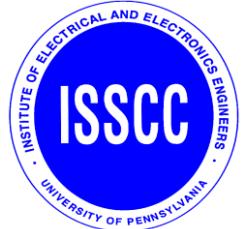




**ISSCC 2024**

**SESSION 34**  
**Compute-In-Memory**



# A 28nm 83.23TFLOPS/W POSIT-Based Compute-in-Memory Macro for High-Accuracy AI Applications

Yang Wang<sup>1</sup>, Xiaolong Yang<sup>1</sup>, Yubin Qin<sup>1</sup>, Zhiren Zhao<sup>1</sup>,  
Ruiqi Guo<sup>1</sup>, Zhiheng Yue<sup>1</sup>, Huiming Han<sup>1</sup>, Shaojun Wei<sup>1</sup>, Yang Hu<sup>1</sup>, Shouyi Yin<sup>1,2</sup>

<sup>1</sup>Tsinghua University, Beijing, China

<sup>2</sup>Shanghai AI Laboratory, Shanghai, China

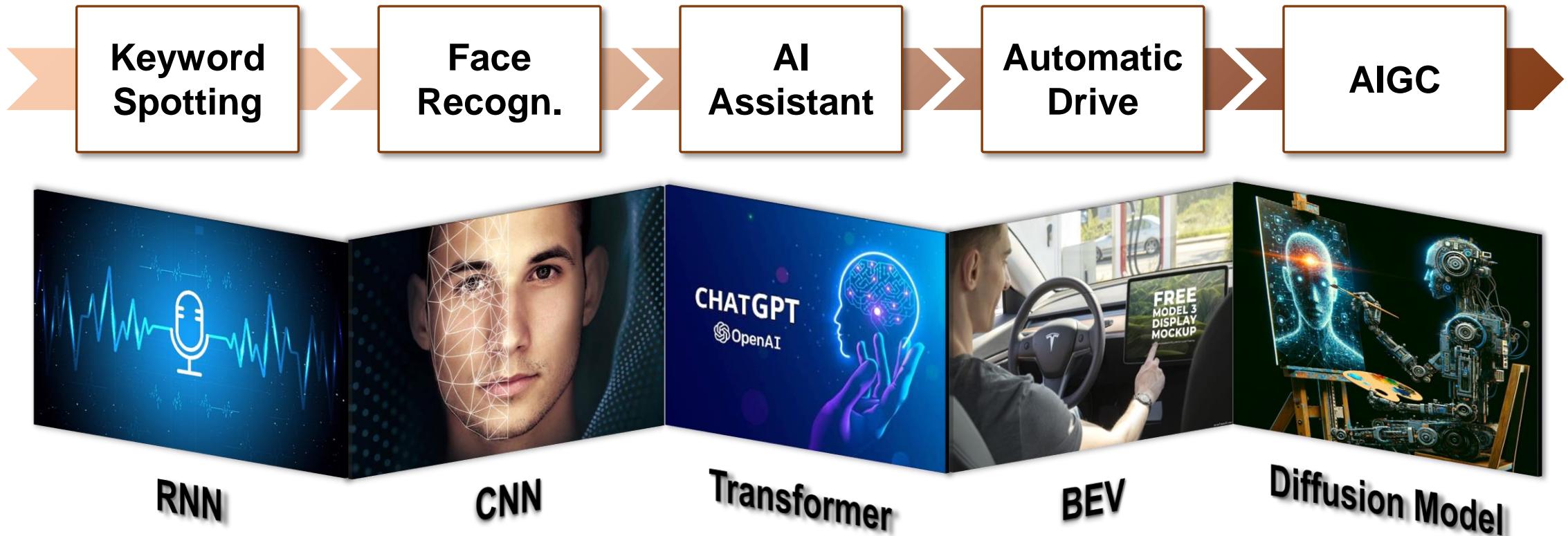


上海人工智能实验室  
Shanghai Artificial Intelligence Laboratory

# Outline

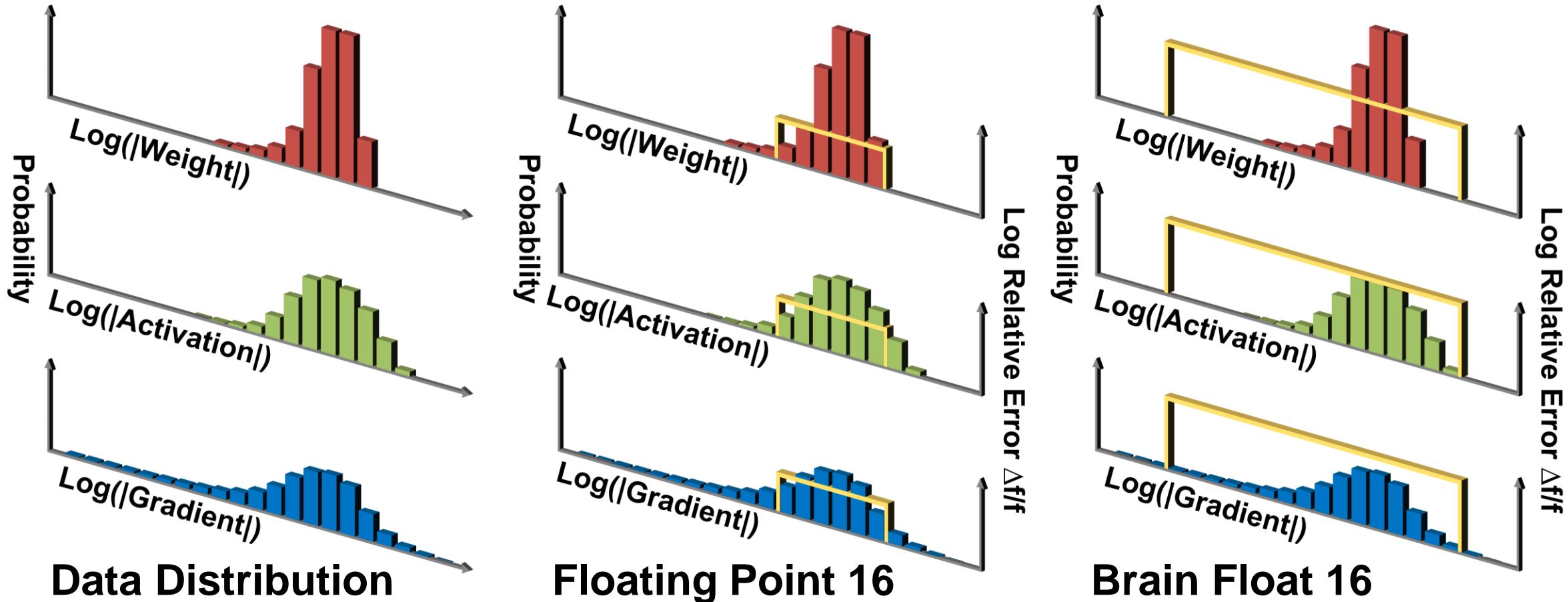
- **Background and Motivation**
- Challenges of POSIT-Based CIM Macro
- Proposed POSIT@CIM Macro Features
  - Bi-directional Regime Processing Codec
  - Critical-bit Pre-compute-and-store CIM Array
  - Cyclically-alternating Scheduling Adder Tree
- Measurement and Comparison
- Conclusion

# FP-CIM for High-accuracy AI Applications



- Recent AI tasks are becoming **increasingly complex**.
- Complex AI application requires **FP-CIM for high accuracy**.

# Limitation of Conventional FP Data Format



■ Conventional FP cannot achieve **high accuracy with low power**.

# Principle of POSIT Data Format

## Posit Data Format

S	Regime(R)	Exponent(E)	Mantissa(M)
---	-----------	-------------	-------------

*(r bits)  
Dynamic*      *(es bits)  
Pre-set*      *(n-r-es-1 bits)  
Dynamic*

$$\text{value} = (-1)^{\text{sign}} \times (2^K)^R \times 2^E \times 1.M, K=2^{\text{es}}$$

**Regime(R): unary (thermometer) code with successive 0s and 1s**

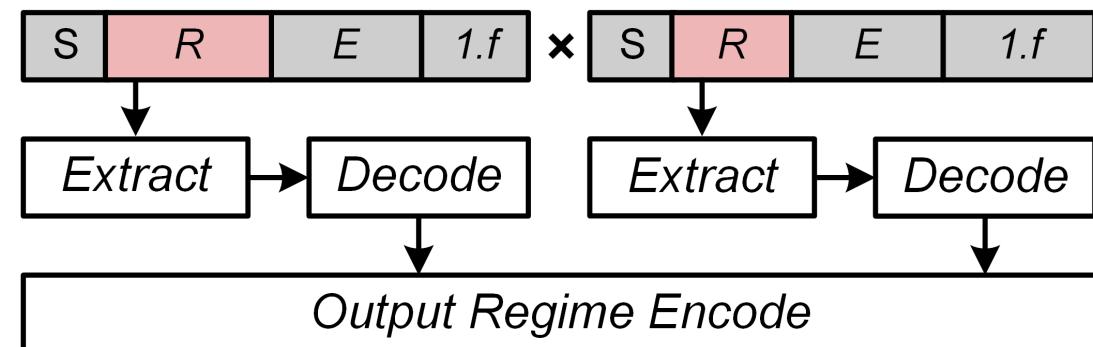
Regime(R) {  
    1111X positive R, 1's count - 1  
    0000X negative R, 0's count

## Example: POSIT(8,2), where es=2

1	1110	01	1
---	------	----	---

$$K=2^{\text{es}}=2^2=4 \quad R=\text{Regime}(1110)=(3-1)=2$$
$$E=\text{Exponent}(10)=1 \quad M=\text{Mantissa}(1.1)=1.5$$

$$\text{value} = -1 \times (2^4)^2 \times 2^1 \times 1.5 = -1.5 \times 2^9$$

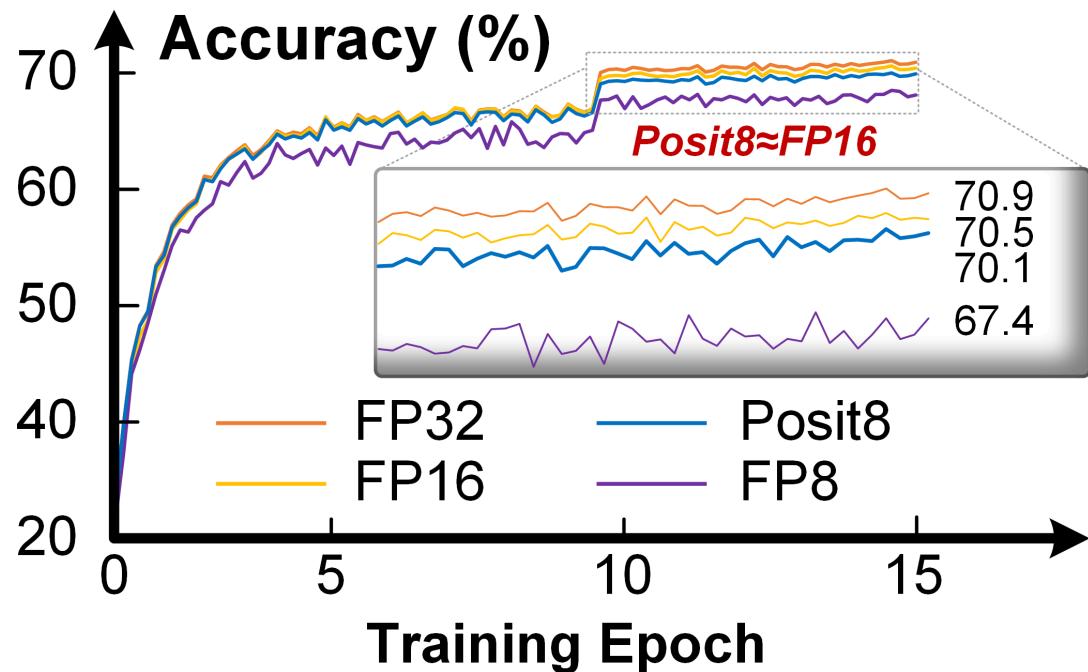


- POSIT exploits **dynamic bit** to adapt to varied distributions.

# Conventional FP VS. POSIT

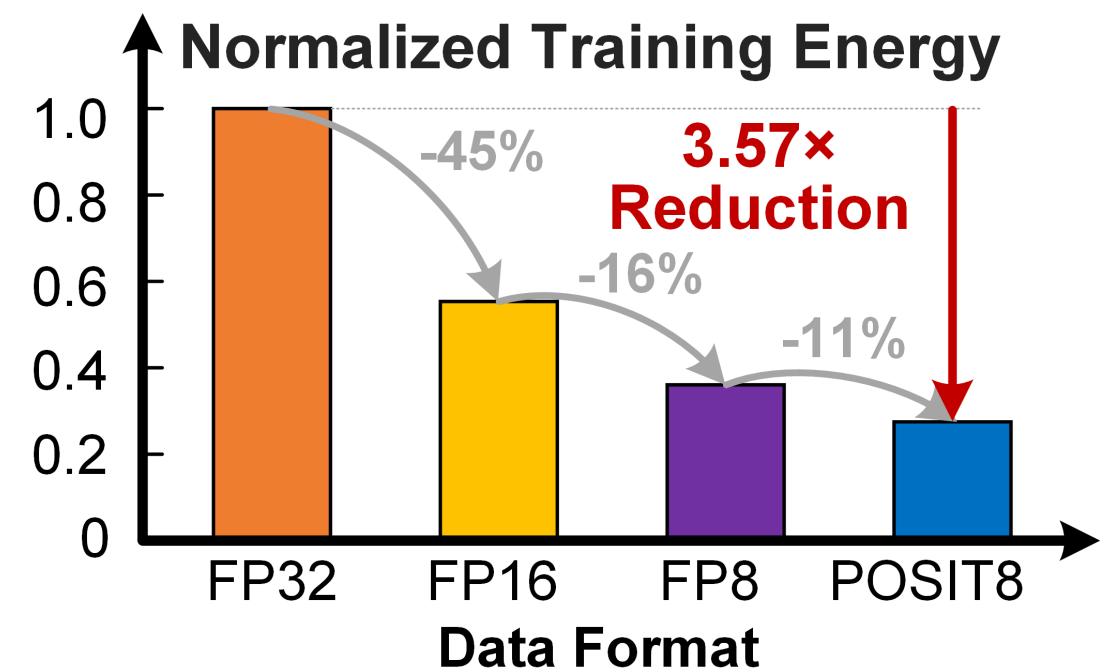
## Training with Different Data Formats

(ImageNet on ResNet18)



## Training Energy Comparison

(Achieving Same Accuracy)

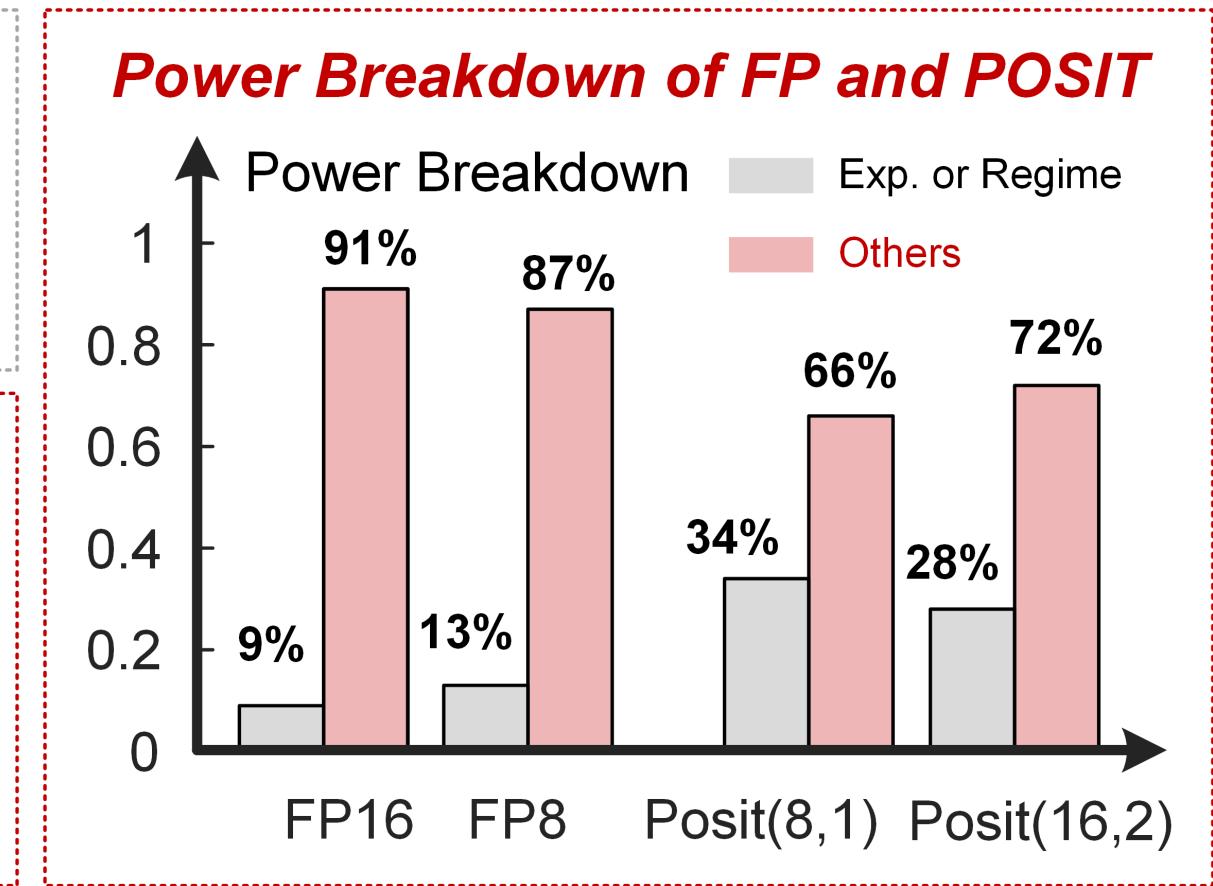
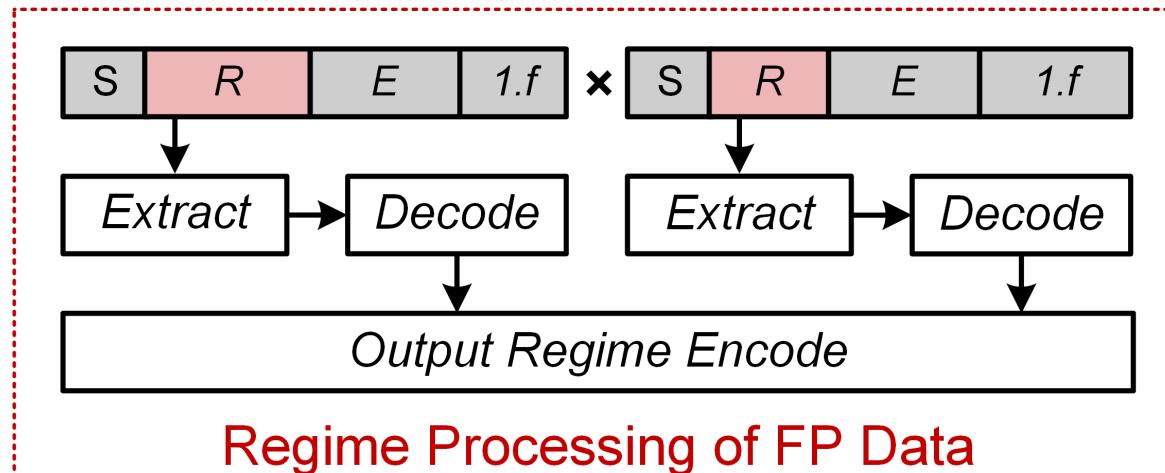
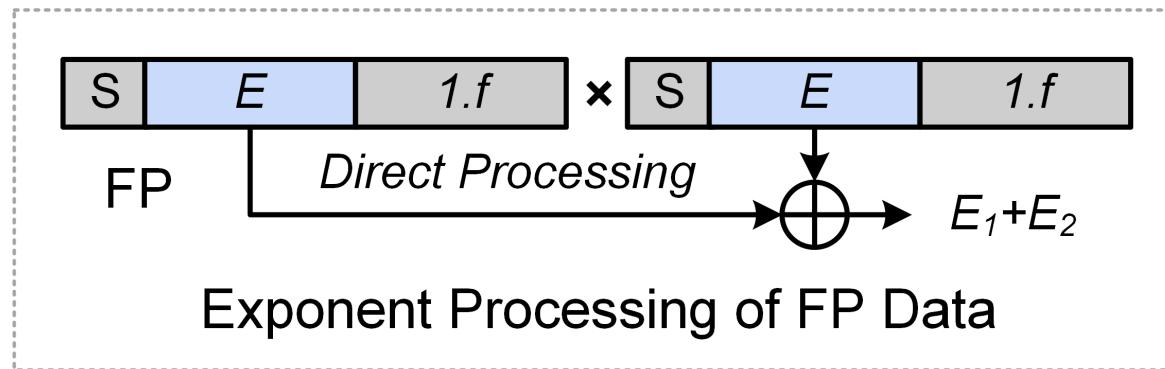


- POSIT8 saves **27% energy** with **0.4% accuracy loss** than FP16.

# Outline

- **Background and Motivation**
- **Challenges of POSIT-Based CIM Macro**
- **Proposed POSIT@CIM Macro Features**
  - Bi-directional Regime Processing Codec
  - Critical-bit Pre-compute-and-store CIM Array
  - Cyclically-alternating Scheduling Adder Tree
- **Measurement and Comparison**
- **Conclusion**

# Challenge 1: Large Power in Regime Processing

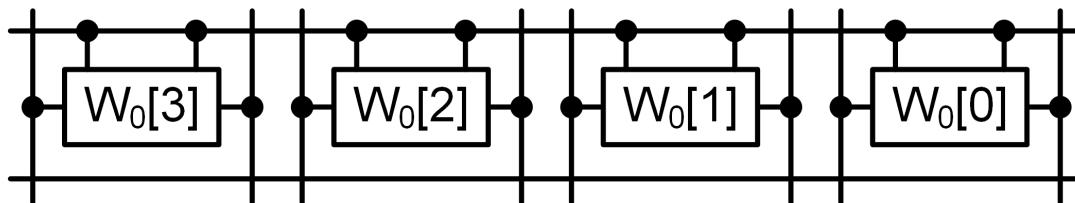


- Dynamic regime increases **2.62 × pre-processing energy**.

# Challenge 2: Cell Under-utilization in CIM Array



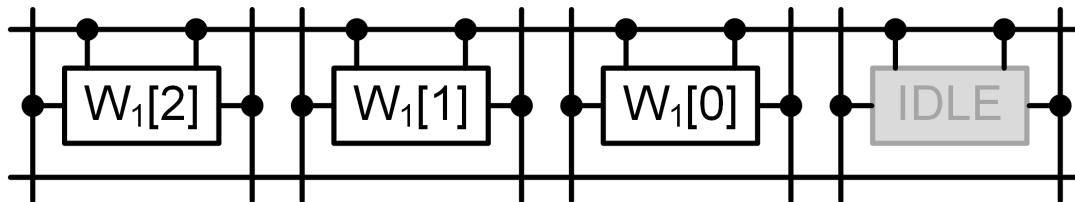
*4 bits Mant. of  $W_0$*



*Full CIM Cells utilization*

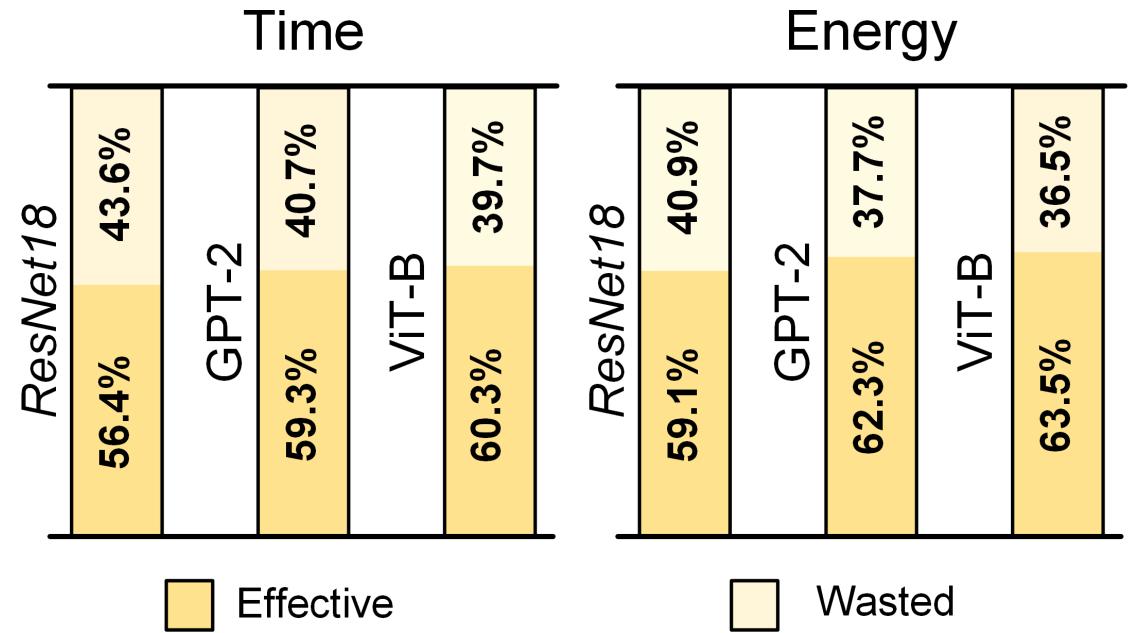


*3 bits Mant. of  $W_1$*



*CIM Cells Under-utilization*

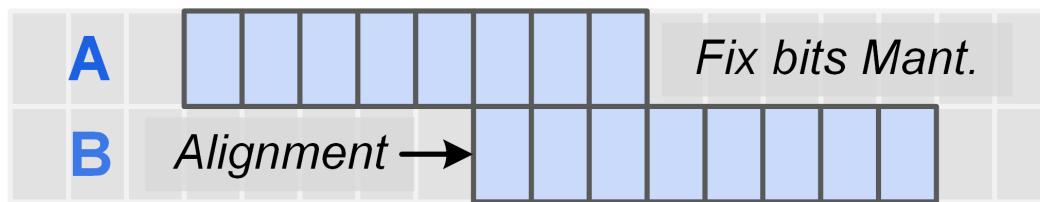
*Time and Energy Waste due to Cell Under-utilization*



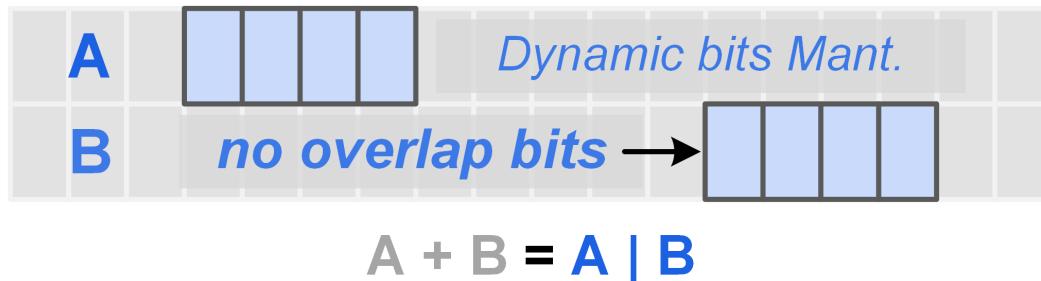
- Dynamic mantissa introduces **41.3% CIM cell underutilization**.

# Challenge 3: Redundant Toggle in Adder Tree

FP: Dynamic Align. + Fixed Bit-width



Posit: Dynamic Align. + Dynamic Bits



## Redundant Logic Toggle Power Consumption in Adder Tree

174.38nW     $A + B = A | B$     29.75nW  
(16b Simulation@400MHz, 0.9V)

Models	Adder Tree Energy		Ratio
	Total	Redun.	
ResNet18	0.11mJ	0.083mJ	76.3%
GPT-2	5.8mJ	3.1mJ	53.5%
ViT-B	1.1mJ	0.63mJ	57.6%

- Dynamic aligned accumulation incurs **66.8% power waste**.

# Outline

- Background and Motivation
- Challenges of POSIT-Based CIM Macro
- Proposed POSIT@CIM Macro Features
  - Bi-directional Regime Processing Codec
  - Critical-bit Pre-compute-and-store CIM Array
  - Cyclically-alternating Scheduling Adder Tree
- Measurement and Comparison
- Conclusion

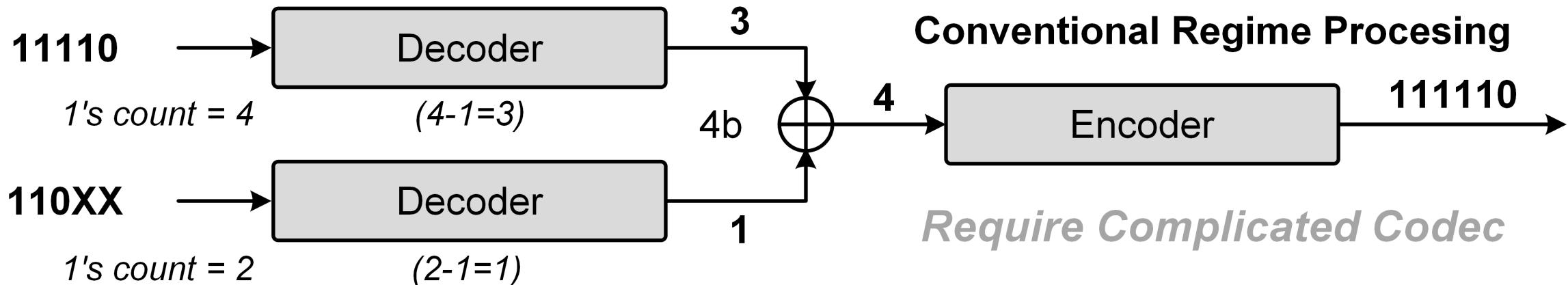
# Feature 1: Bi-directional Regime Processing

Binary	00001	0001x	001xx	.....	110xx	1110x	11110
Regime	-4	-3	-2	.....	1	2	3

0's count for neg. R (0001 for -3)

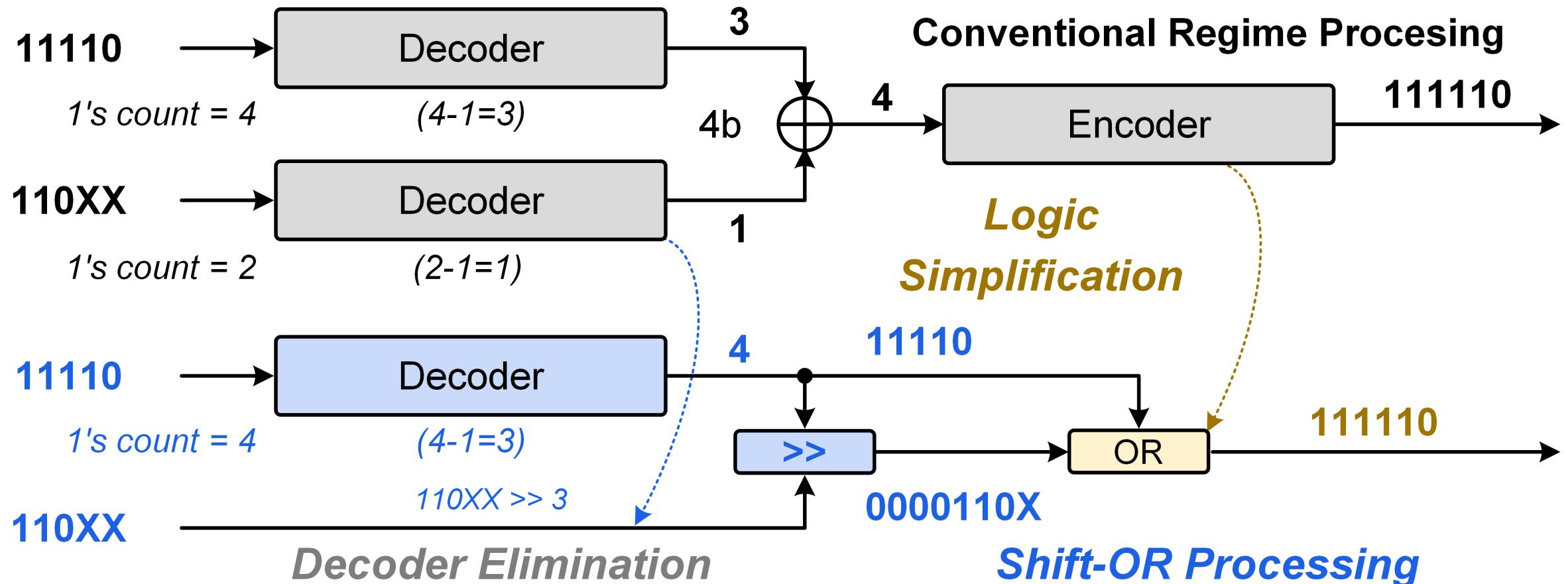
1's count sub 1 for pos. R (1110 for 2)

$$A \times B = (S_A \times S_B) \times (2^K)^{(R_A+R_B)} \times 2^{(E_A+E_B)} \times (1.f_A \times 1.f_B)$$



- Step1: Regime extracting with leading 1/0 detector.
- Step2: Regime processing with **codec and addition**.

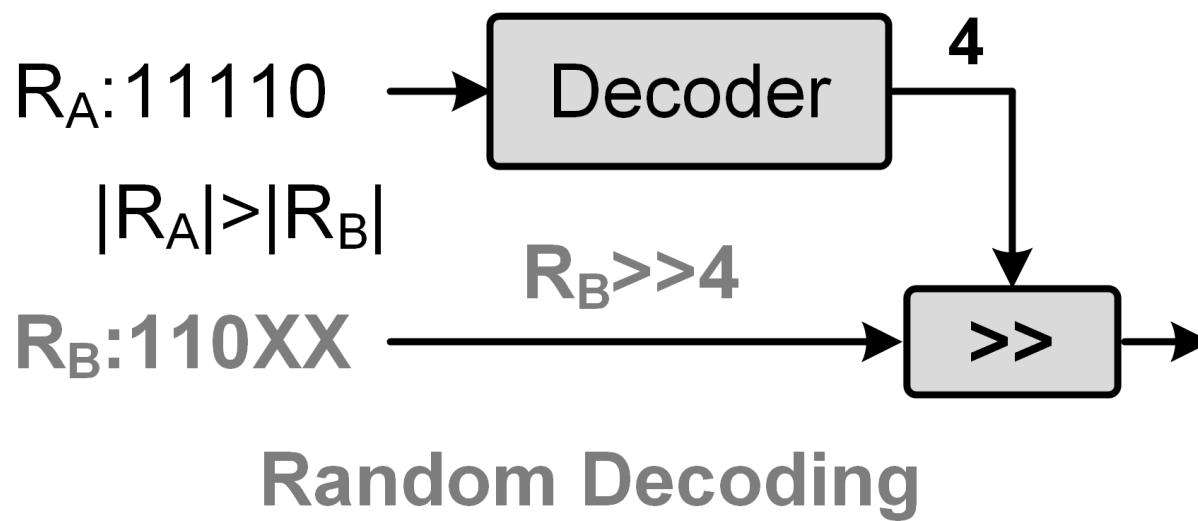
# Feature 1: Bi-directional Regime Processing



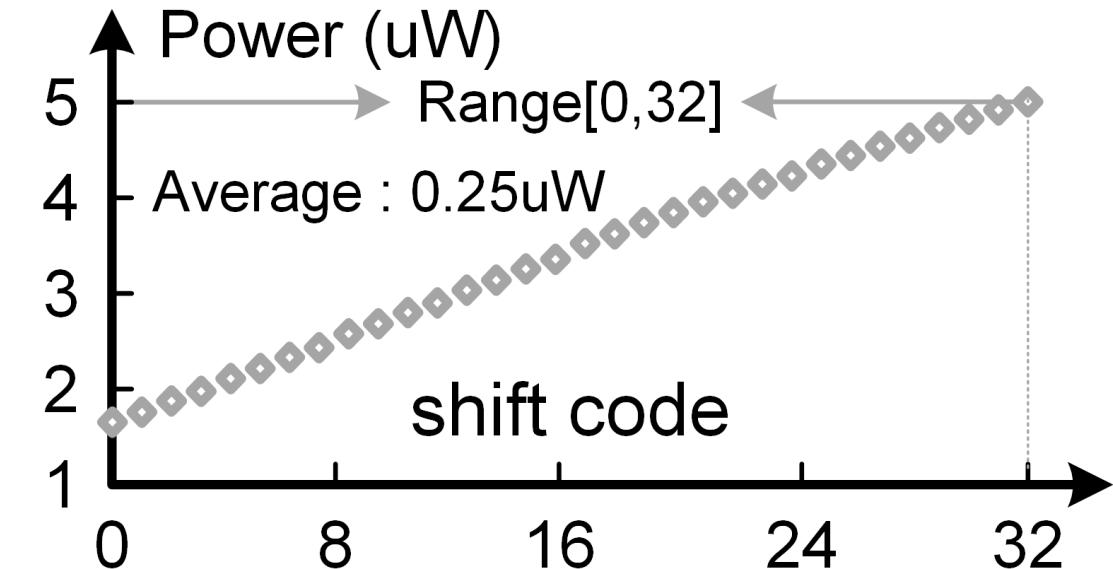
- BRPU replaces **codec-addition** with **shift-or** processing.

# Feature 1: Bi-directional Regime Processing

Same Sign  $R_A + R_B$

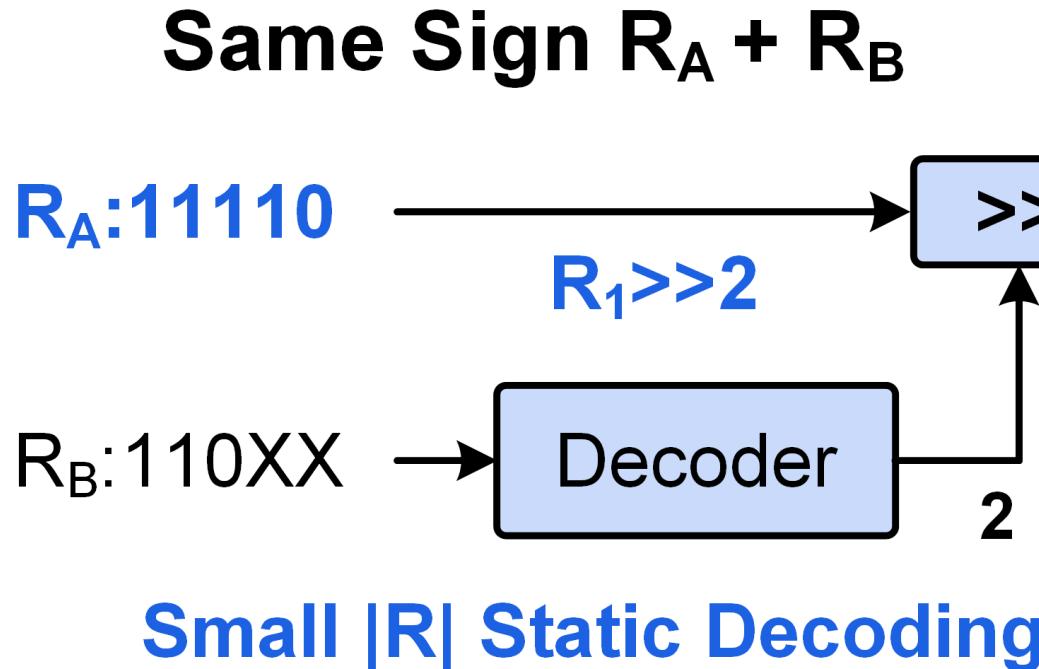


*Simulation with TSMC  
28nm Technology at 400MHz*

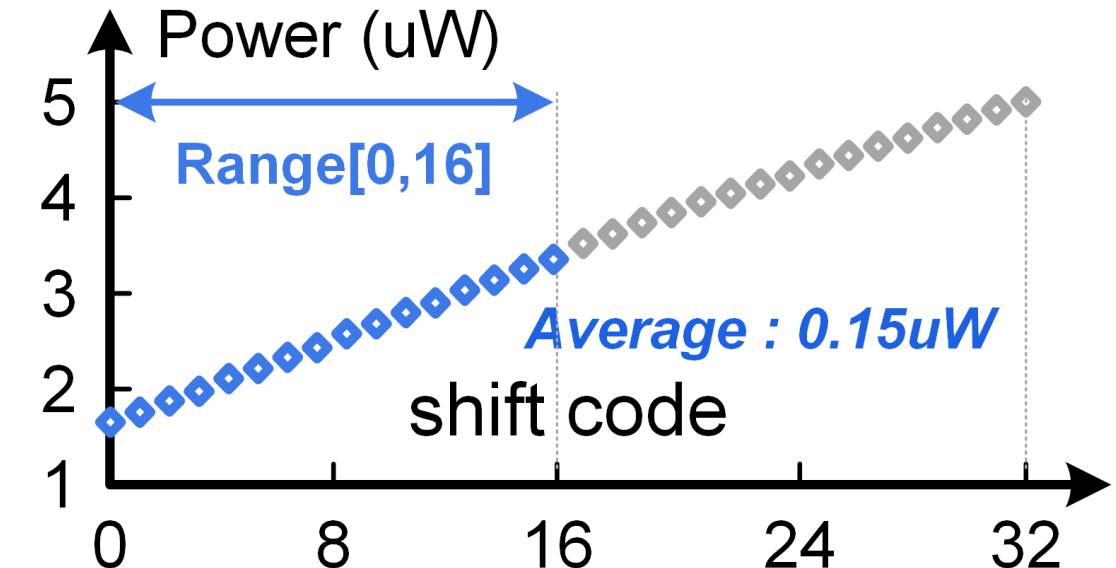


- Same sign addition: logic shift to **increases 1's/0's counts**.
- A **large shift code always denotes more shift power**.

# Feature 1: Bi-directional Regime Processing



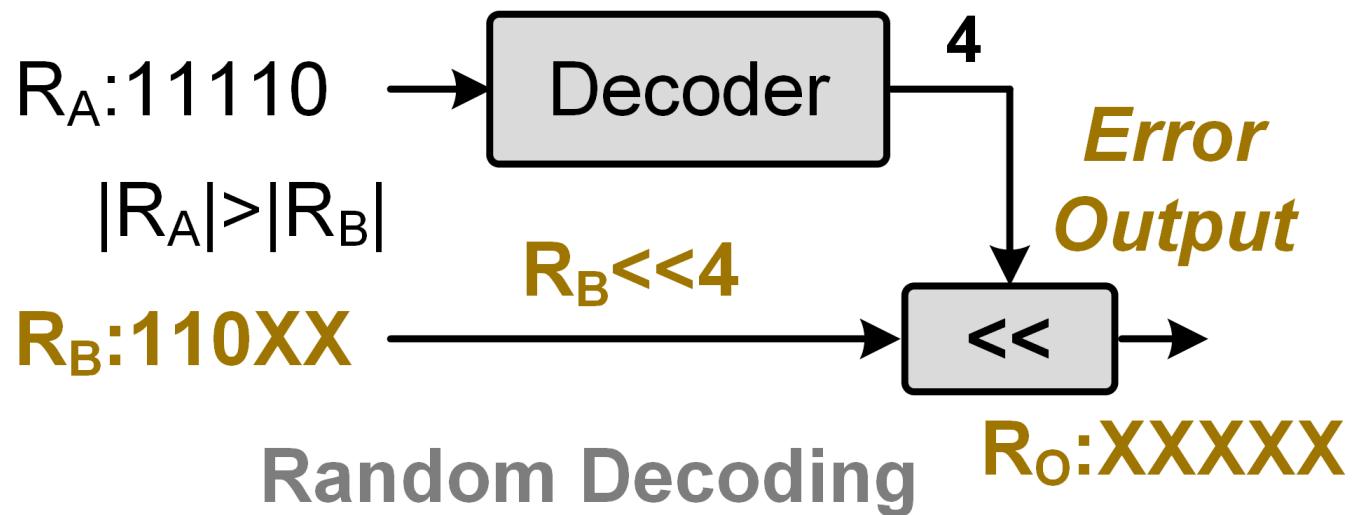
*Simulation with TSMC  
28nm Technology at 400MHz*



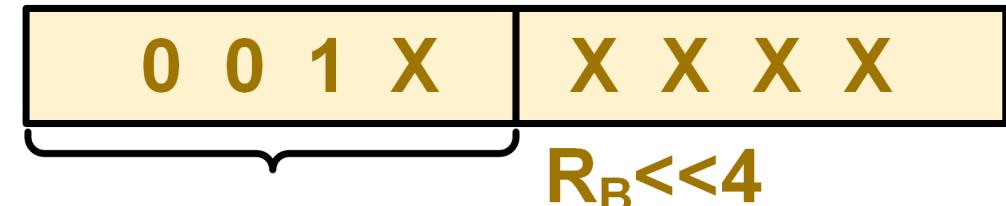
- BRPU dynamically decodes small  $|R_B|$  to shifts large  $|R_A|$ .
- BRPU minimizes shift code to saves 40% of shift energy.

# Feature 1: Bi-directional Regime Processing

Different Sign  $R_A + R_B$



Complexity:  
 $O[2N * \log(2N)]$



Extra Shift Bit  
for Overflow

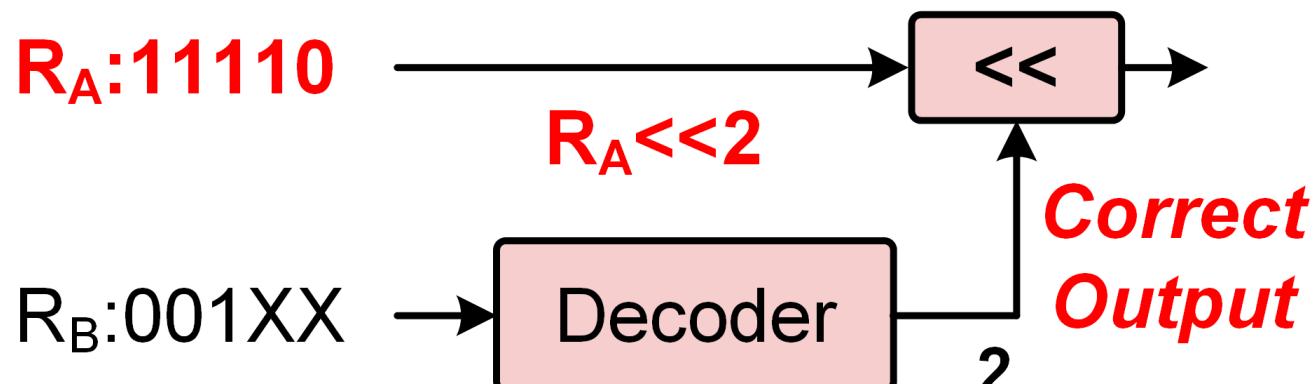
No Overflow

Complexity:  $O[N * \log(N)]$

- Different sign addition: logic shift to **decrease 1's/0's counts**.
- If shift code  $\geq R$ 's effective bit-width, it introduces shift error.

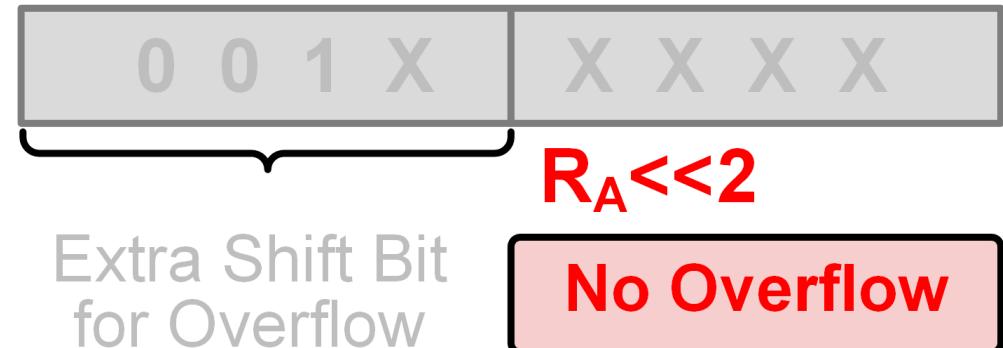
# Feature 1: Bi-directional Regime Processing

Different Sign  $R_A + R_B$   $R_O:110XX$



Small  $|R|$  Static Decoding

Complexity:  
 $O[2N * \log(2N)]$



Complexity:  $O[N * \log(N)]$

- BRPU dynamically decodes small  $|R_B|$  to shifts large  $|R_A|$ .
- BRPU avoids shift overflow to reduce 50% of shift logic.

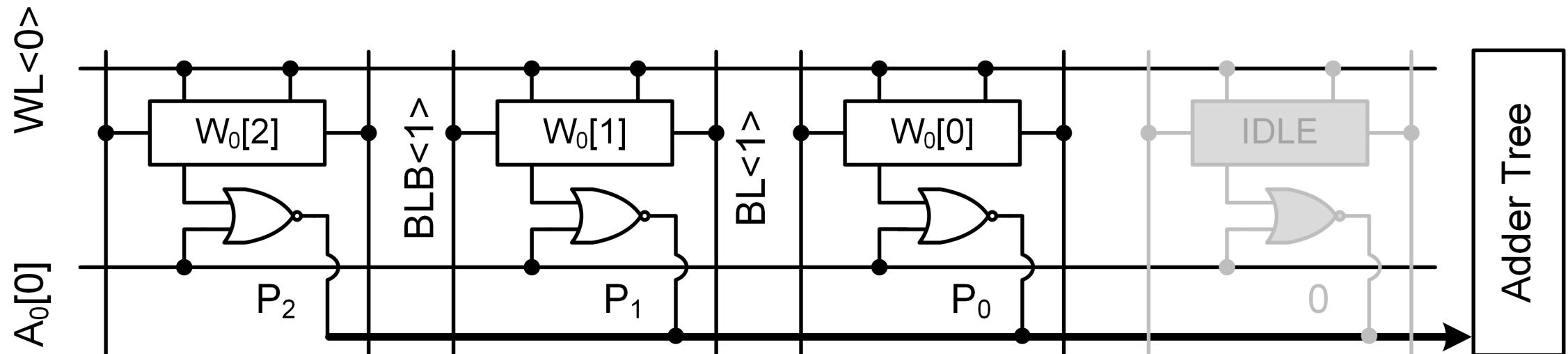
# Outline

- Background and Motivation
- Challenges of POSIT-Based CIM Macro
- Proposed POSIT@CIM Macro Features
  - Bi-directional Regime Processing Codec
  - **Critical-bit Pre-compute-and-store CIM Array**
  - Cyclically-alternating Scheduling Adder Tree
- Measurement and Comparison
- Conclusion

# Feature 2: Critical-bit Pre-compute-and-store

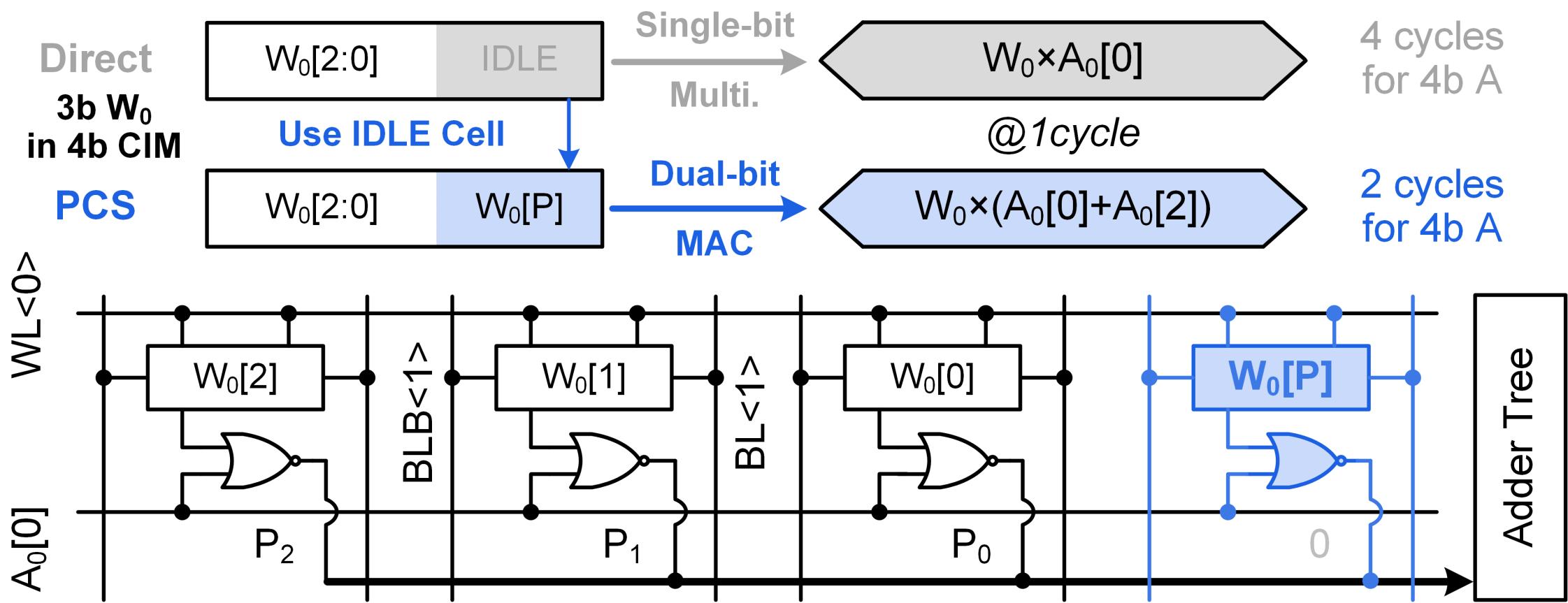
*Mantissa Distribution of Posit Format Weight for ResNet18 Training*

Posit(8,1)	2b(15.8%)	3b(46.1%)	4b(31.6%)	others
Posit(8,2)	2b(10.6%)	3b(51.6%)	4b(34.2%)	others



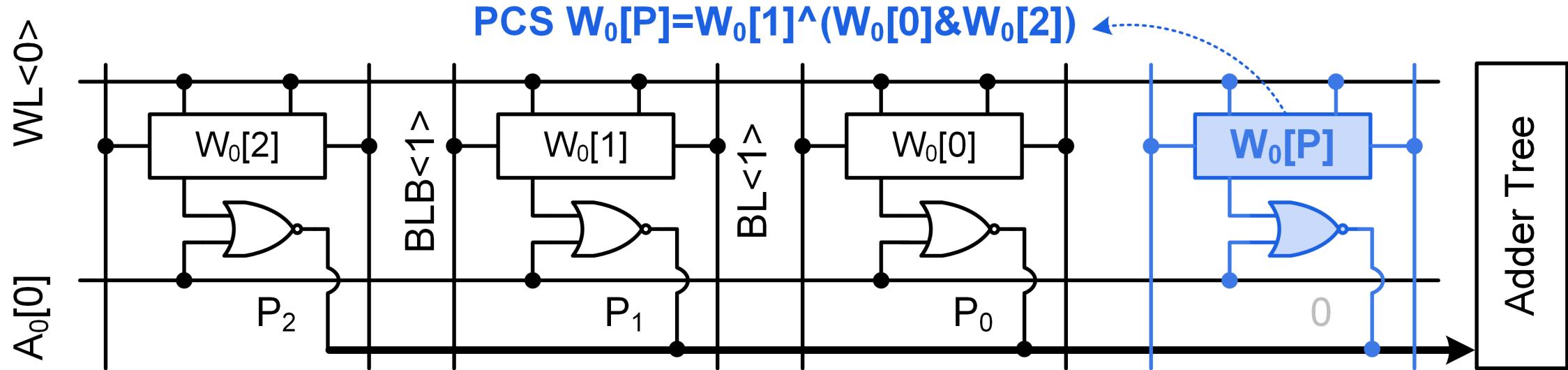
- Dynamic mantissa bit-width introduces **48.9% cell waste**.

# Feature 2: Critical-bit Pre-compute-and-store



- CPCS uses spare bits to achieve **dual-bit MAC** in each cycle.

# Feature 2: Critical-bit Pre-compute-and-store

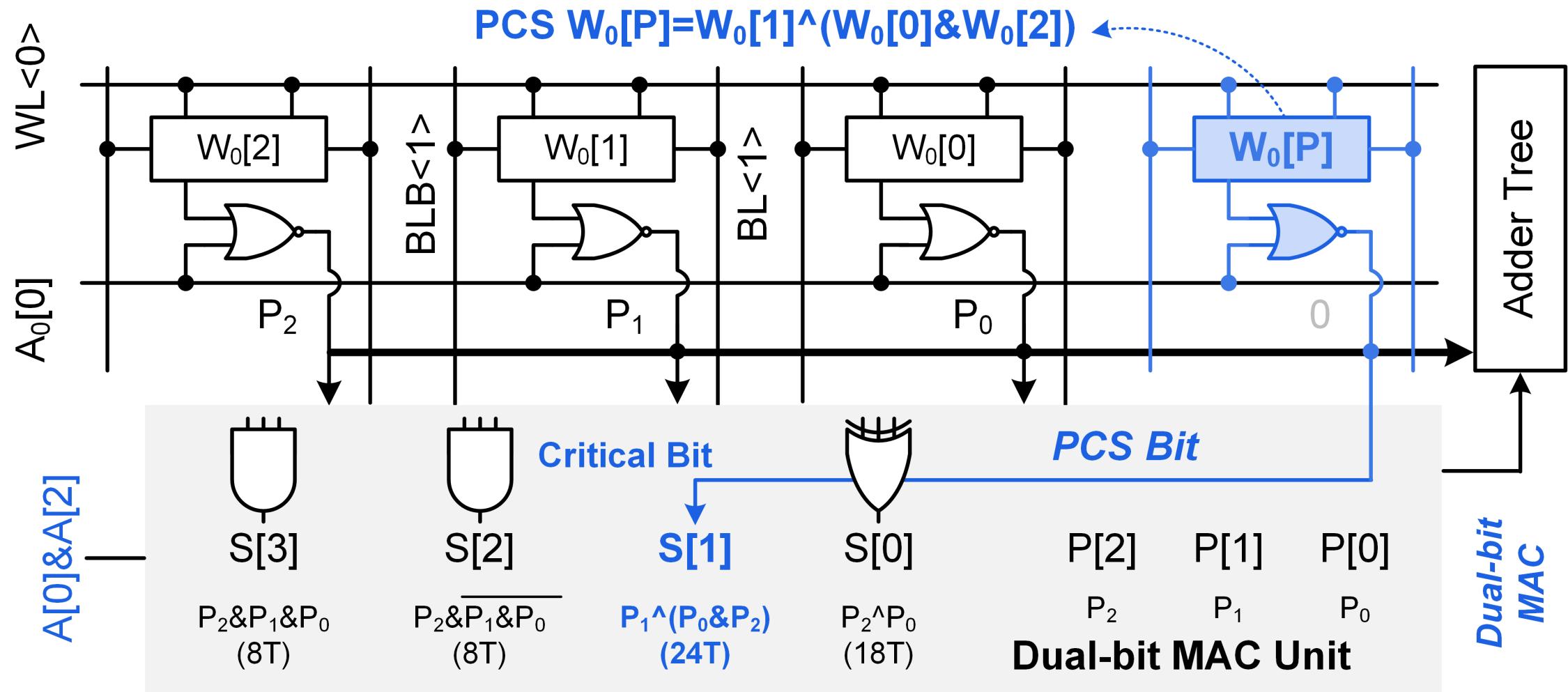


$$\text{Output} = W[2:0] \times A[0] + W[2:0] \times A[2]$$

$$\begin{array}{r} & P_2 & P_1 & P_0 \\ + & P_2 & P_1 & P_0 \\ \hline S_3 & S_2 & S_1 & S_0 & P_1 & P_0 \end{array}$$

$$O = \begin{cases} \{S[3:0], P[1:0]\}, & A[0], A[2] = 11 \\ P[2:0], & A[0], A[2] \neq 11 \end{cases}$$

# Feature 2: Critical-bit Pre-compute-and-store

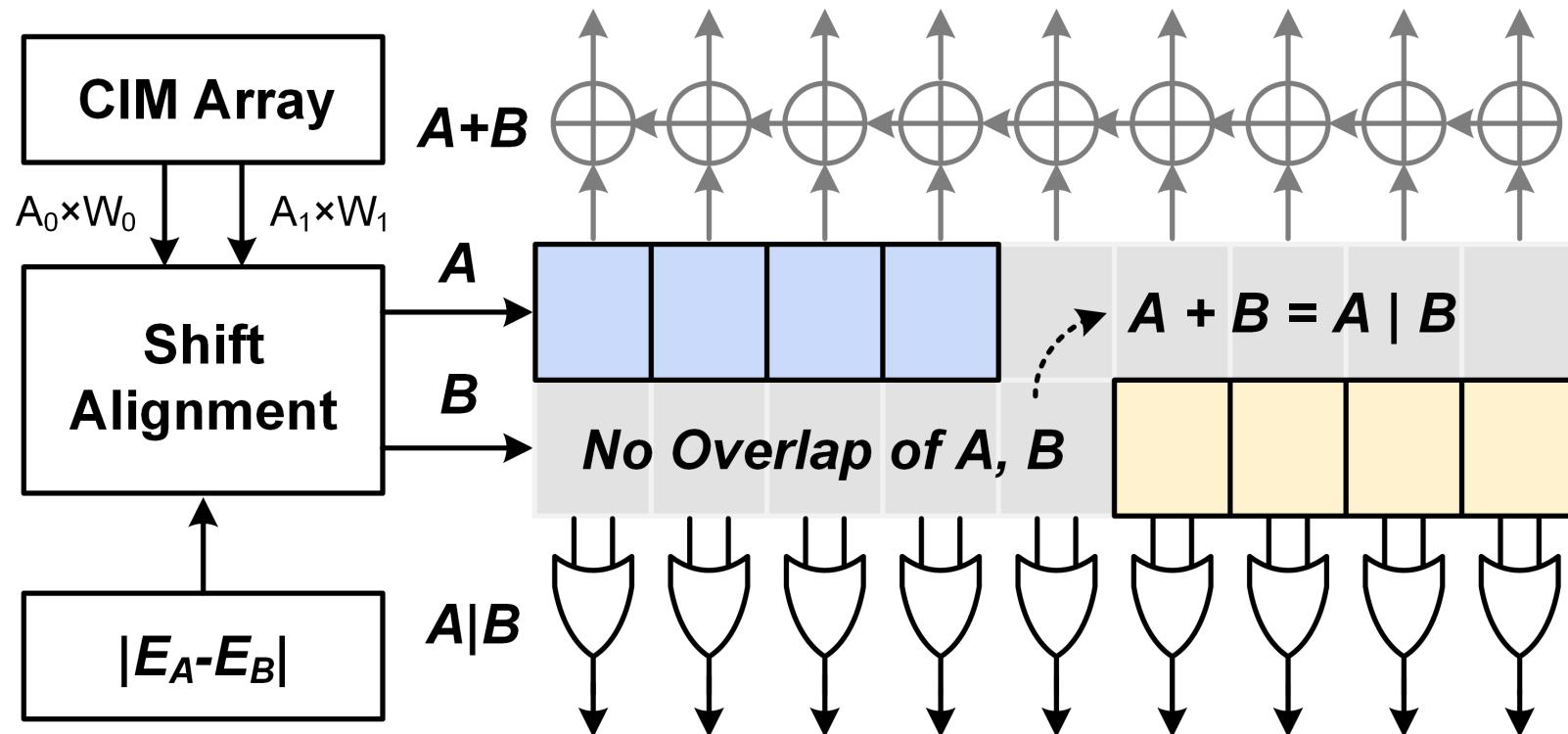


# Outline

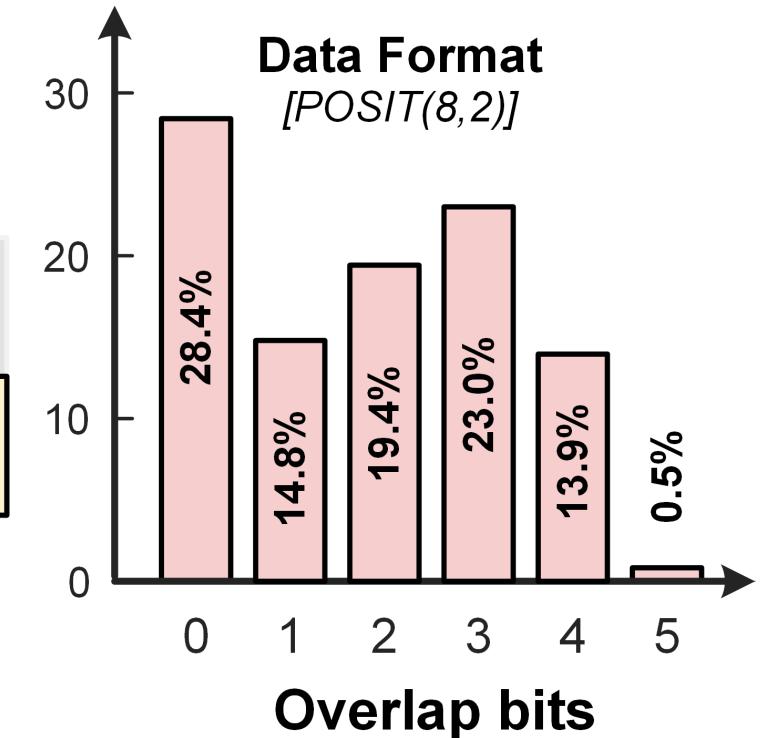
- Background and Motivation
- Challenges of POSIT-Based CIM Macro
- Proposed POSIT@CIM Macro Features
  - Bi-directional Regime Processing Codec
  - Critical-bit Pre-compute-and-store CIM Array
  - **Cyclically-alternating Scheduling Adder Tree**
- Measurement and Comparison
- Conclusion

# Feature 3: Cyclically-alternating Scheduling

## Bit-wise OR-based Accumulation

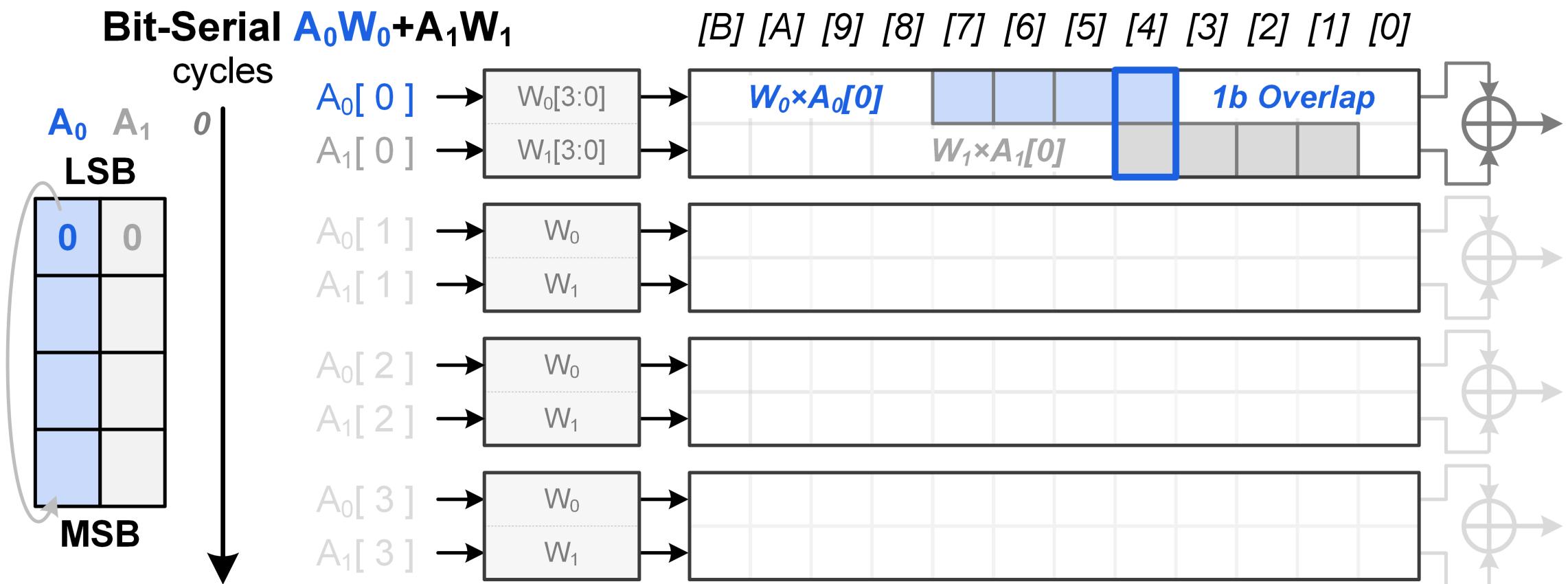


## Overlap Ratio



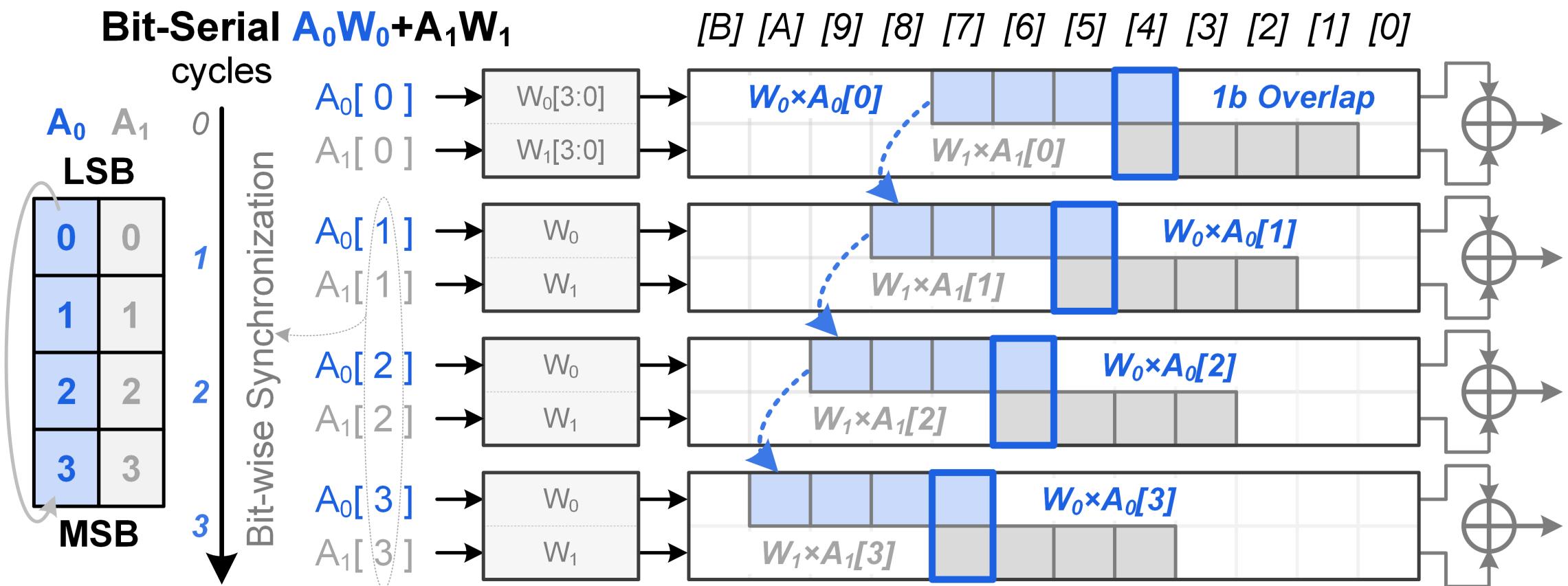
- If A and B have **no overlap bits**,  $A + B$  is equal to  $A | B$ .

# Feature 3: Cyclically-alternating Scheduling



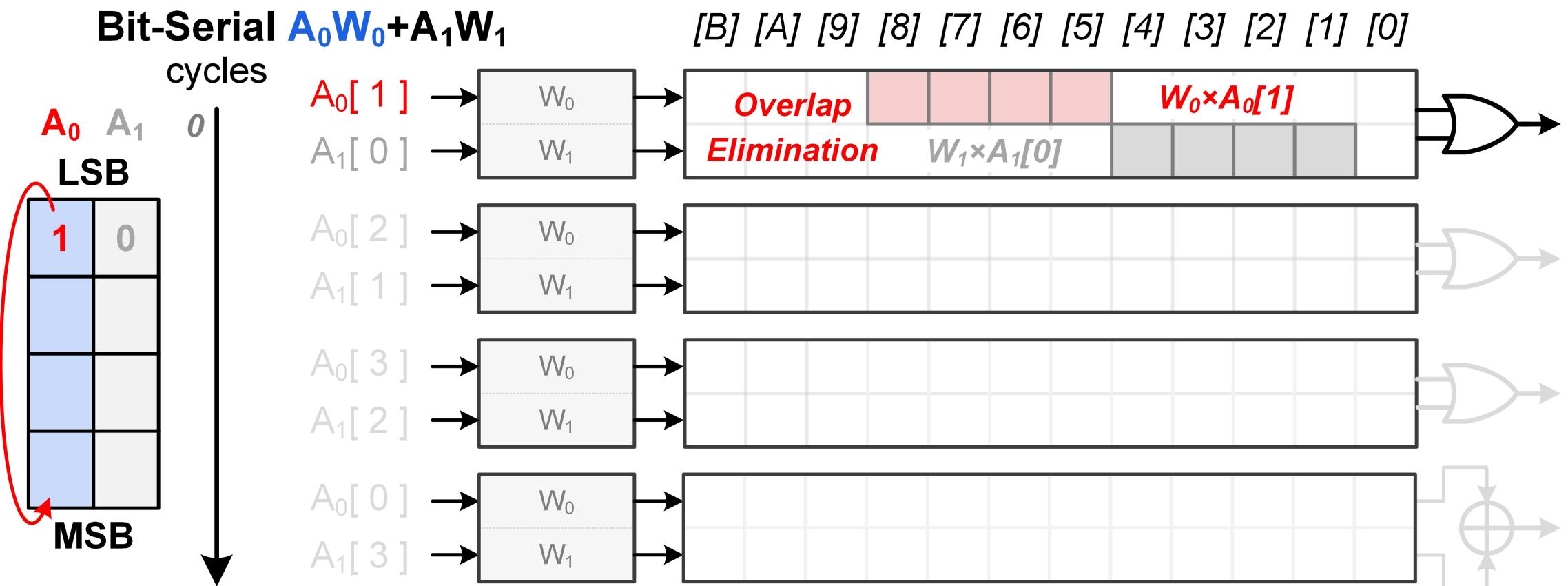
- Even  $A_0/A_1$  have 1 **overlap bit**,  $A_0W_0 + A_1W_1$  has to use adder.

# Feature 3: Cyclically-alternating Scheduling



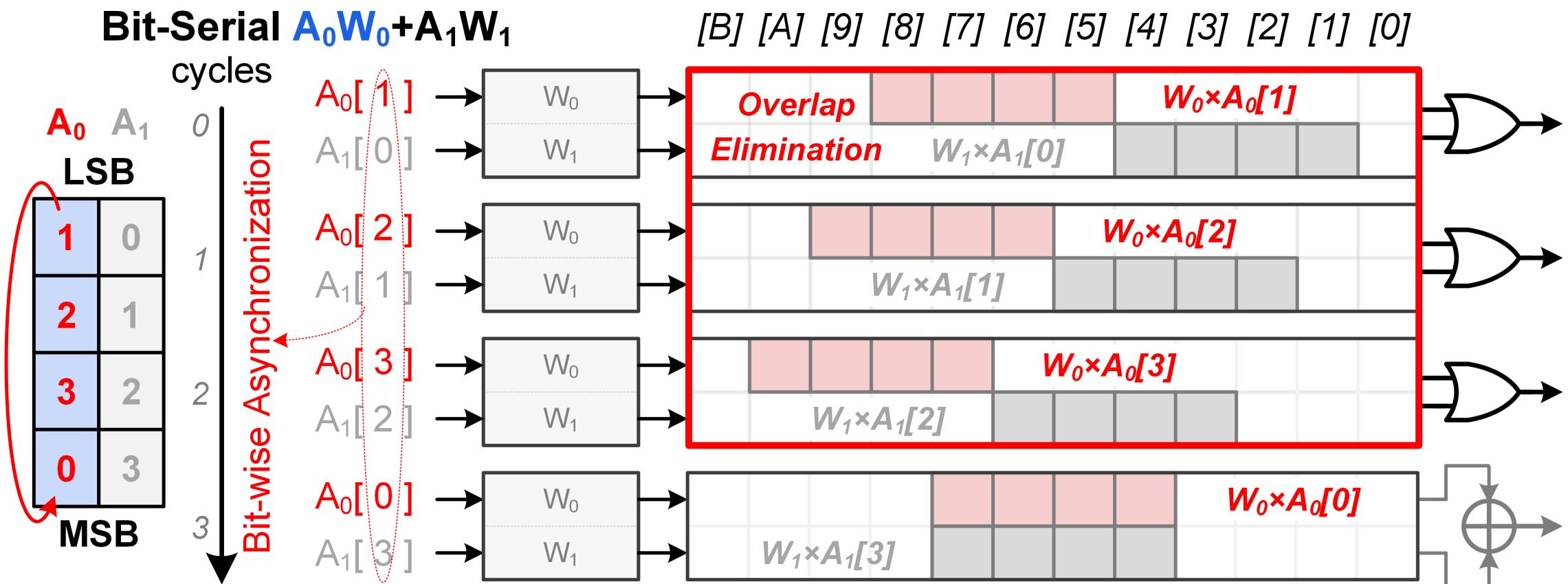
- All cycles need adders for **synchronous bit-serial computing**.

# Feature 3: Cyclically-alternating Scheduling



- CASU cyclically shifts  $A_0$  for asynchronous computing with  $A_1$ .

# Feature 3: Cyclically-alternating Scheduling

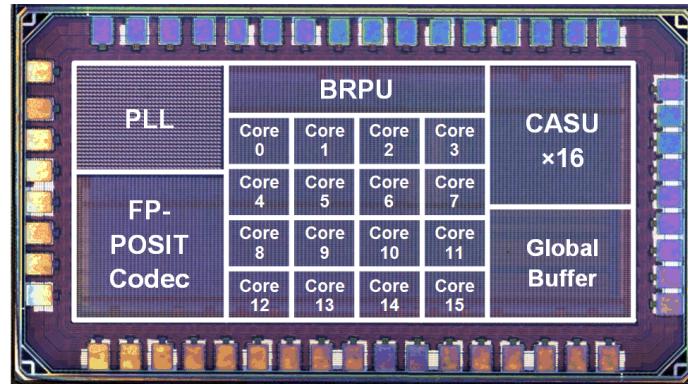
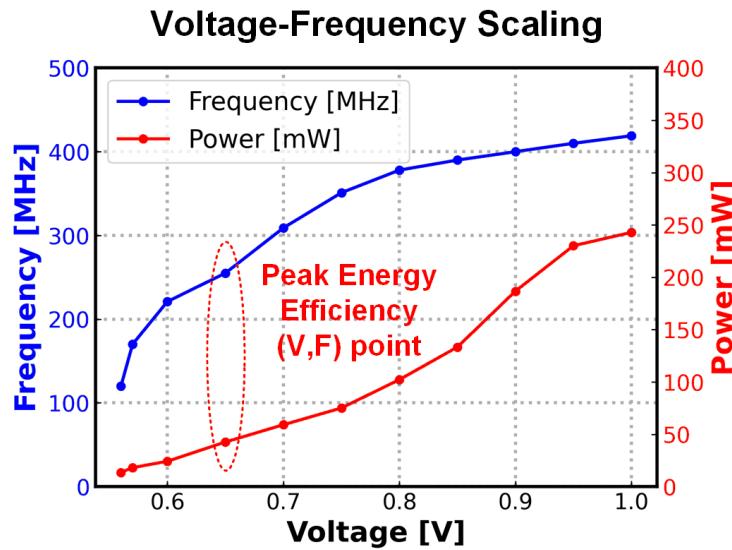


- CASU eliminates overlap bits in former cycles of  $A_0W_0 + A_1W_1$ .

# Outline

- **Background and Motivation**
- **Challenges of POSIT-Based CIM Macro**
- **Proposed POSIT@CIM Macro Features**
  - Bi-directional Regime Processing Codec
  - Critical-bit Pre-compute-and-store CIM Array
  - Cyclically-alternating Scheduling Adder Tree
- **Measurement and Comparison**
- **Conclusion**

# Chip Photograph and Summary



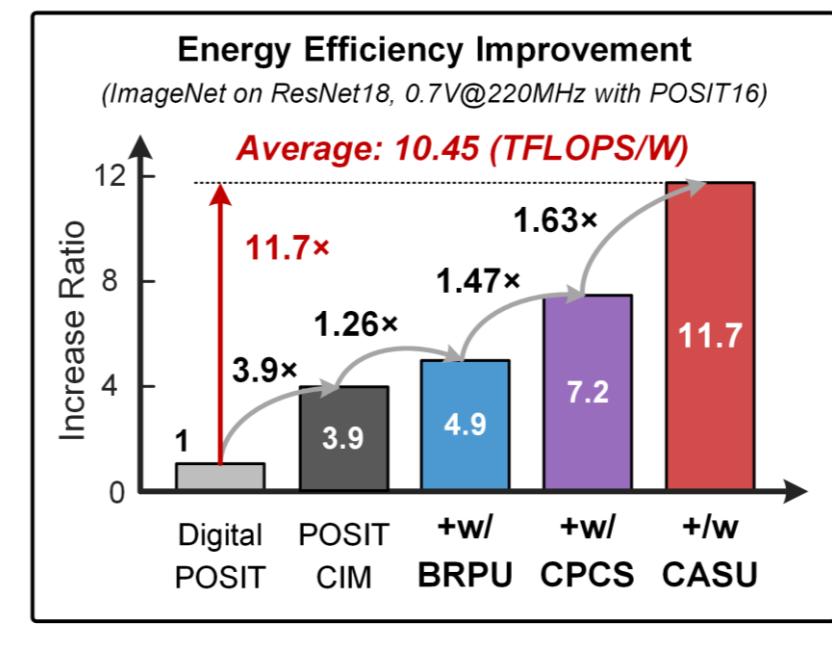
Specifications	
Technology	28nm CMOS
Die Area	1.41 mm <sup>2</sup>
CIM Size	12 KB
Buffer Size	28 KB
Voltage	0.55V-1.0V
Frequency	78MHz-419MHz
Data Precision	Posit(8,1)      Posit(16,2)
Peak Performance <sup>1)</sup>	3.86TOPS      1.91TOPS
Area Efficiency	2.74TOPS/mm <sup>2</sup> 1.35TOPS/mm <sup>2</sup>
Cim Micro Energy Efficiency <sup>2)</sup>	16.34-83.23 TOPS/W      7.47-38.37 TOPS/W
System Energy Efficiency <sup>2)</sup>	10.90-55.60 TOPS/W      5.35-27.61 TOPS/W
Different AI Models	ResNet18 <sup>3)</sup> @Imagenet1k      69.64% (Top1 Acc↑)      69.71% (Top1 Acc↑)
	GPT-2 <sup>4)</sup> @Wikitext-2      21.31 (Perplexity↓)      21.45 (Perplexity↓)
	VIT-B <sup>5)</sup> @Imagenet1k      80.17% (Top1 Acc↑)      80.05% (Top1 Acc↑)

- One operation (OP) represents one multiplication or addition.*
- 1) Highest efficiency point, 0.65V, 78MHz, 50% input sparsity**
  - 2) The baseline is 69.76%**
  - 3) The baseline is 21.30**
  - 4) The baseline is 80.31%**

# Training and Inference Performance

Evaluation on Different Models<sup>1)</sup>

Model	ResNet18	GPT-2	VIT-B
Dataset	Imagenet-1k	Wikitext-2	Imagenet-1k
Task	Training	Inference	Inference
Data Precision	Posit(16,2)	Posit(8,1)	Posit(8,1)
Accuracy	69.71% (Top1 Acc)	21.31 (Perplexity)	80.17% (Top1 Acc)
Accuracy Loss <sup>2)</sup>	-0.04%	-0.05%	-0.14%
Performance(TOPS) <sup>3)</sup>	1.73	2.95	2.91
Cim Micro Energy Efficiency (TOPS/W)	16.27	34.30	33.81
System Energy Efficiency (TOPS/W)	10.45	21.79	21.48
Energy Saving <sup>5)</sup>	8.81x	7.90x	7.12x



1) Measured at 1.0V, 419 MHz for high-performance evaluation. 2) Compared with models training in FP32.

3) Include all on-chip components,. Off-chip memory is not included.

- POSIT(16,2) only incurs 0.04% of training loss than FP32 .
- It achieves 10.45TFLOPS/W of average energy efficiency.

# Performance Comparison

## Comparison with SOTA FP CIM Macros

	VLSI'21[1]	ISSCC'22[2]	ISSCC'23[3]	ISSCC'23[4]	ISSCC'21[5]	This Work
<b>Dynamic Mantissa Length</b>	NO	NO	NO	NO	NO	YES
<b>Technique (nm)</b>	28	28	22	28	28	28
<b>Die Area (mm<sup>2</sup>)</b>	5.83	6.69	18	0.146	4.54	1.41
<b>Supply Voltage (V)</b>	0.76-1.1	0.6-1.0	0.6-0.8	0.6-0.9	0.397-0.90	0.55-1.0
<b>Frequency (MHz)</b>	250	50-220	NA	NA	10-400	78-419
<b>Precision</b>	BF16	FP32/BF16 INT16/INT8	BF16	BF16 INT8	FP16/BF16 INT8/4	<b>POSIT16</b> <b>POSIT8</b>
<b>Power (mW)</b>	1.2-156.1 <sup>1)</sup>	12.5-69.4	NA	NA	0.87-74.9	<b>5.5-237</b>
<b>Performance (TOPS or TFLOPS)</b>	0.12-0.66 <sup>1)</sup>	0.14@FP32 1.35@INT8	1.24-1.28	NA	1.64-9.63 <sup>3)</sup> @INT4	<b>1.91@POSIT16<sup>2)</sup></b> <b>3.86@POSIT8<sup>2)</sup></b>
<b>Energy Efficiency (TOPS/W or TFLOPS/W)</b>	1.43-13.7 <sup>1)</sup>	3.7@FP32 36.5@INT8	16.2-70.2 <sup>1)</sup>	14-31.6@BF16 <sup>2)</sup> 19.5-44@INT8 <sup>2)</sup>	3.2-16.9 <sup>3)</sup> @FP16 51-300 <sup>3)</sup> @INT4	<b>38.37@POSIT16<sup>2)</sup></b> <b>83.23@POSIT8<sup>2)</sup></b>
<b>Area Efficiency (TOPS/ mm<sup>2</sup>)</b>	0.021-1.1 <sup>1)</sup>	0.02@FP32 0.20@INT8	0.069-0.071	NA	0.36-2.12 <sup>3)</sup> @INT4	<b>1.35@ POSIT16<sup>2)</sup></b> <b>2.74@ POSIT8<sup>2)</sup></b>

1) Evaluated with 90% input sparsity.

2) Evaluated with 50% input sparsity.

3) From dense models to average of test sparse NN models.

# Outline

- **Background and Motivation**
- **Challenges of POSIT-Based CIM Macro**
- **Proposed POSIT@CIM Macro Features**
  - Bi-directional Regime Processing Codec
  - Critical-bit Pre-compute-and-store CIM Array
  - Cyclically-alternating Scheduling Adder Tree
- **Measurement and Comparison**
- **Conclusion**

# Conclusion

- An Energy Efficient POSIT-Based CIM Macro
  - Bi-directional Regime Processing Codec
    - ✓ Save Pre-processing Energy by Replacing Codec to Shift-OR
  - Critical-bit Pre-compute-and-store CIM Array
    - ✓ Improve CIM Utilization by Using Spare Bit for Dual-bit MAC
  - Cyclically-alternating Scheduling Adder Tree
    - ✓ Reduce Accumulation Power by Simplifying Addition to OR

A POSIT-Based CIM Macro with Bi-directional Regime Codec,  
Critical-bit Pre-computing-Storing and Cyclically-alternating  
Scheduling Achieving **83.23TFLOPS/W Energy Efficiency**



Please Scan to Rate  
This Paper



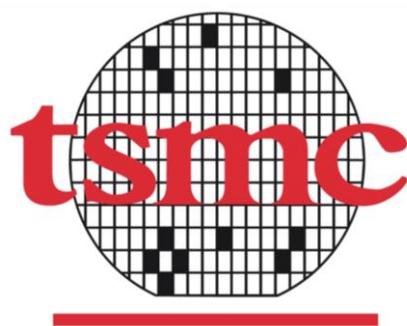
# A 16nm 96Kb Integer/Floating-Point Dual-Mode-Gain-Cell-Computing-in-Memory Macro Achieving 73.3-163.3TOPS/W and 33.2-91.2TFLOPS/W for AI-Edge Devices

**Win-San Vince Khwa<sup>\*1</sup>**, Ping-Chun Wu<sup>\*2</sup>, Jui-Jen Wu<sup>1</sup>, Jian-Wei Su<sup>2,3</sup>, Ho-Yu Chen<sup>2</sup>, Zhao-En Ke<sup>2</sup>, Ting-Chien Chiu<sup>2</sup>, Jun-Ming Hsu<sup>2</sup>, Chiao-Yen Cheng<sup>2</sup>, Yu-Chen Chen<sup>2</sup>, Chung-Chuan Lo<sup>2</sup>, Ren-Shuo Liu<sup>2</sup>, Chih-Cheng Hsieh<sup>2</sup>, Kea-Tiong Tang<sup>2</sup>, Meng-Fan Chang<sup>1,2</sup>

<sup>1</sup>TSMC Corporate Research, Hsinchu, Taiwan

<sup>2</sup>National Tsing Hua University, Hsinchu, Taiwan,

<sup>3</sup>Industrial Technology Research Institute, Hsinchu, Taiwan,



國立清華大學  
NATIONAL TSING HUA UNIVERSITY



工業技術研究院  
Industrial Technology  
Research Institute

# Outline

- **Challenges of Dual-mode Computing-In-Memory**
- **Proposed Computation Gain-cell CIM macro**
  - Overview of Integer / Floating-Point Dual-mode CIM Macro
  - Dual-mode Local-computing-cell (DM-LCC)
  - Dual-mode Zone-based Input Processing Unit Scheme (ZB-IPS)
  - Stationary-based Two-port Gain-cell Array Scheme (SB-TP-GCA)
- **Performance and Measurement Results**
- **Conclusion**

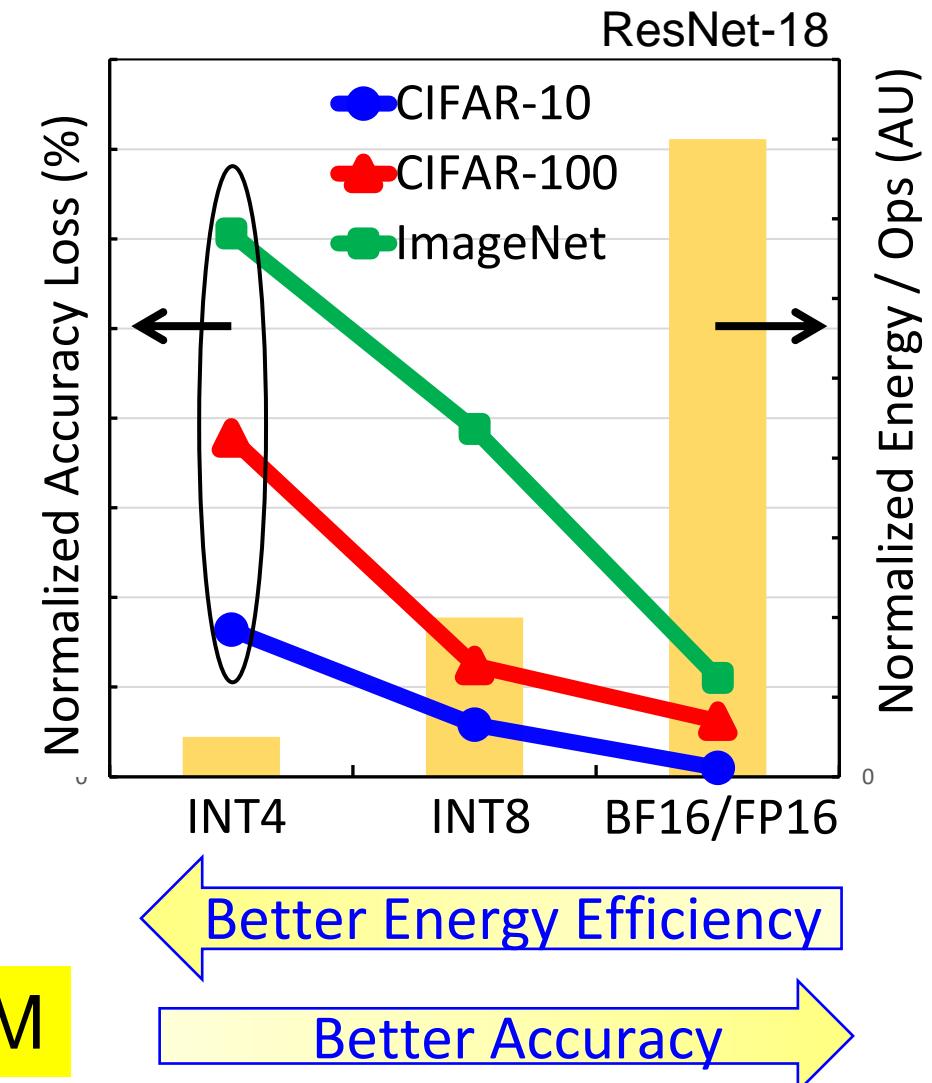
# Outline

- **Challenges of Dual-mode Computing-In-Memory**
- **Proposed Computation Gain-cell CIM macro**
  - Overview of Integer / Floating-Point Dual-mode CIM Macro
  - Dual-mode Local-computing-cell (DM-LCC)
  - Dual-mode Zone-based Input Processing Unit Scheme (ZB-IPS)
  - Stationary-based Two-port Gain-cell Array Scheme (SB-TP-GCA)
- **Performance and Measurement Results**
- **Conclusion**

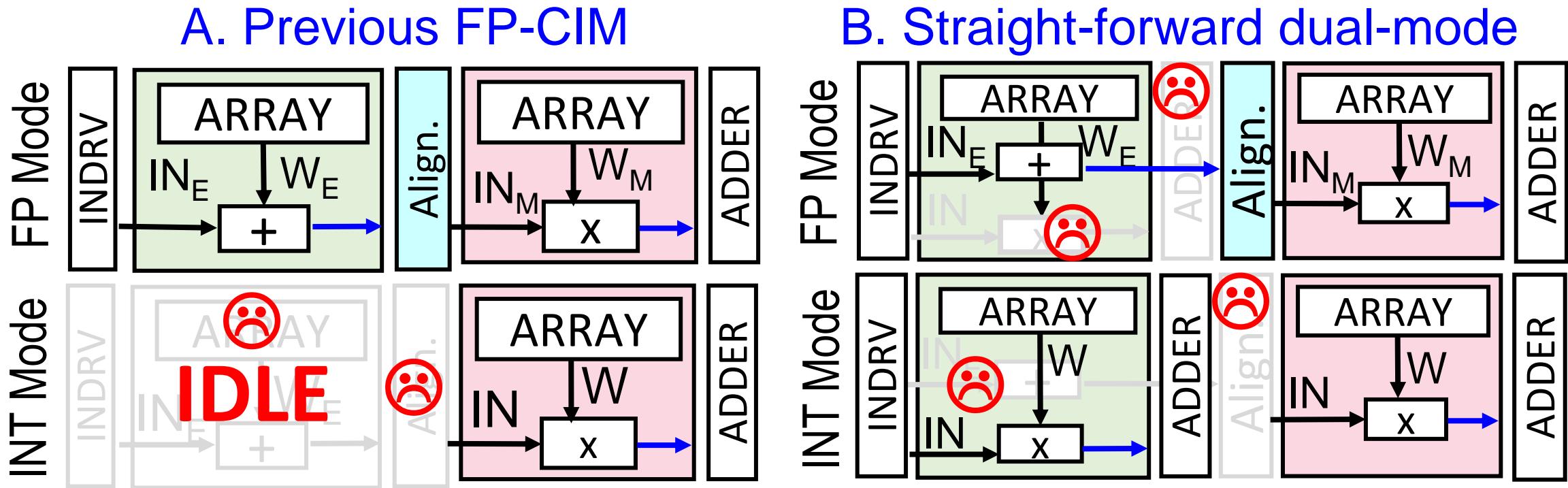
# Motivations of INT/FP Dual-mode CIM

- **Advanced AI-edge chip desirables:**
  - General purpose and computational flexibility
  - High energy efficiency and high area efficiency
  - Sufficient inference accuracy
- **Tradeoff between number formats with energy efficiency and inference accuracy:**
  - Floating-point (FP) mode
    - Pros: high inference accuracy
    - Cons: higher energy and more parameters
  - Integer (INT) mode
    - Pros: higher energy efficiency and less parameters
    - Cons: lower inference accuracy

Desirable to design an INT/FP dual-mode CIM



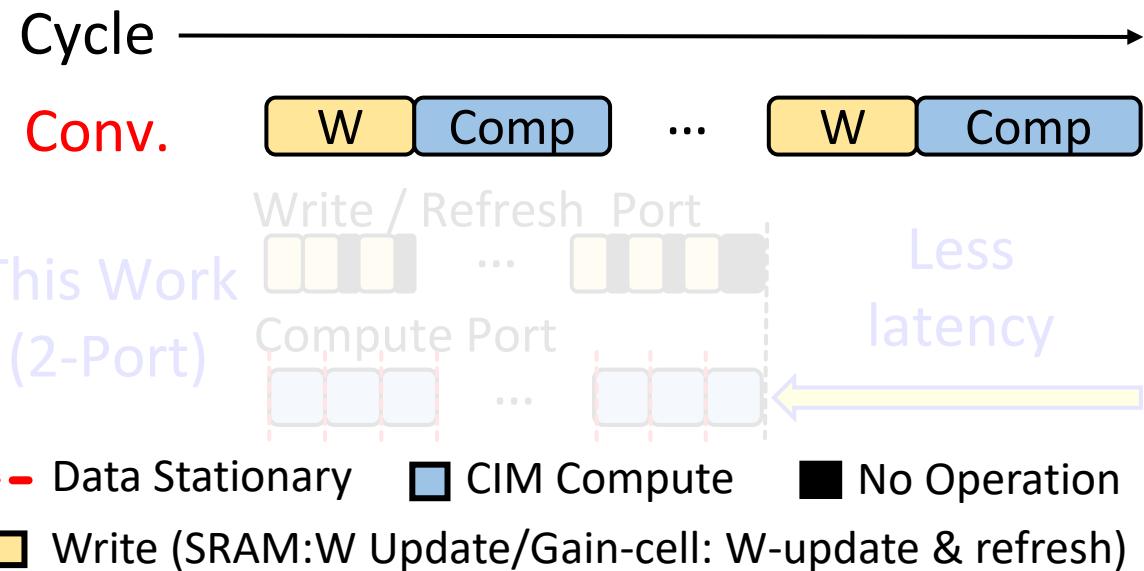
# Challenges of INT/FP Dual-Mode CIM (1/2)



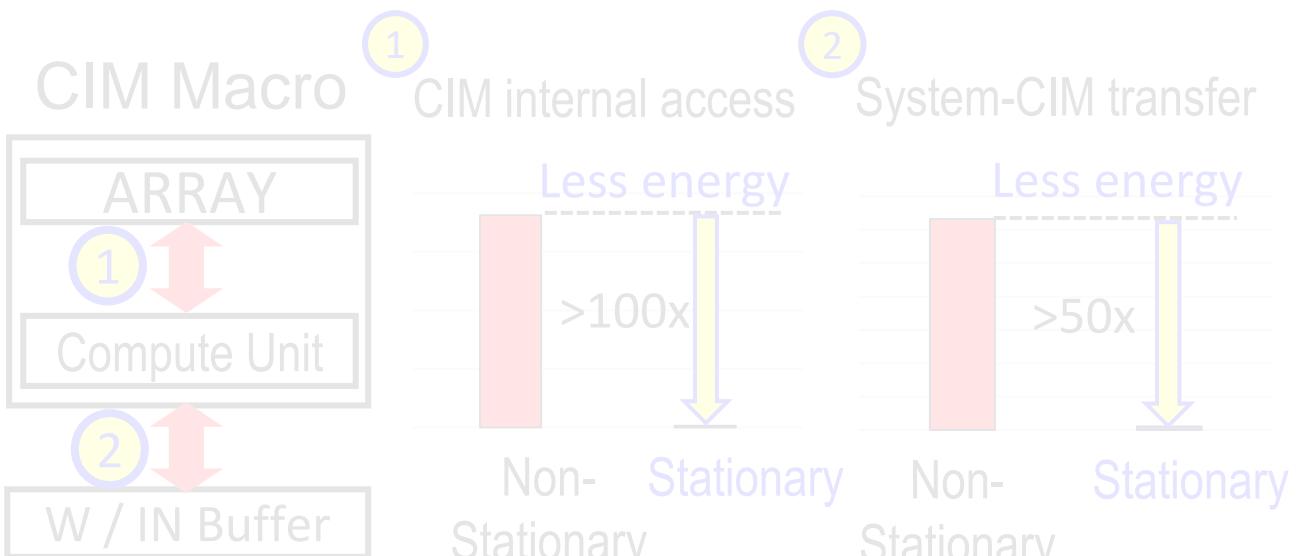
- **Low area utilization in integer-mode multiply-and-accumulation (INT-MAC)**
  - Floating-point multiply-and-accumulation functions becoming idle in INT-MAC
    - e.g. exponent computation, mantissa alignment circuit

# Challenges of INT/FP Dual-Mode CIM (2/2)

Long system latency due to data update



High energy due to frequently data transfer & data access



## ■ Long system-level latency

- NN data update interruptions without supporting concurrent write-and-compute

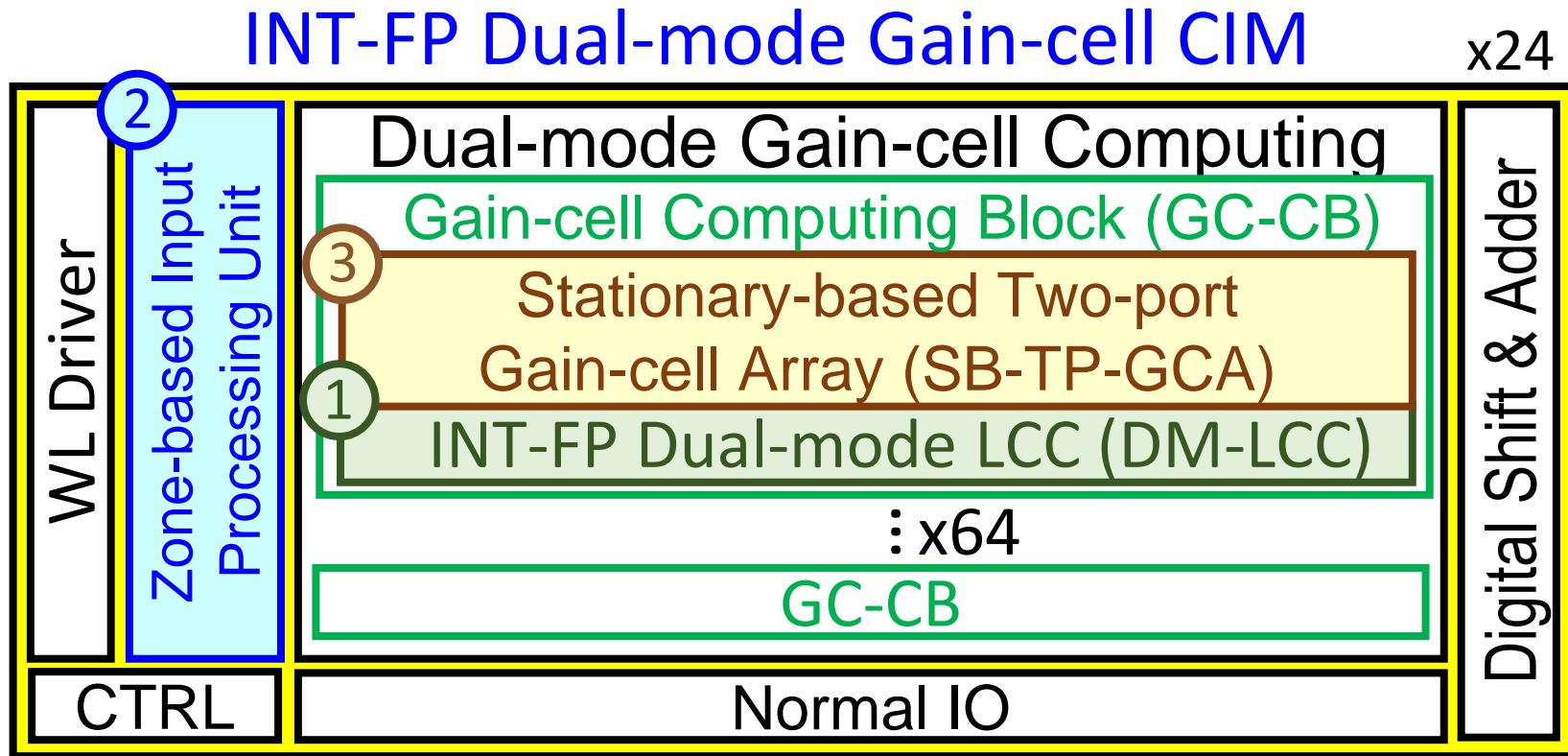
## ■ High energy consumption

- Redundant system-to-CIM data transfer during computation

# Outline

- Challenges of Dual-mode Computing-In-Memory
- **Proposed Computation Gain-cell CIM macro**
  - Overview of Integer / Floating-Point Dual-mode CIM Macro
  - Dual-mode Local-computing-cell (DM-LCC)
  - Dual-mode Zone-based Input Processing Unit Scheme (ZB-IPS)
  - Stationary-based Two-port Gain-cell Array Scheme (SB-TP-GCA)
- Performance and Measurement Results
- Conclusion

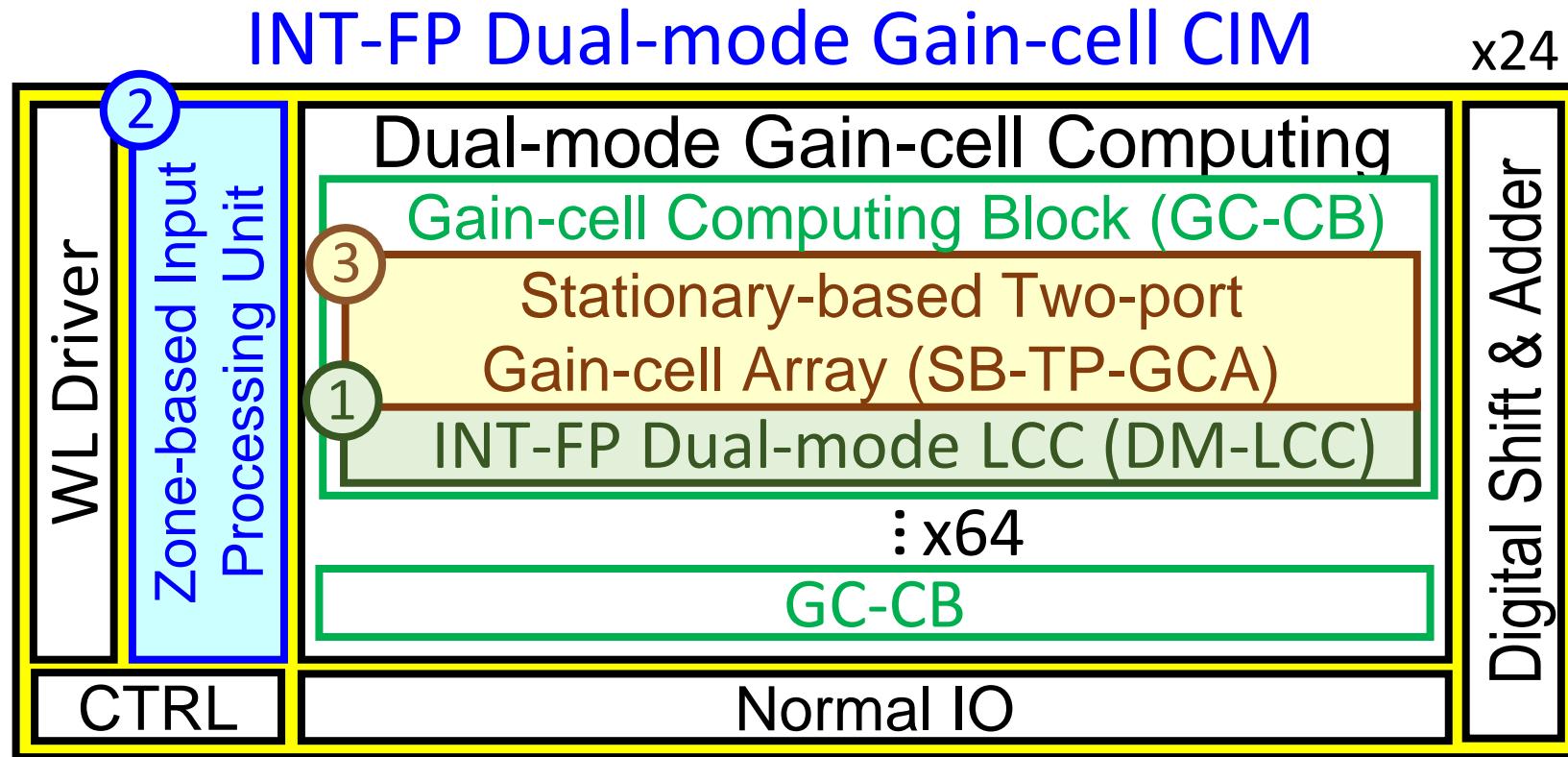
# Overview of Dual-mode Gain-cell-CIM Macro



## ① Dual-mode local-computing-cell (DM-LCC)

- Reuses the exponent addition as an adder tree stage for INT-MAC
- Improves area efficiency in INT mode

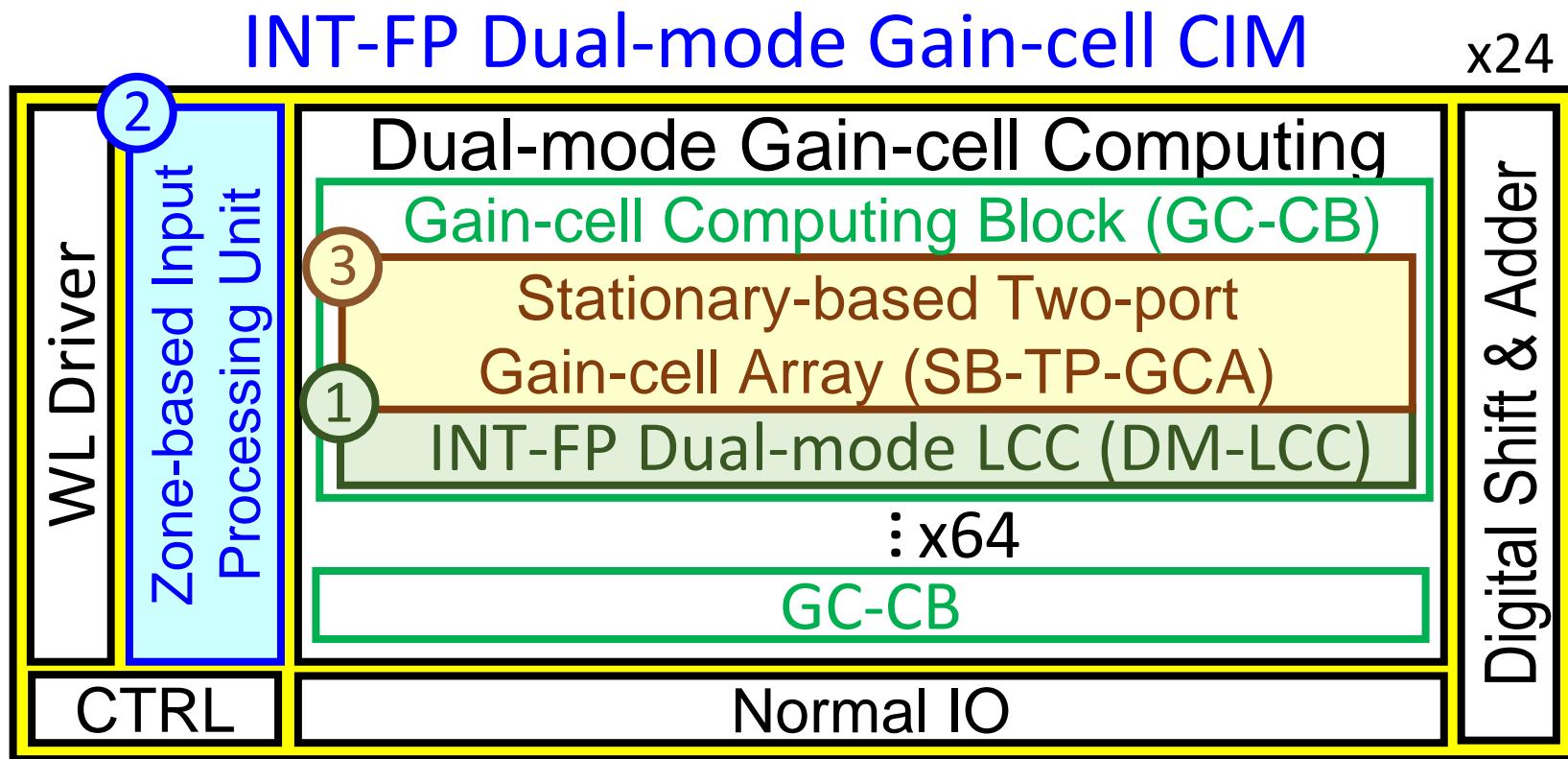
# Overview of Dual-mode Gain-cell-CIM Macro



## ② Dual-mode zone-based input processing scheme (ZB-IPS)

- Simplifies subtraction in exponent computation
- Reuses the alignment circuit in FP-mode as sparsity detection in INT-mode
- Improves energy efficiency and area efficiency

# Overview of Dual-mode Gain-cell-CIM Macro



## ③ Stationary-based Two-port Gain-cell Array (SB-TP-GCA)

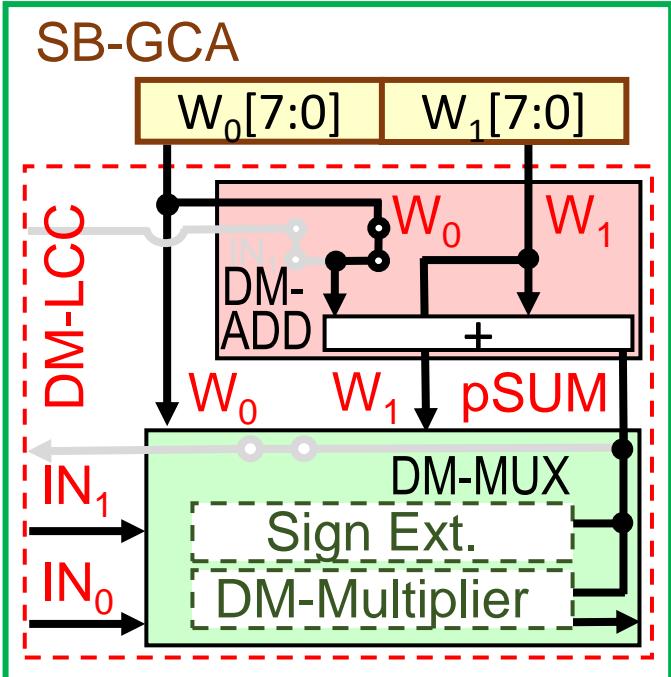
- Supports concurrent data updating and computation
- Reduces system-to-CIM and CIM internal data accesses
- Improves system energy efficiency and latency

# Outline

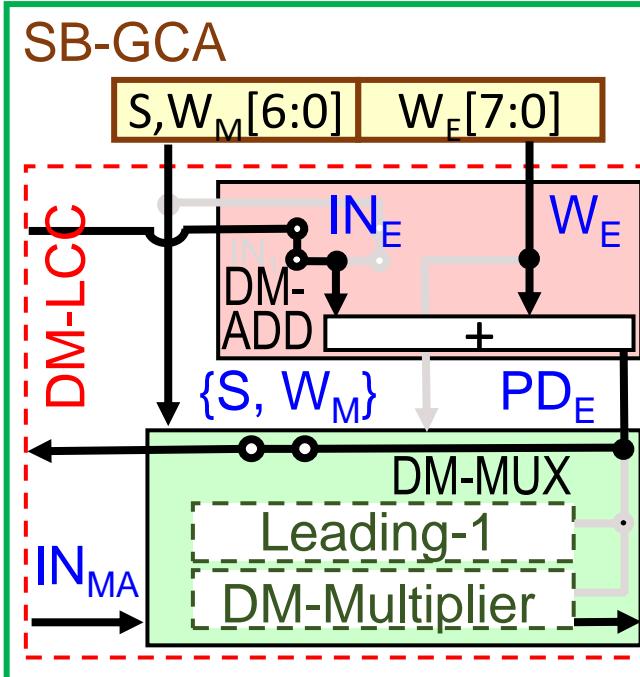
- Challenges of Dual-mode Computing-In-Memory
- **Proposed Computation Gain-cell CIM macro**
  - Overview of Integer / Floating-Point Dual-mode CIM Macro
  - Dual-mode Local-computing-cell (DM-LCC)
  - Dual-mode Zone-based Input Processing Unit Scheme (ZB-IPS)
  - Stationary-based Two-port Gain-cell Array Scheme (SB-TP-GCA)
- Performance and Measurement Results
- Conclusion

