Hybrid Analog-Digital In-Memory Computing

Muhammad Rashedul Haq Rashed
Electrical and Computer Engineering
University of Central Florida
Orlando, USA
rashed09@knights.ucf.edu

Sumit Kumar Jha
Computer Science Department
University of Texas at San Antonio
San Antonio, USA
sumit.jha@utsa.edu

Rickard Ewetz
Electrical and Computer Engineering
University of Central Florida
Orlando, USA
rickard.ewetz@ucf.edu

***Abstract*—Today's high performance computing (HPC) systems are limited by the expensive data movement between processing and memory units. An emerging solution strategy is to perform in-memory computing (IMC) using non-volatile memory. However, state-of-the-art in-memory computing paradigms fail to simultaneously deliver high precision and high energy-efficiency. Analog in-memory computing is extremely energy-efficient but inherently vulnerable to errors. In contrast, digital in-memory computing based on Boolean logic is robust to errors but less energy-efficient. In this paper, we propose a new paradigm called hybrid analog-digital in-memory computing. The paper also proposes the associated in-memory computing platform and design automation tool chain needed to perform computation using the paradigm. The paradigm is capable of performing matrix-vector multiplication with both high energy-efficiency and precision. The key idea of the paradigm is to first decompose the most significant bits (MSBs) of the desired computation into Boolean functions and the least significant bits (LSBs) into matrix-vector multiplication operations. Next, the operations are mapped to digital and analog in-memory computing hardware, respectively. The proposed paradigm is evaluated using applications from the domains of structural engineering, mathematics, and statistics. Compared with analog in-memory computing, the proposed paradigm is capable of meeting the constraints on the computational accuracy. Compared with digital in-memory computing, systems, power, speed, and area are respectively improved with 2.44X, 2.45X and 2.32X.**

I. Introduction

The simulation of complex physical systems is integral to predicting and mitigating the impact of catastrophic events. Complex physical systems within high-energy physics [22], weather forecasting [21], and biology [25] are modeled using systems of partial differential equations (PDEs). These models are commonly required to be simulated for months using high performance computing (HPC) systems. The increasing demand for large scale simulation is putting undue pressure on the underlying computational substrates [8]. Unfortunately, it is notoriously difficult for HPC systems based on von-Neumann architecture to handle exascale or even petascale data. Mainly, due to the separation of memory and computing units, which translates into power hungry and bandwidth limited data transfer [30].

An emerging solution strategy is to perform in-memory computing using emerging non-volatile resistive devices. Non-volatile resistive devices are two terminal devices with programmable resistance, which includes resistive random access memory (ReRAM), phase change memory (PCM), spin-transfer torque magnetic random access memory (STT-MRAM). By integrating the devices in dense crossbar arrays, the amount of data transfer on the system bus can be greatly reduced by performing in-memory computation. The acceleration of matrix-vector multiplication (MVM) using in-memory computing has recently attracted significant attention because it is the dominating computational kernel within many important applications. In particular, it is the dominating computational kernel within the simulation of physical systems within scientific computing applications. Consequently, it is not surprising that the acceleration of MVM has been extensively explored using both analog and digital in-memory computing paradigms.

Analog in-memory computing is based on performing analog MVM using the natural multiply-and-accumulate feature of memristor crossbar arrays, which is extremely energy-efficient [19]. Unfortunately, analog in-memory computing is inherently vulnerable to errors introduced by random telegraph noise, sneak currents, temperature fluctuations, and other sources of variation [11]. While the relaxed precision may be acceptable for image processing [17] and artificial intelligence applications [18], computation within scientific computing applications must meet strict precision requirements [12].

Digital in-memory computing is focused on executing Boolean functions in-memory. The acceleration of MVM using digital in-memory computing has been investigated using logic families such as Material Implication (IMP) [3], MAGIC [14], programmable OR plane [7], path-based logic [28]. While the robustness of digital computing allows arbitrary precision requirements to be satisfied, the energy-efficiency is substantially lower than for analog in-memory computing. Consequently, neither of the state-of-the-art in-memory computing paradigms can deliver both high precision and high energy efficiency.

In this paper, we propose a new computing paradigm called hybrid analog-digital in-memory computing. The paradigm is capable of performing matrix-vector multiplication with high energy-efficiency and precision. The paper also proposes the associated in-memory computing platform and design automation tool chain needed to perform computation using the platform. The key idea is to decompose the most significant bits (MSBs) of the desired computation into Boolean functions and the least significant bits (LSBs) into matrix-vector multiplication operations. Next, the respective kernels are mapped to digital and analog in-memory computing hardware.

This work was in part supported by NSF awards CNS-1908471, CNS-2008339 and CCF-1822976.

TABLE I: Properties of previous and proposed computing paradigms.

Computing Paradigm	Work in	Precision	Energy Efficiency	In-Memory Computation
Digital CMOS based Computing	[9, 29, 30]	**high**	low	no
Analog In-Memory Computing	[8, 17, 26, 32]	limited	**very high**	**yes**
Digital In-Memory Computing	[3, 7, 14, 28]	**high**	moderate	**yes**
Hybrid Analog-Digital In-Memory Computing	**(Proposed)**	**high**	high	**yes**

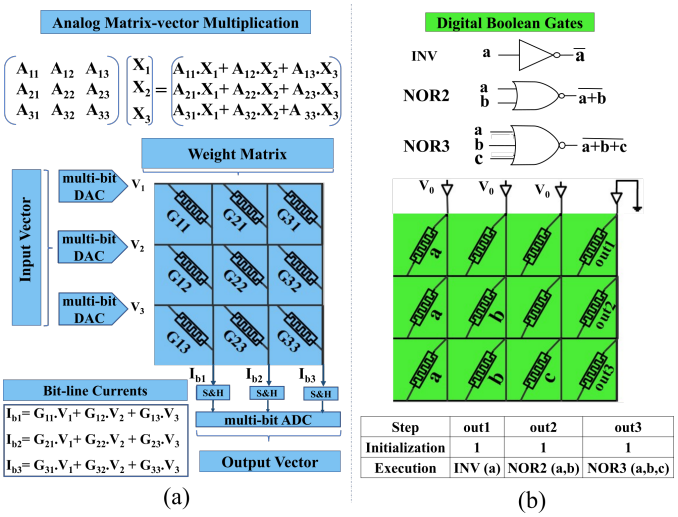


Fig. 1: (a) Analog in-memory computing using analog MVM and (b) digital in-memory computing using Boolean logic.

