

LwIP 的官方模板使用

章节导读

LwIP 是用于嵌入式系统的开源 TCP/IP 网络堆栈，它的内存使用率跟代码使用量都非常的小，只需要十几 KB 的 RAM 和 40KB 左右的 ROM 就可以运行，虽然可以与操作系统一起使用，但并不依赖于操作系统。因此 LwIP 非常适合应用于一些较小的嵌入式系统。

为了方便大家使用，Xilinx 官方在 SDK 中为我们提供了 LwIP 的模板，但是许多用户在使用 ACZ702 开发板运行 LwIP 模板例程时，会出现自动协商失败或者 Phy 设置失败的情况。这主要是由于硬件上的不同导致的，本节将教大家如何使用 LwIP 模板。

1 LwIP 概述

LwIP 是瑞典计算机科学院(SICS)的 Adam Dunkels 开发的一个小型开源的 TCP/IP 协议栈。实现的重点是在保持 TCP 协议主要功能的基础上减少对 RAM 的占用。

在 Vivado2018.3 中为我们提供好的，是 LwIPv2.0.2 版本的 SDK 库，这个库为 Ethernetlite (axi_Ethernetlite)、TEMAC (Axii_ethernet) 以及千兆以太网控制器和 MAC (GigE) 核提供适配。该库能够在 MicroBlaze、ARM Cortex-A9、ARM Cortex-A53、ARM Cortex-R5 处理器上运行。

Ethernetlite 和 TEMAC 核适用于 MicroBlaze 系统。千兆以太网控制器和 MAC (GigE) 核仅适用于 ARM Cortex-A9 (Zynq-7000 处理器设备)、ARM Cortex-A53 和 ARM Cortex-R5 (Zynq UltraScale + MPSoC) 系统。

根据是否基于操作系统，LwIP 提供了两套 API (术语为 A05PI)，分别如下：

- Raw API: 这是一个事件驱动的 API，设计为在没有实现操作系统的情况下使用。这个 API 也被核心栈用于各种协议之间的交互。它是在没有操作系统的情况下运行 lwIP 时惟一可用的 API。
- Socket API: bsd 风格的套接字 API。线程安全，只能从非 tcpip 线程调用。

对于嵌入式而言，通常并不会使用到操作系统，因此大都是使用的 Raw

API。LwIPv2.0.2 具有以下特征：

1. 支持多网络接口下的 IP 转发；
2. 支持 ICMP 协议；
3. 支持 DHCP 协议,动态分配 ip 地址.
4. 支持 ARP 协议（以太网地址解析协议）。
5. 支持 IGMP 协议（互联网组管理协议），可以实现多播数据的接收。
6. 支持 UDP 协议(用户数据报协议)。
7. 支持 TCP 协议(传输控制协议)，包括阻塞控制、RTT 估算、快速恢复和快速转发。

这里的 DHCP 是动态主机配置协议（Dynamic Host Configuration Protocol）的缩写，用于使网络环境中的主机动态的获得 IP 地址、Gateway 地址、DNS 服务器地址等信息。

2 硬件差异

SDK 中官方模板默认使用的是 Realtek 的 RTL8211E 芯片，而我们开发板使用的网卡芯片则是 Realtek 的 RTL8211FDI 芯片。这两个芯片存在着一定的差异，而导致用户使用官方模板时，自动协商失败的原因，就与 PHYSR 寄存器有关，两个芯片的 PHYSR 寄存器位描述具体如下：

Table 39. PHYSR (PHY Specific Status Register, Address 0x1A)

Bit	Name	Type	Default	Description
26.15	RSVD	RO	0	Reserved.
26.14	ALDPS State	RO	0	Link Down Power Saving Mode. 1: Reflects local device entered Link Down Power Saving Mode, i.e., cable not plugged in (reflected after 3 sec). 0: With cable plugged in
26.13	MDI Plug	RO	0	MDI Status. 1: Plugged 0: Unplugged
26.12	NWay Enable	RO	1	Auto-Negotiation (NWay) Status. 1: Enable 0: Disable
26.11	Master Mode	RO	0	Device is in Master/Slave Mode. 1: Master mode 0: Slave mode
26.10:9	RSVD	RO	00	Reserved.
26.8	EEE capability	RO	0	1: Both local and link-partner have EEE capability of current speed
26.7	Rxflow Enable	RO	0	Rx Flow Control. 1: Enable 0: Disable
26.6	Txflow Enable	RO	0	Tx Flow Control. 1: Enable 0: Disable
26.5:4	Speed	RO	00	Link Speed. 11: Reserved 10: 1000Mbps 01: 100Mbps 00: 10Mbps
26.3	Duplex	RO	0	Full/Half Duplex Mode. 1: Full duplex 0: Half duplex
26.2	Link (Real Time)	RO	0	Real Time Link Status. 1: Link OK 0: Link not OK
26.1	MDI Crossover Status	RO	1	MDI/MDI Crossover Status. 1: MDI 0: MDI Crossover
26.0	Jabber (Real Time)	RO	0	Real Time Jabber Indication. 1: Jabber Indication 0: No Jabber Indication

图 1 RTL8211FDI 的 PHYSR 寄存器说明

Table 36. PHYSR (PHY Specific Status Register, Address 0x11)

Bit	Name	RW	Default	Description
17.15:14	Speed	RO	01	Link Speed. 11: Reserved 10: 1000Mbps 01: 100Mbps 00: 10Mbps
17.13	Duplex	RO	0	Full/Half Duplex Mode. 1: Full duplex 0: Half duplex
17.12	Page Received	RC	0	New Page Received. 1: Page received 0: Page not received
17.11	Speed and Duplex Resolved	RO	0	Speed and Duplex Mode Resolved. 1: Resolved 0: Not resolved
17.10	Link (Real Time)	RO	0	Real Time Link Status. 1: Link OK 0: Link not OK
17.9:7	RSVD	RO	000	Reserved.
17.6	MDI Crossover Status	RO	0	MDI/MDI Crossover Status. 1: MDI Crossover 0: MDI
17.2:5	RSVD	RW	0000	Reserved.
17.1	pre_linkok	RO	0	Reflects Local Receiver is OK. 0: Receiver is not OK 1: Receiver is OK
17.0	Jabber (Real Time)	RO	0	Real Time Jabber Indication. 1: Jabber Indication 0: No jabber Indication

图 2 RTL8211E 的 PHYSR 寄存器说明

对比这两张表就可以看到，这两个寄存器无论是偏移地址还是位分布都有很大的差异。而也正是这些差异，导致 PHY 芯片在查询 PHYSR 实时链路是否正常时，读取回错误值，导致自动协商一直处于失败状态。除此之外，由于 Link Speed 位分布的不同，也一直无法获取到正确的速度配置。因此，在使用官方模板时，我们就需要修改这些相关位。

下面就以官方模板中的 LwIP Echo Server 为例，来带领大家一起学习如何使用 LwIP 模板。

3 硬件逻辑系统设计

首先创建一个 Vivado 工程以方便开始我们的设计，在选择工程路径设置工程名时注意要用纯英文。

3.1 IP 核添加与配置

要想使用 LwIP，硬件上我们需要使用到以太网口，ACZ702 开发板的 PS 侧与 PL 都有对应的网口外设，因此我们可以直接添加 ZYNQ 核，使能 PS 侧的以太网外设。PS 侧以太网口的硬件引脚电路如图 3 所示：

