**毕业设计项目**

**《生理信号检测卷积神经网络硬件加速器》**

**仿真测试报告**

|  |  |
| --- | --- |
| 学 院： | 电子科学与工程学院 |
| 专 业： | 物联网专业 |
| 学 号： | 06117113 |
| 姓 名： | 吴中行 |
| 指导老师： | 刘昊 |
| 日 期： |  |
| 成 绩： |  |

目录

[第 1 章 加速器模块介绍 3](#_Toc71593831)

[1. 1 加速器架构 3](#_Toc71593832)

[1. 2 功能介绍和数据流概况 3](#_Toc71593833)

[第 2 章 代码文件结构 5](#_Toc71593834)

[第 3 章 仿真测试 6](#_Toc71593835)

[3. 1 测试方案设计 6](#_Toc71593836)

[3. 2 控制功能仿真测试 7](#_Toc71593837)

[3. 3 数据读写功能仿真测试 7](#_Toc71593838)

[3.3.1 In\_Out\_Buffer模块仿真测试 7](#_Toc71593839)

[3.3.2 Input\_Regfile模块仿真测试 9](#_Toc71593840)

[3. 4 权重读写功能仿真测试 10](#_Toc71593841)

[3. 5 PE计算功能仿真测试 11](#_Toc71593842)

[3. 6 Relu&Pooling计算功能仿真测试 11](#_Toc71593843)

[3.6.1 Relu8模块仿真测试 12](#_Toc71593844)

[3.6.2 Output\_Regfile模块仿真测试 12](#_Toc71593845)

[3.6.3 Pooling模块仿真测试 14](#_Toc71593846)

# 加速器模块介绍

## 加速器架构

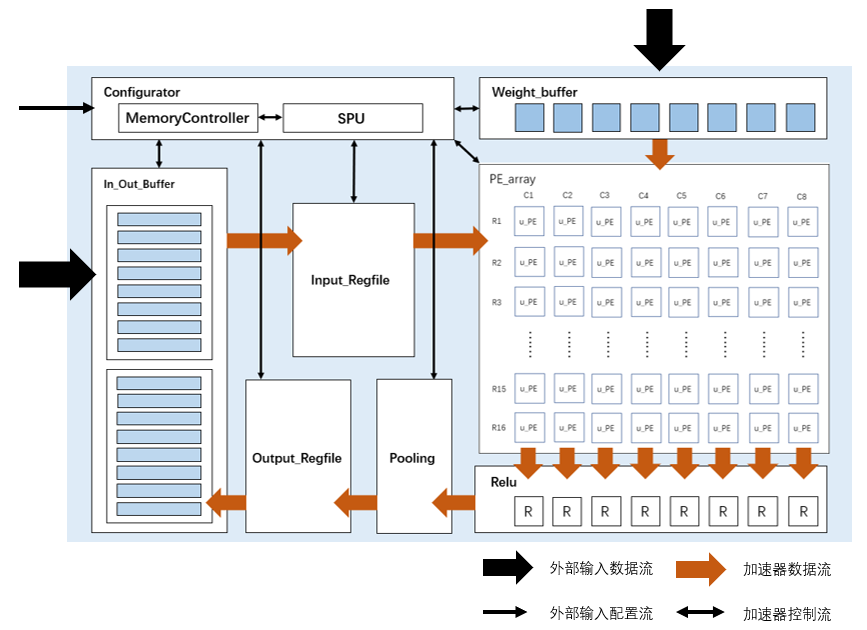
如下图所示，加速器分为Configurator, Weight\_buffer, PE\_array, In\_Out\_Buffer, Input\_Regfile, Output\_Regfile, Relu, Pooling共8个模块，有三股外部的数据流输入，分别是网络参数数据流、权重数据流和ECG信号数据流。

图1-1 硬件加速器架构图

## 功能介绍和数据流概况

ECG信号输入到第一块RAM中，接着按批次输出到Input Regfile，Input Regfile把数据按一定的顺序输出到PE阵列里与权重进行卷积计算。计算结束，把计算结果输出到Output Regfile，待数据整理好将其发送到Relu Pooling模块进行激活池化操作，最后池化结束将结果输出到Inout Buffer的第二块RAM中，待所有结果都进入到RAM之后，第一层运算结束。第二块RAM的数据是下一层的输入，第一块RAM即将存放下一层的输出结果，两块RAM如此乒乓操作，直到6层的卷积层计算结束。

Configurator是整个加速器的控制部分，里面包含SPU（State Processing Unit）和Memory controller。SPU是控制整个加速器工作状态的状态机，Memory controller主要工作是配置网络参数给给个模块，并控制Weight Buffer和Inout Buffer的读写功能，两个模块协同工作控制整个加速器的运行。

Inout Buffer有两块RAM，每块RAM里面有个8个bram，每个bram的深度是1024，位宽是8bits。Weight Buffer有8个bram，每个bram的深度是2048，位宽也是8bits。ECG信号输入到第一块RAM中，然后再按批次输出到PE阵列进行计算。计算一定周期之后，最终把计算结果输出到Inout Buffer的另一块RAM中，直到所有计算的结果都进入到RAM之后，第一层运作结束。第二块RAM的数据是下一层的输入，第一块RAM即将存放下一层的输出结果，两块RAM如此乒乓操作，直到6层的卷积层计算结束。

PE阵列的大小是16\*8，行数16，列数8,8列PE的权重输入对应着Weight Buffer的8个bram的输出，16行PE的特征数据输入是Input Regfile的输出。PE阵列的工作原理如下：在第一个clk的时候，第一列16个PE同时进行第一次计算，然后第二个clk的时候，第二列PE进行第一次计算，第一列PE进行第二次计算，如此从右至左地脉动，直到第八个周期，第八列PE进行第一次计算，第一列PE进行第八次计算，整个脉动阵列被激活起来，都在进行着计算工作。

当计算完一定的周期之后，PE阵列开始输出数据，每列都会有16个计算结果，每个计算结果从上往下逐个往外输出，由于每一列输入之间相隔一个周期，所以每列间的输出都是相隔一个周期。在第一个clk的时候，第一列的PE输出第一个计算结果，然后第二个clk的时候，第二列PE输出第一个计算结果，第一列PE输出第二个计算结果，如此从右至左的顺序，直到第八个周期，第八列PE输出第一个计算结果，第一列PE输出第八个计算结果，整个脉动阵列都在进行着输出工作。

