计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2019.20190099
56(6): 1149-1160, 2019

基于3D忆阻器阵列的神经网络内存计算架构

毛海宇　　舒继武
(清华大学计算机科学与技术系　北京　100084)
(mhy15@mails.tsinghua.edu.cn)

3D Memristor Array Based Neural Network Processing in Memory Architecture

Mao Haiyu and Shu Jiwu
(*Department of Computer Science and Technology*, *Tsinghua University*, *Beijing* 100084)

Abstract　Nowadays, due to the rapid development of artificial intelligence, the memristor-based processing in memory (PIM) architecture for neural network (NN) attracts a lot of researchers' interests since it performs much better than traditional von Neumann architecture. Equipped with the peripheral circuit to support function units, memristor arrays can process a forward propagation with higher parallelism and much less data movement than that in CPU and GPU. However, the hardware of the memristor-based PIM suffers from the large area overhead of peripheral circuit outside the memristor array and non-trivial under-utilization of function units. This paper proposes a 3D memristor array based PIM architecture for NNs (FMC) by gathering the peripheral circuit of function units into a function pool for sharing among memristor arrays that pile up on the pool. We also propose a data mapping scheme for the 3D memristor array based PIM architecture to further increase the utilization of function units and reduce the data transmission among different cubes. The software-hardware co-design for the 3D memristor array based PIM not only makes the most of function units but also shortens the wire interconnections for better high-performance and energy-efficient data transmission. Experiments show that when training a single neural network, our proposed FMC can achieve up to 43.33 times utilization of the function units and can achieve up to 58.51 times utilization of the function units when training multiple neural networks. At the same time, compared with the 2D-PIM which has the same amount of compute array and storage array, FMC only occupies 42.89% area of 2D-PIM. What's more, FMC has 1.5 times speedup and 1.7 times energy saving compared with 2D-PIM.

Key words　3D memristor array; processing in memory (PIM); neural network; peripheral circuit; wire interconnection

摘　要　现如今,由于人工智能的飞速发展,基于忆阻器的神经网络内存计算(processing in memory, PIM)架构吸引了很多研究者的兴趣,因为其性能远优于传统的冯·诺依曼计算机体系结构的性能.配备了支持功能单元的外围电路,忆阻器阵列可以以高并行度以及相比于CPU和GPU更少的数据移动来处理一个前向传播.然而,基于忆阻器的内存计算硬件存在忆阻器的外围电路面积过大以及不容忽视的

收稿日期:2019-02-26 **修回日期**:2019-04-18
基金项目:国家重点研发计划项目(2018YFB1003301);国家自然科学基金项目(61832011)
This work was supported by the National Key Research and Development Program of China (2018YFB1003301) and the National Natural Science Foundation of China (61832011).
通信作者:舒继武(shujw@tsinghua.edu.cn)

功能单元利用率过低的问题.提出了一种基于3D忆阻器阵列的神经网络内存计算架构FMC(function-pool based memristor cube),通过把实现功能单元的外围电路聚集到一起,形成一个功能单元池来供多个堆叠在其上的忆阻器阵列共享.还提出了一种针对基于3D忆阻器阵列的内存计算的数据映射策略,进一步提高功能单元的利用率并减少忆阻器立方体之间的数据传输.这种针对基于3D忆阻器阵列的内存计算的软硬件协同设计不仅充分利用了功能单元,并且缩短了互联电路、提供了高性能且低能耗的数据传输.实验结果表明:在只训练单个神经网络时,提出的FMC能使功能单元的利用率提升43.33倍;在多个神经网络训练任务的情况下,能提升高达58.51倍.同时,和有相同数目的Compute Array及Storage Array的2D-PIM比较,FMC所占空间仅为2D-PIM的42.89%.此外,FMC相比于2D-PIM有平均1.5倍的性能提升,并且有平均1.7倍的能耗节约.

关键词 3D忆阻器阵列;内存计算;神经网络;外围电路;互联线路

中图法分类号 TP391

近年来,基于忆阻器的神经网络内存计算加速器倍受学术研究者和工业界的关注[1-4].研究表明,数据在CPU和片外存储之间的传输消耗的能量比一个浮点运算所消耗的能量高2个数量级[5].基于忆阻器的内存加速器将计算与存储紧密结合,从而省去传统的冯·诺依曼体系结构的中心处理器和内存之间的数据传输,进而提升整体系统的性能并节省大部分的系统能耗[6].此类加速器通过在忆阻器阵列外部加入一些功能单元,使忆阻器阵列能在几乎一个读操作的延迟内完成一次向量乘矩阵操作(matrix-vector-multiplication, MVM)[7],它是神经网络计算中的主要操作.

虽然基于忆阻器的神经网络内存计算加速器有着很高的性能和很低的能耗,但是当它用于神经网络训练任务时,忆阻器阵列的外围电路利用率很低.这是因为:1)当执行前向传播时,用于反向传播的忆阻器阵列的外围功能单元都处于空闲状态; 2)当训练的批大小(*batch_size*)较小时,无论是在前向还是反向的传播过程中,都有一些忆阻器阵列的外围功能单元处于空闲状态.不仅如此,忆阻器阵列的外围功能单元还占据了极大的面积.例如一个8-bit ADC(模拟转数字的原件)的面积就是一个128×128的忆阻器阵列的面积的48倍[2].由于目前的基于忆阻器的神经网络内存计算加速器的外围电路存在上述2个问题,使得整个芯片的面积大且利用率低.

因此,本文针对上述问题提出了一种基于3D忆阻器阵列的神经网络内存计算架构:基于功能单元池的忆阻器立方体(function-pool based memristor cube, FMC)通过提供一个功能单元池给多个堆叠在其上的忆阻器阵列共享,而不是为每一个忆阻器阵列配备所有的功能单元电路,从而达到减小芯片面积并提高功能单元利用率的目的.这种通过3D堆叠的方式进行功能单元共享的结构不仅减小了所有功能单元的占用面积,还极大地缩短了互联线,使得整体的互联面积减小.与此同时,由于互联线路的缩短,FMC还减少了数据的传输,从而使得整体加速器结构的性能提高且能耗降低.

为了更好地利用FMC,本文用软硬件协同设计的方式,进一步提出了基于FMC的计算数据排布策略——功能单元池感知的数据排布(function-pool aware data mapping, FDM).FDM通过配合FMC工作,使得数据移动更少,功能单元的利用率更高,进而提高整体架构的性能.

实验结果表明:在单个训练任务的情况下,我们提出的FMC能使功能单元的利用率提升43.33倍,在多个任务的情况下能提升高达58.51倍.同时,和有相同数目的Compute Array及Storage Array的2D-PIM比较,FMC所占空间仅为2D-PIM的42.89%.而且,FMC相比于2D-PIM有1.5倍的性能提升,且有1.7倍的能耗节约.

