

骑驴玩儿漂移-FPGA之PYNQ - Z2 入门学习笔记1

今天收到了PYNQ的板子，感谢赛灵思大佬的赞助，我有幸尝试一下FPGA的板子，之前拿到一块来自苏老师的小脚丫的板子，那是非常精致啊，但是没有入门，对FPGA很有兴趣，听大家说PYNQ支持python而且非常强大，我就想试试，正好遇到群里的赛灵思大佬，有幸得以尝试一下FPGA，不过我感觉我根本没用到fpga，都是在linux和python的边缘游走。哈哈。

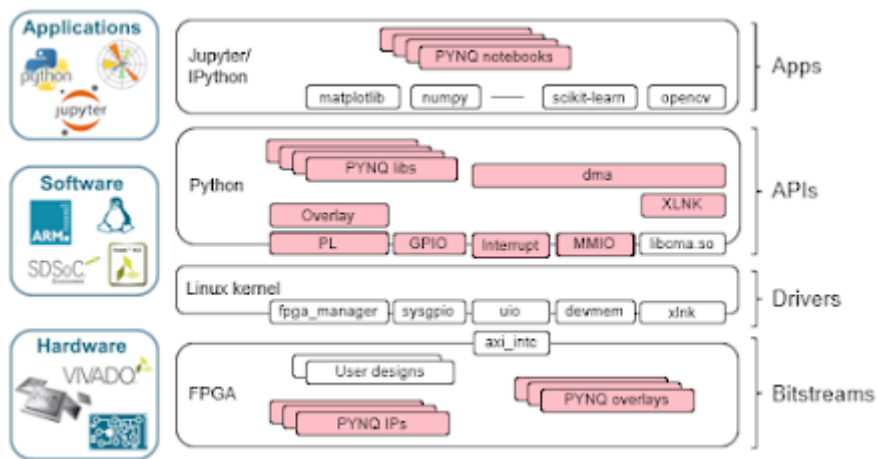
好了，先做个介绍：

PYNQ的来历

PYNQ是Xilinx公司的开源项目[®]，可以很容易地设计与赛灵思ZYNQ嵌入式系统[®] 系统上级芯片（SoC）。使用Python语言和库，设计人员可以利用Zynq中可编程逻辑和微处理器的优势来构建更强大，更令人兴奋的嵌入式系统。PYNQ用户现在可以创建高性能的嵌入式应用程序。

特色：

- 并行硬件执行
- 高帧率视频处理
- 硬件加速算法
- 实时信号处理
- 高带宽IO
- 低延迟控制



PYNQ旨在供各种设计人员和开发人员使用，包括：

希望利用Zynq和可编程硬件功能而无需使用ASIC式设计工具来设计硬件的软件开发人员。

系统架构师需要一个简单的软件界面和框架来快速进行原型设计和Zynq设计的开发。

希望他们的设计被最广泛的受众使用的硬件设计师。

关键技术：

Jupyter Notebook 是一个基于浏览器的交互式计算环境。可以创建**Jupyter**笔记本文档，包括实时代码，交互式小部件，绘图，说明文本，方程式，图像和视频。 可以使用Python在**Jupyter** Notebook中轻松编程支持PYNQ的Zynq板。

使用Python，开发人员可以在可编程逻辑上使用硬件库和覆盖。 硬件库或覆盖可以加速Zynq板上运行的软件，并自定义硬件平台和接口。

PYNQ映像是可引导的Linux映像，包括 `pynq` Python包和其他开源包。



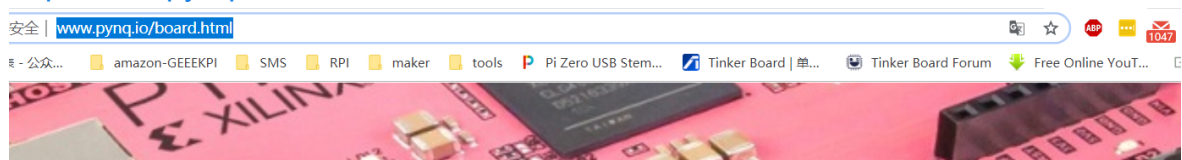
软件需求

Jupyter笔记本 界面基于浏览器。 只需一个 兼容的Web浏览器 即可开始使用Python 编程 PYNQ。 为了获得更高的性能，您还可以将C / C + +与PYNQ结合使用。赛灵思SDK软件开发环境是免费提供的。您还可以使用第三方软件开发工具。 可以使用标准Xilinx和第三方硬件设计工具创建新的 硬件库和叠加层。Xilinx Vivado的免费WebPACK版本完全支持许多Zynq主板。

上面都是官方说明

我来写一下我的操作步骤吧：

1. 访问：<https://github.com/Xilinx/Pynq>
2. 得到信息说明需要在官方下载PYNQ的镜像文件，然后下载镜像文件需要注册账号。
3. <http://www.pynq.io/board.html> 这里有下载文件的路径：



4.

Development Boards

There are currently four boards officially supported by PYNQ: **Pynq-Z1 from Digilent**, **Pynq-Z2 from TUL**, **ZCU104 from Xilinx**, and **ZCU111 from Xilinx**.

