1 基于 LWIP 模板的 TCP 回环测试

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，本节将以 BX71 开发板为例，教大家如何通过 xilinx 提供的 LWIP 模板，在 ZYNQ 系列器件上实现 TCP 回环通信。

1.1 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。

1.2 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，导致用户在直接使用官方模板工程时，会出现自动协商失败的情况。该情况与 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

DRAFT

图 1-1 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 1-2 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。

1.3 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。

1.3.1 IP 核添加与配置

对于设计而言，要想使用 LWIP，我们便需要添加 ZYNQ 核，使能以太网外设控制器。然而，在 BX71 开发板上，以太网接口位于 PL 侧，且使用的 RGMII 接口。

因此，我们需要通过 EMIO 将以太网控制器的引脚路由到 PL 端。除此之外，查询 UG585 手册中以太网章节可知，当以太网控制器的管脚通过 EMIO 路由到 PL 时，其接口类型为 GMII，如下图所示：

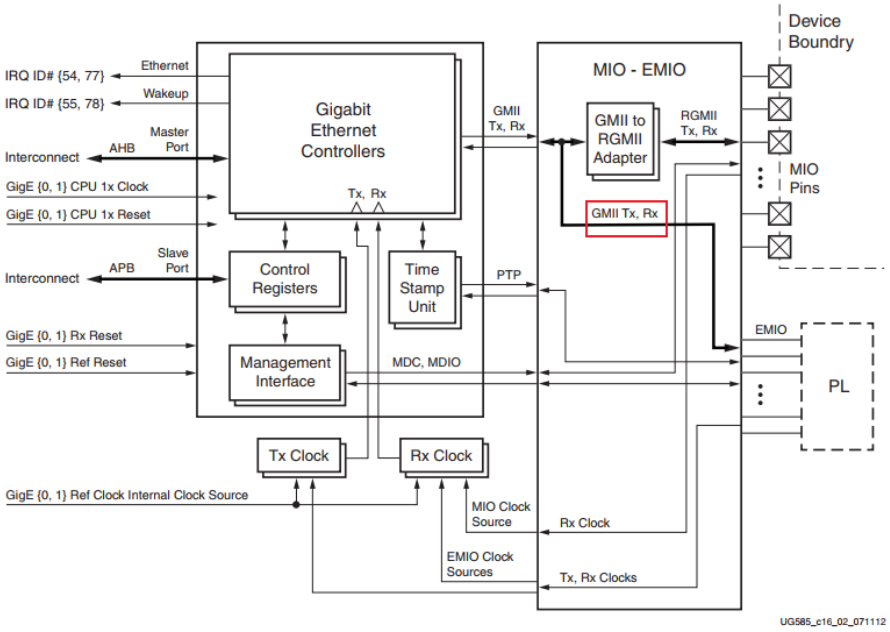


图 1-3　以太网控制器系统结构图

所以，除了 ZYNQ 核之外，设计还需要添加一个 GMII 转 RGMII 核，用来实现接口类型的转换。该 IP 核与 ZYNQ-7000 系列器件的连接方式如下图：

Product Specification

Figure 2-1 illustrates the connection of the Gigabit Ethernet Controller in the Zynq®-7000 SoC to the GMII to RGMII core. The same connection is applicable for Zynq® UltraScale+™ MP SoC too.

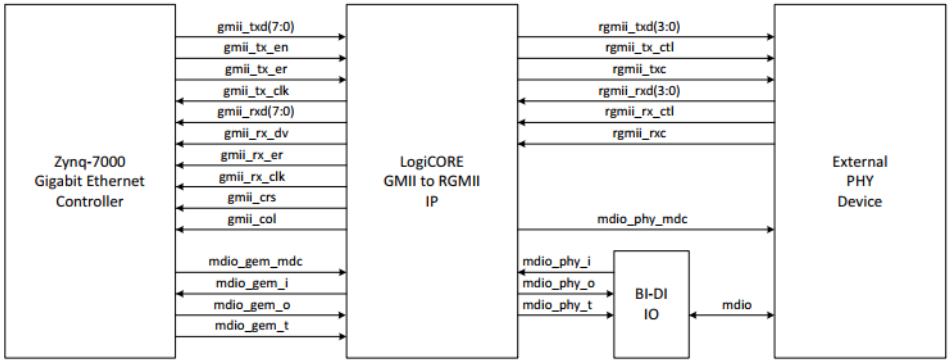


Figure 2-1: **GMII to RGMII Core Ports and Interfaces**

IMPORTANT: *The MDIO interface is necessary for the operation of the core because the auto-negotiated speed of operation from the PHY is communicated to the Ethernet MAC through MDIO.*

The clock input is 200 MHz for Zynq-7000 and 375 MHz for Zynq UltraScale+ MPSoC. It is used as a reference clock for the IDELAYCTRL elements and input for the management modules.

If the GMII clock is sourced internally (C_EXTERNAL_CLOCK = 0), this 200/375 MHz clock is the input clock to the MMCM from which the TX clocks for all line rates (125/12.5/2.5 MHz for 1000/100/10 Mb/s, respectively) are generated.

图 1-4 GMII to RGMII 与 ZYNQ-7000 系列器件连接方式

该图出自 IP 手册 pg160，除了连接结构外，从图中还可得知，对于 ZYNQ-7000 系列器件，在使用 GMII to RGMII 核时，需要为其提供 200MHz 的输入时钟。

除了以上 IP 核外，最后，设计还需要添加一个 Constant 核，用来输出常量，控制以太网复位信号。

综上，在创建好 block design 后，本次设计需要向其中添加以下 IP 核：

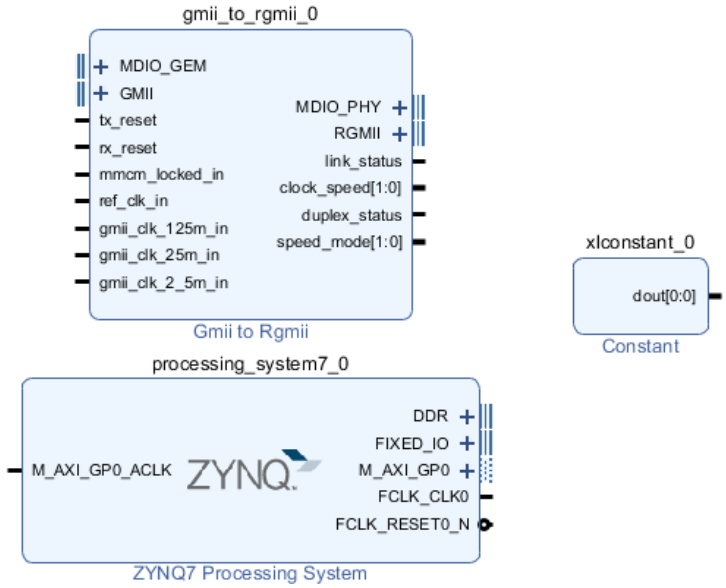


图 1-5 本次设计所需添加 IP 核

添加完成后开始配置，首先是 ZYNQ 核，根据上述分析，这里需使能其中任意一个以太网外设控制器，并设置其引脚路由为 EMIO。除此之外，这里还需要使能 PS 端串口，用来后续打印调试信息，如图 1-6 所示：

