清 华 大 学

**综 合 论 文 训 练**

题目：高能效卷积神经网络

加速器研究

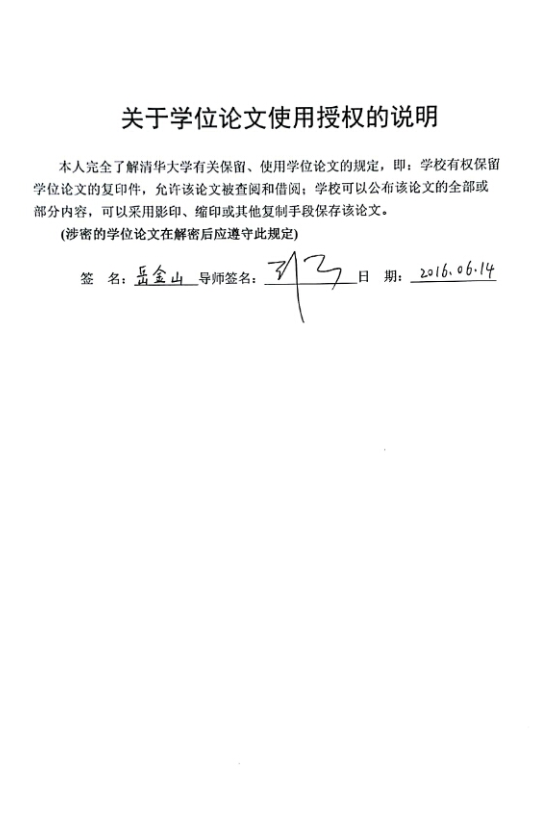
系 别：电子工程系

专 业：电子信息科学与技术

姓 名：岳金山

指导老师：刘勇攀 副教授

2016年6月5日



中文摘要

目前卷积神经网络算法在图像识别领域实现了远超出其他算法的准确度，而卷积神经网络算法所需要的计算量和数据存取使得CPU、GPU无法满足实时性和低功耗的要求。研究高性能低功耗的卷积神经网络加速器硬件是当前很多学者关心的热点，也是卷积神经网络算法要真正广泛地应用于各种移动和低功耗设备所必须解决的问题。

本文分析了卷积神经网络的结构特点及发展趋势，结合已有的加速器硬件，提出了新的卷积神经网络加速器结构，通过高效的数据复用和指令集设计，有效降低了芯片的功耗，提高了性能和能量效率，并且通过Modelsim仿真和FPGA测试验证了其实际功能。理论分析和版图综合结果显示，相比目前性能最好的通用大规模神经网络加速器，本文的设计减小了87.8%的芯片面积，功耗降低了67.1%，而运行速度提高了52.6%，能量效率提升了至少2.4倍。

**关键词：**卷积神经网络加速器；数据复用；指令集；低功耗；能量效率

ABSTRACT

Currently Convolutional Neural Network (CNN) has realized an accuracy that is much higher than other algorithms in image recognition field. Meanwhile, CPU and GPU cannot meet the requirement of low power consumption and real-time property because of the huge amount of calculation and data access. It has become a hot issue to develop a CNN hardware accelerator with high performance and low power consumption, which is also a critical problem to be solved before CNN can really be broadly adopted in various mobile and lower power electronic devices.

This paper analyses the structural specialty and tendency of CNN algorithm, and proposes a novel CNN accelerator structure based the proposed CNN accelerators. The proposed structure reduces power consumption and improves performance and energy efficiency by dedicated instruction set architecture design and efficient data reuse. The functional verification has been implemented on Modelsim simulation and FPGA platform. Theoretical analysis and layout synthesis show that, compared with the state-of-art accelerator that can run generic and large-scale CNN Network, the proposed accelerator has achieved 87.8% reduction on chip area, 68.2% reduction on power consumption, 52.6% on performance enhancement and at least 2.4x energy efficiency.

**Keywords:** Convolutional neural network accelerator; Data reuse; Instruction set architecture; Low power consumption; energy efficiency

目 录

[第1章 引 言 1](#_Toc453922411)

[1.1 研究背景与意义 1](#_Toc453922412)

[1.2 卷积神经网络算法结构 1](#_Toc453922413)

[1.2.1 卷积层 2](#_Toc453922414)

[1.2.2 降采样层和全连接层 2](#_Toc453922415)

[1.3 卷积神经网络加速器 3](#_Toc453922416)

[1.4 研究内容和结构安排 4](#_Toc453922417)

[第2章 加速器硬件设计的关键要素 6](#_Toc453922418)

[2.1 卷积神经网络结构特点 6](#_Toc453922419)

[2.2 现有卷积神经网络加速器 8](#_Toc453922420)

[2.2.1 ASSCC 2015的CNN加速器 8](#_Toc453922421)

[2.2.2 MIT的CNN加速器 9](#_Toc453922422)

[2.3 卷积核小型化趋势 11](#_Toc453922423)

[2.4 本章小结 12](#_Toc453922424)

[第3章 加速器硬件结构设计 13](#_Toc453922425)

[3.1 加速器核心计算电路设计 13](#_Toc453922426)

[3.1.1 核心计算电路结构 13](#_Toc453922427)

[3.1.2 核心计算电路工作模式 13](#_Toc453922428)

[3.1.3 核心计算电路兼容性 14](#_Toc453922429)

[3.2 加速器整体结构设计 15](#_Toc453922430)

[3.3 加速器控制结构设计 16](#_Toc453922431)

[3.3.1 加速器指令集设计 17](#_Toc453922432)

[3.3.2 算法到指令的映射 18](#_Toc453922433)

[3.4 加速器性能分析 18](#_Toc453922434)

[3.4.1 硬件资源分析 19](#_Toc453922435)

[3.4.2 执行性能分析 19](#_Toc453922436)

[第4章 实验验证 21](#_Toc453922437)

[4.1 功能验证 21](#_Toc453922438)

[4.2 性能比较 24](#_Toc453922439)

[第5章 结 论 26](#_Toc453922440)

[插图索引 27](#_Toc453922441)

[表格索引 28](#_Toc453922442)

[参考文献 29](#_Toc453922443)

[致 谢 31](#_Toc453922444)

[附录A 外文资料的书面翻译 33](#_Toc453922445)

[在学期间参加课题的研究成果 41](#_Toc453922446)

# 第1章 引 言

## 研究背景与意义

大数据时代已经到来，国际数据公司IDC2014年的报告显示，通过各种方式获取、存储和处理的数据量正在以每年40%的速率增加，到2020年将会达到44ZB[1]。大数据中包括了各种类型的数据，如图片、视频、语音、监控设备以及各种传感网络获取的数据，如何快速、高效且低功耗地处理大数据成为一个关键的问题。

目前神经网络算法在很多数据处理的领域，尤其是图像、语音方面受到了很大关注。到2011年，在ILSVRC的ImagNet图片测试集上，各种算法对于图片分类和识别的精度提升已经不明显，而在2012年，Hinton等人的神经网络算法将ImageNet图片识别准确率急剧提升了11%，达到了85%[2]。神经网络重新受到重视，并且在2015年，微软亚洲研究院MSRA把神经网络算法的准确率刷新到了96.4%[3]。神经网络算法对于语音图像处理的精确度击败了其他执行同类任务的算法。

然而，神经网络识别精度的提升带来了计算量的增加。在2014年ILSVRC识别精度排名第二的VGG模型[4]中，一张224\*224像素的图片识别需要超过30G次的乘法和加法。这对于传统的CPU和GPU计算而言意味着大量的运算时间和能量消耗，在对于速度和功耗要求苛刻的移动设备上是不可忍受的。因此，研究设计高速低功耗的神经网络加速器硬件，能够更好地满足移动设备上广泛的数据处理需求，能够促进大数据处理的发展。

卷积神经网络是目前应用最广的几种神经网络结构之一，且对于较大图片识别性能最好，远远超过其他算法。本文将研究实现高性能低功耗的卷积神经网络加速器硬件，通过分析卷积神经网络算法结构特点和当前加速器硬件的不足，优化设计新的硬件结构，实现更低的功耗和更高的能量利用效率。

## 卷积神经网络算法结构

在众多神经网络算法中，卷积神经网络(CNN)目前得到广泛应用的网络结构之一[14]。卷积神经网络CNN在图像识别与目标检测方面实现了大大优于其他检测算法的结果[2]。

卷积神经网络的典型结构如图1-1[2]。

卷积神经网络的主要结构有卷积层(Convolution)、降采样层(Pooling)和全连接层(Full-Connected)[2]。

### F:\mine\论文\图1-1.png卷积层

卷积层通过训练好的滤波器(Filter，也称为卷积核 conv kernel或权重Weight)与本层的输入图像(Figure)做二维卷积，这样的滤波器Filter有多组，分别将本层输入图像Figure中包含的不同信息反映到不同的输出图像(Output)中。同时不同的输入图像会将各自携带的同类信息累加到同一个输出图像中，在算法上表现为加法操作。因此卷积层的表达式如下：

图 1.1 卷积神经网络典型结构[2]

(1-)

其中表示第*i*个二维输出图像，表示第个输入图像，表示与和对应的卷积核，和分别表示输入图像和输入图像的数量，是卷积函数，表示对应的偏置项。累加之前的结果，即，一般称为部分和(Partsum)。

卷积得到的结果可能会经过非线性的激活函数ReLU(Rectified Linear Units)处理，如sigmoid函数或者函数[2]。

### 降采样层和全连接层

卷积层得到的输出一般会经过降采样层(Pooling)。Pooling通过降低图像的像素点数量来降低之后的网络的计算量和复杂程度，具体表现方式为对邻近的几个像素点（如2x2的块）求最大值或者取平均变成一个新的像素点。

在整个网络的最后一般是1~2层的全连接层。全连接层类似矩阵乘法，将当前得到的输入图像排成一行，然后右乘一个矩阵得到新的输出图像。相比同样大小的卷积层，全连接层需要更多的权重，但计算操作较少。全连接层表达式如下：

(1-)

其中F是输入图像构成的1xM维行向量，W是MxN维权重矩阵，B是1xN维偏置向量，O是输出的1xM维行向量。

目前对于神经网络算法尚未给出严谨的数学推导和理论解释，研究者主要根据经验来对卷积神经网络进行设计，通过修改卷积层、pooling和全连接层的规模和卷积神经网络的结构来不断提高识别的准确度。

## 卷积神经网络加速器

卷积神经网络在图片识别等领域表现出了远超出其他识别算法的突出效果，但是巨大的计算量和运算时间则是卷积神经网络的瓶颈所在。近年ILSVRC[5]上识别效果最好的几种典型CNN算法的识别精度和计算量列在表1-1中[2][4][6]。运算量的单位是OPS (operations)和MAC(multiply and accumulate)，在CNN算法中乘法和加法的数量相当，1 OPS相当于一次乘法或者一次加法操作，而1MAC是一次乘法和一次加法的组合，即。由于一些CNN算法结构复杂，其运算量难以计算，未在表中列出。

表 1.1 ILSVRC中识别精度最好的几种CNN算法运算量与精度

|  |  |  |  |
| --- | --- | --- | --- |
| 年份/名次 | CNN算法 | 总运算量(OPS) | 识别精度 |
| 2012/ 1st | AlexNet[2] | ~3.2G | 84.70% |
| 2013/ 1st | ZF Net[6] | ~18.2G | 88.80% |
| 2014/ 2nd | VGG-E[4] | ~30.8G | 92.60% |

可以看到随着对于CNN算法的研究不断深入，网络结构在不断改进，CNN的识别精度在不断提升。随着网络结构改进，网络规模也在逐渐扩大，一方面增加了算法的时间开销，另一方面也导致了功耗的急剧上升。

因此，一些研究者开始考虑针对神经网络的硬件加速器设计，中科院的陈云霁等人给出了一种基础的CNN硬件实现方式[7]。表1.2给出了利用CPU、GPU和CNN加速器DianNao来计算同一种CNN网络的性能功耗对比（DianNao数据为理论值）。从表中可以看出，传统的CPU加速器无法实现实时处理，善于并行计算的GPU在保证实时的同时带来了严重的功耗开销，而CNN自身的结构特点则使得专门设计的神经网络加速器硬件则可以在保证实时性的同时，极大地降低计算功耗。

表 1.2 CPU、GPU、DianNao执行VGG的性能功耗对比

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | DianNao[7] | CPU | GPU K20M | 对比 |
| 特征 | 3.02mm2 | 128bit SIMD | 550mm2 | \* |
| 频率 | 0.98GHz | 2GHZ | \* | \* |
| 功耗 | 485mW | 86.6mW | 253W | \* |
| 计算速度 | 482GOP/s | 4.08GOP/s | 266GOP/s | 118:1:65 |
| 能量效率 | 993GOPS/W | 47.1GOPS/W | 1.05GOPS/W | 946:45:1 |
| VGG时间 | 0.064s | 7.54s | 0.116s | 1:118:1.8 |

