NANYANG TECHNOLOGICAL UNIVERSITY

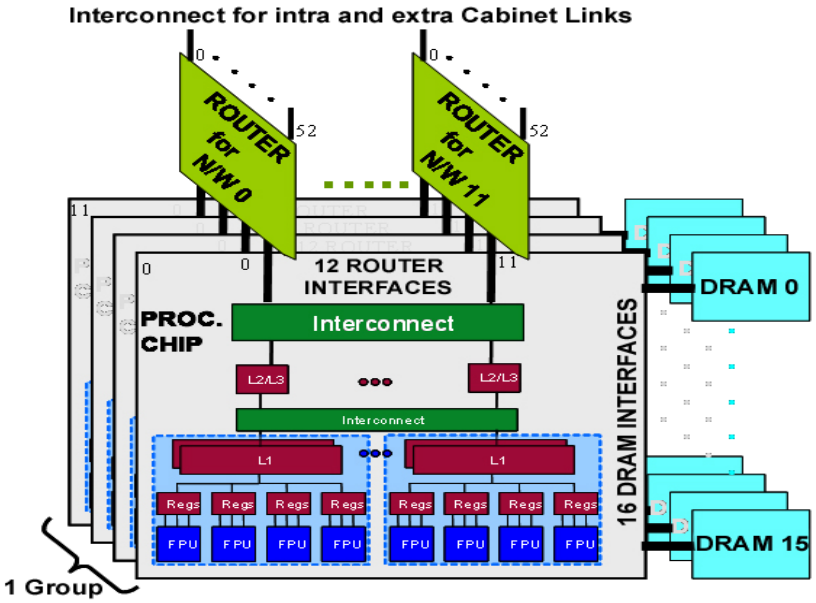
An Energy-efficient Matrix Multiplication Accelerator by Distributed In-memory Computing on Binary RRAM Crossbar

Leibin Ni[1], Yuhao Wang[1], **Hao Yu[1]**, Wei Yang[2], Chuliang Weng[2], Junfeng Zhao[2]

[1]School of EEE, Nanyang Technological University, Singapore

[2]Shannon Laboratory, Huawei Technologies Co., Ltd, China

Data Analytics Challenge



Data center for future big-data-oriented society:

1. Leaving data outside a nation will face serious cyber-security concern
2. Processing data inside a nation by traditional Giga-scale system has high cost

Bandwidth at 100 Gps with Space of 20,000 sq. ft. Power of 68 MW and cost of 100M-USD !!!

Challenge of Data Oriented Computing

Data migration between core and memory is dominated in big-data application: Memory wall = power wall + bandwidth wall

1.Bandwidth wall:

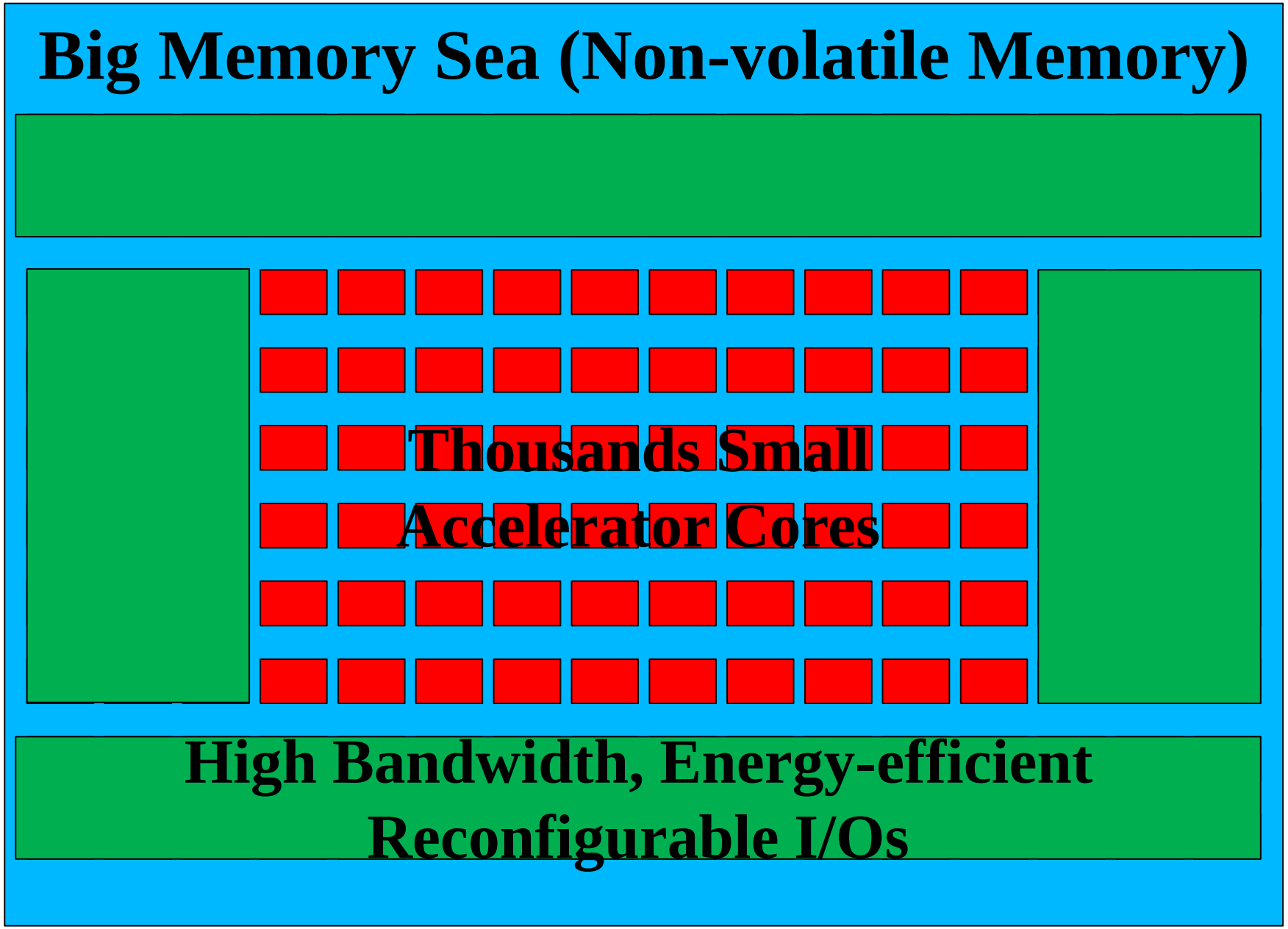
- Technology limitation: non-scalable 2D interconnect
- System limitation: Configurability

2. Power wall:

- Technology limitation: leakage power of DRAM/SRAM
- System limitation: Low-complexity in-memory analytics

