Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks

Yu-Hsin Chen, *Student Member, IEEE*, Tushar Krishna, *Member, IEEE*, Joel S. Emer, *Fellow, IEEE*, and Vivienne Sze, *Senior Member, IEEE*

***Abstract*— Eyeriss is an accelerator for state-of-the-art deep convolutional neural networks (CNNs). It optimizes for the energy efficiency of the entire system, including the accelerator chip and off-chip DRAM, for various CNN shapes by reconfiguring the architecture. CNNs are widely used in modern AI systems but also bring challenges on throughput and energy efficiency to the underlying hardware. This is because its computation requires a large amount of data, creating significant data movement from on-chip and off-chip that is more energy-consuming than computation. Minimizing data movement energy cost for any CNN shape, therefore, is the key to high throughput and energy efficiency. Eyeriss achieves these goals by using a proposed processing dataflow, called row stationary (RS), on a spatial architecture with 168 processing elements. RS dataflow reconfigures the computation mapping of a given shape, which optimizes energy efficiency by maximally reusing data locally to reduce expensive data movement, such as DRAM accesses. Compression and data gating are also applied to further improve energy efficiency. Eyeriss processes the convolutional layers at 35 frames/s and 0.0029 DRAM access/multiply and accumulation (MAC) for AlexNet at 278 mW (batch size $N = 4$), and 0.7 frames/s and 0.0035 DRAM access/MAC for VGG-16 at 236 mW ($N = 3$).**

***Index Terms*— Convolutional neural networks (CNNs), dataflow processing, deep learning, energy-efficient accelerators, spatial architecture.**

I. Introduction

DEEP learning using convolutional neural networks (CNNs) [1] has achieved unprecedented accuracy on many modern AI applications [2]–[9]. However, state-of-the-art CNNs require tens to hundreds of megabytes of parameters on billions of operations in a single inference pass, creating significant data movement from on-chip and off-chip to support the computation. Since data movement can be more energy-consuming than computation [10], [11], the processing of CNNs has to not only provide high parallelism for high throughput but also optimize for the data movement of the entire system in order to achieve high energy efficiency. In addition, this optimization needs to adapt to the varying shapes of the high-dimensional convolutions in CNN.

To address these challenges, it is crucial to design a compute scheme, called a *dataflow*, that can support a highly parallel compute paradigm while optimizing the energy cost of data movement from both on-chip and off-chip. The cost of data movement is reduced by exploiting data reuse in a multilevel memory hierarchy, and the hardware needs to be reconfigurable to support different shapes. To further improve energy efficiency, data statistics can also be exploited. Specifically, CNN data contains many zeros. Techniques such as compression and data adaptive processing can be applied to save both memory bandwidth and processing power.

Previous work has proposed hardware designs for CNN acceleration [12]–[22]. However, most of them only have simulation results that are not verified by the measured results from fabricated chips; implementations using FPGA also do not reveal the actual throughput and energy efficiency of the architecture. A few efforts have demonstrated the measurement results of fabricated chips [23]–[25]. However, these works do not benchmark their implementations using widely used publicly available state-of-the-art CNNs, which is critical to the hardware evaluation. Specifically, Park *et al*. [23] propose a deep-learning processor for running both training and inference using an MIMD architecture, which was tested on a custom four-layer network using 5×5 filters. Cavigelli *et al*. [24] present a CNN accelerator for inference that is tested on a four-layer CNN using 7×7 filters. Sim *et al*. [25] demonstrate a CNN processor and only report the theoretical peak throughput along with the power measured on a CNN for the MNIST data set [26], which has storage and computation requirements that are orders of magnitude lower than the state-of-the-art CNNs. With the exception of [24], these works did not report the required DRAM bandwidth for the proposed compute schemes. It is not sufficient to look at the processor power alone, since DRAM access is one of the most important factors dictating the system energy efficiency.

In this paper, we have implemented and fabricated a CNN accelerator, called Eyeriss, that can support high throughput CNN inference and optimizes for the energy efficiency of the *entire system*, including the accelerator chip and

Manuscript received May 5, 2016; revised July 31, 2016; accepted September 28, 2016. Date of publication November 8, 2016; date of current version January 4, 2017. This paper was approved by Guest Editor Dejan Markovic.

Y.-H. Chen and V. Sze are with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

T. Krishna was with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA. He is now with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA.

J. S. Emer is with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA, and also with Nvidia Corporation, Westford, MA 01886 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSSC.2016.2616357

Eyeriss：一种用于深度卷积神经网络的节能可重构加速器

IEEE学生会成员 陈玉欣 、IEEE会员Tushar Krishna 、IEEE会士Joel S. Emer 、IEEE高级会员Vivienne Sze

***摘要* — Eyeriss是一款专为尖端深度卷积神经网络（CNN）设计的加速器。通过重新配置架构，该系统针对不同CNN形态优化了整体能效，涵盖加速器芯片与片外DRAM的协同运行。尽管CNN在现代AI系统中应用广泛，但其运算特性给底层硬件带来吞吐量与能效挑战——海量数据处理导致芯片内外数据传输能耗远超运算本身。因此，降低各类CNN形态的数据传输能耗成为实现高吞吐量与能效的关键。Eyeriss采用名为行稳态（RS）的数据流处理方案，在168个处理单元的空间架构上实现突破。RS数据流通过动态重构特定CNN形态的计算映射，最大限度本地复用数据以减少昂贵的DRAM访问等数据传输，从而优化能效。同时引入压缩与数据门控技术进一步提升能效。该加速器可实现卷积层处理速度达35帧/秒，且能耗为0。0029 AlexNet的DRAM访问/乘法和累加（MAC）为278mW（*批量N* = 4），0.7 帧 / 秒， VGG -16 的 DRAM 访 问 /MAC 为 0.0035mW（*N* =3）。**

***索引词* —卷积神经网络（CNN）、数据流处理、深度学习、节能加速器、空间架构。**

I. 我 的介绍

由于卷积神经网络（CNN）的运算能耗高于计算[10][11]，其处理过程不仅需要通过高并行性实现高吞吐量，还需优化整个系统的数据传输路径以提升能效。此外，这种优化方案还需适应CNN中高维卷积运算的动态形态变化。

为应对这些挑战，关键在于设计一种名为 *数据流* 的计算方案。该方案需支持高度并行计算范式，同时优化芯片内部与外部数据传输的能耗。通过在多级存储器层级中利用数据复用技术，可有效降低数据传输成本，且硬件架构需具备可重构性以适配不同形态需求。为进一步提升能效，还可利用数据统计特性——具体而言，卷积神经网络数据中存在大量零值。通过应用压缩技术与数据自适应处理等方法，既能节省内存带宽，又能降低运算功耗。

先前的研究提出了多种CNN加速硬件设计方案[12]-[22]。然而，这些方案大多仅提供未经实际芯片验证的仿真结果；采用 FPGA 技术的实现方案也未能展现架构的实际吞吐量和能效表现。少数研究展示了芯片实测数据[23]-[25]，但这些成果均未使用业界主流的公开现成CNN模型进行基准测试，这对硬件评估至关重要。具体而言，Park *等人* [23]提出了一款基于 MIMD 架构的深度学习处理器，该架构通过定制四层网络（5×5滤波器）完成训练与推理。Cavigelli *等人* [24]开发的CNN加速器则在四层CNN（7×7滤波器）上进行测试。Sim *等人* [25]展示的CNN处理器仅报告了理论峰值吞吐量，并在 MNIST 数据集[26]上测得功耗——该数据集的存储和计算需求比当前最先进的CNN模型低几个数量级。除[24]外，这些研究均未报告所提计算方案所需的DRAM带宽。仅关注处理器性能是不够的，因为DRAM访问是决定系统能效的关键因素之一。

在本文中，我们已经实现并制造了一个CNN加速器，称为Eyeriss，它可以支持高吞吐量的CNN推理，并优化 *整个系统* 的能效，包括加速器芯片和

论文于2016年5月5日收到，7月31日修订，9月28日接受。2016年11月8日发表，2017年1月4日更新。本文由客座编辑德扬·马尔科维奇审阅批准。

Y.-H.陈和V.施就职于美国麻省理工学院电气工程与计算机科学系，地址为：美国马萨诸塞州剑桥市02139。

T. Krishna曾任职于美国麻省理工学院（MIT）电气工程与计算机科学系，现就职于美国佐治亚理工学院（Georgia Institute of Technology）电气与计算机工程学院。

J. S. Emer现任美国麻省理工学院（MIT）电气工程与计算机科学系（地址：美国马萨诸塞州剑桥市02139）及英伟达公司（地址：美国马萨诸塞州韦斯特福德01886）职务。

本文中一个或多个图的彩色版本可在线获取，网址为 http://ieeexplore.ieee.org 。

数字对象标识符 10.1109/JSSC.2016.2616357

TABLE I
SHAPE PARAMETERS OF A CNN LAYER

Shape Parameter	Description
N	batch size of 3D fmaps
M	# of 3D filters / # of ofmap channels
C	# of ifmap/filter channels
H/W	ifmap plane height/width
R/S	filter plane height/width
E/F	ofmap plane height/width

off-chip DRAM. It is also reconfigurable to handle different CNN shapes, including square and nonsquare filters. The main features of Eyeriss are as follows.

1) A spatial architecture using an array of 168 processing elements (PEs) that creates a four-level memory hierarchy. Data movement can exploit the low-cost levels, such as the PE scratch pads (spads) and the inter-PE communication, to minimize data accesses to the high-cost levels, including the large on-chip global buffer (GLB) and the off-chip DRAM.
2) A CNN dataflow, called *Row Stationary* (RS), that reconfigures the spatial architecture to map the computation of a given CNN shape and optimize for the best energy efficiency.
3) A network-on-chip (NoC) architecture that uses both multicast and point-to-point single-cycle data delivery to support the RS dataflow.
4) Run-length compression (RLC) and PE data gating that exploit the statistics of zero data in CNNs to further improve energy efficiency.

The performance of Eyeriss, including both the chip energy efficiency and required DRAM accesses, is benchmarked with two publicly available and widely used state-of-the-art CNNs: AlexNet [2] and VGG-16 [3]. These CNNs are designed for the most challenging computer vision task to date: 1000-class image classification on the ImageNet data set [27]. Eyeriss has also been integrated with Caffe [28] to demonstrate such application running in real-time [29].

II. CNN BASICS

The CNN algorithm is constructed by stacking multiple computation layers for feature extraction and classification [30]. Modern CNNs achieve their superior accuracy by building a very deep hierarchy of layers [2]–[5], [7], which transform the input image data into highly abstract representations called feature maps (fmaps).

The primary computation in the CNN layers is performing the high-dimensional convolutions. A layer applies **filters** on the input fmaps (**ifmaps**) to extract embedded characteristics and generate the output fmaps (**ofmaps**) by accumulating the partial sums (**psums**). The dimensions of both filters and fmaps are 4-D: each filter or fmap is a 3-D structure consisting of multiple 2-D planes, i.e., channels,[1] and a batch of

[1]To improve readability, except for explicit references to the *ofmap channel*, the word *channel* is used to only refer to the *ifmap/filter channel* for the rest of this paper.

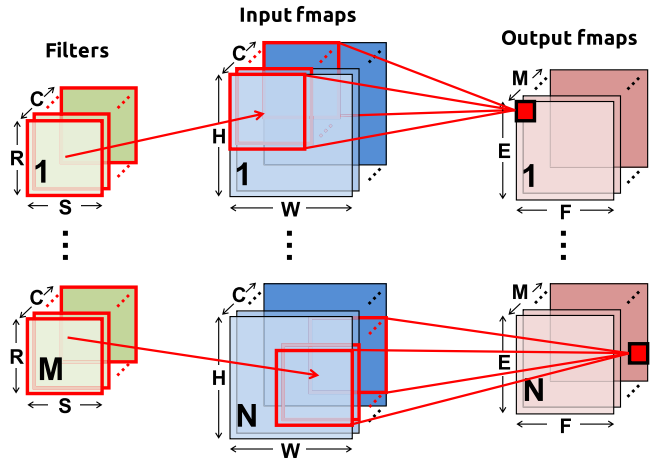


Fig. 1. Computation of a CNN layer.

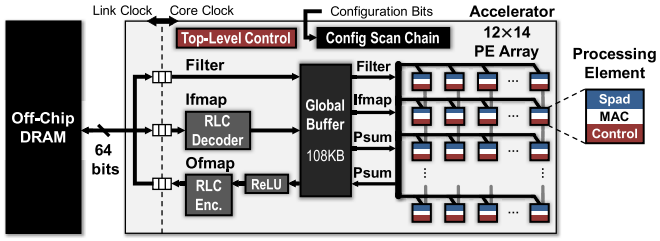


Fig. 2. Eyeriss system architecture.

3-D ifmaps is processed by a group of 3-D filters in a layer. In addition, there is a 1-D bias that is added to the filtering results. Given the shape parameters in Table I, the computation of a layer is defined as

$$\begin{aligned}\mathbf{O}[z][u][x][y] &= \mathrm{ReLU}\left(\mathbf{B}[u]+\sum_{k=0}^{C-1}\sum_{i=0}^{R-1}\sum_{j=0}^{S-1}\mathbf{I}[z][k][Ux+i][Uy+j]\right.\\ &\qquad \left.\times\mathbf{W}[u][k][i][j]\right),\\ &\quad 0\le z<N,\quad 0\le u<M,\ 0\le y<E,\ 0\le x<F\\ E &= (H-R+U)/U,\quad F=(W-S+U)/U \end{aligned} \tag{1}$$

where **O**, **I**, **W**, and **B** are the matrices of the ofmaps, ifmaps, filters, and biases, respectively. U is a given stride size. Fig. 1 shows a visualization of this computation (ignoring biases). After the convolutions, activation functions, such as the rectified linear unit (ReLU) [31], are applied to introduce nonlinearity.

III. SYSTEM ARCHITECTURE

A. Overview

Fig. 2 shows the top-level architecture and memory hierarchy of the Eyeriss system. It has two clock domains: the core clock domain for processing, and the link clock domain for communication with the off-chip DRAM through

表I

CNNL P 的 S HAPE 参数

形状参量谱轮廓参量	描述
N	3D f图的批量大小
M	# of 3D filters / # of ofmap channels
C	并of 如果有地图/过滤通道
H/W	如果地图平面高度/宽度
R/S	过滤器平面高度宽度
E/F	地图平面高度/宽度

片外DRAM。该芯片还可重新配置以支持不同CNN结构，包括方形和非方形滤波器。Eyeriss的主要功能如下。

1) 该架构采用168个处理单元（PE）组成的阵列，构建出四级存储器层级结构。数据传输可利用成本较低的层级（如处理单元的临时存储区和单元间通信），从而减少对高成本层级（包括片上全局缓冲器GLB和片外DRAM）的访问。
2) 一种名为 *Row Stationary*（RS）的CNN数据流，它重新配置空间架构，以映射给定CNN形状的计算，并优化最佳的能源效率。
3) 一种网络芯片（NoC）架构，通过采用多播与点对点单周期数据传输技术，为RS数据流提供支持。
4) 通过运用卷积神经网络中零数据的统计特性，采用运行长度编码（RLC）和峰值能量数据门控技术，从而进一步提升能效。

Eyeriss的性能表现（包括芯片能效和所需DRAM访问量）通过两个公开且广泛使用的前沿CNN模型——AlexNet[2]和 VGG -16[3]进行基准测试。这些CNN专为当前最具挑战性的计算机视觉任务设计：在ImageNet数据集[27]上实现1000类图像分类。Eyeriss还与Caffe框架[28]集成，成功展示了实时运行此类应用的能力[29]。

二、CNN B ASICS

CNN算法通过堆叠多个计算层来实现特征提取和分类[30]。现代CNN通过构建非常深层的层级结构[2]-[5]、[7]，将输入图像数据转化为高度抽象的特征图（fmaps），从而达到更高的准确率。

卷积神经网络（CNN）层的核心运算是对高维数据进行卷积操作。具体来说，每个卷积层会将 **滤波器** 作用于输入的特征图（**ifmap**），通过累加部分和（**psum**）提取特征并生成输出特征图（**ofmap**）。滤波器和特征图的维度均为四维：每个滤波器或特征图本质上是由多个二维平面（即通道）构成的三维结构，[1] 以及一组