# Dual-mode Local-computing-cell (DM-LCC)

Integer 8bit Mode



BF16 Mode



## SB-GCA:

- INT8  $\rightarrow \{W_0[7:0], W_1[7:0]\}$
- BF16  $\rightarrow \{S, W_M[6:0], W_E[7:0]\}$

## DM-ADD:

- INT8  $\rightarrow (8bW_0 + 8bW_1)$
- BF16  $\rightarrow (8bIN_E + 8bW_E)$

## DM-MUX:

- INT8  $\rightarrow (8bIN_0 \times 8bW_0) + (8bIN_1 \times 8bW_1)$
- BF16  $\rightarrow (8bIN_M \times 8bW_M)$

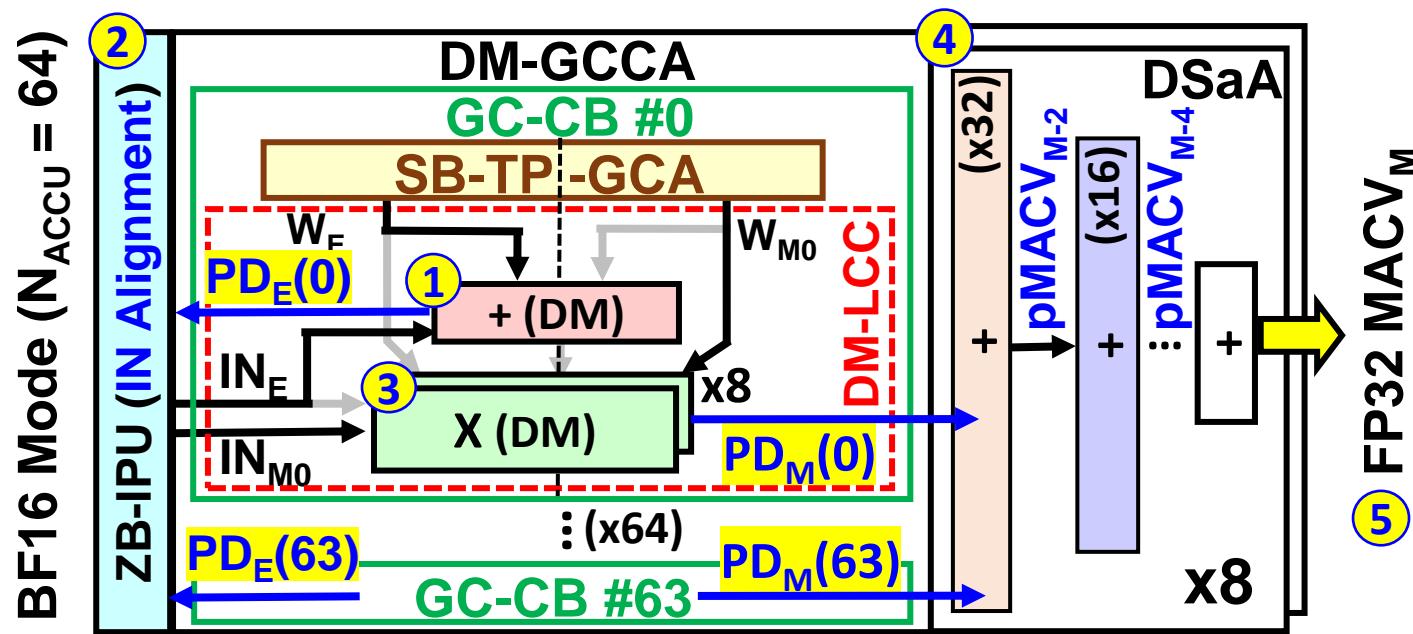
## ■ Gain-cell computing block (GC-CB)

- Stationary-based gain-cell sub-array (SB-GCA)
- Dual-mode adder (DM-ADD)
- Dual-mode multiplexer (DM-MUX)

# Computation Flow (Floating-Point Mode)

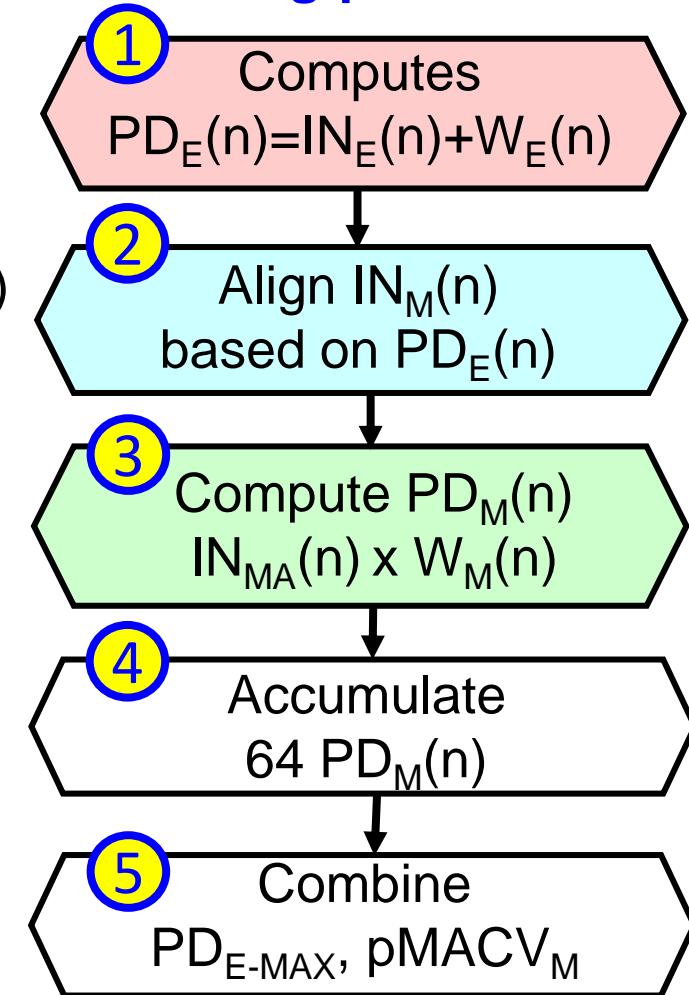
## Floating-point-MAC operation (Five compute phases)

- Phase 1: Computes  $PD_E(n) = (IN_E(n) + W_E(n))$  [n=0~63]
- Phase 2: Aligns  $IN_M(n)$  based on  $PD_E(n)$
- Phase 3: Computes  $PD_M(n) = IN_{MA}(n) \times W_M(n)$  [n=0~63]
- Phase 4: Accumulates 64  $PD_M(n)$ :  $pMACV_M = \sum_{n=0}^{63} (IN_{MA}(n) \times W_M(n))$
- Phase 5: Combines  $PD_{E-MAX}$ ,  $pMACV_M$  & outputs FP32 MACV



## Computation Flow Chart

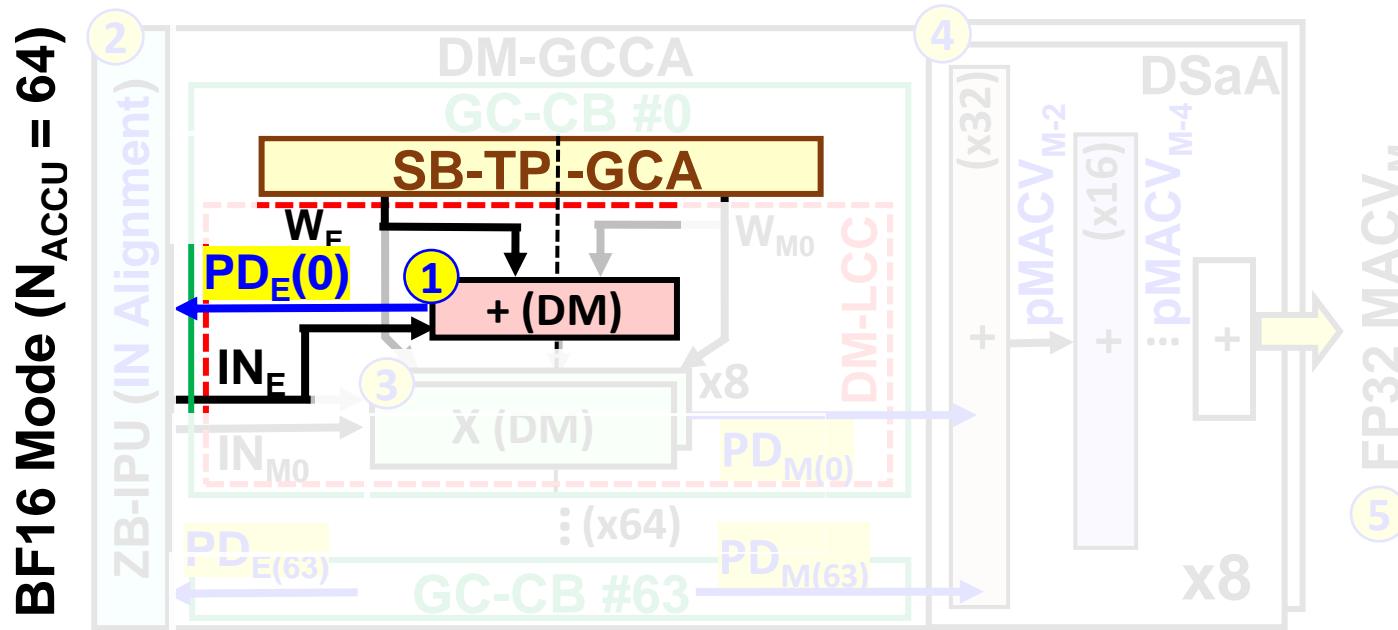
### Floating-point-MAC



# Computation Flow (Floating-Point Mode)

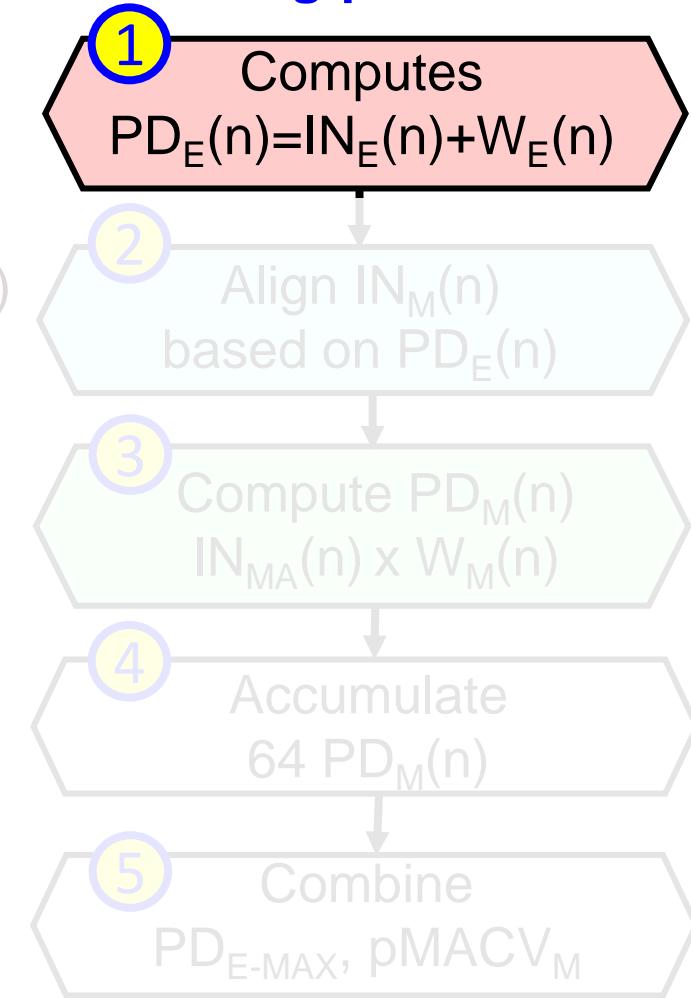
## ■ Floating-point-MAC operation (Five compute phases)

- Phase 1: Computes  $PD_E(n) = (IN_E(n) + W_E(n))$  [n=0~63]
- Phase 2: Aligns  $IN_M(n)$  based on  $PD_E(n)$
- Phase 3: Computes  $PD_M(n) = IN_{MA}(n) \times W_M(n)$  [n=0~63]
- Phase 4: Accumulates 64  $PD_M(n)$ :  $pMACV_M = \sum_{n=0}^{63} (IN_{MA}(n) \times W_M(n))$
- Phase 5: Combines  $PD_{E-MAX}$ ,  $pMACV_M$  & outputs FP32 MACV



## Computation Flow Chart

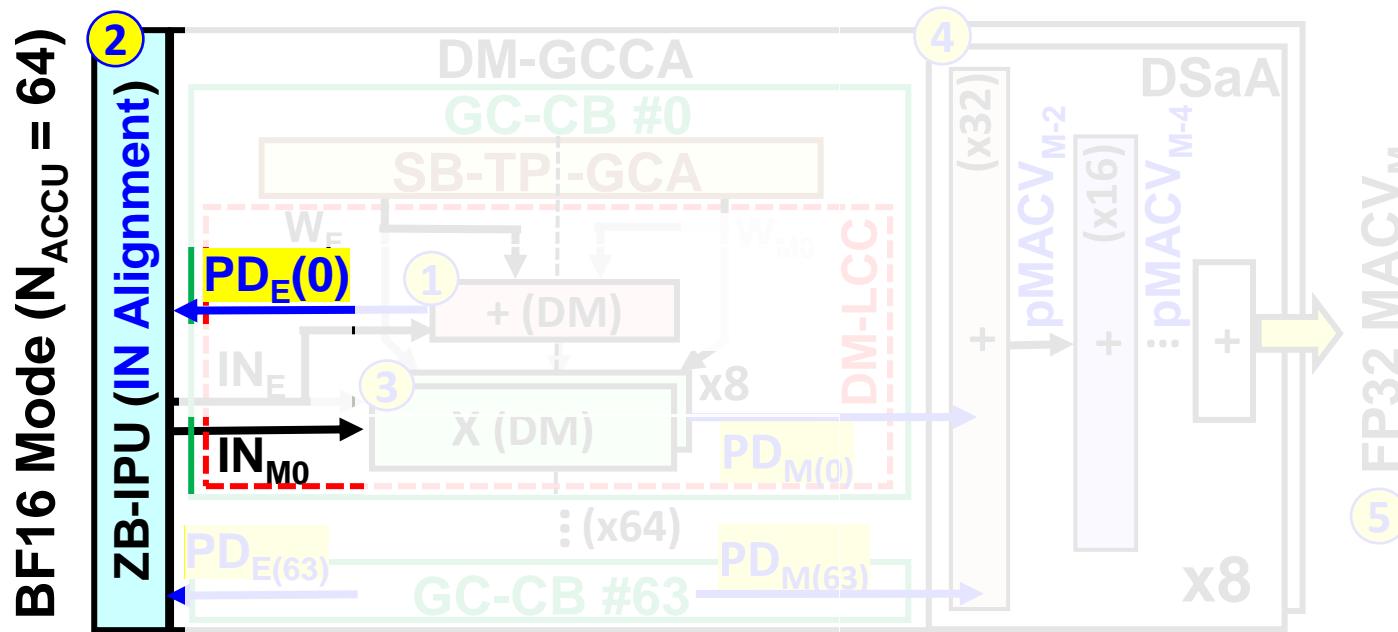
### Floating-point-MAC



# Computation Flow (Floating-Point Mode)

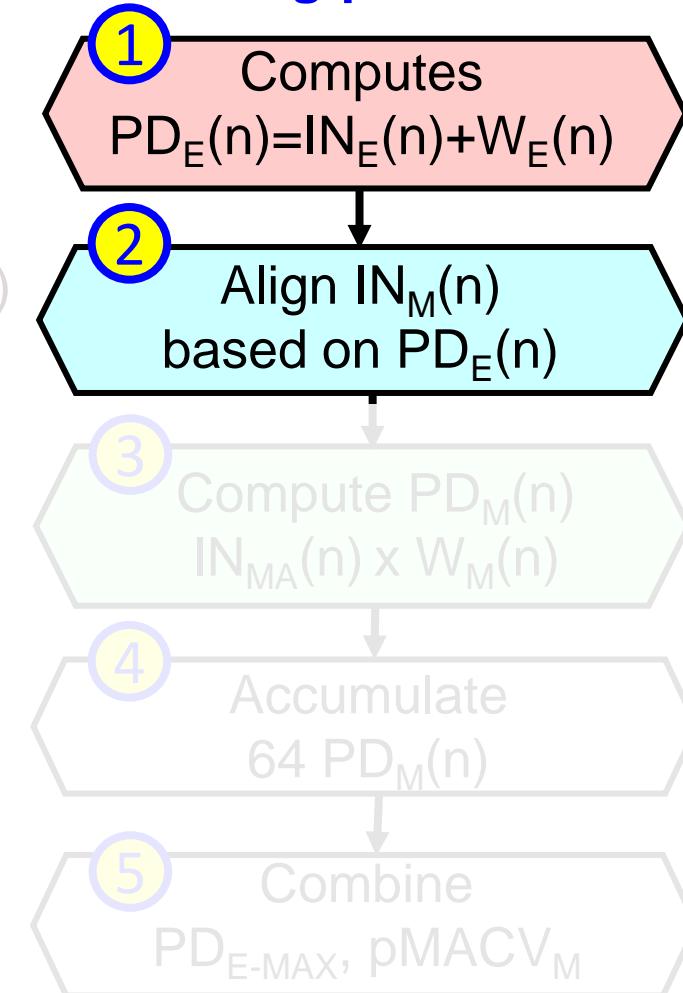
## ■ Floating-point-MAC operation (Five compute phases)

- Phase 1: Computes  $PD_E(n) = (IN_E(n) + W_E(n))$  [n=0~63]
- Phase 2: Aligns  $IN_M(n)$  based on  $PD_E(n)$
- Phase 3: Computes  $PD_M(n) = IN_{MA}(n) \times W_M(n)$  [n=0~63]
- Phase 4: Accumulates 64  $PD_M(n)$ :  $pMACV_M = \sum_{n=0}^{63} (IN_{MA}(n) \times W_M(n))$
- Phase 5: Combines  $PD_{E-MAX}$ ,  $pMACV_M$  & outputs FP32 MACV



## Computation Flow Chart

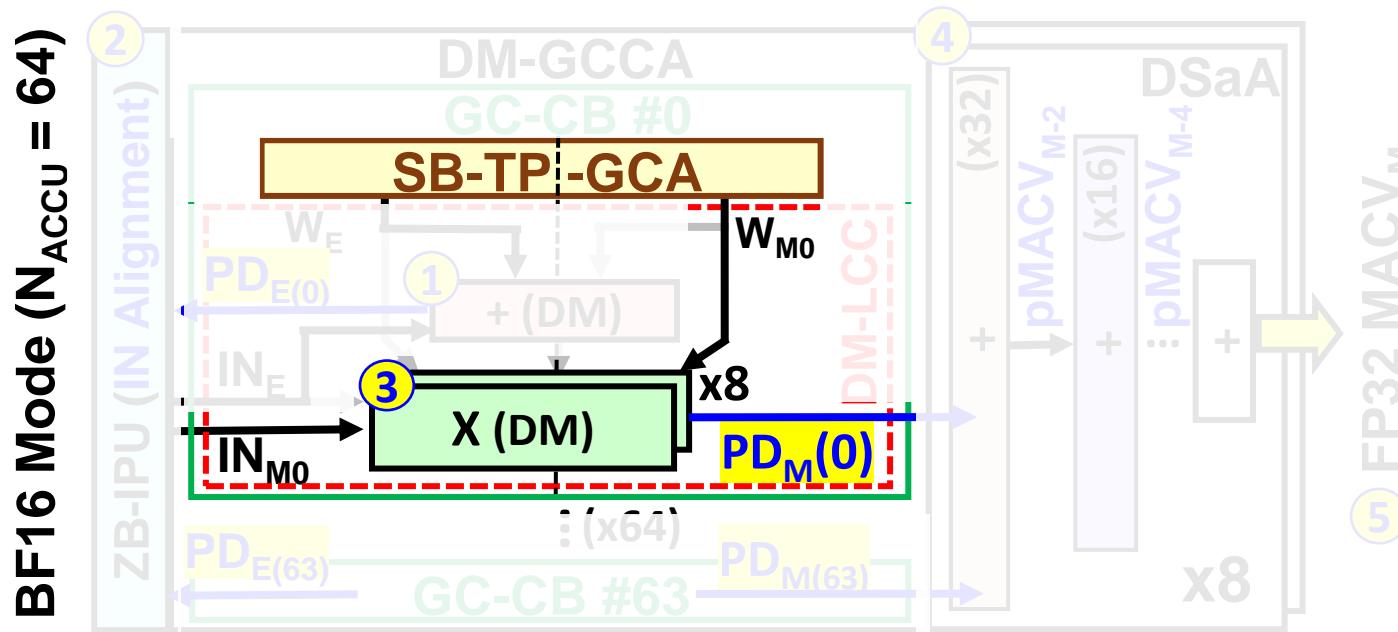
### Floating-point-MAC



# Computation Flow (Floating-Point Mode)

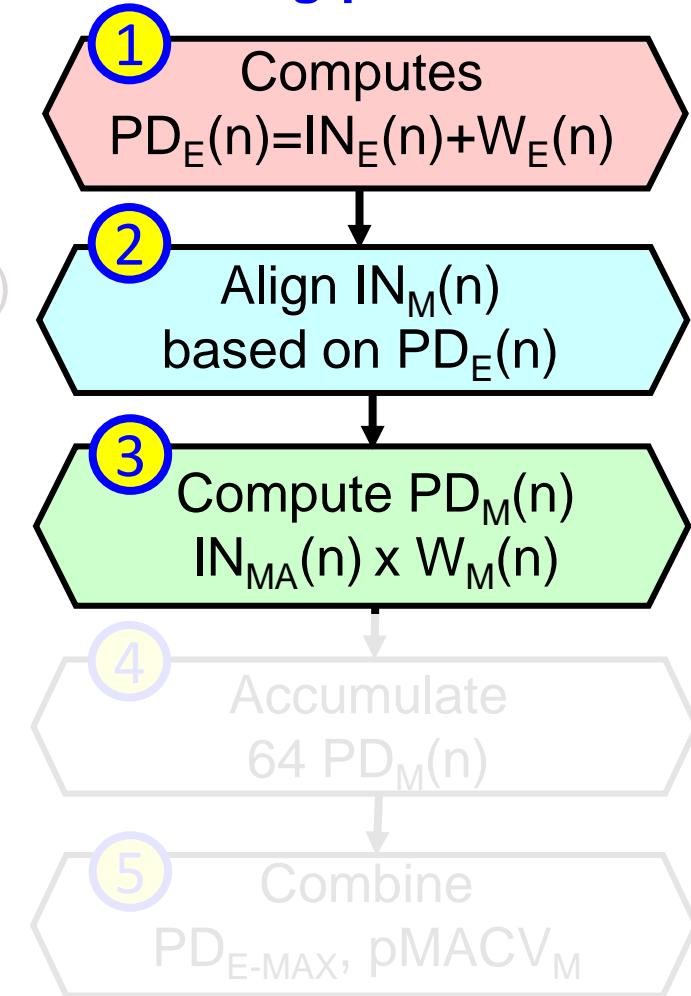
## Floating-point-MAC operation (Five compute phases)

- Phase 1: Computes  $PD_E(n) = (IN_E(n) + W_E(n))$  [n=0~63]
- Phase 2: Aligns  $IN_M(n)$  based on  $PD_E(n)$
- Phase 3: Computes  $PD_M(n) = IN_{MA}(n) \times W_M(n)$  [n=0~63]
- Phase 4: Accumulates 64  $PD_M(n)$ :  $pMACV_M = \sum_{n=0}^{63} (IN_{MA}(n) \times W_M(n))$
- Phase 5: Combines  $PD_{E-MAX}$ ,  $pMACV_M$  & outputs FP32 MACV



## Computation Flow Chart

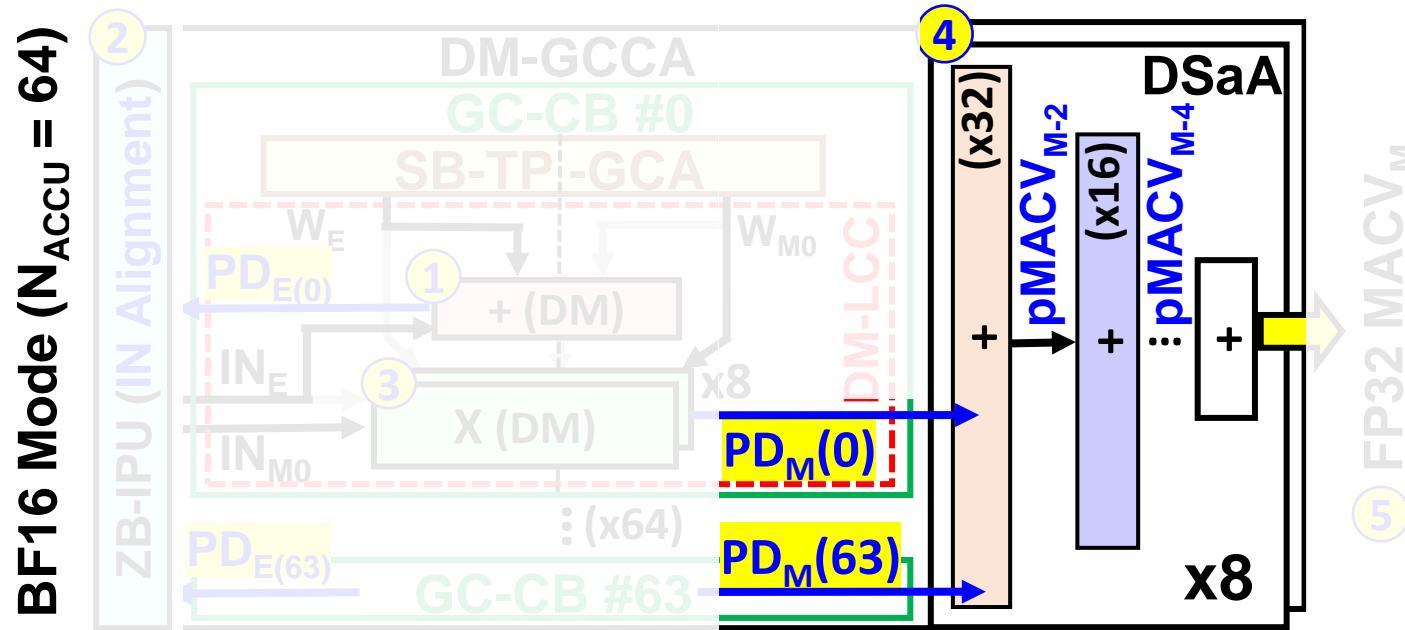
### Floating-point-MAC



# Computation Flow (Floating-Point Mode)

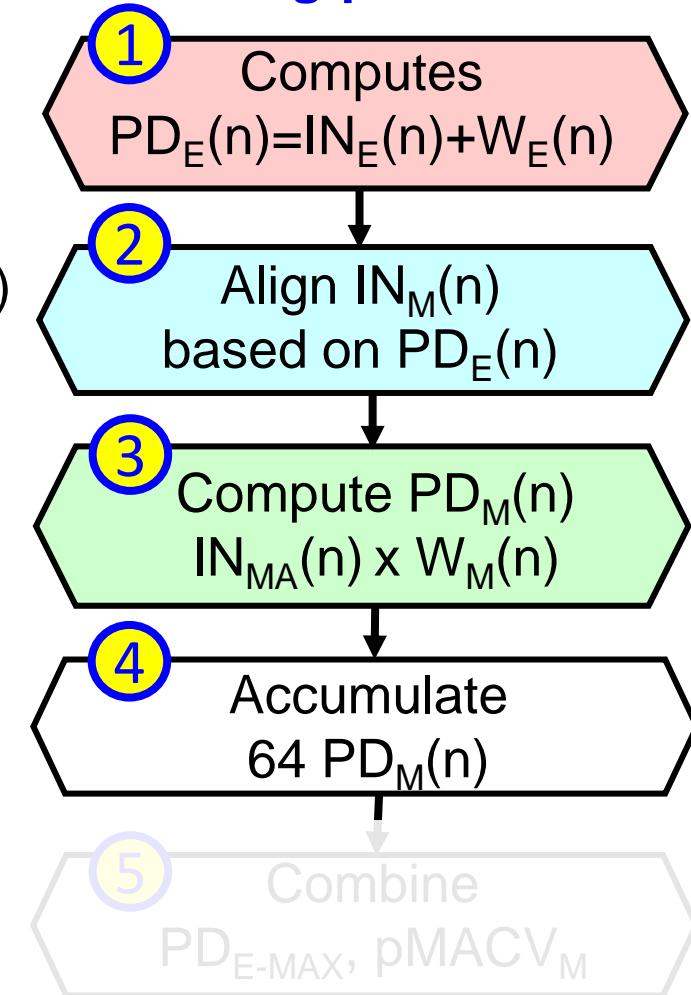
## Floating-point-MAC operation (Five compute phases)

- Phase 1: Computes  $PD_E(n) = (IN_E(n) + W_E(n))$  [n=0~63]
- Phase 2: Aligns  $IN_M(n)$  based on  $PD_E(n)$
- Phase 3: Computes  $PD_M(n) = IN_{MA}(n) \times W_M(n)$  [n=0~63]
- Phase 4: Accumulates 64  $PD_M(n)$ :  $pMACV_M = \sum_{n=0}^{63} (IN_{MA}(n) \times W_M(n))$
- Phase 5: Combines  $PD_{E-MAX}$ ,  $pMACV_M$  & outputs FP32 MACV



## Computation Flow Chart

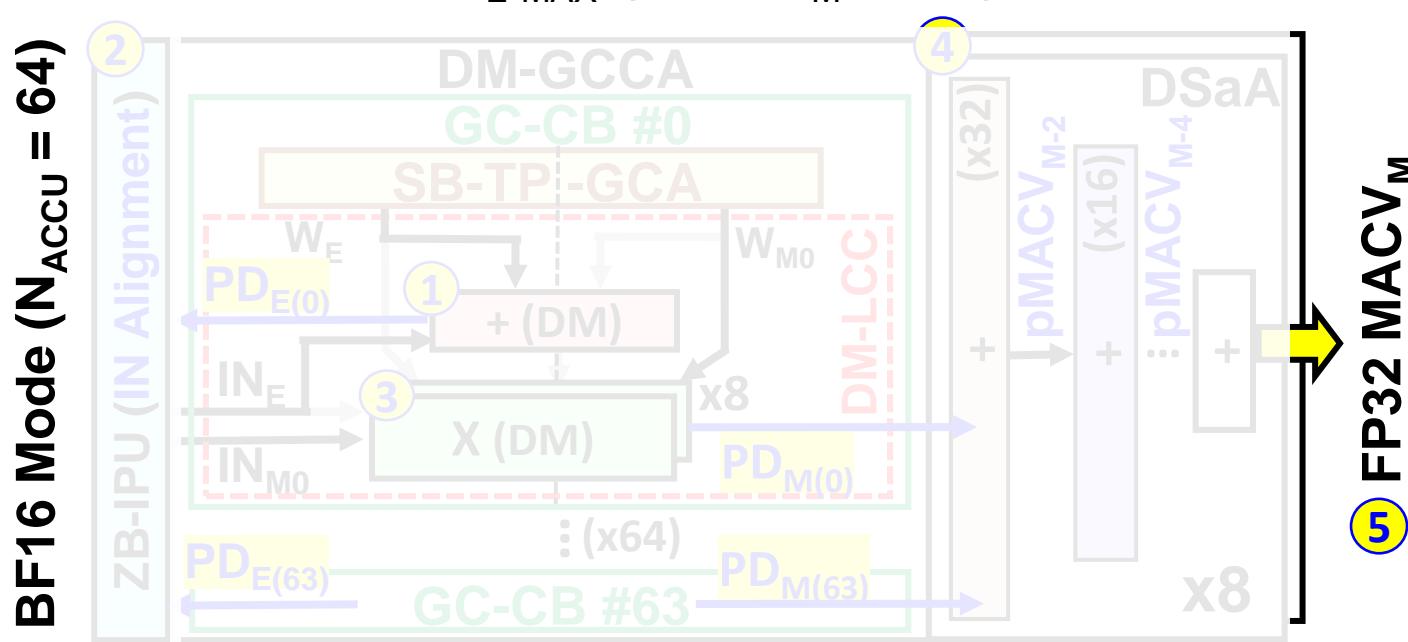
### Floating-point-MAC



# Computation Flow (Floating-Point Mode)

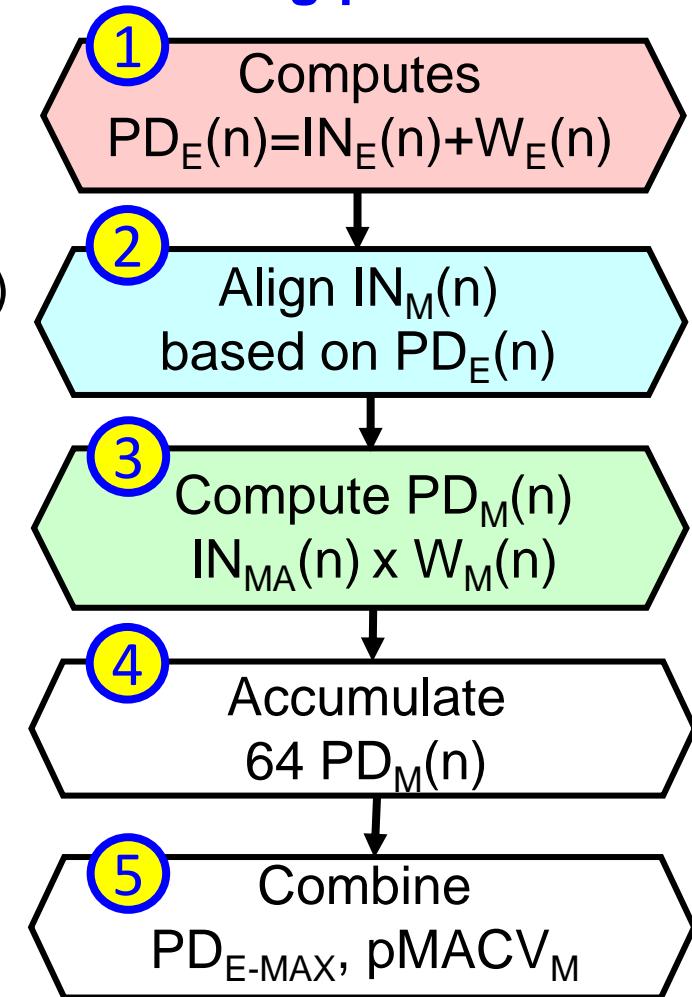
## Floating-point-MAC operation (Five compute phases)

- Phase 1: Computes  $PD_E(n) = (IN_E(n) + W_E(n))$  [n=0~63]
- Phase 2: Aligns  $IN_M(n)$  based on  $PD_E(n)$
- Phase 3: Computes  $PD_M(n) = IN_{MA}(n) \times W_M(n)$  [n=0~63]
- Phase 4: Accumulates 64  $PD_M(n)$ :  $pMACV_M = \sum_{n=0}^{63} (IN_{MA}(n) \times W_M(n))$
- Phase 5: Combines  $PD_{E-MAX}$ ,  $pMACV_M$  & outputs FP32 MACV



## Computation Flow Chart

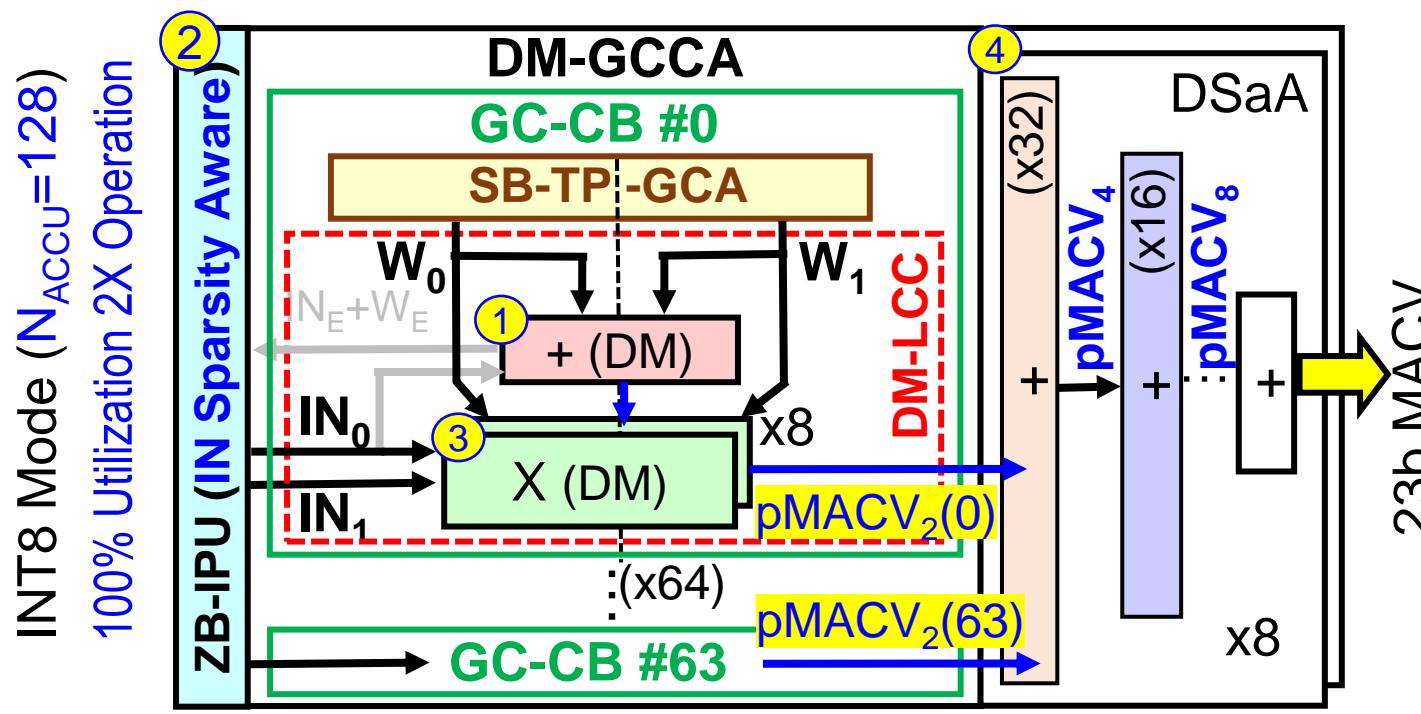
### Floating-point-MAC



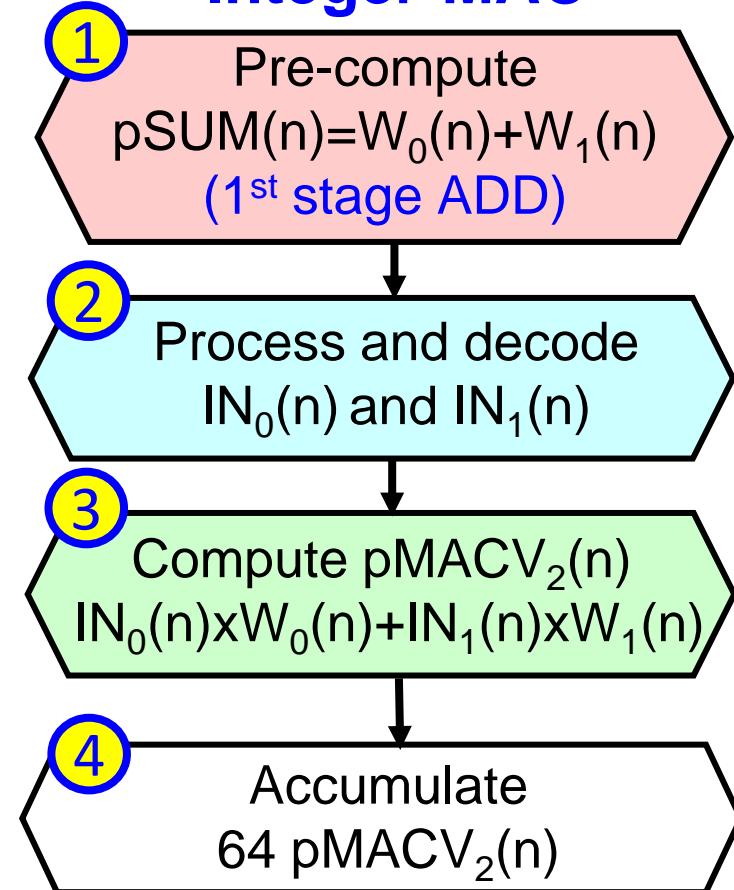
# Computation Flow (Integer Mode)

## ■ Integer-MAC operation (Four compute phases)

- Phase 1: Pre-computes  $p\text{SUM}(n) = (W_0(n) + W_1(n))$  [n=0~63]
- Phase 2: Processes IN Sparsity and  $\text{IN}_0(n)$  &  $\text{IN}_1(n)$  for INT-MAC
- Phase 3: Computes  $p\text{MACV}_2(n)$  [n=0~63]
- Phase 4: Accumulates 64  $p\text{MACV}_2(n)$  [ $\text{MACV} = \sum_{n=0}^{127} (\text{IN}(n) \times W(n))$ ]



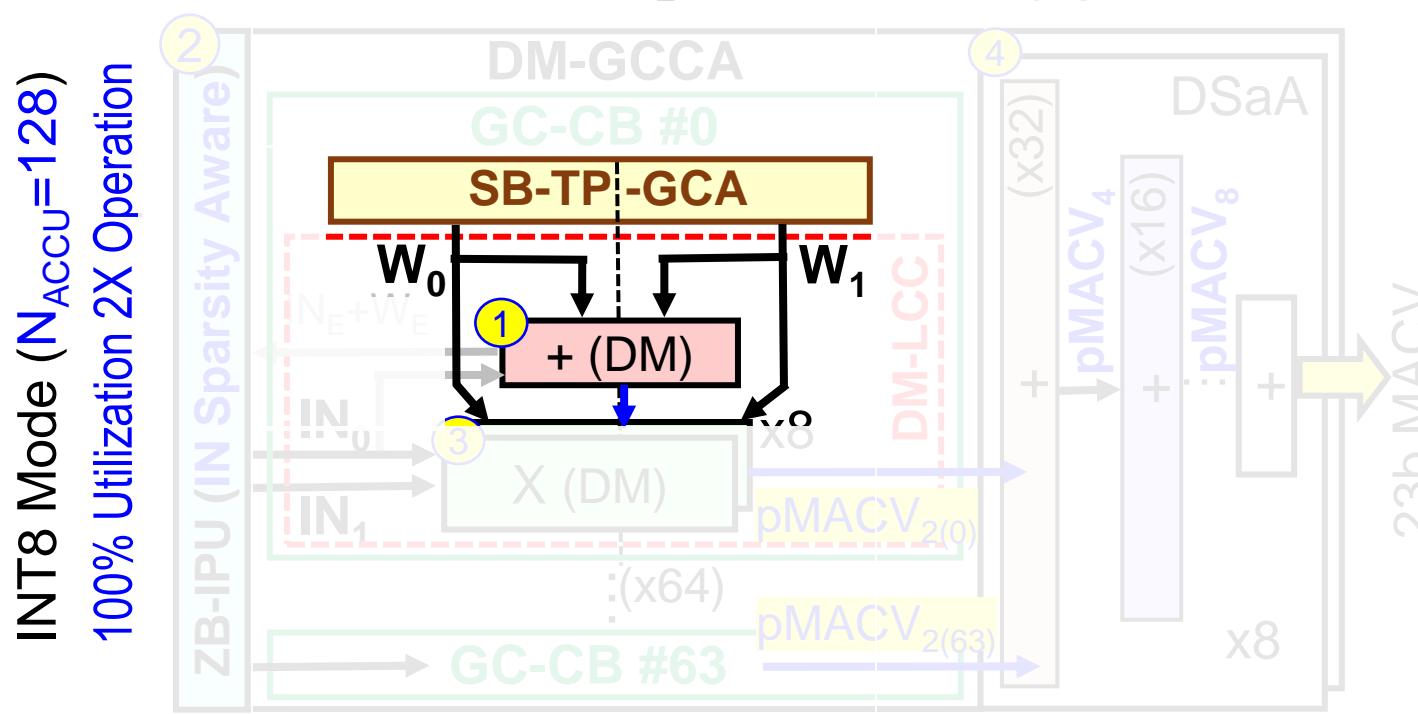
## Computation Flow Chart Integer-MAC



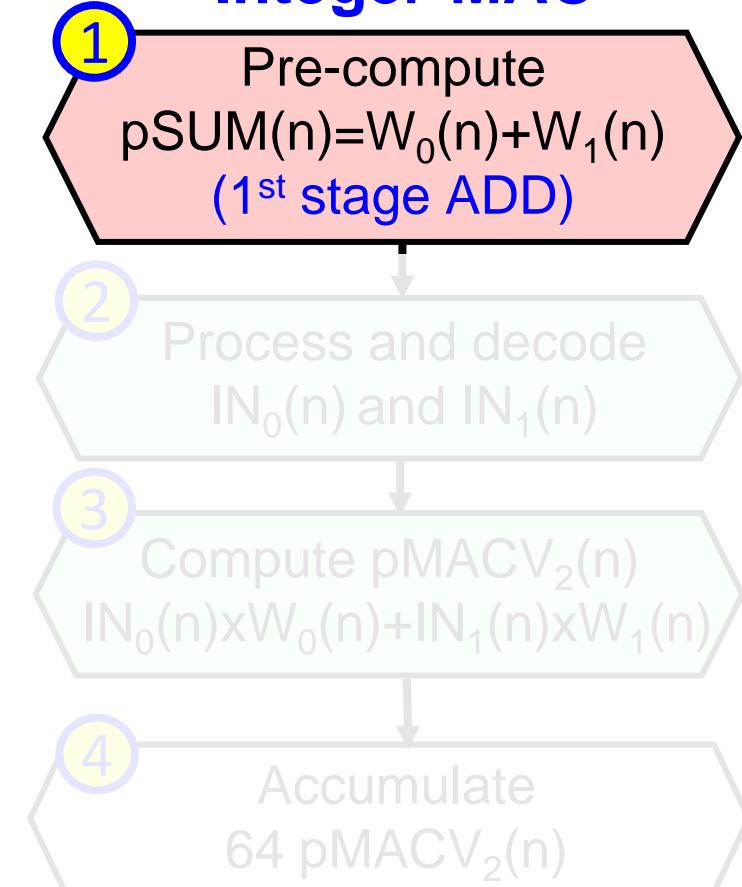
# Computation Flow (Integer Mode)

## ■ Integer-MAC operation (Four compute phases)

- Phase 1: Pre-computes  $p\text{SUM}(n) = (W_0(n) + W_1(n))$  [n=0~63]
- Phase 2: Processes IN Sparsity and  $\text{IN}_0(n)$  &  $\text{IN}_1(n)$  for INT-MAC
- Phase 3: Computes  $p\text{MACV}_2(n)$  [n=0~63]
- Phase 4: Accumulates 64 pMACV<sub>2</sub>(n) [MACV =  $\sum_{n=0}^{127} (\text{IN}(n) \times W(n))$ ]



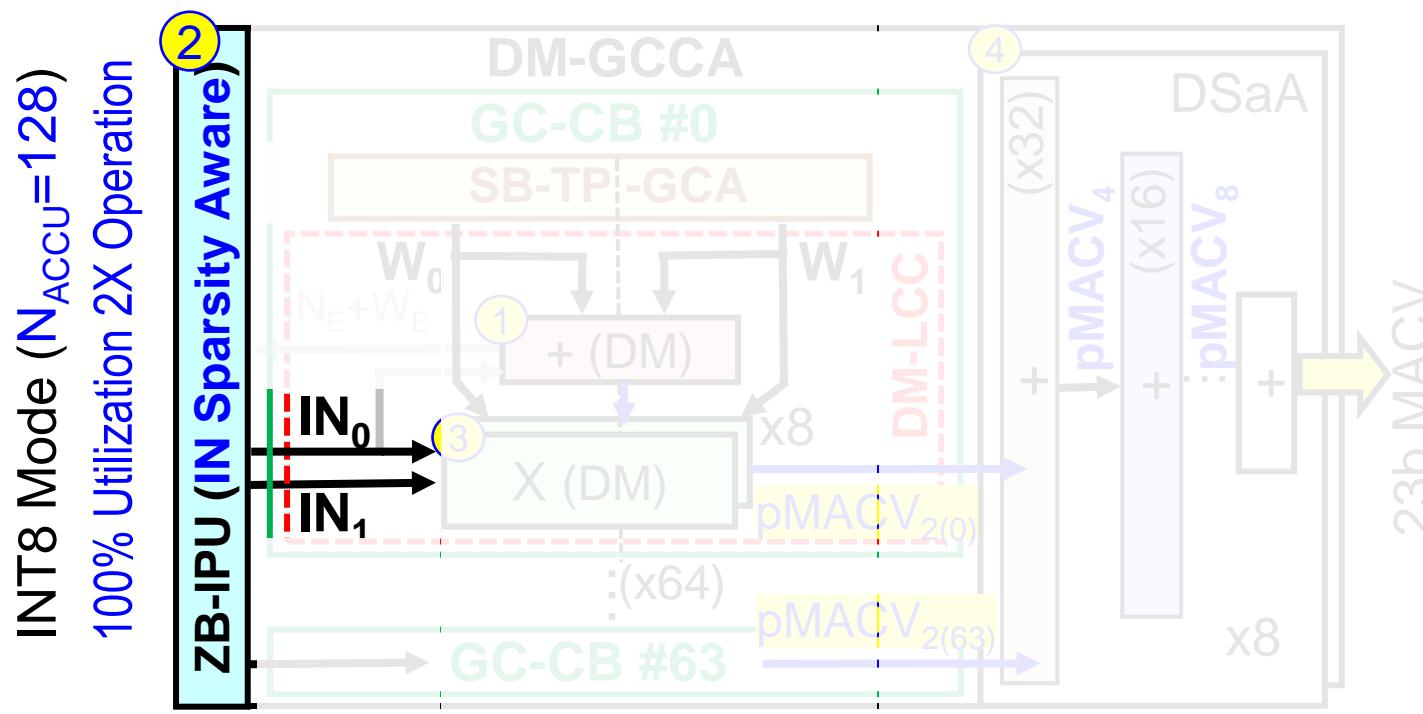
## Computation Flow Chart Integer-MAC



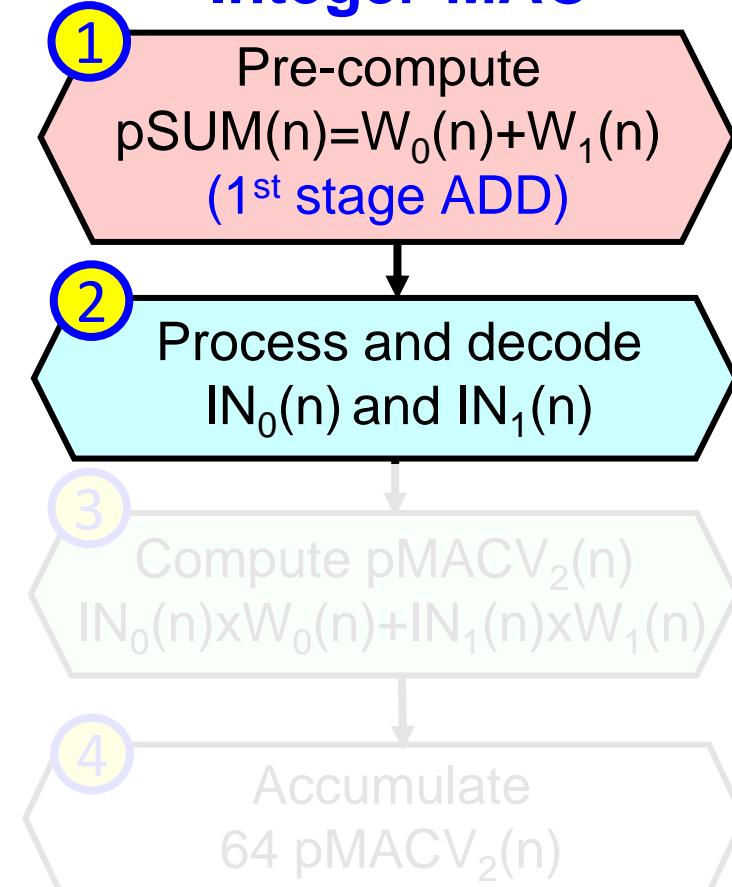
# Computation Flow (Integer Mode)

## ■ Integer-MAC operation (Four compute phases)

- Phase 1: Pre-computes  $p\text{SUM}(n) = (W_0(n) + W_1(n))$  [n=0~63]
- Phase 2: Processes IN Sparsity and  $\text{IN}_0(n)$  &  $\text{IN}_1(n)$  for INT-MAC
- Phase 3: Computes  $p\text{MACV}_2(n)$  [n=0~63]
- Phase 4: Accumulates 64 pMACV<sub>2</sub>(n) [MACV =  $\sum_{n=0}^{127} (\text{IN}(n) \times W(n))$ ]



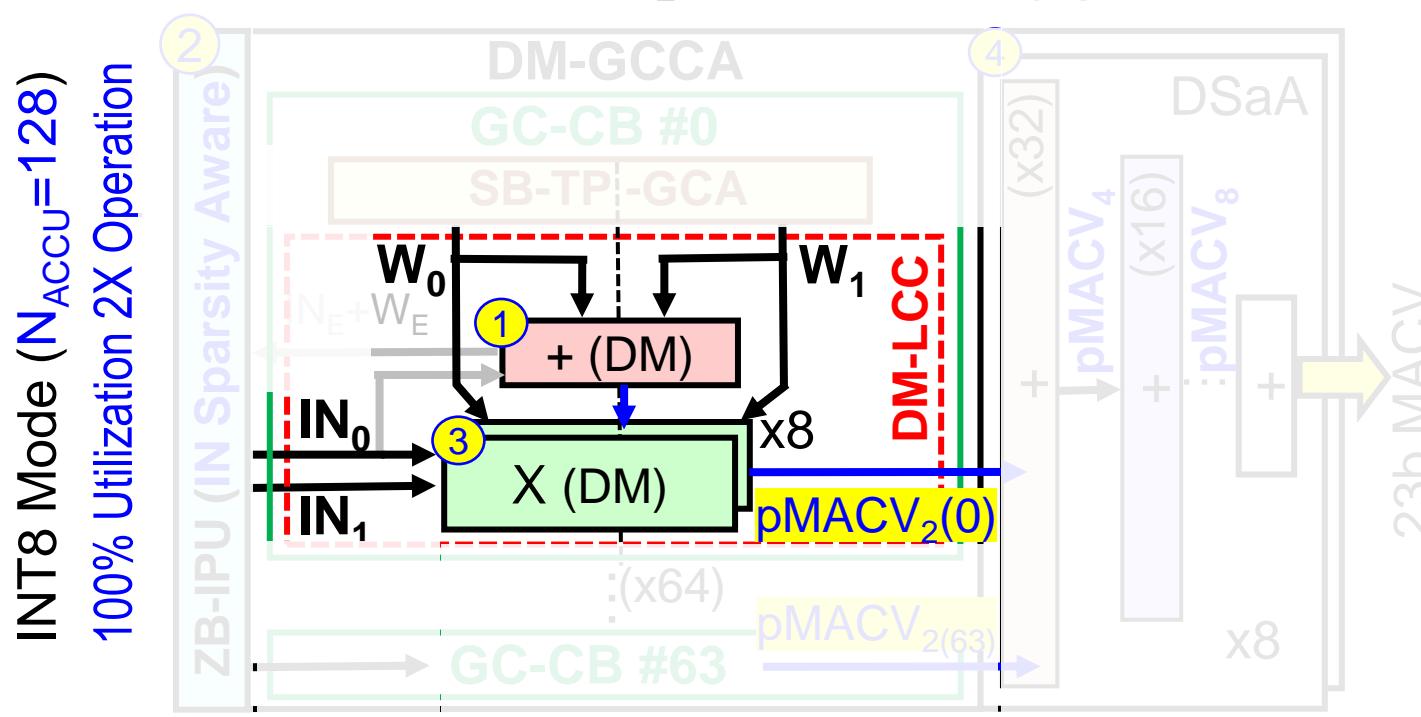
## Computation Flow Chart Integer-MAC



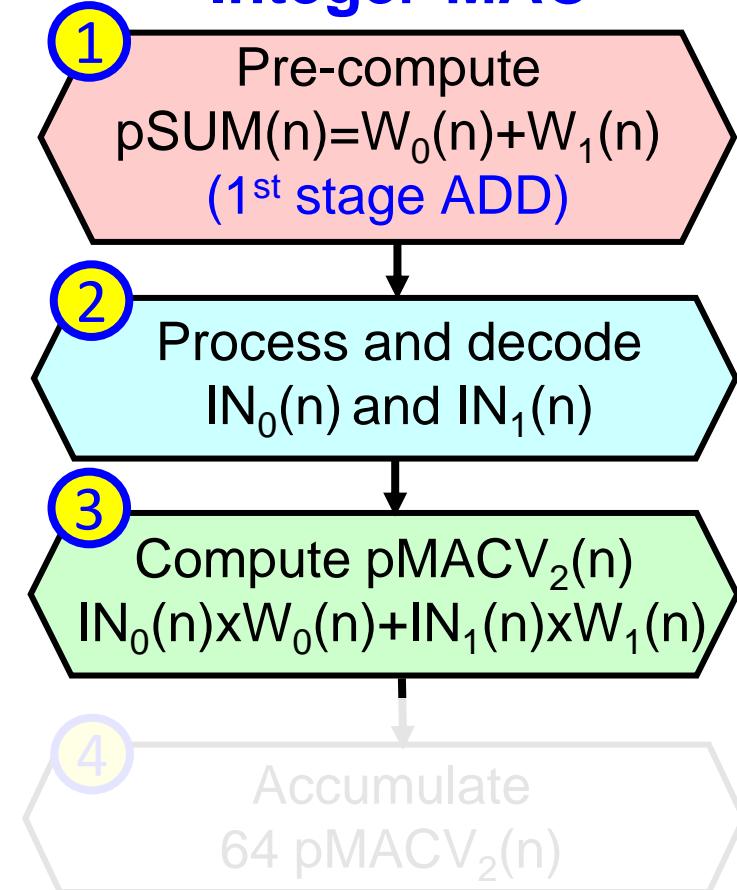
# Computation Flow (Integer Mode)

## ■ Integer-MAC operation (Four compute phases)

- Phase 1: Pre-computes  $p\text{SUM}(n) = (W_0(n) + W_1(n))$  [n=0~63]
- Phase 2: Processes IN Sparsity and  $\text{IN}_0(n)$  &  $\text{IN}_1(n)$  for INT-MAC
- Phase 3: Computes  $p\text{MACV}_2(n)$  [n=0~63]
- Phase 4: Accumulates 64 pMACV<sub>2</sub>(n) [ $\text{MACV} = \sum_{n=0}^{127} (\text{IN}(n) \times W(n))$ ]



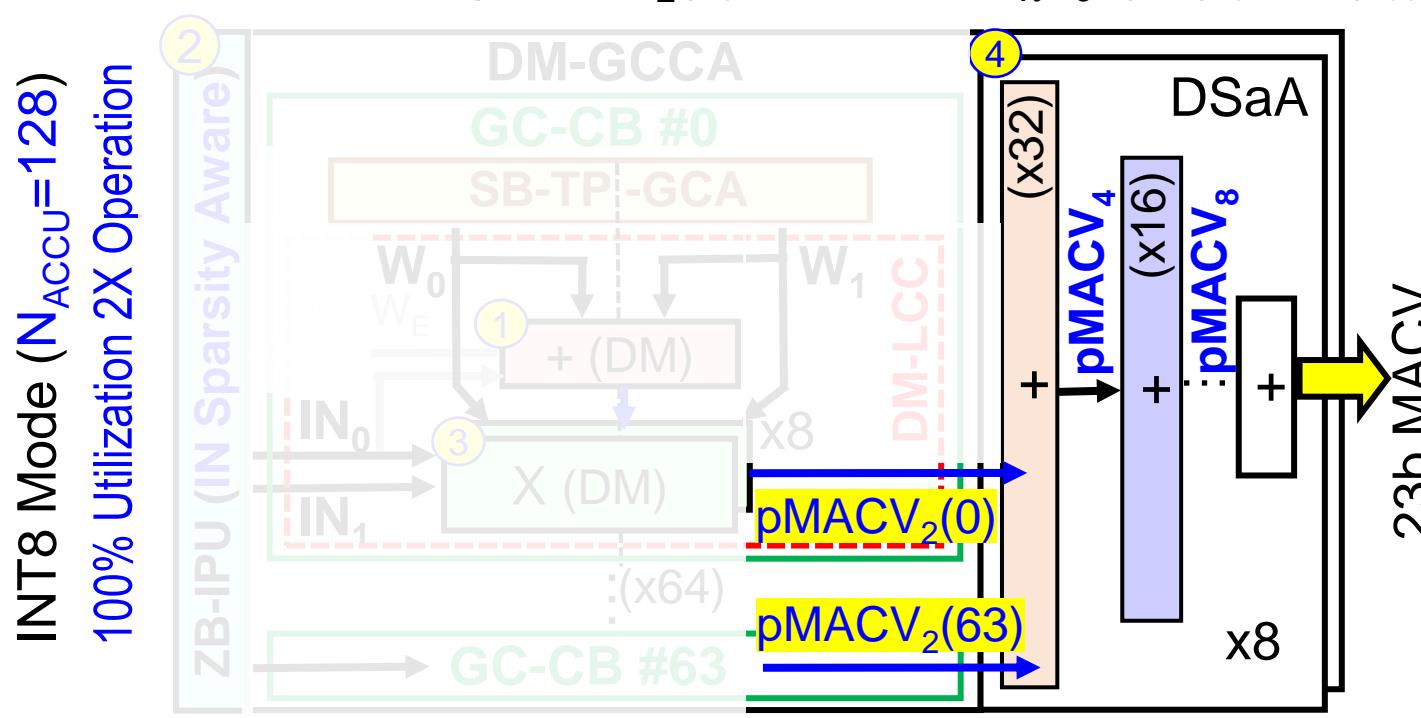
## Computation Flow Chart Integer-MAC



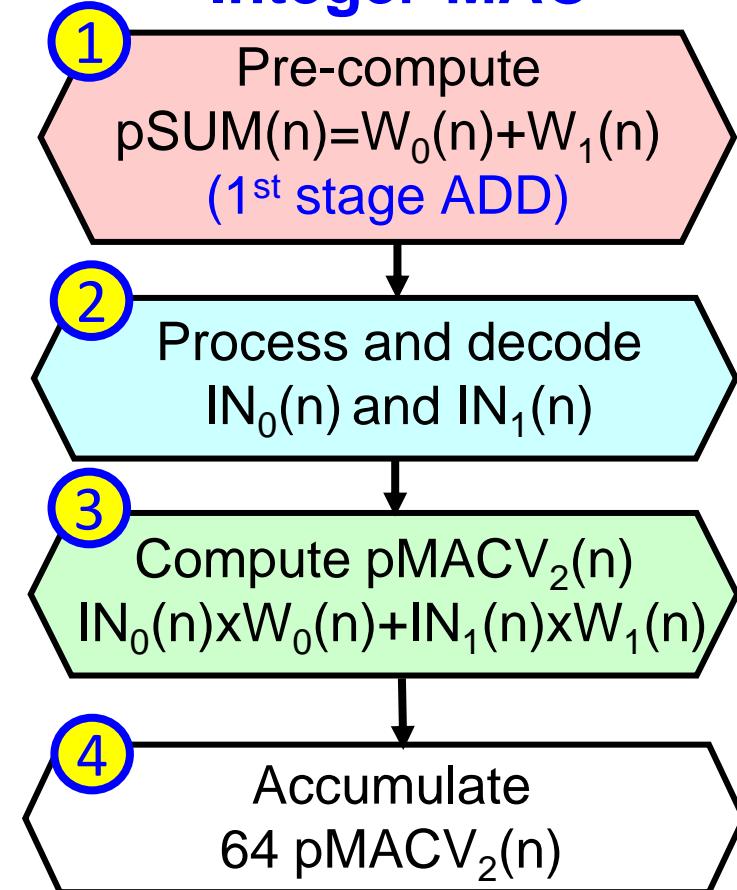
# Computation Flow (Integer Mode)

## ■ Integer-MAC operation (Four compute phases)

- Phase 1: Pre-computes  $p\text{SUM}(n) = (W_0(n) + W_1(n))$  [n=0~63]
- Phase 2: Processes IN Sparsity and  $\text{IN}_0(n)$  &  $\text{IN}_1(n)$  for INT-MAC
- Phase 3: Computes  $p\text{MACV}_2(n)$  [n=0~63]
- Phase 4: Accumulates 64  $p\text{MACV}_2(n)$  [ $\text{MACV} = \sum_{n=0}^{127} (\text{IN}(n) \times W(n))$ ]



## Computation Flow Chart Integer-MAC



# Outline

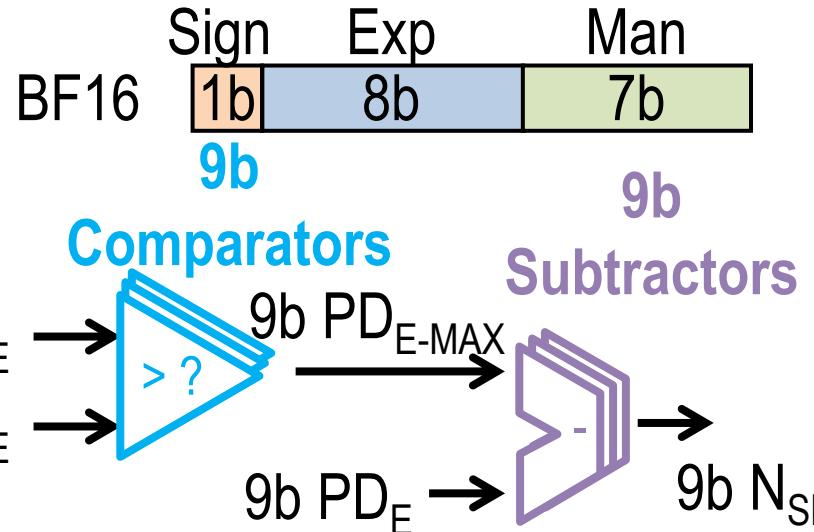
- Challenges of Dual-mode Computing-In-Memory
- **Proposed Computation Gain-cell CIM macro**
  - Overview of Integer / Floating-Point Dual-mode CIM Macro
  - Dual-mode Local-computing-cell (DM-LCC)
  - Dual-mode Zone-based Input Processing Unit Scheme (ZB-IPS)
  - Stationary-based Two-port Gain-cell Array Scheme (SB-TP-GCA)
- Performance and Measurement Results
- Conclusion

# Dual-mode Zone-based Input Processing Unit: Motivation

## Exponent Handling

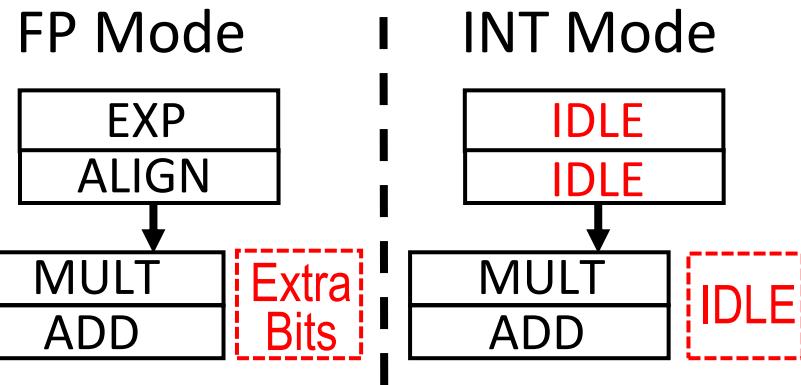
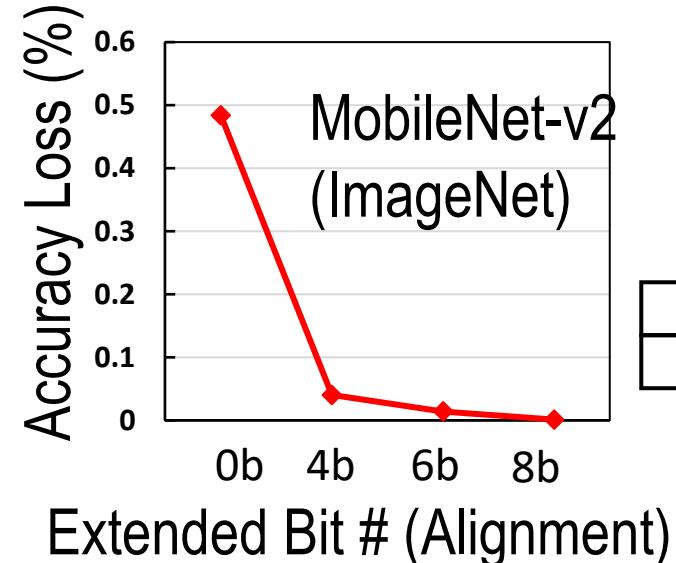
Number of shifting bits

$$N_{SH} = (PD_{E-MAX} - PD_E)$$



$$PD_E = \text{Product Exponent} = IN_E + W_E$$

## Mantissa Bit Extension



Area overhead & low utilization

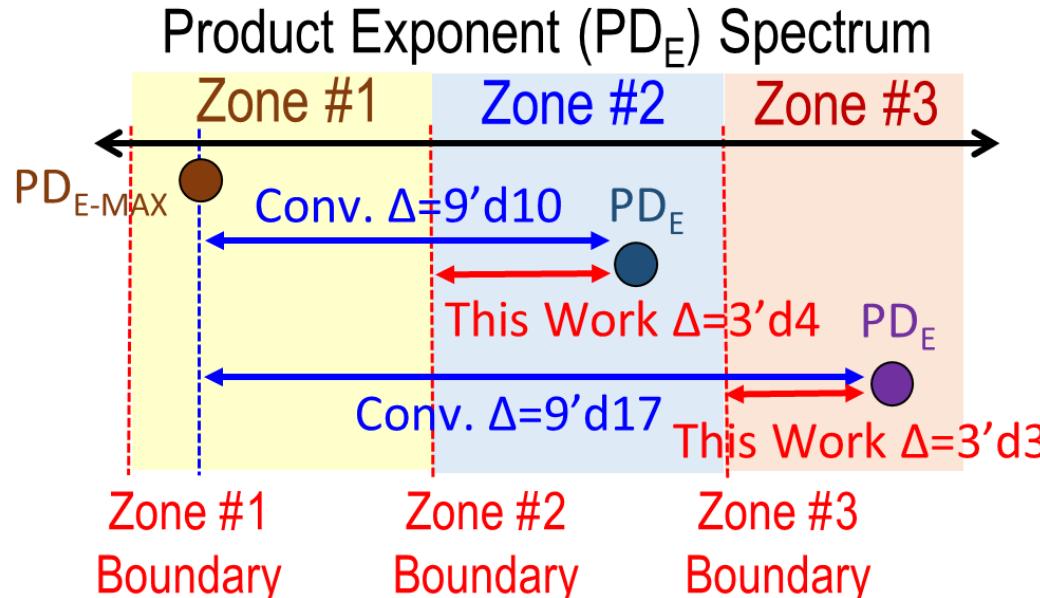
- Two factors that leads to (1) area and energy  $\uparrow$  and (2) hardware utilization  $\downarrow$

- Usage of full exponent bit-width in computing the number of shifting bits
- Mantissa bit extension after alignment to suppress truncation data loss

# Dual-mode Zone-based Input Processing Unit: This Work

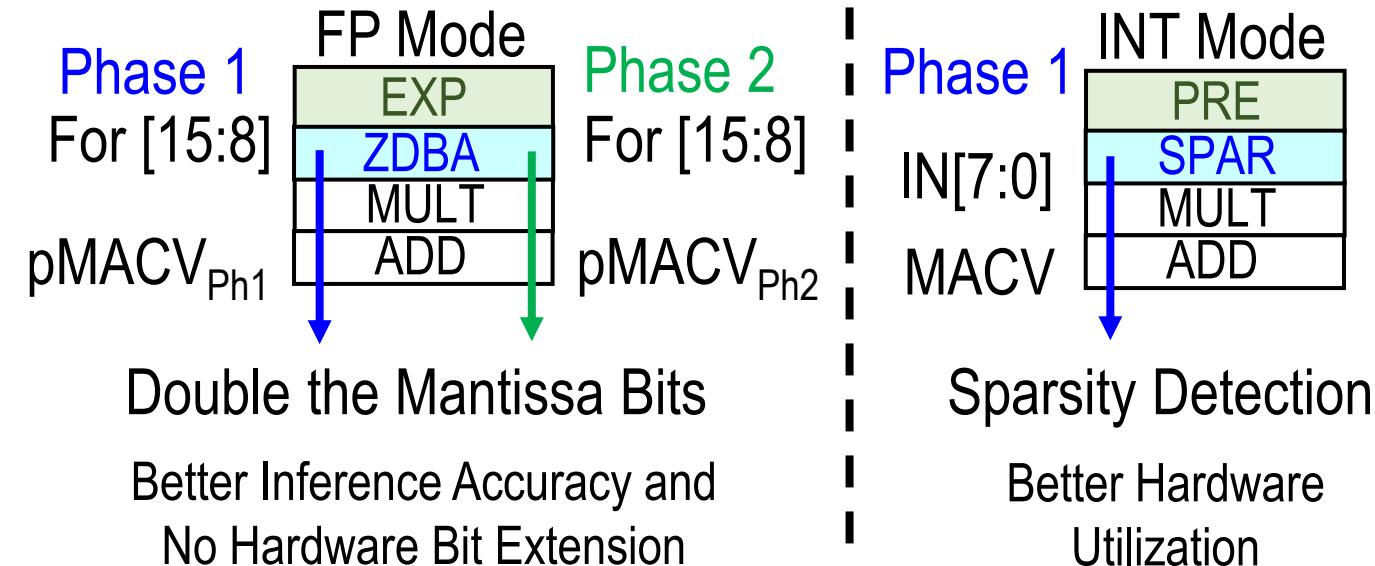
## Exponent Handling

This Work (Zone-based Exponent Handling)



## Mantissa Bit Extension

This Work (2 Phase and Sparsity Detection)



## ■ Zone-based exponent handling

- Divides the product exponent spectrum into zones
- Computes  $N_{SH}$  based on the difference between  $PD_E$  and its zone boundary

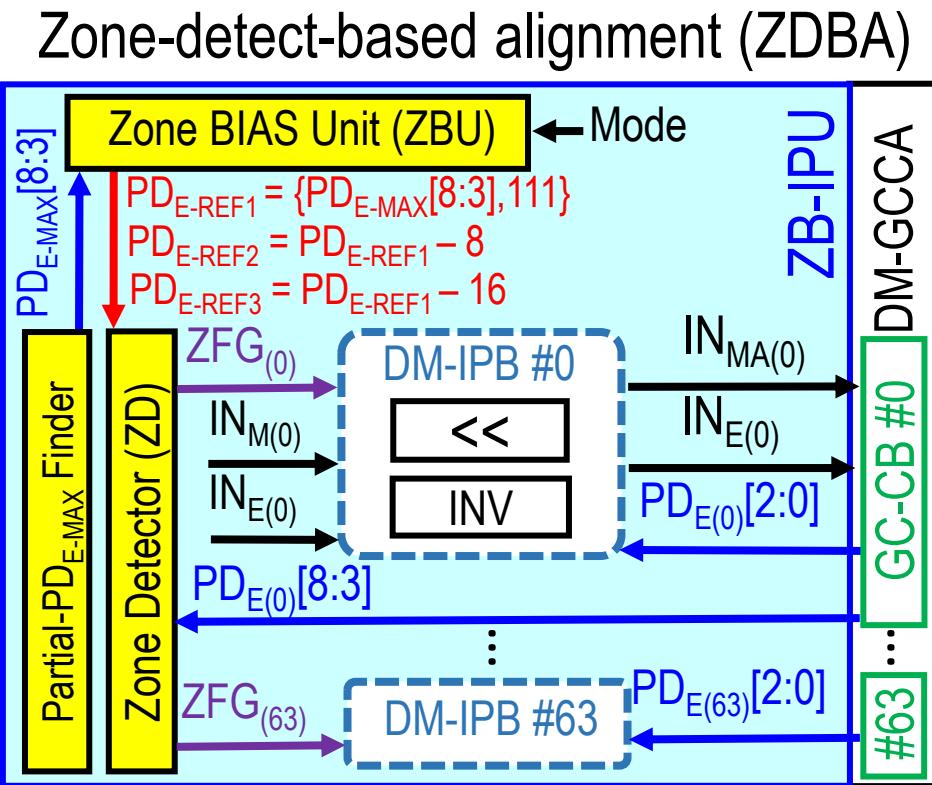
## ■ Mantissa bit extension with phases and re-use hardware for sparsity

- No hardware bit extension and higher hardware utilization

# Dual-mode Zone-based Input Processing Unit: Structure

## ■ Zone-based Input Processing Unit (ZB-IPU)

- Partial-PD<sub>E-MAX</sub> finder (pEMAXF)
  - Finds the MSB-6b (PD<sub>E-MAX</sub>[8:3]) of PD<sub>E-MAX</sub>
- Zone bias unit (ZBU)
  - Generates 3 zone-references (PD<sub>E-REF1~3</sub>)
- Zone detector (ZD)
  - Classifies each PD<sub>E(n)</sub> into one of the three zones
- 64 Dual-mode input processing block (DM-IPB)
  - Aligns the IN<sub>M</sub> according to the zone-shift number (N<sub>SHZ</sub>)

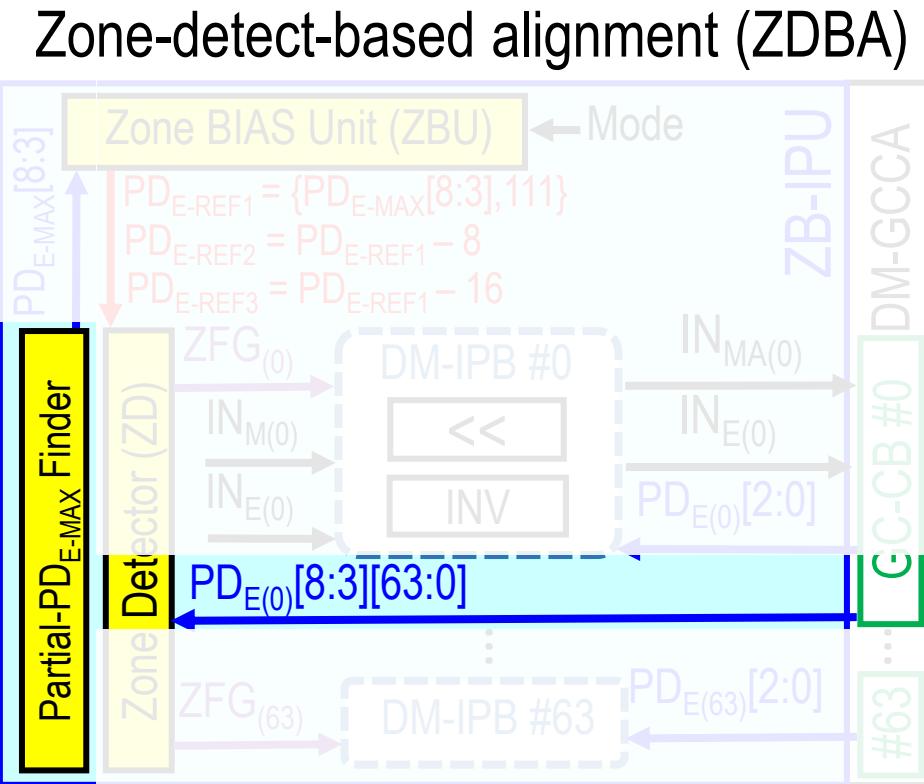


Step-by-Step example will be provided later

# Dual-mode Zone-based Input Processing Unit: Structure

## ■ Zone-based Input Processing Unit (ZB-IPU)

- Partial-PD<sub>E-MAX</sub> finder (pEMAXF)
  - Finds the MSB-6b (PD<sub>E-MAX</sub>[8:3]) of PD<sub>E-MAX</sub>
- Zone bias unit (ZBU)
  - Generates 3 zone-references (PD<sub>E-REF1~3</sub>)
- Zone detector (ZD)
  - Classifies each PD<sub>E(n)</sub> into one of the three zones
- 64 Dual-mode input processing block (DM-IPB)
  - Aligns the IN<sub>M</sub> according to the zone-shift number (N<sub>SHZ</sub>)

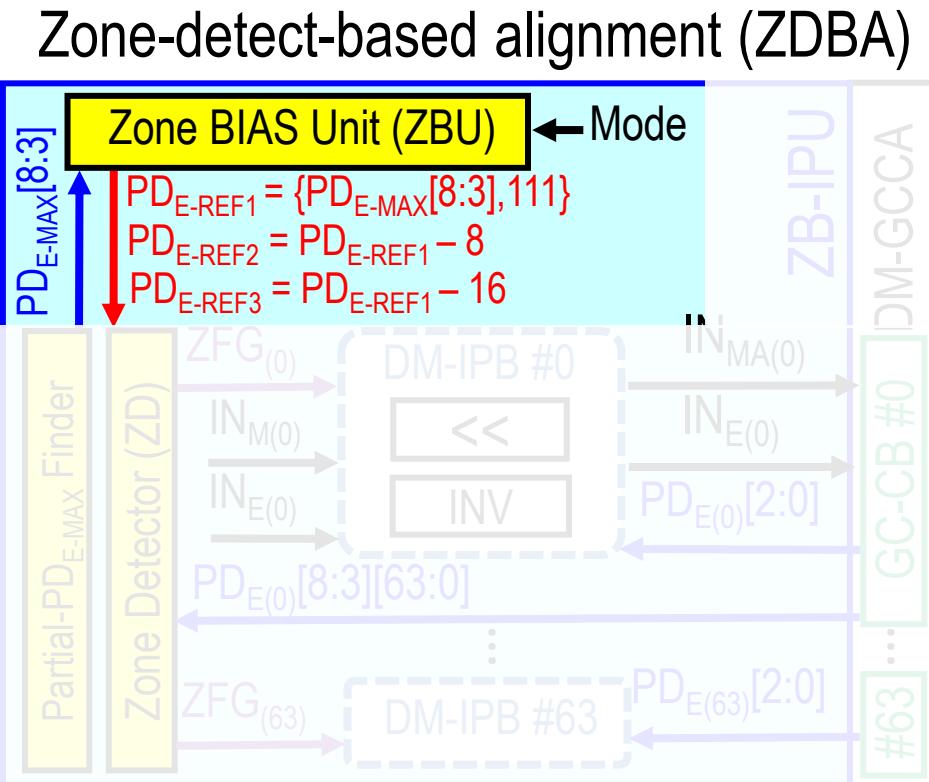


Step-by-Step example will be provided later

# Dual-mode Zone-based Input Processing Unit: Structure

## ■ Zone-based Input Processing Unit (ZB-IPU)

- Partial-PD<sub>E-MAX</sub> finder (pEMAXF)
  - Finds the MSB-6b (PD<sub>E-MAX[8:3]</sub>) of PD<sub>E-MAX</sub>
- Zone bias unit (ZBU)
  - Generates 3 zone-references (PD<sub>E-REF1~3</sub>)
- Zone detector (ZD)
  - Classifies each PD<sub>E(n)</sub> into one of the three zones
- 64 Dual-mode input processing block (DM-IPB)
  - Aligns the IN<sub>M</sub> according to the zone-shift number (N<sub>SHZ</sub>)

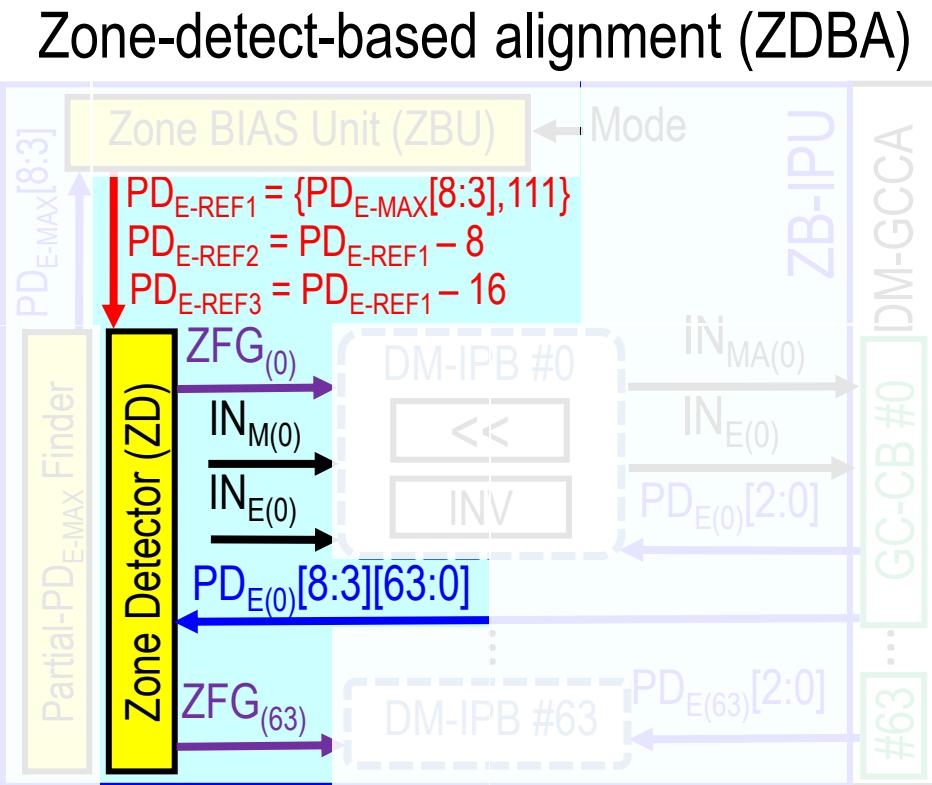


Step-by-Step example will be provided later

# Dual-mode Zone-based Input Processing Unit: Structure

## ■ Zone-based Input Processing Unit (ZB-IPU)

- Partial-PD<sub>E-MAX</sub> finder (pEMAXF)
  - Finds the MSB-6b (PD<sub>E-MAX[8:3]</sub>) of PD<sub>E-MAX</sub>
- Zone bias unit (ZBU)
  - Generates 3 zone-references (PD<sub>E-REF1~3</sub>)
- Zone detector (ZD)
  - Classifies each PD<sub>E(n)</sub> into one of the three zones
- 64 Dual-mode input processing block (DM-IPB)
  - Aligns the IN<sub>M</sub> according to the zone-shift number (N<sub>SHZ</sub>)



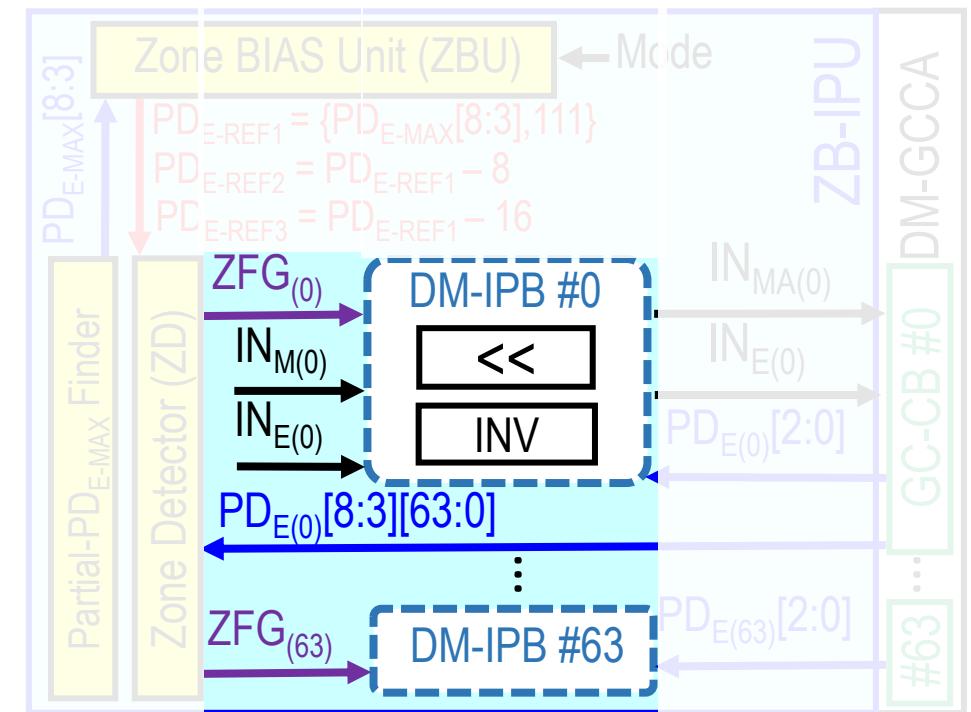
Step-by-Step example will be provided later

# Dual-mode Zone-based Input Processing Unit: Structure

## ■ Zone-based Input Processing Unit (ZB-IPU)

- Partial-PD<sub>E-MAX</sub> finder (pEMAXF)
  - Finds the MSB-6b (PD<sub>E-MAX</sub>[8:3]) of PD<sub>E-MAX</sub>
- Zone bias unit (ZBU)
  - Generates 3 zone-references (PD<sub>E-REF1~3</sub>)
- Zone detector (ZD)
  - Classifies each PD<sub>E(n)</sub> into one of the three zones
- 64 Dual-mode input processing block (DM-IPB)
  - Aligns the IN<sub>M</sub> according to the zone-shift number (N<sub>SHZ</sub>)

### Zone-detect-based alignment (ZDBA)



Step-by-Step example will be provided later

# Dual-mode Zone-based Input Processing Unit: Example

## ■ Two stages operation (FP-MAC)

- Stage 1:

- pEMAXF finds the MSB-6b ( $PD_{E-MAX}[8:3]$ ) of  $PD_{E-MAX}$

- Example:

$$PD_E(0) = 253 = 011111101 = PD_{E-MAX}$$

$$PD_E(1) = 243 = 0111110011$$

...

$$PD_E(64) = 236 = 011101100$$

- ZBU then generates 3 zone-references ( $PD_{E-REF1 \sim 3}$ ) according to  $PD_{E-MAX}[8:3]$

- Example:

$$PD_{E-REF1} = \{PD_{E-MAX}[8:3], 111\} = 255$$

$$PD_{E-REF2} = PD_{E-REF1} - 8 = 247$$

$$PD_{E-REF3} = PD_{E-REF1} - 16 = 239$$

$PD_E$

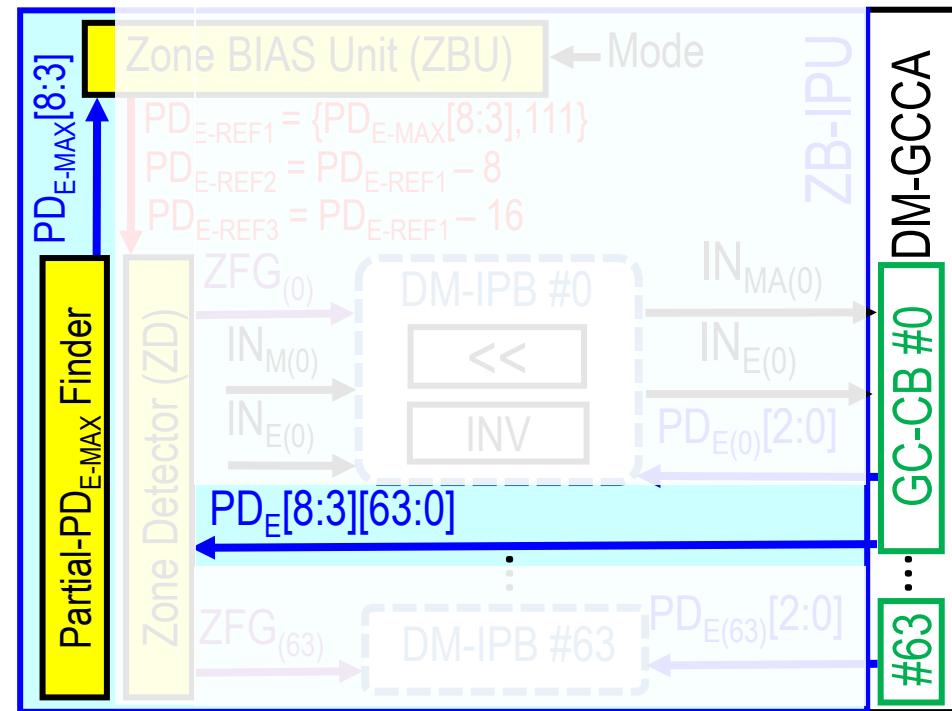
$PD_E(0)=253$

$PD_E(1)=243$        $PD_{E-MAX} = PD_E(0)$

$\vdots x64$

$PD_E(63)=236$

## Zone-detect-based alignment (ZDBA)



# Dual-mode Zone-based Input Processing Unit: Example

## ■ Two stages operation (FP-MAC)

- Stage 1:

- pEMAXF finds the MSB-6b ( $PD_{E-MAX}[8:3]$ ) of  $PD_{E-MAX}$

➤ Example:

$$PD_E(0) = 253 = \text{011111}101 = PD_{E-MAX}$$

$$PD_E(1) = 243 = \text{011110}011$$

...

$$PD_E(64) = 236 = \text{011101}100$$

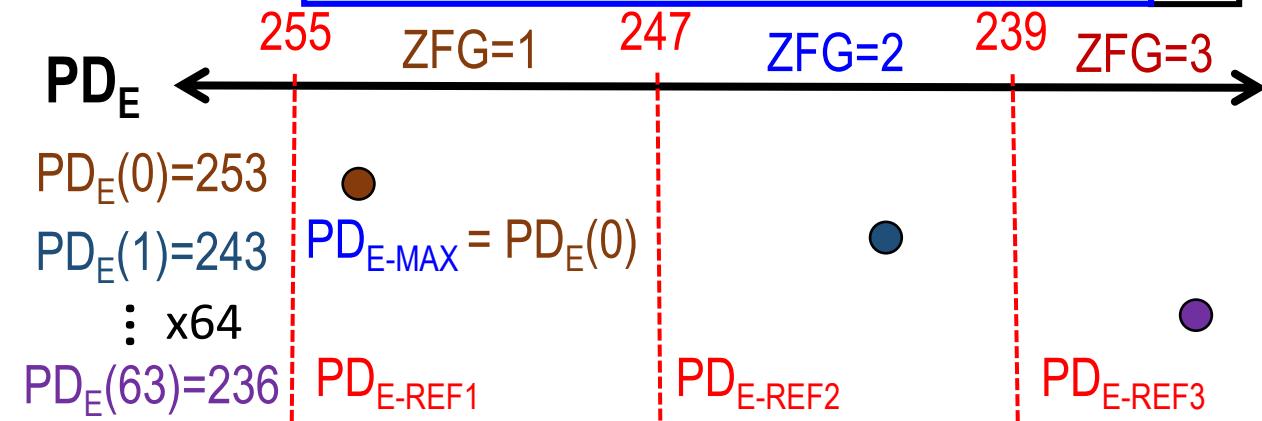
- ZBU then generates 3 zone-references ( $PD_{E-REF1\sim 3}$ ) according to  $PD_{E-MAX}[8:3]$

➤ Example:

$$PD_{E-REF1} = \{PD_{E-MAX}[8:3], \text{111}\} = 255$$

$$PD_{E-REF2} = PD_{E-REF1} - 8 = 247$$

$$PD_{E-REF3} = PD_{E-REF1} - 16 = 239$$



## Zone-detect-based alignment (ZDBA)

# Dual-mode Zone-based Input Processing Unit: Example

## ■ Two stages operation (FP-MAC)

- Stage 2:

- ZD Classified each  $PD_E(N)$  into one of the three zones based on its zone-flag (ZFG)

➤ Example:

$$PD_E(0) = 253 = 011111101 = (ZFG=1)$$

$$PD_E(1) = 243 = 011110011 = (ZFG=2)$$

...

$$PD_E(64) = 236 = 011101100 = (ZFG=3)$$

- DM-IPB computes the zone-shift number ( $N_{SHZ}$ ) and aligns the  $IN_M$  accordingly, where  $N_{SHZ}$  is the inverse of  $PD_E[2:0]$  (LSB3b)

➤ Example:

$$PD_E(0)[2:0] = 3'b010 = PD_{E-REF1} - PD_E(0)$$

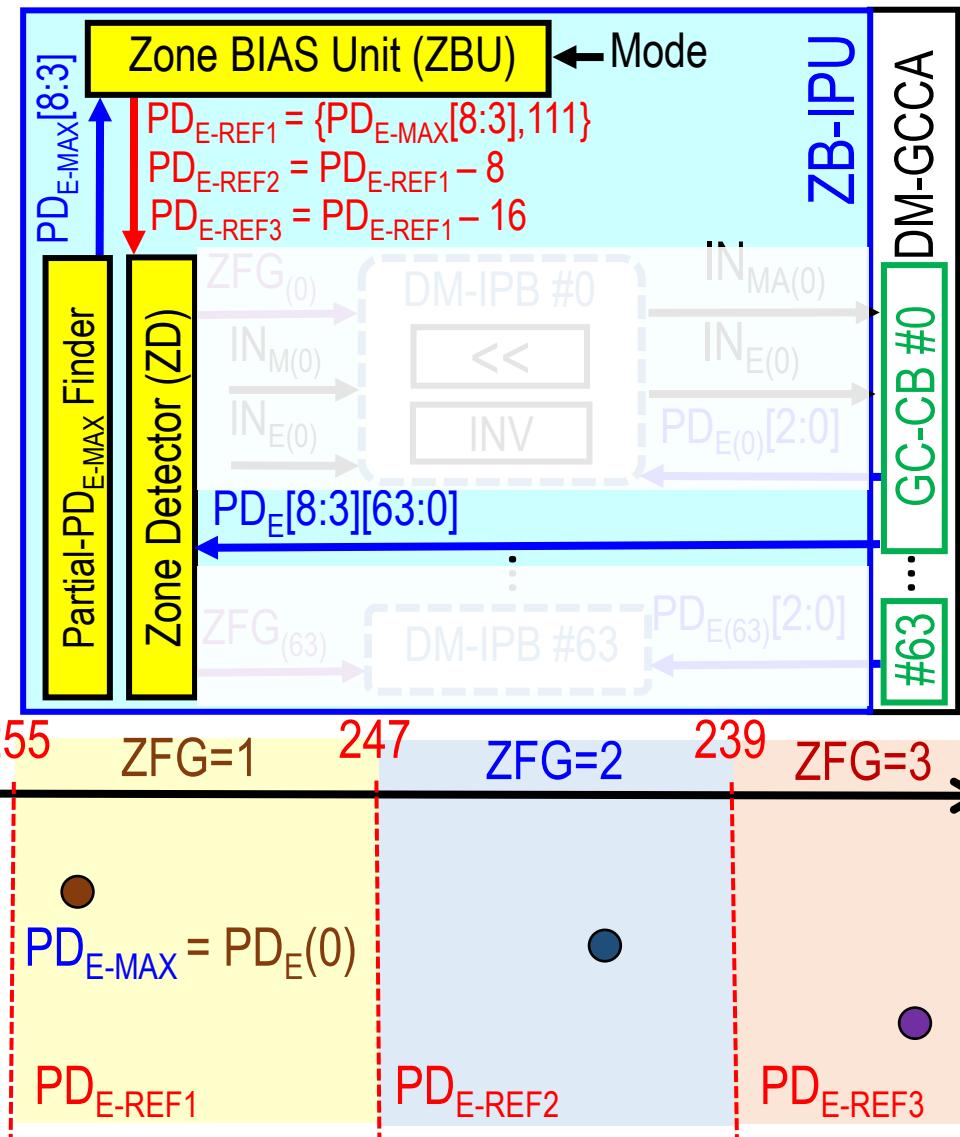
$$PD_E(1)[2:0] = 3'b100 = PD_{E-REF2} - PD_E(1)$$

...

$$PD_E(64)[2:0] = 3'b011 = PD_{E-REF3} - PD_E(2)$$

$PD_E \leftarrow$   
 $PD_E(0)=253$   
 $PD_E(1)=243$   
 $\vdots x64$   
 $PD_E(63)=236$

## Zone-detect-based alignment (ZDBA)



# Dual-mode Zone-based Input Processing Unit: Example

## ■ Two stages operation (FP-MAC)

- Stage 2:

- ZD Classified each  $PD_E(N)$  into one of the three zones based on its zone-flag (ZFG)

➤ Example:

$$PD_E(0) = 253 = 011111101 = (ZFG=1)$$

$$PD_E(1) = 243 = 011110011 = (ZFG=2)$$

...

$$PD_E(64) = 236 = 011101100 = (ZFG=3)$$

- DM-IPB computes the zone-shift number ( $N_{SHZ}$ ) and aligns the  $IN_M$  accordingly, where  $N_{SHZ}$  is the inverse of  $PD_E[2:0]$  (LSB3b)

➤ Example:

$$\underline{PD_E(0)[2:0]} = 3'b010 = PD_{E-REF1} - PD_E(0)$$

$$\underline{PD_E(1)[2:0]} = 3'b100 = PD_{E-REF2} - PD_E(1)$$

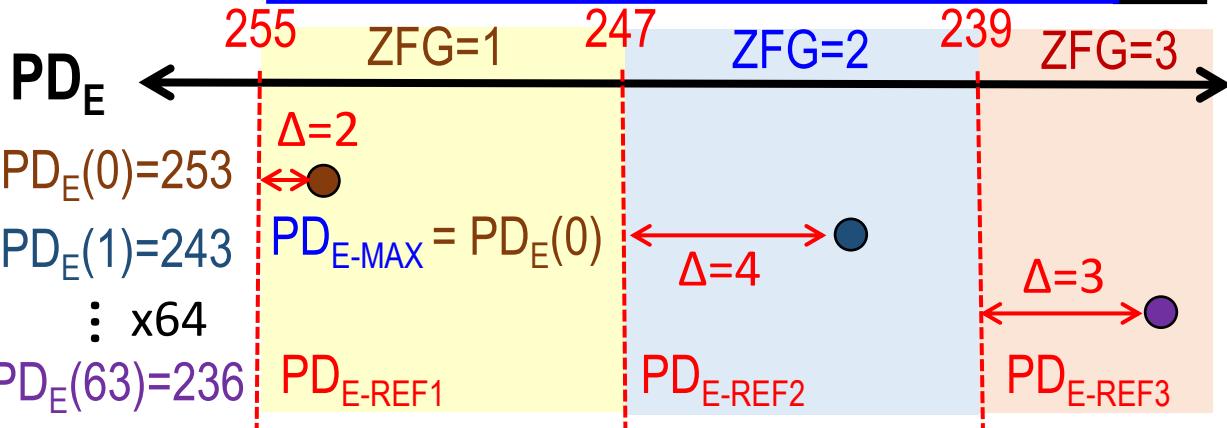
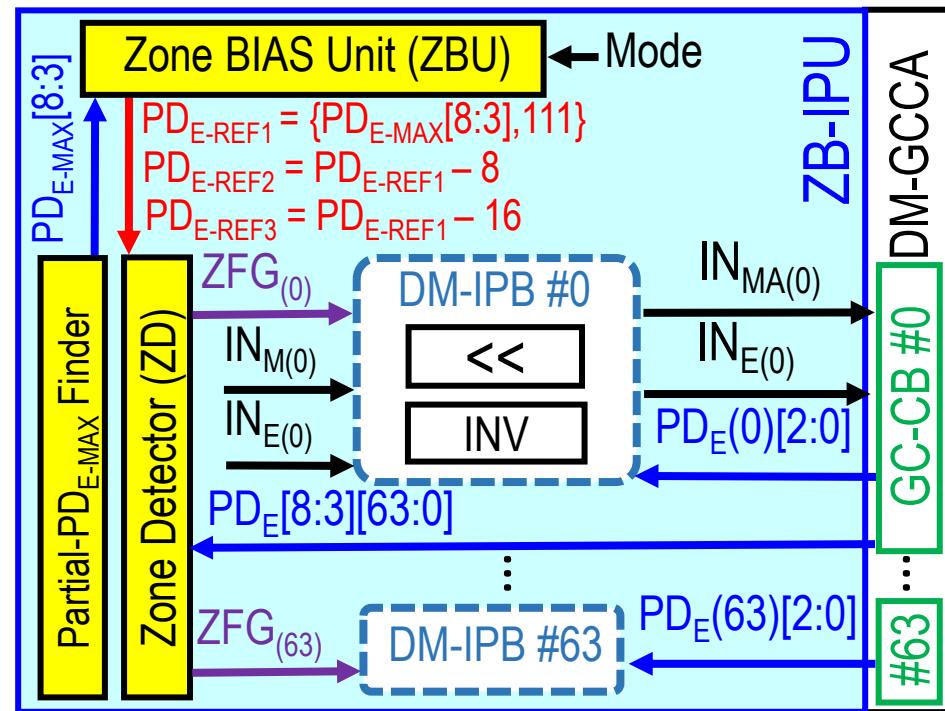
...

$$\underline{PD_E(64)[2:0]} = 3'b011 = PD_{E-REF3} - PD_E(2)$$

$$\vdots \times 64$$

Note: 9b subtractor is simplified by 6b subtractor and 3b inverter

## Zone-detect-based alignment (ZDBA)



# Dual-mode Zone-based Input Processing Unit: Example

BF16 Mode ( $IN_M$  Alignment)  $PD_{E-REF1} = \{PD_{E-MAX}[8:3], 111\}$

$PD_E$ Zone	$PD_E[2:0]$	$N_{SHZ}$	Ph1 $IN_{MA}[7:0]$								Ph2 $IN_{MA}[7:0]$								
$PD_{E-REF2} < PD_E \leq PD_{E-REF1}$ $(ZFG=1)$	111	$000 = 0$	1	$IN_6$	$IN_5$	$IN_4$	$IN_3$	$IN_2$	$IN_1$	$IN_0$	0	0	0	0	0	0	0	0	
	110	$001 = 1$	0	1	$IN_6$	$IN_5$	$IN_4$	$IN_3$	$IN_2$	$IN_1$	$IN_0$	0	0	0	0	0	0	0	
	:	:															⋮		
	000	$111 = 7$	0	0	0	0	0	0	0	1	$IN_6$	$IN_5$	$IN_4$	$IN_3$	$IN_2$	$IN_1$	$IN_0$	0	
$PD_{E-REF3} < PD_E \leq PD_{E-REF2}$ $(ZFG=2)$	111	$000 = 0$	0	0	0	0	0	0	0	0	1	$IN_6$	$IN_5$	$IN_4$	$IN_3$	$IN_2$	$IN_1$	$IN_0$	
	110	$001 = 1$	0	0	0	0	0	0	0	0	0	1	$IN_6$	$IN_5$	$IN_4$	$IN_3$	$IN_2$	$IN_1$	
	:	:															⋮		
	000	$111 = 7$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
$\leq PD_{E-REF3}$ $(ZFG=3)$	XXX	XXX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

## ■ Three zone cases (FP-MAC)

- $PD_E$  with  $ZFG = 1$ :  $IN_M$  alignments are executed in Ph1
- $PD_E$  with  $ZFG = 2$ :  $IN_M$  alignments are executed in Ph2
- $PD_E$  with  $ZFG = 3$ : Triggers input-sparsity-aware circuit to reduce compute energy

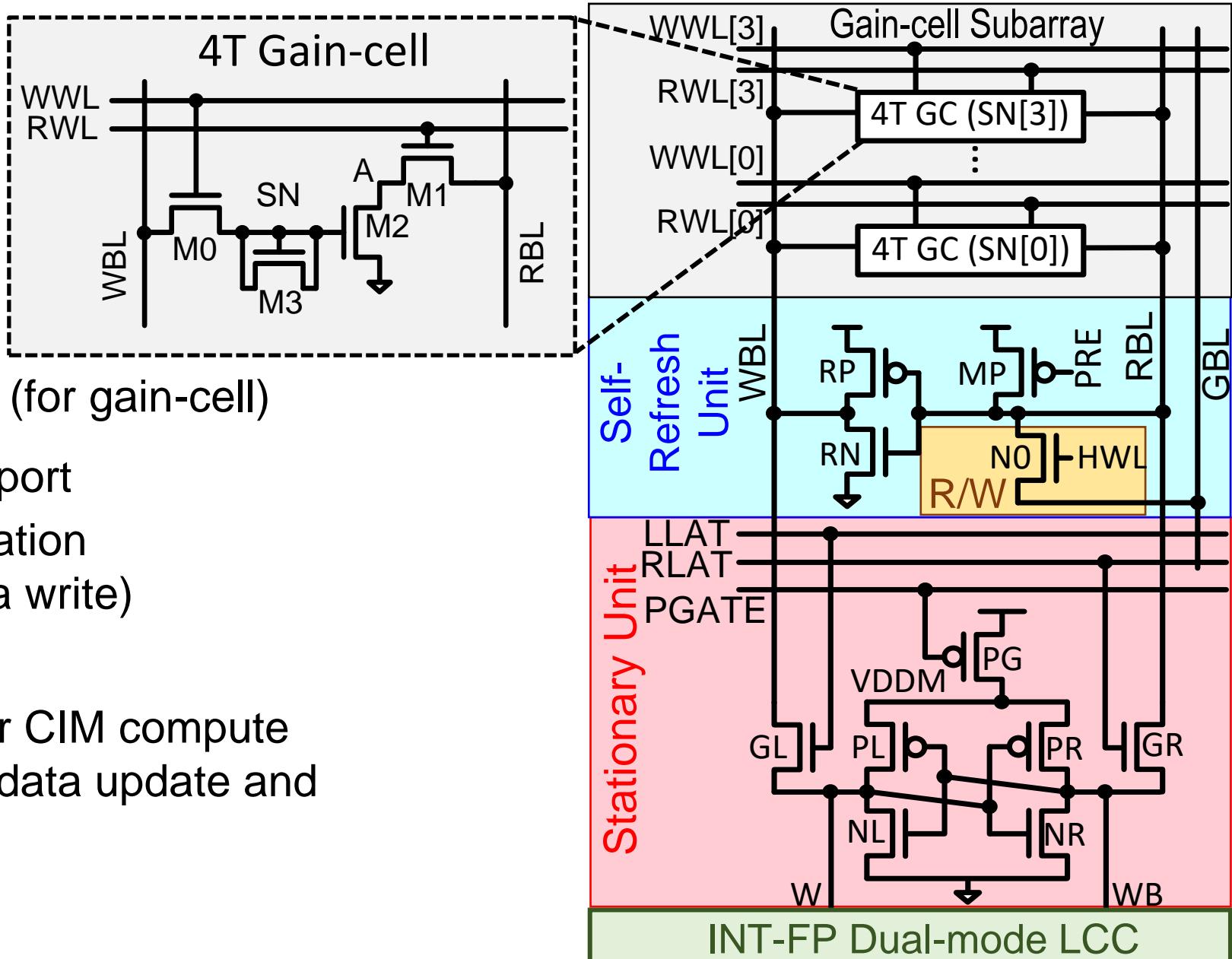
# Outline

- Challenges of Dual-mode Computing-In-Memory
- **Proposed Computation Gain-cell CIM macro**
  - Overview of Integer / Floating-Point Dual-mode CIM Macro
  - Dual-mode Local-computing-cell (DM-LCC)
  - Dual-mode Zone-based Input Processing Unit Scheme (ZB-IPS)
  - Stationary-based Two-port Gain-cell Array Scheme (SB-TP-GCA)
- Performance and Measurement Results
- Conclusion

# Stationary-based Two-port Gain-cell Array : Overview

## ■ Overall Structure

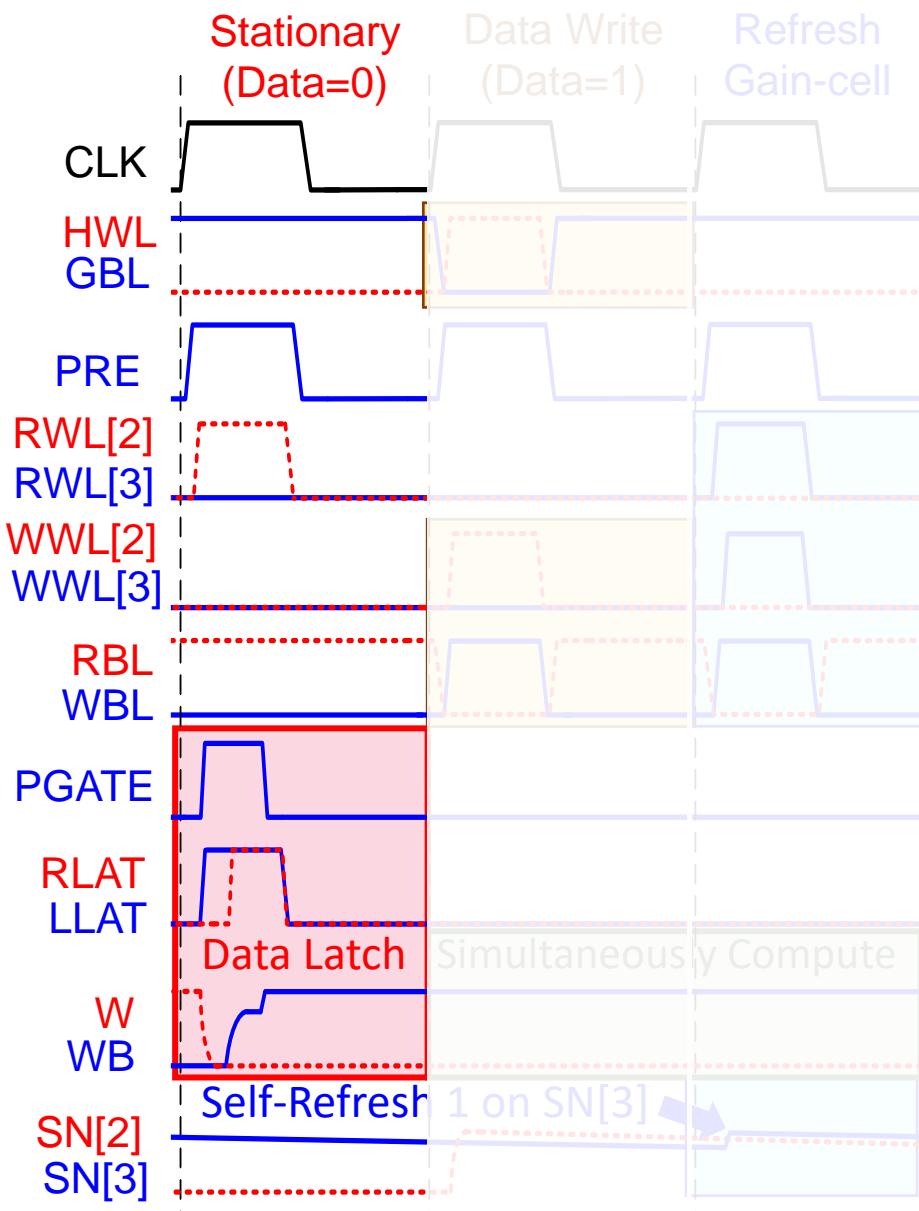
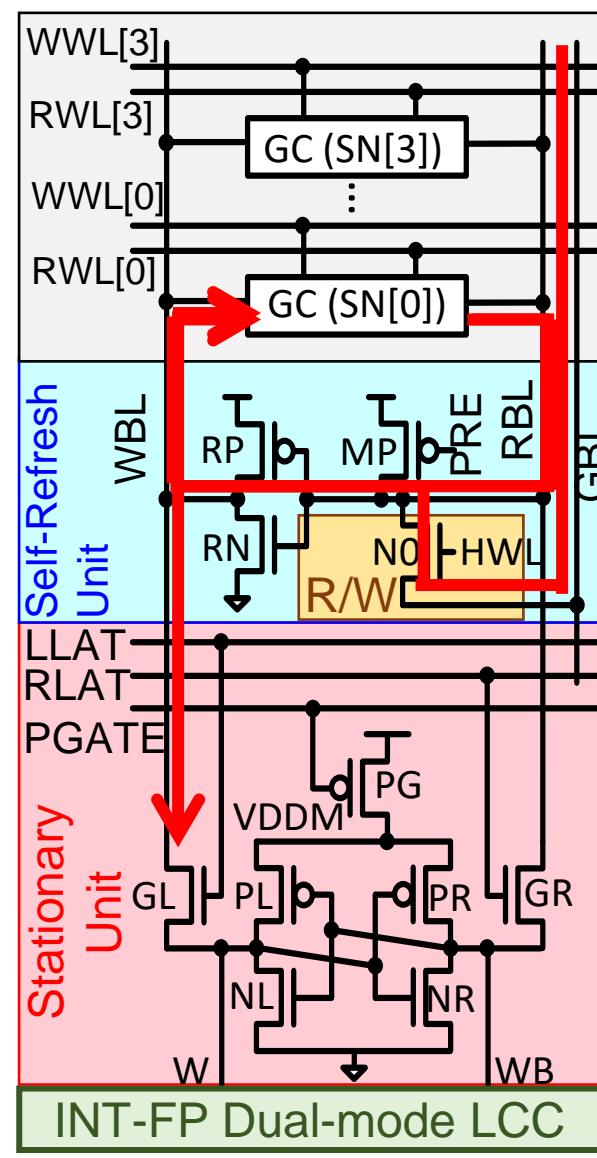
- 4T Gain-cell x4:
  - Weight storage
- 3T Self-refresh unit:
  - Weight local refresh (for gain-cell)
- 1T Global Read / Write port
  - Memory mode operation  
(Data access & Data write)
- 7T Stationary Unit:
  - Weight stationary for CIM compute
  - Support concurrent data update and computation