本文的主要贡献有3个方面:

1) 分析并发现基于忆阻器的神经网络内存计算加速器的外围电路存在占用面积大且在神经网络的训练过程中外围电路存在利用率极低的问题.

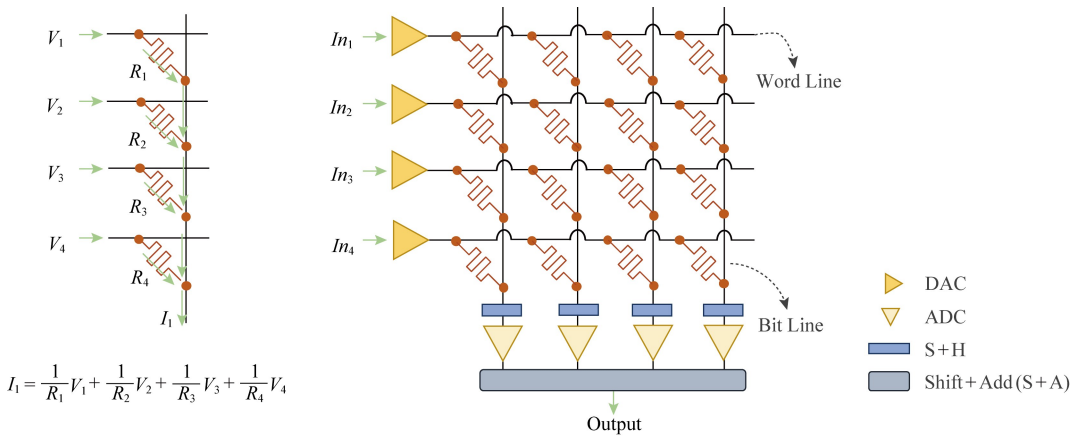
2) 提出了一种基于3D忆阻器阵列的神经网络内存计算架构,通过将忆阻器阵列3D堆叠在根据系统结构配置设计的功能单元池上来共享外围功能电路资源,从而达到减小芯片面积的占用、提高资源利用率的目的.

3) 提出了一种基于3D忆阻器阵列的计算数据排布策略,通过设计基于3D忆阻器阵列的硬件架构的数据排布策略来更好地利用此架构,使得数据移动尽可能少且资源利用率尽可能高.

1 相关工作

由于神经网络的计算(推理和训练)受限于传统的冯•诺依曼体系结构中片上处理器到片外的存储之间有限的带宽,研究者们提出了内存计算,即将计算单元和存储单元相结合,从而避免两者间大量的数据传输.现在的内存计算可分为两大类:基于忆阻器的内存计算(processing in memory, PIM)[1-4,8-10]和基于 DRAM 的内存计算——近数据计算(near data computing, NDC)[11-16].PIM 通过直接利用忆阻器的特性,将存储资源直接用于做计算,取得计算存储相融合的效果.而 NDC 则是通过将计算资源靠近存储单元摆放,并通过一些高带宽的连接使得计算资源能快速地访问存储资源,例如混合记忆立方体(hybrid memory cube, HMC).

图 1 描述了基于忆阻器的 PIM 的基本电路结构和计算原理[2].如图 1(a)所示,每个忆阻器单元的阻值代表一个数值,每个电压代表一个输入值,根据 Kirchoff 定律,通过模拟电路域的电流加,能得到一个代表输出的电流值,如图 1(a)中公式所示.因此,在忆阻器阵列外围加上一些功能单元,如图 1(b)所示,一个 4×4 的忆阻器阵列能够用来存储一个 4×4 的矩阵,然后加上代表 4 个 1×4 的向量的输入电压,就能在几乎一个读延迟内完成一次向量乘矩阵的运算.而向量乘矩阵的运算是神经网络计算中的主要运算,因此,基于忆阻器的 PIM 加速器能极大地提升神经网络计算的性能.



(a) Vector-vector multiplication (b) Vector-matrix multiplication

Fig. 1 Structure of memristor-based PIM array

图 1 基于忆阻器阵列的内存计算硬件结构

NDC 最常用的硬件结构就是 HMC.HMC 通过在逻辑晶粒(计算单元)上堆叠一些 DRAM 晶粒(存储单元),然后用一些垂直贯穿 DRAM 的穿过硅片通道(through silicon vias, TSV)使它们相连,从而使得计算单元到存储单元拥有很高的访存带宽.但是其本质上还是计算和存储相分离的,不同于基于忆阻器的 PIM,NDC 中用于存储的 DRAM 晶粒并不能直接用来做计算.因此,PIM 的性能通常优于 NDC.

目前对于 PIM 的研究更多地针对推理,因为推理的计算以及数据流要比训练简单得多,但是它们所用到的硬件功能单元基本一样.文献[1]提出了用具有高存储密度、低读写延迟、相比于其余 NVM 有较长寿命的 ReRAM[17-20]来作为 PIM 的基础硬件单元,并设计 ReRAM 阵列可被配置为 3 种模式:计算、存储和缓存,再通过加上支持其余神经网络计算的硬件单元来做神经网络的推理计算.文献[2]通过使用 ReRAM 阵列做向量乘矩阵的计算单元,并利用 eDRAM 作为其存储单元,再通过设计推理的流水线计算,从而提高用 PIM 来做神经网络推理的效率.文献[3]通过复制多份计算单元,减少当 PIM 用来做神经网络训练时流水线计算中的空闲,从而提升 PIM 用来做神经网络训练任务的效率.文献[4]使用比 ReRAM 有更接近于线性电阻值更新的 PCM[21-23]加上 CMOS 作为 PIM 的基础硬件单元[24],在同一个阵列中完成神经网络训练时的前向传播和反向传播操作.但是文献[4]只能支持仅有全连接层的神经网络的训练,不能够支持卷积神经网络的训练,而文献[3]可以支持.文献[7]提出了一种用来支持生成对抗网络的基于 ReRAM 的 PIM,它通过软硬件结合的设计,省去生成对抗网络中的冗余计算和存储,并通过运行时可重配的互联提升

PIM 支持复杂训练数据流时的性能.

然而目前针对 PIM 的研究都以提高 PIM 的计算效率为目的,并没有关注 PIM 的芯片面积以及其中的资源利用率.本文考虑了 PIM 芯片中的功能单元的面积占用问题及其利用率问题.还需说明的是:本文提出的基于忆阻器的 3D 结构不同于 HMC 的结构,其存储本身能够用来做计算,且最下层的逻辑晶粒(本文中的资源池)不仅仅是计算资源,还有用来支撑忆阻器做计算的功能单元(例如图 1 中的 DAC 和 ADC).

下文首先介绍设计的 FMC 架构,然后介绍基于 FMC 的数据映射,再后给出 FMC 的功能单元资源利用率、空间占用情况和 FMC+FDM 的性能和能耗等实验结果及相应的分析,最后给出本文总结.

2 FMC 架构设计

在本节中,我们首先给出了 FMC 的架构概要图,如图 2 所示,然后我们分别介绍 FMC 的忆阻器阵列、互联结构、共享资源池.

