Processing-in-Memory Acceleration of MAC-based Applications Using Residue Number System: A Comparative Study

Shaahin Angizi[†], Arman Roohi[‡], MohammadReza Taheri[*], Deliang Fan[†]
[†] School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85281, USA
[‡] Department of Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln, USA
[*] Independent Researcher
The first two authors contributed equally
aroohi@unl.edu

ABSTRACT

Processing-in-memory (PIM) has raised as a viable solution for the memory wall crisis and has attracted great interest in accelerating computationally intensive AI applications ranging from filtering to complex neural networks. In this paper, we try to take advantage of both PIM and the residue number system (RNS) as an alternative for the conventional binary number representation to accelerate multiplication-and-accumulations (MACs), primary operations of target applications. The PIM architecture utilizes the maximum internal bandwidth of memory chips to realize a local and parallel computation to eliminates the off-chip data transfer. Moreover, RNS limits inter-digit carry propagation by performing arithmetic operations on small residues independently and in parallel. Thus, we develop a PIM-RNS, entitled **PRIMS**, and analyze the potential of intertwining PIM architecture with the inherent parallelism of the RNS arithmetic to delineate the opportunities and challenges. To this end, we build a comprehensive device-to-architecture evaluation framework to quantitatively study this problem considering the impact of PIM technology for a well-known three-moduli set as a case study.

CCS CONCEPTS

• **Hardware** → **Memory and dense storage**;

KEYWORDS

processing-in-Memory; residue number system; multiplication-and-accumulation

ACM Reference Format:
Shaahin Angizi[†], Arman Roohi[‡], MohammadReza Taheri[*], Deliang Fan[†]. 2021. Processing-in-Memory Acceleration of MAC-based Applications Using Residue Number System: A Comparative Study . In *Proceedings of the Great Lakes Symposium on VLSI 2021 (GLSVLSI '21), June 22–25, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3453688.3461529

GLSVLSI '21, June 22–25, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8393-6/21/06…$15.00
https://doi.org/10.1145/3453688.3461529

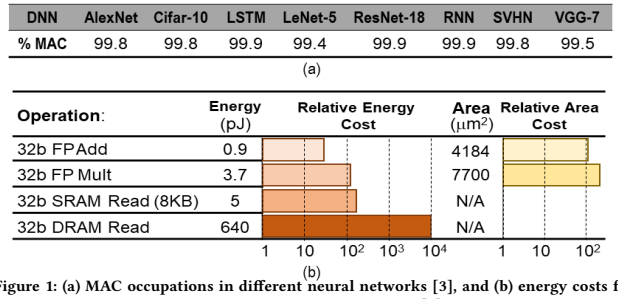

Figure 1: (a) MAC occupations in different neural networks [3], and (b) energy costs for addition, multiplication, and read operations in 45nm 0.9V [4].

1 INTRODUCTION

Since we have hit multiple walls in processor and system design, including power and memory wall, instruction-level parallelism wall, and recently network wall, achieving significant improvements (order-of-magnitude) in the performance of general-purpose architectures is challenging and crucial. Therefore, with the high demands on high performance and energy-efficient designs for resource-limited devices such as edge-AI, domain-specific accelerator designs have attained worldwide attention in both the research community [1, 2] as well the industry such as Graphcore IPU, Google TPU, and Cerebras. On the one hand, numerous signal processing and artificial intelligence applications, including natural language processing, machine learning platforms, speech, pattern, and emotion recognition, require intensive Multiply-Accumulate (MAC) operations. For example, in Fig. 1a, the main portions (>99%) of various deep neural networks (DNNs), as a particular subset of machine learning, are occupied by MAC operations, while MAC units are one of the most energy-hungry building blocks. On the other hand, the aforementioned applications' energy consumption is usually dominated by data accesses to off-chip memory and on-chip storage existed in conventional von Neumann architectures, shown in Fig. 1b. It means MAC units' processing demands face serious challenges for their tractability in terms of memory resources. These have been motivating the development of alternative approaches in both organization and hardware domains to improve MAC design efficiency.

Processing-in-Memory (PIM) architecture has been well designed and developed for different applications [5–7] as a possible way to address the challenges mentioned earlier. The main idea of PIM is to incorporate logic computation within memory units such that memory can process data internally that provides inherent parallel

processing mechanisms exploiting large internal memory bandwidth. Considering the larger memory capacities of DRAM and off-chip data movement reduction as opposed to SRAM-based PIM, processing-in-DRAM platforms [6–9] have gained much more attention as a promising accelerator for different applications. DRISA [7] presents two alternative 3T1C- and 1T1C-based PIM techniques as CNN's accelerators and improves speed and energy-efficiency by 7.7× and 15× over GPUs, respectively. Nevertheless, the implementation of X(N)OR and addition -based tasks faces multiple challenges, making these platforms inefficient. This inefficiency comes from the intrinsic logic complexity of X(N)OR operation that is implemented by majority/AND/OR-based multi-cycle operations and required row initialization methods.