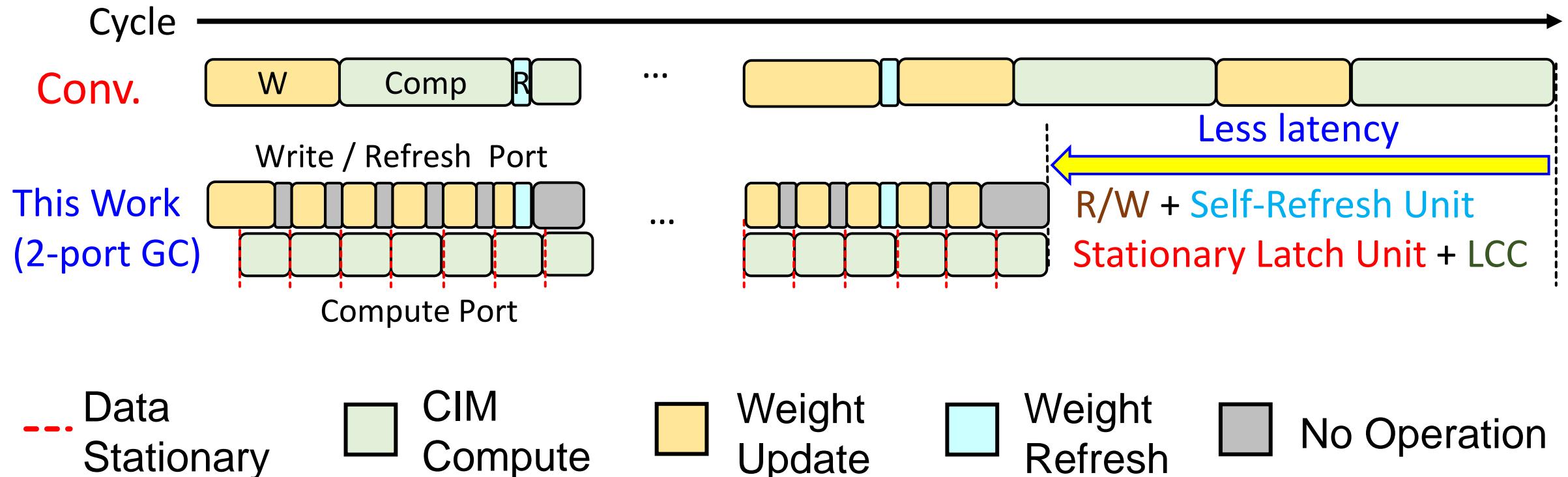
# Stationary-based Two-port Gain-cell Array : Operation

## ■ Three operation modes:

- Stationary-update mode:
  - Load 1b W to stationary unit
- Storage-update mode:
  - Write data from GBL to RBL via N0 transistor ( $HWL=1$ )
- Self-refresh mode:
  - Read data from RBL
  - Drives the WBL to refresh the selected gain-cell



# Stationary-based Two-port Gain-cell Array : Operation

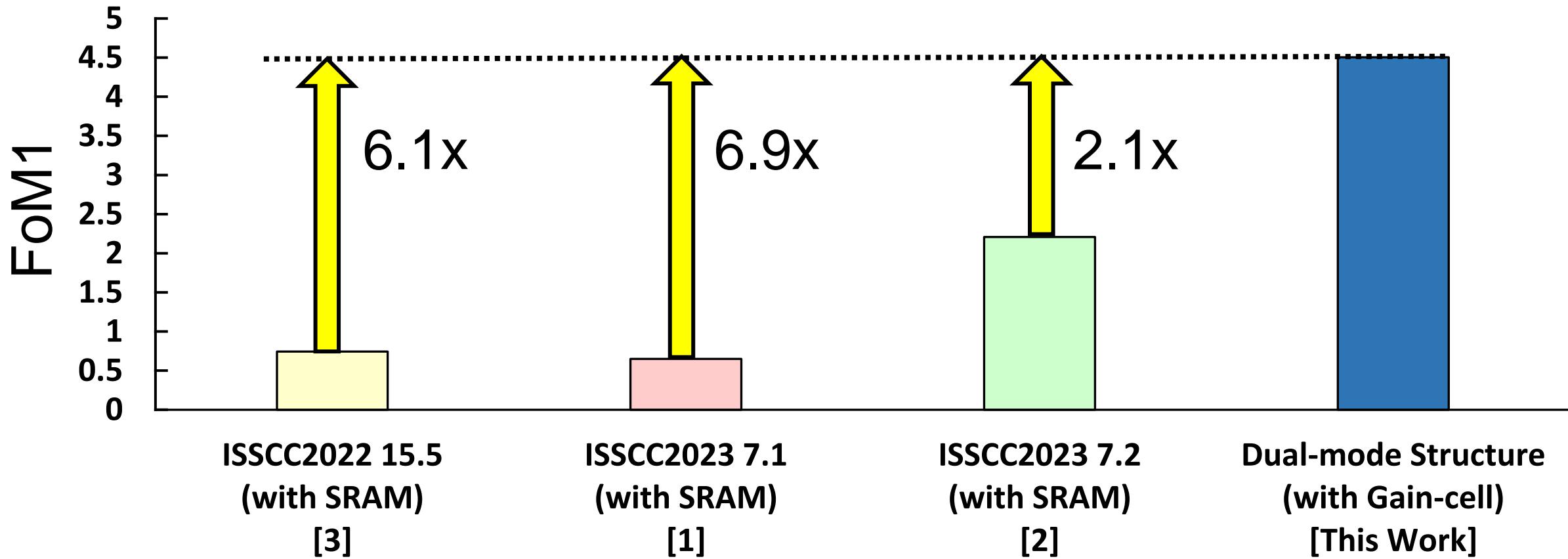


- Supports simultaneous MAC computation and weight updating  
→ shorten the system-level compute latency ( $T_{MAC}$ )

# Outline

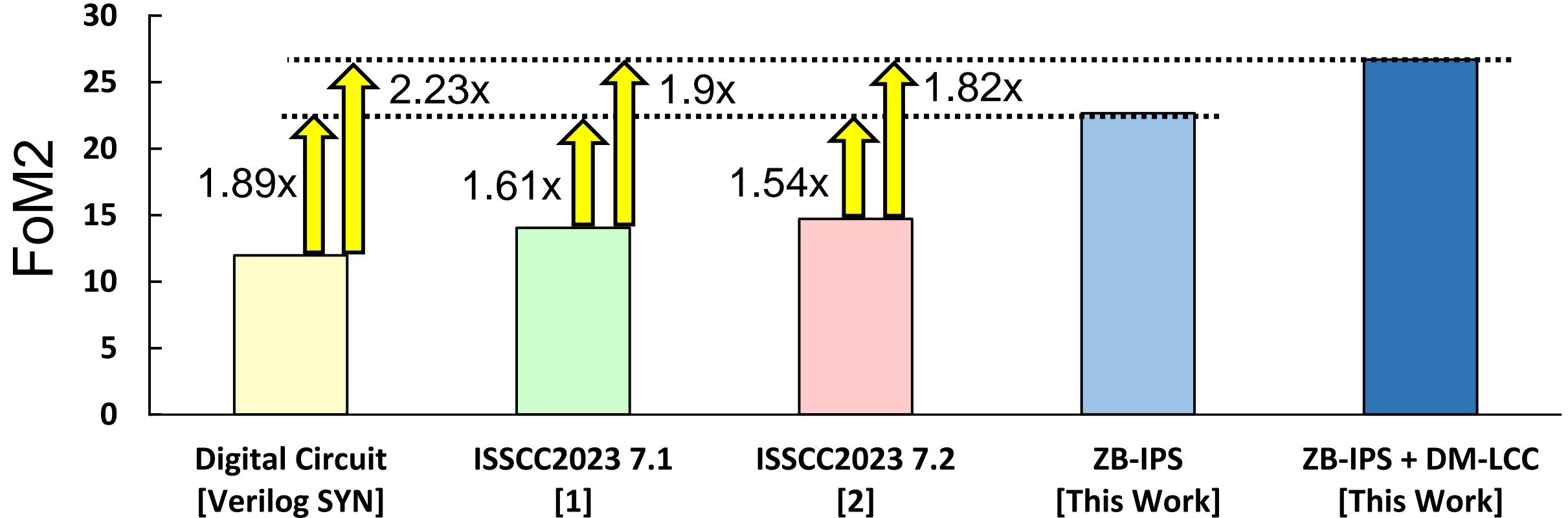
- Challenges of Dual-mode Computing-In-Memory
- Proposed Computation Gain-cell CIM macro
  - Overview of Integer / Floating-Point Dual-mode CIM Macro
  - Dual-mode Local-computing-cell (DM-LCC)
  - Dual-mode Zone-based Input Processing Unit Scheme (ZB-IPS)
  - Stationary-based Two-port Gain-cell Array Scheme (SB-TP-GCA)
- Performance and Measurement Results
- Conclusion

# DM-LCC & ZB-IPS Performance



- Achieves an **FoM1 improvement of  $2.1\times$  to  $6.9\times$** 
  - $\text{FoM1} = \text{Normalized area efficiency (BF16)} \times \text{area efficiency (INT8)}$

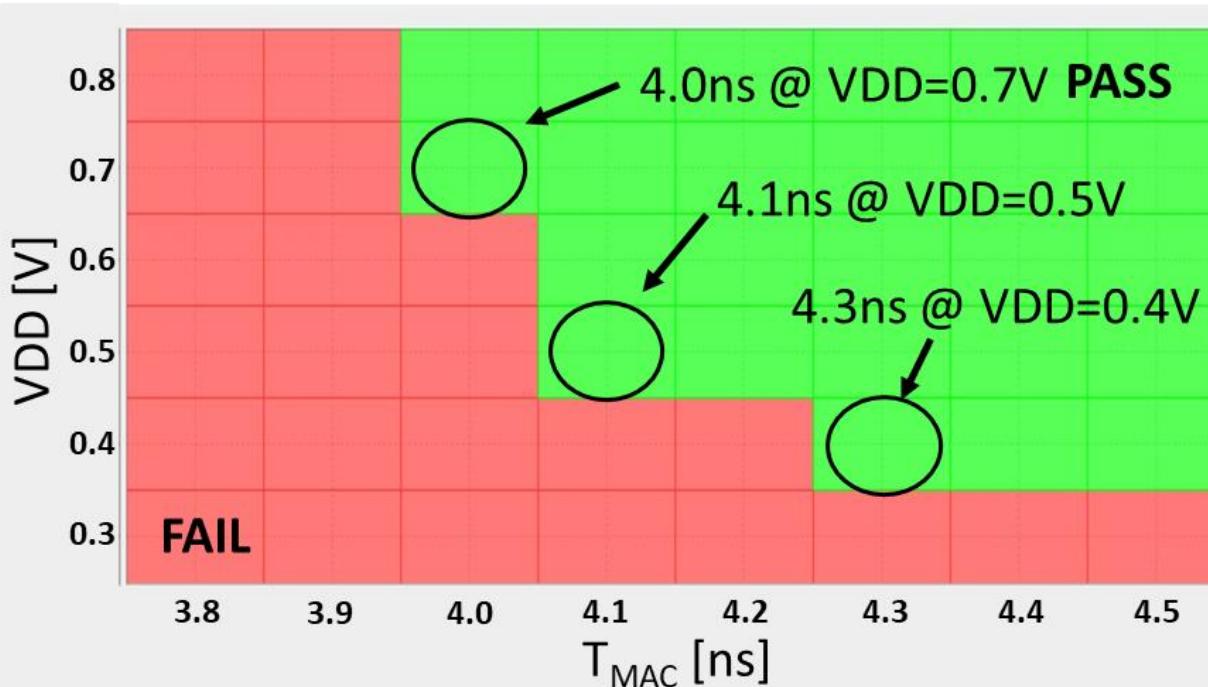
# Dual-mode CIM Structure Performance



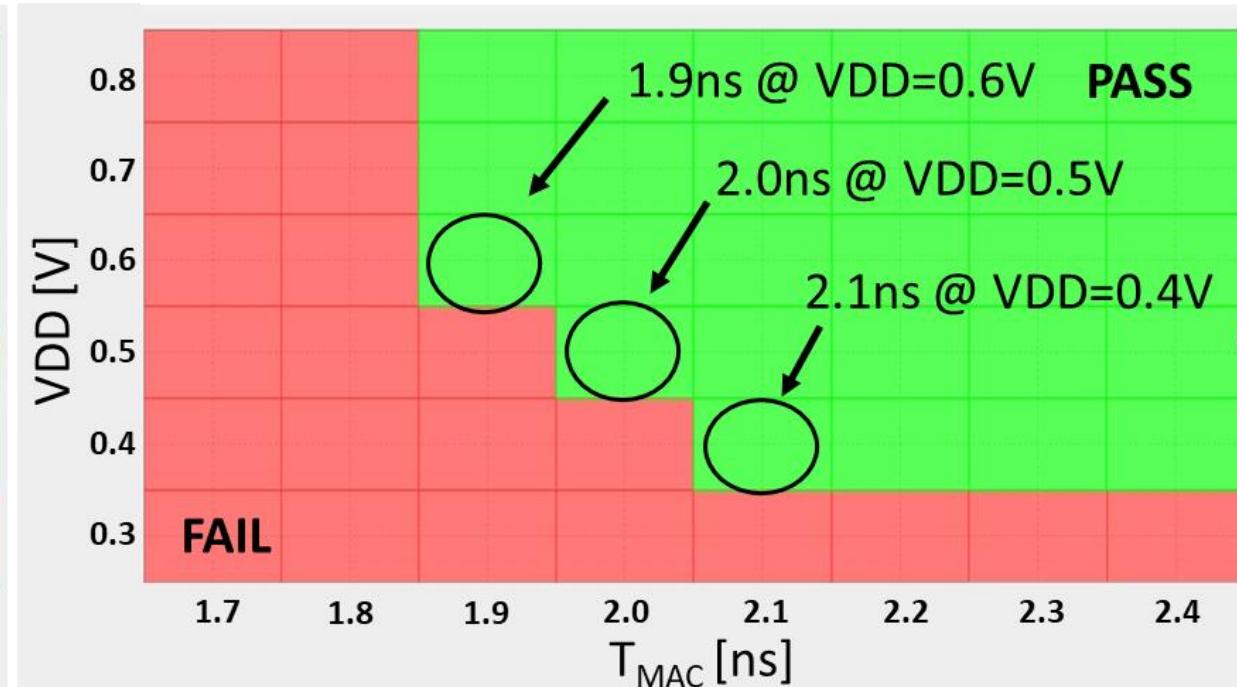
- Achieves an FoM2 improvement of 1.5× to 2.2×
  - $\text{FoM2} = 1 / (\text{Normalized Macro Energy} \times \text{Area} \times \text{Accuracy Loss})$

# Shmoo Plot

Shmoo: BF16 IN – BF16 W – FP32 OUT



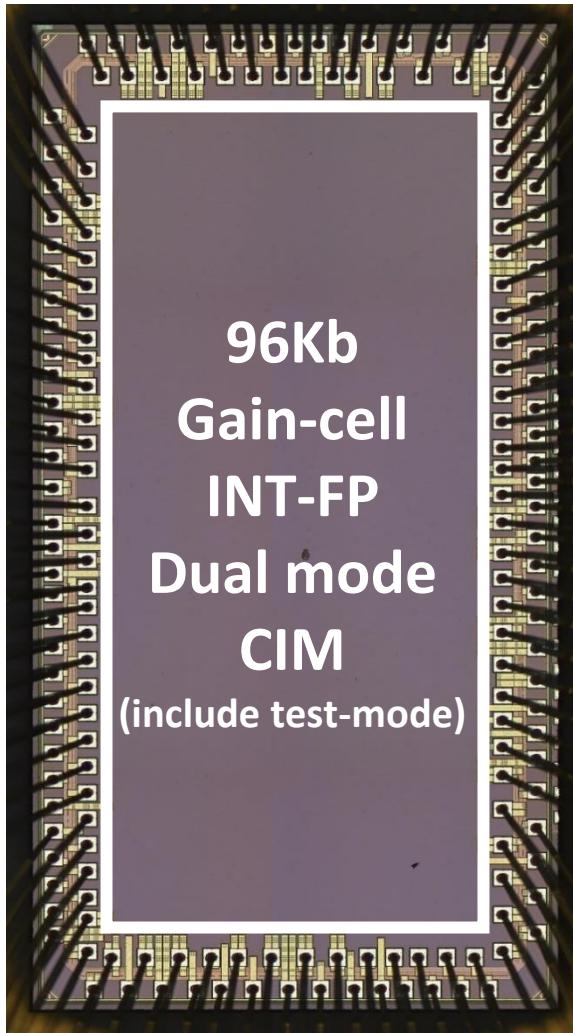
Shmoo: INT8 IN – INT8 W – INT23 OUT



## ■ Measurement results

- BF16-IN, BF16-W, FP32-OUT Access time ( $t_{AC}$ ) = 4.0ns @  $V_{DD}=0.7V$
- INT8-IN, INT8-W, INT23-OUT Access time ( $t_{AC}$ ) = 1.9ns @  $V_{DD}=0.6V$

# Chip Summary



CHIP SUMMARY		
Technology	TSMC 16nm FinFET	
Gaincell Capacity	96Kb	
Input precision (bit)	INT8	BF16
Weight precision (bit)	INT8	BF16
Number of input channels ( $N_{ACCU}$ )	128	64
Number of output channels	24	24
Output precision (bit)	23	FP32
Supply voltage (V)	0.4-0.8	0.4-0.8
Access time (ns)	1.9 (0.8V)	4.0 (0.8V)
Energy efficiency (TOPS/W) (TFLOPS/W) (Average performance)	73.3-98.5	33.2-45.4
Energy efficiency (TOPS/W) (TFLOPS/W) ( <sup>1</sup> Peak performance)	125.6-163.3	72.3-91.2
Area efficiency (TOPS/mm <sup>2</sup> ) (TFLOPS/mm <sup>2</sup> )	11.07 (0.8V)	2.63 (0.8V)
<sup>2</sup> Inference Accuracy Loss (CIFAR-100)	-0.19%	-0.01%
<sup>3</sup> Inference Accuracy Loss (ImageNet)	-0.58%	-0.02%

<sup>1</sup>Performance in 90% input sparsity, 10% input toggle rate (ResNet-20)

<sup>2</sup>Using ResNet-20 model and the software baseline (FP32) was 70.22%

<sup>3</sup>Using ResNet-18 model and the software baseline (FP32) was 68.31%

# Outline

- Challenges of Dual-mode Computing-In-Memory
- Proposed Computation Gain-cell CIM macro
  - Overview of Integer / Floating-Point Dual-mode CIM Macro
  - Dual-mode Local-computing-cell (DM-LCC)
  - Dual-mode Zone-based Input Processing Unit Scheme (ZB-IPS)
  - Stationary-based Two-port Gain-cell Array Scheme (SB-TP-GCA)
- Performance and Measurement Results
- Conclusion

# Conclusion

## ■ A 96Kb Dual-mode Gain-cell-CIM Macro is demonstrated

### ■ Three key features

- Dual-mode Local-computing-cell
  - Reuses the exponent addition as an adder tree stage for INT-MAC
- Dual-mode Zone-based Input Processing Unit
  - Simplifies FP exponent processing and reuses FP hardware as sparsity detection in INT
- Stationary-based Two-port Gain-cell Array
  - Supports concurrent data update and computation

### ■ Key performances

- BF16 IN- BF16 W- FP32 OUT
  - Energy efficiency: 33.2-91.2 TFLOPS/W; Area efficiency: 2.63 TFLOPS/mm<sup>2</sup>
- INT8b IN- INT8b W- INT23b OUT
  - Energy efficiency: 73.3-163.3 TOPS/W; Area efficiency: 11.07 TOPS/mm<sup>2</sup>

# Acknowledgements

- The authors thank the guidance from Philip Wong, Kerem Akarvardar, and TSMC colleagues, and the financial support from NSTC and TSMC-NTHU Major League.

**Thank you for your kind attention**



Please Scan to Rate  
This Paper



# A 22-nm 64-kb Lightning-like Hybrid Computing-in-Memory Macro with a Compressed Adder Tree and Analog-Storage Quantizers for Transformer and CNNs

An Guo<sup>1</sup>, Xi Chen<sup>1</sup>, Fangyuan Dong<sup>1</sup>, Jinwu Chen<sup>1</sup>, Zhihang Yuan<sup>2,3</sup>, Xing Hu<sup>3</sup>, Yuanpeng Zhang<sup>2</sup>, Jingmin Zhang<sup>1</sup>, Yuchen Tang<sup>1</sup>, Zhican Zhang<sup>1</sup>, Gang Chen<sup>3</sup>, Dawei Yang<sup>3</sup>, Zhaoyang Zhang<sup>1</sup>, Lizheng Ren<sup>1</sup>, Tianzhu Xiong<sup>1</sup>, Bo Wang<sup>1</sup>, Bo Liu<sup>1</sup>, Weiwei Shan<sup>1</sup>, Xinning Liu<sup>1</sup>, Hao Cai<sup>1</sup>, Guangyu Sun<sup>2</sup>, Jun Yang<sup>1</sup>, Xin Si<sup>1</sup>

<sup>1</sup>Southeast University, Nanjing, China, <sup>2</sup>Peking University, Beijing, China, <sup>3</sup>HOUМО, Beijing, China



東南大學  
SOUTHEAST UNIVERSITY



北京大学



# Outline

---

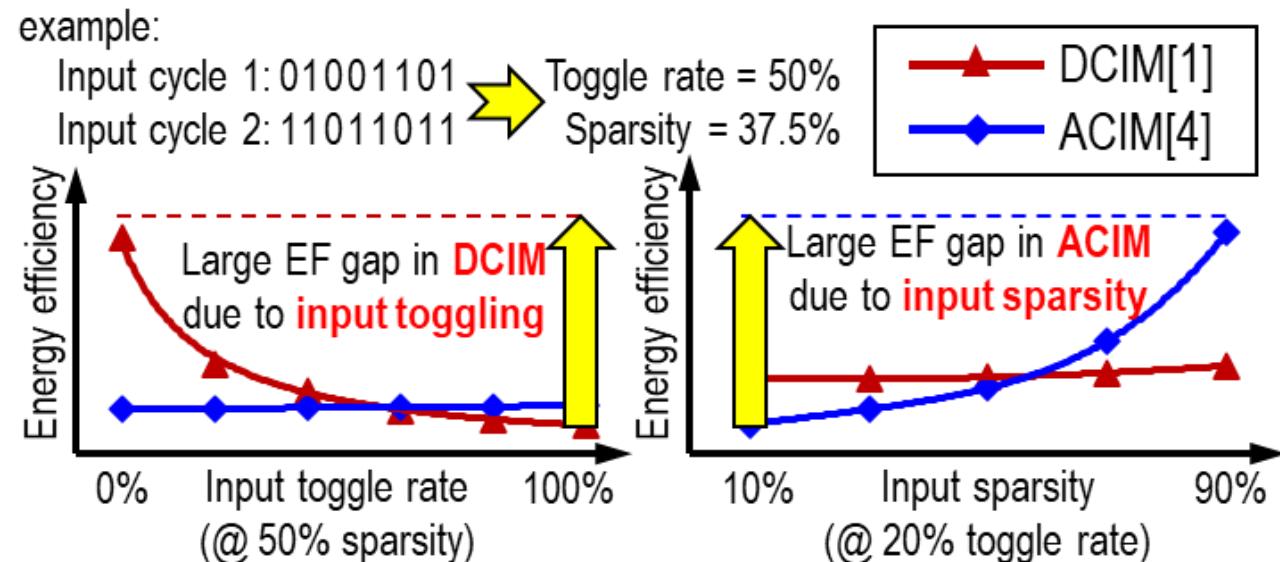
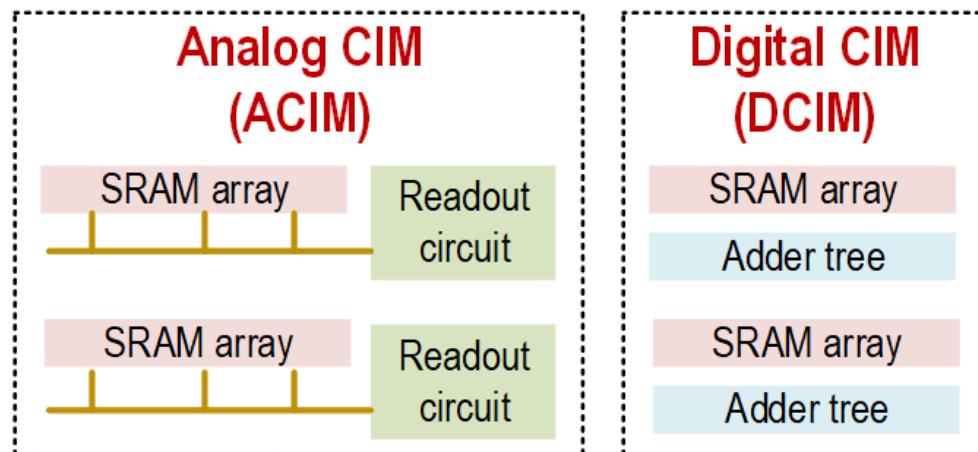
- Motivation, Background, and Challenges
- Proposed Hybrid Computing-in-Memory (CIM) Macro
  - Lightning-like macro level structure
  - Compressed Adder Tree
  - Analog-Storage Quantizer
- Measurement Results
- Conclusion

# Outline

---

- Motivation, Background, and Challenges
- Proposed Hybrid Computing-in-Memory (CIM) Macro
  - Lightning-like macro level structure
  - Compressed Adder Tree
  - Analog-Storage Quantizer
- Measurement Results
- Conclusion

# Motivation



- SRAM CIM macro
  - Digital CIM: High precision computation  
Limited EF(energy efficiency)
  - Analog CIM: Potentially-high EF  
Suffer from PVT variation
- Feature distribution influence on CIM
  - Digital CIM suffers from **input toggling**
    - ◊ Due to CMOS temporary store feature
  - Analog CIM suffers from **input sparsity**
    - ◊ Due to its resetting feature per cycle

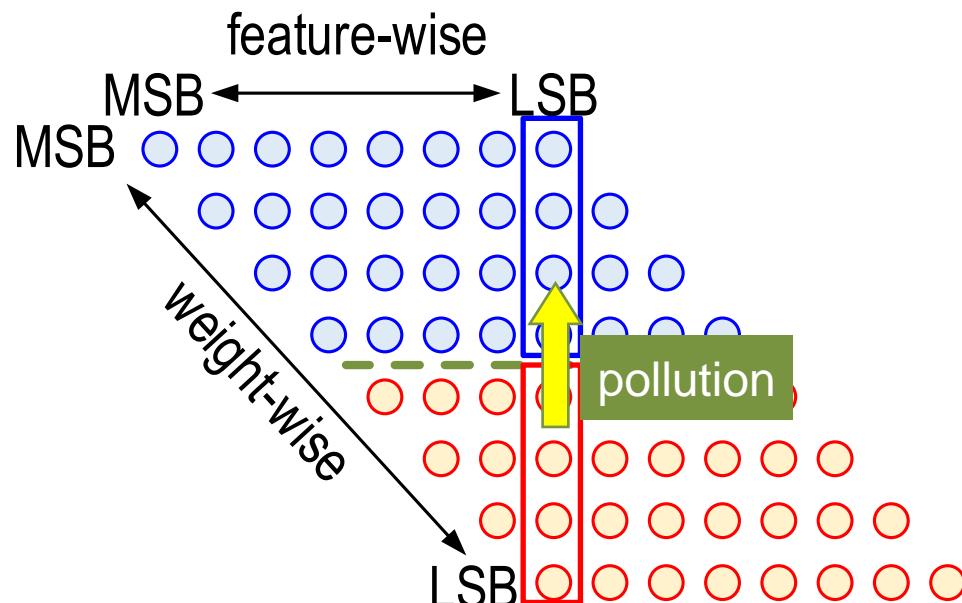
# Challenges of Hybrid CIM

## □ Challenge 1:

- Tradeoff between accuracy and area/power overhead of hybrid CIM

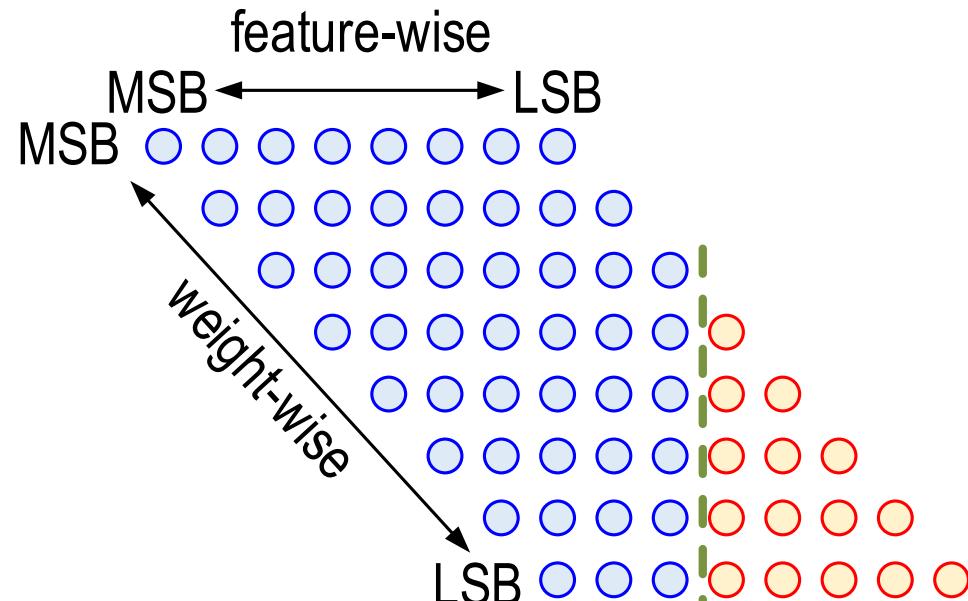
○ W/o calculation error    ○ With calculation error

### Weight-wise-cut structure[1]



(1) Accuracy loss due to error pollution

### Vertical-cut structure[2]

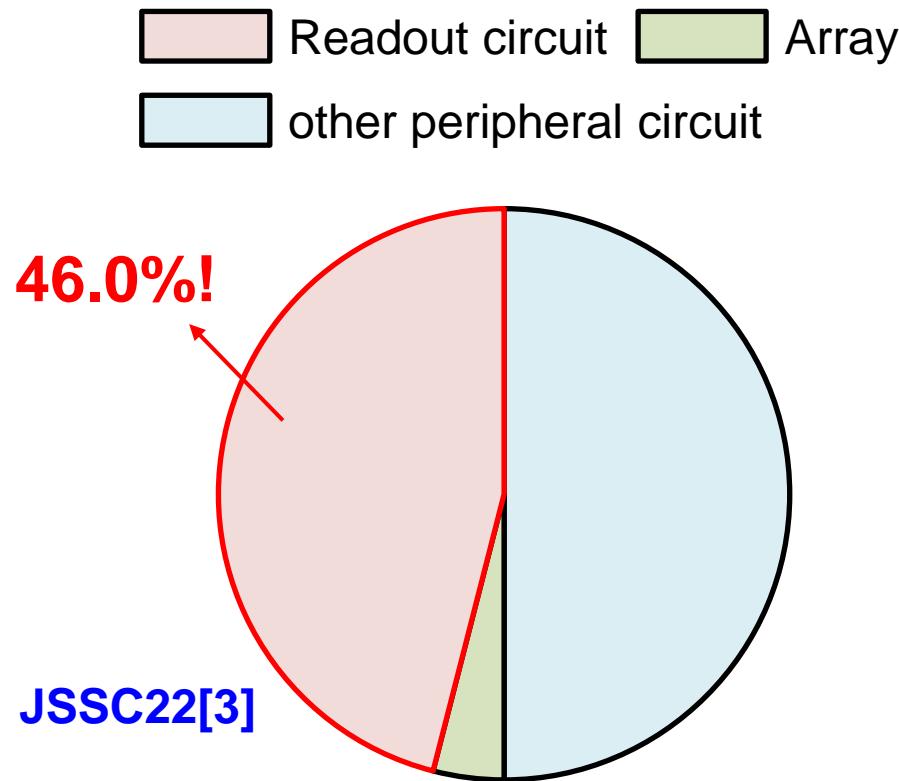


(2) Large RC due to complex LCC

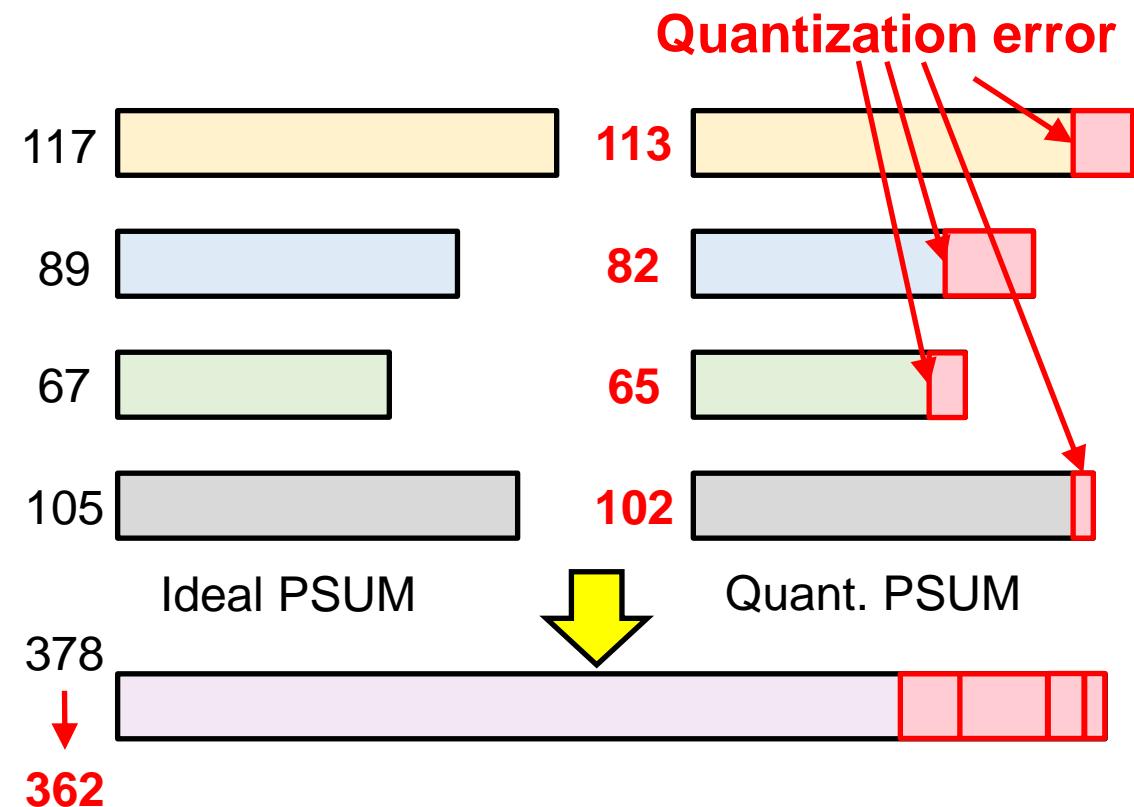
# Challenges of Hybrid CIM

## □ Challenge 2:

- Significant energy cost and error accumulation of the readout circuit in ACIM



(1) **Significant energy cost of the readout circuit in ACIM**

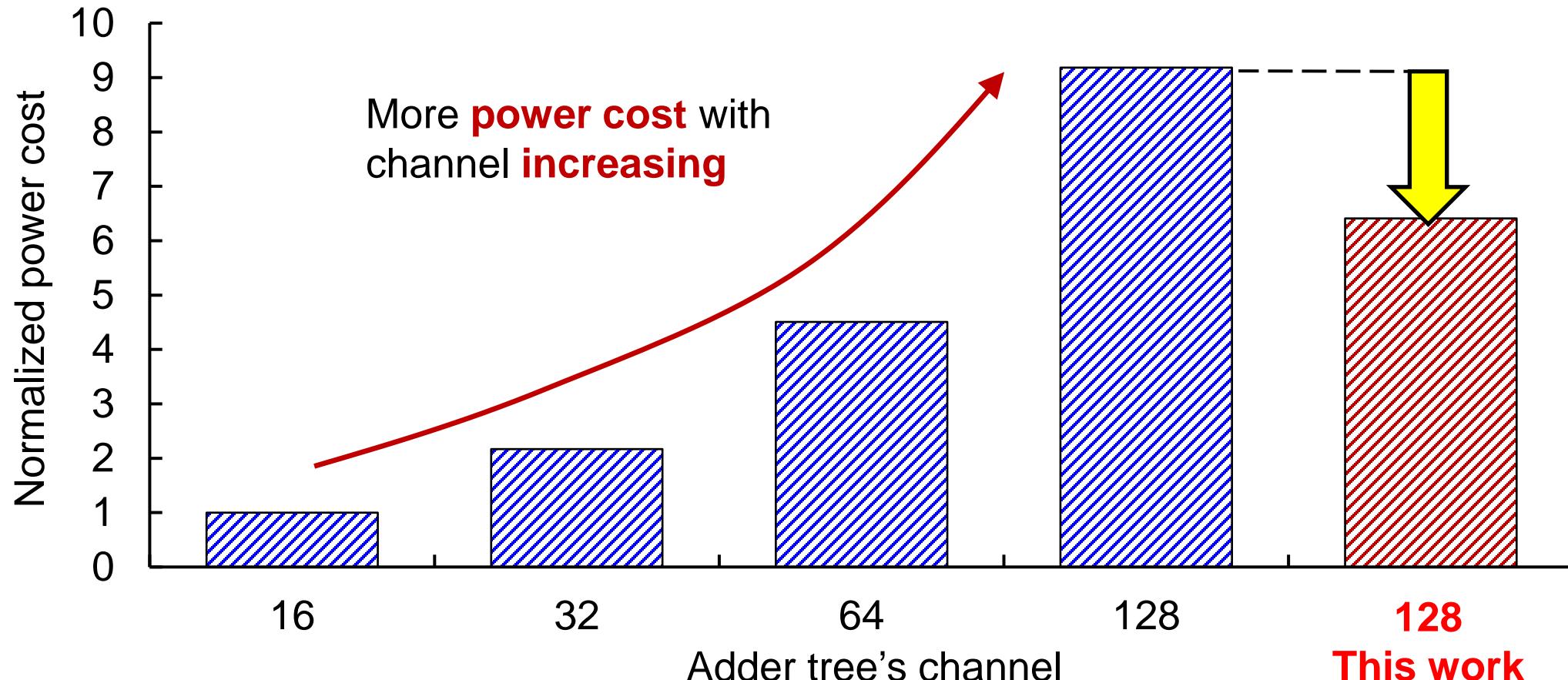


(2) **Error accumulation of the readout circuit in ACIM**

# Challenges of Hybrid CIM

## □ Challenge 3:

- Limited energy efficiency of DCIM due to Large-scale adder-tree

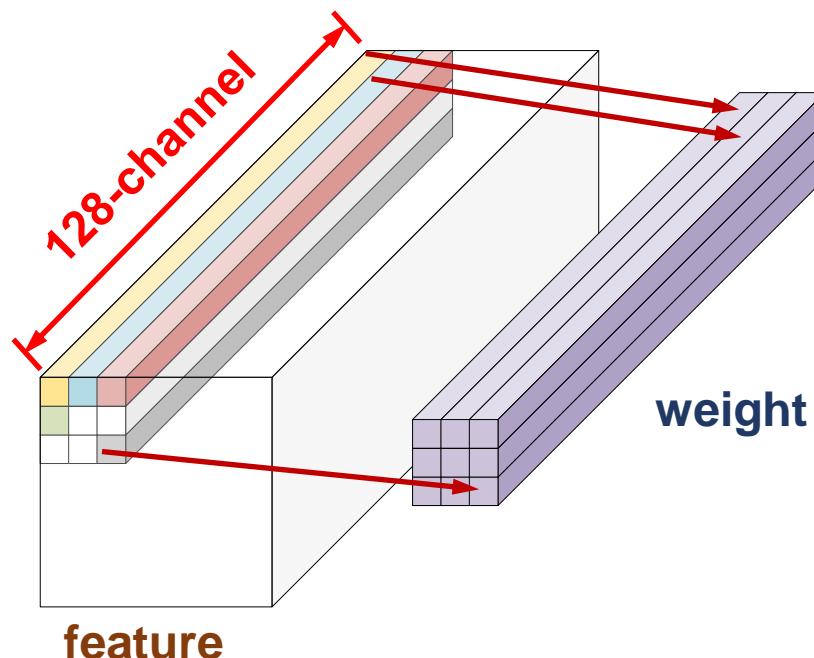


# Challenges of Hybrid CIM

## □ Challenge 4:

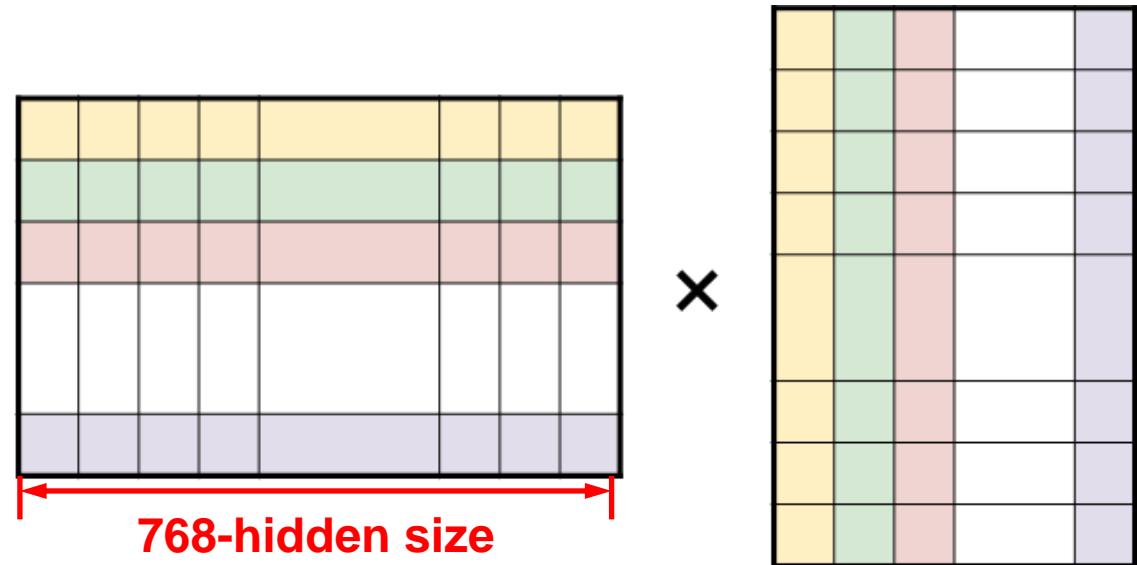
- MAC size could exceed CIM size

ResNet50: conv3\_2-layer



MAC size ( $128 \times 3 \times 3 = 1152$ )  
**exceeds** CIM size (128)!

Vision transformer: self attention



Hidden size (768)  
**exceeds** CIM size (128)!

# Outline

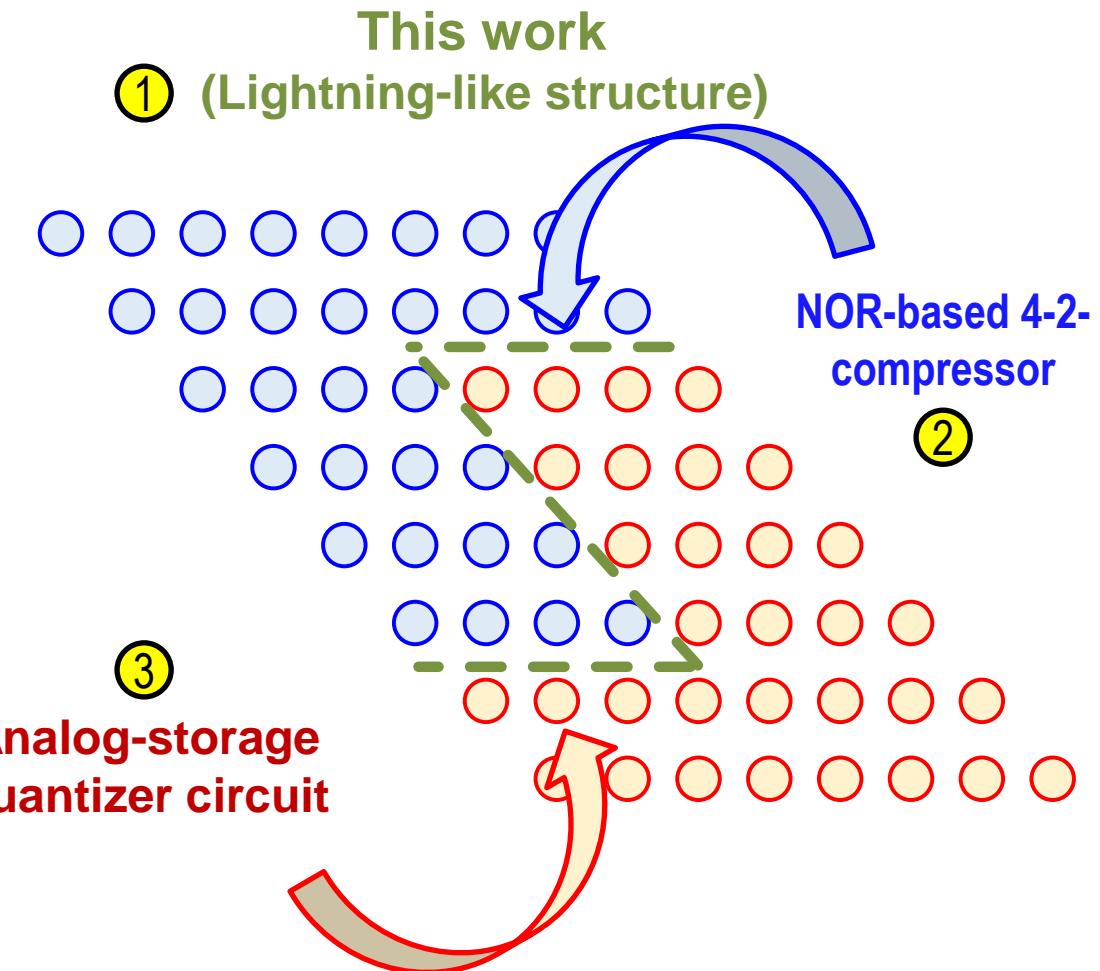
---

- Motivation, Background, and Challenges
- Proposed Hybrid Computing-in-Memory (CIM) Macro
  - Lightning-like macro level structure
    - Compressed Adder Tree
    - Analog-Storage Quantizer
- Measurement Results
- Conclusion

# Overview: Hybrid Lightning CIM Macro

## Key feature

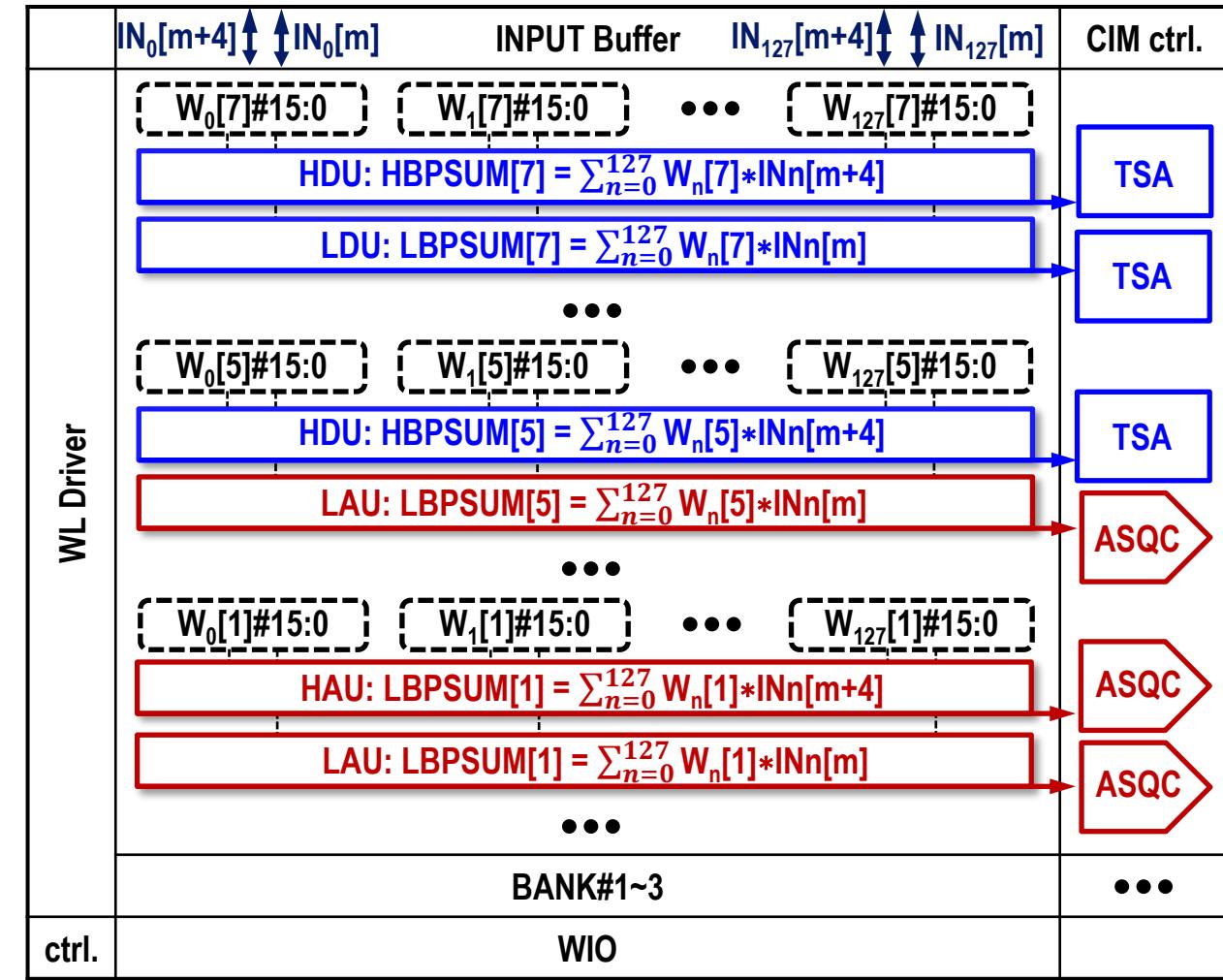
- (1) Lightning-like structure to maintain a better tradeoff between accuracy and area/power overhead
- (2) NOR-based 4-2-compressor with double-regularization to improve energy efficiency
- (3) Analog-storage quantizer circuit to extend accumulation length



# Overview: Hybrid Lightning CIM Macro

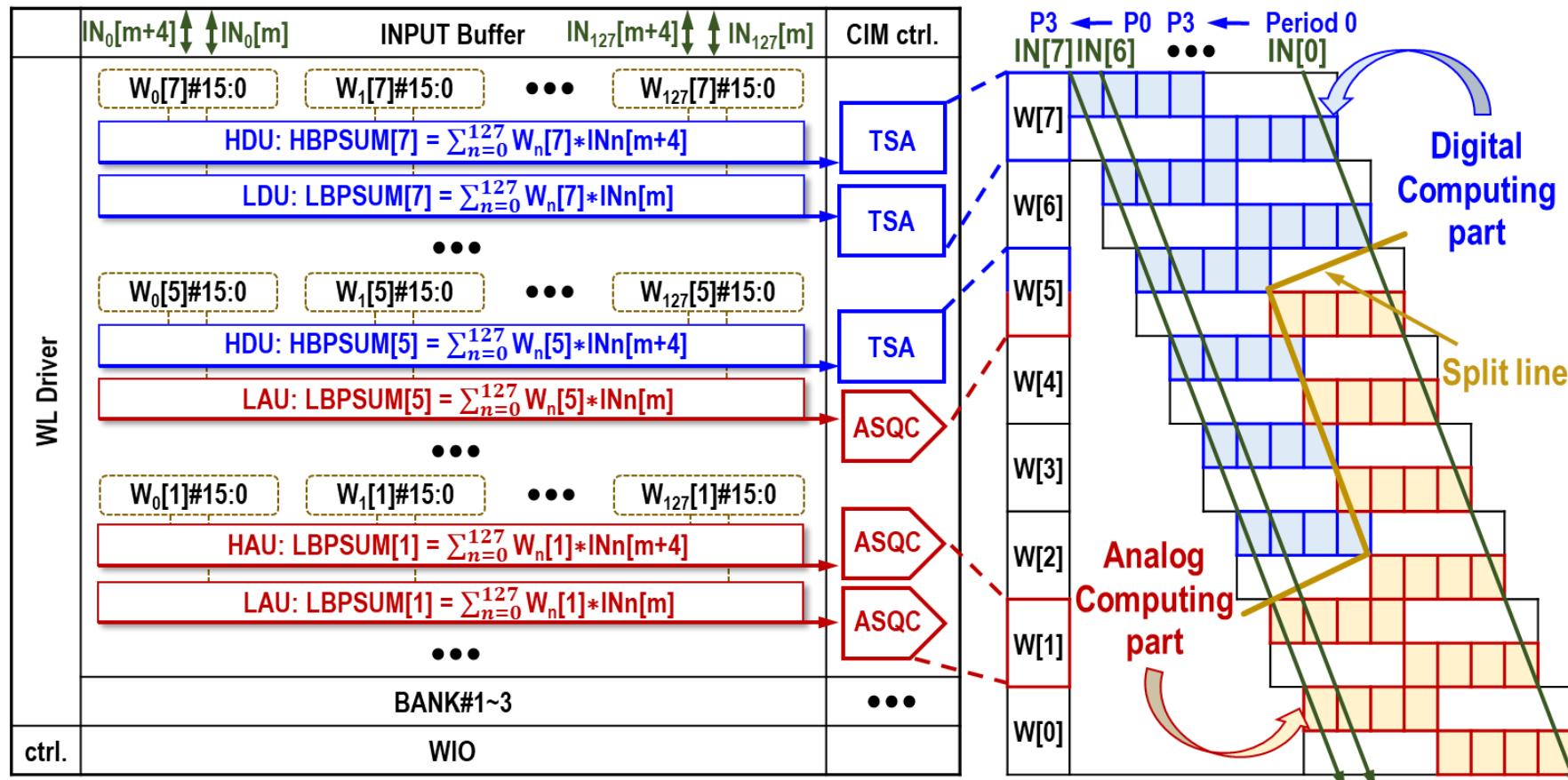
## Macro level structure

- (1) Word-wise MAC units (WMAU)
  - ◆ High and low digital units (HDU & LDU)
  - ◆ High and low analog units (HAU & LAU)
- (2) Post-processor
  - ◆ Time-scale accumulators (TSA)
  - ◆ Analog-storage quantizer circuit (ASQC)
- (3) WL control driver
- (4) Mod4 input buffers
- (5) Other peripheral Circuits



# Overview: Hybrid Lightning CIM Macro

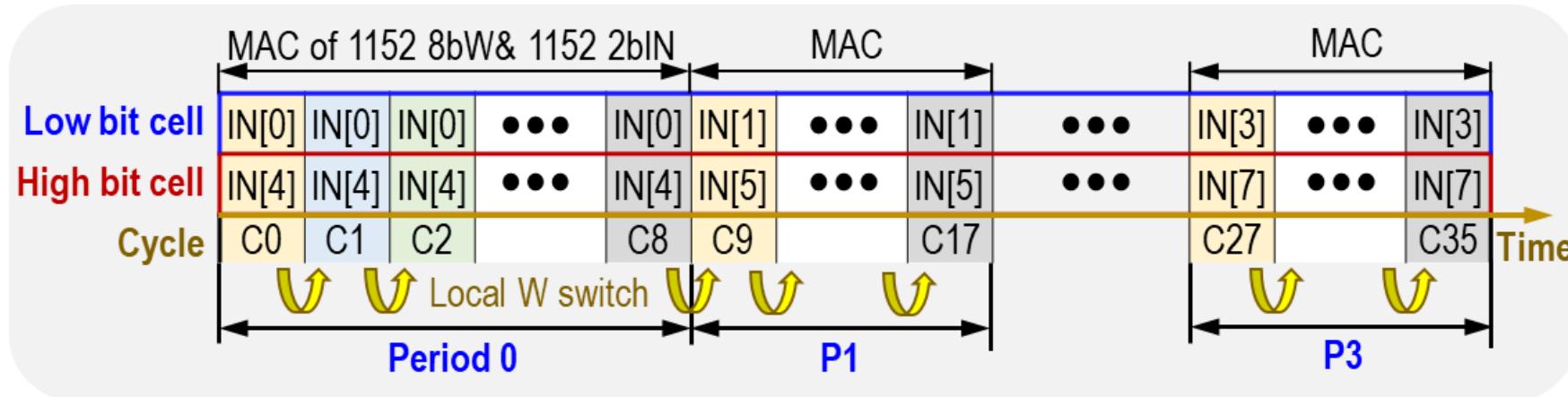
- All calculation divided into two parts based on input data: high bit unit and low bit unit
  - For high bit: W[7:2] in **digit** and W[1:0] in **analog**
  - For low bit: W[7:6] in **digit** and W[5:0] in **analog**
- Time-scale Accumulator (TSA) and analog-storage quantizer circuit (ASQC)
  - **TSA** for **digital** multi-cycle accumulation
  - **ASQC** for **analog** multi-cycle accumulation



34.3: A 22-nm 64-kb Lightning-like Hybrid Computing-in-Memory Macro with a Compressed Adder Tree and Analog-Storage Quantizers for Transformer and CNNs

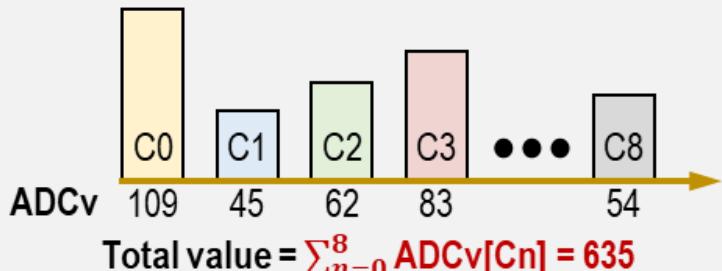
# Multi-cycle accumulation(MCA)

## □ Proposed CIM MCA data flow



$$\text{This work: } \text{MACv} = \sum_{m=0}^3 \sum_{t=0}^7 \sum_{r=0}^8 \sum_{n=0}^{127} W_n^r [t] * (\text{IN}_n^r[m] + \text{IN}_n^r[m+4])$$

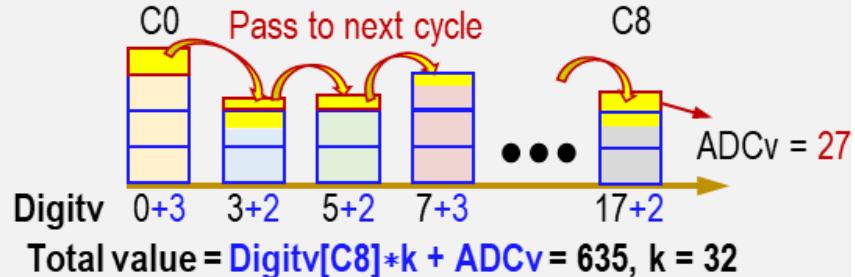
### Previous analog accumulation



One ADC **per cycle**, one PSUM **per cycle**

→ **36 ADC times, 36 PSUM store**

### Proposed analog accumulation



One ADC **per period**, one PSUM **per period**

→ **4 ADC times, 4 PSUM store**

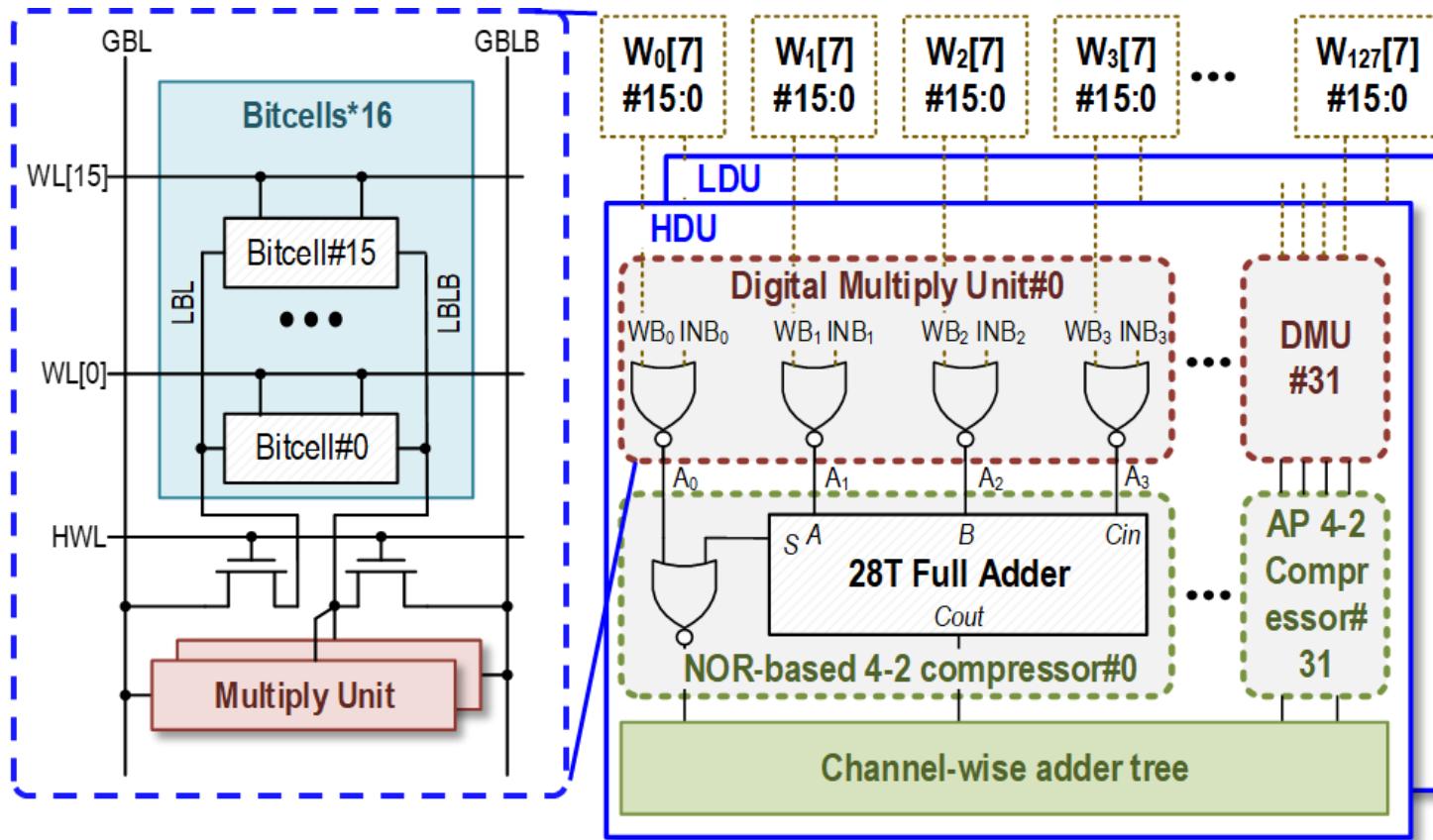
# Outline

---

- Motivation, Background, and Challenges
- Proposed Hybrid Computing-in-Memory (CIM) Macro
  - Lightning-like macro level structure
  - Compressed Adder Tree
  - Analog-Storage Quantizer
- Measurement Results
- Conclusion

# Compressed Adder Tree

## □ Proposed Digital circuit design



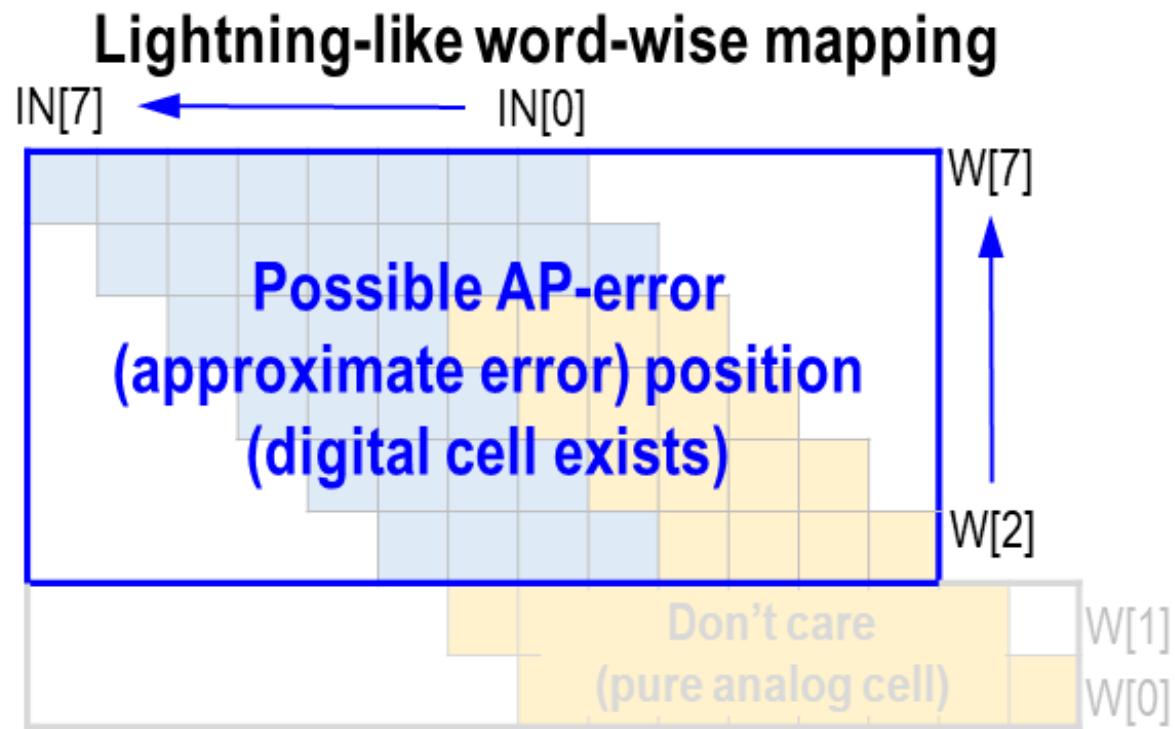
Truth table of NOR-based 4-2 compressor						
Input				Output		No error?
A <sub>0</sub>	A <sub>1</sub>	A <sub>2</sub>	A <sub>3</sub>	S[1]	S[0]	
0	0	0	0	0	0	😊
...	...	...	...	...	...	...
0	1	1	1	1	1	😊
1	0	0	0	0	1	😊
...	...	...	...	...	...	...
1	1	1	1	1	1	😢

No error happens when  $W_0 = 0$ (bit-wise)

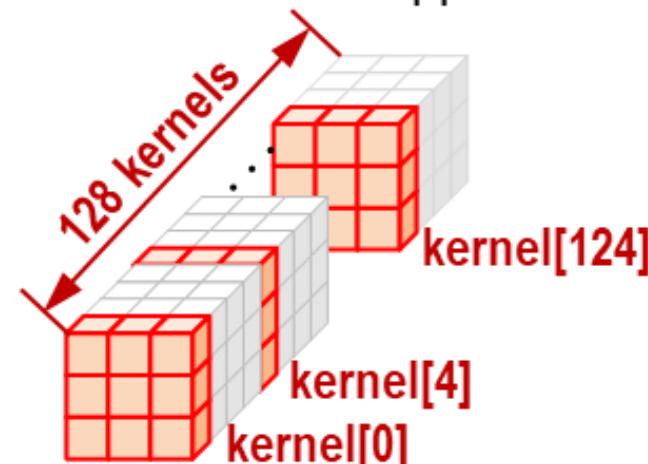
Possible error (21.875% with total random IN&W) happens when  $W_0 = 1$

# Possible error condition

- Approximate(AP) error condition:
  - No AP error happens in analog part → No AP error in  $W[1:0]$
  - No AP error happens in  $A_{1-3}$  → Aimed at  $A_0$



No AP-error happens when  $0 \leq W_{4n}[7:0] \leq 3$ .  
Small AP-error happens when  $4 \leq W_{4n}[7:0] \leq 15$ .



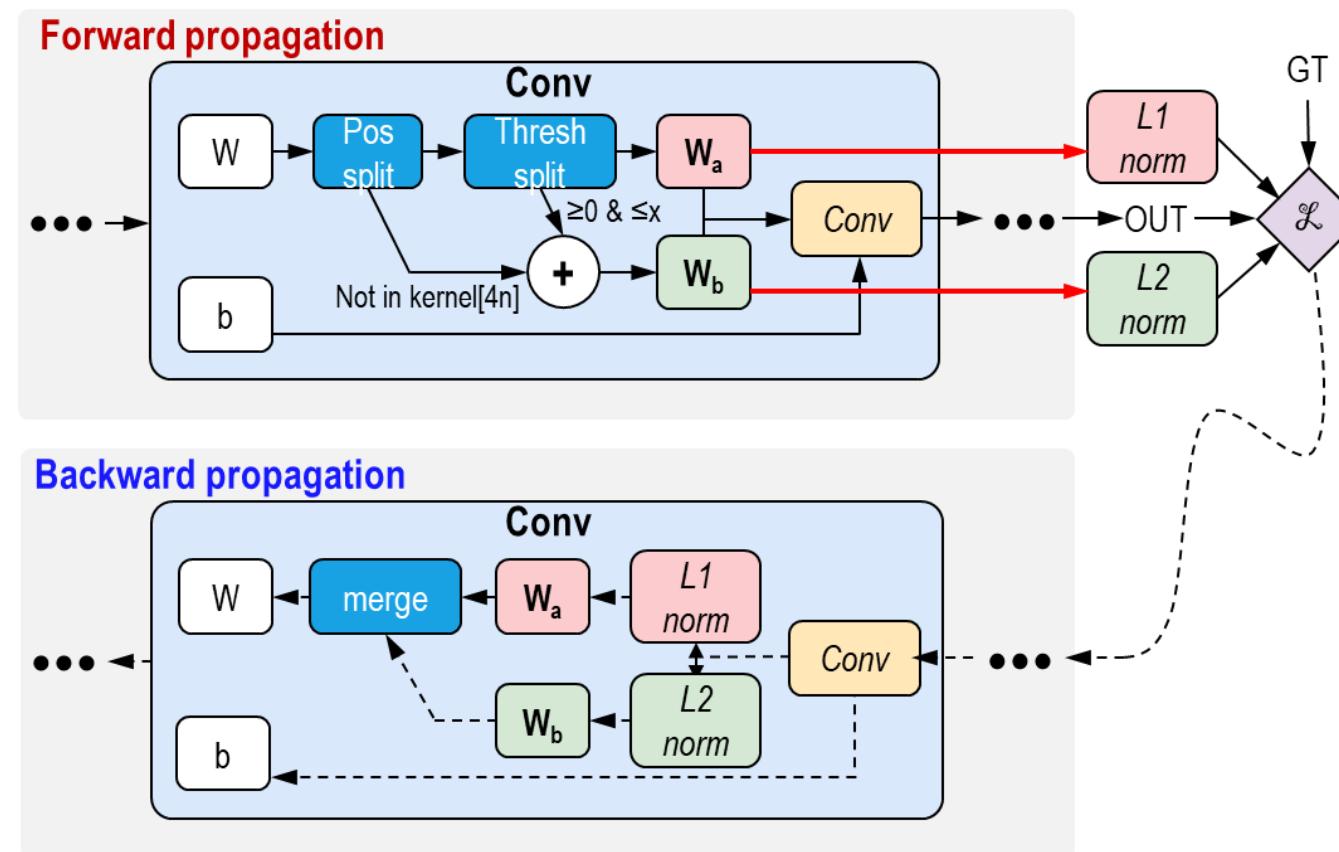
W in **kernel[4n]** will be limited in **[0,3]**

# Double-regularization training method

## □ Proposed training method

$W_a$ : weight in **kernel[4n]** and **>3 or <0**. → Sparsity regularization ( $L_1$ )

$W_b$ : other weight. → Normal regularization ( $L_2$ )



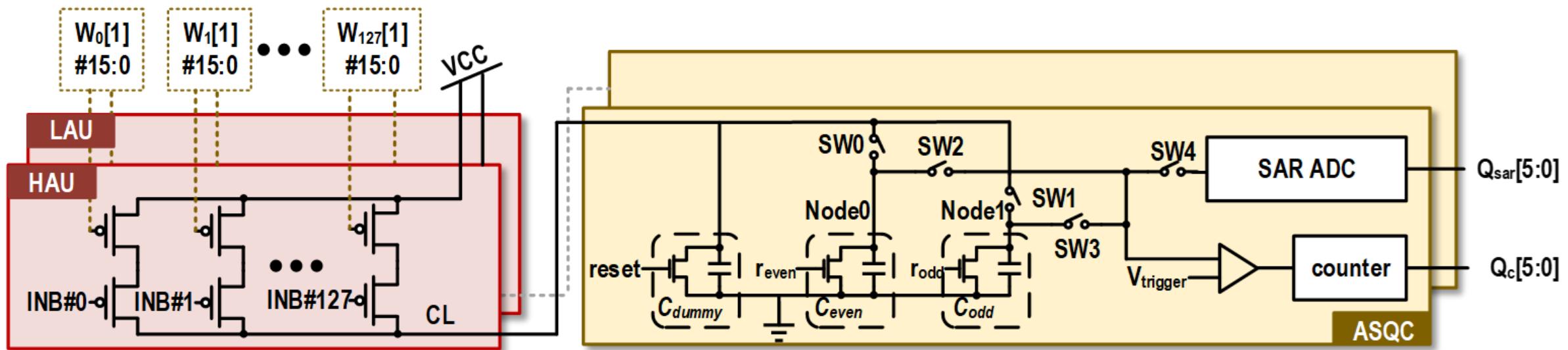
# Outline

---

- Motivation, Background, and Challenges
- Proposed Hybrid Computing-in-Memory (CIM) Macro
  - Lightning-like macro level structure
  - Compressed Adder Tree
  - Analog-Storage Quantizer
- Measurement Results
- Conclusion

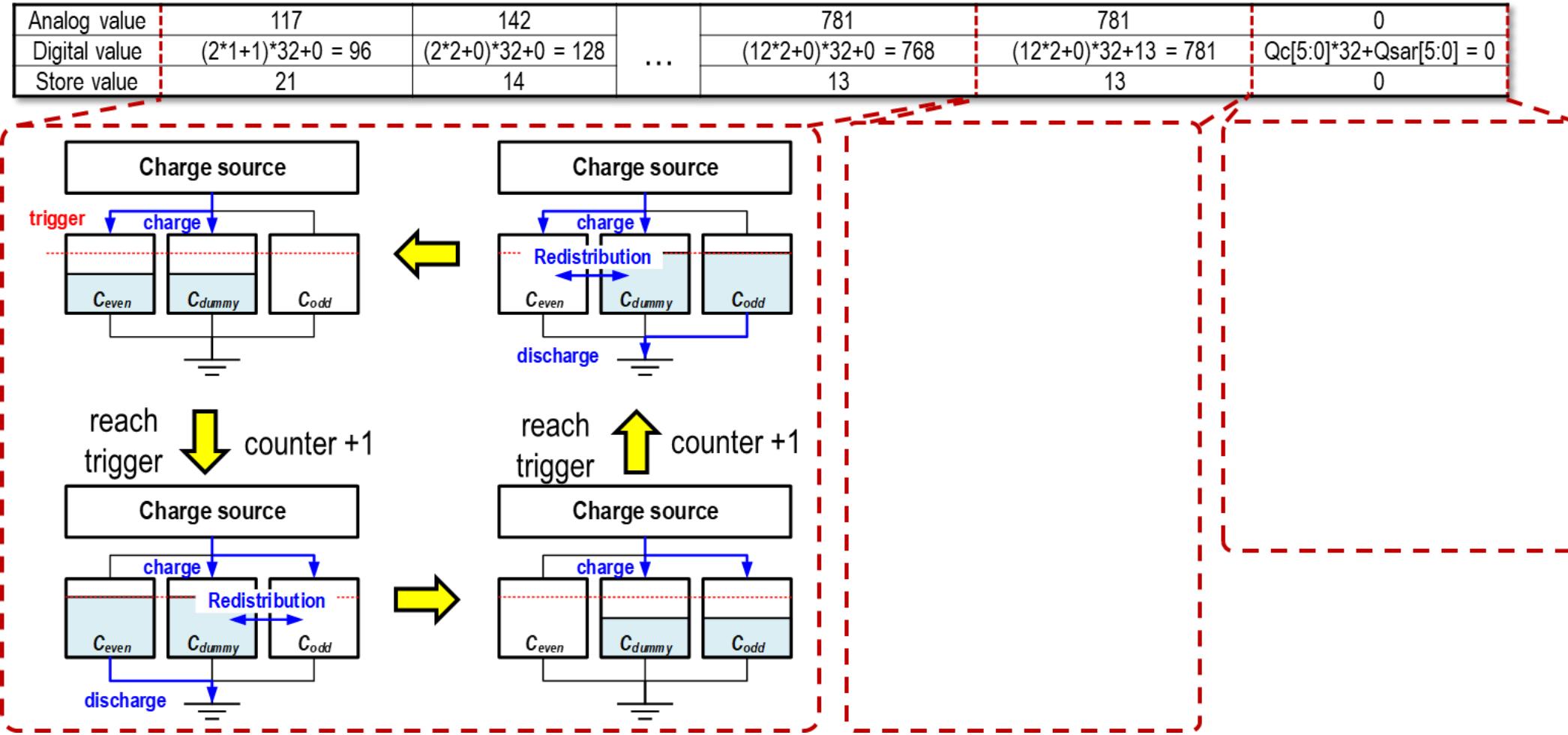
# Analog-Storage Quantizer

- ASQC circuit design
  - Double-PMOS array to generate accumulation charge
  - Two capacitors to store accumulation charge
  - Counter to record coarse results and SAR ADC to quantize fine results



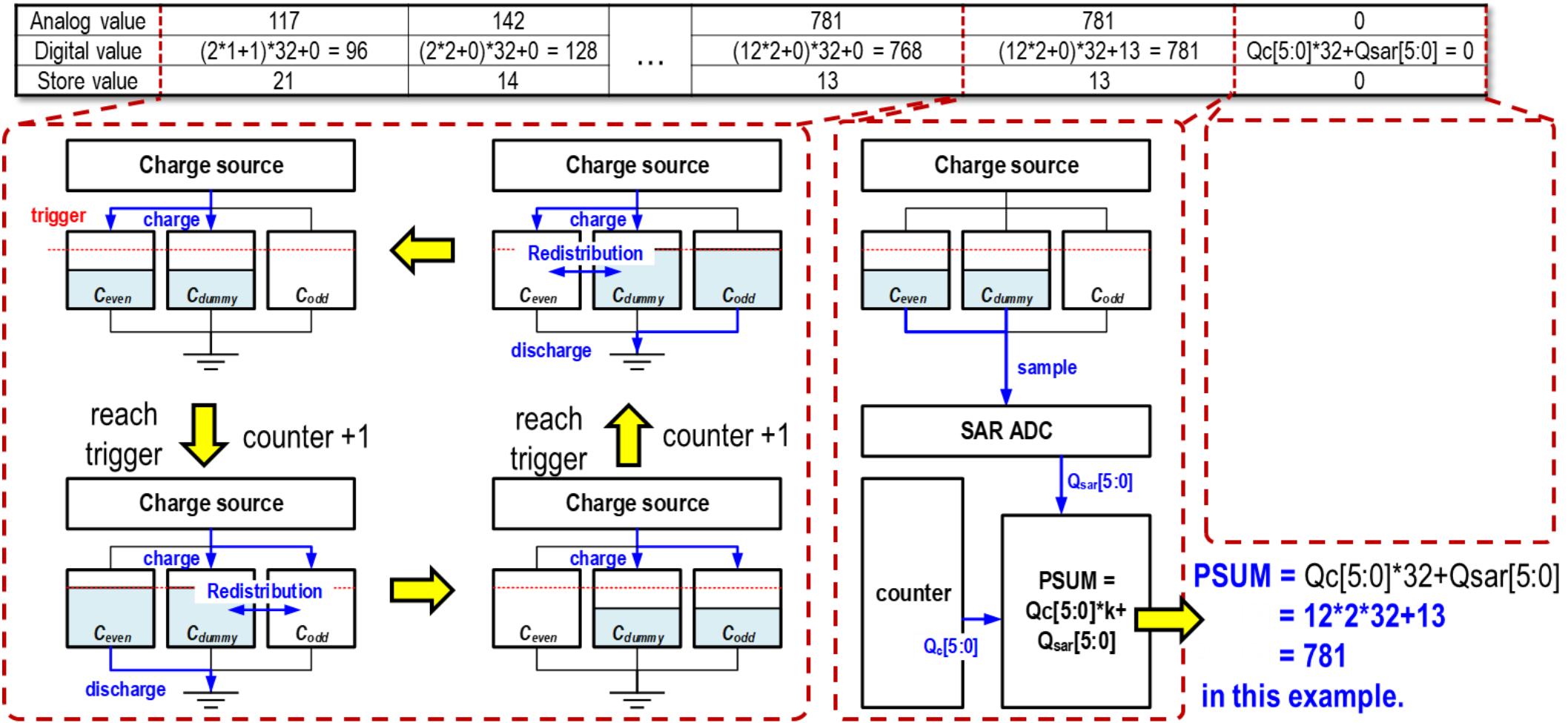
# Analog-Storage Quantizer

## □ Phase 1: analog-storage and coarse quantization



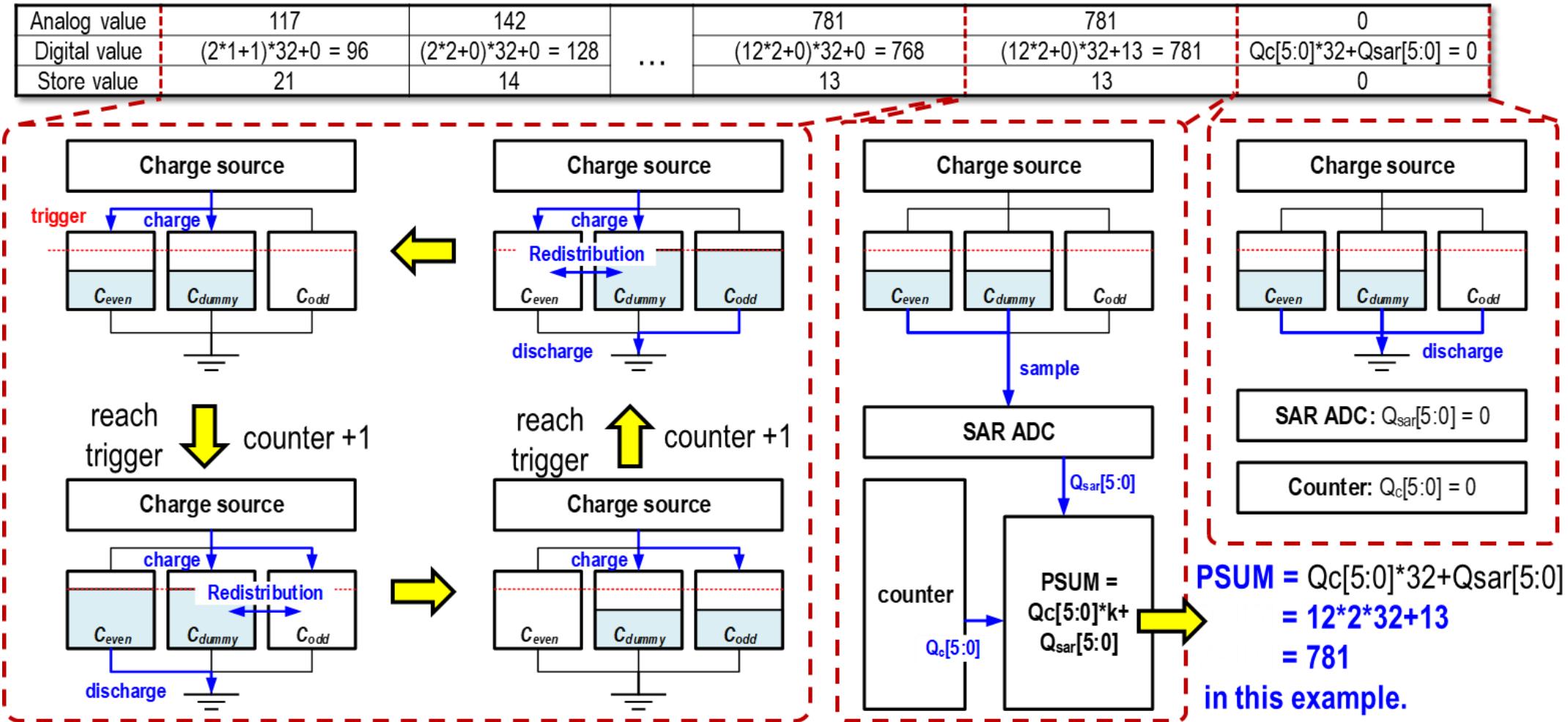
# Analog-Storage Quantizer

## Phase 2: Fine quantization and PSUM generation



# Analog-Storage Quantizer

## Phase 3: Capacity, SAR ADC and counter reset



# Outline

---

- ❑ Motivation, Background, and Challenges
- ❑ Proposed Hybrid Computing-in-Memory (CIM) Macro
  - Lightning-like macro level structure
  - Compressed Adder Tree
  - Analog-Storage Quantizer
- ❑ Measurement Results
- ❑ Conclusion

# Chip Summary



<sup>1</sup>Measured with worst case under 0.9V. Worst case: 50% input sparsity, 50% input toggle rate, 0% weight sparsity.

<sup>2</sup>Measured with average case under 0.6V.

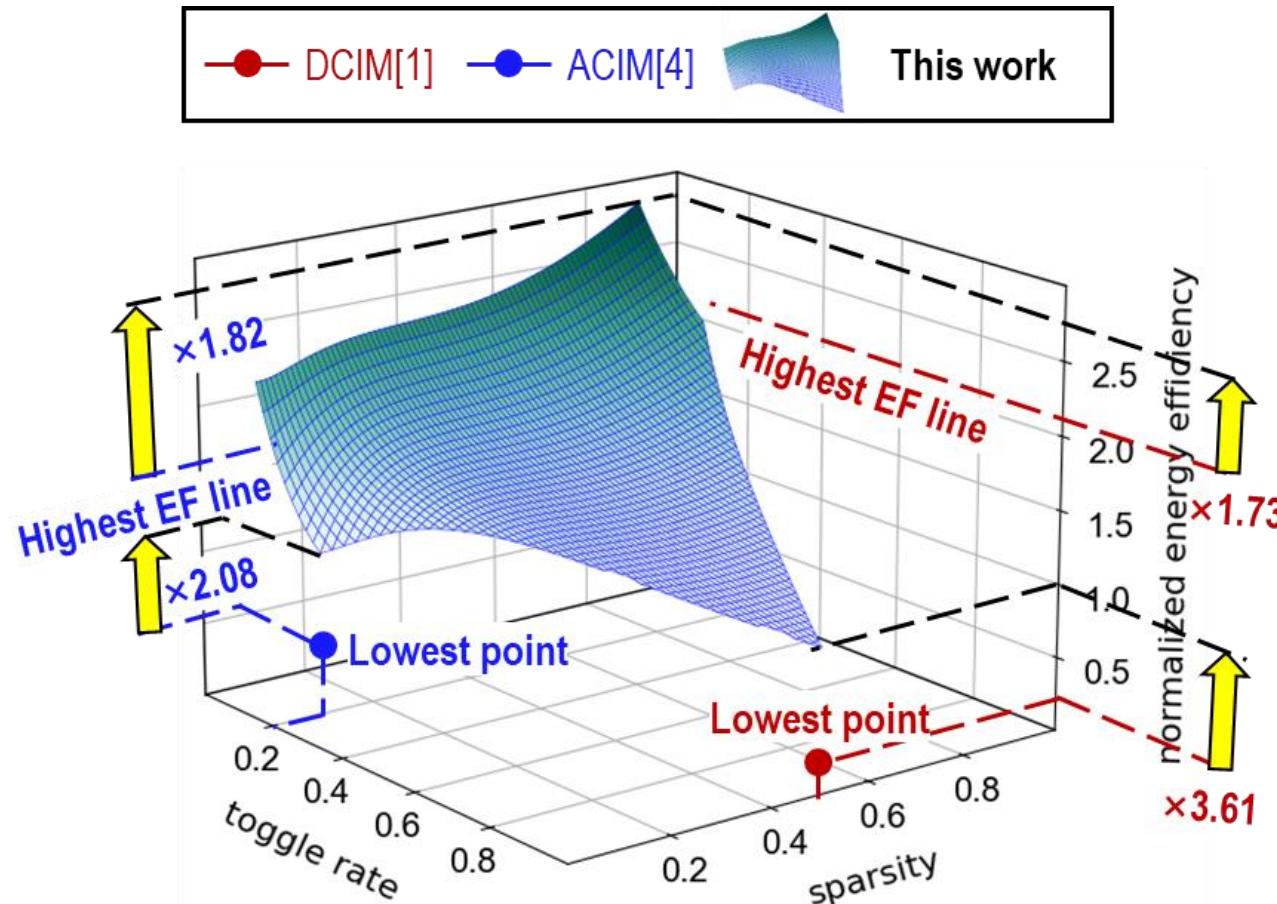
## CHIP SUMMARY

Technology	22nm CMOS
Bitcell rule	Compact rule
Cell structure	6T+DU/AU (logic rule)
Macro size	64Kb
Macro area	472.27umx251.71um =0.119mm <sup>2</sup>
Array size	403.95umx92.68um =0.037mm <sup>2</sup>
Write energy(pJ/bit)	1.53 @0.9V
Input precision(bit)	8
Weight precision(bit)	8
Output precision(bit)	23-27
Supply voltage(V)	0.6-0.9
Accumulation length	128-2048
Output ratio	1
RMS error	1.9%
Access time(ns)	4.1 @0.9V
Energy efficiency(TOPS/W)	<sup>1</sup> 20.7- <sup>2</sup> 60.8

# Lightning-like Hybrid CIM Macro

## □ Better EF distribution due to Hybrid CIM

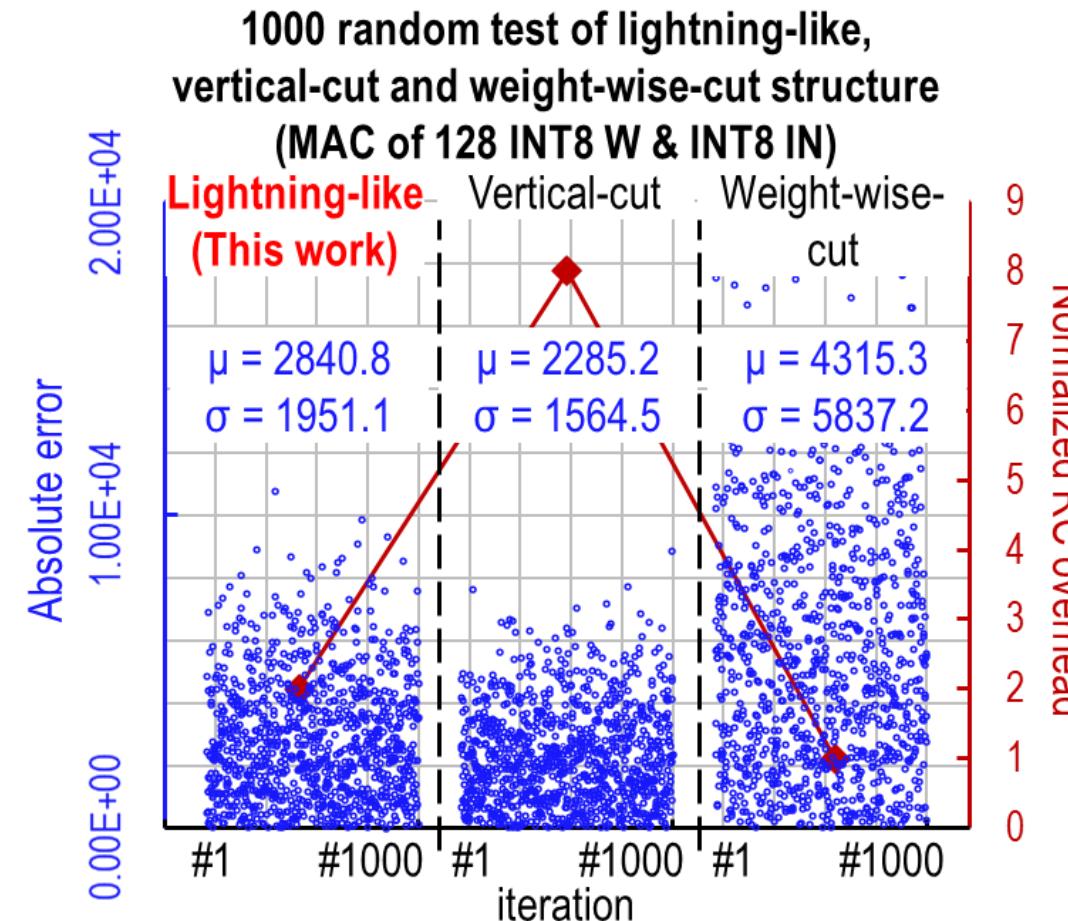
- >2.08 better worst EF, >1.82 better peak EF
- Smaller EF variation compared with DCIM and ACIM



34.3: A 22-nm 64-kb Lightning-like Hybrid Computing-in-Memory Macro with a Compressed Adder Tree and Analog-Storage Quantizers for Transformer and CNNs

# Lightning-like Hybrid CIM Macro

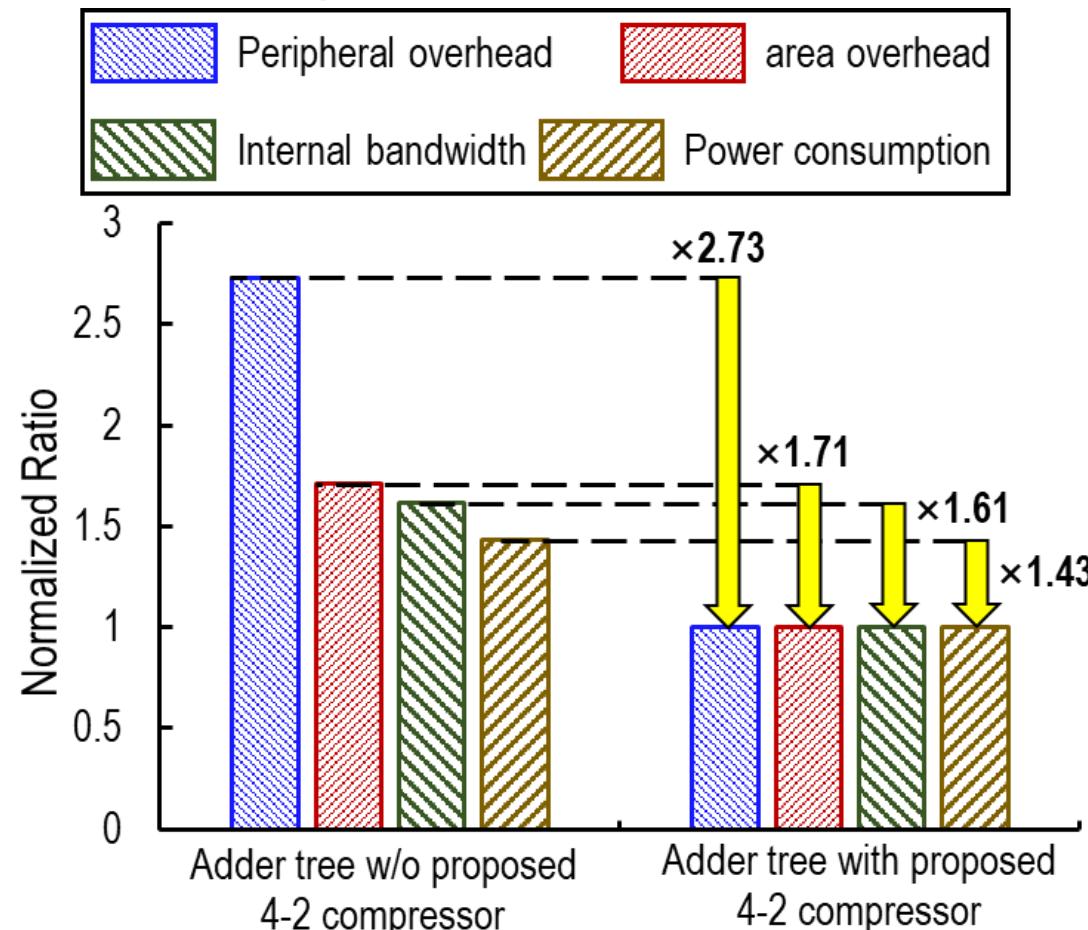
- Better performance due to Lightning structure
  - 4x RC saving compared with vertical-cut structure
  - 2.99x accuracy improvement compared with weight-wise-cut structure



34.3: A 22-nm 64-kb Lightning-like Hybrid Computing-in-Memory Macro with a Compressed Adder Tree and Analog-Storage Quantizers for Transformer and CNNs

# Lightning-like Hybrid CIM Macro

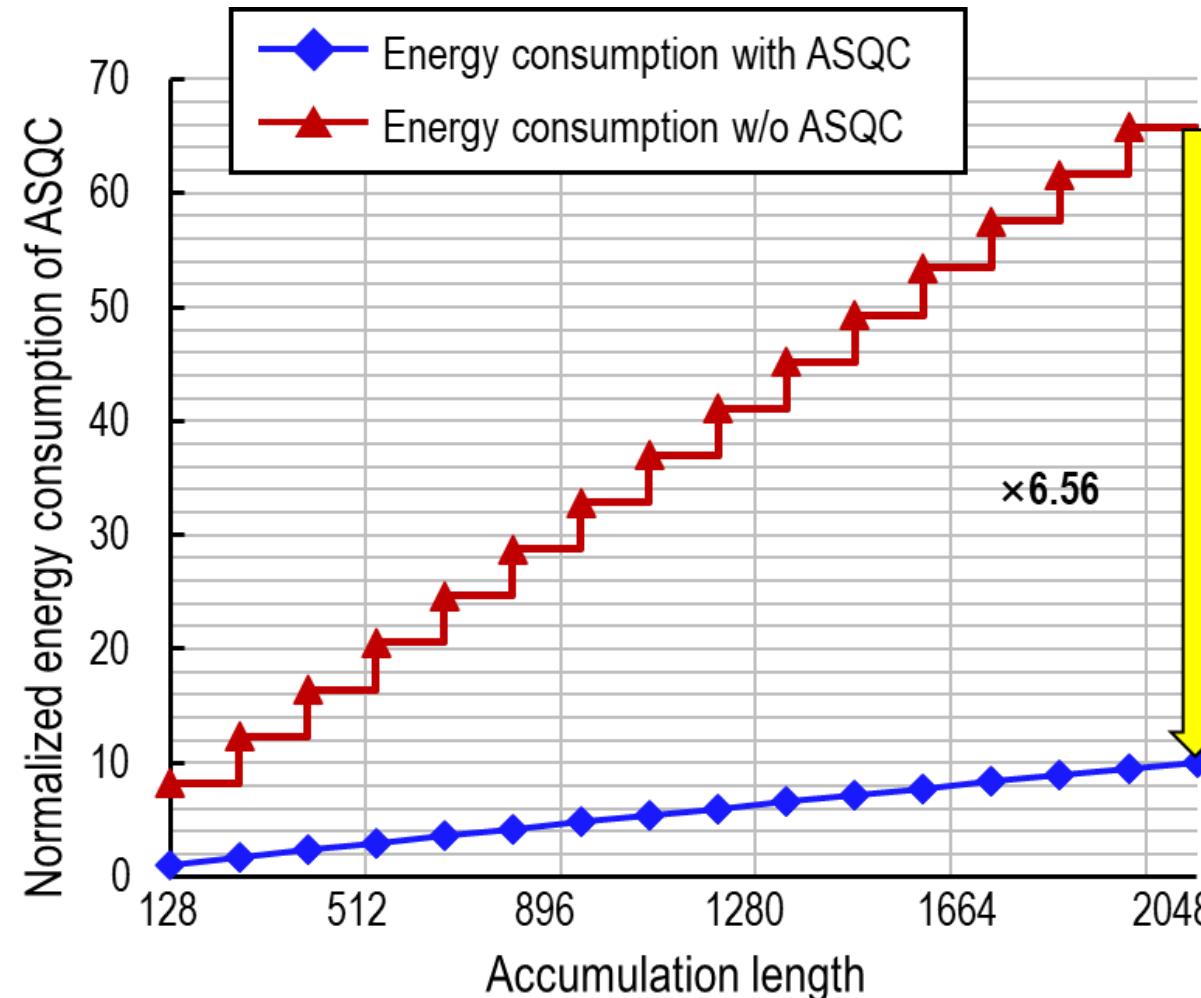
- Better performance due to 4-2 compressor
  - 2.73x less peripheral overhead and 1.61x less internal bandwidth
  - 1.71x and 1.43x less area and power cost



# Lightning-like Hybrid CIM Macro

## □ Better EF due to ASQC

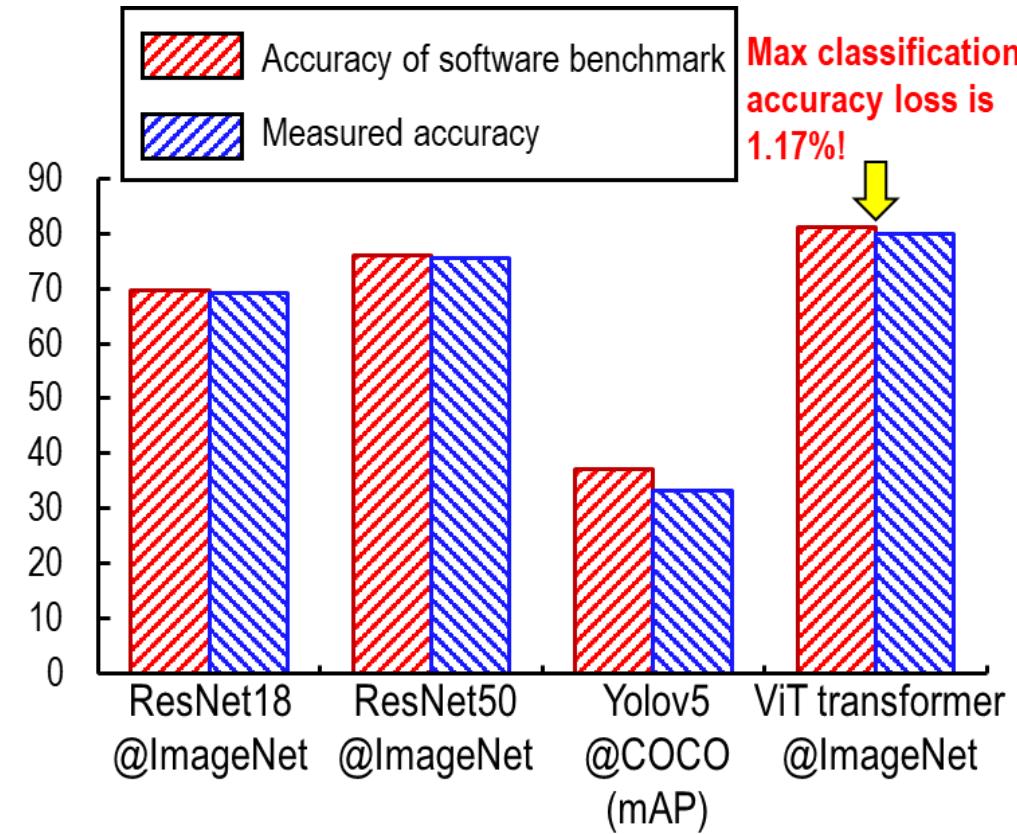
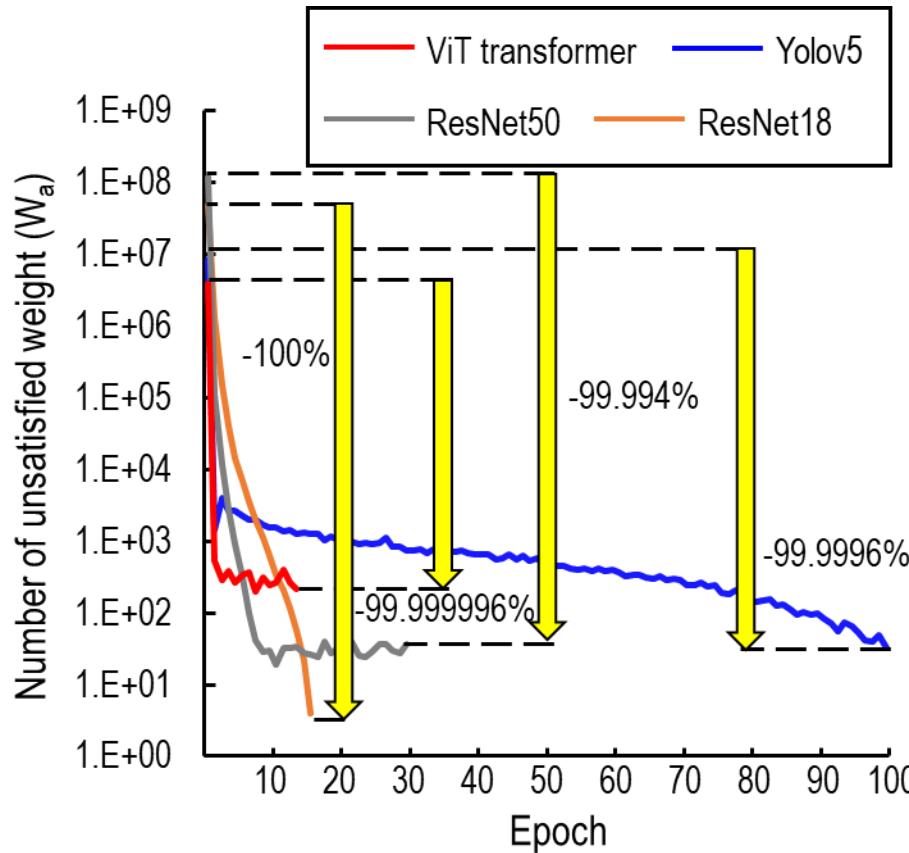
- 6.56x energy consumption saving compared with benchmark



34.3: A 22-nm 64-kb Lightning-like Hybrid Computing-in-Memory Macro with a Compressed Adder Tree and Analog-Storage Quantizers for Transformer and CNNs

# Lightning-like Hybrid CIM Macro

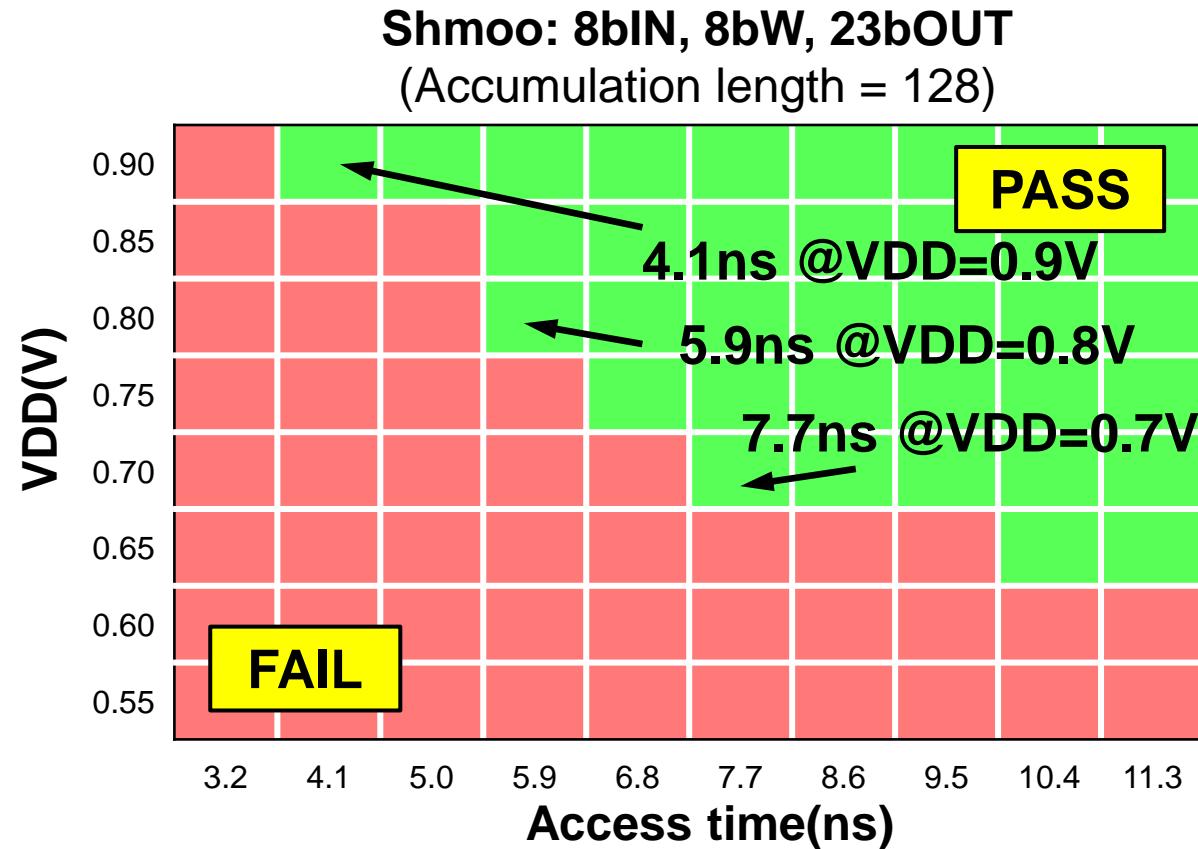
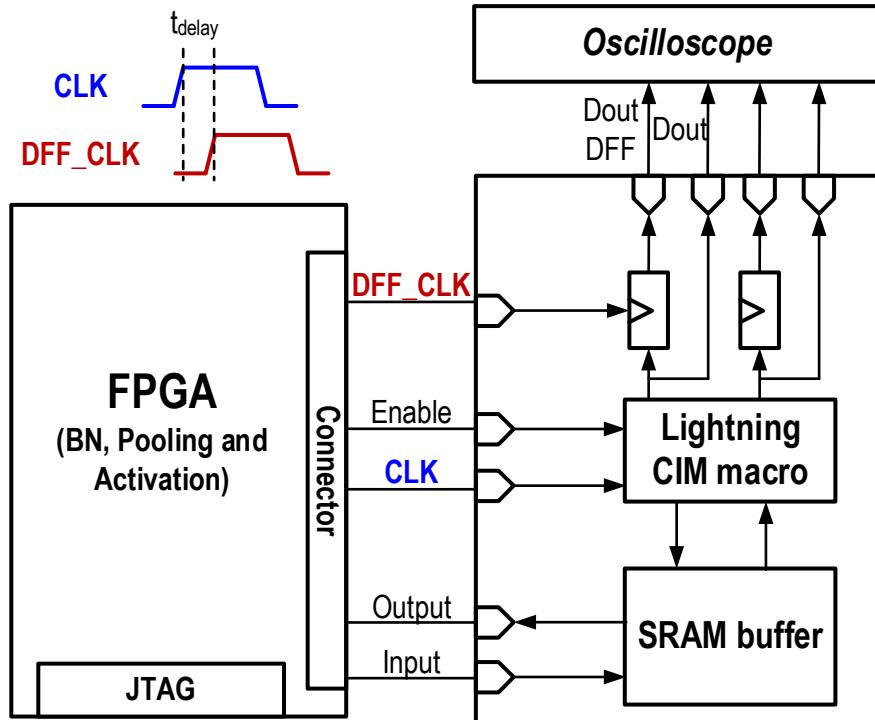
- Application performance based on double-regularization
  - >99.994% error point changed on multi-networks
  - <1.17% accuracy loss compared with software baseline



# Measurement

## □ Proposed Hybrid CIM Macro

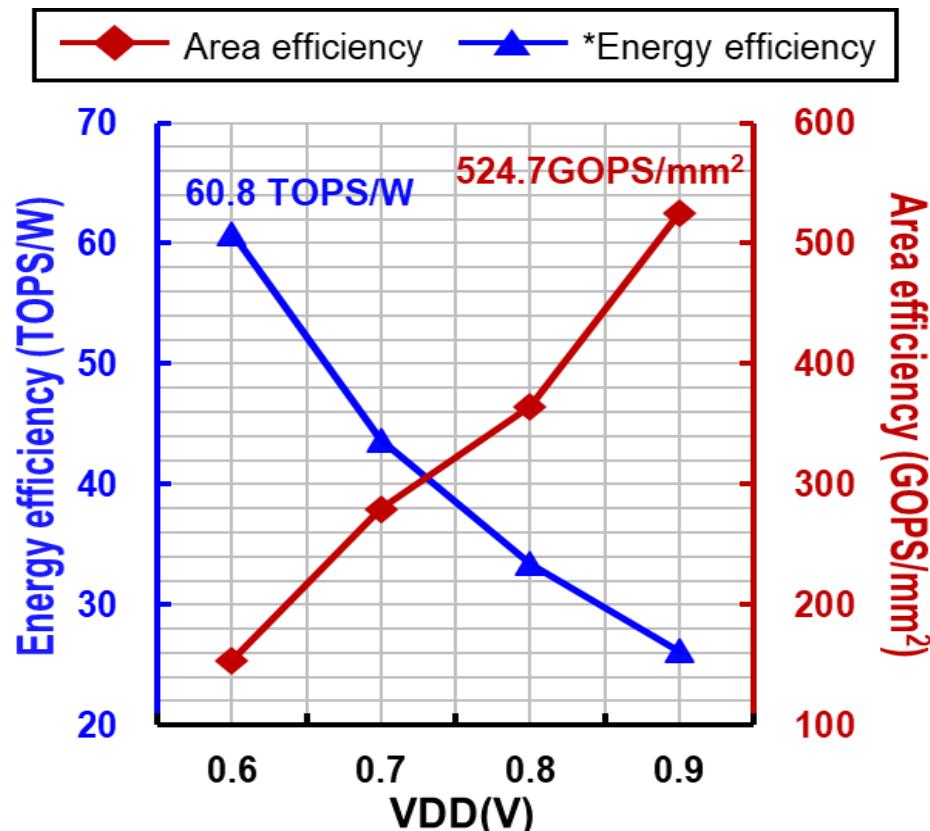
- Access time ( $T_{AC}$ ) = 4.1ns @ 0.9V, 8bIN-8bW-23bOUT
- $T_{AC}$  = 10.4ns @ 0.6V, 8bIN-8bW-23bOUT



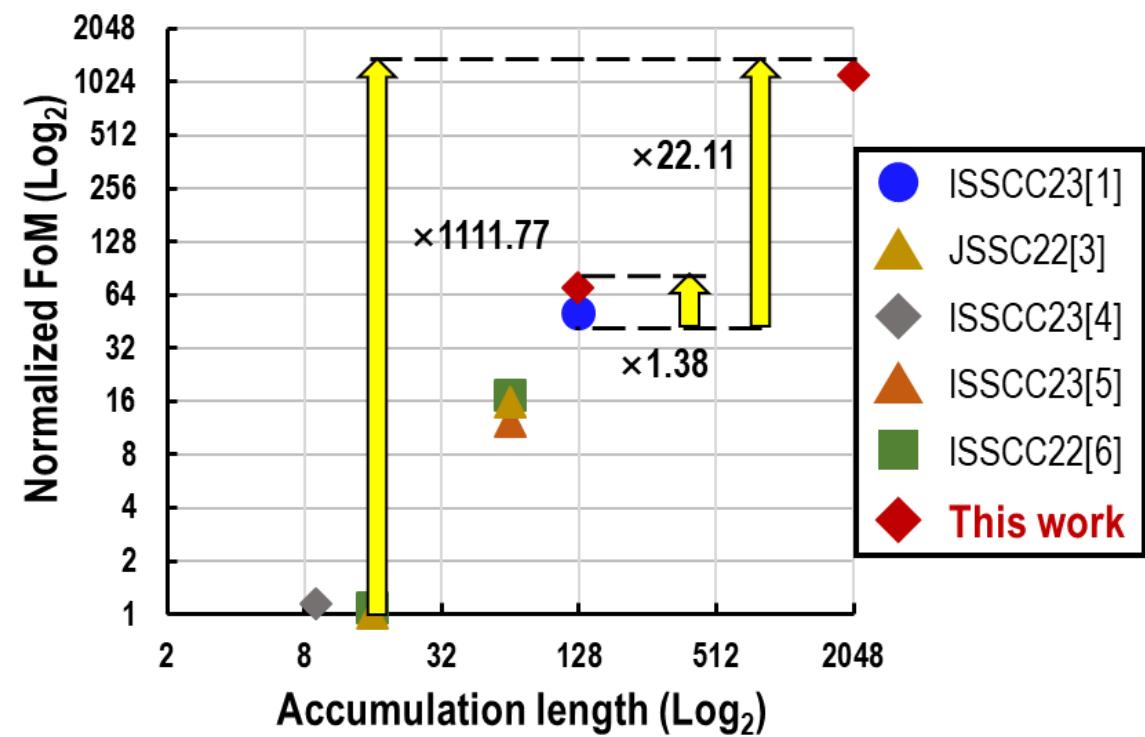
# Measurement

## □ Proposed Hybrid CIM Macro

- 60.8TOPS/W@ 0.6V and 524.7GOPPS/mm<sup>2</sup>@ 0.9V
- >22.11x FoM improvement (Compared with JSSC'22 &'23, ISSCC'22 &23)



\*Measured with actual ResNet18 model @ImageNet.

