IMAGE LICENSED BY INGRAM PUBLISHING

Compute-In-Memory Technologies for Deep Learning Acceleration

Status and Prospect

DEEP LEARNING ACCELERATORS (DLAs) based on compute-in-memory (CIM) technologies have been considered promising candidates to drastically improve the throughput and energy efficiency for running deep neural network models. In this review, we analyze DLA designs reported in the past decade, including both fully digital DLAs and analog CIM based DLAs, to provide insights regarding the current status of CIM technologies and prospective of this emerging field. We observed that the reported CIM designs, even in their early research stage, do provide energy efficiency advantages from measured silicon data over digital DLA. Additionally,

Digital Object Identifier 10.1109/MNANO.2023.3340321
Date of current version: 21 February 2024

FAN-HUSAN MENG AND WEI D. LU

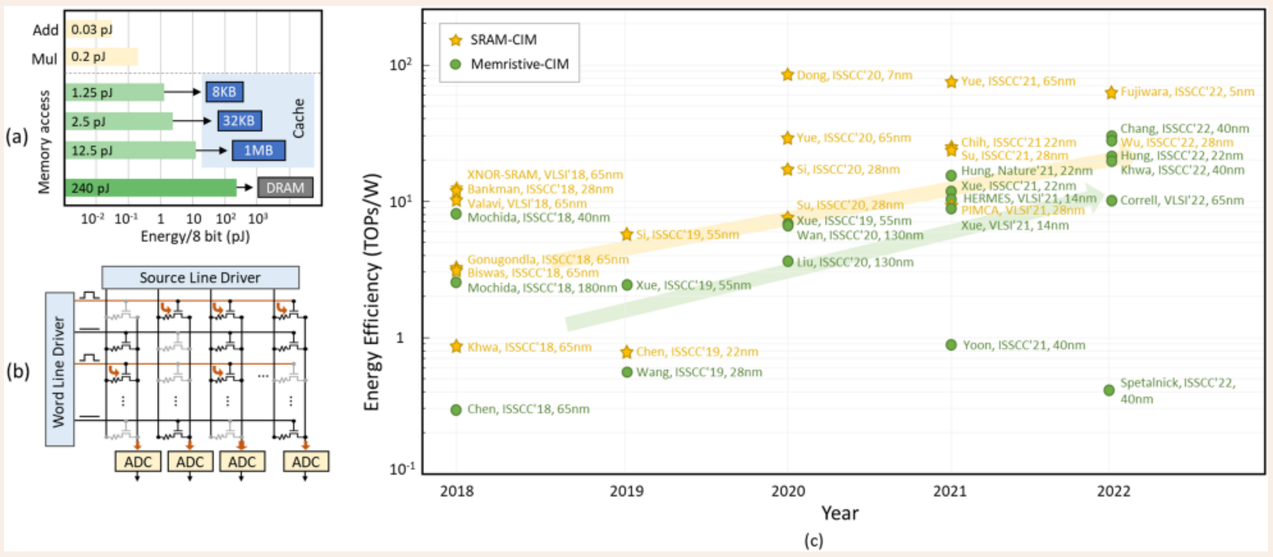

FIGURE 1 (a) Rough energy costs for various operations at 45 nm node, extracted from [24]. (b) Schematics of a CIM unit, including the memory array and periphery. (c) Reported TOPS/W of fully digital, SRAM based CIM, and memristive CIM systems.

it is revealed that the main advantage comes from completely eliminating the run-time DRAM access for weights. For performance benchmarks, we performed a top-down analysis using a generic DLA design and illustrated how fully-weight-stationary CIM DLAs, being no longer bounded by the memory bottleneck, offer large throughput advantages compared to traditional digital DLAs. The benchmark was performed by computing popular models deployed in Google's TPU accelerator.

INTRODUCTION

Hardwares designed specifically to accelerate deep neural networks (DNNs) have become a hot topic of interest in the past decade due to the success of deep learning [1], [2], [13], [14], [15]. DNNs models are largely based on multiply-and-add (MAC) operations with a massive number of parameters (weights) and variables (activations) [4]. Therefore, DLA designs usually consist of large amounts of MAC units or vector-matrix-multiplication (VMM) units to accelerate the computations and optimize data movement to minimize the cost of accessing the weights and activations included in DNN computation [16]. It is known that memory access costs for data movement are orders of magnitude larger than computation costs, and the energy costs grow with memory size [24]. For example, the estimated energy required for 8-bit arithmetic and various-sized memory access (normalized to 8-bit) in 45 nm at 0.9 V extracted from [24] is shown in Figure 1(a), illustrating the memory access problem in traditional computer architecture with separated centralized memory units and computing units.

To improve performance (e.g. throughput) and overcome the high energy cost that comes with data movements, compute-in-memory (CIM) based DLAs are proposed [22], [23]. Many VMM engine designs based on the CIM principle can be achieved with little adjustments in the peripheral circuits of existing memory arrays [17]. A simplified schematic of a CIM unit for computing VMM is shown in Figure 1(b). The matrix, represented as low-precision fixed point values is programmed on the memory cells with the input vectors, encoded as input voltages (amplitude or pulse width), and applied to the wordlines (WLs) of the memory array. The output vector of the VMM operation will then be the current readouts at the bitlines. The output is then typically converted back to digital form through analog-to-digital converters (ADCs). The memory used for CIM implementations can be separated roughly into two groups, mainstream SRAM and emerging memristive memory. Figure 1(c) shows the scaled energy efficiency (defined as Tera-Operations-per-Second (TOPS) per watt) of CIM based DLAs reported since 2018. An overall trend of incremental improvements in energy efficiency in both types of CIM can be observed.

