Lab 3：Zynq 开发环境使用及 Linux 移植

1 实验目标

1) 熟悉 Vivado 使用环境，熟悉 Verilog 语言；
2) 采用课程提供的矩阵乘法单元，在 FPGA 上进行测试验证，后续实验可能会使用该乘法器阵列；
3) 熟悉开发板，移植 Linux 至 ZYNQ7020 开发板，并测试开发环境，后续实验将会使用该开发环境。

2 实验环境

1) Vivado 2019.2 / Vitis 2019.2
2) ZYNQ 7020 开发板及其配件
3) 本实验过程中使用到的数据文件可在北航盘下载：
   https://bhpan.buaa.edu.cn:443/link/96F27E58B8A459CEAE259FA957FE2F31
   Valid Until: 2021-11-25 23:59
4) **注意：本实验指导书中给出的步骤仅为示意步骤作为参考，每人遇到的情况可能有差异，如果遇到问题可根据实际情况进行探索，或向助教寻求帮助。**

3 实验要求

1) 撰写实验报告并回答 4.1.7 小节中提出的问题。实验报告命名：学号+姓名+实验三。（实验报告撰写细节可参考“实验报告撰写格式”）、
2) 将实验报告交按时交至课程中心作业处。

4 实验内容

4.1 Vivado 集成开发环境使用

4.1.1 文件准备

建立工程文件夹，在文件夹下建立 rtl 目录和 tb 目录。其中，rtl 文件夹存放设计文件，tb 文件夹存放仿真激励文件。建立 proj 目录，作为 vivado 工程目录。注意不要包含中文路径。

