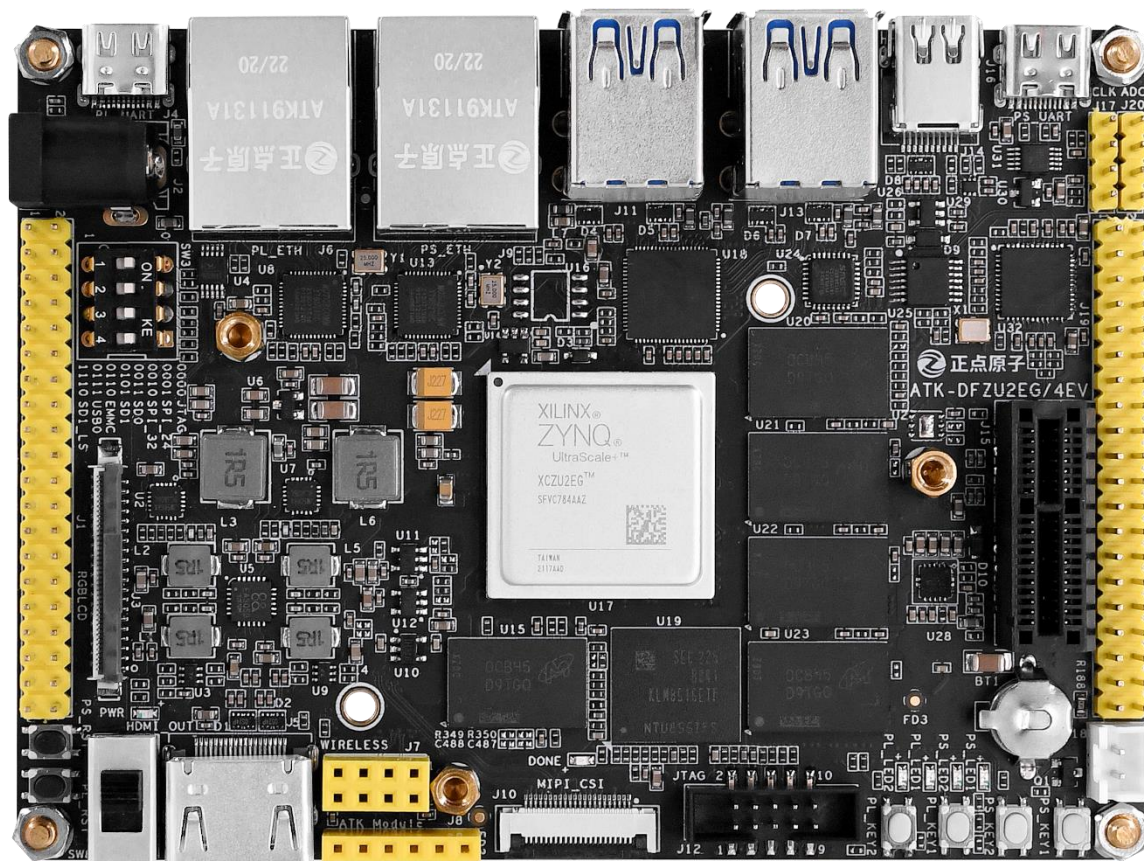


DFZU2EG_4EV 之 编译出厂镜像 V1.0





正点原子公司名称: 广州市星翼电子科技有限公司

公司电话: 020 - 38271790

公司网址: www.alientek.com

在线教学平台: www.yuanzige.com

开源电子网: www.openedv.com/forum.php

天猫旗舰店: <https://zhengdianyuanzi.tmall.com>

正点原子 B 站: <https://space.bilibili.com/394620890>

扫码关注正点原子公众号, 获取更多嵌入式学习资料

扫码下载原子哥 App, 提供数千讲免费开源视频学习

扫码关注 B 站正点原子官方, 所有视频均可免费在线观看

扫码关注抖音正点原子, 技术与娱乐结合体验双倍快乐



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

目录

前言.....	4
第一章 编译出厂镜像.....	5
1.1 安装ZU+交叉编译工具链.....	6
1.2 拷贝源码到Ubuntu系统.....	7
1.2.1 拷贝u-boot源码.....	7
1.2.2 拷贝内核源码.....	8
1.2.3 拷贝xsa文件.....	9
1.3 编译.....	9
1.4 设置交叉编译工具链的工作环境.....	10
1.4.1 创建Petalinux工程.....	11
1.4.2 编译出厂源码u-boot、制作BOOT.BIN.....	12
1.4.3 编译内核、设备树.....	14
1.5 启动.....	16

前言

出厂镜像文件中所用的 uboot 和内核源码版本与教程中是不一样的。出厂镜像中使用的 uboot 是 2020.1 版本, 内核是 5.4 版本, 教程中 uboot 是 19.1 版本, 内核是 4.19 版本。不一样的原因是因为最初(2021 年)我们打算使用 2020.2 版本的 Vivado 和 Petalinux, 后来发现 Windows 版的 Vivado 有些 bug, 有些 IP 核不能生成, 而 19.2 版本的可以, 所以就换成了 19.2 版本的 Vivado, Petalinux 版本最初还是保持 2020.2 版本, 后来发现加了 mipi 后, 使用 2020.2 版本的 Petalinux 就会出问题, 无法解析 xsa 文件生成相应的设备树, 所以就换成了同 Vivado 版本一致的 Petalinux 版本。由于出厂镜像已经完成了一大半, 所以笔者懒的换, 还是继续使用 2020.2 版本的 linux 和 uboot。至于无法生成设备树, 就用 2019.2 版本的 Petalinux 生成设备树。

