智能计算体系结构课程lab5实验报告

作者：北京航空航天大学计算机学院 陈胤佳

学号：19373383

一、实验目的与要求：

本次实验目的为，在Lab3移植Linux至ZYNQ7020开发板、并测试开发环境、Lab4测试挂载BRAM读写操作、理解软硬件交互逻辑并实现矩阵乘法Matmul接口的基础上，进一步理解矩阵乘法的拆分、掌握控制流程的设计；理解模块设计思想，掌握自顶向下的设计方法；学习FPGA设计与实现流程。

二、实验环境：

IDE：Vivado 2019.2 / Vitis 2019.2；

Environment：ZYNQ 7020开发板及其配件；

Environment：ZYNQ 7020上的Linux系统；

OS：Windows 10 Professional

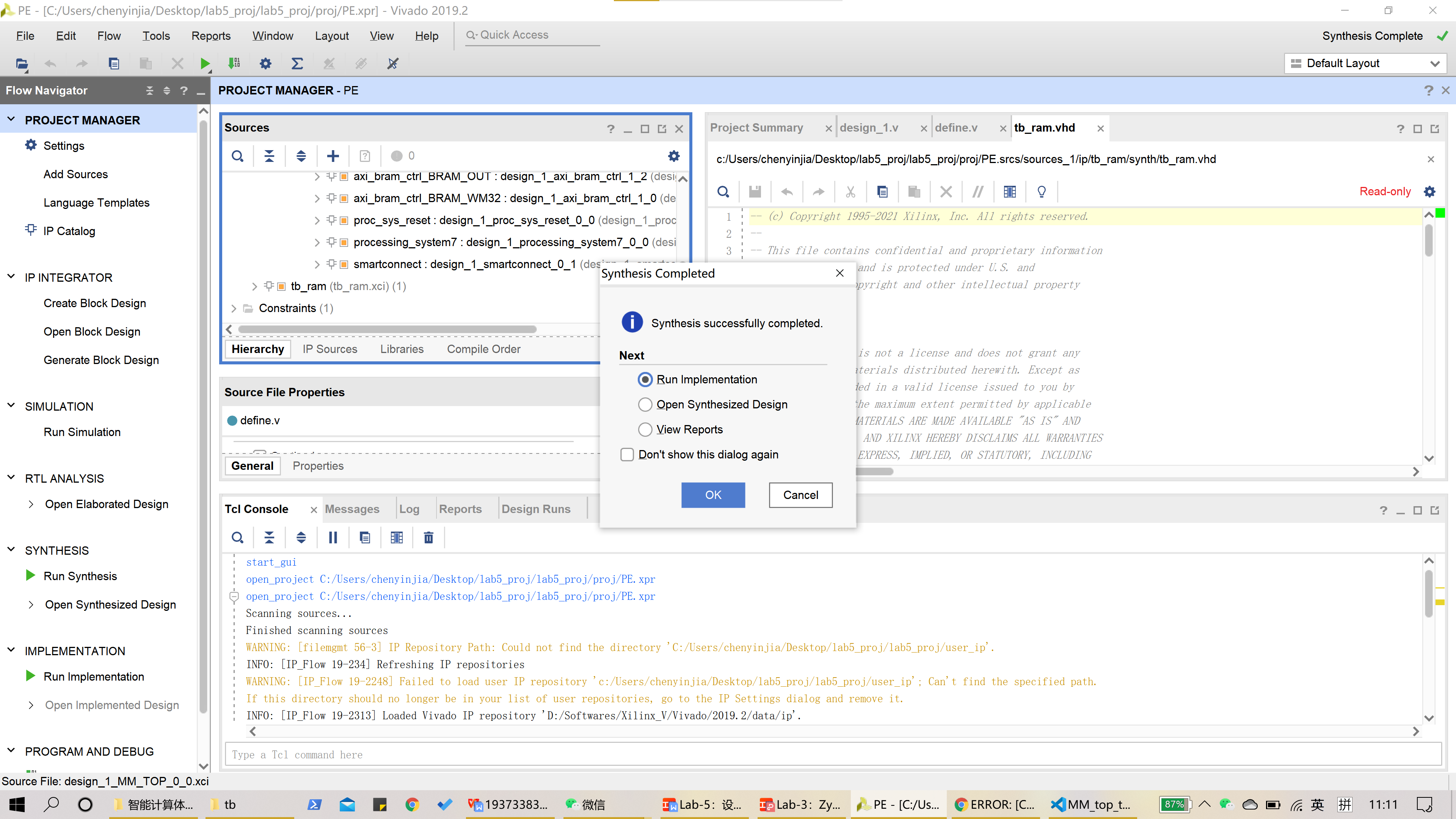
三、实验内容与步骤

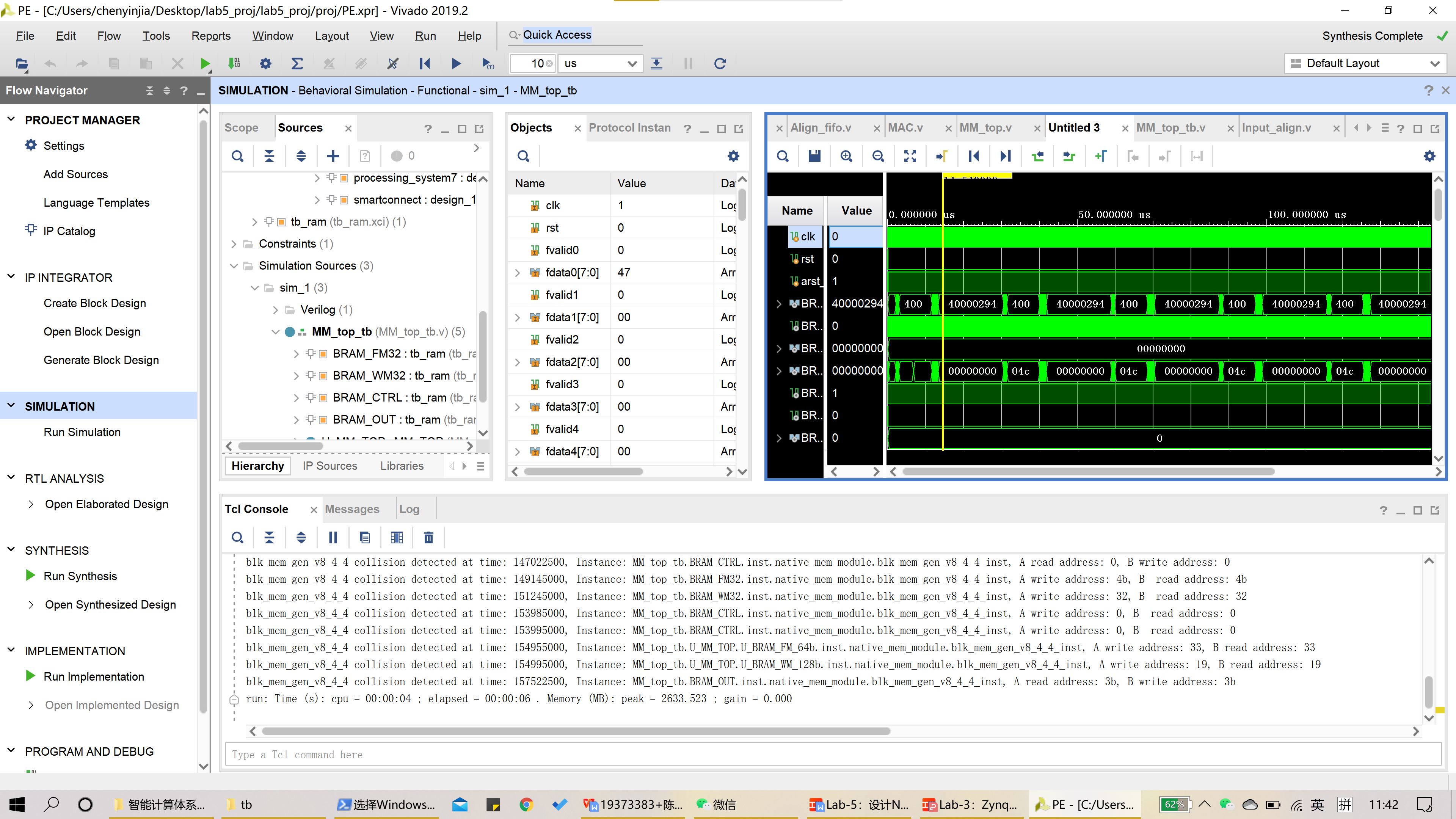
1. 矩阵乘法模块的设计：

主要涉及原理性的理解，具体设计方案见指导书，是Lab3、Lab4的详尽版本。

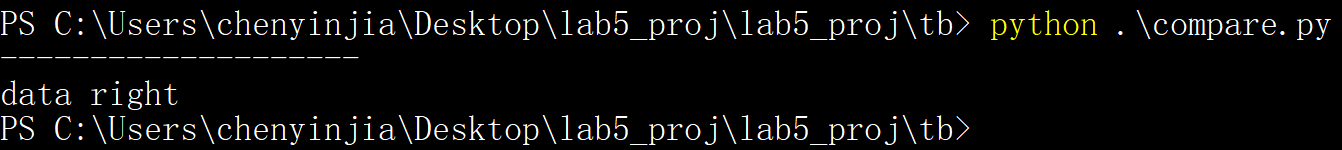
1. 矩阵乘法模块实现：
2. 仿真测试