Downloadable PYNQ images

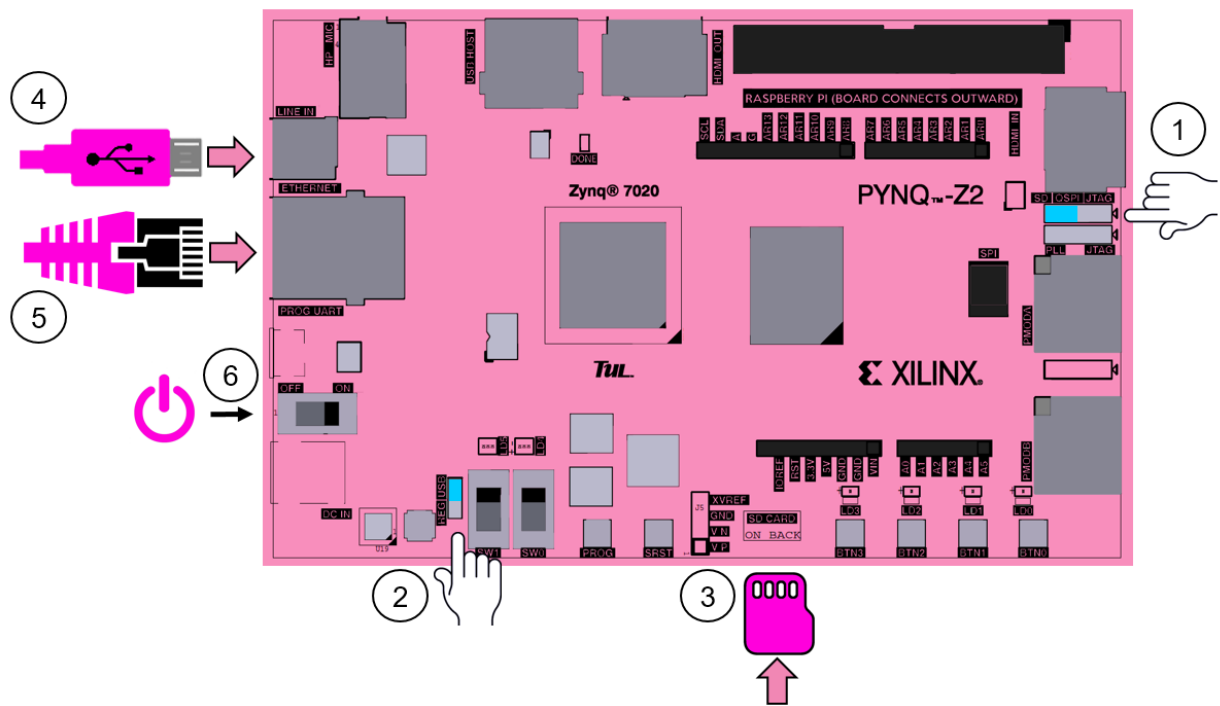
Images for supported boards are available via the links below. The image includes board specific example overlays and Jupyter notebooks.

 - [PYNQ-Z1 v2.4 PYNQ image](#)
 - [PYNQ-Z2 v2.4 PYNQ image](#)
 - [ZCU104 v2.4 PYNQ image](#)
 - [ZCU111 v2.4 PYNQ image](#)

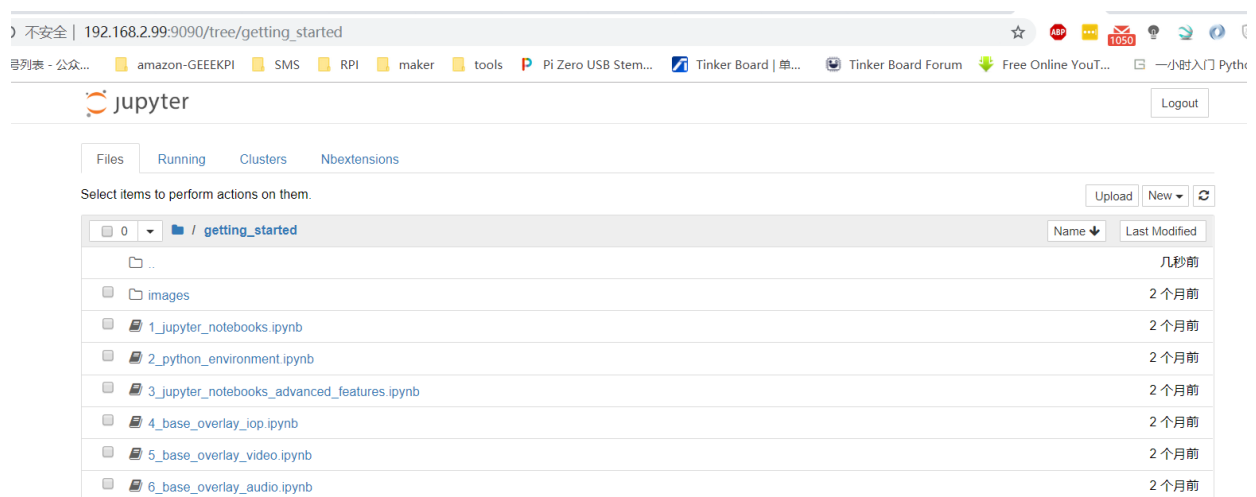
Community boards

The **Avnet Ultra96** (Zynq UltraScale+) also supports PYNQ. A PYNQ image and documentation for the Ultra96 are available from Avnet:
5. 注册账号
6. <https://www.xilinx.com/regISTRATION/create-account.html>
7. https://pynq.readthedocs.io/en/latest/getting_started.html

