AACS Project Report

023039910034 徐浏凯

实验要求

本次课程项目要求设计时序逻辑神经网络加速器，基于实验1中所设计的乘累加单元，实现特定功能的神经网络推理。要求所设计神经网络加速器能够在一定周期内输出结算结果，并进行DC综合，神经网络计算周期数不定，系统计算周期为10ns。本项目不要求系统吞吐、功耗、能效、延时等指标。

电路设计

神经网络算法分析

要求实现神经网络整体结构如下Fig. 1所示。整体网络中包含两个全连接层，两个scale层，和一个ReLU激活层。其中第一个全连接层执行两个尺寸分别为[1,100]和[100,10]的矩阵乘法，第二个全连接层执行两个尺寸分别为[1,10]和[10,10]的矩阵乘法。该神经网络能够实现对手写数字的识别和分类。

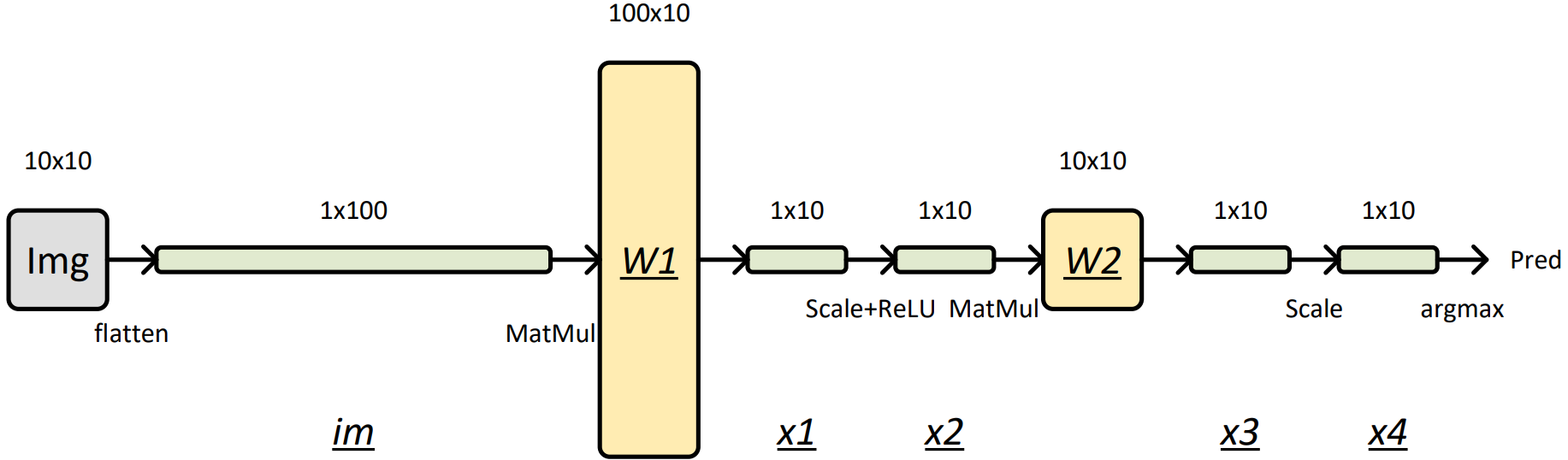


Figure 1 神经网络整体结构图

神经网络输入数据为10\*10的手写数字图片，以矩阵的形式输入，其中只包含0/1值，有笔记的位置为1，而无笔记的位置为0。Fig. 2分别展示了python代码中两张输入图像6和7的示例。

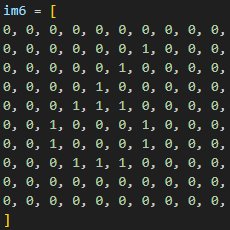
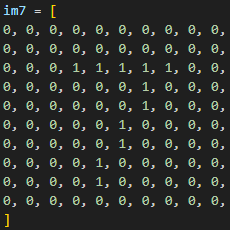
 

Figure 2 输入图像6和7示例

神经网络详细计算流程如下。首先，输入数据在进行计算前将先展开为1维形式，得到[1,100]的矩阵，再输入到第一个全连接层中，得到尺寸为[1,10]的输出矩阵。该输出矩阵将经过scale移位和ReLU激活层，得到尺寸为[1,10]的输出矩阵。该矩阵将输入到第二个全连接层，得到尺寸为[1,10]的输出矩阵，最后取得输出矩阵中最大值所对应的index即为神经网络的最终分类结果。综上所述，该神经网络的整体计算流程可以由Fig.3所示的公式进行表示。

