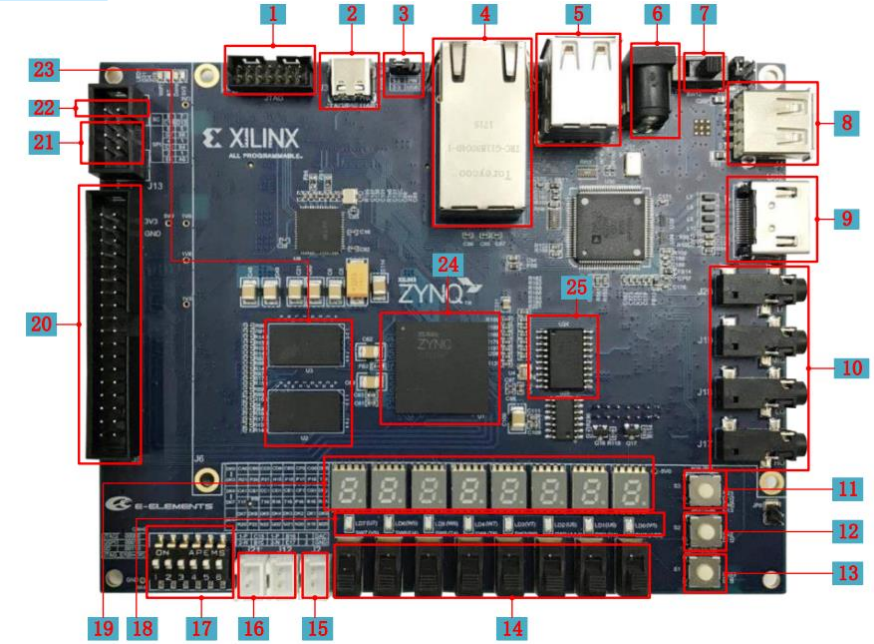
实验五
Linux系统移植
—SD卡启动操作系统

实验硬件环境

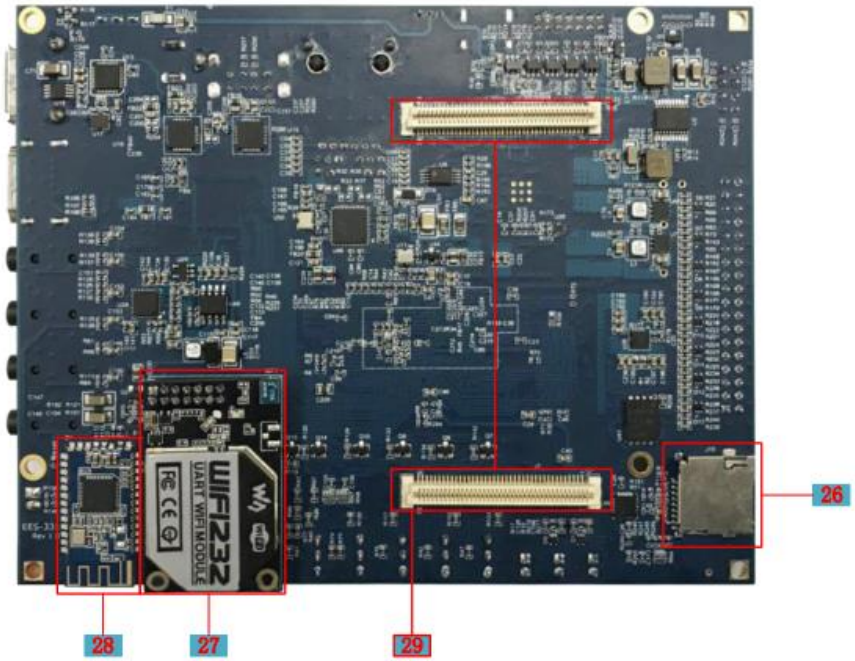


本次实验用到：

2：JTAG/PS_UART/电源

7：电源开关

26：SD卡插槽



还需用到：

SD卡

读卡器

一、实验目的与实验工具

实验目的

了解linux系统启动流程

学习生成linux系统启动文件

熟悉linux系统移植

实验工具

硬件：依元素开发板EES331，JTAG/PS_UART/电源线

SD卡，读卡器

软件：Windows操作系统，Xilinx Vivado&SDK软件，Linux虚拟机环境

二、实验原理与实验内容

实验原理

嵌入式 Linux 系统从软件角度看可以分为四个部分：引导加载程序（Bootloader），Linux 内核，文件系统，应用程序。

其中 Bootloader是系统启动或复位以后执行的第一段代码，用来初始化处理器及外设，然后调用 Linux 内核。Linux 内核在完成系统的初始化之后需要挂载某个文件系统做为根文件系统（Root Filesystem）。根文件系统是 Linux 系统的核心组成部分，它可以做为Linux 系统中文件和数据的存储区域，通常它还包括系统配置文件和运行应用软件所需要的库。应用程序实现的功能通常就是设计该嵌入式系统所要达到的目标。如果没有应用程序的支持，任何硬件上设计精良的嵌入式系统都没有实用意义。

实验内容

使用Vivado&SDK和linux虚拟机环境，生成linux系统启动所需文件，将启动文件拷入SD卡，设置SD卡启动方式，启动linux系统

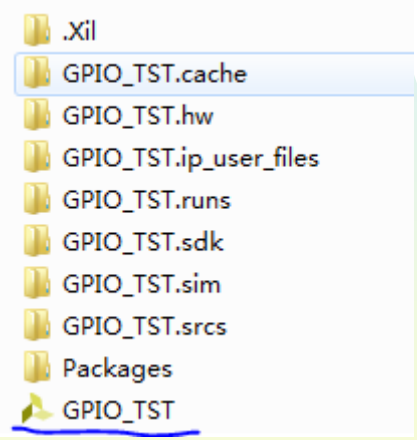
三、实验步骤

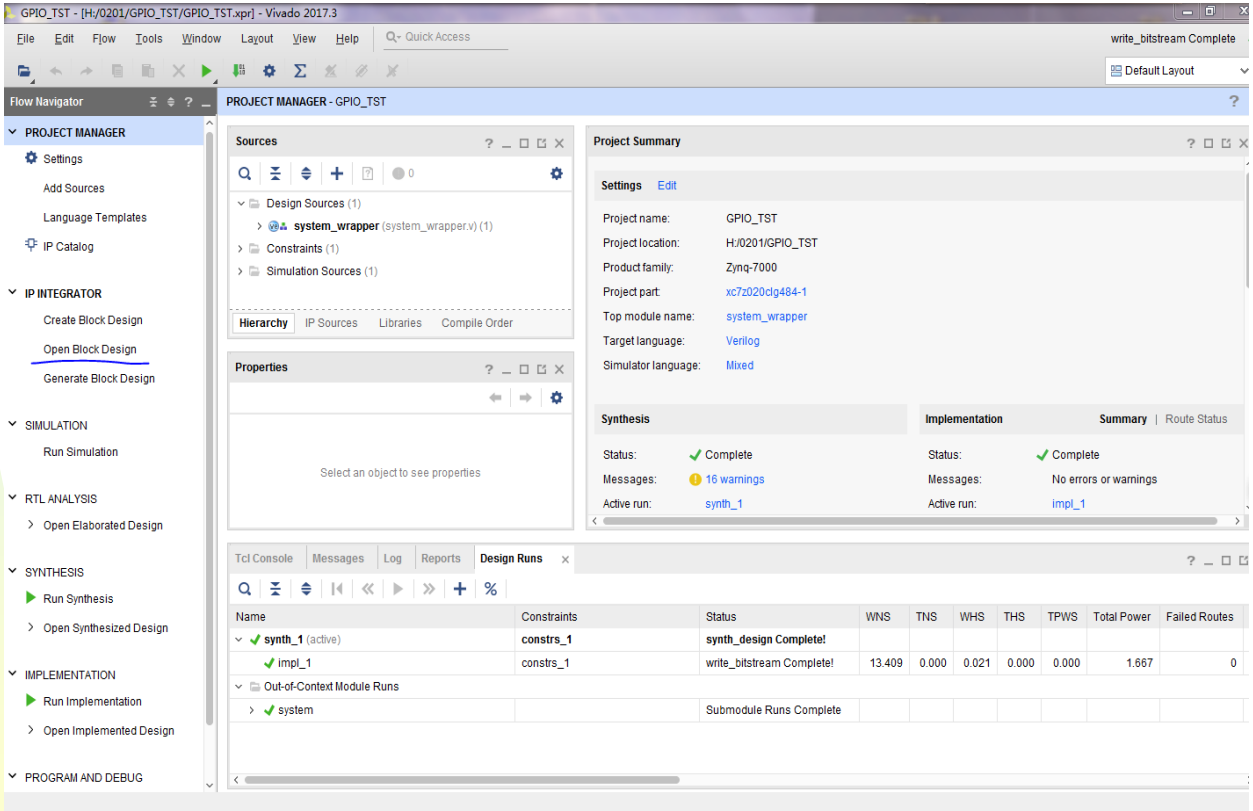
操作步骤概述

1. 打开硬件工程SD0接口并生成比特流（BIT文件）
2. 生成一级启动文件fsbl.elf
3. 生成二级启动文件u-boot.elf
4. 生成BOOT.bin
5. 生成内核文件uImage
6. 生成设备树devicetree.dtb
7. SD卡启动Linux操作系统

1.打开SD0接口生成比特流

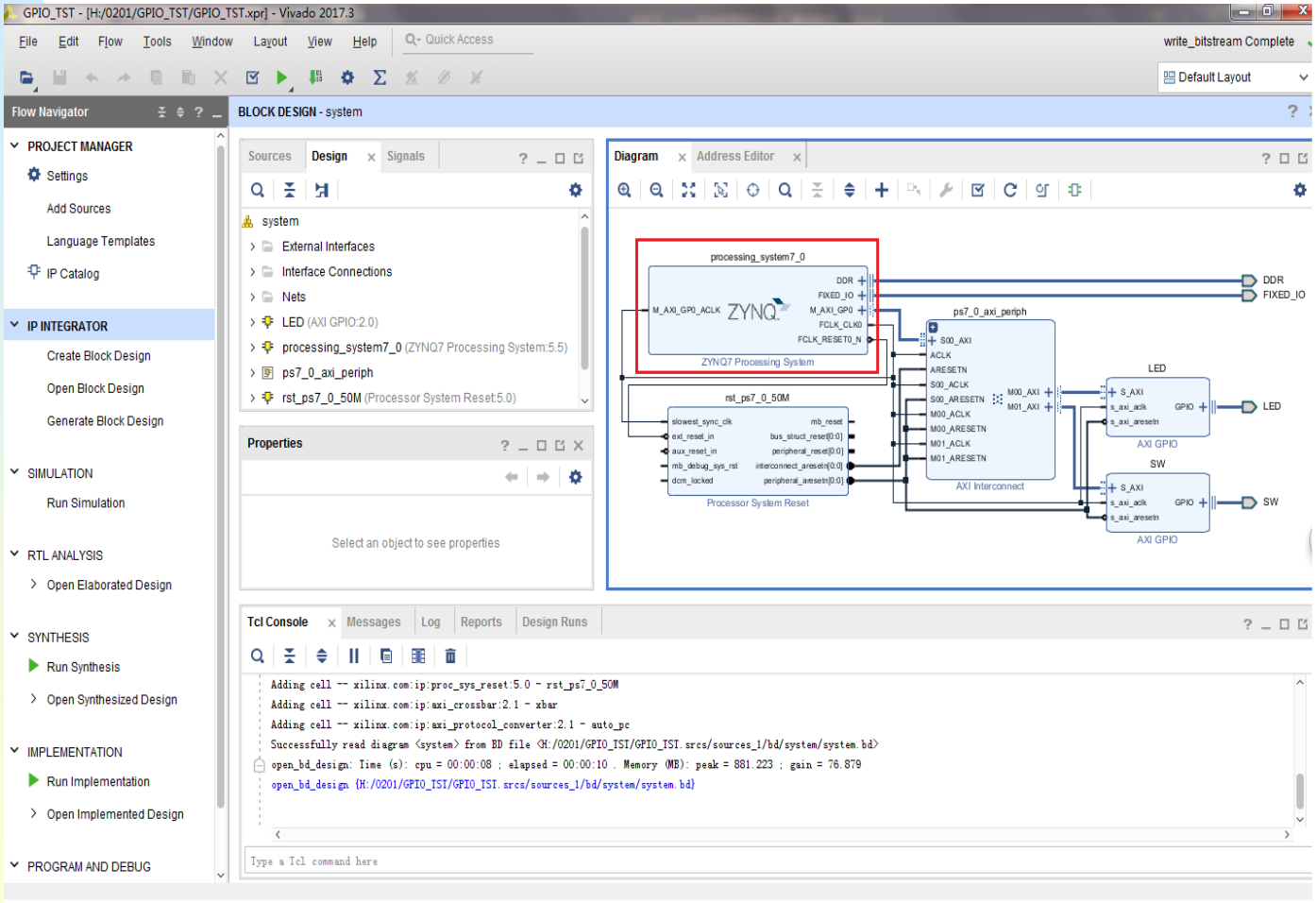
将GPIO_TST工程拷到无中文路径下解压，双击打开工程，点击工程左侧Open Block Design，打开块设计





