MAPIM: Mat Parallelism for High Performance Processing in Non-volatile MemoryArchitecture

Joonseop Sim*, Minsu Kim**, Yeseong Kim*, Saransh Gupta*, Behnam Khaleghi* and Tajana Rosing*
* *University of California San Diego* {j7sim, yek048, sgupta, bkhalegh, tajana}@ucsd.edu
** *University of Minnesota* {kimx4916}@umn.edu

***Abstract*—In the Internet of Things (IoT) era, data movement between processing units and memory is a critical factor in the overall system performance. Processing-in-Memory (PIM) is a promising solution to address this bandwidth bottleneck by performing a portion of computation inside the memory. Many prior studies have enabled various PIM operations on non-volatile memory (NVM) by modifying sense amplifiers (SA). They exploit a single sense amplifier to handle multiple bitlines with a multiplexer (MUX) since a single SA circuit takes much larger area than an NVM 1-bit cell. This limits potential parallelism that the PIM techniques can ideally achieve. In this paper, we propose MAPIM, mat parallelism for high-performance processing in non-volatile memory architecture. Our design carries out multiple bit-lines (BLs) requests under a MUX in parallel with two novel design components, multi-column/row latch (*MCRL*) and shared SA routing (*SSR*). The *MCRL* allows the address decoder to activate multiple addresses in both column and row directions by buffering the consecutively-requested addresses. The activated bits are simultaneously sensed by the multiple SAs across a MUX based on the *SSR* technique. The experimental results show that MAPIM is up to 339× faster and 221× more energy efficient than a GPGPU. As compared to the state-of-the-art PIM designs, our design is 16× faster and 1.8× more energy efficient with insignificant area overhead.**

I. Introduction

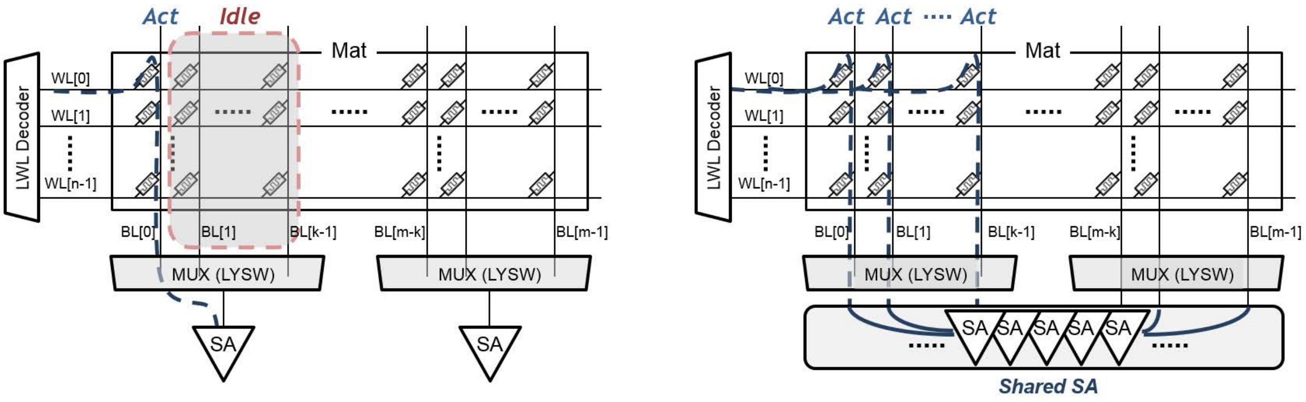
With the advent of Internet of Things (IoT), billions of devices and objects are continuedly gathering and transmitting data [1]. Data movement between processing units and memory needs to be increased to enable massive data analysis needs. However, the conventional memory systems based on off-chip channel networks, e.g. DDR3/4 and GDDR, cannot support such a large amount of data movements due to pin count limitation [2], [3]. *Processing-in-Memory* (PIM) [4]–[9] places processing units close to the memory. This enables computation with high-bandwidth accesses to where the data resides.

