IEEE Xplore® Digital Library

ISCAS 2020
2020 IEEE International Symposium on Circuits and Systems
Virtual, October 10-21, 2020

CAS

A New MRAM-based Process In-Memory Accelerator for Efficient Neural Network Training with Floating Point Precision

*Hongjie Wang**, *Yang Zhao**, *Chaojian Li*, *Yue Wang*, and ***Yingyan Lin***

Department of Electrical and Computer Engineering, Rice University

2020 IEEE International Symposium on Circuits and Systems
Virtual, October 10-21, 2020

RICE

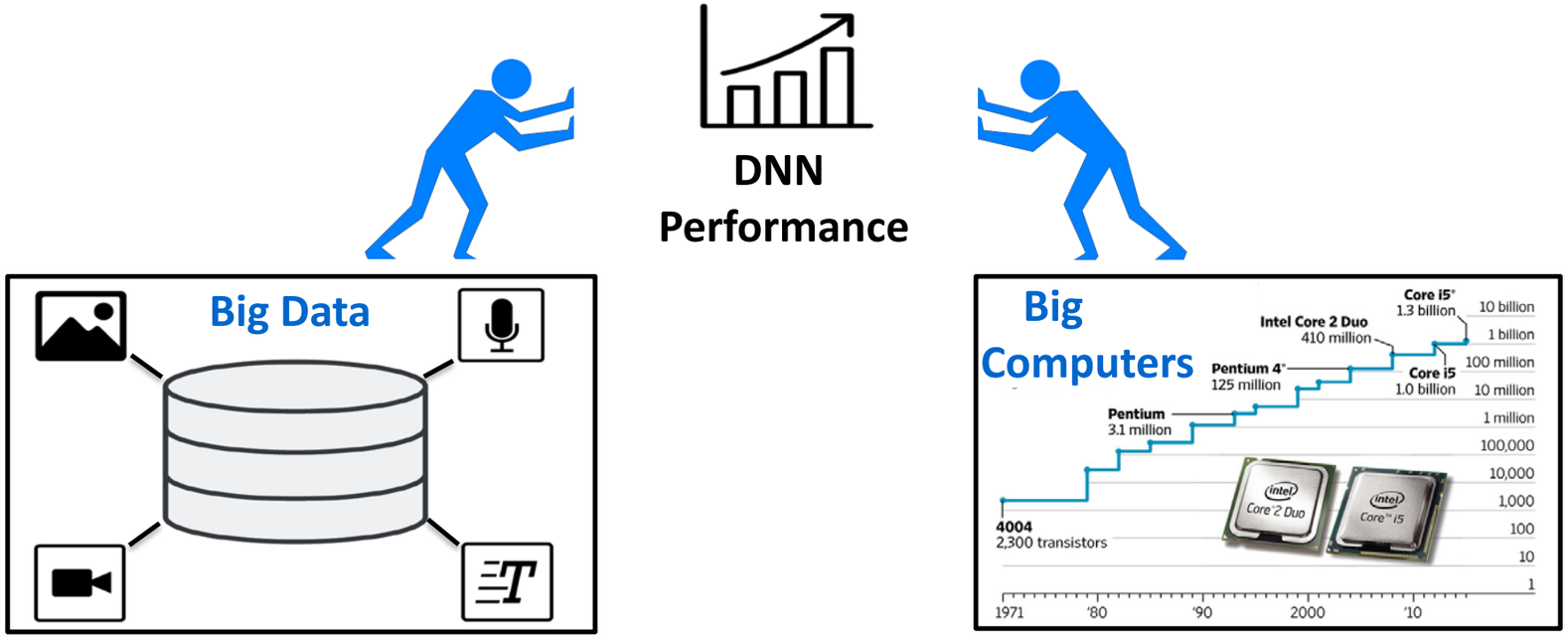
*the first two authors contribute equally

Outline

- Background and Motivation
- *The Proposed Accelerator:* Overview
- *The Proposed Accelerator:* Computing Methodology
- Evaluation of *the Proposed Accelerator*
- Conclusion

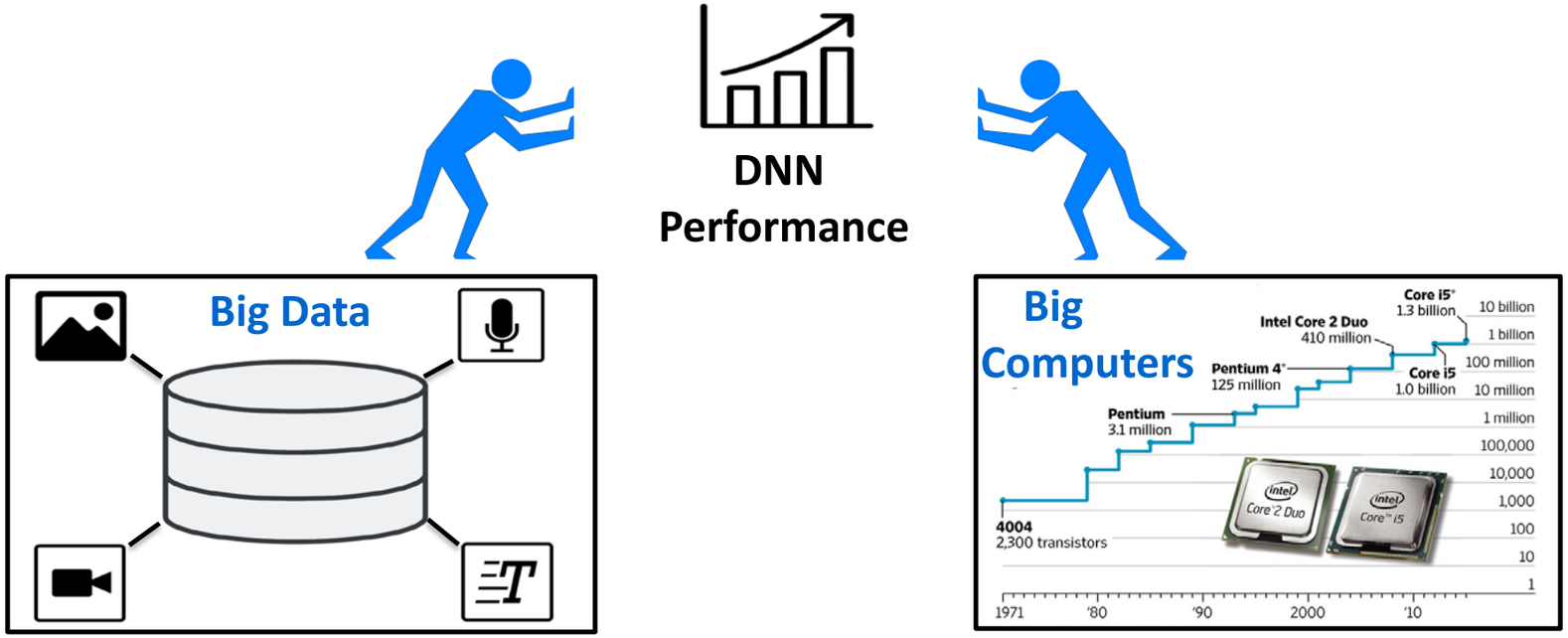
Growing Demand for Efficient DNN Training

