DOI:10.19816/j.cnki.10-1594/TN.2021.02.042

面向忆阻器存算一体架构的仿真器设计：综述与展望*

朱振华，朱　昱，汪　玉，杨华中
（清华大学电子工程系 北京 100084）

摘　要：大数据时代，存储器与处理器之间的数据搬运成为限制传统冯·诺依曼架构处理性能及能效提升的主要瓶颈。基于新兴忆阻器的存算一体架构可以在存储阵列中原位完成矩阵向量乘运算，避免了矩阵数据的搬运，从而可以显著提高神经网络等算法的性能与计算能效。由于神经网络算法模型复杂、存算一体架构规模庞大，使用传统SPICE仿真工具对存算一体架构进行电路级仿真耗时长，严重影响存算一体芯片的设计开发效率。为提高仿真效率并缩短芯片开发周期，研究人员提出了多款面向存算一体架构的专用仿真器。本文将对目前主流的存算一体仿真器进行介绍与归纳总结，并对存算一体仿真器的未来发展提出展望。

关键词：忆阻器；存算一体；仿真器；神经网络
中图分类号：TM712　　**文献标识码**：A　　**国家标准学科分类代码**：510

Simulator design for memristor-based processing-in-memory architecture: Survey and prospect

ZHU Zhenhua, ZHU Yu, WANG Yu, YANG Huazhong
(*Department of Electronic Engineering, Tsinghua University, Beijing 100084, China*)

Abstract: In the big data era, the data movements between memory and processor become the main bottleneck to improve the performance and energy efficiency of traditional von Neumann architecture. Emerging memristor-based processing-in-memory (PIM) architectures can perform in-situ matrix-vector-multiplications (MVM) in memory, which eliminates the matrix data movements, improving the performance and energy efficiency of neural network (NN) algorithms. Because of the complex NN models and large-scale PIM architectures, using conventional SPICE tools to perform circuits-level simulation for PIM architectures will cause unacceptable long time, seriously affecting the design efficiency of PIM chips. In order to improve the simulation efficiency and shorten the PIM chip development cycle, researchers have proposed multiple PIM simulators. In this paper, we present a review of existing representative PIM simulator designs, and put forward the prospects for the future of PIM simulators.

Keywords: memristor; processing-in-memory; simulator; neural network

0　引　言

在人工智能与大数据时代，神经网络（neural network，NN）在人脸识别、物体检测、自然语言处理等领域展现出了强大的能力[1-3]。但是，与神经网络强大能力相对应的是，神经网络的参数量与计算

*基金项目：自然科学基金（61832007）、国家重点研发计划（2017YFA0207600）项目资助
朱振华，博士在读，主要研究方向为基于忆阻器的存内计算与近存储计算。E-mail：zhuzhenh18@mails.tsinghua.edu.cn
汪玉（通信作者），教授，主要研究方向为智能芯片以及高能效电路与系统。E-mail：yu-wang@tsinghua.edu.cn

量呈现出爆炸式的增长趋势。以2012年的AlexNet[4]与2021年的Switch Transformers[5]为例，神经网络的参数量由6.0×10^8个急剧增长至1.6×10^{13}个，相应的浮点运算次数也由7.2×10^8次增加至9×10^{11}次。由于大规模的参数量与计算量，使用传统冯·诺依曼架构计算神经网络时，计算系统需要在分离的存储器与处理器之间进行大量数据搬运，而这部分数据搬运占据了计算系统超过80%的总处理能耗。因此，如何降低存储器与处理器之间的数据搬运代价是后摩尔时代提高硬件计算能效所需要回答的关键问题。

新兴的忆阻器件为后摩尔时代突破冯·诺依曼架构提供了可行方案。典型忆阻器包括阻变式随机访问存储器[6]（resistive random access memory，RRAM）、相变存储器[7]（phase change memory，PCM）、磁性随机访问存储器[8]（magnetic random access memory，MRAM）等多种类型。忆阻器具备阻值可调与非易失的特性，多个忆阻器组成的交叉阵列二维结构可以实现高密度数据存储与存内计算。使用交叉阵列中的忆阻器电导存储矩阵数据，将输入向量转换为电压信号加载在交叉阵列的字线上，即可在阵列位线输出端得到模拟域的矩阵向量乘计算结果。在该类计算模式中，矩阵数据存储在忆阻器交叉阵列内，计算亦发生在交叉阵列中而无需进行矩阵数据搬运，实现了存储计算一体化。已有工作表明，基于忆阻器的存算一体架构相比于基于CMOS的领域定制化专用芯片可以达到2~3个数量级的计算性能与能效提升[9]。

