Part 1: Document introduction

This article describes the AX7021 FPGA development board, Multiple Ethernet for the FL9031 Ethernet Module in the SDK test lwIP Echo Server function, has petalinux 2017.4 driver configuration, device tree configuration, and simple application

This example requires Vivado 2017.4, Petalinux 2017.4, other versions may have unpredictable problems, please solve it yourself.

Part 2: VIVADO hardware Project Establishment

How to use VIVADO to build a project is not the focus of this article. ALINX provides the already completed vivado project.

2.1: Export SDK

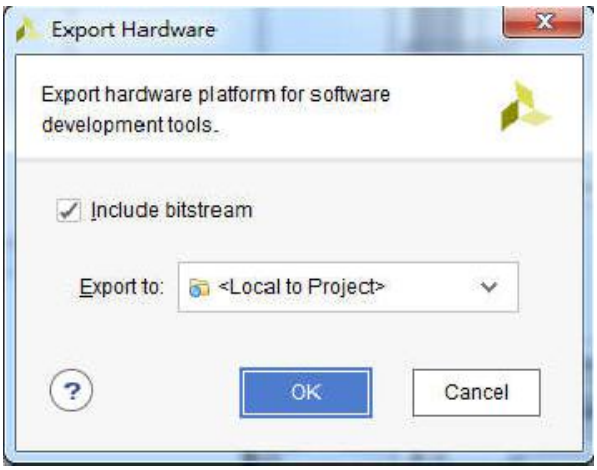


Figure 2-1: Export SDK

Part 3: lwIP Echo Server test

This section describes how to test the Ethernet on the PL and PS side

under sdk and build a simple test using the SDK's own lwIP Echo Server template.

3.1 LWIP library modification

Since the built-in LWIP library can only identify part of the phy chip, if the phy chip used by the development board is not within the default support, modify the library file. You can also replace the original library directly with the modified library.

1) Find the library file directory

"X:\Xilinx\SDK\2017.4\data\embeddedsw\ThirdParty\sw_services"

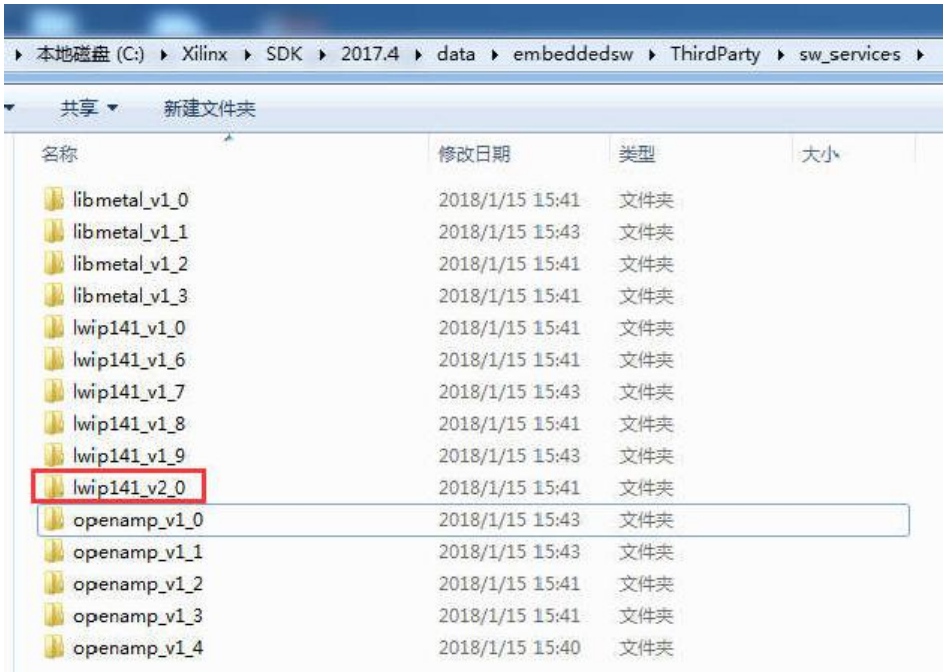


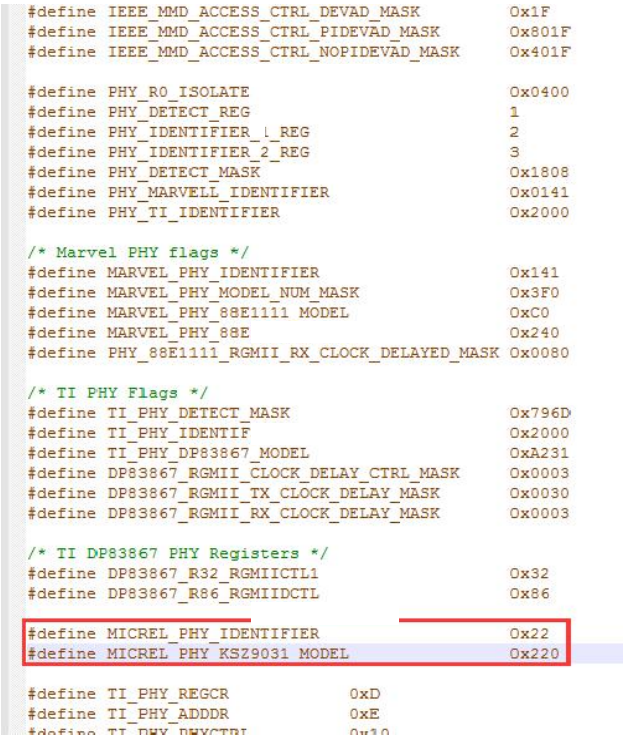
Figure 3-1: Find the library file directory

2) Locate the files "xaxiemacif_physpeed.c" and "xemacpsif_physpeed.c" in the file directory "lwip141_v2_0\src\contrib\ports\xilinx\netif" to be modified. **The v2_0 version is modified here. If it is another version of Vivado, the version of the lwip library will also change.**

Generally, the latest version is selected to be modified.

Figure 3-2: Find the file

3) Modify the "xaxiemacif_physpeed.c" file to add related macro definitions



```
#define IEEE_MMD_ACCESS_CTRL_DEVAD_MASK          0x1F
#define IEEE_MMD_ACCESS_CTRL_PIDEVAD_MASK        0x801F
#define IEEE_MMD_ACCESS_CTRL_NOPIDEVAD_MASK      0x401F

#define PHY_R0_ISOLATE                           0x0400
#define PHY_DETECT_REG                           1
#define PHY_IDENTIFIER_1_REG                     2
#define PHY_IDENTIFIER_2_REG                     3
#define PHY_DETECT_MASK                          0x1808
#define PHY_MARVELL_IDENTIFIER                   0x0141
#define PHY_TI_IDENTIFIER                        0x2000

/* Marvel PHY flags */
#define MARVEL_PHY_IDENTIFIER                    0x141
#define MARVEL_PHY_MODEL_NUM_MASK                0x3F0
#define MARVEL_PHY_88E1111_MODEL                 0xC0
#define MARVEL_PHY_88E                           0x240
#define PHY_88E1111_RGMII_RX_CLOCK_DELAYED_MASK  0x0080

/* TI PHY Flags */
#define TI_PHY_DETECT_MASK                       0x796D
#define TI_PHY_IDENTIF                           0x2000
#define TI_PHY_DP83867_MODEL                     0xA231
#define DP83867_RGMII_CLOCK_DELAY_CTRL_MASK      0x0003
#define DP83867_RGMII_TX_CLOCK_DELAY_MASK        0x0030
#define DP83867_RGMII_RX_CLOCK_DELAY_MASK        0x0003

/* TI DP83867 PHY Registers */
#define DP83867_R32_RGMIICTL1                    0x32
#define DP83867_R86_RGMIIDCTL                    0x86

#define MICREL_PHY_IDENTIFIER                    0x22
#define MICREL_PHY_KSZ9031_MODEL                 0x220

#define TI_PHY_REGCR            0xD
#define TI_PHY_ADDDR            0xE
```

