Systolic-RAM: Scalable Direct Convolution Using In-Memory Data Movement

Jacob N. Rohan, *Graduate Student Member, IEEE*, and Jaydeep P. Kulkarni, *Senior Member, IEEE*,

***Abstract*—A 12.8-kbit static random access memory (SRAM) is demonstrated in 40-nm CMOS for signed 8-bit convolution in-memory. While conventional compute-in-memory (CIM) approaches rely on the indirect convolution algorithm, the proposed "Systolic-RAM" performs a form of direct convolution which eliminates the need for data duplication and near-memory registers. To achieve this, an in-memory data pipeline is employed to move data within the array and mimic the physical movement of the convolution kernel. In between data movement cycles, back-end-of-line (BEOL) structures perform charge-domain vector-matrix multiplication (VMM). The indirect convolution algorithm is illustrated, and the supporting circuits are presented in detail. Quantized neural network training methods are also employed to achieve test accuracy close to that of a floating-point network. The demonstrated array is configured for a 5 × 5 kernel, achieves 175(113) peak (continuous) multiply accumulate (MAC) operations per clock cycle and consumes 3.0 mW at 100 MHz.**

***Index Terms*—Analog computing, artificial neural networks, convolution, random access memory.**

I. Introduction

Data movement significantly impairs power performance in von Neumann systems when large amounts of data are exchanged between computer memory and processing units (referred to as the memory wall bottleneck). Compute-in-memory (CIM) approaches attempt to reduce data movement energy and latency overheads by performing key computations in parallel within the memory array (Fig. 1). Although adequate bit-resolution is commonly considered a leading measure of CIM performance [1], few works have elaborated on the data movement power, duplication, and restructuring required to realize convolution within CIM macros [2]. Duplication occurs since the indirect convolution method was tailor-made for vector-matrix multiplication (VMM)-based accelerators and uses an image-to-column (IM2COL) transformation [3]–[5]. This means conventional methods duplicate and transform the data rather than physically adopting the concept of a sliding kernel (the stride) within hardware. This duplication requires significant data caching at the CIM array periphery to support peak throughput. Other works have shown that a careful implementation of direct convolution can outperform indirect convolution at a system level [6]. The data overhead of indirect convolution becomes more detrimental for large kernels. For a 2-D K-by-K kernel with stride $= 1$, each input pixel will belong to K^2 unique stride locations and would therefore be duplicated in K^2 columns of the resulting IM2COL matrix. This problem is further exacerbated for n-dimensional convolution which requires $O(K^n)$ duplication. As an example, 2-D convolution using a large kernel (such as 11-by-11) will require data to be duplicated (>100 times) within an activation-stationary CIM array [Fig. 1(c)], significantly increasing data movement and cache requirements. When convolution is performed in a weight-stationary crossbar array, the overhead

Manuscript received January 19, 2022; revised April 3, 2022; accepted April 20, 2022. Date of publication May 20, 2022; date of current version May 31, 2022. This work was supported in part by the Intel Corporation and Micron Technology Inc. This article was approved by Associate Editor Shigeki Tomishima. *(Corresponding author: Jacob N. Rohan.)*

The authors are with the Electrical and Computer Engineering Department, The University of Texas at Austin, Austin, TX 78712 USA (e-mail: jacobnrohan@utexas.edu; jaydeep@austin.utexas.edu).