In addition to the PIM platform as a non-von Neumann architecture, using the Residue Number System (RNS) as an unconventional number system can lead to further performance improvements in a vast range of computationally intensive applications. RNS is a non-weighted number system whose unique features and number-theoretic properties can enhance the performance of special types of computations. Compared to the traditional weighted number system that tolerates the long carry propagation chain in the arithmetic operations, RNS restricts the inter-digit carry propagation by carrying out operations like addition and multiplication on relatively small integers in several parallel and individual channels. This number representation leads to shortening the carry propagation chain of addition and reducing partial product matrices' size in multiplication, which results in high performance and low power computations in a vast range of applications. Although these promising features offer high-speed and low-power VLSI implementation in many applications like DSPs and cryptography, RNS is limited by complex and costly intermodulo operations, such as magnitude comparison, sign detection, and reverse conversion [10, 11]. Thanks to the high rate of internal arithmetic operations to forward/reverse conversions in the RNS realization of MAC-based application, herein, leveraging RNS-based computation leads to performance enhancement.

To comprehensive examine the impact of both PIM implementations and the RNS in the MAC-based structures, in summary, our major contributions in this paper can be listed as follows: 1) We introduce a PIM-based MAC engine with RNS number representation, namely PRIMS[1], to reduce the power dissipation and enhance the operational frequency. 2) Comprehensive energy and delay analyses are performed regarding various volatile and non-volatile memory technologies. 3) We developed an evaluation framework to study PIM implementations' impact in the RNS domain, from device-level upwards to architecture-level.

2 BACKGROUND

2.1 Residue Number System in a Nutshell

Residue Number System (RNS) is a non-weighted number system initially proposed during the late 1950s by Garner. The unique feature and number-theoretic properties of RNS can boost up the speed of special-purpose computations. Contrary to the conventional weighted number system, which is suffering from the inevitable lengthy carry propagation chain, RNS limits inter-digit carry propagation. An RNS composed of b relative prime numbers such as $\{m_1, m_2, \ldots, m_b\}$, where $b \geq 2$ is the cardinality of the system, and m_i is called the moduli. In such RNS, any integer X with $0 \leq X < M$ is uniquely represented by a b-tuple of non-negative integers such as $(x_1, x_2, ..., x_b)$, where $x_i = X \bmod m_i$ is called residue, and $M = \prod_{i=1}^{b} m_i$ is the dynamic range of the system [10, 12].

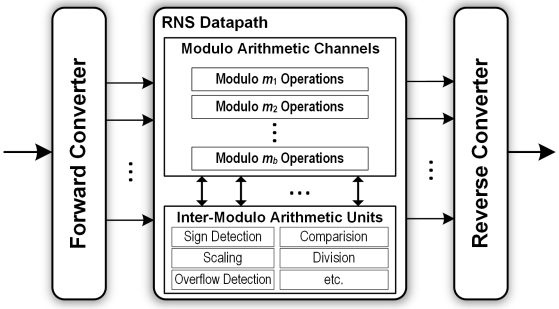

Figure 2: The typical Architecture of an RNS.

Fig. 2 depicts the typical architecture of an RNS, which is consists of three main blocks: Forward converter, RNS datapath, and reverse converter. As it can be seen from Fig. 2, despite the fantastic benefits of RNS, it faces some overhead for interfacing with binary-weighted number system by employing the forward and reverse converters. The forward converter's role is computing the residue for each moduli channel based on the received weighted numbers as input data. In comparison, the reverse converter translates the obtained results from the datapath computations to the weighted number representation. The arithmetic operations are carried out in the RNS datapath. These operations can be categorized into two groups. The main and the primary group consist of modulo m_i operations $\otimes \in \{+, -, \times\}$ with $i \in \{1, 2, \ldots, b\}$, that are executed parallel and independently in modulo arithmetic channels. Let X, Y, and R be three integers, where R is computed by carrying out the arithmetic operation $\otimes$ upon X and Y, i.e. $R = X \otimes Y$. The isomorphic relations of these computations, between the weighted number representation and their RNS counterpart with a given moduli set $\{m_1, m_2, \ldots, m_b\}$ can be expressed as:

