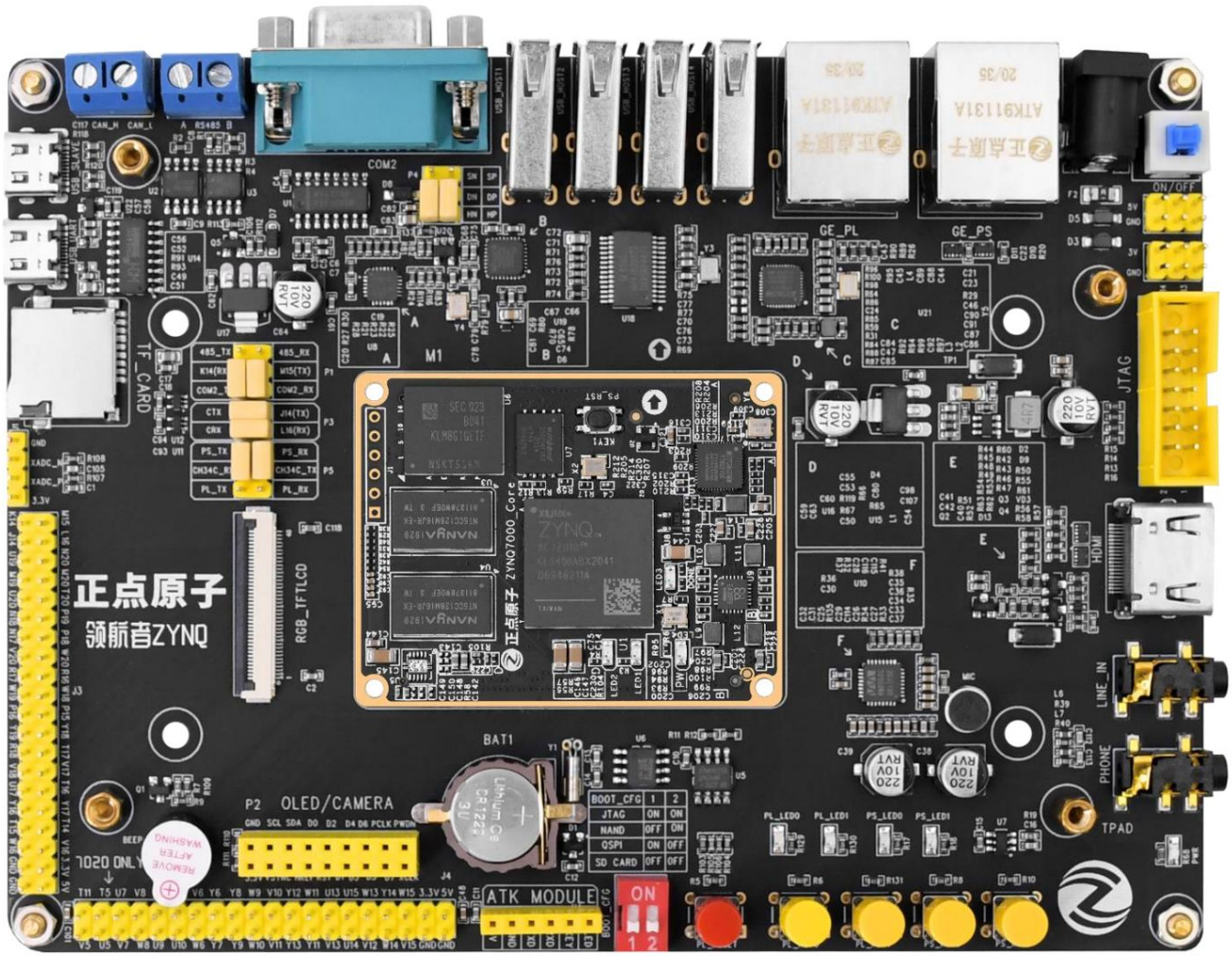
ZYNQ

Compile the factory image V1.0



正点原子

Original company name: Guangzhou Xingyi Electronic Technology Co., Ltd.

Company phone: 020 - 38271790

Company website: www.alientek.com

Online teaching platform: www.yuanzige.com

Open Source Electronics Network: www.openedv.com/forum.php

Tmall flagship store: https://zhengdianyuanzi.tmall.com

Zhengdian Atom B Station: https://space.bilibili.com/394620890

Scan the QR code to follow the Zhengdian Atom public account to get more embedded learning materials

Scan the QR code to download the Atom Brother App, which provides thousands of free open source video lessons

Scan the QR code to follow the official B station Zhengdian Atom, all videos can be watched online for free

Scan the QR code to follow Douyin's Zhengdian Atom, combining technology and entertainment to experience double happiness

扫码关注正点原子公众号

扫码下载“原子哥”APP


扫码关注正点原子B站

扫码关注正点原子抖音账号

Table of contents

Preface

Since many users need to develop based on the factory image, it is inevitable that they will encounter areas that need to be modified, such as adding new Drivers and add new applications, and then regenerate the startup file. This article explains how to compile the factory image startup file from source code.

If the user has modified the uboot and kernel source code, you can recompile uboot and kernel according to the documentation. Just unzip it to the sd card and add or delete files.

Chapter 1 Compile the factory image

This chapter introduces how to compile and generate the image files burned by the development board when it leaves the factory, including the BOOT.BIN image file, u-boot image file, kernel image file and device tree.

1.1 Install ZYNQ-7000 cross-compilation toolchain

To compile uboot and kernel source code, you need to install the corresponding cross-compilation tool chain in the Linux system (virtual machine). The cross-compilation toolchain for ZYNQ-7000 series chips requires the sdk.sh file.

Copy the development board data disk B:\sdk\202002\sdk.sh to the Ubuntu virtual machine.