由于新兴的忆阻器件与忆阻器交叉阵列缺少统一的标准单元库，存算一体架构与芯片设计依赖于SPICE电路仿真进行性能评估。但是SPICE仿真面临仿真时间长的严重问题，以单个512 × 512 RRAM交叉阵列的仿真为例，其SPICE电路仿真时间将超过2 h。此外，神经网络的参数多，存储一个完整神经网络的存算一体架构规模巨大，通常包含成百上千个忆阻器交叉阵列。单个阵列小时量级的电路仿真时间使得电路级仿真工具无法用于存算一体架构的整体仿真。存算一体领域的研究亟需专用高效仿真器。

为解决存算一体架构电路仿真耗时长的问题，研究人员提出了多种面向存算一体架构的定制化仿真器。这些仿真器覆盖准确率评估、架构性能评估、电路性能仿真等多方面功能，显著提高了存算一体架构的评估与设计效率。本文将对目前主流的存算一体仿真器进行介绍与归纳总结，并对存算一体仿真器的未来发展提出展望。

1 忆阻器与存算一体架构

1.1 忆阻器交叉阵列

忆阻器是一种阻值可调且具有非易失性的新型器件，多个忆阻器按照二维阵列结构排列连接构成交叉阵列结构。一种典型的忆阻器交叉阵列如图1所示。

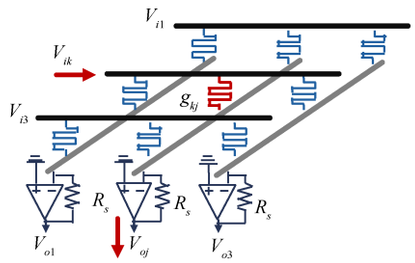

图1 忆阻器交叉阵列

Fig.1 Memristor crossbar array

在忆阻器交叉阵列中，通过式(1)可以使用忆阻器电导（g_{ij}）与跨阻放大器负载（g_s）共同表示矩阵元素（c_{ij}）。

$$c_{i,j} = -\frac{g_{i,j}}{g_s} \tag{1}$$

将输入向量转化为电压（$\vec{V}_i$）加载至交叉阵列的位线上，位线上的电流（i_{oj}）可由式(2)得到。位线电流经过跨阻放大器转换成电压的形式，该输出电压（$\vec{V}_o$）可以表示矩阵向量乘的计算结果。

$$i_{o,j} = \sum_{k=1}^{N} g_{k,j} V_{i,k} \tag{2}$$

基于上述方案，使用忆阻器交叉阵列计算矩阵向量乘时，矩阵数据全部存储在忆阻器交叉阵列中，无需进行矩阵数据的搬运，从而实现存算一体。此外，模拟域实现矩阵向量乘的理论计算复杂度也由

$O(N^2)$降低至$O(1)$,有望进一步提高计算性能。

由于忆阻器交叉阵列在模拟域利用电导、电压与电流完成计算,为了与其他数字计算模块协同工作并存储矩阵向量乘的计算结果,存算一体架构中通常需要在交叉阵列的输入输出端加入数模/模数转换器。例如在字线输入端加入数字-模拟转换器(digital-to-analog converter, DAC),在位线输出端加入模拟-数字转换器(analog-to-digital converter, ADC)。

1.2　面向神经网络的存算一体架构

神经网络的核心计算为矩阵向量乘计算,因此,基于忆阻器的存算一体架构有望提高神经网络的处理性能与计算能效。以卷积神经网络中的代表计算卷积计算为例,该计算可以表示为式(3)的形式:

$$F_o(x,y,z)=f\left(\sum_{i=0}^{K-1}\sum_{j=0}^{K-1}\sum_{k=1}^{C_{in}}F_i(x+i,y+j,k)w_z(i,j,k)\right) \quad (3)$$