The experimental evaluation validates the advantages of the hybrid analog-digital paradigm over both analog and digital in-memory computing, respectively. When utilizing the proposed paradigm to accelerate the simulation of complex physical systems, power, area, and latency is improved by 2.44X, 2.45X and 2.32X respectively. The proposed paradigm also gains 620X and 493X speedup and energy-efficiency over GPUs.

The remainder of the paper is organized as follows: preliminaries and limitations of previous work in Section II. The hybrid analog-digital in-memory computing paradigm is outlined in Section III. The synthesis methodology is presented in Section IV. The proposed architecture is discussed in V. The experimental evaluation is presented in Section VI. The paper is concluded with summary and future work in Section VII.

II. Preliminaries

In this section, we review the state-of-the-art techniques of accelerating MVM using analog and digital in-memory computing. Next, we compare with the proposed hybrid paradigm.

A. Analog in-memory computing

The concept of performing analog matrix-vector multiplication using the natural multiply-and-accumulate feature of memristor crossbar arrays is shown in Figure 1(a) [8, 17, 26, 32]. The elements of a matrix can be represented using the conductance of the corresponding memristors. By applying a voltage input vector to the wordlines, analog matrix vector multiplication is performed using Ohm's law and Kirchoff's current law. The output vector is obtained by measuring the output currents from the bitlines. Digital-to-analog converters (DACs) and analog-to-digital converters (ADCs) are used to convert signals between the analog/digital domain.

While analog in-memory computing is extremely energy-efficient, the computation is vulnerable to errors introduced by parasitics, random telegraph noise (RTN) [4], and other variations [1, 20]. Many prominent architecture level studies have proposed to overcome this "precision challenge" by treating crossbars as fixed-point multipliers. Next, arbitrary high-precision is emulated using paradigm based on bit-slicing and shift-and-add operations [13, 26]. Unfortunately, the concept only works for digital computing paradigms where the precision is deterministic. It is easy to understand that the shift-and-add paradigm may amplify small errors into large errors, which compromises the overall precision.

B. Digital in-memory computing

Matrix-vector multiplication using digital in-memory computing is performed by decomposing the computation into Boolean functions that are decomposed into digital gates. Next, the gates are executed using the digital in-memory computing, which is illustrated in Figure 1(b). Various digital in-memory computing logic styles have recently been investigated, each provide different trade-offs in terms of latency, power, area, and endurance [3, 7, 14, 28]. In this paper, we utilize MAGIC logic family [14] which has gained significant attention for its structured synthesis flow and easy integration into standard design automation tools.

While digital-in memory computing can deliver high-precision, it is easy to understand that the computation is less energy-efficient than analog in-memory computing.

C. Limitation of previous work and proposed paradigm

In this section, we compare the proposed hybrid analog-digital in-memory computing paradigm with state-of-the-art computing paradigms. The overall properties of the different computing paradigms are shown in Table I.

It can be observed in the table that traditional digital CMOS-based computing achieves high-precision but has poor energy-efficiency. This is mainly due to that computation is not performed in-memory, i.e., the energy-efficiency is degraded by the data movement on the system bus. Compared with traditional digital computing, digital in-memory computing improves the energy-efficiency to moderate as the computation is performed in-memory. Analog in-memory computing further improves the energy efficiency at the expense of degrading the precision. The proposed hybrid analog-digital in-memory computing paradigm achieves the best of both worlds, i.e., both high precision and high energy-efficiency. We consider the proposed paradigm to be orthogonal to the mixed-precision computing in [16] and hybrid mode computing in [31].

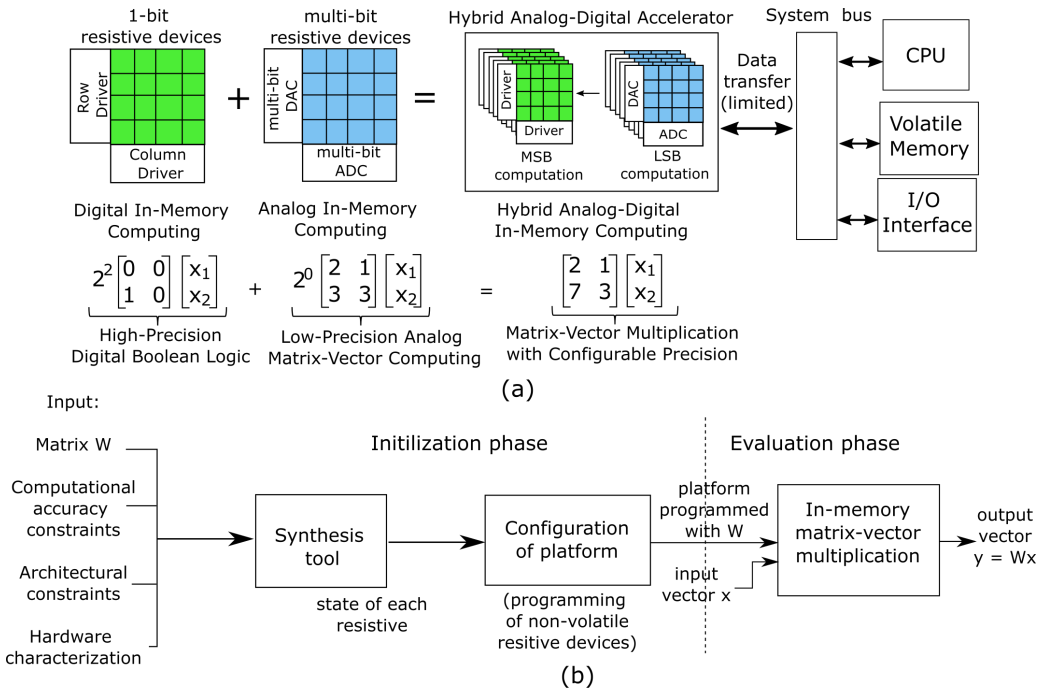


Fig. 2: Proposed hybrid in-memory computing platform .

