Z7-Lite 系列教程
之 Linux 系统篇
Xilinx ZYNQ-7000 SoC

Rev. 1.0

版本记录

版本	时间	描述
Rev. 1.0	2020.6.1	第一版发布

版权声明：

Mizar 系列核心模块、评估板及其相关知识产权由微相科技有限公司所有。

本文档由微相科技有限公司版权所有，并保留一切权利。在未经微相公司书面许可的情况下，不得以任何方式或形式来修改、分发或复制本文档的任何部分。

免责声明：

产品中所提供的程序源代码、软件、资料文档等，微相科技不提供任何类型的担保；不论是明确的，还是隐含的，包括但不限于合适特定用途的保证，全部的风险，由使用者来承担。

目录

版本记录 2
版权声明： 3
免责声明： 3
目录 4
第 1 章 虚拟机和 Ubuntu 系统安装 7
1.1 安装虚拟机 7
1.2 创建虚拟机 11
1.3 虚拟机配置 14
1.4 安装 Ubuntu 系统 19
1.5 配置 Ubuntu 系统 25
1.6 设置/bin/sh 为 bash 30
第 2 章 Ubuntu 系统下 Linux 版 Vivado 安装 32
2.1 安装 Linux 版 Vivado 32
2.2 添加权限 38
2.3 安装下载器驱动 38
2.4 测试下载器驱动 38
第 3 章 PetaLinux 安装 41
3.1 安装依赖库 41
3.2 安装 Petalinux 43
第 4 章 NFS 服务软件安装 45
4.1 安装 NFS 服务 45
4.2 测试 NFS 46
第 5 章 PetaLinux 定制 Linux(SD 卡启动) 48
5.1 Vivado 创建工程 48
5.2 设计导出至 SDK 60

5.3 PetaLinux 工程建立 61
5.4 BOOT.bin 文件打包 67
5.5 SD 卡启动 71
5.6 实验小结 74
第 6 章 PetaLinux 定制 Linux(QSPI 启动) 75
6.1 Vivado 创建工程 75
6.2 设计导出 SDK 77
6.3 PetaLinux 工程建立 78
6.4 BOOT.bin 文件烧录 87
6.5 QSPI 启动 88
6.6 实验小结 92
第 7 章 驱动 USB 外设 93
7.1 创建 Vivado 工程 93
7.2 创建 Petalinux 工程 96
7.3 配置 kernel 98
7.4 修改设备树及编译工程 101
7.5 BOOT.bin 文件打包与烧录验证 102
第 8 章 USB 外设 slave 模式应用实验 105
8.1 创建 Vivado 工程 105
8.2 创建 Petalinux 工程 106
8.3 配置 kernel 107
8.4 配置设备树 111
8.5 编译工程及打包启动文件 112
8.6 系统启动验证 114
第 9 章 Linux 下 GPIO 控制实验 125
9.1 创建 Vivado 工程 125

9.2 创建 Petalinux 工程.....130
9.3 BOOT.bin 文件打包.....133
9.4 系统启动验证.....134
第 10 章 Linux 下 HDMI 显示实验.....145
10.1 创建 Vivado 工程.....145
10.2 创建并配置 Petalinux 工程.....151
10.3 配置 Linux 内核.....156
10.4 配置设备树.....159
10.5 编译工程及打包制作 BOOT.bin.....161
10.6 系统启动验证.....162
第 11 章 Linux 下 LCD 显示实验.....164
11.1 创建 Vivado 工程.....164
11.2 创建并配置 Petalinxu 工程.....168
11.3 配置内核及设备树.....170
11.4 编译工程及打包制作 BOOT.bin 文件.....172
11.5 系统启动验证.....173
第 12 章 开发板运行 Linux 图形界面.....175
12.1 简介.....175
12.2 修改 Vivado 工程.....178
12.3 配置 Petalinux 工程.....179
12.4 编译 Petalinux 工程并打包 BOOT.BIN 文件.....181
12.5 制作 SD 启动卡.....181
12.6 系统启动验证.....188

第1章　虚拟机和 Ubuntu 系统安装

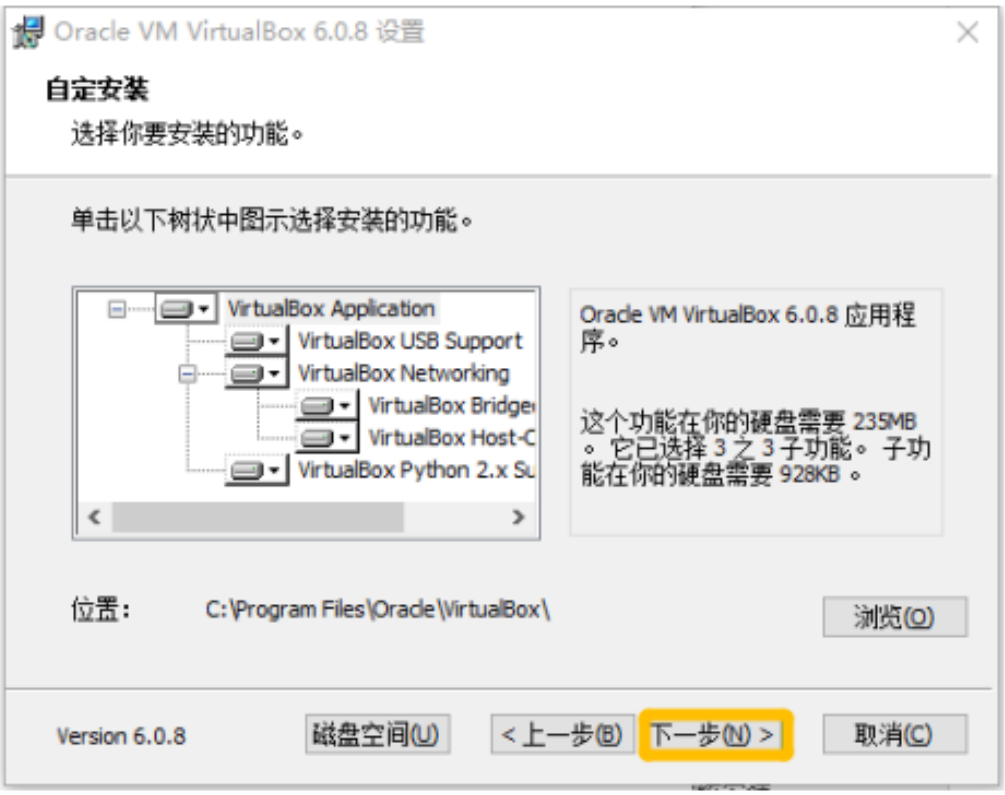
在本教程中，我们将学习嵌入式 Linux 的开发，我们需要建立一个 Linux 开发环境，需要一台安装 Linux 操作系统的主机，用来编译 u-boot 和 Linux-Kernel。在 Windows 操作系统的电脑安装虚拟机，在虚拟机上安装一个 Linux 操作系统是最为方便常用的办法。我们在这里选择最易用的 Ubuntu Linux，并将其安装在虚拟机 VirtualBox 中。

1.1 安装虚拟机

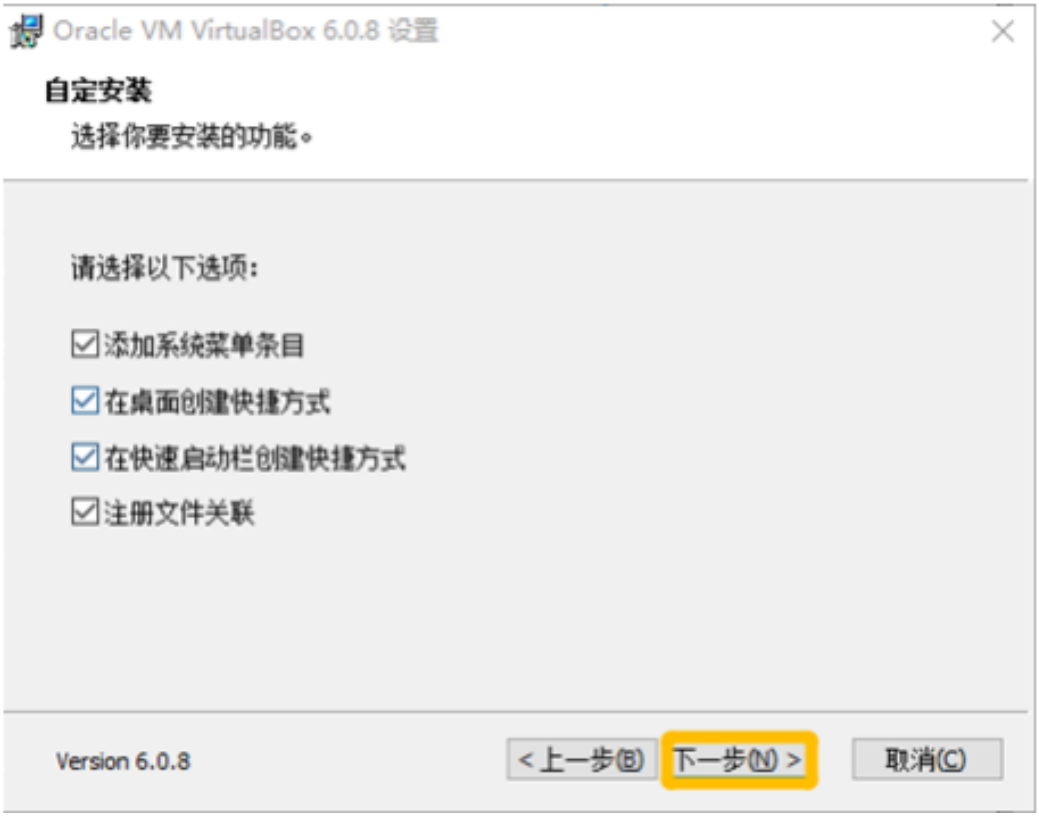
虚拟机版本使用 VirtualBox-6.0.8-130520-Win，读者可以在提供的资料中查找，也可以通过官网 http：//www.virtualbox.org/找到，另外为了支持 USB 设备，还需要下载并安装 VirtualBox Extension Pack。下面介绍虚拟机安装的步骤。

1）双击 VirtualBox-6.0.8-130520-Win.exe，点击下一步。

2）点击下一步。



3) 点击下一步。



4) 点击是。

Oracle VM VirtualBox 6.0.8
警告：
网络界面
安装 Oracle VM VirtualBox 6.0.8 网络功能将重置网络连接并暂时中断网络连接。
立即安装吗？
Version 6.0.8
是(Y)
否(N)

5) 点击安装。

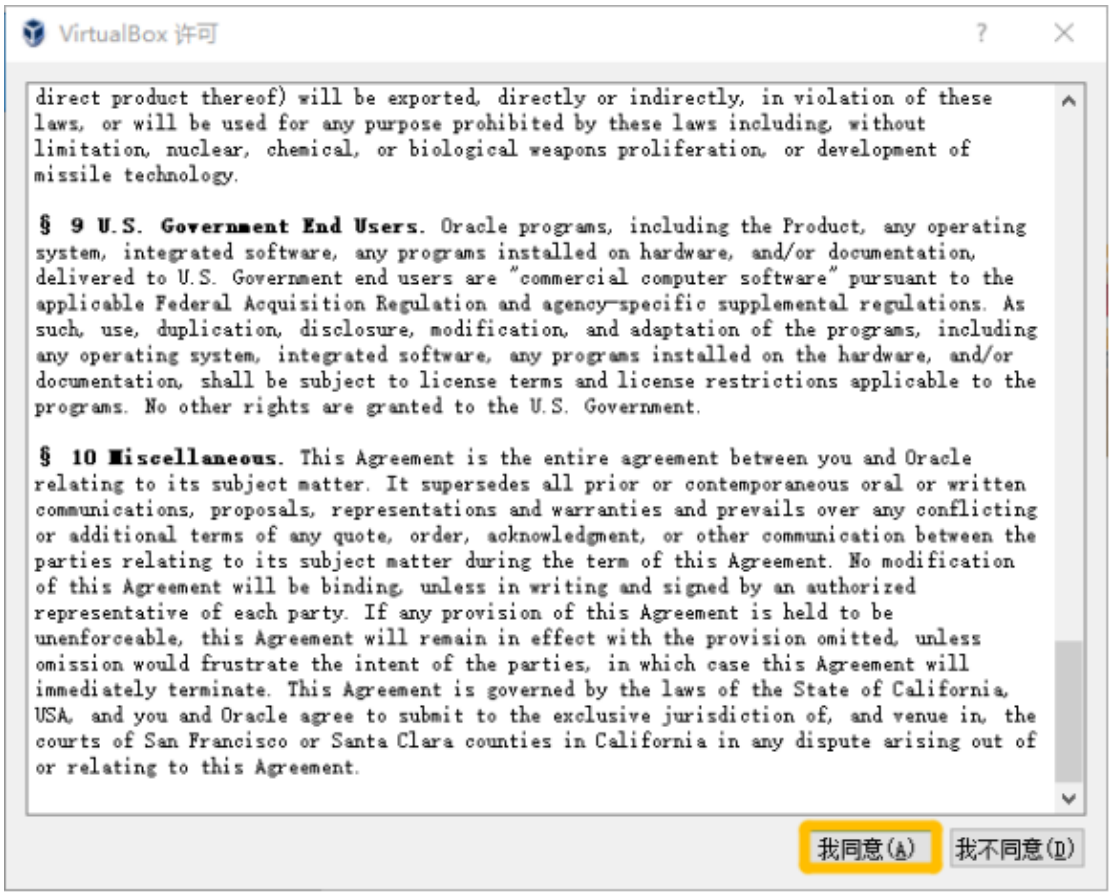
Oracle VM VirtualBox 6.0.8 设置
准备好安装
安装向导准备好进行 自定 安装。
单击"安装"开始安装。 如果你要检查或更改任何安装设置，单击"上一步"。 单击"取消"结束向导。
Version 6.0.8
< 上一步(B)
安装(I)
取消(C)

6) 弹出 Windows 安全对话框，点击是。

7) 点击完成，这样 Virtual Box 就安装完毕。

8) 双击文件 Oracle_VM_VirtualBox_Extension_Pack-6.0.8（注意：扩展包不要放在中文路径下）， 弹出弹窗，点击安装。

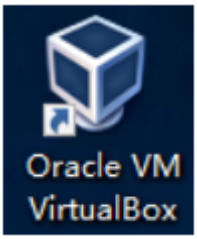
9) 将文档拖至最后，点击我同意。



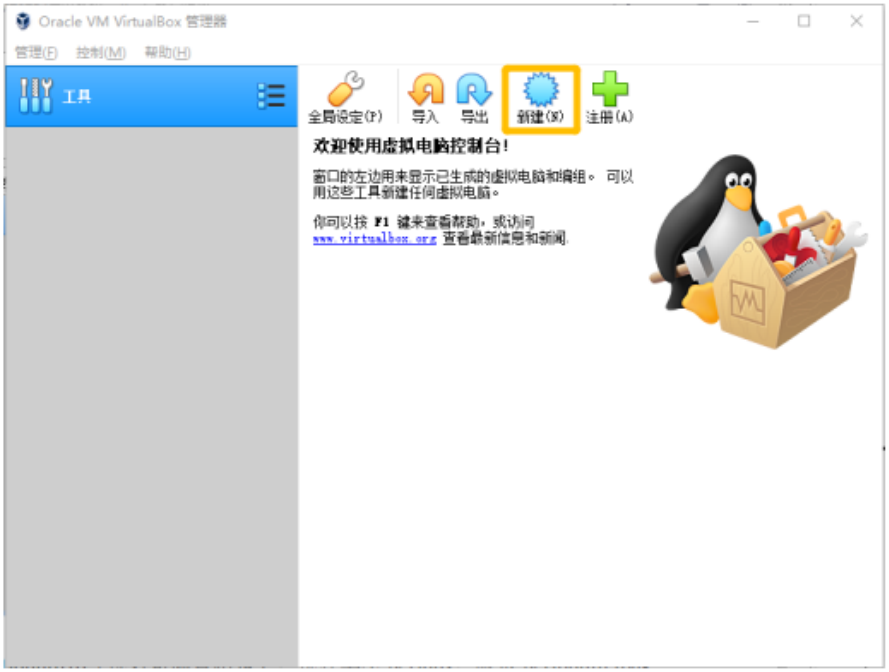
10) 安装完成，点击确定。

1.2 创建虚拟机

1) 双击桌面快捷方式 Oracle VM VirtualBox， 打开 VirtualBox。

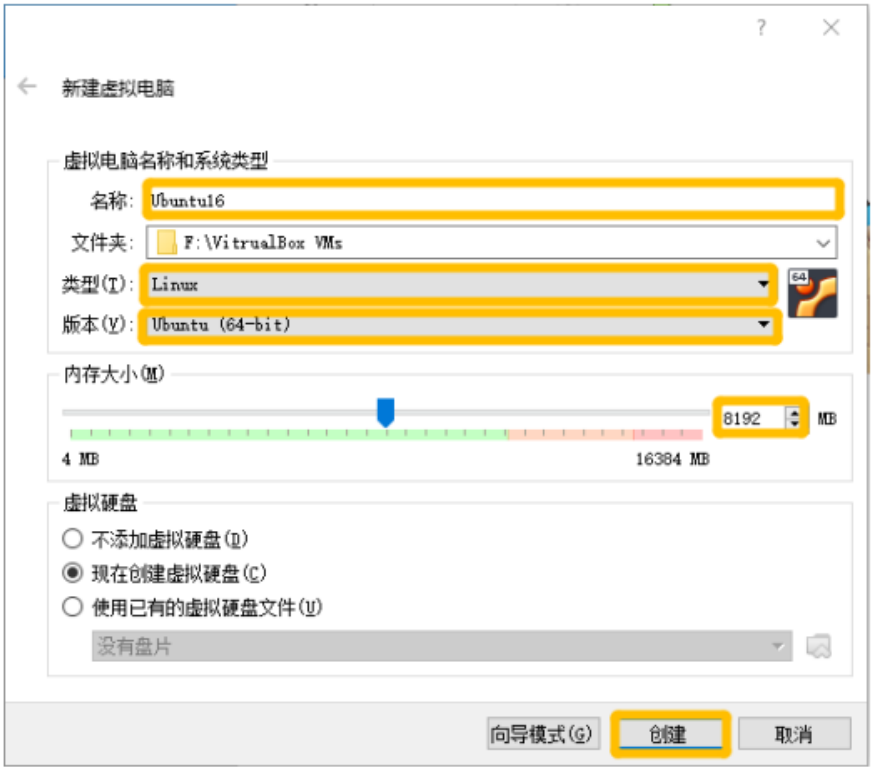


2) 点击新建。

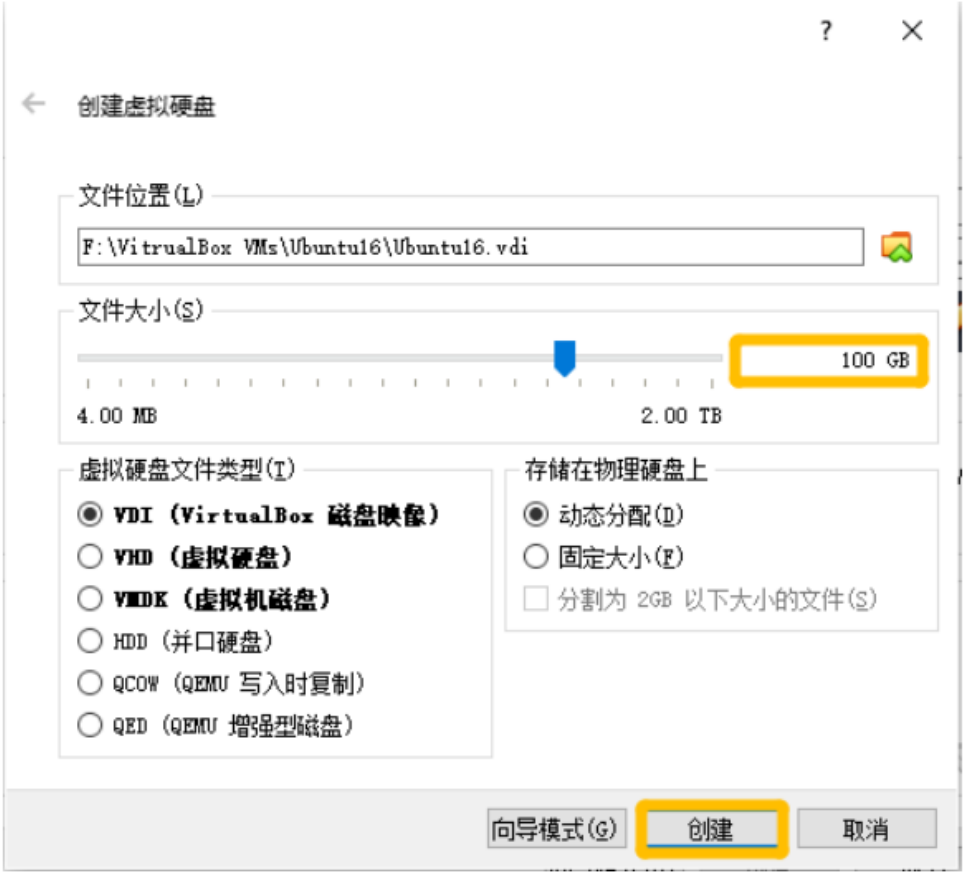


3) 名称输入 Ubuntu16， 类型选择 Linux， 版本选择 Ubunt(64-bit)， 内存选择 8192MB，点击创建。

特别说明：在创建虚拟机过程中，操作系统版本选项只有 64bit， 有些硬件上(32bitCPU)不支持虚拟化技术，大部分的原因是因为主板 Bios 没有开启虚拟化支持选项，少数还有其他原因，这里不做详细阐述，读者可自行查找解决。



4) 文件大小选择 100GB，点击创建。

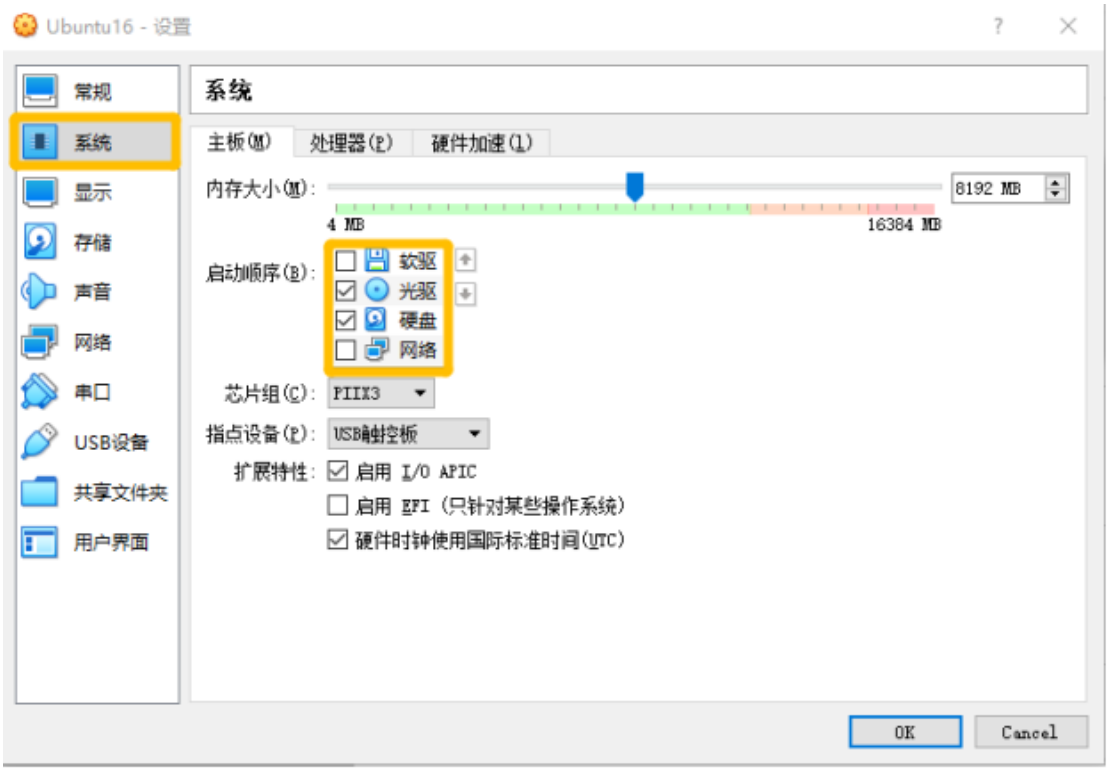


5) 虚拟机 Ubuntu 创建成功。

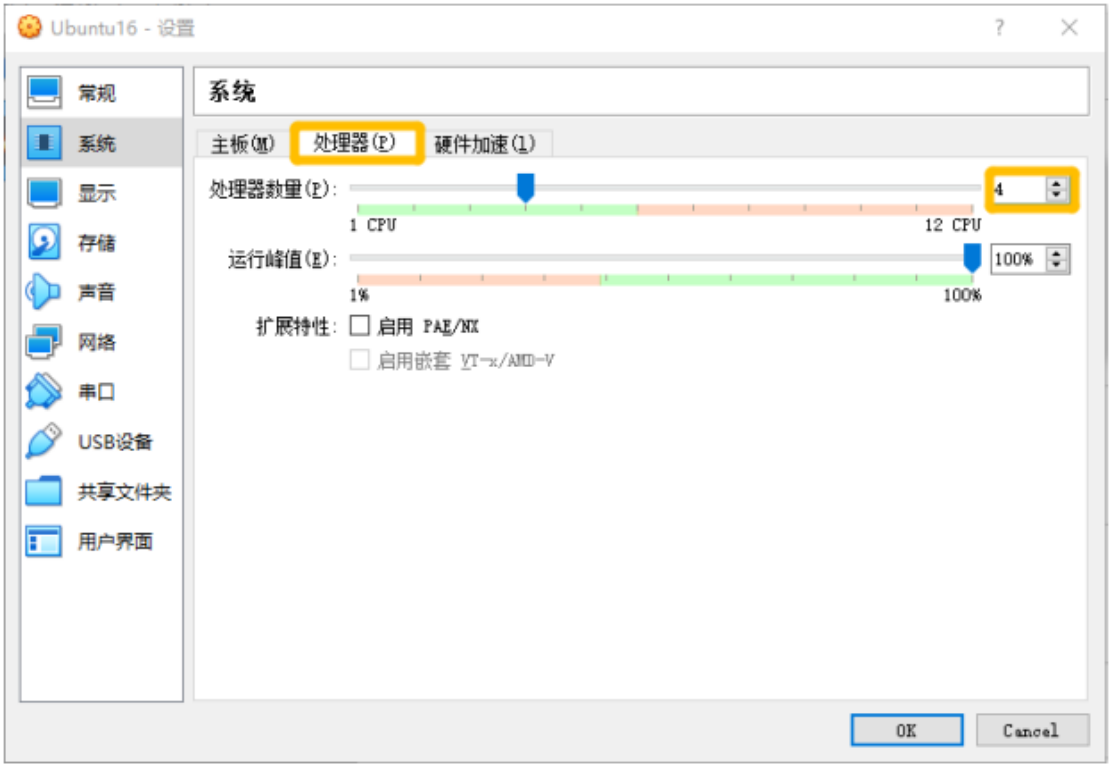
1.3 虚拟机配置

1) 选中虚拟机 Ubuntu16，点击设置。

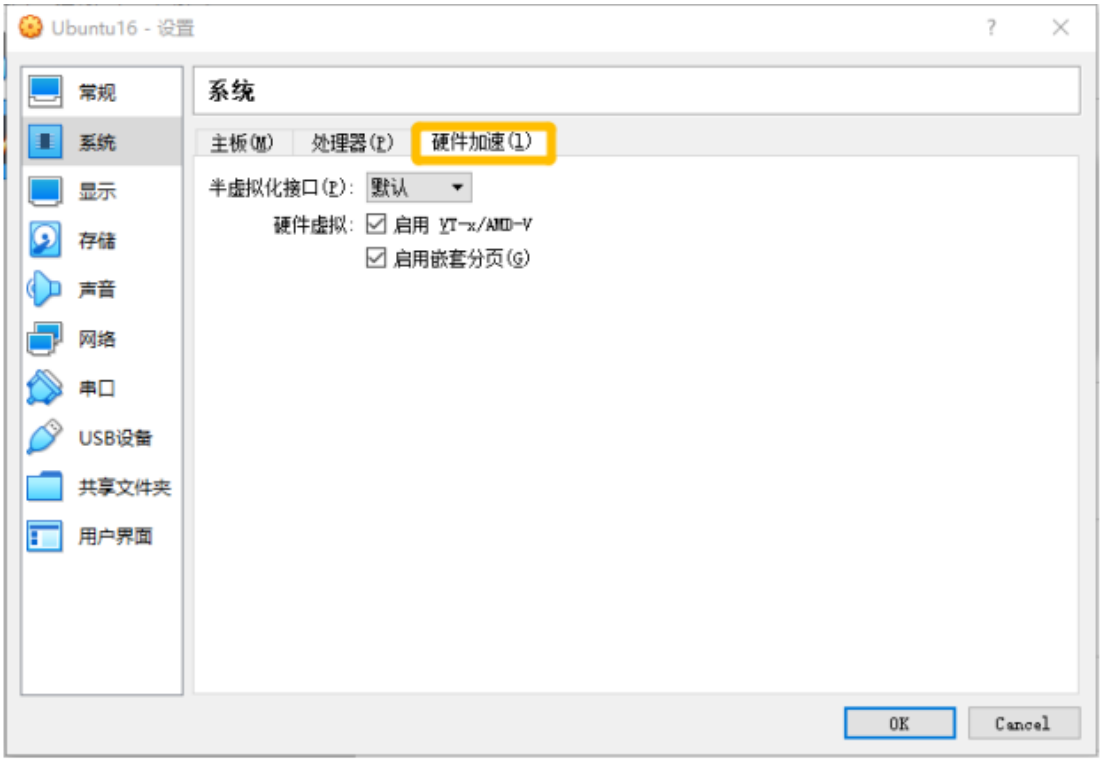
2) 点击左侧“系统”，勾掉默认的软驱，启动顺序为光驱、硬盘。



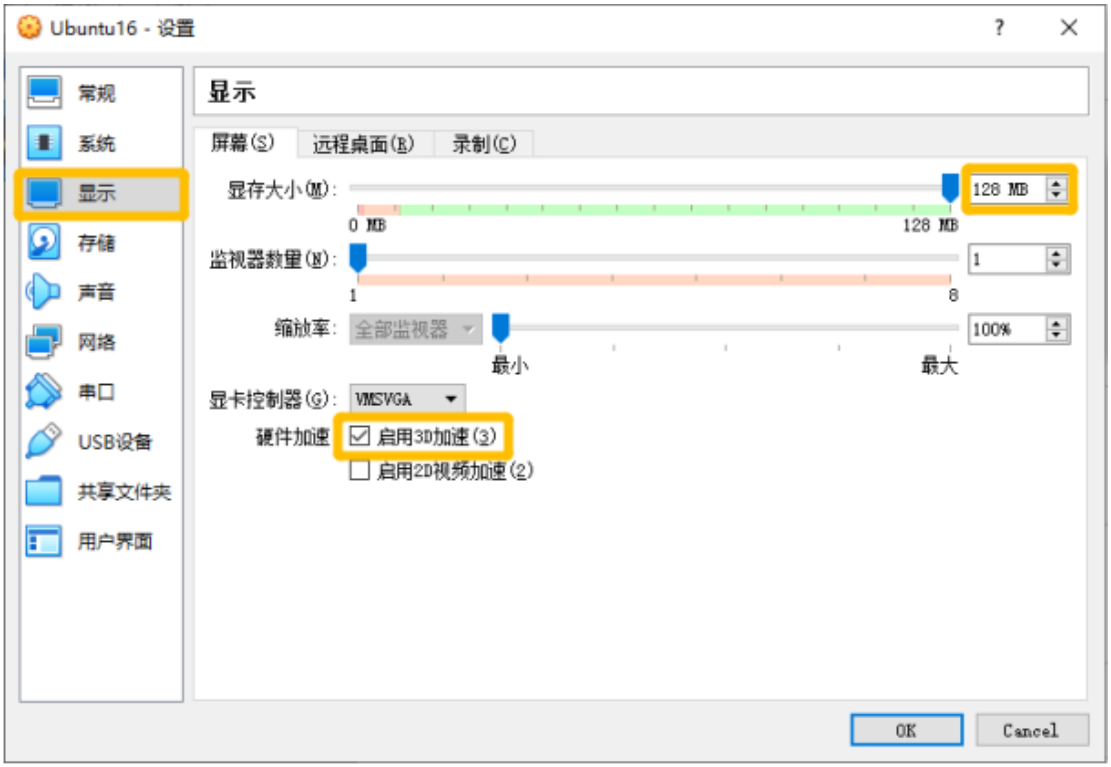
3) 点击处理器选项卡，处理器数量输入 4。



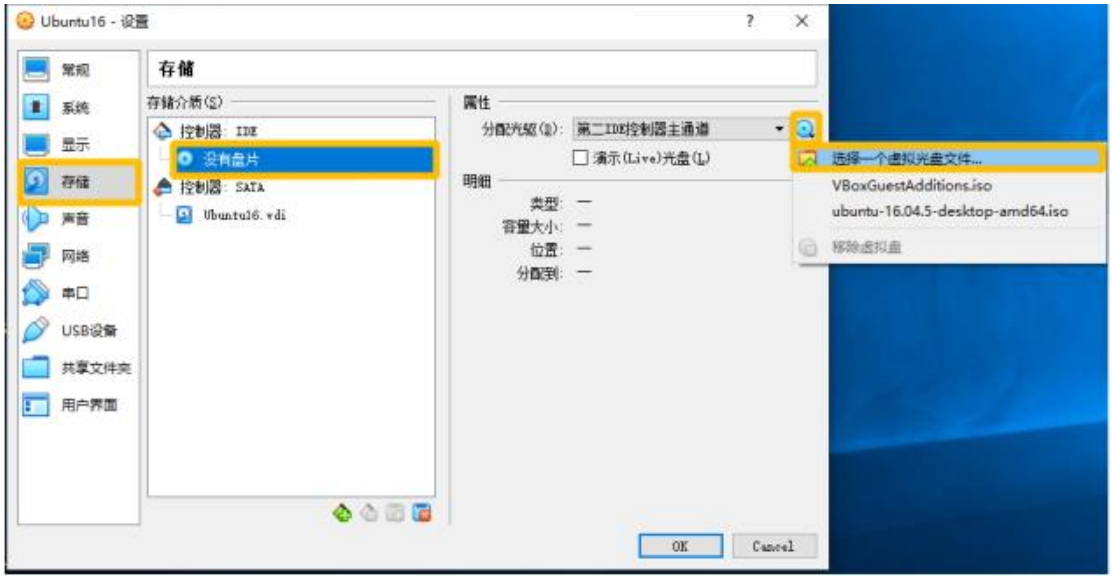
4) 硬件加速项，保持不变。



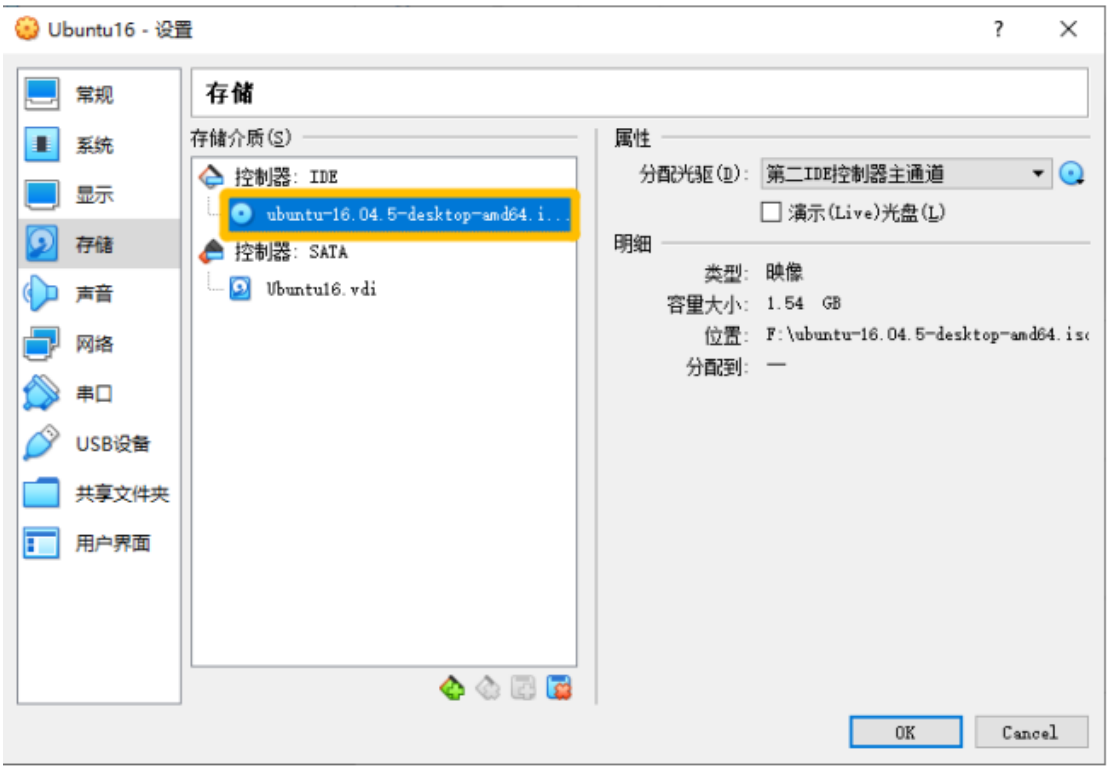
5) 点击左侧“显示”，显存大小输入 128MB， 勾选启用 3D 加速。



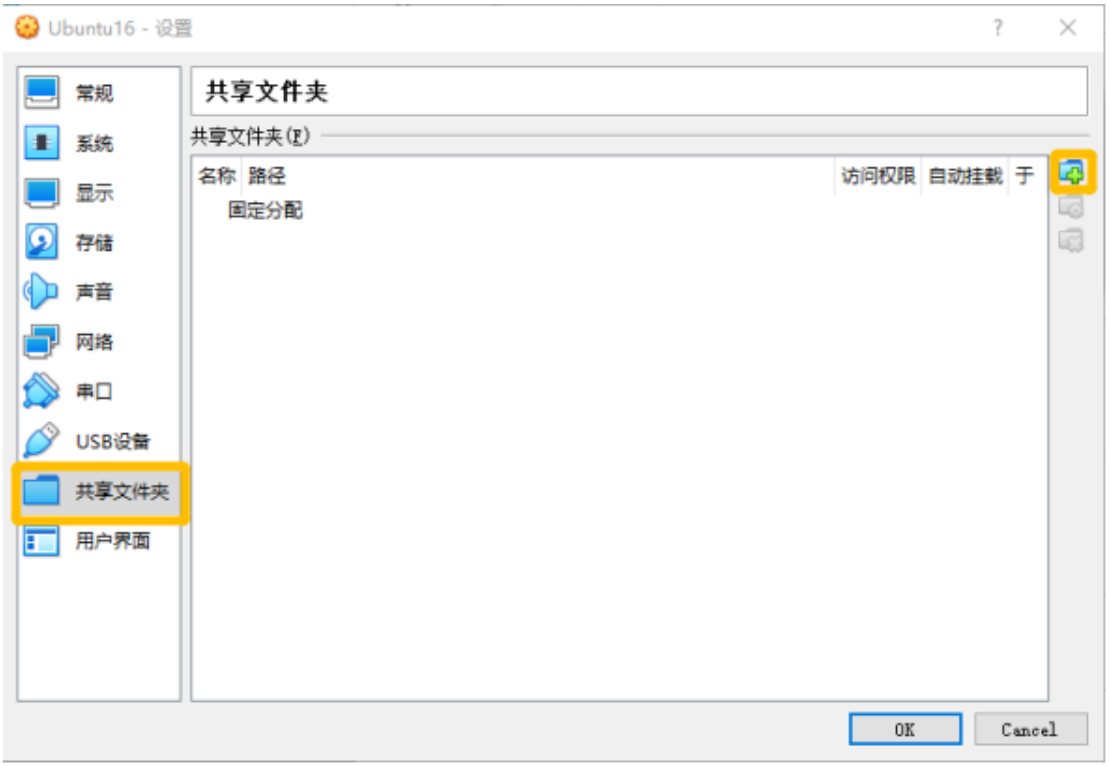
6) 点击左侧 “存储”，选择没有盘片，分配光驱，选择一个虚拟光盘文件，指向 Ubuntu 镜像文件 ubuntu-16.04.5-desktop-amd64.iso，



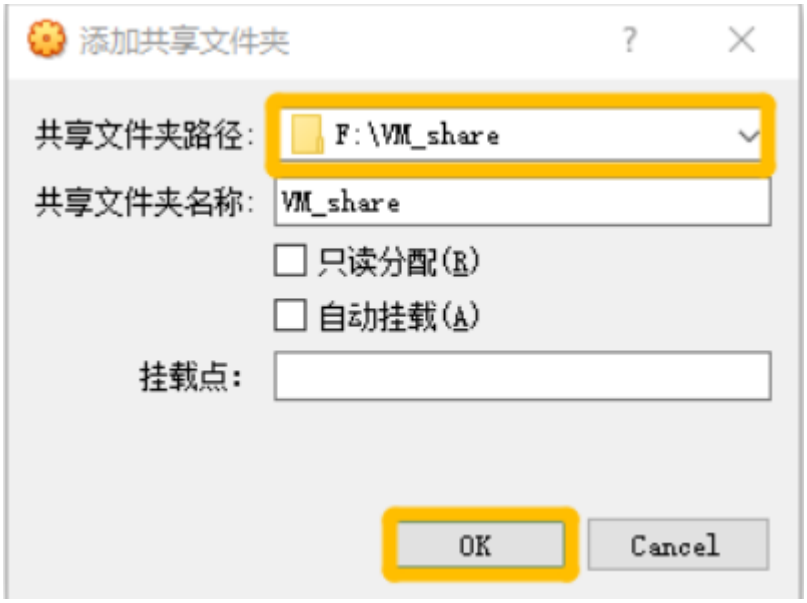
完成后，如图所示。



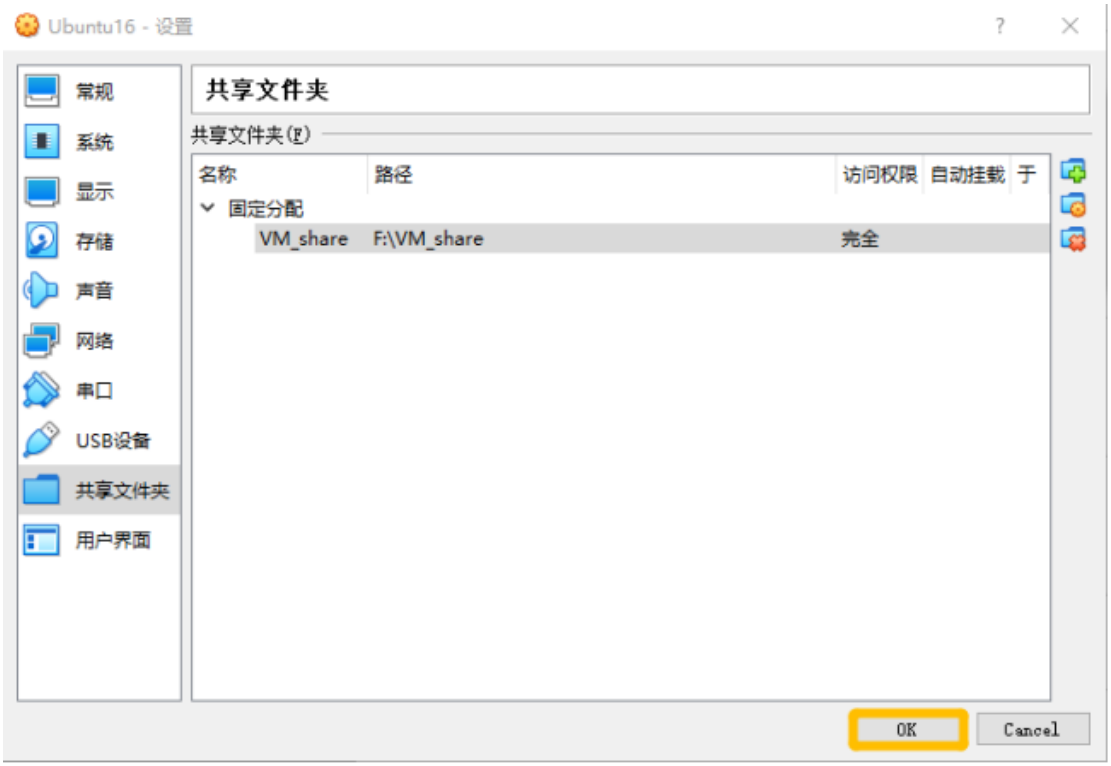
7) 设置共享文件夹，首先在根目录下创建文件夹 VM_share， 作为 Windows 主机和 Linux 虚拟机交换文件的公共空间。点击左侧“共享文件夹”，点击窗口右侧添加共享文件夹按钮。



8) 选中设置的共享文件夹路径，点击 OK。



9) 点击 OK， 虚拟机配置完成。



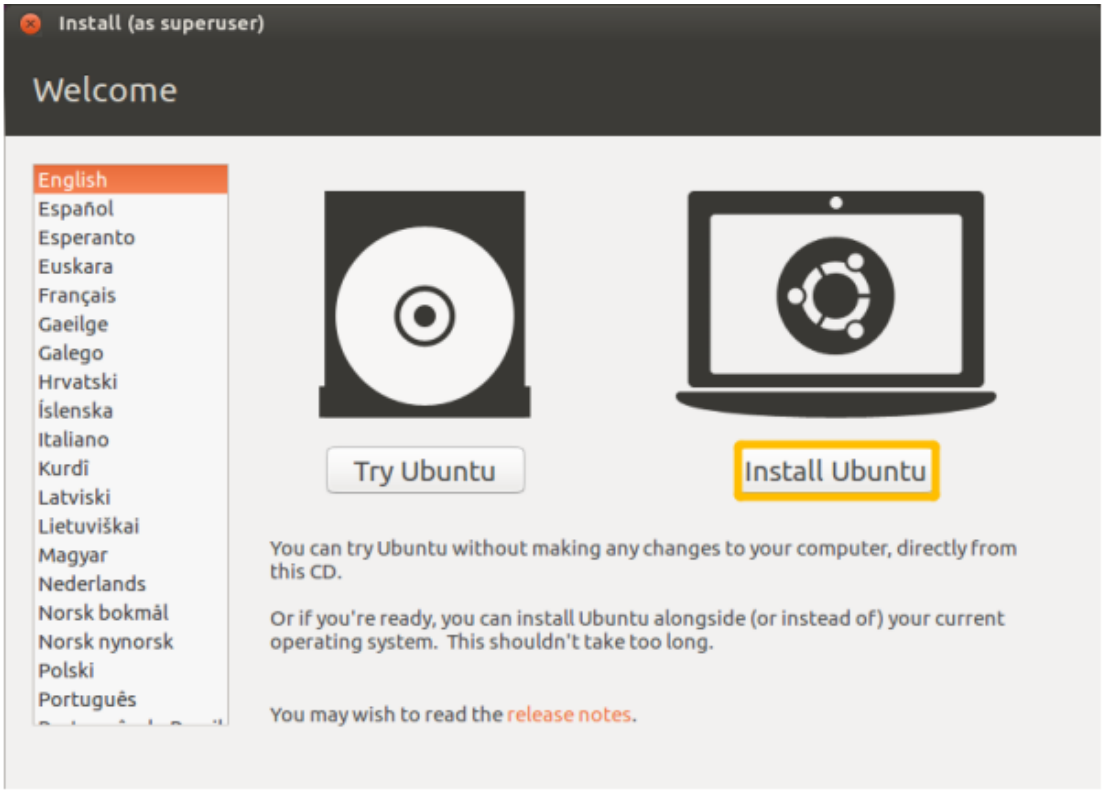
1.4 安装 Ubuntu 系统

安装完虚拟机后，就要在虚拟机上安装 Linux 操作系统，为了方便读者的使用，这里选择最易用的 Ubuntu Linux 界面操作系统，它的安装和配置都很简单。

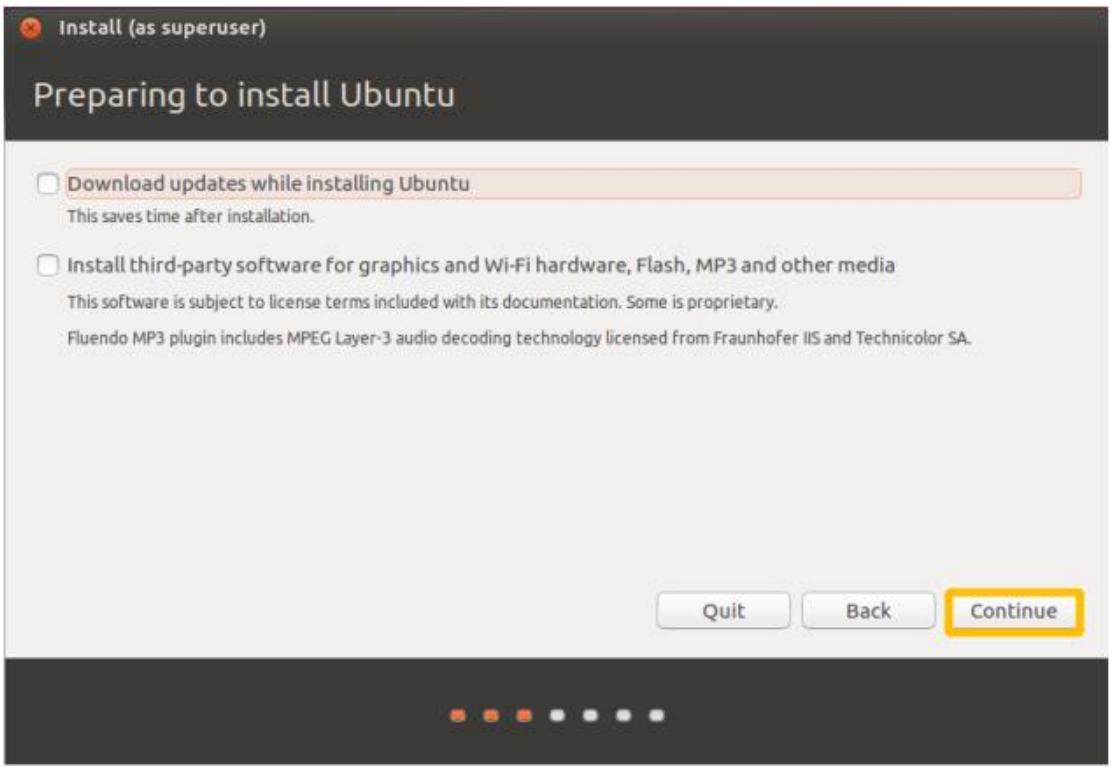
本教程使用 Ubuntu 16.04.5 LTS 64 位操作系统。可以在网上下载镜像安装包也可以使用资料中的安装包 ubuntu-16.04.5-desktop-amd64.iso。安装教程如下。

1) 选中虚拟机 Ubuntu16， 点击工具栏中启动按钮。

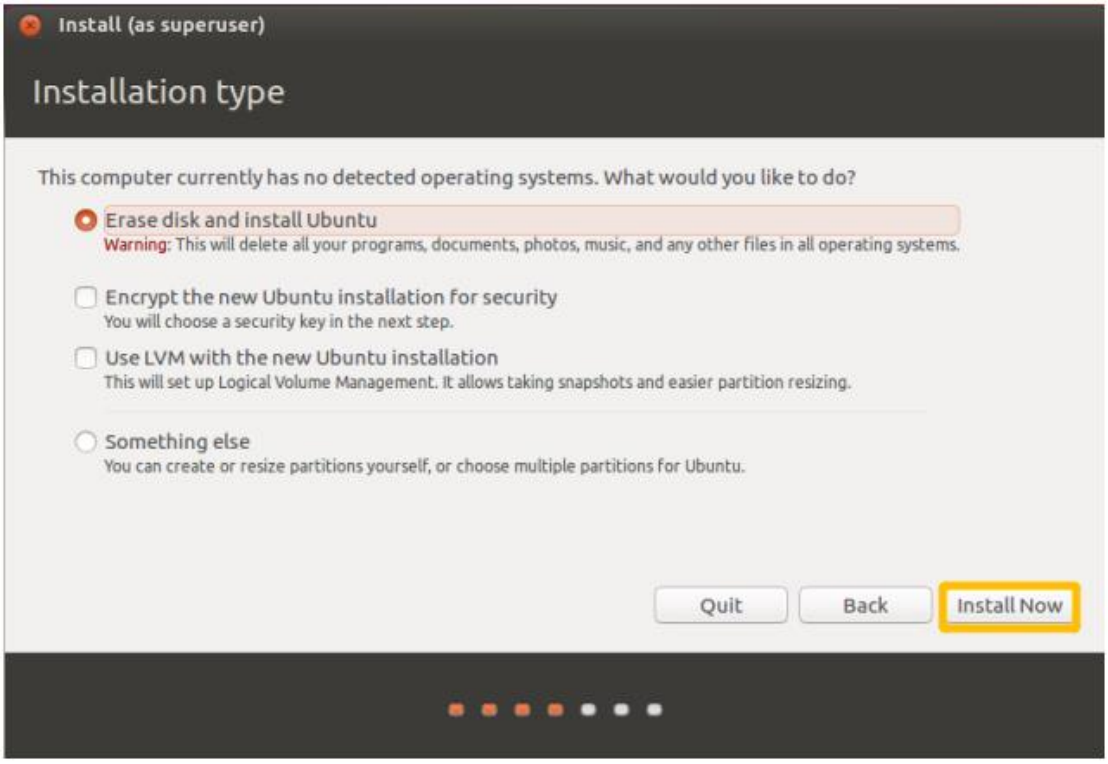
2) 点击 Install Ubuntu。



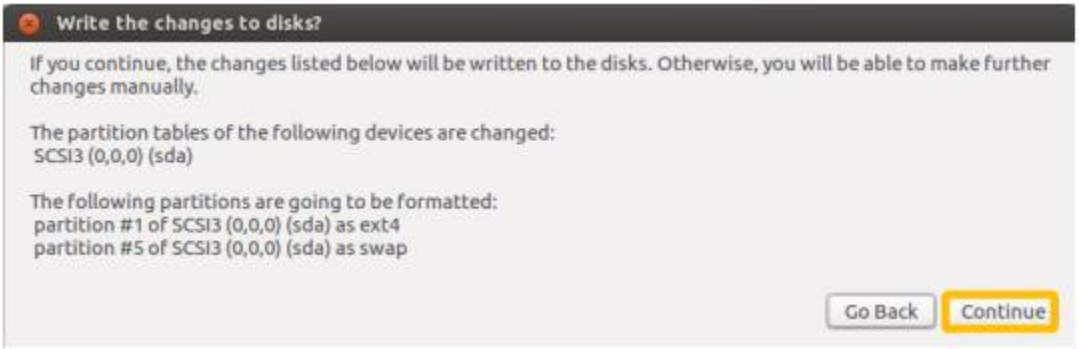
3) 点击 Continue。



4) 点击 Install Now。

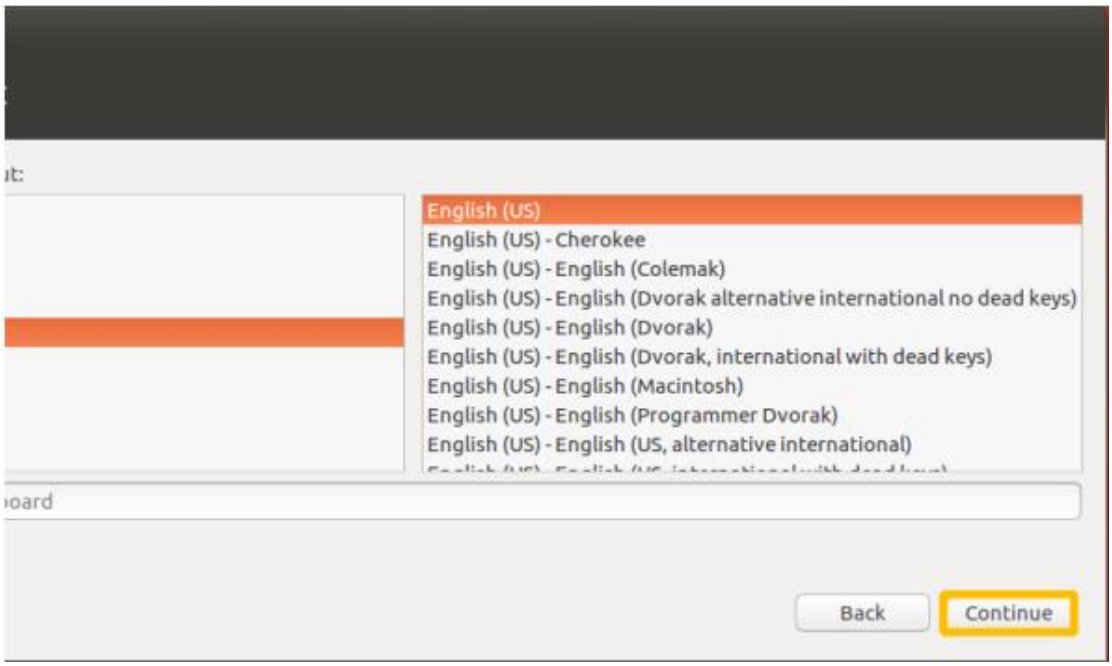


5) 点击 Continue。



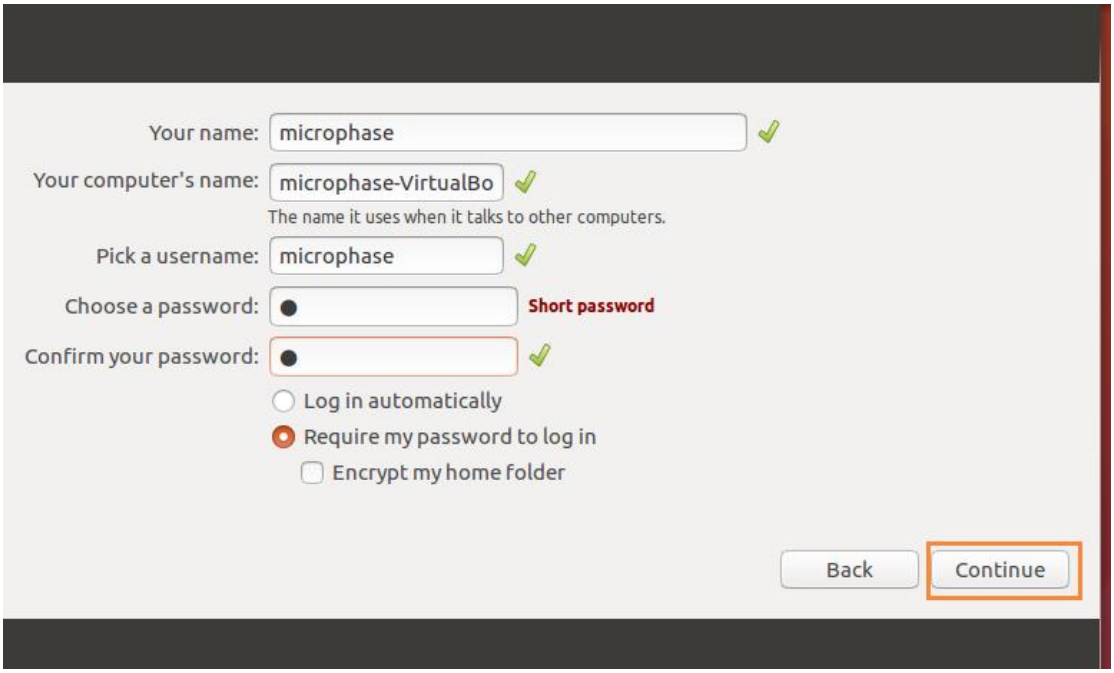
6) 市区选择 Shanghai， 点击 Continue。

7) 键盘布局保持默认，点击 Continue。（有些读者安装过程中无法看到 Continue 按钮，由于系统默认分辨率与电脑分辨率的差异导致的，可以左手按住 alt 键，右手鼠标往左拖动安装界面，即可显示全部内容）。



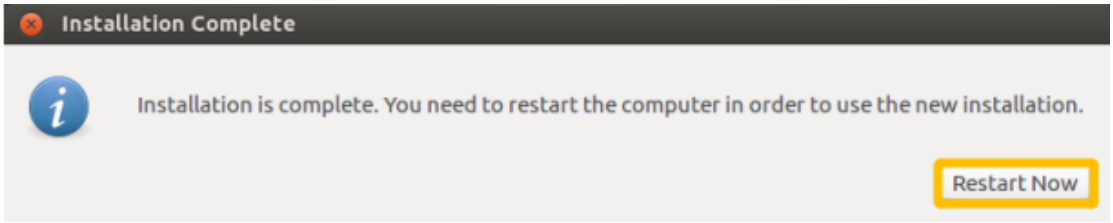
8) 输入用户名和密码（填入用户名过程中， Computer name 和 Username 会自动生成，为简单起见，这普通用户只设置了 1 位密码。读者可根据自己的需

要，进行设置），点击 Continue。

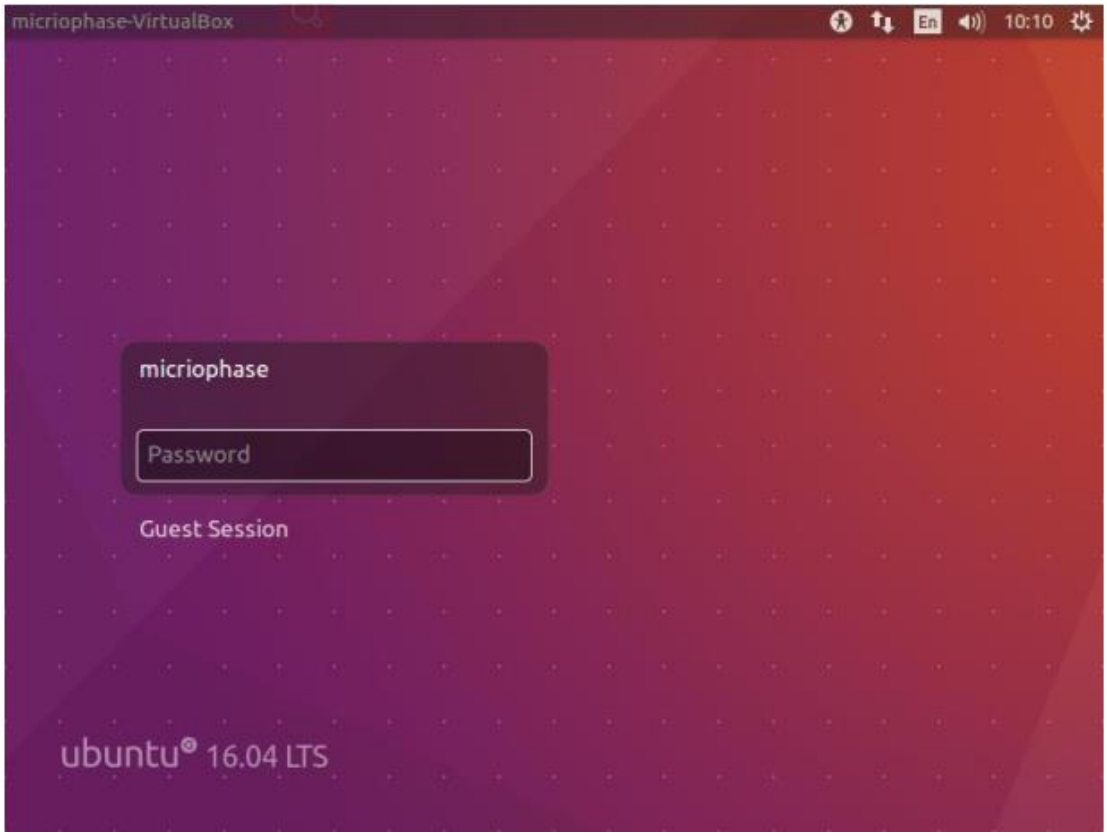


9) Ubuntu16.04 开始安装（安装时间较长，请耐心等待）。

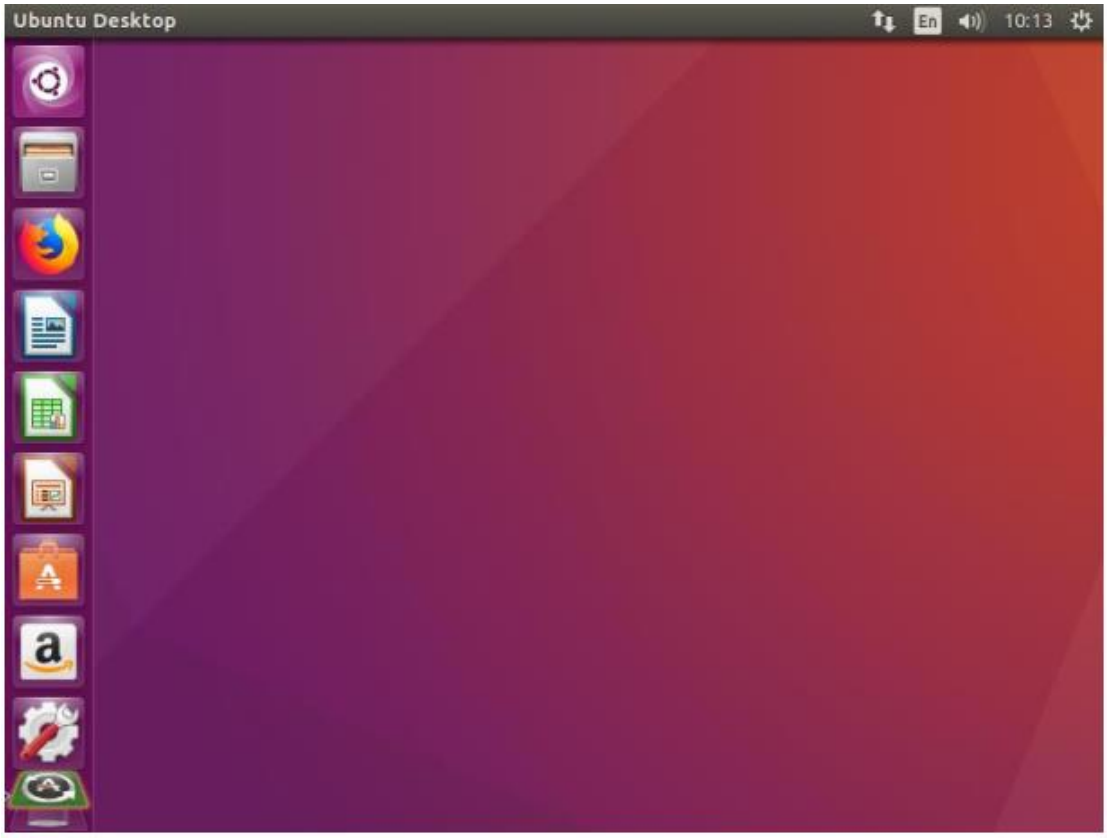
10) 安装完成，点击 Restart Now。



11) 进入系统，输入密码。

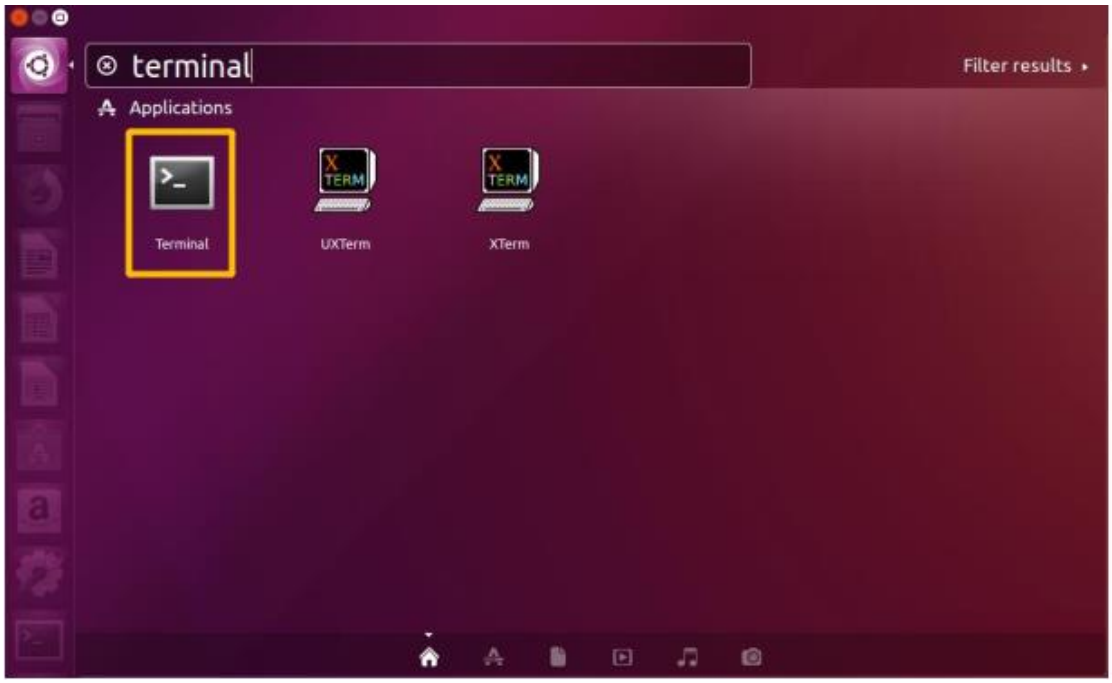


12) 系统如图所示。



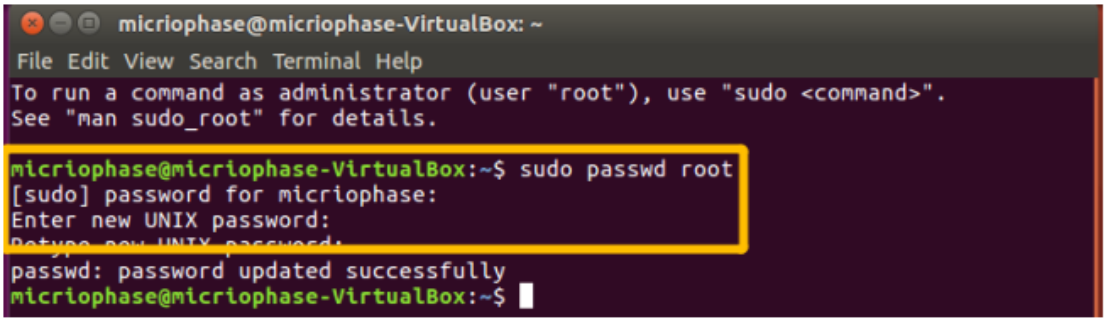
1.5 配置 Ubuntu 系统

1) 点击搜索，在命令行输入 terminal，点击 Terminal。

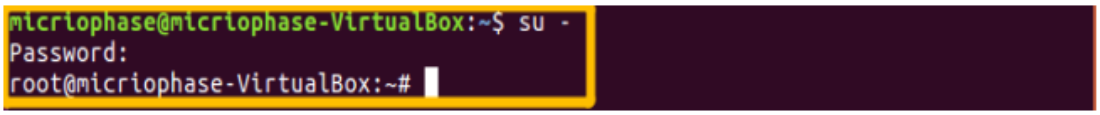


2) 打开 Termianl 后，为了以后方便使用，右键单击 Termianl， 选择 Lock to Launcher，锁定在工具栏上。

3) 创建 root 用户，Terimanl 钟输入命令 sudo passwd root，首先输入普通用户 micriophase 密码，再输入两次设置的 root 密码，设置完成（请读者牢记 root 密码，忘记后比较麻烦）。



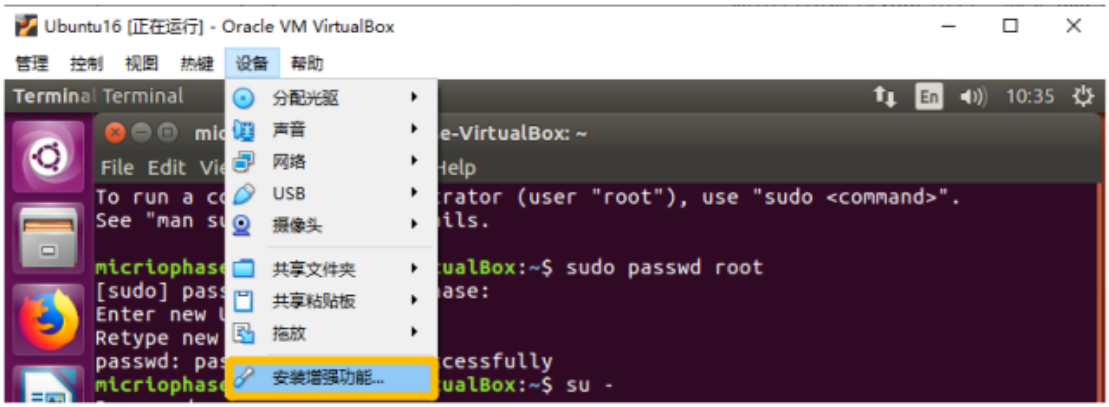
4) 测试 root 是否生效，输入命令 su - 进入 root 用户根目录，提示符从$变成#，表示从普通用户切换到 root 用户，输入 exit 退出 root 用户。



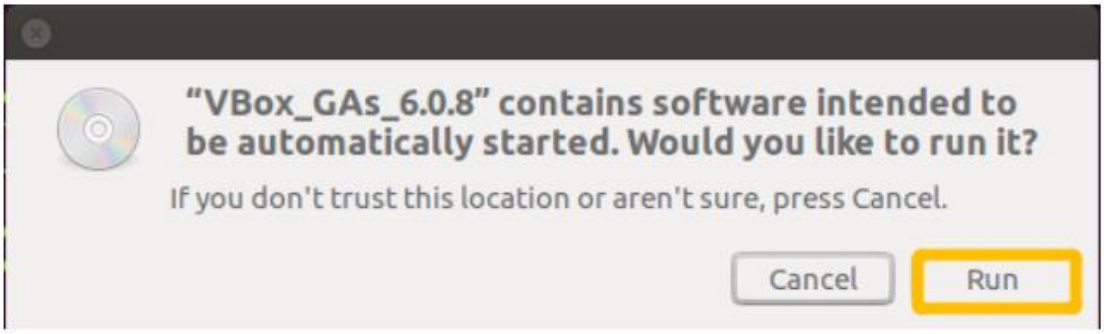
5) 测试共享文件夹。首先输入 ls /mnt， 查看/mnt 目录中由什么内容，如图所示为空。

在 Windows 下的共享文件夹 VM_share 中创建一个测试文件 test.txt。 在

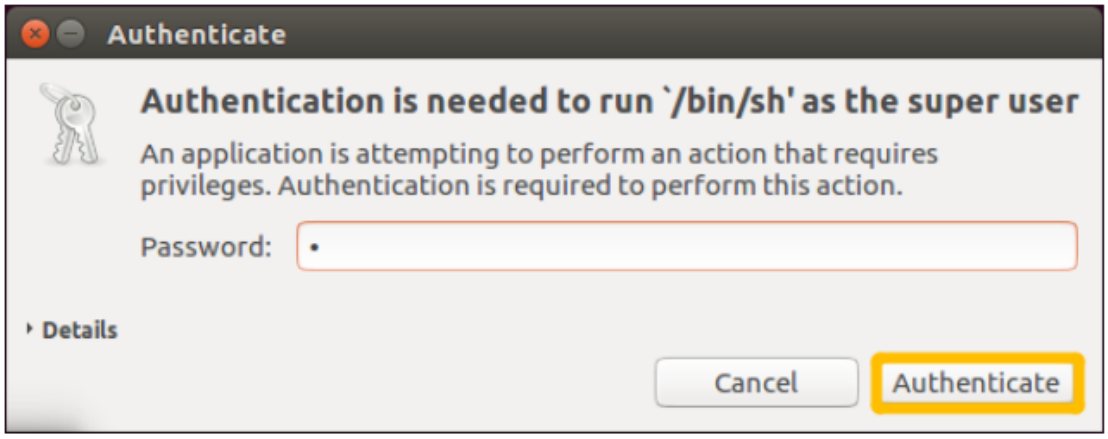
Linux 下，首先进行配置，使其支持 vboxsf 类型的挂载（只配置一次即可），点击 VirtualBox 菜单栏设置-->安装增强功能，



点击 Run，



输入密码，点击 Authenticate，等待安装完成。安装完成后重启 Ubuntu。

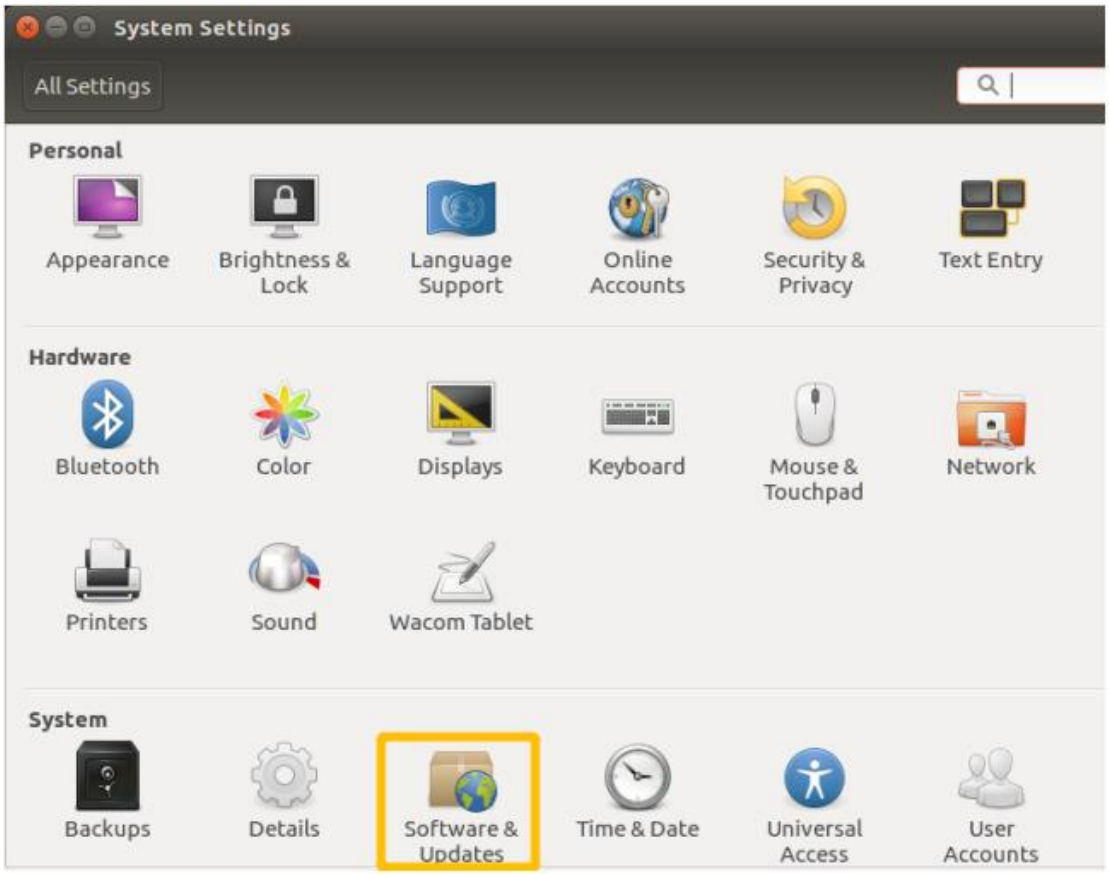


6) 开始挂载。输入命令 sudo mount –t vboxsf VM_share /mnt 将共享文件夹 VM_share 挂载到目录/mnt 下，再使用命令 ls /mnt 查看，可见 test.txt 已经再 /mnt 中，说明挂载成功。（注意：每次重启 Linux 系统后，如果需要使用共享文件夹 VM_share， 都要进行挂载才可以使用）。