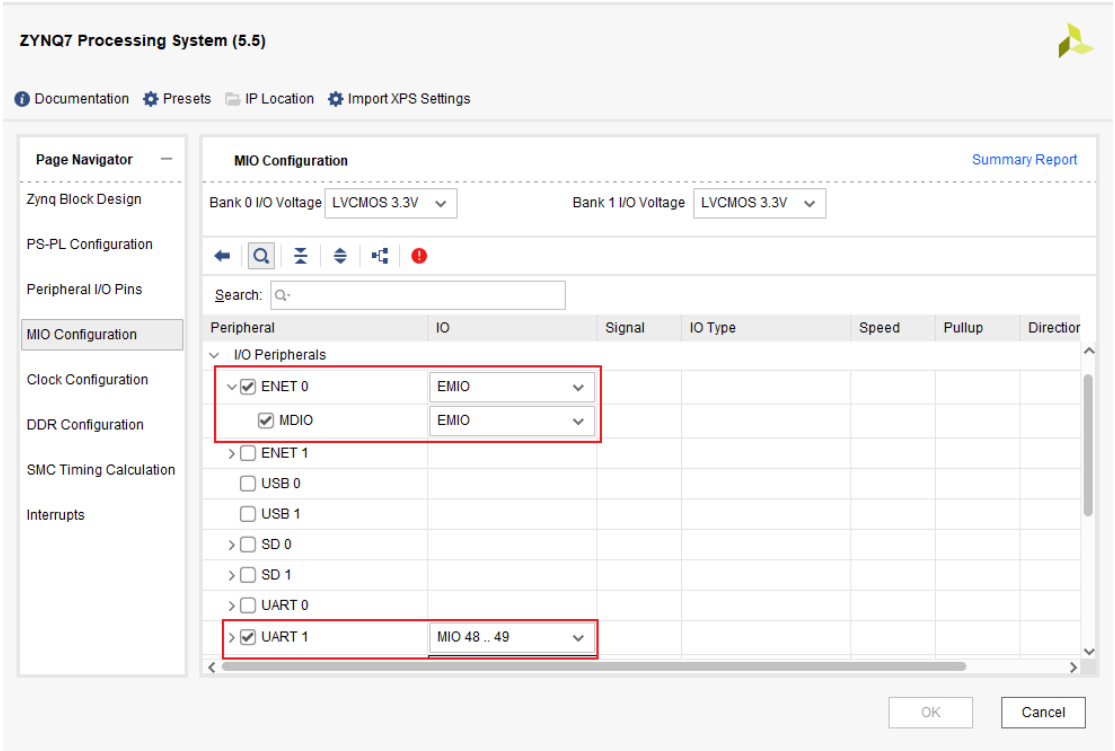


图 1-6 使能以太网外设控制器

随后，配置 DDR 型号及总线位宽便可完成 ZYNQ 核的配置，如下图：

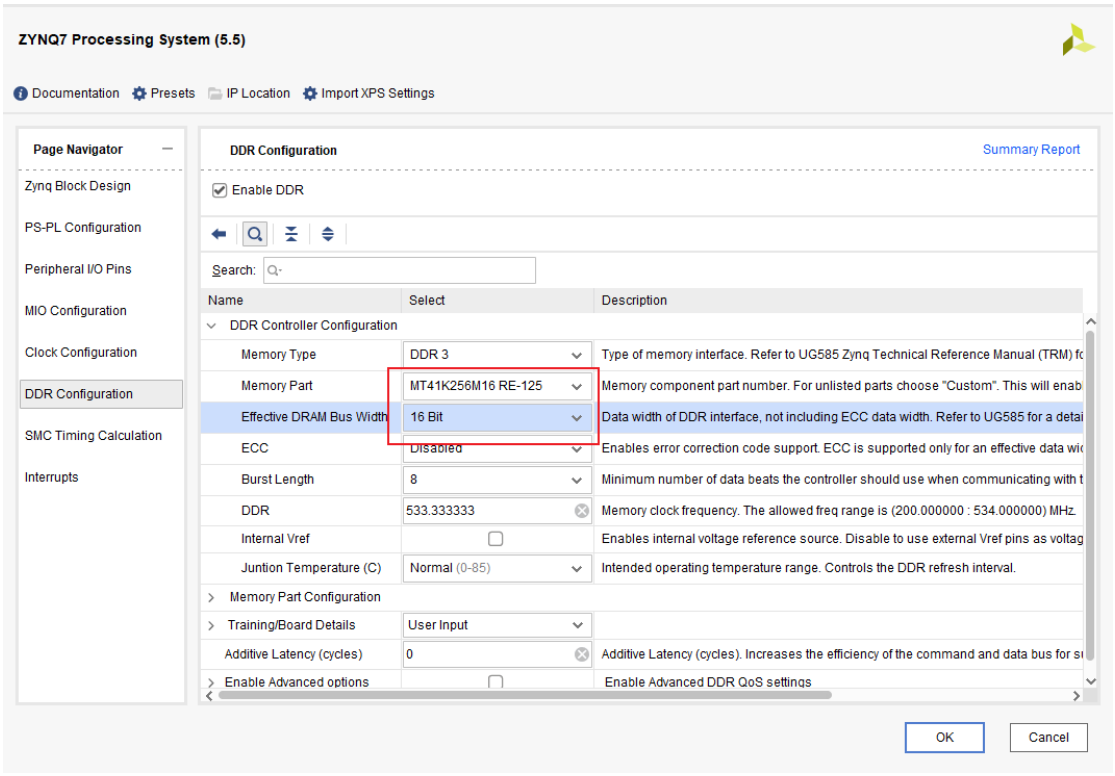


图 1-7 使能并配置 DDR 控制器

最后是时钟配置，根据前面的内容，这里需要配置 ZYNQ 核输出 200MHz 的时钟，用来作为 GMII to RGMII 核的输入时钟，如下图所示：

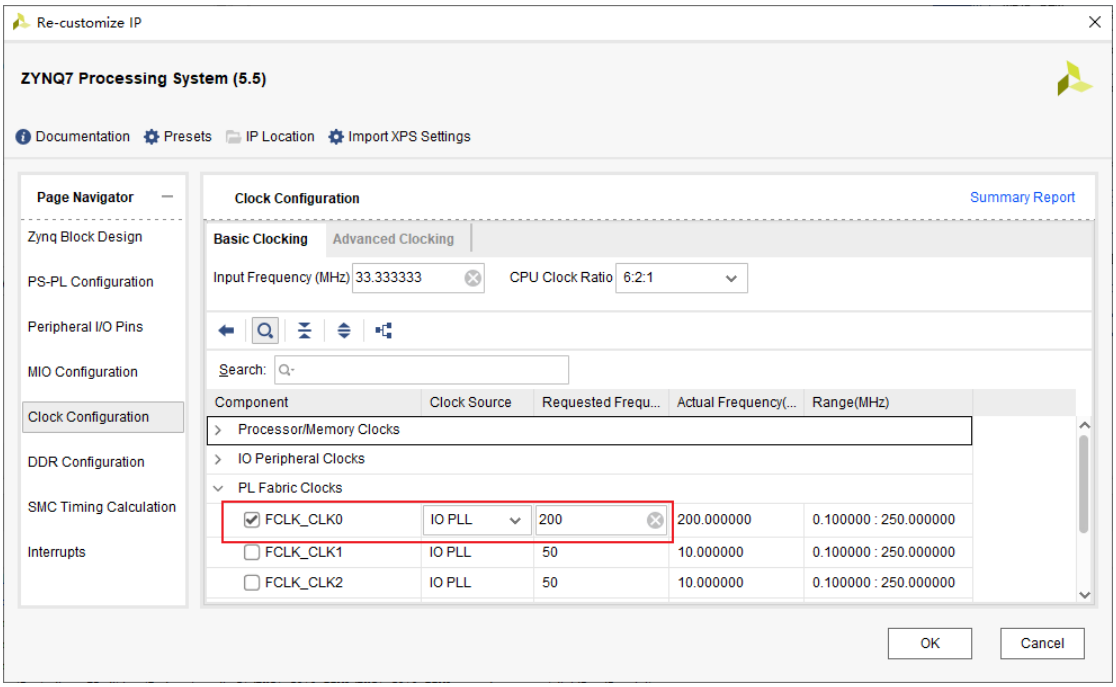


图 1-8 配置 ZYNQ 输出 200MHz 的时钟

接下来是 GMII to RGMII 核，其配置如下：

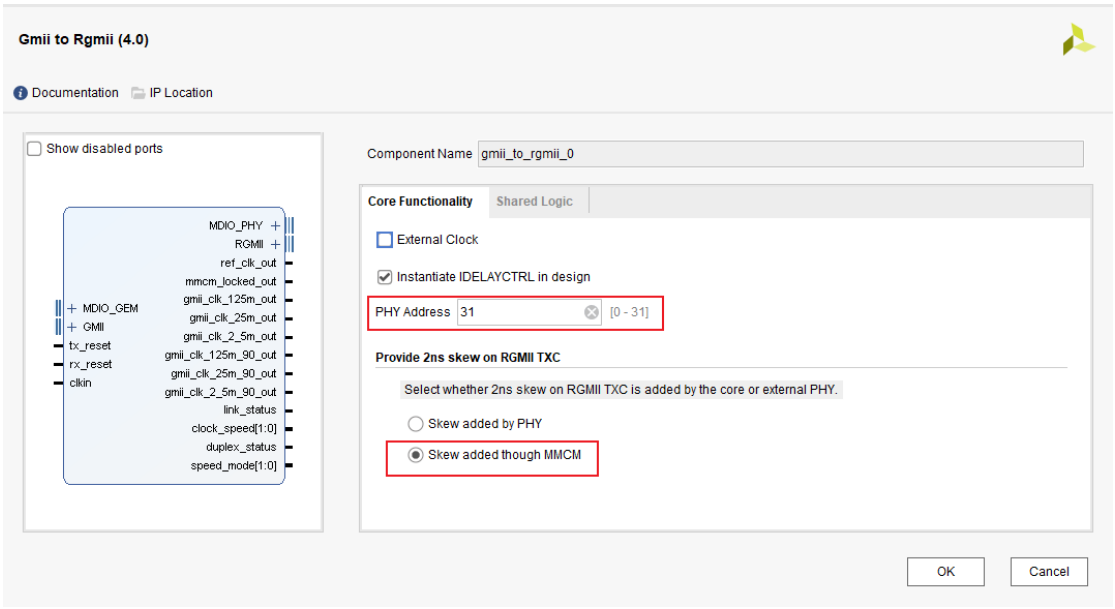


图 1-9 GMII to RGMII 配置页 1

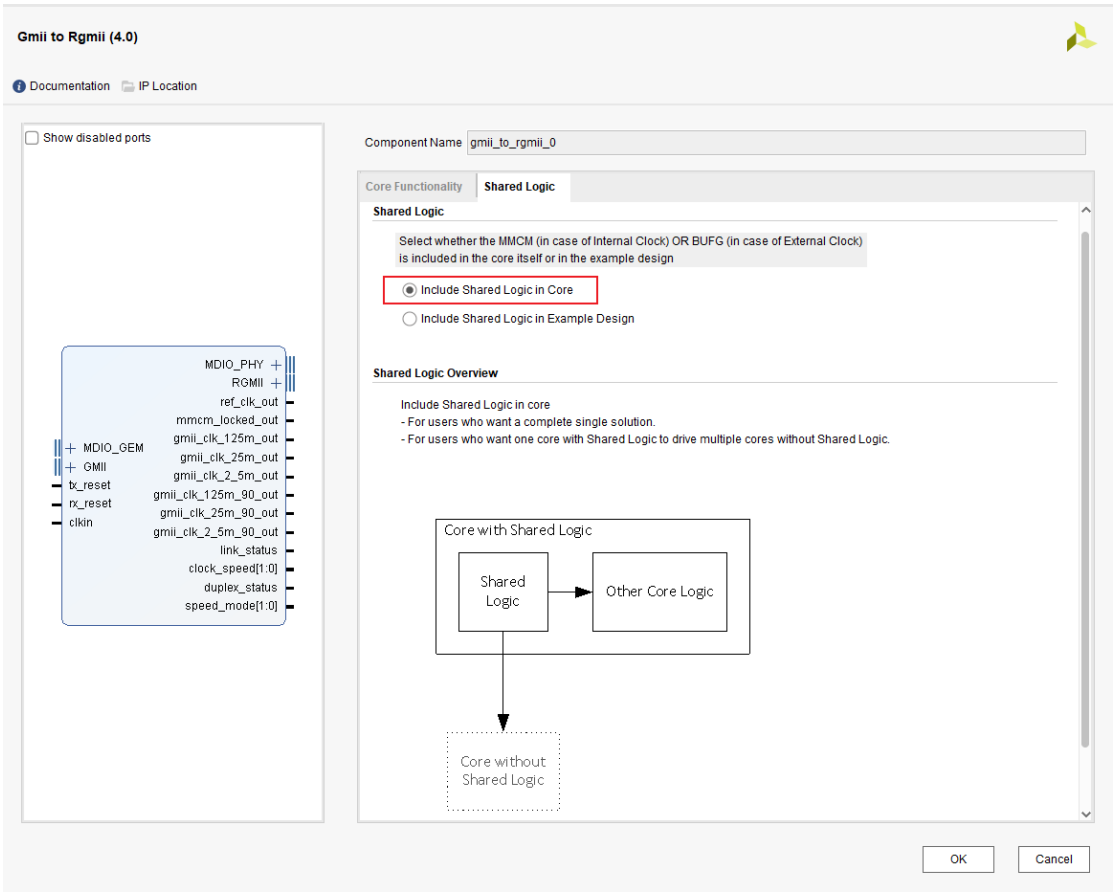


