A high-reliability and low-power computing-in-memory implementation within STT-MRAM

Liuyang Zhang, Erya Deng, Hao Cai, You Wang, Lionel Torres, Aida Todri-Sanial, Youguang Zhang

▶ **To cite this version:**

Liuyang Zhang, Erya Deng, Hao Cai, You Wang, Lionel Torres, et al.. A high-reliability and low-power computing-in-memory implementation within STT-MRAM. Microelectronics Journal, 2018, 81, pp.69-75. 10.1016/j.mejo.2018.09.005 . lirmm-01880058

HAL Id: lirmm-01880058

https://hal-lirmm.ccsd.cnrs.fr/lirmm-01880058

Submitted on 8 Feb 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ACCEPTED MANUSCRIPT

A High-Reliability and Low-Power Computing-in-Memory Implementation Within STT-MRAM

Liuyang Zhang[a], Wang Kang[a,*], Erya Deng[a], Hao Cai[b], You Wang[a], Youguang Zhang[a], Lionel Torres[c], Aida Todri-Sanial[c], Weisheng Zhao[a,*]

[a] *Fert Beijing Institute and School of Electronic and Information Engineering, Beihang University, Beijing, 100191, China*
[b] *Département Communications et Electronique, Télécom-ParisTech, Université Paris-Saclay, Paris, 75013, France.*
[c] *CNRS-LIRMM/University of Montpellier, Montpellier, 34095, France*

Abstract

In the conventional Von-Neumann computer architecture, more energy and time are consumed by the data transport, rather than the computation itself because of the limited bandwidth between the processor and memory. Computing-in-memory (CIM) is therefore proposed to effectively address the issue by moving some specified kinds of computation into the memory. It has been proposed for several decades, however, not really used when considering the reliability and cost. With the emergence of non-volatile memories, the CIM has regained interest to tackle the issue. In this paper, we implement a CIM scheme: ComRef (Complementary Reference) within STT-MRAM (Spin Transfer Torque Magnetic Random-Access Memory), and then compare its reliability and performance with the DualRef (Dual Reference) CIM implementation. Simulation results reveal that the ComRef obviously improves the reliability by decreasing 67.1% of the operation error rate and by increasing up to 57.4% of the sensing margin. It accelerates the bitwise logic operation with cutting down 20.8% (∼41.1 *p*s) of the operation delay. Most importantly, it is highly energy efficient by reducing 23.4% of the average dynamic energy and 65.6% of the average static power. The ComRef provides a pathway to implement high-reliability and low-power CIM paradigm within STT-MRAM.

Keywords: Computing-in-memory (CIM), Nonvolatile memory (NVM), STT-MRAM, Memory wall

1. Introduction

In the current big data era, a huge amount of data is generated, stored and processed all the time. Data are transferred back and forth to be processed between the processor and the memory in the conventional Von-Neumann computer architecture [1]. The processor frequency and the memory access efficiency, however, are in the state of imbalance, especially in the computer system with multicore processors. More energy is consumed during data transport rather than the computation itself. Transferring and processing of data are highly inefficient due to the limited bandwidth between the processor and the memory, which are even worse in the data-intensive applications [2, 3]. According to the measurement in [4], about 200 times more energy is dissipated to access DRAM one time when compared with a floating-point computation. In summary, the limited bandwidth between the processor and the memory has been one of the most critical bottlenecks to improve the system performance. This is the well-known processor-memory gap or "memory wall" [5].

To address the issue, the closer integration of logic and memory has been explored widely, such as logic-in-memory, processing-in-memory, in-memory processing. These explorations can be classified as two types: 1) near memory computing, 2) executing some specified kinds of computations within memory, it is called computing-in-memory (CIM). The CIM can reduce the amount of data transferring between the processors and the memory, to mitigate the stress of data transport on the limited bandwidth [6, 7]. However, the CIM fails to tackle the issue due to some practical considerations. In

*Corresponding author
Email addresses: wang.kang@buaa.edu.cn (Wang Kang), weisheng.zhao@buaa.edu.cn (Weisheng Zhao)

ACCEPTED MANUSCRIPT

CMOS based memories, it is complex and cost inefficient to integrate processing unit and storage unit together until the 3D stacking technology emerges [8]. Moreover, these technologies also suffer from the reliability issues [9, 10].

With the emerging of non-volatile memories (NVMs), the CIM has regained the interests from both academia and industry. The "memory wall" issue becomes possible to be addressed due to some good properties of the NVMs over the other conventional memories [11–15]. For example, STT-MRAM has been considered as one of the most promising candidates due to its distinctive advantages, such as, non-volatility, good scalability, compatibility with CMOS, ultra-fast accessing speed and good endurance[16–20]. Some CIM paradigms are proposed to implement logic operations within the NVMs. These paradigms can be categorized as two kinds. One utilizes two or more reference cells to distinguish the different resistance states of the data cells participating in the logic operations, while the other one executes the logic operations by its complementary structure [5, 21, 22]. Most of the CIM paradigms have been presented and assessed at architecture level [6, 23–25]. They are based on the idea of adding necessary peripheral circuits to the memory, which makes the memory has some kinds of computation capability and storage capability at the same time [26–28]. However, we note that few detailed implementations at the circuit level are introduced among these CIM paradigms, and few studies are evaluated carefully on their reliability and performance [29–32]. Therefore, we implement a complementary CIM scheme: ComRef at the circuit level within STT-MRAM, and then measure its reliability and performance. The proposal is based on the idea that logic operations can be executed within STT-MRAM by enabling multiple word lines (WL) simultaneously, and then the result can be obtained by memory-like read operation. This CIM implementation can perform logic operation in higher reliability, lower power consumption and higher access speed compared with the dual reference (DualRef) CIM implementation [33].

