SpinLiM: Spin Orbit Torque Memory for Ternary Neural Networks Based on the Logic-in-Memory Architecture

Lichuan Luo, He Zhang, Jinyu Bai, Youguang Zhang, Wang Kang and Weisheng Zhao
School of Integrated Circuit Science and Engineering, Beihang University, Beijing, 100191, China
Email: wang.kang@buaa.edu.cn

***Abstract*—Logic-in-memory architecture based on spintronic memories shows fascinating prospects in neural networks (NNs) for its high energy efficiency and good endurance. In this work, we leveraged two magnetic tunnel junctions (MTJs), which are driven by the interplay of field-free spin orbit torque (SOT) and spin transfer torque (STT) effects, to achieve a novel stateful logic-in-memory paradigm for ternary multiplication operations. Based on this paradigm, we further proposed a highly parallel array structure to serve for ternary neural networks (TNNs). Our results demonstrate the advantage of our design in power consumption compared with CPU, GPU and other state-of-the-art works.**

Keywords—Stateful logic-in-memory, magnetic tunnel junction, ternary neural network, spin orbit torque memory.

I. INTRODUCTION

Deep neural network (DNN) has shown impressive potential as a general solution in many applications, such as data analytics, image and speech recognition, etc. However, it needs intensive memory access operations and a huge amount of multiply and accumulate (MAC) operations, resulting in high requirement for power budget, computing resource as well as communication bandwidth. Currently, the logic-in-memory (LiM) architecture shows promising as an alternative solution to alleviate such dilemma. On the other hand, binary neural network (BNN) or ternary neural network (TNN), which quantizes the weights and (or) activation functions to binary or ternary values [1, 2], has been proposed to further reduce the storage and computing complexities. Therefore, BNN/TNN based on LiM architecture has drawn considerable interest in the community.

LiM is a promising solution to deal with the memory wall by moving the computing task into memory, significantly reducing power consumption and delay for data movement [3-5]. LiM based on nonvolatile memories is particularly preferable for IoT inference applications [6-10], owing to the ultralow static power and instant on/off property. Until now, LiM based on RRAM and PCM has been widely studied [11-13]. However, a large part of state-of-the-art designs rely on 'analog computing' paradigm, which requires analog-to-digital converters (ADCs) and digital-to-analog converters (DACs), bringing huge area and energy overheads. Regarding alternative 'digital computing' paradigm, it mainly consists of stateful logic and sensing based logic. Specifically, stateful logic utilizes one single memory cell to achieve Boolean logic operation by write operations, whose performance is decided by the writing features [14-18]. On the other hand, the sensing-based logic is realized by reading two or more cells simultaneously, and by judging the result through a special sensing amplifier (SA) [19-21]. The issues of this kind of logic paradigm include the complex design of the SA and the inefficient mapping of the algorithm in hardware. Meanwhile, proper compilation and optimization are strongly required, since the logic functions are limited and the data organizations are fixed in this logic paradigm. In addition, to our knowledge, when connecting to DNN, most studies of the 'digital computing' are related to BNN instead of TNN [22-23].

This work was supported by the Beijing Nova Program from Beijing Municipal Science and Technology Commission (Z201100006820042), Beijing Natural Science Foundation (Grants No. 4202043), and the National Natural Science Foundation of China (61871008).

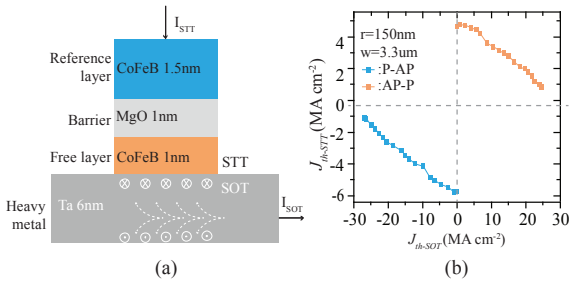
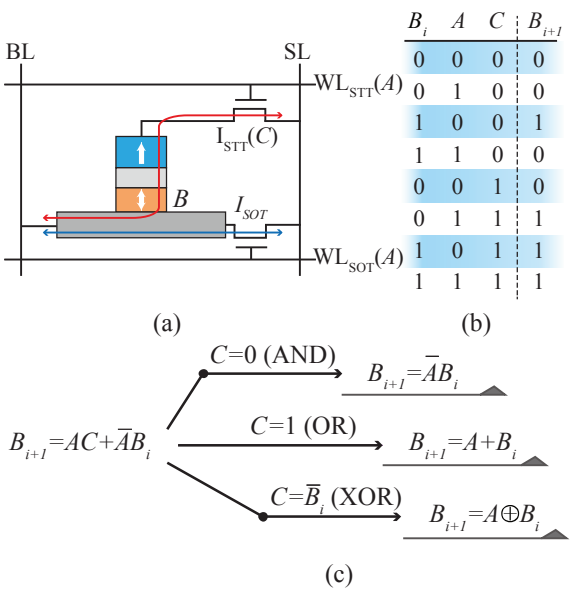


Fig. 1. (a) Structure of the STT-SOT p-MTJ device. The switching of the p-MTJ device is contributed by an in-plane SOT current flowing through the heavy metal and the STT current flowing through the p-MTJ. (b) The critical STT current (J_{th-STT}) measured as a function of the critical SOT current (J_{th-SOT}) by using a DC source [24].