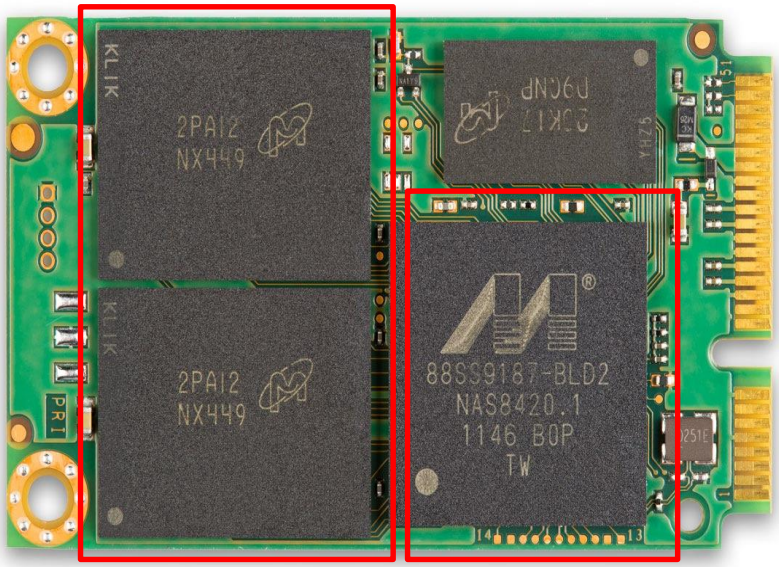
A solution with system and technology co-design

Solution?


Big Memory Sea (Non-volatile Memory)
Thousands Small Accelerator Cores
High Bandwidth, Energy-efficient Reconfigurable I/Os

Existing In-memory Computing Techniques

PCB level in-memory accelerators



Flash chips

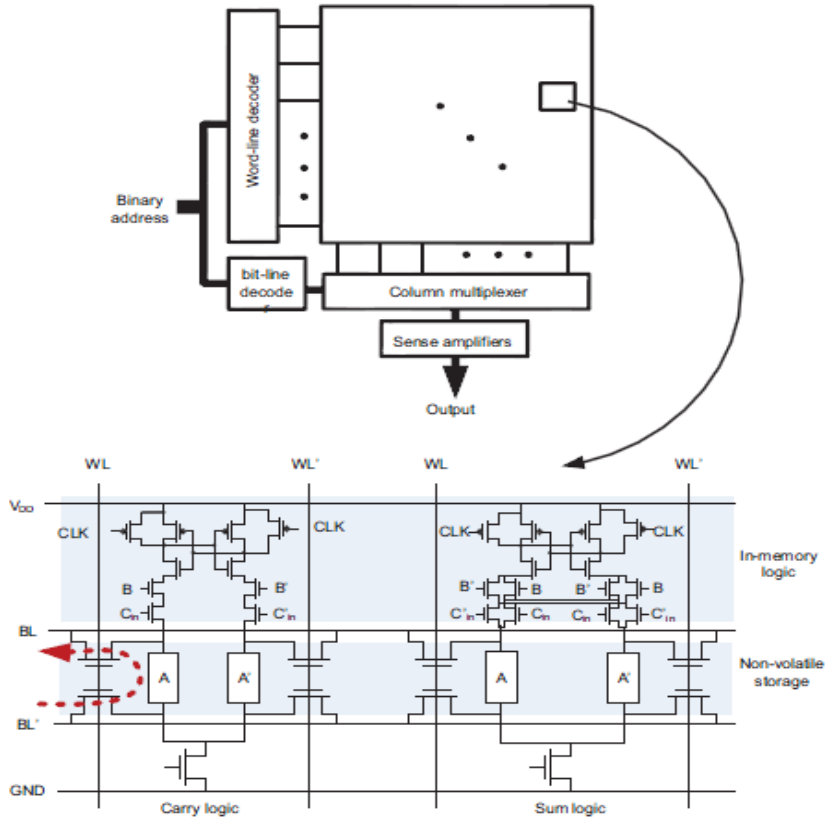
Control & in-memory logic

Micron M500 SSD - Micron Technology, Inc.
http://www.micron.com/products/solid-state-storage/client-ssd/m500-ssd

- Still limited I/O between memory/storage to in-memory logic (up to 256/512 due to packaging limitation)
  - PCB level interconnect has high power overhead: **30pJ/bit/cm** compared to on-chip **40fJ/bit/mm**
  - Long interconnect trail incurs longer RC delay on PCB which impacts on the throughput.
- In-memory logic implemented in CMOS technology, which incurs large leakage power

Existing In-memory Computing Techniques

Cell-level in-memory accelerators with fixed simple function



Matsunaga, Shoun, et al. "MTJ-based nonvolatile logic-in-memory circuit, future prospects and issues." *Proceedings of the Conference on Design, Automation and Test in Europe*. European Design and Automation Association, 2009.

Features:

- In each cell simple logic is implemented
- Read out results directly

Drawbacks:

- Cell structure becomes much more complicated thus area overhead is overwhelming
- Low utilization efficiency of in-memory logic since they cannot be shared (dedicated to their cells only)
- Simple functions only (limited by circuit area for each cell)

Non-volatile In-Memory Accelerator

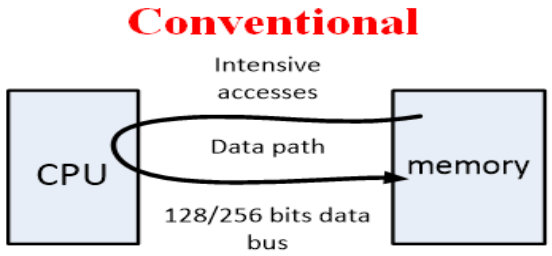
Motivation: Large portion of matrix multiplication is needed in compressive neural network machine learning.

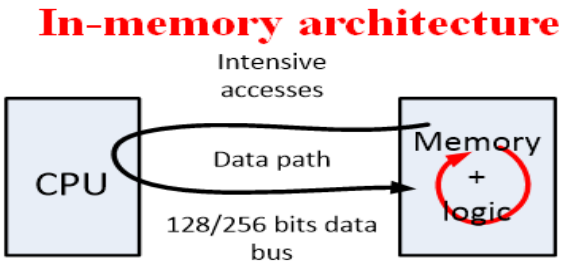
- 64.62% of ELM training time are matrix multiplication.
- Compressive sensing $F = \Psi X$ is fully based on matrix multiplication.