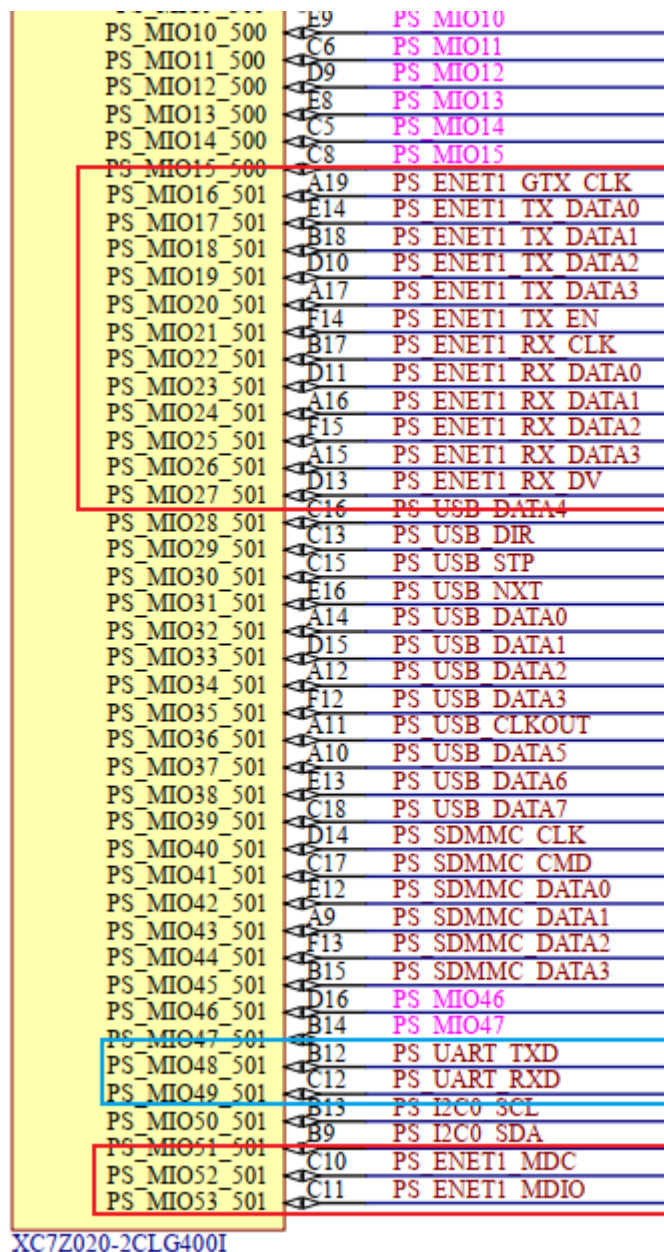


图 3 PS 侧以太网硬件引脚电路图

可以看到 PS 侧以太网对应引脚为 MIO16...27，打开添加的 ZYNQ IP 核，在 MIO Configuration 界面的 I/O Peripherals 栏中勾选 ENET 0，设置 IO 路由为 MIO16...27。同时展开 ENET 0，勾选 MDIO，设置 IO 路由为 MIO52...53，完成后将 Bank 1 的 I/O 电平标准设置为 LVCMOS 1.8V，如图 4 所示：

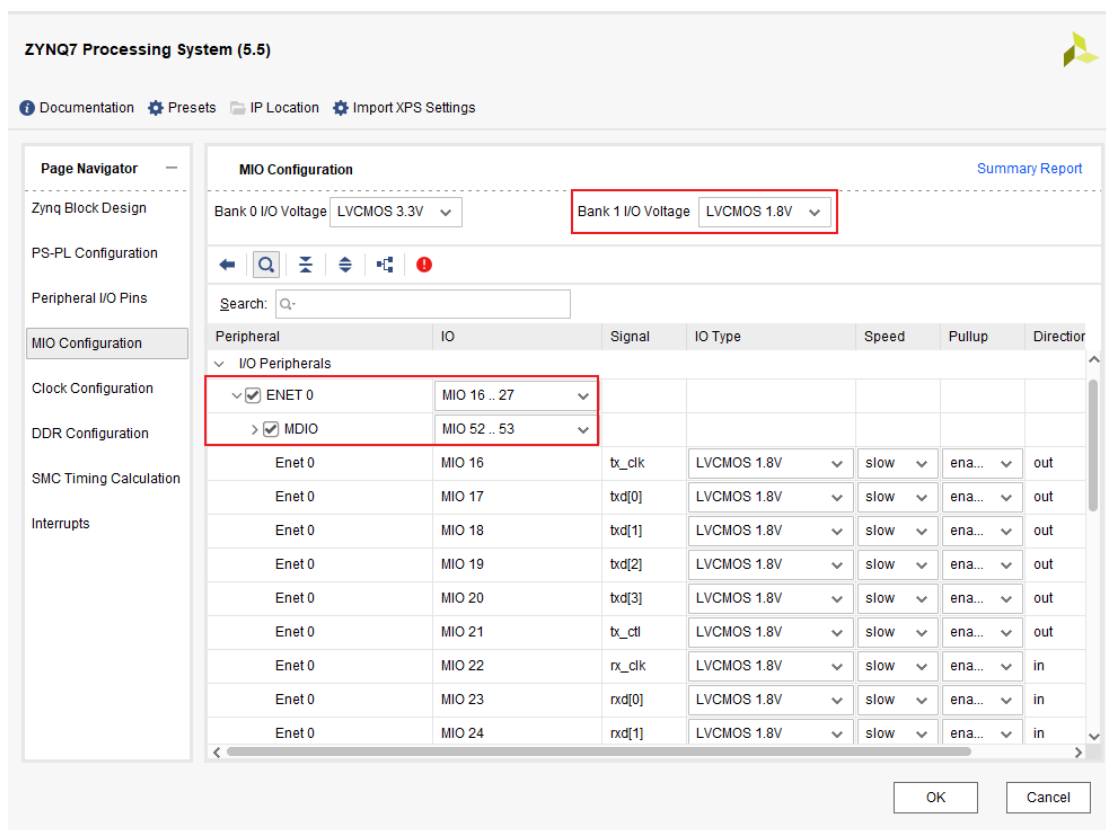


图 4 ZYNQ 核 MIO 配置

这里简单解释下，图 3 中的 MDIO 指的是 MDIO 总线，而图 4 中 MDIO 指的是 MDIO 接口。MDIO 接口是一种简单的双线串行接口，由管理数据的时钟输入总线 MDC 和管理数据输入输出的双向总线 MDIO 构成，也就是图 3 中的 MIO52...53。

LwIP Echo Server 模板工程在进行动态协商时，会通过串口打印相关数据，所以这里我们还需要使能串口外设，ACZ702 开发板上 PS 侧串口引脚对应 MIO48...49，外设使能如图 5 所示：