烧录镜像到卡上，然后按照下图调试好Jumper的位置



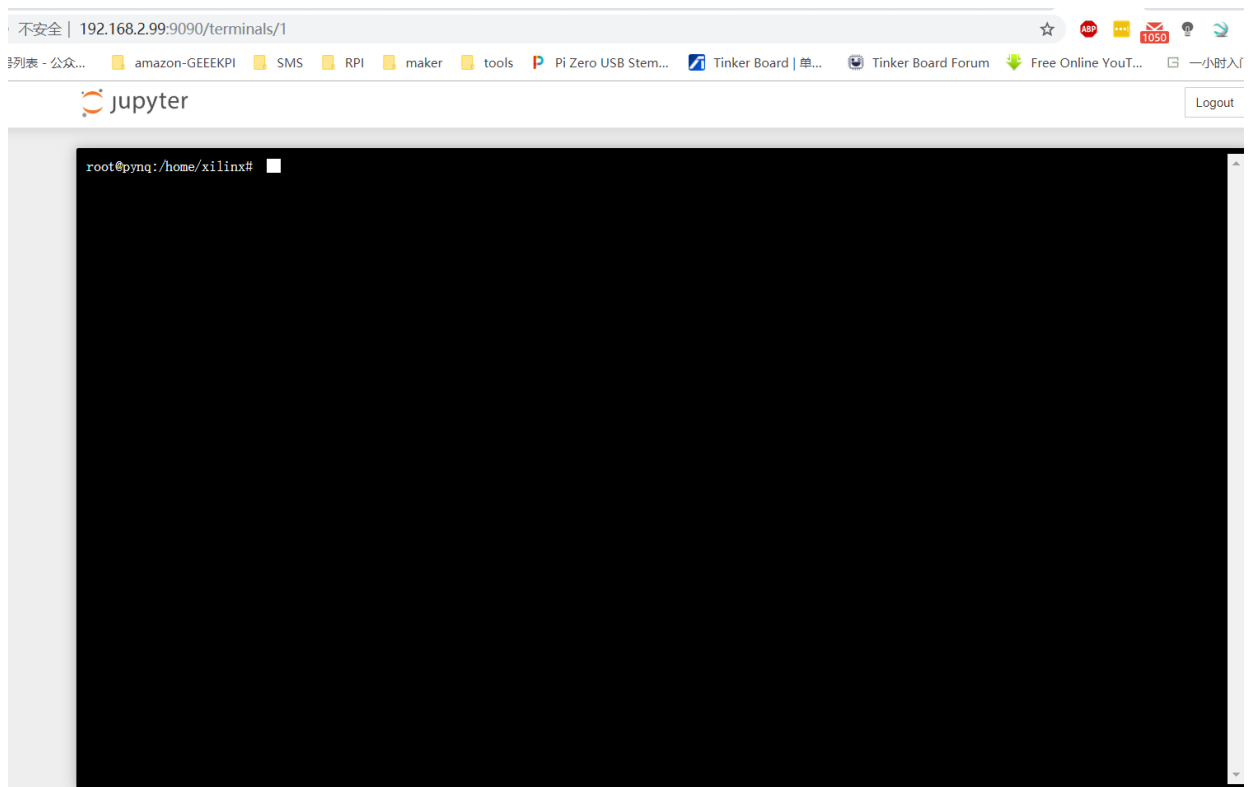
然后上电就启动了。将电源开关滑动到ON。红色 LED 会立刻上来就确认板上有电源。几秒钟后，黄色/绿色/完成 LED 将亮起，表示设备正常运行。一会儿你就会看到两个蓝色 LED，就是板子上的 LD4，LD5 亮一下，和四个黄绿色 LED 同时闪烁。系统现已启动并可供使用。刚开始我选择的方式是：一根网线连着 PYNQ 和我的电脑，然后电脑本地网卡 IP 地址设置为 192.168.2.1/24，因为 xilinx 的板子默认的地址和端口号是：192.168.2.99:9090 通过谷歌浏览器可以顺利登陆并看到下图：



我们尝试一下开个新的终端：



然后看到下面的效果：



```
root@pynq:/home/xilinx#
root@pynq:/home/xilinx#
root@pynq:/home/xilinx# ls
jupyter_notebooks  pynq  REVISION
root@pynq:/home/xilinx# cat REVISION
Release 2019_02_21 2382a55
root@pynq:/home/xilinx#
root@pynq:/home/xilinx# w
 01:30:59 up 14 min,  1 user,  load average: 0.16, 0.15, 0.10
USER            TTY      FROM          LOGIN@   IDLE   JCPU   PCPU   WHAT
xilinx          ttyPS0            -          01:17   13:58    0.41s  0.36s  -bash
root@pynq:/home/xilinx# who am i
root@pynq:/home/xilinx#
root@pynq:/home/xilinx# whoami
root
root@pynq:/home/xilinx#
root@pynq:/home/xilinx# id
uid=0(root) gid=0(root) groups=0(root)
root@pynq:/home/xilinx#
root@pynq:/home/xilinx# groups
root
root@pynq:/home/xilinx# uname -a
Linux pynq 4.14.0-xilinx-v2018.3 #1 SMP PREEMPT Thu Feb 21 00:11:14 UTC 2019 armv7l armv7l armv7l GNU/Linux
root@pynq:/home/xilinx#
root@pynq:/home/xilinx# cat /proc/cpuinfo
processor       : 0
model name     : ARMv7 Processor rev 0 (v7l)
BogoMIPS      : 650.00
Features      : half thumb fastmult vfp edsp neon vfpv3 tls vfpd32
CPU implementer : 0x41
CPU architecture: 7
CPU variant   : 0x3
CPU part      : 0xc09
CPU revision  : 0

processor       : 1
model name     : ARMv7 Processor rev 0 (v7l)
BogoMIPS      : 650.00
Features      : half thumb fastmult vfp edsp neon vfpv3 tls vfpd32
CPU implementer : 0x41
CPU architecture: 7
CPU variant   : 0x3
```

简单操作了一下，还是很爽滑的。

```
root@pynq:/home/xilinx#
root@pynq:/home/xilinx# free -m
              total        used        free      shared  buff/cache   available
Mem:           496            122          234           0            1         140
Swap:          1023             0          1023
root@pynq:/home/xilinx#
root@pynq:/home/xilinx# df -Th
Filesystem      Type      Size  Used Avail Use% Mounted on
/dev/root       ext4      7.1G  4.7G  2.1G  70% /
devtmpfs        devtmpfs  184M   0    184M   0% /dev
tmpfs           tmpfs     249M   0    249M   0% /dev/shm
tmpfs           tmpfs     249M  1.3M  243M   1% /run
tmpfs           tmpfs     5.0M   0    5.0M   0% /run/lock
tmpfs           tmpfs     249M   0    249M   0% /sys/fs/cgroup
tmpfs           tmpfs     50M    0    50M   0% /run/user/1000
root@pynq:/home/xilinx#
```