- **Powerful DNNs require training with big data + big computers**



Growing Demand for Efficient DNN Training

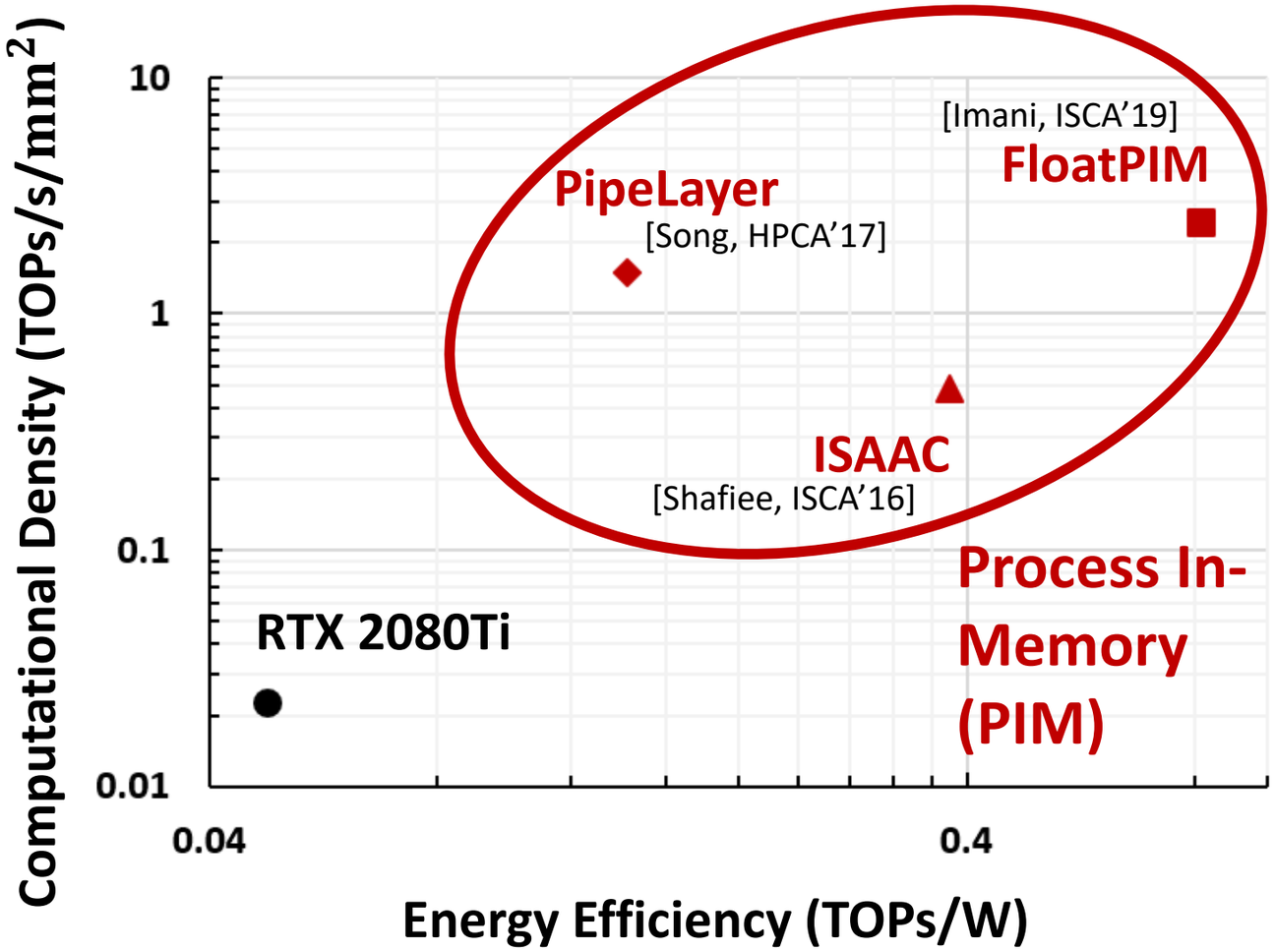
- **Powerful DNNs require training with big data + big computers**



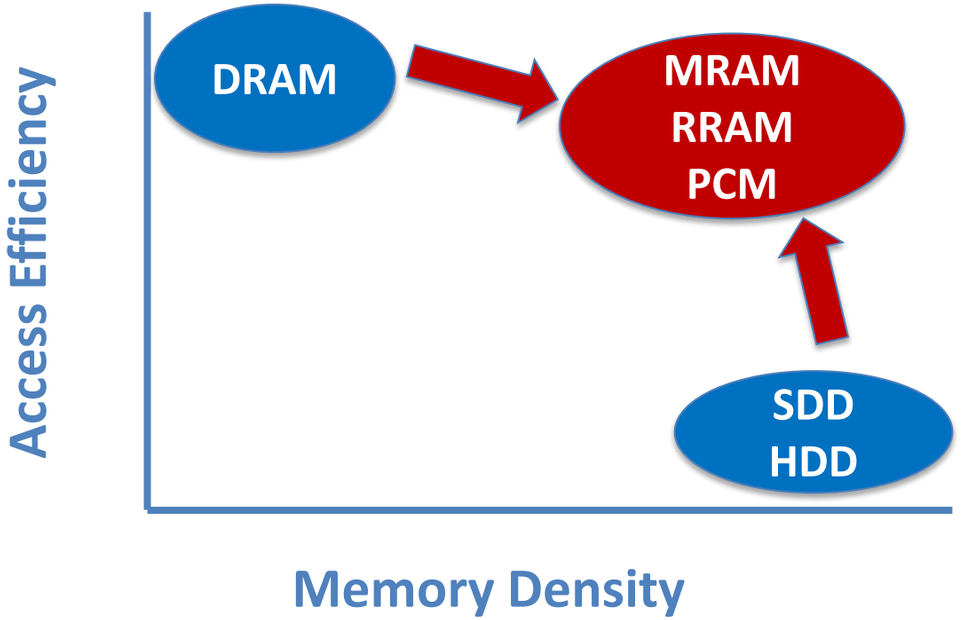
- **DNN training challenges:**
  - **Prohibitive computational and time cost**
    (e.g., 10^{18} FLOPs for training ResNet50 ~ 14 days using a NVIDIA M40 GPU)
  - **Increasing environmental concern**
    (e.g., carbon emission of training a DNN ≈ one car's lifetime emission)