Emerging NVM is a promising candidate to enable the PIM techniques due to its high density and low static power [10]. Recent work modifies sense amplifiers (SA) to enable computing [4], [7]–[9], [11], [12]. Since the NVM uses the current-mode SA, the SA size of the NVM is much larger than that of the DRAM which senses a voltage difference between the bitlines [4], [13]. As a result, an SA of the NVM takes charge of multiple BLs, and multiple bits under a single SA cannot operate in parallel. In the conventional memory, this restriction has not been a critical since the off-chip bandwidth is limited. The work in [14] showed that, since the commodity memory hierarchies were designed targeting the processor sector size, increasing atomic size leads to performance degradation [15]. In a DDR3 channel, for example, the 64-bit I/O interface operates at 800MHz, providing 12.8GB/s of memory bandwidth. To support this channel bandwidth, with a memory subsystem that has 8 chips-per-rank, 8 banks-per-chip, 32 subarrays-per-bank and 32 mats-per-subarray, a bank in the memory, which has a 200MHz internal frequency with an 8-bit prefetch, outputs 8 bits at each channel clock. This estimate implies that a mat atom size is less than 1-bit per a clock frequency, meaning a mat-level parallelism does not effect on system performance due to narrow off-chip bandwidth. In contrast, PIM design needs to execute multiple BLs *in parallel* as much as possible since the channel bandwidth is not the bottleneck. However, the existing NVM-based PIM techniques have to execute each bit connected to a single MUX *sequentially* in the mat array due to limitations to the size of SA, resulting in much lower performance.

In this paper, we propose a high-performance PIM architecture which enables the simultaneous execution of the PIM operations for multiple BLs. Baseline NVM architecture is based on the work in [13], [16] whose $m \times n$ mat structure is shown in Fig. 1(a). A mat is an atomic access unit for a single memory operation [17]. It has its private SAs and decoders. The WLs and BLs are paired with the local wordline (LWL) decoder and SA to control which bits are selected from the array. Compared to DRAM, the most commercialized memory product, a big difference in NVM design is an SA structure. While in DRAM SA, each BL is connected to an individual SA circuit, NVM array structure places a multiplexer (MUX) in front of the SA, which are used to select a BL connecting to a single I/O line from the multiple BLs, denoted as k lines in Fig. 1(a). Therefore, the local BLs tied in a MUX (*i.e.* 8BLs or 16BLs) cannot be accessed in parallel but in bit by bit mode.

Our design, called MAPIM (Mat Parallelism for High-Performance Processing in Non-volatile Memory Architecture), is built with two novel components: multi-column/row latch (*MCRL*) and shared SA routing (*SSR*). Fig. 1(b) shows the sensing scheme of the proposed MAPIM. In baseline NVM structure, the number of SAs in a mat ranges from 32 to 64, which is similar to the number of bits in a word. MAPIM shares the SAs in a mat and thus enables word-size of multiple bits to be sensed in parallel. The *MCRL* activates multiple WLs/BLs at the same time, so that the PIM operations are requested in a row/column-parallel way. The *SSR* allows the requested multi-BLs to use multiple SAs across a MUX, thus fully utilizing the SAs of the mat for the parallel execution in the PIM operations.

Our contribution can be summarized as follows:



(a) Conventional NVM sensing scheme

(b) Proposed ***MAPIM*** sensing scheme

Fig. 1. Conventional NVM and proposed MAPIM sensing structures

- We show that the SA size is a significant constraint to achieve further performance of PIM operations.
- Our MAPIM design parallelizes multiple-BLs execution within a mat. We also show the detailed circuit-level design with the robustness and sensitivity analysis.
- The proposed *MCRL* design enables efficient multi-row/column activations. Unlike the state-of-the-art design, it holds the activation signals with a wide margin of μs order, allowing to select any target operands for the parallel PIM computations.
- With the *SSR* technology, the proposed MAPIM fully utilizes the limited number of SAs located in a mat, when the PIM operations are performed for a large number of rows and columns.

Our evaluation shows that the proposed MAPIM achieves up to 339× speedup and 221× energy saving compared to a GPGPU architecture, and 16× speedup and 1.8× energy saving compared to the state-of-the-art PIM designs.

II. Related Work

Recent spike in research on PIM architecture is due to the development of 3D-stacking technologies [2]. The works in [3], [18] integrate extra computing units using 3D technologies close to the memory, facilitating the computation near the data. However, this approach is still inefficient due to implementing a costly logic stack into the memory stack, including *through-silicon-via* (TSV), micro-bumps, etc. An alternative way to address the issue of 3-D technology is to place the computation units directly inside the memory. However, in commercialized memory, *e.g., SRAM, DRAM,* to minimize the overhead caused by the computation units is a major concern since the PIM function is placed in the matured memory hierarchy potentially causing larger overhead. NVM is a promising candidate for the PIM design due to its analog characteristic based on the current-drive mode and less overhead. [4], [11]. Several works modified the SA to enable the PIM function on the existing architecture [4], [6]–[9], [11], [12]. However, the current-mode SA of NVM, which takes a larger area compared to the voltage-mode SA of DRAM, hinders the local parallelism since the multiple BLs assigned to a multiplexer should operate in sequence. Our work addresses this lack of parallelism by proposing a novel design utilizing address latching and SA sharing techniques.