检查一下支持的python库有哪些：

```
root@pynq:/home/xilinx#
root@pynq:/home/xilinx#
root@pynq:/home/xilinx# pip3 freeze
2ping==4.1
alabaster==0.7.8
attrs==17.4.0
Babel==2.4.0
bleach==2.1.2
certifi==2018.1.18
cffi==1.11.5
chardet==3.0.4
Click==7.0
cycler==0.10.0
Cython==0.26.1
dash==0.21.1
dash-core-components==0.23.0
dash-html-components==0.11.0
dash-renderer==0.13.0
decorator==4.1.2
deltatigma==0.2.2
docutils==0.14
entrypoints==0.2.3.post1
Flask==1.0.2
Flask-Compress==1.4.0
html5lib==0.999999999
httplib2==0.9.2
idna==2.6
imagesize==0.7.1
imutils==0.5.2
ipykernel==4.8.2
ipython==5.5.0
ipython-genutils==0.2.0
ipywidgets==7.4.2
itsdangerous==1.1.0
Jinja2==2.10
jsonschema==2.6.0
jupyter-client==5.2.2
jupyter-contrib-core==0.3.3
jupyter-contrib-nbextensions==0.5.1
jupyter-core==4.4.0
jupyter-highlight-selected-word==0.2.0
jupyter-latex-envs==1.4.6
jupyter-nbextensions-configurator==0.4.1
jupyterlab==0.35.4
jupyterlab-server==0.2.0
```

```
ipython-genutils==0.2.0
ipywidgets==7.4.2
itsdangerous==1.1.0
Jinja2==2.10
jsonschema==2.6.0
jupyter-client==5.2.2
jupyter-contrib-core==0.3.3
jupyter-contrib-nbextensions==0.5.1
jupyter-core==4.4.0
jupyter-highlight-selected-word==0.2.0
jupyter-latex-envs==1.4.6
jupyter-nbextensions-configurator==0.4.1
jupyterlab==0.35.4
jupyterlab-server==0.2.0
lxml==4.2.1
MarkupSafe==1.0
matplotlib==2.1.1
mistune==0.8.3
mpmath==1.0.0
nbconvert==5.3.1
nbformat==4.4.0
nbsphinx==0.3.1
nbwavedrom==0.2.0
netifaces==0.10.4
networkx==1.11
notebook==5.2.2
numexpr==2.6.4
numpy==1.13.3
pandas==0.24.1
pandocfilters==1.4.2
pexpect==4.2.1
pickleshare==0.7.4
Pillow==5.1.0
plotly==3.6.1
pluggy==0.6.0
ply==3.11
prompt-toolkit==1.0.15
psutil==5.4.2
py==1.5.2
pycparser==2.18
pycurl==7.43.0.1
pyeda==0.28.0
Pygments==2.2.0
pygobject==3.26.1
pygraphviz==1.4rc1
```

```
psutil==5.4.2
py==1.5.2
pycparser==2.18
pycurl==7.43.0.1
pyeda==0.28.0
Pygments==2.2.0
pygobject==3.26.1
pygraphviz==1.4rc1
pynq==2.4
pyparsing==2.2.0
pytest==3.3.2
pytest-sourceorder==0.5
python-apt==1.6.0
python-dateutil==2.6.1
pytz==2018.3
PyWavelets==0.5.1
PyYAML==3.12
pyzmq==16.0.2
requests==2.18.4
requests-unixsocket==0.1.5
retrying==1.3.3
rise==5.2.0
roman==2.0.0
scikit-image==0.13.1
scipy==0.19.1
simplegeneric==0.8.1
six==1.11.0
Sphinx==1.6.7
sphinx-rtd-theme==0.4.3
SQLAlchemy==1.1.11
ssh-import-id==5.7
sympy==1.1.1
terminado==0.7
testpath==0.3.1
tornado==4.5.3
traitlets==4.3.2
transitions==0.5.3
unattended-upgrades==0.1
urllib3==1.22
uvloop==0.8.1
wcwidth==0.1.7
webencodings==0.5
Werkzeug==0.14.1
widgets-nbextension==3.4.2
root@nynq:/home/xilinx#
```

简单尝试写了个伪随机代码：

```

root@pynq:/home/xilinx#
root@pynq:/home/xilinx#
root@pynq:/home/xilinx#
root@pynq:/home/xilinx# cat test.py
#!/usr/bin/env python3
from random import randint
import time as t

for i in range(0, 5):
    s = i*randint(0, 99999999)/ t.time()
    print(s)
    t.sleep(1)

root@pynq:/home/xilinx# python3 test.py
0.0
0.03760030628714377
0.005912715598996392
0.15408853706991532
0.09322528039786672
root@pynq:/home/xilinx#