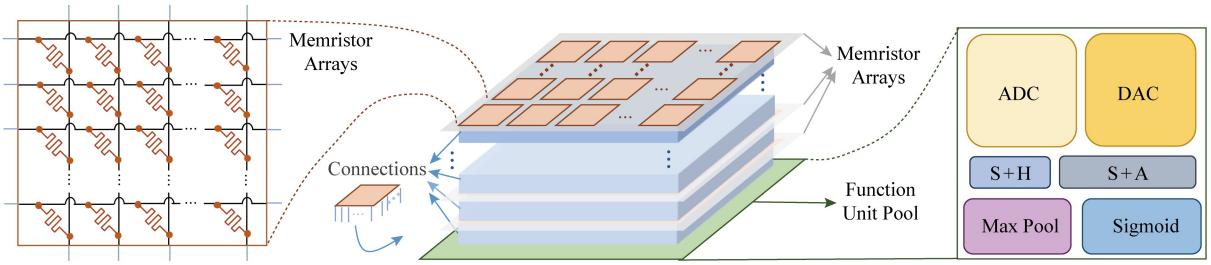


Fig. 2 Outline of the function-pool based memristor cube

图 2 基于功能池的忆阻器立方体的概要图

2.1 FMC 架构

FMC 架构主要分成 3 个部分:忆阻器阵列、3D 互联结构和共享资源池(如图 2 的中部所示).共享功能单元池放置于最下方,其上堆叠若干个忆阻器阵列晶粒层.每一层有若干个忆阻器阵列晶粒,一部分用来做普通的存储(本文中称为 Storage Array),一部分既可以通过配置用来做存储也可以设置其用来做计算(本文中称为 Compute Array).每一层均通过垂直的连接和共享资源层相连.其层数以及每层忆阻器阵列的数目受限于工艺大小以及资源池的资源多少.

2.2 FMC 的忆阻器阵列

FMC 使用 cross-bar 结构的 ReRAM 作为其基础硬件单元,如图 2 左部所示,保留了作为存储器时的外围电路单元,例如写驱动(未在图 2 中标出).每一个正方形表示一个 ReRAM cell.当这个忆阻器阵列被配置成 Compute Array 时,每个 ReRAM cell 用来存储一个神经网络中权重的值或者存储值的一部分(即用多个 cell 存储一个值,每个 cell 存储多个位);当这个忆阻器阵列被配置成 Storage Array 时,整个阵列就被当做普通存储器使用,一个 cell 存储一个位.

2.3 FMC 的互联结构

FMC 中的每层忆阻器阵列 Storage Array 中的忆阻器阵列通过 H-tree 的方式进行连接,Compute Array 中的忆阻器阵列通过基于 H-tree 的可重配的连接方式进行连接(与文献[7]的平面连接方式相同).Storage Array 和 Compute Array 之间通过一个高速的共享线路进行连接.

图 3 给出了忆阻器阵列的平面互联结构.左边是 Storage Array 的互联结构.和普通存储一样,Storage Array 用 H-tree 的连接方式:灰色圆圈代表

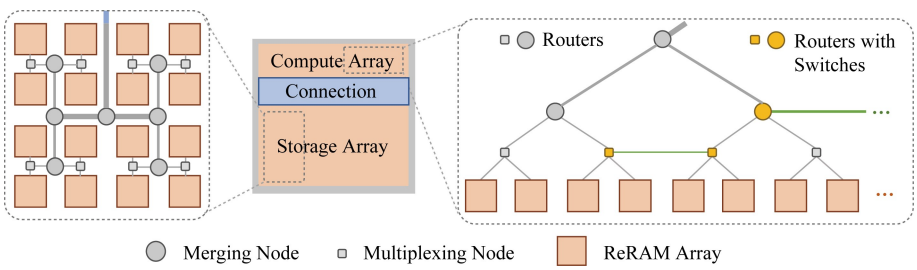


Fig. 3 Intra-connections of 2D memristors

图 3 忆阻器阵列的平面互联

merging node,连接的父节点的线路宽是连接子节点线路宽的 2 倍;灰色方块代表 multiplexing node,连接的父节点的线路宽和连接子节点线路宽一样.右边是 Compute Array 的连接方式:我们保留原来的 H-tree 的连接方式(图 3 中灰色线条),并在同一层连接节点中不共父节点的节点(如图 3 中黄色节点所示)之间加上一条电路线(如图 3 中绿色线条所示).新加的这条电路线的宽度与其连接其父节点的线路宽度相同.由于接口有限,图 3 中每个黄色节点被增加了一个转换接口,使得它们能选择连到它们的父节点或者相邻节点.这种能够在网络训练时的动态配置的连接方式不仅能够支持快速的 Compute Array 之间的数据传输(开关拨到横向连接线上,跨过 H-tree 结构进行通信),还能支持快速的权重更新,即 Compute Array 的读写(开关配置成 H-tree 的连接方式).图 3 中间蓝色部分表示的是 Compute Array 和 Storage Array 之间的共享高速连接,这部分并不是将它们直接相连,而是经由共享资源池将它们相连(如图 2 中蓝色部分所示).

Compute Array 中每一个忆阻器阵列(橘黄色方块)都配备有一个转换接口,能控制每个忆阻器阵列是垂直连接(连向共享资源池或者不同平面的忆阻器阵列)还是平面连接(连向同平面的忆阻器阵列).如果一个忆阻器阵列连接到下一个忆阻器阵列,那么我们称这 2 个忆阻器阵列是在同一个模拟电路域的,反之则不是.我们把一个模拟域内的所有连向共享资源池的阵列连接叫做这个模拟域的出入口,入口连接共享资源池的 DAC 分池,出口连接 S+H 池.

当有多个 FMC 时,各个 FMC 之间通过 C-mesh 的连接方式进行互联,每个 FMC 的共享资源层能够访问其他 FMC 的 Storage Array,但是不能使用其他 FMC 的 Compute Array,即 FMC 之间只在数字域进行通信而不用模拟信号通信,这样确保了模拟信号的稳定性和计算的准确性.

2.4 FMC 的共享资源池

共享资源池由六大部分组成(如图 2 右部所示):DAC 池、S+H 池、ADC 池、S+A 池、MP(Max Pool)池和激活函数单元池.每个分池由若干个功能单元组成.表 1 给出了这些功能单元的名称及其对应的释义.

图 4 给出了各个分池之间的互联结构.DAC 池连接模拟电路域的入口,S+H 池连接其出口.S+H 池和 ADC 池相连接,ADC 池可选择连接 S+A 池、MP 池、激活函数单元池和 Storage Array 部分.S+A 池可选择连接 MP 池、激活函数单元池、DAC 池和 Storage Array 部分.激活函数单元池可选择连接 MP 池、DAC 池和 Storage Array 部分.MP 池可选择连接 DAC 池和 Storage Array 部分.每个分池内的计算资源通过 C-mesh 进行连接,分池之间通过高速共享线路进行连接.