$$R \equiv (r_1, r_2, \ldots, r_b) \tag{1}$$

$$X \otimes Y \equiv \left(\langle x_1 \otimes y_1 \rangle_{m_1}, \langle x_2 \otimes y_2 \rangle_{m_2}, \ldots, \langle x_b \otimes y_b \rangle_{m_b} \right) \tag{2}$$

where $(x_1, x_2, \ldots, x_b)$, $(y_1, y_2, \ldots, y_b)$, and $(r_1, r_2, \ldots, r_b)$ are the residue representation of X, Y, and R, respectively. Arithmetic operations are not limited to multiplication, addition, and subtraction and may include operations such as sign identification, magnitude comparison, and scaling, depending on the application. These operations impose extreme complexity into RNS due to their costly inter-modulo computations, which cannot independently perform parallel. Albeit energy-efficient implementations for inter-modulo operations are vital and challenging, it is out of the scope of this work.

2.2 RNS-based MAC Unit

The MAC unit can be regarded as one of the substantial computing blocks in DSP applications. The architecture of a conventional

[1]Inspired by the name of the greedy algorithm, "Prim's."

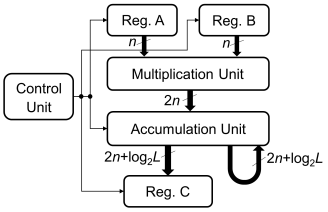

Figure 3: Conventional schematic of MAC unit, where $C = \sum_{j=1}^{L} A_j \times B_j$.

MAC unit is shown in Fig. 3, which accumulates the result of a sequence of multiplication. According to Fig. 3, an n-bit MAC unit includes an n-bit multiplier, a $(2n + log_2 L)$-bit adder, and a control unit, where L is the length of accumulation sequence. The multiplier includes three main components: a partial product generator, a partial product reduction unit, and a carry propagate adder that performs the final 2-operand addition. To accelerate the operation of the MAC unit, a vast range of studies in several design levels are conducted in the literature. One of these solutions is adopting the RNS in the realization of MAC-based applications. Due to the noteworthy features of RNS, like highly parallel arithmetic operation, this number representation has gained attraction for a wide range of compute-intensive MAC-based applications.

To implement MAC operation in the RNS domain, numbers A and B as the inputs of MAC unit are represented in residue form considering the target moduli set $\{m_1, m_2, \ldots, m_b\}$. Therefore, by adopting RNS, the multiplication and accumulation of typical MAC unit in Fig. 3 can be conducted in b independent and parallel datapaths in separate residue channels, as shown in Fig. 4. The modulo MAC operation in each independent residue channel is carried out by using modulo multiplier and adder. The complexity and efficiency of the modulo multiplier and modulo adder of the MAC unit in each channel are highly dependent on its corresponding modulus.

Since the modulo addition and multiplication can be regarded as the vital operations in the datapath of RNS-based MAC with a significant impact on the operational efficiency, several contributions have been made to the literature on the effective realization of these arithmetic blocks. The types of selected moduli set for RNS and the topologies of the adder and multiplier structures can be counted as two influential factors in the performance of modulo adder and multiplier. Considering the first factor, special moduli in the form of 2^α, $2^\alpha - 1$, and $2^\alpha + 1$ are mostly explored for low-cost and effective hardware implementation of modulo arithmetic operations in RNS. Considering the second factor, various techniques and topologies for the hardware realization of modulo adders and multipliers are presented in the literature. Apart from these efforts, one ingenious way to accelerate the RNS-based MAC operation is breaking the memory wall in modulo operations of the MAC unit: modulo addition and multiplication. To enhance the exploitation of the parallelism provided by the RNS, we mainly focus on the RNS-based MAC acceleration using non-von Neumann architectures, i.e., processing in-memory designs. Partial product generation and partial product reduction of modulo multiplier are conducted by employing in-memory *AND* operations and *compressors*, respectively. Also, for improving the modular addition performance in the accumulator and the final step of the multiplier, in-memory

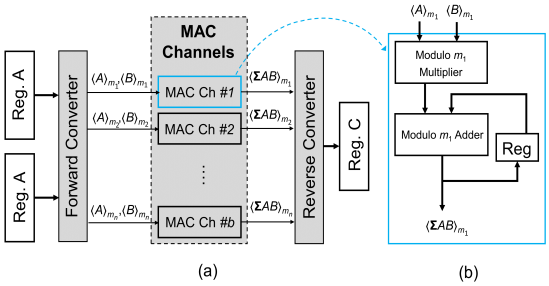

Figure 4: (a) General MAC architecture for arbitrary moduli sets, and (b) block diagram of the modulo $m1$ MAC unit.

parallel prefix structures that provide a brilliant trade-off between delay and hardware cost are utilized in this paper.

