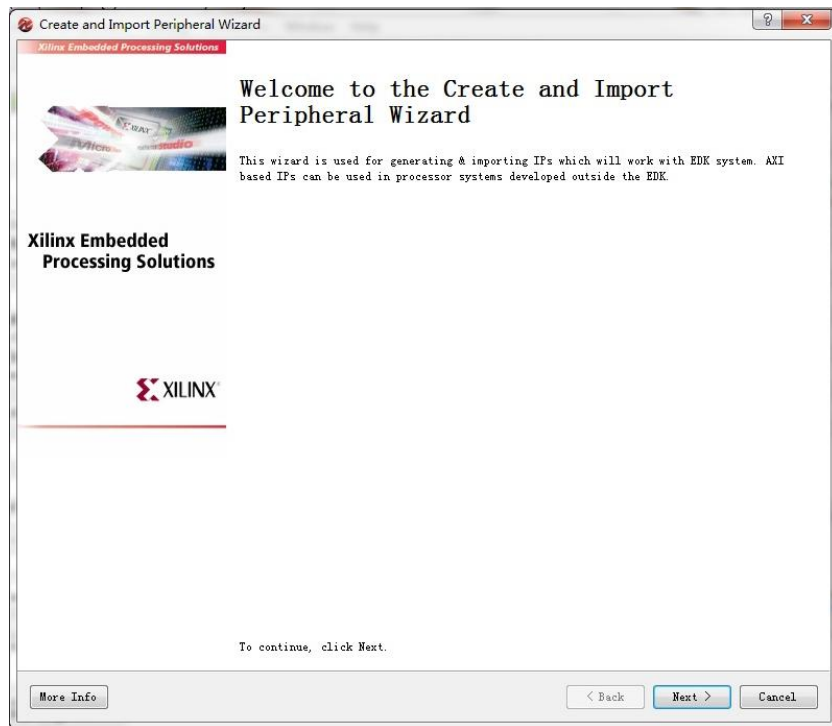


图 12：自定义 IP 设置的欢迎界面

(2) 按照图示选择创建一个模板。

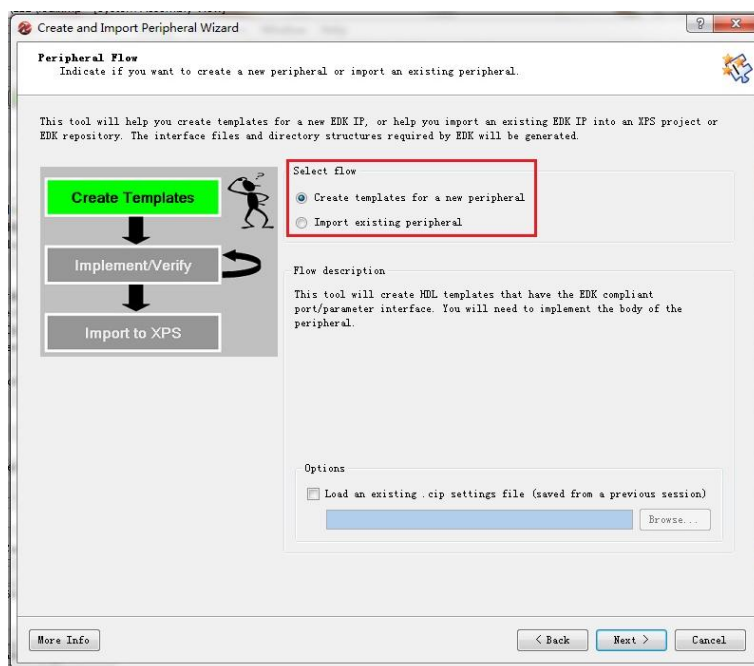


图 13：建立工程模板

(3) 因为我们需要将建立的 IP 核导入到 XPS 工程中，所以如图所示进行选择。

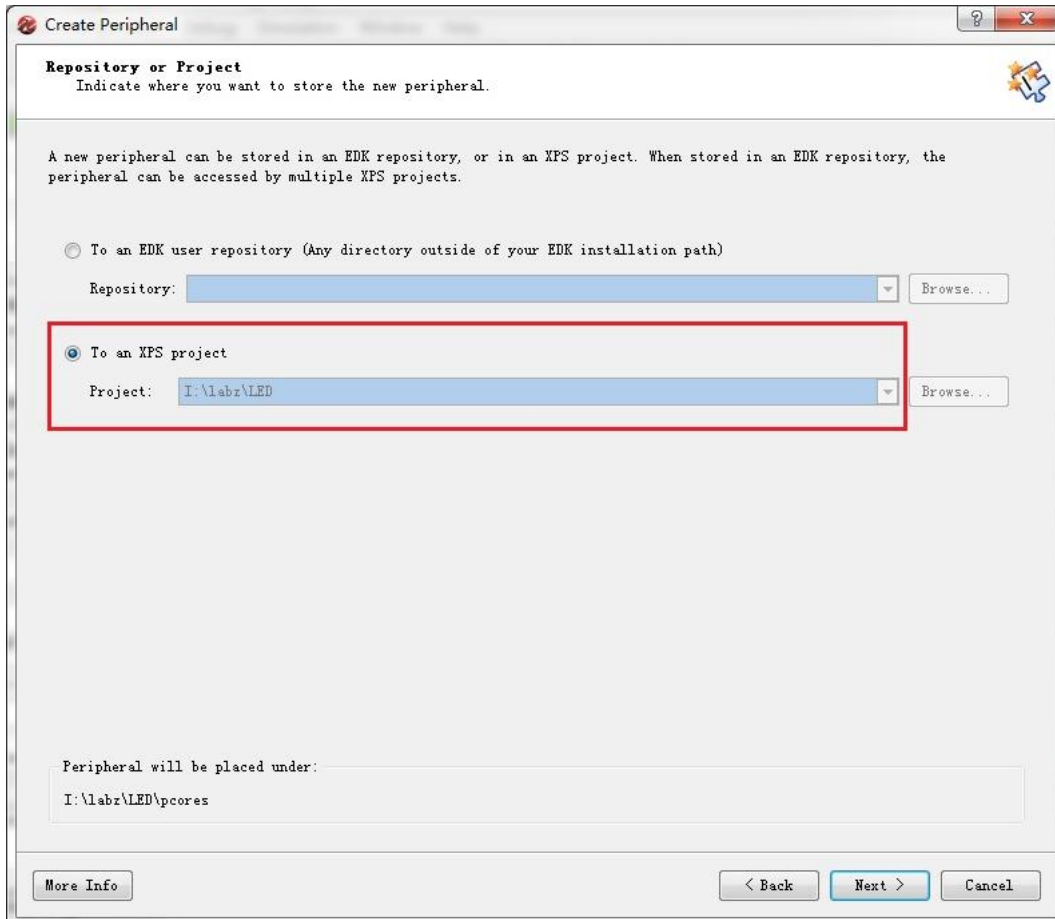


图 14: 选择导入的工程

(4) 下面我们来命名自定义 IP 的名称和版本，如图所示：

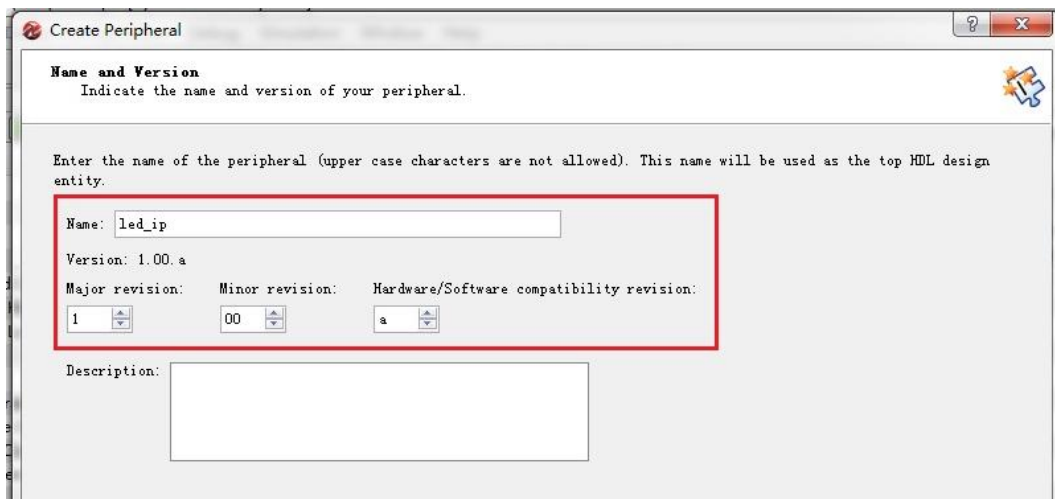


图 15: 命名自定义 IP 的名称和版本

(5) 如图所示，我们来选择总线协议。