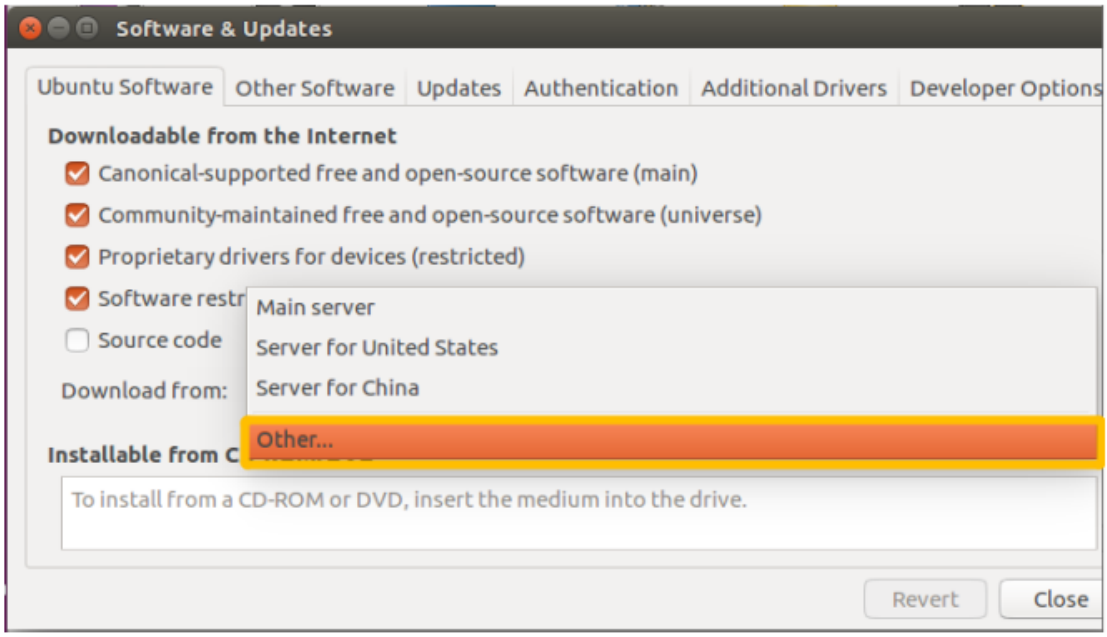
```
micriophase@micriophase-VirtualBox:~$ sudo mount -t vboxsf VM_share /mnt
micriophase@micriophase-VirtualBox:~$ ls /mnt
petalinux_installation_log
petalinux-v2018.3-final-installer.run
test.txt
vivado_lic2037.lic
Xilinx Vivado SDK 2018.3 1207 2324
Xilinx Vivado SDK 2018.3 1207 2324.tar.gz
```

7) 修改软件源服务器。为了方便后期的软件安装，我们需要设置软件源，点击系统设置，

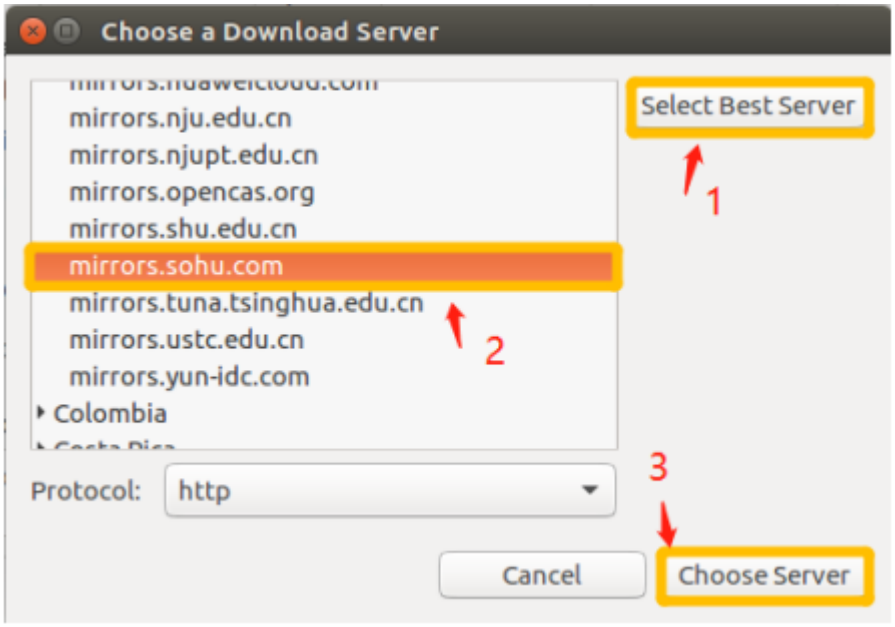
选择 Software&Updates，



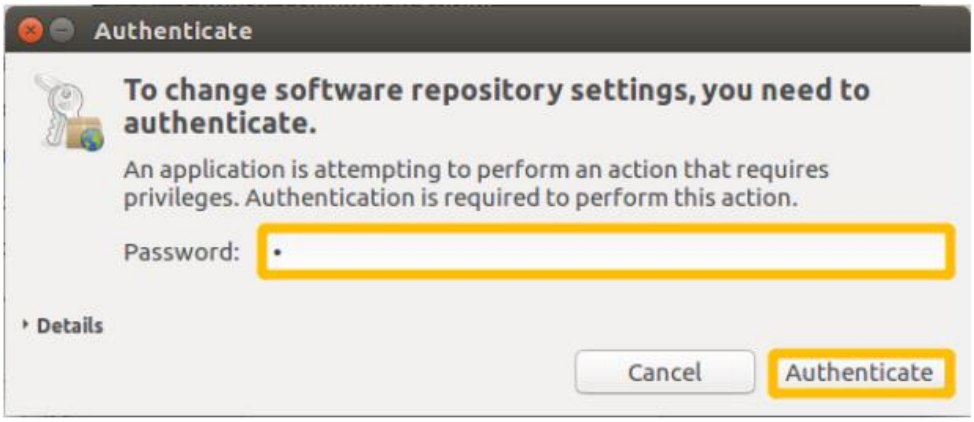
点击 Download from 选项的 Other，



点击 Select Best Server， 系统会选择最适合的镜像网址，点击 Chosose Server，



输入密码，点击 Authenticate，

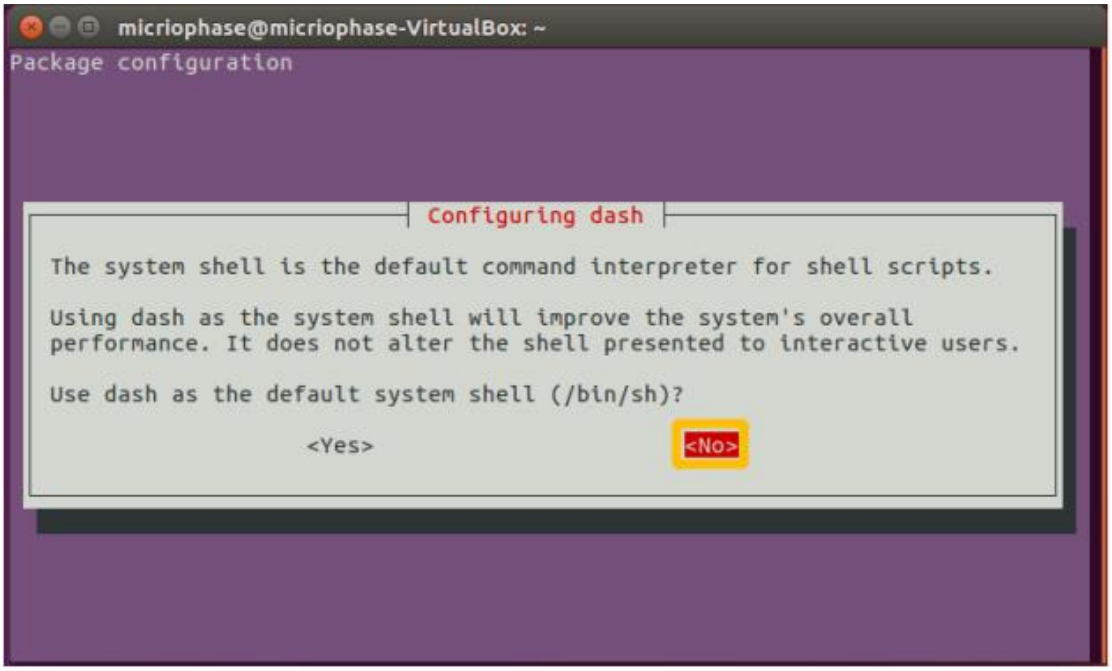


点击 Close。

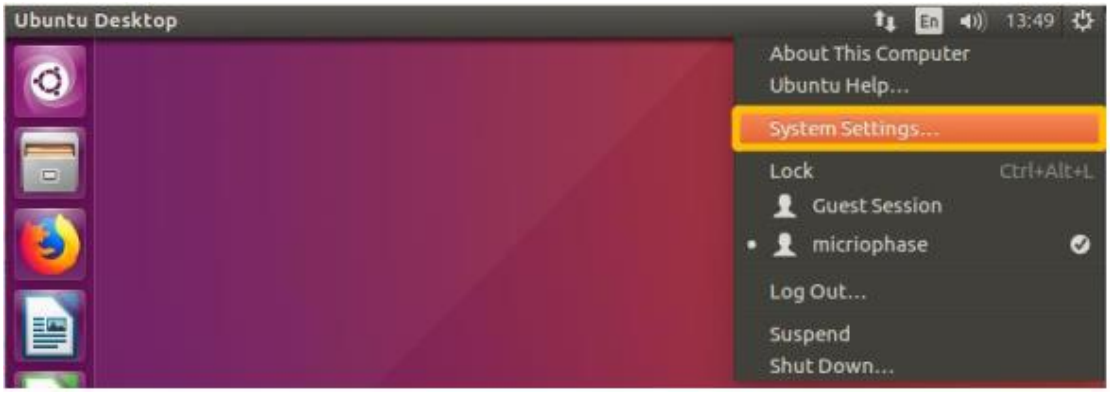
1.6 设置/bin/sh 为 bash

1) 打开 Teriminal，输入命令 sudo dpkg-reconfigure dash。

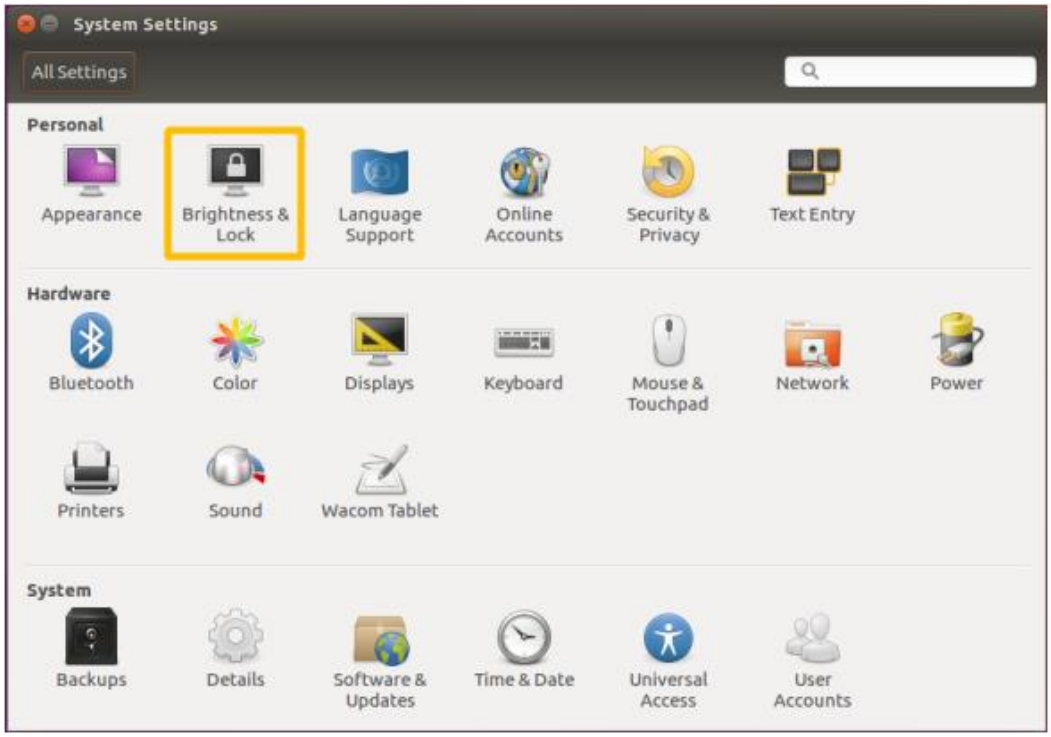
2) 选择 No。



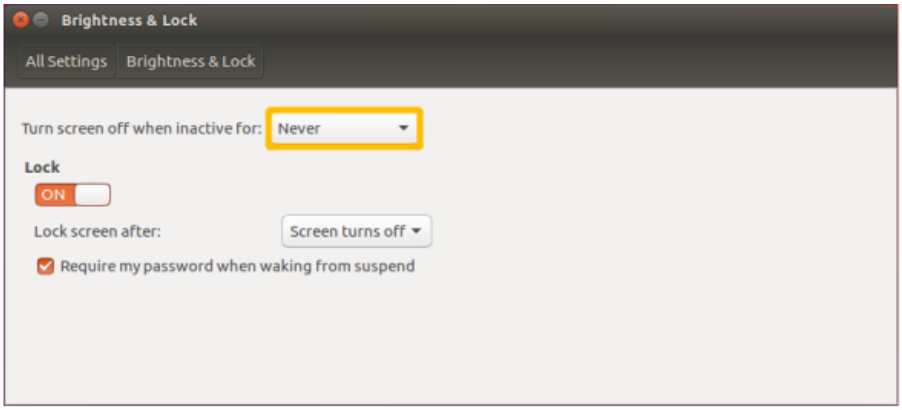
3) 设定锁屏时间。为了可以复制大文件至 Ubuntu 系统，取消屏幕锁定时间。点击系统设定。



4) 选择 Brightness&Lock。



5) Turn screen off when inactive for 选择 Never。



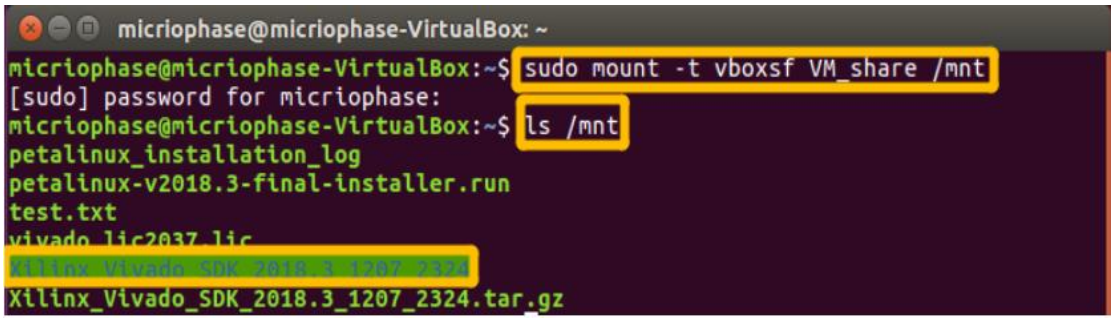
第2章　Ubuntu 系统下 Linux 版 Vivado 安装

虽然很多读者已经习惯于在 Windows 下使用 Vivado,但是在开发 ZYNQ 系统时，很多应用程序需要在 Linux 系统下进行交叉编译，这样就需要开发一套程序可以同时在 Linux 和 Windows 下进行，这种情况下，某些读者更喜欢在同一个系统下完成，那么安装 Linux 版的 Vivado 就十分必要了。

2.1 安装 Linux 版 Vivado

1) 将 Vivado 安装软件复制到 VM_share 文件夹下。

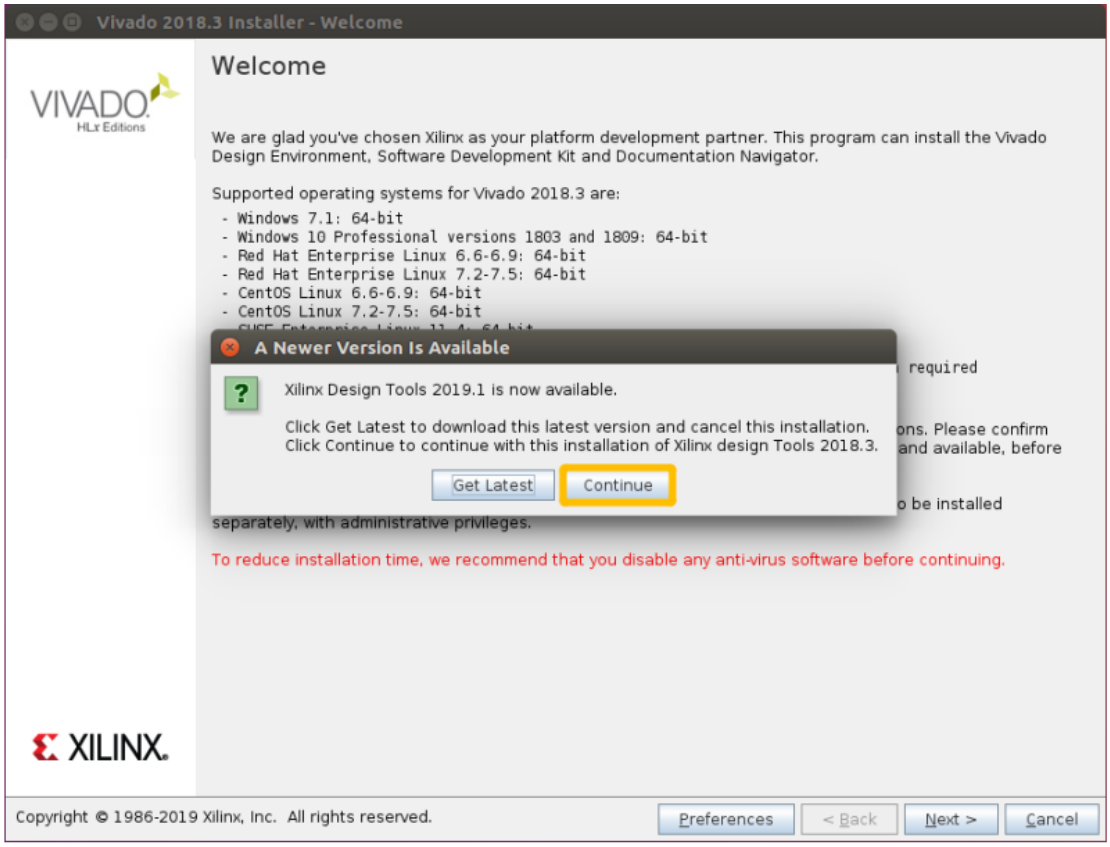
2) 输入命令 sudo mount –t vboxsf VM_share /mnt 将共享文件夹 VM_share 挂载到目录/mnt 下，输入命令 ls /mnt 查看文件夹内容，可以看到 Vivado 安装文件 Xilinx_Vivado_SDK_2018.3_1207_2324 已经存在。



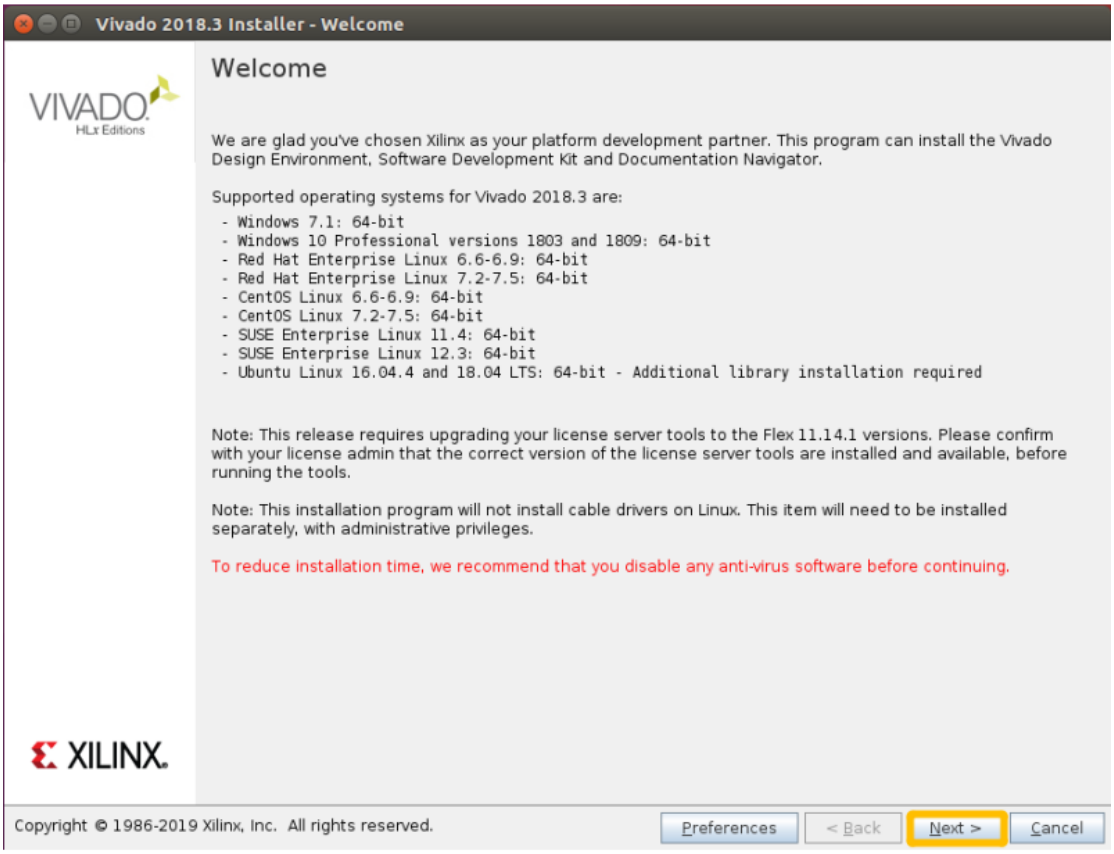
3) 进入安装包文件夹 Xilinx_Vivado_SDK_2018.3_1207_2324。

4) 输入命令 sudo chmod +x xsetup 给安装软件执行权限，输入命令 sudo ./xsetup。

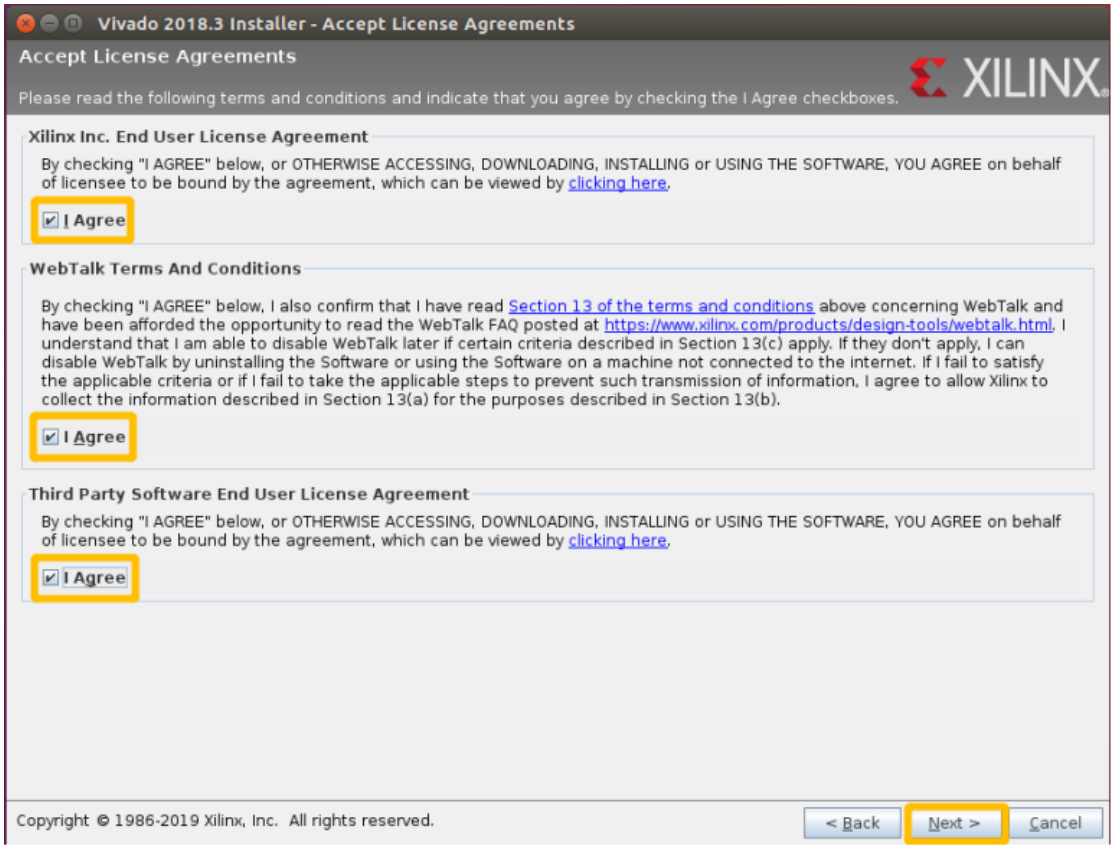
5) 弹出对话框，点击 Continue，选择当前版本。


Vivado 2018.3 Installer - Welcome
VIVADO HLx Editions
Welcome
We are glad you've chosen Xilinx as your platform development partner. This program can install the Vivado Design Environment, Software Development Kit and Documentation Navigator.
Supported operating systems for Vivado 2018.3 are:
- Windows 7.1: 64-bit
- Windows 10 Professional versions 1803 and 1809: 64-bit
- Red Hat Enterprise Linux 6.6-6.9: 64-bit
- Red Hat Enterprise Linux 7.2-7.5: 64-bit
- CentOS Linux 6.6-6.9: 64-bit
- CentOS Linux 7.2-7.5: 64-bit
A Newer Version Is Available
Xilinx Design Tools 2019.1 is now available.
Click Get Latest to download this latest version and cancel this installation.
Click Continue to continue with this installation of Xilinx design Tools 2018.3.
Get Latest
Continue
separately, with administrative privileges.
To reduce installation time, we recommend that you disable any anti-virus software before continuing.
XILINX
Copyright © 1986-2019 Xilinx, Inc. All rights reserved.
Preferences
< Back
Next >
Cancel

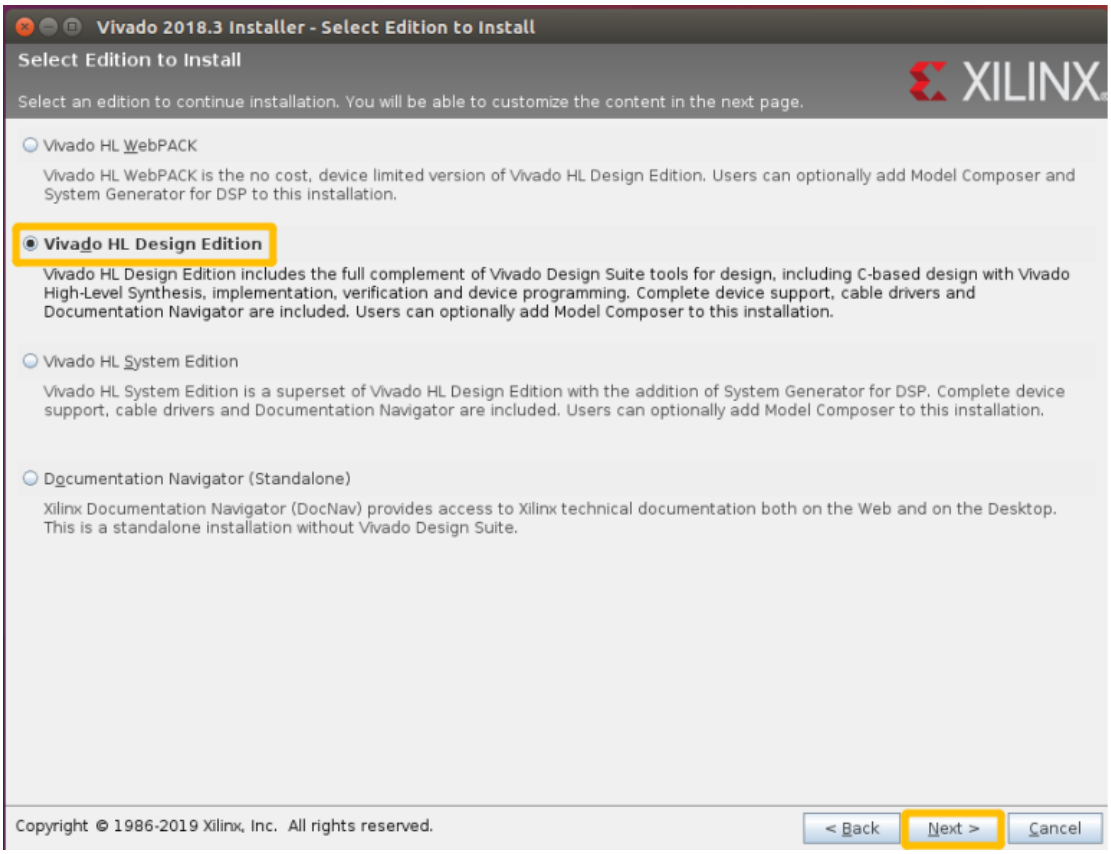
6) 点击 Next。


Vivado 2018.3 Installer - Welcome
VIVADO HLx Editions
Welcome
We are glad you've chosen Xilinx as your platform development partner. This program can install the Vivado Design Environment, Software Development Kit and Documentation Navigator.
Supported operating systems for Vivado 2018.3 are:
- Windows 7.1: 64-bit
- Windows 10 Professional versions 1803 and 1809: 64-bit
- Red Hat Enterprise Linux 6.6-6.9: 64-bit
- Red Hat Enterprise Linux 7.2-7.5: 64-bit
- CentOS Linux 6.6-6.9: 64-bit
- CentOS Linux 7.2-7.5: 64-bit
- SUSE Enterprise Linux 11.4: 64-bit
- SUSE Enterprise Linux 12.3: 64-bit
- Ubuntu Linux 16.04.4 and 18.04 LTS: 64-bit - Additional library installation required
Note: This release requires upgrading your license server tools to the Flex 11.14.1 versions. Please confirm with your license admin that the correct version of the license server tools are installed and available, before running the tools.
Note: This installation program will not install cable drivers on Linux. This item will need to be installed separately, with administrative privileges.
To reduce installation time, we recommend that you disable any anti-virus software before continuing.
XILINX
Copyright © 1986-2019 Xilinx, Inc. All rights reserved.
Preferences
< Back
Next >
Cancel

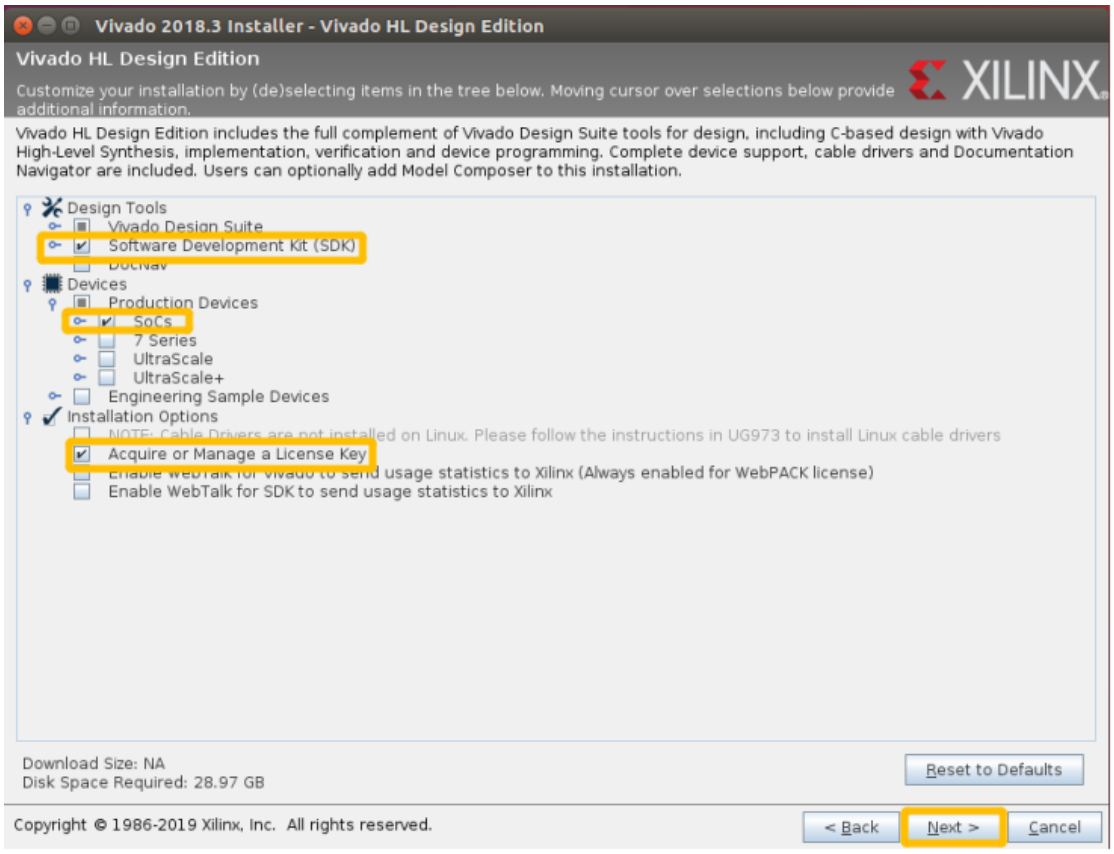
7) 选中三个 I Agree，点击 Next。



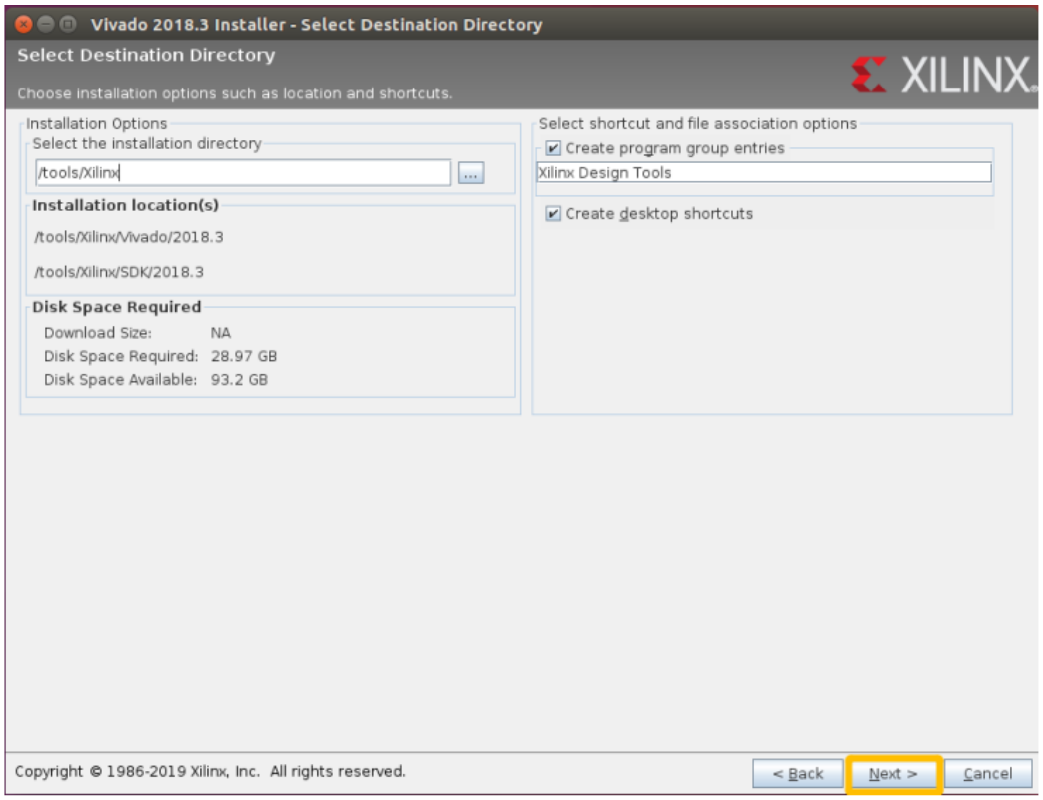
8) 选择 Vivado HL Design Edition，点击 Next。



9) 勾选 Design Tools 的 Software Development Kit(SDK)， Devices 的 SoCs，Installation Options 的 Acquire or Manage a License Key，点击 Next。

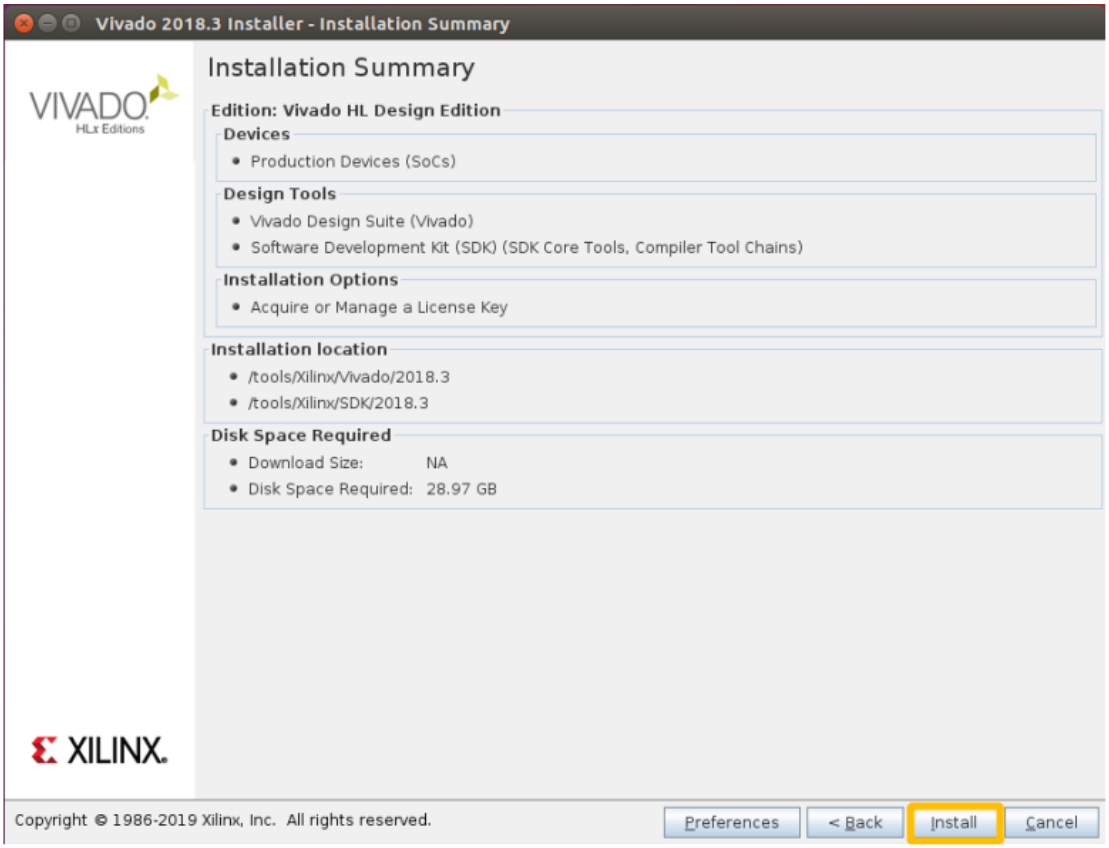


10) 点击 Next。

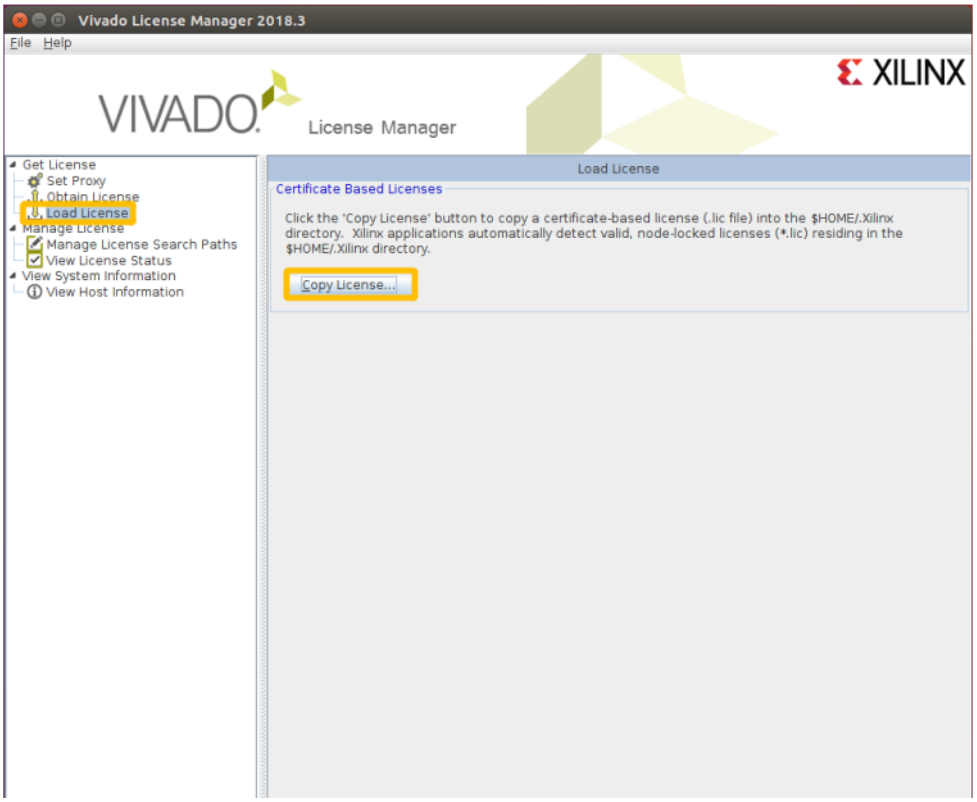


11) 点击 Yes。

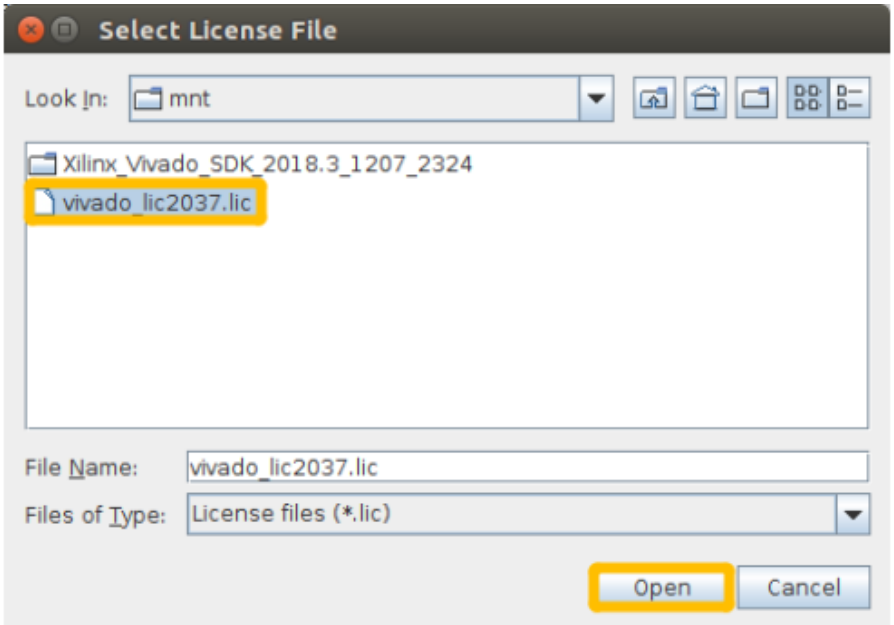
12) 点击 Install。



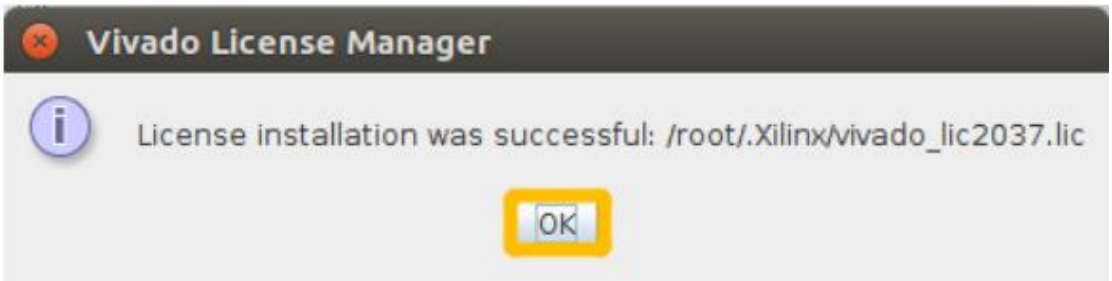
13) 安装完成后，点击 Load License，点击 Copy License。



14) 选中 vivado_lic2037.lic，点击 Open。



15) 点击 OK。



2.2 添加权限

1) 运行命令 sudo chmod 777 –R/tools/Xilinx， sudo chmod 777 -R ~/.Xilinx/。

```
micriophase@micriophase-VirtualBox:~$ sudo chmod 777 -R /tools/Xilinx/
micriophase@micriophase-VirtualBox:~$ sudo chmod 777 -R ~/.Xilinx/
micriophase@micriophase-VirtualBox:~$
```

2.3 安装下载器驱动

1) 进入驱动文件路径，输入 cd /tools/Xilinx/Vivado/2018.3/data/xicom /cable_drivers /lin64/install_script/install_drivers，运行命令 sudo ./install_drivers 安装下载器驱动。

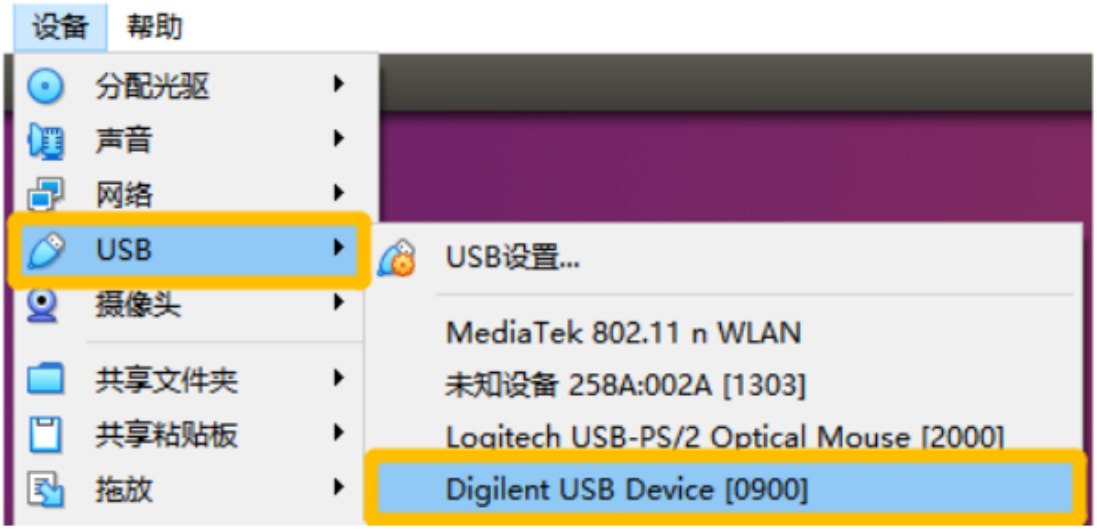
```
micriophase@micriophase-VirtualBox:~$ cd /tools/Xilinx/Vivado/2018.3/data/xicom/
cable_drivers/lin64/install_script/install_drivers
micriophase@micriophase-VirtualBox:/tools/Xilinx/Vivado/2018.3/data/xicom/cable_
drivers/lin64/install_script/install_drivers$ sudo ./install_drivers
INFO: Installing cable drivers.
INFO: Script name = ./install_drivers
INFO: HostName = micriophase-VirtualBox
INFO: Current working dir = /tools/Xilinx/Vivado/2018.3/data/xicom/cable_drivers
/lin64/install_script/install_drivers
INFO: Kernel version = 4.15.0-29-generic.
INFO: Arch = x86_64.
Successfully installed Digilent Cable Drivers
--File /etc/udev/rules.d/52-xilinx-ftdi-usb.rules does not exist.
--File version of /etc/udev/rules.d/52-xilinx-ftdi-usb.rules = 0000.
--Updating rules file.
--File /etc/udev/rules.d/52-xilinx-pcusb.rules does not exist.
--File version of /etc/udev/rules.d/52-xilinx-pcusb.rules = 0000.
--Updating rules file.

INFO: Digilent Return code = 0
INFO: Xilinx Return code = 0
INFO: Xilinx FTDI Return code = 0
INFO: Return code = 0
INFO: Driver installation successful.
CRITICAL WARNING: Cable(s) on the system must be unplugged then plugged back in
order for the driver scripts to update the cables.
```

2.4 测试下载器驱动

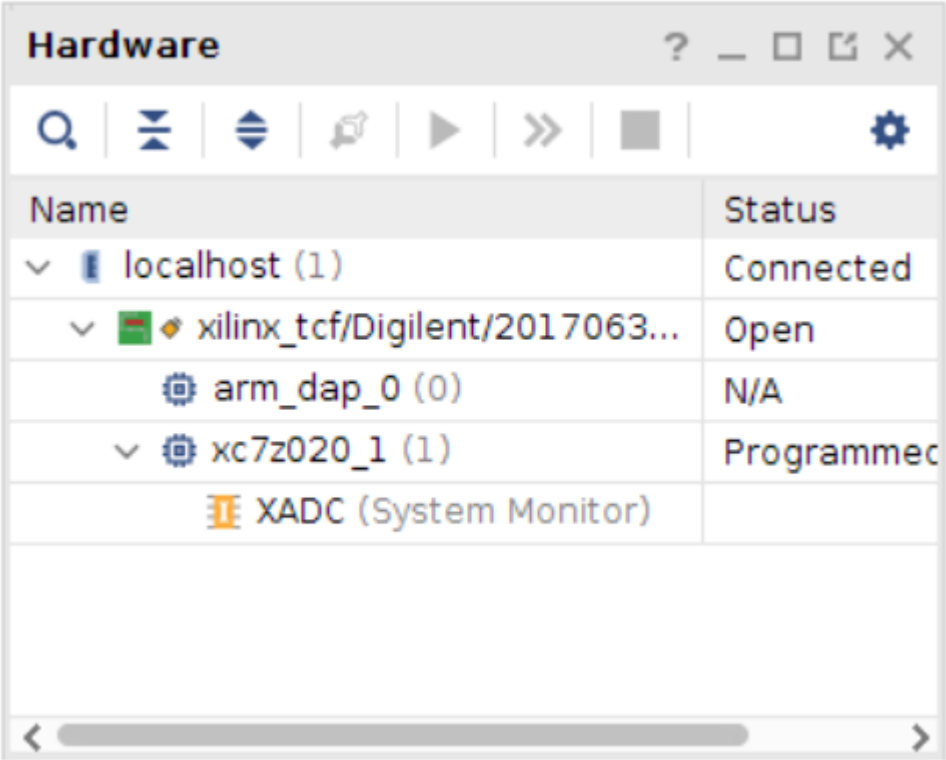
1) 运行命令 source /tools/Xilinx/Vivado/2018.3/settings64.sh， vivado &启动 Vivado。

```
micriophase@micriophase-VirtualBox:/tools/Xilinx/Vivado/2018.3/data/xicom/cable_
drivers/lin64/install_script/install_drivers$ source /tools/Xilinx/Vivado/2018.3
/settings64.sh
micriophase@micriophase-VirtualBox:/tools/Xilinx/Vivado/2018.3/data/xicom/cable_
drivers/lin64/install_script/install_drivers$ vivado &
```

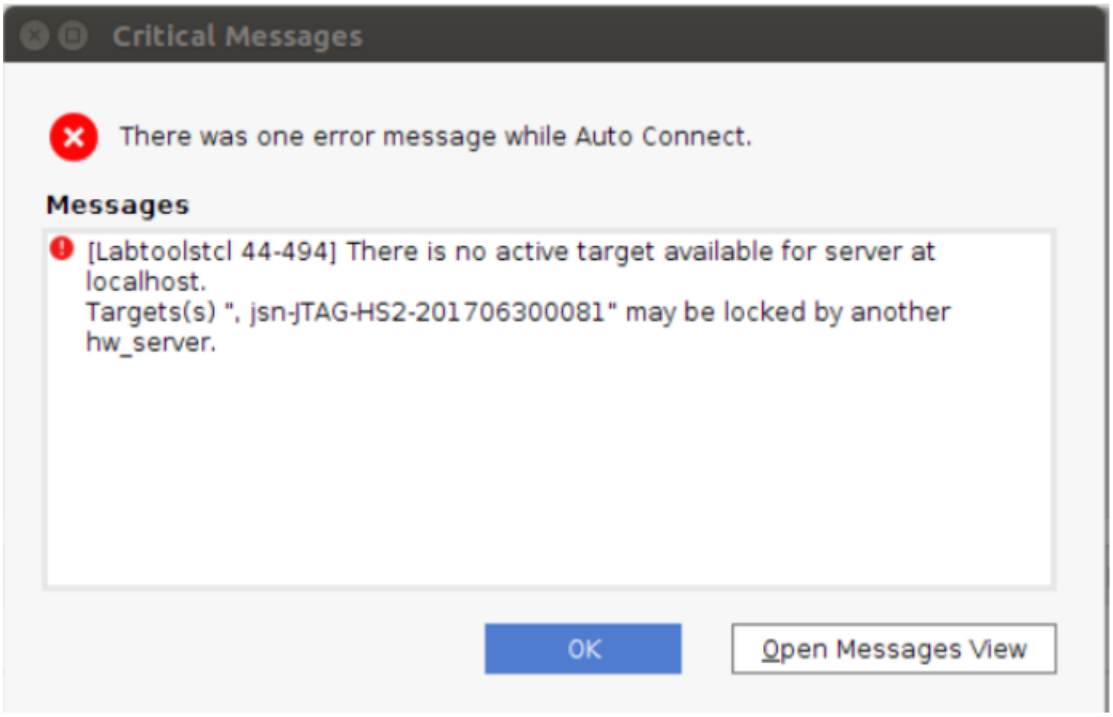
2) 连接下载器至虚拟机。



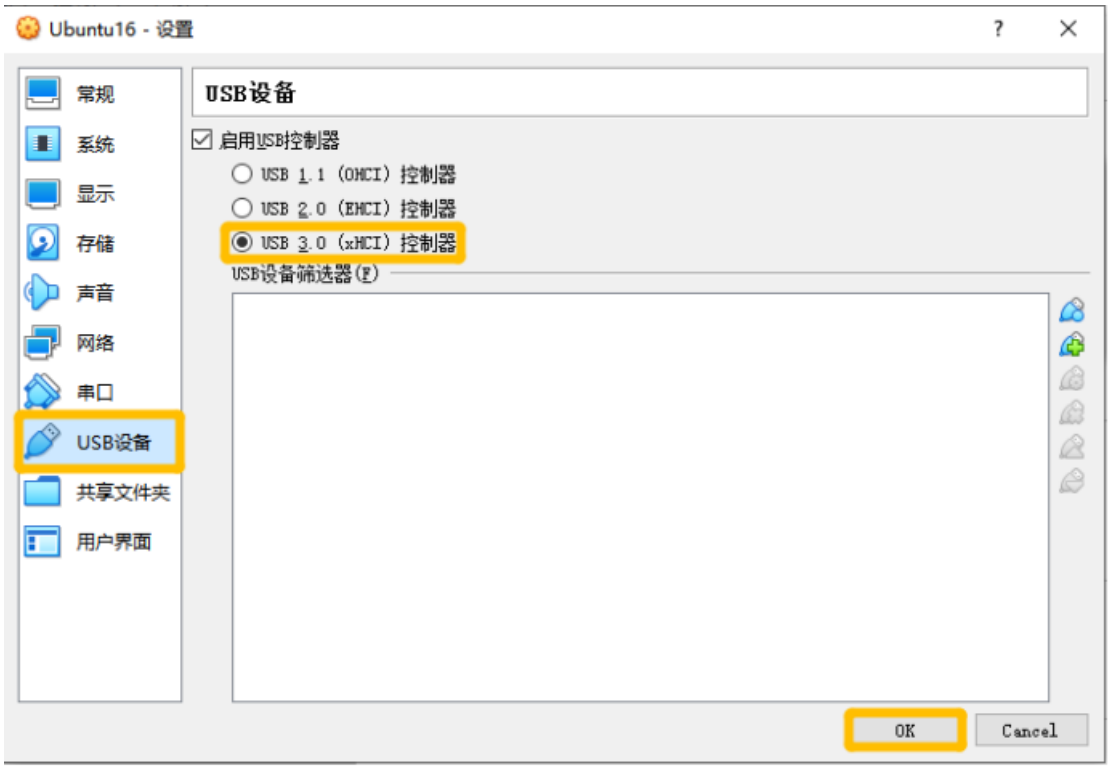
3) 连接开发板和下载器，点击 Vivado 软件界面的“Open Hadrware Manager”，连接板卡，发现芯片，说明安装成功。



4) 某些情况下，可能会出现一下错误，可以发现下载器，但显示被占用不能使用。



5) 需要在 VirtualBox 中修改 USB 控制器选项，先关闭 Ubuntu 系统，点击 VirtualBox 设置，左侧选择“USB 设备”，点击 USB3.0(xHCI)控制器，进入 Ubuntu 系统尝试，如果依然不可以，建议在 Windows 版本下使用 Vivado。



第3章 PetaLinux 安装

PetaLinux 是一个嵌入式 Linux 系统开发工具包，用于 XilinxFPGA 片上系统的定制、建立、测试和部署嵌入式 Linux 系统。

3.1 安装依赖库

1) 运行运行下面命令，安装依赖库。（安装时需要注意，安装库时需要用到 root 权限，安装软件时必须是普通用户，普通目录下）。

```
sudo -s
sudo apt-get install –y gcc git make net-tools libncurses5-dev tftpd zlib1g-dev libssl-dev flex bison libselinux1 gnupg wget diffstat chrpath socat xterm autoconf libtool tar unzip texinfo zlib1g-dev gcc-multilib build-essential libsdl1.2-dev libglib2.0-dev zlib1g:i386 screen pax gzip gawk
```

```
micriophase@micriophase-VirtualBox:~$ sudo -s
[sudo] password for micriophase:
```

```
root@micriophase-VirtualBox:~# sudo apt-get install -y gcc git make net-tools li
bncurses5-dev tftpd zlib1g-dev libssl-dev flex bison libselinux1 gnupg wget diff
stat chrpath socat xterm autoconf libtool tar unzip texinfo zlib1g-dev gcc-multi
lib build-essential libsdl1.2-dev libglib2.0-dev zlib1g:i386 screen pax gzip gaw
k
```

2) 运行命令下列命令，安装 tftp Server，中间会询问 Do you want to continue?[Y/n]，输入 y。

```
apt-get install tftpd-hpa
```

```
root@micriophase-VirtualBox:~# apt-get install tftpd-hpa
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  xinetd
Use 'sudo apt autoremove' to remove it.
Suggested packages:
  pxelinux
The following packages will be REMOVED:
  tftpd
The following NEW packages will be installed:
  tftpd-hpa
0 upgraded, 1 newly installed, 1 to remove and 384 not upgraded.
Need to get 39.1 kB of archives.
After this operation, 32.0 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ftp.sjtu.edu.cn/ubuntu xenial-updates/main amd64 tftpd-hpa amd64 5.
2+20150808-1ubuntu1.16.04.1 [39.1 kB]
Fetched 39.1 kB in 0s (383 kB/s)
Preconfiguring packages ...
(Reading database ... 185908 files and directories currently installed.)
Removing tftpd (0.17-18ubuntu2) ...
Processing triggers for man-db (2.7.5-1) ...
Selecting previously unselected package tftpd-hpa.
(Reading database ... 185902 files and directories currently installed.)
Preparing to unpack .../tftpd-hpa_5.2+20150808-1ubuntu1.16.04.1_amd64.deb ...
Unpacking tftpd-hpa (5.2+20150808-1ubuntu1.16.04.1) ...
Processing triggers for ureadahead (0.100.0-19) ...
Processing triggers for systemd (229-4ubuntu21.4) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up tftpd-hpa (5.2+20150808-1ubuntu1.16.04.1) ...
```

3) 运行下列命令，启动 tft Server，查看状态。

```
service tftpd-hpa restart
Service tftpd-hpa status
```

```
root@micriophase-VirtualBox:~# service tftpd-hpa restart
root@micriophase-VirtualBox:~# service tftpd-hpa status
● tftpd-hpa.service - LSB: HPA's tftp server
   Loaded: loaded (/etc/init.d/tftpd-hpa; bad; vendor preset: enabled)
   Active: active (running) since 一 2019-09-16 17:00:11 CST; 6s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 17334 ExecStop=/etc/init.d/tftpd-hpa stop (code=exited, status=0/SUCC
  Process: 17347 ExecStart=/etc/init.d/tftpd-hpa start (code=exited, status=0/SU
   CGroup: /system.slice/tftpd-hpa.service
           └─17363 /usr/sbin/in.tftpd --listen --user tftp --address :69 --secur

9月 16 17:00:11 micriophase-VirtualBox systemd[1]: Starting LSB: HPA's tftp serv
9月 16 17:00:11 micriophase-VirtualBox tftpd-hpa[17347]:  * Starting HPA's tftpd
9月 16 17:00:11 micriophase-VirtualBox tftpd-hpa[17347]:    ...done.
9月 16 17:00:11 micriophase-VirtualBox systemd[1]: Started LSB: HPA's tftp serve
```

按 Ctrol+Z 停止

4) 输入下列命令，校验 tftp。

```
ps ax | grep tftpd
```

```
root@micriophase-VirtualBox:~# ps ax | grep tftpd
17363 ?        Ss     0:00 /usr/sbin/in.tftpd --listen --user tftp --address :69
 --secure /var/lib/tftpboot
17364 pts/11   T      0:00 systemctl status tftpd-hpa.service
17384 pts/11   S+     0:00 grep --color=auto tftpd
root@micriophase-VirtualBox:~#
```

3.2 安装 Petalinux

1) 运行运行下列命令做安装准备。

```
sudo –s
mkdir –p /opt/pkg/petalinux/2018.3
chown <your_user_name> /opt/pkg/
chown <your_user_name> /opt/pkg/petalinux/
chown <your_user_name> /opt/pkg/petalinux/2018.3/
chgrp <your_user_name> /opt/pkg/
chgrp <your_user_name> /opt/pkg/petalinux/
chgrp <your_user_name> /opt/pkg/petalinux/2018.3/
exit
```

```
root@micriophase-VirtualBox:~# sudo -s
root@micriophase-VirtualBox:~# mkdir -p /opt/pkg/petalinux/2018.3
root@micriophase-VirtualBox:~# chown micriophase /opt/pkg/
root@micriophase-VirtualBox:~# chown micriophase /opt/pkg/petalinux/
root@micriophase-VirtualBox:~# chown micriophase /opt/pkg/petalinux/2018.3
root@micriophase-VirtualBox:~# charp micriophase /opt/pkg/
```

```
root@micriophase-VirtualBox:~# chgrp micriophase /opt/pkg/
root@micriophase-VirtualBox:~# chgrp micriophase /opt/pkg/petalinux/
root@micriophase-VirtualBox:~# chgrp micriophase /opt/pkg/petalinux/2018.3/
root@micriophase-VirtualBox:~# exit
```

2) 将安装文件复制到共享文件夹，进入安装文件所在目录，运行下列命令给安装文件添加运行权限。

```
sudo chmod +x petalinux-v2018.3-final-installer.run
```

```
micriophase@micriophase-VirtualBox:/mnt$ sudo chmod +x petalinux-v2018.3-final-i
nstaller.run
```

3) 输入下列命令，安装 petalinux。

```
./petalinux-v2018.3-final-installer.run /opt/pkg/petalinux/2018.3/
```

```
microphase@microphase-VirtualBox:/mnt$ ./petalinux-v2018.3-final-installer.run /opt/pkg/petalinux/2018.3/
```

4) 按回车查看协议，按 q 退出协议内容。

5) 按 y 同意协议。重复三次。

```
LICENSE AGREEMENTS

PetaLinux SDK contains software from a number of sources.  Please review
the following licenses and indicate your acceptance of each to continue.

You do not have to accept the licenses, however if you do not then you may
not use PetaLinux SDK.

Use PgUp/PgDn to navigate the license viewer, and press 'q' to close

Press Enter to display the license agreementsDo you accept Xilinx End User Licen
se Agreement? [y/N] > y
Do you accept Webtalk Terms and Conditions? [y/N] > y
Do you accept Third Party End User License Agreement? [y/N] >
Do you accept Third Party End User License Agreement? [y/N] > y
INFO: Installing PetaLinux...
```

6) 安装完成后，输入下列命令，验证安装的正确性。

```
source /opt/pkg/petalinux/2018.3/settings.sh
echo $PETALINUX
```

```
micriophase@micriophase-VirtualBox:~$ source /opt/pkg/petalinux/2018.3/settings.
sh
PetaLinux environment set to '/opt/pkg/petalinux/2018.3'
INFO: Checking free disk space
INFO: Checking installed tools
INFO: Checking installed development libraries
INFO: Checking network and other services
micriophase@micriophase-VirtualBox:~$ echo $PETALINUX
/opt/pkg/petalinux/2018.3
```

第4章　NFS 服务软件安装

NFS(Network File System)即网络文件系统，是 FreeBSD 支持的文件系统中的一种，允许网络中的计算机之间共享资源。NSF 服务器可以允许 NFS 客户端将远端 NFS 服务器端的共享目录挂载到本地的 NSF 客户端中，在本地的 NFS 客户端的机器看来，NFS 服务器端共享的目录类似自己的磁盘分区和目录。一般情况下，客户端挂载到本地目录的名字可以随便，但为了方便管理，和服务器端保持一致比较好。

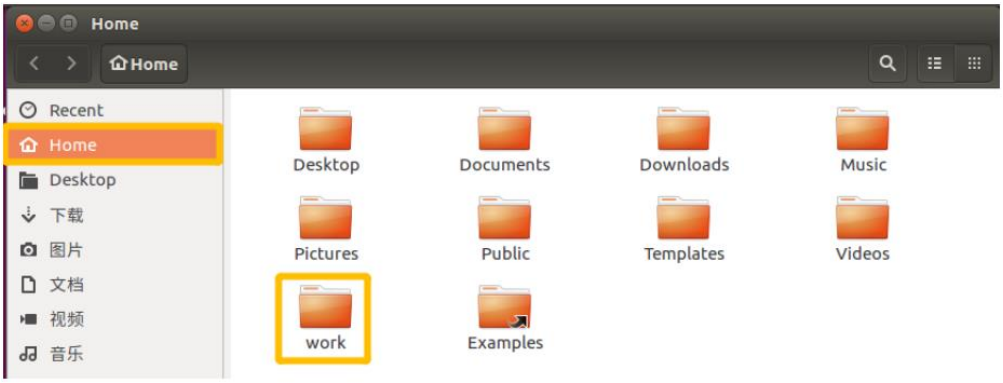
4.1 安装 NFS 服务

1) 运行下列命令，安装 NFS 服务器，当系统询问是否继续时，输入 Y。

sudo apt-get install nfs-kernel-server

```
micriophase@micriophase-VirtualBox:~$ sudo apt-get install nfs-kernel-server
[sudo] password for micriophase:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  xinetd
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  keyutils libnfsidmap2 libtirpc1 nfs-common rpcbind
Suggested packages:
  open-iscsi watchdog
The following NEW packages will be installed:
  keyutils libnfsidmap2 libtirpc1 nfs-common nfs-kernel-server rpcbind
0 upgraded, 6 newly installed, 0 to remove and 389 not upgraded.
Need to get 467 kB of archives.
After this operation, 1,846 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

2) 在 Home 下，新建目录 work 作为 NFS 工作目录，以后进行程序开发时，可以将交叉编译的程序放在此目录中，开发板可以更方便的共享此目录中的文件。



3) 输入下列命令编辑/etc/exports 文件，配置 NFS 服务路径。

```
sudo gedit /etc/exports
```

```
micriophase@micriophase-VirtualBox:~$ sudo gedit /etc/exports
```

4) 在尾部添加 /home/micriophase/work *(rw,sync,no_root_squash, no_subtree_check)，配置/home/micriophse/work 目录为 NFS 的一个工作目录。

```
Open ▾   *exports   Save
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
#
/home/micriophase/work *(rw,sync,no_root_squash,no_subtree_check)
```

5) 执行下列命令重启 rpcbind 服务。NFS 是一个 RPC 程序，使用之前需要映射端口，通过 rpcbind 设定。

```
sudo /etc/init.d/rpcbind restart
```

```
micriophase@micriophase-VirtualBox:~$ sudo /etc/init.d/rpcbind restart
[ ok ] Restarting rpcbind (via systemctl): rpcbind.service.
```

6) 执行下列命令，重启 NFS 服务。

```
sudo /etc/init.d/nfs-kernel-server restart
```

```
micriophase@micriophase-VirtualBox:~$ sudo /etc/init.d/nfs-kernel-server restart
[ ok ] Restarting nfs-kernel-server (via systemctl): nfs-kernel-server.service.
```

4.2 测试 NFS

1) 执行下列命令挂载 NFS，在本机将 NFS 目录挂载在/mnt 目录下。

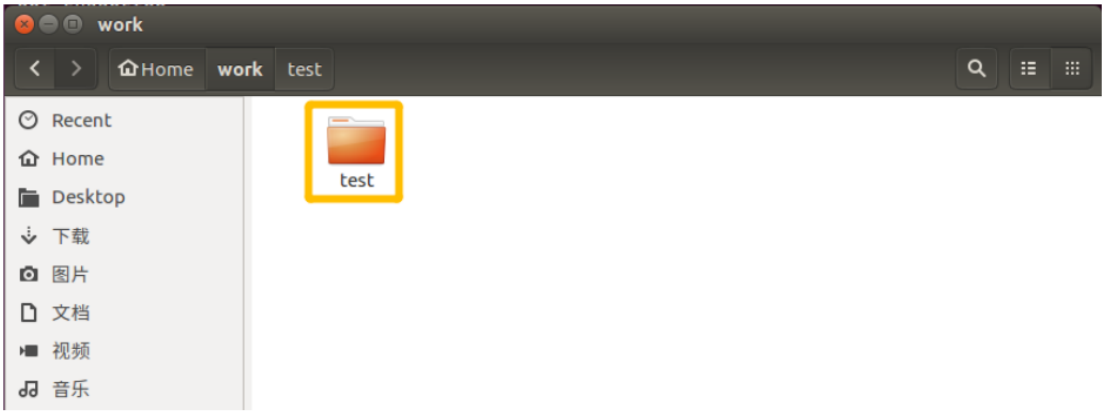
```
mount –t nfs 127.0.0.1:/home/micriophase/work /mnt
```

```
micriophase@micriophase-VirtualBox:~$ su -
Password:
root@micriophase-VirtualBox:~# mount -t nfs 127.0.0.1:/home/micriophase/work /mnt
root@micriophase-VirtualBox:~# exit
```

2) 进入/mnt，新建 test 目录进行测试，可以在/home/micriophase/work 目录同步看到 test 文件夹。