# 代码文件结构

本项目verilog代码文件结构如下图所示。共分为4级。第一级为TOP层，包括ECG\_TOP.v，第二级文件包括Configurator.v, In\_Out\_Buffer.v, Weight\_buffer.v, PEarray16\_8.v, Relu8.v, Output\_Regfile8\_9.v, Pooling8.v, Pool2IOB.v共9个文件。第三级文件包括SPU\_ECG.v, Mem\_Ctrl.v, memory\_map1-8.v, memory.v, weight\_memory1-8.v, u\_PE.v, Relu.v, Output\_regfile.v, Pooling.v 共23个文件，第四级文件包括mult1.v, RCA\_8bit.v共2个。

图2- 1 Verilog设计代码文件结构

# 功能仿真测试

## 测试方案设计

对设计好的加速器各个模块进行独立测试和模块间共同测试从而实现对加速器各个模块的功能的验证和加速器整体功能的验证。

首先将一级模块分为5个主要功能，分别为控制功能，数据读写功能，权重读写功能，PE计算功能和Relu&Pooling计算功能。

表3-1 仿真测试表

|  |  |  |
| --- | --- | --- |
| 功能 | 涉及模块 | 测试功能 |
| 控制功能 | SPU\_ECG  Mem\_Crtl | 主状态机状态跳转  计算状态机状态跳转  当前计算层数跳转 |
| 数据读写功能 | In\_Out\_Buffer  Input\_Regfile | 数据输入  数据输出 |
| 权重读写功能 | Weight\_buffer | 权重输入  权重输出 |
| PE计算功能 | u\_PE | 乘法器  加法器  数据输入  数据输出 |
| Relu&Pooling计算功能 | Relu8  Pooling8  Output\_Regfile  Pool2IOB | 激活函数  数据截断  池化操作  数据输入  数据输出 |

## 控制功能仿真测试

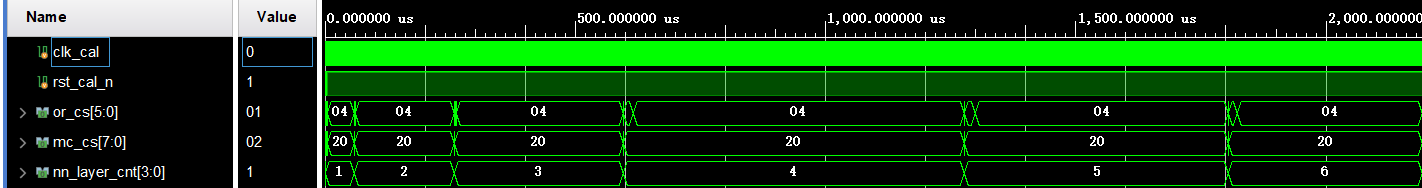
Configurator里面包含SPU和MemoryController，主要是测试SPU和MemoryController之间的通讯情况，检测状态机的跳转状态即可。从下图可以看出nn\_layer\_cnt从1到6再回到1，即6层网络的状态全部跑通。

图3- 1 控制功能仿真测试结果

## 数据读写功能仿真测试

这部分功能模块仿真测试主要涉及In\_Out\_Buffer和Input\_Regfile两个模块。

### In\_Out\_Buffer模块仿真测试

In\_Out\_Buffer从Memory\_Controller获取数据读写的地址，状态机由or\_cs[OR\_FT\_WT]跳转住or\_ns[OR\_CAL]时，开始执行数据读出。当pe\_end置高时，Data\_O\_vld置高，并计算当前所读取的地址。Bm\_cnt用于计数当前读取操作的次数，当Bm\_cnt==Bm\_times时，置高rd\_done，rd\_done作为pe\_end1→0翻转的标志位输出至PE\_Array或mcu。

表3-2仿真第二层的网络参数表格

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名称 | 值 | 参数名称 | 值 |
| nn\_layer\_cnt | 2 | M | 8 |
| IX | 894 | S | 2 |
| OX | 220 | K | 12 |
| N | 16 | Hu | 42 |

**数据输入：**

接收到Output\_Regfile的使能信号Data\_I\_vld后开始接收数据。根据In\_Out\_Buffer数据输入方式：每次写入所有输出map的一列，写完所有输出map后写输出map的下一列，以此类推写完所有map。下图波形中wr\_addr为所有输出map第一列地址。omap\_addr\_span\_cnt为当前写入map的列的索引。

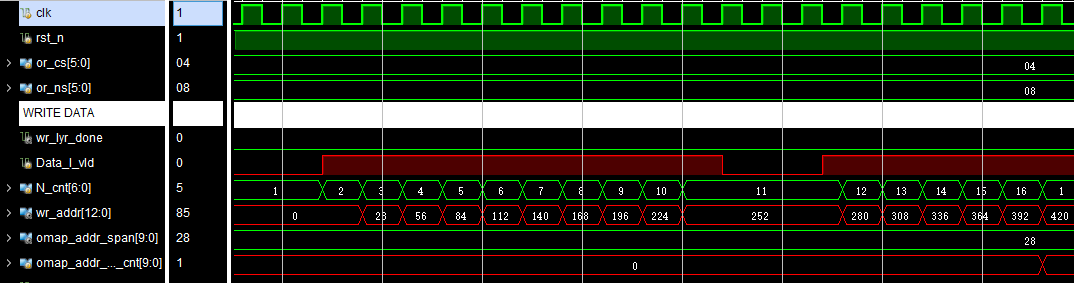


图 3-2 In\_Out\_Buffer模块写入测试结果图

下图波形中omap\_addr\_span\_cnt由0变为1，表示写完输出map第一列后写第二列。

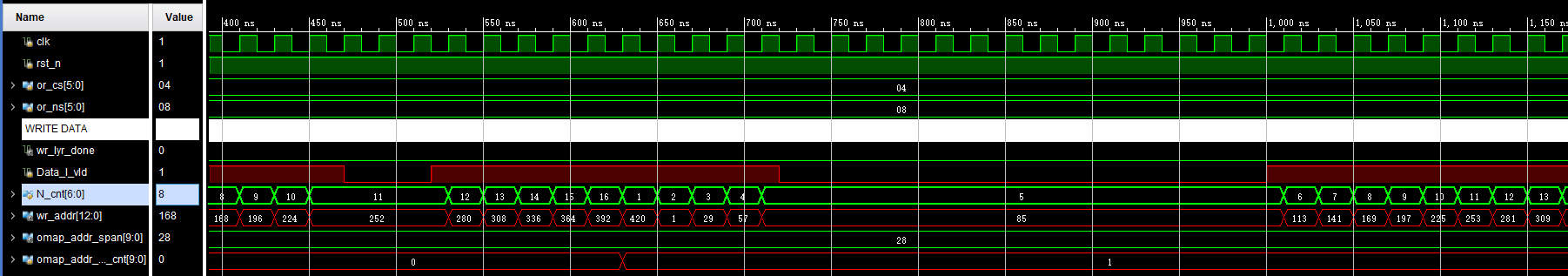


图 3-3 In\_Out\_Buffer模块写入测试结果图

下图波形为输出该网络层所有map后，wr\_lyr\_done置高。在omap\_addr\_span\_cnt变为27时，final\_column置高表示次写入为输出map最后一列，同时输出final\_zeros为4，表示最后一列需要补4个零。

