Accelerating Deep Neural Networks in Processing-in-Memory Platforms: Analog or Digital Approach?

Shaahin Angizi[†], Zhezhi He[†], Dayane Reis[§], Xiaobo Sharon Hu[§], Wilman Tsai[*], Shy Jay Lin[*], Deliang Fan[†]
[†] Department of Electrical and Computer Engineering, University of Central Florida, FL 32816, USA
[§] Department of Computer Science and Engineering, University of Notre Dame, IN 46556, USA
[*] TSMC, Hsinchu, Taiwan
Email: dfan@ucf.edu

***Abstract*—Nowadays, research topics on AI accelerator designs have attracted great interest, where accelerating Deep Neural Network (DNN) using Processing-in-Memory (PIM) platforms is an actively-explored direction with great potential. PIM platforms, which simultaneously aims to address power- and memory-wall bottlenecks, have shown orders of performance enhancement in comparison to the conventional computing platforms with Von-Neumann architecture. As one direction of accelerating DNN in PIM, resistive memory array (aka. crossbar) has drawn great research interest owing to its analog current-mode weighted summation operation which intrinsically matches the dominant Multiplication-and-Accumulation (MAC) operation in DNN, making it one of the most promising candidates. An alternative direction for PIM-based DNN acceleration is through bulk bit-wise logic operations directly performed on the content in digital memories. Thanks to the high fault-tolerant characteristic of DNN, the latest algorithmic progression successfully quantized DNN parameters to low bit-width representations, while maintaining competitive accuracy levels. Such DNN quantization techniques essentially convert MAC operation to much simpler addition/subtraction or comparison operations, which can be performed by bulk bit-wise logic operations in a highly parallel fashion. In this paper, we build a comprehensive evaluation framework to quantitatively compare and analyze aforementioned PIM based analog and digital approaches for DNN acceleration.**

***Index Terms*—Neural network acceleration, Processing-in-memory, crossbar, in-memory bulk logic.**

I. Introduction

Deep Neural Network (DNN) has achieved world-wide attention due to outstanding performance in image recognition over large scale data-set such as ImageNet [1], [2]. For instance, ResNet shows a prominent recognition accuracy of 96.43%, which surpasses human beings (94.9%). Following the trend, when going deeper in DNNs (e.g. ResNet employs 18-1001 layers), memory/computational resources and their communication have faced inevitable limitations. This can be interpreted as "DNN power- and memory-wall" [3], leading to the development of different approaches to enhance the DNN efficiency at either algorithm or hardware level.

In the last two decades, Processing-in-Memory (PIM) architectures, as a potentially viable way to solve the memory wall challenge, have been widely explored by existing works [4]–[12]. The key concept behind PIM is to realize certain operations in memory by leveraging the inherent parallel computing mechanism and exploiting large internal memory bandwidth. It could lead to remarkable savings in off-chip data communication energy and latency. Ideally, the bulk bit-wise operations in a PIM architecture is expected to support not only DNN acceleration but also other data-intensive applications, such as graph processing, data encryption, etc. [9], [13].

For DNN acceleration, analog resistive crossbar memory, as one of the most popular memory array structure, has drawn great interest owing to its high memory accessing bandwidth and *in-situ* computing capability. More importantly, its current-mode weighted summation operation intrinsically matches the dominant Multiplication-and-Accumulation (MAC) in the artificial neural network, making it one of the most promising candidates as the basic computing unit for neural network accelerator design [6]. For example, ISAAC [14] architecture improves throughput and energy by 14.8× and 5.5×, respectively, relative to a well-known ASIC architecture. PipeLayer [15] achieves the speedup and energy saving of 42.45× and 7.17×, respectively, compared with a GPU platform on average. However, many non-ideal effects, such as IR-drop (i.e., wire resistance), Stuck-At-Fault (SAF), thermal noise, shot and random telegraph noise [16], are hampering the progress of real hardware implementation of large-scale DNNs on ReRAM crossbar-based accelerators. Many recent works have investigated such issues with either hardware or software solutions [11], [17].

As an alternative solution to realize massive MAC and memory operations in DNN deployments, researchers have come up with quantized/binarized DNNs, through constraining weights and activations of DNN to be quantized/ binarized in forward propagation [18], [19]. These modifications convert the conventional MAC operation to much simpler bulk bit-wise operations (based addition/subtraction [20] or comparison [10]) that can be accelerated in the content of existing digital memories. For example, Neural Cache [21], as an SRAM-based platform, improves inference latency by 18.3× over the state-of-the-art multicore CPU (Xeon E5), 7.7× over server class GPU. DRISA [7], as a DRAM-based platform, employs 3T1C- and 1T1C-based computing mechanisms and achieves 7.7× speedup and 15× better energy-efficiency over GPUs for DNN accelerations. CMP-PIM [10], as an MRAM-based platform achieves $\sim$ 10× better energy efficiency compared to DNN-ReRAM accelerator.

