Triple-Skipping Near-MRAM Computing Framework for AIoT Era

Juntong Chen, Hao Cai, Bo Liu, Jun Yang
National ASIC System Engineering Research Center
Southeast University
Nanjing, China
{jtchen, hao.cai, liubo_cnasic, dragon}@seu.edu.cn

***Abstract*—Near memory computing (NMC) paradigm shows great significance in non-von Neumann architecture to reduce data movement. The normally-off and instance-on characteristics of spin-transfer torque magnetic random access memory (STT-MRAM) promise energy-efficient storage in the AIoT era. To avoid unnecessary memory-related processing, we propose a novel write-read-calculation triple-skipping (TS) NMC for multiply-accumulate (MAC) operation with minimally modified peripheral circuits. The proposed TS-NMC is evaluated with a custom micro control unit (MCU) in 28-nm high-K metal gate (HKMG) CMOS process and foundry announced universal two-transistor two-magnetic tunnel junction (2T-2MTJ) MRAM cell. The framework consists of a sparse flag which is defined in extra STT-MRAM columns with only 0.73% area overhead, and a calculation block for NMC logic with 9.9% overhead. The TS-NMC can successfully work at 0.6-V supply voltage under 20MHz. This Near-MRAM framework can offer up to ∼95.6% energy saving compared to commercial SRAM refer to ultra-low-power benchmark (ULP-Benchmark). Classification task on MNIST takes 13nJ/pattern. The energy access of memory, calculation, and the total can be reduced by 52.49×, 2.7×, and 11.3× respectively from the TS scheme.**

***Index Terms*—STT-MRAM, near memory computing, neural network, MAC, triple-skipping, AIoT**

I. Introduction

Artificial intelligence (AI) has tremendously promoted the development of Internet of Things (IoT) applications and edge computing. Neural network (NN) is an effective implementation method in the AIoT era [1]. However, high demands for memory access and computing are running counter to energy harvest and resource-constrained AIoT edge devices. These restrictions put forward various challenges. In the traditional von Neumann architecture, frequent data transfers between the memory array and computation units result in large energy and latency overhead. On the other hand, data retention cost as leakage power consumption is significant in previous static random access memory (SRAM) based on-chip memory, as AIoT circuits and systems mostly operate at the standby mode [2].

Towards non-von Neumann computing, the near-memory-computing (NMC) alternative targets at processing closed to where the data is stored. This memory-centric approach couples computing elements as close as possible to the data for minimizing the expensive data movements [3]. Unlike the in-memory-computing (IMC) approach, NMC has the benefit of minimally re-design the bit-cell array, peripheral circuits, and memory sub-system, which presents great potential in the design phase.

Nonvolatile memories (NVMs) provide opportunities for energy-efficient applications. Among them, spin-transfer torque magnetic tunnel junction (STT-MTJ) based MRAM is regarded as a promising candidate for next-generation on-chip memory, for the reasons that free standby power, instance-on, fast access speed, and compatibility with CMOS process [4]. The commercial MRAM bit-cells, *e.g.,* two-transistor two-magnetic tunnel junction (2T-2MTJ), have been applied to NMC and IMC design as a breakthrough in energy efficiency.

Apart from near-MRAM computing framework, the algorithm deployed to AIoT devices should be carefully optimized. Approaches like compression, pruning, and quantization are of great effects to minimize data storage and computing resources in NN applications [5]. Considering that vector-matrix calculation based on multiply-accumulate (MAC) is the major operation, the feature exploiting of sparse vector/matrix can effectively reduce the calculation. However, the precondition is that the processed data should be initially fetched from on/off-chip memories [6]–[8], which consumes lots of unnecessary energy. To solve this problem, we firstly propose a write-read-calculation triple-skipping (TS) near-MRAM computing framework for avoiding unnecessary memory-related processing and saving energy. The main contributions include:

- Minimally modifications of commercial MRAM technology are involved and considerable amounts of unnecessary computations are eliminated according to the TS scheme.
- The triple-skipping near-MRAM computing unit is designed with universal 2T-2M MRAM cell. The energy access of memory, calculation, and the total can be reduced by 52.49×, 2.7×, and 11.3× respectively by exploiting the sparsity of all operands.
- The design space exploration of TS-NMC is evaluated in a custom micro-control unit (MCU), including energy earnings, area penalty, and application scenario.

