doi:10.11887/j.cn.202305026 http://journal.nudt.edu.cn

面向大规模卷积计算的多忆阻器阵列互连结构设计*

唐励勤,刁节涛,陈长林,骆畅航,刘 彪,刘思彤,张宇飞,王 琴
(国防科技大学 电子科学学院,湖南 长沙 410073)

摘 要:针对现有多忆阻器阵列集成架构中存在的数据加载、读出效率低以及阵列协同灵活性差等问题,提出一种高效率、高灵活度的阵列互连架构。该架构所采用的数据加载策略支持多种权重映射模式下的数据复用,减少了片外数据访存需求;所采用的计算结果读出网络支持多个处理单元灵活组合实现不同规模卷积运算,以及对计算结果的快速累加读出,进而提升了芯片灵活性和整体算力。在NeuroSim仿真平台上运行VGG-8网络进行的仿真实验表明,与MAX^{2}神经网络加速器相比,在仅增加6%面积开销的情况下,取得了146%的处理速度提升。

关键词:忆阻器;多阵列互连;卷积运算;神经网络加速器
中图分类号:TN492 **文献标志码**:A **文章编号**:1001-2486(2023)05-222-09

Multi-memristor-array interconnection structure design for large scale CNN acceleration

TANG Liqin, DIAO Jietao, CHEN Changlin, LUO Changhang, LIU Biao, LIU Sitong, ZHANG Yufei, WANG Qin
(College of Electronic Science and Technology, National University of Defense Technology, Changsha 410073, China)

Abstract: To address the problems of inefficient data loading and readout and poor flexibility of array collaboration in existing multi-memristor-array, a highly efficient and flexible multi-array interconnection architecture was proposed. The data loading strategy of the architecture supports data reuse in multiple weight mapping modes, reducing the need for off-chip data access; the readout network supports flexible combination of multiple processing units to achieve different scales of convolutional operations, as well as fast accumulation and readout of computation results, thus improving chip flexibility and overall computing power. Simulation experiments performed on the NeuroSim platform with running VGG-8 networks indicate a 146% increase in processing speed than that of the MAX^2 neural network accelerator, with only a 6% increase in area overhead.

Keywords: memristor; multi array interconnection; convolutional operation; neural network accelerator

在人工智能时代的大背景下,深度神经网络受到市场和学术界的广泛关注,特别是其中的卷积神经网络,在图像识别、数据处理和语音分析等多个领域取得了巨大的成功。为了加速卷积神经网络的推理过程,业界设计了多款神经网络加速器,例如DaDianNao[1]、TPU[2]等。其中绝大多数是采用了存算分离的冯·诺依曼架构的加速器,在卷积网络推理计算过程中,特征图像数据和网络权重数据需要在存储单元与运算单元之间频繁移动,从而导致较大的计算延迟与功耗,目前这一问题已经成为限制神经网络加速器性能提升的瓶颈[3]。模仿生物神经网络结构,将突触连接强度(网络权重)存储和神经冲动传导(特征图像数据与权重相乘)在同一位置实现,则可大大提高计算速度和能效。然而基于CMOS工艺实现权重存储和乘加运算需要较多的晶体管,进而导致较大的芯片面积和功耗。同时随着摩尔定律的延续即将受到物理层面的问题约束,在提高晶体管的集成度方面的发展也遇到了困难。在能效与集成度两方面的制约下,寻找新的高能效计算方式与计算器件是当前重要的研究方向。

忆阻器[4]作为一种新兴的信息器件,由于其阻值可调、非易失[5]、集成密度高[6]、生产工艺与CMOS工艺兼容[7]等特点,在近年来吸引了大量的关注与研究。忆阻器阵列在完成卷积运算时,权重以电导形式存储在忆阻器上,输入信号以电压形式加载到忆阻器两端,使得忆阻器在实现存

* **收稿日期**:2022-06-13
基金项目:国家自然科学基金资助项目(61804181,62074166);国家重点研发计划资助项目(2019YFB2205102)
作者简介:唐励勤(1998—),男,湖南永州人,硕士研究生,E-mail:Tangliqin_18@163.com;
王琴(通信作者),女,湖北武汉人,讲师,硕士,E-mail:qin.wang@nudt.edu.cn

储权重功能的同时又完成了计算,实现了“存算一体”[8],避免了计算中的数据搬移,节省了大量的功耗;同时由于其阻值变化的机制与人脑神经突触链接的可塑性机制相类似,非常适合用于类脑计算的推理过程。因此,忆阻器被认为是顺应人工智能时代、完成类脑计算的最有希望的候选器件之一。

在使用忆阻器阵列实现大规模深度神经网络时需要多个阵列协同工作,主要原因在于:① 在考虑忆阻器阵列的阻值波动、阻值开关比有限等非理想因素的影响下[9],为保证计算精度,通常需要限制参与乘累加运算的阵列规模[10];②为提高计算精度,目前单个网络权重需要由多个忆阻器来表示,在映射较大规模的卷积核时,单个忆阻器阵列无法满足需求;③由于忆阻器阵列的乘累加特性,不同卷积层之间的数据无法在同一阵列中完成运算,因此在实现多层卷积层时,需要多个阵列来实现。