图 1 文件夹结构

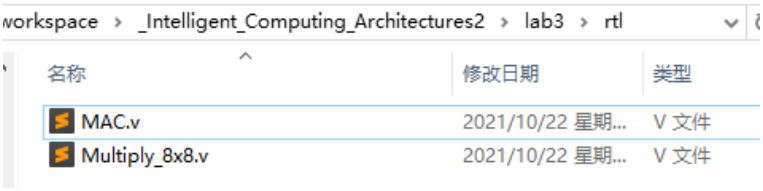


图 2 rtl 文件夹

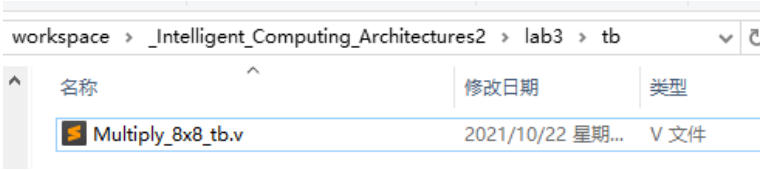


图 3 tb 文件夹

4.1.2 建立工程

需要把 verilog 文件引入工程目录。

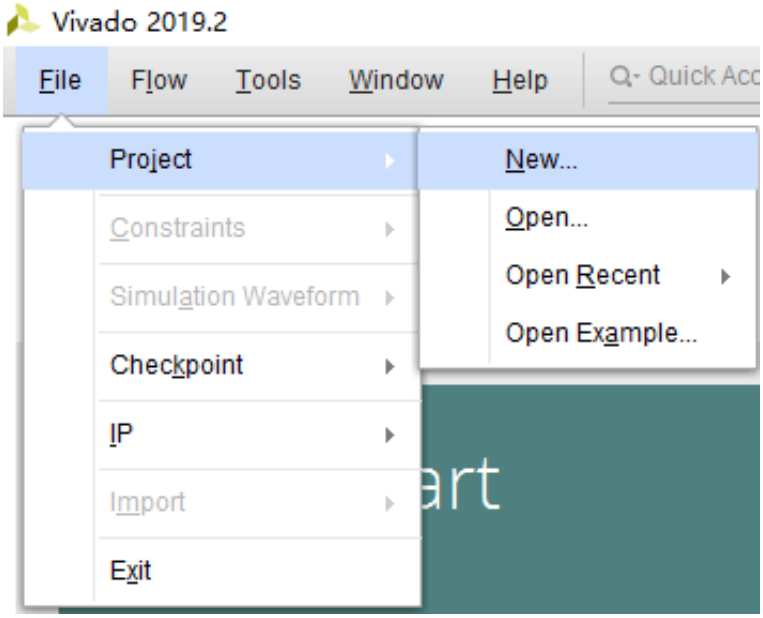


图 4 新建工程

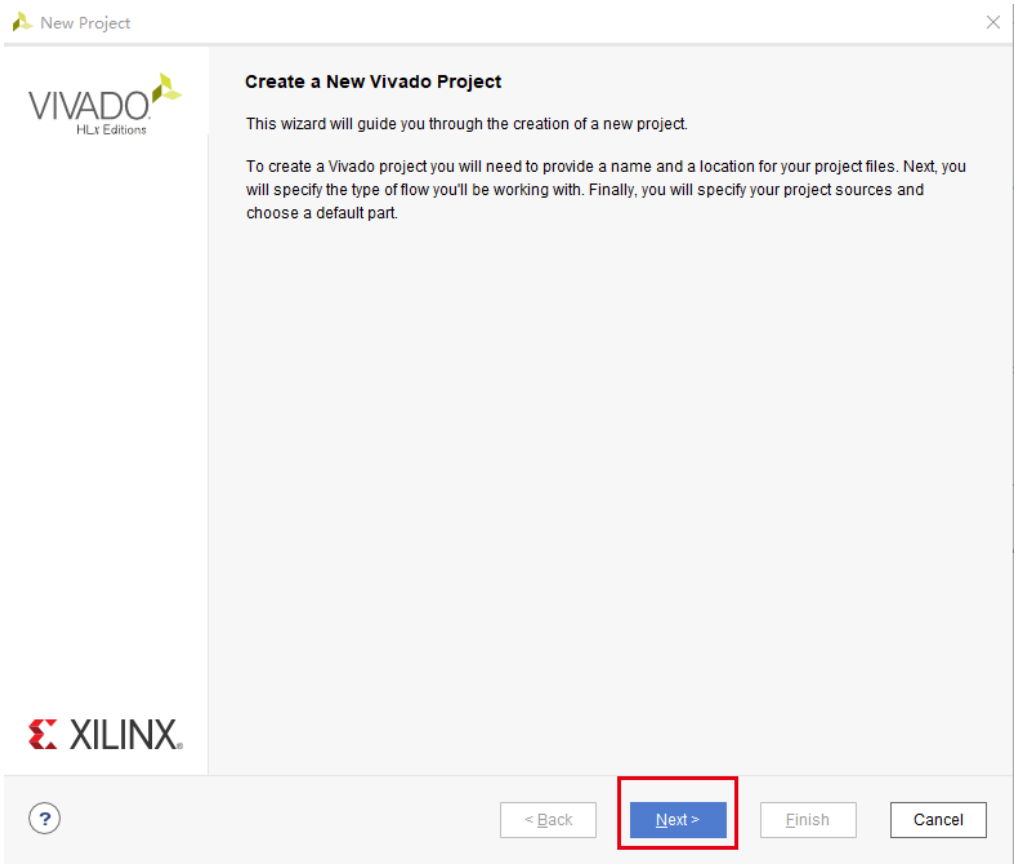


图 5 点击 Next

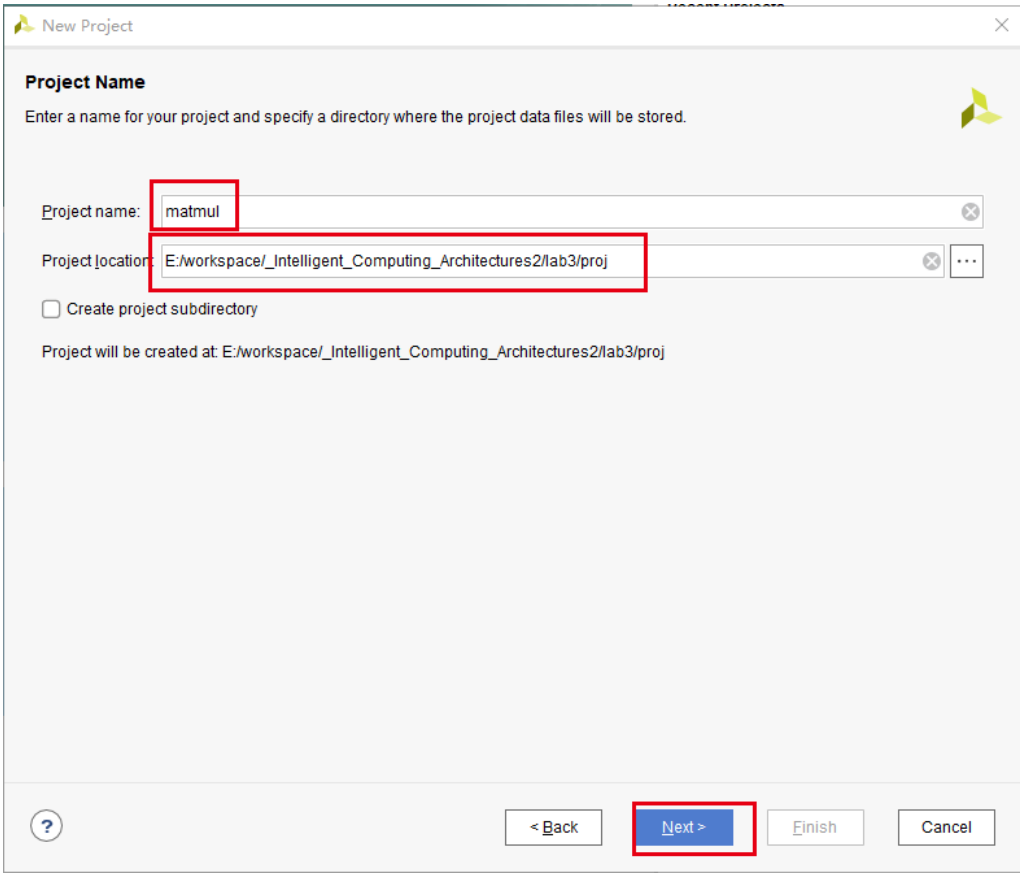


图 6 修改工程名和路径

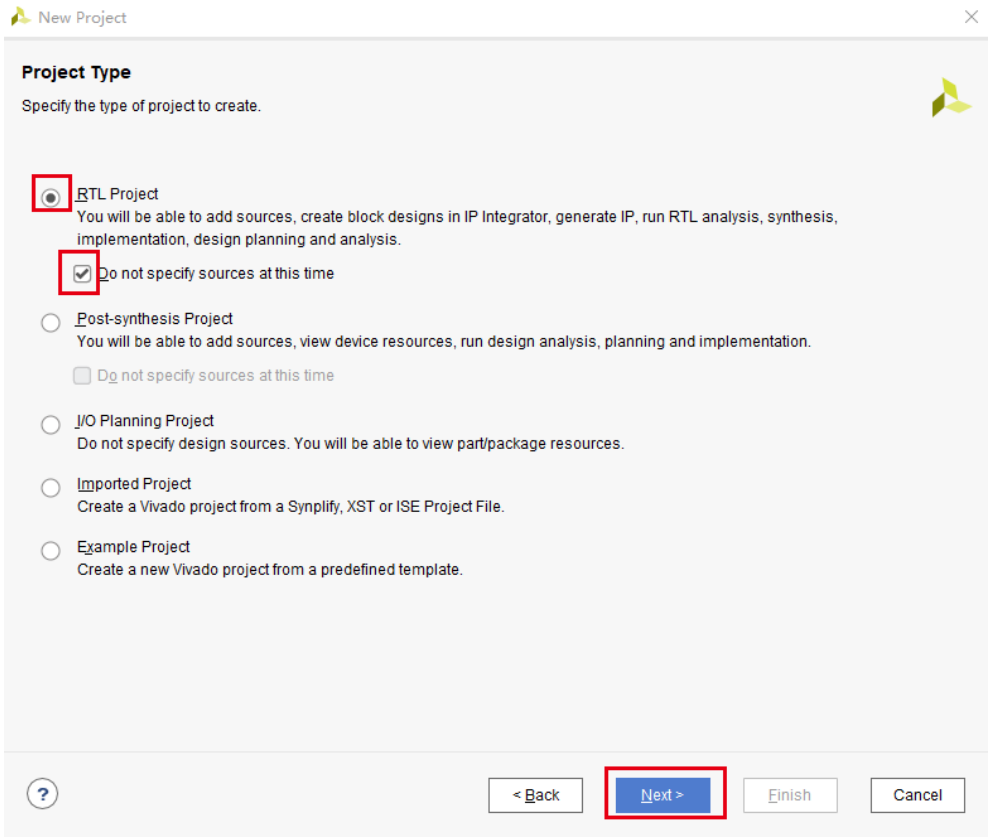


图 7 选择工程类型

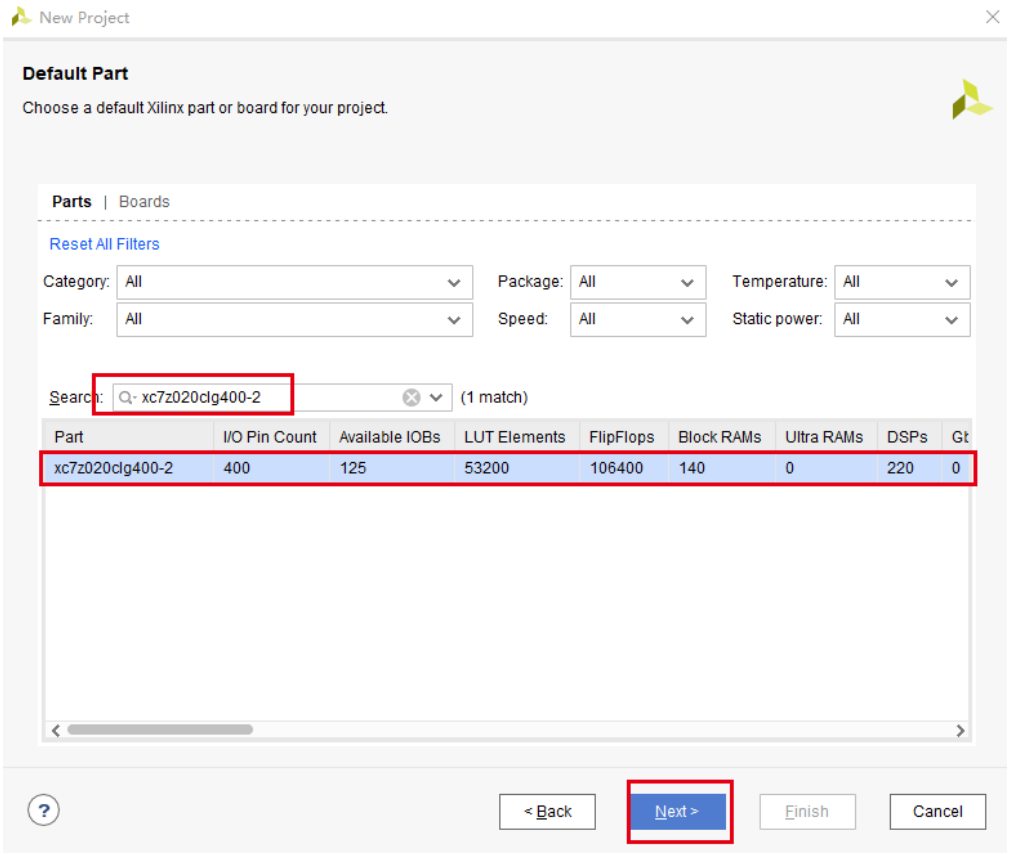


图 8 选择 FPGA 芯片型号

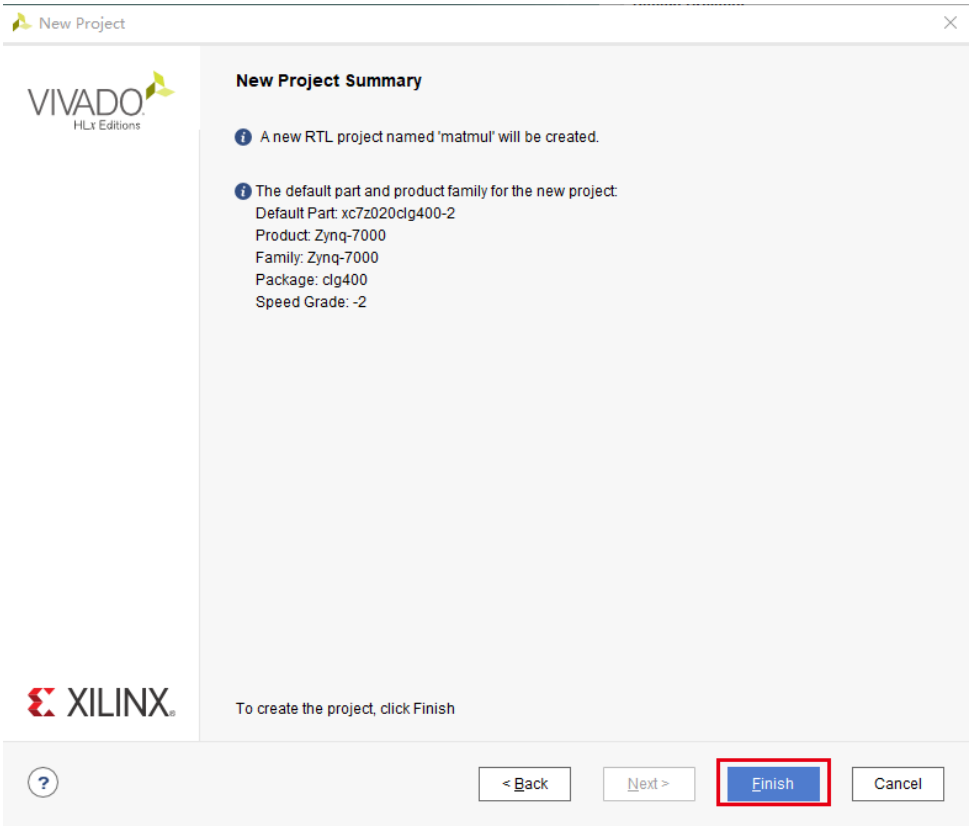


图 9 点击 Finish

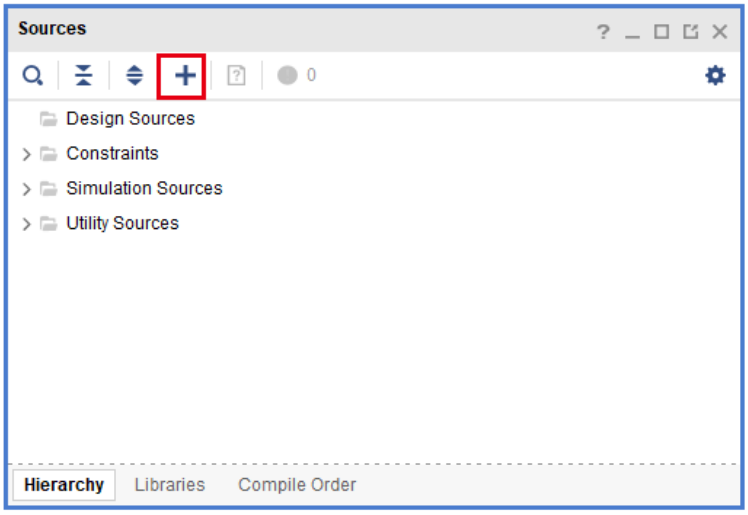


图 10 添加文件

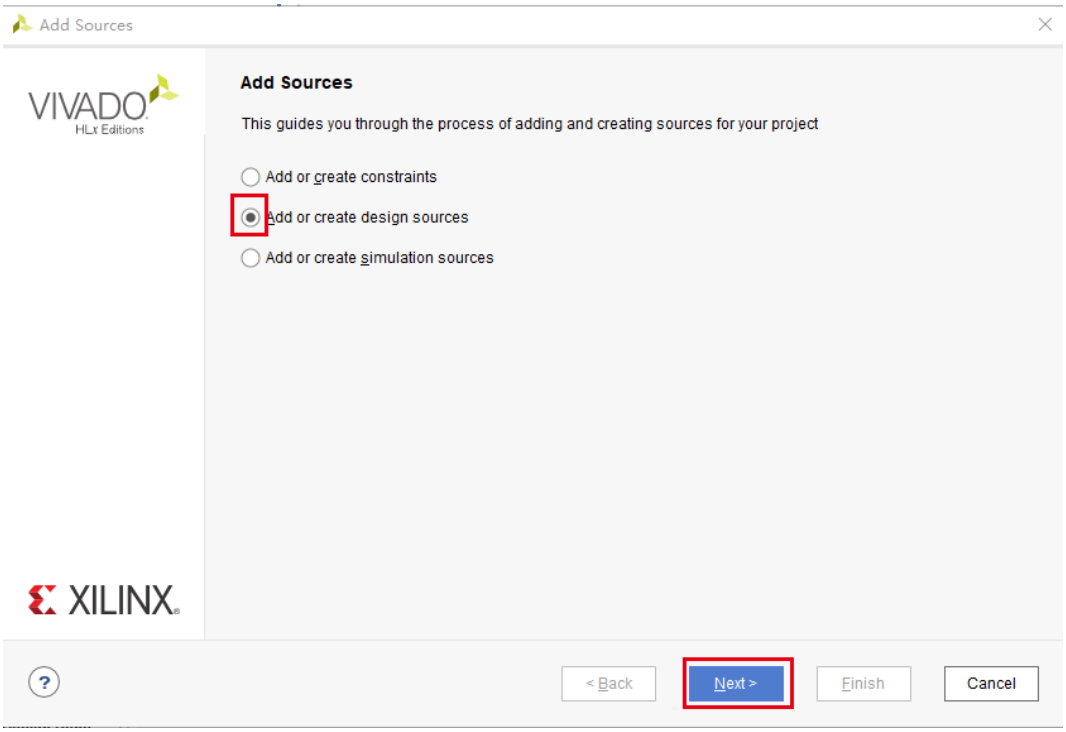


图 11 选择添加设计文件

