25A, No.1369,Dongfang Road, Shanghai
Tel:+86-021-58893151 Fax: +86-021-58894598

Lab13: “网口”

——Xilinx EDK 设计基本流程

*基于 **Nexys 4 FPGA** 平台*

Lab 13: “网口”

实验简介

本实验旨在使读者学会 Xilinx 的 XPS 和 SDK 工具的使用，同时完成一个带有网口应用的简单例程。

实验目标

在完成本实验后，您将学会：

- XPS 工具的使用流程，从新建工程到导入到 SDK。
- SDK 工具的使用流程，从导入到 SDK 到在板卡上运行 C 语言程序。
- 学会使用定制网口的流程

实验过程

本实验旨在指导读者使用 Xilinx 的 XPS 工具，调用串口的 IP 核，并将导入到 SDK，调用这个 IP 核，在串口上显示板卡 xyz 三轴加速度的数据，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．将工程导入到 SDK

5．在 SDK 中添加 c 语言源程序

6．在 Nexys 4 上进行测试验证

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。）。

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 3**.

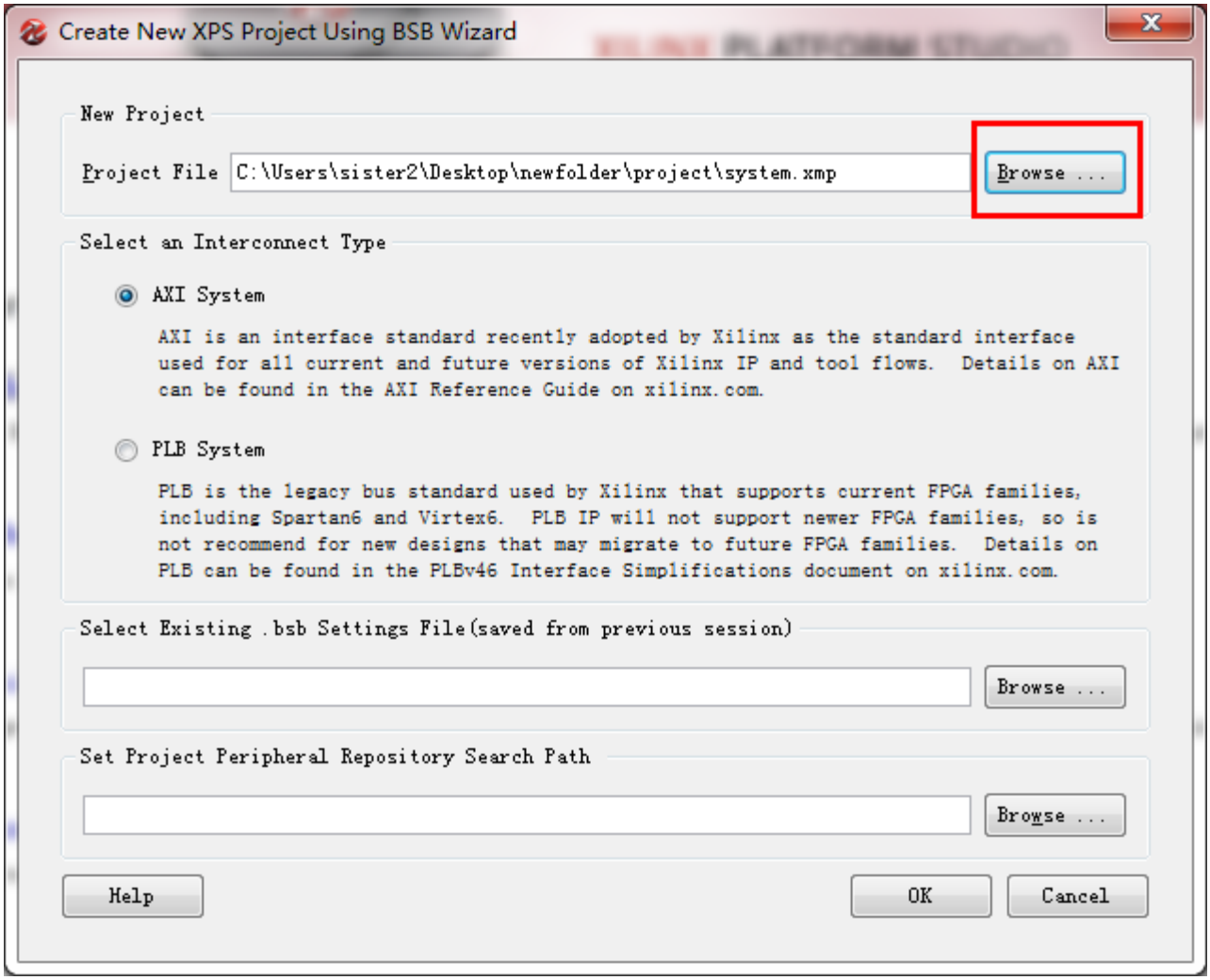


图 3：新工程建立向导

1-1-3.如图 3，在新工程建立向导对话框中点击 browse 选择工程存放的路径，这里以 c:\users\sister2\desktop\newfolder\project\为例，选择完路径后 Project File 栏中的路径变为 **I:\Jungle\verilog_projects\Digilent\ethernet\system.xmp**。点击 OK。.

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

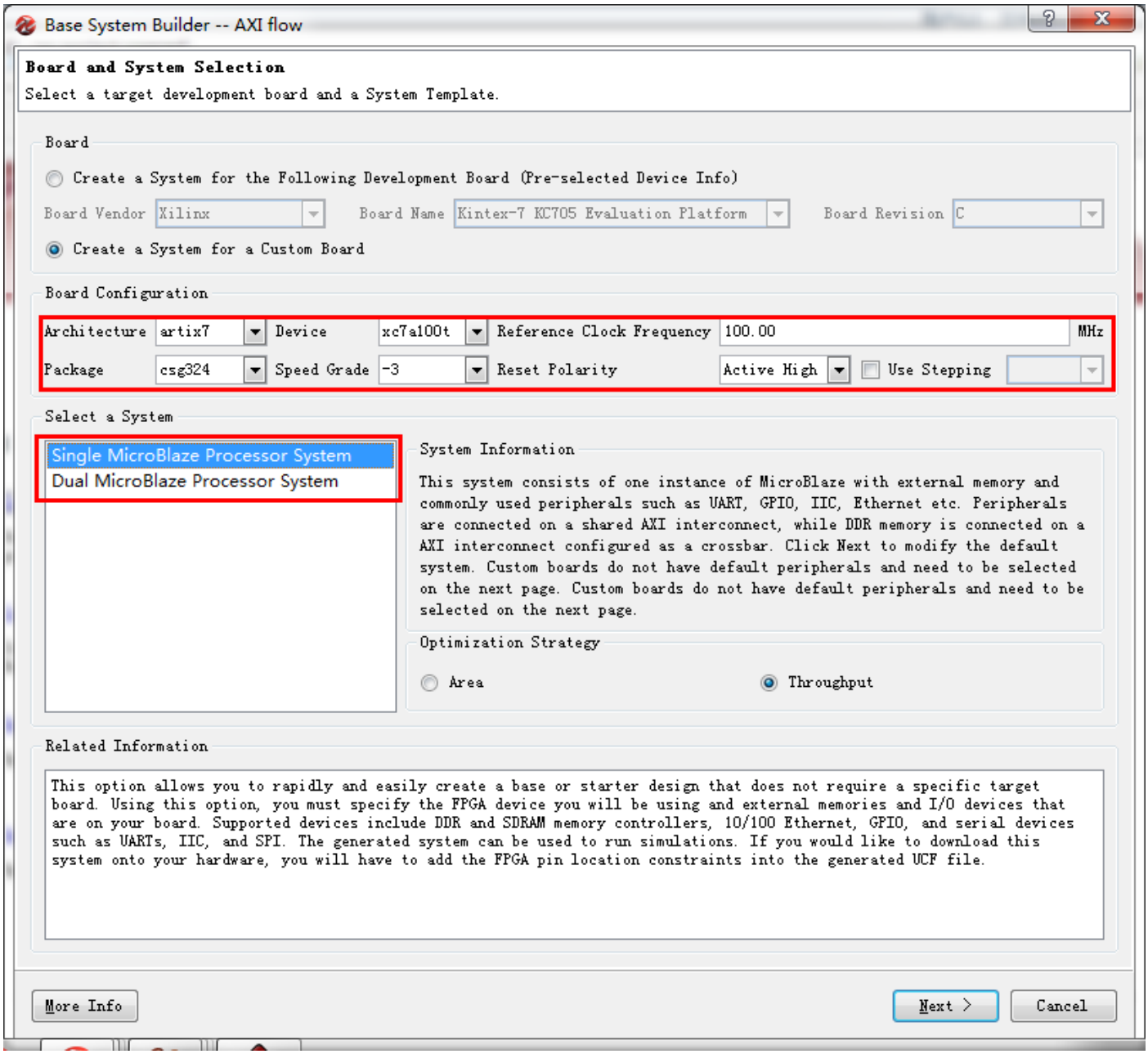
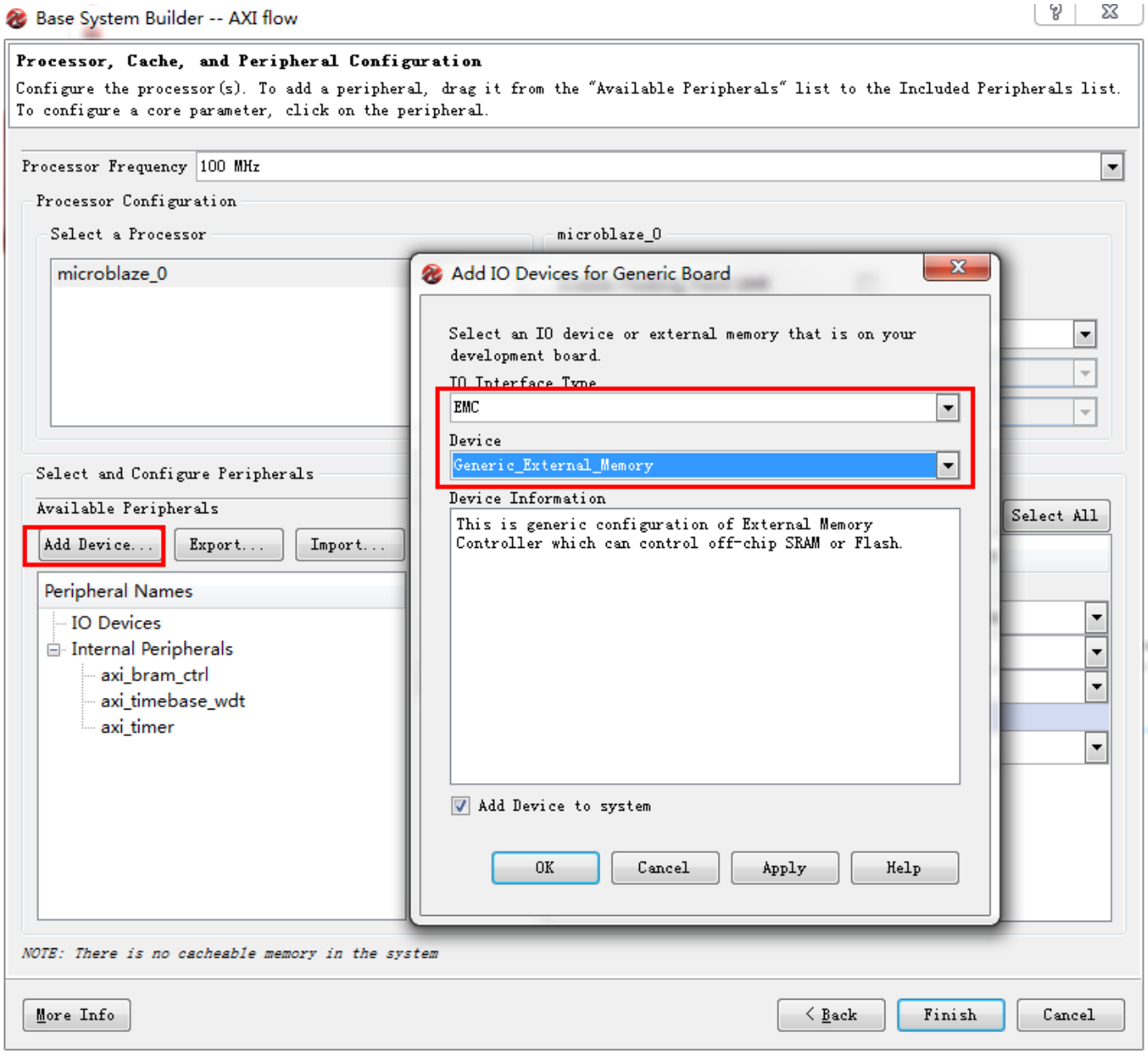