图 1-10 GMII to RGMII 配置页 2

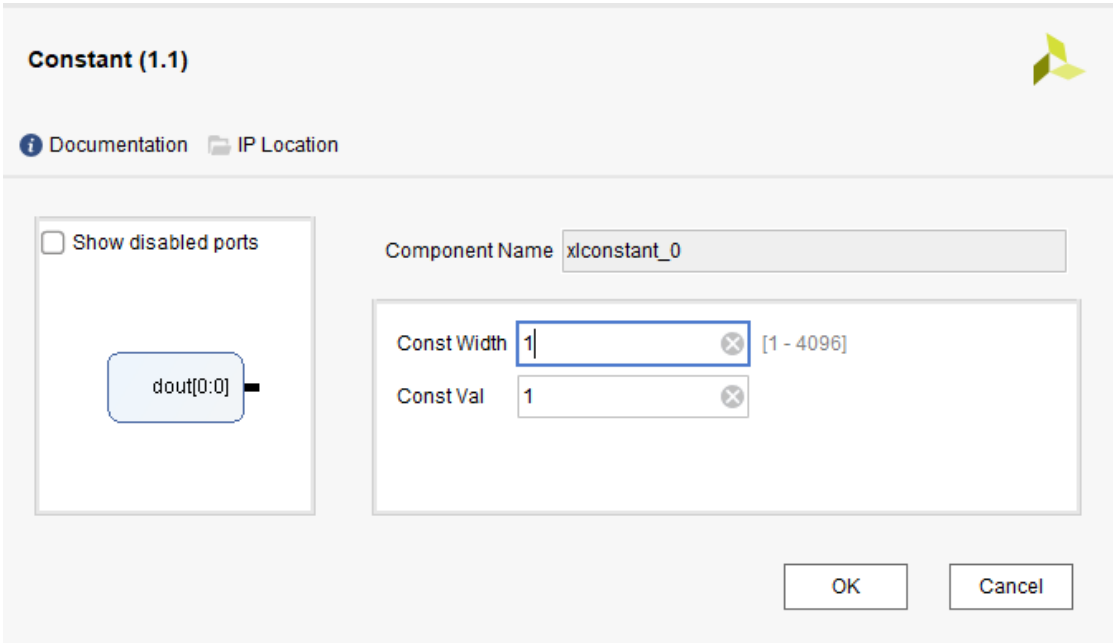
这里讲一下涉及到的几项配置：

- PHY Address，该项不是以太网 PHY 地址，而是用于标识 MDIO 事务中

core 的虚拟地址。因此该项只需设置一个与板载 PHY 不同的地址即可。

- Provide 2 ns Skew on RGMII TXC：选择为 RGMII 的 TXC 添加 2ns 的偏斜，用户可以选择由外部 PHY 添加，或由 IP 核通过 MMCM 添加。
- Shared Logic：选择是否需要共享时钟资源。当设计中只有一个 core，或者存在多个 core 但需要一个 core 共享时钟资源，以驱动其余核时，需要勾选 Include Shared Logic in the Core。而只有当设计中存在多个 core，且已经有一个 core 共享了时钟资源用来驱动其他核时，才需要勾选 Include Shared Logic in the Example Design。

