sudo nautilus

秘钥认证失败时使用

ssh-keygen -R 192.168.2.8

===运行minicom=========================================================

sudo chmod 777 /dev/ttyUSB0

sudo minicom -s 主界面 ctrl+A

===板子挂载主机NFS目录==================================================

主机

修改nfs服务器配置文件

sudo nano /etc/exports

在该文件最后添加一行： /home/xt500/aaa/nfs7010 \*(rw,sync,no\_root\_squash)

exportfs -r

service rpcbind start

service nfs start

或者/etc/init.d/nfs-kernel-server start

输入"showmount -e"命令可以查看 nfs server 路径

-------------------------------------------------

板子：

手动挂载，需要加参数 -o nolock。例如

mount -t nfs -o nolock 192.168.2.10:/home/xt500/aaa/nfs7010 /mnt

自动挂载：修改启动参数kernel bootargs ，将启动参数改为手动配置，然后增加参数

nfsroot=192.168.2.8:/home/xt500/aaa/nfs7010,proto=tcp,nfsvers=3,nolock ip=192.168.2.8

其中192.168.2.8为HOST端IP，192.168.2.8为TARGET端IP地址

===编译cgminer===========================================================

source /home/xt500/aaa/petalinux/mime7010\_prj/images/linux/sdk/environment-setup-cortexa9t2hf-neon-xilinx-linux-gnueabi

先编译curl

./configure --prefix=/home/xt500/ccc/out --host=arm-xilinx-linux-gnueabi CC=arm-xilinx-linux-gnueabi-gcc

然后编译cgminer

export LIBCURL\_CFLAGS=-I/home/xt500/ccc/out/include/curl

export LIBCURL\_LIBS=-L/home/xt500/ccc/out/lib

./configure --prefix=/home/xt500/aaa/cgminer/out --host=arm-xilinx-linux-gnueabi LIBCURL\_LIBS="-L/home/xt500/ccc/out/lib -lcurl" LIBS=-lz --enable-ants2

编译PetaLinux

source /tools/Xilinx/PetaLinux/2020.2/tool/settings.sh

---------创建工程----------------------------

cd ~/aaa/petalinux

petalinux-create --type project --template zynq --name mime7010\_prj

---------引用硬件信息----------------------------

复制 bit，xsa 到mime7010\_prj

cd ~/aaa/petalinux/mime7010\_prj

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

---------第1遍 配置----------------------------

petalinux-config

Subsystem AUTO Hardware Settings --->

Ethernet Settings ---> 改静态ip，成品要改随机MAC

Advanced bootable images storage Settings ---> 不用改，成品根据实际修改

Image Packaging Configuration --->

Root filesystem type (INITRD) --->改为sd

修改后vitis不能联机调试

(/tftpboot) tftpboot directory 改为 /home/xt500/aaa/tftpboot

petalinux-config -c u-boot

Boot media --->

[\*] Support for booting from SD/EMMC

SPL / TPL --->

[\*] Support SHA1 [\*] Support SHA256

petalinux-config -c kernel

Device Drivers ->

I2C support -> i2C Hardware Bus support ->

Xilinx I2C Controller Pressing 启用CONFIG\_I2C\_XILINX=y

+-SPI support--->

<\*> User mode SPI device driver support

Userspace I/O drivers

<\*> Userspace I/O platform driver with generic IRQ handing

<\*> Userspace platform driver with generic irq and dynamic memory

Power management options --->

Suspend to RAM and standby

将 system-user.dtsi 替换为：

/include/ "system-conf.dtsi"

/

&axi\_gpio\_0 {

compatible = "generic-uio";

};

petalinux-config -c rootfs

Filesystem Packages -> base -> i2c-tools -> i2c-tools

Image Features --->

[\*] ssh-server-dropbear │ │

[ ] ssh-server-openssh │ │

[\*] hwcodecs │ │

[ ] package-management │ │

-\*- debug-tweaks │ │

[\*] auto-login

petalinux-build

----安装到SD卡---------------------------

cd ~/aaa/petalinux/mime7010\_prj/images/linux

rm BOOT.BIN

petalinux-package --boot --format BIN --fsbl zynq\_fsbl.elf --fpga system.bit --u-boot