Switch to the directory where the sdk.sh file is located, grant executable permissions to the sdk.sh file and install it. The command is as follows: #If build-essential, git, and u-boot-tools have not been installed before, execute the following command to install them first

```
sudo apt update sudo
apt -y install build-essential git u-boot-tools #Install SDK chmod +x sdk.sh ./
sdk.sh By
default, it is installed
in the /opt/
petalinux/2020.2 directory. If you want to install it in another directory, you can enter the corresponding
```

Here I keep the default path and press Enter to continue. The result is as shown below:

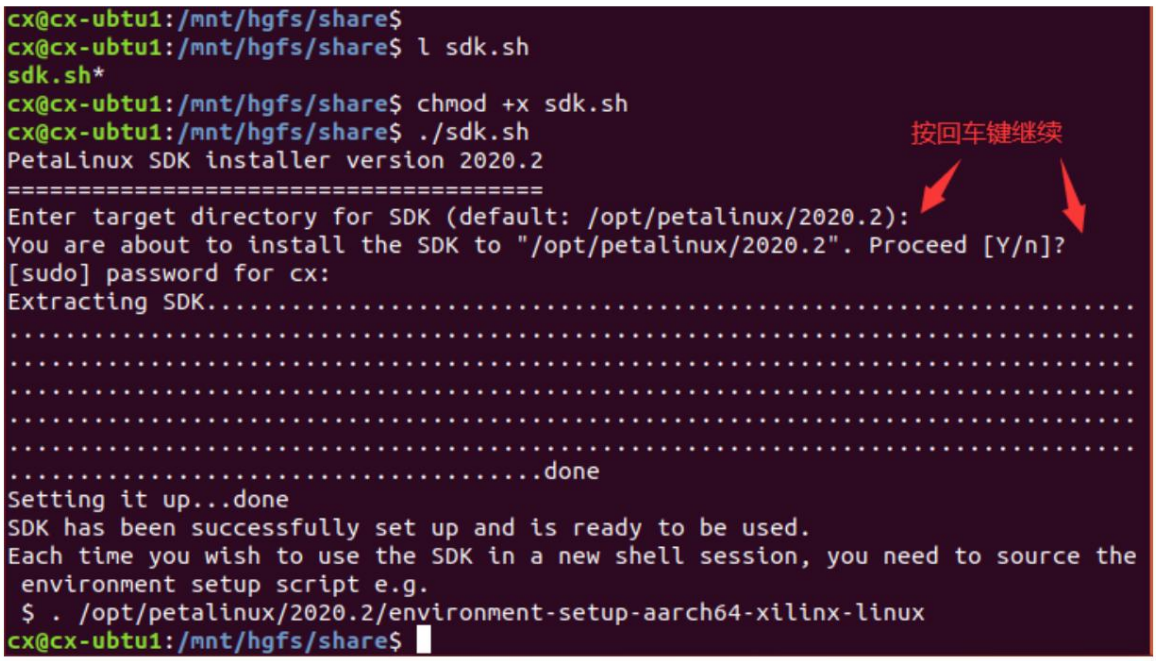


Figure 1.1.1 Install SDK

Confirm again whether to install the SDK in the /opt/petalinux/2020.2 directory. The default is "Y", which means "yes". Press the Enter key to continue. It will display the user password. After entering, press Enter to install and wait for the installation to complete. After the installation is complete, it will prompt that every time you use the SDK in a new terminal, you need to execute ". /opt/petalinux/2020.2/ environment-setup-cortexa9t2hf-neon-xilinx-linux-gnueabi" to set the corresponding environment variables. The "." and source have the same meaning.

1.2 Copy the source code to the Ubuntu system

1.2.1 Copy **u-boot** source code

The u-boot source code path used by the development board factory image is: Development board data disk (A disk) \4_SourceCode\3_Embedded_Linux\Resource files\Factory image related. In this directory, there is a compressed package file named atk-zup-uboot-xlnx.tar.gz, as shown below:

Figure 1.2.1 u-boot source code

atk-zynq-uboot-xlnx.tar.gz is a u-boot source code compressed package file specially used for factory testing of development boards. We copy the atk-zynq-uboot-xlnx.tar.gz compressed package file to the Ubuntu system as follows:

Figure 1.2.2 Copy the u-boot compressed package file to Ubuntu

Next, unzip it. The corresponding unzip directory is the U-Boot source directory. You can set this unzip directory yourself. Because after decompression, a folder named atk-zynq-uboot-xlnx will be automatically created in the decompression directory.

The following is the uboot source code. In order to avoid confusion with the uboot used in the tutorial, the author chooses to unzip it to the user's home directory. In the ~/workspace/src directory under the directory.

Execute the following command to unzip it to the ~/workspace/src/ directory:

```
mkdir -p ~/workspace/src/ #Create ~/workspace/src/folder
cd /mnt/hgfs/share/source_code/ tar -xzf atk-zynq-uboot-xlnx.tar.gz -C ~/workspace/src/ #Unzip
ls ~/workspace/src/
ls ~/workspace/src/atk-zynq-uboot-xlnx/
```

#Switch to the directory where the uboot compressed package file is located

```
2023-04-17 14:59:28 py-2.7.17 cx-ubtu in ~
o → mkdir -p ~/workspace/src/

2023-04-17 15:00:01 py-2.7.17 cx-ubtu in ~
o → cd /mnt/hgfs/share/source_code/

2023-04-17 15:00:13 py-2.7.17 cx-ubtu in /mnt/hgfs/share/source_code
o → mkdir -p ~/workspace/src/

2023-04-17 15:00:17 py-2.7.17 cx-ubtu in /mnt/hgfs/share/source_code
o → tar -xzf atk-zynq-uboot-xlnx.tar.gz -C ~/workspace/src/

2023-04-17 15:00:27 py-2.7.17 cx-ubtu in /mnt/hgfs/share/source_code
o → ls ~/workspace/src/
atk-zynq-uboot-xlnx
```

1 解压后得到的uboot源码根目录

```
2023-04-17 15:00:35 py-2.7.17 cx-ubtu in /mnt/hgfs/share/source_code
o → ls ~/workspace/src/atk-zynq-uboot-xlnx/
api                          cmd        doc        fs        Licenses      README
arch                         common     drivers    include   MAINTAINERS   scripts
atk-zup-boot.cmd.default     config.mk  dts        Kbuild    Makefile      test
atk-zynq-boot.cmd.default    configs    env        Kconfig   net           tools
board                        disk       examples   lib       post

2023-04-17 15:01:09 py-2.7.17 cx-ubtu in /mnt/hgfs/share/source_code
```

Figure 1.2.3 Unzip U-Boot source package

After decompression, the atk-zynq-uboot-xlnx folder contains the uboot source code used by the factory image.