Digital Object Identifier 10.1109/LSSC.2022.3173560

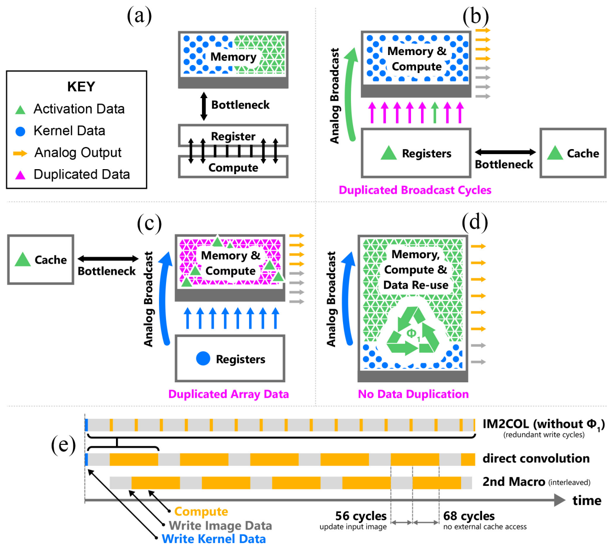


Fig. 1. Representation of several computation methods. (a) Conventional computers are restricted by memory bus bandwidth. (b) and (c) CIM methods aim to eliminate this bottleneck by performing computation in the memory array, but may require data duplication to perform convolutions. (d) Systolic-RAM mitigates this issue by performing both compute and data movement within the array. (e) Without in-memory data movement (ϕ_1 phase) throughput and memory bandwidth are significantly impaired.

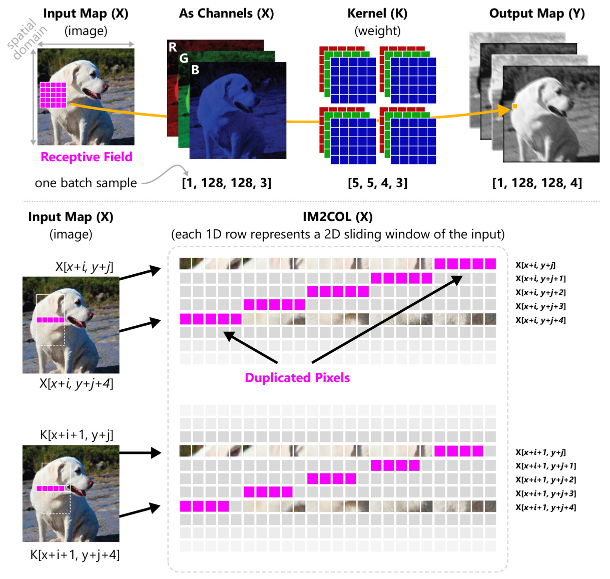


Fig. 2. (TOP) Representation of 2-D convolution in neural networks using a 128 × 128 image with three input channels and four output channels. (BOTTOM) Overlapping stride regions cause duplicated data in IM2COL matrix.

is realized as duplicated broadcast cycles at the array periphery [Fig. 1(b)] which may occur several clock cycles apart. In both cases, this duplication and circuit overhead should be considered as

