LUPIS: Latch-Up Based Ultra Efficient Processing In-Memory System

Joonseop Sim, Mohsen Imani, Woojin Choi, Yeseong Kim and Tajana Rosing
UC San Diego, La Jolla, CA 92093, USA
{j7sim, moimani, woc015, yek048, tajana}@ucsd.edu

Abstract*—Internet of Things (IoT) involves processing massive data. This poses a huge challenge in the current computing systems due to the limited memory bandwidth. Processing in-memory (PIM) is a promising candidate to minimize this bottleneck and reduce the performance gap between processor and memory latency. We propose *LUPIS* (Latch-Up based Processing In-memory System) for nonvolatile memory (NVM). Unlike existing PIM techniques, which mainly focus on bitwise operation based computations and involve considerable latency and area penalty, our design facilitates computations like addition and multiplication with very low latency. This makes the system faster and more efficient as compared to the state-of-the-art technologies. We evaluate LUPIS at both circuit-level and application-level. Our evaluations show that LUPIS can enhance the performance and energy efficiency by 62× and 484× respectively as compared to a recent GPGPU architecture. Compared to the state-of-the-art PIM accelerator, our design presents 12.7× and 20.9× improvement in latency and energy consumption with insignificant overhead of 21% for area increase and one cycle for latency delay.

I. Introduction

The era of the Internet of Things (IoT) has been driven by evolution of several technologies, including wireless communication, machine learning, and embedded systems [1], [2]. To assimilate the information transferred between connected devices, a large amount of data needs to be stored and processed. The movement of this data between the processing units and memory is one of the major power consumption and performance bottlenecks.

Several approaches have been proposed to address the issue of data movement. Near data computing (NDP) brings the computing unit close to the memory to avoid data transfer across the hierarchy. However, these designs need extra processing units near main memory. Some implementations put processing cores in different layers of 3D stacked memories to reduce the data transfer overhead [3], [4]. They however increase the energy consumption of the system and the data still needs to be transferred to the additional processing units. Processing in memory (PIM) is another promising way to address the data movement issue by processing data inside the memory, thus improving both performance and energy efficiency [5]–[19]. Although many PIM techniques have been proposed so far, they support limited basic functionalities such as basic bitwise operations (AND, OR, and IMP), which are only applicable to specific applications. For example, the designs shown in [20], [21] support bitwise operations but can not support arithmetic functions like the addition and multiplication. The techniques in [22] have been designed exclusively for accelerating neural network algorithms. Many applications including machine learning algorithms and image processing involve complex functions [13], [23], [24]. Hence, several techniques have been proposed to perform functions like addition and multiplication in NVM architectures [12], [25]–[27]. However, they execute these functions by combining multiple boolean operations (IMP, NOT, NOR). Therefore, they are inherently slow due to their multi-cycle operation as well as slow processing speed.

In this paper, we propose a new PIM architecture, called *LUPIS*, which enables the addition and multiplication in an efficient manner. We design a new sensing circuit which uses the analog properties of NVM. We simplify computation by exploiting the latch-up effect of thyristor devices to directly generate the results from the input data without any intermediate logic. We further leverage the back down effect at latch up points of thyristor to implement functions with minimal increase in the number of gates. In addition, the proposed design performs the operations in a modified sensing circuit which is compatible with the conventional current sense amplifier (CSA). It does not need additional cells to support calculations, thus requires negligible area overhead. We show that the proposed LUPIS can improve performance and energy efficiency of many popular applications such as machine learning and data analysis, which involve a large number of additions and multiplications.

The main contributions of this paper are listed as follows:

- We propose a high performance and low cost PIM architecture based on the latch-up effect of thyristors, enabling single-cycle addition (ADD), and significantly improving the performance of multiplication (MUL).
- Our design requires no additional cell array for processing, hence can be an excellent candidate for the storage class memory which has been considered as the main application of memristor-based products.
- Our experimental results show that LUPIS can provide 12.7× speedup and 20.9× energy efficiency as compared to the state-of-the-art PIM accelerator, APIM [12].

II. Related Work

Several work has proposed ways to address the issue of data movement between the processor and memory, by supporting basic functionality inside the memory module. Near-data computing adds extra computing units close to the memory in order to locally process the data [28]. However, this approach demands additional logic layers connected to multi-layer memory stack with through-silicon-via (TSV) [3], [4], hence it requires additional energy consumption and increases the fabrication costs. Processing in-memory is another solution to address data movement. PIM modifies the existing memory sense amplifiers to support basic operations and is a preferable approach due to its lower design complexity and cost efficiency.