```
cd /mnt

mkdir test
```

```
micriophase@micriophase-VirtualBox:~$ cd /mnt
micriophase@micriophase-VirtualBox:/mnt$ mkdir test
```


work
Home
work
test
Recent
Home
Desktop
下载
图片
文档
视频
音乐
test

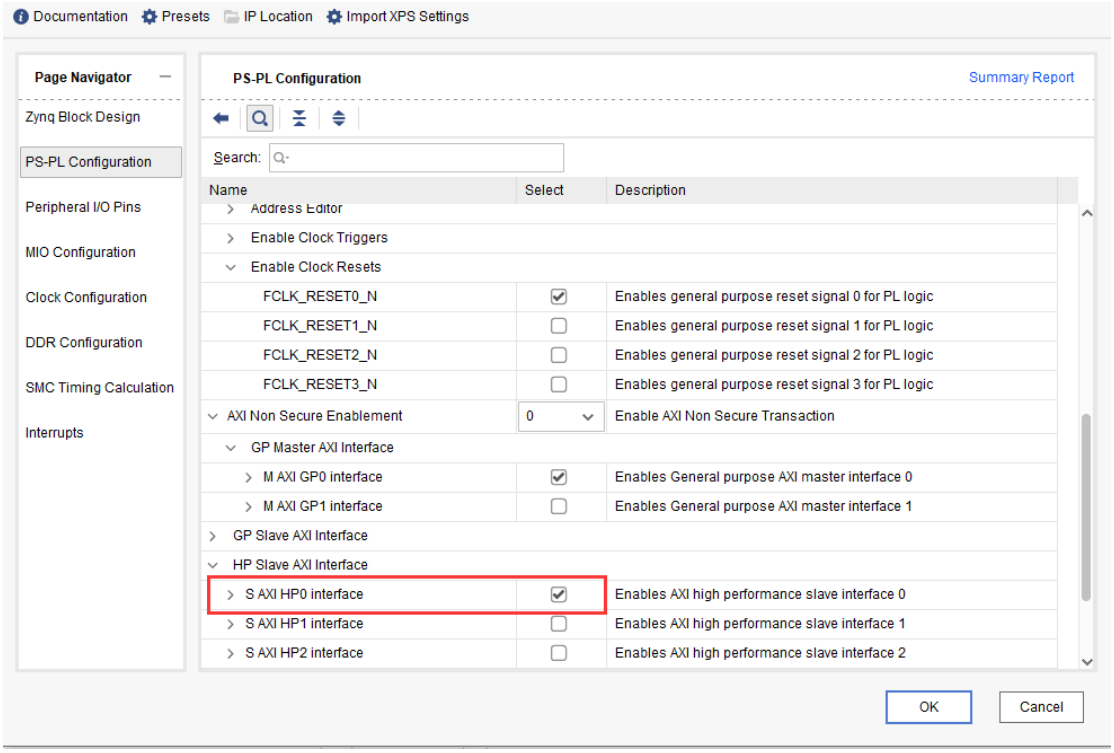
第5章 PetaLinux 定制 Linux(SD 卡启动)

本章主要讲解从 SD 卡启动 PetaLinux。主要实现的功能是： PS 端激活外设 SD，UART(用于打印 log)， PL 端扩展网络相关 IP，最终实现 PetaLinux 可以从 SD 卡启动，并且 Windows 主机可以通过 PL 端网口和开发板的 Linux 系统通信。对应的代码工程目录：1_petalinux_boot _from_sd。

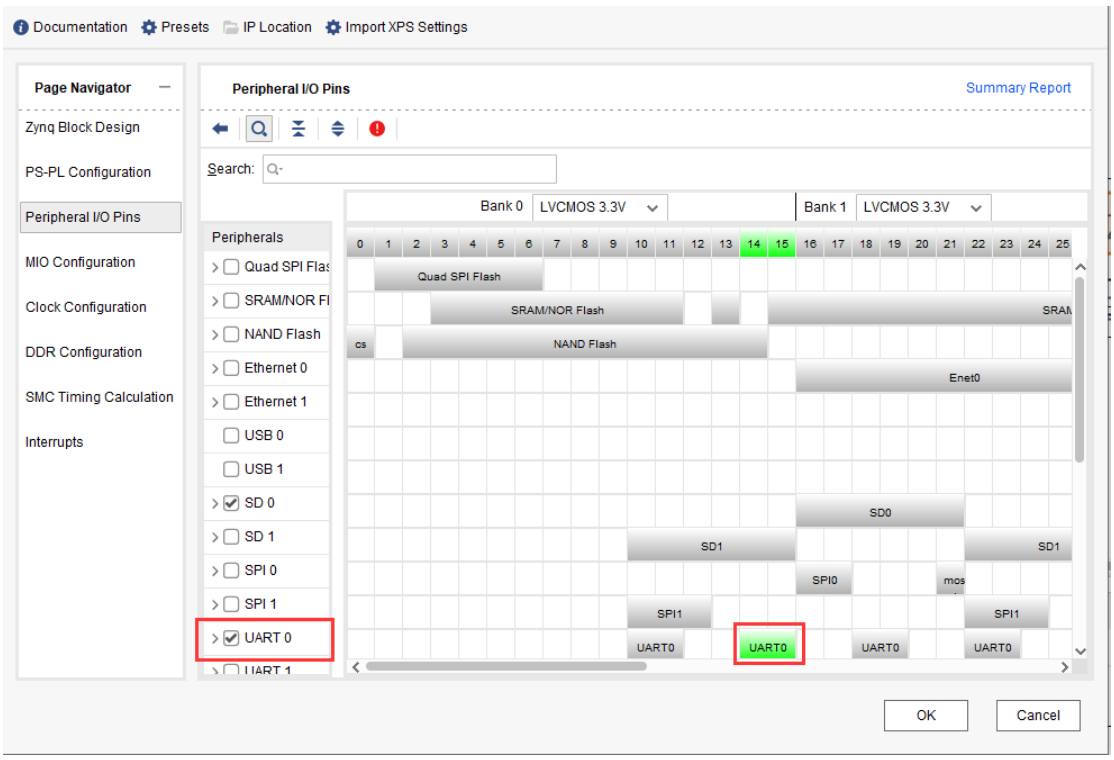
5.1 Vivado 创建工程

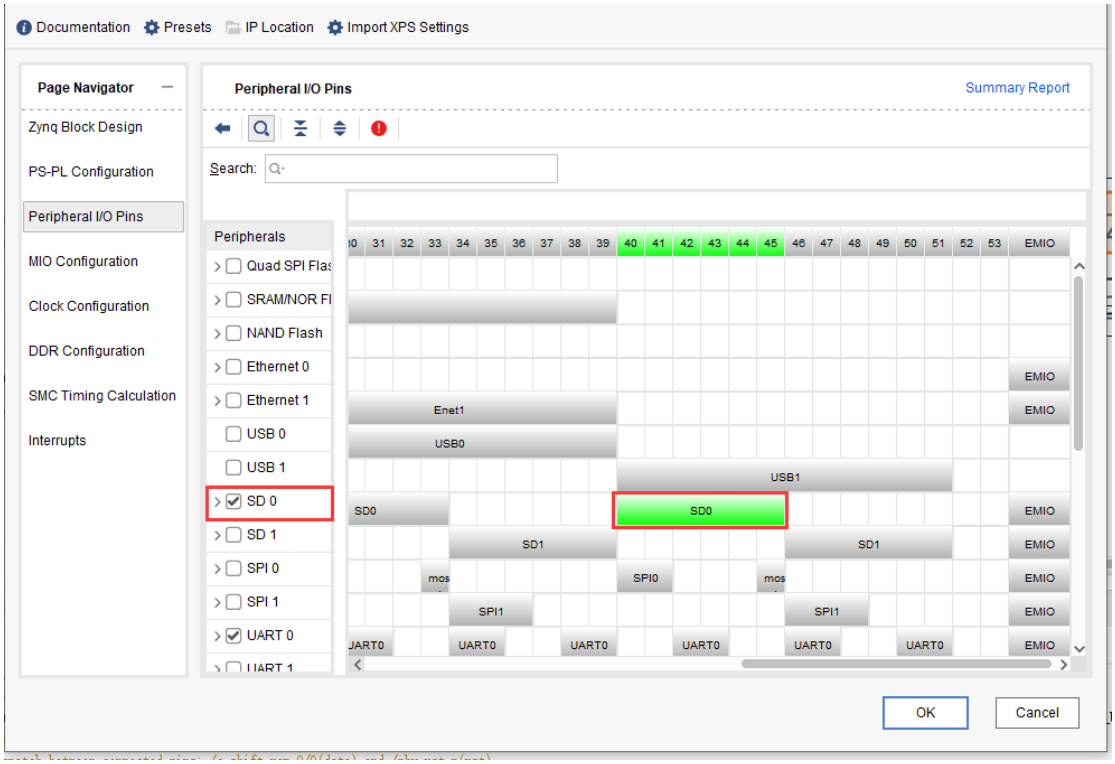
1) 创建一个 petalinx_boot_from_sd 的工程，创建好的工程如下。

2) 添加 ZYNQ 模块。 并双击 ZYNQ 模块，进行参数配置。点击左侧 PS-PL Configuration，勾选 S AXI HP0 interface

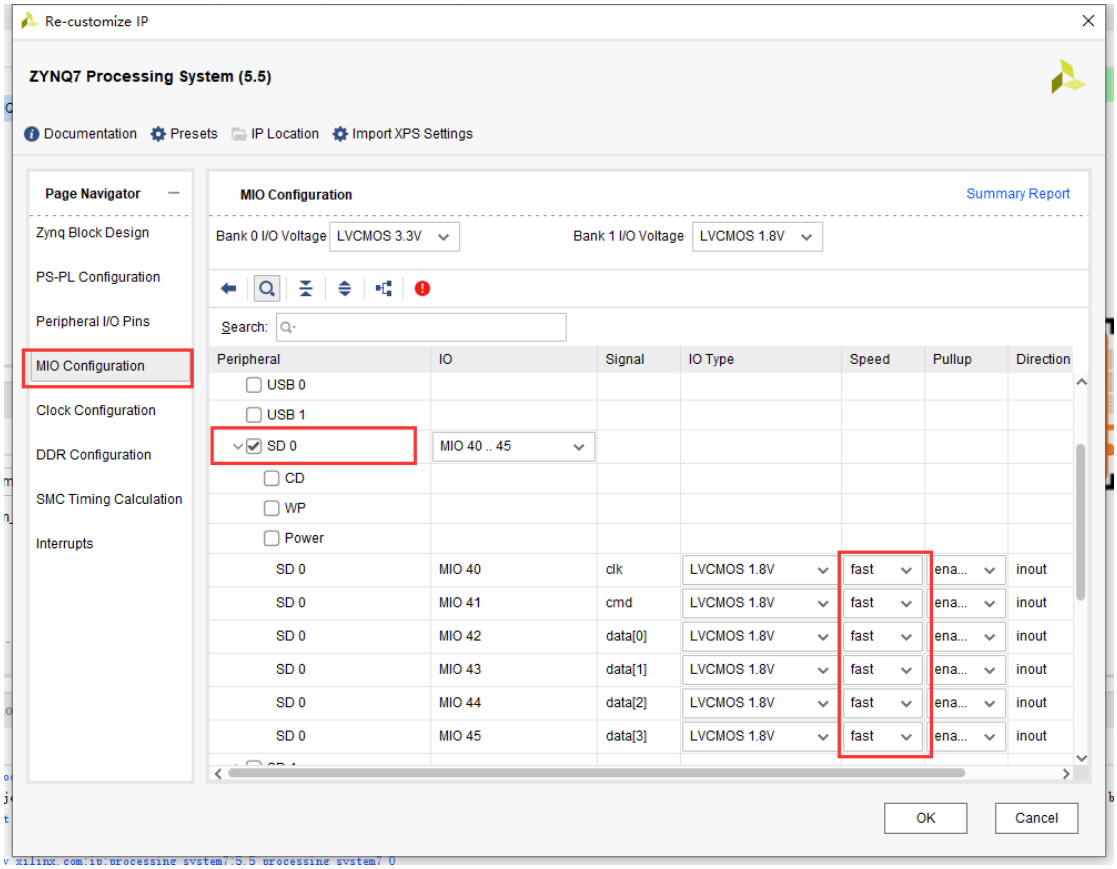


3) 点击左侧 Peripheral I/O Pins，使能位于 MIO14-MIO15 的 UART0，位于 MIO40-MIO45 的 SD0。



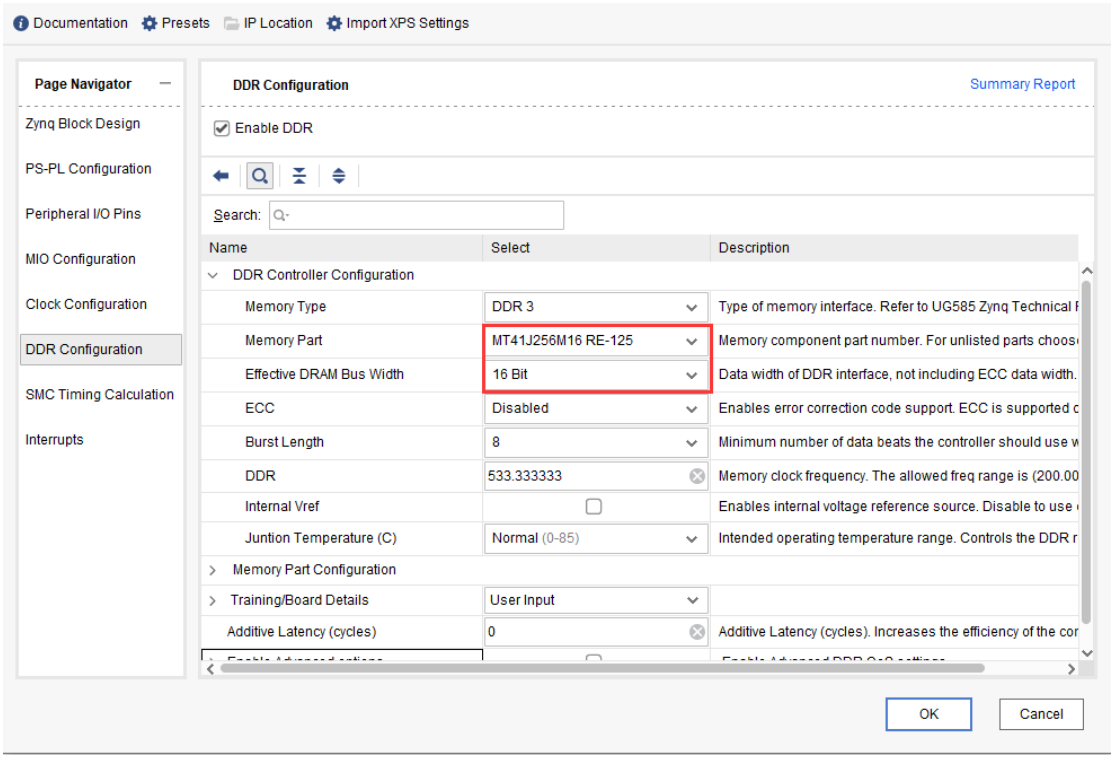


4）设置 SD 卡端口的 I/O 速度为 fast

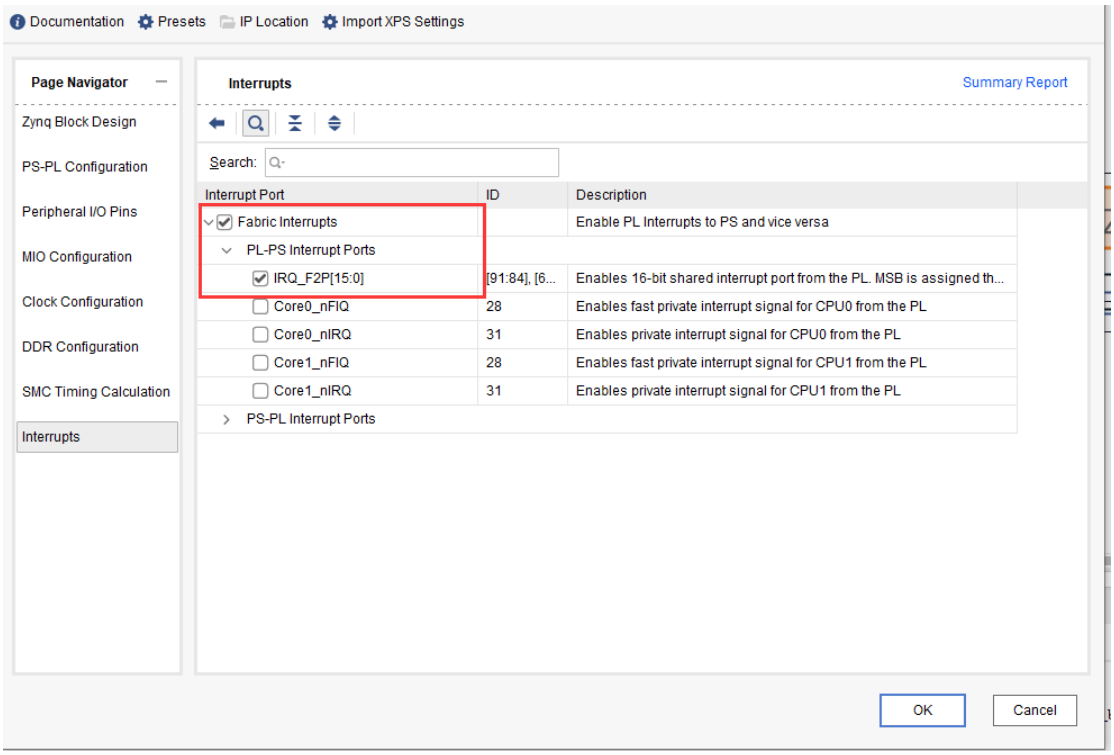


5) 点击左侧 DDR Configuration， Memory Part 选择 MT41J256M16 RE-125，

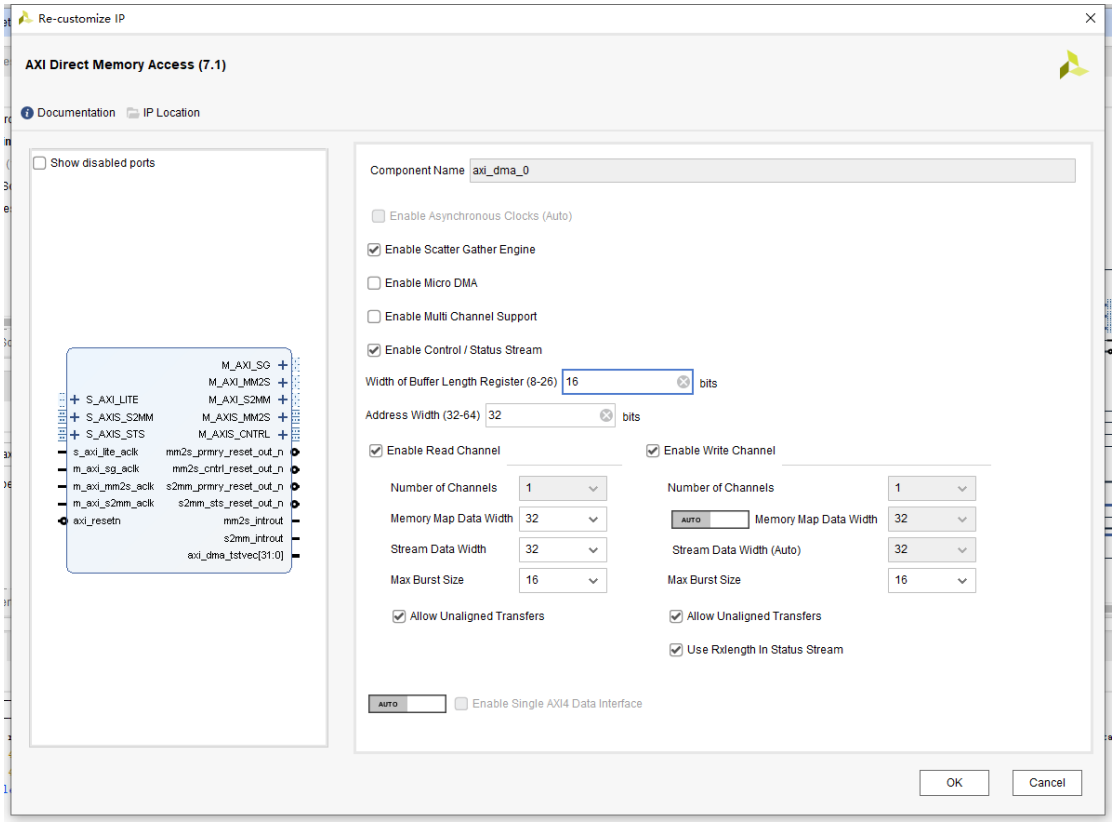
点击 OK。



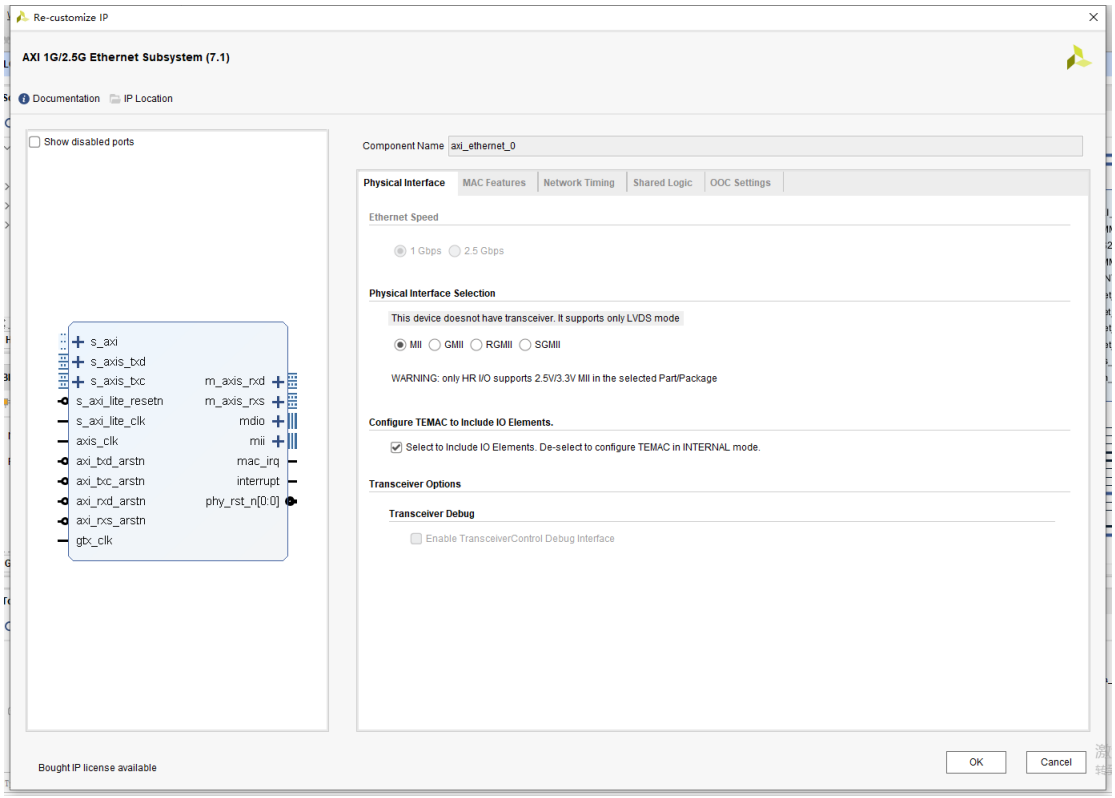
6)使能 PL 到 PS 的中断接口，接收来自 PL 端网络控制 IP 及 DMA 的中断信号。



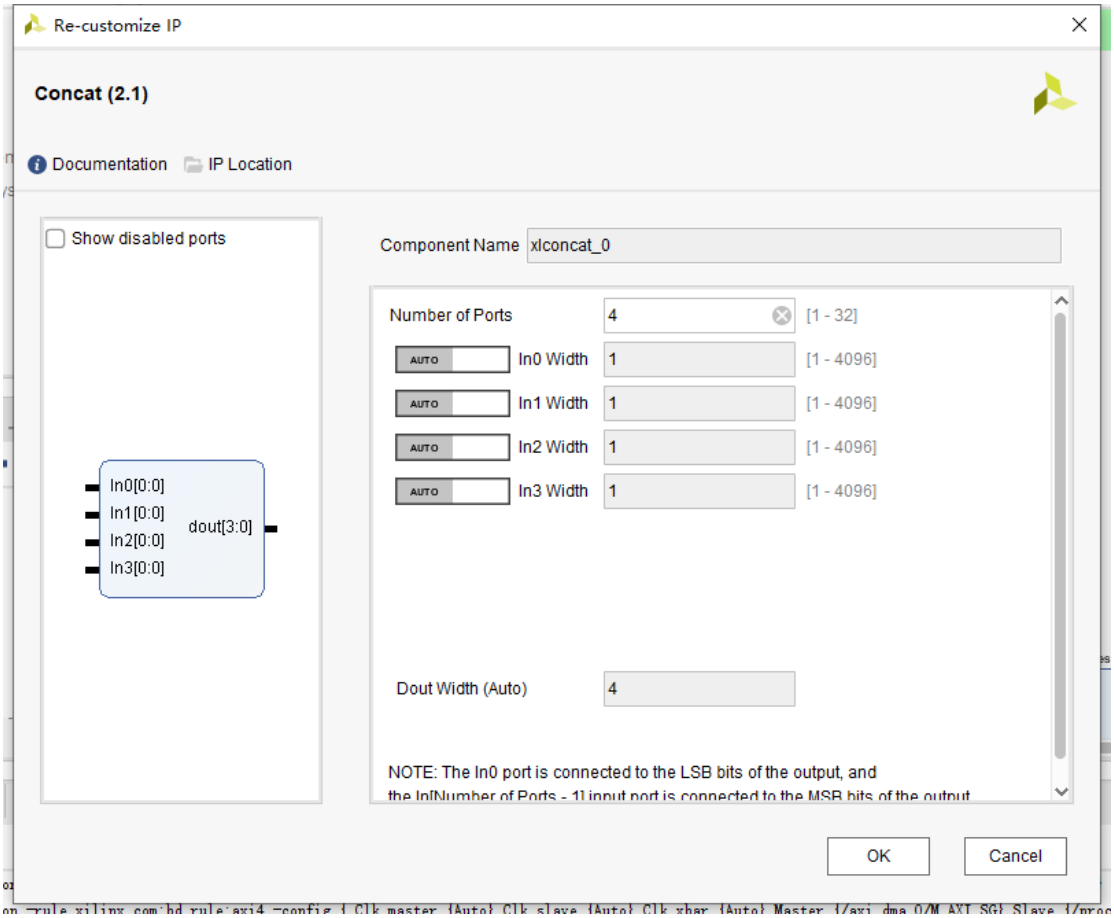
7)添加 DMA IP，并进行相应设置

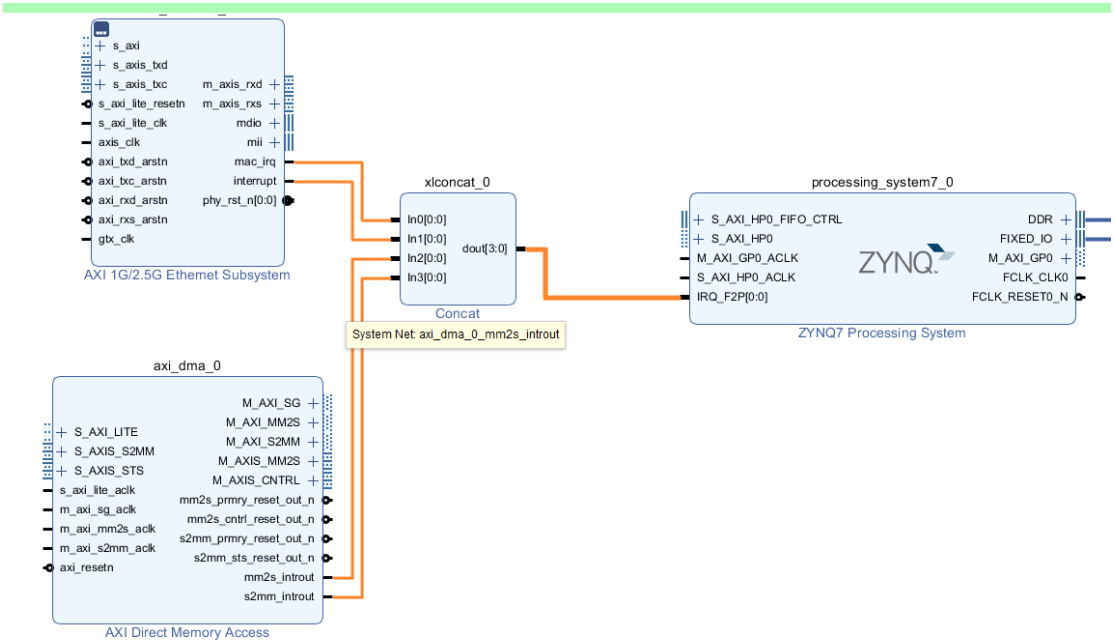


8)添加 AXI 1G/2.5G Ethernet Subsysem IP 核，并进行相应设置

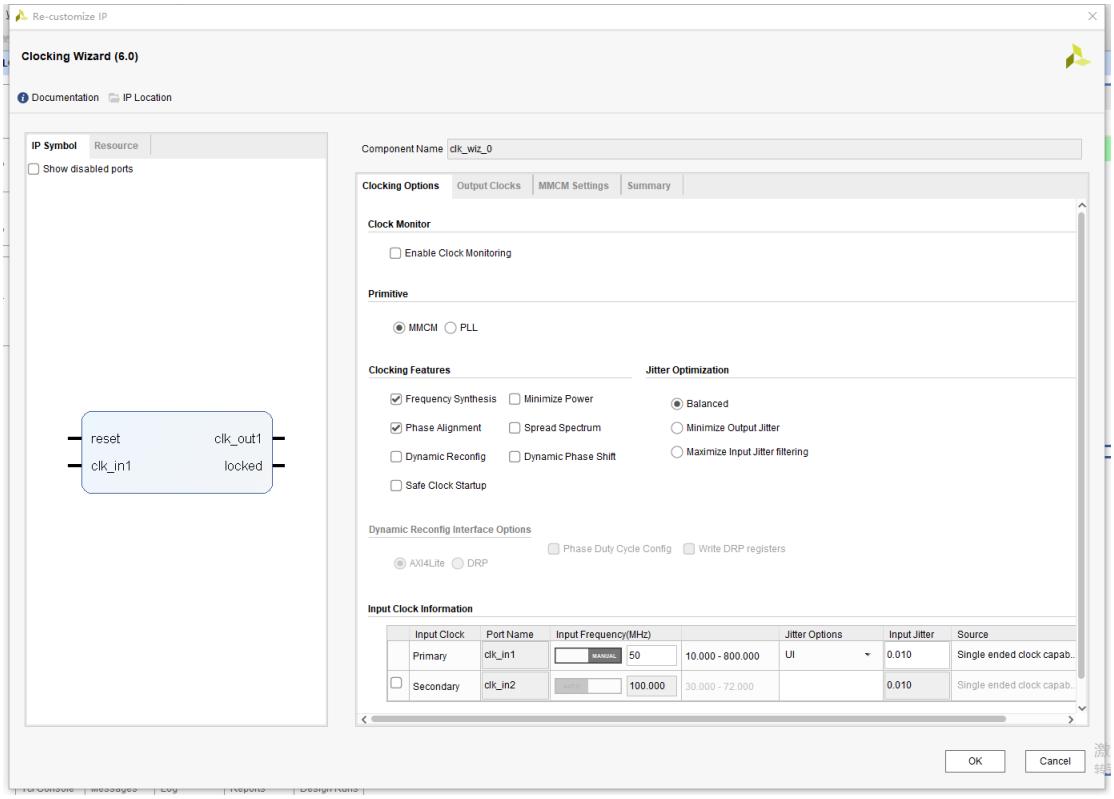


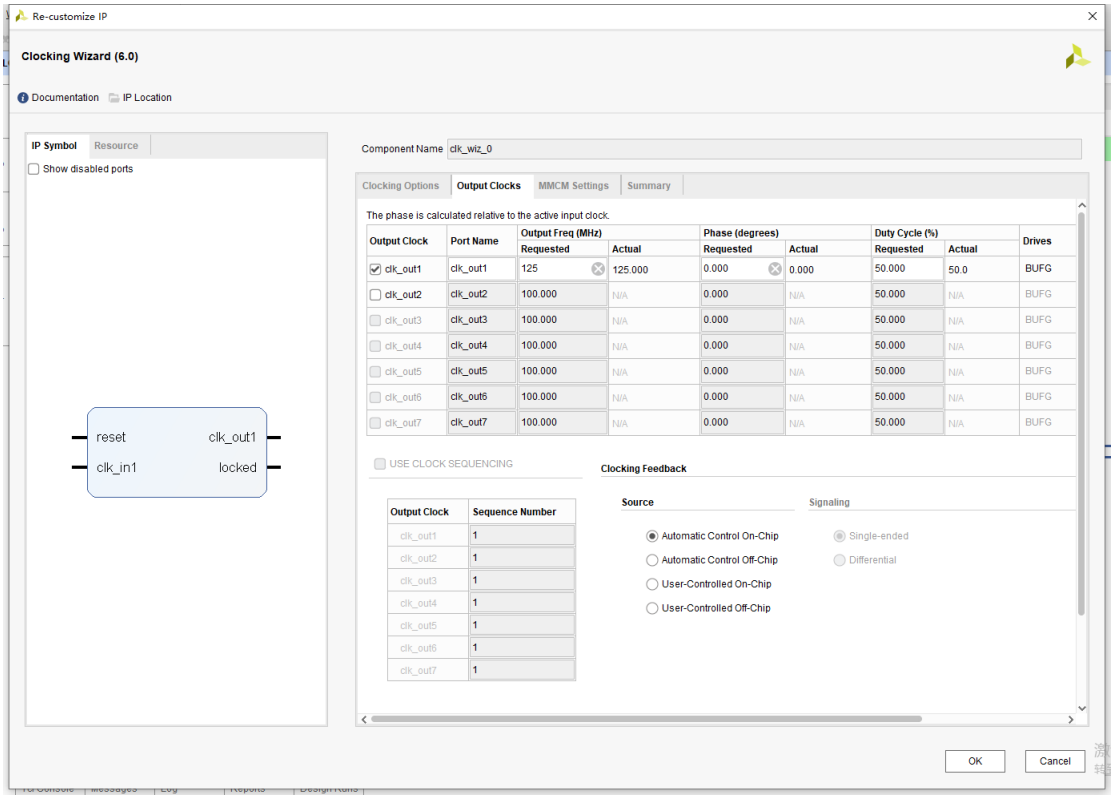
9)添加 concat IP 核，设置信号数为 4，并连接相应中断信号



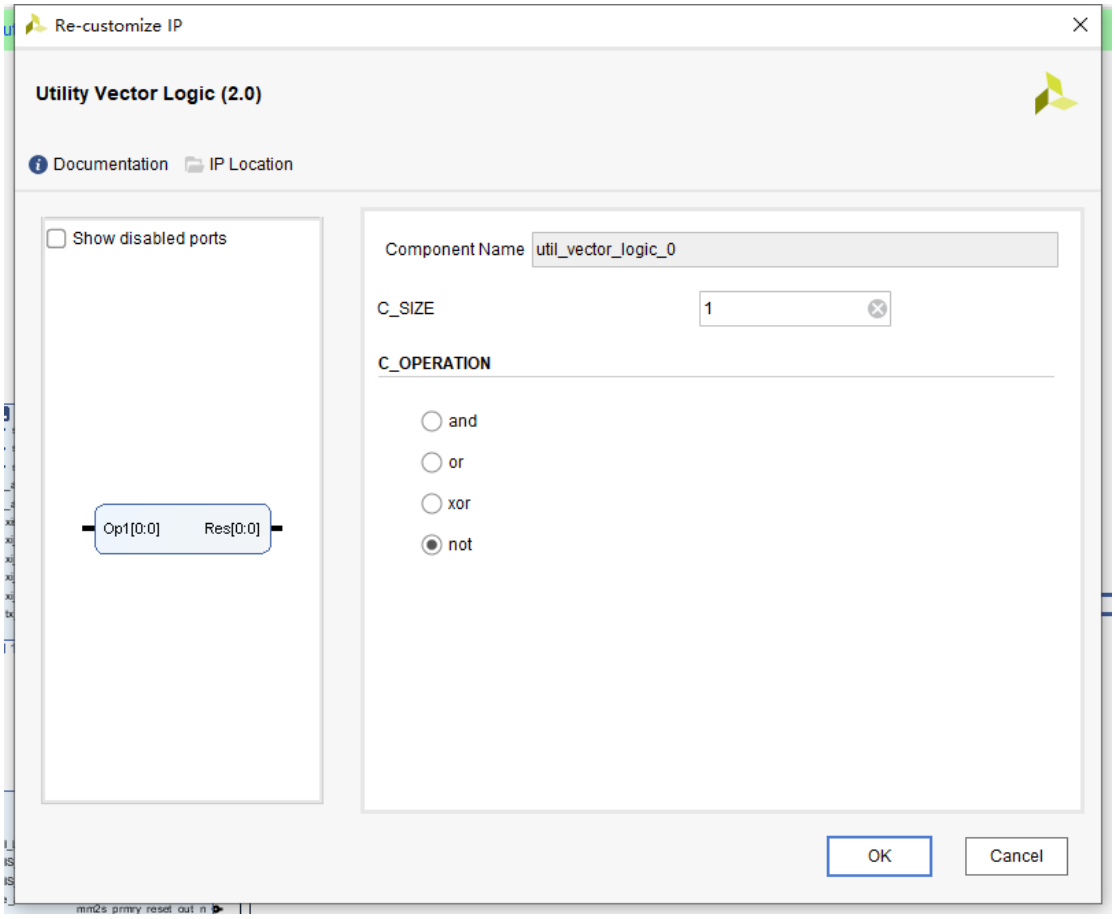


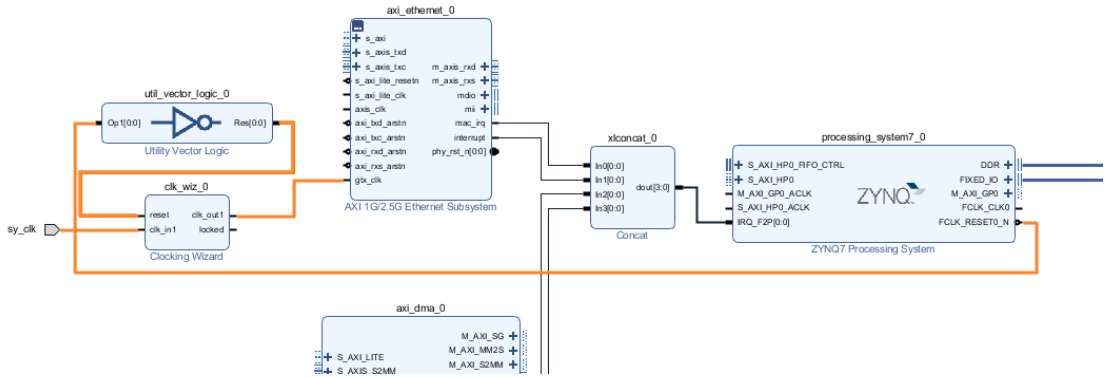
10)添加 clocking wizard 时钟 IP 并进行相应设置



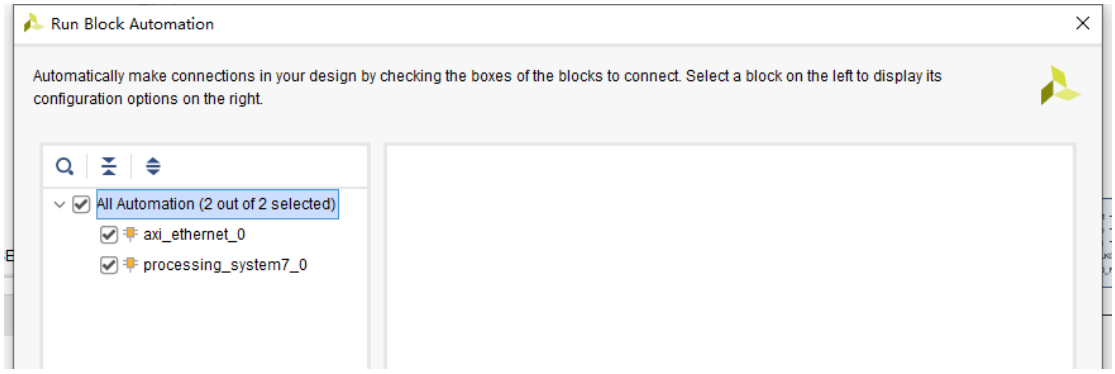


11)添加 Utility Vector Logic IP 并进行相应设置，连接相关信号

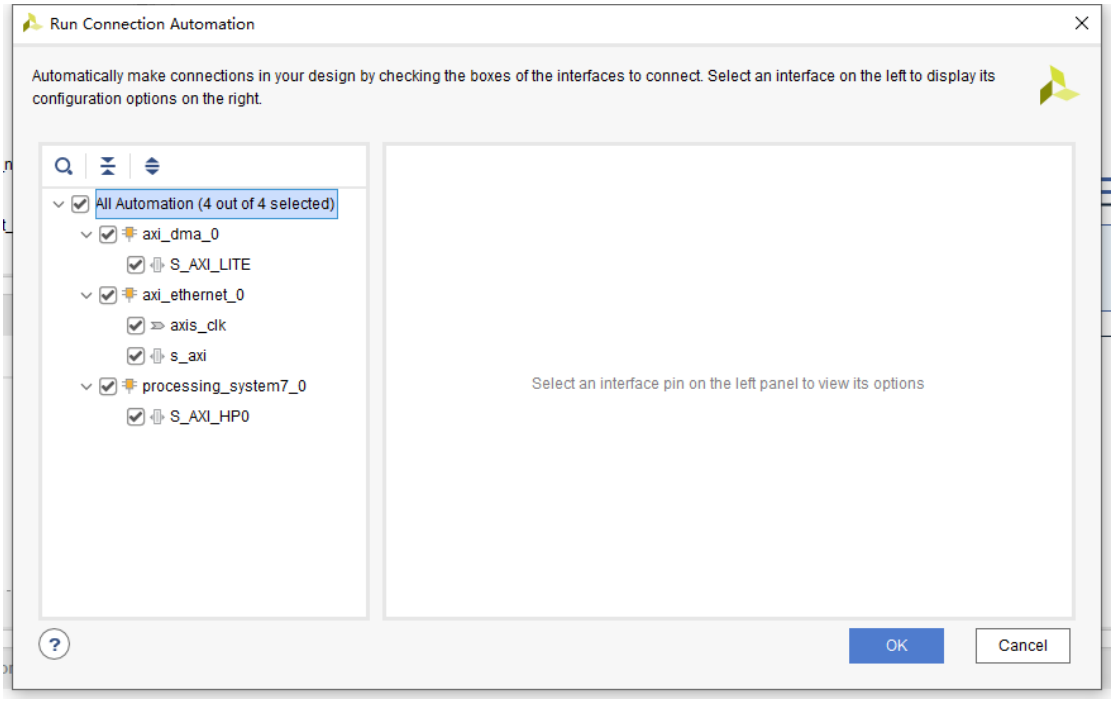




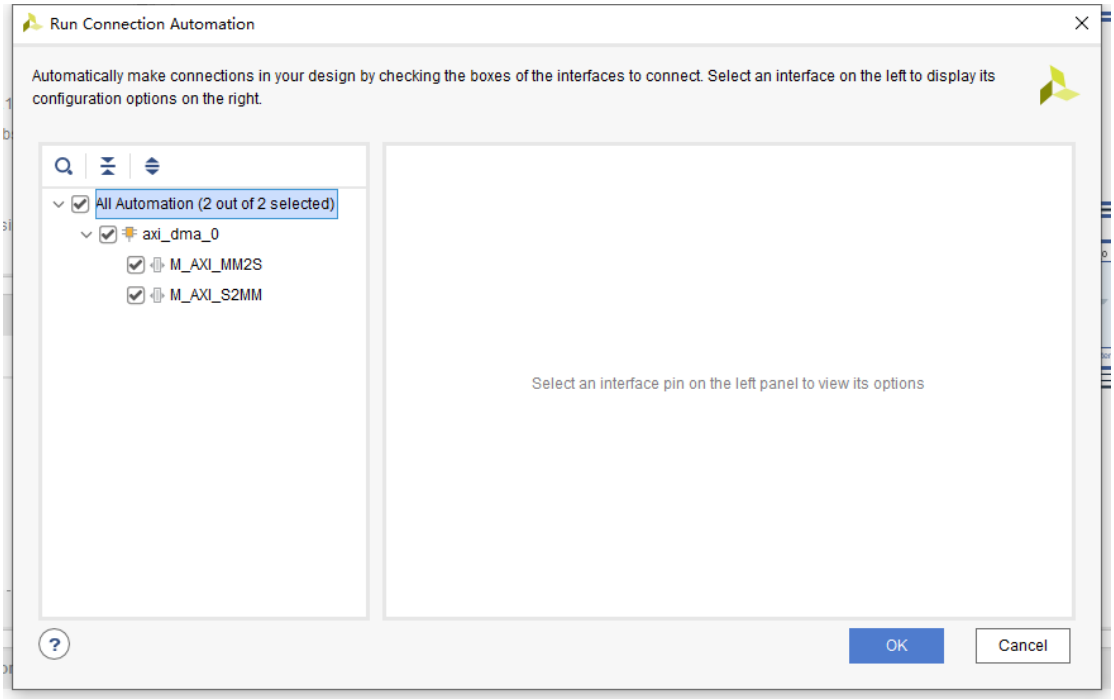
12)点击 Run Block Automation，全选所有 IP



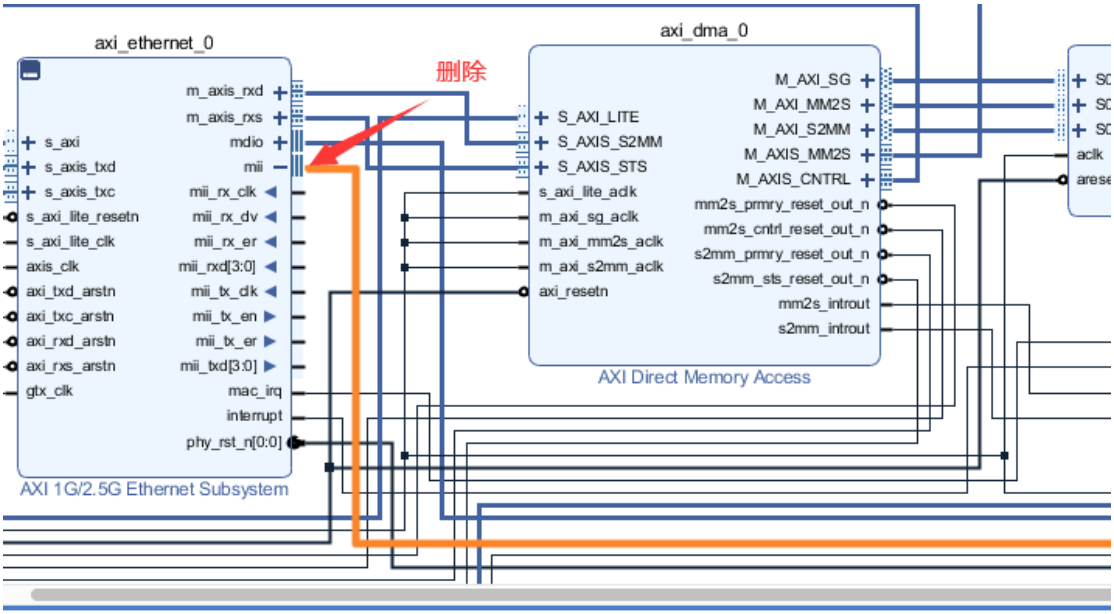
13)点击 Run Connection Automation，全选所有信号



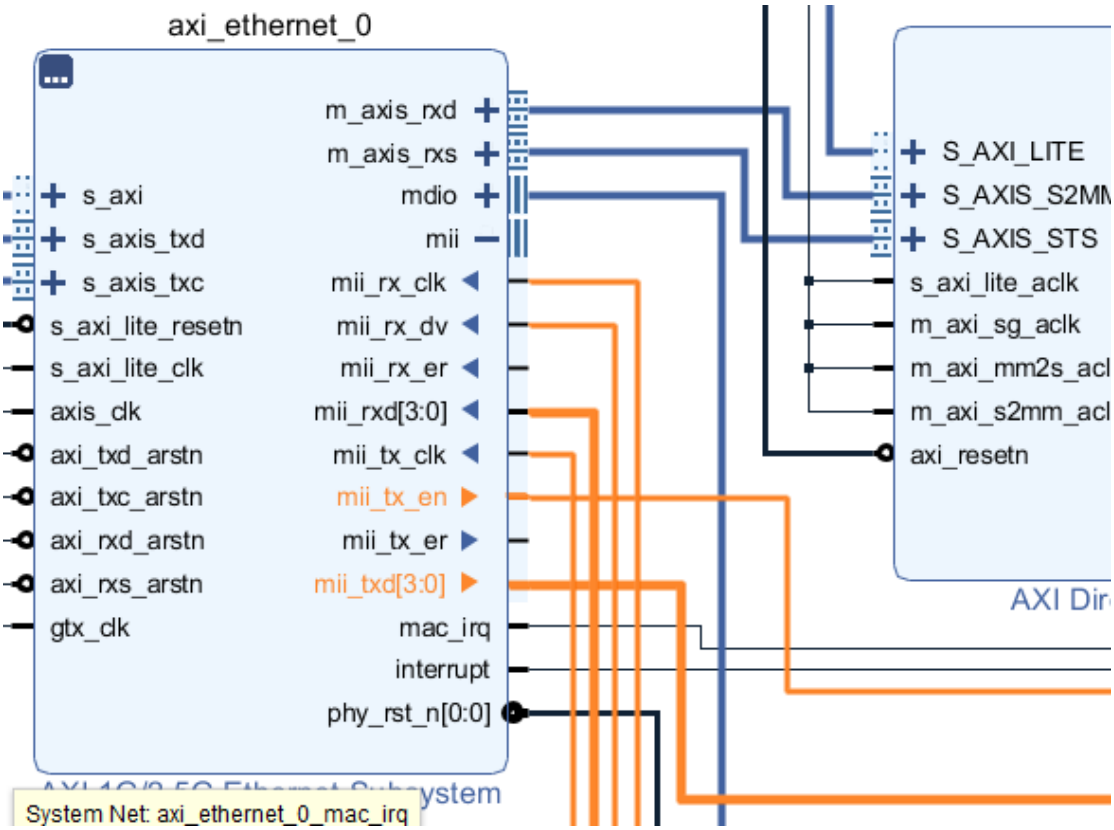
14)再次点击 Run Connection Automation，全选所有信号



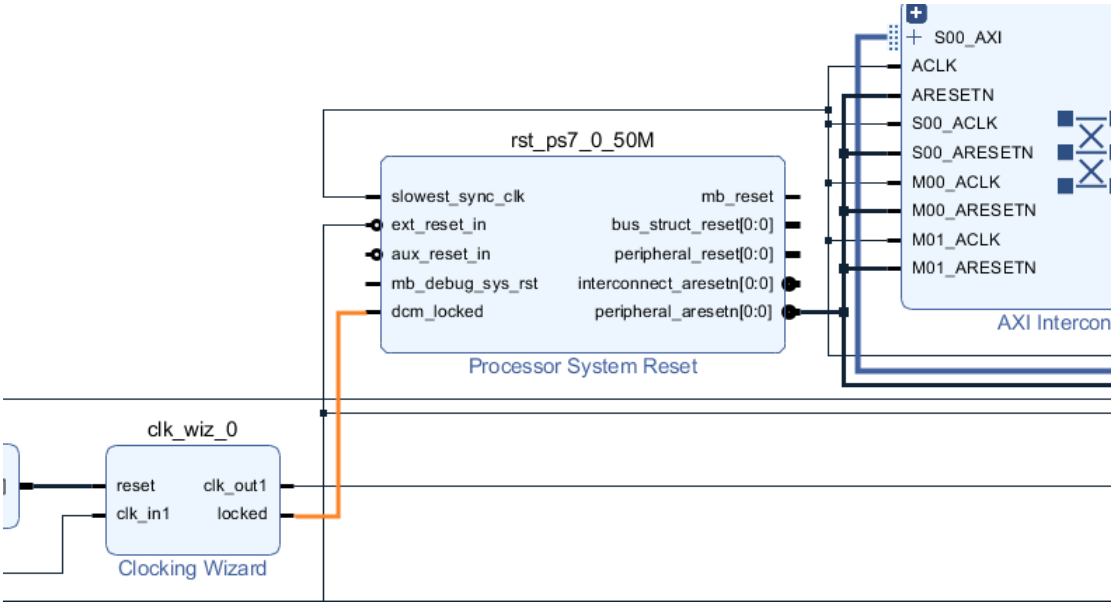
15)删除 MII 接口所有信号



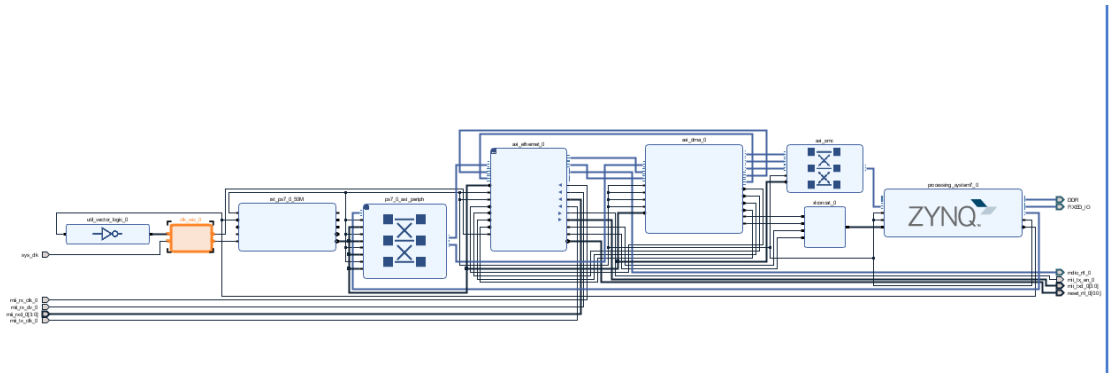
16) 重新将需要的信号引出



17)连接时钟锁定信号



18) 点击 Regenerate Layout，生成标准布局。



19) 点击工具栏 Validate Design 验证设计。

20) 保存文件，生成系统设计代码和顶层文件。

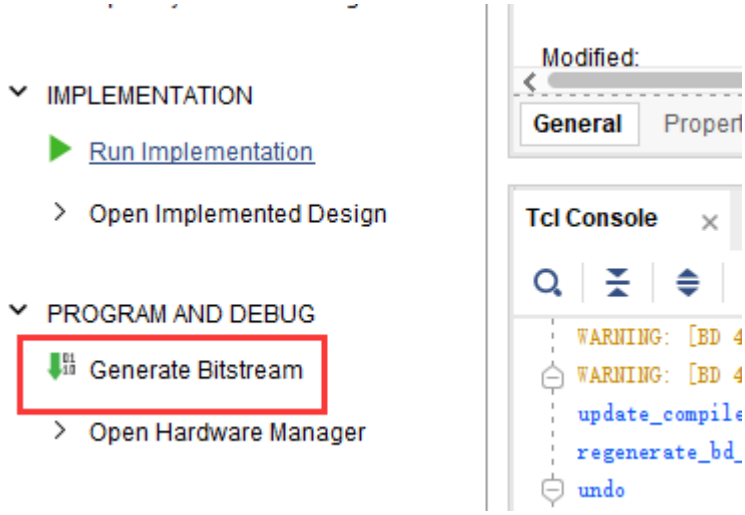
21)约束 phy 芯片引脚

```
set_property IOSTANDARD LVCMOS33 [get_ports {reset_rtl_0[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports sys_clk]
set_property IOSTANDARD LVCMOS33 [get_ports mdio_rtl_0_mdc]
set_property IOSTANDARD LVCMOS33 [get_ports mdio_rtl_0_mdio_io]

set_property IOSTANDARD LVCMOS33 [get_ports mii_tx_en_0]
set_property IOSTANDARD LVCMOS33 [get_ports {mii_txd_0[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {mii_txd_0[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {mii_txd_0[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {mii_txd_0[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {mii_rxd_0[3]}]
```

```
12 set_property IOSTANDARD LVCMOS33 [get_ports {mii_rxd_0[2]}]
13 set_property IOSTANDARD LVCMOS33 [get_ports {mii_rxd_0[1]}]
14 set_property IOSTANDARD LVCMOS33 [get_ports {mii_rxd_0[0]}]
15 set_property IOSTANDARD LVCMOS33 [get_ports mii_rx_dv_0]
16 set_property IOSTANDARD LVCMOS33 [get_ports mii_rx_clk_0]
17 set_property IOSTANDARD LVCMOS33 [get_ports mii_tx_clk_0]
18 set_property PACKAGE_PIN N16 [get_ports mii_tx_en_0]
19 set_property PACKAGE_PIN L14 [get_ports mii_tx_clk_0]
20 set_property PACKAGE_PIN K18 [get_ports mii_rx_dv_0]
21 set_property PACKAGE_PIN K17 [get_ports mii_rx_clk_0]
22 set_property PACKAGE_PIN J14 [get_ports {mii_rxd_0[0]}]
23 set_property PACKAGE_PIN K14 [get_ports {mii_rxd_0[1]}]
24 set_property PACKAGE_PIN M18 [get_ports {mii_rxd_0[2]}]
25 set_property PACKAGE_PIN M17 [get_ports {mii_rxd_0[3]}]
26 set_property PACKAGE_PIN M14 [get_ports {mii_txd_0[0]}]
27 set_property PACKAGE_PIN L15 [get_ports {mii_txd_0[1]}]
28 set_property PACKAGE_PIN M15 [get_ports {mii_txd_0[2]}]
29 set_property PACKAGE_PIN N15 [get_ports {mii_txd_0[3]}]
30 set_property PACKAGE_PIN G14 [get_ports mdio_rtl_0_mdc]
31 set_property PACKAGE_PIN J15 [get_ports mdio_rtl_0_mdio_io]
32 set_property PACKAGE_PIN H20 [get_ports {reset_rtl_0[0]}]
33 set_property PACKAGE_PIN N18 [get_ports sys_clk]
```

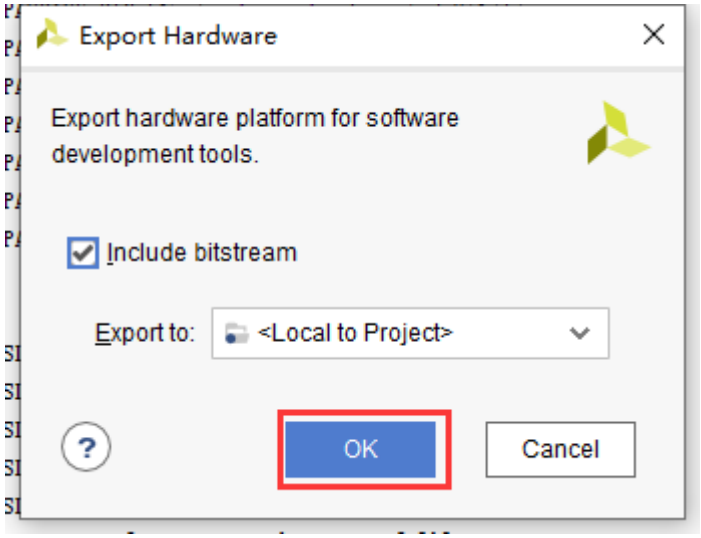
```
36 set_property SLEW FAST [get_ports mdio_rtl_0_mdc]
37 set_property SLEW FAST [get_ports mdio_rtl_0_mdio_io]
38 set_property SLEW SLOW [get_ports {reset_rtl_0[0]}]
39 set_property SLEW FAST [get_ports mii_tx_en_0]
40 set_property SLEW FAST [get_ports {mii_txd_0[3]}]
41 set_property SLEW FAST [get_ports {mii_txd_0[2]}]
42 set_property SLEW FAST [get_ports {mii_txd_0[1]}]
43 set_property SLEW FAST [get_ports {mii_txd_0[0]}]
```

22)综合编译生成 Bitstream 文件

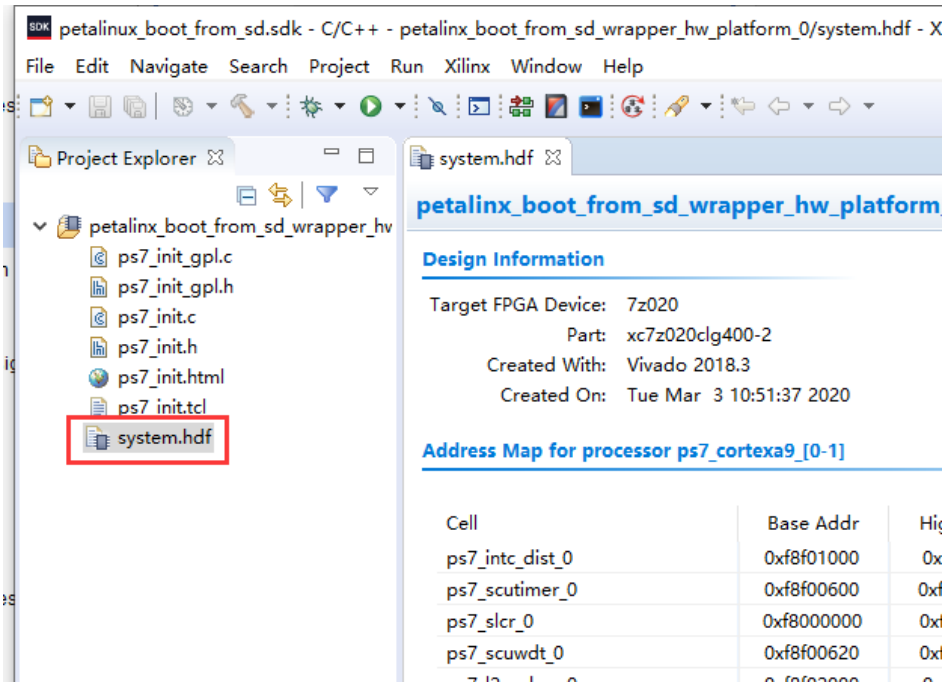


5.2 设计导出至 SDK

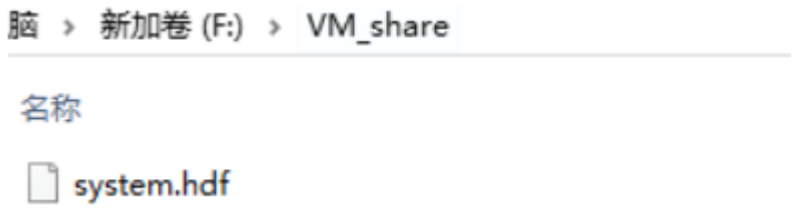
1)编译完成后导出硬件描述文件。点击 File-->Export-->Export Hardware，点击 OK。

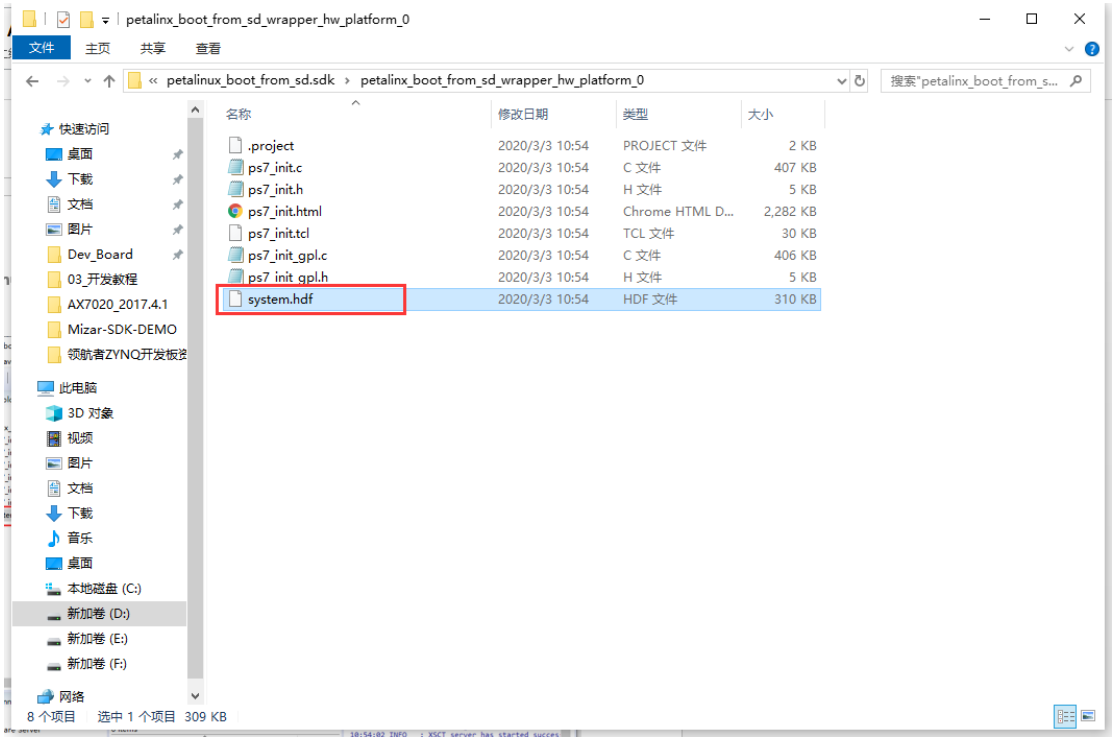


2) 打开 SDK 工程如下，包含系统硬件描述文件 system.hdf。



3) 将目录 D:\Z7-Lite-7020\06_Linux_Demo\1_petalinux_boot_from_sd\petalinux_boot_from_sd\petalinux_boot_from_sd.sdk\petalinx_boot_from_sd_wrapper_hw_platform_0 下的 system.hdf 拷贝到共享文件夹 F:\VM_share 下，PetaLinux 需要用到这个硬件描述语言





5.3 PetaLinux 工程建立

1) 打开 VitrualBox 的 Ubuntu Linux 系统，执行下列命令，挂载 VM_share 到目录/mnt。

```
sudo mount –t vboxsf VM_share /mnt
```

```
micriophase@micriophase-VirtualBox:~$ sudo mount -t vboxsf VM_share /mnt
[sudo] password for micriophase:
```

2) 执行下列命令，使 PetaLinux 工具可以使用。

```
source /opt/pkg/petalinux/2018.3/settings.sh
```

```
micriophase@micriophase-VirtualBox:~$ source /opt/pkg/petalinux/2018.3/settings.sh
PetaLinux environment set to '/opt/pkg/petalinux/2018.3'
INFO: Checking free disk space
INFO: Checking installed tools
INFO: Checking installed development libraries
INFO: Checking network and other services
```

3) 为了统一管理，执行下列命令在当前目录下建立文件夹 PetaLinux_proj，进入该文件夹。

```
mkdir PetaLinux_proj
cd PetaLinux_proj
```

```
micriophase@micriophase-VirtualBox:~$ mkdir PetaLinux_proj
micriophase@micriophase-VirtualBox:~$ cd PetaLinux_proj
```

4) 执行下列命令创建 PetaLinux 工程。

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

```
micriophase@micriophase-VirtualBox:~/PetaLinux_proj$ petalinux-create --type project
 --template zynq --name petalinux_boot_from_sd
INFO: Create project: petalinux_boot_from_sd
INFO: New project successfully created in /home/micriophase/PetaLinux_proj/petalinux
_boot_from_sd
```

5) 进入该目录，执行下列命令将/mnt 下的 system.hdf 文件拷贝至当前目录。

```
cd petalinux_boot_from_sd
cp /mnt/system.hdf .
```

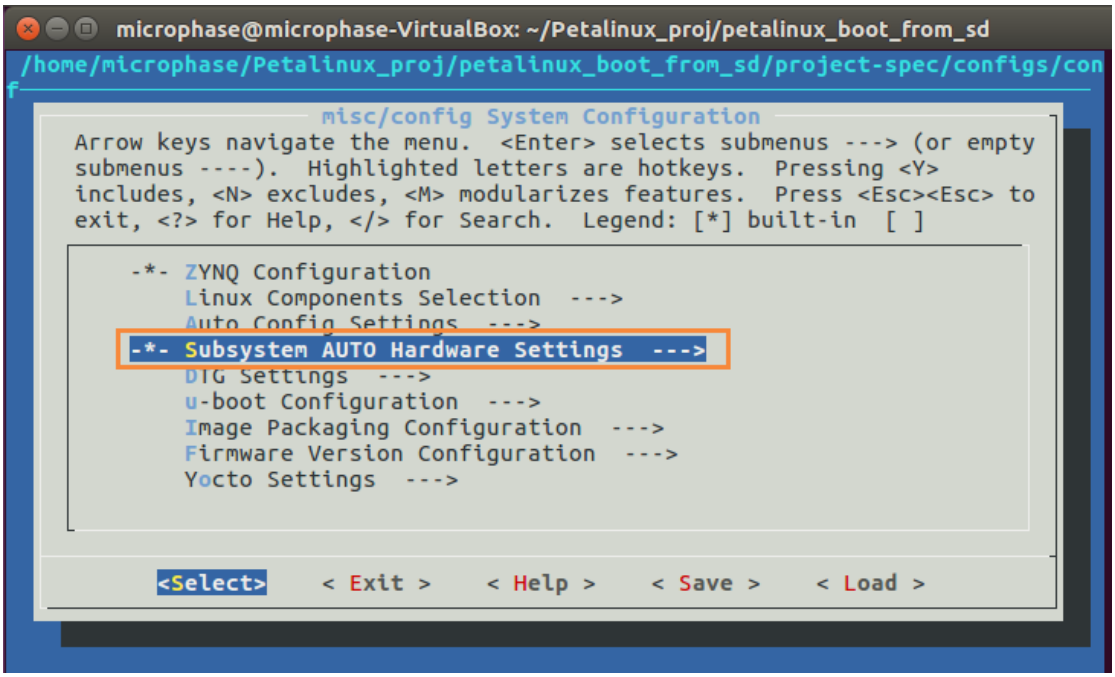
```
micriophase@micriophase-VirtualBox:~/PetaLinux_proj$ cd petalinux_boot_from_sd
micriophase@micriophase-VirtualBox:~/PetaLinux_proj/petalinux_boot_from_sd$ cp /mnt/system.hdf .
```

6) 执行下列命令，指定导入硬件描述文件 system.hdf 的路径，本例就是当前路径。

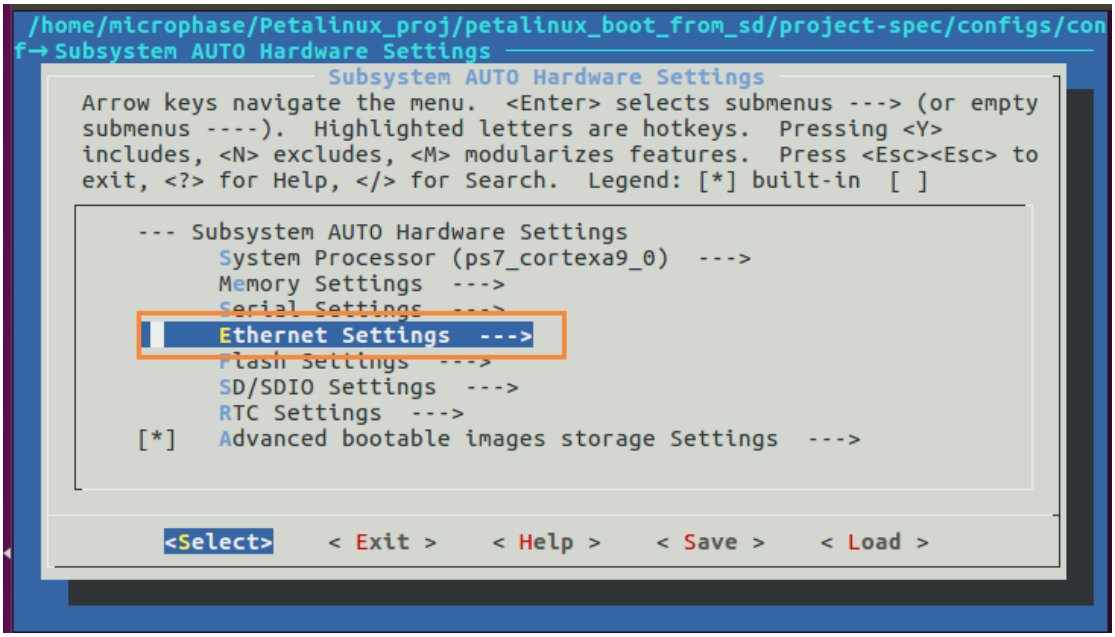
```
petalinux-config --get-hw-description=.
```

```
micriophase@micriophase-VirtualBox:~/PetaLinux_proj/petalinux_boot_from_sd$ petalinux-config
 --get-hw-description=.
```

7) 执行命令打开配置界面。可用方向键移动光标，按 Enter 进入某一条目查看详情，按两次 Esc 退回至上一层。进入 Subsystem AUTO Hardware Settings



进入 Ethernet Settings，



设置 Primary Ethernet 为 axi_ethernet_0

```
microphase@microphase-VirtualBox: ~/Petalinux_proj/petalinux_boot_from_sd
/home/microphase/Petalinux_proj/petalinux_boot_from_sd/project-spec/configs/con
f→ Subsystem AUTO Hardware Settings →Ethernet Settings
                          Ethernet Settings
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]

      Primary Ethernet (axi_ethernet_0)  --->
      [ ] Randomise MAC address
      (00:0a:35:00:22:01) Ethernet MAC address
      [*] Obtain IP address automatically

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

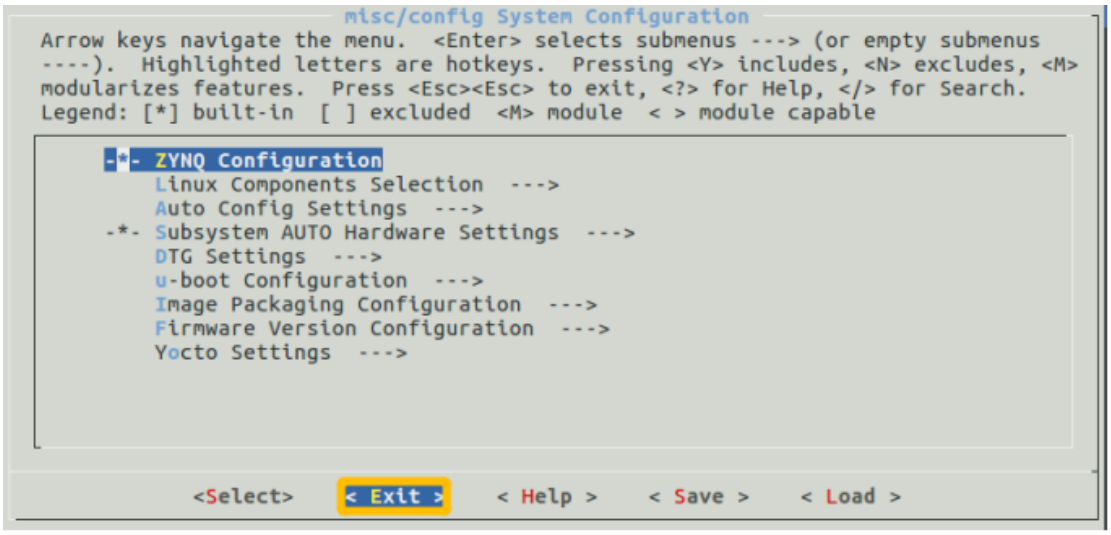
之后返回到主界面，选择 save，回车

```
                          misc/config System Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

      -*- ZYNQ Configuration
          Linux Components Selection  --->
          Auto Config Settings  --->
      -*- Subsystem AUTO Hardware Settings  --->
          DTG Settings  --->
          u-boot Configuration  --->
          Image Packaging Configuration  --->
          Firmware Version Configuration  --->
          Yocto Settings  --->

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

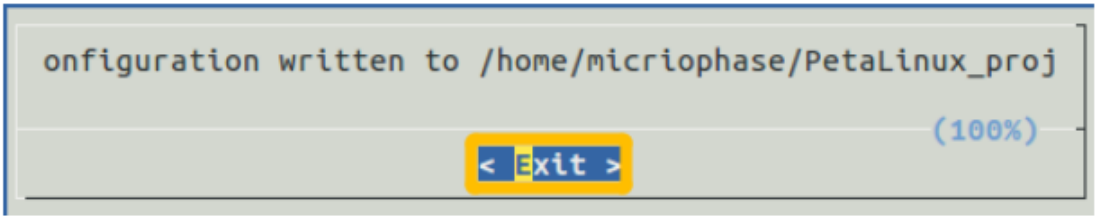
选择 ok，回车，

```
  Enter a filename to which this configuration
  should be saved as an alternate.  Leave blank to
  abort.

  talinux_boot_from_sd/project-spec/configs/config

        <  Ok  >        < Help >
```

选择 Exit，



选择 Exit。



8) 打开以下路径中的设备树文件：/home/microphase/Petalinux_proj/petalinux_boot_from_sd/project-spec/meta-user/recipes-bsp/device-tree/files/system-user.dtsi，向其中添加如下内容：

```
/include/ "system-conf.dtsi"
/ {
};

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

        mdio{
                #address-cells = <1>;
                #size-cells = <0>;
                phy1:phy@1{
                        compatible = "realtek,rtl8201","ethernet-phy-id001c.c816";
                        device_type = "ethernet-phy";
                        reg = <0>;

                };
        };
};
```

9)添加完成后保存，之后执行下列命令，编译工程。

petalinux-build

```
micriophase@micriophase-VirtualBox:~/PetaLinux_proj/petalinux_boot_from_sd$ petalinux-build
[INFO] building project
[INFO] sourcing bitbake
INFO: bitbake petalinux-user-image
Parsing recipes: 100% |#################################################################| Ti
me: 0:01:57
Parsing of 2569 .bb files complete (0 cached, 2569 parsed). 3445 targets, 149 skipped, 0 mas
ked, 0 errors.
NOTE: Resolving any missing task queue dependencies
Initialising tasks: 100% |##############################################################| Ti
me: 0:00:14
Checking sstate mirror object availability: 100% |######################################| Ti
me: 0:00:19
NOTE: Executing SetScene Tasks
NOTE: Executing RunQueue Tasks
WARNING: petalinux-user-image-1.0-r0 do_rootfs: [log_check] petalinux-user-image: found 1 wa
rning message in the logfile:
[log_check] warning: %post(sysvinit-inittab-2.88dsf-r10.plnx_zynq7) scriptlet failed, exit s
tatus 1

NOTE: Tasks Summary: Attempted 3065 tasks of which 2246 didn't need to be rerun and all succ
eeded.