Process In-Memory for DNN Acceleration

- Powerful DNNs require training with big data + big computers

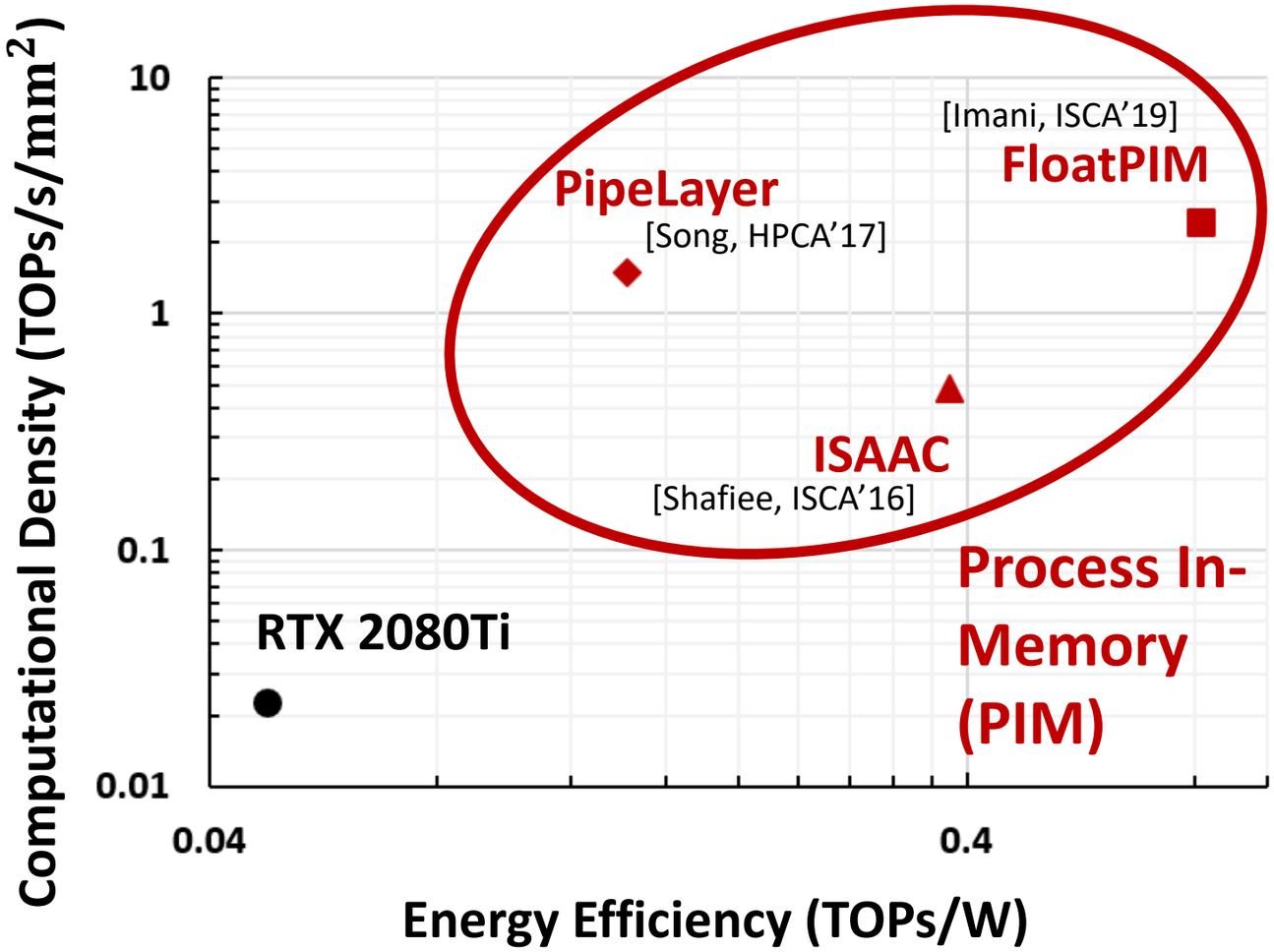


- High density non-volatile memory for huge number of parameters

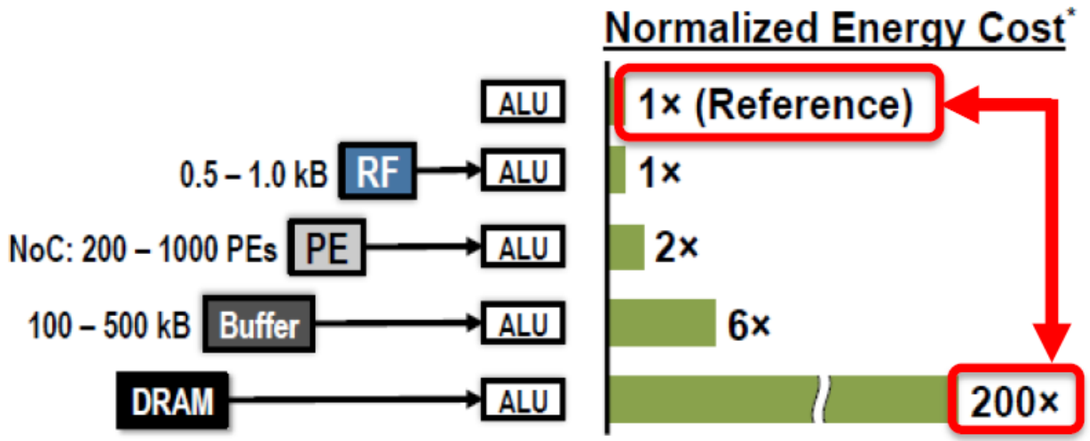


Process In-Memory for DNN Acceleration

- **Powerful DNNs require training with big data + big computers**



- High density non-volatile memory for huge number of parameters
- **Decrease costly data movements in Von Neumann Architecture**



* measured from a commercial 65nm process

[Chen, ISCA'16]

Existing PIM Accelerators for DNN Training

- **Existing PIM Accelerators for Deep Neural Network**

☹ Support **only inference** with low precision [Shafiee, ISCA'16] [Chi, ISCA'16][Patil, ISCAS'19]

☹ **Costly writing** for widely used ReRAM based accelerator [Imani, ISCA'19] [Song, HPCA'17]

☹ **High computational complexity** caused by limited available Boolean functions [Imani, ISCA'19]

- **Our Proposed PIM Accelerator**

☺ Supports **both inference and training** with floating point precision

☺ **Low writing current** thanks to SOT-MRAM based implementation

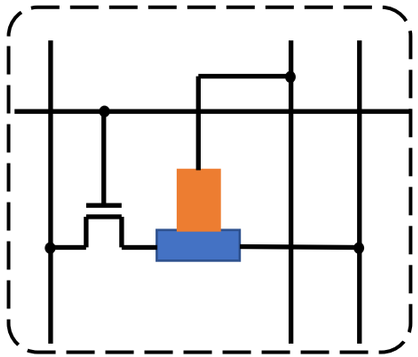
☺ **More efficient computing** caused by plentiful available Boolean functions

Outline

- **Background and Motivation**
- ***The Proposed Accelerator:* Overview**
- ***The Proposed Accelerator:* Computing Methodology**
- **Evaluation of *the Proposed Accelerator***
- **Conclusion**