Parallelism at rank [19], bank [20], subarray [21] has been actively studied, but mat-level parallelism has not been deeply explored since it is not very effective in conventional off-chip memory due to the boundness of channel bandwidth. However, PIM design can fully utilize the mat parallelism to the performance improvement. In this paper, we enable a mat-level parallelism with insignificant modification to state-of-the-art NVM architecture.

III. Proposed MAPIM design

A. Overview

Fig. 2 shows the mat structure of MAPIM. Mat is the building block of *bank*, a fully-functional memory unit [22]. It consists of cell arrays, SA, and local decoder. The figure does not show upper bank level design which remains the same as before [13]. We introduce two new structures to enable mat-level parallelism: multi-column/row latch (*MCRL*) and shared SA routing (*SSR*). A local bitline (LBL) decoder is placed under a local word-line (LBL) decoder to ease layout. The decoded signals from the LWL decoder and LBL decoder are transferred to the WLs and local Y-switch (LYSW), respectively, to activate the corresponding WLs and BLs as shown in Fig. 2. *MCRL*, denoted as (A) in Fig. 2, refers to both multi-column latch and multi-row latch. It serves as the buffer that keeps the sequentially activated WLs and signals to the LYSWs from the decoder and enables them simultaneously. *SSR*, marked as (C), enables multi-bit execution in parallel by sharing the limited number of SAs in a mat to the requested BLs. NVM cell is sensed by the resistance difference between a logical 0 and 1, so it is less sensitive to the BL capacitance as compared to the conventional DRAM. This allows NVM SAs to be delocalized within a mat and be flexible to associate with segmented BLs.

B. Multi-Column/Row Latch (MCRL)

We next discuss each of our design's components in detail. Fig. 2(A) shows a circuit unit of the proposed *MCRL*

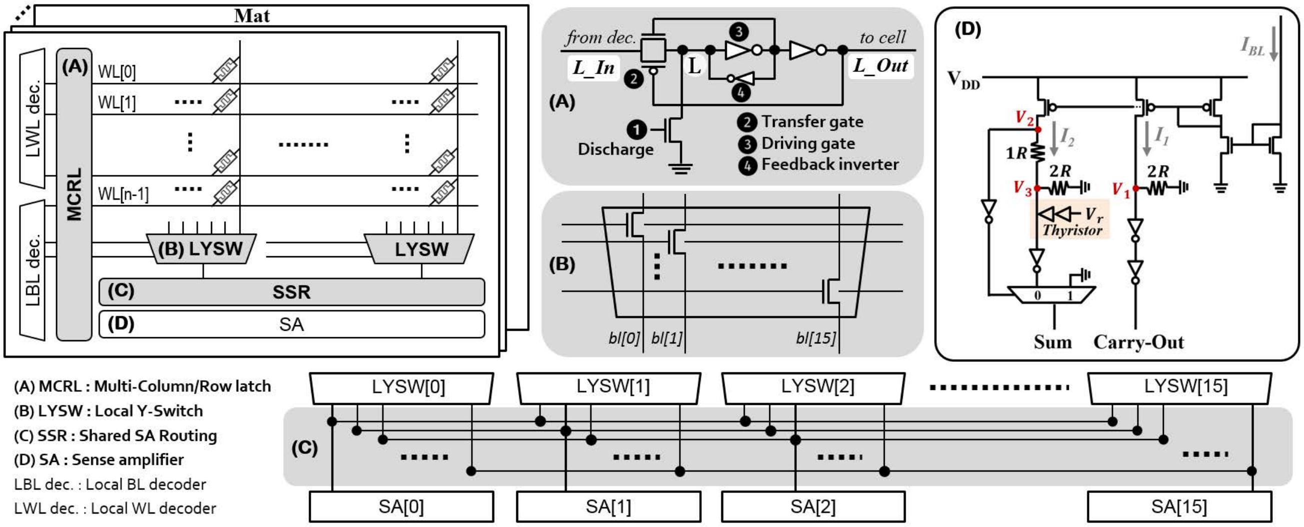


Fig. 2. Mat structure of proposed MAPIM design

corresponding to a single WL as a representative. It consists of four components: ❶ a discharge transistor, ❷ a transfer gate, ❸ a driving gate, and ❹ a feedback inverter. First, the discharge transistor resets all WLs to ground. In the second step, the transfer gate passes the decoded signals from the local decoders to the latch. The transfer gate is made up of two transistors, an n-channel MOSFET (*NMOS*) and a p-channel MOSFET (*PMOS*), connected in parallel, since they need to transfer both 0 and 1 without loss of strength. Following two inverters, a driving gate and a feedback inverter, drive the decoded signal to '*L_Out*' in a cell array. When the input signal from the decoder is low, a node '*L*' potential does not change since the initial status of the node is low. When the decoded signal is high, *i.e.*, a corresponding WL is *activated*, the node '*L*' potential flips to one and the inverted potential turns the transfer gate off, so the input signal is latched in the *MCRL*. The feedback inverter is used to prevent the node '*L*' from floating when the transfer gate is off. It needs to be weaker compared to the transfer gate for faster switching. In our experiments, the drivability of the feedback gate is set to 1/3 of transfer gate for easy signal transference. The circuit evaluation of our *MCRL* design is shown in Sec.IV-B.