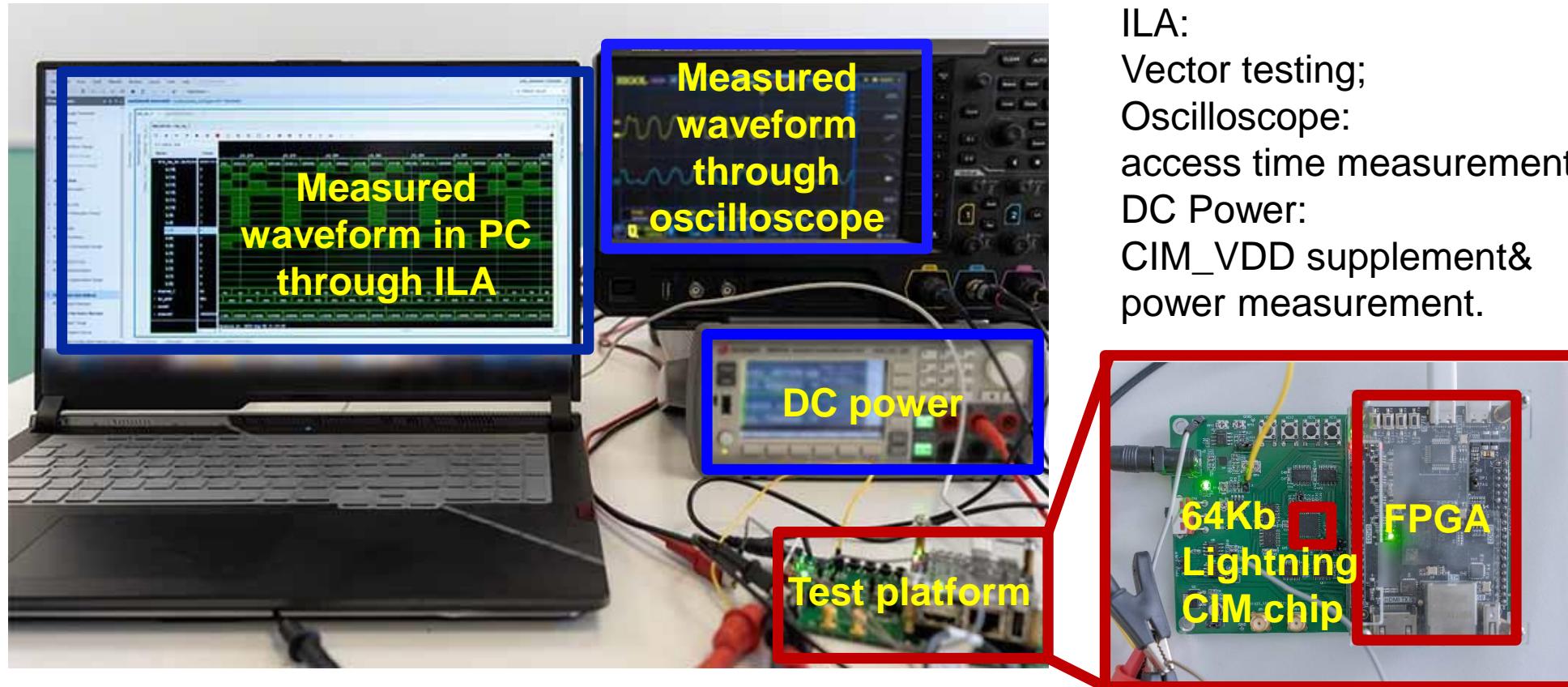


FoM=Energy Efficiency × input precision × weight precision × accumulation length × output ratio

# Demo System Setup

## □ Proposed Hybrid CIM Macro

- Test on ResNet and Vision-Transformer (ViT)



# Conclusion

---

- Features of proposed Lightning-like hybrid CIM macro
  - Lightning-like hybrid CIM architecture
    - ◇ 4x RC decrease and 2.99x accuracy improvement
  - NOR-based 4-2-compressor for high-precision digital calculation
    - ◇ 1.71x less area consumption and 1.43x power saving
  - Analog-Storage Quantizer (ASQC) for low-power analog calculation
    - ◇ 6.56x less energy cost and better mitigating error accumulation
- A 22nm 64Kb lightning-like hybrid CIM macro is verified
  - 8bIN-8bW-23bOUT
    - ◇ Access time: 4.1ns, 524.7GOPS/mm<sup>2</sup>@0.9V
    - ◇ 60.8TOPS/W@0.6V

---

# Thanks for your kind attention!



Please Scan to Rate  
This Paper



# A 3nm 32.5 TOPS/W, 55.0 TOPS/mm<sup>2</sup> and 3.78 Mb/mm<sup>2</sup> Fully Digital Computing-in-Memory Supporting INT12 x INT12 with Parallel MAC Architecture

Hidehiro Fujiwara<sup>1</sup>, Haruki Mori<sup>1</sup>, Wei-Chang Zhao<sup>1</sup>, Kinshuk Khare<sup>1</sup>,  
Cheng-En Lee<sup>1</sup>, Xiaochen Peng<sup>2</sup>, Vineet Joshi<sup>3</sup>, Chao-Kai Chuang<sup>1</sup>,  
Shu-Huan Hsu<sup>1</sup>, Takeshi Hashizume<sup>4</sup>, Toshiaki Naganuma<sup>4</sup>,  
Chen-Hung Tien<sup>1</sup>, Yao-Yi Liu<sup>1</sup>, Yen-Chien Lai<sup>1</sup>, Chia-Fu Lee<sup>1</sup>, Tan-Li Chou<sup>1</sup>,  
Kerem Akarvardar<sup>2</sup>, Saman Adham<sup>3</sup>, Yih Wang<sup>1</sup>, Yu-Der Chih<sup>1</sup>,  
Yen-Huei Chen<sup>1</sup>, Hung-Jen Liao<sup>1</sup>, Tsung-Yung Jonathan Chang<sup>1</sup>



<sup>1</sup> TSMC, Hsinchu, Taiwan

<sup>2</sup> TSMC, San Jose, CA

<sup>3</sup> TSMC, Ottawa, TX

<sup>4</sup> TSMC, Yokohama, Japan

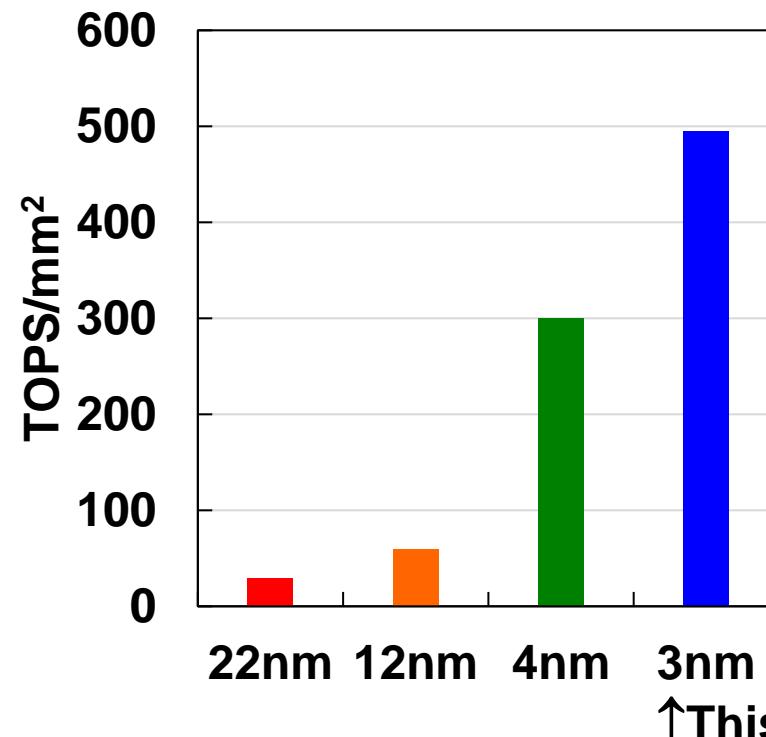
# Outline

- **Background**
- **Circuit Implementation**
- **Si Measurement Results**
- **Conclusion**

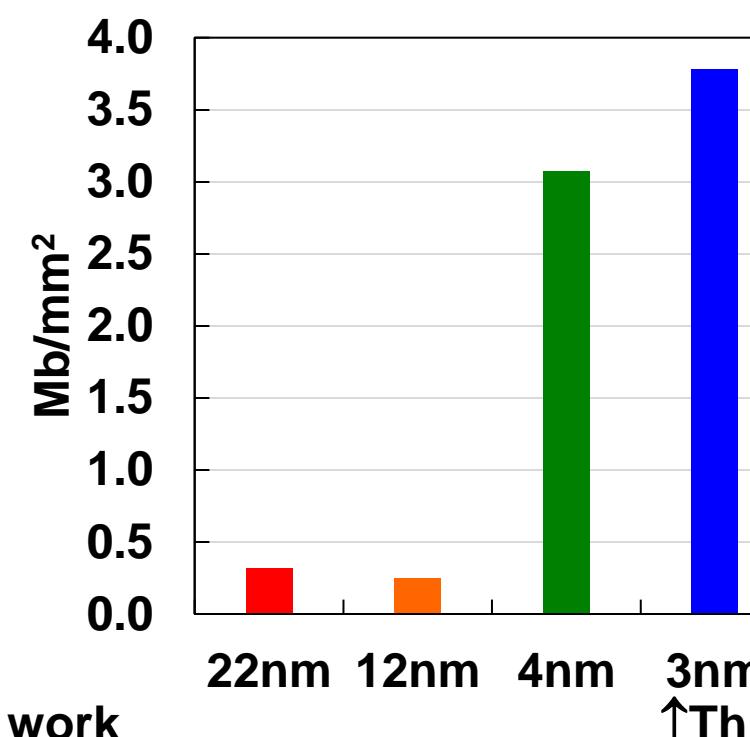
# Digital CIM

- Data movement and multiply and accumulate (MAC) are key
- Digital CIM (DCIM): distributed banks + customized MAC unit
  - Directly leverage technology scaling. Better testability.

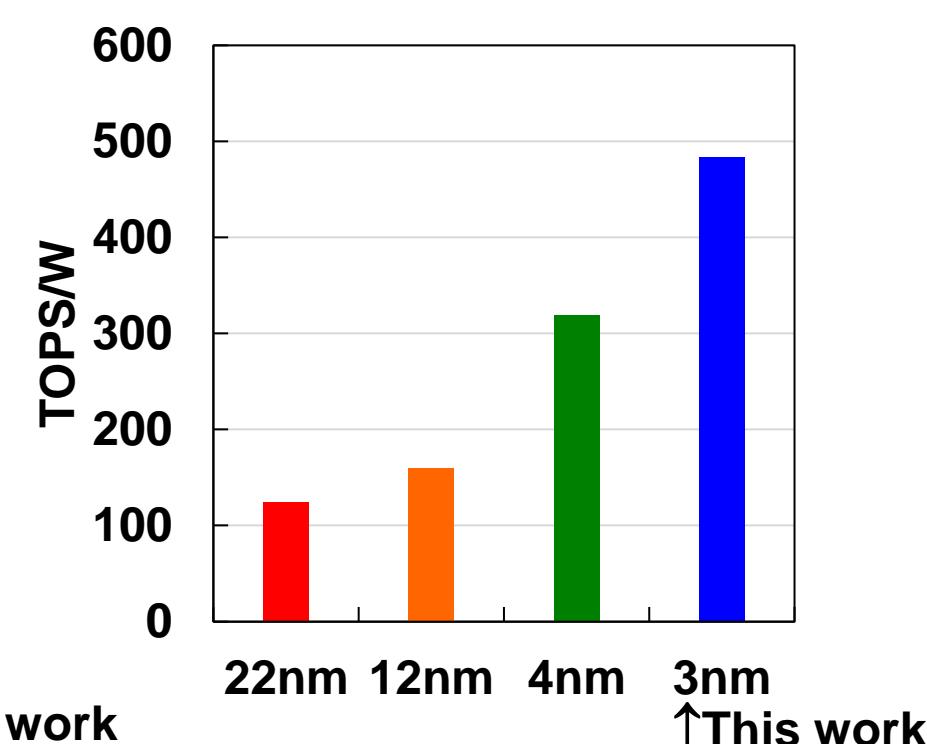
Computing area efficiency



Memory area efficiency



Energy efficiency

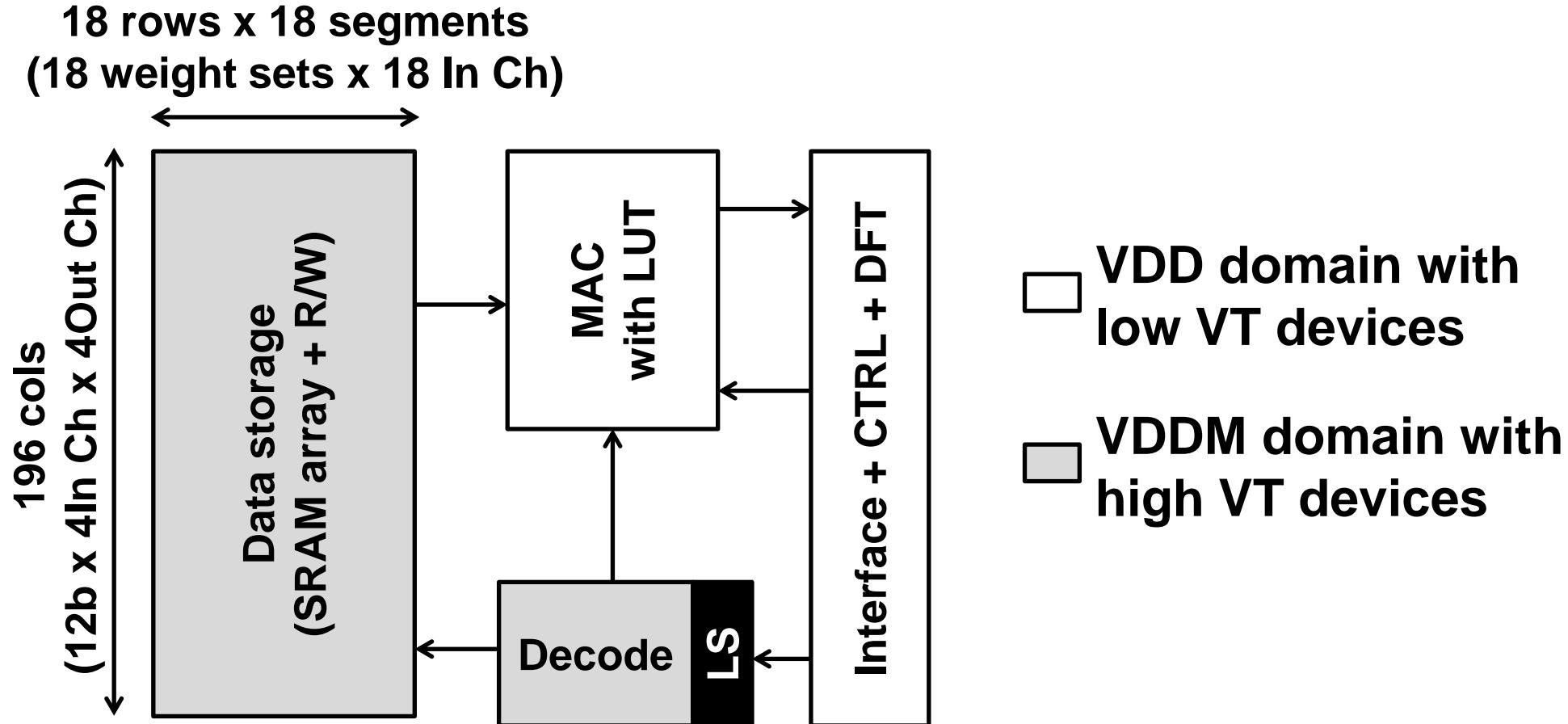


# Key Features

- 72 Input Ch x 4 Out Ch with signed INT12 format
- 18 sets of weight in sub array with foundry 6T cell
- Dual rail design
- Parallel MAC with look up table (LUT)
- DFT & BIST interface + CIM BIST

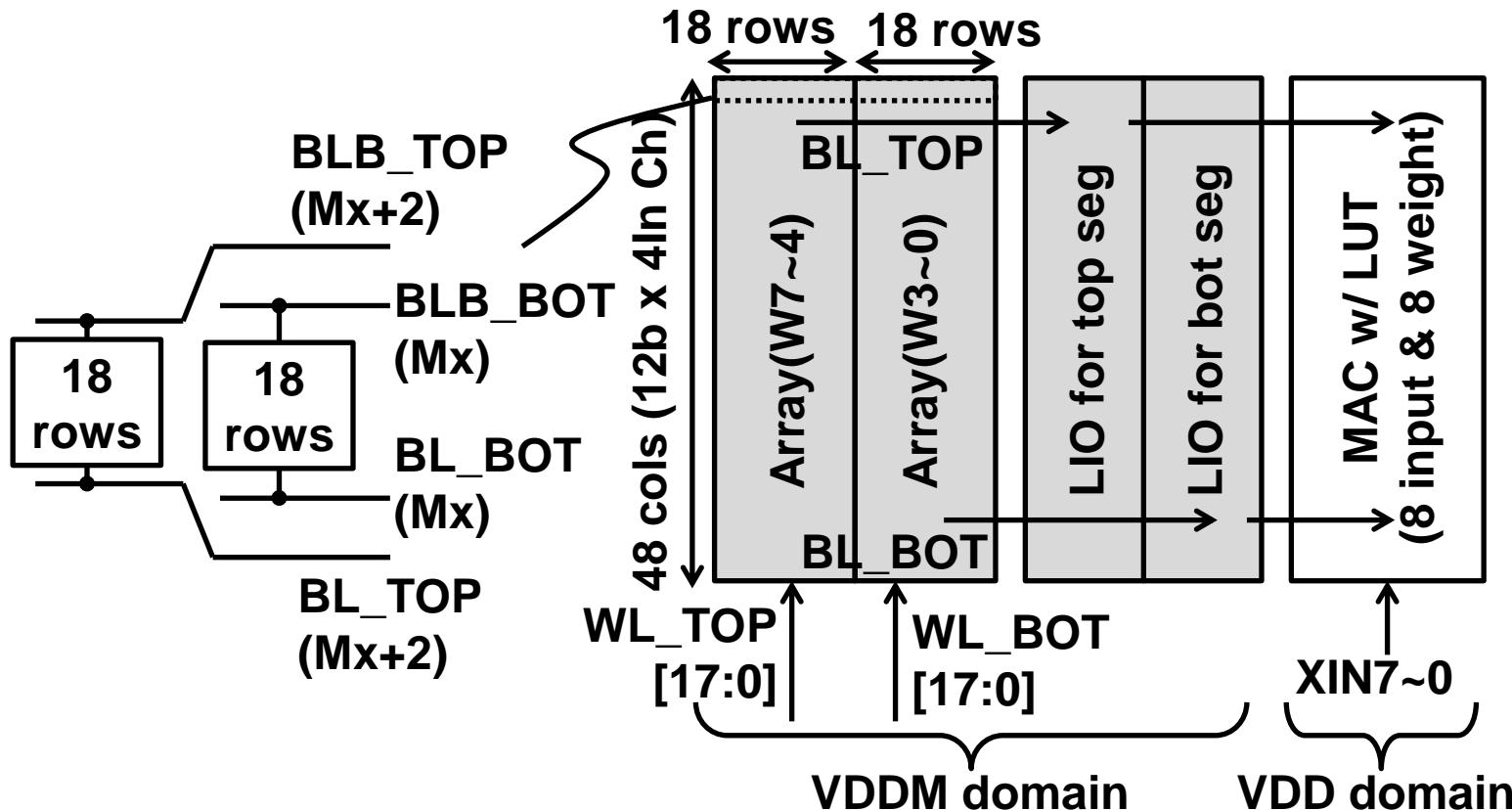
# Block Diagram

- Interface dual rail (IDR) like design
- MAC in VDD (low voltage) domain



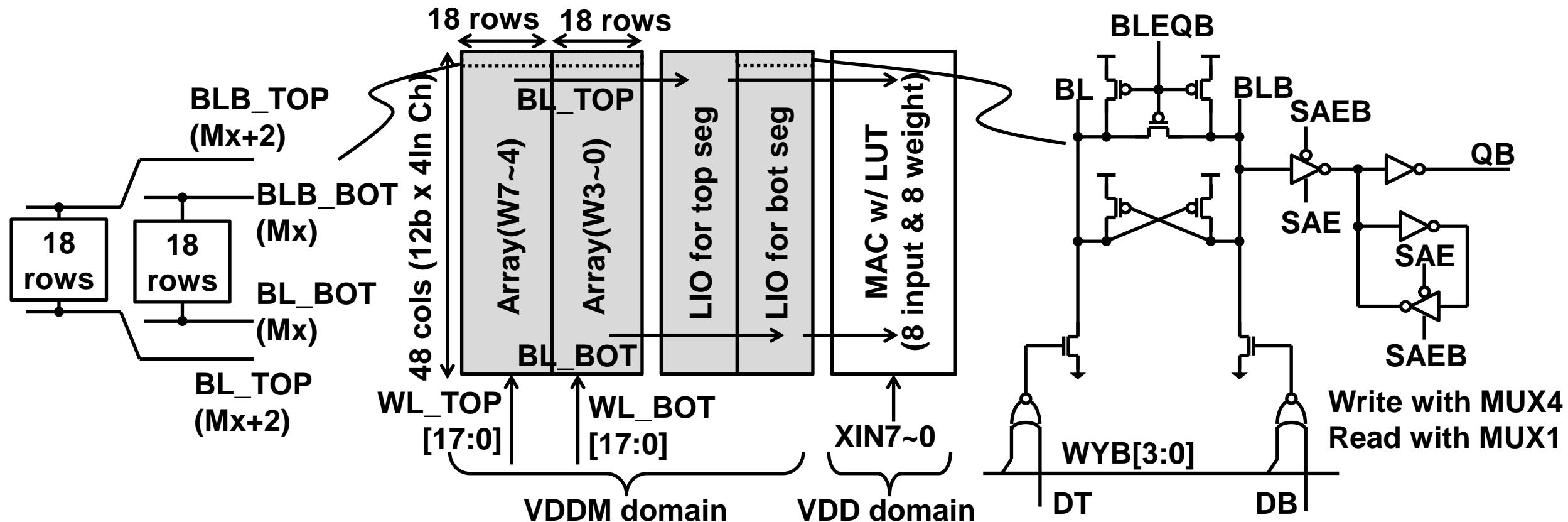
# Sub Array with Flying BL

- Foundry 6T cell with flying BL architecture
  - ~5% macro area reduction by array layout optimization
- Simultaneous access to all banks



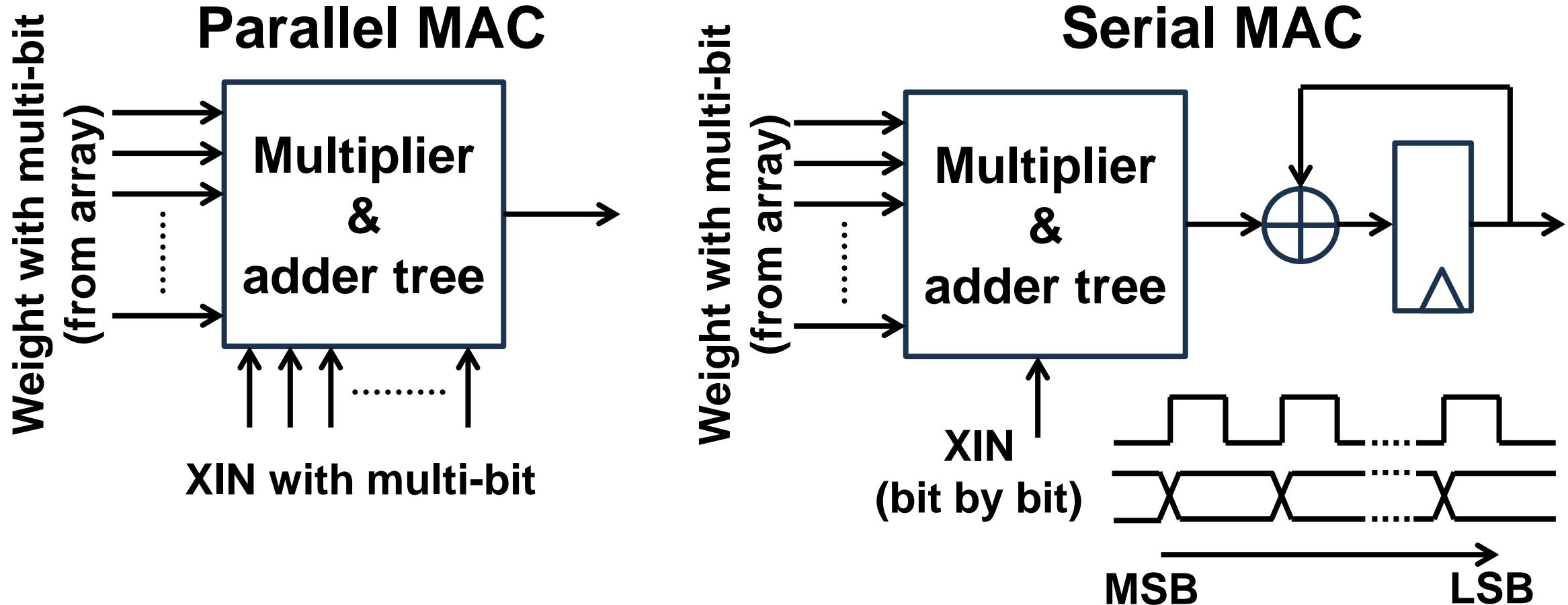
# Simple Local IO

- Read with MUX1: Better throughput
- Write with MUX4: Global routing track saving



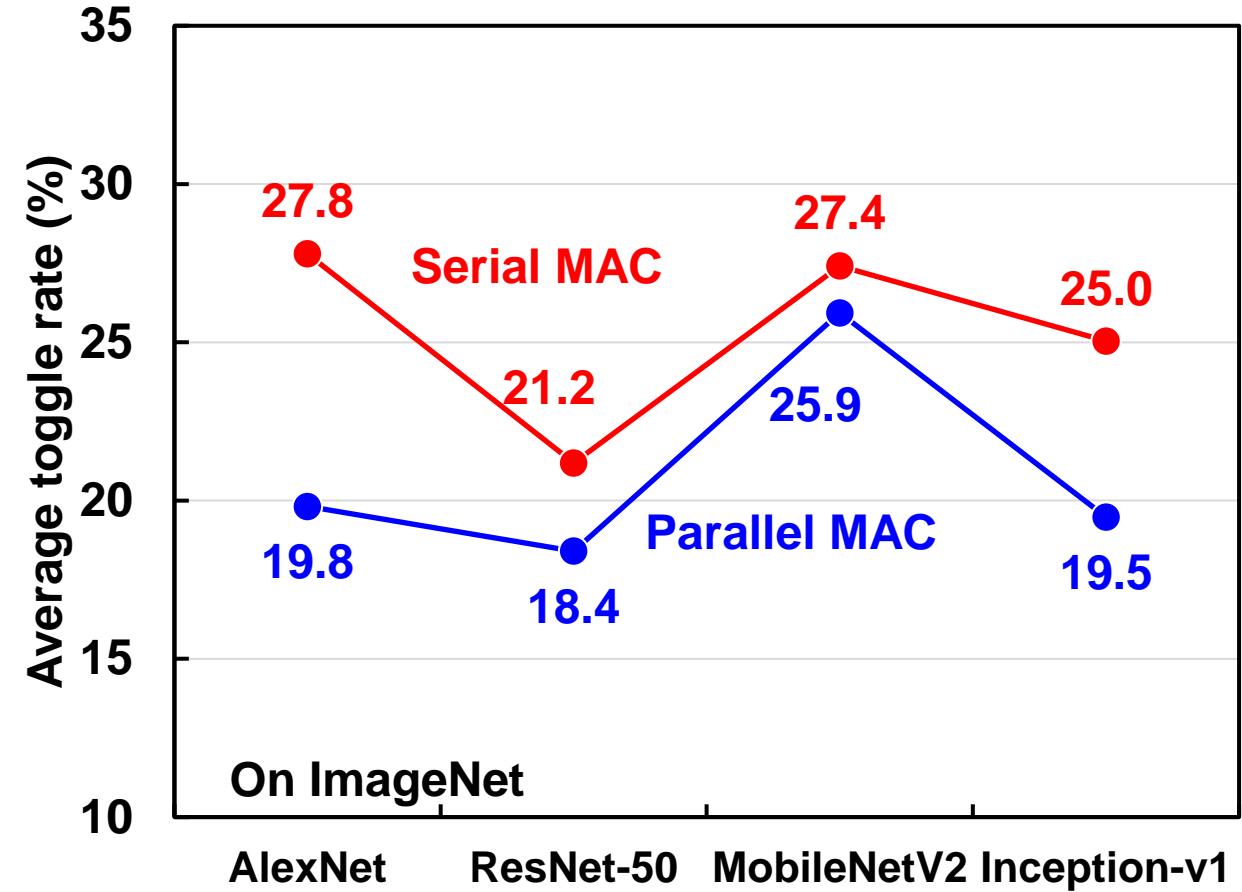
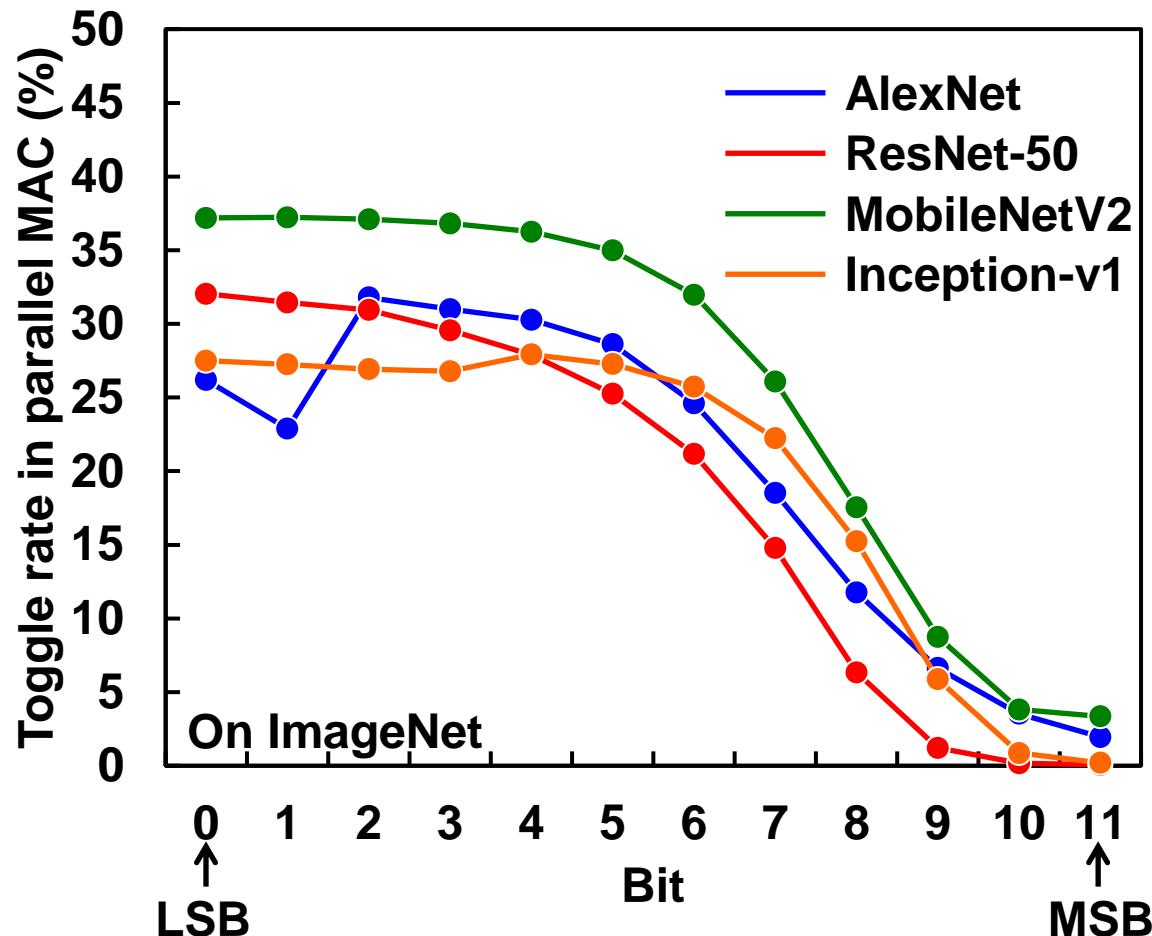
# Parallel MAC vs Serial MAC

- Parallel MAC: multi-bit x multi-bit
- Serial MAC: multi-bit x single bit + shift & add



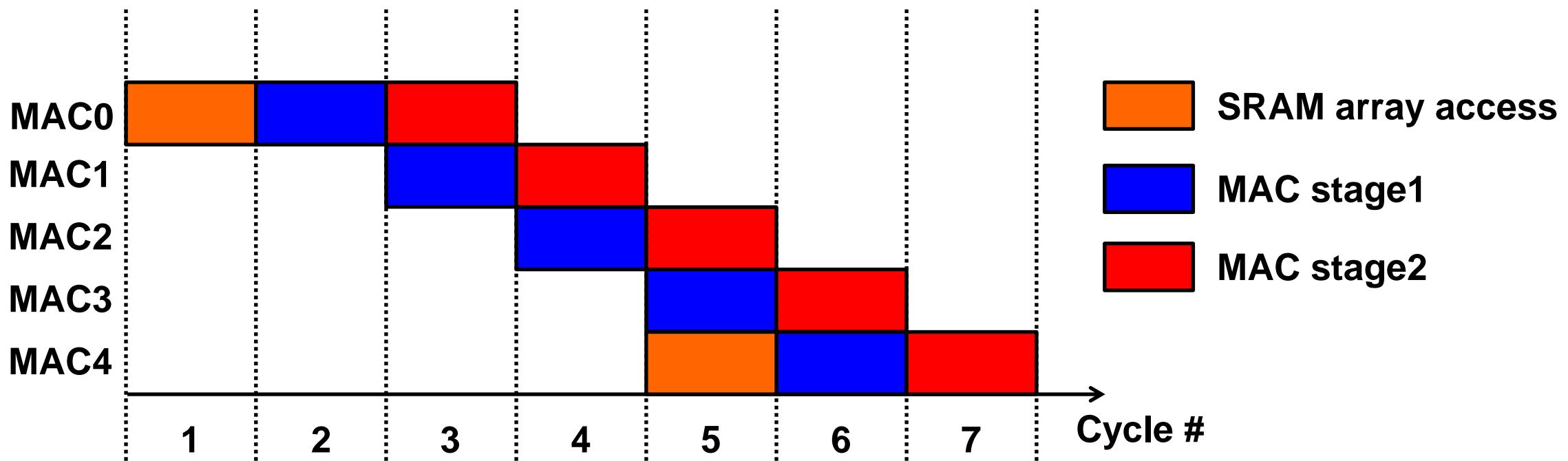
# Less Energy in Parallel MAC

- High throughput & low toggle rate in parallel MAC
- 7% less devices and 5% faster speed using LUT approach



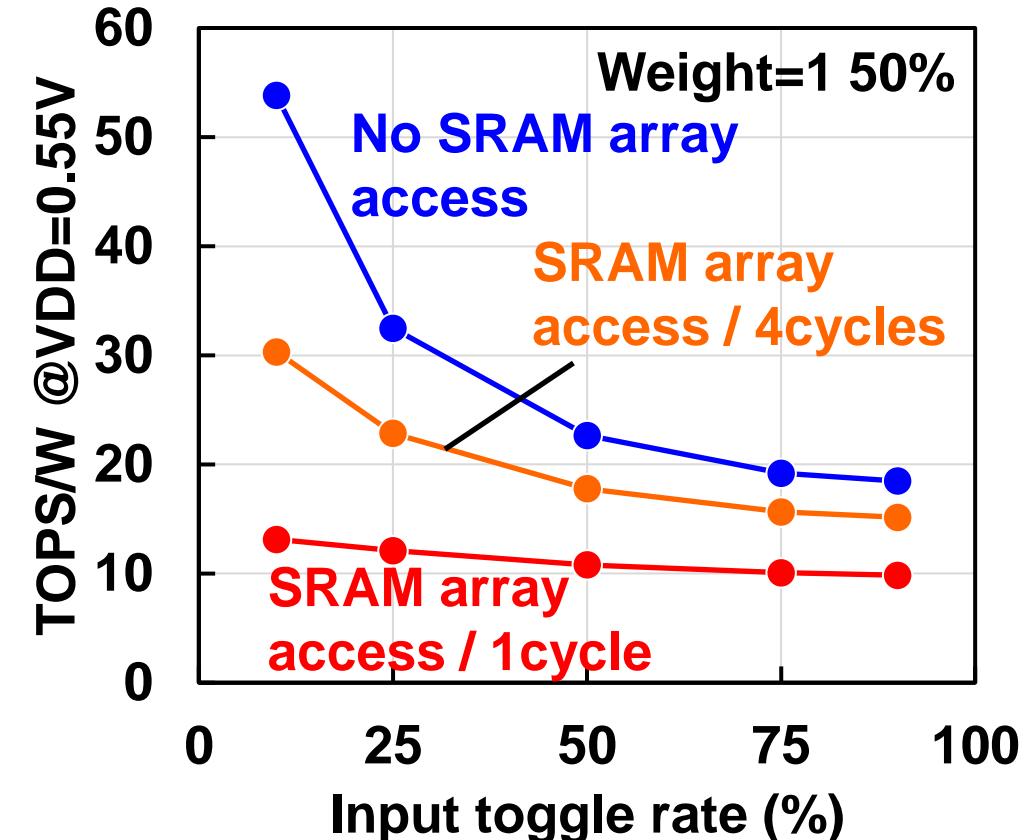
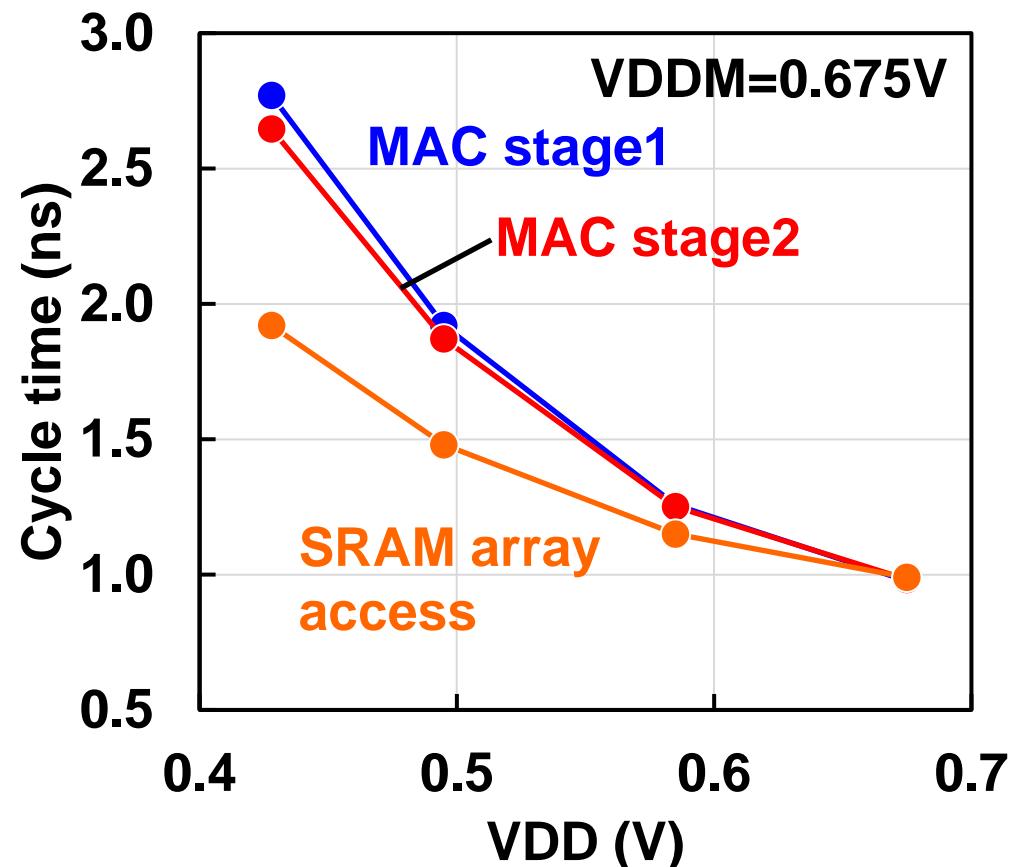
# Pipeline Structure

- 3 stages: SRAM array access + 2 MAC stages
- SRAM array access only when weight update for MAC
  - MAC1 ~ MAC3: no weight update (data reuse)
  - MAC0 & MAC4: weight update



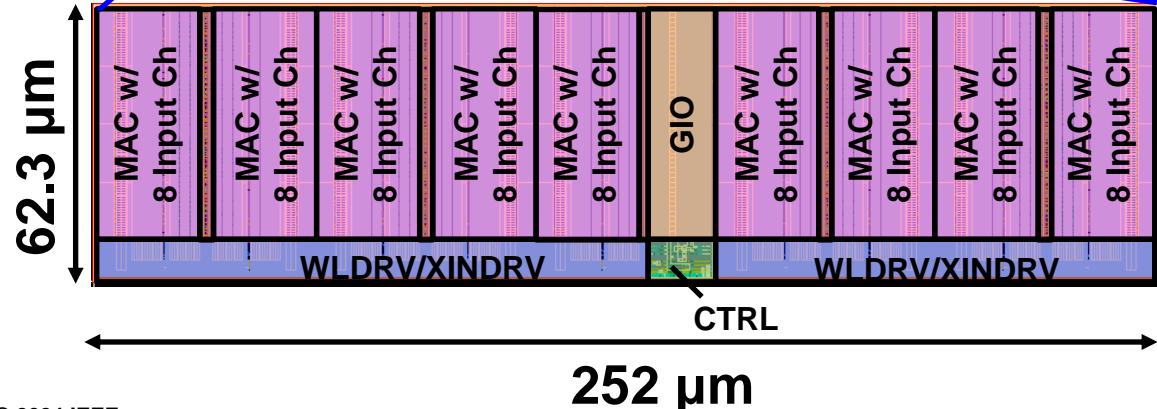
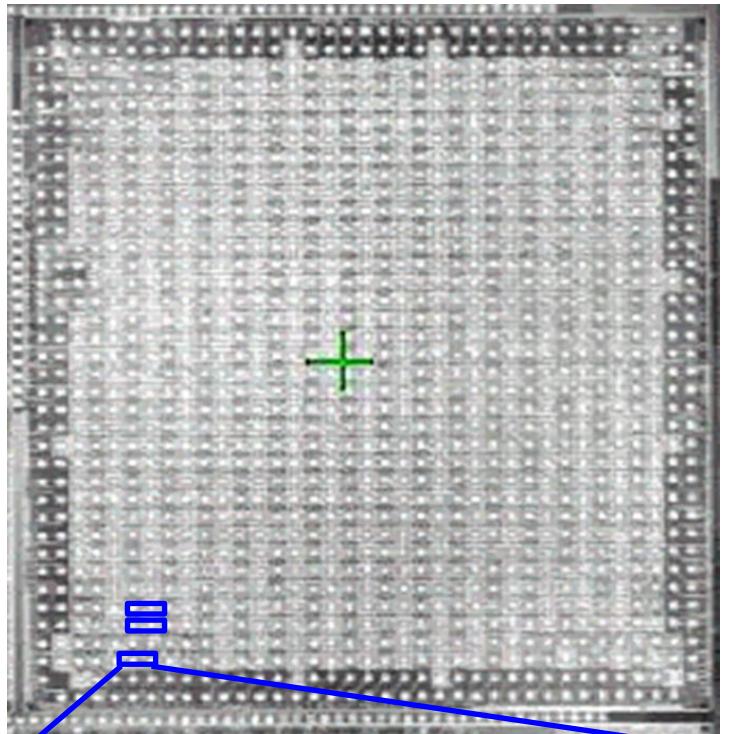
# Simulation Results

- Dedicated pipeline design with transistor level STA tool
- TOPS/W is up to input toggle rate and SRAM array access



SRAM array access	TOPS/W @ 0.55V		Input toggle	
	10%	25%	10%	25%
No	53.8	32.5	53.8	32.5
/ 4cycles	30.3	22.9	30.3	22.9

# 3nm Test Chip Micrograph



Bitcell	6T cell ( $0.026 \mu\text{m}^2$ )
Macro area	$0.0157 \text{ mm}^2$
Bit capacity	60.75 kb
MAC size	72 Input Ch x 4 Output Ch x 18 weight set
Bit density (Mb/mm <sup>2</sup> )	3.78
TOPS/mm <sup>2</sup>	10.6 @0.4V ~ 55.0 @0.9V

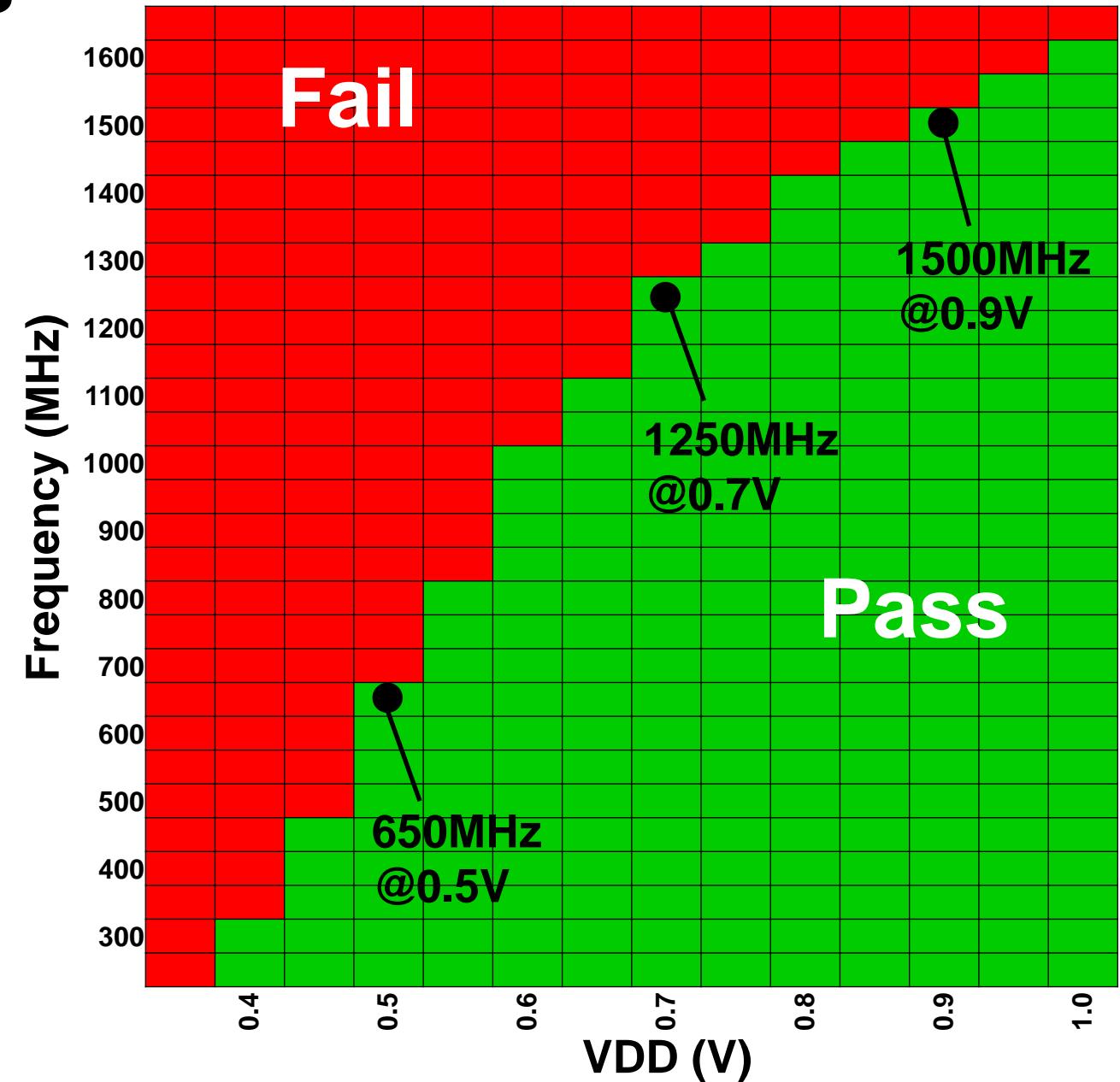
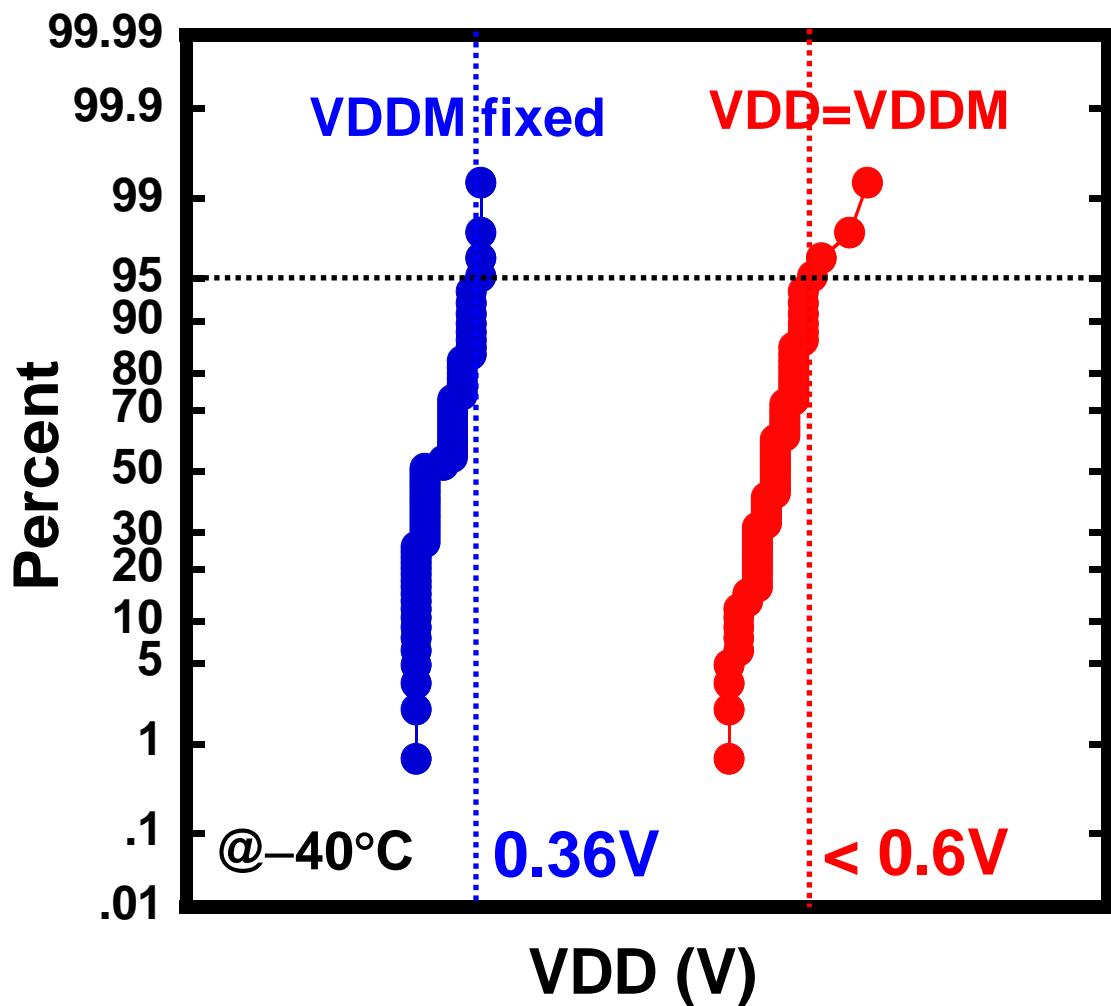
TOPS/W @0.55V	Input toggle	
	10%	25%
SRAM array access	No	53.8
	/ 4cycles	30.3
		32.5
		22.9

CIM BIST based on commercial MBIST engine

- > 99% fault coverage by custom algorithm
- Test time < 2ms / macro

# Measurement Results

@-40°C



# Comparison Table

	VLSI'22 [2]	VLSI'23 [4]	ISSCC'23 [5]	ISSCC'23 [1]	This work
Technology	12nm	12nm	18nm	4nm	3nm
Bitcell	NA	8T2P	8T2P	8Tx2 + OAI (2bit)	6T
VDD	0.72V	0.55 ~ 0.99V	0.525 ~ 1.0V	0.32 ~ 1.1V	0.36 ~ 1.1V
Array size (Kb)	8	64	256	54	60.75
weight sets	1	16	32	2	18
Macro area (mm <sup>2</sup> )	0.0323	0.0455	0.526	0.0172	0.0157
Simultaneous MAC + Write	No	Yes	Yes	Yes	Yes
Mb/mm <sup>2</sup>	0.24	1.37	0.48	3.07	3.78
Input Ch	64	64	32	64	72
Output Ch	16	16	64	64	4
Format	INT4	INT4	INT4	INT12	INT12
TOPS/W (*) (weight=1 50%)	121 (input=1 10%) @0.72V	137 @0.55V	58 (input=1 25%) @0.525V	319 (input=1 10%) 199 (input=1 25%) @0.55V	484 (input 10% toggle) 293 (input 25% toggle) @0.55V
TOPS/mm <sup>2</sup> (*)	41.6 @0.72V	11.3 @0.99V	13.6 @1.0V	299.7 @0.9V	495.3 @0.9V 1.65x

(\*) Normalized to INT4

# Conclusion

- 3nm DCIM with foundry 6T SRAM cell
- PPA summary (normalized to INT4)
  - 495.3 TOPS/mm<sup>2</sup> (1.65x from 4nm)
  - 293 TOPS/W with 25% input toggle (1.47x from 4nm)
  - 3.78 Mb/mm<sup>2</sup> (1.23x from 4nm)



Please Scan to Rate  
This Paper



# An 818-4094 TOPS/W Capacitor-Reconfigured CIM Macro for Unified Acceleration of CNNs and Transformers



Kentaro Yoshioka

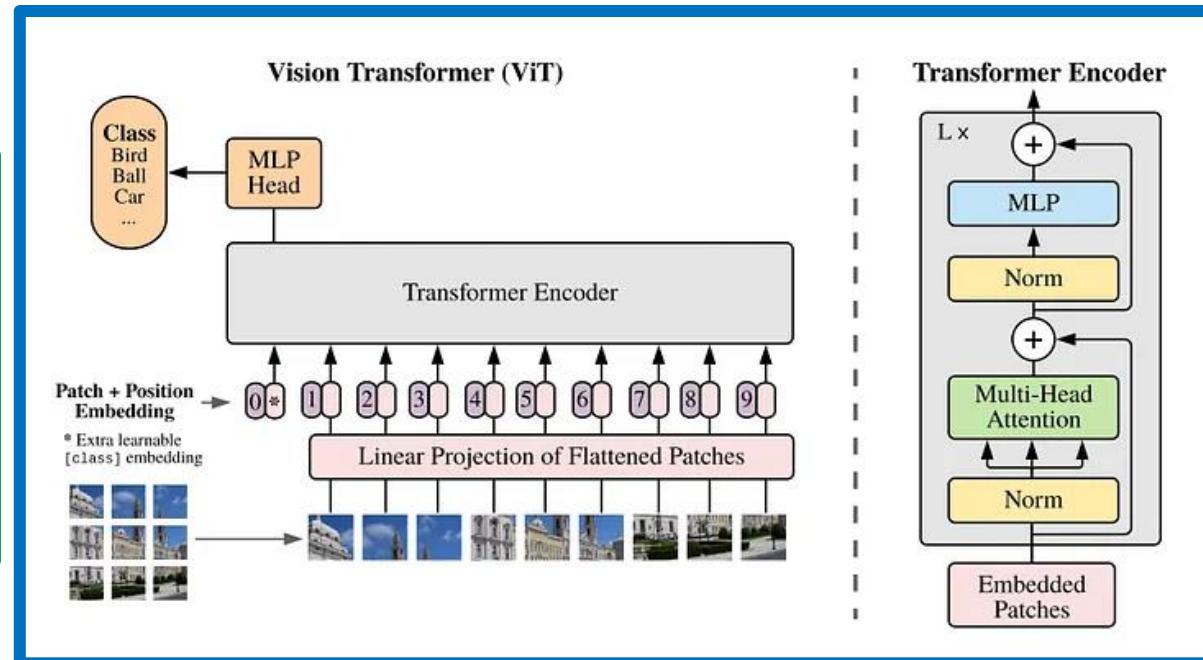
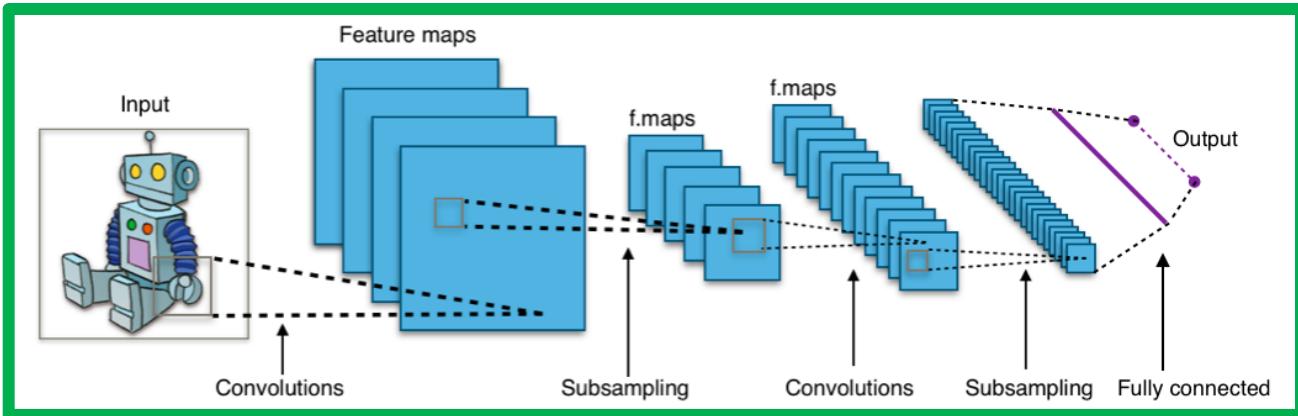
Keio University

Compute and Sensing Group (CSG)

<https://sites.google.com/keio.jp/keio-csg/>

# Background

- Modern edge ML workloads are diverse
    - CNN, Transformer, Hybrid that blends both
- Can analog CIM handle all workloads?



Wikipedia: Convolutional neural network

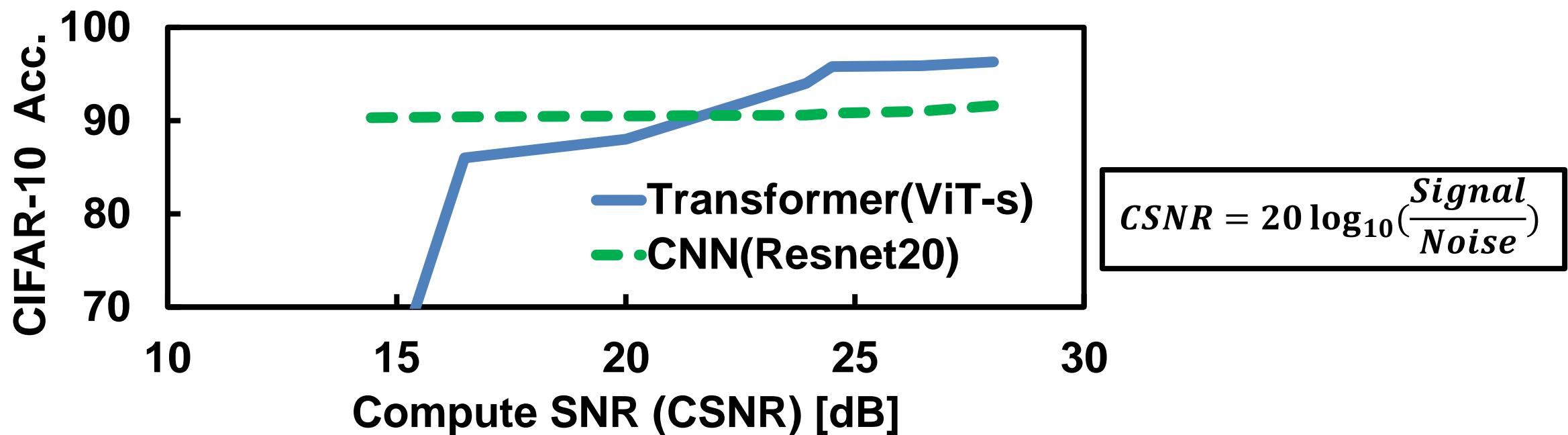
A. Dosovitskiy, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

# CNN/Transformer acceleration challenges

## ■ Unique computational requirements

- Transformer require high compute precision
- CNNs are fine with low precision

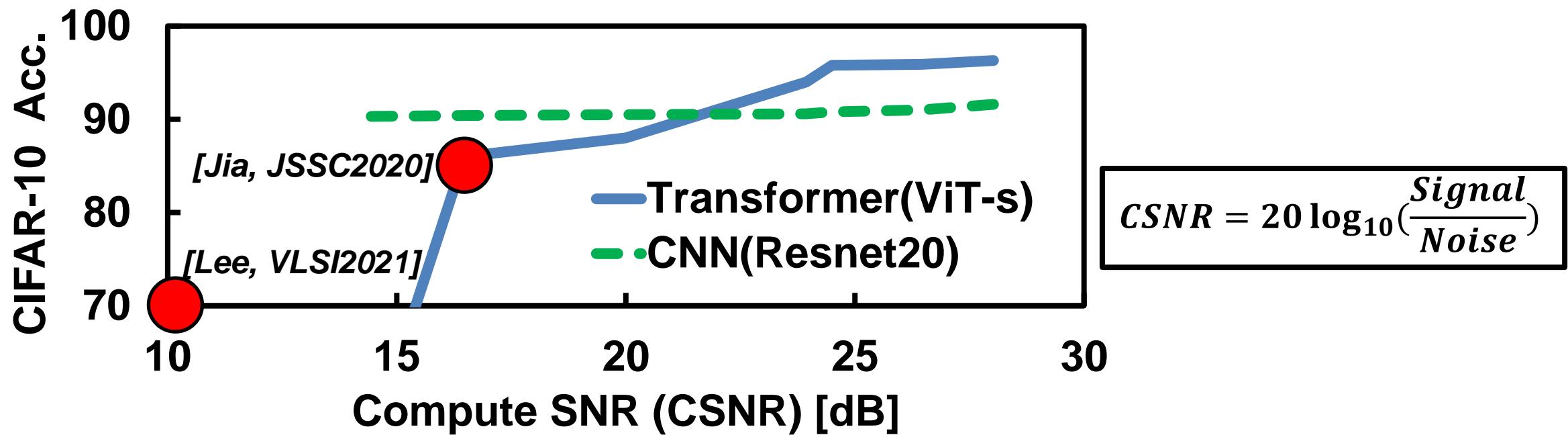
Denote precision as compute SNR (CSNR)



# CNN/Transformer acceleration challenges

## ■ ACIMs: Challenging to achieve high CSNR

- Conventional works target CNNs with relaxed CSNR
- Computational accuracy NOT a primary concern in ACIMs

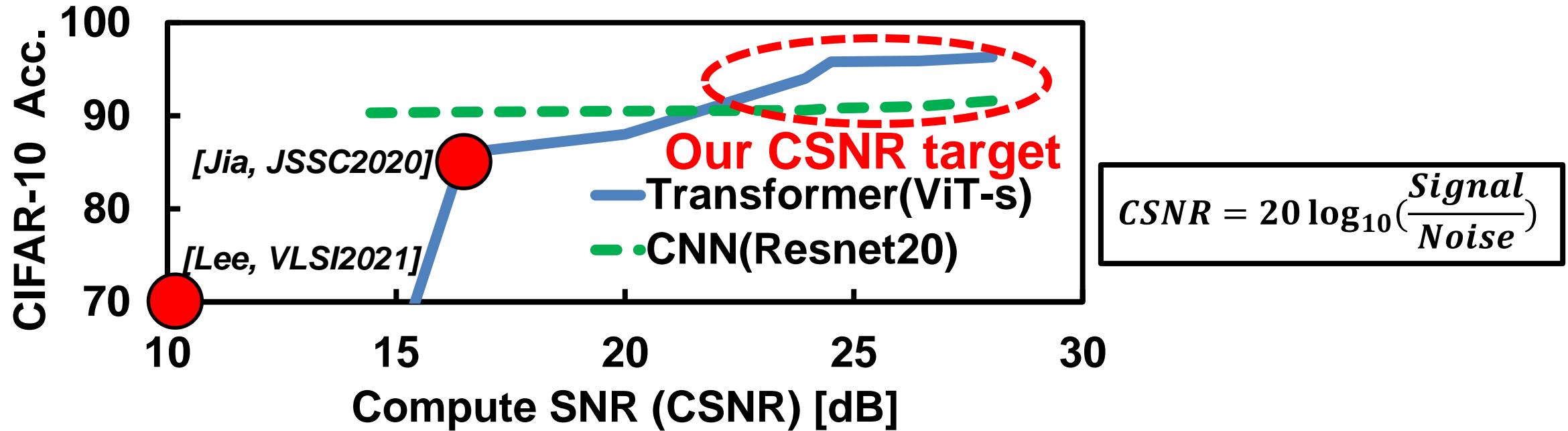


# CNN/Transformer acceleration challenges

## ■ ACIMs: Challenging to achieve high CSNR

- Conventional works target CNNs with relaxed CSNR
- Computational accuracy not a primary concern in ACIMs

## ■ DCIMs: Excel in CSNR, but require bulky adders

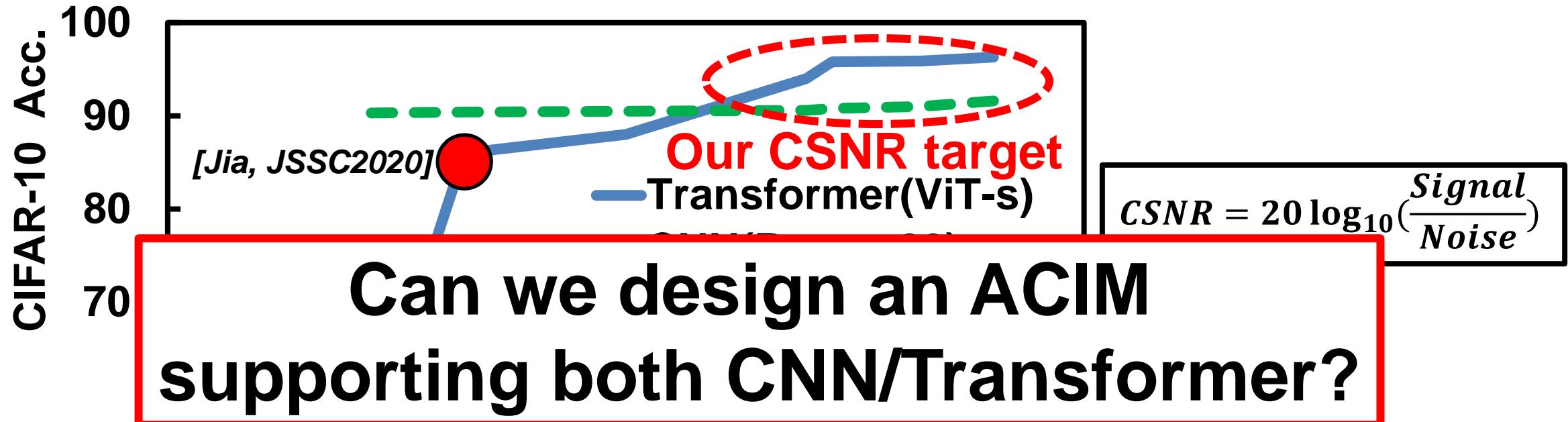


# CNN/Transformer acceleration challenges

## ■ ACIMs: Challenging to achieve high CSNR

- Conventional works target CNNs with relaxed CSNR
- Computational accuracy not a primary concern in ACIMs

## ■ DCIMs: Excel in CSNR, but require bulky adders



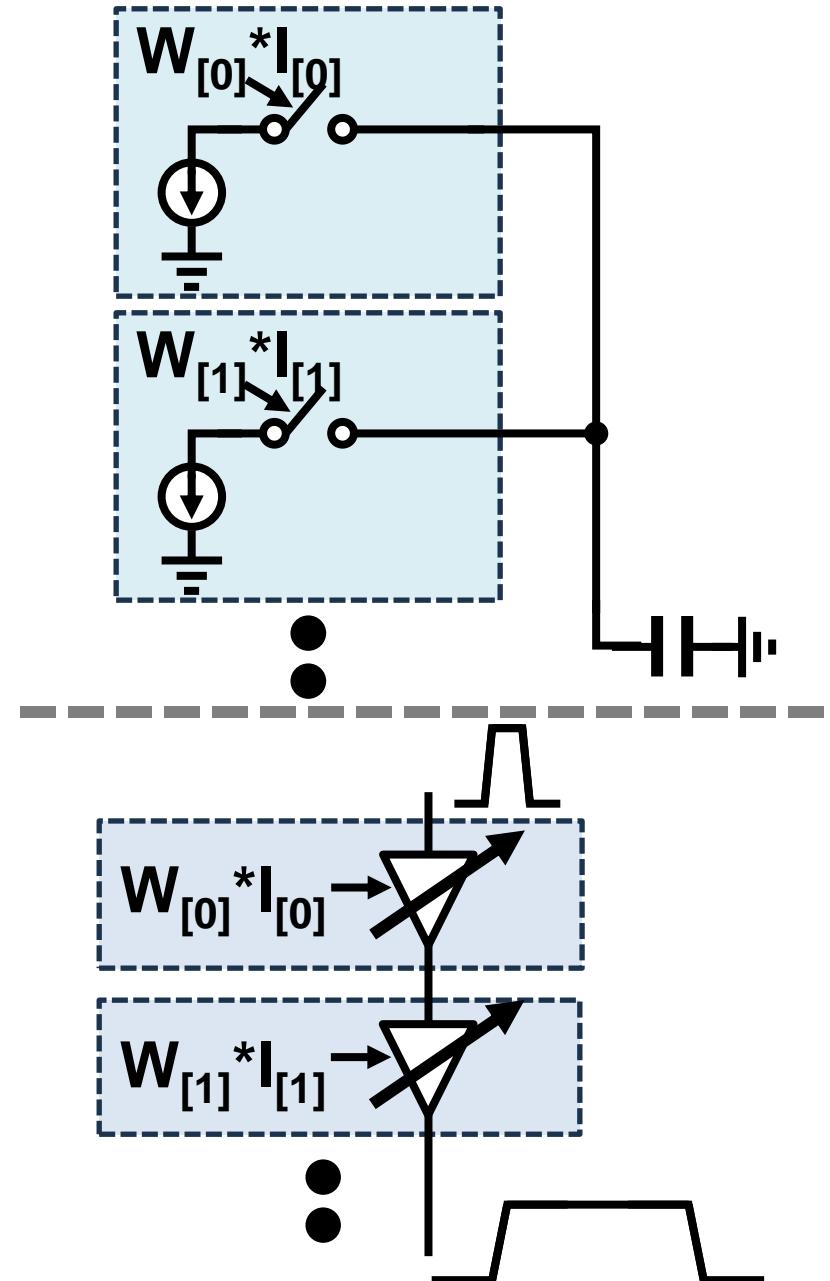
# Review of ACIM Architectures

## ■ Current-domain ACIMs

- 😊 Area, energy efficiency
- 😞 Transistor current inherently non-linear

## ■ Time-domain ACIMs

- 😊 Area, energy efficiency
- 😞 Delay prone to mismatch



# Review of ACIM Architectures

## ■ Current-domain ACIMs

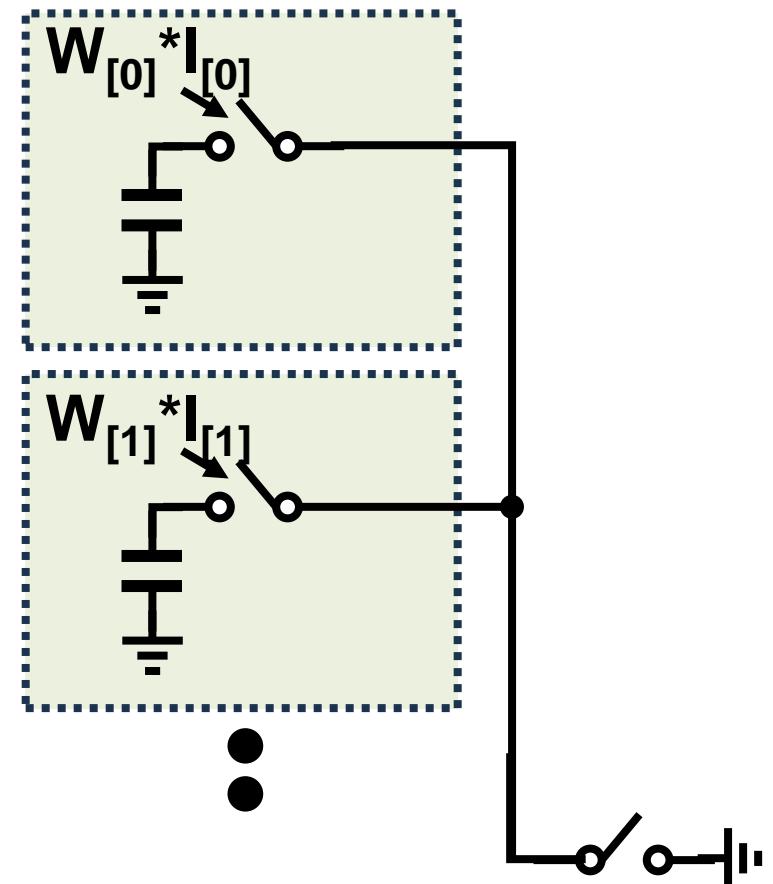
- 😊 Area, energy efficiency
- 😞 Transistor current inherently non-linear

## ■ Time-domain ACIMs

- 😊 Area, energy efficiency
- 😞 Delay prone to mismatch

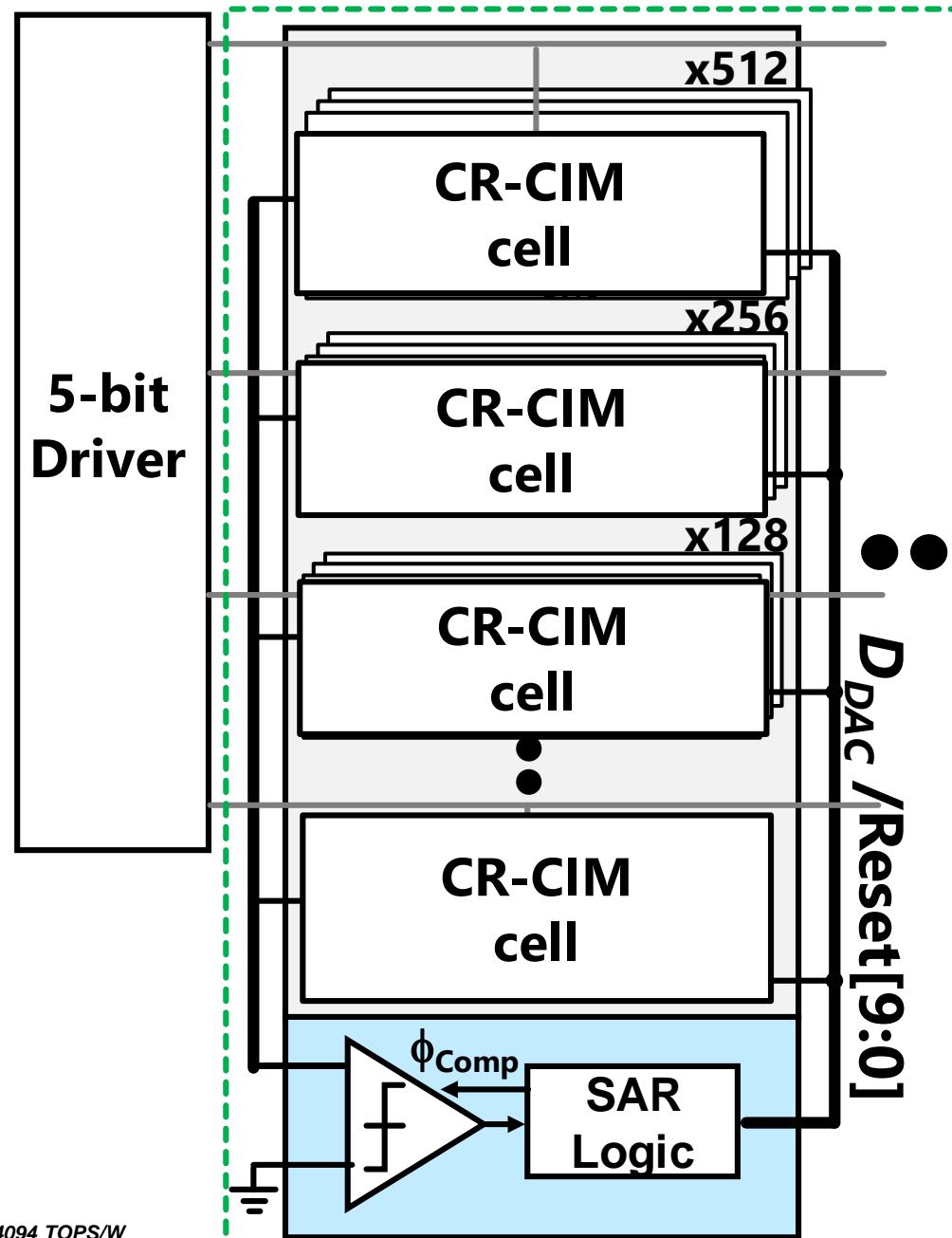
## ■ Charge-domain ACIMs

- 😊 High-linearity of MOM caps
- 😞 High ADC resolution → Worsened area efficiency
- 😞 Noise resilience → Worsened energy efficiency



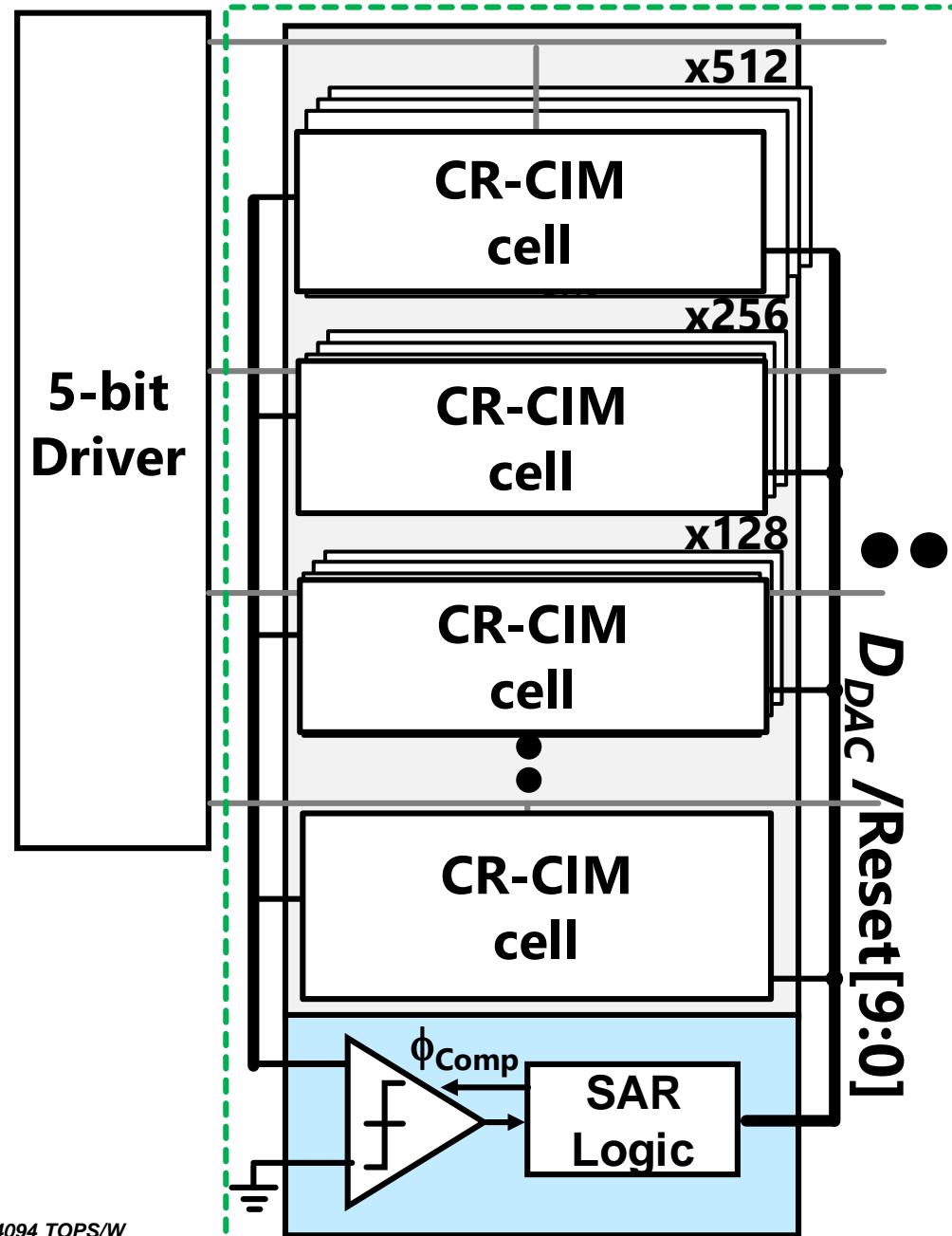
# Our CIM Macro Concept

- **Basic idea:**
  - Dynamically configure macro reflecting the processing DNN
- **High-CSNR Transformer mode:**
- **Low-CSNR CNN mode:**



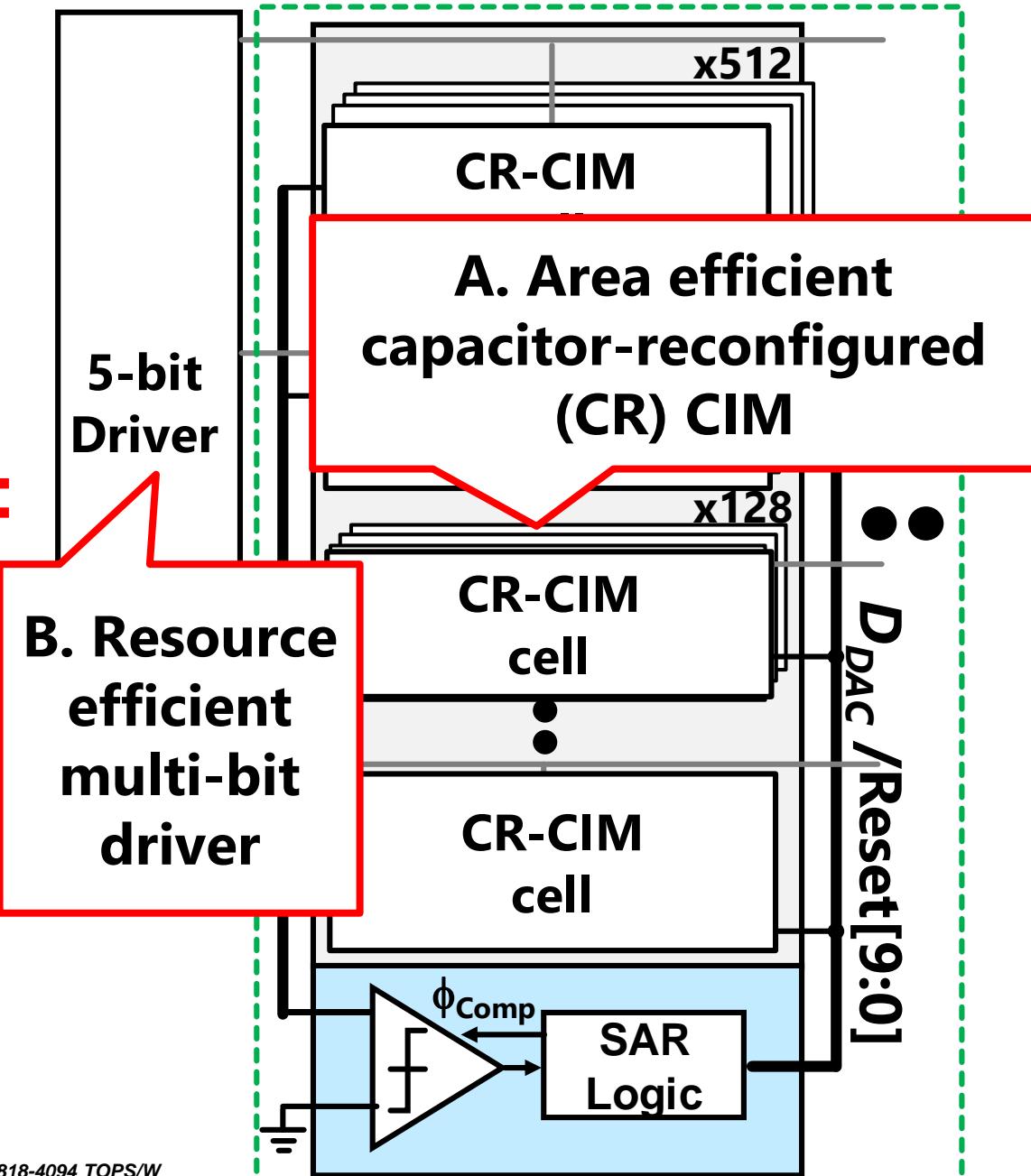
# Our CIM Macro Concept

- **Basic idea:**
  - Dynamically configure macro reflecting the processing DNN
- **High-CSNR Transformer mode:**
  - High-precision *bit-serial* CIM
  - *10-b ADC resolution and noise*
- **Low-CSNR CNN mode:**
  - Low-precision *bit-parallel* CIM
  - Achieve high-efficiency



# Our CIM Macro Concept

- **Basic idea:**
  - Dynamically configure macro reflecting the processing DNN
- **High-CSNR Transformer mode:**
  - High-precision *bit-serial* CIM
  - *10-b ADC resolution and noise*
- **Low-CSNR CNN mode:**
  - Low-precision *bit-parallel* CIM
  - Achieve high-efficiency

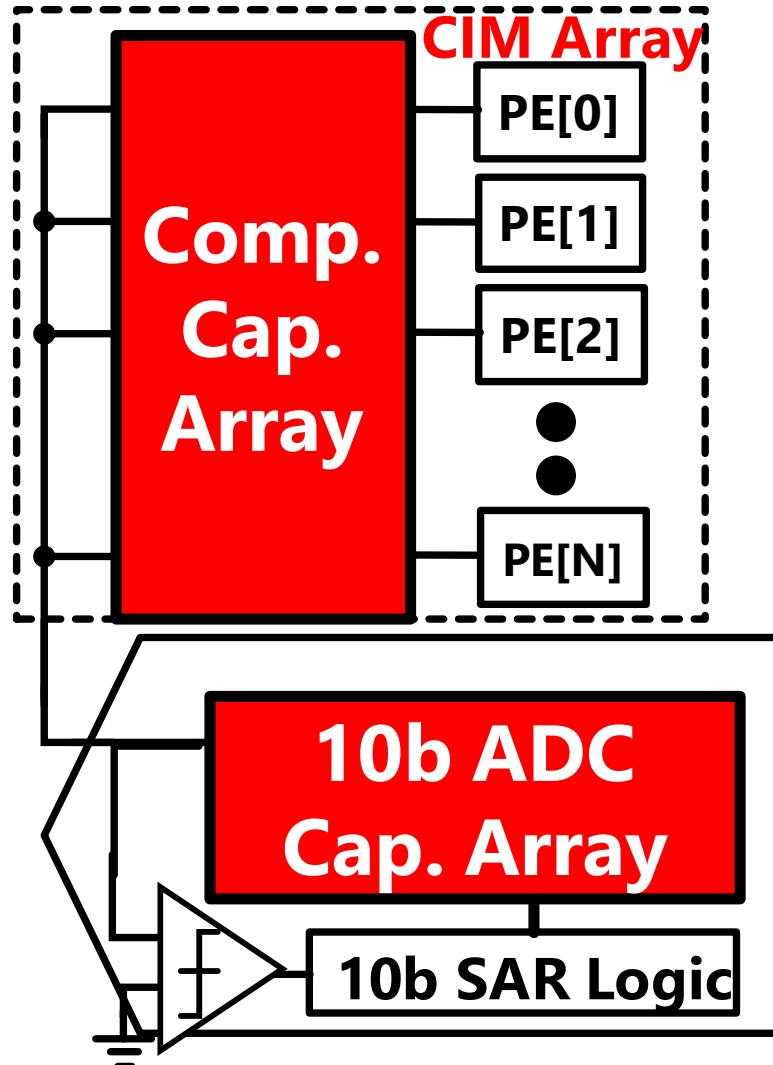


# Outline

- **Background**
- **CNN/Transformer unified acceleration challenges**
- **Our CIM Macro Concept**
  - Capacitor-Reconfigured CIM (CR-CIM)
  - Resource-efficient Multi-bit Driver
- **Measurement results**
- **Conclusion**

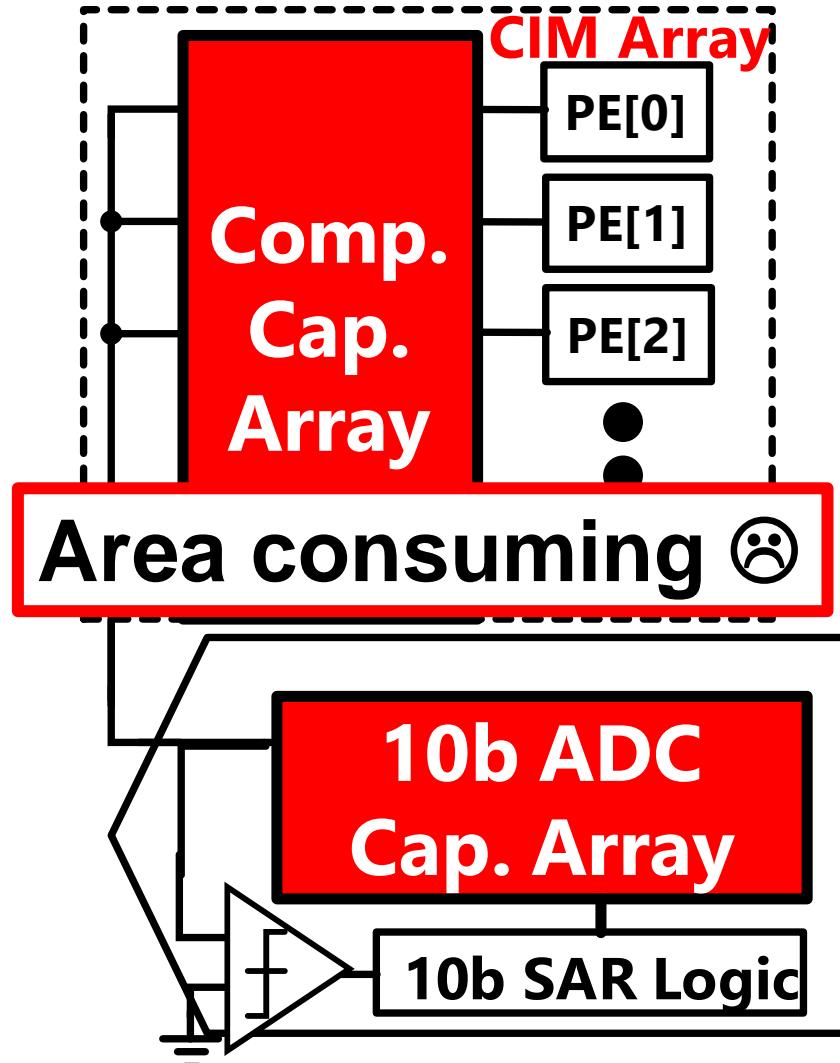
# CR-CIM concept

## Conventional



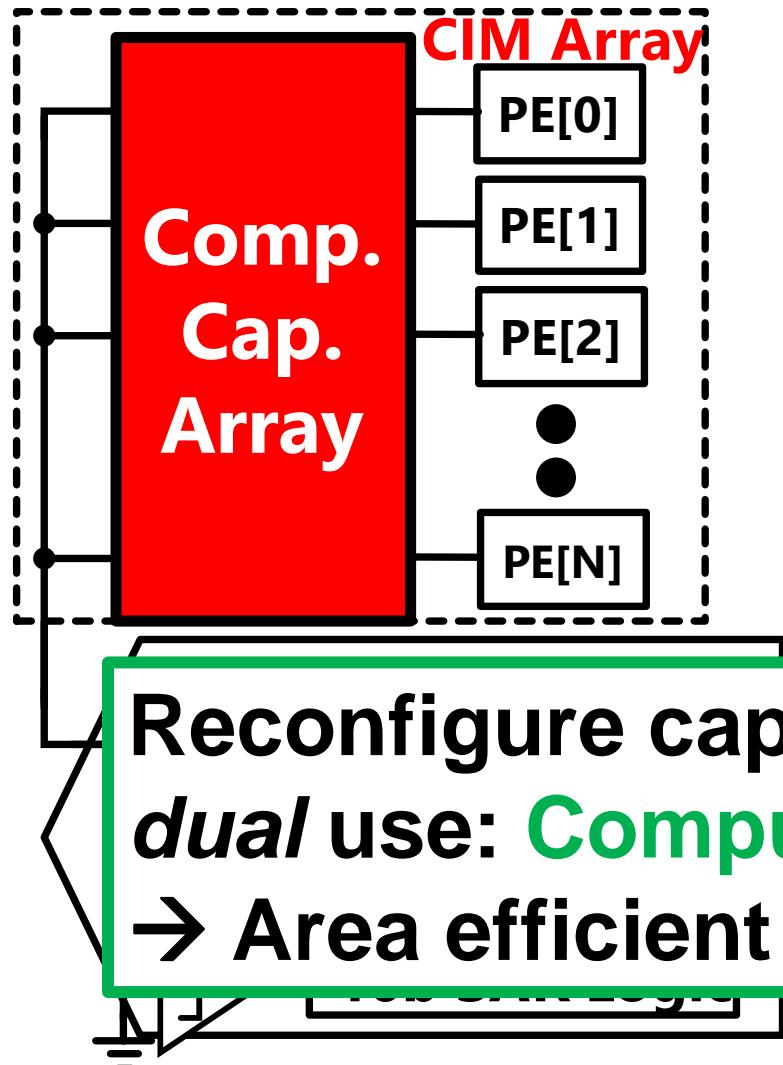
# CR-CIM concept

## Conventional



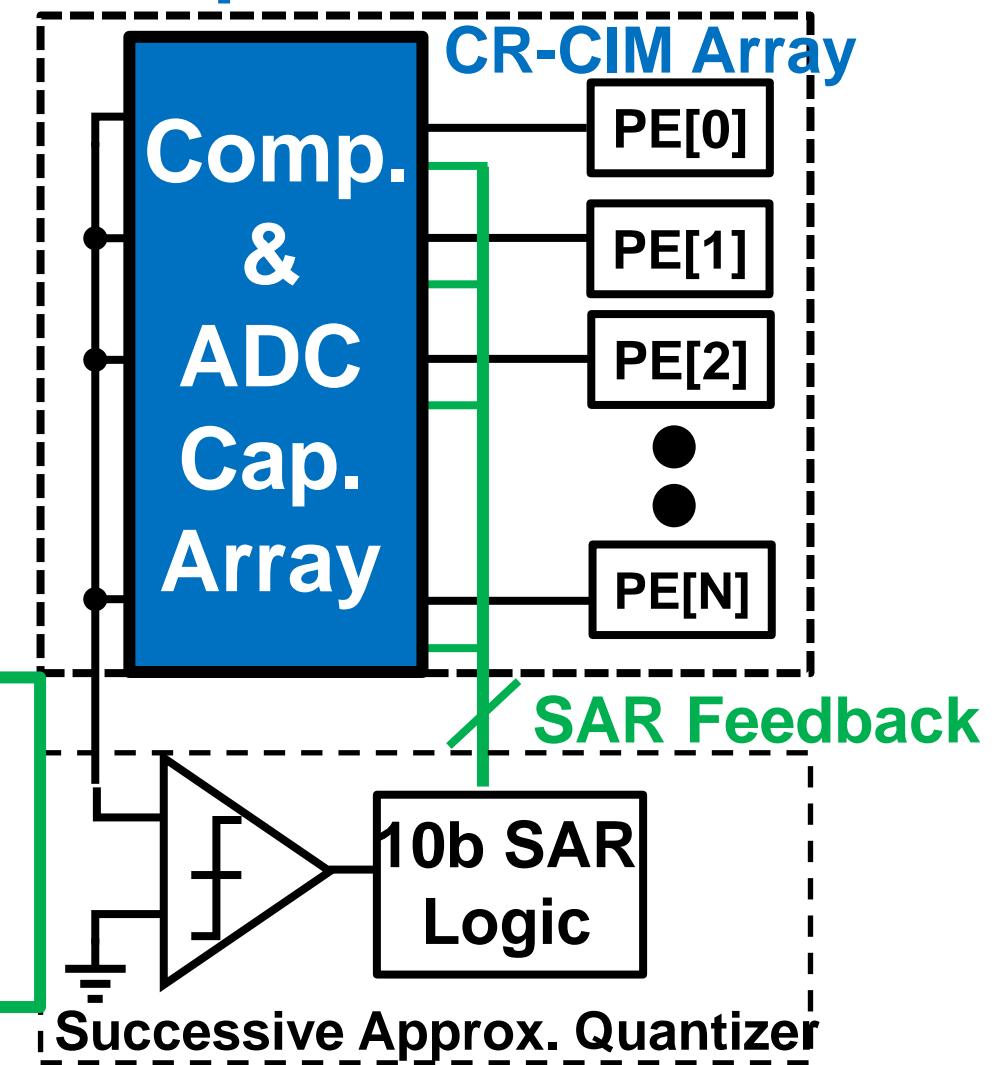
# CR-CIM concept

## Conventional



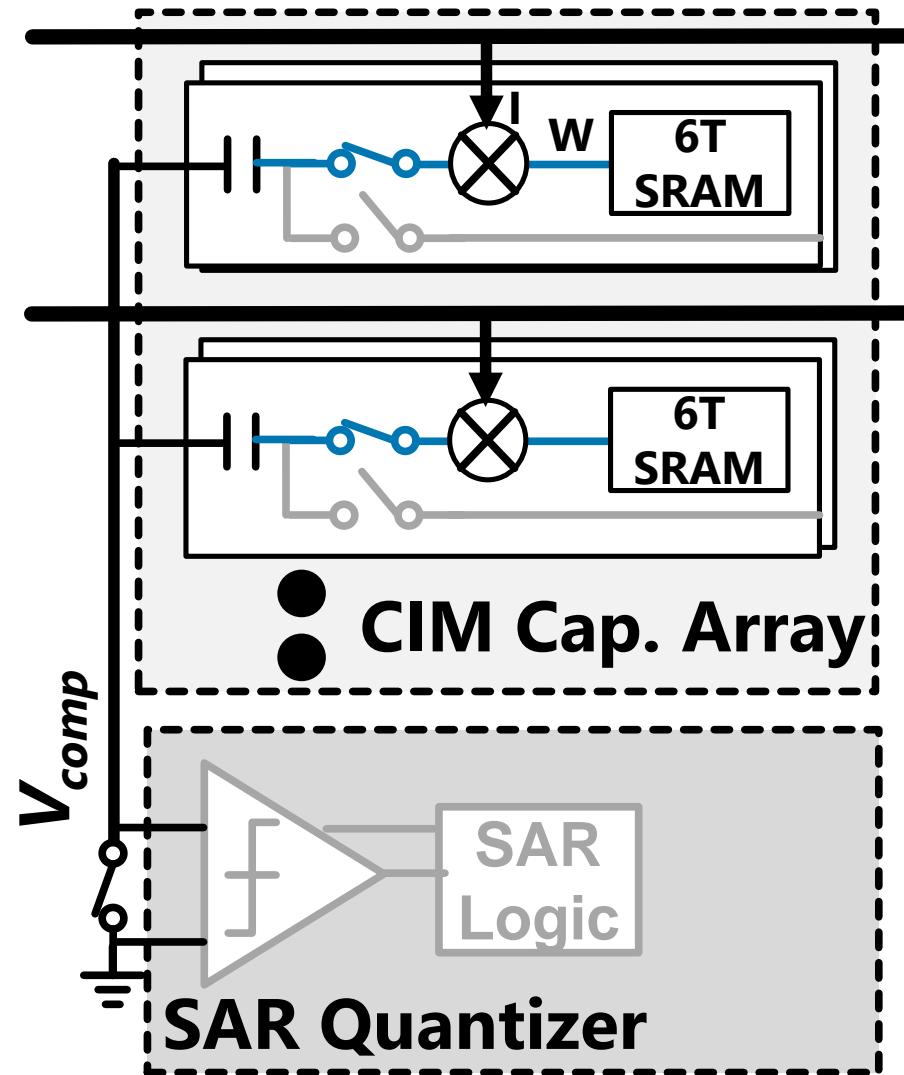
Reconfigure cap array for  
*dual* use: **Computation/ADC**  
→ Area efficient 10-b ADC!