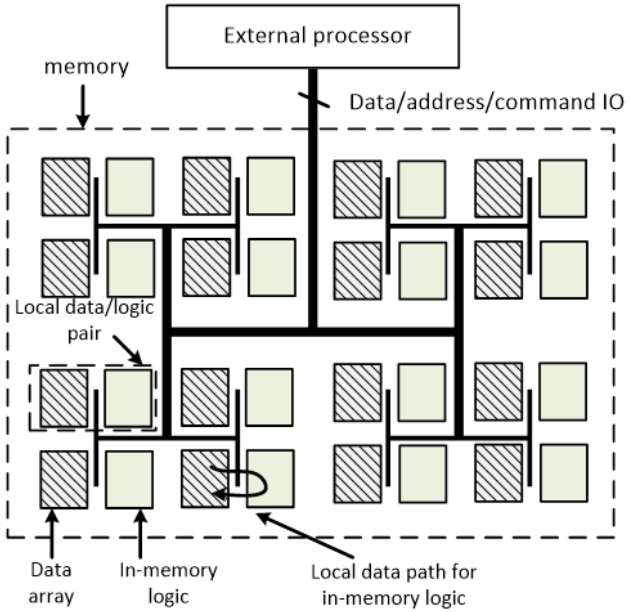
Operation in accelerator: All matrix multiplication operation including $XA, H^T T$ in ELM, and $F = \widehat{\Psi} X$ in compressive sensing.

Why RRAM but not CMOS (GPU or ASIC):

- Logic operation can be done faster with less dynamic power consumption.
- Eliminate static power with non-volatile memory.

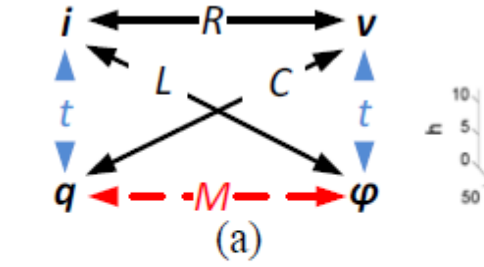




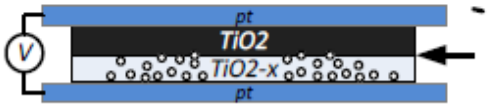


RRAM and RRAM Crossbar (Memristor)

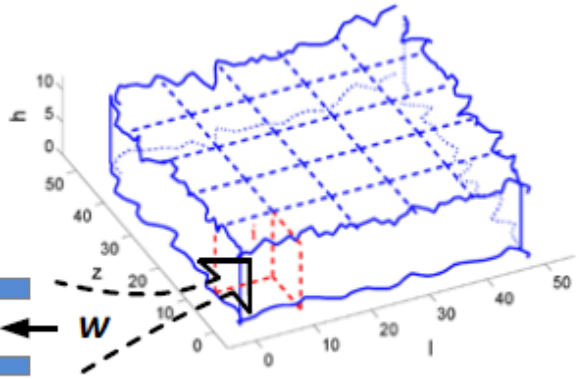
- **Resistive RAM (RRAM) device**
  - Two-terminal non-volatile memory device
  - Doping wall that separates high resistance material and low resistance material
  - Doping wall can be moved by voltage/current a program overall resistance
- **Crossbar structure**
  - Horizontal wires (top layer) for input
  - Vertical wires (bottom layer) for output
  - RRAM (middle layer) at cross-points resistances
- **Intrinsic implementation for matrix-vector multiplication**



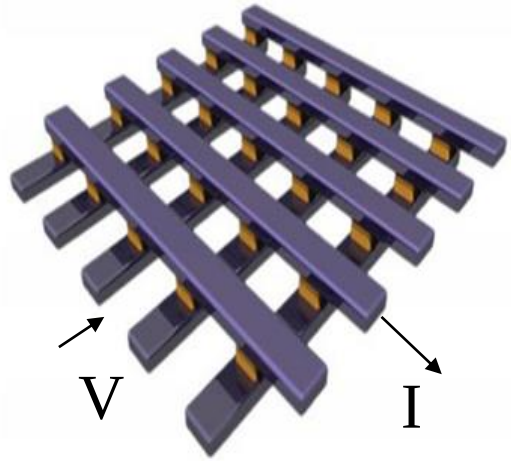
(a)



(b)



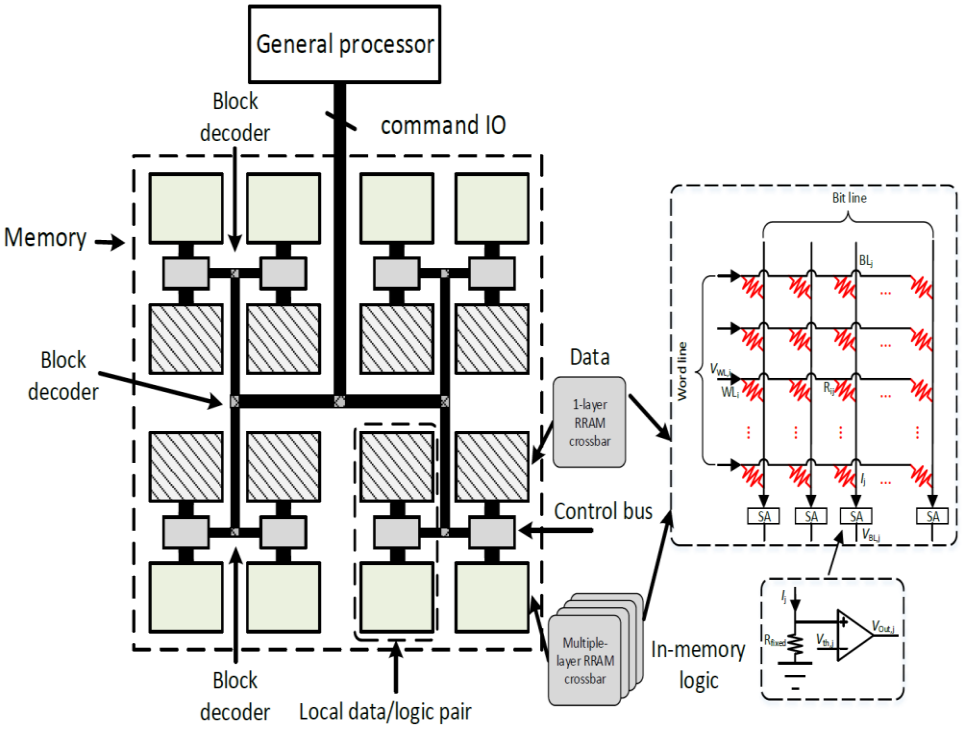
(c)



$$\begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_m \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} & \cdots & G_{1n} \\ G_{21} & G_{22} & \cdots & G_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ G_{m1} & G_{m2} & \cdots & G_{mn} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix}$$

RRAM-crossbar based In-memory Accelerator (XIMA)

