Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices

Yu-Hsin Chen, *Student Member, IEEE*, Tien-Ju Yang, *Student Member, IEEE*, Joel S. Emer, *Fellow, IEEE*, and Vivienne Sze, *Senior Member, IEEE*

***Abstract*— A recent trend in deep neural network (DNN) development is to extend the reach of deep learning applications to platforms that are more resource and energy-constrained, e.g., mobile devices. These endeavors aim to reduce the DNN model size and improve the hardware processing efficiency and have resulted in DNNs that are much more *compact* in their structures and/or have high data *sparsity*. These compact or sparse models are different from the traditional large ones in that there is much more variation in their layer shapes and sizes and often require specialized hardware to exploit sparsity for performance improvement. Therefore, many DNN accelerators designed for large DNNs do not perform well on these models. In this paper, we present Eyeriss v2, a DNN accelerator architecture designed for running compact and sparse DNNs. To deal with the widely varying layer shapes and sizes, it introduces a highly flexible on-chip network, called hierarchical mesh, that can adapt to the different amounts of data reuse and bandwidth requirements of different data types, which improves the utilization of the computation resources. Furthermore, Eyeriss v2 can process sparse data directly in the compressed domain for both weights and activations and therefore is able to improve both processing speed and energy efficiency with sparse models. Overall, with sparse MobileNet, Eyeriss v2 in a 65-nm CMOS process achieves a throughput of 1470.6 inferences/s and 2560.3 inferences/J at a batch size of 1, which is 12.6× faster and 2.5× more energy-efficient than the original Eyeriss running MobileNet.**

***Index Terms*— Deep neural network accelerators, deep learning, energy-efficient accelerators, dataflow processing, spatial architecture.**

I. Introduction

The development of deep neural networks (DNNs) has shown tremendous progress in improving accuracy over the past few years [1]. In addition, there has been an increasing effort to reduce the computational complexity of DNNs, particularly for those targeted at mobile devices [2]. Various different techniques have been widely explored in the design of DNN models including reduced precision of weights and activations [3]–[8], compact network architectures [9]–[11] (i.e., *compact* DNNs), and increasing sparsity in the filter weights [12]–[14] (i.e., *sparse* DNNs). While these approaches provide theoretical reductions in the size and number of operations and storage cost, specialized hardware is often necessary to translate these theoretical benefits into measurable improvements in energy efficiency and processing speed.

Support for reduced precision has been demonstrated in recent hardware implementations, including Envision [15], Thinker [16], UNPU [17], Loom [18], and Stripes [19]. These works have shown various methods that efficiently translate reduced bitwidth from 16-bits down to 1-bit into both energy savings and increase in processing speed. Specialized hardware for binary networks have also been widely explored [20]–[24]. In this work, we focus on complementary approaches that have been less explored, specifically the support for diverse filter shapes for compact DNNs, as well as support for processing in the compressed domain for sparse DNNs. While compact and sparse DNNs have fewer operations and weights, they also introduce new challenges in hardware design for DNN acceleration.

A. Challenges for Compact DNNs

The trend for compact networks is evident in how the iconic DNNs have evolved over time. Early models, such as AlexNet [25] and VGG [26], are now considered *large* and over-parameterized. Techniques such as using deeper but narrower network structures and bottleneck layers were proposed to pursue higher accuracy while restricting the size of the DNN (e.g., GoogLeNet [9] and ResNet [27]). This quest further continued with a focus on drastically reducing the amount of computation, specifically the number of multiply-and-accumulates (MACs), and the storage cost, specifically the number of weights. Techniques such as filter decomposition as shown in Fig. 1 have since become popular for building *compact* DNNs targeted at mobile devices (e.g., SqueezeNet [11] and MobileNet [10]). This evolution has resulted in a more diverse set of DNNs with widely varying shapes and sizes.

One effect of compact DNNs is that *any data dimension in a DNN layer can diminish*. In addition, due to latency constraints, it is increasingly desirable to run DNNs at smaller batch sizes (i.e., smaller N). Table I summarizes the data dimensions that are used to describe a DNN layer and the common reasons for each dimension to diminish. This suggests that less assumptions can be made on the dimensions of a DNN layer.

For hardware designers, widely varying DNN layer shapes, especially diminishing dimensions, is challenging as they

Manuscript received December 7, 2018; revised February 26, 2019; accepted March 19, 2019. Date of publication April 11, 2019; date of current version June 11, 2019. This work was supported by the DARPA Young Faculty Award, MIT's Center for Integrated Circuits and Systems, and gifts from Google, Intel, and Nvidia. This paper was recommended by Guest Editor B. Murmann. *(Corresponding author: Yu-Hsin Chen.)*

Y.-H. Chen, T.-J. Yang, and V. Sze are with the Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: yhchen@mit.edu).

J. S. Emer is with the Massachusetts Institute of Technology, Cambridge, MA 02139 USA, and also with Nvidia, 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/JETCAS.2019.2910232

Eyeriss v2：移动设备上新兴深度神经网络的灵活加速器

Chen Yu-Hsin ® IEEE学生会员，Yang Tien-Ju ® IEEE学生会员，Joel S. Emer，*IEEE会士*，
和Vivienne Sze，*IEEE高级会员*

***摘要*——深度神经网络（DNN）领域近期的发展趋势，是将深度学习应用的触角延伸至资源与能耗更为受限的平台，例如移动设备。这些研究致力于缩小 DNN 模型体积并提升硬件处理效率，已催生出结构*更紧凑*或具有高*数据稀疏性*的深度神经网络。这类紧凑型或稀疏型模型与传统大型模型存在显著差异：其层结构与尺寸变化更为多样，且常需专用硬件来利用稀疏性提升性能。因此，许多专为大型DNN设计的 DNN 加速器在这些模型上表现欠佳。本文提出Eyeriss v2架构，这是一种专为运行紧凑型和稀疏型DNN设计的 DNN 加速器。为应对层结构与尺寸的广泛差异，该架构引入了高度灵活的片上网络——分层网格，可根据不同类型数据的重用程度和带宽需求进行自适应调整，从而优化计算资源利用率。此外，Eyeriss v2能够在压缩域直接处理稀疏数据，同时处理权重和激活值，从而显著提升稀疏模型的运算速度和能效。总体而言，采用稀疏MobileNet架构时，Eyeriss v2在65纳米CMOS工艺中实现1470.6次/秒的推理吞吐量和2560.3次/焦耳的功耗，批量大小为1时，其运算速度较原版Eyeriss运行MobileNet快12.6倍，能效提升2.5倍。**

***索引术语*——深度神经网络加速器、深度学习、节能加速器、数据流处理、空间架构。**

I. 介绍

近年来，深度神经网络（DNNs）在提升准确率方面取得了长足进步[1]。与此同时，针对移动设备的DNNs，业界也在不断努力降低其计算复杂度[2]。在 DNN 模型设计中，研究者们广泛探索了多种技术方案，包括降低权重和激活值的精度[3]–[8]、采用*紧凑型*网络架构[9]–[11]（即*紧凑型DNNs*），以及提升滤波器的稀疏性。

稿件于2018年12月7日收到，2019年2月26日完成修订，2019年3月19日正式录用。论文发表于2019年4月11日，当前版本日期为2019年6月11日。本研究获得美国国防高级研究计划局青年教师奖、麻省理工学院集成电路与系统中心资助，以及谷歌、英特尔和英伟达公司的技术捐赠支持。本文由客座编辑B·默曼推荐。（*通讯作者：陈宇欣。*）

Y.-H. Chen、T.-J. Yang和V. Sze均任职于美国马萨诸塞州剑桥市麻省理工学院（邮编：02139）（电子邮箱：yhchen@mit.edu）。

J. S. Emer任职于美国马萨诸塞州剑桥市麻省理工学院（邮编：02139），同时兼任美国马萨诸塞州韦斯特福德市英伟达公司（邮编：01886）的职务。

本文中一个或多个图的彩色版本可在http://ieeexplore.ieee.org上获得。

数字对象标识符 10.1109/JETCAS.2019.2910232

[12]–[14]（即*稀疏*DNNs）。虽然这些方法在理论上能有效缩减运算规模、减少操作次数并降低存储成本，但要将这些理论优势转化为实际的能效提升和处理速度优化，通常仍需借助专用硬件来实现。

近期的硬件实现已经证明了对降低精度的支持，包括Envision [15]、Thinker [16]、 UNPU [17]、Loom [18] 和Stripes [19]。这些研究展示了多种有效方法，将16位的位宽减少到1位，既节省了能源又提高了处理速度。针对二进制网络的专用硬件也得到了广泛探索 [20]–[24]。在本工作中，我们关注的是较少被探索的互补方法，特别是支持紧凑型DNN的多种滤波器形状，以及支持稀疏DNN在压缩域中的处理。虽然紧凑型和稀疏DNN的操作和权重较少，但它们也为 DNN 加速的硬件设计带来了新的挑战。

A. 紧凑型深度神经网络面临的挑战

紧凑型网络的发展趋势在经典深度神经网络的演进历程中显而易见。早期模型如AlexNet[25]和 VGG [26]，如今已被认为*结构臃肿*且参数过多。为在限制卷积神经网络（DNN）规模的同时追求更高精度，学界提出了采用更深但更窄的网络结构和瓶颈层等技术（例如GoogLeNet[9]和ResNet[27]）。这一探索进一步聚焦于大幅减少计算量（特别是乘积累加运算MACs的数量）和存储成本（特别是权重数量），如图1所示的滤波器分解技术，已成为构建移动设备专用*紧凑型*DNN的主流方案（例如SqueezeNet[11]和MobileNet[10]）。这种演变催生出形态各异、尺寸多样的深度神经网络体系。

紧凑型深度神经网络（DNN）的一个显著优势在于，*DNN 层中的任何数据维度均可缩减*。此外，考虑到网络延迟的限制，采用更小批量（即更小的N）运行DNN的需求日益凸显。表I汇总了描述 DNN 层的数据维度及其缩减的常见原因，这表明在设计 DNN 层时，我们无需对数据维度做出过多预设。

对于硬件设计师来说， DNN 层形状的广泛变化，特别是尺寸的减小，是一个挑战，因为他们

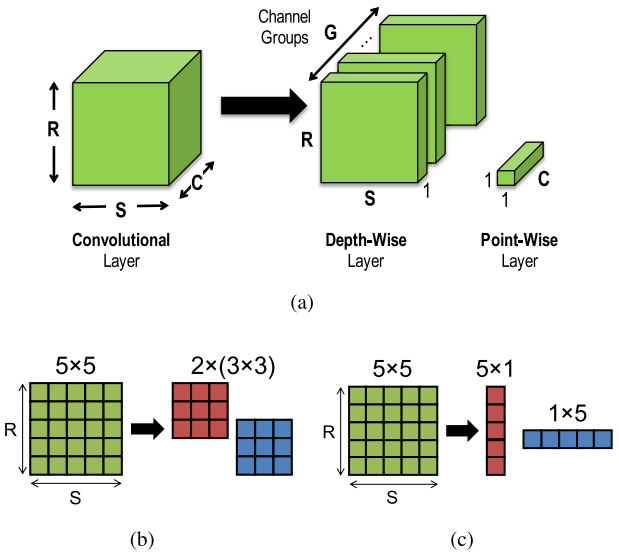


Fig. 1. Various filter decomposition approaches [10], [26], [28].

TABLE I
REASONS FOR DIMINISHING DATA DIMENSIONS IN A DNN LAYER

Data Dimension		Common Reasons for Diminishing Dimension
G	number of channel groups	non-depth-wise layers
N	batch size	low latency requirements
M	number of output channels	(1) bottleneck layers (2) depth-wise layers
C	number of input channels	(1) layers after bottleneck layers (2) depth-wise layers (3) first layer (e.g., 3 in visual inputs)
H / W	input feature map height/width	deeper layers in a DNN
R / S	filter height/width	(1) point-wise layers (i.e., 1×1) (2) decomposed layers (i.e., R×1, 1×S)
E / F	output feature map height/width	(1) deeper layers in a DNN (2) fully-connected (FC) layers

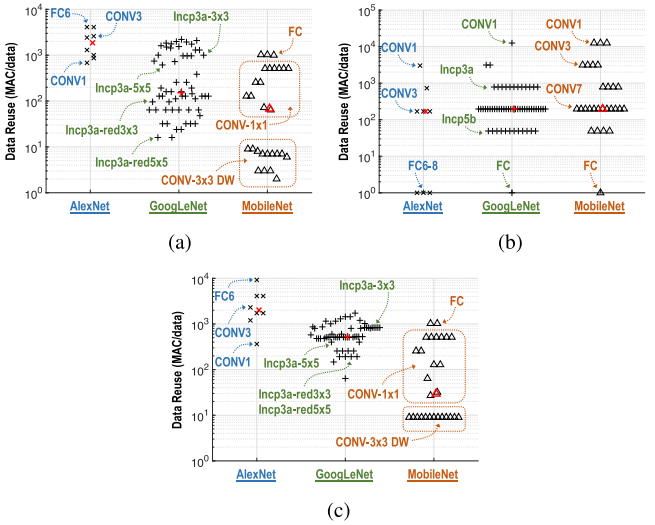


Fig. 2. Data reuse of the three data types in each layer of the three DNNs. Each data point represents a layer, and the red point indicates the median amount of data reuse among all the layers in a DNN. For example, *incp3a-red*5 × 5 means the reduction layer with 5 × 5 filters in Inception module 3a in GoogLeNet. (a) Input activations (iacts). (b) Weights (batch size = 1). (c) Partial sums (psums).

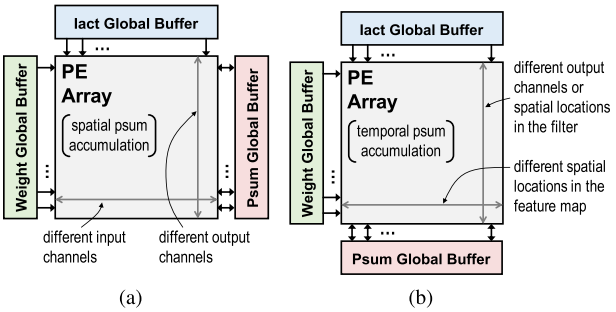


Fig. 3. Two common DNN accelerator designs: (a) Spatial accumulation array [29]–[32]: iacts are reused vertically and psums are accumulated horizontally. (b) Temporal accumulation array [15]–[17]: iacts are reused vertically and weights are reused horizontally.

result in changes in a key property of DNNs: *data reuse*, which is the number of MACs that use the same piece of data, i.e., MACs/data. Most DNN accelerators rely on data reuse as a means to improve efficiency. The amount of data reuse for each of the three data types in a DNN layer, i.e., input activations (iacts), weights and partial sums (psums), is a function of the layer shape and size. For example, the amount of iact reuse is proportional to the number of output channels as well as the filter size in a layer. Therefore, diminished data dimensions suggest that it is more difficult to exploit data reuse from any specific dimension.

Fig. 2 shows that the variation in data reuse increases in all data types in more recent DNNs, and the amount of reuse also decreases in iacts and psums. This variation and overall reduction in data reuse makes the design of DNN accelerators more challenging in two ways.

1) Array Utilization: Many existing DNN accelerators [15]–[17], [29]–[32] rely on a set of pre-selected data dimensions to exploit both high parallelism across an array of processing elements (PEs) for high performance and data reuse for high energy efficiency. For instance, Fig. 3 shows two designs that are commonly used. A spatial accumulation array architecture (Fig. 3a), which is often used for a weight-stationary dataflow, relies on both output and input channels to map the operations spatially onto the PE array to exploit parallelism. At the same time, each iact can be reused across the PE array vertically with weights from different output channels, while psums from the PEs in the same row can be further accumulated spatially together before written back to the global buffer. Similarly, a temporal accumulation array architecture (Fig. 3b), which is often used for a output-stationary dataflow, relies on another set of data dimensions to achieve high compute parallelism. In this case, each iact is still reused vertically across different PEs in the same column, while each weight is reused horizontally across PEs in the same row.

When the set of pre-selected data dimensions diminish due to a change in DNN shapes and sizes, e.g., the number of output channels in a layer (M) is less than the height of the PE array, efficiency decreases. Specifically, these spatial

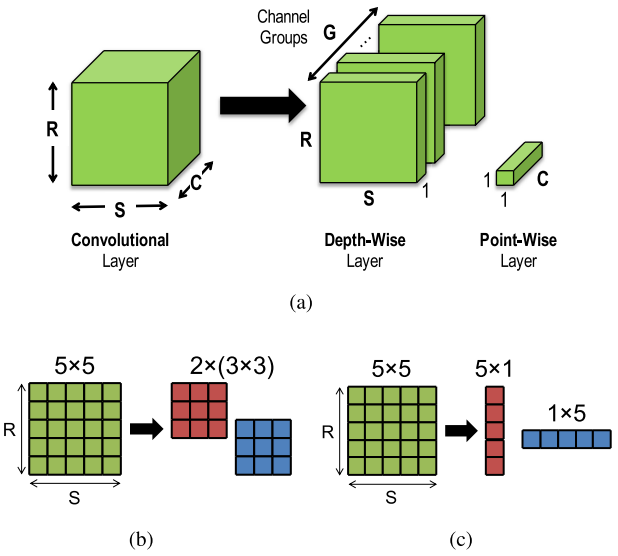


图1. 各种滤波器分解方法[10]、[26]、[28]。

表I

DNN 层中数据维度减少的原因

Data Dimension		Common Reasons for Diminishing Dimension
G	number of channel groups	non-depth-wise layers
N	batch size	low latency requirements
M	number of output channels	(1) bottleneck layers (2) depth-wise layers
C	number of input channels	(1) layers after bottleneck layers (2) depth-wise layers (3) first layer (e.g., 3 in visual inputs)
H / W	input feature map height/width	deeper layers in a DNN
R / S	filter height/width	(1) point-wise layers (i.e., 1×1) (2) decomposed layers (i.e., R×1, 1×S)
E / F	output feature map height/width	(1) deeper layers in a DNN (2) fully-connected (FC) layers

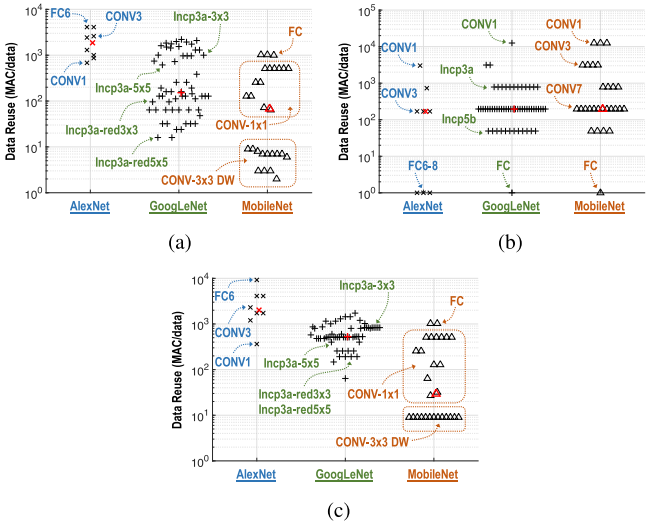


图2展示了三个深度神经网络（DNN）各层中三种数据类型的复用情况。每个数据点代表一个神经层，红色圆点标示了 DNN 中所有层的数据复用中位数。例如，*incp3ared5* × 5表示GoogLeNet模型中Inception模块3a的卷积层，其滤波器尺寸为5×5。(a)输入激活值（iacts）；(b)权重参数（批量大小=1）。(c) 部分求和（psums）。

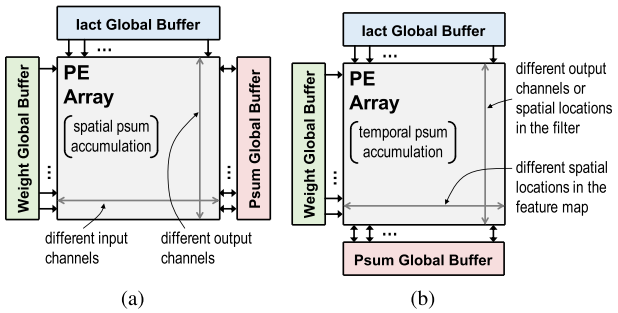


图3. 两种常见的 DNN 加速器设计：(a) 空间累积阵列[29]-[32]：iacts在垂直方向上重复使用，而psums在水平方向上累积。(b) 时间累积阵列[15]-[17]：iacts在垂直方向上重复使用，而weights在水平方向上重复使用。

这将导致深度神经网络（DNN）的一个关键特性——*数据复用*（即使用相同数据块的MAC数量，即MACs/data）发生变化。大多数 DNN 加速器都依赖数据复用来提升效率。在 DNN 层中，三种数据类型（输入激活iacts、权重和部分和psums）的复用程度取决于该层的形状和尺寸。例如，输入激活的复用量与输出通道数及滤波器尺寸成正比。因此，数据维度降低意味着从任何特定维度利用数据复用都变得更加困难。

图2显示，在较新的深度神经网络中，所有数据类型的数据重用率变化增加，而iacts和psums的数据重用率则降低。这种变化和整体数据重用率的下降使得 DNN 加速器的设计在两个方面更具挑战性。

1) 阵列利用率： 许多现有的 DNN 加速器[15]-[17]、[29]-[32]依赖于一组预先选定的数据维度，以利用处理单元（PEs）阵列的高并行性来实现高性能和数据处理。重用机制可显著提升能效。以图3展示的两种典型设计为例：空间累加阵列架构（图3a）常用于权重固定的数据流场景，通过同时利用输入输出通道，将运算操作在并行计算单元阵列上进行空间映射以实现并行处理。在此架构中，每个输入向量（iact）可沿垂直方向在不同计算单元间复用，同时整合来自各输出通道的权重值；而同一行内所有计算单元的累加结果（psums）则会在写回全局缓冲区前完成空间累加。与之类似，时间累加阵列架构（图3b）适用于输出固定的数据流场景，通过引入新的数据维度实现高计算并行度。该架构中，输入向量仍沿垂直方向在列内不同计算单元间复用，而权重值则沿水平方向在行内计算单元间重复使用。

当预选数据维度因 DNN 形状和尺寸变化而缩减时（例如某层输出通道数（M）少于PE阵列高度），效率会下降。具体而言，这些空间

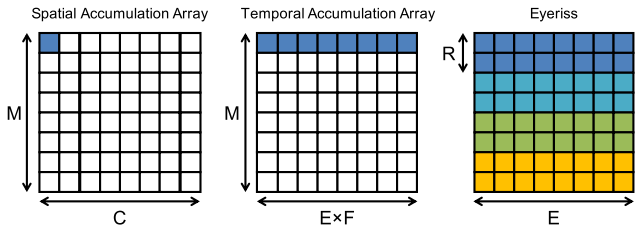


Fig. 4. Array utilization of different architectures for depth-wise (DW) layers in MobileNet. The colored blocks are the utilized part of the PE array. For Eyeriss [33], the different colors denote the parts that run different channel groups (G). Please refer to Table I for the meaning of the variables.

mapping constraints result in both reduced array utilization (i.e., fewer PEs are used) as well as lower energy efficiency. Furthermore, these inefficiencies are magnified as the size of the PE array is scaled up, because the diminished dimension is even more likely to be unable to fill the array. For example, as shown in Fig. 4, the aforementioned spatial and temporal accumulation arrays will find it difficult to fully utilize the array due to the lack of input and output channels in the depth-wise (DW) layers in MobileNet. In contrast, Eyeriss [33] can still achieve high array utilization under such circumstances by mapping the independent channel groups onto different part of the PE array due to the flexibility of its Row-Stationary (RS) dataflow.