While the respective benefits of the aforementioned DNN acceleration-in-memory approaches (i.e. analog and digital) are well known, it still lacks cross-technology comparison and analysis. In this paper, we first briefly review some of existing

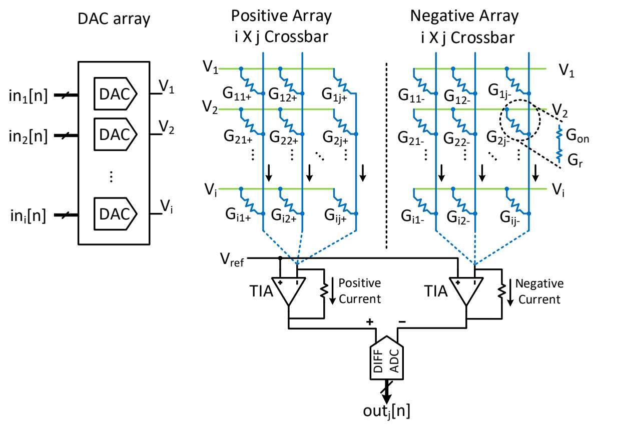


Fig. 1. Hardware implementation of a single $M \times M$ ReRAM crossbar array pair (positive and negative array) as an analog dot-product engine [16].

PIM platforms and then build a comprehensive cross-layer evaluation platform to quantitatively compare and analyze the analog and digital approaches for DNN acceleration-in-memory schemes.

II. Acceleration Based on Analog Crossbar

The primary computation performed by an analog ReRAM crossbar is the current-mode weighted summation operation (i.e., dot-product), where the architecture of the crossbar and its peripheral circuits are described in Fig. 1. Note that the positive and negative array setup is widely used in crossbar-based dot-product engine [16], [22]–[24] for performing convolution computation with positive and negative kernel values.

As shown in Fig. 1, the n-bit binary bit-strings $\text{in}_\text{i}[\text{n}]$ are the inputs to the crossbar array, which is first converted by the digital-to-analog converter (DAC) array into voltages V_i. Since the reference voltage V_ref is set to $V_\text{DD}/2$, the current forward into the differential ADC in the j-th column pair (i.e., two corresponding columns in the positive and the negative array) can be described as:

$$I_{\text{ADC},j} = \sum_{i=1}^{M} \left((V_i - V_\text{ref}) \cdot (G^{+}_{i,j} - G^{-}_{i,j}) \right) \quad (1)$$

where $G^{\pm}_{i,j}$ is the conductance of ReRAM cell indexed by i and j in the positive and negative array respectively. As can be seen, Eq. (1) performs the dot-product computation between two vectors $\boldsymbol{V} - V_\text{ref}$ and $\boldsymbol{G}^{+}_{:,j} - \boldsymbol{G}^{-}_{:,j}$. However, for using the ReRAM crossbar array to accelerate the dot-product computation in DNN, a software-hardware co-design is essential, since mapping the DNN parameters into the crossbar-based accelerator requires a series of signal conversions as introduced in [15], [16].

III. Acceleration Based on Digital Memories

In this section, we briefly review some recent digital PIM platforms supported by different memory technologies (i.e, DRAM, SRAM, MRAM, and ReRAM), where their benchmarking method and results are reported in Section IV and V, respectively. To accelerate DNN leveraging digital memories, the network parameters (inputs/weights) parameters are first

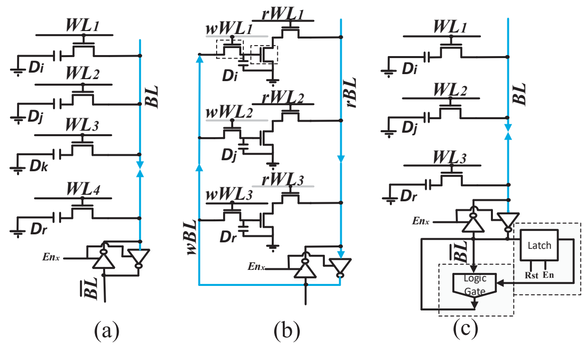


Fig. 2. (a) Ambit's TRA [8], (b) DRISA's 3T1C [7], (c) DRISA's 1T1C-logic [7]. Glossary- D_i/D_j: input rows data, D_k: initialized row data, D_r result row data.

quantized to low bit-width representations [7], [10], [12], [21], which converts a MAC operation into much simpler bulk bit-wise operations such as addition or comparison. Then, the quantized network can be readily mapped and accelerated using bulk bit-wise operations of digital computational memory.