## Proposed CR-CIM



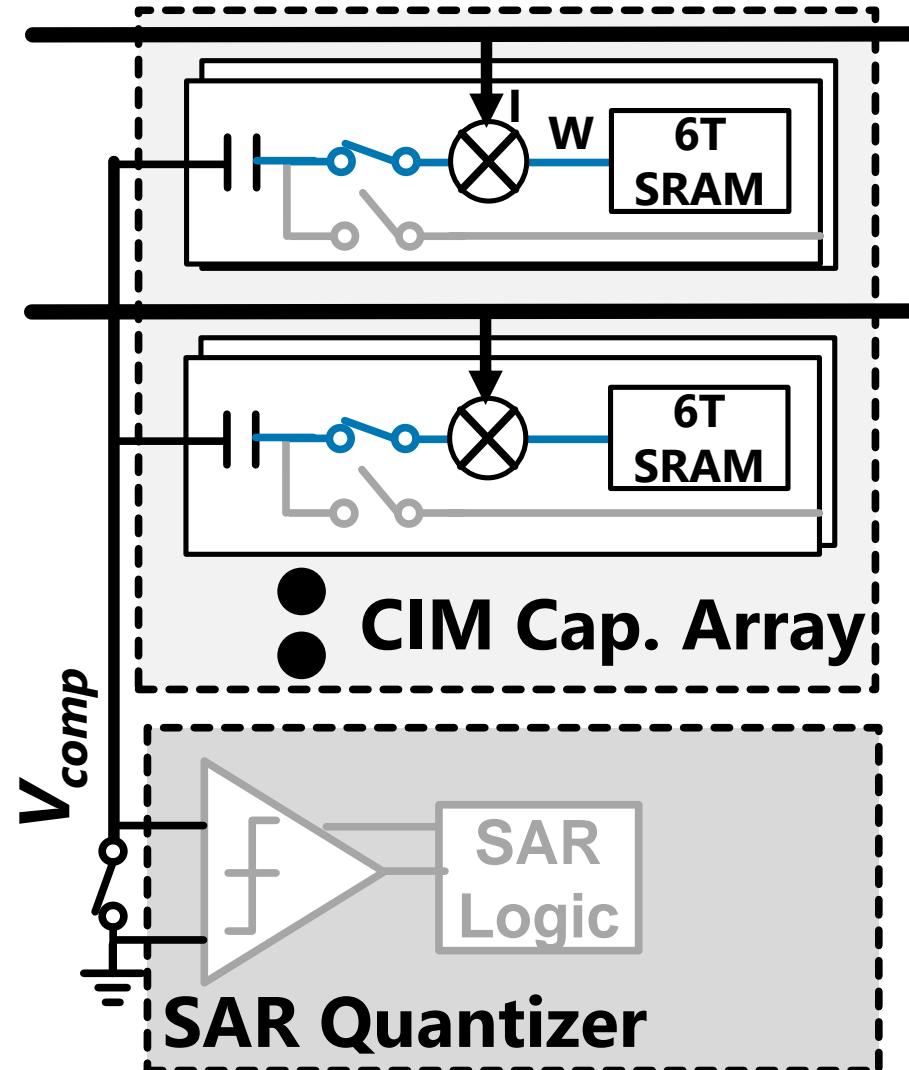
# CR-CIM Array Implementation

## Computation

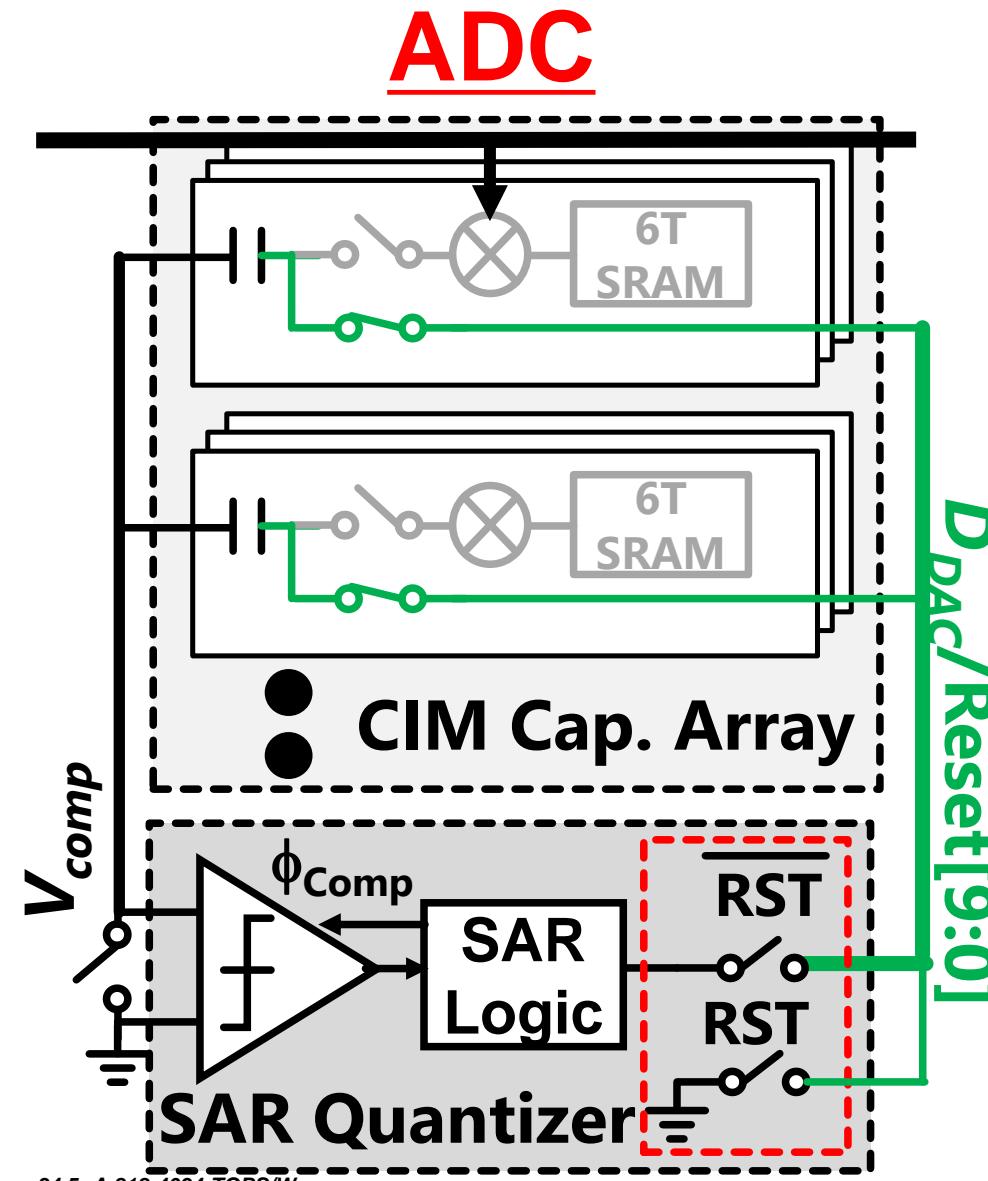


# CR-CIM Array Implementation

## Computation

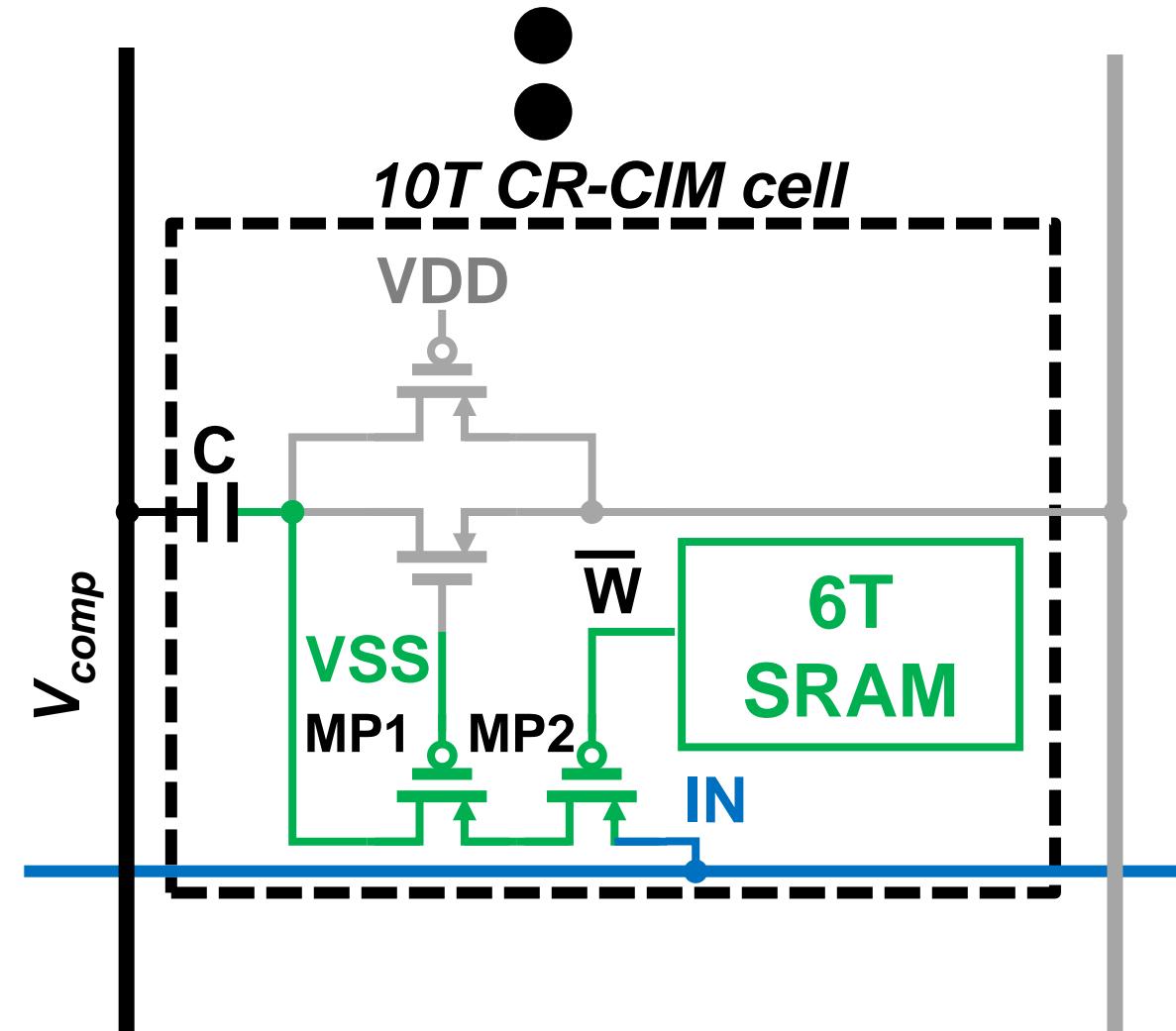


## ADC

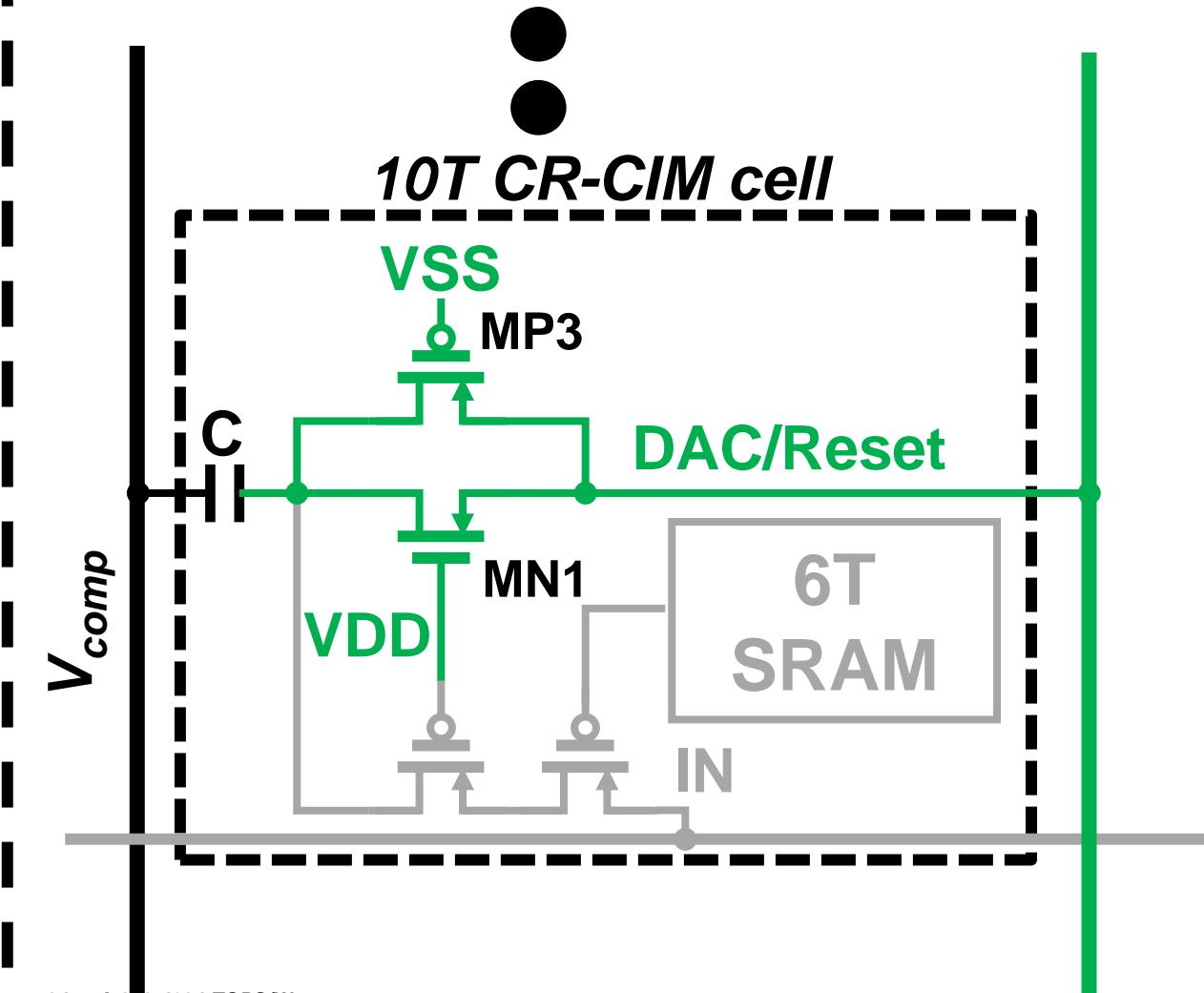


# CR-CIM Transistor Implementation

## Computation



## ADC&Reset

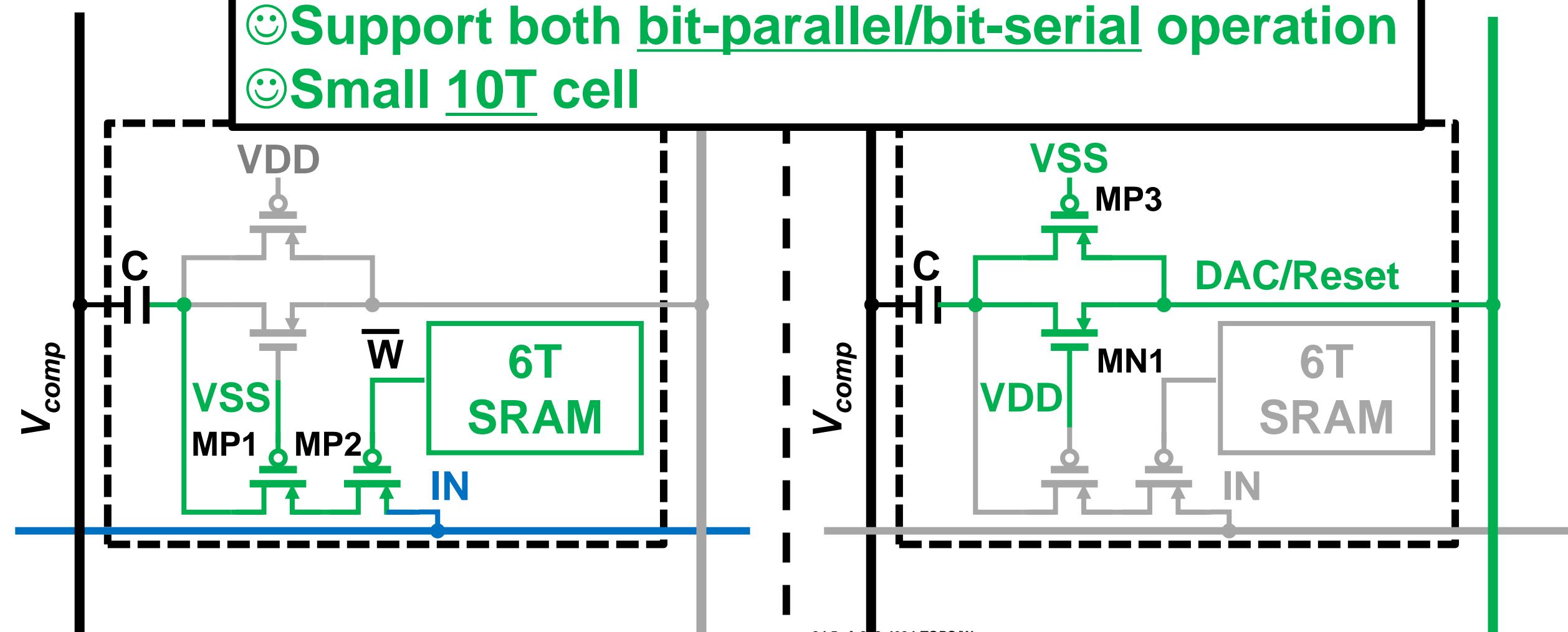


# CR-CIM Transistor Implementation

## Computation

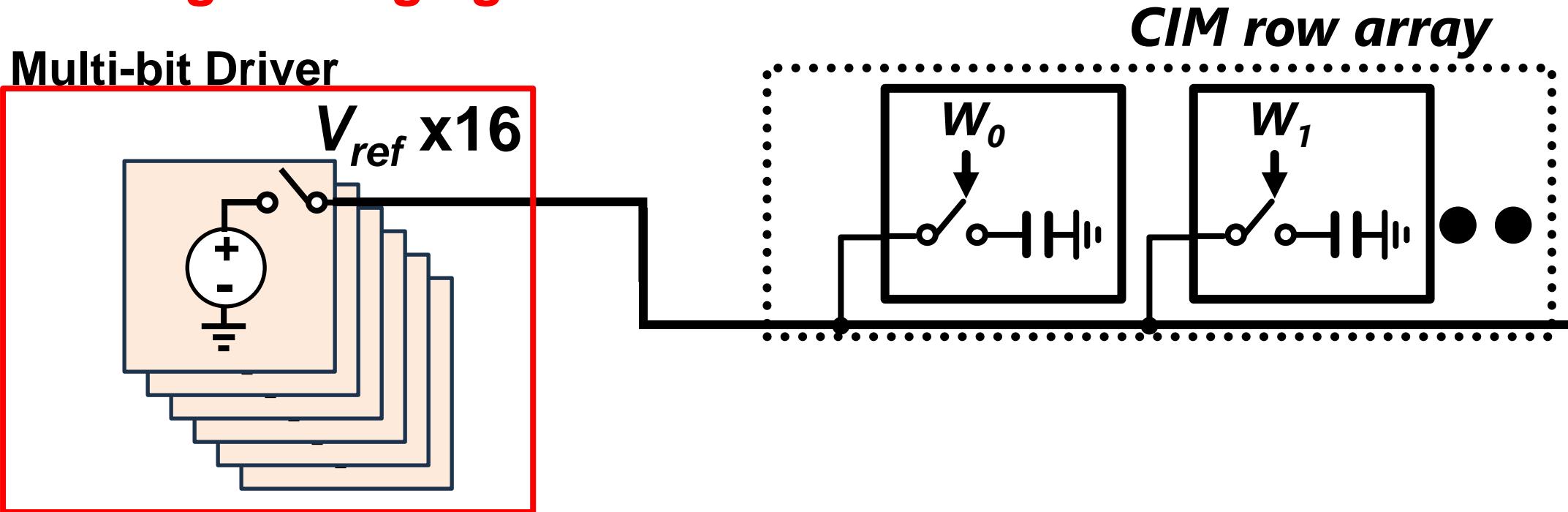
## ADC&Reset

- ☺ Support both bit-parallel/bit-serial operation
- ☺ Small 10T cell



# Resource Efficient Multi-bit Driver

- Bit-parallel operation greatly improves the efficiency under CNN mode
  - [Lee, VLSI2021] uses 16 off-chip generated ref. voltage  
→**Large voltage generation overhead**

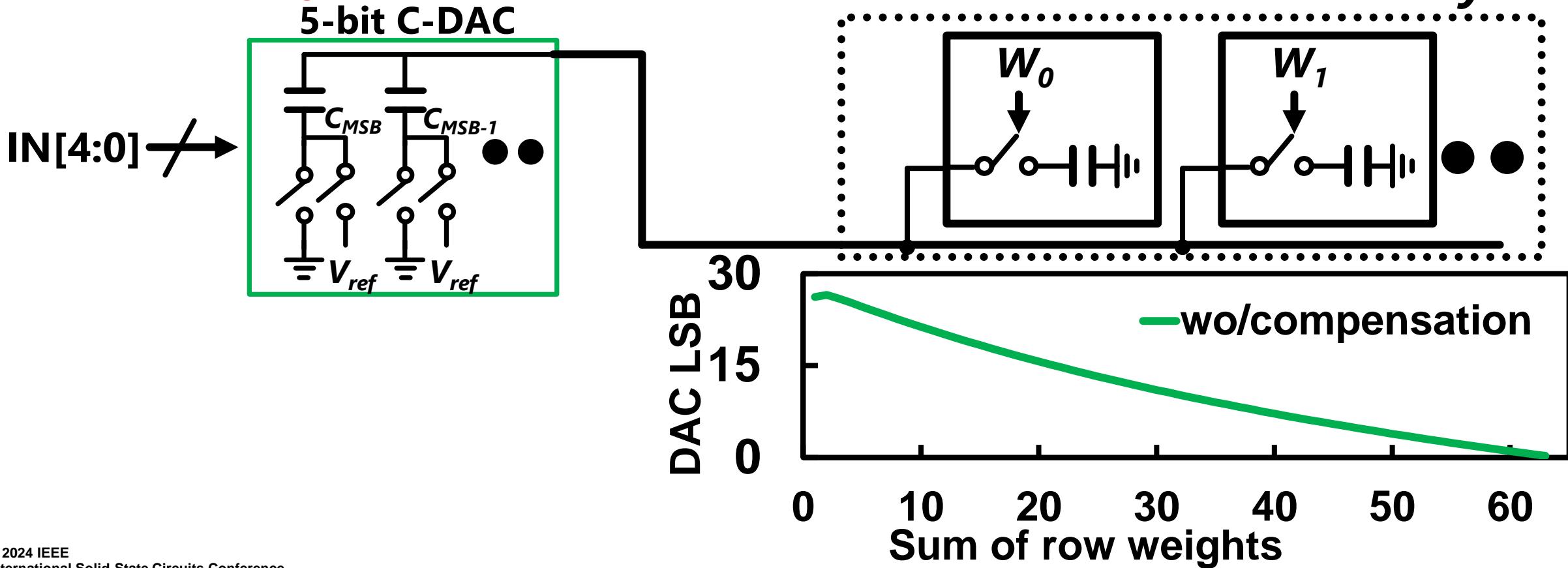


# Resource Efficient Multi-bit Driver

- Capacitive dividing to generate ref. voltage

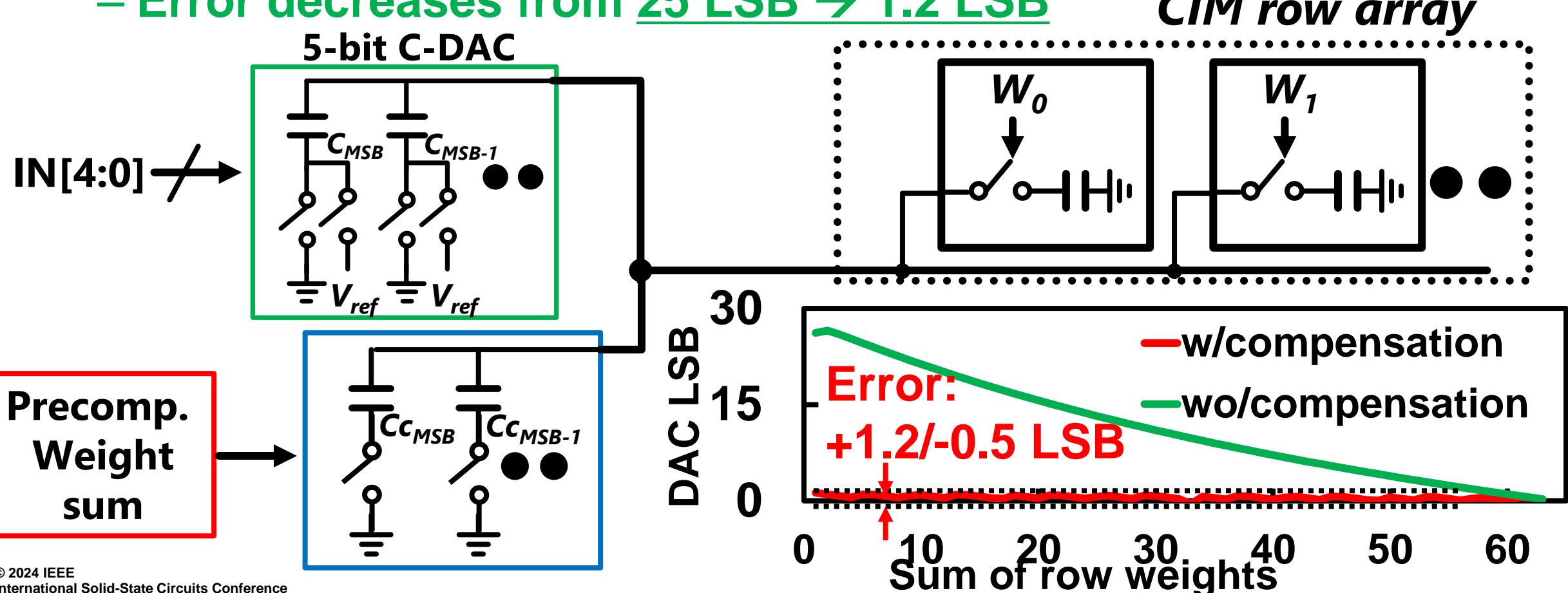
- ☺ Only 1 ref. voltage required

- ☹ Voltage errors due to varied capacitive load



# Resource Efficient Multi-bit Driver

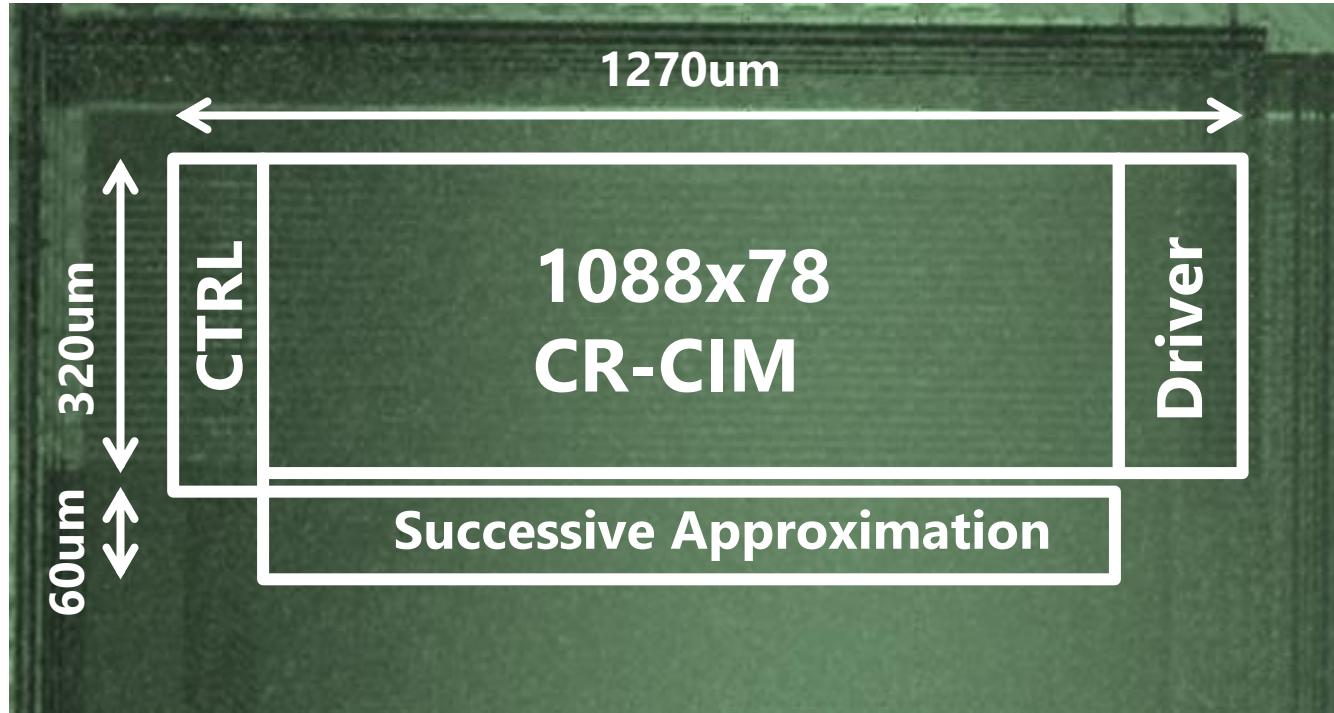
- Capacitive dividing to generate ref. voltage
  - ☺ Employ 4-b load-compensation C-DAC to cancel variation
  - Error decreases from 25 LSB → 1.2 LSB



# Outline

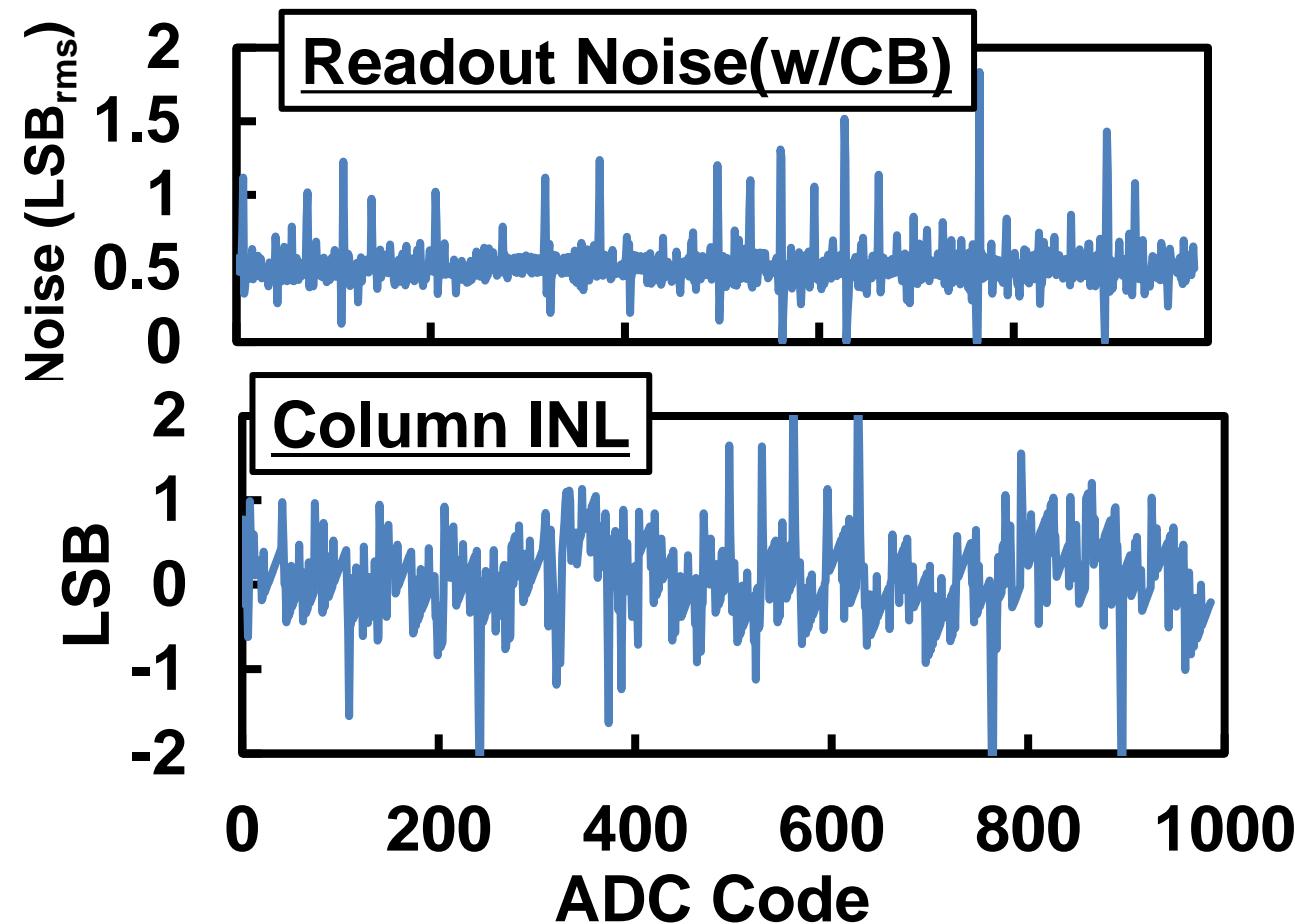
- **Background**
- **CNN/Transformer unified acceleration challenges**
- **Our CIM Macro Concept**
  - Capacitor-Reconfigured CIM
  - Resource-efficient Multi-bit Driver
- **Measurement results**
- **Conclusion**

# Prototype chip in 65nm CMOS



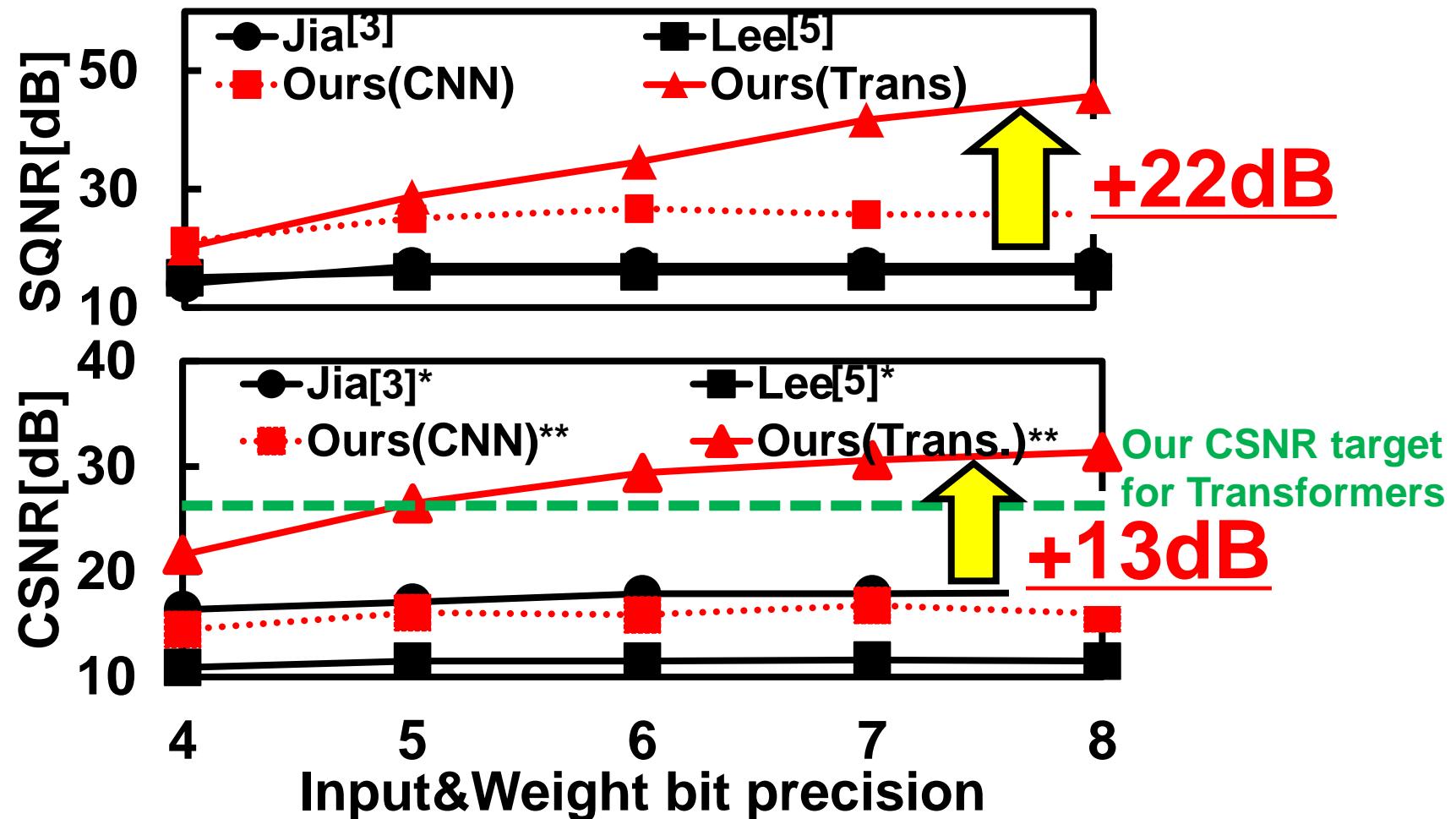
# CIM column characteristics

All measurements@ $V_{DD}=0.6V$  & 10-bit ADC



# CIM column characteristics

All measurements@ $V_{DD}=0.6V$  & 10-bit ADC



# Comparison

	This work	[3] JSSC 2020	[4] ISSCC 2023	[5] VLSI 2021
CIM type	Charge	Charge	Charge	Charge
Process	65nm	65nm	12nm	28nm
Bit Precision	4-8b	1-8b	1-8b	1-5b
Application	CNN	CNN	CNN	CNN
Peak TOPS Normalized to 1-b	6	2.1	6.4	6.1
Peak TOPS/W Norm. to 65nm**	4094	400	837	2496
ADC bit	8	8	8	8
SQNR [dB]	26.7	22	N.A.	17.5
CSNR [dB]	16.8	17	N.A.	10.5
CIFAR-10 Accuracy	91.7	92.4	N.A.	91.1

# Comparison

	<b>This work</b>	[3] JSSC 2020	[4] ISSCC 2023	[5] VLSI 2021
CIM type	<b>Charge</b>	Charge	Charge	Charge
Process	<b>65nm</b>	65nm	12nm	28nm
Bit Precision	<b>4-8b</b>	1-8b	1-8b	1-5b
Application	<b>CNN</b>	<b>Trans- former</b>	CNN	CNN
Peak TOPS Normalized to 1-b	<b>6</b>	<b>1.2</b>	2.1	6.4
Peak TOPS/W Norm. to 65nm**	<b>4094</b>	<b>818</b>	400	827
ADC bit	<b>8</b>	<b>10</b>		
SQNR [dB]	<b>26.7</b>	<b>45.3</b>		
CSNR [dB]	<b>16.8</b>	<b>31.3</b>	17	N.A.
CIFAR-10 Accuracy	<b>91.7</b>	<b>95.8</b>	92.4	N.A.
				<b>91.1</b>

**First ACIM to achieve  
unified operation  
of CNN and Transformers**

# Conclusion

- First realization of an ACIM capable of efficient Transformer & CNN inference
  - Transformer mode: Accurate bit-serial operation for enhanced CSNR
  - CNN mode: Efficient bit-parallel operation
- Key circuit innovations:
  - CR-CIM to achieve area-efficient 10-b ADC
  - Resource-efficient multi-bit driver

# **Thank you for your attention!**

**The author would like to thank..**

- Jim Chen and Kaoru Yamashita for discussions.**
- JST CREST, JSPS KAKENHI for the support.**



Please Scan to Rate  
This Paper



# A 28nm 72.12TFLOPS/W Hybrid-Domain Outer-Product Based Floating-Point SRAM Computing-in-Memory Macro with Logarithm Bit-Width Residual ADC

**Yiyang Yuan<sup>1,2</sup>, Yiming Yang<sup>3</sup>, Xinghua Wang<sup>3</sup>, Xiaoran Li<sup>3</sup>, Cailian Ma<sup>1,2</sup>, Qirui Chen<sup>3</sup>, Meini Tang<sup>3</sup>, Xi Wei<sup>3</sup>, Zhixian Hou<sup>3</sup>, Jialiang Zhu<sup>1,2</sup>, Hao Wu<sup>1,2</sup>, Qirui Ren<sup>1,2</sup>, Guozhong Xing<sup>1</sup>, Pui-In Mak<sup>4</sup>, Feng Zhang<sup>1</sup>**

<sup>1</sup>Institute of Microelectronics of the Chinese Academy of Sciences, Beijing, China

<sup>2</sup>University of Chinese Academy of Sciences, Beijing, China

<sup>3</sup>Beijing Institute of Technology, Beijing, China

<sup>4</sup>University of Macau, Macau, China



中国科学院微电子研究所  
INSTITUTE OF MICROELECTRONICS OF THE CHINESE ACADEMY OF SCIENCES



中国科学院大学  
University of Chinese Academy of Sciences



北京理工大学  
BEIJING INSTITUTE OF TECHNOLOGY



澳门大学  
UNIVERSIDADE DE MACAU  
UNIVERSITY OF MACAU

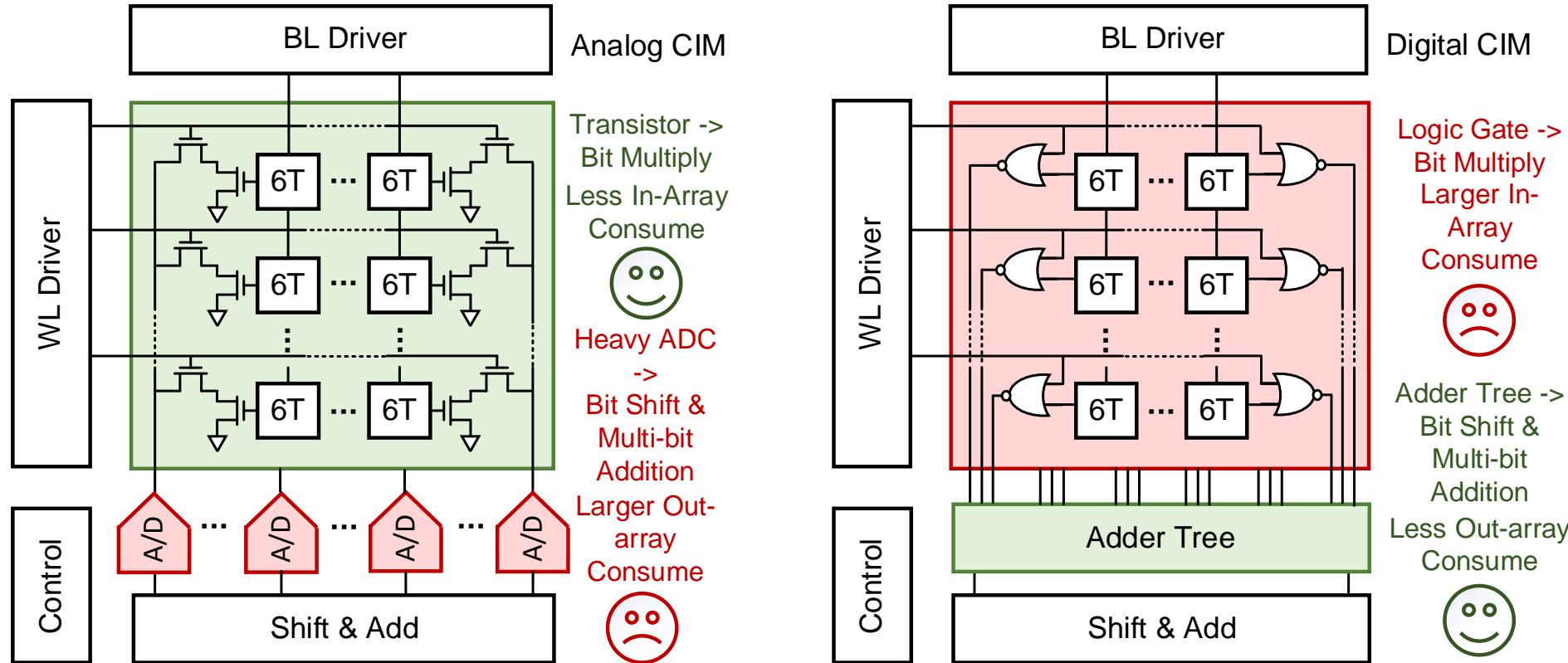
# Outline

- **Challenges of SRAM CIM and Motivation**
- **Proposed Hybrid-domain SRAM-CIM Macro**
  - Macro Overall Architecture
  - Analog In-Array & Digital Out-Array Hybrid-Domain CIM
  - Logarithm Bit-Width Residual ADC Scheme
  - Outer-product based FP/INT CIM Block Architecture
- **Measurement Result**
- **Conclusion**

# Outline

- **Challenges of SRAM CIM and Motivation**
- **Proposed Hybrid-domain SRAM-CIM Macro**
  - Macro Overall Architecture
  - Analog In-Array & Digital Out-Array Hybrid-Domain CIM
  - Logarithm Bit-Width Residual ADC Scheme
  - Outer-product based FP/INT CIM Block Architecture
- **Measurement Result**
- **Conclusion**

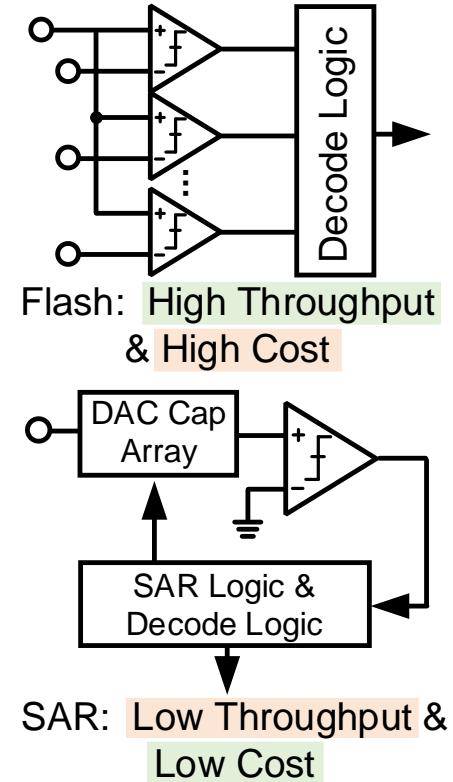
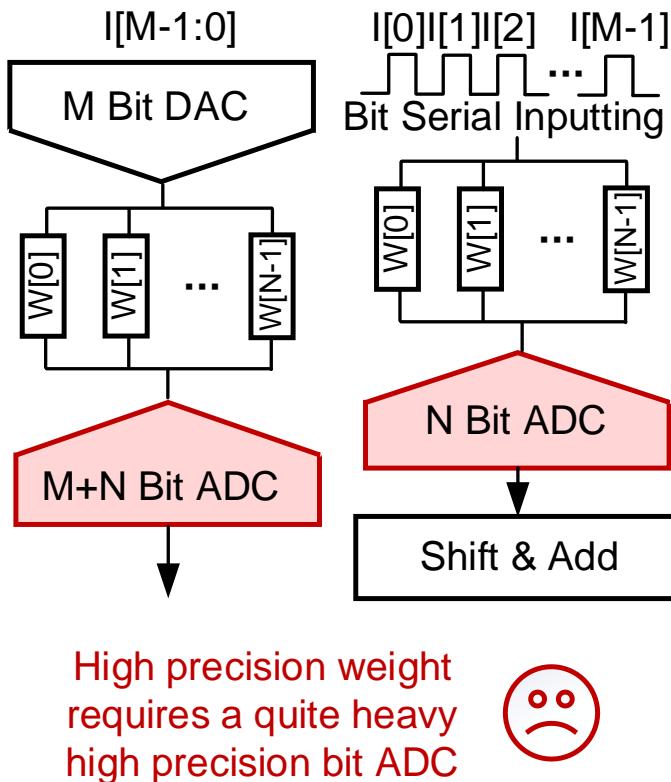
# Challenges of SRAM CIM (1/3)



## ■ How to combine the Advantages of DCIM and ACIM

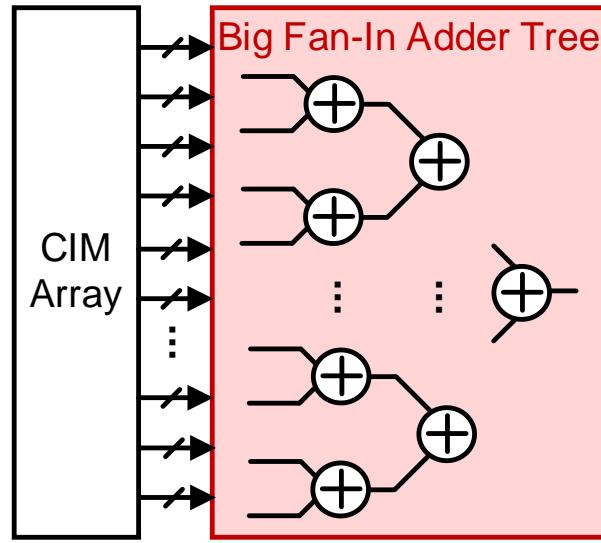
- Analog CIM: Low bit multiply cost but high multi-bit accumulate cost
- Digital CIM: High bit multiply cost but low multi-bit accumulate cost

# Challenges of SRAM CIM (2/3)



- **Trade-off of ADC Precision & Throughput & Overhead**
  - Higher bit precision increases ADC burden
  - Flash ADC: High throughput but high cost,
  - SAR ADC: Low cost but low throughput

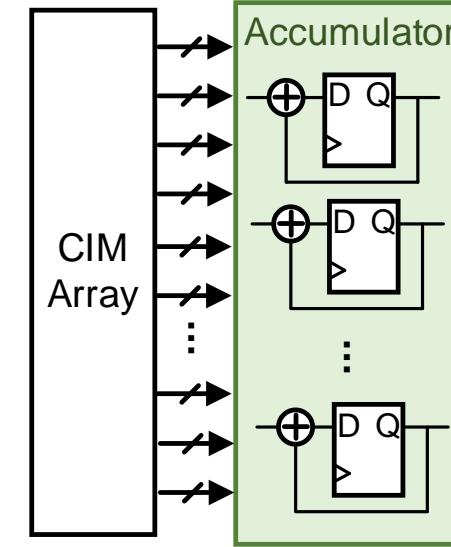
# Challenges of SRAM CIM (3/3)



Element-wise summation

$$c_{i,j} = \sum_{k=1}^n a_{i,k} b_{k,j}$$

Inner Product Based CIM  
With higher propagation delay



Vector-wise accumulation

$$c_i = \sum_{j=1}^n b_{j,i} a_j$$

Outer Product Based CIM  
Much less propagation delay



- **Inner-Product Based Large Fan-In Adder Tree Burden**
  - The transmission delay increases rapidly as the fan-in increases
  - Multi-level adder tree introduce higher transmission delay

# Our Motivation

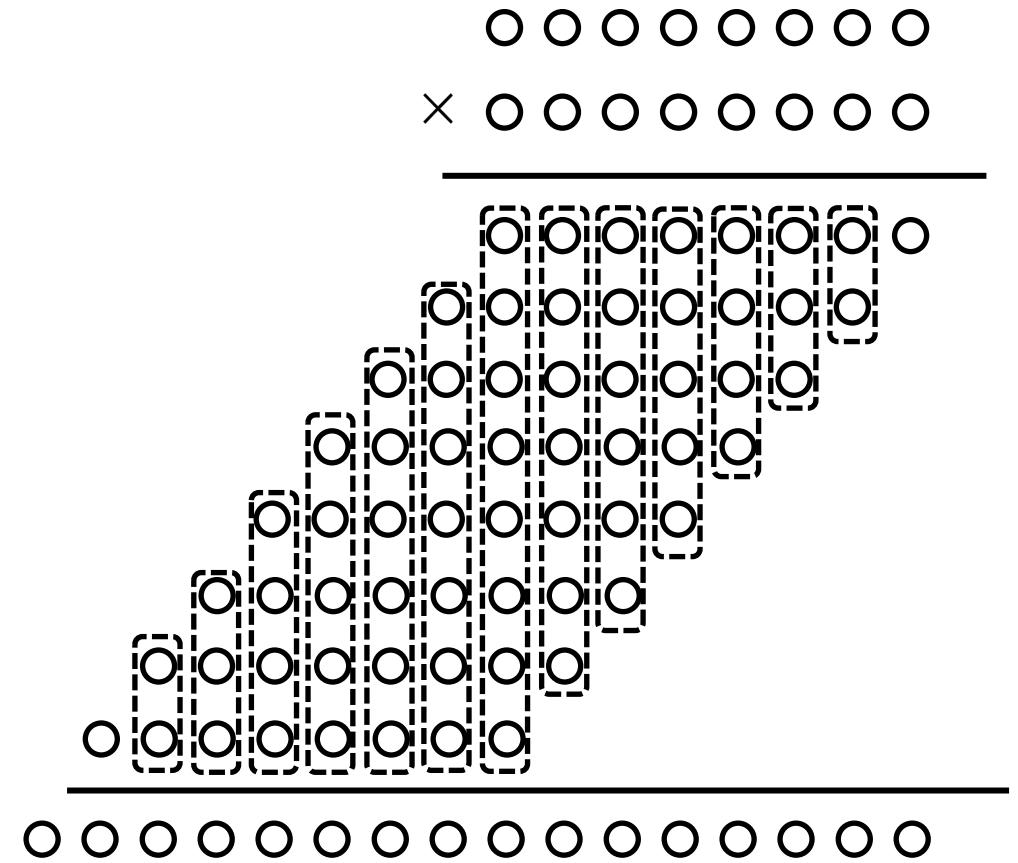
$$IN[N-1:0] \times W[N-1:0]$$

$$= \left( \sum_{i=0}^{N-1} IN[i] \times 2^i \right) \times \left( \sum_{i=0}^{N-1} W[i] \times 2^i \right)$$

$$= \sum_{i=0}^{N-1} \left( W[i] \times \left( \sum_{j=0}^{N-1} IN[j] \times 2^j \right) \times 2^i \right) \quad (1)$$

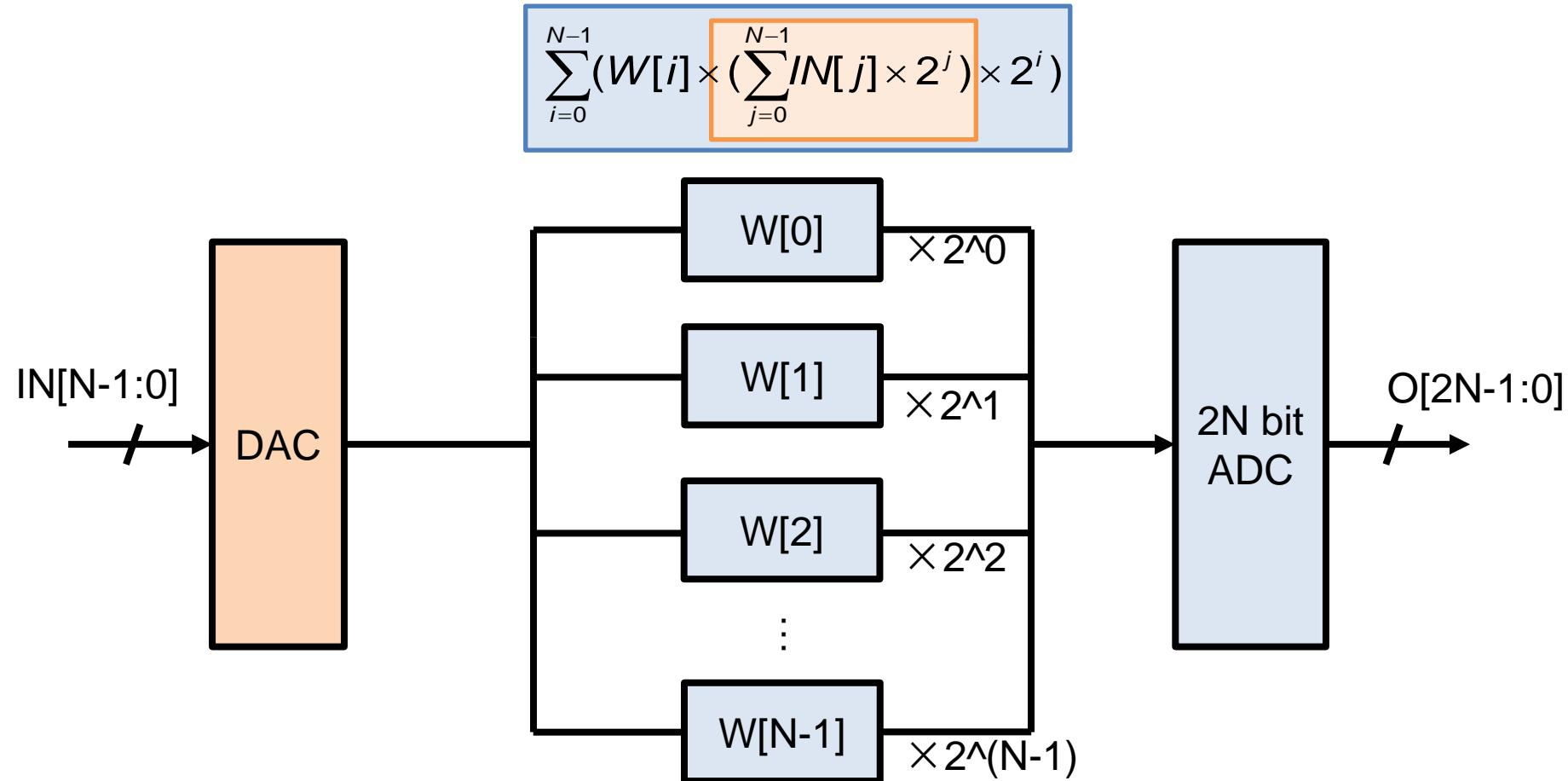
$$= \sum_{i=0}^{N-1} \left( IN[i] \times \left( \sum_{j=0}^{N-1} W[j] \times 2^j \right) \times 2^i \right) \quad (2)$$

$$= \sum_{i=0}^{2N-2} \left( \sum_{j=\max(0, i-(N-1))}^{\min(i, N-1)} IN[j] \times W[i-j] \right) \times 2^i \quad (3)$$



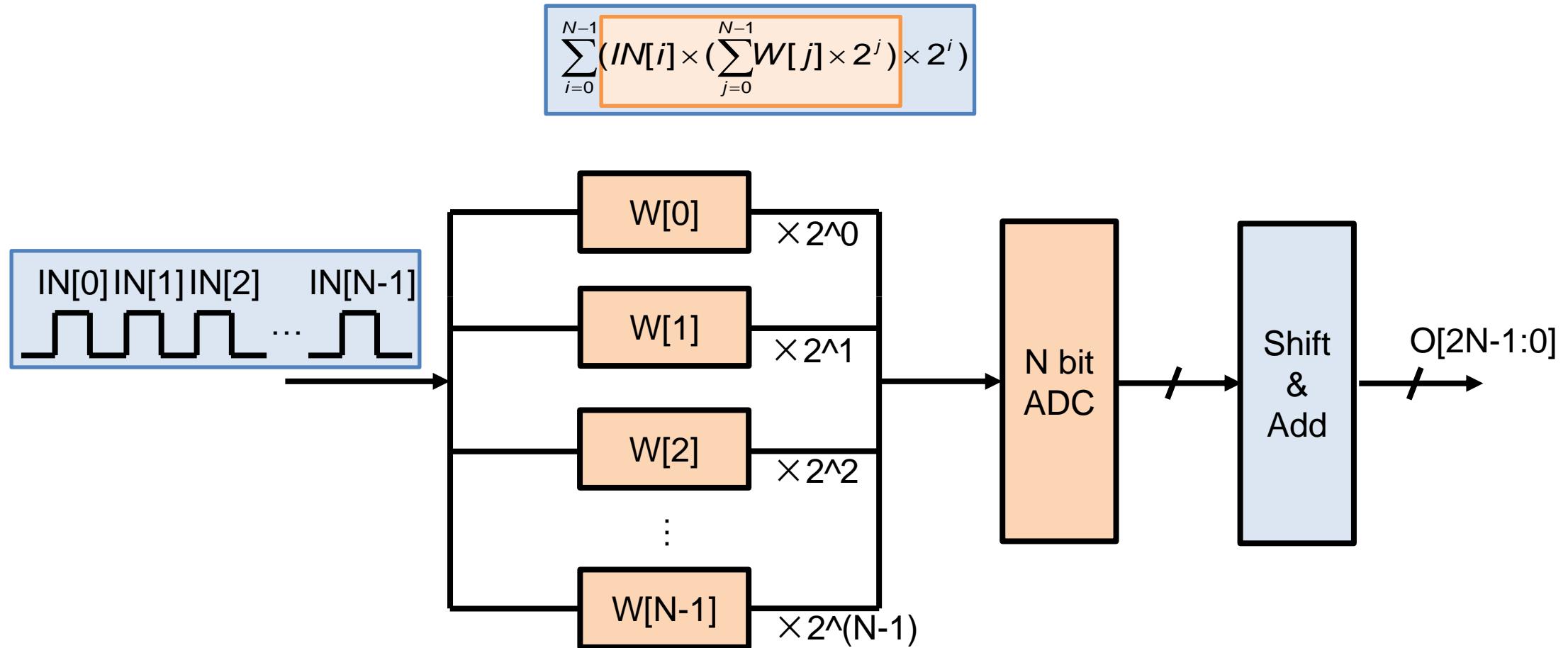
- **Basic observation: N bit × N bit Multiplication can be transformed into different forms, corresponding to different CIM topologies**

# Our Motivation



- **DAC based bit parallel scheme ACIM topology**
  - High throughput but quite high ADC bit precision, more accuracy loss

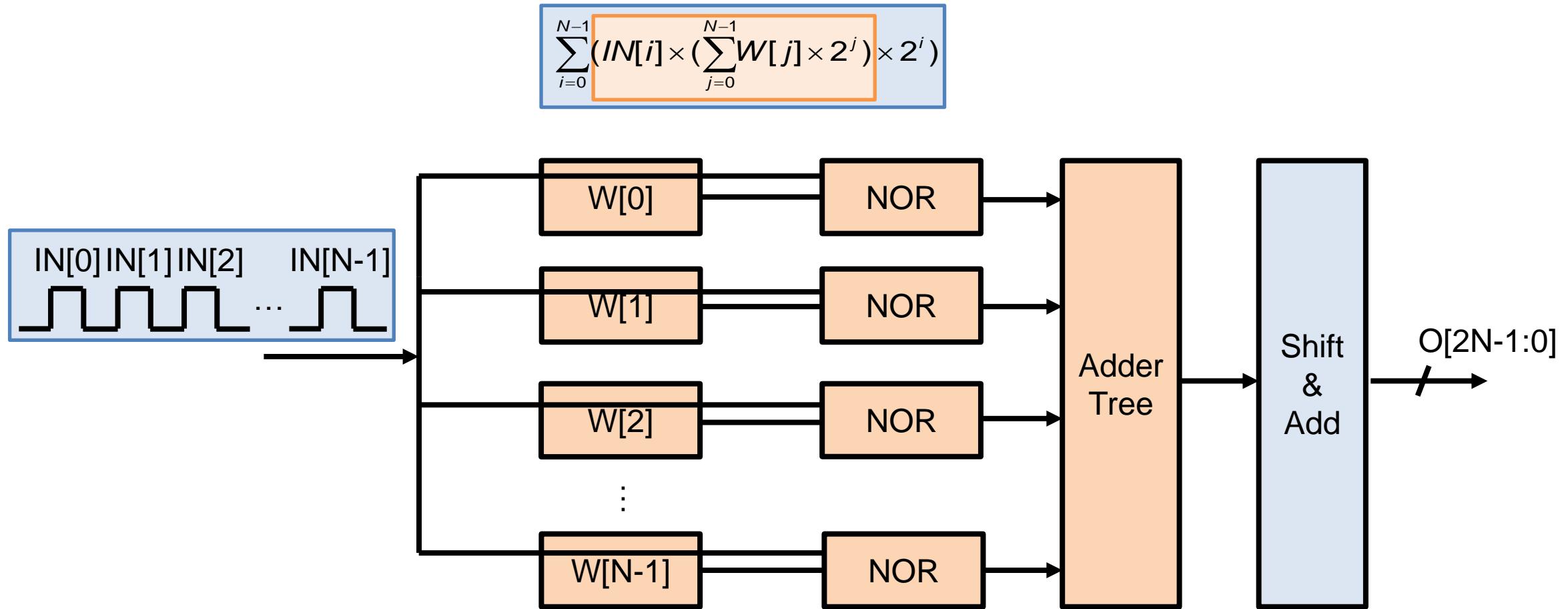
# Our Motivation



## ■ Bit serial scheme based ACIM topology

- Lower throughput but lower ADC bit precision requirement

# Our Motivation

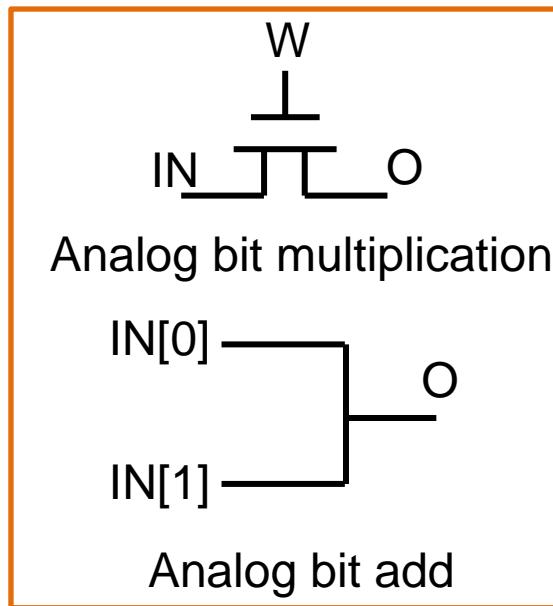


- Bit serial scheme based DCIM topology
  - No ADC, no accuracy loss but embedded logic gates needed

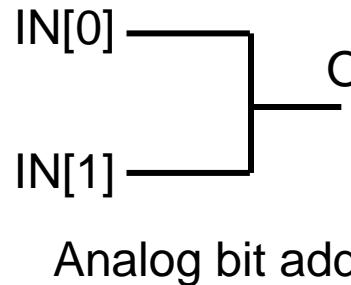
# Our Motivation

$$\sum_{i=0}^{2N-2} \left( \sum_{j=\max(0, i-(N-1))}^{\min(i, N-1)} IN[j] \times W[i-j] \right) \times 2^i$$

Analog Part



Analog bit multiplication



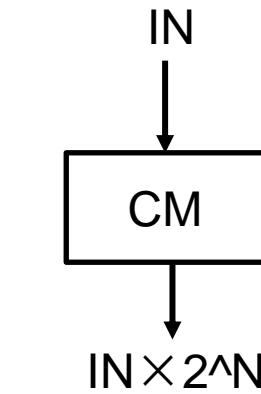
Analog bit add



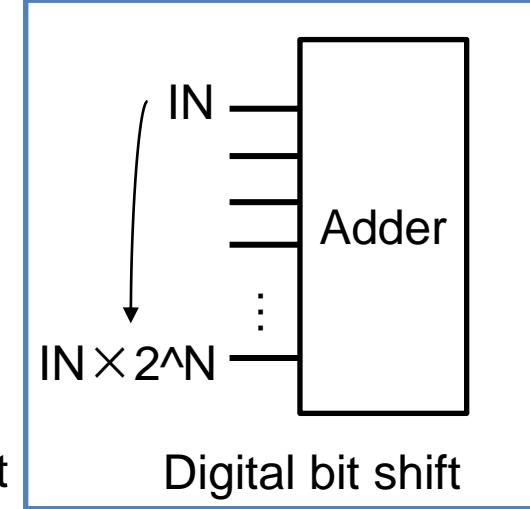
Digital bit multiplication



Digital bit add



Analog bit shift



Digital bit shift

## Motivation of our proposed Hybrid CIM scheme

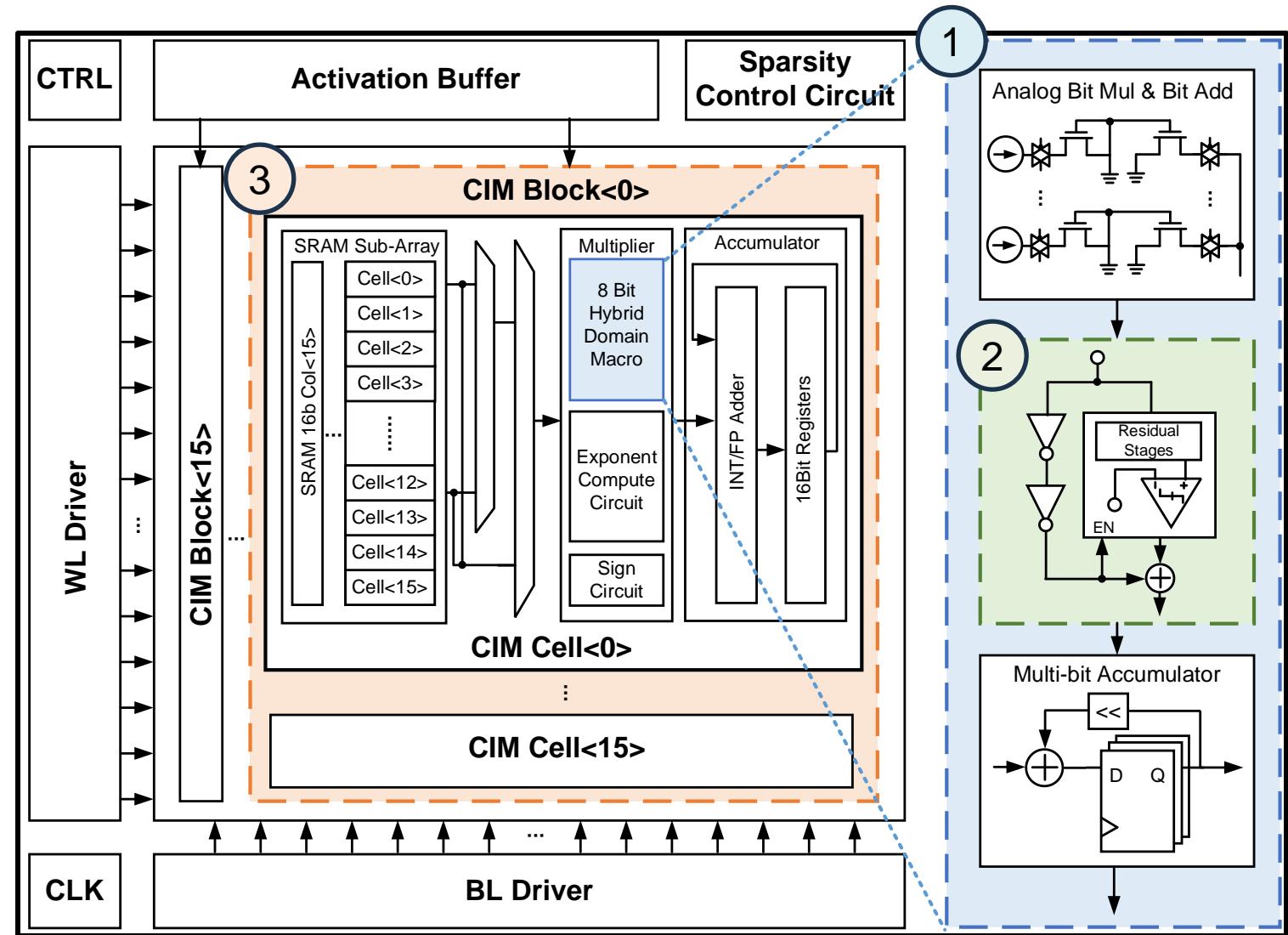
- Remove bit shift process from analog CIM thus lowering ADC bit precision

# Outline

- **Background, Challenges and Motivation**
- **Proposed Hybrid-domain SRAM-CIM Macro**
  - Macro Overall Architecture
  - Analog In-Array & Digital Out-Array Hybrid-Domain CIM
  - Logarithm Bit-Width Residual ADC Scheme
  - Outer-product based FP/INT CIM Block Architecture
- **Measurement Result**
- **Conclusion**

# Macro Overall Architecture

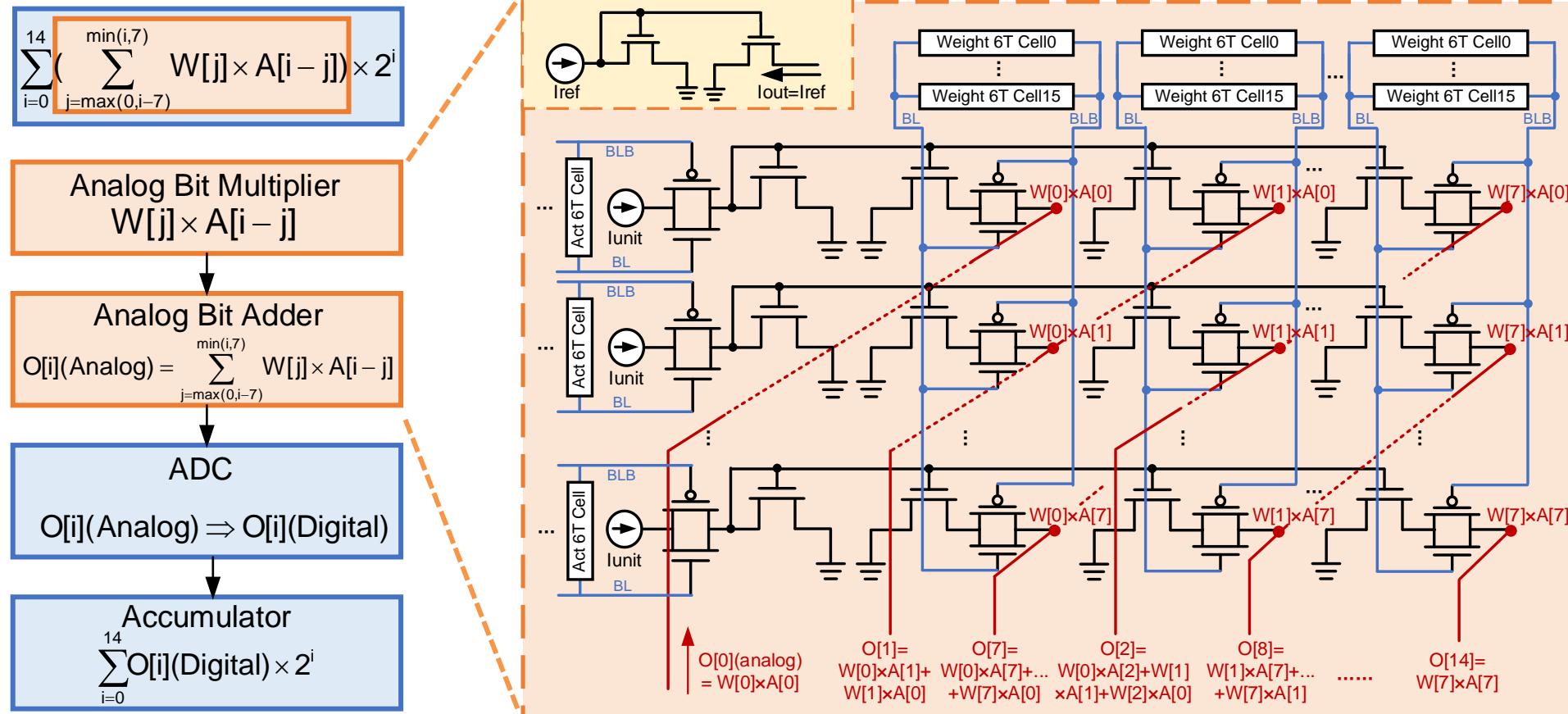
- ① Hybrid-domain 8b Macro with Analog In-array & Digital Out-array Computing
- ② Logarithm Bit-width Zero-canceling Residual Analog-Digital-Convertor
- ③ Outer-product Based Floating-Point/Fixed-Point CIM Block Architecture



# Outline

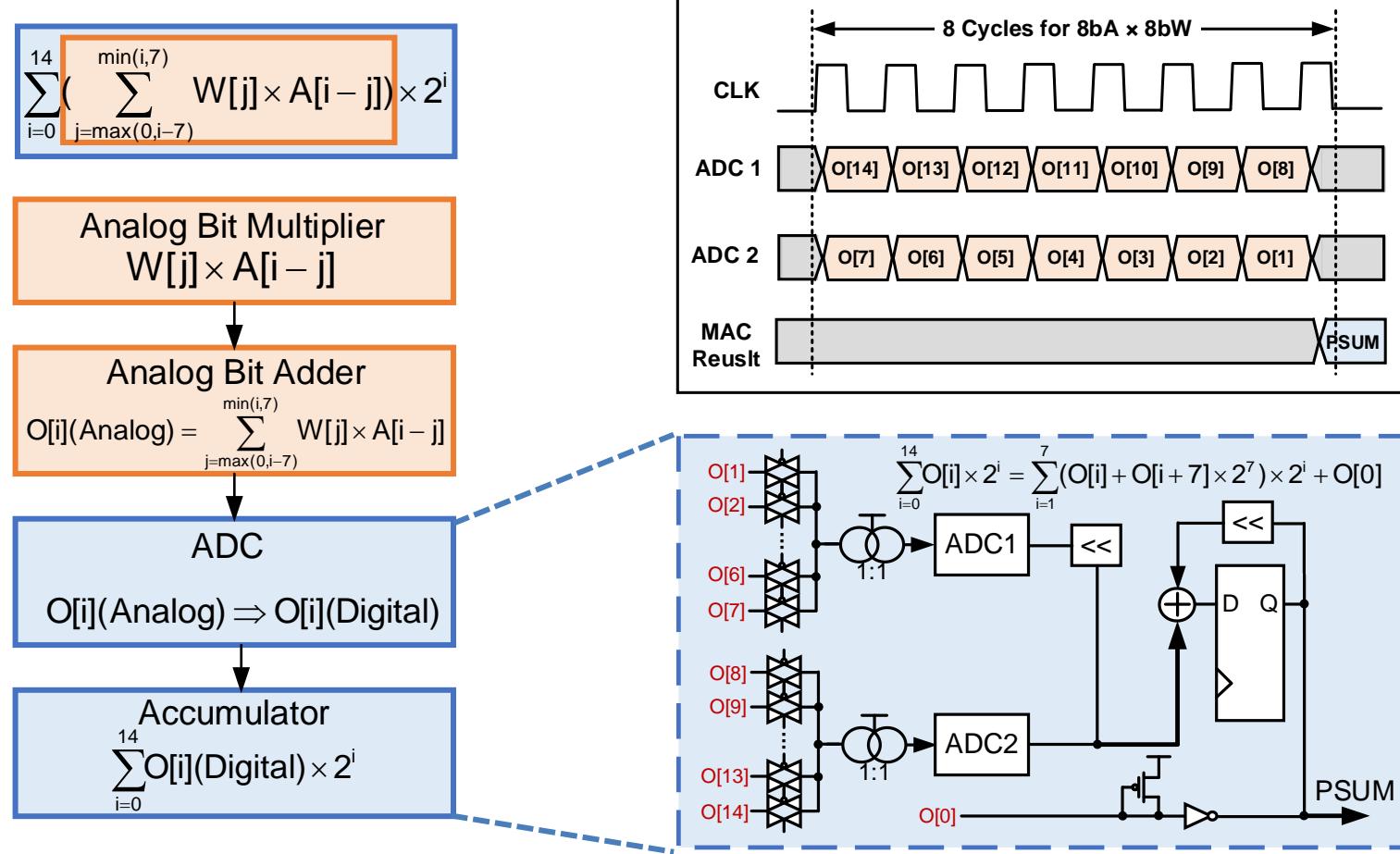
- **Background, Challenges and Motivation**
- **Proposed Hybrid-domain SRAM-CIM Macro**
  - Macro Overall Architecture
  - Analog In-Array & Digital Out-Array Hybrid-Domain Scheme
  - Logarithm Bit-Width Residual ADC Scheme
  - Outer-product based FP/INT CIM Block Architecture
- **Measurement Result**
- **Conclusion**

# Analog In-Array & Digital Out-Array Hybrid-Domain Scheme



- **Analog bit multiply and bit adder for  $\sum_{\max(0,i-N+1)}^{\min(i,N-1)} W[j] \times A[i-j]$** 
  - Current based analog CIM scheme with current mirror and transmission gates array

# Analog In-Array & Digital Out-Array Hybrid-Domain Scheme



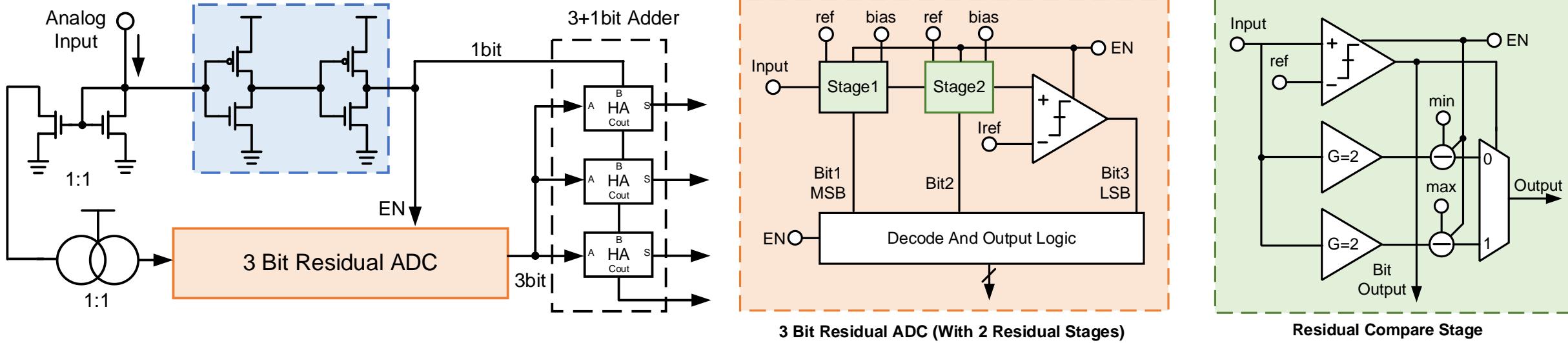
## ■ ADC and accumulator for $\sum_{i=0}^{2N-2} O[i] \times 2^i$

- Partial sums are converted into digital domain by ADC and shifted and added by 8-cycle digital accumulator

# Outline

- **Background, Challenges and Motivation**
- **Proposed Hybrid-domain SRAM-CIM Macro**
  - Macro Overall Architecture
  - Analog In-Array & Digital Out-Array Hybrid-Domain CIM
  - Logarithm Bit-Width Residual ADC Scheme
  - Outer-product based FP/INT CIM Block Architecture
- **Measurement Result**
- **Conclusion**

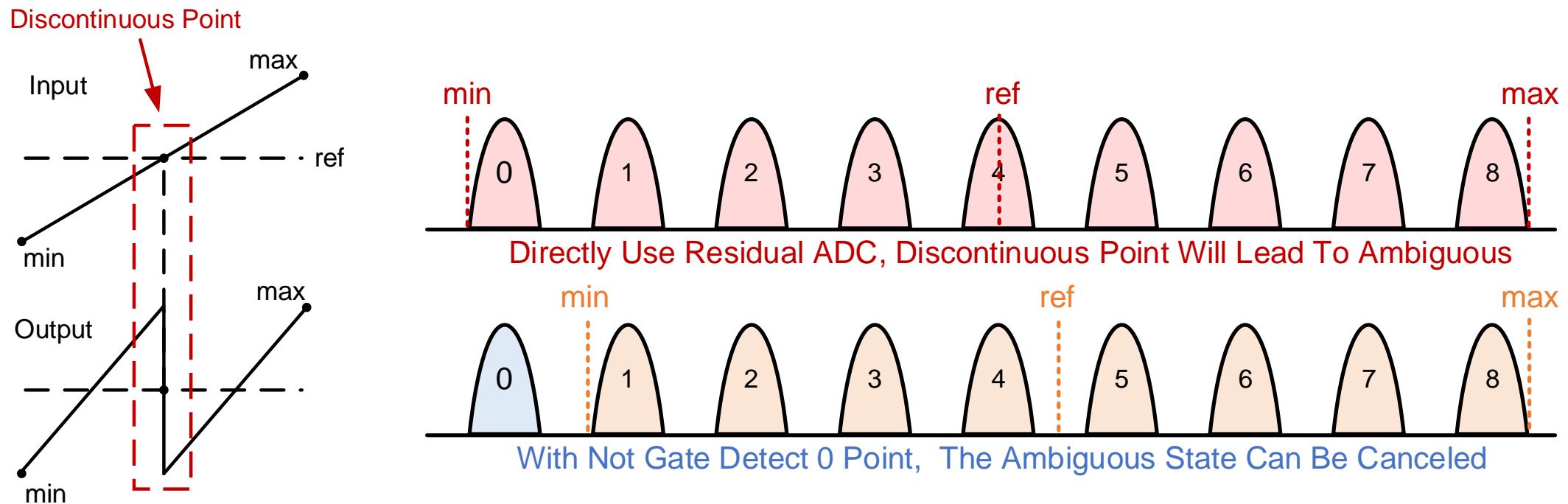
# Logarithm Bit-Width Residual ADC Scheme



## ■ Residual ADC Architecture

- 1. Residual ADC to effectively distinguish non-zero states with  $\log_2 8 = 3b$
- 2. When no current input, the residual ADC shuts down to save power
- 3. Residual stage to ensure that the working current within the ADC remains within a constant range and to prevent it from diminishing over stages

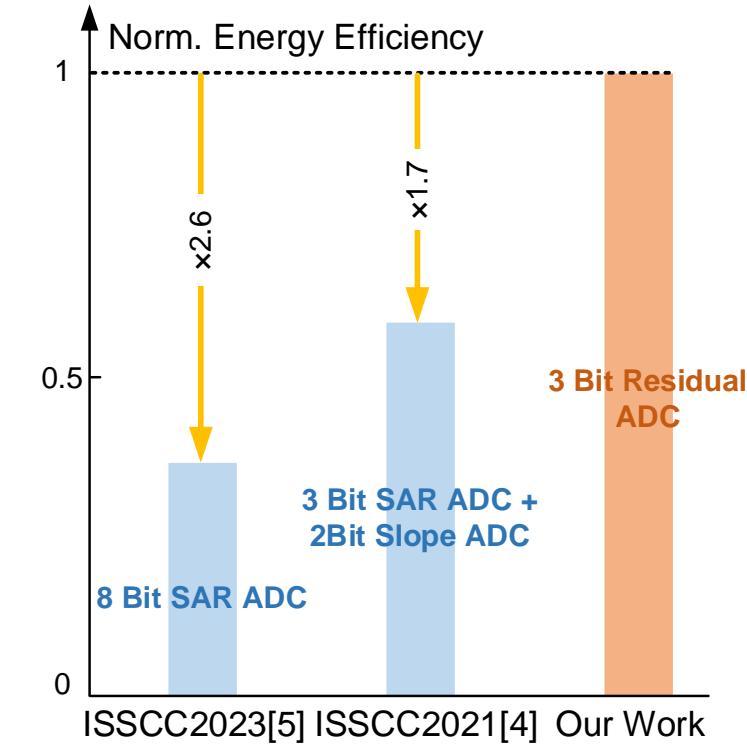
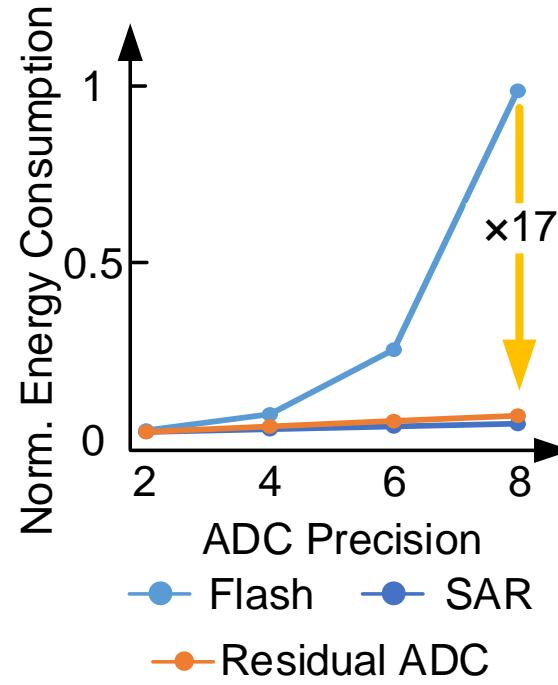
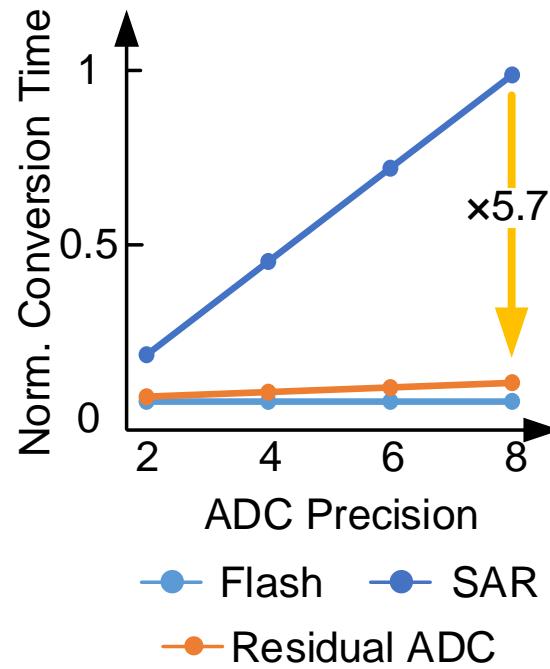
# Logarithm Bit-Width Residual ADC Scheme



## ■ Zero detection to cancel possible ambiguous

- The discontinuous point in the residual waveform will not harm as the zero value is judged by the not gates, allowing the reference current to be set outside of the signal effective range and canceling out ambiguity.

# Logarithm Bit-Width Residual ADC Scheme



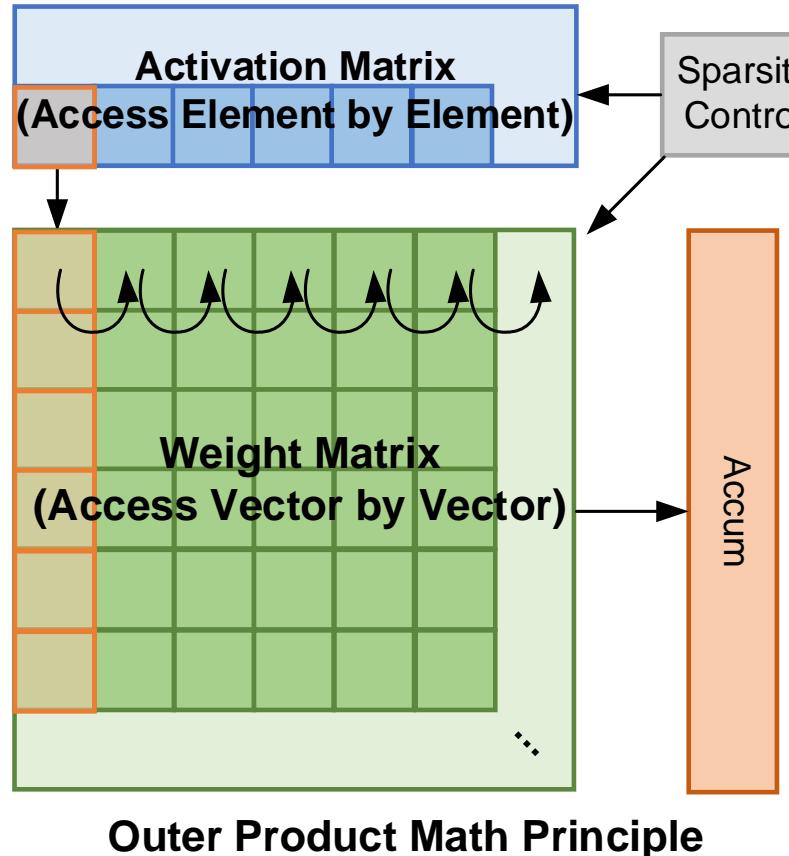
## ■ Comparison with Flash ADC and SAR ADC

- Throughput comparable to that of Flash ADC and energy consumption on par with SAR ADC
- All perform 8bIN  $\times$  8bW, energy efficiency = 1/(conversion time  $\times$  power consumption)

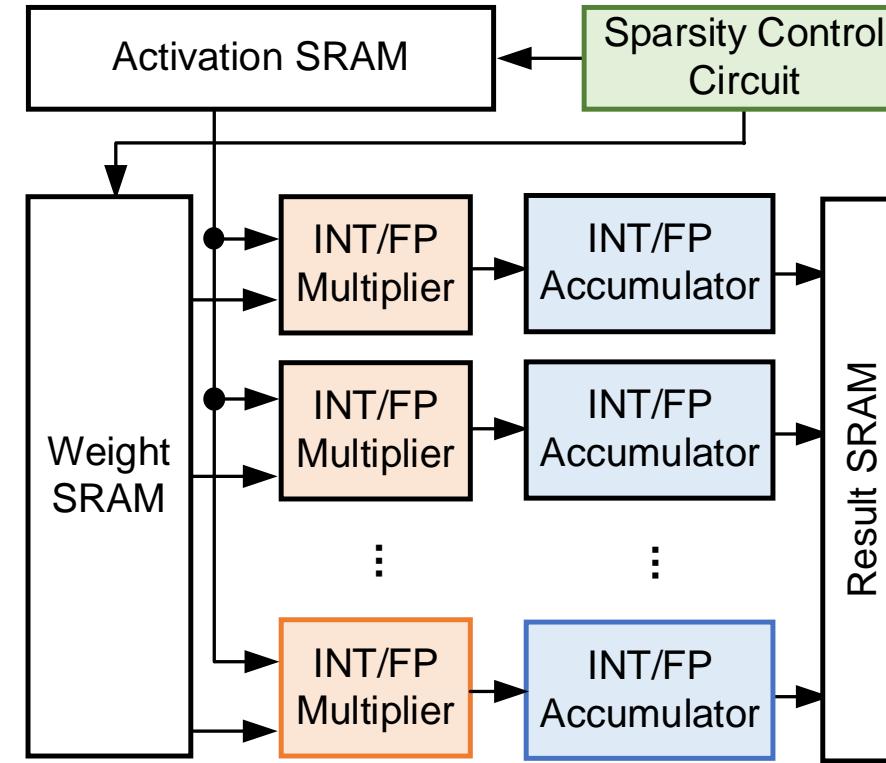
# Outline

- **Background, Challenges and Motivation**
- **Proposed Hybrid-domain SRAM-CIM Macro**
  - Macro Overall Architecture
  - Analog In-Array & Digital Out-Array Hybrid-Domain CIM
  - Logarithm Bit-Width Residual ADC Scheme
  - Outer-product based FP/INT CIM Block Architecture
- **Measurement Result**
- **Conclusion**

# Outer-product based FP/INT CIM Block Architecture



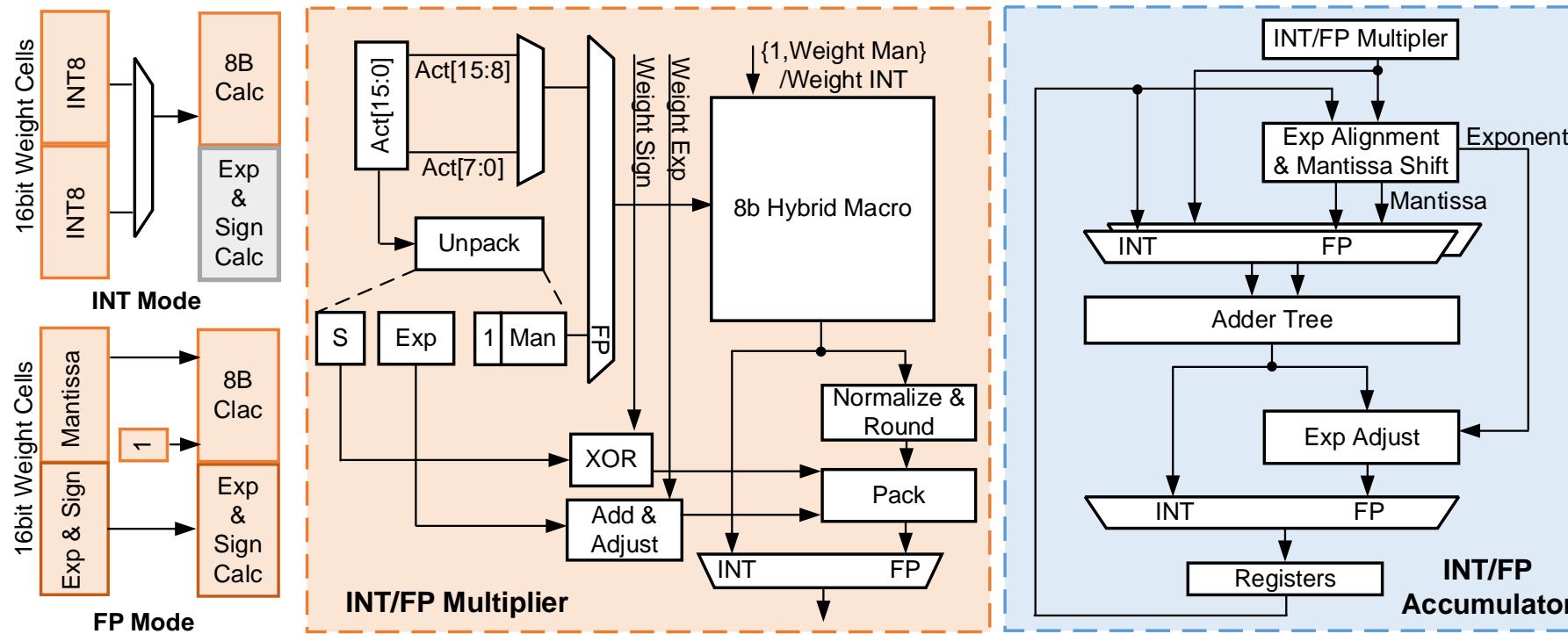
Outer Product CIM Architecture



## ■ Proposed outer product CIM architecture

- Outer product accumulates element-vector products. Consequently, eliminates the need for a large fan-in multi-level adder tree

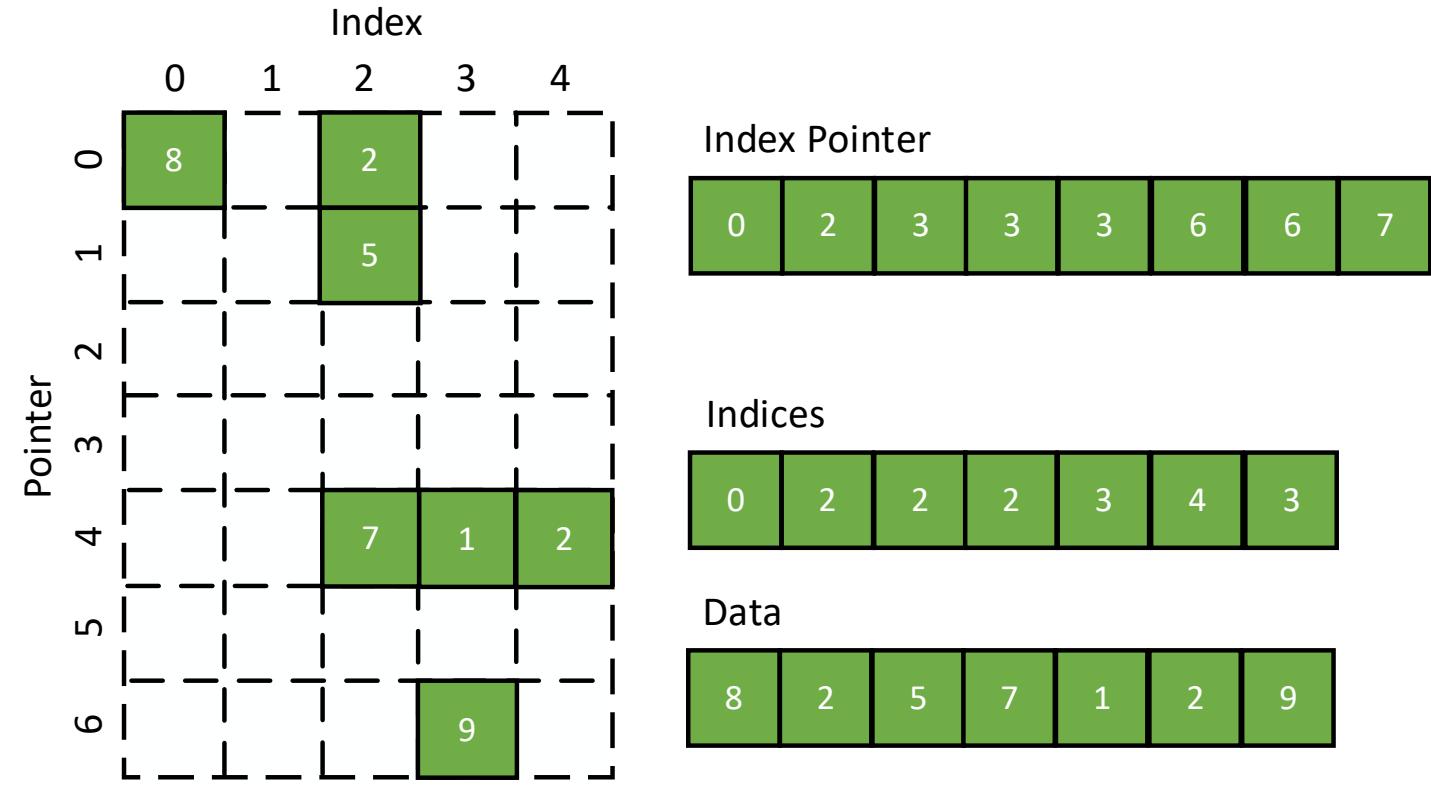
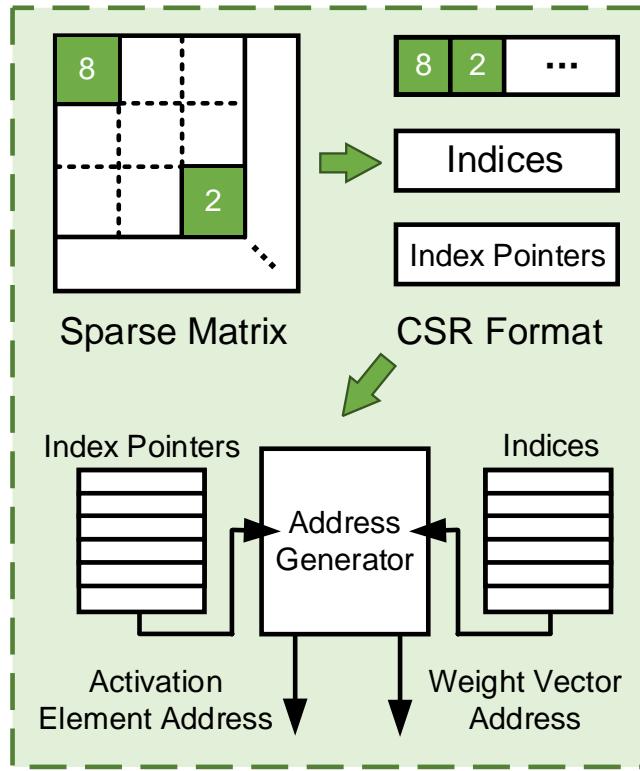
# Outer-product based FP/INT CIM Block Architecture



## ■ INT/FP multiplier and INT/FP accumulator

- In INT mode, either the high 8 bits or the low 8 bits are sent into the 8b hybrid-domain macro, accumulator performs 16b INT accumulation
- In FP mode, the mantissa of BF16 is sent to the macro, and accumulator operates BF16 FP accumulation

# Outer-product based FP/INT CIM Block Architecture



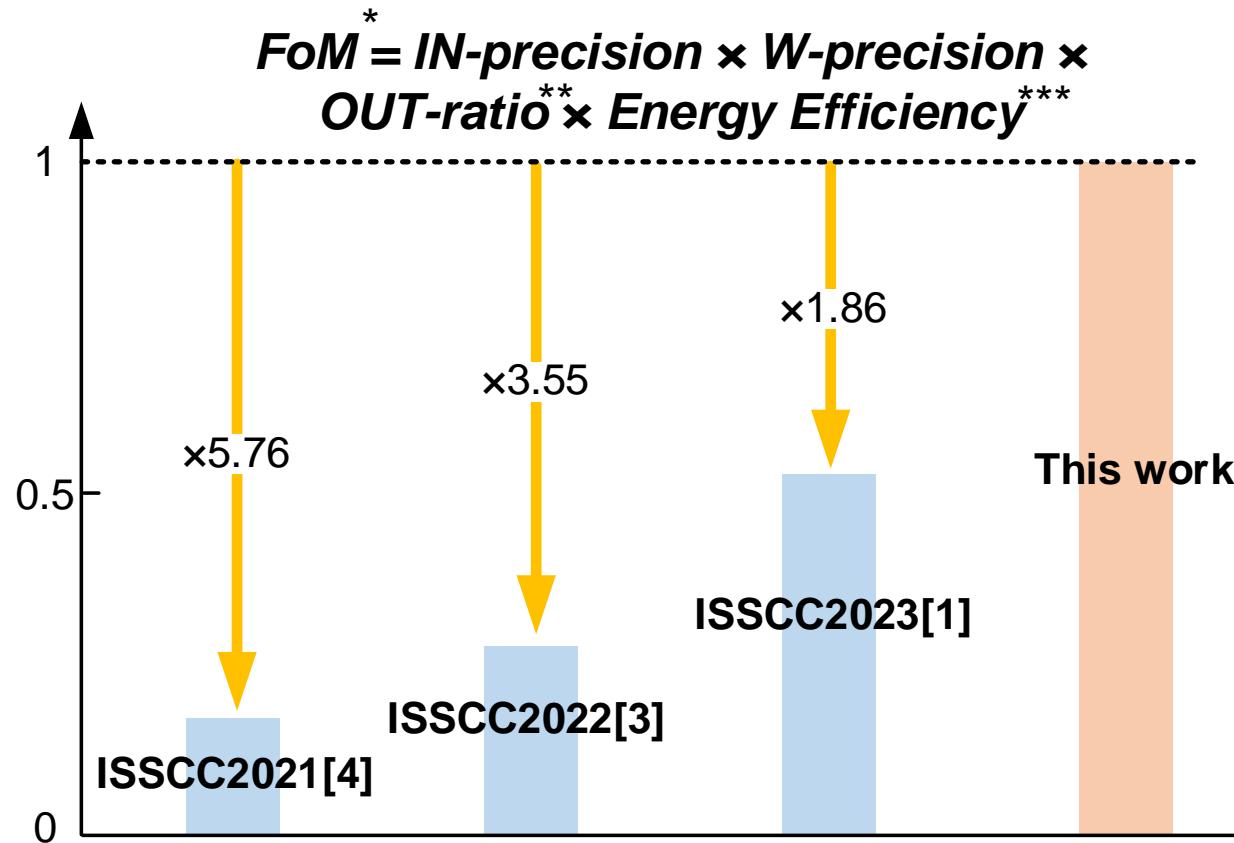
## ■ Compressed sparse row for sparse activation matrix

- The sparse control circuit, guided by the indices and index pointers in the compressed sparse row format of the sparse matrix, avoids accessing and computing zero elements within the matrix

# Outline

- **Background, Challenges and Motivation**
- **Proposed Hybrid-domain SRAM-CIM Macro**
  - Macro Overall Architecture
  - Analog In-Array & Digital Out-Array Hybrid-Domain CIM
  - Logarithm Bit-Width Residual ADC Scheme
  - Outer-product based FP/INT CIM Block Architecture
- **Measurement Result**
- **Conclusion**

# FoM Comparison



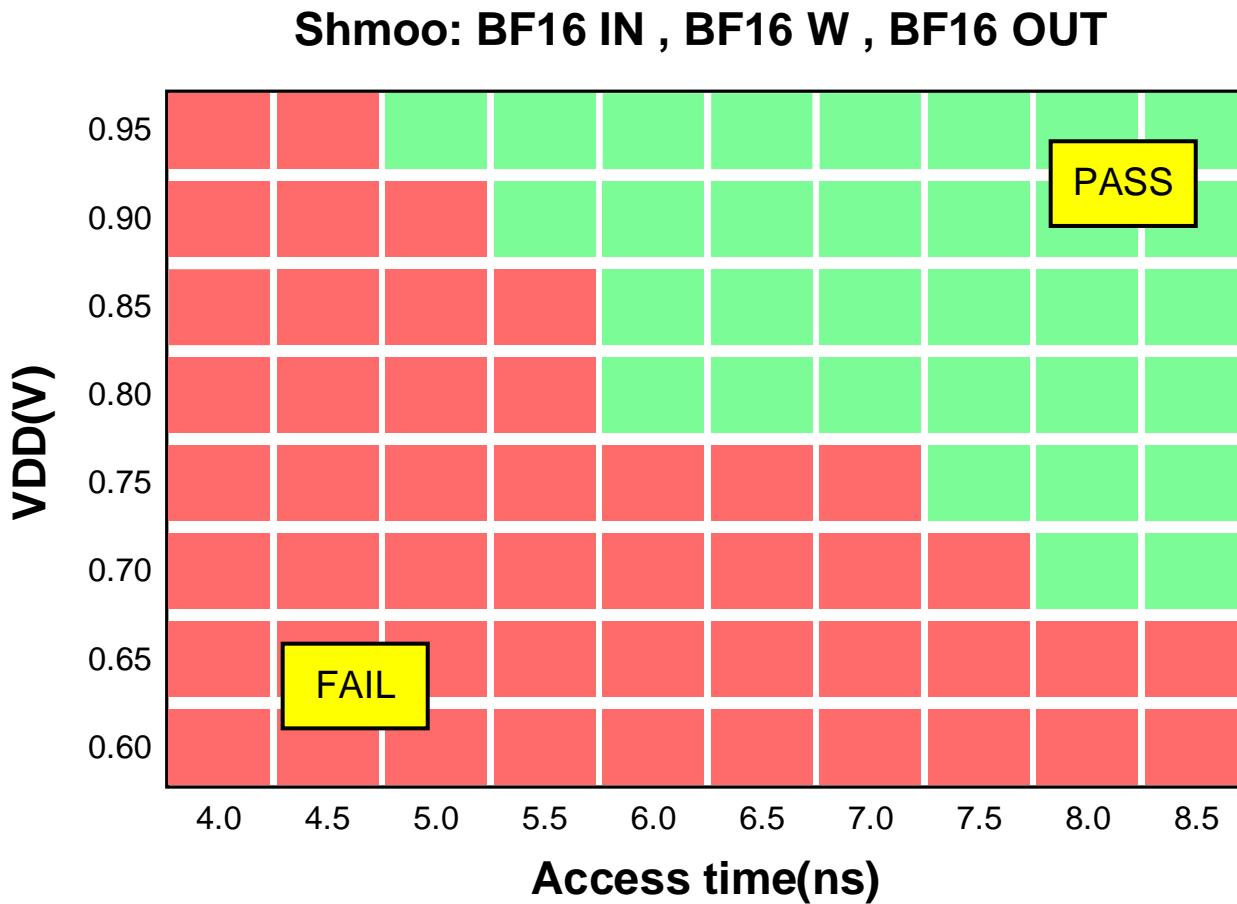
\* When calculating figure of merit (FoM), BF16 is considered as 16

\*\* Output Radio = Real output accuracy / Ideal output accuracy

\*\*\* All take maximum average energy efficiency, accuracy running on Cifar-100

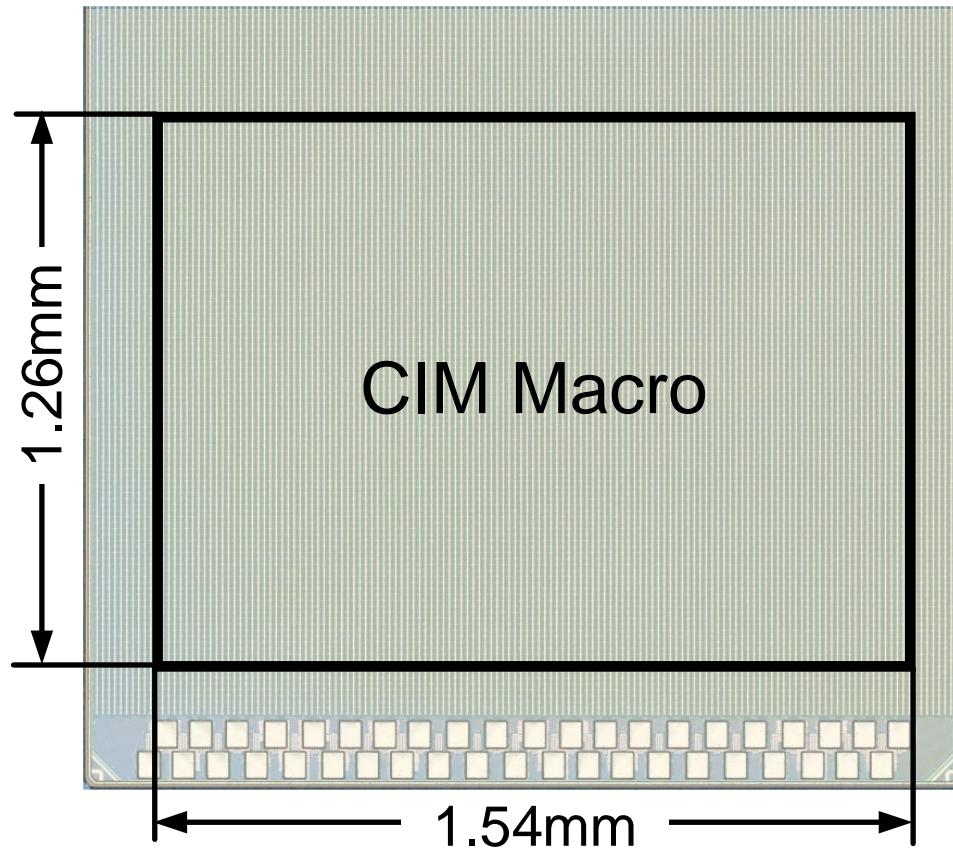
- Compares with ISSCC'21, ISSCC'22, and ISSCC'23
  - >1.86x FoM improvement

# Shmoo Plot



- **Measurement results @ BF16-IN, BF16-W, BF16-OUT**
  - Access time ( $t_{AC}$ ) = 4.78ns @ VDD=0.95V

# Chip Summary

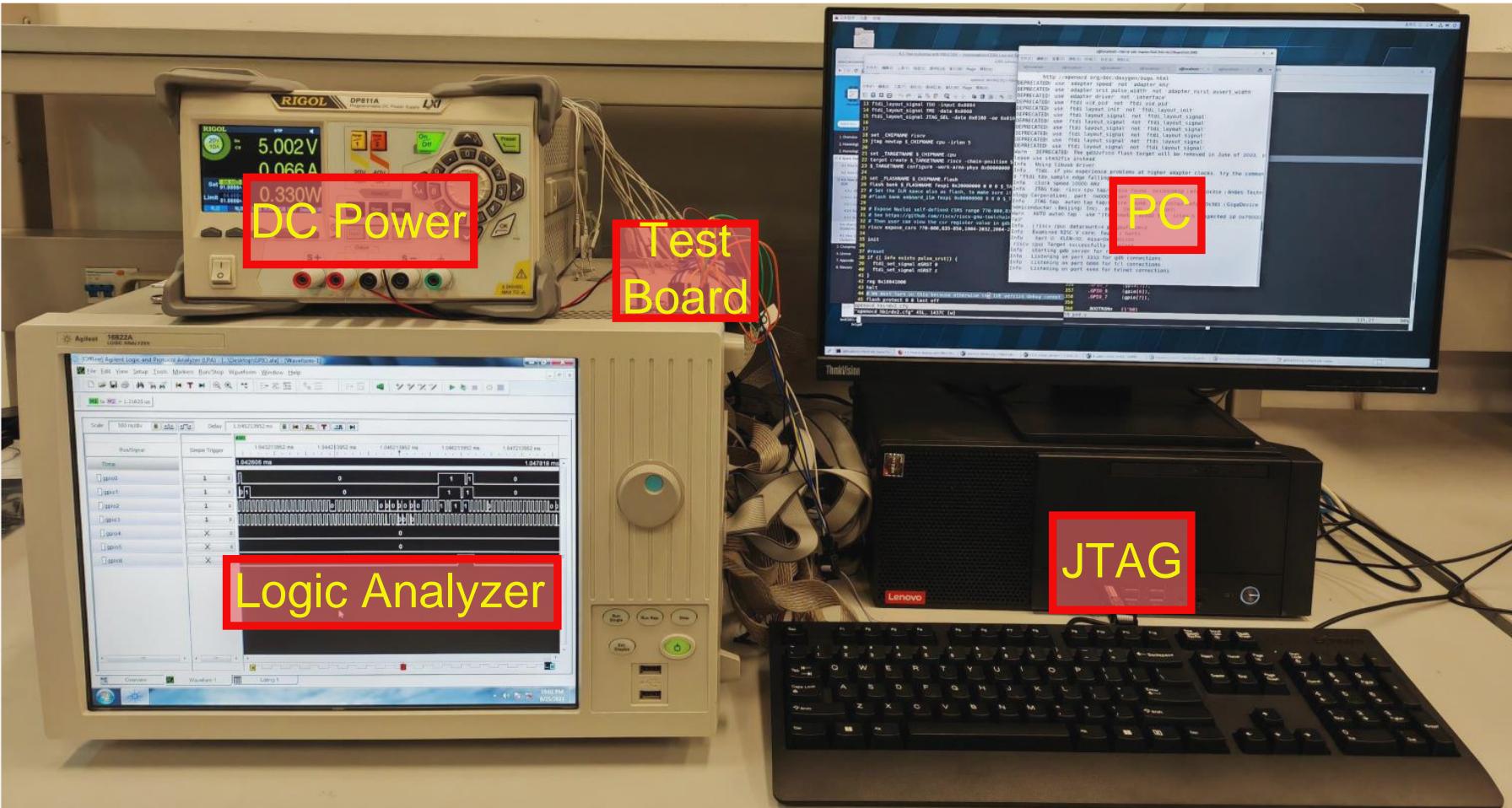


\* Performance in 90% input sparsity (ResNet-18)

\*\* Using ResNet-18 model, the software baseline was 75.31%

Chip Summary		
Technology(nm)	28	
Macro Area( $\text{mm}^2$ )	1.94	
SRAM Capacity(Kb)	192Kb	
Supply Voltage(V)	0.7-0.95	
Number of Input Channels	256	
Input Precision	BF16	INT8
Weight Precision	BF16	INT8
Throughput	1.98-4.28 TFLOPS	2.89-5.31 TOPS
Energy Efficiency (Average performance)	16.55-32.78 TFLOPS/W	22.78-50.53 TOPS/W
Energy Efficiency* (Peak performance)	36.41-72.12 TFLOPS/W	50.12-111.17 TOPS/W
Accuracy Loss** (@Cifar100)	-0.05%	-1.57%

# Measurement System



34.6: A 28nm 72.12TFLOPS/W Hybrid-Domain Outer-Product Based Floating-Point SRAM Computing-in-Memory Macro with Logarithm Bit-Width Residual ADC

# Outline

- **Background, Challenges and Motivation**
- **Proposed Hybrid-domain SRAM-CIM Macro**
  - Macro Overall Architecture
  - Analog In-Array & Digital Out-Array Hybrid-Domain CIM
  - Logarithm Bit-Width Residual ADC Scheme
  - Outer-product based FP/INT CIM Block Architecture
- **Measurement Result**
- **Conclusion**

# Conclusion

## ■ Features of proposed Hybrid-domain FP CIM macro

- Analog In-Array & Digital Out-Array Hybrid-Domain CIM
- Logarithm Bit-Width Residual ADC
- Outer-product based FP/INT CIM Block Architecture

## ■ A 28nm FP SRAM CIM Macro is Verified

- BF16IN-BF16W-BF16OUT
  - Throughput : 1.98-4.28 TFLOPS
  - Energy Efficiency : 16.55-72.12 TFLOPS/W
- INT8IN-INT8W-INT16OUT
  - Throughput : 2.89-5.31 TOPS
  - Energy Efficiency : 22.78-111.17 TOPS/W

# Thank you for your kind attention!



Please Scan to Rate  
This Paper



# A 28-nm 2.4-Mb/mm<sup>2</sup> 6.9-16.3-TOPS/mm<sup>2</sup> eDRAM-LUT-Based Digital Computing-in-Memory Macro with In-Memory Encoding and Refreshing

Yifan He<sup>1</sup>, Shupei Fan<sup>1</sup>, Xuan Li<sup>1</sup>, Luchang Lei<sup>1</sup>, Wenbin Jia<sup>1</sup>, Chen Tang<sup>1</sup>,  
Yaolei Li<sup>1</sup>, Zongle Huang<sup>1</sup>, Zhike Du<sup>1</sup>, Jinshan Yue<sup>2</sup>, Xueqing Li<sup>1</sup>, Huazhong  
Yang<sup>1</sup>, Hongyang Jia<sup>1</sup>, Yongpan Liu<sup>1</sup>

<sup>1</sup>Tsinghua University, Beijing

<sup>2</sup>Institute of Microelectronics of CAS, Beijing



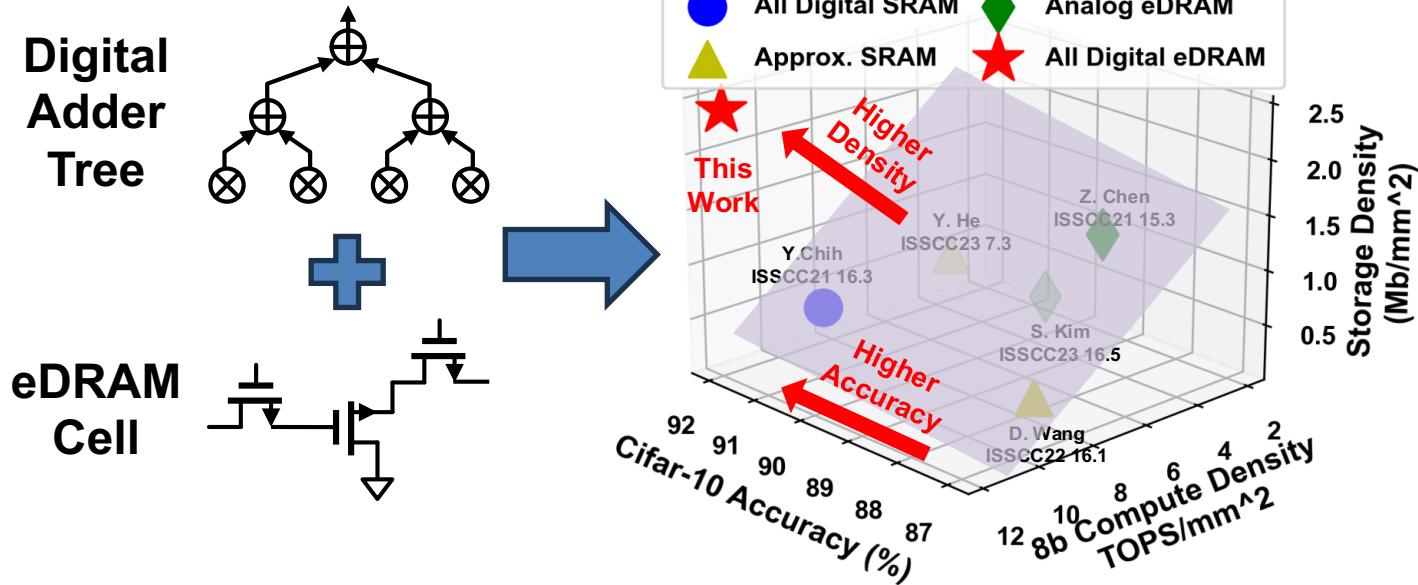
# eDRAM-based digital CIM

- Why digital CIM?

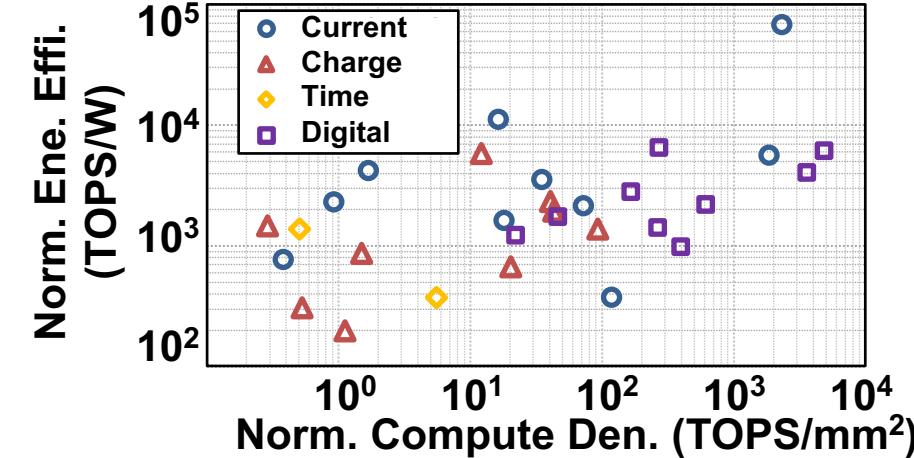
- Higher throughputs & accuracy 😊
- However, large logics overhead & low mem. density 😞

- Why eDRAM CIM?