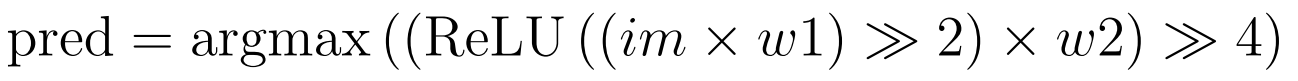


Figure 3 神经网络整体计算表示公式

其中，神经网络中的scale层以数据移位的形式实现，ReLU函数可由如Fig. 4所示的公式进行表示。在神经网络加速器中，所有的网络层的输入输出数据均以8比特有符号数的形式进行表示。

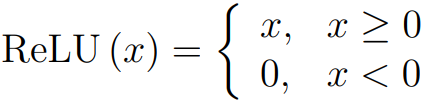


Figure 4 ReLU函数表示公式

另一方面，由于第一层全连接层的输入数据值仅为0/1，因此对于第一个全连接层，利用实验1中的可重构乘累加单元，能够在不损失神经网络精度的前提下实现两张输入图像同时推理。

神经网络加速器电路设计方法

（一）神经网络加速器端口设置

加速器整体包含多个输入输出信号，整体端口设置如下图Fig. 5所示。其中clk为时钟信号，rst\_n为重置信号，start为推理开始信号，split为并行计算信号。Weight\_we，weight\_addr，weight\_addr为加速器与权值数据缓存单元的交互信号，weight\_we为权值数据读取信号，weight\_addr为权值数据读取地址，weight\_data为读取的权值数据。Input\_we，input\_addr，input\_addr为加速器与输入数据缓存的交互信号，input\_we为输入数据读取信号，input\_addr为输入数据读取地址，weight\_data为读取的输入数据。最后的inference\_result信号代表神经网络推理结果，valid表示信号表示此时推理结果有效。

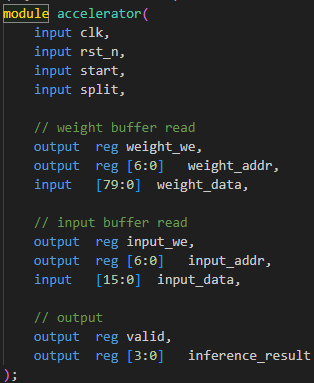


Figure 5 神经网络加速器端口设置

（二）数据流控制

由于整体神经网络包含多个层，因此通过状态机对整体神经网络执行流程进行控制。设置state变量控制整体计算流程，其整体变化流程如下图Fig. 6所示。整体流程上，神经网络逐层执行，因此状态机也设置为顺序跳转，在一张图片执行完成后，根据预先配置的并行执行寄存器，直接跳出计算回到IDLE状态或继续执行第二张图片的后续计算，若继续计算则在完成第二张图片的计算后跳回至IDLE状态。



Figure 6 状态机STATE变量跳转图

实际RTL代码根据状态机STATE变量变化图进行撰写，由于代码段较长因此不在这里贴出完整代码。所设计加速器在接收到start信号后直接开始计算第一个神经网络层，并在后续过程中通过case语句进行判断和状态跳转。在完成神经网络整体计算流程后加速器将重新回到4’b0000的IDLE状态。

（三）加速器矩阵乘单元设计

整体加速器PE阵列设计如下图Fig. 7所示。这里采用多输入并行计算的形式，其原因在于这种形式不需要额外的寄存器用于流水打拍，整体结构上更为紧凑；同时由于整体神经网络尺寸较小，且具有两个全连接层具有相同的输出数据个数（即列数）的特点，因此在进行神经网络计算时，广播输入数据并将网络权值同时输入到各个PE单元即可使整个PE阵列以满载的形式进行工作。



Figure 7 PE阵列设计图

设计加速器单元如下图Fig. 8。输入数据和权值数据会经过寄存器缓存之后进行后续的乘累加计算，避免SRAM单元的读取失败出现PE单元X态的输入数据，导致乘累加单元产生X态输出和问题输出传递的问题。