The remainder of the paper is organized as follows: Section II gives the preliminaries of MRAM-based near-memory computing. Section III presents the proposed TS-NMC framework, Section IV evaluates the performance of this novel framework. Finally, Section V makes the conclusion.

978-3-9819263-6-1/DATE22/©2022 EDAA

II. Preliminaries

A. Universal 2T-2M STT-MRAM cell

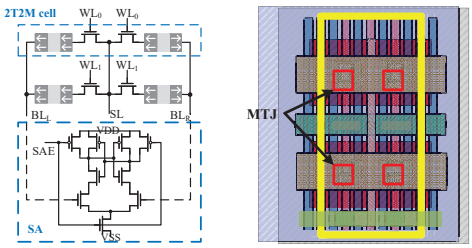


Fig. 1. Structure of a 2T-2M STT-MRAM cell with perpendicular MTJ, typical sensing circuit, and layout overview of 40nm MTJ.

As shown in Fig. 1, the commercial 2T-2M cell is developed from the 1T-1M counterpart with improved access latency. It uses perpendicular MTJ consisting of two ferromagnetic layers separated by an oxide barrier, and the corresponding state of magnetization of the two ferromagnetic layers parallel (P) or anti-parallel (AP) determines the nanopillar resistance (R_P, R_{AP}). The ferromagnetic configuration states of 2-MTJ are always in the opposite, thus a differential sensing scheme can be implemented by this self-reference structure which is more reliable during data access than 1T-1M structure [9], especially at high speed or ultra-low supply voltage.

B. Near-MRAM computing

The NMC approach prefers coupling computation blocks and memory together with minimally design modifications, especially that of peripheral circuits and memory array contradicts with foundry rules and extra area penalty. As MRAM normally-off and instance-on characteristics are appropriate for edge computing. Combining STT-MRAM with NMC, [9] firstly integrated shift/rotate function in sensing amplifiers of MRAM array for security-aware mobile devices. To further reduce the load of the bus, comprehensive functions need to be included.

C. Neural network and sparse vector/matrix multiplication

Sparse neural networks have been intensively studied for speedup and saving resources on algorithms. The challenge lies in keeping consistent with hardware implementation, especially on non-traditional architecture to maximally take advantage of sparse operation. Eyeriss made use of the sparsity by compression and skipping computation of zero-valued activations in processors [10]. [11] introduced a sparsity transfer algorithm from hardware aspects to exploit both inputs and weights sparsity. SCNN [12] used Cartesian product operations between non-zero-valued weights and activations. Generally, encoding/decoding schemes are customized for the deployments, in that case, the amounts of data access are always the premise. Unfortunately, NN is a type of memory-intensive application which means the energy consumption of accessing memory cannot be underestimated. In this study, we propose a novel MRAM-based framework called TS-NMC which supports sparse neural networks by skipping write-read-calculation.

III. Proposed triple skipping near-MRAM computing framework

Fig. 2 provides an overview of the TS-NMC architecture. In Fig. 3(b), the flow chart of this TS-NMC is shown. Before loading data into memory, a sparse flag is generated according to data, and it can instruct the memory-related process to perform triple skipping, compared to the traditional flow, additional write/read skipping is realized.

Algorithm 1: Sparse Flag Generation

Input: number A
Output: sparse flag F
A represents an n-bits binary number ($n > 1, n \in \mathbb{N}^+$);
F represents whether $A = 0$;
$A[n-1:0] = A[MSB:LSB]$;
while $n \neq 2$ **do**
  $i = 0$;
  if n *mod* 2 **then**
    for $i < (n-1)/2$ **do**
      $A[i] = A[2i]$ *or* $A[2i+1]$;
      $i = i + 1$;
    end
    $A[i] = A[2i]$;
    $n = i + 1$;
  else
    for $i < n/2$ **do**
      $A[i] = A[2i]$ *or* $A[2i+1]$;
      $i = i + 1$;
    end
    $n = i$;
  end
end
$F = \overline{A[0] \text{ or } A[1]}$