在此说明下, 无论是 5.4 版本还是 4.19 版本的内核, 使用上是没有区别的, 只是 Petalinux 对内核版本有限制, 2019.2 的只支持 4.19 版本。有区别的是 uboot, 2020.1 版本的 uboot 需要使用 boot.scr。对于 uboot, 一般使用 Petalinux 中的即可, 没必要深究, 当然了, 有兴趣的有时间的也可以上网找资料研究下。

第一章 编译出厂镜像

本章介绍如何编译出开发板出厂时所烧录的镜像文件, 包括 BOOT.BIN 镜像文件、u-boot 镜像文件以及内核镜像文件和设备树。

1.1 安装 ZU+交叉编译工具链

编译 uboot 和内核源码需要在 linux 系统（虚拟机）中安装相应的交叉编译工具链。安装 ZU+的交叉编译工具链需要 sdk.sh 文件。

把**开发板资料盘 B 盘**: \sdk\202002\sdk.sh 拷贝到 Ubuntu 虚拟机。拷贝完成后, 在虚拟机中切换到 sdk.sh 文件所在目录, 为 sdk.sh 文件赋予可执行权限并安装, 命令如下:

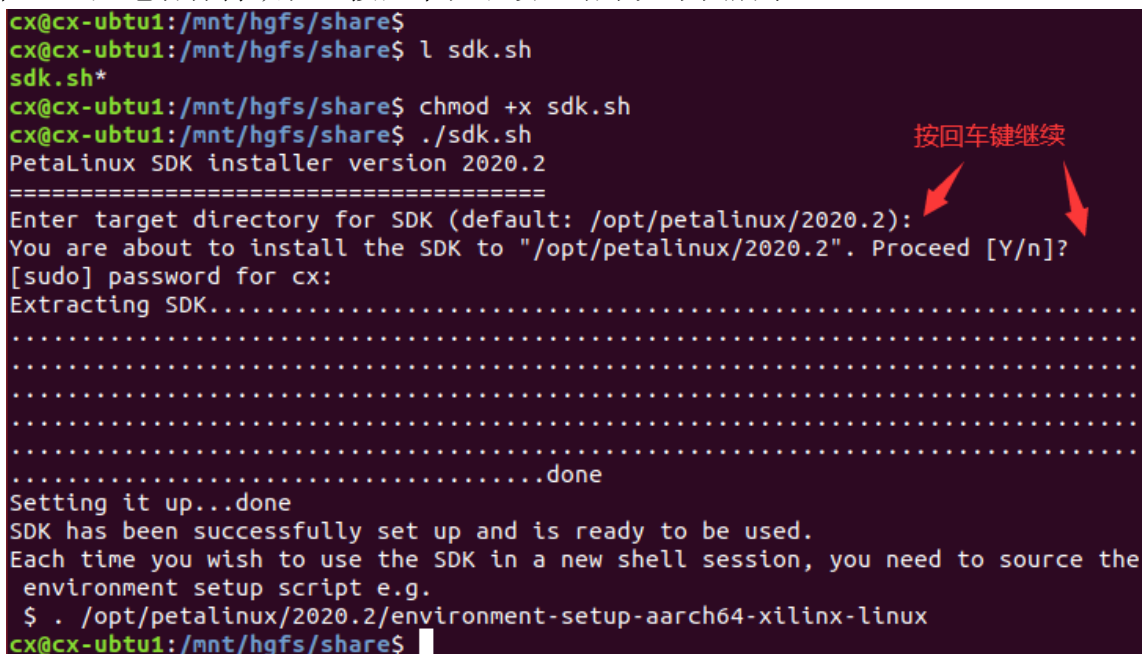
#如果之前没有安装过 build-essential、git, 请先执行如下命令进行安装

```
sudo apt update
sudo apt -y install build-essential git
```

#安装 SDK

```
chmod +x sdk.sh
./sdk.sh
```

默认是安装在/opt/petalinux/2020.2 目录下, 如果想安装在其他目录下, 可以输入相应的路径, 此处笔者保持默认, 按回车键继续, 结果如下图所示:



```
cx@cx-ubtu1:/mnt/hgfs/share$
cx@cx-ubtu1:/mnt/hgfs/share$ l sdk.sh
sdk.sh*
cx@cx-ubtu1:/mnt/hgfs/share$ chmod +x sdk.sh
cx@cx-ubtu1:/mnt/hgfs/share$ ./sdk.sh
Petalinux SDK installer version 2020.2
=====
Enter target directory for SDK (default: /opt/petalinux/2020.2):
You are about to install the SDK to "/opt/petalinux/2020.2". Proceed [Y/n]?
[sudo] password for cx:
Extracting SDK.....done
Setting it up...done
SDK has been successfully set up and is ready to be used.
Each time you wish to use the SDK in a new shell session, you need to source the
environment setup script e.g.
$ . /opt/petalinux/2020.2/environment-setup-aarch64-xilinx-linux
cx@cx-ubtu1:/mnt/hgfs/share$
```

图 1.1.1 安装 SDK

再次确认是否将 SDK 安装在/opt/petalinux/2020.2 目录下, 默认为“Y”, 也就是“是”, 按回车键继续往下执行, 显示要输入用户密码, 输入完成后回车进行安装, 等待安装完成。安装完成后提示每次当你在一个新终端中使用 SDK, 需要先执行“./opt/petalinux/2020.2/environment-setup-aarch64-xilinx-linux”以设置相应的环境变量, 其中的“.”和 source 具有相同含义。