Figure 8 PE单元结构

另一方面，缓存在PE单元中的数据会在新的神经网络层执行时进行刷新，防止预存的数据导致后续网络层的计算出现错误。其中weight部分的代码如下图Fig. 9所示。其中b为PE获得的8比特权值输入数据；working为工作信号，代表加速器正处于工作状态；flush为输入的数据冲刷信号，在完成当前全连接层的计算后重置PE单元中的数据为0，防止后续计算错误。

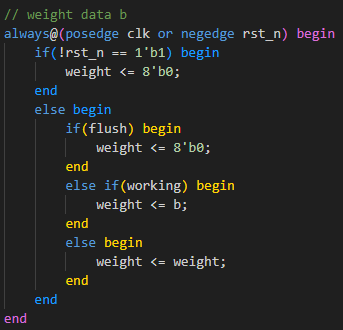


Figure 9 PE单元weight部分代码

全连接层计算过程中通过counter进行计数，控制需要读取的输入数据和权值数据的地址和计算周期数，防止整体计算结果出错。

（四）加速器移位/ReLU计算设计

所设计的神经网络加速器的移位计算通过选取输出计算结果的部分比特来完成，而ReLU激活操作通过对输出结果的最高位判断来完成。下图Fig. 10展示了第一个全连接层输出结果的移位scale和ReLU激活函数的具体实现方式。

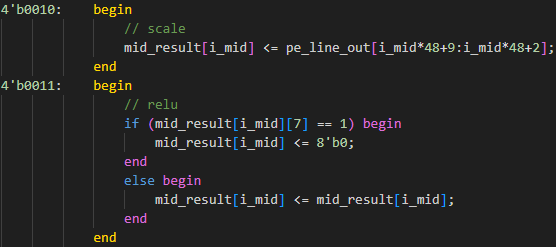


Figure 10 所设计加速器的移位scale和ReLU实现方式

（四）结果输出

在输出神经网络推理结果时，将同时将valid信号拉高表示输出结果有效。下图Fig. 11展示了valid信号的具体输出逻辑。从图中可以看到，valid信号仅在state跳转到相应完成结果输出的状态后拉高一个周期（不会反复拉高的原因是state信号也只会在该状态停留一个周期，若停留时间较长则valid信号会被反复拉高）。

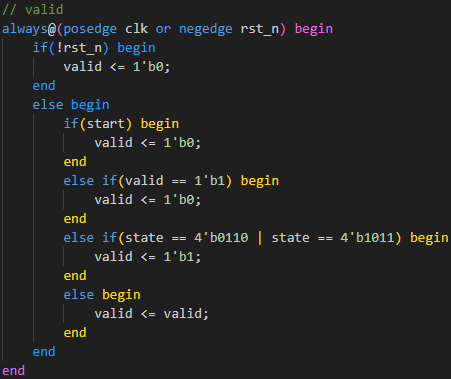


Figure 11 所设计加速器的valid信号控制逻辑

（五）多精度计算设计

由于第一层全连接层具有较长的计算延时，占到整体神经网络计算延时的85%以上；同时由于该层的输入input数据具有0/1的分布特征，因此可以采用多精度的PE阵列计算该层，实现两张输入图片的同时推理，通过这种方式能够有效提高神经网络加速器吞吐量和计算能效。

为支持多精度计算，设置了一个并行计算位，在整体神经网络开始计算前记录外部输入的计算精度要求，是否同时执行两张图片推理。如下图Fig. 12所示，parallel寄存器单元记录了外部输入的多精度要求，并始终保持该信息直到下一次输入的start信号同时对该位进行修改和调整。

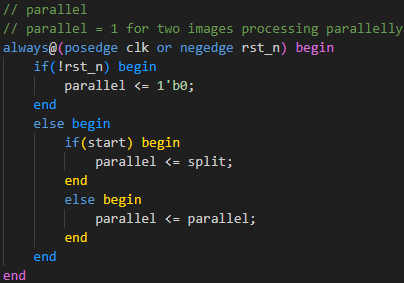


Figure 12 神经网络加速器parallel寄存器设置代码

如Fig. 6所示，在parallel为1时，即要求两张图片同时进行第一层全连接层的计算时，所设计的神经网络加速器将同时计算两张图片推理的第一层全连接层的计算，但后续的神经网络层将先针对第一张输入图片进行计算，在完成第一张图片的计算后，再对第二张图片进行推理。因此，而后续的全连接层在计算过程中并不采用两张图片同时输入的方式，因此设置PE阵列并行计算信号如下图Fig. 13所示。