3 PRIMS ARCHITECTURE

We develop a general RNS-based MAC PIM engine called PRIMS, as shown in Fig. 5 to perform a PIM technology-dependencies study in Section 4. To take advantage of the promising features of RNS, PRIMS performs almost the entire computations in the RNS domain. Thanks to the high performance and low complexity of modulo-$(2^n + 1)$-free arithmetic operations, RNS with moduli set $\{2^{n+1} - 1, 2^n - 1, 2^n\}$ has found great popularity among researchers for boosting a variety of computation-intensive applications. This moduli set's unique theoretical properties also lead to effective hardware implementations for a variety of intermodulo operations [13]. Therefore, to validate and evaluate our PRIMS, this moduli set is chosen.

The PRIMS is developed on top of the existing main memory hierarchy by dividing every memory chip into multiple memory banks all the way into multiple sub-arrays. The PRIMS converts every memory sub-array to a potential computation core, called computational sub-array, capable of performing in-memory logic and MAC operation on the memory side depending on the PIM technology features. For the software support, we developed a particular backbone virtual machine and instruction set for parallel thread execution such that the programs could be translated at install time to the PRIMS hardware instruction set. PRIMS receives one of the PIM instructions from the memory controller and accordingly issues control signals to implement one of the in-memory functions. Besides, we designed add-on forward and backward converters, and a global buffer on the memory die to facilitate RNS processing. Figure 6 gives a clear picture of how the PRIMS data partitioning and mapping occur, considering a three-moduli RNS $\{m_1, m_2, m_3\}$. As shown in step (1), the input batch with a size of

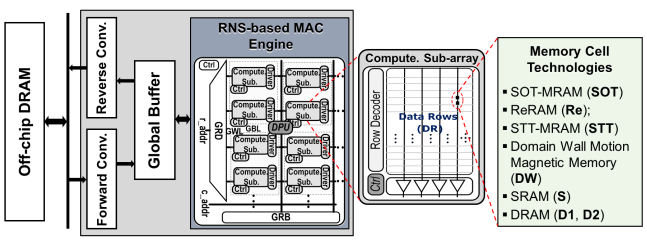

Figure 5: RNS-based MAC PIM engine (PRIMS) schematic regarding studied memory cell technologies.

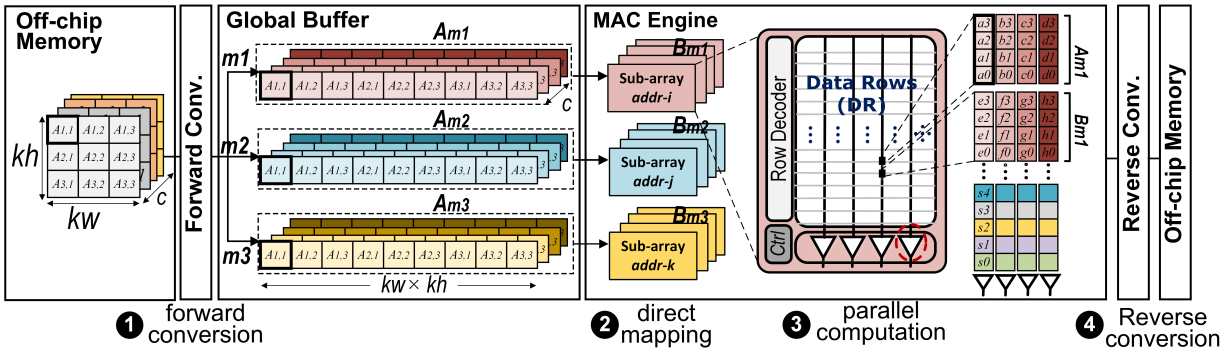

Figure 6: The multiply-accumulate operations for two 3D tensors, A and B, using RNS-based MAC engine.

kw×kh×c is fed to the Forward conv. to provide a triple of integer number three parallel channels for each integer number element corresponding to the target moduli set, represented by A_{m1}, A_{m2}, and A_{m3}. Each channel is then assigned to a certain sub-array address with the memory, where the corresponding operand batch moduli is pre-stored B_{m1}, B_{m2}, and B_{m3} in step (2). For example, sub-array i to $i+3$ are allocated to compute the MAC between A_{m1} and B_{m1} through summation implementation in step (3) parallel computation. The MAC engine of modulo 2^5-1 channel is depicted in Fig. 7. The graph structure of the modulo 2^5-1 parallel prefix adder as a primary building block of RNS-based MAC is demonstrated, wherein each building block is effectively translated into PRIMS's ISA. All the preliminary calculation square blocks could be generated in an ideal PIM platform.