The rest of this paper is organized as follows. Section 2 presents the ComRef implementation. The functionality of the ComRef CIM implementation is validated in Section 3. After that, the reliability and performance assessment are included in Section 4. At last, Section 5 concludes this paper.

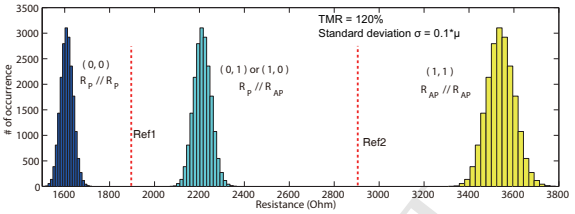

Figure 1: Resistance distribution of two MTJs aligned in low resistance parallel state ($R_P//R_P$), antiparallel state ($R_P//R_{AP}$) and high resistance parallel state ($R_{AP}//R_{AP}$). The MTJ model used comes from [34].

Table 1: Truth Table for ComRef CIM Implementation

Logic	$(M_0,M_1)^1$	(M00,M10)[2]	(M01,M11)[3]	Ouput
AND	$0(R_P, R_{AP})$	$0(R_P, R_{AP})$	$0(R_P, R_{AP})$	0
	$0(R_P, R_{AP})$	$0(R_P, R_{AP})$	$1(R_{AP}, R_P)$	0
	$0(R_P, R_{AP})$	$1(R_{AP}, R_P)$	$0(R_P, R_{AP})$	0
	$0(R_P, R_{AP})$	$1(R_{AP}, R_P)$	$1(R_{AP}, R_P)$	1
OR	$1(R_{AP}, R_P)$	$0(R_P, R_{AP})$	$0(R_P, R_{AP})$	0
	$1(R_{AP}, R_P)$	$0(R_P, R_{AP})$	$1(R_{AP}, R_P)$	1
	$1(R_{AP}, R_P)$	$1(R_{AP}, R_P)$	$0(R_P, R_{AP})$	1
	$1(R_{AP}, R_P)$	$1(R_{AP}, R_P)$	$1(R_{AP}, R_P)$	1

[1] (M_0,M_1) represents $(MTJ_{OpSel0},MTJ_{OpSel1})$
[2] (M00,M10) represents (MTJ00,MTJ10)
[3] (M01,M11) represents (MTJ01,MTJ11)

2. ComRef CIM Implementation

The ComRef CIM implementation is implemented within STT-MRAM by adding necessary peripheral circuits. It can work in both computing mode and memory mode to perform bitwise logic operations or write/read data, and switch between the two modes freely. We will show the working procedures of the ComRef by the case of a four-by-four ComRef CIM array.

2.1. Basic Principles

There are two differences of the ComRef when compared with other CIM paradigms. One is that each bit is expressed by two complementary data cells. Generally, the low-resistance state R_L of the data cell is used to indicate the data "0", and its high-resistance state R_H represents data "1". In the ComRef CIM implementation, the data "0" is expressed by the resistance pattern (R_L, R_H), while the resistance pattern (R_H, R_L) indicates the data "1". The other difference is that the kind of the bitwise logic operation to be performed, for example, OR or AND, is decided by the two operation-selected data cells, which are also complementary. The resistance pattern (R_L, R_H) of the two operation-selected data cells imply an AND