A. Spare flag generation

In equation (1), when $\vec{a} = \mathbf{0}$ *or* $\vec{w} = \mathbf{0}$, the equation (2) can be skipped to save computational energy. However, to judge whether these two operators equal $\mathbf{0}$, the memory assessment is required at first. Assume that every element in $\vec{a}$ and $\vec{w}$ is a j-bits binary number so that each vector occupies n=ij bits. Sensing $2n$ bits data still consumes large energy. Thus, the sparse flag is proposed.

$$\vec{a} = (a_0, a_1, a_2 \cdots, a_{i-1})$$
$$\vec{w} = (w_0, w_1, w_2 \cdots, w_{i-1}), i \in \mathbb{N}^+ \quad (1)$$
$$\vec{a} \cdot \vec{w} = \sum a_m \times w_m \quad (2)$$

According to equation (3) and algorithm 1, sparse flag generator calculates the flag of $\vec{a}$ which represents whether these data equal $\mathbf{0}$. The circuit of the generator consists of n-2 two inputs 'OR' gates and 1 two inputs 'NOR' gate. During the write operation, only flag information needs to be written into memory when the flag is '1'. In the read operation, an additional sensing cycle for single flag data is initially required. Then, the flag data will decide to turn on/off the sensing circuit in the next cycle, and both sensing and calculation can be skipped. Notice that theoretically $n\times$ read/write energy reduction is obtained with only $1/n$ memory area overhead and an extra sensing cycle for the case that $\vec{a} = \mathbf{0}$ *or* $\vec{w} = \mathbf{0}$.

$$flag = \begin{cases} 1, & \forall a_k = 0 \\ 0, & \exists a_k \neq 0, \ k = 0, 1, 2 \cdots i-1 \end{cases} \quad (3)$$

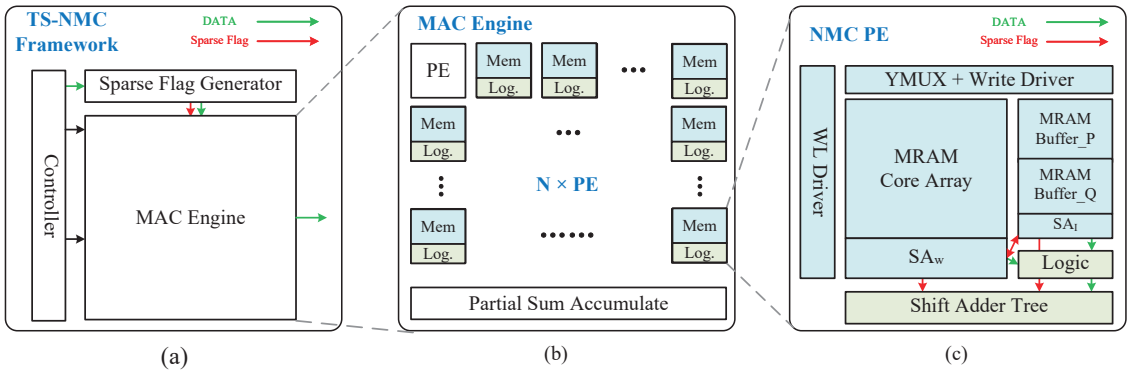


Fig. 2. (a) An overview of the proposed triple-skipping near-MRAM computing framework, a sparse flag generator is configured to mark whether a set of data equals to **0**. (b) The structure of the NMC MAC engine. (c) The NMC processing element construction.

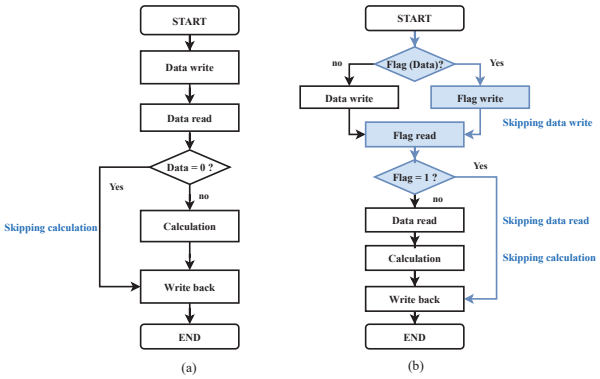


Fig. 3. (a) The traditional flowchart of processing zero vector. (b) The TS-NMC flowchart of processing zero vector.