The Proposed Accelerator: Overview

Read and Write: Cell Design

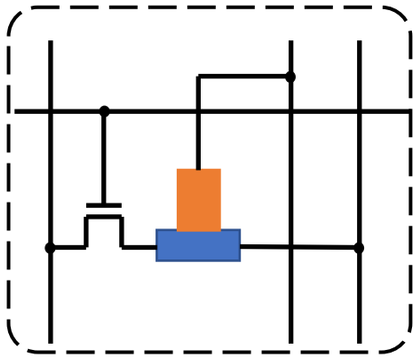


1T-1R structure

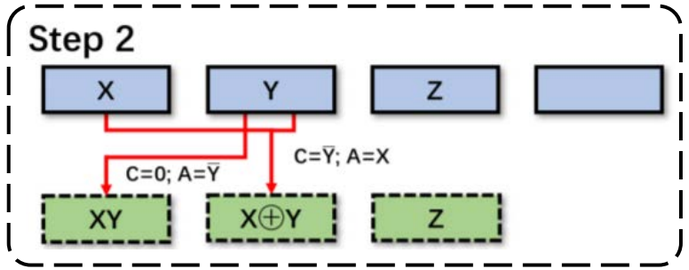
The Proposed Accelerator: Overview

1-Bit Full Addition: Cell Level Data Transfer

Read and Write: Cell Design



1T-1R structure



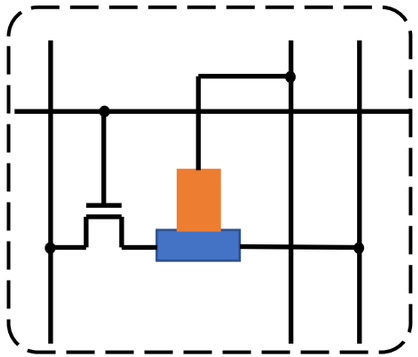
4 steps to perform 1-bit full addition

The Proposed Accelerator: Overview

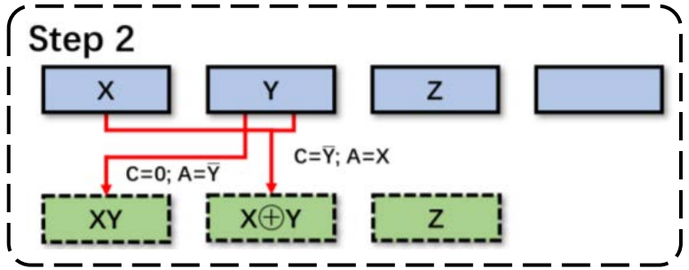
Floating Point Computation: Intra-Array Level Data Transfer

1-Bit Full Addition: Cell Level Data Transfer

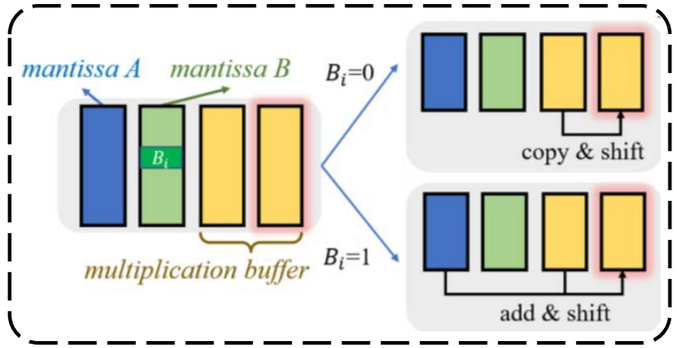
Read and Write: Cell Design



1T-1R structure



4 steps to perform 1-bit full addition



Addition and multiplication design

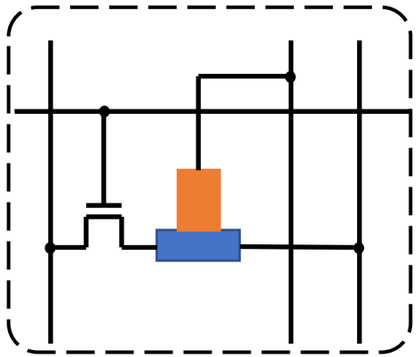
The Proposed Accelerator: Overview

Training and Inference: *the Proposed Accelerator*

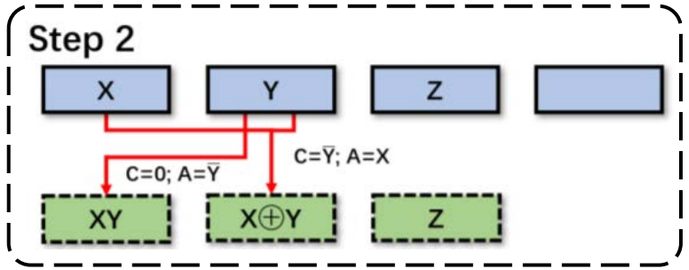
Floating Point Computation: Intra-Array Level Data Transfer