C. Shared SA Routing (SSR)

As mentioned in Sec.III-A, NVM is less sensitive to BL length compared to DRAM. NVM has resistance-based operation, so its SA works by detecting current change across the NVM cells. The work in [22] showed that the current-mode SA has a sub-linear dependency on BL length, so cells can be sensed out of the cell-array. Moreover, NVM read is not destructive while DRAM's is. Thus resistances such as MUX can be placed in front of SA. This gives us more room to increase parallelism with our *SSR* design. Fig. 2(C) shows the proposed *SSR* design. Compared to the conventional NVM SA design in which each SA corresponds to their own LYSW as shown in Fig. 1(a), the LYSWs of our design share the SAs within a mat as shown in Fig. 2(C). 16 BLs in an LYSW, described in Fig. 2(B), are allocated in SA[0]∼SA[15]. All other LYSWs in a mat, *i.e.*, LYSW[0]∼LYSW[15], share the SAs in the same sequence. Consequently, 16BL-sets of all LYSWs in a mat have the same SA allocation as shown in Fig. 2(C). In this paper, we have 16 BLs in an LYSW and 16 LYSW in a mat for illustration purposes. However, the actual array size can be flexible with a typical range of 8 BLs and 8 LYSWs to 32 BLs and 32 LYSWs as a function of the memory granularity. The LBL decoder has two steps. Predecoding selects an LYSW in a mat. A subsequent decoding step selects a BL in that LYSW. At a given LYSW selected in the pre-decoding, proposed *MCRL* holds the consecutively activated BL addresses. Then the multiple signals buffered in the *MCRL* enable multiple-columns to activate by turning on NMOS transistors in an LYSW as shown in Fig.2(B). The selected BLs are read out simultaneously. The accessed data is assigned to the corresponding SAs by our shared-routing method.

The latency of *SSR* is shown in Fig. 3, where *tRCD* is the row activation time, *tRP* is the precharge time, and *tCL* is the column read time. A read cycle (tRC) is represented as *tRC* = *tRCD* + *tCL* + *tRP*. Data restoration time is zero since the NVM is inherently non-destructive in sensing operation. [23]. We define a new timing parameter, *tCIT*, the time interval between sending two consecutive column addresses. The top of Fig. 3 presents the timing graph of the baseline in which the local BLs are read in sequential mode. The row address activates a WL and the first column address (*Col* 1) activates a BL. Then the selected bit is sent to an assigned SA. As our sensing circuit can perform a 1-bit addition, which will be explained in Sec. III-D, a *carry-out* of 1-bit addition is written back to the memory cell in the next bit and a *sum* is sent to the cache. Once the *Col* 1 is activated, 1-bit operation and precharge have completed, during *tCL* and *tRP*, then the *Col* 2 is activated for the next bit operation.

MAPIM temporarily holds on multiple columns requests, *e.g.* *Col* 1 and *Col* 2 as shown on the bottom of Fig. 3, in the