1.2.2 Copy the kernel source code

The Linux kernel source code path used by the development board factory image is: Development Board Data Disk (Disk A)\4_SourceCode\3_Embedded_Linux\Resource Files\Factory Image Related. In this directory, there is a compressed package file named atk-zynq-linux-xlnx.tar.gz, as shown below:

名称	修改日期	类型	大小
atk-zynq-linux-xlnx.tar.gz	2023/4/17 14:30	GZ 压缩文件	169,579 KB

Figure 1.2.4 Kernel source code compressed package file

atk-zynq-linux-xlnx.tar.gz is a Linux kernel source code compressed package file specifically used for the factory image of the development board. We copy the atk-zynq-linux-xlnx.tar.gz compressed package file to the Ubuntu system as follows:

```
2023-04-17 15:01:09 py-2.7.17 cx-ubtu in /mnt/hgfs/share/source_code
o → l /mnt/hgfs/share/source_code
总用量 184M
-rwxrwxrwx 1 root root 166M 4月  17 14:30 atk-zynq-linux-xlnx.tar.gz*
-rwxrwxrwx 1 root root  19M 4月  17 14:29 atk-zynq-uboot-xlnx.tar.gz*
```

Figure 1.2.5 Copy the kernel source code compressed package file to Ubuntu

 Next, unzip it. The corresponding unzip

directory is the Linux kernel source code directory. You can set this unzip directory yourself. Because after unzipping, a folder named linux-xlnx will be automatically created in the unzipped directory, and the Linux kernel source code is in this folder. In order to avoid confusion with the Linux kernel source code used in the tutorial, the author chooses to unzip it to the ~/workspace/src directory under the user's home directory: Execute the following

command to unzip it to the ~/workspace/src directory: mkdir -p ~/workspace/

```
src/ cd /mnt/hgfs/share/source_code/ tar
-xzf atk-zynq-linux-xlnx.tar.gz -C ~/workspace/src/          #Switch to the directory where the uboot compressed package file is located
#Unzip
sync
```

```
2023-04-17 15:08:46  py-2.7.17 cx-ubtu in /mnt/hgfs/share/source_code
o → mkdir -p ~/workspace/src/

2023-04-17 15:10:05  py-2.7.17 cx-ubtu in /mnt/hgfs/share/source_code
o → cd /mnt/hgfs/share/source_code/

2023-04-17 15:10:10  py-2.7.17 cx-ubtu in /mnt/hgfs/share/source_code
o → tar -xzf atk-zynq-linux-xlnx.tar.gz  -C ~/workspace/src/

2023-04-17 15:10:20  py-2.7.17 cx-ubtu in /mnt/hgfs/share/source_code
o → sync

2023-04-17 15:10:29  py-2.7.17 cx-ubtu in /mnt/hgfs/share/source_code
```

Figure 1.2.6 Unzip the kernel source code

After decompression, the atk-zynq-linux-xlnx folder is the root directory of the Linux kernel source code used by the factory image. Use the ls command (ls ~/workspace/src/atk-zynq-linux-xlnx) to see the Linux kernel source code directory structure, as shown below:

```
2023-04-17 15:10:29  py-2.7.17 cx-ubtu in /mnt/hgfs/share/source_code
o → ls ~/workspace/src/
atk-zynq-linux-xlnx  atk-zynq-uboot-xlnx

2023-04-17 15:11:00  py-2.7.17 cx-ubtu in /mnt/hgfs/share/source_code
o → ls ~/workspace/src/atk-zynq-linux-xlnx
arch      Documentation  Kbuild       Makefile             scripts   zynq-fit-image.its
block     drivers        Kconfig      mm                   security
certs     fs             kernel       mpsoc-fit-image.its  sound
COPYING   include        lib          net                  tools
CREDITS   init           LICENSES     README               usr
crypto    ipc            MAINTAINERS  samples              virt

2023-04-17 15:11:36  py-2.7.17 cx-ubtu in /mnt/hgfs/share/source_code
```

1 解压得到的内核源码根目录

Figure 1.2.7 Linux kernel source directory structure

1.2.3 Copy **the xsa** file

Use the xsa file to create the corresponding Petalinux project. In

the development board data package, the corresponding vivado project of the development board is provided.

For the Navigator 7020 development board, the Navigator_7020 project is used, and

for the Navigator 7010 development board, the Navigator_7010 project is used.

The author takes the Navigator 7020 development board as an example, the path is: development board data disk (A Disk)\4_SourceCode\3_Embedded_Linux\vivado_prj , there is a compressed file in this directory Navigator_7020.zip, unzip it in Windows system, and the unzipped file will look like the following figure:

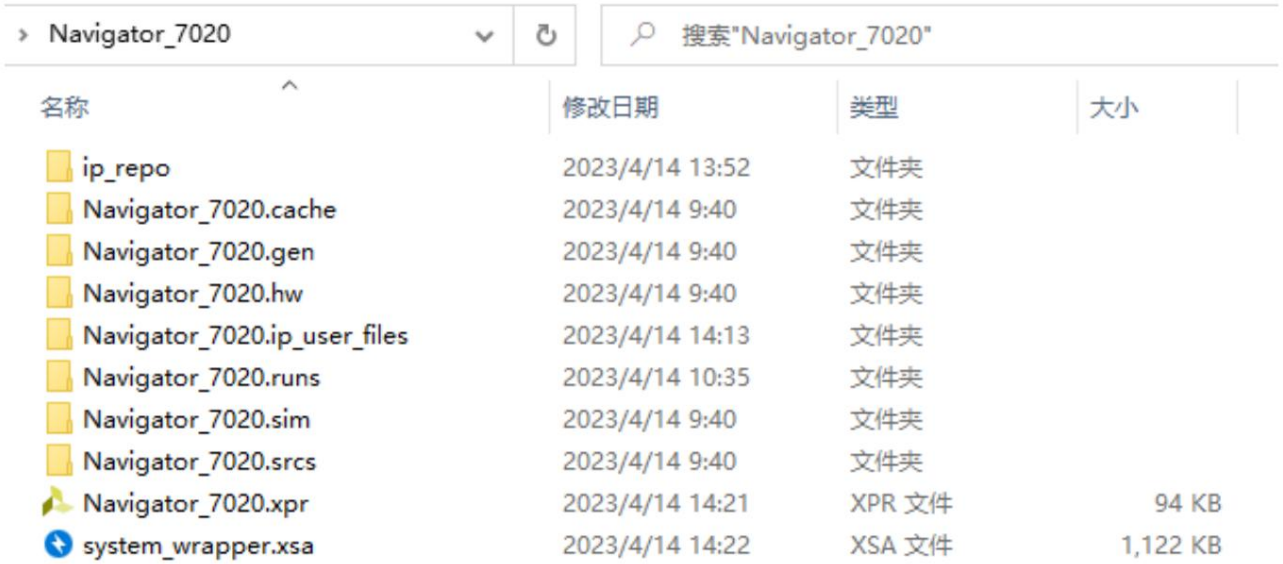


Figure 1.2.8 Vivado project directory

system_wrapper.xsa is the xsa file exported by Vivado. Here, we directly copy the system_wrapper.xsa file Copy the folder to a directory on the Ubuntu system, such as /mnt/hgfs/share/xsa/7020/, as shown in the following figure:

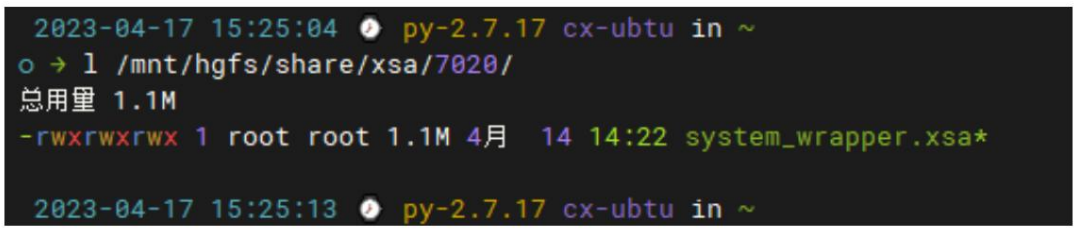


Figure 1.2.9 Copy the xsa file to Ubuntu

1.3 Compile

In the previous section, we have copied all the "raw materials" needed for compilation to the Ubuntu system. Next, we can Compile. Please note that before compiling, you need to install Xilinx's petalinux tool and set up the exchange Fork the working environment of the compilation tool chain. If you haven't installed petalinux yet, you can refer to [On-time Atom] Navigator ZYNQ Chapter 5 of the Embedded Linux Driver Development Guide V1.x.pdf: Installation of Petalinux; Setting up the switch

Please refer to the next section for the working environment of the fork compilation tool chain.

1.4 Setting up the cross-compilation toolchain working environment

Each time you open a new terminal, you need to execute the following command in the terminal to set the SDK environment variables to use cross-compilation Device:

```
. /opt/petalinux/2020.2/environment-setup-cortexa9t2hf-neon-xilinx-linux-gnueabi
```

If you don't want to execute this command every time you open a new terminal, you can put it in ~/.profile or In the /etc/profile file, the command is as follows:

```
echo '. /opt/petalinux/2020.2/environment-setup-cortexa9t2hf-neon-xilinx-linux-gnueabi tail -5 | tee -a ~/.profile #Note the following figure, this command is a complete line
~/.profile
```

The result is shown in the figure below:

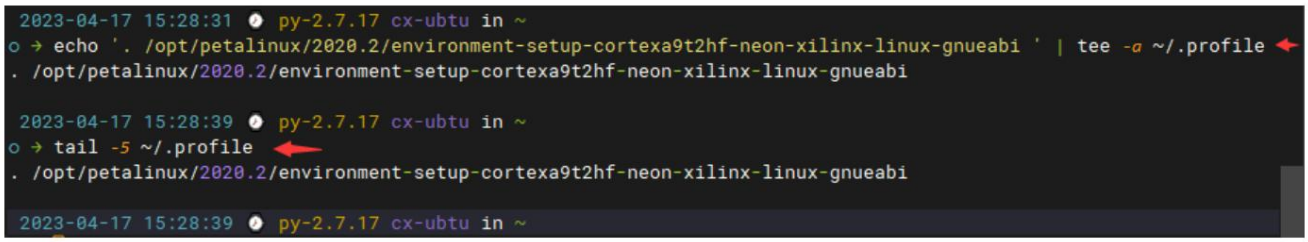


Figure 1.4.1 Setting SDK environment variables when loading ~/.profile file

In this way, write the command ". /opt/petalinux/2020.2/environment-setup-cortexa9t2hf-neon-xilinx-linux-gnueabi" to the ~/.profile file, and it will be automatically executed after starting the virtual machine and logging in to the current user.

This command. However, the current terminal is not available and needs to be restarted to take effect.

Note: If you have previously set up the Petalinux 2020.2 SDK in the same way, please first start from ~/.profile Delete the relevant lines in the files such as ./builds, or just use the previous SDK if available.

