zynq

1. 使用axi pkg ip

当使用axi接口的自定义IP时，打包的时候可能要记得在.h文件中添加如下内容

```
#include "xil_types.h"
#include "xstatus.h"
#include "xil_io.h"
```

不然可能在使用define macro访问AXI IP内容的register时，会出现c编译错误。 -- 会不会添加一个axi gpio ip也能解决， 或者手动在程序.h文件里面把#include “xil_io.h”给加上。

2. petalinux

2.1 create project

```
source /xilinx/settings.sh

petalinux-create -t project --template zynq -n xxx_name
petalinux-config --get-hw-description .
```

2.2 create modules

相当于创建linux模块驱动

```
petalinux-create -t modules -n xxx_module --enable
```

2.3 create app

创建一个linux平台下的app可执行程序

```
petalinux-create -t apps -n xxx_app  --enable
```

然后进入recipes-apps/xxx_app目录，修改files/xxx_app.c文件，默认是一个hello world!打印程序。

2.4 build

```
petalinux-build -c rootfs
petalinux-build -c xxx_module
petalinux-build -c xxx_app -x do_install
```

```
petalinux-build
```

2.5 app单独编译&调试

```
petalinux-build -c xxx_app -x do_compile    #对指定app进行编译,编译后生成的app可执行文件在以下位置
ls build/tmp/work/cortexa9hf-neon-xilinx-linux-gnueabi/myapp3/1.0-r0/myapp3

#然后可以将此文件通过scp的方式copy到zynq设备，修改权限chmod 777 ./myapp3 执行程序./myapp3
#这样的方式不用很次app修改后都烧写到flash 通过scp的方式加快app开发调试速度。
```

2.6 package to BOOT.BIN

```
petalinux-package --boot --fsbl zynq_fsbl.elf --u-boot --kernel --fpga
system.bit --force
# --boot 打包成BOOT.BIN文件
#--输入fsdb文件
# --u-boot 输入默认u-boot文件， 一般是指u-boot.elf
# --kernel 输入petalinux内核， 默认是指image.ub
# --fpga 指定FPGA bits文件。
#一般是建议先不用--kernel选项生成BOOT.BIN文件，然后将BOOT.BIN文件和image.ub一起copy到SD卡，从SD卡启动看看程序效果，程序稳定后可以再考虑弄成为flash启动，这样调试速度会快一些。
```

2.7 一个简单的驱动开发例程——GPIO流水灯（Petalinux部分）

https://blog.csdn.net/u013029731/article/details/85042431/

2.8 实现app开机启动

参考ug1144, Ch.7: Customizing the Rootfs

```
实现开机启动
本章节内容参考UG1144
（1）创建myapp-init应用

cd <plnx-proj-proot>
petalinux-create -t apps --template install -n myapp-init --enable

（2）修改myapp-init.bb配置文件
配置文件的位置在：
```

project-spec/meta-user/recipes-apps/myapp-init/myapp-init.bb
修改文件内容为：

```
#
# This file is the myapp-init recipe.
#
SUMMARY = "Simple myapp-init application"
SECTION = "PETALINUX/apps"
LICENSE = "MIT"
LIC_FILES_CHKSUM ="file://${COMMON_LICENSE_DIR}/
MIT;md5=0835ade698e0bcf8506ecda2f7b4f302"
SRC_URI = "file://myapp-init \
 "
S = "${WORKDIR}"
FILESEXTRAPATHS_prepend := "${THISDIR}/files:"
inherit update-rc.d
INITSCRIPT_NAME = "myapp-init"
INITSCRIPT_PARAMS = "start 99 S ."
do_install() {
 install -d ${D}${sysconfdir}/init.d
 install -m 0755 ${S}/myapp-init ${D}${sysconfdir}/init.d/myapp-init
}
FILES_${PN} += "${sysconfdir}/*"
```

3）修改myapp-init脚本文件内容
脚本文件的位置在：

project-spec/meta-user/recipes-apps/
myapp-init/files/myapp-init

本文修改的内容为加载xilinx-axidma.ko module和在后台启动程序xxx_app

```
#!/bin/sh

cd /lib/modules/5.4.0-xilinx-v2020.1/extra
insmod xxx_module.ko

cd /usr/bin
./xxx_app &
## 注意，最好在此加上&，让其后台执行，不然系统会把当前命令执行完之后再进行下面的任务，
## 比如下面的任务是配置网络，这对于需要网络的开机启动尤其重要
```

完成后进行petalinux-build使用新生成的镜像，下次就可以开机自启动了。

2.9 zynq设备上运行指定驱动的app程序

在zynq fpga运行过程。

```
root@petalinux_boot_from_flash:/lib/modules/4.14.0-xilinx-v2018.3/extra# pwd
/lib/modules/4.14.0-xilinx-v2018.3/extra
```

```
root@petalinux_boot_from_flash:/lib/modules/4.14.0-xilinx-v2018.3/extra# ls
blink.ko

modprobe blink.ko     #  加载驱动
mknod /dev/blink_Dev c 245 0   # 标dev, 这个信息根据modprobe的提示信息输入
ls /dev/blink_Dev    #  已经有dev
blinkapp    # 运用app程序
```