Exploring STT-MRAM based In-Memory Computing Paradigm with Application of Image Edge Extraction

Zhezhi He*, Shaahin Angizi†, Deliang Fan‡
*†‡Department of Electrical and Computer Engineering, University of Central Florida, Orlando, 32816 USA
{*Elliot.He, †Angizi}@Knights.ucf.edu, ‡ dfan@ucf.edu

***Abstract*—In this paper, we propose a novel Spin-Transfer Torque Magnetic Random-Access Memory (STT-MRAM) array design that could simultaneously work as non-volatile memory and implement a reconfigure in-memory logic operation without add-on logic circuits to the memory chip. The computed output could be simply read out like a typical MRAM bit-cell through the modified peripheral circuit. Such intrinsic in-memory computation can be used to process data locally and transfers the "cooked" data to the primary processing unit (i.e. CPU or GPU) for complex computation with high precision requirement. It greatly reduces power-hungry and long distance data communication, and further leads to extreme parallelism within memory. In this work, we further propose an in-memory edge extraction algorithm as a case study to demonstrate the efficiency of in-memory preprocessing methodology. The simulation results show that our edge extraction method reduces data communication as much as 8x for grayscale image, thus greatly reducing system energy consumption. Meanwhile, the F-measure result shows only ∼10% degradation compared to conventional edge detection operator, such as Prewitt, Sobel and Roberts.**

***Index Terms*—In-memory computing, STT-MRAM, image processing, edge detection**

I. Introduction

The era of big data reveals the emerging needs to rethink and redesign the current computing architecture that can support memory-oriented processing for large datasets at exascale (10^{18} bytes/s or flops) [1]. Especially, the ability of conventional computing platforms to address these needs is beginning to stall fundamentally due to the exiting bottlenecks either in architecture (i.e. memory wall [2]) or device technology (power wall [2]). In the existing von Neumann computing architecture, the separation of memory and computing units interconnected via buses has faced serious challenges, such as long memory access latency, significant congestion at I/Os, limited memory bandwidth, and huge leakage power consumption in big data-driven applications [3]. To mitigate these challenges, in-memory computing architecture and non-volatile memory technology have been proposed to integrate memory and logic, leading to a more energy efficient information processing platform. However, such in-memory processing platforms typically impose additional area overhead to the memory chip owning to add-on logic elements, and thus sacrificing valuable memory capacity [3].

This material is based upon work supported in part by the National Science Foundation under Grant No.1740126.

From the device technology perspective, there are many recent researches carried out about emerging Non-Volatile Memories (NVM) that are promising to design such in-memory computing concept, for example, Resistive Random-Access Memory (RRAM) [3], Phase Change Memory (PCM) [4], Spin-Transfer Torque Magnetic Random-Access Memory (STT-MRAM) [5]. With great development of fabrication technology and commercialization progress, STT-MRAM has emerged as a leading non-volatile memory candidate owning to its unique features [5], [6], such as non-volatility, zero standby leakage, high write/read speed, compatibility with CMOS fabrication process, scalability, excellent endurance and high integration density. They are among the most prominent features which could pave a novel way to realize area and energy-efficient system supporting in-memory computing, normally-off computing, instant-on computing [5]–[8].

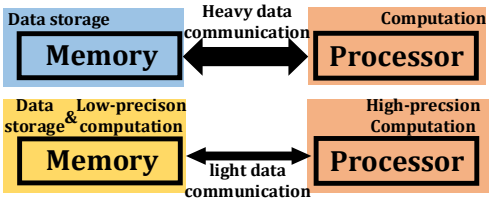


Fig. 1: In-memory computing to reduce the data communication and increase the computation parallelism.

For big data processing, one potential solution is extracting the abstract of data, and utilizing such extracted data for further processing. As depicted in Fig. 1, from in-memory computing architecture perspectives, memory takes care of data feature extraction with bit-wise low precision computation in extreme parallel approach, while the primary processing unit handles high precision operation with limited I/O bandwidth. In this work, edge extraction is used as a case study to demonstrate the efficiency improvement with in-memory computing paradigm. Edge detection is one of the fundamental operations in computer vision and image processing, which drastically reduces the complexity of original image while extracts the crucial boundary information for further processing procedures [9]. Currently, most edge detection algorithms can be classified into two categories [10]: search-based (e.g. Sobel [11], Prewitt [12], Roberts [13]) or zero-crossing based (e.g. Canny [9],

1063-6404/17 $31.00 © 2017 IEEE
DOI 10.1109/ICCD.2017.78

Marr-Hildreth [14]). The search-based algorithms are based on looking for the local maximum of the first-order derivatives, with a predefined threshold to decide the edge pixels. Meanwhile, the zero-crossing based algorithms identify the edge through seeking the zero crossing points in the second-order derivatives of the input image. However, such derivatives operation is computation expensive, which urges the discovery of other energy efficient edge extraction method.