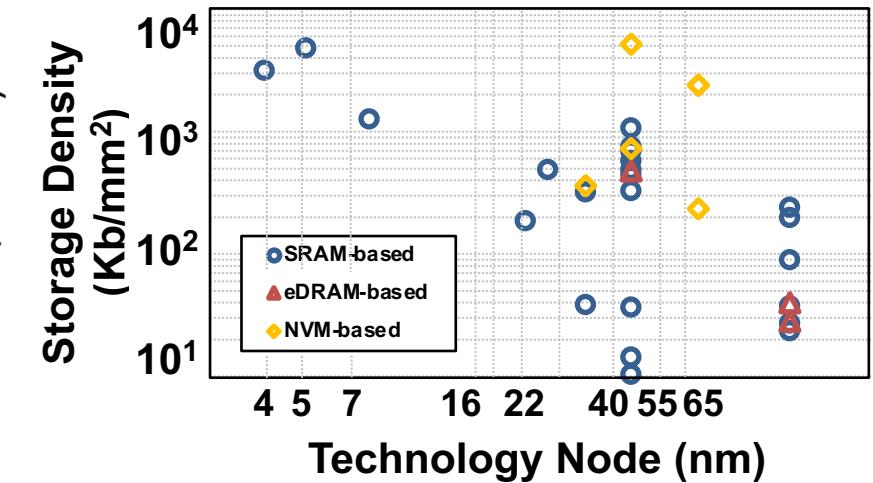
- Less transistor & potential higher density 😊
- But has refreshing overhead 😞



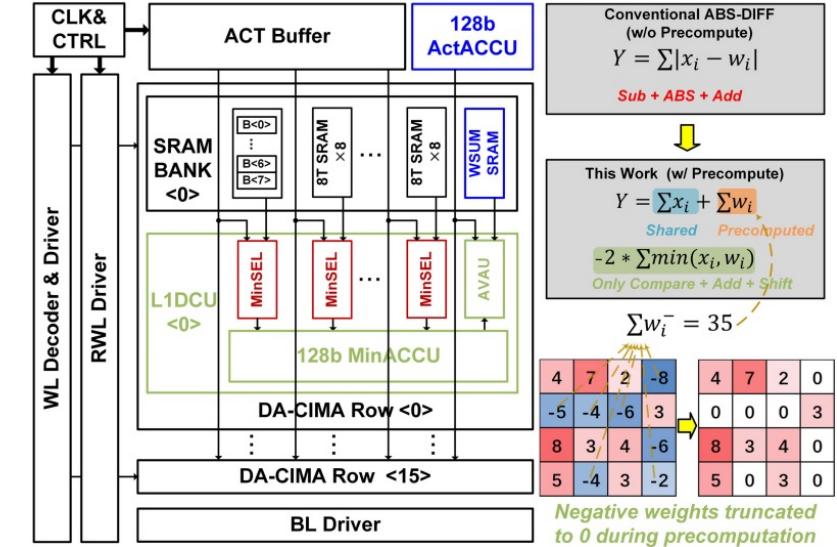
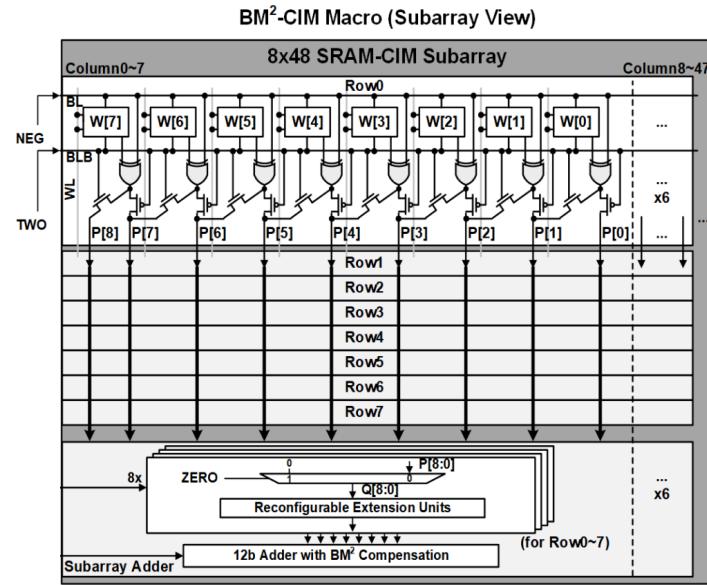
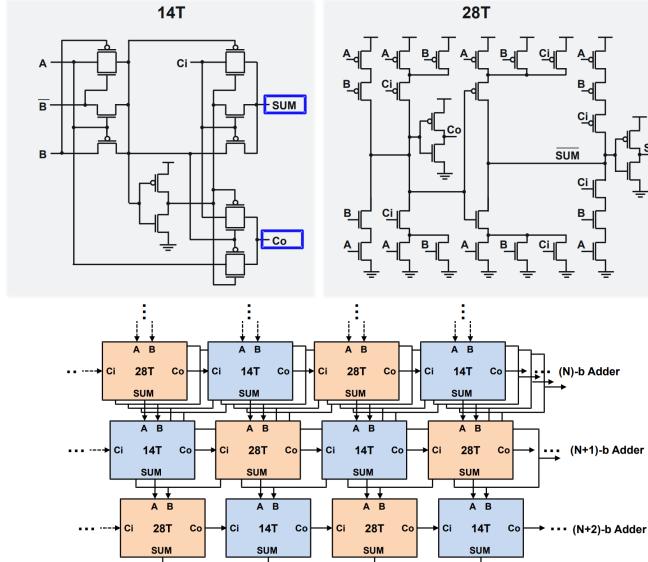
## Effi. Comp. of Analog & Digital CIM



## Den. Comp. of SRAM & eDRAM CIM



# Existing digital CIM solution



## Solution 1: Circuit Optimization

Different types of FA designs

😊 Universal improvement

😢 Limited design space

## Solution 2: Logic Optimization

Efficient MAC implementation

😊 Higher efficiency/throughput

😢 Complex logic & larger area

## Solution 3: Alg. Co-optimization

Approximate computing

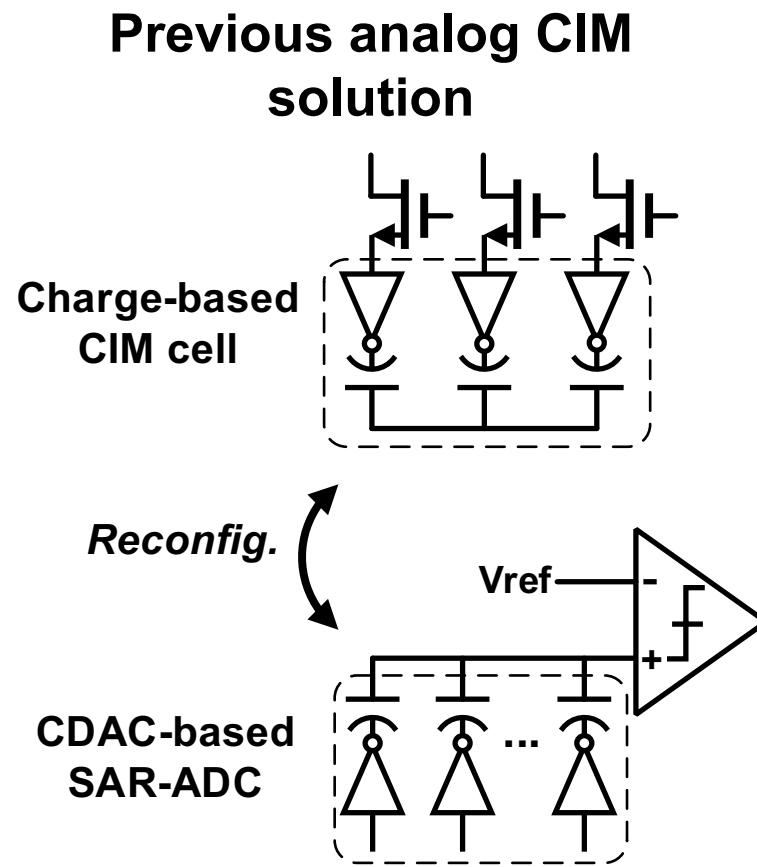
😊 Break digital logic limit

😢 Accuracy loss

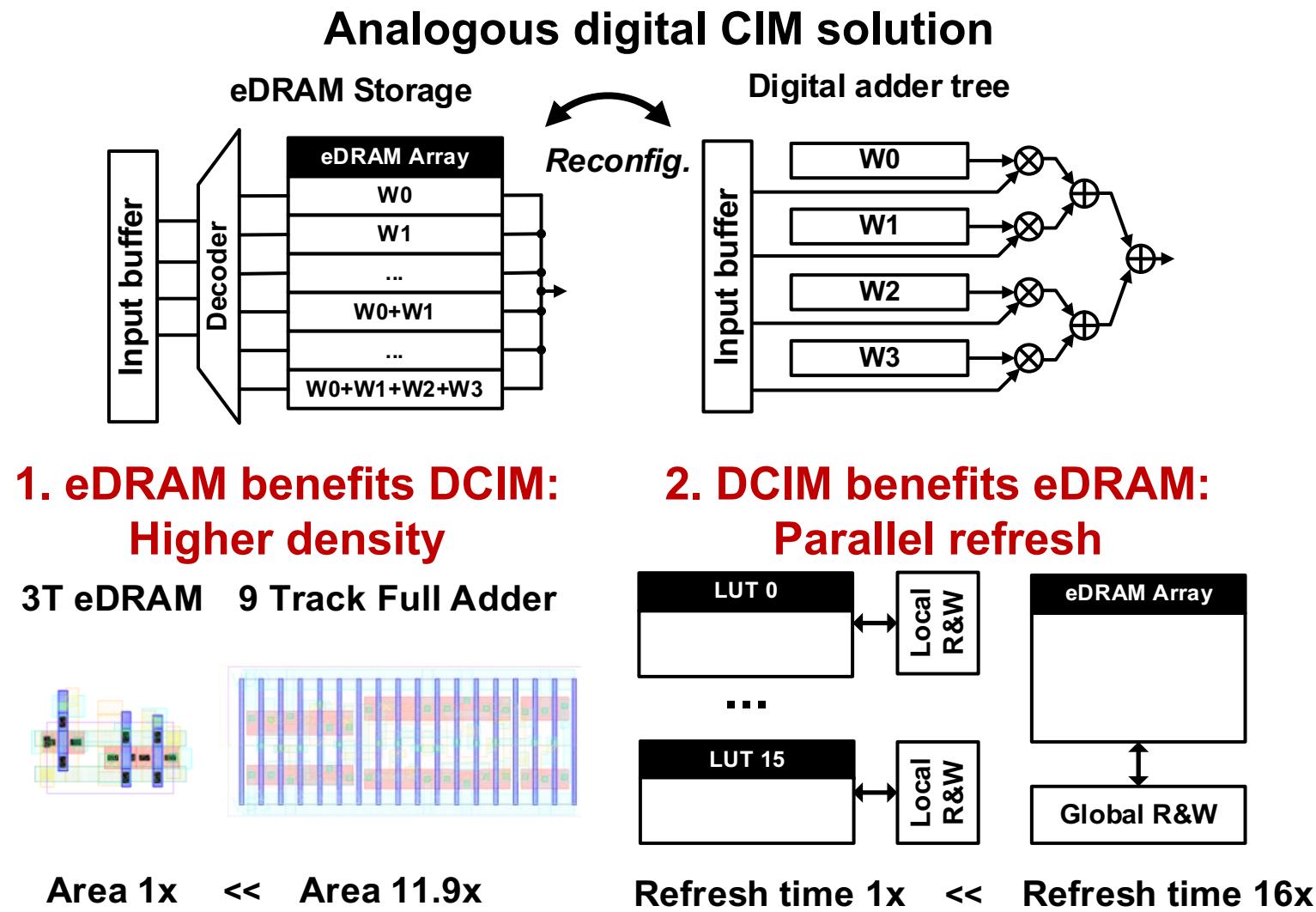
Existing digital CIM designs focus on computation circuits

Source: ISSCC'21 16.4, ISSCC'22 15.5, ISSCC'23 7.3

# Motivation to eDRAM-LUT-based digital CIM



*Motivated by ISSCC'21 15.3  
& ISSCC'23 15.5*



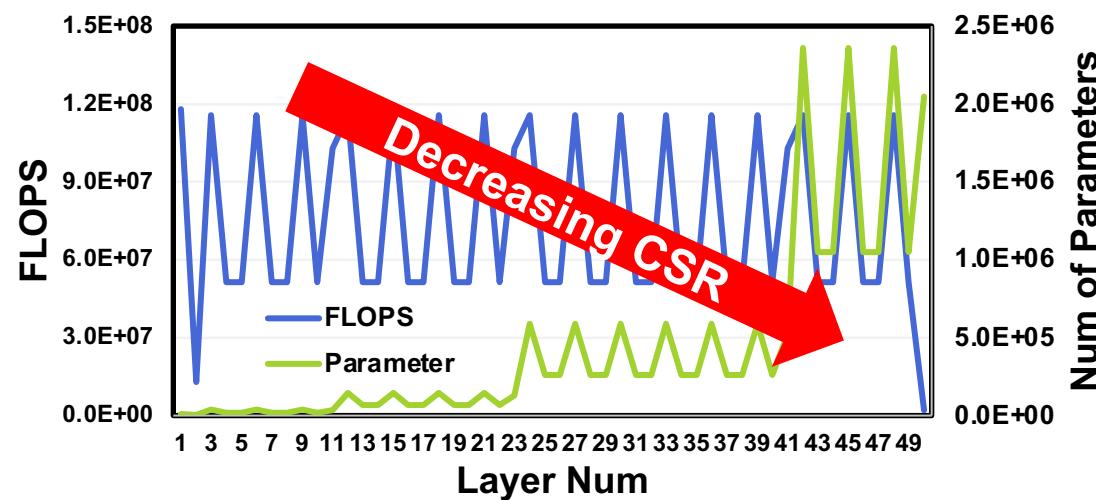
# Outline

---

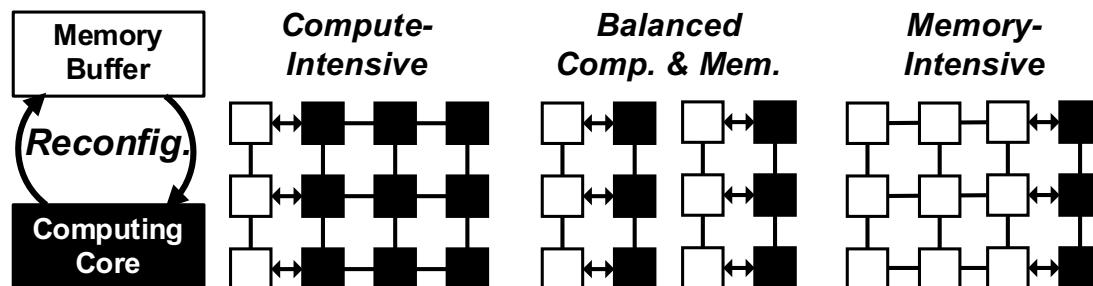
- **Introduction**
- **Challenge of eDRAM-LUT-Based Digital CIM**
- **Proposed eDRAM Digital CIM**
  - Overall architecture and basic principal
  - Two-stage eDRAM-LUT-based adder tree
  - In-memory refreshing and LUT encoding
- **Measurement Results**
- **Conclusion**

# Challenge 1: Compute-storage ratio (CSR)

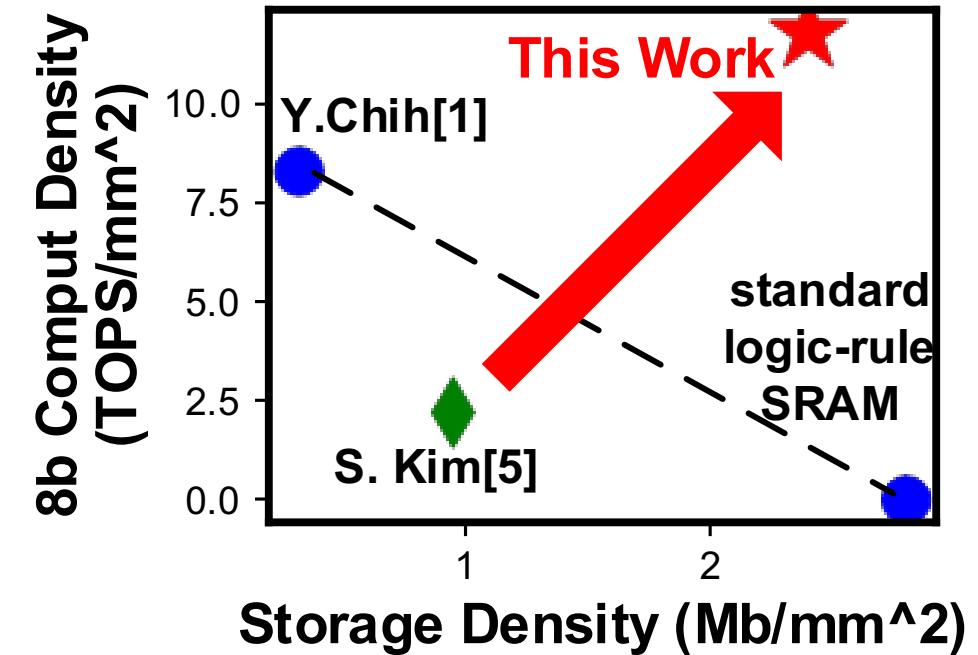
## Dynamic CSR on typical NN models



Ideal CIM : Reconfigurable compute & storage for optimal utili.

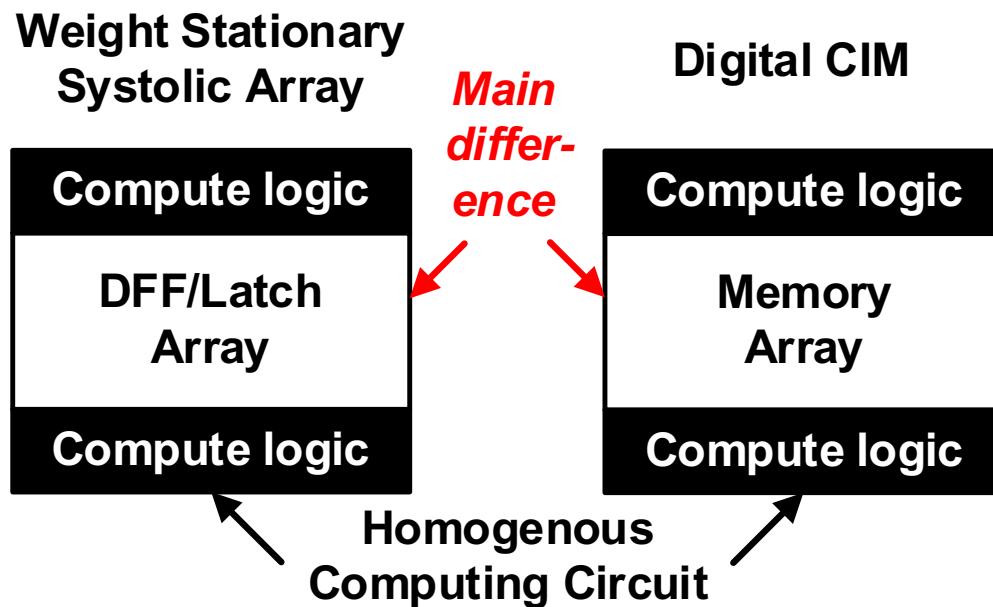


## CSR trade-off of different CIM solutions

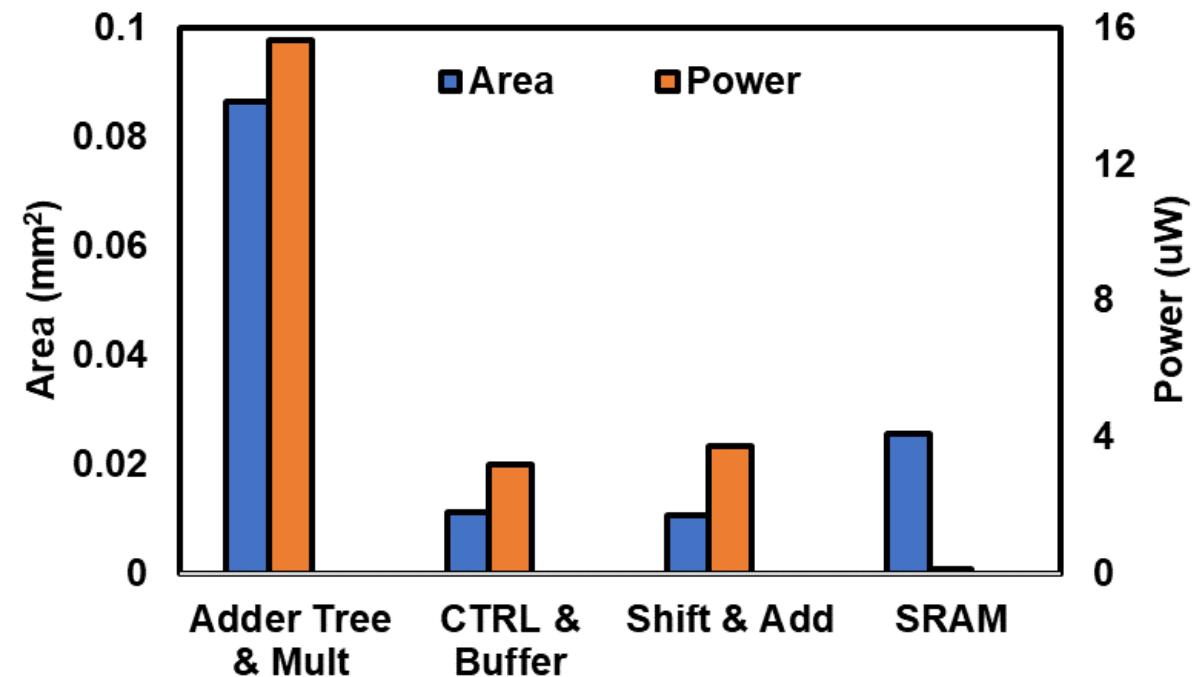


# Challenge 2: Digital logic limit

Comparison between digital CIM & traditional PE array



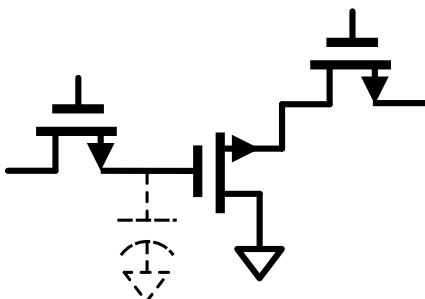
Energy & area breakdown in a typical digital CIM macro



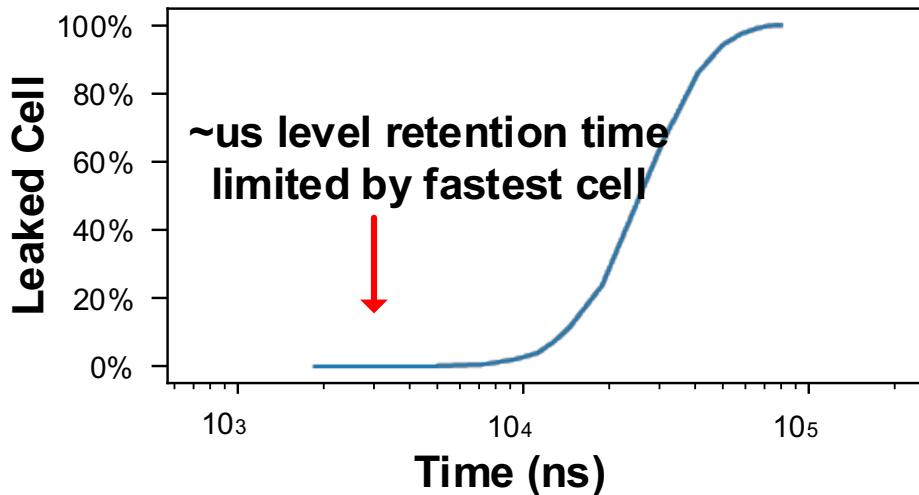
Efficiency of Digital CIM is fundamentally limited by digital logic

# Challenge 3: Refresh & encode cost

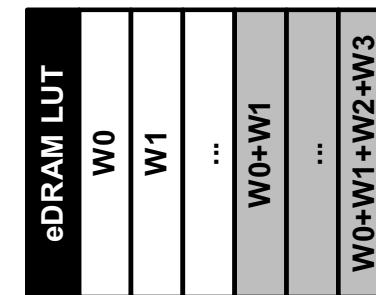
MOM-less eDRAM for ultra high density



Retention time distribution

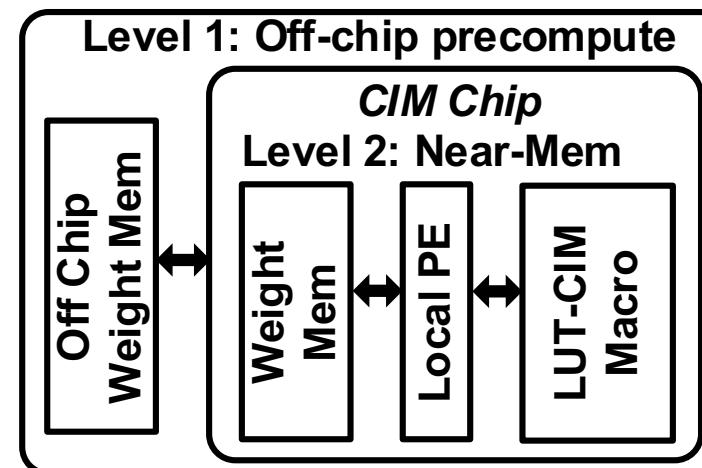


LUT Programming Overhead



$2^N$  Extra Weights for N-entry LUT

Off-chip vs Near memory encoding



	Off-Chip	Near-Mem
Comp. Cost	0x	1x
Wgt . Mem	4x	1x
Write Time	4x	4x

# Outline

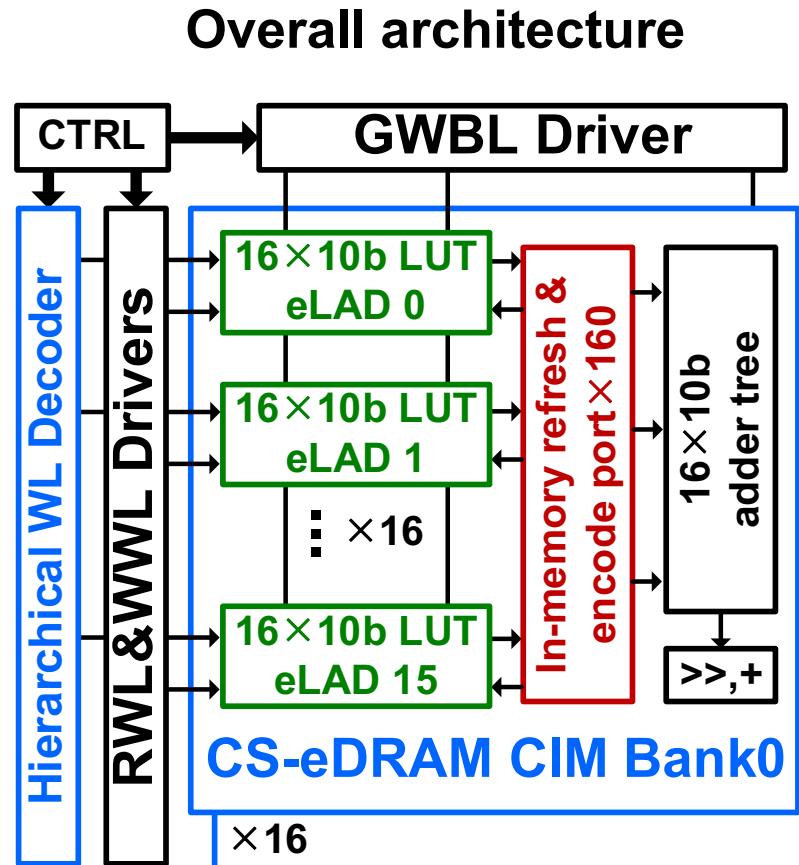
---

- **Introduction**
- **Challenge of eDRAM-LUT-Based Digital CIM**
- **Proposed eDRAM Digital CIM**
  - Overall architecture and basic principal
  - Two-stage eDRAM-LUT-based adder tree
  - In-memory refreshing and LUT encoding
- **Measurement Results**
- **Conclusion**

# eDRAM LUT-based CIM macro architecture

## Design highlights

- Top Level: Compute-storage dual mode eDRAM CIM macro
- Computing Level: eDRAM look-up-table based two-stage adder tree
- Storage Level: In-mem refresh & encoding with local R&W port

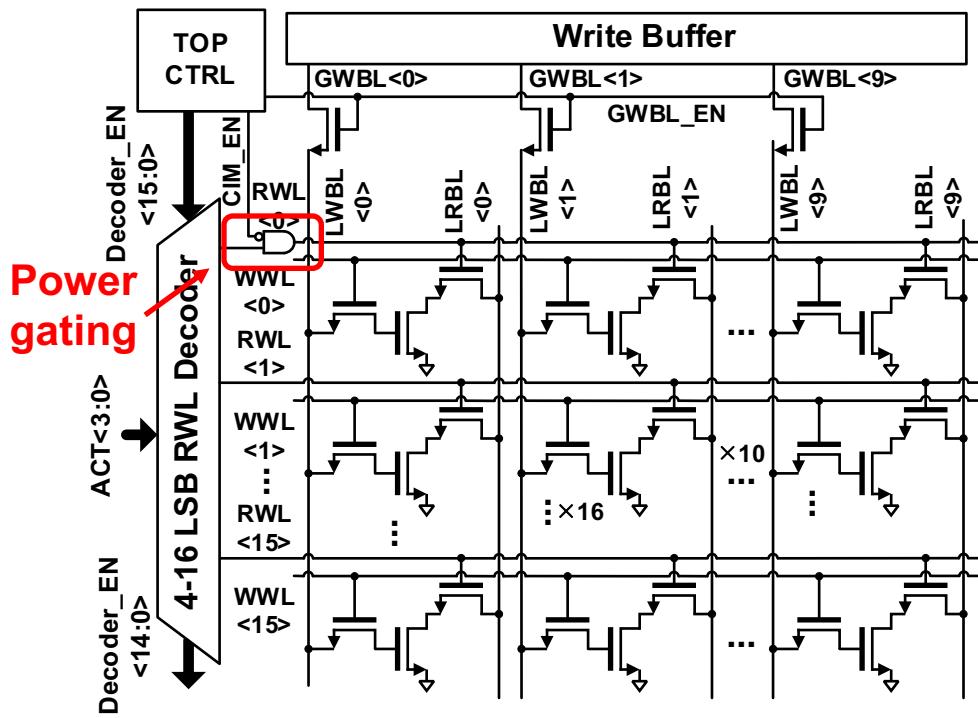


## Component

- CS-DCA: A 320x128 bit compute-storage reconfigurable eDRAM-CIM array
- eLAD: Two stage eDRAM LUT-based adder tree
- IMREP: In-memory refresh and encode port
- Others: Hierarchical WL Decoder, post accumulator, normal IO and controls

# eLAD implementation details

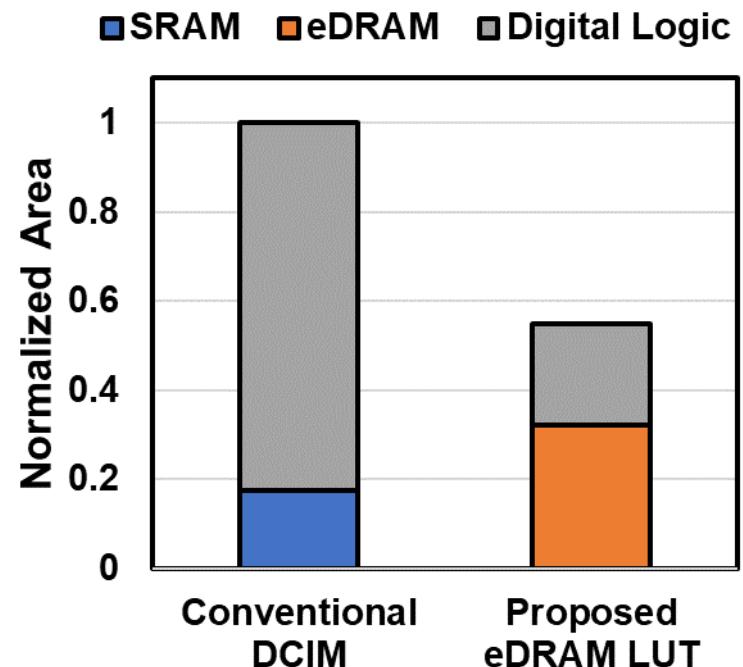
## eLAD schematic



## eLAD data mapping

WL	ACT	CIM MODE	MEM MODE
0	0000	Don't Care	DATA0
1	0001	W <sub>1</sub>	DATA1
2	0010	W <sub>2</sub>	DATA2
3	0011	W <sub>1</sub> +W <sub>2</sub>	DATA3
4	0100	W <sub>3</sub>	DATA4
5	0101	W <sub>1</sub> +W <sub>3</sub>	DATA5
6	0110	W <sub>2</sub> +W <sub>3</sub>	DATA6
7	0111	W <sub>1</sub> +W <sub>2</sub> +W <sub>3</sub>	DATA7
8	1000	W <sub>4</sub>	DATA8
9	1001	W <sub>1</sub> +W <sub>4</sub>	DATA9
10	1010	W <sub>2</sub> +W <sub>3</sub>	DATA10
11	1011	W <sub>1</sub> +W <sub>2</sub> +W <sub>4</sub>	DATA11
12	1100	W <sub>3</sub> +W <sub>4</sub>	DATA12
13	1101	W <sub>1</sub> +W <sub>3</sub> +W <sub>4</sub>	DATA13
14	1110	W <sub>2</sub> +W <sub>3</sub> +W <sub>4</sub>	DATA14
15	1111	W <sub>1</sub> +W <sub>2</sub> +W <sub>3</sub> +W <sub>4</sub>	DATA15

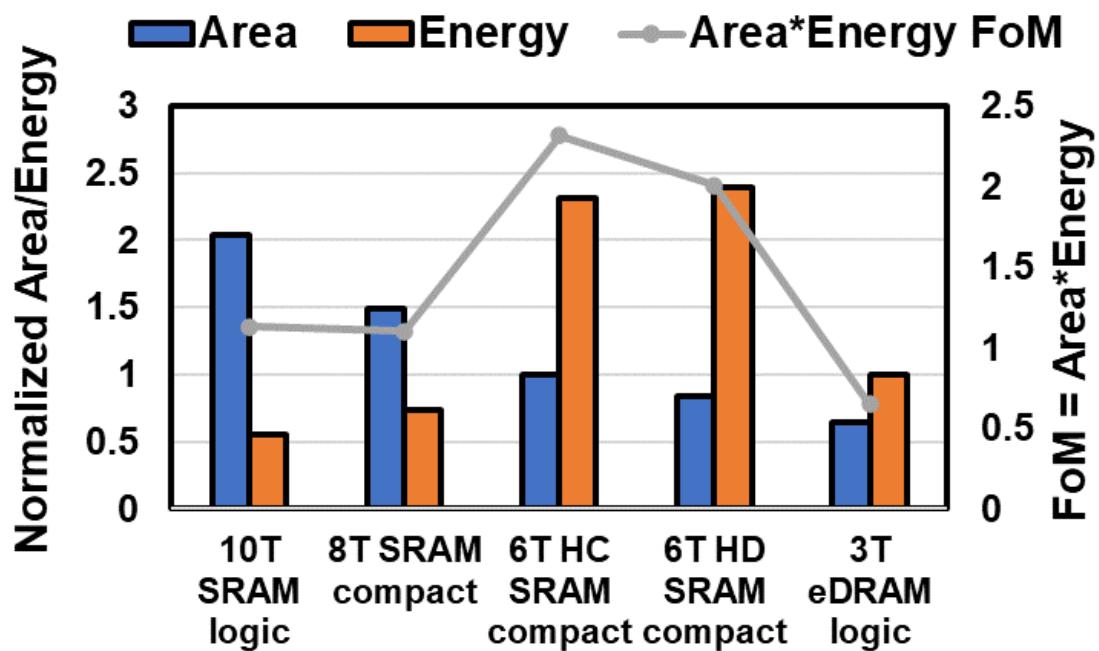
## eLAD improvement



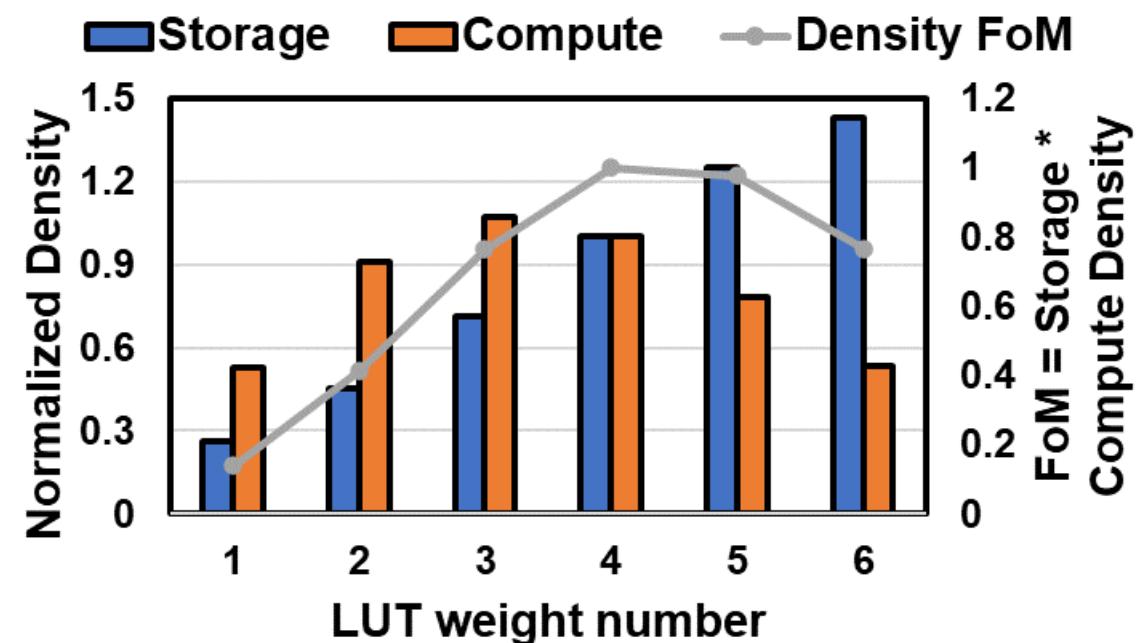
eDRAM-LUT-based CIM saves 40% area compared to conventional DCIM

# Design choice of LUT-based CIM

Area & energy comparison of different LUT cell



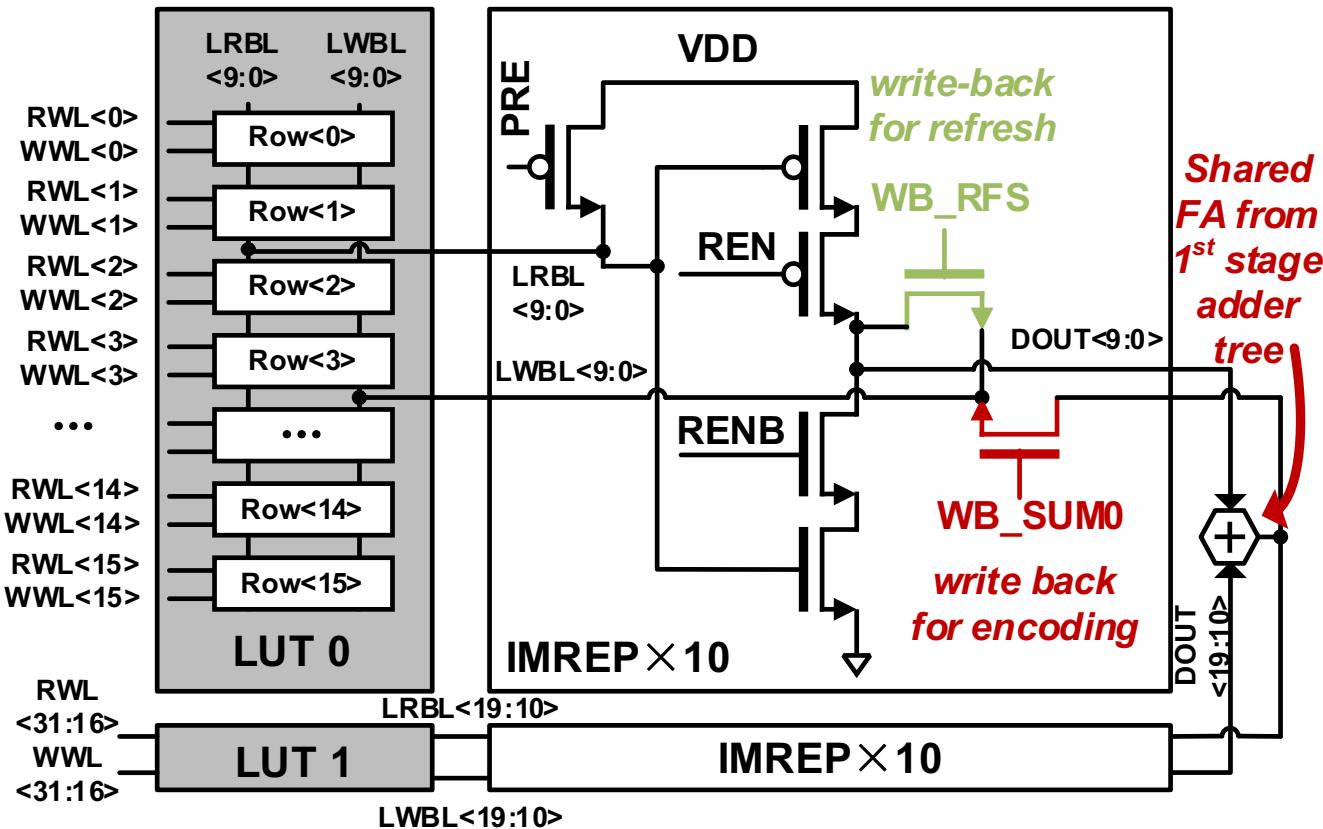
Density trade-off between different numbers of weights



Four-weight eDRAM LUT is a sweet point of density and efficiency trade-off

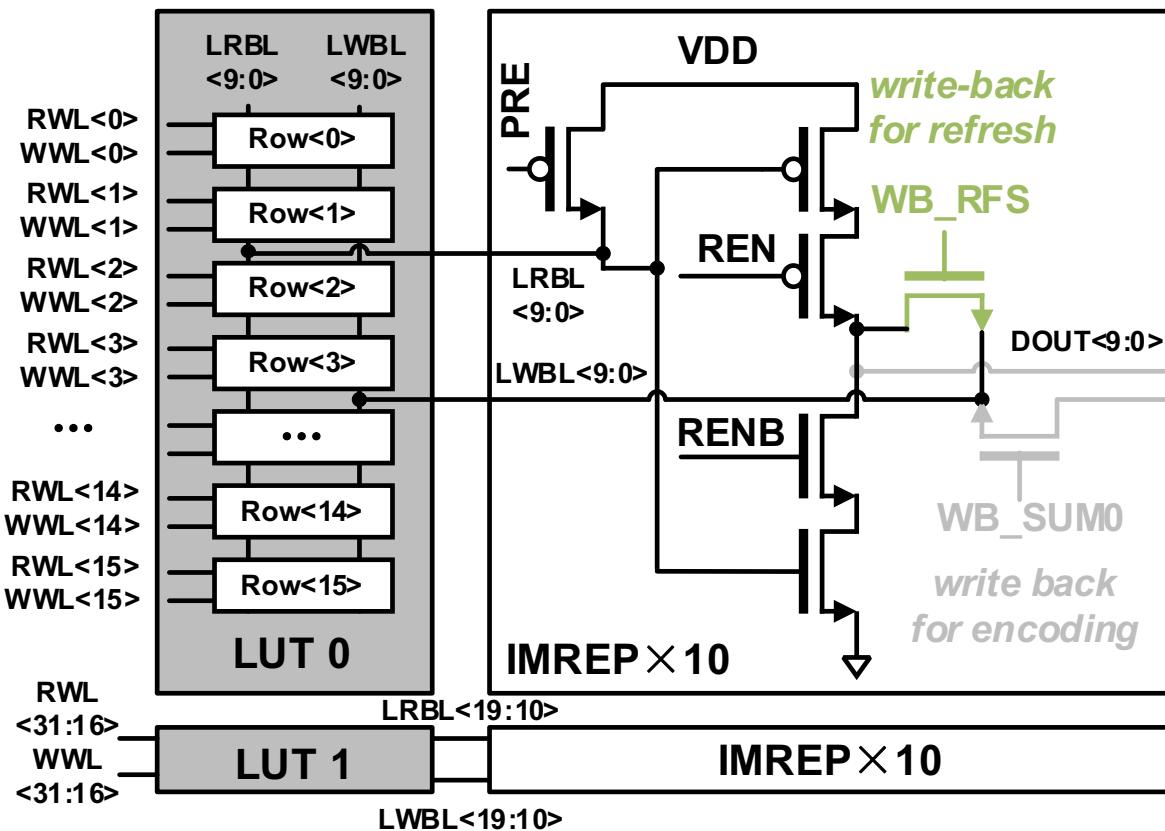
# IMREP implementation details

IMREP schematic

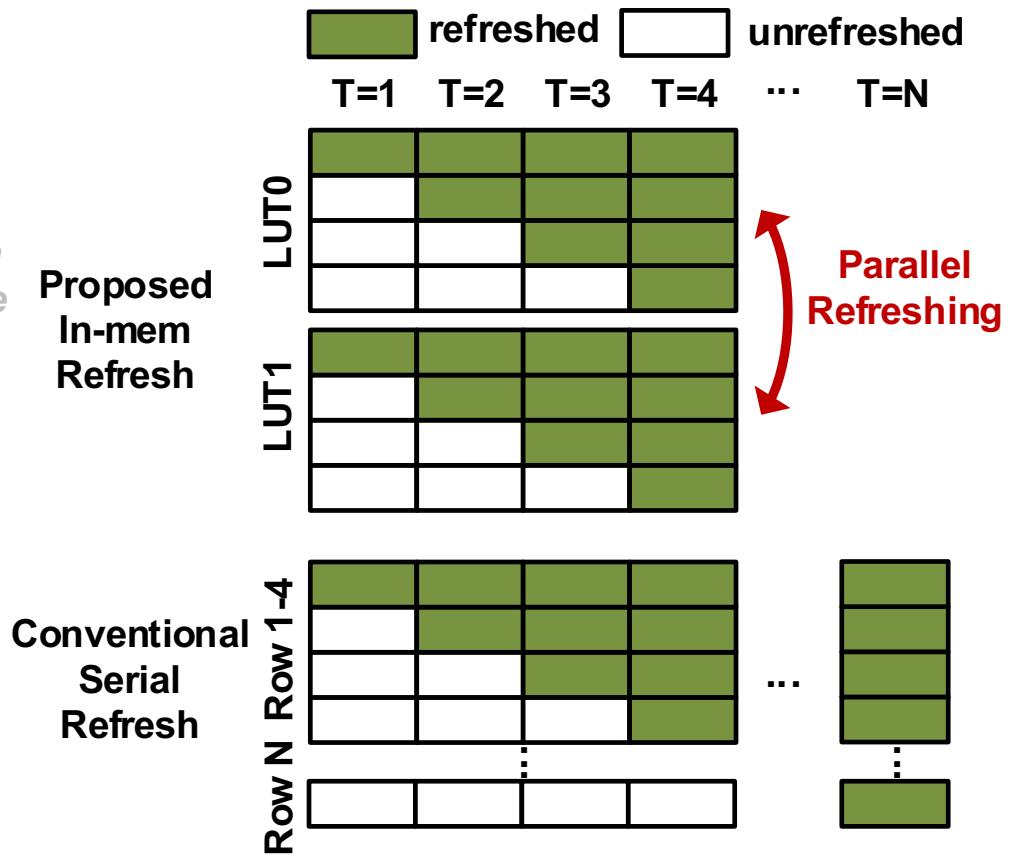


# In-memory refresh

IMREP schematic

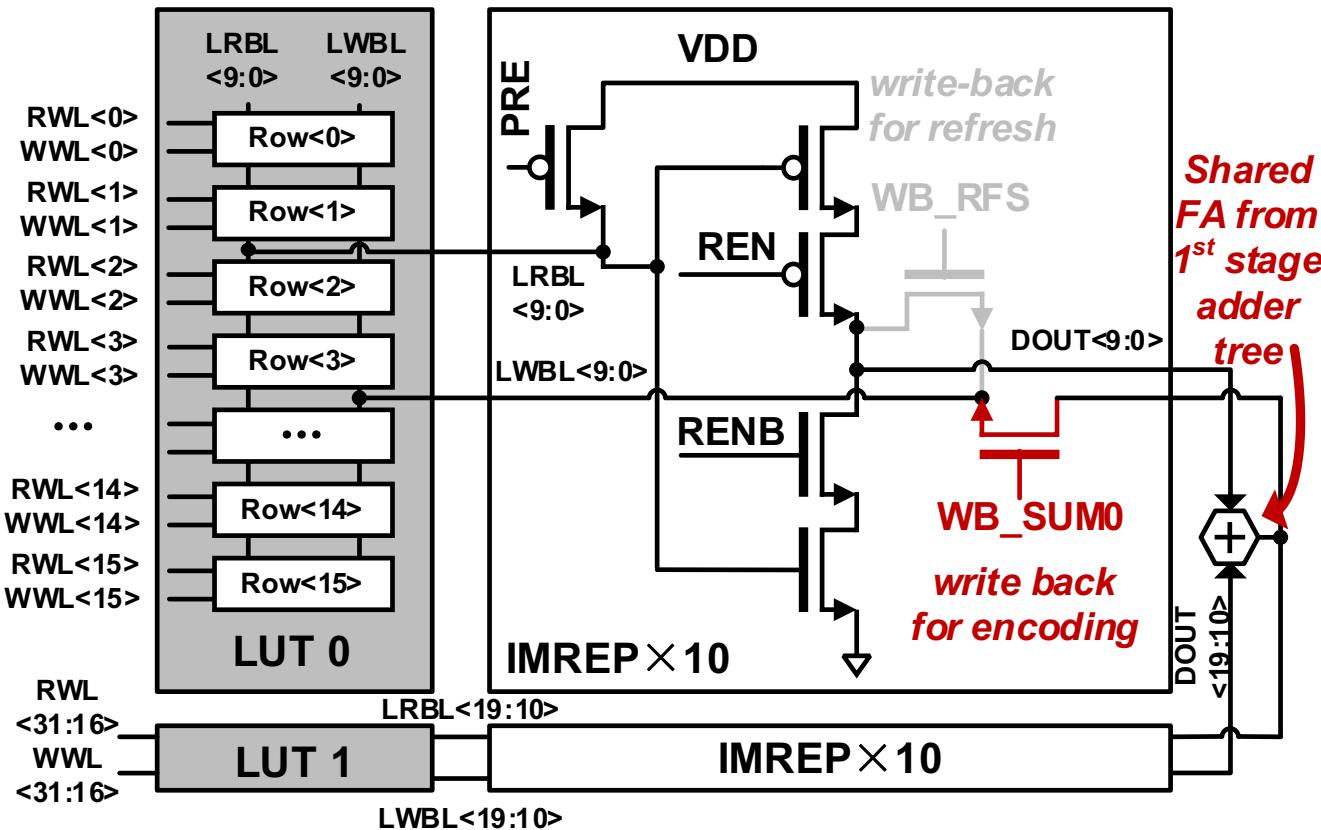


In-memory refresh example

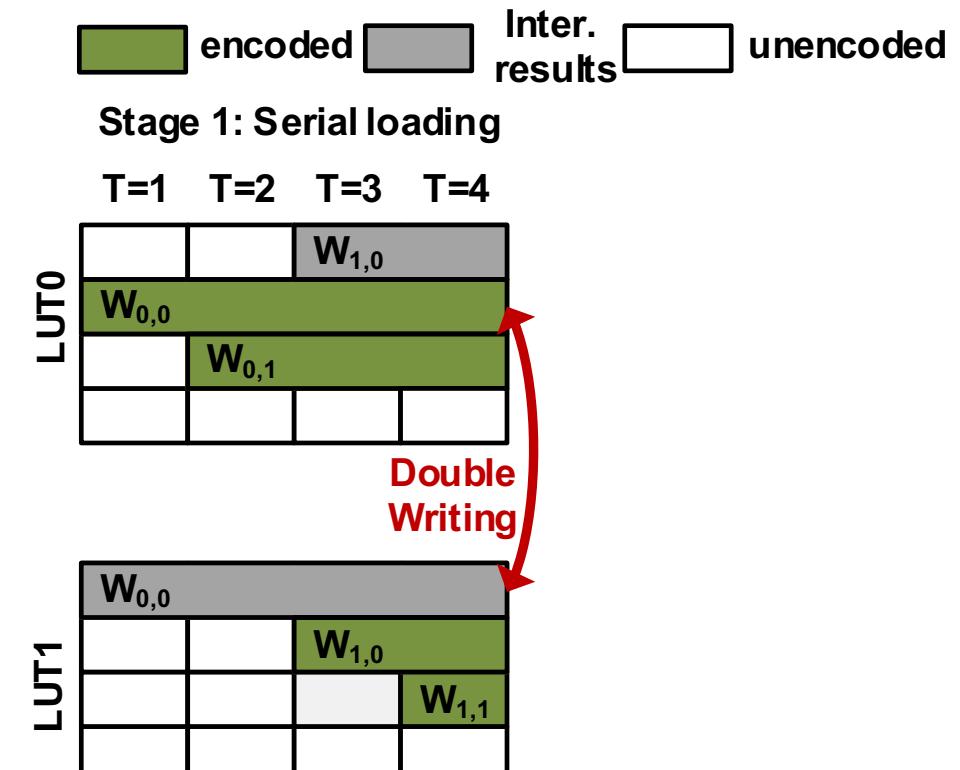


# In-memory encode (1/2)

IMREP schematic

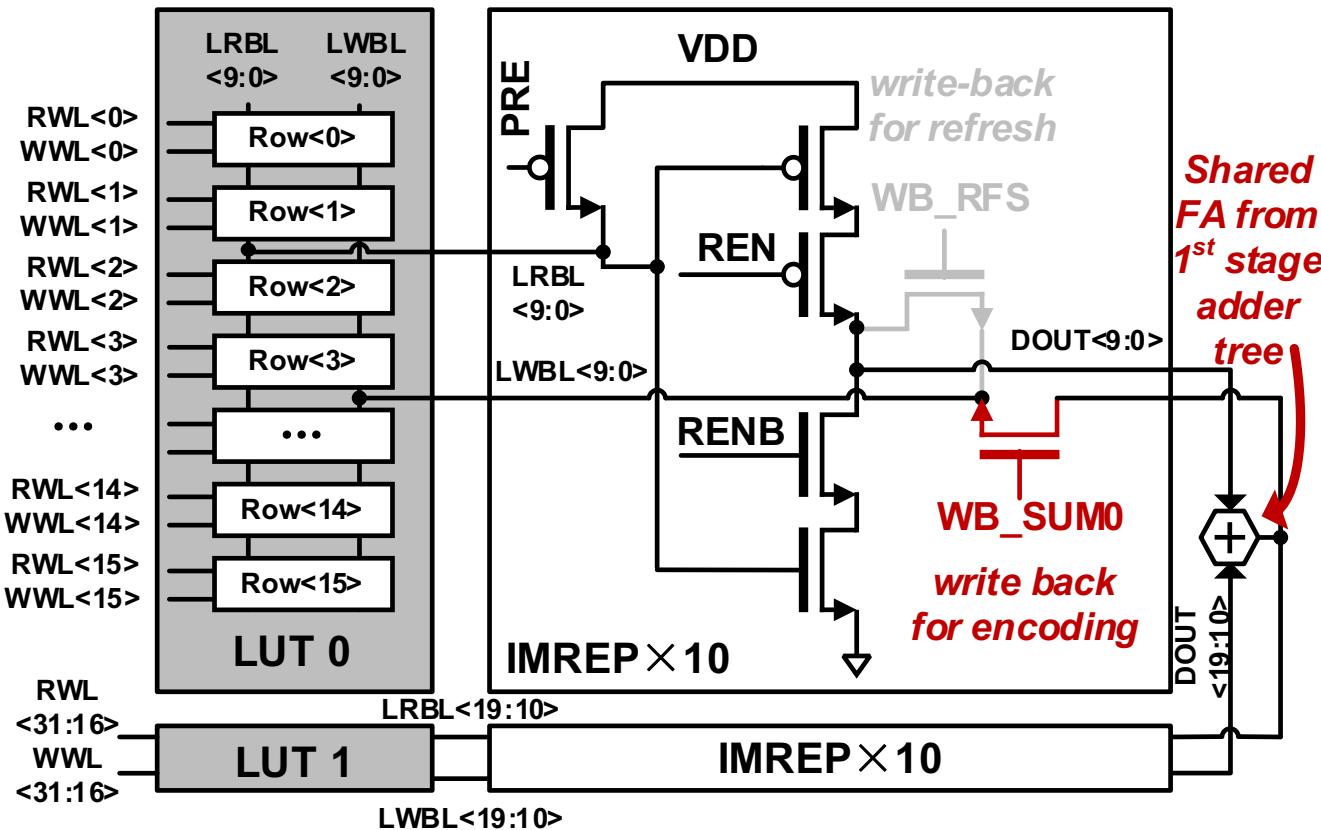


In-memory encode example

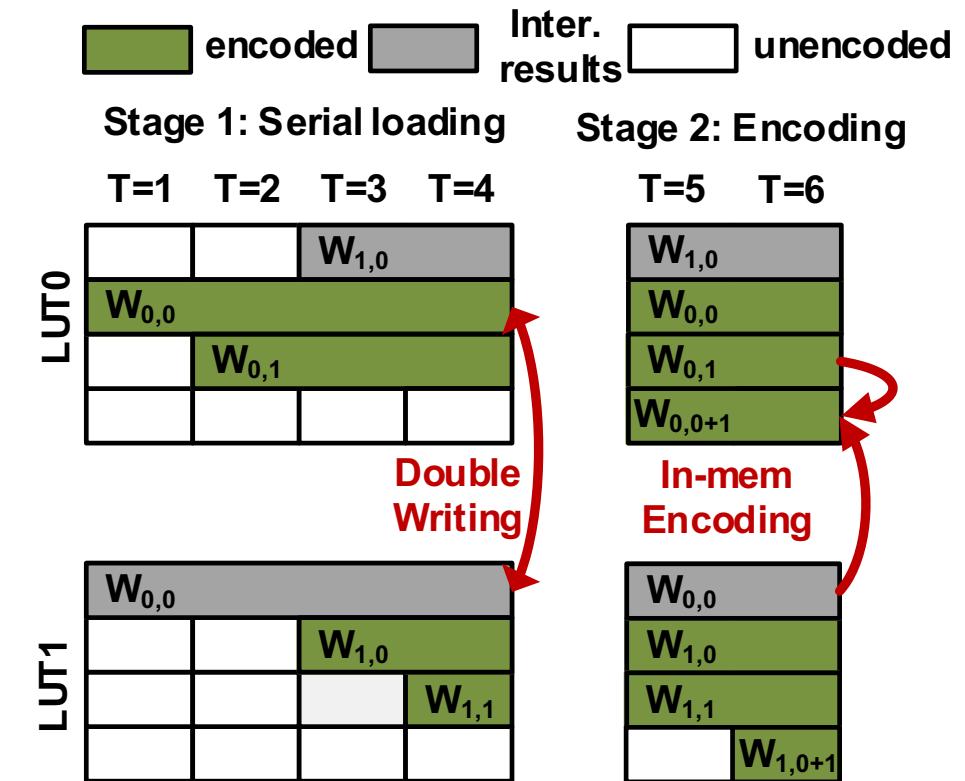


# In-memory encode (2/2)

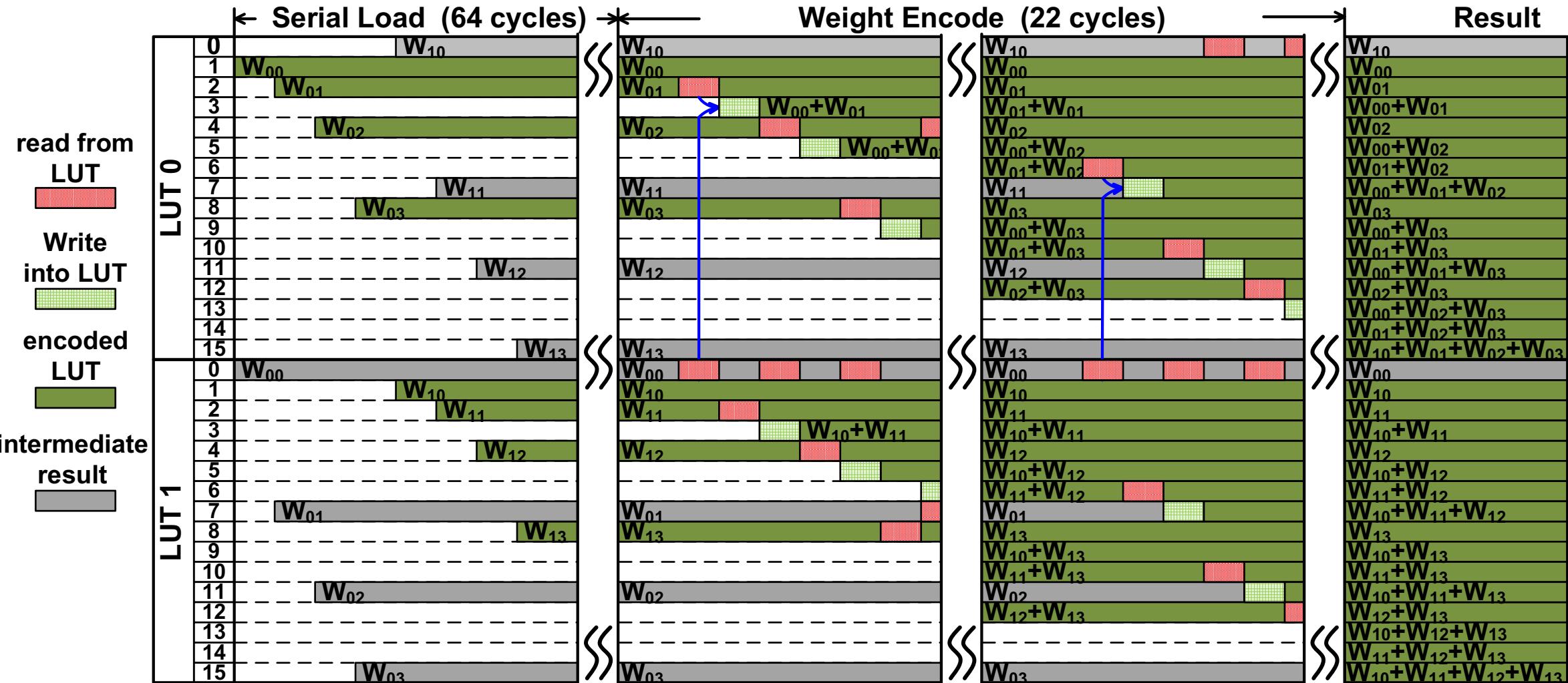
IMREP schematic



In-memory encode example



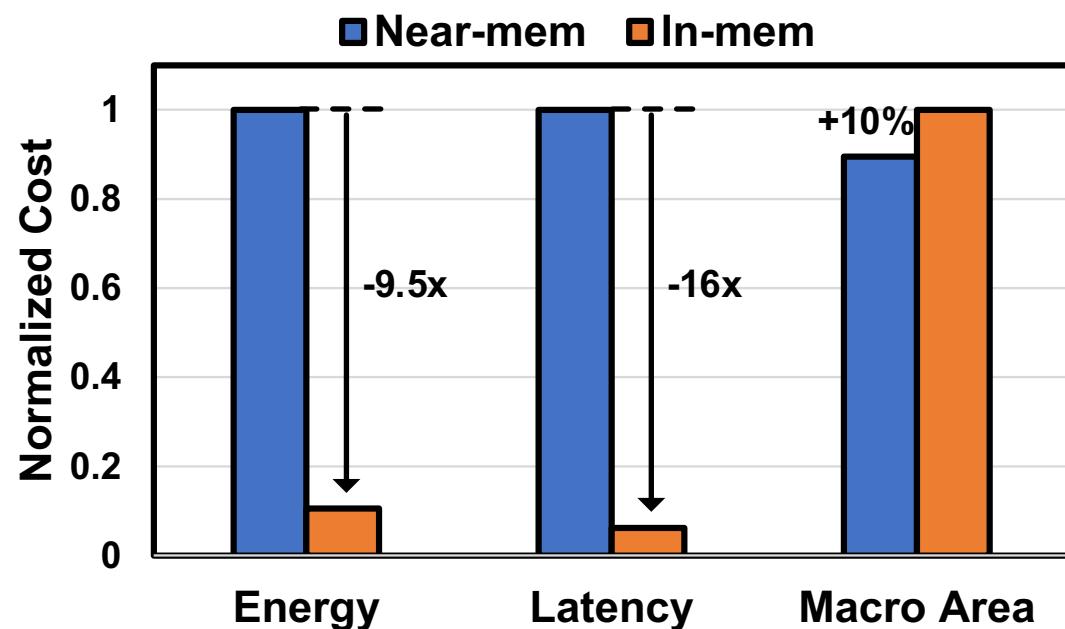
# Complete in-memory encode flow



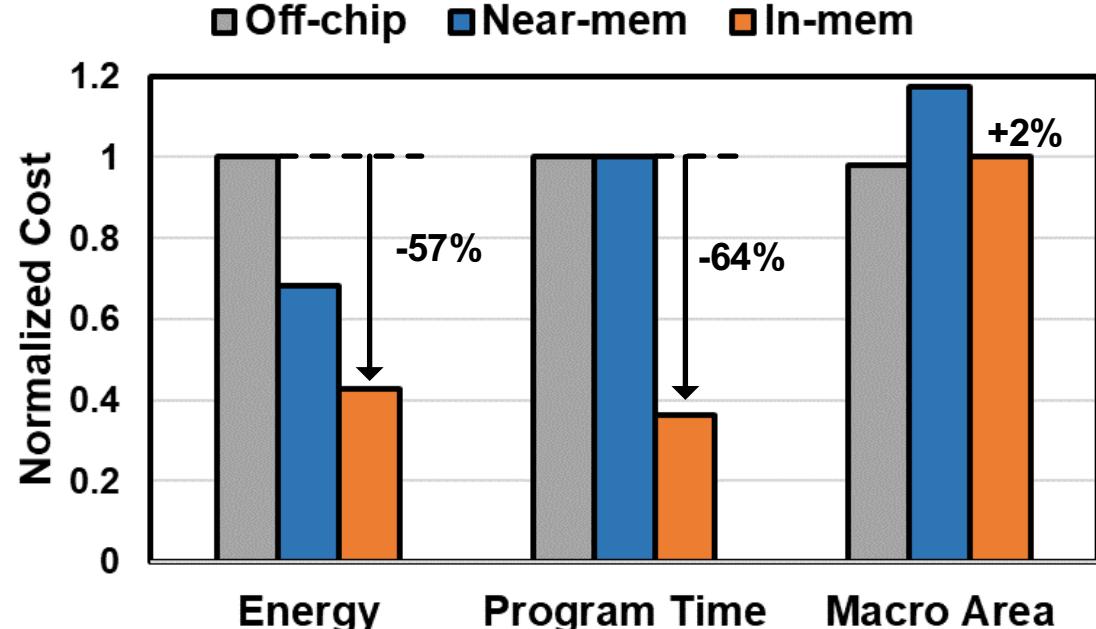
The encoding flow guarantees no read-write conflicts

# IMREP evaluation

## In-mem. refresh improvement



## In-mem. encode improvement



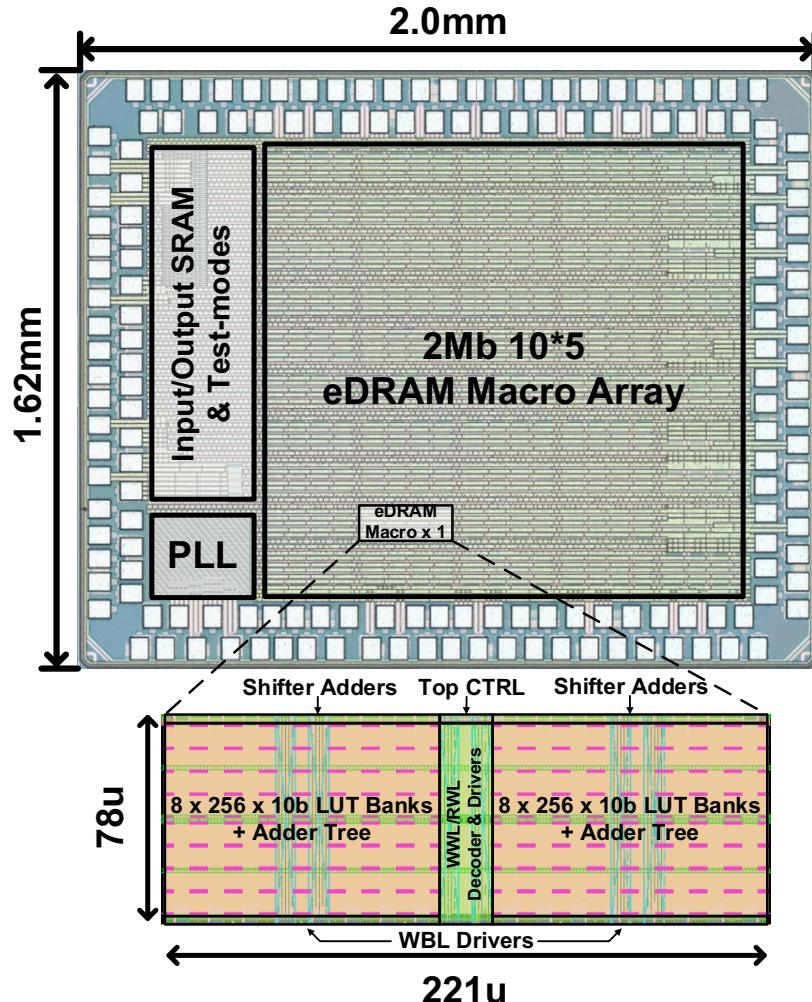
Improved efficiency & throughput by reducing global data movement

# Outline

---

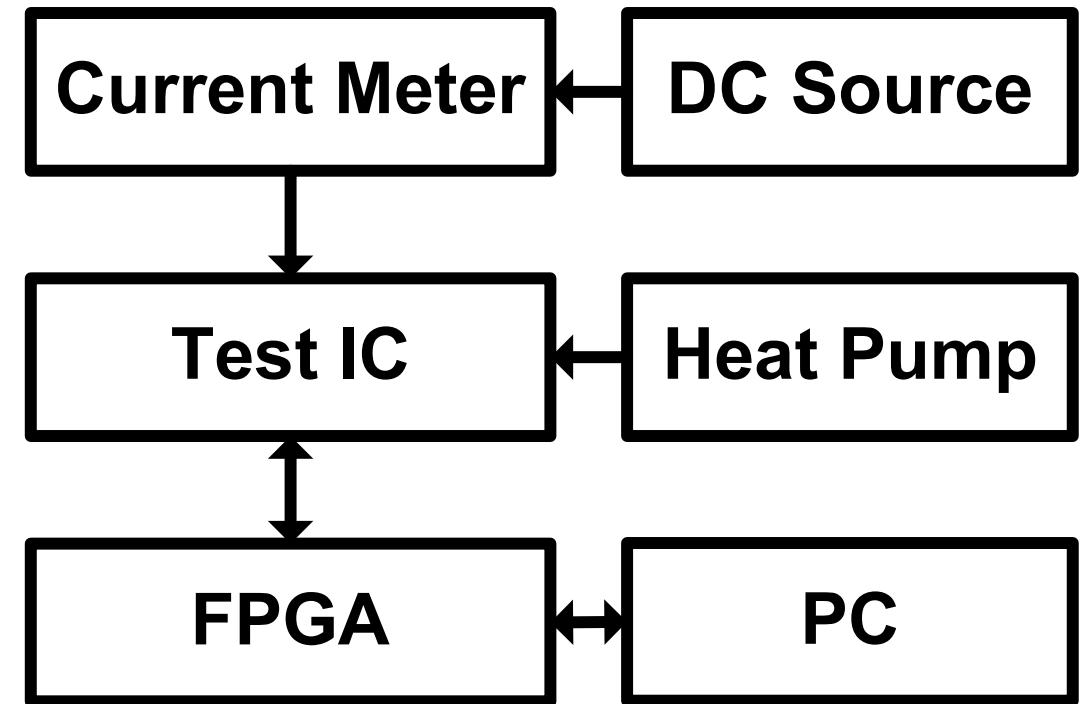
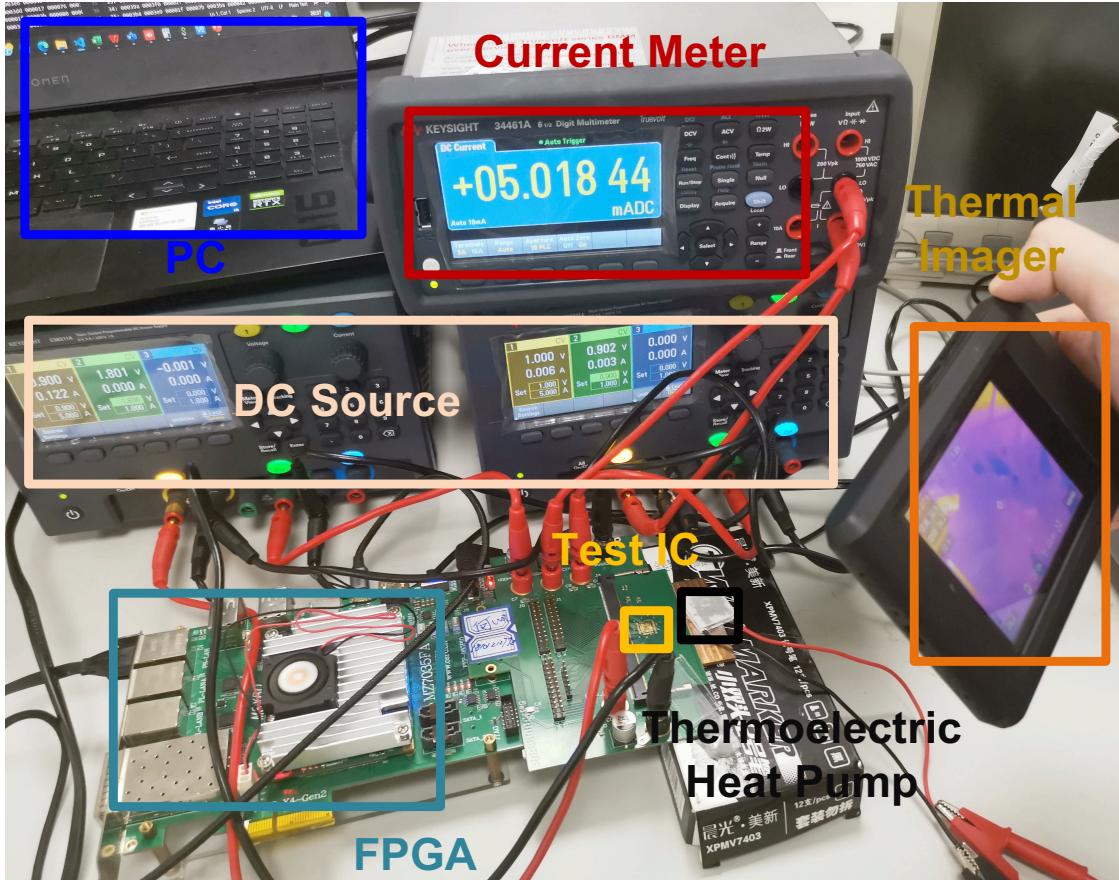
- **Introduction**
- **Challenge of eDRAM-LUT-Based Digital CIM**
- **Proposed eDRAM Digital CIM**
  - Overall architecture and basic principal
  - Two-stage eDRAM-LUT-based adder tree
  - In-memory refreshing and LUT encoding
- **Measurement Results**
- **Conclusion**

# Chip photograph



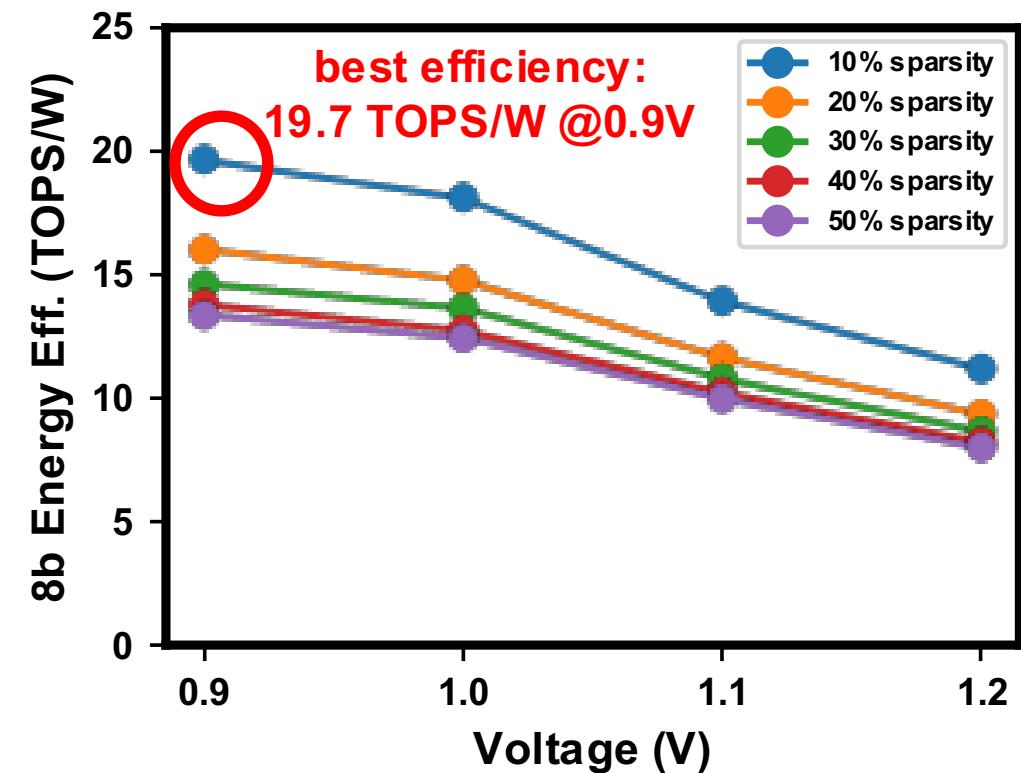
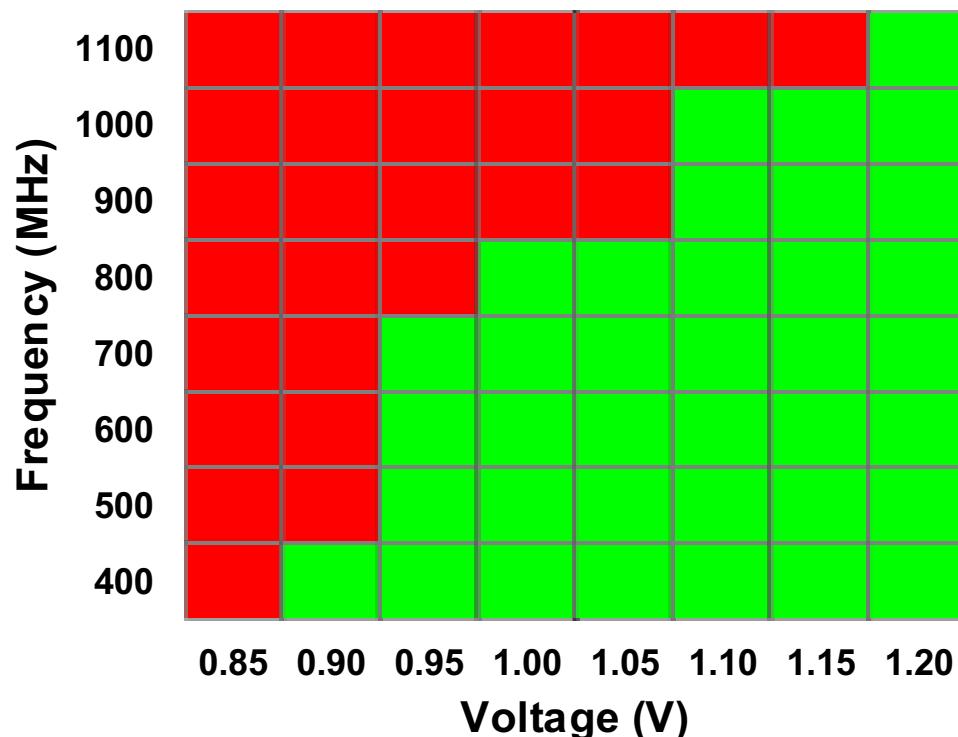
Chip Summary	
Technology	28nm HKMG
Bit Cell	3T eDRAM
Cell Area	0.145 um <sup>2</sup>
Macro Area	0.017 mm <sup>2</sup>
Macro Capacity	40Kb
Total Capacity	2Mb
Total Throughput	10TOPS
Compute Circuit	LUT + Adder Tree
Input Channel	64
Output Channel	16
Supply Voltage	0.9-1.2V
Frequency	400-1100MHz
Storage Density	2.4Mb/mm <sup>2</sup>
Normalized compute Density	2684F <sup>2</sup> /b
Energy Efficiency	19.7 @ 8b, 0.9V
Area Efficiency	16.7 TOPS/mm <sup>2</sup> @8b, 1.2V

# Test platform

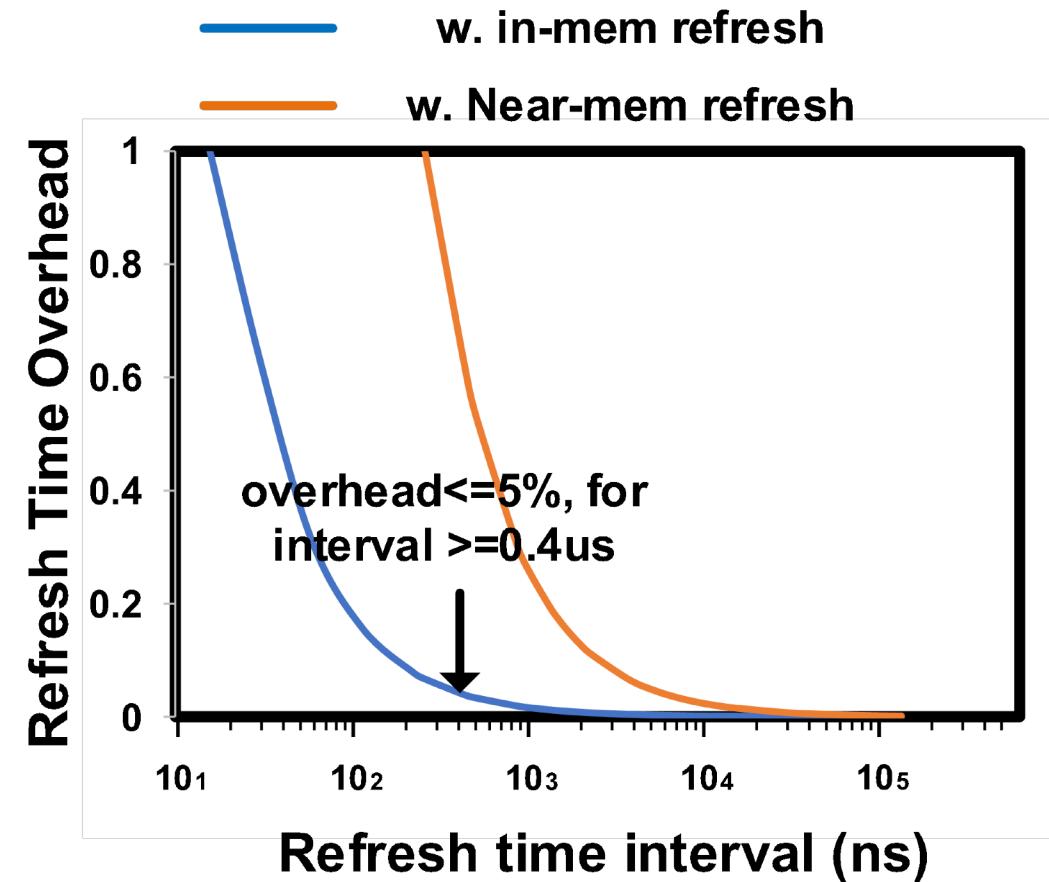
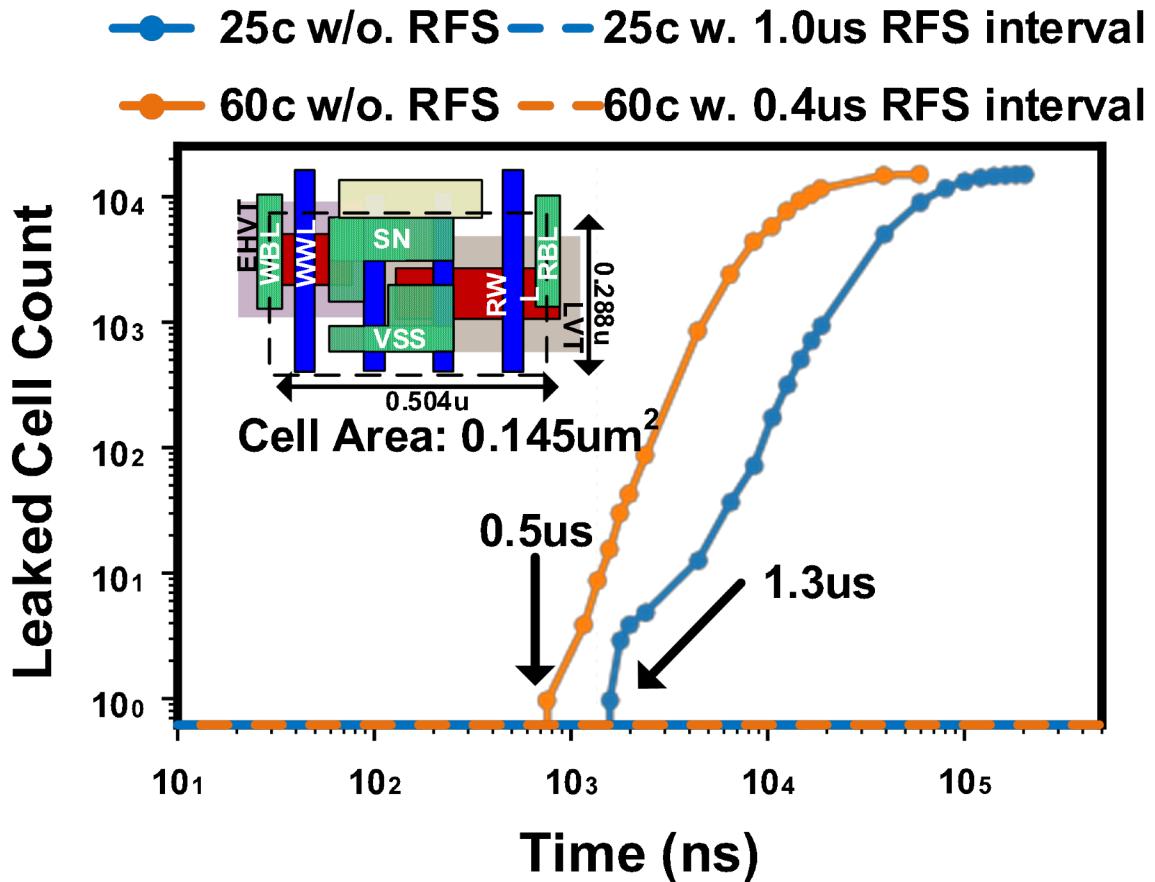


# Voltage & efficiency scaling

- Best efficiency achieved @ 0.9V, 400MHz
- Best performance achieved @ 1.2V, 1100MHz

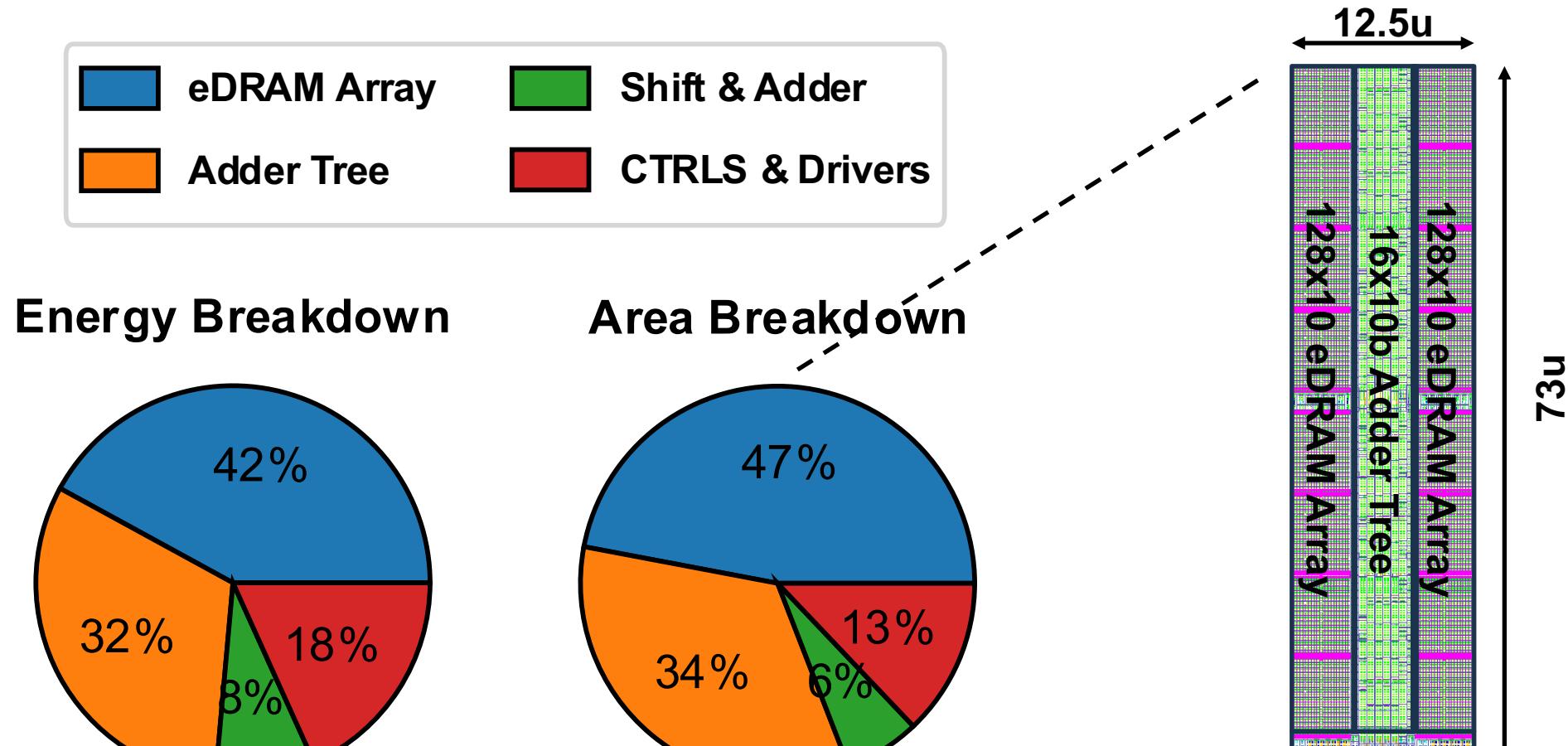


# eDRAM retention test



Minimal refreshing overhead with in-memory acceleration

# Power and area breakdown



Memory circuits dominate both area and energy breakdown

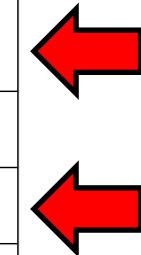
# Comparison with state-of-the-arts CIM macros

	ISSCC'21 16.2	ISSCC'21 15.3	ISSCC'23 16.5	This Work	ISSCC'21 16.4	ISSCC'22 16.1	VLSI'22 C2.5
Technology	65nm	65nm	28nm	28nm	22nm	28nm	12nm
MAC Operation	Analog Charge	Analog Current	Analog Charge	Digital Memory-LUT	Digital Adder tree	Digital Approximate	Digital Logic-LUT
Bit Cell	1T1C eDRAM	3T eDRAM	4T eDRAM	3T eDRAM	6T SRAM	8T SRAM	6T SRAM
Array Size	16Kb	8Kb	100Kb	40Kb	64Kb	16Kb	8Kb
Macro Area (mm <sup>2</sup> )	0.57	0.05	0.103	0.0170	0.202	0.033	0.0323
Storage Den. (Mb/mm <sup>2</sup> )	0.03	0.16	0.95	2.4	0.32	0.50	0.25
8b TOPS/W @0.9-1.1V	4.76 (@1.1V)	54.8 (@1V)	30.1 (@1V)	18.1 (@1V)	11.7 (@0.92V)	17.3 (@0.9V)	17.9 (@0.92V)
8b Peak TOPS/mm <sup>2</sup>	0.008 (@1.1V)	2.2 (@1V)	2.03 (@1V)	16.2 (@1.2V)	8.3 (@0.92V)	9.5 (@1.1V)	14.3 (@0.92V)
Full Precision ?	No	No	No	Yes	Yes	No	Yes
In-memory rfs. ?	No	No	No	Yes	Not applicable	Not applicable	Not applicable
Efficiency FoM <sup>†</sup>	0.038	120.6	61.1	293.2	124.5	164.3	256.0
Density FoM <sup>‡</sup>	0.00024	0.352	1.92	38.9	2.7	4.75	2.6

\*: 8b TOPS/W @0.9-1.1V × Output Bits / (Input Bits + Weight Bits + log<sub>2</sub> (# of Input Channels))

†: 8b TOPS/W @0.9-1.1V × 8b Peak TOPS/mm<sup>2</sup>

‡: 8b Peak TOPS/mm<sup>2</sup> × Storage Density



Exceeding  
Storage &  
Compute  
Density @  
28nm

# Outline

---

- **Introduction**
- **Challenge of eDRAM-LUT-Based Digital CIM**
- **Proposed eDRAM Digital CIM**
  - Overall architecture and basic principal
  - Two-stage eDRAM-LUT-based adder tree
  - In-memory refreshing and LUT encoding
- **Measurement Results**
- **Conclusion**

# Conclusion

---

- **Digital CIM has unique advantages on performance, robustness and scalability**
- **Previous DCIM works mainly optimize on computation logic instead of taking memory and computation as a whole**
- **The eDRAM-LUT-based CIM allow us to achieve 16.7TOPS/mm<sup>2</sup> 8b compute density with 2.4Mb/mm<sup>2</sup> storage density**
- **Potential improvement can be achieved by combining other high-density emerging devices in the future**



Please Scan to Rate  
This Paper



# A 22nm 16Mb Floating-Point ReRAM Compute-in-Memory Macro with 31.2TFLOPS/W for AI Edge Devices

Tai-Hao Wen<sup>\*1</sup>, Hung-Hsi Hsu<sup>\*1,2</sup>, Win-San Khwa<sup>\*2</sup>, Wei-Hsing Huang<sup>1</sup>, Zhao-En Ke<sup>1</sup>, Yu-Hsiang Chin<sup>1</sup>, Hua-Jin Wen<sup>1</sup>, Yu-Chen Chang<sup>1</sup>, Wei-Ting Hsu<sup>1</sup>, Chung-Chuan Lo<sup>1</sup>, Ren-Shuo Liu<sup>1</sup>, Chih-Cheng Hsieh<sup>1</sup>, Kea-Tiong Tang<sup>1</sup>, Shih-Hsin Teng<sup>3</sup>, Chung-Cheng Chou<sup>3</sup>, Yu-Der Chih<sup>3</sup>, Tsung-Yung Jonathan Chang<sup>3</sup>, Meng-Fan Chang<sup>1,2</sup>

<sup>1</sup>National Tsing Hua University (NTHU), Hsinchu, Taiwan,

<sup>2</sup>TSMC Corporate Research, Hsinchu, Taiwan

<sup>3</sup>TSMC, Hsinchu, Taiwan



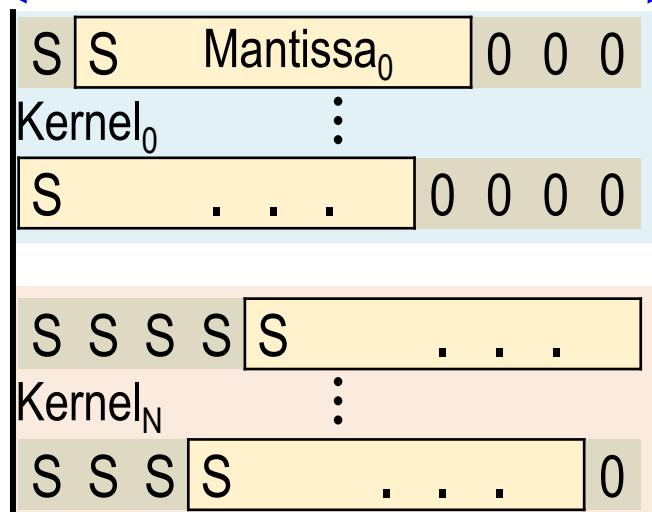
# Outline

- **Challenges of floating-point (FP) nonvolatile compute-in-memory (nvCIM) for AI edge devices**
- **Proposed schemes in nonvolatile CIM macro**
  - Overview of floating-point ReRAM-CIM macro
  - FP computing flow with kernel-wise weight pre-alignment (K-WPA)
  - Rescheduled multi-bit input compression (RS-MIC)
  - HRS-favored dual-sign-bit (HF-DSB) weight encoding
- **Performance and measurement results**
- **Conclusion**

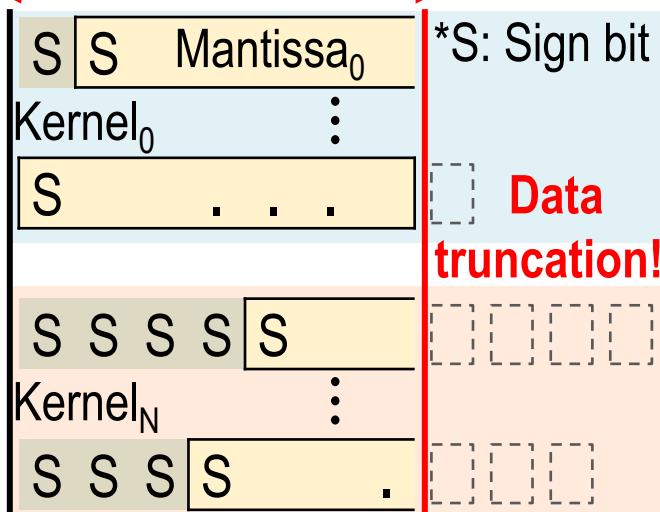
# Challenges of Nonvolatile Compute-In-Memory (1/3)

## Conv. Layer-wise Weight Pre-Alignment (PA)

Bit width for PA = 12b



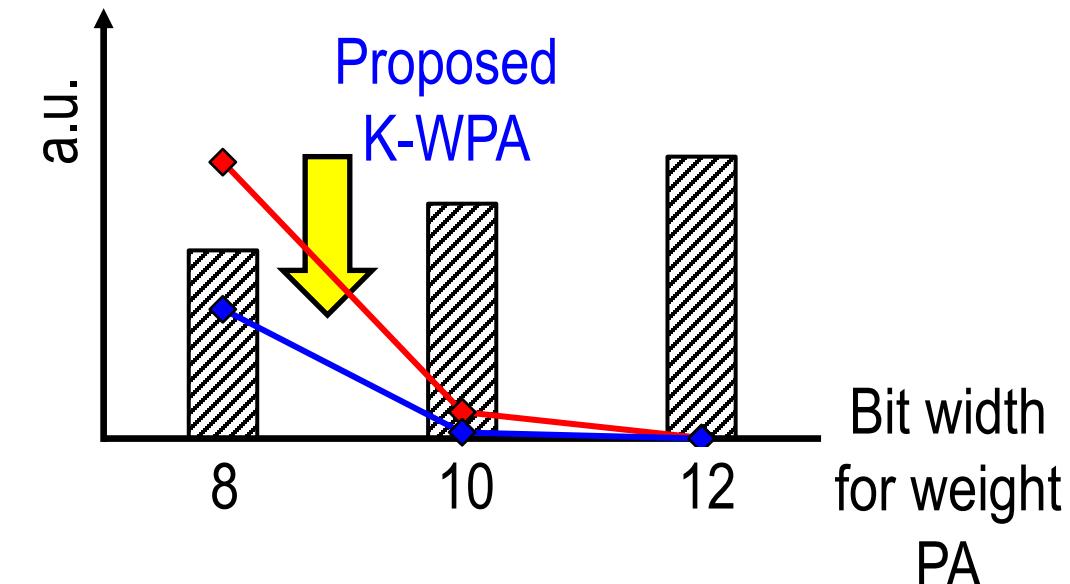
Bit width for PA = 8b



Storage

Accuracy loss (Conv.)

Accuracy loss (Prop.)

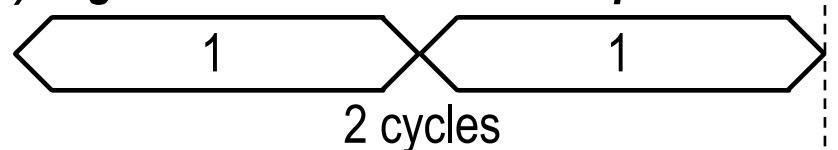


- Tradeoff of bit width for weight pre-alignment between **inference accuracy** and **weight storage**
  - *Increasing bit width enhances inference accuracy*
  - *Decreasing bit width reduces storage needs*

# Challenges of Nonvolatile Compute-In-Memory (2/3)

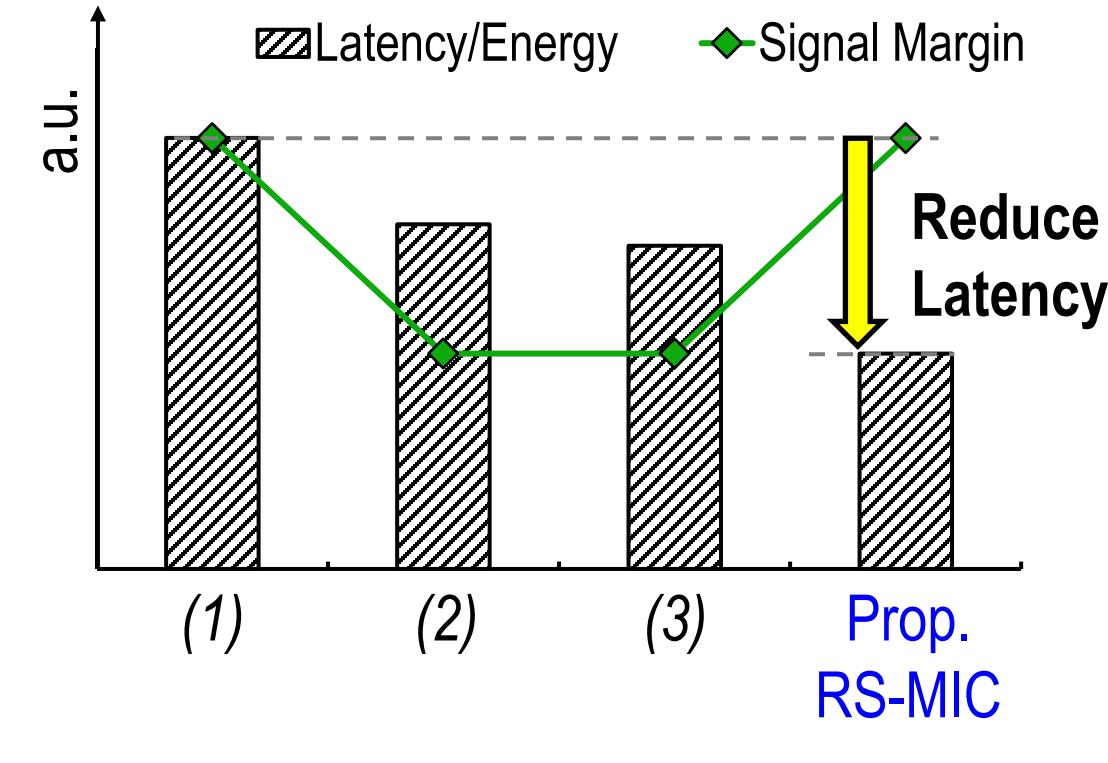
An Example of a 2b input:

(1) Digital Bit-Serial Multi-bit Input

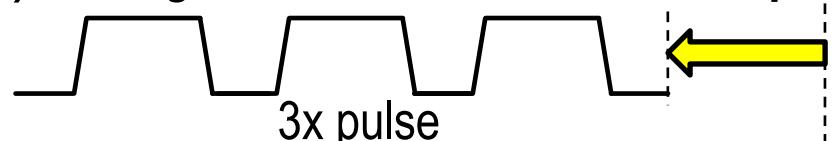


Example: 2b IN = 3

Latency/Energy  
Signal Margin

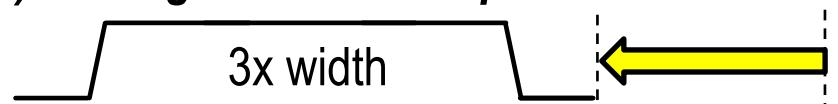


(2) Analog WL Pulse Count Multi-bit Input



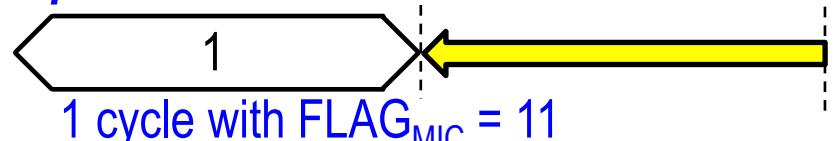
Latency/Energy  
Signal Margin

(3) Analog Decoded WL pulse-width Multi-bit Input



Latency/Energy  
Signal Margin

Proposed RS-MIC

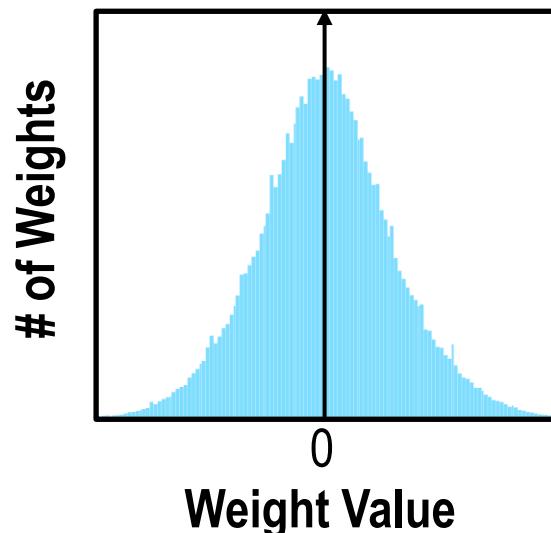


Latency/Energy  
Signal Margin

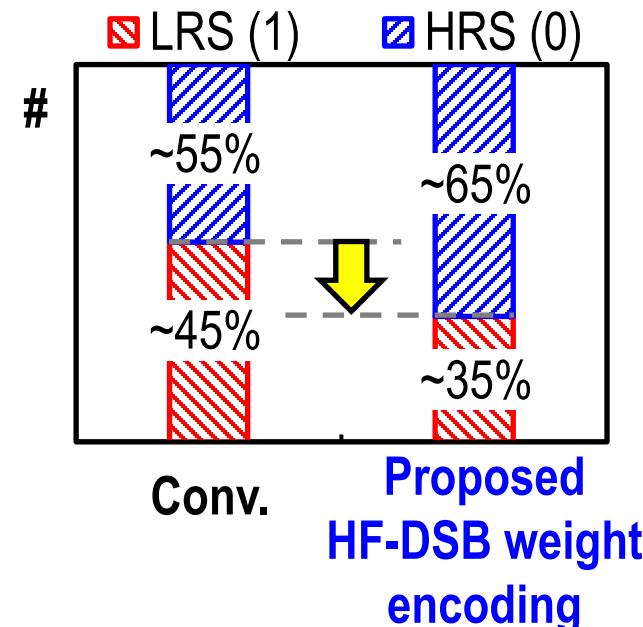
- Long latency/energy of MAC operations due to high input bit width in floating-point format
  - Digital bit-serial multi-bit inputs suffer from long latency/energy
  - Analog multi-bit inputs suffer from low signal margin

# Challenges of Nonvolatile Compute-In-Memory (3/3)

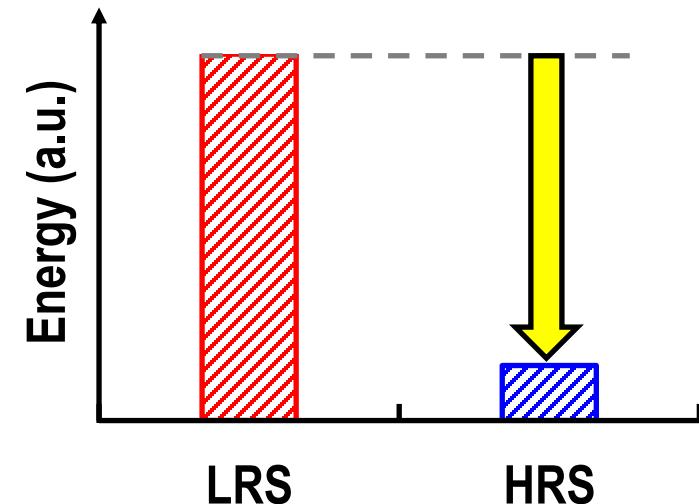
Weight distribution after pre-alignment



Weight Storage in ReRAM Array



Energy of accessing one bit cell



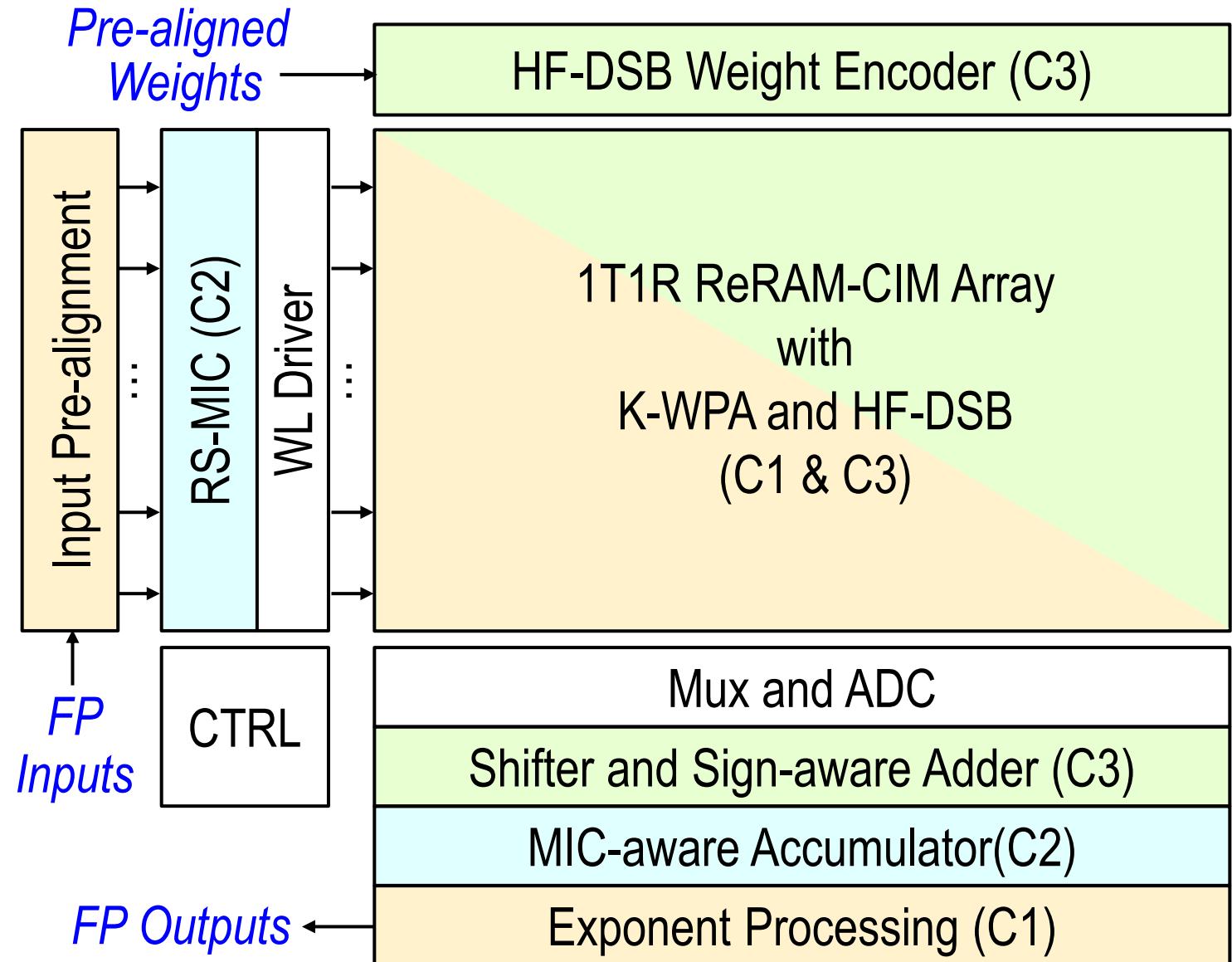
- Large current consumption of cell array due to large number of memory cell for FP weight storage
  - LRS cells consume much larger energy than HRS cells
  - Proposed HF-DSB weight encoding reduce the ratio of LRS cells

# Outline

- Challenges of floating-point (FP) nonvolatile compute-in-memory (nvCIM) for AI edge devices
- Proposed schemes in nonvolatile CIM macro
  - Overview of floating-point ReRAM-CIM macro
  - FP computing flow with kernel-wise weight pre-alignment (K-WPA)
  - Rescheduled multi-bit input compression (RS-MIC)
  - HRS-favored dual-sign-bit (HF-DSB) weight encoding
- Performance and measurement results
- Conclusion

# Overview of floating-point ReRAM-CIM macro

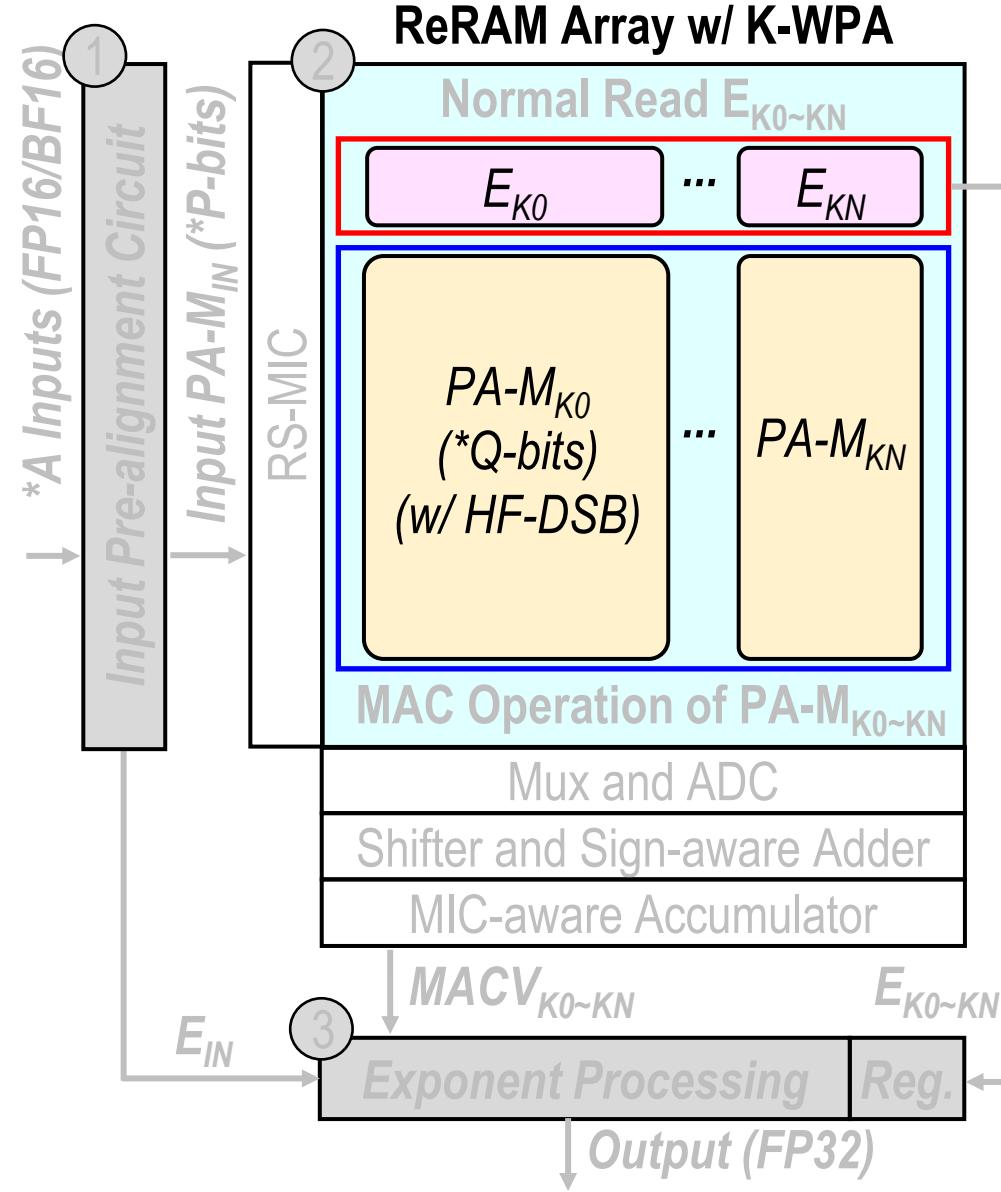
- FP computing flow with kernel-wise weight pre-alignment (K-WPA)
  - Reduce accuracy loss during the data truncation in weight pre-alignment
- Rescheduled multi-bit input compression (RS-MIC)
  - Reduce MAC energy and latency with lossless compression
- HRS-favored dual-sign-bit (HF-DSB) weight encoding
  - Reduce ReRAM array current consumption



# Outline

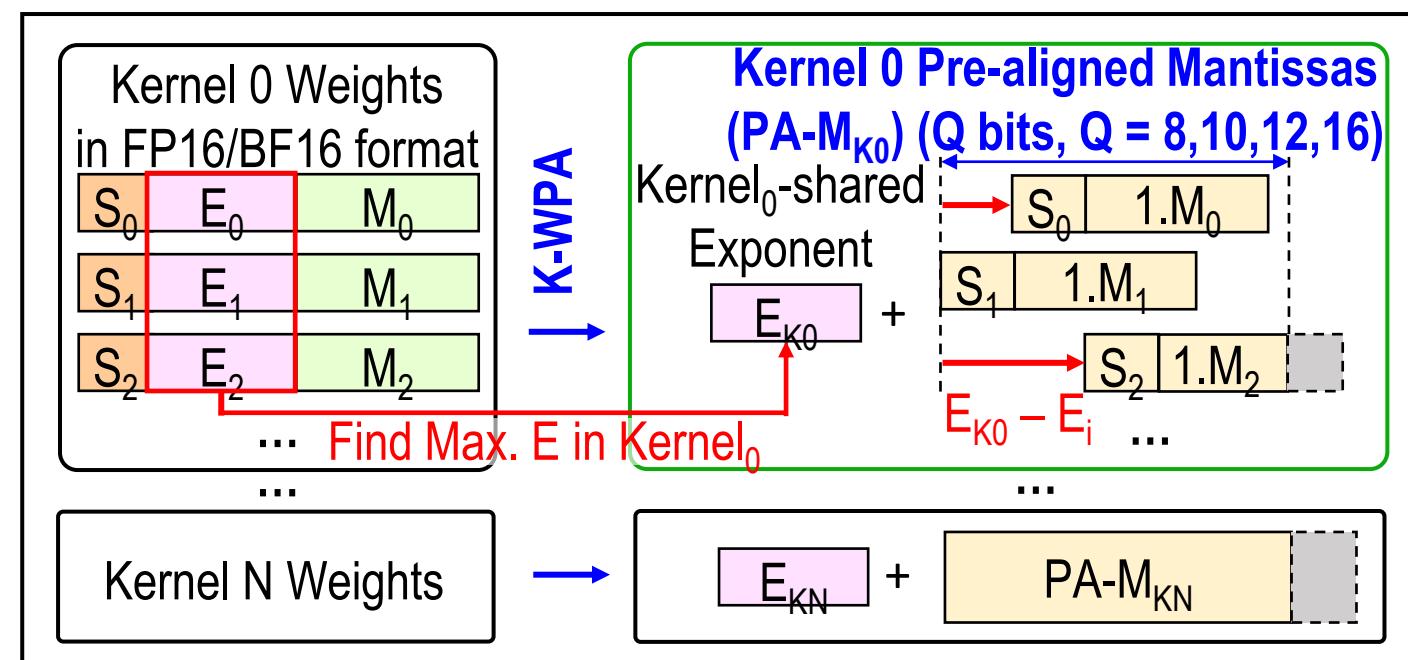
- Challenges of floating-point (FP) nonvolatile compute-in-memory (nvCIM) for AI edge devices
- Proposed schemes in nonvolatile CIM macro
  - Overview of floating-point ReRAM-CIM macro
  - FP computing flow with kernel-wise weight pre-alignment (K-WPA)
  - Rescheduled multi-bit input compression (RS-MIC)
  - HRS-favored dual-sign-bit (HF-DSB) weight encoding
- Performance and measurement results
- Conclusion

# FP computing flow with kernel-wise weight pre-alignment (1/7)

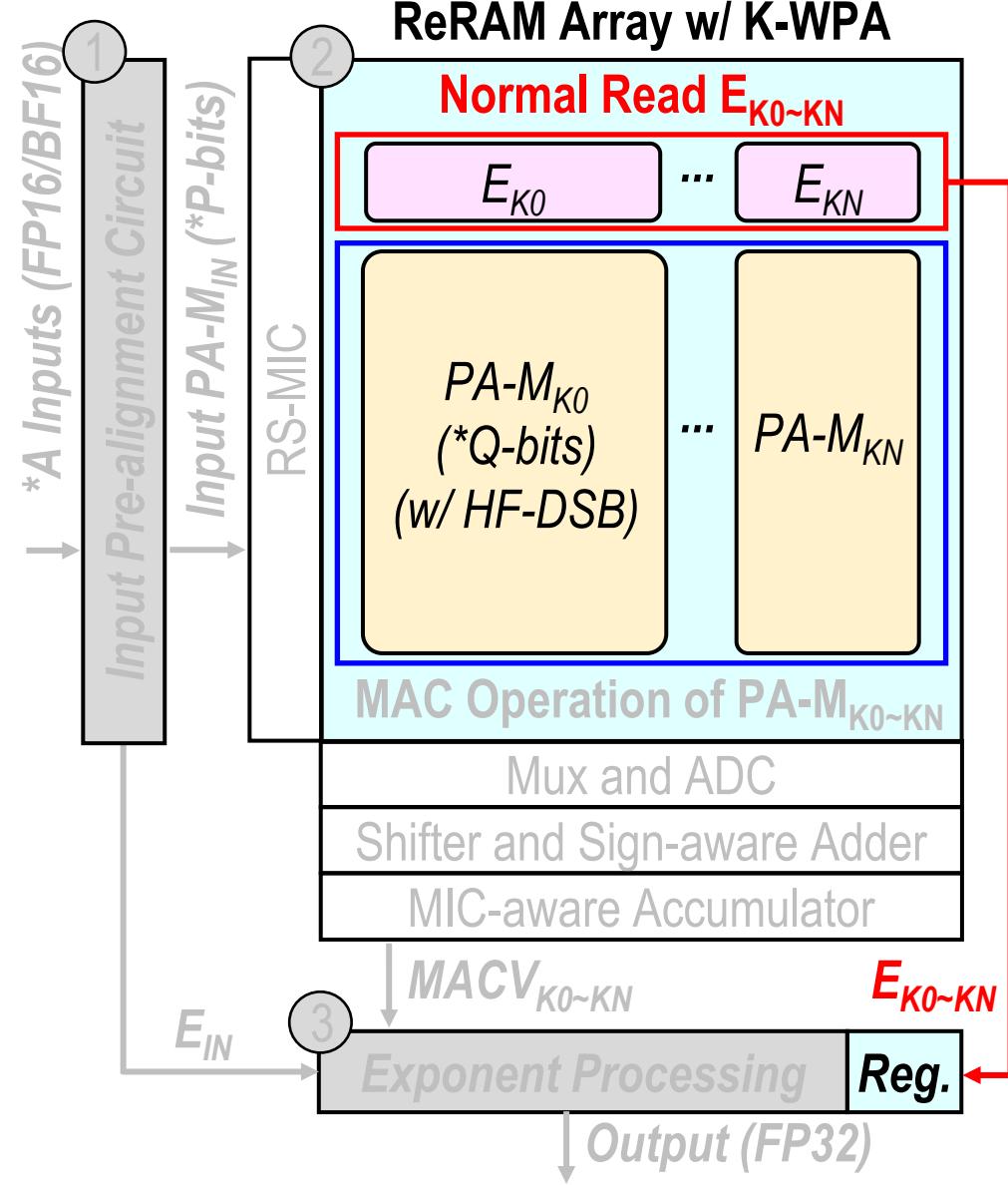


## Kernel-wise weight pre-alignment (K-WPA)

- Offline alignment process to eliminate need for on-chip weight alignment
- Find the maximum exponent ( $E_K$ ) in each kernel
- Aligns each weight's sign and mantissa based on its exponent difference ( $E_K - E_i$ )

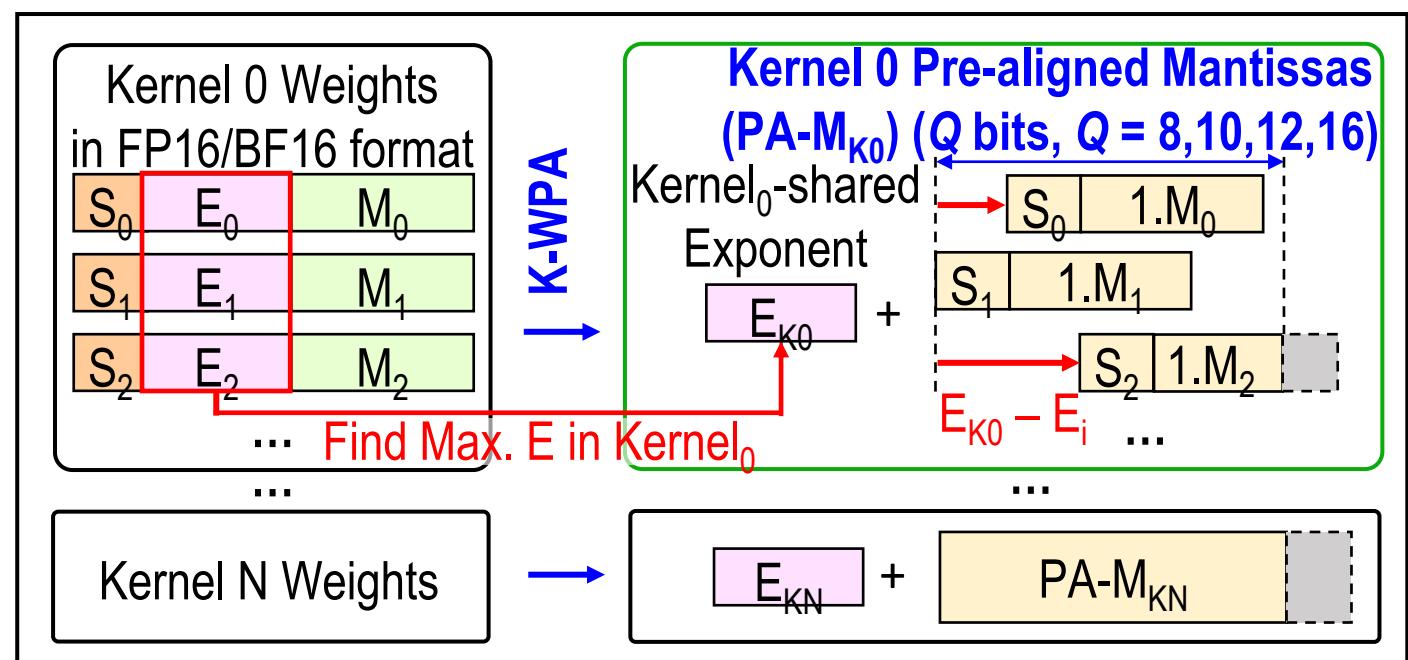


# FP computing flow with kernel-wise weight pre-alignment (2/7)



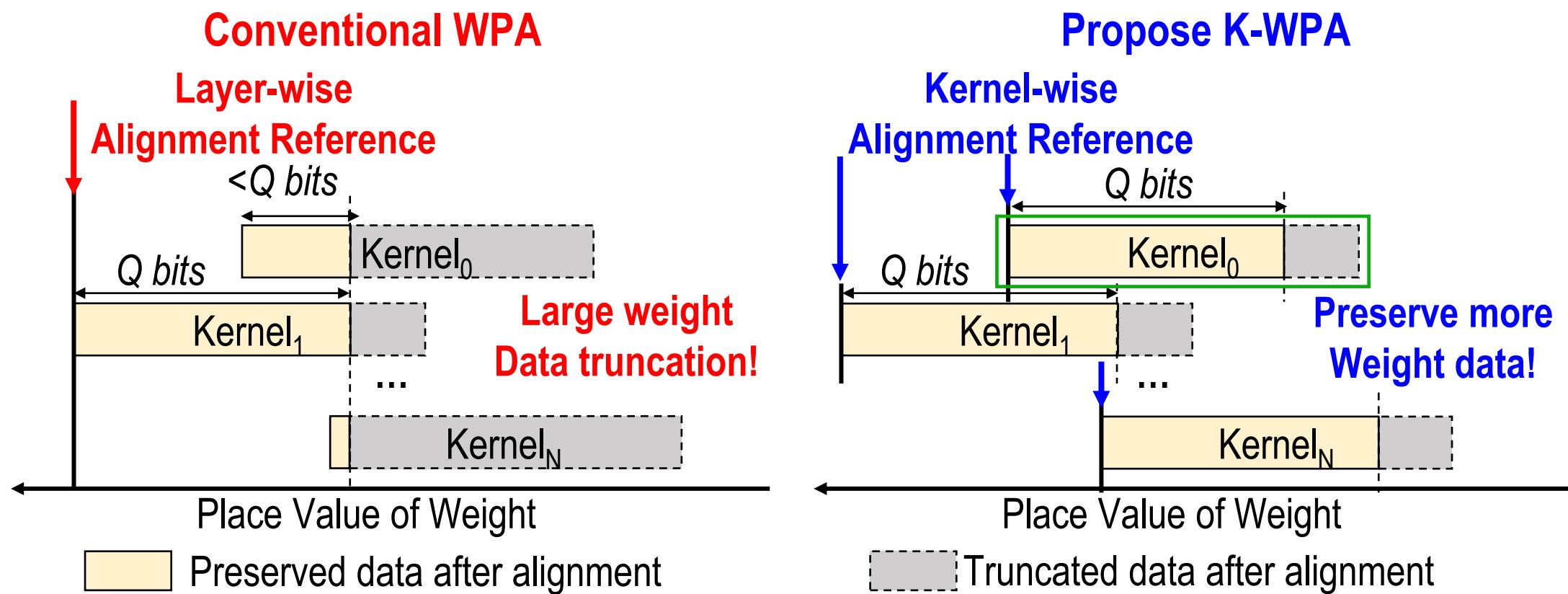
## Kernel-wise weight pre-alignment (K-WPA)

- Before the MAC operation, kernel-shared exponent ( $E_K$ ) are loaded from a ReRAM array row to register.
- The kernel-wise pre-aligned mantissa are ready to conduct MAC operation with input.