The emerging nonvolatile memory (NVM) technologies such as phase change memory (PCM) and resistive RAM (ReRAM) are considered as good candidates for PIM due to their high density, scalability, and low power consumption [29], [30]. However, the supported functionality in most of the PIM designs is limited to either bitwise operations or operations derived from basic bitwise operations which require multiple cycles. For example, [20], [21] proposed a sensing circuit to implement the basic bitwise operations such as AND, OR, and INV. However, they do not support addition and multiplication which are the key arithmetic functions involved in many applications such as machine learning algorithms and image processing [27]. Several designs presented in [12], [25], [26], [31] have designed the full adder function based on bitwise operations. Since these approaches implement the operation by combining multiple basic operations, e.g., NOR or IMP, they require tens of cycles. They include computing the intermediate outcomes until obtaining the final results. Thus, these designs pose inevitable timing overheads. In addition, the huge area overhead due to the extra processing cell arrays make them unsuitable for storage class memory which demands high density integration.

III. Proposed Design

A. NVM Sensing Scheme

Emerging nonvolatile memories such as ReRAM, STT-RAM and PCRAM can be classified as the resistance-based memories. These technologies store and read the data by changing the cell resistance, e.g. its high and low resistance state are interpreted as logic 0 and 1 respectively. One of the major differences between NVM and DRAM is the sense amplifier design. While charge-based DRAM uses a voltage sense amplifier (VSA) which detects the electronic potential between the bitline (BL) and bitbar-line ($\overline{BL}$), NVM uses a different current sense amplifier (CSA) due to its better distinguishability of the resistance difference than the VSA. Fig. 1a shows the sensing scheme of the conventional CSA [32]. The data in a memory cell is determined by assessing the current from the selected memory cell. When the current from the selected BL (I_{BL}) and the current from the reference cell (I_{REF}) are mirrored to (I_1) and (I_2) respectively, they are compared to each other and changed to voltage signals (D_{OUT}) [33]. The state of $R_{CELL} < R_{REF}$ is considered as logic "1" and the other case is considered as logic "0" as shown in Fig. 1b. The conventional CSA is capable of only judging the resistance from the selected cell higher and lower than reference resistance. In this paper, we propose a current sensing scheme, which also enables arithmetic functions, i.e. addition and multiplication inside memory module, compatible with the conventional sensing scheme.

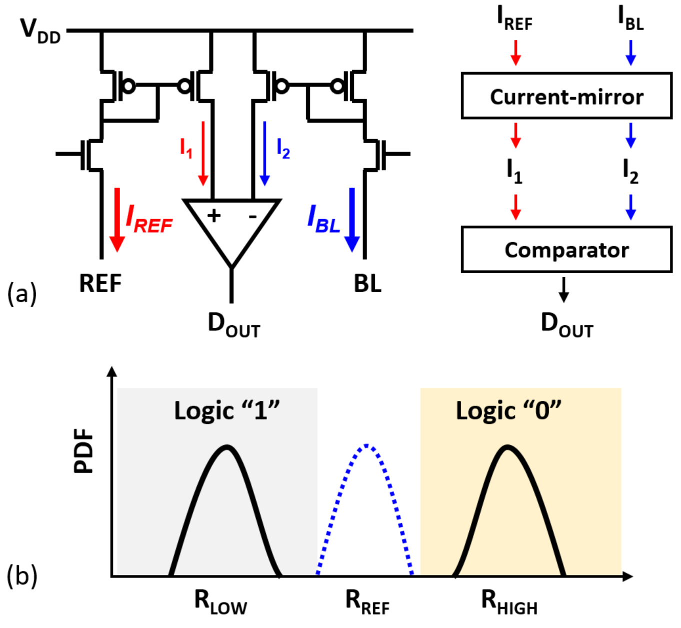


Fig. 1. Conventional Sensing Scheme for NVM

B. Thyristor Latch-Up

We exploit a vertical PNPN structure commonly referred to a *thyristor* [34]. Fig. 2a shows the structure used in our design. The structure has three P-N junctions and is equivalent to two cross-coupled bipolar junction transistors (BJTs) as shown in Fig. 2a. This structure has a short-circuit path, often referred as *latch-up* in CMOS design. When one of the two BJTs gets forward biased, it feeds the base of the other BJT. This positive feedback increases the current until it saturates to I_{short}.