4 PIM IMPACT: CHOICE OF TECHNOLOGY

As different data-intensive applications with various workload sizes and memory access patterns are expected to benefit from PIM in both cache and main memory levels, selecting the proper design for a particular application is a complex task. Besides, by choosing a PIM design, it is imperative to establish uniform evaluation conditions to make an impartial choice between available design options [14, 15]. To study the impact of the choice of NVM/VM technology in MAC engine performance, we first considered a 4 MByte memory unit and adopted the promising PIM techniques, i.e., GraphS [16] for SOT-MRAM (represented by SOT) and ReRAM (Re); STT-CiM [17] as the STT-MRAM design (STT); RIMPA [18] for Domain Wall Motion Magnetic Memory (DW); Neural Cache [19] for SRAM (S); Ambit (D1) [6] and ReDRAM (D2) [8] designs for DRAM. To perform fair technology-dependent analyses among various technologies, the mentioned arithmetic-friendly moduli set with n=5, $\{m_1 = 2^6-1, m_2 = 2^5-1, m_3 = 2^5\}$, is selected.

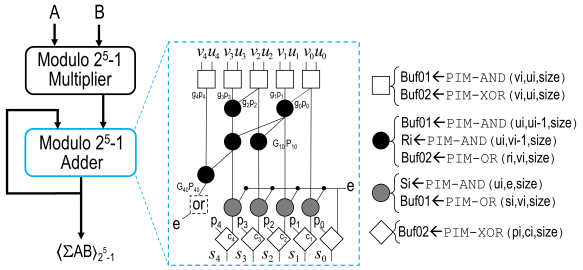

Figure 7: Schematic of a MAC unit for modulo 2^5-1. The PRIMS's instructions to process the accumulation in a computational sub-arrays.

4.1 Framework

We developed a comprehensive device-to-architecture level cross-layer framework from scratch shown in Fig. 8. At the *device level*, we jointly used the Non-Equilibrium Green's Function (NEGF) and Landau-Lifshitz-Gilbert (LLG) equations to model STT-MRAM and SOT-MRAM bitcell [20, 21]. We used the default ReRAM and SRAM .cell configuration of NVSim [22]. Moreover, DRAM cell parameters were taken and scaled from Rambus [23]. At the *circuit level*, we realized the generic digital PRIMS accelerator shown in Section 3 with the aforementioned NVM/VM technologies. At the sub-array level, the specific memory peripheral circuitry, including sense amplifier, a modified row decoder, etc. were replicated in Cadence Spectre with 45nm NCSU Product Development Kit (PDK) library [24] to extract the performance parameters such as latency and energy consumption, etc. It is worth mentioning that forward and reverse converters are exactly the same and are therefore not considered in our assessment for all architectures. At the *architecture level*, we develop a comprehensive PIM compiler and evaluation framework working with the NVSim tool for PRIMS. Given an input network structure, the framework takes the instructions from a user interface, partitions, maps the input data batches into the computational sub-arrays for three channels, and then estimates the performance of PIM platforms. We report our assessment on the choice of technology in Fig. 9.

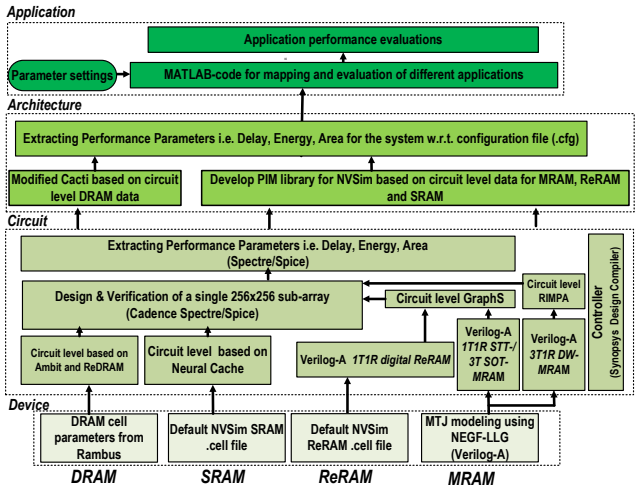

Figure 8: The bottom-up evaluation framework to determine the PIM impact.