A. DRAM

To realize bulk bit-wise operations on a DRAM platform, *Ambit* [8] extends the copy-initialization idea of *RowClone* [25] by implementing 3-input majority (`Maj3`)-based operations in memory, through issuing the `ACTIVATE` command to three rows simultaneously followed by a single `PRECHARGE` command, so-called *Triple Row Activation* (TRA) mechanism. As shown in Fig. 2a, considering one row as the control initialized by D_k= 0/1, Ambit can readily implement in-memory `AND2`/`OR2` in addition to `Maj3` functions via the charge sharing among the activated cells (i.e., D_k, D_i and D_j). Then, the computed result is written back to D_r cell within the same Bit-Line (*BL*). Ambit also leverages the TRA mechanism along with Dual Contact Cells (DCC) [8] to realize the complementary functions. However, though Ambit shows only 1% area increase over the commodity DRAM chip, it suffers from multi-cycle PIM operations while implementing more complex TRA-based functions (e.g., `XOR2`/`XNOR2`). Besides, the DRAM platforms suffer from the initial data overwritten problem, which in turn imposes long-latency in-memory operations. For example, given R=A*op*B function (*op* $\in$ `AND2`/`OR2`), the TRA-based method [8] takes 4 consecutive steps to calculate one result: (1) RowClone data of row A to row D_i (Copying the first operand to a different row to avoid data-overwritten); (2) RowClone of row B to D_j; (3) RowClone of ctrl row to D_k (Copying initialized control row to a computation row); (4) TRA and RowClone data of row D_i to R row (Computation and Writing-back the result).

DRISA-3T1C method [7] utilizes the early 3-transistor DRAM design, where each DRAM cell is controlled by two separated read/write access transistors, and an additional transistor to decouple the capacitor from the read bit-line (rBL), as shown in Fig. 2b. This additional transistor connects the two DRAM cells on the rBL in a NOR style, which naturally performs functionally-complete `NOR2` function. Nevertheless, *DRISA*-3T1C imposes very large area overhead (2T per cell)

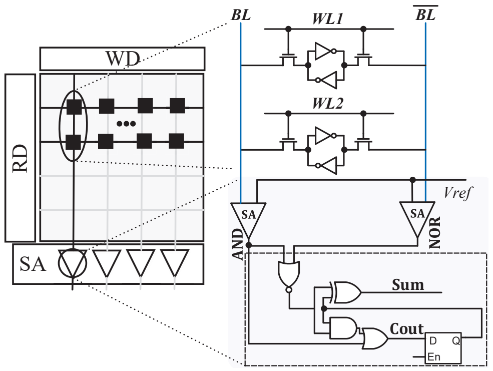


Fig. 3. Neural Cache [21] as a processing-in-SRAM platform. Note that this platform is developed based on Compute Cache [26].

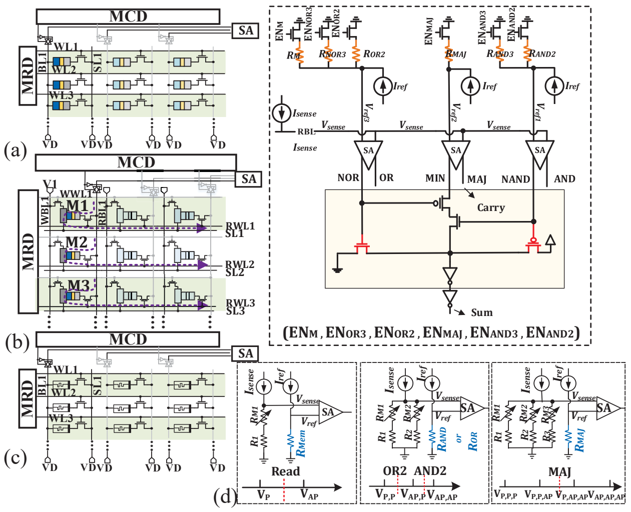


Fig. 4. GraphS [30] as a processing-in-resistive memory platform: (a) STT-MRAM implementation, (b) SOT-MRAM implementation, (c) ReRAM implementation, (d) Reconfigurable SA and the idea of voltage comparison between sense voltage and reference voltage for memory read, 2-input logic, and 3-input logic.

compared to conventional DRAM array, and still requires multi-cycle operations for complex logic operations. *DRISA*-1T1C method [7] offers to perform PIM through upgrading the sense amplifier unit by adding a CMOS logic gate in conjunction with a latch, as depicted in Fig. 2c. Thanks to the add-on circuitry, such an inherently-multi-cycle operation can enhance the performance of a single function in two consecutive cycles. In the first cycle, D_i is read out and stored in the latch. Then, in the second cycle, D_j is sensed to perform the computation. However, this design imposes excessive cycles to implement other logic functions and at least 12 transistors to each sense amplifier.