式中:三维矩阵F_o、F_i、w_z分别代表网络层输出特征图、输入特征图与第z个卷积核,卷积核的尺寸为$K\times K\times C_{in}$;$f(\cdot)$代表非线性激活函数(如整流线性单元)。在存算一体架构中,可以将每个卷积核与输入特征图由三维矩阵转换为一维向量(image-to-column, im2col),并把不同卷积核向量排列在一起构成权重矩阵,由此可以将式(3)转换为矩阵向量乘运算。将不同卷积核映射在忆阻器交叉阵列的不同列上,加载表示输入特征图的电压向量在交叉阵列字线上,即可完成网络层计算。

在存算一体架构设计方面,根据应用计算的类型可将存算一体架构分为神经网络前向推理计算架构与神经网络训练架构。表1所示为代表性的存算一体架构,这些架构相比于CMOS加速器或GPU均可达到2~3个数量级的性能与能效提升。

表1　代表性的存算一体架构

Table 1　Representative PIM architectures

应用类型	代表性架构
神经网络前向推理计算	PRIME[9], ISAAC[10], PUMA[11]
神经网络计算	TIME[12], PipeLayer[13]

2　存算一体架构仿真器设计概述

2.1　存算一体架构仿真器仿真流程

存算一体架构仿真器的整体仿真流程如图2所示,该流程可以概括现今主流存算一体架构仿真器的设计思想与基本工作流程。仿真器的输入包含3方面:神经网络参数(包括网络权重及输入数据)、神经网络结构(如网络深度与宽度)以及存算一体架构单元设计(如忆阻器参数与交叉阵列尺寸等)。仿真器的输出主要由两方面组成,分别是基于存算一体架构的神经网络计算准确率以及存算一体架构的硬件计算性能(如计算延时与功耗)。

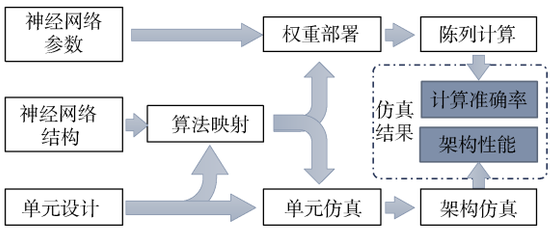

图2　存算一体架构仿真流程

Fig.2　Simulation flow for PIM architectures

如1.1节所述,忆阻器交叉阵列的输入与输出端需要使用DAC/ADC进行模拟信号与数字信号的转换。不同于基于CMOS电路的数字计算,模数转换将会为存算一体架构的计算过程引入量化误差。此外,由于忆阻器工艺尚不成熟,忆阻器器件存在阻值扰动(variation)与卡死型错误(stuck-at-fault, SAF)等非理想因素。这些非理想因素也会使存储在交叉阵列中的权重值发生不可预期的改变。由于模数转换量化误差以及忆阻器非理想因素,存算一体架构在计算神经网络时面临矩阵向量乘计算结果存在偏差、神经网络计算准确率下降的问题。因此,计算准确率的评估是存算一体架构仿真中的重要组成部分。存算一体架构计算准确率的整体仿真流程包括如下几步:1)根据存算一体计算单元设计参数和神经网络结构完成算法映射,为每一个网络层分配硬件计算单元(如使用的忆阻器交叉阵列的数目);2)根据算法映射结果对神经网络参数完成权重部署,即对权重按块拆分,分解分配至不同的交叉阵列

上;3)根据器件非理想因素和模数转换量化误差分别对权重、输入与输出结果进行更新,模拟阵列计算行为;4)将阵列结果汇总为网络层计算结果,逐层传播完成计算准确率仿真。