- RRAM crossbar enables in-memory logic linked by distributed control-bus in pairs
- Digitization at output with paralleled comparators
- With additional encoder/decoder, one can develop matrix-vector multiplication based on distributed in-memory accelerator (XIMA)



$$V_{out} = \begin{cases} 1, & IR_{ref} \geq V_{th} \\ 0, & IR_{ref} < V_{th} \end{cases}$$

Binary interface

Communication Protocol

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

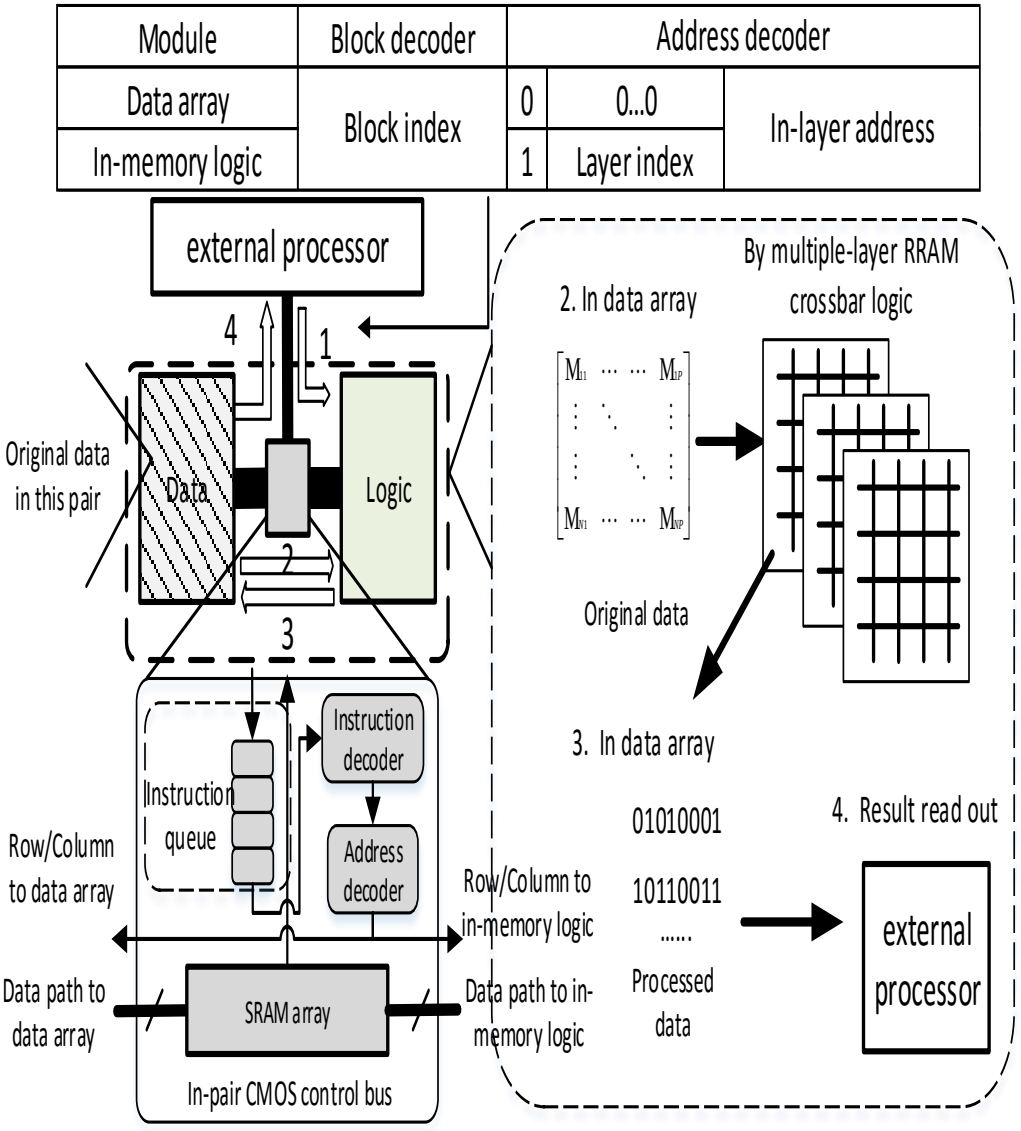
Traditional read/write

Additional in-memory computing

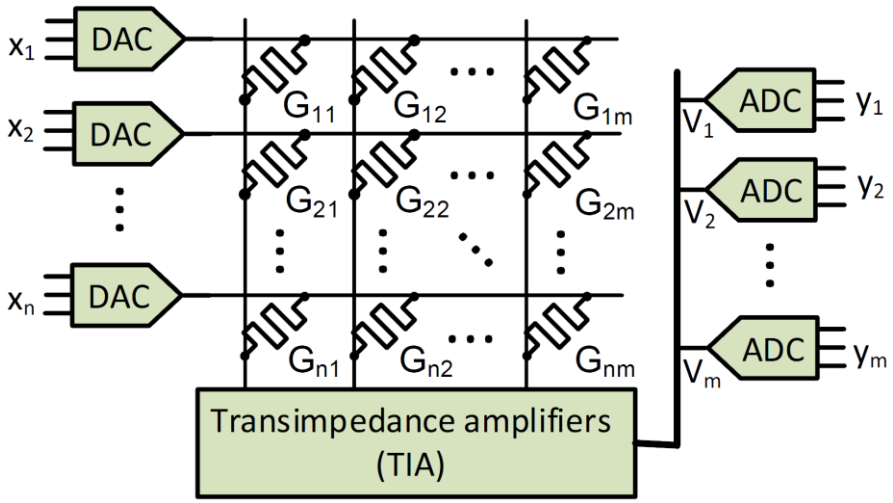
- SW (store word) instruction is to write data into RRAMs in data (conventional write) or in-memory logic (logic configuration).
- LW (load word) instruction performs as conventional read operation.
- ST (start) instruction means to switch on the logic block for computing after computation setup.
- WT (wait) operation is to stop reading from instruction queue during computing.

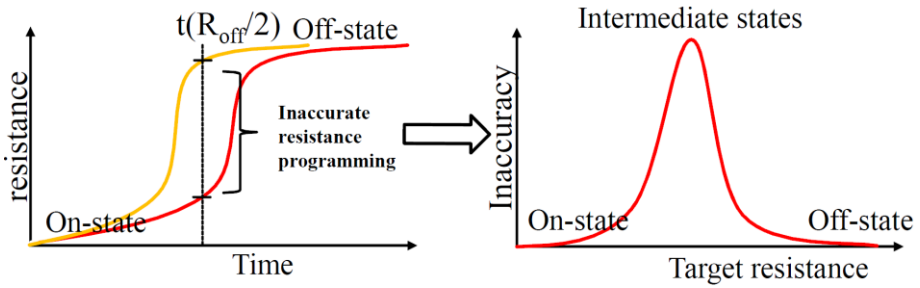
Control-bus Design

- Instruction queue: store instructions issued by external processor.
- Instruction decoder: analyze instructions on a first-come-first-serve (FCFS) basis.
- Address decoder: obtain the row and column index from the instruction.
- SRAM array: store temporary data such as computation results, which are later written back to data array.