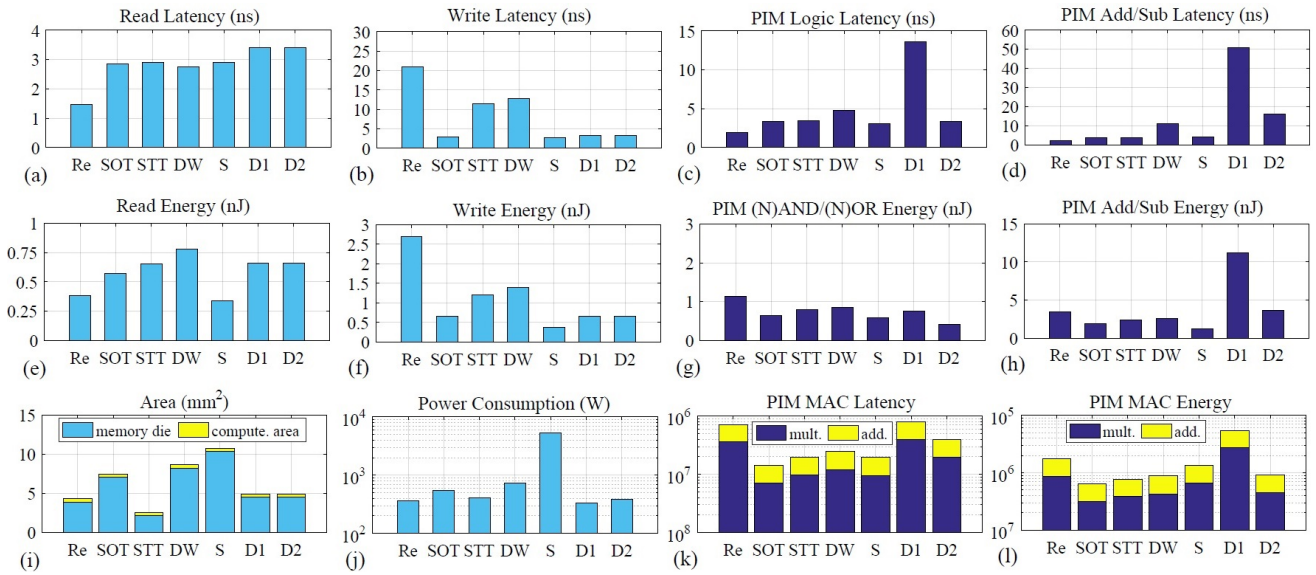

Figure 9: Technology design-dependent performance of various PIM platforms: (a) Read latency, (b) Write latency, (c) In-memory Boolean logic (N)AND/(N)OR latency, (d) In-memory addition/subtraction latency, (e) Read dynamic energy, (f) Write dynamic energy, (g) In-memory Boolean logic (N)AND/(N)OR energy, (h) In-memory addition/subtraction dynamic energy, (i) Breakdown of area, (j) Power consumption in log-scale, (k) In-memory MAC latency in log-scale, and (l) In-memory MAC energy in log-scale.

4.2 Latency

Fig. 9a-d reports the latency for various under-test PIM candidates in terms of read/write, in-memory Boolean logic, and add/sub, respectively. We observe that ReRAM-PIM (indicated by Re) gives the smallest latency for the read operation, whereas for the write operation, it requires ~20ns for programming to LRS and HRS. Therefore, this could not be a proper candidate for the digital write-back-intensive MAC-based applications such as DNNs. We can see that SOTMRAM-PIM (SOT) achieves the smallest write latency among NVMs (~2.6ns) due to the intrinsic SOT-based magnetization switching [15]. As for performing in-memory (N)AND/(N)OR (Fig. 9c) and addition/subtraction operations (Fig. 9d), Re, volatile SRAM (indicated by S) and SOT designs are respectively the fastest designs. Besides, we report the breakdown of PIM MAC latency for bit-wise multiplication and addition operation based on the mapping technique presented in Section 3. In summary, we observe that SOT (~14.3ns) and S (~19.4ns) offer the least MAC latency compared to other counterpart designs among NVMs and VMs, respectively.