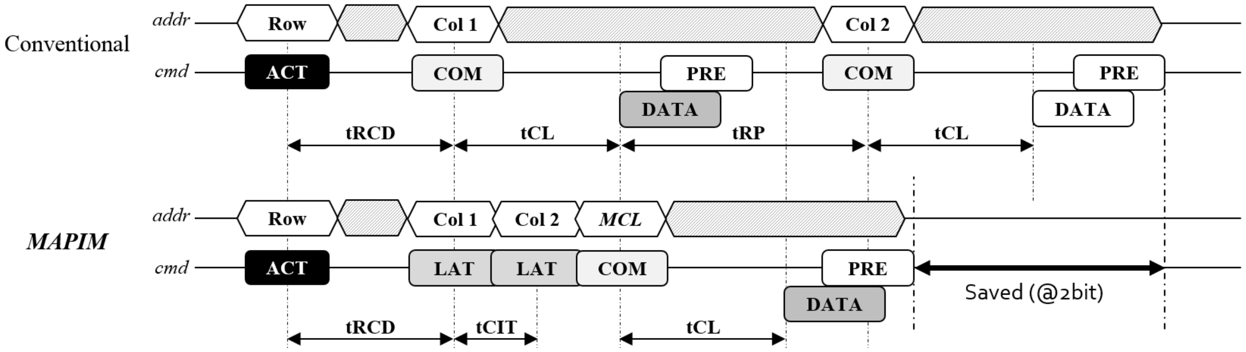


Fig. 3. Timing graph of two requests from two different BLs

latch and they are activated simultaneously with the *MCRL* activation request. While the conventional operation uses two *tCL* and one *tRP* for the two-bit calculation as shown in Fig. 3, MAPIM uses one *tCL* for the same operation, which can accelerate the requested operation by saving *tCL*. Note that *tCIT* is negligible compared to *tCL* and *tRP* since multiple column latches occur inside the decoder logic whereas *tCL* and *tRP* are between the on-chip memory controller and the off-chip memory. In N-bit operation, conventional design takes N×*tCL* + (N-1)×*tRP*, while MAPIM takes *tCL* + (N-1)×*tCIT*. Since MAPIM consumes fixed amount of time, *tCL*, as the number of bits increases, overall speedup of MAPIM increases due to saving column read and precharge latency.

D. Sensing Circuits for Arithmetic Operation

Fig. 2(D) is the sensing circuit for the arithmetic operation in MAPIM. Our design modifies a conventional SA [24], which determines the logical 0 and 1 by reading the cell resistance, to perform 1-bit addition. We exploit 1-bit addition using our sensing circuit since the addition is a building block of the order of operations for our application test. The current mirror in Fig. 2(D) copies the I_{BL} to I_1 and I_2 and delivers them to *Carry-out* (C_{out}) and *Sum* nodes, respectively. I_{BL} is the read current in a BL, accumulated from the selected three bits when three rows are activated in a cell array. Fig. 4 shows how the circuit carries out 1-bit addition. Since each memristor has two states, 0 and 1, there are four equivalent combinations of I_{BL}: I_{000}, I_{100}, I_{110} and I_{111} in the case of three-bit calculation. There are three voltage nodes: V_1, V_2, and V_3, as shown in Fig. 2(D), whose potential determines the final outputs of the *Sum* and C_{out}. The voltage of each node is determined as a function of the current, C_{out} and *Sum* nodes exhibit the corresponding values in each case of I_{BL} and finally result in the execution of 1-bit addition. Our sensing circuit utilizes the thyristor latch-up effect [25] to execute the I_{111} region to create the desired C_{out}=1 and *Sum*=1.

IV. Experimental Results

A. Experimental Setup

We evaluated the performance and energy consumption of the proposed design as compared to the work in [4], [24]. We used *VTEAM* memristor model [26] for our cell design with R_{on} and R_{off} of 10KΩ and 10MΩ respectively. Circuit-level simulations are performed with Cadence Virtuoso and Spectre based on a 45*nm* CMOS process. We compared the performance and energy of MAPIM with AMD Radeon R9 390 GPU with 8GB memory. We modified Multi2sim [27], a cycle-accurate CPU-GPU simulator, to evaluate the impact of different PIM designs when the parallelized instructions are performed with the PIM operations. We experimented with four applications, *Sobel*, *Robert*, *Fast Fourier transform (FFT)* and *DwHaar1D*.

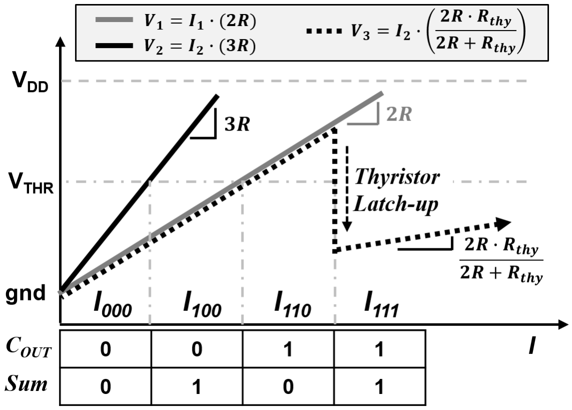


Fig. 4. Sensing circuit for 1-bit addition

B. MCRL Robustness

Fig. 5 presents the result of circuit evaluation of *MCRL* in MAPIM and comparison to the state-of-the-art design in [4]. For the latch design in [4], we observe that i) low signal drivability leads to large area overhead to boost the signal across gates, and ii) the node, which holds the signal (denoted as 'L' at Fig. 2(A)), is likely to be floating. These issues may make the latch unable to hold the signal long enough for the multiple activations due to the leakage-current dissipation. As shown in Fig. 5, the signal of the 'L' node in *Pinatubo* [4] is so weak as that only 48% of the total signal strength at *L_In* node is transferred. Moreover, the holding latency is too short as to keep the signal less than a few *ns*. Considering our experimental setup, *tRC*=*tRP*+*tCL*+*tRP*=28*ns*, the signal

