Lab 7 report 综合设计-手写数字识别

PB21030838 罗浩铭、PB21111691 汤皓宇、PB21051032 许浩峰

实验目的与内容

此次实验的主要内容为：基于Verilog实现一个可以在FPGA开发板上运行的卷积神经网络手写数字识别电路，可以对画图模块写入的数字笔迹进行识别，输出相匹配的数字。

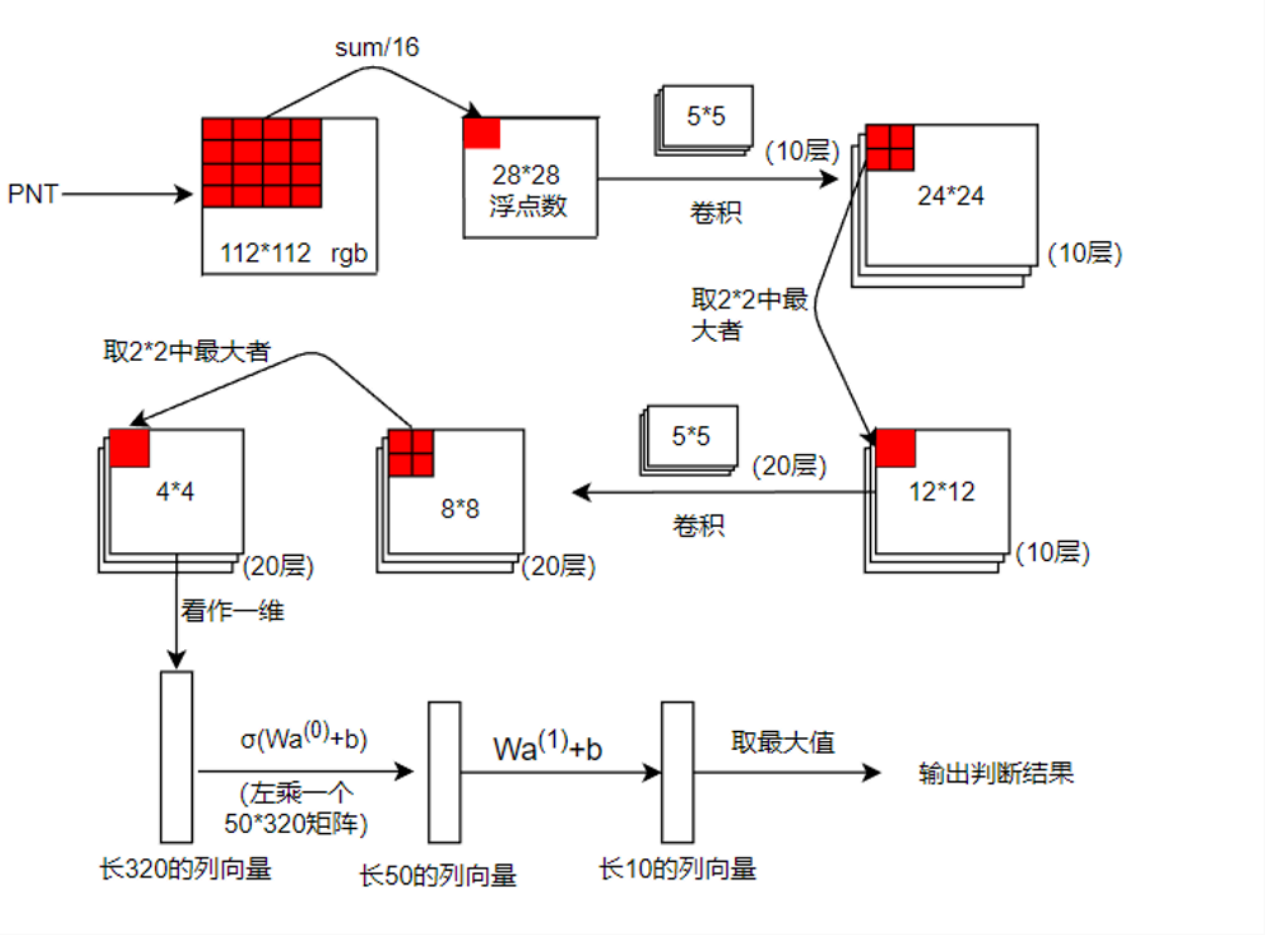
实验目的有：

- 实现一个Verilog大工程，综合锻炼提高我们的Verilog编程能力以及工程能力，提升我们的团队合作素养
- 深入理解神经网络的原理及其底层实现，体会硬件与人工智能结合的魅力
- 深化分模块和分层次的逻辑电路设计思维

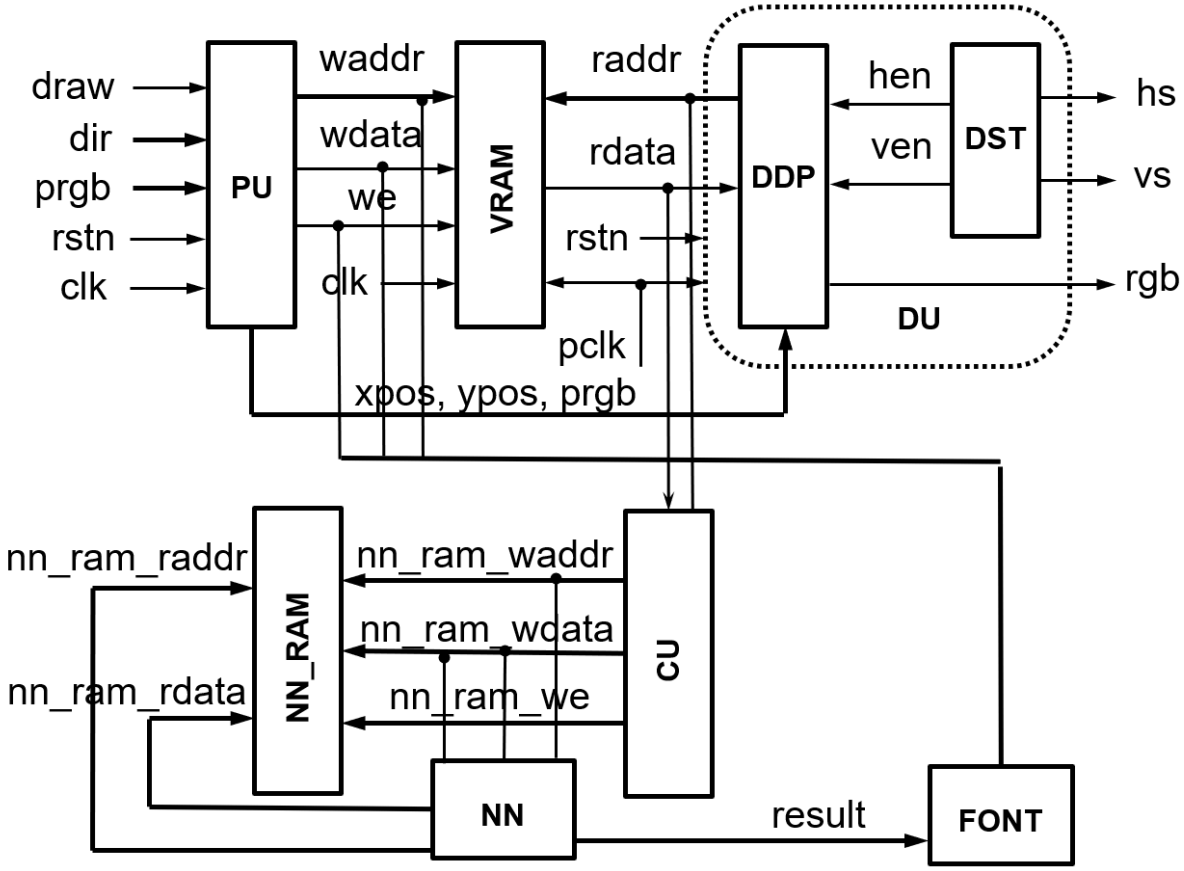
逻辑设计

总设计

电路使用事先训练好并由coe设定的神经网络参数，输入通过归一化、卷积、池化、矩阵乘法、偏置、ReLU、Argmax等计算，最后得到判断的结果。
我们实现的神经网络结构如图(`nn.v`)：



总的电路的数据通路如图：（为了数据通路图的简单，已省略许多控制信号，存储器涉及两个不同的地址输入、写入数据、写使能的，由选择器控制）



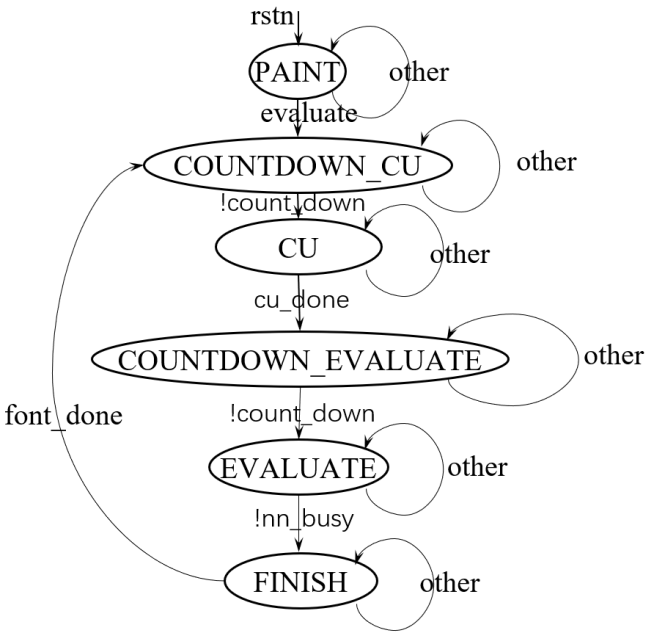
各模块设计

mnist总模块

其状态图如图所示：

状态图

- PAINT: 画图模式
- COUNTDOWN_CU: cu_st
- CU: 读入所写数字
- COUNTDOWN_EVALUATE: nn_st
- EVALUATE:神经网络运算
- FINISH:字库显示