2) PE Utilization: A lower data reuse also implies that a higher data bandwidth is required to keep the PEs busy. If the on-chip network (NoC) for data delivery to the PEs is designed for high spatial reuse scenarios, e.g., a broadcast network, the insufficient bandwidth can lead to reduced utilization of the PEs (i.e., increased stall cycles), which further reduces accelerator performance. For instance, even though Eyeriss can better utilize the array as shown in Fig. 4, its broadcast NoC (which supports multicast) is not going to provide adequate bandwidth to support high throughput processing at high parallelism, thus the performance will still suffer. However, if the NoC is optimized for high bandwidth scenarios, e.g., many unicast networks, it may not be able to take advantage of data reuse when available.

An additional challenge lies in the fact that all DNNs that the hardware needs to run will *not be known at design time* [34]; as a result, the hardware has to be flexible enough to efficiently support a wide range of DNNs. To build a truly flexible DNN accelerator, the new challenge is to design an architecture that can accommodate a wide range of shapes and sizes of DNN layers. In other words, the data has to be flexibly mapped spatially according to the specific shape and size of the layer, instead of with a set of pre-selected dimensions, in order to maximize the utilization of the PE array. Also, the data delivery NoC has to be able to provide high bandwidth when data reuse is low while still being able to exploit data reuse with high parallelism when the opportunity presents itself.

B. Challenges for Sparse DNNs

Sparse activations naturally occur in DNNs for several reasons. One is that many DNNs use the rectified linear unit (ReLU) as the activation function, which sets negative values to zero; this sparsity tends to increase in deeper layers and can go above 90%. Another increasingly important reason is that many popular DNNs are in the form of autoencoders [35]–[37] or generative adversarial networks (GAN) [38], which contain decoder layers that use zero insertion to up-sample the input feature maps, resulting in over 75% zeros.

There has also been a significant amount of work to make the weights in a DNN sparse. Various metrics are used to decide which weights to prune (i.e., set to zero), including saliency [12], magnitude [13], and energy consumption [14]. These pruned networks have weight sparsity of up to 90%.

Sparsity in weights and activations can be translated into improved energy efficiency and processing speed in two ways: (1) The MAC computation can be either gated or skipped; the former reduces energy while the latter reduces both energy and cycles. (2) The weights and activations can be compressed to reduce the amount of storage and data movement; the former reduces energy while the latter reduces both energy and cycles. However, it is quite challenging to design DNN accelerators that can actually harness these benefits from sparsity due to the following reasons:

1) Irregular Accesses Patterns: Computation gating can effectively translate sparsity in both weights and activations into energy savings, and its implementation can be realized at a low cost by recognizing if either the weight or activation is zero and gating the datapath switching and memory accesses accordingly. For example, Eyeriss has demonstrated gating for sparse activations.

To improve throughput in addition to saving energy consumption, it is desirable to skip the cycles of processing MACs that have zero weights or iacts. However, this requires more complex read logic as it must find the next non-zero value to read without wasting cycles reading zeros. A natural way to address this issue is to keep the weights and iacts in a compressed format that can indicate the location of the next non-zero relative to the current one. However, compressed formats tend to be of variable length and thus must be accessed sequentially. This makes it difficult to divide up the compressed data for parallel processing across PEs without compromising compression efficiency. Furthermore, this presents a challenge if sparsity in *both* weights and activation must be simultaneously recognized, as it is difficult to 'jump ahead' (e.g., skip non-zero weights when the corresponding iact is zero) for many of the most efficient compression formats; the irregularity introduced by jumping ahead also prevents the use of pre-fetching as a means of improving throughput. Thus, the control logic to process the compressed data can be quite complex and adds overhead to the PEs.

Accordingly, there has been limited hardware in this space. Cnvlutin [39] only supports skipping cycles for activations and does not compress the weights, while Cambricon-X [40] does not keep activations in compressed form. Due to the complexity of the logic to skip cycles for both weights and activations, existing hardware for sparse processing is typically limited to a specific layer type. For instance, EIE targets fully-connected (FC) layers [41], while SCNN targets convolutional (CONV) layers [42].

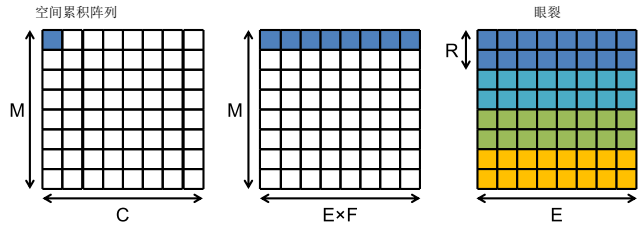


图4. MobileNet中不同架构深度方向（DW）层的阵列利用率。彩色方块表示并行执行阵列（PE阵列）的使用部分。根据Eyeriss[33]的研究，不同颜色对应运行不同通道组（G）的区域。各变量含义请参阅表I。

映射约束不仅导致阵列利用率降低（即使用的处理单元PE更少），还会造成能效下降。更严重的是，随着处理单元阵列规模的扩大，这种低效问题会进一步加剧——因为阵列维度缩减后，更难实现空间填充。例如图4所示，由于MobileNet模型的深度（DW）层缺乏输入输出通道，前述空间与时间累积阵列将难以充分发挥阵列效能。相比之下，Eyeriss[33]通过将独立通道组映射到处理单元阵列的不同区域，凭借其行静态（RS）数据流的灵活性，即便在类似情况下仍能保持高阵列利用率。

*2) PE资源利用率：*数据复用率越低，就需要更大的带宽来维持处理单元（PE）的运行。如果用于向PE传输数据的片上网络（NoC）是为高空间复用场景设计的（例如广播网络），带宽不足会导致PE利用率下降（即增加停滞周期），从而进一步降低加速器性能。例如，尽管Eyeriss如图4所示能更高效地利用阵列资源，但其支持组播的广播型NoC无法提供足够的带宽来支持高并行度下的高吞吐量处理，因此性能仍会受到影响。然而，若NoC针对高带宽场景（例如多组单播网络）进行优化，则可能无法充分利用可用的数据复用资源。

另一个挑战在于，硬件运行所需的所有深度神经网络（DNN）在*设计阶段都无法预知*[34]。因此，硬件必须具备足够的灵活性，以高效支持多种DNN架构。要构建真正灵活的 DNN 加速器，新挑战在于设计一种能够适应不同形状和尺寸 DNN 层的架构。换言之，数据必须根据层的具体形状和尺寸进行灵活的空间映射，而非采用预设的固定维度，这样才能最大限度地利用并行执行阵列（PE array）。此外，数据传输网络（NoC）必须在数据复用率较低时提供高带宽，同时在出现高并行度的数据复用机会时，仍能充分利用这些资源。

B. 稀疏深度神经网络面临的挑战

稀疏激活在深度神经网络（DNN）中自然产生，原因有以下几点：其一，许多DNN采用修正线性单元（ReLU）作为激活函数，该函数会将负值设为0。数值趋向于零；这种稀疏性在更深的层中趋于增强，可超过90%。另一个日益重要的原因是，许多流行的深度神经网络（DNN）采用自编码器[35]–[37]或生成对抗网络（GAN）[38]的形式，其解码器层通过零插入对输入特征图进行上采样，导致零值占比超过75%。

在使 DNN 中的权重稀疏化方面也进行了大量工作。使用各种度量标准来决定哪些权重需要修剪（即设置为零），包括显著性[12]、幅度[13]和能耗[14]。这些修剪后的网络具有高达90%的权重稀疏度。

权重与激活值的稀疏性可通过两种方式转化为能效提升与处理速度加快：(1) MAC计算可以是门控的或跳过的；前者减少能耗，后者则同时减少能耗和周期。(2)权重和激活值可以被压缩以减少存储和数据移动量；前者减少能耗，后者则同时减少能耗和周期。然而，设计能够真正利用稀疏性优势的DNN 加速器相当具有挑战性，原因如下：

*1) 不规则访问模式优化：*计算门控技术能有效将权重和激活值的稀疏性转化为节能优势。该技术通过检测权重或激活值是否为零，并据此调整数据路径切换和内存访问，可实现低成本部署。例如，Eyeriss公司已成功展示了针对稀疏激活值的门控技术应用。

为了在提升吞吐量的同时降低能耗，我们希望跳过那些权重或激活值为零的MAC处理周期。但这样做需要更复杂的读取逻辑，因为必须在读取零值时避免浪费周期，从而找到下一个非零值进行读取。解决这个问题的自然方法是将权重和激活值以压缩格式存储，这种格式能指示下一个非零值相对于当前值的位置。然而，压缩格式往往具有可变长度，因此必须顺序访问。这使得在不影响压缩效率的前提下，难以将压缩数据分割成并行处理单元（PE）进行并行处理。此外，当*权重和激活值的稀疏性*需要同时识别时，这种设计会带来挑战——对于许多高效的压缩格式而言，很难实现“跳跃式处理”（例如当对应激活值为零时跳过非零权重）；这种跳跃带来的不规则性也阻碍了预取技术在提升吞吐量中的应用。因此，处理压缩数据的控制逻辑可能相当复杂，给并行执行单元（PE）增加了额外开销。

因此，这一领域的硬件选择较为有限。Cnvlutin [39] 仅支持激活值的周期跳过，而不压缩权重，而 Cambricon-X [40] 则不以压缩形式存储激活值。由于跳过权重和激活值的逻辑复杂性，现有的稀疏处理硬件通常仅限于特定类型的层。例如， EIE 靶向全连接（FC）层 [41]，而 SCNN 靶向卷积（CONV）层 [42]。

2) Workload Imbalance and PE Utilization: With computation skipping for sparse data, the amount of work to be performed at each PE now depends on sparsity. Since the number of non-zero values varies across different layers, data types, or even regions within the same filter or feature map, it creates an imbalanced workload across different PEs and the throughput of the entire DNN accelerator will be bounded by the PE that has the most non-zero MACs. This leads to a decrease in PE utilization.

C. Contributions of This Work

To address these challenges, we present Eyeriss v2, a flexible architecture for DNN processing that can adapt to a wide range of filter shapes and sizes used in compact DNNs such as MobileNet. This is achieved through the design of a highly flexible on-chip network (NoC), which is currently the bottleneck for dealing with a more diverse set of DNNs. In addition, Eyeriss v2 also supports sparse DNNs by exploiting the sparsity in the weights and activations across a variety of DNN layers and translates them into improvements in both energy efficiency and processing speed. Finally, similar to the original Eyeriss, Eyeriss v2 does not make any assumption about whether the total storage capacity required by a DNN layer can fit on-chip or not; instead, it optimizes the way to tile data of different types to achieve high on-chip reuse and energy efficiency. In summary, the contributions of this paper include:

- A novel NoC, called hierarchical mesh, that is designed to adapt to a wide range of bandwidth requirements. When data reuse is low, it can provide high bandwidth (via unicast) from the memory hierarchy to keep the PEs busy; when data reuse is high, it can still exploit spatial data reuse (via multicast or broadcast) to achieve high energy efficiency. For a compact DNN such as MobileNet, the hierarchical mesh increases the throughput by 5.6× and energy efficiency by 1.8×. (Section III)
- A PE that exploits the sparsity in weights and activations to achieve improved throughput and energy efficiency across a variety of DNN layers. Data is kept in compressed sparse column (CSC) format for both on-chip processing and off-chip access to reduce storage and data movement costs. Mapping of the weights to a PE is performed by taking the sparsity into account to increase reuse within PE, and can therefore reduce the impact of workload imbalance. Overall, exploiting sparsity results in an additional 1.2× and 1.3× improvement in throughput and energy efficiency, respectively, for MobileNet. (Section IV)
- A flexible accelerator, Eyeriss v2, that combines the above contributions to efficiently support both compact and sparse DNNs. Eyeriss v2 running sparse MobileNet is 12.6× faster and 2.5× more energy efficient than the original Eyeriss (scaled to the same number of PEs and storage capacity as Eyeriss v2), i.e., Eyeriss v1, running MobileNet (49.2M MACs). Eyeriss v2 is also 42.5× faster and 11.3× more energy efficient with sparse AlexNet compared to Eyeriss v1 running AlexNet (724.4M MACs). Finally, Eyeriss v2 running sparse MobileNet is 225.1× faster and 42.0 × more energy efficient than Eyeriss v1 running AlexNet. It is evident that supporting sparse and compact DNNs have a significant impact on speed and energy consumption. (Section V)

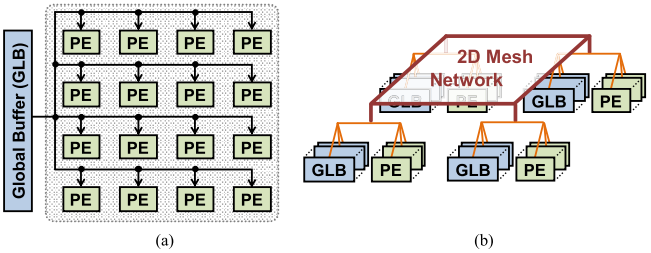


Fig. 5. Comparison of the architecture of original Eyeriss and Eyeriss v2. (a) Original Eyeriss. (b) Eyeriss v2.

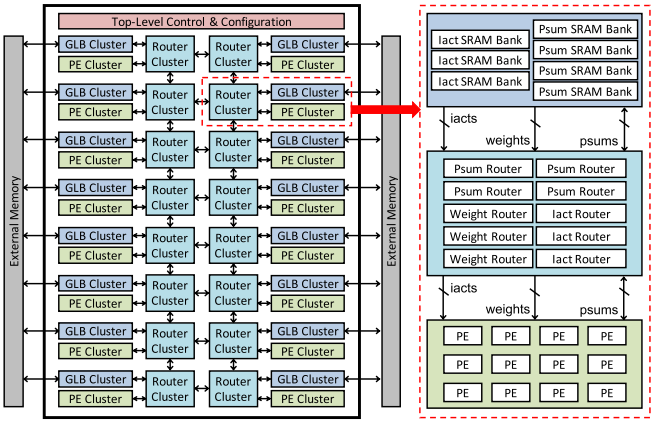


Fig. 6. Eyeriss v2 top-level architecture.

II. Architecture Overview

Fig. 5 shows a comparison between the original Eyeriss [33] and the Eyeriss v2 architecture. Similar to the original Eyeriss architecture, Eyeriss v2 is composed of an array of processing elements (PE), where each PE contains logic to compute multiply-and-accumulate (MAC) and local scratch pad (SPad) memory to exploit data reuse, and global buffers (GLB), which serve an additional level of memory hierarchy between the PEs and the off-chip DRAM. Therefore, both the original Eyeriss and Eyeriss v2 have a two-level memory hierarchy. The main difference is that Eyeriss v2 uses a hierarchical structure, where the PEs and GLBs are grouped into clusters in order to support a flexible on-chip network (NoC) that connects the GLBs to the PEs at low cost; in contrast, the original Eyeriss used a flat multicast NoC between the GLB and PEs. As with the original Eyeriss, Eyeriss v2 uses separate NoCs to transfer each of the three data types, i.e., input activation (iact), weight, and partial sums (psums), between the GLBs and PEs, with each NoC tailored for the corresponding dataflow of that data type. Details of the NoC are described in Section III.

Fig. 6 shows the top-level architecture of Eyeriss v2 and Table II summarizes the components in the architecture. It consists of 16 PE clusters and 16 GLB clusters arranged in an 8×2 array. Each PE cluster contains 12 PEs arranged in a 3 × 4 array. Each GLB cluster has a capacity of 12 KB and consists

*2) 工作负载失衡与处理单元利用率：*当计算跳过稀疏数据时，各处理单元需要执行的工作量将取决于数据的稀疏程度。由于非零值数量会因不同层级、数据类型，甚至同一滤波器或特征图中的不同区域而异，这会导致各处理单元间的工作负载失衡。整个 DNN 加速器的吞吐量将受限于具有最多非零MAC值的处理单元，从而导致处理单元利用率下降。

C. 本工作的贡献

为了应对这些挑战，我们提出了Eyeriss v2，这是一种灵活的 DNN 处理架构，能够适应紧凑型DNN（如MobileNet）中使用的各种滤波器形状和尺寸。这是通过设计一个高度灵活的片上网络（NoC）实现的，而片上网络目前是处理更多样化DNN的瓶颈。此外，Eyeriss v2还通过利用不同 DNN 层中权重和激活值的稀疏性来支持稀疏DNN，并将其转化为能效和处理速度的提升。最后，与原始Eyeriss类似，Eyeriss v2不假设 DNN 层所需的总存储容量是否适合片上存储；相反，它优化了不同类型数据的分片方式，以实现高片上复用和能效。总之，本文的贡献包括：

- 一种新型的网络架构，称为层次化网格，旨在适应广泛的带宽需求。当数据复用率较低时，它可以通过单播从内存层次结构提供高带宽，以保持处理单元的忙碌状态；当数据复用率较高时，它仍能通过多播或广播利用空间数据复用，从而实现高能效。对于像MobileNet这样的紧凑型 DNN，层次化网格将吞吐量提高了5.6倍，能效提高了1.8倍。（第三部分）
- 一种利用权重和激活值稀疏性来提高各种 DNN 层吞吐量和能效的PE。数据以压缩稀疏列（CSC）格式存储，以便在芯片上处理和芯片外访问，从而减少存储和数据移动成本。权重映射到PE时考虑了稀疏性，以增加PE内的重用，因此可以减少工作负载不平衡的影响。总体而言，利用稀疏性可使MobileNet的吞吐量和能效分别提高1.2倍和1.3倍。（第四节）
- Eyeriss v2是一款灵活的加速器，整合了上述技术优势，能高效支持紧凑型和稀疏型深度神经网络。在运行稀疏MobileNet时，Eyeriss v2的运算速度比原版Eyeriss（在并行单元数量和存储容量上与Eyeriss v1保持一致）快12.6倍，能效提升2.5倍（Eyeriss v1运行MobileNet时需4920万MACs）。在处理稀疏AlexNet时，Eyeriss v2的运算速度比Eyeriss v1快42.5倍，能效提升11.3倍。

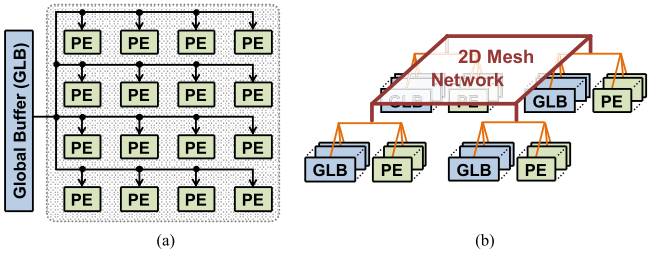


图5. 原始Eyeriss与Eyeriss v2架构的对比
(a) 原始Eyeriss。(b) Eyeriss v2.

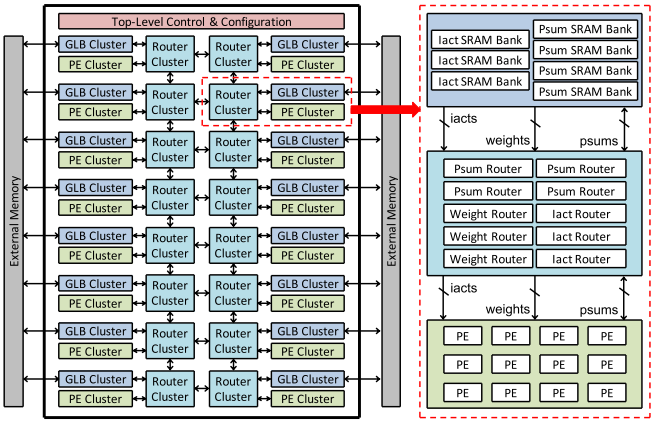


图6. Eyeriss v2顶层架构。

（724.4M MACs）。最终，运行稀疏MobileNet的Eyeriss v2比运行AlexNet的Eyeriss v1快225.1倍，能效提升42.0倍。显然，支持稀疏紧凑型深度神经网络（DNNs）对速度和能耗具有显著影响。（第五节）

二.体系结构概述

图5展示了原始Eyeriss架构[33]与Eyeriss v2架构的对比。与原始Eyeriss架构类似，Eyeriss v2同样由处理单元（PE）阵列构成，每个PE包含用于执行乘积累加（MAC）运算的逻辑单元、支持数据复用的本地临时存储器（SPad），以及作为PE与片外DRAM之间二级存储层次的全局缓冲器（GLB）。因此，原始Eyeriss和Eyeriss v2都采用两级存储层次结构。主要区别在于：Eyeriss v2采用分层结构，将PE和GLB按集群方式组织，以支持低成本连接GLB与PE的灵活片上网络（NoC）；而原始Eyeriss则采用GLB与PE之间的扁平多播NoC。与原始Eyeriss类似，Eyeriss v2通过独立NoC在GLB与PE之间传输三种数据类型——输入激活（iact）、权重和部分和（psums），每个NoC都针对对应数据流进行定制。NoC的具体细节将在第三节详细说明。

图6展示了Eyeriss v2的顶层架构，表II则汇总了该架构的组件。该架构由16个PE集群和16个GLB集群组成，以8×2阵列排列。每个PE集群包含12个PE，以3×4阵列排列。每个GLB集群的容量为12 KB，由

TABLE II
EYERISS V2 ARCHITECTURE HIERARCHY

Hierarchy	# of Components
Cluster Array	8×2 PE clusters 8×2 GLB clusters 8×2 router clusters
PE cluster	3×4 PEs
GLB cluster	3×1.5 kB SRAM banks for iacts 4×1.875 kB SRAM banks for psums
router cluster	3 iact routers (4 src/dst ports, 24b/port) 3 weight routers (2 src/dst ports, 24b/port) 4 psum routers (3 src/dst ports, 40b/port)

of SRAMs that are banked for different data types: iacts have three banks, each of which is 1.5 kB, and psums have four banks, each of which is 1.875 kB.

A hierarchical NoC is used to connect the PEs and GLBs: the PE and GLB clusters are connected through 2D mesh on-chip networks that consist of router clusters. Within each router cluster, there are 3, 3, and 4 routers for iact, weight and psum, respectively. Between the PE cluster and the router cluster, an all-to-all NoC is used to connect all the PEs to the routers for each data type. Between the GLB cluster and the router cluster, each router is paired with a specific port of the GLB cluster, which can read from and write to one SRAM bank or off-chip I/O. Therefore, data from either off-chip or a GLB cluster first goes into the router cluster, and then can be unicast to the local PE cluster, multicast to PE clusters on the same row or column in the mesh network, or broadcast to all PE clusters. The decision is based on the shape and size of the DNN layer and the processing dataflow. The design motivation and implementation details of this hierarchical mesh network and the dataflow are described in Section III.