III. Hybrid Analog-Digital In-Memory Computing

In this section, we propose our hybrid analog-digital in-memory computing paradigm that is capable of performing matrix-vector multiplication ($Wx = y$) with configurable quality, i.e., the energy-efficiency tailored to the requirements on the computational accuracy. First, we provide an overview of the hybrid analog-digital platform. Second, we outline the detailed properties of the platform.

A platform for the proposed hybrid in-memory computing is shown in Figure 2(a). The platform operates using a design initialization phase and an evaluation phase, which is shown in Figure 2(b). In the initialization phase, a synthesis tool is used to decompose the desired computation into Boolean functions and matrix-vector multiplication operations. The goal is to compute the most significant bits (MSB) using high-precision digital in-memory computing and the least significant bits (LSBs) using low-precision analog in-memory computing, which ensures that the overall computation is of high-precision. A simplistic decomposition is shown at the bottom of Figure 2(a). The decomposition heavily depends on the constraints on the computational accuracy, architecture, and hardware characterization (see details in Section IV and Section VI). Next, the hybrid platform is configured to realize the matrix-vector multiplication operations by programming the non-volatile resistive devices within the platform. The synthesis tool performs logic and physical co-optimization while binding the computation to the hardware. In the evaluation phase, matrix-vector multiplication is performed and output vectors are recorded. The proposed hybrid platform can be used as an accelerator in a next generation HPC system, as there is limited data transfer on the system bus, which is shown at the right of Figure 2(a). An underlying assumption of the paradigm is that the matrix is fixed and only the input vectors change, which holds for the simulation of complex physical systems within scientific computing applications.

A. Properties of hybrid paradigm

In this section, we explain the details behind the properties listed for the hybrid paradigm in Table I.

1) High-precision: It is easy to understand that the hybrid analog-digital paradigm can attain high (or arbitrary precision) because digital in-memory computing is deterministic and can achieve arbitrary precision. Moreover, the use of analog in-memory computing does not substantially compromise the precision because it is only used to compute the LSBs. However, computing the LSBs with analog in-memory computing results in better precision compared with simply discarding the LSB computation.
2) High energy-efficiency: The energy-efficiency is high because all computation is performed in memory. Moreover, we maximize the amount of computation that can be performed using analog in-memory computing. In addition, the full potential of analog in-memory computing is utilized by leveraging multi-bit memristors and the natural ability of ADCs to measure the most significant bits within a specified sensing range.
3) In-Memory Computation: The entire computation is performed in memory which reduces the data movement in the system bus. For example, for a matrix of n×m dimension, the total data transfer is improved from $(n \times m) + m + n$ to only $(m+n)$, where $(m+n)$ denotes the input and output vector data.

In the following sections, we describe the details of the proposed hybrid analog-digital synthesis tool and in-memory computing platform.

IV. Analog-Digital Synthesis

In this section, we describe the details of the synthesis tool capable of decomposing matrix-vector multiplication operations into analog and digital in-memory computing kernels. The objective is to perform the mapping while minimizing power consumption and hardware overheads. The synthesis flow is illustrated in Figure 3. The input to the flow is the desired computation, the constraint on the computational accuracy, the hardware characterization, and the architecture constraints. The first step is Analog-Digital decomposition. This step is used to decompose the desired computation into a digital MSB component and an analog LSB component based on the required precision. The details are provided in Section IV-A. Next, an analog synthesis step is performed to map the analog computation into analog MVM operations while meeting hardware and architecutre constraints. The details are provided in Section IV-B. Similarly, a digital synthesis step is performed to map the digital component into digital in-memory computing kernels while meeting hardware and architecture constraints. The digital synthesis part is performed exactly as in SIMPLER [2]. Please refer to that work for the technical details of the digital in-memory computing.

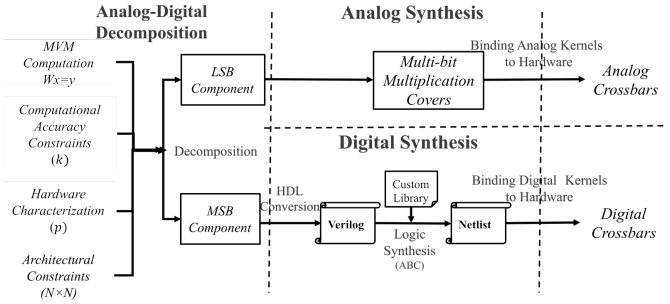


Fig. 3: Hybrid analog-digital in-memory synthesis flow.

A. Analog-digital decomposition

In this section, we explain how the analog-digital decomposition is performed. As matrix-vector multiplication mainly consists of many element-wise multiplications, we perform a case study on element-wise multiplication. Next, we generalize the concept to matrix-vector multiplications.

Analog-digital barrier: The multiplication of two six-bit numbers A and B is shown in Figure 4. A naive MSB and LSB decomposition is shown in Figure 4(a). The two most significant bits of both numbers are mapped to deterministic digital in-memory computing and the four LSBs are mapped to approximate analog in memory computing. The figure shows the portion of the output that has deterministic and approximate precision, respectively. The proposed analog-digital decomposition is shown in Figure 4(b). The new decomposition assigns a bit in A and B to analog and digital based on the bit it is multiplied with. While the amount of digital in-memory computation is similar (20 bit-wise multiplications vs 21 bit-wise multiplications), there are 2 additional outputs with deterministic precision. Therefore, we expect the proposed scheme to be advantageous over the first scheme. Next, we analyze if it is necessary to perform all the analog computation.

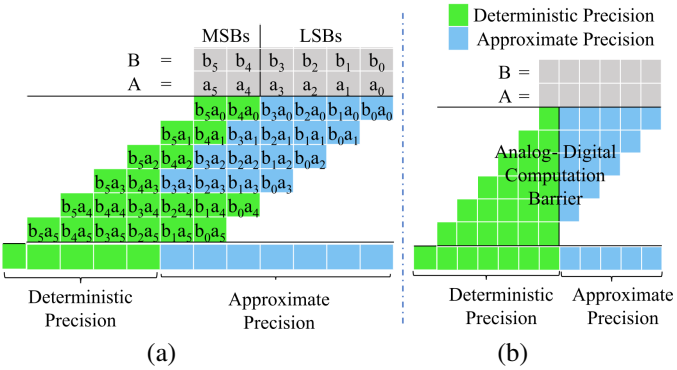


Fig. 4: Bit-wise multiplication of two six bit numbers using (a) naive decomposition and (b) proposed decomposition.