After setting the SDK environment variables, enter the command arm-xilinx-linux-gnueabi-gcc --version in the terminal to

Check the version number of the cross compiler currently in use. If you see the following result, it means that the environment variable has taken effect.

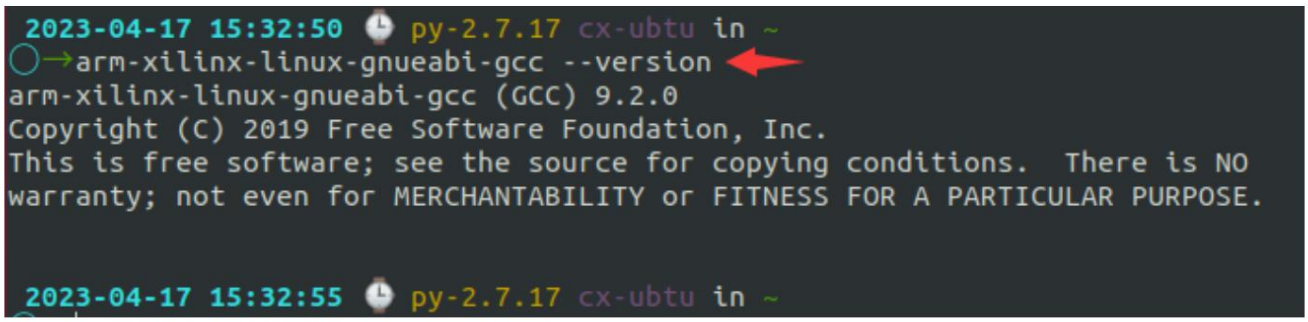


Figure 1.4.2 View gcc version information

As you can see, the zynq cross compiler version we use is 9.2.0.

After setting the environment variables, you can use the env command to view the effective environment variables. The following figure is a partial screenshot:

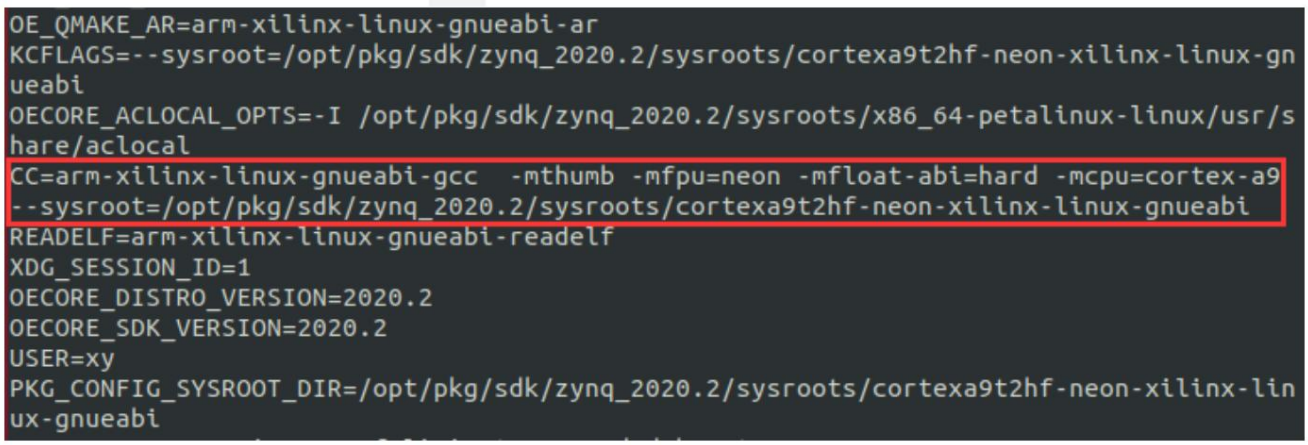


Figure 1.4.3 View the environment variables after setting

 Different environment variables have different functions. For example, the environment variable CC, you can see that the environment variable has configured the parameters used by the gcc cross compiler when compiling. For example, when using the arm-xilinx-linux-gnueabi-gcc compiler, you can directly use the environment variable $CC. For example, if you cross-compile a .c file, such as hello.c, you can directly use $CC hello.c.

1.4.1 Create **a Petalinux** project

The steps to create a Petalinux project are explained in Chapter 6 of the [Zhengdian Atom] Navigator ZYNQ Embedded Linux Driver Development Guide V1.x.pdf, Petalinux Design Process Practice. This chapter will not go into detail. You can also directly use the Petalinux project in Chapter 6 Petalinux Design Process Practice and re-import the xsa file.

First, select a suitable path in the Ubuntu host terminal to create the Petalinux project used by the factory image, and then enter the following

command in the terminal: cd <corresponding directory> #Switch to the

#Set up petalinux working environment

directory where the Petalinux project is created sptl petalinux-create -t project --template zynq -n base #Create the Petalinux project cd base #Enter the petalinux project directory petalinux-

config --get-hw-description /mnt/hgfs/share/xsa/7020 #Import the corresponding xsa file After the above command is executed, a folder named base will be

automatically created in the current directory, which is the

The project directory corresponding to the petalinux project of the factory

image. After the xsa file is successfully imported, the petalinux project configuration window will pop up automatically, as shown below:

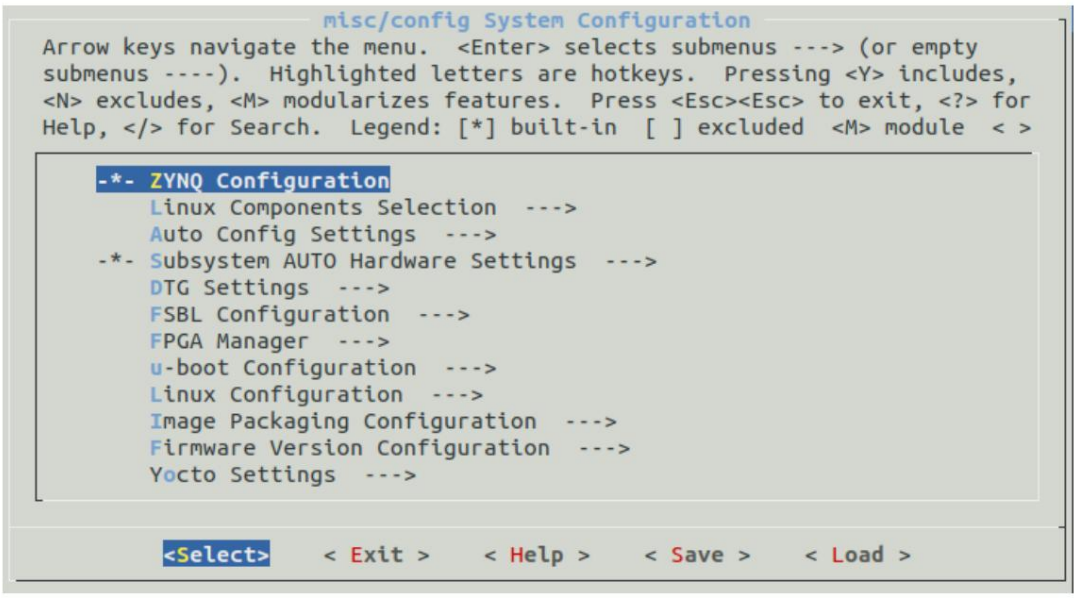


Figure 1.4.4 petalinux project configuration window.

Use the default configuration, save and exit.

1.4.2 Compile the factory source **u-boot and make BOOT.BIN**

Enter the u-boot source code root directory.

The configuration file corresponding to the Navigator development board is: configs/xilinx_zynq_virt_defconfig The device tree file corresponding to the Navigator development board is: arch/arm/dts/zynq-atk.dts Execute the following command to compile the u-boot source code:

```
make distclean                                  # Clean up the project
make xilinx_zynq_virt_defconfig #Configure uboot
make -j                                         # Compile
```

```
 2023-04-17 15:44:04  py-2.7.17 cx-ubtu in ~
o → cd workspace/src/atk-zynq-uboot-xlnx/

 2023-04-17 15:44:06  py-2.7.17 cx-ubtu in ~/workspace/src/atk-zynq-uboot-xlnx
o → make distclean

 2023-04-17 15:44:09  py-2.7.17 cx-ubtu in ~/workspace/src/atk-zynq-uboot-xlnx
o → make xilinx_zynq_virt_defconfig
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  YACC    scripts/kconfig/zconf.tab.c
  LEX     scripts/kconfig/zconf.lex.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#

 2023-04-17 15:44:14  py-2.7.17 cx-ubtu in ~/workspace/src/atk-zynq-uboot-xlnx
o → make -j
scripts/kconfig/conf  --syncconfig Kconfig
  CHK     include/config.h
  UPD     include/config.h
  CFG     u-boot.cfg
  GEN     include/autoconf.mk.dep
  CFG     spl/u-boot.cfg
  GEN     include/autoconf.mk
  GEN     spl/include/autoconf.mk
  CHK     include/config/uboot.release
```

Figure 1.4.5 Compile u-boot source code

After successful compilation, an image file will be generated in the u-boot directory, as shown below:

```
 2023-04-17 15:44:30  py-2.7.17 cx-ubtu in ~/workspace/src/atk-zynq-uboot-xlnx
o → ls
api                          doc       Licenses     tools              u-boot-elf.o
arch                         drivers   MAINTAINERS  u-boot             u-boot.img
atk-zup-boot.cmd.default     dts       Makefile     u-boot.bin         u-boot.lds
atk-zynq-boot.cmd.default    env       net          u-boot.cfg         u-boot.map
board                        examples  post         u-boot.cfg.configs u-boot-nodtb.bin
cmd                          fs        README       u-boot.dtb         u-boot.srec
common                       include   scripts      u-boot-dtb.bin     u-boot.sym
config.mk                    Kbuild    spl          u-boot-dtb.img
configs                      Kconfig   System.map   u-boot.elf
disk                         lib       test         u-boot-elf.lds

 2023-04-17 15:45:22  py-2.7.17 cx-ubtu in ~/workspace/src/atk-zynq-uboot-xlnx
```

Figure 1.4.6 Generate image file

u-boot.elf is a uboot executable file in elf format, and u-boot.dtb is the uboot device tree file.

command to generate the boot.scr startup script file: ./tools/

```
mkimage -c none -A arm -T script -d atk-zynq-boot.cmd.default boot.scr
```

```
2023-04-17 15:45:22 py-2.7.17 cx-ubtu in ~/workspace/src/atk-zynq-uboot-xlnx
o → ./tools/mkimage -c none -A arm -T script -d atk-zynq-boot.cmd.default boot.scr
Image Name:
Created:      Mon Apr 17 15:46:29 2023
Image Type:   ARM Linux Script (gzip compressed)
Data Size:    2542 Bytes = 2.48 KiB = 0.00 MiB
Load Address: 00000000
Entry Point:  00000000
Contents:
   Image 0: 2534 Bytes = 2.47 KiB = 0.00 MiB

2023-04-17 15:46:29 py-2.7.17 cx-ubtu in ~/workspace/src/atk-zynq-uboot-xlnx
```