对于硬件性能仿真,由于存算一体架构设计空间巨大,基于电路矩阵求解的SPICE电路级仿真往往需要花费数天乃至数月的时间,这对于在早期架构设计阶段进行快速设计空间搜索来说是非常不利的。为避免耗时长的电路矩阵求解,主流的存算一体仿真器采用层级分解的分析方法。在层级分解分析中,顶层存算一体架构会被拆解封装为多个架构层级,如芯片级(chip level)、存储片级(tile level)、处理单元级(processing element level)和交叉阵列级(crossbar level)。一个架构层级内使用相同的分析方式,不同网络层中同一架构层级的计算单元仅存在部分参数配置方面的差别。在整个仿真流程中,较低的架构层级将该层级仿真结果作为输入传递至较高架构层级进行后续仿真,较高的电路层级无需了解掌握低架构层级的全部电路信息。得益于层级分解的分析方法,各层级的问题求解规模被有效限制,仿真效率得以显著提升。对于每一个架构层级,已有工作通常使用解析分析法和拟合近似法两种分析方法进行仿真加速。在解析分析法中,仿真器将构建计算模块的晶体管级/门级电路实现,根据晶体管/逻辑门电路拓扑连接结构与器件特性,对电路中部分影响较小的寄生参数进行化简省略,利用电路公式或简单电路方程解析解进行仿真评估。在拟合近似法中,仿真器设计人员首先使用SPICE或其他电路仿真工具对主要的电路模块进行仿真,通过调整器件参数、晶体管特征尺寸、模块参数(比如寄存器位宽)等,构建包含大量预仿真数据的查找表。在仿真存算一体架构时,仿真器将根据存算单元设计参数在预仿真数据查找表中进行查找,通过直接查表或近似拟合的方式仿真电路模块的性能表现。相比于解析分析法,拟合近似法在运行时具有更高效的仿真效率,但是由于仅提供预仿真结果和拟合方程作为用户接口,该方法难以根据用户需求定制化修改底层电路模块的电路设计细节。为保证硬件仿真的准确率,上述两种仿真方法均需要根据芯片实测数据对电路建模或拟合方程进行校准。

2.2 典型存算一体架构仿真器概述

表2列举并对比了典型的存算一体架构仿真器。需要指出的是,本文重点讨论面向神经网络算法以及模拟域矩阵向量乘的存算一体架构,针对其他存算一体实现方式的仿真器设计不在本文的讨论范围内(如面向存内逻辑计算的仿真器PIMSim[23]、面向近存储通用计算的仿真器Eva-cim[24]以及面向混合存储立方的仿真器HMC-sim[25])。从准确率仿真与架构性能仿真两个角度出发,表2根据如下9个方面对已有的仿真工具进行评估与分类。

1)仿真的计算类型:由于神经网络训练需要频繁更新忆阻器电阻值,这对忆阻器器件写寿命提出

表2 典型存算一体架构仿真器对比

Table 2 Comparison of typical PIM architecture simulators

	DL-RSIM[14]	Py-torX[15]	XPESim[16]	CIM-SIM[17]	DNN+NeuroSim[18]	DNN+NeuroSim V2.0[19]	MN-SIM[20]	MNSIM 2.0[21]	MNSIM-TIME[22]
仿真的计算类型	推理	推理	推理	推理	推理	训练	推理	推理	训练
是否支持准确率仿真	√	√	√	×	√	√	—	√	×
是否支持容错方法	×	√	×	×	×	×	×	√	×
是否支持硬件性能仿真	×	×	√	—	√	√	√	√	√
硬件仿真层级	×	×	电路器件级	系统指令级	电路器件级	电路器件级	电路级	架构级	架构级
器件可配置性	中	中	强	弱	强	强	中	中	中
硬件架构可配置性	中	中	中	弱	中	中	强	强	强
硬件仿真方法	×	×	解析分析法	指令分析法	解析分析法	解析分析法	解析分析法	拟合近似法	拟合近似法
是否支持软硬件协同优化	×	√	×	×	×	×	×	√	×

了较高要求,基于忆阻器的神经网络训练架构尚处在早期探索阶段。因此,目前主流的存算一体架构仿真器均面向神经网络推理计算,仅有DNN+NeuroSim V2.0[19]和MNSIM-TIME[22]对存算一体训练架构的性能仿真进行了初步研究。