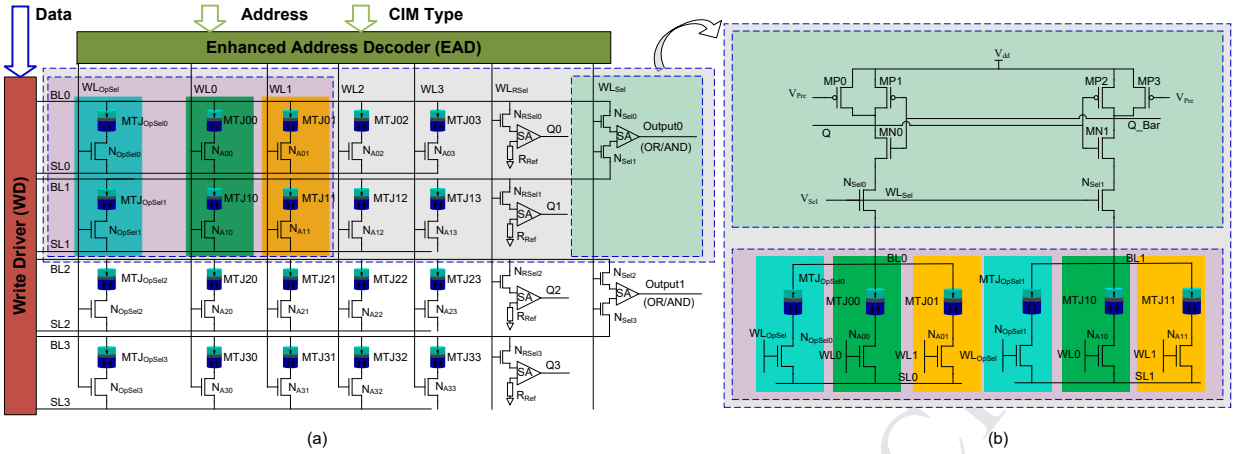

Figure 2: (a) A four-by-four ComRef CIM array. Data is stored in the complementary way. For example, data "0" is stored in MTJ03 (R_P) and its complementary MTJ13 (R_{AP}). Two complementary data cells with the green and yellow background are the two operands respectively, and the data cells with the cyan background determine which kind of bitwise logic operation to be executed. The part with the light green background is the sensing amplifier (SA) for the bitwise logic operations. With the signal of *Address* and *CIM Type*, the EAD can exactly activate one WL to perform normal memory read and write operations or activate three WLs to execute bitwise logic operations. (b) Schematic of the ComRef CIM implementation for one-bit bitwise logic operation. The final bitwise logic operation results are obtained at terminal Q or Q_{Bar}.

bitwise logic operation to be performed, and the execution of an OR bitwise logic operation is determined by the resistance pattern (R_H, R_L), as shown in TABLE 1. In our proposal, bitwise logic operations are performed between the two columns activated simultaneously in a ComRef CIM array. As depicted in Fig. 1, there are three resistance states of the two MTJs in parallel. The key idea of performing bitwise logic operations within STT-MRAM is to differentiate one resistance state from the others. In the conventional CIM paradigms implemented by dual reference cells, the resistance of two reference cells is designed to distinguish the three resistance states as shown in Fig. 1. In the ComRef CIM implementation, the principle we exploited is that the distance of the two resistance states is expanded by adding the operation-selected bit cells in parallel. This idea results in higher reliability, which can be validated by the following simulations in Section 4.

2.2. ComRef CIM Array

Fig. 2(a) shows a four-by-four ComRef CIM array. When compared with the normal four-by-four STT-MRAM array, one column of data cells is added as the operation-selected data cells, the address decoders are replaced by the EAD (Enhanced Address Decoder), and two sensing amplifiers (SA) are added to perform bitwise logic operations. The EAD is designed to be able to activate one WL in read/write operations (memory mode) and activate three WLs simultaneously in the bitwise logic operations (computing mode) according to the signals of *Address* and *CIM Type*. Two complementary data cells are used to indicate one-bit data in the ComRef CIM array. Therefore, the WD (Write Driver) is also enhanced, where the input *Data* are first converted to the complementary form and then wrote into the corresponding data cells. The readout operation in this CIM array is the same as normal STT-MRAM, but the readout results can be more reliable due to the complementary structure. When performing a bitwise logic operation between the second and the third column as shown in Fig. 2(a), the corresponding WL0 and WL1 should be selected and activated. Besides the WL0 and WL1, $\mathrm{WL_{OpSel}}$ should also be activated to determine the kind of operation. When $\mathrm{WL_{Sel}}$ is enabled, the execution of the operation starts. As can be seen, the ComRef CIM implementation is realized by revising the peripheral circuits and adding necessary SAs, which makes it be able to work in between the computing and memory mode, and switch freely.

2.3. Working Procedure

In this subsection, OR bitwise logic operation of one-bit is chosen as an example to present the working procedures. The components participat-

ACCEPTED MANUSCRIPT

ing in the operation are abstracted as shown in Fig. 2(b). The data cells with the cyan background are the two operation-selected data cells, which should be set to data "1". In other words, (MTJ_{OpSel0}, MTJ_{OpSel1}) are configured to (R_{AP}, R_P). Assumed that the OR bitwise logic operation is executed between data "0" and "1". Therefore, the MTJs (MTJ00, MTJ10) in the data cells with the green background are configured to (R_P, R_{AP}), and the MTJs (MTJ01, MTJ11) in the data cells with the yellow background are configured to (R_{AP}, R_P). The inverter (MP1,MN0) and the other one (MP2,MN1), along with the PMOS MP0 and MP3 comprise the precharge sensing amplifier (PCSA), which is used to distinguish the resistance differences of the two discharge branches. In the beginning, V_{Pre}, V_{Sel}, WL_{OpSel}, WL0 and WL1 are all in low-voltage level, the PCSA is precharged. When starting to execute the OR bitwise logic operation, WL_{OpSel}, WL0 and WL1 are activated first by setting to high-voltage level, then V_{Pre} and V_{Sel} are switched from low to high-voltage level. Now, the PCSA begins to discharge. The discharge currents in the two branches are different due to the different resistance of data cells. In this operation, the resistance of the data cells in the left branch is bigger than that of the right branch, so the discharge currents in the right branch are more than that of the other branch. Therefore, the PCSA reaches the stable state (Q="1", Q_{Bar}="0") when the voltage on drain electrode of MN1 is first lower than the threshold voltage of the inverter (MP2,MN1). The final operation result is obtained at Q, which is consistent with TABLE 1.