1-Bit Full Addition: Cell Level Data Transfer

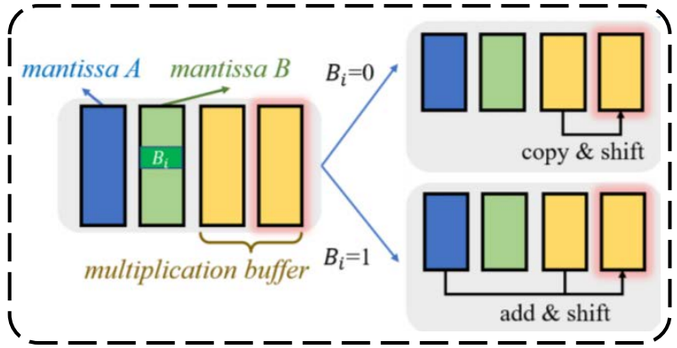
Read and Write: Cell Design



1T-1R structure



4 steps to perform 1-bit full addition

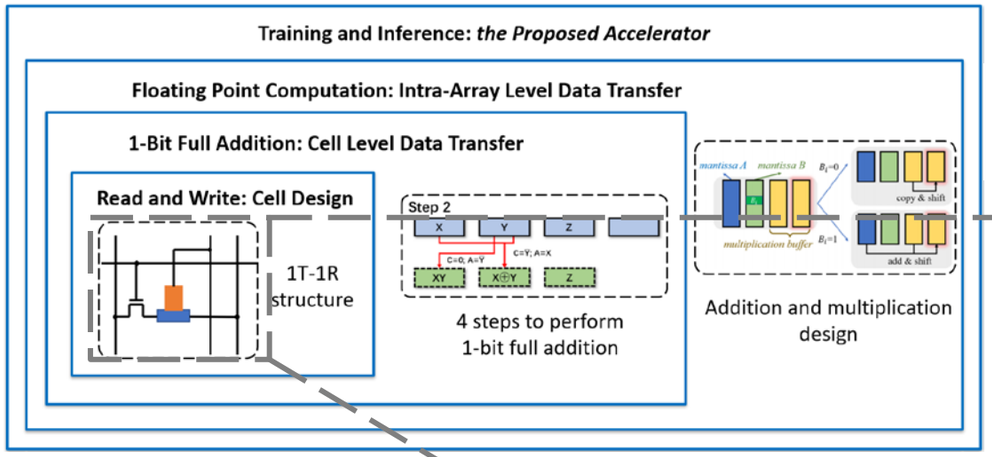


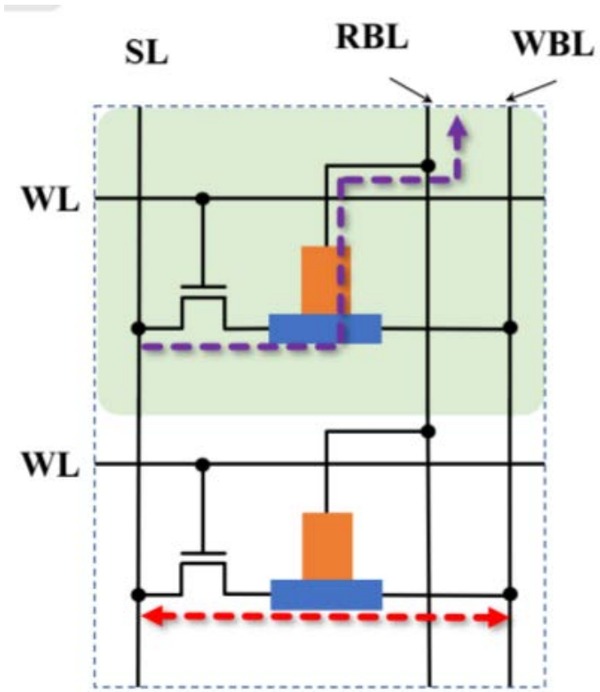
Addition and multiplication design

Outline

- **Background and Motivation**
- ***The Proposed Accelerator:* Overview**
- ***The Proposed Accelerator:* Computing Methodology**
- **Evaluation of *the Proposed Accelerator***
- **Conclusion**

Contribution 1: SOT-MRAM Cell Design

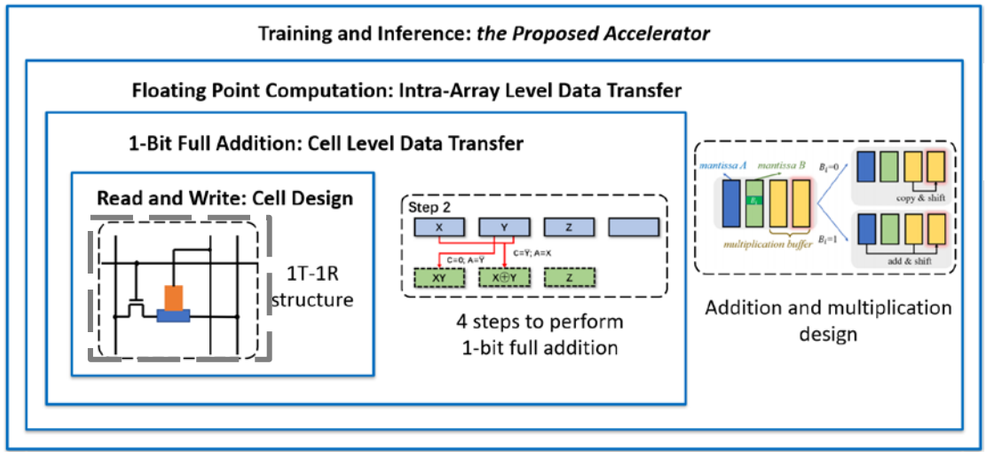


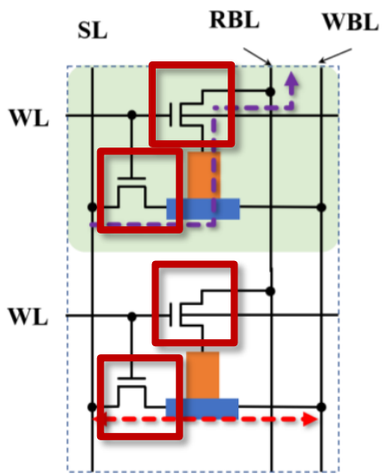


1T-1R *(proposed)*

- **High flexibility**
- **Fair density**

Contribution 1: SOT-MRAM Cell Design

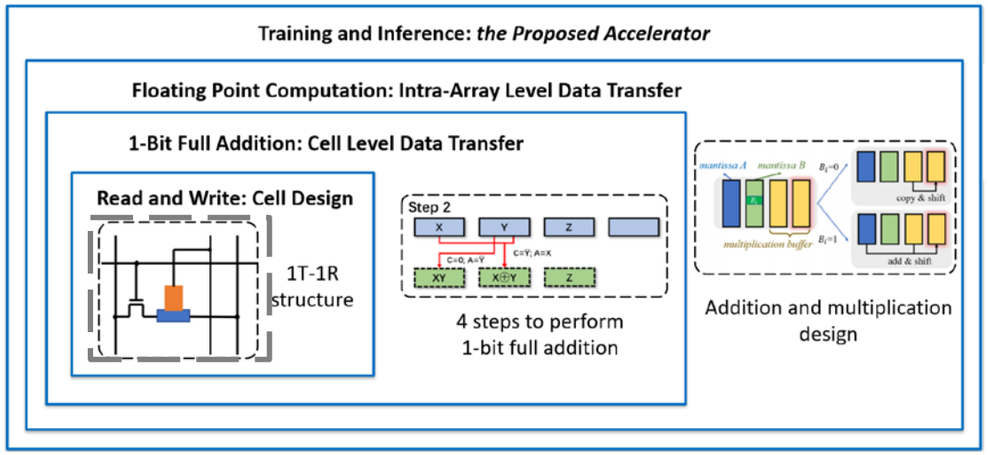


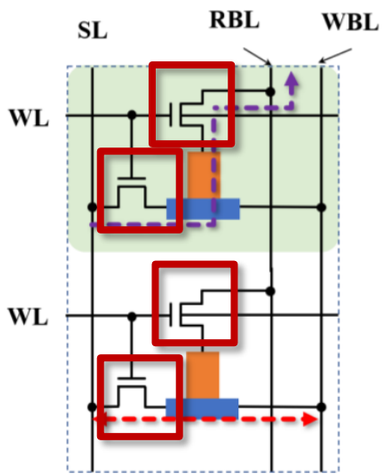


2T-1R

- **High flexibility**
- **Low density**

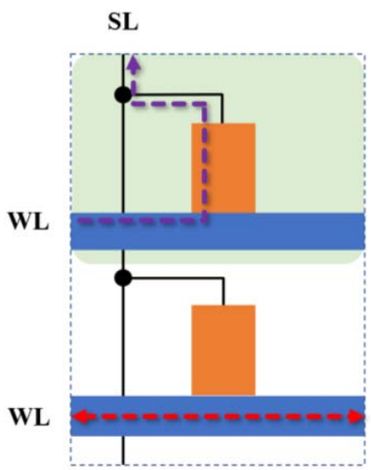
Contribution 1: SOT-MRAM Cell Design





2T-1R

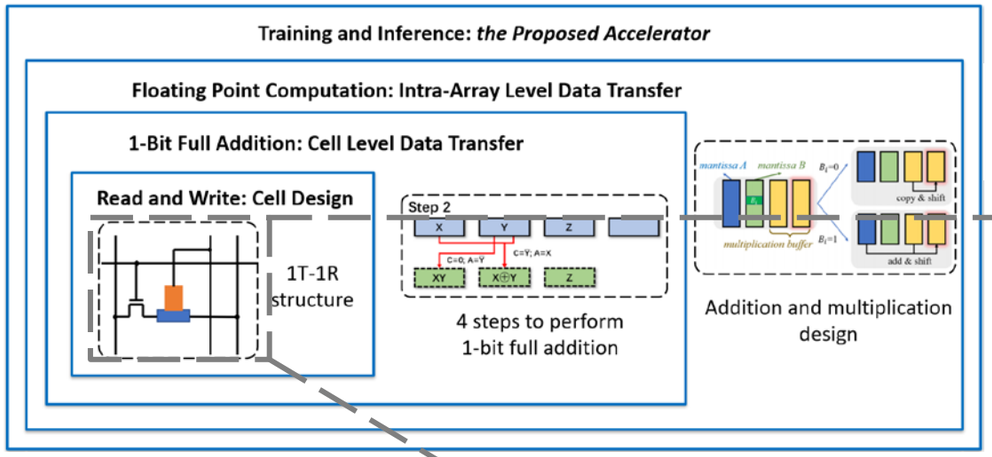
- **High flexibility**
- **Low density**