Analog RRAM-crossbar





Switching curve

Programing inaccuracy

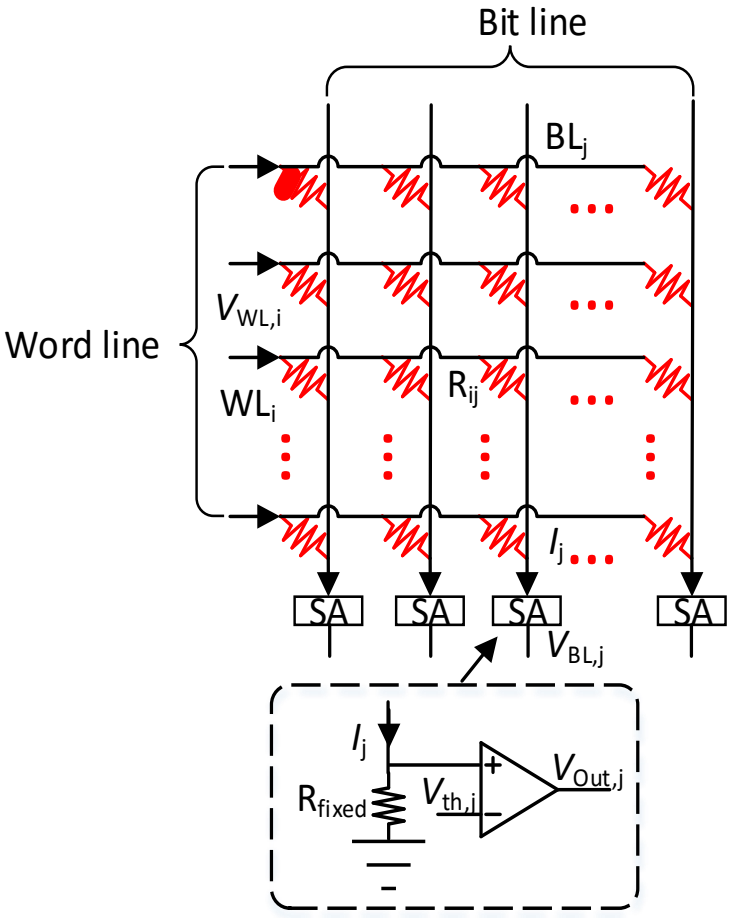
Features:

- Each RRAM denotes a multi-bit value with different resistance.
- DACs/ADCs are designed for I/O.
- Multiple bit result can be read directly.

Drawbacks:

- It is hard to configure an accurate resistance of intermediate states, causing large error.
- DACs/ADCs consumes too much power.

Proposed Binary RRAM-crossbar



$$O_j = \begin{cases} 1, if\ V_j^{BL} \geq V_{th,j} \\ 0, if\ V_j^{BL} < V_{th,j} \end{cases}$$

Features:

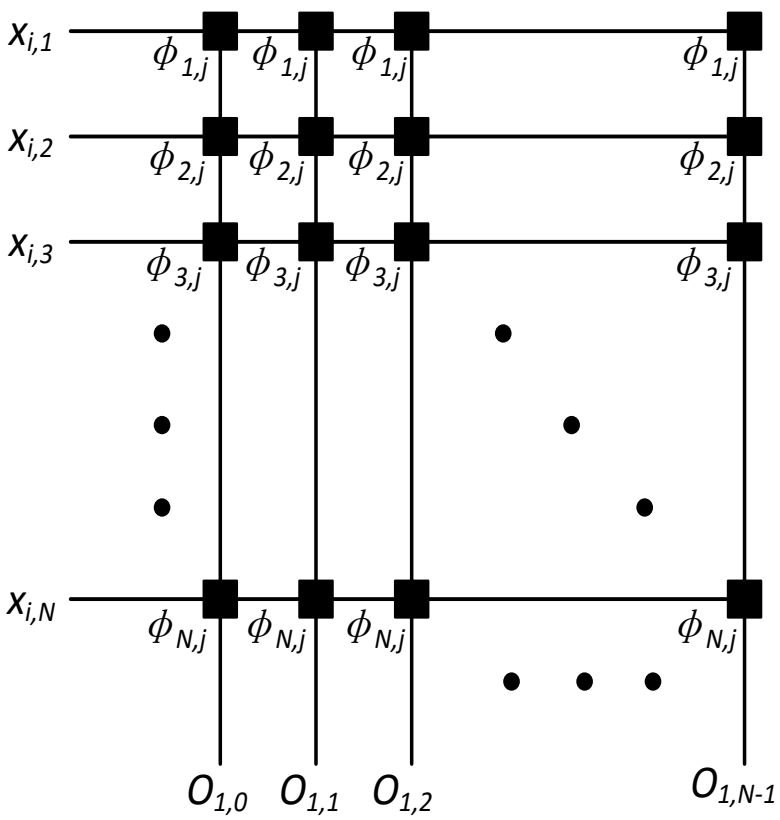
- Each RRAM denotes only 1 or 0.
- Use sense amplifier in each bit-line.
- Output of each bit-line is only 1 or 0.
- Threshold of sense amplifier $V_{th,j}$ can be configured.

Merits:

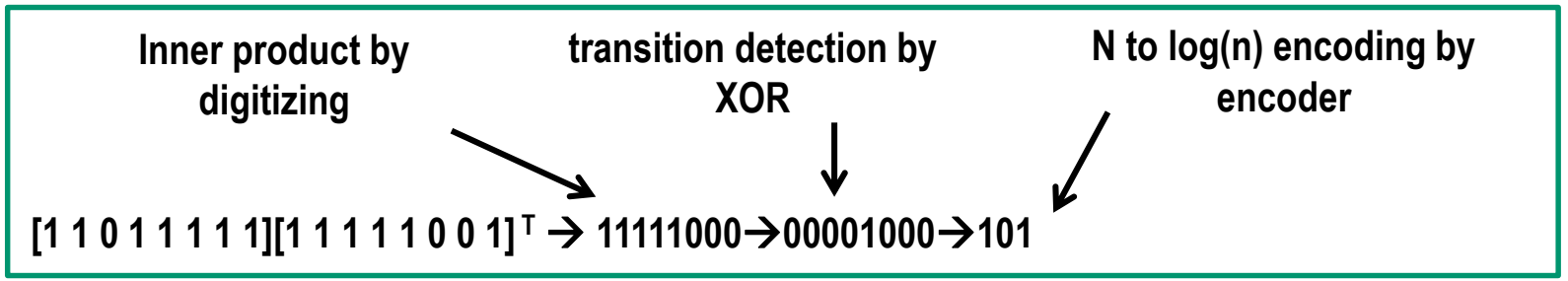
- RRAM can be configured accurately.
- Output result can be accurate without ADCs

XIMA Matrix-Vector Multiply Accelerator

- **Target**: matrix-vector multiplication with binary RRAM crossbar interface
- **Approach**: utilize 3-step digital RRAM crossbar instead of analogous crossbar
- Size of RRAM crossbar is $N \times N$, where N is the maximum possible result.