In this paper, we propose a stateful LiM architecture based on spin orbit torque (SOT) memory for TNN. Our design can perform ternary multiplication operations within two memory cells by configuring the write operation. Please note that we choose perpendicular MTJ (p-MTJ) device [24], which is driven by the interplay of spin transfer torque (STT) and field free spin SOT effect, in our design in order to improve the computing efficiency. Our primary contributions are as following:

- We propose a novel stateful spintronic LiM (SpinLiM) for ternary multiplication operation by utilizing the memory like write operations with two p-MTJ cells.
- A highly parallel array structure and the related write driver and word line driver are proposed to ensure the efficient LiM implementation.
- We propose a mapping method and the corresponding architecture of TNN in our memory array to achieve efficient convolution and full connection calculation.

The remainder of this paper is organized as follows. Section II presents a brief introduction of the STT-SOT p-MTJ and the principle of performing ternary multiplication operation within two memory cells. In Section III, a highly parallel array structure is proposed to achieve logic operations in the memory array. Section IV shows the mapping method of TNN in the memory array and the corresponding architecture of the stateful LiM unit. Simulation results and discussions are shown in section V. In the end, Section VI concludes the paper.



B_i	A	C	B_{i+1}
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	0
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	1

Fig. 2. Logic paradigm based on the STT-SOT p-MTJ device. (a) Structure of the STT-SOT p-MTJ memory cell with two access transistors. I_{STT} and I_{SOT} together completes the write operation. Here, the bias voltage on the two access transistors, the initial content stored in the p-MTJ, and the write voltage across the BL and SL are defined as the logic input operands A, B, and C, respectively; (b) The truth table depending on the three input logic operands. Here B_{i+1} represents the logic output. (c) The principle of implementing reconfigurable logic functions. A logic expression ($B_{i+1} = AC + \bar{A}B_i$) can be obtained from the truth table. By assigning different C, we can get 'AND', 'OR', 'XOR' functions based on the STT-SOT p-MTJ device.

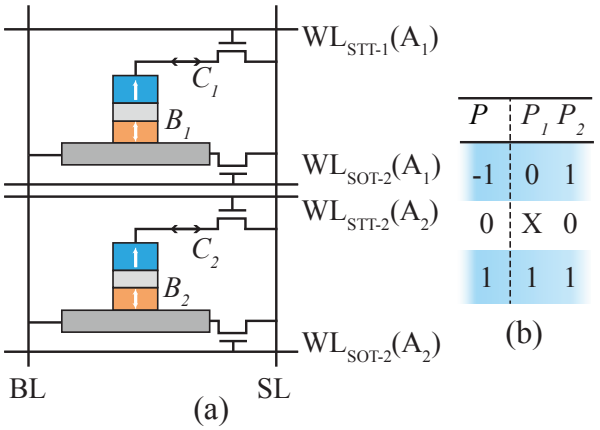
II. Ternary Multiplication Based on the STT-SOT p-MTJ Device

A. Introduction of STT-SOT p-MTJ device

Fig.1(a) shows the structure of the STT-SOT p-MTJ device, which is stacked with Ta (6.0)/ CoFeB (1.0)/ MgO (1.0)/ CoFeB (1.5)/ [Co/Pt] (5.0)/ Ru (0.85)/ [Co/Pt] (10.2)/ Ru (5.0) (the unit of layer thickness is in nm). The in-plane charge current in the bottom heavy metal is to generate SOT current (I_{SOT}), giving rise to the spin torque acting on the free layer of the p-MTJ. Another current flowing through the p-MTJ device is the STT current (I_{STT}), generating spin torque by the spin polarization of the reference layer of the p-MTJ. Both currents contribute to the magnetization switching of p-MTJ device. Fig. 1(b) shows the critical STT current (J_{th-STT}) measured as a function of critical SOT current (J_{th-SOT}) by using a DC source. The J_{th-SOT} can obviously reduce J_{th-STT}, therefore leading to a reduction in power consumption and an increase in switching speed [24].