首先运行tb目录下的gen\_matrix.py，在test1、test2目录下分别生成随机的测试矩阵，而后利用课程组所提供的工程项目文件，修改为本地路径后进行仿真。



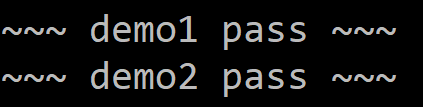


仿真成功后，tb目录下的tb\_MMout.txt会被写入发生变化，运行compare.py对仿真结果和正确结果进行比对，得到结果：



1. 上板测试

Matmul.py运行结果如下：



1. 回答问题

### 4.3.1 问题1（3※）

依据工程中BRAM\_FM64和BRAM\_WM128的位宽、地址深度，写出M、N、P取值的限制。

二者位宽、地址深度、总容量限制如下：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 位宽 | 地址深度 | 总容量限制（B） |
| BRAM\_FM64 | 64 | 4096 | 32768 |
| BRAM\_WM128 | 128 | 8192 | 131072 |

因此M、N、P取值限制为：

MN (B) <= 32768 B

NP (B) <= 131072 B

### 4.3.2 问题2（3※）

依据mlp和lenet模型，判断当前BRAM\_FM32、BRAM\_WM32、BRAM\_FM64和BRAM\_WM128大小是否足够。