最后是 Constant 核的配置，该核用于控制以太网复位信号，因此这里我们只需给一个常量 1，确保以太网不会被复位即可，配置如下：



配置完成后，IP 整体如下图所示：

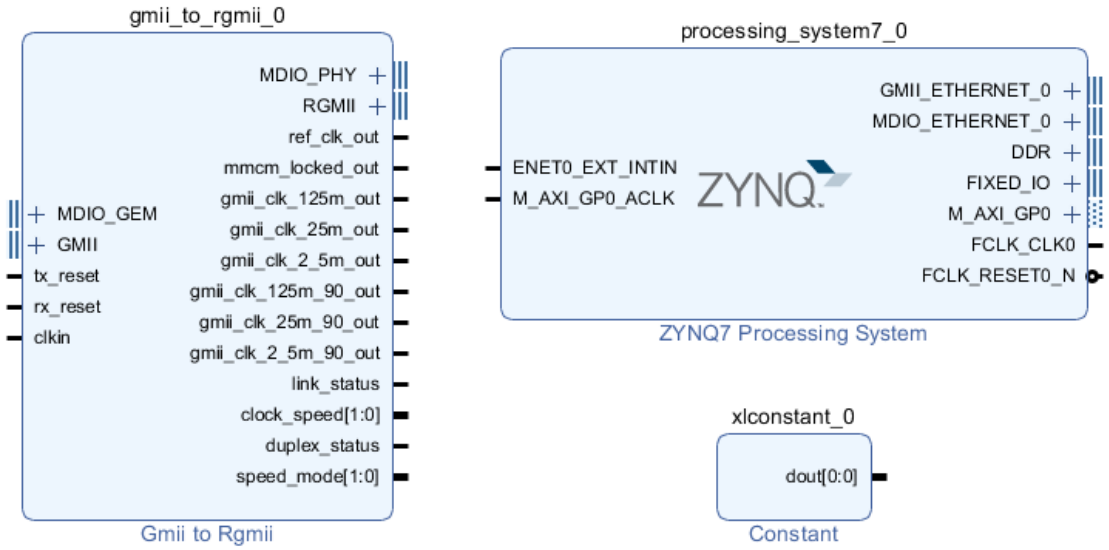


图 1-11 配置完成后 IP

1.3.2 端口连接

点击“Run Block Automation”，让软件导出 PS 相关引脚，如图 1-12 所示：

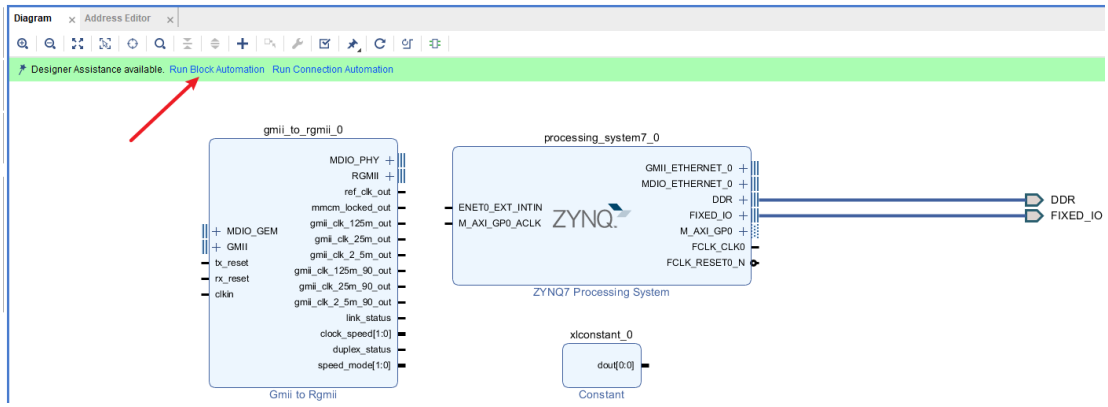


图 1-12 导出 PS 相关引脚

随后，手动连接 GMII 接口和 MDIO 接口，并为设计连接时钟：

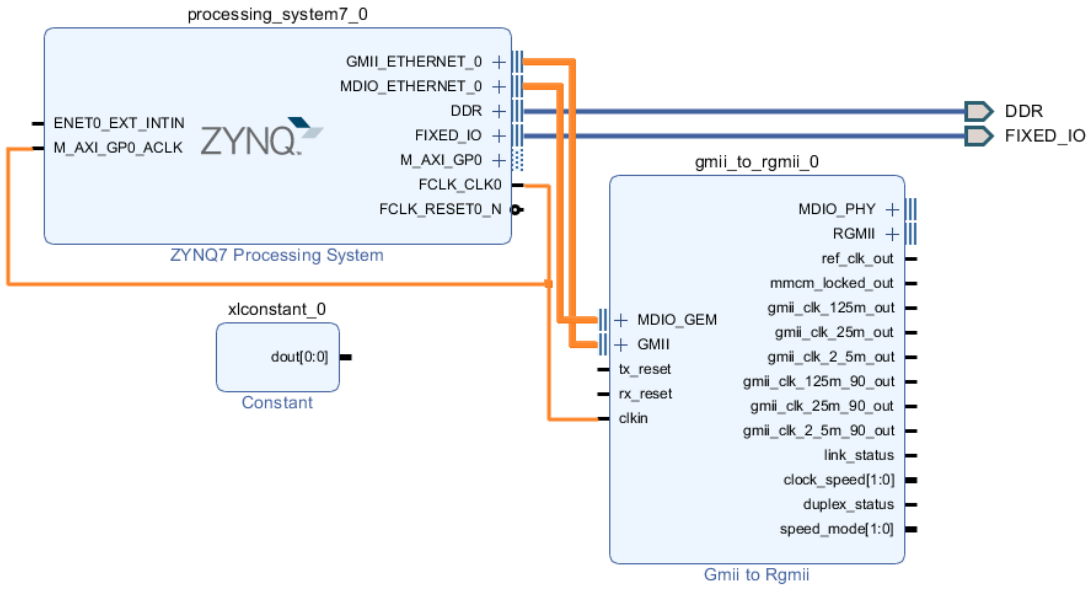


图 1-13 接口连接 1

最后，将 MDIO_PHY、RGMII、dout 端口引出，为了方便区分，这里也可为 dout 信号重新命名。如此，便完成了硬件系统构建，如下图：

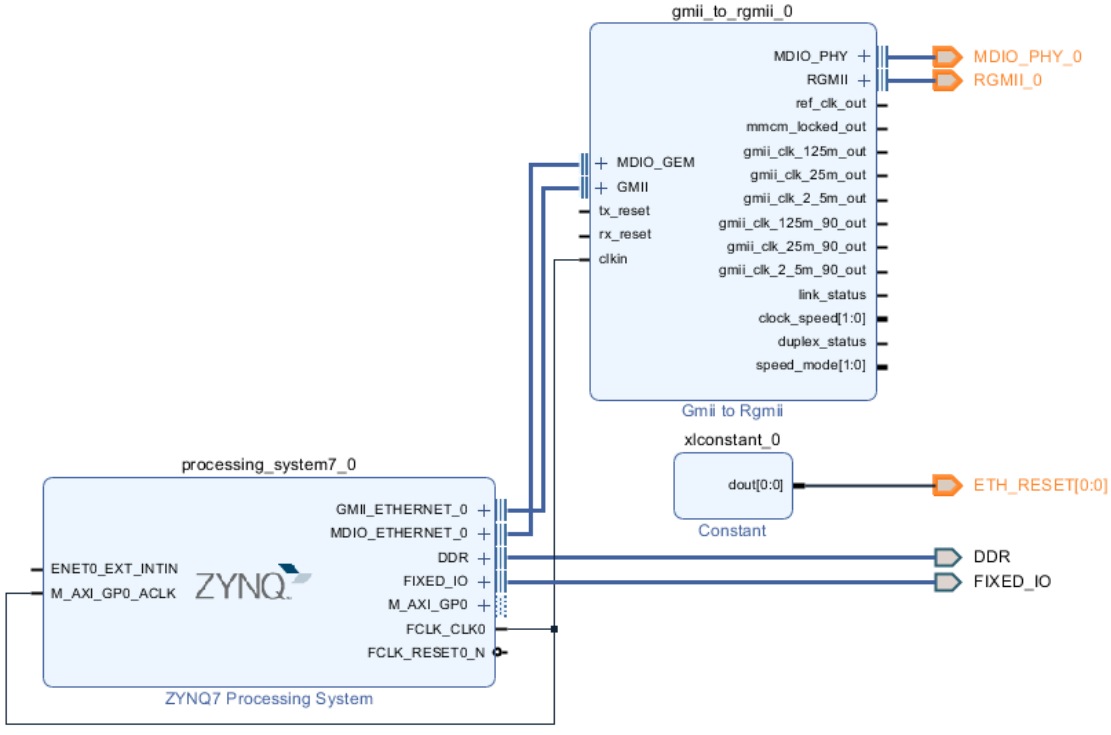


图 1-14 硬件逻辑系统

1.3.3 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，等待输出生成完成

3. 重复步骤 1

4. 点击 Create GHDL Wrapper 生成顶层封装

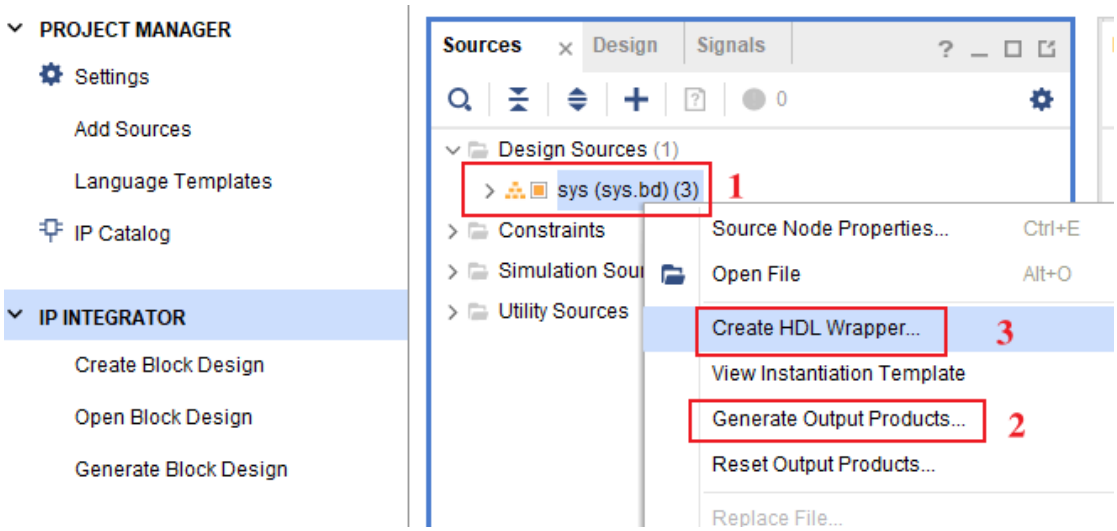


图 1-15 生成输出并封装

1.3.4 管脚约束与硬件规范导出

由于设计涉及到 PL 端以太网，因此在封装完 BD 后，还需分配相关管脚。点击 Open Block Design 进入管脚分配界面，本次管脚分配如下图所示：

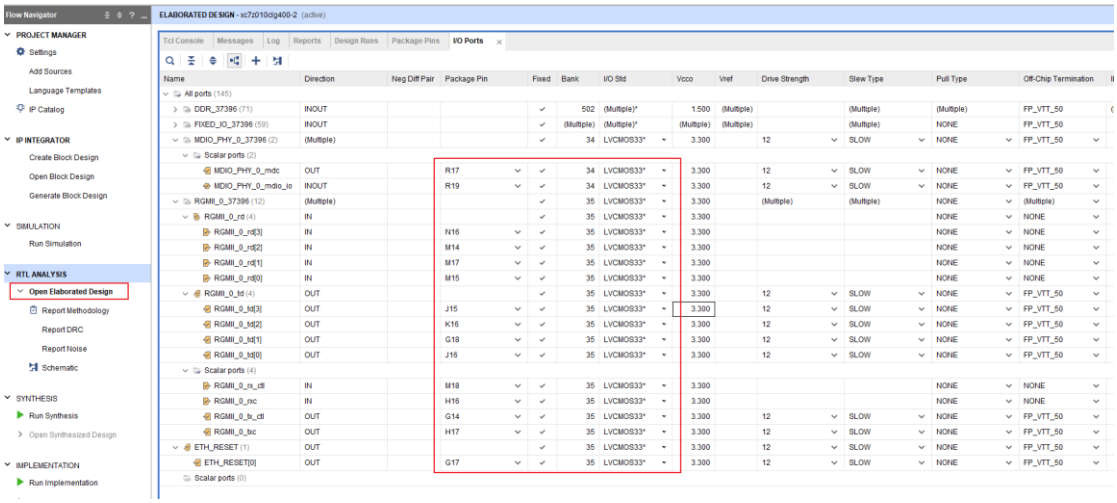


图 1-16 管脚分配

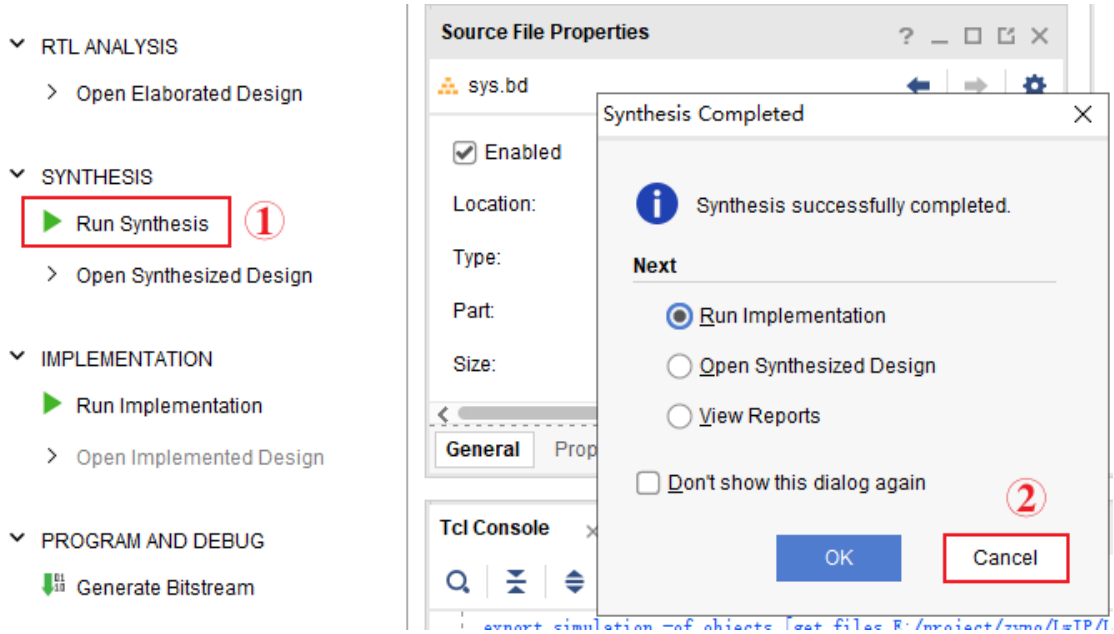
对应 xdc 约束如下：

```
set_property IOSTANDARD LVCMOS33 [get_ports MDIO_PHY_0_mdc]
set_property IOSTANDARD LVCMOS33 [get_ports MDIO_PHY_0_mdio_io]
set_property IOSTANDARD LVCMOS33 [get_ports {RGMII_0_rd[3]}]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports {RGMII_0_rd[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {RGMII_0_rd[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {RGMII_0_rd[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {RGMII_0_td[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {RGMII_0_td[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {RGMII_0_td[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {RGMII_0_td[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports RGMII_0_rx_ctl]
set_property IOSTANDARD LVCMOS33 [get_ports RGMII_0_rxc]
set_property IOSTANDARD LVCMOS33 [get_ports RGMII_0_tx_ctl]
set_property IOSTANDARD LVCMOS33 [get_ports RGMII_0_txc]
set_property PACKAGE_PIN R19 [get_ports MDIO_PHY_0_mdio_io]
set_property PACKAGE_PIN H17 [get_ports RGMII_0_txc]
set_property PACKAGE_PIN H16 [get_ports RGMII_0_rxc]
set_property PACKAGE_PIN G18 [get_ports {RGMII_0_td[1]}]
set_property PACKAGE_PIN N16 [get_ports {RGMII_0_rd[3]}]
set_property PACKAGE_PIN M14 [get_ports {RGMII_0_rd[2]}]
set_property PACKAGE_PIN M15 [get_ports {RGMII_0_rd[0]}]
set_property PACKAGE_PIN M17 [get_ports {RGMII_0_rd[1]}]
set_property PACKAGE_PIN M18 [get_ports RGMII_0_rx_ctl]
set_property PACKAGE_PIN K16 [get_ports {RGMII_0_td[2]}]
set_property PACKAGE_PIN J16 [get_ports {RGMII_0_td[0]}]
set_property PACKAGE_PIN J15 [get_ports {RGMII_0_td[3]}]
set_property PACKAGE_PIN G14 [get_ports RGMII_0_tx_ctl]
set_property PACKAGE_PIN R17 [get_ports MDIO_PHY_0_mdc]

set_property IOSTANDARD LVCMOS33 [get_ports {ETH_RESET[0]}]
set_property PACKAGE_PIN G17 [get_ports {ETH_RESET[0]}]
```

约束完成后，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 bit 文件与硬件描述文件一起导出，并运行 SDK，至此便完成了硬件逻辑系统的设计。