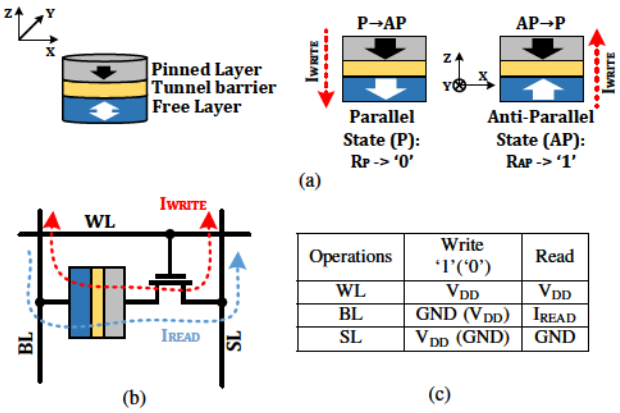
In this paper, we propose a novel STT-MRAM array that could work as conventional memory and in-memory computing kernel, which is inspired by the related works in [15]–[17]. For the additional in-memory computing function, no memory capacity is sacrificed, while any memory bit-cells allocated in the same row/column can be efficiently leveraged to realize bit-wise AND/OR/XOR and other linear/nonlinear operations with larger fan-ins. Moreover, we propose a local edge extraction method for digital image (i.e. binary and grayscale image), utilizing the introduced in-memory computing platform. Thus, only image feature is extracted and transferred to the primary processing unit (i.e. CPU or GPU) for further processing.

II. Spin-Transfer Torque Random-Access Memory

A typical Magnetic Tunnel Junction (MTJ) structure, as shown in Fig.2a, consists of two ferromagnetic layers with a tunnel barrier sandwiched between them. Due to the Tunnel MagnetoResistance (TMR) effect [18]–[20], the resistance of MTJ is high (low) when the magnetization of two ferromagnetic layers are in anti-parallel (parallel) state. The *TMR ratio* is defined as (R_{AP}-R_P)/R_P, which may vary from 10% to 400% depending on materials and temperature [18]–[21]. Thus, the data are stored as the magnetization direction in the free layer, which could be programmed through current induced Spin-Transfer Torque (STT). Note that, the MTJ with Perpendicular Magnetic Anisotropy (PMA) is used in this work. The 1T1R bit-cell is widely used in the typical STT-MRAM design, as depicted in Fig. 2b, which is correspondingly controlled by Bit Line (BL), Word Line (WL) and Source Line (SL). The biasing conditions of memory read and write are presented in Fig. 2c. For both memory read and write operation, the WL is enabled, which turns on the access transistor. Then, a voltage drop -V_{DD} or +V_{DD} is applied across the BL and SL, in order to realize write '1' or '0' respectively. For memory read, a sensing current (I_{READ}) is applied on the BL and consequently generates a sensing voltage, which can be detected by sense amplifier.

We jointly use the Non-Equilibrium Green's Function (NEGF) and Landau-Lifshitz-Gilbert (LLG) equation to model STT-MRAM bitcell (i.e. MTJ) for circuit level simulation. The magnetization dynamics of FL (m) can be modeled by LLG equation with spin-transfer torque terms, which can be mathematically described as [22], [23]:

$$\frac{dm}{dt} = -|\gamma| m \times H_{\text{eff}} + \alpha\left(m \times \frac{dm}{dt}\right) + |\gamma|\beta(m \times m_p \times m) - |\gamma|\beta\epsilon'(m \times m_p) \quad (1)$$



Operations	Write '1'('0')	Read
WL	V_{DD}	V_{DD}
BL	GND (V_{DD})	I_{READ}
SL	V_{DD} (GND)	GND

(c)

Fig. 2: (a) Device structure of conventional magnetic tunnel junction in parallel- and anti-parallel states, with current-induced spin-transfer torque switching scheme. (b) Bit-cell schematic of 1T1R STT-MRAM. (c) Biasing conditions for STT-MRAM operations.

$$\beta = |\frac{\hbar}{2\mu_0 e}|\frac{I_c P}{A_{\text{MTJ}} t_{\text{FL}} M_s} \quad (2)$$

where $\hbar$ is the reduced plank constant, γ is the gyromagnetic ratio, I_c is the charge current flowing through MTJ, t_{FL} is the thickness of free layer, ϵ' is the second Spin transfer torque coefficient, and H_{eff} is the effective magnetic field, P is the effective polarization factor, A_{MTJ} is the cross sectional area of MTJ, m_p is the unit polarization direction. The Fig. 3a has shown the normalized magnetization dynamics of free layer in x-, y- and z-axis, when performing the STT-MRAM write scheme as described in Fig. 2c.