对于卷积神经网络的低功耗高能效硬件设计将能够极大促进各种移动设备上的神经网络应用发展，这样的性能、功耗和能效是传统的CPU和GPU结构很难满足的。KAIST的Injoon Hong[8]、Jaehyeong Sim[9]和MIT的Yu-Hsin Chen[10]等人针对不同规模的CNN算法，分别设计并实现了相应的卷积神经网络硬件加速器，相对于传统的CPU和GPU提升了计算速度，降低了功耗，实现了更高的能量效率。关于具体的神经网络加速器硬件设计将在第二章详细介绍。

## 研究内容和结构安排

本文将对更加高性能低功耗的卷积神经网络加速器进行研究，设计更加合理的电路结构，实现相比于CPU、GPU和其他CNN硬件加速器更好的能量效率和能耗。文章的结构安排如下：

第2章结合卷积神经网络的结构特点和现有的CNN加速器硬件的优点与不足，分析CNN加速器硬件设计优化的关键要素，以及CNN网络结构的发展趋势，确定加速器设计的参考要求。

第3章给出具体的加速器硬件结构设计和运行流程，包括具体的核心电路设计和整体芯片模块设计，介绍了电路工作模式、对于不同结构网络的适应性，并且具体介绍了控制结构与指令集的设计和加速器芯片的数据流图。最后从理论上对于加速器的优缺点进行了分析。

第4章分别基于仿真和GPGA平台验证了所设计CNN硬件加速器的功能，并且基于版图和后端工具给出的相关功耗、面积等参数，结合硬件与指令集设计，对比CPU、GPU和其他专用加速器硬件，评价设计芯片的性能。

第5章概括了本文的研究内容，给出结论，并且探讨卷积神经网络加速器新的设计优化方向。

# 第2章 加速器硬件设计的关键要素

本章分析卷积神经网络结构特点和现有CPU、GPU结构的不足，分析硬件加速器设计的关键要素，并且基于现有的典型硬件加速器结构，在下一章提出新型结构的卷积神经网络硬件设计。

## 卷积神经网络结构特点

本节说明专用硬件如何利用卷积神经网络自身的结构特点，获得大大优于CPU的计算速度，以及远超过CPU和GPU的能量效率。设计专用的卷积神经网络硬件是出于满足更低功耗和更高能效的要求，而CNN算法自身的特点则为硬件设计提供了广阔的优化空间。

表2.1给出了VGG-A卷积神经网络中各卷积层的参数[4]，第一列为输入和输出图像的大小，第二列为卷积核大小，第三/四列为输入和输出图像的数量。根据公式(1-2)和1.3节的相关介绍可以算出所需要的计算量和对应的参数(Filter)数量。

表 2.1 VGG-A结构参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Figure大小 | Filter大小 | I\_Channel | O\_Channel | 计算量(MAC) | 参数数量 |
| 224x224 | 3x3 | 3 | 64 | 86,704,128 | 1,728 |
| 110x110 | 3x3 | 64 | 128 | 892,108,800 | 73,728 |
| 55x55 | 3x3 | 128 | 256 | 892,108,800 | 294,912 |
| 55x55 | 3x3 | 256 | 256 | 1,784,217,600 | 589,824 |
| 27x27 | 3x3 | 256 | 512 | 859,963,392 | 1,179,648 |
| 27x27 | 3x3 | 512 | 512 | 1,719,926,784 | 2,359,296 |
| 13x13 | 3x3 | 512 | 512 | 398,721,024 | 2,359,296 |
| 13x13 | 3x3 | 512 | 512 | 398,721,024 | 2,359,296 |

由表中数据就可以看出，最影响功耗和速度的是计算和数据搬运两部分。在VGG-A的卷积层，总的乘累加操作已经达到了7.03G次，而理想情况下的参数搬运则是9.22M，在双精度浮点运算下的存储空间需求为73.8MB。

对于一个2GHz频率的CPU，理想情况下所需要的时间在秒量级以上，另一方面考虑到CPU的cache 容量大小，这些数据无法一直保留在cache中，导致多次重复从内存中搬运参数的开销，严重影响了CPU功耗；而GPU虽然通过并行提高了速度，但却以更多的数据读写操作为代价，进一步降低了能量效率。

CNN专用硬件的解决思路在于尽可能增大并行计算规模以提高速度，同时降低单次计算的功耗、提高数据复用来降低功耗。

降低计算功耗的方法在于降低计算精度。研究发现将单精度浮点运算变为16bit甚至8bit整数运算，对网络整体精度的影响可以控制在低于1%[11]。这意味着可以牺牲小于1%的识别准确度，将计算部分的功耗降低到原来的13.6%[7]，同时也将图像和参数存储的压力降低到了原来的1/2甚至1/4。

降低数据存取功耗的方法在于数据复用。图2.1给出了“1 Figure-1 Filter”的复用机会。对于一个13x13的图像(图中蓝色部分)和一个3x3的卷积核(图中黄色部分)的卷积操作，当完成图2.1(a)所示位置的乘累加操作后，继续执行图2.1(b)位置的乘累加操作时，红色方框内的6个数据点可重复利用，可以被留在计算单元内部，而不需要从下一级存储单元重新载入。以此类推，整个图像中的绝大部分数据都可以重复利用，理论上只需要载入一次图像就可以完成1 Figure-1 Filter的卷积操作，而如何实现这样高效的数据复用是专用加速器设计的重点。

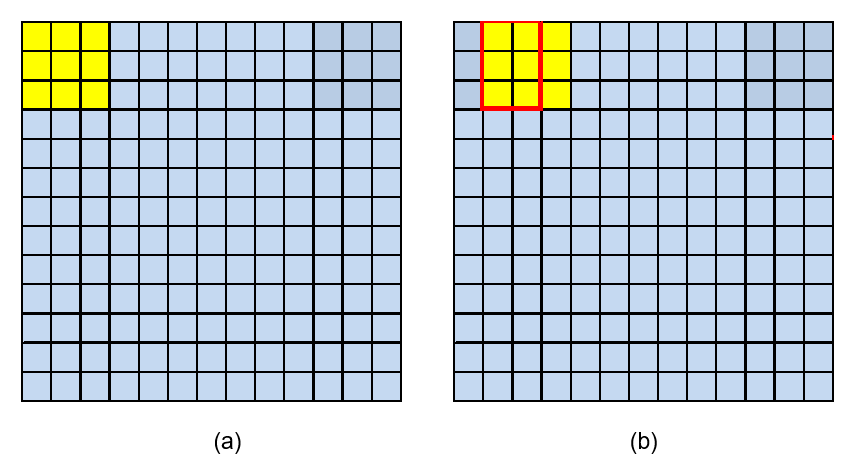
另一种数据复用在于1个输入图像会与多个不同的卷积核进行卷积操作，即“1 Figure-n Filter”，通过一次载入多个卷积核，在计算单元内部轮换完成1 Figure-n Filter的卷积操作，将输入图像的载入次数变为1次，但必须存储当前卷积的结果[[1]](#footnote-1)，用于之后的加法操作，造成额外的存储开销。但从输出角度来看，可以认为是“m Figure-n Filter”，直接由多个输入图像产生本层卷积网络的最终输出，避免了中间输出结果的大量保存，但反过来导致了输入图像的重复载入。两种方法相互冲突，不能同时实现，因此一定的存储开销不可避免。

图 2.1 1 Figure-1 Filter 层次的数据复用

还有一种更大规模的数据复用。对于连续的识别任务，当整个CNN算法执行完一遍后，对另一个输入图像重复执行时，所有的参数又会被重新调用。考虑到网络规模过大，这样的复用只存在于当前存储空间足够放下所有参数的情况下，否则必须从下一层存储空间重新载入参数。并且这种数据复用并没有硬件设计的技术而只取决于网络和硬件规模，因此本论文不考虑这种数据复用。

## 现有卷积神经网络加速器

现有的CNN硬件加速器在网络规模、网络结构适应性、性能参数方面各有不同。对于能够执行大型通用CNN网络的加速器，目前已经可以实现功耗278mW，速度33.6~84.0GOPS，能效83GOPS/W，相比速度是其两倍的GPU能耗降低了97.3%，相比于传统的频率2GHz的GPU速度则提高了10倍以上。表2.2给出了一些代表性的硬件加速器的特征和性能参数[[2]](#footnote-2)。

需要指出的是，数据存取操作的功耗对于CNN网络是非常重要的。很多研究者都强调了减少各层次的数据存取来降低功耗[7][10]。从表2.2中可以看出，相比于小型网络最高TOPS/W量级的能量效率，大型网络的能效却小了近一个数量级。因为小型网络的计算操作更加密集，片上存储可以很小，不需要片外存取，片上存取也较少。而大型网络的图像和参数不可能全部存放在片上，因而产生大量的片上和片外存取操作。因此降低存取是卷积神经网络加速器设计的一个关键。

下面通过两种具体的CNN加速器来说明两种数据复用的实现方式，并基于这些实现方式给出更加合理的加速器硬件设计方案。

### ASSCC 2015的CNN加速器

在2015年的ASSCC会议上，Injoon Hong等人基于“1 Figure-1 Filter”的数据复用提出了一种硬件结构，如图2.2所示[8]。网络采用的卷积核全部为5x5大小，在硬件结构上复用了中间4列的输入图像的数据，即左侧5x5乘法器执行输入图像的第1列到第5列的乘累加的同时，右侧5x5乘法器执行输入图像第2列到第6列的乘累加。通过每一列在5x5乘法器阵列中的数据移位使得每次输入5个数据就可以完成25次乘法操作，大大降低了从下一级存储的数据搬运。

表 2.2 典型神经网络加速器特征和性能参数

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | DianNao[7] | Shi[12] | FPGA[11] | ASIC[8] | KAIST[9] | MIT[10] |
| 年份 | 2014 | 2015 | 2015 | 2015 | 2016 | 2016 |
| 会议/期刊 | ASPLOS | ISCA | unpublished | ASSCC | ISSCC | ISSCC |
| 算法规模 | \* | 小 | 大 | 小 | 小 | 大 |
| 图像大小 | \* | <64\*56 | 224\*224 | 160\*90 | 28\*28 | 224\*224 |
| 是否通用 | 是 | 是 | 是 | 否 | 是 | 是 |
| 工艺 | 65nm | 65nm | \* | 65nm | 65nm | 65nm |
| 计算单元 | 16-bit | 16-bit | 8-bit | \* | 16-bit | 16-bit |
| 带宽要求 | 467.3GB/s | 62.5GB/s | 4.2GB/s | 0 | \* | 127MB/s |
| 电压 | \* | \* | \* | 1.2V | 1.2V | 1V |
| 频率 | 0.98GHz | 1GHz | 150MHz | 200MHz | 125MHz | 200MHz |
| 面积(mm2) | 3.02 | 4.86 | \* | 1.554 | 16 | 12.25 |
| 功耗(mW) | 485 | 320.1 | 9630 | 9 | 45 | 278 |
| 速度(GOPS) | 482 | 194 | 136.97 | 42 | 64 | 84 |
| 能效(GOPS/W) | 993 | 606 | 14.2 | 2335 | 1420 | 166 |
| 是否流片 | 否 | 否 | FPGA实现 | 是 | 是 | 是 |

这种结构的不足一是5x5卷积核的设计不利于扩展，在执行其他大小的卷积核结构(如3x3,7x7)时会造成浪费或者无法运算，而是虽然双计算核的结构将图像重复读入的次数降低到了原来的1/2，但无法实现理论上输入图像只读入一次的最高效率。

### MIT的CNN加速器

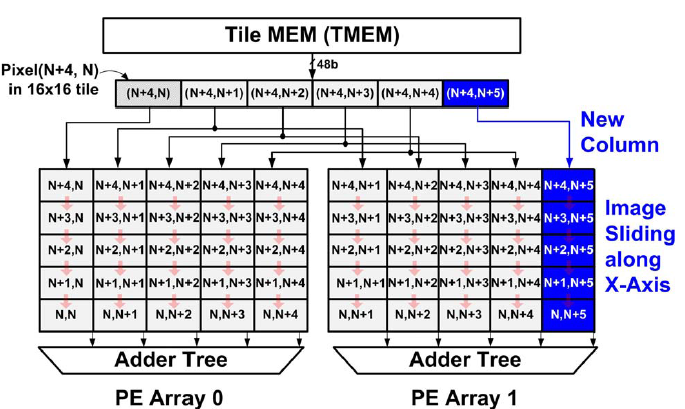
MIT的Yu-Hsin Chen等人提出了一种新的适合大型CNN网络的通用CNN加速器结构Eyeriss [10]。这里的大型一般指用于ILSVRC中的224x224图像的CNN网络或者规模类似的网络，如AlexNet、VGG等，计算量在GOPS量级，参数数量在1M以上。