Fig. 2b shows the voltage and the current behaviors of the structure. In the initial state, a thyristor has a high resistance (2MΩ in our experimental setup). When the voltage across the device (V_{AB}) is increased, the device keeps the high resistance state until V_{AB} reaches the latch-up voltage (V_{LU}). Latch-up occurs at V_{LU} and the current through the cell (i.e., from A to B in Fig. 2a) abruptly increases until the applied bias turns back to the latch-down voltage (V_{LD}). In order to restore the thyristor device resistance to the original state, a reverse bias, V_{RC}, should be applied to V_{AB}. It moves the minor carriers out from the base regions, and the device is set to the initial state again. In the rest of the paper, we call this recovery state as the *write back* step.

C. Sensing Circuit Design for Addition

We propose a new sensing circuit, which exploits the thyristor latch-up effect, to enable ADD operation in a cycle. Fig. 3 is the schematic of the proposed design. The design consists of two parts: the current mirror and adder. Once three rows of a memory block corresponding to the input values (A, B, C_{in}) are activated, the total current from the activated rows denoted as I_{BL} is delivered to the selected BL. The current mirror circuit in Fig. 3 copies the I_{BL} to I_1 and I_2 and delivers them to *Carry Out* (C_{out}) and *Sum* branches, respectively. Our

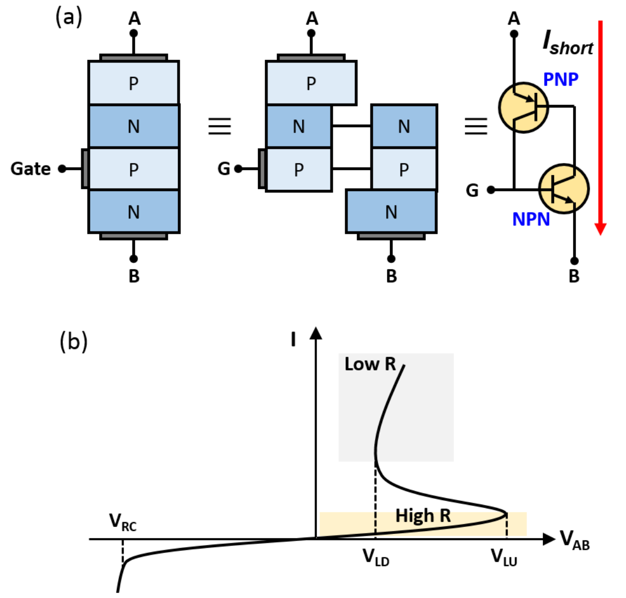


Fig. 2. (a) Thyristor Structure and (b) Voltage-Current Behavior

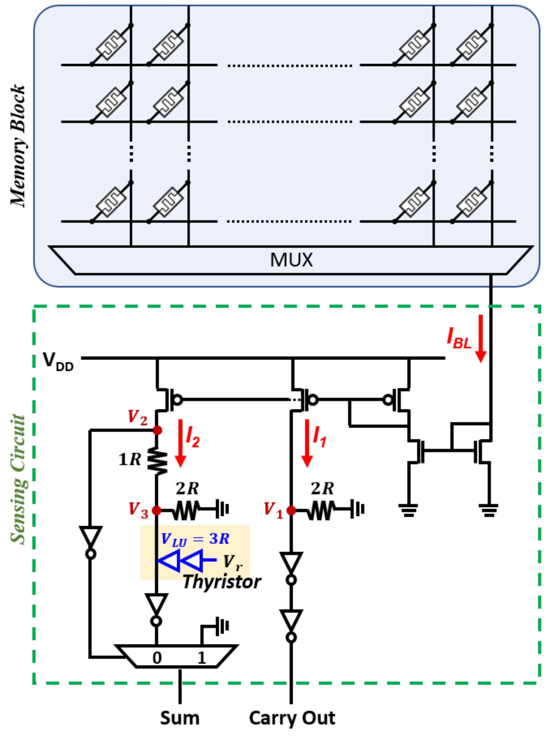


Fig. 3. Proposed Sensing Circuit for Addition

design computes the outputs by distinguishing the total current amount reached to the sensing circuit.

Fig. 4a presents the truth table of a full adder. The sum is the exclusive-or (XOR) result for the three inputs and the carry out is the majority function of the inputs. The three inputs are interchangeable in that the order of them does not affect the output. In the memsistor devices, the amount of the bitline current is the combination of one R_{on} and two R_{off}. Based on this characteristic, there are four different cases depending on the current amount, I_{000}, I_{100}, I_{110} and I_{111}, according to the number of high (0) and low (1) resistances in the memristors of activated rows.