Summary: There was 1 WARNING message shown.
INFO: Copying Images from deploy to images
INFO: Creating images/linux directory
NOTE: Failed to copy built images to tftp dir:  /tftpboot
[INFO] successfully built project
```

9) 编译完成后，在当前目录下生成 images 文件夹。

```
micriophase@micriophase-VirtualBox:~/PetaLinux_proj/petalinux_boot_from_sd$ ll
total 352
drwxr-xr-x 8 micriophase micriophase   4096 9月  18 11:45 ./
drwxrwxr-x 3 micriophase micriophase   4096 9月  18 11:13 ../
drwxrwxr-x 8 micriophase micriophase   4096 9月  18 11:45 build/
drwxrwxr-x 3 micriophase micriophase   4096 9月  18 11:17 components/
-rw-r--r-- 1 micriophase micriophase    248 9月  18 11:13 config.project
-rw-rw-r-- 1 micriophase micriophase    186 9月  18 11:13 .gitignore
drwxrwxr-x 3 micriophase micriophase   4096 9月  18 11:45 images/
drwxrwxr-x 2 micriophase micriophase   4096 9月  18 11:45 .petalinux/
drwxr-xr-x 6 micriophase micriophase   4096 9月  18 11:18 project-spec/
-rwxrwxr-x 1 micriophase micriophase 317372 9月  18 11:14 system.hdf*
drwxrwxr-x 4 micriophase micriophase   4096 9月  18 11:28 .Xil/
```

10) 执行下列命令，将其拷贝到/mnt 下。

cp -r images/ /mnt/

```
micriophase@micriophase-VirtualBox:~/PetaLinux_proj/petalinux_boot_from_sd$ cp -r images/ /mnt/
```

11）在 Windows 共享文件夹 VM_share 下可以看到 images 文件夹了。

12）images 文件夹中，主要用到的文件有三个。

○ zynq_fsbl.elf：当嵌入式系统启动时， ARM 将执行 PS 域上 ROM 中的程序，将非易失性存储如 Flash， SD 卡上的 first stage boot loader 文件加载到 DDR 并执行，first stage boot loader 主要完成以下功能；

① 使用 FPGA 比特流文件配置 FPGA（如果存在比特流）；

② 配置 PS 域的 MIO 端口；

③ 初始化 DDR 控制器；

④ 初始化系统时钟， PLL；

⑤ 将 u-boot 从非易失存储如 Flash， SD 卡等 load 到 DDR 中。

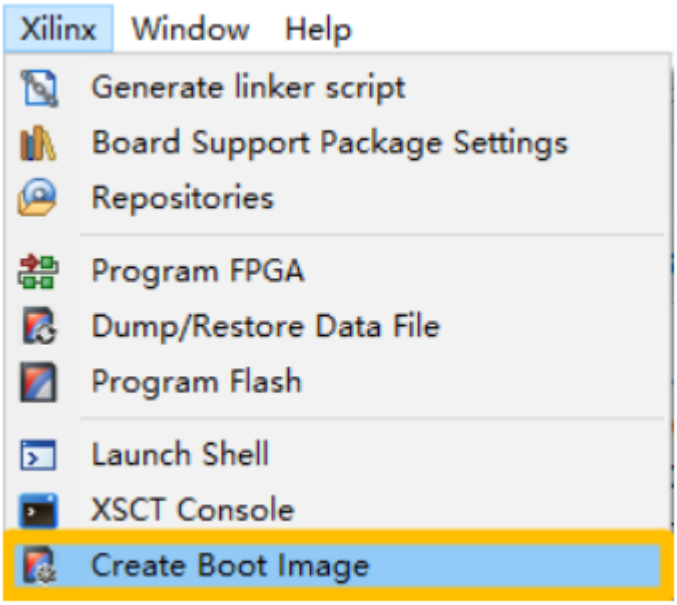
○ u-boot.elf：嵌入式系统通用 Boot 文件，负责将 image.ub 从非易失存储如 Flash，SD 卡等 load 到 DDR 中；

○ image.ub： PetaLinux，包括内核镜像、 ramdisk、设备树 dtb。

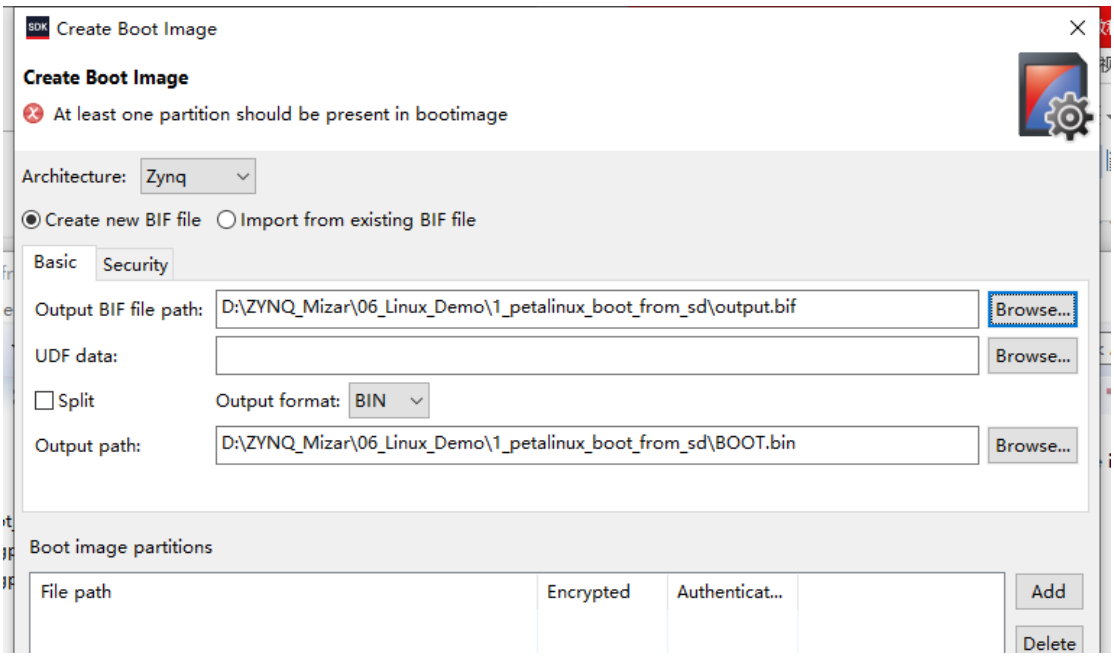
13）将以上几个文件拷贝到当前工程目录下。

5.4 BOOT.bin 文件打包

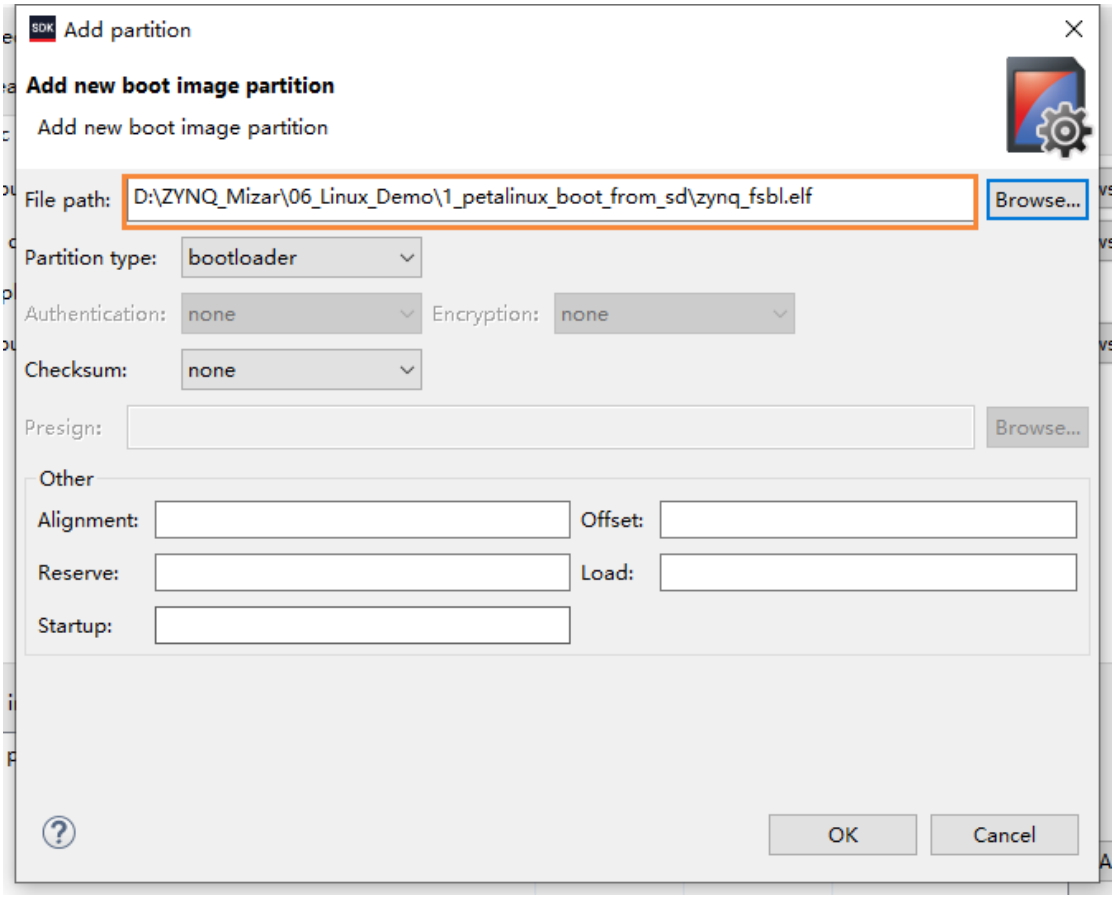
1) 回到 Xilinx SDK 工程，点击菜单栏 Xilinx-->Create Boot Image。



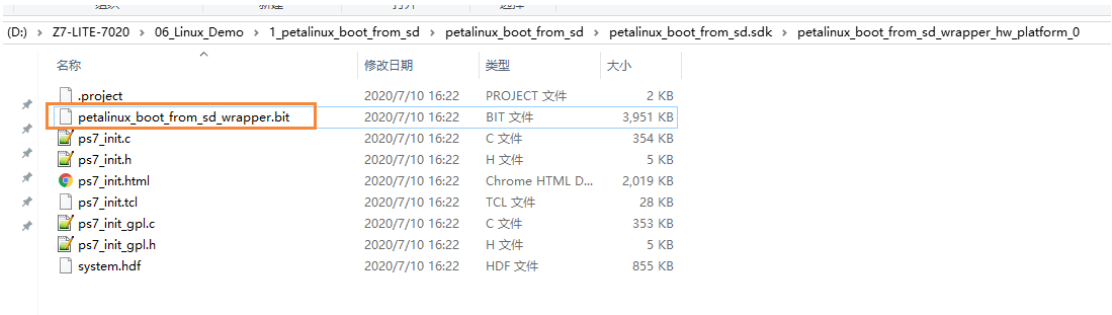
2) 指定系统输出路径为当前路径，点击 Add。

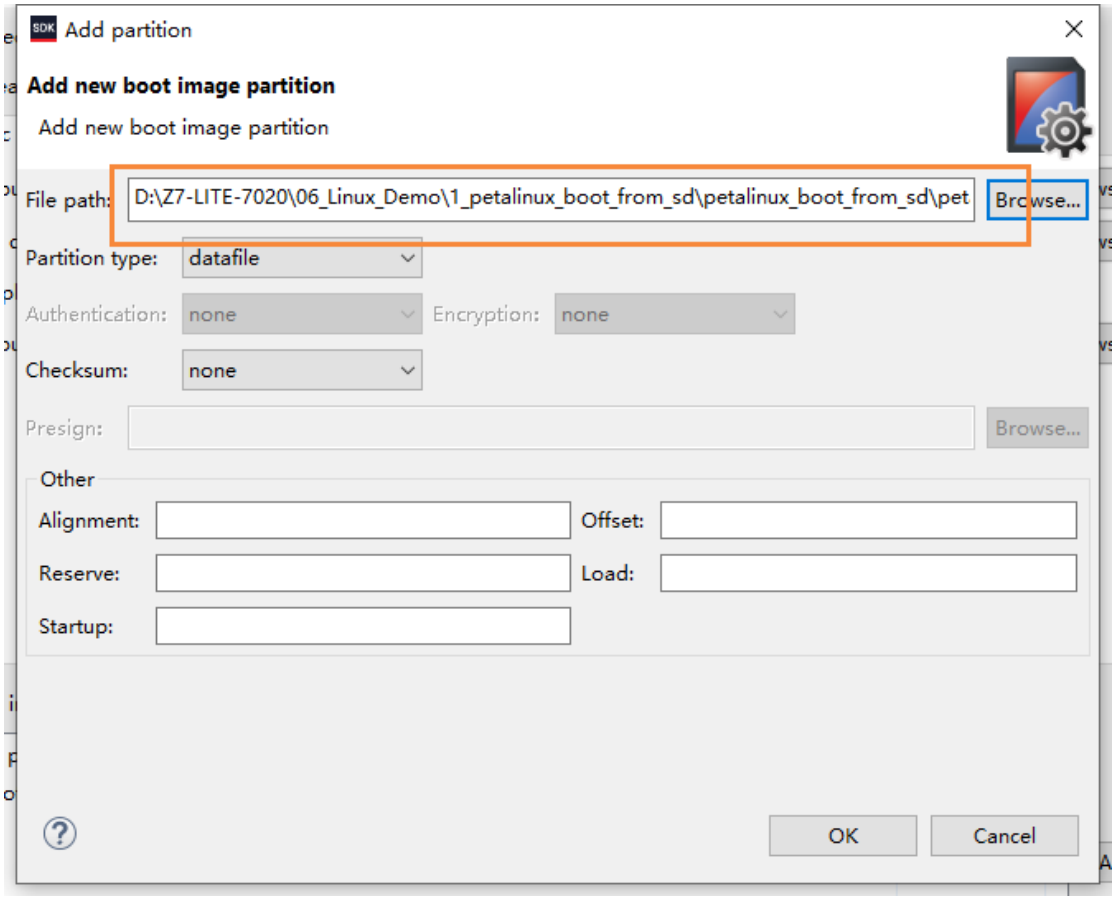


3) 添加 fsbl 文件，定位到当前工程下的 zynq_fsbl.elf，分区类型选择 bootloader，点击 OK。

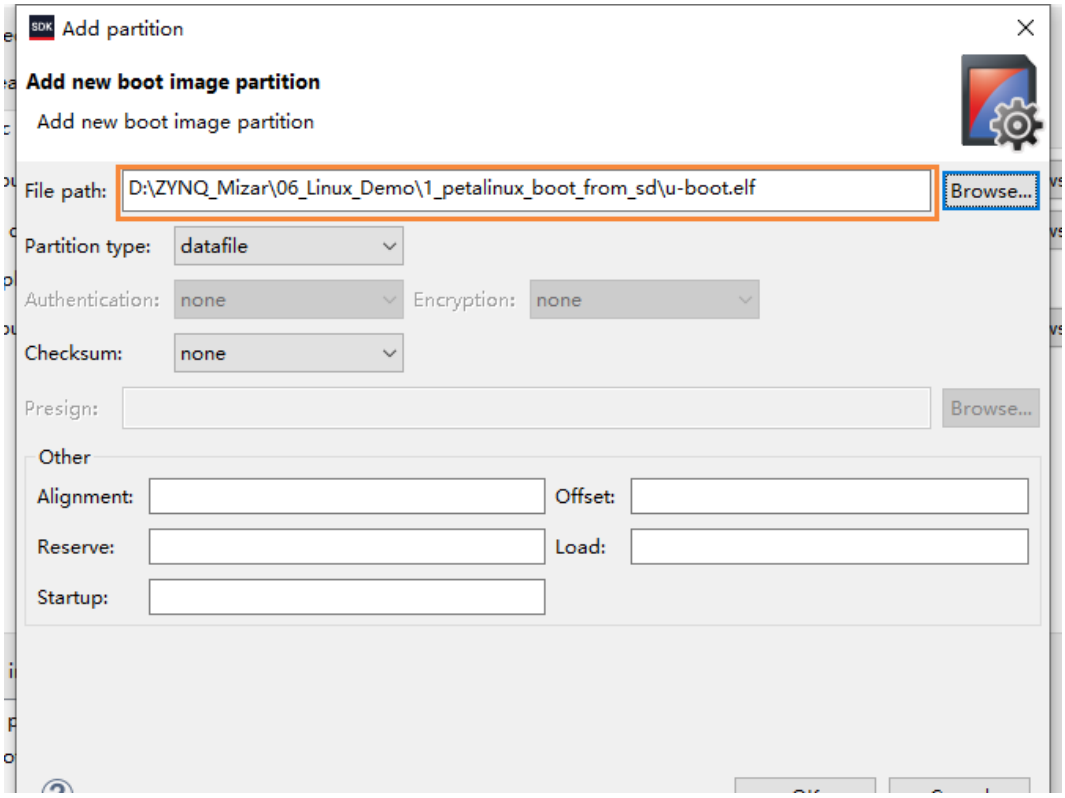


4) 添加 FPGA 比特流文件，此时可以使用 petalinux 工程路径 /home/microphase/Petalinux_proj/petalinux_boot_from_sd/project-spec/hw-description 下的 system_wrapper.bit，或者 Vivado 工程.sdk 文件夹内的 petalinux_boot_from_sd_wrapper.bit。分区类型选择 datafile，点击 OK。

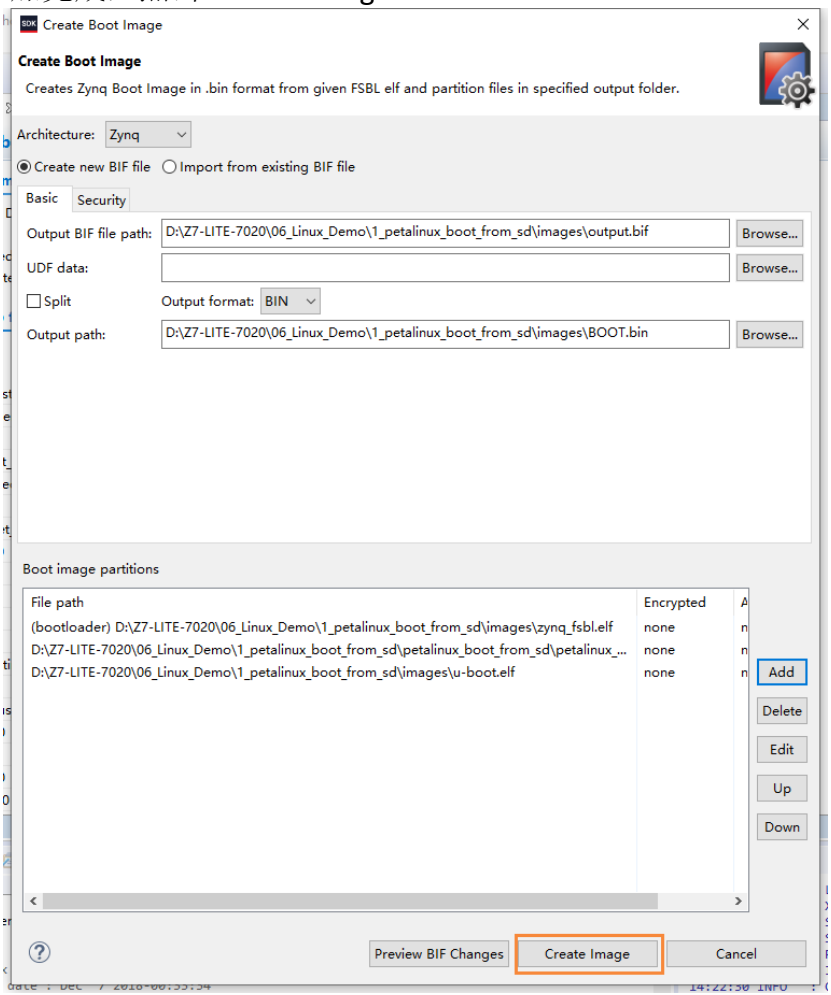




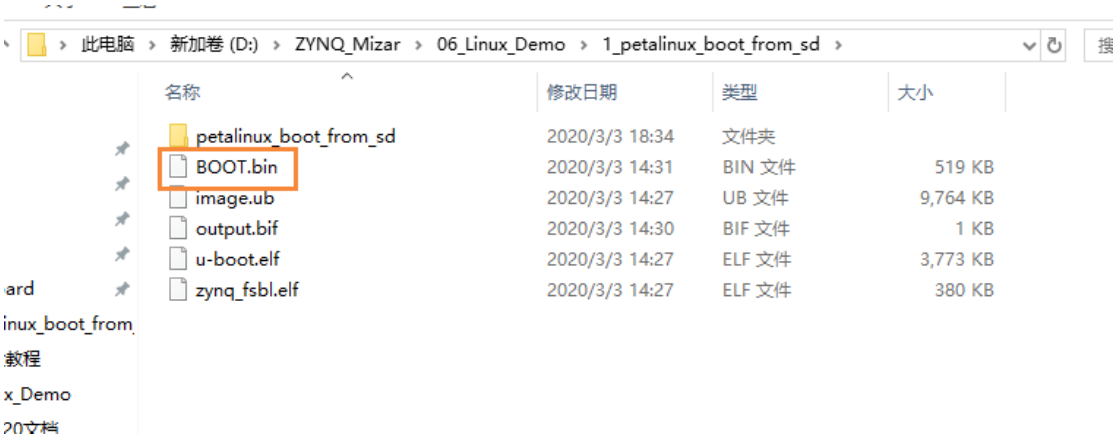
5) 添加 u-boot 文件，定位到当前目录下的 u-boot.elf，分区类型选择 datafile 点击 OK。



6) 添加完成，点击 Create Image。



7) 完成后，将在工程目录下生成生成打包文件 BOOT.bin。

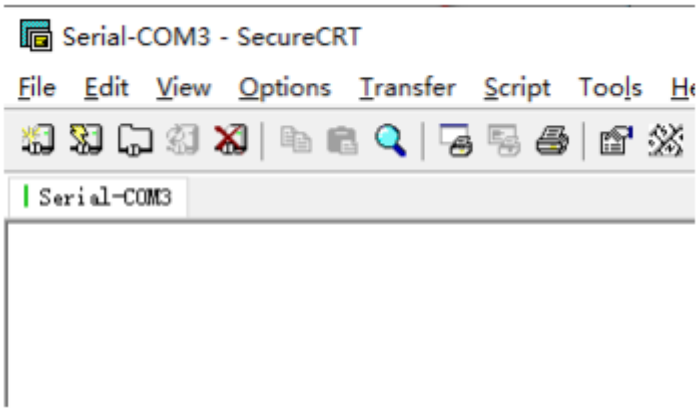


8) 拷贝 BOOT.bin 和 image.ub 至 SD 卡。先将 SD 卡格式化清空，然后再拷贝。拷贝后如图所示。

5.5 SD 卡启动

1) 将开发板的拨码开关调整至 SD 卡启动。

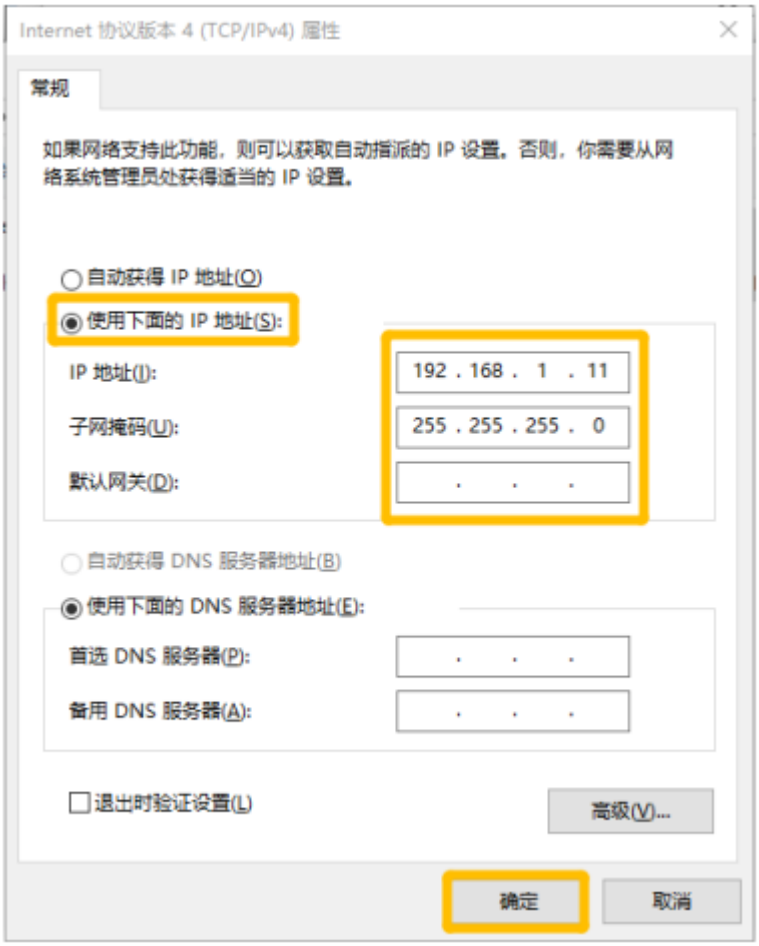
2) 连接电脑的 USB 和开发板的 UART 口，连接成功。



3) 连接电脑网口和开发板 PS 端网口，配置电脑网口，右键，点击属性。

4) 双击 IPv4。

5) 选择“使用下面的 IP 地址”，配置 IP 地址为 192.168.1.11，子网掩码 255.255.255.0，点击确定。



6) 开发板上电，串口打印 Boot 信息如下，以太网正常工作，输入用户名 root，密码 root，登录。

```
 Removing any system startup links for run-postinsts ...
   /etc/rcS.d/S99run-postinsts
INIT: Entering runlevel: 5
Configuring network interfaces... IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
udhcpc (v1.24.1) started
Sending discover...
xilinx_axienet 41000000.ethernet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
Sending discover...
Sending select for 192.168.3.127...
Lease of 192.168.3.127 obtained, lease time 86400
/etc/udhcpc.d/50default: Adding DNS 192.168.3.1
done.
Starting Dropbear SSH server: Generating key, this may take a while...
Public key portion is:
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCAgwOs3pDhmiGsCGMA6AOuzJcxEQVKN3bgWaqw+r68QPLnuLkMCR5SbjXMnr
USNj85+79iEFMillPYI+Arj/OjgHI6Fr/SgZp1SVJjQl18z36cL3OBuSz1Nbl7EO7sfEIpNWnlOFnD5wKgJPfS/kyqYisOrOlM
/uwhbY/Vqa/pGipBL4EJInsUT9O5gIxTOhJqO89qRT4ZcnTDuNtmOhzpDxdmurwM5u74jl8APxcC5urwNFUyhcR/jJtKt5JH4I
tkFLjoUZJjwO/F root@petalinux_boot_from_sd
Fingerprint: md5 53:7d:f1:39:0e:e8:02:02:8e:18:7e:9e:5a:8a:5d:79
dropbear.
hwclock: can't open '/dev/misc/rtc': No such file or directory
Starting syslogd/klogd: done
Starting tcf-agent: OK

PetaLinux 2018.3 petalinux_boot_from_sd /dev/ttyPS0

petalinux_boot_from_sd login: root
Password:
root@petalinux_boot_from_sd:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0A:35:00:22:01
          inet addr:192.168.3.127  Bcast:192.168.3.255  Mask:255.255.255.0
```

7) 输入下列命令，设置开发板 IP 地址为 192.168.1.10，子网掩码 255.255.255.0，设置完成后，尝试 ping 通，可见连接已经成功。

```
root@petalinux_boot_from_sd:~# ifconfig eth0 inet 192.168.1.10
root@petalinux_boot_from_sd:~# ifconfig eth0 netmask 255.255.255.0
root@petalinux_boot_from_sd:~#
root@petalinux_boot_from_sd:~# ping 192.168.1.11
PING 192.168.1.11 (192.168.1.11): 56 data bytes
64 bytes from 192.168.1.11: seq=0 ttl=128 time=0.585 ms
64 bytes from 192.168.1.11: seq=1 ttl=128 time=0.293 ms
64 bytes from 192.168.1.11: seq=2 ttl=128 time=0.293 ms
64 bytes from 192.168.1.11: seq=3 ttl=128 time=0.343 ms
64 bytes from 192.168.1.11: seq=4 ttl=128 time=0.350 ms
64 bytes from 192.168.1.11: seq=5 ttl=128 time=0.349 ms
64 bytes from 192.168.1.11: seq=6 ttl=128 time=0.276 ms
64 bytes from 192.168.1.11: seq=7 ttl=128 time=0.349 ms
64 bytes from 192.168.1.11: seq=8 ttl=128 time=0.357 ms
64 bytes from 192.168.1.11: seq=9 ttl=128 time=0.356 ms
64 bytes from 192.168.1.11: seq=10 ttl=128 time=0.346 ms
64 bytes from 192.168.1.11: seq=11 ttl=128 time=0.288 ms
```

注意：如果出现只能单向 ping 通，或完全 ping 不通的情况，请检查电脑端网卡，将无线网卡、虚拟网卡全部禁掉，只留一个物理网卡接口，并在控制面板中禁止 Windows 防火墙，即可解决。

8) 在电脑端， ping 开发板，也是可以接通。

```
C:\Windows\system32\cmd.exe - ping -t 192.168.1.10
Microsoft Windows [版本 10.0.17763.737]
(c) 2018 Microsoft Corporation。保留所有权利。

C:\Users\mp02>ping -t 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=9ms TTL=64
来自 192.168.1.10 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.1.10 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.1.10 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.1.10 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.1.10 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.1.10 的回复: 字节=32 时间<1ms TTL=64
```

5.6 实验小结

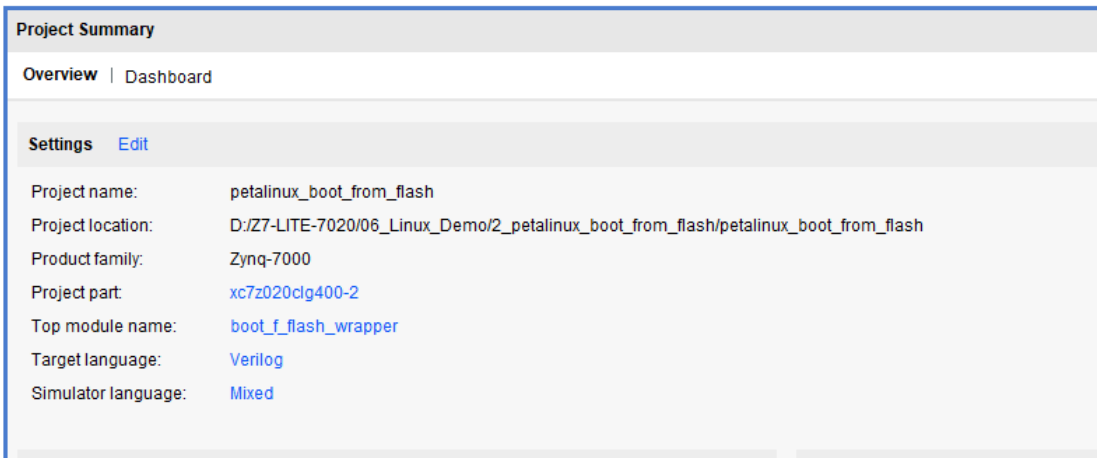
本章主要讲解如何利用 Petalinux 实现 SD 卡启动，对于 PetaLinux 的开发流程，读者要仔细理解。

第6章 PetaLinux 定制 Linux(QSPI 启动)

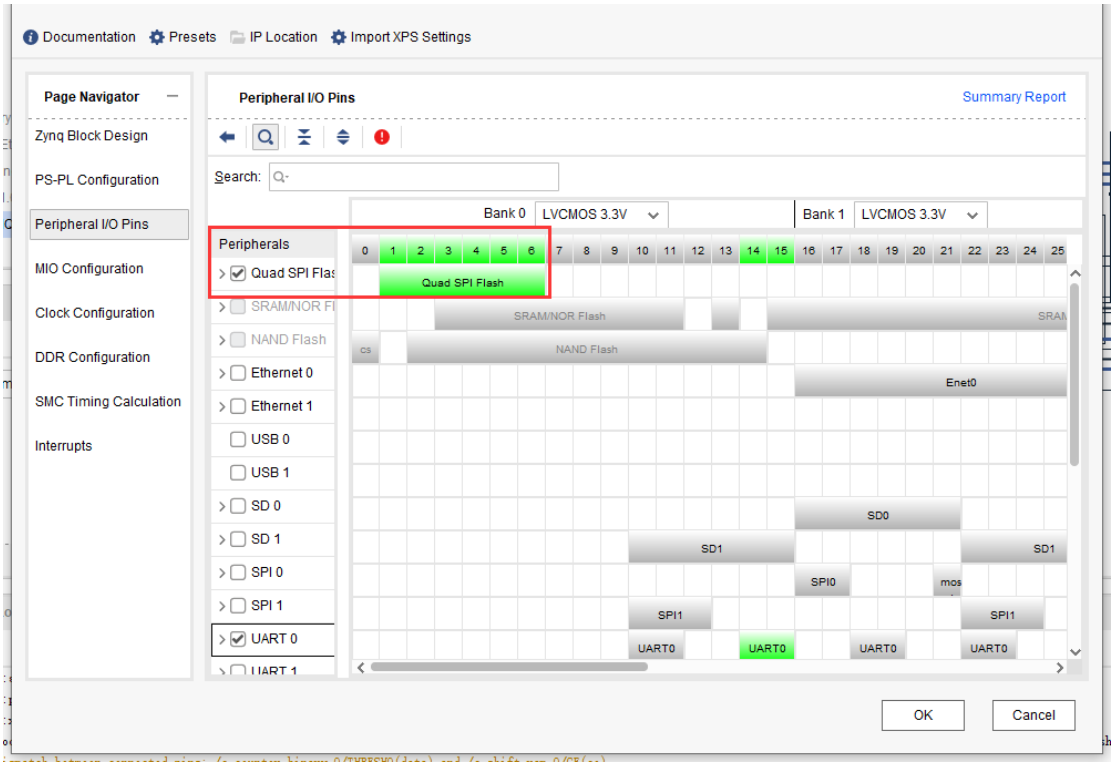
本章主要讲解从 QSPI-Flash 启动 PetaLinux。主要实现的功能是： PS 端激活外设 SD，UART(用于打印 log)， PL 端添加网络控制 IP，最终实现 PetaLinux 可以从 Flash 启动，并且 Windows 主机可以通过 PL 端网口和开发板的 Linux 系统通信。 对应的代码工程目录： **2_petalinux_boot_from_flash**。

6.1 Vivado 创建工程

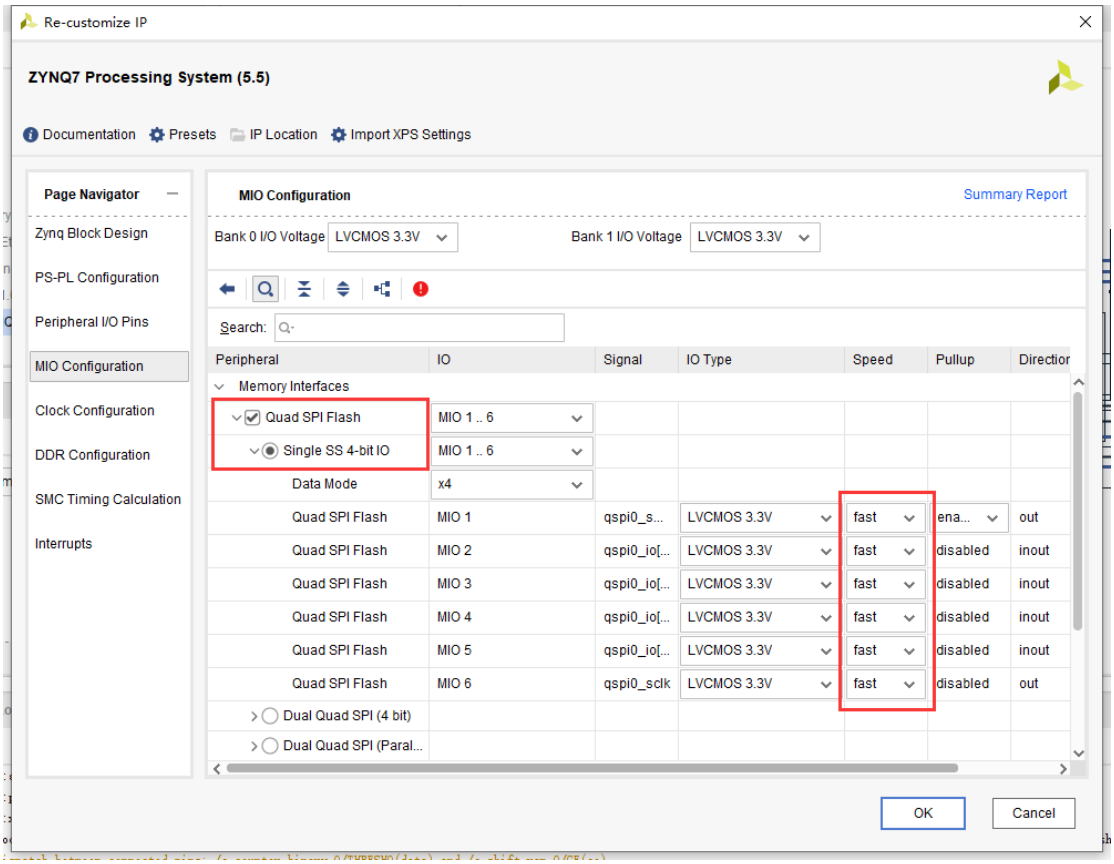
1) 本次实验工程与上一实验基本相同，只需修改少部分设置，所有打开上一实验的 Viv 阿斗工程，将其另存为一个新的工程，另存为之后的新工程如下图所示，删除新工程目录下的.sdk 文件夹。



2) 之后点击 Open Block Design 打开设计文件，双击 ZYNQ 模块。 点击左侧 Peripheral I/O Pins，选择 MIO1-MIO6 的 Quad SPI Flash。



3) 点击左侧 MIO Configuration，Quad SPI Flash 的 Speed 选择 fast。



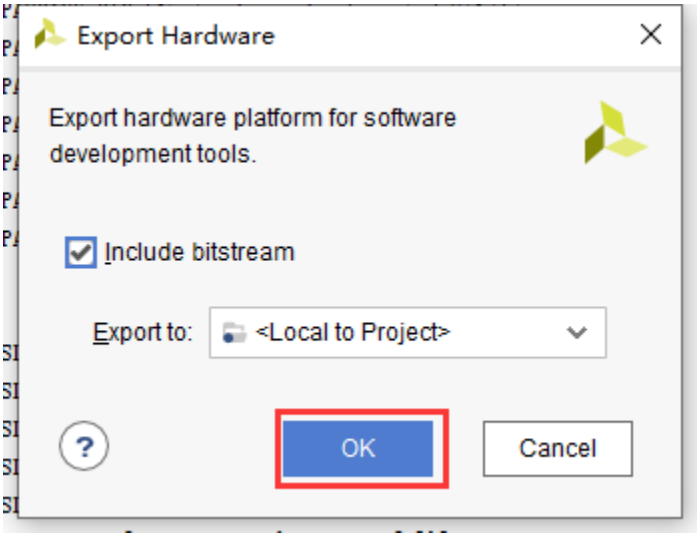
7)保存设置。

8)重新生成系统设计代码和顶层文件。

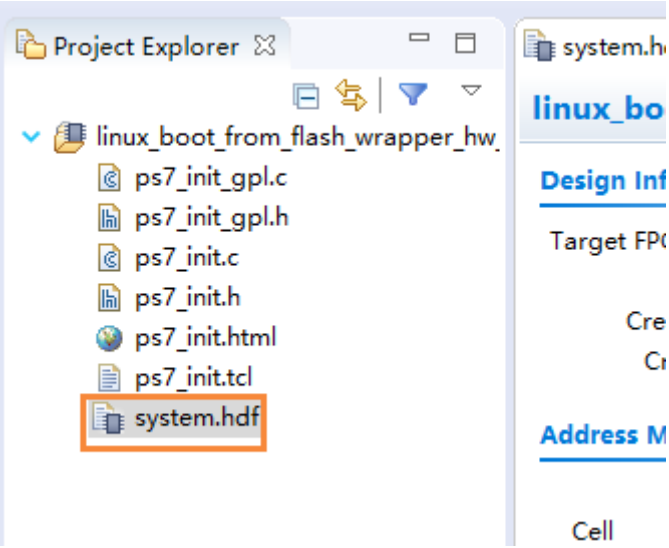
9)重新综合，编译生成 Bitstream 文件

6.2 设计导出 SDK

1) 编译完成后导出硬件描述文件。点击 File-->Export-->Export Hardware，点击 OK。

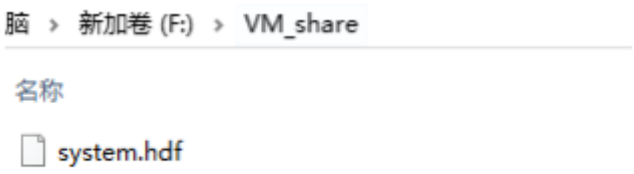


2) 打开 SDK 工程如下，包含系统硬件描述文件 system.hdf。



3) 将目录 06_Linux_Demo\2_petalinux_boot_from_flash\linux_boot_from_flash\linux_boot_from_flash.sdk\linux_boot_from_flash_wrapper_hw_platfo

rm_0 拷贝到共享文件夹 F:\VM_share 下，PetaLinux 需要用到这个硬件描述语言。



6. 3 PetaLinux 工程建立

1) 打开 VitrualBox 的 Ubuntu Linux 系统，执行下列命令，挂载 VM_share 到目录/mnt。

```
sudo mount –t vboxsf VM_share /mnt
```

```
micriophase@micriophase-VirtualBox:~$ sudo mount -t vboxsf VM_share /mnt
```

2) 执行下列命令，使 PetaLinux 工具可以使用。

```
source /opt/pkg/petalinux/2018.3/settings.sh
```

```
micriophase@micriophase-VirtualBox:~$ source /opt/pkg/petalinux/2018.3/settings.sh
PetaLinux environment set to '/opt/pkg/petalinux/2018.3'
INFO: Checking free disk space
INFO: Checking installed tools
INFO: Checking installed development libraries
INFO: Checking network and other services
```

3) 进入上一节建立的文件夹 PetaLinux_proj，执行下列命令创建 PetaLinux 工程。

```
cd PetaLinux_proj

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

```
micriophase@micriophase-VirtualBox:~$ cd PetaLinux_proj
```

```
micriophase@micriophase-VirtualBox:~/PetaLinux_proj$ petalinux-create --type proje
ct --template zynq --name petalinux_boot_from_flash
INFO: Create project: petalinux_boot_from_flash
INFO: New project successfully created in /home/micriophase/PetaLinux_proj/petalin
ux_boot_from_flash
```

4) 进入该目录，执行下列命令将/mnt 下的 system.hdf 文件拷贝至当前目录。

```
cd petalinux_boot_froom_flash

cp /mnt/system.hdf .
```

```
micriophase@micriophase-VirtualBox:~/PetaLinux_proj$ cd petalinux_boot_from_flash
micriophase@micriophase-VirtualBox:~/PetaLinux_proj/petalinux_boot_from_flash$ cp /mnt/system.hdf .
```

5) 执行下列命令，指定导入硬件描述文件 system.hdf 的路径，本例就是当前路径。

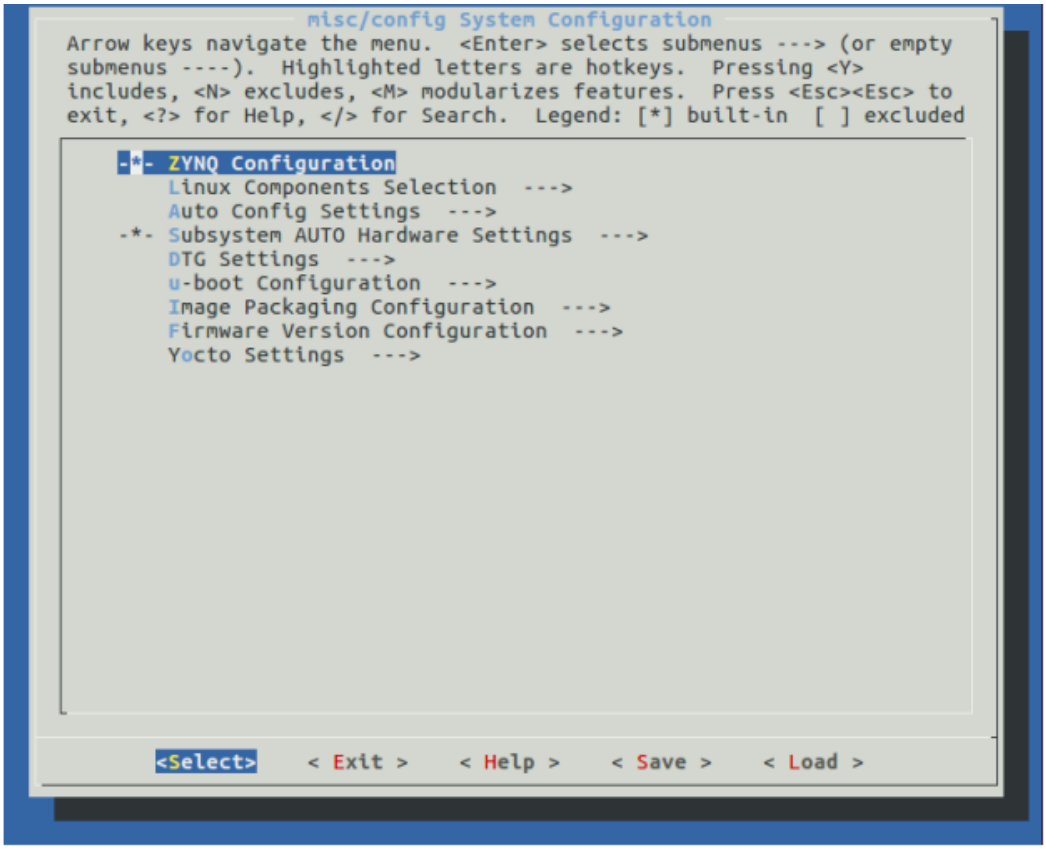
```
petalinux-config --get-hw-description=.
```

```
micriophase@micriophase-VirtualBox:~/PetaLinux_proj/petalinux_boot_from_flash$
petalinux-config --get-hw-description=.
```

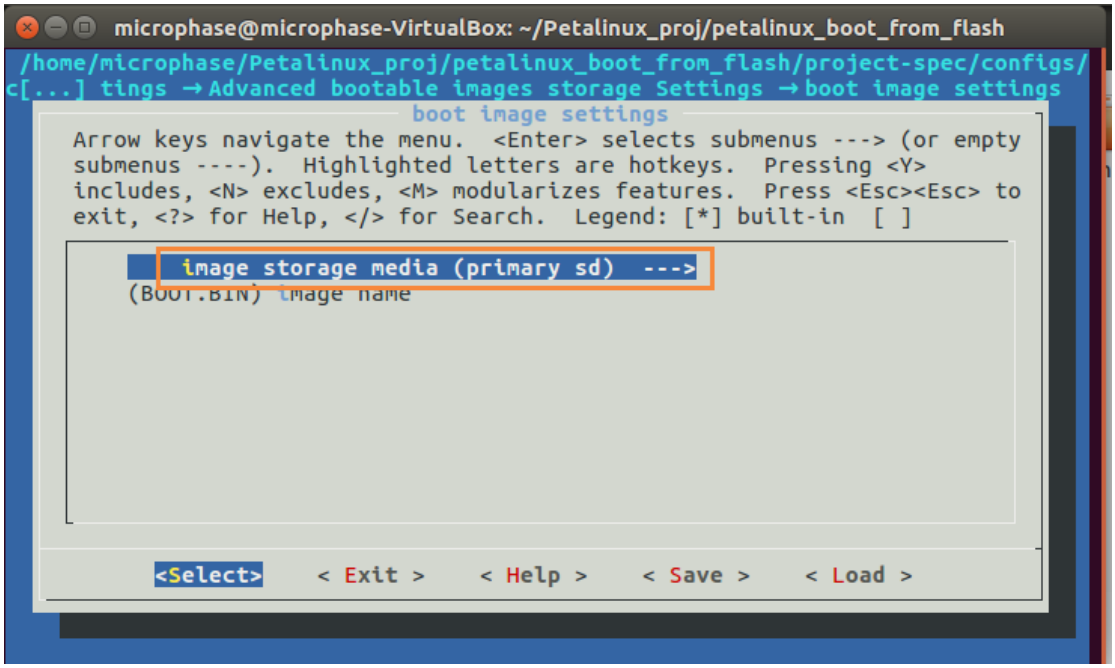
6) 打开配置界面。

```
petalinux-config
```

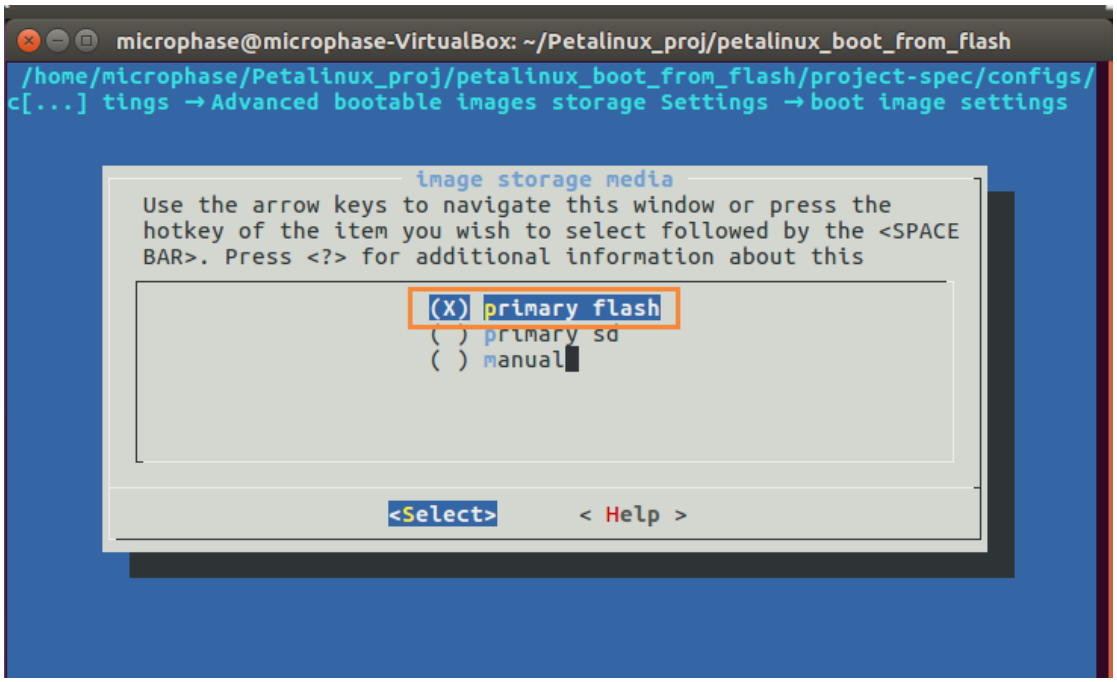
```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_boot_from_flash$ pet
alinux-config
[INFO] generating Kconfig for project
[INFO] menuconfig project
```



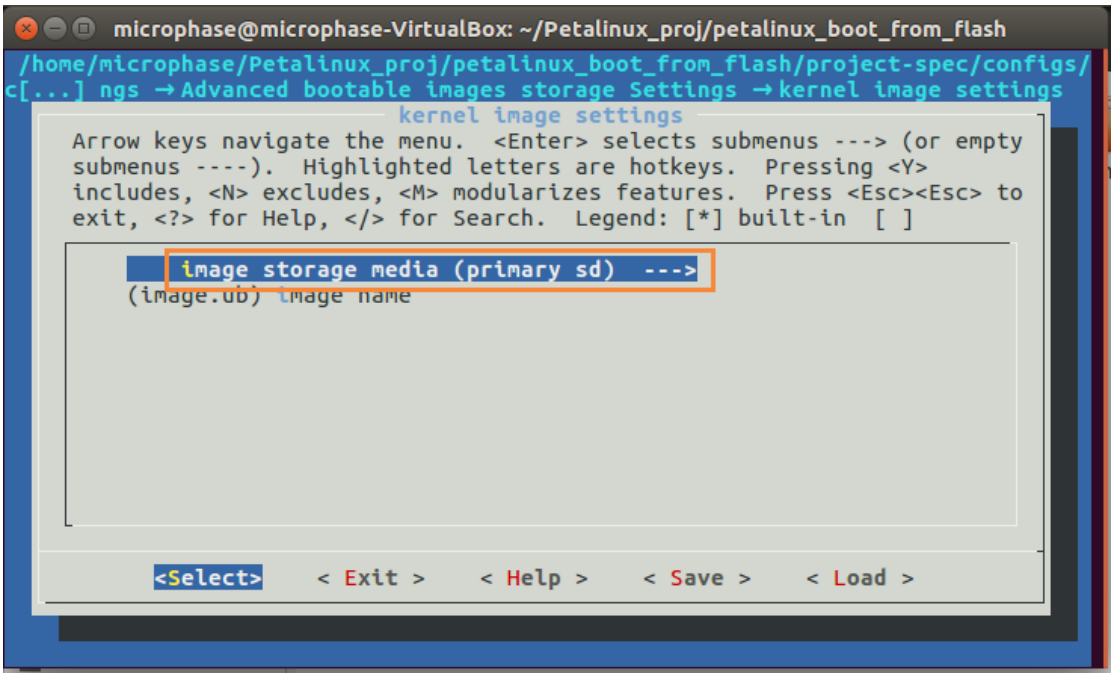
可点击方向键移动光标，点击 Enter 进入某一条目查看详情，按两次 Esc 退回至上一层。进入 Subsystem AUTO Hardware Settings ---> Advanced bootable images storage Settings --->boot image settings ---> image storage media 选项，



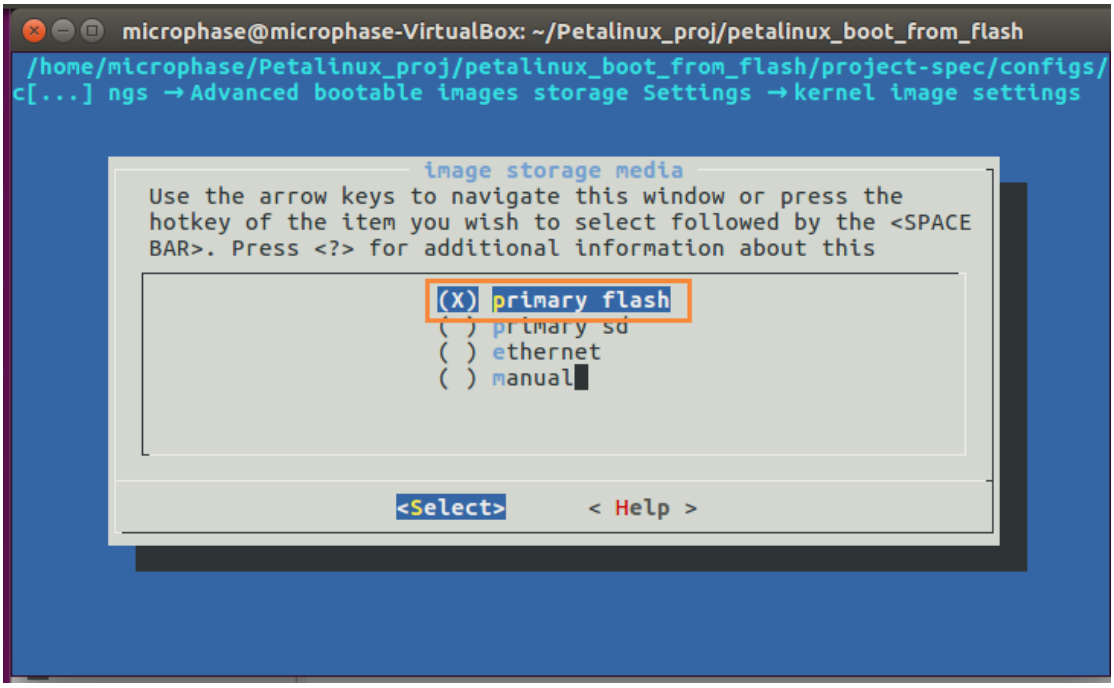
选择 primary flash，



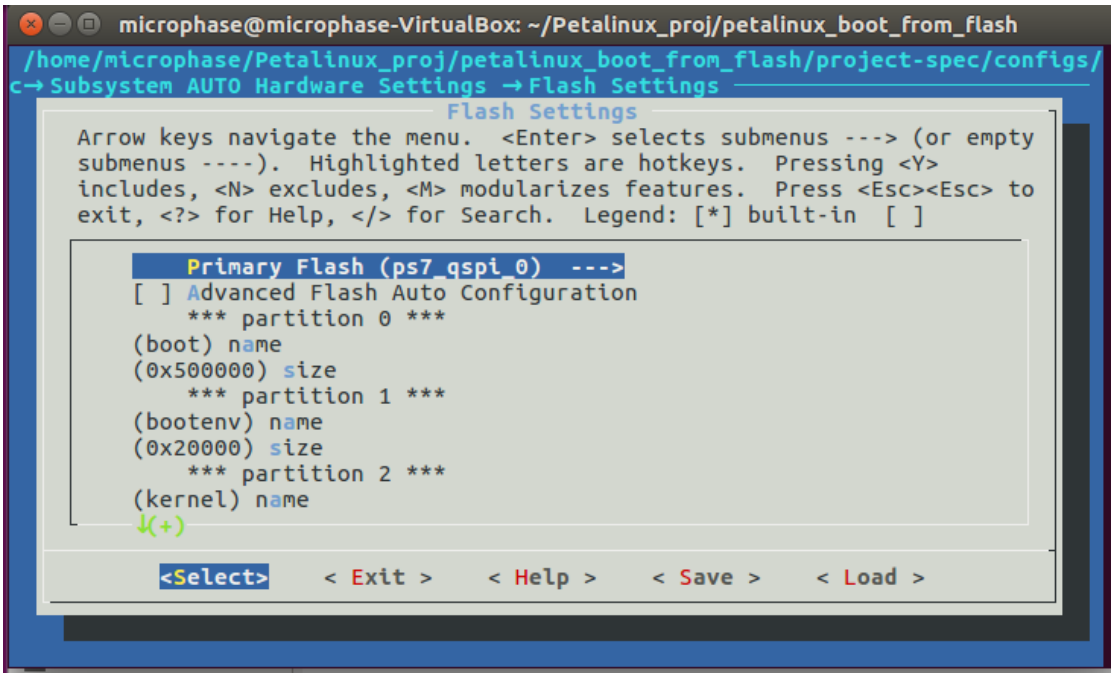
返回上级菜单，进入 Subsystem AUTO Hardware Settings ---> Advanced bootable images storage Settings --->kernel image settings ---> image storage media 选项



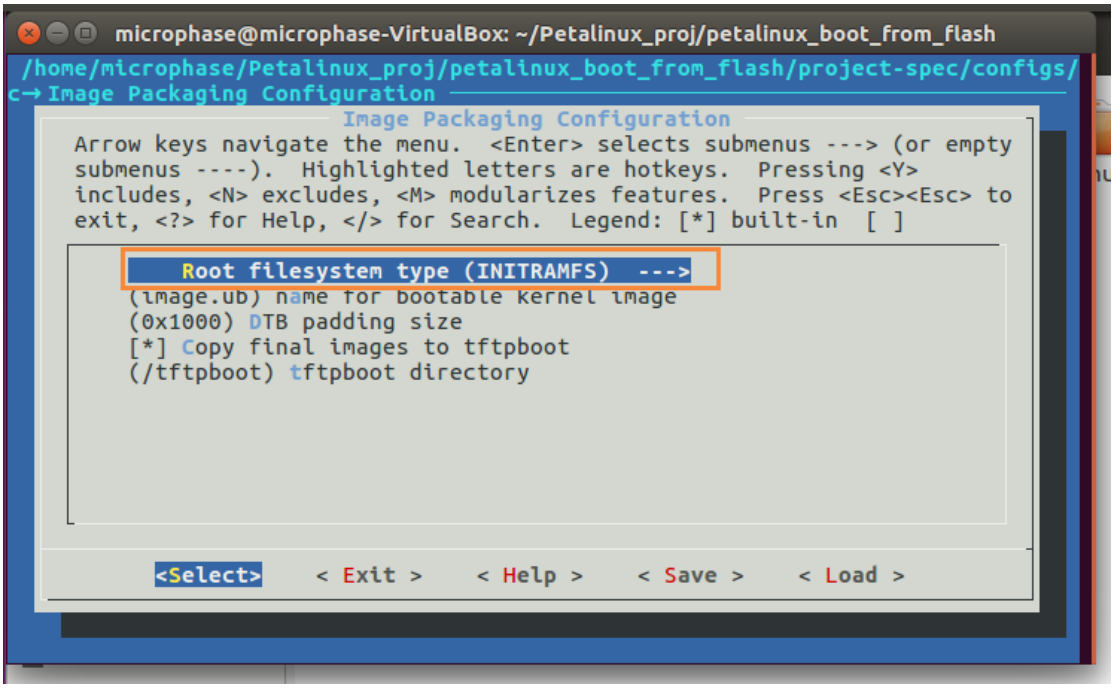
选择 primary flash



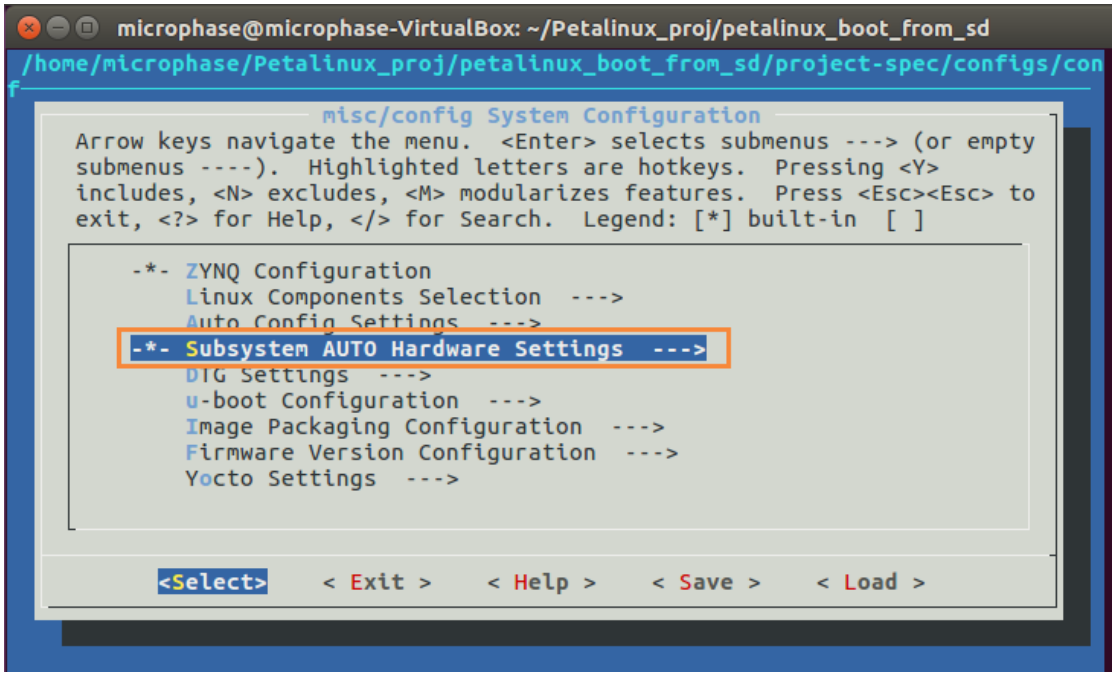
进入 Subsystem AUTO Hardware Settings → Flash Settings 中可以修改 QSPI flash 的分区，默认的就差不多可以使用，如果你的文件大小超过了默认分区的大小，可自行调整。



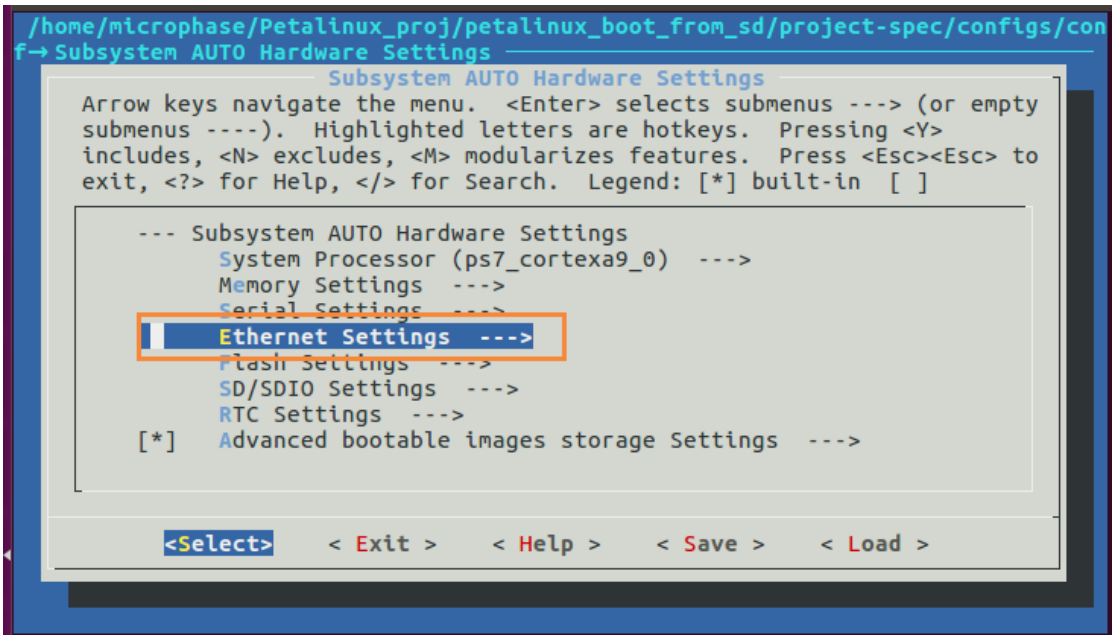
在 Image Packaging Configuration ---> Root filesystem type 选择 INITRAMFS，使用 RAM 类型的根文件系统，这样就可以方便打包烧写到 QSPI Flash



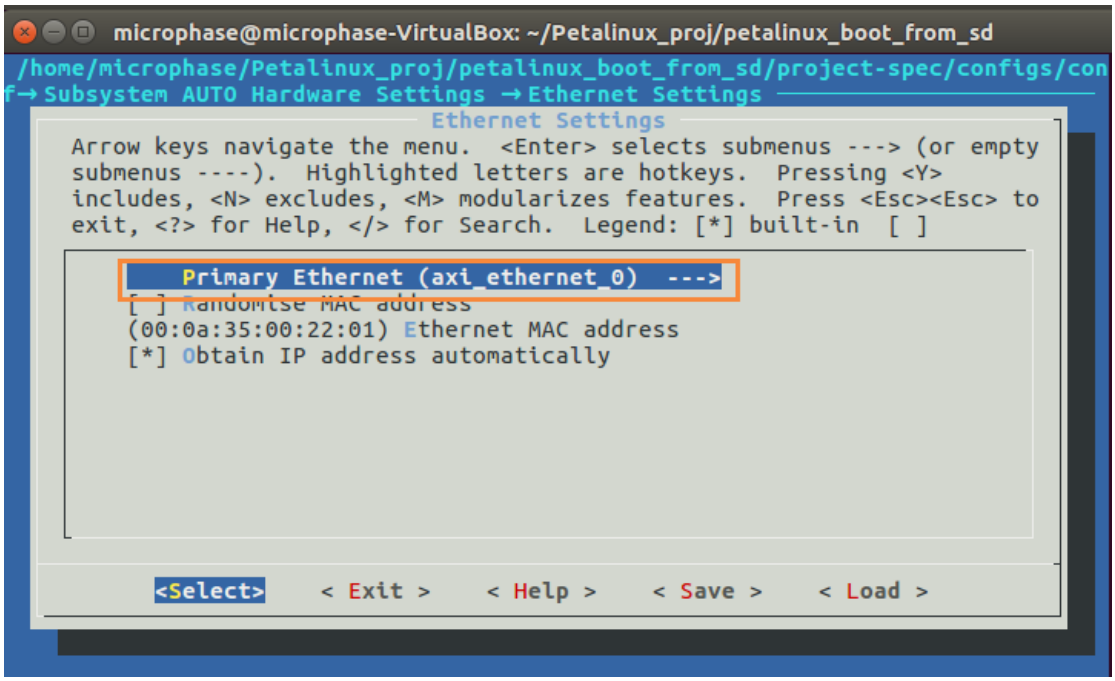
进入 Subsystem AUTO Hardware Settings



进入 Ethernet Settings，

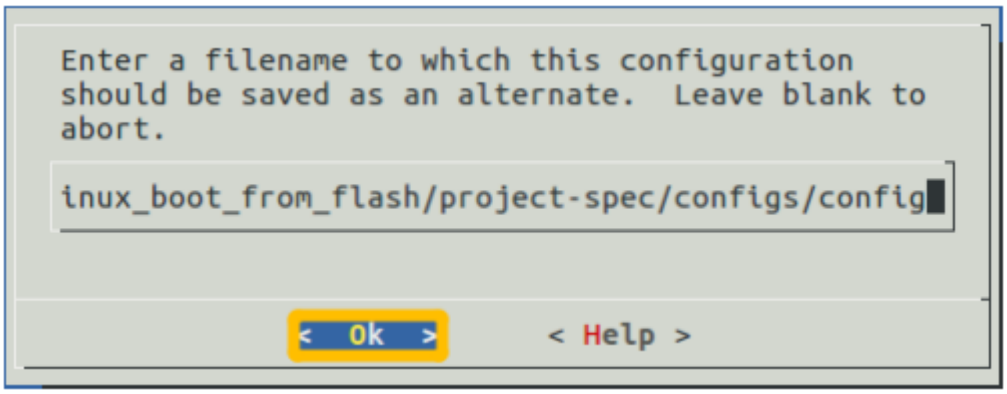


设置 Primary Ethernet 为 axi_ethernet_0

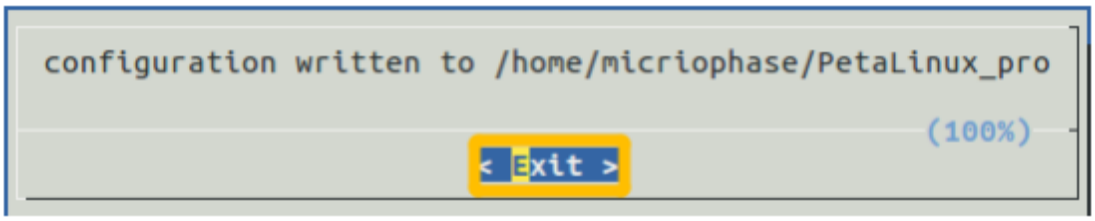


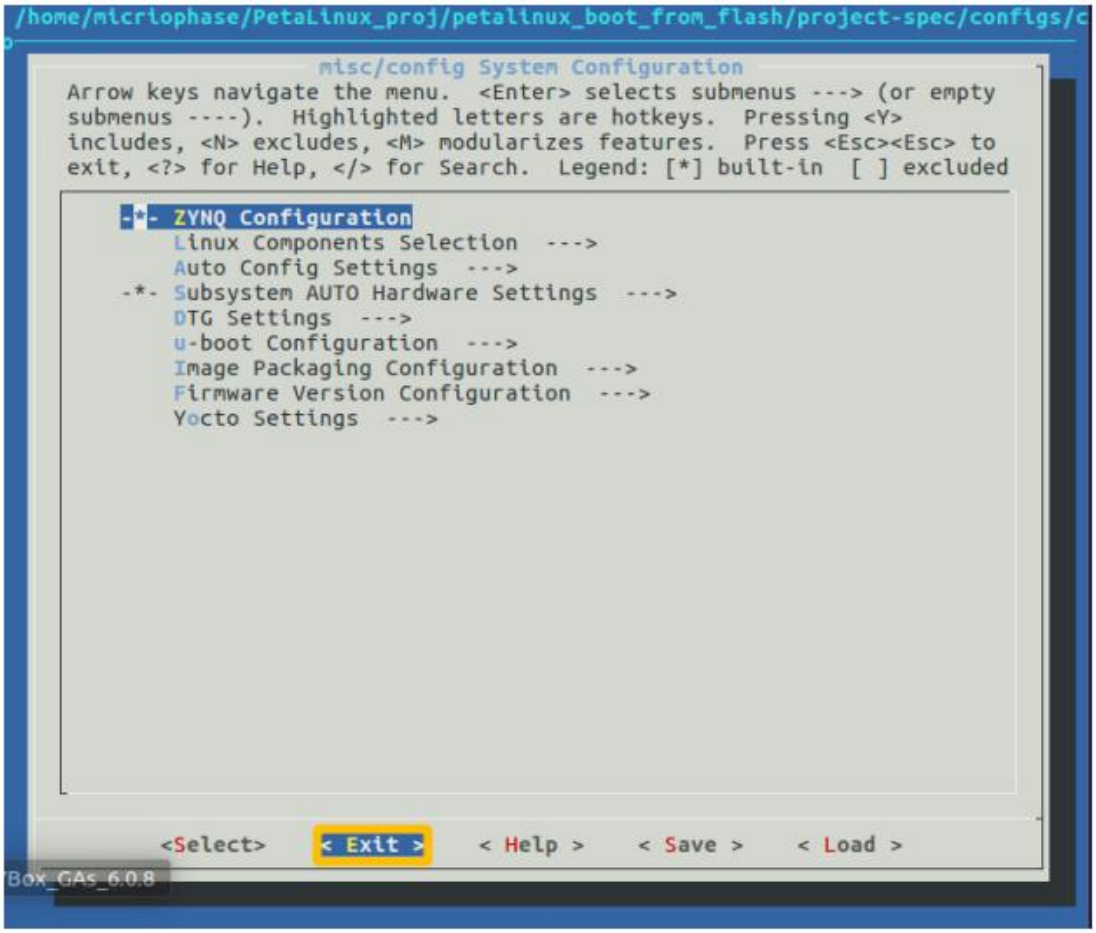
设置修改完成后回到主菜单，选择 Save，回车，

选择 OK，回车，



配置写入完成 Exit，回车，





主窗口选择 Exit，回车。

7) 执行下列命令，编译工程。

```
petalinux-build
```

```
microphase@microphase-VirtualBox:~/PetaLinux_proj/petalinux_boot_from_flash$
petalinux-build
[INFO] building project
[INFO] sourcing bitbake
INFO: bitbake petalinux-user-image
Parsing recipes: 100% |#######################################| Time: 0:02:01
Parsing of 2569 .bb files complete (0 cached, 2569 parsed). 3445 targets, 149
skipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies
Initialising tasks: 100% |####################################| Time: 0:00:14
Checking sstate mirror object availability: 100% |#############| Time: 0:04:10
NOTE: Executing SetScene Tasks
NOTE: Executing RunQueue Tasks
WARNING: petalinux-user-image-1.0-r0 do_rootfs: [log_check] petalinux-user-ima
ge: found 1 warning message in the logfile:
[log_check] warning: %post(sysvinit-inittab-2.88dsf-r10.plnx_zynq7) scriptlet
failed, exit status 1

ystem Settings  ummary: Attempted 3065 tasks of which 2246 didn't need to be reru
                ceeded.

Summary: There was 1 WARNING message shown.
INFO: Copying Images from deploy to images
INFO: Creating images/linux directory
NOTE: Failed to copy built images to tftp dir:  /tftpboot
[INFO] successfully built project
```

8) 编译完成后，在当前目录下生成 images 文件夹。执行下列命令将 zynq_fsbl.elf, system_wrapper.bit, u-boot.elf, image.ub 四个文件打包生成 BOOT.bin 文件

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

```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_boot_from_flash$ pet
alinux-package --boot --fsbl ./images/linux/zynq_fsbl.elf --u-boot --fpga --kern
el --force
INFO: Getting system flash information...
INFO: File in BOOT BIN: "/home/microphase/Petalinux_proj/petalinux_boot_from_fla
sh/images/linux/zynq_fsbl.elf"
INFO: File in BOOT BIN: "/home/microphase/Petalinux_proj/petalinux_boot_from_fla
sh/project-spec/hw-description/boot_f_flash_wrapper.bit"
INFO: File in BOOT BIN: "/home/microphase/Petalinux_proj/petalinux_boot_from_fla
sh/images/linux/u-boot.elf"
INFO: File in BOOT BIN: "/home/microphase/Petalinux_proj/petalinux_boot_from_fla
sh/images/linux/image.ub"
INFO: Generating Zynq binary package BOOT.BIN...


****** Xilinx Bootgen v2018.3
  **** Build date : Nov 15 2018-19:22:29
    ** Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.

INFO: Binary is ready.
WARNING: Unable to access the TFTPBOOT folder /tftpboot!!!
WARNING: Skip file copy to TFTPBOOT folder!!!
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_boot_from_flash$
```

9) 执行下列命令，将其拷贝到/mnt 下。

```
cp -r images/ /mnt
```

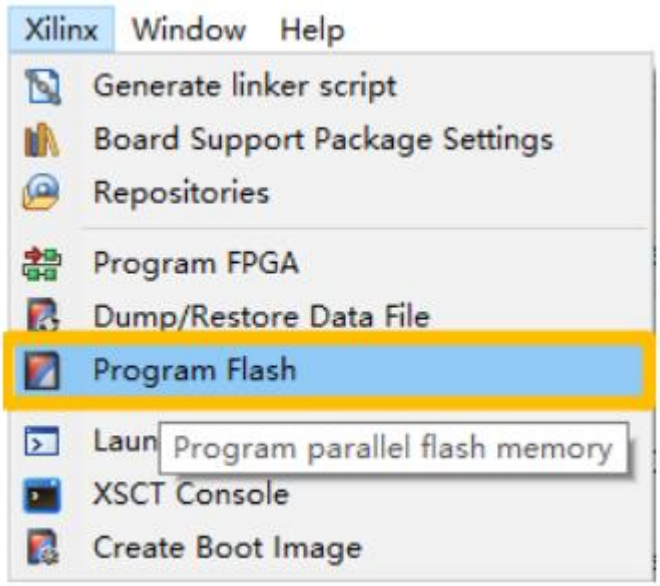
```
microphase@microphase-VirtualBox:~/PetaLinux_proj/petalinux_boot_from_flash$
cp -r images/ /mnt/
```

10) 在 Windows 共享文件夹 VM_share 下可以看到 images 文件夹了。

11) 将文件 BOOT.bin、u-boot.elf、 zynq_fsbl.elf、 image.ub 复制到当前工程目录下。

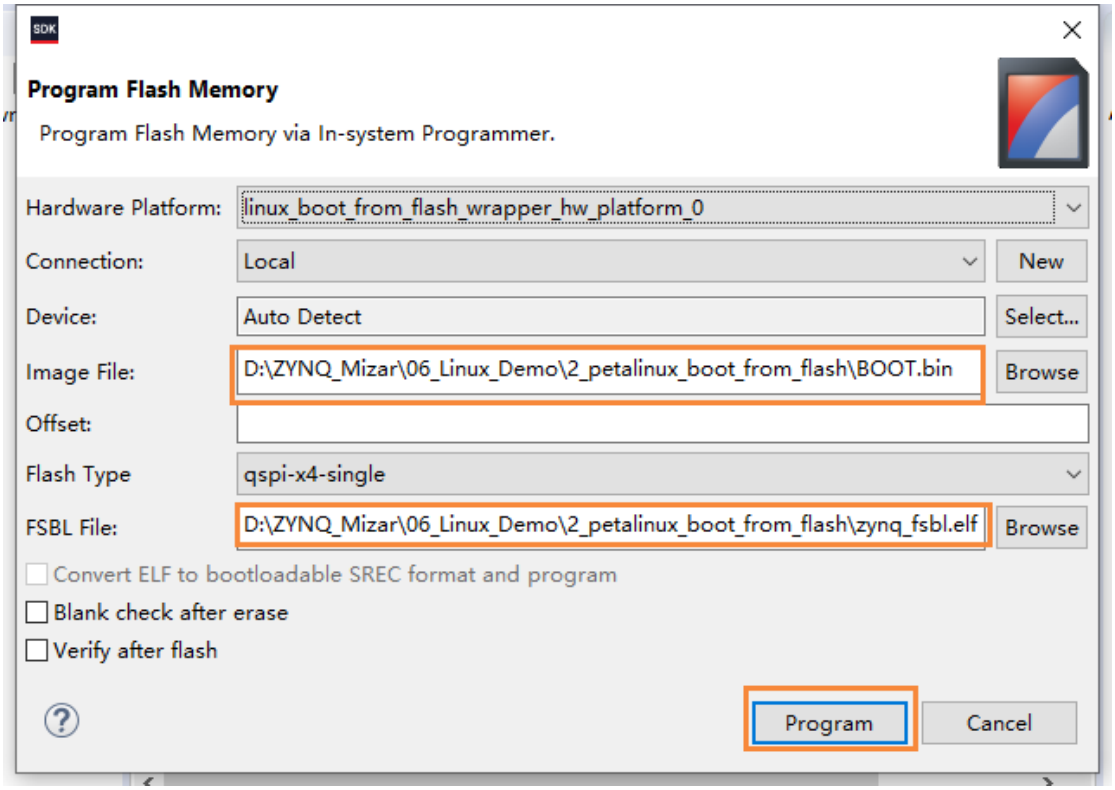
6. 4 BOOT.bin 文件烧录

1) 开发板上电，连接电脑 USB 口和开发板的 JTAG 口，点击菜单栏 Xilinx Tools-->Program Flash。

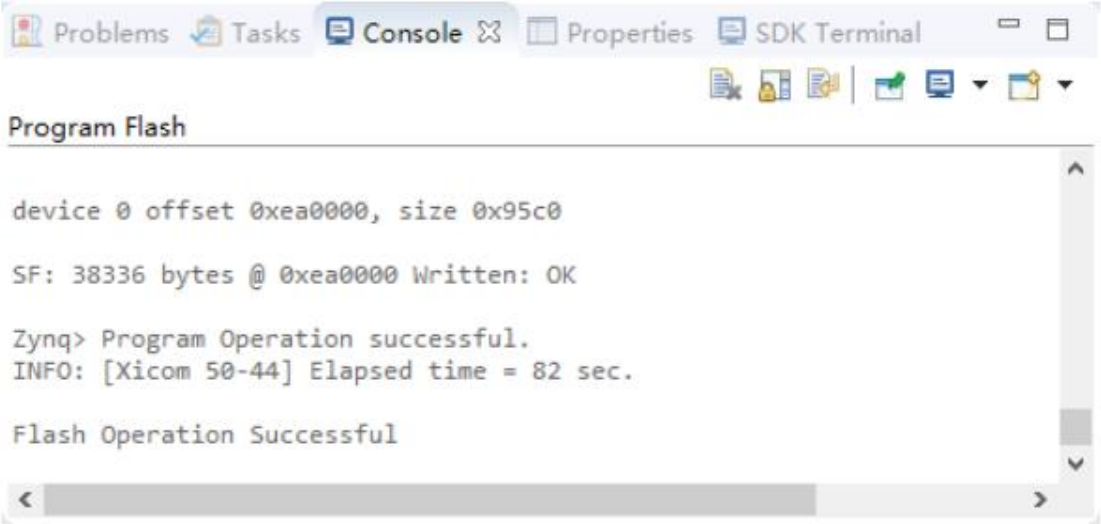


2) Image File 选择工程工程目录下的 BOOT.bin， FSBL File 选择工程目录下

的 zynq_fsbl.elf，点击 Program。



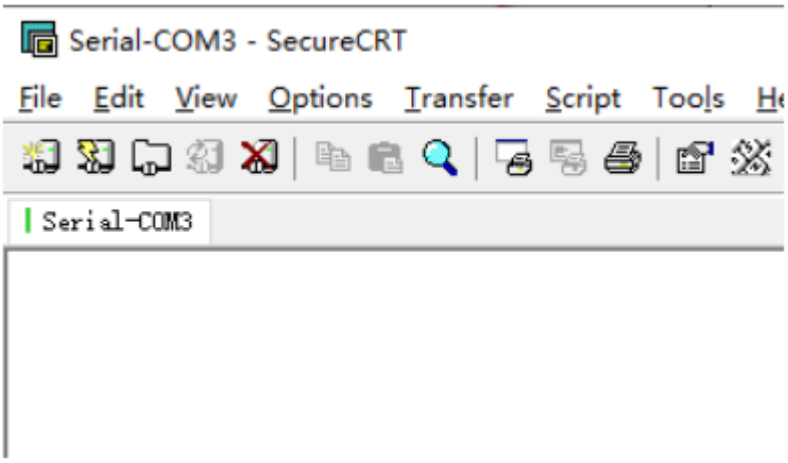
3) 通过 JTAG 口 Program QSPI-Flash 速度较慢，完成后 Console 窗口打印成功信息。