Crossbar

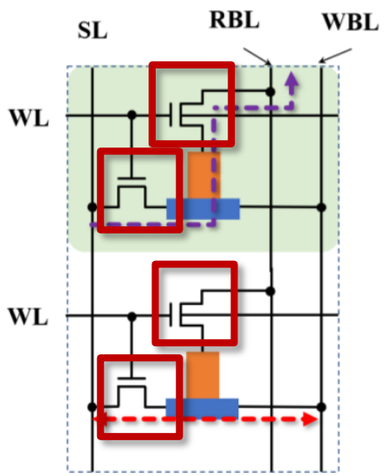
- **High density**
- **Low flexibility**

Contribution 1: SOT-MRAM Cell Design



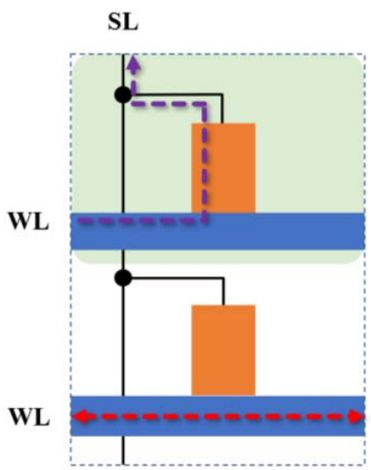
Read current
Write current
MTJ
Heavy metal
One memory cell

transistor in cell



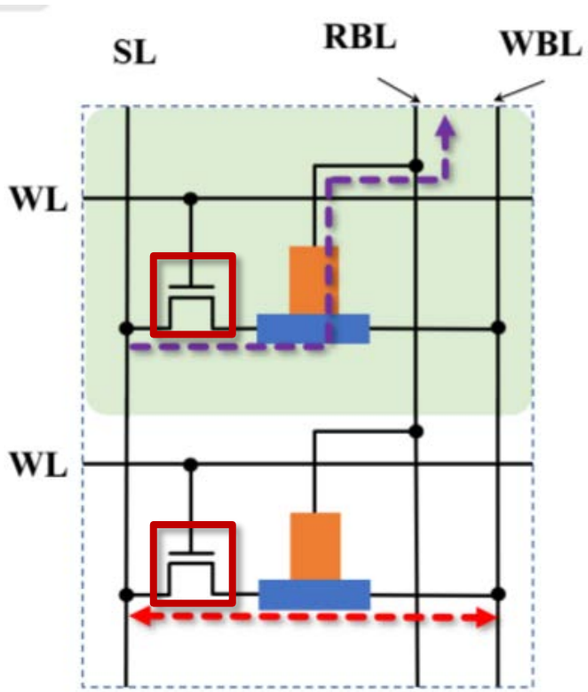
2T-1R

- **High flexibility**
- **Low density**



Crossbar

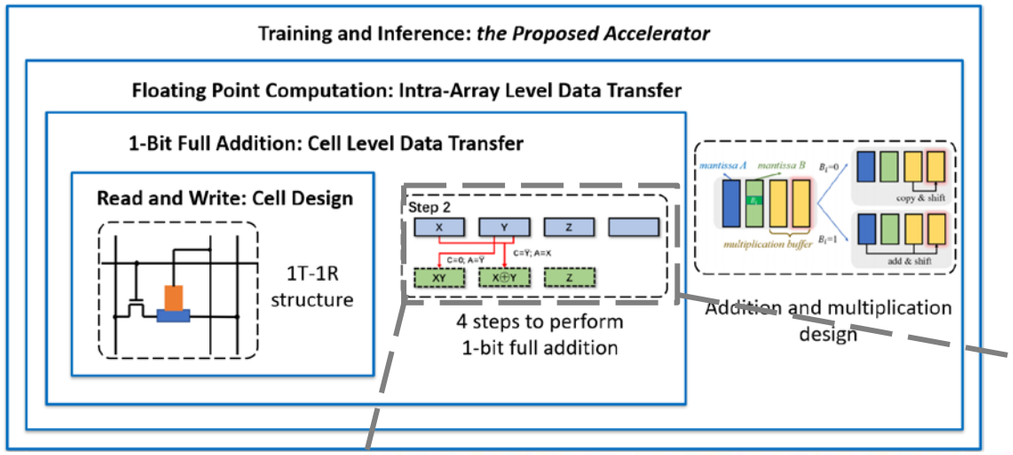
- **High density**
- **Low flexibility**