In summary, we present the basic principles of the ComRef CIM implementation, and then show the CIM array and the work procedures. The reliability and performance of the ComRef CIM implementations are fluctuated with PVT (Process, Voltage, Temperature) variations. In the following, we will first validate the feasibility of the ComRef CIM implementation, and then measure the operation error rate, sensing margin, operation delay and energy consumption to assess their reliability and performance.

3. Functionality Verification

For the purpose of verifying the functionality of the ComRef CIM implementation within STT-MRAM, a hybrid MTJ/CMOS transient simulation is carried out with a 45 nm PTM CMOS model and a 40 nm compact perpendicular magnetic anisotropy MTJ model [35–37]. The supply voltage is fixed at $V_{dd} = 1.1V$, and the TMR equals to $TMR = 300\%$, other related parameters are their default values in the MTJ model. Larger CMOS transistors channel width in the sensing circuit is used to enlarge the sensing margin, and the minimum channel width is employed in the write circuit and the logic circuit to eliminate the influence of the parasitic capacitor as possible.

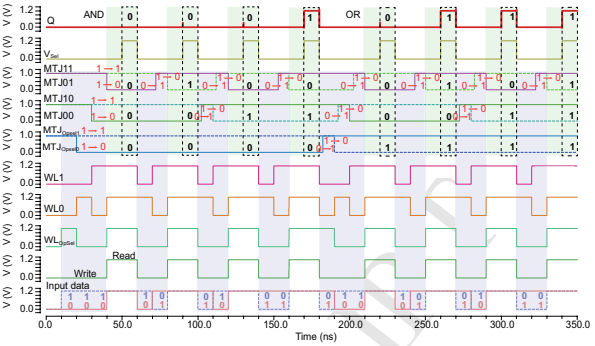

Figure 3: Transient simulation waveforms of the ComRef. The waveforms with the light purple background shows the switching of MTJs participating in the bitwise logic operation. The state of MTJ_{OpSel0} and MTJ_{OpSel1} are complementary, they decide OR or AND bitwise logic operation to be performed. In this simulation, the AND bitwise operation is first to be carried out, and then the OR bitwise logic operation. The operation results are obtained in Q and shown in the waveforms with the light green background.

Fig. 3 depicts the transient simulation waveforms of the ComRef CIM implementation. AND bitwise logic operations are first carried out, and then the OR bitwise logic operations. Although XOR/XNOR and other bitwise logic operation can not be executed directly within the ComRef CIM array, the results can be obtained by adding essential logic circuits. Therefore, only the waveforms of OR/AND bitwise logic operations are presented in this simulation. Assumed that all the MTJs participated in this simulation are in the high-resistance state (antiparallel state). The first step in this simulation is to write data "0" to the three complementary data cells. (MTJ_{OpSel0}, MTJ_{OpSel1}), (MTJ00, MTJ10) and (MTJ01, MTJ11) are all configured to the resistance pattern (R_P, R_{AP}), as shown in Fig. 3 with the red arrows. After that, WL_{OpSel}, WL0 and WL1 are enabled, the operation result is obtained at Q by the memory-like sensing/read operation when V_{Sel} are set to "1", which is shown in the light green background. Other bitwise logic op-

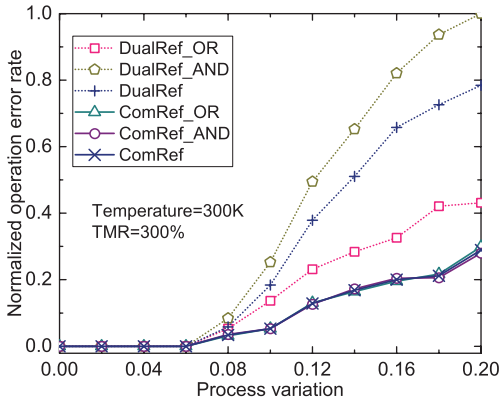

Figure 4: Normalized operation error rates of the ComRef and DualRef CIM implementations with respect to the process variations of CMOS transistor. The TMR and temperature are fixed at 300%, 300K respectively.

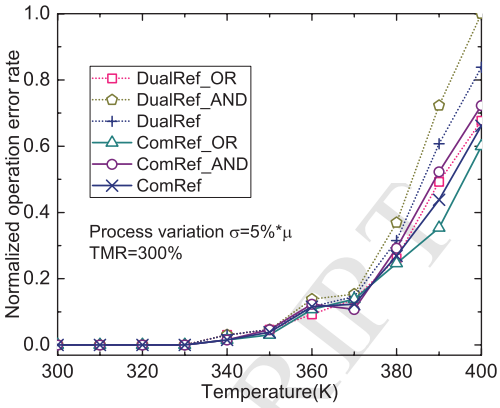

Figure 5: Normalized operation error rates of the ComRef and DualRef CIM implementations with respect to the temperature. The process variation of the CMOS transistors is 5% of the mean μ.

erations are executed in the following stages. The output results are consistent with TABLE 1.