B. Logic paradigm based on the STT-SOT p-MTJ device

As shown in Fig. 2(a), a STT-SOT p-MTJ memory cell consists of two access transistors with one p-MTJ device. The cell is connected with a bit-line (BL), a source-line (SL) and two word-lines (WLs). The read and write operations are achieved by supplying a voltage across the BL and SL, when the two access transistors are on according to the target data. For the logic operations, we define the bias voltages on the two access transistors as operand A (0V for '0' and 1.2V for '1'),



P	P_1	P_2
-1	0	1
0	X	0
1	1	1

Function1: $B_{1,i}=P_1; C_1=\bar{B}_i; A_1=\bar{Q}_1 \longrightarrow B_{1,i+1}=\overline{P_1 \oplus Q_1}$

Function2: $B_{2,i}=P_2; C_2=0; A_2=\bar{Q}_2 \longrightarrow B_{2,i+1}=P_2 Q_2$

P	Q	P_1	P_2	Q_1	Q_2	$B_{1,i}$	$B_{2,i}$	A_1	A_2	C_1	C_2	$B_{1,i+1}$	$B_{2,i+1}$	$P\times Q$
-1	-1	0	1	0	1	0	1	1	0	1	0	1	1	1
-1	0	0	1	X	0	0	1	X	1	1	0	X	0	0
-1	1	0	1	1	1	0	1	0	0	1	0	0	1	-1
0	-1	X	0	0	1	X	0	1	0	X	0	X	0	0
0	0	X	0	X	0	X	0	X	1	X	0	X	0	0
0	1	X	0	1	1	X	0	0	0	X	0	X	0	0
1	-1	1	1	0	1	1	1	1	0	0	0	0	1	-1
1	0	1	1	X	0	1	1	X	1	0	0	X	0	0
1	1	1	1	1	1	1	1	0	0	0	0	1	1	1

(c)

Fig. 3. Principle of performing ternary multiplication operation. (a) Structure of the two memory cells. The definition of the three operands are the same as Fig. 2. (b) Principle of splitting a ternary value into two-bit values. (c) Truth table of the ternary multiplication operation. Here, we set cell-1 to perform XOR operation ($C_1 = \bar{B}_i$) and cell-2 to perform AND operation ($C_2 = 0$).

the initial state of p-MTJ as operand B (low resistance for '0' and high resistance for '1') and the voltage across BL and SL as operand C (C=0/1 when tending to write '0/1'). The truth table of these three operands is obtained in Fig. 2(b), where the B_{i+1} is the next state of the p-MTJ. By analyzing the truth table, Eq. (1) describes the relationship between the three operands.

$$B_{i+1} = AC + \bar{A}B_i \tag{1}$$

As shown in Fig. 2(c), by configuring C as different values, the cell can perform XOR, AND and OR logic operations, respectively. Among them, XOR and AND logic functions are the two basic functions for our TNN.

C. Principle of ternary multiplication operation

In order to achieve ternary multiplication, two memory cells are needed, as shown in Fig. 3(a). The two cells are located in the same BL and SL, denoted as cell-1 and cell-2, respectively. The definition of the three operands in each cell are the same as shown in Fig. 2(a). For a ternary value P{-1, 0, 1}, Fig. 3(b) shows the principle of splitting it into two-bit value, P_1 and P_2,

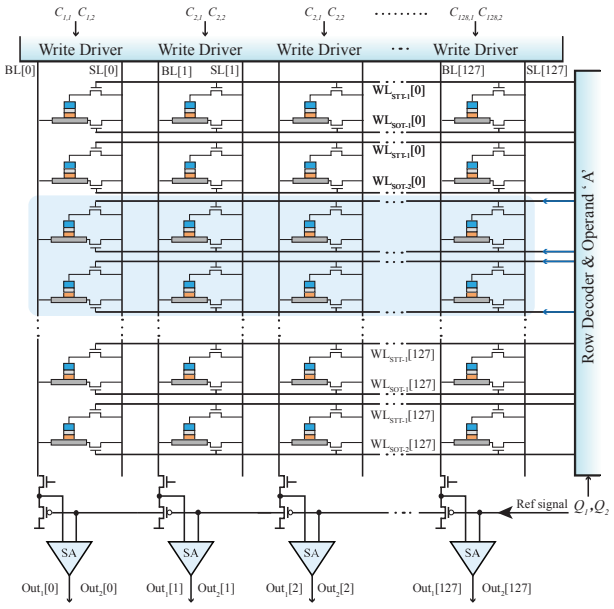


Fig. 4. Reconfigurable array structure between typical memory mode and LiM mode. Each column has a write driver and a sensing amplifier (SA) to achieve parallel memory write and read operations. Furthermore, the WL driver and write driver are modified to meet the stateful LiM requirement. The input operands '*A*' is connected through the row decoder.