Figure 3-3: Add Related Macro Definitions

4) Add phy get function

```
unsigned int get_phy_speed_ksz9031(XAxiEthernet *xaxiemacp, u32 phy_addr)
{
        u16 control;

        u16 status;

        u16 partner_capabilities;
        xil_printf("Start PHY autonegotiation \r\n");

        XAxiEthernet_PhyWrite(xaxiemacp,phy_addr, IEEE_PAGE_ADDRESS_REGISTER, 2);

        XAxiEthernet_PhyRead(xaxiemacp, phy_addr, IEEE_CONTROL_REG_MAC, &control);

        //control |= IEEE_RGMII_TXRX_CLOCK_DELAYED_MASK;

        control &= ~(0x10);

        XAxiEthernet_PhyWrite(xaxiemacp, phy_addr, IEEE_CONTROL_REG_MAC, control);

        XAxiEthernet_PhyWrite(xaxiemacp, phy_addr, IEEE_PAGE_ADDRESS_REGISTER, 0);

        XAxiEthernet_PhyRead(xaxiemacp, phy_addr, IEEE_AUTONEGO_ADVERTISE_REG, &control);
        control |= IEEE_ASYMMETRIC_PAUSE_MASK;

        control |= IEEE_PAUSE_MASK;

        control |= ADVERTISE_100;

        control |= ADVERTISE_10;

        XAxiEthernet_PhyWrite(xaxiemacp, phy_addr, IEEE_AUTONEGO_ADVERTISE_REG, control);

        XAxiEthernet_PhyRead(xaxiemacp, phy_addr, IEEE_1000_ADVERTISE_REG_OFFSET,

                                        &control);
        control |= ADVERTISE_1000;
        XAxiEthernet_PhyWrite(xaxiemacp, phy_addr, IEEE_1000_ADVERTISE_REG_OFFSET,

                                        control);

        XAxiEthernet_PhyWrite(xaxiemacp, phy_addr, IEEE_PAGE_ADDRESS_REGISTER, 0);

        XAxiEthernet_PhyRead(xaxiemacp, phy_addr, IEEE_COPPER_SPECIFIC_CONTROL_REG,
```

```
                                &control);
control |= (7 << 12);   /* max number of gigabit attempts */
control |= (1 << 11);   /* enable downshift */
XAxiEthernet_PhyWrite(xaxiemacp, phy_addr, IEEE_COPPER_SPECIFIC_CONTROL_REG,

                                control);
XAxiEthernet_PhyRead(xaxiemacp, phy_addr, IEEE_CONTROL_REG_OFFSET, &control);
control |= IEEE_CTRL_AUTONEGOTIATE_ENABLE;
control |= IEEE_STAT_AUTONEGOTIATE_RESTART;


XAxiEthernet_PhyWrite(xaxiemacp, phy_addr, IEEE_CONTROL_REG_OFFSET, control);


XAxiEthernet_PhyRead(xaxiemacp, phy_addr, IEEE_CONTROL_REG_OFFSET, &control);
control |= IEEE_CTRL_RESET_MASK;
XAxiEthernet_PhyWrite(xaxiemacp, phy_addr, IEEE_CONTROL_REG_OFFSET, control);


while (1) {
        XAxiEthernet_PhyRead(xaxiemacp, phy_addr, IEEE_CONTROL_REG_OFFSET, &control);
        if (control & IEEE_CTRL_RESET_MASK)
                continue;
        else
                break;
}
xil_printf("Waiting for PHY to complete autonegotiation.\r\n");


XAxiEthernet_PhyRead(xaxiemacp, phy_addr, IEEE_STATUS_REG_OFFSET, &status);
while ( !(status & IEEE_STAT_AUTONEGOTIATE_COMPLETE) ) {
        sleep(1);
        XAxiEthernet_PhyRead(xaxiemacp, phy_addr, IEEE_STATUS_REG_OFFSET,

                                &status);
        }


xil_printf("autonegotiation complete \r\n");
```

```
        XAxiEthernet_PhyRead(xaxiemacp, phy_addr, 0x1f, &partner_capabilities);


        if ( (partner_capabilities & 0x40) == 0x40)/* 1000Mbps */
                return 1000;
        else if ( (partner_capabilities & 0x20) == 0x20)/* 100Mbps */
                return 100;
        else if ( (partner_capabilities & 0x10) == 0x10)/* 10Mbps */
                return 10;
        else
                return 0;
}
```

5) Modify the function "get_IEEE_phy_speed" to add support for KSZ9031

```
unsigned get_IEEE_phy_speed(XAxiEthernet *xaxiemacp)
{
        u16 phy_identifier;
        u16 phy_model;
        u8 phytype;

#ifdef XPAR_AXIETHERNET_0_BASEADDR
        u32 phy_addr = detect_phy(xaxiemacp);

        /* Get the PHY Identifier and Model number */
        XAxiEthernet_PhyRead(xaxiemacp, phy_addr, PHY_IDENTIFIER_1_REG, &phy_identifier);
        XAxiEthernet_PhyRead(xaxiemacp, phy_addr, PHY_IDENTIFIER_2_REG, &phy_model);

/* Depending upon what manufacturer PHY is connected, a different mask is
 * needed to determine the specific model number of the PHY. */
        if (phy_identifier == MARVEL_PHY_IDENTIFIER) {
                phy_model = phy_model & MARVEL_PHY_MODEL_NUM_MASK;
```

```
                if (phy_model == MARVEL_PHY_88E1116R_MODEL) {
                        return get_phy_speed_88E1116R(xaxiemacp, phy_addr);
                } else if (phy_model == MARVEL_PHY_88E1111_MODEL)
                        { return get_phy_speed_88E1111(xaxiemacp,
                        phy_addr);
                }
        } else if (phy_identifier == TI_PHY_IDENTIFIER) {
                phy_model = phy_model & TI_PHY_DP83867_MODEL;
                phytype = XAxiEthernet_GetPhysicalInterface(xaxiemacp);

                if (phy_model == TI_PHY_DP83867_MODEL && phytype == XAE_PHY_TYPE_SGMII)
                        { return get_phy_speed_TI_DP83867_SGMII(xaxiemacp, phy_addr);
                }

                if (phy_model == TI_PHY_DP83867_MODEL) {
                        return get_phy_speed_TI_DP83867(xaxiemacp, phy_addr);
                }
        }
        else if(phy_identifier == MICREL_PHY_IDENTIFIER)
        {
                xil_printf("Phy %d is KSZ9031\n\r",phy_addr);
                get_phy_speed_ksz9031(xaxiemacp, phy_addr);
        }
        else {
            LWIP_DEBUGF(NETIF_DEBUG, ("XAxiEthernet get_IEEE_phy_speed: Detected PHY with
unknown identifier/model.\r\n"));
        }
#endif
#ifdef PCM_PMA_CORE_PRESENT
        return get_phy_negotiated_speed(xaxiemacp, phy_addr);
#endif
}
```