Analog-don't care barrier: As the analog computation is approximate, we speculate that it may not be meaningful to perform all the analog bit-wise multiplications. For example, the expected error of approximately multiplying b_4a_0 may be larger than the result of multiplying b_0a_0. Therefore, we propose to also introduce a don't care barrier, which is shown in Figure 5. The barrier denotes that we don't care if the computation on the right hand side of the barrier is performed or not. We use don't care instead of simply dropping the computation because we observe in the next section that it is sometimes cheaper to include some additional analog computation. The multiplication of two q-bit numbers results in a number with $2q$-bits. The analog-digital barrier is placed to captures the k MSBs of the $2q$-bits. The analog-don't care barrier is denoted with p and defined with respect to k.

Using the detailed experimental setup that is described in the experimental results section, we evaluate varying the value of k and p independently in Figure 6. It can be observed in Figure 6(a) that the maximum error in the output is reduced when k is increased. This is expected as more computation is moved from cheap approximate analog computing into deterministic and more expensive digital in-memory comput-

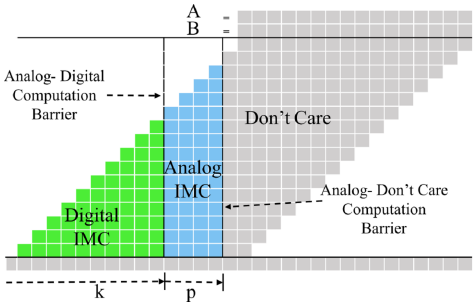
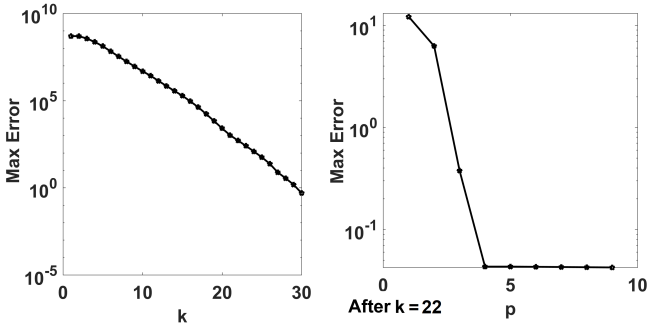


Fig. 5: Analog-digital barrier and analog-don't care barrier.



(a) (b)

Fig. 6: The maximum error with respect to (a) the analog-digital barrier k and, (b) the analog-don't care barrier p. The experiment is performed using two 16 bit numbers.

ing. The value of k is required to be specified based on the precision requirements on the application. At the same time, the maximum error is reduced until p is increased to 4, as demonstrated in Figure 6(b). Next, the maximum error stays the same when p is increased further. This stems from that the non-volatile devices have an accuracy of approximately 4-bits in our experimental setup (see Section VI). This is a conservative estimate of the devices that can be fabricated today [11, 16, 17]. Therefore, we set p to be equal to the approximate bit-accuracy of the non-volatile devices in the remainder of the paper.

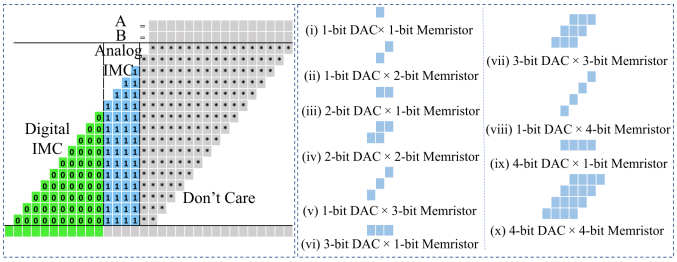
B. Analog synthesis

In this section, we will explain how the analog computation is mapped into analog MVM operations. This is performed by formulating and solving a covering problem. Next, the covering solution is mapped into analog MVM operations.

The analog-don't care covering problem: The mapping of the analog computation into analog MVM operations is shown in Figure 7. The overall idea is to cover the computation in Figure 7(a) using the analog MVM operations in Figure 7(b). The figure shows an example for an element-wise multiplication but we will later show in this section that it can directly be generalized into matrix-vector multiplication operations. The computation in Figure 7(a) consists of three distinct regions:

1) *Digital IMC*: The computation in the digital region is not allowed to be mapped to analog in-memory computing, which is indicated using 0's.
2) *Analog IMC*: The computation in the analog region is required to be covered with analog kernels and is therefore marked with 1's.
3) *Don't Care*: The computation in the don't care region can be covered optionally, which is indicated using *'s.

The shape of the analog MVM operations is shown in Figure 7(b). It can be observed that all the kernels inherently have the shape of tilted rectangles. The shape depends on the bit-accuracy of the non-volatile memristors and DACs. The higher the bit-accuracy, the larger the shape. Each of the kernels also has an associated area cost and power consumption. In particular, the power consumption is correlated with the bit-



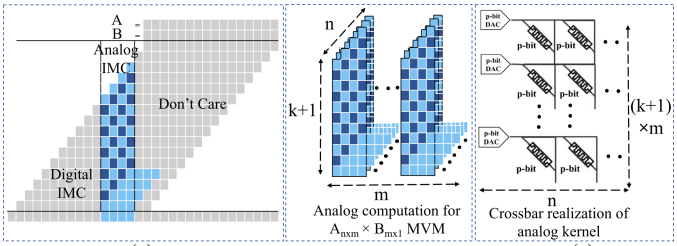
(a) (b)

Fig. 7: (a) Analog-Digital-Don't Care covering problem and, (b) analog multi-bit multiplication cover library.

accuracy of the DACs. There is also a fixed cost per kernel for the ADC used to measure the output of the MVM operation.