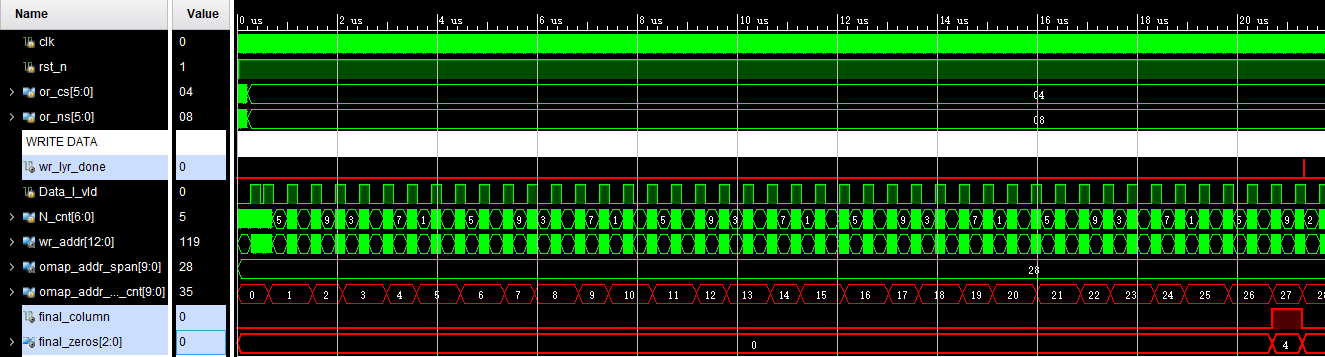


图 3-4 In\_Out\_Buffer模块写入测试结果图

**数据输出：**

如下图波形所示，接受到pe\_end且Bm\_cnt！=Bm\_cnt时开始进行数据传输，其中Bm\_times为数据输出的次数，Bm\_cnt为其计数器。

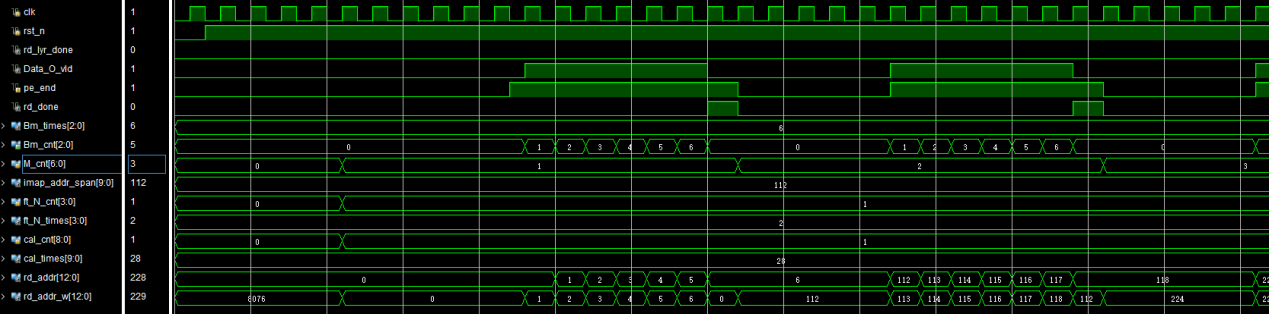


图 3-5 In\_Out\_Buffer模块读出测试结果图

### Input\_Regfile模块仿真测试

**数据输入：**

接收到Data\_I\_vld信号后，根据Bm\_cnt计算写入的首地址wr\_addr，一次写入In\_Out\_Buffer的Bank数量（8个）的数据。写入的周期由K，S，R和Bank数量（8）控制。

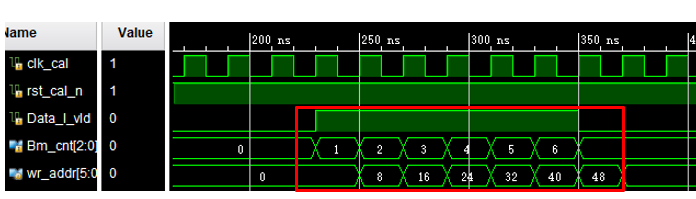


图 3-6Input\_Regfile模块写入测试结果图

按照Input\_Refile数据写入方式，在Data\_I\_vld输入后，Input\_Regfile开始写入数据。由于In\_Out\_Buffer输出至Input\_Regfile的Bank（8）个数据在Input\_Regfile中地址连续，因此只需计算输入数据的起始数据wr\_addr，其余7个输入数据的地址由wr\_addr直接得出。上图中共输入6次，合计48个数据至Input\_Regfile。每次传输起始地址均从0开始。

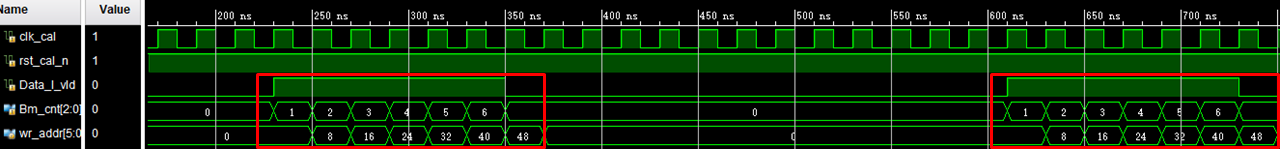


图 3-7 Input\_Regfile模块写入测试结果图

**数据输出：**

接收到pe\_end信号后，产生Data\_O\_vld信号，每个周期写地址加1计算输出的首地址rd\_addr，根据pe\_end保持的周期控制数据输出次数。一次写入PE\_Array的R数量（16个）的数据。

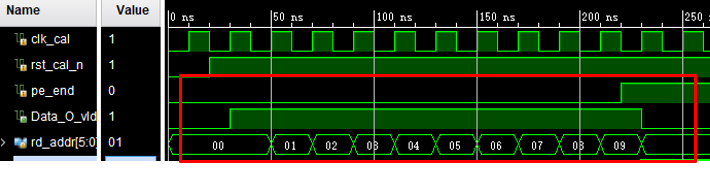
****

图 3-8 Input\_Regfile模块读出测试结果图

按照Input\_Refile数据输出方式，在Data\_O\_vld输入后，Input\_Regfile开始输出数据。由于Input\_Regfile输出至PE\_Array的R（16）个数据在Input\_Regfile中地址相隔为步长S，因此只需计算输出数据的起始数据rd\_addr，其余15个输入数据的地址由rd\_addr直接得出。数据共输入K次至PE\_Array。