```

测试一下看看IP地址，路由信息啥的。

```

root@pynq:/home/xilinx# route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
192.168.2.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@pynq:/home/xilinx# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet6 fe80::205:6bff:fe00:c870 prefixlen 64 scopeid 0x20<link>
    ether 00:05:6b:00:c8:70 txqueuelen 1000 (Ethernet)
    RX packets 3082 bytes 215665 (215.6 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2569 bytes 1083186 (1.0 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 29 base 0xb000

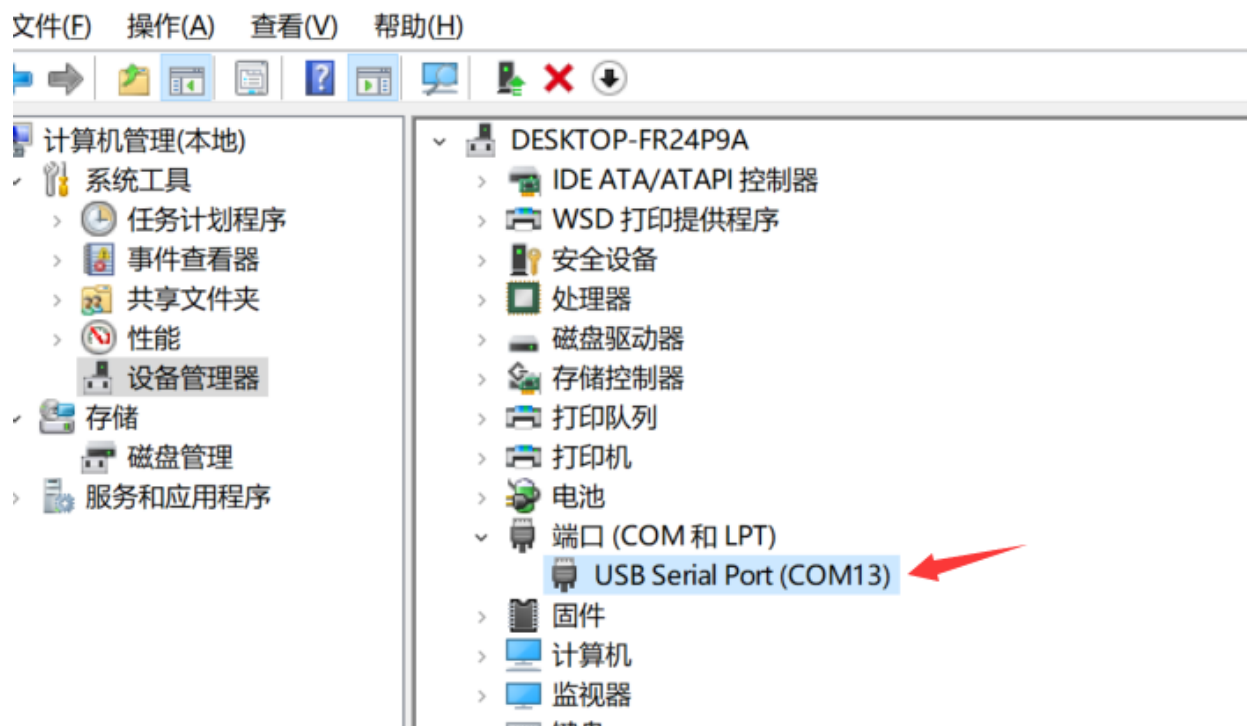
eth0:1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.2.99 netmask 255.255.255.0 broadcast 192.168.2.255
    ether 00:05:6b:00:c8:70 txqueuelen 1000 (Ethernet)
    device interrupt 29 base 0xb000

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 2704 bytes 173192 (173.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2704 bytes 173192 (173.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

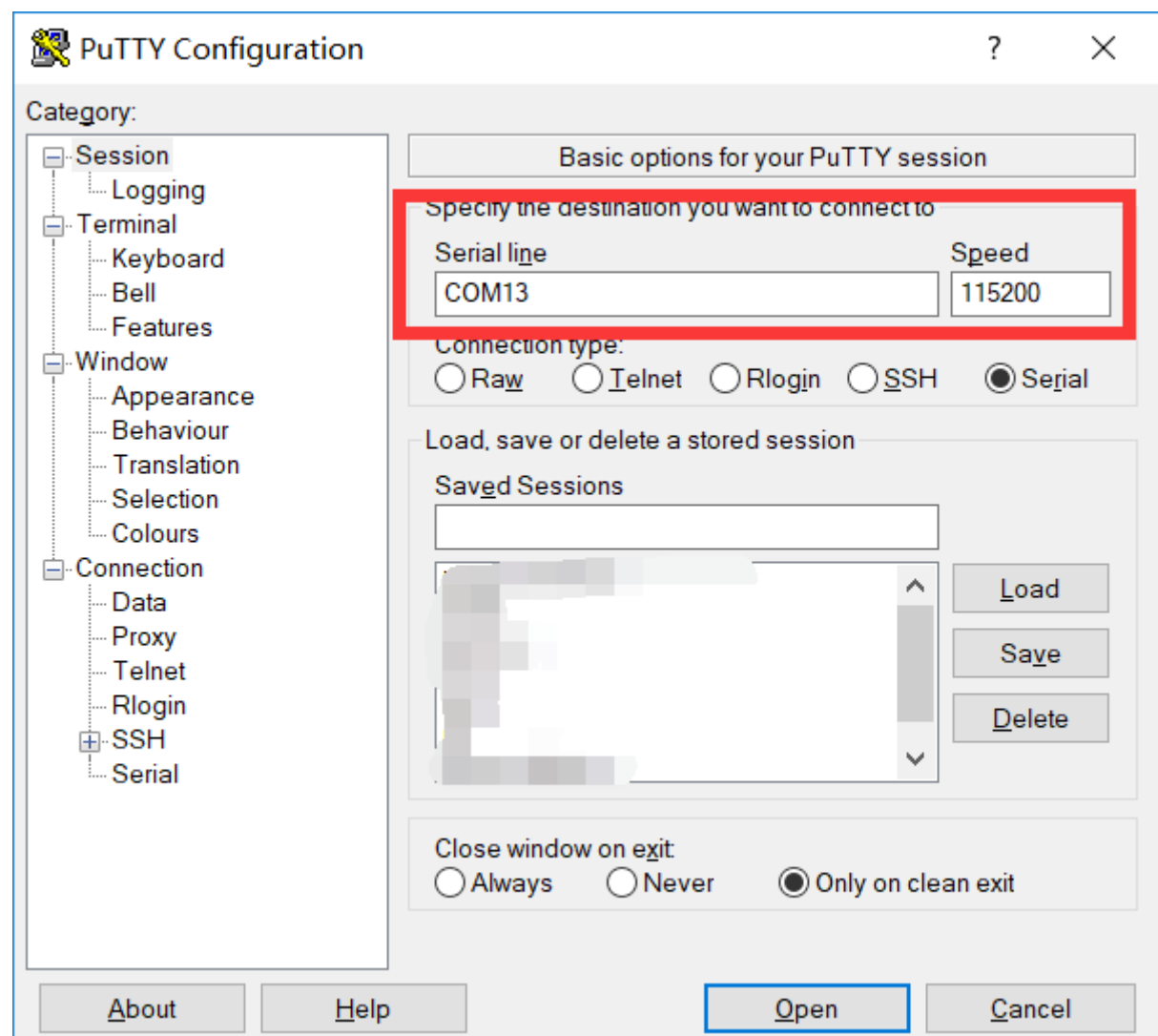
root@pynq:/home/xilinx# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:05:6b:00:c8:70 brd ff:ff:ff:ff:ff:ff
    inet 192.168.2.99/24 brd 192.168.2.255 scope global eth0:1
        valid_lft forever preferred_lft forever
    inet6 fe80::205:6bff:fe00:c870/64 scope link
        valid_lft forever preferred_lft forever
3: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/sit 0.0.0.0 brd 0.0.0.0
root@pynq:/home/xilinx#