6.5 QSPI 启动

1) 将开发板跳线帽调整至 QSPI 启动。

2) 连接电脑的 USB 和开发板的 UART 口，连接成功。



3) 连接电脑网口和开发板 PS 端网口，配置电脑网口，右键，点击属性。

4) 双击 IPv4。

5) 选择“使用下面的 IP 地址”，配置 IP 地址为 192.168.1.11，子网掩码 255.255.255.0，点击确定。

6) 开发板上电，串口打印 Boot 信息如下，以太网正常工作，输入用户名 root，密码 root，登录。

```
 Removing any system startup links for run-postinstS ...
  /etc/rcS.d/S99run-postinsts
INIT: Entering runlevel: 5
Configuring network interfaces... IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
udhcpc (v1.24.1) started
Sending discover...
Sending discover...
macb e000b000.ethernet eth0: link up (1000/Full)
IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
Sending discover...
Sending select for 192.168.3.51...
Lease of 192.168.3.51 obtained, lease time 86400
/etc/udhcpc.d/50default: Adding DNS 192.168.3.1
done.
Starting Dropbear SSH server: Generating key, this may take a while...
Public key portion is:
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCTYOF3Y/iDP3/CNteS/n0qgL+AuskvcVXDxIlp0yV7ayx0GwrZFyLV85+lHZw84Y6Ztyha/UULfV5Vsa+A
HlHwJm4ugtivLyuEcFu7/GWDFPdYanwwltrbY5cCAQitL+sKRJ4f0xJncvVsU/0ASyYskG7lVNkwylfvwXJpXBvndutUPBO8C0UdEN1gXFsl9eArfJgy5Irv
ZOVG9bkGS44FeokFBW2p3edsy9LYU6px8m7ivCE2c/4wrCxHOmHuKMEdr4StpL+Q7ma49lsLQHQiS+DOzxwL1CZ5vJxF8u34/NBQqikVbfb89IOLfaIWUz/i
kXO/4p/+VxCBS2Dp6Czz root@petalinux_boot_from_flash
Fingerprint: md5 9e:ab:e2:c4:87:df:9e:34:ac:ce:78:df:a9:60:00:52
dropbear.
hwclock: can't open '/dev/misc/rtc': No such file or directory
Starting syslogd/klogd: done
Starting tcf-agent: OK

PetaLinux 2018.3 petalinux_boot_from_flash /dev/ttyPS0

petalinux_boot_from_flash login: root
Password:
root@petalinux_boot_from_flash:~#
Ready                    Serial: COM3   45, 35   45 Rows, 120 Cols   VT100   NUM
```

7) 输入下列命令，设置开发板 IP 地址为 192.168.1.10，子网掩码 255.255.255.0，设置完成后，尝试 ping 通，可见连接已经成功。

```
root@petalinux_boot_from_sd:~# ifconfig eth0 inet 192.168.1.10
root@petalinux_boot_from_sd:~# ifconfig eth0 netmask 255.255.255.0
root@petalinux_boot_from_sd:~#
root@petalinux_boot_from_sd:~# ping 192.168.1.11
PING 192.168.1.11 (192.168.1.11): 56 data bytes
64 bytes from 192.168.1.11: seq=0 ttl=128 time=0.585 ms
64 bytes from 192.168.1.11: seq=1 ttl=128 time=0.293 ms
64 bytes from 192.168.1.11: seq=2 ttl=128 time=0.293 ms
64 bytes from 192.168.1.11: seq=3 ttl=128 time=0.343 ms
64 bytes from 192.168.1.11: seq=4 ttl=128 time=0.350 ms
64 bytes from 192.168.1.11: seq=5 ttl=128 time=0.349 ms
64 bytes from 192.168.1.11: seq=6 ttl=128 time=0.276 ms
64 bytes from 192.168.1.11: seq=7 ttl=128 time=0.349 ms
64 bytes from 192.168.1.11: seq=8 ttl=128 time=0.357 ms
64 bytes from 192.168.1.11: seq=9 ttl=128 time=0.356 ms
64 bytes from 192.168.1.11: seq=10 ttl=128 time=0.346 ms
64 bytes from 192.168.1.11: seq=11 ttl=128 time=0.288 ms
```

注意：如果出现只能单向 ping 通，或完全 ping 不通的情况，请检查电脑端网卡，将无线网卡、虚拟网卡全部禁掉，只留一个物理网卡接口，并在控制面板中禁止 Windows 防火墙，即可解决。

8) 在电脑端， ping 开发板，也是可以接通。

6.6 实验小结

本章主要讲解如何利用 Petalinux 实现 QSPI-Flash 启动，对于 PetaLinux 的开发流程，读者要仔细理解。

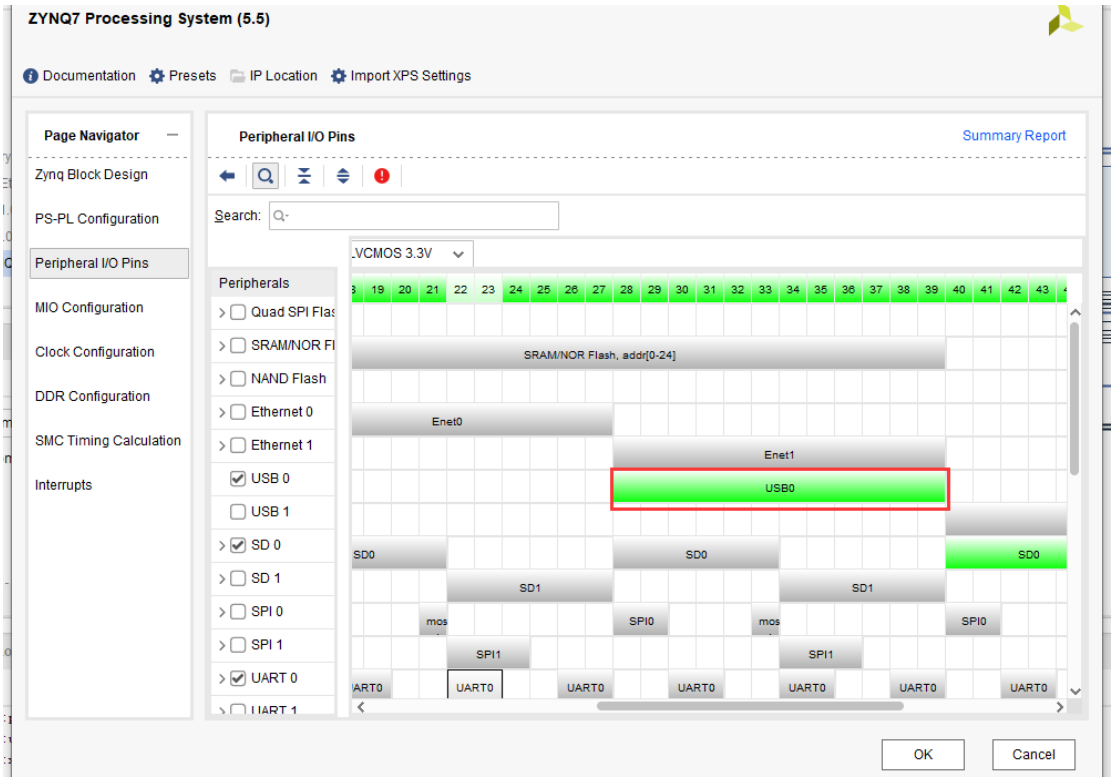
第7章　驱动 USB 外设

在之前的几章中，我们介绍了在开发板上启动 Linux 操作系统的方法，使用了 QSPI Flash、SD 卡等外设，在本章中，我们来学习怎样在 Linux 下驱动 USB 接口外设。

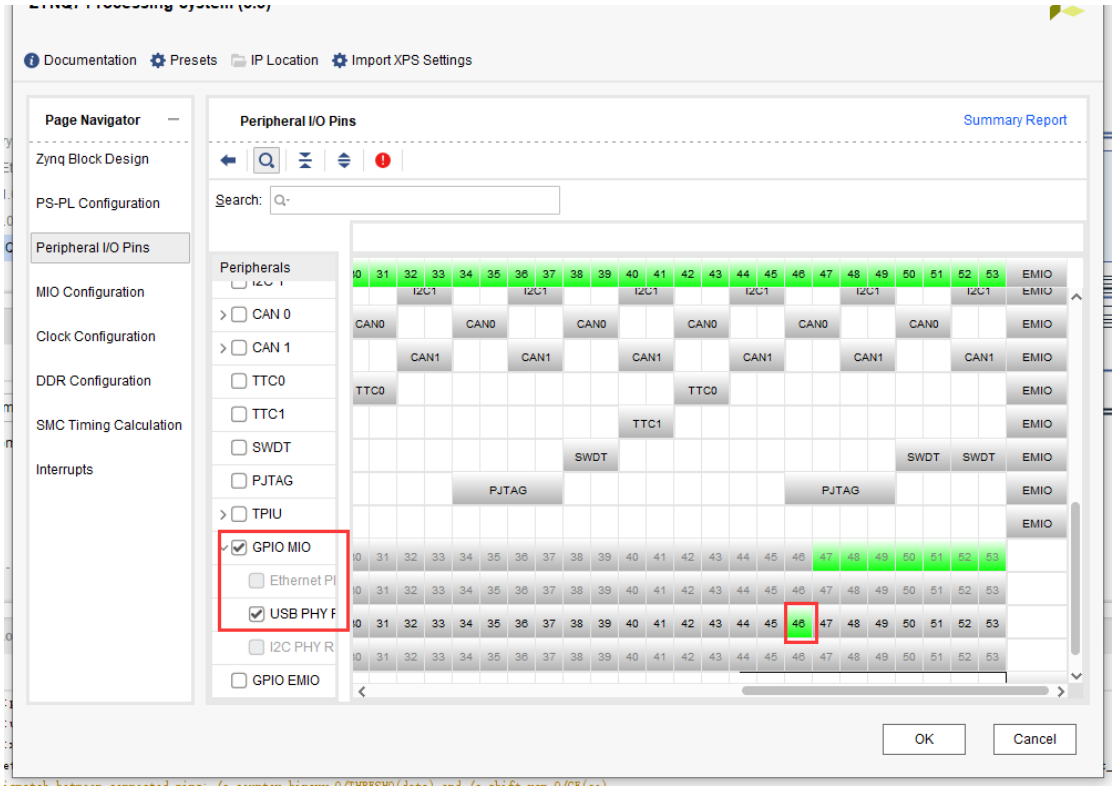
本实验要实现的功能是：PS 端激活外设 USB，SD，UART（用于打印 log），PL 端不做任何设计，最终目的是 PetaLinux 从 SD 卡启动，并且开发板 USB HOST 接口可以操作 U 盘。

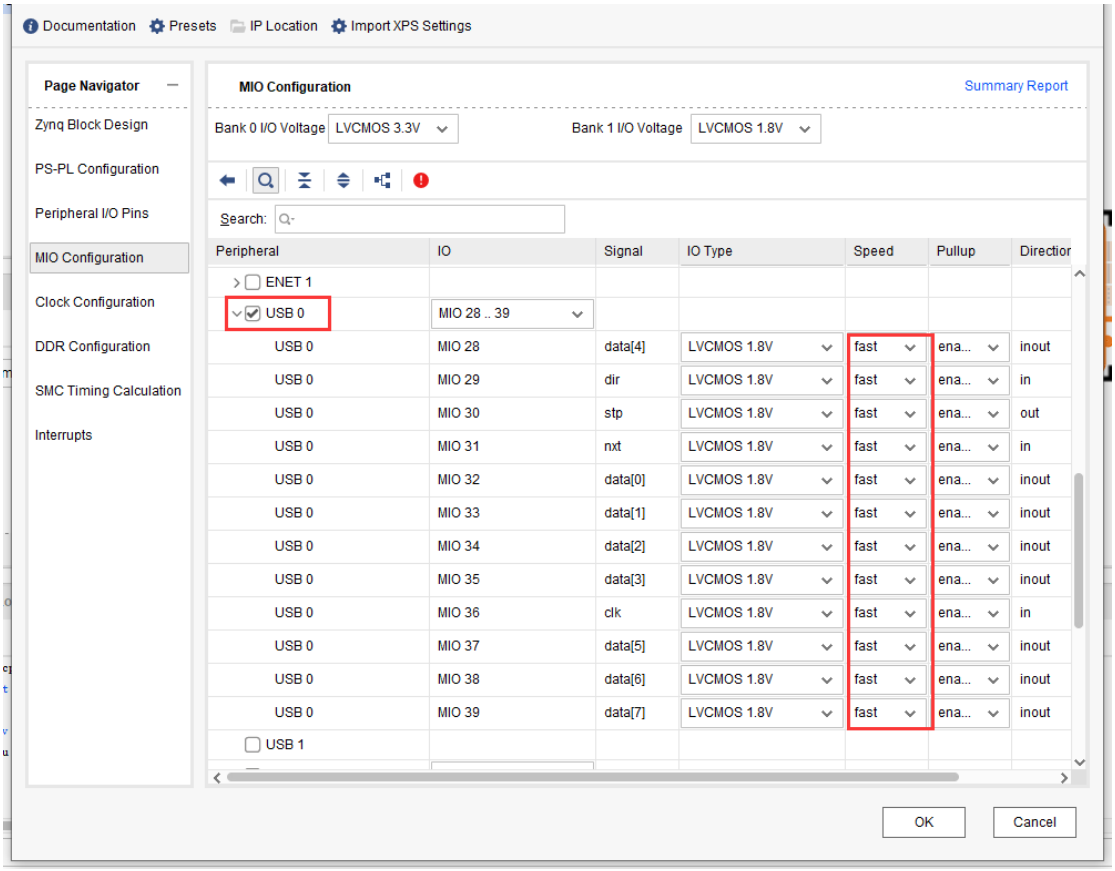
7.1 创建 Vivado 工程

打开实验 1 创建的 Vivado 工程，打开之后将其另存为一个新的工程并命名为“petalinux_usb”，之后删除工程目录下的.sdk 文件夹。打开设计文件，双击 ZYNQ7 Processing System IP 打开配置页面。点击 Perpheral I/O Pins 项，使能 USB0，对应 MIO28~MIO39,



其它闲置 MIO 全部作为 GPIO， 其中 MIO46 作为 USB PHY Reset 使用，





点击 MIO Configuration 项，将 USB 的信号 Speed 模式全部设为 fast，

点击 OK，配置完成。

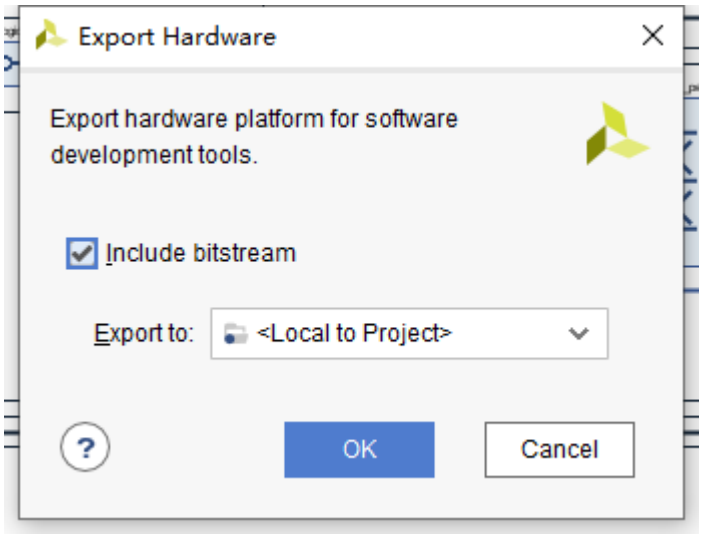
点击工具栏 Validate Design 验证设计，

验证成功， Ctrl + S 保存设计。

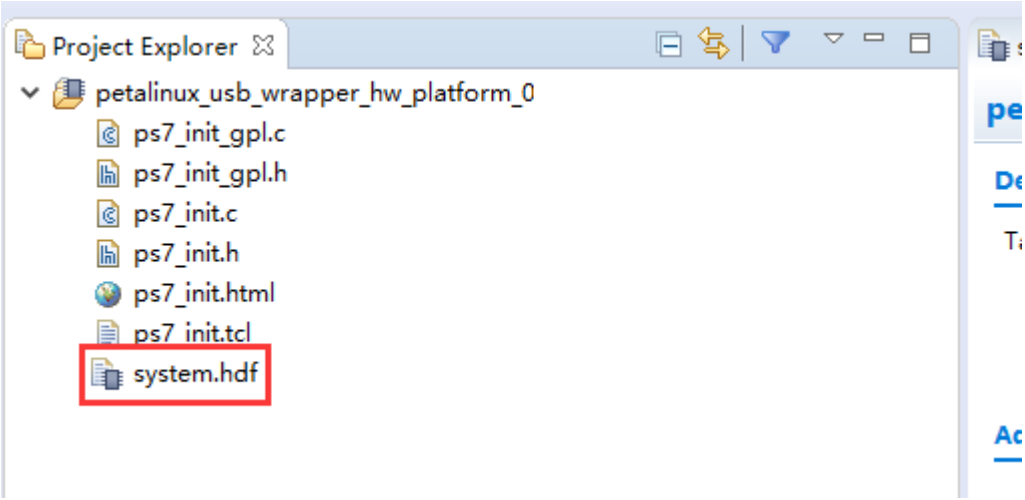
来到 Source 窗口，重新生成设计代码和顶层文件，

重新综合编译生成 Bitstream 文件。

Vivado 菜单栏 File > Export > Export Hardware，弹窗点击 OK，

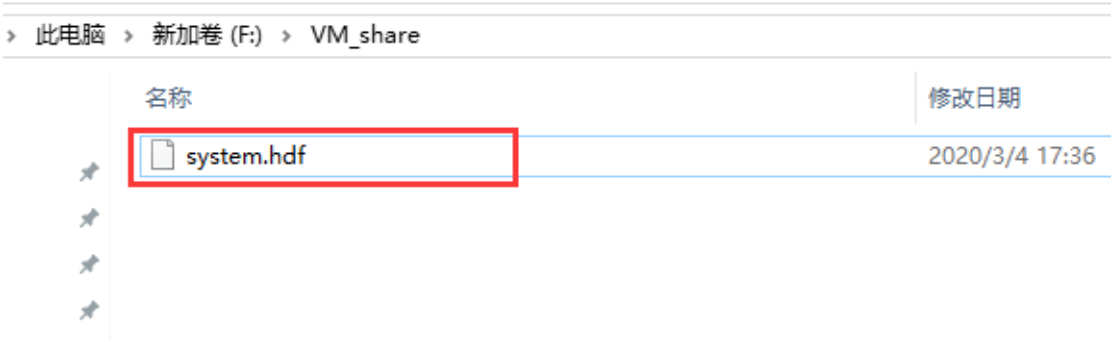


Vivado 菜单栏 File > Launch SDK，打开的 SDK 工程如下，



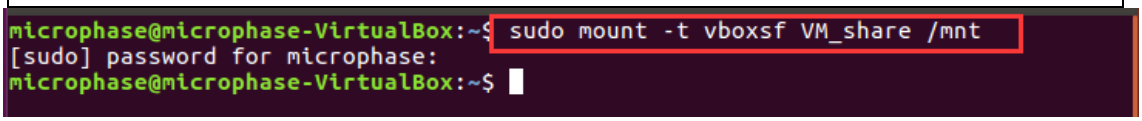
7.2 创建 Petalinux 工程

这样，在路径，D:\Z7-Lite\06_Linux_Demo\3_petalinux_usb\petalinux_usb\petalinux_usb.sdk\petalinux_usb_wrapper_hw_platform_0 下，就有了当前设计的硬件描述文件 system.hdf，将其拷贝到共享文件夹 F:\VM_share，PetaLinux 要用到这个硬件描述文件，



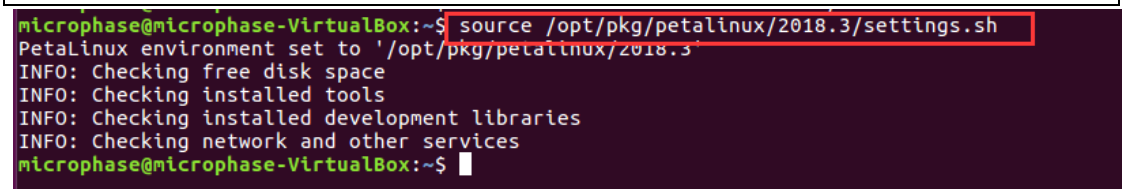
打开 VirtualBox 虚拟机进入 Ubuntu Linux 操作系统，打开 Terminal 窗口，输入下列命令挂载 VM_share 到/mnt，

```
sudo mount –t vboxsf VM_share /mnt
```



输入下列命令运行 Petalinux 设置文件，使 Petalinux 工具可用，

```
source /opt/pkg/petalinux/2018.3/settings.sh
```



新建 PetaLinux 工程，为了统一管理，进入上一节建立的文件夹 PetaLinux_proj，然后输入下列命令建立工程，如下所示，

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

```
microphase@microphase-VirtualBox:~/Petalinux_proj$ petalinux-create --type project --template zynq --name petalinux_usb
INFO: Create project: petalinux_usb
INFO: New project successfully created in /home/microphase/Petalinux_proj/petalinux_usb
```

进入该工程目录并将 system.hdf 拷贝到当前目录，

```
cd petalinux_usb/
cp /mnt/system.hdf .
```

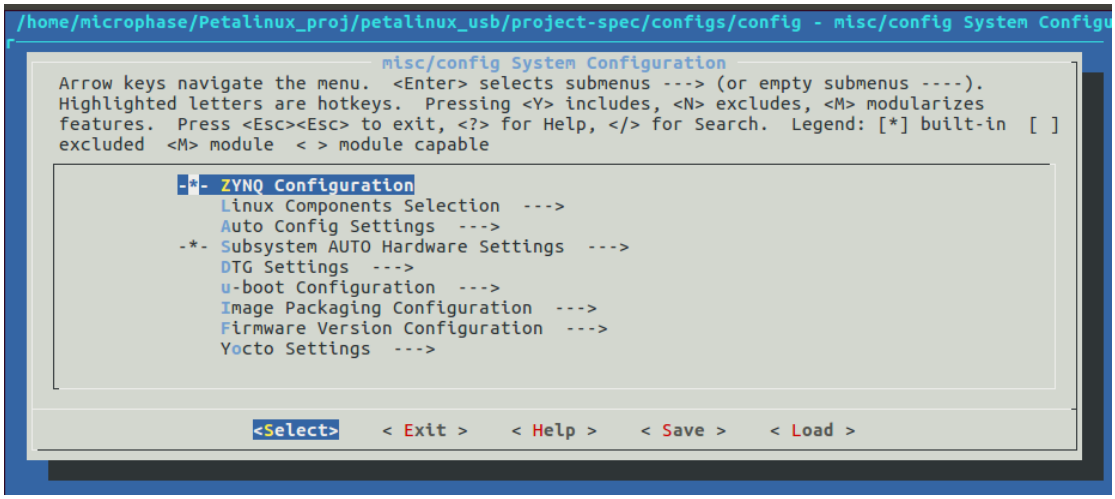
```
microphase@microphase-VirtualBox:~/Petalinux_proj$ cd petalinux_usb/
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_usb$ cp /mnt/system.hdf .
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_usb$
```

导入硬件描述文件 system.hdf，

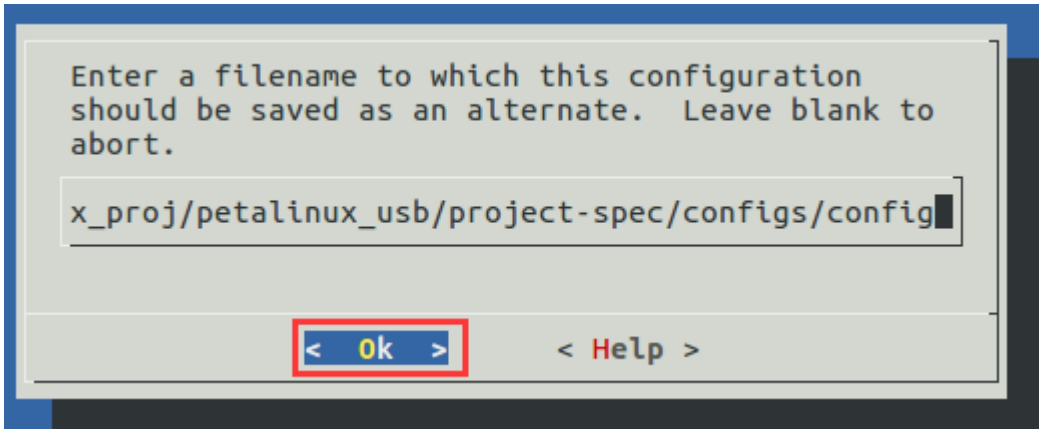
```
petalinux-config –get-hw-description=.
```

```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_usb$ petalinux-config --get-hw-description
INFO: Getting hardware description...
[INFO] generating Kconfig for project
[INFO] menuconfig project
```

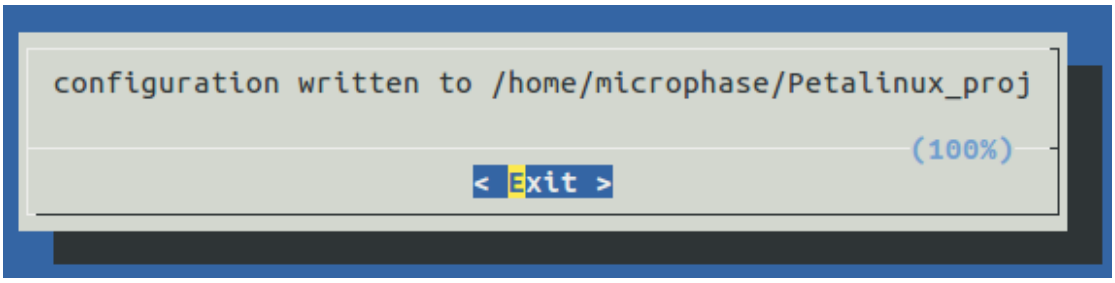
该命令会打开硬件设置界面，可以用方向键移动光标，按 Enter 进入某一条目查看详情，按两次 Esc 退回到上一层菜单。本例关于网口 Ethernet 的设置与之前实验相同，按照之前实验的操作步骤即可，之后直接选择 save，回车。



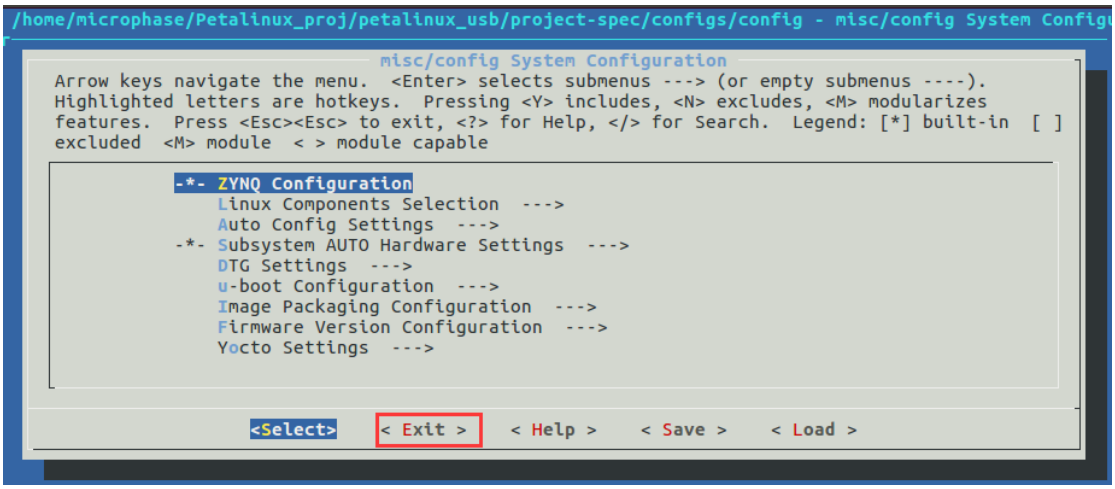
选择 OK，回车，



配置写入完成 Exit， 回车，



主窗口选择 Exit，回车，



7.3 配置 kernel

为了支持 USB 外设，这里要配置 Kernel，从 Xilinx 官方 Wiki http://www.wiki.xilinx.com/Zynq+Linux+USB+Device+Driver 可知，USB 设备有三种配置， 即 HOST 模式，Peripheral 模式和 OTG 模式，各模式的配置如下，

○ HOST 模式

```
Device Drivers
USB support
    <*> Support for Host-side USB
    <*> EHCI HCD (USB 2.0) support
    <*> USB Mass Storage support
    <*> ChipIdea Highspeed Dual Role Controller
    <*> ChipIdea host controller
        USB Physical Layer drivers --->
        <*> Generic ULPI Transceiver Driver
```

○ Peripheral Mode

```
Device Drivers
USB support
    <*> ChipIdea Highspeed Dual Role Controller
    <*> ChipIdea device controller
    <*> USB Gadget Support
          <M> USB Gadget Drivers
          <M> USB functions configurable through configfs
          [*] Mass storage
```

○ OTG

```
Device Drivers
USB support
    <*> Support for Host-side USB
    <*> OTG support
    <*> EHCI HCD (USB 2.0) support
    <*> USB Mass Storage support
    <*> ChipIdea Highspeed Dual Role Controller
    <*> ChipIdea host controller
    <*> ChipIdea device controller
        USB Physical Layer drivers --->
        <*> NOP USB Transceiver Driver
    <*> USB Gadget Support
          <M> USB Gadget Drivers
          <M> USB functions configurable through configfs
          [*] Mass storage
```

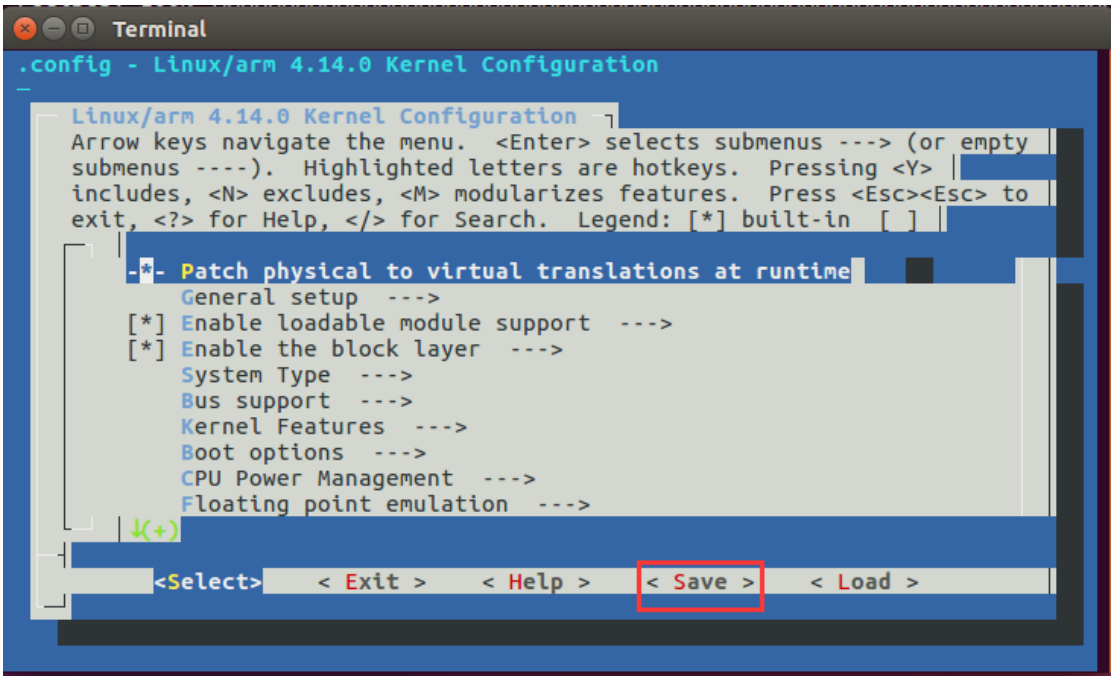
输入下列命令进入 kernal 配置界面，

```
petalinux-config –c kernal
```

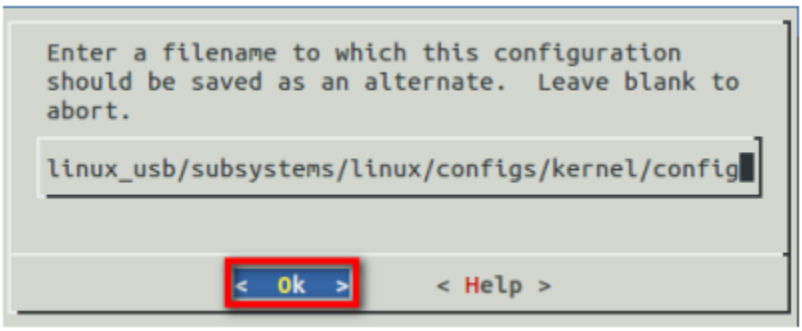
```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_usb$
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_usb$ petalinux-config -c kernel
[INFO] generating Kconfig for project
[INFO] sourcing bitbake
```

执行命令，将打开如下配置窗口，和前面遇到过的 PetaLinux HW 的配置窗口的操作是类似的，使用光标和 y，n，等选项。默认情况下，除了 OTG 模式下的 NOP USB Transceiver Driver 默认没有打开外（这个目前用不到），其它各种模式的配置选项都已经打开。读者可以自己查看一下各个条目，

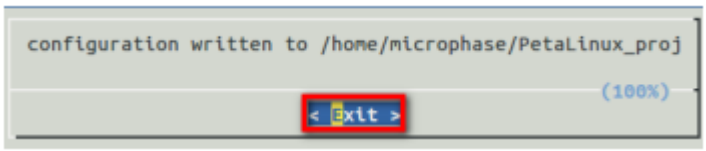
光标直接移到 Save，回车，



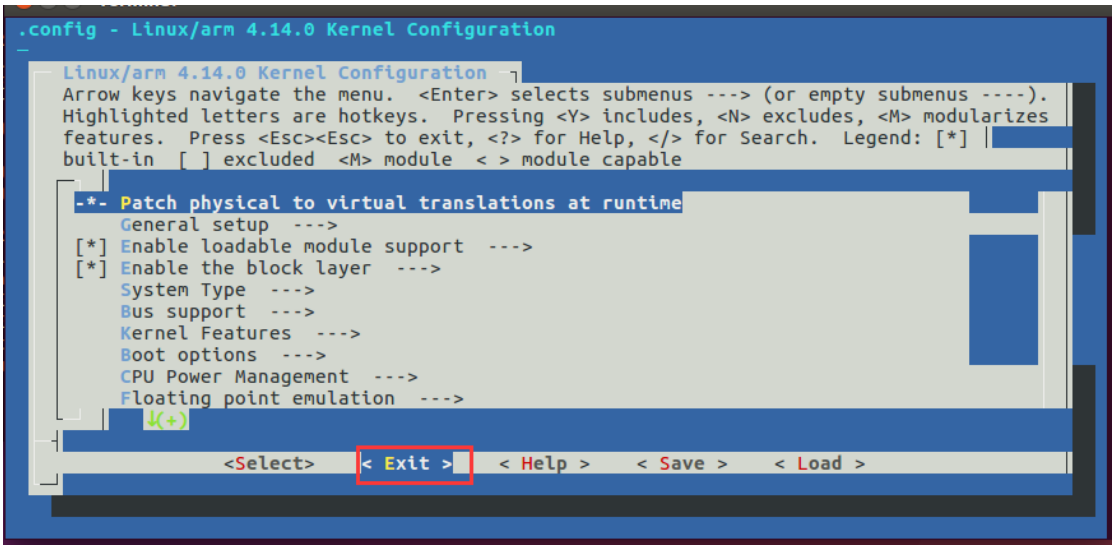
选择 OK，回车，



配置写入完成 Exit， 回车，



主窗口选择 Exit，回车，



kernel 配置完成。

7.4 修改设备树及编译工程

为支持 USB 设备的使用，我们需要在设备树文件中添加关于 USB 的相关内容，打开目录 home/Petalinux_proj/petalinux_usb/project-spec/meta-user/recipes-bsp/device-tree/files 下的 system-user.dtsi 文件，并在其中添加以下内容，添加完成后点击 SAVE 保存。

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

        mdio{
                #address-cells = <1>;
                #size-cells = <0>;
                phy1:phy@1{
                        compatible = "realtek,rtl8201","ethernet-phy-id001c.c816";
                        device_type = "ethernet-phy";
                        reg = <0>;

                };
        };
};
```

修改完成后回到 terminal 窗口，输入下列命令编译工程，

```
petalinux-build
```

```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_usb$ petalinux-build

[INFO] building project
[INFO] sourcing bitbake
INFO: bitbake petalinux-user-image
Loading cache: 100% |##############################################| Time: 0:00:01
```

编译需要较长时间，编译完成后会在当前工程目录下生成 images 文件夹，

```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_usb$ ll
total 392
drwxr-xr-x 8 microphase microphase   4096 3月   5 10:11 ./
drwxrwxr-x 8 microphase microphase   4096 3月   5 09:38 ../
drwxrwxr-x 8 microphase microphase   4096 3月   5 13:44 build/
drwxrwxr-x 3 microphase microphase   4096 3月   5 09:39 components/
-rw-r--r-- 1 microphase microphase    248 3月   5 09:38 config.project
-rw-rw-r-- 1 microphase microphase    186 3月   5 09:38 .gitignore
drwxrwxr-x 3 microphase microphase   4096 3月   5 10:11 images/
drwxrwxr-x 2 microphase microphase   4096 3月   5 13:44 .petalinux/
drwxr-xr-x 6 microphase microphase   4096 3月   5 09:40 project-spec/
-rwxrwxr-x 1 microphase microphase 359255 3月   5 09:38 system.hdf*
drwxrwxr-x 3 microphase microphase   4096 3月   5 09:44 .Xil/
```

7.5 BOOT.bin 文件打包与烧录验证

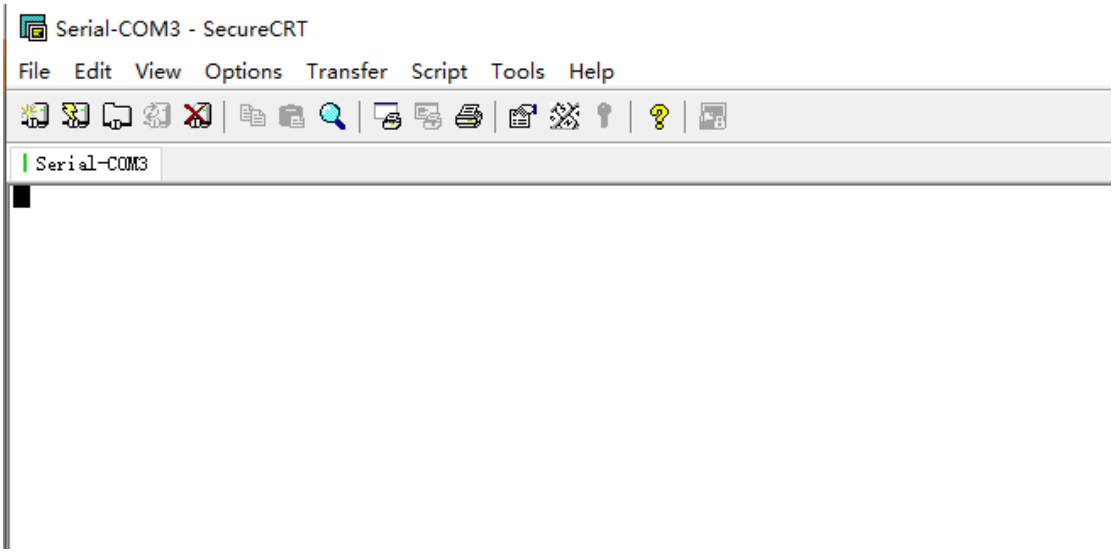
运行下列命令生成启动镜像文件

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

将 iamges/linux 文件夹内的 BOOT.bin 和 image.ub 文件复制到/mnt 文件夹内，这样就可以在 windows 下使用这两个文件，将其复制到 SD 卡中，插入开发板 SD

卡槽中，连接串口 type-c 数据线，连接 JTAG 接口数据线给开发版供电

打开 SecureCRT，连接串口，



上电启动，输入用户名：root，密码：root 登录系统，插入 U 盘（注意，默认情况下只支持 FAT32 格式化的 U 盘）， 可以看到系统成功检测到了 U 盘插入，并打印出相关信息，

```
Starting syslogd/klogd: done
Starting tcf-agent: OK

PetaLinux 2018.3 petalinux_usb /dev/ttyPS0

petalinux_usb login: root
Password:
root@petalinux_usb:~# usb 1-1: new high-speed USB device number 2 using ci_hdrc
usb-storage 1-1:1.0: USB Mass Storage device detected
scsi host0: usb-storage 1-1:1.0
scsi 0:0:0:0: Direct-Access     Mass     Storage Device   1.00 PQ: 0 ANSI: 0 CCS
sd 0:0:0:0: Attached scsi generic sg0 type 0
sd 0:0:0:0: [sda] 30547968 512-byte logical blocks: (15.6 GB/14.6 GiB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] No Caching mode page found
sd 0:0:0:0: [sda] Assuming drive cache: write through
 sda: sda1
sd 0:0:0:0: [sda] Attached SCSI removable disk
FAT-fs (sda1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
```

输入 fdisk –l 可以查看 U 盘信息，可见 U 盘被识别为/dev/sda。

```
root@petalinux_usb:~# fdisk -l

Disk /dev/mmcblk0: 7935 MB, 7935623168 bytes
255 heads, 63 sectors/track, 964 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

        Device Boot      Start         End      Blocks  Id System
/dev/mmcblk0p1               1         965     7748608   c Win95 FAT32 (LBA)
Partition 1 has different physical/logical endings:
     phys=(963, 254, 63) logical=(964, 200, 4)

Disk /dev/sda: 15.6 GB, 15640559616 bytes
255 heads, 63 sectors/track, 1901 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks  Id System
/dev/sda1               1        1902    15273952+  c Win95 FAT32 (LBA)
root@petalinux_usb:~#
```

按照如下操作可以将 U 盘挂载到文件夹下并查看 U 盘中的文件。①创建 my_usb 文件夹； ②挂载 sda1 到文件夹； ③显示文件夹信息，可以看到在 U 盘中的 Test.txt 文件，可见挂载成功（注意这里不支持中文显示）； ④解挂； ⑤重新查看文件夹信息，不显示 U 盘内的文件，可见解挂成功。

拔掉 U 盘， 系统可以检测到 U 盘被拔出，

```
root@petalinux_usb:~#
root@petalinux_usb:~# usb 1-1: USB disconnect, device number 2
^C
root@petalinux_usb:~#
```

```
root@petalinux_usb:~#
root@petalinux_usb:~# mkdir -p /mnt/my_usb 1
root@petalinux_usb:~#
root@petalinux_usb:~# mount /dev/sda1 /mnt/my_usb/ 2
root@petalinux_usb:~#
root@petalinux_usb:~# ls /mnt/my_usb/ 3
System Volume Information  Test.txt
root@petalinux_usb:~#
root@petalinux_usb:~#
root@petalinux_usb:~# umount /dev/sda1 4
root@petalinux_usb:~#
root@petalinux_usb:~# ls /mnt/my_usb/ 5
root@petalinux_usb:~#
```

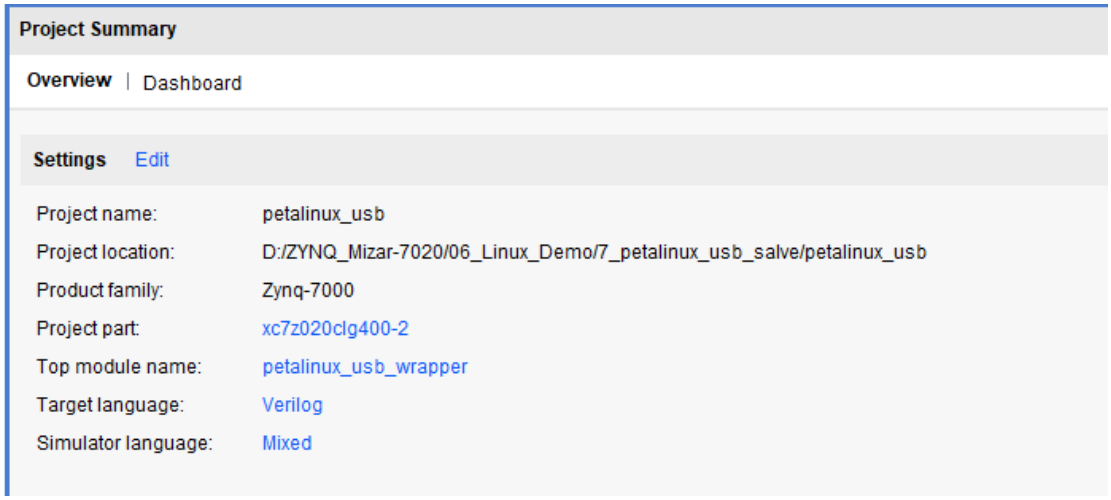
到此实验完成。

第8章　USB 外设 slave 模式应用实验

上一章的实验中我们讲到 USB 外设的使用，主要是 USB 的 HOST 模式，即开发板 USB 接口作为主机读写插入 U 盘、键盘等设备的数据。在本章中我们来学习 USB 的另一模式“peripheral 模式”（外围设备模式），也叫“slave 模式”（从机模式），即开发板 usb 作为外围设备供电脑等主机操作。这种模式主要有两方面应用，一是 Mass Storage，即开发板作为 U 盘，另一种是 Ethernet，即将 USB 网口模式，通过 USB 接口连接网络。接下来就讲解具体操作方法。

8.1 创建 Vivado 工程

本实验的硬件设计与上一实验完全相同，可直接使用之前实验生成的 HDF 文件及 petalinux 工程。本章为保证实验的完整性，将之前实验的工程复制到新的工程文件夹 7_petalinux_usb_slave，如下图所示：



之后在共享文件夹下创建 usb 文件夹，用来保存本实验相关文件。将该工程的 system.hdf 文件复制到该文件夹内。

8.2 创建 Petalinux 工程

打开 VirtualBox 虚拟机进入 Ubuntu Linux 操作系统，打开 Terminal 窗口，输入下列命令挂载 VM_share 到/mnt，运行 petalinux 配置文件使 Petalinux 工具可用

```
sudo mount –t vboxsf VM_share /mnt
source /opt/pkg/petalinux/2018.3/settings.sh
```

```
microphase@microphase-VirtualBox:~$ sudo mount -t vboxsf VM_share /mnt/
[sudo] password for microphase:
microphase@microphase-VirtualBox:~$ source /opt/pkg/petalinux/2018.3/settings.sh

PetaLinux environment set to '/opt/pkg/petalinux/2018.3'
INFO: Checking free disk space
INFO: Checking installed tools
INFO: Checking installed development libraries
INFO: Checking network and other services
```

进入 Petalinux_proj 文件夹，输入下列命令创建 petalinux 工程，

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

```
microphase@microphase-VirtualBox:~/Petalinux_proj$ petalinux-create --type proje
ct --template zynq --name petalinux_usb_salve
INFO: Create project: petalinux_usb_salve
INFO: New project successfully created in /home/microphase/Petalinux_proj/petali
nux_usb_salve
```

进入创建完成的工程文件夹，将硬件描述文件 system.hdf 复制工程文件夹下，

```
cd petalinux_usb_slave
cp /mnt/usb/system.hdf .
```

```
microphase@microphase-VirtualBox:~/Petalinux_proj$ cd petalinux_usb_salve/
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_usb_salve$ cp /mnt/u
sb/system.hdf .
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_usb_salve$ ls
config.project  project-spec  system.hdf
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_usb_salve$
```

输入下列命令将硬件描述文件导入工程中，关于 Ethernet 网口的设置与实验 1 相同

```
petalinux-config    --get-hw-description=.
```

```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_usb_salve$ petalinux
-config --get-hw-description=.
INFO: Getting hardware description...
[INFO] generating Kconfig for project
[INFO] menuconfig project


*** End of the configuration.
```

8.3 配置 kernel

在 Xilinx 官方 WIKI 网站 http://www.wiki.xilinx.com/Zynq+Linux+USB+Device+Driver 下有 USB 的使用指导，读者可以在此处查阅更详细的信息。从此处可知 Peripheral Mode 下两种应用需要进行配置的地方如下图所示：

Mass Storage

```
Device Drivers
USB support
    <*> ChipIdea Highspeed Dual Role Controller
    <*> ChipIdea device controller
    <*> USB Gadget Support
          <M> USB Gadget Drivers
          <M> USB functions configurable through configfs
          [*] Mass storage
```

Ethernet

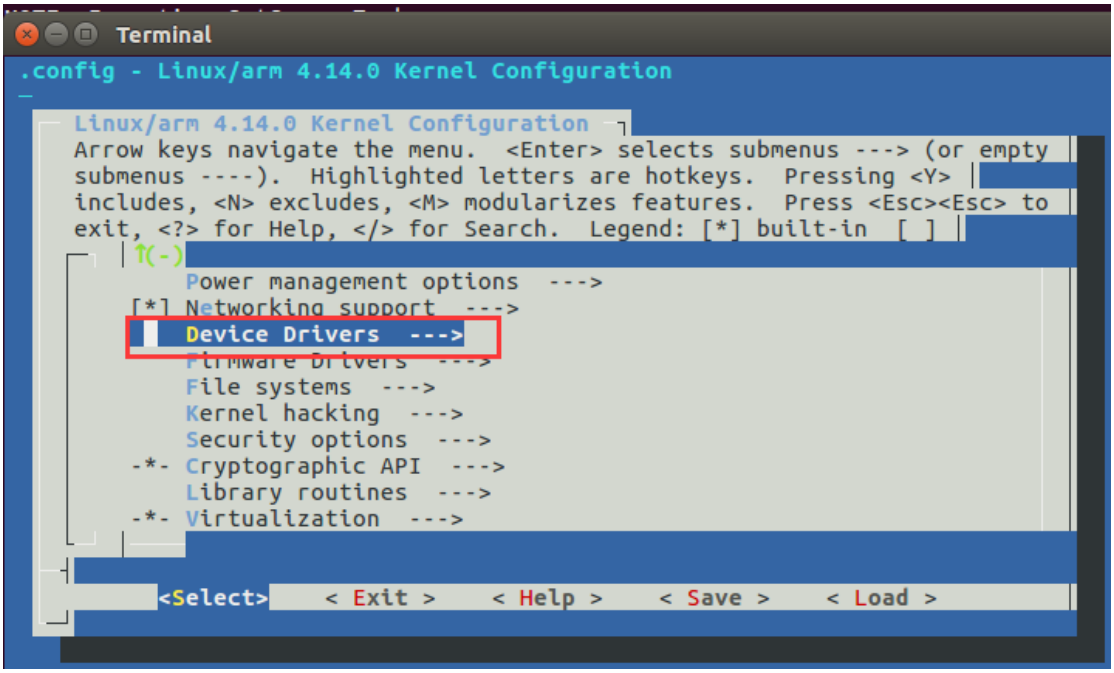
```
<M>  USB Gadget Support
     <M>  USB Gadget Drivers
     <M>  USB functions configurable through configfs
     [*]  RNDIS
```

输入下列命令打开 kernel 配置菜单，

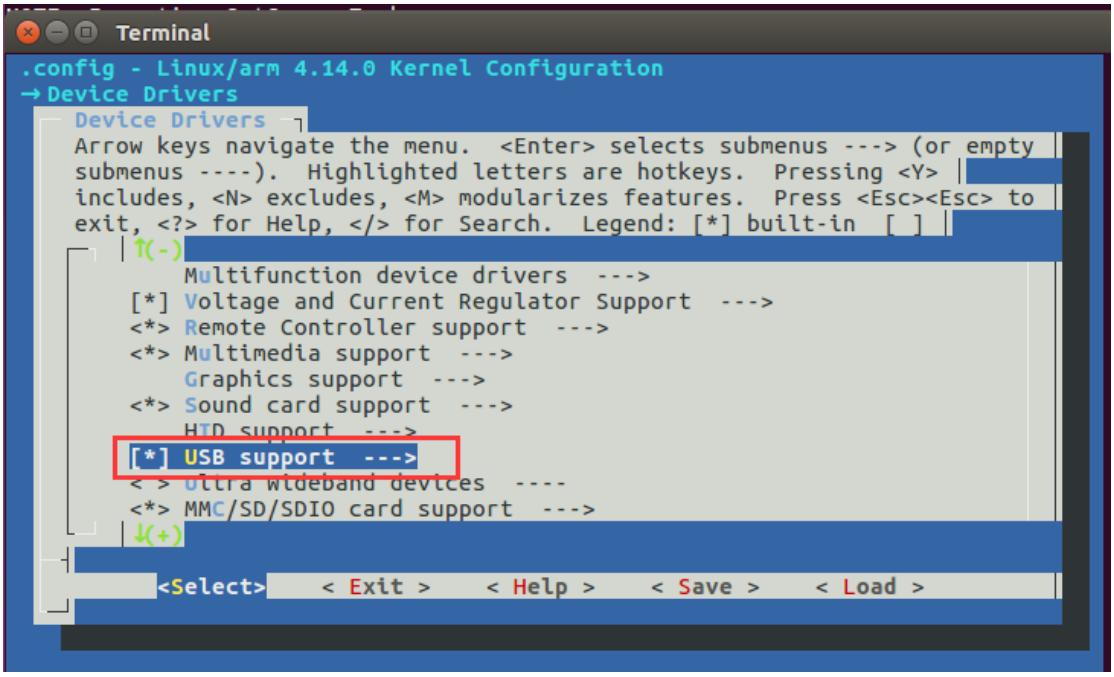
```
petalinux-config   -c   kernel
```

```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_usb_salve$ petalinux
-config -c kernel
[INFO] generating Kconfig for project
[INFO] sourcing bitbake
[INFO] generating plnxtool conf
[INFO] generating meta-plnx-generated layer
[INFO] generating machine configuration
[INFO] configuring: kernel
[INFO] generating kernel configuration files
[INFO] bitbake virtual/kernel -c menuconfig
```

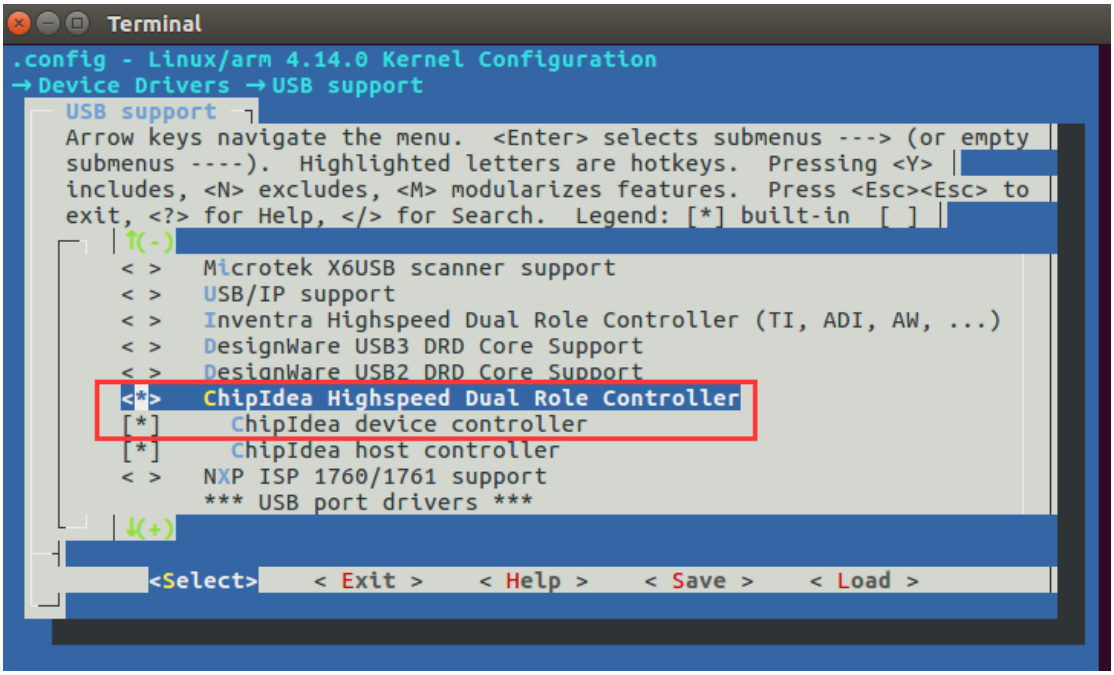
按照下图所示进入 Device Drivers 子菜单，



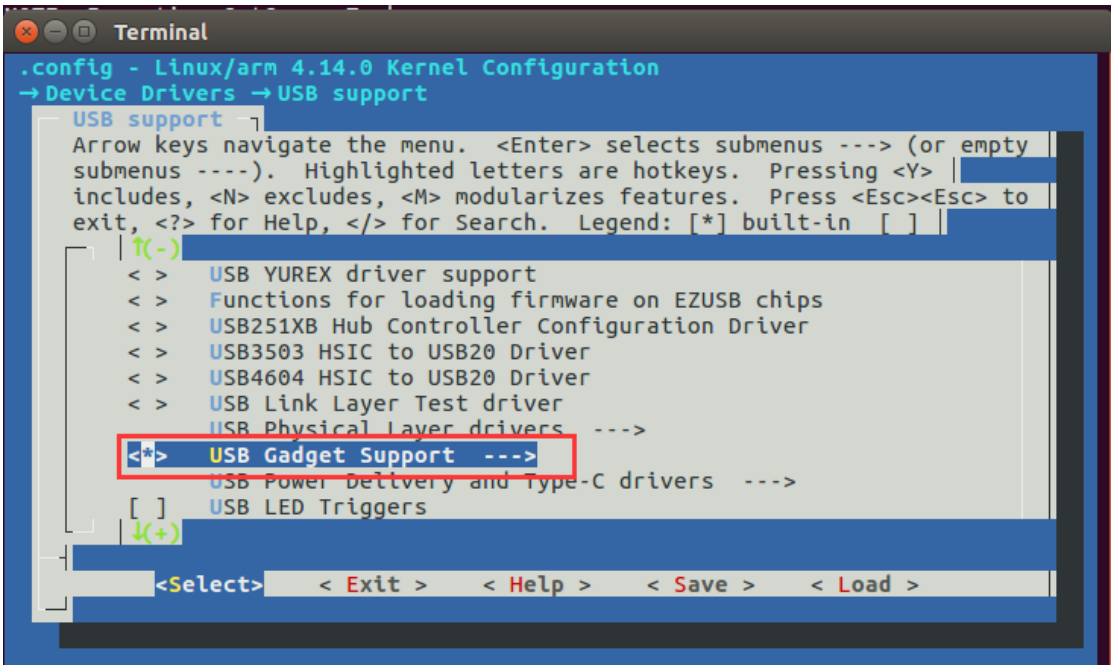
进入 USB support 设置菜单，



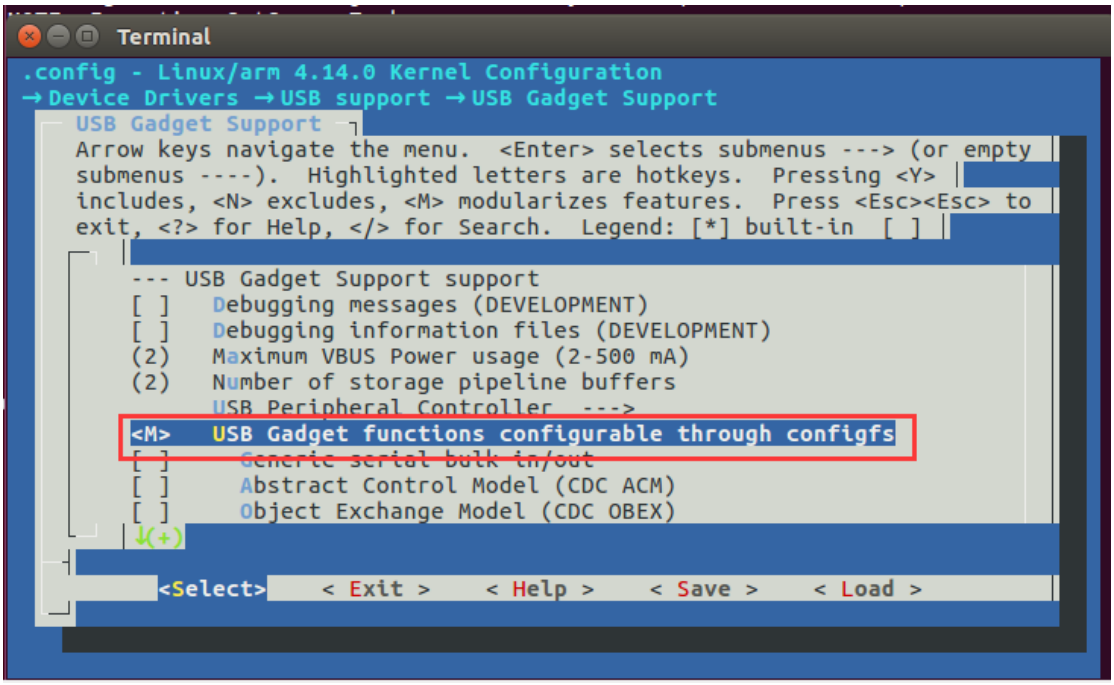
设置将 ChipIdea Highspeed Dual Role Controller 和 ChipIdea device controller 编译入内核，如下图所示：



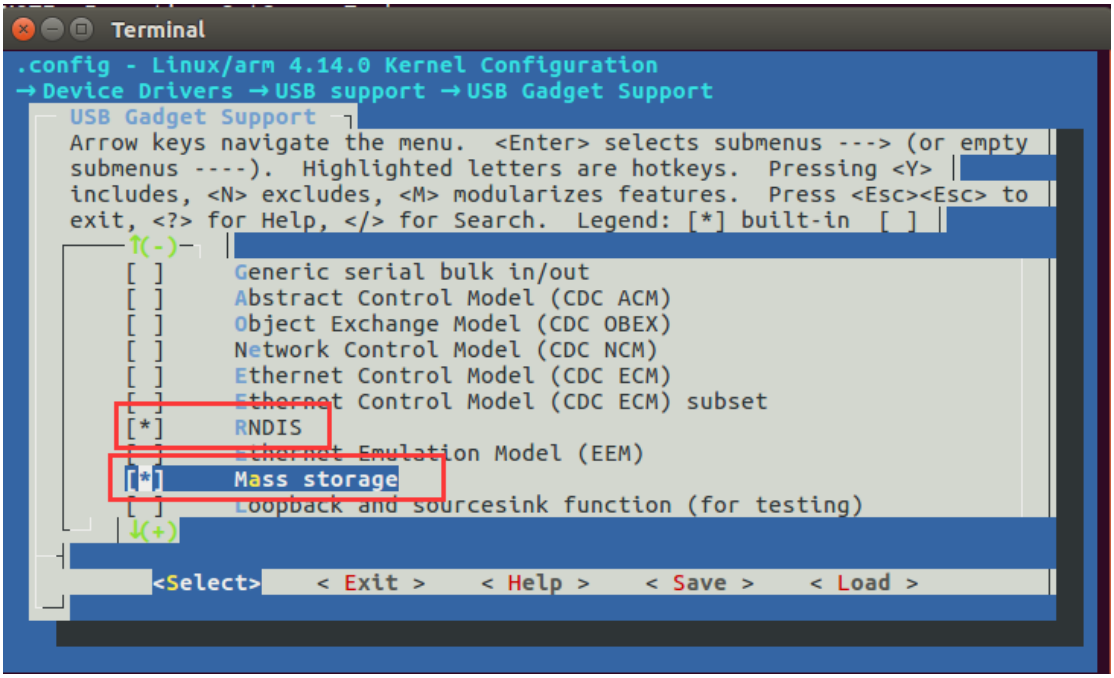
进入 USB Gadget Support 设置菜单，



设置 USB Gadget functions configurable through configfs 为 M，即编译生成驱动文件而不编译进内核当中，如下图所示：



使能 RNDIS 和 Mass storage 编译如内核中，如下图所示：



8.4 配置设备树

为支持 USB 设备的使用，我们需要在设备树文件中添加关于 USB 的相关内容，打开目录 home/Petalinux_proj/petalinux_usb_slave/project-spec/meta-user/recipes-bsp/device-tree/files 下的 system-user.dtsi 文件，并在其中添加以下内容，添加完成后点击 SAVE 保存。

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


/ {
    model = "Mizar Development Board";
    compatible = "microphase,zynq-7020","xlnx,zynq-7000";

        usb_phy0:usb_phy@0{
                compatible = "ulpi-phy";
                #phy-cells = <0>;
                reg = <0xe0002000 0x1000>;
                view-port = <0x170>;
                drv-vbus;
        };

};

&usb0{
        dr_mode = "peripheral";
        usb-phy = <&usb_phy0>;
};




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

        mdio{
                #address-cells = <1>;
                #size-cells = <0>;
                phy1:phy@1{
                        compatible = "realtek,rtl8201","ethernet-phy-id001c.c816";
                        device_type = "ethernet-phy";
                        reg = <0>;

                };
        };
};
```

设备树修改完成之后即可编译工程生成可执行文件和系统镜像。

8.5 编译工程及打包启动文件

输入下列命令编译工程，

```
petalinux-build
```

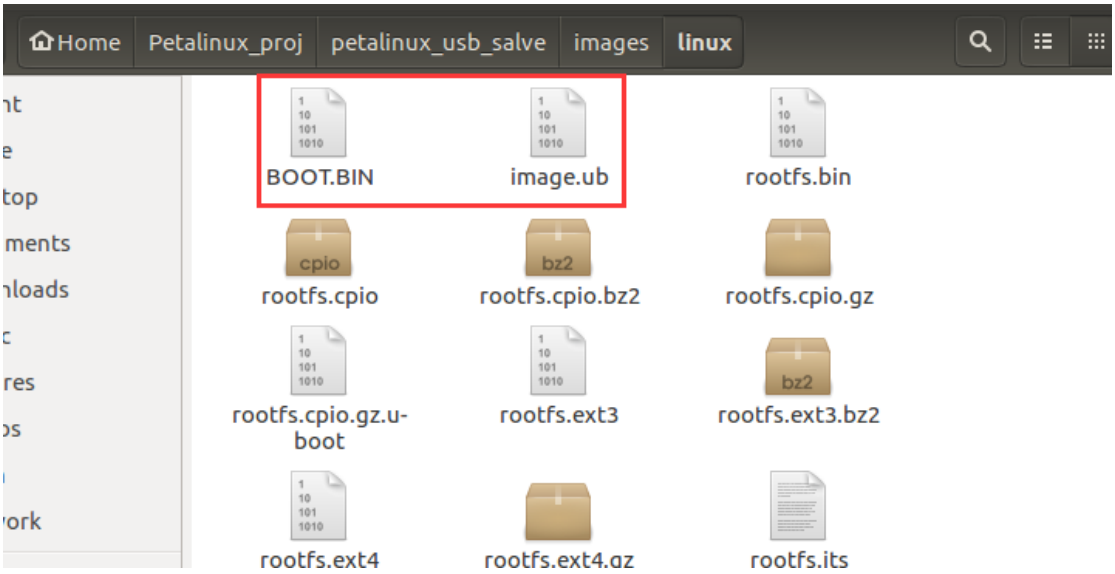
```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_usb_salve$ petalinux
-build
[INFO] building project
[INFO] sourcing bitbake
INFO: bitbake petalinux-user-image
Loading cache: 100% |##############################################| Time: 0:00:00
Loaded 3444 entries from dependency cache.
Parsing recipes: 100% |#############################################| Time: 0:00:10
Parsing of 2569 .bb files complete (2536 cached, 33 parsed). 3445 targets, 149 s
kipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies
Initialising tasks: 100% |##########################################| Time: 0:00:14
Checking sstate mirror object availability: 100% |###############| Time: 0:01:00
NOTE: Executing SetScene Tasks
NOTE: Executing RunQueue Tasks
WARNING: petalinux-user-image-1.0-r0 do_rootfs: [log_check] petalinux-user-image
: found 1 warning message in the logfile:
[log_check] warning: %post(sysvinit-inittab-2.88dsf-r10.plnx_zynq7) scriptlet fa
iled, exit status 1

NOTE: Tasks Summary: Attempted 3065 tasks of which 2246 didn't need to be rerun
and all succeeded.
```

等待一段时间编译完成后输入下列命令打包生成系统启动文件，

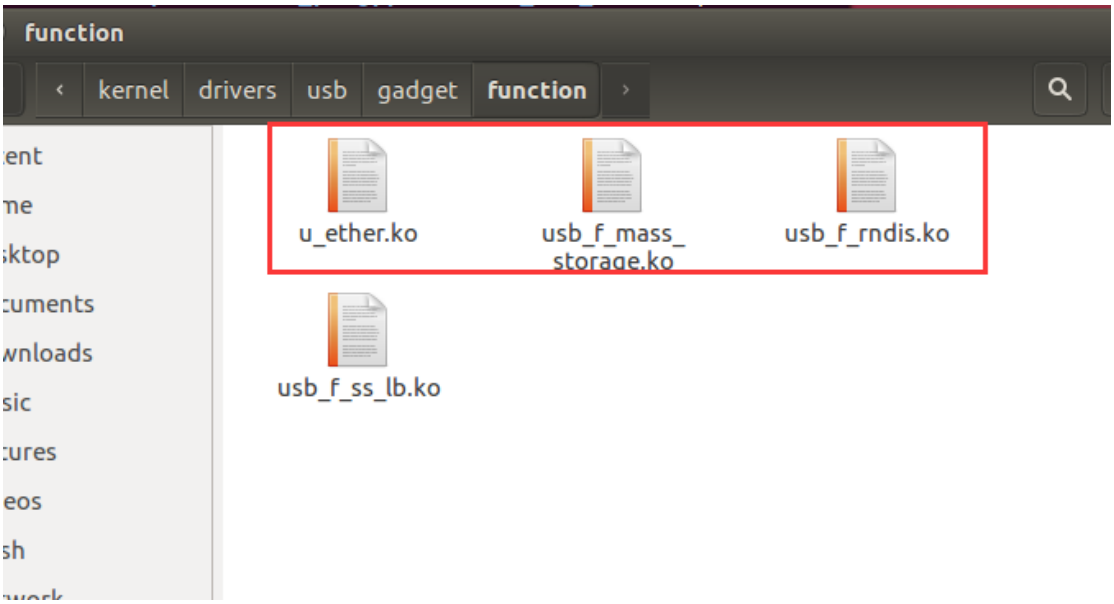
```
petalinux-package --boot --fsbl ./images/linux/zynq_fsbl.elf --u-boot --fpga
--force
```

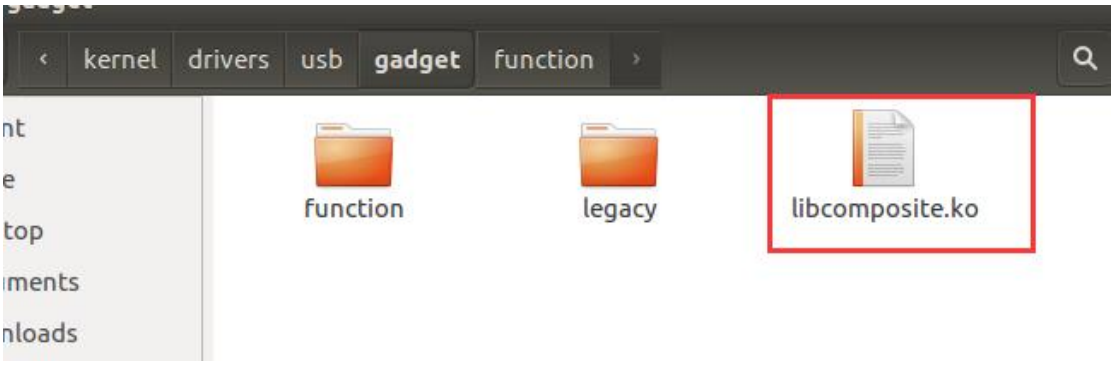
```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_usb_salve$ petalinux
-package --boot --fsbl ./images/linux/zynq_fsbl.elf  --u-boot --force
INFO: File in BOOT BIN: "/home/microphase/Petalinux_proj/petalinux_usb_salve/ima
ges/linux/zynq_fsbl.elf"
INFO: File in BOOT BIN: "/home/microphase/Petalinux_proj/petalinux_usb_salve/ima
ges/linux/u-boot.elf"
INFO: Generating Zynq binary package BOOT.BIN...


****** Xilinx Bootgen v2018.3
  **** Build date : Nov 15 2018-19:22:29
    ** Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.

INFO: Binary is ready.
WARNING: Unable to access the TFTPBOOT folder /tftpboot!!!
WARNING: Skip file copy to TFTPBOOT folder!!!
```

打包生成的文件如下图所示：

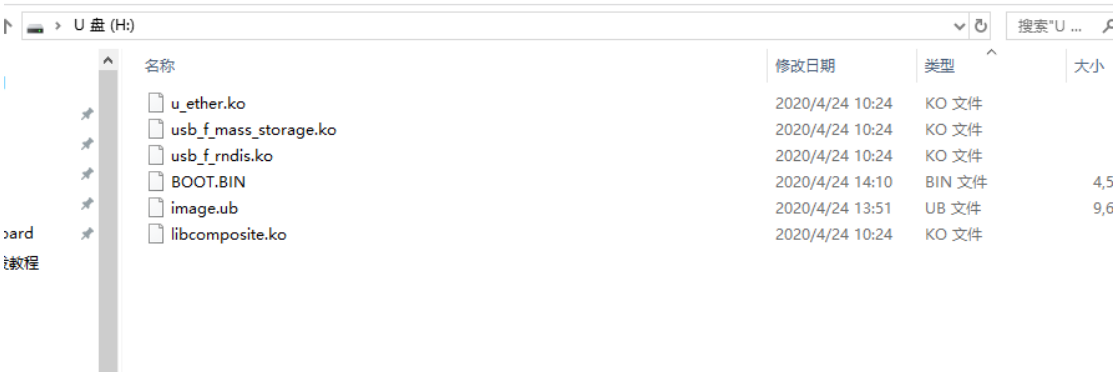


除系统镜像及启动文件外，配置 USB slave 模式还需要以下几个驱动文件，查找以下几个文件，和打包生成的驱动文件一同复制到共享文件夹中，之后再复制到 SD 卡中即可，





复制完成之后即可启动系统验证 usb salve 模式能否正常运行。



8.6 系统启动验证

将复制好相关文件的 SD 卡插入开发板上的卡槽，调整跳线帽设置开发板启动方式为 SD 卡启动。连接串口 type-c 线，打开 SecureCRT 软件连接串口，连接 JTAG 接口上电启动 Linux 操作系统。

输入用户名：root 和密码：root 登录系统，之后按照以下命令列表进行操作。挂载 SD 卡到系统中，将驱动加载到系统中，设置相关参数。

Steps for mounting the sdcard for accessing the compiled modules
zynq> mount /dev/mmcblk0p1 /mnt

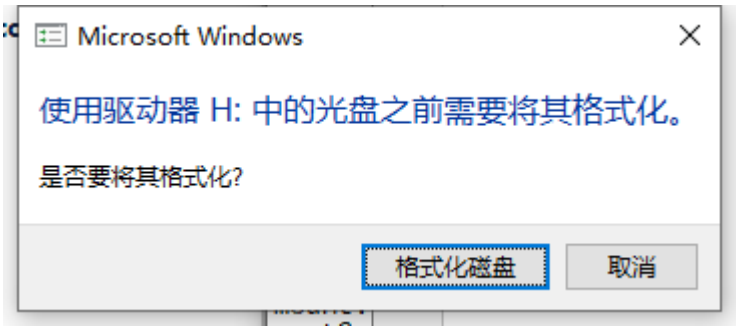
Install the following modules

```
zynq> insmod /mnt/libcomposite.ko
zynq> insmod /mnt/usb_f_mass_storage.ko

zynq> dd if=/dev/zero of=/tmp/mydev count=10 bs=1M
zynq> mount -t configfs none /sys/kernel/config
zynq> cd /sys/kernel/config/usb_gadget
zynq> mkdir g1
zynq> cd g1
zynq> echo "64" > bMaxPacketSize0
zynq> echo "0x200" > bcdUSB
zynq> echo "0x100" > bcdDevice
zynq> echo "0x03FD" > idVendor
zynq> echo "0x0500" > idProduct
zynq> mkdir functions/mass_storage.ms0
Number of LUNs=8
Mass Storage Function, version: 2009/09/11
LUN: removable file: (no medium)
zynq> mkdir configs/c1.1
zynq> echo /tmp/mydev > functions/mass_storage.ms0/lun.0/file
zynq> echo 1 > functions/mass_storage.ms0/lun.0/removable
zynq> ln -s functions/mass_storage.ms0 configs/c1.1/
zynq> echo "ci_hdrc.0" > UDC
configfs-gadget gadget: high-speed config #1: c1
```