B. SRAM

To realize bulk bit-wise operations in an SRAM array, the researchers have developed various solutions, through modifying either the standard SRAM cell [27]–[29] or the sense amplifier designs [21], [26]. *Compute cache* [26] is designed to support several simple operations (e.g., bit-wise logic and copy). It simultaneously activates two rows which in turn an AND2 operation is performed by sensing BL. Similarly, a NOR2 operation can be performed by sensing bit-line complement ($\overline{BL}$). As shown in Fig. 3, the *Neural Cache* [21] as an extension to Compute cache, supports even more complex operations such as addition by adding logic gates into SA and activating the corresponding WLs. Based on this, the Neural Cache platform is capable of fully processing different DNN layers including convolutional, fully connected, and pooling layers in-cache. In addition, it also supports quantization in-cache.

C. NVM

There are many of PIM accelerators representing either reconfigurable platforms or application-specific logic designs within Non-Volatile Memories (NVMs) [6], [9], [13]. Due to space limit, we take one NVM based in-memory logic design introduced in GraphS [30] as an example to perform in-memory computation within different NVM technologies. Fig. 4a-c shows the computational sub-array architecture of GraphS [30], implemented by these technologies, respectively. The architecture mainly consists of Memory Row Decoder (MRD), Memory Column Decoder (MCD), and Sense Amplifier (SA), and can be adjusted by a Ctrl unit to work in a dual mode that performs both memory write/read and in-memory digital logic operations. Each STT-MRAM, SOT-MRAM, and ReRAM cells located in the computational sub-arrays is associated with the Word Lines (WL) and Bit Lines (BL) to perform typical memory read/write operations. In addition, the computational sub-array is designed to perform bulk bit-wise operations between in-memory operands using two distinct methods referred to as 2-row activation and 3-column activation. The main ideas behind the 2-row and 3-column activation methods are to perform bulk bit-wise in-memory (N)AND/(N)OR operation and in-memory addition/subtraction, respectively. The reconfigurable SA, as depicted in Fig. 4d, consists of three sub-SAs and a total of six reference-resistance branches that can be selected by enable bits (EN_M, EN_{OR3}, EN_{OR2}, EN_{MAJ}, EN_{AND3}, EN_{AND2}) by the sub-array's Ctrl to realize the memory and computation schemes. Such reconfigurable SA could implement memory read and one-threshold based logic functions by only activating one enable at a time, e.g., by setting EN_{AND2} to '1', 2-input AND/NAND logic can be readily implemented between operands located in the same bit line. Meanwhile, by activating two or three enables at a time, two or three logic functions can be simultaneously implemented and further used to generate complex logic functions like XOR3/XNOR3 and even full-adder in a single memory cycle. The key idea to perform memory read and bit-line computing is to choose different thresholds (references) when sensing the selected memory cell(s). The idea of voltage comparison for memory read and computation is shown in Fig. 4d, as well, the selected cells are addressed to generate a sense voltage (V_{sense}), which will be compared with selected reference voltage generated by enable bits. Now, if the path resistance is higher (/lower) than the reference resistance, the SA produces high (/low) voltage indicating logic '1' (/'0').

TABLE I

SIMULATION RESULTS FOR DIFFERENT PROCESSING-IN-MEMORY ACCELERATORS. IN AREA PART, M DENOTES MEMORY DIE AREA, AND C DENOTES COMPUTATION AREA OVERHEAD. (ISO-CAPACITY: 32MBIT-SINGLE BANK, DATA WIDTH: 512-BIT)

	Digital					Analog
Metrics	SOT-MRAM[†]	STT-MRAM[‡]	ReRAM[†]	SRAM[*]	DRAM[§]	ReRAM[**]
Non-volatility	Yes	Yes	Yes	No	No	Yes
Area (mm^2)	M: 7.06 C: 0.3	M: 2.14 C: 0.3	M: 3.92 C: 0.4	M: 10.38 C: 0.5	M: 4.53 C: 0.04	M: 3.34 C: 2.5
Read Latency (ns)	2.85	1.90	1.65	2.9	3.4	1.48
Write Latency (ns)	2.59	5.29	19.8	2.7	3.4	20.9
Read Dynamic Energy (nJ)	0.57	0.37	0.76	0.34	0.66	0.38
Write Dynamic Energy (nJ)	0.66	0.67	2.9	0.38	0.66	2.7
(N)AND/(N)OR Computation Energy (nJ)	0.64	0.46	1.13	0.59	0.75	1.96 per MAC
Full Adder Computation Energy (nJ)	1.92	1.59	3.4	1.18	11.25	
Leakage Power (mW)	550	410.2	362.4	5243	335.5	587.6
Endurance	$10^{10}-10^{15}$	$10^{10}-10^{15}$	$10^{5}-10^{10}$	Unlimited	10^{15}	$10^{5}-10^{10}$
Data over-written issue	No	No	No	No	Yes	No

[†]implemented based on [30]. [‡]implemented based on [9]. [*]implemented based on [21]. [§]implemented based on [8]. [**]implemented based on [22].

