SCOPE: A Stochastic Computing Engine for DRAM-based In-situ Accelerator

Shuangchen Li, Alvin Oliver Glova, Xing Hu, Peng Gu, Dimin Niu*, Krishna T. Malladi*, Hongzhong Zheng*, Bob Brennan*, and Yuan Xie

University of California, Santa Barbara

**Memory Solutions Lab, Samsung Semiconductor Inc.*

CRISP Center for Research on Intelligent Storage and Processing in Memory

UC Santa Barbara Scalable Energy-efficient Architecture Lab

Executive Summary

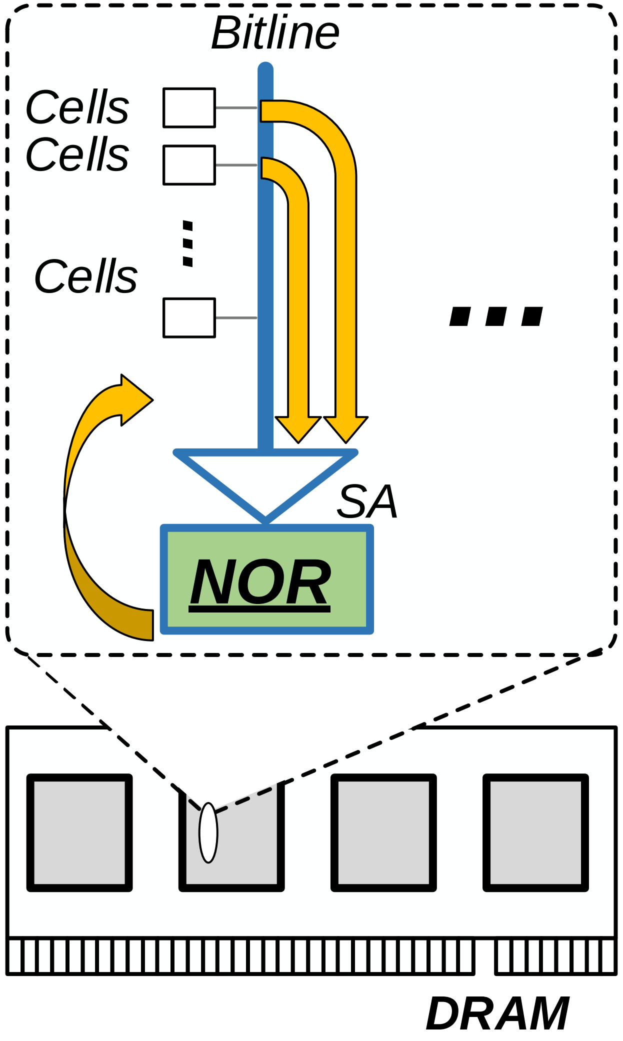
Introducing **Stochastic Computing** to **In-DRAM Computing Architecture:**

- For solving slow multiplication (MUL),
- Leverage large capacity and bandwidth.

Three arithmetic techniques (H^2D) to improve stochastic computing on such architecture.

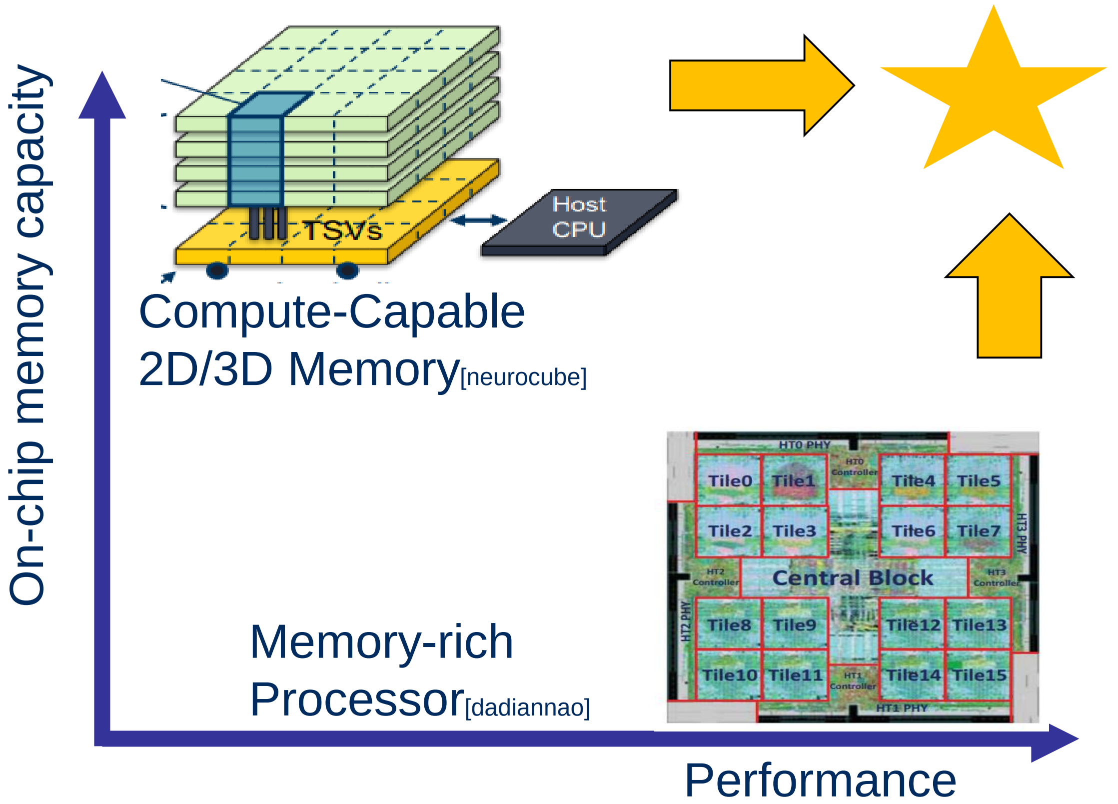
Experiments shows **2.3x performance** improvement v.s. w/o stochastic computing.