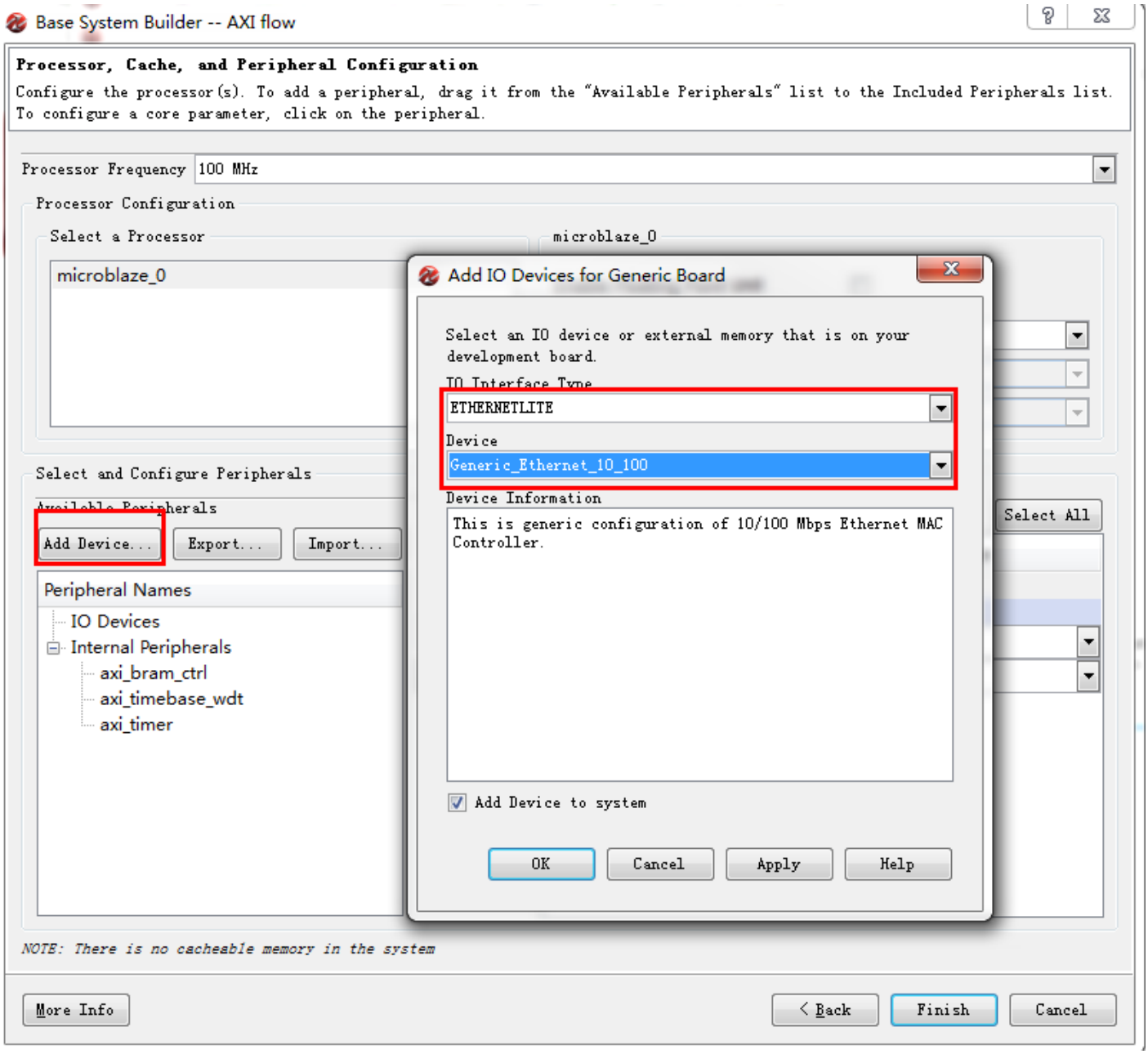
图 4：新工程参数设置

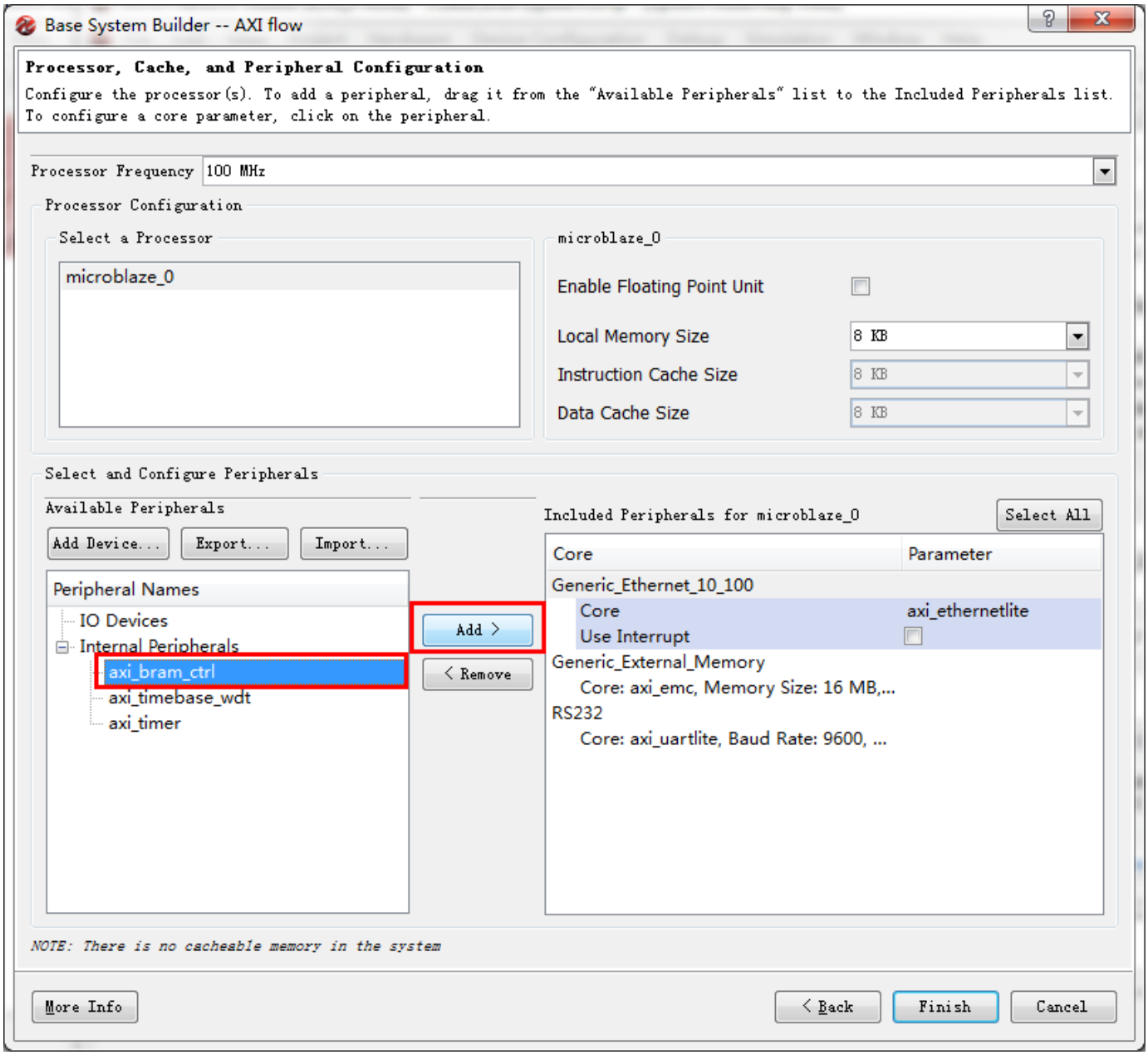
1-1-5. 在接下来出现的页面中选择要添加的 IP 核，并设置 IP 核的参数：

单击 Select and configure Peripherals 下的 Add Device…

然后按照下图所示，依次添加 generic external memory、ethernet、bram controller、uart


Base System Builder -- AXI flow
Processor, Cache, and Peripheral Configuration
Configure the processor(s). To add a peripheral, drag it from the "Available Peripherals" list to the Included Peripherals list.
To configure a core parameter, click on the peripheral.
Processor Frequency
100 MHz
Processor Configuration
Select a Processor
microblaze_0
microblaze_0
Add IO Devices for Generic Board
Select an IO device or external memory that is on your development board.
IO Interface Type
EMC
Device
Generic_External_Memory
Device Information
This is generic configuration of External Memory Controller which can control off-chip SRAM or Flash.
Add Device to system
OK
Cancel
Apply
Help
Select and Configure Peripherals
Available Peripherals
Add Device...
Export...
Import...
Select All
Peripheral Names
IO Devices
Internal Peripherals
axi_bram_ctrl
axi_timebase_wdt
axi_timer
NOTE: There is no cacheable memory in the system
More Info
< Back
Finish
Cancel


Base System Builder -- AXI flow
Processor, Cache, and Peripheral Configuration
Configure the processor(s). To add a peripheral, drag it from the "Available Peripherals" list to the Included Peripherals list.
To configure a core parameter, click on the peripheral.
Processor Frequency
100 MHz
Processor Configuration
Select a Processor
microblaze_0
microblaze_0
Add IO Devices for Generic Board
Select an IO device or external memory that is on your development board.
IO Interface Type
ETHERNETLITE
Device
Generic_Ethernet_10_100
Device Information
This is generic configuration of 10/100 Mbps Ethernet MAC Controller.
Add Device to system
OK
Cancel
Apply
Help
Select and Configure Peripherals
Add Device...
Export...
Import...
Select All
Peripheral Names
IO Devices
Internal Peripherals
axi_bram_ctrl
axi_timebase_wdt
axi_timer
NOTE: There is no cacheable memory in the system
More Info
< Back
Finish
Cancel



1-1-6. 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，然后选择 finish 完成.

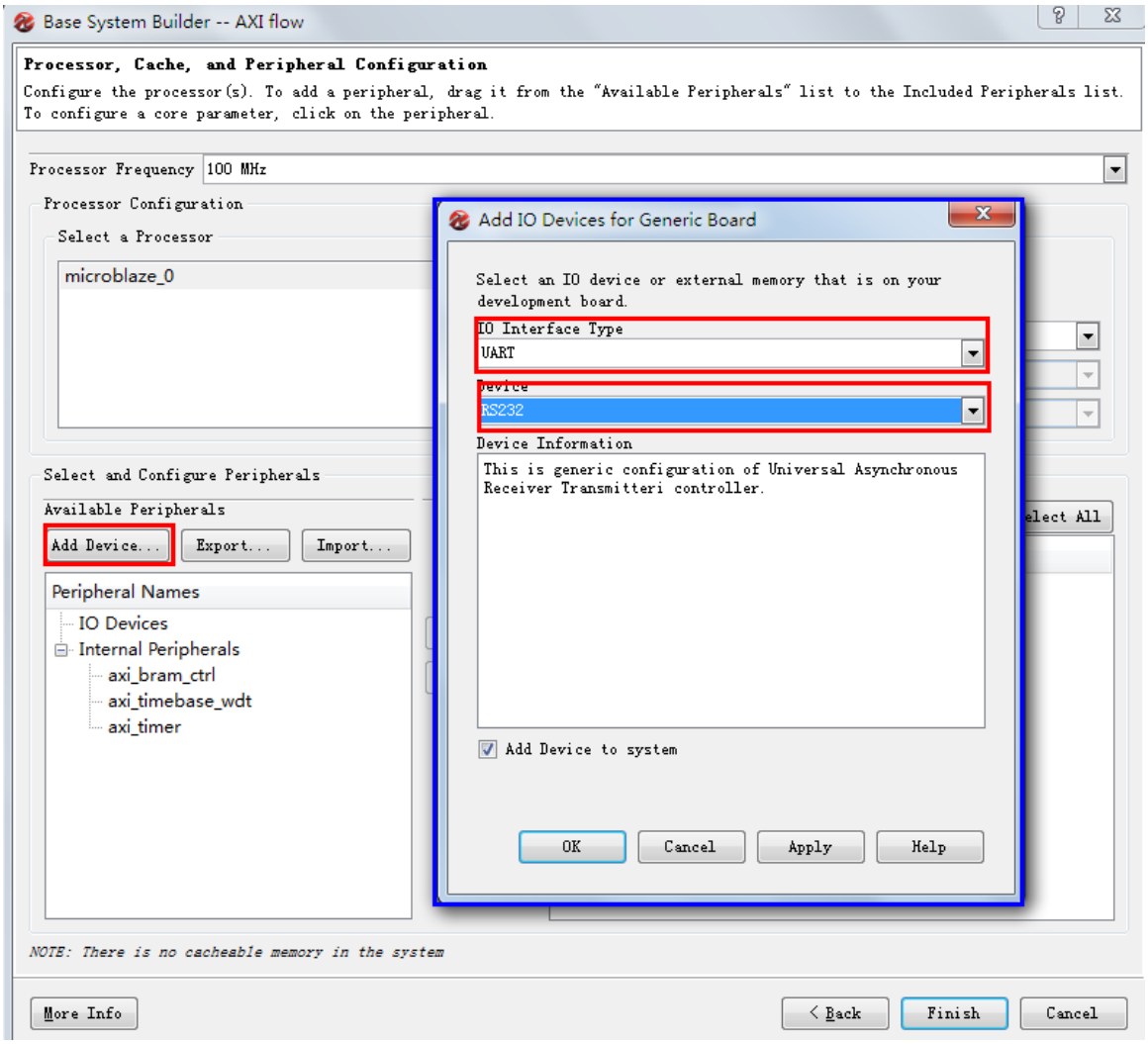


图 5. 添加串口的 IP

1-1-7. 注意,串口的默认波特率设置为 9600。在这个试验中不做修改。以后的设计中根据需要进行调整。但是为了确保串口的正常通讯，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。

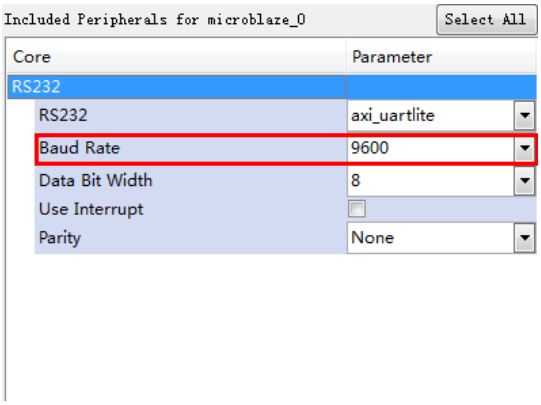
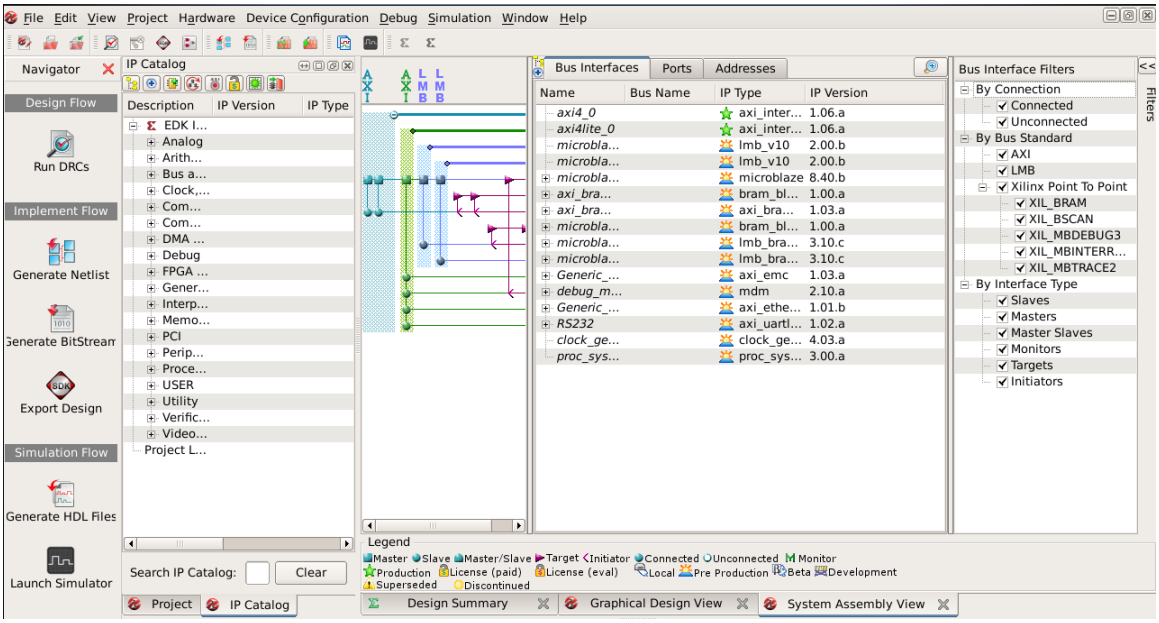


图 6. 串口的设置

第二步 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. 点击 finish 之后会看到如下界面



2-1-2. 选中 Ports 选项卡（展开 External Port），如图 7.

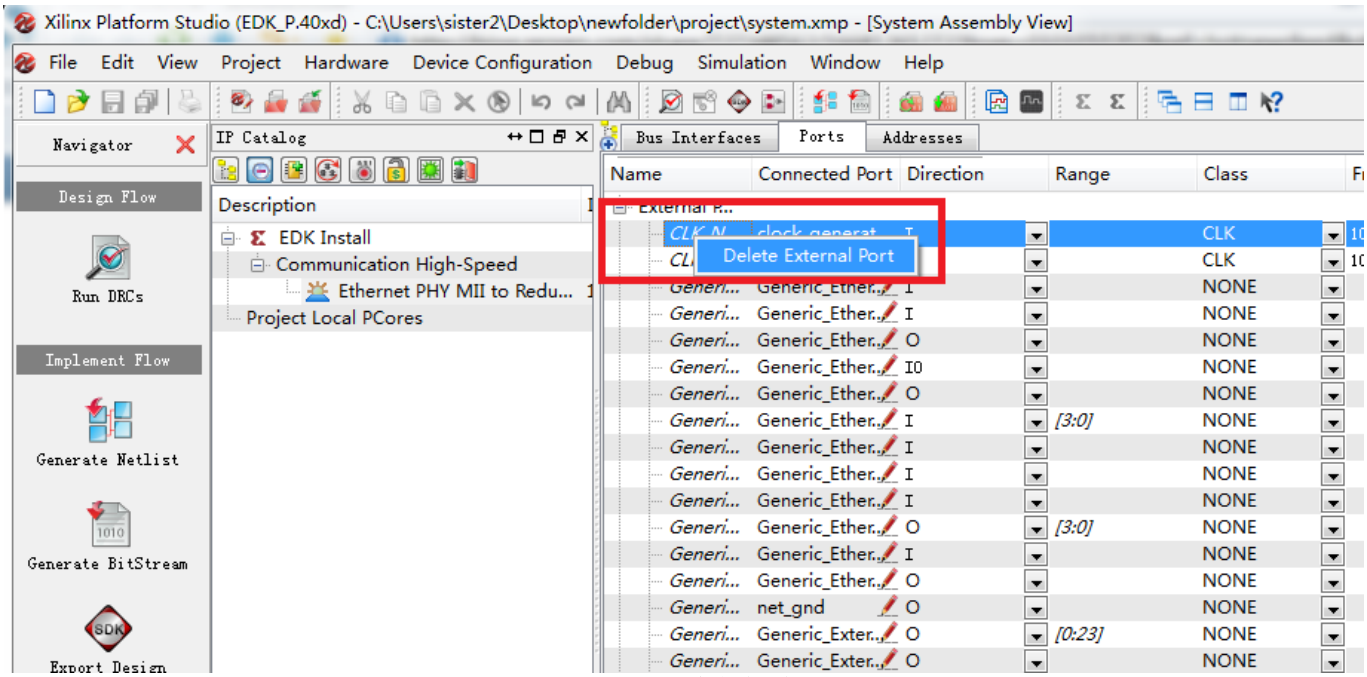


图 7：PORT 选项卡的初始状况

2-1-3. 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 8.