SURVEY OF DLAS

In this review, we analyze silicon data reported in representative venues including International Solid-State Circuits Conference (ISSCC), Symposium on VLSI Technology and Circuits (VLSI), and relevant journals. The works included are separated into three groups, fully digital DLAs, SRAM-based CIM DLAs, and memristive-based CIM DLAs, as listed in Table 1. Digital-based DLAs were reported earlier, e.g., starting from around 2016, and CIM based DLAs were reported later around 2018. For a fair comparison of DLA prototypes, we collect CIM-DLAs from 2018 to 2022, and digital DLAs from 2016 to 2020, five years for each category. Although some works demonstrated acceleration for DNN training, in this analysis we only focus on the DNN inference acceleration.

DLA PERFORMANCE REPORTED WITH SILICON DATA

For simplicity, throughput data (Giga-Operations-per-Second, GOPS or

TABLE 1 List of DLA works included in the analysis.

	LABEL	TECH NODE	YEAR	MEMORY (KB)	PRECISION
Digital	Eyeriss, ISSCC'16, 65 nm	65 nm	2016	Global buffer: PE buffer: 73.5	16b
	Sim, ISSCC'16, 65 nm	65 nm	2016	38 SRAM	24b (16.8 format)
	Moons, VLSI'16, 40 nm	40 nm	2016	144 SRAM	Dynamic 1b × 16b
	ENVISION, ISSCC'17, 28 nm	28 nm	2017	144 SRAM	Dynamic 1xN - 16/N
	Desoli, ISSCC'17, 28 nm	28 nm	2017	Microcontroller: 128, SRAM: 4000	16b
	DNPU, ISSCC'17, 65 nm	65 nm	2017	280 (CNN core) + 10 (FC core)	4b, 16b
	Bong, ISSCC'17, 65 nm	65 nm	2017	160 T-SRAM	16b
	Yin, VLSI'17, 65 nm	65 nm	2017	348 SRAM	8b, 16b
	UNPU, ISSCC'18, 65 nm	65 nm	2018	256 SRAM	1 × 16 continuous activation
	Song, ISSCC'19, 8 nm	8 nm	2019	1568 SRAM	8b, 16b
	Lin, ISSCC'20, 7 nm	7 nm	2020	2176 SRAM	8b, 16b, FP16
SRAM CIM	Bankman, ISSCC'18, 28nm	28 nm	2018	264.5 (Weight) + 64 (activation)	Binary
	Gonugondla, ISSCC'18, 65 nm	65 nm	2018	16 SRAM (512×256 array)	8b
	Khwa, ISSCC'18, 65 nm	65 nm	2018	CIM SRAM: 0.5 (64×64 array)	1b
	Biswas, ISSCC'18, 65 nm	65 nm	2018	CIM SRAM: 2 (256×64 array)	1b to 7b
	Valavi, VLSI'18, 65 nm	65 nm	2018	295 SRAM (64×64 array)	1b
	XNOR-SRAM, VLSI'18, 65 nm	65 nm	2018	CIM SRAM: 2 (256×64 array)	Binary/Ternary
	Wang, ISSCC'19, 28 nm	28 nm	2019	CIM SRAM: 128 (128×256 array)	8b
	Si, ISSCC'19, 55 nm	55 nm	2019	CIM SRAM: 0.47	1b, 2b, 4b, 5b
	Su, ISSCC'20, 28 nm	28 nm	2020	CIM SRAM: 8 (512×128 array)	2b, 4b, 8b
	Si, ISSCC'20, 28 nm	28 nm	2020	CIM SRAM: 8	4b
	Dong, ISSCC'20, 7 nm	7 nm	2020	CIM SRAM: 0.5 (64×64 array)	4b
	Yue, ISSCC'20, 65 nm	65 nm	2020	Digital: 164, CIM SRAM: 0.5 (128×128 array)	2b, 4b, 6b, 8b
	Chih, ISSCC'21 22 nm	22 nm	2021	SRAM CIM: 2 KB (256×64 array)	4b, 8b
	PIMCA, VLSI'21, 28 nm	28 nm	2021	SRAM CIM: 432 KB	4b, 8b
	Yue, ISSCC'21, 65 nm	65 nm	2021	Digital: 294, CIM SRAM: 8	1b to 8b
	Su, ISSCC'21, 28 nm	28 nm	2021	CIM SRAM: 48	4b, 8b
	Wu, ISSCC'22, 28 nm	28 nm	2022	CIM SRAM: 128	4b, 8b
	Fujiwara, ISSCC'22, 5 nm	5 nm	2022	CIM SRAM: 8 (256×256 array)	8b
Memristive CIM	Chen, ISSCC'18, 65 nm	65 nm	2018	CIM RRAM: 128	Binary
	Mochida, ISSCC'18, 40 nm	40 nm	2018	CIM RRAM: 512(40 nm)	1b × 8b
	Mochida, ISSCC'18, 180 nm	180 nm	2018	CIM RRAM: 256(180 nm)	1b × 8b
	Xue, ISSCC'19, 55 nm	55 nm	2019	CIM RRAM: 128 (256×512 array)	16, 2b, 3b
	Chen, ISSCC'19, 22 nm	22 nm	2019	CIM RRAM: 128	Binary/Ternary
	Xue, ISSCC'20, 22 nm	22 nm	2020	CIM RRAM: 256 (512×512 array)	1b, 2b, 4b
	Liu, ISSCC'20, 130 nm	130 nm	2020	CIM RRAM: 19.85	1b × 3b
	Wan, ISSCC'20, 130 nm	130 nm	2020	CIM RRAM: 8 (256×256 array)	1b act, 7 level weight
	Xue, ISSCC'21, 22 nm	22 nm	2021	CIM RRAM: 512 (1024×512 array)	1b, 2b, 8b
	Yoon, ISSCC'21, 40 nm	40 nm	2021	CIM RRAM: 8	1b to 8b
	Hung, Nature'21, 22 nm	22 nm	2021	CIM RRAM: 512	8b
	HERMES Core, VLSI'21, 14 nm	14 nm	2021	CIM RRAM: 8 (512×512 array)	1b, 2b, 4b
	Xue, VLSI'21, 14 nm	14 nm	2021	CIM RRAM: 256 (512×512 array)	8b
	Hung, ISSCC'22, 22 nm	22 nm	2022	CIM RRAM: 1024 (1024×256 array)	1b to 8b
	Khwa, ISSCC'22, 40 nm	40 nm	2022	CIM RRAM: 256 (256×1024)	1b, 2b, 4b, 8b
	Spetalnick, ISSCC'22, 40 nm	40 nm	2022	CIM RRAM: 8	Binary
	Correll, VLSI'22, 65 nm	65 nm	2022	CIM RRAM: 8 (256×64 array)	8b × 4b
	Chang, ISSCC'22, 40 nm	40 nm	2022	SRAM: 768, CIM RRAM:2304	Binary