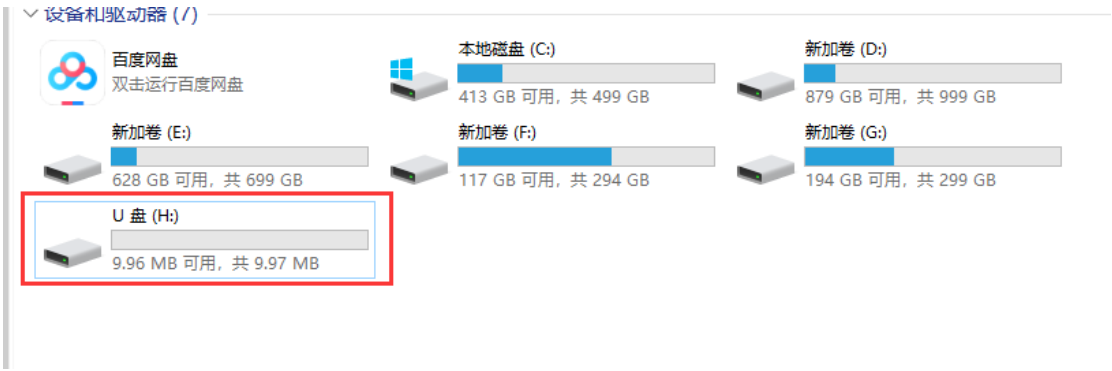
```
root@petalinux_usb:~# mount /dev/mmcblk0p1 /mnt/
root@petalinux_usb:~# insmod /mnt/libcomposite.ko
root@petalinux_usb:~# insmod /mnt/usb_f_mass_storage.ko
root@petalinux_usb:~# dd if=/dev/zero of=/tmp/mydev count=10 bs=1M
10+0 records in
10+0 records out
root@petalinux_usb:~# mount -t configfs none /sys/kernel/config/
mount: mounting none on /sys/kernel/config/ failed: Device or resource busy
root@petalinux_usb:~# cd /sys/kernel/config/usb_gadget/
root@petalinux_usb:/sys/kernel/config/usb_gadget# mkdir g1
root@petalinux_usb:/sys/kernel/config/usb_gadget# cd g1/
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# echo "64" > bMaxPacketSize0
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# echo "0x200" > bcdUSB
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# echo "0x100" > bcdDevice
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# echo "0x03fd" > idVendor
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# echo "0x0500" > idProduct
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# mkdir functions/mass_storage.ms0
Mass Storage Function, version: 2009/09/11
LUN: removable file: (no medium)
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# mkdir configs/c1.1
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# echo /tmp/mydev > functions/mass_storage.ms0/lun.0/file
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# echo 1 > functions/mass_storage.ms0/lun.0/removable
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# ln-s functions/mass_storage.ms0/ configs/c1.1/
-sh: ln-s: command not found
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# ln -s functions/mass_storage.ms0/ configs/c1.1/
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# echo "ci_hdrc.0" > UDC
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# configfs-gadget gadget: high-speed config #1: c1
random: crng init done
```

设置完成之后使用公对公 USB 线连接开发板 USB SLAVE 接口和电脑 USB 接口

之后电脑会弹出如下提示，说明电脑已检测到设备插入，



格式化完成之后即可看到电脑上出现一个 10M 大小的 U 盘，说明 USB 的 Slave 模式下的 Mass storage 功能运行正常，可以向其中写入数据。



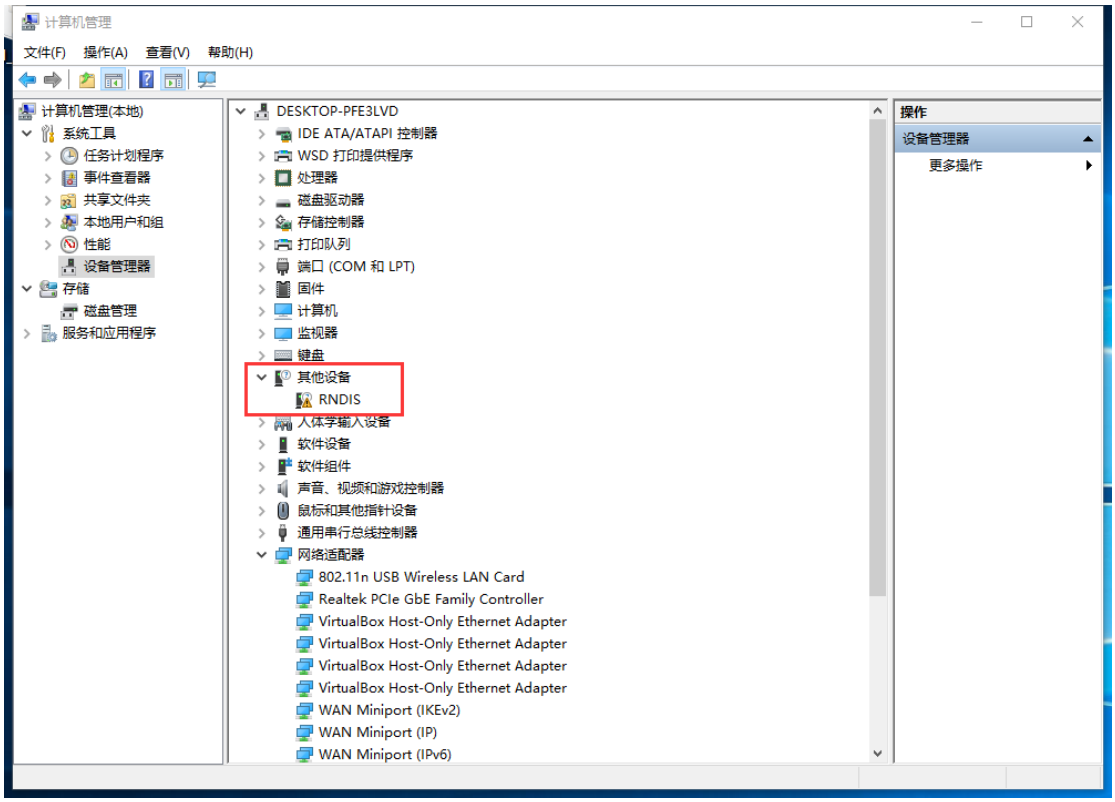
测试运行 Ethernet 功能，按照下列命令列表进行操作，挂载 SD 到系统中，加载相关驱动文件，设置运行相关参数，最后设置 usb 网口 IP 地址为 10.10.70.1，并启动网口

```
Steps for mounting the sdcard for accessing the compiled modules
zynq> mount /dev/mmcblk0p1 /mnt

Install the following modules

zynq> insmod /mnt/libcomposite.ko
zynq> insmod /mnt/u_ether.ko
zynq> insmod /mnt/usb_f_rndis.ko

zynq> mount -t configfs none /sys/kernel/config
zynq> cd /sys/kernel/config/usb_gadget
zynq> mkdir g1
zynq> cd g1
zynq> echo "64" > bMaxPacketSize0
zynq> echo "0x200" > bcdUSB
zynq> echo "0x100" > bcdDevice
zynq> echo "0x03FD" > idVendor
zynq> echo "0x0500" > idProduct
zynq> mkdir functions/rndis.rn0
zynq> mkdir configs/c1.1
zynq> ln -s functions/rndis.rn0 configs/c1.1/
zynq> echo "ci_hdrc.0" > UDC
zynq> ifconfig usb0 10.10.70.1
zynq> ifconfig usb0 up

Run the standard network tests like ping, iperf,netperf...
```

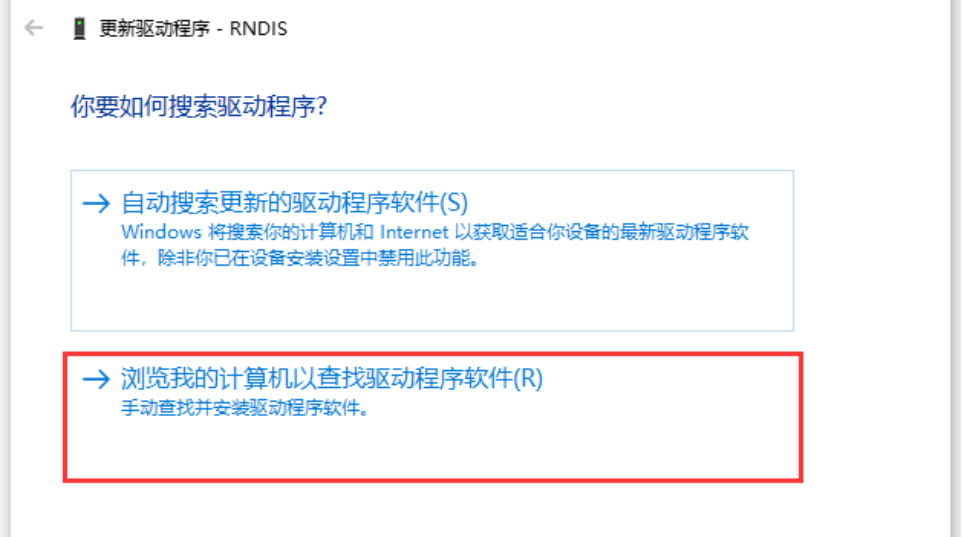
下面是具体执行过程，使用 Micro USB 线连接开发板 USB SLAVE 接口和电脑 USB 接口，可以看到提示 usb0: link becomes ready，说明开发板 USB 已运行在 USB Ethernet 模式下，并连接到主机 USB 接口。

```
Password:
root@petalinux_usb:~# mount /dev/mmcblk0p1 /mnt/
root@petalinux_usb:~# insmod /mnt/libcomposite.ko
root@petalinux_usb:~# insmod /mnt/u_ether.ko
root@petalinux_usb:~# insmod /mnt/usb_f_rndis.ko
root@petalinux_usb:~# mount -t configfs none /sys/kernel/config/
mount: mounting none on /sys/kernel/config/ failed: Device or resource busy
root@petalinux_usb:~# cd /sys/kernel/config/usb_gadget/
root@petalinux_usb:/sys/kernel/config/usb_gadget# mkdir g1
root@petalinux_usb:/sys/kernel/config/usb_gadget# cd g1/
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# echo "64" > bMaxPacketSize0
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# ECHO "0X200" > bcdUSB
-sh: ECHO: command not found
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# echo "0x200" > bcdUSB
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# echo "0x100" > bcdDevice
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# echo "0x03fd" > idVendor
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# echo "0x0500" > idProduct
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# mkdir functions/rndis.rn0
using random self ethernet address
using random host ethernet address
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# mkdir configs/c1.1
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# ln -s functions/rndis.rn0/ configs/c1.1/
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# echo "ci_hdrc.0" > UDC
usb0: HOST MAC 0e:fd:7d:03:6c:b9
usb0: MAC f6:e3:5c:d6:7f:fa
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# ifconfig usb0 10.10.70.1
IPv6: ADDRCONF(NETDEV_UP): usb0: link is not ready
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# ifconfig usb0 up
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# configfs-gadget gadget: high-speed config #1: c1
IPv6: ADDRCONF(NETDEV_CHANGE): usb0: link becomes ready
```

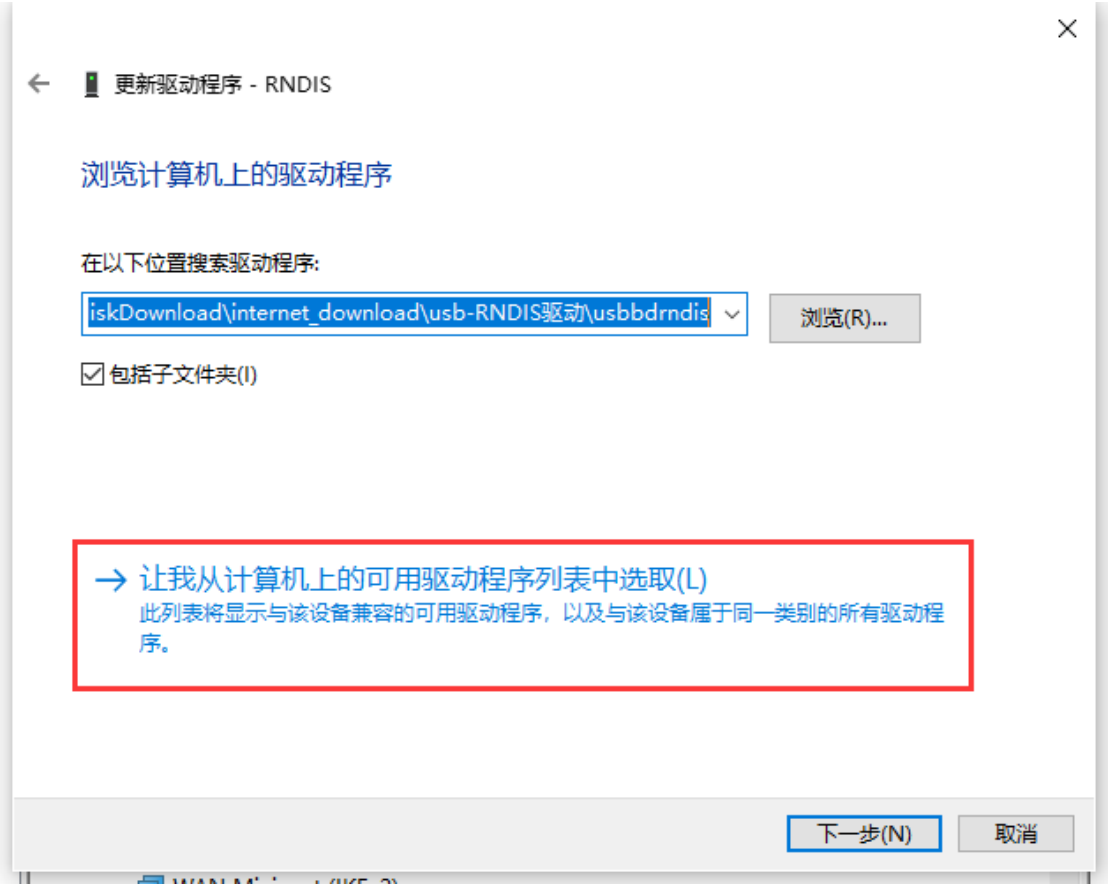
此时打开电脑设备管理器，可以看到电脑检测到其他设备 RNDIS，如下图所示：



此时电脑未安装相关驱动程序，无法进行通讯，右键点击该设备选择更新驱动程序，在弹出的对话框中选择“浏览我的计算机以查找驱动程序软件”



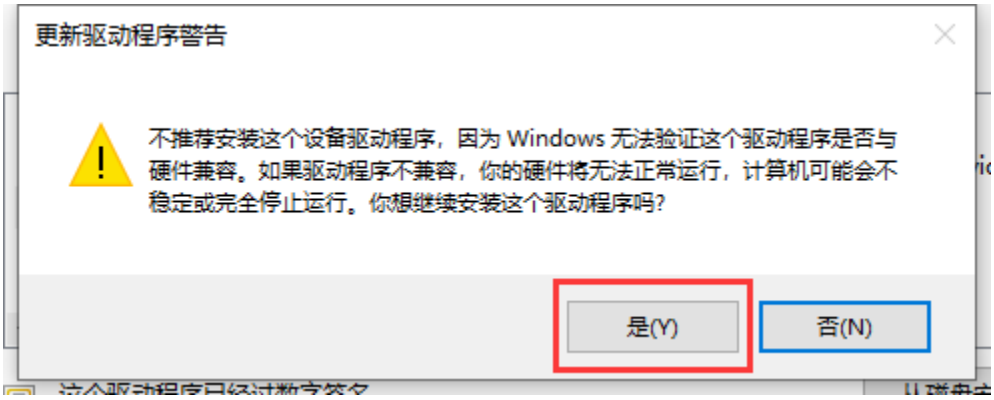
点击“让我从计算机上的可用驱动程序列表中选取”，



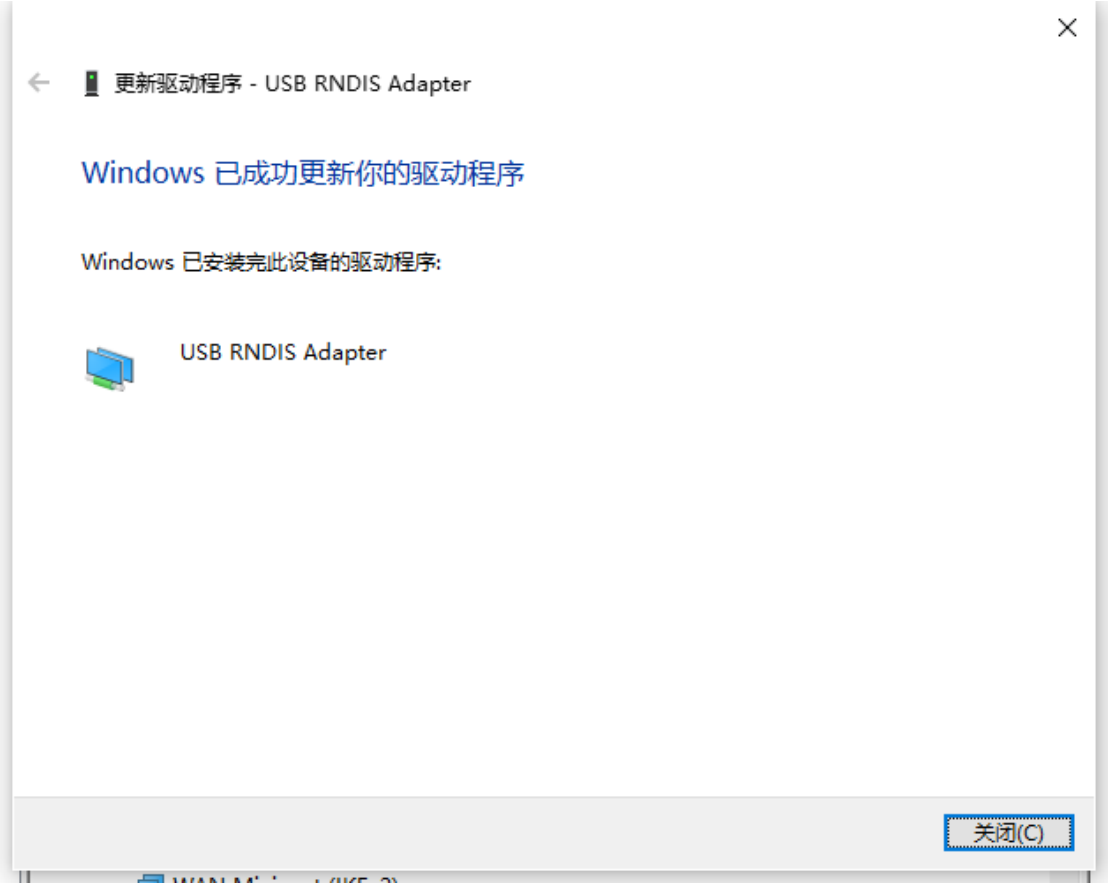
选择“网络适配器”，点击下一步

如下图所示厂商选择 Microsoft，型号选择 USB RNDIS Adapter，点击下一步

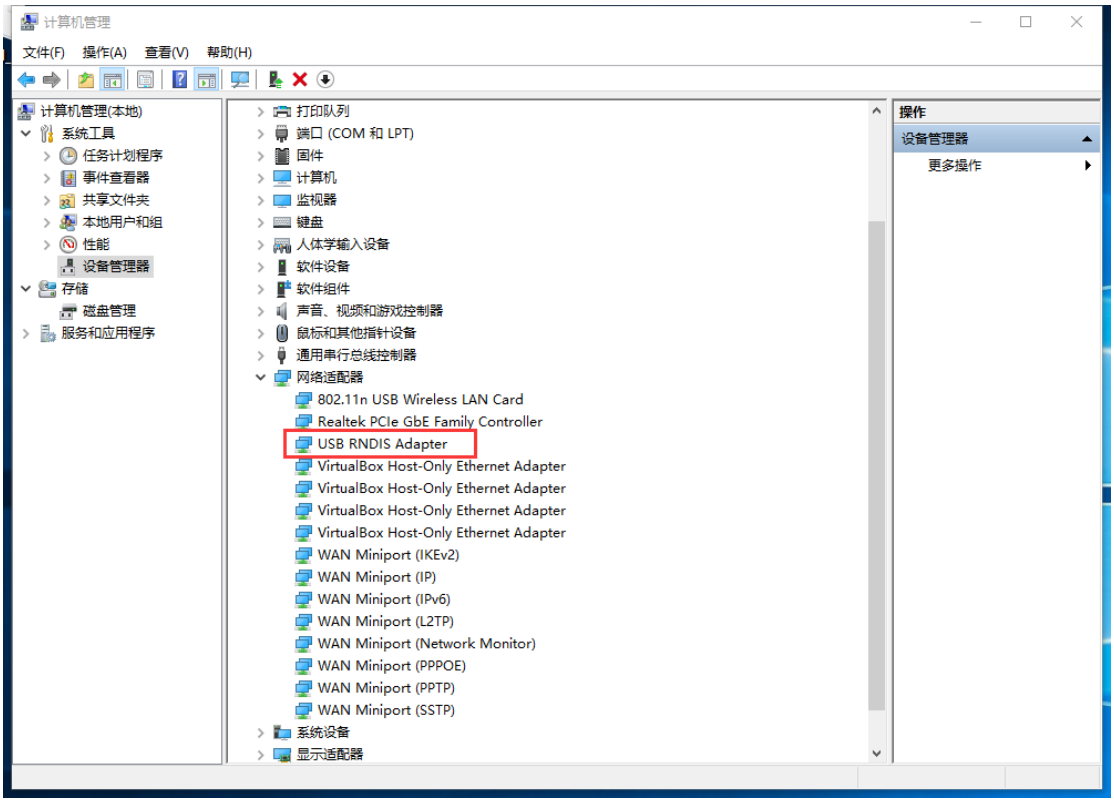
弹出的警告选择是，



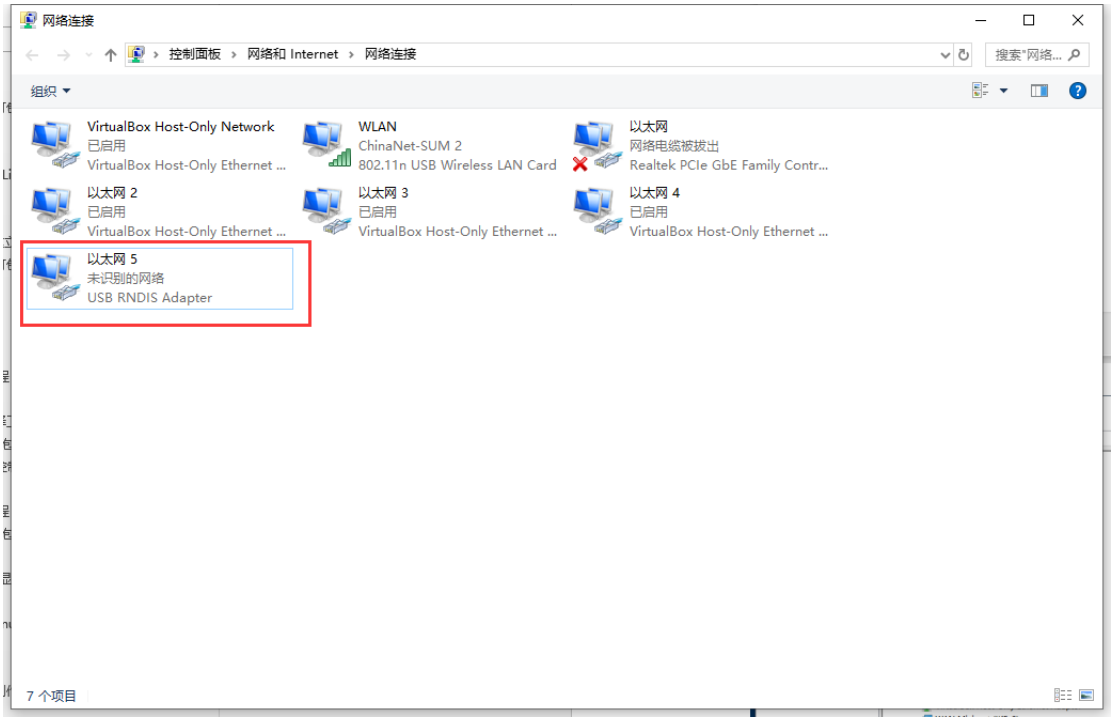
等待驱动程序更新完成。



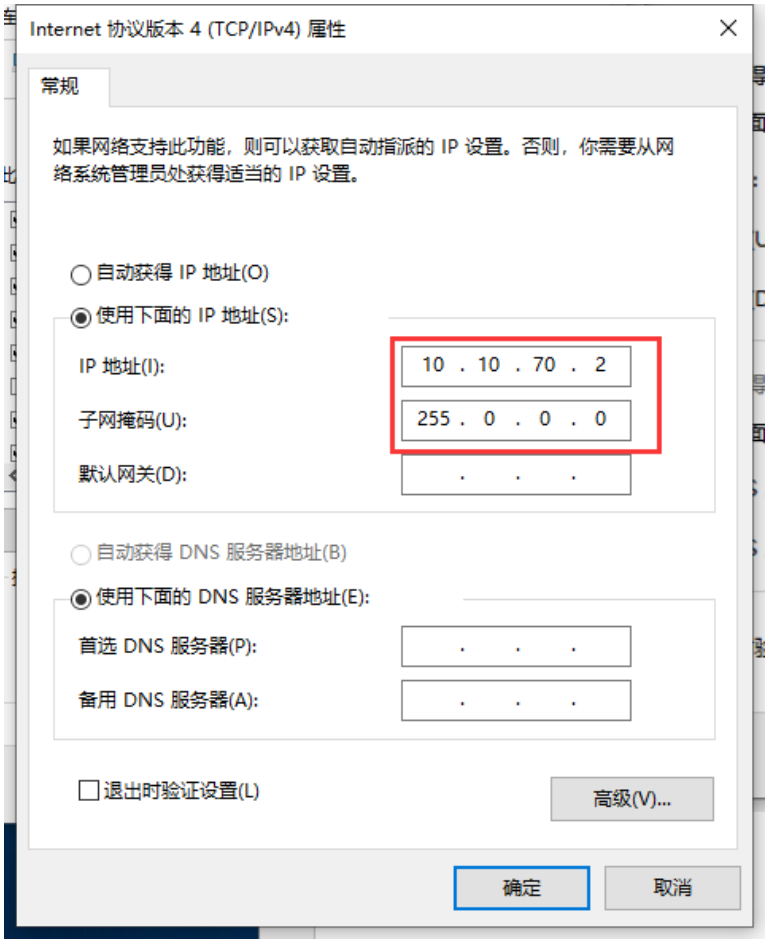
更新完成之后即可看到在网络适配器设备列表中增加了一个设备“USB RNDIS Adapter”，



在网络适配器设置窗口，双击该网络适配器打开设置菜单，



设置 IP 地址与开发板 USB0 的 IP 地址在同一网段，



之后即可测试开发板和电脑能否相互 ping 通，

```
root@petalinux_usb:/sys/kernel/config/usb_gadget/g1# ping 10.10.70.2
PING 10.10.70.2 (10.10.70.2): 56 data bytes
64 bytes from 10.10.70.2: seq=0 ttl=128 time=0.580 ms
64 bytes from 10.10.70.2: seq=1 ttl=128 time=0.415 ms
64 bytes from 10.10.70.2: seq=2 ttl=128 time=0.444 ms
64 bytes from 10.10.70.2: seq=3 ttl=128 time=0.336 ms
64 bytes from 10.10.70.2: seq=4 ttl=128 time=0.461 ms
64 bytes from 10.10.70.2: seq=5 ttl=128 time=0.263 ms
64 bytes from 10.10.70.2: seq=6 ttl=128 time=0.373 ms
64 bytes from 10.10.70.2: seq=7 ttl=128 time=0.510 ms
64 bytes from 10.10.70.2: seq=8 ttl=128 time=0.368 ms
64 bytes from 10.10.70.2: seq=9 ttl=128 time=0.281 ms
64 bytes from 10.10.70.2: seq=10 ttl=128 time=0.423 ms
64 bytes from 10.10.70.2: seq=11 ttl=128 time=0.437 ms
64 bytes from 10.10.70.2: seq=12 ttl=128 time=0.424 ms
64 bytes from 10.10.70.2: seq=13 ttl=128 time=0.278 ms
64 bytes from 10.10.70.2: seq=14 ttl=128 time=0.343 ms
^C
--- 10.10.70.2 ping statistics ---
15 packets transmitted, 15 packets received, 0% packet loss
round-trip min/avg/max = 0.263/0.395/0.580 ms
```

```
C:\Users\mp02>ping 10.10.70.1

正在 Ping 10.10.70.1 具有 32 字节的数据:
来自 10.10.70.1 的回复: 字节=32 时间<1ms TTL=64
来自 10.10.70.1 的回复: 字节=32 时间<1ms TTL=64
来自 10.10.70.1 的回复: 字节=32 时间<1ms TTL=64
来自 10.10.70.1 的回复: 字节=32 时间<1ms TTL=64

10.10.70.1 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms, 最长 = 0ms, 平均 = 0ms

C:\Users\mp02>
```

从上图可以看到开发板和电脑可以通过 USB Ethernet 接口相互 ping 通，说明此时的 USB 接口即可作为 USB 网口使用。

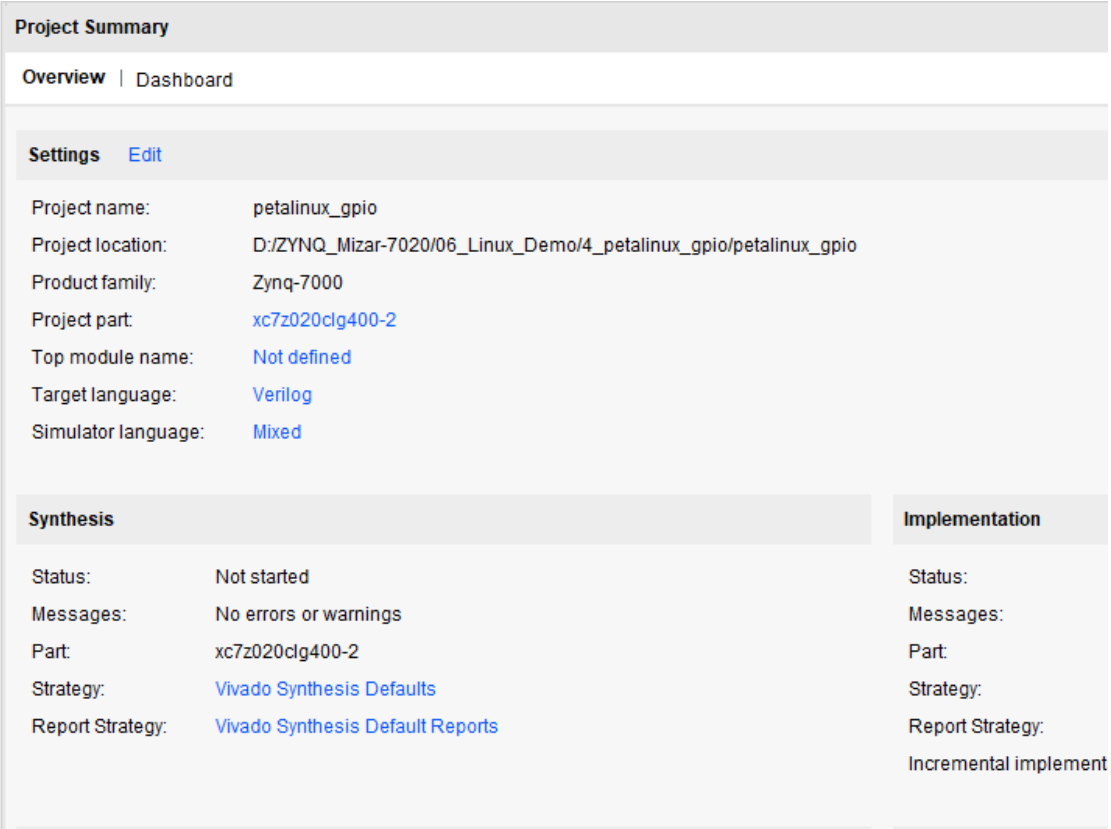
第9章　Linux 下 GPIO 控制实验

经过之前几章的学习，我们了解了一些外设的驱动及使用方式，本章中我们来学习芯片最基本的功能，I/O 口的使用方法。ZYNQ 芯片有 MIO、EMIO、AXI GPIO 三种不同类型 I/O 口，本章中我们同时学习这三种 I/O 在 Linux 中的使用方法。

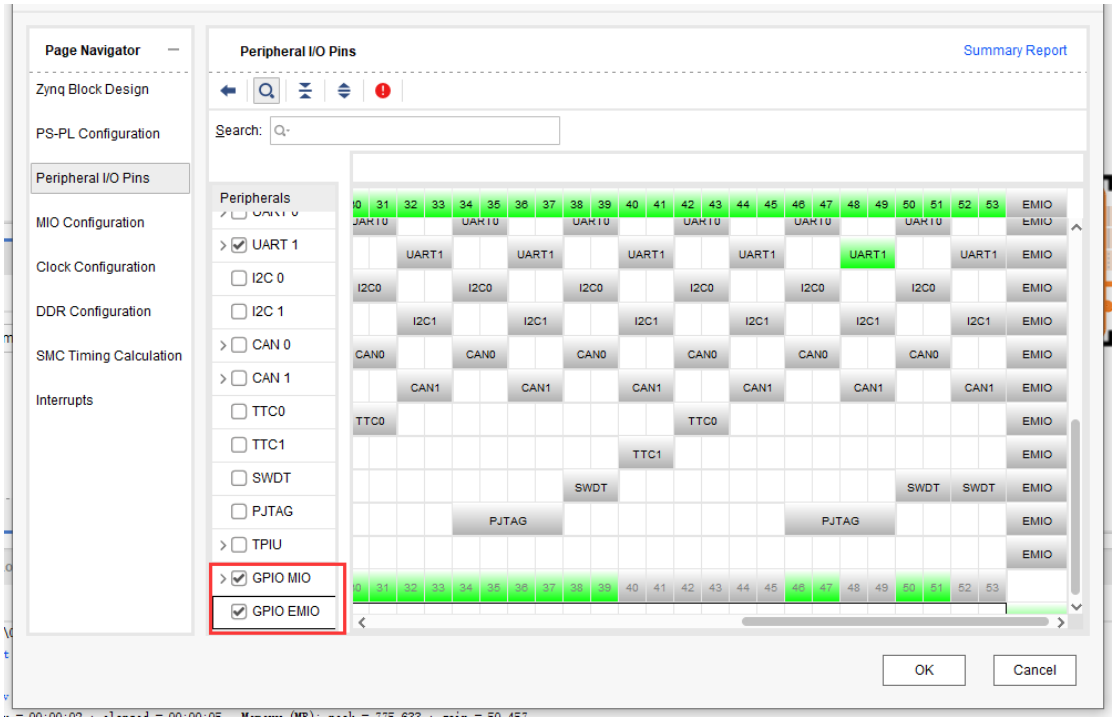
本实验要实现的功能是：PS 端激活外设 MIO，EMIO，SD，UART（用于打印 log），PL 端添加 AXI GPIO IP 核，最终目的是 PetaLinux 从 SD 卡启动，并且可以操控三种 I/O。

9.1 创建 Vivado 工程

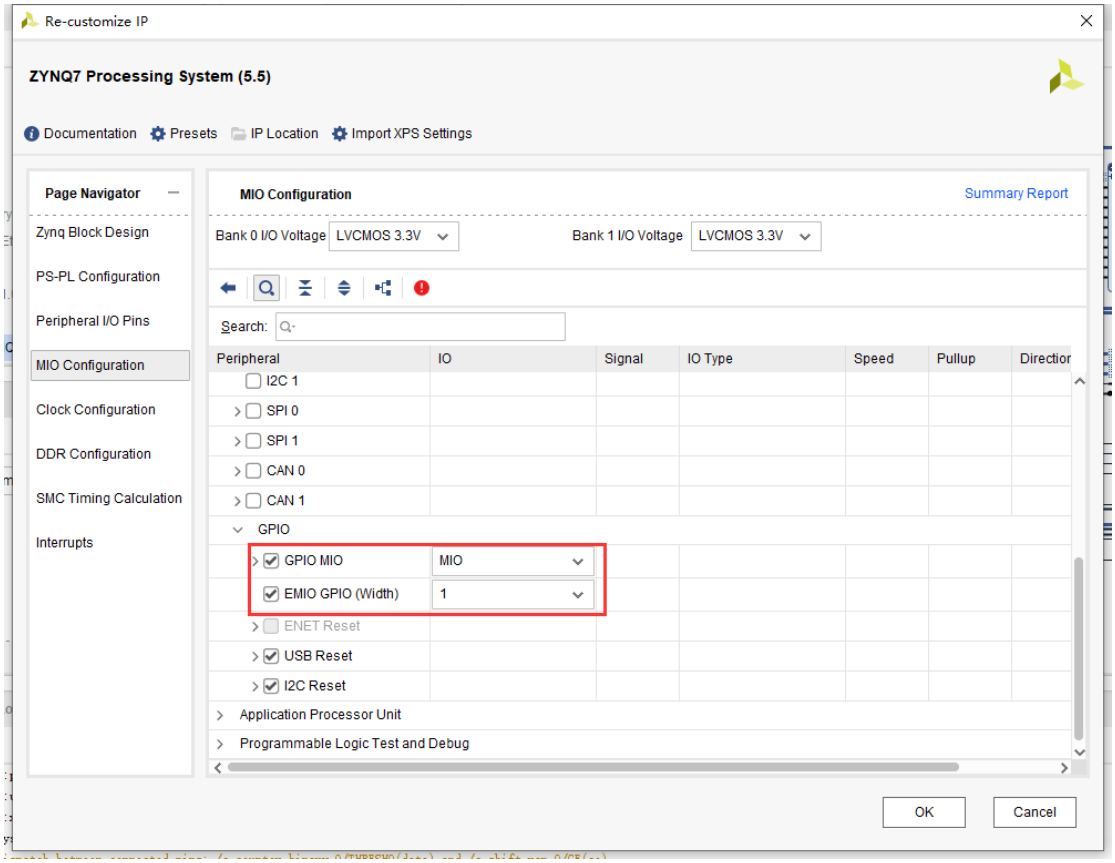
双击 Vivado 图标打开 Vivado 开发平台，打开实验 1 的 Vivado 工程，按照之前的步骤另存为新的工程并命名为“petalinux_gpio”，创建好的工程如下图所示：



工程创建完成后点击 Open Block Design 打开硬件设计，双击 ZYNQ7 Processing System IP 打开配置页面。其它闲置 MIO 全部作为 GPIO， 使能 EMIO，

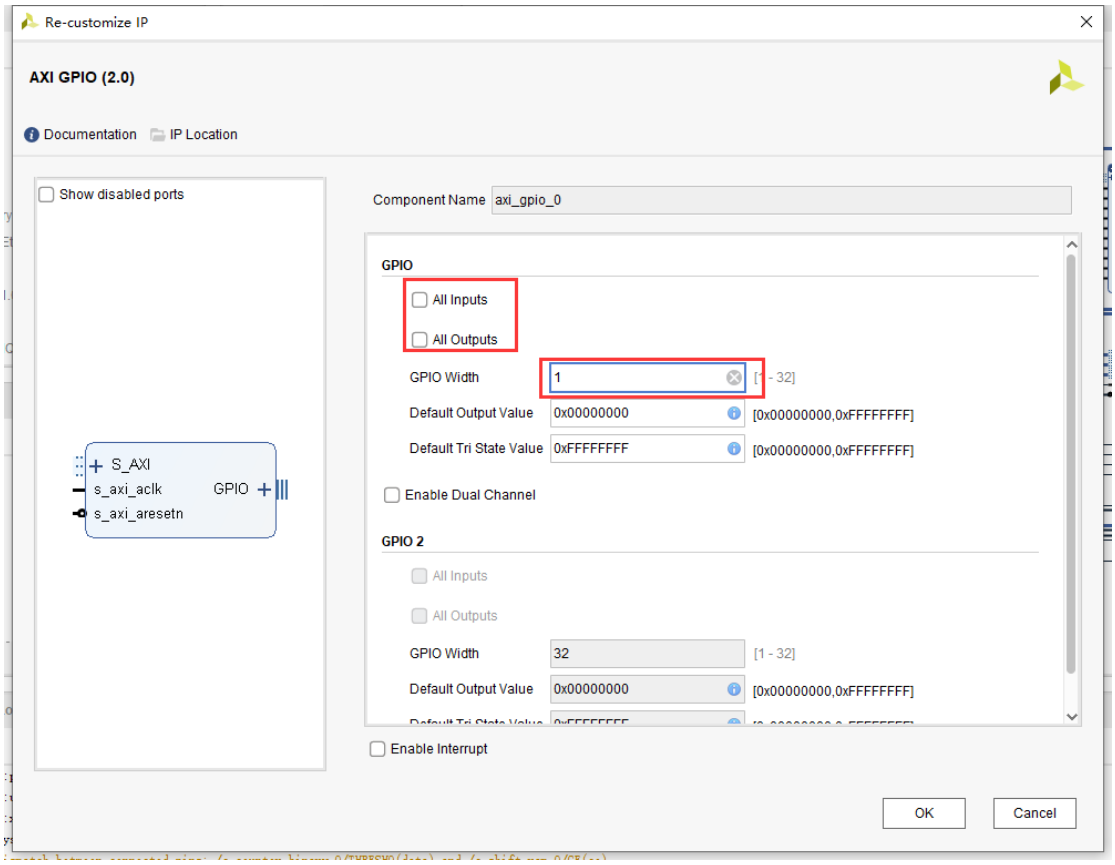


击 MIO Configuration 项，修改 EMIO 的位宽为 1，之后端口映射连接 PL_LED1，

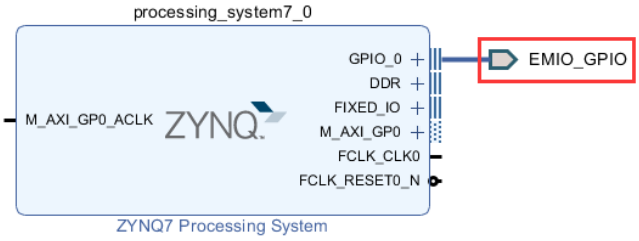


设置完成后点击 OK，之后点击“+”，在搜索框中输入 AXI GPIO 添加 AXI GPIO IP 核，如下图所示：

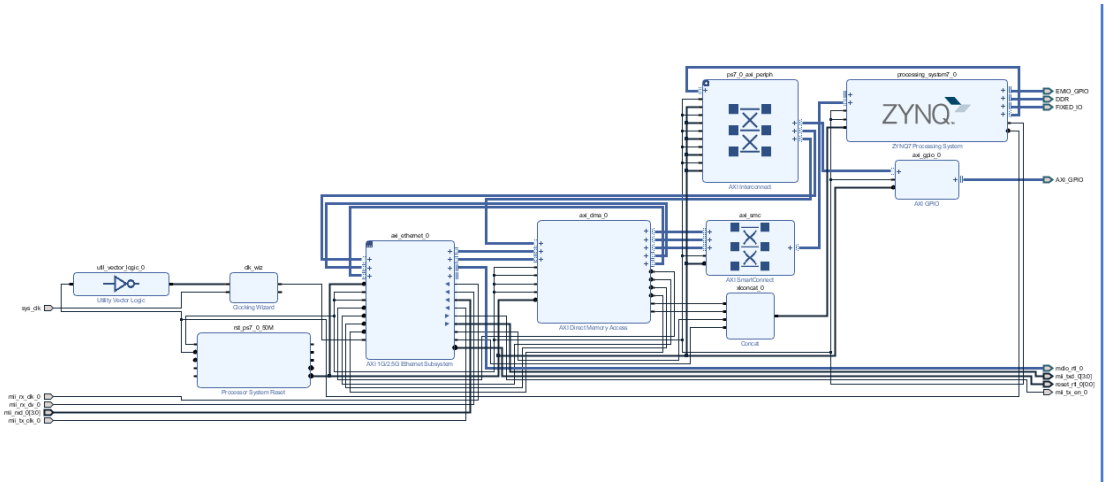
添加完成后双击该 IP 核打开设置页面，设置位宽为 1，连接 PL_LED2，不设置端口默认方向，否则后续无法通过命令修改端口方向，设置完成点击 OK，



右键点击端口将 EMIO 和 AXI GPIO 的端口引出并修改端口名称分别为 EMIO_GPIO 和 AXI_GPIO，方便之后引脚分配时加以区分，

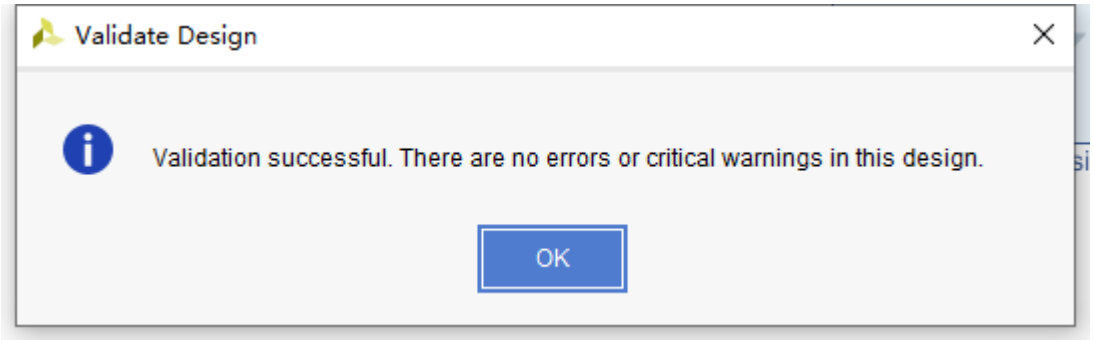


回到 Diagram 窗口，点击 Run Block Automation，自动生成外围管脚，点击 Run Connection Automation，自动连接相关端口，连接完成整个设计如下图所示：



点击工具栏 Validate Design 验证设计，

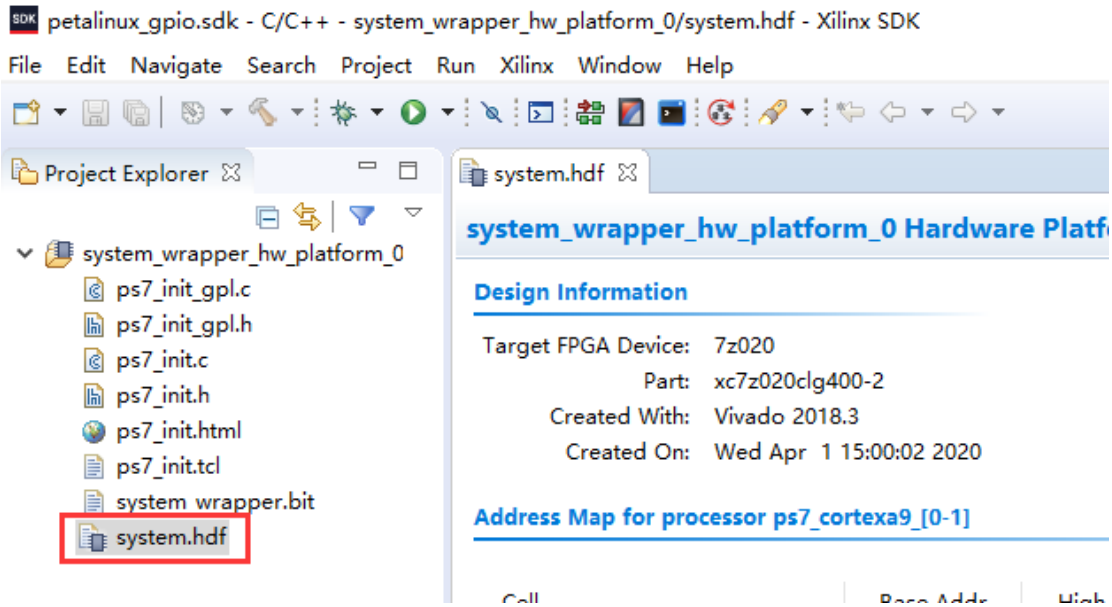
验证成功， Ctrl + S 保存设计。



之后来到 source 窗口，生成设计代码和顶层文件。本实验新增了两个引脚需要分配，如下图所示在之前实验的基础上添加新的分配引脚，AXI GPIO 连接 PL_LED2，EMIO 连接 PL_LED1，

```
set_property IOSTANDARD LVCMOS33 [get_ports {EMIO_GPIO_tri_io[0]}]
set_property PACKAGE_PIN P15 [get_ports {EMIO_GPIO_tri_io[0]}]

set_property IOSTANDARD LVCMOS33 [get_ports {AXI_GPIO_tri_io[0]}]
set_property PACKAGE_PIN U12 [get_ports {AXI_GPIO_tri_io[0]}]
```

之后点击 Generate Bitstream 综合编译生成硬件描述文件，导出并打开 SDK，在 SDK 中可以看到导出的文件如下图所示，我们需要的就是 system.hdf 文件，



将此文件复制到 VM_share 共享文件夹。

9.2 创建 Petalinux 工程

打开 VirtualBox 虚拟机进入 Ubuntu Linux 操作系统，打开 Terminal 窗口，输入下列命令挂载 VM_share 到/mnt，

```
sudo mount –t vboxsf VM_share /mnt
```

```
microphase@microphase-VirtualBox:~$ sudo mount -t vboxsf VM_share /mnt/
[sudo] password for microphase:
microphase@microphase-VirtualBox:~$
```

运行 petalinux 配置文件使 Petalinux 工具可用

```
source /opt/pkg/petalinux/2018.3/setting.sh
```

```
microphase@microphase-VirtualBox: ~
microphase@microphase-VirtualBox:~$ source /opt/pkg/petalinux/2018.3/settings.sh

PetaLinux environment set to '/opt/pkg/petalinux/2018.3'
INFO: Checking free disk space
INFO: Checking installed tools
INFO: Checking installed development libraries
INFO: Checking network and other services
microphase@microphase-VirtualBox:~$
```

进入之前创建的 Petalinux_proj 文件夹，运行下面命令创建 Petalinux 工程

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

```
microphase@microphase-VirtualBox:~/Petalinux_proj$ petalinux-create --type project --template zynq --name petalinux_gpio
INFO: Create project: petalinux_gpio
INFO: New project successfully created in /home/microphase/Petalinux_proj/petalinux_gpio
microphase@microphase-VirtualBox:~/Petalinux_proj$
```

创建完成之后进入工程文件夹，输入下面命令将共享文件/mnt 下的 system.hdf 拷贝到当前文件夹，

```
cp /mnt/system.hdf .
```

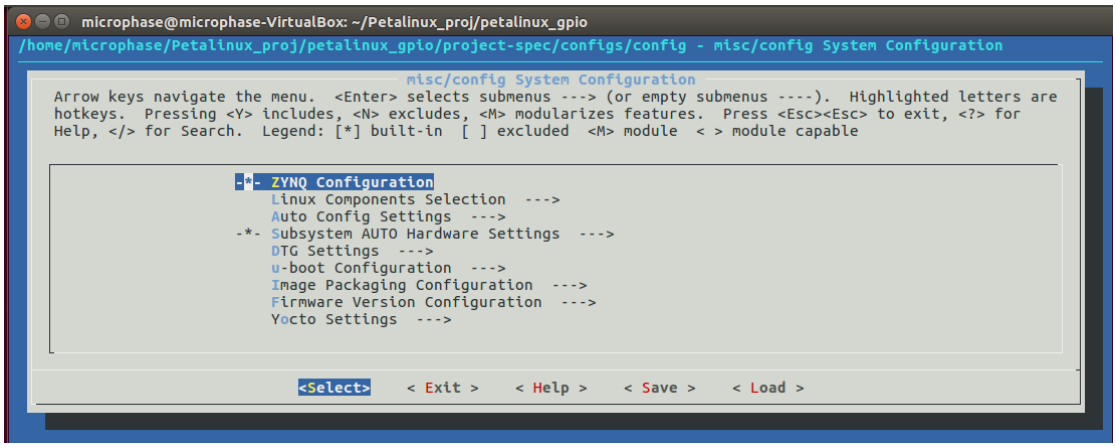
```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_gpio$ cp /mnt/system.hdf .
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_gpio$
```

之后运行下列命令向工程中导入 system.hdf 硬件描述文件，

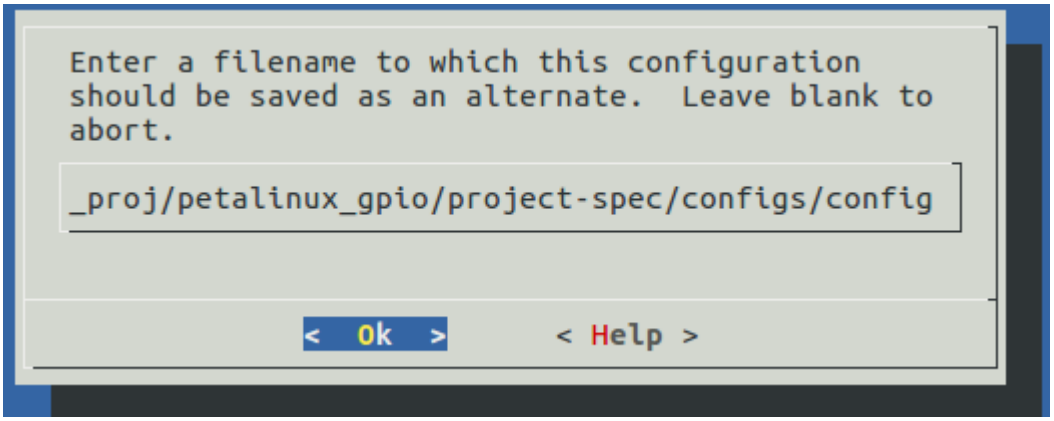
```
petlinux-config --get-hw-description=.
```

```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_gpio$ petalinux-config --get-hw-description=.
INFO: Getting hardware description...
[INFO] generating Kconfig for project
[INFO] menuconfig project


*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.
```

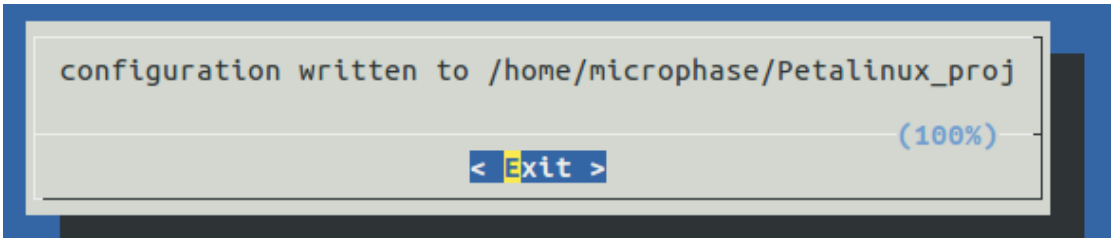
等待一段时间后打开配置目录，如下图所示，本实验需进行修改的设置与实验 1 相同，参照实验一进行设置，之后保存即可，方向键控制光标移动到 Save，点击回车选择，



弹出的对话框选择 OK，



选择 EXIT，之后再选择 EXIT 退出设置页面，



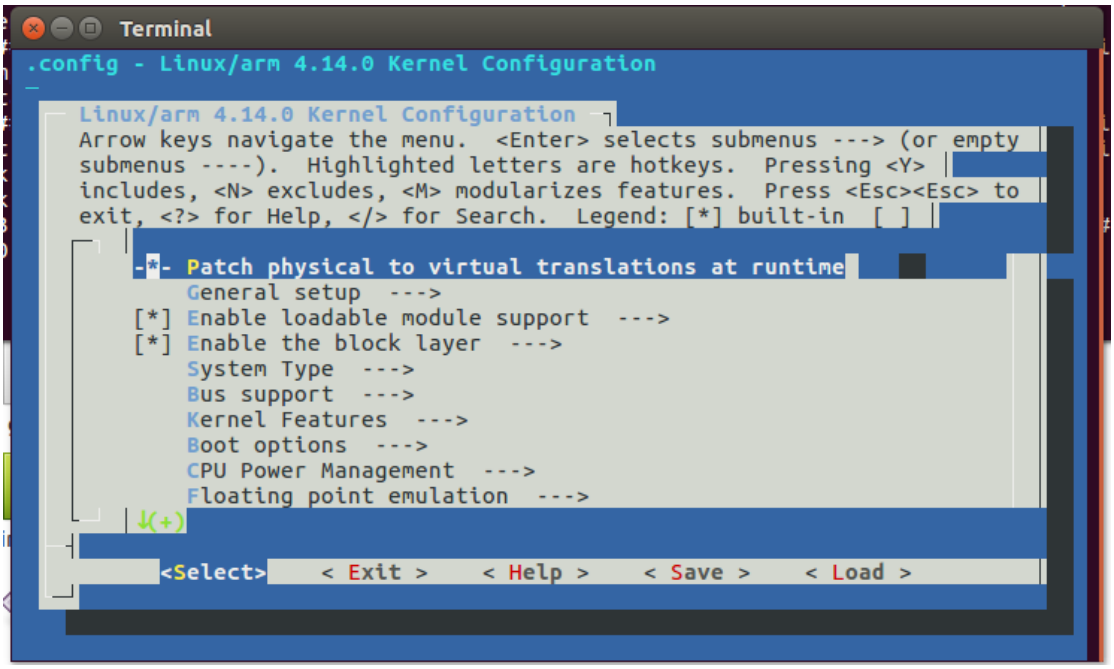
等待一段时间设置完成。

之后输入以下命令配置内核

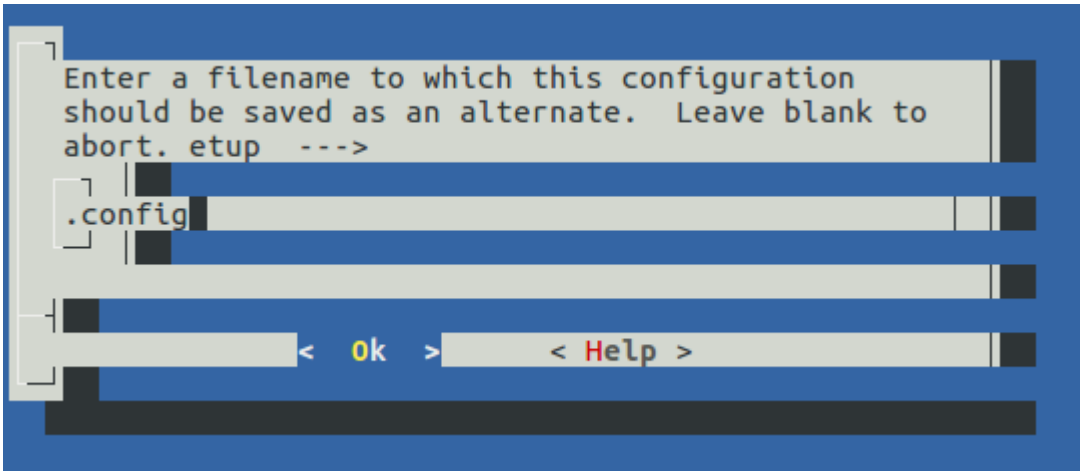
```
petalinux-config –c kernel
```

```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_gpio$ petalinux-config -c kernel
[INFO] generating Kconfig for project
[INFO] sourcing bitbake
[INFO] generating plnxtool conf
[INFO] generating meta-plnx-generated layer
```

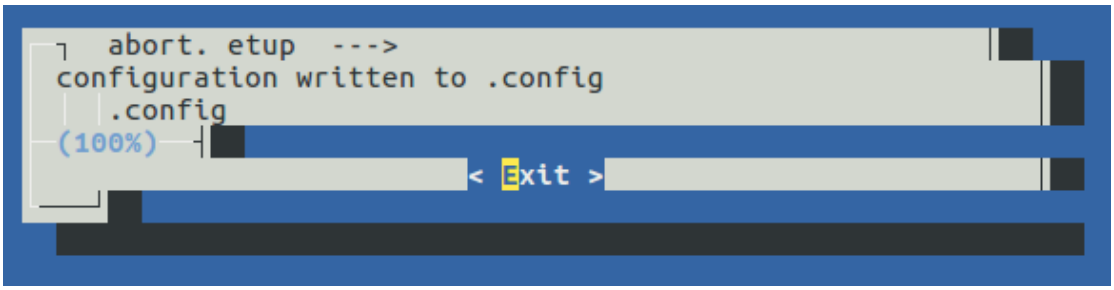
等待一段时间后打开设置页面如下图所示，此处也无需修改配置，直接保存退出即可，移动光标选择 Save，点击回车，



选择 OK，

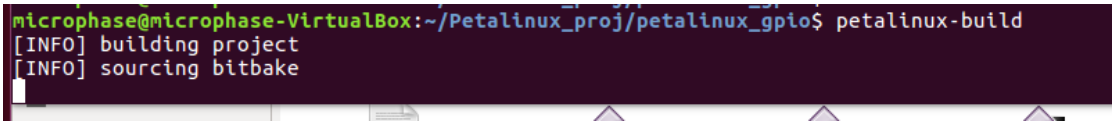


选择 EXIT，在主界面再选择 EXIT 退出设置页面。



配置完成之后按照实验 1 修改设备树文件，添加 etnernet 相关内容，之后即可编译工程，输入以下命令

```
petalinux-build
```



经过漫长时间的等待，工程编译完成

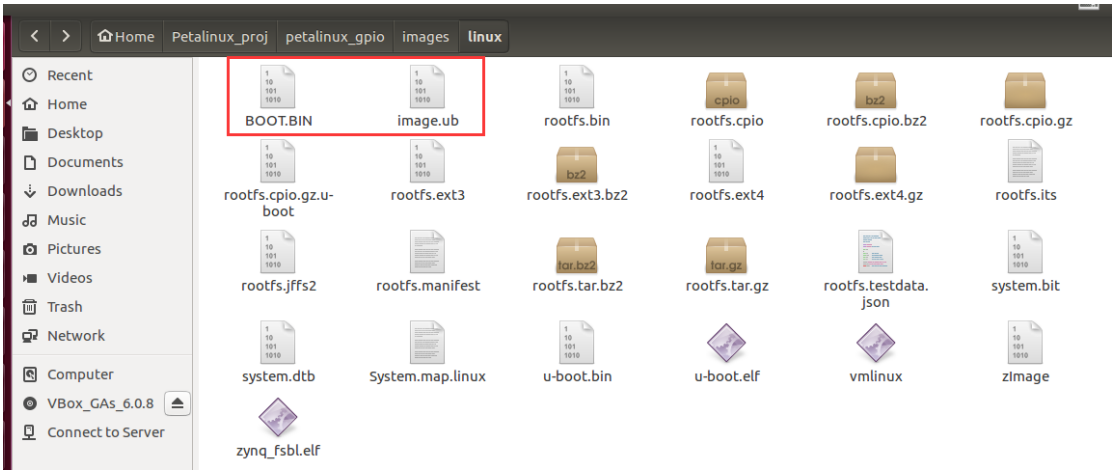
9.3 BOOT.bin 文件打包

除了利用 SDK 软件之外，petalinux 也提供了文件打包的功能，输入如下命令即可，

```
petalinux-package --boot --fsbl ./iamges/linux/zynq_fsbl.elf
--fpga ./imageslinux/system.bit --uboot --force
```

```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_gpio$ petalinux-package --boot --fsbl ./images/linux/zynq_fsbl.elf
 --fpga ./images/linux/system.bit --uboot --force
INFO: File in BOOT BIN: "/home/microphase/Petalinux_proj/petalinux_gpio/images/linux/zynq_fsbl.elf"
INFO: File in BOOT BIN: "/home/microphase/Petalinux_proj/petalinux_gpio/images/linux/system.bit"
INFO: File in BOOT BIN: "/home/microphase/Petalinux_proj/petalinux_gpio/images/linux/u-boot.elf"
INFO: Generating Zynq binary package BOOT.BIN...
```

执行完成即可在/home/Petalinux_proj/petalinux_gpio/images/linux 路径下看到打包完成的文件，



9.4 系统启动验证

将编译打包生成的 BOOT.bin 和 image.ub 文件复制到 SD 卡中，将 SD 卡插入开发板上的卡槽，调整跳线帽设置开发板启动方式为 SD 卡启动。连接串口 usb 线，使用网线连接开发板和路由器。打开 SecureCRT 软件连接串口，上电启动 Linux 操作系统。

系统启动后输入用户名 root，密码 root 登录系统。

sysfs 方式直接操作 GPIO:

按照下图所示进入/sys/class/gpio/文件夹，该文件夹包含 GPIO 操作相关的文件，查看文件夹所包含的文件。从下图可以看出系统识别出的 GPIO 设备有两个，起始编号分别为 905 和 1023，之间的差刚好是 118，就是所有 MIO 和 EMIO 相加的总数。这说明对于系统而言，MIO 和 EMIO 属于同一设备，而 AXI GPIO 属于另一设备，输入 cat gpiochip905/lable，可以看到它属于 zynq_gpio，

```
petalinux_gpio login: root
Password:
root@petalinux_gpio:~# cd /sys/class/gpio/
root@petalinux_gpio:/sys/class/gpio# ls
export        gpiochip1023  gpiochip905   unexport
root@petalinux_gpio:/sys/class/gpio# cat gpiochip905/label
zynq_gpio
root@petalinux_gpio:/sys/class/gpio#
Ready
```

输入 cat gpiochip1023/label，可以看到另一设备的类型及地址，这与 Vivado 工程创建时 AXI GPIO 的地址是相同的，

```
root@petalinux_gpio:/sys/class/gpio# cat gpiochip1023/label
/amba_pl/gpio@41200000
root@petalinux_gpio:/sys/class/gpio#
Ready
```

Cell	Slave Interface	Base Name	Offset Address	Range	High Address
processing_system7_0					
Data (32 address bits : 0x40000000 [1G])					
axi_gpio_0	S_AXI	Reg	0x4120_0000	64K	0x4120_FFFF

查看开发板原理图可知 PS_LED1 连接的是 MIO0，所以在系统中 MIO0 的编号为 905，输入下列命令将 MIO0 节点导入用户空间，

```
echo 905 >    /sys/class/gpio/export
```

```
root@petalinux_gpio:/sys/class/gpio# echo 905 > /sys/class/gpio/export
root@petalinux_gpio:/sys/class/gpio# ls
export        gpio905       gpiochip1023  gpiochip905   unexport
root@petalinux_gpio:/sys/class/gpio#
Ready                                                   Serial: COM7
```

输入下列命令查看 I/O 口方向及电平，可以看到端口方向为输入，电平为 1，

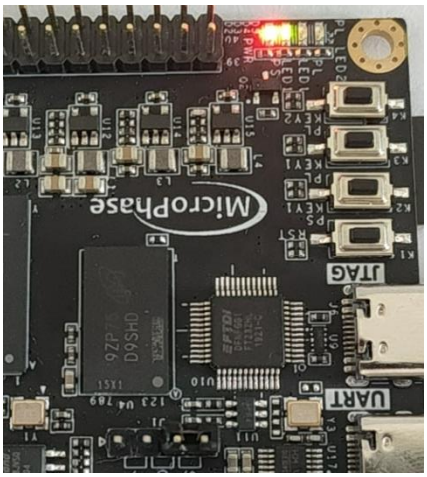
```
cat gpio905/direction
cat gpio905/value
```

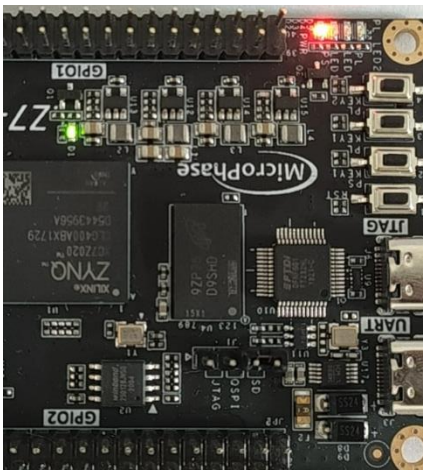
```
root@petalinux_gpio:/sys/class/gpio# cat gpio905/direction
in
root@petalinux_gpio:/sys/class/gpio# cat gpio905/value
1
root@petalinux_gpio:/sys/class/gpio#
Ready                                   Serial: COM7   43,  38   43 F
```

所以如果我们希望点亮 LED，那么应该将 I/O 口方向改为输出，输出电平为 0，输入下列命令修改端口方向，可以看到 led 点亮，这是因为修改端口方向后，默认输出为 0，之后修改输出为 1，可以看到 LED 熄灭。

```
echo out > gpio905/direction
cat gpio05/direction
echo 1 > gpio05/value
cat gpio905/ value
```

```
root@petalinux_gpio:/sys/class/gpio# echo out > gpio905/direction
root@petalinux_gpio:/sys/class/gpio# cat gpio905/direction
out
root@petalinux_gpio:/sys/class/gpio# echo 1 > gpio905/value
root@petalinux_gpio:/sys/class/gpio# cat gpio905/value
1
root@petalinux_gpio:/sys/class/gpio#
```





PL_LED1 连接 EMIO0，那么其 gpio 编号为 54，在系统中对应 gpio959，按照之前的方式将其加入用户空间，查看其端口方向和电平，

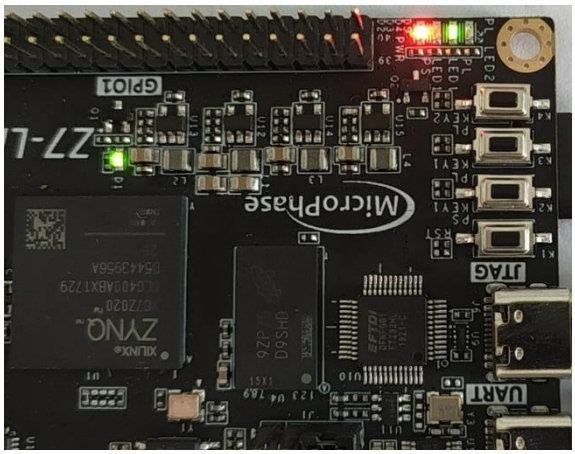
```
echo 959 >    /sys/class/gpio/export
cat gpio959/direction
cat gpio959/value
```

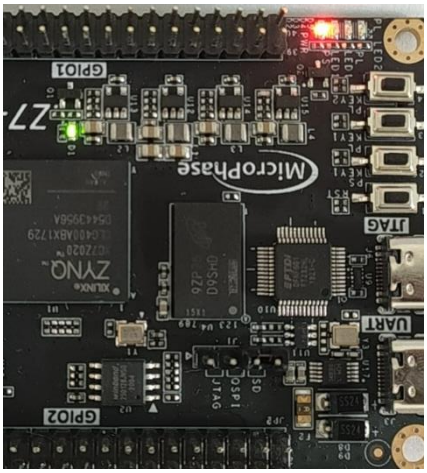
```
root@petalinux_gpio:/sys/class/gpio# echo 959 > /sys/class/gpio/export
root@petalinux_gpio:/sys/class/gpio# cat gpio959/direction
in
root@petalinux_gpio:/sys/class/gpio# cat gpio959/value
1
root@petalinux_gpio:/sys/class/gpio#
```

修改端口方向为输出，默认输出 0，PL_LED1 点亮，修改输出为 1，PL_LED1 熄灭。

```
echo out > gpio959/direction
cat gpio959/direction
echo 1 > gpio959/value
cat gpio959/ value
```

```
root@petalinux_gpio:/sys/class/gpio# echo out > gpio959/direction
root@petalinux_gpio:/sys/class/gpio# cat gpio959/direction
out
root@petalinux_gpio:/sys/class/gpio# echo 1 > gpio959/value
root@petalinux_gpio:/sys/class/gpio# cat gpio959/value
1
root@petalinux_gpio:/sys/class/gpio#
```





PL_LED2 连接 AXI GPIO0，那么在系统中它的编号为 gpio1023，按照之前的操作将 gpio1023 加入用户空间，查看其端口方向及电平，

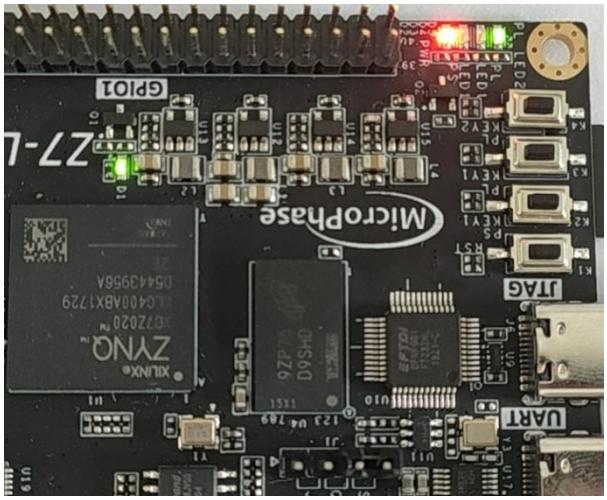
```
echo 1023 >    /sys/class/gpio/export
cat gpio1023/direction
cat gpio1023 /value
```

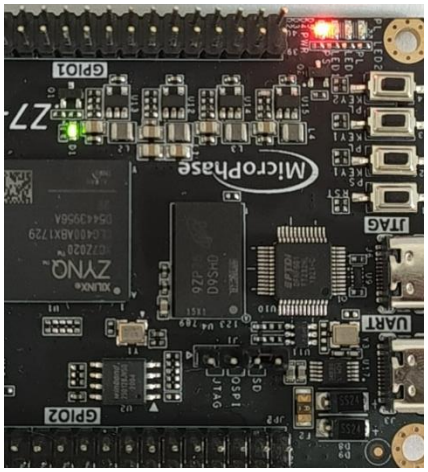
```
root@petalinux_gpio:/sys/class/gpio# echo 1023 > /sys/class/gpio/export
root@petalinux_gpio:/sys/class/gpio# cat gpio1023/direction
in
root@petalinux_gpio:/sys/class/gpio# cat gpio1023/value
1
root@petalinux_gpio:/sys/class/gpio#
Ready                    Serial: COM7    43,  38    43 Rows, 100 Cols    V
```

修改端口方向为输出，默认输出 0，PL_LED3 点亮，修改输出为 1，PL_LED3 熄灭。

```
echo out > gpio1023/direction
cat gpio1023/direction
echo 1 > gpio1023/value
cat gpio1022/ value
```

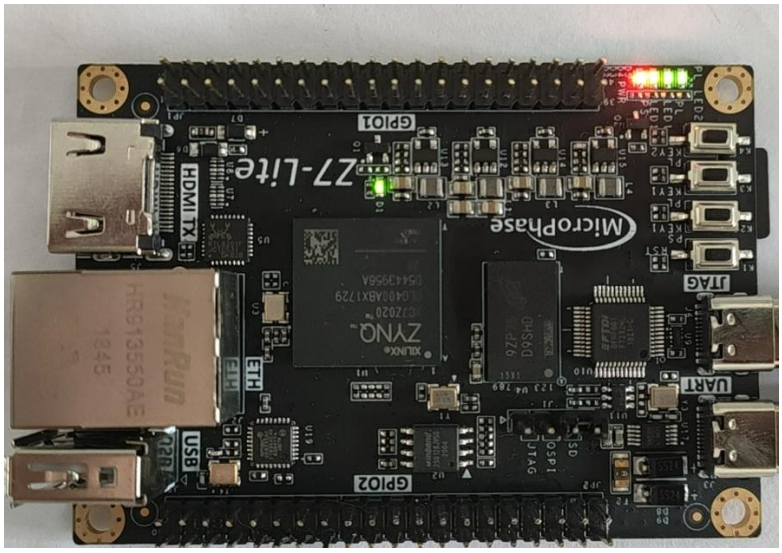
```
root@petalinux_gpio:/sys/class/gpio# echo out > gpio1023/direction
root@petalinux_gpio:/sys/class/gpio# cat gpio1023/direction
out
root@petalinux_gpio:/sys/class/gpio# echo 1 > gpio1023/value
root@petalinux_gpio:/sys/class/gpio# cat gpio1023/value
1
root@petalinux_gpio:/sys/class/gpio#
Ready                    Serial: COM7    43,  38    43 Rows, 100 Co
```





最后我们将三个 LED 全部点亮

```
root@petalinux_gpio:/sys/class/gpio# echo 0 > gpio1023/value
root@petalinux_gpio:/sys/class/gpio# echo 0 > gpio905/value
root@petalinux_gpio:/sys/class/gpio# echo 0 > gpio959/value
root@petalinux_gpio:/sys/class/gpio#
```



下面对上面的一些操作进行说明：

/sys/class/gpio/路径下相关文件说明：

File	Description
export	将GPIO节点导入到用户空间
unexport	将GPIO节点移除用户空间
gpiochip*	该目录下保存系统中GPIO寄存器信息：每个寄存器控制引脚的起始编号base，寄存器名称，引脚总数

/sys/class/gpio/gpio960/路径下相关文件说明：

File	Description
direction	配置GPIO口输方面，"in"为输入；"out"为输出
value	GPIO为输出时可设置GPIO口电平，"0"为低电平；"1"为高电平。
edge	设置GPIO口输入时触发方式：“none”, “rising”, “falling”, or “both”
active_low	按照官方的说法是翻转电平，输入非零的数即上升沿变为下降沿

这就是通过命令行的方式操作 I/O 口的方法，除此之外，还可使用 APP 应用程序的方式控制 I/O 口。

GPIO 用户空间 APP 方式操作 I/O 口：

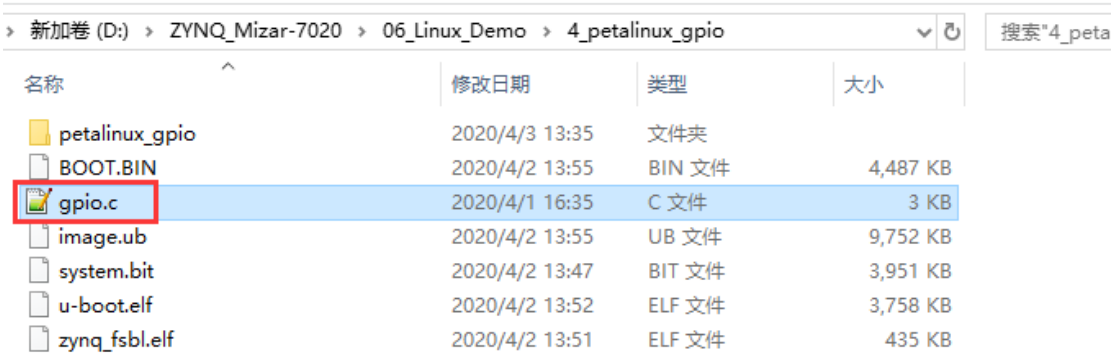
首先在虚拟机 Ubuntu 下编写 gpio.c 程序，程序内容如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>


// The specific GPIO being used must be setup and replaced thru
// this code.  The GPIO of 897 is in the path of most the sys dirs
// and in the export write.
//
// Figuring out the exact GPIO was not totally obvious when there
// were multiple GPIOs in the system. One way to do is to go into
// the gpiochips in /sys/class/gpio and view the label as it should
// reflect the address of the GPIO in the system. The name of the
// the chip appears to be the 1st GPIO of the controller.
//
// The export causes the gpio897 dir to appear in /sys/class/gpio.
// Then the direction and value can be changed by writing to them.