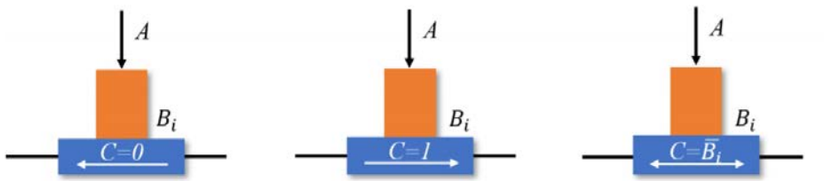
1T-1R *(proposed)*

- **High flexibility**
- **Fair density**

Contribution 2: Full Adder Based on the 1T-1R Cell



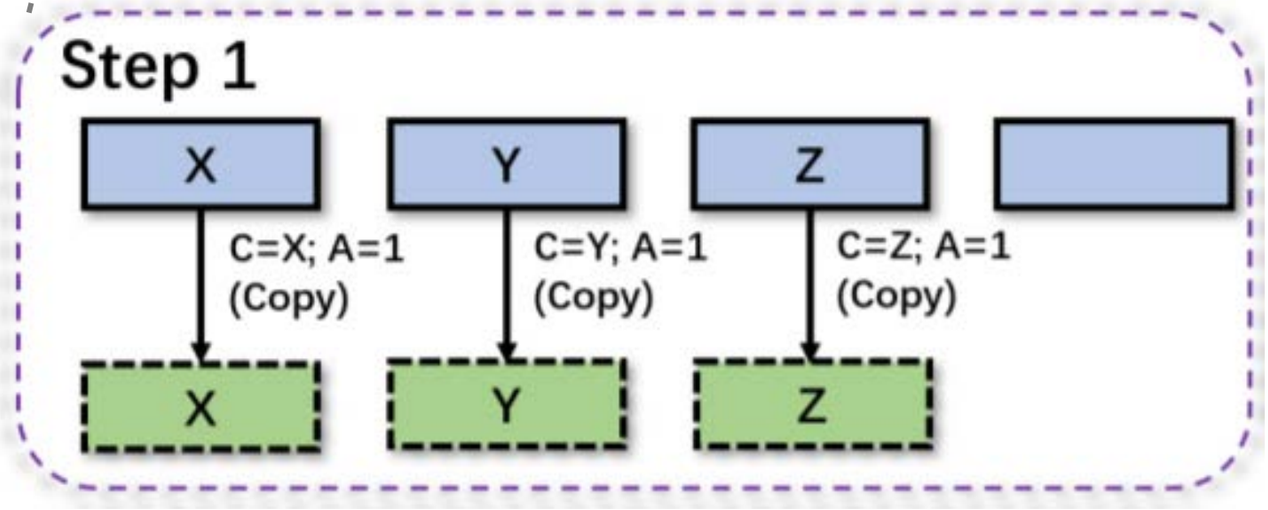
Definition of A, B and C



(a) "AND": $B_{i+1} = \bar{A}B_i$ (b) "OR": $B_{i+1} = A + B_i$ (c) "XOR": $B_{i+1} = A \oplus B_i$

logic functions in the SOT-MRAM cell

[Zhang, T-ED'17]

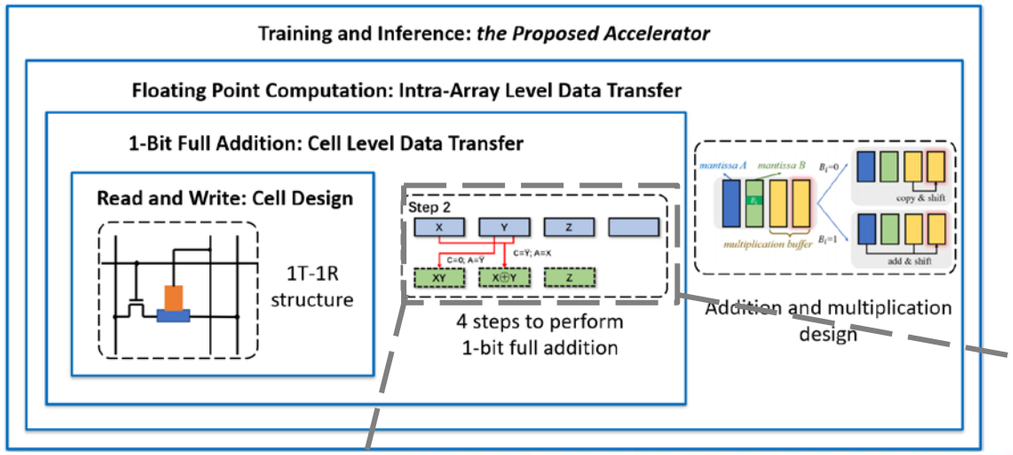


X, Y: operands
Z: input carry
Z': output carry
$S(R)$: sum

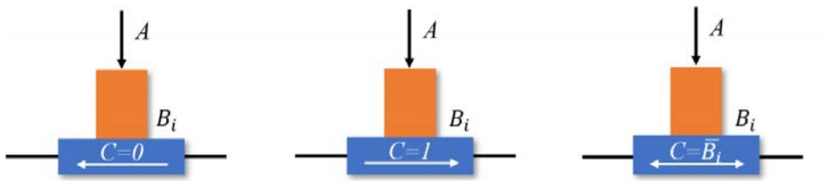
$$S(R) = X \oplus Y \oplus Z$$
$$Z' = XY + Z(X \oplus Y)$$

MRAM | MRAM cache | Data movement | Computation

Contribution 2: Full Adder Based on the 1T-1R Cell



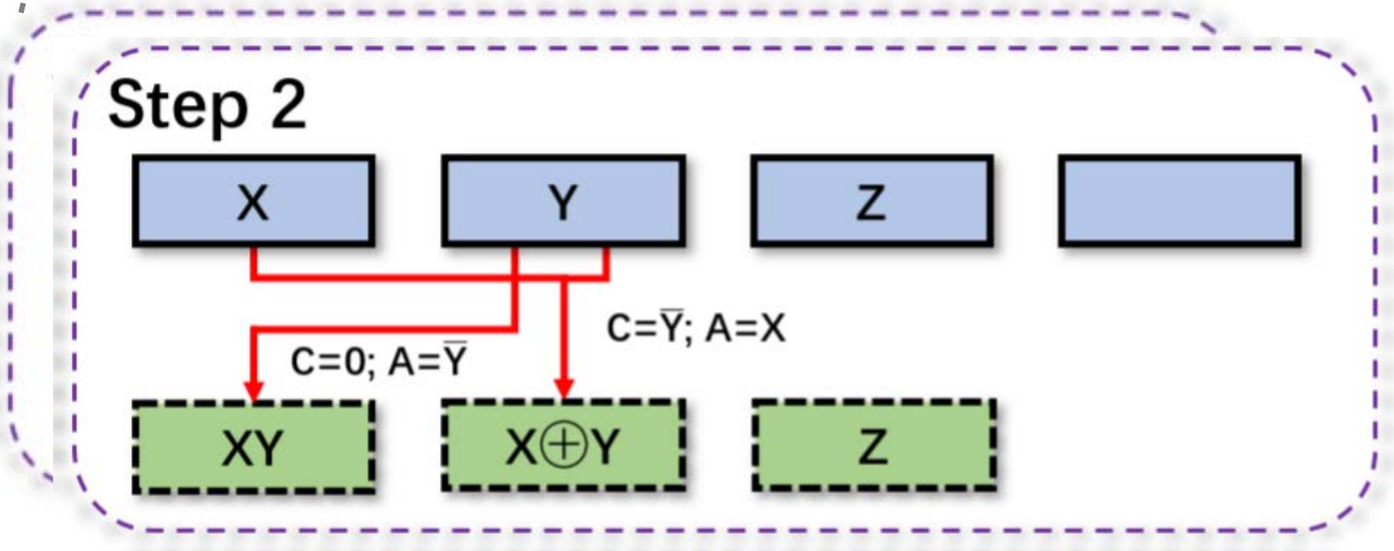
Definition of A, B and C



(a) "AND": $B_{i+1} = \bar{A}B_i$ (b) "OR": $B_{i+1} = A + B_i$ (c) "XOR": $B_{i+1} = A \oplus B_i$

logic functions in the SOT-MRAM cell

[Zhang, T-ED'17]