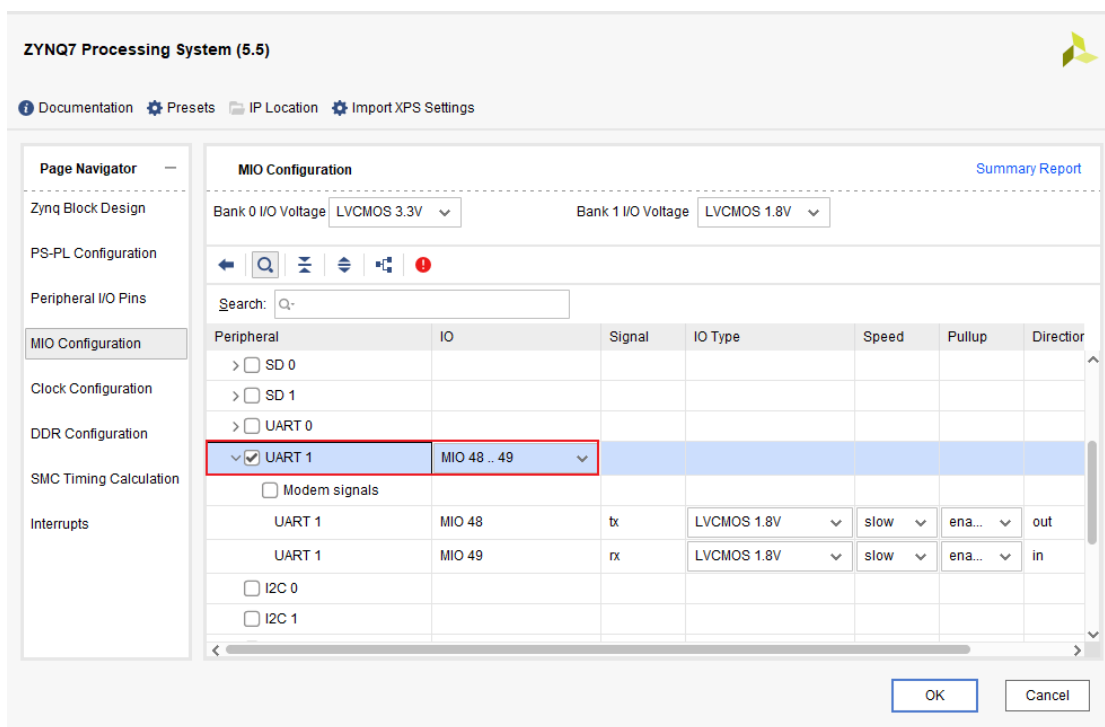


图 5 ZYNQ 核 MIO 配置

随后打开 DDR Configuration 界面，配置 DDR 型号为 MT41K128M16 JT-125，如图 6 所示：

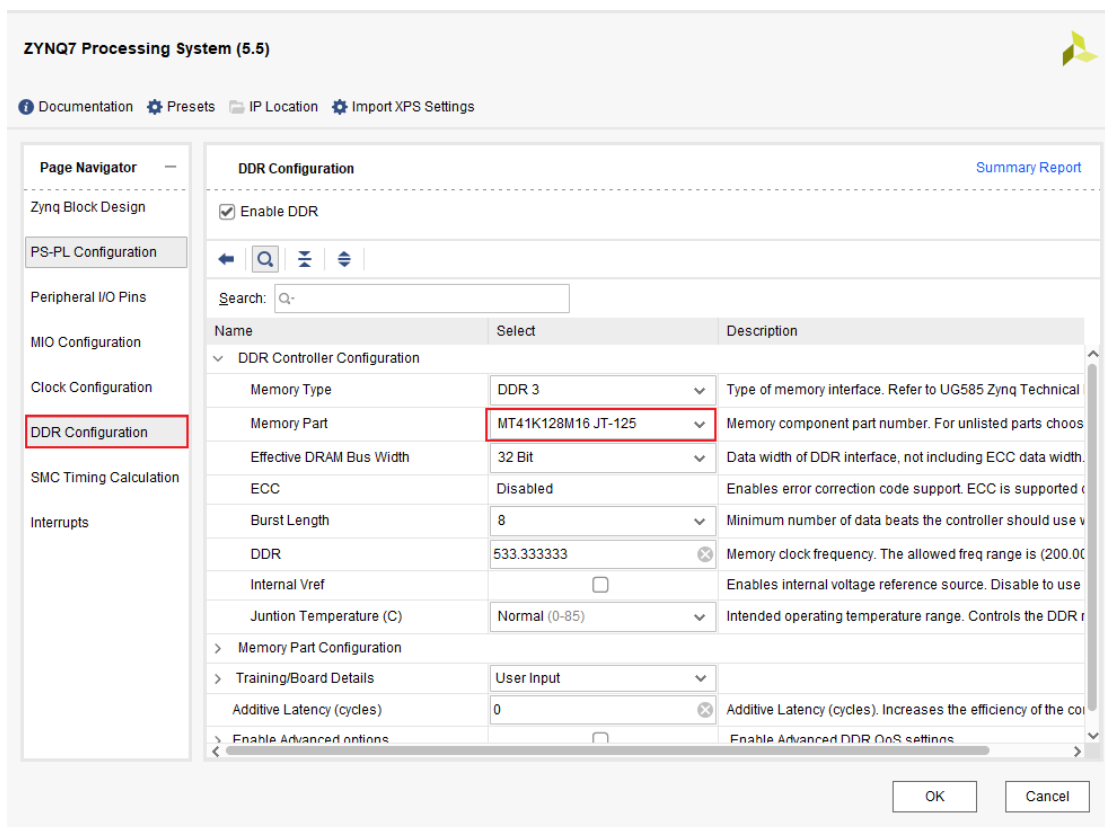


图 6 DDR 型号配置

3.2 端口连接

点击“Run Block Automation”，让软件导出部分引脚，随后手动连接 FCLK_CLK0 和 M_AXI_GP0_ACLK，如图 7 所示：

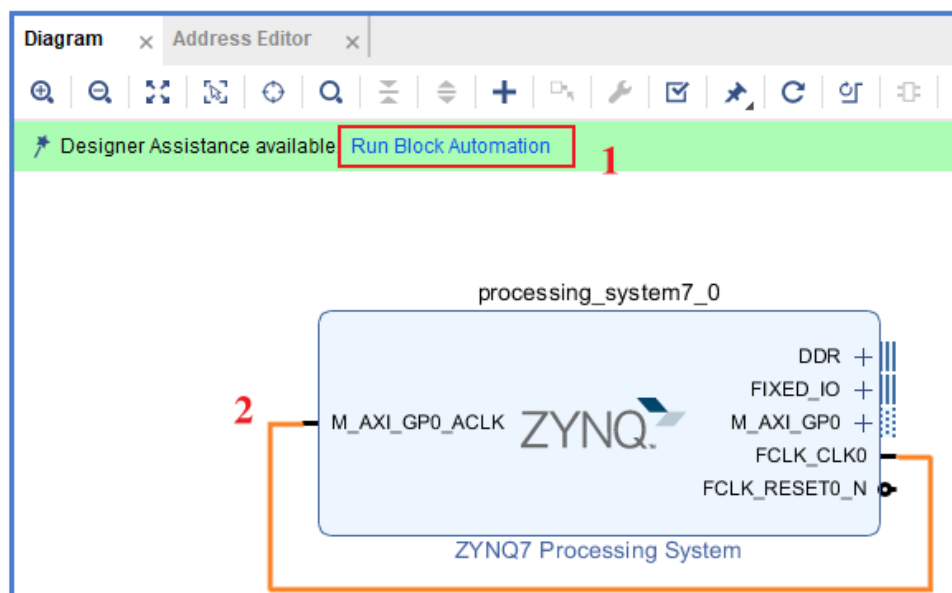


图 7 端口连接

构建完成后的硬件逻辑系统如图 8 所示：

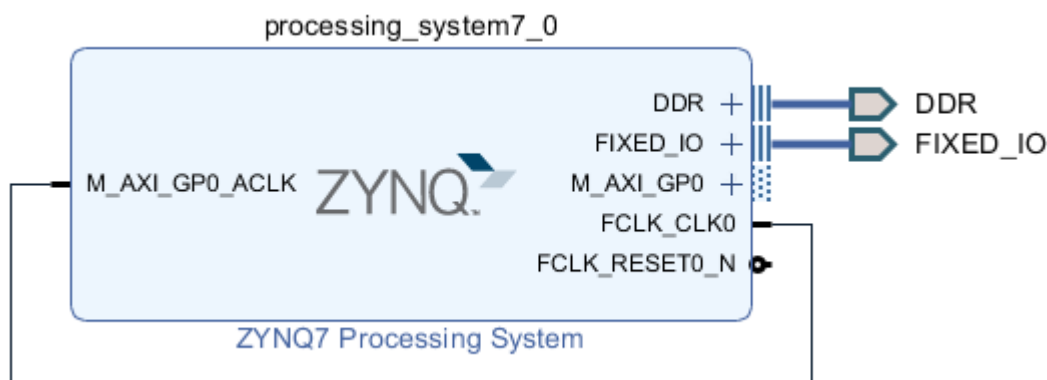


图 8 硬件逻辑系统

3.3 封装设计

接下来右键点击模块设计，生成输出并创建顶层封装，步骤如下：

1. 右键模块设计（sys.bd）
2. 点击 Generate Output Products，等待输出完成
3. 重复步骤 1

4. 点击 Create GHDL Wrapper 生成顶层封装