1.2 拷贝源码到 Ubuntu 系统

1.2.1 拷贝 u-boot 源码

开发板出厂镜像使用的 u-boot 源码路径为：**开发板资料盘 (A 盘)\4_SourceCode\3_Embedded_Linux\资源文件\出厂镜像相关**，在该目录下有一个名为 `atk-zup-uboot-xlnx.tar.gz` 的压缩包文件，如下所示：

名称	修改日期	类型	大小
 <code>atk-zup-uboot-xlnx.tar.gz</code>	2022/6/22 15:07	GZ 压缩文件	193,959 KB

图 1.2.1 u-boot 源码

`atk-zup-uboot-xlnx.tar.gz` 是专门用于开发板出厂测试的 u-boot 源码压缩包文件。我们将 `atk-zup-uboot-xlnx.tar.gz` 压缩包文件拷贝到 Ubuntu 系统中，如下所示：

```

○ cx-ubtu in ~
→ l /mnt/hgfs/share/source_code/
total 190M
-rwxrwxrwx 1 root root 190M 6月 22 15:07 atk-zup-uboot-xlnx.tar.gz*
○ cx-ubtu in ~
→ |

```

图 1.2.2 将 u-boot 压缩包文件拷贝到 Ubuntu

接下来将其解压，对应的解压目录就是 U-Boot 源码目录，这个解压目录大家可以自己设置。因为解压后会自动在解压目录中创建一个名为 `u-boot-xlnx` 的文件夹，该文件夹下即是 u-boot 源码。为了避免和教程中使用的 u-boot 相混淆，笔者这里选择将其解压到用户家目录下的 `~/workspace/src` 目录中。

执行如下命令将其解压到 `~/workspace/src/u-boot-xlnx` 目录中：

```

mkdir -p ~/workspace/src/u-boot-xlnx
tar -xzf atk-zup-uboot-xlnx.tar.gz -C ~/workspace/src/u-boot-xlnx    #解压
sync
ls ~/workspace/src/u-boot-xlnx

```

```

○ cx-ubtu in /mnt/hgfs/share/source_code
→ mkdir -p ~/workspace/src/u-boot-xlnx
○ cx-ubtu in /mnt/hgfs/share/source_code
→ tar -xzf atk-zup-uboot-xlnx.tar.gz -C ~/workspace/src/u-boot-xlnx
○ cx-ubtu in /mnt/hgfs/share/source_code
→ sync
○ cx-ubtu in /mnt/hgfs/share/source_code
→ ls ~/workspace/src/u-boot-xlnx
api          common      drivers     include     MAINTAINERS  scripts
arch         config.mk  dts         Kbuild      Makefile     test
atkboot.cmd.default configs    env         Kconfig     net          tools
board       disk       examples    lib         post
cmd         doc        fs          Licenses    README
○ cx-ubtu in /mnt/hgfs/share/source_code

```

图 1.2.3 解压 U-Boot 源码包

解压完成后, u-boot-xlnx 文件夹存放的即是出厂镜像使用的 uboot 源码。

1.2.2 拷贝内核源码

开发板出厂镜像使用的 Linux 内核源码路径为: 开发板资料盘(A 盘)\4_SourceCode\3_Embedded_Linux\资源文件\出厂镜像相关, 在该目录下有一个名为 atk-zup-linux-xlnx.tar.gz 的压缩包文件, 如下所示:

名称	修改日期	类型	大小
 atk-zup-linux-xlnx.tar.gz	2022/6/22 16:33	GZ 压缩文件	2,345,508...

图 1.2.4 内核源码压缩包文件

atk-zup-linux-xlnx.tar.gz 是专门用于开发板出厂测试的 Linux 内核源码压缩包文件。我们将 atk-zup-linux-xlnx.tar.gz 压缩包文件拷贝到 Ubuntu 系统中, 如下所示:

```

cx-ubtu in ~
→ l /mnt/hgfs/share/source_code/
total 2.5G
-rwxrwxrwx 1 root root 2.3G 6月 22 16:33 atk-zup-linux-xlnx.tar.gz*
-rwxrwxrwx 1 root root 190M 6月 22 15:07 atk-zup-uboot-xlnx.tar.gz*
cx-ubtu in ~

```

图 1.2.5 将内核源码压缩包文件拷贝到 Ubuntu

接下来将其解压, 对应的解压目录就是 Linux 内核源码目录, 这个解压目录大家可以自己设置。因为解压后会自动在解压目录中创建一个名为 linux-xlnx 的文件夹, 该文件夹下即是 Linux 内核源码。为了避免和教程中使用的 Linux 内核源码相混淆, 笔者这里选择将其解压到用户家目录下的 ~/workspace/src 目录中:

执行如下命令将其解压到 ~/workspace/src 目录中:

```

mkdir -p ~/workspace/src/linux-xlnx
tar -xzf atk-zup-linux-xlnx.tar.gz -C ~/workspace/src/linux-xlnx #解压
sync

```

```

cx-ubtu in /mnt/hgfs/share/source_code
→ mkdir -p ~/workspace/src/linux-xlnx
cx-ubtu in /mnt/hgfs/share/source_code
→ tar -xzf atk-zup-linux-xlnx.tar.gz -C ~/workspace/src/linux-xlnx/
cx-ubtu in /mnt/hgfs/share/source_code
→ sync
cx-ubtu in /mnt/hgfs/share/source_code