The ComRef CIM implementation is validated with the hybrid MTJ/CMOS transient simulation. To the best of our knowledge, STT-MRAM suffers from many reliability issues, which are caused by the PVT variations. In follows, we will measure the operation error rate, sensing margin, operation delay, energy dissipation to explore the design space of the CIM implementation. Last, we will compare the ComRef CIM implementation with the DualRef CIM implementation to see their robustness and performance under PVT fluctuations.

4. Reliability and Performance Measurement

In this section, we carry out six groups of simulations to assess the ComRef CIM implementation in terms of their operation error rate, sensing margin, operation delay, dynamic energy consumption, static power dissipation and area overheads. And we also compare it with the DualRef CIM implementation to see their reliability and performance.

4.1. Operation Error Rate

Operation error rates are generally used to indicate the reliability of the two CIM implementations directly. In this subsection, we evaluate the operation error rates of the ComRef and DualRef CIM implementations under the process variations and temperature variations respectively. All these simulations are carried out with only considering the variations of the CMOS transistors. The operation error rate is defined as the arithmetic mean of error rates at the bit patterns: (00), (01), (10) and (11) for OR or AND bitwise logic operations. These operation error rates are calculated with respect to the process variations varying from 0 to 20% of the mean μ and the temperature varying from 300K to 400K. In these simulations, the TMR of the MTJ devices are set to 300%, the value is in accordance with the reported recently [38]. The temperature is fixed at 300K when obtained operation error rates with the process variations; while evaluating the operation error rate with respect to the temperature, the process variation is set to 5%. The measured results are shown in Fig. 4 and 5 respectively. As can be seen from the two figures, the operation error rates keep very low at the small process variations, and it is the same in the low temperatures. However, the operation error rates raise up rapidly after 6% of process variations and 330K. With the large process variations or higher temperatures, both the resistances of the data cells and the reference cells change, the driving currents of the transistors change accordingly. These changes result in the reduction of the sensing margin, which can be the explanation for more and more operation errors occurring. The operation error rate of the ComRef CIM implementation is lower than that of the DualRef as shown in Fig. 4 and 5 under both the process and temperature variations, and which is more obvious under the process variations. Under the same process variations, the ComRef CIM implementation can reduce the average operations

ACCEPTED MANUSCRIPT

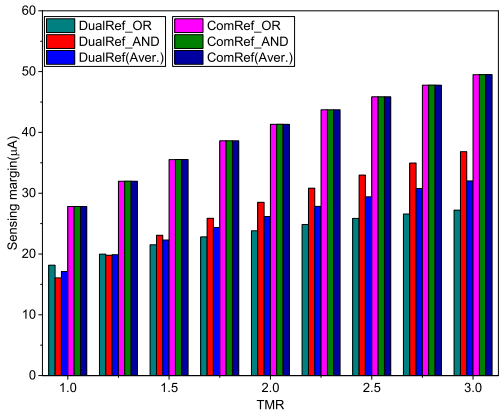

Figure 6: The sensing margins of bitwise logic operations. It is defined as the currents differences of the two discharge branches in the SAs.

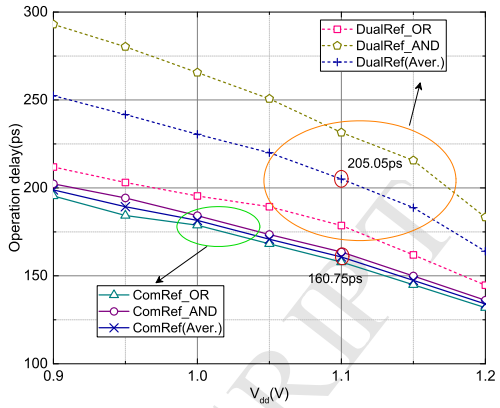

Figure 7: The operation delay of the ComRef and DualRef CIM implementations. The simulations are carried out with the fixed temperature $300K$ and no process variation.

error rate by 67.1% as shown in Fig. 4 when compared with the DualRef CIM implementation.

4.2. Sensing Margin

As described in Section 2, these bitwise logic operations are executed by SAs. The sensing margin of the bitwise logic operations performed within the ComRef CIM implementation are defined as the current differences of the two discharge branches of the SAs. The sensing margins indicate the variations that the ComRef CIM implementation can tolerate to ensure the computing without errors. The differences between the low-resistance and high-resistance states of the MTJ device varies with the TMR. Therefore, the sensing margins are checked by increasing TMR from 100% to 300%. The calculated sensing margins are shown in Fig. 6. Every data is the average value of the sensing margins at the four bit patterns. As can be seen from this figure, the smallest sensing margin is more than 16μA, and the biggest one is about 50μA. Both the sensing margin of the OR and AND bitwise logic operations arise linearly with the TMR increasing. We can also see from the figure, the sensing margin of the ComRef is bigger than that of the DualRef from the TMR=100% to 300%. The ComRef CIM implementation improves about 57.4% of the sensing margins when compared with that of the DualRef CIM implementation. The resistance difference between the data cell and the low or high reference cell becomes large as the TMR arising, which results in the increasing of the sensing margins.