简单来说，AXI4-Lite 适合只有少量寄存器的设计；AXI4 适合对拥有 Memory，需要频繁读写操作的设计；AXI4-Stream 适合具有一对一通道、传输大量数据的设计。

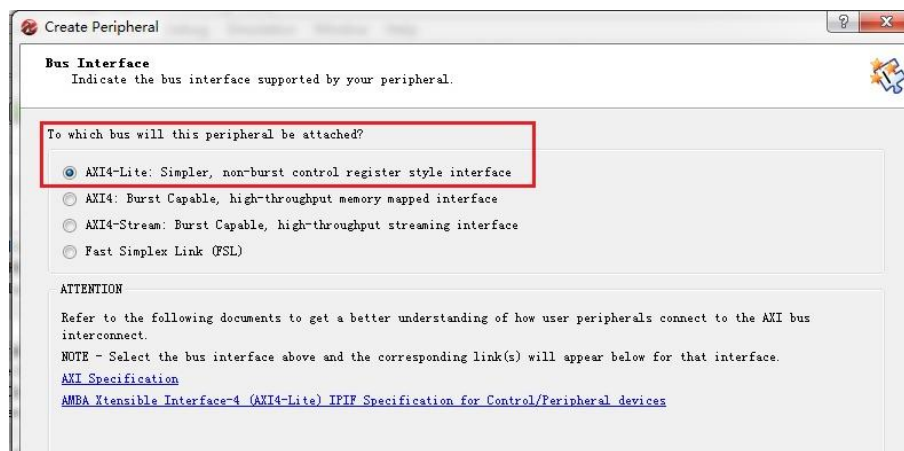


图 16：选择总线协议

(6) 我们把默认的 Include data phase timer 取消掉，因为设计比较简单，不需要添加时间响应上的要求。

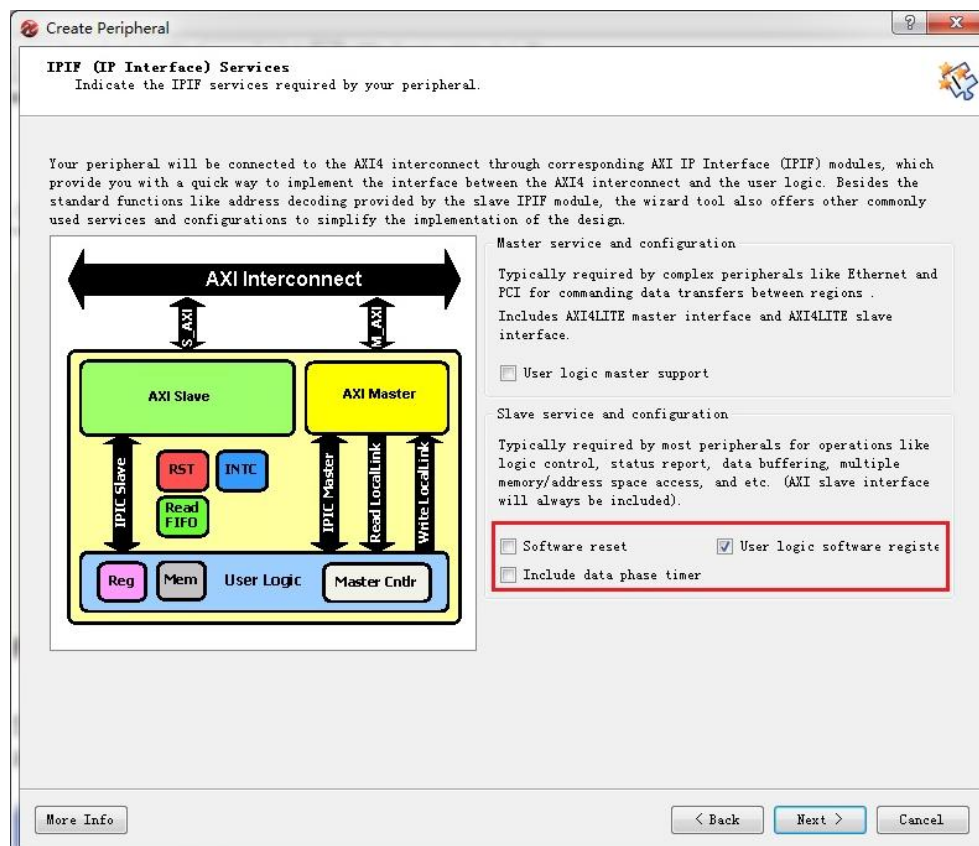


图 17：取消时间相应要求

(7) 下面我们设置寄存器数量为 2。

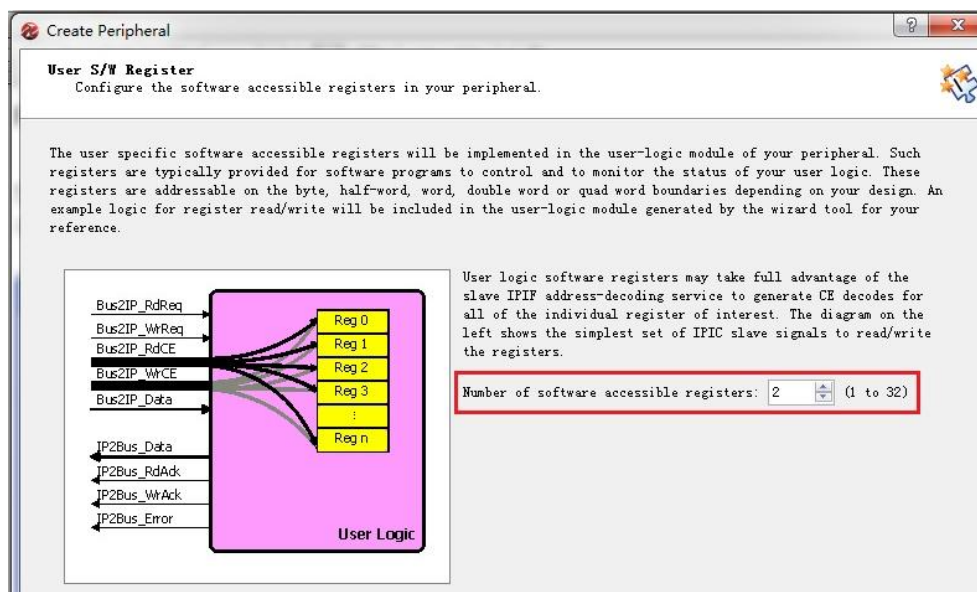


图 18: 设置寄存器数量

(8) 现在显示的是主设备与 IP 核之间的信号设置。BUS2IP 代表从总线输入到 IP 核中的信号，IPP2BUS 代表从 IP 核输出到总线的信号。我们采用默认设置就好。