DRAM in-situ Accelerator Rocks

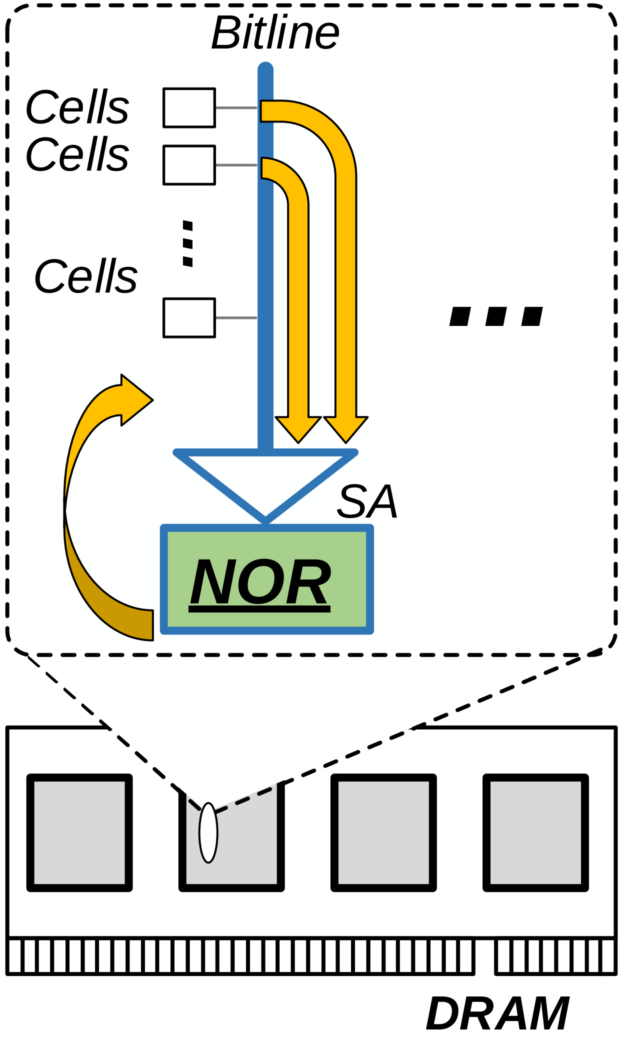


DRAM-based In-situ Accelerator rocks:

- with computing engines at BL-level,
- tightly bound memory and computing.



DRAM in-situ Accelerator ~~Rocks~~ Challenges



Run Boolean logic operation in SERIAL.

For example:

$$R = S \cdot X + \tilde{S} \cdot Y$$

NOR-only logic

$$\tilde{R} = \mathrm{NOR}\Big(\ \mathrm{NOR}(\tilde{S}, \tilde{X}),\ \mathrm{NOR}(S, \tilde{Y})\ \Big)$$

Step-1: $\tilde{X} = \mathrm{NOR}(0, X)$

Step-2: $\tilde{Y} = \mathrm{NOR}(0, Y)$

Step-3: $\tilde{S} = \mathrm{NOR}(0, S)$

Step-4: $\mathrm{tmp1} = \mathrm{NOR}(\tilde{S}, \tilde{X})$

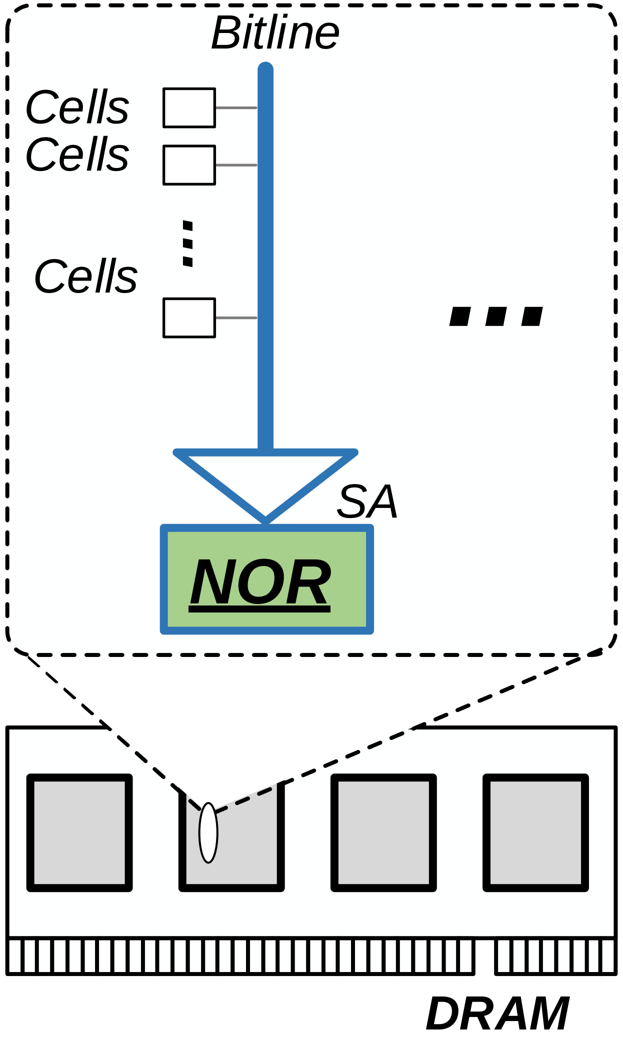
Step-5: $\mathrm{tmp2} = \mathrm{NOR}(S, \tilde{Y})$