1为了提高可读性，本文除明确提及 *ofmap信道* 外，*channel* 一词仅指代 *ifmap/筛选信道* 。

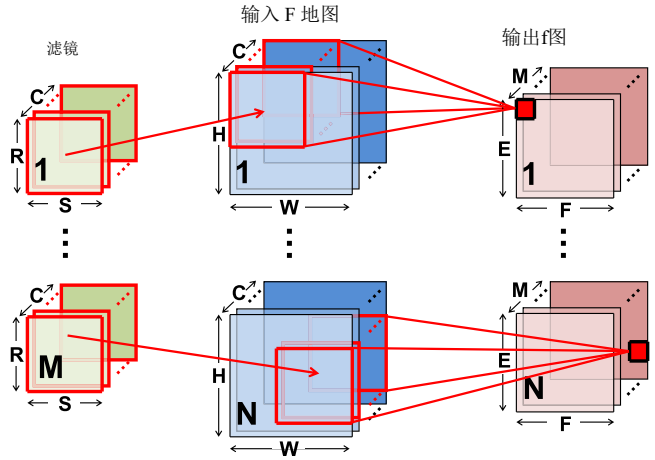


图1. CNN层的计算。

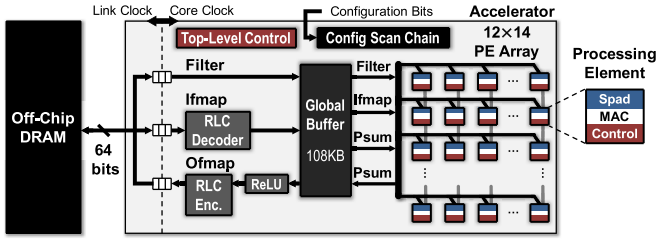


图2. Eyeriss系统架构。

三维ifmap通过层中的三维滤波器组进行处理，同时还会向滤波结果添加一维偏置。根据表I中的形状参数，该层的计算过程定义为

$$\mathbf{O}[z][u][x][y] = \mathrm{ReLU}\left(\mathbf{B}[u] + \sum_{k=0}^{C-1}\sum_{i=0}^{R-1}\sum_{j=0}^{S-1}\mathbf{I}[z][k][Ux+i][Uy+j]\right)$$

×W[u][k][i][j] ,

$$0 \le z < N,\ 0 \le u < M,\ 0 \le y < E,\ 0 \le x < F\ E = (H - R + U)/U,\ F = (W - S + U)/U \qquad (1)$$

其中 **O** 、 **I** 、 **W** 和 **B** 分别表示卷积映射、归一化映射、滤波器和偏置的矩阵。 U 为给定步长。图1展示了该计算过程的可视化（忽略偏置）。卷积操作后，通过激活函数（如修正线性单元ReLU[31]）引入非线性。

III . S YSTEM A 架构

A. 概述

图2展示了Eyeriss系统的顶层架构与存储器层级结构。该系统包含两个时钟域：用于处理的核心时钟域，以及用于与片外DRAM通信的链路时钟域。

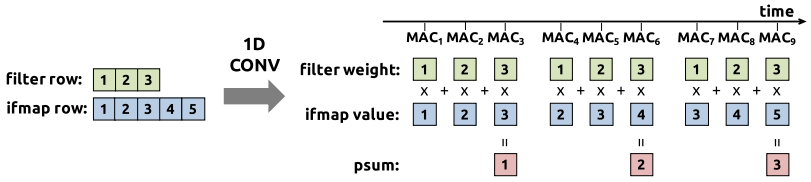


Fig. 3. Processing sequence of a 1-D convolution primitive in a PE. In this example, the filter row size (S) and the ifmap row size (W) are 3 and 5, respectively.

a 64-b bidirectional data bus. The two domains run independently and communicate through an asynchronous FIFO interface. The core clock domain consists of a spatial array of 168 PEs organized as a 12×14 rectangle, a 108-kB GLB, an RLC CODEC, and an ReLU module. To transfer data for computation, each PE can either communicate with its neighbor PEs or the GLB through an NoC, or access a memory space that is local to the PE called spads (Section V-C). Overall, there are four levels of memory hierarchy in the system (in decreasing energy per access): DRAM, GLB, inter-PE communication, and spads.

B. System Control and Configuration

The accelerator has two levels of control hierarchy. The top-level control coordinates: 1) traffic between the off-chip DRAM and the GLB through the asynchronous interface; 2) traffic between the GLB and the PE array through the NoC; and 3) operation of the RLC CODEC and ReLU module. The lower-level control consists of control logic in each PE, which runs independently of each other. Therefore, even though the 168 PEs are identical and run under the same core clock, their processing states do not need to proceed in lock steps, i.e., not as a systolic array. Each PE can start its own processing as soon as any fmaps or psums arrives (fmaps or psums).

The accelerator runs the processing of a CNN layer-by-layer. For each layer, it first loads the configuration bits into a 1794 b scan chain serially to reconfigure the entire accelerator, which takes less than 100 μs. These bits configure the accelerator for the processing of filters and fmaps in a certain shape, which includes setting up the PE array computation mappings (Section IV-A) and NoC data delivery patterns (Section V-B). They are generated offline and are statically accessed at runtime. Then, the accelerator loads tiles of the ifmaps and filters from DRAM for processing, and the computed ofmaps are written back to DRAM. Batches of ifmaps for the same layer can be processed sequentially without further reconfigurations of the chip.

IV. Energy-Efficient Features

The Eyeriss chip focuses on two main approaches to improve the energy efficiency: 1) reducing data movement and 2) exploiting data statistics.

A. Energy-Efficient Dataflow: Row Stationary

In Eyeriss, we propose the RS dataflow that maps the computation of any given CNN shape onto the PE array. It is reconfigurable for different shapes and optimizes for the best energy efficiency [32]. The RS dataflow minimizes data movement for all data types (ifmap, filter, and psums/ofmap) simultaneously and takes the energy costs at different levels of the memory hierarchy into account. Data accesses to the high-cost DRAM and GLB are minimized through maximally reusing data from the low-cost spads and inter-PE communication. Compared with existing dataflows from previous works, the RS dataflow is 1.4–2.5 times more energy efficient in AlexNet, a widely used CNN [2].

To minimize the movement of ifmaps and filters, the goal is to maximize three forms of data reuse.

1) *Convolutional Reuse:* Each filter weight is reused $E \times F$ times in the same ifmap plane, and each ifmap pixel is usually reused $R \times S$ times in the same filter plane.
2) *Filter Reuse:* Each filter weight is reused across the batch of N ifmaps.
3) *Ifmap Reuse:* Each ifmap pixel is reused across M filters (to generate M ofmap channels).

To minimize the movement of psums, it is desirable that the **psum accumulation** across $C \times R \times S$ values into one ofmap value can be done as soon as possible to save both the storage space and memory R/W energy. However, maximum input data reuse cannot be achieved simultaneously with immediate psum reduction, since the psums generated by multiply and accumulations (MACs) using the same filter or ifmap value are not reducible. Thus, the RS dataflow uses a systematic approach to optimize for all data types simultaneously as follows.

1-D Convolution Primitive in a PE: The RS dataflow first divides the computation of (1) into 1-D convolution primitives that can all run in parallel. Each primitive operates on one row of filter weights and one row of ifmap values, and generates one row of psums. The psums from different primitives are further accumulated together to generate the ofmap values. By mapping each primitive to one PE for processing, the computation of each *row pair stays stationary* in the PE. Due to the sliding window processing of the primitive as shown in Fig. 3, each PE can use the local spads for both convolutional data reuse and psum accumulation. Since only a sliding window of data has to be retained at a time, the required spad capacity depends only on the filter row size (S) but not the ifmap row size (W), and is equal to: 1) S for a row of filter weights; 2) S for a sliding window of ifmap values; and 3) 1 for the psum accumulation. In AlexNet, for example, possible values for S are 11 (layer CONV1), 5 (layer CONV2), and 3 (layers CONV3–CONV5). Therefore, the minimum spad capacity required for filter, ifmap, and psum is 11, 11, and 1, respectively, to support all layers.

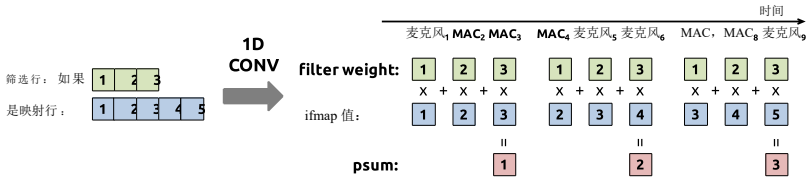


图3. PE中1-D卷积原语的处理序列。在本示例中，滤波器行尺寸（S）和ifmap行尺寸（W）分别为3和5。

该系统采用64位双向数据总线架构。两个域独立运行，通过异步FIFO接口实现通信。核心时钟域包含168个处理单元（PE）组成的矩形阵列（12×14布局）、108KB通用逻辑块（GLB）、 RLC 编解码器及ReLU模块。在数据传输过程中，每个处理单元可通过网络（NoC）与相邻单元或GLB通信，也可访问其本地内存空间——片上存储器（spads，详见第五章C节）。整个系统采用四级存储层级结构（按每次访问能耗递减排序）：动态随机存取存储器（DRAM）、通用逻辑块（GLB）、处理单元间通信通道以及片上存储器。

B. 系统控制和配置

该加速器采用两级控制层级架构。顶层控制模块主要负责：1）通过异步接口协调片外DRAM与通用逻辑块（GLB）之间的数据传输；
2) 3）通过网络芯片（NoC）实现通用逻辑块（GLB）与处理单元（PE）阵列之间的数据传输；4）RLC 编解码器与ReLU模块的协同运作。底层控制架构由各处理单元独立运行的控制逻辑构成。因此，尽管168个处理单元完全相同且运行在相同核心时钟下，其处理状态无需按锁步顺序推进，即不采用脉动阵列模式。当任意浮点映射（fmaps）或浮点求和（psums）数据到达时，每个处理单元即可立即启动独立处理流程。

该加速器采用逐层处理卷积神经网络（CNN）的方式运行。在处理每一层时，首先将配置位以串行方式加载到1794b扫描链中，完成整个加速器的重新配置，整个过程耗时不到100 μ秒。这些配置位用于设定加速器处理特定形状滤波器和滤波器映射（fmaps）的参数，包括配置处理器单元（PE）阵列的计算映射（第四章A节）和网络通信（NoC）的数据传输模式（第五章B节）。这些配置数据采用离线生成方式，运行时通过静态访问获取。随后，加速器从动态随机存取存储器（DRAM）中加载滤波器映射（ifmaps）和滤波器的处理单元进行运算，计算完成的滤波器映射结果则回写至DRAM。对于同一层的多个ifmaps批次，无需对芯片进行额外配置即可实现顺序处理。

IV. Energy-Efficient 特性

Eyeriss芯片主要通过两种方式提升能效：1）减少数据传输；2）利用数据统计特性。

A. 节能数据流：行稳态

在Eyeriss中，我们提出了一种RS数据流，可将任意给定CNN结构的计算映射到PE阵列上。该架构支持不同结构的重新配置，并针对特定需求进行优化。

RS数据流[32]实现了最佳能效。该方案通过同步优化所有数据类型（ifmap、filter及psums/ofmap）的数据传输路径，同时综合考量内存层级不同层级的能耗成本。通过最大限度复用低成本存储单元（spads）和处理器间通信产生的数据，有效减少了对高能耗动态随机存取存储器（DRAM）和全局逻辑块（GLB）的访问。与现有研究中的数据流方案相比，在广泛应用的AlexNet卷积神经网络[2]中，RS数据流的能耗效率提升了1.4至2.5倍。

为减少ifmap和过滤器的移动，目标是实现三种数据重用形式的最大化。

1) *卷积重用：* 同一ifmap平面内，每个滤波器权重被重用 $E \times F$ 次；同一滤波器平面内，每个ifmap像素通常被重用 $R \times S$ 次。
2) *过滤器重用：* 如果 N 个ifmap的批次中重复使用每个滤波器权重。
3) *如果map重用：* 每个ifmap像素在 M 个滤波器中被重用（生成 M 个ofmap通道）。

为最大限度减少累加器（psum）的移动，建议尽早将 $C \times R \times S$ 值对应的 **psum累加** 量整合到单个映射值中，这样既能节省存储空间，又能降低内存读写能耗。但需要特别注意的是，由于使用相同滤波器或映射值的乘积累加单元（MAC）生成的psum值无法直接合并，因此无法同时实现最大输入数据复用和即时psum缩减。为此，RS数据流采用系统化优化策略，针对所有数据类型同步实施以下改进措施。

单元内一维卷积原语： RS数据流首先将计算过程(1)拆分为多个可并行执行的一维卷积原语。每个原语处理一行滤波器权重和一行ifmap值，并生成一行psum值。不同原语产生的psum值会进一步累加生成ofmap值。通过将每个原语分配给一个处理单元进行运算， *每行数据对的计算过程在单元内保持稳定* 。由于原语采用图3所示的滑动窗口处理方式，每个处理单元可同时利用本地spad进行卷积数据复用和psum累加。由于每次仅需保留滑动窗口的数据，所需spad容量仅取决于滤波器行尺寸（S）而非ifmap行尺寸（W），具体计算公式为：1）滤波器权重行对应 S；2）ifmap值滑动窗口对应 S；3）psum累加对应1。以AlexNet为例， S 的可能取值为11（CONV1层）、5（CONV2层）和3（CONV3至CONV5层）。因此，为支持所有层，filter、ifmap和psum的最小spad容量分别为11、11和1。

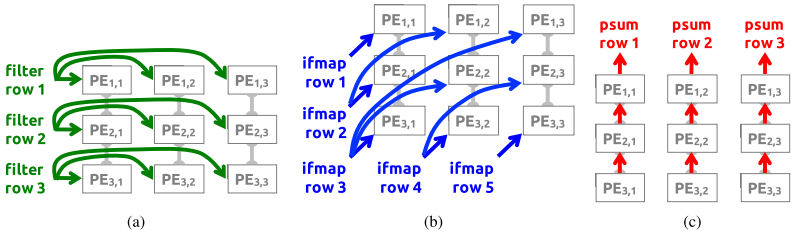


Fig. 4. Dataflow in a PE set for processing a 2-D convolution. (a) Rows of filter weights are reused across PEs horizontally. (b) Rows of ifmap values are reused across PEs diagonally. (c) Rows of psums are accumulated across PEs vertically. Reuse and accumulation of data within a PE set reduce accesses to the GLB and DRAM, saving data movement energy cost. In this example, the number of filter rows (R), ifmap rows (H), and ofmap rows (E) are 3, 5, and 3, respectively. Therefore, the PE set size is 3×3. Filter and ifmap values from different rows are sent to the PE set in a time-interleaved fashion; all the PEs that reuse the same value receive it at the same cycle. The psums generated from one PE are sent to its neighbor PE immediately.

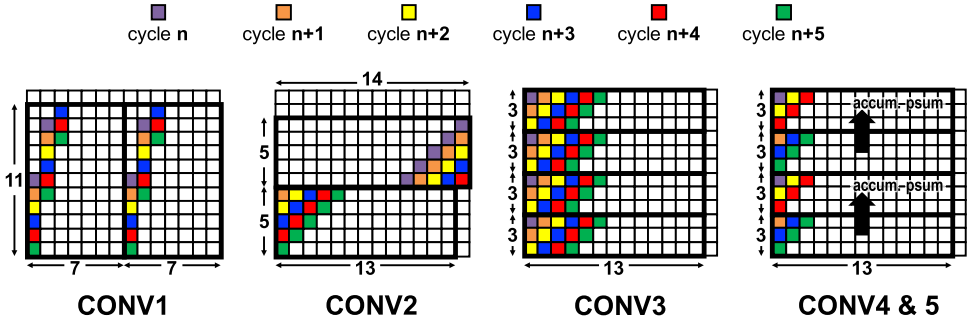


Fig. 5. Mapping of the PE sets on the spatial array of 168 PEs for the CONV layers in AlexNet. For the colored PEs, the PEs with the same color receive the same ifmap value in the same cycle. The arrow between two PE sets indicates that their psums can be accumulated together.

2-D Convolution PE Set: A 2-D convolution is composed of many 1-D convolution primitives, and its computation: 1) shares the same row of filter or ifmap across primitives and 2) accumulates the psums from multiple primitives together. Therefore, a *PE Set*, as shown in Fig. 4, is grouped to run a 2-D convolution and exploit the interprimitive convolutional reuse and psum accumulation, which avoids data accesses from GLB and DRAM. In a set, each row of filter is reused horizontally, each row of ifmap is reused diagonally, and rows of psum are accumulated vertically. The dimensions of a PE set are determined by the filter and ofmap size of a given layer. Specifically, the height and the width of the PE set are equal to the number of filter rows (R) and ofmap rows (E), respectively. In AlexNet, the PE sets are of size 11×55 (CONV1), 45×27 (CONV2), and 3×13 (CONV3–CONV5).

PE Set Mapping: The dimensions of a PE set are a function of the shape of a layer and are independent of the physical dimensions of the PE array. Therefore, a strategy is required to map these PE sets onto the PE array. This mapping strategy should map a set using nearby PEs in the array to take the advantage of local data sharing and psum accumulation. In Eyeriss, a PE set can be mapped to any group of PEs in the array that has the same dimensions. However, there are two exceptions.

1) *PE Set Has More Than 168 PEs:* This can be solved by *strip mining* the 2-D convolution, i.e., the PE set only processes e rows of ofmap at a time, where $e \leq E$. The dimensions of the strip-mined PE set then becomes $R \times e$ and can fit into the PE array.
2) *PE Set Has Less Than 168 PEs, But Has Width Larger Than 14 or Height Larger Than 12:* A PE set that is too wide is divided into separated segments that are mapped independently to the array. Eyeriss currently does not support the mapping of a PE set that is taller than the height of the PE array. Therefore, the maximum natively supported filter height is 12.