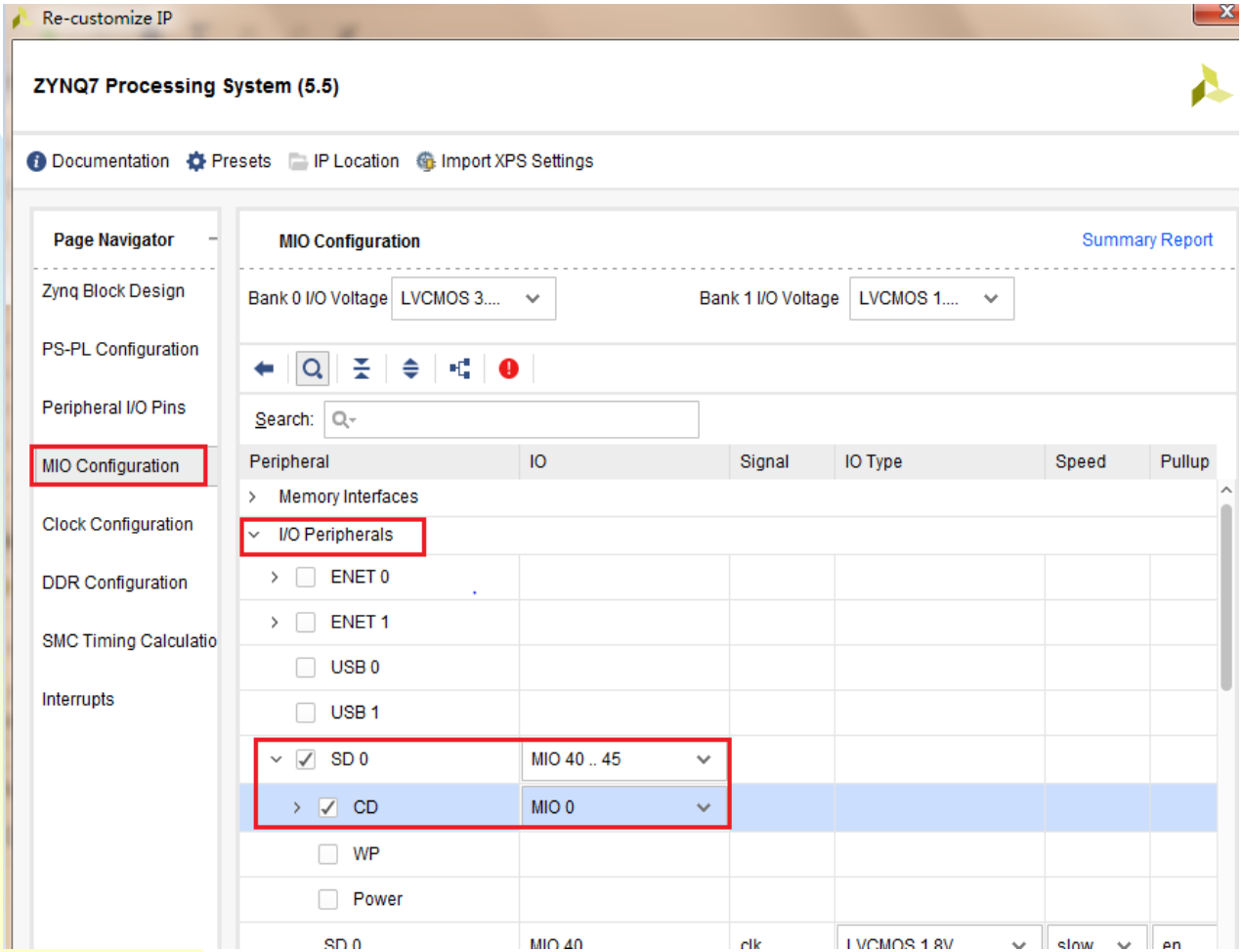
1.打开SD0接口生成比特流

双击打开ZYNQ处理器核，对SD0接口进行配置



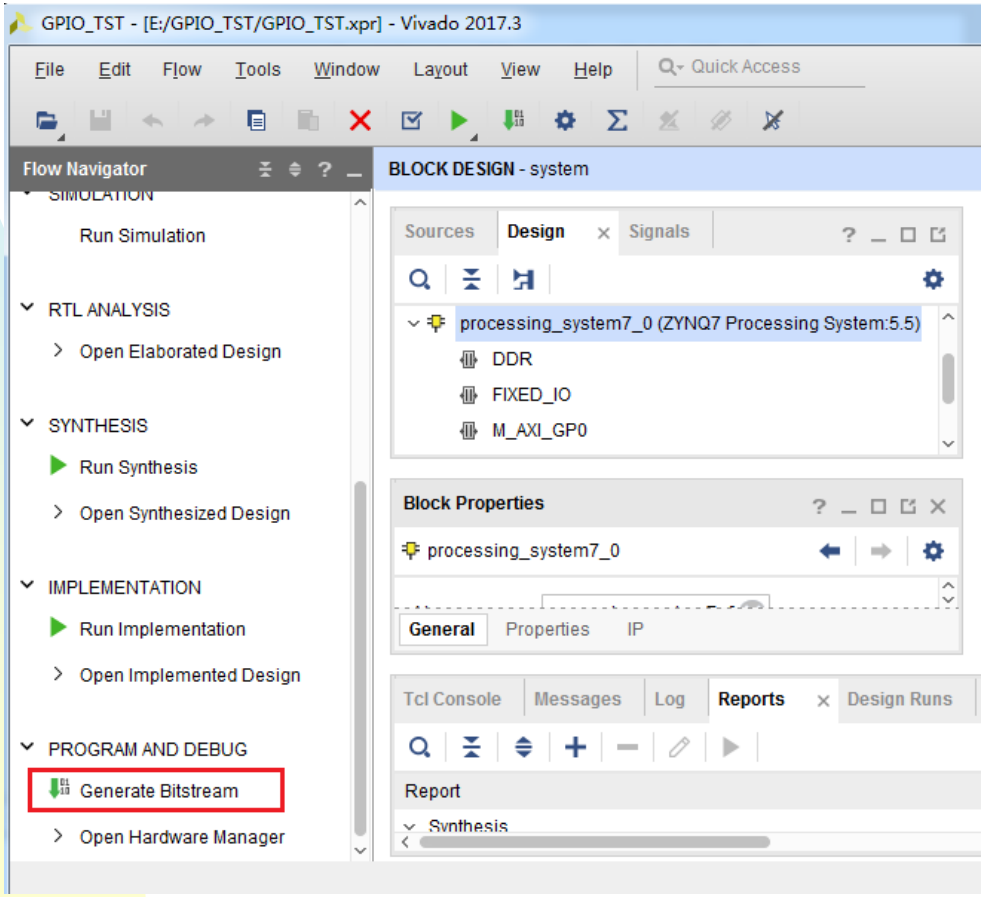
1.打开SD0接口生成比特流

打开SD0接口，SD0接口对应的是SD卡，点击OK退出并Ctrl+S保存



1.打开SD0接口生成比特流

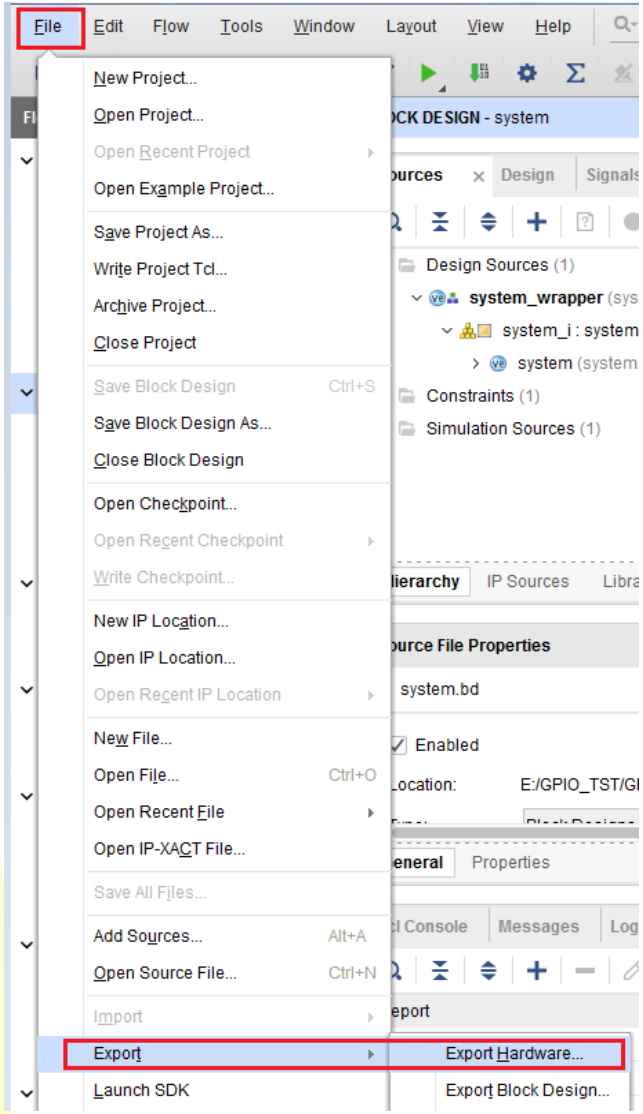
点击左下Generate Bitstream，等待生成比特流，若不熟练请参照实验二PPT操作



新建boot_files文件夹，用于存放生成的启动文件，并将生成的比特流放至boot_files，生成的比特流的路径为GPIO_TST\GPIO_TST.runs\impl_1\xxxx.BIT