Figure 13 PE阵列并行计算控制方式

另一方面，在两张图片同时输入的情形下，需要先对第一个全连接层针对两张图片的输出进行缓存。下图Fig. 14展示了部分结果的缓存方式，输出乘累加结果中的较高位部分的结果将被保存到mid\_result\_buffer寄存器当中。在后续的针对第二张图片的计算过程中，缓存在寄存器mid\_result\_buffer将在状态机控制的buffer state中被载入到mid\_result寄存器中，继续进行下一步的计算，如图Fig. 15所示。中间缓存部分的神经网络加速器电路结构如Fig. 16所示。

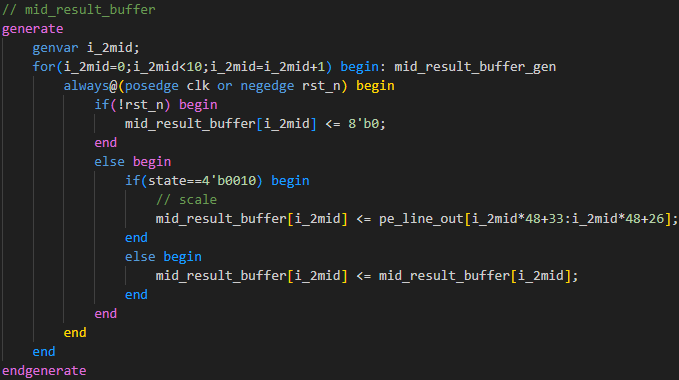


Figure 14 并行图片输入时的部分结果缓存

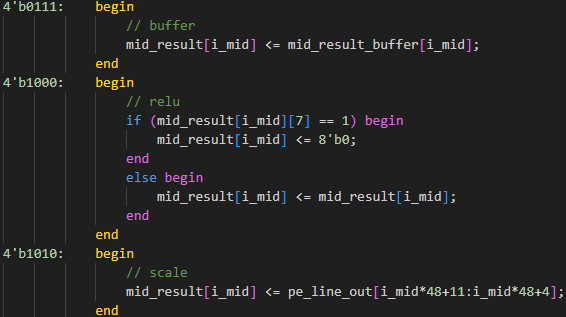


Figure 15 缓存结果的重新载入和后续计算



Figure 16 并行计算模式下，部分结果缓存的加速器电路结构

另外，为实现多精度计算，要求输入数据和权值数据能够有对应的值。这里首先讨论输入数据，这些数据在buffer中以8比特的形式进行存储，而需要同时读入两张输入图像时，就需要同时读入16比特数据，因此input buffer采用16比特位宽的存储形式，在非并行计算的情形下仅有低8比特保存输入数据，而在并行计算模式下低8比特和高8比特分别保存两张图片对应位置的输入数据。此时，需要取输入数据高位中的部分比特和低位中的部分比特作为输入数据，如下图Fig. 17所示。



Figure 17 PE阵列输入数据控制形式

同时，由于第二个全连接层需要执行两次，即读取两次权值，因此需要进行跳转回到第二层网络权值开始的存储地址，而不能直接按顺序继续读取（出现报错，读取值为X），或在buffer中存储两次相同的权值。权值读取地址代码如下图Fig. 18所示，在图中可以看到，在进行计算的过程中，若需要并行计算两张图片，则在第二张输入图片的第二个全连接层计算前，权值地址将被重置为7’B1100100的固定值。

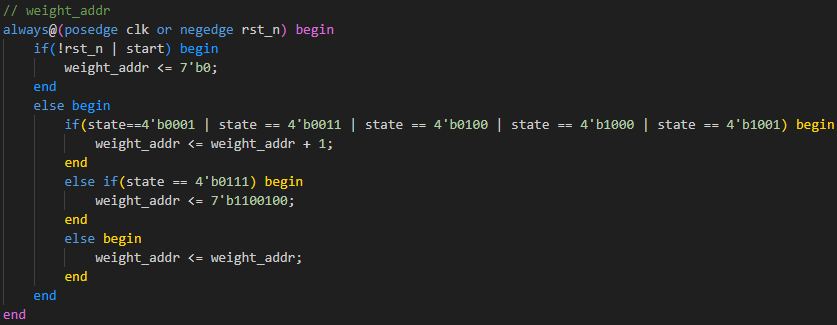


Figure 18 权值读取地址设置代码

（六）其他说明

所设计加速器要求权值数据和输入数据均从地址为0的位置开始连续存储。权值缓存单元的宽度为80比特，深度为128；输入数据缓存单元的宽度为16比特（支持同时存储两张图片，具体存储方式如（五）中所描述），深度为128。