The data movement through the two-level memory hierarchy on Eyeriss v2 is as follows:

- *iacts* are read from off-chip into the GLB cluster, where they can be stored into the GLB memory or get passed directly to the router cluster depending on the configuration.
- *psums* are always stored in the GLB memory once they get out of the PE cluster. The final output activations skip the GLB cluster and go directly off-chip.
- *weights* are not stored in GLB and get passed to the router clusters and eventually stored in the SPads in each PE directly.

Eyeriss v2 adopts the Row-Stationary (RS) dataflow [43] used in the original Eyeriss, and further explores tiling the MAC operations spatially across PEs through any layer dimension, including the channel group dimension (G in Table I). This is especially important for layers such as the depth-wise (DW) CONV layers in MobileNet, which lacks the input and output channels that are commonly used for spatial tiling and therefore greatly improves the array utilization.

Each PE contains multiply-and-accumulate (MAC) datapaths designed to process 8-bit fixed-point iacts and weights, which is the commonly accepted bitwidth for inference. Since many layers receive iacts after ReLU, the iacts can be set to either signed or unsigned, which further extends the scale of iact representation. Psums are accumulated at 20-bit precision, which has shown no accuracy impact in our experiments. When the accumulation is done, the 20-bit psums are converted back to 8-bit output activations and sent off-chip. The PE contains separate SPads for iact, psum and weights. Details of the PE architecture are described in Section IV.

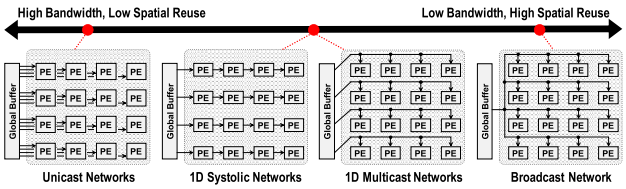


Fig. 7. Common NoC Designs.

Finally, Eyeriss v2 has a two-level control logic similar to the original Eyeriss. The system-level control coordinates the off-chip data accesses and data traffic between the GLB and PEs, and the lower-level control is within each PE and controls the progress of processing of each PE independently. The chip can be reconfigured to run the dataflow that maximizes the energy efficiency and throughput for the processing of each DNN layer. This includes setting up the specific data traffic pattern of the NoCs, data accesses to the GLB and SPads, and workload distribution for each PE. For each layer, a 2134-bit command that describes the optimized configuration is sent to the chip and accessed statically throughout the processing of this layer. Only one layer is processed at a time. When the processing for a layer is done, the chip is reconfigured for the processing of the next layer.

III. FLEXIBLE HIERARCHICAL MESH ON-CHIP NETWORK

One of the key features required to support compact DNNs is a flexible and efficient on-chip network (NoC). This section will provide details on the implementation of the NoC in Eyeriss v2 as well as describe how the NoC is configured for various use cases.

A. Motivation

The NoC is an indispensable part of modern DNN accelerators, and its design has to take the following factors into consideration: (1) support processing with high parallelism by efficiently delivering data between storage and datapaths, (2) exploit data reuse to reduce the bandwidth requirement and improve energy efficiency, and (3) can be scaled at a reasonable implementation cost.

Fig. 7 shows several NoC designs commonly used in DNN accelerators. Due to the property of DNN that data reuse for all data types cannot be maximally exploited simultaneously, a mixture of these NoCs is usually adopted for different data types. For example, a DNN accelerator can use a 1D horizontal multicast network to reuse the same weight across PEs in the same row and a 1D vertical multicast network to reuse the same iact across PEs in the same column. This setup will then require an unicast network that gathers the unique output activations from each PE. This combination, however,

表 II

EYERISS V2 体系结构层次

层级	并 组分
簇阵列	8个X2 PE簇 8 ×2个GLB簇 8 ×2个路由器集群
PE簇	3×4 PEs
GLB簇	3 × 1.5 KB SRAM 用于 i acts 的银行 4x 1.875 KB SRAM p 个银行的金额
路由器簇	3个iACT路由器（4个SRC/DST端口，24位/端口）、3个权重路由器（2个SRC/DST端口，24位/端口）、4个PSUM路由器（3个SRC/DST端口，40位/端口）

针对不同数据类型的SRAM存储器：iacts配备三个存储库，每个容量为1.5 kB；psums则设有四个存储库，每个容量为1.875 kB。

采用分层式网络芯片（NoC）连接处理单元（PE）与通用逻辑块（GLB）：PE 和 GLB 集群通过由路由器集群组成的 2D 网状片上网络连接。每个路由器集群中分别有 3 个、3 个和 4 个路由器，分别用于 iact、weight 和 psum。在 PE 集群和路由器集群之间，使用全连接的 NoC 将所有 PE 连接到每个数据类型的路由器。在 GLB 集群和路由器集群之间，每个路由器都与 GLB 集群的特定端口配对，该端口可以读取和写入一个 SRAM 存储器或片外 I/O。因此，来自片外或 GLB 集群的数据首先进入路由器集群，然后可以单播到本地 PE 集群，组播到网格网络中同一行或列的 PE 集群，或者广播到所有 PE 集群。这一决策基于 DNN 层的形状和大小以及处理数据流。这种分层网格网络和数据流的设计动机和实现细节将在第三节中描述。

Eyeriss v2系统中两级存储器层次结构的数据传输流程如下：

- *iacts*从芯片外读入GLB集群，在那里它们可以存储到GLB内存中，或者根据配置直接传递到路由器集群。
- 一旦*psums*从PE集群输出，就会被存储在GLB内存中。最终的输出激活值会跳过GLB集群，直接输出到芯片外部。
- *权重*不会存储在全局内存块（GLB）中，而是直接传递给路由器集群，并最终存储在每个处理单元（PE）的存储块（SPads）中。

Eyeriss v2沿用了原版Eyeriss采用的行-静态（RS）数据流[43]，并进一步探索了通过任何层维度（包括表I中的通道组维度*G*）将多址接入控制（MAC）操作在处理单元（PE）间进行空间分片。这一改进对MobileNet中的深度卷积（DW）层尤为重要，这类层通常缺乏用于空间分片的输入和输出通道，因此能显著提升阵列利用率。

每个PE单元均配备乘积累加（MAC）数据通路，专门处理8位定点输入/输出（iacts）和权重参数——该位宽是推理领域通用标准。由于多数神经网络层在ReLU激活后接收iacts信号，这些信号可配置为带符号或无符号形式，从而显著扩展了参数处理的规模。

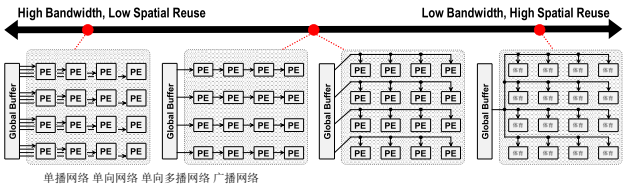


图7. 常见网络芯片设计。

iact表示。psum以20位精度累加，实验表明其精度无显著影响。完成累加后，20位psum将转换为8位输出激活值并输出芯片。处理单元（PE）包含独立的iact、psum和权重存储单元（SPad）。PE架构细节详见第IV节。

最后，Eyeriss v2 采用与原始 Eyeriss 相似的两级控制逻辑。系统级控制协调片外数据访问和 GLB 与 PE 之间的数据流量，而较低级别的控制则位于每个 PE 内部，独立控制每个 PE 的处理进度。该芯片可以重新配置，以运行最大化每个 DNN 层处理的能效和吞吐量的数据流。这包括设置 NoC 的特定数据流量模式、对 GLB 和 SPads 的数据访问以及每个 PE 的工作负载分配。对于每一层，会发送一条描述优化配置的 2134 位命令到芯片，并在整个该层处理过程中静态访问。每次只处理一层。当一层处理完成后，芯片将重新配置以处理下一层。

三、柱状体上的灵活分层网格网络

支持紧凑型深度神经网络（DNN）的关键特性之一是灵活高效的片上网络（NoC）。本节将详细阐述Eyeriss v2中NoC的实现方案，并说明如何针对不同应用场景配置NoC。

A. 动机

NoC是现代 DNN 加速器不可或缺的一部分，其设计必须考虑以下因素：(1) 通过高效地在存储器和数据路径之间传输数据，支持具有高并行性的处理；(2) 利用数据重用来减少带宽需求并提高能效；(3) 能以合理的实现成本进行扩展。

图7展示了 DNN 加速器中常用的几种NoC设计。由于 DNN 的特性，无法同时充分利用所有数据类型的数据重用，因此通常会采用混合的NoC设计来处理不同数据类型。例如， DNN 加速器可以使用一维水平组播网络，在同一行的PE之间重用相同的权重，同时使用一维垂直组播网络，在同一列的PE之间重用相同的激活。这种设置需要一个单播网络来收集每个PE的唯一输出激活。然而，这种组合

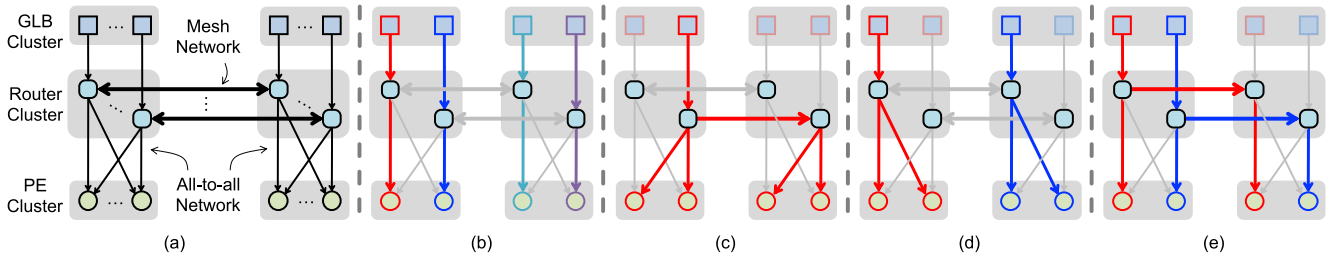


Fig. 8. (a) High-level structure of the hierarchical mesh network (HM-NoC), and its different operating modes: (b) High bandwidth mode, (c) High reuse mode, (d) grouped-multicast mode, and (e) interleaved-multicast mode. In each mode, the colored arrows show the routing path; different colors denote the path for unique data.

implies that each weight needs to have the amount of reuse with different iacts at least equal to the width of the PE array, and the number of iact reuse with different weights at least equal to the height of the PE array. If these conditions are not fulfilled, the PE array will not be fully utilized, which will impact both throughput and energy efficiency.

While it was easy to satisfy such conditions with large DNNs, the rise of compact DNN models has made this design approach less effective. The key reason is that it is much more difficult to assume the amount of data reuse or required data bandwidth for each data type, as it will vary across layers and DNN models. For example, the lack of input or output channels in the depth-wise layers of MobileNet or in the bottleneck layers of ResNet and GoogLeNet has made it very difficult to efficiently utilize the aforementioned example well due to its rigid NoC design. In layers such as fully-connected layers, commonly used in RNNs and CNNs, it will also require a large batch size to improve the amount of reuse for weights, which can be challenging in real-time applications that are sensitive to the processing latency.

The varying amount of data reuse for each DNN data type across different layers or models pose a great challenge to the NoC design. The broadcast network can exploit the most data reuse, but its low source bandwidth can limit the throughput when data reuse is low. The unicast network can provide the most source bandwidth but misses out on the data reuse opportunity when available. Taking the best from both worlds, an all-to-all network that connects any data sources to any destinations can adapt to the varying amount of data reuse and bandwidth requirements. However, the cost of its design increases quadratically with the number of nodes, e.g., PEs, and therefore is difficult to scale up to the amount of parallelism required for DNN accelerators.

B. High-Level Concept and Use Cases

To deal with this problem, we propose the hierarchical mesh network (HM-NoC) in Eyeriss v2 as shown in Fig. 8a. HM-NoC takes advantage of the all-to-all network, but solves the scaling problem by creating a two-level hierarchy. The all-to-all network is limited within the scope of a cluster at the lower level. In Eyeriss v2, there are only 12 PEs in each cluster, which effectively reduce the cost of the all-to-all network. At the top level, the clusters are further connected with a mesh network. While this example shows a 2×1 mesh, Eyeriss v2 uses a 8×2 mesh. Scaling up the architecture at the cluster level with the mesh network is much easier than with the all-to-all network since the implementation cost increases linearly instead of quadratically.

Fig 8b to 8e shows how the HM-NoC can be configured into four different modes depending on the data reuse opportunity and bandwidth requirements.

- In the *high bandwidth mode* (Fig. 8b), each GLB bank or off-chip data I/O can deliver data independently to the PEs in the cluster, which achieves *unicast*.
- In the *high reuse mode* (Fig. 8c), data from the same source can be routed to all PEs in different clusters, which achieves *broadcast*.
- For situations where the data reuse cannot fully utilize the entire PE array with broadcast, different multicast modes, specifically *grouped-multicast* (Fig. 8d) and *interleaved-multicast* (Fig. 8e), can be adopted according to the desired multicast patterns.

Fig. 9 shows several example use cases of how HM-NoC adapts different modes for different types of layers. For simplicity, we are only showing a simplified case with 2 PE clusters with 2 PEs in each cluster, and it omits the NoC for psums. However, the same principles apply to NoC for all data types and at larger scales.

- Conventional CONV layers (Fig. 9a): In normal CONV layers, there is plenty of data reuse for both iacts and weights. To keep all 4 PEs busy at the lowest bandwidth requirement, we need 2 iacts and 2 weights from the data source (ignoring the reuse from SPad). In this case, either the HM-NoC for iact or weight has to be configured into the grouped-multicast mode, while the other one configured into the interleaved-multicast mode.
- Depth-wise (DP) CONV layers (Fig. 9b): For DP CONV layers, there can be nearly no reuse for iacts due to the lack of output channels. Therefore, we can only exploit the reuse of weights by broadcasting the weights to all PEs while fetching unique iacts for each PE.
- Fully-connected (FC) layers (Fig. 9c): Contrary to the DP CONV layers, FC layers usually see little reuse for weights, especially when the batch size is limited. In this case, the modes of iact and weight NoCs are swapped from the previous one: the weights are now unicast to the PEs while the iacts are broadcast to all PEs.

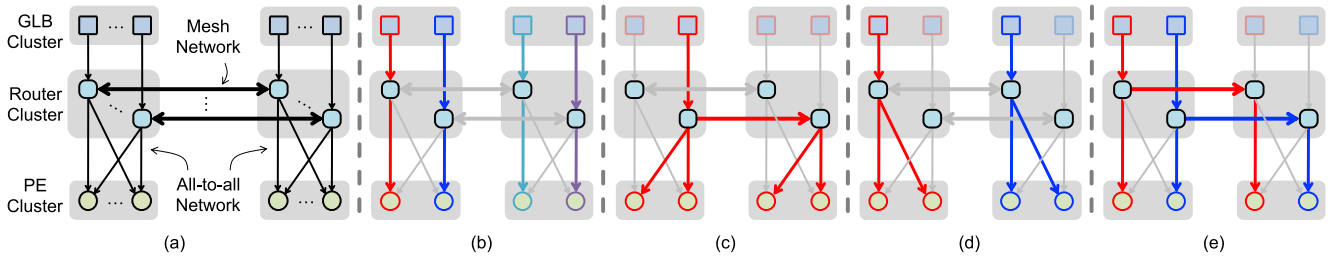


图8. (a) 分层网格网络（HM-NoC）的高层级结构及其不同运行模式：(b) 高带宽模式，(c) 高复用模式，(d) 分组多播模式，(e) 交错多播模式。各模式中，彩色箭头表示路由路径；不同颜色表示唯一数据的路径。

这意味着每个权重在不同iact下的重用次数至少应等于PE阵列的宽度，而不同权重在iact下的重用次数至少应等于PE阵列的高度。若无法满足这些条件，PE阵列将无法得到充分利用，从而影响吞吐量和能效。

虽然使用大型DNN很容易满足这些条件，但紧凑型DNN 模型的兴起使得这种设计方法效果大打折扣。关键原因在于，假设每种数据类型的数据重用量或所需数据带宽要困难得多，因为这会因层和 DNN 模型的不同而异。例如，MobileNet的深度层或ResNet和GoogLeNet的瓶颈层缺乏输入或输出通道，由于其严格的NoC设计，使得上述示例难以高效利用。在RNN和CNN中常用的全连接层等层中，还需要较大的批量大小来提高权重的重用量，这对于对处理延迟敏感的实时应用来说是一个挑战。

不同层次或模型中每种 DNN 数据类型的数据重用量差异给NoC设计带来了巨大挑战。广播网络可以利用最多的数据重用，但其低源带宽在数据重用率低时会限制吞吐量。单播网络可以提供最多的源带宽，但在数据重用机会可用时却错失良机。取两者之长，全连接网络将任何数据源与任何目的地相连，能够适应数据重用量和带宽需求的变化。然而，其设计成本随节点数（如PE）的增加呈二次方增长，因此难以扩展到 DNN 加速器所需的并行度。

B. 高层概念与用例

为解决这一问题，我们在Eyeriss v2中提出了如图8a所示的分层网格网络（HM-NoC）。该方案充分利用全连接网络的优势，通过构建双层分层结构来解决扩展性问题。全连接网络的覆盖范围受限于底层集群，而Eyeriss v2中每个集群仅包含12个处理单元（PE），这有效降低了全连接网络的成本。在顶层，集群之间进一步建立连接。

采用网格网络架构。虽然示例展示的是2×1网格，但Eyeriss v2实际采用的是8×2网格。相较于全连接网络，通过网格网络在集群层面扩展架构要容易得多，因为实现成本呈线性增长而非二次方增长。

图8b至8e展示了如何根据数据复用机会和带宽需求将HM-NoC配置为四种不同模式。

- 在*高带宽模式*（图8b）下，每个GLB银行或片外数据I/O可以独立地将数据传输到集群中的PE，从而实现*单播*。
- 在*高重用模式*（图8c）中，来自同一源的数据可以被路由到不同集群中的所有PE，这实现了*广播*。
- 对于数据重用不能通过广播充分利用整个PE阵列的情况，可以采用不同的多播模式，特别是*分组多播*（图8d）和*交错多播*（图8e），具体取决于所需的多播模式。

图9展示了HM-NoC如何根据不同类型的层适配不同模式的若干示例用例。为简化说明，我们仅展示了一个简化案例：包含2个PE集群，每个集群含2个PE，并省略了psums的NoC。但相同原理适用于所有数据类型及更大规模的NoC。

- 传统卷积神经网络（CONV）层（图9a）：在常规CONV层中，输入向量（iact）和权重参数都存在大量数据复用。为使所有4个处理单元（PE）在最低带宽需求下保持满负荷运行，需从数据源获取2个iact和2个权重（忽略SPad的复用）。此时，iact或权重的HM-NoC必须配置为组播模式，而另一个则需配置为交错组播模式。
- 深度方向（DP）卷积层（图9b）：由于输出通道数量有限，DP卷积层几乎无法对iacts进行重用。因此，我们只能通过将权重广播至所有处理单元（PE），同时为每个PE获取独立iacts来实现权重重用。
- 全连接（FC）层（图9c）：与深度学习中的卷积层（CONV）不同，FC层通常权重重用率较低，尤其在批量处理规模受限时。此时，输入操作（iact）与权重非重用（NoC）的模式会与前一阶段互换：权重采用单播方式传输至处理单元（PE），而输入操作则通过广播方式覆盖所有PE。

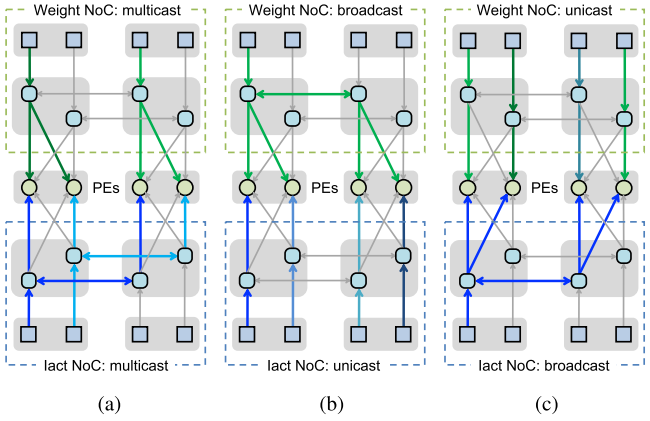


Fig. 9. Examples of weight and iact hierarchical mesh networks configured in different modes for different types of DNN layers: (a) CONV layers; (b) depth-wise (DW) CONV layers; (c) fully-connected (FC) layers. Green arrows and blue arrows show the routing paths in the weight and iact NoC, respectively.

C. Implementation Details

To support the various uses cases described in Section III-B, each of the HM-NoC employs circuit-switched routing, which mainly consists of muxes and is statically configured by the configuration bits as described in Section II. Therefore, the implementation cost of each router is very low. A separate HM-NoC is implemented for each data type (iact, psum, and weights) that is tailored for their given dataflow. The specifications of the routers for each data type are summarized in Table II. For iacts and weights, each port has a bitwidth of 24 bits such that it can send and receive three 8b uncompressed iact values or two 12b compressed iact run-data pairs per cycle. Section IV describes the compression format in more detail. For psum, each port has a bitwidth of 40-bit to send and receive two psums per cycle. We will now describe how the routers, GLB and PEs are connected in the HM-NoC for each data type.

1) HM-NoC for Input Activations: The HM-NoC implementation for *iacts* is shown in Fig. 10. There are three iact routers per router cluster, one for each iact SRAM bank in the GLB cluster. Each router for iact has four source ports (to receive data) and four destination ports (to transmit data). Three of the source and destination ports are used to receive and transmit data from the other clusters in the mesh, which are highlighted with bold arrows in Fig. 10; while a mesh network typically requires four pairs of source and destination ports, we only require three pairs since we only have 8×2 clusters and thus either the east or west port can be omitted. The fourth source port connects to the GLB cluster to receive data either from the memory bank or off-chip, and the fourth destination port connects to all the V PEs in the cluster. Thanks to the all-to-all network in the PE cluster, data from any router can go to any PE in the same cluster.

Fig. 11 shows the implementation details of the mesh network router for iacts. It has four source (src) and four destination (dst) ports. In addition to data (d), each port also has two additional signals, ready (r) and enable (e), for hand-shaking. Each source port generates four enable signals (e.g., e_{00}-e_{03}), each for one destination port, based

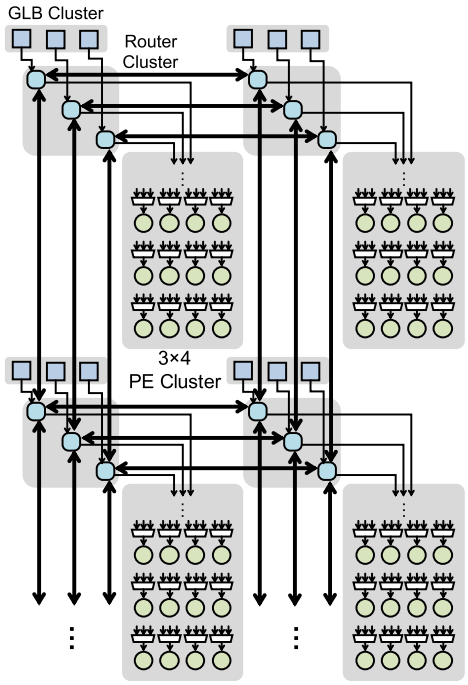


Fig. 10. Hierarchical mesh network for input activations. This only shows the top 2×2 of the entire 8×2 cluster array.