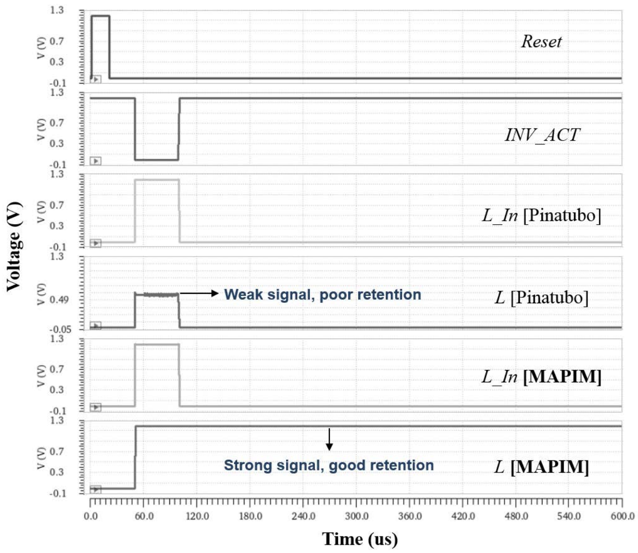


Fig. 5. Circuit evaluation of the multi-activation latch between Pinatubo [4] and MAPIM

holding time in [4] is too low to keep the signal during only one cycle.

In contrast, MAPIM proposed a robust circuit design for the multiple-addressed activation. This benefit derives from the following novelties in our design: 1) the transfer gate can deliver both high(1) and low(0) signals from the decoder without the loss of strength. 2) the feedback inverter prevents the node inside the latch from floating, which suppresses the static power consumption. As seen in Fig. 5, *MCRL* holds the activated signal for over 600*ns*. Although we show the data for the range of 0∼600*ns* for easy illustration, our experimental results show that *MCRL* retains the activated signal over a few μs without losing signal strength. This allows MAPIM to keep the signal long enough for the three-row activations which is common in PIM operation. Furthermore, the signal at the 'L' node is as strong as that at *L_In* node in *MCRL* design, thus ensuring robustness.

C. Performance Sensitivity to the Number of Bits

We compare the mat-parallelism in our design with two state-of-the-art PIM designs, *Pinatubo* [4] and *LUPIS* [24], which execute sequential operations in a MUX. Fig. 6 presents the latency improvement of MAPIM for *addition* and *multiplication*. The data shown in Fig. 6 is normalized to the latency of *Pinatubo*, set to 1. The latency of addition is evaluated from the design in Fig. 2 while considering the timing parameters shown in Fig. 3. The latency of multiplication is estimated by Eq. in [28],

$$t_{MUL} \propto [(M-1)+(N-2)]\cdot t_{carry}+(N-1)\cdot t_{sum}+(N-1)\cdot t_{and}$$

where M and N are the multiplicand and multiplier, and t_{carry}, t_{sum}, and t_{and} are the latencies for evaluating *carry*, *sum*, and *AND*, respectively. Although the latency improvement of MAPIM is around 4 times for 2-bit operations, the improvement grows as the size of operands increases, with 36× improvement for 32-bit operations. Both [4] and [24] use sequential mode in multi-bit operation, thus having much higher latency. On the contrary, MAPIM buffered all required addresses and activated them in a single request, so it takes one *tCL* regardless of the number of bits, hence much better performance as the number of bit increases.