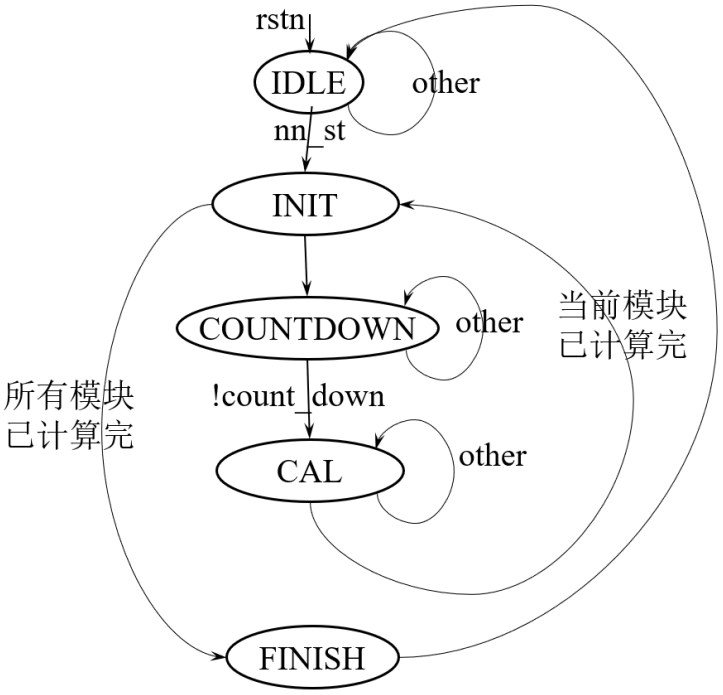
总模块平时处于正常画图状态，一旦按下evaluate按钮，它将先读入所写数字至神经网络输入，然后开始神经网络运算，最后经由字库在屏幕上输出最终结果。
其实现中使用选择器来使得不同模块可以交替与RAM交互

神经网络总模块

其状态图如图所示：

状态图

- IDLE:
- INIT:开始下一个模块计算
- COUNTDOWN:给出st
- CAL:模块计算中
- FINISH:完成计算



开始神经网络运算后，由一个计数器记录当前在计算第几个模块，直到所有模块计算完毕，最后输出结果。
此次使用了block memory来存储神经网络中的所有参数、输入、中间结果及输出，每个张量在内存中都有一个offset值，由相对位置加上offset值可以得到其在内存中的绝对位置。
以下为神经网络中所有offset的值:

```
源数据：784    offset:0

分配：10个卷积核，10*(1*5*5)=250      offset:784
得到10*24*24=5760      offset:1034
max pooling out:10*12*12=1440      offset:6794

20个卷积核，20*(10*5*5)=5000      offset:8234
得到20*8*8=1280      offset:13234
max pooling out:20*4*4=320      offset:14514

(reshape:320)
320*50+50+50=16100
offset:14834
bias(50) offset:30834
output(50) offset:30884

50*10+10+10=520
offset:30934
bias(10) offset:31434
output(10) offset:31444
```

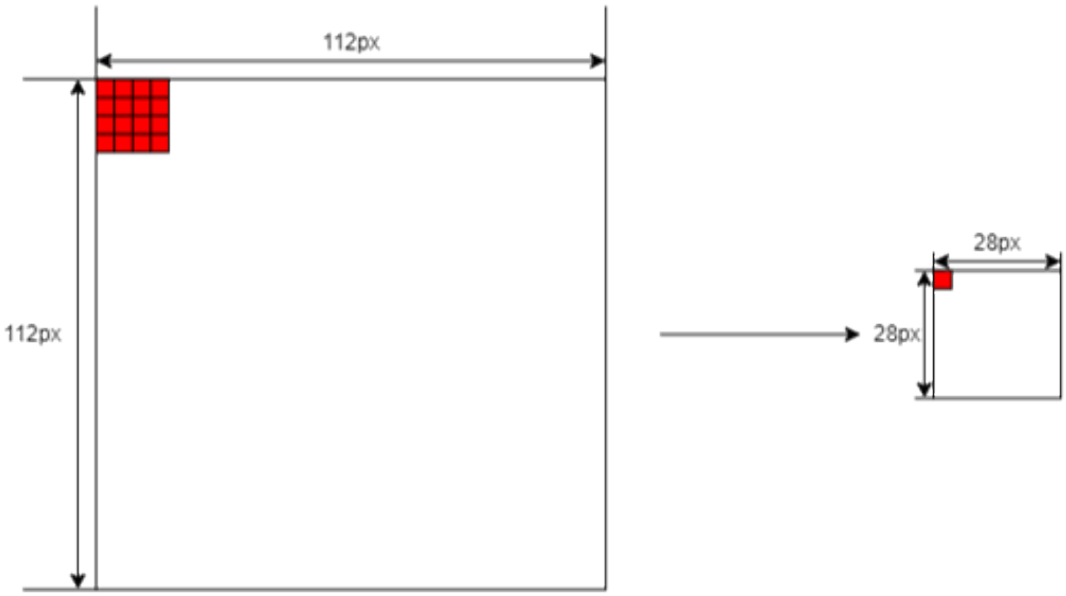
其实现中使用选择器来使得不同模块的地址、写入数据、写使能可以交替与RAM交互

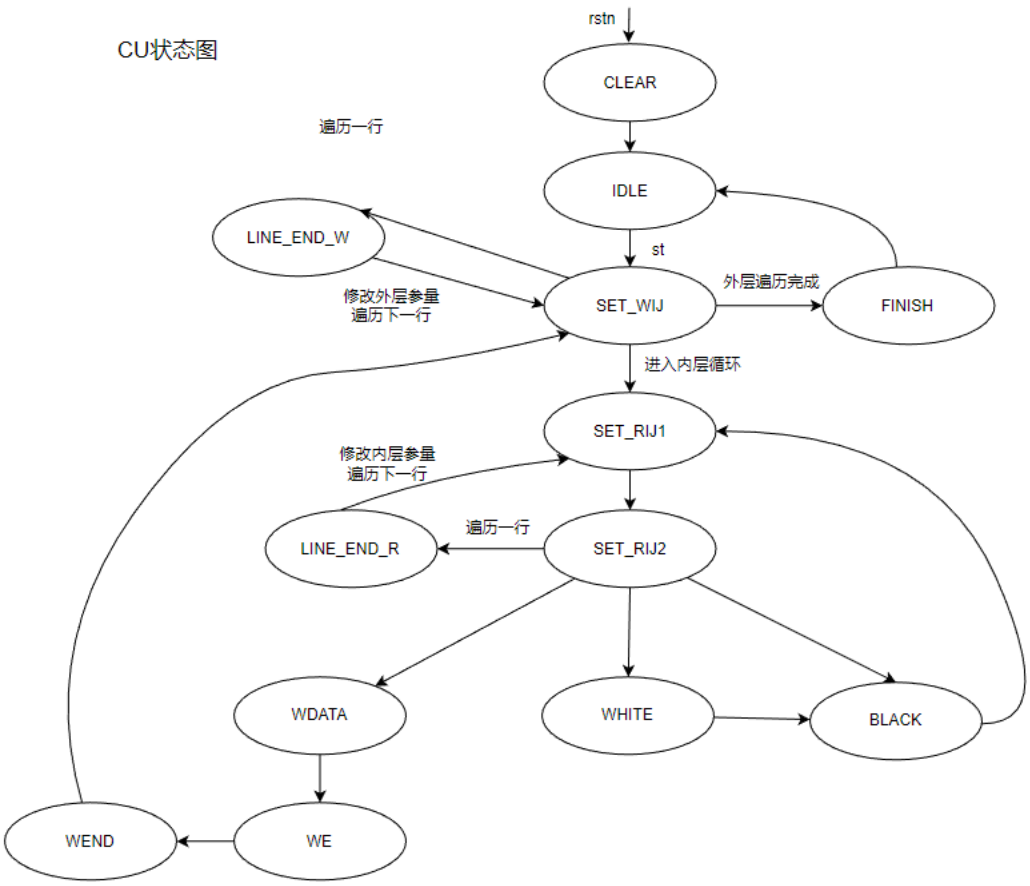
神经网络计算模块

以下运算全部涉及浮点数，需要与浮点数IP核交互，所以逻辑较复杂。
这些模块实现均为为一个二重计数器或多重计数器控制地址，使得内存中的数读入模块并在浮点数计算模块中计算，最终得到结果按照写入地址写回内存。