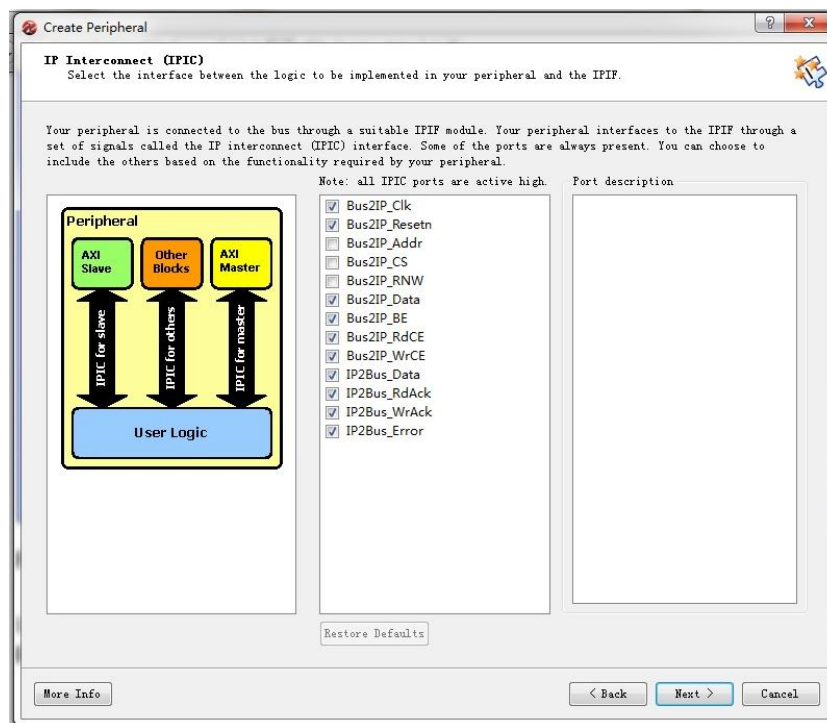


图 19: 信号设置情况

(9) 我们取消总线仿真的设定，因为我们不需要这个功能。

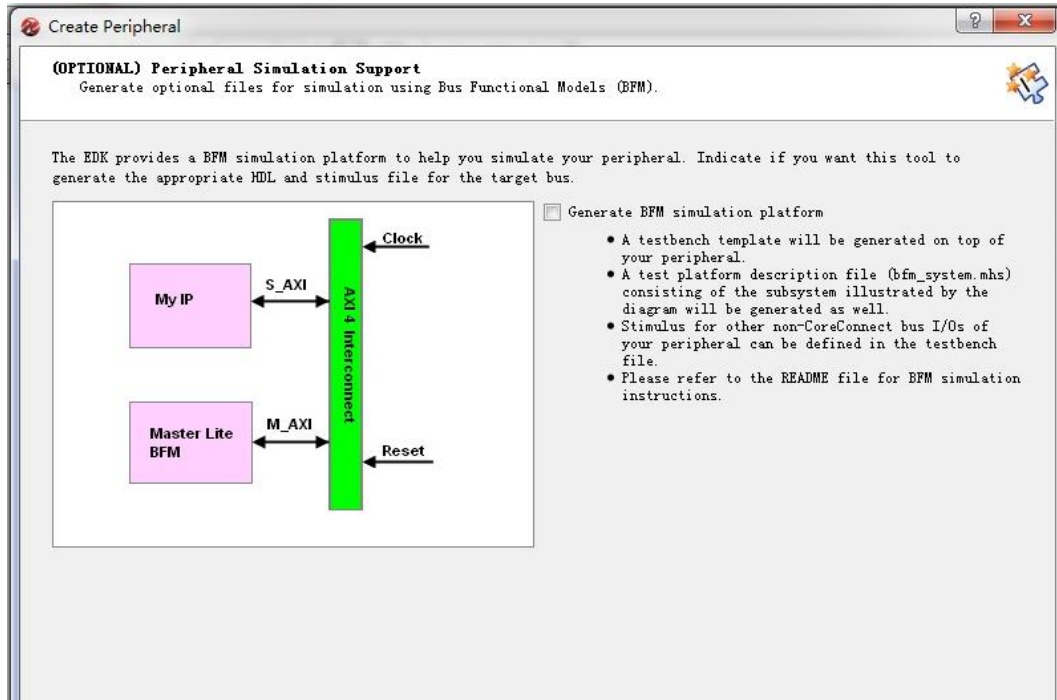


图 20: 取消总线仿真

(10) 下面我们要按照图示选择采用软件驱动模板，

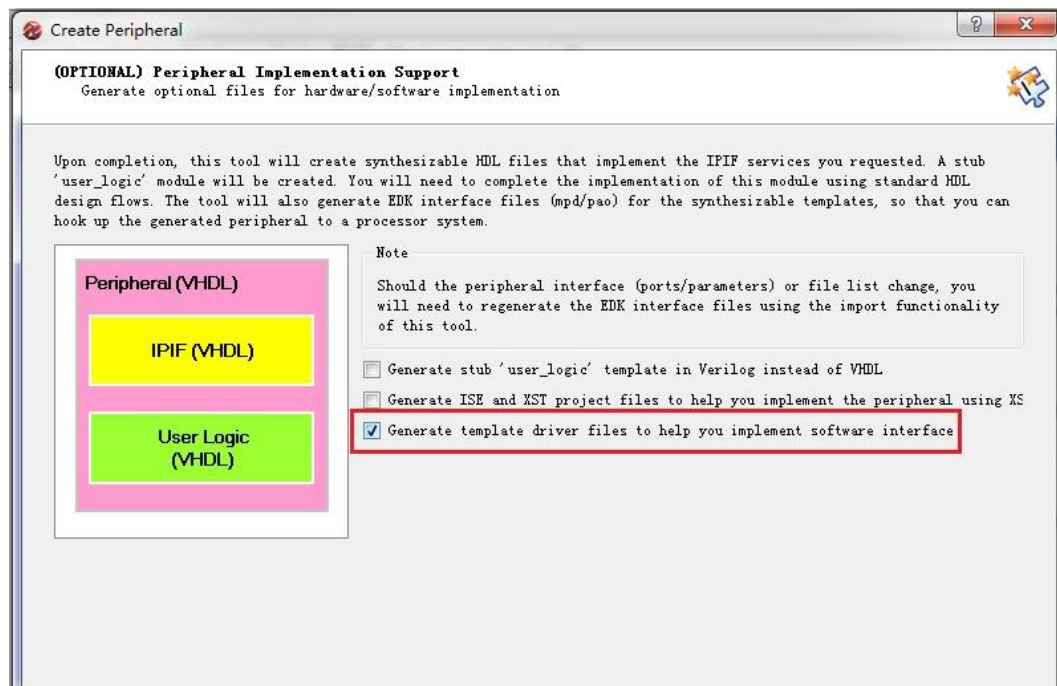


图 21: 采用软件驱动模板



(11) 单击 **Finish**，完成自定义 IP 核的配置。

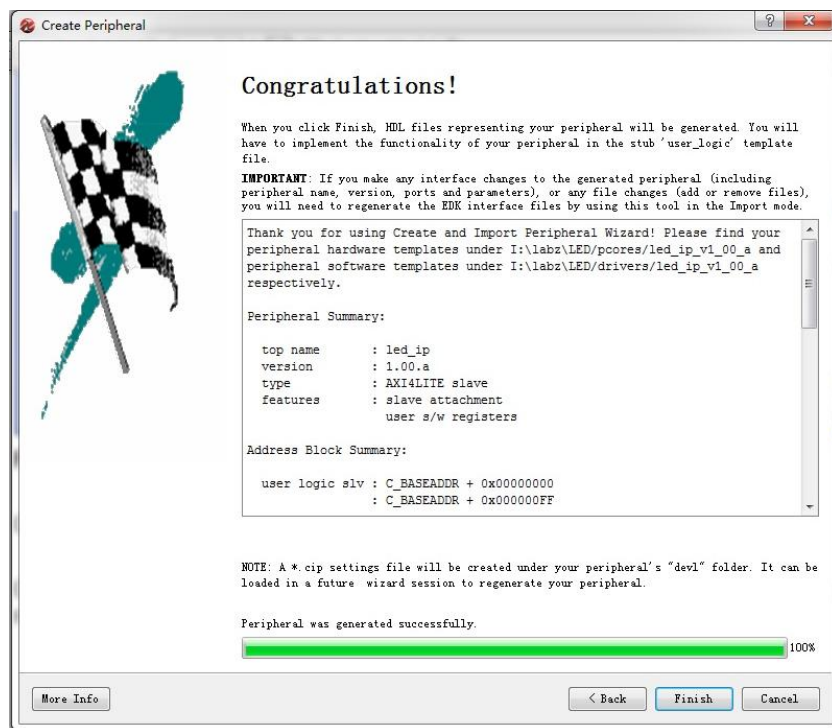


图 22: 完成自定义 IP 核的配置

(12) 完成后的效果，注意到屏幕左边 **IP Catalog** 界面的 **Project Local PCores** 的 **User** 一栏多出了我们自定义的 **LED\_IP** 选项。