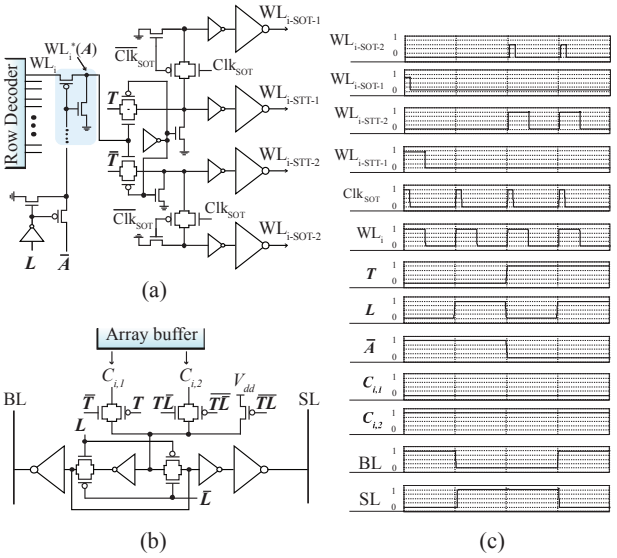


Fig. 5. (a) Modified WL driver combined with the last level decoder, where signal Q is the logic operand, signal T is the last level address to choose between B_1 and B_2, Clk_{SOT} is the duration to supply I_{SOT}, and signal L sets the array to work in typical memory mode or SpinLiM mode. (b) Modified write driver, which can perform different logic functions (XNOR and AND) based on the paradigm shown in Fig. 3. (c) Waveform of an entire multiplication operation.

where X is 0 or 1. P_2 represents that the P is '0' or not. If P_2 =0, it means P=0, no matter what P_1 is. When $P_2 = 1$, P_1 represents that it is '-1' or '1' with '0' and '1', respectively. Following this principle, for any ternary value P and Q, and their two-bit form (P_1, P_2) and (Q_1, Q_2), the multiplication result is exactly the XNOR logic operation between P_1 and Q_1 as well as the AND operation between P_2 and Q_2, i.e., $(P_1 \odot Q_1, P_2 Q_2)$. Fig. 3(c) shows the truth table and the flow to achieve multiplication operation. For cell-1, we set $B_{1,i} = P_1$; $C_1 = \overline{B_{1,i}}$; $A = \overline{Q_1}$, thereby, it performs a XNOR operation, i.e., $B_{1,i+1} = P_1 \odot Q_1$. For cell-2, we set $B_{2,i} = P_2$; $C_1 = 0$; $A = \overline{Q_1}$, and it obtains $B_{2,i+1} = P_1 Q_2$. As shown in the truth table, all the 9 cases of the multiplication achieve the right results, proving the correctness of the proposed logic paradigm.

III. Parallel Array Structure for Stateful LiM

A. Parallel array structure

Fig. 4 shows the parallel structure of the proposed memory array, which is the basic computing unit in SpinLiM. Here we take a 130*256 memory array as an example to illustrate the principle. Except for two column of reference cells, the array contains 128*128 computing cells, with 128 BLs, SLs and 128 groups of WLs, where each group consists of 4 WLs. There are 130 independent write drivers, which can realize parallel writing operations. Each column is also equipped with a current mean pre-charge SA to support the parallel access. The signals $C_{1\sim128}$ are the data intended to be written into the memories (P), and also the logic operands when performing logic operations. The other operand (Q) is supplied to the row decoder to control the WL. The memory array can work either in a typical memory mode or in stateful LiM mode, which can be switched between each other freely by configuring the write and read circuits.

B. WL driver and write driver

To perform stateful LiM, the write driver and WL driver require some modifications. For the WL driver, as shown in Fig. 5(a), it is combined with the last-level row decoder, T is the last level address to choose between B_1 and B_2. Signal Q is the logic operand, Clk_{SOT} is the duration to supply I_{SOT}, and signal L is to set the array to work in either a typical memory mode or a SpinLiM mode. When L=0, the AND gate outputs a low voltage, therefore the PMOS transistor is on so as to transfer the signal from the original WL, working in a typical memory mode. When L=1, we have $WL^*_i = \overline{Q}$, and it is transferred to the 4 WLs, working in the SpinLiM mode.

The write driver is also controlled by the signal L and signal T. When L=0, the write driver works in the typical memory mode, and intends to write $C_{i,1}$ or $C_{i,2}$ into the memory cell, decided by the signal T. When L=1, it works in the SpinLiM mode. If T=0, the first cell requires to perform XNOR logic operation, and the write driver writes $\overline{C_{i,1}}$ into the memory cell. Alternatively, if T=1, the second cell requires to perform AND logic operation, and the write driver directly writes '0'.