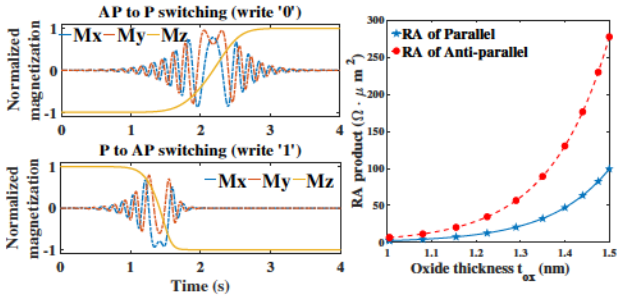
TABLE I. SIMULATION PARAMETERS

Parameter	Value
Free layer dimension,$(W \times L \times t)_{\text{FL}}$	$65 \times 65 \times 2\ nm^3$
Polarization factor, P	0.4
Gilbert Damping Factor, α	0.007
Saturation Magnetization, M_s	$850\ kA/m$
Oxide thickness, t_{ox}	$1.2\ nm$
RA product, RA_p / TMR	$10.58\ \Omega \cdot \mu m^2$ / 171.2%
Supply voltage	$1\ V$
CMOS technology	$45\ nm$
STT-MRAM cell area	$48F^2$
Access transistor width	9F
Cell aspect Ratio	1.34

Based on the simulation parameters listed in Table I, the magnetization dynamic from LLG equation can provide the relative angle θ between the magnetization of PL ($\hat{z}$) and FL (m). Therefore, the real-time conductance of MTJ (G_{MTJ}) is given by [24]:

$$G_{\text{MTJ}} = \frac{G_P + G_{AP}}{2} + \frac{G_P - G_{AP}}{2}\cos\theta \quad (3)$$

where G_P and G_{AP} are the conductance of MTJ in parallel ($\theta = 0$) and anti-parallel ($\theta = 180$) configurations. Both



(a) (b)

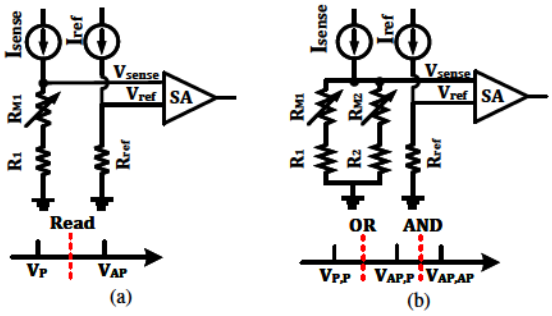
Fig. 3: (a) The transient plot of the normalized magnetization switching in x-, y- and z-axis, with provides STT-MRAM write scheme. (b) The Resistance-Area product w.r.t the thickness of MTJ tunnel oxide (t_{ox}).

G_P and G_{AP} are obtained from the atomistic level simulation framework based on Non-Equilibrium Green's Function (NEGF) [25], while the Resistance-Area Product with respect to the thickness of MTJ tunnel oxide is shown in Fig. 3b.

III. STT-MRAM BASED IN-MEMORY COMPUTING

In addition to the conventional memory function of STT-MRAM, through modifying the row/column decoders of memory sub-array and sensing circuitry, a reconfigurable in-memory logic (i.e. AND, OR, XOR and etc.) can be implemented without add-on logic circuits.

A. Sensing-based in-memory computing scheme



(a) (b)

Fig. 4: The idea of voltage comparison between V_{sense} and V_{ref} for (a) memory read and (b) in-memory logic operation.

The key idea to perform memory read and in-memory computing is to choose different thresholds when sensing the selected memory cell(s). As shown in Fig. 4a, for memory read operation, a single memory cell is addressed and routed in the memory read path to generate a sense voltage (V_{sense}), which will be compared with a reference voltage (V_{ref}). Owing to the parallel- or anti-parallel state of selected STT-MRAM bit-cell (R_{M1}), the sense voltage are V_P or V_{AP} (V_P<V_{AP}) respectively. Thus, through setting the reference voltage at (V_{AP}+V_P)/2, the sense amplifier outputs binary '1' when V_{sense}>V_{ref}, otherwise the sense amplifier outputs '0'.

Fig. 4b depicts the sensing-based method of in-memory Boolean computing, where two memory bit-cells (R_{M1} and R_{M2}) are sensed simultaneously. R_1 and R_2 corresponds to the access transistors within the sensing path. Owing to the different resistance combinations of two selected STT-MRAM bit-cells (i.e. R_{AP}, R_{AP}; R_{AP}, R_P; R_P, R_P), three different sense voltages V_{sense} ($V_{AP,AP}$; $V_{AP,P}$; $V_{P,P}$) could be generated respectively. Consider setting the reference voltage as ($V_{AP,AP}$+$V_{AP,P}$)/2 through tuning reference resistance, the sense amplifier only outputs '1' when both selected STT-MRAMs are in anti-parallel state (V_{sense}>V_{ref}). Thus, this sensing operation with modified reference voltage performs an AND logic operation taken the binary data stored in R_{M1} and R_{M2} as logic inputs. Similarly, when the reference voltage is shifted to ($V_{P,P}$+$V_{AP,P}$)/2, the OR logic operation can be performed as well. Therefore, through tuning the reference voltage for comparison, the sense amplifier can perform reconfigurable in-memory computations.

B. in-memory computing platform design

Based on the STT-MRAM device modeling approach discussed above, we have performed the circuit level STT-MRAM simulations in Cadence Spectre with 45nm NCSU PDK [26] as CMOS library. The MTJ resistive model is obtained from the modeling approach described in Section II with respect to the device parameters listed in Table. I.