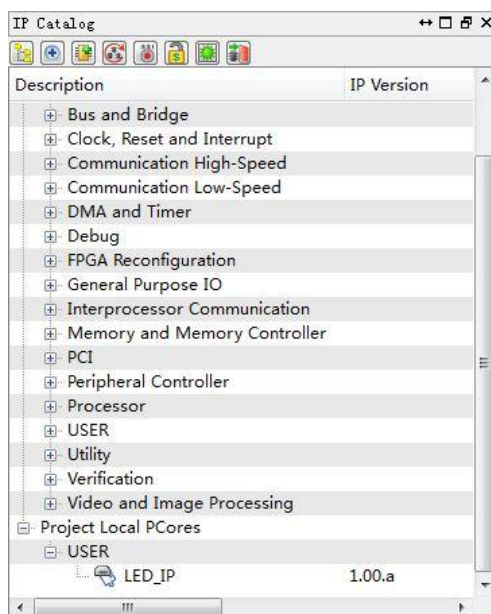


图 23: 自定义 IP 完成后的效果

### 2-2-3. 修改相关文件

(1) 首先到 I:\labz\liushuideng\pcores\led\_ip\_v1\_00\_a\data 文件夹下对 led\_ip\_v2\_1\_0.mpd 文件进行编辑，添加内容如下图所示，修改完成后保存文件：

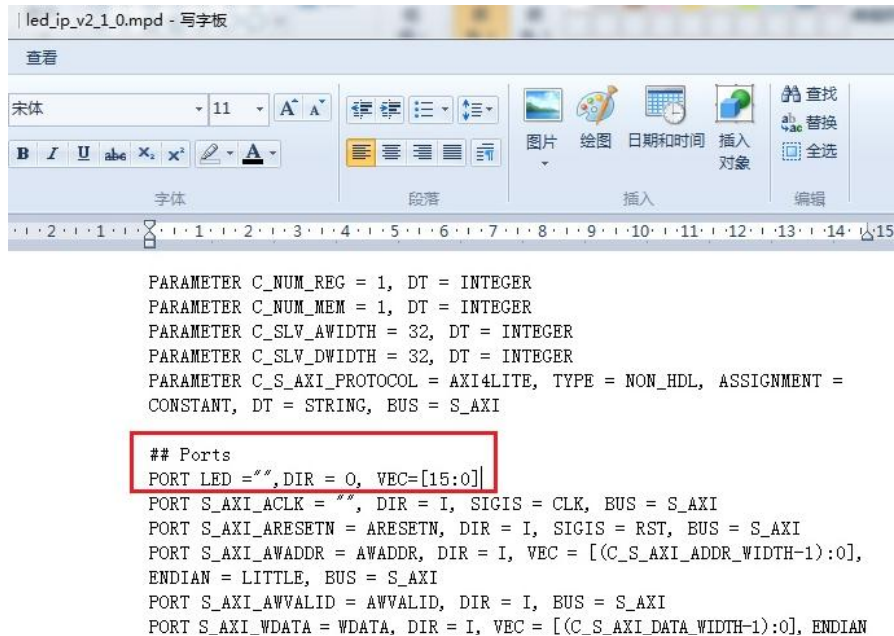
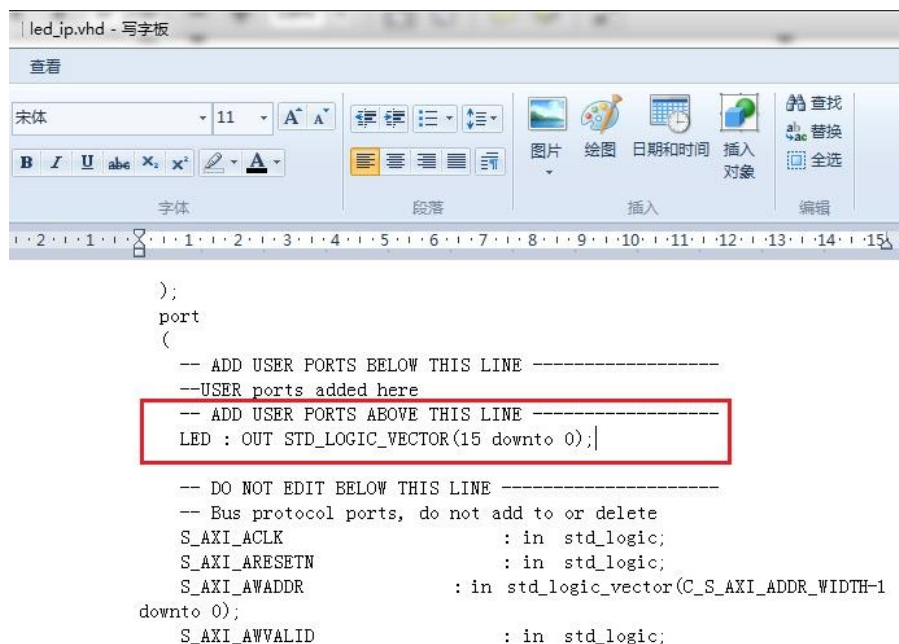


图 24: led\_ip\_v2\_1\_0.mpd 文件添加内容

(2) 接下来到 I:\labz\liushuideng\pcores\led\_ip\_v1\_00\_a\hdl\vhdl 文件夹下对 led\_ip.vhd 文件进行编辑，添加语句如下图所示，修改完成后保存文件：