CU模块

对画图模块的数据样点通过Average Pooling，即局部（在这里是`4*4`正方形内）取平均，得到28*28的标准输入

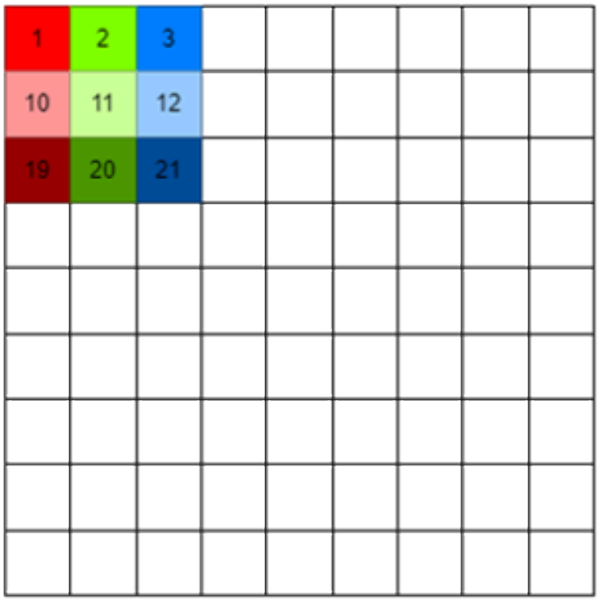


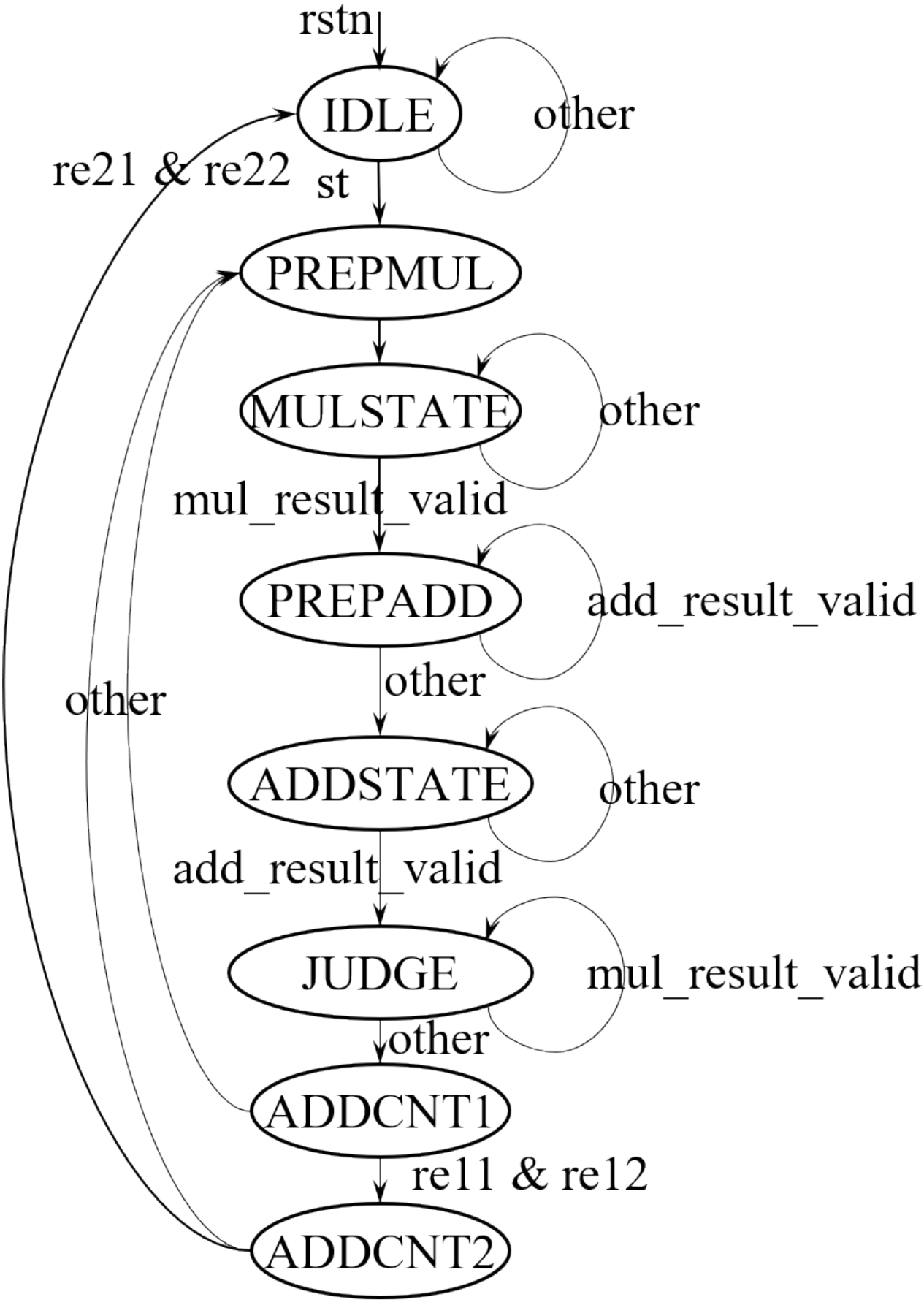


卷积

使用多层卷积提取局部特征，提高识别准确性。其实现即为一个滑窗在图片上滑动，滑窗每到一个位置，就进行对位乘法（如图中颜色一样的位置相乘，再将所有积相加），得到该位

置的结果。滑窗特征与图上对应位置特征越匹配，结果值越大。滑窗滑遍整张图，则得到整张图的特征图。若输入通道不为1（输入图有数层），则卷积核需要有数层（三维对位乘法）；若输出通道不为1，则需要多个不同卷积核输出多张特征图。最终需用到六层计数器，这也使得其成为最复杂的模块。



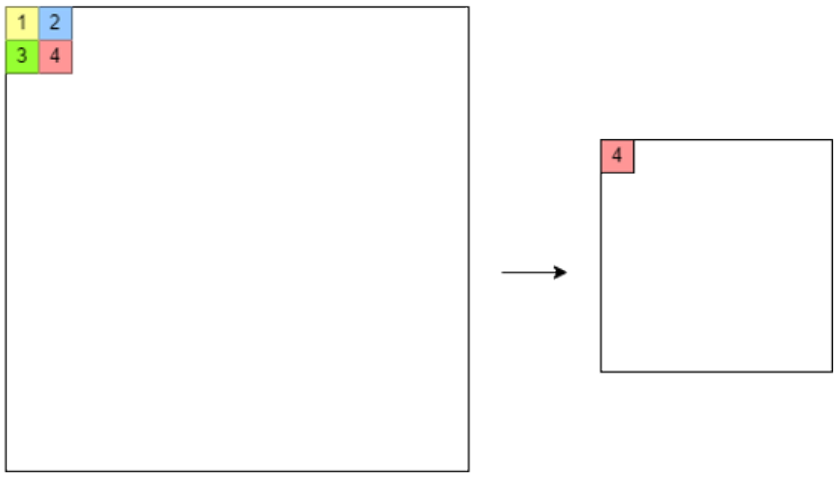

rstn
IDLE
other
re21 & re22
st
PREPMUL
MULSTATE
other
mul_result_valid
PREPADD
add_result_valid
other
other
ADDSTATE
other
add_result_valid
JUDGE
mul_result_valid
other
ADDCNT1
re11 & re12
ADDCNT2

- IDLE: reset

- PREPMUL: mul_a_data, mul_b_data
- MULSTATE: mul_a_valid, mul_b_valid
- PREPADD: add_a_data, add_b_data, add_rstn, mul_result_ready
- ADDSTATE: add_a_valid, add_b_valid
- JUDGE: add_result_ready, mul_rstn
- ADDCNT1: ce1
- ADDCNT2: ce2

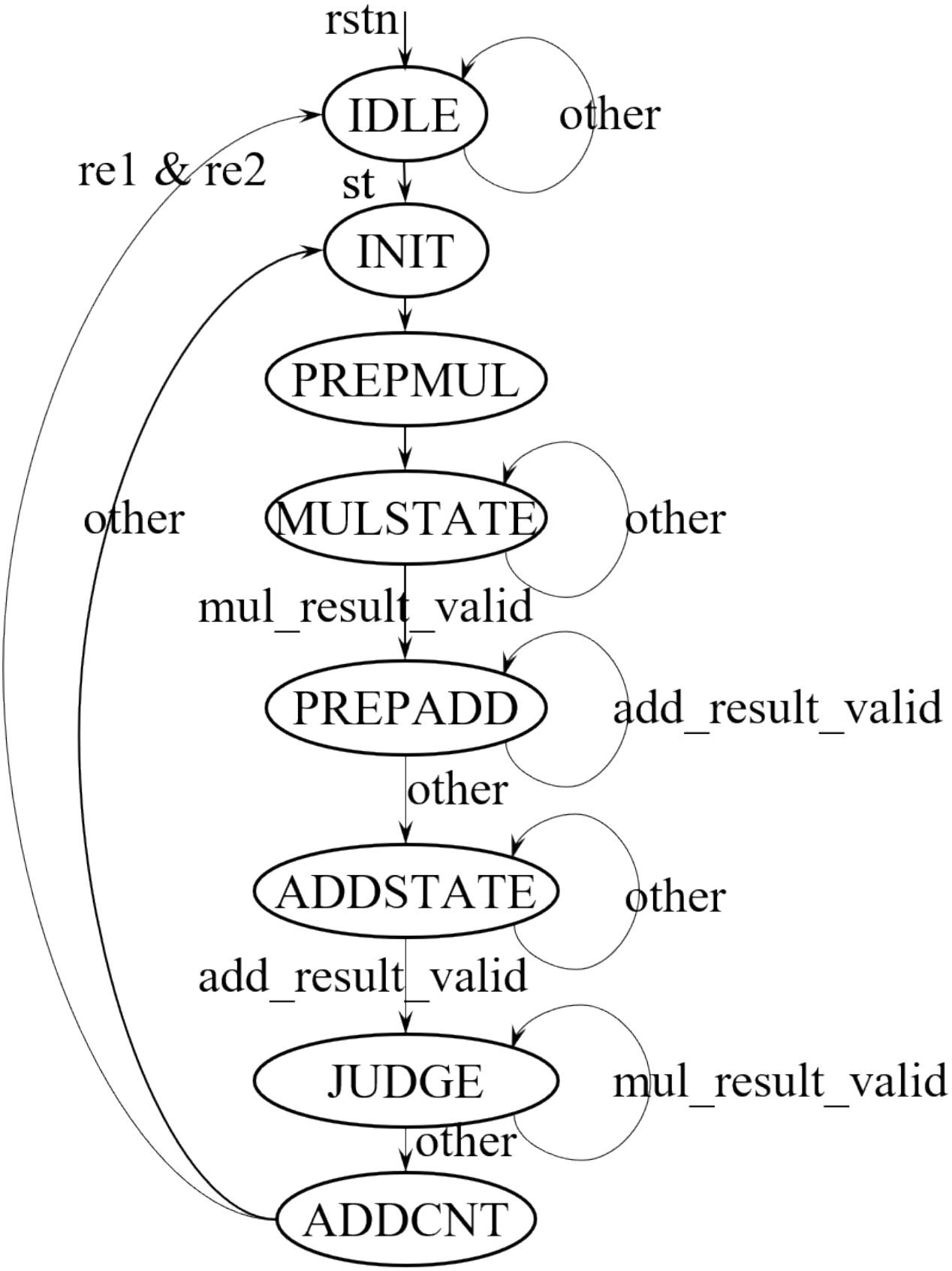
Max Pooling

使用Max Pooling，即局部（这里是`2*2`正方形内）取max的方式合理降低神经网络计算量



> 神经网络最终经过两层全连接层（含矩阵乘法、向量加法、ReLU函数）得到一个长为10的向量，其中数字越大代表其下标是正确结果的可能性越大，最终判断结果即为此数

矩阵乘法



其中，有如下一些状态：

- IDLE: reset
- INIT
- PREPMUL: mul_a_data, mul_b_data
- MULSTATE: mul_a_valid, mul_b_valid
- PREPADD: add_a_data, add_b_data, add_rstn, mul_result_ready
- ADDSTATE: add_a_valid, add_b_valid
- JUDGE: add_result_ready, mul_rstn
- ADDCNT: ce