该命令将zynq\_fsbl.elf、test\_wrapper.bit、u-boot.elf三个文件合成一个BOOT.BIN，方法类似于在SDK中生成BOOT.BIN

rm -rfd /media/xt500/Fat32\_boot

cp boot.scr /media/xt500/Fat32\_boot

cp BOOT.BIN /media/xt500/Fat32\_boot

cp image.ub /media/xt500/Fat32\_boot

sudo rm -rfd /media/xt500/Ext4\_roofs

cp rootfs.cpio /media/xt500/Ext4\_roofs

cd /media/xt500/Ext4\_roofs

sudo pax -rvf rootfs.cpio

cd ~/aaa/petalinux/mime7010\_prj

----创建APP---------------------------------------------------------------------

petalinux-create -t apps --template c --name myapp-gpio --enable

----编译-SDK-----------------------------------------------------------------

cd ~/aaa/petalinux/mime7010\_prj

petalinux-build --sdk

petalinux-package --sysroot

===============================================================================

创建sdk输出目录，建立boot、image两个子目录，建立linux.bif文件，

拷贝boot.scr,system.bit,zynq\_fsbl.elf到boot，

拷贝image.ub,rootfs.cpio.gz到image，

了 sysroot，在 Vitis 中需要使用它来进行交叉编译。

使用petalinux-boot --jtag --fpga --bitstream petalinux\_wrapper.bit即可通过JTAG下载bit流到板子上

使用petalinux-boot --jtag --prebuilt 3即可把linux镜像下载到板子上。  
使用了：system.dtb u-boot.elf zImage zynq\_fsbl.elf

注意：使用QEMU和JTAG时，要先封装预建镜像，使用如下命令：  
petalinux-package --prebuilt --fpga  
之后，会自动创建pre-built文件夹，并自动添加一些文件。

执行petalinux-boot --qemu --kernel或者petalinux-boot --qemu --prebuilt 3 命令，启动kernel的效果如下图：

三、利用SD卡在硬件上启动petalinux镜像

将文件夹linux/images下的BOOT.BIN 和 image.ub 拷贝到SD卡的FAT32分区中。

设置启动方式为SD卡。

如果启动失败，提示找不到内核镜像。可能是bootcmd环境变量不正确。

四、使用TFTP启动硬件上的PETALINUX镜像

普通文件传输协议（TFTP）

TFTP 启动可节省大量时间，因为比 JTAG 启动快很多，而且不必为内核源中的每次更改都刷新镜像。

①　petalinux-boot --jtag --prebuilt 2

②　输入任意字符，使其停在UBOOT中。如下：

③　print serverip 检查TFTP服务器IP地址

Zynq> print serverip

serverip=192.168.157.129

④　set serverip ; saveenv 设置主机的IP地址

Zynq> set serverip 10.113.214.140 ;saveenv

Saving Environment to FAT...

writing uboot.env

done

1

2

3

4

⑤　run netboot 启动内核

Zynq> run netboot

\*\*\* ERROR: `ipaddr' not set

1

2

针对这个error，使用如下命令：

Zynq> set ipaddr 10.113.214.205

Zynq> run netboot

Using ethernet@e000b000 device

TFTP from server 10.113.214.252; our IP address is 10.113.214.205

Filename 'image.ub'.

Load address: 0x10000000

1

2

3

4

5

6

又会出现下面这个错误，是UBUNTU上的TFTP服务器还没有搭建。

Loading: T T T T T T T T T

TFTP server died; starting again

1

2

————————————————

setenv bootargs 'console=ttyPS0,115200 root=/dev/nfs rw nfsroot=192.168.2.125:/home/mklinux/nfs/zynq\_rootfs ip=192.168.2.116: 192.168.2.125:192.168.2.100:255.255.255.0::eth0:off'　　//通过网络挂载文件系统

===编译boa==========================================================

source /home/xt500/aaa/petalinux/mime7010\_prj/images/linux/sdk/environment-setup-cortexa9t2hf-neon-xilinx-linux-gnueabi

先编译curl

./configure --host=arm-xilinx-linux-gnueabi

然后make编译

1、将生成的boa可执行程序（在/src目录下）拷贝到开发板的/bin目录下，在开发板/etc目录下建boa目录，将boa.conf拷贝到板子的/etc/boa目录。.