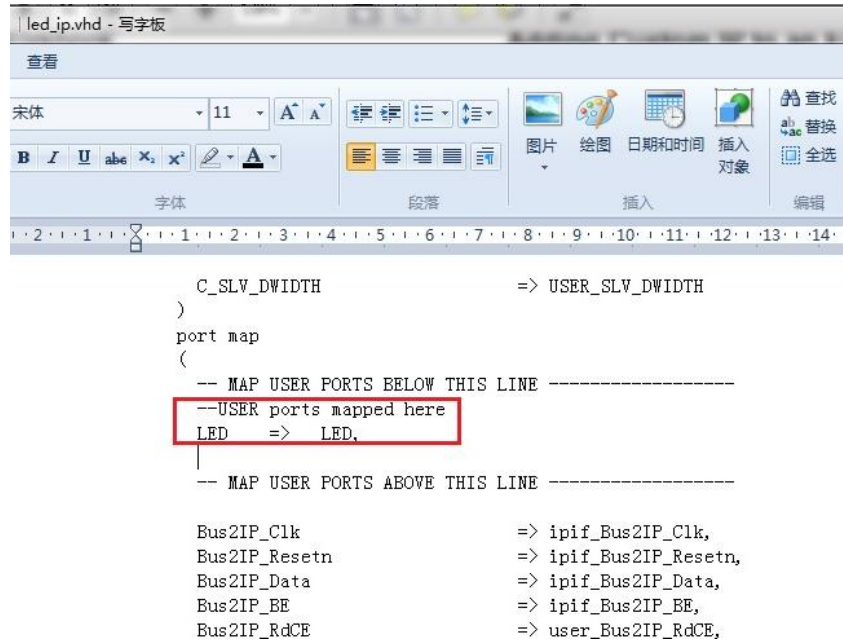
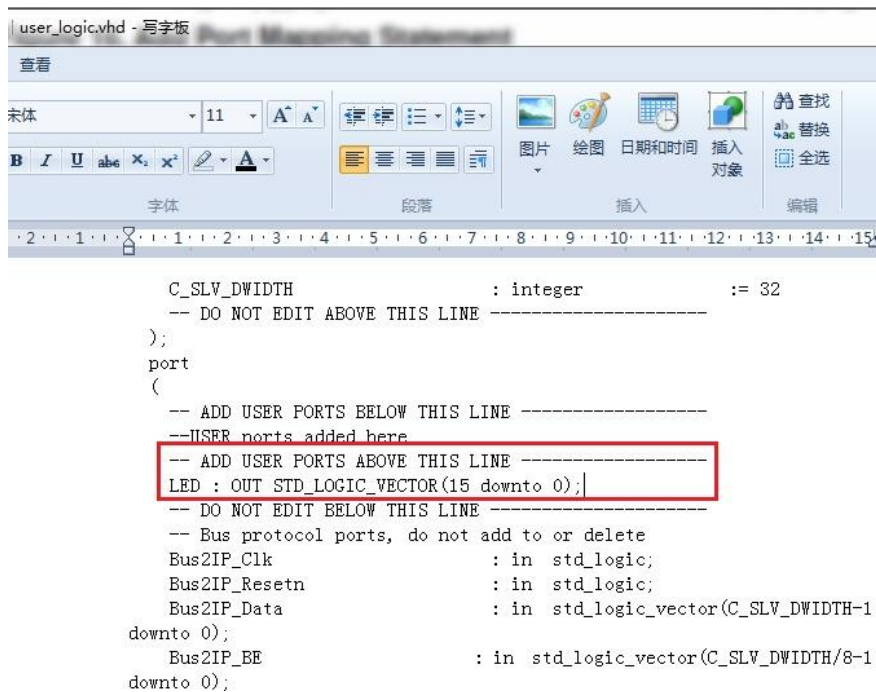