图 2.2 ASSCC 2015的硬件加速器设计[8]

Eyeriss的结构实现了“1 Figure-1 Filter”和“1 Figure-n Filter”和两种数据复用方式，如图2.3所示[10]。

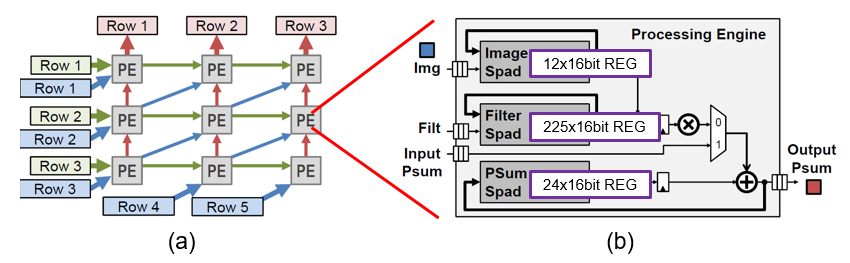
图2.3(a)展示了“1 Figure-1 Filter”数据复用。芯片整体有14行12列一共168个PE计算单元，左侧蓝色框表示Figure，青色框表示Filter，上方红色框表示输出，箭头连接表示这些位置的数据是通过一次载入操作从存储导入到PE中，在网络合适的情况下可以达到只输入一次的理想数据复用。

图 2.3 MIT的Eyeriss硬件加速器设计[10]

图2.3(b)展示了“1 Figure-n Filter”数据复用。每个PE核存放了一行的数据和需要与这一行数据卷积的多个Filter，因此对于多个Filter的操作，不需要在更换Filter时，再次把图像重复载入。

这种结构的不足一是由于PE核数量较多且控制信号并不相同，控制指令会比较复杂，二是对于任一种卷积核都存在PE核浪费的情况，执行AlexNet的平均利用率为88%[10]。此外，硬件规模限制了数据复用的次数，如在输入图像的规模超过16x16的情况下，无法一次把整个Figure放进PE阵列，导致一定的重复开销。

MIT的Eyeriss是我们了解到的所有CNN硬件加速器中，真正实现了AlexNet这样的大型CNN网络结构的硬件设计中性能最好的结构，并且给出了具体的面积、功耗和执行时间等参数，因此第3章和第4章的性能比较均以MIT的Eyeriss作为参考。

## 卷积核小型化趋势

卷积神经网络专用硬件的设计依赖于CNN网络结构，尽管目前的网络结构和规模多种多样，但小规模的卷积核使用越来越多。用三层3x3的卷积核网络即可实现类似一层7x7的卷积核网络，同时将卷积核参数的数量降低了45%，同时计算精度也有明显提升[4]。图2.4给出了近年ILSVRC上对于ImageNet数据集几种最高精度算法的卷积核比例，可以看出3x3卷积核比例的提升对于精度提高是非常明显的，VGG-A甚至全部使用了3x3卷积核。尽管卷积核大小多种多样，但3x3卷积核作为CNN网络结构中的主要部分，需要在硬件实现中优先考虑，在此之外需要考虑兼容1x1,5x5,7x7等其他大小的卷积核。而且相对于硬件的不可更改，在CNN网络结构上的结构调整则更加灵活。目前也有一些很好的工具可以用来训练新的神经网络结构[13]。相比于要求硬件实现更加复杂的通用，通过调整CNN网络结构可以同时得到识别精度和能量效率的提升。

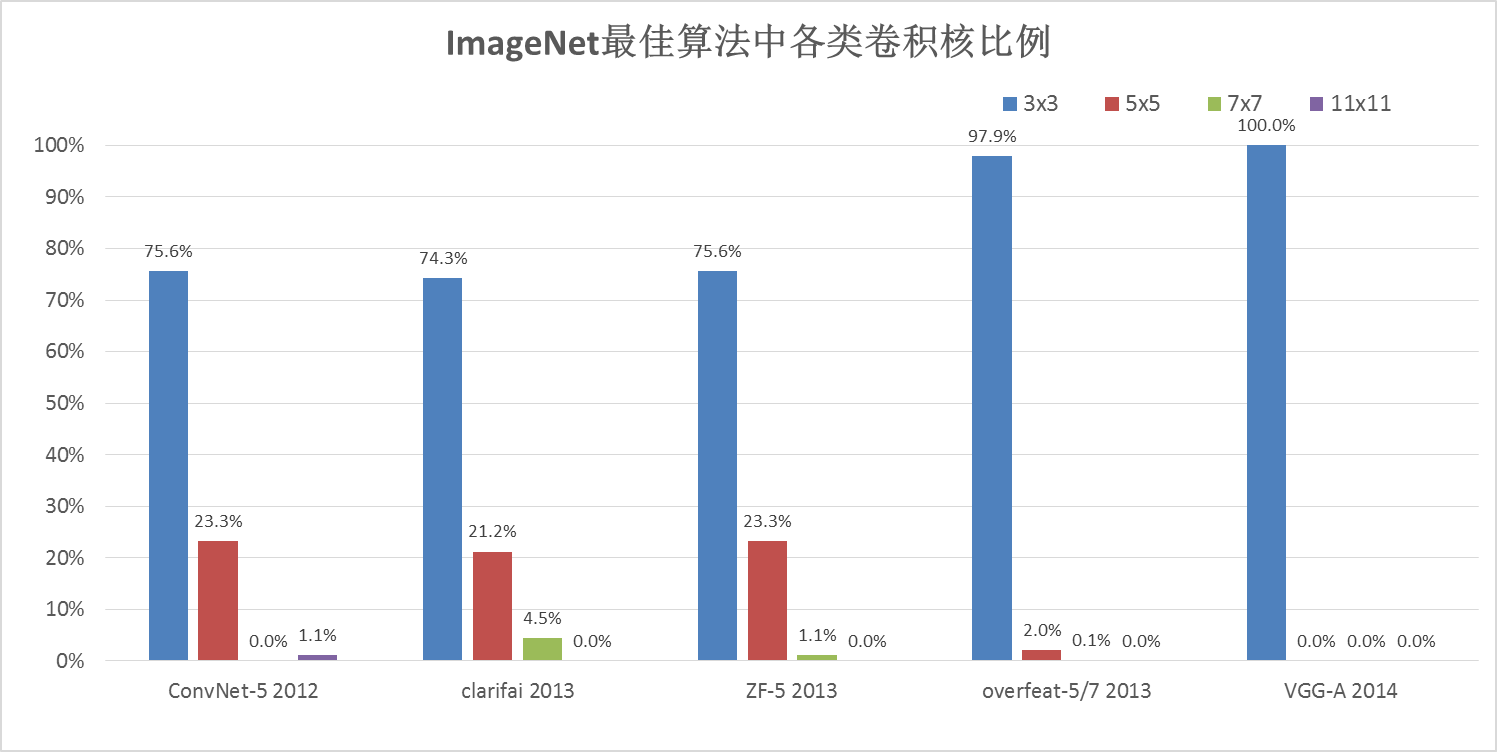


图 2.4 ImageNet 各年最佳算法中的卷积核比例

## 本章小结

由2.1、2.2和2.3节的分析可以知道，CNN加速器的设计需要合理利用CNN网络结构特点实现高速低功耗，具体来说要实现尽可能多的数据复用，减小对下一层存储的数据存取操作；要在满足识别正确率的前提下尽可能降低乘法器规模；网络必须具备足够的通用性且更支持小型卷积核，可以执行当前各种大小的卷积核和图像的卷积操作。

除此之外，CNN加速器还需要考虑计算单元和存储单元的配合，ReLU和Pooling等操作的实现，如何进行指令控制和调度，数据的组织存取和带宽的分配等一系列重要问题。

# 第3章 加速器硬件结构设计

本章根据第2章的分析，给出更加合理的卷积神经网络硬件加速器设计结构，包括加速器核心电路设计和芯片整体结构设计，给出芯片的整体控制结构和指令体系设计，以及如何从软件代码映射到加速器硬件指令。

## 加速器核心计算电路设计

### 核心计算电路结构

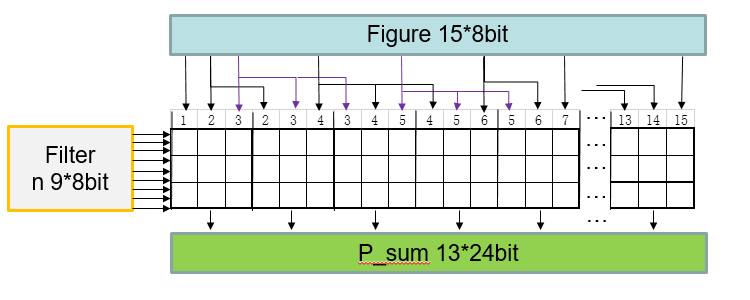
 图3.1给出了新的CNN硬件加速器核心电路结构。上方是输入图像，每次输入一行15个数据，每个数据8bit；左侧是卷积核，放有n个3x3的卷积核（目前的设计值n=4），每个数据8bit，每次输入一个卷积核，即9个数据；中间核心部分每一个小正方形代表一个乘法器单元，每一个3x3的大正方形组成一个3x3卷积单元，将内部9个小正方形的乘法结果加起来得到一个输出。乘法器一共3行39列。下方是输出结果，一共13个，每个占据24bit的存储空间

图 3.1 加速器核心电路结构

### 核心计算电路工作模式

如图3.1所示，核心计算电路运行的过程中，首先将三行的输入图像串行放入正方形的乘法器格子里，每次输入一行15个数据，同时每一行正方形内的数据会在下一个时钟上升沿移动到下一行正方形内，通过位移来完成卷积核在图片上的滑动卷积操作。当输入的图像和卷积核数据准备好后，所有乘法器同时执行，在一个时钟周期后得到结果，并且在两个周期后将每个卷积计算单元的9个乘法结果累加，得到当前的13个输出。整个核心电路以流水线的方式运行。

核心电路左侧Filter输入的9个卷积核参数被13个3x3的大正方形卷积计算单元共享。上方输入的每一个数据被分配给一个或多个乘法器的输入，如连线所示，对于一个不超过15列的图像，只需要按行串行地从上向下穿过整个核心计算电路，就可以完成“1 Figure-1 Filter”的卷积，实现理论上的最优复用。

为了实现“1 Figure-n Filter”的数据复用，每次输入图像数据准备好做乘法时，对应的多个卷积核会轮换进入到乘法器的输入端，当输入图像完成移位再次准备做乘法时，卷积核再次轮换一遍。因此，卷积核心电路实现了“1 Figure-1 Filter”和“1 Figure-n Filter”这两种数据复用模式，同时相比于MIT的Eyeriss减少了大量的Filter存储。

### 核心计算电路兼容性

图3.1所示的电路结构对于3x3的卷积核实现了最佳利用。考虑到从2012年的AlexNet到2014年的VGG，后几层卷积网络中13x13的图像结构始终保持，并且可以较为简单的对前几层网络中27x27和55x55等较大的图像进行分块，因此采用了13个3x3卷积计算单元的形式来设计电路。而对于其他的网络，也可以通过类似的切分来分块实现，但效率可能会有一定的降低。

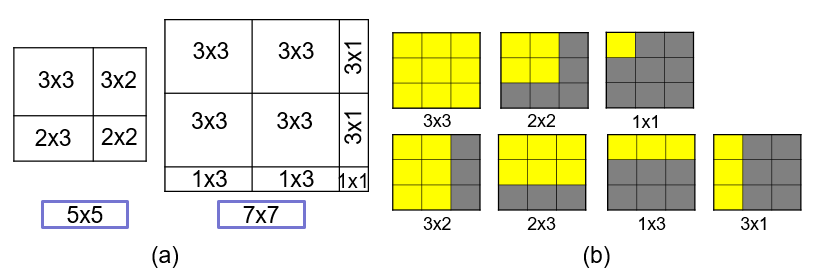
对于其他大小的网络，如5x5或者7x7等等，解决方式如图3.2所示。按照图3.2(a)的方式把规模超过3x3大小的卷积核切分为大小不超过3x3的小模块，即可以按照图3.2(b)所示的方式放到图3.1的核心计算电路中执行。其中黄色的部分表示执行乘法，灰色的部分表示不执行，即对应的乘法器结果为0。这样保证了9个乘法器累加结果的正确性，同时没有增加额外的计算功耗，只有微小的静态功耗开销。然后在核心电路外围，这些模块的输出结果经过累加，完成等效于原来5x5或者7x7的卷积操作。在此过程中需要考虑到输入数据需要重新组织，以保证各个块按照正确的对应位置累加求和。执行其他大小的卷积核会影响电路结构的效率，特别是对于1x1的卷积核而言效率降低到1/9。但考虑到3x3卷积核的占比最大，对于系统整体的效率而言影响并不严重，并且可以通过修改CNN算法将其训练为3x3卷积核网络，而根据2.3节的讨论，这样的修改并不会降低识别准确率。