仿真结果

仿真部分以助教给出的第一版权值和输入数据作为输入，采用im6和im7输入图像进行功能测试。

（一）python仿真

首先采用python仿真观察中间数据的数据结果，两张图像im6和im7的输出结果分别如下图Fig. 19所示。中间数据将用于最后检验输出结果正确性。

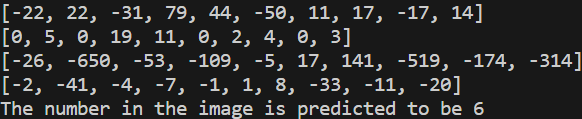
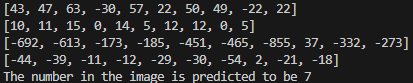
 

Figure 19 python仿真输出结果

（二）缓存单元数据文件生成

调整缓存文件生成代码，将两张图片保存在同一个缓存单元中，图片对应的点在同一行中进行保存，代码如下图Fig. 20所示。另一方面，权值缓存文件不需要进行修改，依然以原始形式进行保存。输出bin文件如图Fig. 21所示。

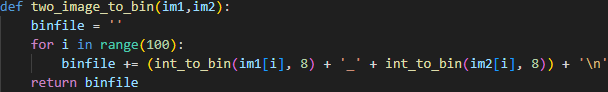


Figure 20 缓存单元数据文件生成代码

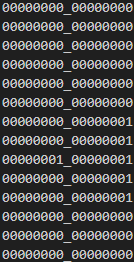


Figure 21 输入数据bin文件示意图

（三）电路仿真结果——非并行模式

该模式下的电路仿真结果如下图Fig. 22所示。从图中可以看到，所设计的神经网络加速器能够正确输出推理结果，并在输出的同时拉高valid信号一个周期。同时，在完成神经网络推理计算后，加速器将跳转回到IDLE状态。



Figure 22非并行模式下的电路仿真波形图

Fig. 23和Fig. 24展示了计算过程的中间结果，两者仿真时间对应。Fig. 23展示了第一个全连接层后的scale、ReLU、和第二个全连接层后的scale结果，从图中可以看到数据结果和Fig. 19中python仿真结果能够对应。Fig. 24展示了第一个全连接层的乘累加结果的变化值和后续的数据变化，可以看到数据结果同样可以和Fig. 19中的python仿真结果对应。