IV. PROPOSED BOTTOM-UP EVALUATION FRAMEWORK

To perform the cross-technology comparison among aforementioned PIM techniques, we have developed a comprehensive bottom-up cross-layer framework shown in Fig. 5.

1- For *Device level*, the device-level model was first extracted from different assessments and models. For example, we jointly use the Non-Equilibrium Green's Function (NEGF) and Landau-Lifshitz-Gilbert (LLG) equations to model STT-MRAM and SOT-MRAM bitcell (indicated under MRAM in Fig. 5) [31], [32]. We used the default ReRAM and SRAM .cell configuration of NVSim [33]. Moreover, DRAM cell parameters were taken and scaled from Rambus [34].

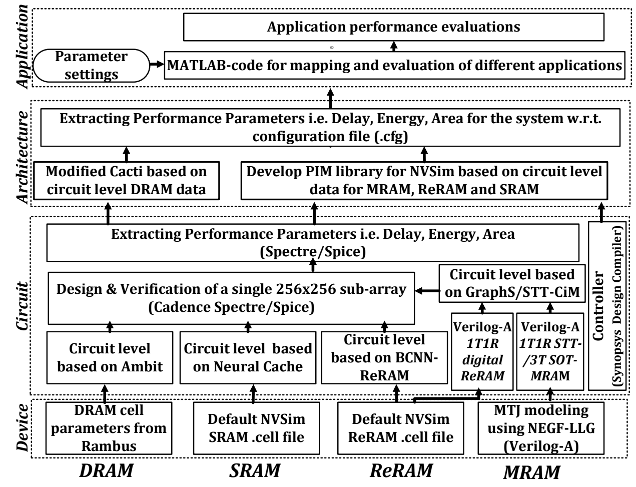
2- For *Circuit level* simulation, we developed a 256×256 memory sub-array for each technology with peripheral circuity (SA, MRD, MCD, etc.) based on an existing PIM platform (GraphS [30] logic design for SOT-MRAM and digital ReRAM; STT-CiM [9] as the STT-MRAM design; BCNN-ReRAM [22] design for analog ReRAM crossbar; Neural Cache [21] design for SRAM; Ambit [8] design for DRAM). The memory sub-arrays are simulated in Cadence Spectre with 45nm NCSU Product Development Kit (PDK) library [35] to verify the PIM's circuit functionality and achieve the circuit performance parameters. The memory controller circuits for all platforms are synthesized by Design Compiler [36] with the same 45nm industry library.

3- For *Architecture level*, we developed PIM libraries for each platforms on top of NVSim [33] and Cacti [37] based on device/circuit level data and extract the performance data (i.e. delay, energy, area) for different PIM platforms w.r.t. a single input memory configuration file (.cfg).

4- For *Application level* simulations, a behavioral-level simulator was developed in Matlab taking architecture-level results to calculate the latency, energy, and area that different PIM platforms spend on DNN acceleration task.

V. PERFORMANCE ANALYSIS

In this section, we perform two distinct experiments under ISO-Capacity and ISO-Computation constraints to compare analog and digital PIM approaches.



Application
Application performance evaluations
Parameter settings
MATLAB-code for mapping and evaluation of different applications
Architecture
Extracting Performance Parameters i.e. Delay, Energy, Area for the system w.r.t. configuration file (.cfg)
Modified Cacti based on circuit level DRAM data
Develop PIM library for NVSim based on circuit level data for MRAM, ReRAM and SRAM
Circuit
Extracting Performance Parameters i.e. Delay, Energy, Area (Spectre/Spice)
Design & Verification of a single 256x256 sub-array (Cadence Spectre/Spice)
Circuit level based on GraphS/STT-CiM
Controller (Synopsys Design Compiler)
Circuit level based on Ambit
Circuit level based on Neural Cache
Circuit level based on BCNN-ReRAM
Verilog-A 1T1R digital ReRAM
Verilog-A 1T1R STT-/3T SOT-MRAM
Device
DRAM cell parameters from Rambus
Default NVSim SRAM .cell file
Default NVSim ReRAM .cell file
MTJ modeling using NEGF-LLG (Verilog-A)
DRAM
SRAM
ReRAM
MRAM

Fig. 5. The bottom-up evaluation framework developed for PIM platform evaluation.