已有多款基于忆阻器的卷积神经网络加速器设计公布,如 ISAAC[11]、Prime[12]、PipeLayer[13]、MAX^2[14]等。上述设计大多由通过片上网络互联的多个计算瓦片组成。计算瓦片(tile)是指可以整体挂载到片上网络或独立完成多个处理任务的一个单元,在处理数据的过程中无须与其他计算瓦片进行数据交互,其中通常集成了多个忆阻器阵列以及驱动阵列完成乘累加运算的外围电路。现有设计在乘累加运算电路实现方案、层间流水处理方案、权重映射策略等方面做出了创新型设计,然而仍然存在数据复用方式单一、多阵列协同灵活性差等问题。

针对上述问题,本文提出一种面向大规模卷积运算的高效率、高灵活度多阵列互连架构,该结构通过共享输入总线与定制读出网络实现了多个处理单元(processing elements)的互连。本文工作中设计了处理单元计算结果条件累加电路,支持多个处理单元的计算结果快速灵活累加与输出,使得瓦片内相邻处理单元可以组合实现多种规模的卷积核,以满足不同权重映射方式的需要;并优化了数据加载电路以支持各阵列之间的数据流动,在多种网络权重映射模式下均可实现数据复用。

1　忆阻器工作原理与现有架构

卷积运算的本质仍然是乘累加运算,而忆阻器交叉阵列结构实现乘累加运算具有极高的计算效率。如何通过多个阵列协同高效实现多层卷积运算是当前基于忆阻器的神经网络加速器需要解决的重点问题。

1.1　忆阻器实现卷积运算

卷积运算过程中,卷积窗口内的特征图像数据与卷积核对应位相乘后进行累加,因此卷积运算的本质仍然是乘累加运算。如图 1 所示,在使用忆阻器阵列完成乘累加运算时,权重以电导形式存储在忆阻器中,并将阵列列线箝位到参考电平,然后由阵列的左侧方向向阵列中输入电压形式的待处理数据,与忆阻器作用产生电流并在列线上汇集,列电流大小对应待处理数据与权重之间的乘累加运算结果。图中 $V_1 \sim V_i$ 代表输入电压,$G_1 \sim G_i$ 代表阵列上的忆阻器,$I_1 \sim I_i$ 代表各忆阻器的运算结果。列线上汇集的电流可通过 ADC 和敏感放大器转换为数字形式。

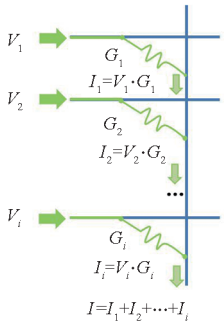


图 1　忆阻器完成乘累加运算

Fig. 1　Multiplication and accumulation of memristor array

1.2　现有神经网络加速器

Shafiee 等[11]提出了首个集成多个忆阻器阵列的神经网络加速器 ISAAC,根据每一层运算的需要,每个计算瓦片中含有多个 128 × 128 规模的忆阻器阵列来完成运算,神经网络中的每一层运算交付给不同的阵列来以流水处理方式提高整体的工作效率,降低了对数据缓冲空间的要求并增加了吞吐量。但计算瓦片中各阵列虽然共享 ADC 模块,但实际上各阵列相对对立,待处理数据加载和处理结果的读出均需单独进行;同时该结构并未充分考虑在神经网络运算中存在的数据和权重的复用特性。

PRIME[12]设计了基于忆阻器的全功能阵列和存储单元,其中全功能单元既可用于数据存储,又可配置成神经网络计算加速模块,显著地降低了面积的开销。在实现大规模神经网络时,它使用了数据总线来实现各个块之间的数据移动。该设计同样未考虑数据复用与多阵列协同工作需求。

PipeLayer[13]重现了 PRIME 全功能阵列的设计,并在 ISAAC 的基础上做出了改进,使得各个

阵列上的运算负载相对均衡。其逐层地计算输出图像数据，并且将输入特征图像广播到子阵列中以提高层内计算速度。但是与 ISAAC 同样未考虑各阵列计算结果的合并问题，每个阵列所完成的工作相对独立。

Mao 等[14]提出的 MAX^2 神经网络加速器，每个计算瓦片中含有排列成 3×3 阵列的 9 个处理单元，每个卷积核的所有通道中处于同一位置的权重映射到忆阻器阵列的一列上。同一行的三个处理单元形成一个一维的脉动阵列，以此实现待处理图像的数据复用。但是该权重映射方式在此硬件架构下仅支持大小为 3×3 的卷积核映射，难以灵活适用于不同的卷积核规模；并且该数据复用策略仅能应用于文中卷积核映射在 9 个处理单元中的情况，其复用方式单一。

总的来说，当前基于忆阻器阵列的神经网络加速器受到了广大研究人员的关注，在基本计算架构、层间流水实现、权重映射等方面进行了优化设计。但目前仍存在数据复用方式单一以及多阵列协同工作灵活性差等问题。本文架构通过优化数据加载方式，实现了对多种权重映射策略下的数据复用；采用定制设计的读出电路，使得多个处理单元之间能够灵活组合实现不同规模的卷积运算。