```

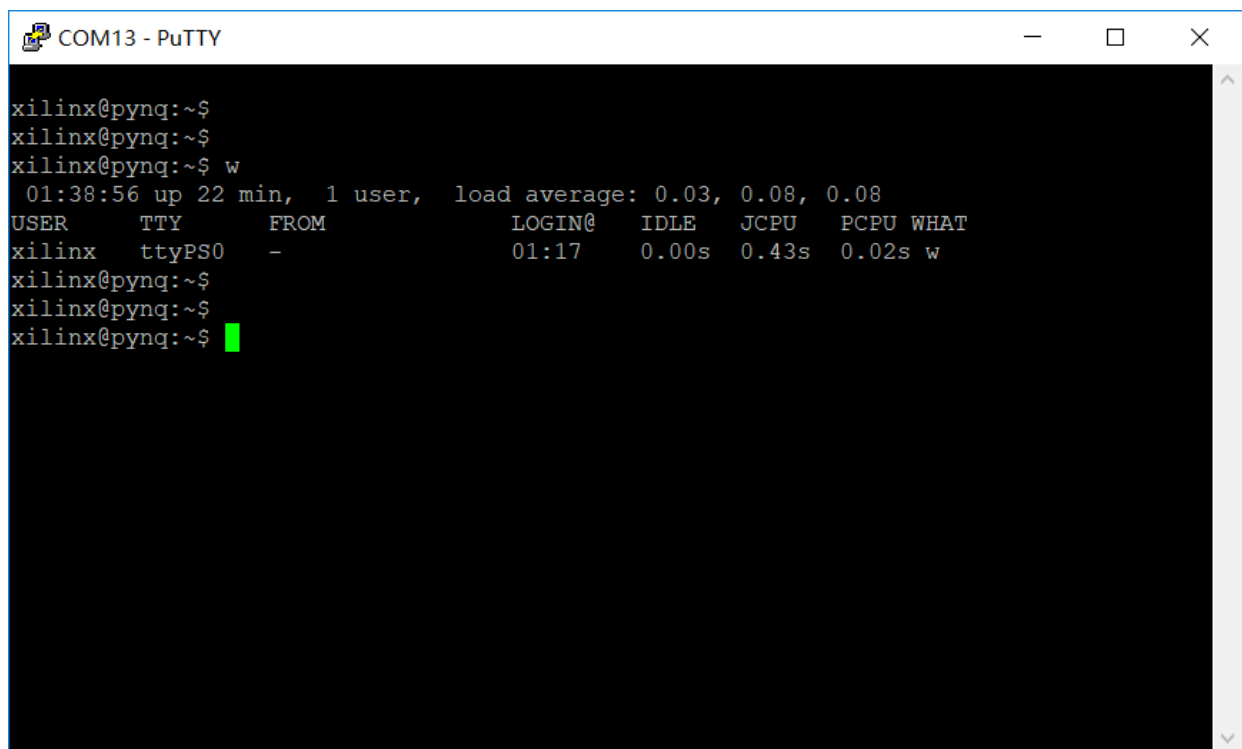
都正常。



因为目前PYNQ的板子是通过USB串口连接到电脑的，也就意味着我应该可以通过串口也能连上来。简单配置putty，使用串口，刚才看到的是com13，波特率我猜115200，尝试一下先。



发现是可以的，而且很顺畅，没有卡顿，没有乱码，哈利路亚。



```
xilinx@pynq:~$
xilinx@pynq:~$
xilinx@pynq:~$ w
 01:38:56 up 22 min,  1 user,  load average: 0.03, 0.08, 0.08
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT
xilinx    ttyPS0    -                01:17    0.00s  0.43s  0.02s  w
xilinx@pynq:~$
xilinx@pynq:~$
xilinx@pynq:~$
```

August说看了下gameboy代码，应该跟audio无关，感觉可以在PYNQ-Z2上跑。

<https://github.com/kastnerkyle/gb/tree/master/verilog> 这个链接

To interact with the IP first we need to load the overlay containing the IP.