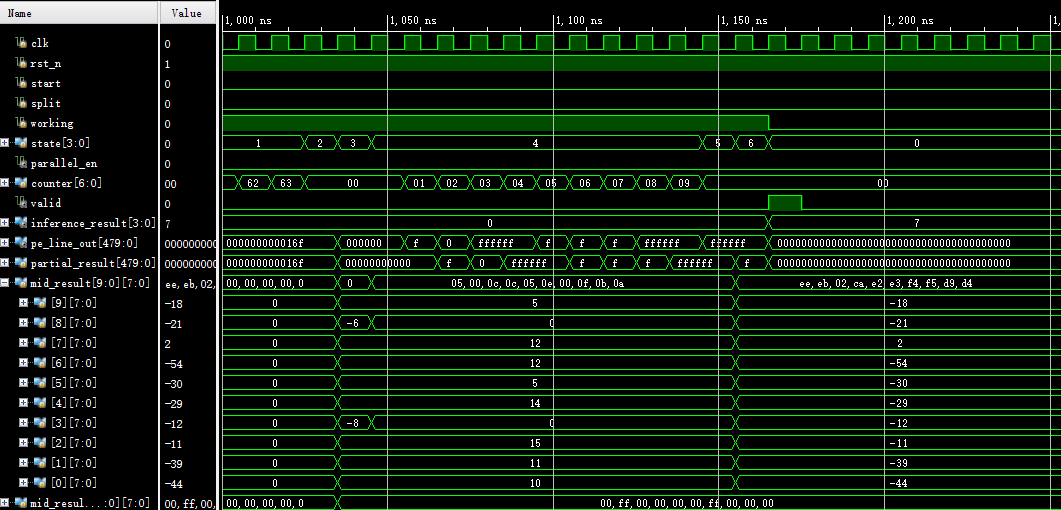


Figure 23 非并行模式下的电路仿真中间数据波形图

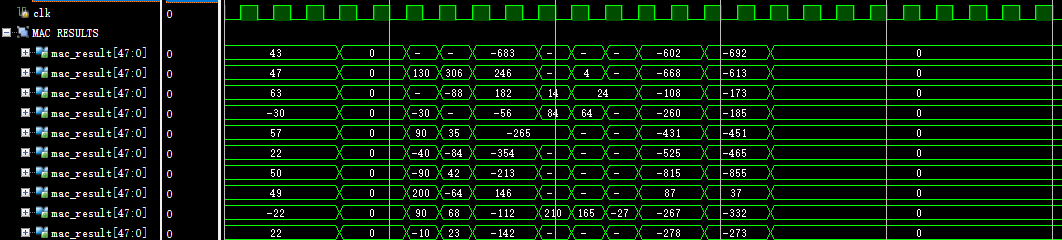


Figure 24 非并行模式下的乘累加结果变化波形图

（四）电路仿真结果——并行模式

该模式下的电路仿真结果如下图Fig. 25所示。从图中可以看到，所设计的神经网络加速器能够正确输出推理结果，并在输出的同时拉高valid信号一个周期，两个输出结果下均分别拉高了一个周期。同时，在完成神经网络推理计算后，加速器将跳转回到IDLE状态。

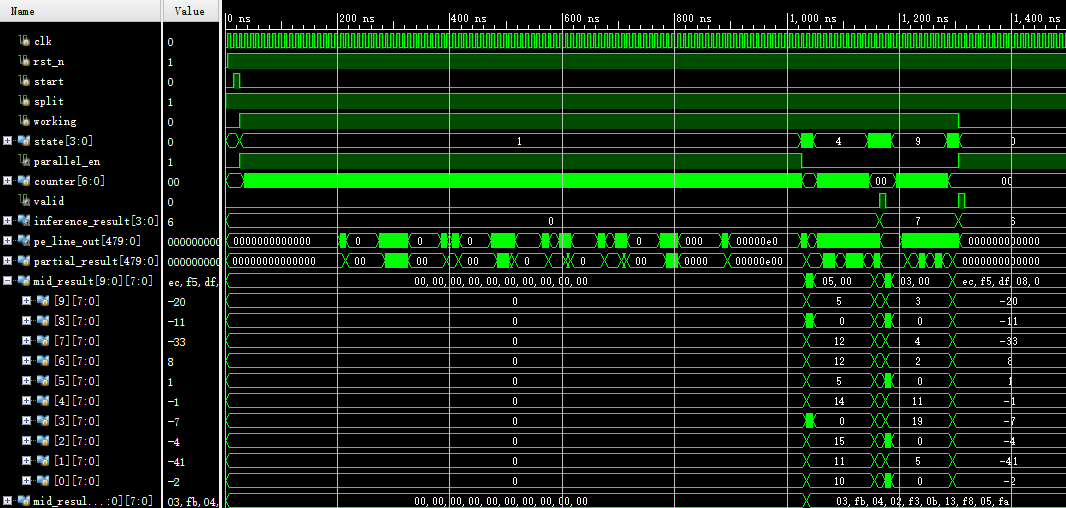


Figure 25 并行模式下的电路仿真波形图

Fig. 26和Fig. 27展示了计算过程的中间结果，两者仿真时间对应。Fig. 26展示了两张输入图片在第一个全连接层后的scale、ReLU、和第二个全连接层后的scale结果，从图中可以看到数据结果和Fig. 19中python仿真结果能够对应。Fig. 27展示了第一个全连接层的乘累加结果的变化值和后续的数据变化，可以看到数据结果同样可以和Fig. 19中的python仿真结果对应。但这里第一层输出的MAC值结果为两个24比特数据的拼接后结果，因此无法直接和python中的结果对应。