Table 1 Notations of Function Units

表 1 功能单元的参数表示

Function Unit	Description
DAC	Digital to Analog
ADC	Analog to Digital
S+H	Sample and Hold
S+A	Shift and Add
MP	Max Pool Unit
Sigmoid	Sigmoid Unit

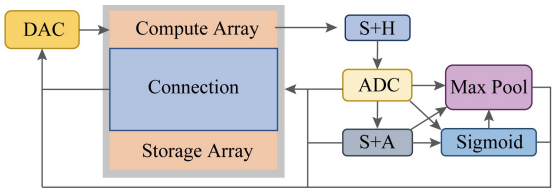


Fig. 4 Intra-connection of sub-pools in function unit pool

图 4 共享资源池各个分池之间的互联结构

表 2 提供了共享资源池中功能单元的各个配置参数表示,我们将使用这些配置参数来介绍共享资源池的具体设计方法.

Table 2 Configurations of Function Units

表 2 功能单元的配置参数

Function Unit	Area/mm^2	Number	Configuration
DAC	A_{DAC}	N_{DAC}	x-bit
ADC	A_{ADC}	N_{ADC}	y-bit, f GSps
S+H	A_{SH}	N_{SH}	
S+A	A_{SA}	N_{SA}	
MP	A_{MP}	N_{MP}	
Sigmoid	A_S	N_S	
Memristor Array	A_{MA}	$N_{storage}$	n-bit/cell
		$N_{compute}$	$m\times m$ cell/array
Hyper Tr			b GBps

表 2 是各个单元的相关配置参数表示,FMC 的功能单元池的面积 A_{FUP} 可表示为

$$A_{FUP}=A_{DAC}N_{DAC}+A_{ADC}N_{ADC}+A_{SH}N_{SH}+A_{SA}N_{SA}+A_{MP}N_{MP}+A_SN_S,$$

即功能单元池的总面积等于各个功能分池的面积和,各个分池的面积又取决于单个功能单元的面积以及其个数.为了节省 FMC 总体占用空间,其功能单元池的面积 A_{FUP} 需满足:

$$0\leqslant\frac{A_{\mathrm{FUP}}-A_{\mathrm{MA}}(N_{\mathrm{storage}}+N_{\mathrm{compute}})}{A_{\mathrm{FUP}}}\leqslant\theta,$$

其中,$0\leqslant\theta\leqslant0.1$.即功能单元池的面积既不能小于其上堆叠的 ReRAM 阵列的面积,否则无法充分利用功能单元池的空间;也不能过分大于 ReRAM 阵列的面积,否则会导致 3D 堆叠时的连接线路过长,不高效.同时,为了使得模拟电路域到数字电路域的顺畅转换,ADC 和 DAC 单元的参数需满足:

$$x\times N_{\mathrm{DAC}}=n\times m \tag{1}$$

$$N_{\mathrm{SubCom}}\times\frac{m\times n}{latency_{\mathrm{PIM}}}\times10^{9}\leqslant f\times y\times N_{\mathrm{DAC}}\leqslant8b, \tag{2}$$

其中,$latency_{\mathrm{PIM}}$ 是每个忆阻器阵列一次模拟域向量乘矩阵运算的延迟,单位是 ns;N_{SubCom} 是在一个共享线路上的所有用于计算工作的忆阻器阵列的数目.式(1)主要使得 DAC 转换的位数和一个阵列中一行的位数相等,避免计算延迟.式(2)主要使得 ADC 转换的位数不能小于计算速度,防止计算出来的数据阻塞在 ADC 之前;且 ADC 转换速率不能大于传输速度,否则算出的数据会被阻塞在 ADC 之后.

有了这些公式的限制,我们就能根据实际系统情况的限制以及任务的需求,很好地制定出我们所需的功能单元池中每个功能单元的数目,从而设计高效的功能单元池.

3 FDM 策略设计

在本节我们介绍 FDM 策略,配合 FMC 的硬件结构,使得存储与计算资源分配合理、数据传输尽可能少、共享资源池的资源利用率尽可能高.由于 FMC 中有多层 PIM 阵列层,每层只有有限个 PIM 阵列,所以在神经网络的训练过程中可能需要多个 PIM 层的多个阵列.每一层中 PIM 阵列的数据传输要比每层间的数据传输延迟小,并且神经网络的每个层之内、每个层之间已经训练的不同阶段都会有数据依赖.因此 FDM 的核心思想是将有数据依赖的数据块尽量存在一个 PIM 阵列层上,将没有数据依赖的计算块放置于不同 PIM 层以方便它们共享资源池中的资源.图 5 给出了在 PIM 中训练一个神经网络的数据存储和数据流图.正方形表示数据存储在 Compute Array 中,直接用来做计算;圆角矩形表示数据存储在 Storage Array 中,用来暂存数据供给 Compute Array 做计算.图 5 中的 W 表示 Weight,E 表示 Error,O 表示 Output.训练时的前向传播和反向传播是串行的,而反向传播包含 2 部分:误差传播和权值计算,二者是并行的.因此,当只有一个训练任务时(训练一个网络),我们尽量把串行的部分切割排布到同一个 FMC 中,使得它们能够时分复用共享资源池;把并行的部分排布到不同的 FMC 中,避免它们对资源池的访问造成冲突.

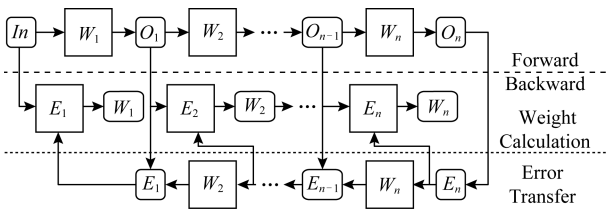


Fig. 5 Data graph of training a neural network

图 5 训练一个神经网络的数据图

图 6 给出了 FDM 策略的一个例子:用 2 个资源共享池相连的 FMC 来训练一个神经网络时的数据排布情况.我们先将整个网络的训练按照时间序列把每层使用的权值矩阵排列,如图 6 中橘黄色方块所示.然后我们将所有的权值矩阵两等分(如图 6 中虚线框所示),依据的准则是尽可能地使需要并行的权值矩阵阵列分配到不同的 FMC 上,同时保证相邻的神经网络的权值矩阵尽量被分配到 FMC 上相邻的位置.如果我们计划使用 m 个 FMC 来训练

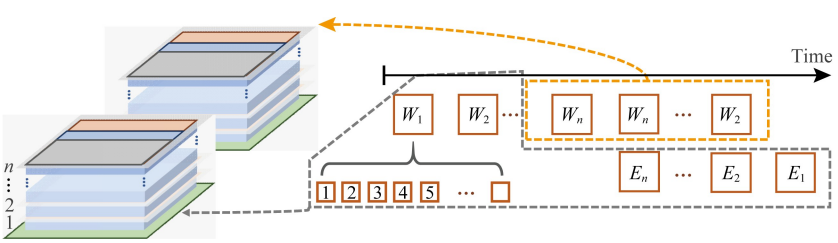


Fig. 6 An example of FDM

图 6 FDM 策略的一个示例

一个神经网络(共需要 n 个 Compute Array 来存储它的权值),$p=\lceil n/m \rceil$,那么我们依次从$\{W'_2,W'_3,\cdots,W'_n\}$,$\{W_n,W_{n-1},\cdots,W_1\}$,$\{E'_n,E'_{n-1},\cdots,E'_1\}$这 3 个序列中每次取 p 个权值矩阵存到相应的 FMC 中.