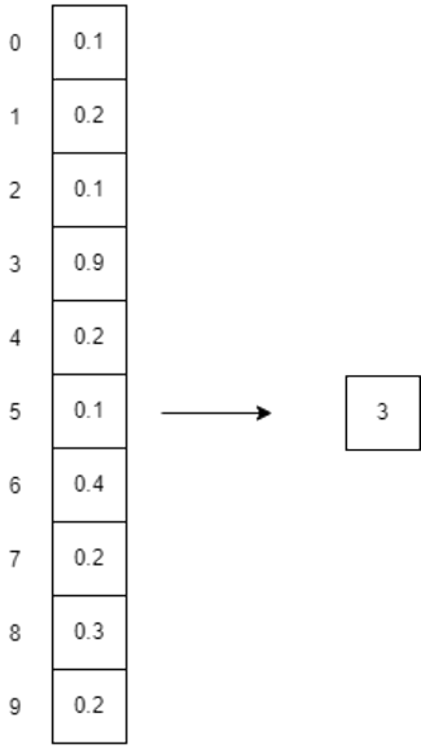
具体代码见附件`matmul.v`

bias（向量加法）

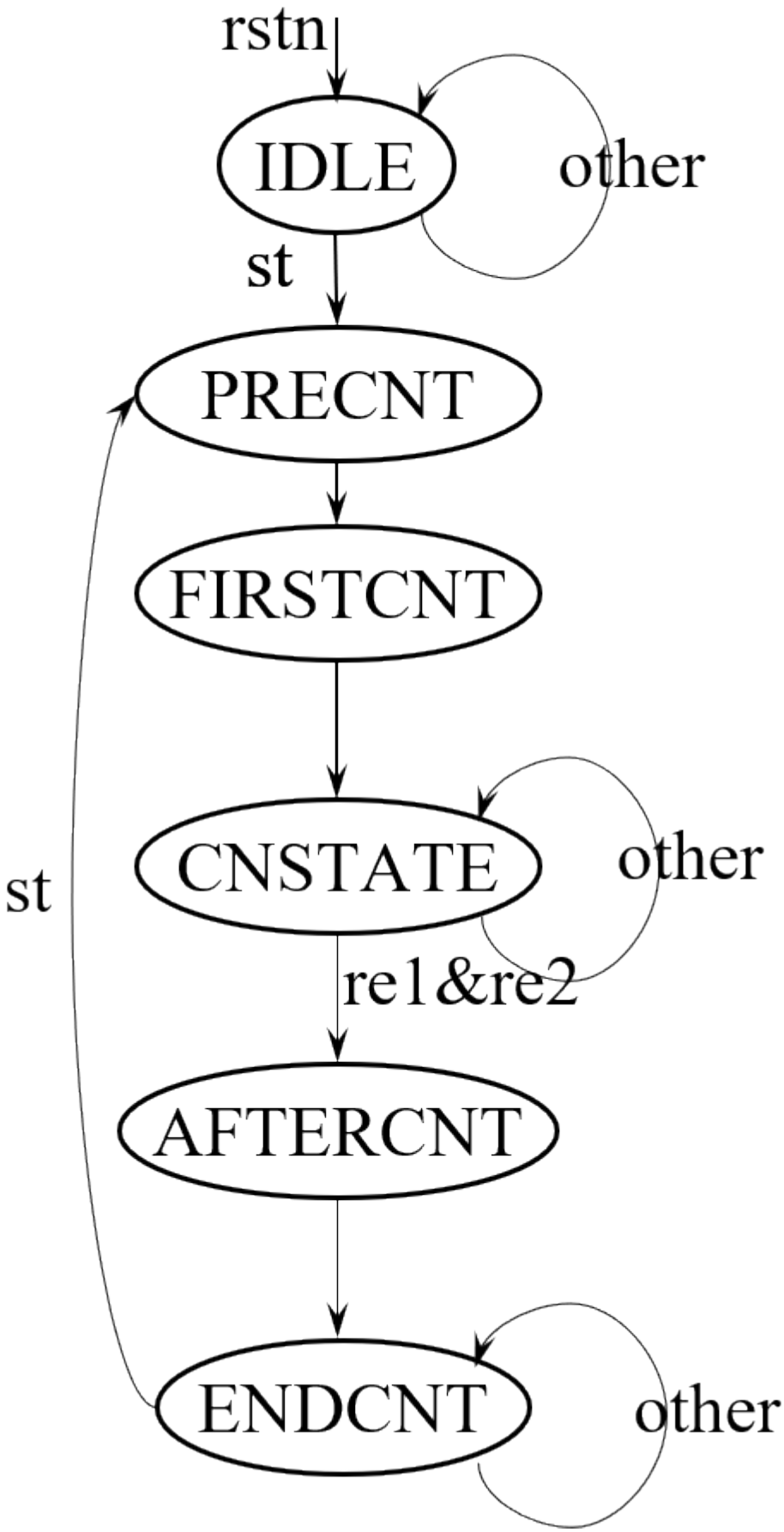
获得两向量的和

argmax

获得向量中最大的数的下标



字库


rstn
IDLE
other
st
PRECNT
FIRSTCNT
CNSTATE
other
st
re1&re2
AFTERCNT
ENDCNT
other

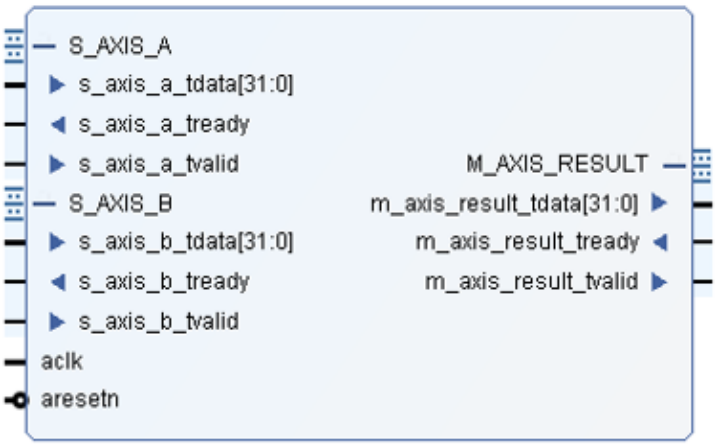
- IDLE：reset
- PRECNT：reset
- FIRSTCNT：ce
- CNTSTATE：ce，we
- AFTERSTATE：we
- ENDCNT：done

pen_point模块

用一个二重循环计数器遍历围绕笔尖的9*9正方形区域内像素，使得画笔粗细为9个像素

IP核模块

此神经网络需要用到3万个以上的浮点数，故使用了一个大的block memory来存储神经网络中的所有参数、输入、中间结果及输出，并且事先确定好各个矩阵、向量在该内存中的位置(即确定它们的offset)。在总状态机的协调下，各个子模块与这个存储空间进行交互即可。神经网络参数由coe文件写入。
浮点数加法及乘法运算同样由IP核实现。



仿真结果与分析

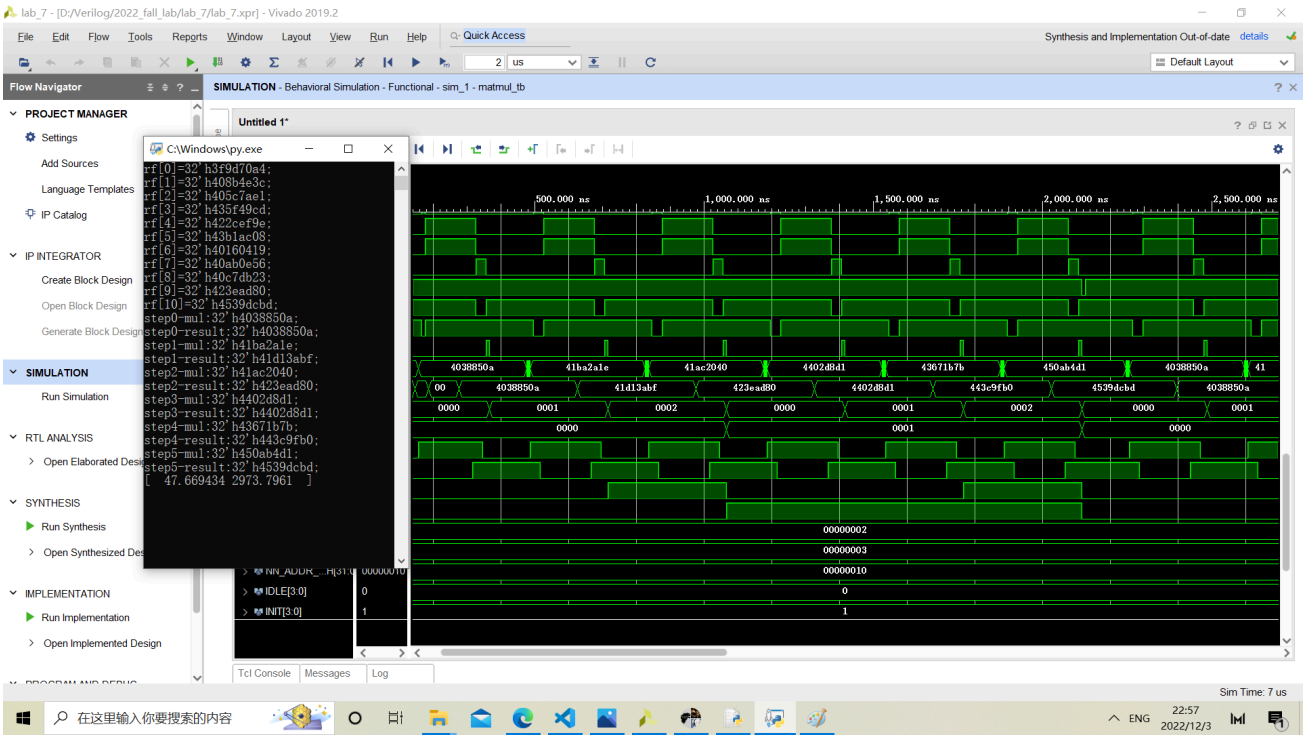
在仿真后，我们可以发现电路正确地输出了我们所期待的波形。

神经网络计算模块

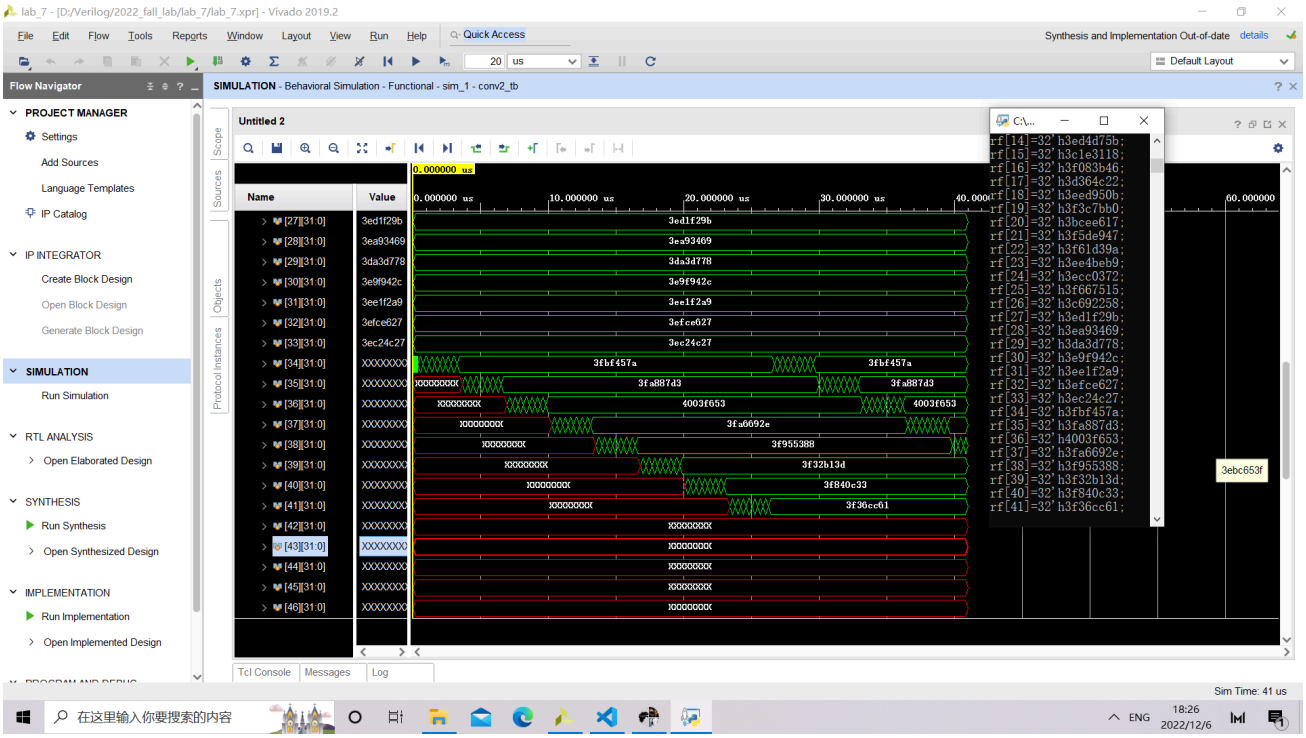
对神经网络所有计算模块的仿真都通过与python对拍来debug，且由此验证：最终所有模块输出结果完全正确（样本一开始为随机生成，此后固定）。
tb中用一个寄存器堆来模拟RAM的行为：

```
reg [31:0] rf [0:15];
always@(posedge clk)
begin
    if(we)
        rf[waddr]<=wdata;
    rdata<=rf[raddr];
end
```