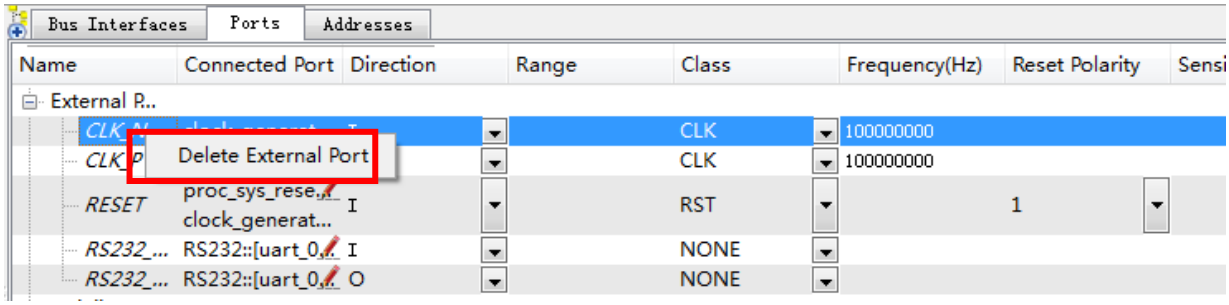


图 8：源文件添加后的 ISE 用户界面

2-1-4. 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**

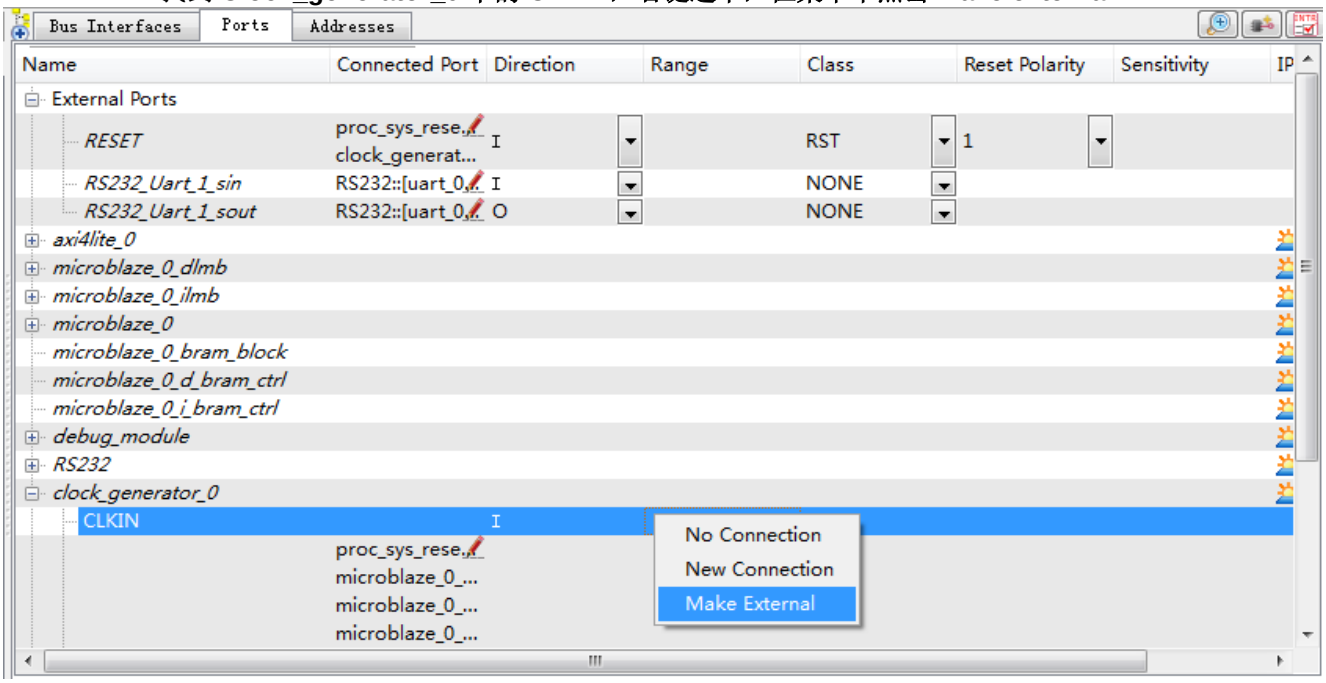
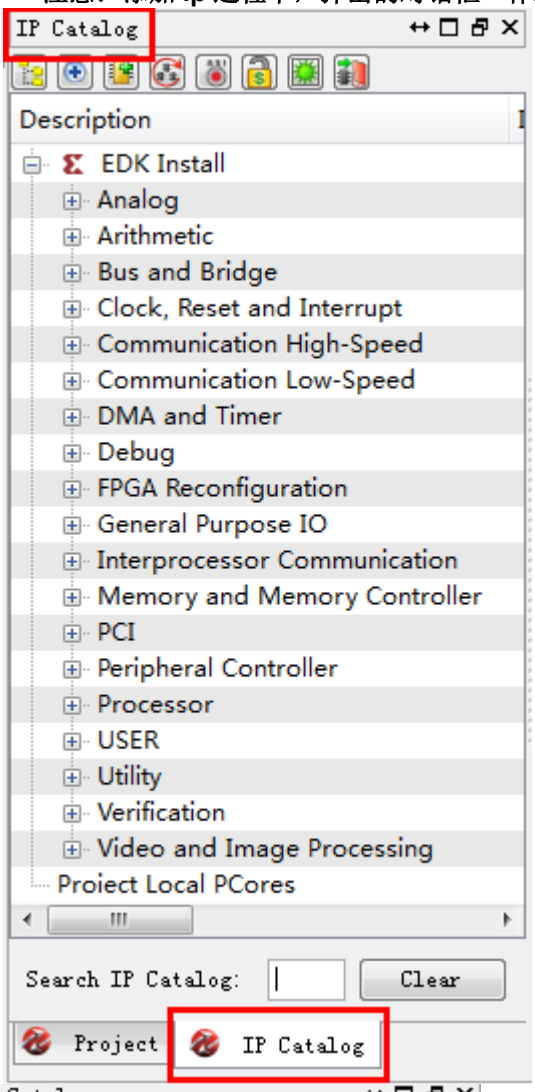
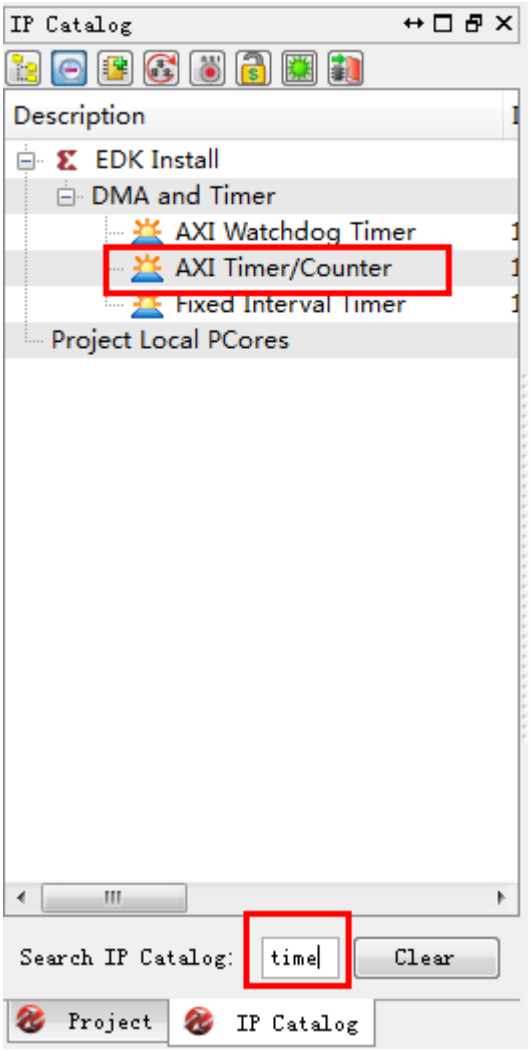


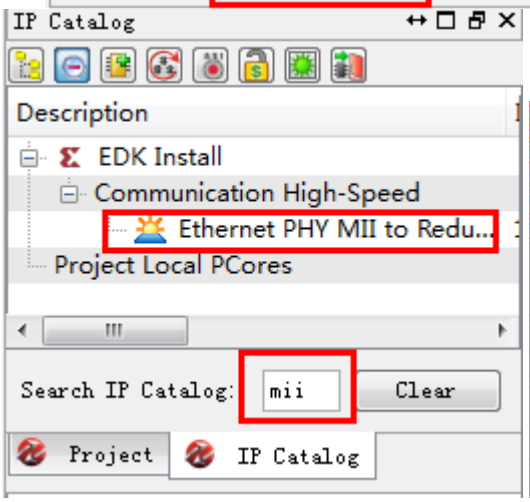
图 9： Clock_generator_0 中的 CLKIN

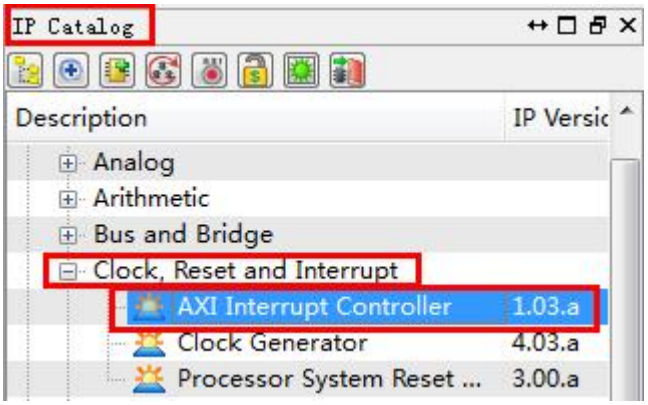
2-1-5. 按照下图，在 IP Catalog 面板中，选择 IP Catalog 标签，在 Search IP Catalog 中输入 time，然后在上端双击“AXI Timer”，以此来添加 axi_timer。然后用相同的方法搜索到“Ethernet phy mii to reduce mii”和“axi interrupt controller”