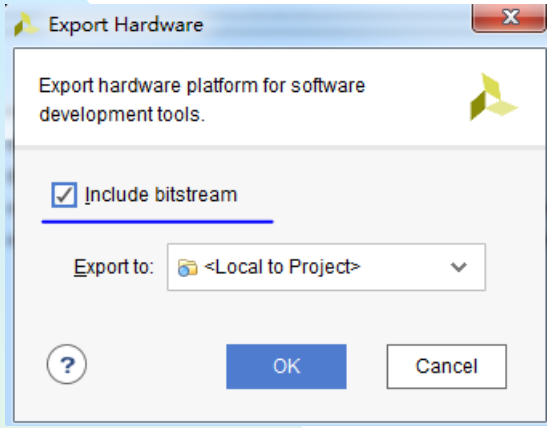
1.打开SD0接口生成比特流

打开File → Export Hardware Design to SDK，导出硬件到SDK

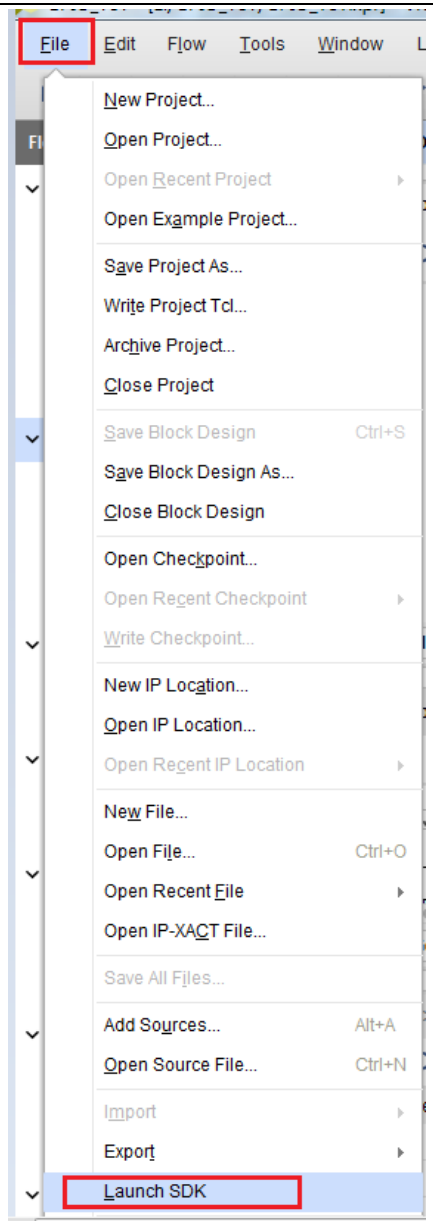


1.打开SD0接口生成比特流

勾选包括比特流

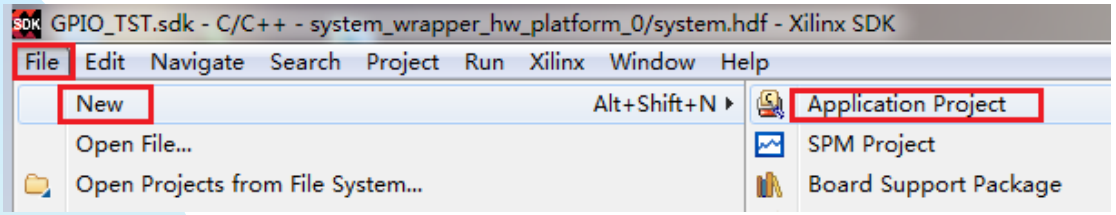


打开SDK： File → Launch SDK → OK，进入SDK界面

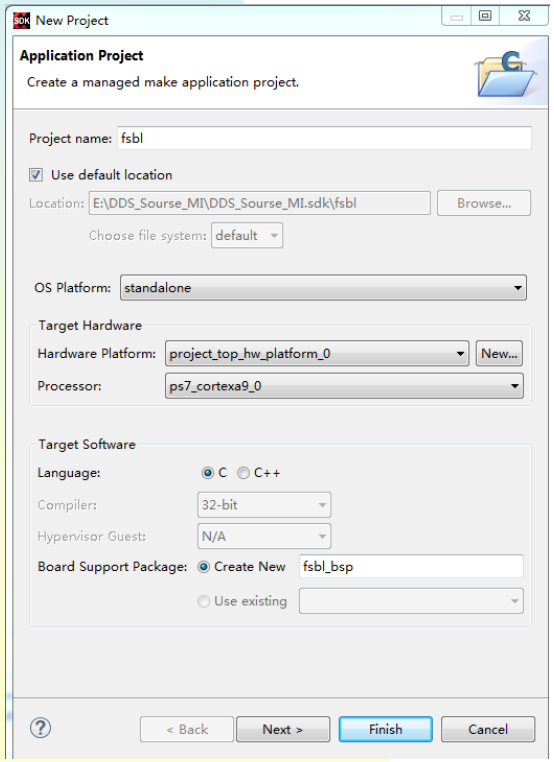


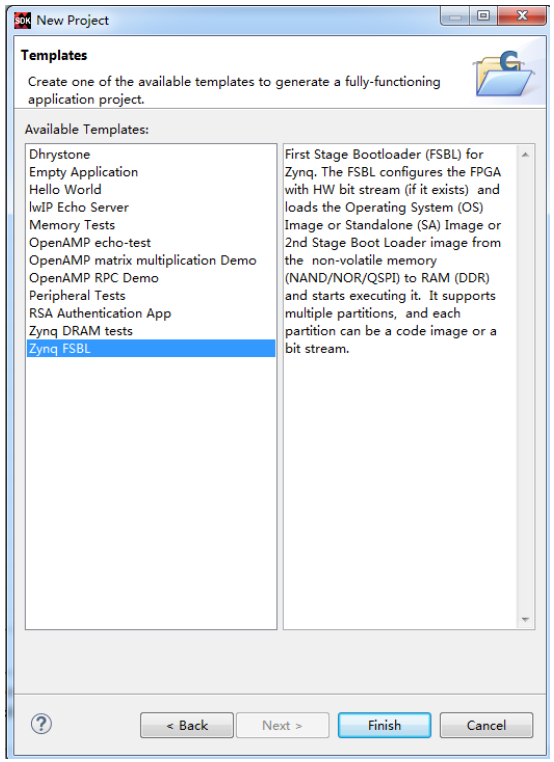
2.生成一级启动文件fsbl.elf

在SDK中新建工程



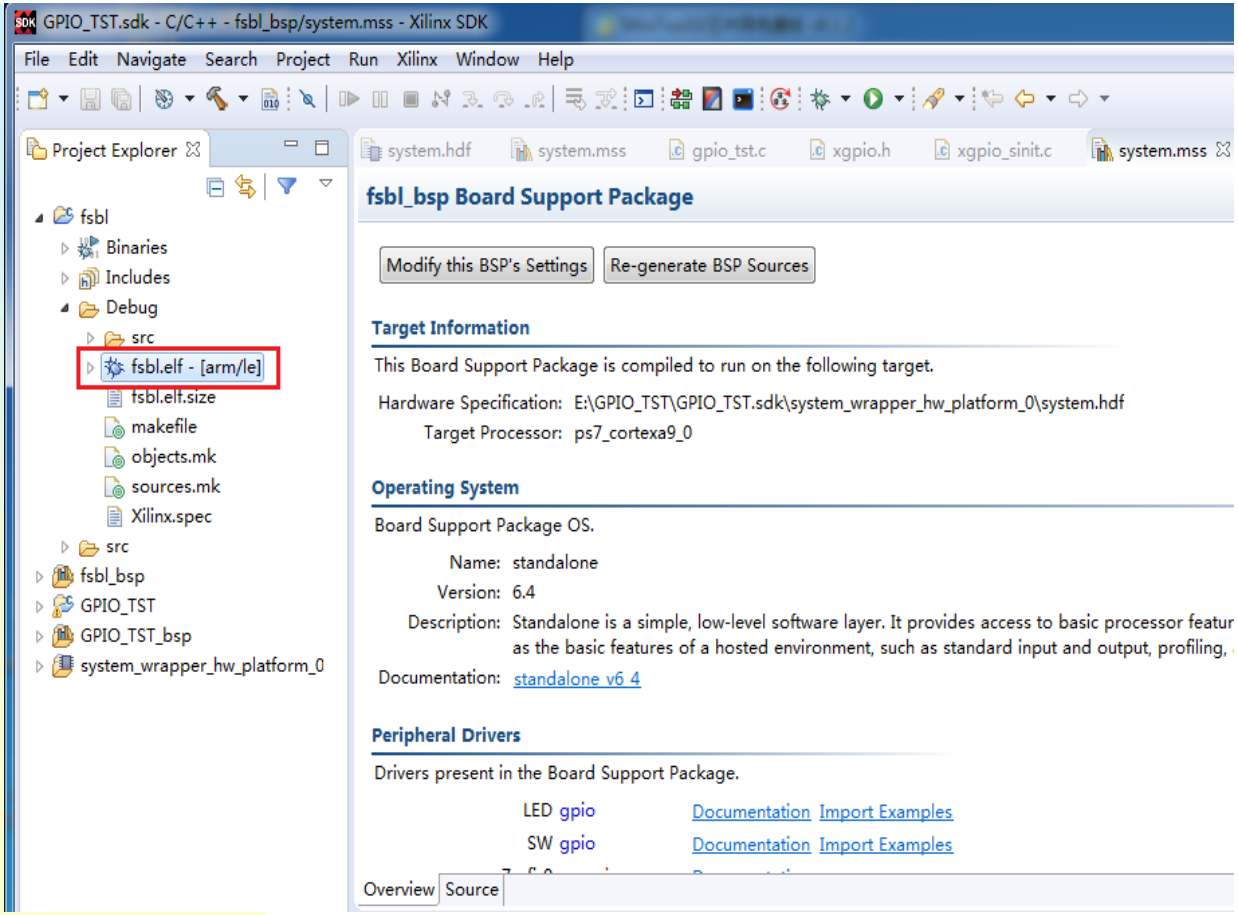
为新工程命名fsbl，点击Next，选择Zynq FSBL模板，点击Finish





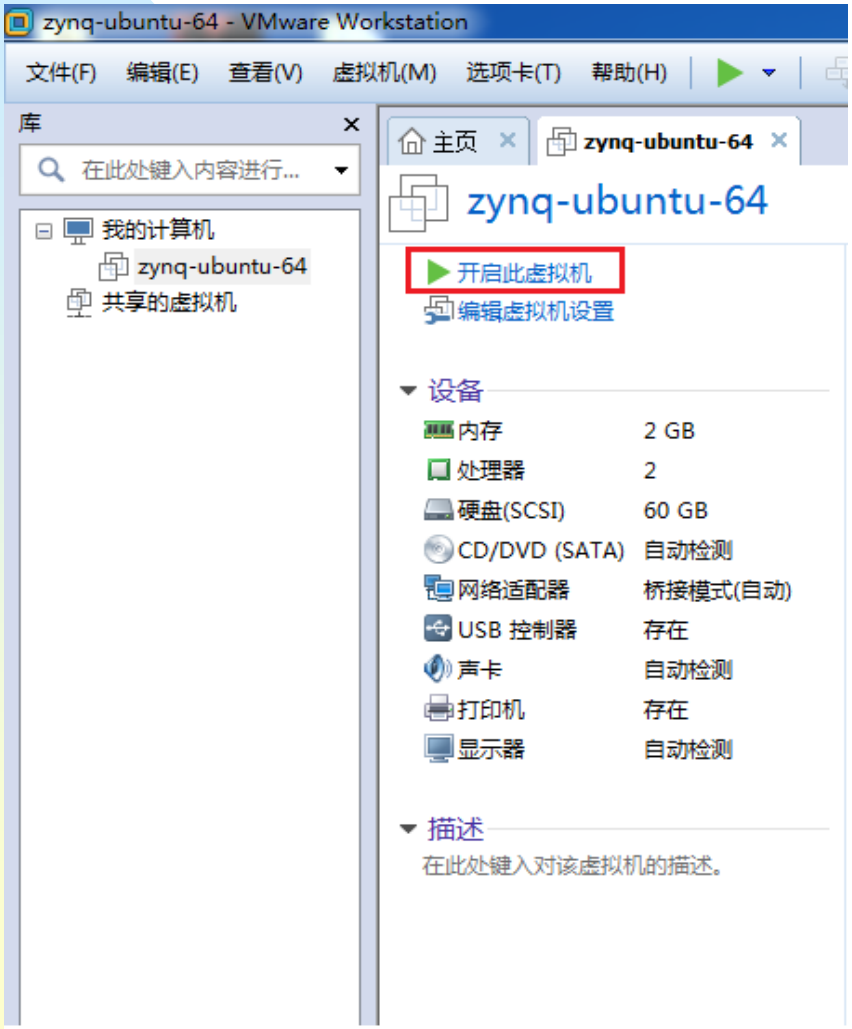
2.生成一级启动文件fsbl.elf

已生成fsbl.elf，点击fsbl.elf，Ctrl+C可直接复制，将fsbl.elf存于boot_file文件夹中

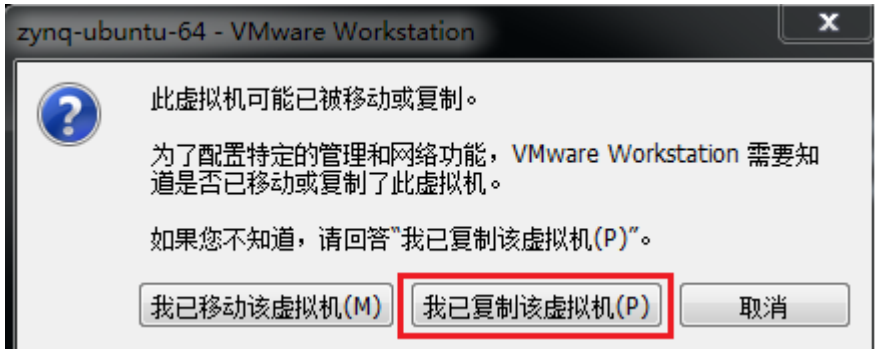


3.生成二级启动文件u-boot.elf

打开虚拟机

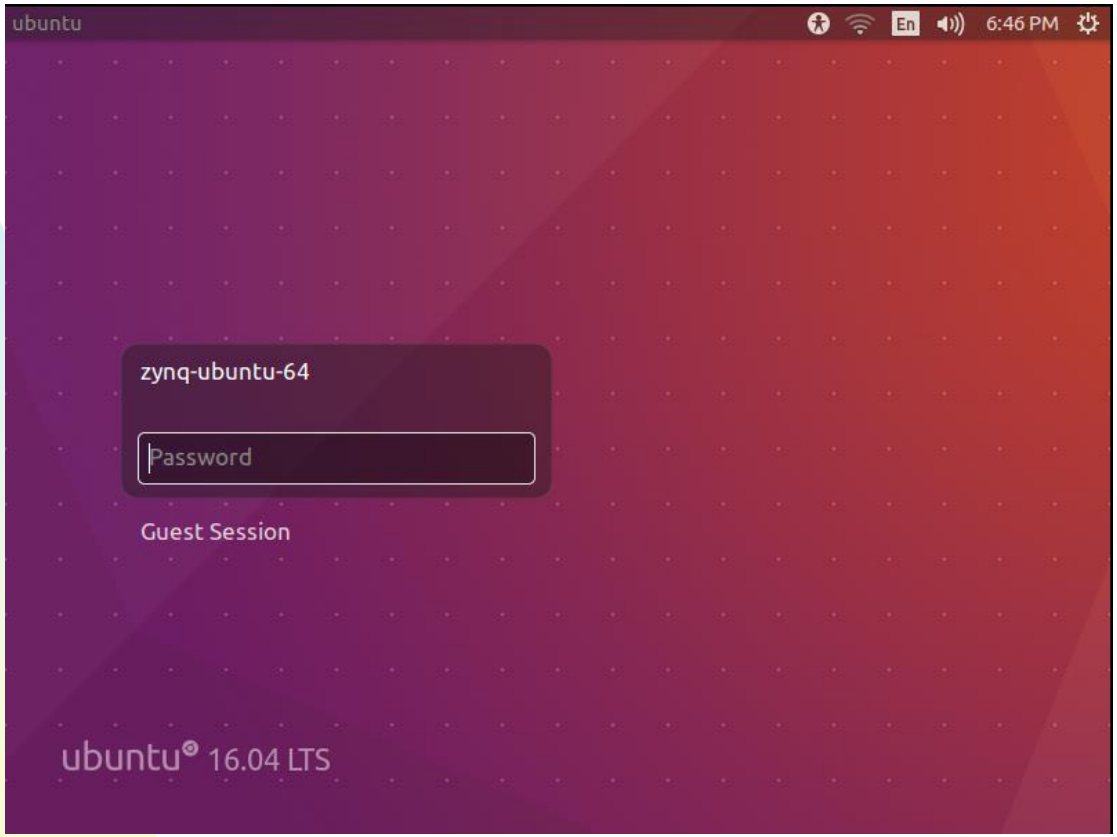


选择我已复制该虚拟机



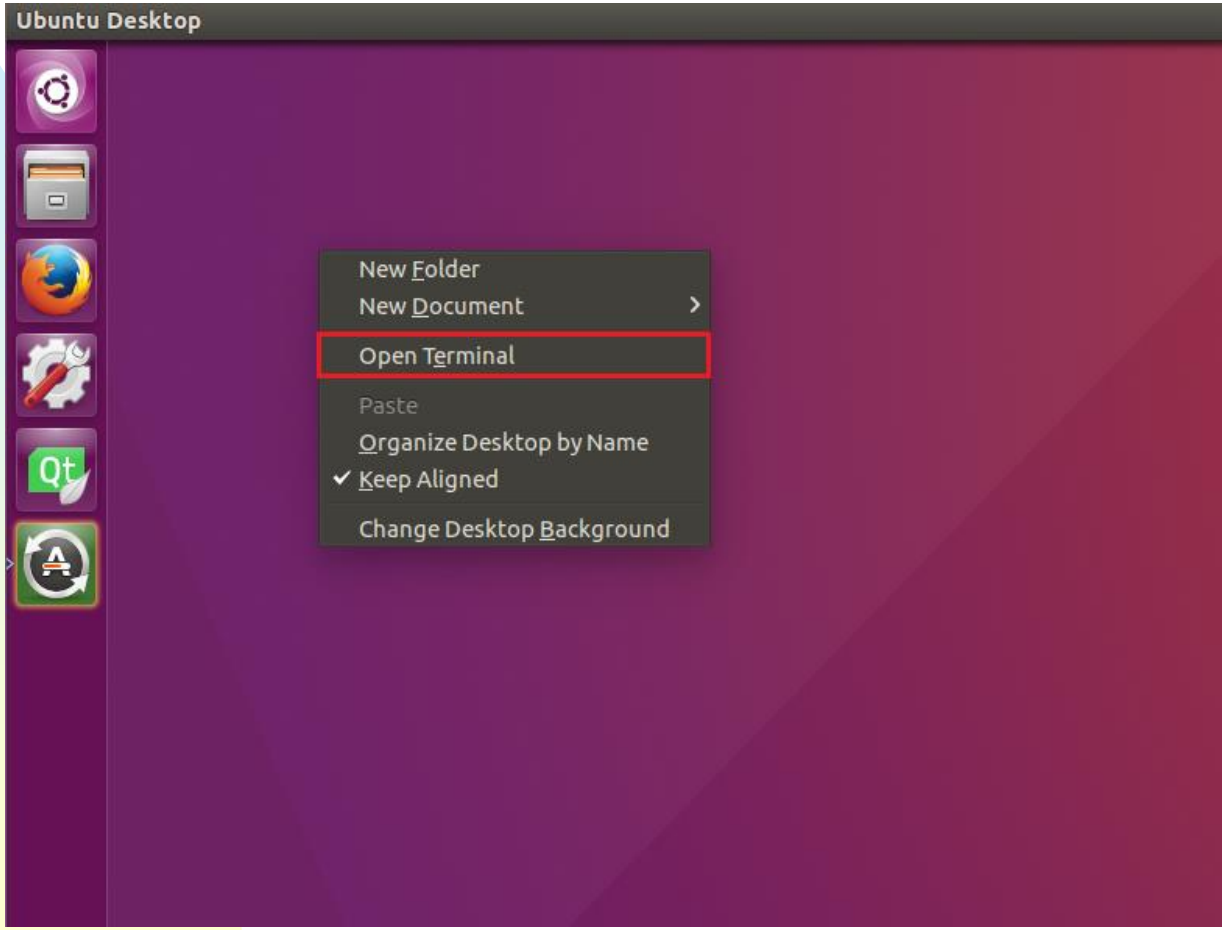
3.生成二级启动文件u-boot.elf

密码为1



3.生成二级启动文件u-boot.elf

在空白处右击，选择Open Terminal，打开终端



3.生成二级启动文件u-boot.elf

首先获取root权限，密码为1

```
zynq@ubuntu:~$ sudo su
[sudo] password for zynq:
```

获取root权限，密码为1

进入xilinx-zynq-linux配置环境变量,进入uboot清除之前的配置

```
root@ubuntu:/home/zynq# cd xilinx-zynq-linux/
root@ubuntu:/home/zynq/xilinx-zynq-linux# source zynq-env.sh
root@ubuntu:/home/zynq/xilinx-zynq-linux# cd u-boot-xlnx-xilinx-v2016.3
root@ubuntu:/home/zynq/xilinx-zynq-linux/u-boot-xlnx-xilinx-v2016.3# make distclean
  CLEAN   dts/../arch/arm/dts
  CLEAN   dts
  CLEAN   examples/standalone
  CLEAN   tools
  CLEAN   tools/lib tools/common
  CLEAN   spl/arch spl/board spl/boot.bin spl/cmd spl/common spl/disk spl/drivers spl/dts spl/fs spl
/lib spl/u-boot-spl spl/u-boot-spl.bin spl/u-boot-spl.cfg spl/u-boot-spl.lds spl/u-boot-spl.map spl/
u-boot-spl-nodtb.bin
  CLEAN   u-boot.lds u-boot.srec u-boot.map u-boot-nodtb.bin u-boot.cfg u-boot.bin u-boot u-boot.img
 u-boot.sym System.map
  CLEAN   scripts/basic
  CLEAN   scripts/kconfig
  CLEAN   include/config include/generated spl
  CLEAN   .config include/autoconf.mk include/autoconf.mk.dep include/config.h
```