Step-6: $\tilde{R} = \mathrm{NOR}(\mathrm{tmp1}, \mathrm{tmp2})$

Step-7: $R = \mathrm{NOR}(0, \tilde{R})$

X
Y
S
!X
!Y
!S
!(!X+!S)
!(!Y+S)
!R
R

DRAM in-situ Accelerator ~~Rocks~~ Challenges

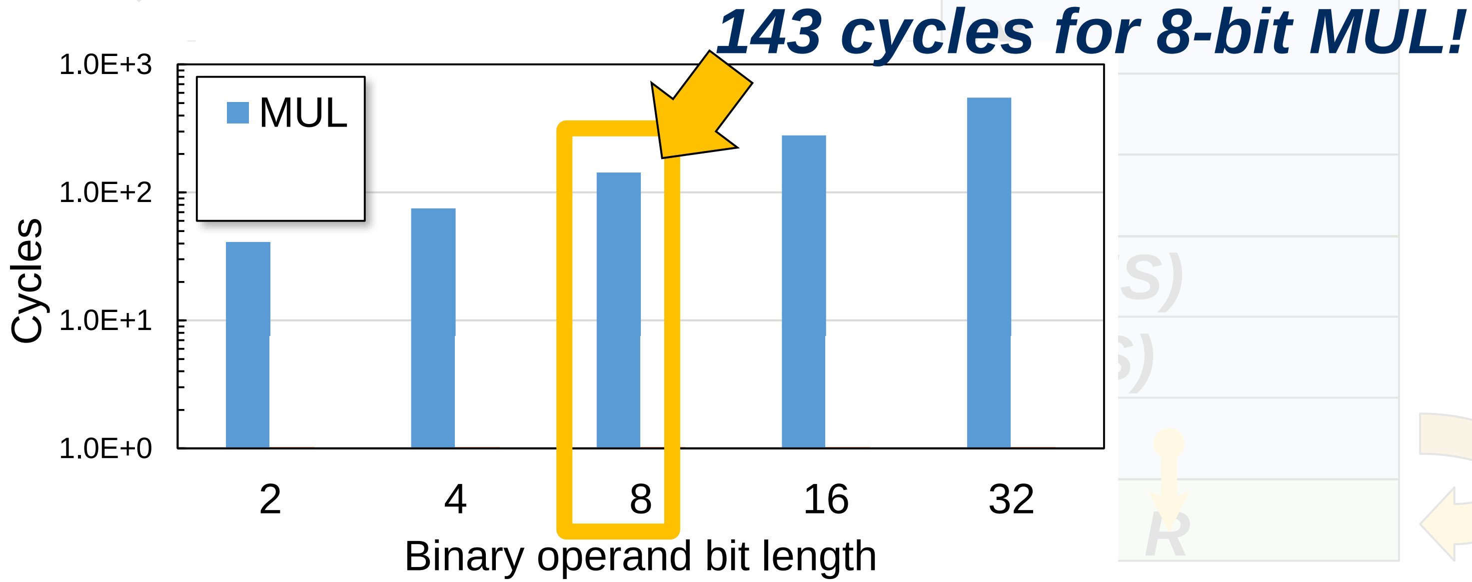


Run Boolean logic operation in SERIAL.

DRAM-Acc's Big Challenge:

- **Very Slow Multiplications!**

NOR-only logic



The Opportunity: Stochastic Computing

- Stochastic Computing (SC):
  - A different data representation and arithmetic (like INT vs. FP)
  - Bitstream representation: value = possibility of appearance of “1”

$$X\ (Binary) \rightarrow \{x_i\}\ (Stoch.),\ \ X = P(x_i = 1)$$

$$X = \frac{3}{6}\ (Binary) \rightarrow \{x_i\} = \{0, 1, 0, 1, 1, 0\}\ (Stoch.)$$

$$\frac{\#\text{“1”s} = 3}{\#\text{bits} = 6}$$

$$Y = \frac{2}{6}\ (Binary) \rightarrow \{y_i\} = \{0, 0, 1, 1, 0, 0\}\ (Stoch.)$$

$$\frac{\#\text{“1”s} = 2}{\#\text{bits} = 6}$$

The Opportunity: Stochastic Computing

- Stochastic Computing (SC):
  - A different data representation and arithmetic (like INT vs. FP)
  - Bitstream representation: value = possibility of appearance of "1"

$$X\ (Binary) \rightarrow \{x_i\}\ (Stoch.),\ \ X = P(x_i = 1)$$

$$X \cdot Y = P(x_i = 1) \cdot P(y_i = 1) = P(x_i = 1\ \&\ y_i = 1)$$

$$X = \frac{3}{6}\ (Binary) \rightarrow \{x_i\} = \{0, 1, 0, 1, 1, 0\}\ (Stoch.)$$