2 架构设计

2.1 权重映射方式

本文工作所设计的计算瓦片架构能够灵活支持多种权重映射方案。如图 2 所示，根据忆阻器阵列规模和卷积核规模，网络权重可以按照全展开、按位置展开或按行/列展开的方式进行映射，以充分利用忆阻器阵列中的忆阻器单元。对于 N 个规模为 $K\times K\times C$ 的卷积核，不同映射模式实现方式如下：

在进行全展开映射时[11-13]，每个卷积核展开为一个长度为 $K\times K\times C$ 的向量并映射到忆阻器阵列的一列中，N 个卷积核则会占据阵列的 N 列。

在进行按位置展开时[14]，即将卷积核中相同位置上所有通道内的权重展开并映射到忆阻器阵列的一列中，完成所有权重映射需要 $K\times K$ 个忆阻器阵列，并且占用每个阵列中的 C 行、N 列。

按行/列展开[15]可以看作全展开与按位置展开方式的折中，即将卷积核中属于同一行或列的所有权重展开为向量并映射在忆阻器阵列的一列中，映射上述规模的卷积核需要 K 个忆阻器阵列，并占用每个阵列中的 $K\times C$ 行、N 列。

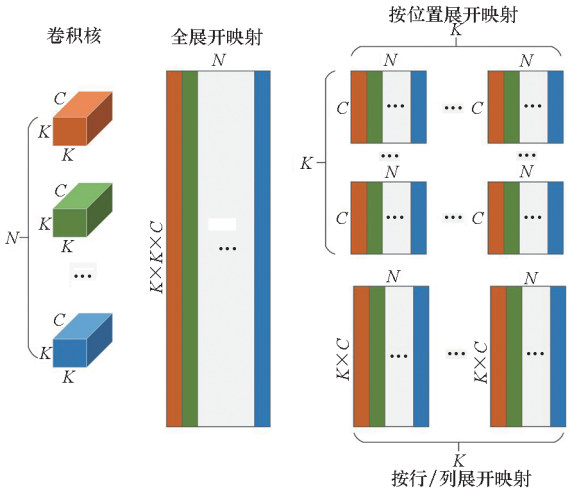


图 2 三种映射方式

Fig. 2 Three types of weight mapping

在上述映射方式中，所需的忆阻器阵列行数较多或单个卷积核需要映射在多个阵列上时，均需要将多个忆阻器阵列对应的列结果进行累加。

在某些应用场景下（如嵌入式应用），其神经网络采用的卷积核通常较小，此时可采用传统全展开的映射方式，并将卷积核在阵列内多次错位重复映射，增加计算的并行度，提高计算效率；而在映射较大规模的卷积核时即可采用按位置展开的方式进行映射，使得阵列中忆阻器利用率更高。

2.2 数据复用策略

在滑窗卷积运算过程中，相邻卷积窗口间存在较多的数据复用，充分利用这一特性，可大幅减少片外访存需求。

为便于实现不同权重映射模式下的数据复用，如图 3 所示，本设计所采用的电路使得处理单元中的数据寄存器除能够接收来自数据总线上的数据以外，还能够接收来自右侧阵列同一行与下方相邻行的数据寄存器中的数据。

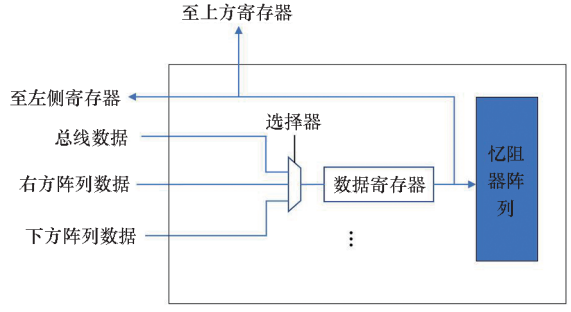


图 3 数据寄存器数据加载电路

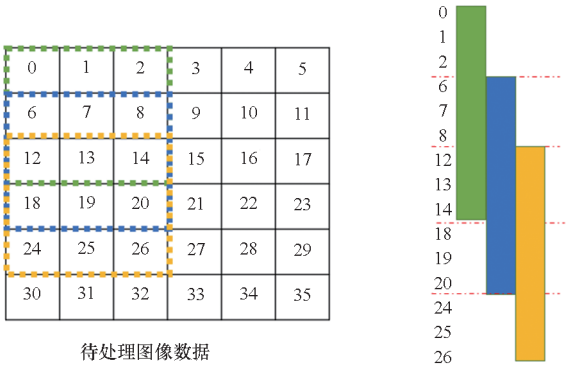
Fig. 3 Data register data loading circuit

在实现数据复用时，首先以在单个处理单元内将卷积核全映射展开的方案为例，如图 4(a)所示。卷积窗口垂直滑动时，在第一个视野里计算

的数据中 6 ~ 8、12 ~ 14 仍会参与下一个窗口内的计算,在滑动中由数据加载电路实现下方阵列数据向上加载并向上加载更新三次,即原本的数据 0、1、2 被丢弃,数据 6、7、8 存储到原本数据 0、1、2 所在的寄存器,数据 12、13、14 存储到原本数据 6、7、8 所在的寄存器,之后向末端三个寄存器加载新的图像数据,从而完成卷积窗口的滑动,并实现了待处理数据在垂直滑窗时的复用。