配置编译环境

进入uboot目录

清除之前的配置

zynq-env.sh脚本中配置ARM架构、交叉编译器、linux内核和uboot的路径

zynq-env.sh (~/xilinx-zynq-linux) - gedit

Open

```
export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabihf-
export PATH=/opt/petalinux/tools/linux-i386/gcc-arm-linux-gnueabi/bin:$PATH
export PATH=~/xilinx-zynq-linux/u-boot-xlnx-xilinx-v2016.3/tools:$PATH
export KBUILD_DIR=~/xilinx-zynq-linux/linux-xlnx-xilinx-v2016.3
```

3.生成二级启动文件u-boot.elf

配 置 xilinx-zynq
的uboot环境

```
root@ubuntu:/home/zynq/xilinx-zynq-linux/u-boot-xlnx-xilinx-v2016.3# make zynq_zed_config
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  SHIPPED scripts/kconfig/zconf.tab.c
  SHIPPED scripts/kconfig/zconf.lex.c
  SHIPPED scripts/kconfig/zconf.hash.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#
```

配置xilinx_zynq的uboot环境

输入make menuconfig
可配置uboot参数

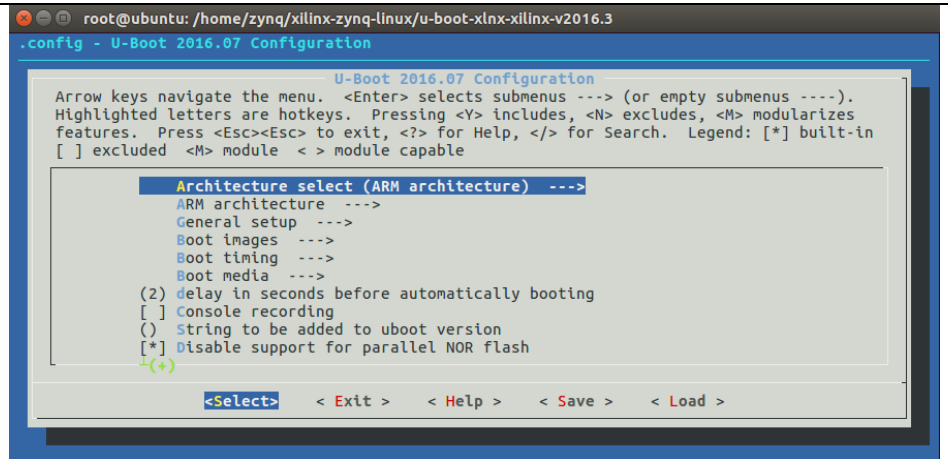
```
root@ubuntu:/home/zynq/xilinx-zynq-linux/u-boot-xlnx-xilinx-v2016.3# make menuconfig
  HOSTCC  scripts/kconfig/mconf.o
  HOSTCC  scripts/kconfig/lxdialog/checklist.o
  HOSTCC  scripts/kconfig/lxdialog/util.o
  HOSTCC  scripts/kconfig/lxdialog/inputbox.o
  HOSTCC  scripts/kconfig/lxdialog/textbox.o
  HOSTCC  scripts/kconfig/lxdialog/yesno.o
  HOSTCC  scripts/kconfig/lxdialog/menubox.o
  HOSTLD  scripts/kconfig/mconf
scripts/kconfig/mconf  Kconfig


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

可设置uboot参数
（这里不用设置）

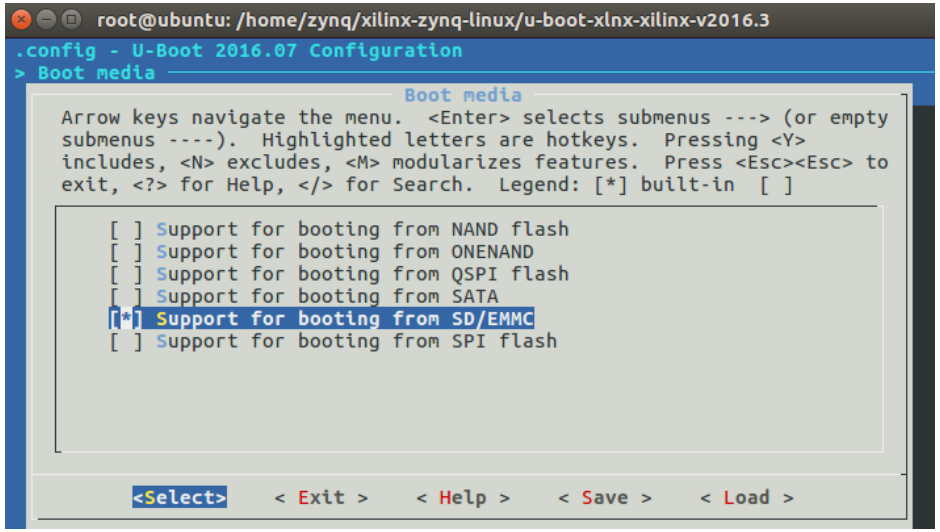
3.生成二级启动文件u-boot.elf

配置界面：

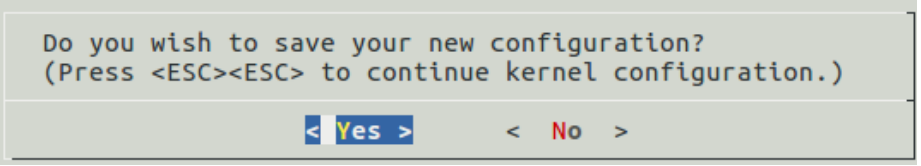


进入Boot media，选择SD卡内启动uboot

（按空格键选择）



选择Exit退出，选择Yes保存



3.生成二级启动文件u-boot.elf

进入xilinx-zynq-linux/u-boot-xlnx-xilinx-v2016.3/include/configs目录，找到zynq-common.h文件。此文件是uboot的配置文件，echo所打印的信息可在linux系统启动时查看。请在zynq-common.h中以下位置添加自己的姓名（拼音）和学号并保存文件

```
"ramdisk_image=uramdisk.image.gz\0"     \
"ramdisk_load_address=0x4000000\0"      \
"devicetree_image=devicetree.dtb\0"     \
"devicetree_load_address=0x2000000\0"   \
"bitstream_image=system.bit.bin\0"      \
"boot_image=BOOT.bin\0" \
"loadbit_addr=0x100000\0"       \
"loadbootenv_addr=0x2000000\0" \
"kernel_size=0x500000\0"        \
"devicetree_size=0x20000\0"     \
"ramdisk_size=0x5E0000\0"       \
"boot_size=0xF00000\0"  \
"fdt_high=0x20000000\0" \
"initrd_high=0x20000000\0"      \
"bootenv=uEnv.txt\0" \
"loadbootenv=load mmc 0 ${loadbootenv_addr} ${bootenv}\0" \
"importbootenv=echo Importing environment from SD your name and student number ...; " \
        "env import -t ${loadbootenv_addr} $filesize\0" \
"sd_uEnvtxt_existence_test=test -e mmc 0 /uEnv.txt\0" \
"preboot=if test $modeboot = sdboot && env run sd_uEnvtxt_existence_test; " \
                "then if env run loadbootenv; " \
                        "then env run importbootenv; " \
                "fi; " \
        "fi; \0" \
"mmc_loadbit=echo Loading bitstream from SD/MMC/eMMC to RAM.. && " \
        "mmcinfo && " \
        "load mmc 0 ${loadbit_addr} ${bitstream_image} && " \
        "fpga load 0 ${loadbit_addr} ${filesize}\0" \
"norboot=echo Copying Linux from NOR flash to RAM... && " \
        "cp.b 0xE2100000 ${kernel_load_address} ${kernel_size} && " \
        "cp.b 0xE2600000 ${devicetree_load_address} ${devicetree_size} && " \
        "echo Copying ramdisk... && " \
```

3.生成二级启动文件u-boot.elf

回到终端，输入make命令编译uboot

```
root@ubuntu:/home/zynq/xilinx-zynq-linux/u-boot-xlnx-xilinx-v2016.3# make 编译uboot
scripts/kconfig/conf  --silentoldconfig Kconfig
  CHK     include/config.h
  UPD     include/config.h
  GEN     include/autoconf.mk
  GEN     include/autoconf.mk.dep
  GEN     spl/include/autoconf.mk
  CHK     include/config/uboot.release
  UPD     include/config/uboot.release
  CHK     include/generated/version_autogenerated.h
  UPD     include/generated/version_autogenerated.h
  CHK     include/generated/timestamp_autogenerated.h
  UPD     include/generated/timestamp_autogenerated.h
```

编译完成

```
  CC      spl/lib/linux_string.o
  CC      spl/lib/membuff.o
  CC      spl/lib/slre.o
  CC      spl/lib/string.o
  CC      spl/lib/time.o
  CC      spl/lib/rand.o
  CC      spl/lib/vsprintf.o
  CC      spl/lib/panic.o
  CC      spl/lib/strto.o
  CC      spl/lib/strmhz.o
  LD      spl/lib/built-in.o
  LDS     spl/u-boot-spl.lds
  LD      spl/u-boot-spl
  OBJCOPY spl/u-boot-spl-nodtb.bin
  COPY    spl/u-boot-spl.bin
  CFG     spl/u-boot-spl.cfg
  MKIMAGE spl/boot.bin
  MKIMAGE u-boot.img
```

3.生成二级启动文件u-boot.elf

将生成的u-boot修改为u-boot.elf，并修改其权限