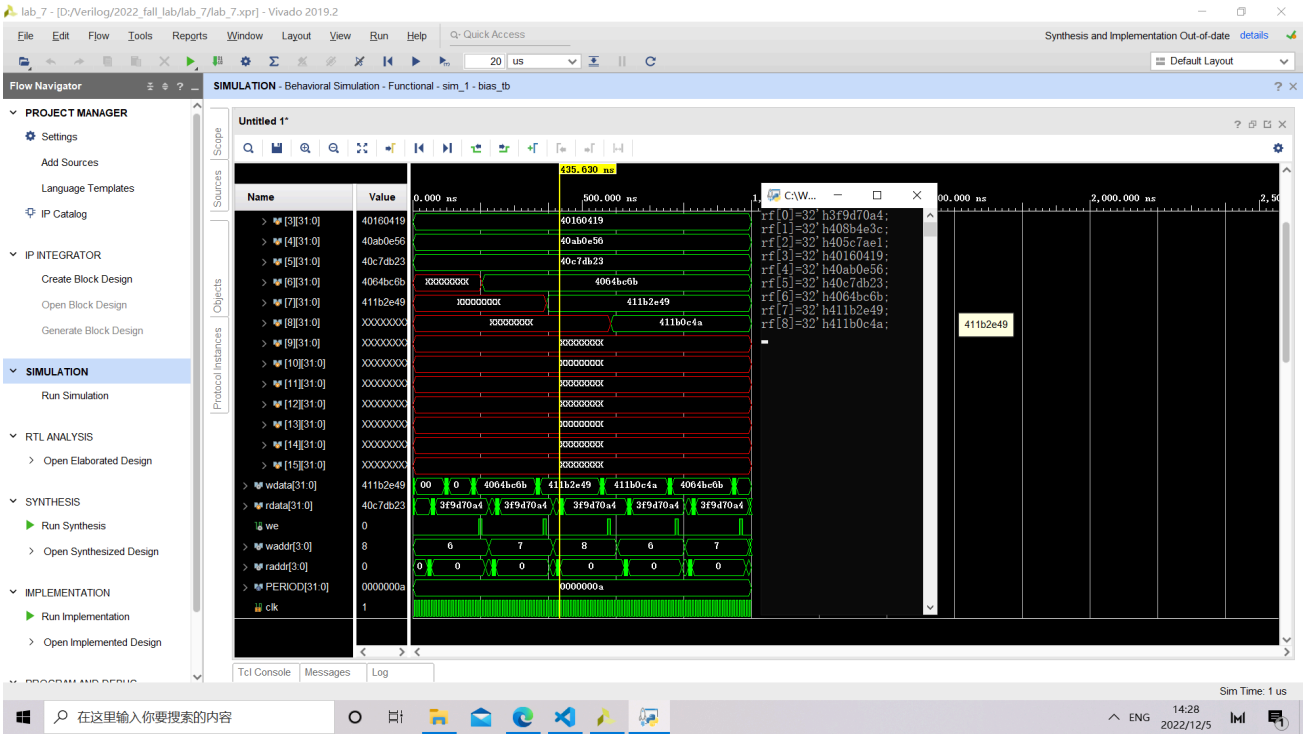
矩阵乘法



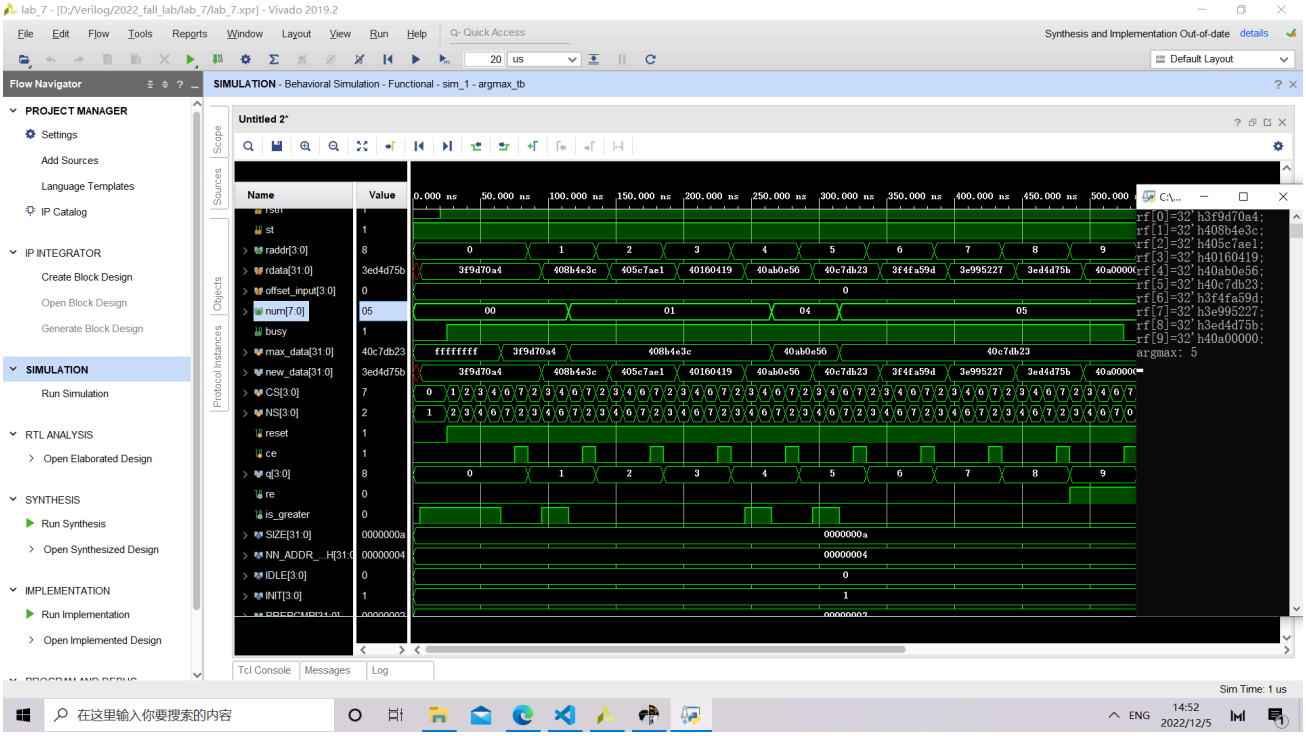
卷积


lab_7 - [D:/Verilog/2022_fall_lab/lab_7/lab_7.xpr] - Vivado 2019.2
SIMULATION - Behavioral Simulation - Functional - sim_1 - conv2_tb
Untitled 2
Sim Time: 41 us

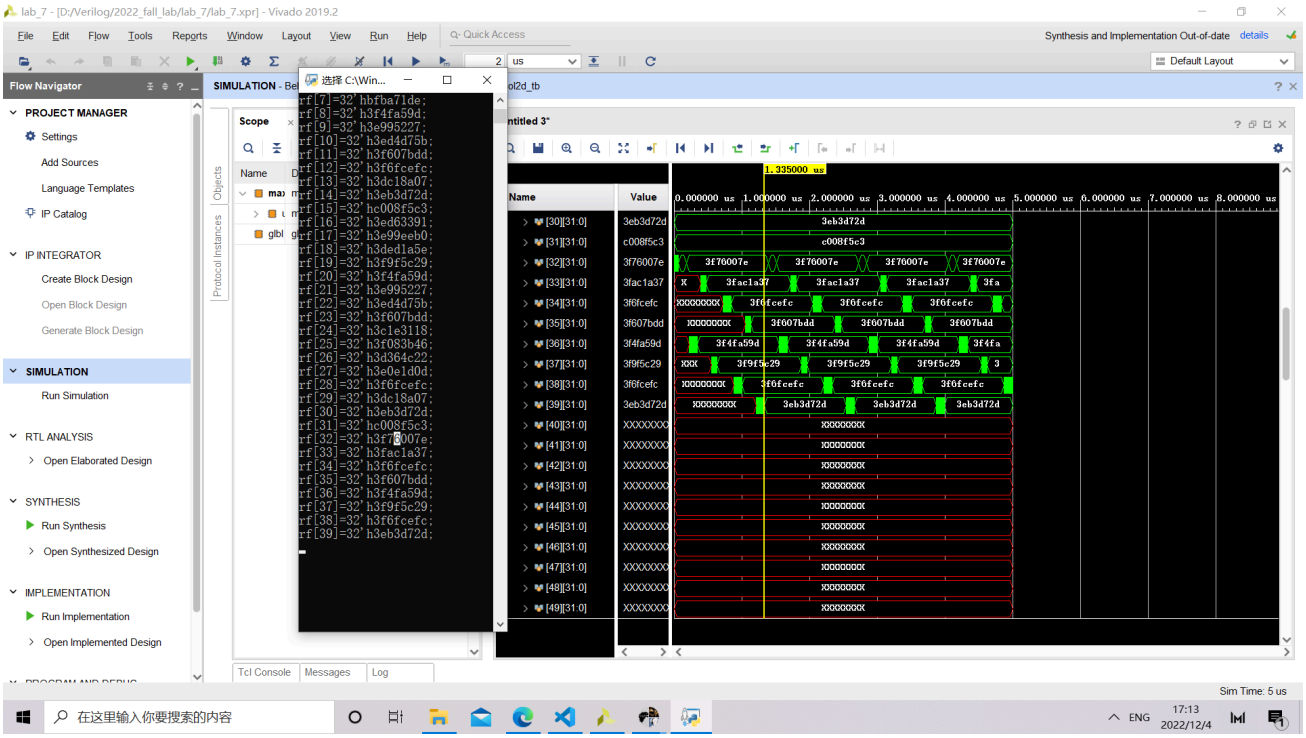
bias


lab_7 - [D:/Verilog/2022_fall_lab/lab_7/lab_7.xpr] - Vivado 2019.2
SIMULATION - Behavioral Simulation - Functional - sim_1 - bias_tb
Untitled 1*
Sim Time: 1 us