Next, a mathematical covering problem can be formulated to cover all the ones, not cover any zeros, and optionally cover the don't cares. In general, this can straightforward be solved using an mixed integer programming formulation (MIP) using techniques in [5]. However, in this paper, we leverage an optimal substructure to avoid formulating the MIP formulation. The optimal solution is shown in Figure 8(a). Intuitively, the solution is optimal because there is no power penalty of using memristors with high bit-accuracy. It is also advantageous to use 4-bit DACs because this reduces the total number of kernels, which reduces the number of ADCs that are required to measure the output of the analog MVM operations.

Mapping of analog kernels to crossbars: Given the decomposition of an element wise multiplication, we now generalize it to a solution for matrix-vector multiplication. We also explain the intuitive mapping into crossbars using the example in Figure 8.



(a) (b) (c)

Fig. 8: (a) Covering solution. (b) Generalization to mxn matrix-vector multiplication. (c) Mapping of MVM operations to crossbars.

An optimal solution to the covering problem in Figure 7 is shown in Figure 8(a). The solution is generalized to a $n \times m$ matrix. The number of rows n expands the cover in Figure 8(a) depth wise into a volume. The number of columns m replicates the computation m times. The generalization is shown in Figure 8(b). Next, the mapping into a crossbar with dimensions of $(km+m) \times (n)$ is shown in Figure 8(c). The crossbar is trivially split into multiple smaller crossbars if the maximum crossbar dimension is exceeded.

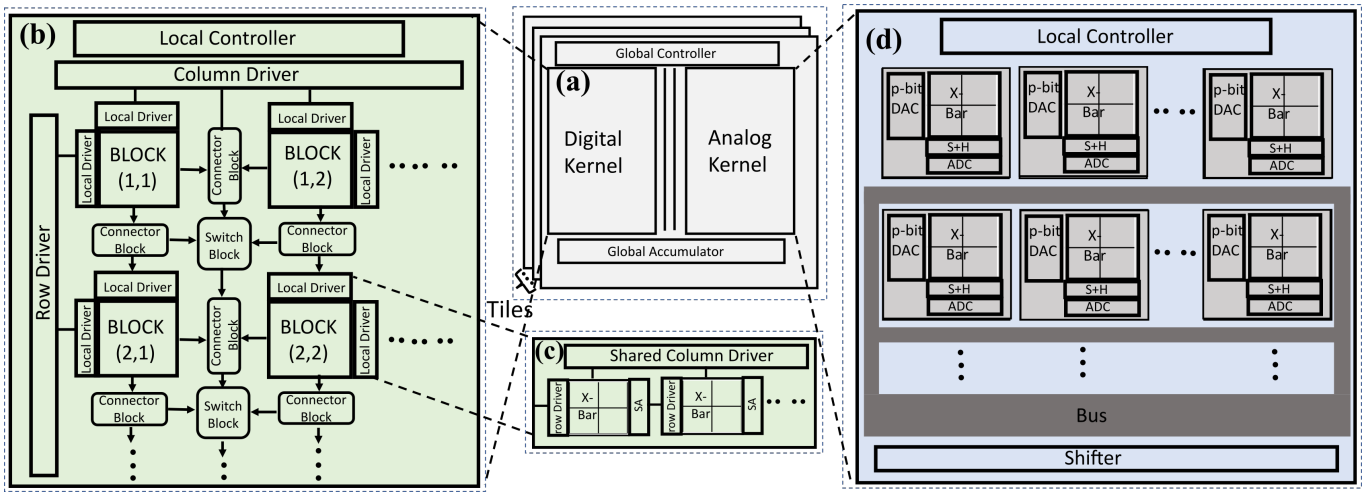


Fig. 9: Architecture overview of analog-digital hybrid accelerator.

V. Architecture

In this section we discuss the architecture of the analog-digital hybrid in-memory computing. First, we discuss the overview of the micro-architecture. Next, we discuss the parallelism and cross-architecture data transfer costs.

Overview: The overall architecture of the analog-digital hybrid accelerator is presented in Figure 9. The architecture has several accelerator tiles, each divided into two kernels: digital and analog, which is shown in Figure 9(a). The kernels are controlled by a global controller. The controller receives CPU instructions to generate control signals for in-memory computing. The computational results of each kernel is collected by a global accumulator.

The digital kernel is divided into many computational blocks as shown in Figure 9(b). An island-style FPGA inspired routing architecture is adopted to communicate among the blocks [24]. The connector blocks enable the computational blocks to communicate with the neighbouring blocks and the switch-blocks enable cross-architecture communication. Each computational block consists of several 1-bit memristor crossbars connected in row-parallel fashion [12], which is shown in Figure 9(c). The sense amplifiers (SA) and row drivers enable row-parallel copying of data which is particularly beneficial for implementing MAGIC.

The analog kernel is divided into analog hardware blocks consisting of multi-bit DACs, multi-bit ADCs and multi-bit memristor crossbars. A shared bus collects the results of different analog blocks and carries them to the global accumulator as shown in Figure 9(d).

Parallelism: Matrix-vector multiplication operation is inherently parallelizable. For a $n \times m$ matrix multiplied with a $m \times 1$ vector, the resultant vector is of size $n \times 1$. The computations of these n output elements are independent of each other. The multiplication of the i-th row of matrix with the input vector is independent of the multiplication of the $i+1$-th row of the matrix with the input vector. This high order of parallelism is particularly beneficial to the comparatively costly digital in-memory computing. The proposed architecture adopts the state-of-the art MAGIC-based synthesis and mapping tool SIMPLER [2]. SIMPLER maps boolean functions to a single row. This enables us to fully exploit the parallelism offered by MAGIC by performing different row-wise multiplication operations of MVM into different rows simultaneously. If the target computation exceeds the crossbar size, the computation can easily be carried into the adjacent crossbars using the row-parallel operations in Figure 9(c).