当前内存最大容量：

BRAM\_FM32：32000 B

BRAM\_WM32：128000 B

BRAM\_FM64：32768 B

BRAM\_WM128：131072 B

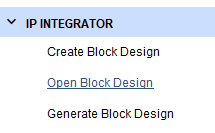
mpl和lenet模型特征矩阵大小1\*784 B < 25 \* 784 B < 32000 B < 32768 B

mpl和lenet模型权重矩阵大小400 \* 120 B < 100 \* 784 B < 128000 B < 131072 B，

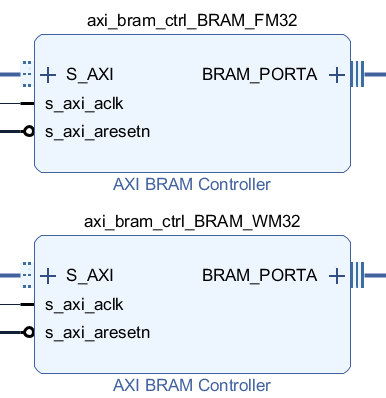
因此，大小足够。

### 4.3.4 问题4（8※）

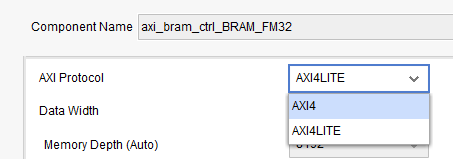
如图 2 矩阵乘法模块框图，输入矩阵首先写入BRAM\_FM\_32b和BRAM\_WM\_32b，再转换写入BRAM\_FM\_64b和BRAM\_WM\_128b。修改代码和工程，将PS侧输入矩阵直接写进BRAM\_FM\_64b和BRAM\_WM\_128b。提示，PS侧和PL侧的交互不一定非要采用AXI4LITE协议，这意味着总线的数据位宽不一定非得是32bit。