## 权重读写功能仿真测试

这部分功能模块仿真测试主要涉及Weight\_buffer模块。

**数据输入：**

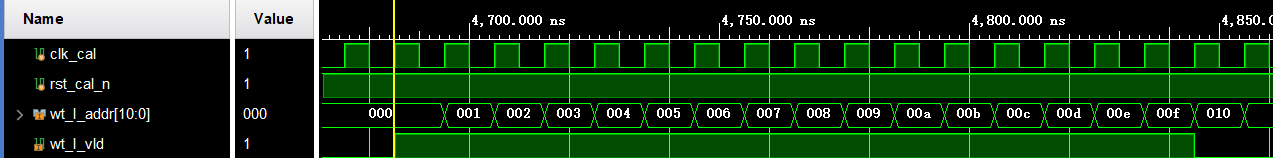
 第一层的权重值只有16\*8个，每一个weight的位宽是1byte，每一次同时写入8个数据分别写入8bank中，因此对于第一层来说只需要写入16clk就可以把第一层的权重值全部写入到bank中。

图 3-9 Weight\_buffer模块写入测试结果图

**数据输出：**

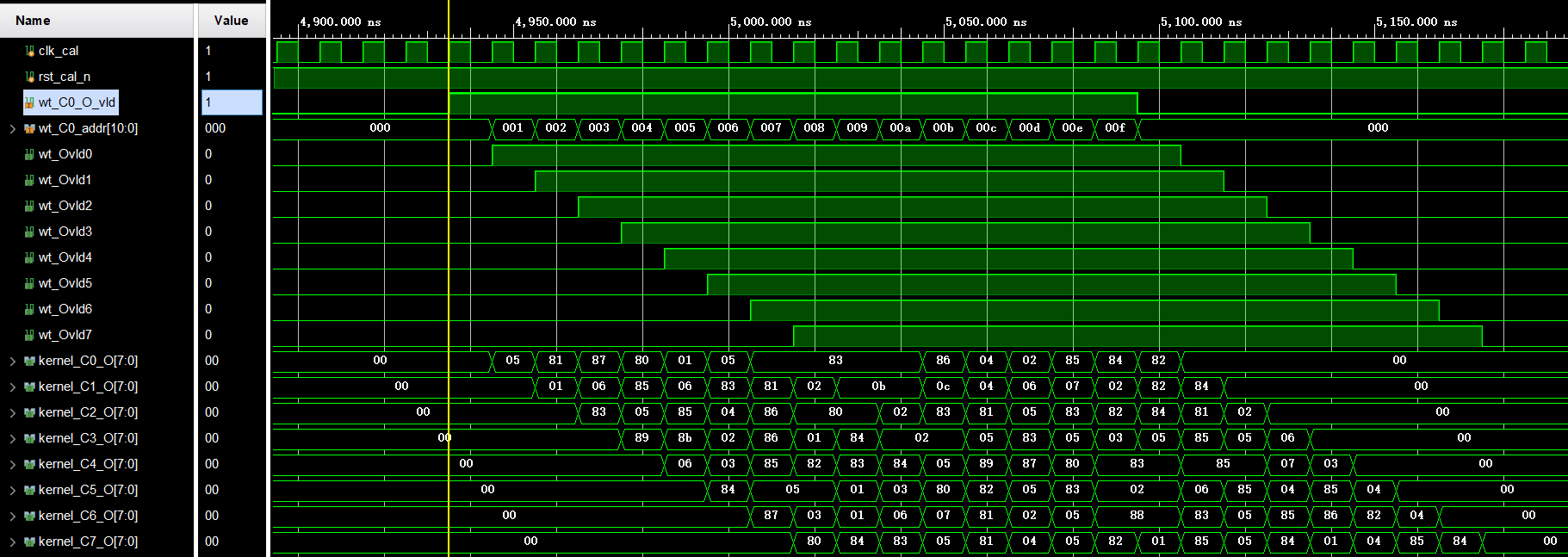
Weight\_Buffer接收到memctrl的wt\_C0\_O\_vld和wt\_C0\_addr信号，就开始读出bank0中的数据，从波形上可以看出每个bank都是依次读出数据的，每个bank隔一个clk读出数据，与PE阵列从左到右的脉动计算匹配。

图 3-10 Weight\_buffer模块读出测试结果图

## PE计算功能仿真测试

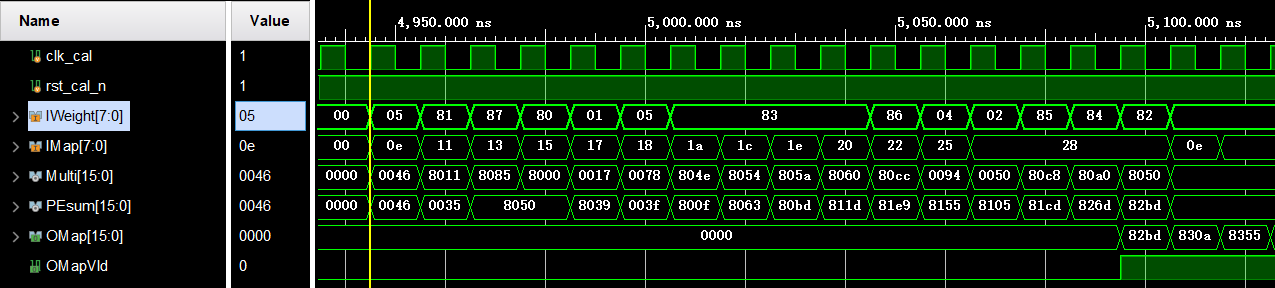
这里主要验证pe单元的计算结果正确性，一个pe的主要组成是一个8位的乘法器和一个16位的加法器。Multi是乘法器的运算结果，刚开始两个输入数据为05和0e（HEX），即乘法结果为(5)\*(14) = 70，即16进制的0046结果正确。紧接着的两个周期81\*11和87\*13，81\*11即为(-1)\*(17) = -17，87\*13即为(-7)\*(19) = -133，所以Multi的结果输出是8011和8085（十六进制的-17和-133）。PEsum是加法器的运算结果，第一个周期就是乘法器的结果0046，第二个周期的PEsum结果为0035是0046和8011相加，结果正确。