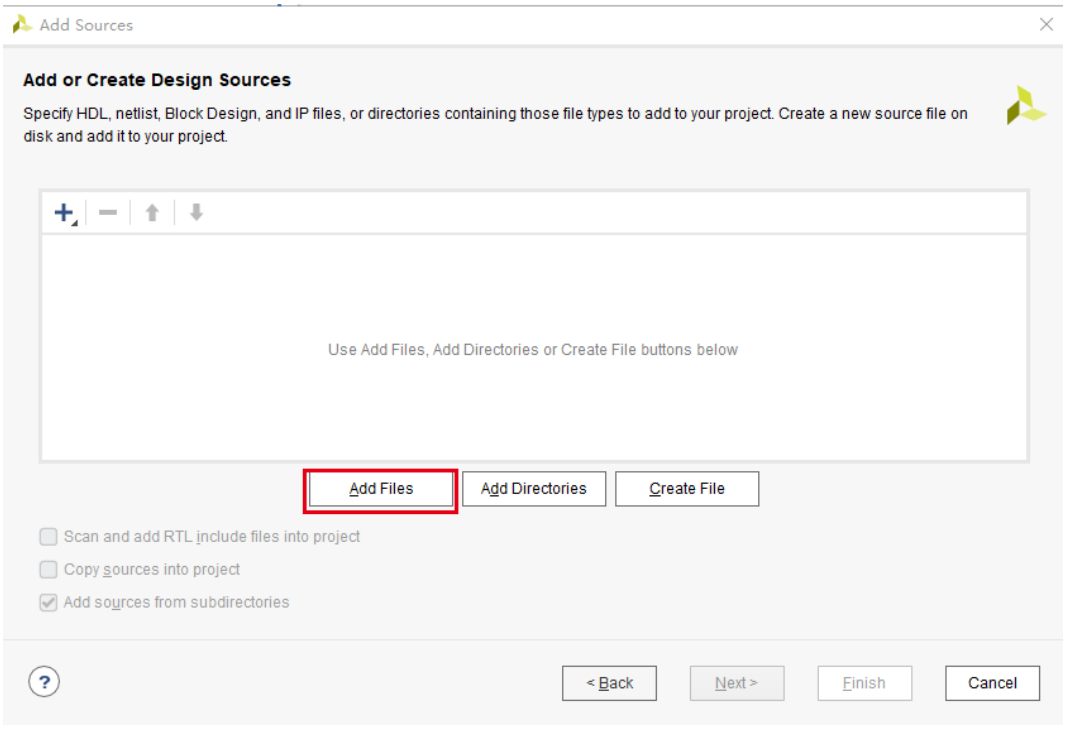


图 12 添加文件

图 13 选择文件

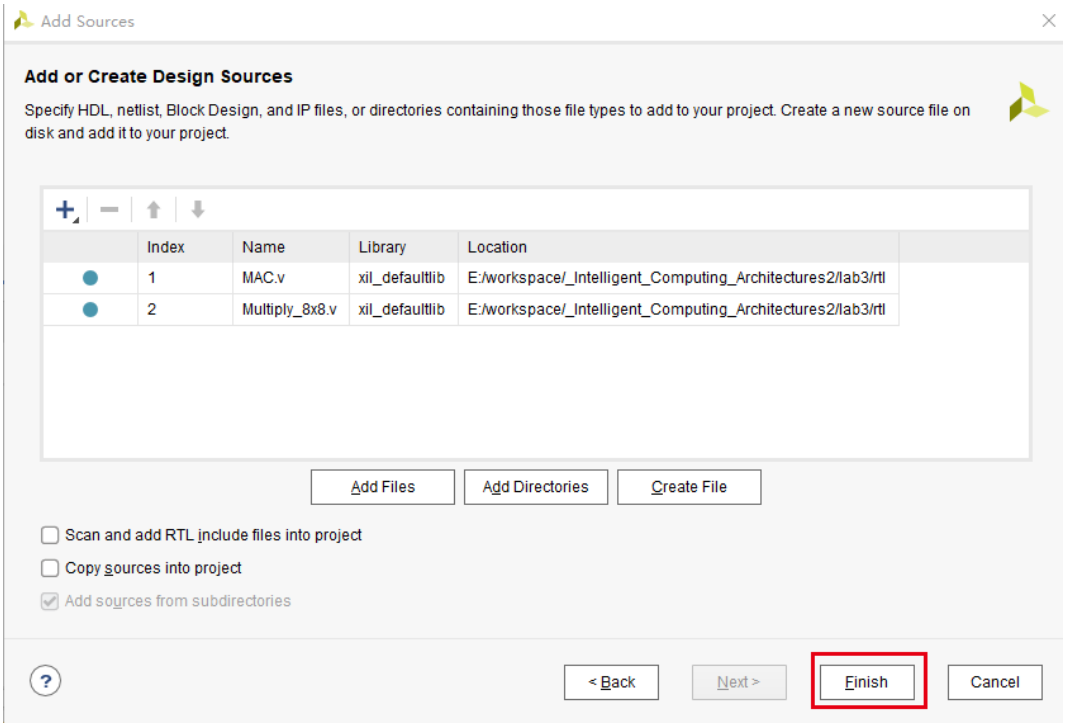


图 14 完成

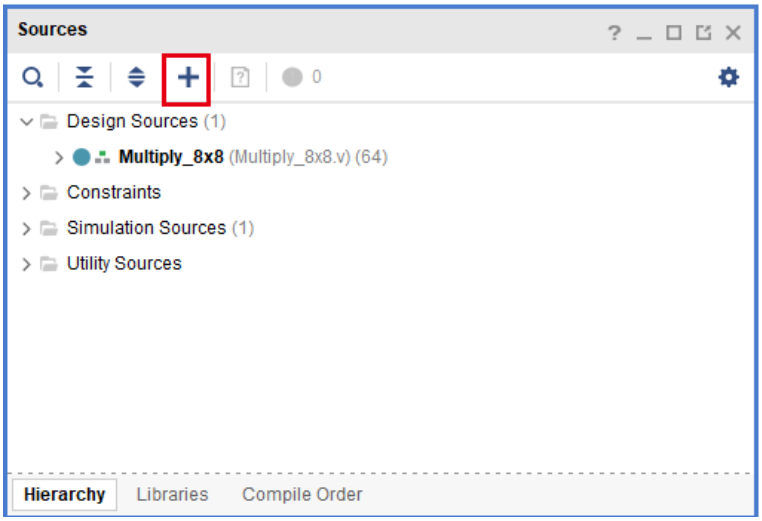


图 15 再次添加文件

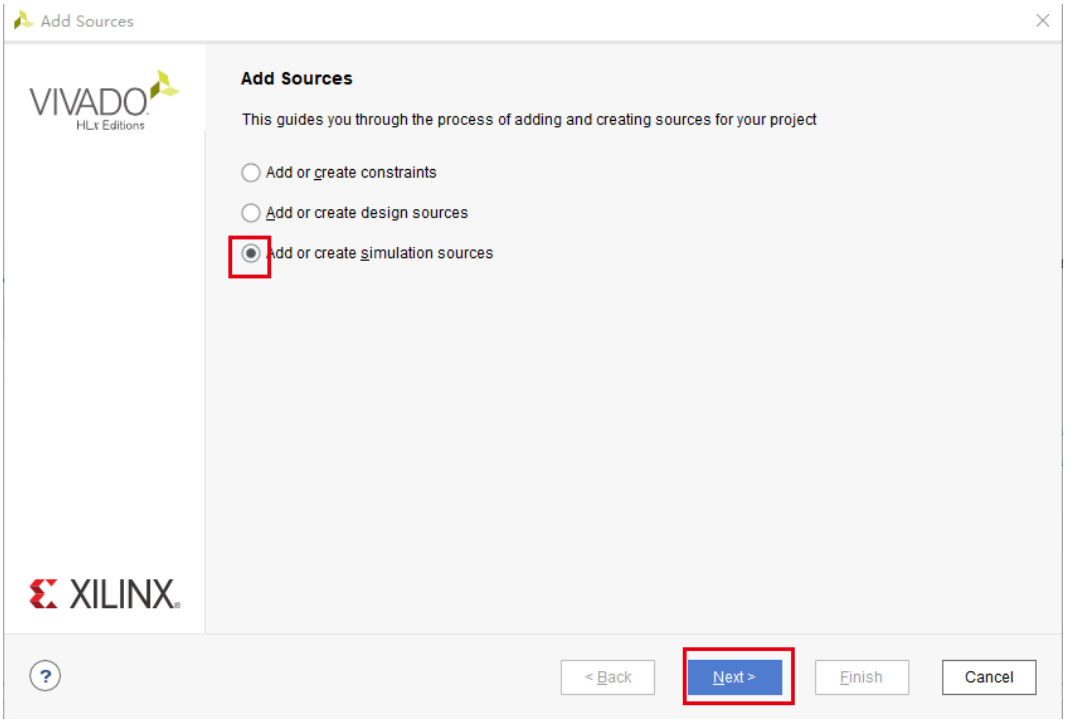


图 16 选择仿真文件

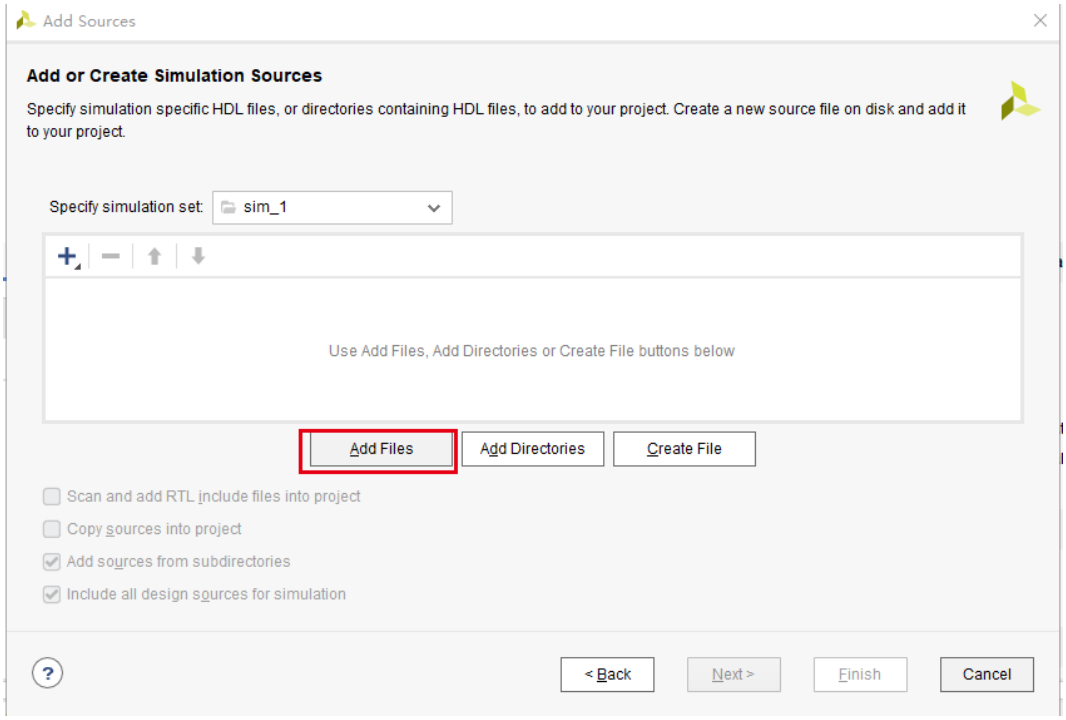


图 17 添加文件

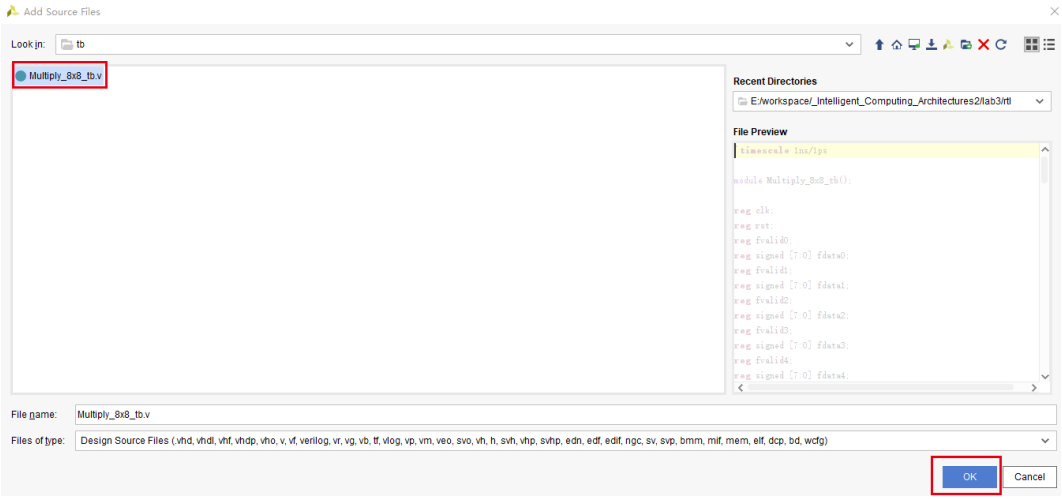


图 18 选择文件

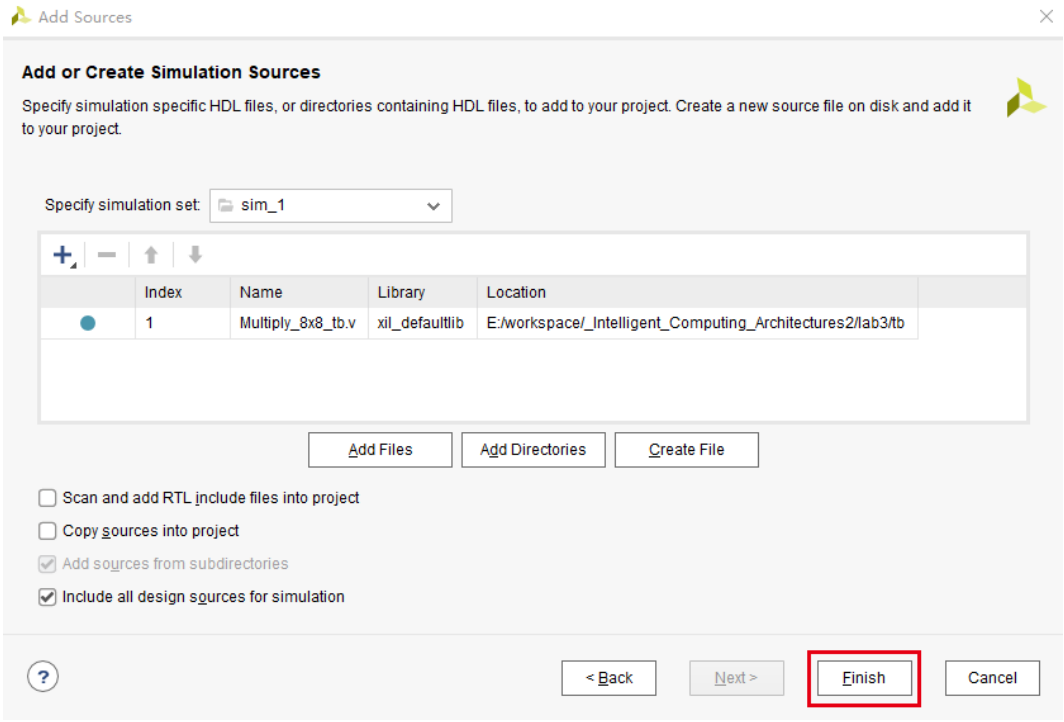


图 19 点击 Finish

4.1.3 仿真

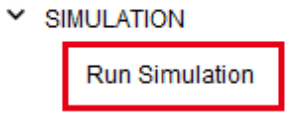


图 20 开始仿真

跑一段时间的仿真，然后暂停。

图 21 工具栏

打开波形，用有符号数显示。

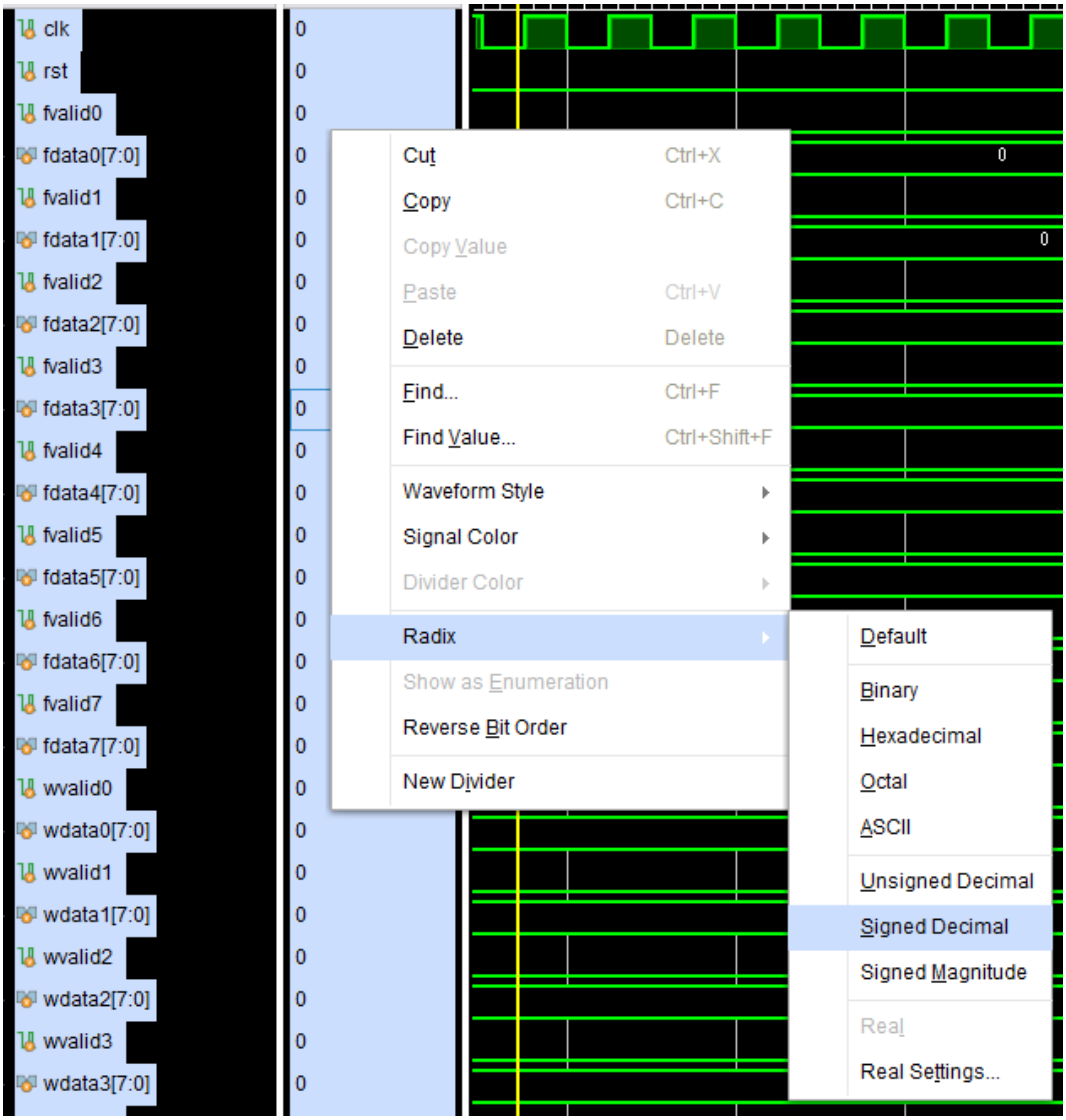


图 22 选择有符号十进制

查看输入矩阵波形。

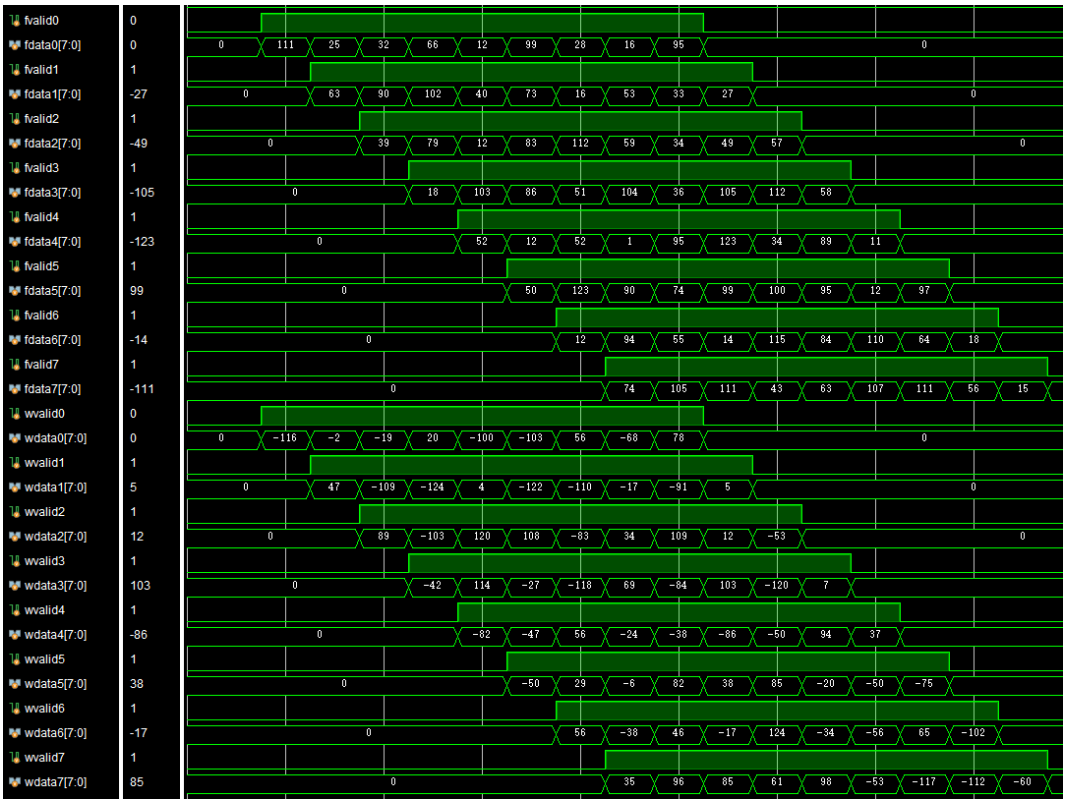