argmax

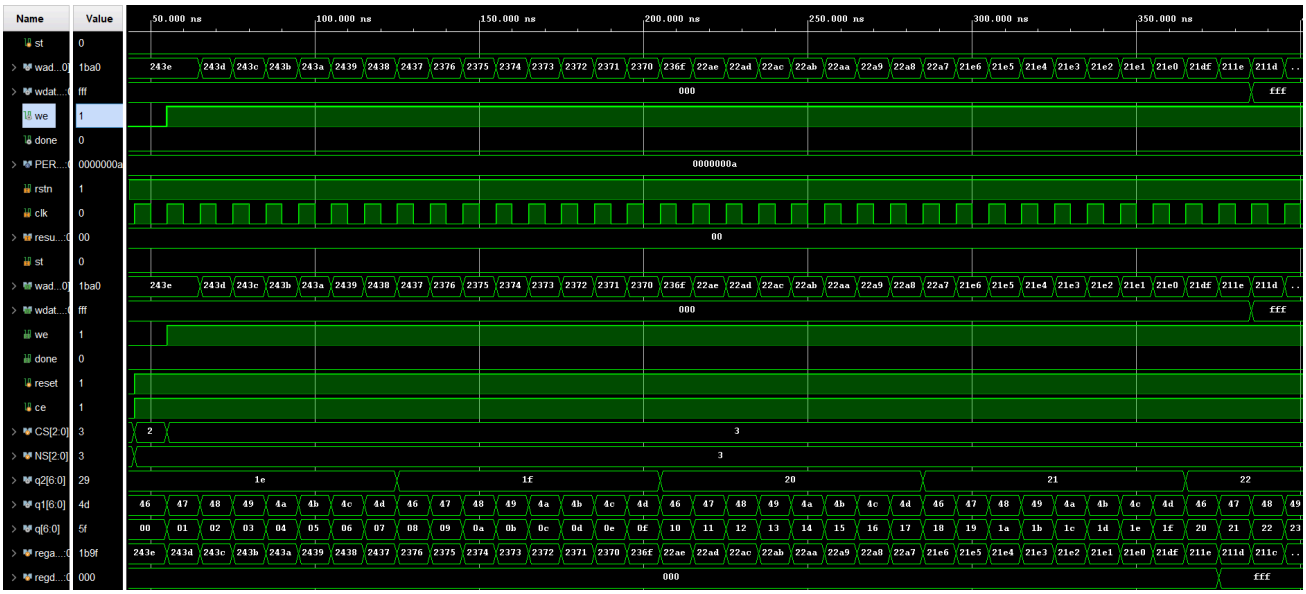


maxpool2d



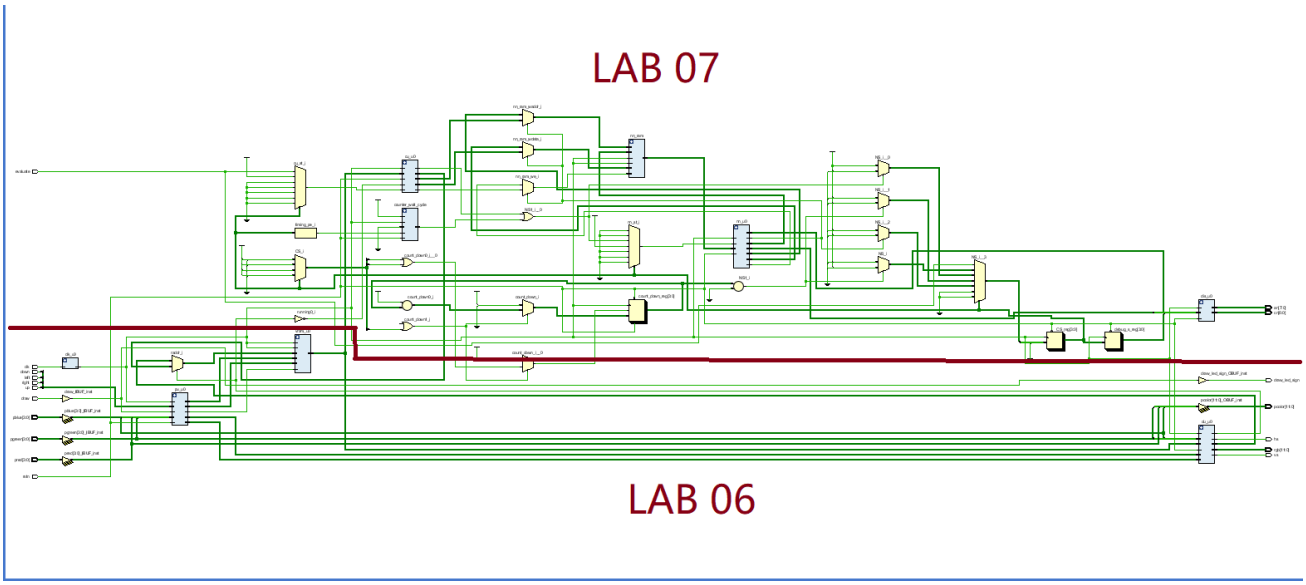
字库

对字库作仿真，因为从低位到高位的字符对应的地址是与实际相反的，需要对addr作对应的变换，且验证we对应的值，得到如下仿真

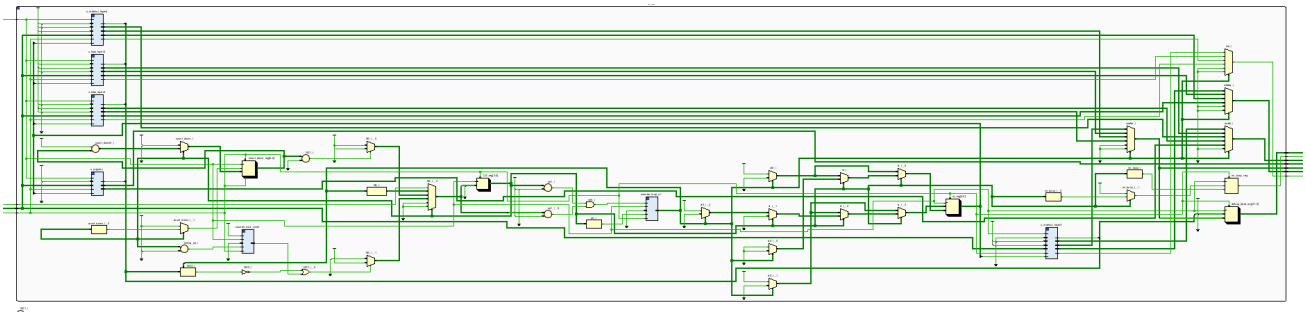


电路设计与分析

综合电路的RTL电路图如下，基本与设计图里的模块与数据通路相符，其RTL图可以视作在lab6的基础上，加了一大块非常复杂的神经网络计算通路，从VRAM出，最后在VRAM回：