6) Modify the "xemacpsif_physpeed.c" file to add a macro definition

```
#define IEEE_PAGE_ADDRESS_REGISTER          22
#define IEEE_CTRL_1GBPS_LINKSPEED_MASK      0x2040
#define IEEE_CTRL_LINKSPEED_MASK            0x0040
#define IEEE_CTRL_LINKSPEED_1000M           0x0040
#define IEEE_CTRL_LINKSPEED_100M            0x2000
#define IEEE_CTRL_LINKSPEED_10M             0x0000
#define IEEE_CTRL_RESET_MASK                0x8000

#define IEEE_SPEED_MASK     0xC000
#define IEEE_SPEED_1000     0x8000
#define IEEE_SPEED_100      0x4000

#define IEEE_CTRL_RESET_MASK                0x8000
#define IEEE_CTRL_AUTONEGOTIATE_ENABLE      0x1000
#define IEEE_STAT_AUTONEGOTIATE_COMPLETE    0x0020
#define IEEE_STAT_AUTONEGOTIATE_RESTART     0x0200
#define IEEE_RGMII_TXRX_CLOCK_DELAYED_MASK  0x0030
#define IEEE_ASYMMETRIC_PAUSE_MASK          0x0800
#define IEEE_PAUSE_MASK                     0x0400
#define IEEE_AUTONEG_ERROR_MASK             0x8000

#define PHY_DETECT_REG                          1
#define PHY_IDENTIFIER_1_REG                    2
#define PHY_IDENTIFIER_2_REG                    3
#define PHY_DETECT_MASK                     0x1808
#define PHY_MARVELL_IDENTIFIER              0x0141
#define PHY_TI_IDENTIFIER                   0x2000
#define PHY_XILINX_PCS_PMA_ID1          0x0174
#define PHY_XILINX_PCS_PMA_ID2          0x0C00

#define MICREL_PHY_IDENTIFIER                   0x22
#define MICREL_PHY_KSZ9031_MODEL                0x220

#define XEMACPS_GMII2RGMII_SPEED1000_FD     0x140
#define XEMACPS_GMII2RGMII_SPEED100_FD      0x2100
#define XEMACPS_GMII2RGMII_SPEED10_FD       0x100
#define XEMACPS_GMII2RGMII_REG_NUM          0x10
```

Figure 3-4: Add a macro definition

7) Add phy speed get function

```
static u32_t get_phy_speed_ksz9031(XEmacPs *xemacpsp, u32_t phy_addr)
{
        u16_t temp;

        u16_t control;

        u16_t status;

        u16_t status_speed;
        u32_t timeout_counter = 0;

        u32_t temp_speed;

        u32_t phyregtemp;

        xil_printf("Start PHY autonegotiation \r\n");
```

```
XEmacPs_PhyWrite(xemacpsp,phy_addr, IEEE_PAGE_ADDRESS_REGISTER, 2);
XEmacPs_PhyRead(xemacpsp, phy_addr, IEEE_CONTROL_REG_MAC, &control);
control |= IEEE_RGMII_TXRX_CLOCK_DELAYED_MASK;
XEmacPs_PhyWrite(xemacpsp, phy_addr, IEEE_CONTROL_REG_MAC, control);

XEmacPs_PhyWrite(xemacpsp, phy_addr, IEEE_PAGE_ADDRESS_REGISTER, 0);

XEmacPs_PhyRead(xemacpsp, phy_addr, IEEE_AUTONEGO_ADVERTISE_REG, &control);
control |= IEEE_ASYMMETRIC_PAUSE_MASK;
control |= IEEE_PAUSE_MASK;
control |= ADVERTISE_100;
control |= ADVERTISE_10;
XEmacPs_PhyWrite(xemacpsp, phy_addr, IEEE_AUTONEGO_ADVERTISE_REG, control);

XEmacPs_PhyRead(xemacpsp, phy_addr, IEEE_1000_ADVERTISE_REG_OFFSET,
                                &control);
control |= ADVERTISE_1000;
XEmacPs_PhyWrite(xemacpsp, phy_addr, IEEE_1000_ADVERTISE_REG_OFFSET,
                                control);

XEmacPs_PhyWrite(xemacpsp, phy_addr, IEEE_PAGE_ADDRESS_REGISTER, 0);
XEmacPs_PhyRead(xemacpsp, phy_addr, IEEE_COPPER_SPECIFIC_CONTROL_REG,

                                &control);
control |= (7 << 12);   /* max number of gigabit attempts */
control |= (1 << 11);   /* enable downshift */
XEmacPs_PhyWrite(xemacpsp, phy_addr, IEEE_COPPER_SPECIFIC_CONTROL_REG,

                                control);
XEmacPs_PhyRead(xemacpsp, phy_addr, IEEE_CONTROL_REG_OFFSET, &control);
control |= IEEE_CTRL_AUTONEGOTIATE_ENABLE;
control |= IEEE_STAT_AUTONEGOTIATE_RESTART;
XEmacPs_PhyWrite(xemacpsp, phy_addr, IEEE_CONTROL_REG_OFFSET, control);

XEmacPs_PhyRead(xemacpsp, phy_addr, IEEE_CONTROL_REG_OFFSET, &control);
```

```
control |= IEEE_CTRL_RESET_MASK;
XEmacPs_PhyWrite(xemacpsp, phy_addr, IEEE_CONTROL_REG_OFFSET, control);

while (1) {
        XEmacPs_PhyRead(xemacpsp, phy_addr, IEEE_CONTROL_REG_OFFSET, &control);
        if (control & IEEE_CTRL_RESET_MASK)
                continue;
        else
                break;
}

XEmacPs_PhyRead(xemacpsp, phy_addr, IEEE_STATUS_REG_OFFSET, &status);

xil_printf("Waiting for PHY to complete autonegotiation.\r\n");

while ( !(status & IEEE_STAT_AUTONEGOTIATE_COMPLETE) ) {
        sleep(1);
        XEmacPs_PhyRead(xemacpsp, phy_addr,
                                        IEEE_COPPER_SPECIFIC_STATUS_REG_2,  &temp);
        timeout_counter++;

        if (timeout_counter == 30) {
                xil_printf("Auto negotiation error \r\n");
                return;
        }
        XEmacPs_PhyRead(xemacpsp, phy_addr, IEEE_STATUS_REG_OFFSET, &status);
}
xil_printf("autonegotiation complete \r\n");

XEmacPs_PhyRead(xemacpsp, phy_addr,0x1f,
                                &status_speed);

if ( (status_speed & 0x40) == 0x40)/* 1000Mbps */
        return 1000;
```

```
        else if ( (status_speed & 0x20) == 0x20)/* 100Mbps */
                return 100;
        else if ( (status_speed & 0x10) == 0x10)/* 10Mbps */
                return 10;
        else
                return 0;
        return XST_SUCCESS;
}
```

8) Modify the function "get_IEEE_phy_speed" to add support for KSZ9031

```
static u32_t get_IEEE_phy_speed(XEmacPs *xemacpsp, u32_t phy_addr)
{
        u16_t phy_identity;
        u32_t RetStatus;

        XEmacPs_PhyRead(xemacpsp, phy_addr, PHY_IDENTIFIER_1_REG,
                                        &phy_identity);

        if(phy_identity == MICREL_PHY_IDENTIFIER)
        {
                RetStatus = get_phy_speed_ksz9031(xemacpsp, phy_addr);
        }
        else if (phy_identity == PHY_TI_IDENTIFIER) {
                RetStatus = get_TI_phy_speed(xemacpsp, phy_addr);
        } else {
                RetStatus = get_Marvell_phy_speed(xemacpsp, phy_addr);
        }

        return RetStatus;
}
```