Fig. 5a depicts the architecture of memory sub-array, where memory read/write path of the specific bit-cell is enabled by the row/column decoders. As shown in Fig. 5c, the modified row/column decoders can enable either single line (memory write/read) or double lines (bit-wise Boolean computation), depending on the addresses (Addr1 and Addr2) provided. For memory write, the voltage drop across BL and SL is generated by the Voltage Drivers (VD), which realize the fast memory switching as depicted in Fig. 3a. For memory read and Boolean computation, a small sense current ($I_{sense} \simeq 3\mu A$) is injected into the read path to generate a sense voltage (V_{sense}), which is taken as the input of modified sense circuit. As shown in Fig. 5d, the modified sense circuit can provide memory read, AND/NAND, OR/NOR and XOR/XNOR functions, through combining two sense amplifiers (i.e. StrongARM latch shown in Fig. 5b), external CMOS logic gate and control units. Owing to the complementary outputs of SA, the modified sensing circuit can provide NAND and NOR without additional cost. Moreover, according to the Boolean representation of $p \bigoplus q = (p \vee q) \wedge \overline{(p \wedge q)}$, the XOR logic can be realized with two sense amplifiers (i.e. performing AND and NOR logic respectively) and an additional CMOS NOR gate. It is noteworthy that, other computation requiring two threshold can also be implement by adding additional reference branches. The operation of such sense circuit is determined by the control signals (EN_{AND}, EN_{OR} and EN_M), while the desired result is acquired by the selection signal (SEL) of the output multiplexer. Note that, only one sense amplifier is used during AND/OR/memory-read operation, in order to reduce the power consumption of sensing.

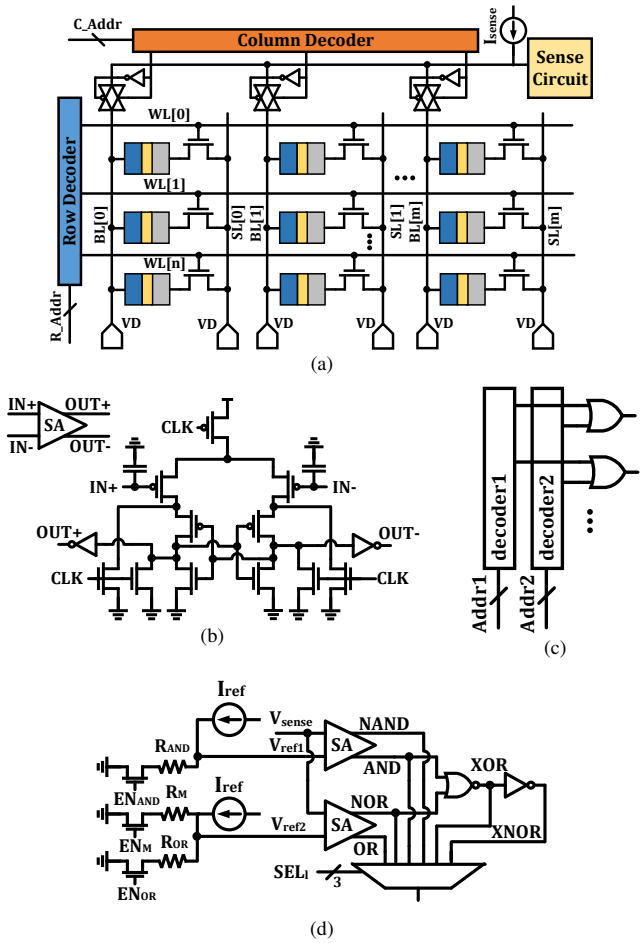


Fig. 5: (a) The modified sub-array structure of STT-MRAM. (b) The schematic of StrongARM latch as Sense Amplifier (SA). (c) Modified decoder which provides single/multiple lines enable function. (d) Modified sense circuit for regular memory W/R and in-memory computing operations.

C. performance evaluation

Fig. 6 depicts the transient simulation result of the sense circuit under a 2ns period clock signal (CLK), which takes the data stored in MRAM1 and MRAM2 as inputs. When CLK is high, the sense amplifier is in pre-charge phase and the output is reset to '0'. When CLK is low, the sense amplifier is in sampling phase, and generates logic computation result depending on the reference voltage configuration. V_{cmp} includes all the input signals of SAs, which are sense voltage (V_{sense}) and two reference voltage (V_{ref1} and V_{ref2}). V_{ref1} is set to ($V_{AP,AP}$+$V_{AP,P}$)/2, and V_{ref2} is set to ($V_{P,P}$+$V_{AP,P}$)/2, for performing AND and OR respectively. Note that, the ripple of V_{cmp} are resulted from the kickback noise due to the clock switching of SA. In general, the transient curve has demonstrated the correct logic operation of sense circuit with around 200ps output latency.