Tera-Operations-per-Second, TOPS) reported in various works are linearly scaled to $8b \times 8b$ activation × weight multiplication precision, which is typical for DNN inference [26]. where each MAC is considered as two operations (OPs): multiplication and add. However, some factors which could affect the performance should also be kept in mind. Firstly, a more precise evaluation should also consider output precision as a figure of merit of the VMM operation. For example, most CIM based DLAs truncate the output precision directly at each VMM unit to reduce the required ADC precision. However, this introduces larger errors compared to approaches that keep higher output precision at each VMM and truncate only after the final output is obtained. We do not take this effect into account while scaling the performance because many studies do not specify the output precision and only list drops in the DNN accuracy. Secondly, most CIM based DLAs, especially the earlier ones are based on low multiplication precision, for example, binary or ternary inputs and weights. For these cases, aggressively scaling up to $8b \times 8b$ multiplications might lead to large overestimation from overhead.

Throughput and power consumption are important for evaluating the performance of a DLA. The scatter plot of peak throughput vs. average power of the DLAs in Table 1 with reported GOPS and TOPS/W or power numbers are shown in Figure 2. For works without reported average power, we use reported TOPS/W and GOPS to extrapolate the power values. It can be observed that CIM works generally

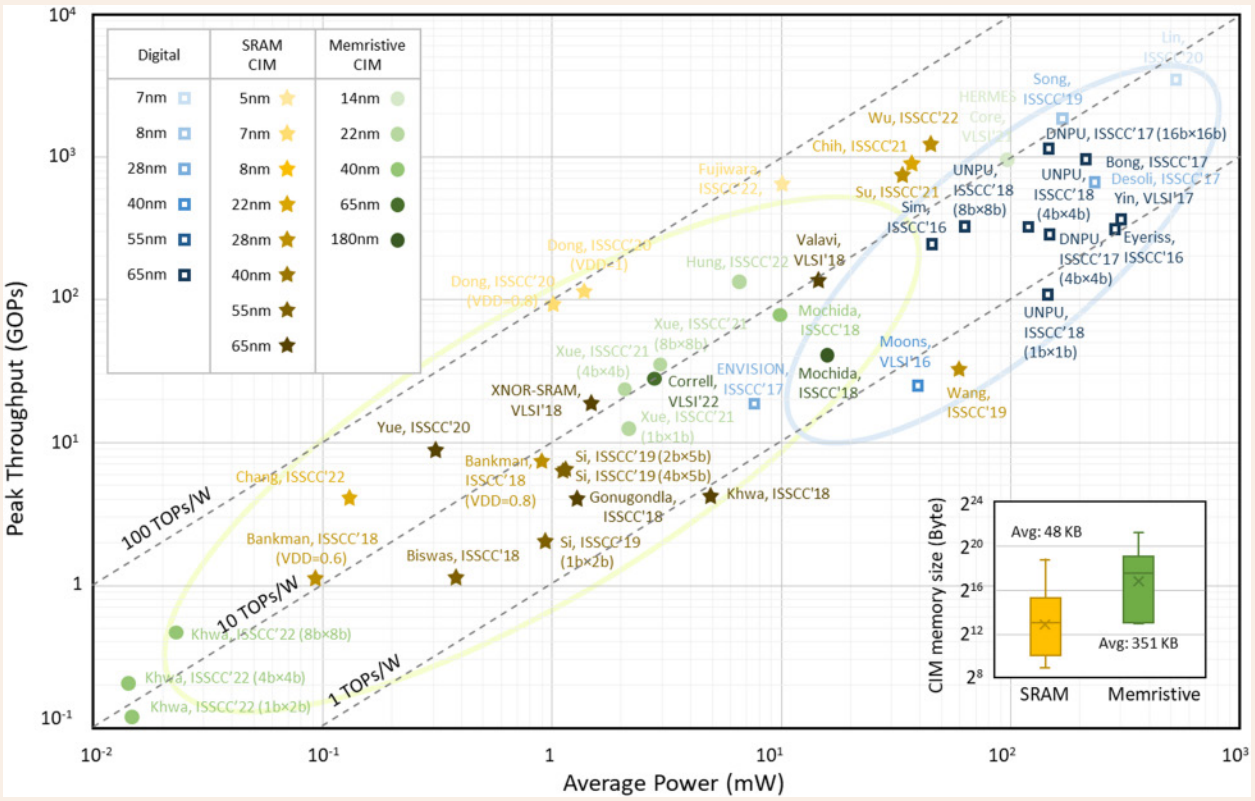

FIGURE 2 Scatter plot of throughput (GOPS) vs. power (mW) for digital, SRAM-CIM, and memristive-CIM based DLAs.