Binary → Long SC bitstream

$$Y = \frac{2}{6}\ (Binary) \rightarrow \{y_i\} = \{0, 0, 1, 1, 0, 0\}\ (Stoch.)$$

MUL → Simple bitwise AND!

& & & & & &

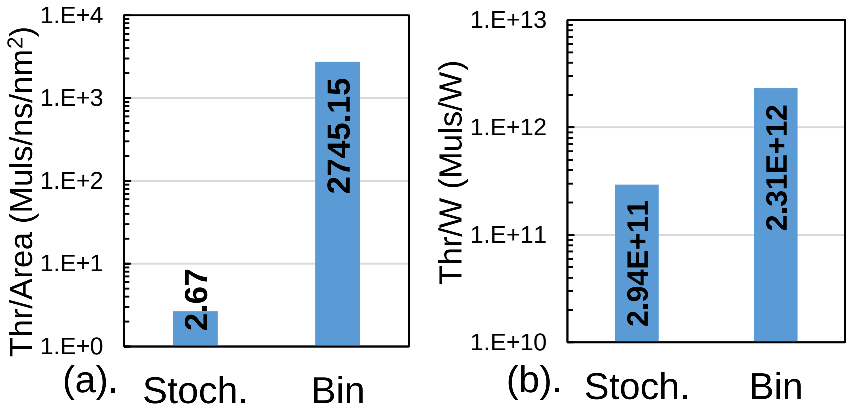
$$X \cdot Y = \frac{1}{6}\ (Binary) \rightarrow \{x_i \& y_i\} = \{0, 0, 0, 1, 0, 0\}\ (Stoch.)$$

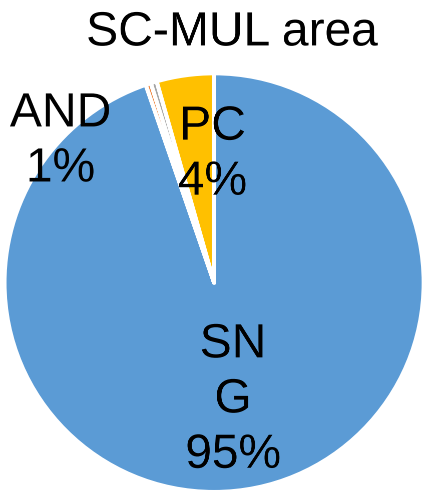
But not a Free Lunch…

The Good:
- MUL → AND, reducing MUL latency by 47x.

The Bad:
- Hurt throughput & energy efficiency
- Exp-long bitstream (8 → 256), **intensive BW and Capacity demands**
- Stoc-and-Binary **conversion overhead**

The Ugly:
- Numerical **precision loss**
- **Unreproducible error**, no debug





Key Idea: Combining DRAM-Acc with SC

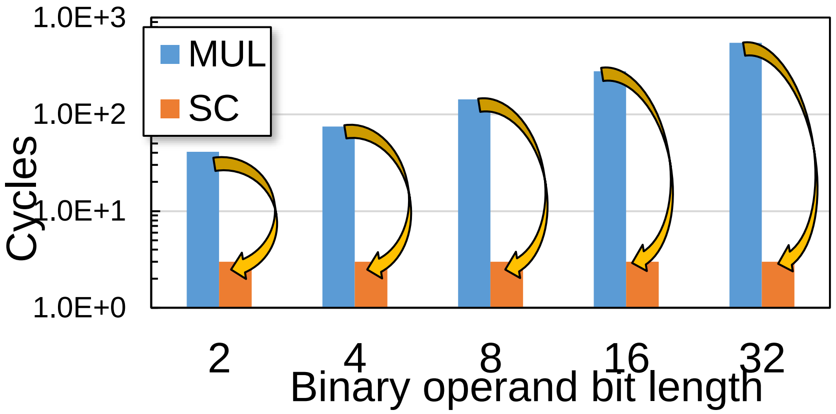
Stochastic Computing

DRAM-based In-situ Acc.