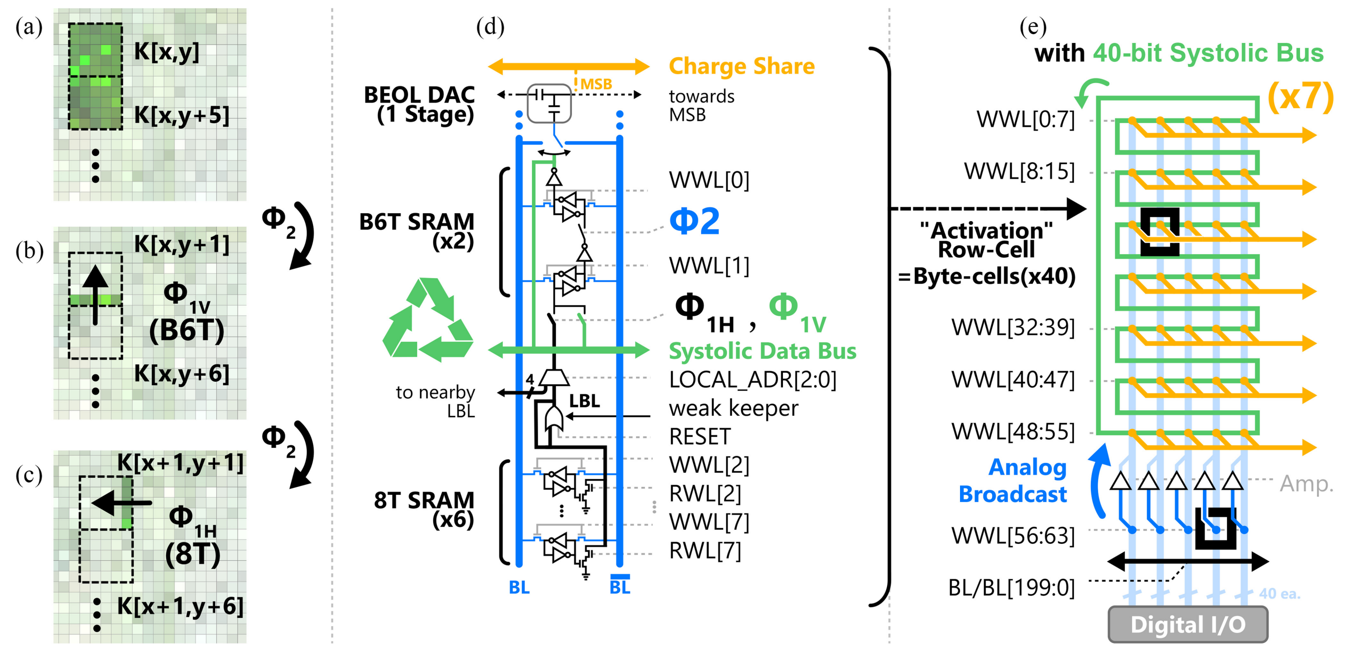


Fig. 3. Essential phases (ϕ_{1V}, ϕ_{1H}, ϕ_2) and corresponding circuit organization

a substantial factor in determining the energy efficiency and scalability of convolution in-memory. Systolic-RAM is unique: rather than duplicating data, required data is physically moved within the array. Since the in-memory data bus scales with array size, the total volume of data moved in a single clock cycle scales proportionally with array area and compensates for the required bandwidth. The Systolic-RAM test chip presented may still operate using indirect convolution and IM2COL. However, throughput and bandwidth are significantly improved when the proposed data movement in-memory (ϕ_1 phase) is used [Fig. 1(e)].

II. Convolution as Matrix Multiplication

Multiply accumulate (MAC) operations are fundamental component in artificial neural networks. The basic MAC operation involves *multiplying* two values and *adding* the product to an accumulator. Convolution is a unique process of data filtering which may be applied over a variety of domains, both continuous and discrete [7]. While 1-D, 3-D, and even 4-D convolution have been shown to be useful in machine learning [8] neural networks for image processing necessitates convolution to be applied over the discrete 2-D spatial domain. In discrete 2-D convolution, a 2-D kernel is translated over a 2-D input activation and incremental MAC products form each pixel value in the output

$$Y_{x,y} = K * X = \sum_{i=0}^{D-1} \sum_{j=0}^{D-1} X_{x+i,y+j} \cdot K_{i,j}. \tag{1}$$

This process provides location invariant feature detection unlike fully connected layers which do not adopt the principle of a sliding weight. For any given pixel at the output, a receptive field is formed as a region of the input activation covered by the kernel's finite span (Fig. 2). Both the size of this receptive field and the total number of MAC computations required for each output pixel are proportional to the size of the 2-D kernel. When 2-D convolution is used in a convolutional neural network (CNN), channels are also incorporated to retain a greater amount of information between layers (2). This is especially important as pooling layers repeatedly reduce the size of the activation's spatial domain

$$\begin{aligned} Y_{x,y,c_{out}} &= K * X \\ &= \sum_{c_{in}} \sum_{i=0}^{D-1} \sum_{j=0}^{D-1} X_{x+i,y+j,c_{in}} \cdot K_{i,j,c_{out},c_{in}}. \end{aligned} \tag{2}$$

Discrete convolution may be computed indirectly to be compatible with graphics processing units (GPUs) and other custom accelerator which support VMM. When indirect convolution is applied, the operands (K and X) are transformed (into A and B) to be compatible with general matrix-matrix multiplication

$$Y_{z,w} = AB = \sum_{m} a_{z,m} \cdot b_{m,w}. \tag{3}$$

This involves flattening every receptive field into a single row of the IM2COL matrix. In this way, data in overlapping stride regions will be duplicated (Fig. 2). Thus, there is a need for future CIM systems to efficiently reuse data.