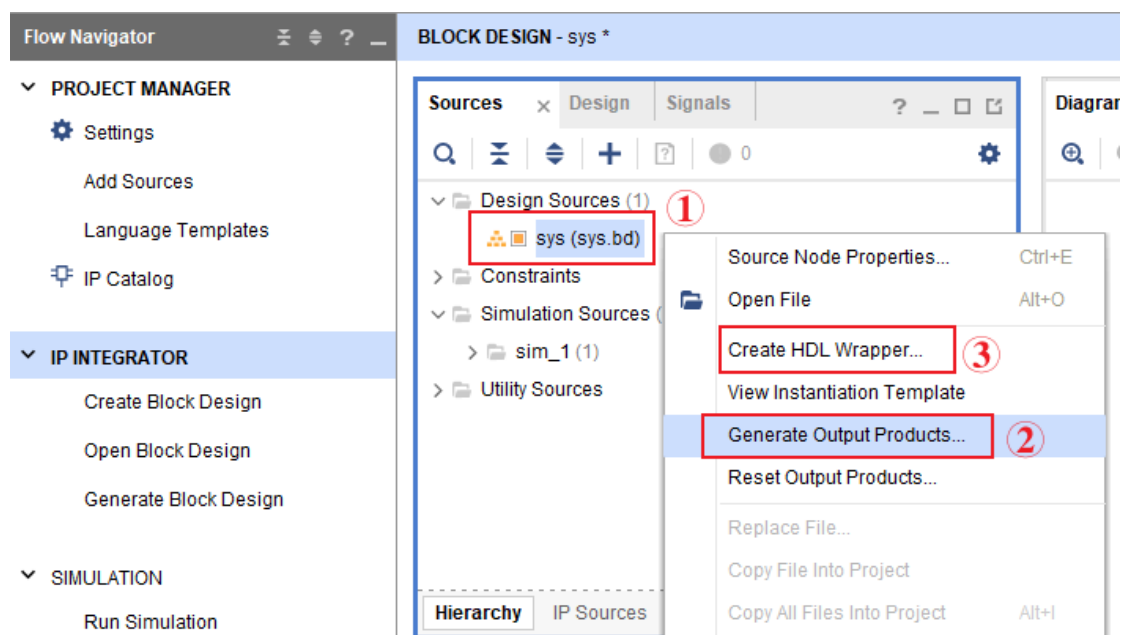
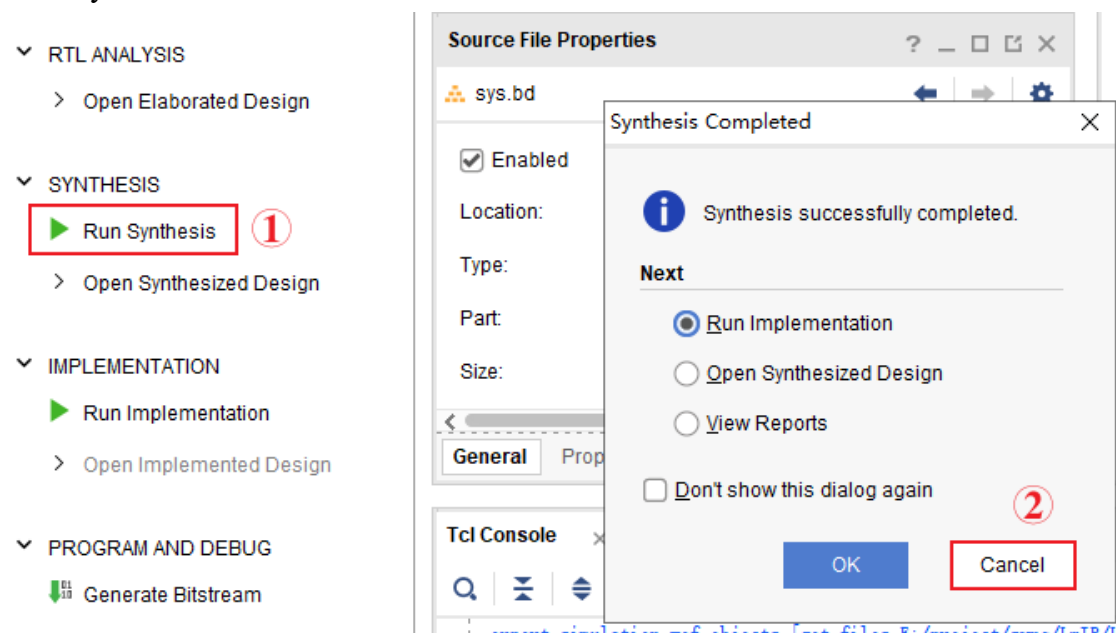


图 9 生成输出并封装

3.4 导出硬件资源描述文件

由于我们全部使用的是 PS 侧的资源，所以不需要进行引脚约束，直接点击 Run Synthesis，对设计进行综合，综合完成后点击 Cancel



随后点击 File，将硬件资源描述文件导出，并运行 SDK，至此便完成了硬件逻辑系统的设计。

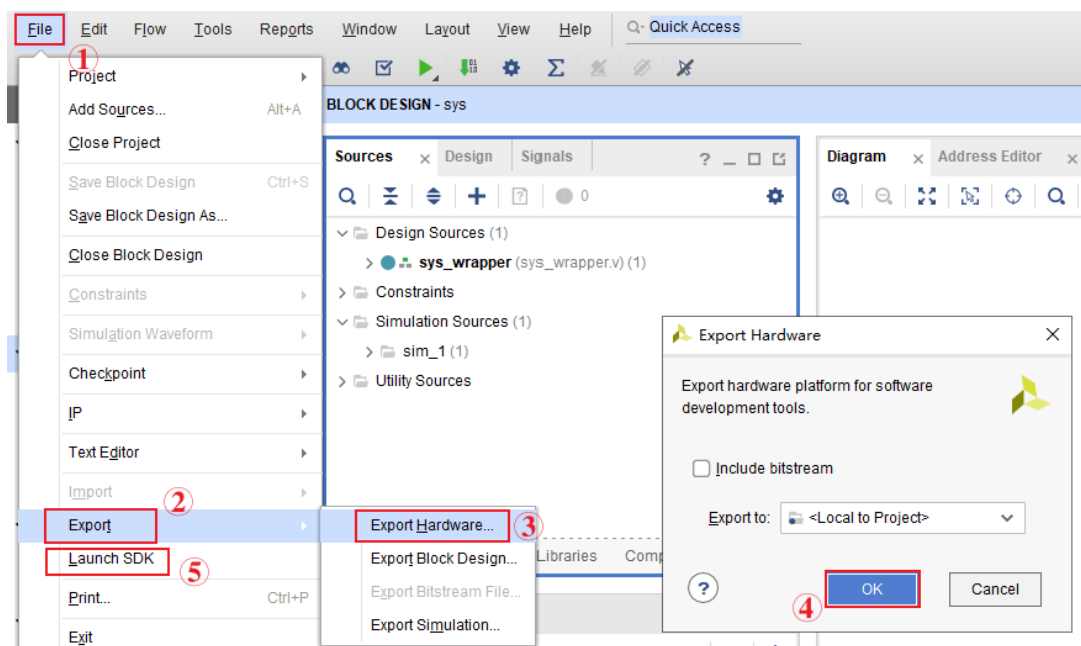


图 10 导出硬件描述文件

4 CPU 软件程序设计

4.1 添加模板

在 SDK 运行后，点击 File 新建一个 SDK 工程，在选则模板时选择 LwIP Echo Server，如图 11 所示：

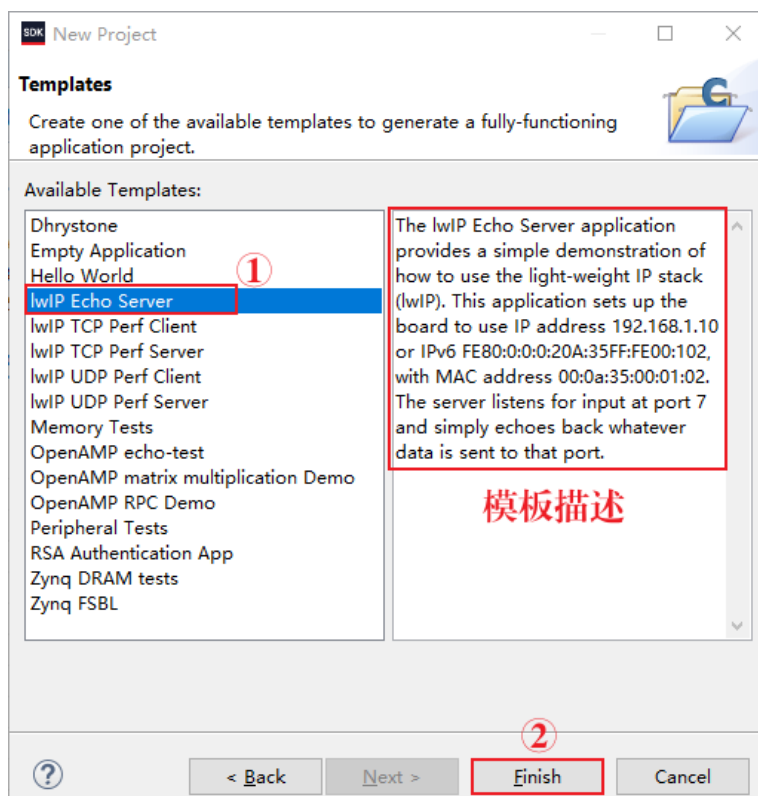


图 11 选择工程模板

模板右侧对模板进行了简单的说明，可以看到，该模板工程会将开发板的 IP 设置为 192.168.1.10，并在端口 7 监听数据，对接收到的数据进行回发。点击 Finish 我们便完成了创建。