An example of these two exceptions can be seen from the PE set mapping of layers CONV1–CONV5 in AlexNet onto the 12×14 PE array of Eyeriss as shown in Fig. 5. The 11×55 PE set of CONV1 is strip-mined to 11×7. The strip-mined PE set width is determined by a process that optimizes for overall energy efficiency as introduced in [32]. The 5×27 PE set of CONV2 is divided into two segments with dimensions 5×14 and 5×13, respectively, and each segment is independently mapped onto the PE array. Finally, the 3×13 PE set of CONV3–CONV5 can easily fit into the PE array. Except for CONV2, the PE array can fit multiple PE sets in parallel as shown in Fig. 5, and the RS dataflow further defines how to fully utilize hardware resources to minimize data movement in the dimensions beyond 2-D. This mapping strategy is realized by a custom designed NoC that is also optimized for energy efficiency (Section V-B).

Dimensions Beyond 2-D in PE Array: Processing of many 2-D convolutions is required to complete the computation of (1) due to the three additional dimensions: batch size (N), number of channels (C), and number of filters (M). Assuming varying only 1-D at a time and fixing the rest of the two the same, two 2-D convolutions that use: 1) different ifmaps reuse the same filter (i.e., filter reuse); 2) different filters reuse the same ifmap (i.e., ifmap reuse); and 3) filters and ifmaps from different channels can accumulate their psums together (i.e., psum accumulation). The filter reuse can be exploited

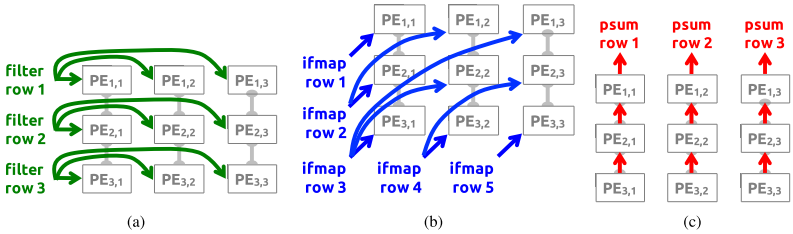


图4.并行执行单元组处理二维卷积的数据流。(a)滤波器权重行在各PE之间水平共享。(b)ifmap值行在各PE之间对角共享。(c)psum值行在各PE之间垂直累加。PE组内部的数据共享与累加机制可减少对GLB和DRAM的访问次数，从而节省数据传输能耗。本例中滤波器行数（R）、ifmap行数（H）和ofmap行数（E）分别为3、5和3，因此PE组规模为3×3。不同行的滤波器和ifmap值以时分复用方式传输至PE组，共享同一数值的所有PE会在同一时钟周期接收。由单个PE生成的psum值会立即传输至其相邻PE。

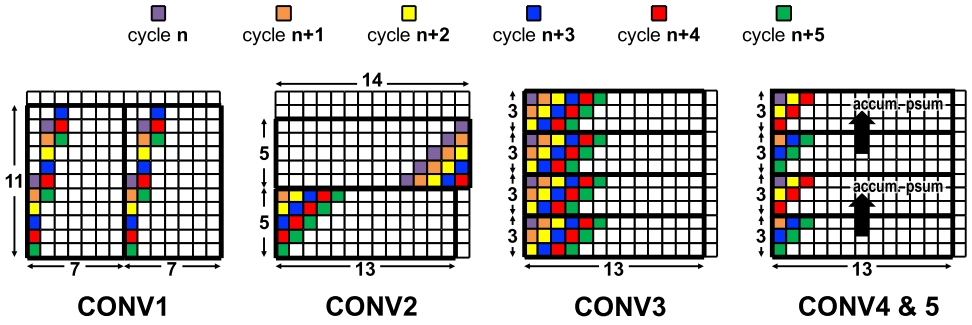


图5展示了AlexNet网络CONV层中168个处理单元（PE）的空间阵列映射关系。对于彩色标记的处理单元，同色单元在相同计算周期内会共享相同的ifmap值。两个处理单元集之间的箭头表示它们的psum值可以合并计算。

2-D卷积PE集： 2-D卷积由许多1-D卷积原语组成，其计算：
1) 若原始组件间共享同一行过滤器或ifmap
2) 将多个基础运算单元的累加和（psum）结果进行整合。因此，如图4所示，*每个处理单元组*（PE Set）被组合运行二维卷积运算，通过跨运算单元的卷积复用和psum累加机制，有效避免了对通用逻辑块（GLB）和动态随机存取存储器（DRAM）的数据访问。在每个处理单元组中，滤波器的行在水平方向上复用，ifmap的行在对角方向上复用，而psum的行则在垂直方向上累加。处理单元组的维度由给定网络层的滤波器和输出映射（ifmap）尺寸决定，具体而言，其高度和宽度分别等于滤波器行数（R）和ifmap行数（E）。以AlexNet网络为例，其处理单元组的尺寸分别为11×55（CONV1）、45×27（CONV2）以及3×13（CONV3-CONV5）。

任务执行单元（PE）集合映射： 任务执行单元集合的维度由层级结构决定，与PE阵列的实际物理尺寸无关。因此需要制定映射策略，将这些PE集合合理分配到阵列中。该映射策略应优先选择阵列中相邻的PE进行组合，以充分利用本地数据共享和累加求和（psum）功能。在Eyeriss系统中，任何具有相同维度的PE组均可作为映射对象。但存在两个例外情况：

1) *PE集包含超过168个处理单元：* 通过*分段式提取*二维卷积运算即可解决，即每次仅处理 e 行输出图，其中 $e \le E$。这样分段提取后的处理单元集合维度变为 $R \times e$，可适配处理单元阵列。
2) *PE集包含少于168个PE，但宽度大于14或高度大于12：* PE集太大宽幅数据被分割成多个独立映射到阵列的片段。Eyeriss目前不支持映射超过PE阵列高度的PE集。因此，原生支持的最大滤波器高度为12。

这两个例外情况的典型示例，可通过图5所示的AlexNet网络CONV1至CONV5层在Eyeriss平台12×14 PE阵列上的映射方案观察。CONV1的11×55 PE阵列采用条带式切片技术缩减为11×7，其切片宽度的确定遵循文献[32]提出的整体能效优化方案。CONV2的5×27 PE阵列被划分为两个子阵列（尺寸分别为5×14和5×13），每个子阵列独立映射至PE阵列。而CONV3至CONV5的3×13 PE阵列则能完美适配该阵列。除CONV2外，如图5所示，PE阵列可并行容纳多个PE阵列，RS数据流架构进一步明确了如何充分利用硬件资源，最大限度减少二维维度之外的数据传输。这种映射策略通过专门设计的网络芯片（NoC）实现，该芯片同样经过能效优化（详见第五章B节）。

PE阵列突破二维维度限制： 由于新增了批量大小（N）、通道数（C）和滤波器数（M）这三个维度，完成(1)的计算需要处理大量二维卷积操作。假设每次仅调整其中一个维度，同时固定另外两个维度，那么在二维卷积中可以采用以下三种策略：1）不同ifmap共享同一滤波器（即滤波器复用）；2）不同滤波器共享同一ifmap（即ifmap复用）；3）来自不同通道的滤波器和ifmap可以累加其psum值（即psum累加）。其中滤波器复用可被有效利用

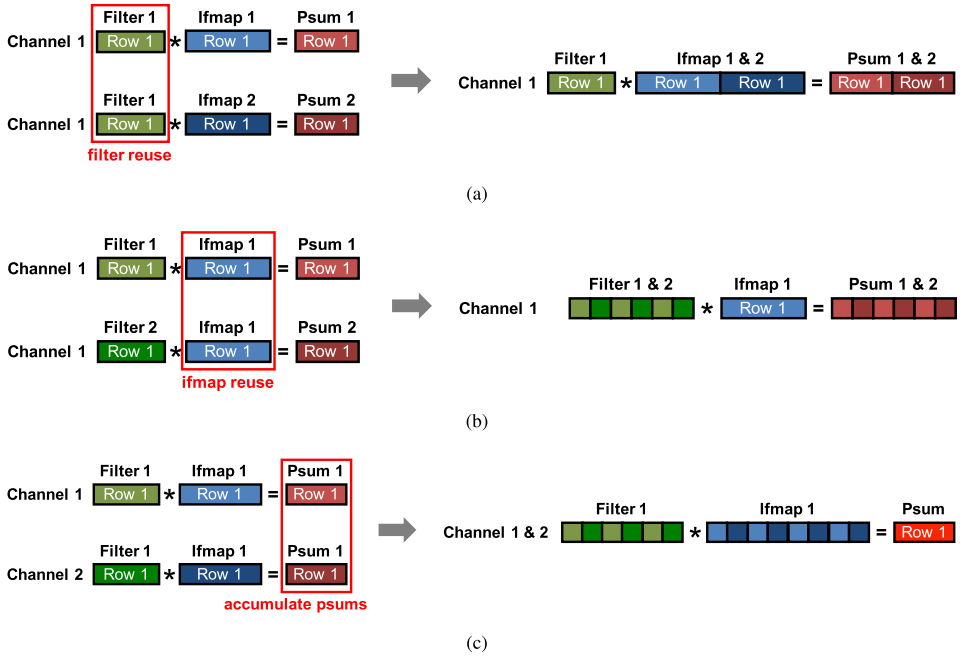


Fig. 6. Handling the dimensions beyond 2-D in each PE by (a) concatenating the ifmap rows, each PE can process multiple 1-D primitives with different ifmaps and reuse the same filter row and (b) time interleaving the filter rows, each PE can process multiple 1-D primitives with different filters and reuse the same ifmap row. (c) By time interleaving the filter and ifmap rows, each PE can process multiple 1-D primitives from different channels and accumulate the psums together.

simply by streaming different ifmaps through the same PE set sequentially [Fig. 6(a)], since the filter stays constant in the set. The ifmap reuse and psum accumulation opportunities can also be exploited by using either the spads or the spatial parallelism of the PE array, so DRAM and GLB accesses are further reduced.

1) Multiple 2-D Convolutions in a PE Set: If the spad size is large enough, each PE can run multiple 1-D convolution primitives simultaneously by interleaving their computation. Equivalently, this means *each PE set is running multiple 2-D convolutions* on different filters and channels. There are two scenarios.

1) By interleaving the computation of primitives that run on the same ifmap with different filters, the spads can buffer the same ifmap value and reuse it to compute with a weight from each filter sequentially [Fig. 6(b)]. It requires increasing the filter and psum spad size.
2) By interleaving the computation of primitives that run on different channels, the PE can accumulate through all channels sequentially on the same psum [Fig. 6(c)]. This requires increasing the ifmap and filter spad size.

The mapping of multiple primitives in the same PE can be described by parameters p and q. Each PE runs $p \times q$ primitives simultaneously from q different channels of p different filters. The required spad capacity for each data type is: 1) $p \times q \times S$ for the rows of filter weights from q channels of p filters; 2) $q \times S$ for q sliding windows of ifmap values from q different channels; and 3) p for the accumulation of psums in p ofmap channels. In Eyeriss, where ifmap spad is 12 b $\times$16 b, filter spad is 224 b $\times$16 b, and psum spad is 24 b $\times$16 b, p can be up to 24, and q can be up to 4 in AlexNet since the minimum S is 3.

2) Multiple PE Sets in the PE Array: As shown in Fig. 5, the PE array can fit more than one PE set if the set is small enough. Mapping multiple sets not only increases the utilization of PEs, which increases processing throughput, but also brings two extra advantages: 1) the same ifmap is read once from the GLB and reused in multiple sets simultaneously and 2) the psums from different sets are further accumulated within the PE array directly.

The mapping of multiple sets is described by parameters r and t. The PE array fits $r \times t$ PE sets in parallel that run r different channels of t different filters simultaneously. Every t sets share the same ifmap with t filters, and every r sets that run on r channels accumulate their psums within the PE array. Fig. 5 shows the mapping of multiple sets and the reuse of ifmaps in Eyeriss. Specifically, CONV1 and CONV3 have $t = 2$ and 4, respectively, and the same ifmap value is sent to all sets. CONV4 and CONV5 have $r = t = 2$. The same ifmap value is sent to every other set, and the psums from the top and bottom two sets are accumulated together. In each layer, the PEs that are not covered by any sets are clock gated to save energy consumption.

a) PE array processing passes: So far we have described a way to exploit data reuse by maximally utilizing the storage of spads and the spatial parallelism of the PE array. The PE array can run multiple 2-D convolutions from up to $q \times r$ channels of $p \times t$ filters simultaneously. Multiple ifmaps can also be processed sequentially through the array. The amount of computation done in this fashion is called a *Processing Pass*. In a pass, each input data are read only once from the GLB, and the psums are stored back to the GLB only once when the processing is finished.

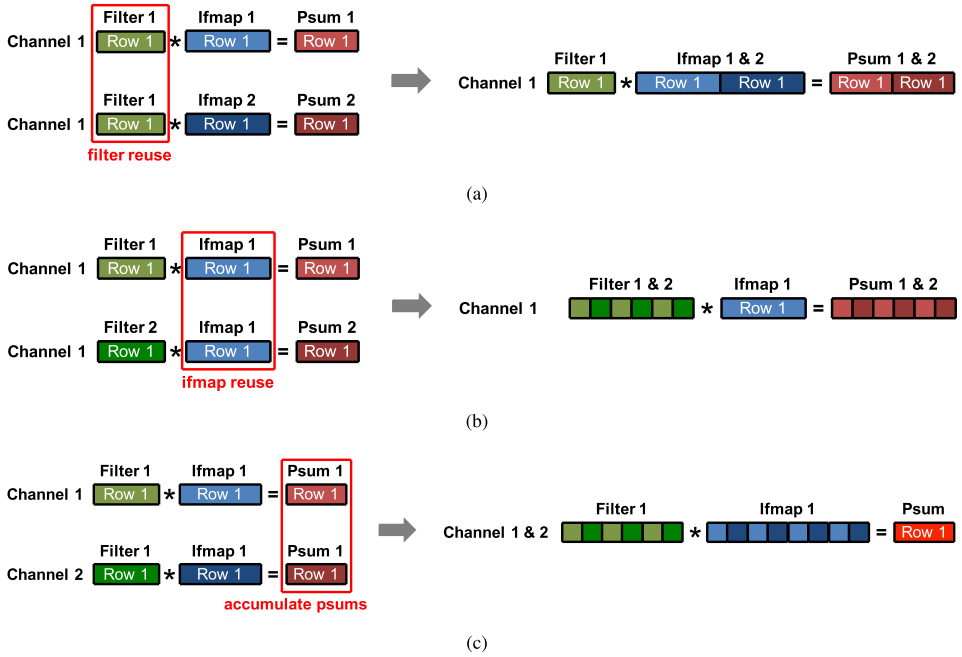


图6展示了各处理单元（PE）处理二维（2-D）以上维度的两种方案：(a)通过拼接ifmap行，每个PE可同时处理多个具有不同ifmap的一维运算单元，并复用同一滤波器行；(b)通过时间交织滤波器行，每个PE可处理不同滤波器的一维运算单元，并复用同一ifmap行；(c)通过时间交织滤波器行和ifmap行，每个PE可处理来自不同通道的多维运算单元，并将psum结果累加。

只需将不同ifmap按顺序流式传输至同一PE组[图6(a)]，由于该组内滤波器保持恒定。通过利用PE阵列的spad或空间并行性，可进一步提升ifmap复用和psum累加效率，从而显著降低DRAM和GLB的访问需求。

1) PE组中的多个2-D卷积： 如果spad尺寸足够大，每个PE可以通过交错计算同时运行多个1-D卷积原语。等效地，这意味着 *每个PE组在不同的滤波器和通道上运行多个2-D卷积* 。有两种情况。

1) 通过交替执行同一ifmap上运行的不同滤波器的原始运算，spad可以缓存相同的ifmap值，并依次使用每个滤波器的权重进行计算[图6(b)]。这需要增加滤波器和psum spad的尺寸。
2) 通过交错执行不同通道上的基本运算，处理器执行单元（PE）可将所有通道的运算结果依次累加至同一累加器[图6(c)]。这需要增大输入映射（ifmap）和滤波器的卷积核尺寸。

在单个处理单元（PE）中映射多个基础运算单元时，可通过参数 p 和 q 进行描述。每个处理单元会从 p 个不同滤波器的 q 个不同通道中并行运行 $p \times q$ 个基础运算单元。各数据类型所需的滑动窗口加速器（spad）容量分别为：1）来自 q 个通道的 p 个滤波器权重行的运算单元数为 $p \times q \times S$；2）来自 q 个不同通道的ifmap值 q 滑动窗口运算单元数为 $q \times S$；3）来自 p 个map通道的psum累加运算单元数为 p 。在Eyeriss架构中，ifmap的spad容量为12 b ×16 b，滤光片光斑尺寸为224 b ×16 b，而psum光斑尺寸为24b×16b，AlexNet中 p 的最大值为24， q 的最大值为4，因为最小 S 为3。

2) PE阵列中的多组处理单元： 如图5所示，当处理单元组足够小时，PE阵列可容纳多个处理单元组。这种多组映射不仅提升了处理单元的利用率，从而提高处理吞吐量，还带来两大额外优势：1）同一ifmap可从GLB中一次性读取并在多个组中同步复用；2）不同组的psum结果可直接在PE阵列内部进行累加。

多组映射关系由参数 r 和 t 描述。PE阵列并行处理 $r \times t$ 组PE数据集，这些数据集同时运行 r 个不同通道的 t 个不同滤波器。每个 t 组共享同一组包含 t 个滤波器的ifmap，而每个在 r 个通道上运行的 r 组则在PE阵列内累加各自的psum值。图5展示了Eyeriss架构中多组映射及ifmap复用的具体实现：CONV1和CONV3分别具有 t =2和4的滤波器数量，所有组共享同一组ifmap；CONV4和CONV5的 $r = t =$ 2，其他组共享同一组ifmap，且上下两组的psum值会合并累加。在每一层中，未被任何组覆盖的PE单元会通过时钟门控技术来降低能耗。

a) PE阵列处理流程详解： 我们已介绍过如何通过最大化利用存储单元（spads）的存储空间和PE阵列的空间并行特性来实现数据复用。该阵列可同时处理多达 $q \times r$ 通道的 $p \times t$ 滤波器进行多维卷积运算，同时还能按顺序处理多个输入流（ifmaps）。这种计算方式被称为 *处理通道* 。在每个处理通道中，输入数据仅从全局存储器（GLB）读取一次，而累加运算结果（psums）也仅在处理完成后回写一次至GLB。

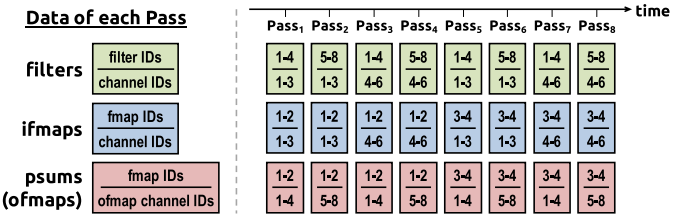


Fig. 7. Scheduling of processing passes. Each block of filters, ifmaps, or psums is a group of 2-D data from the specified dimensions used by a processing pass. The number of channels (C), filters (M), and ifmaps (N) used in this example layer created for demonstration purpose are 6, 8, and 4, respectively, and the RS dataflow uses eight passes to process the layer.