A. ISO-Memory-Capacity Comparison

We first study the performance of digital and analog PIM platforms with an ISO-memory-capacity constraint. We developed a 32Mb, single bank unit based on digital (SOT-MRAM, STT-MRAM, ReRAM, SRAM, and DRAM) and analog ReRAM crossbar. Table I reports eleven performance parameters for each platform. We list our observations below:

a) Area: We divide the area metric into two parts: memory die area (*M*), and computational area (*C*) which includes controller, modified decoder, SA, 8-bit ADC for the relevant analog ReRAM crossbar, etc. In terms of memory die area, the digital PIM platforms impose a relatively larger area than that of analog ReRAM cross-bar exept for STT-MRAM design [9]. However, if we take the computational area into account, the ReRAM crossbar consumes 2.5 mm^2, which is much larger than that of digital counterparts, such as digital ReRAM (0.4 mm^2). Accordingly, we can define a memory to computational area ratio as M/C. The M/C ratio equals to 23.53, for SOT-MRAM based PIM, while the analog ReRAM crossbar shows a ratio of 1.33. The low M/C ratio of ReRAM crossbar is the consequence of large peripheral circuit's overhead, such as buffers and DAC/ADC, which contributes more than 85%

TABLE II
ESTIMATED ROW PERFORMANCE OF DIFFERENT PIMS WITHOUT PARALLELISM TECHNIQUES

	Digital					Analog
Parameters	SOT-MRAM	STT-MRAM	ReRAM	SRAM	DRAM	ReRAM
Area (mm^2) (memory + logic)	0.018 $\sim$(0.0172+0.0008)	$\sim$0.012 $\sim$(0.011+0.0008)	0.0097 $\sim$(0.009+0.0007)	0.64 $\sim$(0.608+0.032)	0.16 $\sim$(0.158 + 0.002)	0.06 $\sim$(0.011+0.049)
Energy (μJ) (write-back+read-based Ops)	0.85 $\sim$(0.31+0.54)	0.78 $\sim$(0.25+0.53)	1.9 $\sim$(0.75+1.15)	1.6 $\sim$(0.42+1.18)	2.1 $\sim$(0.8+1.3)	13.5 $\sim$(0+13.5)
Latency (ms)	0.9	1.8	1.3	0.7	13.5	5.8

of the computational area [6], [22]. Furthermore, according to the results reported in Table I, the STT-MRAM and SRAM platforms occupy the smallest and the largest overall area, respectively, in comparison to other PIM counterparts.

b) Latency: As listed in Table I, the analog ReRAM crossbar achieves the shortest read latency (1.48 ns) as compared with digital platforms, but it has the longest write latency (20.9 ns). The SOT-MRAM platform achieves the shortest write latency compared to other technologies, and has a higher endurance (10^{10}-10^{15}) compared to ReRAM-based platforms.

c) Energy: We can see that SOT-MRAM and STT-MRAM platforms consume the smallest write dynamic energy among all the NVM platforms, due to its intrinsically low-power device operation, while SRAM achieves the smallest read and write energy compared to all the platforms. The analog ReRAM crossbar achieves a close-to-SRAM read dynamic energy, but it consumes a large write dynamic energy. In terms of computational energy, for digital platforms, it is measured based on the PIM's capability to perform (N)AND/(N)OR and full adder functions. As seen from Table I, the STT-MRAM [9] and SRAM [21] PIM respectively consume the smallest computational energy compared to different technologies to perform different operations, where SOT-MRAM stands as the third most energy-efficient platform. Note that, although the DRAM PIM design based on Ambit [8] consumes 0.75 nJ to perform (N)AND/(N)OR based TRA mechanism, it requires over 14 memory cycles to perform the addition operation to avoid overwriting data, which leads to much higher energy consumption compared to other platforms. For the analog crossbar, we report the computational energy per MAC, which is comparable with addition operation in digital SOT-MRAM platform. In terms of leakage power consumption, the digital ReRAM along with DRAM can be observed as relatively more power efficient platforms. Moreover, we observe that the SRAM platform consumes $\sim$14.5$\times$ and $\sim$9$\times$ more power as compared to digital and analog ReRAM, respectively.

B. ISO-Computation Comparison

In this subsection, we evaluate the performance of the digital and analog PIM platforms for DNN acceleration. Hereby, we take the classical LeNet-5 [38] as a simple example, to perform the hand written digit classification task with MNIST dataset. For properly mapping the target DNN into the PIM, offline training of the LeNet-5 network is conducted with weight and activation quantization, following the methods proposed in [18], [39]. A description model of each platform developed based on the data reported in Table I is employed in the application level DNN simulator. For fair hardware comparison, we use the bit-width configuration of [1:8] for [Weight:Activation], although ReRAM crossbar based accelerator supports higher weight bit-width ($>$ 1 bit) with better DNN performance (i.e., classification accuracy in this work). No quantization is applied in the first and last layer of DNN, and the full-precision computations are also handled by the PIM-based accelerator. For the sake of simplicity, we only reflect the estimated performance results (area, energy, latency) of convolutional layers.