report higher TOPS/mW. On the other hand, the reported digital DLAs tend to offer higher power and higher throughput, which is expected since CIM based DLAs prototypes are relatively smaller in size as a relatively new technology. We also include the memory size of the CIM based DLAs in the inset of Figure 2. It is shown that memristive-based DLAs, although at a very early research stage, already demonstrated larger memory macro for CIM. This could be an indication of the higher memory density compared to SRAM provided by current technology.

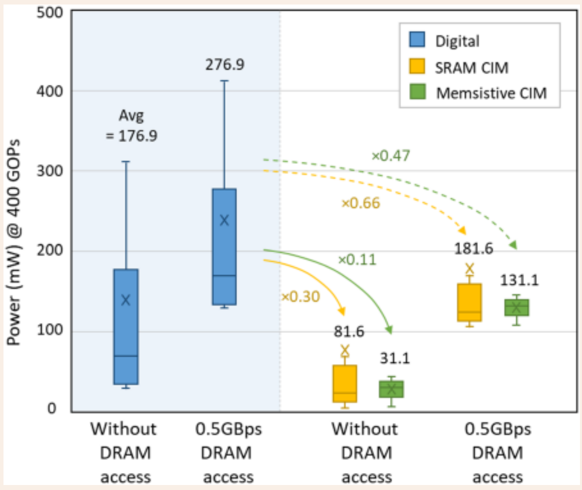

FIGURE 3 Box plots of estimated power consumption with and without 0.5 GBps DRAM access at 400 GOPS, for reported DLAs with multiplication precision above 8b.

EFFECTS OF DRAM ACCESS

ENERGY EFFICIENCY CONSIDERING DRAM ACCESS

The data evaluated in the previous section only consider the on-chip power consumption. With previous DLAs demonstrating smaller throughput, it's unlikely for the DLAs, both digital and CIM based, to keep all required data, including weight and intermediate activations, during DNN computations on chip. However, for DLAs of smaller scale, it is unrealistic to keep all weights and intermediate activations on-chip during computation, and frequent off-chip DRAM access is needed. In this section, we examine the effects of DRAM access for traditional digital as well as CIM architectures and discuss the importance for CIM based DLA to keep all weights stationary on-chip in order to exploit the CIM benefits.

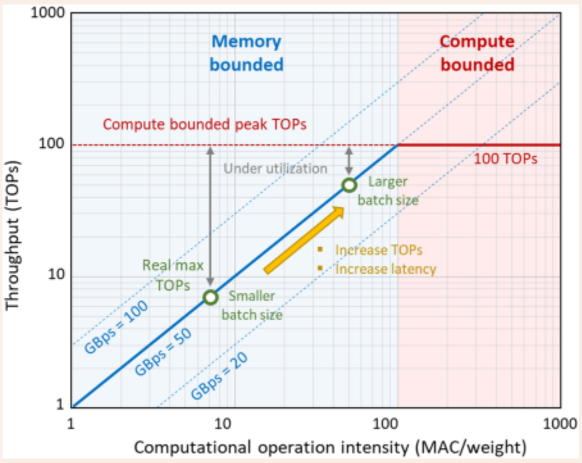

FIGURE 4 Roofline analysis of throughput vs. computational operation intensity.

In traditional digital architectures, separate memory units and computing units (CUs) are implemented, and weights are loaded on chip layer-by-layer from off-chip memory. Each time a set of new weights is loaded on chip, the corresponding activations are prepared.

In the following, to avoid large scaling error resulting from large overheads when scaling systems with very low precision weights, we use the total multiplication (i.e., activation × weight) precision as a parameter and only consider reported silicon data from implementations with total multiplication precision over 8b (e.g. $4b \times 2b$ or higher). Of these 54% of the GOPS reported falls between 10 to 400, with 29% under 10 GOPS and 17% above 400 GOPS. For comparison, we estimate the required power (mW) for 400 GOPS operations for each system, based on their reported performance and energy numbers. Our analysis shows digital DLAs, SRAM CIM, and memristive CIM to have an average power consumption of 176.9 mW, 81.6 mW, and 31.1 mW respectively, for the hypothetical 400 GOPS operations. Note this analysis does not consider DRAM access costs. The box charts of the data points are shown as the *without-DRAM access* items in Figure 3.