图 23 输入矩阵波形

查看输出矩阵。

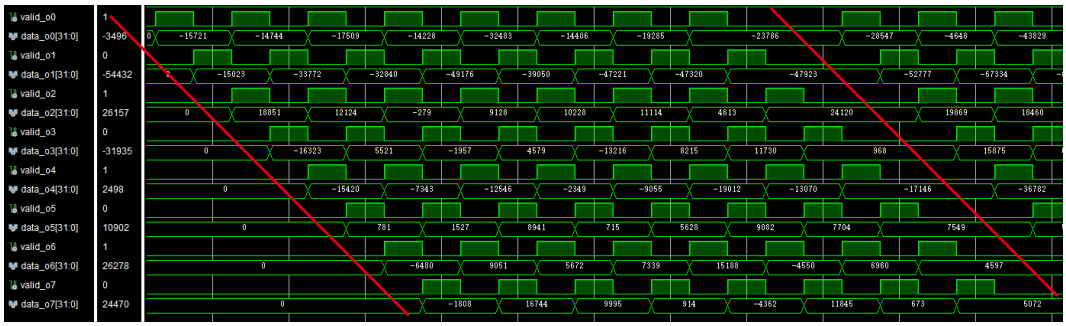


图 24 输出矩阵波形

4.1.4 对比结果

执行 test.py，得到矩阵运算结果。

```
[[-15721 -15023  18851 -16323 -15420    781  -6480  -1808]
 [-14744 -33772  12124   5521  -7343   1527   9051  16744]
 [-17509 -32840   -279  -1957 -12546   8941   5672   9995]
 [-14228 -49176   9128   4579  -2349    715   7339    914]
 [-32483 -39050  10228 -13216  -9055   5628  15188  -4362]
 [-14406 -47221  11114   8215 -19012   9082  -4550  11845]
 [-19285 -47320   4613  11730 -13070   7704   6960    673]
 [-23786 -47923  24120    968 -17146   7549   4597   5072]]
```

图 25 矩阵运算结果

该结果和 4.1.3 仿真中的仿真输出一致。