图3-11 PE模块计算测试结果图

## Relu&Pooling计算功能仿真测试

这部分功能涉及了Relu8，Pooling8，Output\_Regfile，Pool2IOB四个模块。每一列的PE都接一个Relu，Output\_regfile的数据输入是Relu8的计算结果输出，再输入Pooling8和Pool2IOB之中，最终重组后进入In\_Out\_Buffer。

### Relu8模块仿真测试

Relu模块的功能主要为作为激活函数。有两种激活的模式[0:2]和[0:1]，即大于上限取上限，小于下线取下限，在区域中取本身，前两层用[0:2]模式，后四层用[0:1]模式。

图3-12 激活函数

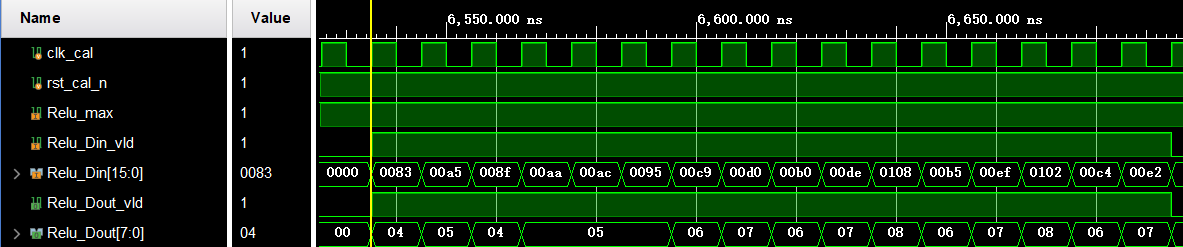
这里还有很重要的一点就是，16bit的加法器输出是16bit的累加结果，因为这一层的输出仍然是下一层的输入，所以位宽要保持8bit，要进行截断。因为输入8bit的数是一个小数，位数分配是1（符号位）+2（整数位）+5（小数位）=8位，所以16位的累加结果，位数分配是1（符号位）+5（整数位）+10（小数位）=16位，进行截断取8位，即取最高的符号位1、整数位低2位、小数位的高5位，一共8位，所以0083（0000\_0000\_1000\_0011）的截断结果为0000\_0100（HEX为04）。之后进行激活操作，输入小于2，输出还是04，结果正确。

图3-13 Relu模块仿真测试结果图

### Output\_Regfile模块仿真测试

**数据写入：**

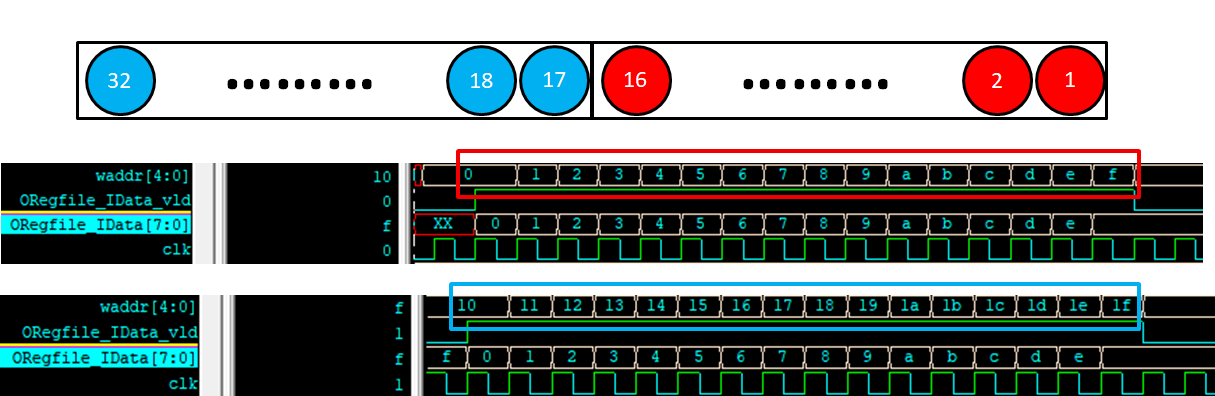
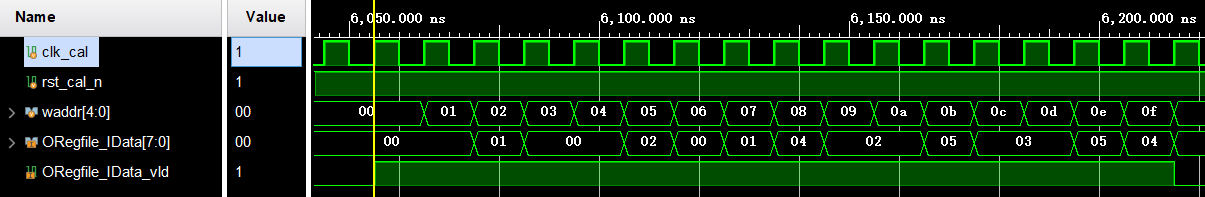
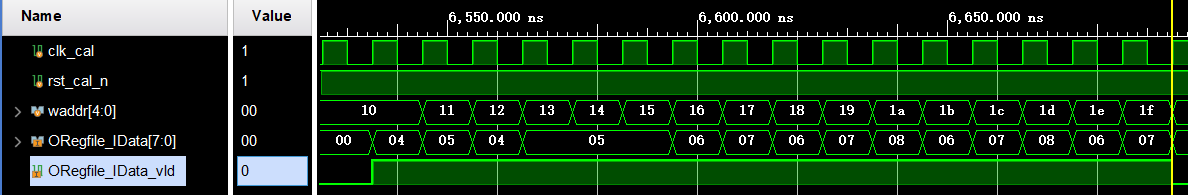