III. Systolic-RAM Operation

Fundamentally, the test chip presented is a 25-by-8 multiplicative digital-to-analog converter (MDAC) array positioned in the back-end-of-line (BEOL) above a custom 200-by-64 bit static random access memory (SRAM). Thus, each MDAC has access to the bit-line data and an additional 64-bits of local memory [Fig. 3(d)]. Without in-memory data movement, simple VMM can be performed using an 8-bit 25-element vector with a 25-by-7 matrix; this could represent a blocked region of an FCN layer [9], or the dot product of the 5 × 5 kernel at seven unique rows of an IM2COL matrix. In both cases, the array is significantly bandwidth limited.

A. Data Movement Phases

Systolic-RAM computes convolutions without data duplication by recycling data between neighboring SRAM bit-cells. The process consists of two alternating phases: 1) ϕ_1 (data movement) and 2) ϕ_2 (compute). Fig. 3(a) illustrates how adjacent stride-regions are computed simultaneously. After computation, the vertical stride is

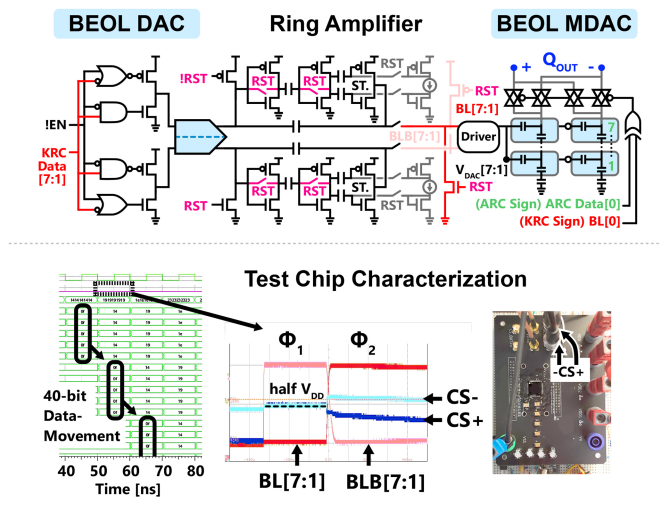


Fig. 4. (TOP) Schematic for ring amplifier (BOTTOM) oscilloscope waveform showing interleaving ϕ_1 and ϕ_2 phases and resulting CS MAC output.

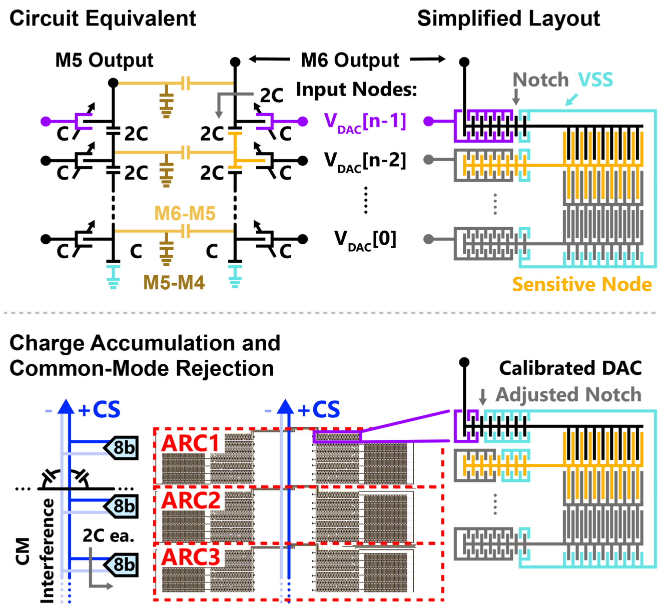


Fig. 5. (TOP) Circuit and layout for the BEOL C2C ladder MDAC. (BOTTOM) Notches are tuned to improve linearity and several DACs are combined to perform row-wise charge sharing.