2)是否支持准确率仿真:由于存算一体架构中存在诸多影响计算准确率的架构设计参数,大多数仿真器均会将准确率仿真作为其重要功能。发表于2016年的MNSIM[20]是第一款面向存算一体架构的行为级仿真器,其在准确率仿真方面仅关注到矩阵向量乘计算准确率,相关成果作为研究基础用于MNSIM 2.0[21]的算法准确率仿真中。由于神经网络训练的计算量远超于推理计算,存算一体训练架构的准确率仿真耗时可达数十小时,MNSIM-TIME[22]仅支持对硬件计算性能进行快速评估用以进行设计空间搜索与寻优。

3)是否支持容错方法:支持算法准确率仿真的仿真器均考虑到器件非理想因素与模数转换接口量化误差对于神经网络准确率的影响。除了对上述误差进行分析外,PytorX[15]和MNSIM 2.0[21]还提供了帮助存算一体架构恢复算法准确率的容错训练方法。PytorX[15]在仿真框架中引入了SAF校正算法和噪声注入的神经网络训练方法,在神经网络训练过程中将SAF错误和可能的IR压降建模为训练噪声,通过离线的带噪声训练使神经网络准确率不受非理想因素的干扰。MNSIM 2.0[22]则是将关注点放在ADC量化误差上,通过在训练阶段统计各层输出特征图的数据分布,采用ADC量化区间压缩与非线性量化方法,在有限的ADC量化精度下达到最高的算法准确率,缓解量化误差对准确率的影响。

4)是否支持硬件性能仿真:在典型的存算一体架构仿真器中,DL-RSIM[14]与PytorX[15]主要致力于解决存算一体架构中算法准确率的仿真问题,并未在其仿真流程中考虑硬件性能评估。CIM-SIM[17]在系统层级提出面向存算一体架构的指令集,将传统冯·诺依曼架构与存算一体架构结合起来,从指令及编译层面评估存算一体架构的计算性能,即计算延时(以时钟周期为单位)。除上述三者外,其他仿真器均支持“架构-电路-器件”的多层级硬件性能仿真。

5)硬件仿真层级:在支持硬件性能仿真的仿真器中,CIM-SIM[17]是唯一一款从系统指令层面对存算一体架构进行仿真的工具。CIM-SIM[17]为存算一体架构设计了包含输入端寄存器写入、忆阻器交叉阵列初始化、执行存内矩阵计算以及执行存内逻辑计算在内的多条微指令(nano instructions),并根据这些指令设计了存算一体架构处理内核(kernel)。微指令与处理内核的设计使CIM-SIM[17]可以在系统层面评估存算一体架构的功能正确性。XPESim[16]、DNN+NeuroSim[18]、DNN+NeuroSim V2.0[19]以及MNSIM[20]均从电路层面对存算一体架构进行仿真。这里的电路级仿真并非指利用电路矩阵求解进行的仿真,而是指这些仿真器中的基础电路模块均显式描述了门级/晶体管级电路的连接关系。以MNSIM[20]为例,MNSIM[20]中构建了逻辑门的基类,其他电路模块(如加法器)均是从逻辑门基类上根据逻辑门连接关系继承派生出的新类。各个类中均有刻画性能的成员,通过逐层访问调用实现层级化的性能评估。在这些电路级仿真器中,DNN+NeuroSim[18]与DNN+NeuroSim V2.0[19]支持更多类型的忆阻器件以及基于SRAM的存算一体,XPESim[16]对多比特RRAM器件不同阻值分布和非理想因素进行了详细探讨与分析。MNSIM 2.0[21]与MNSIM-TIME[22]从架构级对存算一体架构进行仿真,相比于其他电路级仿真器着重分析底层计算模块的电路实现,架构级仿真器重点考虑在运行神经网络算法时各个计算单元的连接交互关系以及数据流分析。以MNSIM 2.0[21]为例,该工具考虑了神经网络计算过程中的层间细粒度流水,即神经网络计算不再是逐层完成,当前一层计算产生足够多的中间数据时,后一层神经网络层即可同时开始计算,呈现出多个网络层以流水方式并行计算的特点。实验表明层间细粒度流水可以降低40%~60%的计算延时。

