An Energy-efficient Matrix Multiplication Accelerator by Distributed In-memory Computing on Binary RRAM Crossbar

Leibin Ni*, Yuhao Wang*, Hao Yu*, Wei Yang†, Chuliang Weng† and Junfeng Zhao†
*School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore
†Shannon Laboratory, Huawei Technologies Co., Ltd, China
Email:haoyu@ntu.edu.sg

***Abstract*—Emerging resistive random-access memory (RRAM) can provide non-volatile memory storage but also intrinsic logic for matrix-vector multiplication, which is ideal for low-power and high-throughput data analytics accelerator performed in memory. However, the existing RRAM-based computing device is mainly assumed on a multi-level analog computing, whose result is sensitive to process non-uniformity as well as additional AD- conversion and I/O overhead. This paper explores the data analytics accelerator on binary RRAM-crossbar. Accordingly, one distributed in-memory computing architecture is proposed with design of according component and control protocol. Both memory array and logic accelerator can be implemented by RRAM-crossbar purely in binary, where logic-memory pairs can be distributed with protocol of control bus. Based on numerical results for fingerprint matching that is mapped on the proposed RRAM-crossbar, the proposed architecture has shown 2.86x faster speed, 154x better energy efficiency, and 100x smaller area when compared to the same design by CMOS-based ASIC.**

I. Introduction

Data intensive analytics will frequently require matrix multiplication with data exchange between memory and logic units. In conventional Von Neumann architecture, the processor and memory are separated with interconnect I/O in between for data communication [1][2]. All entries in database need to be read out from memory to processor where the computation is performed. A large-volume data needs to be hold and communicated, for example, when analyzing image processing. As such, both substantial leakage and dynamic power will be experienced in the memory buffer as well as in I/O communication.

Therefore, for data-oriented computation, it is beneficial to place logic accelerators as close as possible to the memory to alleviate the I/O communication overhead. The cell-level in-memory computing is proposed in [3], where simple logic circuits are embedded among memory arrays. Nevertheless, the according in-memory logic that is equipped in memory cannot be made for complex function, and also the utilization efficiency is low as logic cannot be shared among memory cells. In addition, the significant memory leakage power cannot be resolved in CMOS based technology.

Emerging resistive random-access memory (RRAM) [4][5] has shown great potential to be the solution for data-intensive applications. Besides the minimized leakage power due to non-volatility, RRAM in crossbar structure has been exploited as computational elements [5][6]. As such, both memory and logic components can be realized in a power- and area-efficient manner. More importantly, it can provide a true in-memory logic-memory integration architecture without using I/Os. Nevertheless, the previous RRAM-crossbar based computation is mainly based on an analog fashion with multi-level values [7] or Spike Timing Dependent Plasticity (STDP) [8]. Though it improves computation capacity, the serious non-uniformity of RRAM-crossbar at nano-scale limits its wide applications for data analytics. Moreover, there is significant power consumption from additional AD-conversion and I/Os mentioned in [9].

In this paper, we propose a distributed in-memory computing accelerator on RRAM-crossbar for matrix multiplication. Both computational energy-efficiency and robustness are both greatly improved by using a digitalized RRAM-crossbar for memory and logic units. The memory arrays are paired with the in-memory logic accelerators in a distributed fashion, operated with a protocol of control bus for each memory-logic pair. Moreover, different from the multi-leveled analog RRAM-crossbar, a three-step digitalized RRAM-crossbar is proposed in this paper to perform a binary (or digital) matrix-vector multiplication. One can map the data analytic of fingerprint matching on the proposed RRAM-crossbar. Simulation results show that significant power reduction can be achieved when compared to the CMOS-based ASIC implementation.

The rest of this paper is organized as follows. Section II shows the novel distributed RRAM-crossbar based in-memory computing architecture (XIMA). Section III introduces RRAM-crossbar for matrix-vector multiplication. Section IV presents the mapping details for matrix multiplication on the digitalized RRAM crossbar. Experimental results are presented in Section V with conclusion in Section VI.