3.2 Create an APP based on the LWIP template

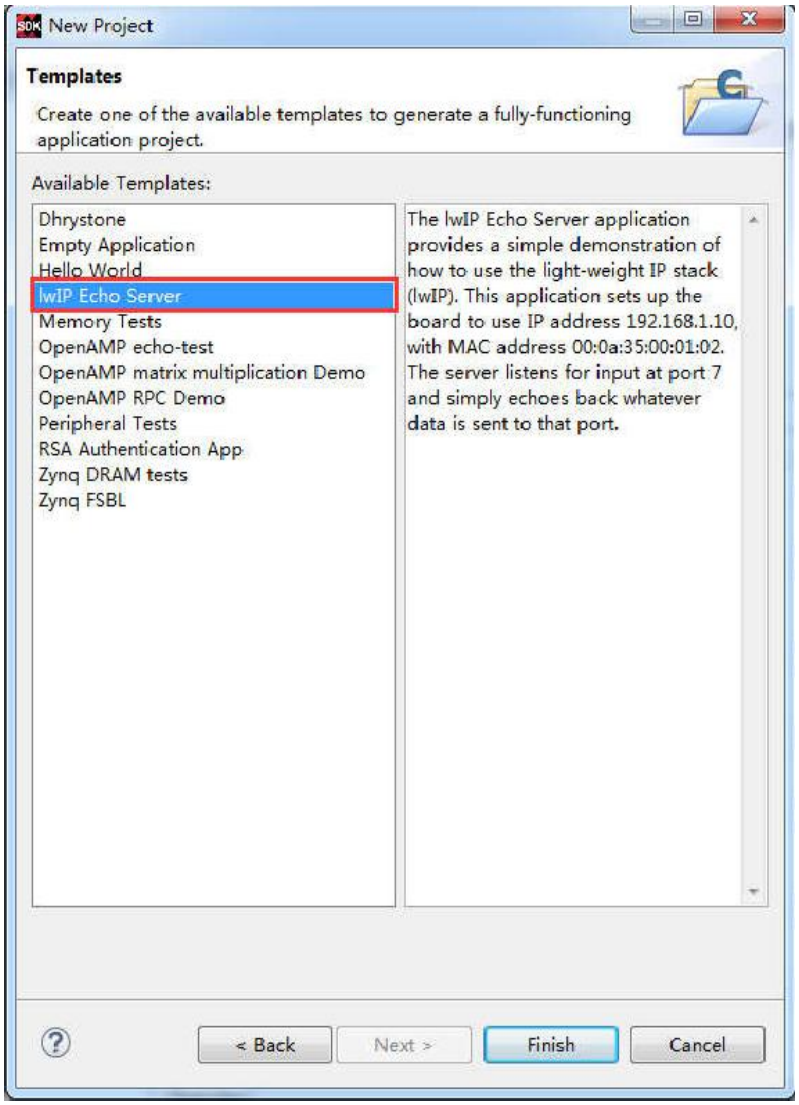


Figure 3-5: Create an APP based on the LWIP template

3.3 Download debugging

If the system has both a PS Ethernet controller and a PL AXI Ethernet controller, the LWIP template will select the PL AXI Ethernet controller by default. We first test the PL port Ethernet. The test environment requires a router that supports "dhcp". The FPGA development board connects to the router to

automatically obtain the IP address. The experiment host and the FPGA development board are in a network and can communicate with each other.

3.4 PL port Ethernet Test

- Connect the serial port to open the serial debugging terminal, connect the PL Ethernet cable to the router (ETH1), and only use ETH1 in the Vivado project, so other ports are not available.
- Run the SDK

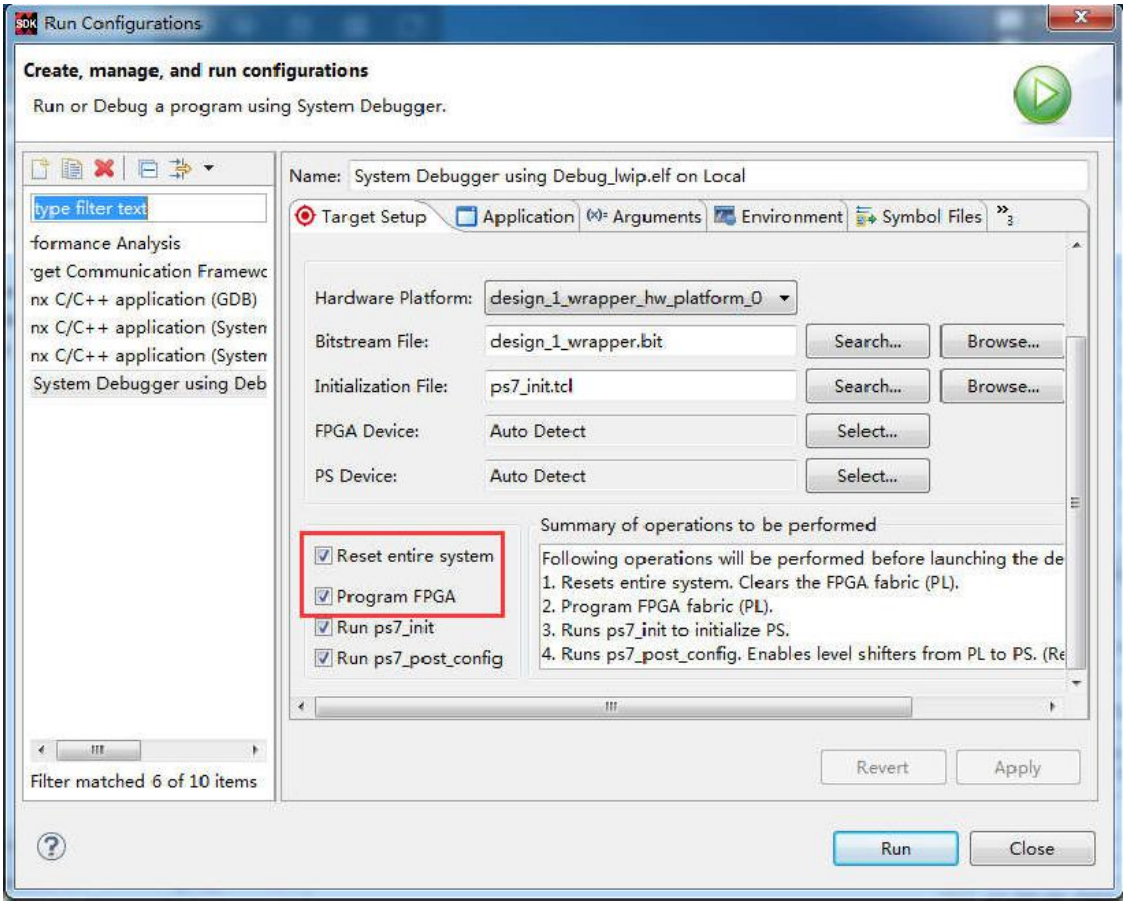


Figure 3-6: Run Configuration

You can see that the serial port prints some information, you can see that the address is automatically obtained as "192.168.1.68", the connection speed is 1000Mbps, and the tcp port is 7