TABLE II
MAPPING PARAMETERS OF THE RS DATAFLOW

Parameter	Description
m	number of ofmap channels stored in the global buffer
n	number of ifmaps used in a processing pass
e	width of the PE set (strip-mined if necessary)
p	number of filters processed by a PE set
q	number of channels processed by a PE set
r	number of PE sets that process different channels in the PE array
t	number of PE sets that process different filters in the PE array

A CNN layer usually requires hundreds to thousands of processing passes to complete its processing, and ifmap reuse and psum accumulation also exist across these passes. The GLB is used to exploit these opportunities by buffering two types of data: ifmaps and psums. The ifmaps stored in the GLB can be reused across multiple processing passes; the psums that are accumulated across passes use the GLB as the intermediate storage, so they do not go off-chip until the final ofmap values are obtained.

Fig. 7 shows the scheduling of processing passes. This example layer, created only for illustrative purposes, has six channels (C), eight filters (M), and four ifmaps (N). A pass is assumed to process three channels ($q \times r$) and four filters ($p \times t$). Also, the number of ifmaps that a pass processes, denoted as n, is assumed to be 2. Overall, the computation of this layer uses eight processing passes. Each group of ifmaps is read from DRAM once, stored in the GLB, and reused in two consecutive passes with total eight filters to generate eight ofmap channels. However, this also requires the GLB to store psums from two consecutive passes so they do not go to DRAM. In this case, the GLB needs to store $m = 8$ ofmap channels. Each filter weight is read from DRAM into the PE array once for every four passes.

The scheduling of the processing passes determines the storage allocation required for ifmaps and psums in the GLB. Specifically, $n \times q \times r$ 2-D ifmaps and $n \times m$ 2-D psums have to be stored in the GLB for reuse. Since these parameters change based on the mapping of each layer, the GLB allocation for ifmaps and psums has to be reconfigurable to store them in different proportions.

b) Summary: Table II summarizes a list of dataflow mapping parameters that define the mapping of the RS dataflow. For a given CNN shape, these parameters are determined by an optimization process that takes: 1) the energy cost at each level of the memory hierarchy and 2) the hardware resources, including the GLB size, spad size, and number of PEs, into account [32]. Table III lists the RS dataflow mapping parameters used for AlexNet in Eyeriss. It also shows the storage required in the GLB for both ifmap and psum.

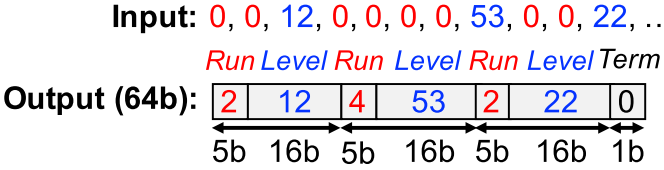


Fig. 8. Encoding of the RLC.

B. Exploit Data Statistics

Even though the RS dataflow optimizes data movement for all data types, the intrinsic amount of data and the corresponding computation are still high. To further improve energy efficiency, data statistics of CNN is explored to: 1) reduce DRAM accesses using compression, which is the most energy consuming data movement per access, on top of the optimized dataflow; and 2) skip the unnecessary computations to save processing power (Section V-C).

The ReLU function introduces many zeros in the fmaps by rectifying all negative filtering results to zero. While the number of zeros in the fmaps depends on the input data to the CNN, it tends to increase with deep layers. In AlexNet, almost 40% of ifmap values of CONV2 are zeros on average, and it goes up to around 75% at CONV5. In addition to the fmap, a recent study has also shown that 16%–78% filter weights in a CNN can be pruned to zero [33].

RLC is used in Eyeriss to exploit the zeros in fmaps and save DRAM bandwidth. Fig. 8 shows an example of RLC encoding. Consecutive zeros with a maximum run length of 31 are represented using a 5-b number as the *Run*. The next value is inserted directly as a 16-b *Level*, and the count for run starts again. Every three pairs of run and level are packed into a 64-b word, with the last bit indicating if the word is the last one in the code. Based on our experiments using AlexNet with the ImageNet data set, the compression rate of RLC only adds 5%–10% overhead to the theoretical entropy limit.

Except for the input data to the first layer of a CNN, all the fmaps are stored in RLC compressed form in the DRAM. The accelerator reads the encoded ifmaps from DRAM, decompresses it with the RLC decoder, and writes it into the GLB. The computed ofmaps are read from the GLB, processed by the ReLU module optionally, compressed by the RLC encoder, and transmitted to the DRAM. This saves both space and R/W bandwidth of the DRAM. From our experiments using AlexNet, the DRAM accesses for fmaps alone, including both ifmaps and ofmaps, can be saved by nearly 30% in CONV1, and nearly 75% in CONV5. Fig. 9 shows the overall DRAM accesses in AlexNet before and after RLC. The traffic includes filters, ifmaps, and ofmaps. The DRAM access could be further reduced if RLC was applied to filter weights.

V. SYSTEM MODULES

A. Global Buffer

The Eyeriss accelerator has a GLB of 108 kB that can communicate with DRAM through the asynchronous interface

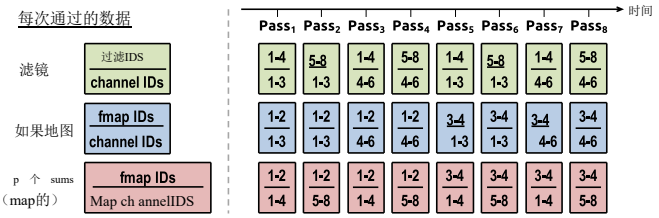


图7.处理通道的调度。每个filter块、ifmap或psum都是处理通道所使用的指定维度的二维数据组。此示例层创建的用于演示目的的层中使用的通道数（C）、filter数（M）和ifmap数（N）分别为6、8和4，RS数据流使用8个通道来处理该层。

表格 II

RS数据流的 M APPING P 参数

参数	描述
m	全局缓冲区中存储的地图通道数量
n	处理过程中使用的IF映射数量
e	pE Set的宽度（必要时采用条带式开采）
p	ApE set处理的过滤器数量
q	ApE集处理的通道数量
r	pE阵列中处理不同通道的pE集数量
t	pE阵列中处理不同过滤器的pE集合数量

一个CNN层通常需要经过数百到数千次处理才能完成运算，而ifmap复用和psum累加也会在这些处理过程中持续存在。GLB通过缓冲两种数据类型来利用这些机会：ifmap和psum。存储在GLB中的ifmap可以在多次处理中重复使用；而跨处理累加的psum则将GLB作为中间存储，因此直到最终获得ofmap值时才会离开芯片。

图7展示了处理通道的调度安排。这个示例层仅为说明目的而创建，包含六个通道（C）、八个滤波器（M）和四个ifmap（N）。每个处理通道假设需要处理三个通道（$q \times r$）和四个滤波器（$p \times t$）。同时，假设每个处理通道处理的ifmap数量 n 为2。总体而言，该层的计算过程共需进行八次处理通道操作。每个ifmap组会从DRAM读取一次数据，存储在GLB中，并在连续两次处理通道中重复使用，总共生成八个ifmap通道。不过这也意味着GLB需要存储两次连续处理通道的psum值，以避免这些数据再次进入DRAM。因此，GLB需要存储 m =8个ifmap通道。每个滤波器权重则需在每次处理通道时从DRAM读取一次并存入PE阵列。

处理通道的调度决定了GLB中ifmap和psum所需的存储分配。具体而言， $n \times q \times r$ 二维ifmap和 $n \times m$ 二维psum需要在GLB中存储以供复用。由于这些参数会根据各层的映射关系发生变化，因此GLB中ifmap和psum的存储分配必须具备可重构性，以便根据实际情况调整存储比例。

b) 摘要： 表II汇总了定义RS数据流映射的参数列表。对于给定的CNN架构，这些参数通过优化过程确定，该过程需考虑：1）存储器层级各层级的能量成本；2）硬件资源，包括全局逻辑块（GLB）尺寸、存储单元（spad）尺寸及数量。

输入：0,0,12， 0,0,0,0,53， 0,0,22， ...运行级别 运

行级别运行级别术语

输出（64b）：区工工团工更工回工至工回

5b 16b 5b 16b 5b 16b 1b

图8. RLC 的编码。

将PEs纳入考量[32]。表III列出了Eyeriss中AlexNet使用的RS数据流映射参数，同时展示了GLB对ifmap和psum所需的存储空间。

B. 利用数据统计

尽管RS数据流优化了所有数据类型的数据移动，但其固有的数据量和计算量仍然较高。为了进一步提高能效，我们探索了CNN的数据统计特性，具体包括：

1) 1）在优化数据流的基础上，采用压缩技术减少DRAM访问次数——压缩是每次访问时能耗最高的数据传输方式；2）跳过冗余计算以节省处理能力（详见第五章C节）。

ReLU函数通过将所有负值滤波结果修正为零，会在特征图中引入大量零值。虽然特征图中的零值数量取决于输入到CNN的数据，但通常会随着网络层数的增加而增多。以AlexNet为例，CONV2层的特征图平均有近40%的值为零，到CONV5层时这一比例会攀升至约75%。除了特征图本身，近期研究还表明，CNN中16%-78%的滤波器权重可以通过剪枝操作被置零[33]。

RLC 技术在Eyeriss架构中被用于利用fmaps中的零值以节省DRAM带宽。图8展示了 RLC 编码的示例：当连续出现长度不超过31位的零值时，会使用5位数字表示 *运行长度* ；随后直接插入16位 *水平位* ，并重新开始计数。每三个运行长度与水平位组合会被打包成64位字，最后一位用于标识该字是否为编码序列的末尾。根据我们在ImageNet数据集上使用AlexNet进行的实验， RLC 的压缩率仅对理论熵极限增加了5%–10%的开销。

除了输入到卷积神经网络（CNN）第一层的数据外，所有滤波器映射（fmaps）都以 RLC 压缩形式存储在动态随机存取存储器（DRAM）中。加速器从DRAM读取编码后的输入滤波器映射（ifmaps），通过 RLC 解码器解压后写入全局流缓冲区（GLB）。计算完成的输出滤波器映射（ofmaps）从GLB读取后，可选经过ReLU模块处理，再由 RLC 编码器压缩后传输回DRAM。这种设计既节省了存储空间，又降低了DRAM的读写带宽消耗。通过AlexNet实验验证，仅fmaps的DRAM访问量（包括输入和输出滤波器映射）在CONV1层可减少近30%，CONV5层则能节省约75%。图9展示了AlexNet在应用 RLC 前后整体DRAM访问量的变化，流量包含滤波器、输入滤波器映射和输出滤波器映射。若将 RLC 技术应用于滤波器权重处理，DRAM访问量还能进一步降低。

V. System Modules

A. 全局缓冲区

Eyeriss加速器的GLB为108 kB，可通过异步接口与DRAM通信

TABLE III

SHAPE PARAMETERS OF AlexNet AND ITS RS DATAFLOW MAPPING PARAMETERS ON EYERISS. THIS MAPPING ASSUMES A BATCH SIZE (N) OF 4

Layer	CNN Shape Parameters						RS Dataflow Mapping Parameters							Global Buffer Allocation	
	H/W[1]	R/S	E/F	C	M	U	m	n	e	p	q	r	t	ifmap	psum
CONV1	227	11	55	3	96	4	96	1	7	16	1	1	2	15.5KB	72.2KB
CONV2	31	5	27	48	256	1	64	1	27	16	2	1	1	3.8KB	91.1KB
CONV3	15	3	13	256	384	1	64	4	13	16	4	1	4	7.0KB	84.5KB
CONV4	15	3	13	192	384	1	64	4	13	16	3	2	2	10.5KB	84.5KB
CONV5	15	3	13	192	256	1	64	4	13	16	3	2	2	10.5KB	84.5KB

[1] This is the padded size

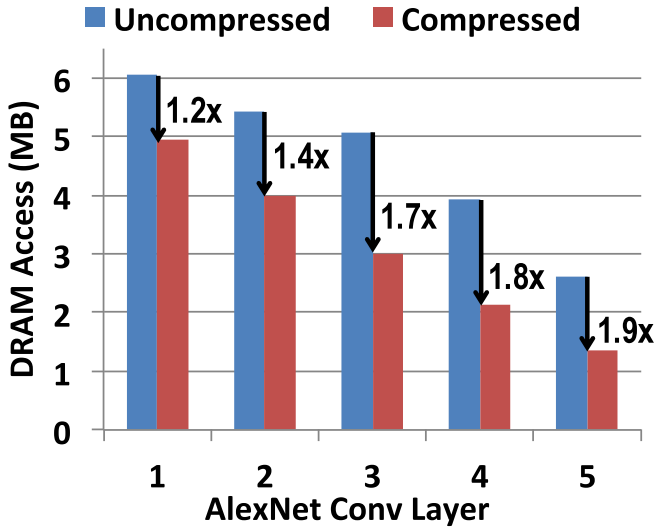


Fig. 9. Comparison of DRAM accesses (read and write), including filters, ifmaps, and ofmaps, before and after using RLC in the five CONV layers of AlexNet.

and with the PE array through the NoC. The GLB stores all the three types of data: ifmaps, filters, and psums/ofmaps. 100 kB of the GLB is allocated for ifmaps and psums as required by the RS dataflow for reuse. Even though it is not required by the dataflow, the remaining 8 kB (two banks of 512-b × 64-b SRAMs) of the GLB is allocated for filter weights to compensate for insufficient off-chip traffic bandwidth. While the PE array is working on a processing pass, the GLB preloads the filters used by the next processing pass.

The 100-kB storage space for ifmaps and psums has to be reconfigurable to fit the two data types in different proportions for supporting different shapes (Table III). It also has to provide enough bandwidth for accesses from the PE array. To meet the two demands, the space is divided into 25 banks, each of which is a 512-b ×64-b (4 kB) SRAM. Each bank is assigned entirely to ifmaps or psums, and the assignment is reconfigurable based on the scan chain bits. Therefore, the PE array can access both ifmaps and psums simultaneously, each from one of the 25 banks.

B. Network-on-Chip

The NoC manages data delivery between the GLB and the PE array as well as between different PEs. The NoC architecture needs to meet the following goals. First, the NoC has to support the data delivery patterns used in the RS dataflow. While the data movement within a PE set is uniform (Fig. 4), there are three scenarios in the mapping of real CNNs that can break the uniformity and should be taken care of: 1) different convolution strides (U) result in the ifmap delivery, skipping certain rows in the array (AlexNet CONV1 in Fig. 5); 2) a set is divided into segments that are mapped onto different parts of the PE array (AlexNet CONV2 in Fig. 5); and 3) multiple sets are mapped onto the array simultaneously and different data is required for each set (AlexNet CONV4 and CONV5 in Fig. 5). Second, the NoC should leverage the data reuse achieved by the RS dataflow to further improve energy efficiency. Third, it has to provide enough bandwidth for data delivery in order to support the highly parallel processing in the PE array.

Conventional approaches usually use hop-by-hop mesh NoC at the cost of increased ramp-up time and router overhead [34], [35]. To avoid this overhead, we chose to implement a custom NoC for the required data delivery patterns that is optimized for latency, bandwidth, energy, and area. The custom NoC comprises three different types of networks as described in the following.