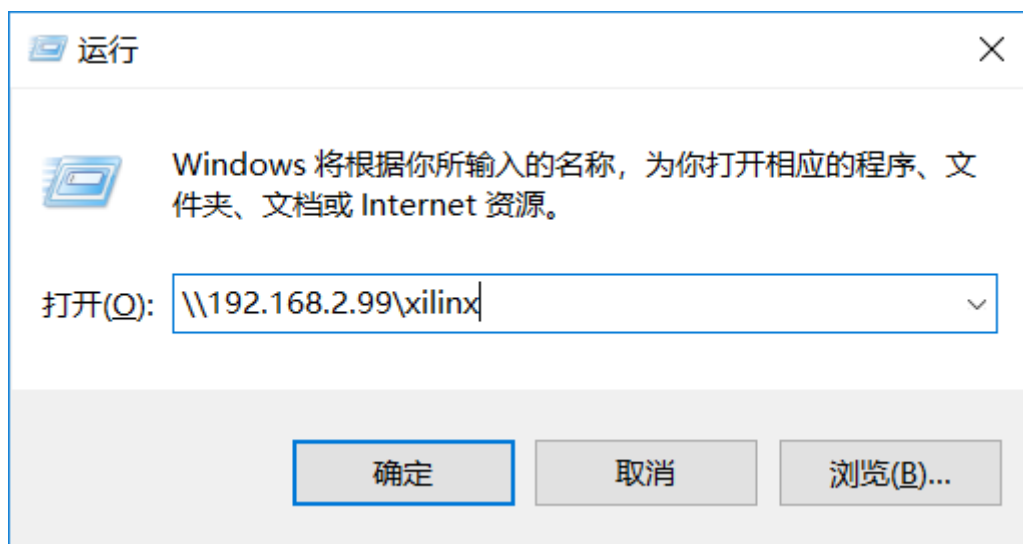
```
[1]: from pynq import Overlay

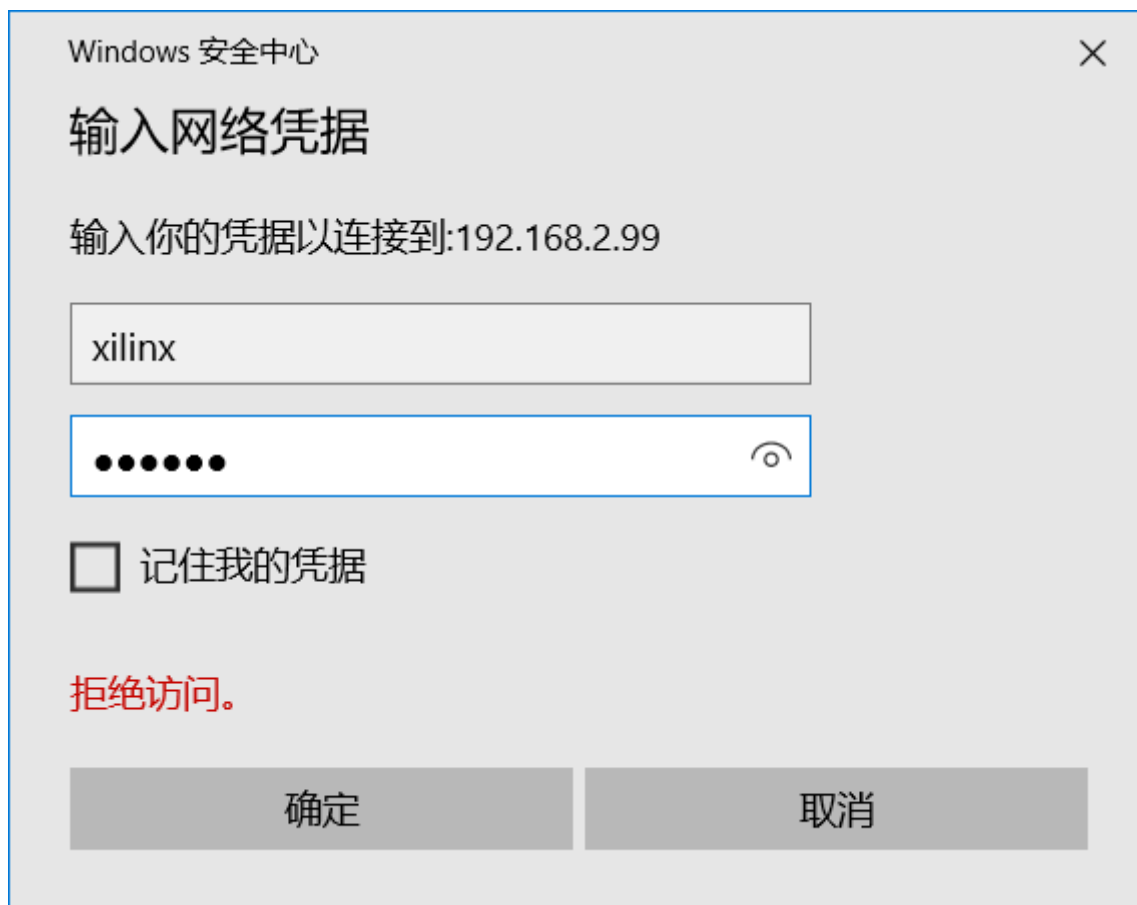
overlay = Overlay('/home/xilinx/tutorial_1.bit')
```

Creating the overlay will automatically download it. We can now use a question mark to find out what is in the overlay.

```
[2]: overlay?
```

我突然想起刚才尝试的时候发现有samba服务，尝试了一下果然发现有共享：





用户名密码都应该是xilinx,今天好幸运，果然都猜对了。

网络 > 192.168.2.99 > xilinx

名称	修改日期	类型	大小
.cache	2019/2/21 9:13	文件夹	
.gnupg	2019/2/21 9:13	文件夹	
jupyter_notebooks	2019/2/21 11:16	文件夹	
pynq	2019/2/21 9:17	文件夹	
.bash_logout	2019/2/21 9:13	Bash Logout 源...	1 KB
.bashrc	2019/2/21 9:13	Bash RC 源文件	4 KB
.profile	2019/2/21 9:13	PROFILE 文件	1 KB
REVISION	2019/2/21 11:17	文件	1 KB
sds_trace_data.dat	2019/2/21 10:07	DAT 文件	1 KB
test.py	2019/2/21 9:20	JetBrains PyChar...	1 KB
test_gpio.py	2019/2/21 9:29	JetBrains PyChar...	1 KB

从github上下载下来直接传上去就好了，非常方便。

名称	修改日期	类型	大小
.cache	2019/2/21 9:13	文件夹	
.gnupg	2019/2/21 9:13	文件夹	
jupyter_notebooks	2019/2/21 11:16	文件夹	
pynq	2019/2/21 9:17	文件夹	
.bash_logout	2019/2/21 9:13	Bash Logout 源...	1 KB
.bashrc	2019/2/21 9:13	Bash RC 源文件	4 KB
.profile	2019/2/21 9:13	PROFILE 文件	1 KB
REVISION	2019/2/21 11:17	文件	1 KB
sds_trace_data.dat	2019/2/21 10:07	DAT 文件	1 KB
test.py	2019/2/21 9:20	JetBrains PyChar...	1 KB
test_gpio.py	2019/2/21 9:29	JetBrains PyChar...	1 KB
gb-master.zip	2019/4/17 18:06	压缩(zipped)文件...	3,309 KB

```

root@pynq:/home/xilinx#
root@pynq:/home/xilinx#
root@pynq:/home/xilinx#
root@pynq:/home/xilinx# ls
gb-master.zip  jupyter_notebooks  pynq  REVISION  sds_trace_data.dat  test_gpio.py  test.py
root@pynq:/home/xilinx#
root@pynq:/home/xilinx#
root@pynq:/home/xilinx#
root@pynq:/home/xilinx#