如果一层的权值需要多个忆阻器阵列进行存储,那么我们把它们均匀地分布到一个 FMC 中的所有忆阻器阵列层上(图 6 中从第 1 层到第 n 层进行循环分配,直到分配结束),因为它们之间不需要进行通信,所以不必在同一个平面层中,其余各层按照这个方式进行分配.这样做还为了使得所有神经网络层能有几乎一样的计算效率,能够有几乎相等的访问共享资源池的延迟(避免如将第 1 层神经网络权值分配到 FMC 的第 1 层忆阻器阵列、最后一层神经网络分配到 FMC 的最后一层忆阻器阵列,从而导致因最后一层访问共享资源池的延迟大而造成性能差的问题).当有多个训练任务时,我们将多个训练任务均匀分配到所有的 FMC 中.

FDM 的整体流程如图 7 所示.首先将多个网络分配到多个 FMC 中,每个网络可能在一个或多个 FMC 里;然后将每个网络的不同层部分分配到 FMC 中;最后把网络每个层的各个部分分配到 FMC 的各个 PIM 阵列层中.

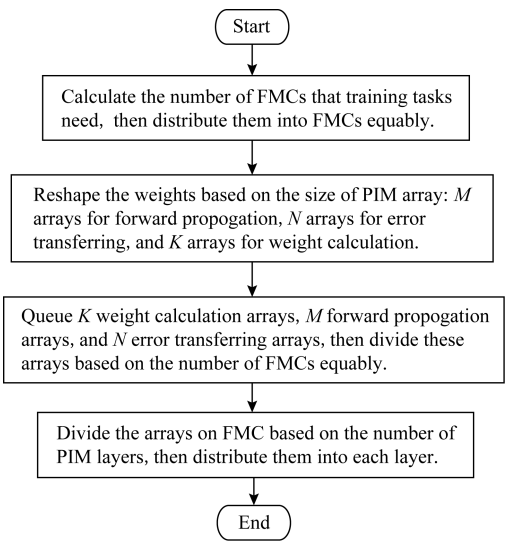


Fig. 7 Flow chart of FMC

图 7 FMC 的流程总体流程

4 实验与分析

本节首先介绍实验环境、参数配置以及实验方法,并简要介绍所用的对比实验的结构和参数配置;接着给出实验结果和相应的分析,包括 FMC 的功能单元资源利用率、空间占用情况和 FMC+FDM 在训练各个神经网络时的性能和能耗比较.

4.1 实验方法

表 3 给出了实验环境和具体的基于 ReRAM 的主存参数配置.整个系统将 CPU 作为中心任务处理器,统计好训练任务的数量以及每个任务所占空间大小后,按照 FMD 的策略将训练任务发布到整个 PIM 中.发布完任务后 CPU 不再参与整个训练工作,所有的训练工作包括计算和数据存取都在基于 ReRAM 的主存中进行.本系统不考虑 DRAM 和 ReRAM 的混合主存,所有用来支持训练的数据都存放在基于 ReRAM 的主存中.

Table 3 Hardware Configurations

表 3 硬件配置

Hardware		Configurations
Host Processor		Intel Xeon CPU E5520, 2.27 GHz, 4 cores
L1 I/D Cache		32 KB/32 KB, 4-way, 2 cycles access
L2 Cache		256 KB, 8-way, 10 cycles access
ReRAM-based Main Memory	Overview	TaO_x/TiO_2-based ReRAM, 128 GB, 64 MB per FMC
	Bank	32.8 ns/41.4 ns read/write latency, 413 pJ/665 pJ read/write energy
	H-Tree	29.9 ns latency, 386 pJ energy
	Tile	2.9 ns/11.5 ns read/write latency, 3.3 pJ/34.8 pJ read/write energy
I/O Frequency		1.6 GHz

我们使用 CACTI 6.5[25]对所有 FMC 中的连接进行建模;使用 CACTI-IO[26]来对 FMC 之间的连接进行建模.关于 ReRAM 的参数配置,我们使用 DESTINY[27]进行模拟获取.需要说明的是,我们的模拟系统中使用的基于忆阻器的主存包含 2 个 chip;每个 chip 包含 16 个 tile,通过 H-tree 方式连接;每个 tile 包含 16 个 FMC.

如 2.4 节所述,FMC 的资源池里各个资源的数目在不同的系统限制下会有不同的取值;同时,在同一个系统配置下,FMC 的资源池的配置也可能会有多种可能.该实验部分测试了在表 3 的系统配置情况下的所有配置可能,并选取了 FMC 的资源池的资源利用率最高的一种配置来做性能和能耗的相关实验.

表 4 给出了 FMC 的单元配置参数,其中忆阻器阵列给出的参数是一层的配置,每个 FMC 有 4 层忆阻器阵列(考虑到传输线路长度和共享资源池的资源利用率).共享资源池中每个单元的占用面积直接

使用 ISAAC 中的配置[2]；每个单元的数目根据 2.4 节中的限制条件给出.

Table 4 Parameters of Units in FMC

表 4 FMC 单元的实验配置参数

Unit	Area/mm²	Number	Configuration
DAC	0.000 17	1 024	1-bit
ADC	0.009 60	16	8-bit, 1.2 GBps
S+H	0.000 04	2 048	
S+A	0.000 06	4	
MP	0.000 24	1	
Sigmoid	0.000 60	1	
Storage Array	0.000 20	1 920	4-bit/cell, 128×128
Computer Array	0.000 20	128	4-bit/cell, 128×128
Hyper Tr			6.4 GBps

我们将配备如表 4 所示的 FMC 结构和 2D 的 PIM 进行实验比较，用 PipeLayer[3] 的结构.即一个 PIM 由多个 PIM 阵列组成，它们之间采用 H-tree 的连接方式进行连接，并且每个 PIM 阵列周围都配备有表 4 中的功能单元，这些功能单元之间互相不能共享，属于每个阵列私有.我们使用 LeNet[28]，ConvNet[29] 和 Caffe Model Zoo[30] 中的 6 个流行的网络来作为实验的测试网络，包含：AlexNet，NiN，GoogLeNet，VGG_M，VGG_S 和 VGG_19.它们中既有大网络也有小网络，既有全连接层也有卷积层，数据集也涵盖了用来做图像分类的黑白 MNIST 手写数字集[31] 和著名的用来做图像分类、目标定位和检测、场景分类的彩色图片集 ImageNet[32].我们使用 1，8，16，32，64 和 128 六种 *batch_size* 来训练测试网络集，每个单独的网络分别执行 1 000 次训练迭代(用来测试单个任务时的架构性能)，并将所有网络同时执行 1 000 次训练迭代(用来测试多个任务时的架构性能).