Figure 1.4.7 Generate boot.scr

boot.scr is the script file used by uboot in the factory image to load the kernel image and device tree from the corresponding storage device. Switch to the created Petalinux project directory and execute the following command to make the

boot file BOOT.BIN required by the Navigator development board: petalinux-build -c bootloader petalinux-package --boot --fsbl --fpga \ --u-boot ~/workspace/src/atk-zynq-uboot-xlnx/u-boot.elf \ --dtb ~/workspace/src/atk-zynq-uboot-xlnx/u-boot.dtb --force

```
#Generate fsbl.elf file
```

```
2023-04-17 17:57:14 py-2.7.17 cx-ubtu in ~/workspace/petalinux/base
o → petalinux-package --boot --fsbl --fpga --u-boot ~/workspace/src/atk-zynq-uboot-xlnx/u-boot.elf --dtb ~/workspace/src/atk-zynq-uboot-xlnx/u-boot.dtb --force
INFO: Sourcing build tools
INFO: File in BOOT BIN: "/home/xy/workspace/petalinux/base/images/linux/zynq_fsbl.elf"
INFO: File in BOOT BIN: "/home/xy/workspace/petalinux/base/project-spec/hw-description/system_wrapper.bit"
INFO: File in BOOT BIN: "/home/xy/workspace/src/atk-zynq-uboot-xlnx/u-boot.elf"
INFO: File in BOOT BIN: "/home/xy/workspace/src/atk-zynq-uboot-xlnx/u-boot.dtb"
INFO: Generating Zynq binary package BOOT.BIN...


****** Xilinx Bootgen v2020.2
  **** Build date : Nov 15 2020-06:11:24
    ** Copyright 1986-2020 Xilinx, Inc. All Rights Reserved.


[INFO]   : Bootimage generated successfully

INFO: Binary is ready.

2023-04-17 17:59:17 py-2.7.17 cx-ubtu in ~/workspace/petalinux/base
```

Figure 1.4.8 Creating BOOT.BIN file

Note: If the compilation reports errors related to the device tree, please configure the device as described in Chapter 6 Petalinux Design Process Practice We don't use the device tree compiled by Petalinux, we just solve the error. After the command is successfully executed, the BOOT.BIN startup file will be generated in the images/linux directory of the current Petalinux project, as shown in the following figure:

```
2023-04-17 18:02:46 py-2.7.17 cx-ubtu in ~/workspace/petalinux/base
o → l images/linux/
总用量 5.2M
-rw-rw-r-- 1 xy xy 4.7M 4月  17 17:59 BOOT.BIN
-rw-r--r-- 1 xy xy 539K 4月  17 17:52 zynq_fsbl.elf
```

Figure 1.4.9 Generate BOOT.BIN file

1.4.3 Compile kernel, device tree

the root directory of the Linux kernel source code.

The kernel defconfig configuration file is: arch/arm/configs/xilinx_zynq_defconfig

The device tree file corresponding to the Navigator 7020 development board is: arch/arm/boot/dts/atk-navigator-7020.dts

The device tree file corresponding to the Navigator 7010 development board is: arch/arm/boot/dts/atk-navigator-7010.dts

Execute the following command to compile the kernel source code:

```
make distclean make                    # Clean up the project
xilinx_zynq_defconfig make -j8         # Configuration
                                       # Compile zImage and device tree
```

```
 2023-04-17 15:49:46  py-2.7.17 cx-ubtu in ~/workspace/src/atk-zynq-linux-xlnx
o → make distclean

 2023-04-17 15:51:37  py-2.7.17 cx-ubtu in ~/workspace/src/atk-zynq-linux-xlnx
o → make xilinx_zynq_defconfig
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  HOSTCC  scripts/kconfig/confdata.o
  HOSTCC  scripts/kconfig/expr.o
  LEX     scripts/kconfig/lexer.lex.c
  YACC    scripts/kconfig/parser.tab.[ch]
  HOSTCC  scripts/kconfig/lexer.lex.o
  HOSTCC  scripts/kconfig/parser.tab.o
  HOSTCC  scripts/kconfig/preprocess.o
  HOSTCC  scripts/kconfig/symbol.o
  HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#

 2023-04-17 15:51:47  py-2.7.17 cx-ubtu in ~/workspace/src/atk-zynq-linux-xlnx
o → make -j8
  HOSTCC  scripts/dtc/dtc.o
  HOSTCC  scripts/dtc/flattree.o
  HOSTCC  scripts/dtc/data.o
  HOSTCC  scripts/dtc/fstree.o
  HOSTCC  scripts/dtc/livetree.o
  SYSHDR  arch/arm/include/generated/uapi/asm/unistd-common.h
  SYSHDR  arch/arm/include/generated/uapi/asm/unistd-oabi.h
  UPD     include/config/kernel.release
  HOSTCC  scripts/dtc/treesource.o
```

Figure 1.4.10 Compiling kernel source code

The compiled kernel image file zImage is in the arch/arm/boot directory, as shown below:

```
 2023-04-17 15:53:32  py-2.7.17 cx-ubtu in ~/workspace/src/atk-zynq-linux-xlnx
o → l arch/arm/boot/
总用量 14M
drwxrwxr-x 2 xy xy 4.0K 4月  14 17:56 bootp/
drwxrwxr-x 2 xy xy 4.0K 4月  17 15:53 compressed/
-rwxrwxr-x 1 xy xy 1.7K 4月  14 17:56 deflate_xip_data.sh*
drwxrwxr-x 2 xy xy  96K 4月  17 15:51 dts/
-rwxrwxr-x 1 xy xy  13M 4月  17 15:53 Image*
-rw-rw-r-- 1 xy xy 1.7K 4月  14 17:56 install.sh
-rw-rw-r-- 1 xy xy 3.1K 4月  14 17:56 Makefile
-rwxrwxr-x 1 xy xy 4.7M 4月  17 15:53 zImage*
```

Figure 1.4.11 The device tree file compiled

from the generated kernel image file is in the arch/arm/boot/dts directory, as shown below:

```
 2023-04-17 15:55:06  py-2.7.17 cx-ubtu in ~/workspace/src/atk-zynq-linux-xlnx
o → l arch/arm/boot/dts/atk*
-rw-rw-r-- 1 xy xy 22K 4月  17 15:51 arch/arm/boot/dts/atk-navigator-7010.dtb
-rw-rw-r-- 1 xy xy 643 4月  14 17:56 arch/arm/boot/dts/atk-navigator-7010.dts
-rw-rw-r-- 1 xy xy 24K 4月  17 15:51 arch/arm/boot/dts/atk-navigator-7020.dtb
-rw-rw-r-- 1 xy xy 629 4月  14 17:56 arch/arm/boot/dts/atk-navigator-7020.dts

 2023-04-17 15:55:13  py-2.7.17 cx-ubtu in ~/workspace/src/atk-zynq-linux-xlnx
```

Figure 1.4.12 The device tree file atk-

navigator-7020.dtb corresponds to the device tree file of the Navigator 7020 development

board; atk-navigator-7010.dtb corresponds to the device tree file of the Navigator 7010

development board. So far, all the image files required to start the development board have been compiled, including the BOOT.BIN file, boot.scr, zImage, device tree dtb four files.