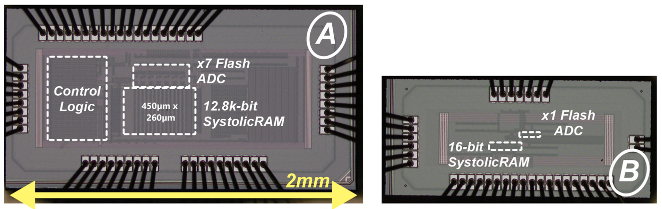


Fig. 6. Test chips for (a) system-level and (b) isolated measurements.

achieved in a ϕ_{1V} phase by cycling data through buffered-6T (B6T) bitcells to exchange K pixels from one stride location to the next [Fig. 3(b)] while reusing $K \times (K-1)$ pixels from the previous computation. When a horizontal kernel translation is required [Fig. 3(c)], ϕ_{1H} phase is used to insert new data from 8T bitcells into the B6T datapath. Fig. 3(d) illustrates both these translations within the B6T/8T cell structures.

This digital data movement allows Systolic-RAM to perform several MAC products every clock cycle. Seven unique 5×5-pixel regions of the input image are computed simultaneously for a total of 175 operations per cycle (Ops/Cy). This corresponds to the seven charge-share (CS) lines in Fig. 3(e). Total effective Ops/Cy varies based on application. For example, configuring this design for a ResNet layer (which requires padding for input/output sizes to be identical) achieves a continuous 113 [Ops/Cy] (24 025 effective operations in 155 compute cycles + 56 digital write cycles) when zero-padded for 31×31 input/output images. The design is scalable such that additional groups of word lines provide additional parallel computation.

B. Computation Phase

Analog MAC is achieved using MDAC capacitor-2capacitor (C2C) structures positioned in the BEOL above the array. These structures require no significant silicon-area overhead but are susceptible to nonideal analog effects, such as noise, parasitic interference, and nonlinearity. In the ϕ_2 (compute) phase, the kernel data is broadcast as an analog differential bit-line voltage using ring amplifiers to modulate all of the MDACs connected to the bit lines (Fig. 4). The MDAC's inputs are selectively switched to either bit-line based on the digital data in the bitcells. The resulting output charge produced is proportional to the product of signed 8-bit kernel and signed 8-bit input data. While the differential analog datapath rejects common-mode interference, parasitic capacitance in the MDAC results in significant nonlinearity (Fig. 5). To mitigate this effect, 3-D parasitic extraction is performed and notches in the MOM-CAP structure are adjusted at design time to reduce input capacitances and improve the final output linearity [10], [11]. In this way, linearity is achieved at the expense of attenuation.

IV. Measured Results

Two test chips were fabricated to validate this work (Fig. 6). The measured multiplication characteristics [Fig. 7(a)] demonstrate a 74% reduction in worst-case DAC DNL with only 45% reduction in amplitude. Least squares regression was used to extract the relative significance of each bit and clearly demonstrate the tradeoff between DAC linearity and attenuation [Fig. 7(b)]. Noise and nonlinearity with respect to input and weight were modeled as a differential convolution layers in Pytorch to match these characteristics. A gradient-blocking technique for quantization was used for autograd calculation and network retraining [12]. Pretrained ResNet-18 CNNs using float32 demonstrated 91.9% test accuracy on CIFAR-10 dataset [4], [13]. Immediately after 8-bit quantization, test accuracy was 90.3% using the calibrated MDAC and 81.6% with the uncalibrated MDAC. After retraining for three epochs, test accuracy recovered to 91.6% (0.3% below float32 baseline) with calibration but only 86.2% for the uncalibrated MDAC [Fig. 7(c)]. The effect of this nonlinearity on image quality is illustrated [Fig. 7(d)] and the recorded nerual network mispredictions for each method are shown [Fig. 7(e)].

SystolicRAM performs convolution at 14.4 bit-TOPS/W for 100 MHz, 1.1 V (Fig. 8). We found the largest contributor of power in SystolicRAM to be the ring amplifier topology chosen since the ϕ_1 reset (RST) phase requires the input and output of inverter-like structures to be shorted together (Fig. 4). Changing devices in this design to have a high-threshold voltage (HVT) is estimated to yield a static power reduction of 25× for digital elements (logic and bitcells) and