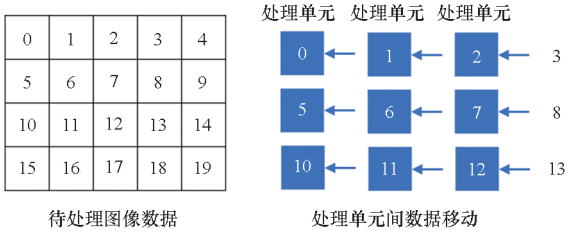
在实现水平卷积上的数据复用时,仍以图 4 中的待处理图像数据为例,在第一个视野窗口中的 1、2、7、8、13 与 14 仍会参与下一个窗口内的计算,在由数据加载电路将下方阵列数据向上更新一次之后,最上方三个数据由 0、1、2 变为 1、2、6 传入数据 3 将数据 6 替换即可完成视野里第一行的更新,以此类推完成整个视野的更新,实现了待处理数据在水平滑窗时的复用。

在本架构下考虑按位置展开的权重映射方案时,可依靠数据加载电路中能够加载相邻阵列数据的功能完成对输入图像数据的复用。如图 4(b)所示,相邻的 9 个处理单元加载了输入特征图像中一个 3 ×3 卷积视野的数据,当卷积窗口滑动时,通过上述功能将右方处理单元存储的数据依次向左进行移动更新,然后向最右方处理单元中加载新的待处理数据,即可完成数据的复用。为便于举例仅在图 4(b)中展示单通道下的数据。



(a) 全展开映射数据复用策略

(a) Fully expanded data reuse strategy



(b) 处理单元间数据移动

(b) Horizontal data multiplexing between processing units

图 4　支持的数据复用方式

Fig. 4　Supported data multiplexing methods

实现按行展开权重映射方案,当卷积窗口垂直滑动时,需要由数据加载电路将相邻阵列的数据加载至左侧阵列进行更新,同时向最右侧阵列加载新的待处理数据;当卷积窗口水平滑动时,则需要将下方阵列数据向上更新 C 次(C 为卷积核通道数),之后向末端的 C 个寄存器加载新的图像数据。凭借上述两种数据加载模式能够完成整个视野的更新,并分别实现了待处理数据在垂直与水平滑窗时的复用。

2.3　结果读出策略

当卷积核规模较大时,需要由多个处理单元来完成映射,相应地,在每个处理单元计算得出结果后,需要将输出整合累加以获得完整卷积计算的结果。为适应这一需求,本文设计了定制化的结果读出电路,如图 5 所示。

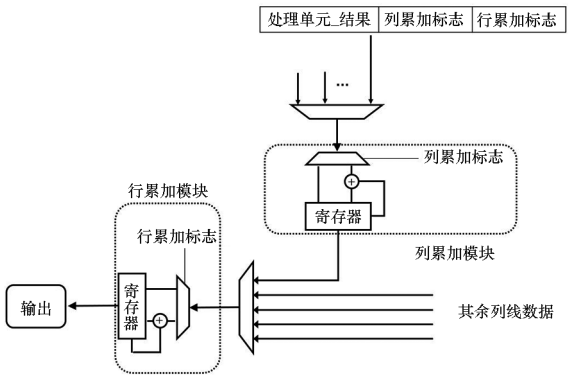


图 5　计算结果条件累加

Fig. 5　Calculation of conditional accumulation of results

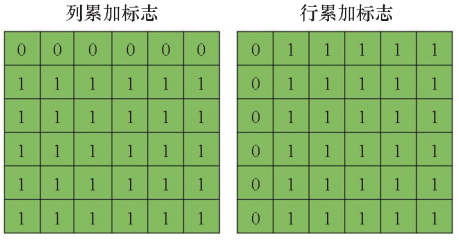
为了支持计算瓦片中处理单元灵活组合实现不同规模卷积运算,设计了行、列累加标志用于判断该阵列计算结果是否需要同相邻阵列相累加。每个处理单元的计算结果输出时附带行、列累加标志。当列累加标志位为 1 时,则代表该处理单元的计算结果需要同上方处理单元的计算结果相累加,若为 0 则不累加;当行累加标志位为 1 时,则代表该处理单元的计算结果需要同左上方处理单元的计算结果相累加,若为 0 则不累加。

在进行累加时,由列累加模块通过多路选择器自上而下依次读取各处理单元的计算结果,并根据列累加标志位判断是否需要与之前暂存在模块内寄存器的值累加,累加结果存入模块内部的寄存器中。所有列中的列累加模块同步读取计算结果,并将当前周期读取的列累加标志与上一周期获得的行累加标志同步传输至行累加模块。

列累加模块每读取一个处理单元结果,行累加模块即查看一次各列结果中的列累加标志。将

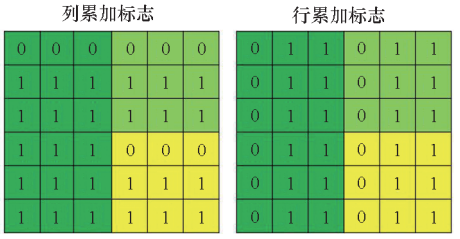
列累加标志连续为 1 的结果进行累加，若为 0 则无须再与上方结果累加。行条件累加模块将行累加标志连续为 1 的列的计算结果进行累加，遇到某一列的行累加标志为 0 时，代表着所需的累加已经完成，可以将累加结果送入计算瓦片缓存内，以备输出至计算瓦片外或再次送入其他处理单元中进行下一层卷积运算。