Suffer from Slow MUL

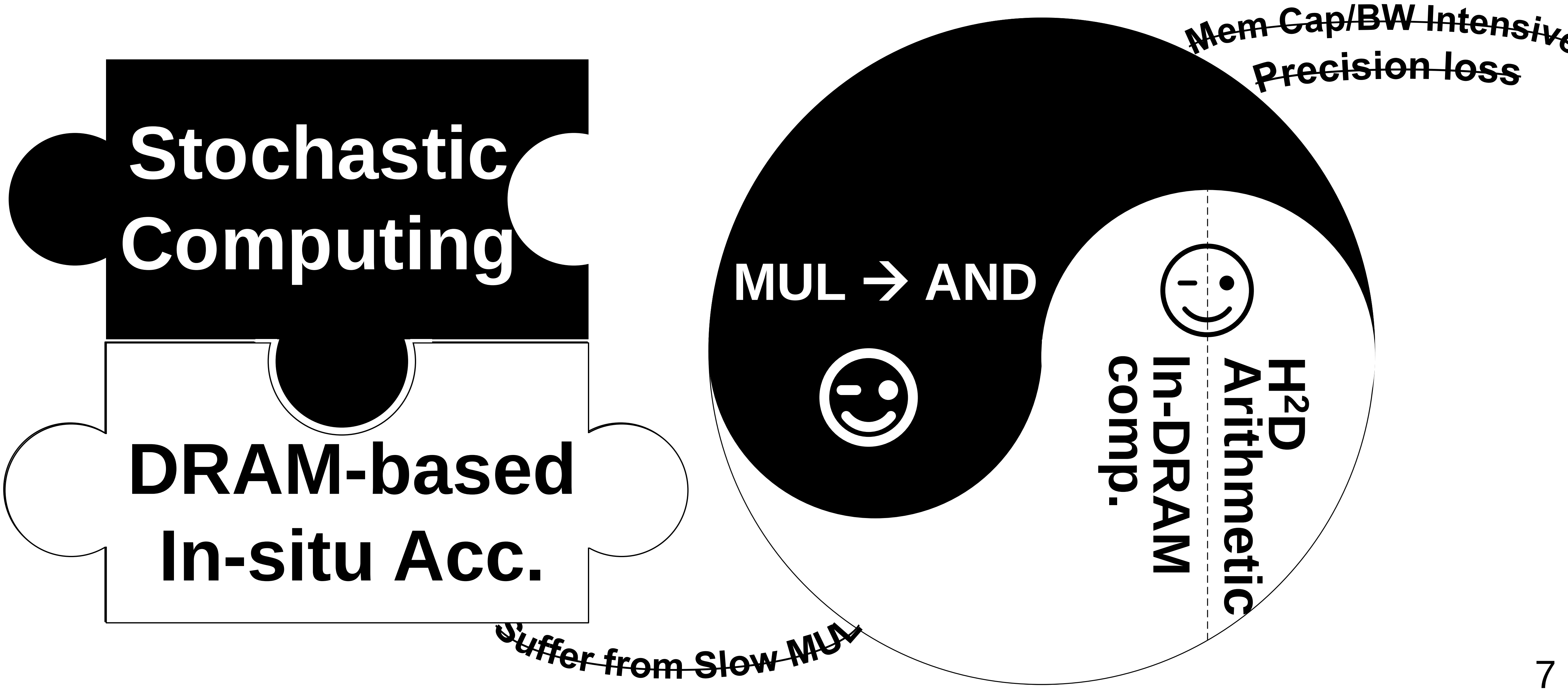
MUL → AND

😉



Key Idea: Combining DRAM-Acc with SC

Key Idea: Combining DRAM-Acc with SC



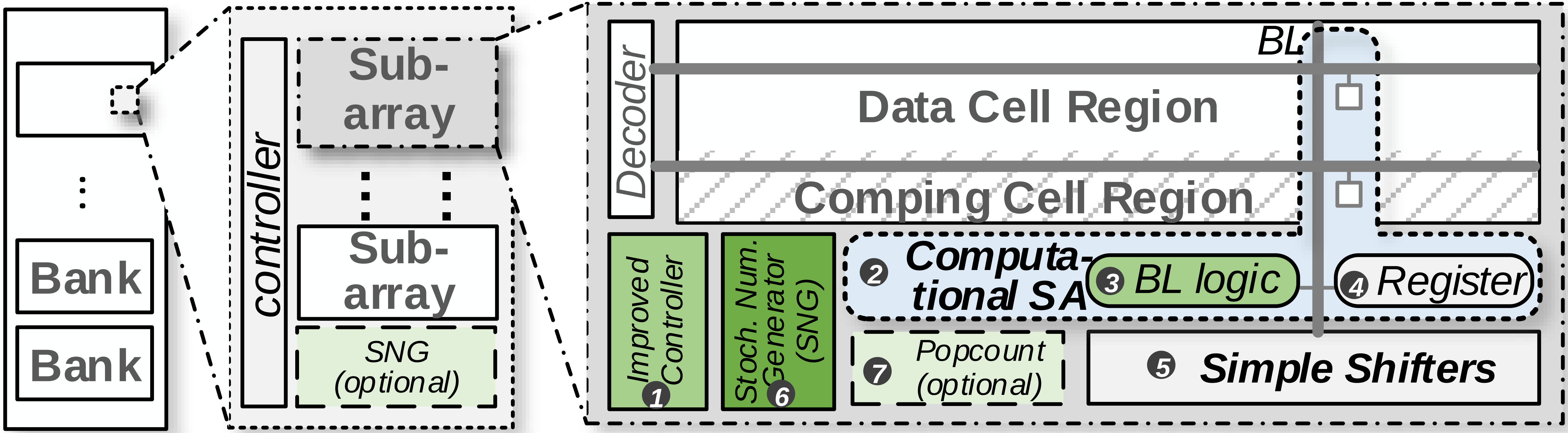
Related Work

- BL-level in memory computing architecture is hot.
  - AMBIT[MICRO’17], DIRSA[MICRO’17],Compute Caches[HPCA’17]…
- Stochastic computing is well study since 1960s.
  - Showing promising results on DNN workloads
  - J. Dickson[ICNN’93], K.Kim[DAC’16], DSCNN[ASPLOS’16], DPS[DAC’18], S.K. Khatamifard[CAL’18]…

This is the first work combines them together.