To estimate the required DRAM bandwidth for the 400 GOPS operations, we extract data reported in Eyeriss, which is a (scaled to $8b \times 8b$) 336 GOPS DLA with dataflow designs to optimize energy efficiency by limiting data movement, including minimizing off-chip DRAM access. With the MAC and DRAM access breakdowns for layers in AlexNet and VGG16 reported in Eyeriss, we estimated that the theoretical average DRAM access for the DLA to support 400 GOPS is 0.29 and 0.36 GBps for AlexNet and VGG16, respectively. Based on this analysis, we assume 0.5 GBps DRAM bandwidth is needed for a hypothetical 400 GOPS DLA for convolutional neural network (CNN) inference. The associated energy for the DRAM access is estimated as 100 mW [24]

With-DRAM access costs, in Figure 3 show that the power of digital DLA is estimated to be 1.6× of the chip-only power. If DRAM access cannot be eliminated in CIM systems either, the power of CIM based architectures increases to 2.2× and 4.2× of chip-only power; and becomes less impressive 0.66× and 0.47× of that of traditional digital DLA architectures, for SRAM-based and memristive-based CIM systems, respectively. However, if CIM based architecture is designed to completely eliminate DRAM access, the power consumption is brought down to 0.3× and 0.11× of the digital DLA systems, respectively, showing the potential for CIM systems to drastically improve energy efficiency and the necessity for CIM systems to have enough on-chip memory capacity to store all weights on chip.

THROUGHPUT AND THE DRAM MEMORY BOTTLENECK

In traditional digital architectures, separate memory units and computing units (CUs) are implemented, and weights are loaded on chip layer-by-layer from off-chip memory. Each time a set of new weights is loaded on chip, the corresponding activations are prepared. Batching of inputs is often employed to help reduce the required weight reads for computation. To model the performance considering such cross-chip data-movement of weights, the Roofline performance model used in high-performance computing (HPC) is adopted, following [18].

The roofline model plots throughput (TOPS) vs. operation intensity (OI). There are two types of OIs, one is the intrinsic OI of a model, defined as the number of MACs per weight. The intrinsic OI is thus a measure of the intrinsic reuse factor of a given weight. The other is the computational OI, which considers the intrinsic OI and the batching done for a given workload; The computational OI can be defined as, $\frac{MAC}{weight} \times batch$, i.e., the larger the batch size the larger the computational OI, leading to larger reuse of the weights. As illustrated in Figure 4, the roofline typically consists of two parts, a flat part on the right and a slanted part on the left. The flat part (red) marks the max throughput

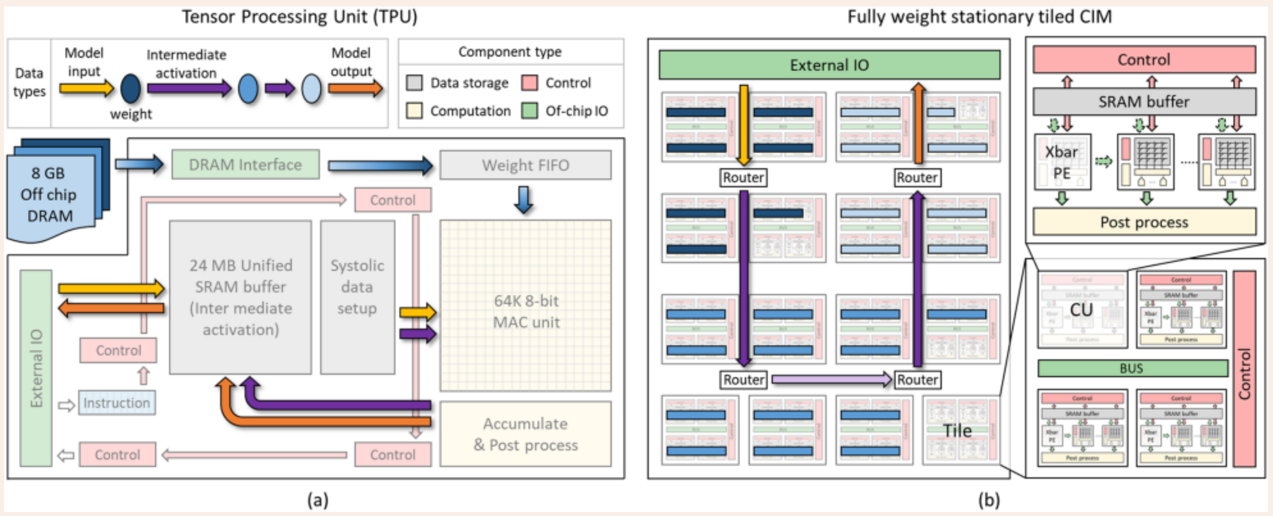

FIGURE 5 Schematic of high-level architecture of (a) TPU and (b) FWS-tiled CIM.

provided by the DLA compute unit(s) while the slanted part (blue) marks the memory-bounded throughput, i.e. when the CUs are starved of data.

A given hardware would have a specific roofline profile, and different workloads would fall in different regions of the roofline. For example, consider a perfect case corresponding to the hollow dots that sit on the roofline in Figure 4. If the throughput is limited by the memory bandwidth, the point could be pushed to the right and gets higher TOPS by increasing the batch size of a DNN workload; However, a larger batch size means longer latency, and inference applications usually have response time requirements [18] and therefore restricts the points to be pushed too far out right.

The memory-bounded slant in the roofline illustrates the fundamental difficulty of obtaining optimal throughput and response time in a traditional fully digital DLA for a targeted workload. Oftentimes the workload is memory bandwidth limited, preventing the system from achieving high utilization of the CU(s) and thus high throughput. As shown below, this problem could be eliminated completely with a fully-weight-stationary (FWS) CIM architecture.

CIM DLA PERFORMANCE BENCHMARKS: A TOP-DOWN ANALYSIS

Since most reported CIM prototypes are too small to map full CNN models, to analyze CIM DLA performance for practical CNN workloads, we performed top-down analysis using a hypothetical, generic CIM DLA system. The goal is to use real-world applications to benchmark CIM DLA performance and to help identify important future directions of CIM research. We choose the Tensor Processing Unit (TPU) hardware [18] as the digital baseline to compare against the CIM architecture, and use similar workloads deployed in Google's datacenter to evaluate the performance of the different architectures.