图 3.2 卷积核的切分兼容

## 加速器整体结构设计

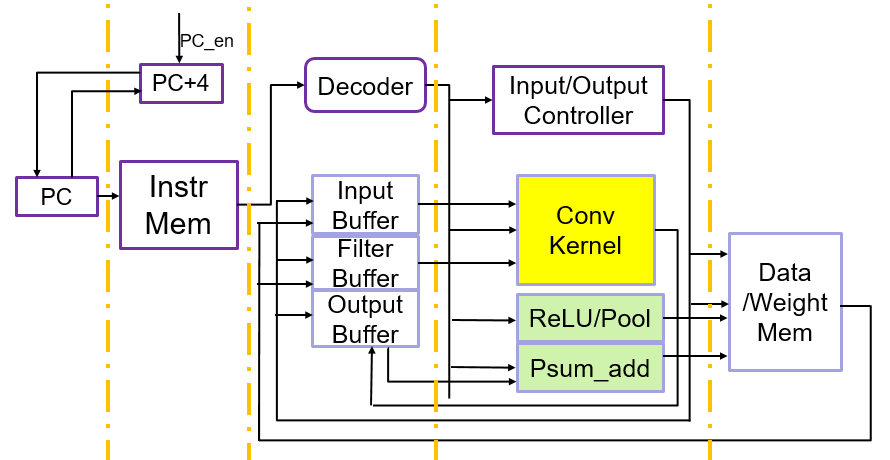
本节给出CNN加速器芯片的整体结构设计，以配合核心计算单元完成完整的卷积网络运算。图3.3给出了芯片整体结构图。整个芯片类似于流水线CPU的结构，深紫色边框的是控制模块，包括程序计数器PC、指令存储单元、译码单元、输入输出控制单元等，黄色的是3.1节具体介绍的核心计算电路，两个绿色框负责完成卷积之外的加法、ReLU和Pooling操作，其他淡紫色边框的是片上的输入输出缓存以及存储图像和卷积参数的Memory。

图 3.3 CNN加速器整体结构

图3.3黄色的Conv kernel中包含了两块相同结构的卷积核心电路并行运算，两者结果相加，并把结果写入output buffer。在遇到奇数个图像输入时，可以控制第二个卷积核心电路停止工作，以保证结果正确并降低功耗。一般来讲，AlexNet、VGG等网络除了第一层的RGB 3通道输入，其他的输入输出均为偶数[2][4][6]，因此两个卷积核都可以保持高效利用。

加速器芯片的执行过程也类似于传统的流水线CPU，由指令控制相应的计算和存储单元完成卷积运算，具体的控制指令设计在3.3节介绍。根据核心计算模块的需要精细设计了芯片各个模块之间的带宽，以保证计算模块尽可能高效率地运行。芯片上还有一个小的控制模块来与外界通信，控制整个芯片的时钟、工作状态、读入指令和存取数据，未在图3.3的流水线中画出。

## 加速器控制结构设计

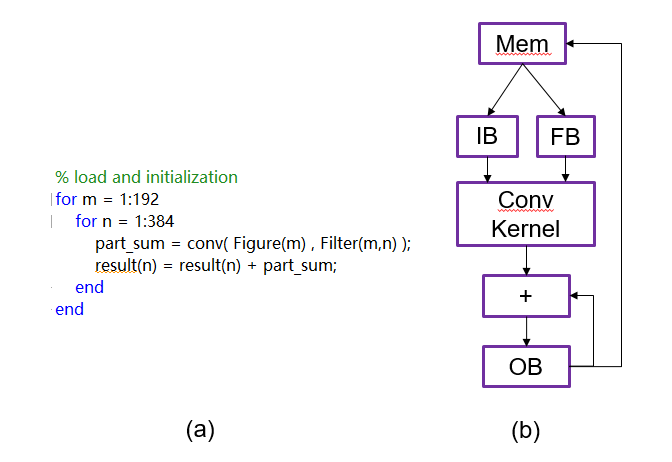
加速器的控制结构是否合理决定了核心计算单元能否保持最高效率的工作状态。尽可能减少乘法器的空置可以加快整个卷积网络的执行，提高效率，降低功耗。下面详细介绍指令控制设计。

图 3.4 CNN卷积层的Matlab代码和硬件执行流程图

### 加速器指令集设计

首先需要了解整个CNN算法的执行过程才能对指令做出很好的划分。图3.4(a)给出了一层典型CNN网络的Matlab代码，参考公式(1-1)，其中m、n分别表示输入图像和输出图像的数量。图3.4(b)显示了卷积操作在硬件加速器上执行的数据流图，其中IB/FB/OB分别是Input/Filter/Output Buffer，“+”表示部分和累加器。首先从片上Memory中取出数据放入图像和卷积核的缓冲区IB和FB，然后这些数据被放进核心计算电路进行卷积乘法和累加运算，输出的部分和在下方的加法器中和之前的部分和累加，之前的部分和存放在缓冲区OB里面。所有部分和累加执行完毕后，结果从OB写回到片上Memory。

指令集设计的目标是控制整个数据流图的正确运转，在此基础上提高硬件执行效率，并且尽可能使指令结构较为简单，降低指令结构自身带来的功耗、面积等开销。

基于硬件的具体实现，指令集结构主要分为四段的控制：从Mem 到IB/FB、从IB/FB经过Conv Kernel到部分和加法器之前、部分和加法器运算、从OB写回到Mem。根据这四段所需要对各个模块的配置参数，将指令设置为32bit，并且给出相应的控制字段，由译码器将各个控制位分配给各个模块，使其按照指令要求的方式完成数据流图的正确执行，实现CNN卷积操作。

ReLU/Pooling等操作所需要的指令类似，如ReLU需要在部分和累加完成之后进行，因此部分和加法器模块可以添加一个控制位和相应硬件结构实现ReLU。Pooling的实现则需要新加一类指令，考虑到Pooling操作的像素块大小不定以及兼容设计的核心电路结构，Pooling通过依次比较两个点的大小得到最后的Max结果。因此Pooling的操作类似于RISC指令集下的“>”实现，而由于Pooling对应的计算量相对于整个网络微乎其微，因此对Pooling的灵活性做了提升而没有提高执行速度。

图 3.5 指令集具体结构示意图

### 算法到指令的映射

图3.5给出了指令集具体结构的示意图，为了清晰展示而做了简化，实际指令相对要复杂一些。一共32bit的指令，第一列表示3.3.1节划分的指令类型，前两位标记了指令所属的类型，后面三列表示指令需要配置的各个模块的参数。基于这样的指令结构，参照图3.4给出的代码和数据流图，可以给出实际的CNN网络在硬件加速器上的实现。

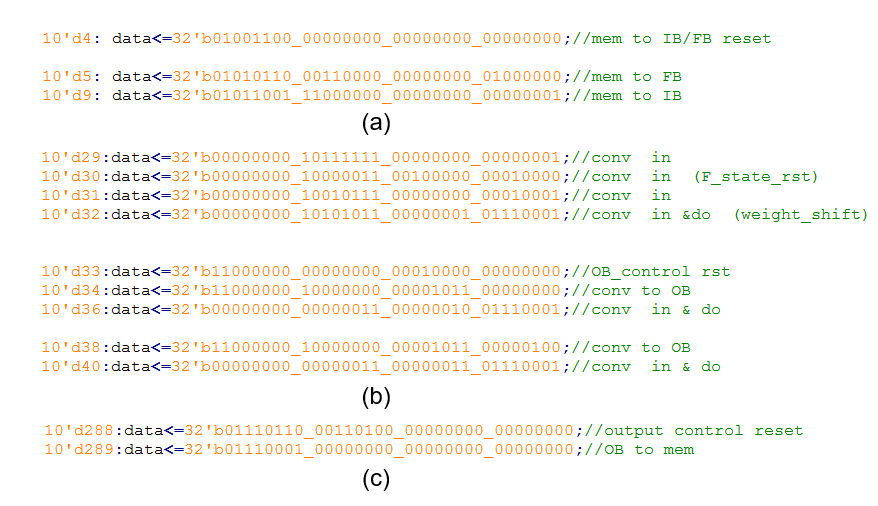
图3.6展示了3段写在ROM中的实际的指令代码。图3.6(a)(c)分别是从Mem中读取数据和最后数据写回Mem的指令。图3.6(b)给出了卷积核心部分的计算指令和部分和加法器的控制指令，在数据流图中这两部分存在串行的依赖关系，因此在流水线执行时这两部分指令交替出现。

图 3.6 加速器实际指令代码

## 加速器性能分析

本节将设计的CNN加速器芯片和之前的硬件加速器进行对比，给出所设计的加速器芯片在面积、性能等方面的理论分析。

### 硬件资源分析

本部分不涉及实际芯片面积功耗的对比，仅从存储大小等参数进行比较。首先给出各部分的存储配置，Mem采用64KB的SRAM，IB/FB/OB一共约4.5KB的Register。核心单元采用了8bit的乘法器，相比于MIT16bit的乘法器，不仅降低了乘法器功耗，并且数据位宽降低一半，理论上输入和输出带宽和存储都可以降低到原来的一半。

由于使用了8bit乘法器，且卷积核和图像数据复用更加充分，相对于MIT 108KB的SRAM和75.3KB的Spad，片上SRAM和Register分别减少了40.7%和94.0%，从而带来对应的面积和功耗节省[10]。

### 执行性能分析

本部分根据指令集设计，结合实际的CNN网络，给出核心计算资源的理论利用效率，从而给出芯片在特定频率下的运行速度。

我们对于核心计算单元的控制指令设计比较特别，由于采用“1 Figure-n Filter”的数据复用，因此对于放入核心计算单元的一个图像，需要n个周期来轮换与之对应的卷积核(当前设计n=8)，但只需要在第一个周期发出一条指令就可以控制8个卷积核的轮换。同样，部分和加法模块也只需要一条指令即可控制核心计算模块给出的8个周期的部分和。因此在剩余7个周期指令可以控制其他模块工作，如pooling、bias等。

以AlexNet第5层为例，输入图像192个，输出图像256个，大小均为13x13，卷积核大小为3x3[2]。首先将所有图像存到片上，需要个时钟周期。对于每个核心计算单元，载入8个卷积核各需要个数据，每个数据8bit。设输入带宽为16bit，则需要的时钟周期为。每个卷积核只需要载入一次。每个图像输入后与8个卷积核卷积的时钟周期为，因此载入卷积核的时钟周期数可以隐藏在当前图像与卷积核计算的时钟周期内。每个图像载入核心计算单元的次数为256/8=32次，本层CNN网络卷积计算所需总时钟周期为。连续执行时，图像载入和导出的时间也可以隐藏在计算所用时钟周期内。假设运行频率为200MHz，则运行时间为1.72ms。考虑与MIT一样执行四帧图像的性能对比如表3.1所示：

表 3.1 本文与MIT性能比较

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 时间  (ms) | MAC | 乘法器数量  (硬件利用率) | 片上存取  (MB) | 片外存取  (MB) |
| MIT[10] | 10.5 | 299M  MAC | 156(93%) | 24.9 | 1.3 |
| 本文 | 6.88 | 234(100%) | 4.67 | 0.65 |

由表3.1中数据，本文的设计相对于MIT在执行时间上减少了34.5%（即速度提高了52.6%），速度达到86.9GOPS。这主要是由于两者的乘法器规模导致的(MIT的乘法器数量比本文少28.2%)，乘法器运行时的工作效率也有一定的影响，MIT的乘法器运行占用率为299M/(156\*200MHz\*10.5ms)=91.2%，而我们的乘法器运行占用率接近13/14=92.9%。而由于我们的设计更偏重对3x3卷积核的优化，因此硬件利用率达到了100%。由于更加合理的数据复用设计，部分和的累加在部分和加法器中完成，片上存取降低了81.2%，由于采用8bit乘法，降低了数据位数，片外存取降低了50%[[3]](#footnote-3)。

# 第4章 实验验证

本章基于第3章提出的硬件结构完成设计，给出完成结果的实际验证，包括实际功能验证以及是否可以得到3.4节理论分析的指标的测试，对比目前最好的几种卷积神经网络加速器硬件，验证所设计硬件结构的性能参数。

## 功能验证

本节给出所提出的硬件设计的实验与功能验证。

硬件加速器通过verilog代码具体实现。通过一个顶层的测试模块，控制输入一层CNN网络数据和相应的指令，可以测试整个芯片的实际功能。考虑到对于设计的硬件没有现成的编译器，相应的指令通过人工编译方式实现，因此测试CNN网络规模不能太大，但必须包含卷积、ReLU、Pooling等操作。