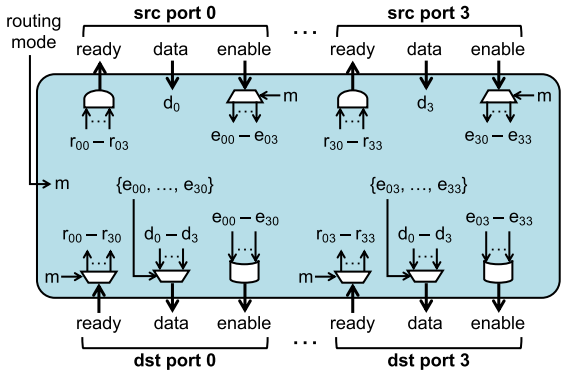


Fig. 11. Implementation details of the mesh network router for input activations. Routers for the other data types use similar logic but with different numbers of ports. d, e, r and m are data, enable, ready and routing mode signals, respectively.

on its own enable signal and the statically configured routing mode (m). The routing mode can be one of the following: unicast, horizontal multicast, vertical multicast, or broadcast. It determines which ports can be enabled for passing data. For example, in the horizontal multicast mode, ports that connect to other routers in the vertical direction of the mesh network will not be enabled. At each destination port, the destination-specific enable signals from all source ports (e.g., e_{00}-e_{30} for destination port 0) go through an OR gate to generate the final enable output. The ready signal from the destination ports to the source ports are generated in a similar fashion with the difference that the source-specific ready signals (e.g., r_{00}-r_{03} for source port 0) go through an AND gate to generated the final ready output at each source port. The output data from all source ports (d_0-d_3) is MUXed at each destination port, and is chosen based on the enable signals from the

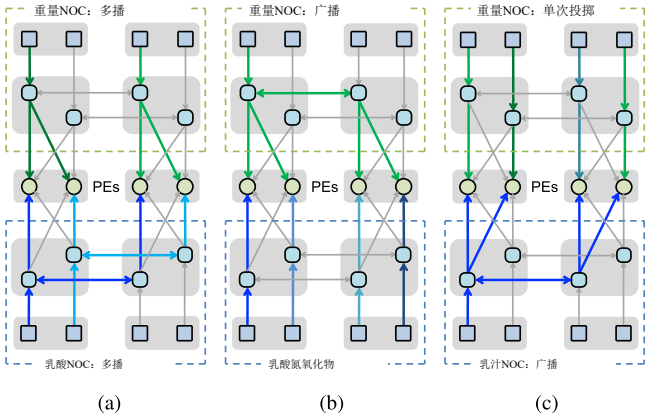


图9. 不同模式下配置的权重和iact层次网格网络示例，适用于不同类型的DNN 层：(a) CONV层；(b) 深度（DW）CONV层；(c) 全连接（FC）层。绿色箭头和蓝色箭头分别显示权重和iact NoC中的路由路径。

C. 实施细节

为支持第三章B节所述的各类应用场景，每个HM-NoC均采用电路交换路由技术，该技术主要由多路复用器构成，并通过第二章所述的配置位进行静态配置。因此，每个路由器的实现成本非常低廉。针对iact、psum和weights三种数据类型，我们分别设计了适配其特定数据流的独立HM-NoC。各数据类型路由器的规格参数详见表II。对于iact和weights，每个端口采用24位位宽，可实现每个周期内发送和接收三个8位未压缩iact值或两个12位压缩iact运行数据对。第四章将详细阐述压缩格式的具体实现。对于psum，每个端口采用40位位宽，可实现每个周期内发送和接收两个psum值。接下来我们将详细说明HM-NoC中各数据类型对应的路由器、全局逻辑块（GLB）与处理单元（PE）的连接方式。

*1) 输入激活的HM-NoC架构：iact*的HM-NoC实现方案如图10所示。每个路由器集群配备三个iact路由器，对应GLB集群中的每个iact SRAM 存储库。每个iact路由器设有四个源端口（用于接收数据）和四个目的端口（用于传输数据）。其中三个源端口和目的端口用于与网格中的其他集群进行数据收发（图10中以粗箭头标示）；虽然传统网格网络通常需要四对源端口和目的端口，但由于我们仅配置了8×2个集群，因此只需三对端口，东端口或西端口可省略。第四个源端口连接至GLB集群，用于接收来自存储库或片外的数据；第四个目的端口则连接集群内所有虚拟处理单元（VP）。得益于处理单元集群的全连接网络架构，任何路由器的数据均可传输至同一集群内的任意处理单元。

图11展示了iacts网状网络路由器的实现细节。该设备配备四个源端口（src）和四个目的端口（dst）。除了数据传输端口（*d*）外，每个端口还设有两个用于握手操作的附加信号——就绪信号（*r*）和使能信号（*e*）。每个源端口会生成四个使能信号（例如*e00-e03*），分别对应四个目的端口。

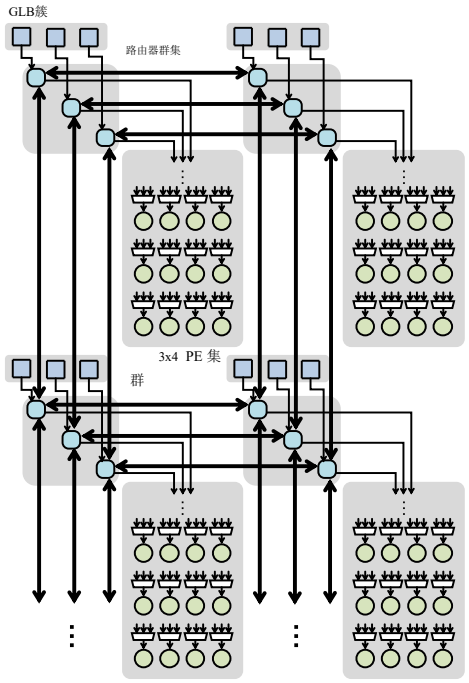


图10. 输入激活的分层网格网络。仅显示整个8×2簇阵列的前2×2部分。

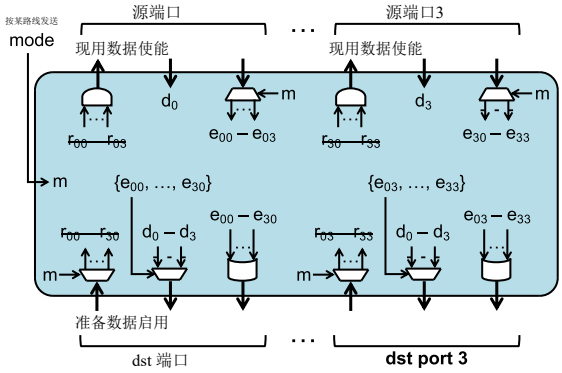


图11.输入激活信号的网状网络路由器实现细节。其他数据类型路由器采用类似逻辑，但端口数量不同。*d*、*e*、*r*和*m*分别是数据、使能、就绪和路由模式信号。

该设备通过自身的启用信号和静态配置的路由模式（*m*）进行工作。路由模式可选单播、水平组播、垂直组播或广播，用于确定哪些端口可启用数据传输。例如在水平组播模式下，连接到网状网络垂直方向其他路由器的端口将不会启用。在每个目的端口，所有源端口（如目的端口0的*e00-e30*）发出的目的端口专用启用信号会通过或门运算生成最终启用输出。目的端口向源端口发送的就绪信号采用类似方式生成，不同之处在于源端口专用就绪信号（如源端口0的*r00-r03*）会通过与门运算在每个源端口生成最终就绪输出。所有源端口（*d0-d3*）的输出数据在每个目的端口进行多路复用，其选择依据是来自

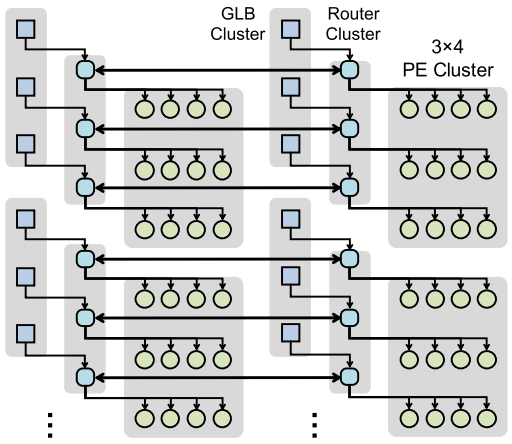


Fig. 12. Hierarchical mesh network for weights. This only shows the top 2×2 of the entire 8×2 cluster array.

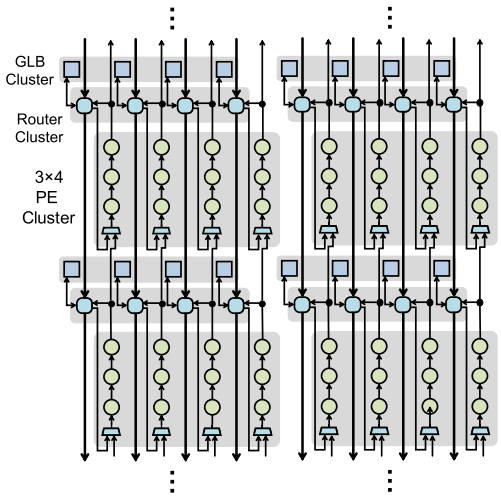


Fig. 13. Hierarchical mesh network for psums. This only shows the a 2×2 portion of the entire 8×2 cluster array.

source ports, i.e., the data from the enabled source port will be passed through.

2) HM-NoC for Weights: The HM-NoC implementation for *weights* is shown Fig. 12. There are three weight routers per router cluster, one for each row of PEs within a cluster. Since Eyeriss v2 uses the RS dataflow, a significant amount of weight reuse can be exploited using the SPad within the PE, and only spatial reuse across horizontal PEs needs to be further exploited. Therefore, the implementation of the NoC for the weights can be simplified at both levels of the HM-NoC to reduce cost but still satisfy the flexibility requirements. Specifically, the vertical connections of the 2D mesh between the clusters can be removed. Furthermore, within each cluster, each router only needs to connect to one row of PEs. Accordingly, each weight router has two source ports and two destination ports. A source port and a destination port are used to receive and transmit weights coming from neighboring cluster; again, we only need one pair of ports here since we only have 8×2 clusters and thus either the east or west port can be omitted. The second source port connects to the GLB cluster to receive data from off-chip, while the second destination port connects to one row of PEs within the cluster. The implementation of the mesh network router for weights is similar to that in Fig. 11.

3) HM-NoC for Partial Sums: The HM-NoC implementation for *psums* is shown in Fig. 13. There are four psum routers per router cluster, one for each psum SRAM bank in the GLB cluster or, equivalently, one for each column of PEs within a cluster. Similar to the weight NoC, the psum NoC is simplified for its given dataflow; specifically, the psums are only allowed to be accumulated across PEs in the vertical direction. This is due to the fact that, in the row-stationary dataflow, weights are reused across PEs horizontally, which makes it impossible to accumulate psums across PEs horizontally. Thus, the horizontal connections of the 2D mesh between the clusters can be removed since psums won't be passed horizontally. Within each cluster, the PEs are vertically connected and each router in the cluster only needs to transmit the psum from the psum bank in the GLB cluster to the bottom of each PE column, and receive the updated psum from the top of the same PE column. Accordingly, each psum router has three source ports and three destination ports. One of the source ports is used to receive data from the neighboring router cluster to the north, while one of the destination ports is used to transmit data to the neighboring router cluster to the south. The second pair of source and destination ports are assigned to the psum bank in the GLB, while the third destination port is assigned to the bottom PE in a column of the PE cluster and the third source port is assigned to the top PE in a column of the PE cluster.

D. Scalability

A key design focus of the HM-NoC is to enable strong scaling for Eyeriss v2. In other words, as the architecture scales with more PEs, the performance, i.e., throughput, should scale accordingly for the same problem size. Performance, however, is a function of many factors, including the dataflow, NoC design, available on-chip and off-chip data delivery bandwidth, etc. To examine the impact of the HM-NoC, we will assume no limitation on the off-chip bandwidth and no workload imbalance (i.e., no sparsity) in the following scalability experiments.

We profile the performance of Eyeriss v2 at three different scales: 256 PEs, 1024 PEs, and 16384 PEs, where each PE is capable of processing at 1 MAC/cycle. The PE cluster for all scales has a fixed array size of 4×4 PEs, and the number of PE clusters scales at 4×4, 8×8, and 32×32. For comparison, we also examine the scalability of the original Eyeriss, i.e., Eyeriss v1, at the same set of scales. For Eyeriss v1, the PEs are arranged in square arrays, i.e., 16×16, 32×32, and 128×128. Both versions of Eyeriss use the row-stationary dataflow. For rapid evaluation of architectures at large scales, we have built an analytical model that can search for the operation mappings with the best performance at different scales considering the data distribution and bandwidth limitations of different NoC designs in the two versions of Eyeriss.

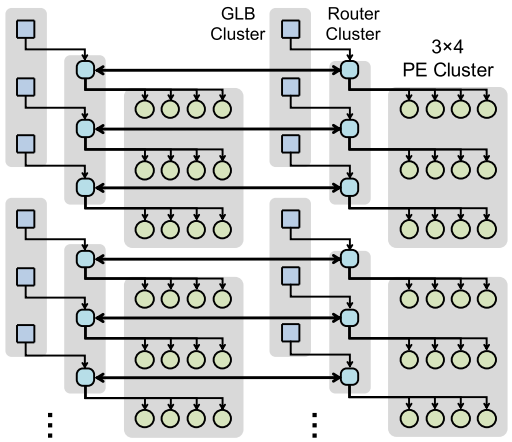


图12. 权重的层次化网格网络。仅显示顶部
2 整个8×2簇阵列的2倍。

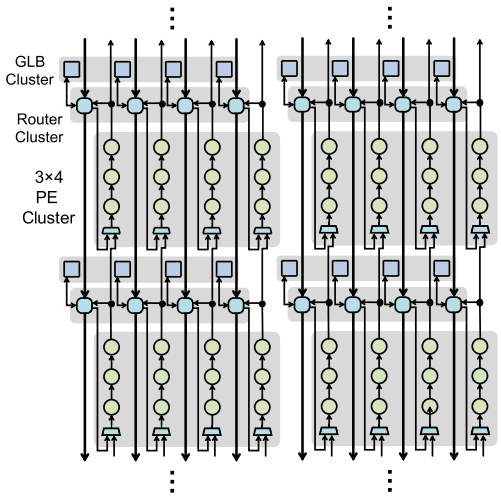


图13. psums的分层网格网络。此图仅显示整个8×2簇阵列中的a 2×2部分。

源端口，即启用的源端口数据将被转发。

*2) 权重网络的HM-NoC架构：*如图12所示，*权重网络的HM-NoC实现方案*采用每簇三个权重路由器的配置，每个簇对应PE节点的一行。由于Eyeriss v2采用RS数据流架构，PE内部的SPad模块可实现大量权重复用，仅需进一步优化水平方向PE间的空间复用。因此，通过简化HM-NoC架构的两个层级，既能降低成本又能满足灵活性需求。具体而言，可移除簇间二维网格的垂直连接。此外，每个簇内的路由器仅需连接单行PE节点，因此每个权重路由器配备两个源端口和两个目的端口。源端口与目的端口分别用于接收和传输来自相邻簇的权重值——由于系统仅配置8×2个簇，因此只需配置一对端口，东/西端口均可省略。第二个源端口连接至GLB集群以接收片外数据，而第二个目标端口则连接至集群内的一行处理单元（PE）。权重网格网络路由器的实现方式与图11所示类似。

*3) 部分和运算的HM-NoC架构：*如图13所示，该架构在每个路由器集群中配置了四个部分和运算路由器，分别对应GLB集群中的每个部分和运算 SRAM 存储库，或等效地对应集群内各列处理单元（PE）的列存储库。与权重NoC架构类似，部分和NoC也针对特定数据流进行了简化设计——具体而言，部分和运算仅允许在处理单元的垂直方向进行累加。这是因为行静态数据流中权重会在处理单元间水平方向重复使用，导致无法实现水平方向的部分和运算累加。因此，集群间二维网格的水平连接可被移除，因为部分和运算不会在水平方向传递。在每个集群内部，处理单元通过垂直方向连接，集群中的每个路由器只需将部分和运算从GLB集群的存储库传输至对应处理单元的底部即可。

在PE集群的列结构中，每个psum路由器会从同一列的顶部接收更新后的psum值。因此，每个psum路由器都配备三个源端口和三个目的端口。其中，一个源端口用于接收来自北侧相邻路由器集群的数据，另一个目的端口则用于向南侧相邻路由器集群传输数据。第二对源端口和目的端口被分配到GLB中的psum存储库，而第三对目的端口对应PE集群某列底部的PE节点，第三对源端口则对应该列顶部的PE节点。

D. 可扩展性

HM-NoC架构的核心设计理念在于为Eyeriss v2提供强大的扩展能力。具体而言，当架构通过增加处理单元（PE）实现扩展时，针对相同问题规模的性能（即吞吐量）也应同步提升。但需要指出的是，性能表现受诸多因素影响，包括数据流特性、网络核心（NoC）设计、片上与片外数据传输带宽等。在后续的扩展性测试中，我们将设定以下假设条件：片外带宽不受限制，且工作负载不存在不平衡（即不存在稀疏性）。

我们对Eyeriss v2在三个不同规模下的性能进行了分析：256个处理单元（PE）、1024个处理单元和16384个处理单元，每个处理单元的处理速率为1 MAC/周期。所有规模的处理单元集群均采用固定阵列尺寸4×4个处理单元，处理单元集群数量按4×4、8×8和32×32的比例扩展。作为对比，我们还考察了原始Eyeriss（即Eyeriss v1）在相同规模下的可扩展性。对于Eyeriss v1，处理单元采用方形阵列排列，具体为16×16、32×32和128×128。两个版本的Eyeriss均采用行固定数据流机制。为快速评估大规模架构性能，我们构建了一个分析模型，该模型能根据两种Eyeriss版本中不同网络-on-chip（NoC）设计的数据分布和带宽限制，搜索出不同规模下性能最优的操作映射。

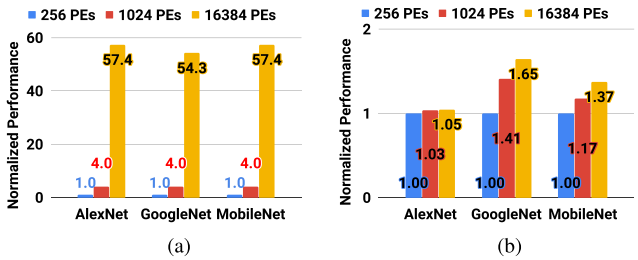


Fig. 14. Normalized performance of (a) Eyeriss v2 and (b) Eyeriss v1 running AlexNet, GoogLeNet, and MobileNet with a batch size of 1 at three different scales. Note that the MobileNet model has a width multiplier of 1.0 and an input size of 224 × 224, which is different from the MobileNet benchmarked in Section V.

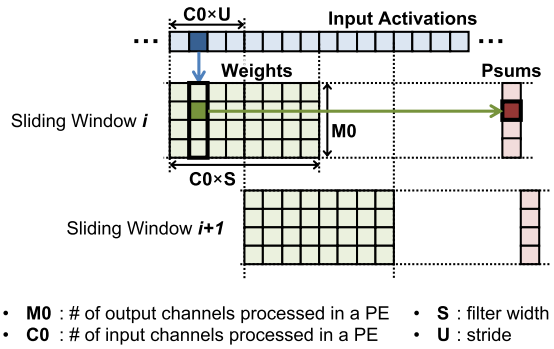


Fig. 15. Processing in the PE.

Fig. 14a and 14b show the normalized performance of Eyeriss v2 and Eyeriss v1, respectively, running three DNNs: AlexNet, GoogLeNet, and MobileNet (with width multiplier of 1.0 and input size of 224 × 224)[1] at the three different scales with a batch size of 1. For all three DNNs, the performance of Eyeriss v2 scales linearly from 256 to 1024 PEs, and achieves more than 85% of the linearly scaled performance at 16384 PEs. In contrast, the performance of Eyeriss v1 hardly improves when scaled up. This is due to the insufficient bandwidth provided by the broadcast NoC in Eyeriss v1 as discussed in Section III-A. For example, the performance of the FC layers in AlexNet and depth-wise layers in MobileNet do not see any improvement going from 256 PEs to 16384 PEs in Eyeriss v1 due to the insufficient NoC bandwidth for delivering weights and input activation, respectively, to the PEs. The HM-NoC in Eyeriss v2, however, is capable of adapting to the bandwidth requirements, therefore achieving higher performance at large scales. The HM-NoC is doing so while still being able to exploit available data reuse to achieve high energy efficiency, which will be demonstrated in Section V-A and V-B. Also note that, at large scales, the external data bandwidth will eventually become the performance bottleneck, and it will require more efforts to integrate the accelerator into the system to harness its full potential.

The implementation of the HM-NoC described in Section III-C targets the size of 8 × 2 PE clusters, and will require modifications when scaled up. Specifically, the mesh routers for input activations and weights need an extra pair of source and destination ports in order to handle data delivery for more than two columns of PE clusters. As the area and energy cost of the router grows with the number of ports, the overall cost will increase. However, the same routers can then be used for any architectural scales. Also, as will be shown in Section V, the entire NoC only accounts for less than 3% of the area and 6%-10% of the total energy consumption. The additional complexity in the routers is unlikely to add significant cost. In addition, the proportion of cost of different components will stay roughly constant as the system scales thanks to the design of the hierarchical mesh network.

[1]The large MobileNet model used here is not used for performance and energy efficiency benchmarking in Section V since the post-place-and-route simulation turn-around time is not practical; the smaller MobileNet model also has the same accuracy as AlexNet, which makes it a better comparison.

IV. Sparse Processing With SIMD Support

In the original Eyeriss, sparsity of input activations (iacts), i.e., zeros, is exploited to improve energy efficiency by gating the switching of logic and data accesses. In Eyeriss v2, we want to exploit sparsity further in *both* weights and iacts and improve not only energy efficiency but also throughput. Whereas the original Eyeriss only used compression between the GLB and off-chip DRAM, in Eyeriss v2, we keep the data in compressed form all the way to the PE. Processing in the compressed domain provides benefits in terms of reducing on-chip bandwidth requirements as well as on-chip storage, which can result in energy savings and throughput improvements. However, as compressed data often has variable length, this presents challenges in terms of how to manipulate the data (e.g., distributing data across PEs, and sliding window processing within the PE). In this section, we will introduce a new PE architecture that can process sparse data in the compressed domain for higher throughput. We will also introduce support for SIMD in the PE such that each PE can process two MACs per cycle.