4.3. Operation Delay

The operation delay represents the time consumed by a bitwise logic operation. The average delays of the ComRef and DualRef CIM implementations are measured from the rising edge of the enable signal to the time when the output results reach a stable state. The operation delay is calculated with respect to the supply voltage V_{dd} by varying from 0.9V to 1.2V, and the TMR and temperature is set to 300% and $300K$ respectively. The results are shown in Fig. 7. It can be seen from the figure that the operation delay of both the ComRef and DualRef CIM implementations decline with increasing the supply voltage V_{dd}. The delay time is dominated by the charge and discharge time of the capacitors in the circuit. A larger supply voltage can reduce the time. In $V_{dd} = 1.1V$, the average delay of the ComRef CIM implementation is $160.75ps$, which is less than $205.05ps$ of that of the DualRef. The average operation delays of the ComRef is reduced by about $41.1ps$ (20.8%) when compared with that of the DualRef CIM implementation. In summary, it is faster to perform bitwise logic operations in the ComRef than in the DualRef.

4.4. Dynamic Energy Consumption

The operation dynamic energy consumption is calculated by integrating the product of the voltage and the currents with respect to its operation delay, both the dynamic energies of the two discharge branches in the SAs are counted. The results are shown in Fig. 8. These results are obtained by fixing the temperature at $300K$. As can be seen from the figure, the dynamic energy consumptions

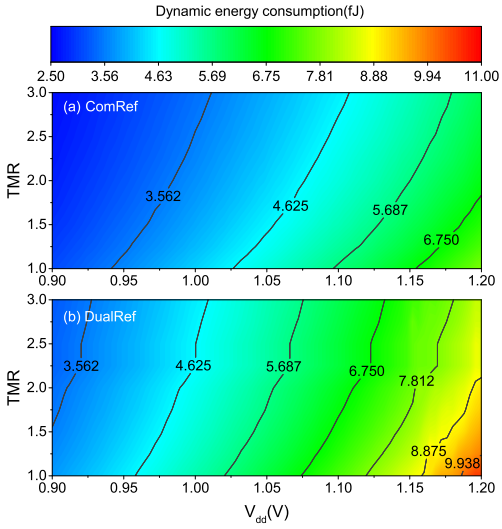

Figure 8: The dynamic energy consumption of the ComRef and DualRef CIM implementations. The temperature is fixed at $300K$.

of both the ComRef and DualRef CIM implementations increase with the supply voltage V_{dd} arising, but decrease with the raising of the TMR. Increasing the V_{dd} will surely result in the raising of the dynamic energy consumption. The resistance of the discharge branches arises with the TMR increasing, which results in the reduction of the discharge currents, so the dynamic energy consumed by single bitwise logic operation decreases with the TMR rising. It is also found that the ComRef CIM implementation reduces the average dynamic energy consumptions by 23.4% when compared with the DualRef CIM implementation.

4.5. Static Power Dissipation

The static power dissipation is calculated by the product of the supply voltage V_{dd} and the leakage currents of the CMOS transistors, which is shown in Fig. 9. We measure the static power dissipation of both the ComRef and DualRef CIM implementations with respect to V_{dd} from 0.9V to 1.2V. The static power dissipation of the ComRef CIM implementation increases from 1.7 nW to 8.83 nW with increasing V_{dd}. However, the static power dissipation of the DualRef CIM implementation is 8.99 nW at $V_{dd} = 0.9V$, and 25.7 nW at $V_{dd} = 1.2V$. Both of them are more than that of the ComRef CIM implementation respectively. The ComRef CIM implementation reduces more than 65.6% of the static power dissipation when compared with the DualRef CIM implementation.

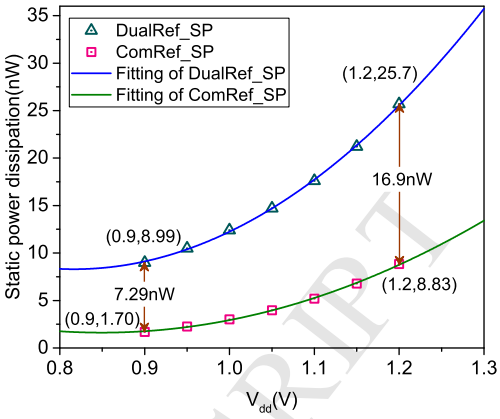

Figure 9: The static power dissipation of the ComRef and DualRef CIM implementations. The temperature is fixed at 300K. The fitting curves are obtained with quadratic polynomial.

4.6. Area Overheads

When compared with the normal STT-MRAM array, an SA is added for every two rows in one array, and the address decoder and write driver are enhanced to support CIM operations. Due to the complementary structure, two data cells are used to represent one bit, there will be more area overheads when realizing the ComRef CIM implementation. Regarding the DualRef CIM implementation, two SAs and the related reference cells, an XOR gate and a multiplexor are needed for every row in one array. In summary, both of the ComRef and DualRef CIM implementations have area overheads, but the area overhead of the ComRef is more than that of the DualRef.

