智能计算体系结构课程lab3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，熟悉Vivado使用环境，熟悉Verilog语言，并仔细阅读、理解一个已给出的Verilog实现的矩阵乘法单元的具体实现，并用Vivado复现其波形、在FPGA上进行测试验证；另外，熟悉开发板，移植Linux至ZYNQ7020开发板，并测试开发环境。

二、实验环境：

IDE：Vivado 2019.2 / Vitis 2019.2；

Environment：ZYNQ 7020开发板及其配件；

OS：Windows 10 Professional

三、实验内容与步骤

1. Vivado集成开发环境使用：

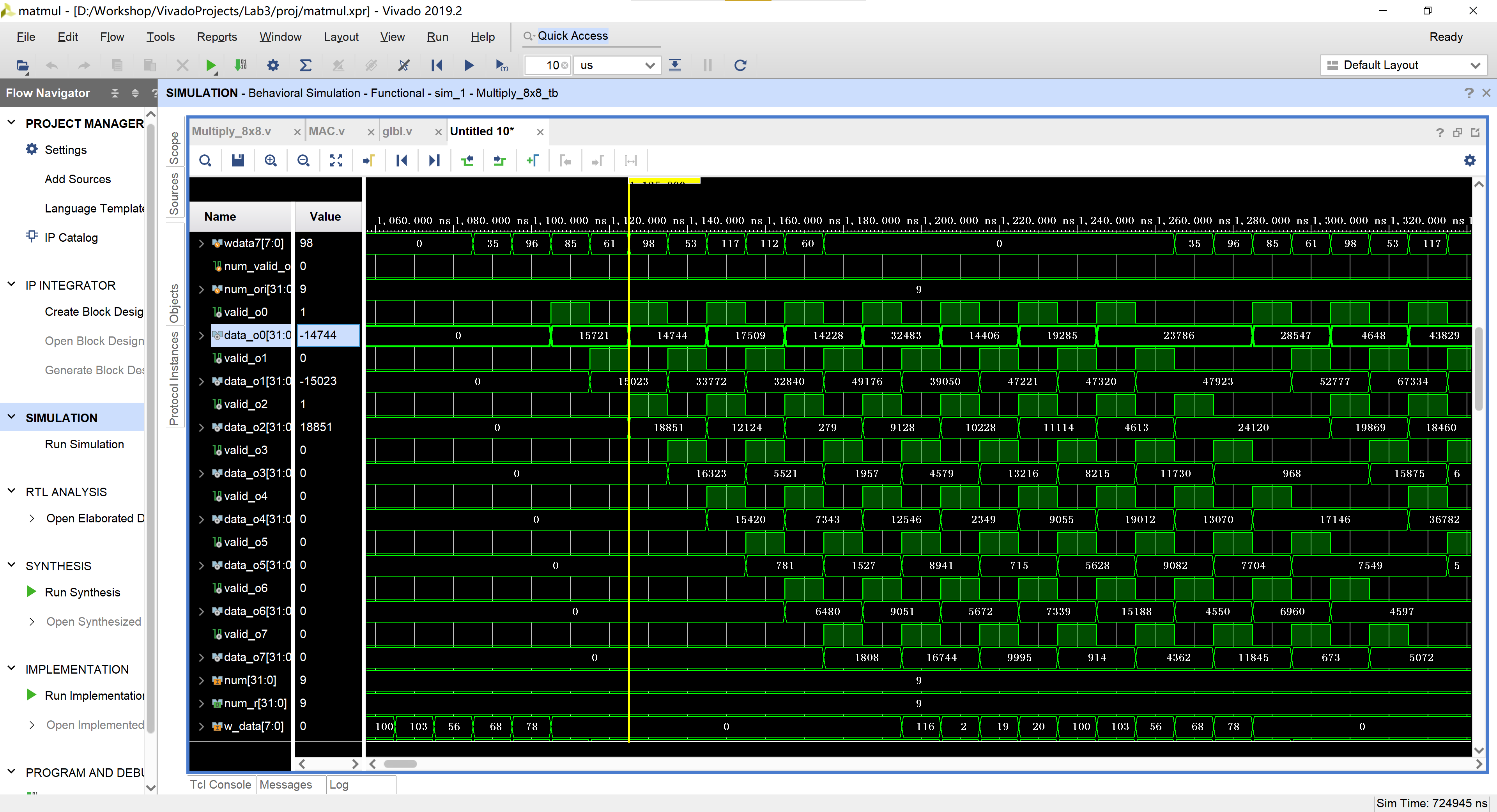
1. 采用ftp访问课程组提供的共享文件夹、并下载Vivado 2019.2安装包，安装后加载提供的lic文件，获得软件使用权限。
2. 建立工程文件夹，使用Vivado引入后，添加源文件，包括：