Putting together, they synergistically reinforce the strengths and address the weaknesses of each other.

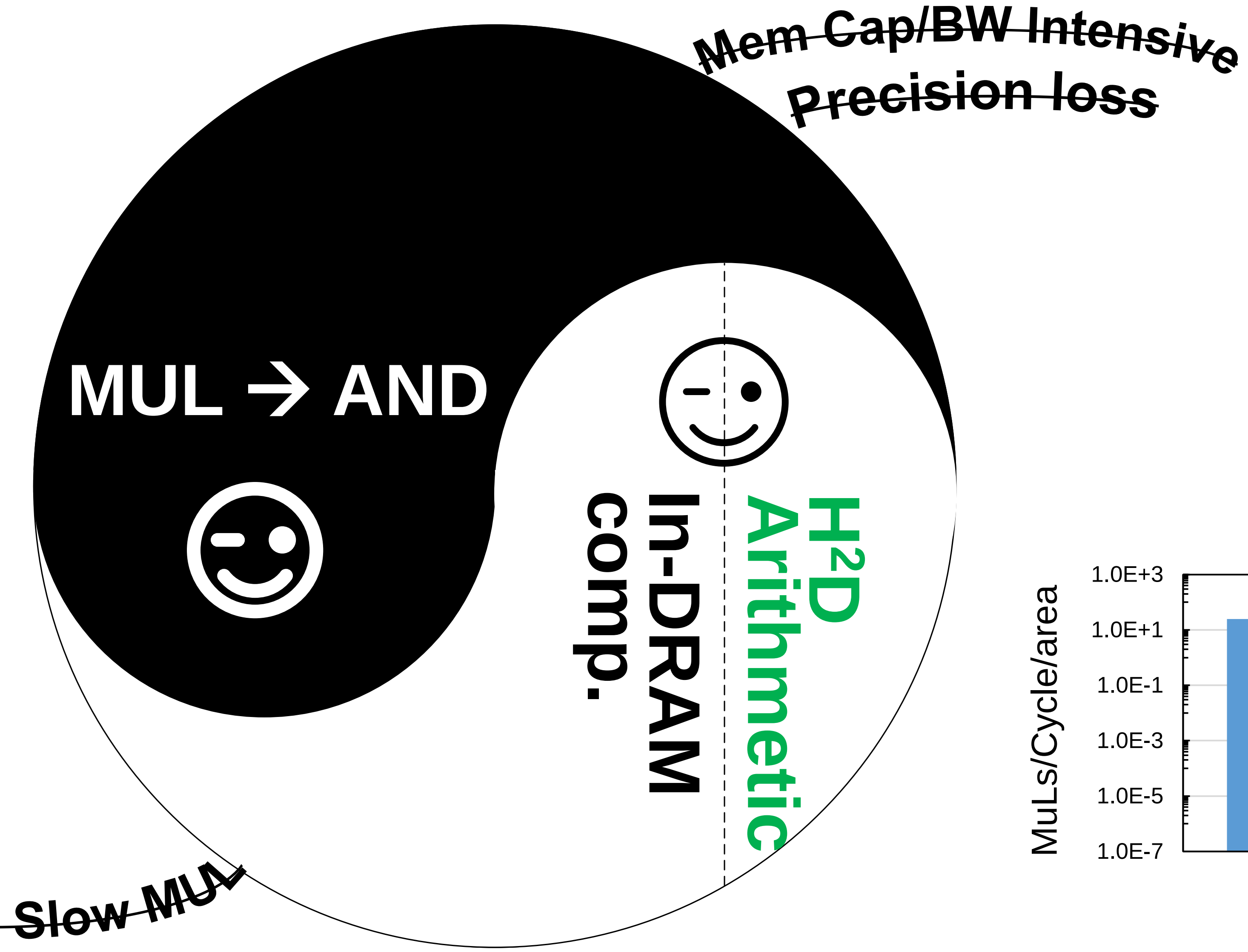
Overview of the Architecture



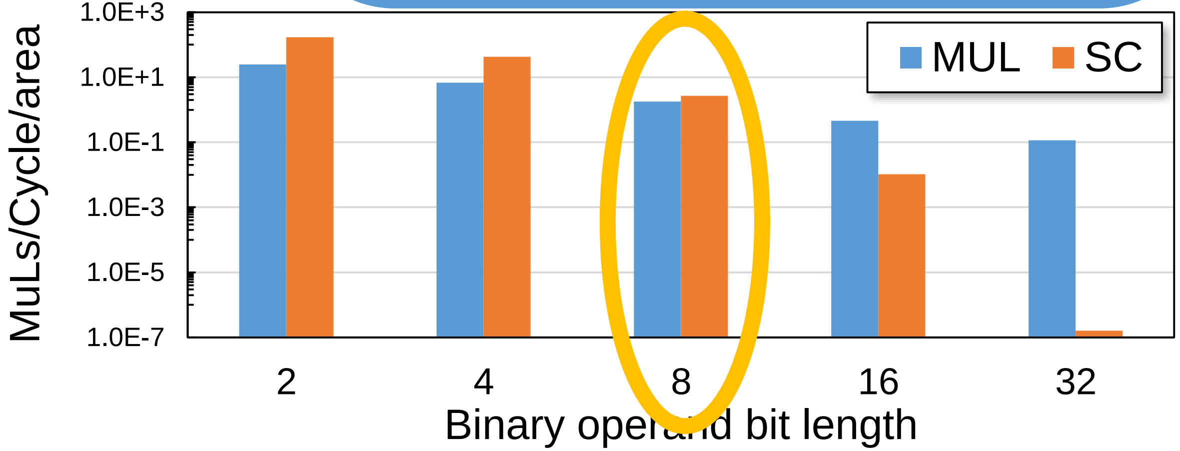
(b) Bank (c) Subarray

- Building upon BL-level in-DRAM computing architecture.
- Adding Stochastic number generator (SNG) and popcount (PC), and Improving controller and BL logic design

Can we do better: Introducing H^2D Arithmetic



- Exp-long bitstream.
- SNG overhead.
- Precision loss.
- Unreproducible error.



H^2D Arithmetic-1: Hierarchical Representation

- Observation:
  - trick for $O(2^n)$, change 2^n to $(2^{n/2} + 2^{n/2})$.
- Converting MSB-part and LSB-part separately!
- Example:

~~Exp-long bitstream.~~
SNG overhead.
Precision loss.
Unreproducible error.

Binary: [1,0,0,1] ⇒ [1,1,1,0,1,0,1,0,1,1,0,1,0,1,0,1]
(n width)
SC bitstream (2^n-1 width)

[MSBs,LSBs]
Binary: [1,0,0,1]
[1,0,1],[0,1,0]
SC bitstream: (($2^{n/2}$-1)*2 width)

H^{2}D Arithmetic-2: Hybrid Binary-Stochastic