4.3 Energy Consumption

We plot the dynamic energy consumption for various PIM candidates in Fig. 9e-h. Based on our experiment results, SOT along with S design could be again selected as the most promising energy-efficient candidates among NVMs and VMs. As for D1, the PIM requires a relatively small (0.75 nJ in Fig. 9g) to execute (N)AND/(N)OR, however, it imposes 14 memory cycles energy for addition/subtraction operation to avoid overwriting data, which leads to much higher energy consumption compared to other platforms (Fig. 9h). Fig. 9l depicts the breakdown of PIM MAC energy for bit-wise multiplication and addition operation. Based on the results, SOT and D2 achieve the least energy consumption compared with other designs among NVMs and VMs, respectively. In terms of leakage power consumption reported in Fig. 9j, the digital ReRAM and D1 could be considered as more power-efficient candidates. The SRAM platform consumes ~14.5× more power when compared to the Re platform.

4.4 Area

Fig. 9i plots the breakdown of the area for various PIM technology choices for memory die area and computational area. The computational area includes areas imposed by the modified memory controller, decoder, sense amplifiers, etc. In terms of memory die area, the STT and D1 platforms show the smallest footprint, while S owns the largest overall area compared to other PIM counterparts.

5 APPLICATION-LEVEL PERFORMANCE

To explore the trade-off between energy consumption and latency, we implemented various MAC-based applications with the well-known Slansky prefix structure using the selected technologies from the cross-technology comparison in Section 4. We choose five compact DNNs models for image classification, i.e., AlexNet, SqueezeNet v1.0, MobileNet, GoogLeNet, and Inception-V2, SPDNN for iris segmentation, WDRN-s and LapSRN for single-image super-resolution. We then developed a correlated data partitioning and mapping methodology for SOT, STT, S, and D2 platforms to make a fair comparison.

The energy consumption results of this comparative study are shown in Fig. 10a in log scale. Our first observation is that SOT implementation of PRIMS consumes the smallest energy consumption compared with others. This comes from the efficient MAC and write-back operations in this technology. To show the impact of micro-architectural design and mapping strategy in overall PIM performance, we developed a second more optimized mapping technique. More number of local row buffers are used to avoid data write-back. This new architecture requires a more specific mapping

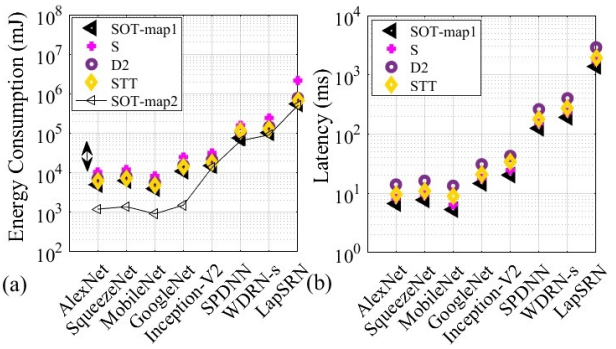

Figure 10: The comparative results of (a) Energy Consumption and (b) Latency for SOT-MRAM (SOT), STT-MRAM (STT), SRAM (S), and DRAM (D2) PIM platforms for various MAC-based applications.

method (map2) for SOT, as shown in Fig. 10a. The SOT-Slansky-map2 is able to reduce energy consumption on average by a factor of 2 compared with baseline SOT. Our second observation is as the workload size increases, the platforms with more energy-efficient write-back operations such as S and D2 will show better performance. Our third observation is that D2 consumes relatively smaller energy compared with S in different benchmarks. Therefore, it could be considered as the most energy-efficient volatile PIM.

Fig. 10b shows the log-scaled normalized latency of the same under-test platforms to execute the MAC-based applications. Based on our observation, SOT again stands as the fastest design, where the S design stands as the second-best. Here, D2 shows the worst performance mainly due to its intrinsic multi-cycle operations to avoid data overwritten issues [8].

6 CONCLUSION AND FUTURE WORK

In this work, we studied the opportunities and challenges of leveraging the residue number system (RNS) as an alternative for the conventional binary number representation to accelerate multiplication-and-accumulations (MACs) in the Processing-in-Memory (PIM) domain. We first developed a PIM-RNS architecture, entitled PRIMS, and analyzed PIM architecture's potential with the inherent parallelism of the RNS arithmetic. We then built a comprehensive cross-layer evaluation framework to quantitatively study this problem in eight MAC-based applications considering both PIM and RNS impacts. The obtained results show that with a proper choice of PIM technology, the RNS could be a promising approach to accelerate the PIM operations further and reduce data communication overhead.