设计文件MAC.v、Multiply\_8x8.v、

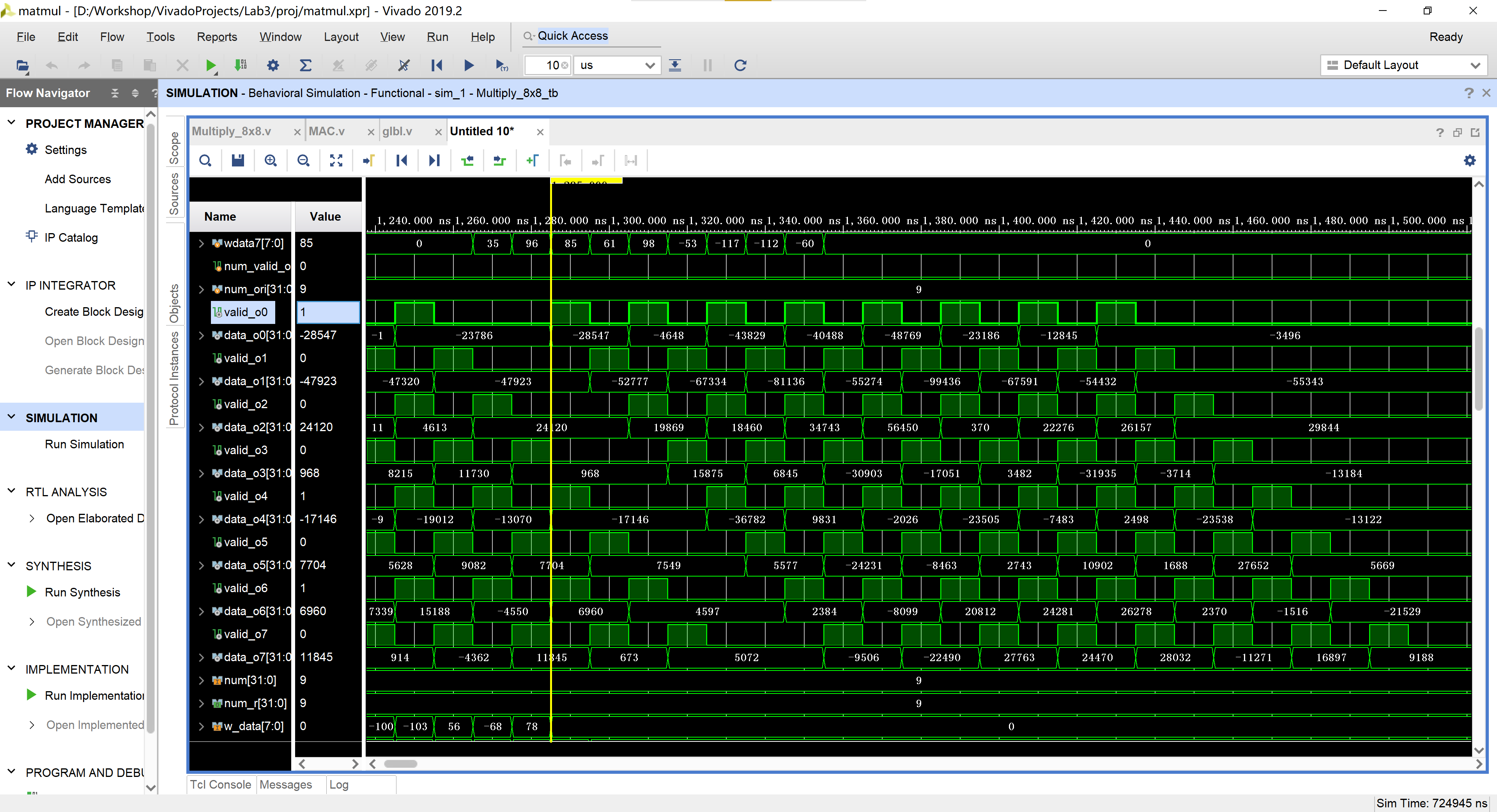
仿真文件Multiply\_8x8\_tb.v；

1. 点击Run Simulation，开始仿真，跑一段时间后查看矩阵输出波形，并与test.py运行后的输出结果进行比对；

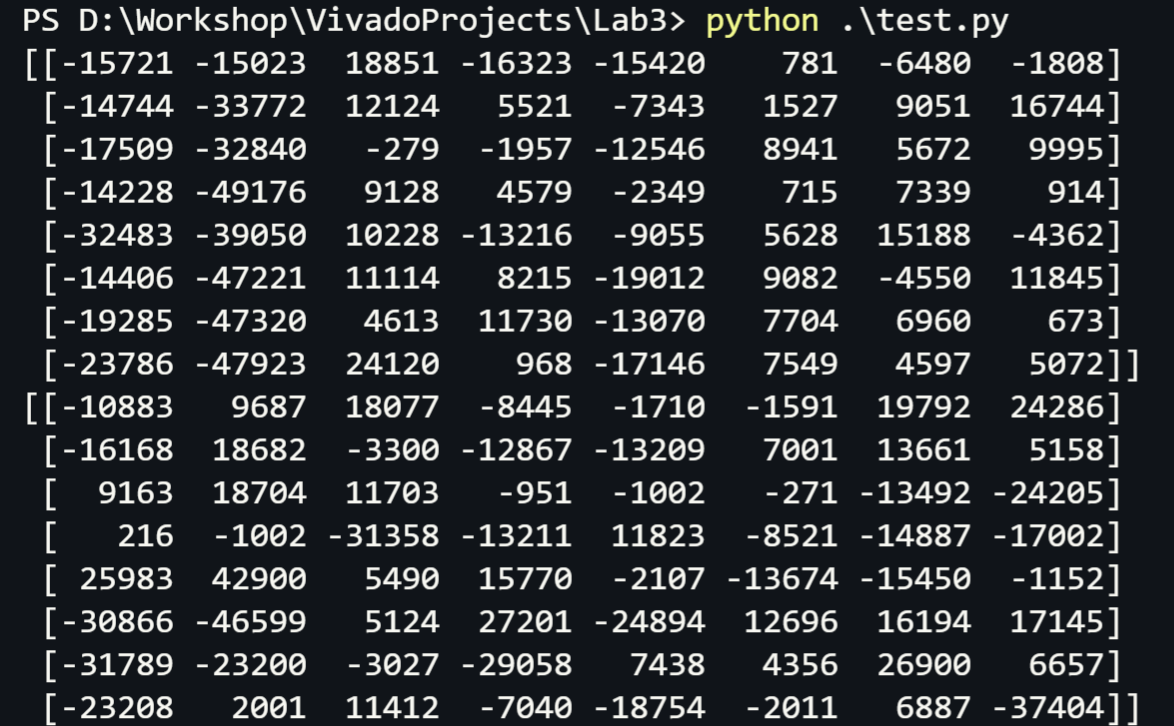
第一次矩阵计算输出部分波形：



第二次矩阵计算输出部分波形：



test.py输出结果：