4.2 FMC 的功能单元资源利用率

我们首先将 2D 的 PIM 的外围电路资源利用率和 FMC 的共享资源池中的功能单元利用率在训练单个神经网络的情况下进行比较，结果如图 8 所示.我们将训练 8 个测试网络时 2D-PIM 的外围电路资源利用率归一化成 1，给出在每个测试网络下 FMC 的资源利用率与 2D-PIM 的倍数比较.图 8 中 FMC-x 的 x 表示训练时所用的 *batch_size*，所有网络测试结果呈现按照名称首字母降序排序.

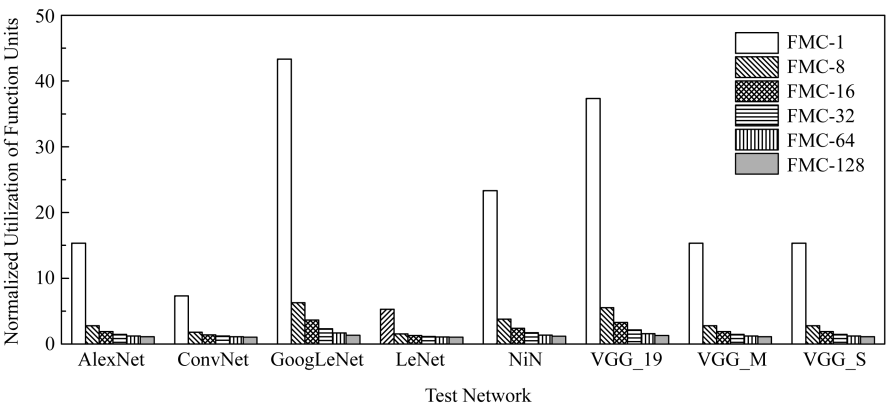


Fig. 8 Utilization of function units in FMC compared with 2D-PIM when training a single network

图 8 单个网络训练时 FMC 中功能单元的资源利用率和 2D-PIM 的比较

我们可以从图 8 中看到，FMC 的资源利用率要比 2D-PIM 的高(所有的资源利用率的提升均在 1 倍以上)，且在 *batch_size*=1 时有非常显著的利用率提升(如图 8 中 FMC-1 的柱形所示)，最高能达到 43.33 倍(训练 GoogLeNet 时).另外 FMC 的资源利用率随 *batch_size* 增大而减小.这是因为当 *batch_size* 增大时，2D-PIM 中整个传播过程可以很好地利用管道并行(pipeline)起来，从而提高功能单元的资源利用率.因此 FMC 在 *batch_size* 较大时，资源利用率的提升相比于 2D-PIM 不是很明显，但仍然有一定提升.例如用 *batch_size*=128 来训练 GoogLeNet 时，资源利用率有 1.33 倍的提升.这样的提升来源于 pipeline 时候的气泡部分以及前向传播和反向传播的中断部分.图 8 还显示出训练不同深度的网络时 FMC 资源利用率提升的差别：当训练深度大的网络(GoogLeNet，VGG_19)时，FMC 的资源率提升相对大；当训练深度小的网络(LeNet，ConvNet)时，FMC 的资源利用率提升较小.这是因为网络的深度越大，处理后面的网络层需要等待的时间就越长，进而导致处理后面网络层的

2D-PIM 的功能单元空闲时间过长,最终使得 2D-PIM 的资源利用率低,而 FMC 通过使前后层共享功能单元的方式提高了资源利用率,因此 FMC 的资源利用率相比较于 2D-PIM 的提升会随着网络层数的增大而更高,反之亦然.

图 9 展示了多个神经网络同时训练时在 FMC 中的资源利用率和在 2D-PIM 中的资源利用率的比较结果.其中,L 代表 LeNet,C 代表 ConvNet,G 代表 GoogLeNet,V19 代表 VGG_19,ALL 代表所有 8 个测试网络的组合.

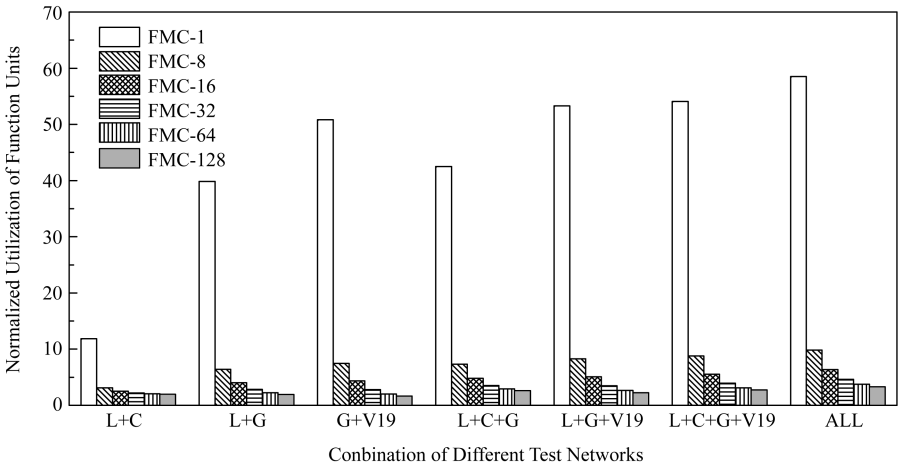


Fig. 9 Utilization of function units in FMC compared with 2D-PIM when training multiple networks

图 9 多个网络训练时 FMC 中功能单元的资源利用率和 2D-PIM 的比较

从图 9 中我们可以看出,当同时训练的网络越多且网络深度越大时,FMC 相比较于 2D-PIM 的资源利用率提升更大.这是因为,当同时训练多个网络时,2D-PIM 的资源利用率取决于最小网络训练时的资源利用率,而 FMC 因为能够使得多个网络共享功能单元,从而在训练网络任务多时,由于各个网络之间不存在依赖关系可以并行,因此它们可以通过 pipeline 的方式共享功能单元,从而提升 FMC 的资源利用率,进而使得 FMC 相比较于 2D-PIM 的资源利用率的提升更为明显.但是这样的提升速率并不会随着网络大小和网络深度的增长速率而增长,这是因为,当网络的大小和数目大到一定程度时,共享资源池的资源利用率趋近于 100%(不能到达 100%,因为训练起步时会有一定的气泡).