Fig. 4b shows how the proposed circuit distinguishes the four current regions to create the desired C_{out} and *Sum*. In our circuit design, there are three major voltage nodes (i.e., V_1, V_2, and V_3 shown in Fig. 3) whose potential determine the final outputs of the *Sum* and C_{out} by the following digital logic gates. The voltage of each node is transferred as a function of the current in the selected bitline. V_{THR} is a threshold value which determines whether an input potential is interpreted as a logic 0 or 1 (i.e., any value less than V_{THR} is 0 and any value above V_{THR} is 1.) Let R_{thy} be the resistance of the thyristor explained in Section III-B. Then, the electric potentials at V_1, V_2 and V_3 are represented by $V_1 = I_1 \cdot (2R)$, $V_2 = I_2 \cdot (3R)$ and $V_3 = I_2 \cdot (2R \cdot R_{thy})/(2R + R_{thy})$, respectively.

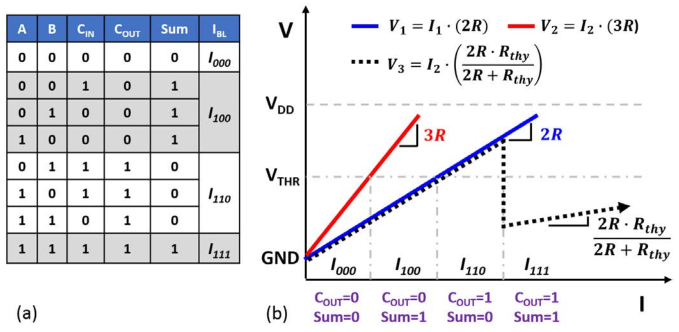
As for the carry-out function, V_1 node has higher electric potential than V_{THR} in cases of I_{110}, I_{111} when it delivers a logic 1 to C_{out} through two inverters which strengthen the signal. For the I_{000} and I_{100} cases, V_1 node has lower potential than V_{THR} and C_{out} presents a logic 0.

The *Sum* function uses branches where V_2 and V_3 nodes are located. As shown in Fig. 4b, V_2 node has lower potential than V_{THR} only in case of I_{000} and its inverted logic 1 is delivered to the MUX, pulling down the *Sum* potential to the ground, i.e., logic 0. In the opposite cases, i.e., I_{100}, I_{110} and I_{111}, V_2 has a logic 1, and the MUX delivers the data from the connection where V_3 is located, so that V_3 decides the outputs for the three cases. For I_{100} and I_{110} case, the V_3 shows either logic 0 or logic 1 depending on the input current in a similar way of V_1, since the thyristor is not activated in this region. However, once the current increases and reaches the I_{111} range, the latch-up occurs in the thyristor device, and thus the electric potential of V_3 abruptly falls below V_{THR}, making *Sum* logic 1. With this logic, our design is able to complete all *Sum* and C_{out} results. Once the final outputs are generated, we reset the thyristor for the next cycle, by invoking the write-back procedure to turn the thyristor resistance back to the original state. In case of N-bit addition, C_{out} is written back to the memory and is used to calculate the results for the next bit.

In our experimental setup, we assume $V_{THR} = V_{DD}/2$, and set R=20kΩ to yield the described voltage transfer functions. The latch-up also affects the potential of the V_2 node. However, the potential V_2 does not drop below V_{THR} in our design, since the thyristor resistance on the conductance state is very low compared to 2R, a higher portion of the supply voltage is applied between V_2 and V_3. This keeps the V_2 over the V_{THR} and V_3 below V_{THR} with a marginal window in the case of I_{111}.

D. Multiplier Design

Implementing multiplication in memory is challenging due to the large number of parallel computations and shift operations for each multiplicand bit. The multiplication is performed in three stages, partial product generation, fast addition, and final product generation. The partial product generation stage, generates n partial products, where n represents the size of



A	B	C_{IN}	C_{OUT}	Sum	I_{BL}
0	0	0	0	0	I_{000}
0	0	1	0	1	I_{100}
0	1	0	0	1	
1	0	0	0	1	
0	1	1	1	0	I_{110}
1	0	1	1	0	
1	1	0	1	0	
1	1	1	1	1	I_{111}

Fig. 4. Voltage Transfer as a Input Current (I_{BL})

the multiplier. They are propagated to the next stage. The fast addition method used in [12], optimizes the latency involved in the addition of the generated partial products. They implement tree-based carry save addition to push carry propagation to the last stage, hence enabling faster operations. A carry save adder implements half-addition at each bit. It takes in three inputs and generates two outputs, sum and carry, resulting in a 3:2 reduction. Successive carry save additions reduce the initial n partial products into two numbers which are then added in the final stage.