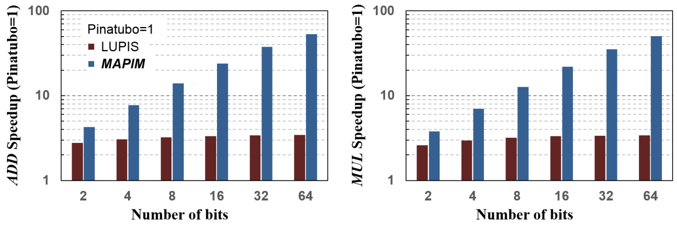


Fig. 6. Latency improvement of MAPIM for arithmetic operations

D. Parallelism Efficiency in Applications

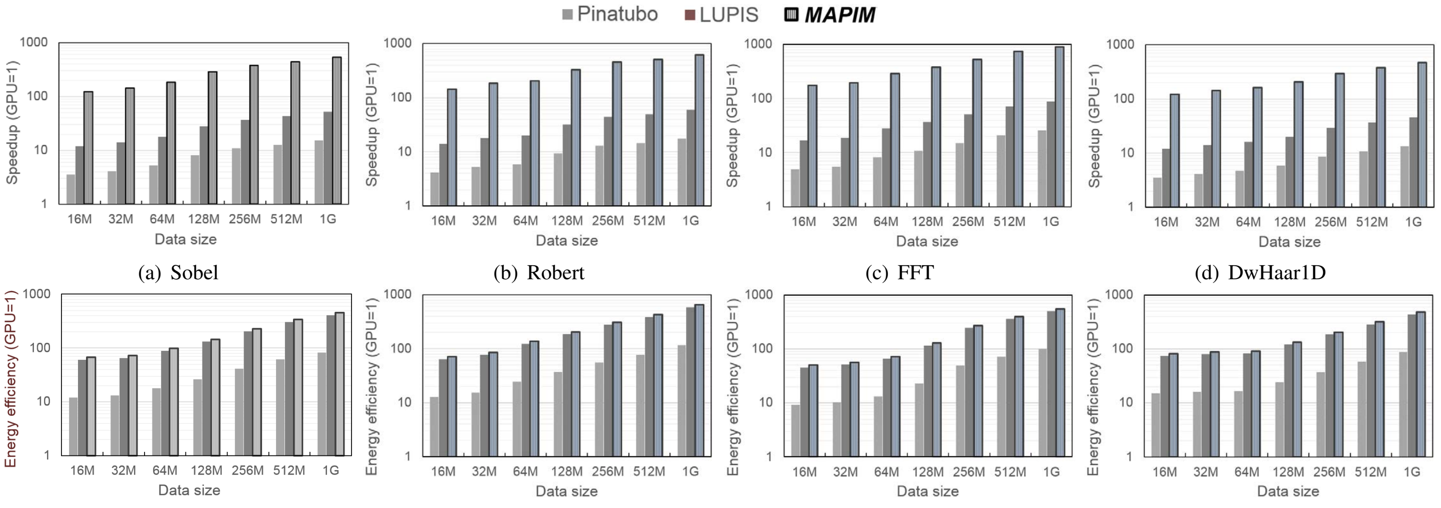
We evaluate the efficiency of MAPIM with four OpenCL applications. Fig. 7 shows the speedup and the energy efficiency of proposed MAPIM and two state-of-the-art PIM designs, *Pinatubo* [4] and *LUPIS* [24]. All results are normalized to the value of unmodified AMD GPU. The results show that the three PIM designs outperform the GPU-based computation for the wide range of the memory size. It is because PIM designs reduce the data movement costs. Among the PIM designs, MAPIM achieves the best performance improvement as compared to the other PIM techniques [4], [24]. For example, the proposed design is 339× faster and 221× more energy efficient as compared to the GPU, and 16× faster as compared to the state-of-the-art PIM designs, *Pinatubo* [4] and *LUPIS* [24]. This suggests that utilizing mat-level parallelism is key to designing highly efficient PIM architectures.

E. Overhead

MAPIM adds a latch circuit and a routing block to each mat array. The area overhead has been estimated by the ratio of the additional latch and sharing routing area over the corresponding cell area. We assume 32 BLs-per-LYSW and 32 LYSWs-per-mat. The area estimates from Cadence p-cell data with 45*nm* process technology, the overhead incurred in MAPIM occupies an area of 1.5um^2, which is a 0.57% area overhead compared to a corresponding row of sub-array. This is a very similar value to the work in [4] which has 0.49% area overhead but works for better performance as shown in Sec. IV-B. MAPIM also shows outstanding area efficiency compared to the work in [21], a state-of-the-art latch design for DRAM, showing an area of 42.9um^2.

V. Conclusion

We present a high-performance PIM architecture which enables mat-level parallel operations by enabling higher parallelism in its mat structures. The proposed design hugely accelerates the PIM applications whose performance is not restricted by the off-chip channel bandwidth. The experimental results show that our design presents 16× and 339× performance improvement compared to the state-of-the-art PIM designs and the GPU architecture, respectively.



(a) Sobel (b) Robert (c) FFT (d) DwHaar1D

(e) Sobel (f) Robert (g) FFT (h) DwHaar1D

Fig. 7. Speedup and energy efficiency of proposed MAPIM to other PIM architectures [4], [24] for different applications.

Acknowledgements

This work was partially supported by CRISP, one of six centers in JUMP, an SRC program sponsored by DARPA, and also NSF grants #1730158 and #1527034. Joonseop Sim is partially supported by a Ph.D. fellowship from SK Hynix Inc.