图 25: led\_ip.vhd 文件添加内容

(3) 最后我们还是在该文件夹下对 **user\_logic.vhd** 文件进行编辑，添加语句如下图所示，修改完成后保存文件：



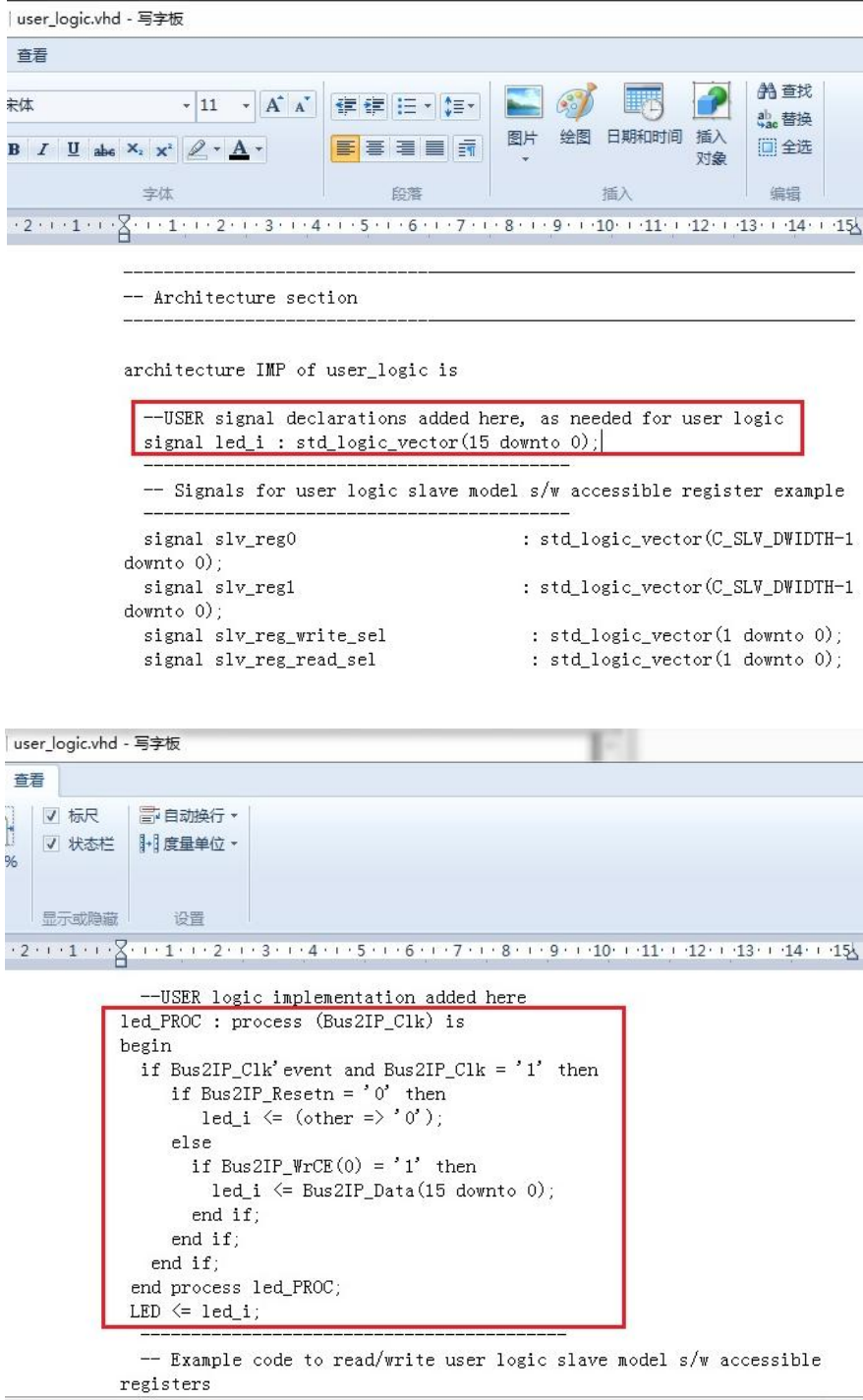


图 26: user\_logic.vhd 文件添加内容

#### 2-2-4. 对工程进行重新扫描，以显示新添加的器件。

打开屏幕上方 **Project** 选项卡，点击 **Rescan User Repositories**，从而对整个工程重新扫描。

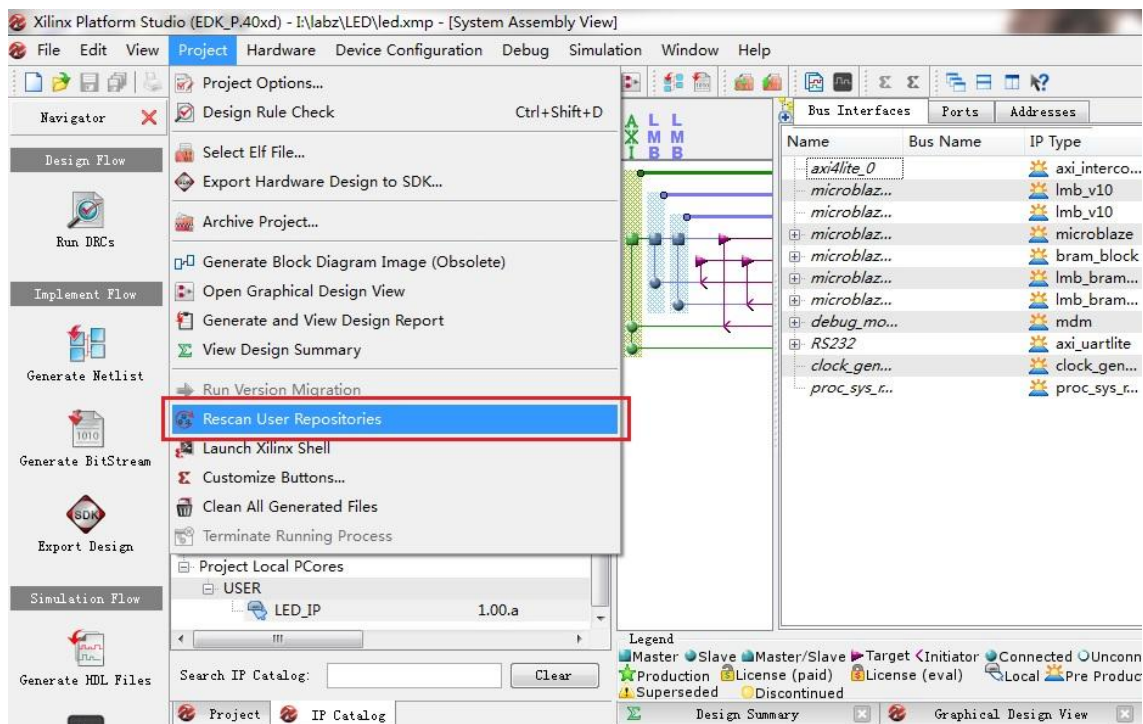


图 27：重新扫描工程

## 2-2-5. 添加 LED 的 IP 核到工程中。

如图所示，选择屏幕左方的 **IP Catalog** 中的 **Project Local PCores** 一项，右键点击 **User** 中的 **LED\_IP**，选择 **Add IP** 选项。