图 3-14 Output\_regfile模块写入测试结果图



按照上面的原理的原理介绍，红色为单数次的16个byte数据写入，蓝色为双数次的16个byte数据写入，如此反复循环写入直到结束。

**数据读出：（举例：P=2,S=1的情况）**

Output\_regfile的数据读出有三种情况，输入的数据是0-f的循环，具体如下：

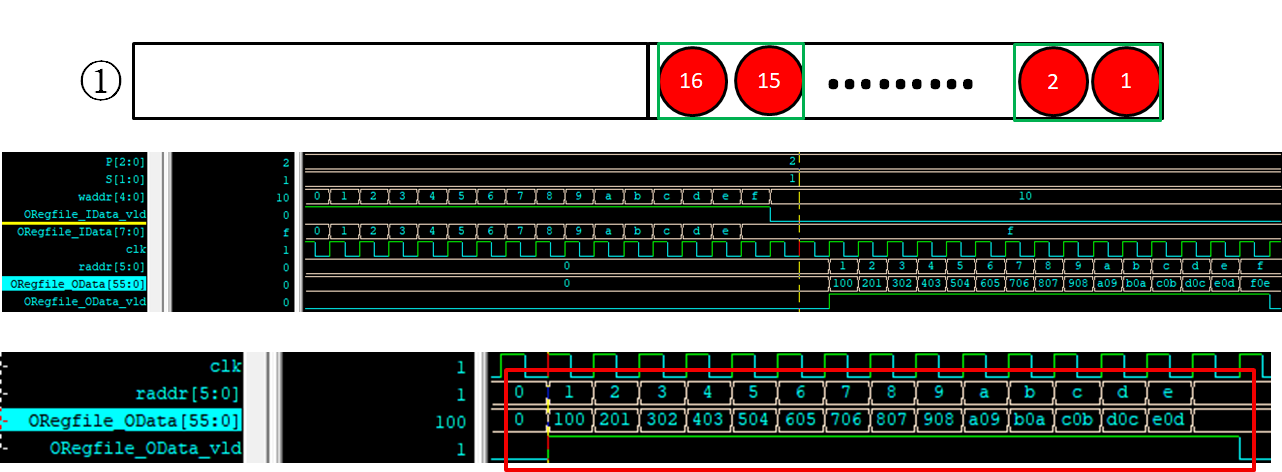
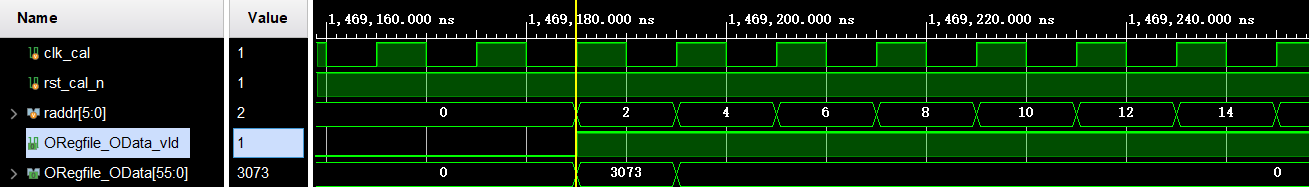


图 3-15 Output\_regfile模块读出测试结果图

如红框所示，地址从0（红圈1）开始读，到地址e（红圈14）结束，每次读出2个（P个）byte的数据输出给RP，而且数据的间隔为2（S=2），验证正确。

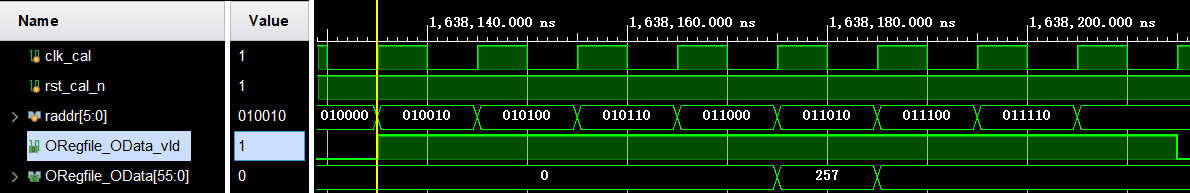
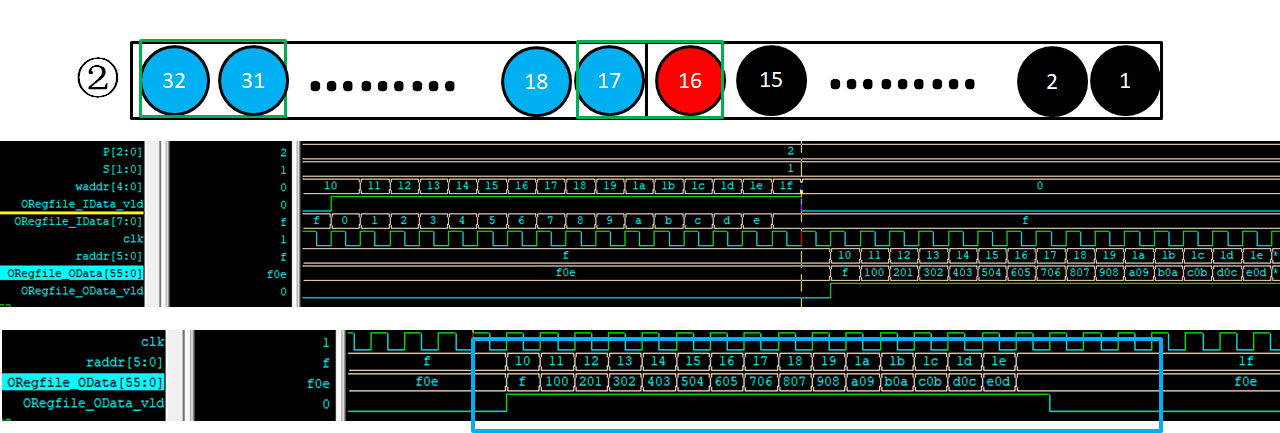