B. NMC processing element

The design principle of the proposed sparsity-aware near memory computing processing element (NMC-PE) is to lower the sensing amplifier (SA) circuits operating cycles and register/combinational logic switching maximally to save energy consumption. The framework of NMC-PE is demonstrated in Fig. 2(c), word line (WL) driver is used for address selection, the multiple WLs are corresponding to multiple $\vec{w}$. MRAM core array stores the entire matrix ($e.g.$ weights) and ping-pong MRAM buffer stores the vector ($e.g.$ activations). The SA and logic 'AND' perform multiplication, and finally shift adder tree (SAT) completes the accumulation.

$$\begin{aligned} D_{in} &= \vec{w} \\ \vec{i}_m &= (a_0[m], a_1[m] \cdots a_7[m]) \qquad (4) \\ psum &= a_0[m]\&w_0 + a_1[m]\&w_1 \cdots + a_7[m]\&w_7 \\ &\quad + (psum \ll 1) \qquad (5) \end{aligned}$$

The calculation sequence is shown in Fig. 4. Assume that i=j=8, n=64, a row of MRAM core array stores 1×64 bits $\vec{w}$ and 1 flag bit (f_w), MRAM buffer stores 8×8 bits $\vec{a}$ and 1 flag bit (f_i). Firstly, sae_{flag} will enable flag sensing, at edge ①, $flag$ signal will be calculated as $f_w|f_i$. If $flag$ is '0':

(a) $weight$ and $input$ will be updated as equation (4) (m=7) and partial sum ($psum$) will be reset to zero at edge ②.
(b) During the next cycle, SAT calculates S_0 as equation (5) (m=7). Meanwhile, SA_w turns off and maintains D_{in}.
(c) At edge ③, $input$ and $psum$ will be updated to $\vec{i}_6$ and S_0, then repeat step (b) and (c) until $psum$ is updated to S_7.

If $flag$ equals to '1', it represents that at least one of $\vec{w}, \vec{a}$ is **0**, thus calculation and sensing phase can be skipped. For further energy reduction, $psum, weight, input$ will not update at edge ④ to save switching power in registers. This behavior will keep the logic in SAT block as an immutable case. The SAT result will be set to '0' according to combinational logic.

C. MAC Engine

Although NMC-PE can complete the assigned task of equation (2) (data width equals to j), these parameters are finite to avoid unexpected large parasitic capacitance of memory array. Thus, the long vector needs to be divided into several numbers of NMC-PE. As shown in Fig. 2(b), concatenate $N\times$ NMC-PE is capable to execute equation (6).

$$\vec{w}_{dim} \times \vec{a}_{dim}, \ dim = N \times i \qquad (6)$$

Notice that all NMC-PEs work in parallel, and the partial results are accumulated in a partial sum accumulator. "vector-matrix multiplication" as the key operation in the NN accelerator is performed by this MAC engine.

IV. Experimental Results

A. Simulation setup

In this subsection, to evaluate the proposed TS-NMC strategy, we provide a device-to-system level analysis and verify the performance in a related application scenario.

*1) **Bit-cell and array-level configuration**:* The MTJ device (CoFeB/MgO/CoFeB) is configured in the Verilog-A model. Table I lists the important MTJ parameters used in simulations which are sourced from [13]. MTJ process has been configured according to [14].