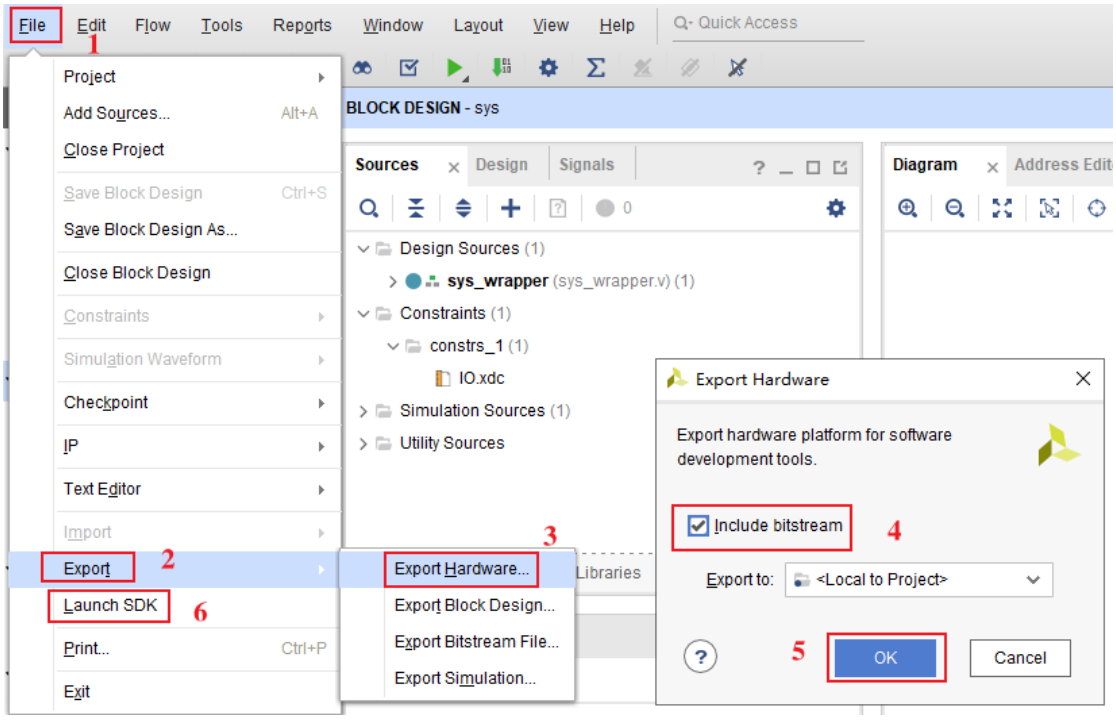


图 1-17 导出硬件描述文件

1.4 CPU 软件程序设计

1.4.1 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 1-18 所示：

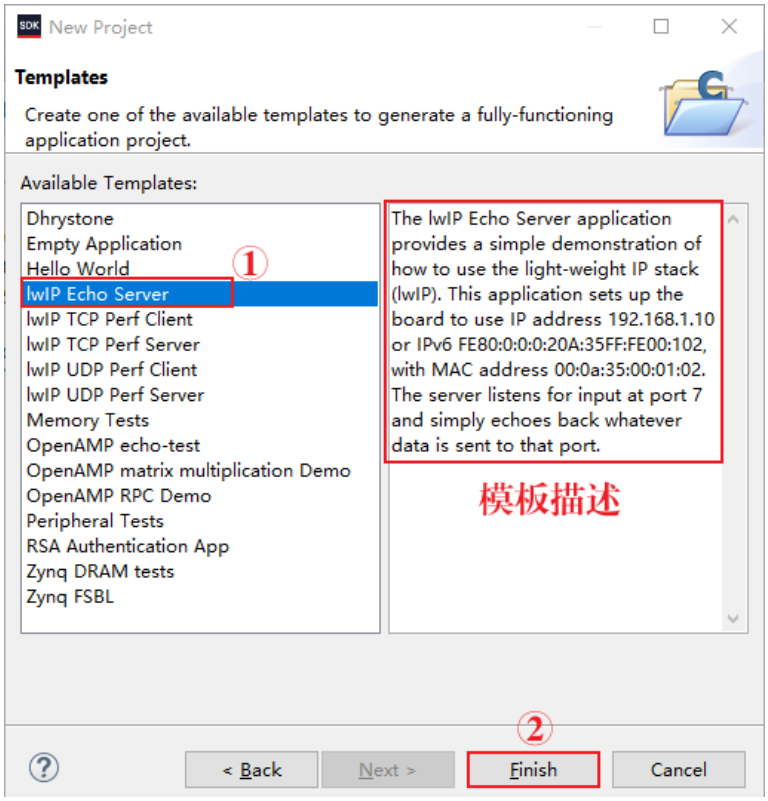


图 1-18 选择工程模板

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。

1.4.2 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：

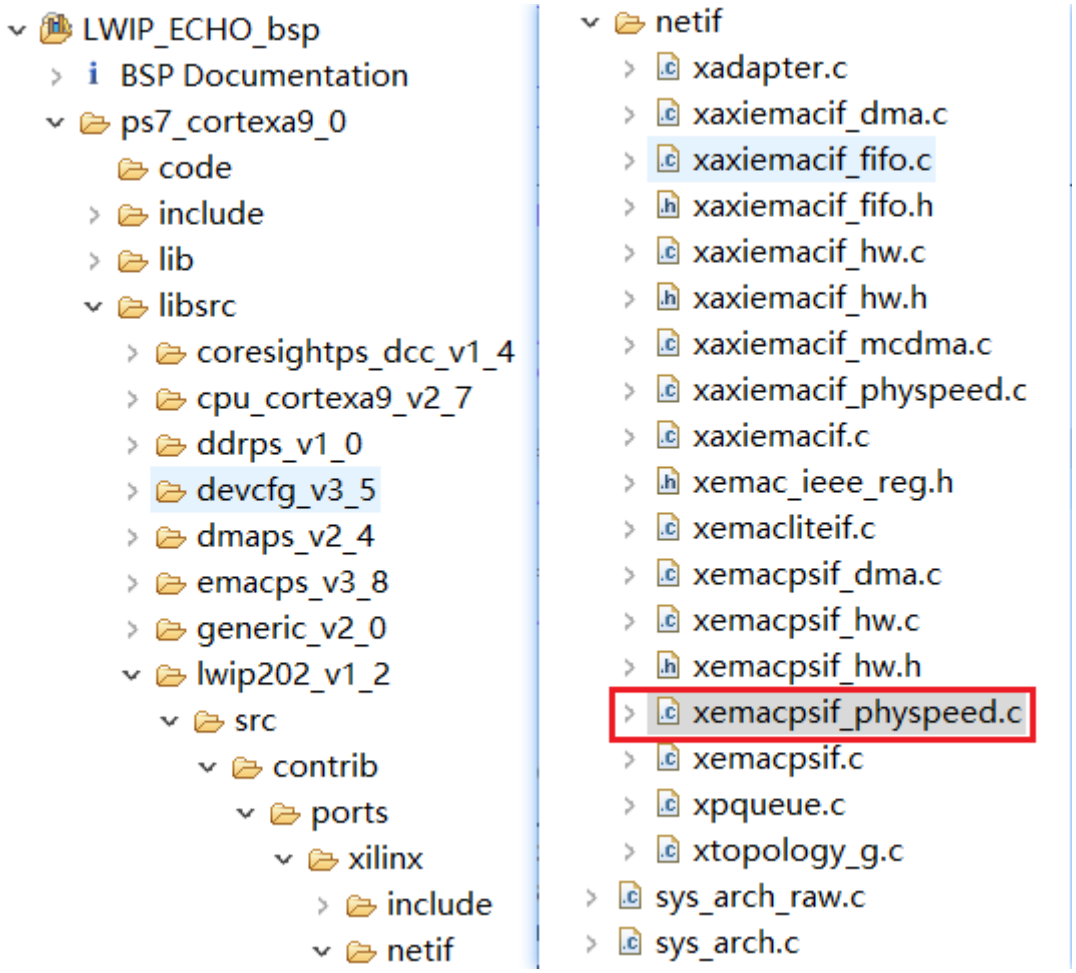


图 1-19 文件路径

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。

然后，在 626 行左右的“xil_printf("Start PHY autonegotiation \r\n");”语句下方添加以下内容：

```
    //开灯
#if 1
    XEmacPs_PhyWrite(xemacpsp, phy_addr, 0x1F, 0x0D08);
    XEmacPs_PhyWrite(xemacpsp, phy_addr, 0x11, 0x0009);
    XEmacPs_PhyWrite(xemacpsp, phy_addr, 0x1F, 0x0000);
#endif

#if 1
    //参考 uboot 启动
    XEmacPs_PhyWrite(xemacpsp, phy_addr, 0x1F, 0x0D04);
    XEmacPs_PhyWrite(xemacpsp, phy_addr, 0x10, 0x617F);
    XEmacPs_PhyWrite(xemacpsp, phy_addr, 0x1F, 0x0000);
#endif
```

该段语句仅在 PS 以太网控制器控制器 PL 网口时需要添加。需要注意的是，以上所有配置均是对 bsp 中的文件进行修改，一旦 bsp 重新编译，程序就会从官方源文件重新拷贝，并对文件进行覆盖。因此，要想修改永久有效，需要修改官方库函数的源文件，相关修改方法可参考总结内容。

最后，根据使用习惯，这里我们也可以修改开发板的 IP 地址、MAC 地址以及端口号。其中，MAC 地址和 IP 地址在 main.c 的 128~129 行和 208~216 行修改，修改为以下内容：

```
//MAC 地址
unsigned char mac_ethernet_address[] =
{ 0x00, 0x0a, 0x35, 0x01, 0xfe, 0xc0 };

//IP 地址
if (dhcp_timoutcntr <= 0) {
```

```
    if ((echo_netif->ip_addr.addr) == 0) {
        xil_printf("DHCP Timeout\r\n");
        xil_printf("Configuring default IP of 192.168.0.2\r\n");
        IP4_ADDR(&(echo_netif->ip_addr),  192, 168,   0, 2);
        IP4_ADDR(&(echo_netif->netmask), 255, 255, 255,  0);
        IP4_ADDR(&(echo_netif->gw),      192, 168,   0,  1);
    }
}
```

端口号则是在 echo.c 中修改，将第 100 行的 port 修改为 5000，如下：

```
unsigned port = 5000;
```

顺带地，这里我们也可以将 49 行中的调试打印信息一并修改，确保打印信息与我们实际配置一致，如下：

```
xil_printf("TCP packets sent to port 5000 will be echoed back\n\r");
```

这样，我们便将开发板的 MAC 地址设置为了 00_0a_35_01_fe_c0，IP 地址设置为了 192.168.0.2，监听端口号设置为了 5000。

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。

1.5 板级验证

1.5.1 所需硬件

1. BX71 开发板一个
2. 六类千兆网线一根
3. Type-C 数据线一根

1.5.2 硬件连接

硬件连接如图 1-20 所示：

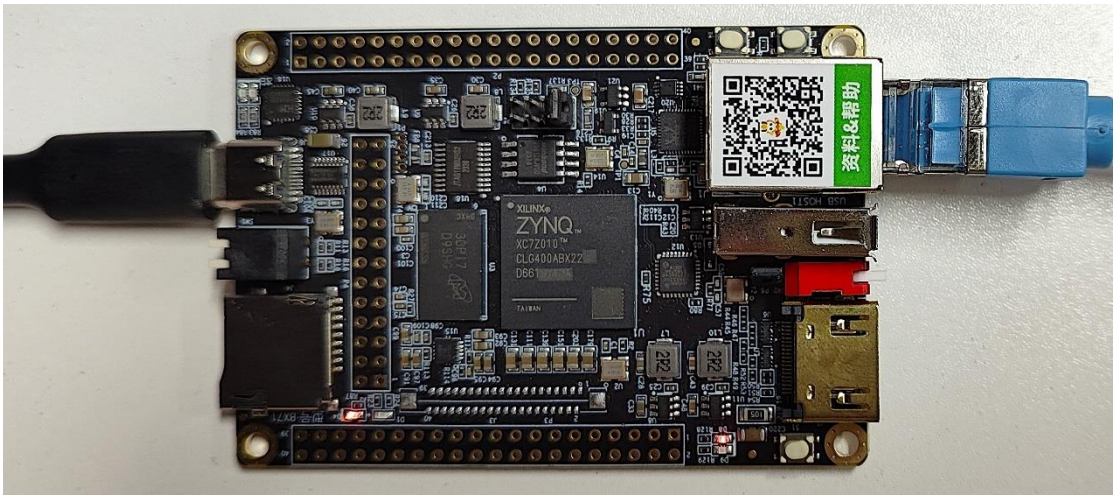


图 1-20 硬件连接图

设计仅需连接 JTAG 和以太网口，连接时需确保千兆网线一端连接在开发板 PL 以太网口，一端连接在电脑网口。

1.5.3 修改 IP 地址

连接好硬件且开发板上电后，接下来根据代码中的内容来设置电脑以太网 IP 地址，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