II. Architecture Overview

A. Architecture

Conventionally, processor and memory are separate components that are connected through I/Os. With limited width and considerable RC-delay, the I/Os are considered the bottleneck of system overall throughput. As memory is typically organized in H-tree structure, where all leaves of the tree are data arrays, it is promising to impose in-memory computation

978-1-4673-9569-4/16/$31.00 ©2016 IEEE

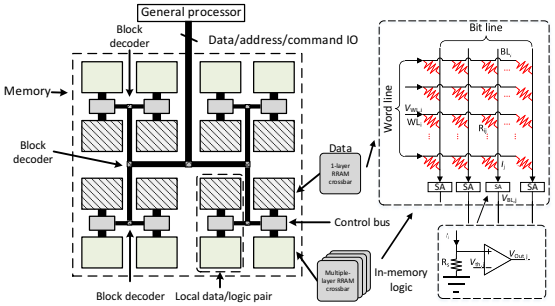

Fig. 1: Overview of distributed in-memory computing architecture on RRAM crossbar

with parallelism at this level. In this paper, we propose a distributed RRAM-crossbar in-memory architecture (XIMA). Because both data and logic units have uniform structure when implemented on RRAM-crossbar, half of the leaves are exploited as logic elements and are paired with data arrays. The proposed architecture is illustrated in Fig. 1. The distributed local data-logic pairs can form one local data path such that the data can be processed locally in parallel, without the need of being readout to the external processor.

Coordinated by the additional controlling unit called *in-pair control bus* the in-memory computing is performed in following steps. (1) logic configuration: processor issues the command to configure logic by programming logic RRAM-crossbar into specific pattern according to the functionality required; (2) load operand: processor sends the data address and corresponding address of logic accelerator input; (3) execution: logic accelerator can perform computation based on the configured logic and obtain results after several cycles; (4) write-back: computed results are written back to data array directly but not to the external processor.

With emphasis on different functionality, the RRAM crossbars for data storage and logic unit have distinctive interfaces. The data RRAM-crossbar will have only one row activated at one time during read and write operations, and logic RRAM-crossbar; however, can have all rows activated spontaneously as rows are used to take inputs. As such, the input and output interface of logic crossbar requires AD/DA conversions, which could outweigh the benefits gained. Therefore, in this paper, we propose a conversion-free digital-interfaced logic RRAM crossbar design, which uses three layers of RRAM crossbars to decompose a complex function into several simple operations that digital crossbar can tackle.

B. Communication Protocol

The conventional communication protocol between external processor and memory is composed of *store* and *load* action identifier, address that routes to different locations of data arrays, and data to be operated. With additional in-memory computation capacity, the proposed distributed in-memory computing architecture requires modifications on the current communication protocol. The new communication instructions are proposed in TABLE I, which is called in-pair control.

TABLE I: Protocols between external processor and control bus

Inst.	Op. 1	Op. 2	Action	Function
SW	Addr 1	Addr 2	Addr 1 data to Addr 2	store data, configure logic, in-memory results write-back
	Data	Addr	store data to Addr	
LW	Addr	-	read data from Addr	standard read
ST	Block Idx	-	switch logic block on	start in-memory computing
WT	-	-	wait for logic block response	halt while performing in-memory computing

In-pair control bus needs to execute instructions in TABLE I. SW (store word) instruction is to write data into RRAMs in data array or in-memory logic. If target address is in data array, it will be a conventional write or result write-back; otherwise it will be logic configuration. LW (load word) instruction performs as conventional read operation. ST (start) instruction means to switch on the logic block for computing after computation setup. WT (wait) operation is to stop reading from instruction queue during computing.

Besides communication instructions, memory address format is also different from that in the conventional architecture. To specify a byte in the proposed architecture, address includes the following identifier segments. Firstly, the data-logic pair index segment is required, which is taken by block decoders to locate the target data-logic pair. Secondly, one-bit flag is needed to clarify that whether the target address is in data array or in-memory logic crossbar. Thirdly, if logic accelerator is the target, additional segment has to specify the layer index. Lastly, rest of address segment are row and column indexes in each RRAM-crossbar. An address example for data array and in-memory logic is shown in Fig. 2.