In summary, the reliability and performance of the ComRef and DualRef CIM implementations are analyzed quantitatively. Both the ComRef and DualRef CIM implementations suffer from the PVT variations. However, as compared with the DualRef CIM implementation, we find that the ComRef CIM implementation has less operation error rate, faster operation speed and less energy consumption than the DualRef CIM implementation besides for area overheads.

5. Conclusion

A CIM implementation: ComRef is implemented here within STT-MRAM. The ComRef can be used to perform bitwise logic operation and store data. The functionality, reliability and performance of the ComRef are evaluated at 45nm technology node.

Simulations reveal that the ComRef can work correctly when the temperature is less than 330K and process variations of the CMOS transistor do not surpass 6%. Most importantly, the ComRef CIM implementation has higher reliability and lower energy dissipation, which make the ComRef stand out from others.

References

[1] M. Imani, S. Gupta, T. Rosing, Ultra-efficient processing in-memory for data intensive applications, in: Proceedings of the 54th Annual Design Automation Conference 2017, 2017, pp. 6:1–6:6.

[2] J. Zhou, X. Yang, J. Wu, X. Zhu, X. Fang, D. Huang, A memristor-based architecture combining memory and image processing, Sci. China Inform. Sci. 57 (5) (2014) 1–12.

[3] N. S. Kim, T. Austin, D. Blaauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, V. Narayanan, Leakage current: Moore's law meets static power, Computer 36 (12) (2003) 68–75.

[4] S. W. Keckler, W. J. Dally, B. Khailany, M. Garland, D. Glasco, GPUs and the future of parallel computing, IEEE Micro 31 (5) (2011) 7–17.

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

[6] L. Koskinen, J. Tissari, J. Teittinen, E. Lehtonen, M. Laiho, J. H. Poikonen, A performance case-study on memristive computing-in-memory versus Von Neumann architecture, in: 2016 Data Compression Conference, 2016, pp. 613–613.

[7] H. Li, T. F. Wu, S. Mitra, H. S. P. Wong, Resistive RAM-centric computing: Design and modeling methodology, IEEE Transactions on Circuits and Systems I: Regular Papers 64 (9) (2017) 2263–2273.

[8] J. Ahn, S. Hong, S. Yoo, O. Mutlu, K. Choi, A scalable processing-in-memory accelerator for parallel graph processing, in: 2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture, 2015, pp. 105–117.

[9] K. Chen, S. Li, N. Muralimanohar, J. H. Ahn, J. B. Brockman, N. P. Jouppi, CACTI-3DD: Architecture-level modeling for 3D die-stacked DRAM main memory, in: 2012 Design, Automation Test in Europe Conference Exhibition, 2012, pp. 33–38.

[10] X. Dong, X. Wu, G. Sun, Y. Xie, H. Li, Y. Chen, Circuit and microarchitecture evaluation of 3D stacking magnetic RAM (MRAM) as a universal memory replacement, in: 2008 45th ACM/IEEE Design Automation Conference, 2008, pp. 554–559.

[11] K. W. Kwon, X. Fong, P. Wijesinghe, P. Panda, K. Roy, High-density and robust STT-MRAM array through device/circuit/architecture interactions, IEEE Trans. Nanotechnol. 14 (6) (2015) 1024–1034.

[12] W. Zhao, G. Prenat, Spintronics-Based Computing, Springer, Switzerland, 2015.

[13] X. Fong, Y. Kim, K. Yogendra, D. Fan, A. Sengupta, A. Raghunathan, K. Roy, Spin-transfer torque devices for logic and memory: Prospects and perspectives, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 35 (1) (2016) 1–22.

[14] C. J. Xue, G. Sun, Y. Zhang, J. J. Yang, Y. Chen, H. Li, Emerging non-volatile memories: opportunities and challenges, in: 2011 Proceedings of the Ninth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, 2011, pp. 325–334.

[15] F. Parveen, S. Angizi, Z. He, D. Fan, Low power in-memory computing based on dual-mode SOT-MRAM, in: 2017 IEEE/ACM International Symposium on Low Power Electronics and Design, 2017, pp. 1–6.

[16] L. Zhang, Y. Cheng, W. Kang, L. Torres, Y. Zhang, A. Todri-Sanial, W. Zhao, Addressing the thermal issues of STT-MRAM from compact modeling to design techniques, IEEE Trans. Nanotechnol. 17 (2) (2018) 345–352.

[17] W. Kang, Y. Zhang, Z. Wang, J.-O. Klein, C. Chappert, D. Ravelosona, G. Wang, Y. Zhang, W. Zhao, Spintronics: Emerging ultra-low-power circuits and systems beyond MOS technology, ACM J. Emerg. Technol. Comput. Syst. 12 (2) (2015) 16:1–16:42.

[18] H. S. P. Wong, S. Salahuddin, Memory leads the way to better computing, Nat. Nanotechnol. 10 (3) (2015) 191–194.

[19] W. Kang, L. Zhang, J.-O. Klein, Y. Zhang, D. Ravelosona, W. Zhao, Reconfigurable codesign of STT-MRAM under process variations in deeply scaled technology, IEEE Trans. Electron Devices 62 (6) (2015) 1769–1777.