// The performance of this is pretty good, using a nfs mount,
// running on open source linux,
// the GPIO can be toggled about every 1sec.
// The following commands from the console setup the GPIO to be
// exported, set the direction of it to an output and write a 1
// to the GPIO.
//
// bash> echo 897 > /sys/class/gpio/export
// bash> echo out > /sys/class/gpio/gpio897/direction
// bash> echo 1 > /sys/class/gpio/gpio897/value

// if sysfs is not mounted on your system, the you need to mount it
// bash> mount -t sysfs sysfs /sys

// the following bash script to toggle the gpio is also handy for
// testing
//
// while [ 1 ]; do
//  echo 1 > /sys/class/gpio/gpio897/value
//  echo 0 > /sys/class/gpio/gpio897/value
// done
```

```
// to compile this, use the following command
// gcc gpio.c -o gpio

// The kernel needs the following configuration to make this work.
//
// CONFIG_GPIO_SYSFS=y
// CONFIG_SYSFS=y
// CONFIG_EXPERIMENTAL=y
// CONFIG_GPIO_XILINX=y

int main()
{
    int valuefd, exportfd, directionfd;

    printf("GPIO test running...\n");

    // The GPIO has to be exported to be able to see it
    // in sysfs

exportfd = open("/sys/class/gpio/export", O_WRONLY);
if (exportfd < 0)
{
    printf("Cannot open GPIO to export it\n");
    exit(1);
}

write(exportfd, "959", 4);
close(exportfd);

printf("GPIO exported successfully\n");

// Update the direction of the GPIO to be an output

directionfd = open("/sys/class/gpio/gpio959/direction", O_RDWR);
if (directionfd < 0)
{
    printf("Cannot open GPIO direction it\n");
    exit(1);
}

write(directionfd, "out", 4);
close(directionfd);

printf("GPIO direction set as output successfully\n");

// Get the GPIO value ready to be toggled

valuefd = open("/sys/class/gpio/gpio959/value", O_RDWR);
if (valuefd < 0)
{
    printf("Cannot open GPIO value\n");
    exit(1);
}

printf("GPIO value opened, now toggling...\n");

// toggle the GPIO as fast a possible forever, a control c is needed
// to stop it
```

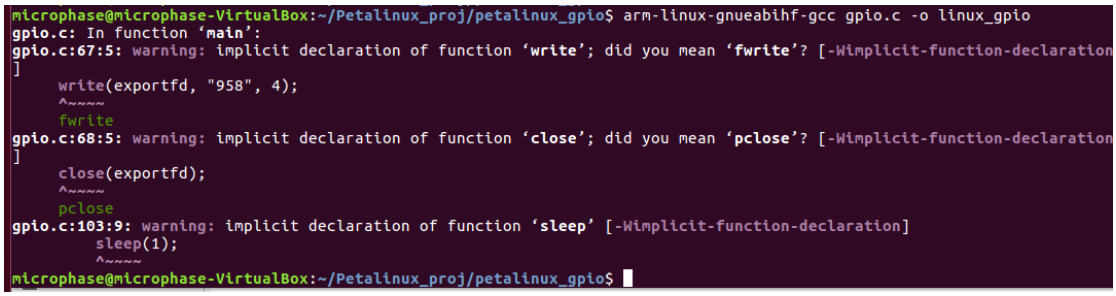
```
    while (1)
    {
        write(valuefd,"1", 2);
        sleep(1);
        write(valuefd,"0", 2);
        sleep(1);
    }
}
```

该程序已放到该实验例程文件夹内，可直接复制到共享文件夹，之后复制到虚拟机中使用，

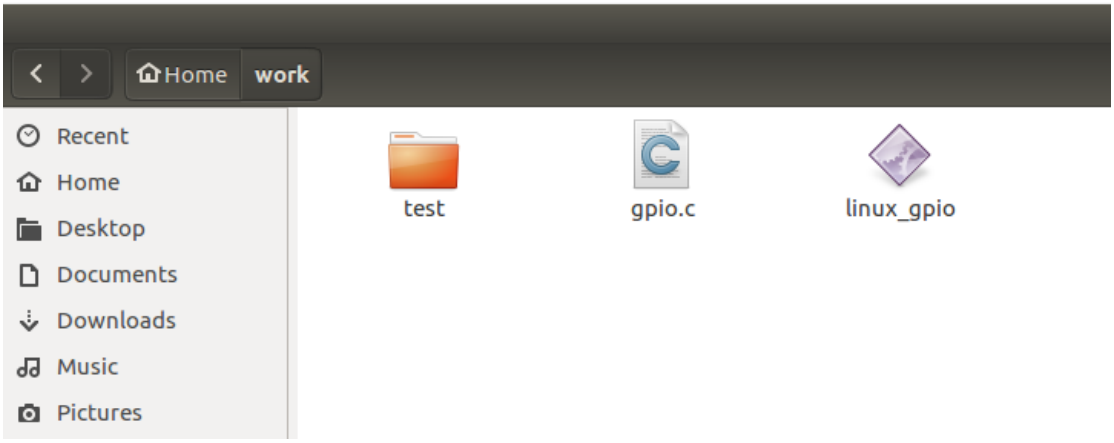


程序编写完成之后使用交叉编译工具链编译程序生成可执行文件，

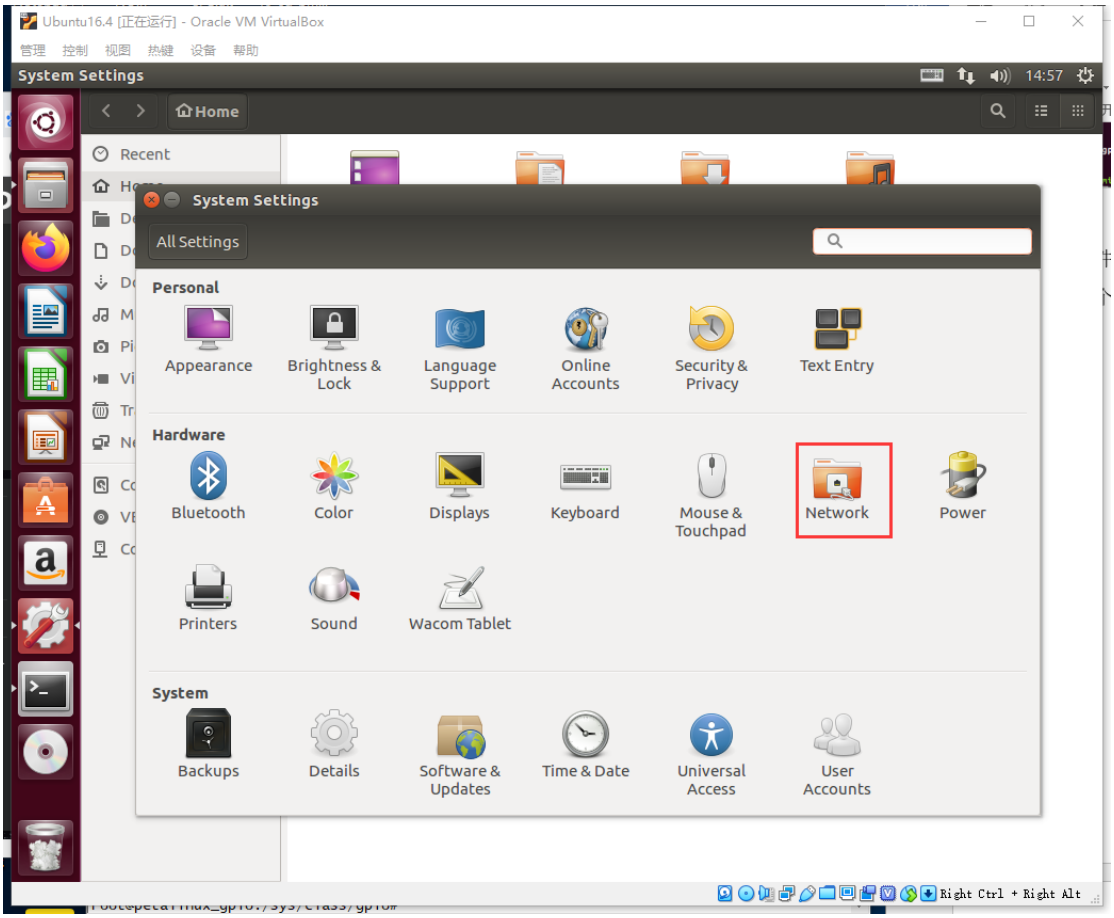
```
arm-linux-gnueabihf-gcc    gpio.c    –o    linux_gpio
```



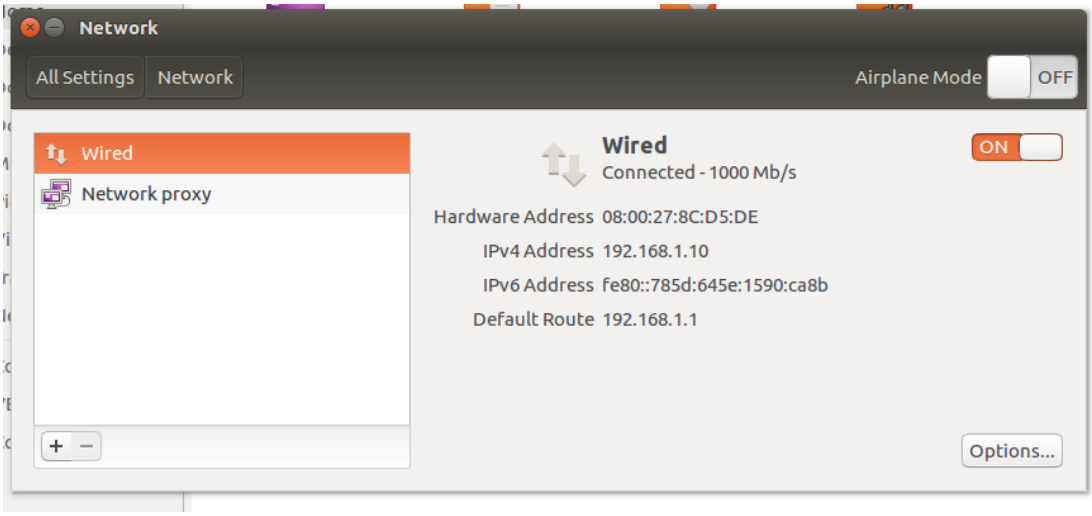
之后我们就要让此文件在开发板上运行。在之前的教程中讲解了 NFS 服务软件的安装，并创建了一个文件夹 work 作为共享文件夹，将 gpio.c 和 linux_gpio 两个文件复制到该共享文件夹内，如下图所示：



之后打开虚拟机设置页面查看虚拟机 IP 地址设置，确认虚拟机、电脑 Windows 操作系统、开发板 IP 处于同一网段。



虚拟机 IP 地址为 192.168.1.10



开发板 IP 地址为 192.168.1.6，

```
root@petalinux_gpio:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0A:35:00:1E:53
          inet addr:192.168.1.6  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  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)
          Interrupt:27 Base address:0xb000

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: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)
```

并且开发板输入 ping 192.168.1.10 可以 ping 通虚拟机。

之后在开发板端输入下列命令通过 NFS 挂载到虚拟机，

mount –t nfs -o nolock 192.168.1.10(虚拟机 IP 地址):/home/microphase（用户名）/work　/mnt

```
mount –t nfs -o nolock 192.168.1.10:/home/microphase/work /mnt
```

```
root@petalinux_gpio:/sys/class/gpio# mount -t nfs -o nolock 192.168.1.10:/home/microphase
/work /mnt
root@petalinux_gpio:/sys/class/gpio# ls /mnt/
gpio.c      linux_gpio  test
```

挂载成功后可以看到在/mnt 下的文件与/home/microphase/work 文件夹下的相同，进入/mnt，输入下列命令运行 linux_gpio 可执行程序，

```
./linux_gpio
```

可以通过输出信息看到程序正在运行，gpio958 即 PL_LED1 端口输出电平没秒钟反转一次，PL_LED1 不停闪烁。输入 Ctrl + c 可以停止执行程序。

```
root@petalinux_gpio:/mnt# ls
gpio.c      linux_gpio  test
root@petalinux_gpio:/mnt# ./linux_gpio
GPIO test running...
GPIO exported successfully
GPIO direction set as output successfully
GPIO value opened, now toggling...
```

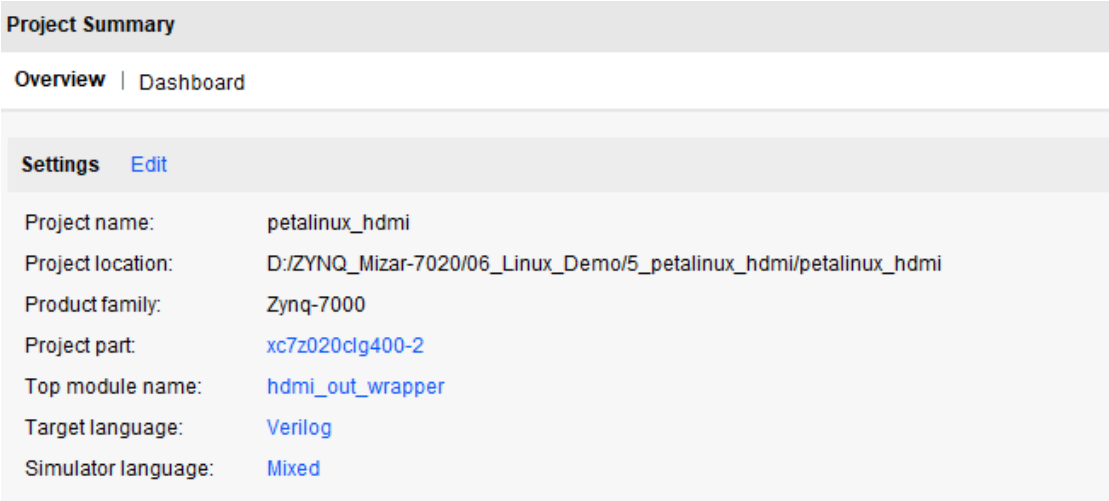
读者可以在此基础上尝试实现按键控制 LED 或者流水灯等功能，多加练习熟练掌握 I/O 口的使用方法。

第10章　Linux 下 HDMI 显示实验

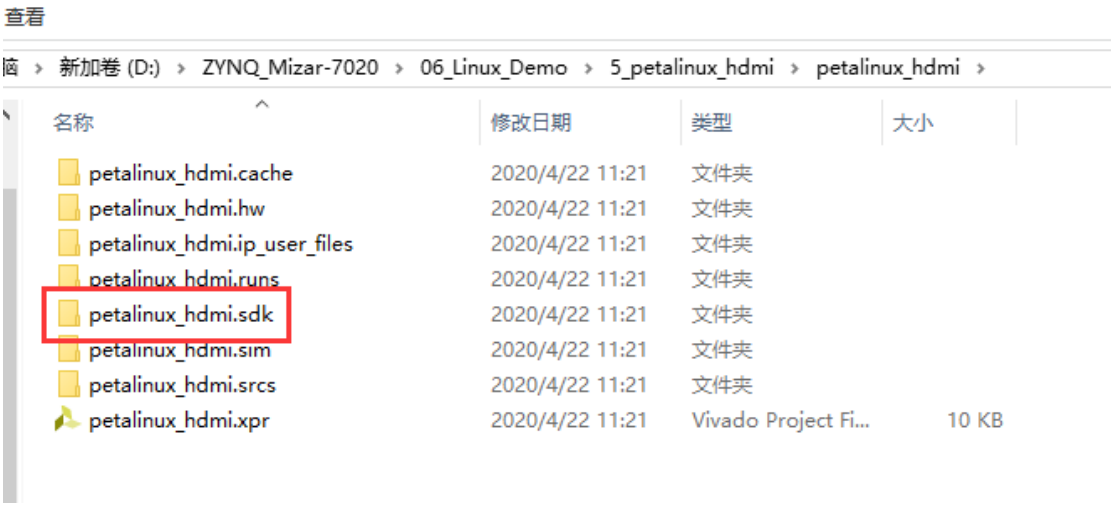
之前在 SDK 的实验中，我们通过添加自定义 IP 的方式，利用开发板上的 HDMI 接口实现了将图片显示在屏幕上的结果。在本章中，我们来学习怎样在之前实验硬件设计的基础上，修改 Linux 内核驱动程序，实现在 Linux 下的 HDMI 显示。

10.1 创建 Vivado 工程

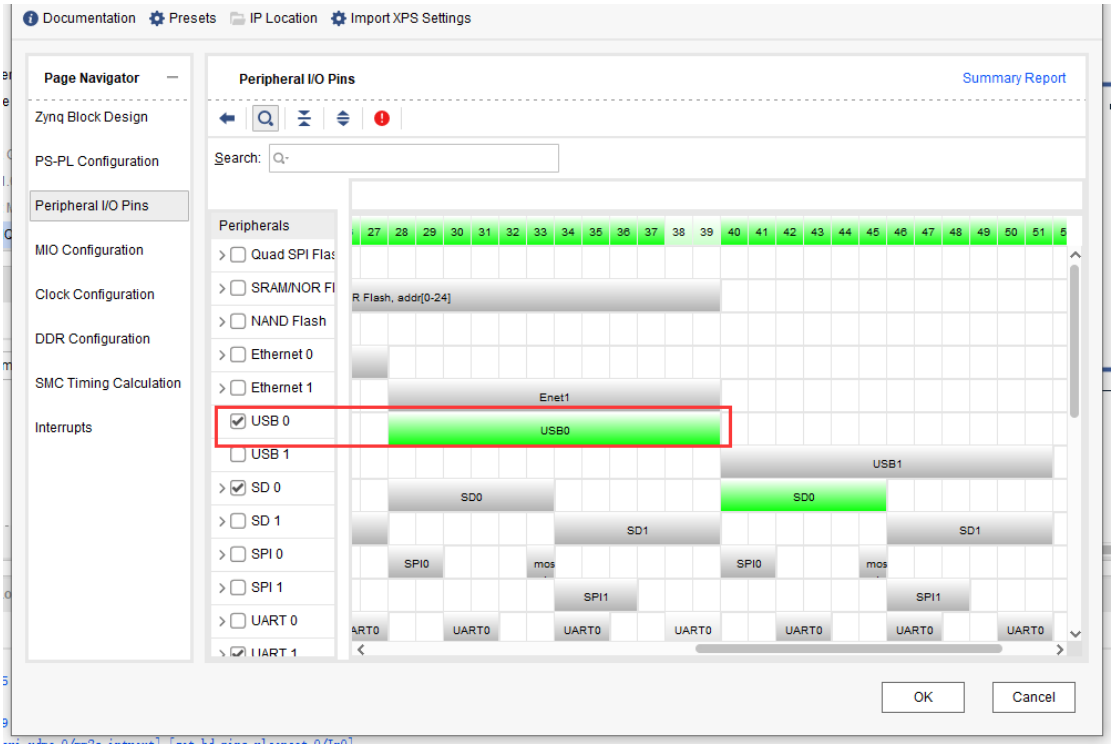
打开 SDK 实验例程 sd_hdmi_out 的 Vivado 工程，另存为新的工程并命名为“petalinux_hdmi”，



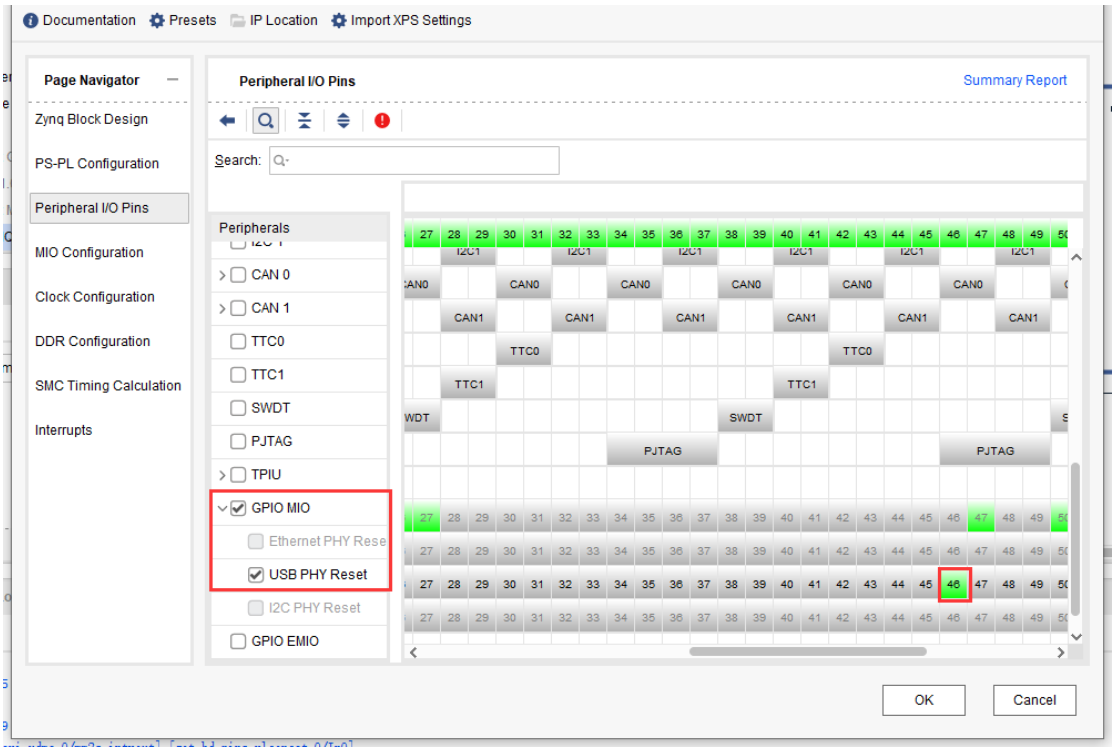
之后删除新工程下的.sdk 文件夹。



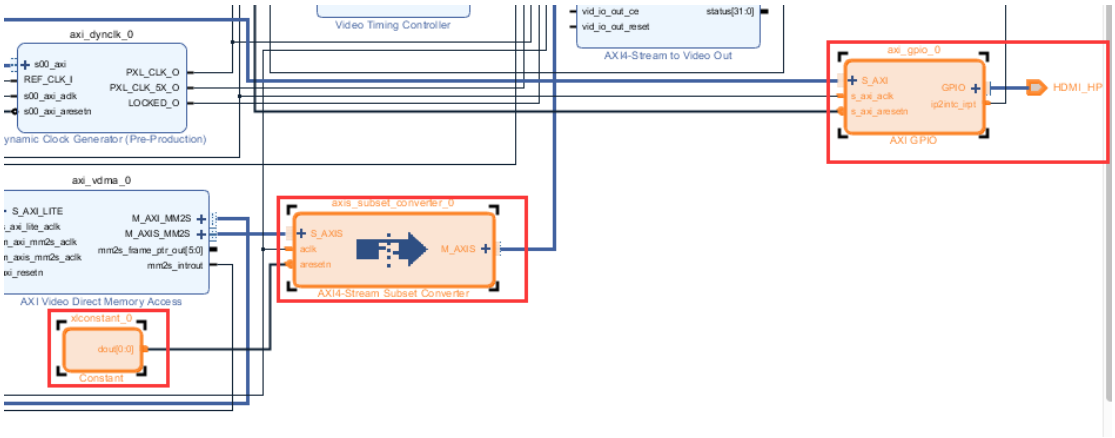
在 Vivado 软件中点击 Open Block Design 打开设计页面，打开 ZYNQ7 Processing System IP 的设置页面，使能 USB 端口，



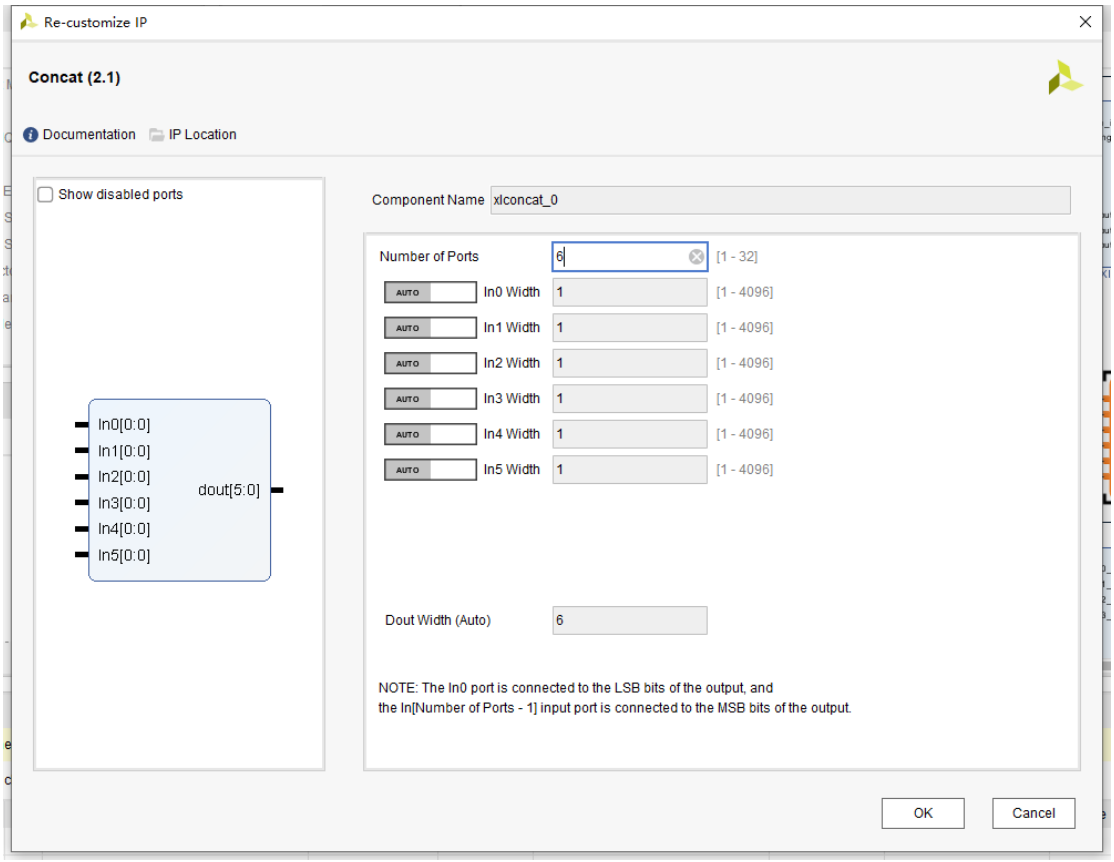
使能 GPIO MIO，设置 MIO46 为 USB PHY Reset，如下图所示：



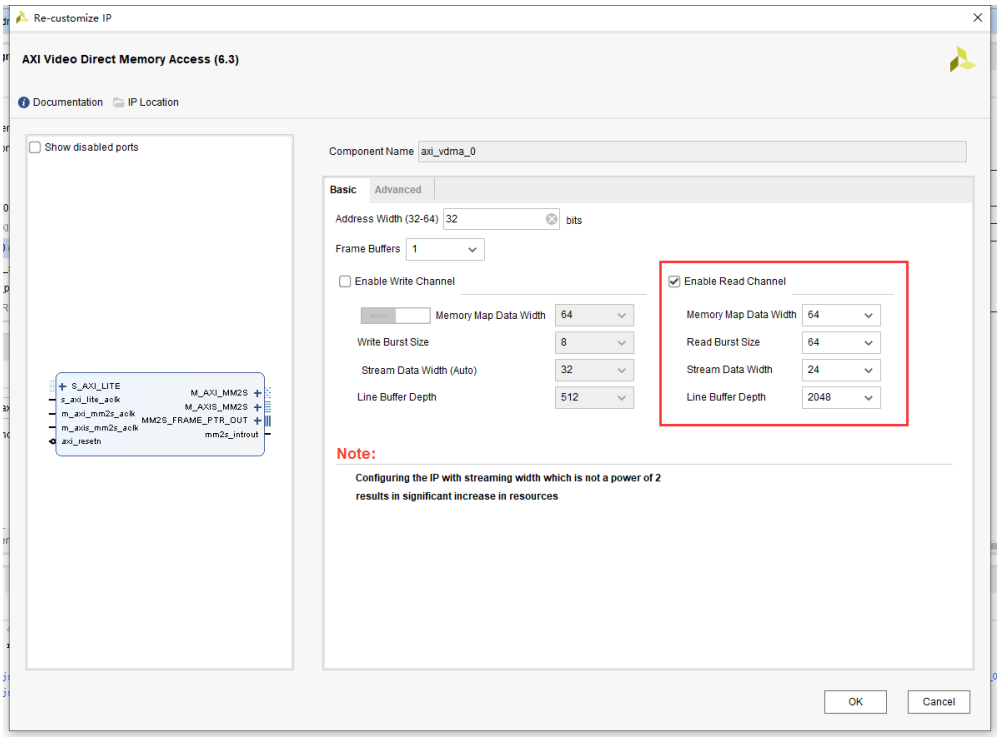
之后删除如下图所示的三个 IP 及输出信号，



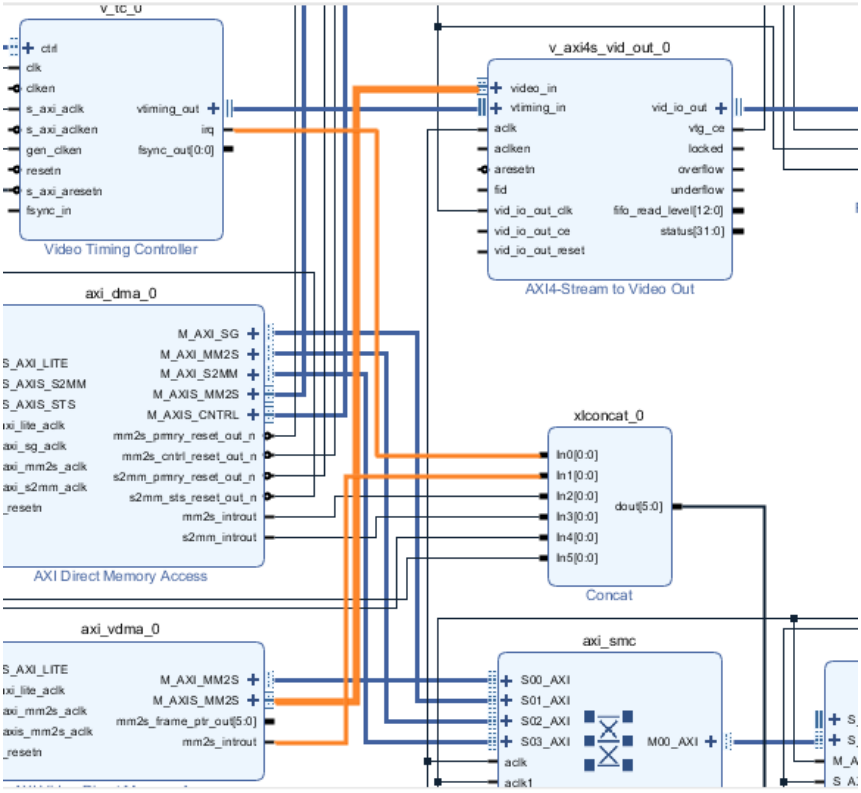
双击 VDMA 打开设置页面，如下图所示修改相关设置，



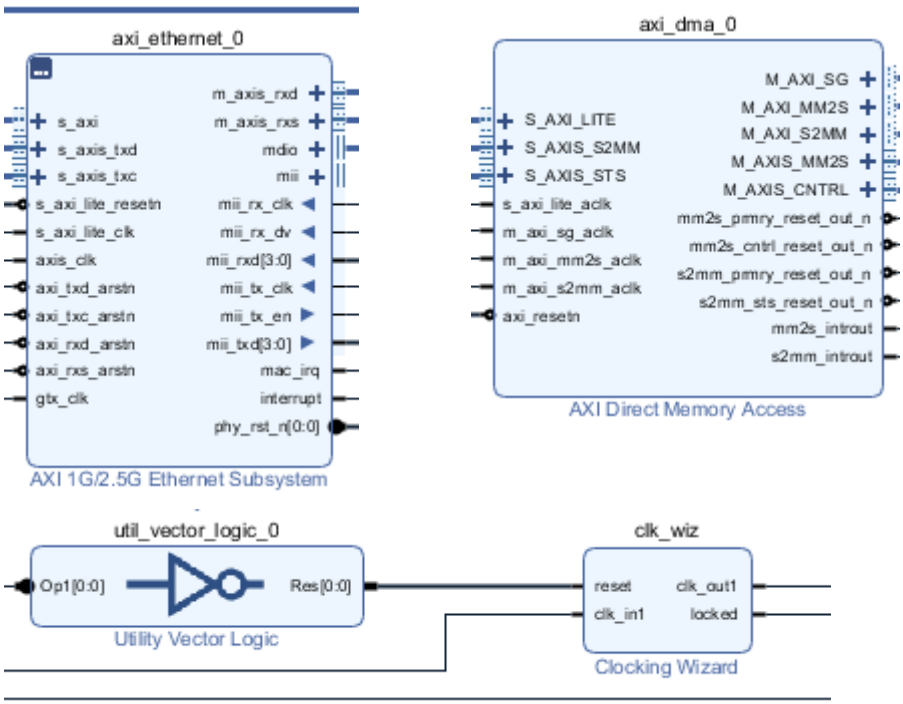
修改 Concat IP 的端口数，修改为 6，在之后的操作中将添加网络相关 IP，增加中断信号数量。



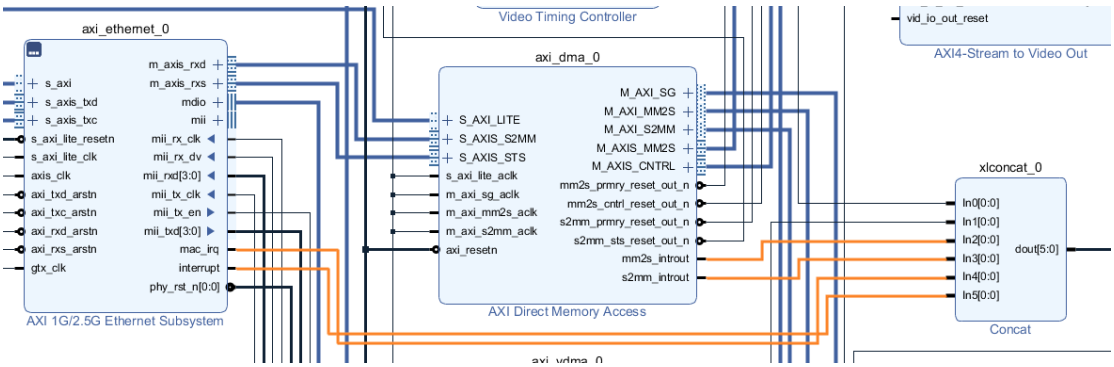
重新连接中断信号及 VDMA 到 Video out IP 的数据总线，如下图所示：



之后添加网络相关 IP，如下图所示。



各个 IP 设置及信号连接与实验 1 完全相同，参照实验 1 进行设置，连接时钟 IP 相关信号，引出 MII 接口相关信号。设置完成后连接中断信号，如下图所示：



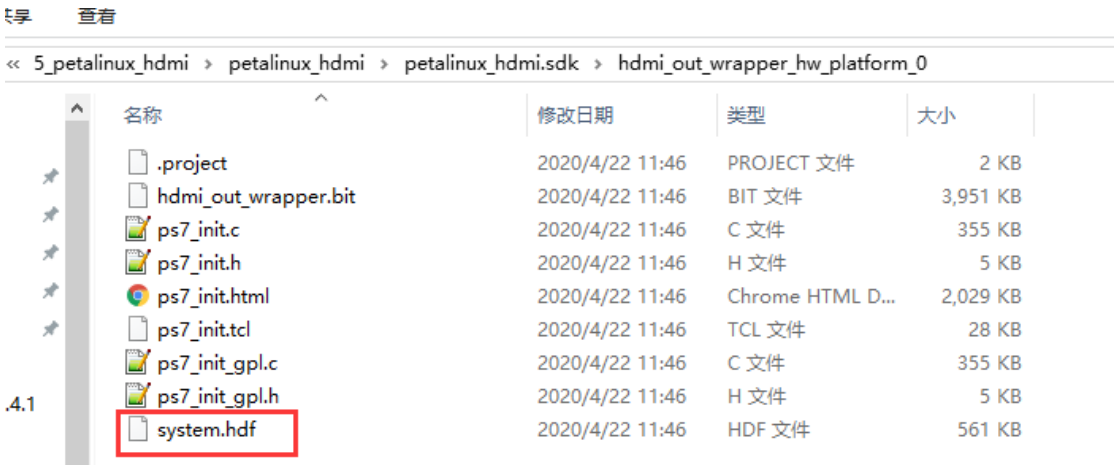
硬件设计的修改就完成了，保存并重新生成 HDL。

之后打开引脚约束文件 hdmi_out.xdc，重新对引脚进行约束，如下图所示：

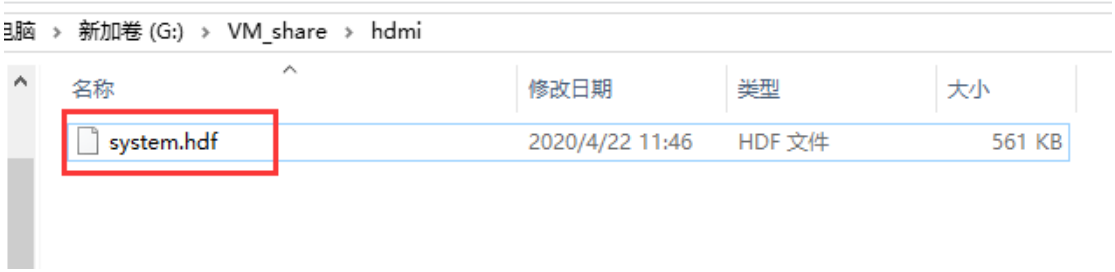
```
set_property PACKAGE_PIN U18 [get_ports TMDS_0_clk_p]
set_property PACKAGE_PIN V20 [get_ports {TMDS_0_data_p[0]}]
set_property PACKAGE_PIN T20 [get_ports {TMDS_0_data_p[1]}]
set_property PACKAGE_PIN N20 [get_ports {TMDS_0_data_p[2]}]
```

```
6  set_property IOSTANDARD LVCMOS33 [get_ports {reset_rtl_0[0]}]
7  set_property IOSTANDARD LVCMOS33 [get_ports sys_clk]
8  set_property IOSTANDARD LVCMOS33 [get_ports mdio_rtl_0_mdc]
9  set_property IOSTANDARD LVCMOS33 [get_ports mdio_rtl_0_mdio_io]
10
11 set_property IOSTANDARD LVCMOS33 [get_ports mii_tx_en_0]
12 set_property IOSTANDARD LVCMOS33 [get_ports {mii_txd_0[3]}]
13 set_property IOSTANDARD LVCMOS33 [get_ports {mii_txd_0[2]}]
14 set_property IOSTANDARD LVCMOS33 [get_ports {mii_txd_0[1]}]
15 set_property IOSTANDARD LVCMOS33 [get_ports {mii_txd_0[0]}]
16 set_property IOSTANDARD LVCMOS33 [get_ports {mii_rxd_0[3]}]
17 set_property IOSTANDARD LVCMOS33 [get_ports {mii_rxd_0[2]}]
18 set_property IOSTANDARD LVCMOS33 [get_ports {mii_rxd_0[1]}]
19 set_property IOSTANDARD LVCMOS33 [get_ports {mii_rxd_0[0]}]
20 set_property IOSTANDARD LVCMOS33 [get_ports mii_rx_dv_0]
21 set_property IOSTANDARD LVCMOS33 [get_ports mii_rx_clk_0]
22 set_property IOSTANDARD LVCMOS33 [get_ports mii_tx_clk_0]
23 set_property PACKAGE_PIN N16 [get_ports mii_tx_en_0]
24 set_property PACKAGE_PIN L14 [get_ports mii_tx_clk_0]
25 set_property PACKAGE_PIN K18 [get_ports mii_rx_dv_0]
26 set_property PACKAGE_PIN K17 [get_ports mii_rx_clk_0]
27 set_property PACKAGE_PIN J14 [get_ports {mii_rxd_0[0]}]
28 set_property PACKAGE_PIN K14 [get_ports {mii_rxd_0[1]}]
29 set_property PACKAGE_PIN M18 [get_ports {mii_rxd_0[2]}]
30 set_property PACKAGE_PIN M17 [get_ports {mii_rxd_0[3]}]
31 set_property PACKAGE_PIN M14 [get_ports {mii_txd_0[0]}]
32 set_property PACKAGE_PIN L15 [get_ports {mii_txd_0[1]}]
33 set_property PACKAGE_PIN M15 [get_ports {mii_txd_0[2]}]
34 set_property PACKAGE_PIN N15 [get_ports {mii_txd_0[3]}]
35 set_property PACKAGE_PIN G14 [get_ports mdio_rtl_0_mdc]
36 set_property PACKAGE_PIN J15 [get_ports mdio_rtl_0_mdio_io]
37 set_property PACKAGE_PIN H20 [get_ports {reset_rtl_0[0]}]
38 set_property PACKAGE_PIN N18 [get_ports sys_clk]
39
40
41 set_property SLEW FAST [get_ports mdio_rtl_0_mdc]
42 set_property SLEW FAST [get_ports mdio_rtl_0_mdio_io]
43 set_property SLEW SLOW [get_ports {reset_rtl_0[0]}]
44 set_property SLEW FAST [get_ports mii_tx_en_0]
45 set_property SLEW FAST [get_ports {mii_txd_0[3]}]
46 set_property SLEW FAST [get_ports {mii_txd_0[2]}]
47 set_property SLEW FAST [get_ports {mii_txd_0[1]}]
48 set_property SLEW FAST [get_ports {mii_txd_0[0]}]
```

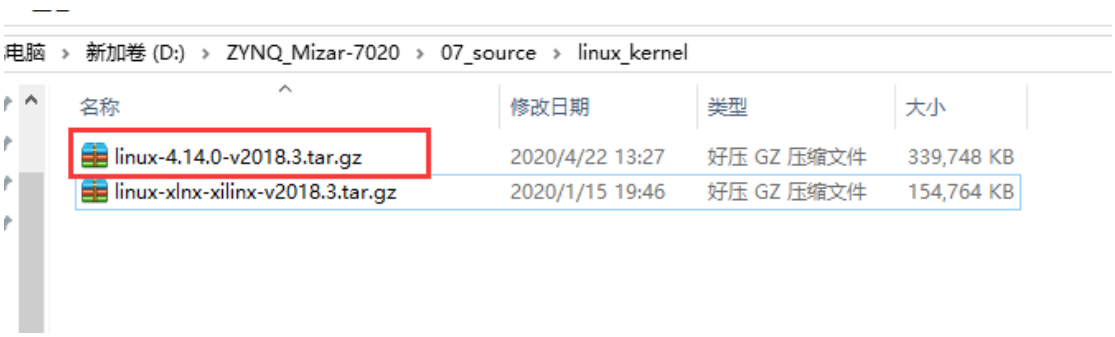
修改完成后保存重新生成 Bitstream，导出 bit 文件并打开 SDK，即可在 petalinux_hdmi.sdk->hdmi_out_wrapper_platform_0 文件夹内看到导出的硬件描述文件 system.hdf，



之后将该文件复制到共享文件夹下，如下图所示：



如下图所示为 petalinux2018.3 的原始内核以及添加了相关驱动的内核，将此内核压缩文件同样复制到共享文件夹，在之后的 petalinux 工程中使用该 Linux 内核。



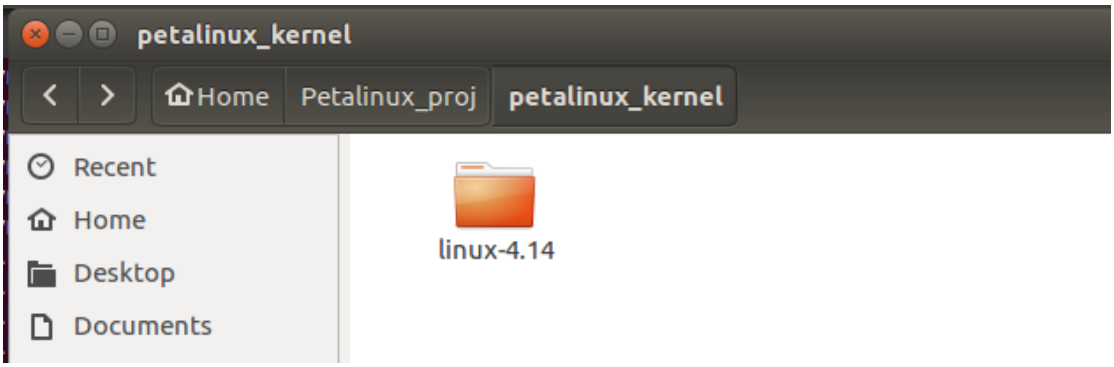
10.2 创建并配置 Petalinux 工程

打开 VirtualBox 虚拟机进入 Ubuntu Linux 操作系统，打开 Terminal 窗口，输入下列命令挂载 VM_share 到/mnt，

```
sudo mount –t vboxsf VM_share /mnt
```

```
microphase@microphase-VirtualBox:~$ sudo mount -t vboxsf VM_share /mnt
[sudo] password for microphase:
microphase@microphase-VirtualBox:~$
```

挂载完成之后在 Petalinux_proj 文件夹下创建内核文件夹 petalinux_kernel 之后在该文件夹下创建 linux-4.14 文件夹，之后将复制到共享文件夹下的内核文件解压到该文件夹下，解压完成后如下图所示：



输入下列命令，设置 Petalinux 运行所需的环境变量，

```
source /opt/pkg/petalimix/2018.3/setting.sh
```

```
microphase@microphase-VirtualBox:~/Petalinux_proj$ source /opt/pkg/petalinux/2018.3/settings.sh
PetaLinux environment set to '/opt/pkg/petalinux/2018.3'
INFO: Checking free disk space
INFO: Checking installed tools
INFO: Checking installed development libraries
INFO: Checking network and other services
microphase@microphase-VirtualBox:~/Petalinux_proj$
```

运行下列命令创建新的 petalinux 工程并命名为“petalinux_hdmi”，

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

创建完成之后进入工程文件夹，之后将复制到共享文件夹的 system.hdf 文件复制到工程文件夹下，

```
cd petalinux_hdmi/
cp /mnt/hdmi/system/hdf .
```

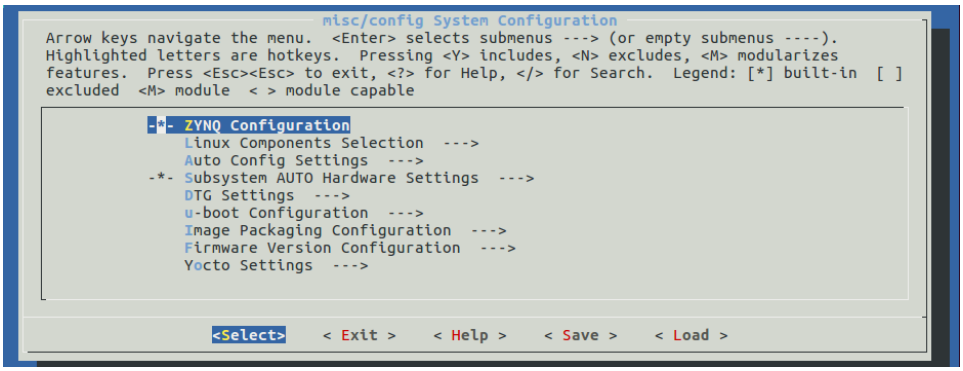
```
microphase@microphase-VirtualBox:~/Petalinux_proj$ cd petalinux_hdmi/
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_hdmi$ cp /mnt/hdmi/system.hdf .
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_hdmi$ ls
config.project  project-spec  system.hdf
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_hdmi$
```

之后运行下列命令将硬件描述文件导入工程中，

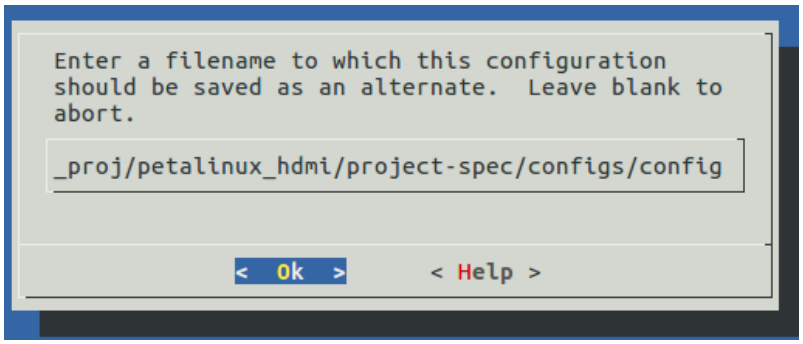
```
petalinux-config --get-hw-description=.
```

```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_hdmi$ petalinux-config --get-hw-description=.
INFO: Getting hardware description...
[INFO] generating Kconfig for project
[INFO] menuconfig project


*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.
```

一段时间后打开如下图所示硬件设置菜单，该实验相关设置按照实验 1 修改 Ethernet 相关设置即可，之后保存退出，



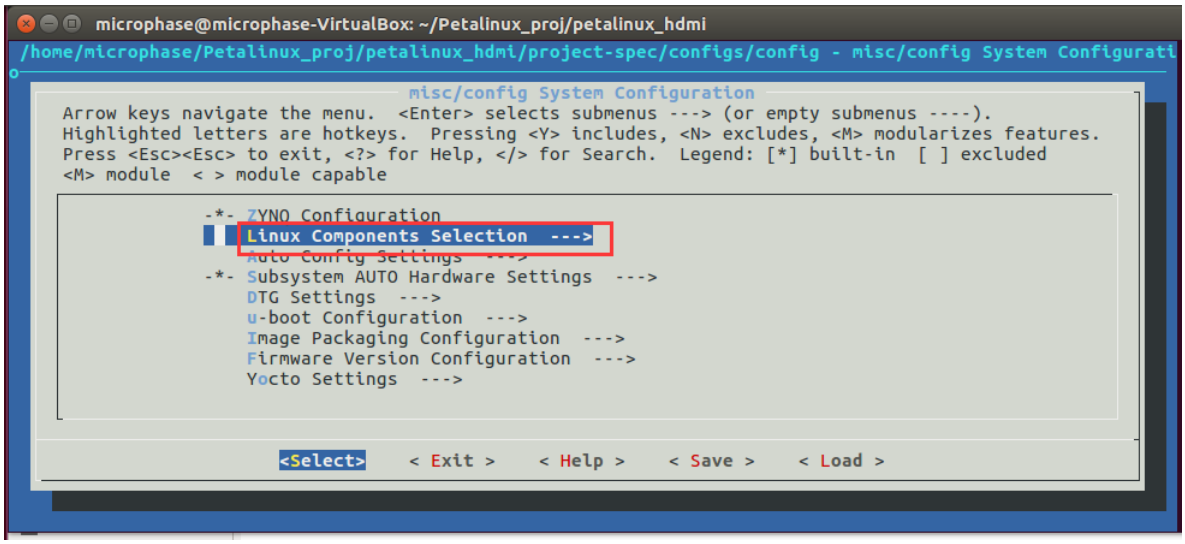
在弹出的窗口选择 OK，



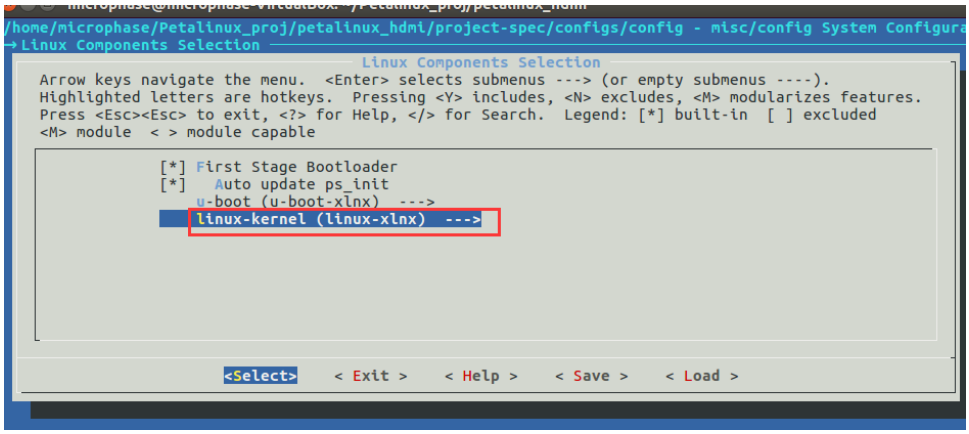
连续选择 Exit 退出设置菜单，

输入下列命令打开工程设置菜单，设置 Linux 内核来源，在打开的菜单中选择 Linux Components Selection，如下图所示：

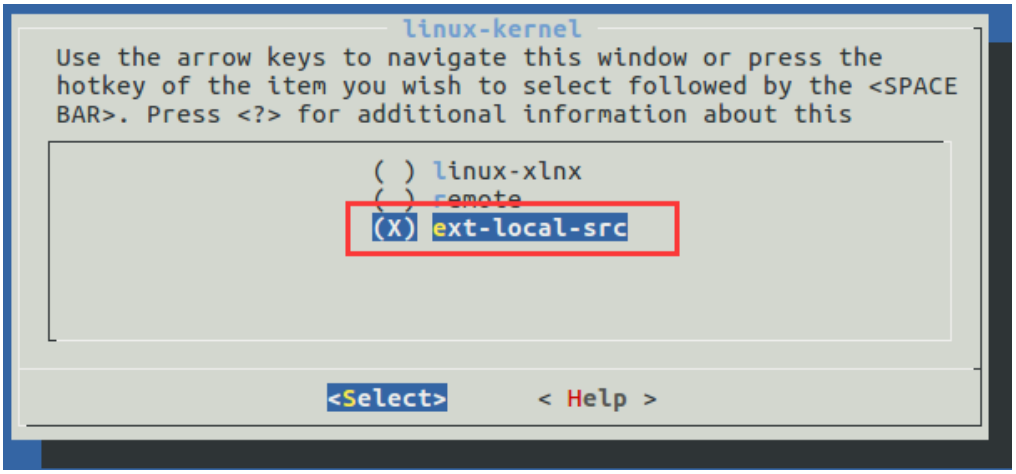
```
petalinux-config
```



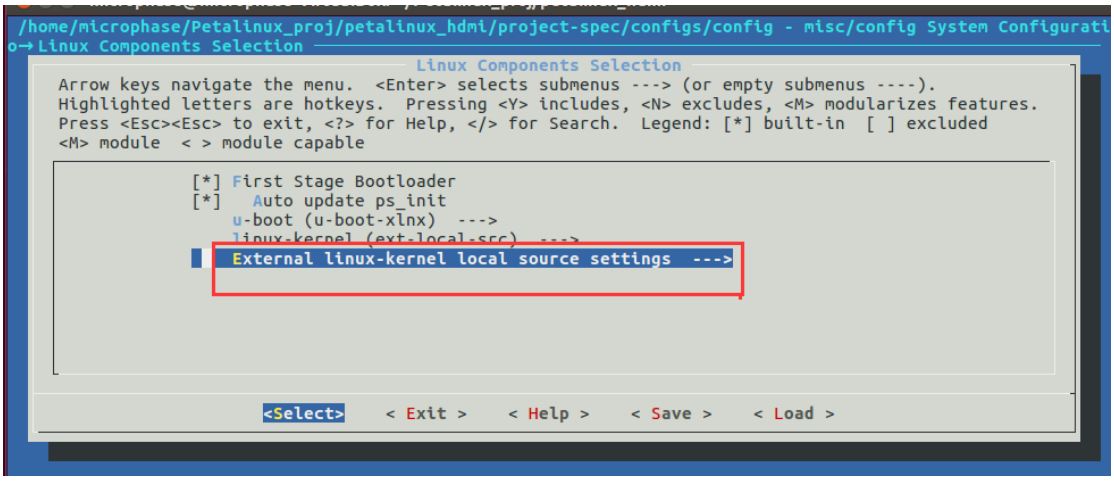
之后进入 linux-kernel(linux-xlnx)设置选项，如下图所示：



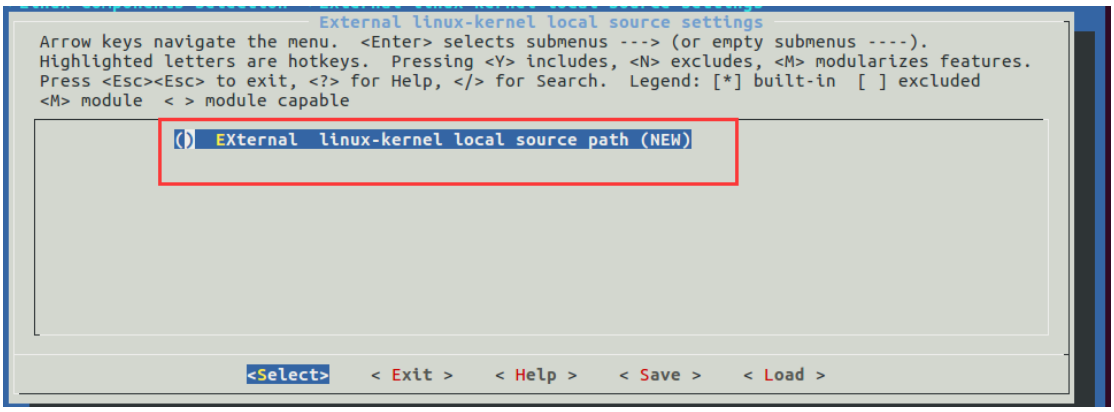
此处选择“ext-local-src”，也就是本地存放的 Linux 内核源码，如下图所示：



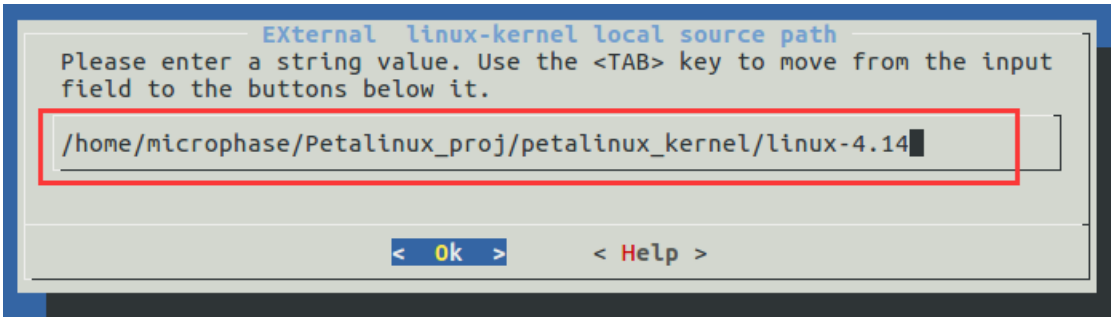
将光标移动到“ext-local-src”，然后按回车返回上一界面，



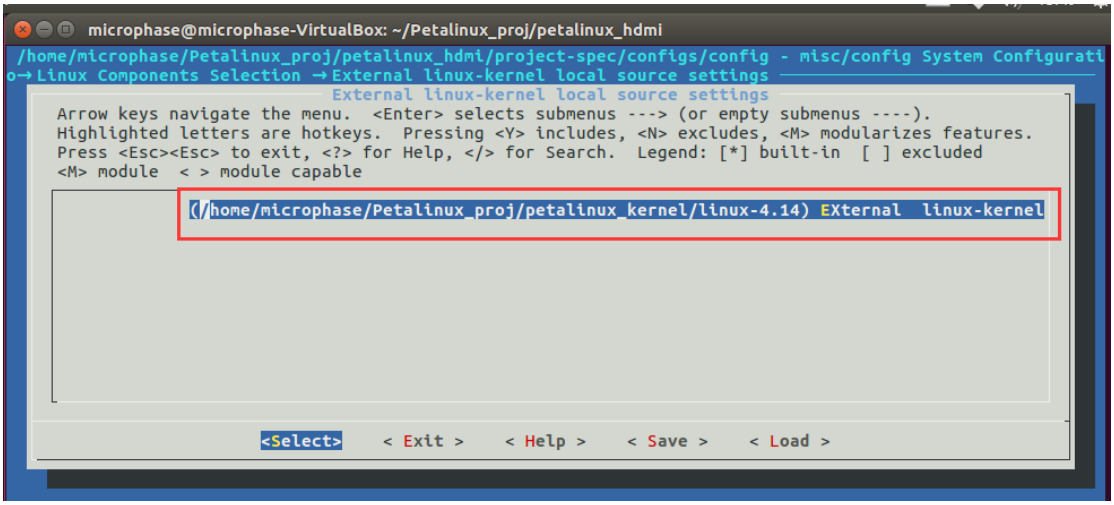
进入“External linux-kernel local source settings”子菜单， 如下图所示：



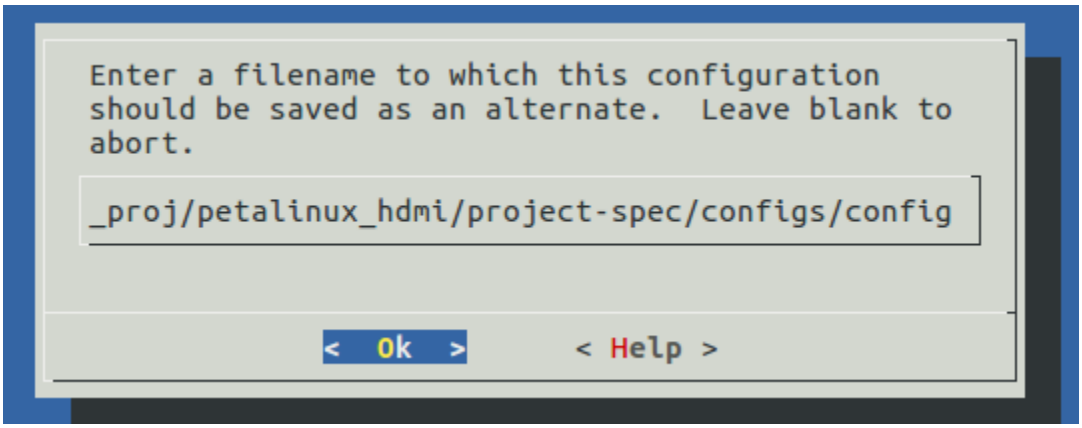
按回车键配置“External linux-kernel local source path”，如下图所示，在此处填写之前解压的得到的内核源码绝对路径，之后按回车保存配置并返回上一界面，



显示之前输入的内核路径，之后保存设置，



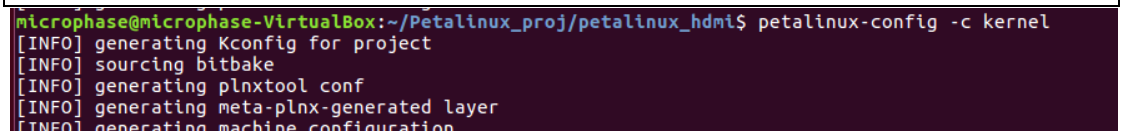
在弹出的窗口选择 OK，之后退出设置菜单。



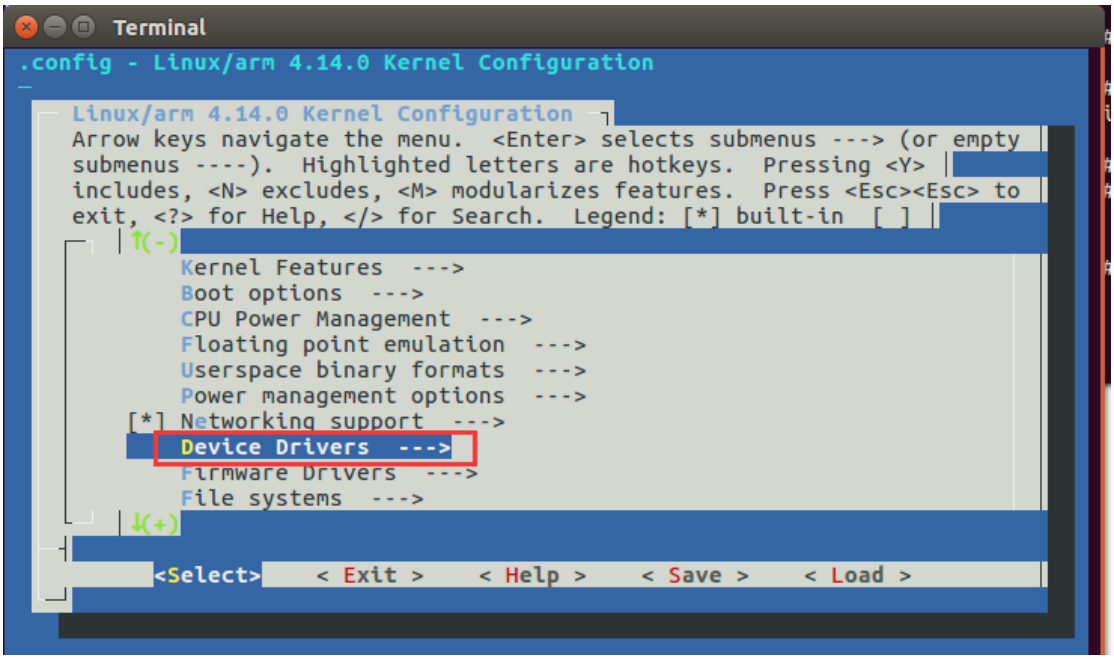
10.3 配置 Linux 内核

输入下列命令进入内核设置菜单，

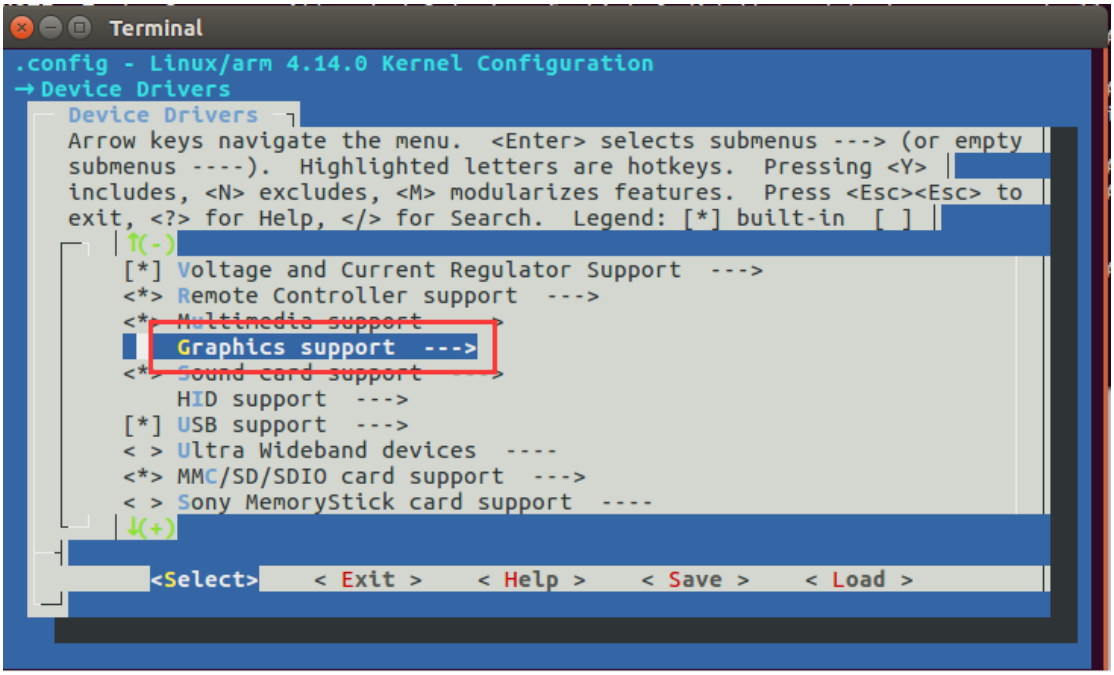
```
petalinux-config -c kernel
```



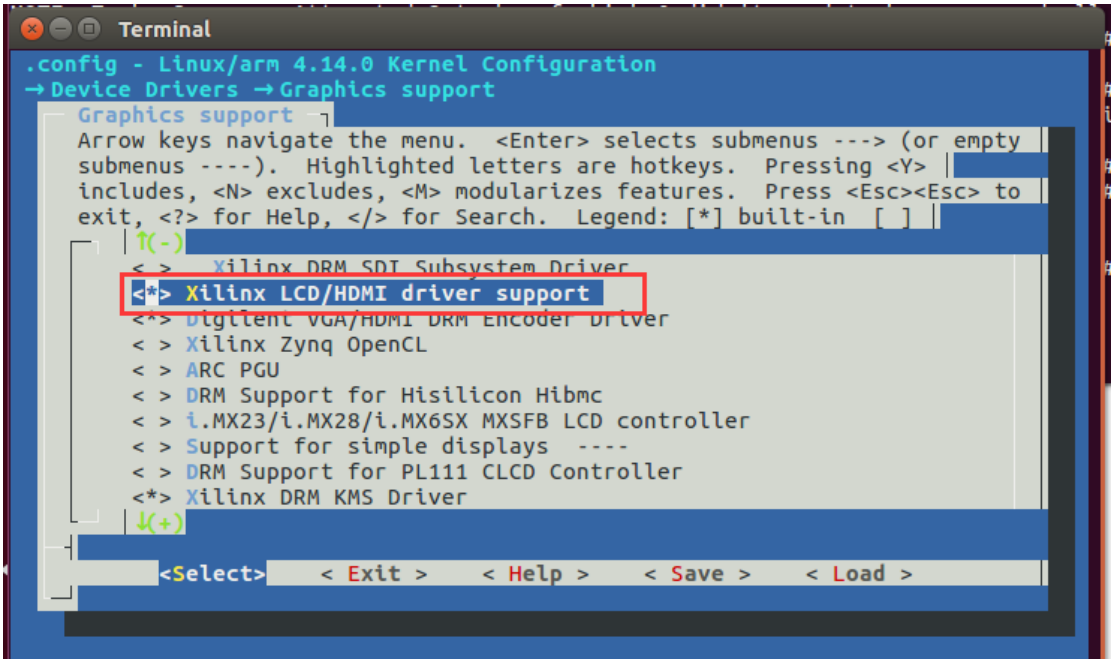
进入“Device Drivers”菜单，



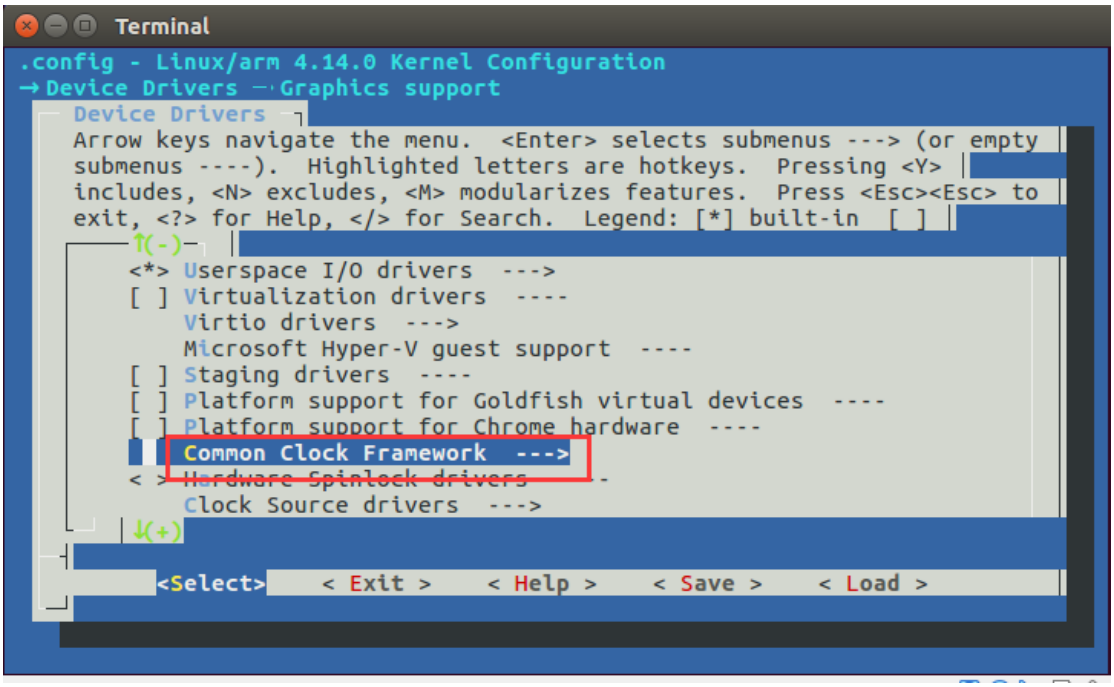
进入“Graphics support”子菜单，



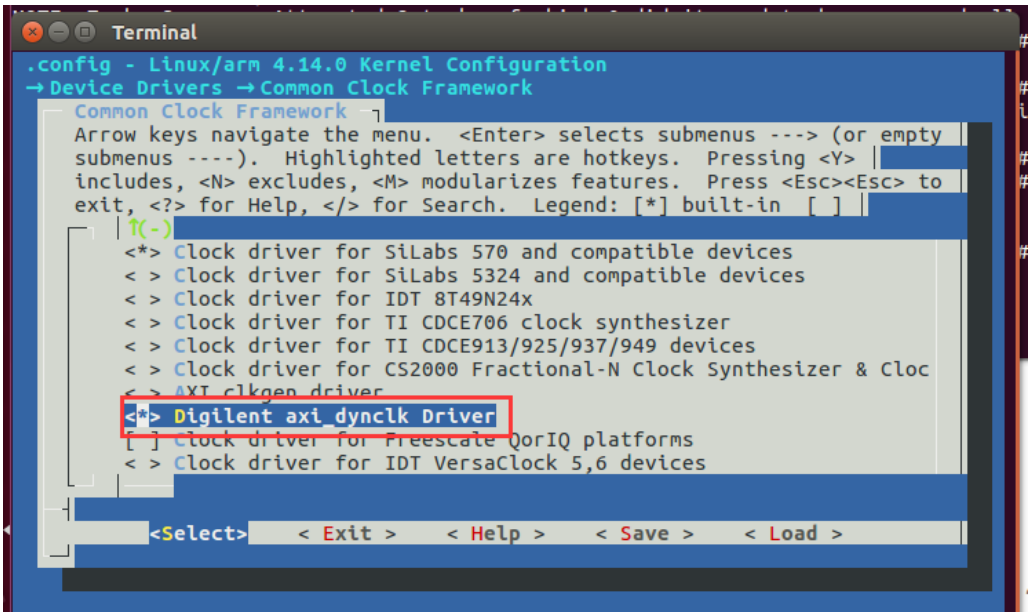
可以看到“Xilinx LCD/HDMI driver support”选项已配置，该驱动用来驱动 HDMI 和 LCD 的显示。



返回上级菜单，进入“Common Clock Framework”子菜单，



可以看到“Digilent axi_dynclk Driver”选项已配置，该驱动根据不同的分辨率输出不同的像素时钟。



10.4 配置设备树

编辑当前工程目录下的 project-spec/meta-user/recipes-bsp/device-tree/files/system-user.dtsi 文件，内容如下：

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


/ {
    model = "Navigator Development Board";
    compatible = "microphase,zynq-7020","xlnx,zynq-7000";


        usb_phy0:usb_phy@0{
                compatible = "ulpi-phy";
                #phy-cells = <0>;
                reg = <0xe0002000 0x1000>;
                view-port = <0x170>;
                drv-vbus;
        };

    video_timings {
            timing_4x3_480x272: timing0 {
                clock-frequency = <9000000>;
                hactive = <480>;
                vactive = <272>;

                hback-porch = <40>;
                hsync-len = <20>;
                hfront-porch = <5>;
                vback-porch = <8>;
                vsync-len = <3>;
                vfront-porch = <8>;

                hsync-active = <0>;
                vsync-active = <0>;
                de-active = <1>;
                pixelclk-active = <0>;
            };
```

```
                timing_1920x1080: timing1 {
                        clock-frequency = <148500000>;
                        hactive = <1920>;
                        vactive = <1080>;

                        hback-porch = <148>;
                        hsync-len = <44>;
                        hfront-porch = <88>;
                        vback-porch = <36>;
                        vsync-len = <5>;
                        vfront-porch = <4>;

                        hsync-active = <0>;
                        vsync-active = <0>;
                        de-active = <1>;
                        pixelclk-active = <1>;
                };
        };
};
```

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

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

        mdio{
                #address-cells = <1>;
                #size-cells = <0>;
                phy1:phy@1{
                        compatible = "realtek,rtl8201","ethernet-phy-id001c.c816";
                        device_type = "ethernet-phy";
                        reg = <0>;

                };
        };
};
```

```
&axi_dynclk_0 {
    compatible = "digilent,axi-dynclk";
    clocks = <&clkc 15>;
    #clock-cells = <0>;
};

&v_tc_0 {
    compatible = "xlnx,v-tc-5.01.a";
};
```

```
&amba_pl {
    xlnx_vdma_hdmi {
        compatible = "xilinx,vdmafb";
        status = "okay";

        xlnx,vtc = <&v_tc_0>;
        clocks = <&axi_dynclk_0>;
        clock-names = "hdmi_pclk";
        dmas = <&axi_vdma_0 0>;
        dma-names = "hdmi_vdma";

        is-hdmi = <0x1>;

        display-timings = <&timing_1920x1080>;
        xlnx,pixel-format = "bgr888";
    };
};
```

输入完成之后保存并关闭文件，之后就可以编译工程了。

10.5 编译工程及打包制作 BOOT.bin

输入下列命令编译工程，

```
petalinux-build
```

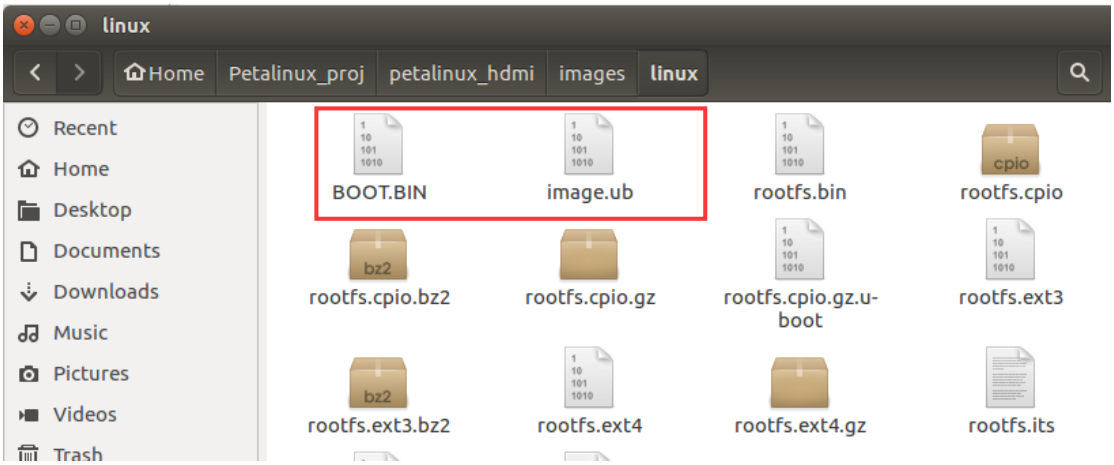
```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_hdmi$ petalinux-build
[INFO] building project
[INFO] sourcing bitbake
INFO: bitbake petalinux-user-image
Loading cache: 100% |#################################################################| Time: 0:00:00
Loaded 3443 entries from dependency cache.
Parsing recipes: 100% |################################################################| Time: 0:00:10
Parsing of 2569 .bb files complete (2535 cached, 34 parsed). 3445 targets, 149 skipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies
Initialising tasks: 100% |##############################################################| Time: 0:00:14
Checking sstate mirror object availability: 100% |#######################################| Time: 0:01:23
NOTE: Executing SetScene Tasks
NOTE: Executing RunQueue Tasks
NOTE: linux-xlnx: compiling from external source tree /home/microphase/Petalinux_proj/petalinux_kernel/linux
-4.14
WARNING: petalinux-user-image-1.0-r0 do_rootfs: [log_check] petalinux-user-image: found 1 warning message in
 the logfile:
[log_check] warning: %post(sysvinit-inittab-2.88dsf-r10.plnx_zynq7) scriptlet failed, exit status 1

NOTE: Tasks Summary: Attempted 3060 tasks of which 2246 didn't need to be rerun and all succeeded.
```

编译完成后将生成文件打包生成启动镜像文件，输入下列命令

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

```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_hdmi$ petalinux-package --boot --fsbl ./images/l
inux/zynq_fsbl.elf --fpga --u-boot --force
INFO: File in BOOT BIN: "/home/microphase/Petalinux_proj/petalinux_hdmi/images/linux/zynq_fsbl.elf"
INFO: File in BOOT BIN: "/home/microphase/Petalinux_proj/petalinux_hdmi/project-spec/hw-description/hdmi_out
_wrapper.bit"
INFO: File in BOOT BIN: "/home/microphase/Petalinux_proj/petalinux_hdmi/images/linux/u-boot.elf"
INFO: Generating Zynq binary package BOOT.BIN...