图 3-16 Output\_regfile模块读出测试结果图

如蓝框所示，地址从16（红圈16）开始读，到地址30（蓝圈31）结束，每次读出2个（P个）byte的数据输出给RP，而且数据的间隔为2（S=2），验证正确。

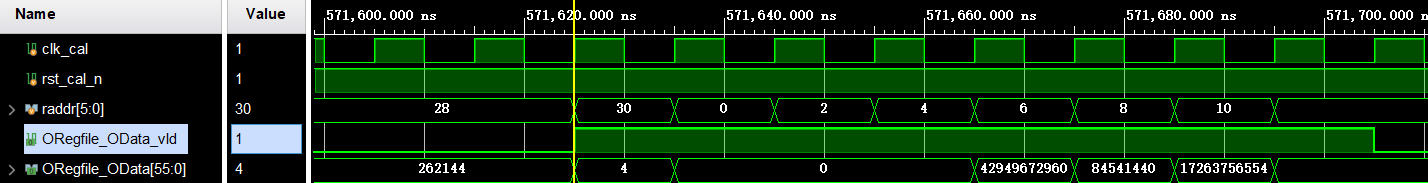
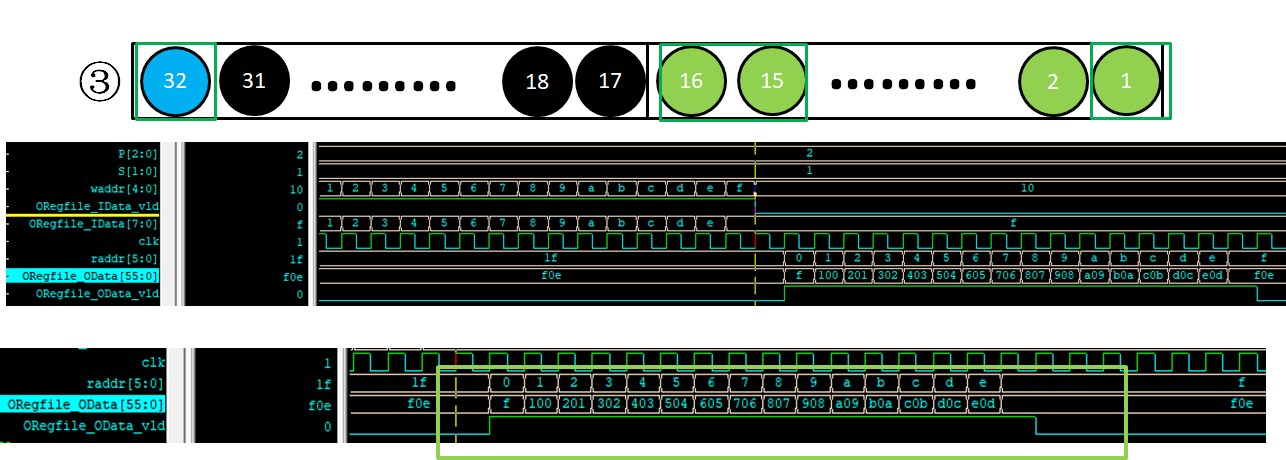
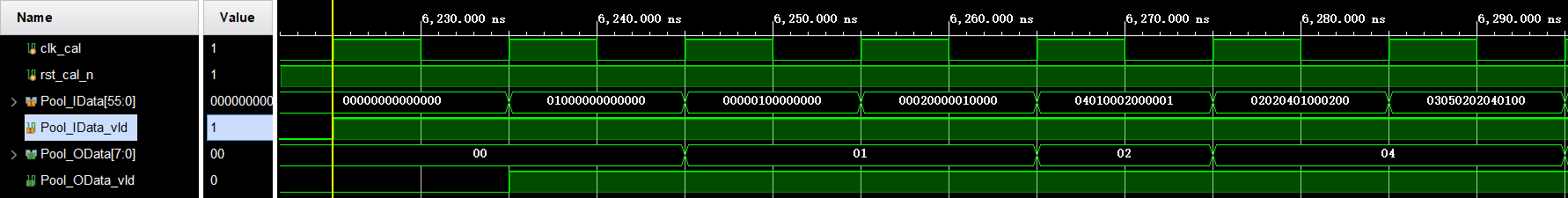


图 3-17 Output\_regfile模块读出测试结果图

如绿框所示，地址从30（蓝圈32）开始读，到地址10（绿圈15）结束，每次读出2个（P个）byte的数据输出给RP，而且数据的间隔为2（S=2），验证正确。

### Pooling模块仿真测试

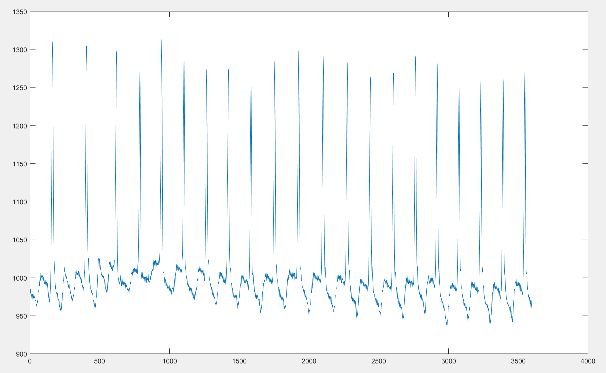
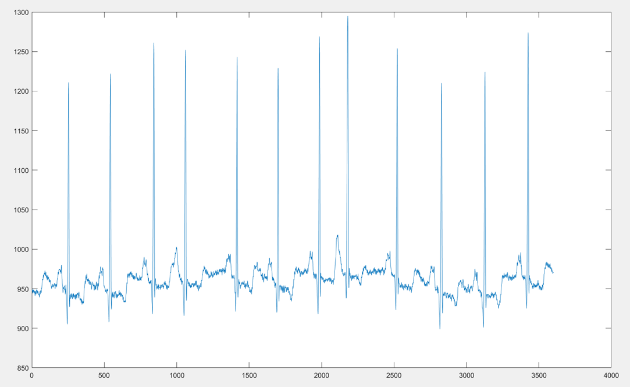
输出P个输入中的最大值。结果如图，正确。

图3-18 Pooling模块测试结果图

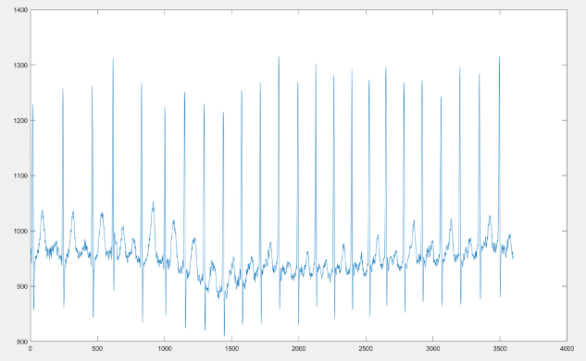
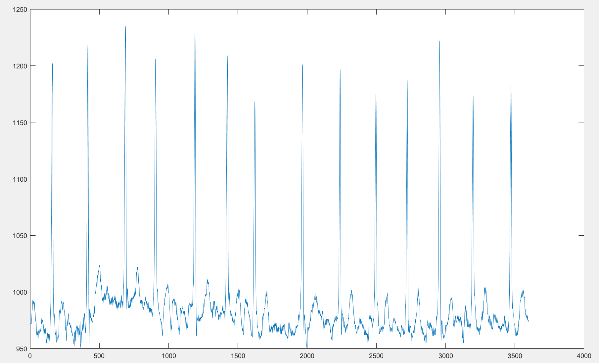
# 准确率测试

## 数据集概况

本网络用于训练和测试的ECG数据来自Physio Net的MIT-BIH Arrhythmia数据集。该数据集共包含48条长度为30分钟的双导联ECG数据，所有的QRS波都有对应的标注，所有的ECG数据经0.1-100Hz的带通滤波器滤波后，在360Hz下进行采样。如图2-1所示，列举了四种来自modified limb lead II具有代表性的心率类型。



(a) (b)

(c) (d)