METHOD ARCHITECTURE

TPU is Google's inference-only DLA and has been deployed in its data-centers since 2015 [18]. TPU uses a typical weight stationary design with a large enough SRAM buffer to keep intermediate activations on chip. The heart of the TPU is a 256×256 8-bit integer systolic matrix multiplier array that offers a peak throughput of 92 TOPS, and a large 24 MB SRAM buffer for storing intermediate activations. A simplified illustration of the TPU is shown in Figure 5(a).

For the hypothetical CIM DLA, we assume it is based on the fully-weight-stationary (FWS) design principle with tiled CIM computation components that share data through a mesh network-on-chip (NoC). In this approach, the weights of a CNN model are fully stationary during inference without being rewritten or

TABLE 2 Comparison of TPU and FWS-CIM architectures.

	WEIGHT		ACTIVATION		COMPUTATION
	STORAGE	MOVEMENT	STORAGE	MOVEMENT	
TPU	• Off-chip DRAM • On-chip FIFO and buffer	• WS • Rewrite for each layer each batch	Full feature map in large centralized SRAM buffer	Between MAC array and buffer	Centralized MAC array temporally shared between NN layers
CIM	• On-chip CIM unit • Distributed	• FWS • No rewrites during runtime	Partial feature map in distributed SRAM buffers	Passed with hierarchical interconnect to downstream tiles	Distributed CIM units computing only parts of the NN each

TABLE 3 TPU and FWS-CIM specs used for analysis in this work.

CIM		TPU	
Frequency	200 MHz	Frequency	700 MHz
Tech node	22 nm	Tech node	28 nm
Crossbar (Xbar) size	256×256	MAC unit (8b×8b)	256×256
Cell/Weight	8	DRAM type	DDR3
Compute	32 Xbar/CU, 4 CU/tile	DRAM size	8 GB
# of Tiles	56	DRAM BW	34 GBps
Total RRAM	56 MB	Weight buffering	128 KB
SRAM buffer	14 MB (64 KB)	SRAM buffer	24 MB
Peak TOPs	92 TOPs	Peak TOPs	92 TOPS
Area	672 mm^2	Die area	< 331 mm^2

TABLE 4 Parameters of MLP0 and CNN0 for TPU bench marking, and relative pseudo models.

MODEL	WEIGHT	MAC	# OF LAYERS	BATCH SIZE	MAPPED WEIGHTS
MLP0 (TPU)	20 M	20 M	5	200	N/A
CNN0 (TPU)	8 M	2888 M	16	8	N/A
MLP (pseudo)	19.5 M	19.5 M	5	1	55.9M$^{1)}$
CNN (pseudo)	11.7 M	3277 M	14	1	55.8M$^{2)}$

1) Three copies of all weights
2) 192/192/48/48/12/12/3/3/1/1/1 copies for CONV layers; 1/1 copies for FC layer

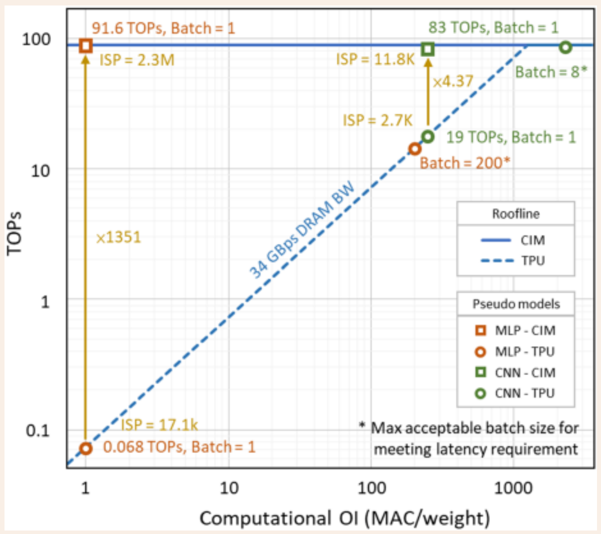

FIGURE 6 Roofline analysis of the generic CIM chip vs. TPU for the two DNN model workloads.

moved. The inputs are streamed into the tiles and the mesh NoC moves the resulting intermediate activations to the corresponding downstream tiles. The main differences between the two approaches are listed in Table 2.

In our analysis, we assume a generic CIM architecture, shown in Figure 5(b). A CU includes a set of crossbar (Xbar)-based processing elements (PEs) working together through a shared SRAM buffer. Here for brevity, we assume the crossbars are based on resistive random-access memory (RRAM), although other memristive crossbars can be used instead without losing generality. A number of CU are connected through a shared BUS, forming a tile; where a set of tiles form the complete CIM chip through NoC routers. The CIM chip is linked to the outside through an input/output (IO) interface. In order to facilitate a comparison between CIM and digital TPU, we examine prior works [19], [20], [21] and selected the architectural specs for the FWS-CIM as shown in Table 3. The peak TOPS is set to 92 TOPS to match the TPU baseline performance. Most RRAM CIM PE-related numbers are extracted from [Hung, ISSCC'22, 22nm], with TOPS/RRAM scaled up to match TPU throughput under acceptable total area. With these parameters, the 92 TOPS CIM chip requires 56 MB total on-chip RRAM.