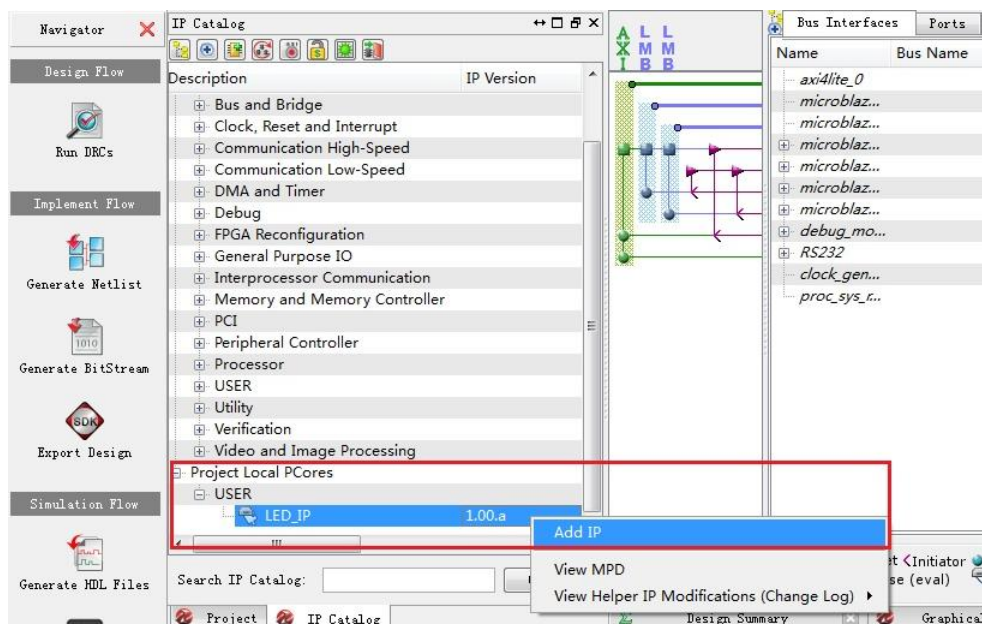


图 28：添加 IP

## 2-2-6. 查看 IP 核添加情况并连线

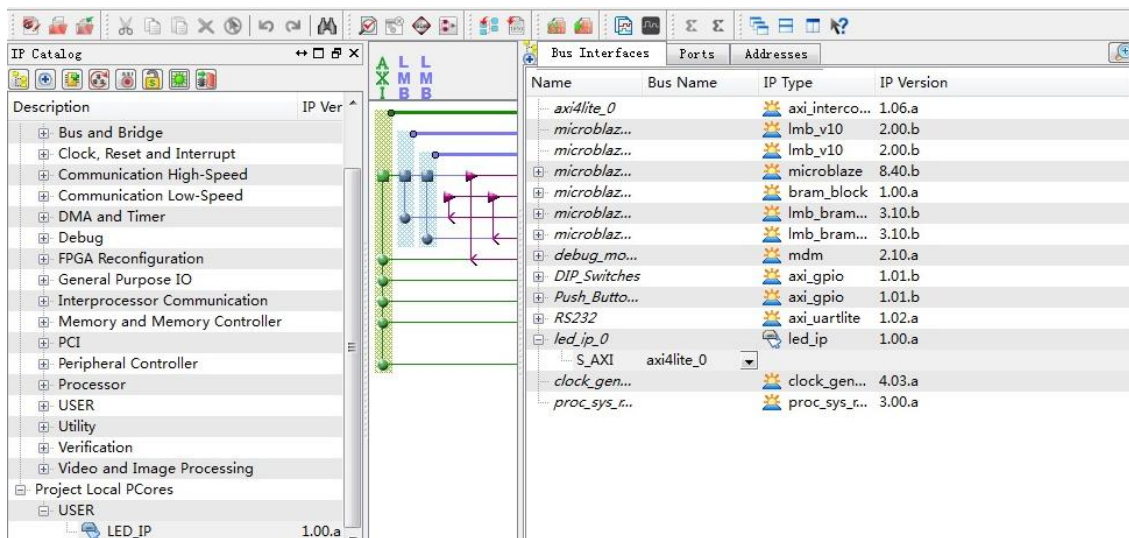


图 29: 添加 IP 核后的情况

接下来我们点击 **Port** 界面，找到 **led\_ip\_0**，右键单击 **LED** 一项并选择 **Make External**。

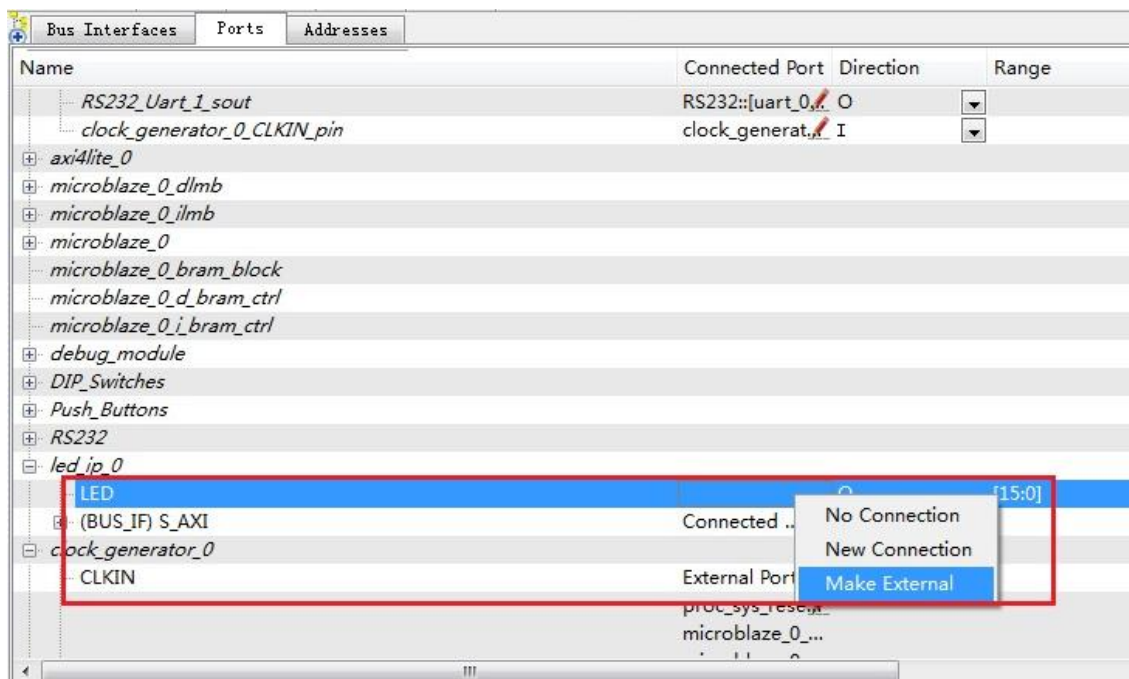
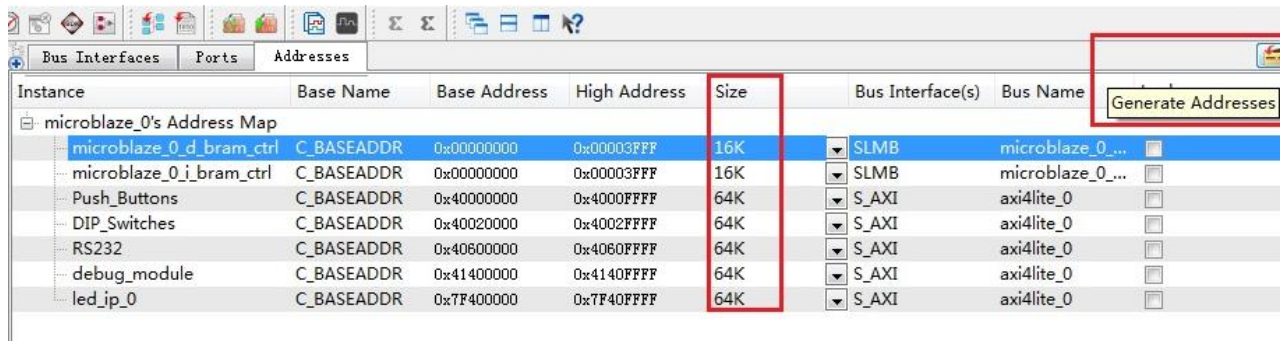


图 30: 为 IP 核连线

## 2-2-7. 修改寄存器大小，并生成地址。

按照图示大小修改各个部件的容量大小，并单击屏幕右上方的黄色小方格，为各部件自动分配地址。



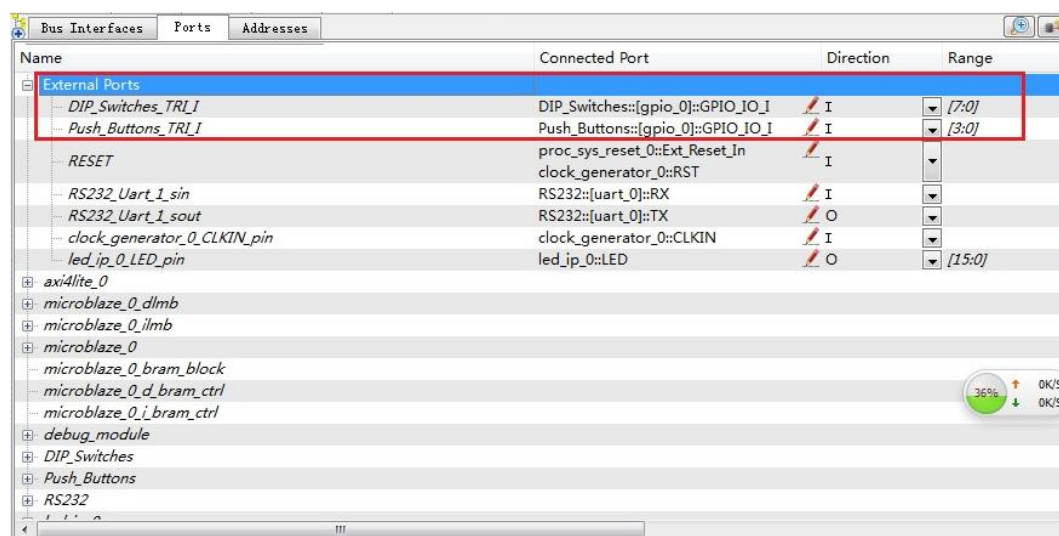


Instance	Base Name	Base Address	High Address	Size	Bus Interface(s)	Bus Name
microblaze_0's Address Map						
microblaze_0_d_bram_ctrl	C_BASEADDR	0x00000000	0x00003FFF	16K	SLMB	microblaze_0_...
microblaze_0_i_bram_ctrl	C_BASEADDR	0x00000000	0x00003FFF	16K	SLMB	microblaze_0_...
Push_Buttons	C_BASEADDR	0x40000000	0x4000FFFF	64K	S_AXI	axi4lite_0
DIP_Switches	C_BASEADDR	0x40020000	0x4002FFFF	64K	S_AXI	axi4lite_0
RS232	C_BASEADDR	0x40600000	0x4060FFFF	64K	S_AXI	axi4lite_0
debug_module	C_BASEADDR	0x41400000	0x4140FFFF	64K	S_AXI	axi4lite_0
led_ip_0	C_BASEADDR	0x7F400000	0x7F40FFFF	64K	S_AXI	axi4lite_0

图 31：修改容量&分配地址

2-2-8. 注意 External 中的 Name 一项，这是我们添加用户约束文件（UCF）的依据。

另外，还要特别注意我们添加的这些接口的范围，要按照图示范围那样修改好。



Name	Connected Port	Direction	Range
External Ports			
DIP_Switches_TRI_I	DIP_Switches::[gpio_0]::GPIO_IO_I	I	[7:0]
Push_Buttons_TRI_I	Push_Buttons::[gpio_0]::GPIO_IO_I	I	[3:0]
RESET	proc_sys_reset_0::Ext_Reset_In	I	
RS232_Uart_1_sin	RS232::[uart_0]::RX	I	
RS232_Uart_1_sout	RS232::[uart_0]::TX	O	
clock_generator_0_CLKIN_pin	clock_generator_0::CLKIN	I	
led_ip_0_LED_pin	led_ip_0::LED	O	[15:0]
axi4lite_0			
microblaze_0_dlm_b			
microblaze_0_ilmb			
microblaze_0			
microblaze_0_bram_block			
microblaze_0_d_bram_ctrl			
microblaze_0_i_bram_ctrl			
debug_module			
DIP_Switches			
Push_Buttons			
RS232			

图 32：修改后的 External PORT

## 第三步 添加用户约束文件&生成比特流文件

### 3-1. 打开初始 UCF 文件，根据需求进行修改

3-1-1. 在页面偏左找到 IP catalogue / Project 选项卡，双击 UCF File: data\system.ucf，ucf 文件在右侧打开