a) Area: Contrary to the approach used to report the area in Table I, we leverage the method presented in [10], [22] to report the results. Specifically, we consider the area overhead due to computation by calculating the number of crossbars or sub-arrays. Table II reports the area for digital and analog PIM platforms by dividing it into the memory and logic parts. We observe that the digital ReRAM and STT-MRAM platforms require the smallest area compared to other platforms, respectively, mainly owning to their single transistor cell structure. It is noteworthy that, while the DRAM platform has one of the least die areas owning to its single-transistor cell, and owns the least computational area under ISO-capacity constraint due to its almost unchanged peripheral circuitry (1% as listed in Table I), it requires accessing to multiple sub-arrays to avoid overwriting data problem as well as fitting the network at the same time, resulting in a larger area requirement compared to NVMs. As for the analog crossbar platform, the logic part contributes $\sim$4$\times$ more than memory area. Overall, it imposes larger area than that of other digital NVM platforms due to matrix splitting and extra large add-on area overhead [6].

b) Energy: Table II also reports the energy consumption of different platforms. It can be seen that SOT-MRAM and STT-MRAM based platforms save 15.8$\times$ and 17.3$\times$ energy compared to the analog crossbar. In addition, the digital volatile memories consume much smaller energy to that of the analog platform. Therefore, from energy saving stand point, digital PIM platforms could be a better choice in comparison to the analog crossbar. Note that, for PIM platforms, all operands are assumed to be stored in memory. Unlike traditional computation, an extra intermediate data write-back is needed, which has a large effect on the overall energy and latency. Based on this, we split the reported energy into write-back and read-based logic operations energy. The write-back energy involves the energy required to write the weights or inputs into PIM plus the energy required write the computation results back to the memory for computation in the next layer. The read-based operation energy involves the read and bit-line computing energy. The analog crossbar [22] can accomplish the MAC operation without writing back the intermediate data, that's why we omit the write-back energy for this platform.

c) Latency: We summarize the latency of each platform required to process the convolutional layers of the CNN, as tabulated in Table II. According to the table, the SRAM platform is the fastest one with 0.7 ms latency. This mainly comes from its short read and write latency as well as fast two-cycle addition scheme [21]. In addition, we observe that the SOT-MRAM platform achieves 0.9 ms latency and stands as the second fastest platform. The DRAM platform shows an extremely long latency mainly due to the excessive copy operations needed to avoid overwriting data, as explained in Section III.A. The analog crossbar needs 5.8 ms to process the convolutional layers.

VI. Conclusion

To accelerate DNN in PIM platforms, the analog current-mode weighted summation operation in resistive memory crossbars intrinsically matches the dominant MAC operation in the DNN. Alternatively, latest algorithmic progression has brought competitive classification accuracy for neural networks despite constraining the network parameters to limited-bit representations, which essentially converts the MAC operation to much simpler bulk bit-wise operations such as addition or comparison that can be accelerated inside existing digital memories (e.g., SRAM, DRAM, MRAM). In this paper, we presented a comprehensive evaluation platform to quantitatively compare such analog and digital PIM platforms based on different technologies to provide a guideline to the research community. We reported our observations considering three key evaluation metrics i.e. area, energy, latency.

Acknowledgements

This work is supported in part by the National Science Foundation under Grant No. 1740126, Semiconductor Research Corporation nCORE and the Nanoelectronics Research Corporation (NERC), a wholly-owned subsidiary of the SRC, through Extremely Energy Efficient Collective Electronics (EXCEL), an SRC-NRI Nanoelectronics Research Initiative under Research Task ID 2698.004.

References

[1] L. Cavigelli *et al.*, "Accelerating real-time embedded scene labeling with convolutional networks," in *DAC, 2015 52nd ACM/IEEE*, 2015.

[2] H. H. Li *et al.*, "Looking ahead for resistive memory technology: A broad perspective on reram technology for future storage and computing," *IEEE Consumer Electronics Magazine*, pp. 94–103, 2017.

[3] X. S. Hu and M. Niemier, "Cross-layer efforts for energy-efficient computing: towards peta operations per second per watt," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, pp. 1209–1223, 2018.

[4] X. S. Hu, "A cross-layer perspective for energy efficient processing:- from beyond-cmos devices to deep learning," in *GLSVLSI*. ACM, 2018, pp. 7–7.

[5] X. Yin *et al.*, "Ferroelectric fets-based nonvolatile logic-in-memory circuits," *IEEE TVLSI*, vol. 27, pp. 159–172, 2019.

[6] P. Chi *et al.*, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *ISCA*. IEEE Press, 2016.

[7] S. Li *et al.*, "Drisa: A dram-based reconfigurable in-situ accelerator," in *Micro*. ACM, 2017, pp. 288–301.

[8] V. Seshadri *et al.*, "Ambit: In-memory accelerator for bulk bitwise operations using commodity dram technology," in *Micro*. ACM, 2017, pp. 273–287.

[9] S. Jain *et al.*, "Computing in memory with spin-transfer torque magnetic ram," *IEEE TVLSI Systems*, vol. 26, pp. 470–483, 2018.