2、修改/etc/boa下的boa.conf文件：

User（48行）默认nobody，改为0

Group（49行）默认nogroup，改为0

3、创建日志文件所在目录/var/log/boa，

创建HTML文档的主目录/var/www，

创建CGI脚本所在目录/var/www/cgi-bin/，

在/var/www中放置一个index.html文件。

上述目录若不存在，均需手动创建，这些都是boa运行需要的目录。

index.html就是打开网址时的网页。

将虚拟机/etc目录下的mime.types文件到开发板/etc目录

在开发板运行boa:

#/bin/boa

然后就可以通过其他机器访问了。

http://192.168.0.58

就可以访问到你放置的那个index页面了。

第二部分 测试CGI

https://blog.csdn.net/hnxyxiaomeng/article/details/86029449参考了这篇文章。

一、创建一个test.c文件，内容如下

#include <stdlib.h>

#include <stdio.h>

int main()

{

printf("Content-type: text/html\n\n") ;

printf("\n") ;

printf("CGI Output\n") ;

printf("\n") ;

printf("Hello, world.\n") ;

printf("\n") ;

printf("\n") ;

exit(0) ;

return 1;

}

二、编译

arm-linux-gnueabihf-gcc -o test.cgi test.c

错误，要改为 $CC -o test.cgi test.c

三、拷贝到 ZYNQ开发板

将生成的test.cgi拷贝到移植Boa时在开发板文件系统中建立的/var/www/cgi-bin中，并修改文件权限为777。

在/etc/boa/boa.conf中找到ScriptAlias（最后一行）

本来为：ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/

修改为：ScriptAlias /cgi-bin/ /var/www/cgi-bin/

四、验证

打开网页，http://10.113.214.205/cgi-bin/test.cgi， 即可看到test.c中输出的字符串。

第三部分 移植CGIC

一、从github下载CGIC源码，解压。

git clone https://github.com/boutell/cgic.git

二、修改Makefile文件。

CFLAGS=-g -Wall

#CC=gcc

#AR=ar

#RANLIB=ranlib

LIBS=-L./ -lcgic

all: libcgic.a cgictest.cgi capture

install: libcgic.a

cp libcgic.a /usr/local/lib

cp cgic.h /usr/local/include

@echo libcgic.a is in /usr/local/lib. cgic.h is in /usr/local/include.

libcgic.a: cgic.o cgic.h

rm -f libcgic.a

$(AR) rc libcgic.a cgic.o

$(RANLIB) libcgic.a

#mingw32 and cygwin users: replace .cgi with .exe

cgictest.cgi: cgictest.o libcgic.a

$(CC) cgictest.o -o cgictest.cgi ${LIBS}

capture: capture.o libcgic.a

$(CC) capture.o -o capture ${LIBS}

clean:

rm -f \*.o \*.a cgictest.cgi capture cgicunittest

test:

$(CC) -D UNIT\_TEST=1 cgic.c -o cgicunittest

./cgicunittest

三、编译

make

1

生成了CGIC库libcgic.a。还有capture调试程序、cgitest.cgi测试程序用来验证生成CGIC库的正确性。

四、拷贝到 ZYNQ开发板

将libcgic.a拷贝到开发板的/lib文件夹下，将capture和cgictest.cgi拷贝到开发板的/var/www/cgi-bin目录下，并修改capture和cgictest.cgi的权限为777。

五、验证

在Windows的浏览器地址栏输入http://192.168.2.8/cgi-bin/cgictest.cgi，可以看到页面，CGIC库和测试脚本都移植成功。该页面中含有CGIC的常用控件，通过分析代码可看出怎么使用控件。