A. Sparse PE Architecture

Fig. 15 illustrates how the PE processes uncompressed weights and iacts in the original Eyeriss, where $M0$ and $C0$ are the output and input channels processed within the PE, S is the filter width, and U is the stride. Recall that for the row-stationary dataflow, multiple 1-D rows of weights and iact are mapped to a given PE and processed in a sliding window fashion; here, the $C0 \times M0$ rows of weights with width S are assigned to the PE, and the weights belong to $M0$ output channels and $C0$ input channels. For each iact, the PE runs through $M0$ MAC operations sequentially in consecutive cycles with the corresponding column of $M0$ weights in the weight matrix, and accumulates to $M0$ partial sums (psums). By going through a window of $C0 \times S$ iacts in the stream, the processing goes through all $M0 \times C0 \times S$ weights in the matrix and accumulates to the same $M0$ psums. It then slides to the next window in the iact stream by replacing $C0 \times U$ iacts at the front of the window with new ones, and repeats the processing with the same weight matrix but accumulates to another set of psums. Note that the access pattern of weights goes through the entire weight matrix once sequentially in a column-major fashion for each window of iacts.

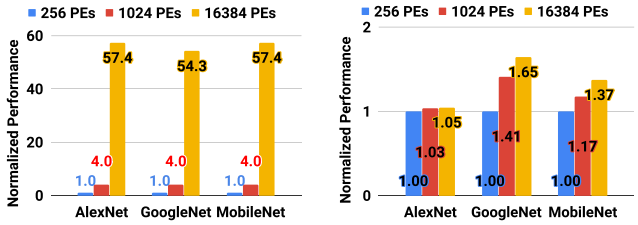


图14. (a) Eyeriss v2与(b) Eyeriss v1在AlexNet、GoogLeNet及MobileNet模型上运行时的标准化性能表现（批量大小为1），展示三种不同尺度下的对比结果。需特别说明：MobileNet模型的宽度乘数为1.0，输入尺寸为224×224，该参数与第五节中基准测试的MobileNet模型存在差异。

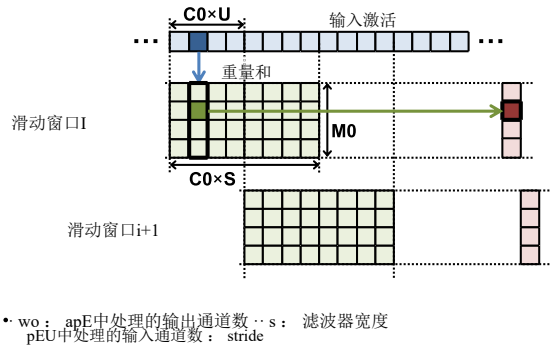


图15. PE中的处理过程。

图14a和14b分别展示了Eyeriss v2和Eyeriss v1在运行三个深度神经网络（AlexNet、GoogLeNet和MobileNet，宽度乘数为1.0，输入尺寸为224×224）1时，在三个不同规模下（批量大小为1）的标准化性能表现。对于所有三个DNN，Eyeriss v2的性能从256个处理单元线性扩展至1024个，当扩展到16384个处理单元时，性能仍保持线性扩展后85%以上的水平。相比之下，Eyeriss v1的性能在扩展时几乎毫无提升。这源于第三章A节讨论的广播型NoC在Eyeriss v1中提供的带宽不足问题。例如，由于NoC带宽不足，Eyeriss v1中AlexNet的全连接层和MobileNet的深度方向层在从256个处理单元扩展到16384个时，其性能均未见改善——前者无法向处理单元传输足够的权重，后者则无法传输足够的输入激活值。而Eyeriss v2采用的混合模式NoC（HM-NoC）能够适应带宽需求，因此在大规模扩展时实现了更高的性能表现。HM-NoC在实现高能效的同时，仍能充分利用现有数据复用技术，这一特性将在V-A和V-B章节中详细论证。值得注意的是，当系统规模扩大时，外部数据带宽将逐渐成为性能瓶颈，此时需要投入更多精力将加速器深度集成到系统中，才能充分发挥其全部潜力。

第三节C节所述的HM-NoC架构在设计时以8×2个并行执行器（PE）集群的规模为目标，但实际部署时需要进行扩展调整。具体而言，当处理超过两列PE集群的数据传输时，输入激活值和权重的网格路由器需要增加一对源端口和目的端口。虽然路由器的面积和能耗会随着端口数量增加而上升，但其整体成本仍可控。更重要的是，这类路由器可适配不同架构尺寸。如第五节所示，整个网络芯片（NoC）仅占系统面积的3%以下，能耗占比也控制在6%-10%。由于采用了分层网格网络设计，即使系统规模扩大，路由器的复杂度提升也不会带来显著成本增加。此外，得益于这种分层架构，不同组件的成本占比在系统扩展过程中将保持相对稳定。

1本文采用的大型MobileNet模型不适用于第五节的性能与能效基准测试，因其后置布局与布线的仿真周转时间不切实际；而小型MobileNet模型的准确度与AlexNet相当，因此成为更优的对比对象。

4.带有SIMD支持的稀疏处理

在原始Eyeriss架构中，通过利用输入激活值（iacts）的稀疏性（即零值），通过控制逻辑访问和数据访问的切换来提升能效。在Eyeriss v2版本中，我们希望进一步挖掘权重和iacts的稀疏特性，不仅优化能效，还要提升吞吐量。相较于原始Eyeriss仅在通用逻辑块（GLB）与片外DRAM之间进行压缩，Eyeriss v2将压缩数据完整保留在处理单元（PE）中。这种压缩域处理方式既能降低片上带宽需求，又能节省片上存储空间，从而实现节能增效。然而，由于压缩数据通常具有可变长度，这给数据处理方式（如跨PE数据分发、PE内部滑动窗口处理）带来了挑战。本节将介绍一种新型PE架构，该架构能在压缩域处理稀疏数据以提升吞吐量。同时，我们还将支持PE的SIMD指令集，使每个PE每个周期可处理两个MAC指令。

A. 稀疏PE体系结构

图15展示了原始Eyeriss中并行执行单元（PE）处理未压缩权重和iact的方式。其中，*M0*和*C0*分别是PE内部处理的输出通道和输入通道，*S*为滤波器宽度，*U*为步长。对于行平稳的数据流，多个一维权重行和iact行会被映射到特定PE并采用滑动窗口方式处理：$C0 \times M0$条宽度为*S*的权重行被分配到PE中，这些权重对应*M0*个输出通道和*C0*个输入通道。对于每个iact行，PE会在连续周期中依次执行*M0*次多级加法（MAC）运算，对应权重矩阵中的*M0*条列权重进行累加，最终得到*M0*个部分和（psums）。当处理$C0 \times S$条iact行的窗口时，整个处理过程会遍历矩阵中所有$M0 \times C0 \times S$条权重，并累积生成相同的*M0*个部分和。随后，系统通过用新权重矩阵替换窗口前端的$C0 \times U$ iacts，将处理流程滑动至iact流的下一个窗口，并重复相同权重矩阵的运算过程，但会累积生成另一组psums。需要特别说明的是，每个iact窗口的权重访问模式都会以列优先方式，依次遍历整个权重矩阵。

To speed up the processing when the iacts and/or weights are sparse, the goal is to read only the non-zero data in the iact stream and the weight matrix for processing. In addition, we only want to perform the read when *both* iact and weights are non-zero. The challenge, however, is to correctly and efficiently address data for all three data types. For example, when jumping between non-zero iacts in a window, the access pattern of weights does not go through the weight matrix sequentially anymore. Instead, additional logic is required to fetch the corresponding column of weights for the non-zero iact, which is not deterministic. Similarly, when jumping between non-zero weights in a weight column, it also has to calculate the address of the corresponding psum instead of just incrementing the address by one. Since the access order is also not deterministic, prefetching from the weight SPad is very challenging.

In order to achieve the processing of sparse data as described above, we take advantage of the compressed sparse column (CSC) compression format similar to what is described in [41], [44]. For each non-zero value in the data, the CSC format records a *count* value that indicates the number of leading zeros from the previous non-zero value in the uncompressed data stream; this is similar to the run length in run length coding (RLC). The count value can then be used to calculate the address change between the non-zero data. The added advantage of CSC over RLC is that it has an additional *address* value that allows the data to be broken into segments (e.g., columns) for easy handling, which we will discuss next; this, of course, also adds overhead in the compression.

Both the iact and weights are encoded in the CSC format. For iacts, the data stream is divided into non-overlapping $C0 \times U$ segments, and each segment is CSC encoded separately. Doing so enables sliding window processing, which replaces a segment of data with a new one from the stream when the window slides. Since the data length of each segment will be different after CSC coding, additional information is needed to address each encoded segment. Therefore, for each encoded segment, an *address* value is also recorded in the CSC format that indicates the start address of the encoded segment in the entire encoded stream. The filter weights are also encoded with CSC compression by dividing each column of $M0$ weights as a segment and encoding each segment separately. This helps enable fast access of each column of non-zero weights.

Fig. 16 shows an example of CSC compressed weights. The characters in the weight matrix indicate the locations of non-zero values. To read the non-zero weights from a specific column, e.g., column 1 (assuming indexing starts from 0), the PE first reads address[1] and address[2] from the address vector in the CSC compressed weights, which gives the inclusive lower bound and non-inclusive upper bound of the addresses, i.e., 2 and 5, respectively, for reading the data and count vector. The first address (in this example, address[1]) is the location of the first non-zero weight in each column, highlighted in bold in Fig. 16, within the data vector; it then goes through the three non-zero weights in the column, i.e., c, d and e, to perform the computation. If there is no non-zero weight in a column, the location of the next first non-zero value is repeated (e.g., since there are no non-zero values in column 3, the value 6 which is the location of g, is repeated such that the difference in consecutive address values is zero, which reflects the all zero column.). At the same time, the corresponding addresses of the psums to update can be calculated by accumulating the counts from the count vector.

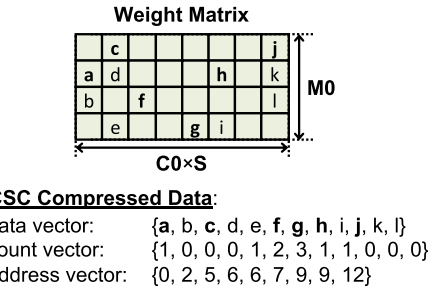


Fig. 16. Example of compressing sparse weights with compressed sparse column (CSC) coding. The first non-zero weight in each column is highlighted in bold; the address vector points to the location of these weights within the data vector. If there are no non-zero weights in a column next location is repeated (e.g., repeated 6 in address vector reflects the all zero column between **f** and **g**).

In the CSC format, the count vector is an overhead in addition to the non-zero data. If the bitwidth of the count is low, it may affect the compression efficiency when sparsity is high since the number of consecutive zeros can exceed the maximum count. If the count bitwidth is high, however, the overhead of the count vector becomes more significant. From our experiments, setting each count at 4b yields the best compression rate for the 8b iact and weights. Therefore, each count-data pair is 12b and is stored in a 12b word of the data SPad for both iact and weight. This is similar to setting the run-length in the RLC, where 5b was allocated to the run-length in [33].

In summary, both the weights and iacts can be processed directly in the CSC format. The processing can skip the zeros entirely without spending extra cycles, thus improving the processing throughput as well as energy efficiency.

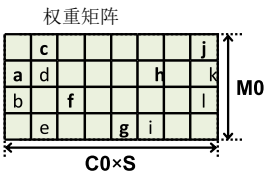
Fig. 17 shows the block diagram of the *sparse PE* that can perform the processing of CSC encoded iacts and weights directly as described above. Processing only non-zero data in the compressed format introduces read dependencies. For the compressed format, the address must be read before the data-count pair. To ensure only non-zero values are read, iact is read before the weight such that a non-zero weight is only read when the corresponding iact is non-zero. To handle these dependencies while still maintaining throughput, the PE is implemented using seven pipeline stages and five SPads. The first two pipeline stages are responsible for fetching non-zero iacts from the SPads. The iact address SPad stores the address vector of the CSC compressed iacts, which is used to address the reads from the iact data SPad that holds the non-zero data vector as well as the count vector. After a non-zero iact is fetched, the next three pipeline stages read the corresponding weights. Similarly, there is a weight address SPad to address the reads from the weight data SPad for the correct column of weights. The final two stages in the pipeline perform the MAC computation on the fetched non-zero iact and weight,

当iact和/或权重矩阵呈现稀疏特性时，为加速处理流程，我们的目标是仅读取iact数据流中的非零数据及权重矩阵进行处理。此外，我们仅在*iact和权重矩阵同时存在非零值时*执行读取操作。然而，真正的挑战在于如何正确且高效地寻址这三种数据类型。例如，在窗口内跳转非零iact时，权重矩阵的访问模式不再保持顺序性。此时需要额外逻辑来获取非零iact对应的权重矩阵列，这种操作具有非确定性。同理，当在权重矩阵列中跳转非零权重时，也需要计算对应psum的地址而非简单递增地址值。由于访问顺序同样具有非确定性，从权重矩阵的SPad预取数据变得极具挑战性。

为实现上述稀疏数据处理目标，我们采用了与文献[41][44]类似的压缩稀疏列（CSC）格式。对于数据中的每个非零值，CSC格式会记录一个*计数值*，该值表示从原始数据流中前一个非零值开始的前导零数量——这类似于游程编码（RLC）中的游程长度。通过该计数值，我们可以计算出非零数据之间的地址变化。相较于 RLC，CSC的额外优势在于其包含的*地址*值可将数据分割成便于处理的段（如列），我们将在下文详细讨论；当然，这种设计也会增加压缩过程中的开销。

iact和权重均采用CSC格式进行编码。对于iact数据流，会将其分割为$C0 \times U$个不重叠的段，每个段分别进行CSC编码。这种编码方式支持滑动窗口处理机制——当窗口移动时，会用数据流中的新段替换原有段。由于CSC编码后各段数据长度不同，需要额外信息来定位每个编码段。因此，每个编码段都会以CSC格式记录一个*地址*值，标明其在完整编码流中的起始位置。滤波器权重同样采用CSC压缩编码：将*M0权重矩阵*的每一列分割为独立段进行编码。这种设计能快速访问非零权重列，显著提升数据访问效率。

图16展示了CSC压缩权重的一个示例。权重矩阵中的字符表示非零值的位置。要从特定列（例如第1列，假设索引从0开始）读取非零权重时，处理单元（PE）首先从CSC压缩权重的地址向量中读取地址[1]和地址[2]，这给出了读取数据向量和计数向量的地址范围下限和上限（即2和5）。第一个地址（本例中为地址[1]）是数据向量中每列第一个非零权重的位置（图16中以粗体标出），随后依次遍历该列中的三个非零权重*c*、*d*和*e*进行计算。若某列中没有非零权重，则重复查找该列下一个第一个非零值的位置（例如，由于该列中没有非零权重...



压缩数据
数据向量：{a，b，C，d，e，f，g，h，i，j，k，I}
计数向量：{1,0,0,0,1,2,3,1,1,0,0,0}
地址向量：(0,2,5,6,6,7,9,9,12}

图16展示了压缩稀疏列（CSC）编码压缩稀疏权重的示例。每列中第一个非零权重会用粗体标出，地址向量则指向数据向量中这些权重的位置。若某列中没有非零权重，则该列的下一个位置会被重复（例如地址向量中重复的6表示**f**和**g**之间存在全零列）。

在第三列中，将*g*所在位置的数值6进行重复，使得相邻地址值的差值为零，从而形成全零列。与此同时，通过累加计数向量中的计数值，即可计算出需要更新的psum对应地址。

在CSC格式中，计数向量是除非零数据外的额外开销。如果计数的位宽较低，当稀疏度较高时，连续零的数量可能超过最大计数，这可能影响压缩效率。然而，如果计数的位宽较高，计数向量的开销会变得更加显著。根据我们的实验，将每个计数设置为4位时，对于8位的iact和权重，压缩率最佳。因此，每个计数-数据对为12位，并存储在数据SPad的12位字中，无论是iact还是权重。这类似于在 RLC 中设置游程长度，其中[33]中分配了5位给游程长度。

总之，权重和iacts均可直接以CSC格式进行处理。该处理过程可完全跳过零值，无需额外消耗周期，从而提高处理吞吐量及能效。

图17展示了可直接处理CSC编码iact和权重的*稀疏并行执行单元*（PE）的框图。在压缩格式下仅处理非零数据会引入读取依赖性——对于压缩格式，地址必须在数据计数对之前读取。为确保仅读取非零值，iact会在权重之前读取，使得非零权重仅在对应iact非零时才会被读取。为在保持吞吐量的同时处理这些依赖关系，该PE采用七个流水线阶段和五个SPad模块实现。前两个流水线阶段负责从SPad中获取非零iact。iact地址SPad存储CSC压缩iact的地址向量，用于从同时包含非零数据向量和计数向量的iact数据SPad中进行读取寻址。获取非零iact后，接下来的三个流水线阶段将读取对应的权重。类似地，还有一个权重地址SPad用于从权重数据SPad中对正确列的权重进行读取寻址。流程的最后两个阶段对获取的非零iact和权重执行MAC计算。

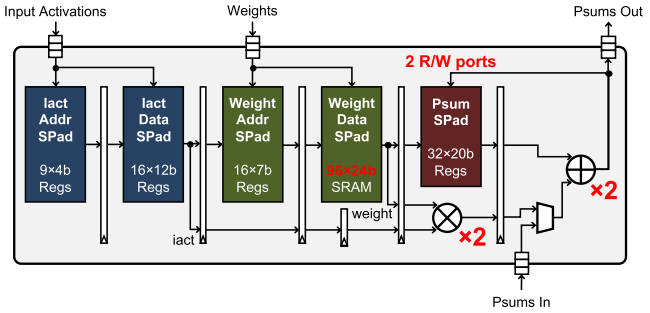


Fig. 17. Eyeriss v2 PE Architecture. The address SPad for both iact and weight are used to store addr vector in the CSC compressed data, while the data SPad stores the data and count vectors. The text in red denote changes for SIMD (Section IV-B).

and then send the updated psum either back to the psum SPad or out of the PE.

Since either iact and weight can be zero or non-zero, there are three possible scenarios:

- *If the iact is zero*, the CSC format will ensure that it is not read from the spad and therefore no cycles are wasted.
- *If the iact is not zero*, its value will be fetched from the iact data SPad and passed to the next pipeline stage.
  - *If there are non-zero weights corresponding to the non-zero iacts*, they will be passed down the pipeline for computation. The zero weights will be skipped since the weights are also encoded with the CSC format.
  - *If there are no non-zero weights corresponding to the non-zero iacts*, the non-zero iacts will not be further passed down in the pipeline. This may not necessarily introduce bubbles in the pipeline since the later stages, i.e., after the weight data Spad stage, can still be working on the computation for the previous non-zero iact if it has multiple corresponding weights.

In the Eyeriss v2 PE, the sizes of the iact address and data SPads are 9×4b and 16×12b, respectively, which allow for a maximum iact window size of 16. The sizes of the weight address and data SPads are 16×7b and 96×24b, respectively. This allows for a maximum weight matrix size of $96 \times (24b/12b) = 192$. The size of the psum SPad is 32×20b, and allows for a maximum weight matrix height of 32 (i.e., maximum number of output channels $M0$). If we fully utilize the iact SPads and psum SPad, it will require a weight matrix size of $32 \times 16 = 512$, which is larger than the limit of 192; however, the sparse PE design takes advantage of the fact that the sparse pattern of weights is known at compile time; therefore, it is possible to guarantee that the compressed weights will fit in a smaller SPad. Table III shows how many weights are stored in the SPad of each PE for sparse AlexNet. While in most layers the nominal number of weights is higher than 192, the number of non-zero weights after compression in the worst case is smaller and fits in the SPad for processing. By mapping more non-zero weights into each PE instead of mapping based on the nominal number of weights, more operations are performed in a PE, which statistically helps to reduce the amount of workload imbalance caused by the sparsity.

TABLE III
DISTRIBUTION OF WEIGHTS IN SPARSE ALEXNET TO THE SPAD IN EACH PE OF EYERISS V2

	$M0$	$C0$	S	Num. of Non-Zero Weights	
				Nominal	Compressed
CONV1	12	1	11	132	64
CONV2	32	2	5	320	86
CONV3	32	5	3	480	126
CONV4	24	4	3	288	100
CONV5	32	4	3	384	174
FC6	32	2	6	384	92
FC7	32	15	1	480	84
FC8	32	15	1	480	170

Since the degree of sparsity varies across different DNNs and data types, the PE is also designed to adapt to the scenarios when sparsity is low. In such cases, the PE can directly take in uncompressed iacts and weights instead of the CSC compressed versions to reduce the overhead in data traffic. Both iact and weight address SPads are not used and therefore clock-gated to save energy consumption, and the count in the CSC format is fixed to zero to address the data SPads correctly for processing.

B. SIMD Support in PE

Profiling results of the PE implementation shows that the area and energy consumption of the MAC unit is insignificant compared to other components in a PE. In Eyeriss, for example, the MAC unit takes less than 5% of the PE area, and only consumes 2%–9% of the PE power. This motivates the exploration of SIMD processing in a PE in order to achieve speedup of at most two times.

SIMD is applied to the PE architecture as shown in Fig. 17 by fetching two weights instead of one for computing two MAC operations per cycle with the same iact, i.e., a SIMD width of two. The changes are noted in the red text in the figure. SIMD processing not only improves the throughput but also further reduces the number of iact reads from the SPad. In terms of architectural changes, SIMD requires the word width of the weight data SPad to be two-word wide, which is why the size of the weight data SPad is 96×24b instead of 192×12b. The psum SPad also has to have two read and two write ports for updating two psums per cycle. In the case where only an odd number of non-zero weights exist in the column of $M0$ weights, the second 12b of the last 24b word in a column of non-zero weights is filled with zero. When the PE logic encounters the all-zero count-data pair, it clock-gates the second MAC datapaths as well as the read and write of the second ports in the psum SPad to avoid unnecessary switching, which reduces power consumption.