C. Control Bus

Given the new communication protocol between general processor and memory is introduced, one can design the according control bus as shown in Fig. 2. The control bus is composed of an instruction queue, an instruction decoder, an address decoder and a SRAM array. As the operation frequency of RRAM-crossbar is slower than that of external processor, instructions issued by the external processor will be stored in the instruction queue first. They are then analyzed by instruction decoder on a first-come-first-serve (FCFS) basis. The address decoder obtains the row and column index from the instruction; and SRAM array is used to store temporary

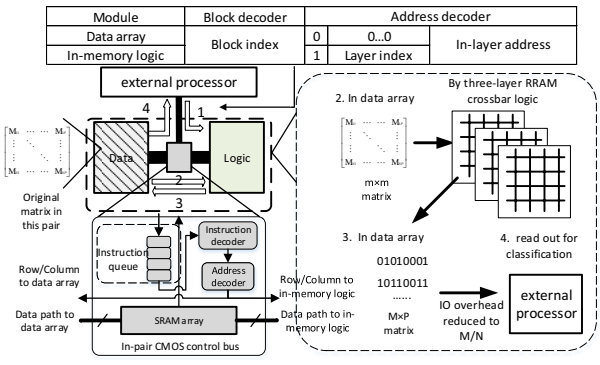

Fig. 2: Detailed structure of control bus and communication protocol

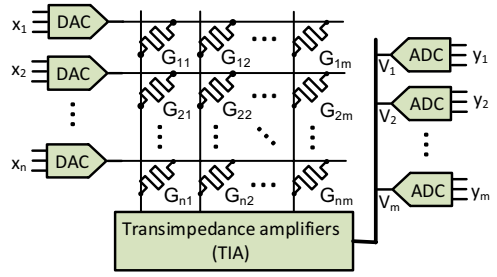

Fig. 3: Traditional analog-fashion RRAM crossbar with ADC and DAC

data such as computation results, which are later written back to data array.

III. RRAM-CROSSBAR FOR MATRIX-VECTOR MULTIPLICATION

In this paper, we implement matrix-vector multiplication in the proposed XIMA. It is one always-on operation in various data-analytic applications such as compressive sensing, machine learning. For example, the feature extraction can be achieved by multiplying Bernoulli matrix in [10].

Matrix multiplication can be denoted as $Y = \Phi X$, where $X \in \{0,1\}^{N\times P}$ and $\Phi \in \{0,1\}^{M\times N}$ are the multiplicand matrices, and $Y \in \mathbb{Z}^{M\times P}$ is the result matrix.

A. Traditional Analog RRAM Crossbar

RRAM is an emerging non-volatile memory based on two-terminal junction devices, whose resistance can be controlled by the integral of externally applied currents. The fabric of crossbar intrinsically supports matrix-vector multiplication where vector is represented by row input voltage levels and matrix is denoted by mesh of RRAM resistances. As shown in Fig. 3, by configuring Φ into the RRAM crossbar, analog computation $y = \Phi x$ by RRAM crossbar can be achieved.

However, such analog RRAM crossbar has two major drawbacks. Firstly, the programming of continuous-valued RRAM resistance is practically challenging due to large RRAM process variation. Specifically, the RRAM resistance is determined by the integral of current flowing through, which leads to a switching curve as shown in Fig. 4 (a) With the process variation, the curve may shift and leave intermediate values very unreliable to program, as shown in Fig. 4 (b). Secondly, the A/D and D/A converters are both timing-consuming and power-consuming. In our simulation, the A/D and D/A conversion may consume up to 85.5% of total operation energy in $65nm$ as shown in Fig. 5.