[20] S. Peng, W. Zhao, J. Qiao, L. Su, J. Zhou, H. Yang, Q. Zhang, Y. Zhang, C. Grezes, P. K. Amiri, K. L. Wang, Giant interfacial perpendicular magnetic anisotropy in mgo/cofe/capping layer structures, Appl. Phys. Lett. 110 (7) (2017) 072403.

[21] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, Y. Xie, PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory, in: 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture, 2016, pp. 27–39.

[22] W. Kang, H. Wang, Z. Wang, Y. Zhang, W. Zhao, In-memory processing paradigm for bitwise logic operations in STT-MRAM, IEEE Trans. Magn. 53 (11) (2017) 1–4.

[23] J. Yu, R. Nane, A. Haron, S. Hamdioui, H. Corporaal, K. Bertels, Skeleton-based design and simulation flow for computation-in-memory architectures, in: 2016 IEEE/ACM International Symposium on Nanoscale Architectures, 2016, pp. 165–170.

[24] A. Haron, J. Yu, R. Nane, M. Taouil, S. Hamdioui, K. Bertels, Parallel matrix multiplication on memristor-based computation-in-memory architecture, in: 2016 International Conference on High Performance Computing Simulation, 2016, pp. 759–766.

[25] S. Hamdioui, M. Taouil, H. A. D. Nguyen, A. Haron, L. Xie, K. Bertels, CIM100x: Computation in-memory architecture based on resistive devices, in: 2016 15th International Workshop on Cellular Nanoscale Networks and their Applications, 2016, pp. 1–2.

[26] D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R. Thomas, K. Yelick, Intelligent RAM (IRAM): chips that remember and compute, in: 1997 IEEE International Solids-State Circuits Conference. Digest of Technical Papers, 1997, pp. 224–225.

ACCEPTED MANUSCRIPT

[27] M. Imani, Y. Kim, T. Rosing, MPIM: Multi-purpose in-memory processing using configurable resistive memory, in: 2017 22nd Asia and South Pacific Design Automation Conference, 2017, pp. 757–763.

[28] D. Fan, S. Angizi, Z. He, In-memory computing with spintronic devices, in: 2017 IEEE Computer Society Annual Symposium on VLSI, 2017, pp. 683–688.

[29] S. Hamdioui, L. Xie, H. A. D. Nguyen, M. Taouil, K. Bertels, H. Corporaal, H. Jiao, F. Catthoor, D. Wouters, L. Eike, J. van Lunteren, Memristor based computation-in-memory architecture for data-intensive applications, in: Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, 2015, pp. 1718–1725.

[30] S. Hamdioui, M. Taouil, H. A. D. Nguyen, A. Haron, L. Xie, K. Bertels, Memristor: the enabler of computation-in-memory architecture for big-data, in: 2015 International Conference on Memristive Systems, 2015, pp. 1–3.

[31] S. Hamdioui, Computation in memory for data-intensive applications: Beyond CMOS and beyond Von-Neumann, in: Proceedings of the 18th International Workshop on Software and Compilers for Embedded Systems, 2015, pp. 1–1.

[32] J. J. Yang, D. B. Strukov, D. R. Stewart, Memristive devices for computing, Nat. Nanotechnol. 8 (1) (2013) 13–24.

[33] L. Zhang, W. Kang, H. Cai, P. Ouyang, L. Torres, Y. Zhang, A. Todri-Sanial, W. Zhao, A robust dual reference computing-in-memory implementation and design space exploration within STT-MRAM, accepted.

[34] L. Zhang, W. Kang, Y. Zhang, Y. Cheng, L. Zeng, J.-O. Klein, W. Zhao, Channel modeling and reliability enhancement design techniques for STT-MRAM, in: 2015 IEEE Computer Society Annual Symposium on VLSI, 2015, pp. 461–466.

[35] W. Zhao, Y. Cao, Predictive technology model for nano-cmos design exploration, J. Emerg. Technol. Comput. Syst. 3 (1).

[36] Y. Wang, Y. Zhang, E. Deng, J.-O. Klein, L. A. D. B. Naviner, W. Zhao, Compact model of magnetic tunnel junction with stochastic spin transfer torque switching for reliability analyses, Microelectron. Reliab. 54 (9) (2014) 1774 – 1778.

[37] L. Zhang, A. Todri-Sanial, W. Kang, Y. Zhang, L. Torres, Y. Cheng, W. Zhao, Quantitative evaluation of reliability and performance for STT-MRAM, in: 2016 IEEE International Symposium on Circuits and Systems, 2016, pp. 1150–1153.

[38] M. Wang, W. Cai, K. Cao, J. Zhou, J. Wrona, S. Peng, H. Yang, J. Wei, W. Kang, Y. Zhang, J. Langer, B. Ocker, A. Fert, W. Zhao, Current-induced magnetization switching in atom-thick tungsten engineered perpendicular magnetic tunnel junctions with large tunnel magnetoresistance, Nat. Commun. 9 (1) (2018) 671.

ACCEPTED MANUSCRIPT