4.1.5 疑问

Multiply_8x8_tb.v 中共进行了两次矩阵运算，仔细对比可以发现，第二个矩阵的运算结

果并不对。

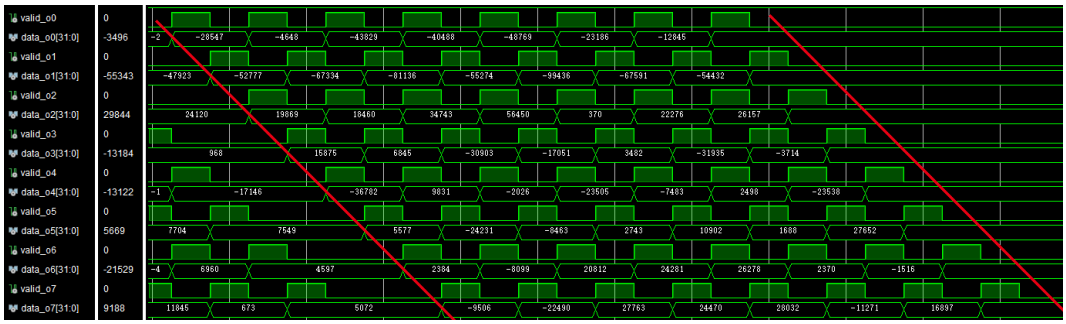


图 26 第二个矩阵

```
[[-10883   9687  18077  -8445  -1710  -1591  19792  24286]
 [-16168  18682  -3300 -12867 -13209   7001  13661   5158]
 [  9163  18704  11703   -951  -1002   -271 -13492 -24205]
 [   216  -1002 -31358 -13211  11823  -8521 -14887 -17002]
 [ 25983  42900   5490  15770  -2107 -13674 -15450  -1152]
 [-30866 -46599   5124  27201 -24894  12696  16194  17145]
 [-31789 -23200  -3027 -29058   7438   4356  26900   6657]
 [-23208   2001  11412  -7040 -18754  -2011   6887 -37404]]
```