Overall flow



XIMA Matrix-Vector Multiply Accelerator I

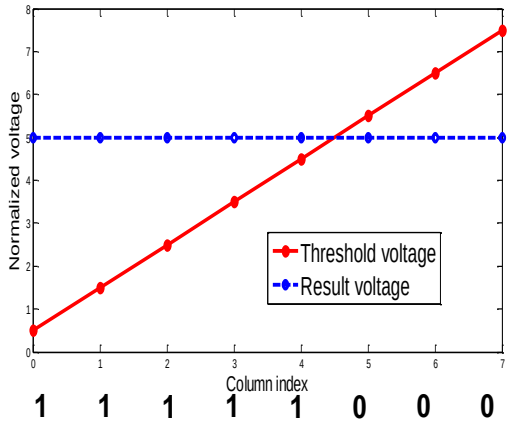
❑Step 1: Digitizing

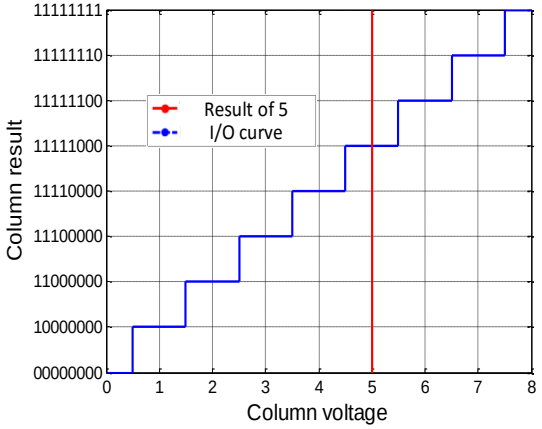
- Identical columns but with different readout threshold V_{th}.
- Ladder-like threshold voltages V_{th} to identify the result:

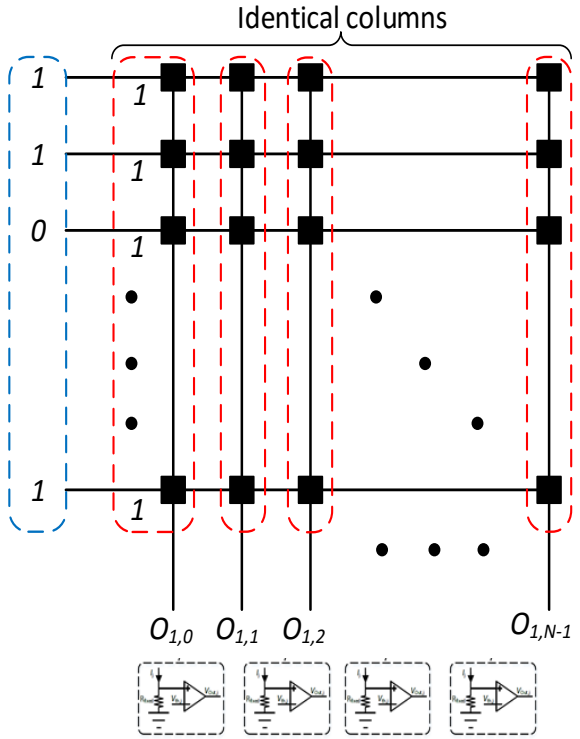
$$V_{th,k} = \frac{(2k+1)V_r g_{on} R_s}{2}$$

$$O_{1,k} = \begin{cases} 1, k \le s \\ 0, k > s \end{cases}$$

Important Property:
For the inner-product result s, the first s bits are 1 and the rest $(N - s)$ bits are 0 $(s <= N)$.







For example, the output that corresponds to 5 is 11111000 $(N = 8)$.

XIMA Matrix-Vector Multiply Accelerator II

❑Step 2: XOR

- The result is indicated by transition bit

111110000

↑

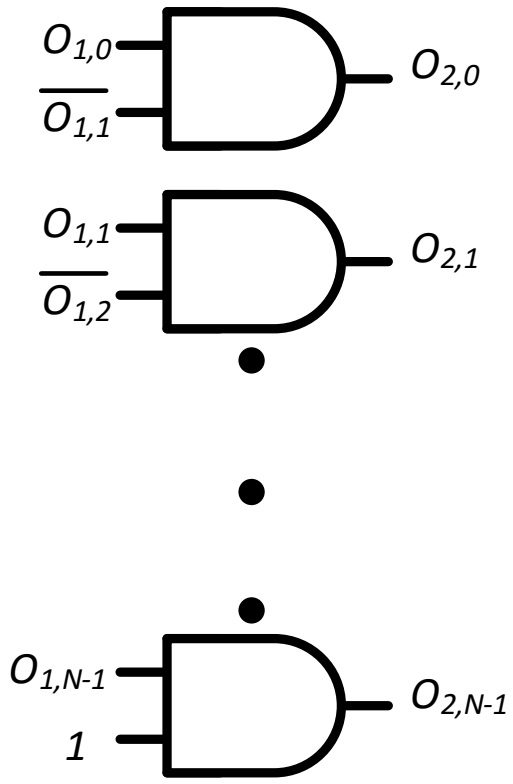
Fifth bit

- Detect the transition bit by XOR (⊕) adjacent bits

First bit **inverse** + Second bit

result	0	1	0
First bit	1	1	0
Second bit	1	0	0

→ Transition occurs when there is no current!



Implemented by AND gate.

For example, the output that corresponds to 11111000 is 00001000 $(N = 8)$.