6)器件可配置性:器件可配置性体现在仿真器描述忆阻器器件行为的详细程度。CIM-SIM[17]具有最低的器件可配置性,由于该工具专注于指令集设计,其功能与仿真结果并不受底层器件的影响,因此该工具并未考虑使用可调的器件参数。DL-RSIM[14]、PytorX[15]、MNSIM[20]、MNSIM 2.0[21]与MN-

SIM-TIME[22]具备中等程度的器件可配置性，这些工具考虑到了器件开关比、忆阻器比特数、器件阻值分布与阻值扰动等器件参数，上述参数可以用以准确评估忆阻器计算功耗与计算准确率。XPESim[16]、DNN+NeuroSim[18]与DNN+NeuroSim V2.0[19]具有最高的器件可配置性，除了上述提到的器件特性外，这些工具还考虑了器件间扰动（device-to-device variation）、周期间扰动（cycle-to-cycle variation）以及器件阻值改变行为（如阻值非线性变化）等器件参数，这些参数可以更为精准地协助忆阻器件研究人员评估新型器件的计算能力。

7）硬件架构可配置性：与器件可配置性相似，硬件架构可配置性体现在用户使用仿真器时存算一体架构设计的可调整空间。系统级仿真工具CIM-SIM[17]采用了类ISAAC[10]形式的架构，其架构的可调整性主要体现在数据位宽以及寄存器宽度的调整，因此架构可配置性较差。DL-RSIM[14]、PytorX[15]、XPESim[16]、DNN+NeuroSim[18]与DNN+NeuroSim V2.0[19]的硬件可配置性适中，这些工具规定了架构的层级结构与连接关系，采用固定的算法映射方法，忆阻器交叉阵列尺寸、交叉阵列导通行列数等参数可以根据用户需求进行调整。架构级仿真器MNSIM[20]、MNSIM 2.0[21]与MNSIM-TIME[22]具有最高的硬件架构可配置性，这些工具自顶层架构到交叉阵列均为用户预留了一系列可以配置的接口，用户可根据需求自行改变某一层级某一硬件单元的数量与性能参数（如1个交叉阵列使用的ADC/DAC数量），也可以由用户自行定义神经网络算法在不同存算一体芯片与单元上的映射策略。通过改变架构设计参数，MNSIM 2.0[21]的内置架构抽象可以用来描述PRIME[9]、ISAAC[10]及其他多种存算一体架构的设计方案。

8）硬件仿真方法：2.1节介绍了存算一体架构最常使用的两种硬件仿真方法。具体到每个仿真器，其使用的硬件仿真方法与硬件仿真层级紧密相关。电路级仿真器通常使用解析分析法，架构级仿真器通常使用拟合近似法。系统级仿真器CIM-SIM[17]稍有区别，使用指令分析法进行仿真。具体而言，CIM-SIM[17]为存内计算定义了写入与计算微指令，在编译层面将源程序编译为普通处理器指令集中的指令和存算指令。将这些指令输入至仿真器中进行周期级仿真（cycle-level simulation），逐指令仿真确认计算行为的正确性并分析硬件性能。

9）是否支持软硬件协同优化：神经网络具有一定的容错能力，可以在保持准确率的前提下对网络权重进行定点量化与剪枝，从而达到更低的硬件计算开销。已有众多存算一体的相关工作采用软硬件协同优化的思想，在维持计算准确率的前提下降低硬件面积，提高计算速度[26-28]。将软硬件协同优化框架集成在存算一体架构仿真器中，可以在仿真阶段探索算法在存算一体架构上的最优性能。如前所述，PytorX[15]和MNSIM 2.0[21]在仿真器中集成了容错训练方法，亦可以视为支持软硬件协同优化。此外，MNSIM 2.0[21]在仿真器中集成了混合精度神经网络训练方法。在该训练方法中，不同网络层使用不同位宽对权重与特征图进行量化，从而在保持准确率的情况下降低非关键层的权重存储量，实现协同优化。

2.3 存算一体架构仿真器设计总结与展望

存算一体架构的仿真器发展呈现出百花齐放的局面，众多仿真器从不同层面解决了存算一体架构仿真耗时长的关键问题。从细分领域来看，各类仿真器呈现出差异化的趋势，例如DNN+ NeuroSim[18]考虑了诸多忆阻器器件及电路特性，适合于器件研究人员测试器件性能，而MNSIM 2.0[21]在架构层面提供了众多可配置的设计参数，同时包含算法训练及部署的完整流程，对于算法与架构的研究人员而言使用更为简单便利。