图 27 第二个矩阵运算结果

在 Scope 中可以添加波形。

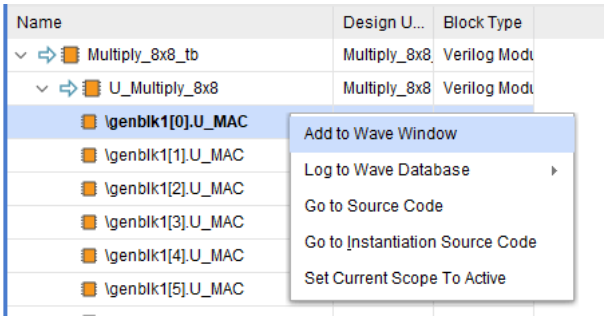


图 28 添加波形

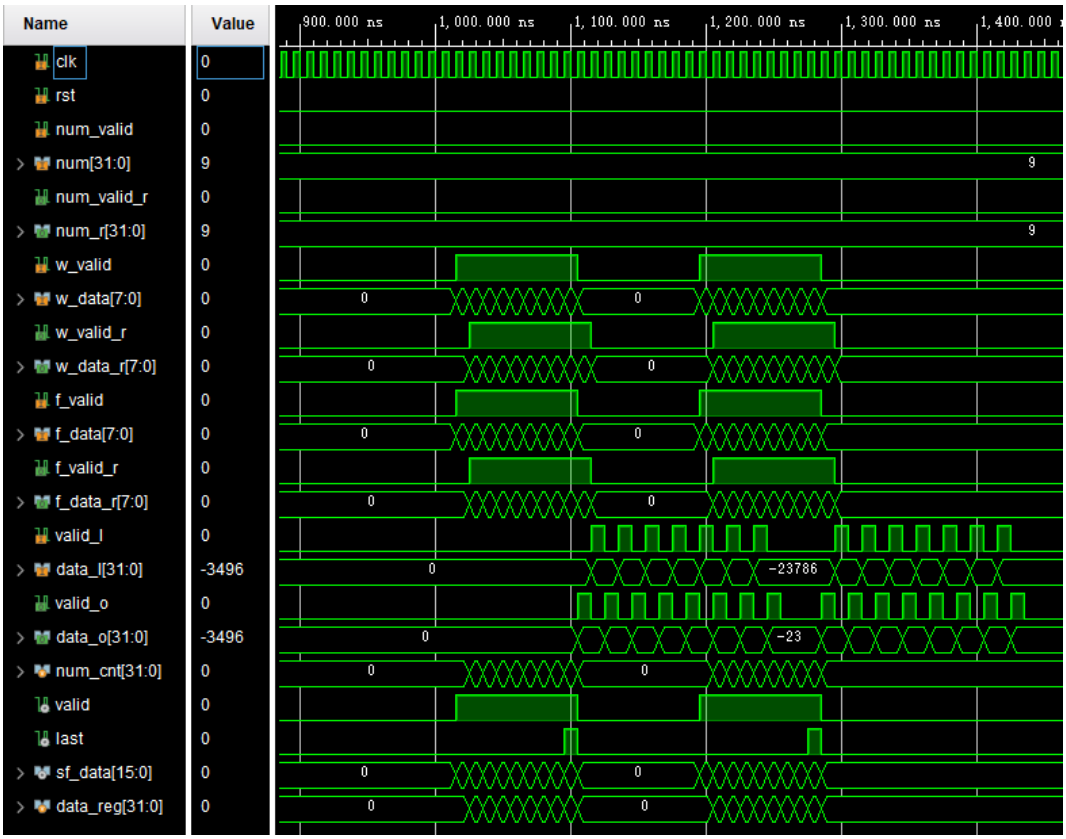


图 29 某 MAC 模块波形

请大家阅读 MAC.v 代码，找出第二个矩阵乘计算出错的原因。

提示：思考下图代码

```
102  wire signed [15:0] sf_data;
103  assign sf_data = $signed({8'b0,f_data});
104  // 本级计算
105  reg signed [31:0] data_reg;
106  always @(posedge clk or posedge rst) begin
107      if (rst) begin
108          data_reg <= 32'b0;
109      end
110      else if (valid) begin
111          if (last) begin
112              data_reg <= 32'b0;
113          end
114          else begin
115              data_reg <= data_reg + $signed(w_data)*$signed(sf_data);
116          end
117      end
118      else begin
119          data_reg <= data_reg;
120      end
121  end
```

图 30 MAC 模块代码片段

结合输入矩阵数据格式（feature 是 uint8，weight 是 int8），考虑本模块是否有修改的必要。

4.1.6 矩阵乘模块设计框图

1) MAC.v

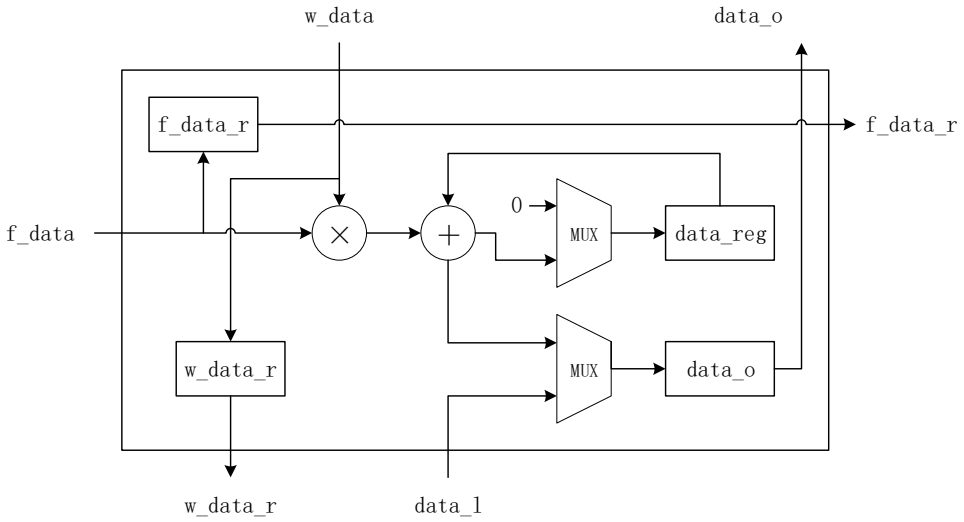


图 31 MAC 模块框图

2) Multiply_8x8.v

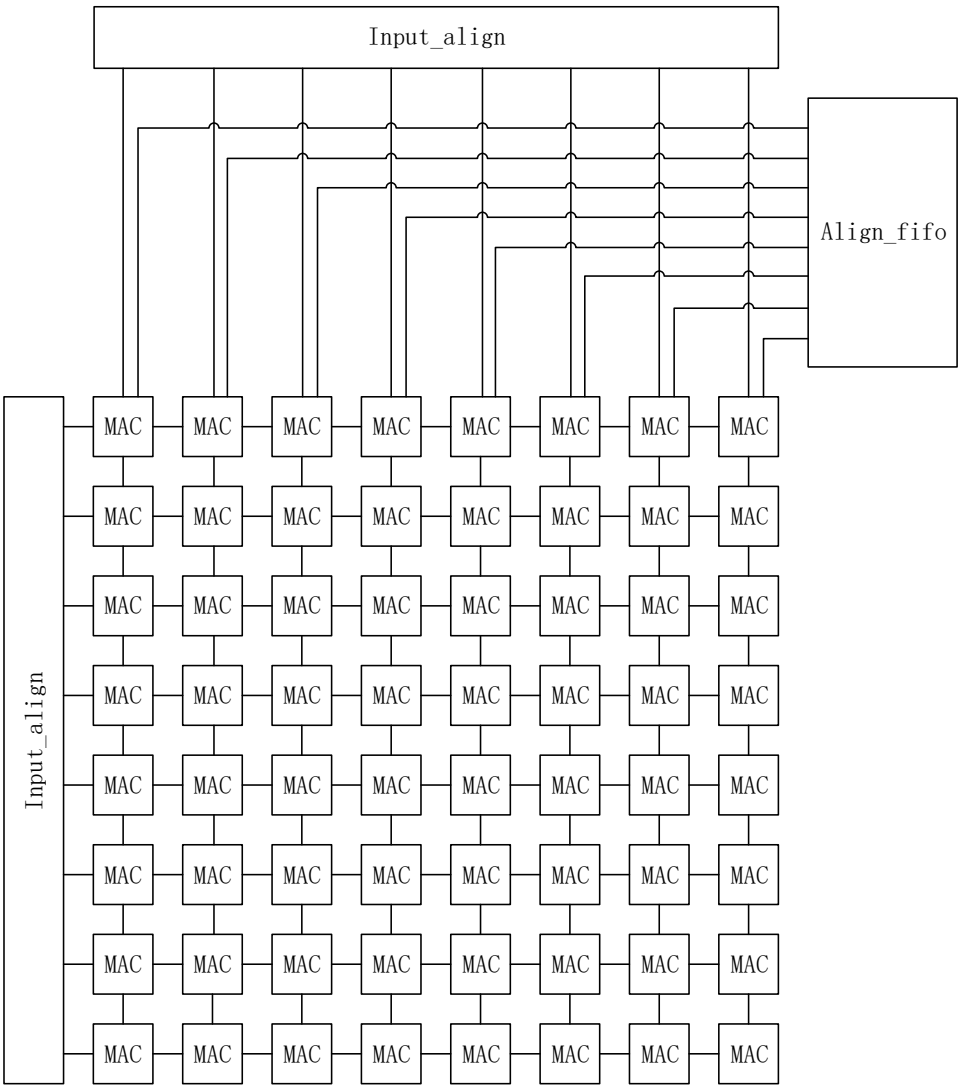


图 32 Multiply_8x8 模块框图

其中，Input_align 和 Align_fifo 暂未提供。Input_align 用于将原始输入的对齐数据错位，从波形中也能看出；Align_fifo 用于将错位的矩阵乘结果对齐，收集起来再输出出去。