V. IMPLEMENTATION RESULTS

Eyeriss v2 was implemented in a 65nm CMOS process and the specifications of the design are summarized in Table IV. The design was placed-and-routed and the results reported in this section are from post-layout cycle-accurate gate-level simulations with (1) technology library from the worst PVT corner, (2) switching activities profiled from running the

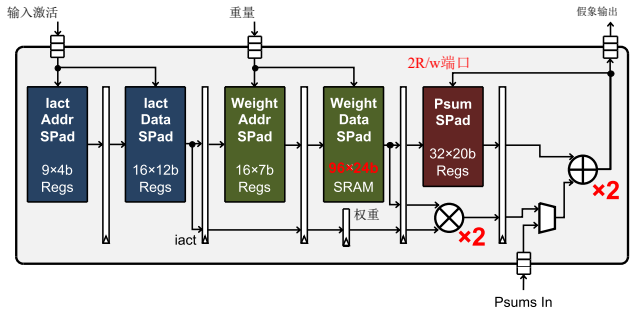


图17. Eyeriss v2 PE架构。地址SPad（地址存储单元）用于存储CSC压缩数据中的地址向量，同时存储iact和权重；数据SPad则存储数据向量和计数向量。红色文本标注了SIMD架构的修改内容（参见第IV-B节）。

随后将更新后的psum数据返回至psum SPad，或从PE中输出。

由于iact和重量均可为零或非零值，因此存在三种可能情况：

- *如果iact为零*，CSC格式将确保它不会从spad中读取，因此不会浪费周期。
- *如果iact不为零*，则从iact数据SPad中获取其值并传递给下一个管道阶段。
  - *如果存在与非零iacts相对应的非零权重*，它们将被传递到计算管道中。由于权重也采用CSC格式编码，零权重将被跳过。
  - *若无非零权重对应非零输入*，这些输入将不会在计算流水线中继续传递。这种情况未必会在流水线中引发数据泡，因为后续阶段（即权重数据Spad阶段之后）仍可对前序非零输入进行计算，前提是该输入存在多个对应权重。

在Eyeriss v2并行引擎中，iact地址和数据SPad的尺寸分别为9×4b和16×12b，允许最大iact窗口尺寸为16。权重地址和数据SPad的尺寸分别为16×7b和96×24b，这使得最大权重矩阵尺寸可达96×（24b/12b）=192。*psum SPad的尺寸为32×20b，允许最大权重矩阵高度为32（即最大输出通道数M0）*。若完全利用iact和psum SPad，将需要32×16=512的权重矩阵尺寸，这超过了192的限制；但稀疏并行引擎设计利用了编译时已知权重稀疏模式的特点，因此可以保证压缩后的权重能适配更小的SPad。表III展示了稀疏AlexNet中各并行引擎SPad存储的权重数量。虽然大多数层的名义权重数高于192，但在最坏情况下压缩后的非零权重数量会减少，仍能适配SPad进行处理。通过将更多非零权重映射至每个并行执行单元（PE）而非基于名义权重数量进行映射，可增加PE中的运算操作，从统计学角度有助于减少由稀疏性引起的负载不平衡。

表 III

稀疏 ALEXNET 中权重在 EYERISS V2 每个PE的SPAD中的分布

	$M0$	钴	S	非零权重的数量	
				名义的	压缩的
CONV1	12	1	11	132	64
CONV2	32	2	5	320	86
CONV3	32	5	3	480	126
CONV4	24	4	3	288	100
CONV5	32	4	3	384	174
FC6	32	2	6	384	92
FC7	32	15	1	480	84
FC8	32	15	1	480	170

由于不同深度神经网络和数据类型具有不同的稀疏程度，PE（处理单元）还专门设计了适应低稀疏场景的方案。在此类场景下，PE可直接接收未压缩的指令和权重数据，而非CSC压缩版本，从而降低数据传输的开销。由于指令和权重地址的SPad（数据存储单元）未被使用，因此采用时钟门控技术以节省能耗，同时将CSC格式的计数器固定为零，确保数据SPad能被正确寻址进行处理。

B. PE中的SIMD支持

对并行执行单元（PE）的性能分析表明，其MAC单元的面积和能耗与其他组件相比微不足道。以Eyeriss为例，MAC单元仅占PE面积的5%以下，功耗仅占2%–9%。这促使研究者探索在PE中采用SIMD（单指令多数据）处理技术，以实现最高两倍的加速效果。

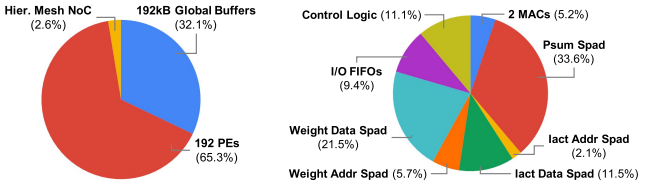
如图17所示，SIMD技术应用于并行执行（PE）架构时，通过每个周期获取两个权重数据而非单个权重，实现两个MAC运算的并行处理，此时SIMD宽度为2。图中红色文字标注了相关技术改进。SIMD处理不仅提升了吞吐量，还进一步减少了从SPad读取iact数据的次数。从架构层面来看，SIMD要求权重数据SPad的字宽为双字宽，因此权重数据SPad的尺寸为96×24位（192×12位的1/2），而非192×12位。此外，psum SPad需要配置两个读取端口和两个写入端口，以便每个周期更新两个psum值。当*M0权重列中存*在奇数个非零权重时，非零权重列中最后一个24位字的后12位会被填充为零。当PE逻辑检测到全零计数数据对时，会通过时钟门控技术同时阻断psum SPad中第二个MAC数据路径以及第二个端口的读写操作，从而避免不必要的开关切换，有效降低功耗。

V. 实施结果

Eyeriss v2采用65纳米CMOS工艺实现，其设计规格详见表IV。该设计已完成布局布线，本节报告结果来自布局后周期精确的门级仿真，仿真参数包括：(1)采用最严苛PVT条件下的工艺库，(2)基于运行时分析的开关活动特征。

TABLE IV
EYERISS V2 SPECIFICATIONS

Technology	TSMC 65nm LP 1P9M
Gate Count (logic only)	2695k (NAND-2)
On-Chip SRAM	246 KB
Number of PEs	192
Global Buffer	192 KB (SRAM)
Scratch Pads (per PE)	weight addr: 14B (Reg) weight data: 288B (SRAM) iact addr: 4.5B (Reg) iact data: 24B (Reg) psum: 80B (Reg)
Clock Rate	200 MHz
Peak Throughput	153.6 GOPS
Arithmetic Precision	weights & iacts: 8b fixed-point psums: 20b fixed-point



(a) (b)

Fig. 18. Eyeriss v2 area breakdown. (a) Overall Area Breakdown. (b) PE Area Breakdown.

actual weights of the DNNs and data from the ImageNet dataset [1], and (3) a batch size of one, which represents a more challenging setup for energy efficiency and throughput, but captures the low latency use case.

The overall gate count of Eyeriss v2, excluding SRAMs, is approximately 2695k NAND-2 gates. The area breakdown (Fig. 18) shows that the 192 PEs dominates the area cost, while the area of the hierarchical mesh networks of all data types combined only account for 2.6% of the area. This result proves that it is possible to build in high flexibility at a low cost. Within each PE, all of the SPads combined account for around 72% of the area, while the two MAC units only account for 5%.

A. Performance Analysis

To demonstrate the throughput and energy efficiency improvements brought on by the hierarchical mesh network and sparse PE architecture, we have implemented three different variants of Eyeriss: v1, v1.5, and v2. Table V lists the key differences between these Eyeriss variants. For the PE architecture, *Dense* means the PE can only clock-gate the cycles with zero data but not skip it, while *Sparse* means the PE can further skip the processing cycles with zero data. Eyeriss v1 is the same design as the original Eyeriss [33], but with the storage capacity, number of PEs and data precision scaled to the same level as v1.5 and v2 for a fair comparison. In short, the comparison between v1 and v1.5 shows the impact of the hierarchical mesh network, while the comparison between v1.5 and v2 shows the impact of the sparse PE architecture along with the support for SIMD processing. These architectures are placed-and-routed and benchmarked with four DNNs that have the same accuracy on the ImageNet

TABLE V
KEY DIFFERENCES BETWEEN THE THREE EYERISS VARIANTS. THE AREA IS LOGIC ONLY

	Eyeriss v1	**Eyeriss v1.5**	**Eyeriss v2**
Data Precision	activations & weights: 8b; partial sums: 20b		
# of PEs	192		
# of MACs	192	192	384
NoC	Multicast	Hier. Mesh	Hier. Mesh
PE Architecture	Dense	Dense	Sparse
PE SIMD Support	No	No	Yes
Global Buffer Size	192 kB		
Area (NAND-2 gates)	1394k	1354k	2695k

dataset: AlexNet [25], MobileNet (with a width multiplier of 0.5 and input size of 128×128) [10], and the sparse version of them as pruned by the method introduced in [14]. In this section, unless otherwise specified, AlexNet and MobileNet are referring to the dense model.

The implementation shows that Eyeriss v2 has an area increase of around two times compared to the other versions. The increase in mostly in the PE, which is 73% larger than the original one. The main reason is due to the need to support sparse processing, which requires deeper pipelining in the control logic and additional SPads to store the CSC compressed data. This contributes to a nearly 50% increase in area. Supporting SIMD also contributes to an additional 15% area increase due to the two sets of read and write ports for the psum SPad and the wider bus-width of the PE I/O in addition to the doubling of MAC units.

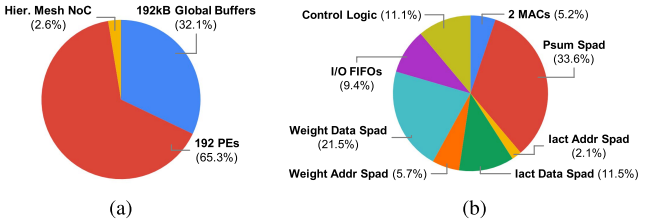
1) AlexNet: Fig. 19a shows the throughput improvements of different versions of Eyeriss on AlexNet over Eyeriss v1. Results on sparse AlexNet are also included (yellow bars) along with a breakdown of the processing latency across the different layers shown in Fig. 20. For AlexNet, the result shows that Eyeriss v1.5 significantly speeds up FC layers. This is because the throughput of FC layers is bandwidth-limited in Eyeriss v1, which is addressed by the hierarchical mesh network in Eyeriss v1.5. Eyeriss v2, on the contrary, significantly speeds up the CONV layers over Eyeriss v1.5 due to the increased number of multipliers and sparsity in the activations. However, the throughput of the FC layers only shows a marginal improvement because the FC layers are still bandwidth-limited even with the hierarchical mesh network. Therefore, speeding up the processing with sparsity and SIMD does not improve the throughput of FC layers as significantly as in CONV layers.

The full potential of Eyeriss v2, however, is fully revealed when coupled with sparse AlexNet. The bandwidth requirement of weights is lower in sparse AlexNet since it is very sparse, and the CSC compression can effectively reduce the data traffic. As a result, exploiting sparsity becomes more effective. Overall, Eyeriss v2 achieves 42.5× speedup with sparse AlexNet over Eyeriss v1 with AlexNet.

Fig. 19b shows the improvement on energy efficiency. It largely correlates to the speedup in Fig. 19a since the higher overall utilization of the PEs reduces the proportion of the static power consumption, e.g., clock network. Overall, Eyeriss v2 with sparse AlexNet is 11.3× more energy efficient than Eyeriss v1 with AlexNet.

表IV eyeriss v2规格

技术	TSMC 65nm LP 1P9M
门计数（仅逻辑）	2695K（NAND-2）
片上 SRAM	246 KB
pES数量	192
全局缓冲区	192 KB（SRAM）
刮板（每pE）	重量添加r：14B（Reg）重量数据：288B（SRAM）iact a ddr：4。5B（Reg）iact 数据：24B（Reg）p sum：80B（Reg）
时钟速率	200 MHz
峰值吞吐量	153.6 GOPS
算术精度	权重与i作用：8b定点p求和：20b定点



(a) (b)

图18. Eyeriss v2区域分解。(a) 总体区域分解。(b) PE区域分解。

(1) DNNs的实际权重及ImageNet数据集[1]的数据，以及(3)批量大小为一，这代表了对能效和吞吐量更具挑战性的设置，但能够捕捉低延迟用例。

Eyeriss v2芯片的总门电路数（不含SRAM）约为2695k个NAND-2门。面积占比分析（图18）显示，192个处理单元（PE）占据了主要面积，而所有数据类型组合的分层网格网络仅占2.6%。这一结果证明，以低成本实现高灵活性是完全可行的。在每个处理单元内部，所有存储单元（SPads）合计约占面积的72%，而两个MAC单元仅占5%。

A. 性能分析

为展示分层网格网络与稀疏并行计算架构带来的吞吐量提升和能效优化，我们开发了Eyeriss的三个版本：v1、v1.5和v2。表V列出了各版本的核心差异。在并行计算架构方面，*Dense*版本仅支持对无数据周期进行时钟门控（即暂停处理），无法直接跳过；*Sparse*版本则可进一步实现对无数据周期的跳过处理。Eyeriss v1的设计与原始版本[33]相同，但存储容量、并行计算单元数量及数据精度均调整至与v1.5和v2相当的水平，以确保公平对比。简而言之，v1与v1.5的对比展示了分层网格网络的优化效果，而v1.5与v2的对比则凸显了稀疏并行架构结合SIMD处理技术的优势。这些架构采用放置与路由设计，并通过四个在ImageNet数据集上达到相同准确率的深度神经网络进行基准测试。

表V

三种 EYERISS 变体之间的关键差异。

该区域仅限逻辑操作

	艾里斯S1	艾里斯S v1.5	艾里斯S2
数据精确度	激活函数与权重：8b；部分和：20b		
pES数	192		
MACS数	192	192	384
痛敏肽	多播	赫尔梅什	赫尔梅什
PE体系结构	稠密	稠密	稀疏
PE SIMD支持	不	不	对
全局缓冲区大小	192 kB		
区域（NAND-2门）	1394k	1354k	2695k

数据集包含：AlexNet [25]、MobileNet（宽度乘数为0.5，输入尺寸为128×128）[10]，以及通过[14]所述方法剪枝的稀疏版本。本节中，除非另有说明，AlexNet和MobileNet均指密集模型。

实际应用表明，Eyeriss v2的芯片面积较其他版本增加了约两倍。其中处理单元（PE）的面积增幅最大，较原版扩大了73%。这主要源于需要支持稀疏处理，该功能要求控制逻辑采用更深层次的流水线架构，并增加存储CSC压缩数据的专用处理单元（SPad）。这一改进导致芯片面积增加近50%。此外，由于支持SIMD指令集，加上psum SPad新增的两组读写端口、PE输入输出总线宽度的扩展以及MAC单元数量翻倍，使得芯片面积再增加15%。

*1) AlexNet：*图19a展示了Eyeriss不同版本在AlexNet上的吞吐量提升效果（相较于Eyeriss v1）。图中同时包含稀疏AlexNet的测试结果（黄色柱状图），以及图20所示各层处理延迟的详细分解。数据显示，Eyeriss v1.5在全连接层（FC层）实现了显著提速。这是因为Eyeriss v1的FC层吞吐量存在带宽限制，而v1.5通过引入分层网格网络结构解决了这一问题。与之形成对比的是，Eyeriss v2在激活函数稀疏性和乘法器数量增加的双重优势下，相较v1.5在卷积层（CONV层）实现了更显著的加速。但值得注意的是，即便采用分层网格网络，FC层的吞吐量提升幅度仍然有限。这说明通过稀疏化和SIMD技术加速处理，对FC层的吞吐量提升效果远不及卷积层显著。

然而，当Eyeriss v2与稀疏AlexNet结合使用时，其全部潜力得以充分释放。稀疏AlexNet因其高度稀疏性，其权重带宽需求较低，而CSC压缩技术能有效降低数据流量。因此，利用稀疏性变得更为高效。总体而言，相较于采用AlexNet的Eyeriss v1，Eyeriss v2与稀疏AlexNet结合可实现42.5倍的速度提升。

图19b展示了能效提升情况。这与图19a中的加速效果高度相关，因为处理单元（PE）的整体利用率提高，降低了静态功耗（如时钟网络）的比例。总体而言，采用稀疏AlexNet的Eyeriss v2比采用AlexNet的Eyeriss v1能效提升了11.3倍。

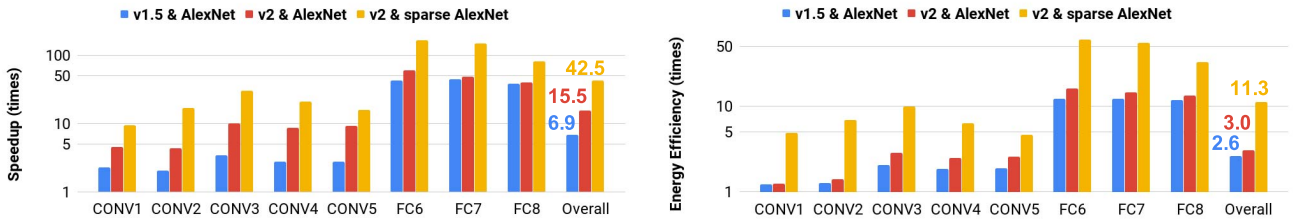


Fig. 19. (a) Speedup and (b) energy efficiency improvement of different versions of Eyeriss over Eyeriss v1 benchmarked with AlexNet.

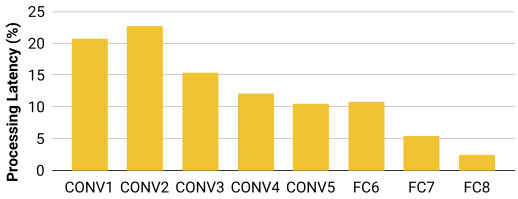


Fig. 20. Breakdown of processing latency across the different layers of sparse AlexNet running on Eyeriss v2.

2) MobileNet (Width Multiplier of 0.5, Input Size of 128 × 128*):* Fig. 21a and 21b show the improvement on throughput and energy efficiency, respectively, of different versions of Eyeriss on selected layers of MobileNet over Eyeriss v1. Results on sparse MobileNet are also included (yellow bars). The lack of data reuse in MobileNet results in low throughput on Eyeriss v1 due to the low-bandwidth NoC, which is why Eyeriss v1.5 can achieve a significant speedup over v1. However, the speedup of Eyeriss v2 over v1.5 is a mixed bag. While layers such as CONV1 and the point-wise (PW) layers can still take advantage of the sparsity in input activations to improve the throughput, the throughput of the Depth-wise (DW) CONV layers becomes worse. This is because the CSC compression does not create skippable cycles when the number of input and output channels are both one. Therefore, the sparse PE in Eyeriss v2 does not bring any advantage over the dense PE in Eyeriss v1.5. Furthermore, the deeper pipeline of the sparse PE actually makes the throughput slightly worse in the DW CONV layers.

Sparse MobileNet brings additional benefits on throughput and energy efficiency on Eyeriss v2; however, the improvement is not as significant as with the sparse AlexNet, since the CSC compression is less effective on sparse MobileNet than on sparse AlexNet due to its small layer sizes. Overall, Eyeriss v2 with sparse MobileNet is 12.6× faster and 2.5× more energy efficient than Eyeriss v1 with MobileNet.

B. Benchmark Results

Table VI summarizes the throughput and energy efficiency of Eyeriss v2 benchmarked with four DNNs that have comparable accuracy at a batch size of one. Although Eyeriss v2 achieves the highest GOPS/W[2] with sparse AlexNet, it consumes the least amount of time and energy per inference with the sparse MobileNet. *This result echoes the trend of DNN development going toward compact models that are more lightweight but also have less reuse for the hardware to explore, which makes it harder to reduce GOPS/W but can still improve inference/J*. Also, Eyeriss v2 achieves 12.6× higher inference/sec for MobileNet than AlexNet, which correlates well to the 14.7× reduction in the nominal number of MACs. This proves that the design has high flexibility to perform well for compact DNN models.

Fig. 22 shows the normalized power breakdown of Eyeriss v2 running a variety of DNN layers. We pick a representative set of layers to show how the different characteristics of the DNN layers impact the hardware. Note that these layers have different energy consumption and efficiency. The results are summarized as follows:

- CONV1 of AlexNet (148.1 GOPS/W) shows the case of no sparsity in both activations and weights. Compared to other layers, the high utilization of the PEs makes the proportion of the clock network power consumption low. It also has the highest proportion of MAC power consumption.
- CONV3 of sparse AlexNet (1423.2 GOPS/W) has the highest amount of sparsity in all layers we have tested. Compared to CONV3 of AlexNet (392.0 GOPS/W), the proportions of the clock network, HM-NoC and GLB power consumption are higher. This is mainly due to the workload imbalance induced by sparsity, which lowers the utilization of the active PEs. However, judging from the large proportion of the SPad and MAC power consumption compared to other components such as PE control logic, the PE is still kept fairly busy and data reuse is effectively exploited by the SPads.
- CONV13 DW layer of MobileNet (77.7 GOPS/W) has the lowest GOPS/W among all the layers we have tested. As expected, most of the energy is spent on the clock network. Inside the PE, the lack of reuse and not being able to utilize SIMD also hurt the energy efficiency, which is evident by the fact that most of the energy is spent in the control logic instead of the SPads or MACs.
- FC8 of sparse AlexNet (465.1 GOPS/W) shows the case of high sparsity and low data reuse. This combination makes the architecture more bandwidth-limited, and therefore the utilization of active PEs becomes low. That is why this layer has the highest proportion of power consumed by the clock network. The lack of reuse also makes the proportion of the SPad power consumption low

[2]In this paper, we calculate GOPS based on the nominal number of operations in the DNN, i.e., including operations with data values of zero.