注意：添加 ip 过程中，弹出的对话框一律选择默认设置，直接一路点击确认到最后即可

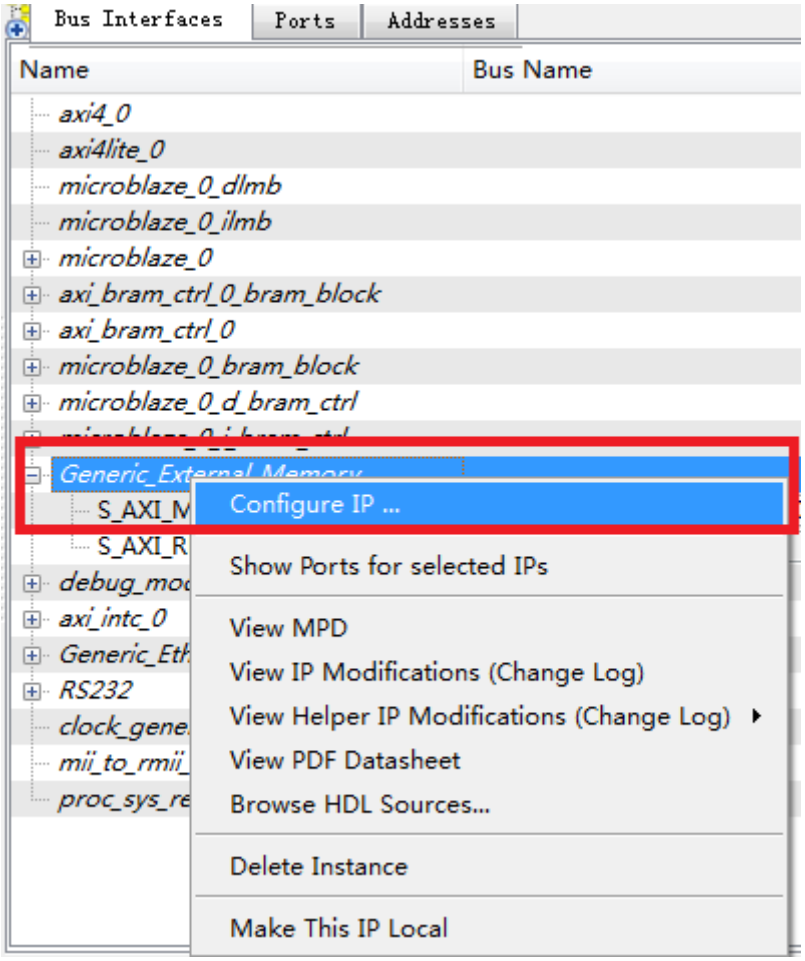


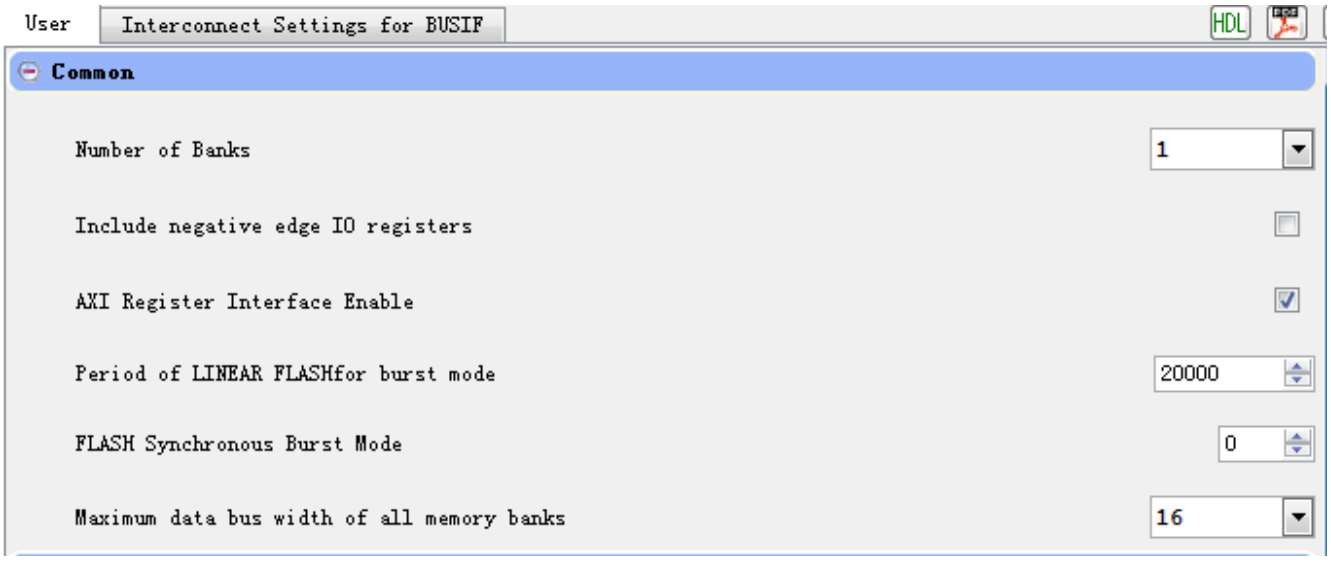


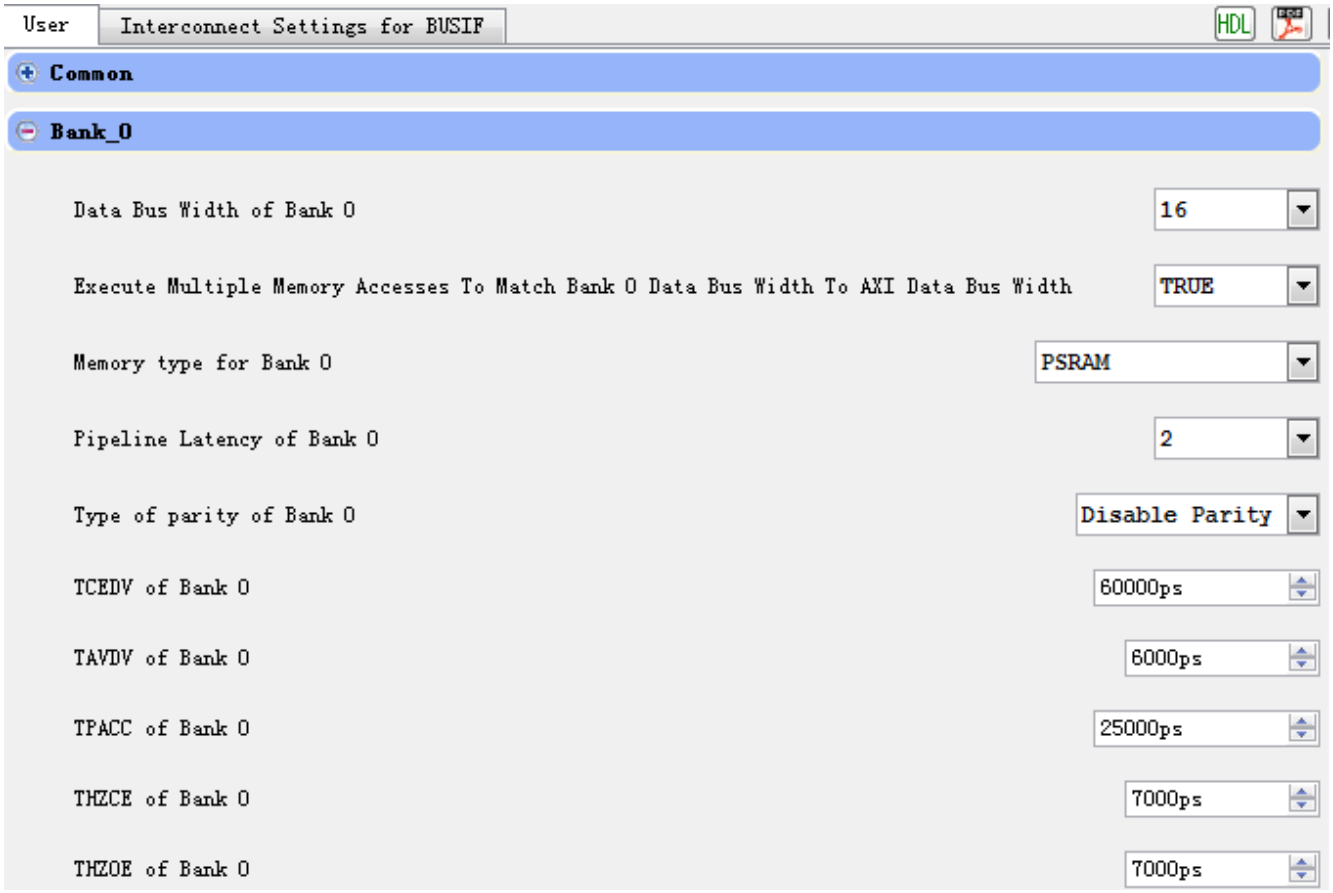


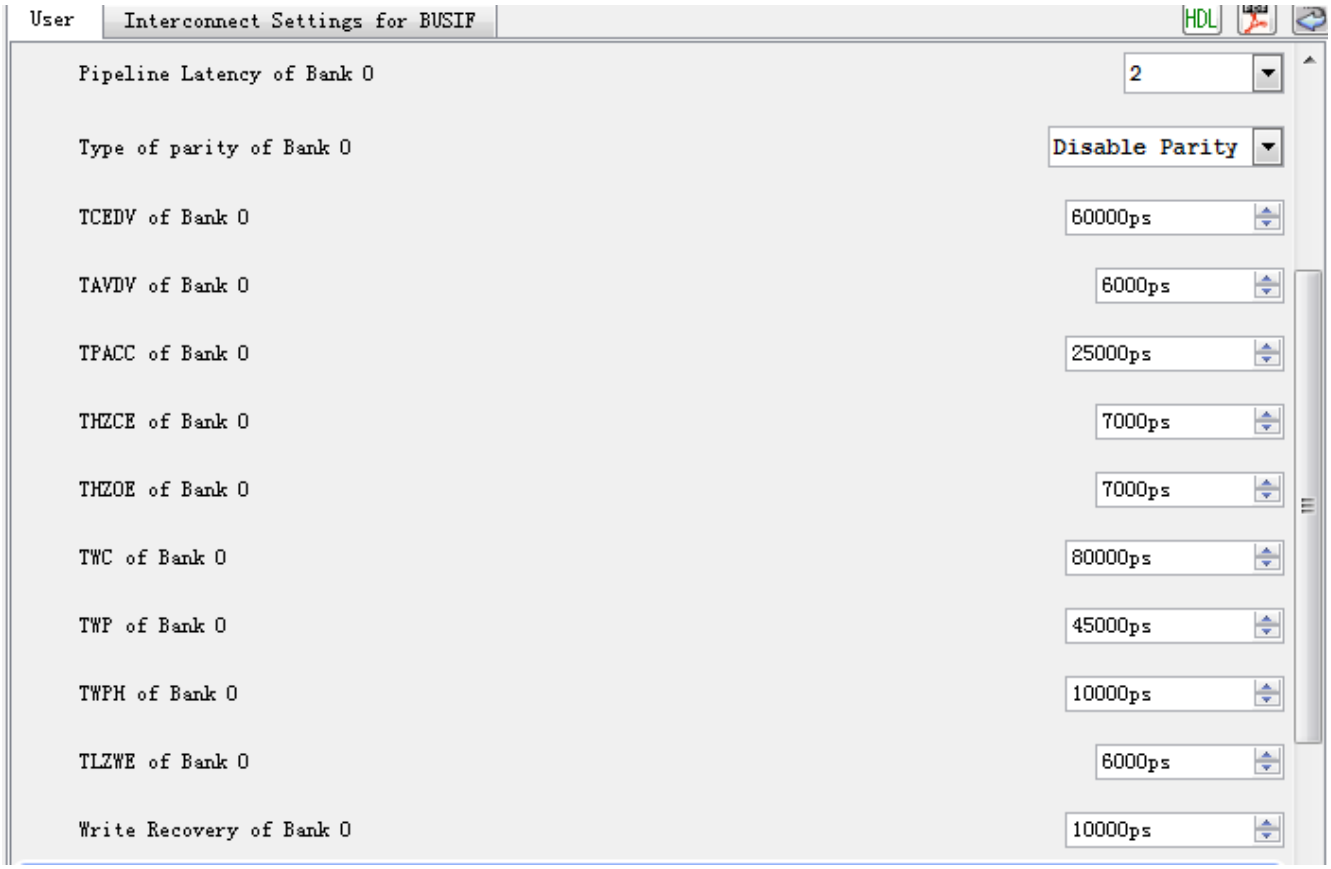


按照下图更改 external memory 的参数

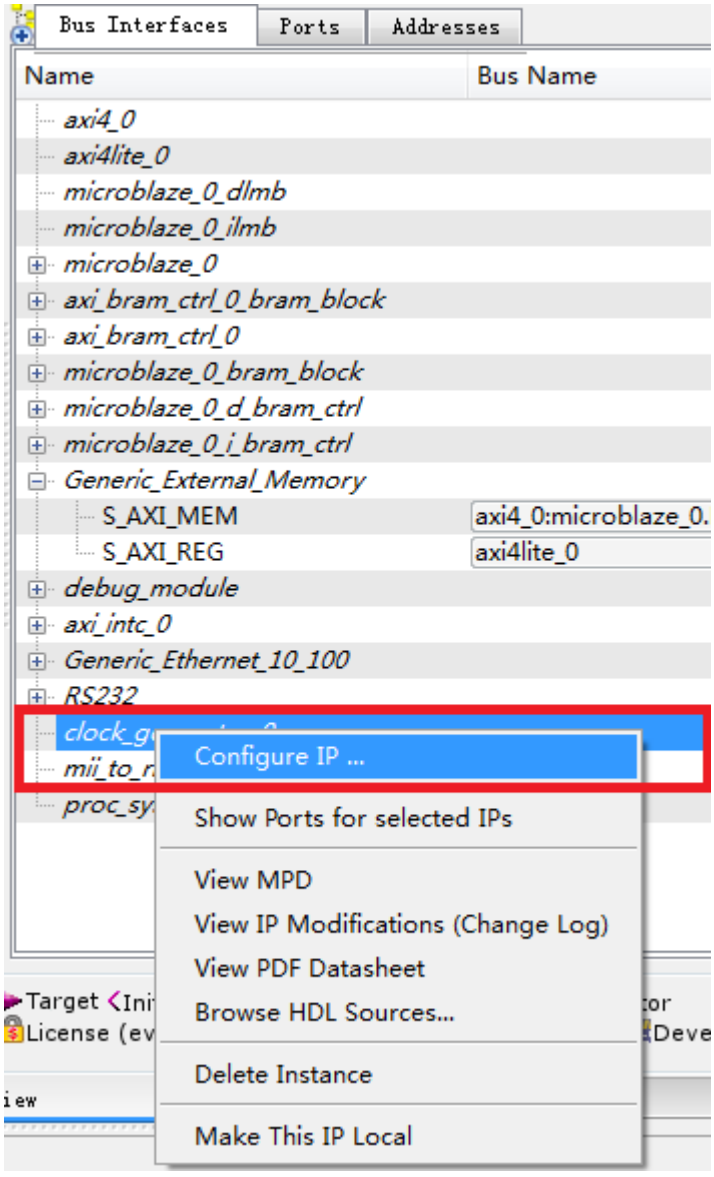


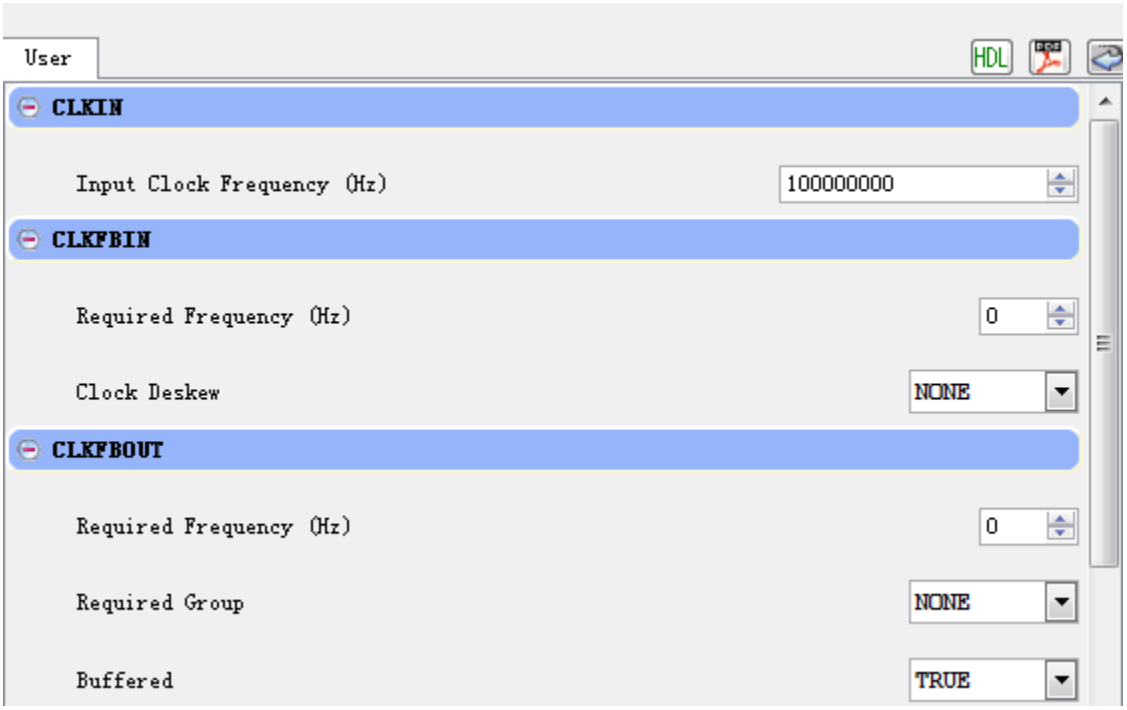


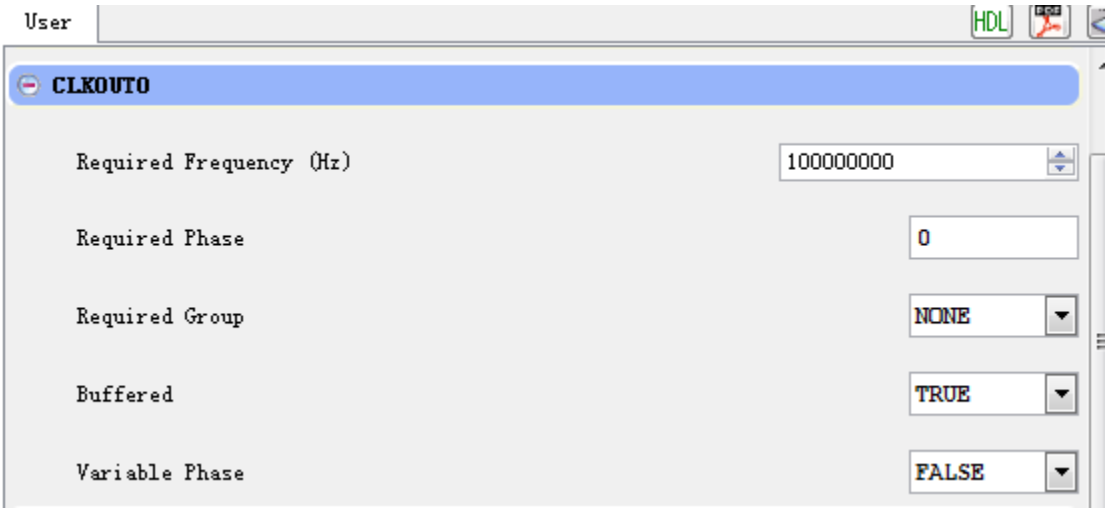

User
Interconnect Settings for BUSIF
HDL
Common
Bank_0
Data Bus Width of Bank 0
16
Execute Multiple Memory Accesses To Match Bank 0 Data Bus Width To AXI Data Bus Width
TRUE
Memory type for Bank 0
PSRAM
Pipeline Latency of Bank 0
2
Type of parity of Bank 0
Disable Parity
TCEDV of Bank 0
60000ps
TAVDV of Bank 0
6000ps
TPACC of Bank 0
25000ps
THZCE of Bank 0
7000ps
THZOE of Bank 0
7000ps


User
Interconnect Settings for BUSIF
HDL
Pipeline Latency of Bank 0
2
Type of parity of Bank 0
Disable Parity
TCEDV of Bank 0
60000ps
TAVDV of Bank 0
6000ps
TPACC of Bank 0
25000ps
THZCE of Bank 0
7000ps
THZOE of Bank 0
7000ps
TWC of Bank 0
80000ps
TWP of Bank 0
45000ps
TWPH of Bank 0
10000ps
TLZWE of Bank 0
6000ps
Write Recovery of Bank 0
10000ps

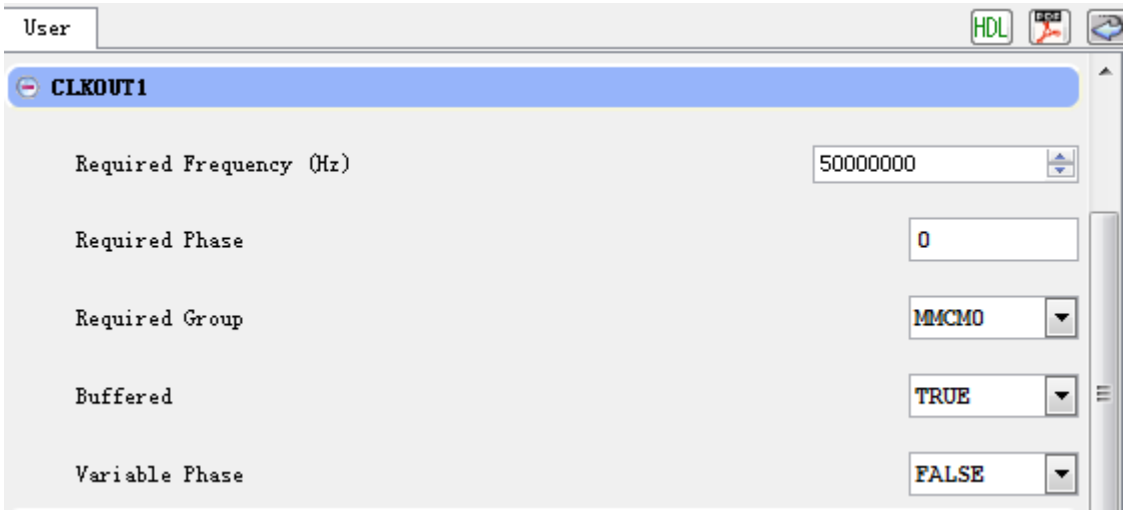
2-1-6. 按照下图更改 clock generator 的参数



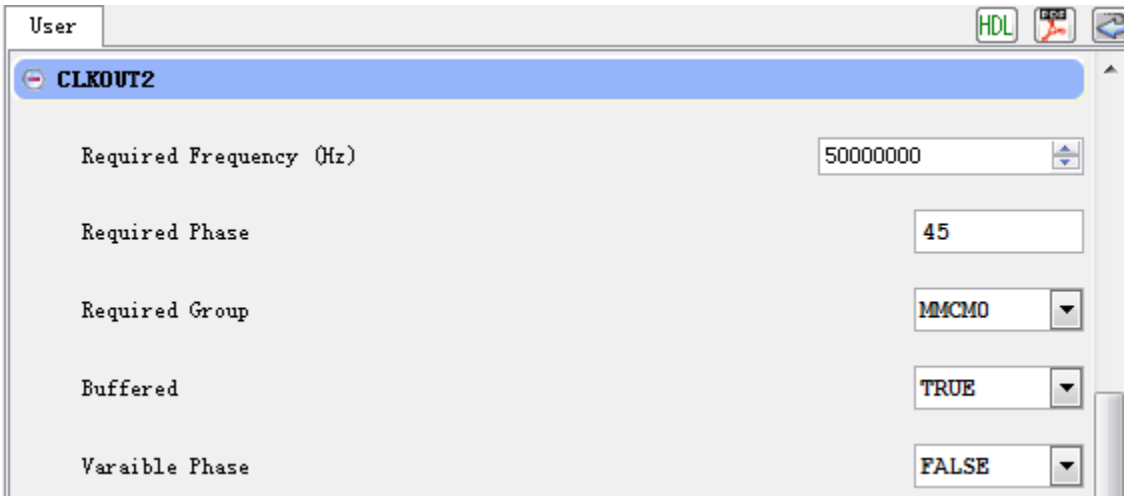

User
HDL
CLKIN
Input Clock Frequency (Hz)
100000000
CLKFBIN
Required Frequency (Hz)
0
Clock Deskew
NONE
CLKFBOUT
Required Frequency (Hz)
0
Required Group
NONE
Buffered
TRUE


User
HDL
CLKOUT0
Required Frequency (Hz)
100000000
Required Phase
0
Required Group
NONE
Buffered
TRUE
Variable Phase
FALSE

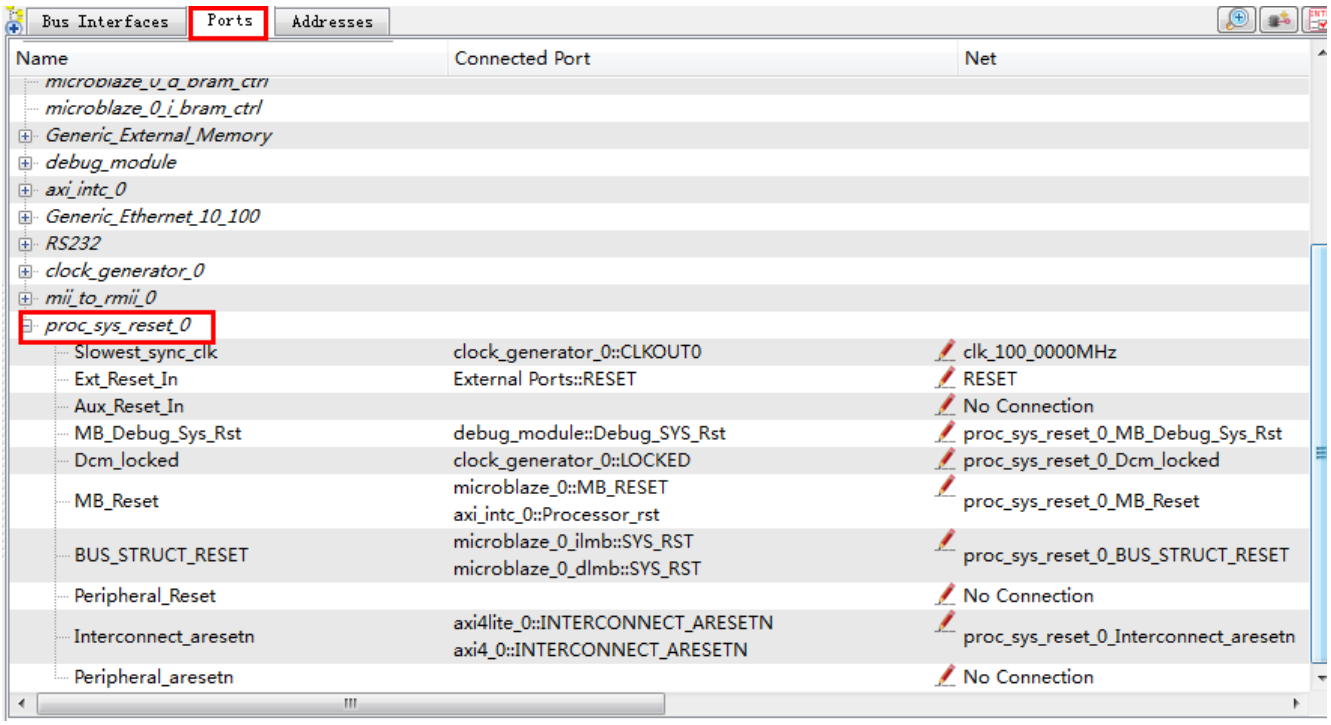
设置一个 50MHz 的时钟



设置一个差 45 度的同样是 50MHz 的时钟，注意 CLKOUT1 和 CLKOUT2 的 required group 一定要相同，否则可能出现时钟管理不同步的现象