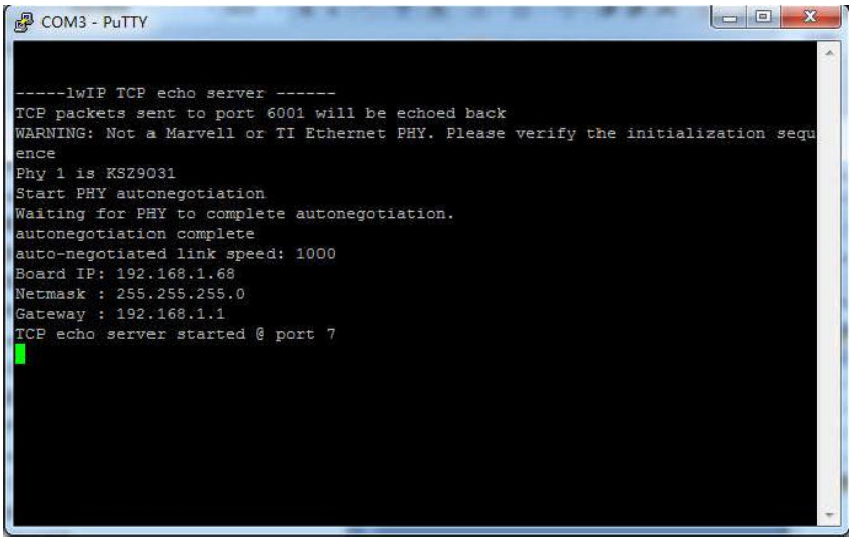


Figure 3-7: Board IP

Connect using telnet

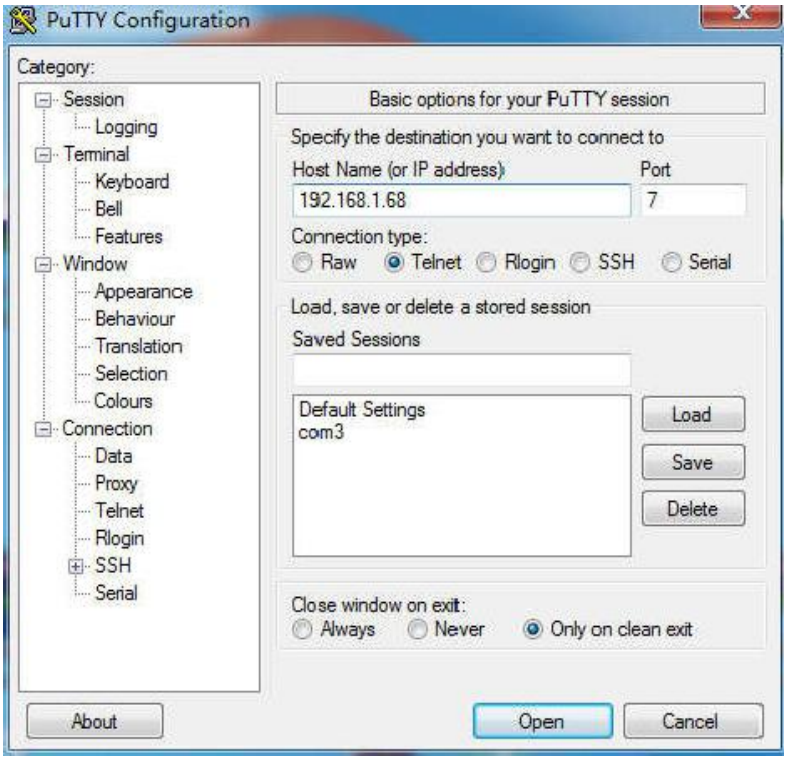


Figure 3-8: Board IP

The development board returns the same character when entering a character

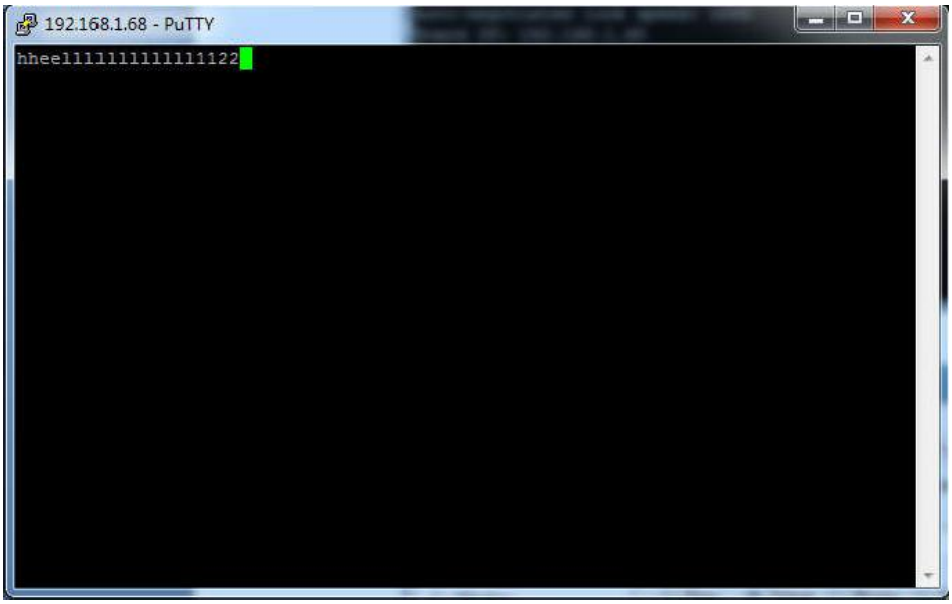


Figure 3-9: Returns the same character when entering a character

3.4.1： Test other channels

By default, the program uses the first network port of the PL end, and can test other channels by modifying the file platform_config.h.

```
#define PLATFORM_EMAC_BASEADDR XPAR_AXI_ETHERNET_0_BASEADDR Test channel 1
#define PLATFORM_EMAC_BASEADDR XPAR_AXI_ETHERNET_1_BASEADDR Test channel 2

#define PLATFORM_EMAC_BASEADDR XPAR_AXI_ETHERNET_2_BASEADDR Test channel 3
#define PLATFORM_EMAC_BASEADDR XPAR_AXI_ETHERNET_3_BASEADDR Test channel 4
```

Due to the LWIP library itself, it is not possible to test multiple channels at the same time. If you use a Linux system, you can easily use multiple channels.