1) Global Input Network: The global input network (GIN) is optimized for a single-cycle multicast from the GLB to a group of PEs that receive the same filter weight, ifmap value, or psum. Fig. 5 shows an example of ifmap delivery in AlexNet. The challenge is that the group of destination PEs varies across layers due to the differences in data type, convolution stride, and mapping. Broadcasting each data with a bit-vector tag of the same size of the PE array (i.e., 168 b), which indicates the IDs of destination PEs, can support any arbitrary mapping. However, doing so is also very costly in terms of both area and energy consumption due to the increased GIN bus width. Instead, we implemented the GIN, as shown in Fig. 10, with two levels of hierarchy: Y-bus and X-bus. A vertical Y-bus consists of 12 horizontal X-buses, one at each row of the PE array, and each X-bus connects to 14 PEs in the row. Each X-bus has a row ID, and each PE has a col ID. These IDs are all *reconfigurable*, and a unique ID is given to each group of X-buses or PEs that receives the same data in a given CNN layer. Each data read from the GLB is augmented with a (row, col) tag by the top-level controller, and the GIN guarantees that the data are delivered to *all and only* the X-buses and then PEs with the ID that matches the tag within a single cycle. The tag-ID matching is done using the *Multicast Controller* (MC). There are 12 MCs on the

表格 III

AlexNet 的 参数 与 I TS RS 数据流 的 参数 在 E YERISS 上的 对比 A B ATCH S IZE（N）4

层	CNN 形状参数						RS数据流映射参数							全局缓冲区分配	
	H/W^1	R/S	E/F	C	M	U	m	n	e	p	q	r	t	如果地图	psum
CONV1	227	11	55	3	96	4	96	1	7	16	1	1	2	15.5KB	72.2KB
CONV2	31	5	27	48	256	1	64	1	27	16	2	1	1	3.8KB	91.1KB
CONV3	15	3	13	256	384	1	64	4	13	16	4	1	4	7.0KB	84.5KB
CONV4	15	3	13	192	384	1	64	4	13	16	3	2	2	10.5KB	84.5KB
CONV5	15	3	13	192	256	1	64	4	13	16	3	2	2	10.5KB	84.5KB

这是加厚尺寸

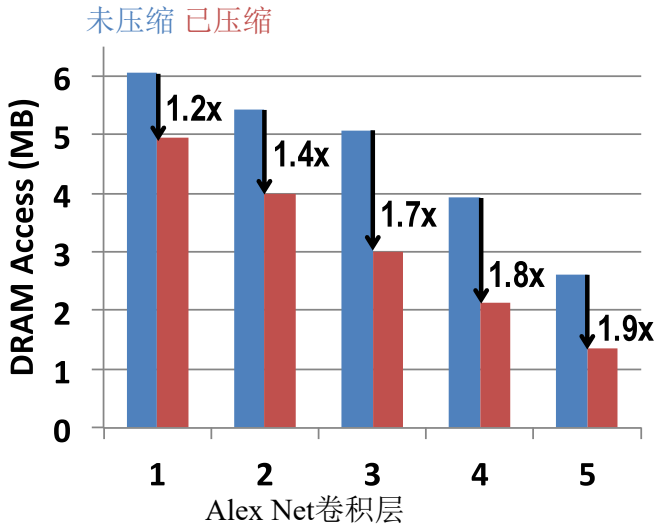


图9对比了AlexNet网络五个卷积层在启用 RLC 前后，包含滤波器、ifmap和ofmap的DRAM读写访问情况。

通过NoC与PE阵列进行通信。GLB存储所有三种类型的数据：ifmaps、filters以及psums/ofmaps。

根据RS数据流的复用需求，GLB中预留了100KB空间用于ifmap和psum运算。尽管数据流本身并未要求这部分存储，但GLB中剩余的8KB空间（由两组512位×64位SRAM组成）被专门分配给滤波器权重，用于弥补芯片外带宽不足的问题。当处理单元阵列执行当前处理周期时，GLB会预先加载下一周期所需的滤波器。为支持不同数据结构（见表III），需将ifmap和psum的100KB存储空间进行可重构配置，使其能按不同比例适配两种数据类型。同时，该存储空间还需确保提供足够的带宽，以满足处理单元阵列的访问需求。

为满足两项需求，该存储空间被划分为25个存储库，每个存储库为512字节×64字节（4KB）的 SRAM 。每个存储库完全分配给ifmap或psum操作，且分配方式可根据扫描链位进行动态调整。因此，PE阵列可同时访问ifmap和psum操作，每个操作均可从25个存储库中的任意一个进行调用。

B. 片上网络

网络核心（NoC）负责管理通用逻辑块（GLB）与处理执行器（PE）阵列之间的数据传输，以及不同处理执行器之间的数据传输。该架构需满足以下目标：首先，网络核心（NoC）必须支持卷积神经网络（CNN）资源池（RS）数据流中的数据传输模式。虽然处理器执行单元（PE）组内的数据传输是均匀的（图4），但在实际CNN映射中存在三种可能破坏这种均匀性的场景需要特别处理：1）不同的卷积步长（U）会导致ifmap传输时跳过数组中的某些行（如图5所示的AlexNet CONV1）；2）一组数据被分割成多个段，分别映射到PE数组的不同区域（如图5所示的AlexNet CONV2）；3）多个数据组同时映射到数组上，且每个组需要不同的数据（如图5所示的AlexNet CONV4和CONV5）。其次，网络核心应充分利用RS数据流实现的数据复用，进一步提升能效。第三，它必须提供足够的带宽支持数据传输，以满足PE数组中高度并行处理的需求。

传统方法通常采用逐跳网状NoC架构，但会增加系统启动时间并带来路由器开销[34][35]。为避免这些开销，我们针对所需的数据传输模式设计了定制化NoC架构，该架构在延迟、带宽、能耗和面积等方面均经过优化。具体而言，该定制化NoC由以下三种不同类型的网络构成。

1) 全球输入网络： 全球输入网络（GIN）专为从全局流表（GLB）向一组具有相同滤波器权重、ifmap值或psum值的处理单元（PE）进行单周期组播而优化。图5展示了AlexNet中ifmap传输的示例。其核心挑战在于：由于数据类型、卷积步长和映射方式的差异，目标PE组会随着网络层级变化而不同。若采用与PE阵列同尺寸（即168位）的位向量标签广播数据（该标签用于标识目标PE的ID），则可支持任意映射关系。但这种方式会因GIN总线宽度增加而显著提升面积和功耗成本。为此，我们如图10所示，采用Y总线与X总线的双层级架构实现GIN。垂直Y总线由12条水平X总线构成，每行PE阵列对应一条X总线，每条X总线连接该行14个PE。每条X总线具有 *行ID* ，每个PE拥有 *列ID* 。所有ID均可 *重新配置* ，且在特定CNN层中接收相同数据的X总线组或PE组将获得唯一ID标识。顶层控制器为从GLB读取的每个数据添加 *（行号，列号）* 标签，GIN确保数据在单个周期内仅传输至 *所有且唯一* 的X总线，再由匹配该标签的PE接收。标签与ID的匹配通过 *组播控制器* （MC）实现。系统中共配置12个MC。

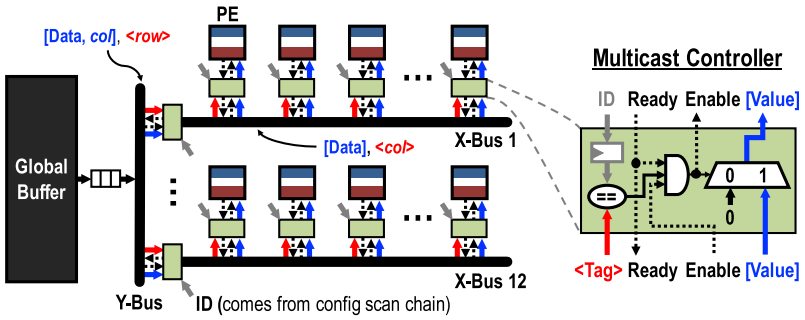


Fig. 10. Architecture of the GIN.

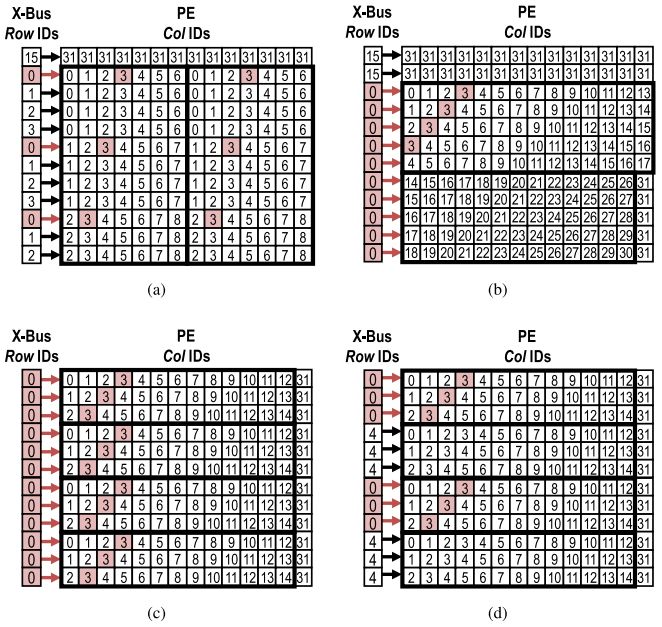


Fig. 11. *row* IDs of the X-buses and *col* IDs of the PEs for ifmap delivery using GIN in AlexNet layers. (a) CONV1. (b) CONV2. (c) CONV3. (d) CONV4 and CONV5. In this example, assuming the tag on the data has $row = 0$ and $col = 3$, the X-buses and PEs in red are the activated buses and PEs that receive the data, respectively.

Y-bus to compare the *row* tag with the *row* ID of each X-bus, and 14 MCs on each of the X-buses to compare the *col* tag with the *col* ID of each PE. The unmatched X-buses and PEs are gated to save energy. For flow control, the data are passed from the GLB down to the GIN only when all destination PEs have issued a ready signal. An example of the *row* and *col* ID setup for ifmap delivery using GIN in AlexNet is shown in Fig. 11.

Eyeriss has separate GINs for each of the three data types (filter, ifmap, and psum) to provide sufficient bandwidth from the GLB to the PE array. All GINs have 4-b *row* IDs to address the 12 rows. The filter and psum GINs use 4-b *col* IDs to address the 14 columns, while ifmap GIN uses 5 b to support maximum 32 ifmap rows passing in diagonal. The filter and psum GINs have data bus width of 64 b ($4b \times 16$ b), while the ifmap GIN has the data bus width of 16 b.

2) Global Output Network: The GON is used to read the psums generated by a processing pass from the PE array back to the GLB. The GON has the same architecture as the GIN; only the direction of data transfer is reversed. The data bus width is also 64b as the psum GIN.

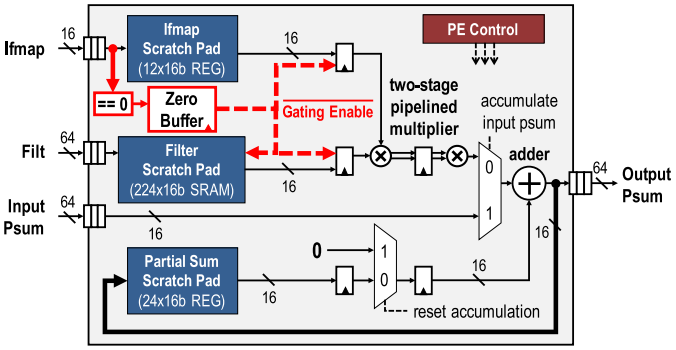


Fig. 12. PE architecture. The datapaths in red show the data gating logic to skip the processing of zero ifmap data.

3) Local Network: Between every pair of PEs that are on two consecutive rows of the same column, a dedicated 64b data bus is implemented to pass the psums from the bottom PE to the top PE directly. Therefore, a PE can get its input psums either from the psum GIN or LN. The selection is static within a layer, which is controlled by the scan chain configuration bits and only depends on the dataflow mapping of the CNN shape.

C. Processing Element and Data Gating

Fig. 12 shows the architecture of a PE. FIFOs are used at the I/O of each PE to balance the workload between the NoC and the computation. The numbers of filters (p) and channels (q) that the PE processes at once are statically configured into the control of a PE, which determines the state of processing. This configuration controls the pattern with which the PE steps through the three spads. The datapath is pipelined into three stages: one stage for spad access, and the remaining two for computation. The computation consists of a 16-b two-stage pipelined multiplier and adder. Since the multiplication results are truncated from 32 to 16 b, the selection of 16 b out of the 32 b is configurable, and can be decided by the dynamic range of a layer from offline experiments. Spads are separated for three data types to provide enough access bandwidth. The filter spad is implemented in a 224-b $\times$16-b SRAM due to its large size; the ifmap and psum spads of size 12 b $\times$16 b and 24 b $\times$16 b, respectively, are implemented using registers.

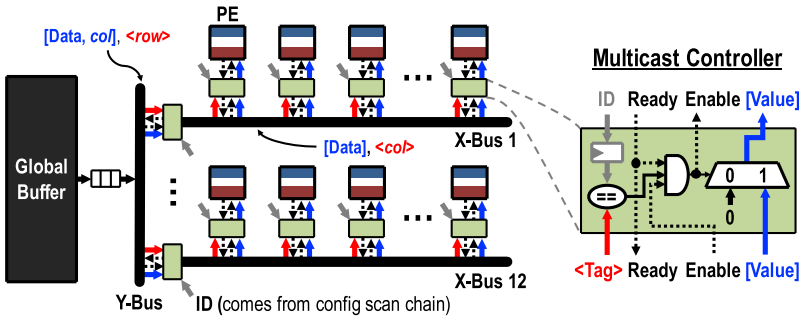


图10.GIN架构。

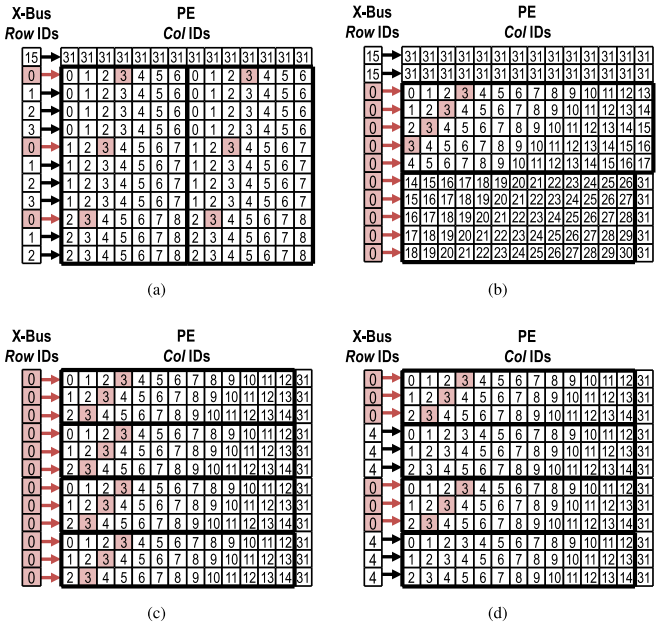


图11. AlexNet各层使用GIN进行ifmap传输时的X总线 *行* ID和 *PE列* ID示意图。(a)CONV1；(b)CONV2；(c)CONV3；(d)CONV4和CONV5。本示例中，假设数据标签的 *行* =0且 *列* =3，红色标注的X总线和PE分别表示激活的总线和接收数据的PE。

通过Y总线将 *行标签* 与各X总线的 *行ID* 进行比对，并在每个X总线上配置14个MC单元，用于将 *列标签* 与各处理单元（PE）的 *列ID* 进行比对。未匹配的X总线和PE会被门控以节省能耗。在流量控制方面，只有当所有目标PE都发出就绪信号时，数据才会从全局标签缓冲器（GLB）传递到全局输入节点（GIN）。图11展示了AlexNet中使用GIN进行ifmap传输时， *行* 和 *列ID* 的配置示例。

Eyeriss为三种数据类型（滤波器、ifmap和psum）分别配置独立的通用输入节点（GIN），以确保从通用逻辑块（GLB）到处理执行器（PE）阵列的带宽充足。所有GIN均采用4 *字节行* ID来寻址12行存储单元。其中滤波器和psum GIN使用4 *字节列* ID寻址14列存储单元，而ifmap GIN采用5字节寻址方案，支持最多32行ifmap数据沿对角线输入。滤波器和psum GIN的数据总线宽度为64字节（4 *字节行* ×16字节列），ifmap GIN的数据总线宽度则为16字节。

2) 全局输出网络： GON用于读取PE阵列处理通道生成的psums并将其返回GLB。GON的架构与GIN相同；

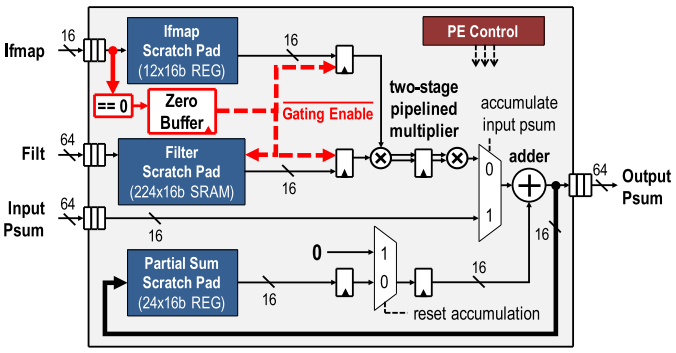


图12. PE架构。红色数据路径展示了数据门控逻辑，用于跳过对零ifmap数据的处理。

仅数据传输方向被反转，数据总线宽度仍为64位，与psum GIN相同。

3) 局域网络架构： 在同一列相邻两行的处理单元（PE）之间，通过专用的64位数据总线实现从底层PE到顶层PE的直接数据传输。因此，每个PE既可通过psum全局输入（GIN）也可通过局部输入（LN）获取输入psum值。该选择机制在单层内保持静态，由扫描链配置位控制，其具体选择仅取决于CNN结构的数据流映射关系。

C. 处理元件和数据门控

图12展示了处理器的架构设计。每个处理器的输入输出端口均采用先进先出（FIFO）结构，用于平衡网络通信与计算任务的负载分配。处理器可同时处理的滤波器数量（p）和通道数量（q）通过静态配置参数进行控制，这些参数决定了处理器的工作状态。该配置参数控制着处理器在三个信号处理单元（SPAD）之间的切换模式。数据路径采用三阶段流水线设计：第一阶段负责SPAD访问，其余两阶段用于计算处理。计算模块包含16位双阶段流水线乘法器和加法器。由于乘法结果需从32位截断为16位，因此从32位中选择16位的配置参数可根据离线实验确定的层动态范围进行调整。为确保足够的访问带宽，处理器针对三种数据类型分别配置了独立的SPAD单元。其中，滤波器SPAD因体积较大采用224位×16位的 SRAM 架构实现；而ifmap和psum SPAD分别采用12位×16位和24位×16位的寄存器架构实现。

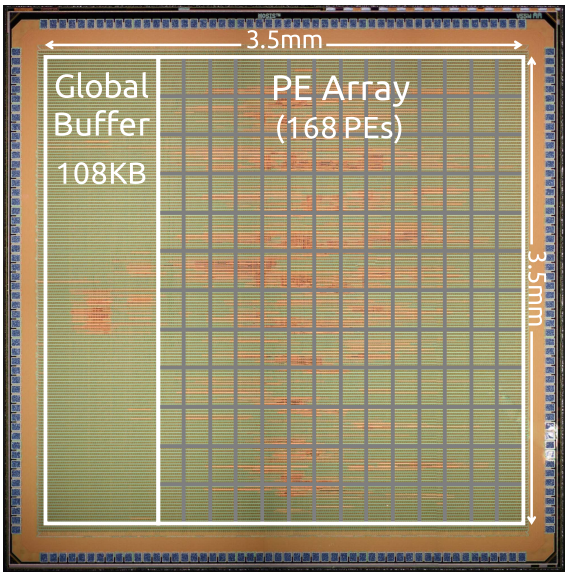


Fig. 13. Die micrograph and floorplan of the Eyeriss chip.