NN WORKLOADS FOR EVALUATION

For evaluation, we consider the NN models reported in the TPU work which represent 95 % of the NN inference workload in Google's data-center at that time. The benchmarked models in TPU include 2 multi-layer perceptron (MLP) layers, 2 CNN layers, and 2 long short-term memory (LSTM) layers. The inference of MLP and CNN models has been extensively studied, therefore we chose one of each (*MLP0* and *CNN0*) for the analysis in this section. Although the exact model architecture was not released, some model information including weight size, MAC size, and total number of layers was reported, listed in Section 1 of Table 4. Since the exact NN model architecture is needed to estimate DLA performance, we created two pseudo models that match the model specifications of *MLP* and *CNN*, listed in the second section of Table 4.

- *MLP* - Five fully connected (FC) layers with input and output size of each layer ranging from 512 to 2816 and a total of 19.5 M weights.
- *CNN* - A VGG-like model with 12 3×3 convolutional (CONV) layers with filter count ranging from 64 to 512 and 2 FC layers. The model has a total of 11.7 M weights and 3277 M MAC.

EVALUATION

From the roofline analysis, we know that for traditional DLA with separate memory and computing units, the computational OI limits the maximum throughput a DLA can achieve. The computational OI is determined by the

TABLE 5 Comparison of different memory technologies. Data collected from [29], [30], [31].

	MAINSTREAM MEMORIES				EMERGING MEMORIES			
	SRAM	DRAM	NOR FLASH	NAND FLASH	FEFET	STT MRAM	PCRAM	RRAM
Cell area	>100 F^2	6–8 F^2	10 F^2	<4 F^2 (3D)	6–30 F^2	6–30 F^2	4–30 F^2	4–12 F^2
Bit/cell	1	1	~2	~3	1	1	~2	~2
Voltage	1 V	~1 V	10 V	10 V	3.5 V	1.5 V	3 V	3 V
Read time	~1 ns	~10 ns	~50 ns	~10 mu s	100 ns	~10 ns	100 ns	100 ns
Write time	~1 ns	~10 ns	~1 mu s	~100 mu s	100 ns	~10 ns	~50 ns	100 ns
Retention	N/A	~64 ms	~10 years	~10 years	~10 years	~10 years	~10 years	~10 years
Endurance	10^{16}	10^{16}	10^5	10^4	10^5	10^{15}	10^9	10^9
Write energy	~1 fJ	~10 fJ	~100 pJ	~10 fJ	~1 fJ	~0.1 pJ	~10 pJ	~1 pJ

batch size and the intrinsic OI of the targeted DNN model. The larger the batch size the better the TOPS (for memory-bound cases), and the lower the DRAM access power (for computation-bounded cases). However, in real-world applications, the latency of the computation largely affects a user's satisfaction with the service. Since batching increases latency, the number of batches selected in a workload during NN inference is thus normally restricted below a certain level. In [18] batch sizes of 200 and 8 are selected for *MLP0* and *CNN0* after latency considerations (including the server time as well as accelerator time).

For FWS-CIM, throughput can be maximized by leveraging pipelining. To avoid stalling in the pipeline stages, copies of the slower layers can be mapped to additional PEs to accelerate these slower layers. After mapping the models on chip with the optimal number of copies for different layers to maximize overall throughput, *MLP* and *CNN* require 55.9 M and 55.8 M weight storage, respectively.

RESULTS

As shown in Figure 6, the two models mapped on the hypothetical FWS-CIM architecture can utilize 91.6 TOPS and 83 TOPS, respectively, out of the total 92 theoretical TOPS with no batching (i.e. batch size of 1). For comparison, the max utilization for TPU is 0.068 TOPs and 19 TOPs, respectively, at batch size of 1. It is clear that both models are in the memory-bounded region and lead to significant underutilization of the TPU. After increasing the batch numbers as suggested by the TPU paper, the CNN model with a batch size of 8 can be moved to the computation-bounded region. However, the MLP model even with an aggressive batch size of 200 is still memory-bounded, as shown in Figure 6.

DISCUSSION

Our survey of reported silicon data and our top-down analysis of a generic CIM DLA verify that CIM based DLAs have clear advantages over traditional digital DLAs, especially for highly memory-bound NN workloads. This advantage comes in two folds, the power saved by eliminating DRAM access, and the higher throughput achieved by removing the constraint of the memory bottleneck. However, these advantages are only achieved if all the weight of a model can be stored on-chip. Figure 7 shows the number of parameters vs. ImageNet top-1 accuracy [27] of recent DNN models [1], [2], [3], [5], [6], [7], [8], [9], [10], [11], [12], [28]. It can be observed that state-of-the-art models can easily hold over tens of millions of parameters. In our previous mapping example shown in Table 4, it is shown that to minimize pipeline stalls duplication of weights needs to be employed for CIM, further increasing the needed on-chip memory size, as shown in Figure 6. The limited on-chip memory capacity, combined with the increasing model size, thus becomes a major challenge for CIM deployments, and storage density will be the most important spec when evaluating potential memory technologies for CIM applications.