[10] S. Angizi, Z. He *et al.*, "Cmp-pim: an energy-efficient comparator-based processing-in-memory neural network accelerator," in *Proceedings of the 55th Annual Design Automation Conference*. ACM, 2018, p. 105.

[11] S. Jain, A. Sengupta, K. Roy, and A. Raghunathan, "Rx-caffe: Framework for evaluating and training deep neural networks on resistive crossbars," *arXiv preprint arXiv:1809.00072*, 2018.

[12] S. Angizi *et al.*, "Imce: energy-efficient bit-wise in-memory convolution engine for deep neural network," in *Proceedings of the 23rd ASP-DAC*. IEEE Press, 2018, pp. 111–116.

[13] S. Li *et al.*, "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," in *2016 53nd DAC*. IEEE, 2016.

[14] A. Shafiee *et al.*, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM SIGARCH Computer Architecture News*, vol. 44, 2016.

[15] L. Song *et al.*, "Pipelayer: A pipelined reram-based accelerator for deep learning," in *HPCA*. IEEE, 2017, pp. 541–552.

[16] Z. He *et al.*, "Noise injection adaption: End-to-end reram crossbar non-ideal effect adaption for neural network mapping," *56-th Design Automation Conference (DAC)*, 2019.

[17] I. Chakraborty *et al.*, "Technology aware training in memristive neuromorphic systems for nonideal synaptic crossbars," *IEEE TETCI*, pp. 335–344, 2018.

[18] S. Zhou *et al.*, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[19] Z. He *et al.*, "Optimize deep convolutional neural network with ternarized weights and high accuracy," in *WACV*. IEEE, 2019, pp. 913–921.

[20] Z. He and D. Fan, "Simultaneously optimizing weight and quantizer of ternary neural network using truncated gaussian approximation," *CVPR*, 2019.

[21] C. Eckert *et al.*, "Neural cache: Bit-serial in-cache acceleration of deep neural networks," in *ISCA*. IEEE Press, 2018, pp. 383–396.

[22] T. Tang *et al.*, "Binary convolutional neural network on rram," in *22nd ASP-DAC*. IEEE, 2017, pp. 782–787.

[23] T.-h. Yang and M.-f. Chang, "Sense amplifier of resistive memory and operating method thereof," Jan. 22 2019, uS Patent App. 15/939,262.

[24] M. Hu, H. Li *et al.*, "Memristor crossbar-based neuromorphic computing system: A case study," *IEEE transactions on neural networks and learning systems*, pp. 1864–1878, 2014.

[25] V. Seshadri *et al.*, "Rowclone: fast and energy-efficient in-dram bulk data copy and initialization," in *Micro*. ACM, 2013, pp. 185–197.

[26] S. Aga *et al.*, "Compute caches," in *High Performance Computer Architecture (HPCA), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 481–492.

[27] R. Liu *et al.*, "Parallelizing sram arrays with customized bit-cell for binary neural networks," in *DAC*. ACM, 2018, p. 21.

[28] A. Agrawal *et al.*, "X-sram: Enabling in-memory boolean computations in cmos static random access memories," *IEEE TCASI*, pp. 1–14, 2018.

[29] S. Srinivasa *et al.*, "Robin: Monolithic-3d sram for enhanced robustness with in-memory computation support," *IEEE TCASI, year=2019, publisher=IEEE.*

[30] S. Angizi *et al.*, "Graphs: A graph processing accelerator leveraging sot-mram," in *Design, Automation and Test in Europe (DATE)*. IEEE Press, 2019.

[31] X. Fong *et al.*, "Spin-transfer torque devices for logic and memory: Prospects and perspectives," *IEEE TCAD*, vol. 35, 2016.

[32] X. Fong, S. K. Gupta *et al.*, "Knack: A hybrid spin-charge mixed-mode simulator for evaluating different genres of spin-transfer torque mram bit-cells," in *SISPAD*. IEEE, 2011, pp. 51–54.

[33] X. Dong *et al.*, "Nvsim: A circuit-level performance, energy, and area model for emerging non-volatile memory," in *Emerging Memory Technologies*. Springer, 2014, pp. 15–50.

[34] . DRAM Power Model. https://www.rambus.com/energy/.

[35] (2011) Ncsu eda freepdk45. [Online]. Available: http://www.eda.ncsu.edu/wiki/FreePDK45:Contents

[36] S. D. C. P. V. . Synopsys, Inc.

[37] K. Chen *et al.*, "Cacti-3dd: Architecture-level modeling for 3d die-stacked dram main memory," in *DATE, 2012*. IEEE, 2012, pp. 33–38.

[38] Y. LeCun *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[39] M. Courbariaux *et al.*, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems*, 2015, pp. 3123–3131.