图 6 展示了几种通过设置行、列累加标志来实现不同处理单元结果组合的示例，计算瓦片内各处理单元的计算结果可根据累加标志完成不同情况的组合。以图 6(c)为例进行说明，列累加模块从上至下读取处理单元的累加标志位，在第四行读取到 0 时，则该行内结果无须再参与累加；此时行累加模块从左至右读取其行累加标志位，在读取至第四列的 0 时停止累加。至此浅绿色模块内处理单元的计算结果完成累加合并，以此类推该计算瓦片能够对浅绿、深绿、蓝色和黄色四个模块内的处理单元分别进行结果的累加合并。需要注意的是，参与结果合并的处理单元需要能够组成一个矩形。凭借该结果读出策略，本架构得以完成多阵列的协同工作，灵活完成多种规模的卷积运算。



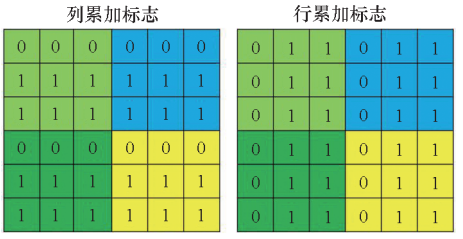
(a) 实现一组卷积核

(a) Implementation of one set of convolutional kernels



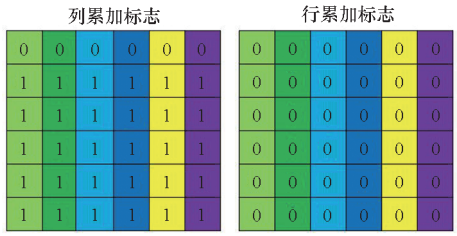
(b) 实现三组卷积核

(b) Implementation of three sets of convolutional kernels



(c) 实现四组卷积核

(c) Implementation of four sets of convolutional kernels



(d) 实现六组卷积核

(d) Implementation of six sets of convolutional kernels

图 6 多处理单元实现多规模卷积核

Fig. 6 Multi-processing elements for multi-scale convolutional kernels

3 硬件设计

3.1 芯片级设计

如图 7 所示，芯片中包含多个计算瓦片，1 个全局缓存、激活函数单元和池化单元，片上的所有单元与计算瓦片由 H 树总线完成互连，根据每一层的卷积核大小将其分配给一个或者多个计算瓦片来完成计算。计算过程：从全局缓存中提取各层的输入特征图像数据，通过数据总线传入相应的计算瓦片中进行处理。计算完成后，计算瓦片将得到的计算结果传入累加单元与其余结果完成累加，之后结果再被送入外部的激活函数单元、池化单元、全局缓存或是在需要的情况下作为下一层的输入再次传入计算瓦片。

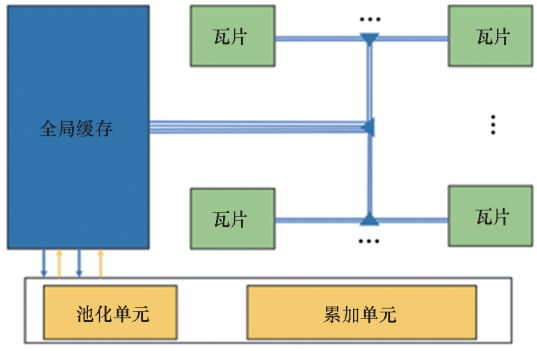


图 7 芯片总体架构

Fig. 7 Chip-level architecture

3.2 计算瓦片级设计

每个计算瓦片由 256 个处理单元构成，其排列成 16 × 16 的阵列，另外包含 1 个计算瓦片缓存与 17 个条件累加单元。各个处理单元之间由定制化的数据总线完成互连，每一列处理单元末端设置一个列累加单元来完成该列上处理单元计算结果的累加，由一个行累加单元与所有的列累加单元相连。借由以上的互连设计实现了处理单元

之间计算结果的合并,以获得多通道下合并的输出特征图像数据。

以下举例说明单个计算瓦片在完成一层卷积运算时,其相关的数据流情况。例如使用该计算瓦片完成特征图像与 16 个大小为 3 × 3 × 16 的卷积核的卷积运算,瓦片内单个处理单元的忆阻器规模为 64 × 64,在采用全展开映射方案时需要将单个卷积核映射在 3 个处理单元上,如图 8 中处理单元 PE1 ~ PE3 所示。在进行运算时其状态转移过程如图 9 所示。

步骤 1:将第一个卷积窗口内的输入特征图像数据通过总线加载到处理单元的各个数据寄存器上;

步骤 2:由外围电路驱动将寄存器中的数据加载至忆阻器阵列上并完成乘累加运算,随后相关的 DAC 电路将输出电流转换为数字信号缓存在处理单元配备的缓存中;

步骤 3:在计算结果读出时,列累加模块读取到处理单元 PE1 ~ PE3 的累加标志位为 1,将上述三个处理单元的计算结果累加合并后向外读出;