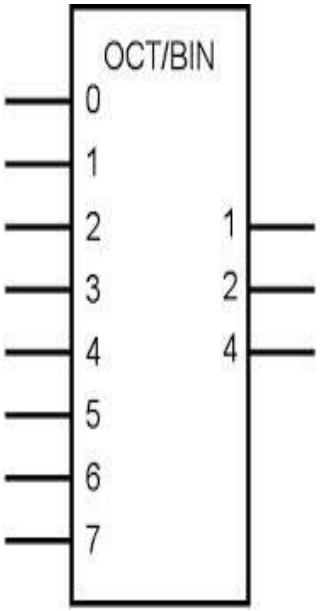
XIMA Matrix-Vector Multiply Accelerator III

❑Step 3: Encoder

- Pre-configured binary in each row
- Only one row can be active due to transition output of XOR layer
- Row-select to encode then transition to binary

❑ Second step output:
- High at the transition
- low for the rest

8-3 encoder



Encoding by row selection

Step 2 output	Step 3 output
10000000	001
01000000	010
00100000	011
00010000	100
00001000	101
00000100	110
00000010	111
00000001	000

Look-up table of 8-3 encoder

For example, the output that corresponds to 00001000 is 101 ($N = 8$).

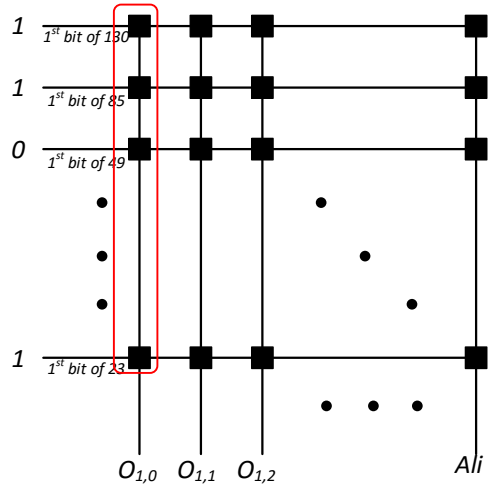
Extension for Real-value Multiplication

- **Adders are needed in CMOS-ASIC.**
  - Apply 3-step binary multiplication for the same bit of 16 data.
  - Only need 3x12-bit adders instead of 15.
- **For N-dimensional data, reduce from (N-1) adders to** $(\log_2 N - 1)$

$$[130, 85, 49, \ldots 23] * [1,1,0, \ldots, 1]^T$$

8-bit data * 16 | 8-bit data * 16

(130) 10000010
(85) 01010101
(49) 00110001
⋮
(23) 00010111



- **Result of 1st bit:**

$$[0, 1, 1, \ldots 1] * [1,1,0, \ldots, 1]^T = \mathbf{1101(13)}$$

1 1 0 1 result of the 1st bit
1 0 1 0 result of the 2nd bit
0 1 1 0 result of the 3rd bit

\+ 0 1 1 0 result of the 8th bit

inner-product result

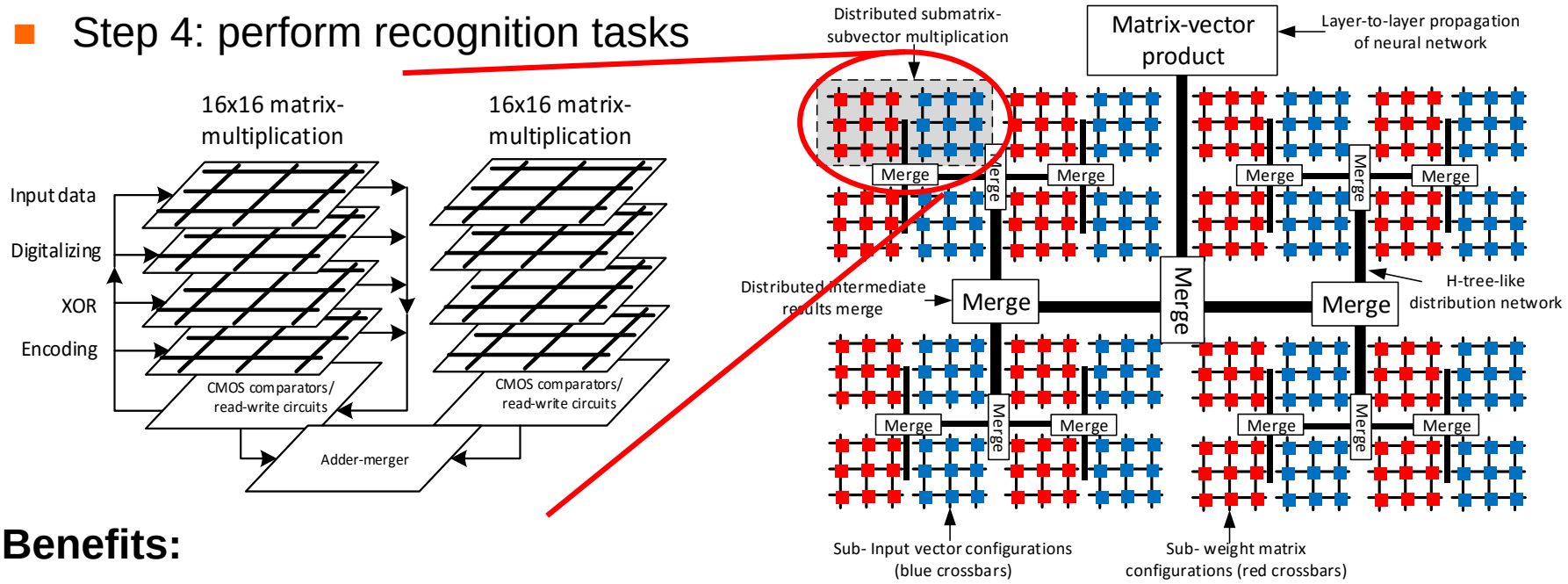
5th bit 1 0 0 1 1 1 0 1 1st bit
6th bit 1 1 0 0 1 0 1 0 2nd bit
7th bit 1 1 1 0 0 1 1 0 3rd bit
\+ 8th bit 0 1 1 0 0 1 1 1 4th bit

inner-product result

XIMA for Machine Learning

Working flow:

- Step 1: offline unsupervised training for Boolean first neural layer Ψ
- Step 2: configure Ψ distributedly to 3-step digitalized RRAM by stacking
- Step 3: online ASIC incremental LSM training for second neural layer
- Step 4: perform recognition tasks



Benefits:

- 1. distributed in-memory computing → high throughput
- 2. RRAM decomposition → high fabrication feasibility
- 3. Boolean Ψ optimization → crossbar compatibility with high performance
- 4. non-volatile RRAM → low static power