```
root@ubuntu:/home/zynq/xilinx-zynq-linux/u-boot-xlnx-xilinx-v2016.3# ls
api      config.mk  dts       Kconfig      Makefile  snapshot.commit  u-boot      u-boot.map
arch     configs    examples  lib          net       spl              u-boot.bin  u-boot-nodtb.bin
board    disk       fs        Licenses     post      System.map       u-boot.cfg  u-boot.srec
cmd      doc        include   MAINTAINERS  README    test             u-boot.img  u-boot.sym
common   drivers    Kbuild    MAKEALL      scripts   tools            u-boot.lds
root@ubuntu:/home/zynq/xilinx-zynq-linux/u-boot-xlnx-xilinx-v2016.3# cp u-boot u-boot.elf
root@ubuntu:/home/zynq/xilinx-zynq-linux/u-boot-xlnx-xilinx-v2016.3# chmod 777 u-boot.elf
```

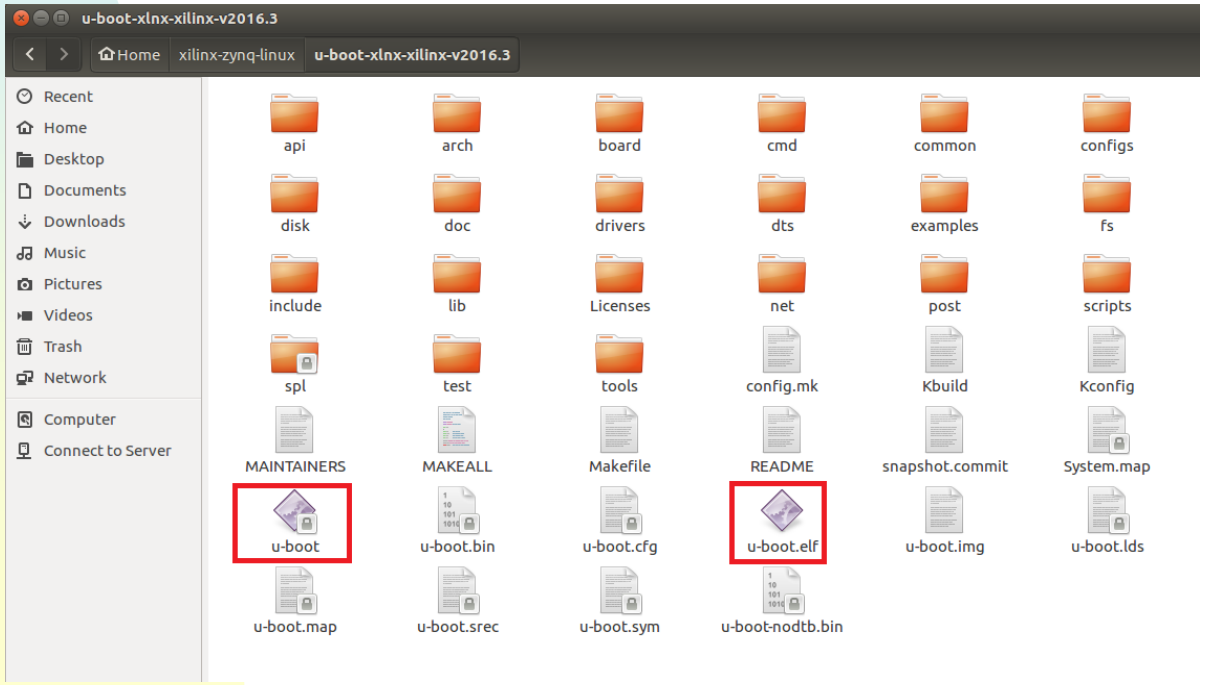
ls命令查看当前目录下的文件

编译生成的uboot

改为u-boot.elf

修改uboot.elf权限

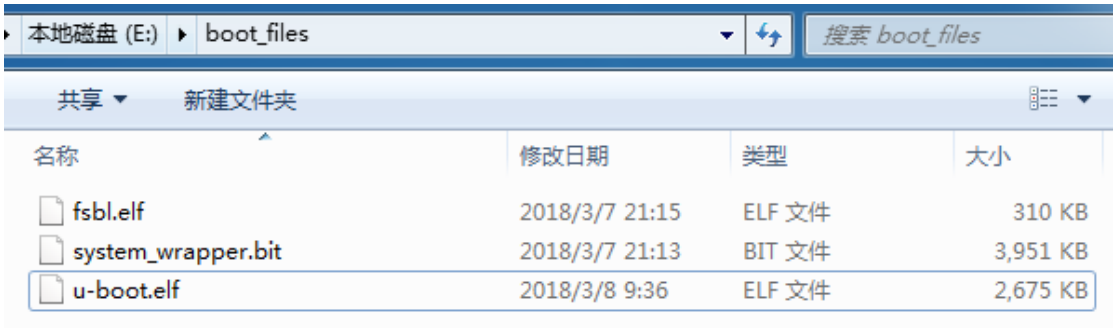
uboot目录下



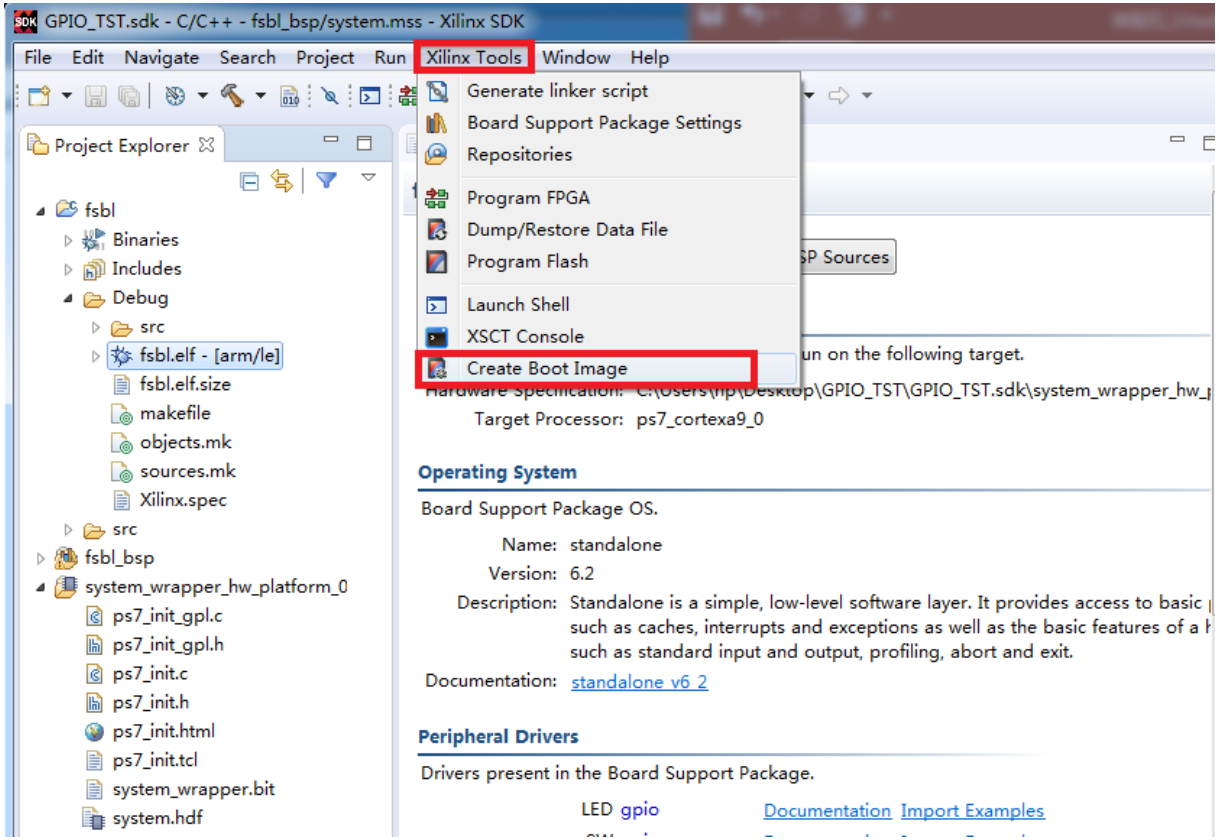
将u-boot.elf放到boot_files文件夹中

4.生成BOOT.BIN

boot_files中包括三个文件，在SDK中生成BOOT.BIN



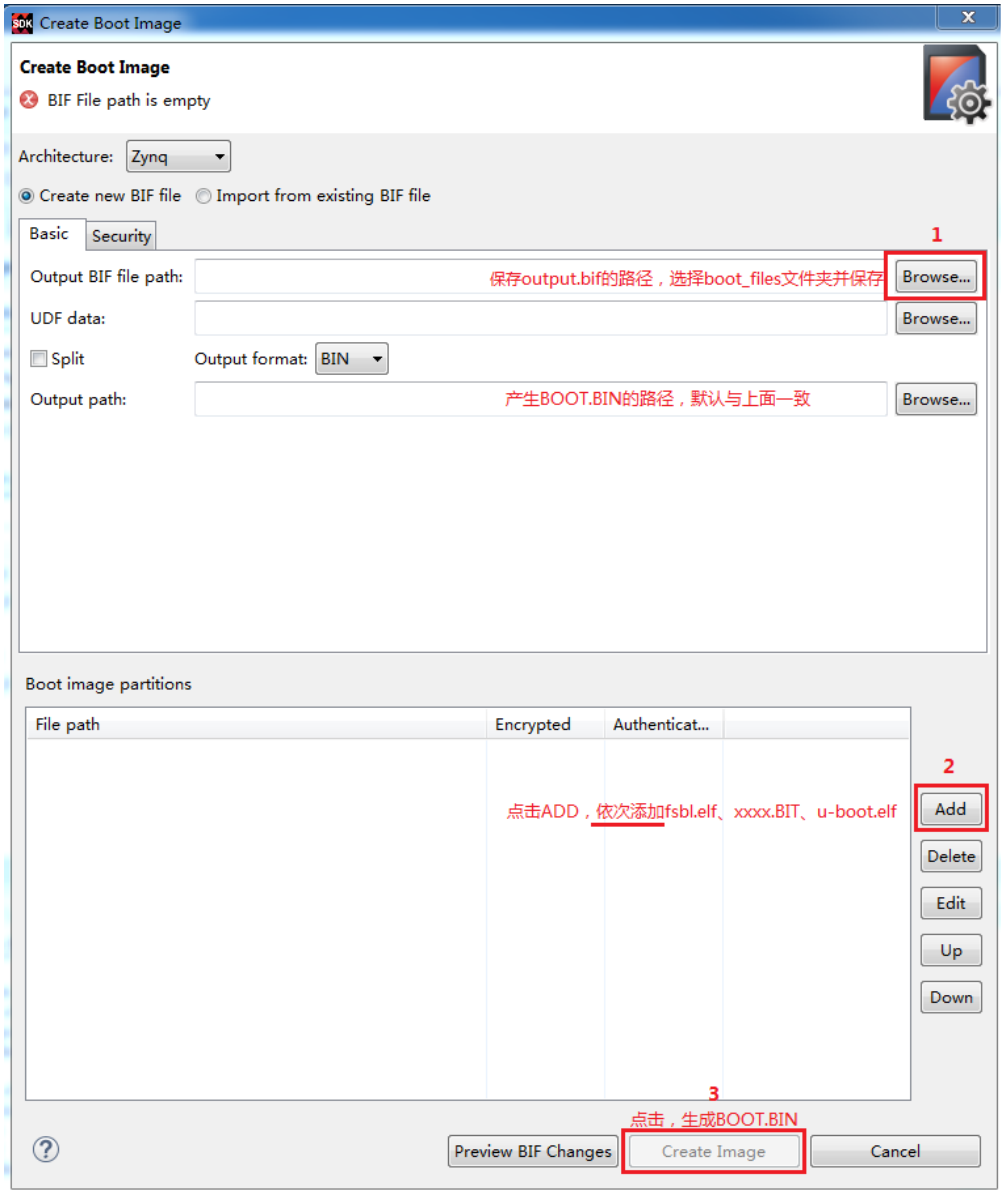
在 SDK 中打开 Xilinx Tools——Create Boot Image



4.生成BOOT.BIN

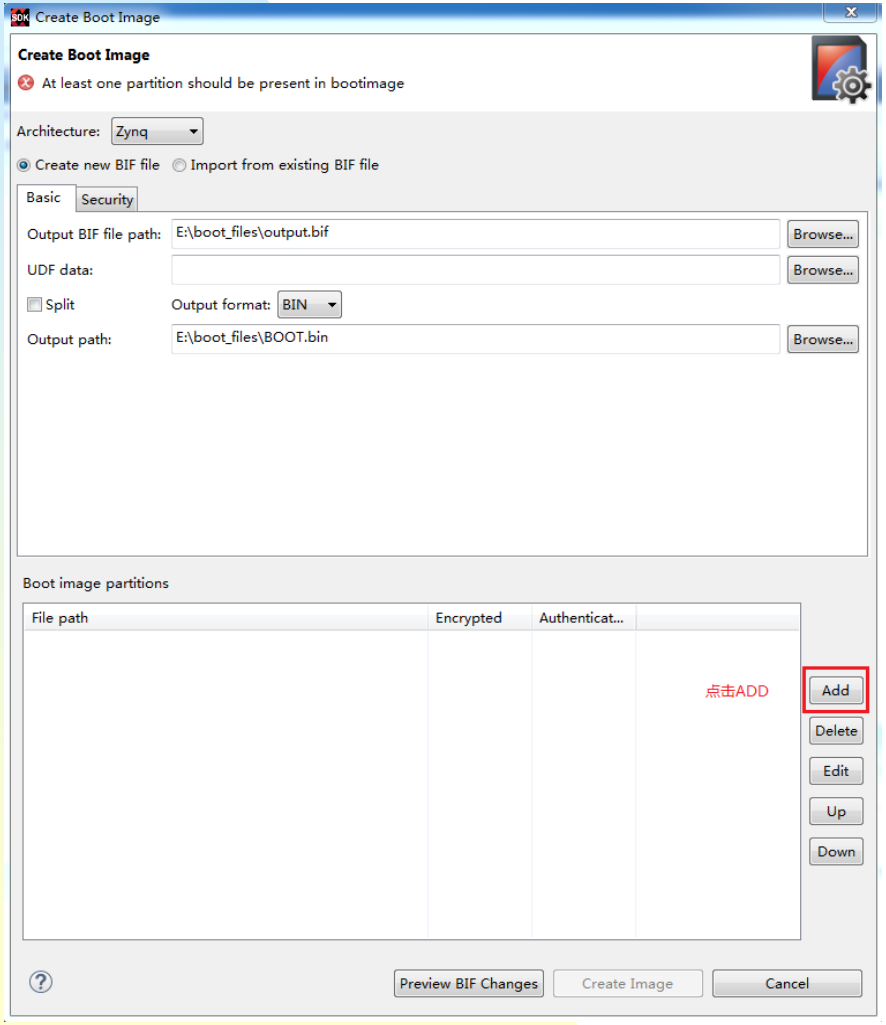
按照图中顺序
依次添加文件

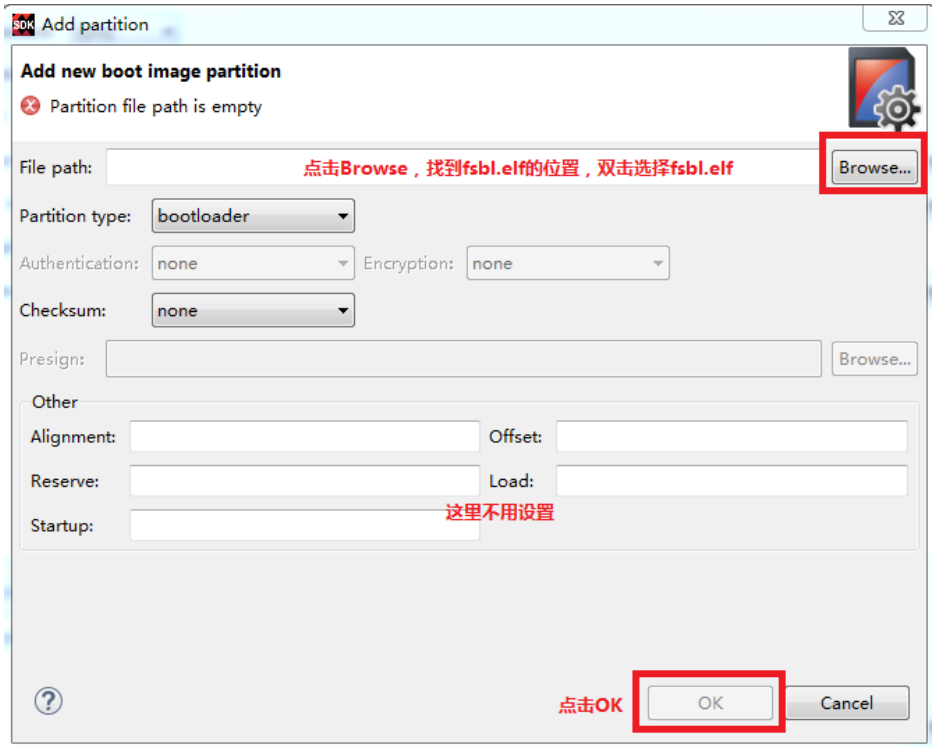
生成BOOT.BIN



4.生成BOOT.BIN

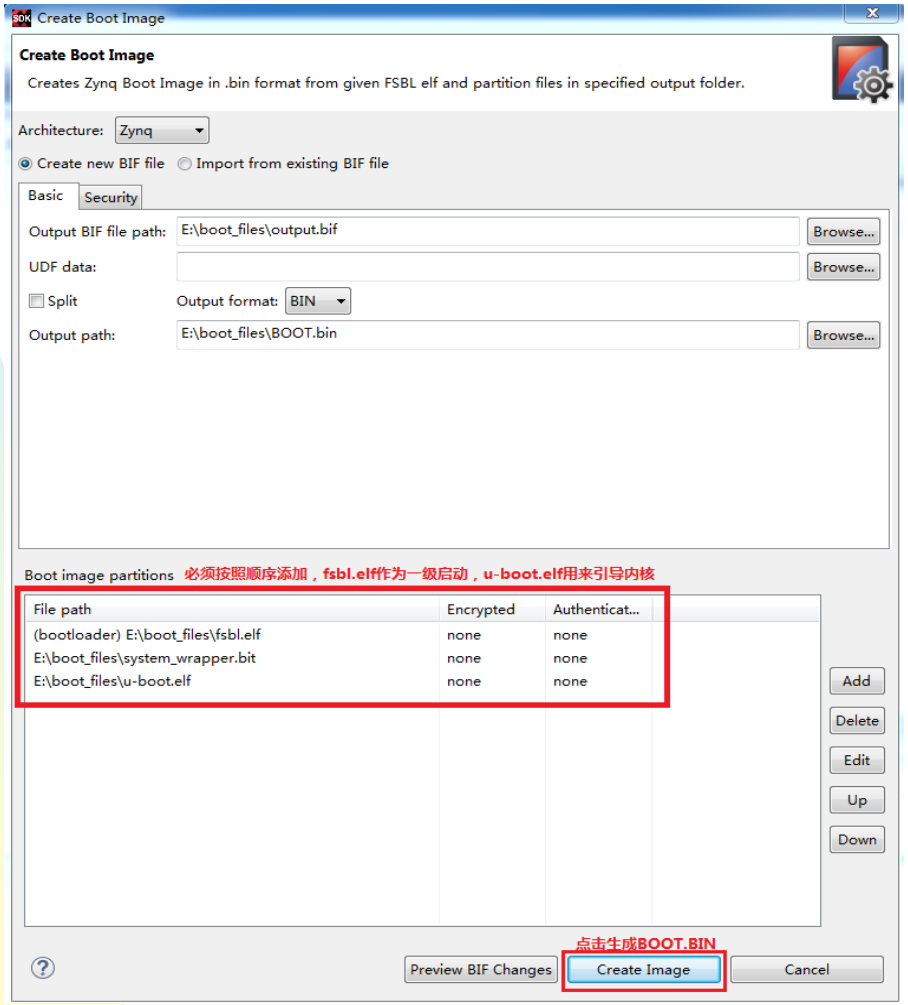
以添加fsbl.elf为例





4.生成BOOT.BIN

同样的方式添加BIT文件和u-boot.elf，添加完成：



生成的BOOT.BIN在boot_files中

5.生成内核uImage

再打开一个终端，获取root权限，配置编译环境