```

下面开始解压，然后测试一下

```

root@pynq:/home/xilinx#
root@pynq:/home/xilinx#
root@pynq:/home/xilinx# mkdir gb
root@pynq:/home/xilinx# cd gb
root@pynq:/home/xilinx/gb# mv ../gb-master.zip .
root@pynq:/home/xilinx/gb#
root@pynq:/home/xilinx/gb# ls
gb-master.zip
root@pynq:/home/xilinx/gb# unzip gb-master.zip
Archive:  gb-master.zip
93c4502b4003771f8743a62f3f66b899013c83da
  creating: gb-master/
  inflating: gb-master/LICENSE
  inflating: gb-master/Makefile
  inflating: gb-master/README.md
  inflating: gb-master/bootrom.h
  inflating: gb-master/gameboy.c
  inflating: gb-master/gameboy.h
  inflating: gb-master/gameboy.py
  creating: gb-master/gifs/
  inflating: gb-master/gifs/Micro_Machines_(USA,_Europe).gb_124856.gif
  inflating: gb-master/gifs/Micro_Machines_(USA,_Europe).gb_184680.gif
  inflating: gb-master/gifs/Micro_Machines_(USA,_Europe).gb_244488.gif
  inflating: gb-master/gifs/Micro_Machines_(USA,_Europe).gb_304912.gif
  inflating: gb-master/gifs/Micro_Machines_(USA,_Europe).gb_364336.gif
  inflating: gb-master/gifs/Micro_Machines_(USA,_Europe).gb_423184.gif
  inflating: gb-master/gifs/Micro_Machines_(USA,_Europe).gb_484080.gif
  inflating: gb-master/gifs/Micro_Machines_(USA,_Europe).gb_63216.gif
  inflating: gb-master/gifs/wario_walking.gif
  inflating: gb-master/main_lcd.c
  inflating: gb-master/make_gifs.py
  inflating: gb-master/typedefs.h
  creating: gb-master/verilog/
  creating: gb-master/verilog/.Xil/
  inflating: gb-master/verilog/.Xil/top_propImpl.xdc
  inflating: gb-master/verilog/.top_pynq.v.swp

```

用unzip解压。

```

gb-master  gb-master.zip
root@pynq:/home/xilinx/gb# cd gb-master
root@pynq:/home/xilinx/gb/gb-master# ls
bootrom.h  gameboy.c  gameboy.h  gameboy.py  gifs  LICENSE  main_lcd.c  Makefile  make_gifs.py  README.md  typedefs.h  verilog  wario_walking.gb
root@pynq:/home/xilinx/gb/gb-master#
root@pynq:/home/xilinx/gb/gb-master# ls -l
total 176
-rw-r--r-- 1 root root 1610 Dec 17 19:38 bootrom.h
-rw-r--r-- 1 root root 28484 Dec 17 19:38 gameboy.c
-rw-r--r-- 1 root root 3392 Dec 17 19:38 gameboy.h
-rw-r--r-- 1 root root 3272 Dec 17 19:38 gameboy.py
drwxr-xr-x 2 root root 4096 Dec 17 19:38 gifs
-rw-r--r-- 1 root root 11357 Dec 17 19:38 LICENSE
-rw-r--r-- 1 root root 6300 Dec 17 19:38 main_lcd.c
-rw-r--r-- 1 root root 65086 Dec 17 19:38 Makefile
-rw-r--r-- 1 root root 3186 Dec 17 19:38 make_gifs.py
-rw-r--r-- 1 root root 1778 Dec 17 19:38 README.md
-rw-r--r-- 1 root root 192 Dec 17 19:38 typedefs.h
drwxr-xr-x 3 root root 4096 Dec 17 19:38 verilog
-rw-r--r-- 1 root root 32768 Dec 17 19:38 wario_walking.gb
root@pynq:/home/xilinx/gb/gb-master#

```

然后调用python3 的IDLE执行了一下，发现报错。

```

root@pynq:/home/xilinx/gb/gb-master/verilog#
root@pynq:/home/xilinx/gb/gb-master/verilog# python3
Python 3.6.5 (default, Apr 1 2018, 05:46:30)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> from pynq import Overlay
>>> Overlay("/home/xilinx/gb/gb-master/verilog/mario.bit")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/local/lib/python3.6/dist-packages/pynq/overlay.py", line 302, in __init__
    self.bitfile_name))
ValueError: Cannot find HWH or TCL file for /home/xilinx/gb/gb-master/verilog/mario.bit.
>>> overlay = Overlay("/home/xilinx/gb/gb-master/verilog/mario.bit")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/local/lib/python3.6/dist-packages/pynq/overlay.py", line 302, in __init__
    self.bitfile_name))
ValueError: Cannot find HWH or TCL file for /home/xilinx/gb/gb-master/verilog/mario.bit.
>>> quit
Use quit() or Ctrl-D (i.e. EOF) to exit
>>> overlay = Overlay("/home/xilinx/gb/gb-master/verilog/mario.bit")
/usr/local/lib/python3.6/dist-packages/pynq/overlay.py:299: UserWarning: Users will not get PARAMETERS / REGISTERS information through TCL files. recommended.
  warnings.warn(message, UserWarning)

```

然后根据august提供的建议，将build.tcl 拷贝了一份改名为mario.tcl就好了。。

但是仍然在加载mario.bit时候死机，灯都灭了，可能是bitstream不正确吧，回家慢慢研究。

今天先到这里，给大家发学习心得哈~