2-1-7. 按照下图 proce_sys_reset_0 的连线规则进行调整，不一样的请按照下图更改

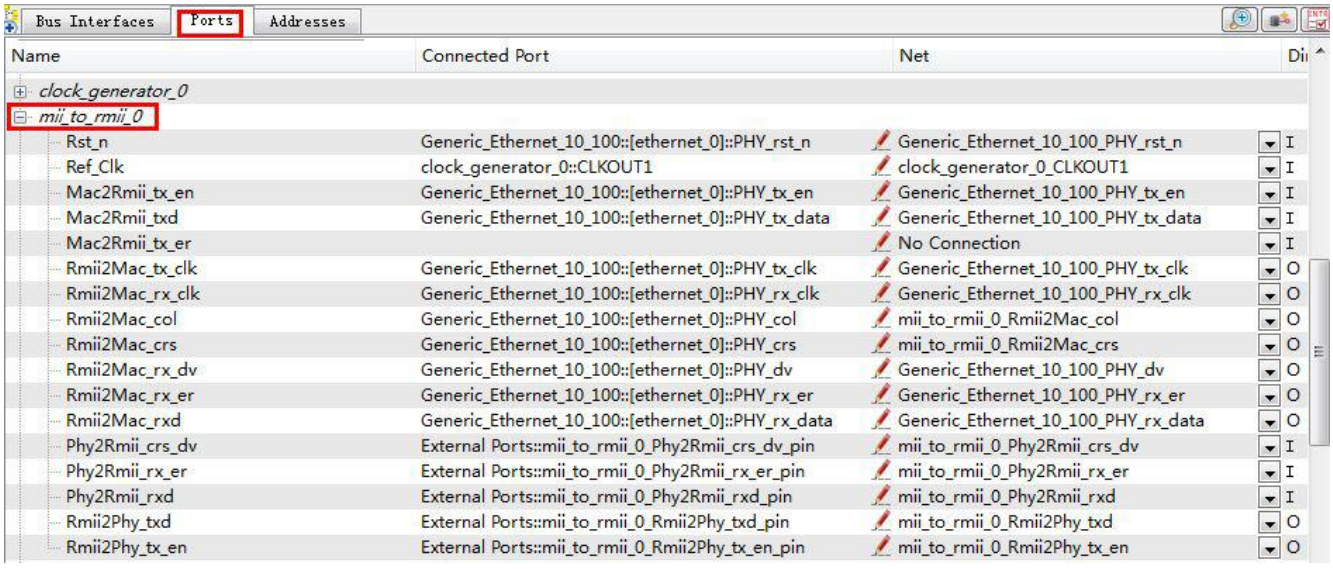


2-1-8. 按照下图更改 mii_to_rmii_0 的参数

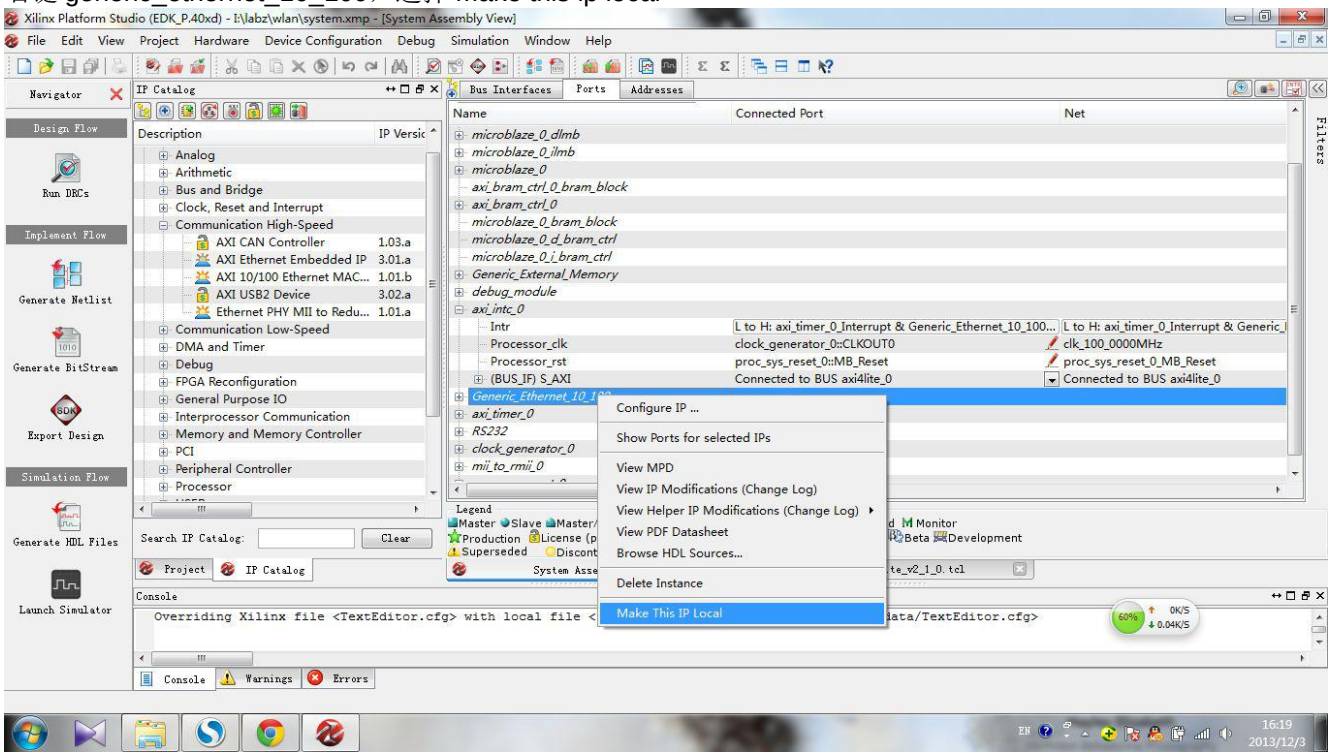
由于 ethernet 的 ip 核输出的是 mii 类型数据，而 phy 芯片的物理端口用 rmii 数据类型处理，所以要加入 mii_to_rmii_0 这个 ip 核，相当于在 ethernet ip 核和 phy 芯片之间起到沟通的作用。

将 Phy2Rmii_crs_dv、Phy2Rmii_rx_er、Phy2Rmii_rxd、Phy2Rmii_txd、Phy2Rmii_tx_en 均设置成 external port（右键端口，然后选择 make external）

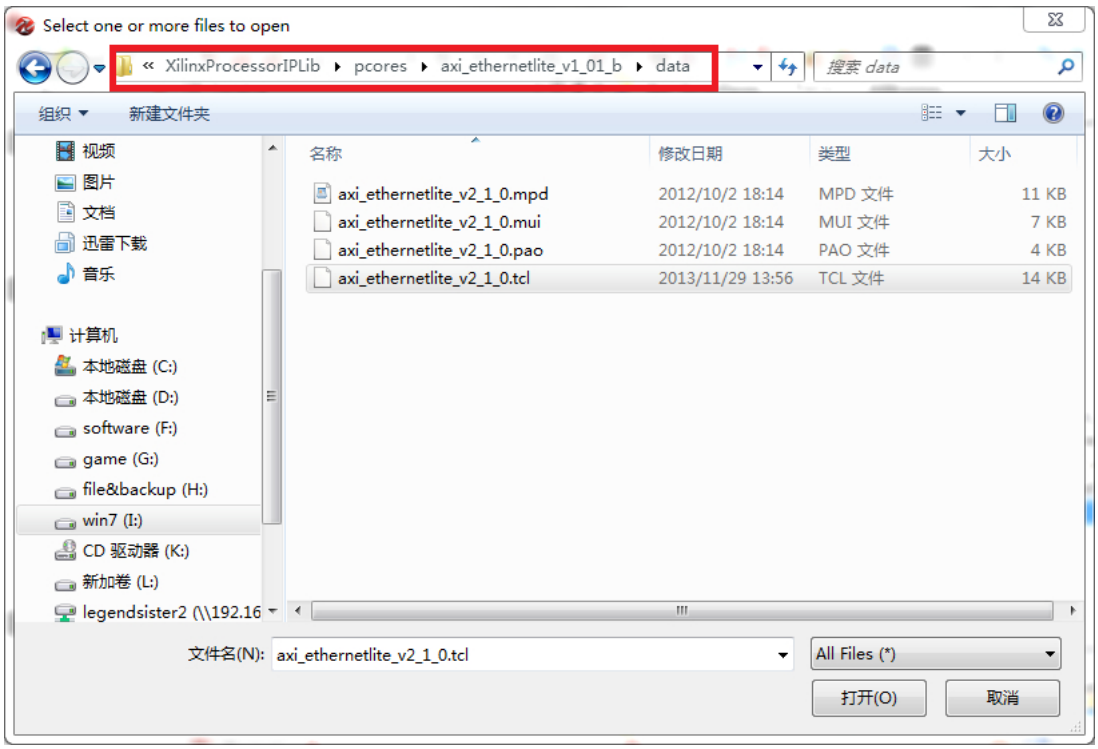
由于 mii_to_rmii 是一个 phy 和 ethernet ip 核之间起到数据转换作用的 ip 核，所以另外的接口链接到 ethernet ip 核的端口上，具体的连接规则请参考下图



2-1-9. 按照下图更改 generic_ethernet_10_100 的设置
右键 generic_ethernet_10_100，选择 make this ip local



然后右键选择 browse hdl source，在弹开的对话框中将目录定位到相对路径为…\pcores\axi_ethernetlite_v1_01_b\data 的目录中（可以利用下图中的红框所示的地方定位目录），在右下角选择“all files(*)”，现在应该出现如下图所示的内容，选择 axi_ethernetlite_v2_1_0.tcl 打开。

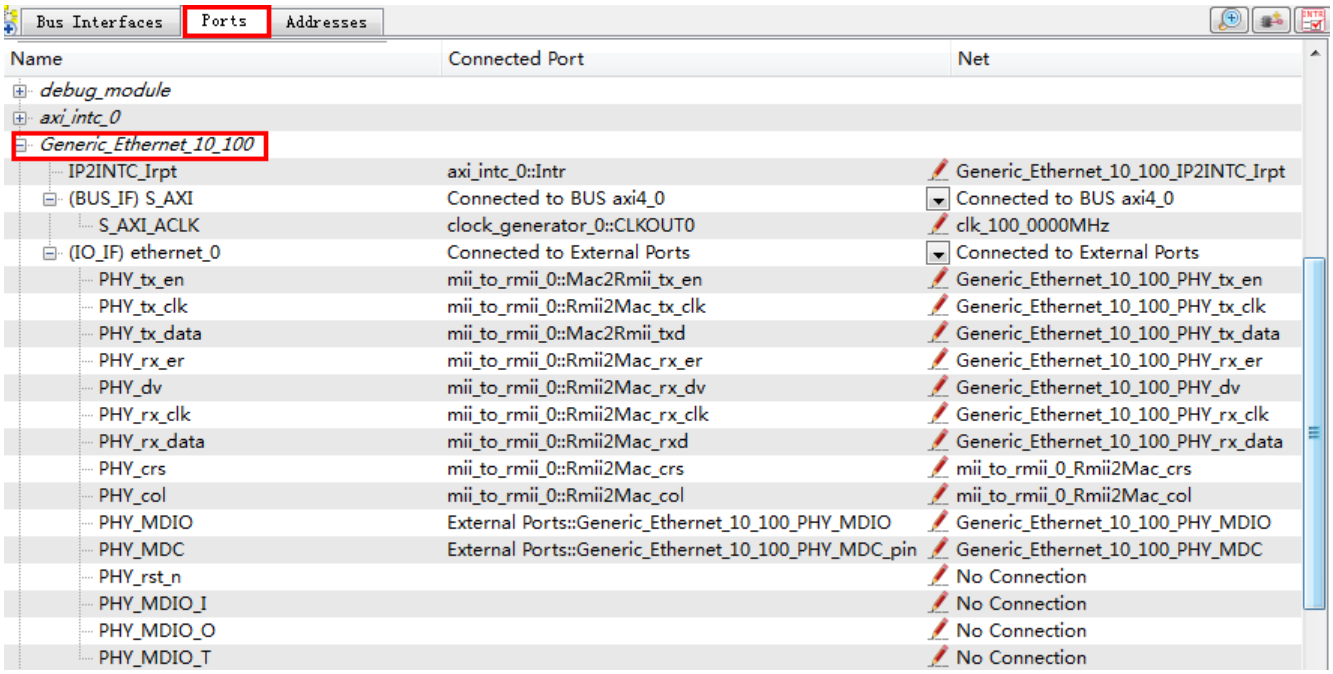


用“lab13 网口” 中的 src 目录的.tcl 文件的内容替换这些代码，然后保存。

接下来主要的工作就是将 ethernet 部分端口连出来（连到 mii_to_rmii 这个 ip 核上）

注意：将 phy_mdio、phy_mdc 两个端口设置成 external port，其余的 phy_rst_n、phy_mdio_i、phy_mdio_o、phy_mdio_t 这几个端口不做修改，保持默认状态