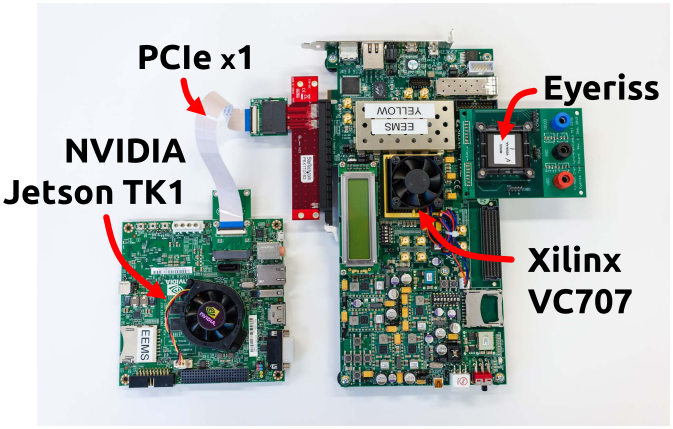


Fig. 14. Eyeriss-integrated deep-learning system that runs Caffe [28], which is one of the most popular deep-learning frameworks. The customized Caffe runs on the NVIDIA Jetson TK1 development board, and offloads the processing of a CNN layer to Eyeriss through the PCIe interface. The Xilinx VC707 serves as the PCIe controller and does not perform any processing. We have demonstrated an 1000-class image classification task [27] using this system, and a live demo can be found in [29].

Data gating logic is implemented to exploit zeros in the ifmap for saving processing power. An extra 12-b *Zero Buffer* is used to record the position of zeros in the ifmap spad. If a zero ifmap value is detected from the zero buffer, the gating logic will disable the read of the filter spad and prevent the MAC datapath from switching. Compared with the PE design without the data gating logic, it can save the PE power consumption by 45%.

VI. Results

The Eyeriss chip shown in Fig. 13 was implemented in 65-nm CMOS [36] and had been integrated into the Caffe framework [28] (Fig. 14). Table IV lists a summary of the chip specifications. At 1 V, the peak throughput is 33.6 GMAC/s (GMACS) with a 200-MHz core clock. Most of the state-of-the-art CNNs have shapes that lie within the native support of Eyeriss, so they can easily leverage Eyeriss for acceleration with no modification required.

Fig. 15(a) shows the area breakdown of the Eyeriss core, i.e., the area without I/O pads. It includes the logic cells, registers, and SRAMs from both the core and link clock domains. The area of the PE array includes all 168 PEs, and the area breakdown of each PE is shown in Fig. 15(b). The spads from all PEs take nearly half of the total area, which is 2.5 × larger than that of the GLB. However, the aggregated capacity of the spads is 1.5 times smaller than the size of the GLB. Overall, the on-chip storage, including the GLB and all spads, takes two-thirds of the total area while the multipliers and adders from all 168 PEs only account for 7.4%.

We benchmark the chip performance using two publicly available and widely used CNNs: AlexNet [2] and VGG-16 [3]. The input frames are resized according to the requirement of each CNN: 227 × 227 for AlexNet and 224 × 224 for VGG-16. A batch size (N) of 4 and 3 is used for AlexNet and VGG-16, respectively; these batch sizes deliver the highest energy efficiency on Eyeriss according to the optimization in [32].

TABLE IV
Chip Specifications

Technology	TSMC 65nm LP 1P9M
Chip Size	4.0 mm × 4.0 mm
Core Area	3.5 mm × 3.5 mm
Gate Count (logic only)	1176k (2-input NAND)
On-Chip SRAM	181.5K bytes
Number of PEs	168
Global Buffer	108.0K bytes (SRAM)
Scratch Pads (per PE)	filter weights: 448 bytes (SRAM) feature maps: 24 bytes (Registers) partial sums: 48 bytes (Registers)
Supply Voltage	core: 0.82–1.17 V I/O: 1.8 V
Clock Rate	core: 100–250 MHz link: up to 90 MHz
Peak Throughput	16.8–42.0 GMACS
Arithmetic Precision	16-bit fixed-point
Natively Supported CNN Shapes	filter height (R): 1–12 filter width (S): 1–32 num. of filters (M): 1–1024 num. of channels (C): 1–1024 vertical stride: 1, 2, 4 horizontal stride: 1–12

A. AlexNet

Table V shows the measured performance breakdown of the five CONV layers in AlexNet at 1 V. The chip power consumption gradually decreases through deeper layers, since data gating can leverage more zeros in the ifmaps. On average, the Eyeriss chip achieves a frame rate of 34.7 frames/s, or equivalently a processing throughput of 23.1 GMACS. The measured chip power is 278 mW, and the corresponding energy efficiency is 83.1 GMACS/W. The actual throughput is lower than the peak throughput for three reasons: 1) only 88% of the PEs are active; 2) it takes time to load data from the

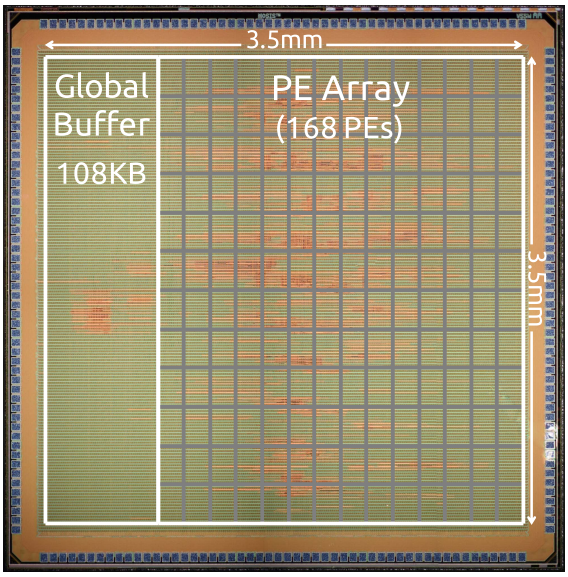


图13. Eyeriss芯片的显微照片与平面图。

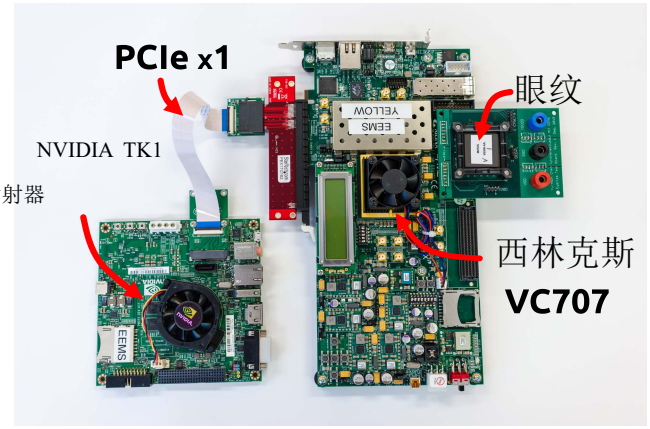


图14展示了集成Eyeriss的深度学习系统，该系统运行Caffe框架[28]——这是当前最主流的深度学习框架之一。定制版Caffe运行在 NVIDIA Jetson TK1开发板上，通过PCIe接口将CNN层的处理任务卸载给Eyeriss。Xilinx VC707作为PCIe控制器不参与任何处理。我们已通过该系统成功完成1000类图像分类任务[27]，实时演示版本可参考[29]。

为节省处理功耗，系统采用数据门控逻辑来利用ifmap中的零值。通过额外配置12位 *零值缓冲区* 记录ifmap spad中的零值位置，当检测到零值时，门控逻辑会禁用滤波器spad的读取操作，并阻止MAC数据通路切换。与未采用数据门控逻辑的PE设计相比，该方案可使PE功耗降低45%。

第六 章 ESULTS

图13展示的Eyeriss芯片采用65纳米CMOS工艺[36]制造，并已集成到Caffe框架[28]中（图14）。表IV汇总了该芯片的规格参数。在1伏电压下，其峰值吞吐量达到33.6 GMAC /s（GMACS），核心时钟频率为200 MHz。由于大多数前沿卷积神经网络（CNN）的结构都符合Eyeriss的原生支持，因此无需任何修改即可直接利用该芯片进行加速处理。

图15(a)展示了Eyeriss核心的面积分布情况，即不含输入/输出焊盘的区域。该区域包含来自核心域和链路时钟域的逻辑单元、寄存器及静态随机存取存储器（SRAM）。PE阵列的面积包含全部168个处理单元，各处理单元的具体面积分布详见图15(b)。所有处理单元的存储器占芯片总面积的近半，其面积是通用逻辑块（GLB）的2.5倍。然而，存储器的总容量仅为GLB的1.5倍。总体而言，包含GLB和所有存储器的片上存储器占芯片总面积的三分之二，而来自168个处理单元的乘法器和加法器仅占7.4%。

我们采用两种公开且广泛使用的卷积神经网络（CNN）——AlexNet[2]和 VGG -16[3]来评估芯片性能。输入帧的尺寸根据各网络需求调整：AlexNet采用227×227， VGG -16采用224×224。AlexNet和 VGG -16分别使用4和3的批量大小（N），根据文献[32]的优化研究，这两种批量大小在Eyeriss架构上能实现最高的能效。

表IV

C HIP S 规格

技术	TSMC 65nm LP 1P9M
小片尺寸	4.0 mm × 4.0 mm
核心区域	3.5 mm × 3.5 mm
仅限门计数逻辑）	1176k（2输入NAND）
片上sRAM	181.5 千字节
pES数量	168
全局缓冲区	108.0k 字节（SRAM）
刮痕垫（每 pEB）	滤波器权重：448字节（SRAM）；特征图：24字节（寄存器）；部分和：48字节（寄存器）
电源电压	核心电压：0.82-1.17 V I/O: 1.8 V
时钟速率	核心：100-250 MHz 链接：最高可达MHz
峰值吞吐量	16.8–42.0 GMACS
算术精度	16 位 fx ed-point
原生支持的 CNN 形状	滤波器高度(R)： 1-12 滤波器宽度（秒）：1-32 滤波器数量 (M)：1 - 1024； 通道数（c）：1 - 1024；垂直 步长：1 、2 、4；水平步长： 1-12

A. AlexNet

表V展示了AlexNet网络中五个CONV层在1伏电压下的性能测量数据。随着网络层深度增加，芯片功耗呈现逐层递减趋势，这得益于数据门控技术能更充分地利用ifmap中的零值。Eyeriss芯片平均帧率为34.7帧/秒，相当于23.1 GMACS 的处理吞吐量。实测功耗为278毫瓦，能效达到83.1 GMACS /W。实际吞吐量低于峰值值主要受以下三方面影响：1）处理器执行单元（PE）仅有88%处于激活状态；2）数据从...加载需要时间。

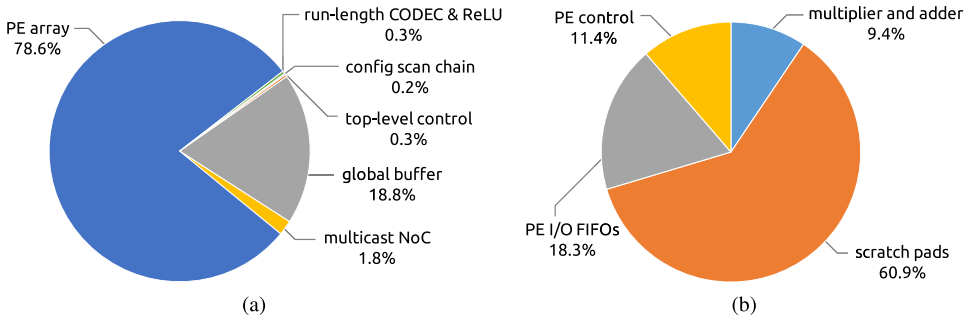


Fig. 15. Area breakdown of (a) Eyeriss core and (b) PE.

TABLE V

PERFORMANCE BREAKDOWN OF THE FIVE CONV LAYERS IN AlexNet AT 1 V. BATCH SIZE (N) IS 4. THE CORE AND LINK CLOCKS RUN AT 200 AND 60 MHz, RESPECTIVELY

Layer	Power (mW)	Total Latency (ms)	Processing Latency (ms)	Num. of MACs	Num. of Active PEs	Zeros in Ifmaps (%)	Global Buff. Accesses	DRAM Accesses
CONV1	332	20.9	16.5	0.42G	154 (92%)	0.01%	18.5 MB	5.0 MB
CONV2	288	41.9	39.2	0.90G	135 (80%)	38.7%	77.6 MB	4.0 MB
CONV3	266	23.6	21.8	0.60G	156 (93%)	72.5%	50.2 MB	3.0 MB
CONV4	235	18.4	16.0	0.45G	156 (93%)	79.3%	37.4 MB	2.1 MB
CONV5	236	10.5	10.0	0.30G	156 (93%)	77.6%	24.9 MB	1.3 MB
Total	**278**	**115.3**	**103.5**	**2.66G**	**148 (88%)**	**57.53%**	**208.5 MB**	**15.4 MB**

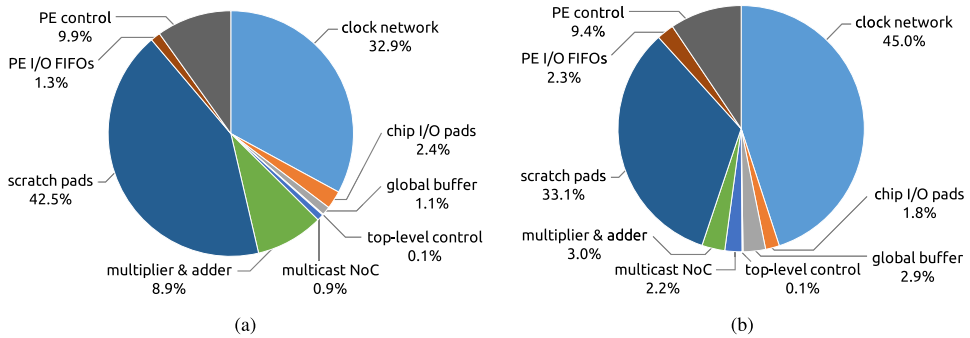


Fig. 16. Power breakdown of the chip running layer. (a) CONV1 and (b) CONV5 of AlexNet.

GLB into the PE array to ramp up each processing pass; and 3) the chip does not perform processing while it is loading ifmap from DRAM or dumping ofmaps to DRAM. The last point, nevertheless, can be optimized with the refined control of the DRAM traffic at negligible cost. Therefore, we also provide the *Processing Latency* in Table V that shows the performance when DRAM traffic is fully overlapped with processing. For a batch of four frames, the required DRAM access is 15.4 MB, or 0.0029 access/MAC (37.4 access/input pixel).[2]

Fig. 16 shows the power breakdown of the chip running CONV1 and CONV5. This is obtained by performing post-place and route simulations using actual workloads as in chip measurement. Different dataflow mappings and data reuse patterns result in different power distributions. Specifically, the power consumed in the spads as well as multipliers and adders is much lower in CONV5 than CONV1 due to the zeros in ifmaps. Overall, the ALUs only account for less than 10% of the total power, while data movement related components, including spads, GLB, and NoC, account for up to 45%. This confirms that data movement is more energy consuming than

[2]Each access is for a 16-b data value (fmaps or filters).