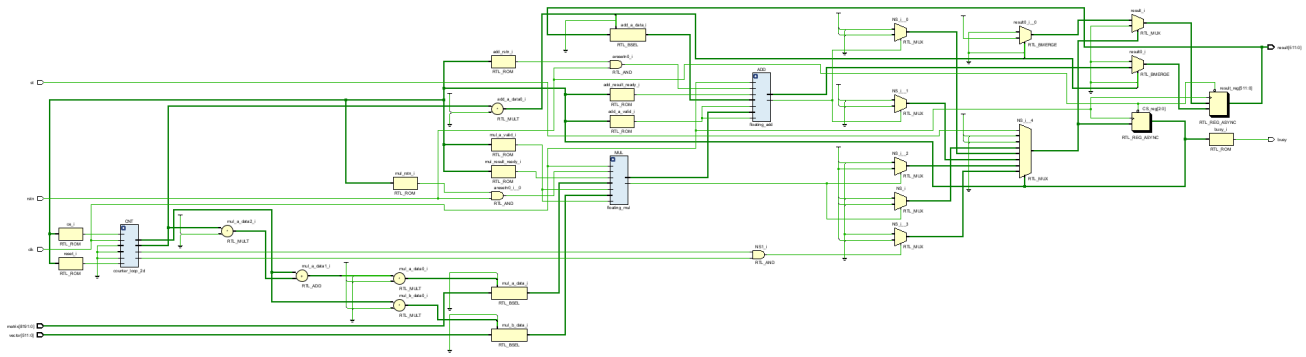


其中神经网络模块的RTL电路图如下，其中含数个神经网络计算模块以及大量简单逻辑电路用于控制这些计算模块：

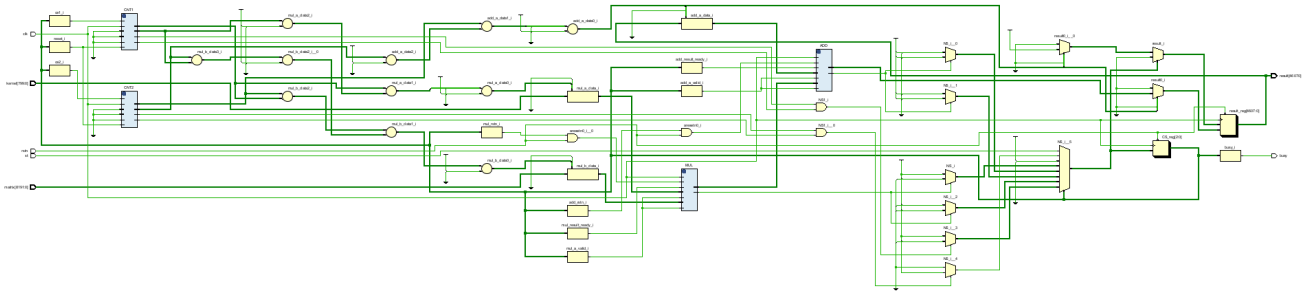


神经网络计算模块的RTL图均为二重计数器或多重计数器控制下的地址电路以及浮点数计算模块，以下仅举矩阵乘法和卷积两个例子：

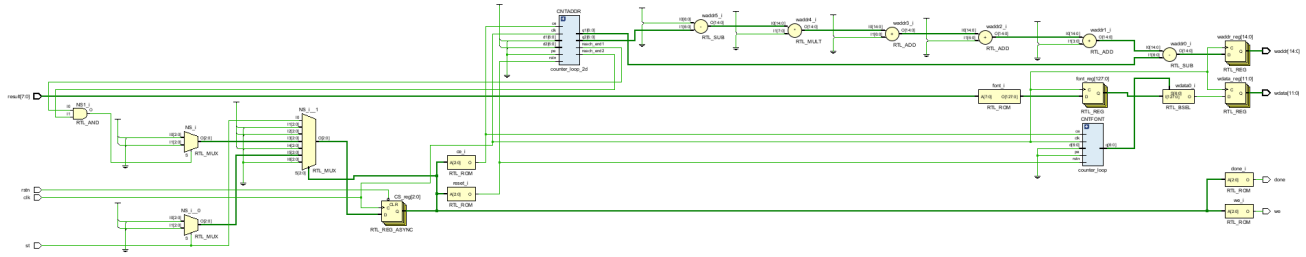
矩阵乘法的RTL电路图如下：



卷积的RTL电路图如下：

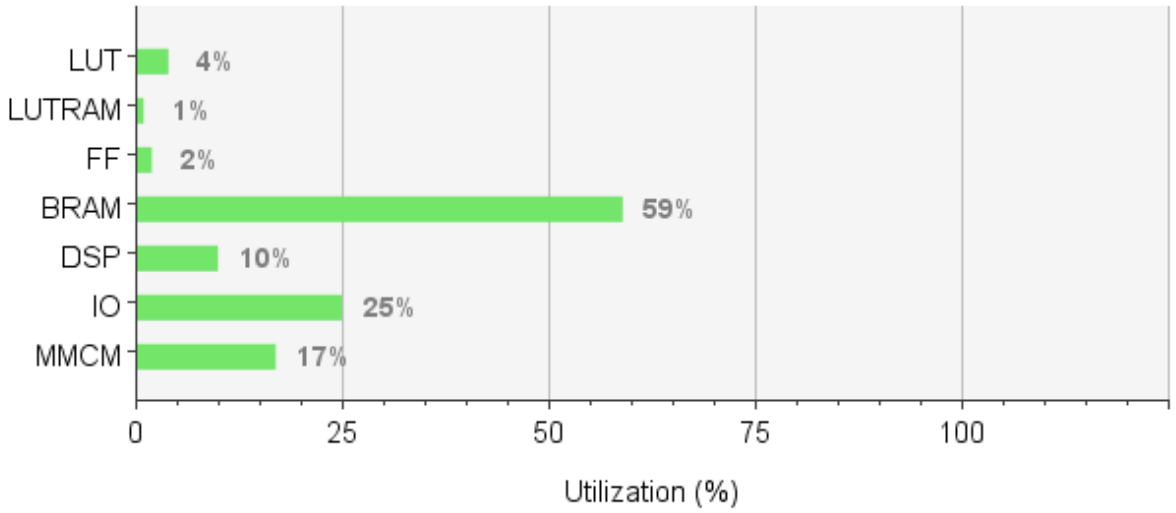


字库的RTL电路图如下：



电路的资源使用情况及WNS如下图:

Resource	Utilization	Available	Utilization %
LUT	2248	63400	3.55
LUTRAM	29	19000	0.15
FF	2827	126800	2.23
BRAM	80	135	59.26
DSP	23	240	9.58
IO	52	210	24.76
MMCM	1	6	16.67



Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	7.299 ns	Worst Hold Slack (WHS):	0.028 ns	Worst Pulse Width Slack (WPWS):	3.000 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	7602	Total Number of Endpoints:	7602	Total Number of Endpoints:	3053

All user specified timing constraints are met.

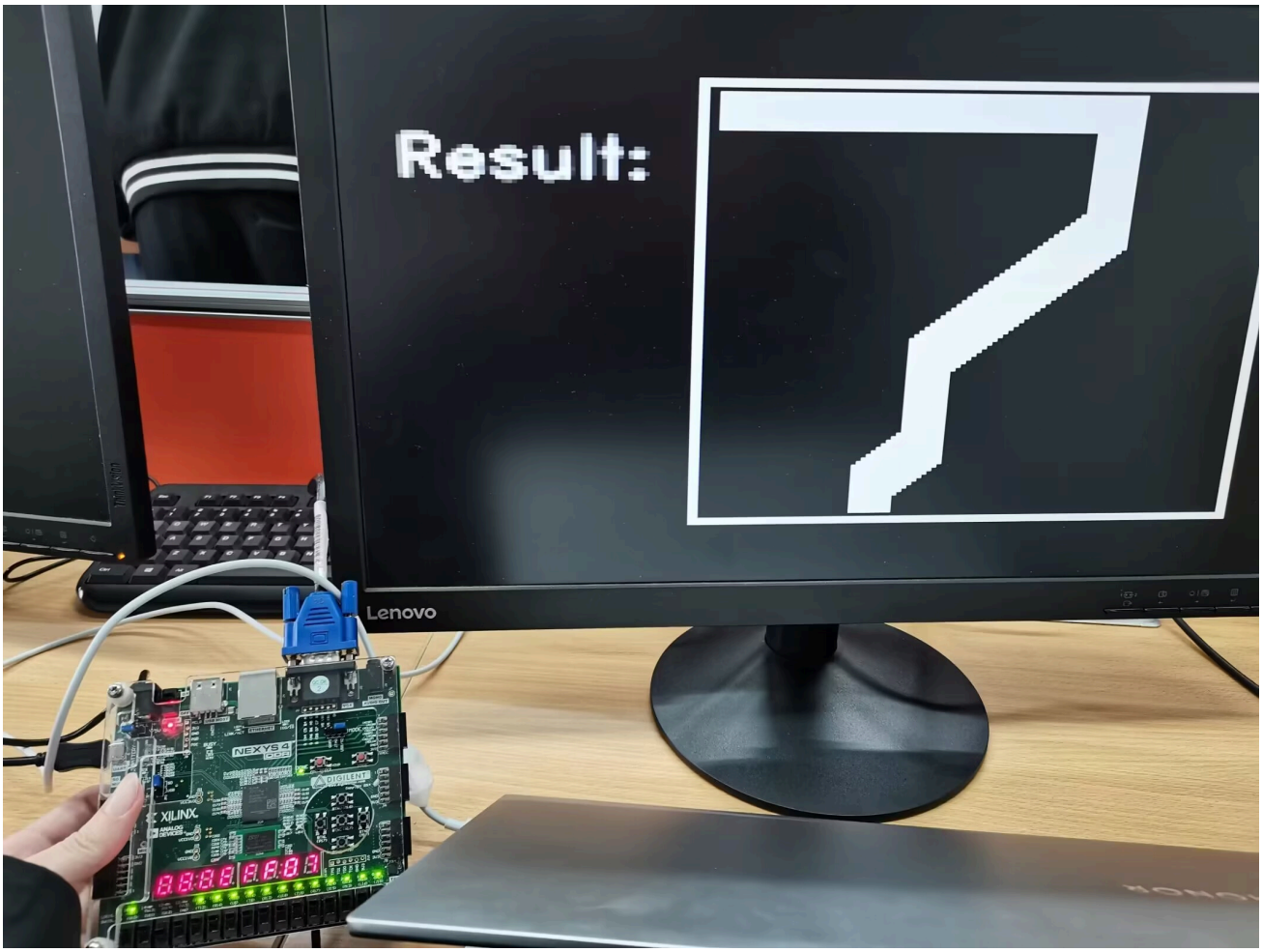
WNS非负，电路可以正常运行(时钟全部使用周期20ms的pclk)。
电路使用了2248个LUT(查找表)、29个LUTRAM(查找表RAM)和2827个FF(触发器)，使用量大部分由浮点数计算模块贡献；使用了80个BRAM(Block RAM)，因为使用了大量存储资源来存放VGA显示内容、默认背景以及神经网络，占总资源59.26%；23个DSP（数字信号处理模块）；1个MMCM（混合模式时钟管理器）。

测试结果与分析

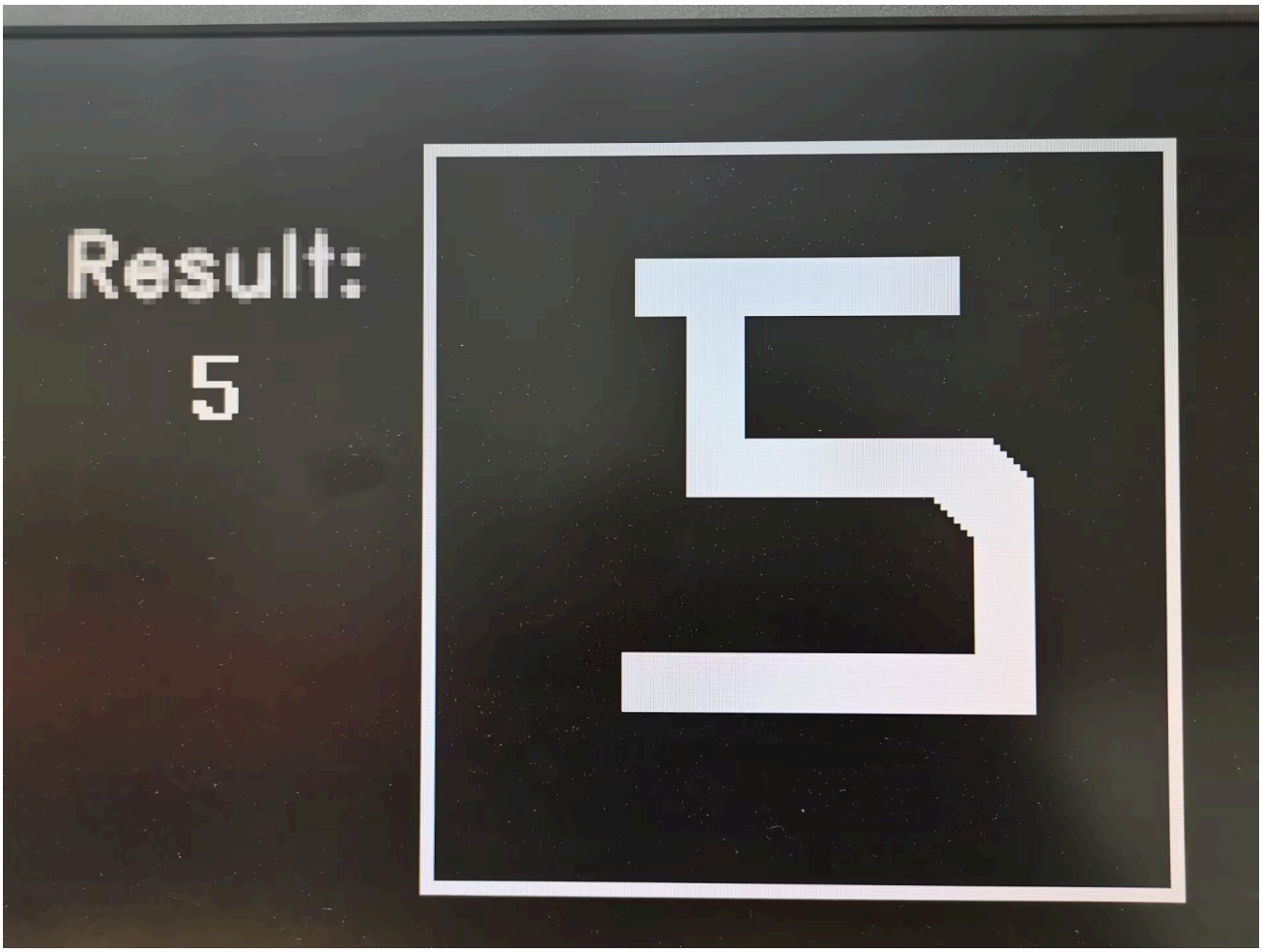
正如答辩视频里所展示的那样，我们的手写数字识别电路可以根据输入笔迹得到正确的数字