Although we use similar techniques as those in [12] to support multiplication, our work is different from the implementation perspective. We design novel circuits using thyristor device to enable single-cycle computations. For single-bit addition, which is the basic operation to implement MUL, the design in previous work has larger latency than the proposed design. Moreover, it uses interconnects to enable shift operations which requires large number of additional transistors. It induces significant area overhead, which grows exponentially as the block size increases. Our design generates intermediate results at the sensing circuits and writes them back for further computations. Hence, the shift operations are dealt with while writing the results back to the memory. This does not require the expensive interconnects used by the previous work. Furthermore, we can utilize the thyristor write-back, which happens at the end of the ADD operation, to hide the write latency.

IV. Experimental Results

A. Experimental Setup

Performance and energy consumption of the proposed design have been obtained from circuit-level simulations in a 45nm CMOS process and design kits of Cadence Virtuoso and Spectre simulators. We use VTEAM memristor model [35] for our resistance-based memory design and simulation with R_{on} and R_{off} of 10KΩ and 10MΩ respectively. The thyristor device has been designed and simulated using Silvaco ATLAS TCAD software to investigate the latch-up effects to the PIM architecture and optimize the process conditions.

To evaluate the efficiency of LUPIS for practical applications by designing a cycle-accurate simulator which models the memory functionality. We compare proposed design with

TABLE I
PERFORMANCE OF 1-BIT ADDER FOR LUPIS AND OTHER TECHNOLOGIES

	[25]	[26]	[12]	[37]	**This work**
No. Memristors	3N+5	3N+3	3N+8	N+2	**3N**
Cell efficiency	38%	50%	27%	33%	**100%**
Latency	149.6ns	31.9ns	14.3ns	9.9ns	**33.3ps**
Energy	3237fJ	690fJ	289fJ	214fJ	**7.9fJ**

AMD Southern Island GPU, Radeon HD 7970 device, which is one of the most recent GP-GPU architectures. We compared the efficiency of LUPIS to the GPU architecture for four OpenCL applications: *Sobel, Robert, Fast Fourier transform (FFT)* and *DwHaar1D*. For image processing, we used random images from *Caltech 101* [36] library, while for non-image processing applications inputs were generated randomly. These applications involve many additions and multiplications, and we further approximated other common operations such as square root with the two operations. In the application level, we also modified source code of the applications so that applications utilize PIM-based addition/multiplications as much as possible, e.g., using Taylor expansion.

B. Device Optimization

In order to optimize the process conditions of a thyristor used in our proposed design, a device simulation was performed using Silvaco Atlas. As shown in Fig. 5a, we used a lateral PNPN structure consisting of a p-type Si substrate, n+contact, n-well, and p+contact, with doping concentrations of $1 \times 10^{16} cm^{-3}$, $1 \times 10^{20} cm^{-3}$, $2 \times 10^{16} cm^{-3}$ and $1 \times 10^{20} cm^{-3}$. A width of 0.1 um was used for the 2-dimensional simulation. For the latch up simulation, 1.0 us was used for Shockley-Read-Hall life times for both electrons and holes, and the Selberherr model was applied for the impact ionization. Fig. 5(b) illustrates the simulation result with d_1=d_2=$0.2um$. Based on this simulation result, we exploited a V_{LD} of 0.89 V, a V_{LU} of 0.98 V, a R_H of 1.9 MΩ, and a R_L of 1.7 kΩ, where R_L and R_H are the resistance of thyristor in the higher and lower conductance state respectively. As shown in Fig. 5c and Fig. 5d, R_H and V_{LU} can be easily controlled by changing either the device structure or the doping concentration of the p- and n- regions. This is because R_H and V_{LU} in the high resistance regime are dominated by the characteristics of the reverse biased PN junction at the central p- and n- regions, so we can tune them by varying the device structure and/or doping process conditions. Since there is a clear dependency between R_H and V_{LU}, i.e. V_{LU} increases as increasing R_H based on our simulation results, the thyristor ensure a stable support of device characteristics that upper circuit and system design need with marginal process window.