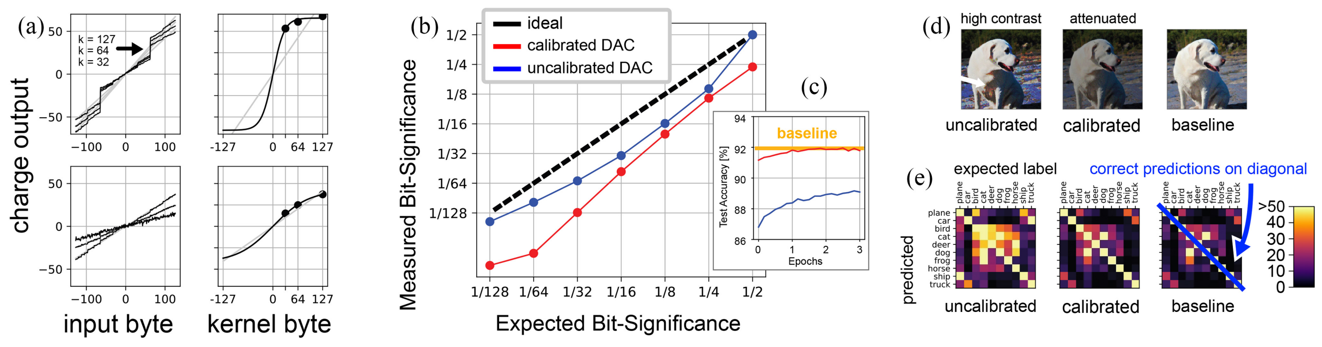
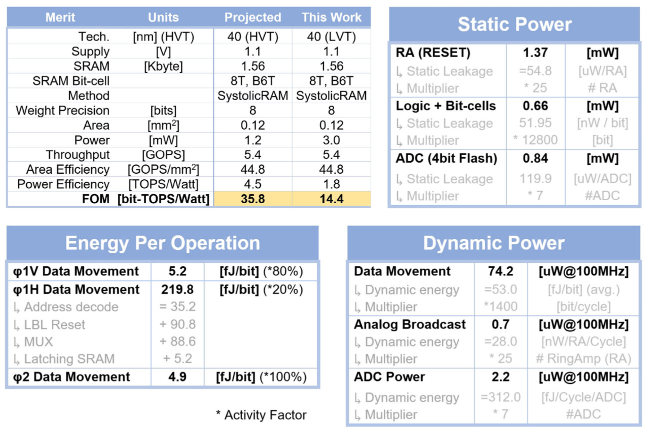


Fig. 7. (a) and (b) Linearity improvement using DAC calibration. (c) Neural network retraining methods recover test accuracy when retraining with linear calibrated DAC characteristics. (d) Effect of nonidealities on test image. (e) ResNet-18 mispredictions.



Merit	Units	Projected	This Work
Tech.	[nm] (HVT)	40 (HVT)	40 (LVT)
Supply	[V]	1.1	1.1
SRAM	[Kbyte]	1.56	1.56
SRAM Bit-cell		8T, B6T	8T, B6T
Method		SystolicRAM	SystolicRAM
Weight Precision	[bits]	8	8
Area	[mm^2]	0.12	0.12
Power	[mW]	1.2	3.0
Throughput	[GOPS]	5.4	5.4
Area Efficiency	[GOPS/mm^2]	44.8	44.8
Power Efficiency	[TOPS/Watt]	4.5	1.8
FOM	**[bit-TOPS/Watt]**	**35.8**	**14.4**

Static Power		
RA (RESET)	**1.37**	**[mW]**
↳ Static Leakage	=54.8	[uW/RA]
↳ Multiplier	* 25	# RA
Logic + Bit-cells	**0.66**	**[mW]**
↳ Static Leakage	51.95	[nW / bit]
↳ Multiplier	* 12800	[bit]
ADC (4bit Flash)	**0.84**	**[mW]**
↳ Static Leakage	119.9	[uW/ADC]
↳ Multiplier	* 7	#ADC

Energy Per Operation		
φ1V Data Movement	**5.2**	**[fJ/bit]** (*80%)
φ1H Data Movement	**219.8**	**[fJ/bit]** (*20%)
↳ Address decode	= 35.2	
↳ LBL Reset	+ 90.8	
↳ MUX	+ 88.6	
↳ Latching SRAM	+ 5.2	
φ2 Data Movement	**4.9**	**[fJ/bit]** (*100%)