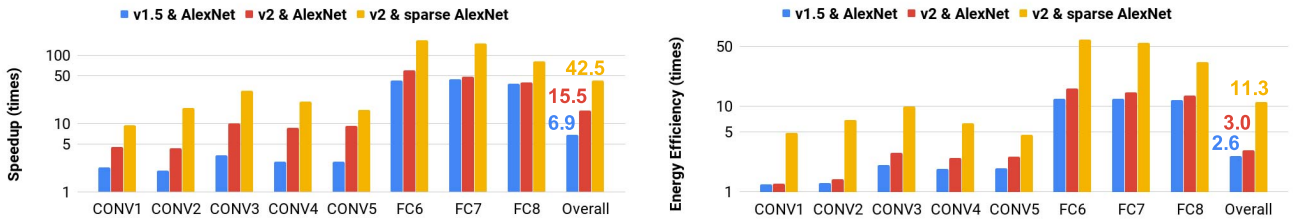


图19. (a) 不同版本Eyeriss相对于AlexNet基准测试的Eyeriss v1的加速比与(b) 能效提升情况。

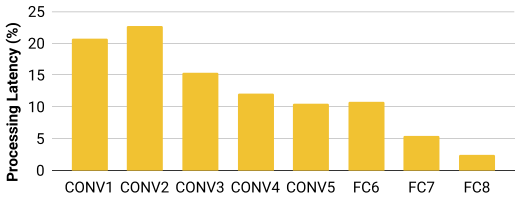


图20. 在Eyeriss v2上运行的稀疏AlexNet各层处理延迟分布

*2) MobileNet（宽度乘数0.5，输入尺寸*128×128）：图21a和21b分别展示了Eyeriss不同版本在MobileNet选定层上相较于v1版本在吞吐量和能效方面的提升。稀疏MobileNet的测试结果也包含在内（黄色柱状图）。由于MobileNet缺乏数据复用机制，导致Eyeriss v1在低带宽网络芯片上吞吐量较低，因此Eyeriss v1.5相较v1实现了显著加速。然而Eyeriss v2相较于v1.5的加速效果参差不齐：虽然CONV1层和逐点（PW）层仍能利用输入激活的稀疏性提升吞吐量，但深度（DW）卷积层的吞吐量却有所下降。这是因为当输入输出通道数均为1时，CSC压缩不会产生可跳过循环，导致Eyeriss v2的稀疏并行执行（PE）相较于v1.5的密集PE并无优势。更值得注意的是，稀疏PE的深层流水线结构反而使DW卷积层的吞吐量略有下降。

稀疏MobileNet在Eyeriss v2上带来了吞吐量和能效的额外优势；然而，由于其层尺寸较小，CSC压缩在稀疏MobileNet上的效果不如稀疏AlexNet显著，因此改进幅度不如稀疏AlexNet明显。总体而言，采用稀疏MobileNet的Eyeriss v2比采用MobileNet的Eyeriss v1快12.6倍，能效提升2.5倍。

B. 基准结果

表VI总结了Eyeriss v2与四个在批量大小为一的情况下具有可比准确性的DNN进行基准测试的吞吐量和能效。尽管Eyeriss v2在稀疏AlexNet上实现了最高的 GOPS /W2，但它消耗的时间和能量最少。

2在本文中，我们根据 DNN 中的标称操作数计算 GOPS，即包括数据值为零的操作。

在稀疏MobileNet架构下，*这一结果印证了深度学习模型（DNN）发展的趋势——追求更轻量化的紧凑型模型，但同时也减少了硬件可探索的复用空间，这使得在降低功耗（GOPS /W）的同时仍能提升推理速度变得更具挑战性*。此外，Eyeriss v2模型在MobileNet架构下的推理速度比AlexNet提升了12.6倍，这与模型中名义上MACs数量减少14.7倍的现象高度吻合。该设计充分展现了其在紧凑型 DNN 模型中保持高性能的高灵活性。

图22展示了Eyeriss v2运行多种 DNN 层时的归一化功耗分布。我们选择了一组具有代表性的层来展示不同 DNN 层特性对硬件的影响。请注意，这些层具有不同的能耗和效率。结果总结如下：

- AlexNet的CONV1（148.1 GOPS /W）显示激活和权重中没有稀疏性的情况。与其他层相比，PE的高利用率使得时钟网络功耗比例较低。它还具有最高的MAC功耗比例。
- 稀疏AlexNet的CONV3（1423.2 GOPS /W）在我们测试的所有层中具有最高的稀疏度。与AlexNet的CONV3（392.0 GOPS /W）相比，时钟网络、HM-NoC和GLB功耗的比例更高。这主要是由于稀疏性引起的负载不平衡，降低了活动PE的利用率。然而，从SPad和MAC功耗占比较其他组件（如PE控制逻辑）的比例来看，PE仍然保持相当繁忙，且SPad有效地利用了数据重用。
- 在我们测试的所有层中，MobileNet 的 CONV13 DW 层（77.7 GOPS /W）具有最低的 GOPS /W。正如预期的那样，大部分能量消耗在时钟网络上。在处理单元内部，缺乏重用和无法利用 SIMD 也影响了能效，这从控制逻辑消耗了大部分能量而不是 SPads 或 MACs 可以看出。
- 稀疏 AlexNet 的 FC8 层（465.1 GOPS /W）展示了高稀疏性和低数据复用的情况。这种组合使得架构更加受限于带宽，因此活动处理单元的利用率较低。这就是为什么该层的时钟网络消耗的功率比例最高。缺乏复用也使得 SPad 的功耗比例较低。

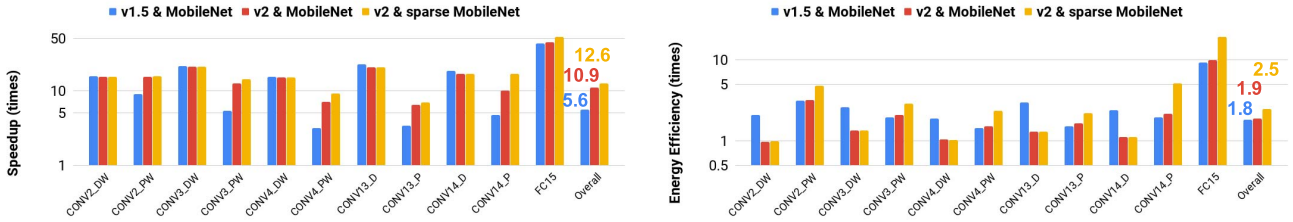


Fig. 21. (a) Speedup and (b) energy efficiency improvement of different versions of Eyeriss over Eyeriss v1 benchmarked with MobileNet. Due to the large number of layers, only a few representative layers are presented.

TABLE VI

THROUGHPUT AND ENERGY EFFICIENCY OF EYERISS V2 BENCHMARKED WITH FOUR DNNS THAT HAVE COMPARABLE ACCURACY AND A BATCH SIZE OF 1. NOTE THAT THE MOBILENET USED FOR BENCHMARK HAS A WIDTH MULTIPLIER OF 0.5 AND AN INPUT SIZE OF 128 × 128

DNN	ImageNet Accuracy[1]	Nominal Num. of MACs	Inference/sec	Inference/J	GOPS/W	DRAM Acc. (MB)	PE Utilization[2]
AlexNet	80.43%	724.4M	102.1	174.8	253.2	71.9	100%
sparse AlexNet	79.56%	724.4M	278.7	664.6	962.9	22.3	100%
MobileNet	79.37%	49.2M	1282.1	1969.8	193.7	4.1	91.5%
sparse MobileNet	79.68%	49.2M	1470.6	2560.3	251.7	3.9	91.5%

[1] top-5 accuracy for the image classification task.
[2] measured in terms of number of utilized MAC datapaths; each PE has 2 MAC datapaths.

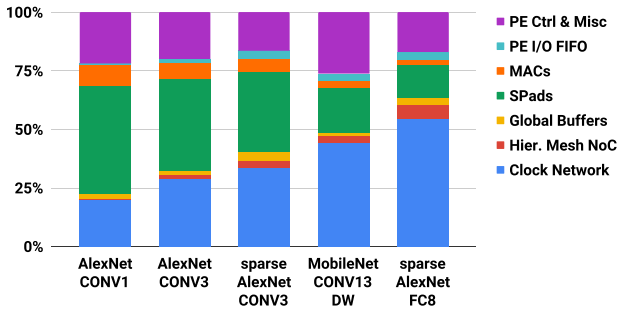


Fig. 22. Eyeriss v2 power breakdown running different DNN layers.

and the NoC power consumption high. However, thanks to sparsity, the overall energy efficiency of this layer is still better than CONV1 of AlexNet.

In terms of external DRAM accesses, AlexNet requires much more data than MobileNet as shown in Table VI, which is mainly due to the large amount of weights in the fully-connected layers. For CONV layers only, the required DRAM accesses are 7.1 MB and 4.6 MB for AlexNet and sparse AlexNet, respectively. Note that Eyeriss v2 does not perform the pooling layers on-chip, and the required DRAM accesses will further decrease if pooling layers are processed on-chip. We have also profiled the impact of a limited peak external bandwidth on the performance. With an aggregated external read and write bandwidth of 25600 MB/s, which is at the level of DDR4-3200, the throughput of Eyeriss v2 running sparse AlexNet and sparse MobileNet will decrease by 16% and 24%, respectively, due to the bursty external data access patterns. However, we believe that additional on-chip buffering can alleviate the performance degradation, which we will leave for future endeavors. This result also confirms that, with efficient hardware that can maximize utilization even when data reuse is low, DNNs that do not have enough data reuse to exploit will put more pressure on the external data bandwidth, which should be addressed in the design of future DNN models.

C. Comparison With Prior Art

Table VII shows the comparison between Eyeriss v2 and the state-of-the-art prior art. Eyeriss v2 is the first one to report benchmark results on both large DNNs, e.g., AlexNet, and compact DNNs, e.g., MobileNet. For AlexNet, Eyeriss v2 still achieves comparable throughput and slightly less energy efficiency compared to other works that are tailored for the large models. This result is achieved with a batch size of one (while other results use larger batch sizes), and the overhead associated with its additional flexibility to handle the drastically different layer shapes in the compact models. We report results for Eyeriss v2 on a sparse network which is a widely used approach for large DNN models, particularly on mobile devices; unfortunately, the available results for the other works are only on AlexNet. We would expect the sparse AlexNet to potentially provide additional energy efficiency improvements on those works, but not throughput improvements.

For MobileNet, Eyeriss v2 achieves 5.3× throughput improvement and 3.9× energy improvement over AlexNet, with the same accuracy. Although the other designs do not report results for MobileNet, our understanding of those designs leads us to believe that they would not achieve comparable improvements, similar to the original Eyeriss, due to the NoC limitations as well as additional mapping inefficiencies of the dataflow. However, we conjecture that the NoC limitations can be addressed by the proposed HM-NoC.

D. Discussion

Eyeriss v2 focuses its design on improving the throughput and energy efficiency for compact and sparse DNN models,

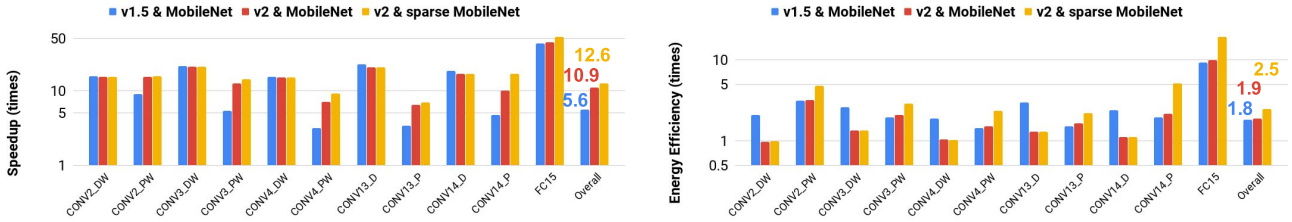


图21. (a) 不同版本Eyeriss相对于Eyeriss v1基准模型在MobileNet上的加速比及(b) 能效提升情况。由于层数较多，仅展示部分代表性层。

表VI

吞吐量和能效的 EYERISS V2 基准测试，使用了四个具有可比精度且批量为1的 DNNS 。请注意，用于基准测试的 MOBILENET 宽度乘数为0.5，输入尺寸为 128 × 128 。

5-二硝基萘	图像网络准确度[1]	MACS的名义数量	每秒推断次数	推断/判断	GOPS/W	动态随机存取存储器 (MB)	PE利用率[2]
亚历克斯·内特	80.43%	724.4M	102.1	174.8	253.2	71.9	100%
稀疏Alex网络	79.56%	724.4M	278.7	664.6	962.9	22.3	100%
移动网络	79.37%	49.2M	1282.1	1969.8	193.7	4.1	91.5%
稀疏移动网	79.68%	49.2M	1470.6	2560.3	251.7	3.9	91.5%

[1] 在阳离子任务中，图像类别的前5名准确率。
以使用的MAc数据路径数量为衡量标准；每个pE具有2条MAc数据路径。

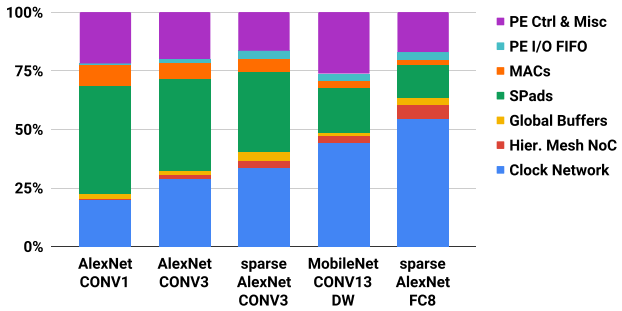


图22. Eyeriss v2电源在运行不同 DNN 层时的功率分解。

且神经网络（NoC）功耗较高。然而，得益于稀疏性，该层的整体能效仍优于AlexNet的CONV1层。

在外部DRAM访问方面，如表VI所示，AlexNet需要比MobileNet多得多的数据，这主要是由于全连接层中权重数量庞大。仅针对CONV层，AlexNet和稀疏AlexNet所需的DRAM访问量分别为7.1 MB和4.6 MB。需要注意的是，Eyeriss v2并未在芯片上执行池化层，如果在芯片上处理池化层，所需的DRAM访问量将进一步减少。我们还分析了有限峰值外部带宽对性能的影响。当外部读写总带宽为25600 MB/s（相当于DDR4-3200的水平）时，由于突发性的外部数据访问模式，运行稀疏AlexNet和稀疏MobileNet的Eyeriss v2吞吐量将分别下降16%和24%。然而，我们认为额外的片上缓冲可以缓解性能下降，这将留待未来研究。该结果也证实，即使数据重用率较低，只要硬件高效利用资源，即使数据重用不足，深度神经网络也能充分发挥潜力。

将对外部数据带宽施加更大压力，这应在未来 DNN 模型的设计中予以解决。

C. 与现有技术的比较

表七展示了Eyeriss v2与当前最先进技术的对比。Eyeriss v2是首个同时报告大型DNN（如AlexNet）和紧凑型DNN（如MobileNet）基准测试结果的系统。对于AlexNet，Eyeriss v2仍能实现与其他针对大型模型优化的工作相当的吞吐量，但能效略低。这一结果是在单批次大小下实现的（而其他结果使用了更大的批次大小），并考虑了其处理紧凑型模型中截然不同层形状的额外灵活性所带来的开销。我们报告了Eyeriss v2在稀疏网络上的结果，这是大型DNN 模型（特别是在移动设备上）广泛采用的方法；遗憾的是，其他工作的可用结果仅限于AlexNet。我们预计稀疏AlexNet可能在这些工作中提供额外的能效改进，但不会带来吞吐量的提升。

在MobileNet架构上，Eyeriss v2在保持相同准确率的前提下，相比AlexNet实现了5.3倍的吞吐量提升和3.9倍的能耗优化。虽然其他设计方案未公布MobileNet的具体数据，但根据现有设计分析，由于NoC架构的限制以及数据流映射效率的额外损耗，这些方案难以达到与原始Eyeriss相当的性能提升。不过我们推测，所提出的HM-NoC架构或许能有效解决NoC架构的局限性。

D. 讨论

Eyeriss v2的设计重点是提高紧凑和稀疏 DNN 模型的吞吐量和能效，

TABLE VII

COMPARISON WITH STATE-OF-THE-ART DESIGNS. FOR EYERISS V2, THE THROUGHPUT AND ENERGY EFFICIENCY ARE BENCHMARKED ON THE SPARSE VERSION OF ALEXNET AND MOBILENET

		Eyeriss [33]	**ENVISION** [15]	**Thinker** [16]	**UNPU** [17]	**This Work**	
Technology		65nm	28nm	65nm	65nm	65nm	
Area		1176k gates (NAND-2)	1950k gates (NAND-2)	2950k gates (NAND-2)	4.0mm×4.0mm (Die Area)	2695k gates (NAND-2)	
On-chip SRAM (kB)		181.5	144	348	256	246	
Max Core Frequency		200 MHz	200 MHz	200 MHz	200 MHz	200 MHz	
Bit Precision		16b	4b/8b/16b	8b/16b	1b-16b	8b	
Num. of MACs		168 (16b)	512 (8b)	1024 (8b)	13824 (bit-serial)	384 (8b)	
DNN Model		AlexNet	AlexNet	AlexNet	AlexNet	sparse AlexNet	sparse MobileNet
Batch Size		4	N/A	15	N/A	1	1
Core Frequency (MHz)		200	200	200	200	200	200
Bit Precision		16b	N/A	adaptive	8b	8b	8b
Inference/sec	(CONV only)	34.7	47	-	346	342.4	-
	(Overall)	-	-	254.3	-	278.7	1470.6
Inference/J	(CONV only)	124.8	1068.2	-	1097.5	743.4	-
	(Overall)	-	-	876.6	-	664.6	2560.3

which is very different from the direction taken in many of the state-of-the-art previous works. With a similar amount of resources, i.e., area, Eyeriss v2 has much fewer number of MACs. However, with the flexibility of the on-chip network and the sparse processing logic that effectively improve throughput based on the sparsity of the data, Eyeriss v2 still achieves comparable throughput and energy efficiency for large DNNs against the state-of-the-art that optimizes directly for them. Furthermore, Eyeriss v2 shows a significant throughput and energy efficiency improvement on sparse MobileNet against Eyeriss v1 as shown in Section V-A.

Supporting sparse processing is a challenging task from an architecture design point of view. First of all, the PE design complexity and cost becomes much higher due to the additional required logic and storage. This has resulted in a significant increase in area as shown in Table V. In addition, it makes the support for high SIMD width processing difficult because of the workload imbalance and the high cost in the SPad due to the non-deterministic access patterns. Eyeriss v2, however, still demonstrates a design that can effectively translate the sparsity to significant throughput and energy efficiency improvement as compared to Eyeriss v1.

It is worth noting that the flexibility provided by the hierarchical mesh network and the throughput boost from the sparse processing logic can be applied separately. Therefore, if sparse networks are not the target workload, the flexible NoC can still be used in conjunction with other techniques such as lower precision and higher parallelism to achieve higher throughput and energy efficiency.

VI. CONCLUSION

DNNs are rapidly evolving due to the significant amount of research in the field; however, the current direction of DNN development also brings new challenges to the design of DNN accelerators due to the widely varying layer shapes in compact DNNs and the varying data sparsity in sparse DNNs. In this work, we propose a new DNN accelerator architecture, called Eyeriss v2, that addresses these challenges. First, the varying layer shapes makes the on-chip network (NoC) the performance bottleneck since conventional NoC design poses strong assumptions on the amount of data reuse and required data bandwidth for each data type, which is too rigid to adapt. We solve this problem by introducing the hierarchical mesh network (HM-NoC). HM-NoC can be configured into different modes that can deliver from high bandwidth to high data reuse. More importantly, its implementation cost is also minimized through the hierarchical design that limit the costly all-to-all communication within local clusters as well as the circuit-switched routing. This helps to bring over an order of magnitude speedup for processing MobileNet compared to the original Eyeriss, i.e., Eyeriss v1, scaled to the same number of multipliers and storage capacity as Eyeriss v2. Furthermore, Eyeriss v2 incorporates a new PE architecture that support processing sparse weights and input activations directly in compressed domain to improve not only energy efficiency but also throughput. It also adds SIMD support so that each PE can process 2 MACs per cycles. Overall, Eyeriss v2 achieves 42.5× and 11.3× improvement in throughput and energy efficiency, respectively, with sparse AlexNet compared to Eyeriss v1 running AlexNet; it also achieves 12.6× and 2.5× improvement in throughput and energy efficiency, respectively, with sparse MobileNet compared to Eyeriss v1 running MobileNet.

REFERENCES

[1] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[2] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[3] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," in *Proc. NIPS*, 2015, pp. 3123–3131.

[4] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. ECCV*, 2016, pp. 525–542.

[5] E. H. Lee, D. Miyashita, E. Chai, B. Murmann, and S. S. Wong, "LogNet: Energy-efficient neural networks using logarithmic computation," in *Proc. ICASSP*, Mar. 2017, pp. 5900–5904.

[6] F. Li and B. Liu, "Ternary weight networks," in *Proc. NIPS Workshop Efficient Methods Deep Neural Netw.*, 2016.

[7] B. Moons, B. De Brabandere, L. Van Gool, and M. Verhelst, "Energy-efficient ConvNets through approximate computing," in *Proc. WACV*, Mar. 2016, pp. 1–8.

表VII

与现有最先进设计的比较。对于 EYERISS V2，，其吞吐量和能效在稀疏版本的 ALEXNET 和 MOBILENET 上进行了基准测试。

	眼裂[33]	远见[15]	Therapewa[16]	**UNPU** [17]	本工作	
技术语言	65纳米	28纳米	65纳米	65纳米	65纳米	
区域	1176k 门 （ 与非-2）	1950k 门 （ 与非-2）	2950k 门 （ 与非-2）	4。0mmx4。0mm （ 模具面积）	2695k 门 （ 与非-2）	
片上存储器	181.5	144	348	256	246	
最大核心频率	200 MHz	200 MHz	200 MHz	200 MHz	200 MHz	
位精度	16b	4b/8b/16b	8b/16b	1b-16b	8b	
MACS编号	168 (16b)	512 (8b)	1024 (8b)	13824（位串行）	384 (8b)	
DNN 模型	亚历克斯·内特	亚历克斯·内特	亚历克斯·内特	亚历克斯·内特	稀疏Alex网络	稀疏移动网
批量大小	4	N/A	15	N/A	1	1
核心频率	200	200	200自适应的	200	200	200
位精度	16b	N/A		8b	8b	8b
每秒推断次数（仅CON v）（总体）	34.7 -	47 -	- 254.3	346 -	342.4 278.7	- 1470.6
推理/J（仅CON v）（总体）	124.8 -	1068.2 -	六六六	1097.5 -	743.4 664.6	- 2560.3

这与许多现有前沿研究的开发方向大相径庭。在资源面积相近的情况下，Eyeriss v2的MAC数量显著减少。但凭借片上网络的灵活性和基于数据稀疏性优化的稀疏处理逻辑，Eyeriss v2仍能为大型深度神经网络（DNN）实现与直接针对其优化的前沿方案相当的吞吐量和能效。如第五章A节所示，相较于Eyeriss v1，Eyeriss v2在稀疏MobileNet上的吞吐量和能效表现更是实现了质的飞跃。

从架构设计的角度来看，支持稀疏处理是一项极具挑战性的任务。首先，由于需要额外的逻辑和存储模块，处理单元（PE）的设计复杂度和成本显著增加，导致芯片面积大幅扩大（如表V所示）。此外，由于工作负载分布不均以及非确定性访问模式导致的SPad区域高成本，这也使得高SIMD宽度处理的支持变得困难。不过，Eyeriss v2的设计仍能有效将稀疏性转化为显著的吞吐量提升和能效优化，相比Eyeriss v1实现了质的飞跃。

值得注意的是，分层网格网络提供的灵活性与稀疏处理逻辑带来的吞吐量提升可独立应用。因此，若稀疏网络并非目标工作负载，灵活的NoC仍可与其他技术（如降低精度和提高并行度）结合使用，以实现更高的吞吐量和能效。

六.结论

深度神经网络由于该领域的大量研究而迅速发展；然而，当前 DNN 的发展方向也给 DNN 加速器的设计带来了新的挑战，这主要是因为紧凑型DNN中层形状的广泛变化以及稀疏DNN中数据稀疏性的差异。在本工作中，我们提出了一种新的 DNN 加速器架构，称为Eyeriss v2，以应对这些挑战。首先，层形状的差异使得片上网络（NoC）成为性能瓶颈，因为传统的NoC设计该方案对各类数据的复用量和所需带宽提出了过于严苛的假设，导致其灵活性不足。我们通过引入分层网格网络（HM-NoC）解决了这一问题。HM-NoC可配置为不同模式，既能提供高带宽，又能实现高数据复用。更重要的是，其分层设计通过限制本地集群内的全连接通信和电路交换路由，有效降低了实现成本。相较于原始Eyeriss（即Eyeriss v1），在保持相同乘法器数量和存储容量的前提下，Eyeriss v2的处理速度提升了十倍以上。此外，Eyeriss v2采用新型处理单元（PE）架构，支持直接在压缩域处理稀疏权重和输入激活，不仅提升了能效，还优化了吞吐量。同时新增SIMD支持，使每个处理单元每周期可处理2个MAC。总体而言，与运行AlexNet的Eyeriss v1相比，稀疏AlexNet在Eyeriss v2上实现了吞吐量42.5倍、能效11.3倍的提升；同时达到12.6倍和2倍的性能提升。与运行MobileNet的Eyeriss v1相比，稀疏MobileNet分别实现了5倍的吞吐量和能效提升。