4.1.7 需要提交完成内容

1) 回答 4.1.5 疑问中的疑问，问题 1——第二个矩阵计算为何出错，问题二——MAC.v 是否有修改的必要；
2) 参考 4.1.6 矩阵乘模块设计框图中的两幅图，为 MAC.v 和 Multiply_8x8.v 两个模块编写设计文档（写在实验报告中即可）。设计文档需要包括：一，模块设计功能；二，模块接口说明；三，模块设计思想及流程描述；四，内部关键信号与变量描述。

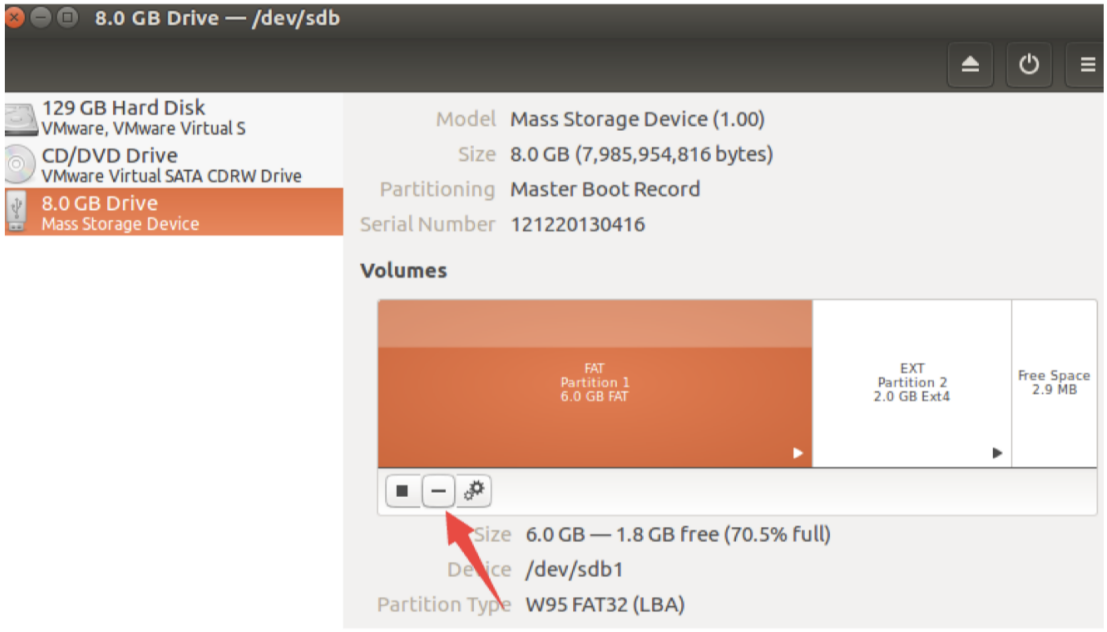
4.2 Linux 移植

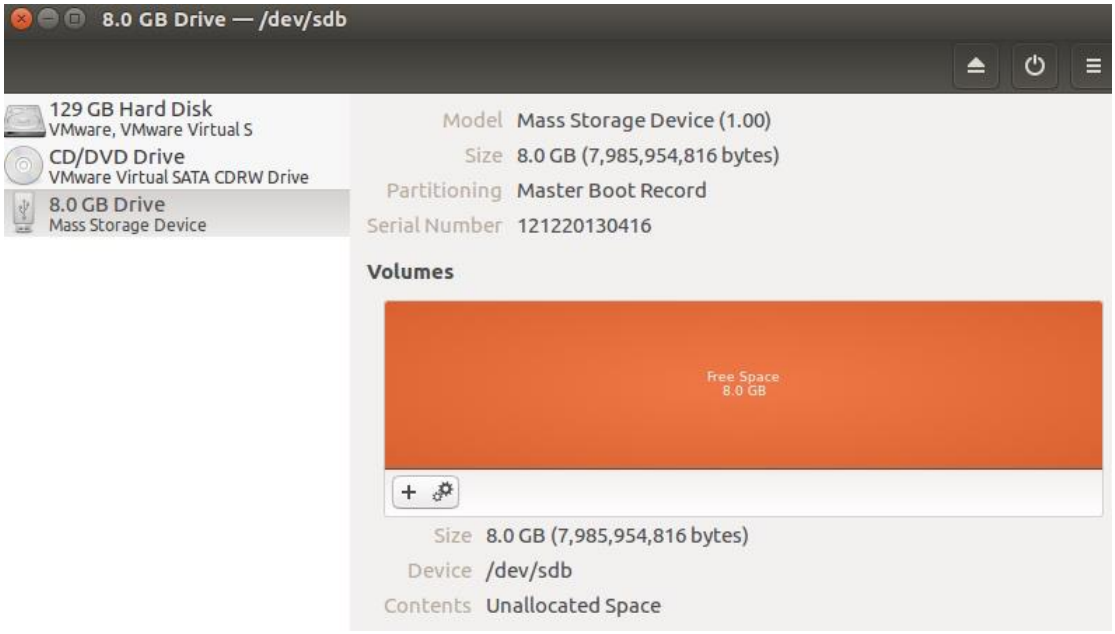
4.2.1 制作 SD 卡文件系统

制作 SD 卡文件系统会导致 SD 卡里内容丢失，请先做好备份。

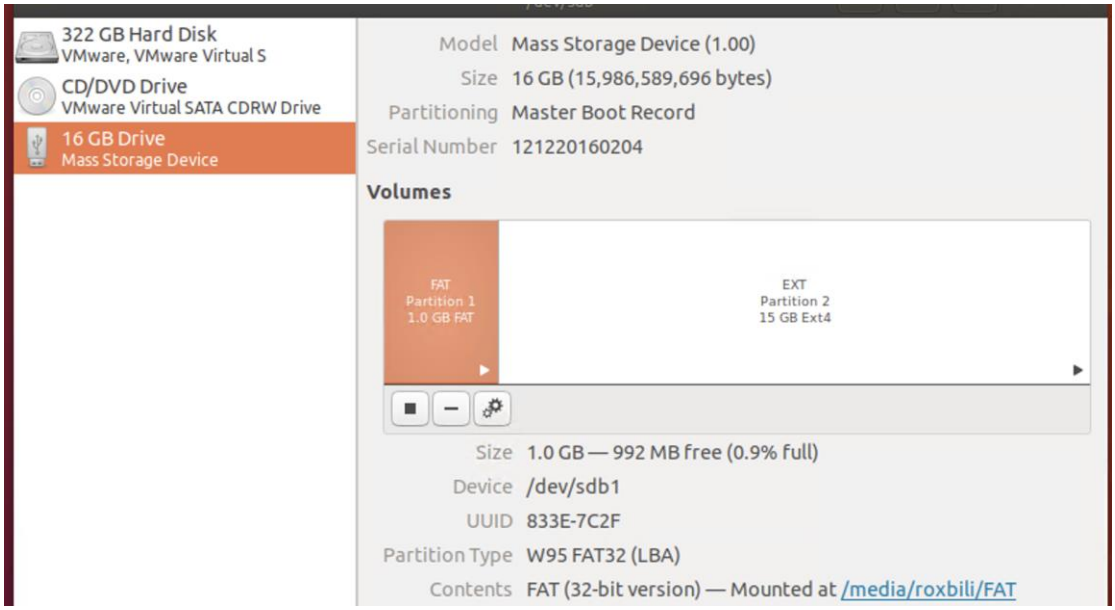
1) 把 SD 卡插入读卡器，然后插入电脑 USB 口。
2) 在 ubuntu 的搜索路径中，输入 disk，会出现 Disks 的图标，打开该软件。（别的分区软件同理）

3) 点击-号，删除 SD 卡中的所有分区。

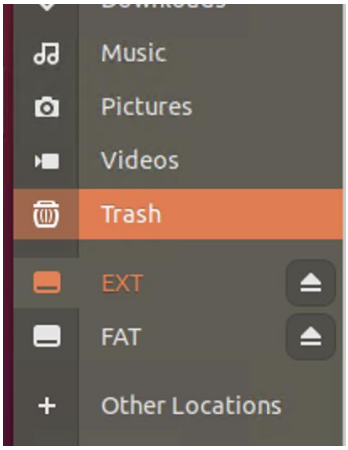




4) 点击添加分区的图标，添加第一个分区，格式为 FAT，用于存放 ZYNQ 的启动文件 BOOT.bin 和内核文件、设备树，名称为 FAT。（参考分配大小：1G）
5) 创建第二个分区，用于存放根文件系统，格式为 EXT4，名称为 EXT
6) 最终分区效果如图所示：