Furthermore, in order to validate the variation tolerance of

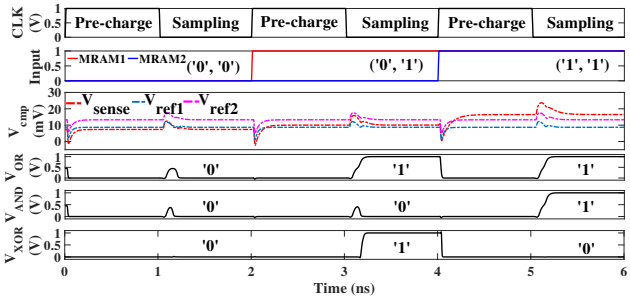


Fig. 6: Transient simulation results of in-memory computing operations (i.e. AND, OR and XOR).

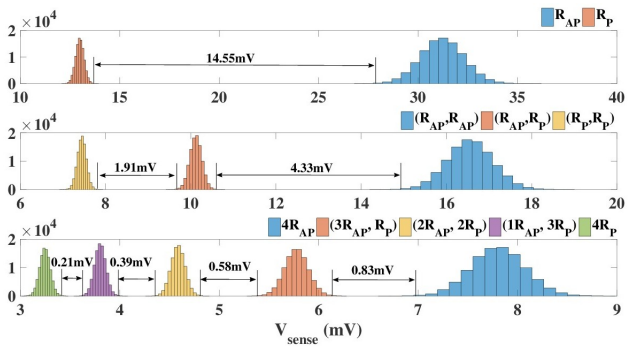


Fig. 7: Monte-Carlo simulation result of sense voltage (V_{sense}) distribution for (top) conventional memory read operation with single STT-MRAM, and in-memory computing operation with (middle) two selected STT-MRAM bit-cells (down) four selected STT-MRAM bit-cells.

sense circuit, we have performed Monte-Carlo simulation with 100000 trials. A $\sigma = 2\%$ variation is added to the Resistance-Area product (RA_P), and a $\sigma = 5\%$ process variation is added on the TMR. The simulation result of sense voltage (V_{sense}) distributions in Fig. 7 shows the sense margin for conventional memory read, two fan-in in-memory computation and four fan-ins sense-based operation. It can be seen that sense margin gradually reduces when increasing the number of fan-ins (selected STT-MRAM cells for computation). Note that, such sense margin could be improved by increasing the sense current, but with a sacrifice of read operation energy efficiency.

TABLE II. Performance evaluation of STT-MRAM based in-memory computing platform

Metrics	Memory mode		Compute mode
	Write	Read	
Dynamic Energy	826.149pJ	870.042pJ	985.851pJ
Mat Dynamic Energy	9.151pJ	14.637pJ	21.303pJ
Subarray Dynamic Energy	2.218pJ	3.590pJ	5.252pJ
Leakage Power	830.847mW		
Area	1.271mm x 5.216mm = 6.632mm^2		

In order to evaluate the performance of the proposed STT-MRAM based in-memory computing platform, we configure

the memory chip organization by dividing it into multiple Banks consisting of multiple Mats. Each Mat includes multiple sub-arrays organized in a H-tree routing manner. We employ modified self-consistent NVSim [27] along with an in-house developed C++ code to verify the architecture level performance of STT-MRAM in-memory computing platform, which includes the metrics for read/write in memory mode and logic operations in compute mode. Table II lists the energy consumption of a sample 4MB memory with 512 word-width in 45nm process node. Note that, the energy overhead comes from the previous described modified decoders, sense amplifier, etc.

IV. Edge detection using in-memory Boolean computing

In this section, an in-memory edge extraction algorithm is proposed to acquire the image edge feature, which leverages the intrinsic in-memory computing functionality and massively reduces the data communication between memory and main processing unit.

A. Binary Image

Considering a binary image with dimension of $m \times n$. As described in Fig. 8a, a 2 by 2 edge detection operator shift from the top-left to the bottom-right corner. If and only if all four neighboring pixels are of the identical magnitude (binary '1' or '0'), it indicates the absence of image variation. Then, the corresponding pixel at the edge map is set to binary '0' (black). For the other combinations, the corresponding edge map pixel is set to binary '1' (white). The corresponding edge extraction algorithm is described in Algorithm 1.

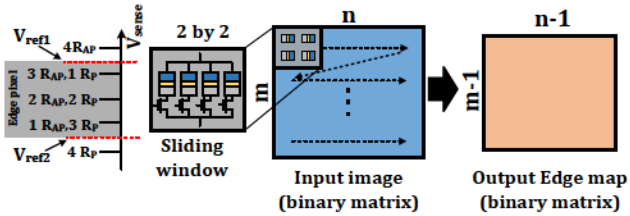


Fig. 8: (a) The edge extraction procedure for binary image (i.e. individual bit-plane). (b) Original binary image and (c) its edge map extracted from memory.