4.2 修改参数

接下来我们需要修改 `xemacpsif_physpeed.c` 文件，该文件实现对连接到 MAC 的可用物理层检测、协商和配置速度、配置相关寄存器等。

文件位于 bsp 文件夹下，具体路径参考下图：

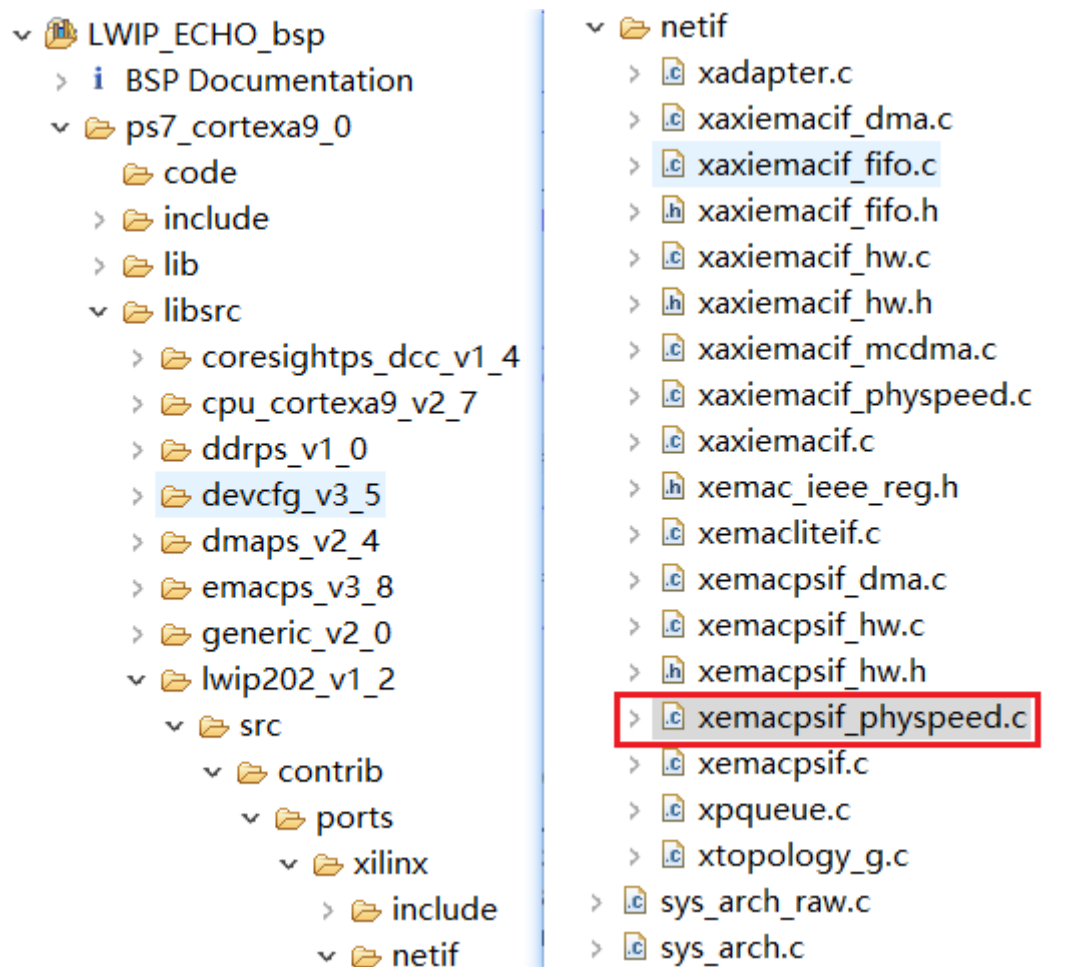


图 12 文件路径

在文件的头文件声明下方添加以下代码，重新定义寄存器地址以及掩码等数值：

```
//此处为了兼容 RTL8211FDI 重定义寄存器地址和掩码
#undef IEEE_SPECIFIC_STATUS_REG
#undef IEEE_SPEED_MASK
#undef IEEE_SPEED_1000
#undef IEEE_SPEED_100

#define IEEE_SPECIFIC_STATUS_REG    0X1A
#define IEEE_SPEED_MASK             0x30
#define IEEE_SPEED_1000             0x20
#define IEEE_SPEED_100              0x10
```

修改完成后我们还需要找到 `get_Realtek_phy_speed` 函数，找到函数中的如下语句，将 `0x400` 改为 `0x4`：

```
XEMacPs_PhyRead(xemacpsp, phy_addr, IEEE_SPECIFIC_STATUS_REG,
                &status_speed);
if (status_speed & 0x400) {
```

```
temp_speed = status_speed & IEEE_SPEED_MASK;

if (temp_speed == IEEE_SPEED_1000)
    return 1000;
else if(temp_speed == IEEE_SPEED_100)
    return 100;
else
    return 10;
}
```

可以看到这段语句用来返回自动协商时网口的速度，首先通过读取状态寄存器的值，随后通过与操作确认链路是否正常，确认链路正常后，将读取的值与速度掩码相与，保留对应的速度位，随后再与速度掩码相比较，得出需要配置的速度值。

修改完成后保存设计，软件默认会自动编译，由于我们修改的是官方底层驱动库，所以编译的时间可能会较长，等待编译完成，确认工程无误我们便完成了对模板的修改。接下来进行板级验证，确认工程是否能够正常运行。