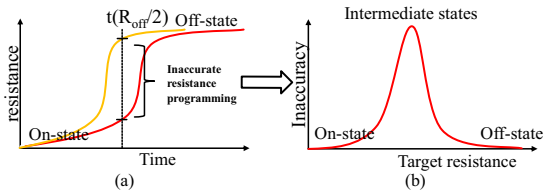

Fig. 4: (a) Switching curve of RRAM under device variations (b) Programing inaccuracy for different RRAM target resistances

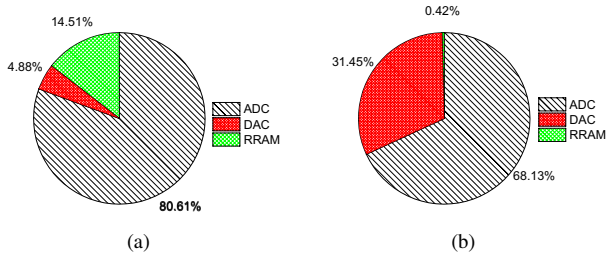

Fig. 5: (a) Power consumption of analog-fashion RRAM crossbar (b) Area consumption of analog-fashion RRAM crossbar

B. Proposed Digitalized RRAM Crossbar

To overcome the aforementioned issues, we propose a full-digitalized RRAM crossbar for matrix-vector multiplication. Firstly, as ON-state and OFF-state are much more reliable than intermediate values shown in Fig. 4, only binary values of RRAM are allowed to reduce the inaccuracy of RRAM programming. Secondly, we deploy a pure digital interface without A/D conversion.

In RRAM crossbar, We use V_{wl}^i and V_{bl}^j to denote voltage on ith wordline (WL) and jth bitline (BL). R_{off} and R_{on} denote the resistance of off-state and on-state. In each sense amplifier (SA), there is a sense resistor R_s with fixed and small resistance. The relation among these three resistance is $R_{off} \gg R_{on} \gg R_s$. Thus, the voltage on jth BL can be presented by

$$V_{bl}^j = \sum_{i=1}^{m} g_{ij} V_{wl}^i R_s \tag{1}$$

where g_{ij} is the conductance of R_{ij}.

The key idea behind digitalized crossbar is the use of comparators. As each column output voltage for analog crossbar is continuous-valued, comparators are used to digitize it according to the reference threshold applied to SA in Fig. 1,

$$O_j = \begin{cases} 1, \ if \ V_{bl}^j \geq V_{th}^j \\ 0, \ if \ V_{bl}^j < V_{th}^j \end{cases} \tag{2}$$

However, the issue that rises due to the digitalization of analog voltage value is the loss of information. To overcome this, three techniques are applied. Firstly, multi-thresholds are used to increase the quantization level so that more information can be preserved. Secondly, the multiplication operation is decomposed into three sub-operations that binary crossbar can well tackle. Thirdly, the thresholds are delicately selected at the region that most information can be preserved after the digitalization.

IV. IMPLEMENTATION OF DIGITAL MATRIX MULTIPLICATION

In this section, hardware mapping of matrix multiplication on the proposed architecture is introduced. The logic required is a matrix-vector multiplier by the RRAM-crossbar. Here, a three-step RRAM-crossbar based binary matrix-vector multiplier is proposed, in which both the input and output of the RRAM-crossbar are binary data without the need of ADC.

The three RRAM-crossbar step: parallel digitizing, XOR and encoding are presented in details as follows.

A. Parallel Digitizing

The first step is called parallel digitizing, which requires $N \times N$ RRAM crossbars. The idea is to split the matrix-vector multiplication to multiple inner-product operations of two vectors. Each inner-product is produced by one RRAM crossbar. For each crossbar, as shown in Fig. 6, all columns are configured with same elements that correspond to one column in random Boolean matrix Φ, and the input voltages on word-lines (WLs) are determined by x. As $g_{on} \gg g_{off}$, current on RRAMs with high impedance are insignificant, so that the voltages on BLs approximately equal to $kV_r g_{on} R_s$ according to Eq. (1) where k is the number of RRAM with in low-resistance state (g_{on}).

It is obvious that voltages on bit-lines (BLs) are all identical. Therefore, the key to obtain the inner-product is to set ladder-type sensing threshold voltages for each column,

$$V_{th,j} = \frac{(2j+1)V_r g_{on} R_s}{2}, \tag{3}$$

where $V_{th,j}$ is the threshold voltage for the j_{th} column. The $O_{i,j}$ is used to denote the output of column j in RRAM crossbar step i after sensing. Therefore, for the output we have

$$O_{1,j} = \begin{cases} 1, & j \leq s \\ 0, & j > s, \end{cases} \tag{4}$$

where s is the inner-product result. In other words, the first $(N-s)$ output bits are 0 and the rest s bits are 1 ($s <= N$). For example, the output that corresponds to 3 is 11100000 ($N = 8$).