存算一体架构仿真器的发展仍存在诸多问题有待解决。首先，由于存算一体芯片的发展仍处在早期阶段，芯片工作主要停留在宏（macro）单元设计与验证层面，文献[29]是第1个在存算一体硬件上实现完整卷积神经网络计算的工作，但是该工作仅针对MNIST数据集与LeNet网络，数据量和硬件规模较小。因此，今天仍缺少大规模存算一体系统的实际部署与落地，这造成存算一体仿真器在进行架构仿真时缺少架构级性能参数的真值用于参照和校准。当前的仿真器可以在保证公平的前提下对不同架构、电路与器件进行相对优劣的对比，并用于设计空

间的搜索。但是如果希望存算一体架构仿真器在未来更为准确地指导芯片设计,仿真器设计和芯片设计必须相辅相成齐头并进。其次,目前的存算一体架构仿真器的侧重点不同,各个工具间的关联性仍未打通。例如,如何探索架构级仿真器与电路级仿真器在仿真准确率和仿真效率之间的权衡,或配合使用二者以在设计流程中实现高效准确的仿真,如何将系统级仿真器与架构级/电路级仿真器通过指令集设计这一"桥梁"连接在一起。仿真器的开发人员应在未来加强协作,规范化开源代码,标准化各层级仿真工具的输入输出接口,合作打造"系统-架构-电路-器件"的完整仿真工具链。最后,已有的存算一体架构仿真器主要面向卷积神经网络与深度神经网络两类算法,但是随着算法领域的不断发展,涌现出了一批如图神经网络(graph neural network, GNN)、自注意力网络(self-attention network)、强化学习(reinforcement learning, RL)等新型算法,如何为仿真器设计更为灵活的算法和架构描述接口,使其可以支持对新算法的仿真也是今天存算一体架构仿真器所需要改进和完善的方面。

3 结　论

本文在对存算一体架构仿真器仿真流程进行概述的基础上,对已有的典型存算一体架构仿真器进行了总结归纳。从仿真器的硬件仿真层级、器件与架构可配置性等9个角度,对比分析了代表性存算一体架构的相对优劣。

存算一体架构仿真器以层次抽象、化繁为简的方式,将复杂的大规模电路矩阵求解问题转换为简单电路模型的求解与硬件计算行为的分析,已有仿真器相比于SPICE电路仿真至少可达3个数量级的仿真效率提升。

展望存算一体架构仿真器的未来发展,我们认为仍有3方面关键问题有待解决:1)如何验证仿真器对于大规模存算一体架构的仿真准确率;2)如何打通不同层级仿真器的关联性,实现一套全流程、多层次的仿真工具链;3)如何设计与完善仿真器的算法和架构描述接口,使其可以快速适应不断涌现的新型算法和与之对应的新型存算一体架构设计。

参考文献

[1] HE K, ZHANG X, REN S, et al. Deep residual learning for image recognition[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016: 770-778.

[2] REDMON J, DIVVALA S, GIRSHICK R, et al. You only look once: Unified, real-time object detection[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016: 779-788.

[3] YIN W, KANN K, YU M, et al. Comparative study of CNN and RNN for natural language processing[J]. ArXiv Preprint, 2017, ArXiv: 1702.01923.

[4] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Imagenet classification with deep convolutional neural networks[J]. Advances in Neural Information Processing Systems, 2012, 25: 1097-1105.

[5] FEDUS W, ZOPH B, SHAZEER N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity[J]. ArXiv Preprint, 2021, ArXiv:2101.03961.

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

[7] WONG H S P, RAOUX S, KIM S B, et al. Phase change memory[J]. Proceedings of the IEEE, 2010, 98(12): 2201-2227.

[8] HUAI Y. Spin-transfer torque MRAM (STT-MRAM): Challenges and prospects[J]. AAPPS Bulletin, 2008, 18(6): 33-40.