步骤 4:将处理单元中的各数据寄存器根据卷积窗口的滑动进行垂直方向上的移位更新;

步骤 5:从外部通过总线加载新的图像数据到数据寄存器中以完成卷积窗口的滑动。

完成更新后再次启动计算,重复步骤 2 ~ 5 直至完成该层的卷积运算。在一块计算瓦片中包含多个处理单元,将卷积核在不同的处理单元上多次例化,能够实现多路并行,提高计算效率。

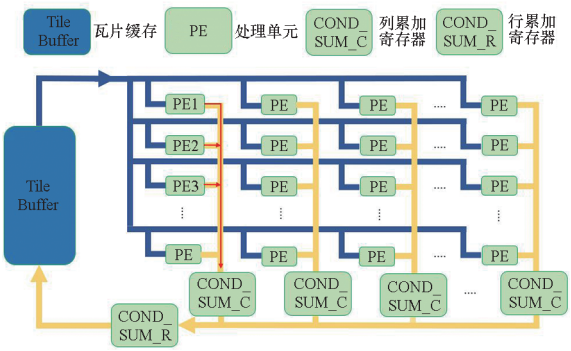


图 8　计算瓦片内总架构

Fig. 8　Tile-level architecture

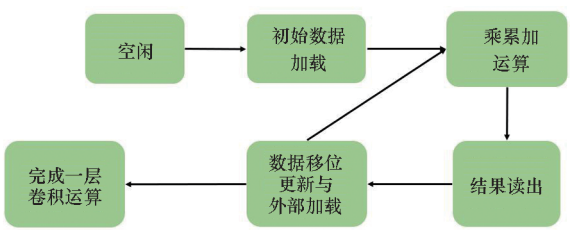


图 9　实现卷积运算的状态转移图

Fig. 9　Finite state machine for implementing convolutional operations

3.3　处理单元级设计

每个处理单元内含有一个 64 × 64 规模的忆阻器阵列,以及搭配工作的外围电路如读出电路、多路选择器、移位累加器、数据寄存器、控制寄存器与输出寄存器。其中控制寄存器中包含自动更新使能与更新标志位。每 8 列忆阻器共享一个读出电路,读出电路的读出精度为 5 bit,这足以避免显著的精度损失[9]。在运行 VGG - 8 网络时,考虑到单个子阵列的阵列行数为 64,在每个处理单元内部均设置了 64 个数据寄存器用于每一行忆阻器的输入图像数据暂存。计算过程:从计算瓦片缓存中取得待处理图像数据首先存入处理单元的 64 个数据寄存器中,每个寄存器的大小为 8 bit,即所有寄存器总容量为 512 bit,运算开始后向忆阻器阵列输入电压数据。经过忆阻器计算后由读出电路将结果电流读出并在移位累加器中完成对整列结果的累加,每列结果均为 64 个 8 bit 数据相加,需要由 14 bit 的输出寄存器进行寄存。

通过总线对待处理数据进行加载时,可对指定处理单元的地址进行单个处理单元的写入,同时为降低数据总线的负载,设计了共享标志位用于数据多播,在该标志位为高电平时,可向多个处理单元设置为同一地址 ID 以完成单次数据加载到多个处理单元的功能。处理单元中各数据寄存器带有更新标志位,当标志位为低电平时该数据寄存器才接受写入新的数据。

4　实验验证

基于 NeuroSim 平台对以本文架构为原型的加速器进行了仿真,以更好体现其在实际应用中的意义,并将其与相关的神经网络加速器进行比较,评估其优劣势。

4.1　评估设置

使用 DNN + NeuroSim 框架来评估在 32 nm 工艺节点下本文所提出的互连架构,修改相关的 NeuroSim 代码来反映该互连架构的权重映射方式与数据流。考虑到运行的神经网络需要能够覆盖使用瓦片级与芯片级上的各项处理性能,在对比评估工作中采用了处理 CIFAR - 10 数据集的 VGG - 8 网络来评估性能与能效,其中网络的权重位宽设置为 5 bit,单个忆阻器的存储状态数为 32,数字电路的工作频率为 1 GHz。在这项对比

工作中,对比对象使用 MAX^2 的映射方式与数据流,两种体系结构使用总规模大小相同的可变电阻式存储器(resistive random access memory, ReRAM)阵列来运行 VGG-8 网络。

在对比对象中,使用了 128×128 规模大小的 ReRAM 阵列作为子阵列,每个处理单元的规模为 512×512,即单个处理单元包含了 4×4 规模的子阵列,每个计算瓦片由 4 个处理单元组成。在本文的互连结构中,由小阵列互连组成大规模阵列,采用了 64×64 规模大小的子阵列作为单个处理单元内部的忆阻器计算阵列,每个计算瓦片由 16 * 16 个处理单元组成,单个计算瓦片含有的 ReRAM 阵列数量与对比对象相同,均为 1 024×1 024。