B. XOR

The inner-product output of parallel digitizing step is determined by the position where $O_{1,j}$ changes from 0 to 1. The XOR takes the output of the first step, and performs XOR operation for every two adjacent bits in $O_{1,j}$, which gives the result index. For the same example of 3, the first step output 11100000 will become 001000000. The XOR operation based on RRAM crossbar is shown in Fig. 7. According to parallel digitizing step, $O_{1,j}$ must be 1 if $O_{1,j+1}$ is 1.

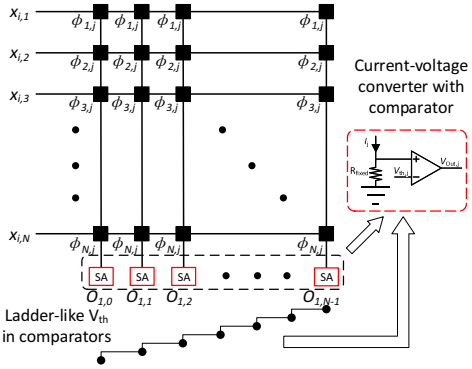

Fig. 6: Parallel digitizing step of RRAM crossbar in matrix multiplication

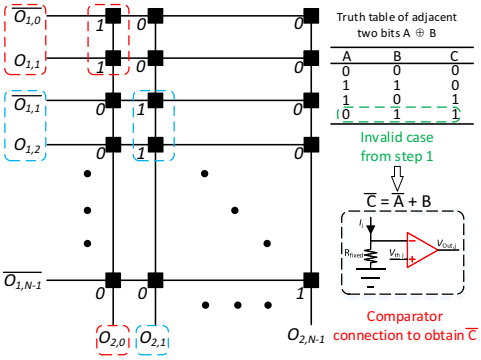

Fig. 7: XOR step of RRAM crossbar in matrix multiplication

Therefore, XOR operation is equivalent to the AND operation $O_{1,j} \oplus O_{1,j+1} = O_{1,j}\overline{O_{1,j+1}}$, and therefore we have

$$\overline{O_{2,j}} = \begin{cases} \overline{O}_{1,j} + O_{1,j+1}, & j < N-1 \\ \overline{O}_{1,j}, & j = N-1. \end{cases} \tag{5}$$

In addition, the threshold voltages for the columns have to follow

$$V_{th,j} = \frac{V_r g_{on} R_s}{2} \tag{6}$$

Eqs. (5) and (6) show that only output of s_{th} column is 1 on the second step, where s is the inner product result. Each crossbar in XOR step has the size of $N \times (2N-1)$.

C. Encoding

The third step takes the output of XOR step and produces s in binary format as an encoder. For example, the output of this step will be $(0...010)$ if $s = 3$. In the output of XOR step, as only one input will be 1 and others are 0, according binary information is stored in corresponding row, as shown in Fig. 8. Encoding step needs $N \times n$ RRAMs, where $n = \lceil \log_2 N \rceil$ is the number of bits in order to represent N in binary format. The thresholds for the encoding step are set following Eq. 6 as well.

V. Experimental Result

A. Experimental Application: Feature extraction for Fingerprint Matching

Feature extraction or sparse–representation is commonly applied such as in fingerprint image matching, which can be

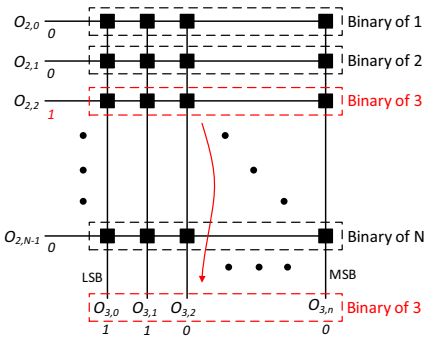

Fig. 8: Encoding step of RRAM crossbar in matrix multiplication

TABLE II: Performance comparison under among software and hardware implementation