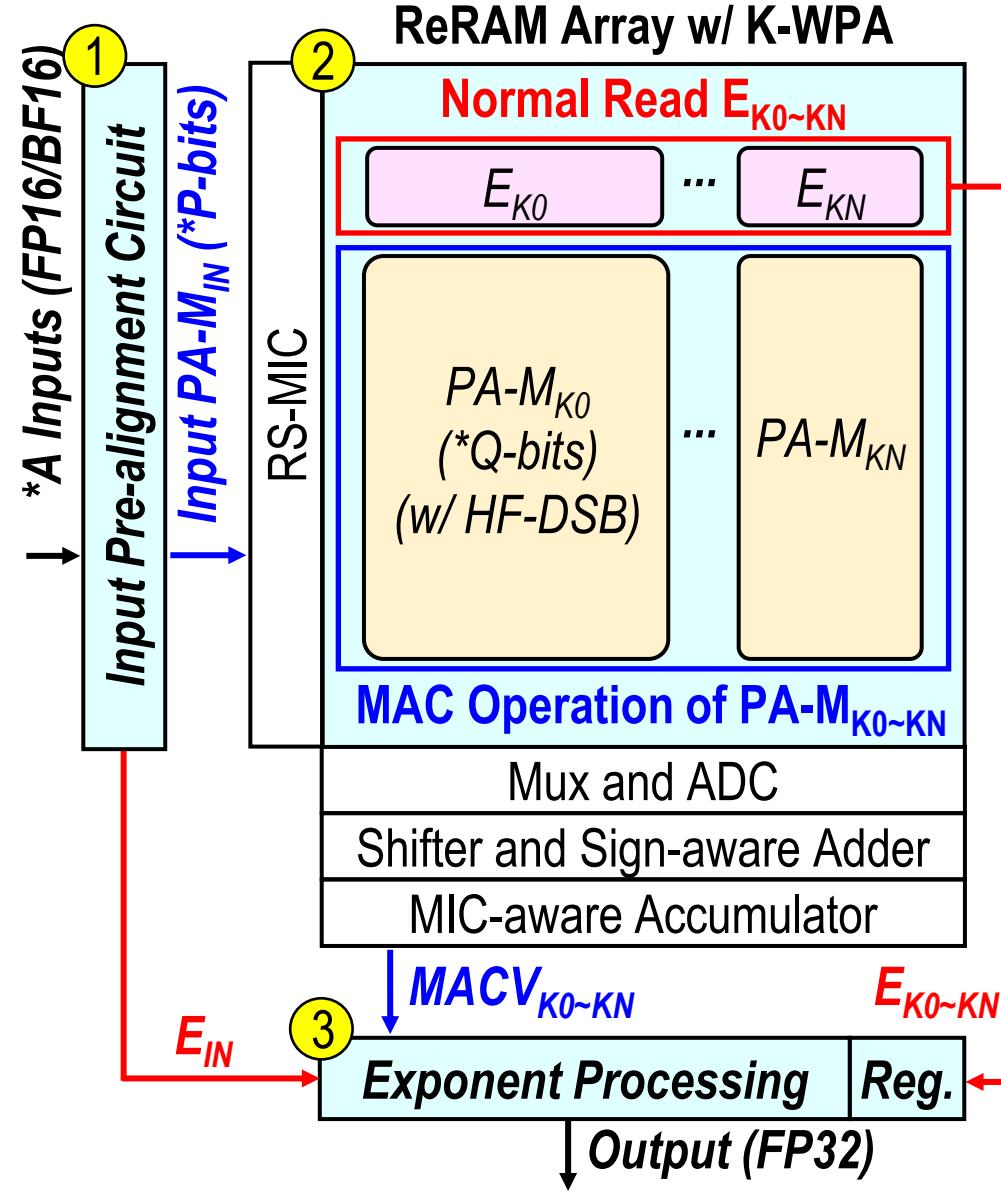


# FP computing flow with kernel-wise weight pre-alignment (3/7)

- The Kernel-wise weight pre-alignment (K-WPA) reduce the data truncation during weight alignment
  - K-WPA achieve fine-grained kernel-wise alignment reference

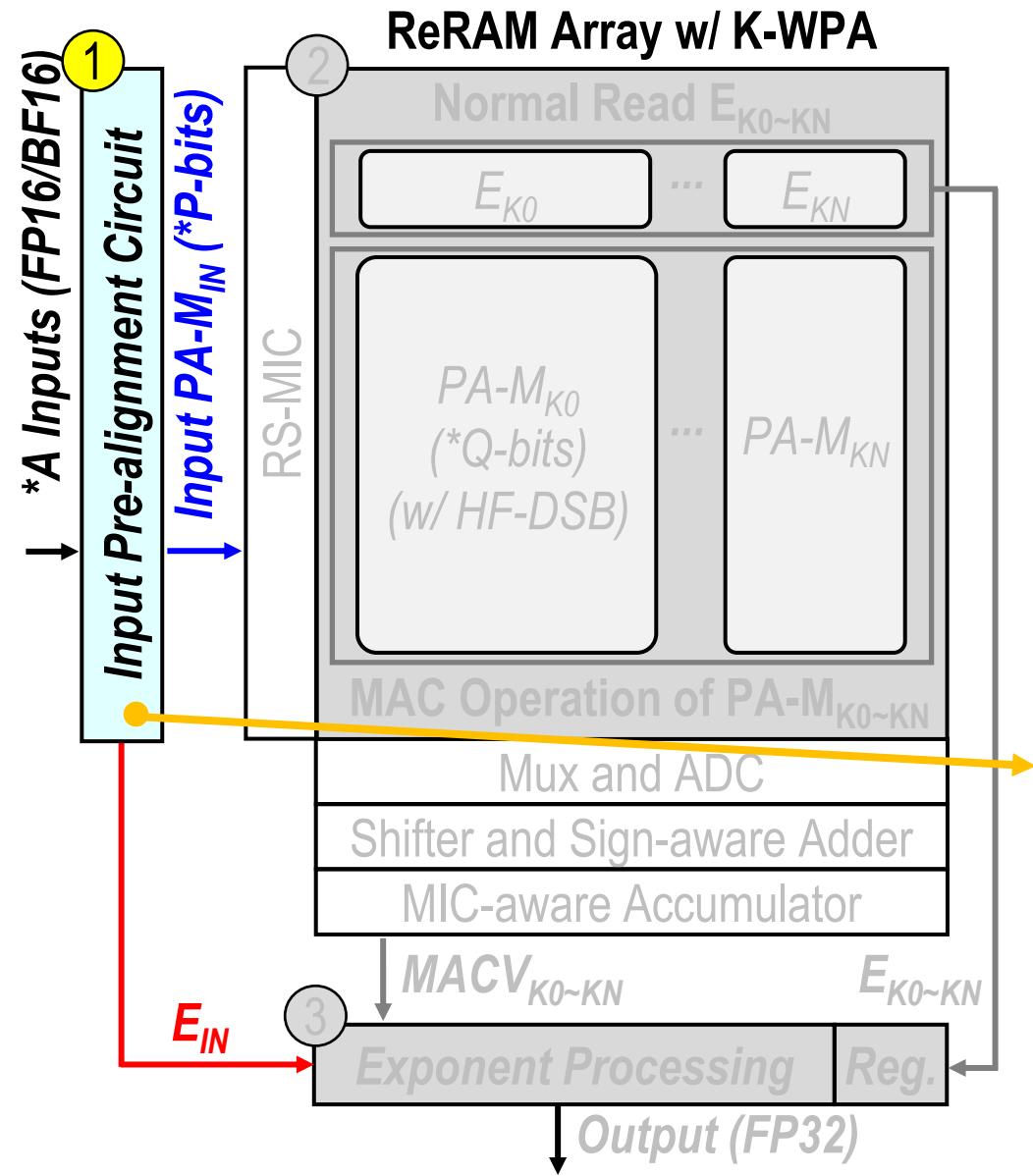


# FP computing flow with kernel-wise weight pre-alignment (4/7)



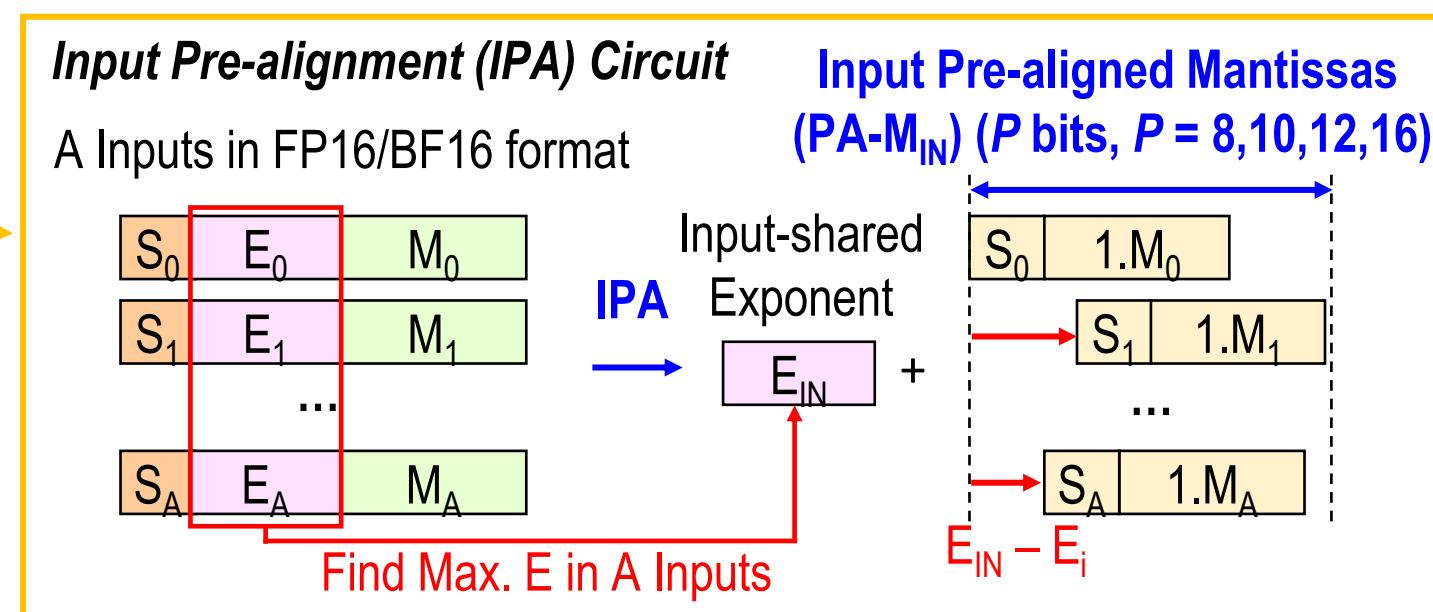
- Kernel-wise weight pre-alignment
- Floating-point computing flow
  - Step1 : Input pre-alignment
  - Step2 : MAC operation between pre-aligned input and weight
  - Step3 : Exponent processing

# FP computing flow with kernel-wise weight pre-alignment (5/7)

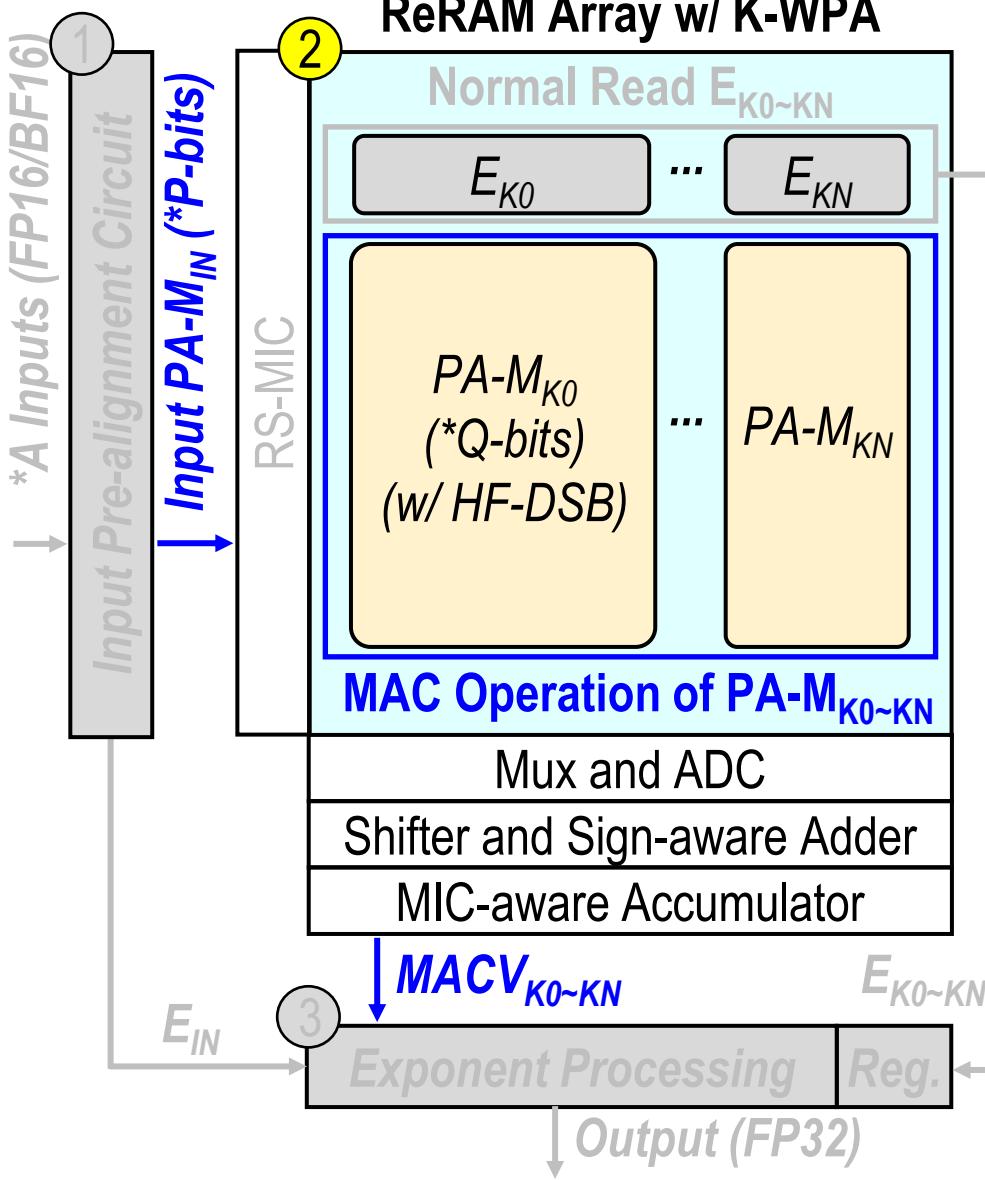


## □ Step1 : Input pre-alignment circuit

- Fetches A inputs
- Find maximum exponent
- Aligns the sign and mantissa according to its exponent difference ( $E_{IN} - E_i$ ) to generate input pre-aligned mantissas ( $PA-M_{IN}$ )



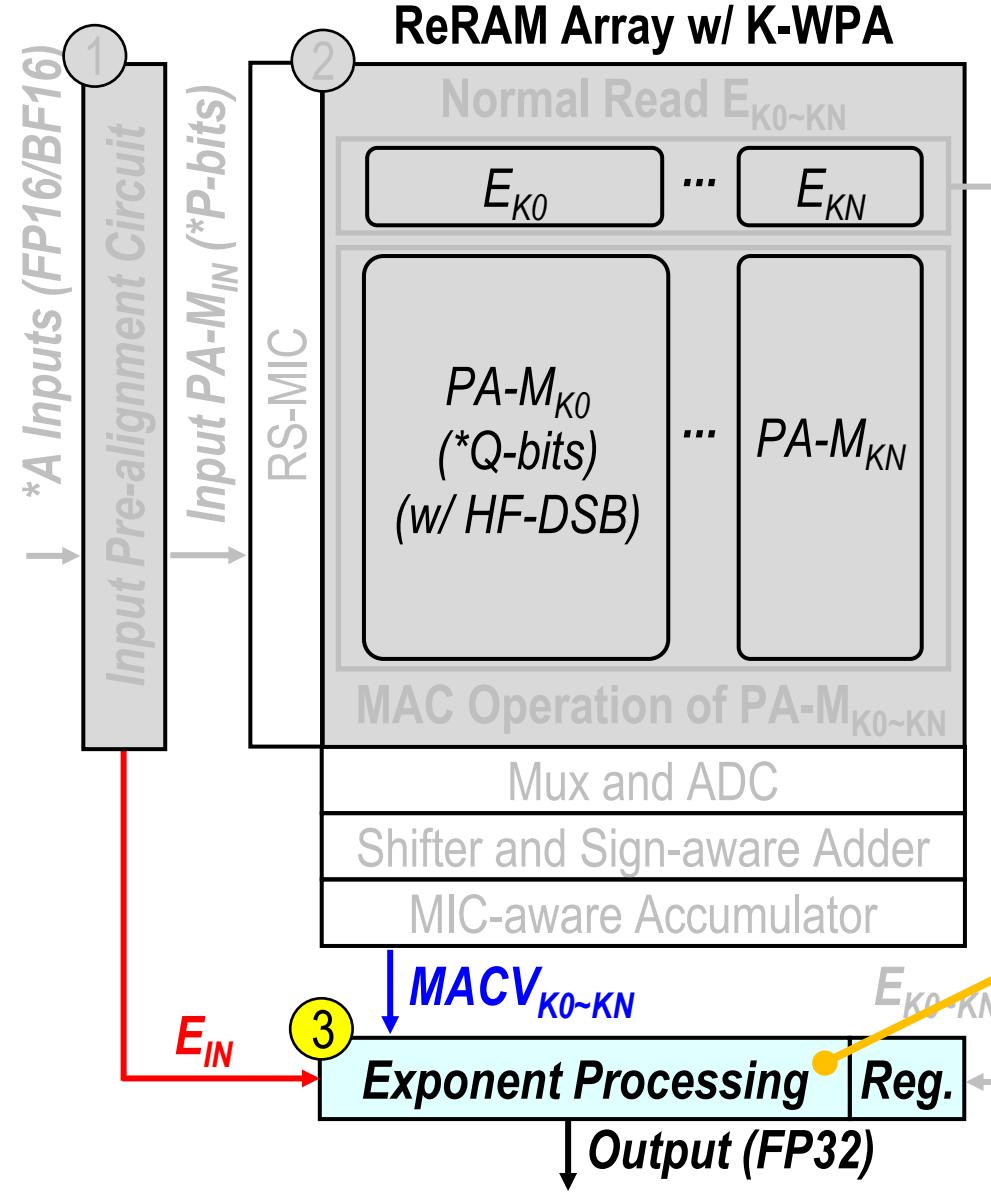
# FP computing flow with kernel-wise weight pre-alignment (6/7)



## □ Step2 : MAC operation between pre-aligned input and weight

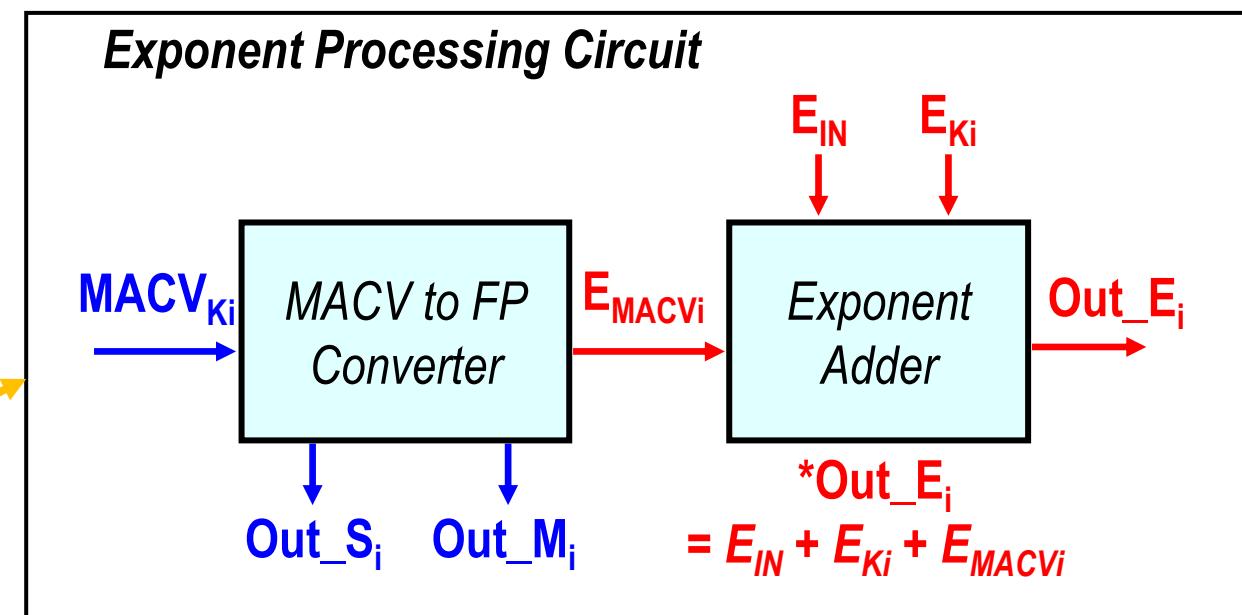
- $PA\text{-}M_{IN}$  are sent to the ReRAM array for MAC operations with the  $PA\text{-}M_K$
- The ADC and Adder readout and generate the MAC value (MACV)

# FP computing flow with kernel-wise weight pre-alignment (7/7)



## □ Step3 : Exponent processing

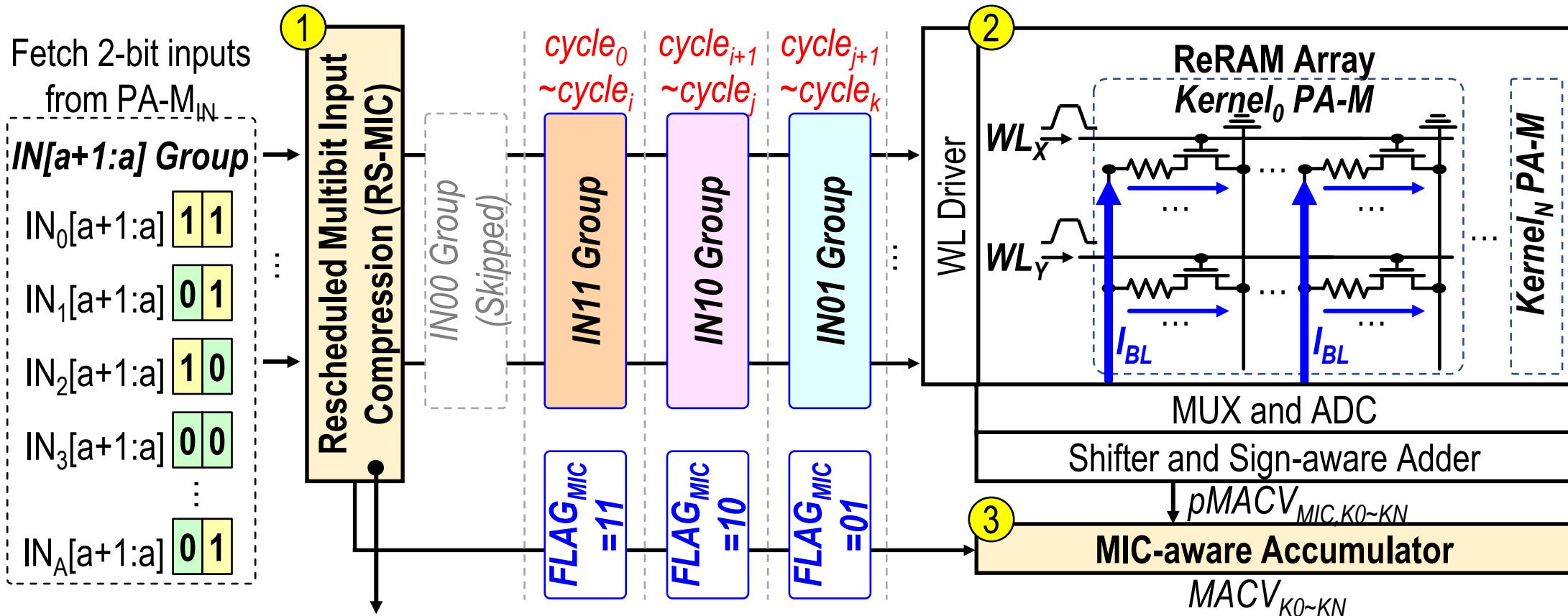
- **MACV to FP Converter:** Decompose MACV into sign, mantissa, exponent
- **Exponent Adder:** Add exponent to the input-shared exponent ( $E_{IN}$ ) and the kernel-shared exponent ( $E_K$ ).



# Outline

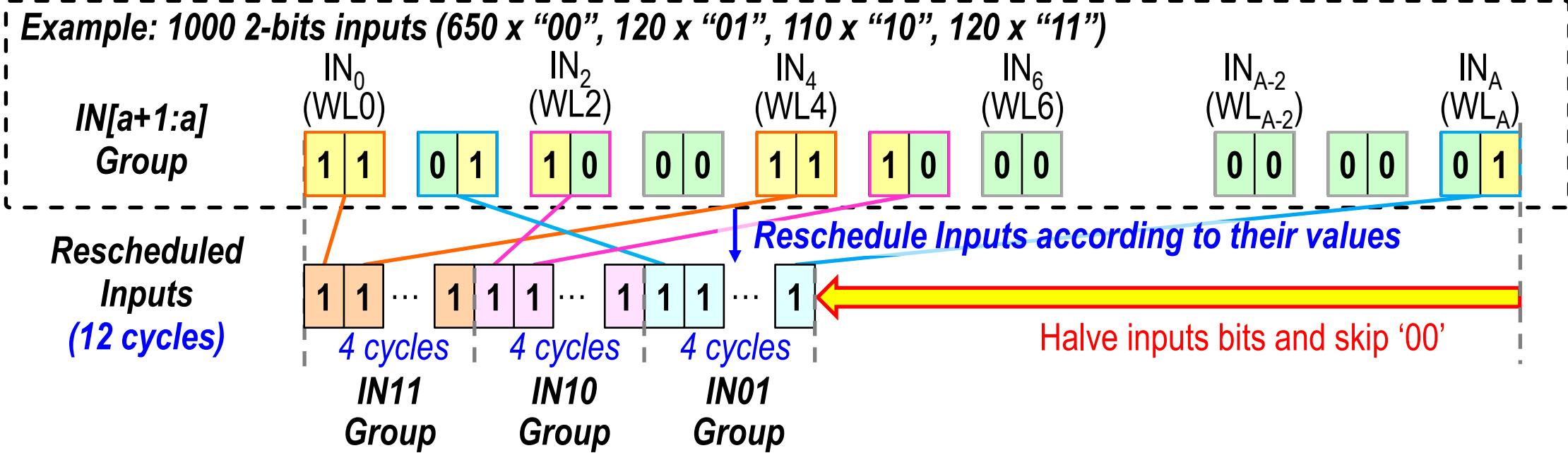
- Challenges of floating-point (FP) nonvolatile compute-in-memory (nvCIM) for AI edge devices
- Proposed schemes in nonvolatile CIM macro
  - Overview of floating-point ReRAM-CIM macro
  - FP computing flow with kernel-wise weight pre-alignment (K-WPA)
  - Rescheduled multi-bit input compression (RS-MIC)
  - HRS-favored dual-sign-bit (HF-DSB) weight encoding
- Performance and measurement results
- Conclusion

# Rescheduled multi-bit input compression (1/4)



- Step 1: Reschedule and compress inputs according to place value
- Step 2: Conduct MAC operation in ReRAM array
- Step 3: Decompress and accumulate MAC values

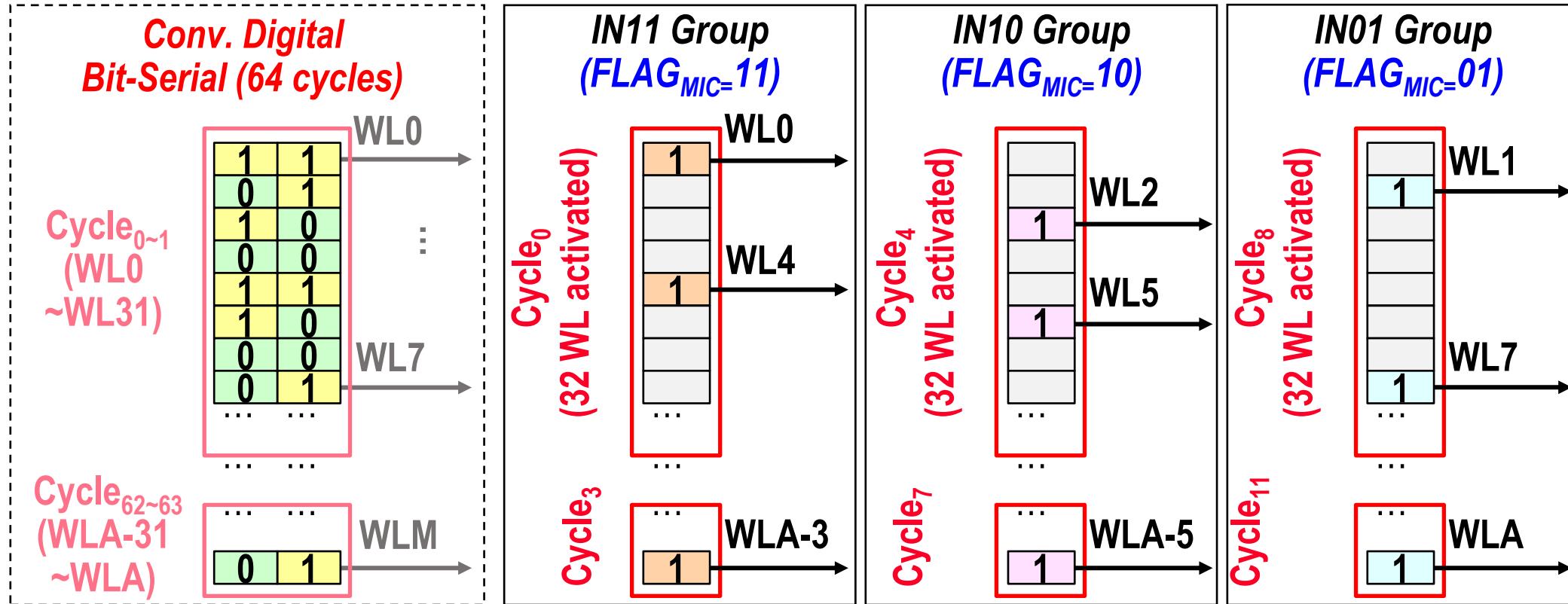
# Rescheduled multi-bit input compression (2/4)



## □ RS-MIC reschedules inputs according to their input values

- Inputs with values 11, 10 or 01 are rescheduled and compressed into a 1b input in the IN11, IN10 or IN01 group
- Inputs with a value of 00 are skipped
- This process halves the number of input bits and skip ‘00’ inputs.

# Rescheduled multi-bit input compression (3/4)



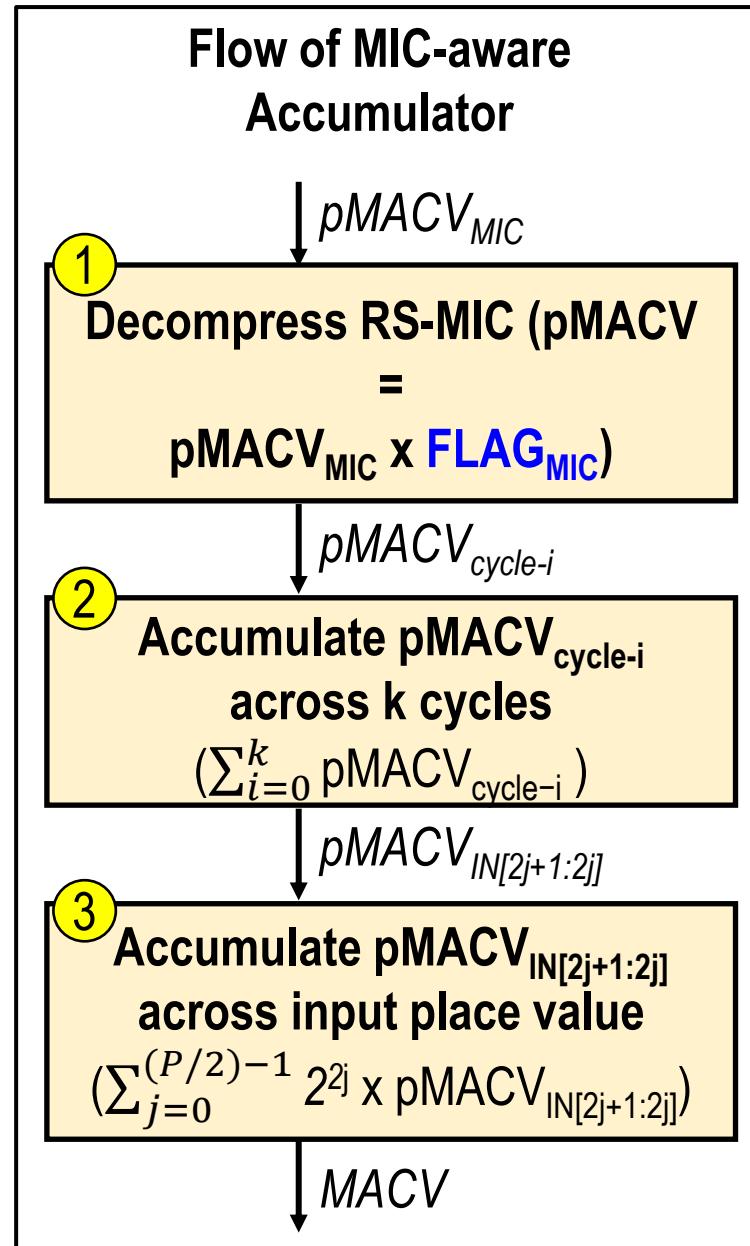
## □ Conduct MAC operation in ReRAM array

- The inputs are sent to ReRAM array where a fixed number of WLs activated in each cycle.
- The correspondence between input and WL does not change after RS-MIC.
- The example shows RS-MIC reduce cycles from 64 cycles to 12 cycles.

# Rescheduled multi-bit input compression (4/4)

## □ Decompress and accumulate partial MAC values (pMACV)

- Decompress pMACV by multiplying it with FLAG<sub>MIC</sub>
- Accumulates pMACV over multiple cycles within the IN11, IN10 or IN01 group
- Accumulator pMACV across input place value to derive the final MACV



# Outline

- Challenges of floating-point (FP) nonvolatile compute-in-memory (nvCIM) for AI edge devices
- Proposed schemes in nonvolatile CIM macro
  - Overview of floating-point ReRAM-CIM macro
  - FP computing flow with kernel-wise weight pre-alignment (K-WPA)
  - Rescheduled multi-bit input compression (RS-MIC)
  - HRS-favored dual-sign-bit (HF-DSB) weight encoding
- Performance and measurement results
- Conclusion

# HRS-favored dual-sign-bit weight encoding (1/4)

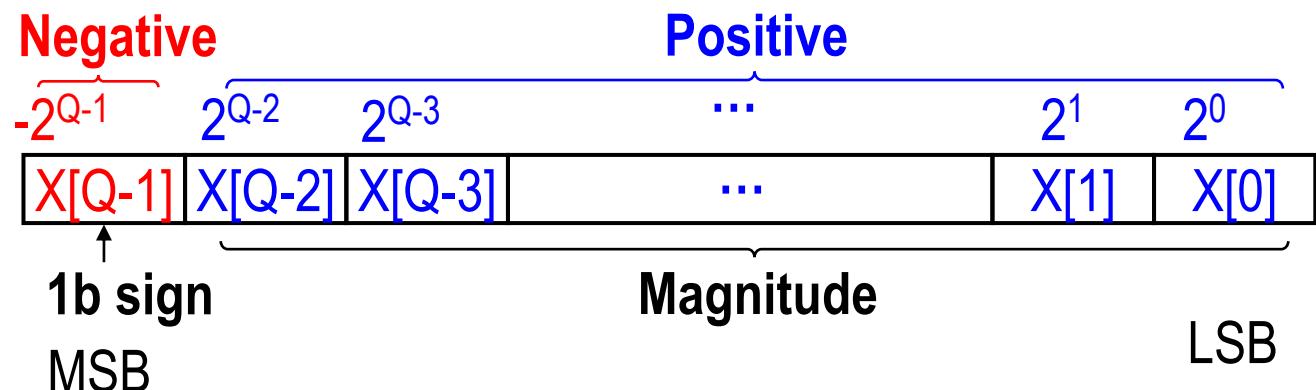
## □ Conventional 2's complement

- Only the first bit serves as the sign bit
- Remaining bits are used as magnitude bits

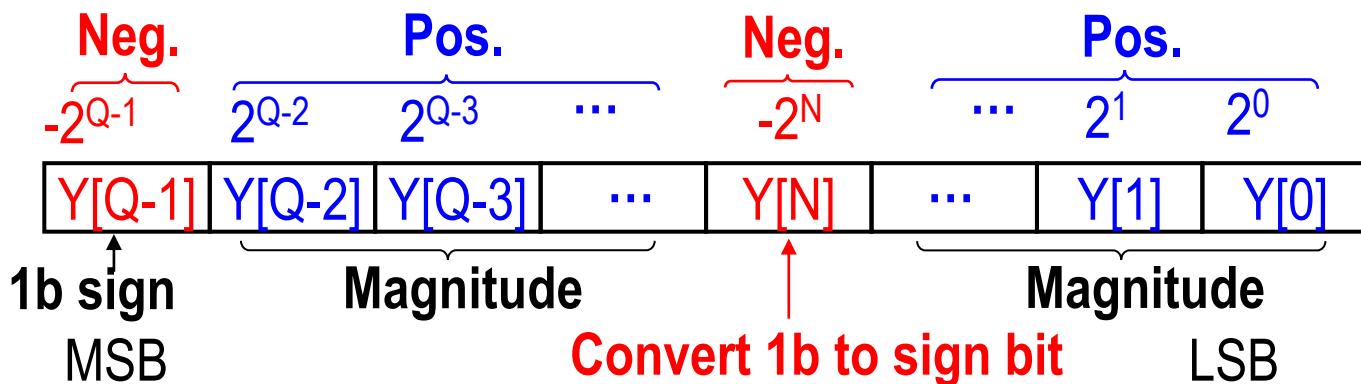
## □ The HF-DSB representation (Y(DSB))

- Convert a magnitude bit into a sign bit
- Maintain same weight value as the original weight value

### Conventional 2's complement



### Prop. HF-DSB representation

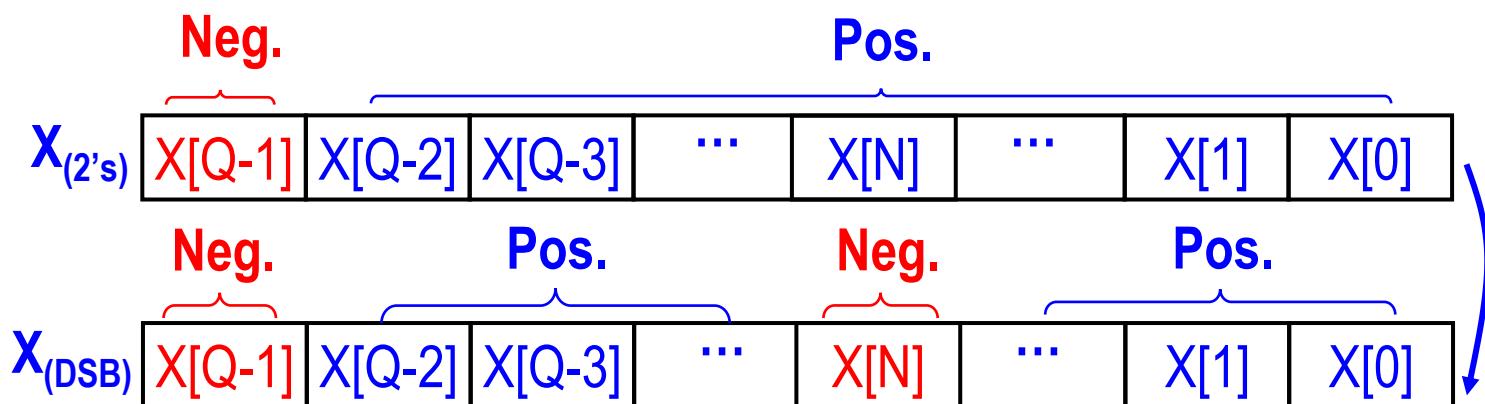


# HRS-favored dual-sign-bit weight encoding (2/4)

- The conversion term between conventional format and DSB format is  $X[N] \times 2^{N+1}$
- N represents place value undergoes conversion into sign bit

$$Y_{(DSB)} = X_{(2's)} = X_{(DSB)} + \text{Conversion term}$$

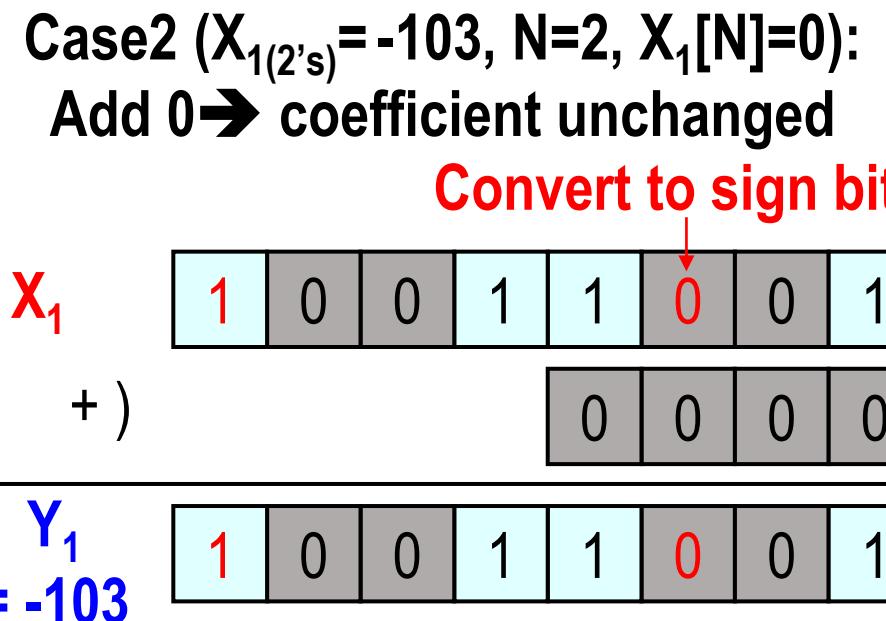
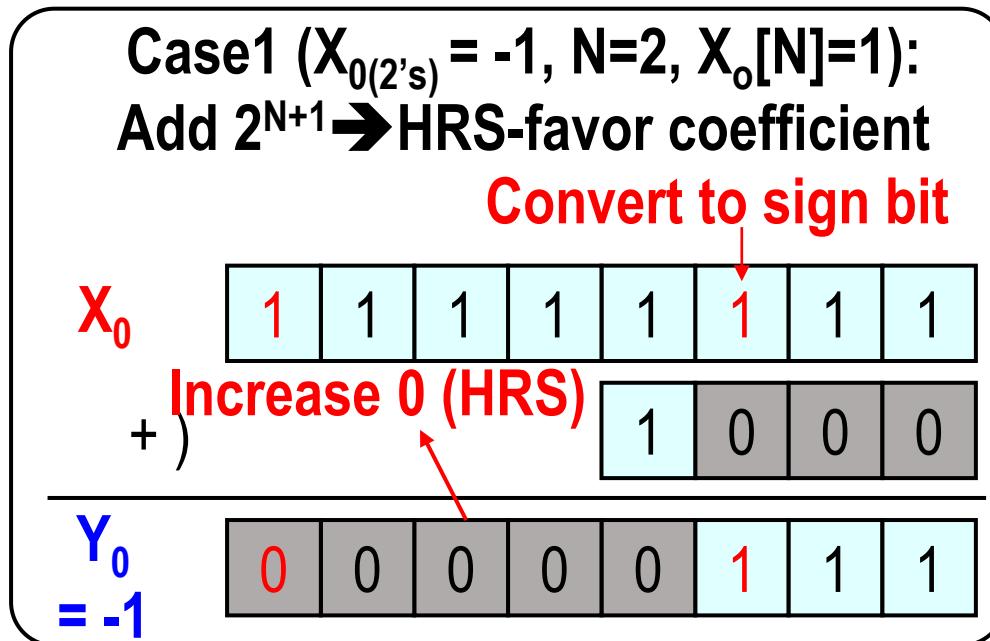
$$\begin{aligned}\text{Conversion term} &= X_{(2's)} - X_{(DSB)} = X[N] \times 2 \times 2^N \\ &= X[N] \times 2^{N+1}\end{aligned}$$



$$\begin{aligned}\Delta &= \\ X[N] \times 2^N - &\\ (-X[N] \times 2^N) &\\ &= X[N] \times 2^{N+1}\end{aligned}$$

# HRS-favored dual-sign-bit weight encoding (3/4)

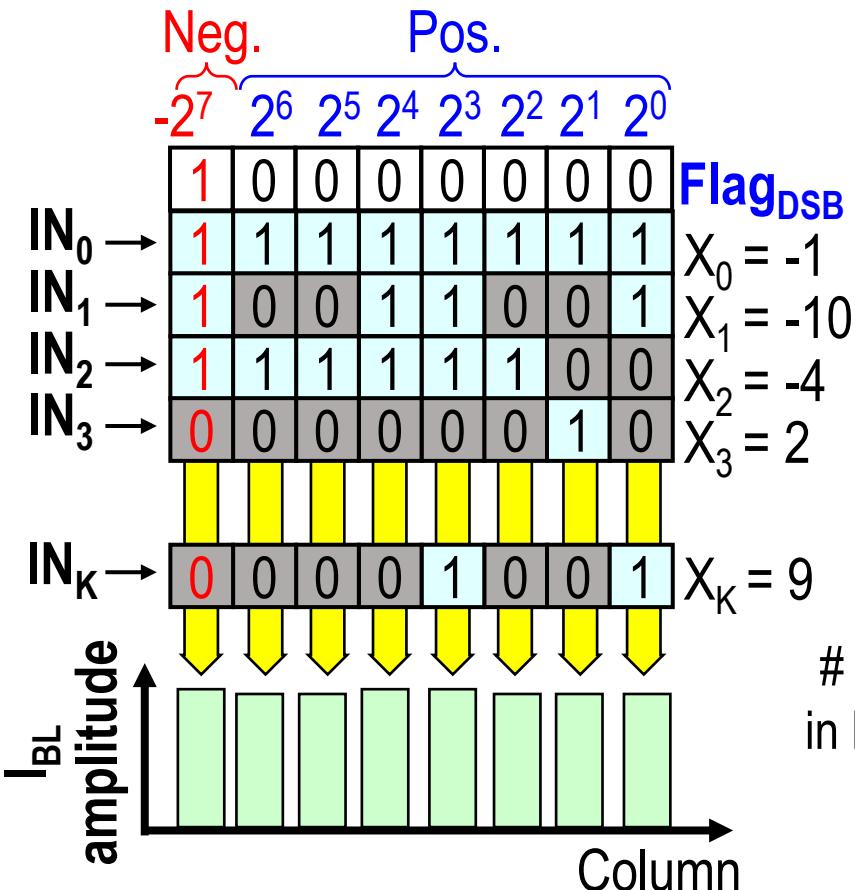
2's complement  
DSB format



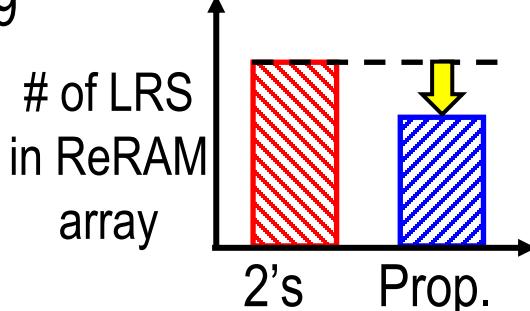
- Two case to showcase the conversion from conventional 2's complement to DSB format
  - Case1:  $X[N] \times 2^{N+1} = 2^3 \rightarrow$  Add HRS-favor coefficient to increase 0 (HRS)
  - Case2:  $X[N] \times 2^{N+1} = 0 \rightarrow$  Coefficient unchanged

# HRS-favored dual-sign-bit weight encoding (4/4)

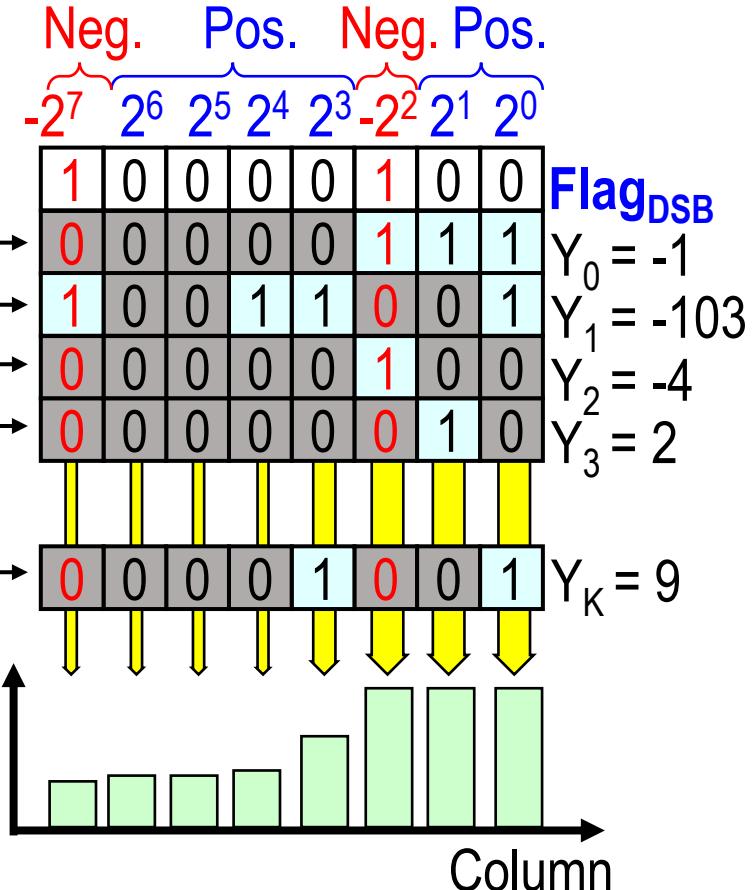
## One Kernel PA-M in 2's Complement



HF-DSB  
Weight  
Encoding



## One Kernel PA-M in HF-DSB

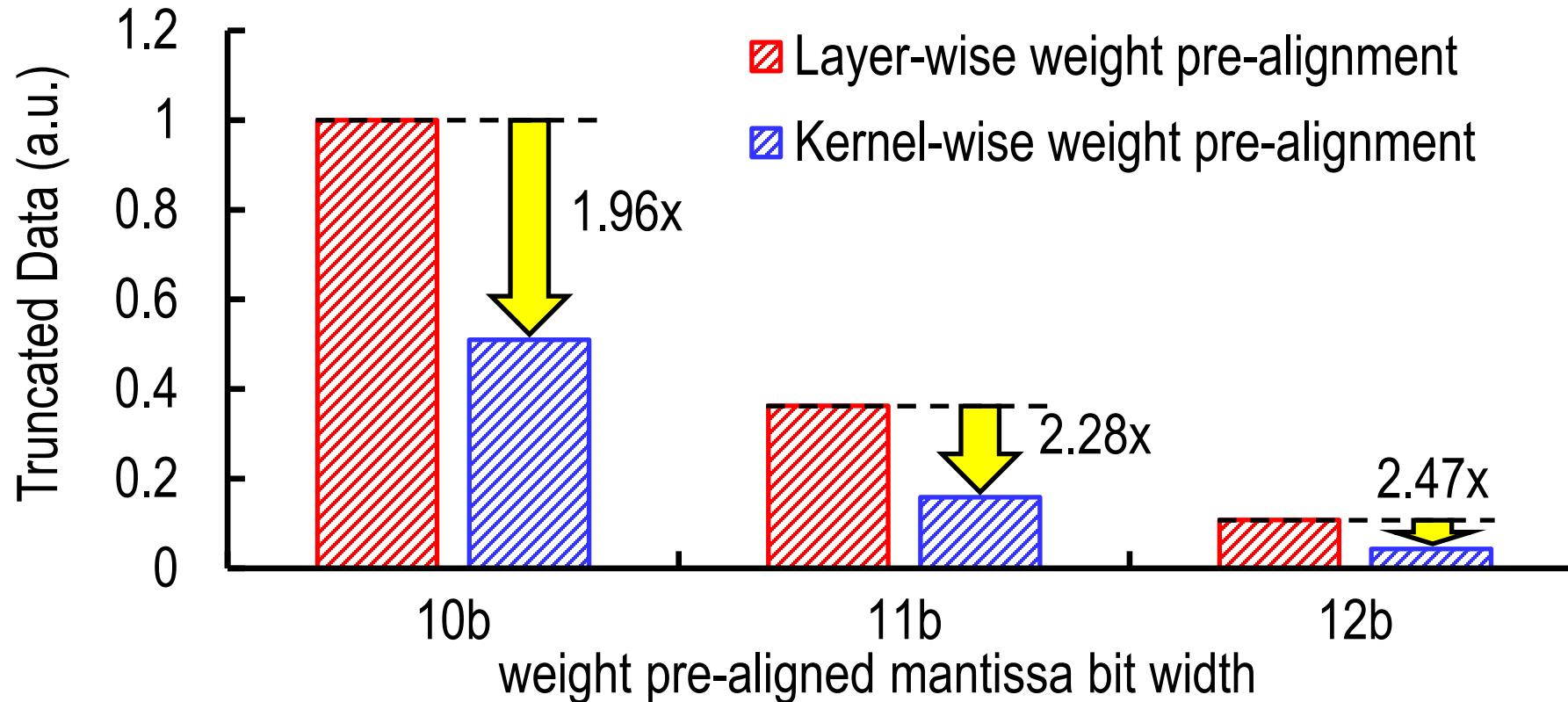


- HF-DSB minimizes bit line currents by reducing LRS cell access during accumulation.

# Outline

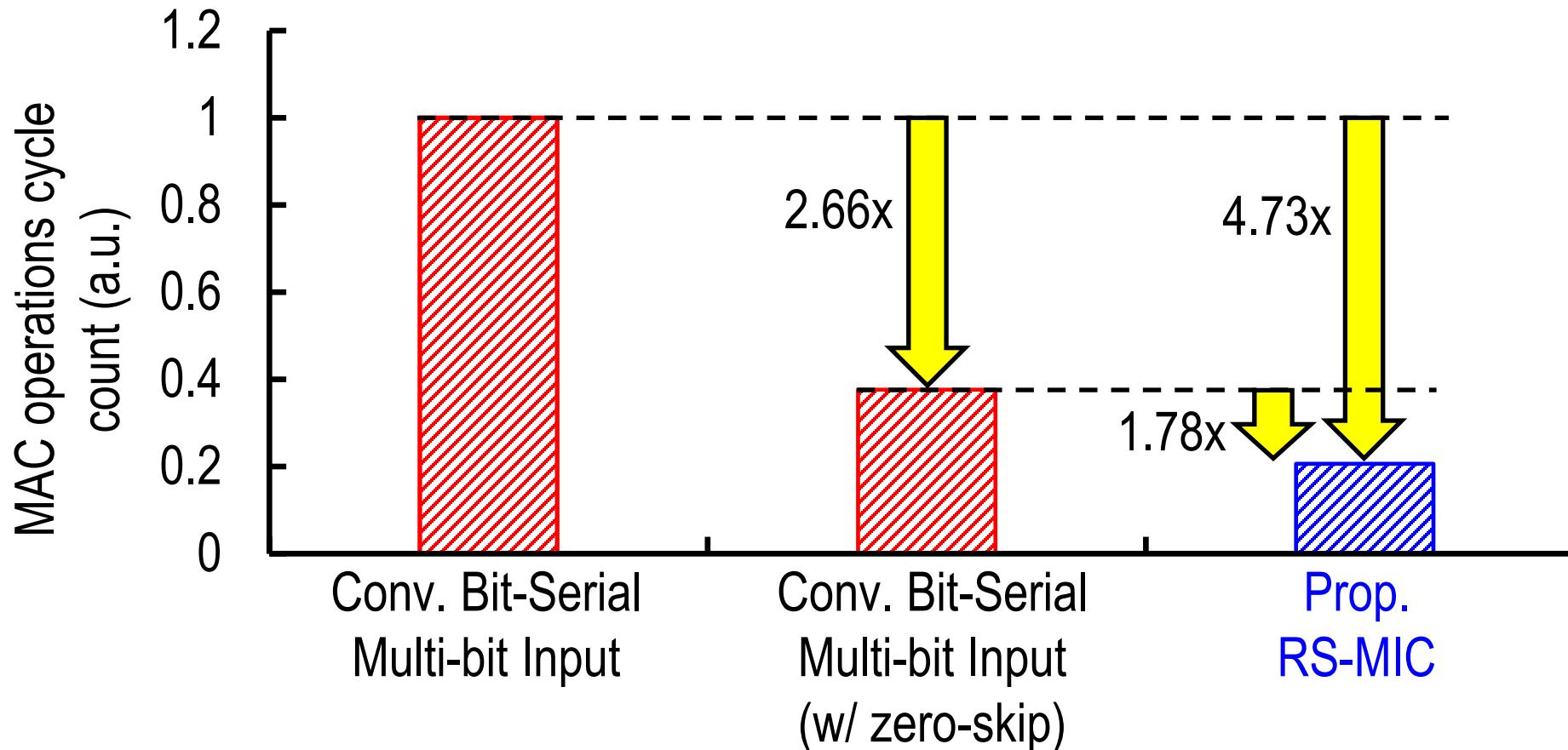
- Challenges of floating-point (FP) nonvolatile compute-in-memory (nvCIM) for AI edge devices
- Proposed schemes in nonvolatile CIM macro
  - Overview of floating-point ReRAM-CIM macro
  - FP computing flow with kernel-wise weight pre-alignment (K-WPA)
  - Rescheduled multi-bit input compression (RS-MIC)
  - HRS-favored dual-sign-bit (HF-DSB) weight encoding
- Performance and measurement results
- Conclusion

# Amount of truncated data during weight pre-alignment



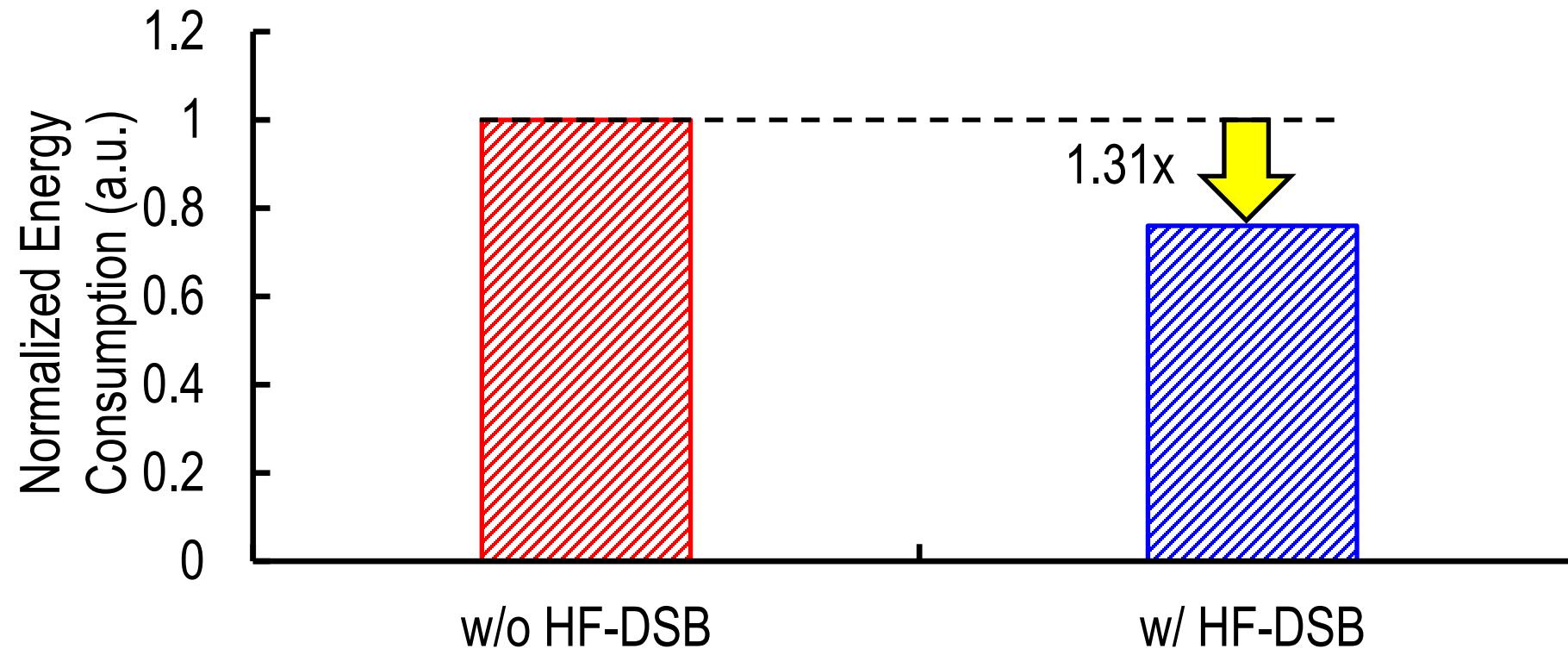
- The proposed K-WPA reduce the truncated data during weight pre-alignment by 1.96-2.47x
  - ResNet-20 & CIFAR-100 under BF16

# MAC operation cycle count



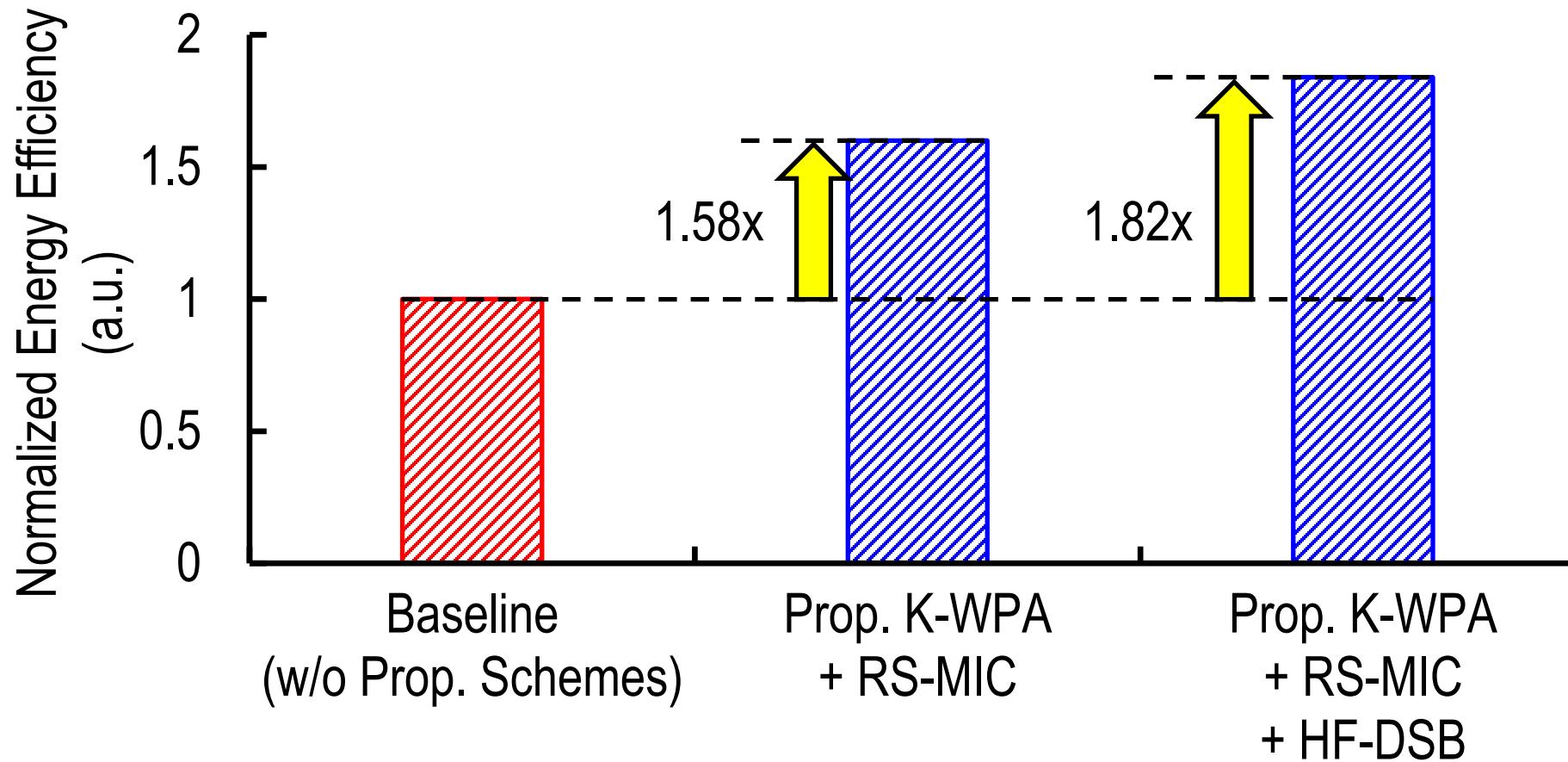
- MAC operation cycle count under various multi-bit input schemes
  - The proposed RS-MIC reduce the MAC operation cycle count by 1.78-4.73x
  - ResNet-20 & CIFAR-100 under BF16

# Energy consumption of ReRAM cell array



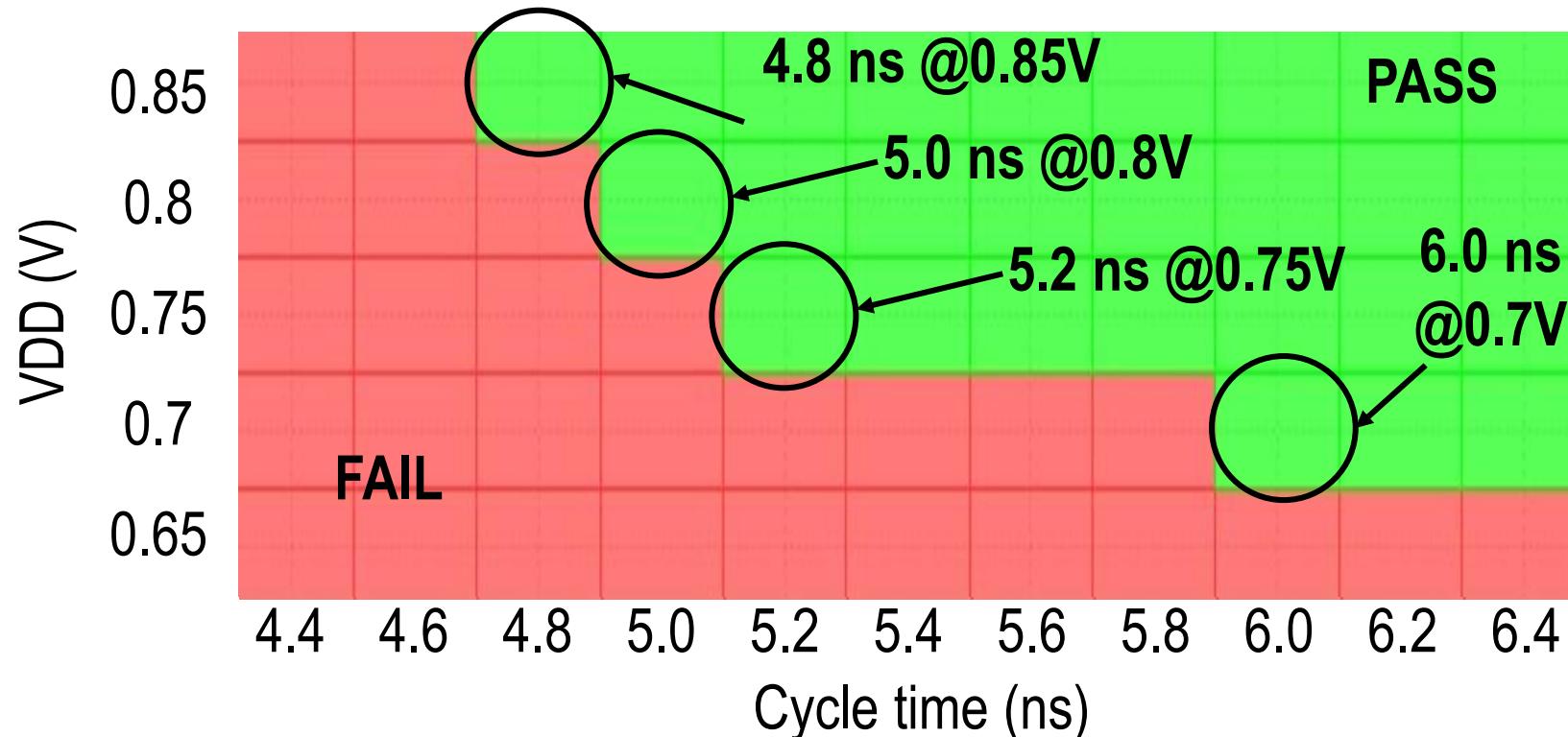
- Simulated energy consumption of ReRAM cell array
  - The proposed HF-DSB reduce the ReRAM energy consumption by 1.31x
  - ResNet-20 & CIFAR-100 under BF16

# Macro-level performance of each scheme



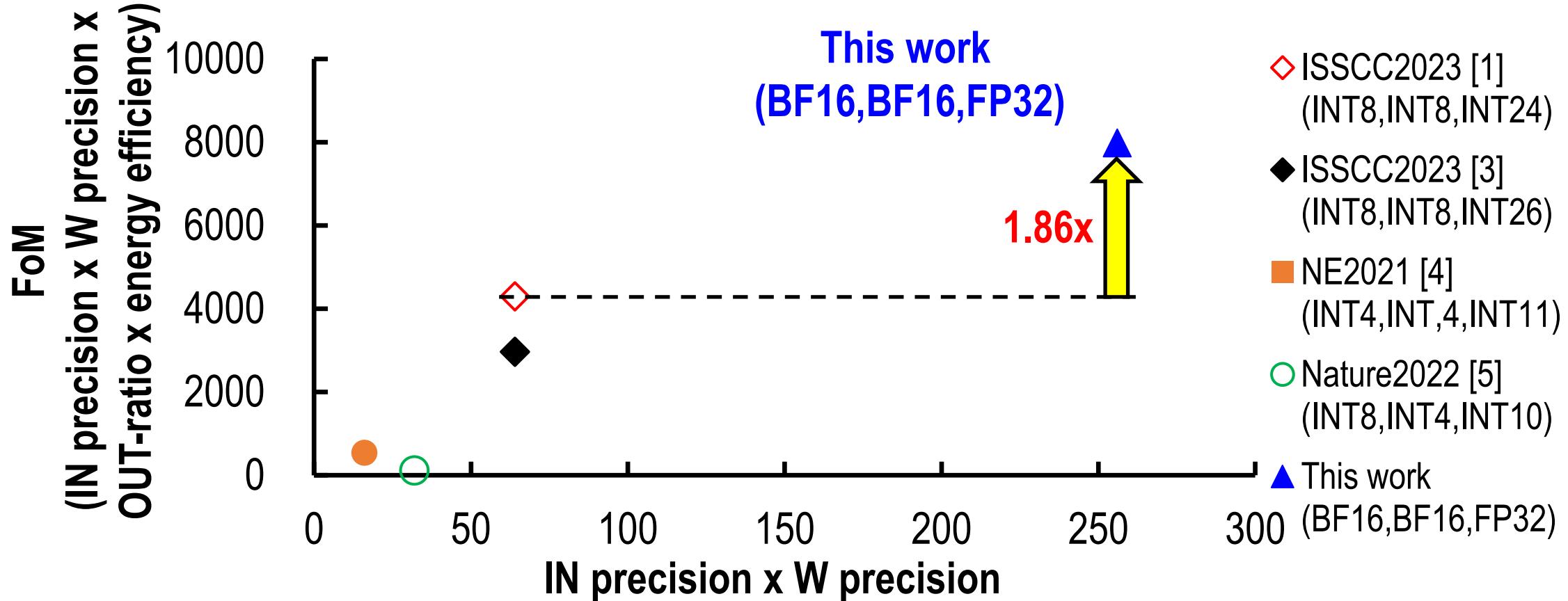
- Measured energy efficiency with proposed scheme
  - The proposed scheme improve the macro-level energy efficiency by 1.82x
  - ResNet-20 & CIFAR-100 under BF16

# Shmoo plot for floating-point ReRAM-nvCIM macro cycle time



- Measurement results
  - Macro cycle time Access time ( $t_{AC}$ ) = 5.0ns @  $V_{DD}=0.8V$

# Position Chart



- Position chart among silicon-verified nvCIM works.
- FoM improvement of >1.86× over ISSCC'23

# Chip Summary



Technology	22nm CMOS logic process (Ultra low leakage)	
Memory device	Foundry provided 1T1R ReRAM	
ReRAM-CIM Capacity	16Mb (16 sub-banks)	
Input/Weight precision	FP16 / BF16	
Output precision	FP32	
Macro area (Inc. test mode)	8.2mm <sup>2</sup>	
Supply voltage	0.7 - 0.8V	
Throughputs (TFLOPS) <sup>*1*3</sup>	0.86 (BF16)	0.78 (FP16)
Computing density (TFLOPS/mm <sup>2</sup> ) <sup>*1*3</sup>	0.104 (BF16)	0.095 (FP16)
Energy efficiency (TFLOPS/W) <sup>*3</sup>	31.2 <sup>*1</sup> - 65.5 <sup>*2</sup> (BF16)	28.7 <sup>*1</sup> - 60.4 <sup>*2</sup> (FP16)
Inference Accuracy (CIFAR-100) <sup>*4</sup>	69.48% (Top-1), 91.59% (Top-5) (BF16)	
Inference Accuracy (ImageNet) <sup>*5</sup>	71.55% (Top-1), 90.17% (Top-5) (BF16)	

# Outline

- Challenges of floating-point (FP) nonvolatile compute-in-memory (nvCIM) for AI edge devices
- Proposed schemes in nonvolatile CIM macro
  - Overview of floating-point ReRAM-CIM macro
  - FP computing flow with kernel-wise weight pre-alignment (K-WPA)
  - Rescheduled multi-bit input compression (RS-MIC)
  - HRS-favored dual-sign-bit (HF-DSB) weight encoding
- Performance and measurement results
- Conclusion

# Conclusion

- **Features of proposed floating-point ReRAM-nvCIM macro:**
  - **FP computing flow with kernel-wise weight pre-alignment (K-WPA)**
    - Reduce accuracy loss due to the data truncation in weight pre-alignment
  - **Rescheduled multi-bit input compression (RS-MIC)**
    - Reduce MAC energy and latency with lossless compression
  - **HRS-favored dual-sign-bit (HF-DSB) weight encoding**
    - Reduce ReRAM array current consumption
- **A 16Mb floating-point ReRAM-nvCIM macro is verified:**
  - Supporting both BF16 and FP16 computing format
  - Measured energy efficiency was 31.2 TFLOPS/W under BF16 precision

# Thanks for your kind attention

Acknowledgements: NSTC-Taiwan, TSRI, NTHU-TSMC JDP



Please Scan to Rate  
This Paper



# A Flash-SRAM-ADC-Fused Plastic Computing-in-Memory Macro for Learning in Neural Networks in a Standard 14nm FinFET Process

Linfang Wang<sup>1,2</sup>, Weizeng Li<sup>1,2</sup>, Zhidao Zhou<sup>1,2</sup>, Hanghang Gao<sup>1,2</sup>, Zhi Li<sup>1,2</sup>,  
Wang Ye<sup>1,2</sup>, Hongyang Hu<sup>1</sup>, Jing Liu<sup>1</sup>, Jinshan Yue<sup>1</sup>, Jianguo Yang<sup>1</sup>, Qing Luo<sup>1</sup>,  
Chunmeng Dou<sup>1,2</sup>, Qi Liu<sup>1,3</sup>, Ming Liu<sup>1,3</sup>

<sup>1</sup>Institute of Microelectronics of the Chinese Academy of Sciences, Beijing, China

<sup>2</sup>University of Chinese Academy of Sciences, Beijing, China

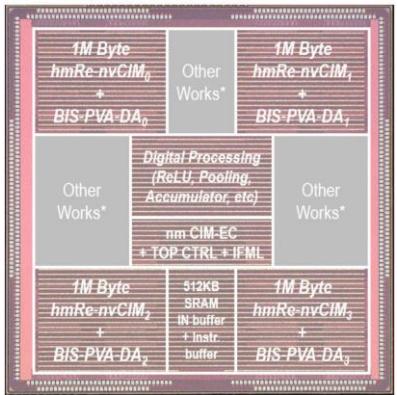
<sup>3</sup>Fudan University, Shanghai, China



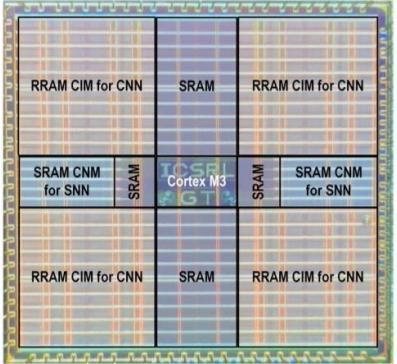
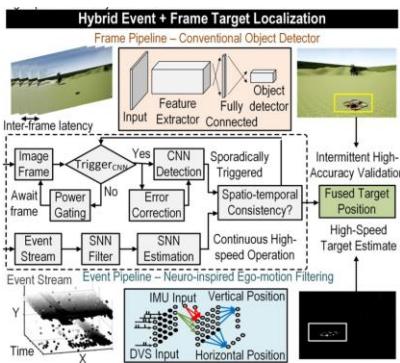
# Outline

- **Motivation and Challenges**
- **Proposed Plastic-CIM Macro**
  - Overview of Plastic-CIM Macro
  - Plastic Cell Array
  - Differential Merged-into-Array ADC
- **Performance and Measurement Results**
- **Conclusion**

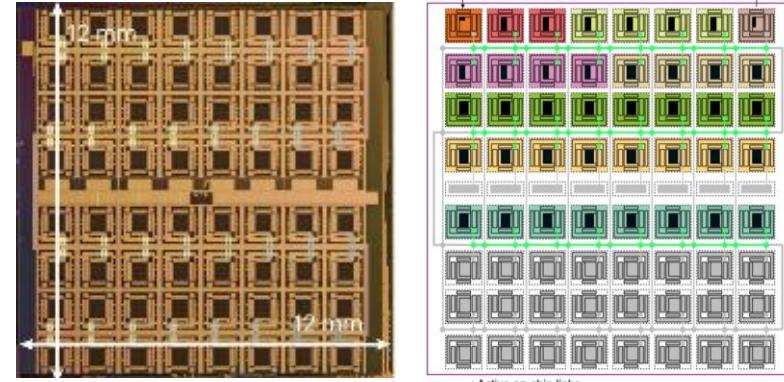
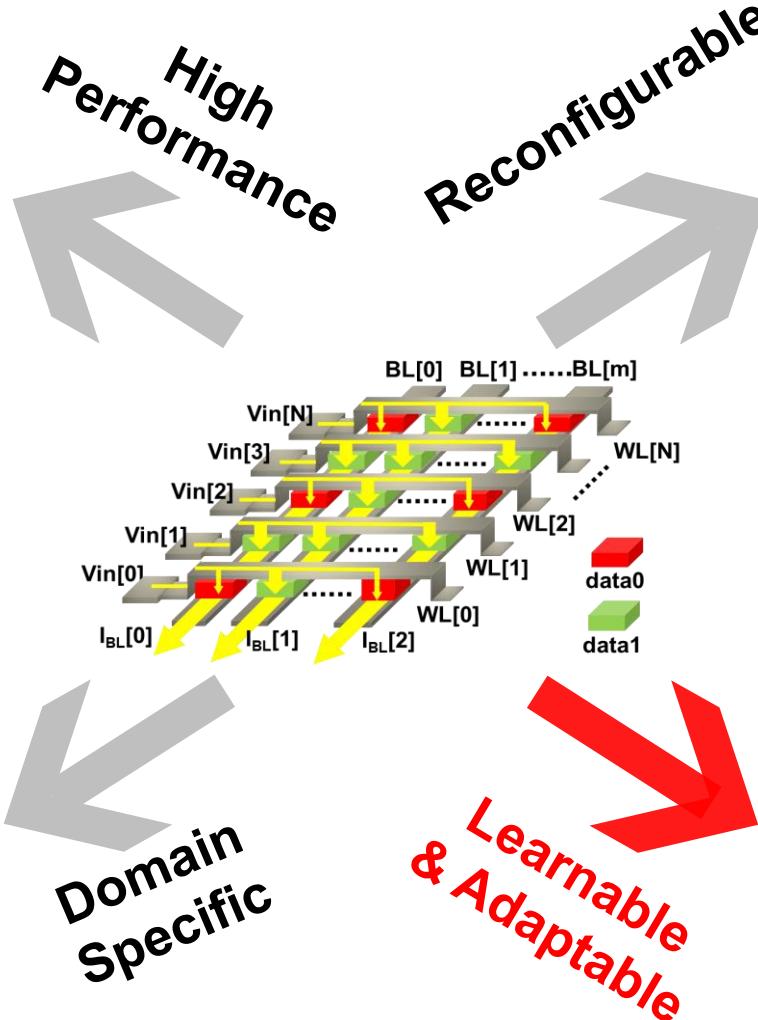
# Development of nvCIM based Accelerator



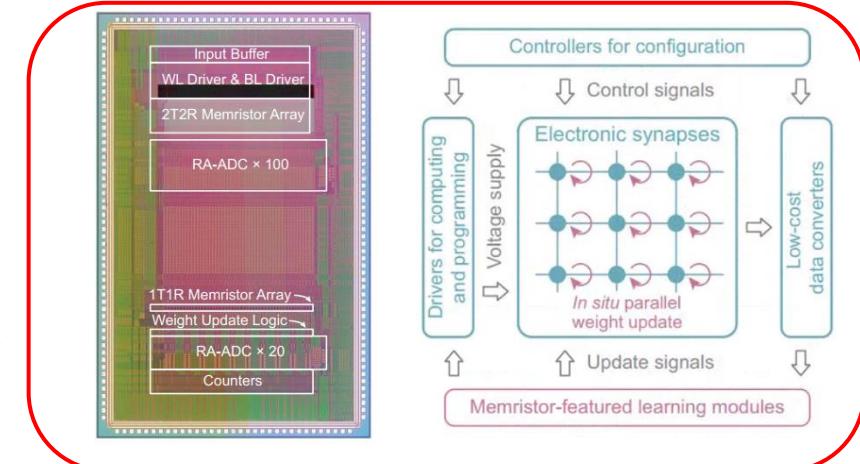
[W. Huang, ISSCC, 2023]



[M. Chang, ISSCC, 2023]



[M. Le Gallo, Nat Electron, 2023]

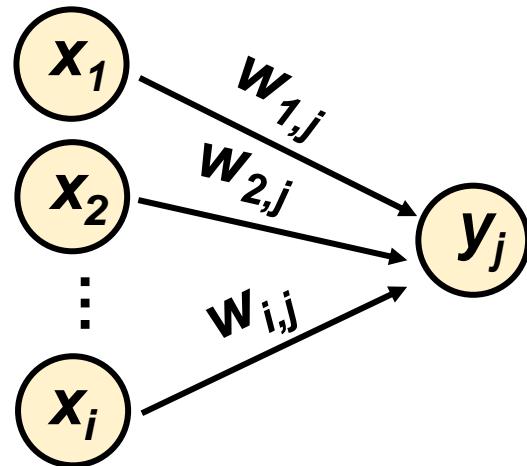


[W. Zhang, Science, 2023]

■ Edge AI applications call for CIM with learning abilities to adapt to the dynamic and unpredictable environments

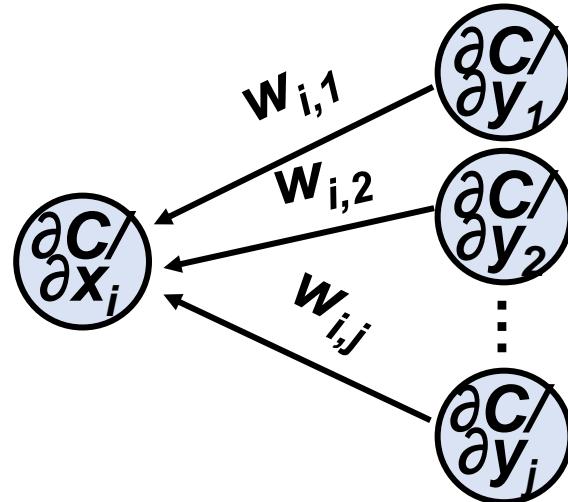
# Learning by Back Propagation

## Feed-forward



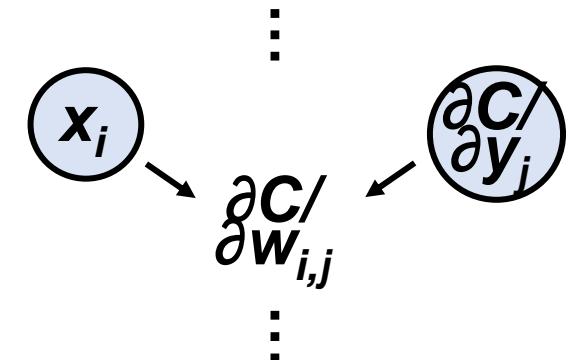
$$y_j = \sum w_{i,j} x_i$$

## Error-computation



$$\frac{\partial C}{\partial x_i} = \sum w_{i,j} \frac{\partial C}{\partial y_j}$$

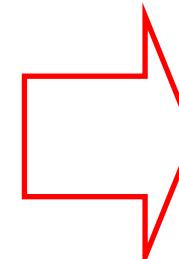
## Weight Gradient-computation



$$\frac{\partial C}{\partial w_{i,j}} = x_i \frac{\partial C}{\partial y_j}$$

## Conventional NN training:

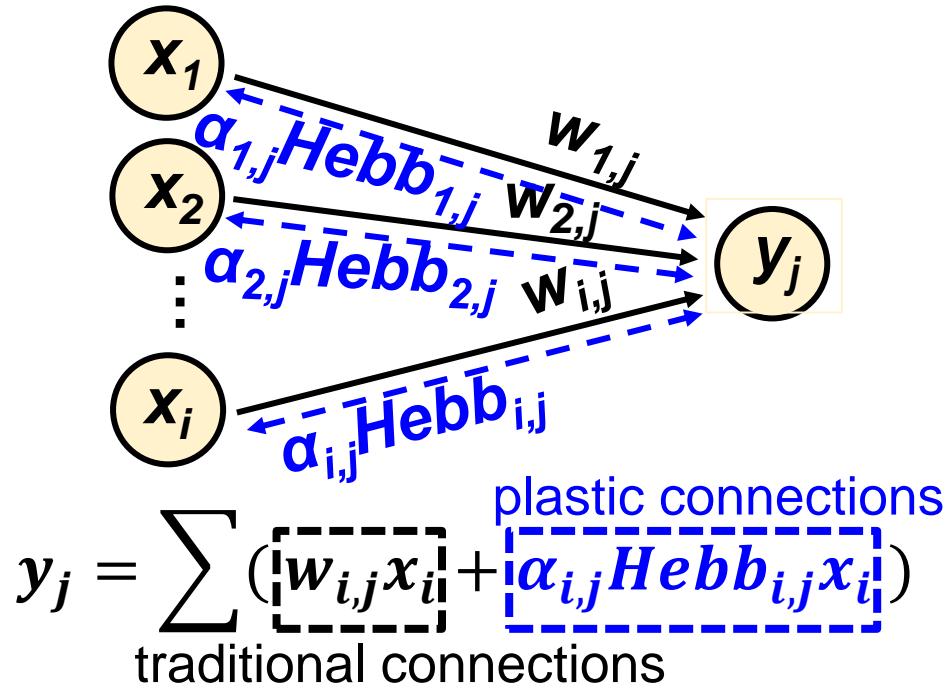
- High computational precision
- Large amounts of intermediate data
- Cannot keep learning after initial training



Hard to be deployed  
on edge devices  
for learning!

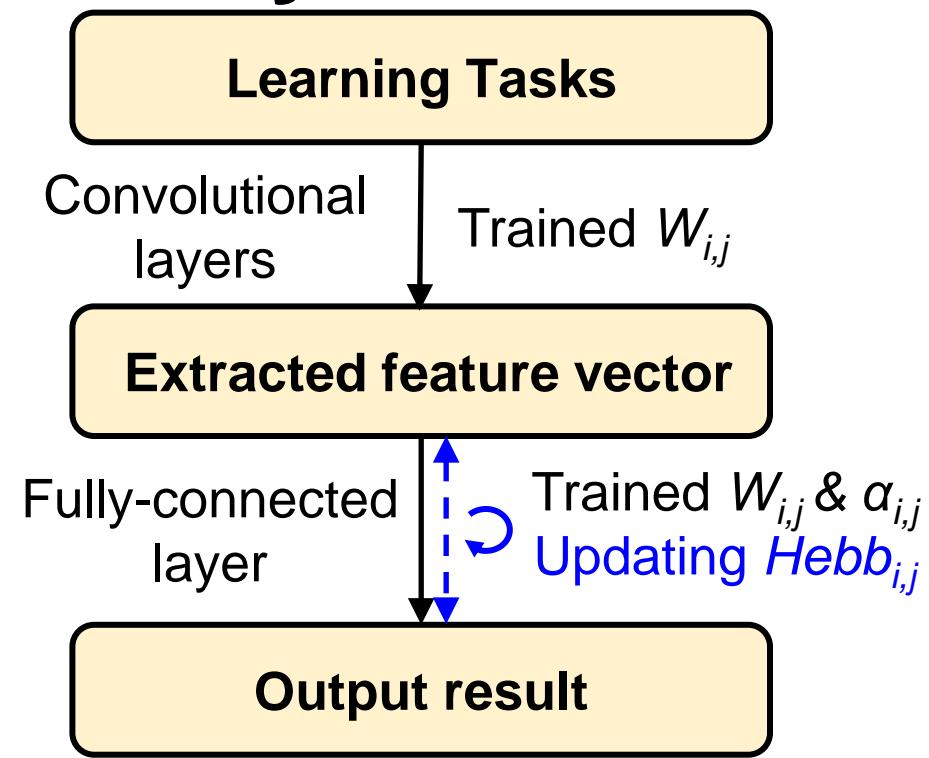
# Learning by Synaptic Plasticity

## Feed-forward



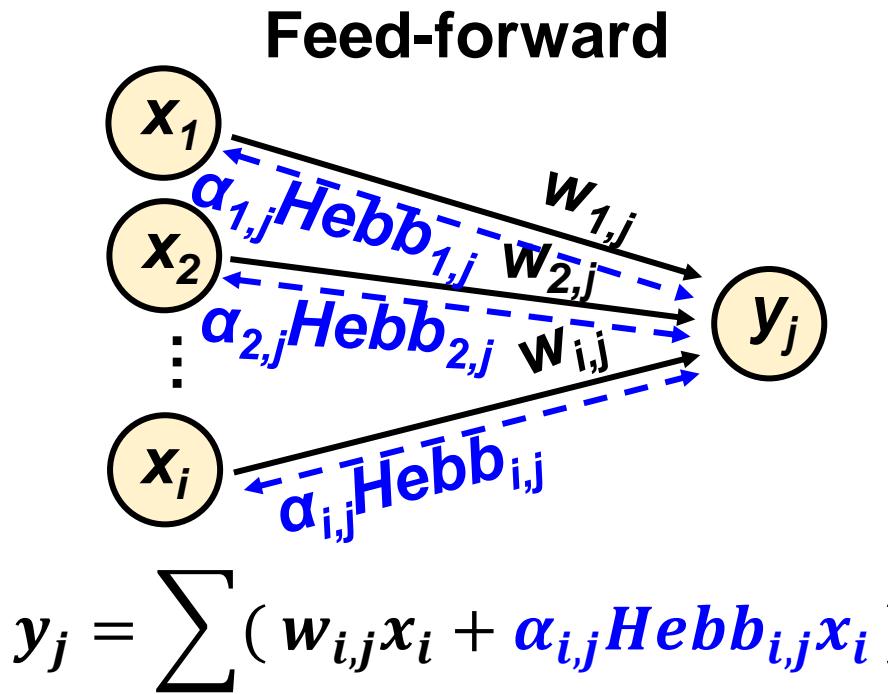
## ■ Plastic NN\* learns from experience:

- Keep learning in feed-forward after initial training
- Updating Hebb by Hebbian theory: “neurons that fire together, wire together”  $Hebb_{i,j}(t + 1) = \eta x_i(t)y_j(t) + (1 - \eta)Hebb_{i,j}(t)$



\*[T. Miconi , ICML, 2018]

# Learning by Synaptic Plasticity



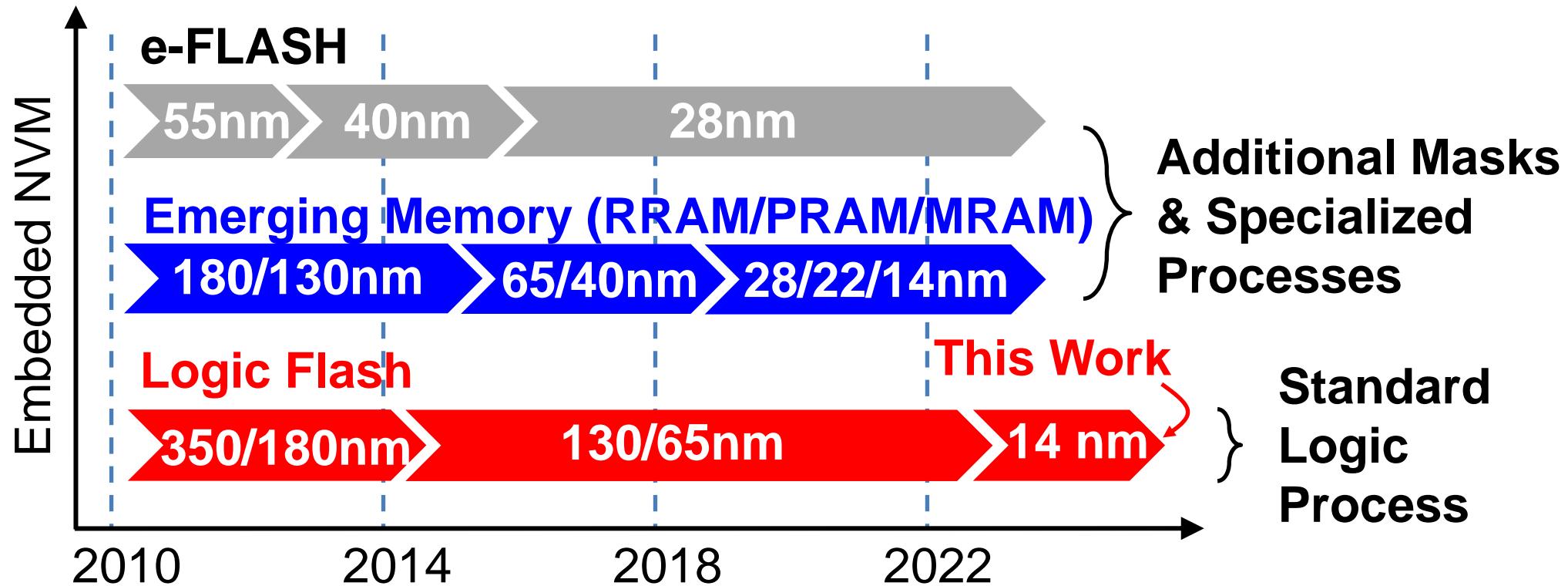
Parameters	Storage	Computation
Plasticity coefficient $\alpha_{i,j}$	NVM for long-term information	Matrix Element-wise Multiplication (MEM) and
Hebbian trace $Hebb_{i,j}$	VM for short-term information	Matrix-Vector Multiplication (MVM)

## ■ Plastic NN\* learns from experience:

- Long-/short-term information expects non-volatile/volatile memory
- Matrix element-wise multiplication for the plastic connections

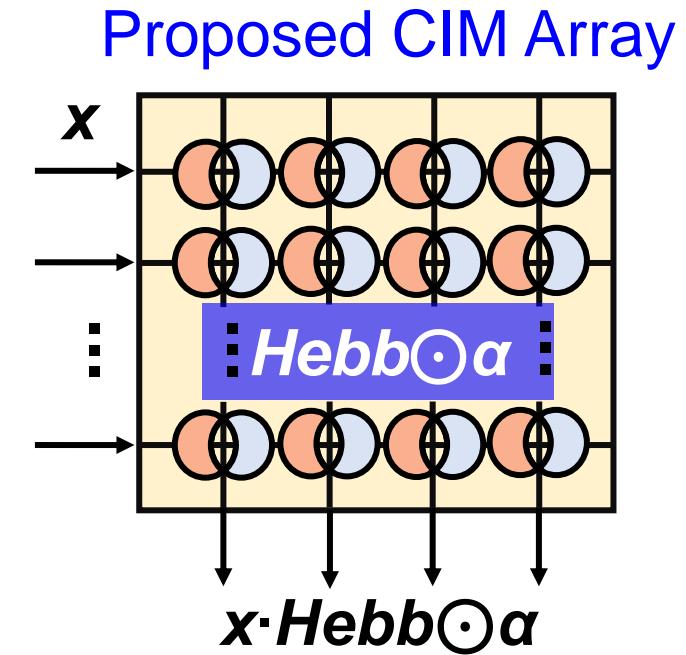
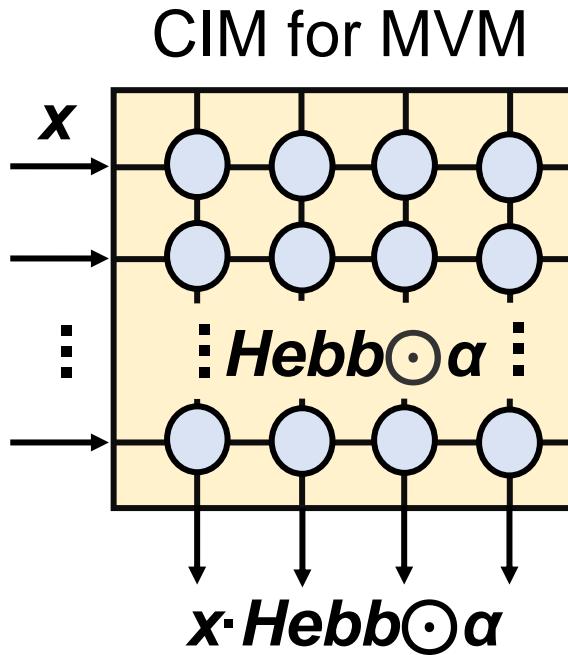
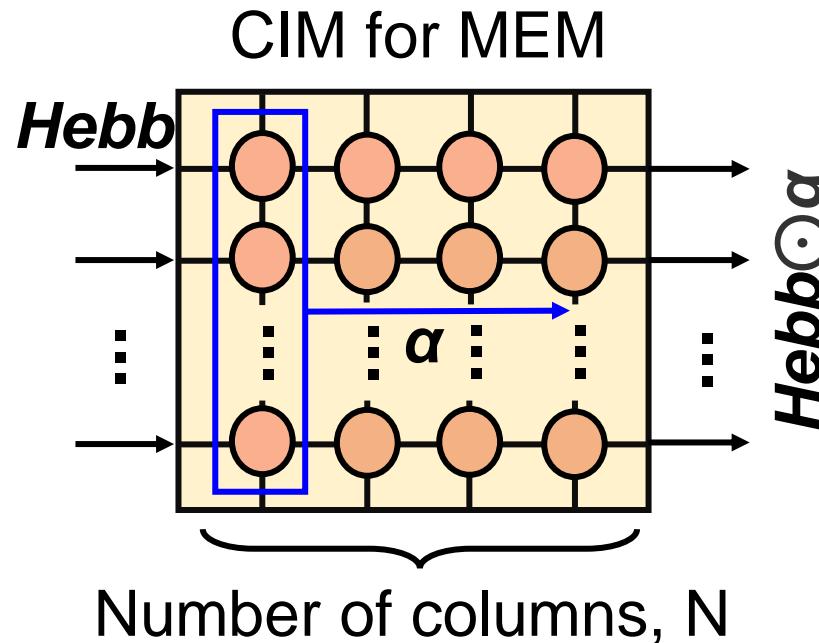
\*[T. Miconi , ICML, 2018]

# Challenge 1: NVM Solution in Advanced Nodes



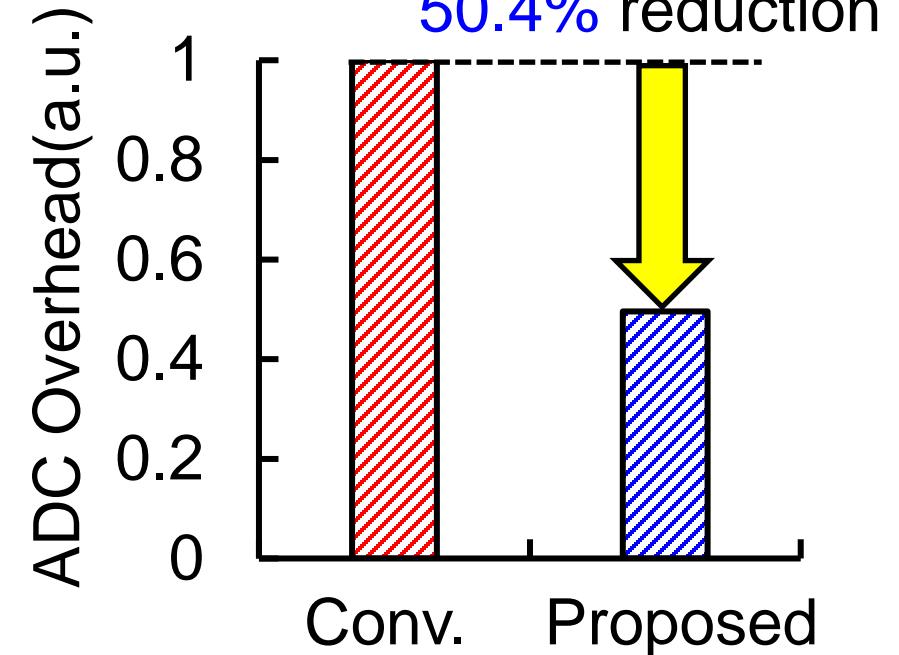
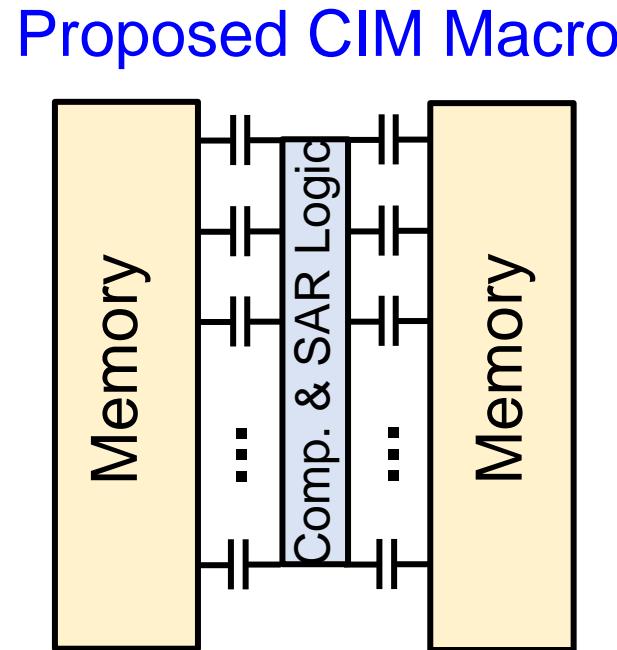
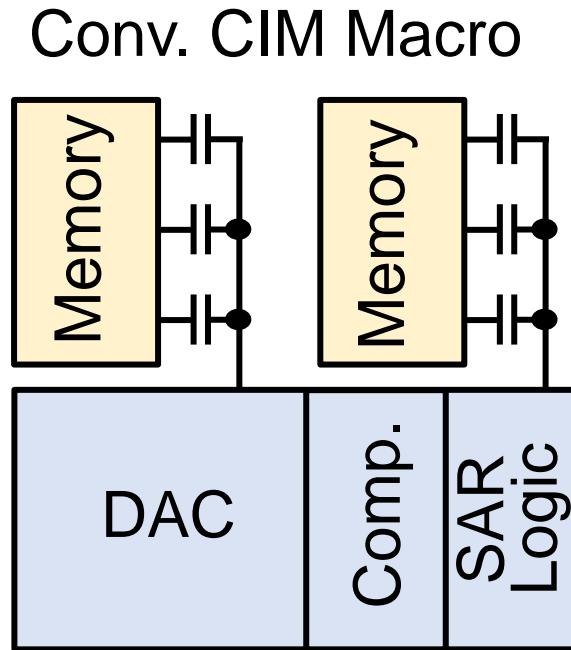
- 5 Transistors Logic-FLASH (5T-LF) provides a cost-effective solution for non-volatile memory in advanced logic platforms

# Challenge 2: MEM-MVM solution in CIM



- Previous works need **(N+1)** cycles to perform MEM-MVM
- Proposed CIM Array supporting MEM-MVM in **1** cycle

# Challenge 3: Low hardware cost ADC solution



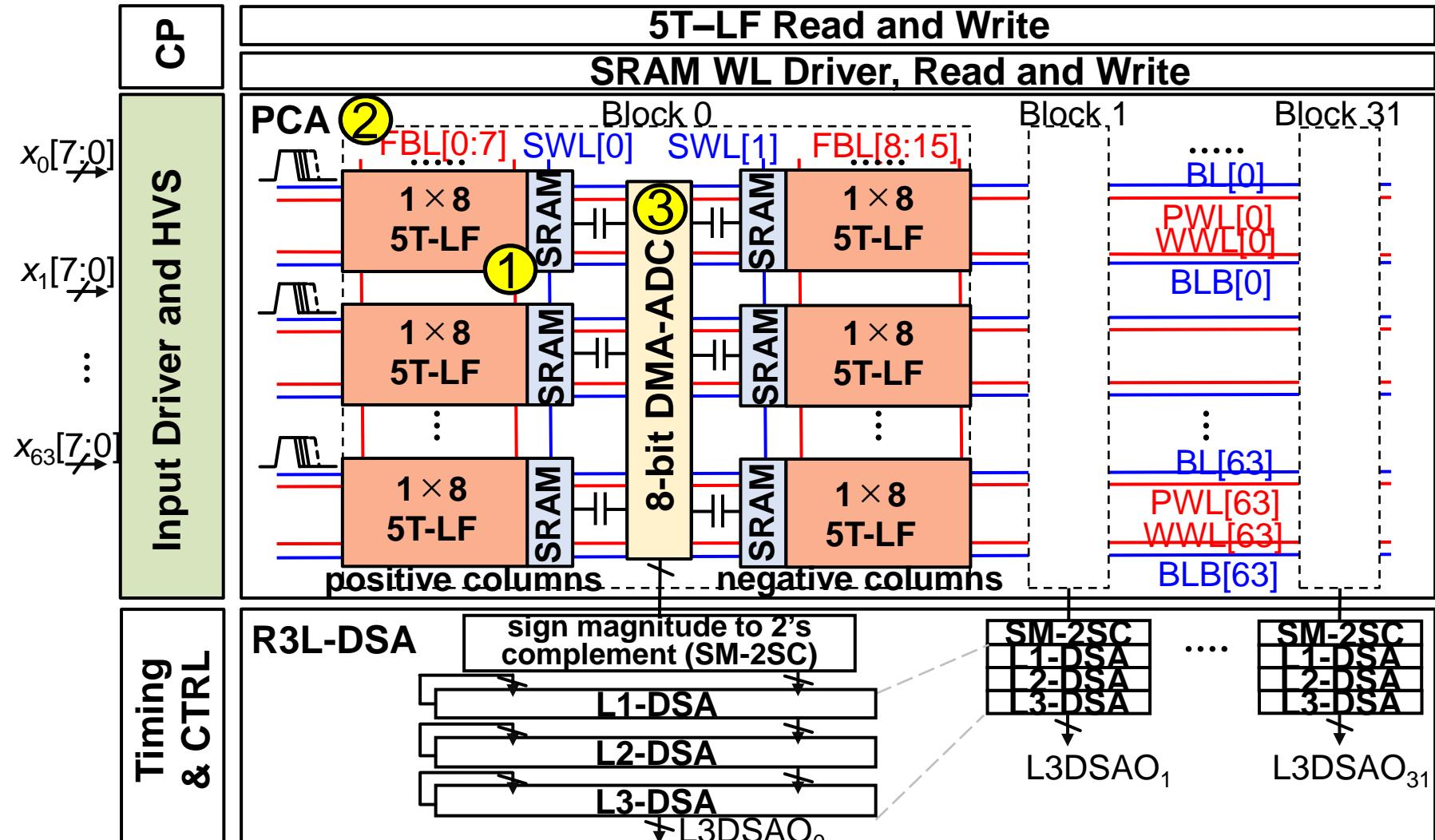
- The area cost of DAC array exponentially increases with ADC resolution, incurring area-efficiency degradation
- Proposed CIM macro saves 50.4% ADC area by reusing MAC signal sampling capacitors

# Outline

- Motivation and Challenges
- Proposed Plastic-CIM Macro
  - Overview of Plastic-CIM Macro
  - Plastic Cell Array
  - Differential Merged-into-Array ADC
- Performance and Measurement Results
- Conclusion

# Overview of Plastic-CIM Macro

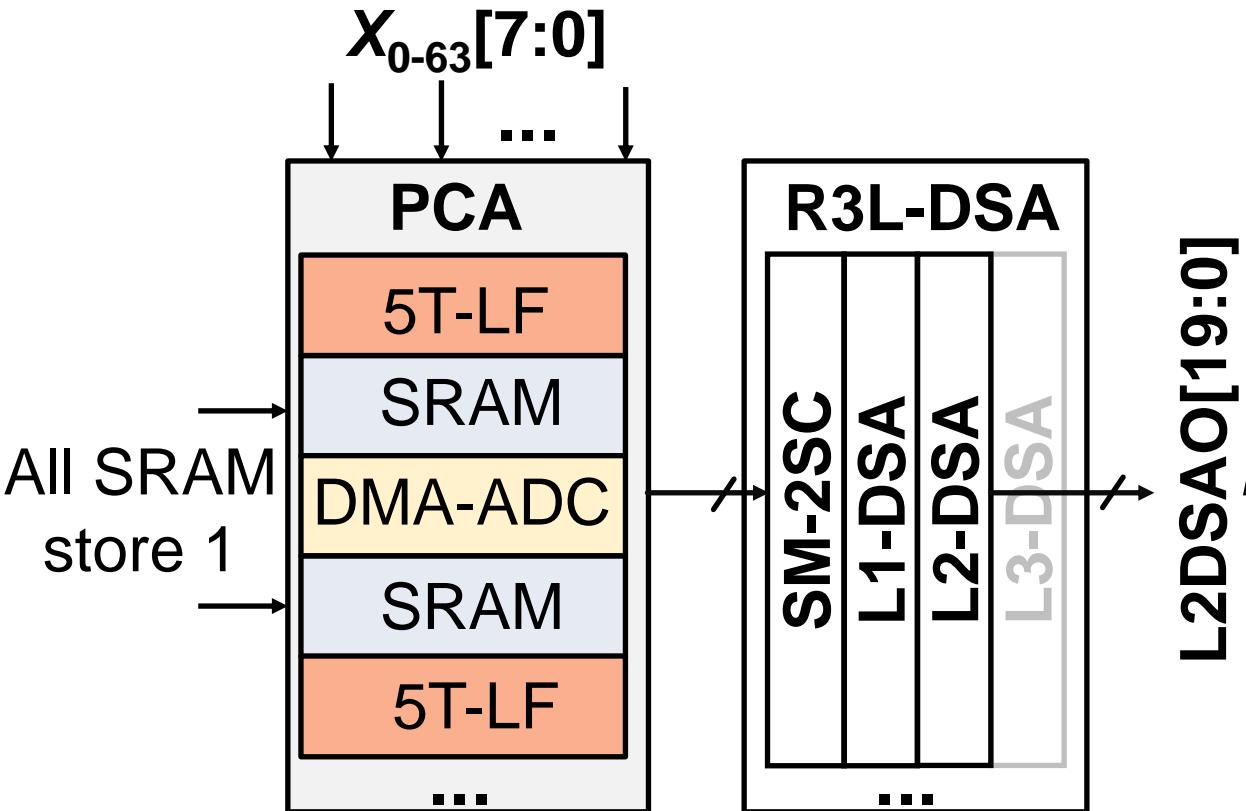
- ① 14nm fused  
32Kb 5T-LF&  
4Kb SRAM
- ② PCA for  
efficiently  
MEM-MVM
- ③ DMA-ADC  
with reduced  
hardware  
overhead



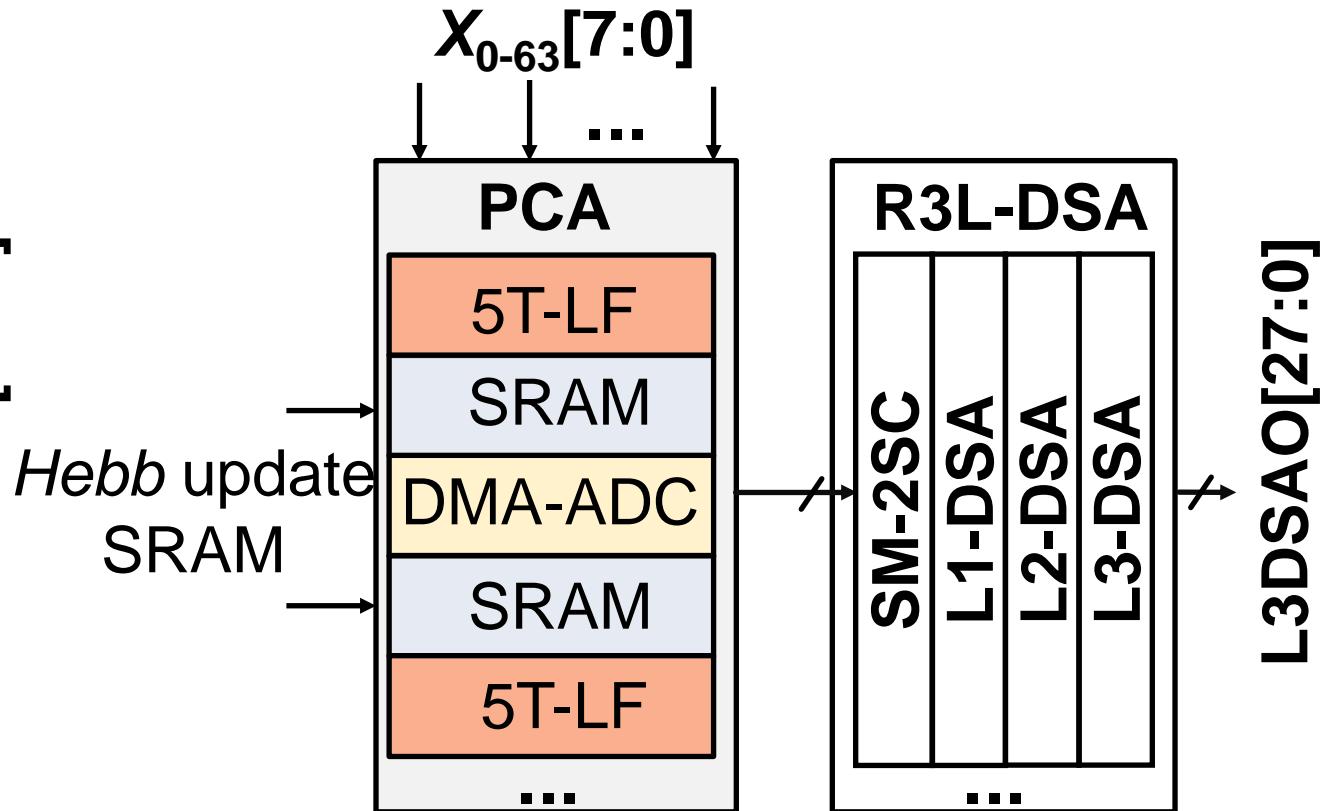
CP, charge pump; HVS, high voltage switches; R3L-DSA, reconfigurable 3-level digital shifter-and-adder; L3DSA<sub>O</sub>, 3<sup>rd</sup>-level DSA output.