In order to extract the edge map through the proposed in-memory computing platform, the dimension of memory sub-array should be sufficient to store the entire binary image matrix. For extracting image edge, four neighboring STT-MRAM bit-cells are selected simultaneously by the modified row and column decoders. Then, the edge detection operator can be intrinsically performed through the previous described modified sense amplifier by configuring the two reference voltages to $(V_{4AP}+V_{3AP,P})/2$ and $(V_{4P}+V_{3P,AP})/2$, respectively, as marked in Fig. 8a. A binary image and its extracted edge image are displayed in 8b and 8c as an example. Note that, in such process, the edge map could be directly 'read out' using the modified sense amplifier. While, in traditional design, the raw image needs to be read out from memory and sent to processing unit for edge extraction computation.

Algorithm 1 In-memory edge extraction algorithm for binary image (single bit-plane)

```
Input: input binary image I(m,n)
Output: edge map OUT = edge( I );
  for i = 1 to m-1 do
    for j = 1 to n-1 do
      tmp = sum( I(i:i+1, j:j+1) );
      if tmp < 1 or > 3 then
        OUT(i,j) = 1;
      else
        OUT(i,j) = 0;
      end if
    end for
  end for
  return OUT
```

Algorithm 2 In-memory edge extraction algorithm for grayscale image (multiple bit-planes)

```
Input: input grayscale image I(m,n,8); p: number of bitplanes
  used for edge extraction
Output: edge map OUT;
  OUT = zeros(m-1,n-1);
  for k = 8+1-p to 8 do
    tmp_edge = edge( I(:,:,k) );
    for i = 1 to m-1 do
      for j = 1 to n-1 do
        OUT = or( tmp_edge(i,j), OUT(i,j) );
      end for
    end for
  end for
  return OUT
```

B. Grayscale Image

The proposed in-memory edge extraction algorithm could also be extended to greysalce image, as described in Algorithm 2. Considering an 8-bit grayscale image (Fig. 9a) is divided into eight bit-planes, from most significant bit (MSB) to least significant bit (LSB) as shown in Figs. 9b to 9i. It presents that the high order bit-planes contains more information for local processing, while the low order bit-plane are noisy. In order to perform the edge extraction based on the grayscale image, the same algorithm shown in Algorithm 1 is applied on bit-planes separately. Then, an in-memory OR operation is implemented on each pixel over the calculated

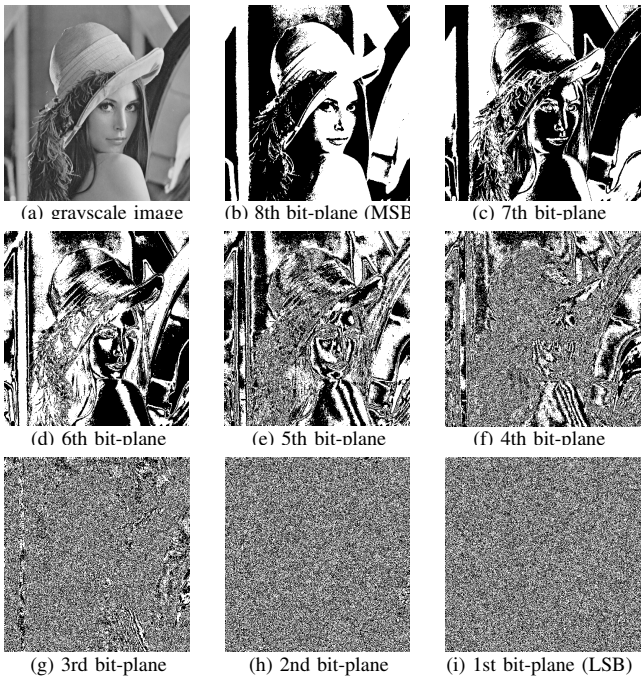


Fig. 9: (a) grayscale image of lena. The bit-planes from MSB to LSB are shown (b-i), which indicate that most image informations are included in (b-e).

bit-planes. The extracted edge images are shown in Fig. 10, through the combination of different numbers of bit-planes.

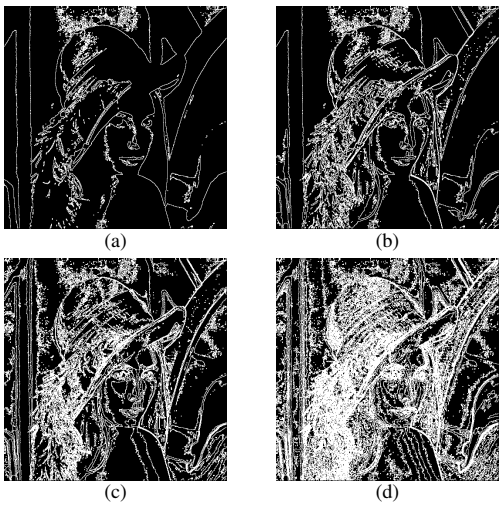


Fig. 10: The edge image that combines (i.e. OR operation) the in-memory edge extraction result of (a) 8th (b) 8th-7th (c) 8th-6th (d) 8th-5th most significant bit-planes.