Fig. 5(c) shows the waveform of the two drivers in the entire multiplication operation. Firstly, the first cell is chosen by the signal T, and P_1 is written into the cell in the memory mode. Then, the first cell works in the SpinLiM mode, and Q_1 is supplied to the WL driver. An XNOR operation is performed when the write driver intends to write $\overline{P_1}$. Afterwards, the signal T chooses the second cell and P_2 is written into the cell in the

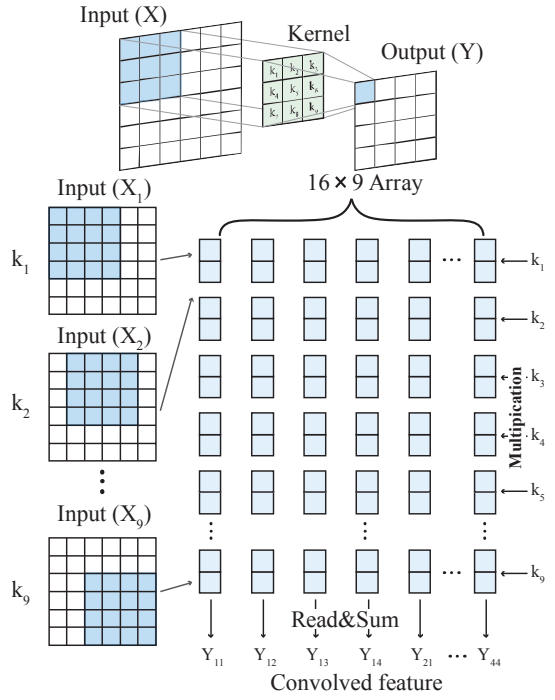


Fig. 6. The mapping principle of convolution operations in TNN. Take a 6×6 input matrix together with a 3×3 kernel as an example, the input matrix generates 9 new matrixes according to the kernel, which are loaded into the array in 9 rows. Then, each row performs a multiplication with the related kernel. Finally, the convolved features are the sum of each column.

memory mode. Finally, the second cell performs an AND operation when the write driver writes '0' and Q_2 controls the WL driver in the SpinLiM mode. As can be seen, the entire multiplication operation is achieved by 4 continuous write operations, and the memory array can freely switch between the memory mode and LiM mode. It should be noted that most part of the two drivers in Fig. 5 are also required for a typical memory and the extra circuits for the SpinLiM mode is the part marked in blue shadows in these drivers.

IV. TNN Mapping in Array and Architecture

A. TNN mapping method

Before implementing TNN in our proposed SpinLiM, one important key is the mapping method in order to efficiently realize convolution and full connection calculation. For the convolution computing, as shown in Fig. 6, take the first layer of a 6×6×D input matrix together with a 3×3×D kernel as an example to show the mapping principle. We can find that the 6×6 input matrix is firstly treated as 9 new matrixes according to the 9 kernel values. For example, the first matrix is the value performing multiplication with k_1. Then, load these 9 matrixes into 9 rows in the memory array and perform multiplication operations with $k_{1\sim9}$ respectively. Afterward, a 16×9 new matrix is generated, and each column contains 9 two-bit values which are the results of the multiplication. Repeating this operation D-1 times in the other rows. In the end, by reading out the values in each row, we can obtain the accumulation results, which are the outputs of the convolution feature maps.

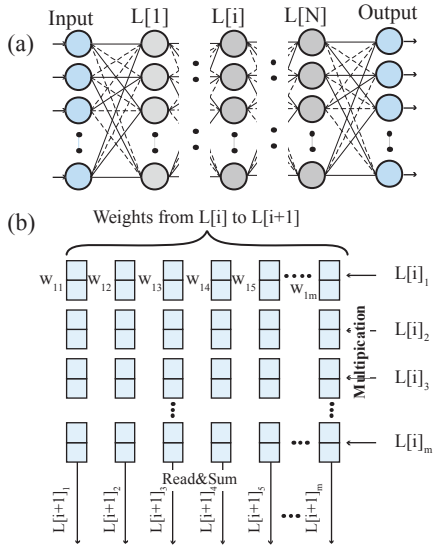


Fig. 7. (a) Structure of the fully connected layers in TNN. (b) The mapping method for the fully connected layers. The weights between layer L[i] and L[i+1] are loaded into the array according to the layout on the diagram, when the multiplication is performed by supplying the inputs to the WL drivers. Then, the sum of each column represents the outputs.

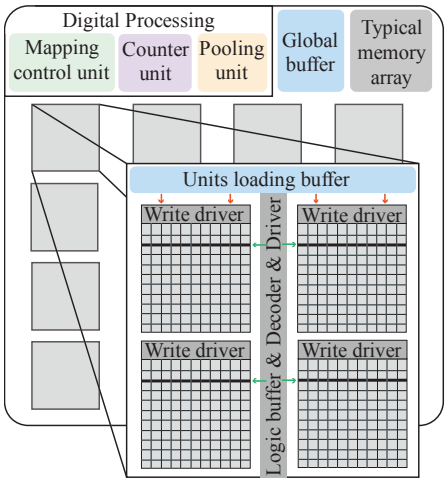


Fig. 8. Architecture of the proposed SpinLiM. The digital processing part consists of the mapping control, pooling and counter units. The memory array stores the weights of TNN. The computing unit (SpinLiM) performs both convolution and full connection operations.