7) 成功后系统会自动挂载分区，并弹出窗口



8) 同步根文件系统到 SD 卡的 EXT 分区

将根文件系统移动到 Linux 主机，输入以下命令解压压缩包

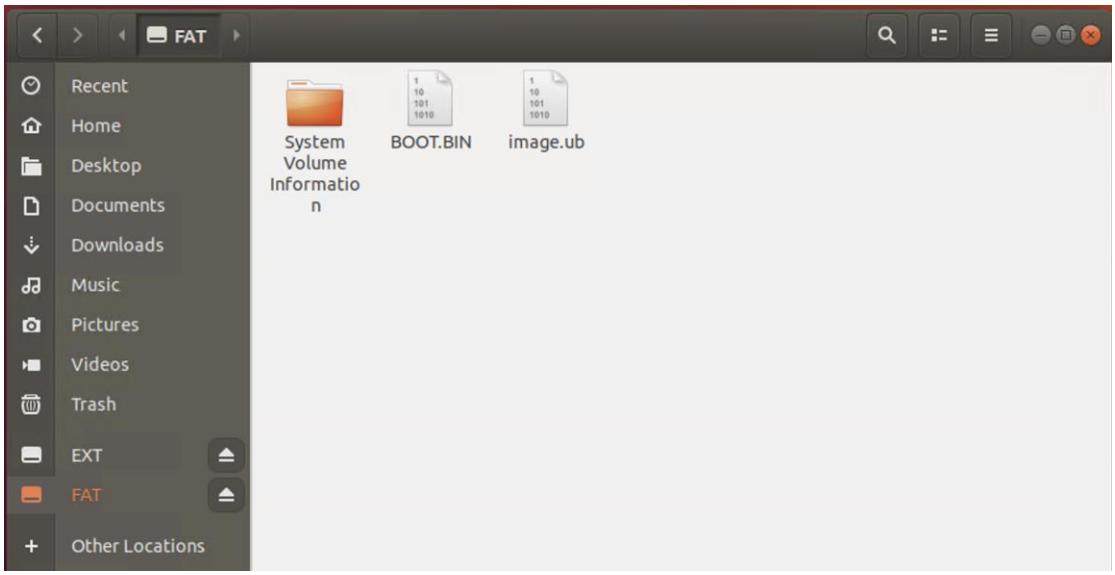
```
tar zxpvf linaro-jessie-alip-20161117-32.tar.gz
```

进入解压后的目录并将根文件系统同步到 sd 卡的 EXT 分区中

```
cd binary
sudo rsync -av ./ /media/<your user name>/EXT
```

9) 解压 boot.zip 目录，将 BOOT.BIN 和 image.ub 文件移动至 sd 卡的 FAT 分区下。

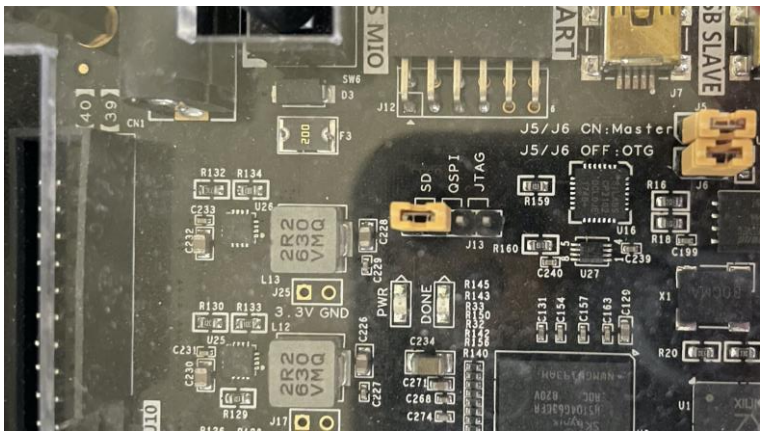
10) 最终 SD 卡文件系统如图所示。

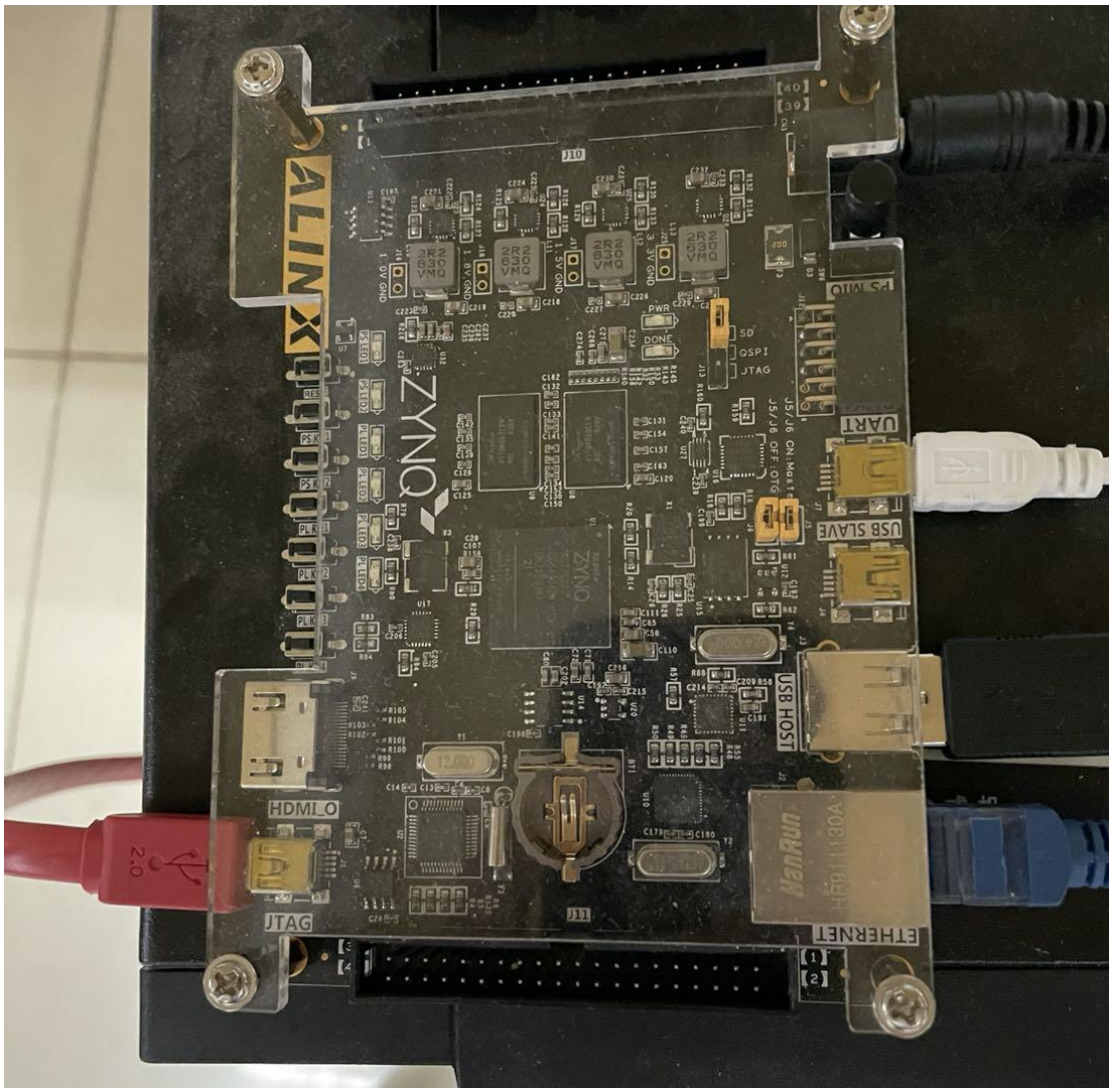


11) 推出 U 盘，如果无法正常推出，等待一阵子即可。

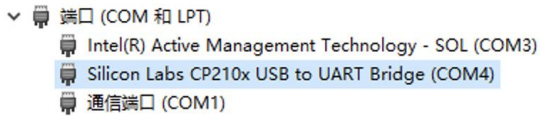
4.2.2 启动 ZYNQ 上的文件系统

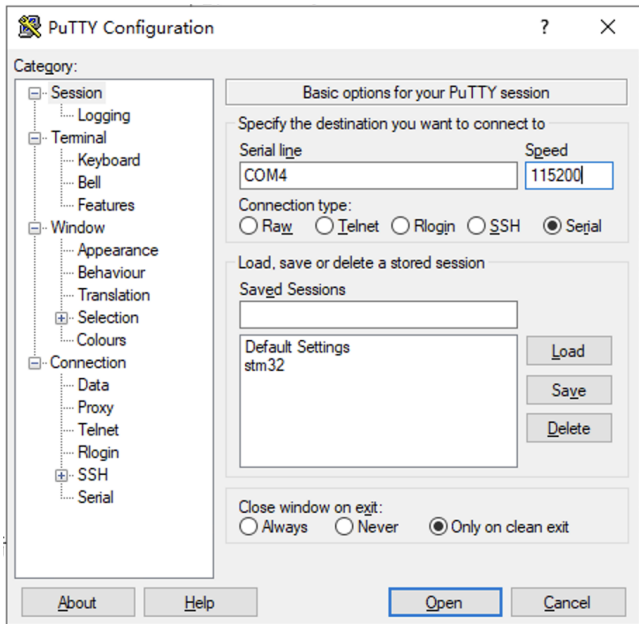
1) 将 SD 卡插入 ZYNQ 开发板中
2) 开发板上的跳线选择连接 SD 模式，并将开发板的 JTAG 线和 UART 线连接到主机，按下电源按钮



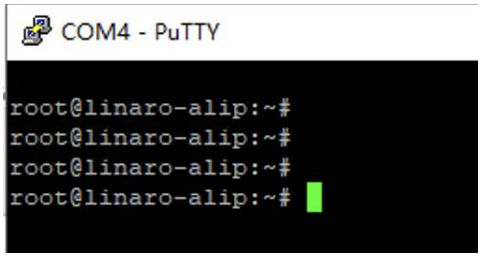


3) 在设备管理器中找到 UART 线对应的端口号（图片中为 COM4），使用 putty 软件打开该端口，波特率 115200





点击 open 打开串口，输入回车看到以下信息则说明系统正常启动



4.2.3 安装开发环境

后续实验需要 numpy 库支持，因此此处在系统自带的 python3 中安装 numpy 环境。（以下步骤需要给 zynq 开发板插上网线）

1) 更新 pip

```
python3 -m pip install --upgrade pip
```

2) 安装 numpy

```
python3 -m pip install numpy
```

此时会开始长时间的编译过程，请耐心等待。

3) 测试 numpy

```
python3
import numpy
```

```
root@linaro-alip:~# python3
Python 3.6.1 | packaged by rpi | (default, Apr 20 2017, 19:35:19)
[GCC 4.9.2] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>>
```

import 未报错则说明成功。