# Computation Flow of P-CIM Macro

## MVM Mode



## MEM-MVM Mode

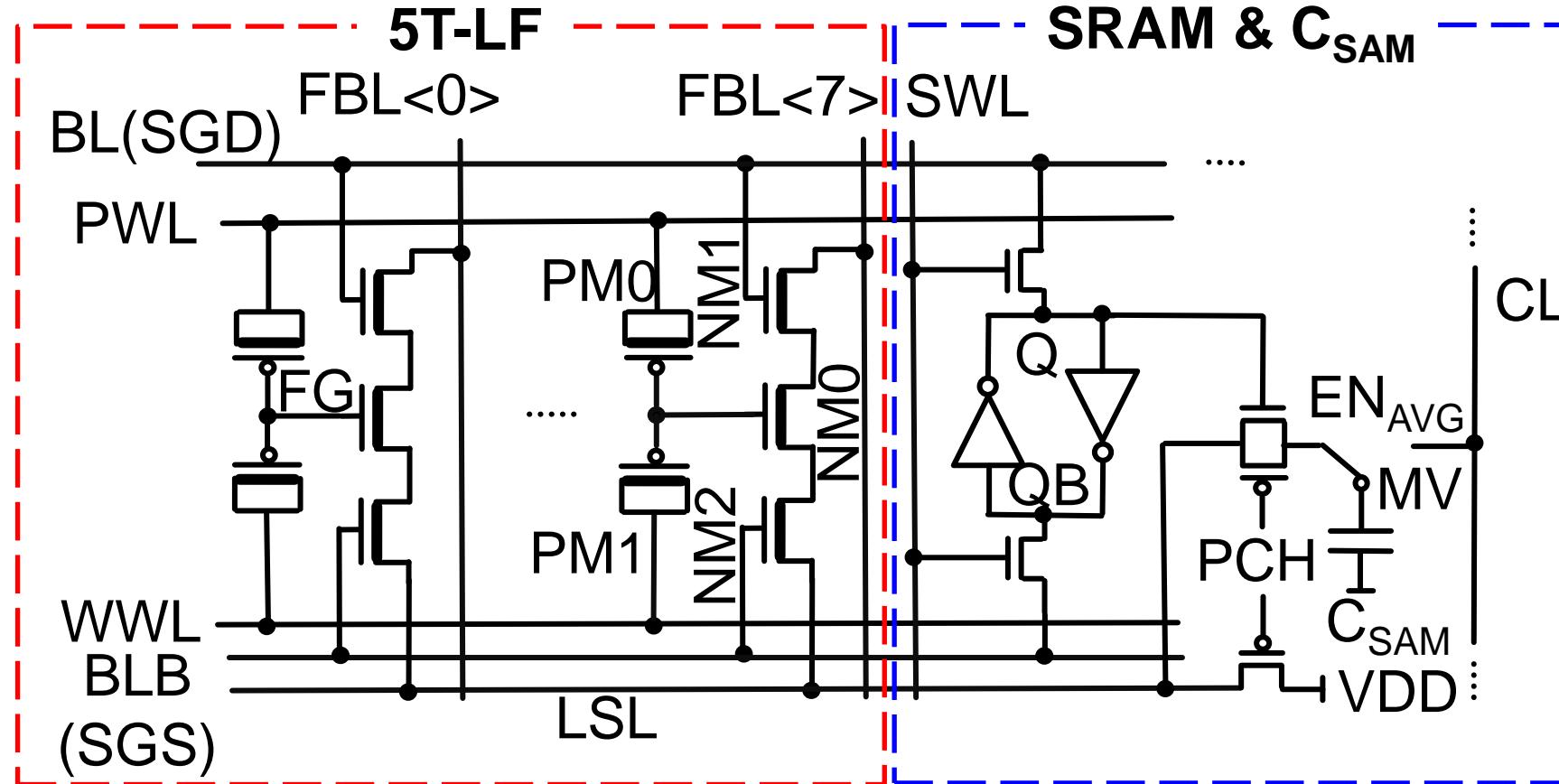


- MVM mode for traditional connections, MEM-MVM mode for plastic connections

# Outline

- Motivation and Challenges
- Proposed Plastic-CIM Macro
  - Overview of Plastic-CIM Macro
  - Plastic Cell Array
  - Differential Merged-into-Array ADC
- Performance and Measurement Results
- Conclusion

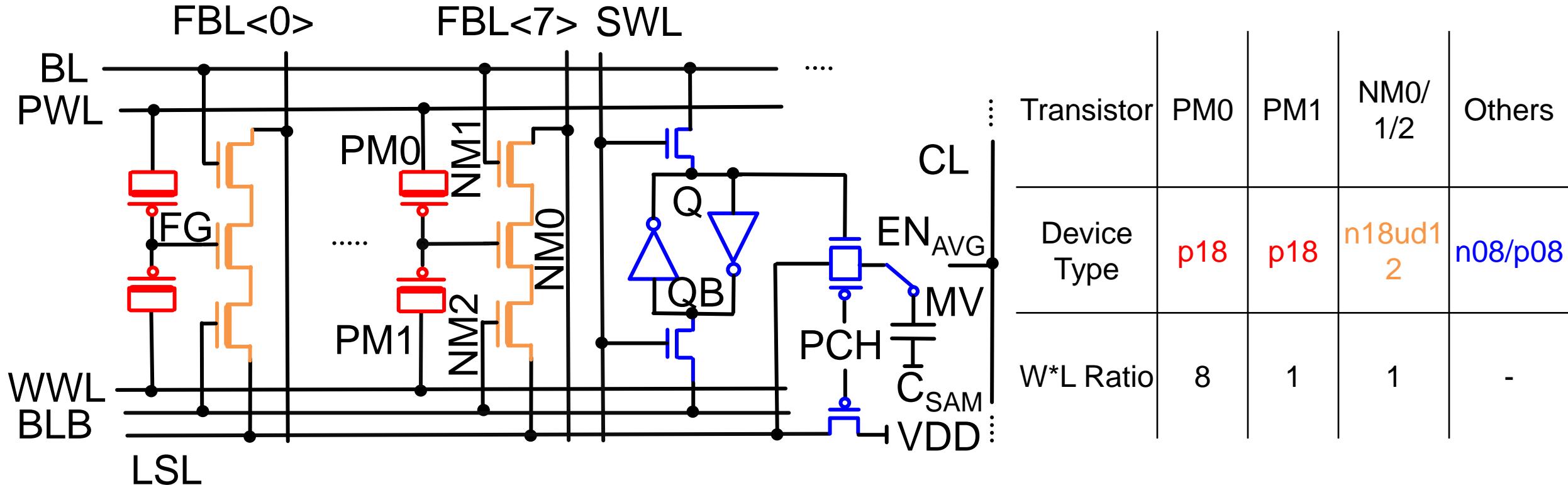
# PCA Unit Structure



\* 5T-LF cell  
refers to [M. Kim,  
JSSC, 2022]

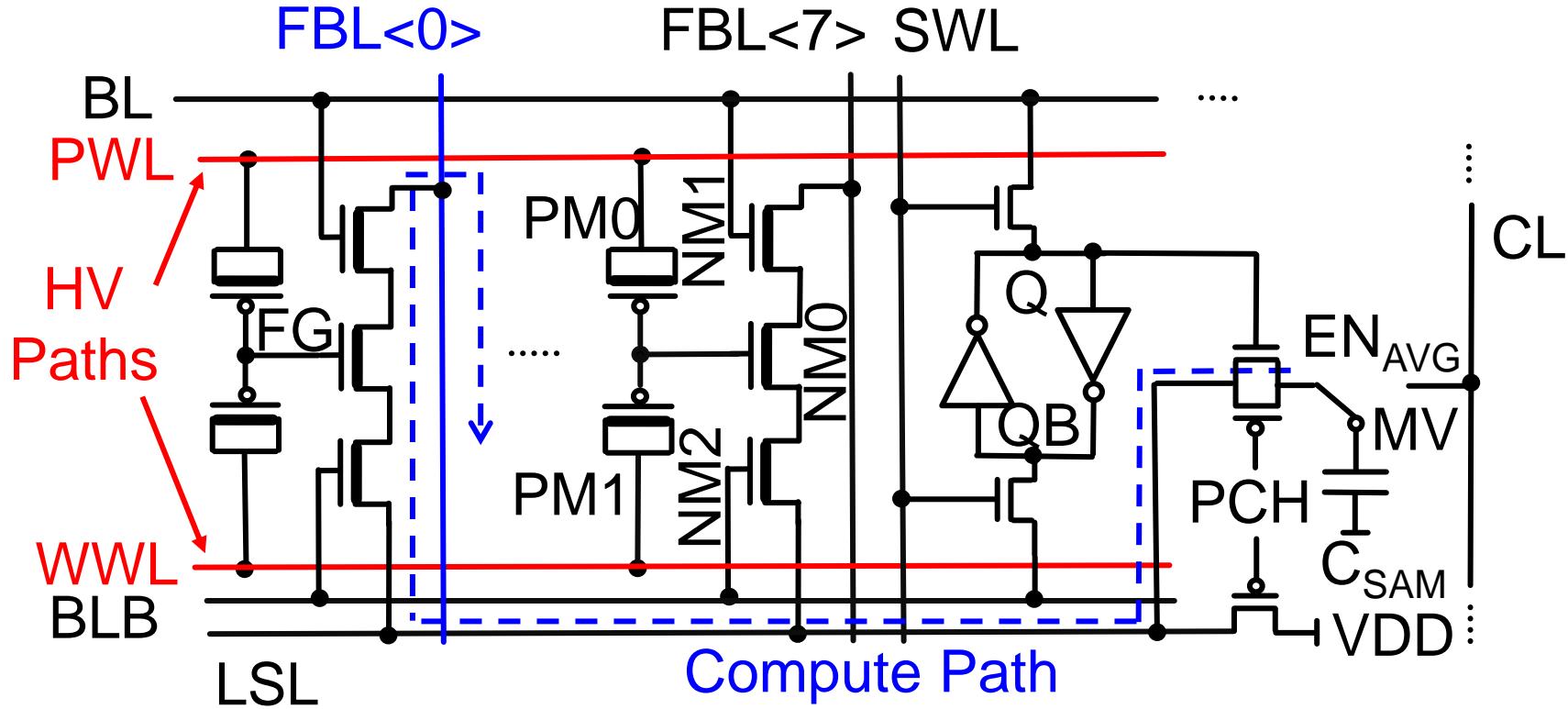
- Merged 5T-LF\* and SRAM cells thanks to the process compatibility
- Shared SGD/SGS and BL/BLB to reduce the number of metal lines

# PCA Unit Structure



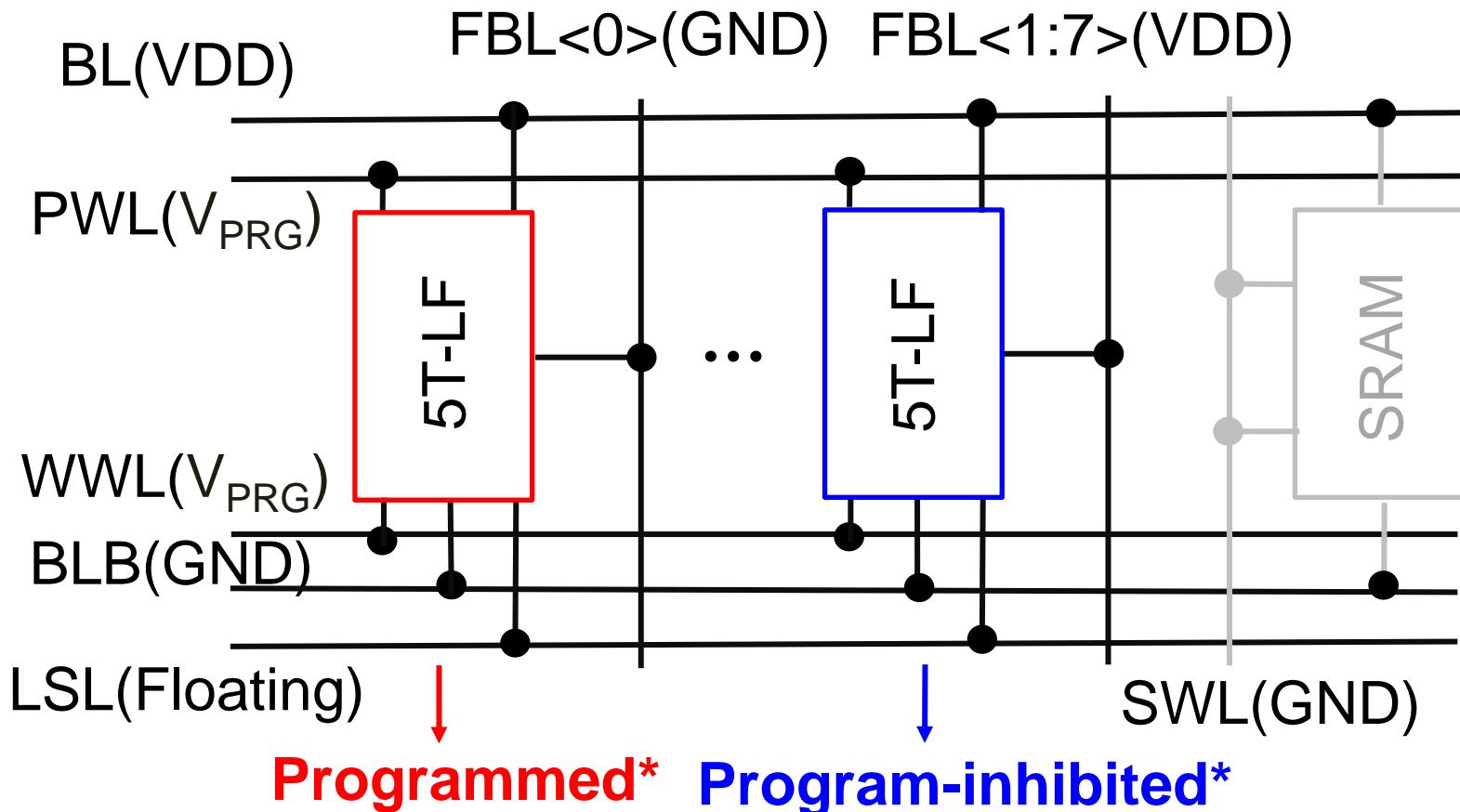
- 5T-LF cells built by different IO devices
- Under-drive IO transistors for the read and pass gates in the 5T-LF to increase the cell read currents

# PCA Unit Structure



- SRAM and computing switches are built by the core devices
- HV paths decoupled from SRAM,  $C_{SAM}$ , and the computing path

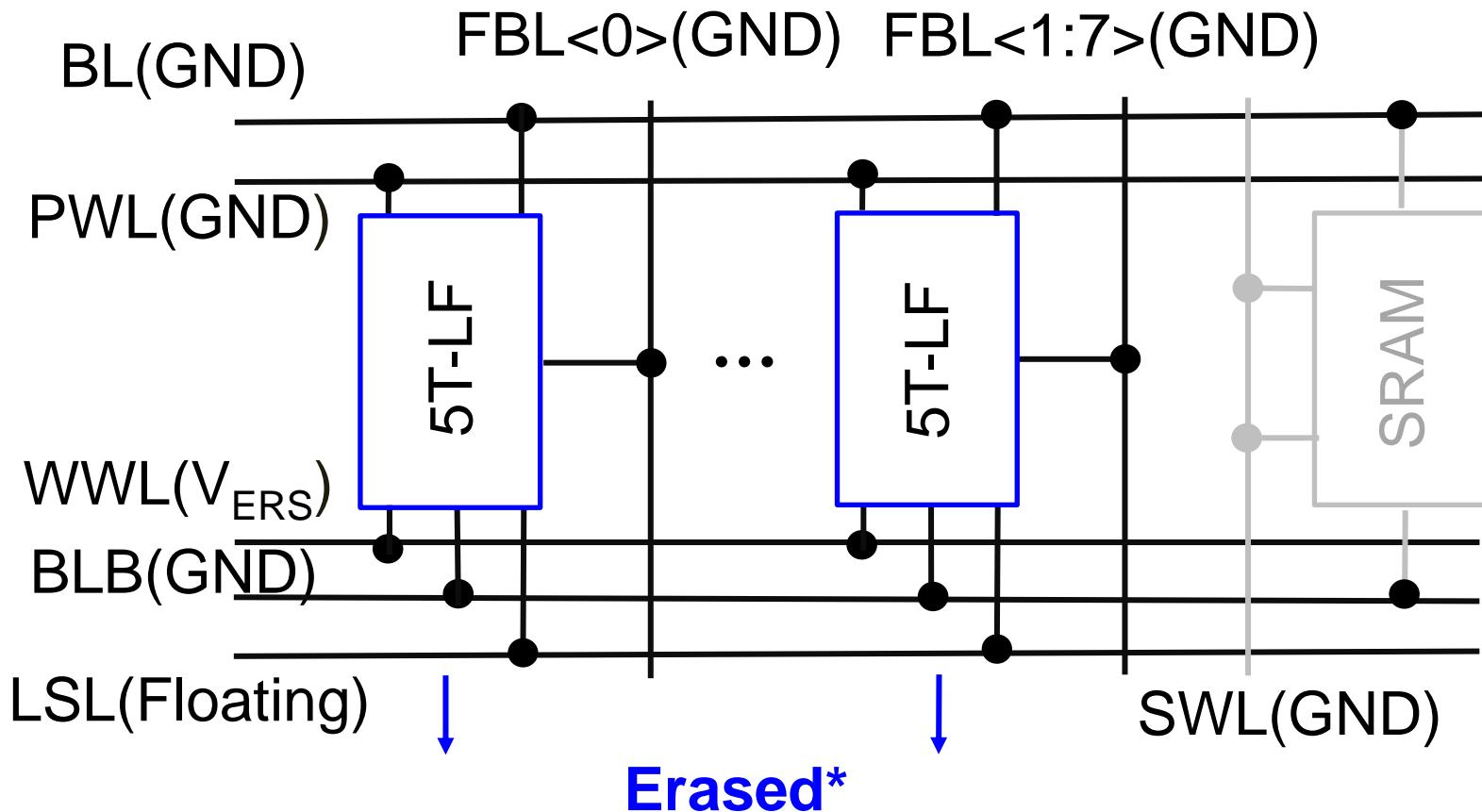
# 5T-LF Programing



\* 5T-LF cell  
refers to [M. Kim,  
JSSC, 2022]

- Bit-wise programing by applying  $V_{PRG}$  on PWL/WWL, and inhibit the half-selected cells by applying  $V_{DD}$  on BL

# 5T-LF Erasing

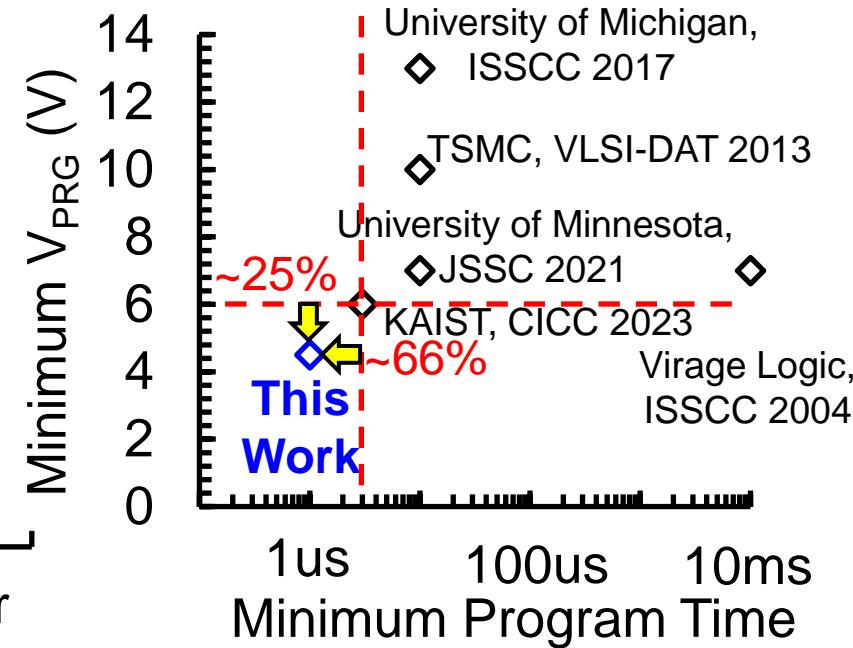
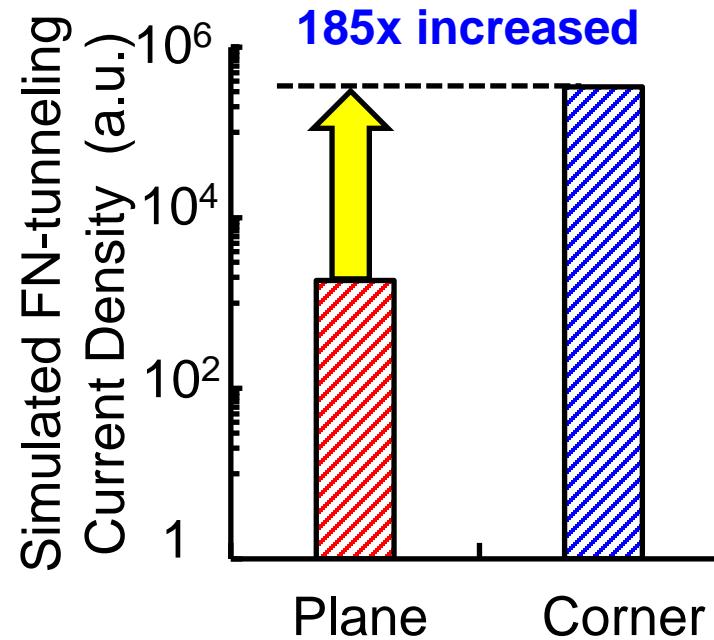
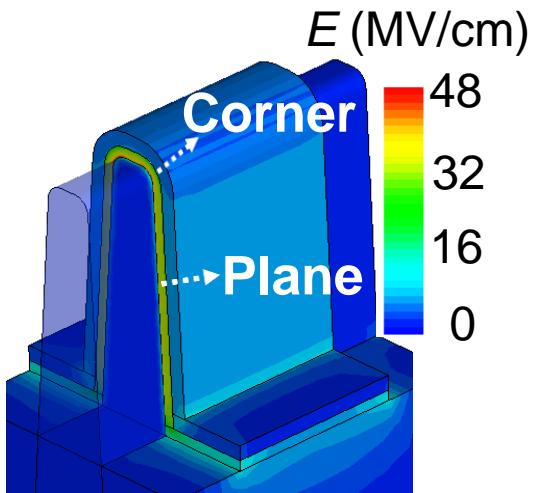
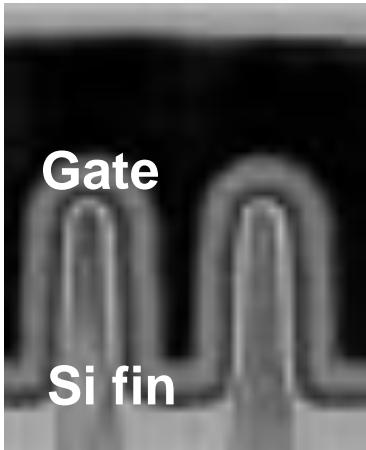


\* 5T-LF cell  
refers to [M. Kim,  
JSSC, 2022]

- Row-wise erasing of 5T-LF by applying  $V_{ERS}$  between WWL and PWL

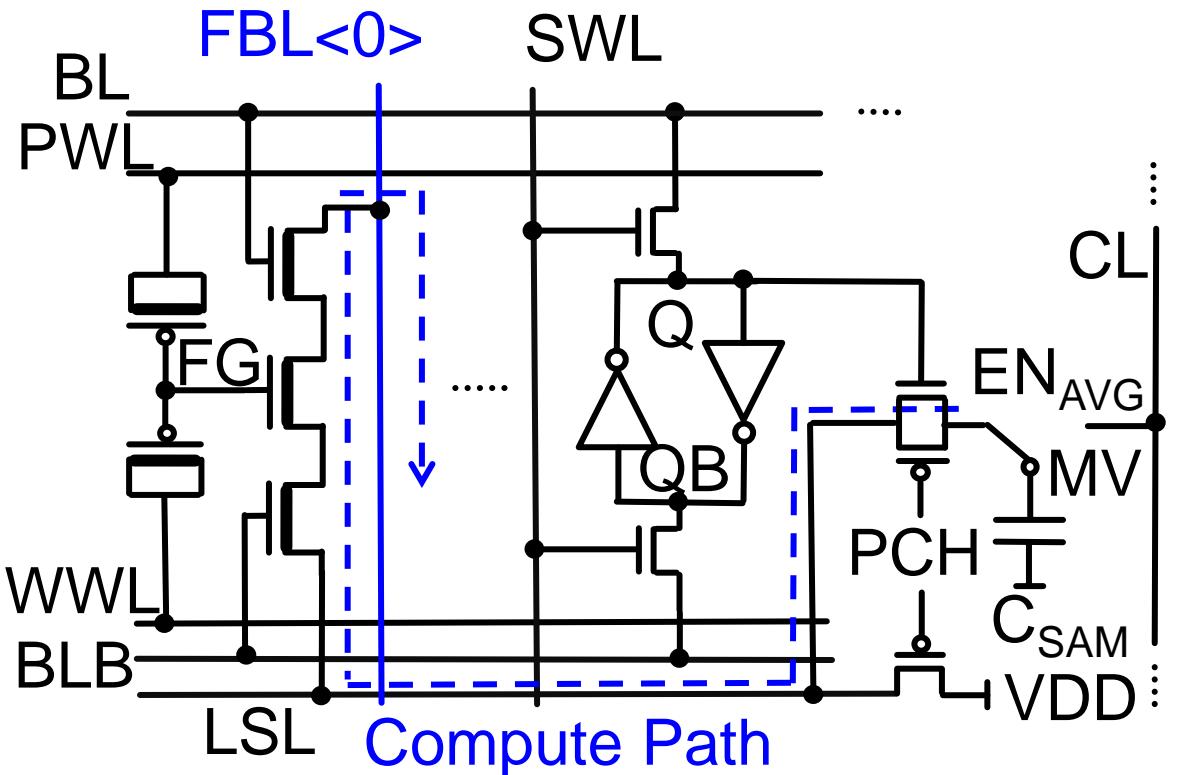
# Enhanced FN Tunneling in the Si Fin Structure

## TEM Cross-Sectional View and TCAD Electric Field Simulation



- Enhanced electric field during programming at the Si fin corner, leading to 185x enhanced local FN tunneling current
- 25% reduced minimum  $V_{PRG}$  and 66% reduced minimum time

# Computing of PCA Unit - Truth Table

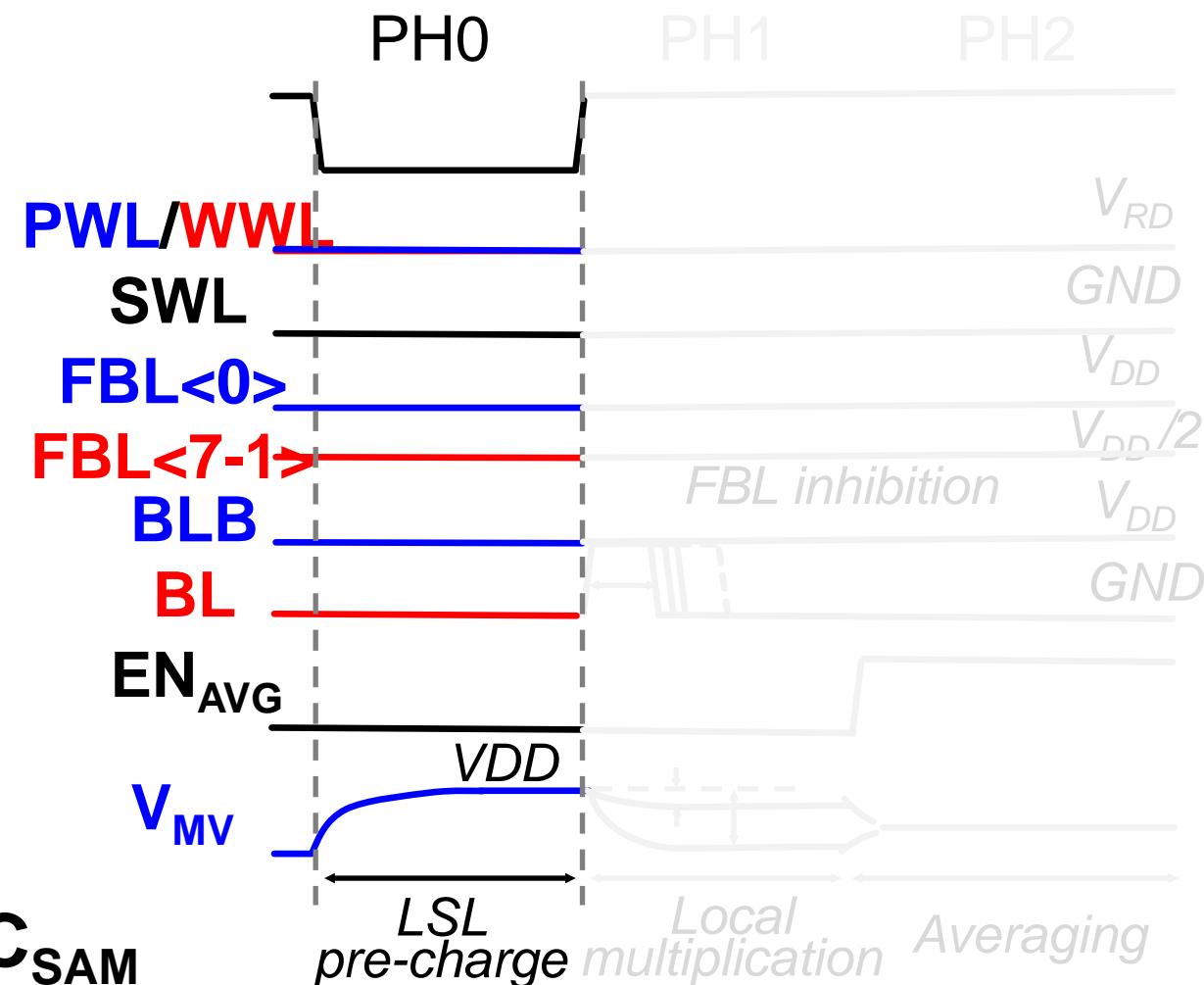
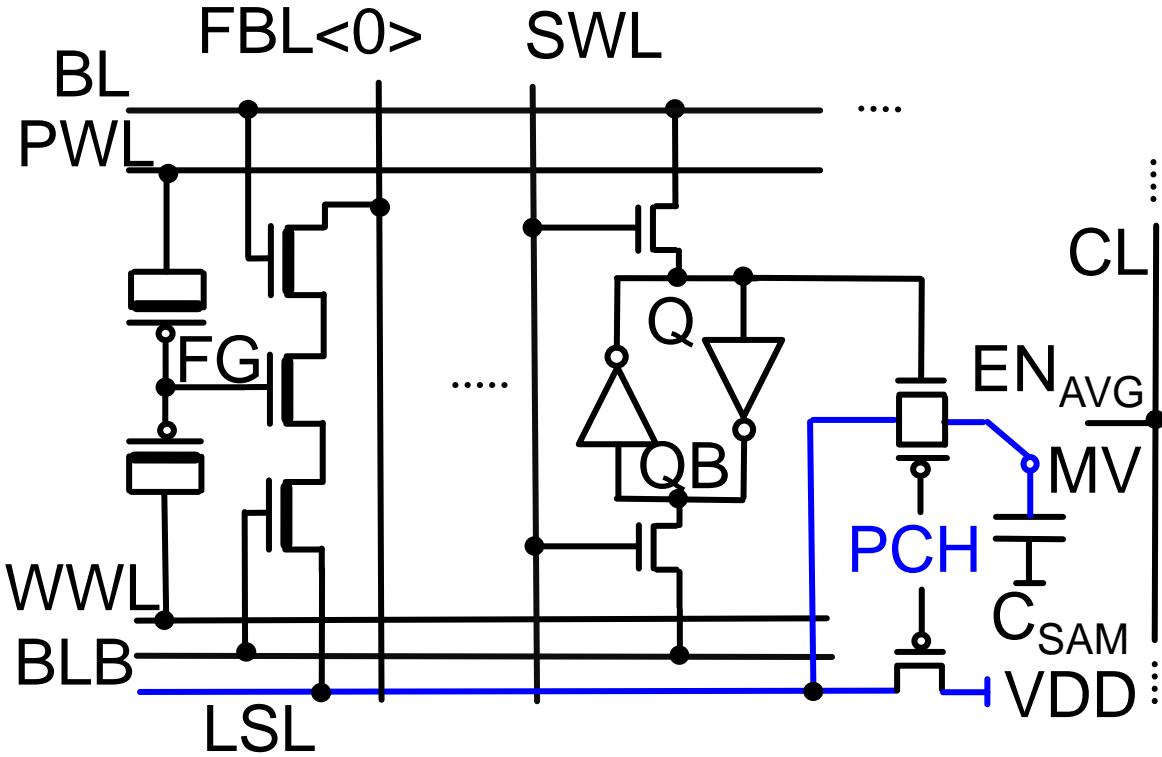


$x$ (BL Pulse Width <sup>*1</sup> )	$\alpha$ or $w$ (5T-LF $V_{TH}$ <sup>*2</sup> )	Hebb (SRAM Q)	Multiplication Value ( $V_{MV}$ )
00 ( $t_{ACT} = 0$ )	-	-	
-	0000 ( $V_{TH0}$ )	-	00000 (VDD)
-	-	0 (GND)	
01 ( $t_{ACT} = t_0$ )	0001 ( $V_{TH1}$ )	1 (VDD)	000001 (VDD- $\Delta V$ )
01 ( $t_{ACT} = t_0$ )	0010 ( $V_{TH2}$ )	1 (VDD)	000010 (VDD- $2\Delta V$ )
01 ( $t_{ACT} = t_0$ )	0011 ( $V_{TH3}$ )	1 (VDD)	000011 (VDD- $3\Delta V$ )
10 ( $t_{ACT} = 2t_0$ )	0100 ( $V_{TH4}$ )	1 (VDD)	000001 (VDD- $2\Delta V$ )
...			
11 ( $t_{ACT} = 3t_0$ )	1111 ( $V_{TH15}$ )	1 (VDD)	101101 (VDD- $45\Delta V$ )

\*1.  $t_{ACT}$ : BL activation time;  $t_0$ : the unit time for pulse width modulation.

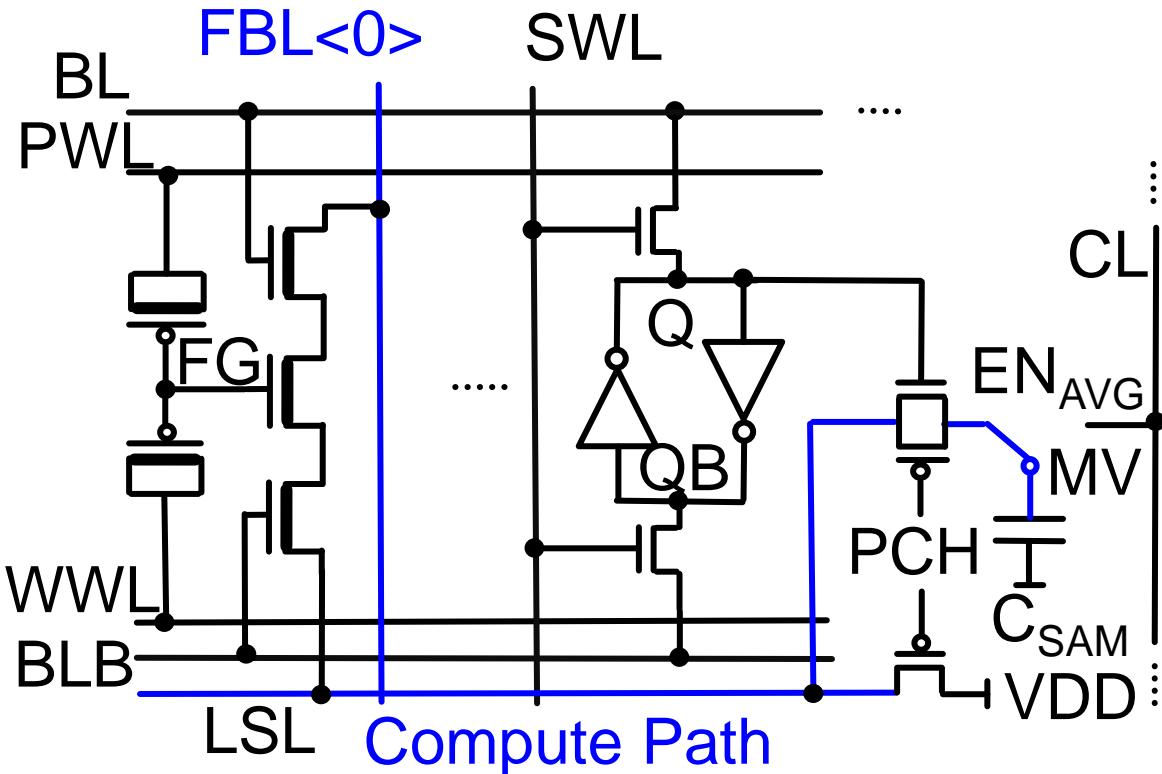
\*2. Cell current linearly increases as the threshold voltage ( $V_{TH}$ ) changes from  $V_{TH0}$  to  $V_{TH15}$ .

# Computing of PCA Unit – Phase0 (Pre-charging)

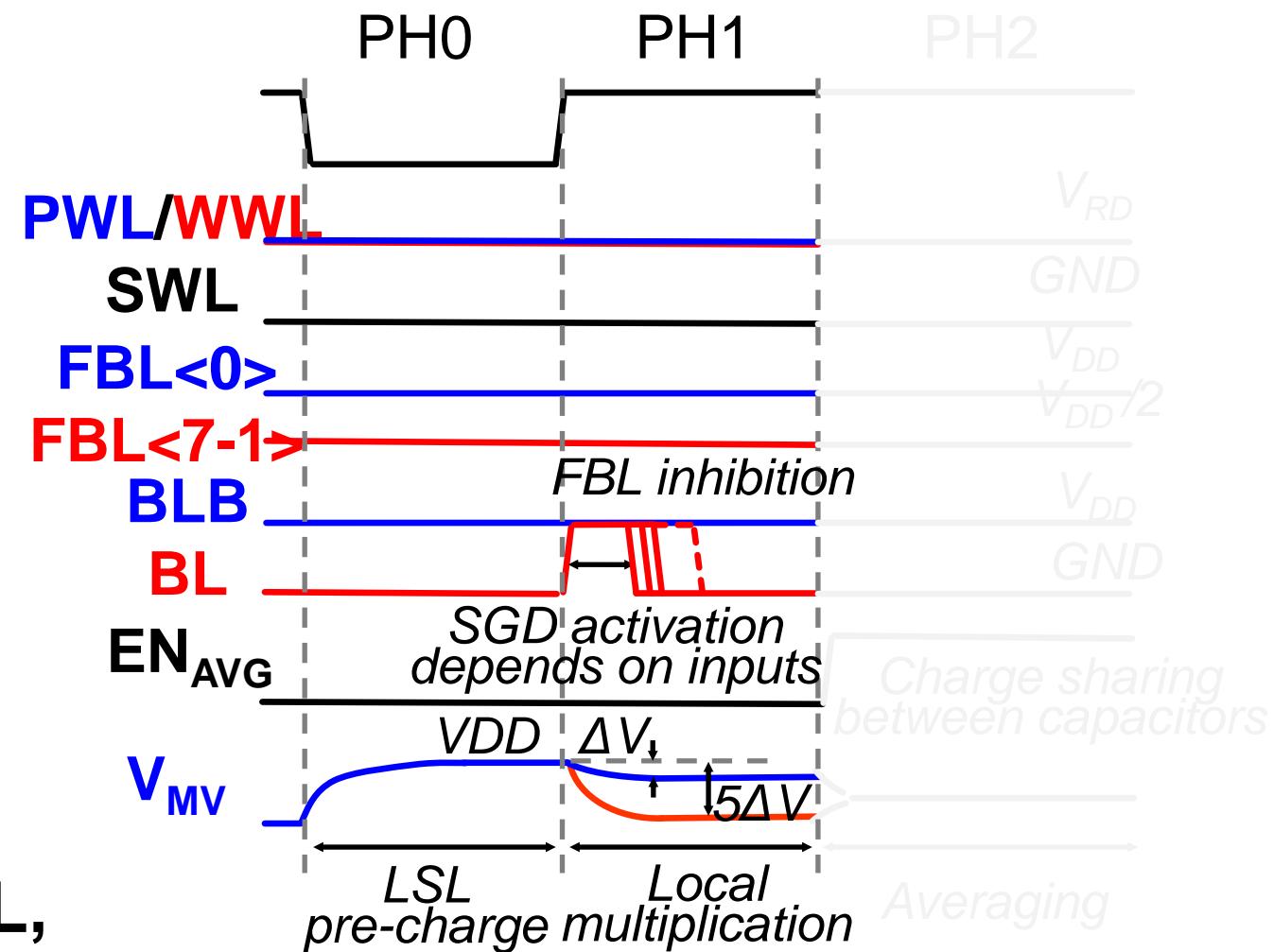


- Pre-charge top-plates of  $C_{SAM}$  and parasitic cap of LSL

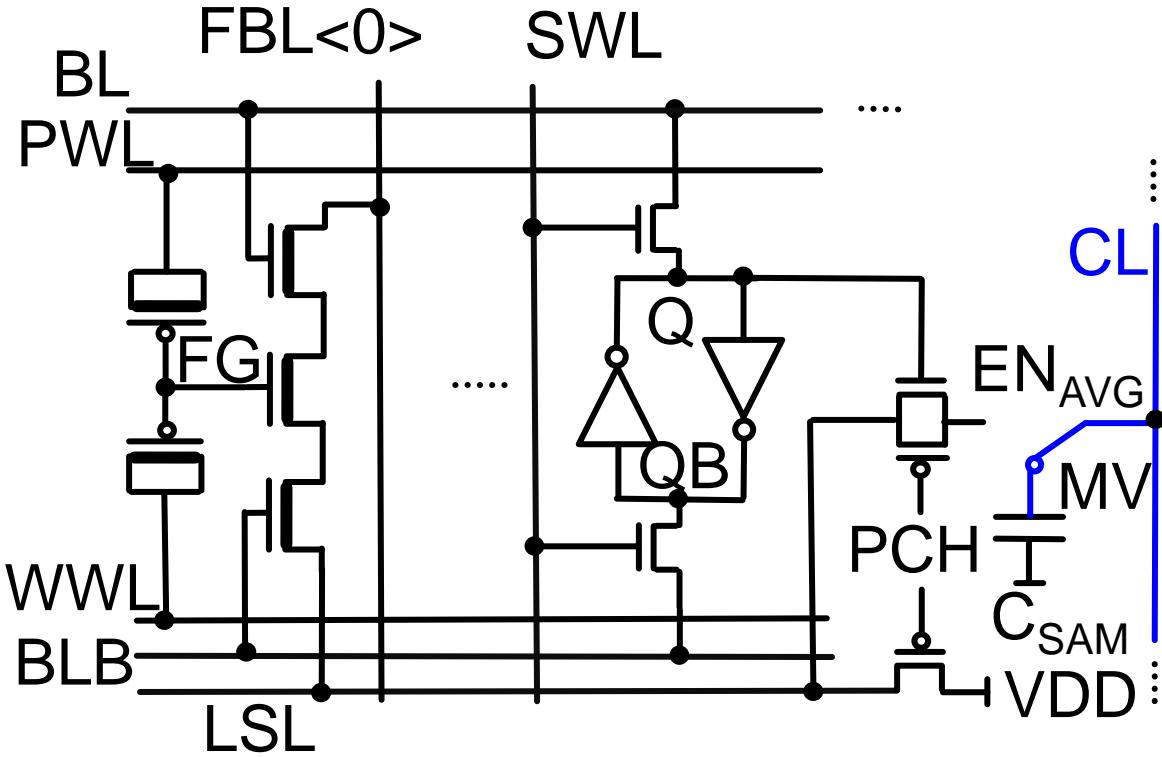
# Computing of PCA Unit – Phase1 (Multiplying)



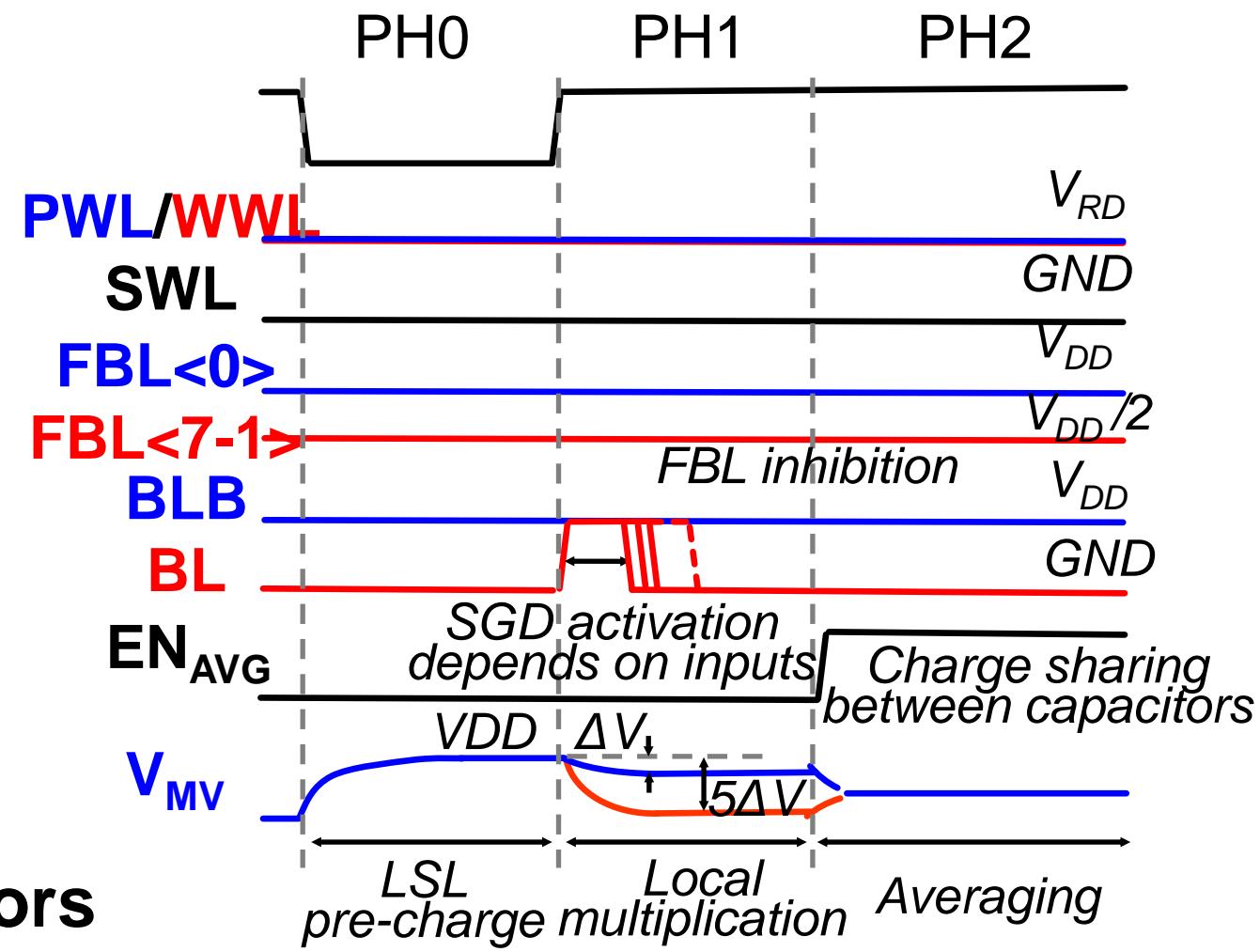
- Input pulses applied to BL, multiply through dis-charge  $C_{SAM}$ s



# Computing of PCA Unit – Phase2 (Averaging)



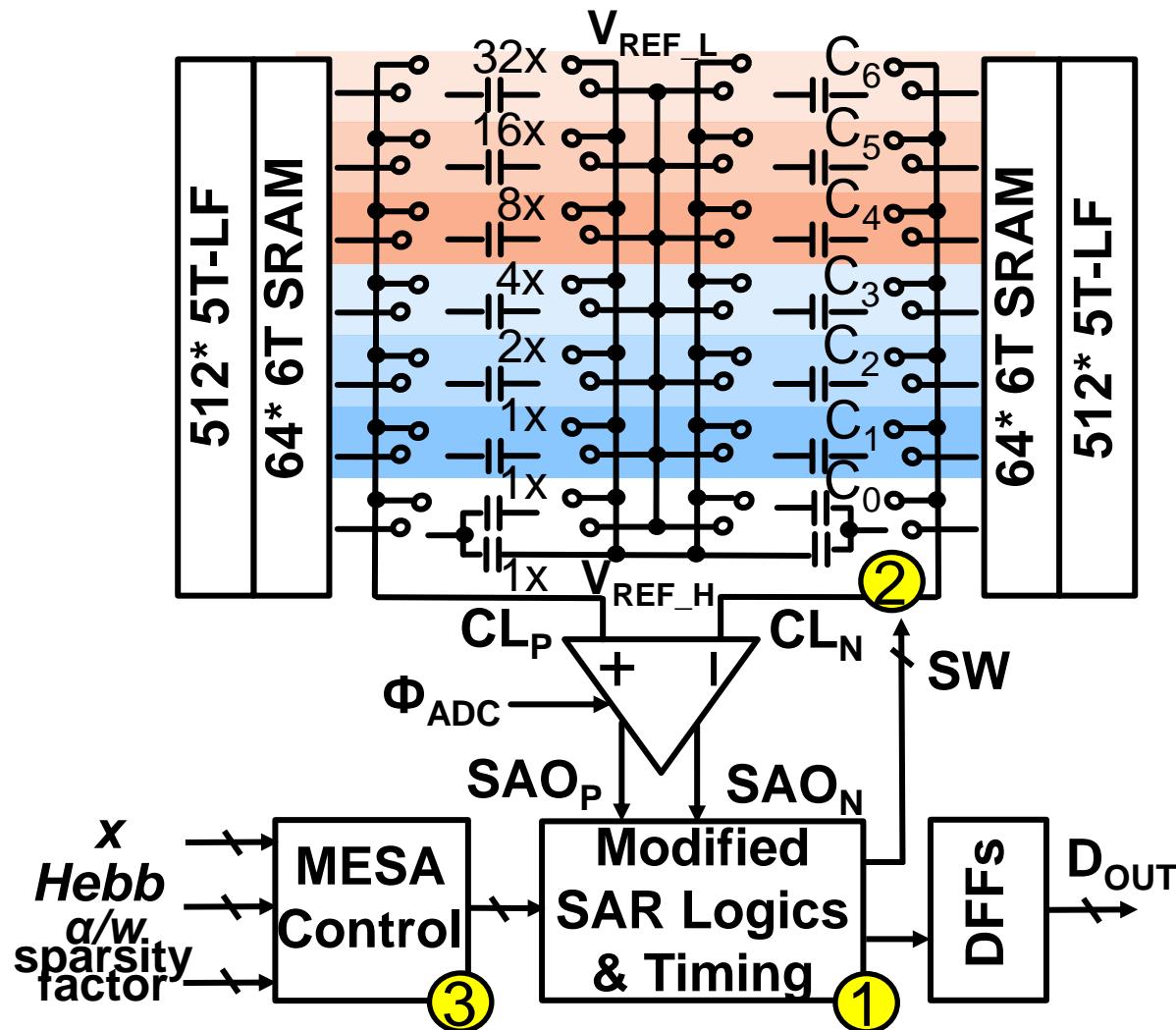
- Average between capacitors through charge sharing



# Outline

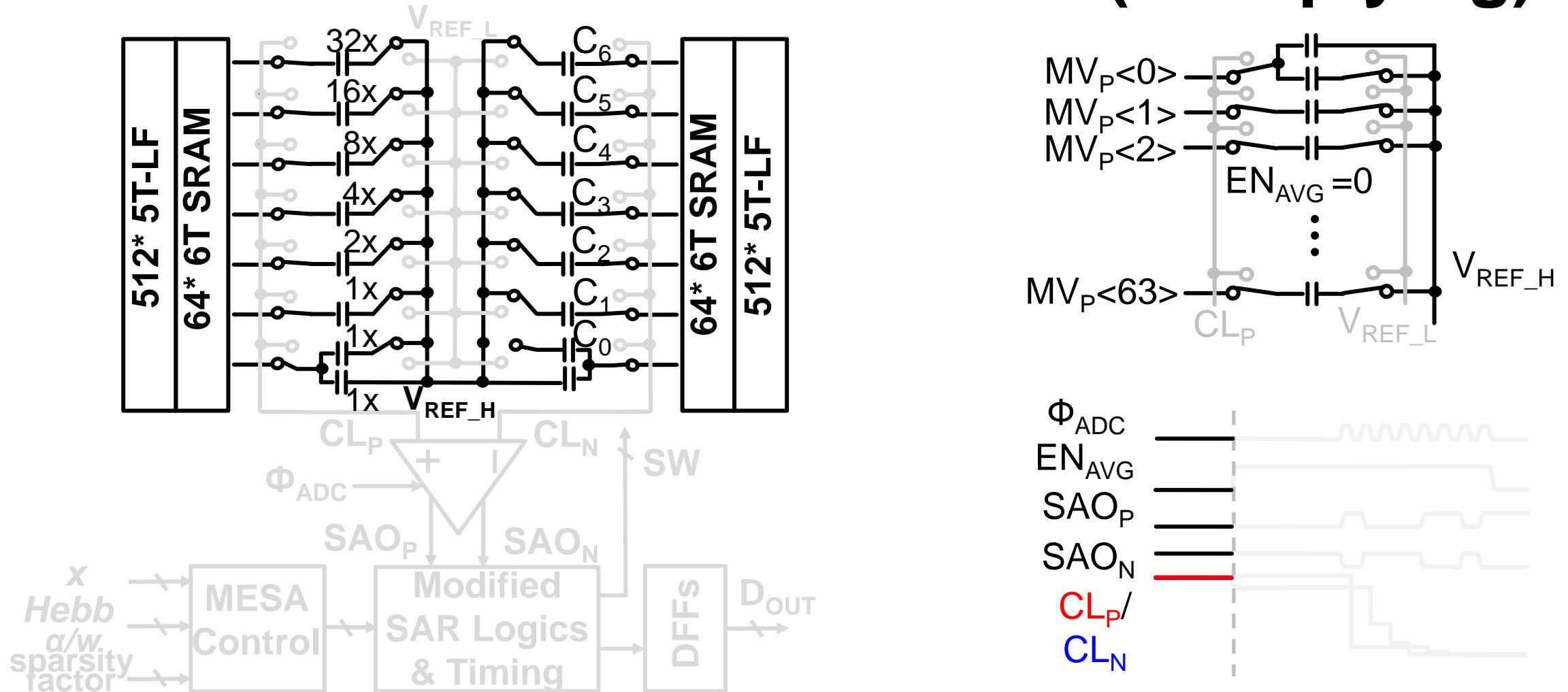
- Motivation and Challenges
- Proposed Plastic-CIM Macro
  - Overview of Plastic-CIM Macro
  - Plastic Cell Array
  - Differential Merged-into-Array ADC
- Performance and Measurement Results
- Conclusion

# DMA-ADC Structure



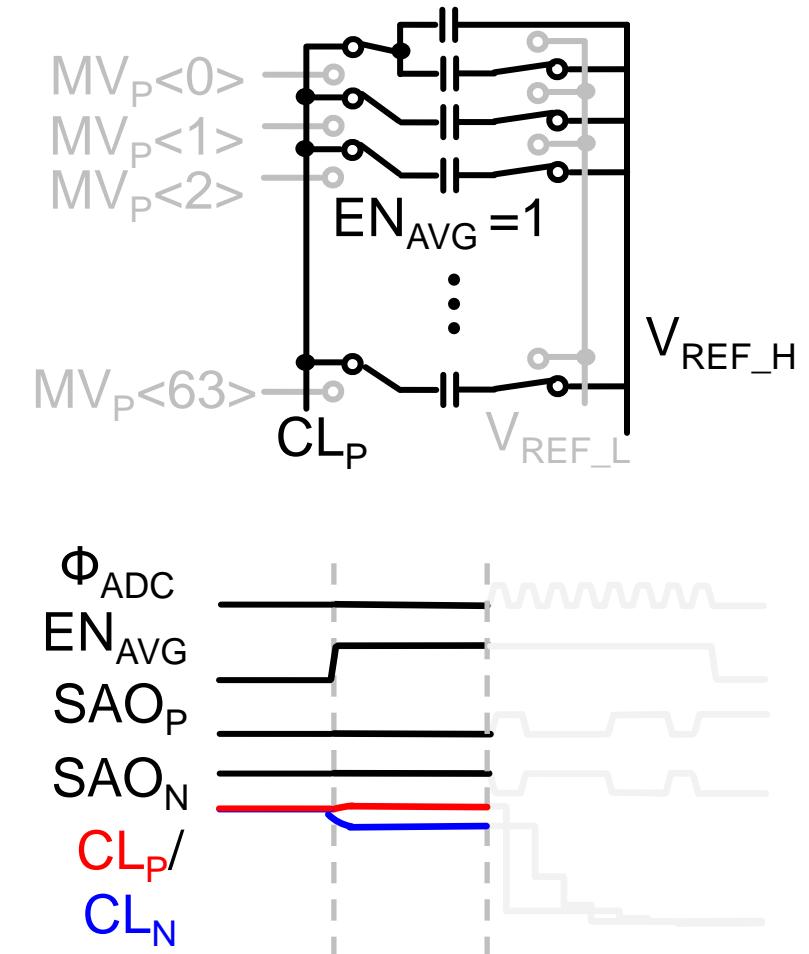
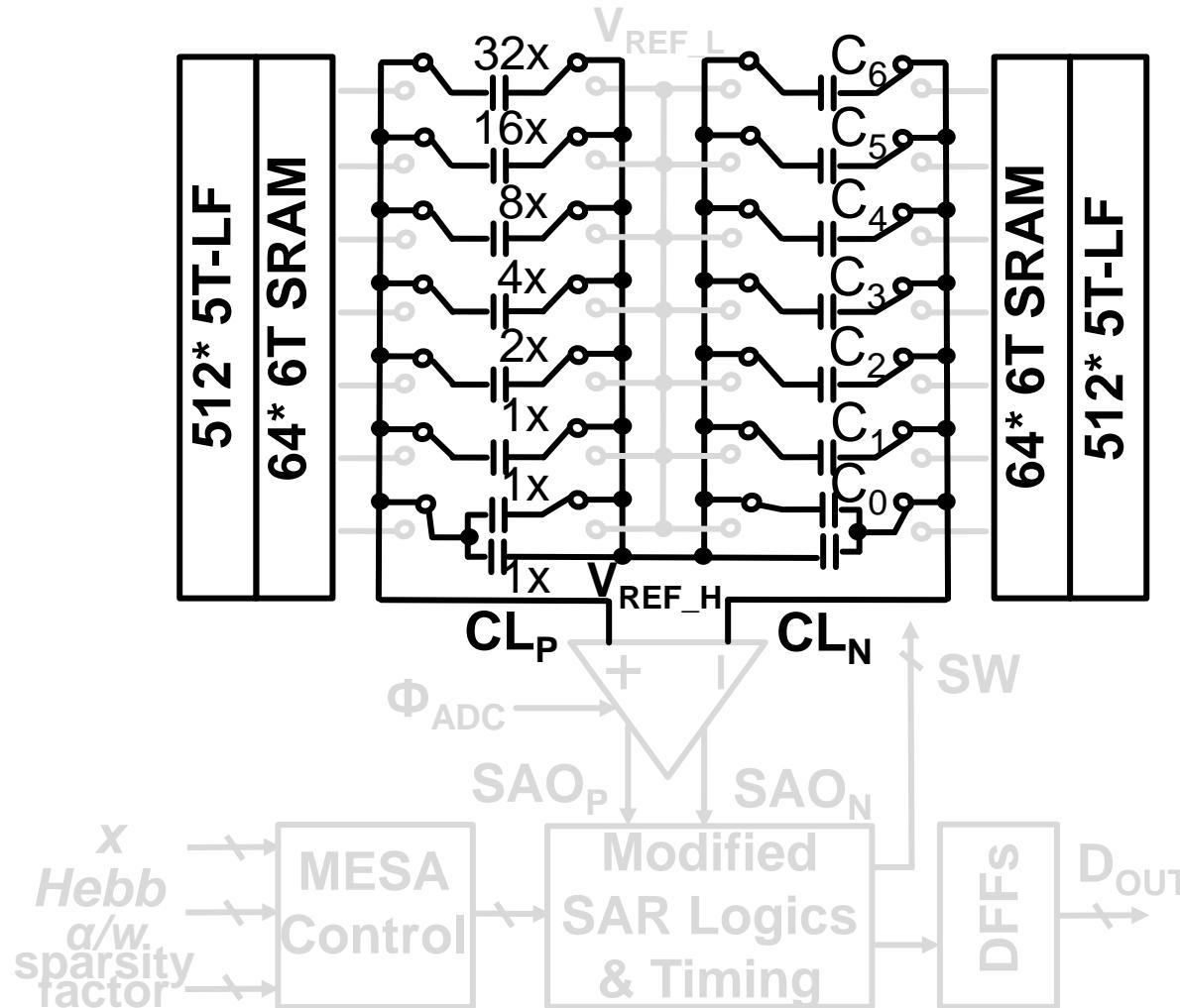
- ① Modified SAR logics for grouping and quantizing
- ② Common-centroid capacitors grouping strategy to reduce mismatch
- ③ Multi-Element Sparsity-Aware (MESA) control to leverage sparsity in plastic-NN

# DMA-ADC Work Flow – Phase1 (Multiplying)



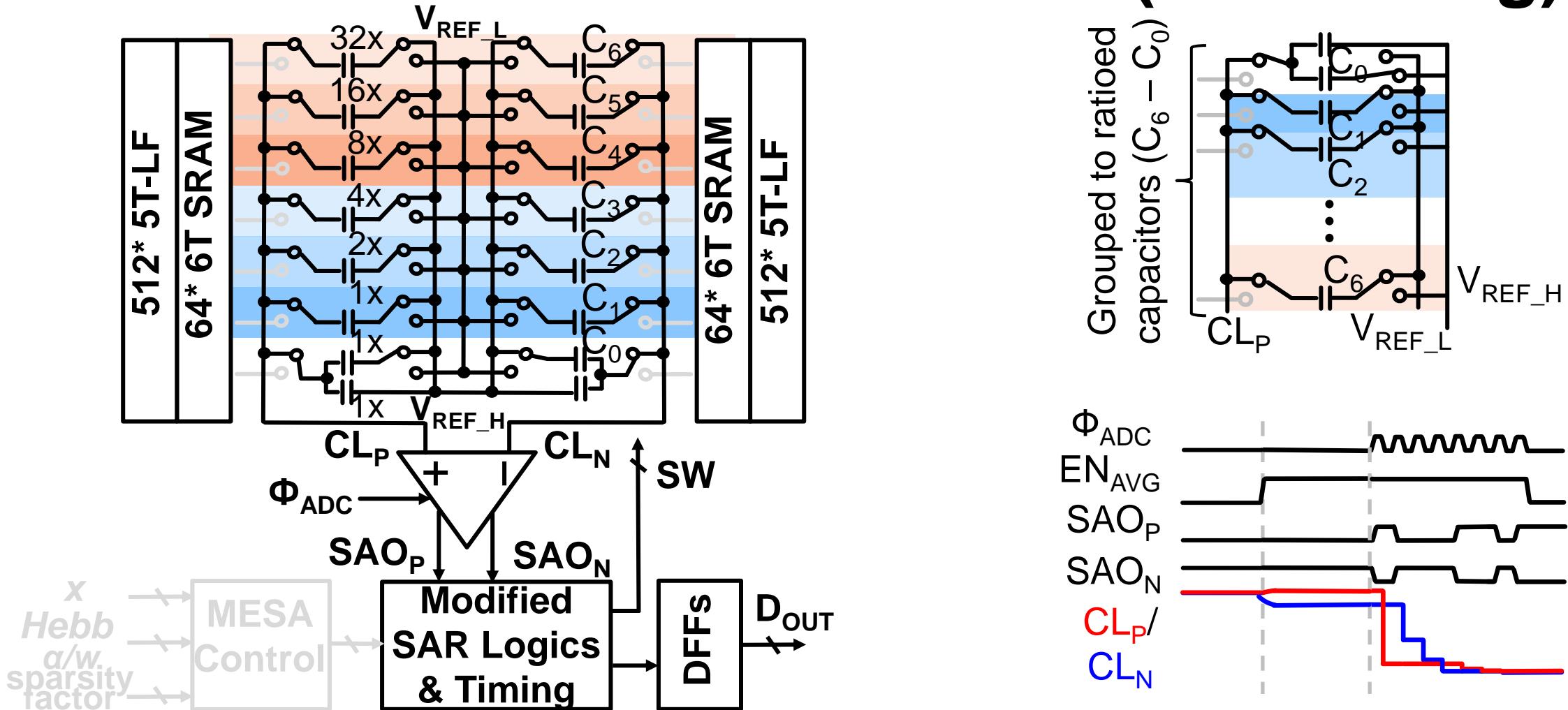
- Top-plate of  $C_{SAM}$  connected to MV, while bottom-plate of  $C_{SAM}$  connected to  $V_{REF\_H}$  because the input common-mode voltage is near VDD

# DMA-ADC Work Flow – Phase2 (Averaging)

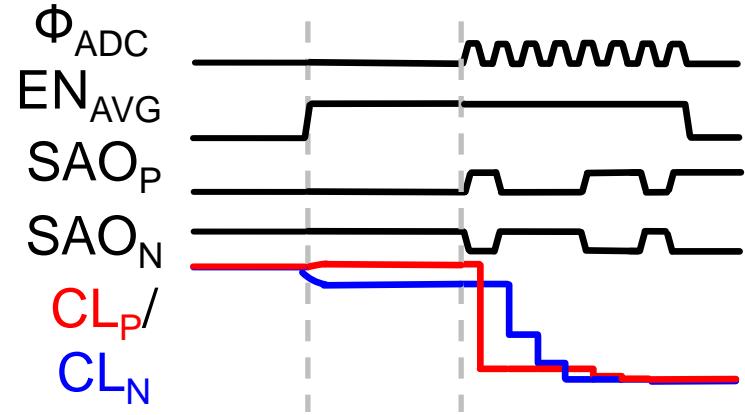


- Top-plate of  $C_{SAM}$  connected to CL to enable the charge sharing

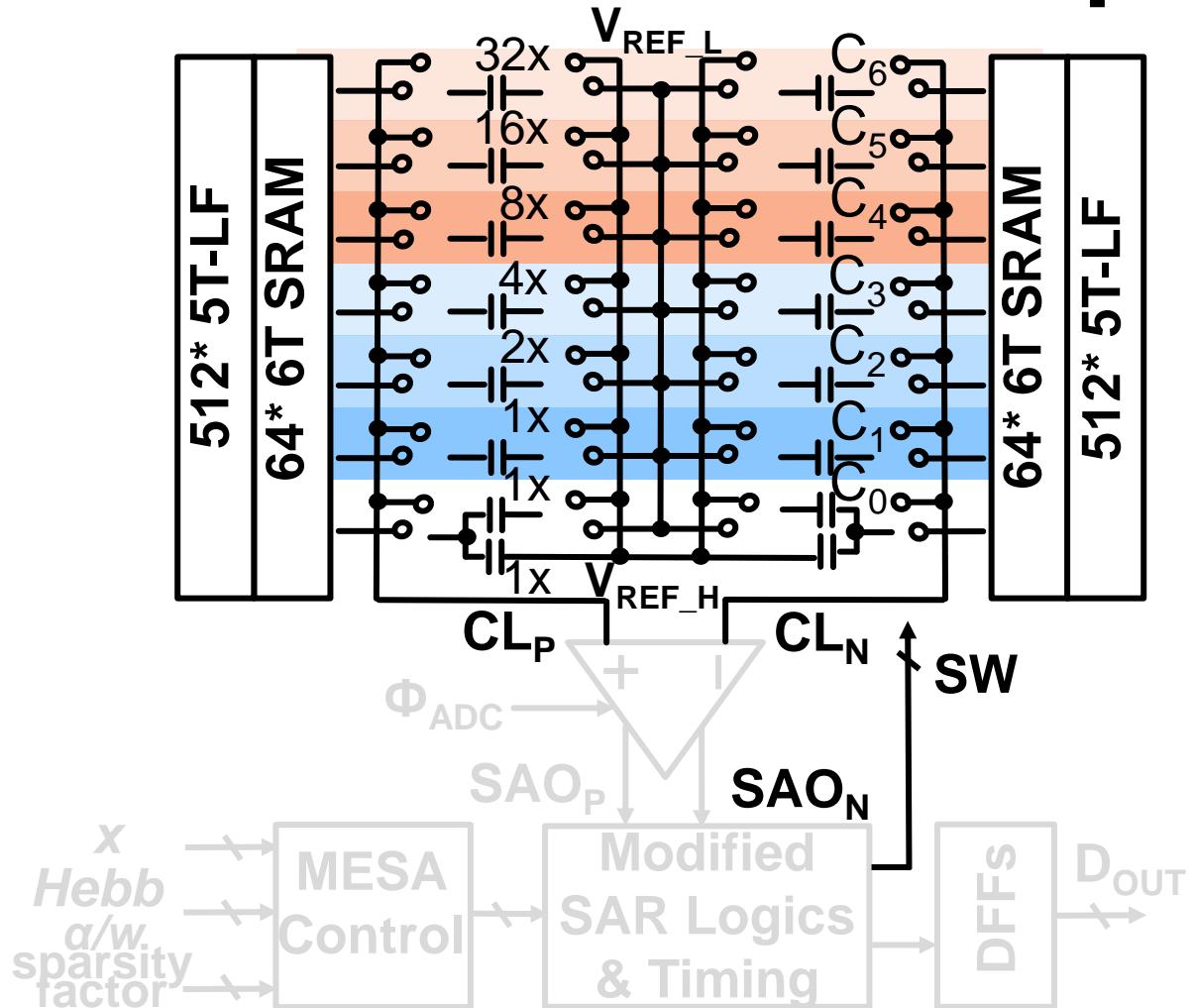
# DMA-ADC Work Flow – Phase3 (Quantizing)



- Bottom-plate of  $C_{SAM}$  grouped to ratioed capacitors and switched to quantize the MACV



# Common-Centroid Capacitors Grouping Strategy



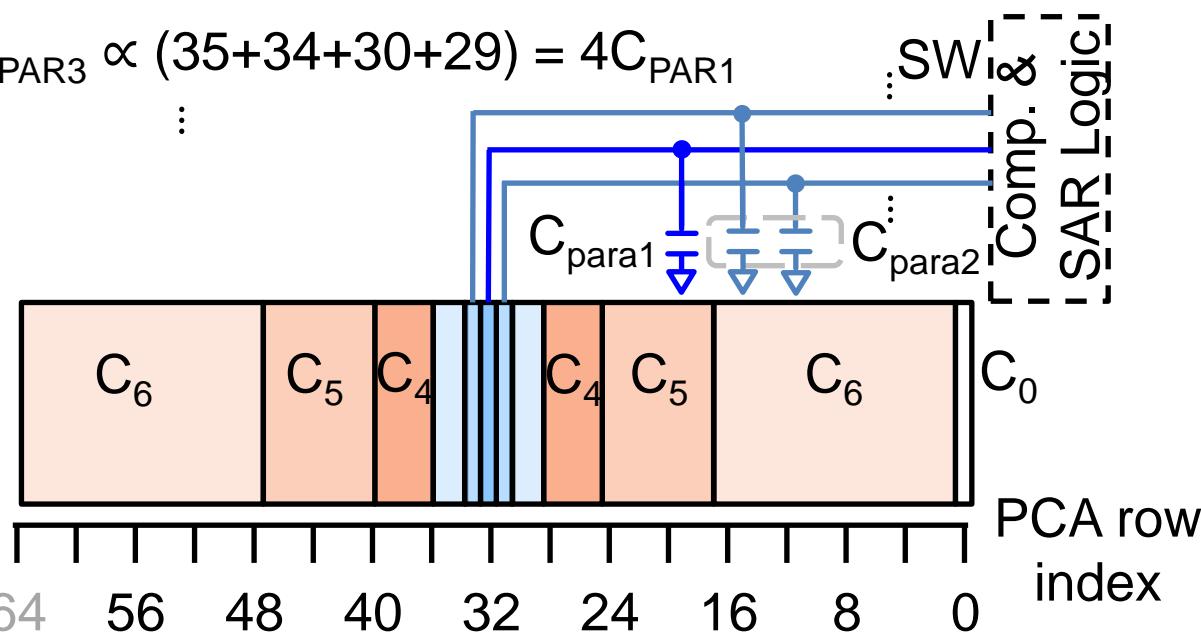
Ratioed wire parasitic capacitors  $C_{PAR}$  of  $C_6$  to  $C_1$ :

$$C_{PAR1} \propto 32$$

$$C_{PAR2} \propto (33+31) = 2C_{PAR1}$$

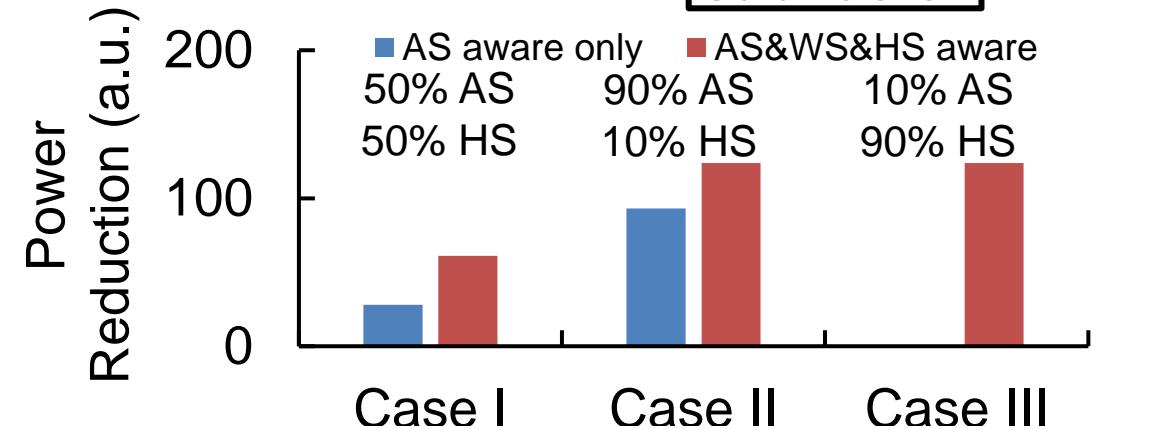
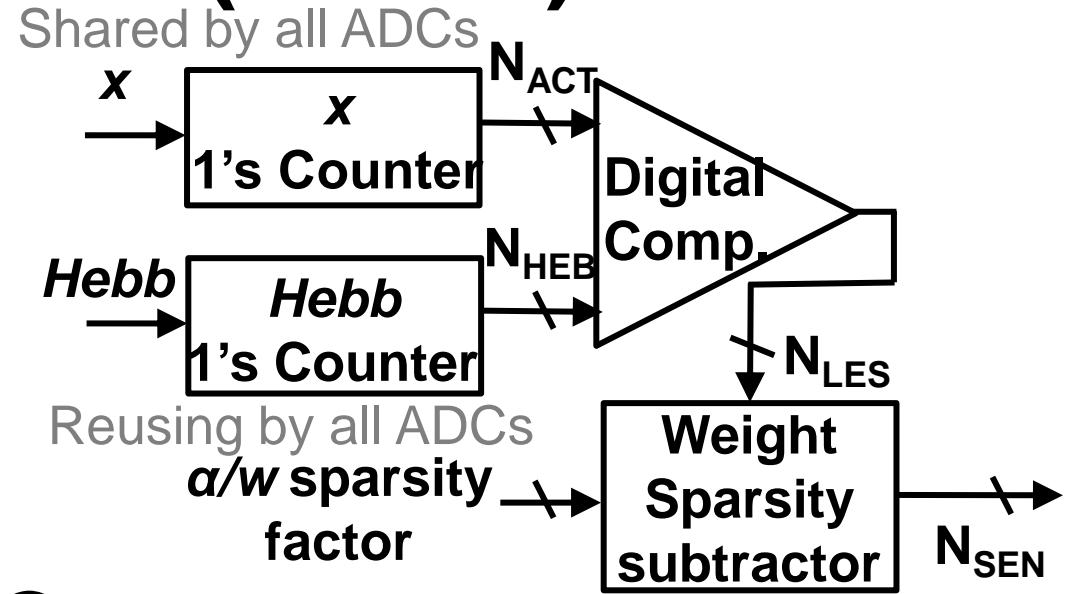
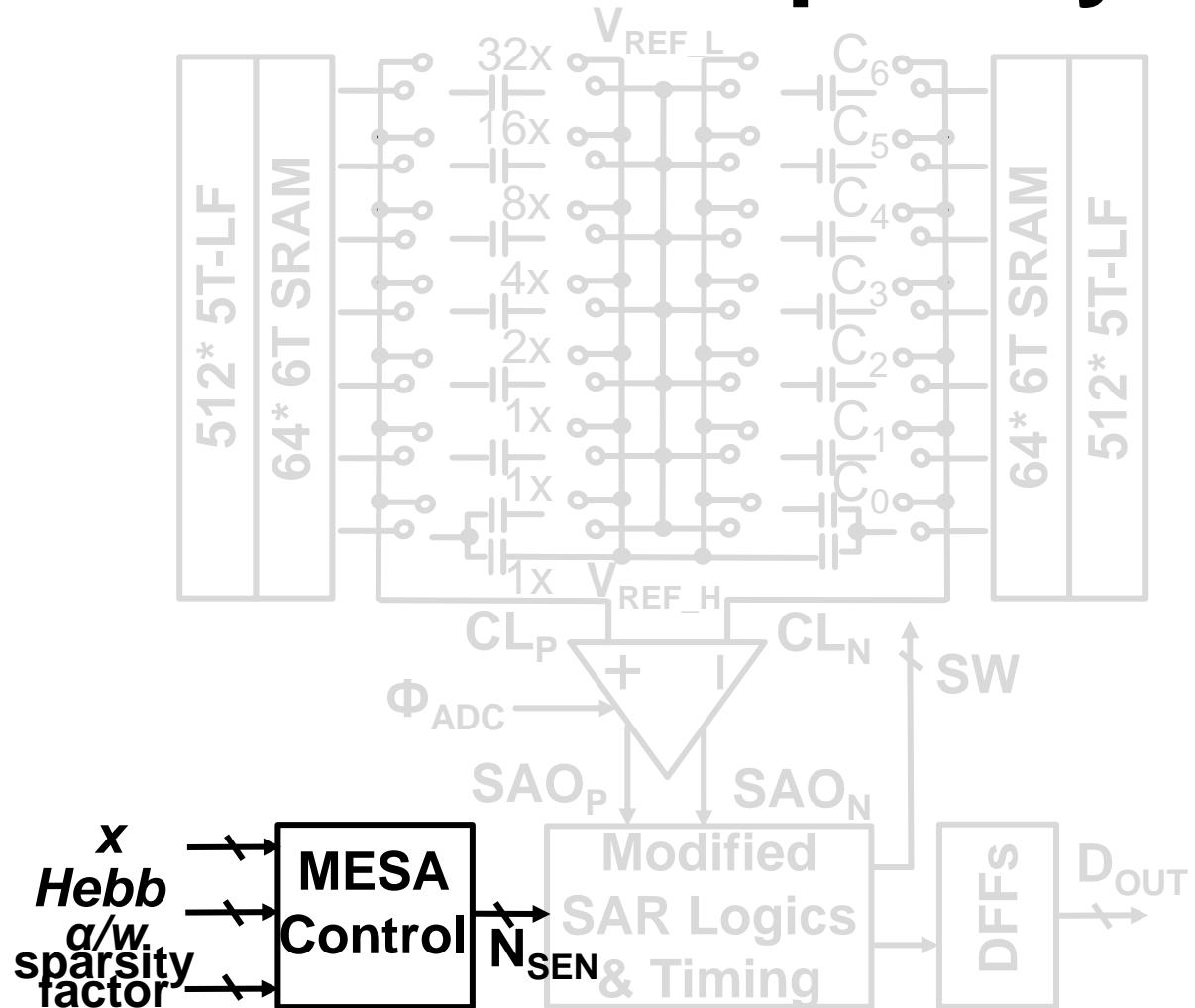
$$C_{PAR3} \propto (35+34+30+29) = 4C_{PAR1}$$

⋮



- Ensuring the parasitic capacitors of  $C_6 - C_0$  ( $C_{PAR6} - C_{PAR0}$ ) maintains the same ratios

# Multi-Element Sparsity-Aware (MESA) Control

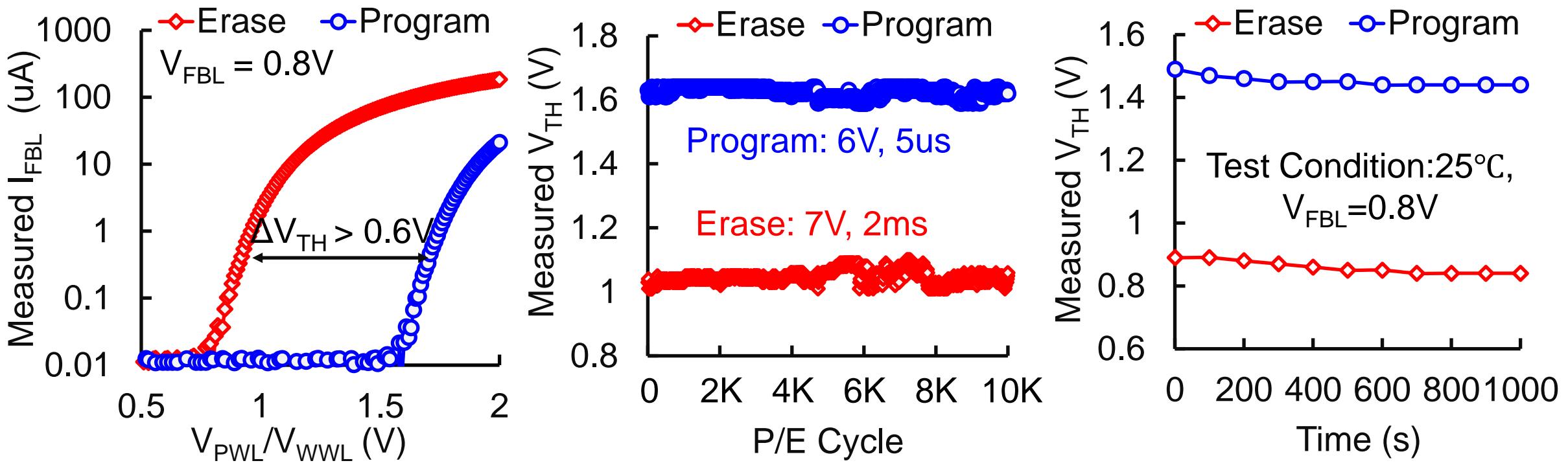


- Dynamically modulate sensing steps by leveraging the sparsity in  $x$ , Hebb,  $a$  or  $w$  to save power

# Outline

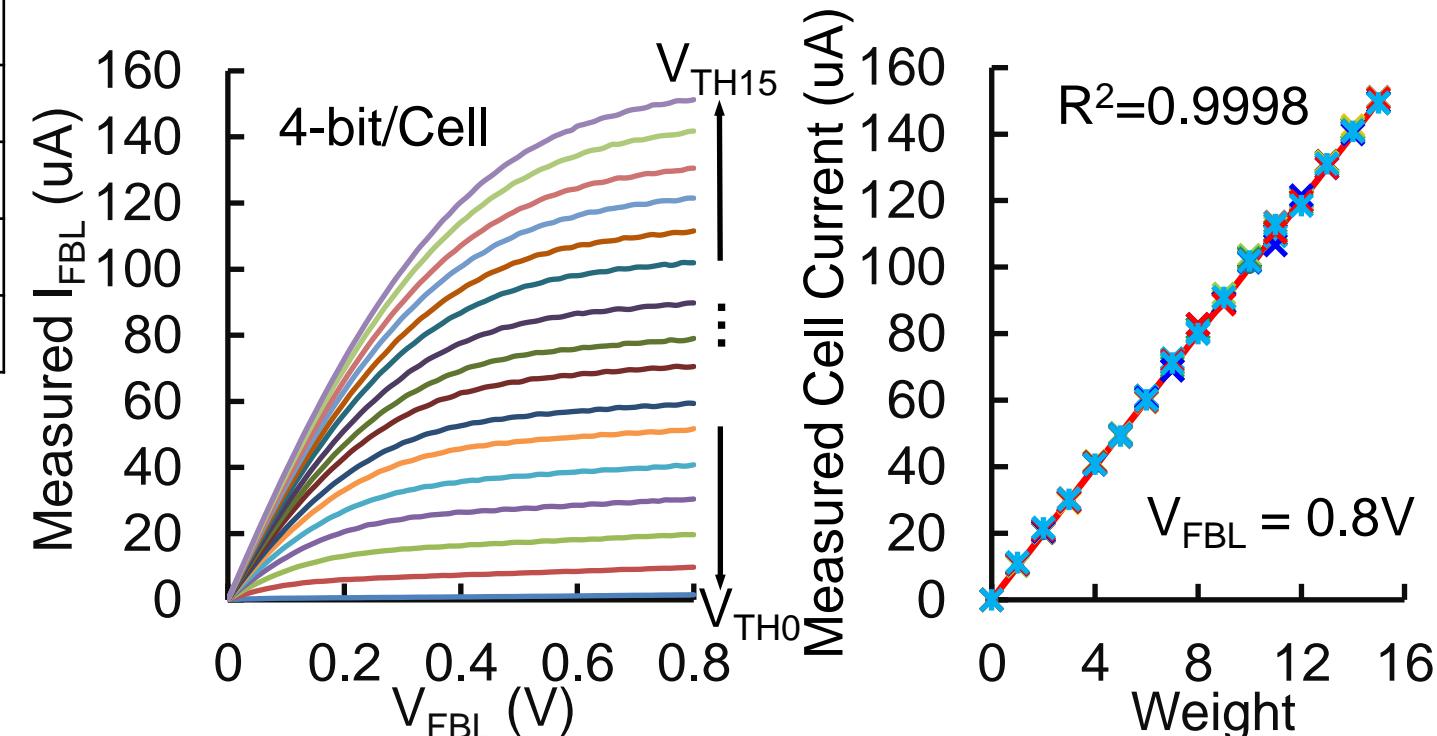
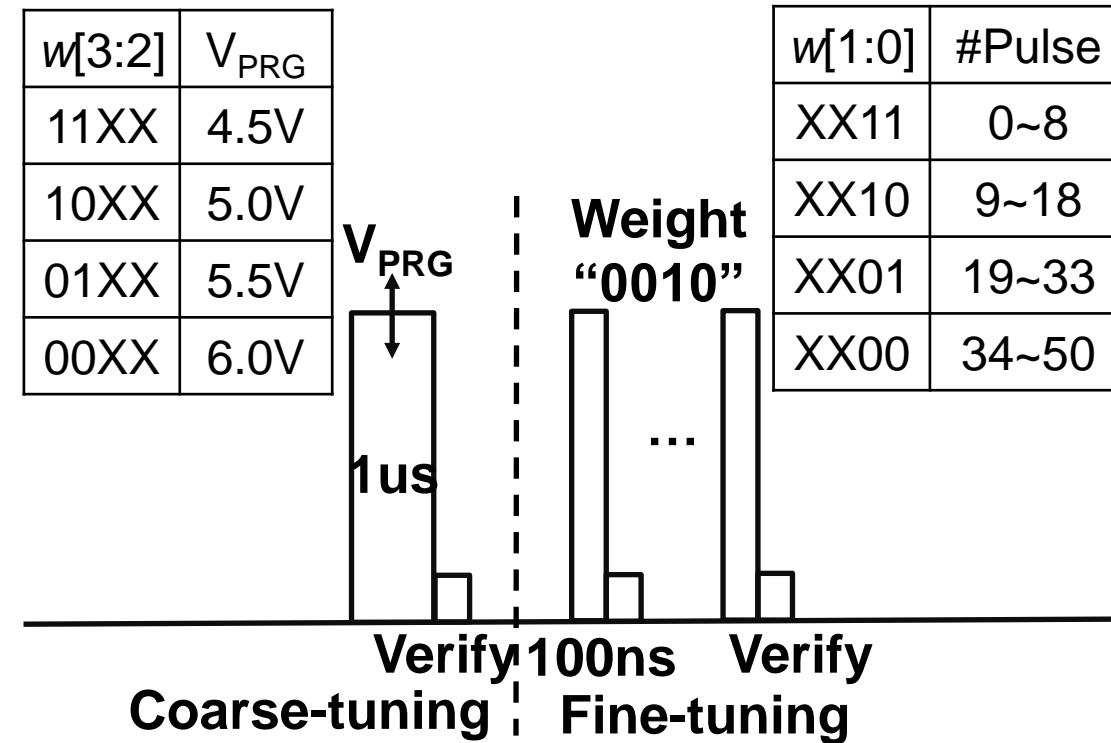
- Motivation and Challenges
- Proposed Plastic-CIM Macro
  - Overview of Plastic-CIM Macro
  - Plastic Cell Array
  - Differential Merged-into-Array ADC
- Performance and Measurement Results
- Conclusion

# Measured 14nm FinFET 5T-LF Cell Behavior



- 14nm FinFET 5T-LF cell shows a  $V_{TH}$  window ~0.6V over 10,000 P/E cycles, and the data retention is >1000s

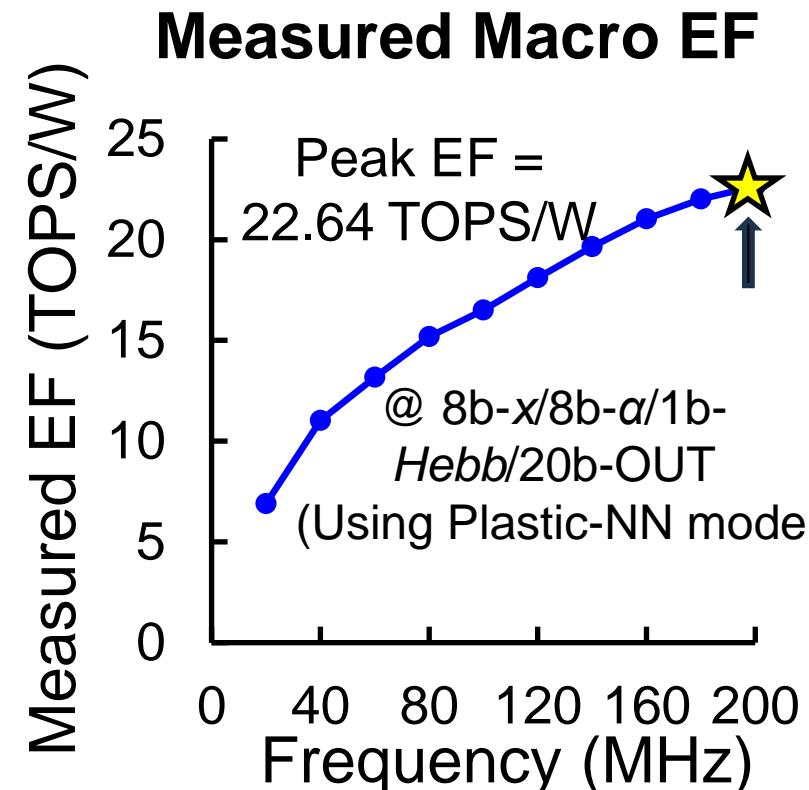
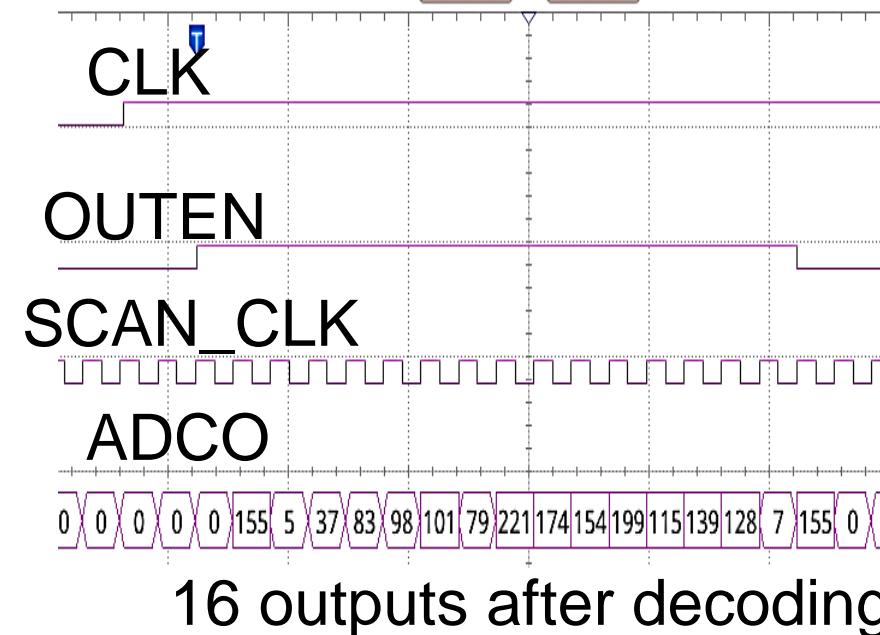
# Measured MLC Programming Characteristics



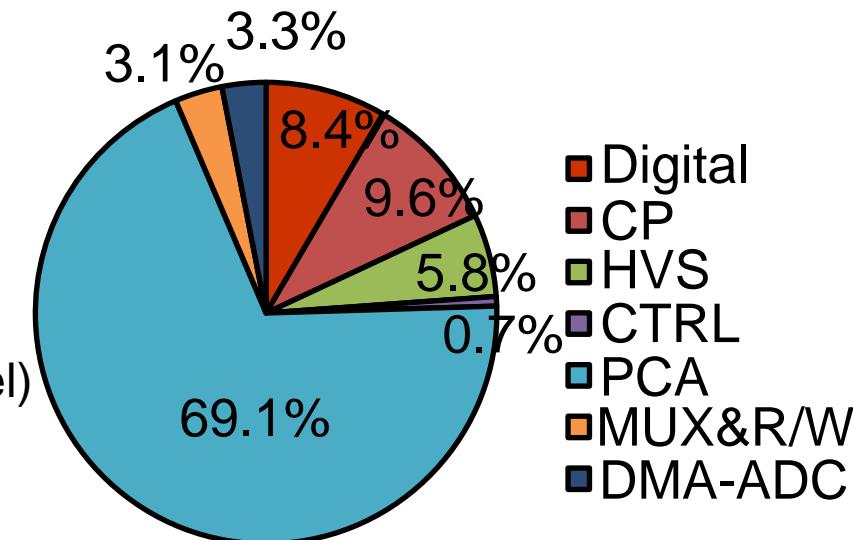
- MLC programming by two steps: coarse-tune by pulse amplitudes and fined tune by pulse numbers
- 4-bit/Cell is achieved with linearly distributed cell currents

# Measured Macro Level Results

## Measured ADC Functional Waveforms

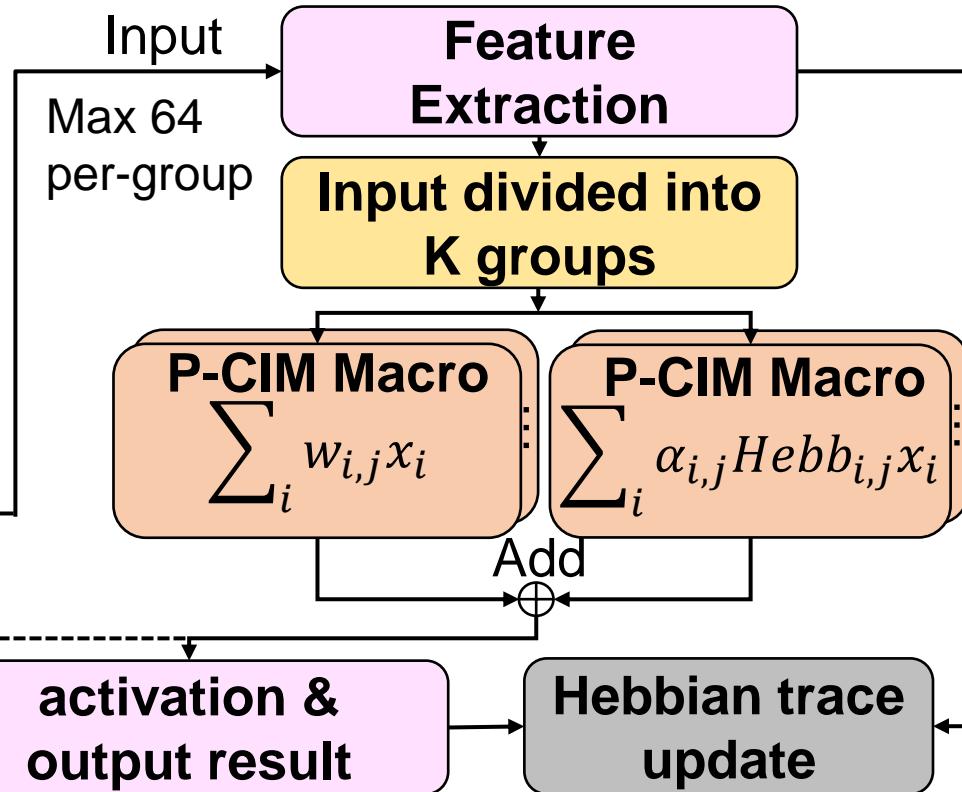
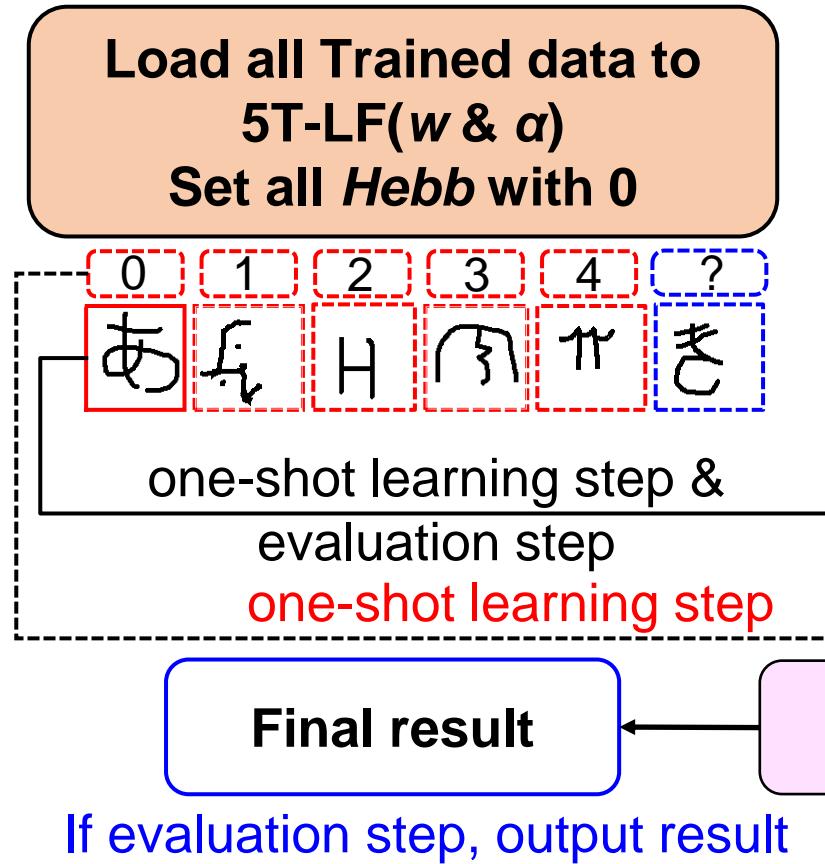


## Macro Area Breakdown



- Achieve peak EF = 22.64TOPS/W at 200MHz for MEM-MVM with 8b-x/8b-a/1b-Hebb and 20b-output
- ADC area costs are effectively reduced by reusing C<sub>SAM</sub>

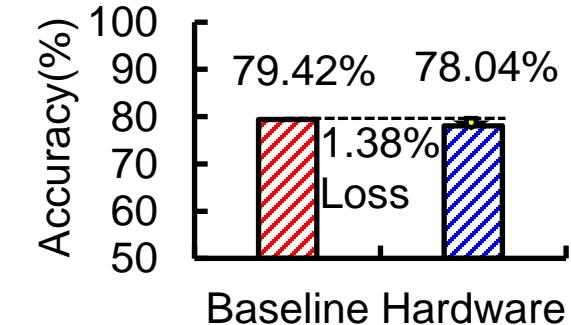
# P-CIM macro for One-Shot Learning



Network params (plastic layer)

element	# of params
$\alpha$ (int8)	320
$w$ (int8)	320
$Hebb$ (int8)	320

One-shot learning (Omniglot 5-way 1-shot)



- Processing one-shot learning tasks using the P-CIM macro achieves an accuracy of 78.04% on the Omniglot dataset

# Benchmark Table (I)

		JSSC21 [1]	ISSCC22[2]	ISSCC22[3]	CICC23 [4]	VLSI23[5]	This Work
Memory Characteristics	Technology	65 nm	40nm	40nm	65 nm	40nm	14nm
	Memory Type	3T Logic Flash	1T1R RRAM	NOR FLASH	5T Logic Flash	1T1R RRAM	5T-LF &SRAM
	Number of Memory Cells	20K	64K	152M	2K	64K	32K 5T-LF & 4K SRAM
	MLC Capability	Yes	No	Yes	Yes	No	Yes
Function Versatility	CIM Function	MVM (IN·W <sub>1</sub> )	MVM(IN·W <sub>1</sub> ), MEM-MVM(IN·W <sub>1</sub> ⊕W <sub>2</sub> )				
	Input Vector (IN) Precision (bit)	8	1	2-8	4	1	1/2/8
	Weight Matrix 1(W <sub>1</sub> ) Precision (bit)	8	1	8	3	1	5/8
	Weight Matrix 2(W <sub>2</sub> ) Precision (bit)	Not support	1-8				
	ADC Precision (bit)	Analog	1-4	2-8	6	6	1-8
	Output Precision (bit)	Analog	1-4	2-8	6	6	20(MVM Mode)/28(MEM-MVM Mode)
	Application	Inference	Inference	Inference	Inference	Inference	Inference + Learning
	Accuracy	98.5% @ MNIST	N.A.	29.3% mAP @ COCO	90.06% @ CIFAR10	N.A.	78.04%* <sup>6</sup> @ Omnistiglot

# Benchmark Table (II)

		JSSC21 [1]	ISSCC22[2]	ISSCC22[3]	CICC23 [4]	VLSI23[5]	This Work	
Computation Efficiency	Measured Throughput (GOPS)	N.A.	8.1 <sup>*3</sup> (1b/1b)	59300 (2b/8b)	186.2 (4b/3b)	184 <sup>*3</sup> (1b/1b)	153.6 <sup>*1</sup> (8b/8b/1b)	19.2 <sup>*1,2</sup> (8b/8b/8b)
	Measured EF (TOPS/W)	N.A.	26.56 (1b/1b)	18.5 (2b/8b)	333 (4b/3b)	350 (1b/1b)	22.64 <sup>*1</sup> (8b/8b/1b)	2.23 <sup>*1,2</sup> (8b/8b/8b)
	Measured AF (TOPS/mm <sup>2</sup> )	N.A.	0.3 (1b/1b)	N.A.	0.56 <sup>*4</sup> (4b/3b)	7.01 (1b/1b)	0.15 <sup>*1</sup> (8b/8b/1b)	0.018 <sup>*1,2</sup> (8b/8b/8b)
	Normalized EF <sup>*5</sup> (TOPS/W)	N.A.	26.56	296	3996	350	1448.96	
	Normalized AF <sup>*5</sup> (TOPS/mm <sup>2</sup> )	N.A.	0.3	N.A.	6.72	7.01	9.6	

\*1. Measured at 25° C, 0.8V and 200MHz.

\*2. Hebb update can be pipelined between multiple macros.

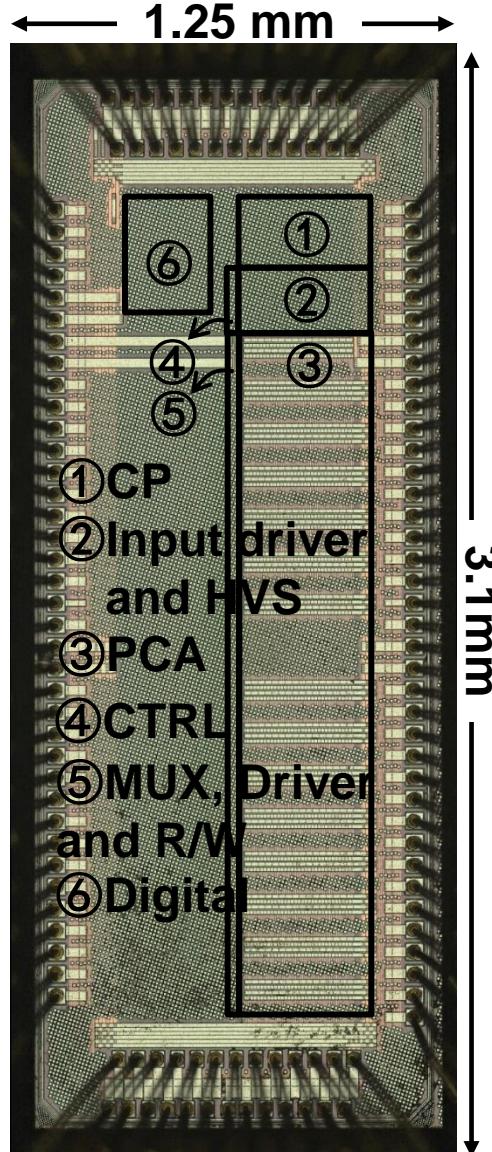
\*3. Estimated by (Area Efficiency \* Active area).

\*4. Estimated by (Throughput / Active area).

\*5. Normalized to 1b MVM.

\*6. Software five-way, one-shot learning accuracy baseline = 79.42%.

# Die Photo & Chip Summary



## Chip Summary

<b>Technology</b>	14nm FinFET	
<b>Memory Type</b>	32K 5T-LF & 4K SRAM	
<b>PCA Area</b>	0.75 mm <sup>2</sup>	
<b>5T-LF Cell Area (μm<sup>2</sup>/bit)</b>	2.3	
<b>Macro Read and Write Energy</b>	121.2pJ / 1.05μJ (CP&HVS included)	
<b>Compute Functionality</b>	MVM, MEM-MVM	
<b>Supply Voltage</b>	0.8V	
<b>Frequency</b>	20-200MHz	
<b>Temperature</b>	25° C	
Performance		
<b>Computing Latency (ns)</b>	8bIN-8bW <sub>1</sub> -1bW <sub>2</sub> -20bOUT	40
	8bIN-8bW <sub>1</sub> -8bW <sub>2</sub> -28bOUT	320
<b>Throughput (GOPS)</b>	8bIN-8bW <sub>1</sub> -1bW <sub>2</sub> -20bOUT	153.6
	8bIN-8bW <sub>1</sub> -8bW <sub>2</sub> -28bOUT	19.2
<b>Energy Efficiency (TOPS/W)</b>	8bIN-8bW <sub>1</sub> -1bW <sub>2</sub> -20bOUT	22.64
	8bIN-8bW <sub>1</sub> -8bW <sub>2</sub> -28bOUT	2.23
<b>Output Ratio</b>	8bIN-8bW <sub>1</sub> -1bW <sub>2</sub> -20bOUT	90.1%
	8bIN-8bW <sub>1</sub> -8bW <sub>2</sub> -28bOUT	93.3%
<b>Macro-level Learning Accuracy</b>	78.04% @ Omnistigot	

# Outline

- Motivation and Challenges
- Proposed Plastic-CIM Macro
  - Overview of Plastic-CIM Macro
  - Plastic Cell Array
  - Differential Merged-into-Array ADC
- Performance and Measurement Results
- Conclusion

# Conclusion

## ■ Proposed Plastic-CIM Macro

- Dedicated CIM macro structure that hybrid NVM and SRAM cell to mimic synaptic plasticity for on-chip learning
- Customized cell array to process long-/short-term information and accelerate matrix-vector and matrix-matrix operation
- ADC area-/energy-costs are lowered by reusing the sampling capacitors and leveraging multi-elements sparsity

## ■ A 14nm Flash-SRAM-ADC-Fused Plastic-CIM Macro is Verified

- The 5T-LF cells are firstly verified at the 14nm FinFET platform
- The macro achieves a EF of 22.64TOPS/W at 200MHz for MEM-MVM with 8b-x/8b-a/1b-Hebb and 20b output
- The normalized area efficiency is >1.4x than that of the previous 5T-LF CIM design



Please Scan to Rate  
This Paper