4.3 FMC 空间占用

图 10 给出了 FMC 中的各个功能单元和一层忆阻器的面积占用的比较.其中,一层忆阻器的面积占用为总共面积占用的 49.97%,也就意味着一层忆阻器的面积稍稍小于功能单元池的面积.在功能单元池中,DAC 和 ADC 是占主要面积的分单元池,分别为 21.24%和 18.74%.

FMC 中新加的开关和连接线占总空间的 6.79%,而一个 FMC 所占空间仅是具有同等数目的 Compute Array 及 Storage Array 的 2D-PIM 所占空间的 42.89%.

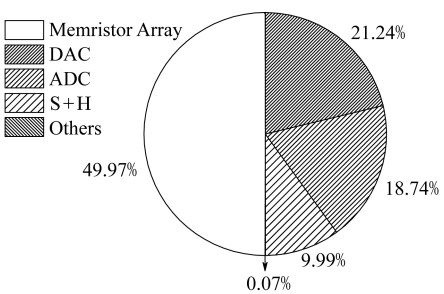


Fig. 10 Area breakdown of FMC(one layer of memristor)

图 10 FMC 中各个单元的空间占用比率(包含一层忆阻器阵列)

4.4 FMC+FDM 性能和能耗

由于 *batch_size* 对 FMC 的性能和能耗与 2D-PIM 的比较几乎没有影响,因此我们选取 *batch_size*=64(常用的 *batch_size*)作为 FMC 的性能和能耗测试结果呈现,具体如图 11 和图 12 所示.

从图 11 中我们可以看出,相比于 2D-PIM,使用 FMC 训练越大的神经网络取得的性能加速要大.这是因为 FMC 各层忆阻器阵列之间是 3D 堆叠的,并通过 3D 堆叠的方式共享资源池.这使得忆阻器阵列之间的连接充分缩短,而网络越大需要的忆阻器阵列就越多,因此 FMC 相比于 2D-PIM 的优势就更为明显.而用 FMC 训练 LeNet 时几乎没有性能上的提升,这是因为 LeNet 很小,从而使得 FMC 的 3D

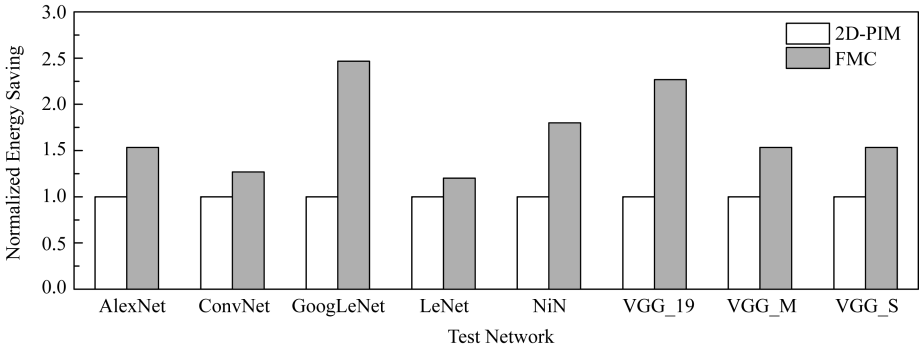


Fig. 11 Performance of FMC compared with 2D-PIM

图 11 FMC 与 2D-PIM 的性能比较

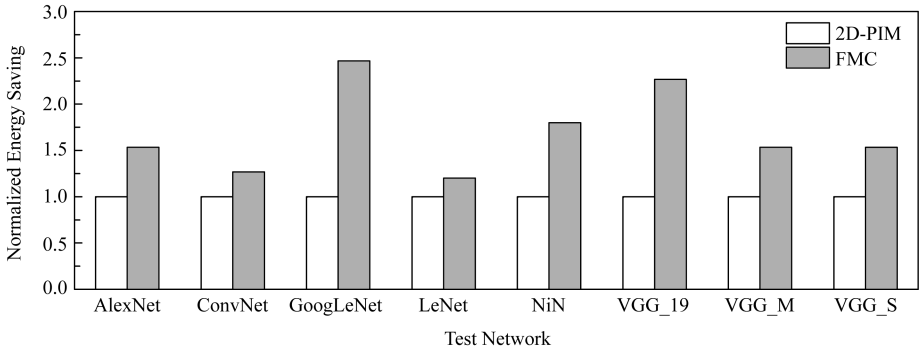


Fig. 12 Energy consumption of FMC compared with 2D-PIM

图 12 FMC 与 2D-PIM 的能耗比较

堆叠的连接方式相比较于 2D-PIM 的优势小,且数据传输的延迟被计算的延迟隐藏.总体来看,FMC 相比于 2D-PIM 有 1.5 倍的性能提升.

图 12 展示了用 FMC 和 2D-PIM 训练单个神经网络时能耗比较.FMC 在训练神经网络时能比 2D-PIM 有明显的能耗节省,也是因为 FMC 的 3D 堆叠的结构比 2D-PIM 的平面结构大大减少数据传输的线路长度.因此,所训练的神经网络越大,FMC 相比于 2D-PIM 的能耗节省就更加明显.当训练 GoogLeNet 时,能耗节省能达到 2.47 倍.而当训练 LeNet 时,FMC 也比 2D-PIM 有 1.2 倍的能耗节省.总的来看,FMC 平均比 2D-PIM 节省 1.7 倍的能耗.

5 总 结

现如今,基于忆阻器的内存计算(PIM)如火如荼,但是它存在着除忆阻器阵列之外的电路单元面积过大且利用率低的问题.本文提出了一种基于 3D 忆阻器阵列的神经网络内存计算架构,将功能单元抽取出来形成一个资源池提供给忆阻器阵列共享,并通过 3D 堆叠的方式缩短各个忆阻器阵列的连接以及忆阻器阵列和功能单元池之间的连接.同时,我们还提出了一种基于 3D 忆阻器阵列的计算数据排布策略,配合上 3D 忆阻器阵列的结构,使得训练神经网络时的数据移动尽可能小.实验结果显示,我们提出的基于 3D 忆阻器阵列加共享资源池的架构能使功能单元的利用率在单个训练任务的情况下提升 43.33 倍,在多个任务的情况下最高提升 58.51 倍.同时,我们提出 3D 架构所占空间是有相同数目的 Compute Array 及 Storage Array 的 2D-PIM 所占空间的 42.89%.此外,我们提出 3D 架构相比于 2D-PIM 有平均 1.5 倍的性能提升,且有平均 1.7 倍的能耗节约.