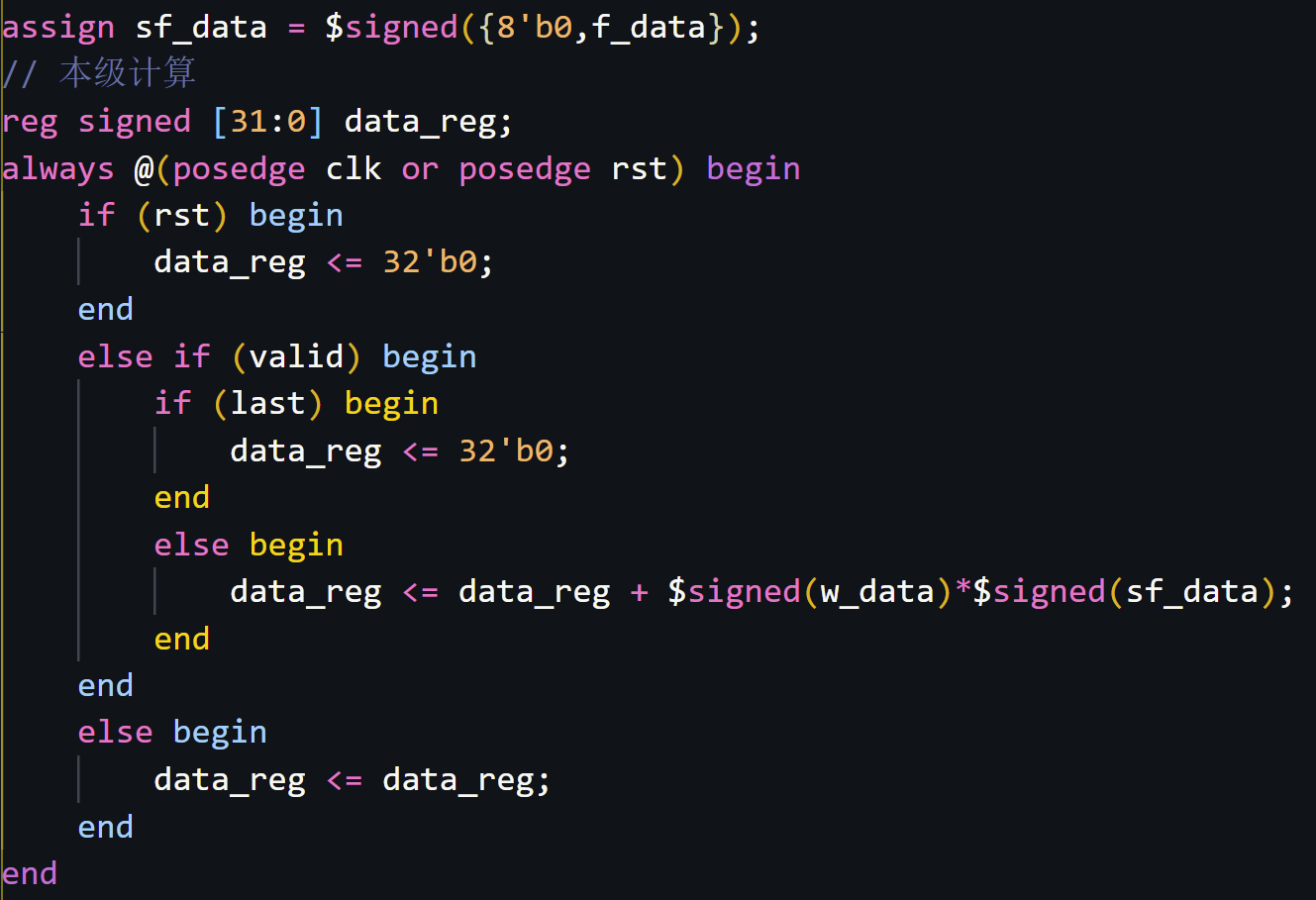


比对后可以发现，第一次计算结果正确，第二次计算结果错误。

2. 阅读源文件，解决问题

1. 问题1：第二个矩阵乘计算出错的原因？

MAC.v部分源码如下：



其中，sf\_data[15:0]变量表示有符号的f\_data[7:0]输入，然而在扩位时采取了在8位的f\_data前添加8位0的方式获取16位的sf\_data，导致当输入f\_data为2进制补码格式的负数时，获得的新sf\_data为与之不对应的正数。例如：当f\_data[7:0]=8’b10010001，对应有符号十进制数-111；sf\_data[15:0]=16’b0000000010010001，对应有符号十进制数145，显然最终乘计算的结果也将是错误的。

1. 问题2：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。

没有修改的必要。因为feature矩阵的数据类型为uint8，即8位无符号整数，在前面加8位0扩位的方式是正确的，不会造成负数扩位值变化的问题。上述第二次矩阵乘计算问题是输入格式有误造成的。