```
root@ubuntu: /home/zynq/xilinx-zynq-linux/linux-xlnx-xilinx-v2016.3
zynq@ubuntu:~$ sudo su          获取root权限
[sudo] password for zynq:       密码为1
root@ubuntu:/home/zynq# cd xilinx-zynq-linux/          进入xilinx-zynq-linux
root@ubuntu:/home/zynq/xilinx-zynq-linux# source zynq-env.sh          配置编译环境
```

进入Linux内核目录，清除之前的配置

```
root@ubuntu:/home/zynq/xilinx-zynq-linux# cd linux-xlnx-xilinx-v2016.3          进入内核
root@ubuntu:/home/zynq/xilinx-zynq-linux/linux-xlnx-xilinx-v2016.3# make distclean          清除之前的设置
  CLEAN   .
  CLEAN   arch/arm/kernel
  CLEAN   drivers/tty/vt
  CLEAN   kernel
  CLEAN   lib
  CLEAN   usr
  CLEAN   arch/arm/boot/compressed
  CLEAN   arch/arm/boot
  CLEAN   .tmp_versions
  CLEAN   scripts/basic
  CLEAN   scripts/dtc
  CLEAN   scripts/genksyms
  CLEAN   scripts/kconfig
  CLEAN   scripts/mod
  CLEAN   scripts
  CLEAN   include/config include/generated arch/arm/include/generated
  CLEAN   .config .version Module.symvers
```

5.生成内核uImage

进行xilinx-zynq默认配置

```
root@ubuntu:/home/zynq/xilinx-zynq-linux/linux-xlnx-xilinx-v2016.3# make xilinx_zynq_defconfig
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  SHIPPED scripts/kconfig/zconf.tab.c
  SHIPPED scripts/kconfig/zconf.lex.c
  SHIPPED scripts/kconfig/zconf.hash.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#
```

配置xilinx-zynq的内核环境

将uboot目录下tools中的mkimage拷贝至/usr/bin/下面并修改权限，用于编译内核

```
root@ubuntu:/home/zynq/xilinx-zynq-linux/linux-xlnx-xilinx-v2016.3# cp ../u-boot-xlnx-xilinx-v2016.3/tools/mkimage /usr/bin/
root@ubuntu:/home/zynq/xilinx-zynq-linux/linux-xlnx-xilinx-v2016.3# chmod 777 /usr/bin/mkimage
```

将uboot下tools中的mkimage拷贝至/usr/bin/

修改mkimage权限

5.生成内核uImage

若有需要可输入make menuconfig进入内核参数配置界面，这里不需要配置

```
root@ubuntu:/home/zynq/xilinx-zynq-linux/linux-xlnx-xilinx-v2016.3# make menuconfig
  HOSTCC  scripts/kconfig/mconf.o
  HOSTCC  scripts/kconfig/lxdialog/checklist.o
  HOSTCC  scripts/kconfig/lxdialog/util.o
  HOSTCC  scripts/kconfig/lxdialog/inputbox.o
  HOSTCC  scripts/kconfig/lxdialog/textbox.o
  HOSTCC  scripts/kconfig/lxdialog/yesno.o
  HOSTCC  scripts/kconfig/lxdialog/menubox.o
  HOSTLD  scripts/kconfig/mconf
scripts/kconfig/mconf  Kconfig


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

内核配置界面：

不需要配置，选择Exit退出配置界面

```
root@ubuntu: /home/zynq/xilinx-zynq-linux/linux-xlnx-xilinx-v2016.3
.config - Linux/arm 4.6.0 Kernel Configuration

                    Linux/arm 4.6.0 Kernel 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

        -*- Patch physical to virtual translations at runtime
            General setup  --->
        [*] Enable loadable module support  --->
        [*] Enable the block layer  --->
            System Type  --->
            Bus support  --->
            Kernel Features  --->
            Boot options  --->
            CPU Power Management  --->
            Floating point emulation  --->
        └(+)

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

5.生成内核uImage

输入make UIMAGE_LOADADDR=0X8000 uImage -j2编译内核：

注意UIMAGE和uImage中的‘I’都是大写的i

```
root@ubuntu:/home/zynq/xilinx-zynq-linux/linux-xlnx-xilinx-v2016.3# make UIMAGE_LOADADDR=0x8000 uImage -j2
scripts/kconfig/conf  --silentoldconfig Kconfig
  CHK     include/config/kernel.release
  UPD     include/config/kernel.release
  WRAP    arch/arm/include/generated/asm/bitsperlong.h
  WRAP    arch/arm/include/generated/asm/clkdev.h
  WRAP    arch/arm/include/generated/asm/cputime.h
  CHK     include/generated/uapi/linux/version.h
  WRAP    arch/arm/include/generated/asm/current.h
  WRAP    arch/arm/include/generated/asm/early_ioremap.h
  UPD     include/generated/uapi/linux/version.h
  WRAP    arch/arm/include/generated/asm/emergency-restart.h
  WRAP    arch/arm/include/generated/asm/errno.h
  WRAP    arch/arm/include/generated/asm/exec.h
  WRAP    arch/arm/include/generated/asm/ioctl.h
  WRAP    arch/arm/include/generated/asm/irq_regs.h
  WRAP    arch/arm/include/generated/asm/kdebug.h
  WRAP    arch/arm/include/generated/asm/ipcbuf.h
```

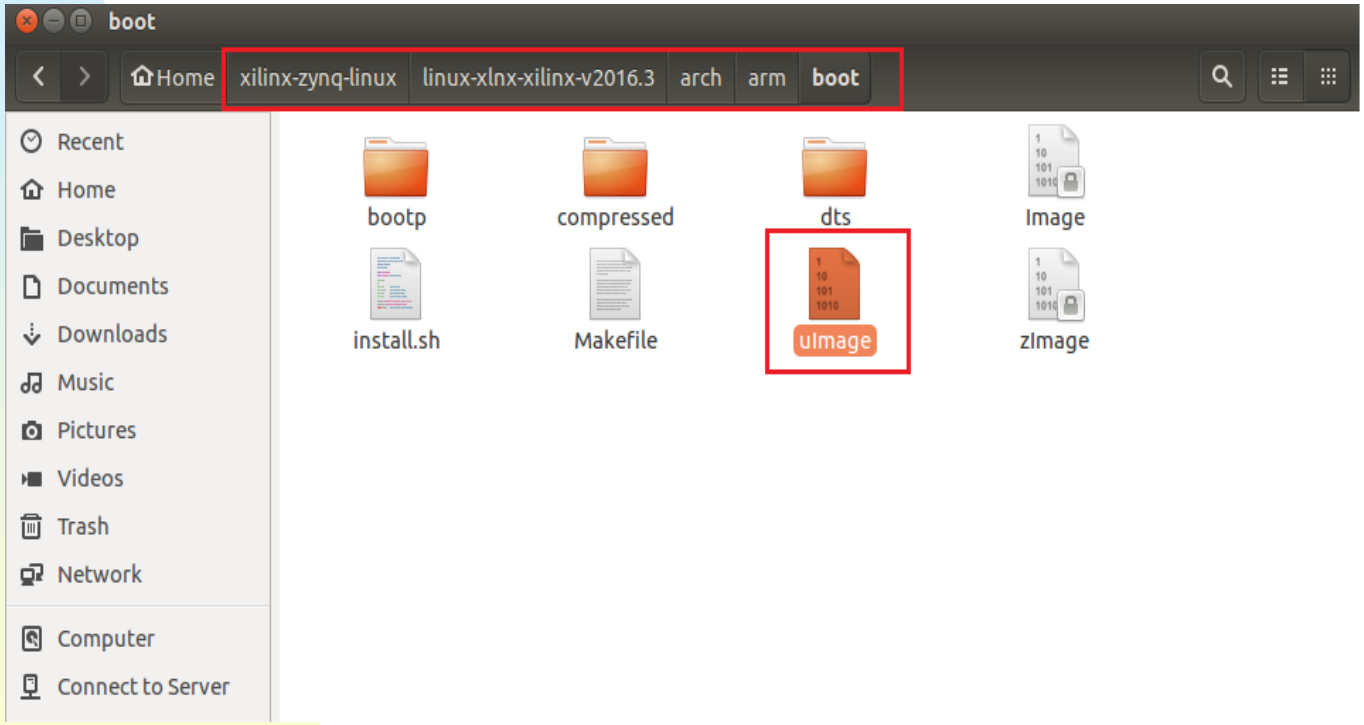
等几分钟，内核编译完成：

```
root@ubuntu: /home/zynq/xilinx-zynq-linux/linux-xlnx-xilinx-v2016.3
  CC      arch/arm/boot/compressed/string.o
  SHIPPED arch/arm/boot/compressed/hyp-stub.S
  SHIPPED arch/arm/boot/compressed/lib1funcs.S
  SHIPPED arch/arm/boot/compressed/ashldi3.S
  SHIPPED arch/arm/boot/compressed/bswapsdi2.S
  AS      arch/arm/boot/compressed/hyp-stub.o
  AS      arch/arm/boot/compressed/lib1funcs.o
  AS      arch/arm/boot/compressed/ashldi3.o
  AS      arch/arm/boot/compressed/bswapsdi2.o
  AS      arch/arm/boot/compressed/piggy.o
  LD      arch/arm/boot/compressed/vmlinux
  OBJCOPY arch/arm/boot/zImage
  Kernel: arch/arm/boot/zImage is ready
  Kernel: arch/arm/boot/Image is ready
  Kernel: arch/arm/boot/zImage is ready
  UIMAGE  arch/arm/boot/uImage
Image Name:   Linux-4.6.0-xilinx
Created:      Wed Mar  7 19:01:19 2018
Image Type:   ARM Linux Kernel Image (uncompressed)
Data Size:    3827080 Bytes = 3737.38 kB = 3.65 MB
Load Address: 00008000
Entry Point:  00008000
  Image arch/arm/boot/uImage is ready
root@ubuntu:/home/zynq/xilinx-zynq-linux/linux-xlnx-xilinx-v2016.3#
```

5.生成内核uImage

生成的内核为arch/arm/boot/下的uImage，需修改其权限：