The F-measure ($\frac{2 \cdot Precision \cdot recall}{precision + recall}$) is taken as indicator of extracted edge quality. We perform the edge detection evaluation based on the Berkeley Segmentation Dataset (BSD300) [28], which contains 300 natural images and human annotations as ground truth. The simulation results listed in Table III shows that the best F-measure (only 8_{th} bit-plane) of our in-memory edge method merely has 0.05∼0.06 (∼10%) degradation in comparison with Prewitt [12], Sobel [11] and Roberts [13] such conventional edge operators. Increasing the number of bit-planes for computation may involves more image details for edge extraction, but introducing amounts of noise as well which lowers F-measure of extracted edge image.

TABLE III. Evaluation of different edge detection algorithms on BSD300 dataset

Conventional edge oeprator	Canny [1]	Prewitt [2]	Sobel [3]	Roberts [4]
F-measure	0.58	0.48	0.48	0.47
Our methods:	1 bit-plane (Fig. 10a)	2 bit-planes (Fig. 10b)	3 bit-planes (Fig. 10c)	4 bit-planes (Fig. 10d)
F-measure	0.42	0.40	0.35	0.32

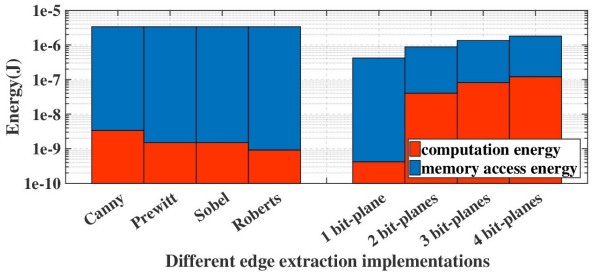


Fig. 11: Energy consumption of conventional edge detection algorithm using ASIC and our in-memory edge extraction method for one image (lena.tif, 512×512 pixels).

We further employ NVsim (memory setup in Section III-C) and design compiler (processing element synthesis) to evaluate the image edge extraction system energy consumption in 45nm CMOS technology. Fig. 11 shows the energy consumption comparison between various image edge extraction algorithms. It shows that our edge extraction method achieves more than $8\times$ energy reduction in comparison with conventional edge extraction algorithms. Note that, the total energy consumption includes two parts: computation energy and memory access energy. Since the memory access energy for transferring data from memory to processing element (PE) is much larger than computation energy, reducing data communication between memory and PE greatly reduces the total energy dissipation. The "computation energy" for in-memory computing algorithm shown in Fig. 11 comes from decoders and SAs configuration, data buffer, copy and write-back operations in in-memory data processing. However, such overhead is much smaller, due to local processing, compared with memory access energy.

C. Edge-based Motion Detection

In additional to using the F-measure for evaluating our in-memory edge extraction quantitatively, we utilize the edge-based motion detection [29] to demonstrate the extracted images have relatively good quality for the edge related application. Note that, the extracted edge from memory is

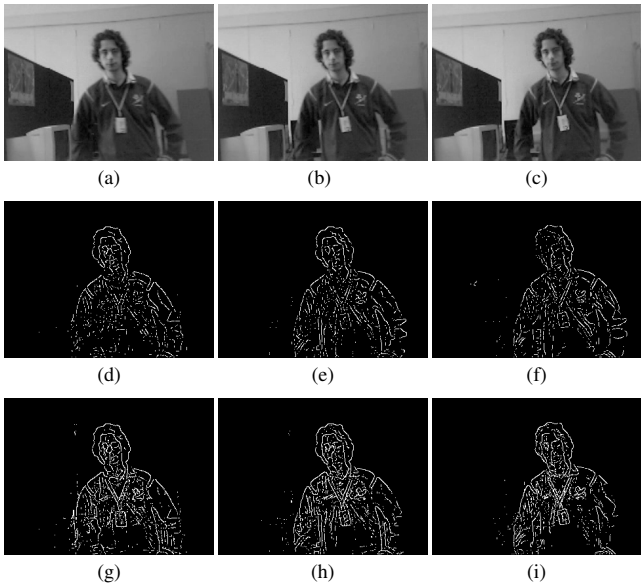


Fig. 12: (a-c) The three successive frames captured from a stationary camera. The corresponding edge representation of moving object are shown in (d-f) using Canny edge detector. (g-h) our in-memory edge extraction algorithm.

forwarded to the main processing unit (i.e. CPU or GPU) for processing. Such motion detection is performed with single face dataset [30], where three successive frames and its motion detection result using conventional canny edge operator and our in-memory edge extraction method are taken as example and shown in Fig. 12. It can be seen that the object motion is successfully detected using our extracted edge maps, similar as Canny edge detector.

V. Conclusion

In this work, we have presented a STT-MRAM based in-memory computing platform and novel edge extraction algorithm for in-memory image preprocessing. In-memory computing has emerged as promising computation architecture for big-data application, where the data communication between memory and primary processing unit (i.e. CPU and GPU) merely involves the preprocessed data. Such computation methodology drastically reduces the data communication, thus leading to energy-efficient operation and extreme computation parallelism. As a case study, we have proposed an image edge extraction algorithm that can be intrinsically performed by our in-memory computing platform with 8× energy reduction, and only 10% F-measure score degradation.