The industrial 28-nm CMOS design kit is used to implement memory array and peripheral circuits. In each NMC-PE (as shown in Fig. 4(a)), MRAM sub-array is implemented as

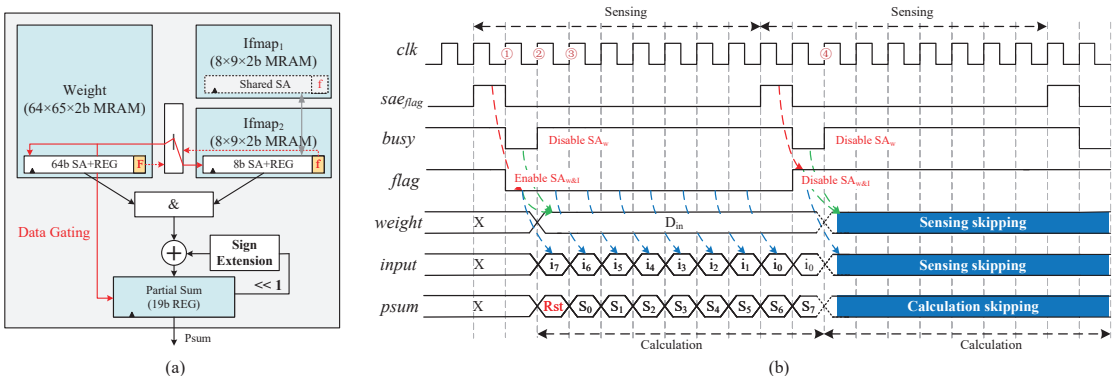


Fig. 4. (a) The design of NMC-PE to perform sparsity-aware vector multiplication. (b) The timing diagram of sparsity aware NMC-PE.

64×65×2 (∼1KB), where 64 is row number, 65 is (8×8-bits data+1 flag), 2 is the number of sub-arrays which share the same peripheral circuits. Each MRAM buffer is shaped as 8×9×2, where 8 is data width, 9 is (8 data ×1 specified bit +1 flag).

For the comparison, SRAM (with retention mode) based PE is generated by an industrial 28-nm memory compiler, with the same configurations as STT-MRAM.

*2) **System-level evaluation**:* As shown in Fig. 6, the system-level evaluation is performed within a MCU platform, which consists of a Cortex-M3 (CM3) processor, clock module, power module, off-chip FLASH, and I/O devices. The proposed TS-NMC is connected to CM3 through advanced micro-controller bus architecture (AMBA). Both normal/sparsity aware SRAM (SRAM-N/SRAM-S) and MRAM-N/MRAM-S based NMC-PE are designed.

*3) **Application scenario**:* As the same with most AI applications, the MAC and memory access intensive scenario is validated. The system performs an image classification task on the MNIST dataset, a 50816 synapses neural network model is constructed, and every synapse is quantized to 8-bits.

TABLE I
MAIN PHYSICAL PARAMETERS IN USED MTJ COMPACT MODEL [13]

Parameter	Description	Proposed	Variation
ΔH_0	Activation energy	0.8eV	N/A
TMR	TMR ratio	50% - 200%	σ=5%
T_{ox}(nm)	MgO thickness	0.85	σ=0.01
T_{FL}(nm)	Free-layer thickness	1.3	σ=0.01
$Area$	MTJ layout surface	40nm×40nm×π/4	N/A
R_P,R_{AP}	MTJ resistance	4kΩ, 6-12kΩ	dependence
I_{c0}	Switching current	Minimum 40μA	dependence
α	Damping factor	0.027	N/A
β	Shape parameter	1.5	N/A
H_K	Effective anisotropy field	$1.14\times10^5 A/m$	N/A
M_s	Saturation magnetization	$1.26\times10^6 A/m$	N/A

B. Experimental Results

To evaluate the power consumption and classification accuracy of the proposed TS-NMC framework, the prototype system is functional with logic supply 0.6-0.8-V under 40MHz. The power consumption is evaluated with Synopses PTPX and SPICE at 25°C typical-typical (TT) corner.

*1) **Energy saving of memory**:* According to the ULP-Benchmark [15], the active portion of the benchmark is only

Fig. 5. Layout overview of main MRAM array (weight, 64×65×2b), 2 sparse flags are implemented at the end of columns with 1 SA and shift adder tree is at bottom of the array.