The fully connected layer can also be realized in a similar way in our SpinLiM. Fig.7 (a) shows the structure of a fully connected layer in TNN. The output of each cell is the accumulation of the multiplication result between weights and all inputs from the last layer. Fig.7 (b) shows the mapping approach for the fully connected layer. The weights between L[i] and L[i+1] can be seen as the operand P in Fig. 3, while the output of L[i] is the operand Q. Each row contains the weights from one cell in L[i] to all cells in L[i+1]. The loading of weights and multiplication with inputs are performed at the same time. Then a ternary multiplication can be obtained by leveraging counters to get accumulation of each column.

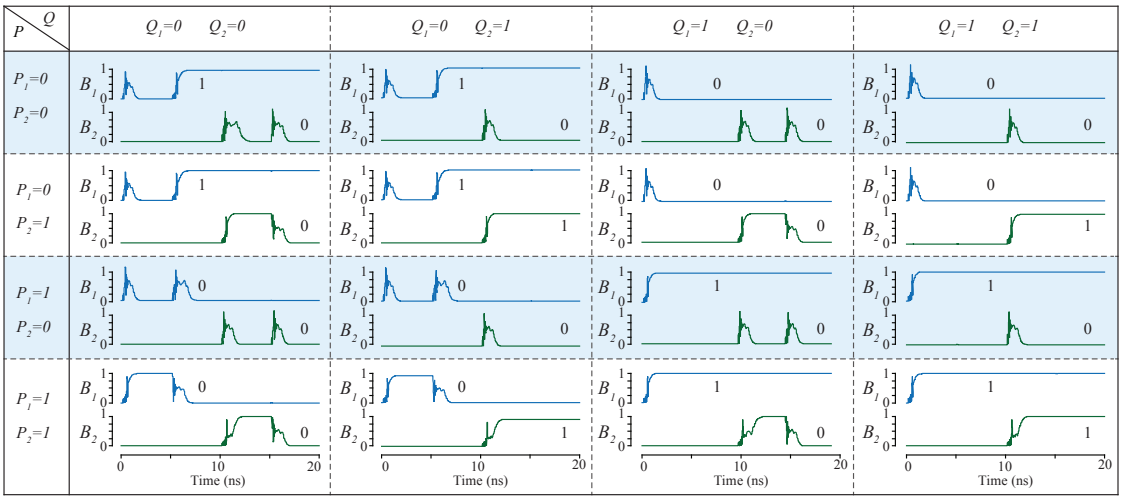


Fig. 9. Transient simulation waveforms of the proposed ternary multiplication based on two STT-SOT p-MTJ cells with hybrid MTJ/CMOS circuit simulation. All 16 cases of (P_1, P_2) and (Q_1, Q_2) are shown with correct results.

TABLE I. KEY PARAMETERS IN OUR SIMULATION

Parameters	Default Value
MTJ size (W×L)	40×40 nm
Heavy metal size (W×L× t)	40× 80 ×4 nm
TMR ratio at zero bias	120%
Supply voltage	1.2 V
Temperature	300 K
CMOS technology	40 nm
I_{SOT} average value	145.0 μA
I_{STT} average value	55.5μA

TABLE II. PERFORMANCE OF OUR PROPOSED SPINLIM

	Value
Average power consumption to perform ternary multiplication	~2.02 pJ
Computing delay for (parallel) ternary multiplication	~28.8 ns
Area overhead of the memory array for LiM	~0.8%

B. *Overall architecture*

To efficiently perform TNN in our SpinLiM, Fig. 8 shows the overall architecture. The digital processing part contains a mapping control unit, a pooling unit and a counter unit (details are omitted here owing to space limitation). The typical memory array stores the weights of TNN. The computing unit (SpinLiM) performs both convolution and full connection computing operations depending on the configuration profile.

V. SIMULATION RESULTS AND DISCUSSIONS

The circuit level simulations were performed in Cadence and Virtuoso platform by utilizing a 40nm CMOS design kit and a physically-verified STT-SOT p-MTJ device model[14]. Hybrid CMOS/MTJ circuits were designed and simulated to evaluate the functionality and performance of the proposed SpinLiM paradigm. Some key device parameters are shown in Table I. The memory level simulations were performed by a modified

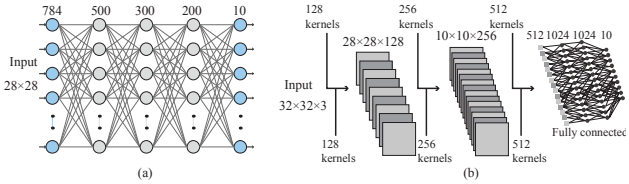


Fig. 10. The TNN Structures of (a) MMINST and (b) CIFAR 10.