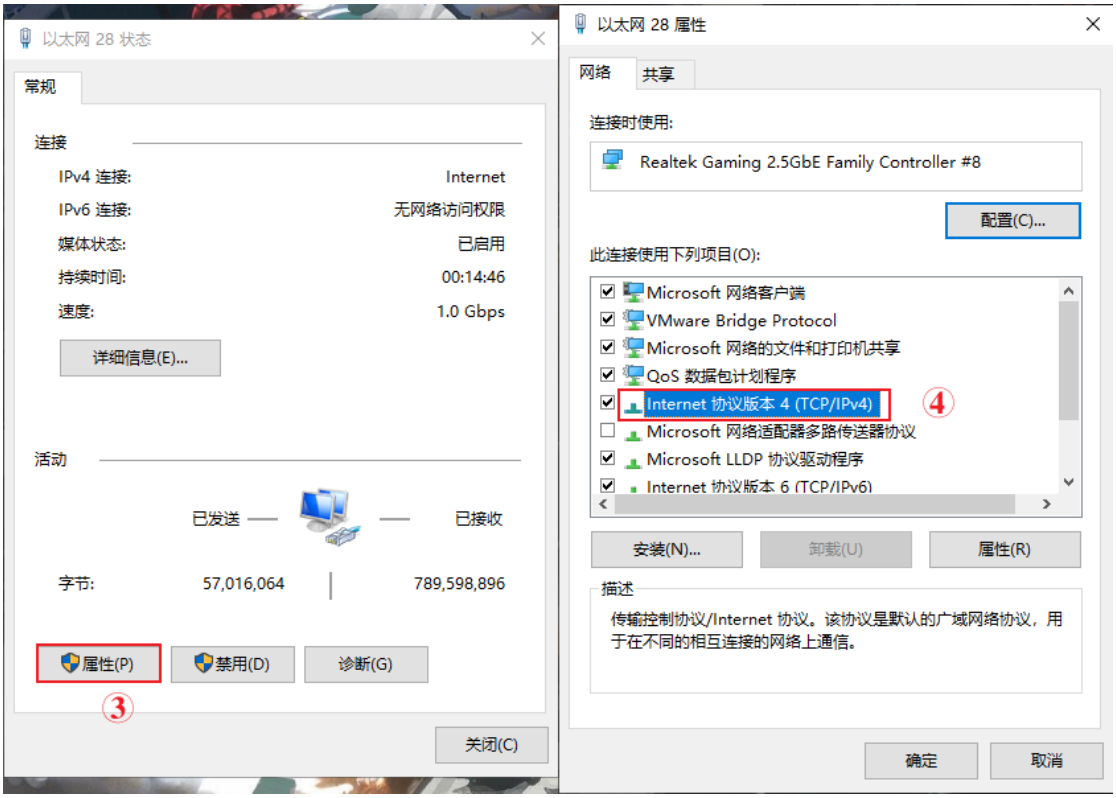


图 21　修改 IPv4 参数

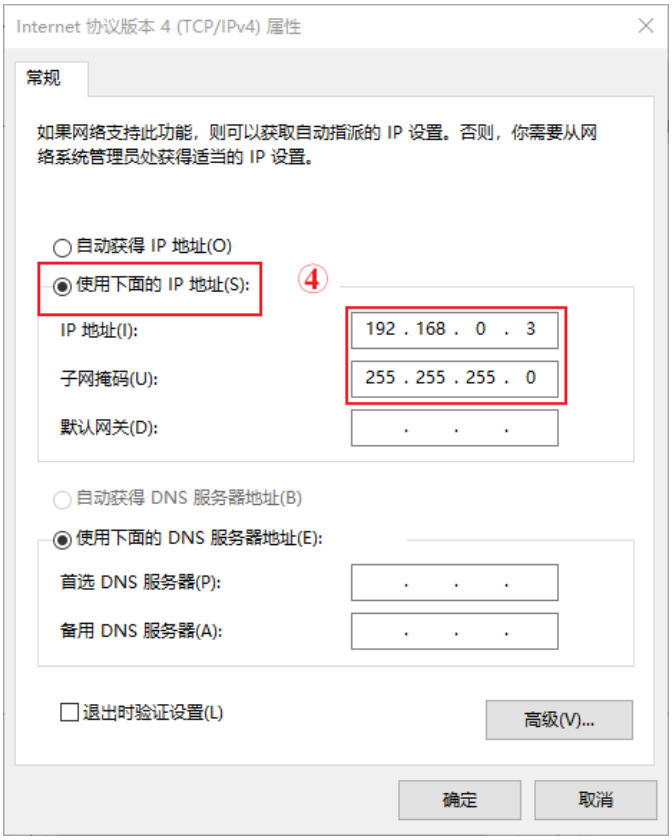


图 22　设置静态 IP

1.5.4 连接串口终端

修改好电脑 IP 地址后，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。笔者电脑端口号如图 1-23 所示：

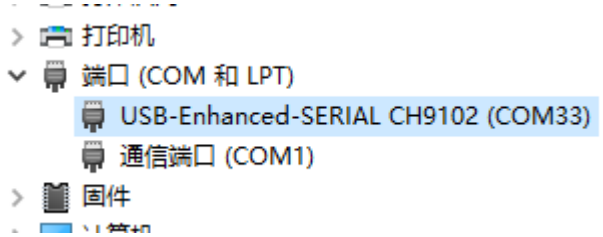


图 1-23 查询端口号

打开 SDK 终端，添加串口连接， 端口号选择为查询到的 COM33，如图 1-24 所示：

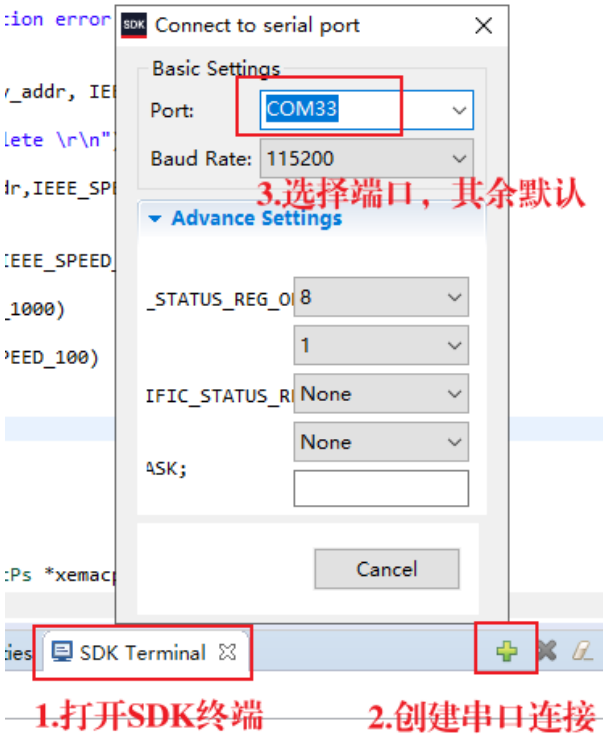


图 1-24 连接串口

点击 OK，连接完成后会出现图 1-25 所示连接成功字样：

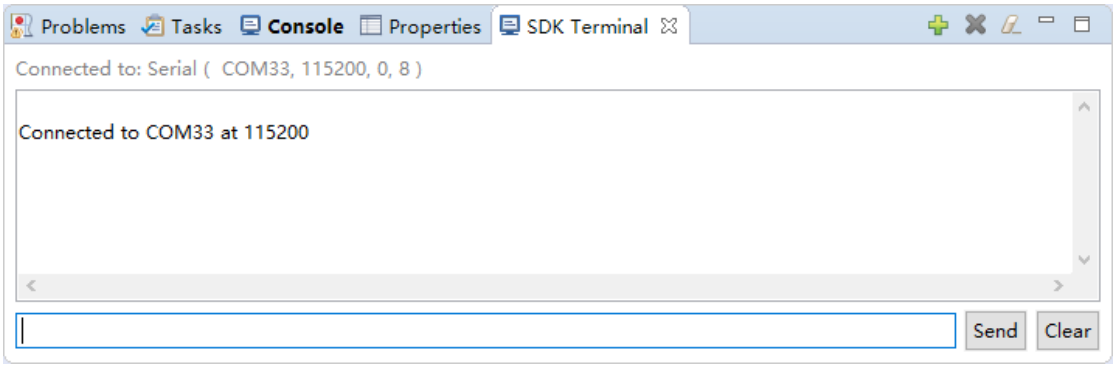


图 1-25 连接成功

连接成功后便可将设计烧录到开发板中进行验证了。

1.5.5 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 1-26 和图 1-27 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. 确认下方四个选项都被勾选，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