图4-1（a）Atrial premature beat心率信号（b）Supraventricular tachyarrhythmia心率信号

（c）Atrial flutter心率信号 （d） Atrial fibrillation心率信号

该数据集的特点为：(1)共有1000条数据，每条数据的长度为10秒，不同样本之间互不重叠；(2)信号共来自45名患者：包括19位女性（23-89岁）和26位男性（32-89岁）；(3)信号包含17种类别：正常窦性心律，起搏器节律和15种类型的心律失常，每种类型至少包含10段信号；(4)信号均来自modified limb lead II。心律类型分布如下表所示：

表4-1 心律类型分布

|  |  |  |
| --- | --- | --- |
| No. | Class | No. of Instances |
| 1 | Normal sinus rhythm | 283 |
| 2 | Ventricular tachycardia | 10 |
| 3 | Idioventricular rhythm | 10 |
| 4 | Ventricular flutter | 10 |
| 5 | Fusion of ventricular and normal beat | 11 |
| 6 | Left bundle branch block beat | 103 |
| 7 | Right bundle branch block beat | 62 |
| 8 | Second-degree heart block | 10 |
| 9 | Pacemaker rhythm | 45 |
| 10 | Atrial premature beat | 66 |
| 11 | Atrial flutter | 20 |
| 12 | Atrial fibrillation | 135 |
| 13 | Supraventricular tachyarrhythmia | 13 |
| 14 | Pre-excitation (WPW) | 21 |
| 15 | Premature ventricular contraction | 133 |
| 16 | Ventricular bigeminy | 55 |
| 17 | Ventricular trigeminy | 13 |
|  | Total | 1000 |

在实验阶段，我们使用70%的数据作为训练集训练网络，剩余的30%的数据作为测试集，共304个，来对训练后的网络性能进行测试。考虑到数据集中不同类型的样本数量差异巨大可能会导致训练效果不佳，我们对训练集的数据进行了样本均衡，对样本数较少的种类进行过采样，使不同类型样本数量趋近一致。

图4-2 数据集与测试集数量分布情况

## 算法准确率

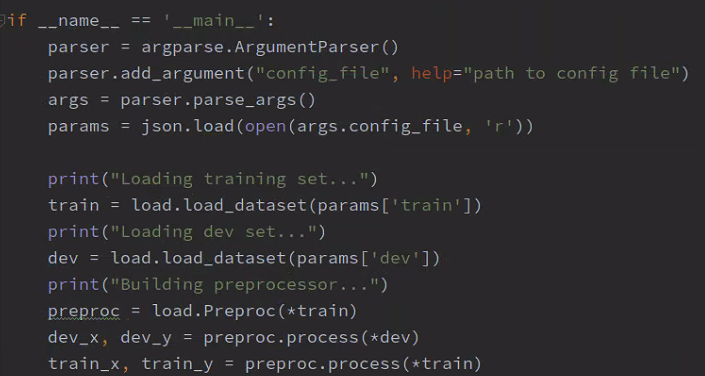
如图4-3所示，算法准确率可以通过keras内置函数得到。

图4-3 Keras内置函数算法准确率

利用Keras模型内置函数可得：

精度acc

**acc = = 94.4%**

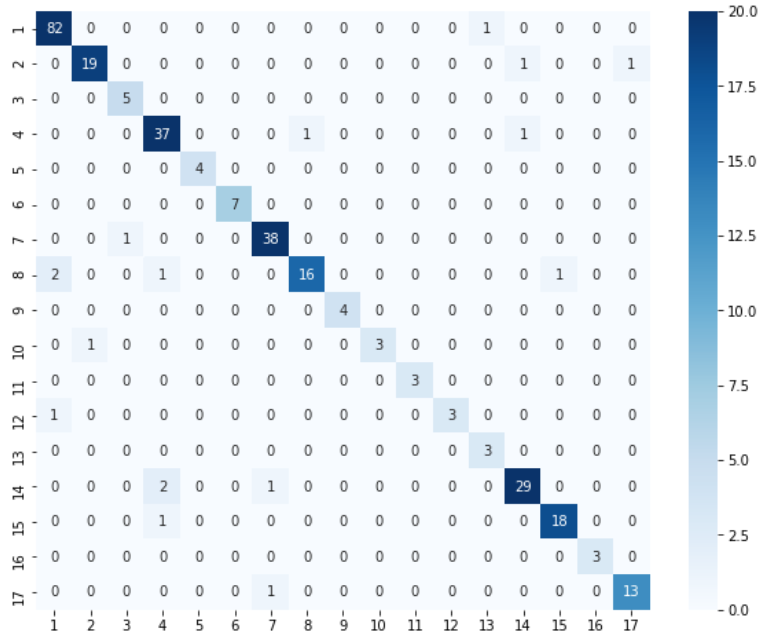
还有一个指标召回率recall（每一类）

**recall =**

最终可得该模型在测试集上的混淆矩阵。图4-4所示。

召回率在前三且样本数量不低于20的三类分别为1 NSR，7 PVC，14 LBBBB

召回率分别为96.5%，95%， 93.5%。

图4-4 混淆矩阵

## 加速器仿真准确率

### 测试方案

加速器精度应该定义为

**ACC =**

**=**

将定义为一致率CR

**ACC =**

此时，选取召回率recall前3的类别，每个类别选取数据集总的3%，一共选15~20个。

由于，这五类召回率高，可以近似认为我们选取的3%的数据，都被算法预测正确。

所以CR可以近似为：

**CR =**

则

**ACC =**

### 实验结果

Recall召回率前三的分别为1 NSR 7 PVC 14 LBBBB

表4-2 测试表格

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 数据集总数 | 测试集数量 | 加速器测试数量 | 仿真测试正确数量 |
| 1 NSR | 283 | 85 | 10 | 9 |
| 2 APB | 66 | 20 |  |  |
| 3 AFL | 20 | 6 |  |  |
| 4 AFIB | 135 | 41 |  |  |
| 5 SVTA | 13 | 4 |  |  |
| 6 WPW | 21 | 7 |  |  |
| 7 PVC | 133 | 40 | 5 | 5 |
| 8 Bigeminy | 55 | 17 |  |  |
| 9 Trigeminy | 13 | 4 |  |  |
| 10 VT | 10 | 3 |  |  |
| 11 IVR | 10 | 3 |  |  |
| 12 VFL | 10 | 3 |  |  |
| 13 Fusion | 11 | 4 |  |  |
| 14 LBBBB | 103 | 31 | 5 | 5 |
| 15 RBBBB | 62 | 19 |  |  |
| 16 SDHB | 10 | 3 |  |  |
| 17 PR | 45 | 14 |  |  |

由上述公式可得，

**CR =**

则

**ACC =**