```
root@ubuntu:/home/zynq/xilinx-zynq-linux/linux-xlnx-xilinx-v2016.3# cd arch/arm/boot/ 进入arch/arm/boot/
root@ubuntu:/home/zynq/xilinx-zynq-linux/linux-xlnx-xilinx-v2016.3/arch/arm/boot# ls
bootp  compressed  dts  Image  install.sh  Makefile  uImage  zImage
root@ubuntu:/home/zynq/xilinx-zynq-linux/linux-xlnx-xilinx-v2016.3/arch/arm/boot# chmod 777 uImage  修改uImage权限
```

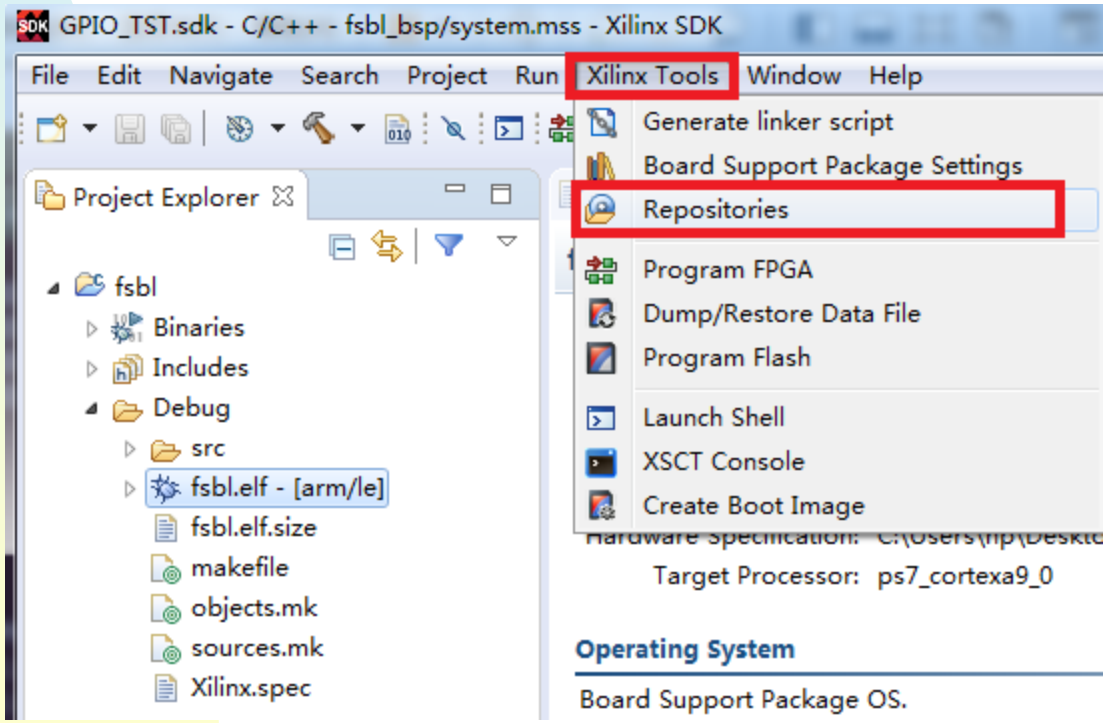


将uImage复制到boot_files中

6.生成设备树devicetree.dtb

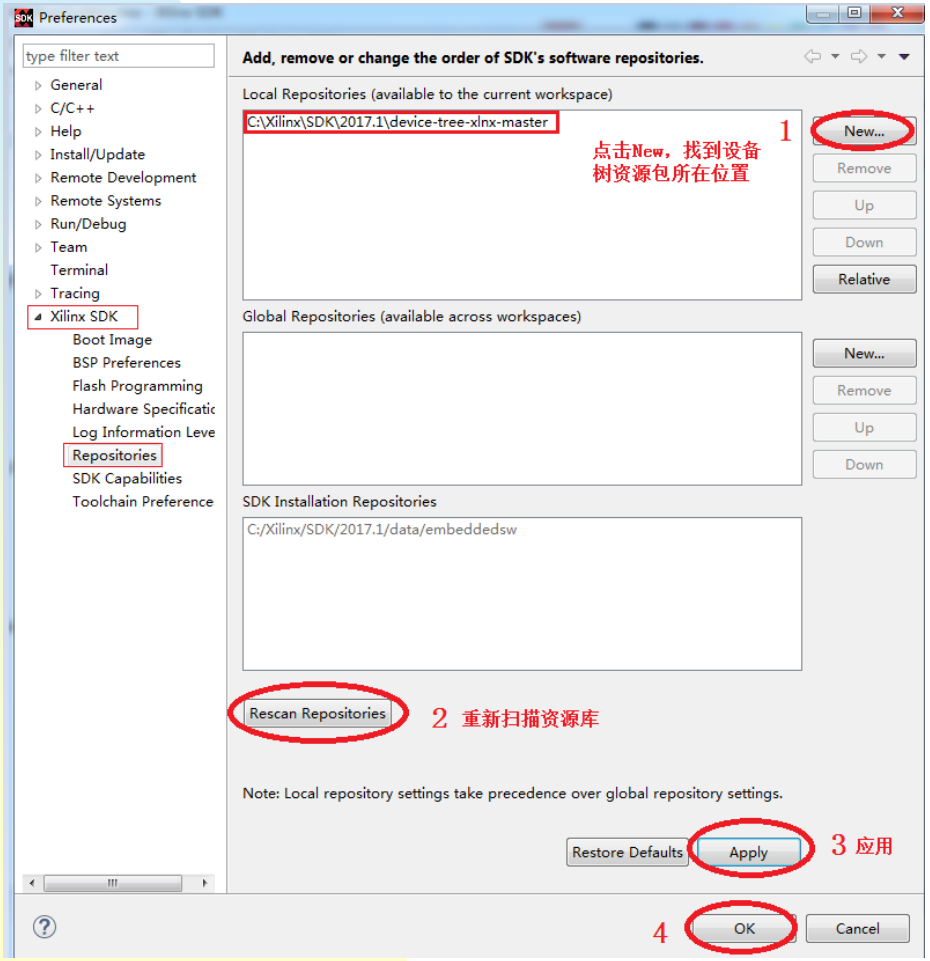
SDK中可依据硬件工程生成设备树源文件，为在SDK中生成与硬件工程适配的设备树，需要在SDK中先加载设备树资源包

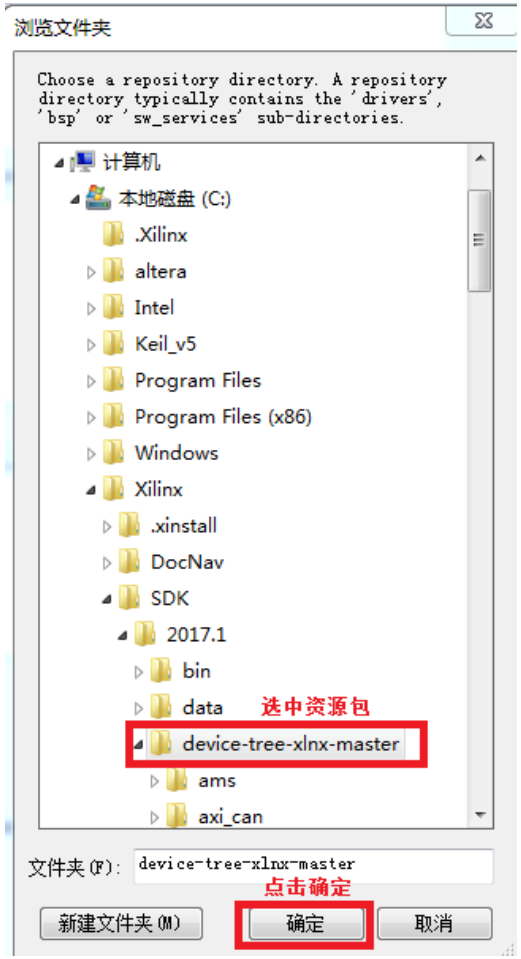
在SDK中打开Xilinx -> Respositories



6.生成设备树devicetree.dtb

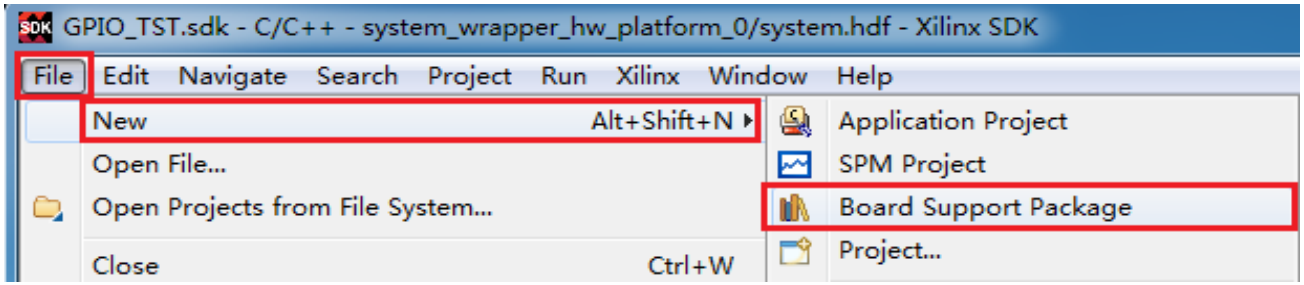
设备树资源包在Vivado安装目录下，如下图所示位置，在Xilinx SDK -> Repositories中点击New，找到设备树资源包，点击Rescan Repositories，点击OK完成加载



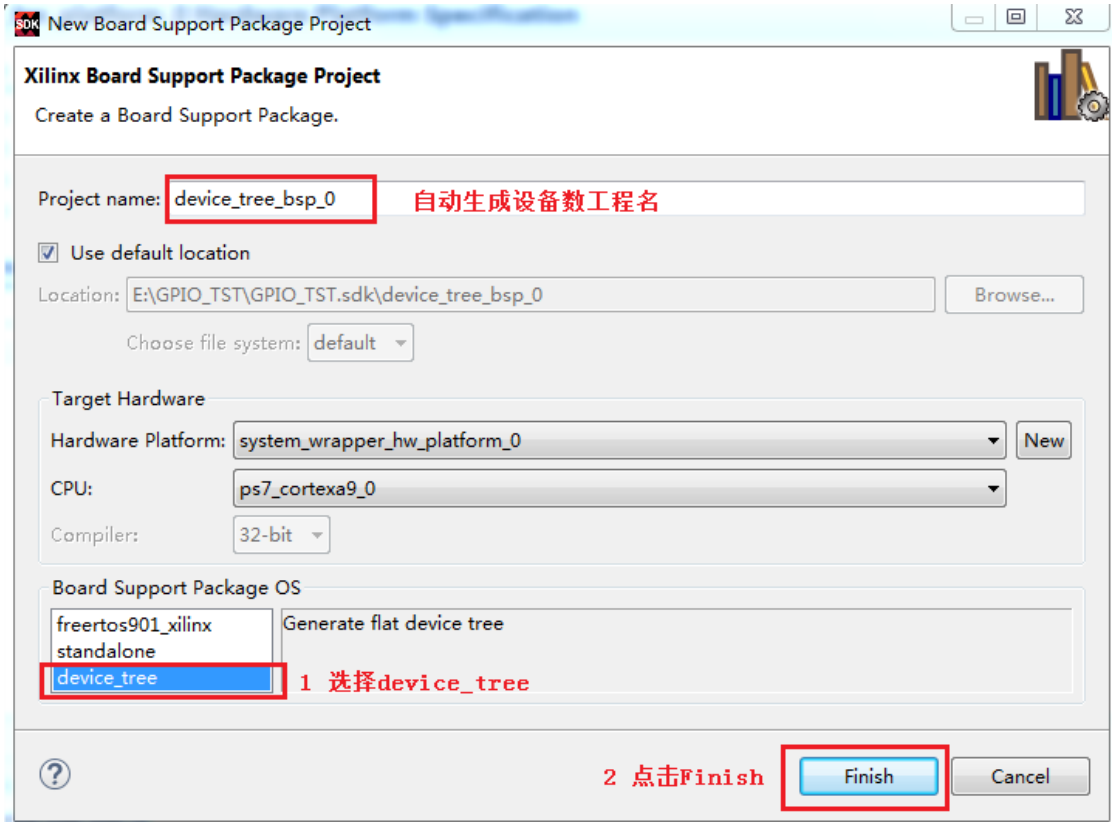


6.生成设备树devicetree.dtb

创建设备树BSP
点击File->New->Xilinx Board Support Package

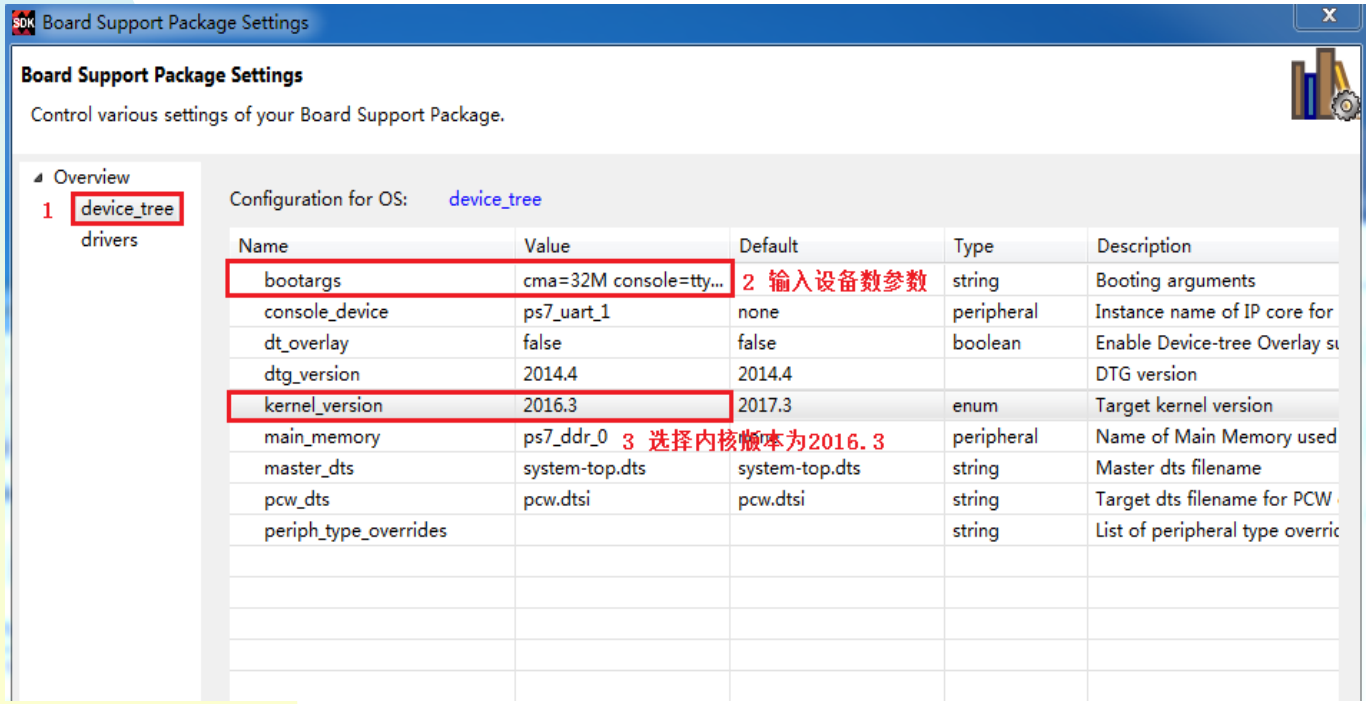


在Board Support Package OS选择device_tree。



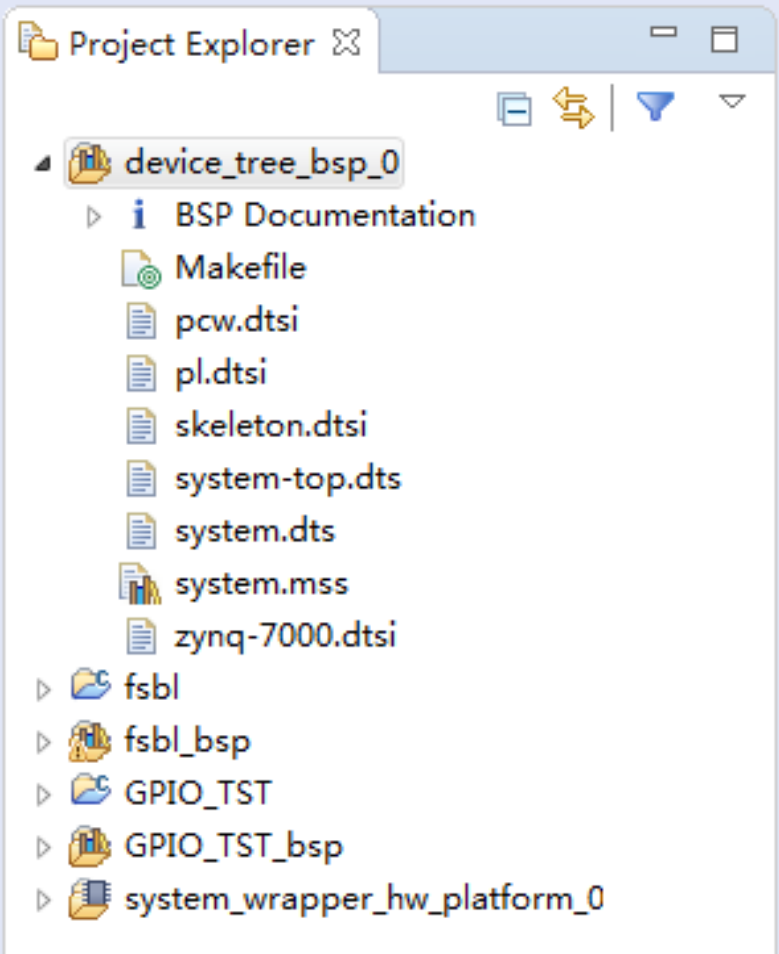
6.生成设备树devicetree.dtb

设置设备数参数，bootargs设为cma=32M console=ttyPS0,115200 root=/dev/mmcblk0p2 rw earlyprintk rootfstype=ext4 rootwait devtmpfs.mount=1，内核版本设为2016.3，设置完成后点击“OK”生成设备树文件。



6.生成设备树devicetree.dtb

生成的设备树工程:

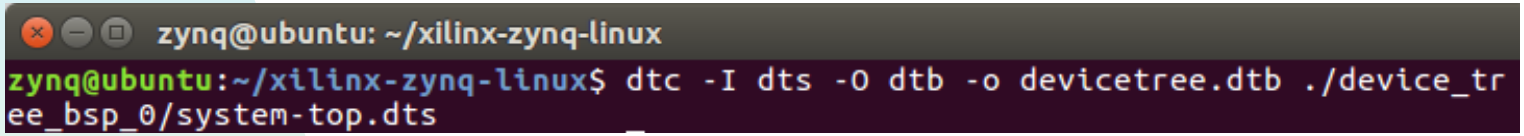


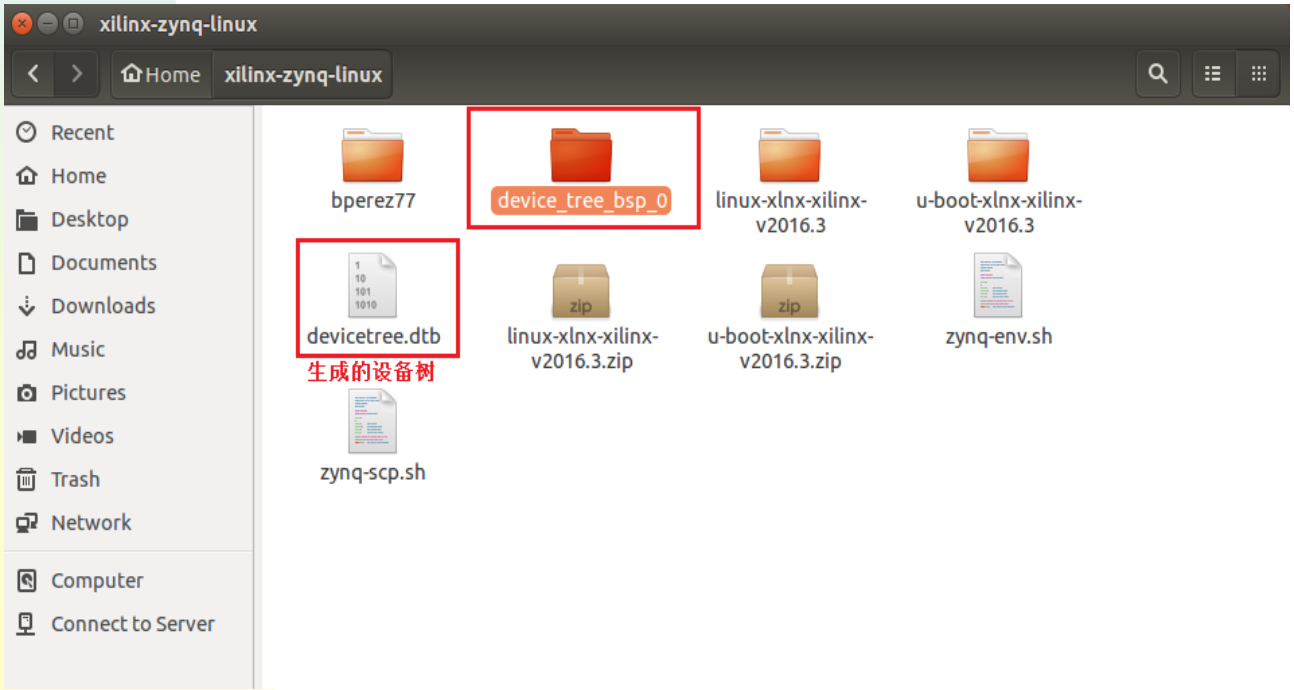
6.生成设备树devicetree.dtb

复制device_tree_bsp_0到虚拟机中，在device_tree_bsp_0所在目录下打开终端输入

dtc -I dts -O dtb -o devicetree.dtb ./device_tree_bsp_0/system-top.dts

会在目录下产生devicetree.dtb，将devicetree.dtb存于boot_files文件夹中





7. SD卡启动Linux操作系统

至此，boot_files中应有的文件如下，其中uEnv.txt是uboot配置过程中调用的文件，可以设置内核、设备树、文件系统的启动参数，此文件已提供，可直接拷贝使用。

BOOT.bin	2018/3/8 16:39	BIN 文件	4,491 KB
devicetree.dtb	2018/3/8 11:53	DTB 文件	10 KB
fsbl.elf	2018/3/8 16:07	ELF 文件	310 KB
output.bif	2018/3/8 16:39	BIF 文件	1 KB
system_wrapper.bit	2018/3/8 16:03	BIT 文件	3,951 KB
u-boot.elf	2018/3/8 16:38	ELF 文件	2,675 KB
uEnv	2018/1/19 17:30	文本文档	1 KB
uImage	2018/3/8 11:01	文件	3,738 KB

将SD卡插入读卡器，读卡器插入电脑，将BOOT.BIN、devicetree.dtb、uEnv.txt、uImage拷贝至SD卡BOOT分区，拔下SD卡，插到板子的SD卡插槽里，将模式拨码开关的3和4拨到下面，表示从SD卡启动

注意：

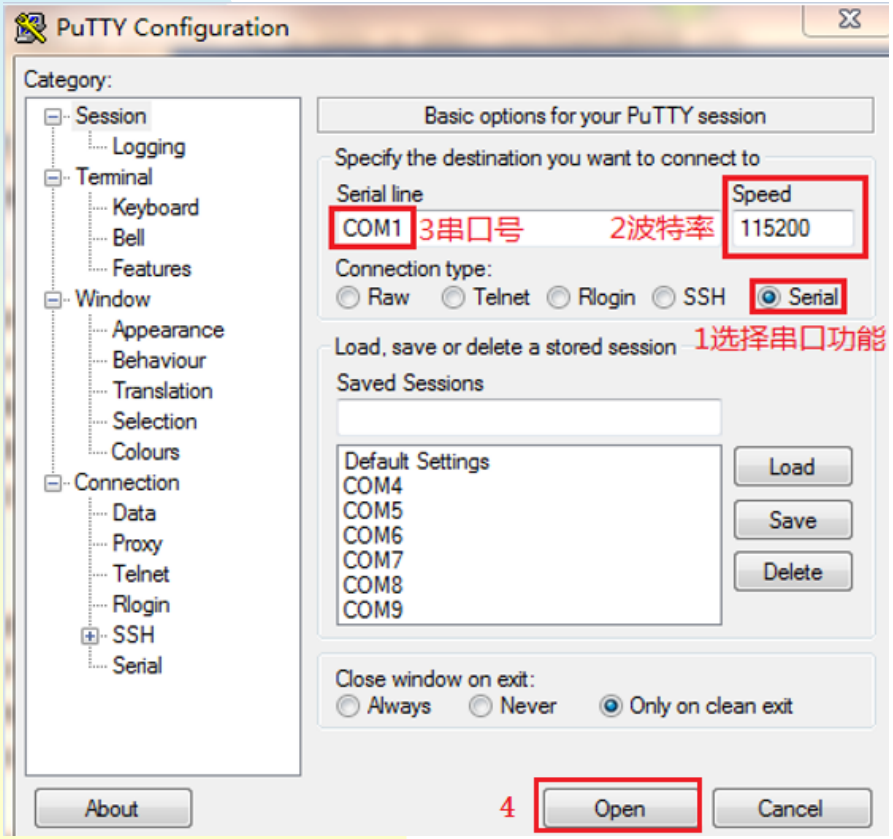
1. Windows下只能读到BOOT分区，虚拟机下可读到BOOT和rootfs两个分区
2. 从板子上插拔SD卡最好断电操作
3. 从板子上拔出SD卡时，需先将SD卡往里面按一下才会弹出，不要直接拔出

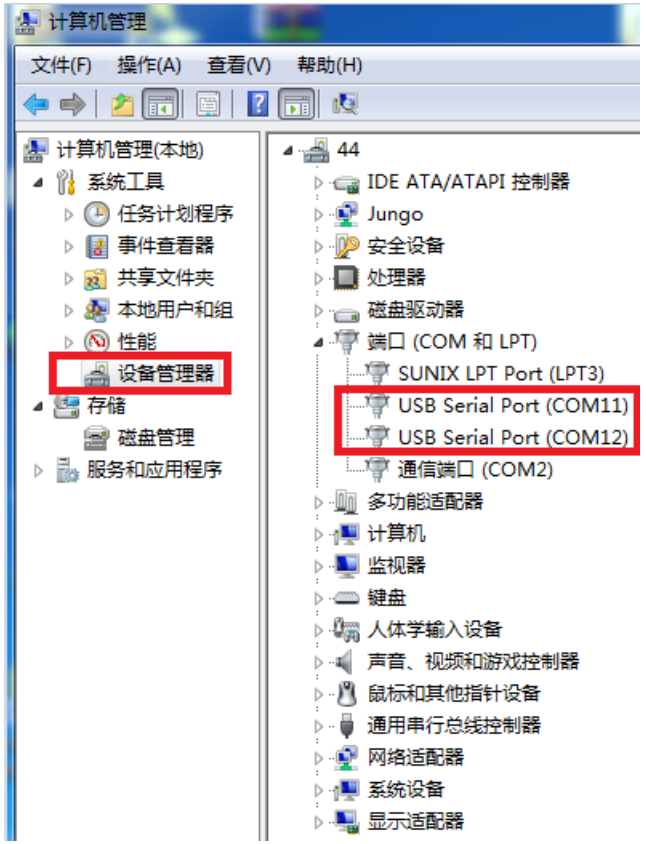
7. SD卡启动Linux操作系统

打开串口

UART线连接板子和PC机，查看端口号，打开桌面串口工具Putty，设置波特率和端口号。

查看端口号：右击桌面计算机→ 管理 → 设备管理器 → 端口





7. SD卡启动Linux操作系统

开发板上电，若串口打印如下，则linux系统启动成功：