- ~~Exp-long bitstream.~~
- SNG overhead.
- ~~Precision loss.~~
- Unreproducible error.

- Observation:
  - Unnecessarily redundant representation,
  - DRISA is fast at 2-bit MUL.
- Encoding part of SC as BIN, hybrid SC-BIN!

SC sub-stream: [1,1,1,0,1,0,1,0,1,0,0,0,0,0,1] 2^n-1 width

Count #-of-'1' in each sub-stream:

[3, 1, 2, 0, 1]

Hybrid-SC:
>intra sub-stream is BIN
>whole stream is still SC

[11, 01, 10, 00, 01] $(2^n-1)/3*2$ width

H²D Arithmetic-3: Deterministic SNG

- Observation:
  - Randomness is used to ensure low correlation,
  - We only do one OP in SC domain anyway,
  - "Real" random bitstream is unnecessary.
- LUT-based Stochastic Number Generator:

~~Exp-long bitstream.~~
~~SNG overhead.~~
~~Precision loss.~~
~~Unreproducible error.~~

$$\text{Operand X (5/16): } [\underbrace{0,0,0,0,0,0}_{\text{offset}},\underbrace{1,1,1,1,1}_{\text{“1”s}},\underbrace{0,0,0,0}_{\text{“0”s}}]$$

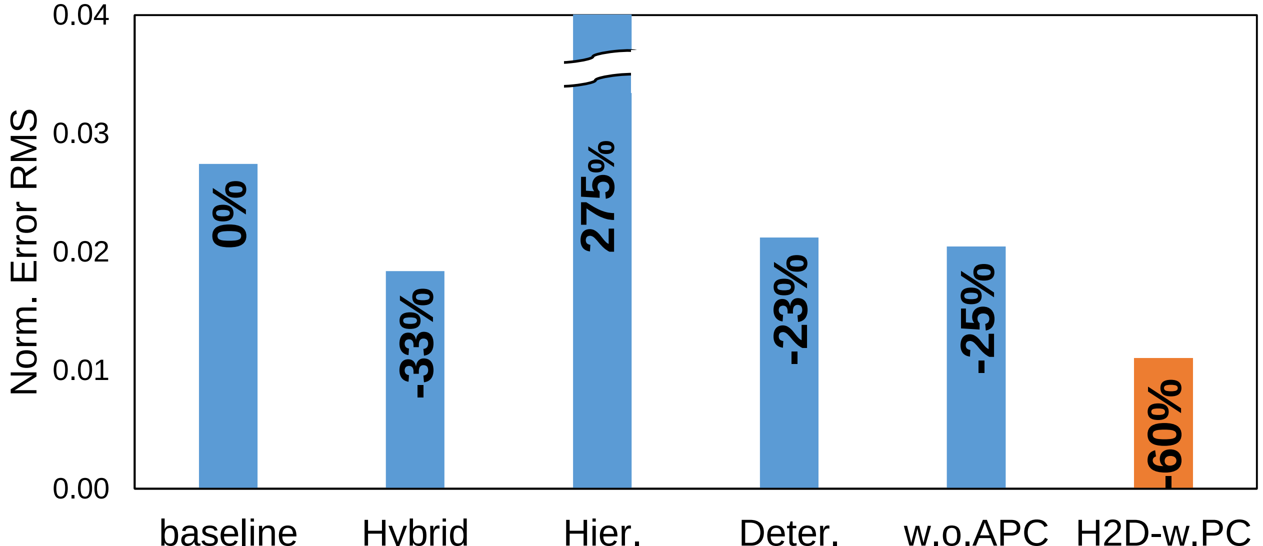
$$\text{Operand Y (5/16): } [\underbrace{1,0,0}_{\text{periodically}},1,0,0,1,0,0,1,0,0,1,0,0]$$