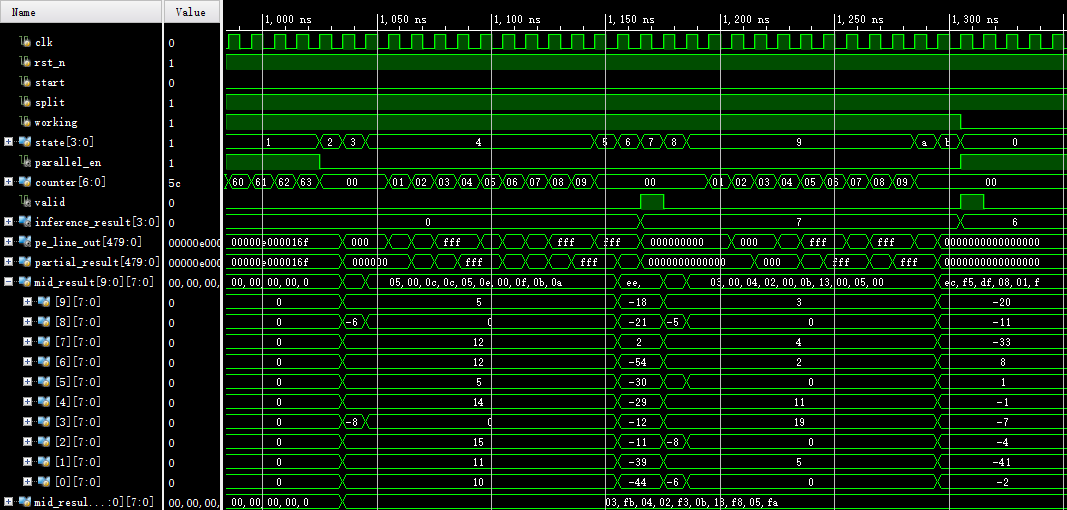


Figure 并行模式下的电路仿真中间数据波形图

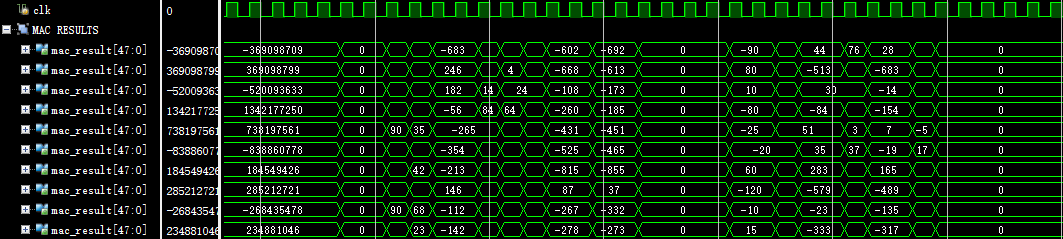


Figure 并行模式下的乘累加结果变化波形图

（四）计算延时

从Fig. 22和Fig. 25中可以看出，非并行模式下完成单张输入图像的推理和并行模式下完成两张输入图像的推理分别需要1150ns和1310ns。相比之下，并行计算模式能够降低43%的神经网络推理延时。

DC综合结果

对所编写的电路通过DC工具进行综合，功耗和面积结果如下表所示。详细DC输出文件见附件。

Table 2 Power and area of the proposed accelerator

|  |  |  |
| --- | --- | --- |
|  | Power (uW) | Area (μm2) |
| Proposed accelerator | 134.4940 | 2336.983098 |

由表可知，所设计的CORDIC计算单元功耗为134.4940uW，面积为2336.983098um2。其中，功耗部分，寄存器功耗为74.2504uW（占比55.21%），组合逻辑功耗为60.2436uW (占比44.79%)。面积部分，组合逻辑面积为1335.272581 um2，缓冲单元面积为44.221056 um2，非组合逻辑部分面积为491.006217 um2，互连线部分面积为510.704301 um2。

在时序上，输出slack为6.03ns，有足够的裕量进行计算。

Discussion

在单元设计之初曾考虑通过实时监测向量的变化，并在每一个周期都控制向量模长为1的方式。这种方式能够在输入的角度值正好等于部分角度组合的情况下提前输出计算结果，整体计算周期数不固定，但是这种情况在所有角度中的占比非常小，因此最后没有考虑这种方案，采用了固定周期数输出的方案。相比不固定周期数的输出方案，固定周期数的方案不需要每个周期保持输出向量模长相等，因此也不需要额外的LUT来输出每个周期不同的K值，能够在一定程度上降低整体电路的面积。

Appendix

Testbench top\_tb.v

accelerator.area.rpts

accelerator.power.rpts

accelerator.timing.rpts