3. MAC.v设计文档

1. 模块设计功能

本模块是一个独立的计算模块，主功能为负责对目标矩阵对应位置结果的计算、与其他结果向输出端的无处理传输。计算部分先对时钟周期传入的行数据与列数据进行乘运算，并进行累加、输出结果；输出本单元的计算结果后，在每个时钟周期又会向上级输出下级单元传输来的结果。

1. 模块接口说明

module MAC(

    input clk, // 时钟信号

    input rst, // 复位使能信号

    // 乘累加长度

    input num\_valid, // 乘累加长度更新使能信号

    input [31:0] num, // 乘累加长度

    output reg num\_valid\_r, // 传给下一单元的本级乘累加长度使能信号

    output reg [31:0] num\_r, // 传给下一单元的本级乘累加长度

    // 纵向数据

    input w\_valid, // 纵向数据有效信号

    input signed [7:0] w\_data, // 纵向数据

    output reg w\_valid\_r, // 传给下一个单元的本级纵向数据有效信号

    output reg signed [7:0] w\_data\_r, // 传给下一单元的本级纵向数据

    // 横向数据

    input f\_valid, // 横向数据有效信号

    input [7:0] f\_data, // 横向数据

    output reg f\_valid\_r, // 传给下一个单元的本级横向数据有效信号

    output reg [7:0] f\_data\_r, // 传给下一单元的本级横向数据

    // 数据输出，纵向向上传播

    input valid\_l, // 更新数据无处理输出的使能信号

    input signed [31:0] data\_l, // 上级传来的数据

    output reg valid\_o, // 数据输出有效信号

output reg signed [31:0] data\_o // 数据输出端

// 输出数据需要经过选择

    );

1. 模块设计思想及流程描述

一个独立的MAC单元独立负责对于结果矩阵对应位置的值的计算。当对应行、列均传入到本单元时，将f\_valid与w\_valid赋为真，对f\_data与w\_data进行乘计算并于data\_reg累加。当根据num\_cnt与num判断当前计算是最后一次乘计算（即last为真）时，在下一个时钟周期输出本单元计算值，并将valid\_l赋为真，在下一个时钟周期在data\_o输出上一级传入的data\_l。

1. 内部关键信号与变量描述

reg [31:0] num\_cnt; // 当前已进行乘计算的数据对数

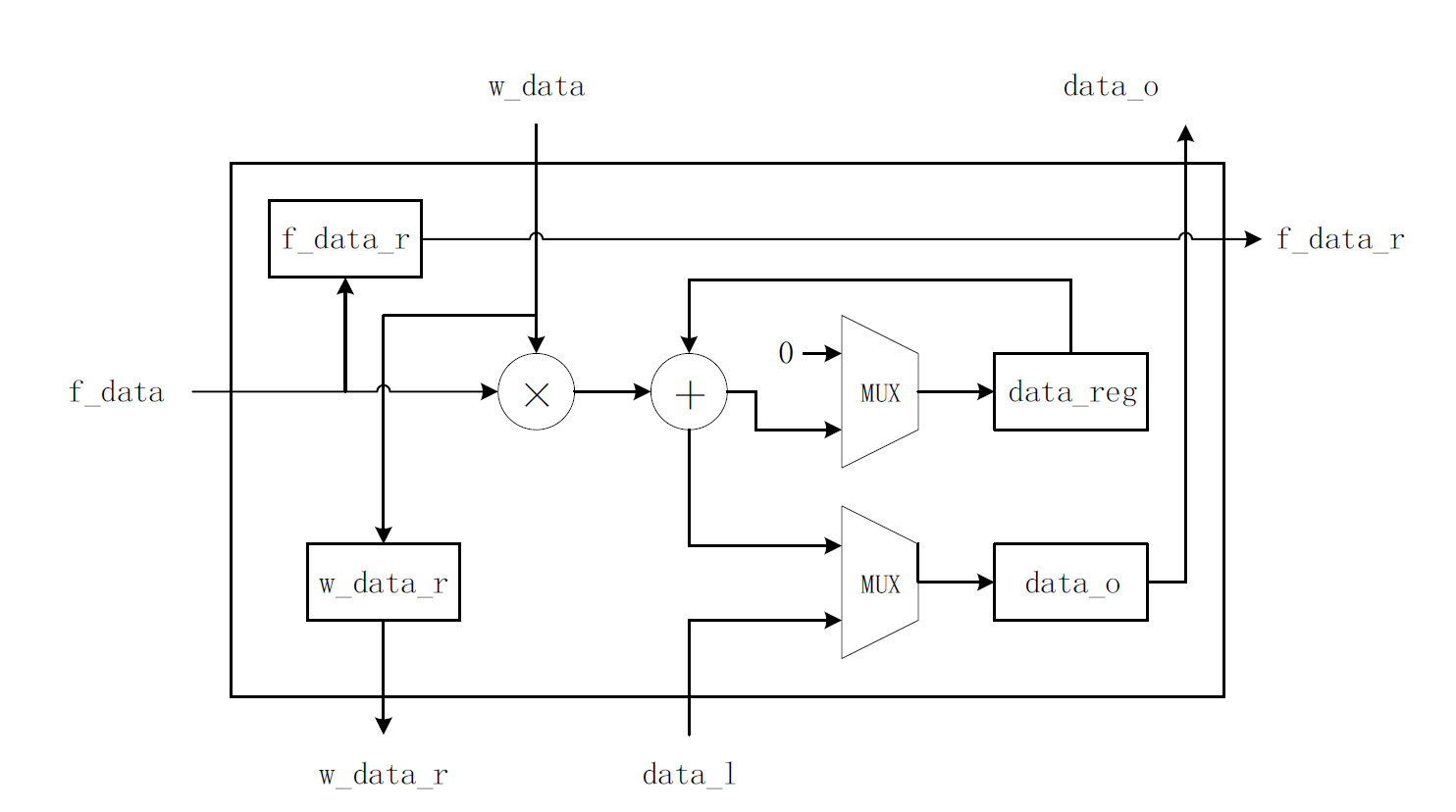
wire valid; // 当且仅当w\_valid与f\_valid同时为真时为真