[9] CHI P, LI S, XU C, et al. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory[J]. ACM SIGARCH Computer Architecture News, 2016, 44(3): 27-39.

[10] 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.

[11] ANKIT A, HAJJ I E, CHALAMALASETTI S R, et al. PUMA: A programmable ultra-efficient memristor-based accelerator for machine learning inference[C]. Proceedings of the Twenty-Fourth International Conference on

Architectural Support for Programming Languages and Operating Systems, 2019: 715-731.

[12] CHENG M, XIA L, ZHU Z, et al. Time: A training-in-memory architecture for memristor-based deep neural networks[C]. 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC), IEEE, 2017: 1-6.

[13] SONG L, QIAN X, LI H, et al. Pipelayer: A pipelined reram-based accelerator for deep learning[C]. 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA), IEEE, 2017: 541-552.

[14] LIN M Y, CHENG H Y, LIN W T, et al. DL-RSIM: A simulation framework to enable reliable ReRAM-based accelerators for deep learning[C]. 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), IEEE, 2018: 1-8.

[15] HE Z, LIN J, EWETZ R, et al. Noise injection adaption: End-to-end ReRAM crossbar non-ideal effect adaption for neural network mapping[C]. Proceedings of the 56th Annual Design Automation Conference 2019, 2019: 1-6.

[16] ZHANG W, PENG X, WU H, et al. Design guidelines of RRAM based neural-processing-unit: A joint device-circuit-algorithm analysis[C]. 2019 56th ACM/IEEE Design Automation Conference (DAC), IEEE, 2019: 1-6.

[17] BANAGOZAR A, VADIVEL K, STUIJK S, et al. Cim-sim: Computation in memory simuiator[C]. Proceedings of the 22nd International Workshop on Software and Compilers for Embedded Systems, 2019: 1-4.

[18] PENG X, HUANG S, LUO Y, et al. DNN+ NeuroSim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies[C]. 2019 IEEE International Electron Devices Meeting (IEDM), IEEE, 2019: 32.5. 1-32.5. 4.

[19] PENG X, HUANG S, JIANG H, et al. DNN+ NeuroSim V2.0: An end-to-end benchmarking framework for compute-in-memory accelerators for on-chip training[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, arXiv: 2003.06471.

[20] XIA L, LI B, TANG T, et al. MNSIM: Simulation platform for memristor-based neuromorphic computing system[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2017, 37(5): 1009-1022.

[21] ZHU Z, SUN H, QIU K, et al. MNSIM 2.0: A behavior-level modeling tool for memristor-based neuromorphic computing systems[C]. Proceedings of the 2020 on Great Lakes Symposium on VLSI, 2020: 83-88.

[22] QIU K, ZHU Z, CAI Y, et al. MNSIM-TIME: Performance modeling framework for training-in-memory architectures[C]. 2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS), IEEE, 2021: 1-4.

[23] XU S, CHEN X, WANG Y, et al. PIMSim: A flexible and detailed processing-in-memory simulator[J]. IEEE Computer Architecture Letters, 2018, 18(1): 6-9.

[24] GAO D, REIS D, HU X S, et al. Eva-cim: A system-level performance and energy evaluation framework for computing-in-memory architectures[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39(12): 5011-5024.

[25] LEIDEL J D, CHEN Y. Hmc-sim: A simulation framework for hybrid memory cube devices[J]. Parallel Processing Letters, 2014, 24(4): 1442002.

[26] YANG T H, CHENG H Y, YANG C L, et al. Sparse reram engine: Joint exploration of activation and weight sparsity in compressed neural networks[C]. Proceedings of the 46th International Symposium on Computer Architecture, 2019: 236-249.

[27] LIN J, ZHU Z, WANG Y, et al. Learning the sparsity for ReRAM: Mapping and pruning sparse neural network for ReRAM based accelerator[C]. Proceedings of the 24th Asia and South Pacific Design Automation Conference, 2019: 639-644.

[28] ZHU Z, SUN H, LIN Y, et al. A configurable multi-precision CNN computing framework based on single bit RRAM[C]. 2019 56th ACM/IEEE Design Automation Conference (DAC), IEEE, 2019: 1-6.

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