Key specs of various memory technologies including mainstream memory and emerging memory are listed in Table 5. In general, emerging memories shown in the table can offer higher density compared to SRAM, and some have demonstrated multi-level-storage making them attractive for CIM. Another factor to consider is energy proportionality, i.e., power consumption with respect to the amount of work being processed. [25] shows that servers are rarely 100% busy;

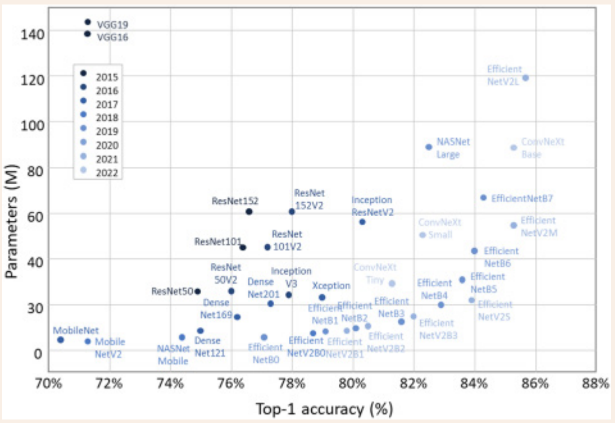

FIGURE 7 Weight size and ImageNet top-1 accuracy for popular CNN models.

therefore, good energy proportionality can help reduce the overall system power. With many emerging devices being nonvolatile, the power of unused PEs can be completely turned off. From these points one can see memristive CIMs are more attractive than SRAM CIMs, and material, device, and architecture co-optimizations that can further improve the CIM crossbar density, energy efficiency, and reliability will help bring the CIM DLAs to practice.

CONCLUSION

CIM based DLA is a promising field of study due to its potential of high throughput and high energy efficiency by eliminating the memory bottleneck. Reviewing recently published CIM silicon data verified that CIM DLAs indeed showed an advantage over digital DLAs even at this early development stage. When normalized to 400 GOPS operations, SRAM and memristive based CIM DLAs consume on average 81.6 mW and 31.1 mW, respectively, as opposed to 176.9 mW for conventional digital DLAs. However, noting the power consumption of required DRAM access is similar, the benefits of CIM-DLAs can only be fully achieved when DRAM access is completely eliminated.

To highlight the effects of DRAM bandwidth on DLA performance (throughput), we analyzed a hypothetical CIM DLA and benchmarked its performance against TPU for reported NN workloads. The elimination of DRAM bottleneck allows CIM DLA to achieve very high utilization even without batching. However, like energy efficiency gains, this performance gain can only be realized if all weights can be stored fully stationary on-chip. These results highlight the benefits of CIM DLA over digital DLAs and the importance of memory density when choosing memory technologies for CIM-DLA implementations.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under Grant CCF-1900675. The authors thank Dr. Zhengya Zhang, Xinxin Wang and Dr. Yuting Wu for insightful discussions.

ABOUT THE AUTHORS

Fan-husan Meng (fanhsuan@umich.edu) is with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA.

Wei D. Lu (corresponding author: (wluee@umich.edu)) is with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA.

REFERENCES

[1] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *Proc. Int. Conf. Learn. Representations*, 2015.

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[3] C. Szegedy et al., “Going deeper with convolutions,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[5] M. Tan and Q. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[6] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, “A convnet for the 2020s,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 11976–11986.

[7] A. G. Howard et al., “MobileNets: Efficient convolutional neural networks for mobile vision applications,” 2017, *arXiv:1704.04861*.

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.

[9] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, inception-ResNet and the impact of residual connections on learning,” in *Proc. AAAI Conf. Artif. Intell.*, 2017, vol. 31, pp. 4278–4284.

[10] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning transferable architectures for scalable image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8697–8710.

[11] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.

[12] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1251–1258.

[13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.

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

[15] 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.

[16] 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.

[17] P. Chi et al., “Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory,” *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 27–39, 2016.

[18] N. P. Jouppi et al., “In-datacenter performance analysis of a tensor processing unit,” in *Proc. 44th Annu. Int. Symp. Comput. Archit.*, 2017, pp. 1–12.

[19] A. Shafiee et al., “Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars,” *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 14–26, 2016.

[20] Q. Wang, X. Wang, S. H. Lee, F.-H. Meng, and W. D. Lu, “A deep neural network accelerator based on tiled RRAM architecture,” in *Proc. IEEE Int. Electron Devices Meeting*, 2019, pp. 14.4.1–14.4.4.

[21] X. Wang et al., “TAICHI: A tiled architecture for in-memory computing and heterogeneous integration,” *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 69, no. 2, pp. 559–563, Feb. 2022.

[22] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, “A survey of quantization methods for efficient neural network inference,” *Low-Power Comput. Vis.: Improving Efficiency Artif. Intell.*, Jun. 2021.

[23] A. Sebastian, M. Le Gallo, and E. Eleftheriou, “Computational phase-change memory: Beyond von neumann computing,” *J. Phys. D: Appl. Phys.*, vol. 52, no. 44, 2019, Art. no. 443002.

[24] M. Horowitz, “1.1 computing’s energy problem (and what we can do about it),” in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, 2014, pp. 10–14.

[25] L. A. Barroso and U. Hölzle, “The case for energy-proportional computing,” *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[26] R. Krishnamoorthi, “Quantizing deep convolutional networks for efficient inference: A whitepaper,” 2018, *arXiv:1806.08342*.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[28] A. Paszke et al., “Pytorch: An imperative style, high-performance deep learning library,” in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[29] M. Lanza et al., “Memristive technologies for data storage, computation, encryption, and radio-frequency communication,” *Science*, vol. 376, no. 6597, 2022, Art. no. eabj9979.

[30] Y. Xi et al., “In-memory learning with analog resistive switching memory: A review and perspective,” *Proc. IEEE*, vol. 109, no. 1, pp. 14–42, Jan. 2021.

[31] W. Chen, Z. Qi, Z. Akhtar, and K. Siddique, “Resistive-ram-based in-memory computing for neural network: A review,” *Electronics*, vol. 11, no. 22, 2022, Art. no. 3667.

N