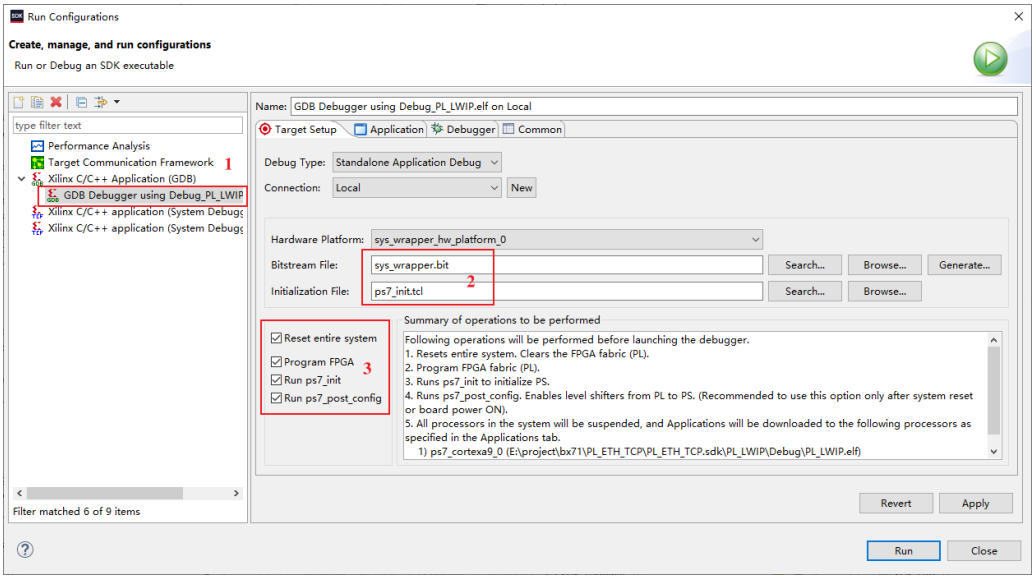


图 1-26 配置烧录任务

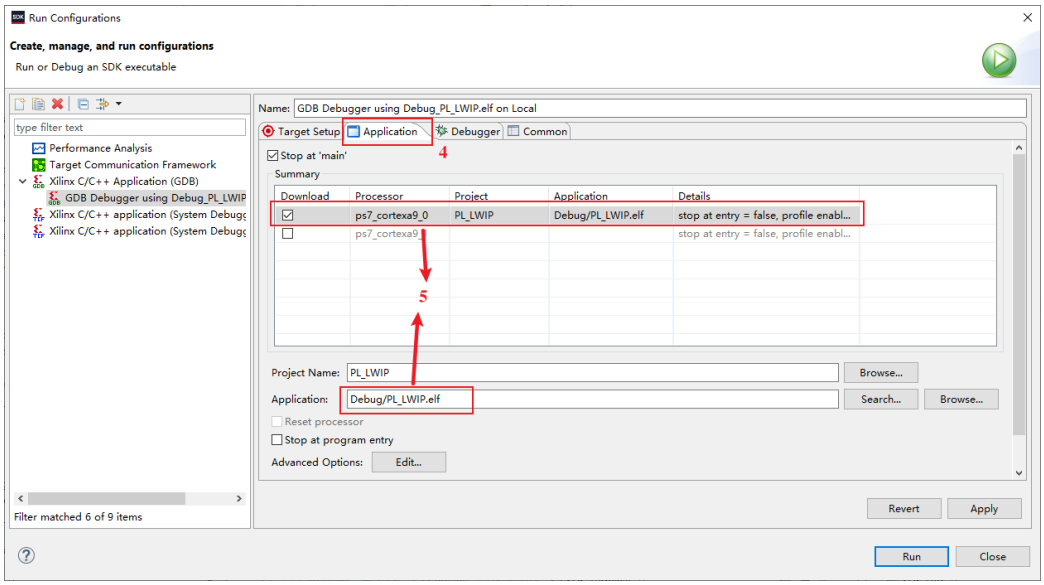


图 1-27 配置烧录任务

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 6102 will be echoed back

Start PHY autonegotiation
Waiting for PHY to complete autonegotiation.
autonegotiation complete
link speed for phy address 5: 1000
DHCP Timeout
Configuring default IP of 192.168.0.2
Board IP: 192.168.0.2

Netmask : 255.255.255.0

Gateway : 192.168.0.1

TCP echo server started @ port 6102
```

图 1-28 串口打印数据

可以看到，TCP 连接成功，此时的连接速度为 1000Mbps。打开网络调试软件 netassist，按照前面的配置信息进行连接，如图 1-29 所示：

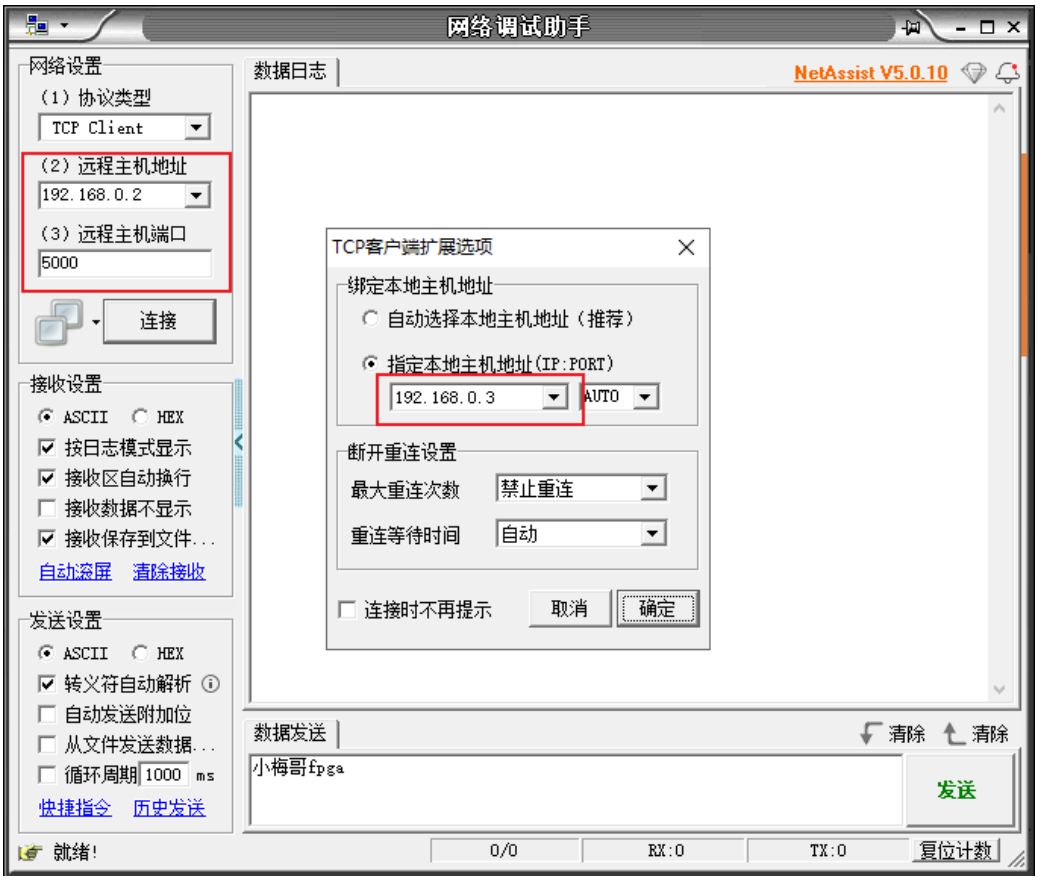


图 1-29 连接以太网调试助手

此时，发送数据进行测试，可以看到 TCP 回环成功。

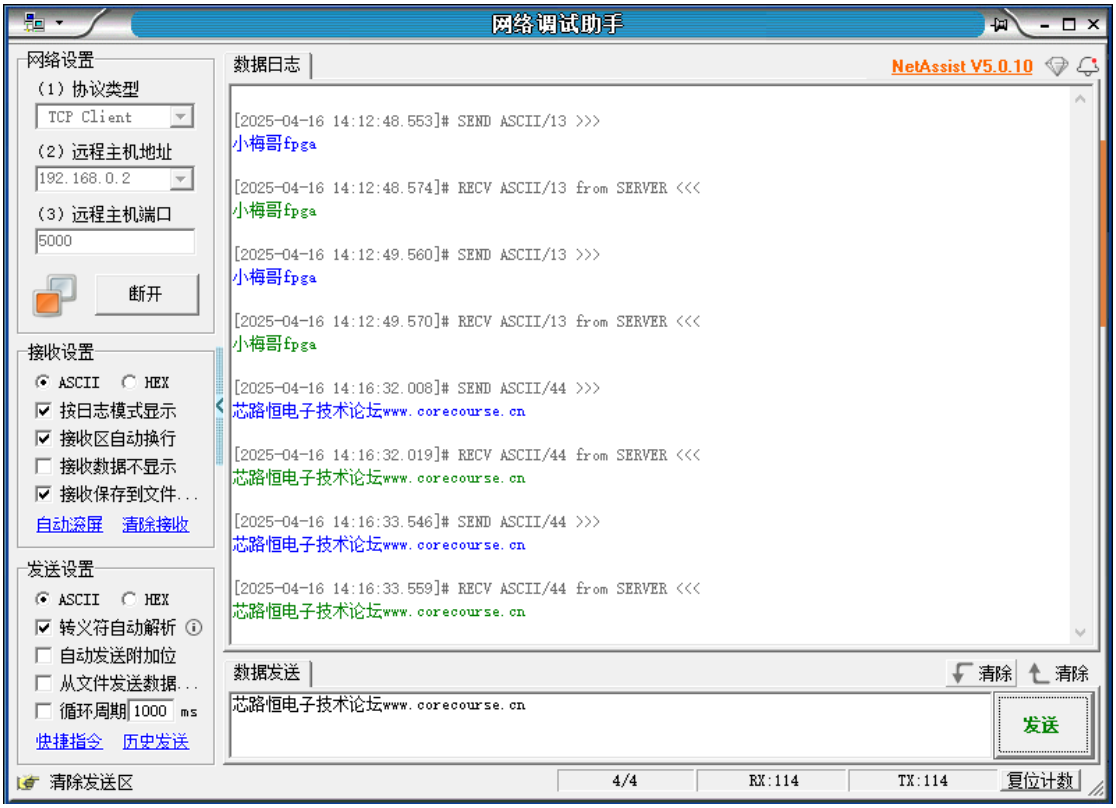


图 1-30 TCP 回环测试

至此，我们便通过 xilinx 官方模板实现了 TCP 回环系统的搭建。用户在使用时，可以以该系统框架为基础，通过添加一些自己的代码内容，来此实现基于 TCP 的相关应用。

1.6 总结

本节主要为大家介绍了如何通过修改官方 LWIP 模板例程，实现 BX71 开发板上 PS 端以太网控制器，控制器 PL 网口实现 TCP 回环功能。并且通过一系列验证，验证了修改后模板例程的可行性。

该方案流程不仅适用于BX71开发板外，同样适用于其他ZYNQ系列器件，但需要注意的是，该流程仅针对于 PS 端以太网控制器实现 PL 网口的 TCP 回环功能。如果是 PS 端网口实现 TCP 功能，则流程上要更为简单。

最后，这里再为大家补充两个可能遇到的问题：

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时，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\embeddedsw\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。