C. Energy and Performance Comparison

Circuit level: Table I shows the 1-bit addition results of proposed LUPIS and other prior technologies. As explained in Section II, most of the current PIM approaches including selected ones [12], [25], [26], [37] use bitwise-based logics (i.e. calculating IMP, NOR, NOT) to execute 1-bit addition. Thus, they require inevitable sub-cycle executions for the

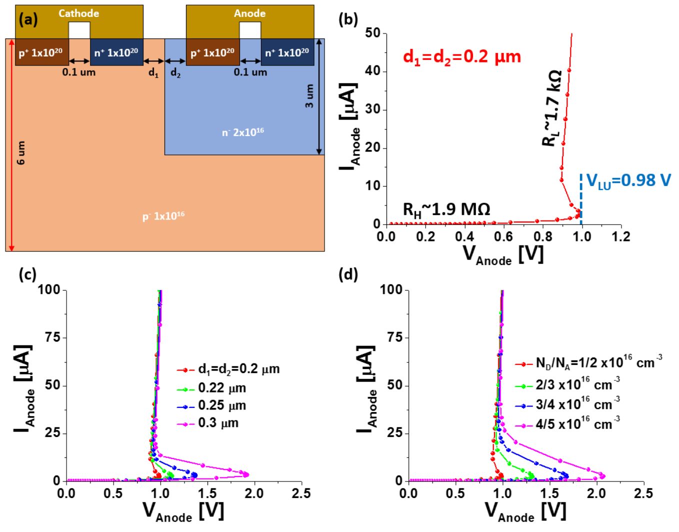


Fig. 5. (a) A cross-sectional schematic and current-voltage characteristics of the simulated lateral PNPN structure with a width of 0.1 um, (b) d_1=d_2=0.2 um, (c) various d_1=d_2 for 0.2, 0.22, 0.25, and 0.3 um, and (d) various N_D/N_A for 1/2×10^{16}, 2/3×10^{16}, 3/4×10^{16}, and 4/5×$10^{16}cm^{-3}$

intermediate bitwise computations, creating huge latency bottleneck, e.g., the long latency of SET cycle [12]. In contrast, since our LUPIS design executes the ADD operation in the sensing circuit in a single cycle, the total latency is determined only by the sensing circuit delay, without any extra latency from the memristor. Thus, our proposed design can achieve higher speedup than all the other techniques. Furthermore, our design shows superior cell efficiency since it does not require additional cells. This makes our design a good candidate for NVM-based PIM architectures, in particular, for the storage class memory which should handle a large amount of data.

Application level: There are several applications which can benefit by the PIM-based addition and multiplication. Fig. 6 shows the speedup and energy efficiency improvement of, i) the proposed design and ii) a state-of-the-art PIM technique, APIM [12], over the AMD GPU core. The results present that our proposed design can achieve significantly better energy and performance efficiency. Apart from the superior efficiency improvement, our evaluation also shows that LUPIS energy consumption increases linearly with the data set size, since the PIM capability can highly hide the cost of data movements. In contrast, the energy and execution time of the GPU case do not scale linearly with the data size, as the larger dataset requires higher costs for the data movements before processing. To sum up, our design can achieve up to 62× speedup and 484× lower energy consumption than the GPU architecture. As compared to APIM, the results present 12.7× and 20.9× higher efficiency for speedup and energy respectively.

D. Overhead

Fig. 7 shows the area and latency overhead of our design compared to the TC-adder [37], the most competitive design in cell efficiency, as described in Section IV-C. The area overhead has been estimated by the ratio of the additional number of cells and gates as compared to the conventional memory design. As shown in Fig. 7a, LUPIS has 21% area overhead, which outperforms the TC-Adder by 10× since it just takes insignificant modifications to the conventional CSA circuit and no additional cells are required.

As for the latency overhead, our design requires the write back step after the sensing operation to initialize the state of thyristor for the next operations. As shown in Fig. 7b, the overhead caused by the write back inclusion is one cycle. This is negligible compared to the latency in the TC-Adder requiring 9 cycles per operation [37]. Furthermore, the overhead due to the write back step can be utilizing in the the MUL operation to hide the latency of shift operations as explained in Sec. III-D.