4.2 功耗与面积详情

在 NeuroSim 下仿真得到基于本文工作所提出架构的加速器的各项参数,如表 1 所示。在处理单元层级,阵列的上方与左方配置有字线/位线的开关矩阵以控制电压的加载,每个读出电路负责完成 8 列忆阻器的结果电流读出,子阵列中移位加法器完成了处理单元所需要的累加功能,将所有列电流结果的移位累加并存入 14 bit 的基于 D 触发器的处理单元缓存中,输入缓存同样是基于 D 触发器。单个处理单元面积为 6.081×10^3 μm^2,其中读取电路所占面积为 78.9%。在计算瓦片层级,共含有 16×16 个处理单元,配置的基于静态随机存储器(static random access memory, SRAM)的缓存面积为 2.64×10^4 μm^2,每读出 1 bit 的数据消耗能量为 0.002 74 pJ,所用于完成累加功能的电路单元共有 17 个,每个累加单元进行一次累加操作消耗的能量为 0.080 pJ,总的面积为 1.64×10^4 μm^2。在芯片层级,由 16 个计算瓦片完成 1 至 6 层的卷积层运算,由 9 个计算瓦片完成层 7 与层 8 的全连接层运算,基于 SRAM 的全局缓存大小为 128 KB,最大池化单元所占面积为 3.17×10^4 μm^2,在运行 VGG-8 处理 CIFAR-10 数据集时,整个芯片的动态功耗为 $2.414\ 52\times10^7$ pJ。

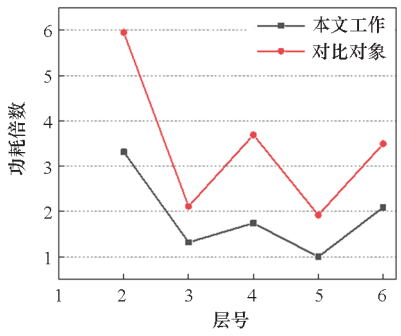
表 1 架构参数

Tab. 1 Architecture parameters

层级	模块	规模	功耗	面积/μm^2
阵列级	忆阻器	64×64		1.05×10^2
	移位加法器	9	每个移位加法器进行一次运算:0.151 pJ	5.94×10^2
	开关矩阵	1	每个开关矩阵进行一次运算:0.139 pJ	2.55×10^2
	多级感应放大器及其编码器	8		4.57×10^3
处理单元级	子阵列	1		5.79×10^3
	输入缓存	512 bit	0.002 72 pJ/bit	2.46×10^2
	输出缓存	14 bit	0.002 72 pJ/bit	3.90×10^1
计算瓦片级	处理单元	16×16		1.56×10^6
	缓存	2 KB	0.002 74 pJ/bit	2.64×10^4
	累加单元	17	每个累加单元进行一次运算:0.080 pJ	1.64×10^4
芯片级	卷积层	16		2.68×10^7
	全连接层	9		1.51×10^7
	最大池化单元	1 024	每个最大池化单元进行一次运算:0.068 pJ	3.17×10^4
	全局缓存	128 KB	0.002 74 pJ/bit	2.74×10^6
	累加单元	128	每个累加单元进行一次运算:0.021 7 pJ	1.65×10^4

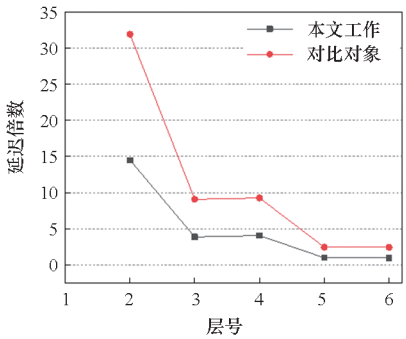
4.3 实验结果与讨论

图 10 列出了本文工作与 MAX^2 架构在运行 VGG-8 网络的 2~6 层时的缓存与内部互连的能耗与计算延迟,所有数据均以本文工作中第 5 层的数据为量化指标。2~6 层其输入特征图像逐层减小、卷积核数量逐层增加,可见由于第 2 层的输入特征图像数据最大,其需要向缓存访问数据的次数最大并且数据的移动量最大,故第 2 层的相关功耗与延迟均最大,而第 5 层的结果最小正是因为其输入特征图像最小。在第 4 层与第 6



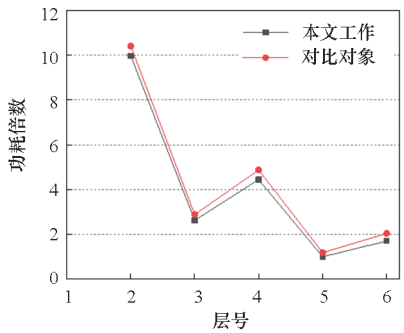
(a) 缓存功耗对比

(a) Cache power comparison



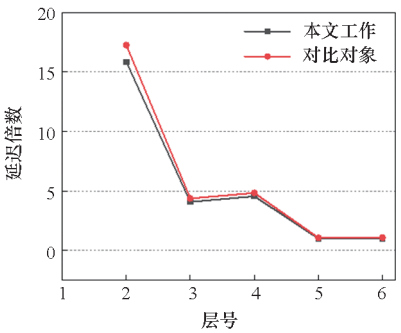
(b) 缓存延迟对比

(b) Cache latency comparison



(c) 内部互连功耗对比

(b) Internal interconnect power comparison



(d) 内部互连延迟对比

(b) Internal interconnect latency comparison

图 10　缓存与内部互连的指标对比

Fig. 10　Comparison of metrics in cache and internal interconnect

层的结果呈现上升趋势的原因是其计算后需要通过最大池化层的计算。并且在第 6 层时,其卷积运算所需要的卷积核数量达到最大值,参与计算的处理单元数量最多,故在第 6 层的缓存功耗开销仅次于第 2 层,借助于忆阻器阵列计算时的高并行度,其计算延迟并未受到影响。而在内部互连方面,尽管数据复用减少了总线的使用次数,但是由于需要完成小阵列之间计算结果的合并增加了总线负载。