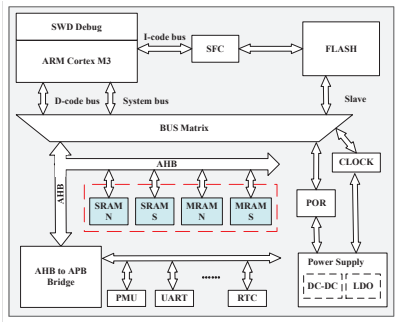
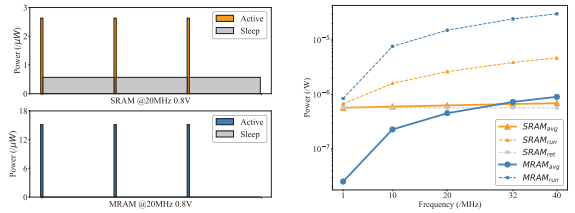


Fig. 6. System-level evaluation platform.

running for ∼3% of the total runtime, it requires data to be saved during deep sleep through the use of retention RAM. As shown in Fig. 7(a), SRAM is implemented with lower active power than STT-MRAM at 20-MHz under 0.8-V supply voltage. However, tremendous sleep power and sleep time result in higher average power/energy dissipation. Design trade-off should be taken into consideration between operation frequency and power consumption, as resistive-type STT-MRAM is not suitable for high-speed application. According to Fig. 7(b), the average power of STT-MRAM based NMC-PE is more appropriate below 30-MHz, each STT-MRAM (1KB) array can save ∼95.6% compared to SRAM at 1MHz.



(a) (b)

Fig. 7. (a) Power consumption of SRAM vs. MRAM according to ULPBenchmark. (b) The comparison of SRAM and MRAM under different frequencies.

*2) **Energy benefits of triple skipping**:* Fig. 8 shows the normalized power breakdown of a single NMC-PE. In the non-zero multiplication, SA senses all the data and SAT performs near memory vector-matrix multiplication. During zero multiplication, SA only senses the 2-bits flag and then turns off, all the data registers maintain the value through data gating (see Fig. 4), the state of combinational logic will not flip in the calculation stage, which reduce the dynamic power. The result of NMC-PE is set to '0' through AND gates which costs a little combinational energy. The total power of zero vector calculation can be reduced to 11.3× compared to normal calculation. The memory, sequential (clock network and register), and combination part are reduced 52.49×, 4.79× and 2.24× respectively. Thus avoiding memory assess and logic switching behavior are effective methods for energy saving.

*3) **Area overhead of triple skipping strategy**:* The layout of PE is shown in Fig. 5. Every 8 bytes generate 1 sparse flag. Due to no modification of row/column decoder, area penalty only includes 1 col/8 bytes which is 0.73%. The SAT can effectively reduce the NMC calculation logic area overhead with 9.9%. Further, notice that the main area consumption is resulted by SA, write driver and column decoder, the extension of rows will be friendly to area cost, *e.g.* array capacity from 64×65×2b to 128/256×65×2b, the area overhead of sparse flag remains 0.74%, however, the area overhead of NMC calculation logic can be reduced to 8.2%/5.5%.

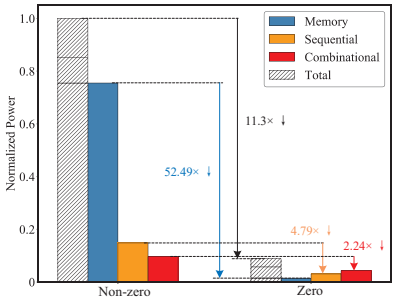
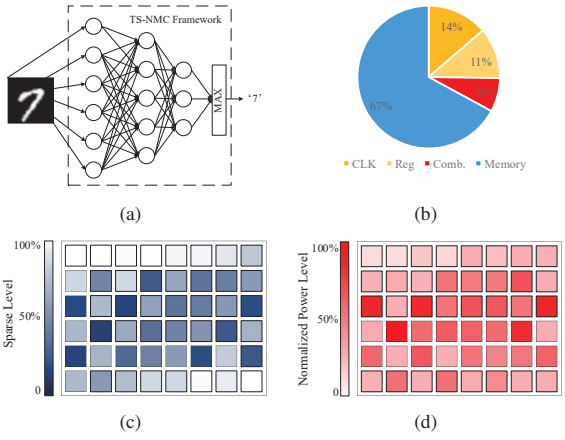


Fig. 8. Power breakdown of non-zero and zero multiplication.