5 板级验证

5.1 所需硬件

1. ACZ702 开发板一个
2. 六类千兆网线一根
3. Type-C 数据线一根
4. 支持 DHCP 的路由器一个

5.2 硬件连接

硬件连接如图 13 所示：

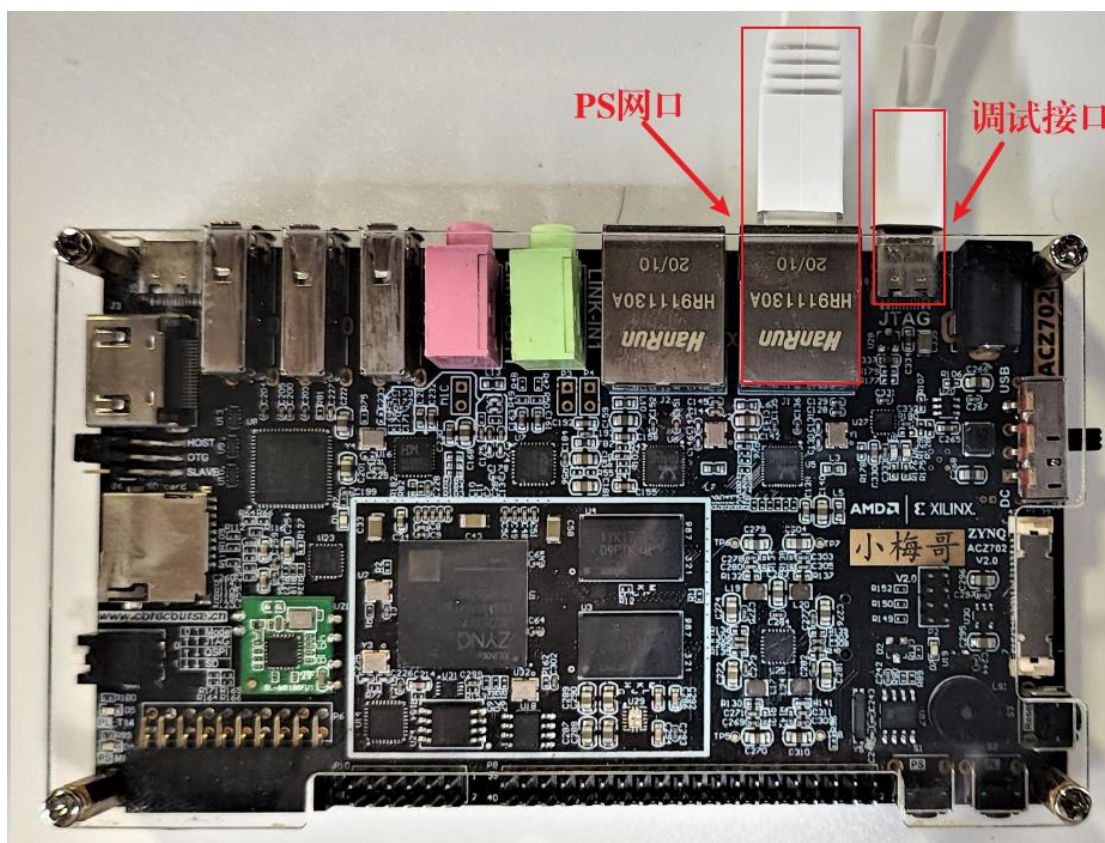


图 13 硬件连接图

设计对供电的要求不高，因此可以直接使用 type-c 供电，开发板上一共有两个以太网接口，靠近调试接口的为 PS 侧的以太网接口，用户可以通过开发板背侧的丝印来确定自己当前连接的是哪个接口。连接网线时，一头连接在开发板 PS 侧网口，另一头连接在电脑/路由器的网口上。这里我们首先测试连接开发板和支持 DHCP 的路由器网口时的情况，因此此时网线一头连接开发板一头连接路由器。

5.3 连接串口终端

接下来我们连接串口终端。打开设备管理器，查询开发板上串口对应端口号，ACZ702V2.0 和 V1.0 的串口芯片不同，对应的名称也不同，笔者电脑端口号如图 14 所示：

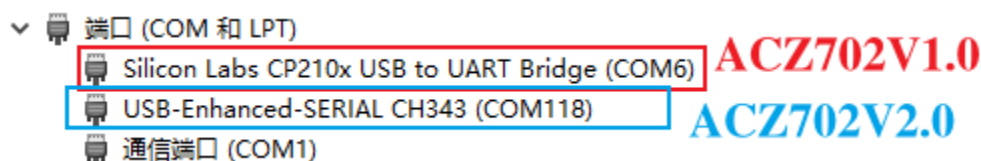


图 14 查询端口号

以 ACZ702V2.0 为例，那么此时对应端口号为 COM118，打开 SDK 终端，添加串口连接，端口号选择为查询到的 COM118，如图 15 所示：

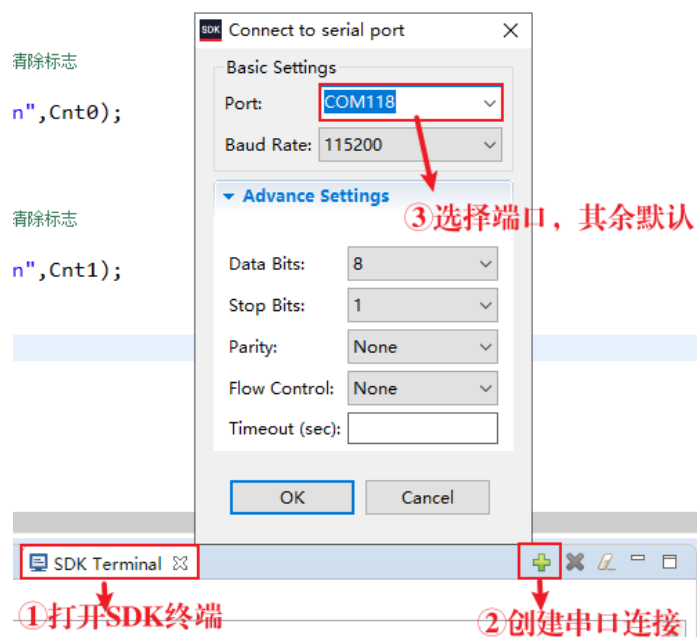


图 15 连接串口

点击 OK，连接完成后会出现图 16 所示连接成功字样：

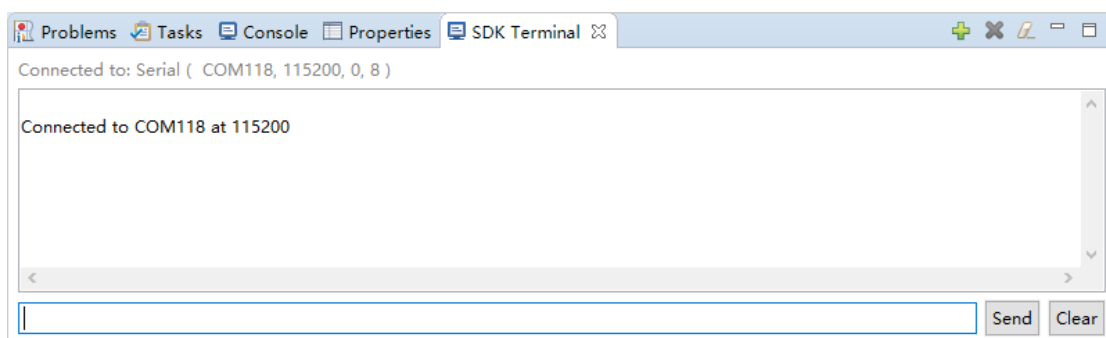


图 16 连接成功

连接成功后便可将设计烧录到开发板中进行验证了。

5.4 下载验证

点击模板工程资源文件，将生成的烧录文件下载至开发板中如图 17 和图 18 所示，烧录流程如下：

1. 双击 GDB 或 System Debugger 新建配置任务，推荐使用 GDB。
2. 在右侧检查是否添加比特流文件和 PS 初始化脚本，通常软件会自动添加，如果没有，用户可以关闭并重新打开该界面或自己手动添加。

3. 确认下方三个选项都被勾选，这四个选项分别是系统复位、配置 FPGA、PS 初始化和 PL-PS 电平转换。后两个选项通常是默认勾选的，用户需要手动勾选前两项以确保 PL 部分能够被正确配置。
4. 点击上方的 **Application** 切换到应用界面。
5. 检查.elf 文件是否添加且烧录任务是否被选中。这里.elf 文件由软件编译产生，参与用户程序的运行。SDK 默认情况下设置了自动编译，用户保存设计后软件便会编译并生成.elf 文件，所以当搜索不到.elf 文件时检查设计是否保存。
6. 点击 “Run” 开始烧录