```

图 1.2.6 解压内核源码

解压完成后, 得到的 linux-xlnx 文件夹即是出厂镜像使用的 Linux 内核源码。使用 ls 命令 (ls ~/workspace/src/linux-xlnx), 可看到 Linux 内核源码目录结构, 如下图所示:


```

○ cx-ubtu in /mnt/hgfs/share/source_code
→ ls ~/workspace/src/linux-xlnx
arch      crypto      include  kernel      mm          security
block     Documentation  init     lib          net         sound
certs     drivers       ipc      LICENSES    README     tools
COPYING   fit-image.its  Kbuild   MAINTAINERS  samples    usr
CREDITS   fs            Kconfig  Makefile     scripts    virt
○ cx-ubtu in /mnt/hgfs/share/source_code

```

图 1.2.7 Linux 内核源码目录结构

1.2.3 拷贝 xsa 文件

使用 xsa 文件以创建相应的 Petalinux 工程，
在开发板资料包中，已经给大家提供了开发板出厂时所对应的 vivado 工程。
对于 ATK_DFZU4EV 开发板，使用的是带 VCU 的 base_vcu 工程，
对于 ATK_DFZU2EG 开发板，使用的是 base 工程。

笔者以 ATK_DFZU4EV 开发板为例，路径为：**开发板资料盘 (A 盘)\4_SourceCode\3_Embedded_Linux\vivado_prj\4ev**，在这个目录下有一个压缩包文件 base_vcu.zip，将其在 Windows 系统下解压，解压后如下所示：

isoc > 3_Embedded_Linux > vivado_prj > 4ev > base_vcu					搜索"base_vcu"
名称	修改日期	类型	大小		
base_vcu.cache	2022/6/17 13:56	文件夹			
base_vcu.hw	2022/6/17 13:56	文件夹			
base_vcu.ip_user_files	2022/6/17 13:56	文件夹			
base_vcu.runs	2022/6/17 15:03	文件夹			
base_vcu.sim	2022/6/17 13:56	文件夹			
base_vcu.srcs	2022/6/17 13:56	文件夹			
base_vcu.xpr	2022/6/17 15:38	Vivado Project Fi...	126 KB		
system_wrapper.xsa	2022/6/17 15:41	XSA 文件	3,383 KB		

图 1.2.8 vivado 工程目录

Vivado 导出的 xsa 文件名为 system_wrapper.xsa，这里直接将 system_wrapper.xsa 文件夹拷贝到 Ubuntu 系统某个目录下，例如 /mnt/hgfs/share/xsa/4ev/，如下所示：

```

○ cx-ubtu in ~
→ l /mnt/hgfs/share/xsa/4ev/
total 3.4M
-rwxrwxrwx 1 root root 3.4M 6月 17 15:41 system_wrapper.xsa*
○ cx-ubtu in ~

```

图 1.2.9 将 xsa 文件拷贝到 Ubuntu

1.3 编译

在上一节中我们已经将编译所需要的所有“原材料”拷贝到 Ubuntu 系统了，接下来可以进行编译了，需要注意的是，在编译之前，需要安装 xilinx 的 petalinux 工具，以及设置交

交叉编译工具链的工作环境。如果还没安装 petalinux 的, 可以参考【正点原子】DFZU2EG_4EV MPSoC 嵌入式 Linux 驱动开发指南 V1.x.pdf 的第 5 章内容 Petalinux 的安装进行安装; 设置交叉编译工具链的工作环境请参考下一小节。

1.4 设置交叉编译工具链的工作环境

每打开一个新终端, 都需要在终端中执行如下命令设置 SDK 的环境变量以使用交叉编译器:

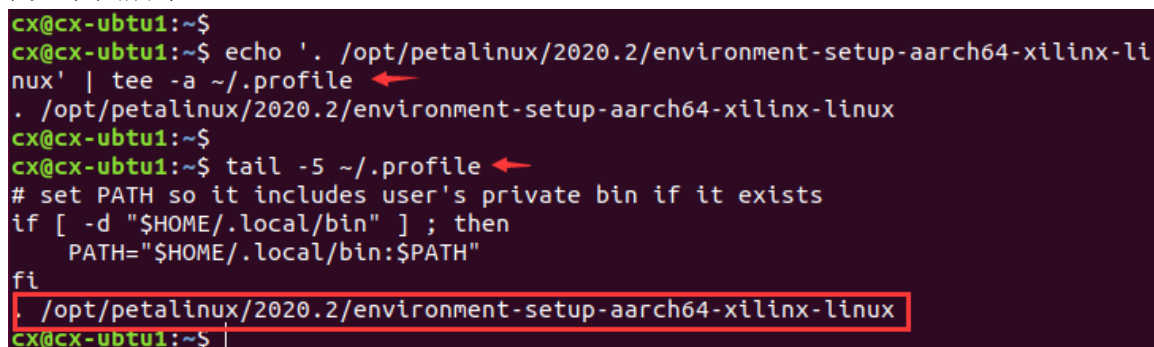
```
. /opt/petalinux/2020.2/environment-setup-aarch64-xilinx-linux
```

如果不想每打开一个新终端就执行这个命令, 可以将其放入 ~/.profile 或者 /etc/profile 文件中, 命令如下:

```
echo '. /opt/petalinux/2020.2/environment-setup-aarch64-xilinx-linux' | tee -a ~/.profile
```

```
tail -5 ~/.profile
```

结果如下图所示:



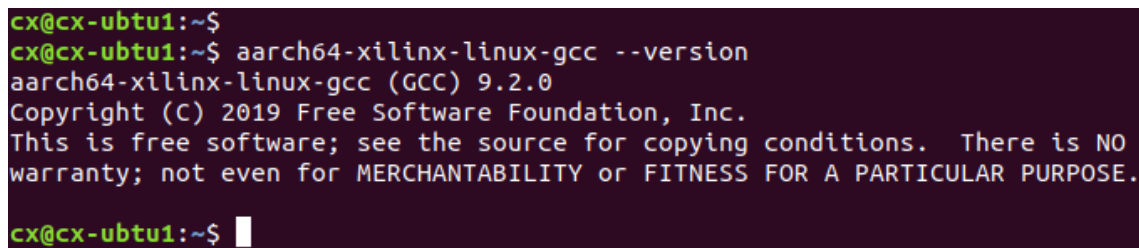
```
cx@cx-ubtu1:~$  
cx@cx-ubtu1:~$ echo '. /opt/petalinux/2020.2/environment-setup-aarch64-xilinx-linux' | tee -a ~/.profile  
. /opt/petalinux/2020.2/environment-setup-aarch64-xilinx-linux  
cx@cx-ubtu1:~$  
cx@cx-ubtu1:~$ tail -5 ~/.profile  
# set PATH so it includes user's private bin if it exists  
if [ -d "$HOME/.local/bin" ]; then  
    PATH="$HOME/.local/bin:$PATH"  
fi  
. /opt/petalinux/2020.2/environment-setup-aarch64-xilinx-linux  
cx@cx-ubtu1:~$
```

图 1.4.1 加载 ~/.profile 文件时设置 SDK 环境变量

这样, 将命令 “. /opt/petalinux/2020.2/environment-setup-aarch64-xilinx-linux” 写入到 ~/.profile 文件中, 启动虚拟机登录当前用户后会自动执行该命令。不过当前终端不可用, 需要重启才能生效。

注: 如果读者之前用相同的方式设置过 Petalinux2019.2 版本的 SDK, 请先从 ~/.profile 等文件中删除相关的行。

设置 SDK 的环境变量后, 在终端输入命令 aarch64-xilinx-linux-gcc --version 来查看当前使用的交叉编译器版本号。看到如下结果, 表明环境变量已经生效。



```
cx@cx-ubtu1:~$  
cx@cx-ubtu1:~$ aarch64-xilinx-linux-gcc --version  
aarch64-xilinx-linux-gcc (GCC) 9.2.0  
Copyright (C) 2019 Free Software Foundation, Inc.  
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  
cx@cx-ubtu1:~$
```

图 1.4.2 查看 gcc 版本信息

可以看到, 我们使用的 ZU+交叉编译器版本为 9.2.0, 也就是用的 2020.2 版本的 SDK。设置环境变量后可以使用 env 命令查看生效的环境变量, 下图为部分截图:

```
OE_QMAKE_AR=aarch64-xilinx-linux-ar
KCFLAGS=-sysroot=/opt/petalinux/2020.2/sysroots/aarch64-xilinx-linux
OECORE_ACLOCAL_OPTS=-I /opt/petalinux/2020.2/sysroots/x86_64-petalinux-linux/usr/share/a
clocal
CC=aarch64-xilinx-linux-gcc -march=armv8-a+crc -mtune=cortex-a72.cortex-a53 --sysroot=/
opt/petalinux/2020.2/sysroots/aarch64-xilinx-linux
READELF=aarch64-xilinx-linux-readelf
LC_NAME=zh_CN.UTF-8
XDG_SESSION_ID=1
OECORE_DISTRO_VERSION=2020.2
OECORE_SDK_VERSION=2020.2
```

图 1.4.3 查看设置后的环境变量

不同的环境变量有不同的作用。比如环境变量 CC, 可以看出该环境变量已经配置好 gcc 交叉编译器编译时所用的参数, 比如使用 aarch64-xilinx-linux-gcc 编译器, 可直接使用环境变量 \$CC, 如交叉编译某个 .c 文件, 如 hello.c, 直接使用 \$CC hello.c 就可以了。

1.4.1 创建 Petalinux 工程

创建 Petalinux 工程的步骤在【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南 V1.x.pdf 第六章 Petalinux 设计流程实战中已讲解, 本章就不细述, 也可直接使用第六章 Petalinux 设计流程实战的 Petalinux 工程, 重新导入 xsa 文件即可。

先在 Ubuntu 主机终端中选一个合适的路径以创建出厂镜像使用的 Petalinux 工程, 然后在终端中输入如下命令:

```
cd <相应的目录> #切换到创建 Petalinux 工程的目录
sptl #设置 petalinux 工作环境
petalinux-create -t project --template zynqMP -n base #创建 Petalinux 工程
cd base #进入到 petalinux 工程目录下
petalinux-config --get-hw-description /mnt/hgfs/share/xsa/4ev #导入相应的
xsa 文件
```

以上命令执行完成之后会自动在当前目录下创建一个名为 base 的文件夹, 也就是我们出厂镜像的 petalinux 工程对应的工程目录。

xsa 文件导入成功之后会自动弹出 petalinux 工程配置窗口, 如下图所示:

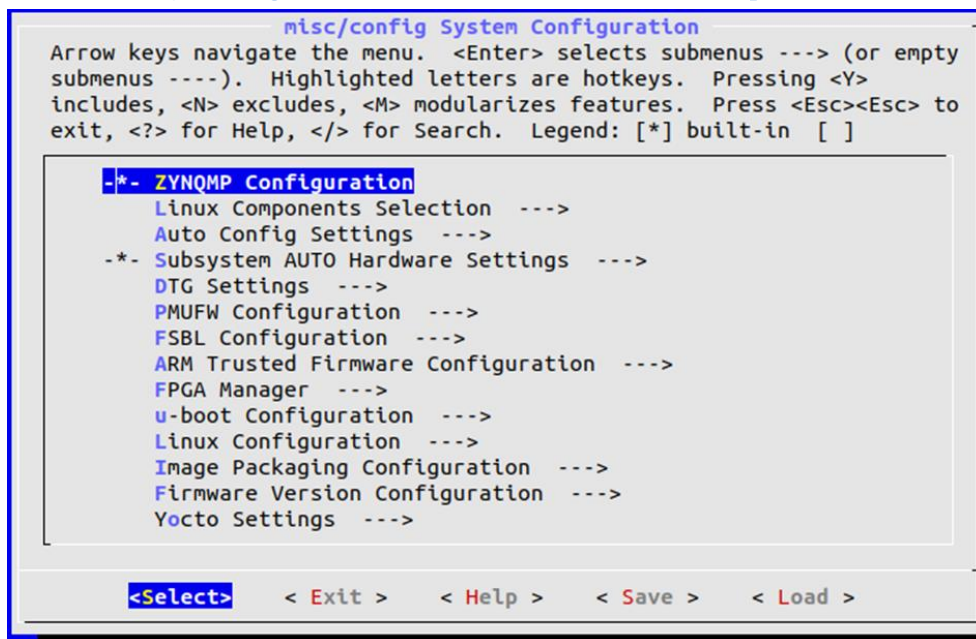


图 1.4.4 petalinux 工程配置窗口

使用默认配置即可，保存并退出。

1.4.2 编译出厂源码 u-boot、制作 BOOT.BIN

进入到 u-boot 源码目录下。

正点原子的 ZU+ MPSoC 开发板对应的配置文件为：
configs/xilinx_zynqmp_virt_defconfig

正点原子的 ZU+开发板对应的设备树文件为：arch/arm/dts/zynqmp-atk.dts

执行如下命令编译 u-boot 源码：

```
make distclean          # 清理工程
make xilinx_zynqmp_virt_defconfig #配置 uboot
make -j                  # 编译
```

```

± |u-boot-xlnx ✓| cx-ubtu in ~/workspace/src/u-boot-xlnx
→ make distclean
± |u-boot-xlnx ✓| cx-ubtu in ~/workspace/src/u-boot-xlnx
→ make xilinx_zynqmp_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
#
± |u-boot-xlnx ✓| cx-ubtu in ~/workspace/src/u-boot-xlnx
→ 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

```

图 1.4.5 编译 u-boot 源码

编译成功之后, 会在 u-boot 目录下生成镜像文件, 如下所示:

```

± |u-boot-xlnx ✓| cx-ubtu in ~/workspace/src/u-boot-xlnx
→ ls
api                drivers            MAINTAINERS        tools              u-boot-elf.o
arch               dts               Makefile           u-boot             u-boot.img
atkboot.cmd.default env               net                u-boot.bin         u-boot.itb
board              examples          post               u-boot.cfg         u-boot.its
cmd                fs                README             u-boot.cfg.configs u-boot.lds
common             include           scripts            u-boot.dtb         u-boot.map
config.mk          Kbuild           singletask.lock    u-boot-dtb.bin     u-boot-nodtb.bin
configs            Kconfig          spl                u-boot-dtb.img     u-boot.srec
disk              lib              System.map         u-boot-elf.lds     u-boot.sym
doc                Licenses         test
± |u-boot-xlnx ✓| cx-ubtu in ~/workspace/src/u-boot-xlnx

```

图 1.4.6 生成镜像文件

u-boot.elf 文件是一个 elf 格式文件。

输入如下命令生成 boot.scr 文件:

```
./tools/mkimage -c none -A arm -T script -d atkboot.cmd.default boot.scr
```

```

± |u-boot-xlnx ✓| cx-ubtu in ~/workspace/src/u-boot-xlnx
→ ./tools/mkimage -c none -A arm -T script -d atkboot.cmd.default boot.scr
Image Name:
Created:      Mon Jun 27 14:04:45 2022
Image Type:   ARM Linux Script (gzip compressed)
Data Size:    3410 Bytes = 3.33 KiB = 0.00 MiB
Load Address: 00000000
Entry Point:  00000000
Contents:
  Image 0: 3402 Bytes = 3.32 KiB = 0.00 MiB
± |u-boot-xlnx ✓| cx-ubtu in ~/workspace/src/u-boot-xlnx

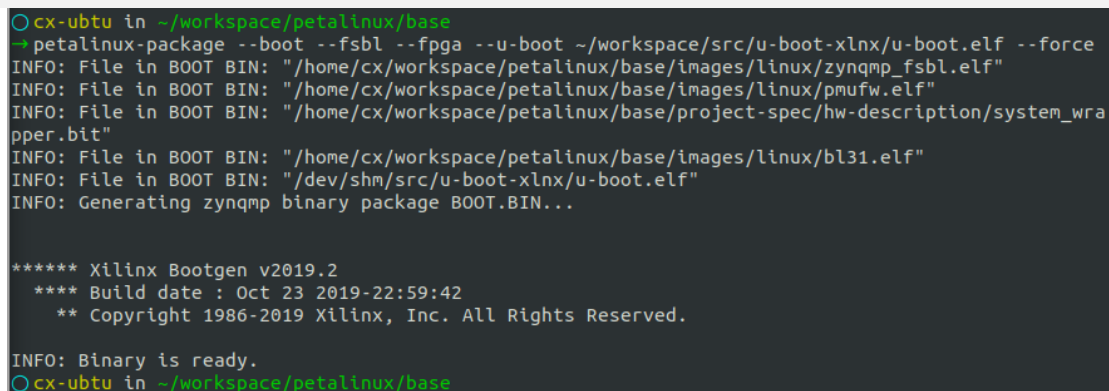
```