1.5 start up

Make an SD boot card. For the method of making an SD boot card, please refer to the section on making an SD boot card in Chapter 6 of [On-point Atom] Navigator ZYNQ Embedded Linux Driver Development Guide V1.x.pdf, Petalinux Design Process Practice. I will not

repeat it here! Copy the four image files compiled in the previous section to the first partition of the SD boot card, which is the Fat32

partition. The BOOT.BIN file is located in the images/linux directory of the Petalinux

project; boot.scr is located in the uboot source code

root directory; the zImage file is located in the arch/arm/boot/directory in the linux kernel source code

root directory; the dtb file is located in the arch/arm/boot/dts/directory in the linux kernel source code

root directory. After copying, it is as follows:

```
 2023-04-17 16:07:53  py-2.7.17 cx-ubtu in ~
o → cd /media/xy/BOOT/

 2023-04-17 16:07:56  py-2.7.17 cx-ubtu in /media/xy/BOOT
o → ls
atk-navigator-7010.dtb  atk-navigator-7020.dtb  BOOT.BIN  boot.scr  zImage

 2023-04-17 16:07:57  py-2.7.17 cx-ubtu in /media/xy/BOOT
```

Figure 1.5.1 SD boot card Fat partition file

Note: Generally, you only need to copy the device tree dtb file corresponding to the development board. You can also copy multiple device tree files. Only the device tree dtb file corresponding to the development board is used during

the startup process. Next, we need to copy the root file system to the second partition of the SD startup card. The root file system used when leaving the factory has been provided in the development board data package. There is a compressed package file named rootfs.tar.gz in the development board data disk (disk A)\4_SourceCode\3_Embedded_Linux\resource files\factory image related directory , as shown below:

名称	修改日期	类型	大小
rootfs.tar.gz	2022/6/27 17:24	GZ 压缩文件	232,352 KB
atk-zup-uboot-xlnx.tar.gz	2022/6/22 15:07	GZ 压缩文件	193,959 KB
atk-zup-linux-xlnx.tar.gz	2022/6/22 16:33	GZ 压缩文件	2,345,508...

Figure 1.5.2 Root file system

The root file system contained in rootfs.tar.gz is compiled using Petalinux and is also the root file system used by the factory image. Due to testing needs, the author also transplanted some software into it. This root file system is relatively large and contains a lot of content, including common library files such as Qt5, Python, and opencv.

Copy rootfs.tar.gz to the Ubuntu system, and then use the tar command to compress the rootfs.tar.gz file into

Unzip the file to the second partition of the SD boot card as follows: sudo

```
tar -xzf rootfs.tar.gz -C /media/$USER/rootfs ls /media/$USER/rootfs


sync
```

```
cx-ubtu in /mnt/hgfs/share/source_code
→ sudo tar -xzf rootfs.tar.gz -C /media/$USER/rootfs
cx-ubtu in /mnt/hgfs/share/source_code
→ ls /media/$USER/rootfs
bin   dev  home  media  modules  proc  sbin  tmp  var
boot  etc  lib   mnt    opt      run   sys   usr
cx-ubtu in /mnt/hgfs/share/source_code
→ sync
cx-ubtu in /mnt/hgfs/share/source_code
```

Figure 1.5.3 Unzip the root file system to the second partition of the SD card

After the synchronization is complete, unmount the SD card (umount command), unplug the SD card and insert it into the SD card slot of the development board. Set the development board boot mode to SD card boot, connect the serial port, LCD screen and power supply to start the development board.

1.6 Extensions

1.6.1 How to make the factory image boot from **QSPI ?**

The factory image puts the boot file in QSPI and the root file system in eMMC. To boot from QSPI, you first need to package the zImage file and dtb file into an image.ub file like the factory image. This is also the default file format of Petalinux. The packaging method is as follows: First enter the root directory of the Linux kernel source code. For the 7020 core board, enter the following command: mkimage -f zynq-fit-image.its image.ub For the 7010 core board, enter the following command: sed -i 's/7020/7010/g' zynq-fit-image.its mkimage -f zynq-fit-image.its image.ub The generated image.ub file is in the root directory of the kernel source code. Enter the command ls -l image.ub to see it. After packaging is complete, copy the image.ub file and the previously compiled BOOT.BIN and boot.scr files to the SD card.

The first partition of the SD card is the Fat32 partition, where the BOOT.BIN file is located in the images/linux directory of the Petalinux project; boot.scr is located in the uboot source code root directory; image.ub file is located in the Linux source code root directory. The root file is still unzipped to the second partition of the SD card as before. After booting from the SD card, enter the following command in the serial terminal to solidify the boot image to QSPI and copy the root file system eMMC:

```
/opt/image/burn_qspi.sh
```

```
root@ALIENTEK-ZYNQ:~#
root@ALIENTEK-ZYNQ:~# /opt/image/burn_qspi.sh
##########################################
######### ALIENTEK ZYNQ Flash QSPI #########
######### Version: 1.0             #########
######### www.openedv.com          #########
######### Author: DengTao          #########
##########################################

Erase partition /dev/mtd0
```

Figure 1.6.1 Execute the fixed script

After the system is solidified, the buzzer will sound, indicating that the system image and root file system have been burned to the QSPI and eMMC storage media respectively, which also means that the development board can start the system in QSPI mode.