H²D Arithmetic: Putting Them Together

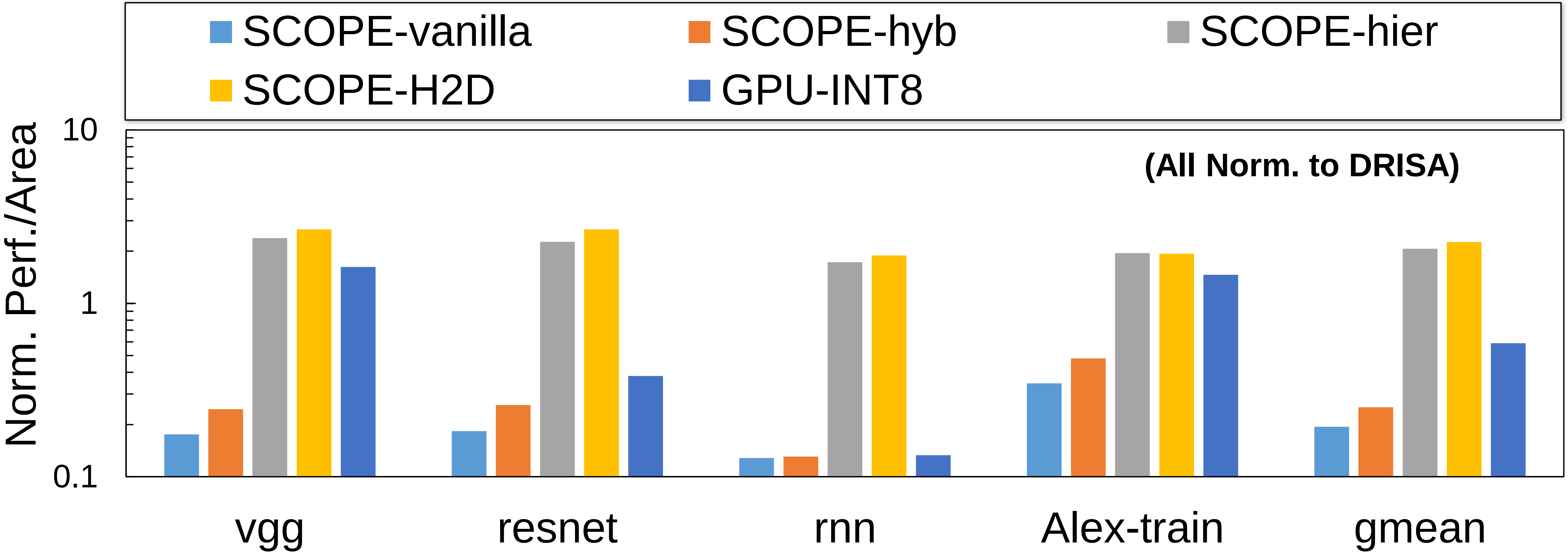
	DRISA[a]	SCOPE			
		vanilla	hier	hybrid	*H²D*
MUL latency[b]	143	3	17	4	21
Peak TOPs[c]	1.65	1.36	5.98	1.55	7.08
Area (mm²)	258.2	259.42	258.2	273.38	
Peak GOPs/Area	6.39	5.24	23.16	5.67	**25.90**

~~Exp-long bitstream.~~
~~SNG overhead.~~
~~Precision loss.~~
~~Unreproducible error.~~

- H²D further increases the throughput by 6x.
- H²D improves precision by 60%.



Experiments: A Case Study on DNN



- SCOPE w/ H^2D: 2.3x than DRISA, 11.6x than w/o H^2D

More In the Paper

- Architecture design detail.
- H2D design detail.
- DNN case study detail.
- More experiments.

- Come to our poster!

Summary

Stochastic Computing ➕ In-DRAM Computing Architecture:

- Synergistically reinforce the strengths and address the weaknesses of each other.

Three arithmetic techniques (H^2D) to further improve performance, and solve precision problems.

Thanks! Questions

SCOPE: A Stochastic Computing Engine for DRAM-based In-situ Accelerator

Shuangchen Li, Alvin Oliver Glova, Xing Hu, Peng Gu,
Dimin Niu*, Krishna T. Malladi*, Hongzhong Zheng*,
Bob Brennan*, and Yuan Xie

University of California, Santa Barbara
**Memory Solutions Lab, Samsung Semiconductor Inc.*

CRISP
Center for Research on Intelligent Storage and Processing in Memory

UC Santa Barbara
Scalable Energy-efficient Architecture Lab