图 1.4.7 生成 boot.scr

boot.scr 是出厂镜像使用的 uboot 用于从相应的存储设备加载内核镜像和设备树的脚本文件。

切换到创建的 Petalinux 工程目录, 执行下面的命令制作 ZU+ MPSoC 开发板所需的启动文件 BOOT.BIN:

```
petalinux-build -c bootloader #生成 fsbl.elf 文件
petalinux-build -c pmufw
petalinux-build -c arm-trusted-firmware
petalinux-package --boot --fsbl --fpga --u-boot ~/workspace/src/u-boot-xlnx/u-boot.elf --force
```



```

cx-ubuntu in ~/workspace/petalinux/base
→ petalinux-package --boot --fsbl --fpga --u-boot ~/workspace/src/u-boot-xlnx/u-boot.elf --force
INFO: File in BOOT BIN: "/home/cx/workspace/petalinux/base/images/linux/zynqmp_fsbl.elf"
INFO: File in BOOT BIN: "/home/cx/workspace/petalinux/base/images/linux/pmufw.elf"
INFO: File in BOOT BIN: "/home/cx/workspace/petalinux/base/project-spec/hw-description/system_wra
pper.bit"
INFO: File in BOOT BIN: "/home/cx/workspace/petalinux/base/images/linux/bl31.elf"
INFO: File in BOOT BIN: "/dev/shm/src/u-boot-xlnx/u-boot.elf"
INFO: Generating zynqmp binary package BOOT.BIN...

***** Xilinx Bootgen v2019.2
**** Build date : Oct 23 2019-22:59:42
** Copyright 1986-2019 Xilinx, Inc. All Rights Reserved.

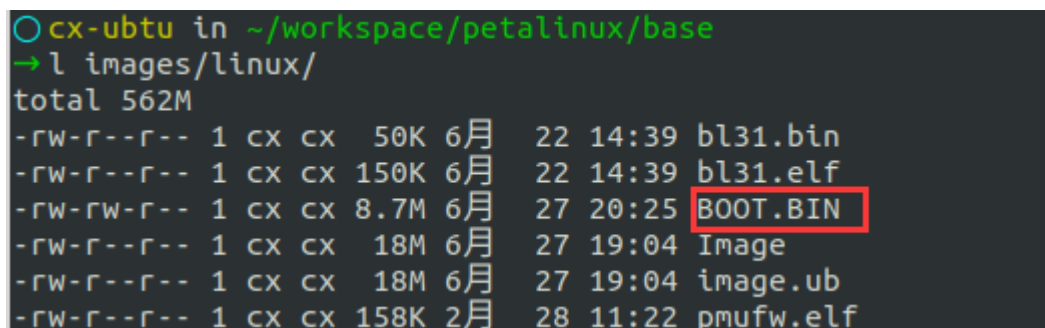
INFO: Binary is ready.
cx-ubuntu in ~/workspace/petalinux/base

```

图 1.4.8 制作 BOOT.BIN 文件

注: 如果编译报设备树相关的错误, 请按照第六章 Petalinux 设计流程实战那样配置设备树, 我们不用 Petalinux 编译的设备树, 只是解决错误。

命令执行成功之后, 会在当前 Petalinux 工程的 images/linux 目录下生成 BOOT.BIN 启动文件, 如下图所示:



```

cx-ubuntu in ~/workspace/petalinux/base
→ l images/linux/
total 562M
-rw-r--r-- 1 cx cx 50K 6月 22 14:39 bl31.bin
-rw-r--r-- 1 cx cx 150K 6月 22 14:39 bl31.elf
-rw-rw-r-- 1 cx cx 8.7M 6月 27 20:25 BOOT.BIN
-rw-r--r-- 1 cx cx 18M 6月 27 19:04 Image
-rw-r--r-- 1 cx cx 18M 6月 27 19:04 image.ub
-rw-r--r-- 1 cx cx 158K 2月 28 11:22 pmufw.elf

```

图 1.4.9 生成 BOOT.BIN 文件

1.4.3 编译内核、设备树

进入到 Linux 内核源码目录下。

内核 defconfig 配置文件为: arch/arm64/configs/xilinx_zynqmp_defconfig

ATK_DFZU4EV 开发板对应的设备树文件为: arch/arm64/boot/dts/xilinx/atk-dfzu4ev.dts

ATK_DFZU2EG 开发板对应的设备树文件为: arch/arm64/boot/dts/xilinx/atk-dfzu2eg.dts

执行如下命令编译内核源码:

```
make distclean # 清理工程
make xilinx_zynqmp_defconfig # 配置
```


make -j8 # 编译 Image 和设备树

```
± |xlnx_rebase_v5.4 ↑17 ✓| cx-ubuntu in ~/workspace/src/linux-xlnx
→ make distclean
± |xlnx_rebase_v5.4 ↑17 ✓| cx-ubuntu in ~/workspace/src/linux-xlnx
→ make xilinx_zynqmp_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
#
± |xlnx_rebase_v5.4 ↑17 ✓| cx-ubuntu in ~/workspace/src/linux-xlnx
→ make -j8
WRAP arch/arm64/include/generated/uapi/asm/kvm para.h
```