References

[1] D. E. O'Leary, "Artificial intelligence and big data," *IEEE Intelligent Systems*, 2013.

[2] J. Ahn *et al.*, "Pim-enabled instructions: a low-overhead, locality-aware processing-in-memory architecture," in *Computer Architecture (ISCA), 2015*.

[3] J. Ahn *et al.*, "Aim: Energy-efficient aggregation inside the memory hierarchy," *ACM Transactions on Architecture and Code Optimization (TACO)*, 2016.

[4] S. Li *et al.*, "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," in *Design Automation Conference (DAC), 2016*.

[5] S. Kvatinsky *et al.*, "Magicmemristor-aided logic," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2014.

[6] E. Lehtonen and M. Laiho, "Stateful implication logic with memristors," in *Proceedings of the 2009 IEEE/ACM International Symposium on Nanoscale Architectures*, pp. 33–36, IEEE Computer Society, 2009.

[7] V. Seshadri *et al.*, "Fast bulk bitwise and and or in dram," *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 127–131, 2015.

[8] V. Seshadri *et al.*, "Ambit: In-memory accelerator for bulk bitwise operations using commodity dram technology," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 273–287, ACM, 2017.

[9] S. Li *et al.*, "Drisa: A dram-based reconfigurable in-situ accelerator," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 288–301, ACM, 2017.

[10] C. S. Hwang, "Prospective of semiconductor memory devices: from memory system to materials," *Advanced Electronic Materials*, vol. 1, no. 6, 2015.

[11] M. Imani *et al.*, "Mpim: Multi-purpose in-memory processing using configurable resistive memory," in *Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific*, pp. 757–763, IEEE, 2017.

[12] P. Chi *et al.*, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *ACM SIGARCH Computer Architecture News*, IEEE Press, 2016.

[13] M. Poremba *et al.*, "Fine-granularity tile-level parallelism in non-volatile memory architecture with two-dimensional bank subdivision," in *Proceedings of the 53rd Annual Design Automation Conference*, p. 168, ACM, 2016.

[14] M. Rhu *et al.*, "A locality-aware memory hierarchy for energy-efficient gpu architectures," in *Microarchitecture (MICRO), 2013 46th Annual IEEE/ACM International Symposium on*, pp. 86–98, IEEE, 2013.

[15] M. O'Connor *et al.*, "Fine-grained dram: energy-efficient dram for extreme bandwidth systems," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 41–54, ACM, 2017.

[16] Y. Choi *et al.*, "A 20nm 1.8 v 8gb pram with 40mb/s program bandwidth," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, pp. 46–48, IEEE, 2012.

[17] T. Zhang *et al.*, "Half-dram: a high-bandwidth and low-power dram architecture from the rethinking of fine-grained activation," in *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*, pp. 349–360, IEEE, 2014.

[18] C. Zhang *et al.*, "Pm3: Power modeling and power management for processing-in-memory," in *High Performance Computer Architecture (HPCA), 2018 IEEE International Symposium on*, pp. 558–570, IEEE, 2018.

[19] W. Shin *et al.*, "Rank-level parallelism in dram," *IEEE Transactions on Computers*, vol. 66, no. 7, pp. 1274–1280, 2017.

[20] C. J. Lee *et al.*, "Improving memory bank-level parallelism in the presence of prefetching," in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*.

[21] Y. Kim *et al.*, "A case for exploiting subarray-level parallelism (salp) in dram," *ACM SIGARCH Computer Architecture News*, vol. 40, no. 3, pp. 368–379, 2012.

[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*, 2012.

[23] M. Poremba *et al.*, "Nvmain: An architectural-level main memory simulator for emerging non-volatile memories," in *VLSI (ISVLSI), IEEE Computer Society Annual Symposium on*, 2012.

[24] J. Sim *et al.*, "Lupis: Latch-up based ultra efficient processing in-memory system," in *Quality Electronic Design (ISQED), 2018 19th International Symposium on*, pp. 55–60, IEEE, 2018.

[25] X. Tong *et al.*, "Two-terminal vertical memory cell for cross-point static random access memory applications," *Journal of Vacuum Science & Technology B, Nanotechnology and Microelectronics: Materials, Processing, Measurement, and Phenomena*, 2014.

[26] 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*, vol. 62, no. 8, pp. 786–790, 2015.

[27] R. Ubal *et al.*, "Multi2sim: a simulation framework for cpu-gpu computing," in *Parallel Architectures and Compilation Techniques (PACT), 21st International Conference on*, IEEE, 2012.

[28] N. H. Weste *et al.*, *Cmos vlsi design*. Pearson India, 2010.