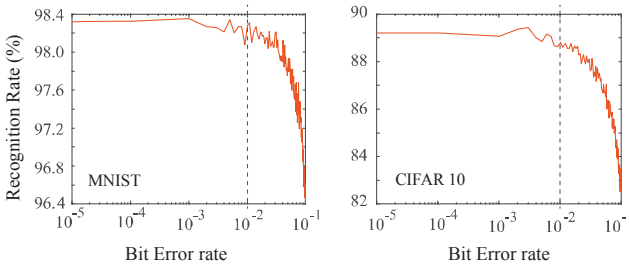


Fig. 11. Recognition rate of (a) MINST, and (b) CIFAR 10, as a function of bit error rate (BER) of the p-MTJ.

TABLE III. PERFORMANCE OF PERFOMING ONE RECOGNITION

Work		Power consumption
CPU [1]	MNIST	100.7mJ
	CIFAR10	646mJ
GPU [1]	MNIST	15.2mJ
	CIFAR10	156.3mJ
[2] FPGA	CIFIA 10	27.9mJ
[8] DASM	CIFIA 10	4.3mJ
Our work	MNIST	3.1μJ
	CIFAR10	740.1μJ

NVSim simulator [26], which contains the parameters of latency and power consumption. We modified the parameters of the bit cell, the array structure and the peripheral circuits in NVSim based on Fig. 4. Fig. 9 shows the transient simulation waveforms

of the proposed ternary multiplication. The blue line and green line are the resistance states of the two p-MTJs when performing 4 continuous write operations. As can be seen, all 16 cases of (P_1, P_2) and (Q_1, Q_2) were performed successfully, based on the values shown in Fig. 3(c). Furthermore, the performance of the proposed memory array is shown in Table II. The average power consumption and delay for parallel ternary multiplication are ~2.02 pJ and ~28.8 ns respectively. It should be noted that the area overhead of SpinLiM is only 0.8 %compared with a typical memory array. To evaluate the performance, we simulated two TNN applications, including the MINST and CIFAR10, whose structure are shown in Fig. 10. We simulated the recognition rate firstly as a function of bit error rate (BER, caused by stochastic switching and process variations) of the p-MTJ, as shown in Fig. 11. The recognition rate nearly unchanged when BER is below 10^{-2}. According to our circuit level Monte Carlo simulations, the BER is far less than 10^{-2}. Therefore, the reliability will not be a big problem for our design. Table III shows the performance to achieve one recognition compared with CPU, GPU and other related works. As can be seen, our proposed SpinLiM has a big advantage in terms of energy consumption.

VI. Conclusion

We exploit a spintronic memory based on the STT-SOT p-MTJ device, which has low writing power and high switching speed, to achieve a stateful LiM paradigm (SpinLiM) for TNN. A highly parallel array structure and the related write driver and WL driver are designed to ensure efficient LiM implementation. We also propose a novel mapping method for SpinLiIM and evaluate the performance in TNN applications. Our cross-layer simulation results show that our proposed SpinLiM has a big improvement in power consumption in comparison with CPU, GPU and other state-of-the-art works.

References

[1] S. Liang, S. Yin, L. Liu, W. Luk and S. Wei, "FP-BNN: Binarized neural network on FPGA," *Neurocomputing*, vol. *275*, pp. 1072-1086, Jan. 2018.

[2] R. Zhao, W. Song, W. Zhang, T. Xing, J. H. Lin, M. Srivastava, Z. Zhang, "Accelerating binarized convolutional neural networks with software-programmable FPGAs," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 15–24, 2017.

[3] M. Cofano, M. Vacca, G. Santoro, G. Causapruno, G. Turvani and M. Graziano, M, "Exploiting the Logic-In-Memory paradigm for speeding-up data-intensive algorithms," *Integration*, vol. *66*, pp. 153-163, 2019.

[4] R. Ben-Hur, R. Ronen, A. Haj-Ali, D. Bhattacharjee, A. Eliahu, A, N. Peled and S. Kvatinsky, "SIMPLER MAGIC: Synthesis and Mapping of In-Memory Logic Executed in a Single Row to Improve Throughput," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Jul. 2019*.

[5] G. Santoro, G. Turvani and M. Graziano, "New Logic-In-Memory Paradigms: an architectural and technological perspective," Micromachines, vol. 10, no. 6, pp. 368, May. 2019.

[6] 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 *ACM Proceedings of the 53rd Annual Design Automation Conference,* p. 173, 2016.

[7] E. Linn, R. Rosezin, S. Tappertzhofen, U. Böttger, and R. Waser, "Beyond von Neumann—logic operations in passive crossbar arrays alongside memory operations," *Nanotechnology*, vol. 23, no. 30, pp. 30520, Jul. 2012.

[8] L. Chang, X. Ma, Z. Wang, Y. Zhang, Y. Ding, W. Zhao and Y. Xie, "DASM: Data-Streaming-Based Computing in Nonvolatile Memory Architecture for Embedded System," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 27, no. 9, pp. 2046-2059, Set. 2019.