*4) **Application analysis**:* To evaluate the performance of the TS-NMC framework, a memory access intensive application is performed. A 50816 synapses neural network model for MNIST dataset classification is constructed (see Fig. 9(a)). The weights and inputs of this model are stored in STT-MRAM. In the run phase, inputs (Ifmap) will update according to the input image, in the standby/sleep phase, the TS-NMC can be shut off and data will be kept in STT-MRAM without power consumption. Besides MAC operation, the activation function (ReLU) and quantized layer (8-bits quantification) are embedded in partial sum accumulate (See Fig. 2(b), which results in more energy of combinational part than single NMC-PE). As demonstrated in Fig. 9(b), the total energy consumption can be divided into 4 parts (clock network, register, combinational, and memory). Because of the larger sensing margin, 2T-2M structure allows lower operating voltage to save energy, *e.g.* at 0.6-V, compared



(a) (b) (c) (d)

Fig. 9. (a) MNIST classification. (b) Power consumption of classification at 0.6-V, 20Mhz. (c) Sparsity level of PEs according to MNIST dataset. (d) The normalized power distribution of PEs in classification.

with sensing yield of 1T-1M structure (99.63%), 2T-2M structure is 99.79% ($> 3\sigma$) with higher reliability. Each pattern classification takes 13 nJ on average with 98.6% inference accuracy at 0.6-V. Furthermore, the statistics of sparsity level are shown in Fig. 9(c), which shows the percentage of TS scheme used in the classification. The increased sparsity indicates a higher possibility of corresponding PEs processing zero MAC, these PEs can skip write-read-calculation using the TS scheme. Power distribution is shown in Fig. 9(d), which consists of sparsity (reduced 54.9% on average), the leakage power results in non-zero power PEs. Due to the analog IMC suffering from limited data precision and large power of high-precision ADC, Table II shows the superiority of this work from the aspects of parameters width, inference accuracy, and energy consumption for overall consideration.

The sparsity information of application mainly stems from two aspects. One is the widely used ReLU activation function mapping negative value to zero. The other is that pruning and quantification for network model compression increase the sparsity of the model, which are effective to minimize either storage or consumption of the application, especially in resource-limited AIoT devices.

TABLE II
COMPARISON WITH REFERENCE WORKS

	IEDM[1] [16]	TNNLS[1] [17]	DAC[1] [18]	This work
Type	NMC[2] RRAM	CIM FLASH	NMC[2] SOT	NMC STT
Parameters	1/8 bits	N/A	8 bits	8 bits
Accuracy	98.4%/N/A	94.7%	99.4%	98.6%
Energy/nJ[3]	25/80/img	20/img	740/img	13/img

[1] Published in 2018.
[2] These works use in-memory and near-memory logic simultaneously to perform classification.
[3] Classification on MNIST dataset.

V. CONCLUSION

In this work, a triple-skipping near-MRAM computing framework for AIoT applications scenarios was proposed. Universal 2T-2M structure-based STT-MRAM provides remarkably low standby power and operating power, and the TS-NMC scheme shows the preferable capability of reducing power by skipping memory write/read and calculation. The framework was evaluated with a custom MCU in a 28-nm HKMG CMOS process and the foundry announced a universal 2T-2M MRAM cell. The TS-NMC can successfully work at 0.6-V supply voltage under 20MHz. This Near-MRAM framework can offer up to $\sim$95.6% energy saving compared to commercial SRAM refer to ULPBenchmark. Minimally modified peripheral circuits were designed to realize TS-NMC with 0.73%/9.9% area overhead of sparse flag storage/NMC computing logic. Classification task on MNIST takes 13nJ/pattern. The energy access of memory, calculation, and the total can be reduced by 52.49×, 2.7×, and 11.3× respectively from the TS scheme.

VI. ACKNOWLEDGMENT

This work is funded by the National Key Research and Development Program of China under Grant 2018YFB2202800.