识别结果。

答辩时用的`mnist v9 no font.bit`没有加入字库，识别结果正确且展示在数码管的最右端
其操作说明如下：
识别结果展示在数码管的最右端
SW[15]为draw使能,SW[11:0]为RGB值（只有SW[11],SW[7],SW[3]均为1，即RGB值均不小于128，当前色块才会被判定为白色，否则均为黑色）
中心按钮btnc为令神经网络开始运算的信号，上下左右键为移动画笔
led16_r亮时神经网络处于运算状态



`mnist v11 font.bit`加入了字库，但识别结果不准确
操作说明如下：
识别结果展示在数码管的最右端与屏幕上，数码管前四个数分别展示画笔x,y坐标
SW[15]为draw使能,SW[0]为1时画笔为白色，0时为黑色
中心按钮btnc为令神经网络开始运算的信号，上下左右键为移动画笔
led16_r亮时神经网络处于运算状态



`nn_check.bit` 用于检查神经网络各内存位置的值是否正确（依照 `nn_simple_check.txt` ），经检查发现神经网络计算只有舍入误差。
操作说明如下：
SW[15:0]为内存地址
数码管显示该地址的内存存储的值

总结

罗浩铭

项目总结

这回数电实验大作业我们组做的是手写数字识别。一开始项目进展顺利，在上周日前所有底层模块已经基本完成（全部通过仿真检验，自始至终都非常可靠，正如前面各仿真截图右下角显示的时间），周一晚也完成了它们的拼接。然而，故事才刚刚开始。
拼接后的电路不负众望地不能运行，输出没有任何响应。
问了助教，助教让我们看RTL信息里的warning，结果只发现了两个bug，其它都是没用的warning，说明我们的代码是严丝合缝的，只能是逻辑错误（比如状态机的状态转换逻辑）。

我接着进行了许多尝试让电路输出各中间结果来debug，经一系列检验发现电路运行卡在了中间状态。

时间来到了周三傍晚，答辩在第二天下午2点开始。我把澡洗完、衣服洗完、夜宵买好，做好了鏖战一夜的准备。

我前半夜借助仿真全力检查状态机，终于在凌晨2：30发现神经网络总模块的状态机一个比较明显而严重的逻辑错误。在这之后，所有模块仿真一切正常，但上板子之后却无法工作。

整个后半夜我只确认了几个事实：1.vivado仿真有问题，我们的模块改小参数后跑仿真没问题，上板却会卡死在某个状态，最离谱的是有一个模块状态机NS和下一个CS居然不一致（详见波形图）；2.几乎同样的代码（改了些无关紧要的地方），跑几次综合之后得到的几个bit流写进板子上之后行为不同，比如卡在不同的状态（同一个bit流不同次烧写得到的行为是一致的，说明不确定性不源于电路上）。3.我们的卷积模块行为和PyTorch不同，且不清楚PyTorch的卷积模块的具体运算实现

天亮了，离答辩只剩几小时，我们组却一筹莫展。虽说我的态度一贯是即使做不出来，也要足够悲壮，但在无从下手的情况下，我很想回宿舍补觉。

上午9:30，我突然想到，可以先做出两层全连接层的小破网络，这样既可抛弃掉行为不明的卷积模块，也可简化电路。接着，我又想到可以用计时器，在保证每一个模块有足够时间完成其工作的前提下，强制跳转状态让下一模块运行，不需要等待不知为何不能发挥作用的握手信号（表明模块运行结束的信号），虽然很“工程”（低情商：粗糙），但不管黄毛黑猫还是白猫，能让它跑起来再说。

接下来三个小时（除去吃饭时间）的工作真可谓是奇迹。

我先在半个小时内完成了新神经网络训练、参数导出、写入coe文件、修改电路神经网络结构、给出内存各个位置参数检查表（我们编写了顶层模块可进行这种检查）的全流程。

接着我开始修改状态机，让电路跑通。先是跑通了神经网络，队友对跑完神经网络后的内存参数进行检查，确证了神经网络运算没有问题。接着要想办法让电路输出结果。答辩前一个半小时，电路终于能显示出非零的结果，但只能显示一种结果。

在继续修改状态机无效后，我突然意识到可能是神经网络读入图像的模块没有运行，使得神经网络的输入一直是默认图像，于是我又使用很“工程”的方式令其一直运行，无需等待令其开始的信号。

终于，13:36，答辩前不到半小时，电路终于输出了正确结果！虽然准确率不知为何降低了，但不影响这一奇迹的光辉。

剩下的时间，我们组拍完了答辩视频，准备完答辩的演讲，最终顺利地完成了答辩。
这回为了这个项目在一周之内动员出了39.5(±10%)小时（占了7天内总时间约60%），可以说是试探出了我的极限。
最后，感谢汤皓宇同学，你完成的矩阵乘法和卷积模块是最难的两个计算模块，后来的所有神经网络模块都是基于它们修改的。感谢许浩峰同学的辛勤付出，每次接到任务你都完成得非常积极、非常认真！

课程感想

从第一个实验时的懵懂，到第二、三个实验时渐有的得心应手，再到又快又好地完成中间几个实验的欣喜，最后到倾尽全力完成大作业的热火朝天，我手上的开发板见证着我在从零开始的万里长征路上的飞速前行。每一次实验，完成的都是前一次实验时难以想象的大项目；每一次实验，体会到的都是数字系统设计的精髓；每一次实验，都成为推动课内数电学习的强劲动力；每一次实验，都为未来的计算机组成原理等课铺下了坚实基础。正是由于数电是理解计算机原理的基石，我坚持认真学习数电知识，努力提高硬件编程水平，最终在提高班里收获良多。最后的数电实验大作业，也给了我和我的组员一个机会，把我们熟悉的人工智能与数字系统结合起来，最终既锻炼了我们完成数字系统大工程的能力，全方位提高硬件编程水平，又深化了对神经网络底层原理的认识，真可谓一举两利。感谢提高班给了我如此好的锻炼平台，我将带着这份收获、这份力量，在新征程上再出发！

汤皓宇

项目总结

总的来说，这次实验实现的这些模块都是对状态机的操作。只是浮点数调试起来比较地复杂，可以考虑写个程序进行辅助。另外，字库最初并没有考虑到MSB和LSB的问题，显示出了一个对称颠倒的数字，这也是在写程序时应该考虑的部分。

课程感想

虽然以后大概率不会从事硬件设计，这学期与verilog斗智斗勇的过程依然让我学到了很多。某种意义上，这学期的数电实验还是比较困难的，我几乎每周都需要花上十几二十个小时在写verilog上，但因为verilog语言的特殊，debug的过程实在是很折磨，尤其有时看起来仿真没什么问题，却因为实现中的时序问题，导致上板之后不能正确显示，又很难找出真正的问题所在，然而仿真又几乎是唯一可以迅速判断程序是否正确的方式。不过，在程序正确跑通之后的正反馈倒也确实是十分大的。最后建议以后的数电实验课程可以考虑使学习曲线不那么得陡峭。

许浩峰

实验7总结

这次实验真是增长了见识，一方面是来自于硬件的奇妙，居然可以用开发板做这么多事情；另一方面是同学们的奇思妙想让我大开眼界，居然能想出那么多有意思的游戏。我跟着两个队友做的是数字识别，我想着前六次实验虽然费事，但是说到底还是不难，我以为自己的水平做一些打杂的工作是没有问题的。不过确实是实实在在打杂了。最大的收获是认识了两个队友，他们真的好好。给我讲东西的时候那么细心，有时候还会讲两遍。然后也借此机会了解了一下神经网络的算法，至少看的时候觉得比不少知识有趣多了。不过早知道就趁做前六个实验的时候多学点东西了，每次都是只做了基础部分就开溜。导致做实验七的时候不能给队友特别有用的帮助。不过看到最后可以输出正确的结果，我真的也开心的要死。

提高班感想

回想做的每一次实验，都是看着老师的ppt边学边做，因为张老师一般是给出了顶层模块，我可以照着逻辑设计图一步步进行模块填空，一知半解地做着module填空。感觉每一次都挺有难度的，这样的难度是体现在方方面面的：比如verilog语法、vivado的使用或者来自数电知识本身。比如实验六，我完成了基本实验模块的书写后感觉很奇怪，才通过询问助教意识到，我们是要将block memory中的rgb数据点按比例放大显示在屏幕上，而不是在屏幕上选择一个小区域一比一显示数据点。现在回想起来，当时的情况也是好笑。
写感想的时候不禁回想自己最开始为什么要加入提高班呢？当时是被开发板和上一届的作品展示吸引，一时兴起参加了提高班。但现在想来，这个决定无疑是完全正确的。虽然花的时间不少（考虑到只有一个学分，这样说应该是恰当的），不过确实是实打实学到了不少东西，自己也得以用开发板做很多事情。从最开始写一个编码器译码器都要犯一堆错误，到之后的复杂状态机的编写。自己的进步也是挺大的，而且是每一次的实验都有一次小的进步。不过现在想来，还是不应该停留在每次只完成基本实验的层次，可以多做做拔高部分，多拓展一些额外的知识，毕竟都来提高班了，也没有什么限制，这其实已经是不错的环境了。而且能通过提高班遇见这么好的两个队友，还真是一件幸运的事。下学期还可以继续学组成原理，还可以继续学习硬件，还挺开心的。到时候还要继续学习，不过还是希望自己下学期可以学得更好，可以学习更多的新东西。和队友合作是很快乐的事情，我从他们身上学到了很多，不过感觉还是有点拖队友后腿了。提高班的学习经历、自己对着电脑一次次地debug的样子以及最后上板跑出正确结果的时候真是让人难忘。希望以后的课也能这么快乐。