参考文献

[1] O.Russakovsky等人，“ImageNet大规模视觉识别挑战”，*国际计算机视觉杂志*，第115卷，第3期，第211-252页，2015年12月。

[2] V. Sze，Y.-H. Chen，T.-J. Yang，and J. S. Emer，《深度神经网络的高效处理：教程和综述》，*IEEE学报*，第105卷，第12期，第2295-2329页，2017年12月。

[3] M.库尔巴里奥、Y.本吉奥与J.-P.大卫合著的《二进制连接：在传播过程中使用二进制权重训练深度神经网络》，载于2015年*NIPS会议论文集*，第3123–3131页。

[4] M. Rastegari、V. Ordonez、J. Redmon和A. Farhadi，《 XNOR -Net：基于二元卷积神经网络的ImageNet分类》，收录于《*ECCV会议论文集*》2016年，第525–542页。

[5] E.H.Lee，D.Miyashita，E.Chai，B.Murmann，和S.S.Wong，“LogNet：使用对数计算的节能神经网络”，在*Proc. ICASSP*，2017年3月，第5900-5904页。

[6] F. Li和B. Liu，“三元权重网络”，*NIPS高效方法深度神经网络研讨会论文集*，2016。

[7] B. Moons、B. De Brabandere、L. Van Gool和M. Verhelst合著的《通过近似计算实现节能的卷积神经网络》一文，发表于2016年3月的*WACV 会议论文集*，第1-8页。

[8] P. Judd, J. Albericio, T. Hetherington, T. M. Aamodt, N. E. Jerger, and A. Moshovos, "Proteus: Exploiting numerical precision variability in deep neural networks," in *Proc. Int. Conf. Supercomput.*, 2016, Art. no. 23.
[9] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE CVPR*, Jun. 2015, pp. 1–9.
[10] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, Apr. 2017.
[11] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. (2016). "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size." [Online]. Available: https://arxiv.org/abs/1602.07360
[12] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. NIPS*, 1990, pp. 598–605.
[13] 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.*, 2015, pp. 1135–1143.
[14] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proc. CVPR*, Jul. 2017, pp. 6071–6079.
[15] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "14.5 envision: A 0.26-to-10 TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28 nm FDSOI," in *ISSCC Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2017, pp. 246–247.
[16] S. Yin *et al.*, "A 1.06-to-5.09 TOPS/W reconfigurable hybrid-neural-network processor for deep learning applications," in *Proc. Symp. VLSI Circuits*, Jun. 2017, pp. C26–C27.
[17] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: A 50.6 TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *ISSCC Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2018, pp. 218–220.
[18] S. Sharify, A. D. Lascorz, K. Siu, P. Judd, and A. Moshovos, "Loom: Exploiting weight and activation precisions to accelerate convolutional neural networks," in *Proc. 55th Annu. Design Automat. Conf.*, 2018, pp. 20:1–20:6.
[19] P. Judd, J. Albericio, T. Hetherington, T. M. Aamodt, and A. Moshovos, "Stripes: Bit-serial deep neural network computing," in *Proc. 49th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2016, pp. 19:1–19:12.
[20] R. Andri, L. Cavigelli, D. Rossi, and L. Benini, "YodaNN: An ultra-low power convolutional neural network accelerator based on binary weights," in *Proc. ISVLSI*, Jul. 2016, pp. 236–241.
[21] K. Ando *et al.*, "BRein memory: A 13-layer 4.2 K neuron/0.8 M synapse binary/ternary reconfigurable in-memory deep neural network accelerator in 65 nm CMOS," in *Proc. Symp. VLSI Circuits*, Jun. 2017, pp. C24–C25.
[22] Z. Jiang, S. Yin, M. Seok, and J.-S. Seo, "XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks," in *Proc. IEEE Symp. VLSI Technol.*, Jun. 2018, pp. 173–174.
[23] D. Bankman, L. Yang, B. Moons, M. Verhelst, and B. Murmann, "An always-on 3.8 μJ/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28-nm CMOS," in *ISSCC Dig. Tech. Papers*, 2018, pp. 222–224.
[24] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A mixed-signal binarized convolutional-neural-network accelerator integrating dense weight storage and multiplication for reduced data movement," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2018, pp. 141–142.
[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.
[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, Sep. 2014.
[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, Jun. 2016, pp. 770–778.
[28] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. CVPR*, Jun. 2016, pp. 2818–2826.
[29] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," *CoRR*, Jun. 2015.
[30] Z. Du *et al.*, "ShiDianNao: Shifting vision processing closer to the sensor," in *Proc. ISCA*, 2015, pp. 92–104.
[31] Nvidia. (2017). *NVDLA Open Source Project*. [Online]. Available: http://nvdla.org/
[32] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. ACM/IEEE 44th Annu. Int. Symp. Comput. Archit.*, Jun. 2017, pp. 1–12.
[33] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.
[34] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Understanding the limitations of existing energy-efficient design approaches for deep neural networks," in *Proc. SysML*, 2018.
[35] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1520–1528.
[36] A. Dosovitskiy *et al.*, "FlowNet: Learning optical flow with convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 2758–2766.
[37] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *Proc. 4th Int. Conf. 3D Vis. (3DV)*, Oct. 2016, pp. 239–248.
[38] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
[39] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos, "Cnvlutin: Ineffectual-neuron-free deep neural network computing," in *Proc. ISCA*, 2016, pp. 1–13.
[40] S. Zhang *et al.*, "Cambricon-x: An accelerator for sparse neural networks," in *Proc. ISCA*, 2016, Art. no. 20.
[41] S. Han *et al.*, "EIE: Efficient inference engine on compressed deep neural network," in *Proc. ISCA*, Jun. 2016, pp. 243–254.
[42] A. Parashar *et al.*, "SCNN: An accelerator for compressed-sparse convolutional neural networks," in *Proc. ACM/IEEE 44th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2017, pp. 27–40.
[43] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *Proc. 43rd Int. Symp. Comput. Archit. (ISCA)*, 2016, pp. 367–379.
[44] R. Dorrance, F. Ren, and D. Marković, "A scalable sparse matrix-vector multiplication kernel for energy-efficient sparse-blas on FPGAs," in *Proc. ISFPGA*, 2014, pp. 161–170.

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. and Ph.D. degrees in electrical engineering and computer science (EECS) from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2013 and 2018, respectively.

Since 2018, he has been a Research Scientist with the Nvidia's Architecture Research Group, Santa Clara, CA, USA. His current research focuses on the design of computer architectures for machine learning, deep learning, and domain-specific processors. In 2018, he received the Jin-Au Kong Outstanding Doctoral Thesis Prize in electrical engineering from MIT. He was a recipient of the 2015 Nvidia Graduate Fellowship, the 2015 ADI Outstanding Student Designer Award, and the 2017 IEEE SSCS Predoctoral Achievement Award. His work on the dataflows for CNN accelerators was selected as one of the Top Picks in computer architecture in 2016. He also co-taught a tutorial on Hardware Architectures for Deep Neural Networks at MICRO-49, ISCA2017, and MICRO-50.

Tien-Ju Yang (S'11) received the B.S. degree in electrical engineering and the M.S. degree in electronics engineering from National Taiwan University (NTU), Taipei, Taiwan, in 2010 and 2012, respectively. He is currently pursuing the Ph.D. degree in electrical engineering and computer science with the Massachusetts Institute of Technology, Cambridge, MA, USA, with a focus on energy-efficient deep neural network design. From 2012 to 2015, he was with the Intelligent Vision Processing Group, MediaTek Inc., Hsinchu, Taiwan, as an Engineer. His research interest spans the area of computer vision, machine learning, image/video processing, and VLSI system design. He won the First Place in the 2011 National Taiwan University Innovation Contest.

[8] P. Judd，J. Albericio，T. Hetherington，T. M. Aamodt，N. E. Jerger，and A. Moshovos，“Proteus：利用深度神经网络中的数值精度变异性”，*国际超级计算会议论文集*，2016，文章编号23。
[9] C.Szegedy等人，“用卷积深入挖掘”，*IEEECVPR会议论文集*，2015年6月，第1-9页。
[10] A. G. Howard等人，“MobileNets：用于移动视觉应用的高效卷积神经网络”，*CoRR*，2017年4月。
[11] F. N. Iandola、S. Han、M. W. Moskewicz、K. Ashraf、W. J. Dally和K. Keutzer（2016年）。“SqueezeNet：参数量减少50倍且模型大小小于0.5MB的AlexNet级准确率。”[在线]。可用网址：https://arxiv.org/abs/1602.07360
[12] Y. LeCun，J. S. Denker，和 S. A. Solla，“最佳脑损伤，”*NIPS会议论文集*，1990年，第598-605页。
[13] 韩胜（S. Han）、普尔（J. Pool）、陈杰（J. Tran）与达利（W. Dally）合著的《神经网络权重与连接权重的联合学习：提升网络效率的策略》，载于《*进步神经信息处理系统会议论文集*》2015年，第1135−1143页。
[14] 杨天杰、陈永华和谢文在《*2017年7月CVPR会议论文集*》第6071-6079页发表的《基于能量感知剪枝的高效卷积神经网络设计》一文中提出：
[15] B. Moons，R. Uytterhoeven，W. Dehaene，和 M. Verhelst，“14.5 envision：一个0.26到10 TOPS/W的子词并行动态电压精度频率可扩展卷积神经网络处理器在28nm FDSOI 中，”在*ISSCC Dig. Tech. Papers*，San Francisco，CA，USA，Feb. 2017，pp. 246-247。
[16] S. Yin等人，“一种用于深度学习应用的 1.06-to-5.09 TOPS/W可重构混合神经网络处理器”，*VLSI电路研讨会论文集*，2017年6月，第C26-C27页。
[17] J. Lee，C. Kim，S. Kang，D. Shin，S. Kim，和 H.-J. Yoo，“UNPU：一种50.6 TOPS/W的统一深度神经网络加速器，具有1b到16b的完全可变权重位精度”，*ISSCC Dig. Tech. Papers*，美国加利福尼亚州旧金山，2018年2月，第218-220页。
[18] S. Sharify，A. D. Lascorz，K. Siu，P. Judd，和 A. Moshovos，“Loom：利用权重和激活精度加速卷积神经网络，”*第55届设计自动化年会论文集*，2018年，第20：1−20：6。
[19] P. Judd、J. Albericio、T. Hetherington、T. M. Aamodt与A. Moshovos合著的《条纹：位串行深度神经网络计算》一文，收录于2016年10月*IEEE/ACM第49届微架构国际年会论文集*，第19卷第1-12页。
[20] R. Andri、L. Cavigelli、D. Rossi和L. Benini合著的《YodaNN：基于二进制权重的超低功耗卷积神经网络加速器》，发表于2016年7月的*ISVLSI会议论文集*，第236-241页。
[21] K.Ando等人，《BRein记忆：一个13层4.2K神经元/0.8M突触二进制/三进制可重构65nmCMOS内存中深神经网络加速器》，*VLSI电路研讨会论文集*，2017年6月，第C24-C25页。
[22] Z. Jiang，S. Yin，M. Seok，and J.-S. Seo，“XNOR - SRAM：In-memory computing SRAM macro for binary/ternary deep neural networks，” in *IEEE Symp. VLSI Technol.会议录*，2018年6月，第173-174页。
[23] D. Bankman，L. Yang，B. Moons，M. Verhelst，和B. Murmann，“一个3.8 μ J/86%的 CIFAR -10混合信号二进制CNN处理器，所有内存都在28 nm CMOS芯片上，”在*ISSCC Dig. Tech. Papers*，2018，第222-224页。
[24] H. Valavi、P. J. Ramadge、E. Nestler与N. Verma合著的《一种混合信号二值化卷积神经网络加速器：集成密集权重存储与乘法以减少数据移动》，收录于2018年6月出版的《IEEE超大规模集成电路研讨会论文集》第141-142页。
[25] A. Krizhevsky，I. Sutskever，和G. E. Hinton，“用深度卷积神经网络对ImageNet进行分类”，在*NIPS会议论文集*，2012年，第1097-1105页。
[26] K. Simonyan和A. Zisserman，“用于大规模图像识别的超深度卷积网络”，*CoRR*，2014年9月。
[27] K.何、X.张、S.任和J.孙，《深度残差学习在图像识别中的应用》，收录于*IEEE计算机视觉与模式识别会议论文集*，2016年6月，第770−778页。
[28] C. Szegedy，V. Vanhoucke，S. Ioffe，J. Shlens，和 Z. Wojna，“重新思考计算机视觉的起始架构，”*CVPR会议论文集*，2016年6月，第2818-2826页。
[29] S. Gupta，A. Agrawal，K. Gopalakrishnan，和P. Narayanan，“有限数值精度下的深度学习”，*CoRR*，2015年6月。
[30] Z. Du等人，《ShiDianNao：将视觉处理移近传感器》，*Proc. ISCA* 2015，第92-104页。
[31] Nvidia.（2017年）.*NVDLA 开源项目。[*在线].可获取地址：http://nvdla.org/
[32] N. P. Jouppi等人，“一个张量处理单元在数据中心的性能分析”，*ACM/IEEE第44届国际计算机体系结构年会论文集*，2017年6月，第1-12页。
[33] Y.-H. Chen、T. Krishna、J. S. Emer和V. Sze，《Eyeriss：一种用于深度卷积神经网络的能效可重构加速器》，*IEEE固态电路杂志*，第52卷，第1期，第127-138页，2017年1月。
[34] Y.-H. Chen、T.-J. Yang、J. Emer和V. Sze，“理解现有深度神经网络能效设计方法的局限性”，收录于*SysML会议论文集*，2018年。
[35] H. Noh、S. Hong和B. Han在2015年12月的*IEEE国际计算机视觉会议论文集*中发表了《学习解卷积网络进行语义分割》一文，该论文发表于2015年12月，页码1520-1528。
[36] A.*Dosovitskiy等*，“FlowNet：用卷积网络学习光流”，*IEEE国际计算机视觉会议论文集*，2015年12月，第2758-2766页。
[37] I. Laina，C. Rupprecht，V. Belagiannis，F. Tombari，和 N. Navab，“使用全卷积残差网络进行更深层次的深度预测，”在*第四届国际三维视觉会议（3DV）论文集*，2016年10月，第239-248页。
[38] I.Goodfellow等人，“生成对抗网络”，在*先进神经信息处理系统会议论文集*，2014年，第2672-2680页。
[39] J. Albericio、P. Judd、T. Hetherington、T. Aamodt、N. E. Jerger和A. Moshovos，《Cnvlutin：无无效神经元的深度神经网络计算》，载于*Proc. ISCA* 2016，第1-13页。
[40] S.Zhang等人，“Cambricon-x：稀疏神经网络加速器”，*Proc. ISCA* 2016，文章编号20。
[41] S.Han等人，“EIE：压缩深度神经网络上的高效推理引擎”，*Proc. ISCA*，2016年6月，第243-254页。
[42] A.Parashar等人，“SCNN：压缩稀疏卷积神经网络加速器”，*ACM/IEEE第44届国际计算架构年会（ISCA）论文集*，2017年6月，第27-40页。
[43] Y.-H. Chen、J. Emer和V. Sze在《第43届国际计算架构会议（ISCA）论文集》（2016年，第367-379页）中发表了题为《Eyeriss：一种用于卷积神经网络的节能数据流空间架构》的研究成果。
[44] R. 多兰斯、F. 任和 D. 马尔科维奇ć“FPGA上节能稀疏blas的可扩展稀疏矩阵向量乘法内核”，载于*ISFPGA 会议论文集*，2014年，第161-170页。

Chen Yu-Hsin（S ‘11）于2009年获得台湾大学电气工程学士学位，2013年和2018年分别获得美国麻省理工学院电气工程和计算机科学（EECS）硕士和博士学位。

自2018年起，他担任美国加利福尼亚州圣克拉拉市英伟达架构研究组的研究科学家。他目前的研究重点是为机器学习、深度学习和特定领域处理器设计计算机架构。2018年，他获得了麻省理工学院颁发的金奥孔杰出博士论文奖（电气工程领域）。他曾获得2015年英伟达研究生奖学金、2015年ADI杰出学生设计师奖以及2017年IEEE SSCS 预博士成就奖。他关于CNN加速器数据流的研究在2016年被选为计算机架构领域的顶级推荐之一。此外，他还与人共同在micro-49、ISCA2017和micro-50上教授了关于深度神经网络硬件架构的教程。

杨天儒（S ‘11）先后于2010年和2012年在台湾台北的台湾大学（NTU）获得电气工程学士学位和电子工程硕士学位。目前他正在美国马萨诸塞州剑桥市的麻省理工学院攻读电气工程与计算机科学博士学位，研究方向聚焦于节能型深度神经网络设计。2012年至2015年期间，他曾在台湾新竹市联发科公司智能视觉处理组担任工程师。其研究领域涵盖计算机视觉、机器学习、图像/视频处理及超大规模集成电路系统设计。2011年，他荣获台湾大学创新竞赛一等奖。

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, where he was a member of 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 the exploration of future architectures and modeling and analysis methodologies. He is also a Professor of the Practice with 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, 2015, and 2016. He was the Program Chair of ISCA in 2000 and Micro in 2017.

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

She was a member of the Technical Staff, Systems and Applications Research and Development Center, Texas Instruments (TI), Dallas, TX, USA, where she designed low-power algorithms and architectures for video coding. She also represented TI in the JCT-VC Committee of ITU-T and ISO/IEC standards body during the development of High Efficiency Video Coding (HEVC), which received a Primetime Engineering Emmy Award. 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 currently an Associate Professor with the Electrical Engineering and Computer Science Department, MIT. She has co-edited the book *High Efficiency Video Coding (HEVC): Algorithms and Architectures* (Springer, 2014). Her research interests include energy-aware signal processing algorithms, and low-power circuit and system design for portable multimedia applications, including computer vision, deep learning, autonomous navigation, image processing, and video coding.

Dr. Sze received the Jin-Au Kong Outstanding Doctoral Thesis Prize in electrical engineering from MIT in 2011. She was a recipient of the 2019 Edgerton Faculty Achievement Award from MIT, the 2018 Facebook Faculty Award, the 2017 and 2018 Qualcomm Faculty Award, the 2016 and 2018 Google Faculty Research Award, the 2016 AFOSR Young Investigator Research Program (YIP) Award, the 2016 3M Non-Tenured Faculty Award, the 2014 DARPA Young Faculty Award, and the 2007 DAC/ISSCC Student Design Contest Award. She was a co-recipient of the 2018 VLSI Best Student Paper Award, the 2017 CICC Outstanding Invited Paper Award, the 2016 IEEE Micro Top Picks Award, and the 2008 A-SSCC Outstanding Design Award. She will be the Systems Program Chair of SysML in 2020. She is a Distinguished Lecturer of the IEEE Solid-State Circuits Society (SSCS) and currently serves on the Technical Program Committee for the International Solid-State Circuits Conference (ISSCC) and the SSCS Advisory Committee (AdCom). She has also served on the technical program committees for VLSI Circuits Symposium, Micro and the Conference on Systems and Machine Learning (SysML), and as a Guest Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (TCSVT).

Joel S. Emer（M ‘73-SM’ 03-F ‘04）获得理学学士学位。1974年和1975年分别获得美国印第安纳州西拉斐特普渡大学电气工程学士和硕士学位，1979年获得美国伊利诺伊州香槟厄巴纳-香槟大学电气工程博士学位。

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 小组，并成为康柏和数字设备公司的成员。目前，他是美国马萨诸塞州韦斯特福德市英伟达架构研究小组的高级杰出研究科学家，负责探索未来架构及建模与分析方法。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，讲授计算机架构并指导研究生。他曾担任多个研究与高级开发职位，致力于处理器微架构研究及性能建模与评估技术的开发。他在VAX、Alpha和X86处理器架构方面做出了重要贡献，并被公认为广泛采用的处理器性能评估定量方法的开发者之一。他因在同步多线程技术、处理器可靠性分析、缓存组织以及深度学习空间架构方面的贡献而受到认可。

荣誉博士是ACM的会士。他获得了众多公开认可。2009年，他因在计算机架构领域的终身贡献而获得埃克特-莫奇利奖，2010年和2011年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。他1996年关于同时多线程的论文在2011年获得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 和Micro名人堂。他的五篇论文被选入IEEE Micro的计算机架构顶级推荐，分别是2003年、2004年、2007年、2013年、2015年和2016年。2000年和2017年，他分别担任 ISCA 和Micro的项目主席。

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

她曾任职于美国德克萨斯州达拉斯市德州仪器公司（TI）系统与应用研发中心技术团队，负责视频编码领域的低功耗算法与架构设计。在高效视频编码（HEVC）标准制定期间，她代表TI参与国际电信联盟电信标准化部门（ITU-T）JCT -VC委员会及国际标准化组织/国际电工委员会（ISO/IEC）标准制定工作，该项目最终荣获黄金时段工程艾美奖。在该委员会中，她担任系数扫描与编码核心实验的首席协调员，并主持/联席主持了多个熵编码专项工作组。现任麻省理工学院电气工程与计算机科学系副教授，曾参与编纂*《高效视频编码（HEVC）：算法与架构》*（斯普林格出版社，2014年）一书。其研究领域涵盖节能型信号处理算法，以及面向计算机视觉、深度学习、自主导航、图像处理和视频编码等便携式多媒体应用的低功耗电路与系统设计。

谢博士于2011年获得麻省理工学院颁发的金-奥孔杰出博士论文奖，获奖领域为电气工程。她曾荣获2019年麻省理工学院埃杰顿教职成就奖、2018年脸书教职奖、2017年和2018年高通教职奖、2016年和2018年谷歌教职研究奖、2016年 AFOSR 青年研究者研究计划（YIP）奖、2016年3M非终身教职奖、2014年DARPA青年教职奖以及2007年DAC/ ISSCC 学生设计竞赛奖。2018年，她与他人共同获得VLSI最佳学生论文奖、2017年CICC杰出特邀论文奖、2016年IEEE微系统精选奖和2008年A- SSCC 杰出设计奖。2020年，她将担任SysML系统项目主席。她是IEEE固态电路学会（SSCS）的杰出讲师，目前担任国际固态电路会议（ISSCC）技术项目委员会成员和 SSCS 咨询委员会（AdCom）成员。她还担任过VLSI电路研讨会、Micro和系统与机器学习会议（SysML）的技术程序委员会成员，并担任IEEE电路和视频技术系统交易（TCSVT）的客座编辑。