参 考 文 献

[1] Chi Ping, Li Shuangchen, Xu Cong, et al. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory [C] //Proc of 2016 ACM/IEEE the 43rd Annual Int Symp on Computer Architecture (ISCA). Piscataway, NJ: IEEE, 2016: 27-39

[2] Shafiee A, Nag A, Muralimanohar N, et al. ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars [J]. ACM SIGARCH Computer Architecture News, 2016, 44(3): 14-26

[3] Song Linghao, Qian Xuehai, Li Hai, et al. PipeLayer: A pipelined ReRAM-based accelerator for deep learning [C] // Proc of 2017 IEEE Int Symp on High Performance Computer Architecture (HPCA). Piscataway, NJ: IEEE, 2017: 541-552

[4] Stefano A, Pritish N, Hsinyu T, et al. Equivalent-accuracy accelerated neural-network training using analogue memory [J]. Nature, 2018, 558(7708): 60-67

[5] Keckler S W, Dally W J, Khailany B, et al. Gpus and the future of parallel computing [J]. IEEE Micro, 2011, 31(5): 7-17

[6] Li Shuangchen, Xu Cong, Zou Qiaosha, et al. Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories [C] //Proc of Design Automation Conf. Piscataway, NJ: IEEE, 2016: 173-179

[7] Mao Haiyu, Song Mingcong, Li Tao, et al. Lergan: A zero-free, low data movement and pim-based gan architecture [C] //Proc of the 51st Annual IEEE/ACM Int Symp on Microarchitecture (MICRO). Los Alamitos, CA: IEEE Computer Society, 2018: 669-681

[8] Wenqin Huangfu, Li Shuangchen, Hu Xing, et al. Radar: A 3D-ReRAM based DNA alignment accelerator architecture [C] //Proc of 2018 the 55th ACM/ESDA/IEEE Design Automation Conf (DAC). Piscataway, NJ: IEEE, 2018: 1-6

[9] Cheng Ming, Xia Lixue, Zhu Zhenhua, et al. Time: A training-in-memory architecture for memristor-based deep neural networks [C] //Proc of 2017 the 54th IEEE Design Automation Conf (DAC). New York: ACM, 2017: 26-31

[10] Bojnordi M N. Memristive Boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning [C] //Proc of the 22nd IEEE Int Symp on High Performance Computer Architecture. Los Alamitos, CA: IEEE Computer Society, 2016: 1-13

[11] Kim D, Kung J, Chai S, et al. Neurocube: A programmable digital neuromorphic architecture with high-density 3D memory [C] //Proc of the 43rd Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2016: 380-392

[12] Gao Mingyu, Kozyrakis C. Hrl: Efficient and flexible reconfigurable logic for near-data processing [C] //Proc of the 22nd IEEE Int Symp on High Performance Computer Architecture. Los Alamitos, CA: IEEE Computer Society, 2016: 126-137

[13] Kim H, Kim H, Yalamanchili S, et al. Understanding energy aspects of processing-near-memory for HPC workloads [C] //Proc of the 2015 Int Symp on Memory Systems. New York: ACM: 276-282

[14] Farmahini-Farahani A, Ahn J H, Morrow K, et al. Drama: An architecture for accelerated processing near memory [J]. IEEE Computer Architecture Letters, 2017, 14(1): 26-29

[15] Ahn J, Yoo S, Mutlu O, et al. Pim-enabled instructions: A low-overhead, locality-aware processing-in-memory architecture [J]. ACM SIGARCH Computer Architecture News, 2015, 43(3): 336-348

[16] Chen Yunji, Luo Tao, Liu Shaoli, et al. Dadiannao: A machine-learning supercomputer [C] //Proc of the 47th Annual IEEE/ACM Int Symp on Microarchitecture (MICRO). Los Alamitos, CA: IEEE Computer Society, 2014: 609-622

[17] Liu T Y, Yan T H, Scheuerlein R, et al. A 130.7 mm^2, 2-layer 32-GB ReRAM memory device in 24-nm technology [C] //Proc of IEEE Int Solid-State Circuits Conf (Digest of Technical Papers). Piscataway, NJ: IEEE, 2013: 140-153

[18] Akinaga H, Shima H. Resistive random access memory (ReRAM) based on metal oxides [J]. Proceedings of the IEEE, 2010, 98(12): 2237-2251

[19] Wei Z, Kanzawa Y, Arita K, et al. Highly reliable taox ReRAM and direct evidence of redox reaction mechanism [C] //Proc of IEEE Int Electron Devices Meeting. Piscataway, NJ: IEEE, 2008: 1-4

[20] Liu Qi, Sun Jun, Lü Hangbing, et al. Real-time observation on dynamic growth/dissolution of conductive filaments in oxide-electrolyte-based ReRAM [J]. Advanced Materials, 2012, 24(14): 1844-1849

[21] Burr G W, Shelby R M, Sidler S, et al. Experimental demonstration and tolerancing of a large-scale neural network (165000synapses) using phase-change memory as the synaptic weight element [J]. IEEE Transactions on Electron Devices, 62(11): 3498-3507

[22] Burgt Y, Lubberman E, Fuller E, et al. A non-volatile organic electrochemical device as a low-voltage artificial synapse for neuromorphic computing [J]. Nature Materials, 2017, 16(4): 414-418

[23] Agarwal S, Gedrim R B J, Hsia A H, et al. Achieving ideal accuracies in analog neuromorphic computing using periodic carry [C] //Proc of Symp on VLSI Technology. Piscataway, NJ: IEEE, 2017: 174-175

[24] Narayanan P, Fumarola A, Sanches L L, et al. Toward on-chip acceleration of the backpropagation algorithm using nonvolatile memory [J]. IBM Journal of Research and Development, 2017, 61(4): 1-11

[25] Muralimanohar N, Balasubramonian R, Jouppi N. Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0 [C] //Proc of the 38th Annual IEEE/ACM Int Symp on Microarchitecture (MICRO). Los Alamitos, CA: IEEE Computer Society, 2007: 3-14

[26] Jouppi N P, Kahng A B, Muralimanohar N, et al. CACTI-IO: CACTI with off-chip power-area-timing models [J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2015, 23(7): 1254-1267

[27] Poremba M, Mittal S, Li D, et al. DESTINY: A tool for modeling emerging 3D NVM and eDRAM caches [C] //Proc of Design Automation and Test in Europe (DATE). Piscataway, NJ: IEEE, 2015: 1543-1546
[28] Lecun Y. LeNet [OL]. [2019-01-04]. http://yann.lecun.com/exdb/lenet/
[29] Liu Liu. ConvNet [OL]. [2019-01-04]. http://libccv.org/doc/doc-convnet/
[30] Subramanian A S. Caffe Model Zoo [OL]. [2019-01-04]. https://github.com/BVLC/caffe/wiki/Model-Zoo
[31] Lecun Y. MNIST [OL]. [2019-01-04]. http://yann.lecun.com/exdb/mnist/
[32] Stanford University, Princeton University. ImageNet [OL]. [2019-01-04]. http://www.image-net.org/

Mao Haiyu, born in 1993. PhD candidate. Her main research interests include NVM based main memory, processing in memory and machine learning.

Shu Jiwu, born in 1968. Professor and PhD supervisor. Fellow of CCF. His main research interests include non-volatile memory systems and technologies, network (cloud/big data) storage system, storage security and reliability, and parallel and distributed computing.