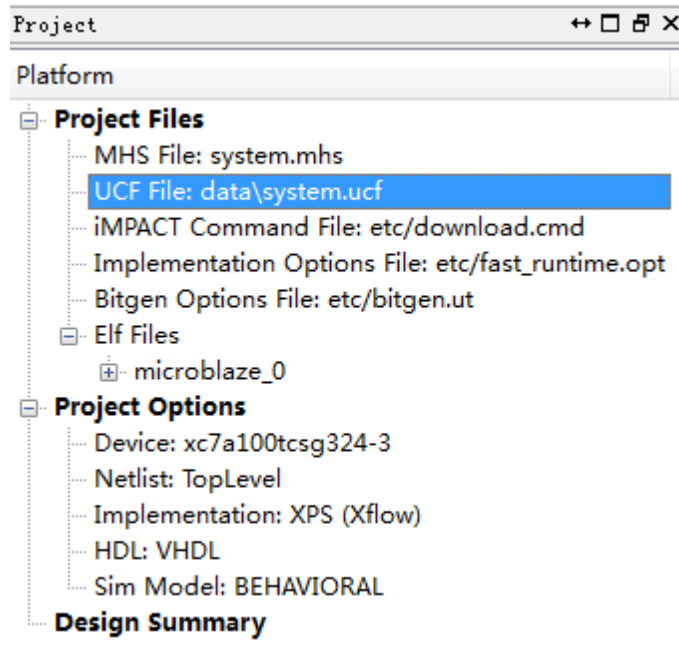
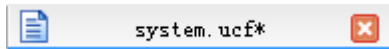


图 33: UCF 文件的位置

3-1-2. 这里我们手动输入 LOC（引脚位置）约束代码，如图 16。点击保存。

小技巧：可以通过 **ctrl+s** 的快捷键组合进行保存。如果 ucf 窗口下方的文件名旁边有\*，如图所示：



这说明 ucf 文件经过编辑但还未保存。

```

1  # Clock signal
2  NET "clock_generator_0_CLKIN_pin" LOC = "E3" | IOSTANDARD = "LVCMOS33";
3  NET "clock_generator_0_CLKIN_pin" TNM_NET = sys_clk_pin;
4  TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 100 MHz HIGH 50%;
5
6  # Switches
7  NET "DIP_Switches_TRI_I<0>" LOC = "U9" | IOSTANDARD = "LVCMOS33";
8  NET "DIP_Switches_TRI_I<1>" LOC = "U8" | IOSTANDARD = "LVCMOS33";
9  NET "DIP_Switches_TRI_I<2>" LOC = "R7" | IOSTANDARD = "LVCMOS33";
10 NET "DIP_Switches_TRI_I<3>" LOC = "R6" | IOSTANDARD = "LVCMOS33";
11 NET "DIP_Switches_TRI_I<4>" LOC = "R5" | IOSTANDARD = "LVCMOS33";
12 NET "DIP_Switches_TRI_I<5>" LOC = "V7" | IOSTANDARD = "LVCMOS33";
13 NET "DIP_Switches_TRI_I<6>" LOC = "V6" | IOSTANDARD = "LVCMOS33";
14 NET "DIP_Switches_TRI_I<7>" LOC = "V5" | IOSTANDARD = "LVCMOS33";
15 NET "RESET" LOC = "P4" | IOSTANDARD = "LVCMOS33";
16
17 # LEDs
18 NET "led_ip_0_LED_pin<0>" LOC = "T8" | IOSTANDARD = "LVCMOS33";
19 NET "led_ip_0_LED_pin<1>" LOC = "V9" | IOSTANDARD = "LVCMOS33";
20 NET "led_ip_0_LED_pin<2>" LOC = "R8" | IOSTANDARD = "LVCMOS33";
21 NET "led_ip_0_LED_pin<3>" LOC = "T6" | IOSTANDARD = "LVCMOS33";
22 NET "led_ip_0_LED_pin<4>" LOC = "T5" | IOSTANDARD = "LVCMOS33";
23 NET "led_ip_0_LED_pin<5>" LOC = "T4" | IOSTANDARD = "LVCMOS33";
24 NET "led_ip_0_LED_pin<6>" LOC = "U7" | IOSTANDARD = "LVCMOS33";
25 NET "led_ip_0_LED_pin<7>" LOC = "U6" | IOSTANDARD = "LVCMOS33";
26 NET "led_ip_0_LED_pin<8>" LOC = "V4" | IOSTANDARD = "LVCMOS33";
27 NET "led_ip_0_LED_pin<9>" LOC = "U3" | IOSTANDARD = "LVCMOS33";
28 NET "led_ip_0_LED_pin<10>" LOC = "V1" | IOSTANDARD = "LVCMOS33";
29 NET "led_ip_0_LED_pin<11>" LOC = "R1" | IOSTANDARD = "LVCMOS33";
30 NET "led_ip_0_LED_pin<12>" LOC = "P5" | IOSTANDARD = "LVCMOS33";
31 NET "led_ip_0_LED_pin<13>" LOC = "U1" | IOSTANDARD = "LVCMOS33";
32 NET "led_ip_0_LED_pin<14>" LOC = "R2" | IOSTANDARD = "LVCMOS33";
33 NET "led_ip_0_LED_pin<15>" LOC = "P2" | IOSTANDARD = "LVCMOS33";
34
35 # Buttons
36 NET "Push_Buttons_TRI_I<0>" LOC = "F15" | IOSTANDARD = "LVCMOS33";
37 NET "Push_Buttons_TRI_I<1>" LOC = "T16" | IOSTANDARD = "LVCMOS33";
38 NET "Push_Buttons_TRI_I<2>" LOC = "R10" | IOSTANDARD = "LVCMOS33";
39 NET "Push_Buttons_TRI_I<3>" LOC = "V10" | IOSTANDARD = "LVCMOS33";
40
41 # USB-RS232 Interface
42 NET "RS232_Uart_1_sin" LOC = "C4" | IOSTANDARD = "LVCMOS33";
43 NET "RS232_Uart_1_sout" LOC = "D4" | IOSTANDARD = "LVCMOS33";

```

图 34: UCF 文件

## 3-2. 生成比特流文件

3-2-1. 在页面左边，依次点击 **Generate Netlist** 和 **Generate BitStream**，如图 35 所示。如果没有错误显示则说明硬件工程建立成功。

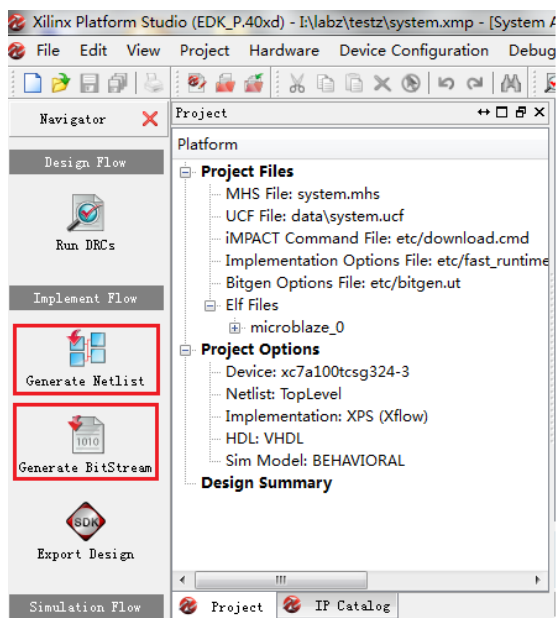


图 35：生成比特流文件

等待程序运行结束，如果屏幕下方的提示栏中出现“Done”的字样，就说明工程设计没有问题。

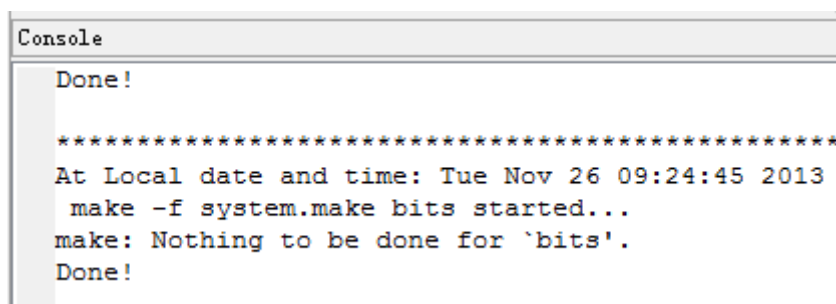


图 36：验证成功结果