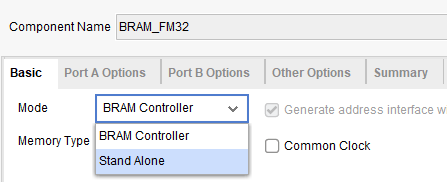
**图 41 打开Block Design**



**图 42 双击AXI BRAM Controller进行设置**



**图 43选择AXI协议**



**图 44设置BRAM模式**

**（可以在Stand Alone模式下修改位宽后切换回BRAM Controller模式）**

**注意，输入矩阵直接写进BRAM\_FM\_64b和BRAM\_WM\_128b时，PS侧补0的个数也需要进行修改。本题目另提供工程，无须对原有工程进行修改，修改PS侧补0代码，跑通流程即可。**

根据课程组提供的工程文件，可以找到 tb 文件夹下的 gen\_matrix.py 文件已经发生了改变，可以看到生成的 FM.txt 矩阵文件和 WM.txt 矩阵文件的行单位改变8和16，说明此时已经是存储到 BRAM\_FM\_64b 和 BRAM\_WM\_128b 内存中的格式。在 Block Design的设计图中可以看到原来的 BRAM\_FM32b 和 BRAM\_WM\_128b 模块均已变成BRAM\_FM\_64b 和 BRAM\_WM\_128b 模块，同时控制器也发生了相应的改变，但是内存大小没有发生变化，因为大小足够。

修改路径后，正常仿真代码并上板实测能够得到正确的结果。

### 4.3.5 问题5（3※）

PS侧输入矩阵会进行补0操作。从PL操作流程考虑，补0是否必要？如果不进行补0，无效位置保留默认值，是否会影响矩阵运算结果？给出原因。

补0不是必要的操作。

由于在最后的 Align\_fifo 收集模块中，会接受到 Multiply\_ctrl 传来的矩阵实际大小规模即sub\_scale\_M 和 sub\_scale\_P ，该模块会根据实际大小将边缘数据裁剪掉，得到去掉无效位置的计算结果，因此不必补零。

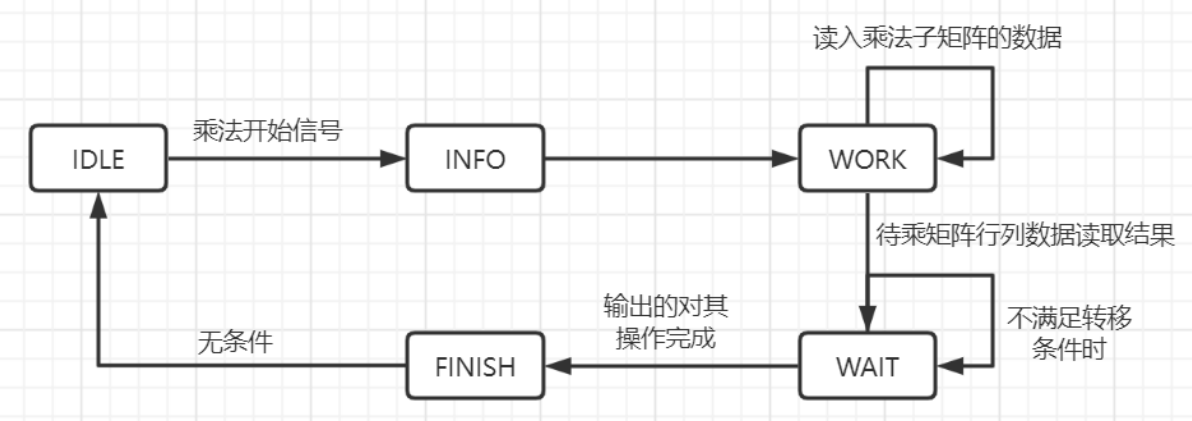
### 4.3.7 问题7（2※）

本工程中FPGA的时钟频率是多少？提示，在Block Design中双击ZYNQ7 Processing System，点击Clock Configuration查看PL Fabric Clocks的时钟。