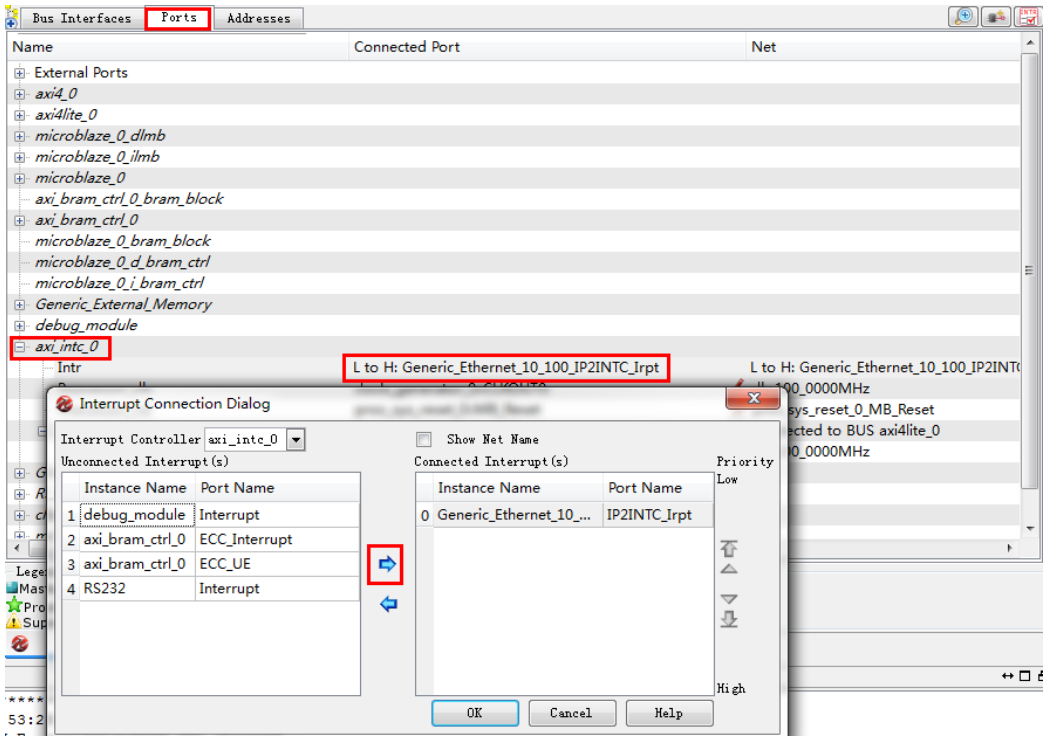
具体的连接方法请参考下图



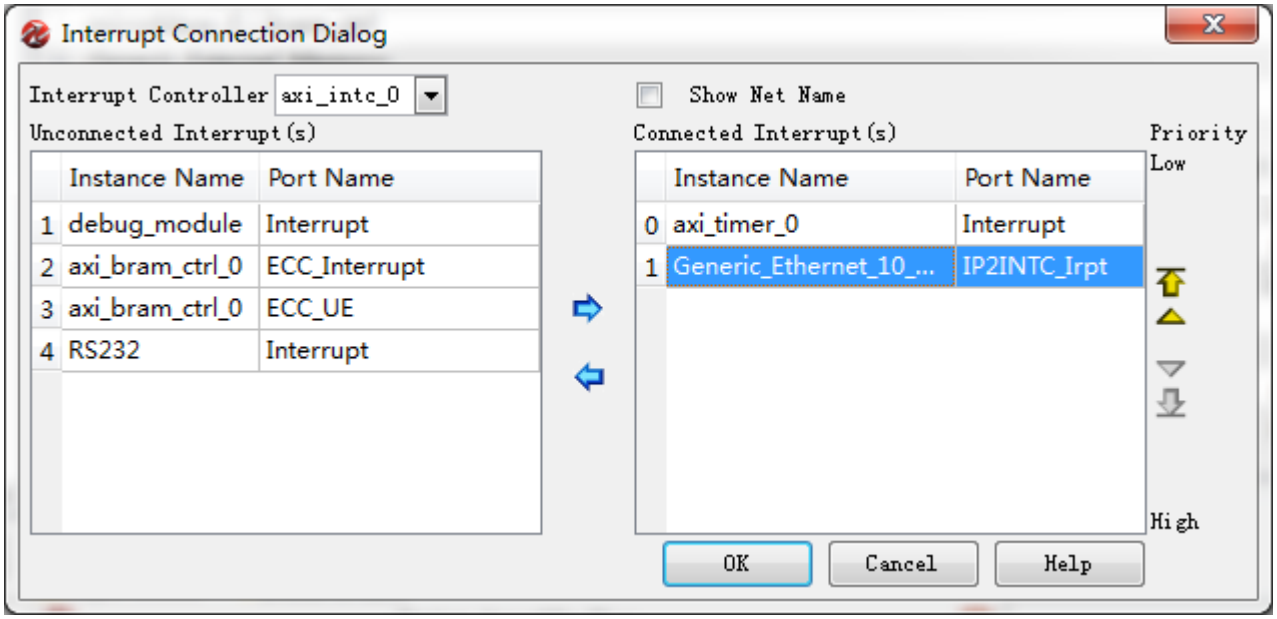
2-1-10. 按照下图更改 axi_intc_0 的中断设置

展开 axi_intc0_0，在 intr 一行中有 “L to H……”字样，点击这个“L to H……”（稍等 1 到 2 秒即可弹出对话框）

利用蓝色的向右的箭头，将 generic ethernet 和 axi_timer0 添加到右侧（注意而这上下的顺序）

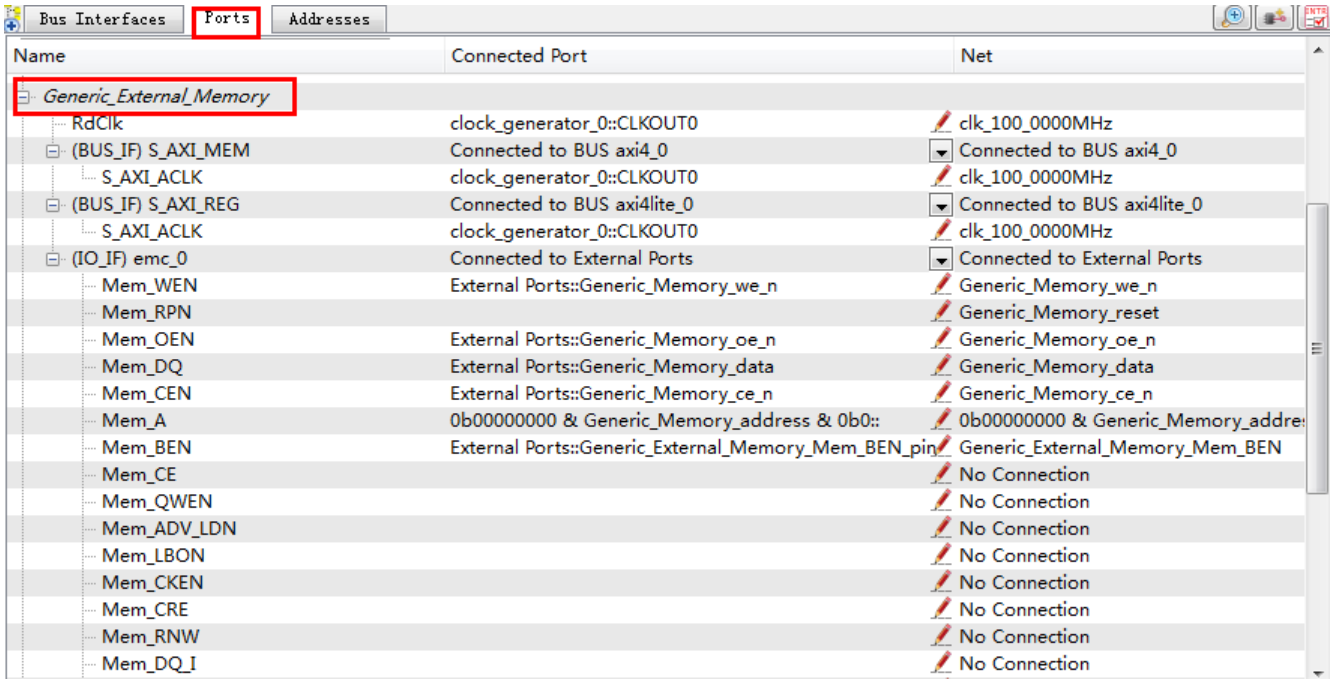


最终添加后应该如下图所示：

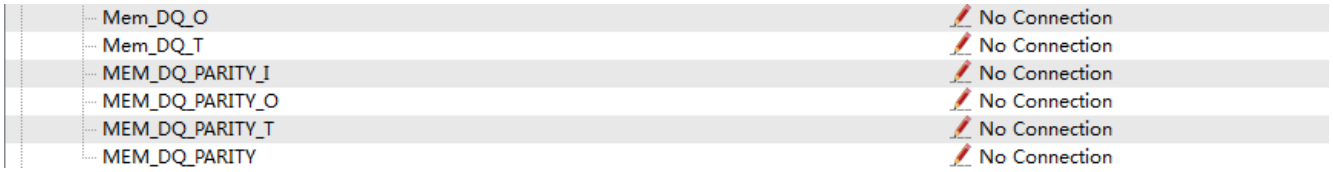


2-1-11. 按照下图更改 generic external memory 的连线方式

主要要更改的是 emc_0 的端口，将其中的端口 make external，具体连接方法如下图所示，图中未更改的端口表示不需要更改

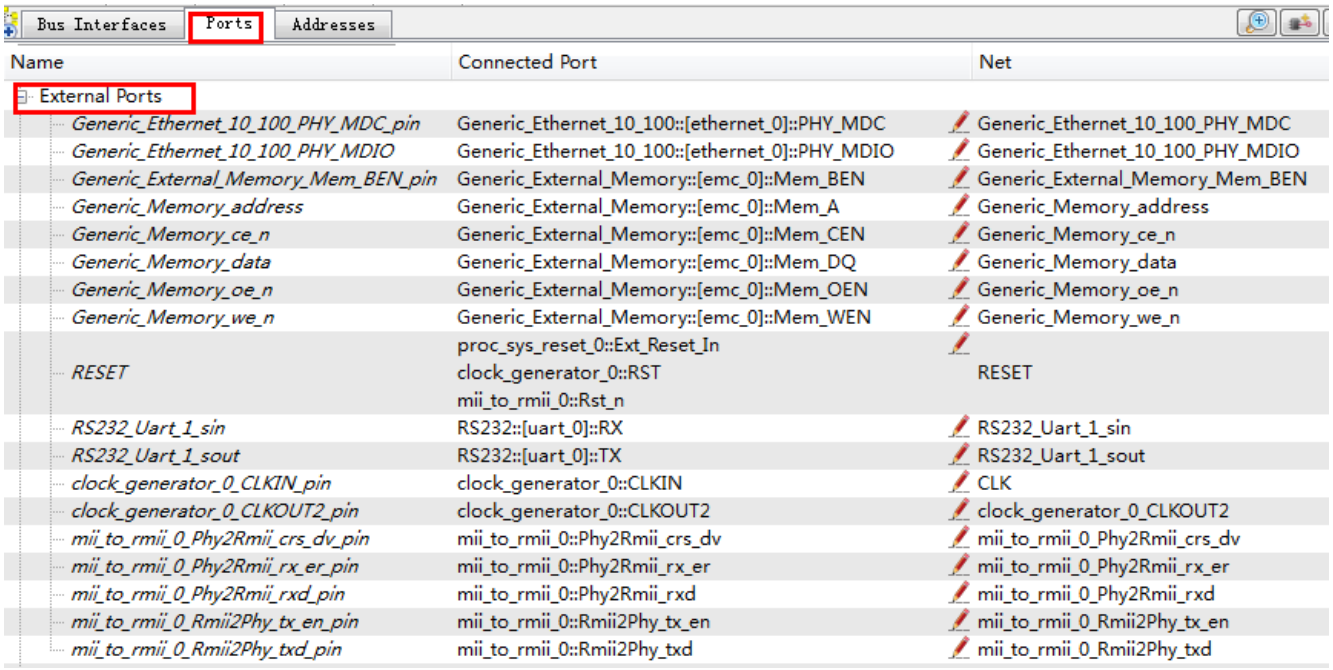


Name	Connected Port	Net
Generic_External_Memory		
RdClk	clock_generator_0::CLKOUT0	clk_100_0000MHz
(BUS_IF) S_AXI_MEM	Connected to BUS axi4_0	Connected to BUS axi4_0
S_AXI_ACLK	clock_generator_0::CLKOUT0	clk_100_0000MHz
(BUS_IF) S_AXI_REG	Connected to BUS axi4lite_0	Connected to BUS axi4lite_0
S_AXI_ACLK	clock_generator_0::CLKOUT0	clk_100_0000MHz
(IO_IF) emc_0	Connected to External Ports	Connected to External Ports
Mem_WEN	External Ports::Generic_Memory_we_n	Generic_Memory_we_n
Mem_RPN		Generic_Memory_reset
Mem_OEN	External Ports::Generic_Memory_oe_n	Generic_Memory_oe_n
Mem_DQ	External Ports::Generic_Memory_data	Generic_Memory_data
Mem_CEN	External Ports::Generic_Memory_ce_n	Generic_Memory_ce_n
Mem_A	0b00000000 & Generic_Memory_address & 0b0::	0b00000000 & Generic_Memory_addre
Mem_BEN	External Ports::Generic_External_Memory_Mem_BEN_pin	Generic_External_Memory_Mem_BEN
Mem_CE		No Connection
Mem_QWEN		No Connection
Mem_ADV_LDN		No Connection
Mem_LBON		No Connection
Mem_CKEN		No Connection
Mem_CRE		No Connection
Mem_RNW		No Connection
Mem_DQ_I		No Connection



Mem_DQ_O		No Connection
Mem_DQ_T		No Connection
MEM_DQ_PARITY_I		No Connection
MEM_DQ_PARITY_O		No Connection
MEM_DQ_PARITY_T		No Connection
MEM_DQ_PARITY		No Connection

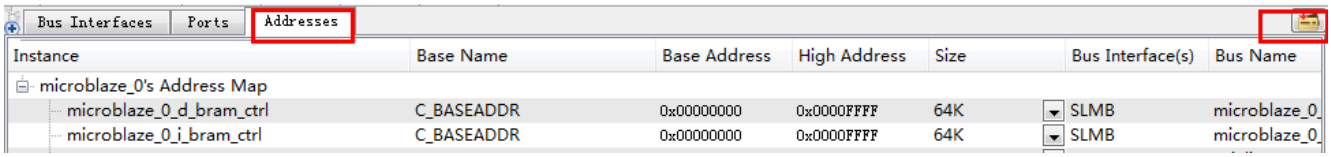
2-1-12. 按照上面的流程做，external port 展开后应该得到如下图所示的结果，请参照下图做适当检查：



Name	Connected Port	Net
External Ports		
Generic_Ethernet_10_100_PHY_MDC_pin	Generic_Ethernet_10_100::[ethernet_0]::PHY_MDC	Generic_Ethernet_10_100_PHY_MDC
Generic_Ethernet_10_100_PHY_MDIO	Generic_Ethernet_10_100::[ethernet_0]::PHY_MDIO	Generic_Ethernet_10_100_PHY_MDIO
Generic_External_Memory_Mem_BEN_pin	Generic_External_Memory::[emc_0]::Mem_BEN	Generic_External_Memory_Mem_BEN
Generic_Memory_address	Generic_External_Memory::[emc_0]::Mem_A	Generic_Memory_address
Generic_Memory_ce_n	Generic_External_Memory::[emc_0]::Mem_CEN	Generic_Memory_ce_n
Generic_Memory_data	Generic_External_Memory::[emc_0]::Mem_DQ	Generic_Memory_data
Generic_Memory_oe_n	Generic_External_Memory::[emc_0]::Mem_OEN	Generic_Memory_oe_n
Generic_Memory_we_n	Generic_External_Memory::[emc_0]::Mem_WEN	Generic_Memory_we_n
RESET	proc_sys_reset_0::Ext_Reset_In clock_generator_0::RST mii_to_rmii_0::Rst_n	RESET
RS232_Uart_1_sin	RS232::[uart_0]::RX	RS232_Uart_1_sin
RS232_Uart_1_sout	RS232::[uart_0]::TX	RS232_Uart_1_sout
clock_generator_0_CLKIN_pin	clock_generator_0::CLKIN	CLK
clock_generator_0_CLKOUT2_pin	clock_generator_0::CLKOUT2	clock_generator_0_CLKOUT2
mii_to_rmii_0_Phy2Rmii_crs_dv_pin	mii_to_rmii_0::Phy2Rmii_crs_dv	mii_to_rmii_0_Phy2Rmii_crs_dv
mii_to_rmii_0_Phy2Rmii_rx_er_pin	mii_to_rmii_0::Phy2Rmii_rx_er	mii_to_rmii_0_Phy2Rmii_rx_er
mii_to_rmii_0_Phy2Rmii_rxd_pin	mii_to_rmii_0::Phy2Rmii_rxd	mii_to_rmii_0_Phy2Rmii_rxd
mii_to_rmii_0_Rmii2Phy_tx_en_pin	mii_to_rmii_0::Rmii2Phy_tx_en	mii_to_rmii_0_Rmii2Phy_tx_en
mii_to_rmii_0_Rmii2Phy_txd_pin	mii_to_rmii_0::Rmii2Phy_txd	mii_to_rmii_0_Rmii2Phy_txd

2-1-13. 按照下图映射地址

选中 Address 标签，点击右上角红色方框所示的按钮完成地址自动映射过程



第三步 添加用户约束文件

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\lab1.ucf**，ucf 文件在右侧打开

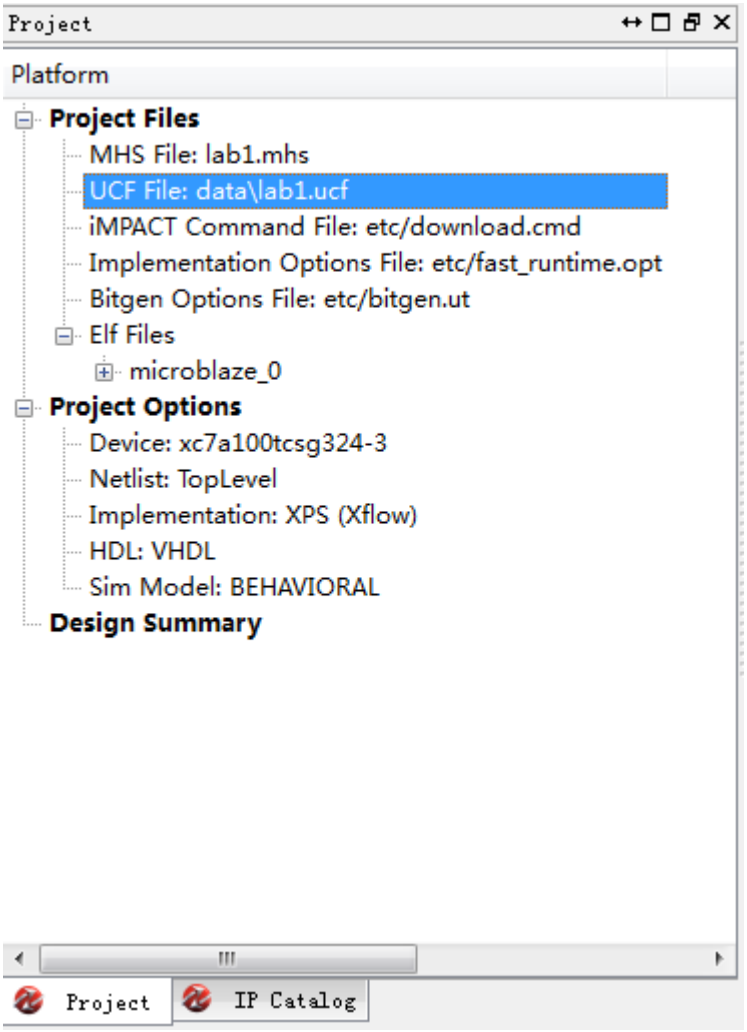
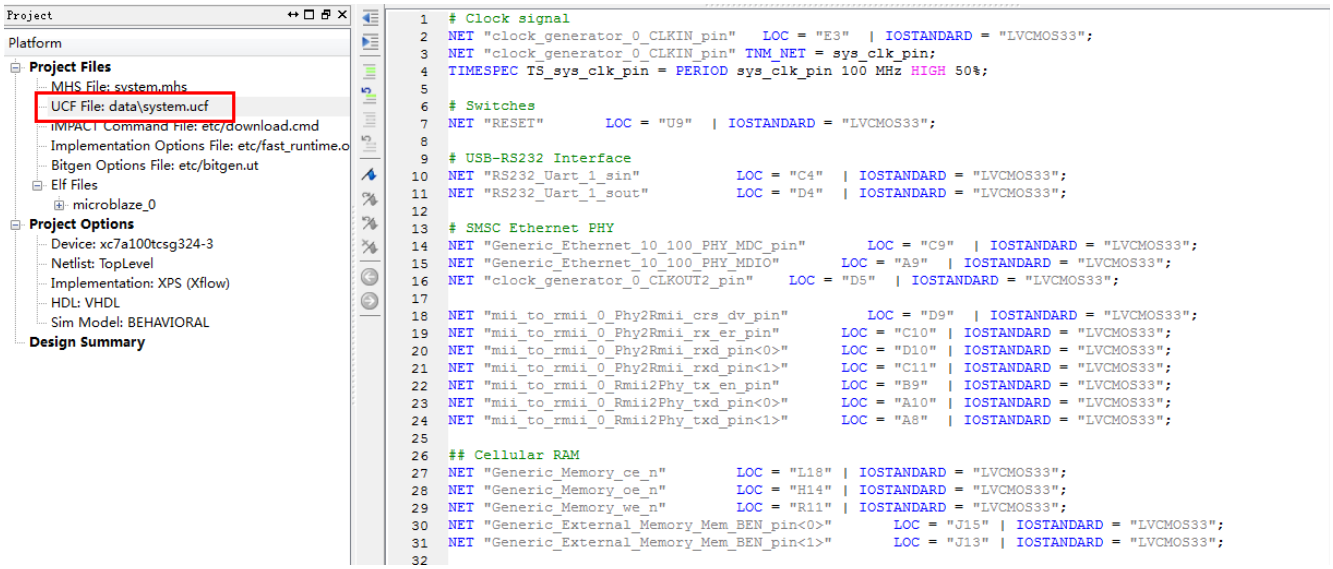