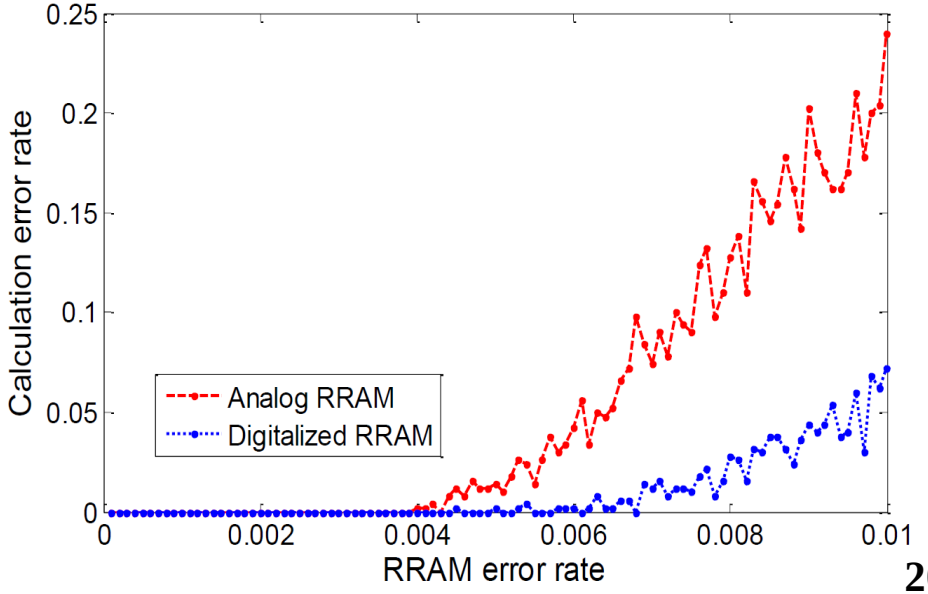
Variability Study

Implementation	General purpose processor (MatLab)	CMOS ASIC	Non-distributed digitalized RRAM crossbar	Distributed digitalized RRAM crossbar	Distributed analog RRAM crossbar
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: $0.131\mu J$

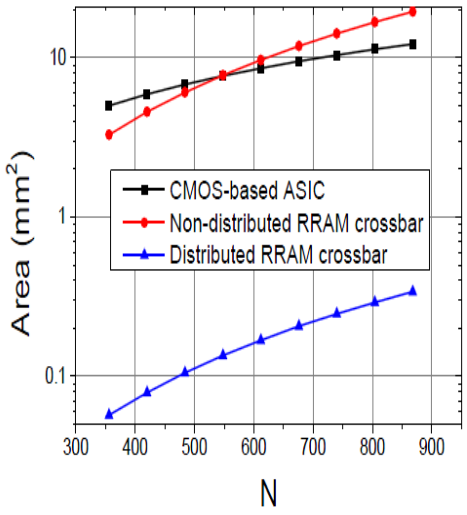
- **Experimental Setup: 328x356 * 356x64 Boolean Matrix**

- **Compare to CMOS-ASIC**
  - 2.86x faster
  - 154x more energy efficient
  - 100x smaller area

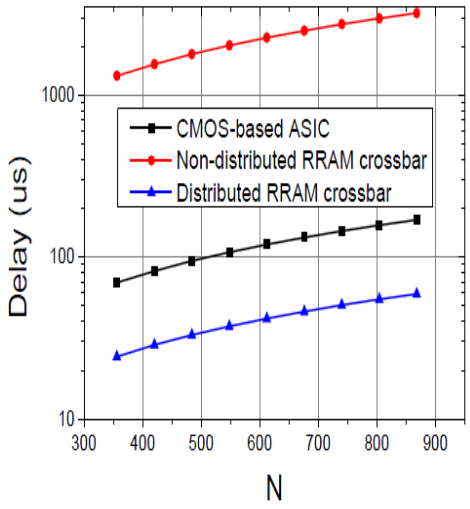
- **Compare to analog RRAM crossbar**
  - Almost the same computational time
  - 10x worse energy efficient
  - 160x smaller area



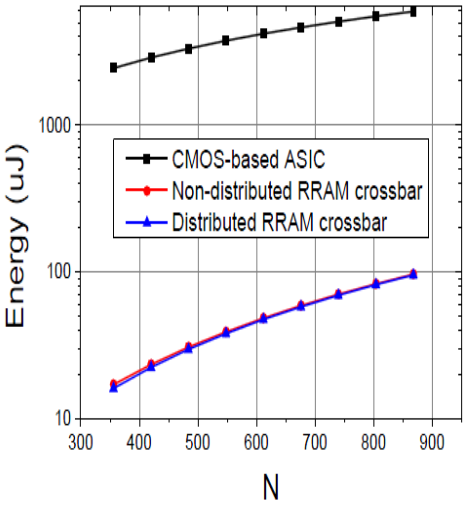
Scalability Study



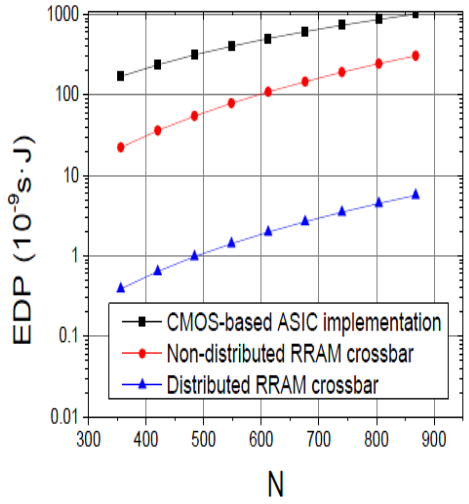
(a)



(b)



(c)



(d)

- We compare the performance among CMOS-ASIC design, non-distributed RRAM-crossbar design, and distributed RRAM-crossbar design.
- X-axis is the original image dimension (number of pixels).
- Distributed RRAM-crossbar is the best among three implementations.
- The EDP improvement compared to CMOS-ASIC reduces from 514x to 196x with dimension increasing from 356 to 864, but still considerate.

Face-recognition Simulation Results

- **Hardware performance**
  - 15x faster (due to one cycle)
  - 17x more energy efficient with leakage elimination
  - More than 100x smaller area

- **Conclusion**: the RRAM crossbar with proposed Boolean optimization overcomes the power-performance dilemma and enables both
  - high performance first layer dimension reduction/feature extraction (less info loss),
  - Power- and area-efficient hardware.

Embedding configuration		Energy (nJ)	Leakage (uW)	Area (um^2)
25×64	CMOS ASIC	5.6700	59	86650
	RRAM crossbar	0.3324	-	54
28×64	CMOS ASIC	6.3504	66	97050
	RRAM crossbar	0.3741	-	60
30×64	CMOS ASIC	6.8040	71	103980
	RRAM crossbar	0.3993	-	65
34×64	CMOS ASIC	7.7112	81	117850
	RRAM crossbar	0.4558	-	74
39×64	CMOS ASIC	8.8452	114	135180
	RRAM crossbar	0.5228	-	84

Thank You!

http://www.ntucmosetgp.net
Email: haoyu@ntu.edu.sg
Skype: hao.yu.ntu