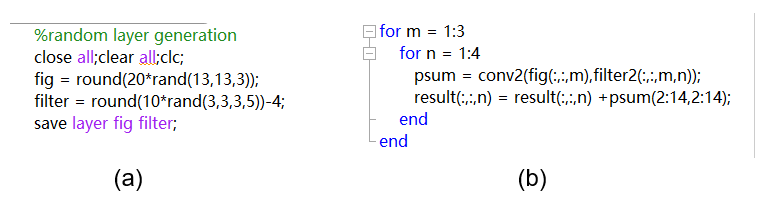
图3.7给出了CNN网络的生成与执行的Matlab代码，用来生成参数，并且与硬件执行得到的结果进行对比。(a)中随机生成的网络包含了一定数量的负数，用来检验乘法器对于负数运算是否正确。

图 4.1 CNN网络的生成与执行Matlab代码

图4.2和图4.3分别给出了硬件加速器的Modelsim仿真结果和Matlab的软件计算结果。图4.3中显示出的12行的结果依次对应图4.2从黄色线开始的12列数据。在硬件指令中对于前7列做了ReLU操作，因此前7列所有的负数结果在硬件执行时变为0，验证了ReLU操作的硬件实现正确性。图4.2显示的结果是OB向Mem输出的结果，即可以认为是整个网络完成执行之后的最后结果。

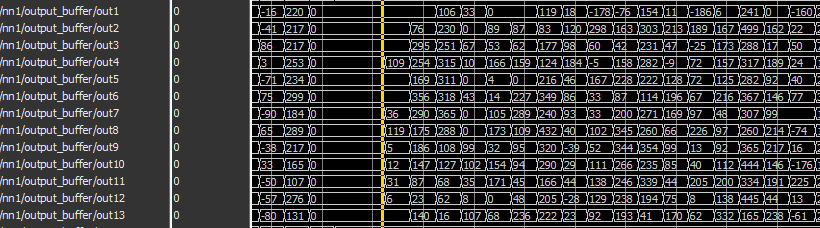


图 4.2 CNN硬件加速器执行结果

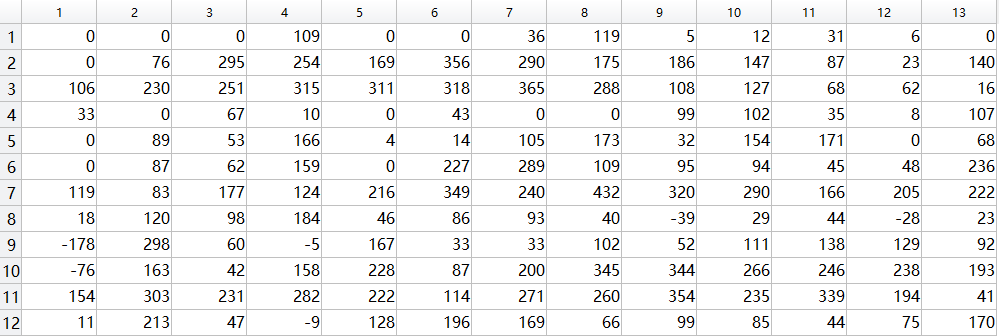
Pooling操作的验证如图4.4所示。我们对图4.3中第2行和第3行的数据两两做了max pooling[[4]](#footnote-4)。图4.4中的in1~in13是输入的两列数据，在时钟的上升沿依次对数据两两比较，结果记录在p1到p13中。仿真结果正确，验证了pooling模块的正确性。

图 4.3 Matlab软件执行结果

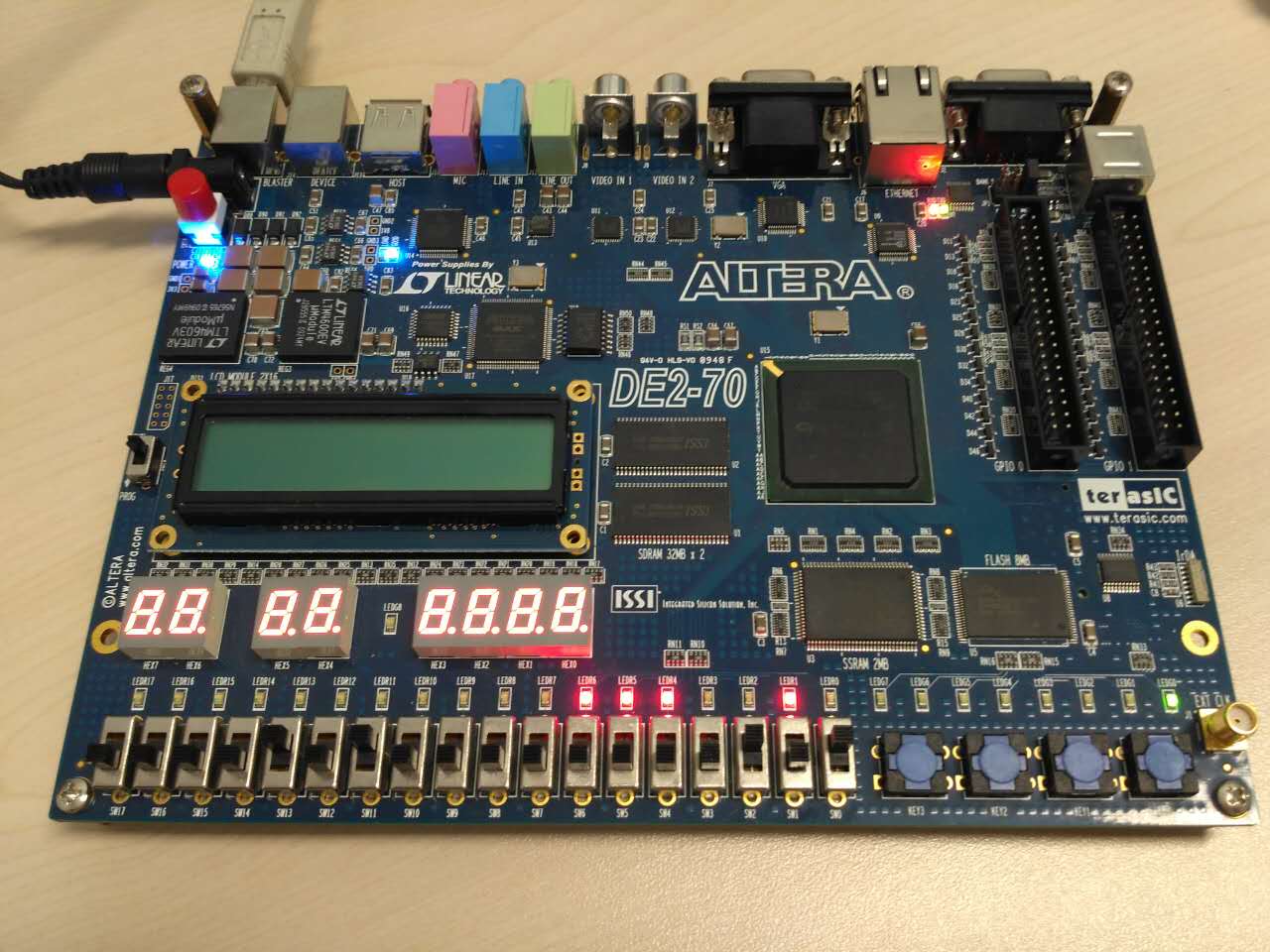
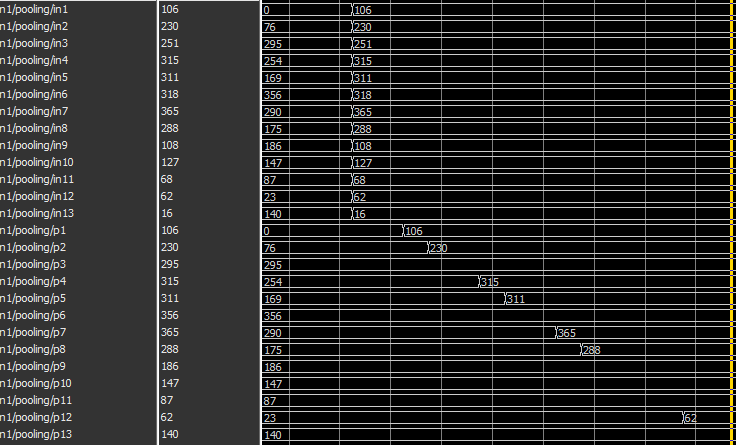
芯片与片外的数据读写通过一个小的读写控制器实现，输出带宽为8bit，外部控制单元可以将加速器执行得到的结果取出用于验证以及后续计算。我们通过ALTRA公司的FPGA开发板DE2-70进行了实际测试，如图4.5所示。选择开关SW16~SW10是地址位，每一个地址存放15个8bit数据，SW3~SW0表示数据的编号，图4.5中要求输出地址位“0001010”处的第“0101”个数据，结果为“01110010”，即十进制数114，与Matlab和Modelsim仿真中对应位置的数据相同(图4.3第11行第6列)。每次更改地址位和数据编号并且使用reset按键，可以更新结果。我们对于CNN网络输出的所有4个13x13的输出图像的每一个数据以及加入测试性的ReLU、Pooling、bias等操作的结果进行了验证，所有结果均正确。

图 4.4 Max pooling 的Modelsim仿真结果

图 4.5 FPGA芯片测试验证

## 性能比较

本节给出芯片版图和实际的功耗面积参数以及基于芯片设计和硬件参数对于实际运行性能的理论分析，并与其他CNN硬件加速器进行比较。

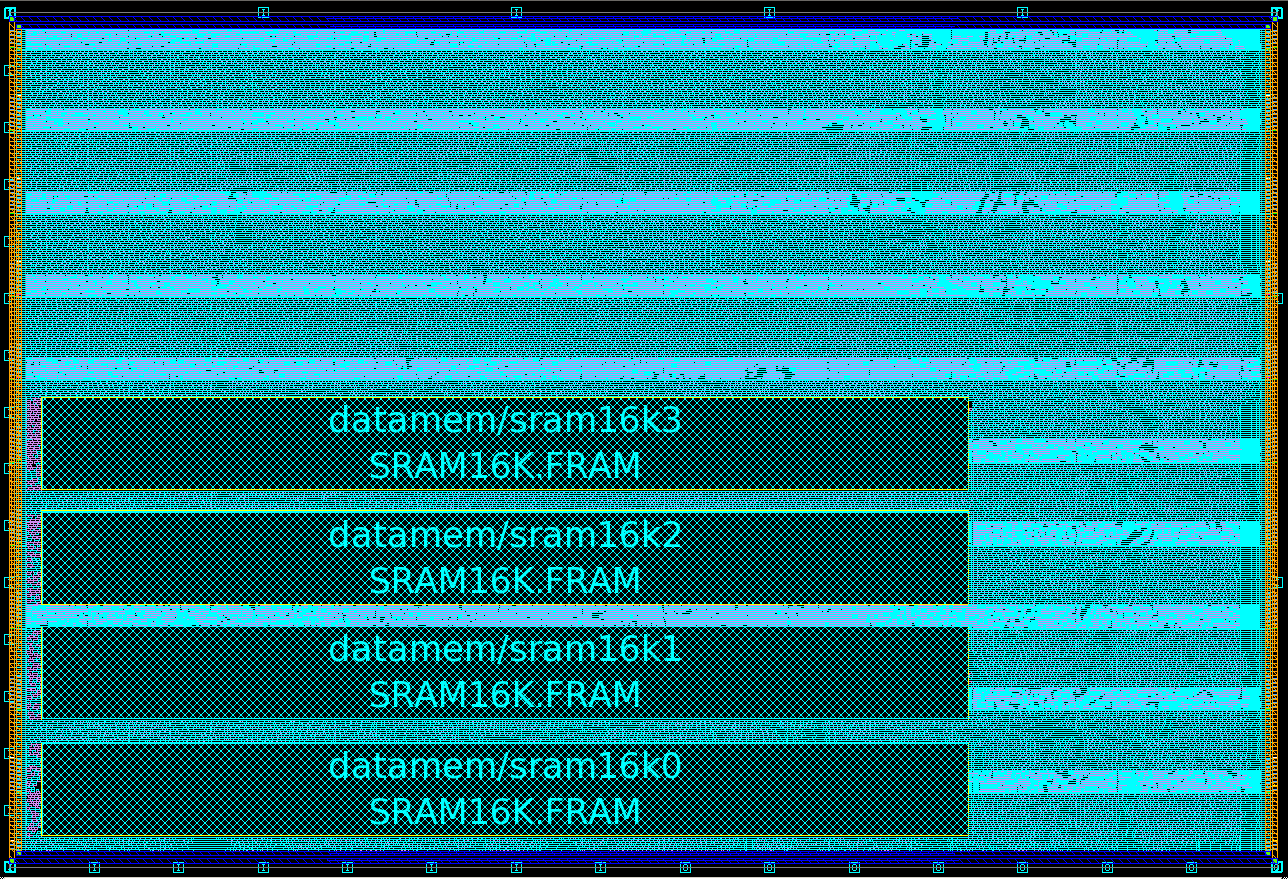
利用synopsis公司的DC综合工具和后端设计工具得到相关的硬件参数和最终的版图设计。版图设计如图4.6所示，图4.7列出了相应的功耗、延时和面积参数(不包含图4.6中片上SRAM存储部分)。片上每块SRAM在200MHz频率下的平均功耗为9.64mW。