V. Conclusion

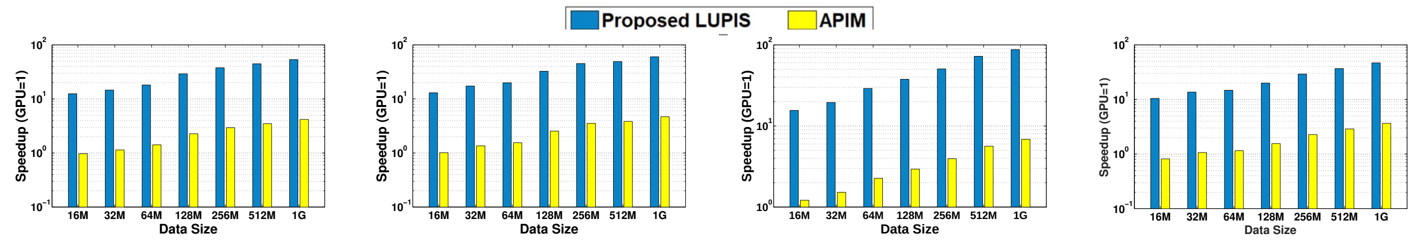
We have presented an ultra efficient PIM architecture which effectively enables addition and multiplication inside memory by utilizing the thyristor latch-up effect. The proposed design also addresses the low cell-efficiency issue of other PIM technologies due to redundant cell requirements for logic operations by executing the calculations in the sensing circuitry. The experimental results show that, compared to a state-of-the-art PIM accelerator, our design presents 12.7× and 20.9× improvement of latency and energy consumption.

VI. Acknowledgment

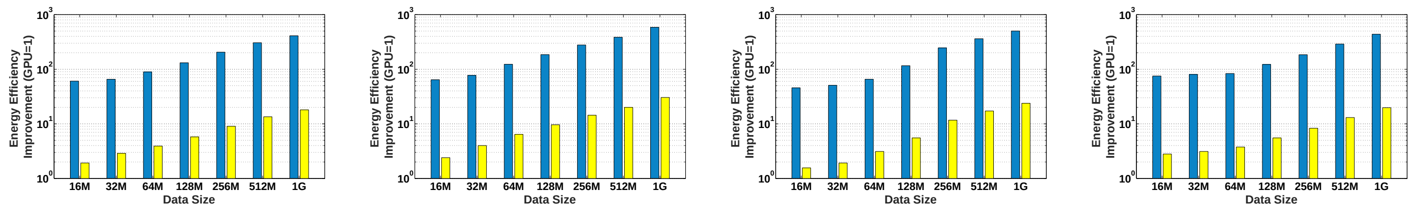
This work was supported by NSF grants #1730158 and #1527034.

References

[1] I. Wigmore, "Internet of things (iot)," *TechTarget*, 2014.
[2] B. Yao *et al.*, "Multifractal analysis of image profiles for the characterization and detection of defects in additive manufacturing," *Journal of Manufacturing Science and Engineering*, 2017.
[3] R. Nair, S. F. Antao, C. Bertolli, P. Bose, J. R. Brunheroto, T. Chen, C.-Y. Cher, C. H. Costa, J. Doi, C. Evangelinos, *et al.*, "Active memory cube: A processing-in-memory architecture for exascale systems," *IBM Journal of Research and Development*, vol. 59, no. 2/3, pp. 17–1, 2015.
[4] V. Seshadri, K. Hsieh, A. Boroum, D. Lee, M. A. Kozuch, O. Mutlu, P. B. Gibbons, and T. C. Mowry, "Fast bulk bitwise and and or in dram," *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 127–131, 2015.
[5] G. H. Loh, N. Jayasena, M. Oskin, M. Nutter, D. Roberts, M. Meswani, D. P. Zhang, and M. Ignatowski, "A processing in memory taxonomy and a case for studying fixed-function pim," in *Workshop on Near-Data Processing (WoNDP)*, 2013.
[6] A. M. Aly, A. Sallam, B. M. Gnanasekaran, L.-V. Nguyen-Dinh, W. G. Aref, M. Ouzzani, and A. Ghafoor, "M3: Stream processing on main-memory mapreduce," in *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pp. 1253–1256, IEEE, 2012.
[7] M. Imani *et al.*, "Nngine: Ultra-efficient nearest neighbor accelerator based on in-memory computing," in *ICRC*, IEEE.
[8] Y. Kim *et al.*, "Orchard: Visual object recognition accelerator based on approximate in-memory processing," in *ICCAD*, 2017.
[9] M. Imani *et al.*, "Nvalt: Non-volatile approximate lookup table for gpu acceleration," *Embedded Systems Letters*, 2017.
[10] M. Imani *et al.*, "Resistive cam acceleration for tunable approximate computing," *IEEE Transactions on Emerging Topics in Computing*, 2017.
[11] M. S. Razlighi *et al.*, "Looknn: Neural network with no multiplication," in *DATE*, pp. 1775–1780, IEEE, 2017.
[12] M. Imani, S. Gupta, and T. Rosing, "Ultra-efficient processing in-memory for data intensive applications," in *Proceedings of the 54th Annual Design Automation Conference 2017*, p. 6, ACM, 2017.
[13] M. Imani *et al.*, "Efficient neural network acceleration on gpgpu using content addressable memory," in *DATE*, pp. 1026–1031, IEEE, 2017.