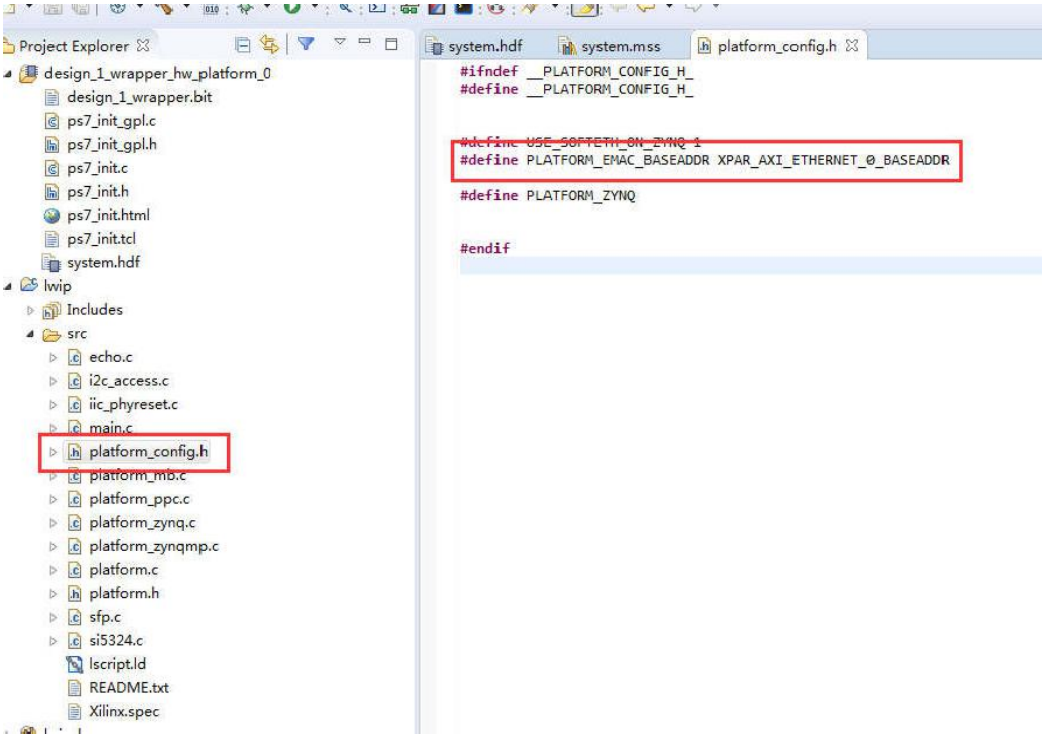


Figure 3-10: Test other channels by modifying the file platform_config.h

3.5 PS port Ethernet Test

- Modify BSP settings

Figure 3-11: Modify BSP settings

- "use_axieth_on_zynq" is changed to 0, using PS Ethernet

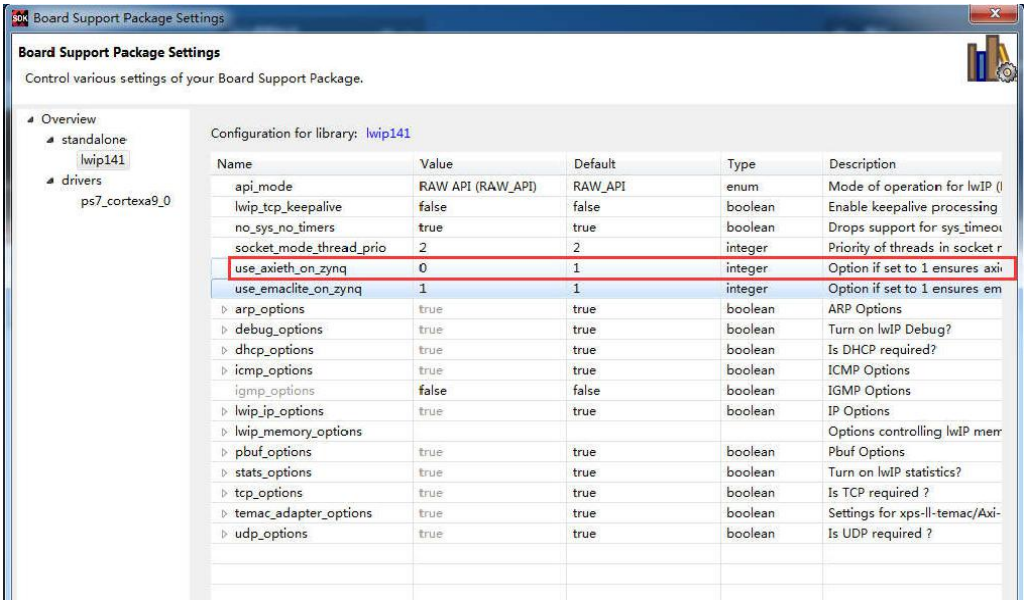


Figure 3-12: "use_axieth_on_zynq" is changed to 0

- Modify the "platform_config.h" file

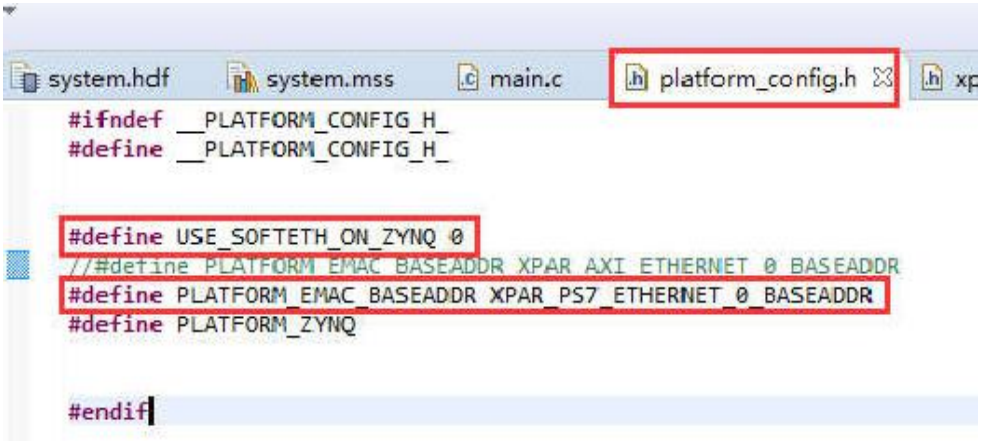


Figure 3-13: Modify the "platform_config.h" file

- Network cable connects PS end Ethernet to router
- Run the program and observe the serial output

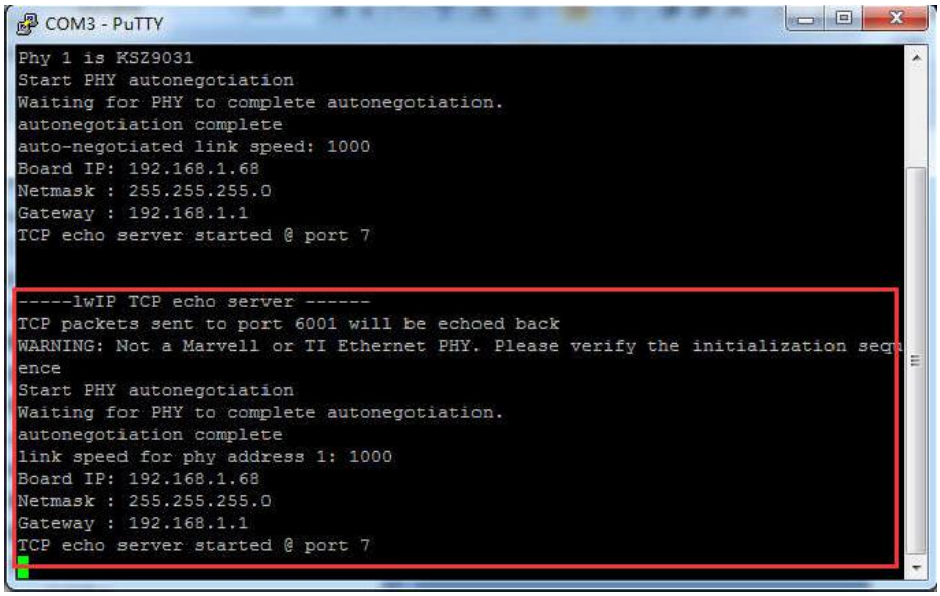


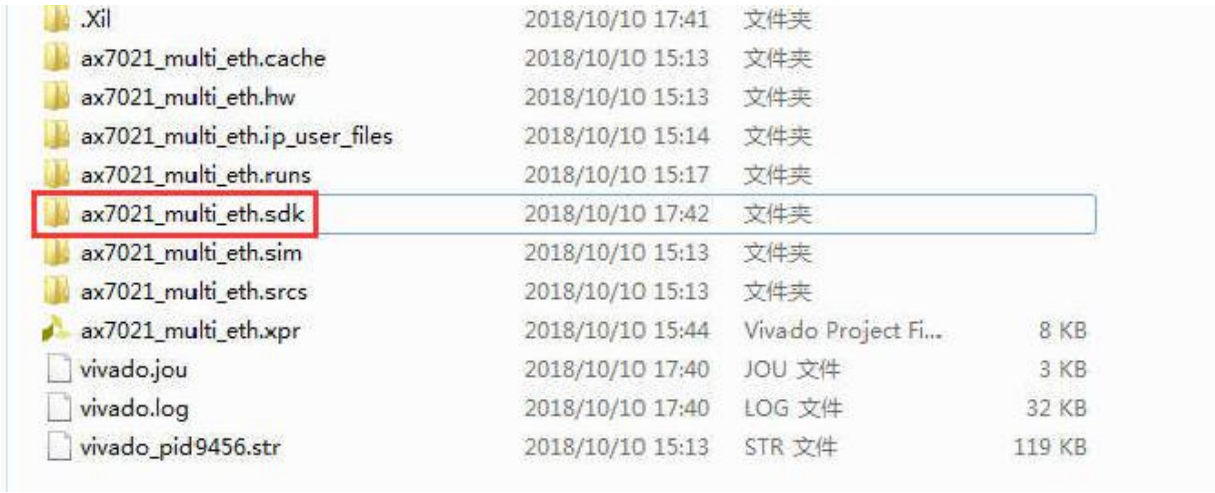
Figure 3-14: Run the program and observe the serial Output

Part 4: Application under petalinux

This section describes how to configure multiple Ethernet drivers in petalinux.