查看PL Fabric Clocks 可知本工程中FPGA的实际时钟频率为125MHz。

### 4.3.8 问题8（5※）

仿照图 3 CTRL模块状态转移图，画出Multiply\_ctrl模块的状态转移图。并给出以下寄存器的功能：sub\_scale\_M1、sub\_scale\_P1、sub\_scale\_M2、sub\_scale\_P2。



寄存器功能如下：

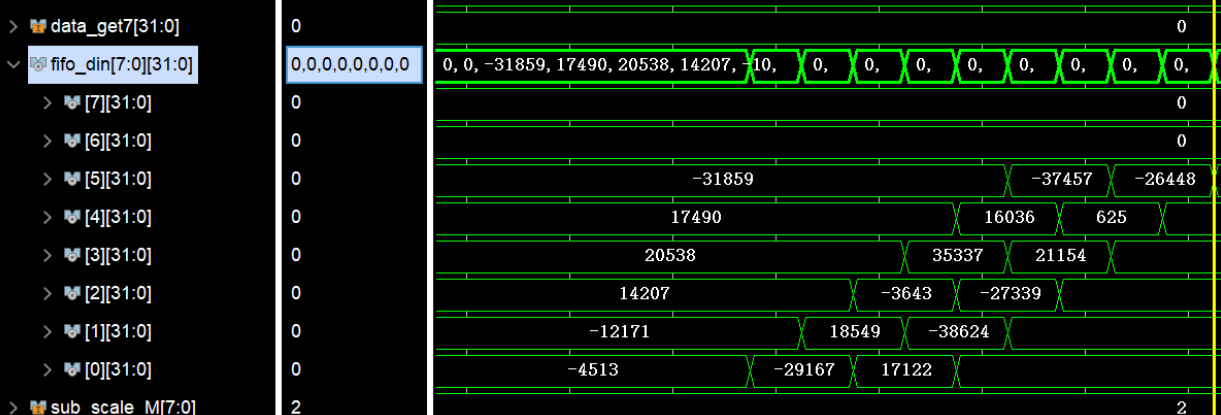
* sub\_scale\_M1：矩阵乘法计算MAC1模块行实际输入计算数据大小；
* sub\_scale\_P1：矩阵乘法计算MAC1模块列实际输入计算数据大小；
* sub\_scale\_M2：矩阵乘法计算MAC2模块行实际输入计算数据大小；
* sub\_scale\_P2：矩阵乘法计算MAC2模块列实际输入计算数据大小；

### 4.3.9 问题9（8※）

阅读Align\_fifo模块代码，结合仿真波形描述数据写入和读出的过程。

整体来讲，Align\_fifo 模块的功能是将从 MAC 模块收集到的错位的并且并列的多行数据整理成为一个串行的数据并送入到 Out\_ctrl 中写入存储器。

首先传入本模块的 data\_get 数据通过接入数据线 fifo\_din [8:0] 而被分别传入到实例化好的8个out\_align\_fifo 模块，如下图所示：



从 out\_align\_fifo 模块返回的数据为，当 out\_ctrl 写入准备好时，存在下图的对应关系，当

fifo\_dout 的值发生变化时，就会改写data的数据，对应关系如下，同时要根据传入的 sub\_scale\_P来记录需要捕获的真实行大小，所以传换成了串行数据以便存储。由于存在两个 Align\_fifo 模块分别对应 MAC1 和 MAC2 的输出，因此需要通过 out\_ctrl\_ready 信号来判断交替来输出。

四、实验结果

环境配置成功，测试结果详见实验步骤。

五、分析与讨论

本次实验综合了前两次lab进行了综合运用，整体对于上板矩阵乘法运行的整体逻辑又有了进一步的了解。