图 1.4.10 编译内核源码

编译得到的内核镜像文件 Image 在 arch/arm64/boot 目录下, 如下图所示:

```
± |xlnx_rebase_v5.4 ↑17 ✓| cx-ubuntu in ~/workspace/src/linux-xlnx
→ l arch/arm64/boot
total 24M
drwxr-xr-x 31 cx cx 640 1月 26 17:28 dts/
-rwxrwxr-x 1 cx cx 17M 6月 27 14:38 Image*
-rw-rw-r-- 1 cx cx 7.7M 6月 27 14:38 Image.gz
-rw-r--r-- 1 cx cx 1.6K 1月 26 17:28 install.sh
-rw-r--r-- 1 cx cx 1.2K 1月 26 17:28 Makefile
± |xlnx_rebase_v5.4 ↑17 ✓| cx-ubuntu in ~/workspace/src/linux-xlnx
```

图 1.4.11 生成的内核镜像文件

编译得到的设备树文件在 arch/arm64/boot/dts/xilinx 目录下, 如下图所示:

```
± |xlnx_rebase_v5.4 ↑17 ✓| cx-ubuntu in ~/workspace/src/linux-xlnx
→ l arch/arm64/boot/dts/xilinx/
total 436K
-rw-rw-r-- 1 cx cx 43K 6月 27 14:35 atk-dfzu2eg.dtb
-rw-r--r-- 1 cx cx 703 5月 21 13:46 atk-dfzu2eg.dts
-rw-r--r-- 1 cx cx 14K 6月 20 18:46 atk-dfzu2eg-pl.dtsi
-rw-rw-r-- 1 cx cx 43K 6月 27 14:35 atk-dfzu4ev.dtb
-rw-r--r-- 1 cx cx 741 5月 21 13:54 atk-dfzu4ev.dts
-rw-r--r-- 1 cx cx 15K 6月 17 15:55 atk-dfzu4ev-pl.dtsi
-rw-r--r-- 1 cx cx 386 1月 26 17:28 avnet-ultra96-rev1.dts
-rw-r--r-- 1 cx cx 180 5月 21 13:59 Makefile
-rw-r--r-- 1 cx cx 3.0K 5月 18 09:59 pcw.dtsi
-rw-r--r-- 1 cx cx 8.5K 6月 17 14:31 system-user.dtsi
```

图 1.4.12 设成的设备树文件

atk-dfzu4ev.dtb 对应 ATK_DFZU4EV 开发板的设备树文件;

atk-dfzu2eg.dtb 对应 ATK_DFZU2EG 开发板的设备树文件。

至此, 启动开发板所需所有镜像文件都已经编译完成了, 包括 BOOT.BIN 文件、boot.scr、Image、设备树 dtb 四个文件。

1.5 启动

制作一张 SD 启动卡, SD 启动卡的制作方法, 可以参考【正点原子】DFZU2EG_4EV MPSoC 嵌入式 Linux 驱动开发指南 V1. x. pdf 第六章 Petalinux 设计流程实战中的制作 SD 启动卡小节。这里不再重述!

将上一节编译得到的四个镜像文件拷贝到 SD 启动卡的第一个分区, 也就是 Fat32 分区, 其中

BOOT.BIN 文件位于 Petalinux 工程的 images/linux 目录下;

boot.scr 位于 uboot 源码根目录下;

Image 文件位于 linux 内核源码根目录下的 arch/arm64/boot/目录下;

dtb 文件位于 linux 内核源码根目录下的 arch/arm64/boot/dts/xilinx 目录下。

拷贝完成后如下所示:

```

cx-ubtu in ~
→ cd /media/cx/boot/
cx-ubtu in /media/cx/boot
→ ls
atk-dfzu2eg.dtb  atk-dfzu4ev.dtb  BOOT.BIN  boot.scr  Image
cx-ubtu in /media/cx/boot
  
```

图 1.5.1 SD 启动卡 Fat 分区文件

注: 一般只需要拷贝开发板对应的设备树 dtb 文件就可以了, 拷贝多个设备树文件也是可以的, 启动过程中只用到开发板对应的设备树 dtb 文件。

接下来我们要拷贝根文件系统到 SD 启动卡第二个分区, 在开发板资料包中已经给大家提供了出厂时使用的根文件系统, 在开发板资料盘(A 盘)\4_SourceCode\3_Embedded_Linux\资源文件\出厂镜像相关目录下有一个名为 rootfs.tar.gz 的压缩包文件, 如下所示:




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...

图 1.5.2 根文件系统

rootfs.tar.gz 中包含的根文件系统是使用 Petalinux 编译出来的, 也是出厂镜像使用的根文件系统, 由于测试需要, 笔者也移植了一些软件到其中。该根文件系统比较大、内容比较多, 包括 Qt5 库文件、Python、opencv 等常用库文件。

将 rootfs.tar.gz 拷贝到 Ubuntu 系统中, 然后使用 tar 命令将 rootfs.tar.gz 压缩包文件解压到 SD 启动卡的第二个分区, 使如下所示:

```

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
```

图 1.5.3 解压根文件系统到 SD 卡第二分区

sync 同步完成之后, 卸载 SD 卡, 拔掉 SD 卡将其插入开发板 SD 卡卡槽, 将开发板启动方式设置为 SD 卡启动, 连接串口、LCD 屏和电源启动开发板。