4.1 Create a petalinux project

Copy the *.sdk folder to the linux system and use it for petalinux



.Xil	2018/10/10 17:41	文件夹	
ax7021_multi_eth.cache	2018/10/10 15:13	文件夹	
ax7021_multi_eth.hw	2018/10/10 15:13	文件夹	
ax7021_multi_eth.ip_user_files	2018/10/10 15:14	文件夹	
ax7021_multi_eth.runs	2018/10/10 15:17	文件夹	
ax7021_multi_eth.sdk	2018/10/10 17:42	文件夹	
ax7021_multi_eth.sim	2018/10/10 15:13	文件夹	
ax7021_multi_eth.srcs	2018/10/10 15:13	文件夹	
ax7021_multi_eth.xpr	2018/10/10 15:44	Vivado Project Fi...	8 KB
vivado.jou	2018/10/10 17:40	JOU 文件	3 KB
vivado.log	2018/10/10 17:40	LOG 文件	32 KB
vivado_pid9456.str	2018/10/10 15:13	STR 文件	119 KB

Figure 4-1: Copy the *.sdk folder to the linux system

Open terminal

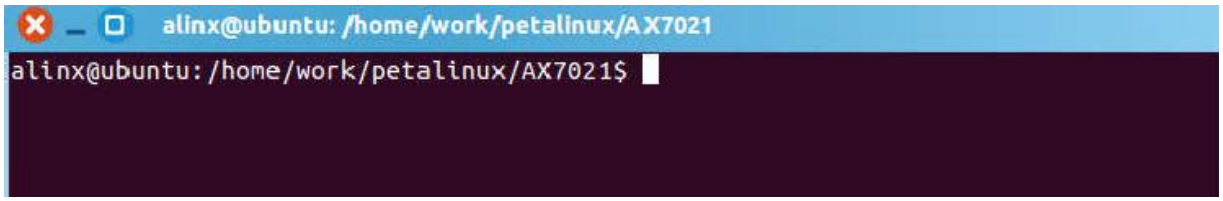


Figure 4-2: Open the Terminal

Enter the command in Figure 4-3 for petalinux and VIVADO software environment variable settings

```
source  /opt/pkg/petalinux/settings.sh

source  /opt/Xilinx/Vivado/2017.4/settings64.sh
```

Figure 4-3: Software Environment Variable Settings

```
petalinux-create --type project --template zynq --name axi_eth
```

Figure 4-4: Create a Petalinux Project

After the project is created, petalinux will automatically create a project directory "axi_eth" and run the command to enter the directory.

```
cd  axi_eth
```

Figure 4-5: Run theCcommand to enter the directory "axi_eth"

4.2 Configure petalinux hardware related information

Run the command in Figure 4-6 to configure the hardware information of the petalinux project. "../*.sdk/" is the hardware information directory exported by VIVADO. After the command is run, a configuration interface will pop up.

Here you can mainly configure the startup method. The following describes how to configure the startup mode. The default configuration starts from the startup of the SD card. If you need to modify it, you can directly “Save” and then “Exit” to exit.

“sdk” to be modified according to the actual file name

```
petalinux-config --get-hw-description ../*.sdk/
```

Figure 4-6: Configure the hardware information of the petalinux project

Then petalinux started the automatic configuration process, there are some warnings, no errors.

4.3 Modify the device tree

Find the "system-top.dts" file in the project directory, which is the top-level file for the device tree.

Figure 4-7: "system-top.dts" file in the Project Directory

Modify the system-user.dtsi content to:

```
/include/ "system-conf.dtsi"


/ {
        model = "Zynq ALINX Development Board";
        compatible = "alinx,axi eth", "xlnx,zynq-7000";
        usb_phy0: usb_phy@0 {
                compatible = "ulpi-phy";
                #phy-cells = <0>;
                reg = <0xe0002000 0x1000>;
                view-port = <0x0170>;
                drv-vbus;
        };



};



&usb0 {
        usb-phy = <&usb_phy0>;
};
```

```
&sdhci0 {
        u-boot,dm-pre-reloc;
};

&uart1 {
        u-boot,dm-pre-reloc;
};


&flash0 {
        compatible = "micron,m25p80", "w25q256", "spi-flash";
};


&gem0 {
        phy-handle = <&ethernet_phy>;
        ethernet_phy: ethernet-phy@1 {
                reg = <1>;
                device_type = "ethernet-phy";
        };
};

&axi_ethernet_0 {
        local-mac-address = [00 0a 35 00 03 22];
        phy-handle = <&phy1>;
        xlnx,has-mdio = <0x1>;
```

```
        phy-mode = "rgmii";
        mdio {
                phy1: phy@1 {
                        device_type = "ethernet-phy";
                        reg = <1>;
                };
        };
};

&axi_ethernet_1 {
        local-mac-address = [00 0a 35 00 03 23];
        phy-handle = <&phy2>;
        xlnx,has-mdio = <0x1>;
        phy-mode = "rgmii";
        mdio {
                phy2: phy@1 {
                        device_type = "ethernet-phy";
                        reg = <1>;
                };
        };
};

&axi_ethernet_2 {
        local-mac-address = [00 0a 35 00 03 24];
        phy-handle = <&phy3>;
        xlnx,has-mdio = <0x1>;
```

```
        phy-mode = "rgmii";
        mdio {
                phy3: phy@1 {
                        device_type = "ethernet-phy";
                        reg = <1>;
                };
        };
};

&axi_ethernet_3 {
        local-mac-address = [00 0a 35 00 03 25];
        phy-handle = <&phy4>;
        xlnx,has-mdio = <0x1>;
        phy-mode = "rgmii";
        mdio {
                phy4: phy@1 {
                        device_type = "ethernet-phy";
                        reg = <1>;
                };
        };
};
```

4.4 Configuring the kernel

Enter the command in Figure 4-8 to configure the kernel.

```
petalinux-config -c kernel
```

Figure 4-8: Command to configure the kernel

4.4.1： Configure phy driver

Select "<*> Micrel PHYs" in the "Device Drivers > Network device support > PHY Device support and infrastructure", then "Save", and finally "Exit" to exit the configuration interface.