Micro-architectural data transfer: It is understandable that MVM is a computationally expensive task. Therefore, it is expected that the computation will need a significant portion of the available architectural resource. Therefore, it is very crucial to consider micro-architectural data transfer between different units which will limit the benefit of in-memory computing to some extent. We consider the case study on data transfer in memristive Memory Processing Unit (mMPU) presented in [27]. We appropriately modify the data transfer cost for the proposed architecture. The row-based MAGIC operation relies on reuse of memristors in the row. Therefore, several copy (achieved by performing two MAGIC NOT operations) operations are performed. This operation introduces the following intra-crossbar cost for each data movement:

$$Cost_{intra-crossbar} = 2.T_{MAGIC} \tag{1}$$

where, T_{MAGIC} is the time taken by a MAGIC operation. Next, we consider the intra-block data transfer for communication between two crossbars inside the same block. Intra-block data transfer consists of a series of read and write operations. We also need to consider the read/write mode switching latency of the peripherals. For simplicity, we consider that the sense amplifiers have sufficient bandwidth to perform the row-parallel read operation in one cycle. Each intra-block communication cost can be computed as follows:

$$Cost_{intra-block} = T_{read} + T_{RTW} + T_{write} + T_{WTR} \tag{2}$$

TABLE II: Architecture Components Area-Power Cost

Component	Parameter	Specs	Area	Power
Crossbar	Size	128×128	0.0002 mm^2	2.4 mW
DAC	Resolution	4 bits	1.328 μm^2	0.0312 mW
ADC	Resolution	8 bits	0.0012 mm^2	2 mW
Sample+Hold	# Unit	1	0.039 μm^2	10 nW
Total (analog block)	# Unit	1	0.001570023 mm^2	9.6736 mW
Controller	# Unit	1	0.000401 mm^2	0.65 mW
Total (digital crossbar)	# Unit	1	0.000601 mm^2	3.05 mW
Shifter	# Unit	1	0.00006 mm^2	0.05 mW
Bus	Bandwidth	128-bits	15.7 mm^2	13 mW
Connector Block	Bandwidth	128-bits	0.5108 μm^2	0.00656 mW
Switch Block	Bandwidth	128-bits	2.0432 μm^2	0.02625 mW

where, T_{RTW} and T_{WTR} are read-to-write and write-to-read switching latencies respectively. Finally, we consider the inter-block communication cost. One useful aspect of inter-block communication is that multiple blocks can be pipelined to perform simultaneous read/write operations. The inter-block data transfer cost can be computed as follows:

$$Cost_{inter-block} = T_{read} + max\{T_{wait}, T_{routing}\} + T_{write} + T_{wait} \quad (3)$$

where, T_{wait} is the wait time between two consecutive read/write commands and $T_{routing}$ is the data routing time from source to target block.

VI. Experimental Evaluation

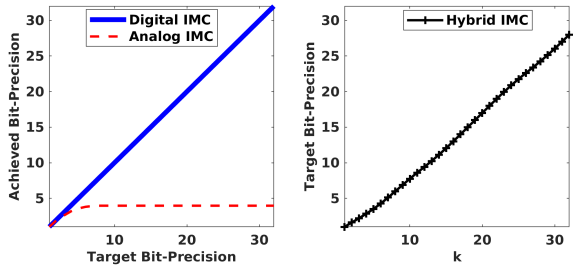
In this section, we evaluate the performance of the hybrid analog-digital in-memory computing paradigm.

The properties of the proposed hybrid in-memory computing platform are provided in Table II. The parameters are adapted from earlier studies in [8, 15, 23, 26]. In particular, we use memristors with 4 bits, which is conservative with respect to the fabrication results reported in [11, 16, 17]. We adopt the fitting characteristics of the VTEAM [15] model to emulate experimental memristor devices. We set the R_{LRS} and R_{HRS} to be $10k\Omega$ and $10M\Omega$ respectively. To account for the write-error of multi-bit cells and other external noises, we introduce normally distributed random noises, $R \sim \mathcal{N}(\mu, \sigma^2)$, where, μ is the expected value, and σ is the standard deviation. We perform a Monte-Carlo simulation assuming 5% standard deviation. The DACs are set to 4 bits due to the dependency with the memristor bit-accuracy. The per-unit costs for different architectural components are summarized in Table II [8, 23, 26]. We use 8-bit ADCs in the evaluation to balance accuracy and overheads. The cross-architecture data transfer cost is calculated using equations $(1)-(3)$. The read, write and MAGIC NOR latency is adopted from [27] as 10ns, 25ns and 32.5ns respectively. For the choice of von Neumann machine, we use an Intel Core i9 processor with NVIDIA GeForce RTX 2070S GPU.

We compare the proposed hybrid paradigm with analog in-memory computing, digital-in-memory computing, and a von-Neumann machine with a GPU. Results on the MVM level and for scientific computing applications are presented in Section VI-A and Section VI-B.

A. Evaluation of MVM

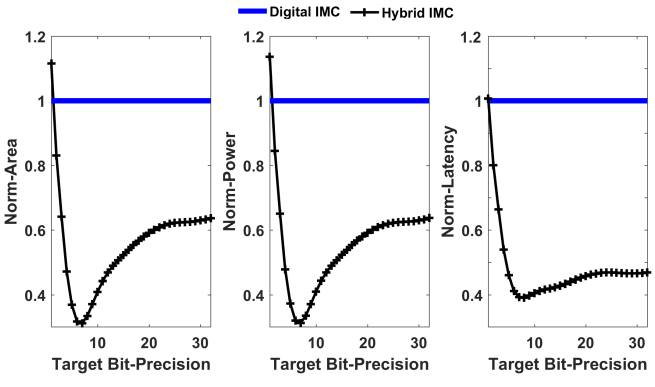
In this section, we first compare and contrast the precision and performance of the different computing paradigms. The comparison is performed using 128x128 bit matrices, where each element is represented with 32-bit precision. We compare the precision with the other in-memory computing paradigms in Figure 10. It can be observed that analog in-memory computing is not able to achieve high precision due to that errors are scaled-up using the shift-and-add paradigm. Both digital and hybrid in-memory computing are capable of achieving arbitrary precision. Consequently, there is no need to compare the other metrics with analog in-memory computing.



(a) (b)

Fig. 10: Precision comparison of different in-memory computing paradigms. (a) target bit precision vs. achieved bit precision for digital and analog in-memory computing and, (b) target bit-precision vs. k for the hybrid paradigm.