REFERENCES

[1] H.-S. P. Wong and S. Salahuddin, "Memory leads the way to better computing," *Nature Nanotechnology*, vol. 10, pp. 191–194, Mar 2015.
[2] X. Li, Y. Xu, L. Ren, W. Ge, J. Cai, X. Liu, and J. Yang, "29.8 115nA@3V ULPMark-CP Score 1205 SCVR-Less Dynamic Voltage-Stacking Scheme for IoT MCU," in *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 100–102.
[3] O. Mutlu, "Intelligent Architectures for Intelligent Computing Systems," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2021, pp. 318–323.
[4] D. Rossi, F. Conti, M. Eggiman, S. Mach, A. D. Mauro, M. Guermandi, G. Tagliavini, A. Pullini, I. Loi, J. Chen, E. Flamand, and L. Benini, "4.4 A 1.3TOPS/W @ 32GOPS Fully Integrated 10-Core SoC for IoT End-Nodes with 1.7μW Cognitive Wake-Up From MRAM-Based State-Retentive Sleep Mode," in *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 60–62.
[5] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A Survey of Quantization Methods for Efficient Neural Network Inference," *CoRR*, vol. abs/2103.13630, 2021. [Online]. Available: https://arxiv.org/abs/2103.13630
[6] G. Dai, T. Huang, Y. Wang, H. Yang, and J. Wawrzynek, "GraphSAR: A Sparsity-Aware Processing-in-Memory Architecture for Large-Scale Graph Processing on ReRAMs," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, ser. ASPDAC '19, New York, NY, USA, 2019, p. 120–126.
[7] A. Mohanty, X. Du, P. Chen, J. Seo, S. Yu, and Y. Cao, "Random Sparse Adaptation for Accurate Inference with Inaccurate Multi-level RRAM Arrays," in *2017 IEEE International Electron Devices Meeting (IEDM)*, Dec 2017, pp. 6.3.1–6.3.4.
[8] C. Deng, S. Liao, Y. Xie, K. K. Parhi, X. Qian, and B. Yuan, "PermDNN: Efficient Compressed DNN Architecture with Permuted Diagonal Matrices," in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Oct 2018, pp. 189–202.
[9] T.-C. Chang, Y.-C. Chiu, C.-Y. Lee, J.-M. Hung, K.-T. Chang, C.-X. Xue, S.-Y. Wu, H.-Y. Kao, P. Chen, H.-Y. Huang, S.-H. Teng, and M.-F. Chang, "13.4 A 22nm 1Mb 1024b-Read and Near-Memory-Computing Dual-Mode STT-MRAM Macro with 42.6GB/s Read Bandwidth for Security-Aware Mobile Devices," in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2020, pp. 224–226.
[10] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.
[11] P. Wang, Y. Ji, C. Hong, Y. Lyu, D. Wang, and Y. Xie, "SNrram: An efficient sparse neural network computation architecture based on resistive random-access memory," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, June 2018, pp. 1–6.
[12] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally, "SCNN: An accelerator for compressed-sparse convolutional neural networks," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, 2017, pp. 27–40.
[13] "MTJ compact model," http://www.spinlib.com/, 2020, [Online; accessed 03-Feb-2020].
[14] S. Wang, H. Lee, F. Ebrahimi, P. K. Amiri, K. L. Wang, and P. Gupta, "Comparative Evaluation of Spin-Transfer-Torque and Magnetoelectric Random Access Memory," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 2, pp. 134–145, June 2016.
[15] https://www.eembc.org/, "Embedded Microprocessor Benchmark Consortium (EEMBC)," 2021.
[16] M. Bocquet, T. Hirztlin, J.-O. Klein, E. Nowak, E. Vianello, J.-M. Portal, and D. Querlioz, "In-memory and error-immune differential rram implementation of binarized deep neural networks," in *2018 IEEE International Electron Devices Meeting (IEDM)*, 2018, pp. 20.6.1–20.6.4.
[17] F. Merrikh-Bayat, X. Guo, M. Klachko, M. Prezioso, K. K. Likharev, and D. B. Strukov, "High-performance mixed-signal neurocomputing with nanoscale floating-gate memory cell arrays," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4782–4790, 2018.
[18] S. Angizi, Z. He, A. S. Rakin, and D. Fan, "Cmp-pim: An energy-efficient comparator-based processing-in-memory neural network accelerator," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, 2018, pp. 1–6.