[9] D. Ielmini and H. S. P. Wong, "In-memory computing with resistive switching devices," *Nature Electronics*, vol. 1, no. 6, pp. 333-343, Jun. 2018.

[10] A. Sebastian, M. Le Gallo, G. W. Burr, S. Kim, M. BrightSky and E. Eleftheriou, "Tutorial: Brain-inspired computing using phase-change memory devices," *Journal of Applied Physics*, vol. 124, no. 11, pp. 111101, Sep. 2018.

[11] W. H. Chen, K. X. Li, W. Y. Lin, et. al, "A 65nm 1Mb nonvolatile computing-in-memory ReRAM macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors," in *IEEE International Solid-State Circuits Conference-(ISSCC),* pp. 494-496, 2018.

[12] K. Qiu, W. Chen, Y. Xu, L. Xia, Y. Wang and Z. Shao, "A peripheral circuit reuse structure integrated with a retimed data flow for low power RRAM crossbar-based CNN," in *Design, Automation & Test in Europe Conference & Exhibition (DATE),* pp. 1057-1062, 2018.

[13] M. Le Gallo, A. Sebastian, G. Cherubini, H. Giefers and E. Eleftheriou, "Compressed sensing with approximate message passing using in-memory computing," *IEEE Transactions on Electron Devices*, vol. *65,* no, 10, pp. 4304-4312, Oct. 2018.

[14] H. Zhang, W. Kang, L. Wang, K. L. Wang, and W. Zhao, "Stateful reconfigurable logic via a single-voltage-gated spin Hall-effect driven magnetic tunnel junction in a spintronic memory," *IEEE Transactions on Electron Devices*, vol. *64,* no. 10, pp. 4295-4301, Oct. 2017.

[15] H. Zhang, W. Kang, B. Wu, P. Ouyang, E. Deng, Y. Zhang and W. Zhao, "Spintronic Processing Unit Within Voltage-Gated Spin Hall Effect MRAMs," *IEEE Transactions on Nanotechnology*, vol. *18*, pp. 473-483, May. 2019.

[16] S. Liang, S. Yin, L. Liu, W. Luk and S. Wei, "FP-BNN: Binarized neural network on FPGA," *Neurocomputing*, vol. *275*, pp. 1072-1086, Jan. 2018.

[17] W. Shen, P. Huang, M. Fan, R. Han, Z. Zhou, B. Gao and X. Zhang, "Stateful Logic Operations in One-Transistor-One-Resistor Resistive Random Access Memory Array," *IEEE Electron Device Letters*, vol. *40, no.* 9, pp. 1538-1541, Sep. 2019.

[18] Z. R. Wang, Y. T. Su, Y. Li, Y. X. Zhou, T. J. Chu, K. C. Chang, T. C. Chang, S. M. Sze, and X. S. Miao, "Functionally complete Boolean logic in 1T1R resistive random access memory," *IEEE Electron Device Letters*, vol. 38, no. 2, pp. 179-182, Dec. 2017..

[19] S. Hamdioui, H. A. Du Nguyen, M. Taouil, A. Sebastian, M. Le Gallo, S. Pande and G. Karunaratne, "Applications of computation-in-memory architectures based on memristive devices," in *IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE),* pp. 486-491, 2019.

[20] L. Xu, R. Yuan, Z. Zhu, K. Liu, Z. Jing, Y. Cai and R. Huang, "Memristor‐Based Efficient In‐Memory Logic for Cryptologic and Arithmetic Applications," *Advanced Materials Technologies*, pp. 1900212, Jun. 2019.

[21] H. Li, B. Gao, Z. Chen, Y. Zhao, P. Huang, H. Ye, L. Liu, X. Liu, and J. Kang, "A learnable parallel processing architecture towards unity of memory and computing," *Scientific reports*, vol. 5, pp. 1-8, Aug. 2015,

[22] L. Chang, X. Ma, Z. Wang, Y. Zhang, Y. Xie and W. Zhao, "PXNOR-BNN: In/With Spin-Orbit Torque MRAM Preset-xnor Operation-Based Binary Neural Networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* Jul. 2019.

[23] 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 *IEEE International Electron Devices Meeting (IEDM),* pp. 20-6, 2018.

[24] M. Wang, W. Cai, D. Zhu, Z. Wang, J. Kan, Z. Zhao and C. Park, "Field-free switching of a perpendicular magnetic tunnel junction through the interplay of spin–orbit and spin-transfer torques," *Nature electronics*, vol. *1,* no. 11, pp. 582, Nov. 2018.

[25] W. Kang, Y. Ran, Y. Zhang, W. Lv and W. Zhao, "Modeling and exploration of the voltage-controlled magnetic anisotropy effect for the next-generation low-power and high-speed MRAM applications," *IEEE Transactions on Nanotechnology*, vol. 16, no. 3, pp. 387-395, May. 2017.

[26] X. Dong, C. Xu, Y. Xie and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 994-1007, Jul. 2012.