Implementation	General purpose processor (MatLab)	CMOS ASIC	Non-distributed digitalized XIMA	Distributed digitalized XIMA	Distributed analog XIMA
Area	$177mm^2$	$5mm^2$	$3.28mm^2$ (800 MBit RRAMs) + $128\mu m^2$	$0.05mm^2$ (12 MBit RRAMs) + 8192 μm^2	$8.32mm^2$
Frequency	4GHz	1GHz	200MHz	200MHz	200MHz
Cycles	-	69,632	Computing: 984 Pre-computing: 262,144	Computing: 984 Pre-computing: 4,096	Computing: 328 Pre-computing: 4,096
Time	$1.78ms$	$69.632\mu s$	Computing: $4{,}920ns$ Pre-computing: $1.311ms$	Computing: $4{,}920ns$ Pre-computing: $20.48\mu s$	Computing: $1{,}640ns$ Pre-computing: $20.48\mu s$
Dynamic power	$84W$	$34.938W$	RRAM: $4.096W$ Control-bus: $100\mu W$	RRAM: $4.096W$ Control-bus: $6.4mW$	RRAM: $1.28W$ Control-bus: $6.4mW$
Energy	$0.1424J$	$2.4457mJ$	RRAM: $20.15\mu J$ Control-bus: $0.131\mu J$	RRAM: $20.15\mu J$ Control-bus: $0.131\mu J$	RRAM: $2.1\mu J$ RRAM: $0.131\mu J$

mapped with ($M \ll N$). This operation can minimize the volume of data to be stored in memory as well as reduce complexity in data analytics. In the following we show how to map this operation on the digitalized RRAM-crossbar. In such process, X is the original fingerprint image in high dimension with ($N \times P$) pixels, Φ a random Bernoulli matrix with ($M \times N$) for feature extract, and Y the features in low dimension with ($M \times P$) pixels. The according dimension reduction ratio γ is

$$\gamma = \frac{M}{N}. \tag{7}$$

In feature extraction of fingerprint image, the random Bernoulli matrix Φ is with fixed elements. Therefore, elements in Bernoulli matrix are stored in RRAMs of logic block, and original image as the input of logic accelerator.

B. Experiment Settings

The hardware evaluation platform is implemented on a computer server with 4.0GHz core and 16.0GB memory. Feature extraction is implemented by general processor, CMOS-based ASIC, non-distributed and distributed in-memory computing based on digitalized RRAM crossbar respectively. For the RRAM-crossbar design evaluation, the resistance of RRAM is set as 1kΩ and 1MΩ as on-state and off-state resistance respectively according to [11]. The general processor implementation is based on MatLab simulation on computer server. A CMOS-based feature extraction design is implemented by Verilog and synthesized with CMOS 65nm low power PDK. The working frequency of general processor implementation is 4.0GHz while the CMOS ASIC feature extraction design frequency is 1.0GHz. For in-memory computing based on the proposed RRAM crossbar, write voltage V_w is set as $0.8V$ and read voltage V_r is set as $0.1V$ as well as duration time of $5ns$. In addition, the analog computation on RRAM-crossbar is performed for comparison based on design in [12].

C. General Performance Comparison

In this section, 1,000 fingerprint images selected from [13] are processed as binarization images and stored in memory with 328×356 resolution. To agree with patch size, random Bernoulli $N \times M$ matrix is with fixed N and M of 356 and 64, respectively. The detailed comparison is shown in Table II with numerical results including energy consumption and delay obtained for one image on average of 1,000 images.