图 4.6 后端芯片版图

MIT的芯片大小为3.5mm\*3.5mm，运行频率200MHz，执行AlexNet第5层卷积层的功率为236mW，其他数据列在表3.1中[10]。

图4.6版图的大小为1mm\*1.5mm，相对于MIT的芯片面积减小了87.8%，但执行速度却提高了52.6%。芯片的最高工作频率为，而芯片在200MHz频率下总的平均功耗为77.69mW，相比MIT降低了67.1%[[5]](#footnote-5)。而且由于芯片设计大大降低了数据存取数量，也会对于实际运行的功耗有很大降低。在3.4节的运行情况下，芯片的能量效率为2\*299M/6.88ms/77.69mW=1.12TOPS/W[[6]](#footnote-6)，是MIT(241.3GOPS/W)的4.6倍，考虑最大功耗的情况，我们的能效是MIT的至少2.4倍。

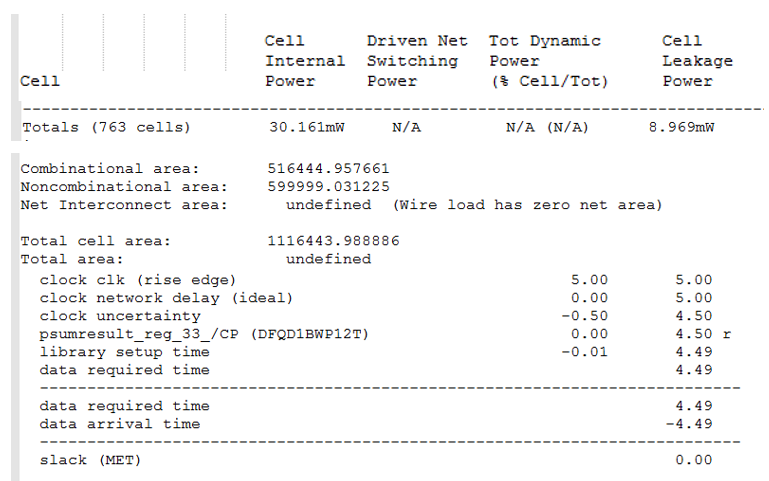


图 4.7 不含SRAM的芯片功耗、面积和延时

# 第5章 结 论

卷积神经网络是目前对于图像识别等应用准确度最高的算法，但大量的计算和存取操作使其在CPU和GPU上的执行时间与能耗大大增加，限制了在各种移动设备和低功耗设备上的应用，而目前的卷积神经网络加速器则有进一步的提升空间。

本文提出了一种新的卷积神经网络硬件加速器结构，充分利用了卷积神经网络的结构特点，通过卷积核心电路、整体芯片结构和指令集体系的设计，实现了高性能低功耗的神经网络加速器。相对于目前最好的通用卷积神经网络加速器芯片，我们将芯片整体的功耗降低了67.1%，面积减小了87.8%，而速度提高了52.6%，能量效率则提升了至少2.4倍。实际运行的片上数据存取降低了81.2%，片外存取降低了50%。

本文通过软件仿真和FPGA测试验证了设计芯片的功能。未来可以通过编译器优化设计进一步提高执行效率，并且可以通过数据压缩等方式降低片外数据读取和功耗。

# 插图索引

[图 1.1 卷积神经网络典型结构[2] 2](file:///C:\Users\xzz\Desktop\023%20岳金山%202012011029.docx#_Toc453775486)

[图 2.1 1 Figure-1 Filter 层次的数据复用 7](file:///C:\Users\xzz\Desktop\023%20岳金山%202012011029.docx#_Toc453775479)

[图 2.2 ASSCC 2015的硬件加速器设计[8] 10](file:///C:\Users\xzz\Desktop\023%20岳金山%202012011029.docx#_Toc453775480)

[图 2.3 MIT的Eyeriss硬件加速器设计[10] 10](file:///C:\Users\xzz\Desktop\023%20岳金山%202012011029.docx#_Toc453775481)

[图 2.4 ImageNet 各年最佳算法中的卷积核比例 12](file:///C:\Users\xzz\Desktop\023%20岳金山%202012011029.docx#_Toc453775482)

[图 3.1 加速器核心电路结构 13](file:///C:\Users\xzz\Desktop\023%20岳金山%202012011029.docx#_Toc453775468)

[图 3.2 卷积核的切分兼容 14](file:///C:\Users\xzz\Desktop\023%20岳金山%202012011029.docx#_Toc453775469)

[图 3.3 CNN加速器整体结构 15](file:///C:\Users\xzz\Desktop\023%20岳金山%202012011029.docx#_Toc453775470)

[图 3.4 CNN卷积层的Matlab代码和硬件执行流程图 16](file:///C:\Users\xzz\Desktop\023%20岳金山%202012011029.docx#_Toc453775471)

[图 3.5 指令集具体结构示意图 17](file:///C:\Users\xzz\Desktop\023%20岳金山%202012011029.docx#_Toc453775472)

[图 3.6 加速器实际指令代码 18](file:///C:\Users\xzz\Desktop\023%20岳金山%202012011029.docx#_Toc453775473)

[图 4.1 CNN网络的生成与执行Matlab代码 21](file:///C:\Users\xzz\Desktop\023%20岳金山%202012011029.docx#_Toc453775454)

[图 4.2 CNN硬件加速器执行结果 22](file:///C:\Users\xzz\Desktop\023%20岳金山%202012011029.docx#_Toc453775455)

[图 4.3 Matlab软件执行结果 22](file:///C:\Users\xzz\Desktop\023%20岳金山%202012011029.docx#_Toc453775456)

[图 4.4 Max pooling 的Modelsim仿真结果 23](file:///C:\Users\xzz\Desktop\023%20岳金山%202012011029.docx#_Toc453775457)

[图 4.5 FPGA芯片测试验证 23](file:///C:\Users\xzz\Desktop\023%20岳金山%202012011029.docx#_Toc453775458)

[图 4.6 后端芯片版图 24](file:///C:\Users\xzz\Desktop\023%20岳金山%202012011029.docx#_Toc453775459)

[图 4.7 不含SRAM的芯片功耗、面积和延时 25](file:///C:\Users\xzz\Desktop\023%20岳金山%202012011029.docx#_Toc453775460)

# 表格索引

[表 1.1 ILSVRC中识别精度最好的几种CNN算法运算量与精度 3](#_Toc453775543)

[表 1.2 CPU、GPU、DianNao执行VGG的性能功耗对比 4](#_Toc453775544)

[表 2.1 VGG-A结构参数 6](#_Toc453775537)

[表 2.2 典型神经网络加速器特征和性能参数 9](#_Toc453775538)

[表 3.1 本文与MIT性能比较 20](#_Toc453775529)

# 参考文献

[1] The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things, EMC Digital Universe with Research & Analysis by IDC, April 2014 http://www.emc.com/leadership/digital-universe/2014iview/index.htm

[2] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[3] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. arXiv preprint arXiv:1512.03385.

[4] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[5] http://image-net.org/challenges/LSVRC/

[6] Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In Computer vision–ECCV 2014 (pp. 818-833). Springer International Publishing.

[7] Chen, T., Du, Z., Sun, N., Wang, J., Wu, C., Chen, Y., & Temam, O. (2014, February). Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. In ACM Sigplan Notices (Vol. 49, No. 4, pp. 269-284). ACM.

[8] Hong, I., Park, S., Park, J., & Yoo, H. J. (2015, November). A 1.9 nJ/pixel embedded deep neural network processor for high speed visual attention in a mobile vision recognition SoC. In *Solid-State Circuits Conference (A-SSCC), 2015 IEEE Asian* (pp. 1-4). IEEE.

[9] Sim, J., Park, J. S., Kim, M., Bae, D., Choi, Y., & Kim, L. S. (2016, January). 14.6 A 1.42 TOPS/W deep convolutional neural network recognition processor for intelligent IoE systems. In 2016 IEEE International Solid-State Circuits Conference (ISSCC) (pp. 264-265). IEEE.

[10] Chen, Y. H., Krishna, T., Emer, J., & Sze, V. (2016, January). 14.5 Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. In *2016 IEEE International Solid-State Circuits Conference (ISSCC)* (pp. 262-263). IEEE.

[11] Qiu, J., Wang, J., Yao, S., Guo, K., Li, B., Zhou, E., ... & Wang, Y. (2016, February). Going deeper with embedded fpga platform for convolutional neural network. In Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (pp. 26-35). ACM.

[12] Du, Z., Fasthuber, R., Chen, T., Ienne, P., Li, L., Luo, T., ... & Temam, O. (2015, June). ShiDianNao: shifting vision processing closer to the sensor. In Proceedings of the 42nd Annual International Symposium on Computer Architecture (pp. 92-104). ACM.

[13] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., ... & Darrell, T. (2014, November). Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the ACM International Conference on Multimedia (pp. 675-678). ACM.

[14] Arel, I., Rose, D. C., & Karnowski, T. P. (2010). Deep machine learning - a new frontier in artificial intelligence research [research frontier]. IEEE Computational Intelligence Magazine, 5(4), 13-18.

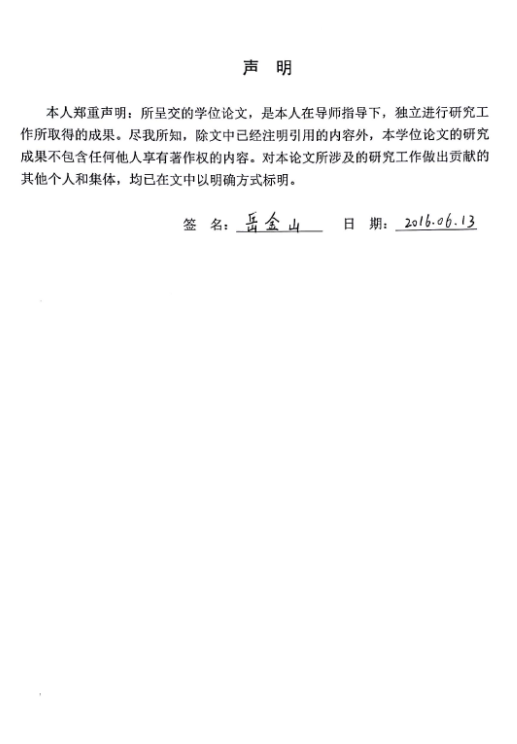
# 致 谢

首先衷心感谢刘勇攀副教授对我毕业设计课题的悉心指导。刘勇攀老师严谨的科研态度、敏锐深入的思维和丰富的研究经验给了我莫大的帮助。非常荣幸能够在您的引领下开启我的科研生涯。

感谢汪玉老师和郭开元师兄等在算法方面提供的无私支持，感谢苏放师兄在后端仿真方面的耐心指导，感谢王智博师兄、袁哲师兄和郭庆伟师兄等在各方面的帮助，感谢实验室其他师兄在各个方面给予的无私帮助。你们解答了我的许多疑问，你们的帮助让我能够顺利完成毕业设计课题。

感谢开题答辩和中期答辩的各位老师，你们的建议给了我很多启发，感谢魏特辅导员和熊兵老师，感谢你们认真负责的态度，也非常感谢各位同学在我毕业设计期间的帮助。

感谢我的父母和朋友给我的督促、鼓励和支持。感谢你们在我困惑的时候给我精神上的安慰和鼓励，让我能始终保持积极高效的工作状态，也让我在毕业设计期间的生活更加丰富多彩。



# 附录A 外文资料的书面翻译

移动视觉识别芯片中用于高速视觉感知的1.9nJ/像素的嵌入式深度神经网络处理器

摘要

本文提出了一种高能效的深度神经网络(DNN)处理器，用于移动视觉芯片中的高速视觉感知引擎。这种嵌入式深度神经网络能够快速找到可能的目标物体所在的像素范围，减少视觉芯片大约70%的工作量。与之前的视觉感知相比，深度神经网络的视觉感知减少了90%的执行时间，因此带来了73.4%的整体目标识别时间的减少。在DNN处理器中实现了高并行度的200路处理单元(PE)，二维的滑动图片窗结构，实现了3ms的DNN视觉感知延时。并且，本文提出了双路模式的PE集成可以用来做深度神经网络DNN和多层感知机(MLP)，以共享相同硬件结构，实现高的能量利用效率。因此，本文的工作实现了1.9 nJ/像素的能量效率，是目前最好的视觉感知加速器的7.7倍。