图 11：UCF 文件的位置

3-1-2. 这里我们手动输入 LOC（引脚位置）约束代码，如图 12。点击保存。

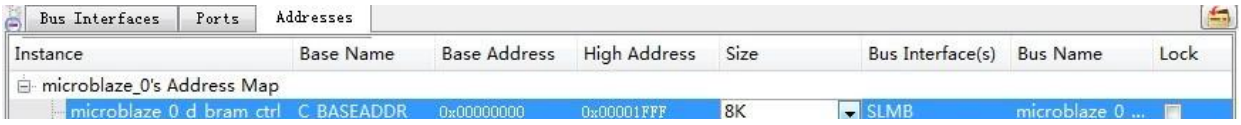



```
32
33
34  NET "Generic_Memory_data<0>"        LOC = "R12" | IOSTANDARD = "LVCMOS33";
35  NET "Generic_Memory_data<1>"        LOC = "T11" | IOSTANDARD = "LVCMOS33";
36  NET "Generic_Memory_data<2>"        LOC = "U12" | IOSTANDARD = "LVCMOS33";
37  NET "Generic_Memory_data<3>"        LOC = "R13" | IOSTANDARD = "LVCMOS33";
38  NET "Generic_Memory_data<4>"        LOC = "U18" | IOSTANDARD = "LVCMOS33";
39  NET "Generic_Memory_data<5>"        LOC = "R17" | IOSTANDARD = "LVCMOS33";
40  NET "Generic_Memory_data<6>"        LOC = "T18" | IOSTANDARD = "LVCMOS33";
41  NET "Generic_Memory_data<7>"        LOC = "R18" | IOSTANDARD = "LVCMOS33";
42  NET "Generic_Memory_data<8>"        LOC = "F18" | IOSTANDARD = "LVCMOS33";
43  NET "Generic_Memory_data<9>"        LOC = "G18" | IOSTANDARD = "LVCMOS33";
44  NET "Generic_Memory_data<10>"    LOC = "G17" | IOSTANDARD = "LVCMOS33";
45  NET "Generic_Memory_data<11>"    LOC = "M18" | IOSTANDARD = "LVCMOS33";
46  NET "Generic_Memory_data<12>"    LOC = "M17" | IOSTANDARD = "LVCMOS33";
47  NET "Generic_Memory_data<13>"    LOC = "P18" | IOSTANDARD = "LVCMOS33";
48  NET "Generic_Memory_data<14>"    LOC = "N17" | IOSTANDARD = "LVCMOS33";
49  NET "Generic_Memory_data<15>"    LOC = "P17" | IOSTANDARD = "LVCMOS33";
50
```

```
50
51  NET "Generic_Memory_address<0>"     LOC = "J18" | IOSTANDARD = "LVCMOS33";
52  NET "Generic_Memory_address<1>"     LOC = "H17" | IOSTANDARD = "LVCMOS33";
53  NET "Generic_Memory_address<2>"     LOC = "H15" | IOSTANDARD = "LVCMOS33";
54  NET "Generic_Memory_address<3>"     LOC = "J17" | IOSTANDARD = "LVCMOS33";
55  NET "Generic_Memory_address<4>"     LOC = "H16" | IOSTANDARD = "LVCMOS33";
56  NET "Generic_Memory_address<5>"     LOC = "K15" | IOSTANDARD = "LVCMOS33";
57  NET "Generic_Memory_address<6>"     LOC = "K13" | IOSTANDARD = "LVCMOS33";
58  NET "Generic_Memory_address<7>"     LOC = "N15" | IOSTANDARD = "LVCMOS33";
59  NET "Generic_Memory_address<8>"     LOC = "V16" | IOSTANDARD = "LVCMOS33";
60  NET "Generic_Memory_address<9>"     LOC = "U14" | IOSTANDARD = "LVCMOS33";
61  NET "Generic_Memory_address<10>"    LOC = "V14" | IOSTANDARD = "LVCMOS33";
62  NET "Generic_Memory_address<11>"    LOC = "V12" | IOSTANDARD = "LVCMOS33";
63  NET "Generic_Memory_address<12>"    LOC = "P14" | IOSTANDARD = "LVCMOS33";
64  NET "Generic_Memory_address<13>"    LOC = "U16" | IOSTANDARD = "LVCMOS33";
65  NET "Generic_Memory_address<14>"    LOC = "R15" | IOSTANDARD = "LVCMOS33";
66  NET "Generic_Memory_address<15>"    LOC = "N14" | IOSTANDARD = "LVCMOS33";
67  NET "Generic_Memory_address<16>"    LOC = "N16" | IOSTANDARD = "LVCMOS33";
68  NET "Generic_Memory_address<17>"    LOC = "M13" | IOSTANDARD = "LVCMOS33";
69  NET "Generic_Memory_address<18>"    LOC = "V17" | IOSTANDARD = "LVCMOS33";
70  NET "Generic_Memory_address<19>"    LOC = "U17" | IOSTANDARD = "LVCMOS33";
71  NET "Generic_Memory_address<20>"    LOC = "T10" | IOSTANDARD = "LVCMOS33";
72  NET "Generic_Memory_address<21>"    LOC = "M16" | IOSTANDARD = "LVCMOS33";
73  NET "Generic_Memory_address<22>"    LOC = "U13" | IOSTANDARD = "LVCMOS33";
```

图 12：UCF 文件

3-1-3. 接下来要扩大板卡的存储空间，选择”Address”选项卡，然后选择 **microblaze_0_bram_ctrl** 的 **Size** 属性，从默认的 **8K** 改到 **64K** 对于这个实验来讲就足矣了，如下图所示。