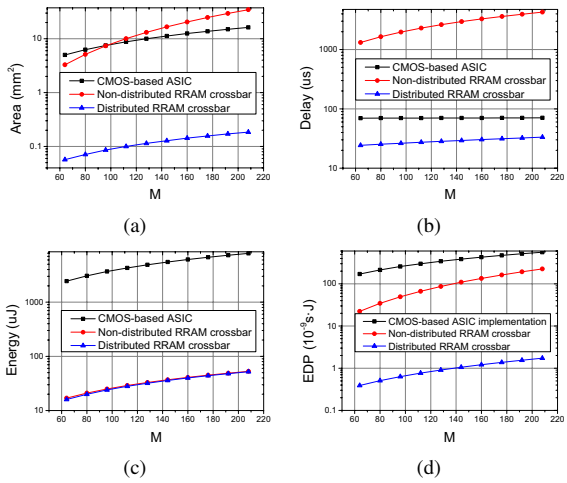

Fig. 9: Hardware Performance Scalability under Different Reduced Dimension for (a) area; (b) delay; (c) energy (d) EDP

Among hardware implementations, in-memory computing based on the proposed XIMA achieves better energy-efficiency than CMOS-based ASIC. Non-distributed XIMA (only one data and logic block inside memory) needs fewer CMOS control bus but large data communication overhead on a single-layer crossbar compared to distributed RRAM crossbar. Although distributed analog RRAM crossbar can achieve the best in energy perspective but has larger area compared to the digitalized one. Shown in Table II, RRAM crossbar in analog fashion only consumes $2.1\mu J$ for one vector multiplication while the proposed architecture requires $20.15\mu J$ because most of power consumption comes from RRAM in computing instead of ADCs. However, ADCs need more area so that RRAM crossbar with analog fashion is $8.32mm^2$ while the proposed one is only $0.05mm^2$ because of the high density of RRAM crossbar.

Calculation error of analog and digitalized RRAM crossbar are compared in Fig. 10, where M and N are both set as 256. Calculation error is very low when RRAM error rate is smaller than 0.004 for both analog and digitalized fashion RRAM. However, when RRAM error rate reaches 0.01, calculation error rate of analog RRAM crossbar goes to 0.25, much higher than the other one with only 0.07. As such, computational error can be reduced in the proposed architecture compared to analog fashion RRAM crossbar.

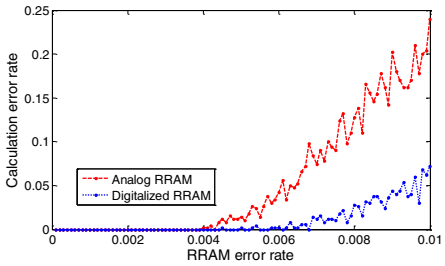

Fig. 10: Calculation error comparison between multi-leveled and binary RRAM

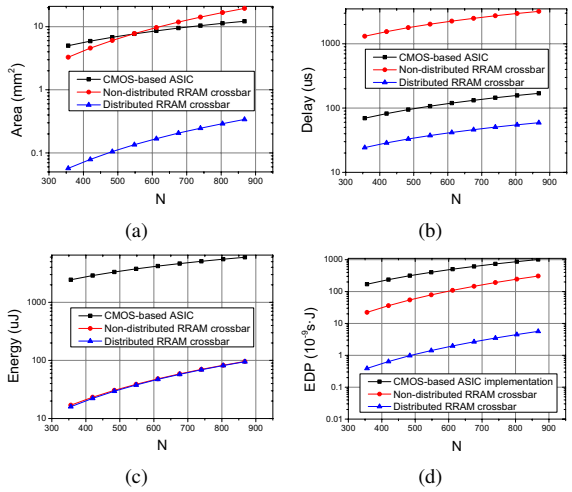

Fig. 11: Hardware Performance Scalability under Different Original Dimension for (a) area; (b) delay; (c) energy (d) EDP