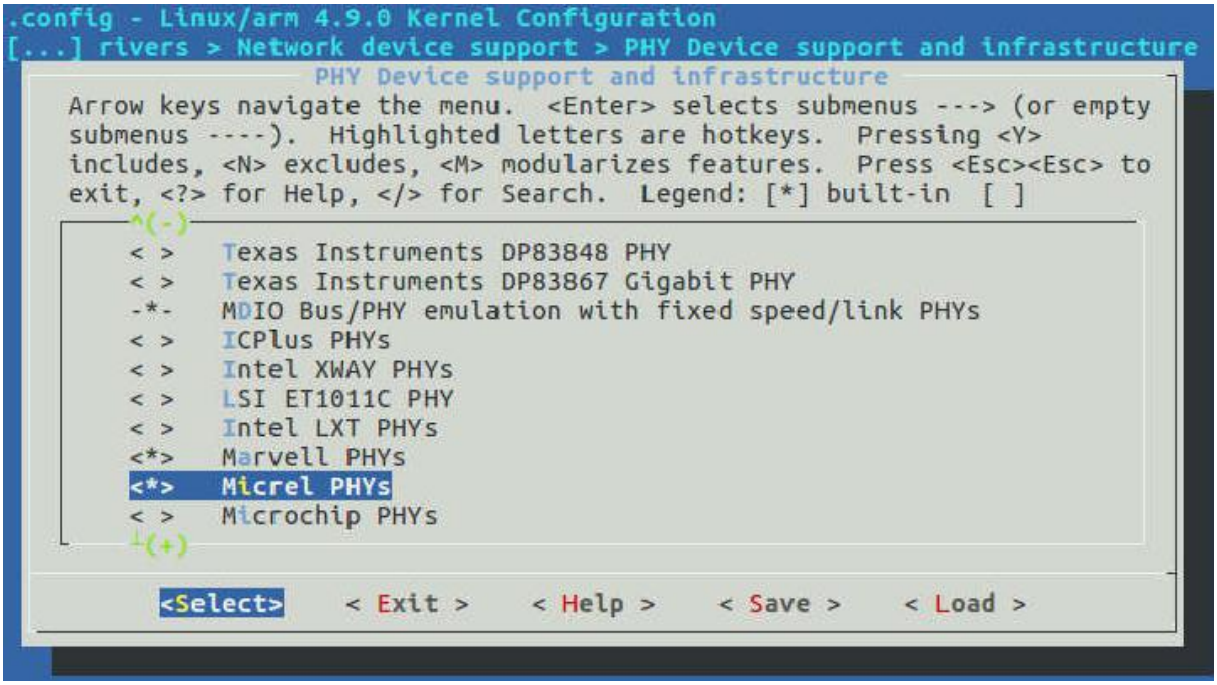


Figure 4-9: Configure phy driver

4.5 Configuring the root file system

Do not make any changes to the linux root file system. If you need to modify the root file system, you can run the following command in Figure 4-10:

```
petalinux-build
```

Figure 4-10: Modify the Root File System

4.6 Build petalinux

Run the command in Figure 4-11 to compile and build the petalinux project.

```
petalinux-build
```

Figure 4-11: Compile and build the petalinux project

4.7 Start petalinux from SD card

Run the command in Figure 4-12 to generate the startup file.

```
petalinux-package --boot --fsbl ./images/linux/zynq_fsbl.elf  --fpga  --uboot  --force
```

Figure 4-12: Generate the startup file

Copy the BOOT.BIN and image.ub files to the FAT32 partition of the SD card

4.8 Configuring the Ethernet port

First use the serial terminal tool to log in to the system, use the root account, password root

Enter the following configuration Ethernet IP address and start it as follows

```
ifconfig eth1 192.168.1.101 netmask 255.255.255.0 up

ifconfig eth2 192.168.1.102 netmask 255.255.255.0 up

ifconfig eth3 192.168.1.103 netmask 255.255.255.0 up

ifconfig eth4 192.168.1.104 netmask 255.255.255.0 up
```

Figure 4-13: Configuration Ethernet IP address

```
PetaLinux 2017.4 axi_eth /dev/ttyPS0

axi_eth login: root
Password:
root@axi_eth:~# ifconfig eth1 192.168.1.101 netmask 255.255.255.0 up
IPv6: ADDRCONF(NETDEV_UP): eth1: link is not ready
root@axi_eth:~# ifconfig eth2 192.168.1.102 netmask 255.255.255.0 up
IPv6: ADDRCONF(NETDEV_UP): eth2: link is not ready
root@axi_eth:~# ifconfig eth3 192.168.1.103 netmask 255.255.255.0 up
IPv6: ADDRCONF(NETDEV_UP): eth3: link is not ready
root@axi_eth:~# ifconfig eth4 192.168.1.104 netmask 255.255.255.0 up
IPv6: ADDRCONF(NETDEV_UP): eth4: link is not ready
root@axi_eth:~# xilinx_axienet 41000000.ethernet eth1: Link is Down
xilinx_axienet 41040000.ethernet eth2: Link is Down
xilinx_axienet 41080000.ethernet eth3: Link is Down
xilinx_axienet 410c0000.ethernet eth4: Link is Down
xilinx_axienet 41040000.ethernet eth2: Link is Up - 1Gbps/Full - flow control rx/tx
IPv6: ADDRCONF(NETDEV_CHANGE): eth2: link becomes ready

root@axi_eth:~#
```

Figure 4-14: Configuration Ethernet IP address and start it

Run "ifconfig" command to view status

```
root@axi_eth:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0A:35:00:1E:53
          inet addr:192.168.1.46  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20a:35ff:fe00:1e53%lo/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10 errors:0 dropped:1 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3419 (3.3 KiB)  TX bytes:1192 (1.1 KiB)
          Interrupt:27 Base address:0xb000

eth1      Link encap:Ethernet  HWaddr 00:0A:35:00:03:22
          inet addr:192.168.1.101  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth2      Link encap:Ethernet  HWaddr 00:0A:35:00:1E:53
          inet addr:192.168.1.102  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20a:35ff:fe00:1e53%lo/64 Scope:Link
          UP BROADCAST  MTU:1500  Metric:1
          RX packets:532 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:77058 (75.2 KiB)  TX bytes:998 (998.0 B)

eth3      Link encap:Ethernet  HWaddr 00:0A:35:00:03:24
          inet addr:192.168.1.103  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth4      Link encap:Ethernet  HWaddr 00:0A:35:00:03:25
          inet addr:192.168.1.104  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1%1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:2356 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2356 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:685456 (669.3 KiB)  TX bytes:685456 (669.3 KiB)
```

Figure 4-15: Configuring the Ethernet port

4.9 Test the Ethernet port

Plug the first PL port Ethernet port (ETH1) into the cable, Use eth1 to ping

192.168.1.10 in the LAN

ping -I eth1 192.168.1.10

```
root@axi_eth:~# ping  -I   eth1 192.168.1.10
PING 192.168.1.10 (192.168.1.10): 56 data bytes
64 bytes from 192.168.1.10: seq=0 ttl=64 time=0.382 ms
64 bytes from 192.168.1.10: seq=1 ttl=64 time=0.207 ms
64 bytes from 192.168.1.10: seq=2 ttl=64 time=0.197 ms
64 bytes from 192.168.1.10: seq=3 ttl=64 time=0.183 ms
64 bytes from 192.168.1.10: seq=4 ttl=64 time=0.177 ms
```

Figure 4-16: Test the Ethernet port

Part 5: Q&A

5.1 Petalinux Startup Error

If the error shown in Figure 5-1 appears, you can format the sd card and try again.

```
U-Boot 2015.07 (Oct 12 2017 - 13:33:23 +0800)

DRAM:  ECC disabled 1 GiB
MMC:   zynq_sdhci: 0
SF: Detected W25Q256 with page size 256 Bytes, erase size 4 KiB, total 32 MiB
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Net:   Gem.e000b000
U-BOOT for ax7021_multi_eth

Gem.e000b000 Waiting for PHY auto negotiation to complete......... TIMEOUT !
Gem.e000b000: No link.
Hit any key to stop autoboot:  0
Device: zynq_sdhci
Manufacturer ID: 3
OEM: 5344
Name: SL08G
Tran Speed: 50000000
Rd Block Len: 512
SD version 3.0
High Capacity: Yes
Capacity: 7.4 GiB
Bus Width: 4-bit
Erase Group Size: 512 Bytes
reading image.ub
Invalid FAT entry
77824 bytes read in 16 ms (4.6 MiB/s)
## Loading kernel from FIT Image at 01000000 ...
Bad FIT kernel image format!
ERROR: can't get kernel image!
U-Boot-PetaLinux>
```

Figure 5-1: Petalinux Startup Error