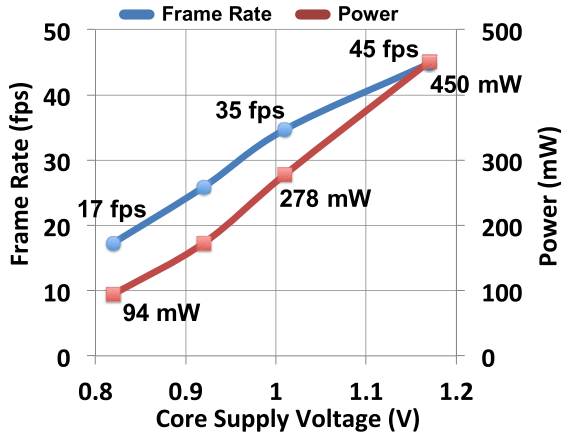
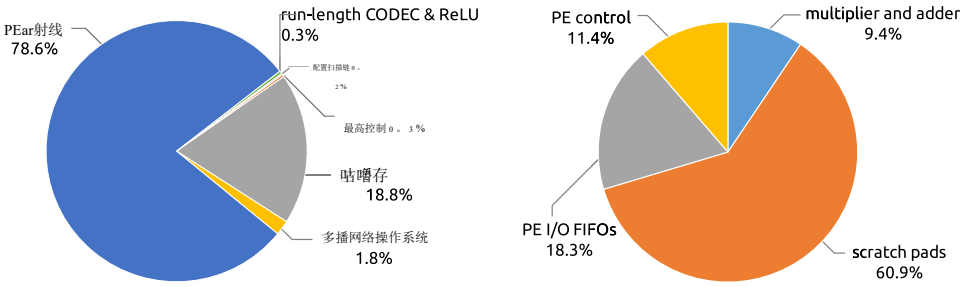


Fig. 17. Impact of voltage scaling on AlexNet Performance.



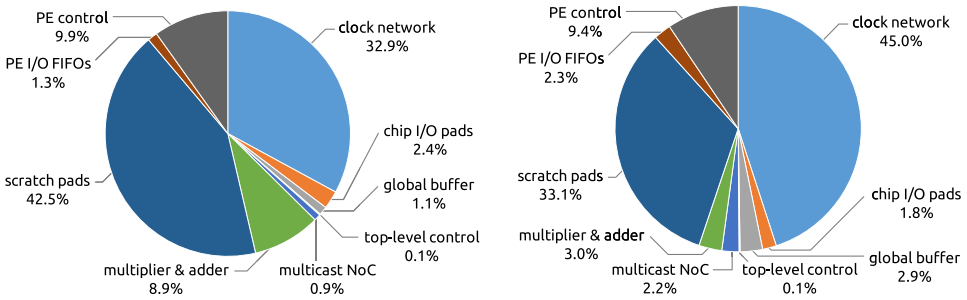
(a) (b)

图15. (a) Eyeriss核心与(b) PE的面积分布。

表V

AlexNet B 的 F 比 速 1V ATCH S IZE（*N*）I S 4 . T HE C ORE AND L INK C locks R UN AT 200 AND 60MHz，R RESPECTIVELY

层	权力 (mW)	总延迟（毫秒）	处 理 延 迟（ms）	MACS数量	活性PES 的数量	0 s和 如果地图（%）	全局缓冲区。 访问	DRAM 访问
CONV1	332	20.9	16.5	0.42G	154 (92%)	0.01%	18.5 MB	5.0 MB
CONV2	288	41.9	39.2	0.90G	135 (80%)	38.7%	77.6 MB	4.0 MB
CONV3	266	23.6	21.8	0.60G	156 (93%)	72.5%	50.2 MB	3.0 MB
CONV4	235	18.4	16.0	0.45G	156 (93%)	79.3%	37.4 MB	2.1 MB
CONV5	236	10.5	10.0	0.30G	156 (93%)	77.6%	24.9 MB	1.3 MB
总计	**278**	**115.3**	**103.5**	**2.66G**	**148 (88%)**	**57.53%**	**208.5 MB**	**15.4 MB**



(a) (b)

图16. AlexNet网络中CONV1和CONV5的运行层功率分布。(a) CONV1和(b) CONV5的运行层功率分布。

将全局光场（GLB）输入PE阵列以逐步提升每次处理通道的处理量；3）当芯片正在从DRAM加载ifmap或向DRAM写入maps时，不会执行任何处理操作。不过，最后一个环节可以通过精细化控制DRAM流量以微乎其微的成本进行优化。因此，我们在表V中还提供了 *处理延迟* 数据，该数据展示了DRAM流量与处理完全同步时的性能表现。对于四帧的批量处理，所需的DRAM访问量为15.4 MB，即0.0029次访问/MAC（37.4次访问/输入像素）。[2]

图16展示了运行CONV1和CONV5的芯片功耗分布情况。该数据通过实际工作负载进行的后置布局布线仿真获得，与芯片测量结果一致。不同的数据流映射和数据复用模式会导致功耗分布差异。具体而言，由于ifmap中的零值特性，CONV5中SPAD单元、乘法器和加法器的功耗显著低于CONV1。总体而言，ALU单元的功耗占比不足10%。

2每个访问权限对应16位数据值（即fmaps或滤波器）。

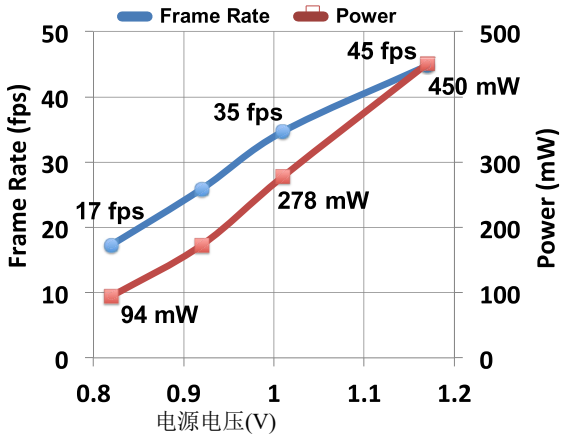


图17. 电压缩放对AlexNet性能的影响。

在总功耗中，数据移动相关组件（包括spads、GLB和NoC）的能耗占比高达45%。这表明数据移动比其他操作更耗电。

TABLE VI
PERFORMANCE BREAKDOWN OF THE 13 CONV LAYERS IN VGG-16 AT 1 V. BATCH SIZE (N) IS 3. THE CORE AND LINK CLOCKS RUN AT 200 AND 60 MHz, RESPECTIVELY

Layer	Power (mW)	Total Latency (ms)	Processing Latency (ms)	Num. of MACs	Num. of Active PEs	Zeros in Ifmaps (%)	Global Buff. Accesses	DRAM Accesses
CONV1-1	247	76.2	38.0	0.26G	156 (93%)	1.6%	112.6 MB	15.4 MB
CONV1-2	218	910.3	810.6	5.55G	156 (93%)	47.7%	2402.8 MB	54.0 MB
CONV2-1	242	470.3	405.3	2.77G	156 (93%)	24.8%	1201.4 MB	33.4 MB
CONV2-2	231	894.3	810.8	5.55G	156 (93%)	38.7%	2402.8 MB	48.5 MB
CONV3-1	254	241.1	204.0	2.77G	156 (93%)	39.7%	607.4 MB	20.2 MB
CONV3-2	235	460.9	408.1	5.55G	156 (93%)	58.1%	1214.8 MB	32.2 MB
CONV3-3	233	457.7	408.1	5.55G	156 (93%)	58.7%	1214.8 MB	30.8 MB
CONV4-1	278	135.8	105.1	2.77G	168 (100%)	64.3%	321.8 MB	17.8 MB
CONV4-2	261	254.8	210.0	5.55G	168 (100%)	74.7%	643.7 MB	28.6 MB
CONV4-3	240	246.3	210.0	5.55G	168 (100%)	85.4%	643.7 MB	22.8 MB
CONV5-1	258	54.3	48.3	1.39G	168 (100%)	79.4%	90.0 MB	6.3 MB
CONV5-2	236	53.7	48.5	1.39G	168 (100%)	87.4%	90.0 MB	5.7 MB
CONV5-3	230	53.7	48.5	1.39G	168 (100%)	88.5%	90.0 MB	5.6 MB
Total	**236**	**4309.5**	**3755.2**	**46.04G**	**158 (94%)**	**58.6%**	**11035.8 MB**	**321.1 MB**

computation. Besides the clock network, the spads dominate on-chip power consumption, which shows that RS dataflow effectively reuses data locally for reducing DRAM accesses and optimizing overall system energy efficiency as estimated in [32]. This is also why looking at the chip power alone is not sufficient to assess the energy efficiency of the system. Fig. 17 shows the impact of voltage scaling on-chip performance running AlexNet. The maximum throughput is 45 frames/s at 1.17 V, and the maximum energy efficiency is 122.8 GMACS/W at 0.82 V.

B. VGG-16

Table VI shows the measured performance breakdown of the 13 CONV layers in VGG-16 at 1 V. On average, the chip operates at 0.7 frames/s with a measured power consumption of 236 mW. The frame rate is lower than that of AlexNet mainly since VGG-16 requires 23 times more computations per frame than AlexNet. The performance, however, depends not only on the computation but also on the shape configuration. For example, CONV1-2 and CONV4-2 have the same amount of MAC operations, but the former takes nearly four times longer to process than the latter. This is because the early layers require more processing passes than the deeper layers. Therefore, it spends more time on ramping up the processing in the PE array. The large number of processing passes is dictated by the large fmap size. The required DRAM access for a batch of three frames is 321.1 MB, or 0.0035 access/MAC (1066.6 access/input pixel).

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25. 2012, pp. 1097–1105.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, pp. 1–14, Sep. 2014.

[4] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2014, pp. 580–587.

[7] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated recognition, localization and detection using convolutional networks," *CoRR*, vol. abs/1312.6229, pp. 1–16, Dec. 2013.

[8] M. Bojarski *et al.* (2016). "End to end learning for self-driving cars." [Online]. Available: https://arxiv.org/abs/1604.07316

[9] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[10] R. Hameed *et al.*, "Understanding sources of inefficiency in general-purpose chips," in *Proc. 37th Annu. Int. Symp. Comput. Archit.*, 2010, pp. 37–47.

[11] M. Horowitz, "Computing's energy problem (and what we can do about it)," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers (ISSCC)*, Feb. 2014, pp. 10–14.

[12] M. Sankaradas *et al.*, "A massively parallel coprocessor for convolutional neural networks," in *Proc. 20th IEEE Int. Conf. Appl.-Specific Syst., Archit. Process.*, Jul. 2009, pp. 53–60.

[13] V. Sriram, D. Cox, K. H. Tsoi, and W. Luk, "Towards an embedded biologically-inspired machine vision processor," in *Proc. Int. Conf. Field-Program. Technol. (FPT)*, Dec. 2010, pp. 273–278.

[14] S. Chakradhar, M. Sankaradas, V. Jakkula, and S. Cadambi, "A dynamically configurable coprocessor for convolutional neural networks," in *Proc. 37th Annu. Int. Symp. Comput. Archit.*, 2010, pp. 247–257.

[15] M. Peemen, A. A. A. Setio, B. Mesman, and H. Corporaal, "Memory-centric accelerator design for convolutional neural networks," in *Proc. IEEE 31st Int. Conf. Comput. Design (ICCD)*, Oct. 2013, pp. 13–19.

[16] V. Gokhale, J. Jin, A. Dundar, B. Martini, and E. Culurciello, "A 240 G-ops/s mobile coprocessor for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2014, pp. 696–701.

[17] T. Chen *et al.*, "DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," in *Proc. 19th Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, 2014, pp. 269–284.

[18] Z. Du *et al.*, "ShiDianNao: Shifting vision processing closer to the sensor," in *Proc. 42nd Annu. Int. Symp. Comput. Archit.*, 2015, pp. 92–104.

[19] Y. Chen *et al.*, "DaDianNao: A machine-learning supercomputer," in *Proc. 47th Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2014, pp. 609–622.

[20] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," *CoRR*, vol. abs/1502.02551, pp. 1–10, Feb. 2015.

[21] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, 2015, pp. 161–170.

[22] F. Conti and L. Benini, "A ultra-low-energy convolution engine for fast brain-inspired vision in multicore clusters," in *Proc. Design, Autom. Test Eur. Conf. Exhibit.*, 2015, pp. 683–688.

表VI

13<

层	权力 (mW)	总延迟（毫秒）	处理延迟（ms）	MACS数量	活性pES 的 数量	0 s和 如果地图（%）	全局缓冲区。 访问	DRAM 访问
CONV1-1	247	76.2	38.0	0.26G	156 (93%)	1.6%	112.6 MB	15.4 MB
CONV1-2	218	910.3	810.6	5.55G	156 (93%)	47.7%	2402.8 MB	54.0 MB
CONV2-1	242	470.3	405.3	2.77G	156 (93%)	24.8%	1201.4 MB	33.4 MB
CONV2-2	231	894.3	810.8	5.55G	156 (93%)	38.7%	2402.8 MB	48.5 MB
CONV3-1	254	241.1	204.0	2.77G	156 (93%)	39.7%	607.4 MB	20.2 MB
CONV3-2	235	460.9	408.1	5.55G	156 (93%)	58.1%	1214.8 MB	32.2 MB
CONV3-3	233	457.7	408.1	5.55G	156 (93%)	58.7%	1214.8 MB	30.8 MB
CONV4-1	278	135.8	105.1	2.77G	168 (100%)	64.3%	321.8 MB	17.8 MB
CONV4-2	261	254.8	210.0	5.55G	168 (100%)	74.7%	643.7 MB	28.6 MB
CONV4-3	240	246.3	210.0	5.55G	168 (100%)	85.4%	643.7 MB	22.8 MB
CONV5-1	258	54.3	48.3	1.39G	168 (100%)	79.4%	90.0 MB	6.3 MB
CONV5-2	236	53.7	48.5	1.39G	168 (100%)	87.4%	90.0 MB	5.7 MB
CONV5-3	230	53.7	48.5	1.39G	168 (100%)	88.5%	90.0 MB	5.6 MB
总计	**236**	**4309.5**	**3755.2**	**46.04G**	**158 (94%)**	**58.6%**	**11035.8 MB**	**321.1 MB**

计算性能方面，除了时钟网络外，片上动态随机存取存储器（DRAM）的动态电压调节（spad）模块主导着芯片功耗。这表明RS数据流技术能有效实现数据本地化复用，既减少了DRAM访问次数，又优化了系统整体能效，相关评估详见文献[32]。这也解释了为何仅凭芯片功耗数据无法全面评估系统能效。图17展示了电压调节对AlexNet模型运行时片上性能的影响：在1.17伏电压下，最大吞吐量可达45帧/秒；而当电压降至0.82伏时，系统能效最高可达122.8 GMACS /W。

B. VGG-16

表VI展示了 VGG -16在1V电压下13个卷积层的性能测量数据。该芯片平均帧率为0.7帧/秒，实测功耗为236毫瓦。其帧率低于AlexNet，主要原因是 VGG -16每帧需要进行的计算量是AlexNet的23倍。不过性能不仅取决于计算量，还与结构配置密切相关。例如，CONV1-2和CONV 4-2的MAC运算量相同，但前者处理时间却比后者长近四倍。这是因为早期层需要比深层进行更多次处理通道，因此在PE阵列中需要更长时间来提升处理能力。大量处理通道的产生源于较大的fmap尺寸。处理三帧数据所需的DRAM访问量为321.1MB，相当于每MAC运算需要0.0035次访问（1066.6次访问/输入像素）。

参考

[1] Y. LeCun，Y. Bengio，和G. Hinton，“深度学习”， *Nature*，第521卷，第436-444页，2015年5月。

[2] A. Krizhevsky，I. Sutskever，和G.E. Hinton，“用深度卷积神经网络进行ImageNet分类”，在 *先进神经信息处理系统会议论文集* ，第25卷，2012年，第1097-1105页。

[3] K. Simonyan和A. Zisserman，“用于大规模图像识别的非常深的卷积网络”， *CoRR*，卷 abs/1409.1556，第1-14页，2014年9月。

[4] C. Szegedy 等人 ，“用卷积深入挖掘”，在 *IEEE计算机视觉与模式识别会议论文集（CVPR）*，2015年6月，第1-9页。

[5] K. He，X. Zhang，S. Ren，and J. Sun，“Deep residual learning for image recognition，” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.（CVPR）*，2016.

[6] R. Girshick，J. Donahue，T. Darrell，and J. Malik，“精确物体检测和语义分割的丰富特征层次结构”，在 *IEEE计算机视觉和模式识别会议论文集（CVPR）*，2014年6月，第580-587页。

[7] P. Sermanet，D. Eigen，X. Zhang，M. Mathieu，R. Fergus，and Y. LeCun，“OverFeat： Integrated recognition，localization and detection using convolutional networks，” *CoRR*，vol. abs/1312.6229，pp. 1–16，Dec. 2013.