wire last; // 当且仅当已进行num-1次乘计算时赋为真

wire signed [15:0] sf\_data; // 扩展为16位后的f\_data

reg signed [31:0] data\_reg; // 用于累加乘计算结果，并在结束时输出到data\_o

1. 模块框图



4. Multiply\_8x8.v设计文档

1. 模块设计功能

本模块是MAC.v的父模块，由64（8x8）个MAC模块组成，其中包含了各MAC模块之间、以及输入输出接口与MAC模块之间的连接，使得按时钟周期输入的信号输入阵列后，可以实现两个矩阵的乘计算，并输出结果。

1. 模块接口说明

module Multiply\_8x8(

    input clk, // 时钟信号

    input rst, // 复位使能信号

    input fvalid0, // fdata0输入有效使能信号

    input [7:0] fdata0, // 特征矩阵第1行输入

    input fvalid1, // fdata1输入有效使能信号

    input [7:0] fdata1, // 特征矩阵第2行输入

    input fvalid2, // fdata2输入有效使能信号

    input [7:0] fdata2, // 特征矩阵第3行输入

    input fvalid3, // fdata3输入有效使能信号

    input [7:0] fdata3, // 特征矩阵第4行输入

    input fvalid4, // fdata4输入有效使能信号

    input [7:0] fdata4, // 特征矩阵第5行输入

    input fvalid5, // fdata5输入有效使能信号

    input [7:0] fdata5, // 特征矩阵第6行输入

    input fvalid6, // fdata6输入有效使能信号

    input [7:0] fdata6, // 特征矩阵第7行输入

    input fvalid7, // fdata7输入有效使能信号

    input [7:0] fdata7, // 特征矩阵第8行输入

    input wvalid0, // wdata0输入有效使能信号

    input signed [7:0] wdata0, // 权重矩阵第1行输入

    input wvalid1, // wdata1输入有效使能信号

    input signed [7:0] wdata1, // 权重矩阵第2行输入

    input wvalid2, // wdata2输入有效使能信号

    input signed [7:0] wdata2, // 权重矩阵第3行输入

    input wvalid3, // wdata3输入有效使能信号

    input signed [7:0] wdata3, // 权重矩阵第4行输入

    input wvalid4, // wdata4输入有效使能信号

    input signed [7:0] wdata4, // 权重矩阵第5行输入

    input wvalid5, // wdata5输入有效使能信号

    input signed [7:0] wdata5, // 权重矩阵第6行输入

    input wvalid6, // wdata6输入有效使能信号

    input signed [7:0] wdata6, // 权重矩阵第7行输入

    input wvalid7, // wdata7输入有效使能信号

    input signed [7:0] wdata7, // 权重矩阵第8行输入

    input num\_valid\_ori, // 矩阵乘累加长度有效使能信号

    input [31:0] num\_ori, // 矩阵乘累加长度

    output valid\_o0, // data\_o0输出有效信号

    output signed [31:0] data\_o0, // 输出矩阵的第1列结果

    output valid\_o1, // data\_o1输出有效信号

    output signed [31:0] data\_o1, // 输出矩阵的第2列结果

    output valid\_o2, // data\_o2输出有效信号

    output signed [31:0] data\_o2, // 输出矩阵的第3列结果

    output valid\_o3, // data\_o3输出有效信号

    output signed [31:0] data\_o3, // 输出矩阵的第4列结果

    output valid\_o4, // data\_o4输出有效信号

    output signed [31:0] data\_o4, // 输出矩阵的第5列结果

    output valid\_o5, // data\_o5输出有效信号

    output signed [31:0] data\_o5, // 输出矩阵的第6列结果

    output valid\_o6, // data\_o6输出有效信号

    output signed [31:0] data\_o6, // 输出矩阵的第7列结果

    output valid\_o7, // data\_o7输出有效信号

    output signed [31:0] data\_o7 // 输出矩阵的第8列结果

    );

1. 模块设计思想及流程描述

参照模块框图，对于第i行j列的MAC模块，其序号为j+(i-1)\*8-1。该单元的计算需要从特征矩阵的第i行与权重矩阵的第j列获取数据并做乘运算累加。输入的每行间均具有一个时钟周期的延迟、数据的纵向、横向传播，以保证对应行数据与列数据到达每个MAC单元时间的一致性。计算完成后，数据会纵向向输入数据的相反方向传输回去，在每个输出端按时钟周期连续输出。

1. 内部关键信号与变量描述

wire    [  63:   0]   num\_valid                ; // 64个单元对应的num输入有效信号

wire    [  31:   0]   num           [63:0]     ; // 64个单元对应的乘累加长度

wire    [  63:   0]   num\_valid\_r              ; // 连接64个单元对应num\_valid\_r输出的接口