3-1-4. 保存之后将工程导入到 **SDK**，在页面左边，点击 **Export Design**，如图 **13**。

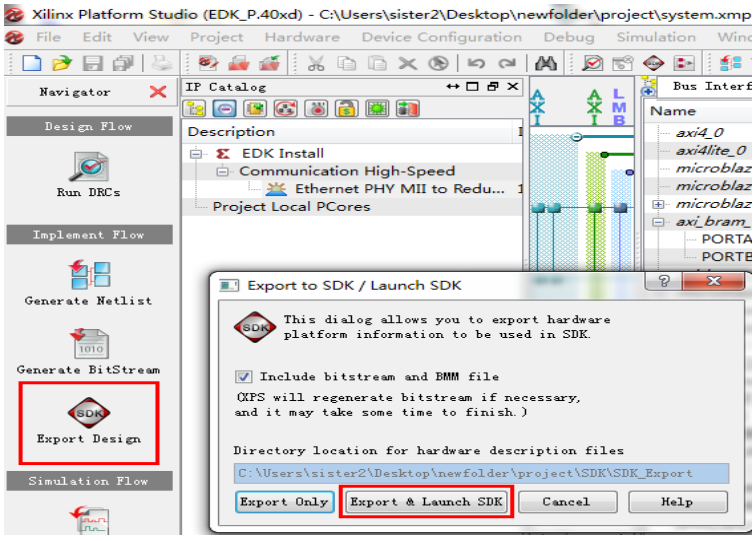


图 14：在弹出的对话框中选择 Export & launch sdk

3-1-4．选择 SDK 导入路径，注意要具体到..\sdk\sdk_export，如图 14，点击 ok

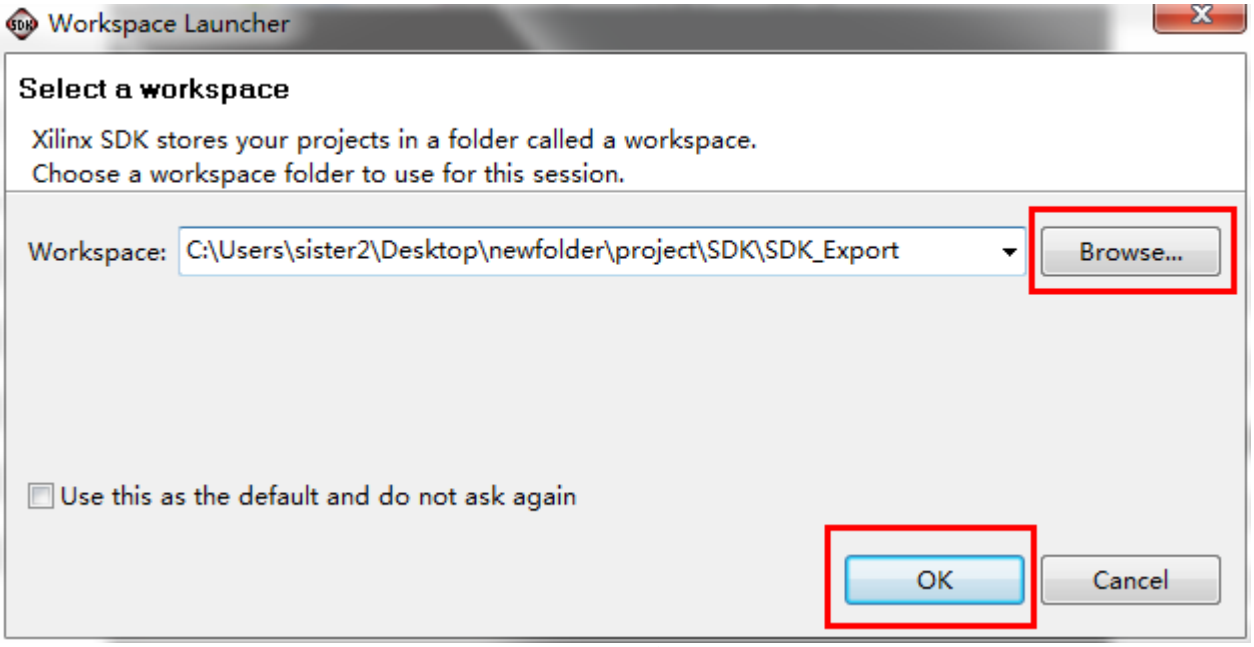


图 14：导入 sdk 的工作空间选择

第四步 添加 app

4-1. 添加软件应用。

4-1-1. 在 SDK 的用户界面中，**选择 file—new—application project，如图 15。**

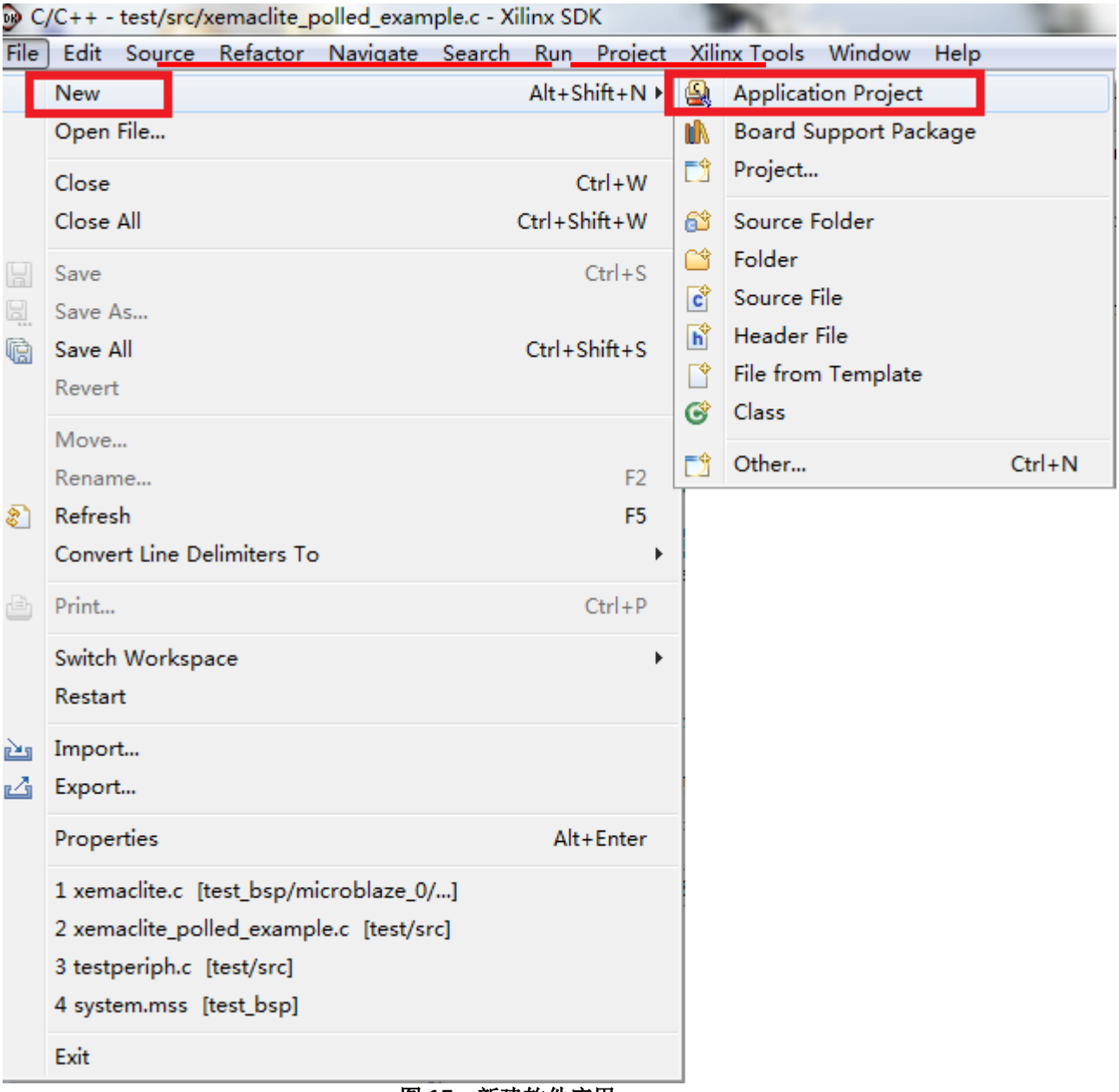


图 15：新建软件应用

4-1-2. 输入工程的名称，这里使用 **ethernet_test,** 同样不要包含空格和中文，点击 **next**

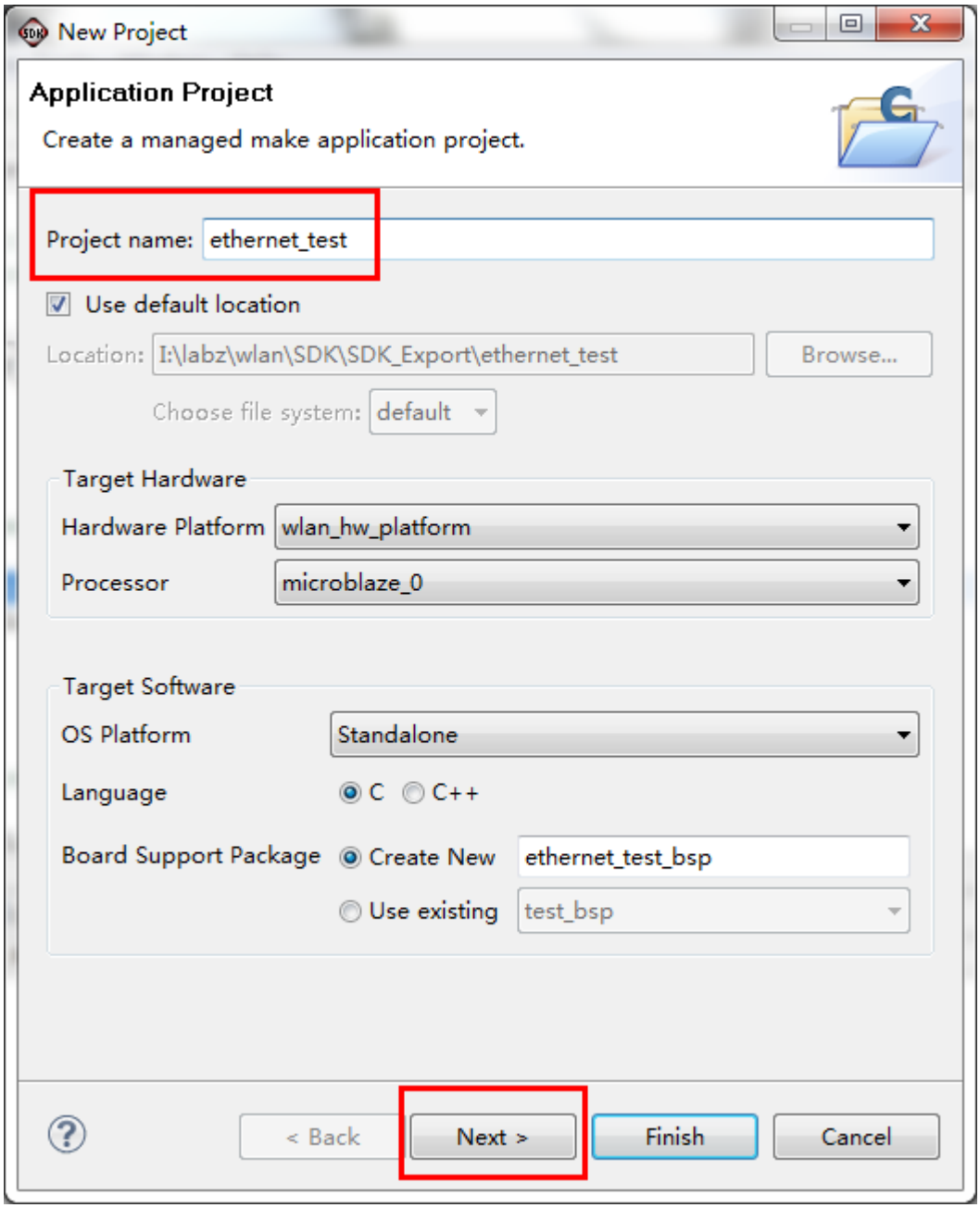


图 16：新建工程---命名

4-1-3. 在下一步弹出的对话框中选择 **peripheral tests**，然后点击 **finish**

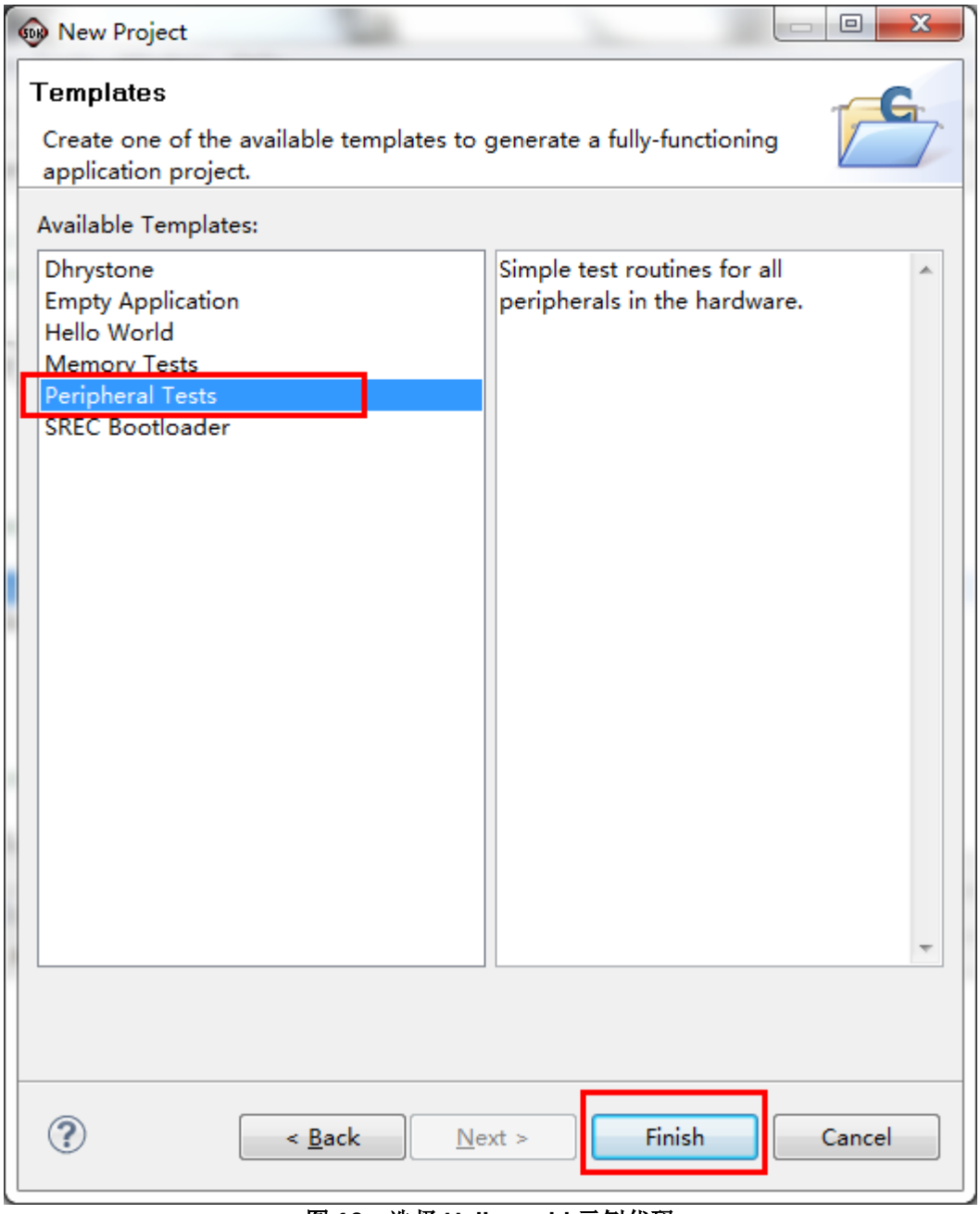


图 16：选择 Helloworld 示例代码

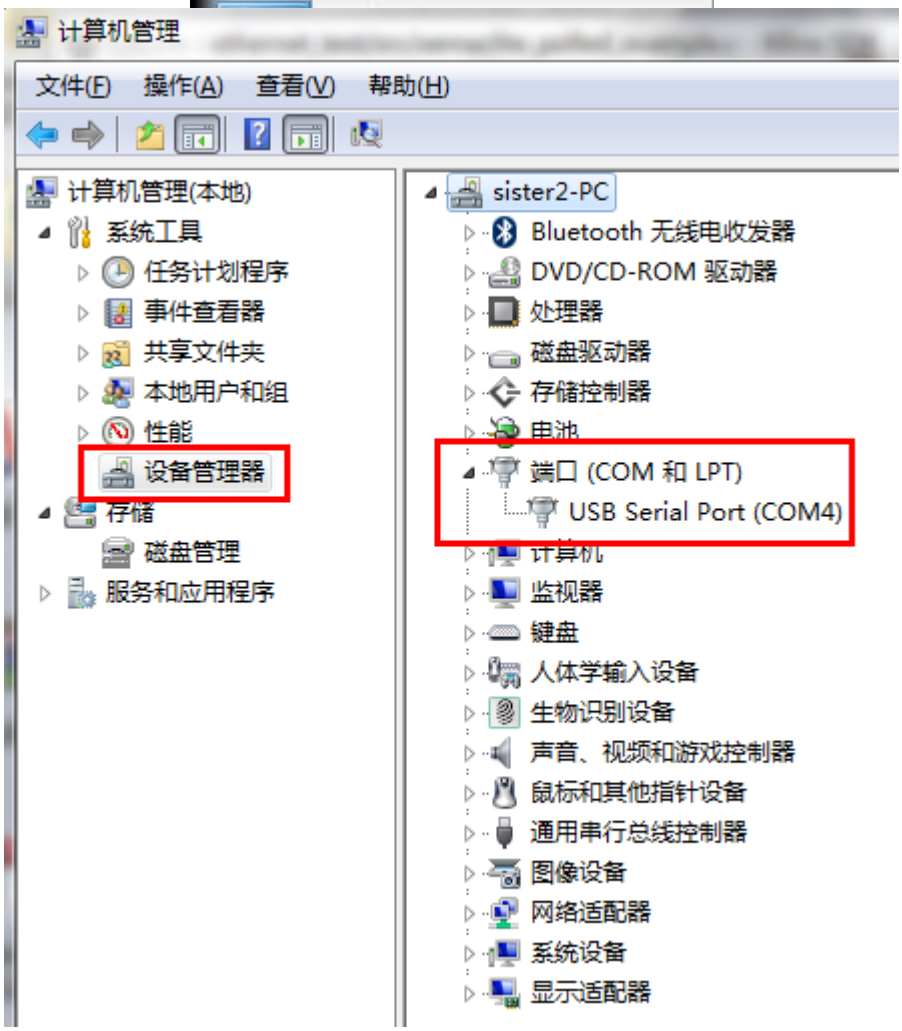
4-1-4. 添加完毕以后可以在左侧双击源文件，查看这段代码：

第五步 上板验证

5-1. 将 Nexys4 与 Pc 的 USB 接口连接

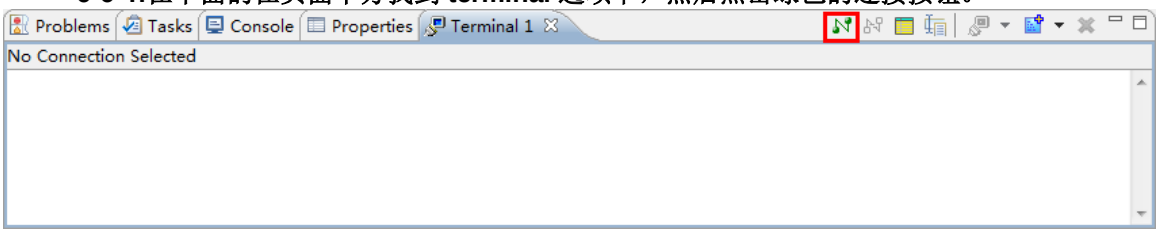
5-2. 查看端口号：

右键“我的电脑”—“管理”—“设备管理器”—“端口”，在此查看串口号（此处为 COM4）

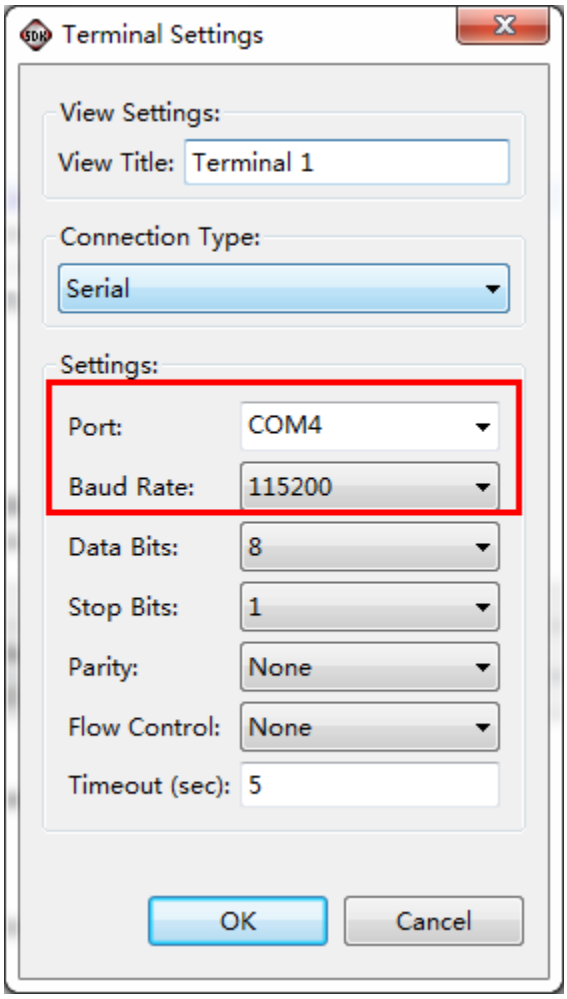


5-3. 在 SDK 中打开串口：

5-3-1.在下面的在页面下方找到 **terminal** 选项卡，然后点击绿色的连接按钮。

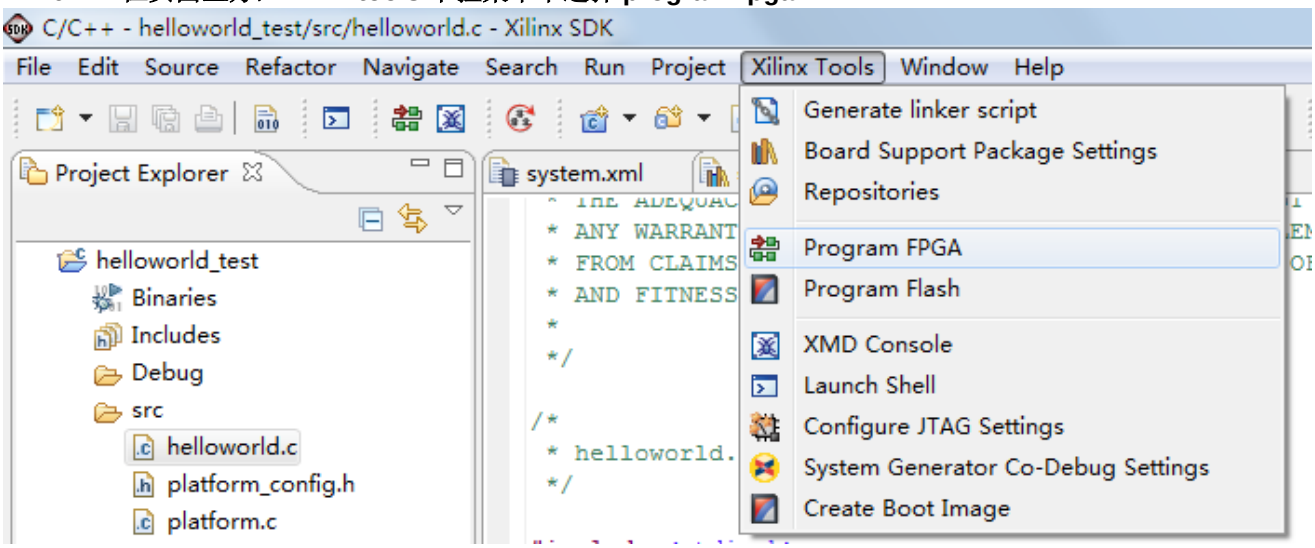


5-3-2.按照端口号和 **XPS** 中的波特率（**baud rate**）进行如下设置：

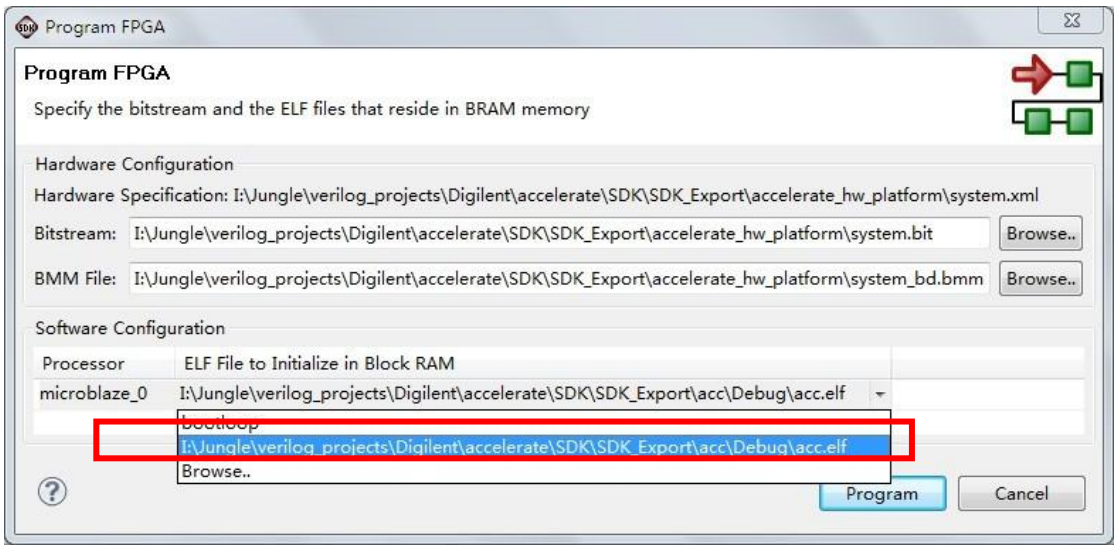


5-4. 将程序下载到板子上并运行

5-4-1.在页面上方，xilinx tools 下拉菜单中选择 program fpga



5-4-2.注意要选择正确的 elf 文件：



点击 program

5-5. 在串口中看到结果：


```
---Entering main---

 Running IntcSelfTestExample() for axi_intc_0...
IntcSelfTestExample PASSED
Intc Interrupt Setup PASSED

 Running TmrCtrSelfTestExample() for axi_timer_0...
TmrCtrSelfTestExample PASSED

 Running Interrupt Test  for axi_timer_0...
Timer Interrupt Test PASSED

Running UartLiteSelfTestExample() for debug_module...
UartLiteSelfTestExample PASSED

Running EmacLitePolledExample() for generic_ethernet_10_100...
EmacLite Polled Example PASSED
---Exiting main---
```