D. Scalability Study

Hardware performance comparison among CMOS-based ASIC, non-distributed and distributed XIMA with varying M is shown in Fig. 9. From area consumption perspective shown in Fig. 9(a), distributed RRAM-crossbar is much better than the other implementations. With increasing M from 64 to 208, its total area is from $0.057mm^2$ to $0.185mm^2$, approximately 100x smaller than the other two approaches. Non-distributed RRAM crossbar becomes the worst one when $M > 96$. From delay perspective shown in Fig. 9(b), non-distributed RRAM crossbar is the worst because it has only one control bus and takes too much time on preparing of computing. Delay of non-distributed RRAM crossbar grows rapidly while distributed RRAM crossbar and CMOS-based ASIC implementation maintains on approximately $21\mu s$ and $70\mu s$ respectively as the parallel design. For energy-efficiency side shown in Fig. 9(c), both non-distributed and distributed RRAM crossbar do better as logic accelerator is off at most of time. The proposed architecture also performs the best in energy-delay product (EDP) shown in Fig. 9(d). Distributed XIMA performs the best among all implementation under different specifications. The EDP is from $0.3 \times 10^{-9} s \cdot J$ to $2 \times 10^{-9} s \cdot J$, which is 60x better than non-distributed RRAM crossbar and 100x better than CMOS-based ASIC.

What is more, hardware performance comparison with varying N is shown in Fig. 11. Area and energy consumption trend is similar to Fig. 9. But for computational delay, the proposed architecture cannot maintain constantly as Fig. 9(b) because it needs much time to configure the input, but still the best among the three. Distributed XIMA still achieves better performance than the other two.

VI. Conclusion

The distributed in-memory accelerator is introduced in this paper based on digitalized RRAM crossbar. A three-step RRAM-crossbar based digital matrix multiplier design is presented. Different from previous analog based RRAM crossbar, binarization matrix multiplication can be achieved in proposed architecture with small area, low computing delay and high energy-efficiency simultaneously. With numerous testing images in fingerprint matching, numerical results show that the proposed architecture has shown 2.86x faster speed, 154x better energy efficiency, and 100x smaller area when compared to the same implementation by CMOS-based ASIC. Compared to RRAM structure with analog fashion, it achieves 167x smaller area though it is not energy-efficient enough.

Acknowledgment

The work of H. Yu was supported in part by Singapore NRF-CRP Fund (NRF2011NRF-CRP002-014).

References

[1] V. Kumar and et al., "Airgap interconnects: Modeling, optimization, and benchmarking for backplane, pcb, and interposer applications," 2014.

[2] S. Park and et al., "40.4 fj/bit/mm low-swing on-chip signaling with self-resetting logic repeaters embedded within a mesh noc in 45nm soi cmos," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, 2013.

[3] S. Matsunaga and et al., "Mtj-based nonvolatile logic-in-memory circuit, future prospects and issues," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, 2009.

[4] H. Akinaga and H. Shima, "Resistive random access memory (reram) based on metal oxides," *Proceedings of the IEEE*, 2010.

[5] K.-H. Kim and et al., "A functional hybrid memristor crossbar-array/cmos system for data storage and neuromorphic applications," *Nano letters*, vol. 12, no. 1, pp. 389–395, 2011.

[6] X. Liu and et al., "Reno: a high-efficient reconfigurable neuromorphic computing accelerator design," in *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*. IEEE, 2015, pp. 1–6.

[7] Y. Kim and et al., "A digital neuromorphic vlsi architecture with memristor crossbar synaptic array for machine learning," in *SOC Conference (SOCC)*, 2012.

[8] W. Lu, K.-H. Kim, T. Chang, and S. Gaba, "Two-terminal resistive switches (memristors) for memory and logic applications," in *Design Automation Conference (ASP-DAC)*, 2011.

[9] C. Liu and et al., "A spiking neuromorphic design with resistive crossbar," in *Proceedings of the 52nd Annual Design Automation Conference*. ACM, 2015, p. 14.

[10] J. Wright and et al, "Robust face recognition via sparse representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 2, pp. 210–227, 2009.

[11] H. Lee and et al., "Low power and high speed bipolar switching with a thin reactive ti buffer layer in robust hfo2 based rram," in *Electron Devices Meeting,*. IEEE, 2008.

[12] P. Singh and et al., "20mw, 125 msps, 10 bit pipelined adc in 65nm standard digital cmos process," in *Custom Integrated Circuits Conference (CICC)*, 2007.

[13] T. Tan and Z. Sun, "CASIA-FingerprintV5," 2010. [Online]. Available: http://biometrics.idealtest.org/