There are numerous applications accelerated using RNS, which can be divided into two categories: (1) applications that persist against error, error-resilient, which means they function correctly in the presence of faults at the cost of accuracy or performance degradation. Their operations are comparable with the approximate computation in the weighted number system, where we decrease the precision to reduce the hardware cost with a slight accuracy loss. In this case, a smaller number of bits can be used to perform modular arithmetic operations, leading to a reduction in area, delay, complexity, and accuracy. (2) applications that cannot tolerate error during the operations, including cryptography and critical-mission tasks. In this case, the error occurrence possibility is omitted using a larger dynamic range and/or extending the dynamic range. The defined dynamic range must cover all possible intermediate results to avoid overflow (error). As future work, the proposed framework will be extended to explore a proper tradeoff between *accuracy* and the *performance metrics*, including complexity, power consumption, and operating frequency with respect to the dynamic range of a target moduli set.

7 ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation under Grant No. 2005209 and No. 2003749.

REFERENCES

[1] Y.-H. Chen *et al.*, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE journal of solid-state circuits*, vol. 52, no. 1, pp. 127–138, 2016.
[2] A. Roohi *et al.*, "Apgan: Approximate gan for robust low energy learning from imprecise components," *IEEE Transactions on Computers*, vol. 69, no. 3, pp. 349–360, 2019.
[3] H. Sharma *et al.*, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network," in *ISCA*. IEEE, 2018.
[4] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *ISSCC*, 2014, pp. 10–14.
[5] P. Chi *et al.*, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 27–39, 2016.
[6] V. Seshadri *et al.*, "Ambit: In-memory accelerator for bulk bitwise operations using commodity dram technology," in *MICRO*. IEEE, 2017, pp. 273–287.
[7] S. Li *et al.*, "Drisa: A dram-based reconfigurable in-situ accelerator," in *2017 MICRO*. IEEE, 2017, pp. 288–301.
[8] S. Angizi and D. Fan, "Redram: A reconfigurable processing-in-dram platform for accelerating bulk bit-wise operations," in *38th ICCAD*, 2019, p. 8942101.
[9] A. Roohi, S. Angizi, D. Fan, and R. F. DeMara, "Processing-in-memory acceleration of convolutional neural networks for energy-effciency, and power-intermittency resilience," in *20th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2019, pp. 8–13.
[10] C.-H. Chang *et al.*, "Residue number systems: A new paradigm to datapath optimization for low-power and high-performance digital signal processing applications," *IEEE circuits and systems magazine*, vol. 15, no. 4, pp. 26–44, 2015.
[11] M. Taheri *et al.*, "Efficient incorporation of the rns datapath in reverse converter," *IEEE TCAS II: Express Briefs*, 2020.
[12] M. Taheri, N. Shafiee, M. Esmaeildoust, Z. Amirjamshidi, R. Sabbaghi-nadooshan, and K. Navi, "A high speed residue-to-binary converter for balanced 4-moduli set," *Journal of Computing and Security*, vol. 2, no. 1, pp. 43–54, 2015.
[13] M. Taheri, K. Navi, and A. Sabbagh Molahosseini, "Efficient programmable power-of-two scaler for the three-moduli set {2n+ p, 2n- 1, 2n+ 1- 1}," *ETRI Journal*, vol. 42, no. 4, pp. 596–607, 2020.
[14] D. Reis *et al.*, "Modeling and benchmarking computing-in-memory for design space exploration," in *GLSVLSI*, 2020, pp. 39–44.
[15] S. Angizi *et al.*, "Accelerating deep neural networks in processing-in-memory platforms: Analog or digital approach?" in *ISVLSI*. IEEE, 2019, pp. 197–202.
[16] S. Angizi, J. Sun, W. Zhang, and D. Fan, "Graphs: A graph processing accelerator leveraging sot-mram," in *DATE*. IEEE, 2019, pp. 378–383.
[17] S. Jain *et al.*, "Computing in memory with spin-transfer torque magnetic ram," *IEEE TVLSI*, vol. 26, no. 3, pp. 470–483, 2017.
[18] S. Angizi *et al.*, "Rimpa: A new reconfigurable dual-mode in-memory processing architecture with spin hall effect-driven domain wall motion device," in *ISVLSI*. IEEE, 2017, pp. 45–50.
[19] C. Eckert *et al.*, "Neural cache: Bit-serial in-cache acceleration of deep neural networks," pp. 383–396, 2018.
[20] X. Fong *et al.*, "Spin-transfer torque devices for logic and memory: Prospects and perspectives," *IEEE TCAD*, vol. 35, no. 1, pp. 1–22, 2015.
[21] 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.
[22] X. Dong *et al.*, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 994–1007, 2012.
[23] . DRAM Power Model. https://www.rambus.com/energy/.
[24] (2011) Ncsu eda freepdk45. [Online]. Available: http://www.eda.ncsu.edu/wiki/FreePDK45:Contents