Now we turn our attention to comparing the area, power, and latency with digital in-memory computing in Figure 11. The results are obtained by setting k such that an equivalent precision is obtained. Note that k is required to be experimentally determined for each evaluated matrix. Next, the results are normalized with respect to the digital in-memory computing. The figure shows that the hybrid paradigm reduces the area, power, latency with between 50% to 70% for reasonable values of the target bit-precision. Compared with a von-Neumann based machine, the hybrid paradigm achieves 620X speed and 493X energy efficiency gain.



(a) (b) (c)

Fig. 11: Overhead comparison of digital in-memory computing and hybrid in-memory computing paradigms with respect to (a) area, (b) power, and (c) latency. All results are normalized with respect to digital in-memory computing.

TABLE III: Summary of Applications for Evaluation

Applications	System Type	Matrix Size	Non-zeros
Trefethen-20	Combinatorial	20x20	158
bcsstk02	Structural	66x66	4356
Journals	Undirected Weighted Graph	124x124	12068
Trefethen-150	Combinatorial	150x150	2040
Trefethen-200b	Combinatorial	199x199	2873
Trefethen-200	Combinatorial	200x200	2890
mesh3em5	Structural	289x289	1377
Trefethen-300	Combinatorial	300x300	4678
Trefethen-500	Combinatorial	500x500	8478
Trefethen-700	Combinatorial	700x700	12654
Chem97ZtZ	Statistical/Mathematical	2541x2541	7361

B. Evaluation with scientific computation

In this section, we evaluate and compare the performance of the different state-of-the-art in-memory computing paradigms for the simulation of physical systems within scientific computing applications which demand high-precision. As a choice of the state-of-the art in-memory computing paradigms, we use SIMPLER [2] for digital in-memory computing and ISAAC [26] for analog in-memory computing. Note that we tailor the size of the in-memory computing platforms with respect to each application. We evaluate the paradigms with a number of applications from the sparse matrix collection of [6], which are shown in Table III. We aim to solve these systems of linear equations using the conjugate gradient (CG) method [10]. The CG method uses an iterative refinement algorithm to solve a system of linear equations. The algorithm terminates when a certain error tolerance is satisfied. In each iterative refinement, an expensive MVM is performed. We aim to improve the efficiency of the MVM operation using hybrid in-memory computing. To ensure high precision, we set the error tolerance of the CG method to 10^{-15}. However, any arbitrary precision requirement could have been selected.

TABLE IV: Results of Conjugate Gradient Solver

Applications	Order of Concluding Error		
	ISAAC [26]	SIMPLER [2]	Hybrid Analog-Digital IMC
Trefethen-20	7.33×10^{-5}	4.7×10^{-16}	7.2×10^{-16}
bcsstk02	0.79×10^{-3}	0.37×10^{-16}	0.52×10^{-16}
Journals	9.54×10^{-2}	5.11×10^{-16}	7.45×10^{-16}
Trefethen-150	7.43×10^{-4}	5.79×10^{-16}	9.75×10^{-16}
Trefethen-200b	7.92×10^{-4}	1.49×10^{-16}	1.73×10^{-16}
Trefethen-200	4.82×10^{-4}	1.92×10^{-16}	2.30×10^{-16}
mesh3em5	3.98×10^{-5}	8.25×10^{-16}	9.34×10^{-16}
Trefethen-300	8.63×10^{-4}	4.36×10^{-16}	4.98×10^{-16}
Trefethen-500	0.47×10^{-3}	3.19×10^{-16}	3.39×10^{-16}
Trefethen-700	9.25×10^{-2}	3.21×10^{-16}	5.68×10^{-16}
Chem97ZtZ	6.04×10^{-3}	1.51×10^{-16}	2.74×10^{-16}
Summary	**ISAAC**	**0/11 application converged**	
	SIMPLER	**11/11 applications converged**	
	Hybrid	**11/11 applications converged**	

We show the performance of different paradigms as CG solver in Table IV. In the table, we show the final error for different paradigms. The results indicates that the SIMPLER and the hybrid in-memory computing successfully solves all the systems. But the ISAAC does not meet the error tolerance and therefore does not converge for any of these systems. As ISAAC model does not converge for any of the systems, we only present the overhead analysis of the hybrid in-memory computing and SIMPLER in Figure 12. Figure 12 presents the normalized performance in terms of area, energy and latency. The experimental results show that hybrid in-memory computing system improves power, area, and latency over SIMPLER by 2.44X, 2.45X and 2.32X, respectively. The experimental findings indicate that the hybrid model is capable of performing high-precision scientific computation with superior overhead over the digital in-memory computing. Based on the comparisons with a GPU on the MVM level, similar performance benefits on the application level are also expected.

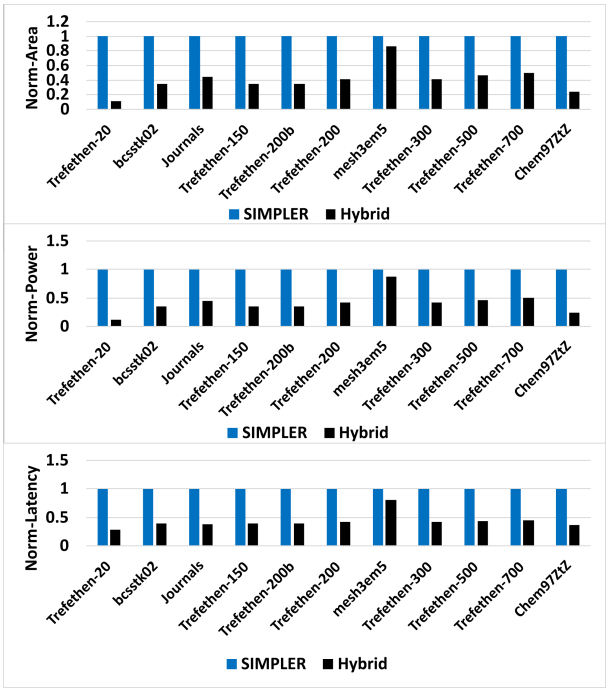


Fig. 12: Evaluation of scientific computing applications.

In summary, the proposed hybrid paradigm is capable of achieving the same high precision as digital-in memory computing by appropriately decomposing the computation into analog and digital in-memory computing kernels. However, the computation is more efficient in terms of power, latency, and area because the paradigm enables efficient analog in-memory computing to be used without compromising the computational accuracy.