References

[1] Y. Wang *et al.*, "An energy-efficient nonvolatile in-memory computing architecture for extreme learning machine by domain-wall nanowire devices," *IEEE TNANO*, vol. 14, no. 6, pp. 998–1012, 2015.

[2] B. Hoefflinger, "Itrs: The international technology roadmap for semiconductors," in *Chips 2020*. Springer, 2011, pp. 161–174.

[3] P. Chi *et al.*, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *Proceedings of the 43rd ISCA*. IEEE Press, 2016, pp. 27–39.

[4] H.-S. P. Wong, S. Raoux, S. Kim, J. Liang, J. P. Reifenberg, B. Rajendran, M. Asheghi, and K. E. Goodson, "Phase change memory," *Proceedings of the IEEE*, vol. 98, no. 12, pp. 2201–2227, 2010.

[5] Y. Kim *et al.*, "Write-optimized reliable design of stt mram," in *Proceedings of the 2012 ACM/IEEE ISLPED*. ACM, 2012, pp. 3–8.

[6] X. Fong *et al.*, "Spin-transfer torque memories: Devices, circuits, and systems," *Proceedings of the IEEE*, vol. 104, pp. 1449–1488, 2016.

[7] Y. Seo *et al.*, "High performance and energy-efficient on-chip cache using dual port (1r/1w) spin-orbit torque mram," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 3, pp. 293–304, 2016.

[8] G. Prenat *et al.*, "Ultra-fast and high-reliability sot-mram: From cache replacement to normally-off computing," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, pp. 49–60, 2016.

[9] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[10] X. Jia, H. Huang, Y. Sun, J. Yuan, and D. M. Powers, "A novel edge detection approach using a fusion model," *Multimedia Tools and Applications*, vol. 75, no. 2, pp. 1099–1133, 2016.

[11] I. Sobel, "Camera models and machine perception," DTIC Document, Tech. Rep., 1970.

[12] J. M. Prewitt, "Object enhancement and extraction," *Picture processing and Psychopictorics*, vol. 10, no. 1, pp. 15–19, 1970.

[13] L. G. Roberts, "Machine perception of three-dimensional soups," Ph.D. dissertation, Massachusetts Institute of Technology, 1963.

[14] D. Marr and E. Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 207, no. 1167, pp. 187–217, 1980.

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

[16] S. Jain, A. Ranjan, K. Roy, and A. Raghunathan, "Computing in memory with spin-transfer torque magnetic ram," *arXiv preprint arXiv:1703.02118*, 2017.

[17] Z. He, S. Angizi, F. Parveen, and D. Fan, "Leveraging dual-mode magnetic crossbar for ultra-low energy in-memory data encryption," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*. ACM, 2017, pp. 83–88.

[18] G. Autès, J. Mathon, and A. Umerski, "Strong enhancement of the tunneling magnetoresistance by electron filtering in an fe/mgo/fe/gaas (001) junction," *Physical review letters*, p. 217202, 2010.

[19] P. Mavropoulos *et al.*, "Half-metallic ferromagnets for magnetic tunnel junctions by ab initio calculations," *Physical Review B*, vol. 72, no. 17, p. 174428, 2005.

[20] M. Bowen *et al.*, "Nearly total spin polarization in la 2/3 sr 1/3 mno 3 from tunneling experiments," *Applied Physics Letters*, vol. 82, no. 2, pp. 233–235, 2003.

[21] J. Hayakawa *et al.*, "Dependence of giant tunnel magnetoresistance of sputtered cofeb/mgo/cofeb magnetic tunnel junctions on mgo barrier thickness and annealing temperature," *Japanese Journal of Applied Physics*, vol. 44, no. 4L, p. L587, 2005.

[22] M. J. Donahue, "Oommf user's guide, version 1.0," *-6376*, 1999.

[23] X. Fong, S. K. Gupta, N. N. Mojumder, S. H. Choday, C. Augustine, and K. Roy, "Knack: A hybrid spin-charge mixed-mode simulator for evaluating different genres of spin-transfer torque mram bit-cells," in *2011 International Conference on Simulation of Semiconductor Processes and Devices*, Sept 2011, pp. 51–54.

[24] D. Fan, S. Maji, K. Yogendra, M. Sharad, and K. Roy, "Injection-locked spin hall-induced coupled-oscillators for energy efficient associative computing," *IEEE Transactions on Nanotechnology*, vol. 14, no. 6, pp. 1083–1093, Nov 2015.

[25] G. Panagopoulos *et al.*, "A framework for simulating hybrid mtj/cmos circuits: Atoms to system approach," in *DATE*. IEEE, 2012.

[26] (2011) Ncsu eda freepdk45. [Online]. Available: http://www.eda.ncsu.edu/wiki/FreePDK45:Contents

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

[28] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 5, pp. 898–916, 2011.
[29] A. Sappa and F. Dornaika, "An edge-based approach to motion detection," *Computational Science–ICCS 2006*, pp. 563–570, 2006.
[30] E. Maggio and A. Cavallaro, "Hybrid particle filter and mean shift tracker with adaptive transition model," in *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, vol. 2, March 2005, pp. 221–224.