[8] M. Bojarski 等人（2016）。“自动驾驶汽车的端到端学习”。[在线]。可获取： https://arxiv.org/abs/1604.07316

[9] D. Silver 等人 ，“用深度神经网络和树搜索掌握围棋游戏，” *Nature*，第529卷，第7587号，第484-489页，2016年1月。

[10] R. Hameed 等人 ，“理解通用芯片中效率低下的原因”，在 *第37届国际计算机体系结构年会论文集* ，2010年，第37-47页。

[11] M. Horowitz，“计算机的能源问题（以及我们可以做些什么）”，在 *IEEE国际固态电路会议技术论文集（ISSCC）*，2014年2月，第10-14页。

[12] M. Sankaradas 等人 ，“用于卷积神经网络的大量并行协处理器”，在 *第20届IEEE国际会议 Appl.-Specific 系统、架构和处理* 论文集，2009年7月，第53-60页。

[13] V. Sriram，D. Cox，K. H. Tsoi和W. Luk，“迈向嵌入式生物启发式机器视觉处理器”，在 *国际现场程序技术会议论文集（FPT）*，2010年12月，第273-278页。

[14] S. Chakradhar，M. Sankaradas，V. Jakkula和S. Cadambi，“用于卷积神经网络的动态可配置协处理器”，在 *第37届国际计算机架构年会论文集* ，2010年，第247-257页。

[15] M. Peemen，A. A. A. Setio，B. Mesman，和 H. Corporaal，“卷积神经网络的内存中心加速器设计”，在 *IEEE第31届国际计算机设计会议（ICCD）论文集* ，2013年10月，第13-19页。

[16] V. Gokhale，J. Jin，A. Dundar，B. Martini，and E. Culurciello，“用于深度神经网络的240 G-ops/s移动协处理器”，在 *IEEE计算机视觉和模式识别会议论文集（CVPRW）*，2014年6月，第696-701页。

[17] T. Chen 等人 ，“DianNao：一种小型高通量加速器，用于无处不在的机器学习”，在 *第19届国际会议架构支持程序。语言操作系统* ，2014年，第269-284页。

[18] Z. Du 等人 ，“ShiDianNao：将视觉处理移近传感器”，在 *第42届国际计算机架构年会论文集* ，2015年，第92-104页。

[19] Y. Chen 等人 ，“DaDianNao：一台机器学习超级计算机”，在 *第47届IEEE/ACM国际微架构年会论文集* ，2014年，第609-622页。

[20] S. Gupta，A. Agrawal，K. Gopalakrishnan和P. Narayanan，“有限数值精度的深度学习”， *CoRR*，卷 abs/1502.02551，第1-10页，2015年2月。

[21] C. Zhang，P. Li，G. Sun，Y. Guan，B. Xiao，and J. Cong，“Optimizing FPGA -based accelerator design for deep convolutional neural networks，” in *Proc. ACM/ SIGDA Int. Symp. Field-Program. Gate Arrays*，2015，pp. 161–170.

[22] F. Conti和L. Benini，“用于多核集群中快速脑启发视觉的超低能耗卷积引擎”，在 *Proc. Design，Autom. Test Eur. Conf. Exhibit.*，2015，pp. 683-688。

[23] S. Park, K. Bong, D. Shin, J. Lee, S. Choi, and H.-J. Yoo, "A 1.93TOPS/W scalable deep learning/inference processor with tetra-parallel MIMD architecture for big-data applications," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, Feb. 2015, pp. 1–3.
[24] L. Cavigelli, D. Gschwend, C. Mayer, S. Willi, B. Muheim, and L. Benini, "Origami: A convolutional network accelerator," in *Proc. 25th Ed. Great Lakes Symp. VLSI*, 2015, pp. 199–204.
[25] J. Sim, J.-S. Park, M. Kim, D. Bae, Y. Choi, and L.-S. Kim, "A 1.42TOPS/W deep convolutional neural network recognition processor for intelligent IoE systems," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, Jan./Feb. 2016, pp. 264–265.
[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[27] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[28] Y. Jia *et al.* (2014). "Caffe: Convolutional architecture for fast feature embedding." [Online]. Available: https://arxiv.org/abs/1408.5093
[29] Y.-H. Chen. *An 1000-Class Image Classification Task Performed by the Eyeriss-Integrated Deep Learning System*, accessed on 2016. [Online]. Available: https://vimeo.com/154012013
[30] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May/Jun. 2010, pp. 253–256.
[31] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.
[32] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *Proc. 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, 2016, pp. 367–379.
[33] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28. 2015, pp. 1135–1143.
[34] J. Howard *et al.*, "A 48-core IA-32 message-passing processor with DVFS in 45 nm CMOS," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers (ISSCC)*, Feb. 2010, pp. 108–109.
[35] B. K. Daya *et al.*, "SCORPIO: A 36-core research chip demonstrating snoopy coherence on a scalable mesh NoC with in-network ordering," in *Proc. 41st Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2014, pp. 25–36.
[36] Y.-H. Chen, T. Krishna, J. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers (ISSCC)*, Jan./Feb. 2016, pp. 262–263.

Yu-Hsin Chen (S'11) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2009, and the M.S. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2013, where he is currently pursuing the Ph.D. degree with the focus on the architecture design and hardware implementation for deep learning accelerators.

His current research interests include energy-efficient VLSI system design, computer vision, and digital signal processing.

Mr. Chen was a recipient of the 2015 NVIDIA Graduate Fellowship and the 2015 ADI Outstanding Student Designer Award.

Tushar Krishna (S'08–M'15) received the B.Tech. degree in electrical engineering from IIT Delhi, New Delhi, India, in 2007, the M.S.E. degree in electrical engineering from Princeton University, Princeton, NJ, USA, in 2009, and the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2014.

He was a Researcher with the VSSAD Group, Intel, Hudson, MA, USA, from 2013 to 2015, and a Post-Doctoral Researcher with the MIT SMART-LEES Center in 2015. He has been an Assistant Professor with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA, since 2015. His current research interests include computer architecture, interconnection networks, on-chip networks, HPC, and reconfigurable architectures.

Joel S. Emer (M'73–SM'03–F'04) received the B.S. (Hons.) and M.S. degrees in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1974 and 1975, respectively, and the Ph.D. degree in electrical engineering from the University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 1979.

He was with Intel, where he was an Intel Fellow and the Director of Microarchitecture Research. At Intel, he led the VSSAD Group, which he had previously been a member of at Compaq and Digital Equipment Corporation. He is currently a Senior Distinguished Research Scientist with the Nvidia's Architecture Research Group, Westford, MA, USA, where he is responsible for exploration of future architectures and modeling and analysis methodologies. He is also a Professor of the Practice at the Massachusetts Institute of Technology, Cambridge, MA, USA, where he teaches computer architecture and supervises graduate students. He has held various research and advanced development positions investigating processor microarchitecture and developing performance modeling and evaluation techniques. He has made architectural contributions to a number of VAX, Alpha, and X86 processors and is recognized as one of the developers of the widely employed quantitative approach to processor performance evaluation. He has been recognized for his contributions in the advancement of simultaneous multithreading technology, processor reliability analysis, cache organization, and spatial architectures for deep learning.

Dr. Emer is a Fellow of the ACM. He has been a recipient of numerous public recognitions. In 2009, he received the Eckert-Mauchly Award for lifetime contributions in computer architecture, the Purdue University Outstanding Electrical and Computer Engineer Alumni Award, and the University of Illinois Electrical and Computer Engineering Distinguished Alumni Award in 2010 and 2011, respectively. His 1996 paper on simultaneous multithreading received the ACM/SIGARCH-IEEE-CS/TCCA: Most Influential Paper Award in 2011. He was named to the ISCA and Micro Halls of Fame in 2005 and 2015, respectively. He has had five papers selected for the IEEE Micro's Top Picks in Computer Architecture, in 2003, 2004, 2007, 2013, and 2015. He was the Program Chair of ISCA in 2000. He is the 2017 Program Chair for Micro.

Vivienne Sze (S'04–M'10–SM'16) received the B.A.Sc. (Hons) degree from the University of Toronto, Toronto, ON, Canada, in 2004, and the S.M. and Ph.D. degree from the Massachusetts Institute of Technology (MIT), Cambridge, MA, in 2006 and 2010, respectively, all in electrical engineering.

She has been an Assistant Professor with MIT in the Electrical Engineering and Computer Science Department since August 2013. Her research interests include energy-aware signal processing algorithms and low-power circuit and system design for portable multimedia applications. Prior to joining MIT, she was a Member of Technical Staff with the Systems and Applications R&D Center, Texas Instruments (TI), Dallas, TX, USA, where she designed low-power algorithms and architectures for video coding. She also represented TI at the international JCT-VC standardization body developing HEVC. Within the committee, she was the primary coordinator of the core experiment on coefficient scanning and coding, and has chaired/vice-chaired several ad hoc groups on entropy coding. She is a coeditor of *High Efficiency Video Coding (HEVC): Algorithms and Architectures* (Springer, 2014).

Prof. Sze was a recipient of the 2016 AFOSR Young Investigator Research Program (YIP) Award, 2016 3M Non-Tenured Faculty Award, 2014 DARPA Young Faculty Award, 2007 DAC/ISSCC Student Design Contest Award and a co-recipient of the 2008 A-SSCC Outstanding Design Award. In 2011, she received the Jin-Au Kong Outstanding Doctoral Thesis Prize in Electrical Engineering at MIT. She received the Natural Sciences and Engineering Research Council of Canada (NSERC) Julie Payette fellowship in 2004, the NSERC Postgraduate Scholarships in 2005 and 2007, and the Texas Instruments Graduate Women's Fellowship for Leadership in Microelectronics in 2008.

[23] S. Park，K. Bong，D. Shin，J. Lee，S. Choi，and H.-J. Yoo，“用于大数据应用的1.93TOPS/W可扩展深度学习/推理处理器，具有四并行 MIMD 架构”，在 *IEEE国际固态电路会议论文集（ISSCC）*，2015年2月，第1-3页。

[24] L. Cavigelli，D. Gschwend，C. Mayer，S. Willi，B. Muheim和L. Benini，“Origami：一个卷积网络加速器”，在 *第25届大湖VLSI研讨会论文集*，2015年，第199-204页。

[25] J. Sim，J.-S. Park，M. Kim，D. Bae，Y. Choi，and L.-S. Kim，“用于智能IoE系统的1.42TOPS/W深度卷积神经网络识别处理器”，在 *IEEE国际固态电路会议论文集（ISSCC）*，2016年1月/2月，第264-265页。

[26] Y. LeCun、L. Bottou、Y. Bengio和P. Haffner，“基于梯度的学习应用于文档识别”，*IEEE学报*，第86卷，第11期，第2278-2324页，1998年11月。

[27] O. Russakovsky *等人*，“ImageNet大规模视觉识别挑战”，*Int.J. Comput.Vis.*，第115卷，第3期，第211-252页，2015年12月。

[28] Y. Jia *等人*（2014）。“Caffe：快速特征嵌入的卷积架构。”[在线]。可获取：https://arxiv.org/abs/1408.5093

[29] 陈耀宏. *Eyeriss集成深度学习系统完成的千类图像分类任务*，2016年访问。[在线]。可获取：https://vimeo.com/154012013

[30] Y. LeCun、K. Kavukcuoglu和C. Farabet，“卷积网络和视觉中的应用”，*IEEE国际电路系统研讨会论文集（ISCAS）*，2010年5/6月，第253-256页。

[31] V. Nair和G. E. Hinton，“修正线性单元改进受限玻尔兹曼机”，在 *第27届国际机器学习会议（ICML）*，2010年，第807-814页。

[32] Y.-H. Chen，J. Emer，和 V. Sze，“Eyeriss：一种用于卷积神经网络的节能数据流的空间架构，” *第43届国际计算机架构年会（ISCA）论文集*，2016，第367-379页。

[33] S. Han，J. Pool，J. Tran和W. Dally，“学习权重和连接以实现高效的神经网络”，在 *先进神经信息处理系统会议论文集*，第28卷，2015年，第1135-1143页。

[34] J. Howard *等人*，“48核IA-32消息传递处理器与45nmCMOS中的DVFS”，在 *IEEE国际固态电路会议技术论文集（ISSCC）*，2010年2月，第108-109页。

[35] B. K. Daya *等人*，“scorpio：一个36核研究芯片，展示了可扩展网格NoC上的snoopy相干性，具有网络内排序”，在 *第41届国际计算机架构年会（ISCA）*，2014年6月，第25-36页。

[36] Y.-H. Chen，T. Krishna，J. Emer和V. Sze，“Eyeriss：一种用于深度卷积神经网络的节能可重构加速器”，在 *IEEE国际固态电路会议技术论文集（ISSCC）*，2016年1月/2月，第262-263页。

陈宇欣（S ‘11）于2009年获得台湾大学（台北）电气工程学士学位，2013年获得麻省理工学院（美国马萨诸塞州剑桥市）电气工程与计算机科学硕士学位，目前正攻读博士学位，研究方向为深度学习加速器的架构设计与硬件实现。

他目前的研究兴趣包括节能VLSI系统设计、计算机视觉和数字信号处理

陈先生曾荣获2015年 NVIDIA 研究生奖学金及ADI杰出学生设计师奖。

Tushar Krishna（S ‘08–M’ 15）于2007年在印度新德里的印度理工学院获得电气工程学士学位，于2009年在新泽西州普林斯顿的普林斯顿大学获得电气工程硕士学位，于2014年在马萨诸塞州剑桥市的麻省理工学院获得电气工程和计算机科学博士学位。

2013至2015年，他任职于美国马萨诸塞州哈德逊市英特尔公司 VSSAD 研究组，2015年加入麻省理工学院SMART-LEES中心担任博士后研究员。自2015年起，他担任美国佐治亚理工学院电气与计算机工程学院助理教授。其研究领域涵盖计算机体系结构、互连网络、片上网络、 HPC 及可重构架构。

Joel S. Emer（M ‘73-SM’ 03-F ‘04）获得了理学学士学位。1974年和1975年分别获得美国印第安纳州西拉斐特市普渡大学电气工程荣誉学士和硕士学位，1979年获得美国伊利诺伊州尚佩恩市伊利诺伊大学厄巴纳-尚佩恩分校电气工程博士学位。

他曾任职于英特尔公司，担任英特尔研究员及微架构研究总监。在英特尔期间，他领导了VSSAD 团队——该团队此前由他在康柏和数字设备公司时就曾参与。目前，他担任美国马萨诸塞州韦斯特福德市英伟达架构研究组的高级杰出研究科学家，负责未来架构探索及建模分析方法的研究工作。同时，他还是美国马萨诸塞州剑桥市麻省理工学院的实践教授，讲授计算机体系结构课程并指导研究生。他曾在多个研究与高级开发岗位上深耕处理器微架构领域，致力于性能建模与评估技术的研发。其在VAX、Alpha和X86处理器架构方面作出重要贡献，被公认为处理器性能评估量化方法的主要开发者之一。他因在多线程同步技术、处理器可靠性分析、缓存组织和深度学习空间架构方面的贡献而受到认可。

埃默博士是ACM会士，曾荣获多项公开表彰。2009年，他因在计算机体系结构领域的毕生贡献获得埃克特-莫奇利奖；2010年和2011年，他分别获得普渡大学杰出电气与计算机工程校友奖和伊利诺伊大学电气与计算机工程杰出校友奖。其1996年发表的《同步多线程》论文在2011年荣获ACM/sigarch-IEEE-CS/ TCCA 最具影响力论文奖。他分别于2005年和2015年入选 ISCA 与Micro名人堂。其五篇论文入选IEEE Micro《计算机体系结构精选》榜单，分别发表于2003年、2004年、2007年、2013年和2015年。2000年，他担任 ISCA 大会主席。2017年，他再次出任Micro大会主席。

Vivienne Sze（S ‘04-M’ 10-SM ‘16）于2004年在加拿大多伦多大学获得理学学士（荣誉）学位，于2006年和2010年分别在麻省理工学院（MIT）获得硕士和博士学位，均在电气工程领域。

自2013年8月起，她担任麻省理工学院电气工程与计算机科学系助理教授。研究领域涵盖节能信号处理算法及便携式多媒体应用的低功耗电路与系统设计。加入麻省理工学院前，她曾在美国德州仪器公司（TI）系统与应用研发中心担任技术成员，负责视频编码领域的低功耗算法与架构设计。她还代表TI参与国际 JCT -VC标准化组织 HEVC 标准的制定工作，在该委员会中担任系数扫描与编码核心实验的首席协调员，并主持/副主任多个熵编码专项工作组。她还是《*高效视频编码（HEVC）：算法与架构*》（Springer出版社，2014年）的合编者。

施教授曾荣获多项重要奖项：2016年 AFOSR 青年研究者计划（YIP）奖、2016年3M非终身教职奖、2014年DARPA青年教师奖、2007年DAC/ISSCC 学生设计大赛奖，以及2008年A- SSCC 杰出设计奖的共同获得者。2011年，她荣获麻省理工学院电气工程领域金-奥孔杰出博士论文奖。学术生涯中，她还曾获得加拿大自然科学与工程研究理事会（NSERC）朱莉·佩特特奖学金（2004年）、 NSERC 研究生奖学金（2005年和2007年），以及德州仪器研究生女性领袖力奖学金（2008年）。