VII. Summary and Future Work

In this paper we propose a novel hybrid analog-digital in-memory computation scheme. The proposed model can perform matrix-vector multiplication operation with both high precision and high energy-efficiency. The hybrid in-memory computing is ideal for accelerating scientific computing applications with high-precision requirements. Hybrid in-memory computing paradigm opens up a new dimension in the field of in-memory computing research. There are several opportunities to explore the use of hybrid analog-digital in-memory computing to accelerate other kernels and applications. While the proposed synthesis tool chain is sophisticated, there may exist opportunities to further improve the synthesis flow.

REFERENCES

[1] E. Balestrieri, P. Daponte, and S. Rapuano. A state of the art on adc error compensation methods. *IEEE Transactions on Instrumentation and Measurement*, 54(4):1388–1394, 2005.

[2] R. Ben-Hur, R. Ronen, A. Haj-Ali, D. Bhattacharjee, A. Eliahu, N. Peled, and S. Kvatinsky. Simpler magic: Synthesis and mapping of in-memory logic executed in a single row to improve throughput. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(10):2434–2447, 2019.

[3] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, and R. S. Williams. 'memristive'switches enable 'stateful'logic operations via material implication. *Nature*, 464(7290):873–876, 2010.

[4] S. Choi, Y. Yang, and W. Lu. Random telegraph noise and resistance switching analysis of oxide based resistive memory. *Nanoscale*, 6(1):400–404, 2014.

[5] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.

[6] T. A. Davis and Y. Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1–25, 2011.

[7] A. Dehon. Nanowire-based programmable architectures. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 1(2):109–162, 2005.

[8] B. Feinberg, U. K. R. Vengalam, N. Whitehair, S. Wang, and E. Ipek. Enabling scientific computing on memristive accelerators. In *2018 ACM/IEEE 45th ISCA*, pages 367–382. IEEE, 2018.

[9] T. P. Haraszti. *CMOS memory circuits*. Springer Science & Business Media, 2007.

[10] M. R. Hestenes, E. Stiefel, et al. *Methods of conjugate gradients for solving linear systems*, volume 49. NBS Washington, DC, 1952.

[11] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams. Dot-product engine for neuromorphic computing: Programming 1t1m crossbar to accelerate matrix-vector multiplication. In *2016 53nd ACM/EDAC/IEEE DAC*, pages 1–6. IEEE, 2016.

[12] M. Imani, S. Gupta, Y. Kim, and T. Rosing. Floatpim: In-memory acceleration of deep neural network training with high precision. In *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*, pages 802–815. IEEE, 2019.

[13] H. Jiang, S. Huang, X. Peng, and S. Yu. Mint: Mixed-precision rram-based in-memory training architecture. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2020.

[14] S. Kvatinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser. Magic—memristor-aided logic. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 61(11):895–899, 2014.

[15] S. Kvatinsky, M. Ramadan, E. G. Friedman, and A. Kolodny. Vteam: A general model for voltage-controlled memristors. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 62(8):786–790, 2015.

[16] M. Le Gallo, A. Sebastian, R. Mathis, M. Manica, H. Giefers, T. Tuma, C. Bekas, A. Curioni, and E. Eleftheriou. Mixed-precision in-memory computing. *Nature Electronics*, 1(4):246–253, 2018.

[17] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, C. E. Graves, et al. Analogue signal and image processing with large memristor crossbars. *Nature Electronics*, 1(1):52, 2018.

[18] S. Liu, Y. Wang, M. Fardad, and P. K. Varshney. A memristor-based optimization framework for artificial intelligence applications. *IEEE Circuits and Systems Magazine*, 18(1):29–44, 2018.

[19] L. Ni, Y. Wang, H. Yu, W. Yang, C. Weng, and J. Zhao. An energy-efficient matrix multiplication accelerator by distributed in-memory computing on binary rram crossbar. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 280–285. IEEE, 2016.

[20] T. Ogawara and T. Takiguchi. Digital-to-analog converter with conversion error compensation, Mar. 30 1993. US Patent 5,198,814.

[21] R. A. Pielke Sr. *Mesoscale meteorological modeling*. Academic press, 2013.

[22] F. Pop. High performance numerical computing for high energy physics: a new challenge for big data science. *Advances in High Energy Physics*, 2014, 2014.

[23] M. Saberi, R. Lotfi, K. Mafinezhad, and W. A. Serdijn. Analysis of power consumption and linearity in capacitive digital-to-analog converters used in successive approximation adcs. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 58(8):1736–1748, 2011.

[24] H. Schmit. Extra-dimensional island-style fpgas. In *New Algorithms, Architectures and Applications for Reconfigurable Computing*, pages 3–13. Springer, 2005.

[25] B. Schölkopf, K. Tsuda, and J.-P. Vert. *Kernel methods in computational biology*. MIT press, 2004.

[26] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM SIGARCH Computer Architecture News*, 44(3):14–26, 2016.

[27] N. Talati, A. H. Ali, R. B. Hur, N. Wald, R. Ronen, P.-E. Gaillardon, and S. Kvatinsky. Practical challenges in delivering the promises of real processing-in-memory machines. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1628–1633. IEEE, 2018.

[28] A. Velasquez and S. K. Jha. In-memory computing using paths-based logic and heterogeneous components. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1512–1515. IEEE, 2018.

[29] M. V. Wilkes. The memory wall and the cmos end-point. *ACM SIGARCH Computer Architecture News*, 23(4):4–6, 1995.

[30] W. A. Wulf and S. A. McKee. Hitting the memory wall: implications of the obvious. *ACM SIGARCH computer architecture news*, 23(1):20–24, 1995.

[31] Q. Zheng, X. Li, Z. Wang, G. Sun, Y. Cai, R. Huang, Y. Chen, and H. Li. Mobilattice: A depth-wise dcnn accelerator with hybrid digital/analog nonvolatile processing-in-memory block. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2020.

[32] Z. Zhu, J. Lin, M. Cheng, L. Xia, H. Sun, X. Chen, Y. Wang, and H. Yang. Mixed size crossbar based rram cnn accelerator with overlapped mapping method. In *2018 IEEE/ACM ICCAD*, pages 1–8. IEEE, 2018.