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



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

Fig. 6. Speedup and Energy Efficiency of Proposed LUPIS and APIM [12] over Different Applications.

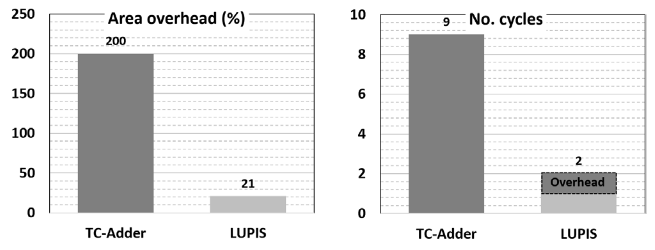


Fig. 7. The Overhead of Area (a) and Latency (b)

[14] M. Imani *et al.*, "Exploring hyperdimensional associative memory," in *HPCA*, IEEE, 2017.

[15] M. Imani *et al.*, "Acam: Approximate computing based on adaptive associative memory with online learning.," in *ISLPED*, pp. 162–167, 2016.

[16] M. Imani *et al.*, "Processing acceleration with resistive memory-based computation," in *MEMSYS*, pp. 208–210, ACM, 2016.

[17] M. Imani *et al.*, "Masc: Ultra-low energy multiple-access single-charge tcam for approximate computing," in *DATE*, pp. 373–378, IEEE, 2016.

[18] M. Imani *et al.*, "Multi-stage tunable approximate search in resistive associative memory," *TMSCS*, 2017.

[19] M. Imani *et al.*, "Approximate computing using multiple-access single-charge associative memory," *TETC*, 2016.

[20] M. Imani, Y. Kim, and T. Rosing, "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.

[21] S. Li, C. Xu, Q. Zou, J. Zhao, Y. Lu, and Y. Xie, "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," in *Design Automation Conference (DAC), 2016 53nd ACM/EDAC/IEEE*, pp. 1–6, IEEE, 2016.

[22] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *Proceedings of the 43rd International Symposium on Computer Architecture*, pp. 14–26, IEEE Press, 2016.

[23] M. Imani *et al.*, "Cfpu: Configurable floating point multiplier for energy-efficient computing," in *IEEE/ACM DAC*, p. 76, ACM, 2017.

[24] J. Sim *et al.*, "Enabling efficient system design using vertical nanowire transistor current mode logic,"

[25] 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.

[26] S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Memristor-based material implication (imply) logic: Design principles and methodologies," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 10, pp. 2054–2066, 2014.

[27] M. Imani, A. Rahimi, and T. S. Rosing, "Resistive configurable associative memory for approximate computing," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016*, pp. 1327–1332, IEEE, 2016.

[28] V. Seshadri and O. Mutlu, "The processing using memory paradigm: In-dram bulk copy, initialization, bitwise and and or," *arXiv preprint arXiv:1610.09603*, 2016.

[29] Y. Wang, Y. Han, L. Zhang, H. Li, and X. Li, "Propram: exploiting the transparent logic resources in non-volatile memory for near data computing," in *Proceedings of the 52nd Annual Design Automation Conference*, p. 47, ACM, 2015.

[30] J. Ahn, S. Yoo, O. Mutlu, and K. Choi, "Pim-enabled instructions: A low-overhead, locality-aware processing-in-memory architecture," in *Computer Architecture (ISCA), 2015 ACM/IEEE 42nd Annual International Symposium on*, pp. 336–348, IEEE, 2015.

[31] S. Kvatinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Magicmemristor-aided logic," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 11, pp. 895–899, 2014.

[32] M.-F. Chang, S.-J. Shen, C.-C. Liu, C.-W. Wu, Y.-F. Lin, Y.-C. King, C.-J. Lin, H.-J. Liao, Y.-D. Chih, and H. Yamauchi, "An offset-tolerant fast-random-read current-sampling-based sense amplifier for small-cell-current nonvolatile memory," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 3, pp. 864–877, 2013.

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

[34] X. Tong, J. Luo, H. Wu, Q. Liang, H. Zhong, H. Zhu, and C. Zhao, "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*, vol. 32, no. 2, p. 021205, 2014.

[35] 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.

[36] "Caltech 101." http://www.vision.caltech.edu/ImageDatasets/Caltech101/.

[37] A. Siemon, S. Menzel, R. Waser, and E. Linn, "A complementary resistive switch-based crossbar array adder," *IEEE journal on emerging and selected topics in circuits and systems*, vol. 5, no. 1, pp. 64–74, 2015.