A.1 引言

最近，深度神经网络(DNN)成为热门的研究方向，因为其有效的识别准确度而被应用于各种视觉识别的应用中。即使DNN需要很长的处理时间和很高性能的计算机，它的精细设计的片上系统解决方案使即使是智能设备上的实时处理成为可能。

除了基于硅的解决方案，视觉感知算法（图A-1）也被成功用于目标识别，通过限制只分析可能包含目标的区域来减少能量消耗和处理时间。之后只有有限数量的被选中区域会被拿来做精细的目标识别。这样的处理方式显著降低了约70%的工作量。

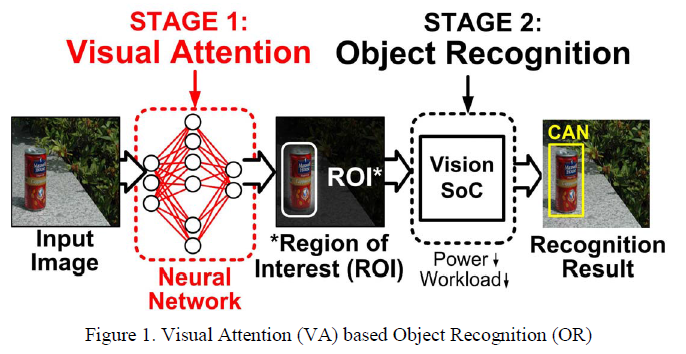
目前存在两种类型的视觉感知方法：自底向上和自顶向下的方法。自底向上的方法通过细胞神经网络硬件结构来加速，使用低层次的特征，包括颜色、强度和方向，来选择特征区域作为可能存在目标的区域。然而，自底向上的方法精确度不高，特别是当背景的聚类特征比目标更加显著时，精确度会降低。相反，自顶向下的方法使用目标的信息而不是图片的特征来找到可能存在目标的区域。例如基于SIFT的目标识别预处理能够快速地在320x240大小的图像中执行，这样的320x240的图像来自对640x480的原始输入的快速可能目标范围近似。因此，即使是在有特征背景的情况下,自顶向下的视觉感知也展示了高精确度的视觉感知。

图 A-1 基于视觉感知的目标识别

然而这样基于SIFT的自顶向下的视觉感知要求大量的计算消耗，在Intel i5 2.8GHz的CPU中的执行时间约为400ms，这在要求处理延时<33ms的移动平台上是不够的。除此之外，基于SIFT的自顶向下的视觉感知要求超过4MB的数据内存来存储高位的SIFT特征，这增加了内存存取时间和移动视觉系统的能耗。

本文提出了一种快速的容错性强的基于DNN的视觉感知系统，它的精细的高能效硬件加速器有以下三种关键特征：1）卷积神经网络CNN，目前最流行的DNN算法，用来实现新的 自顶向下视觉感知，不仅将约400ms的执行时间减少了90%，并且不需要外部存储。2）高并行度200路处理单元PE和新的二维图像滑动窗结构用于快速的视觉感知加速器。3）双路PE集成和3层加法树设计来实现高能量效率。因此，本文提出的片上DNN处理器在65nm CMOS逻辑工艺下实现，仅仅花费3s用于DNN的视觉感知加速器，仅用了8KB的特征数据内存和平均9mW的能量消耗。

本文其他部分组织如下。A.2介绍了提出的基于DNN的视觉感知模型和仿真结果。A.3介绍了提出的DNN处理器结构，具有200路处理单元和双模式的PE集成和加法树。实现情况和测量结果显示在A.4部分。A.5对本文做了总结。

A.2 基于DNN的自顶向下视觉感知

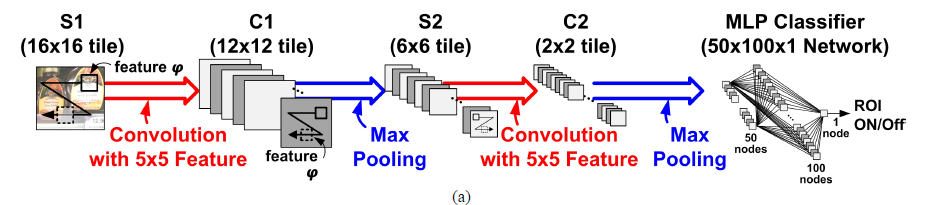
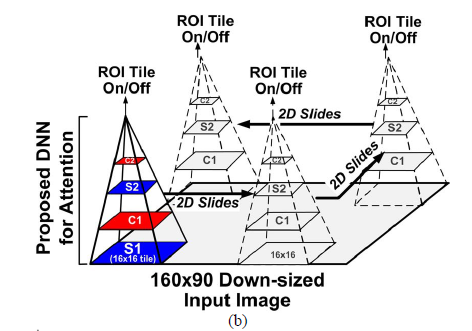
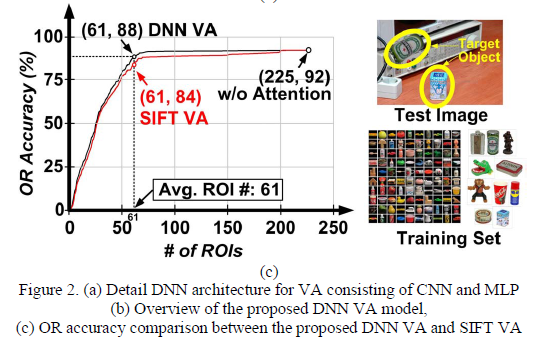
图A-2(b)显示了基于DNN的自顶向下的视觉感知的整体框架。DNN视觉感知用了含有两个卷积层S1和S2的CNN架构，有两个最大值合并(pooling)层C1和C2。S1层的输入为16x16的图像块，DNN的输出确定了输入的16x16图像块是否被选为可能存在目标的区域。输入的16x16图像块位于从640x360输入图像降采样得到的160x90图像块中。因此当16x16的图像块被选为可能存在目标的区域后，相当于在原始图像的64x64的图像块中进行目标识别。

图 A-2 (a)用于CNN和MLP的DNN具体结构 (b)提出的DNN VA模型的整体结构

(c)DNN VA和SIFT VA的目标识别精度对比

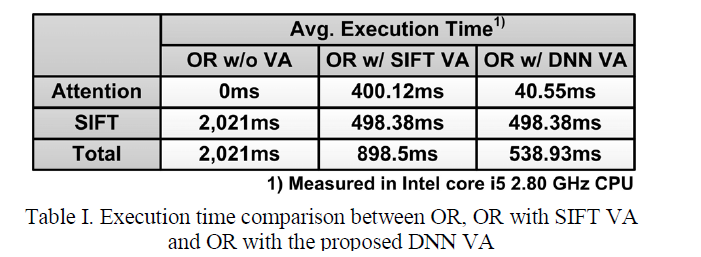
包含CNN的神经网络架构的细节和多层感知分类显示在图A-2(a)中。首先，执行输入S1图像块和5x5的S1特征之间的卷积。在CNN中使用的特征从离线的稀疏编码表示中得到，其中特征是构建输入图像的基。由于稀疏的特征相互正交，它可以实现很高的视觉感知精度，即使是用<8KB的离线5x5特征训练集合。在12x12的卷积图像之后，C1层从切分的2x2的像素块中选择最大值取出组成6x6的S2图像块。之后在S2层再一次执行卷积操作，来构建C2图像，C2取最大值用于之后的S层和C层。最后，50x100x1的MLP分类器用来确定最后的可能存在目标区域选择。

表 A-1 OR、带SIFT VA的OR和带DNN VA的OR执行时间对比

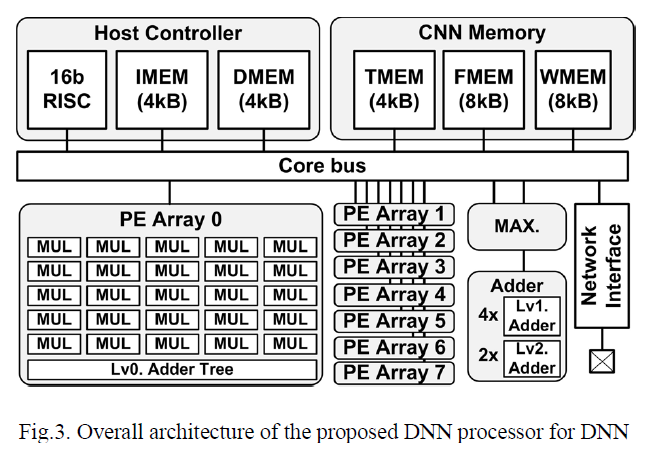
图A-2(c)显示了目标识别精确随存在目标选择区域大小变化的数据图。一共约8000张图片用来做CNN和MLP的离线学习，一共360张图片用来做测试集，检测综合在图片中的目标物体，有6种不同的大小范围和12种不同的旋转方向，如图A-2(c)右侧所示。从仿真结果来看，本文提出的DNN视觉感知模型选择了一共225个区域中的27%作为可能存在目标物体的区域用来做快速目标识别，而仅仅减小了约4%的整体目标识别精度，但是比基于SIFT的自顶向下视觉感知提高了4%。表A-1显示了没有自顶向下的视觉感知、使用SIFT的自顶向下视觉感知和DNN视觉感知执行时间对比。结果显示DNN的视觉感知相对之前基于SIFT的视觉感知减小了90%的执行时间，同时减小了73.4%的原始目标识别算法的执行时间。

图 A-3 提出的DNN处理器的整体结构

A.3 片上DNN处理器

A.3.1 整体结构

图A-3展示了片上DNN处理器的硬件结构。16b的RISC是主控制器。有8个PE阵列和200个乘法器单元来实现8个图像块的卷积。每个PE阵列有5x5大小的特征。PE中的加法树在一个周期内把5x5的结果加起来。为了支持最大值合并（pooling）操作，有一个由8个PE阵列共享的最大值ALU。三级加法树支持用同样的硬件实现图A-2（a）中的CNN和MLP操作，实现了高能效。

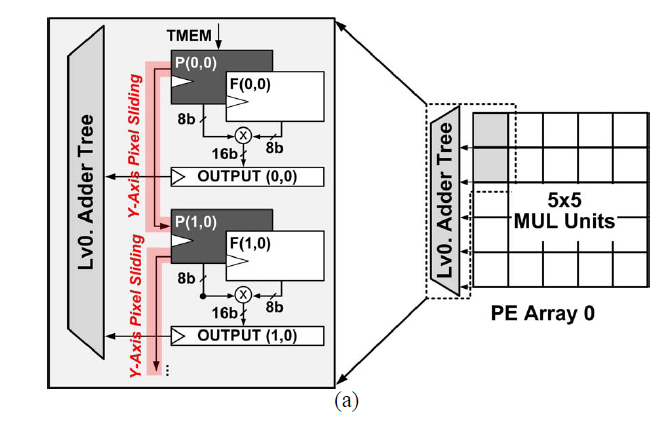
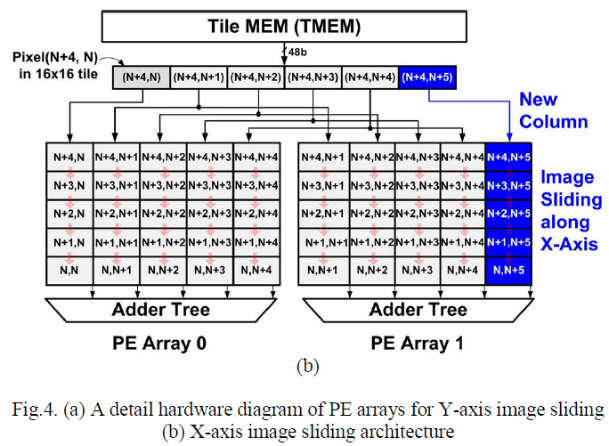
 有3个SRAM存储相关的DNN数据。图像块存储TMEM存储多个16x16的输入图像块，而8KB特征存储RMEM存储所有的离线训练的5x5特征集。因为提出的DNN用了稀疏编码，8kB的特征集可以保证目标之别的精度。8KB的权重存储WMEM存储用来计算MLP的权重。片上DNN处理器的网络接口用通用协议读取16x16的图像块输入。而且发送最后选择的16x16的可能存在目标区域块给其他的IP核，用于进一步的目标识别。

图 A-4 (a)沿Y方向图像滑动的PE阵列的硬件细节流图 (b)X方向图像滑动结构

A.3.2二维图像滑动窗结构

图A-4(a)和图A-4(b)显示了PE阵列的细节结构，支持二维的图像滑动窗，能够快速载入5x5的图像块。在y轴滑动时，一个8b的像素寄存器连接到相邻的靠近y轴的像素寄存器，如图A-4（a）。因此，只有第一行的5x5单元需要从TMEM中读取像素，减少了图像块载入的消耗。图A-4（b）显示了图像滑动窗沿x轴滑动的情况。当卷积操作结束之后，48b的6个像素数据从TMEM中读取出来。开始的5个像素放入PE0和PE1，剩下的一个像素放入PE1来做X轴滑动，结果是临近x轴的5x5的块可以在一个周期内放到2个PE中。