```
COM12 - PuTTY
[  OK  ] Started System Logging Service.
[  OK  ] Started Provide limited super user privileges to specific users.
[  OK  ] Started D-Bus System Message Bus.
[  OK  ] Started WPA supplicant.
[  OK  ] Started LSB: Load kernel modules needed to enable cpufreq scaling.
[  OK  ] Started LSB: Xen daemons.
[  OK  ] Started Permit User Sessions.
[  OK  ] Started LSB: Set the CPU Frequency Scaling governor to "ondemand".
[  OK  ] Started Initialize hardware monitoring sensors.
         Starting LSB: Start/stop secondary xen domains...
         Starting LSB: set CPUFreq kernel parameters...
         Starting Network Service...
[  OK  ] Started Login Service.
         Starting Authenticate and Authorize Users to Run Privileged Tasks...
[  OK  ] Started LSB: set CPUFreq kernel parameters.
[  OK  ] Started Network Service.
         Starting Network Name Resolution...
[  OK  ] Started LSB: Start/stop secondary xen domains.
[  OK  ] Started Authenticate and Authorize Users to Run Privileged Tasks.
[  OK  ] Started Network Name Resolution.
[  OK  ] Started Network Manager.
[  OK  ] Reached target Network.
         Starting /etc/rc.local Compatibility...
[  OK  ] Started /etc/rc.local Compatibility.
[  OK  ] Started Getty on tty1.
         Starting Getty on tty1...
[  OK  ] Started Serial Getty on ttyPS0.
         Starting Serial Getty on ttyPS0...
[  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.

Ubuntu 15.04 linaro-developer ttyPS0

linaro-developer login: root (automatic login)

Last login: Mon Feb 16 20:55:09 UTC 2015 on ttyPS0
root@linaro-developer:~#
```

uboot启动倒计时之前可看到你的学号和姓名

```
reading uEnv.txt
398 bytes read in 9 ms (43 KiB/s)
Importing environment from SD your name and student number...
Hit any key to stop autoboot:  0
```