* Activity Factor

Dynamic Power		
Data Movement	**74.2**	**[uW@100MHz]**
↳ Dynamic energy	=53.0	[fJ/bit] (avg.)
↳ Multiplier	*1400	[bit/cycle]
Analog Broadcast	**0.7**	**[uW@100MHz]**
↳ Dynamic energy	=28.0	[nW/RA/Cycle]
↳ Multiplier	* 25	# RingAmp (RA)
ADC Power	**2.2**	**[uW@100MHz]**
↳ Dynamic energy	=312.0	[fJ/Cycle/ADC]
↳ Multiplier	* 7	#ADC

Fig. 8. Summary of system performance and detailed energy breakdown.

2× for analog components resulting in a projected FOM of 35.8 bit-TOPS/W. The proposed approach can improve data/energy efficiency, bit-precision, and supported kernel size of VMM macros used for convolution computations.

V. Conclusion

Systolic-RAM demonstrates the first in-memory direct convolution engine as an all-in-one approach to data-efficient convolution. The $O(K^2)$ duplication required by previous methods is eliminated, effectively reducing this system's memory requirement by 25×. Systolic-RAM makes good use of BEOL wiring for data movement and signed analog computation, and demonstrates the importance of DAC calibration to achieve good neural network test accuracy. Unlike conventional CIM methods which solely perform computation, Systolic-RAM demonstrates the benefit for emerging memories to support data movement to more efficiently match a data's shape to its algorithm.

Acknowledgment

The authors would like to thank TSMC for test chip fabrication.

References

[1] T.-J. Yang and V. Sze, "Design considerations for efficient deep neural networks on processing-in-memory accelerators," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, 2019, pp. 22.1.1–22.1.4.

[2] M. Cho and D. Brand, "MEC: Memory-efficient convolution for deep neural network," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 815–824.

[3] M. Dukhan, "Indirect deconvolution algorithm," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, Los Alamitos, CA, USA, May 2020, pp. 922–926. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/IPDPSW50202.2020.00154

[4] A. Paszke *et al.*, "Automatic differentiation in PyTorch," in *Proc. NIPS Workshop Autodiff*, 2017, pp. 1–4. [Online]. Available: https://openreview.net/forum?id=BJJsrmfCZ

[5] A. Biswas and A. P. Chandrakasan, "CONV-SRAM: An energy-efficient SRAM with in-memory dot-product computation for low-power convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 217–230, Jan. 2019.

[6] J. Zhang, F. Franchetti, and T. M. Low, "High performance zero-memory overhead direct convolutions," in *Proc. ICML*, 2018, pp. 1–10.

[7] D. R. Fannin, R. E. Ziemer, and W. H. Tranter, *Signals and Systems: Continuous and Discrete*. Harlow, U.K.: Pearson, 1998.

[8] C. Choy, J. Gwak, and S. Savarese, "4D spatio-temporal ConvNets: Minkowski convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 3070–3079.

[9] P. Labini, M. Bernaschi, F. Silvestri, and F. Vella, "Blocking techniques for sparse matrix multiplication on tensor accelerators," Feb. 2022, *arXiv:2202.05868*.

[10] H. Balasubramaniam, W. Galjan, W. H. Krautschneider, and H. Neubauer, "12-bit hybrid C2C DAC based SAR ADC with floating voltage shield," in *Proc. 3rd Int. Conf. Signals Circuits Syst. (SCS)*, 2009, pp. 1–5.

[11] P. Harpe, "A 0.0013mm2 10b 10MS/s SAR ADC with a 0.0048mm2 42dB-rejection passive FIR filter," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, 2018, pp. 1–4.

[12] C. N. Coelho *et al.*, "Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors," *Nat. Mach. Learn.*, vol. 3, pp. 675–686, Jun. 2021.

[13] A. Krizhevsky, V. Nair, and G. Hinton. "CIFAR-10." [Online]. Available: http://www.cs.toronto.edu/ kriz/cifar.html (Accessed: Jul. 2021).