****** Xilinx Bootgen v2018.3
  **** Build date : Nov 15 2018-19:22:29
    ** Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.

INFO: Binary is ready.
WARNING: Unable to access the TFTPBOOT folder /tftpboot!!!
WARNING: Skip file copy to TFTPBOOT folder!!!
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_hdmi$
```

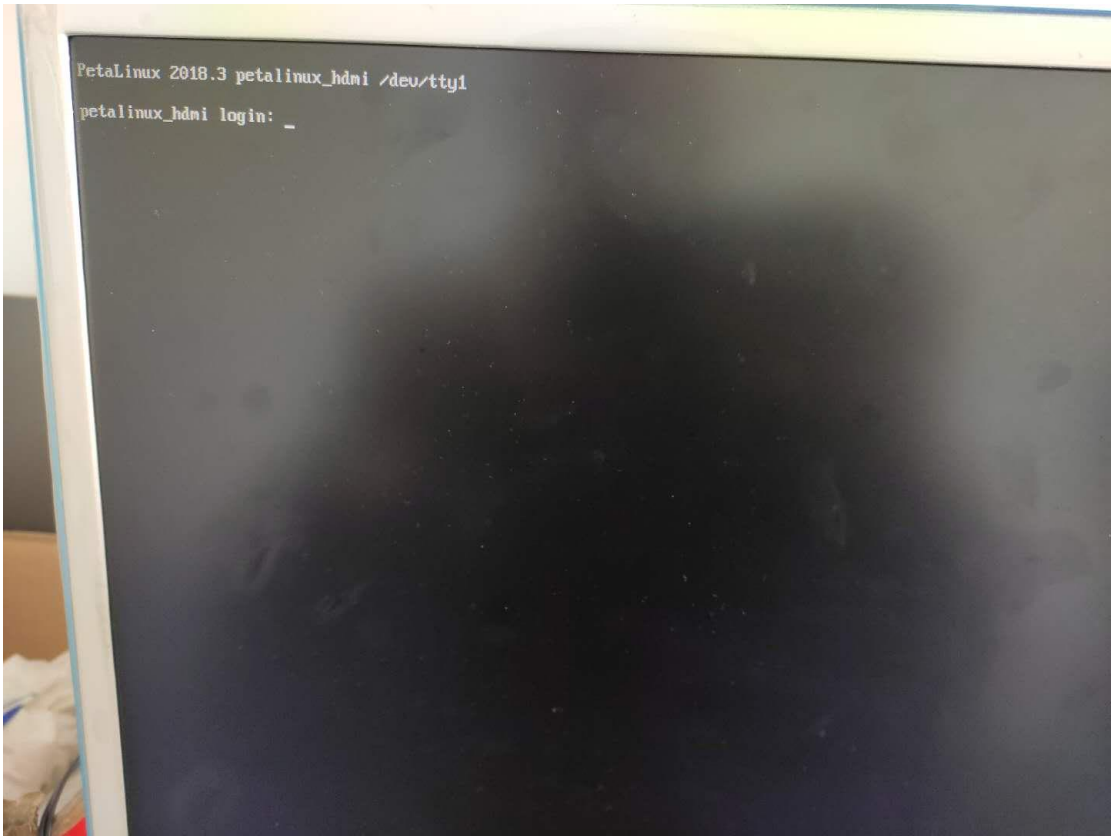
执行完成之后即可在工程目录 images/linux 路径下看到生成的 BOOT.bin 和 image.ub 文件，如下图所示：



10.6 系统启动验证

将编译打包生成的 BOOT.bin 和 image.ub 文件复制到 SD 卡中，插入开发板上的卡槽，调整跳线帽设置开发板启动方式为 SD 卡启动。连接开发板电源，串口 usb 线。打开 SecureCRT 软件连接串口，上电启动 Linux 操作系统。

使用 HDMI 连接线连接 HDMI 接口和显示屏，即可看到显示器显示内容如下图所示：



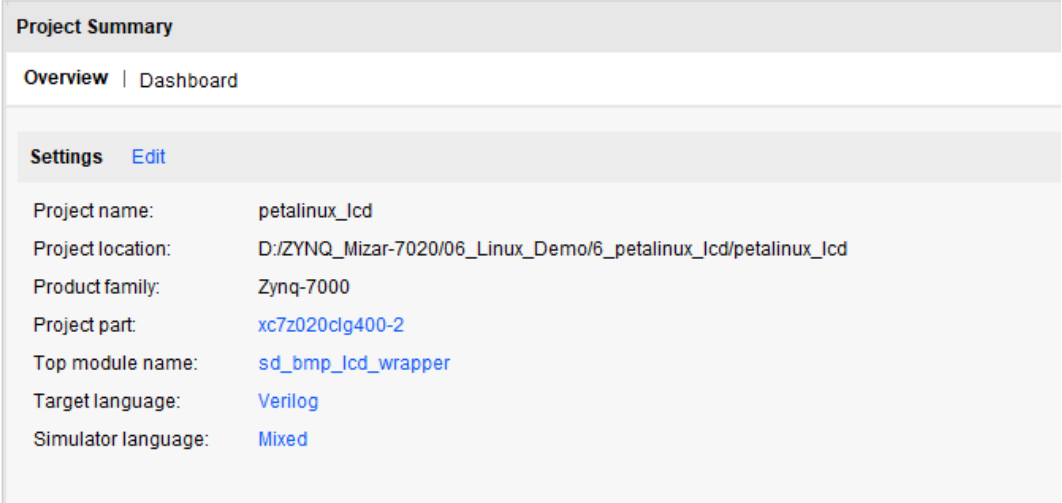
此时可以将键盘插入 USB 接口登录系统。

第11章　Linux 下 LCD 显示实验

在之前的实验中，我们实现了开发板运行 Linux 操作系统驱动 HDMI 显示的功能。但在实际应用中，受限于体积，便携性等因素，使用 HDMI 显示屏并不非常合适。所以就需要另一种使用更加方便的显示设备，那就是 LCD 显示屏。在本实验中，我们就利用我司出品的 LCD 显示屏来实现 Linux 操作系统下 LCD 屏幕的显示。

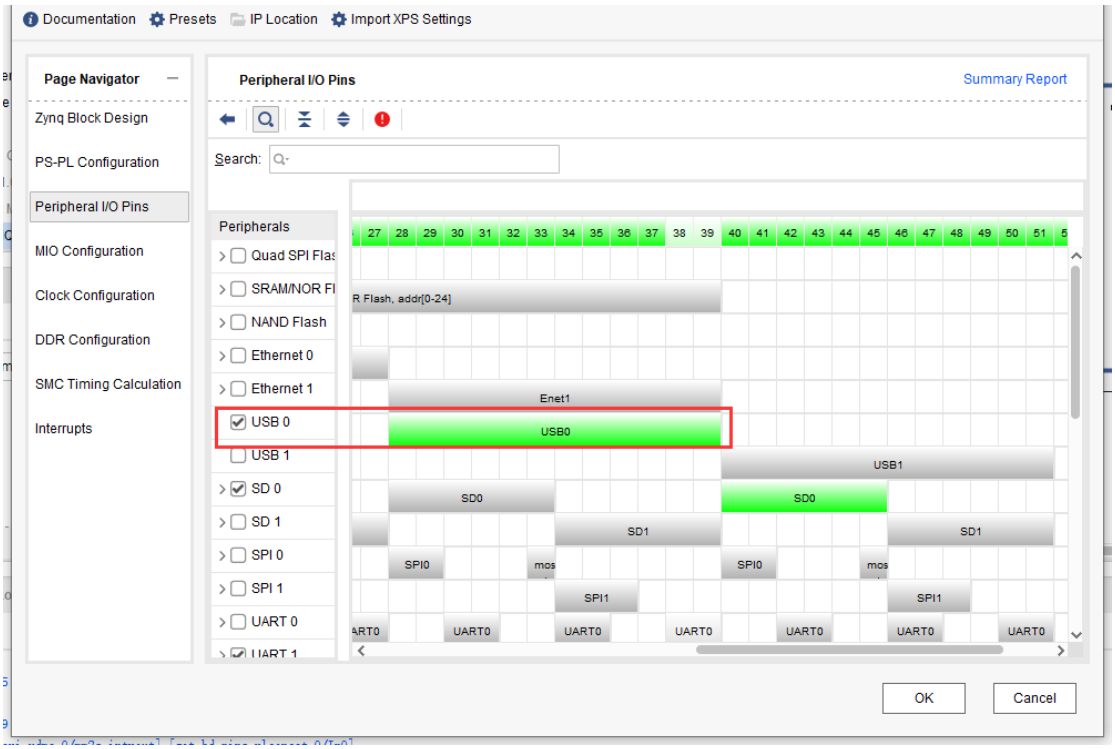
11.1 创建 Vivado 工程

本实验的硬件设计基于 SDK 实验 19，双击打开该实验 Vivado 工程，将其另存为一个新的工程并命名为“petalinux_lcd”，另存为完成后的工程如下图所示：

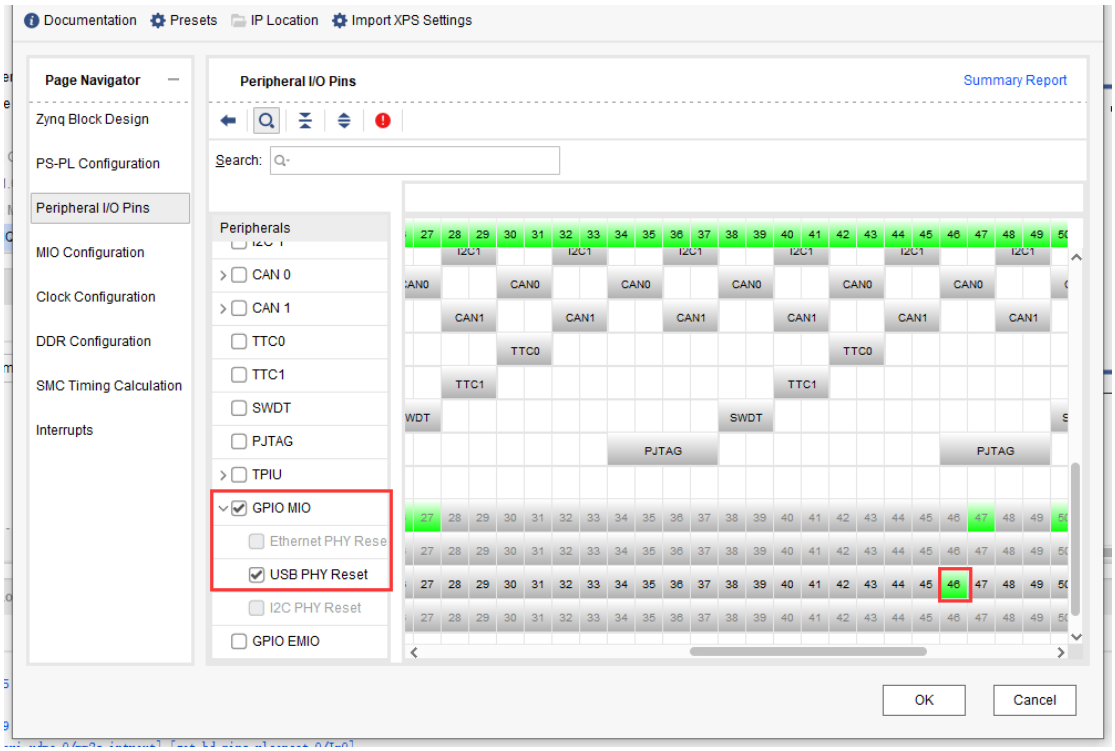


之后删除如下图所示工程目录下的.sdk 文件夹，这是之前实验生成的，本实验不需要，

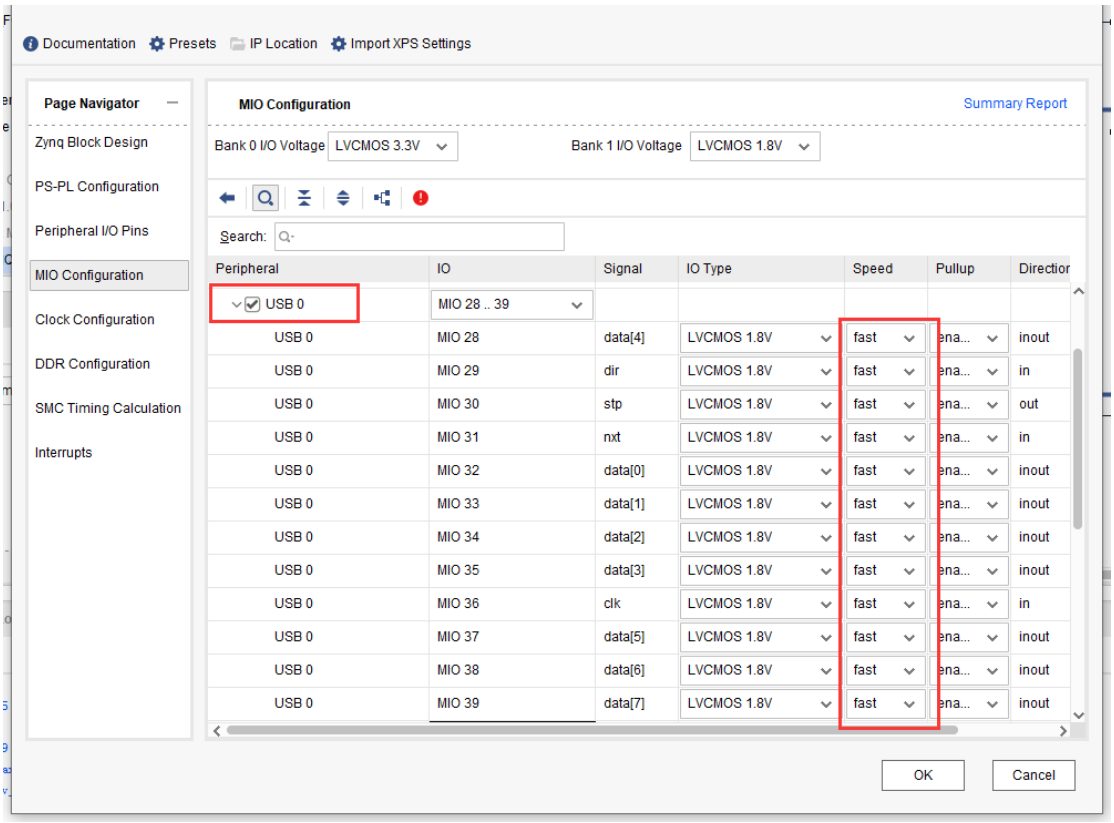
点击 Open Block Design 打开设计窗口，双击 ZYNQ7 Processing System 打开设置窗口，使能 USB0，对应 MIO28~MIO39，如下图所示：



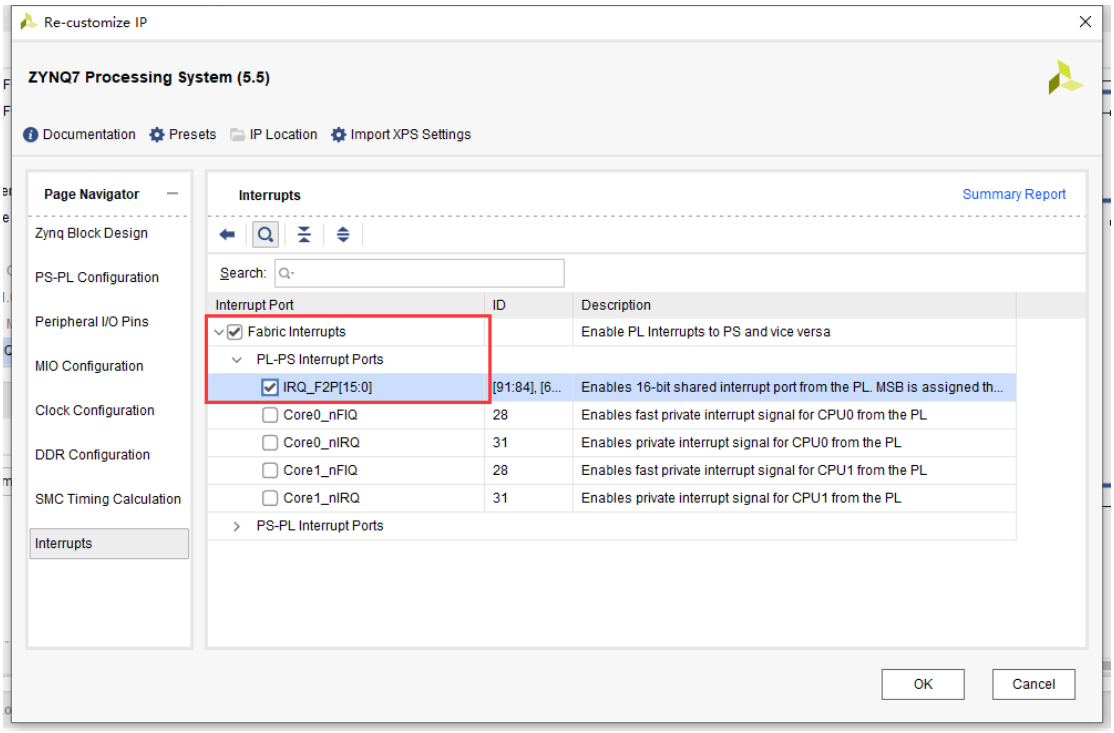
使能 GPIO MIO，设置 MIO46 为 USB PHY Reset，如下图所示：



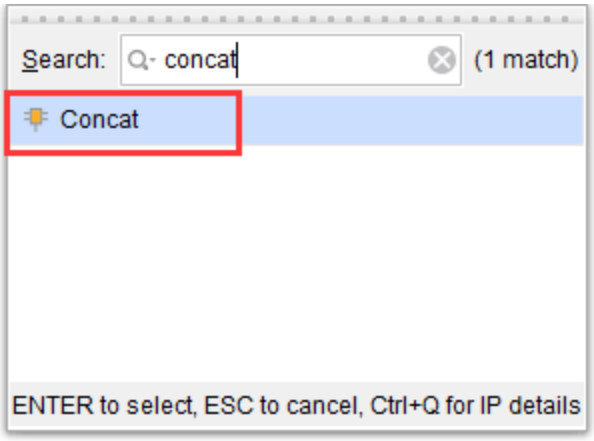
点击 MIO Configurastion，设置 USB0 的引脚速度为 fast，



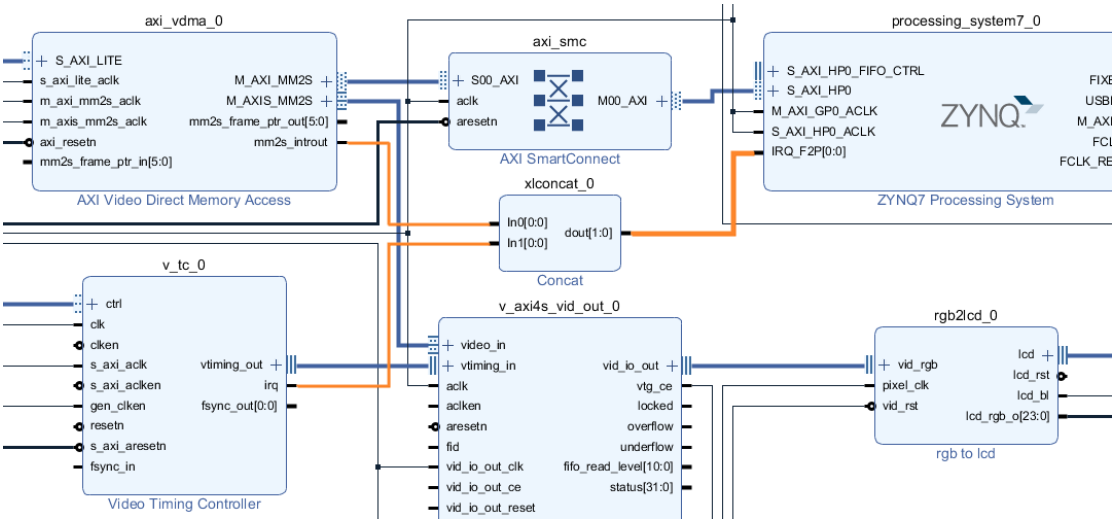
点击 Interrupt，使能 PL 到 PS 端的中断输入端口，



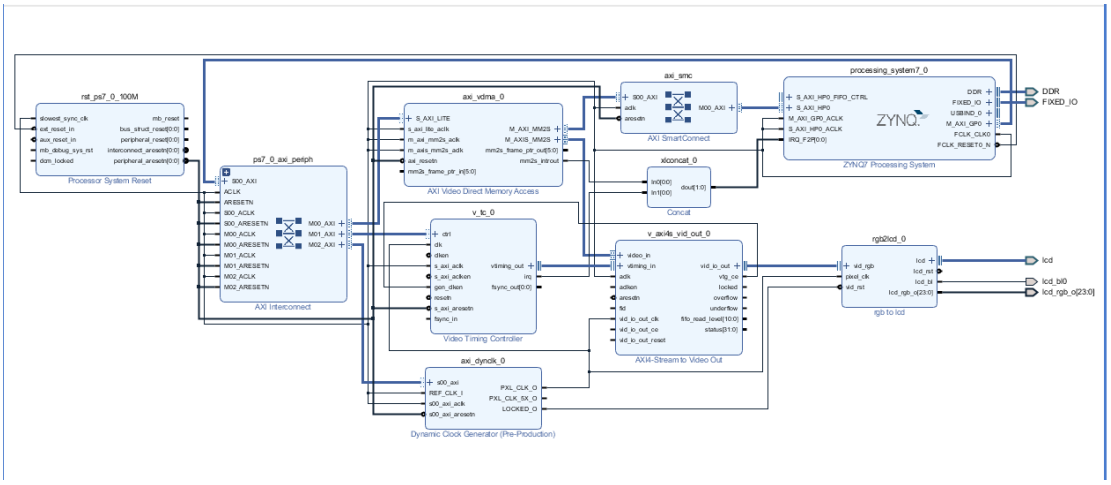
点击“+”打开 IP 核列表，输入 concat 搜索添加 Concat IP 核，



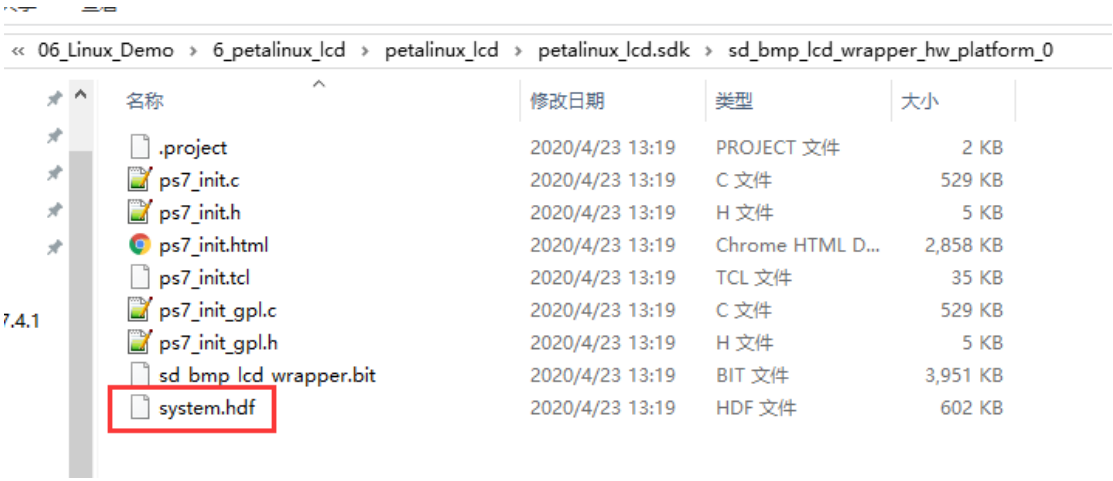
连接 VDMA 的 mm2s_introut 中断信号和 Video Timing Controller 的 irq 中断信号到 concat IP 的输入端，连接 Concat 的输出端到 ZYNQ7 Processing System 的中断信号输入端，如下图所示：



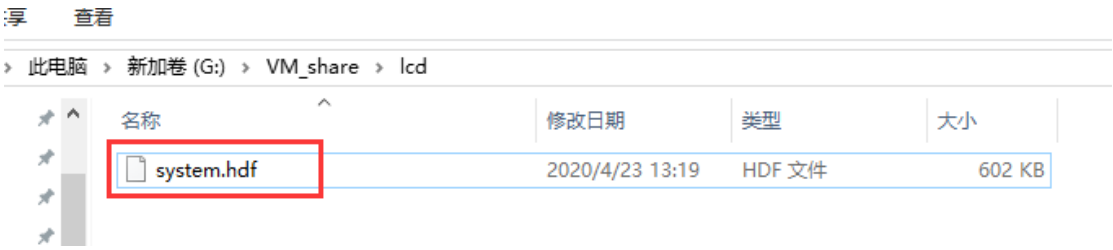
修改完成后硬件设计如下图所示：



本实验无需修改引脚约束，硬件设计修改完成之后，重新生成顶层 HDL 文件，点击左侧导航栏 Generate Bitstream 重新生成 bit 文件。生成完成之后导出到 SDK，之后点击 File->Launch SDK 打开 SDK 软件，即可在工程目录下 petalinux_lcd.sdk/sd_bmp_lcd_wrapper_hw_platform_0 路径下看到硬件描述文件 system.hdf，



在共享文件夹下创建 lcd 文件夹保存该实验相关文件，将 system.hdf 文件复制到该文件夹下，



11.2 创建并配置 Petalinxu 工程

打开 VirtualBox 虚拟机进入 Ubuntu Linux 操作系统，打开 Terminal 窗口，输入下列命令挂载 VM_share 到/mnt，设置 Petalinux 运行所需的环境变量，

```
sudo mount –t vboxsf VM_share /mnt
source /opt/pkg/petalinux/2018.3/setting.sh
```

```
microphase@microphase-VirtualBox:~$ sudo mount -t vboxsf VM_share /mnt/
[sudo] password for microphase:
microphase@microphase-VirtualBox:~$ source /opt/pkg/petalinux/2018.3/settings.sh

PetaLinux environment set to '/opt/pkg/petalinux/2018.3'
INFO: Checking free disk space
INFO: Checking installed tools
INFO: Checking installed development libraries
INFO: Checking network and other services
```

进入 Petalinux_proj 文件夹，输入下列命令创建本实验的 petalinux 工程并命名为“petalinux_lcd”，

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

```
microphase@microphase-VirtualBox:~/Petalinux_proj$ petalinux-create --type proje
ct --template zynq --name petalinux_lcd
INFO: Create project: petalinux_lcd
INFO: New project successfully created in /home/microphase/Petalinux_proj/petali
nux_lcd
```

进入新建的工程文件夹，输入下列命令将共享文件夹内的硬件描述文件复制到工程目录下

```
cd petalinux_lcd
cp /mnt/lcd/system.hdf .
```

```
microphase@microphase-VirtualBox:~/Petalinux_proj$ cd petalinux_lcd/
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_lcd$ ls
config.project  project-spec
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_lcd$ cp /mnt/lcd/sys
tem.hdf .
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_lcd$ ls
config.project  project-spec  system.hdf
```

输入下列命令将硬件描述文件导入工程，

```
petalinux_config –get-hw-description=.
```

```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_lcd$ petalinux-confi
g --get-hw-description=.
INFO: Getting hardware description...
[INFO] generating Kconfig for project
[INFO] menuconfig project


*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.
```

在弹出的菜单中直接选择 SAVE 保存并退出即可，无需修改设置。

运行下列命令打开工程配置窗口配置编译内核路径为本地路径，操作方式和 HDMI 显示实验相同，此处不再赘述，

```
petalinux-config
```

内核设置为与之前实验相同的本地路径之后即可修改设备树配置。

11.3 配置内核及设备树

运行下列命令进入内核设置菜单，本实验与之前实验内核设置相同，默认设置中已包含实验所需驱动，无需修改，

```
petalinux-config –c kernel
```

编辑当前工程目录下的 project-spec/meta-user/recipes-bsp/device-tree/files/system-user.dtsi 文件，添加如下内容：

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

#define GPIO_ACTIVE_HIGH 0
#define GPIO_ACTIVE_LOW  1

/ {
    model = "Mizar Development Board";
    compatible = "microphase,zynq-7020","xlnx,zynq-7000";

        usb_phy0:usb_phy@0{
                compatible = "ulpi-phy";
                #phy-cells = <0>;
                reg = <0xe0002000 0x1000>;
                view-port = <0x170>;
                drv-vbus;
        };

    video_timings {
            timing_4x3_480x272: timing0 {
                clock-frequency = <9000000>;
                hactive = <480>;
                vactive = <272>;

                hback-porch = <40>;
                hsync-len = <20>;
                hfront-porch = <5>;
                vback-porch = <8>;
                vsync-len = <3>;
                vfront-porch = <8>;

                hsync-active = <0>;
                vsync-active = <0>;
                de-active = <1>;
                pixelclk-active = <0>;
            };
```

```
            timing_1920x1080: timing1 {
                    clock-frequency = <148500000>;
                    hactive = <1920>;
                    vactive = <1080>;

                    hback-porch = <148>;
                    hsync-len = <44>;
                    hfront-porch = <88>;
                    vback-porch = <36>;
                    vsync-len = <5>;
                    vfront-porch = <4>;

                    hsync-active = <0>;
                    vsync-active = <0>;
                    de-active = <1>;
                    pixelclk-active = <1>;
            };
    };
};

&usb0{
        dr_mode = "host";
        usb-phy = <&usb_phy0>;
};
&axi_dynclk_0 {
    compatible = "digilent,axi-dynclk";
    clocks = <&clkc 15>;
    #clock-cells = <0>;
};

&v_tc_0 {
    compatible = "xlnx,v-tc-5.01.a";
};


&amba_pl {
    xlnx_vdma_lcd {
        compatible = "xilinx,vdmafb";
        status = "okay";

        xlnx,vtc = <&v_tc_0>;
        clocks = <&axi_dynclk_0>;
        clock-names = "lcd_pclk";
        dmas = <&axi_vdma_0 0>;
        dma-names = "lcd_vdma";

        is-hdmi = <0x0>;

        rst-gpios = <&gpio0 62 GPIO_ACTIVE_LOW>;
        bl-gpios = <&gpio0 61 GPIO_ACTIVE_HIGH>;

        display-timings = <&timing_4x3_480x272>;
        xlnx,pixel-format = "rgb888";
    };
};
```

修改完成之后保存修改内容。

11.4 编译工程及打包制作 BOOT.bin 文件

全部设置及设备树文件修改完成之后即可编译工程，输入下列命令编译工程，

```
petalinux-build
```

```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_lcd$ petalinux-build

[INFO] building project
[INFO] sourcing bitbake
INFO: bitbake petalinux-user-image
Loading cache: 100% |##############################################| Time: 0:00:01
Loaded 3443 entries from dependency cache.
Parsing recipes: 100% |############################################| Time: 0:00:11
Parsing of 2569 .bb files complete (2532 cached, 37 parsed). 3445 targets, 149 s
kipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies
Initialising tasks: 100% |##########################################| Time: 0:00:13
Checking sstate mirror object availability: 100% |################| Time: 0:00:22
NOTE: Executing SetScene Tasks
NOTE: Executing RunQueue Tasks
WARNING: petalinux-user-image-1.0-r0 do_rootfs: [log_check] petalinux-user-image
: found 1 warning message in the logfile:
[log_check] warning: %post(sysvinit-inittab-2.88dsf-r10.plnx_zynq7) scriptlet fa
iled, exit status 1
```

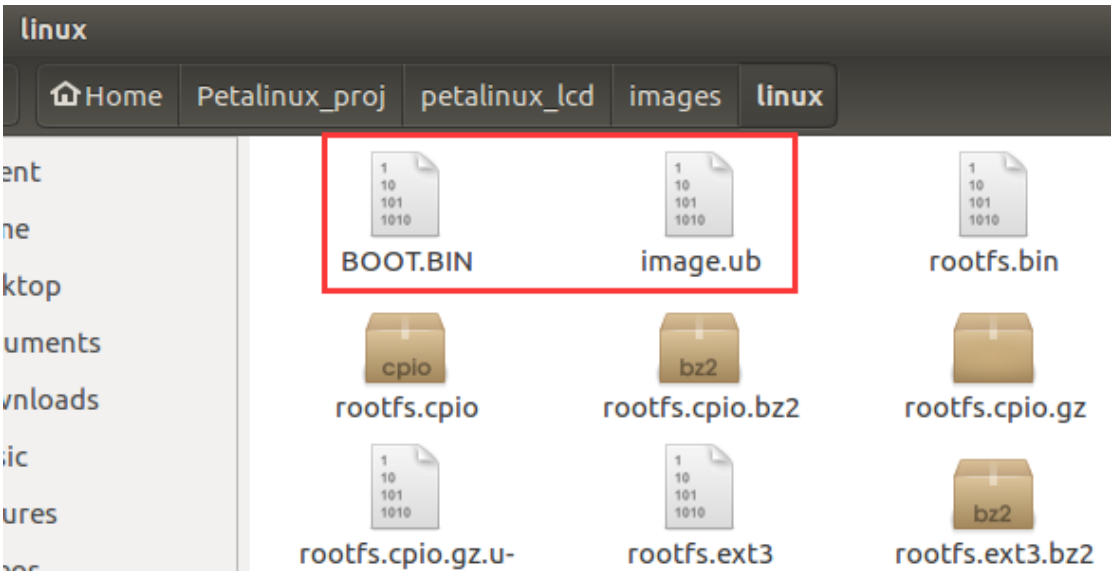
等待一段时间后编译完成，运行下列命令将运行文件打包生成 BOOT.bin 文件，

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

```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_lcd$ petalinux-packa
ge --boot --fsbl ./images/linux/zynq_fsbl.elf --fpga --u-boot --force
INFO: File in BOOT BIN: "/home/microphase/Petalinux_proj/petalinux_lcd/images/li
nux/zynq_fsbl.elf"
INFO: File in BOOT BIN: "/home/microphase/Petalinux_proj/petalinux_lcd/project-s
pec/hw-description/sd_bmp_lcd_wrapper.bit"
INFO: File in BOOT BIN: "/home/microphase/Petalinux_proj/petalinux_lcd/images/li
nux/u-boot.elf"
INFO: Generating Zynq binary package BOOT.BIN...


****** Xilinx Bootgen v2018.3
  **** Build date : Nov 15 2018-19:22:29
    ** Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.

INFO: Binary is ready.
WARNING: Unable to access the TFTPBOOT folder /tftpboot!!!
WARNING: Skip file copy to TFTPBOOT folder!!!
```

执行完成之后即可在工程目录下 images/linux 路径下看到生成的 BOOT.bin 文

件及 image.ub 文件，



11.5 系统启动验证

将编译打包生成的 BOOT.bin 和 image.ub 文件复制到 SD 卡中，插入开发板上的卡槽，调整跳线帽设置开发板启动方式为 SD 卡启动。连接串口 type-c 线。打开 SecureCRT 软件连接串口，将 LCD 屏幕插入开发板 GPIO1 插座，插上 JTAG 接口 type-c 线上电启动 Linux 操作系统。

系统启动后将键盘插头插入开发板，可以看到串口输出的信息显示已检测到键盘插入，

```
petalinux_lcd login: usb 1-1: new full-speed USB device number 2 using ci_hdrc
input: SINO WEALTH Keyboard as /devices/soc0/amba/e0002000.usb/ci_hdrc.0/usb1/1-1/1-1:1.0
/0003:258A:002A.0001/input/input0
hid-generic 0003:258A:002A.0001: input: USB HID v1.11 Keyboard [SINO WEALTH Keyboard] on
usb-ci_hdrc.0-1/input0
input: SINO WEALTH Keyboard as /devices/soc0/amba/e0002000.usb/ci_hdrc.0/usb1/1-1/1-1:1.1
/0003:258A:002A.0002/input/input1
hid-generic 0003:258A:002A.0002: input: USB HID v1.11 Keyboard [SINO WEALTH Keyboard] on
usb-ci_hdrc.0-1/input1
```

此时即可通过键盘控制系统运行，在 LCD 屏幕上显示键盘输入的命令及相应信息，

到此我们就实现了 Linux 系统驱动下的 LCD 显示功能。

第12章　开发板运行 Linux 图形界面

通常我们在开发板上运行 Linux 系统都是使用终端进行操作，也就是命令行的方式。那么 Linux 是否可以使用类似 Windows 的图形界面呢？答案显然是肯定的，例如我们现在使用的 Ubuntu 系统，就是默认使用的 GNOME 图形界面。不同于 Windows 这种将图形界面作为操作系统的一部分的方式，Linux 的图形界面更加类似于一个应用程序，当希望更换图形界面时，只需要卸载当前界面并安装新的界面即可，常用的 Linux 图形界面有 GNOME、KDE、Xfce、LXDE 等。而 Windows 图形界面一旦出错就意味着系统崩溃。本章我们来学习在 zynq 开发板上使用 Linux 图形界面。

12.1 简介

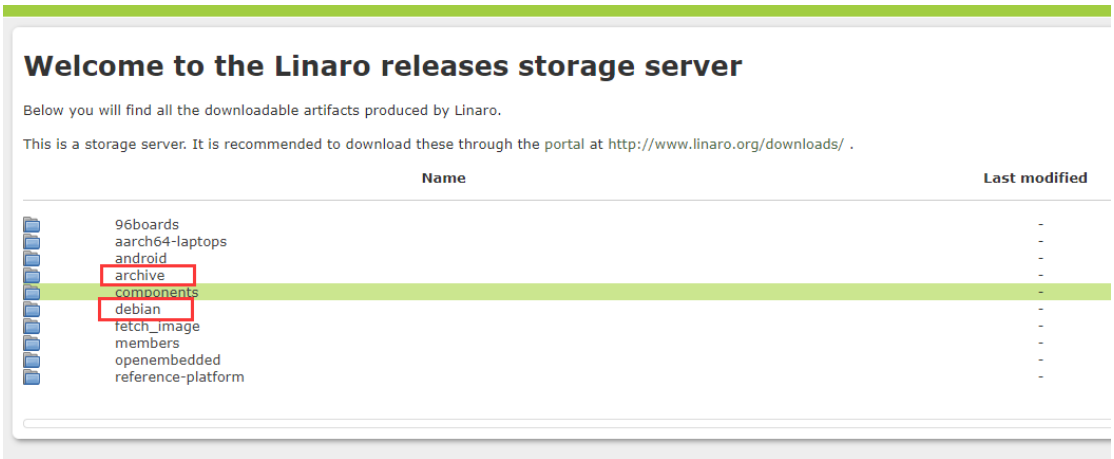
在当前，绝大多数的电脑用户使用的都是 Windows 或 iOS 这种图形界面，熟悉并享受图形界面带来的操作简便、内容丰富带来的娱乐效果。不同于这类定位于服务一般用户日常使用的桌面系统，Linux 更多的是面向软件开发，服务器使用及维护等领域，注重实际效果以及节约硬件资源，所以 Linux 的图形界面发展缓慢。而众多 Linux 的初学者刚开始面对命令行界面的时候也非常的不适应。为了促进 Linux 的发展，拓展它的使用人群，以及随着近年来 Linux 操作系统的发展，Linux 的图形界面已经取得了长足的进步，就像我们现在使用的 Ubuntu 就是一个具有成熟图形界面的 Linux 操作系统。

常见的 Linux 图形界面有多种，例如 GNOME、KDE、XFCE、LXDE 等，各种图形界面具有不同的定位，面对不同的应用场景，XFCE、LXDE 定位于轻量级的桌面，可用于硬件性能配置不高的电脑或嵌入式领域。例如不带有 GPU 的嵌入式开发板，因为图形界面对硬件性能还是有要求的。

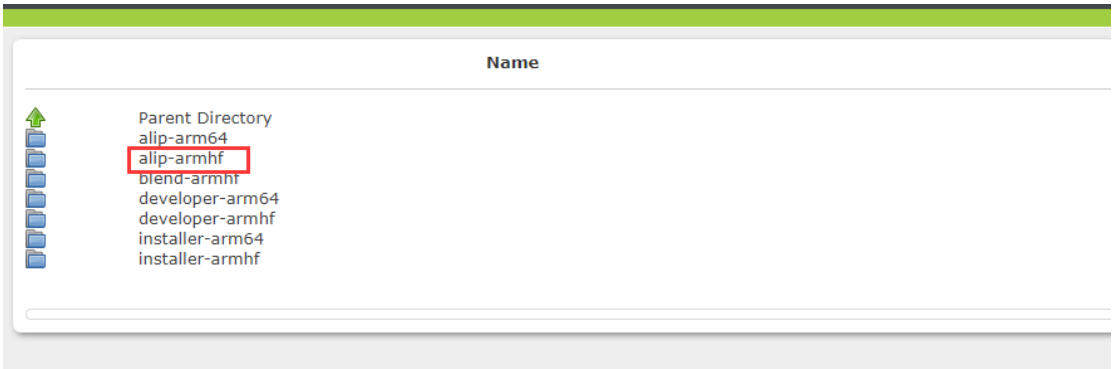
不同于 Windows 等桌面操作系统，图形界面是操作系统的一部分，Linux 的图形界面更加类似于一个应用程序，可以切换移植不同的图形界面。就是将图形

界面移植入根文件系统，在系统启动时加载即可。本章不讲解具体的移植方法，直接使用移植好的根文件系统。所以我们在这介绍一家公司—Linaro，简介如下：

Linaro ，一间非营利性质的开放源代码软件工程公司，主要的目标在于开发不同半导体公司系统单芯片（SoC）平台的共通软件，以促进消费者及厂商的福祉。由 ARM、飞思卡尔、IBM、Samsung、ST-Ericsson 及德州仪器(TI)等半导体厂商联合，在 2010 年 3 月成立。针对于各个成员推出的 ARM 系统单芯片（SoC），它开发了 ARM 开发工具、Linux 内核以及 Linux 发行版（包括 Android 及 Ubuntu）的主要自动建构系统。一般我们使用的 ARM 编译工具链是由这家公司提供的。我们需要的 Linux 发行版的根文件系统该公司也已经做好了，提供在网站：https://releases.linaro.org/上，我们访问该网站，网站主界面如下图所示：

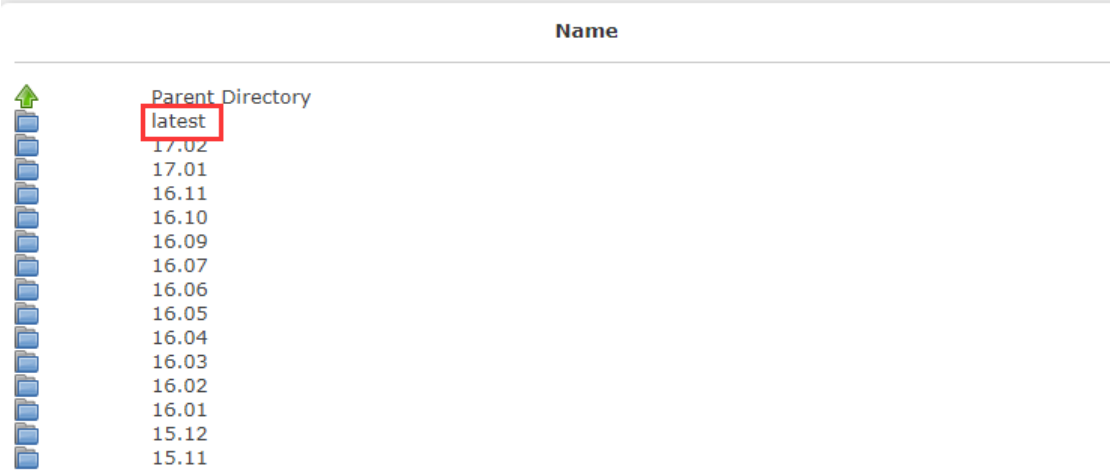


可以看到有“archive”和“debian”两个目录分别代表 Ubuntu 和 Debian 两种系统，打开 debian 目录。进入“debian”目录下的“images”，如下图所示：

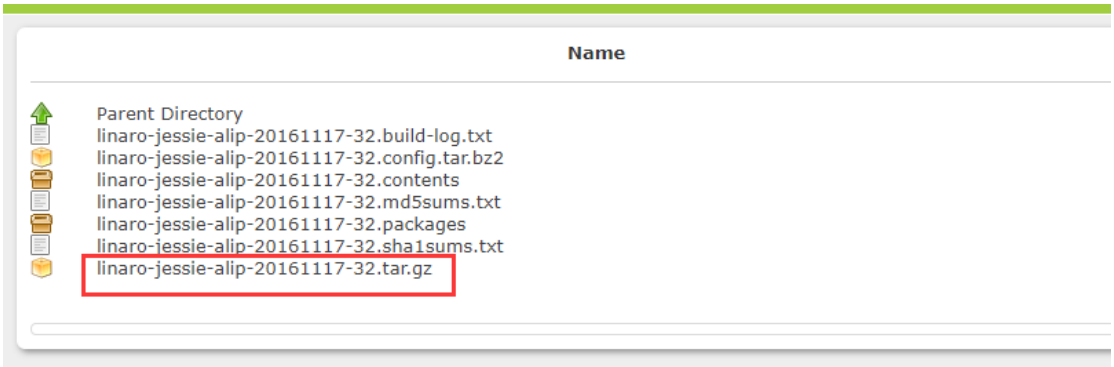


可以看到 Linaro 提供的 debian 发行版不同的根文件系统。其中名称带“arm64”

表明是用于 ARM 64 位的架构，“arm-hf”用于 ARM 32 位的架构。我们使用的 ZYNQ 是 ARM 32 位的，所以选择名称带“arm-hf”的都是可以的。由于这里我们使用 Linux 图形界面，所以选择 alip-armhf，如果使用不带图形界面的 debian 根文件系统，可以选择另外两个。进入 alip-armhf 目录下， 如下图所示：

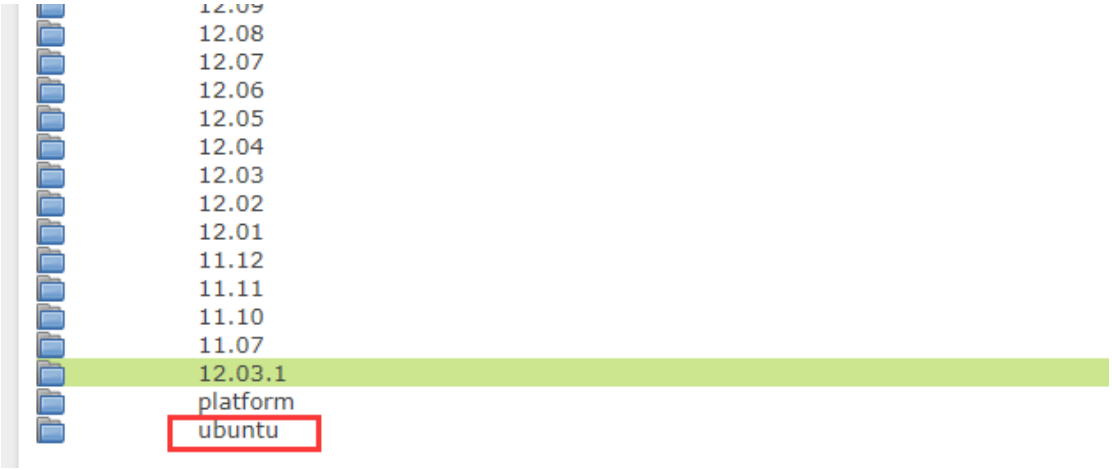


在该界面可以看到很多的目录，不同的数字代表过去不同的发布时间，我们进入“latest“目录下，也就是最近的发布版本，如下图所示：

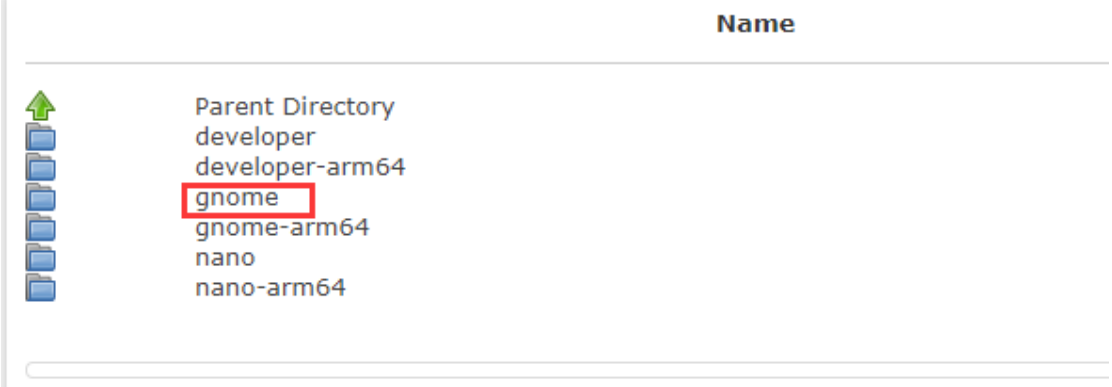


里面有 7 个文件，其中根文件系统放在 linaro-jessie-alip-20161117-32.tar.gz 文件中，我们可单击该文件进行下载，本章我们使用的就是该文件。

Ubuntu 根文件系统类似，返回到主界面，进入“archive”目录下，拉到页面底部，如下图所示：



看到一个名为“ubuntu”的目录，进入该目录的“images”目录下，如下图所示：



可以看到 Linaro 提供的 Ubuntu 发行版不同的根文件系统。其中名称带“arm64”表明是用于 ARM 64 位的架构，不带“arm64”的用于 ARM 32 位的架构。其中 gnome 表示该目录下的根文件系统，使用的是 gnome 图形界面。Developer 是不带桌面的，可自行安装。

该网站提供的 Debian 和 Ubuntu 的根文件系统都是可以直接使用的。下面我们以 Debian 根文件系统的 linaro-jessie-alip-20161117-32.tar.gz 文件为例，讲解如何使用。

12.2 修改 Vivado 工程

本实验是在之前 Linux 下 HDMI 显示实验的基础上实现的，所以硬件设计完

全相同

12.3 配置 Petalinux 工程

打开 VirtualBox 虚拟机进入 Ubuntu Linux 操作系统，打开 Terminal 窗口，输入下列命令挂载 VM_share 到/mnt，设置 Petalinux 运行所需的环境变量，

```
sudo mount –t vboxsf VM_share /mnt
source /opt/pkg/petalinux/2018.3/setting.sh
```

进入第 10 章创建的 petalinux 工程目录中，按照之前实验的操作步骤将共享文件夹中的 hdf 文件复制到工程目录，输入下列命令导入硬件设计

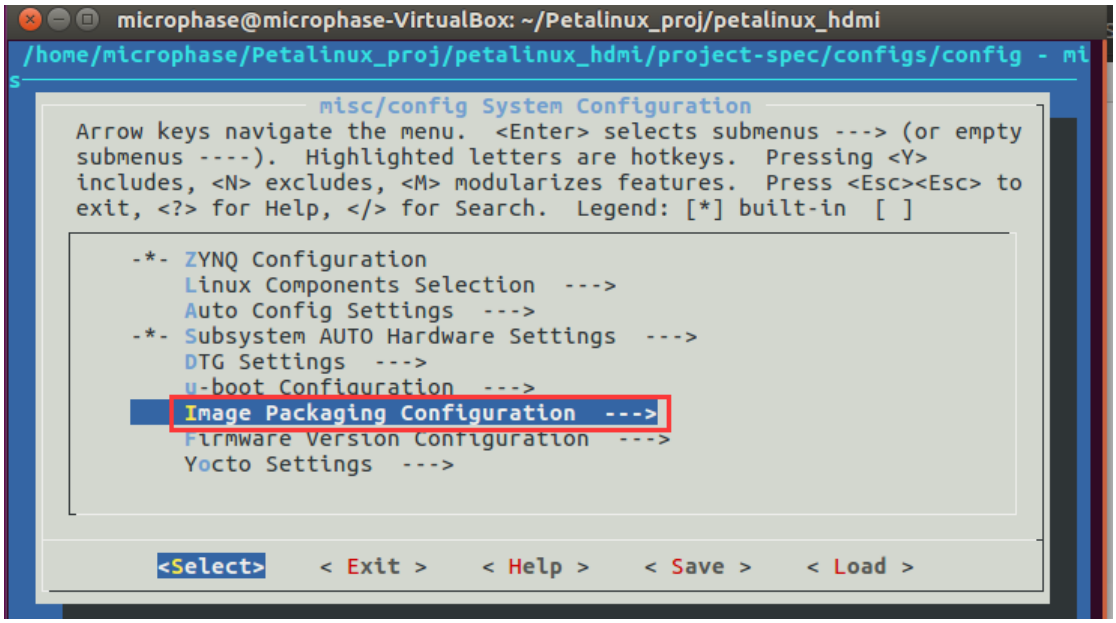
```
petalinux-config --get-hw-description=.
```

之后会打开设置菜单，保持默认，直接选择 SAVE 并退出即可。

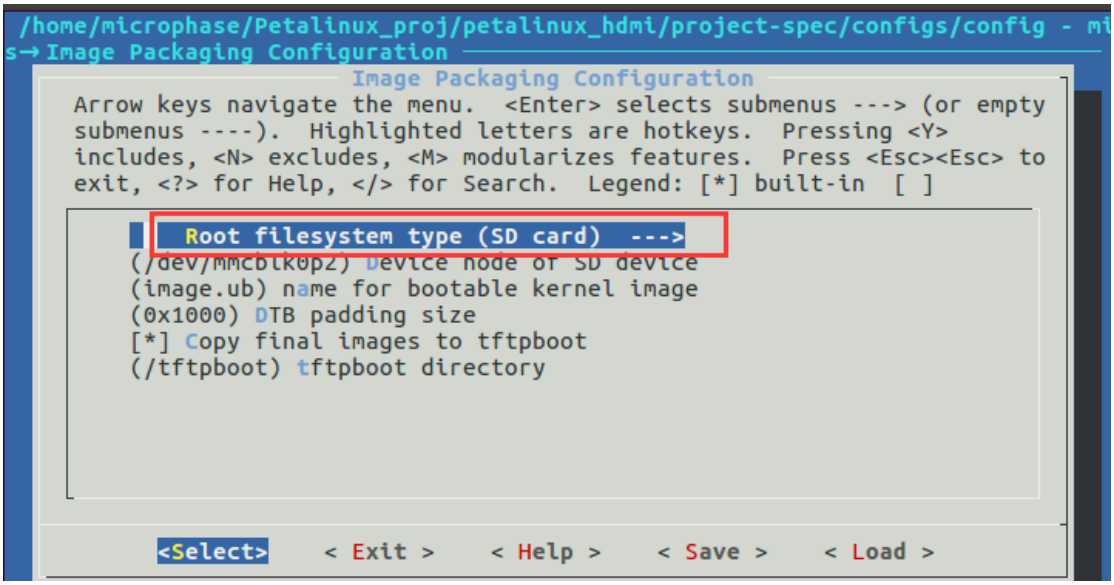
之后输入下列命令打开工程设置菜单，重新设置根文件系统类型，

```
petalinux-config
```

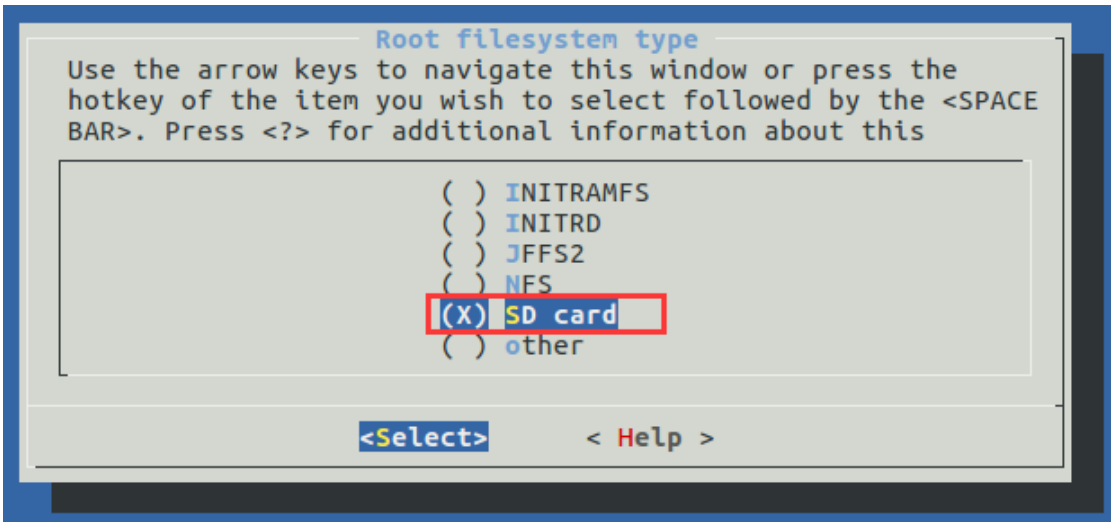
在弹出的配置菜单界面，进入到 “Image Packaging Configuration” 菜单，



进入“Root filesystem type(INITRAMFS)”子菜单，



选择“SD card”，按下回车选择并返回，



之后保存设置并退出，其他设置不动。

12.4 编译 Petalinux 工程并打包 BOOT.BIN 文件

输入下列命令编译工程

```
petalinux-build
```

```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_hdmi$ petalinux-buil
d
[INFO] building project
[INFO] sourcing bitbake
INFO: bitbake petalinux-user-image
Loading cache: 100% |##############################################| Time: 0:00:01
Loaded 3443 entries from dependency cache.
Parsing recipes: 100% |##########################################| Time: 0:00:11
Parsing of 2569 .bb files complete (2532 cached, 37 parsed). 3445 targets, 149 s
kipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies
Initialising tasks: 100% |#######################################| Time: 0:00:14
Checking sstate mirror object availability: 100% |###############| Time: 0:00:26
NOTE: Executing SetScene Tasks
NOTE: Executing RunQueue Tasks
NOTE: linux-xlnx: compiling from external source tree /home/microphase/Petalinux
_proj/linux_kernel
WARNING: petalinux-user-image-1.0-r0 do_rootfs: [log_check] petalinux-user-image
: found 1 warning message in the logfile:
```

编译完成后执行下列命令生成 BOOT.bin 文件

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

```
microphase@microphase-VirtualBox:~/Petalinux_proj/petalinux_hdmi$ petalinux-pack
age --boot --fsbl ./images/linux/zynq_fsbl.elf --fpga --u-boot --force
INFO: File in BOOT BIN: "/home/microphase/Petalinux_proj/petalinux_hdmi/images/l
inux/zynq_fsbl.elf"
INFO: File in BOOT BIN: "/home/microphase/Petalinux_proj/petalinux_hdmi/project-
spec/hw-description/hdmi_out_wrapper.bit"
INFO: File in BOOT BIN: "/home/microphase/Petalinux_proj/petalinux_hdmi/images/l
inux/u-boot.elf"
INFO: Generating Zynq binary package BOOT.BIN...


****** Xilinx Bootgen v2018.3
  **** Build date : Nov 15 2018-19:22:29
    ** Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.

INFO: Binary is ready.
WARNING: Unable to access the TFTPBOOT folder /tftpboot!!!
WARNING: Skip file copy to TFTPBOOT folder!!!
```

12.5 制作 SD 启动卡

在之前的实验中，我们没有使用单独的文件系统，故而不需要对 SD 卡进行

分区分别保存启动文件及文件系统。在本实验中我们就来学习在 Linux 下如何对 SD 卡进行分区。

将读卡器插入电脑 USB 接口，默认情况下挂载到 Windows 操作系统，在操作前先要将其挂载到 Ubuntu 下。右键点击虚拟机操作系统窗口右下脚的 UBS 图标打开 USB 选项菜单，选择挂载读卡器，如下图所示：

输入下列命令查看读卡器的挂载点，

```
ls /dev/sd*
```

其中 sda 为系统硬盘，所以 sdb 即为读卡器。如果显示磁盘较多无法确认，可以拔出读卡器再次查看，消失的磁盘即为读卡器。

输入下列命令运行磁盘管理工具，

```
sudo umount /dev/sdb*
sudo fdisk /dev/sdb
```

之后输入选项“p”，如下图所示：

```
microphase@microphase-VirtualBox:~$ sudo umount /dev/sdb*
[sudo] password for microphase:
umount: /dev/sdb: not mounted
microphase@microphase-VirtualBox:~$ sudo fdisk /dev/sdb

Welcome to fdisk (util-linux 2.27.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): p
Disk /dev/sdb: 7.4 GiB, 7935623168 bytes, 15499264 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x0b5db4cc

Device     Boot   Start      End  Sectors  Size Id Type
/dev/sdb1  *       2048  1026047  1024000  500M  c W95 FAT32 (LBA)
/dev/sdb2       1026048 15499263 14473216  6.9G 83 Linux

Command (m for help):
```

可以看到当前 SD 卡的分区情况，在重新分区之前需删除之前的分区，所以在分区之前请先保存好 SD 卡中的文件，按键输入“d”删除分区，默认从较大编号开始删除，

```
Command (m for help): d
Partition number (1,2, default 2):

Partition 2 has been deleted.

Command (m for help): d
Selected partition 1
Partition 1 has been deleted.

Command (m for help):
```

删除完成后再次输入“d”，可以看到红色字体提示，说明当前已无分区，

```
Command (m for help): p
Disk /dev/sdb: 7.4 GiB, 7935623168 bytes, 15499264 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x0b5db4cc

Command (m for help): d
No partition is defined yet!
Could not delete partition 1
```

下面开始新建分区。 输入“n”创建一个新分区。通过选择“p”使其为主分区，使用默认分区号 1 和第一个扇区 2048。设置最后一个扇区，也就是设置第一个分区的大小，一般设置 500M 足够了， 通过输入“+ 500M”，为该分区预留 500MB，如下图所示：

```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-15499263, default 2048): 2048
Last sector, +sectors or +size{K,M,G,T,P} (2048-15499263, default 15499263): +500M

Created a new partition 1 of type 'Linux' and of size 500 MiB.

Command (m for help):
```

接下来设置分区类型，输入“t”，然后输入“c”，设置为“W95 FAT32 (LBA)”，如下图所示：

```
Command (m for help): t
Selected partition 1
Partition type (type L to list all types): c
Changed type of partition 'W95 FAT32 (LBA)' to 'W95 FAT32 (LBA)'.

Command (m for help):
```

在设置分区类型时可以输入“l”，如下图所示，可以看到可设置的所有分区类型，

```
Partition type (type L to list all types): l

 0  Empty           24  NEC DOS         81  Minix / old Lin bf  Solaris
 1  FAT12           27  Hidden NTFS Win 82  Linux swap / So c1  DRDOS/sec (FAT-
 2  XENIX root      39  Plan 9          83  Linux           c4  DRDOS/sec (FAT-
 3  XENIX usr       3c  PartitionMagic  84  OS/2 hidden or  c6  DRDOS/sec (FAT-
 4  FAT16 <32M      40  Venix 80286     85  Linux extended  c7  Syrinx
 5  Extended        41  PPC PReP Boot   86  NTFS volume set da  Non-FS data
 6  FAT16           42  SFS             87  NTFS volume set db  CP/M / CTOS / .
 7  HPFS/NTFS/exFAT 4d  QNX4.x          88  Linux plaintext de  Dell Utility
 8  AIX             4e  QNX4.x 2nd part 8e  Linux LVM       df  BootIt
 9  AIX bootable    4f  QNX4.x 3rd part 93  Amoeba          e1  DOS access
 a  OS/2 Boot Manag 50  OnTrack DM      94  Amoeba BBT      e3  DOS R/O
 b  W95 FAT32       51  OnTrack DM6 Aux 9f  BSD/OS          e4  SpeedStor
 c  W95 FAT32 (LBA) 52  CP/M            a0  IBM Thinkpad hi ea  Rufus alignment
 e  W95 FAT16 (LBA) 53  OnTrack DM6 Aux a5  FreeBSD         eb  BeOS fs
 f  W95 Ext'd (LBA) 54  OnTrackDM6      a6  OpenBSD         ee  GPT
10  OPUS            55  EZ-Drive        a7  NeXTSTEP        ef  EFI (FAT-12/16/
11  Hidden FAT12    56  Golden Bow      a8  Darwin UFS      f0  Linux/PA-RISC b
12  Compaq diagnost 5c  Priam Edisk     a9  NetBSD          f1  SpeedStor
14  Hidden FAT16 <3 61  SpeedStor       ab  Darwin boot     f4  SpeedStor
16  Hidden FAT16    63  GNU HURD or Sys af  HFS / HFS+      f2  DOS secondary
17  Hidden HPFS/NTF 64  Novell Netware  b7  BSDI fs         fb  VMware VMFS
18  AST SmartSleep  65  Novell Netware  b8  BSDI swap       fc  VMware VMKCORE
```

分区类型设置完成之后输入“a”，将该分区设置为引导分区，如下图所示：

```
Command (m for help): a
Selected partition 1
The bootable flag on partition 1 is enabled now.
```

之后创建第二个分区，输入“n”开始创建分区，之后连续按回车使用默认设置即可，如下图所示：

```
Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p):

Using default response p.
Partition number (2-4, default 2):
First sector (1026048-15499263, default 1026048):
Last sector, +sectors or +size{K,M,G,T,P} (1026048-15499263, default 15499263):

Created a new partition 2 of type 'Linux' and of size 6.9 GiB.

Command (m for help):
```

创建完成输入“p”查看分区列表，检查刚刚创建完成的两个分区是否正确，没有问题则输入“w”将分区信息写入 SD 卡并退出。

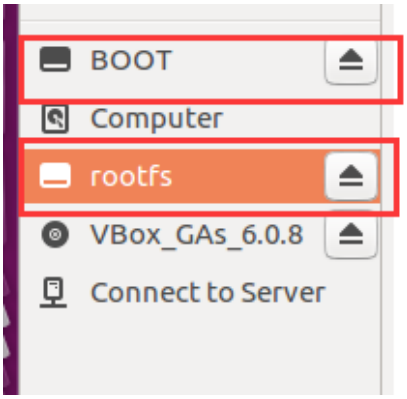
```
Command (m for help): p
Disk /dev/sdb: 7.4 GiB, 7935623168 bytes, 15499264 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x0b5db4cc

Device     Boot   Start      End  Sectors  Size Id Type
/dev/sdb1  *       2048  1026047  1024000  500M  c W95 FAT32 (LBA)
/dev/sdb2       1026048 15499263 14473216  6.9G 83 Linux

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

输入下列命令格式化分区，

```
sudo mkfs.vfat -F 32 -n BOOT /dev/sdb1
sudo mkfs.ext4 -L rootfs /dev/sdb2
```

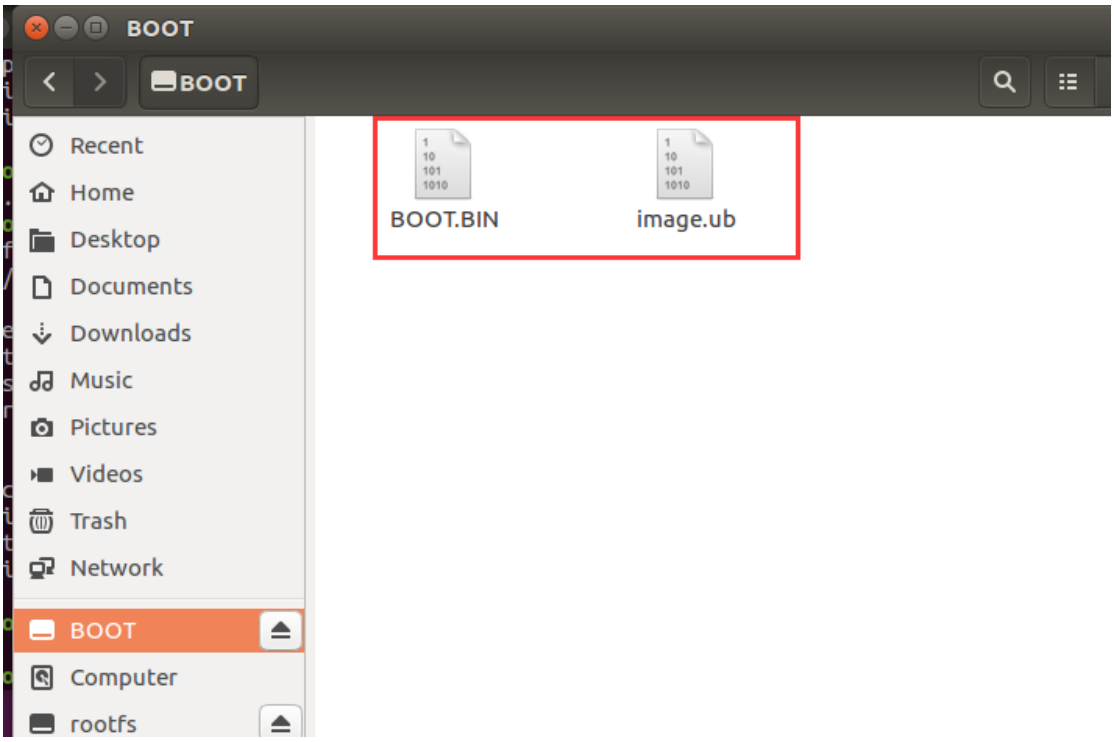
将第一个分区格式化成 FAT32 分区并命名为 BOOT，将第二个分区格式化成 ext4 分区并命名为 rootfs。

```
microphase@microphase-VirtualBox:~$ sudo mkfs.vfat -F 32 -n BOOT /dev/sdb1
mkfs.fat 3.0.28 (2015-05-16)
microphase@microphase-VirtualBox:~$ sudo mkfs.ext4 -L rootfs /dev/sdb2
mke2fs 1.42.13 (17-May-2015)
/dev/sdb2 contains a ext4 file system labelled 'rootfs'
        last mounted on /media/microphase/rootfs on Thu Apr 30 13:21:59 2020
Proceed anyway? (y,n) y
Creating filesystem with 1809152 4k blocks and 452480 inodes
Filesystem UUID: 327b4e79-35cf-49d5-b024-6bf2a24492f1
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

microphase@microphase-VirtualBox:~$
```



之后重新挂载 SD 卡，即可在文件管理器中看到设置的结果，

我们将该工程 image/linux 目录下的 BOOT.BIN 和 image.ub 文件拷贝到名为 BOOT 的分区也即/dev/sdb1 分区中，如下图所示：



将下载完成的 linaro-jessie-alip-20161117-32.tar.gz 文件复制到共享文件夹中，输入下列命令可以看到两个分区挂载的位置

```
ls /meida/microphase/
```

```
microphase@microphase-VirtualBox:~$ ls /media/microphase/
BOOT  rootfs  VBox_GAs_6.0.8
microphase@microphase-VirtualBox:~$
```

输入下列命令将根文件系统解压到 SD 卡的 rootfs 分区之中，

```
sudo tar zxvf /mnt/hdmi/share/linaro-jessie-alip-20161117-32.tar.gz -C
/media/microphase/rootfs/
```

```
microphase@microphase-VirtualBox:~$ sudo tar zxvf /mnt/hdmi/linaro-jessie-alip-2016
1117-32.tar.gz -C /media/microphase/rootfs/
```

输入下列命令查看解压后的文件，

```
ls /meida/microphase/rootfs
```

```
microphase@microphase-VirtualBox:~$ ls /media/microphase/rootfs/
binary  lost+found
microphase@microphase-VirtualBox:~$
```

可以看到有一个 binary 目录，我们需要将 binary 目录下的根文件系统移

动到 rootfs 分区下，输入如下命令

```
cd /media/microphase/rootfs/
sudo mv binary/* .
sudo rmdir binary/
```

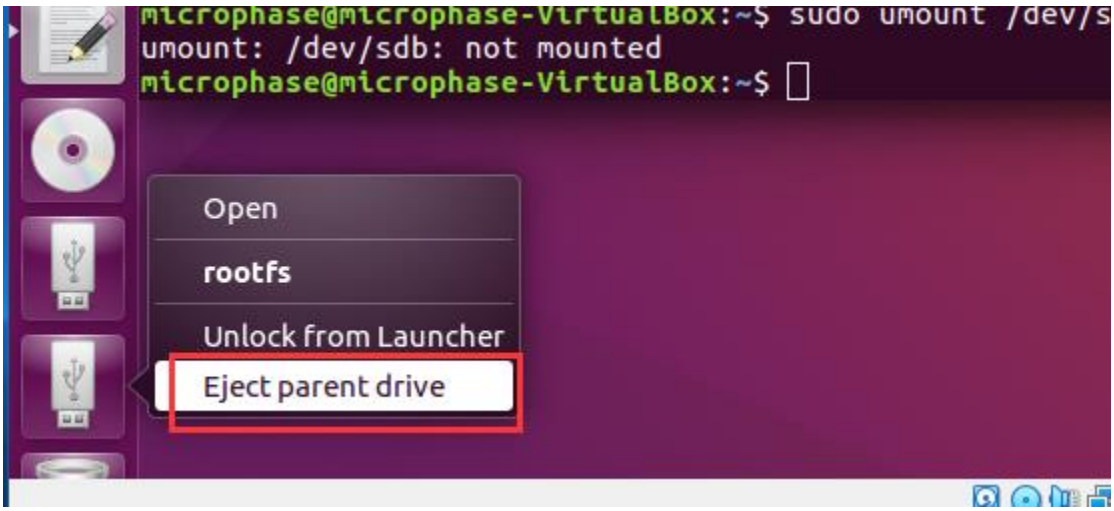
```
microphase@microphase-VirtualBox:~$ cd /media/microphase/rootfs/
microphase@microphase-VirtualBox:/media/microphase/rootfs$ sudo mv binary/* .
microphase@microphase-VirtualBox:/media/microphase/rootfs$ sudo rmdir binary/
microphase@microphase-VirtualBox:/media/microphase/rootfs$
```

之后输入下列命令卸载 SD 卡，输入上面命令后，终端中弹出下一个提示符时才能拔出 SD 卡，如下图所示：

```
sudo umount /dev/sdb*
```

```
microphase@microphase-VirtualBox:/media/microphase/rootfs$ cd ~
microphase@microphase-VirtualBox:~$ sudo umount /dev/sdb*
umount: /dev/sdb: not mounted
microphase@microphase-VirtualBox:~$
```

之后在桌面上右键点击 SD 卡图标，选择 Eject parent drive 弹出读卡器，之后即可拔出读卡器。



12.6 系统启动验证

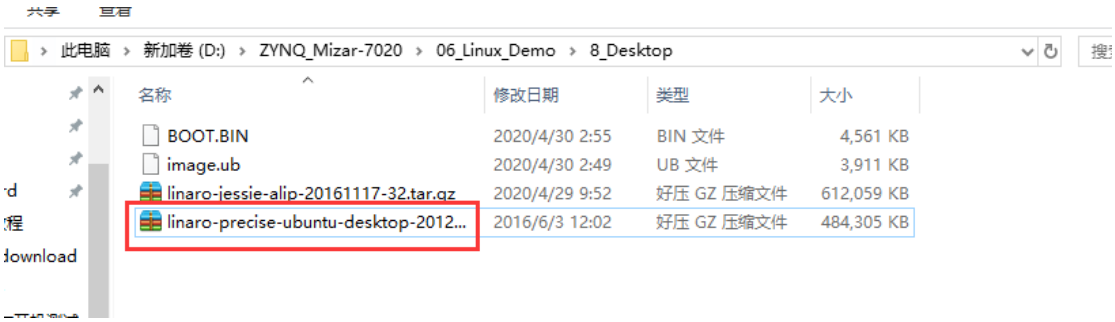
连接开发板电源，串口 usb 线。打开 SecureCRT 软件连接串口，上电启动 Linux 操作系统。使用 HDMI 连接线连接 HDMI TX 接口和显示屏，即可看到显示

器显示内容如下图所示：

串口终端打印信息如下图所示：

```
random: crng init done
[  OK  ] Started Bluetooth service.
         Stopping OpenBSD Secure Shell server...
[  OK  ] Stopped OpenBSD Secure Shell server.
[  OK  ] Started OpenBSD Secure Shell server.
[  OK  ] Started /etc/rc.local Compatibility.
         Starting Terminate Plymouth Boot Screen...
         Starting Wait for Plymouth Boot Screen to Quit...
[  OK  ] Started Terminate Plymouth Boot Screen.
[  OK  ] Started Wait for Plymouth Boot Screen to Quit.
[  OK  ] Started Serial Getty on ttyPS0.
[  OK  ] Started Getty on tty1.
[  OK  ] Reached target Login Prompts.
[  OK  ] Reached target Multi-User System.
[  OK  ] Reached target Graphical Interface.
         Starting Update UTMP about System Runlevel Changes...
[  OK  ] Started Update UTMP about System Runlevel Changes.

Debian GNU/Linux 8 linaro-alip ttyPS0

linaro-alip login: root (automatic login)

Linux linaro-alip 4.14.0-xilinx #1 SMP PREEMPT Thu Apr 30 01:24:59 UTC 2020 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@linaro-alip:~#
Ready                    Serial: COM3    45, 21    45 Rows, 120 C
```

现在将键盘鼠标插入 USB 接口即可进行操作，操作方式同 Windows 系统桌面类似。在资料文件夹内已包含相关的启动文件及图形界面根文件系统，如下图所示：



读者可以尝试使用 Ubuntu 根文件系统，不同的是解压之后，根文件系统的文件保存在 binary/boot/filesystem.dir/中。