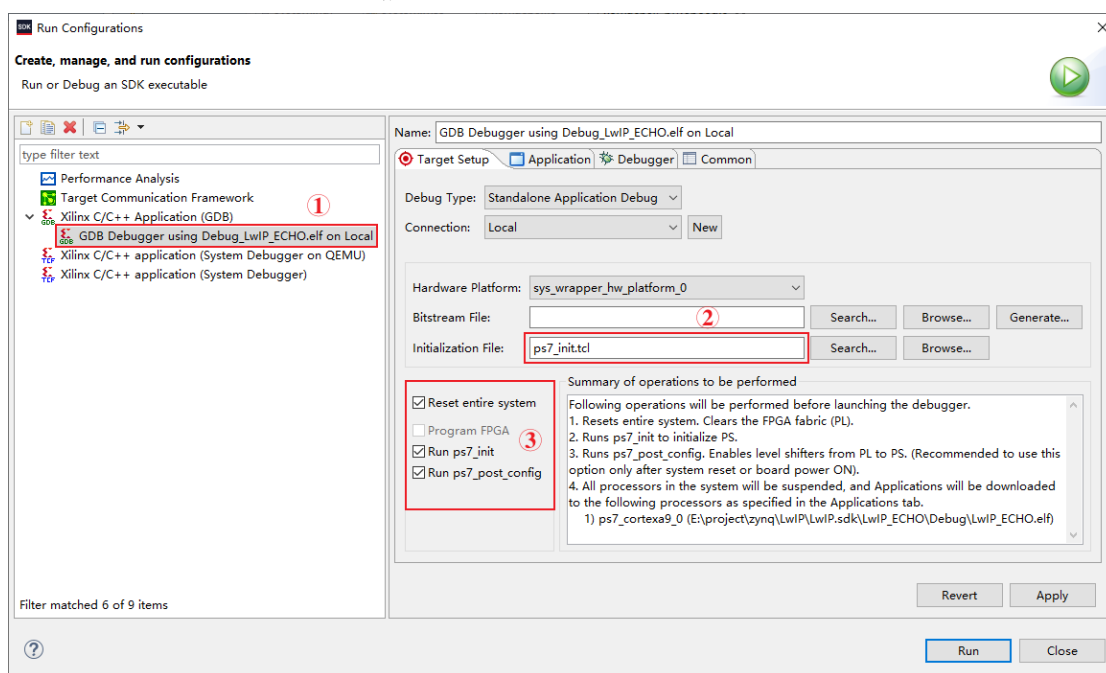


图 17 配置烧录任务

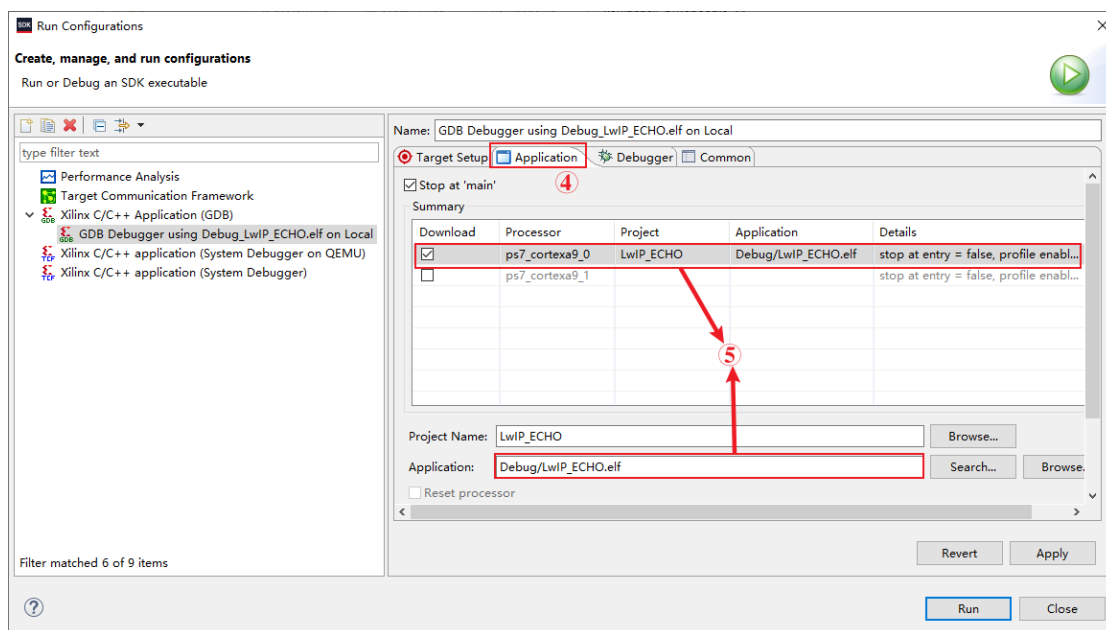


图 18 配置烧录任务

当 SDK 右下角的进度条消失且没有出现报错弹窗时，代表烧录成功。此时串口中打印的数据如下：

-----lwIP TCP echo server -----

TCP packets sent to port 6001 will be echoed back

Start PHY autonegotiation

Waiting for PHY to complete autonegotiation.

autonegotiation complete

link speed for phy address 1: 1000

Board IP: 192.168.6.199

Netmask : 255.255.255.0

Gateway : 192.168.6.1

TCP echo server started @ port 7

图 19 串口打印数据

可以看到，PHY 自动协商成功，链接速度为 1000Mbps，此时开发板 IP 并没有按默认的被设置为 192.168.1.10，而是通过 DHCP 被分配为了 192.168.6.199。

接下来我们再使用网线连接开发板与电脑，为了验证修改后的例程确实能自动协商，我们将电脑网卡的速度设置为 10Mbps，将 IP 设置为静态 IP192.168.1.100，具体设置步骤如下：

1. 右键单击电脑的网络连接状态图标，选择“打开网络和 Internet 设置”
2. 点击更改适配器选项
3. 双击本地连接，在网络状态页面点击属性
4. 找到“Internet 协议版本 4”，双击进入，设置 IP 地址为静态的 192.168.1.100，子网掩码为 255.255.255.0，随后点击确认
5. 返回属性界面后依次点击配置和高级
6. 找到“连接速度和双工模式”项，设置为“10Mbps 全双工”