总的来说,在内部互连的延迟与功耗相差无几的情况下,本文提出的方案相比于基准在缓冲区减少了 40.2% 的动态功耗与 57% 的延迟。本文工作能够取得上述优势,主要得益于阵列实现了在处理单元层级上的数据复用减少了数据的远程移动量以及定制化的数据总线降低了从全局缓存中取出数据的访问次数。

表 2 对比了本文工作与基准的芯片面积、延迟与能效。同基准相比,本文工作在采用新的数据流策略的情况下,仅增加 6% 的面积开销,取得了 146% 的处理速度提升。主要原因在于本文工作提高了片内数据的复用率以及减少了数据向片外的搬移次数,同时在外围电路上单个处理单元所配置的更小更快的外围电路提供了更高的计算速率。超额的面积资源主要在于实现该设计时需要较多子阵列的外围电路,能效的下降同样也是由于总的外围电路的功耗开销稍高。

表 2　评估结果

Tab. 2　Assessment results

架构	面积/mm^2	延迟/ms	能效(TOPS/W)
本文工作	53.66	9.407	49.226
对比对象	50.44	13.801	51.183

5　结论

本文提出了一个新型的基于忆阻器阵列的互连架构,高效地实现了阵列数据加载、输入与多个阵列结果的合并。使用了 DNN + NeuroSim 架构来评估了这项工作在 32 nm 工艺节点下运行 VGG - 8 网络处理 CIFAR - 10 数据集时的功耗与延迟表现。实验结果表明,以本文架构为基础的原型加速器对比现有的神经网络加速器,在仅增加 6% 的面积开销的情况下,取得了 146% 的处理速度提升。

参考文献(References)

[1]　LUO T, LIU S L, LI L, et al. DaDianNao: a neural network

supercomputer[J]. IEEE Transactions on Computers, 2017, 66(1): 73 – 88.

[2] JOUPPI N P, YOUNG C, PATIL N, et al. In-datacenter performance analysis of a tensor processing unit [C]// Proceedings of the 44th Annual International Symposium on Computer Architecture, 2017.

[3] PENG J, LI Z W, LIU H J, et al. Network pruning towards highly efficient RRAM accelerator[J]. IEEE Transactions on Nanotechnology, 2022, 21: 340 – 351.

[4] WONG H S P, LEE H Y, YU S M, et al. Metal-oxide RRAM[J]. Proceedings of the IEEE, 2012, 100 (6): 1951 – 1970.

[5] LANZA M, SEBASTIAN A, LU W D, et al. Memristive technologies for data storage, computation, encryption, and radio-frequency communication [J]. Science, 2022, 376(6597): eabj9979.

[6] MA M L, YANG Y, QIU Z C, et al. A locally active discrete memristor model and its application in a hyperchaotic map[J]. Nonlinear Dynamics, 2022, 107 (3): 2935 – 2949.

[7] HAMDIOUI S, AZIZA H, SIRAKOULIS G C. Memristor based memories: technology, design and test [C]// Proceedings of 2014 9th IEEE International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS), 2014.

[8] LIU C S, CHEN H W, WANG S Y, et al. Two-dimensional materials for next-generation computing technologies [J]. Nature Nanotechnology, 2020, 15(7): 545 – 557.

[9] CHEN P Y, PENG X C, YU S M. NeuroSim: a circuit-level macro model for benchmarking neuro-inspired architectures in online learning[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems: a Publication of the IEEE Circuits and Systems Society, 2018, 37(12): 3067 – 3080.

[10] YAO P, WU H Q, GAO B, et al. Fully hardware-implemented memristor convolutional neural network [J]. Nature, 2020, 577(7792): 641 – 646.

[11] SHAFIEE A, NAG A, MURALIMANOHAR N, et al. ISAAC: a convolutional neural network accelerator with in-situ analog arithmetic in crossbars [C]// Proceedings of ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), 2016.

[12] CHI P, LI S C, XU C, et al. PRIME: a novel processing-in-memory architecture for neural network computation in ReRAM-based main memory [C]//Proceedings of ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), 2016.

[13] SONG L H, QIAN X H, LI H, et al. PipeLayer: a pipelined ReRAM-based accelerator for deep learning[C]//Proceedings of 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA), 2017.

[14] MAO M Q, PENG X C, LIU R, et al. MAX2: an ReRAM-based neural network accelerator that maximizes data reuse and area utilization [J]. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 2019, 9(2): 398 – 410.

[15] LUO C H, DIAO J T, CHEN C L. FullReuse: a novel ReRAM-based CNN accelerator reusing data in multiple levels[C]// Proceedings of 2020 IEEE 5th International Conference on Integrated Circuits and Microsystems (ICICM), 2020.