wire    [  31:   0]   num\_r         [63:0]     ; // 连接64个单元对应num\_r输出的接口

wire    [  63:   0]   w\_valid                  ; // 64个单元对应的w\_data输入有效接口

wire    [   7:   0]   w\_data        [63:0]     ; // 64个单元对应的权重矩阵数据输入接口

wire    [  63:   0]   w\_valid\_r                ; // 连接64个单元对应w\_valid\_r输出的接口

wire    [   7:   0]   w\_data\_r      [63:0]     ; // 连接64个单元对应w\_data\_r输出的接口

wire    [  63:   0]   f\_valid                  ; // 64个单元对应的f\_data输入有效接口

wire    [   7:   0]   f\_data        [63:0]     ; // 64个单元对应的特征矩阵数据输入接口

wire    [  63:   0]   f\_valid\_r                ; // 连接64个单元对应f\_valid\_r输出的接口

wire    [   7:   0]   f\_data\_r      [63:0]     ; // 连接64个单元对应f\_data\_r输出的接口

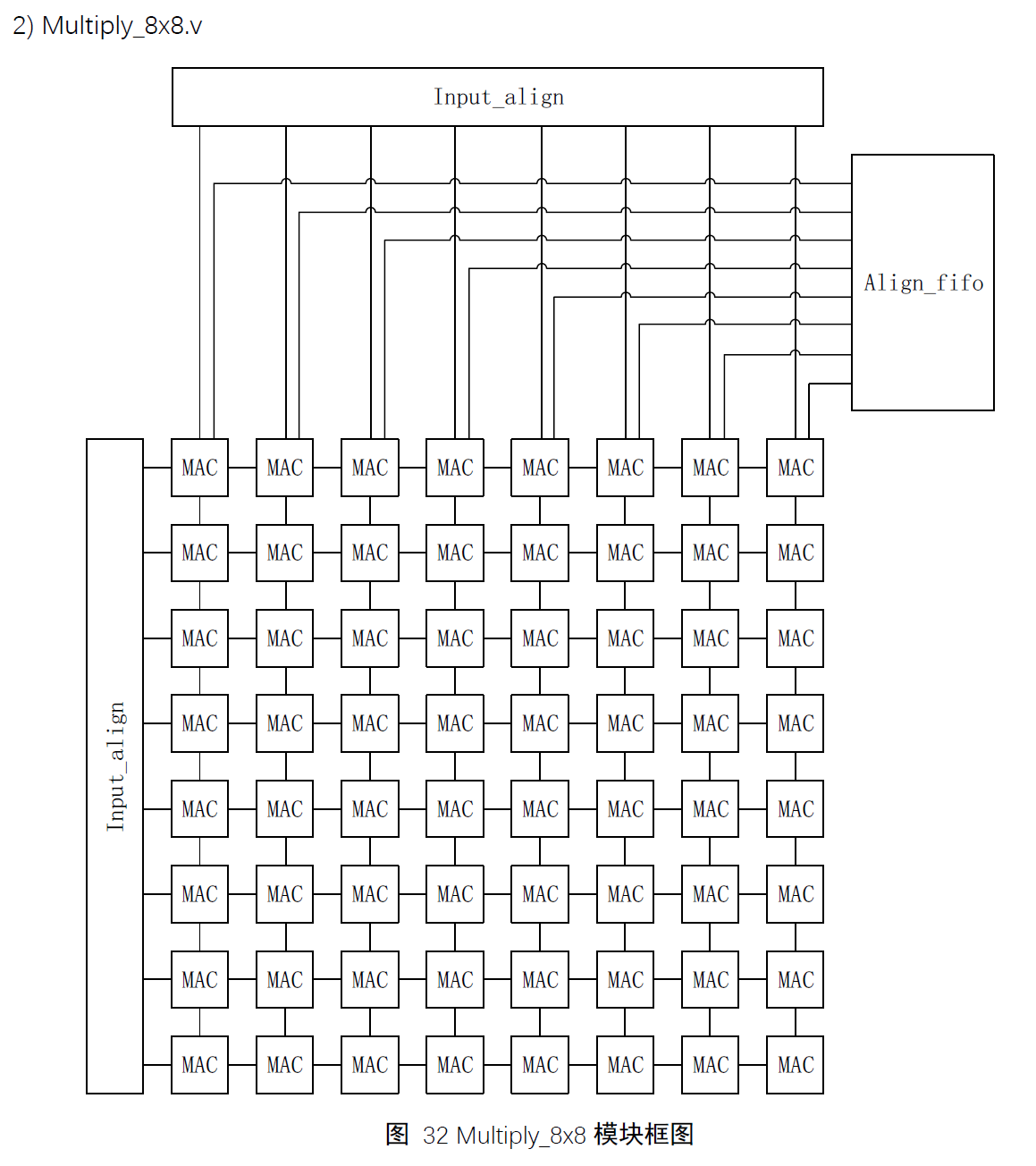
wire   [  63:   0]   valid\_l                  ; // 64个单元对应的data\_l输入有效接口

wire    [  31:   0]   data\_l        [63:0]     ; // 64个单元对应的data\_l输入接口（上级数据返回传输）

wire    [  63:   0]   valid\_o                  ; // 64个单元对应的data\_o输出有效接口

wire    [  31:   0]   data\_o        [63:0]     ; // 64个单元对应的data\_o输出接口（上级、本单元数据返回传输）

1. 模块框图



5. Linux移植

制作SD卡文件系统、启动ZYNQ上的文件系统、安装开发环境。

四、实验结果

环境配置成功，详细波形结果见第三部分实验步骤。

五、分析与讨论

本次实验熟悉了Vivado的开发环境与Verilog语言，并通过已给出的矩阵乘法加速硬件实现，初步理解了通过硬件加速智能计算常用的计算模式的方式。另外，完成了Linux移植，为后续开发过程完成了环境准备。