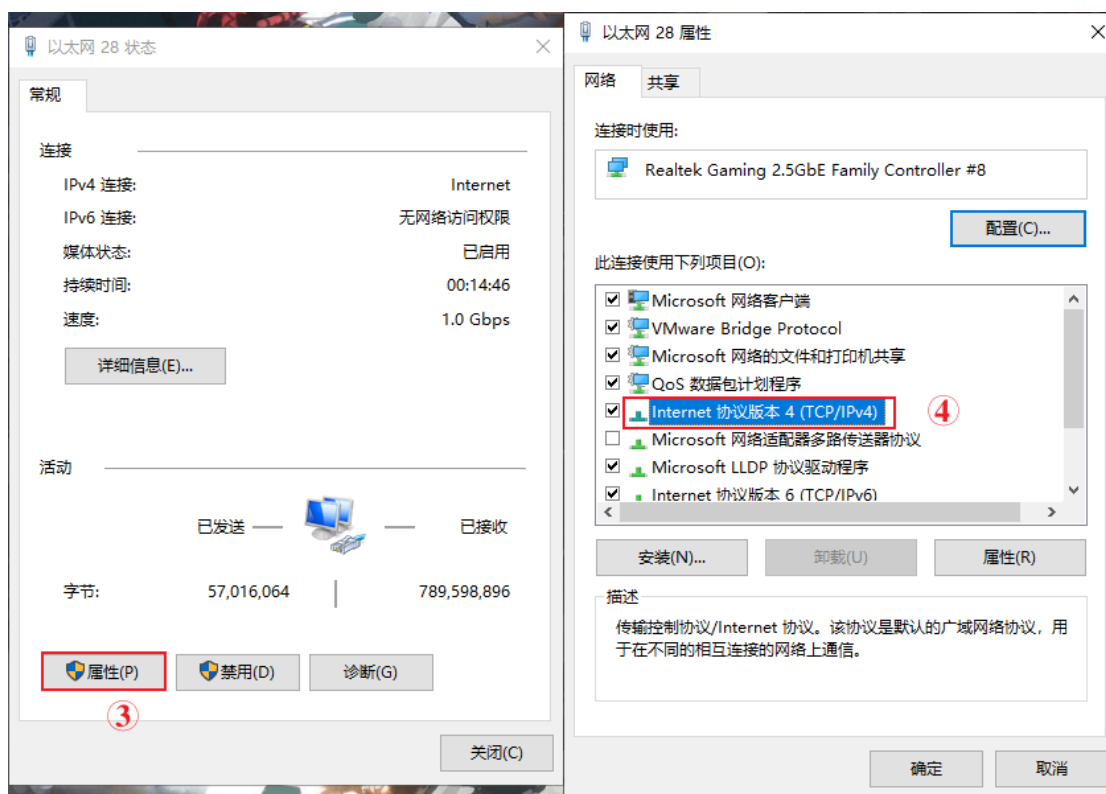


图 20 修改 IPv4 参数

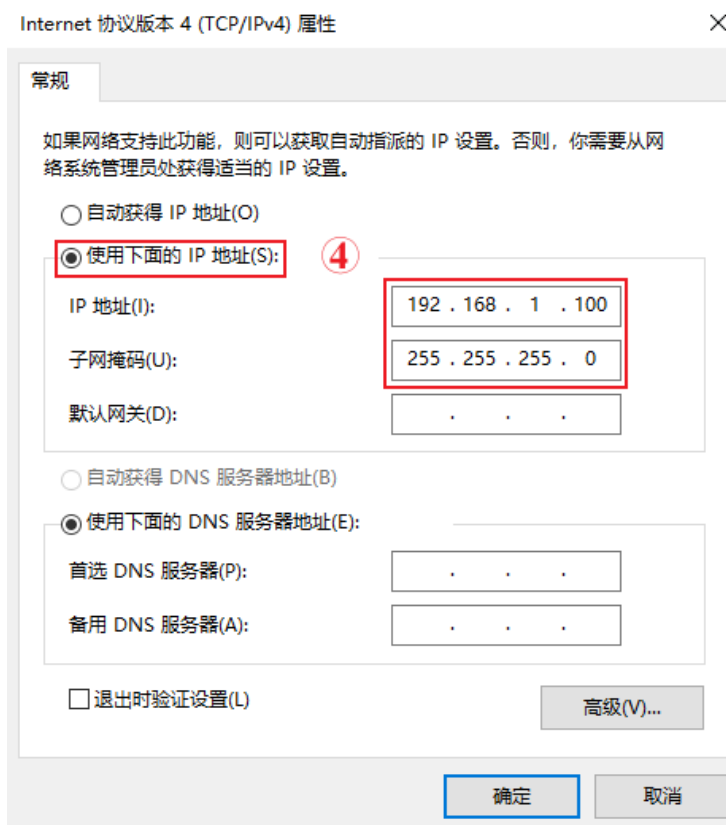


图 21 设置静态 IP

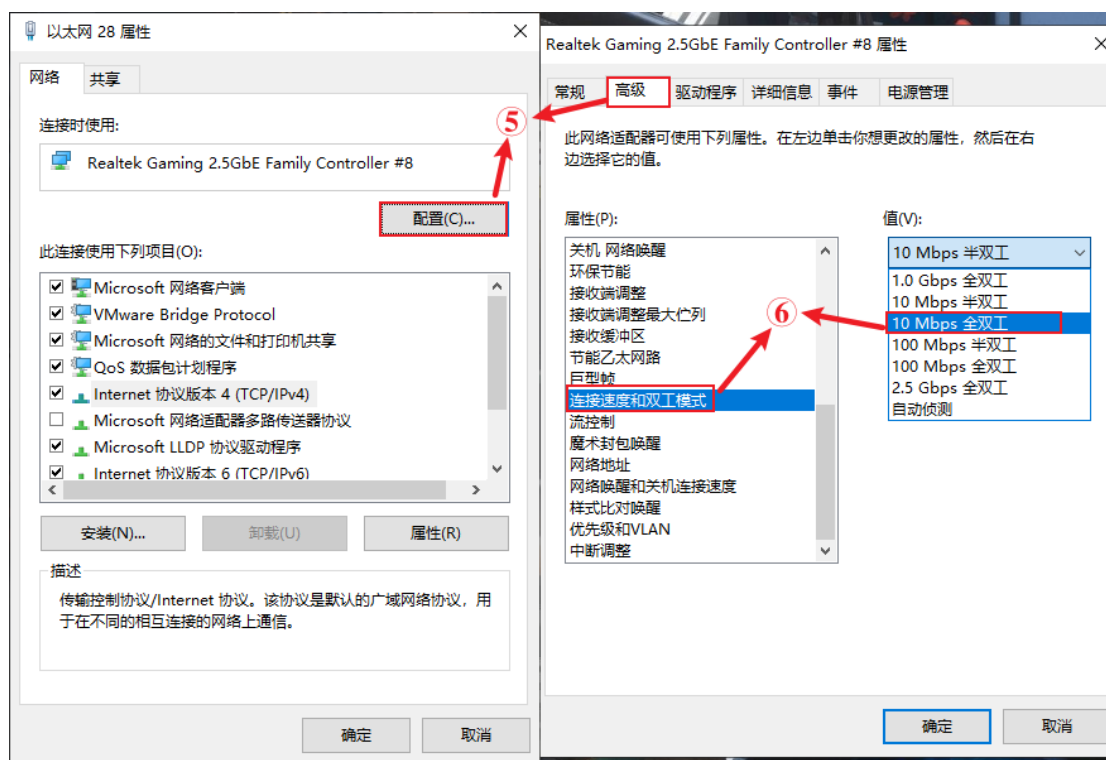


图 22 修改连接速度

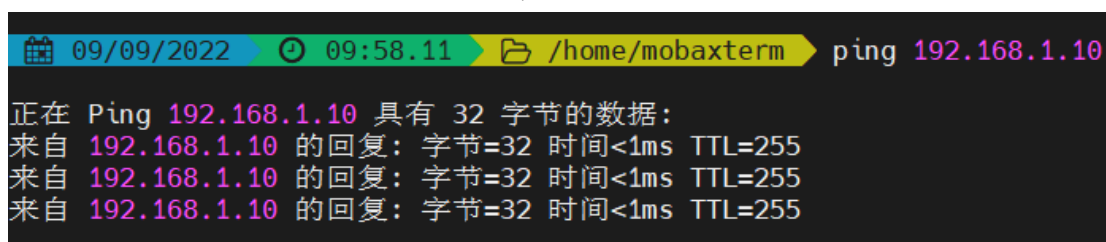
设置完成后，再次将烧录文件下载到开发板中，等待烧录完成后，可以看到串口中打印的数据如下：

```
-----lwIP TCP echo server -----  
  
TCP packets sent to port 6001 will be echoed back  
  
Start PHY autonegotiation  
Waiting for PHY to complete autonegotiation.  
autonegotiation complete  
link speed for phy address 1: 10  
DHCP Timeout  
Configuring default IP of 192.168.1.10  
Board IP: 192.168.1.10  
  
Netmask : 255.255.255.0  
  
Gateway : 192.168.1.1  
  
TCP echo server started @ port 7
```

图 23 串口打印数据

可以看到此时检测出的速度为 10Mbps，由于 windows 默认是没有安装 DHCP 驱动的，因此这里 DHCP 超时。

最后，打开 MobaXterm 软件，用电脑 ping 开发板，测试此时的链路是否正常，网络是否通畅（需要关闭网络防火墙）。



```
09/09/2022 09:58.11 /home/mobaxterm ping 192.168.1.10  
  
正在 Ping 192.168.1.10 具有 32 字节的数据：  
来自 192.168.1.10 的回复: 字节=32 时间<1ms TTL=255  
来自 192.168.1.10 的回复: 字节=32 时间<1ms TTL=255  
来自 192.168.1.10 的回复: 字节=32 时间<1ms TTL=255
```

图 24 检测链路状态

可以看到，成功 ping 通，我们对例程的修改成功，模板例程能够正确的运行在 ACZ702 开发板上。

6 总结

本节主要介绍了如何修改官方代码，让官方的 LwIP 模板例程能够运行在

ACZ702 开发板中，并通过一系列验证，验证了修改后模板例程的可行性。最后，这里再为大家补充几点大家跟随设计时可能需要用到的知识点：

1. 无法修改 IP 为静态 IP 地址

出现这种情况，通常都是相关驱动缺少部分文件造成的，此时我们可以使用管理员权限打开 cmd，在其中输入以下代码来修改静态 IP：

```
netsh interface ip set address "以太网名" static 静态 IP 255.255.255.0
```

其中，红色部分为需要用户自己输入的部分，以太网名通常由“以太网”+“空格”+“数字”组成，用户输入时，注意不要忘记空格。

如果想将 IP 恢复到动态 IP 地址，只需要使用以下语句：

```
netsh interface ip set address name="以太网名" source=dhcp
```

2. 修改 LwIP 源工程模板

前面我们对模板例程的修改，只是针对本次创建的模板例程而言的。一旦我们重新生成 bsp，或者新建一个同样模板工程是，都需要再次修改。

对此，我们可以直接修改工程模板的源文件（修改前请一定要进行备份，方便恢复），具体步骤如下：

- a) 找到 Vivado 安装目录下的 Xilinx\SDK\2018.3\data\embeddedsd\ThirdParty\sw_services\lwip202_v1_2\src\contrib\ports\xilinx\netif 路径下的 xemacpsif_physpeed.c 文件
- b) 按照前文中的修改方法来修改它，然后保存。修改完成后，新创建的 LwIP 模板例程，都不需要再进行修改。