A.3.3双模式(CNN,MLP)PE阵列集成

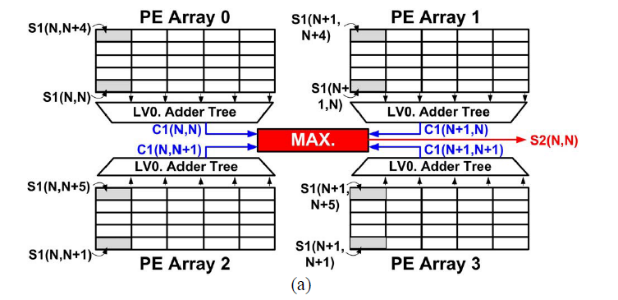
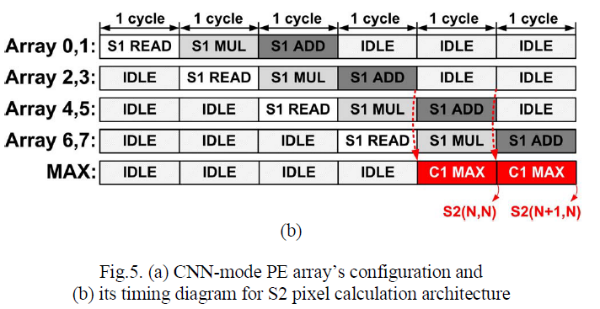
图A-5(a)显示了CNN模式的硬件集成，有4层加法器树。图A-5(a)中的4个PE阵列通过4个相邻的5\*5块并行做卷积操作。2x2的C1值从输出的4个加法器中取最大值用在S2层的计算中。

图 A-5 (a)CNN模式PE阵列的集成 (b)S2像素计算结构的时序图

图A-5(b)显示了8个PE计算两个S2的流水线的时间图表。第一个周期，阵列0和1的第一行寄存器从TMEM读入6个像素数据，其他行从上一行读入像素数据。下一周期，阵列0和1计算5x5的图像块和5x5的特征之间的乘法操作。而阵列2和3从TMEM读取像素数据。第三个周期，阵列0和1用加法器计算两个C1输入值，阵列2、3执行乘法，阵列4、5读取像素数据。在阵列2和3计算出C1数据后，从阵列0-3的4个数据求最大值计算S2层的输入。因此DNN处理器用6个周期计算2个S2像素点。

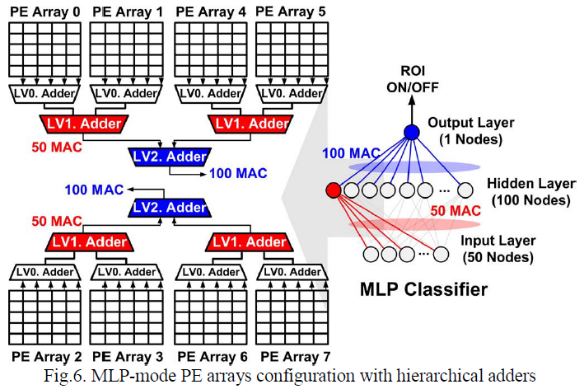
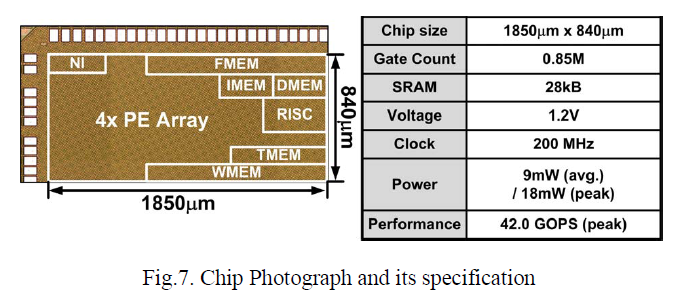
图A-6显示了MLP模式下3层加法器硬件集成。50x100x1的隐藏单层节点输出从两个PE阵列中用级联的0层和1层加法器得到。因此用8个PE可以并行得到4个隐藏节点的值。最后一层输出用4个PE和级联的0,1,2级加法器得到。因此两个输出节点可以由8个PE组成的两个MLP分类器得到。

图 A-6 芯片和具体参数

图 A-7 MLP模式含多层加法器的PE阵列的集成

A.4 完成结果

图A-7显示了片上DNN处理器，65nm工艺，0.85M NAND2等价逻辑门和28KB SRAM。总平均能耗是9mW，峰值功耗18mW，频率200MHz，性能为42.0GOPS，电压1.2V，面积1.554mm2。

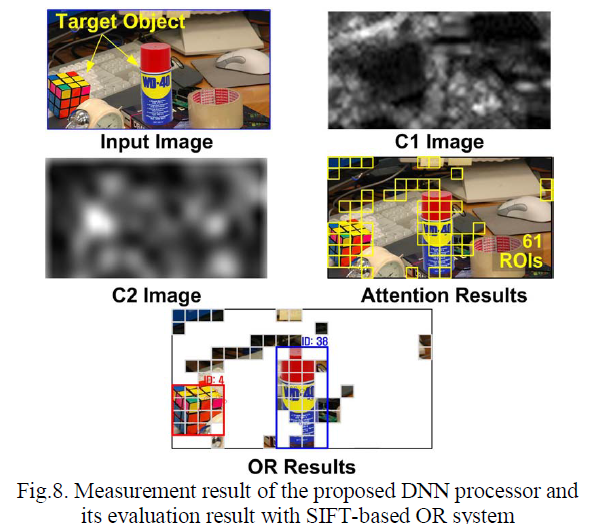
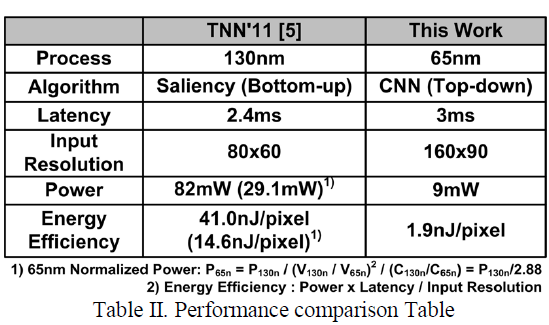
图A-8显示了DNN处理器的测量结果，目标检测评估结果集成在多核时间处理器中。C1图像显示第一个pooling层用了50个稀疏特征，第二个pooling层用了250个稀疏特征。因此只有61个可能存在目标区域被选中。我们用多核处理器和全连接的目标识别算法验证了DNN视觉感知的性能。在目标区域识别选择之后，多核处理器从61个ROI中提取SIFT特征和数据库中的100个数据图像比较。DNN处理器选出的可能存在目标区域显示在图A-8中，目标检测成功快速检测到了目标物体。因此，实现了<33ms的目标识别时间减少了80%的分块计算。

表 A-2 性能比较表

图 A-8 DNN处理器的测量结果和基于SIFT的OR系统的分析结果

表A-2展示了目前最好的视觉感知硬件和自底向上的特征视觉算法的比较。品质因数FOM考虑了延时、输入图像分辨率、能量消耗和处理器归一化参数，用来比较能量效率。本文实现了1.9nJ/像素的能量FOM，比之前的工作小7.7倍。

A.5 结论

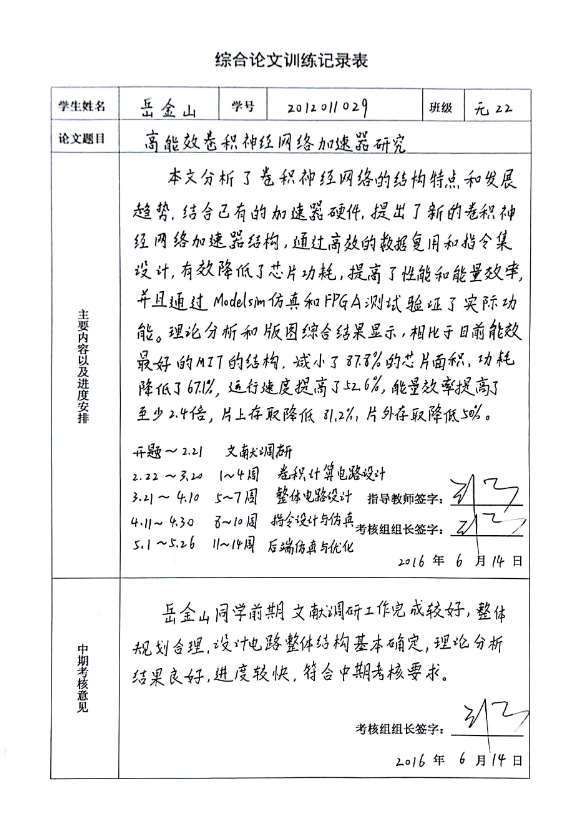
本文提出了一种高能效的片上DNN处理器，用于移动视觉芯片的高速的自顶向下视觉感知。我们开发了基于DNN的视觉感知方法，比之前的SIFT方法减少了90%的执行时间。而且我们完成了高并行度的200路PE阵列，二维图像滑动窗结构。而且双模式PE阵列和3级加法器支持CNN和MLP，实现了7.7倍的能量消耗降低。总的来说，我们提出了适应性强且快速的DNN处理器，成功实现了实时移动视觉处理器。

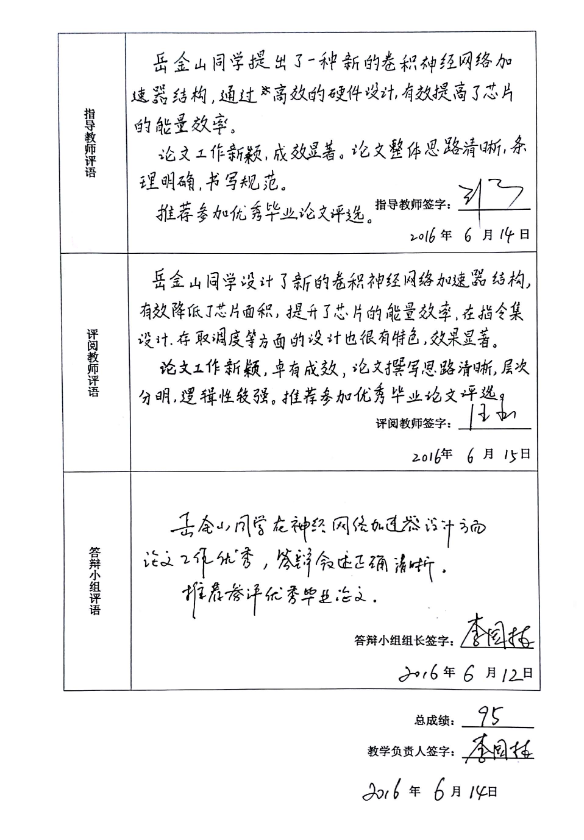
书面翻译对应的原文索引

Hong, I., Park, S., Park, J., & Yoo, H. J. (2015, November). A 1.9 nJ/pixel embedded deep neural network processor for high speed visual attention in a mobile vision recognition SoC. In *Solid-State Circuits Conference (A-SSCC), 2015 IEEE Asian* (pp. 1-4). IEEE.

# 在学期间参加课题的研究成果

Li, Hehe, Liu, Yongpan, Fu, Chenchen, Xue, Chun Jason, Xiang, Donglai, & **Yue, Jinshan**, et al. (2016). Performance-aware task scheduling for energy harvesting nonvolatile processors considering power switching overhead. *Proceedings of the 53rd Annual Design Automation Conference*. ACM.





1. 这时的结果还是部分和。参考公式(2-1)，需要把多个部分和加起来才是一层卷积网络的输出。 [↑](#footnote-ref-1)
2. 小型网络的性能一般均为理论值或峰值最优，实际运行的平均值要低于标称的速度和能效。 [↑](#footnote-ref-2)
3. 这里的片外存储并未计算卷积核参数。因为卷积核参数为440K，显然不可能全在片上存放，因此MIT的数据并不包含此项。但根据参数位宽，这一项所需要的存取数量理论上也是MIT的50%。 [↑](#footnote-ref-3)
4. 选取这两列仅为使验证结果更加直观，展示所有可能情况，并非实际CNN算法的pooling过程。 [↑](#footnote-ref-4)
5. 这里的功耗估算并不准确，因为SRAM的功耗与读写操作的数量有直接关系，而且我们在片上使用了clock gate的方式降低了平均功耗的仿真值，但总的平均功耗不会超过150mW。 [↑](#footnote-ref-5)
6. 考虑功耗为150mW，则能量效率为580GOPS/W，是MIT的2.4倍。 [↑](#footnote-ref-6)