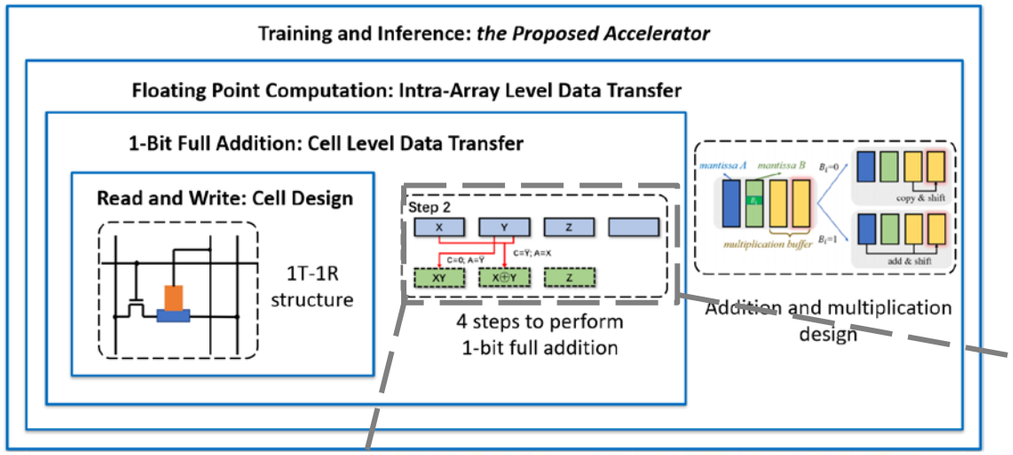


X, Y: operands
Z: input carry
Z': output carry
$S(R)$: sum

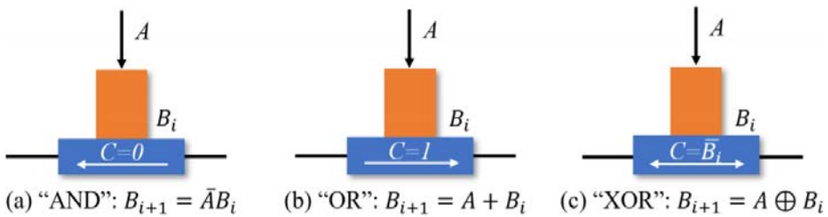
$$S(R) = X \oplus Y \oplus Z$$
$$Z' = XY + Z(X \oplus Y)$$

MRAM | MRAM cache | Data movement | Computation

Contribution 2: Full Adder Based on the 1T-1R Cell

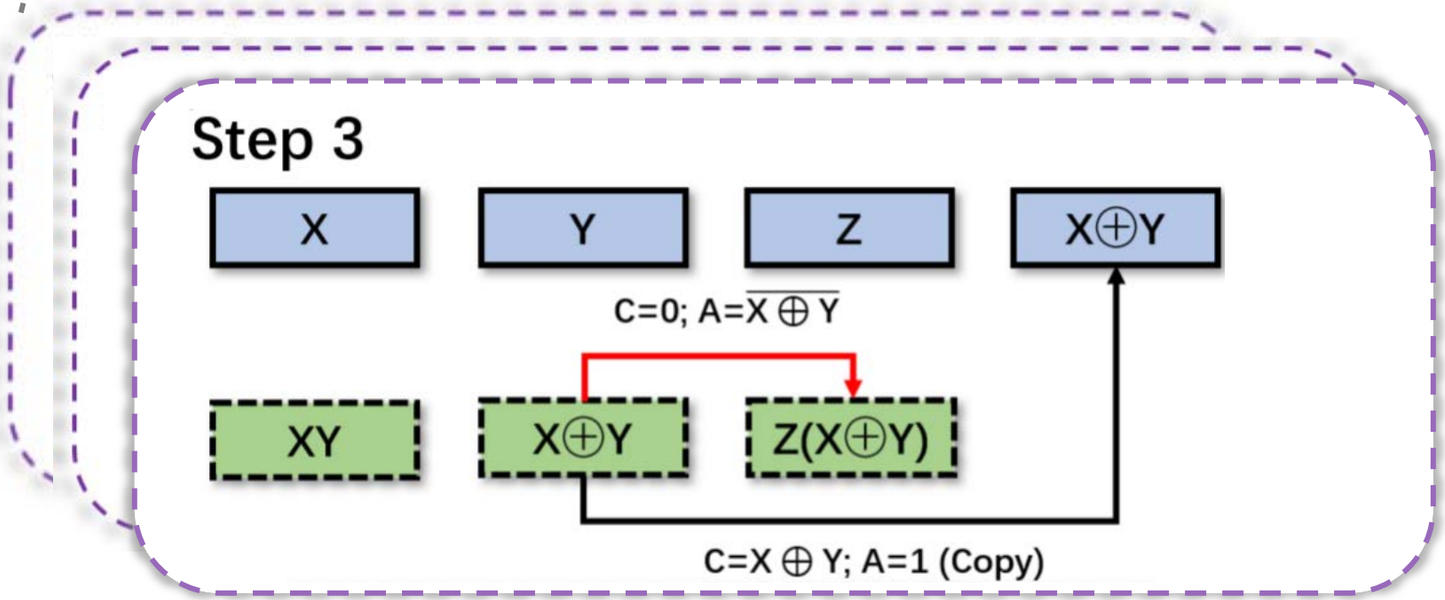


Definition of A, B and C



logic functions in the SOT-MRAM cell

[Zhang, T-ED'17]

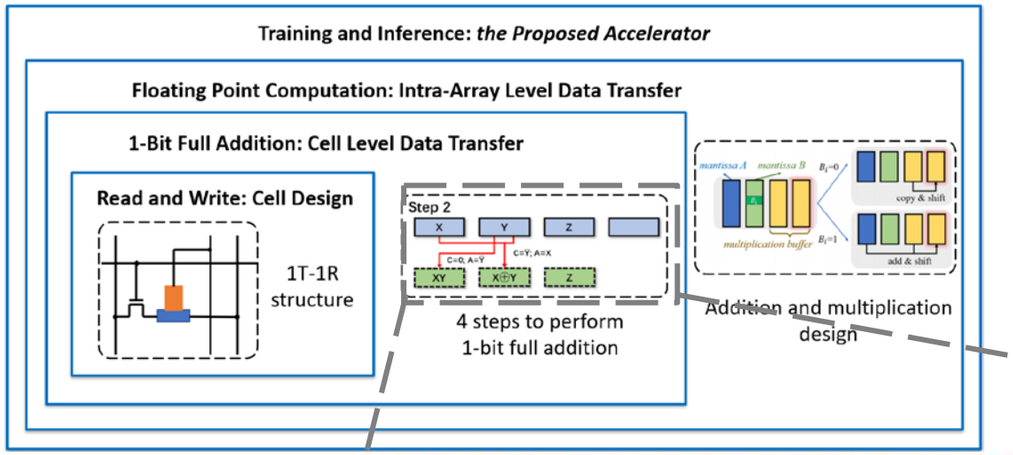


X, Y: operands
Z: input carry
Z': output carry
$S(R)$: sum

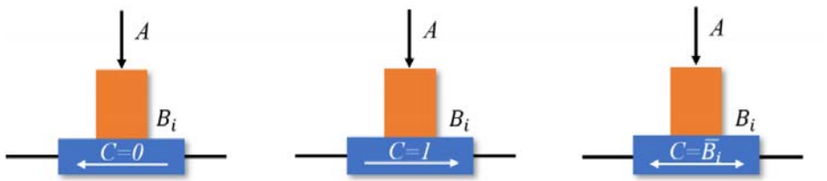
$$S(R) = X \oplus Y \oplus Z$$
$$Z' = XY + Z(X \oplus Y)$$

MRAM | MRAM cache | Data movement | Computation

Contribution 2: Full Adder Based on the 1T-1R Cell



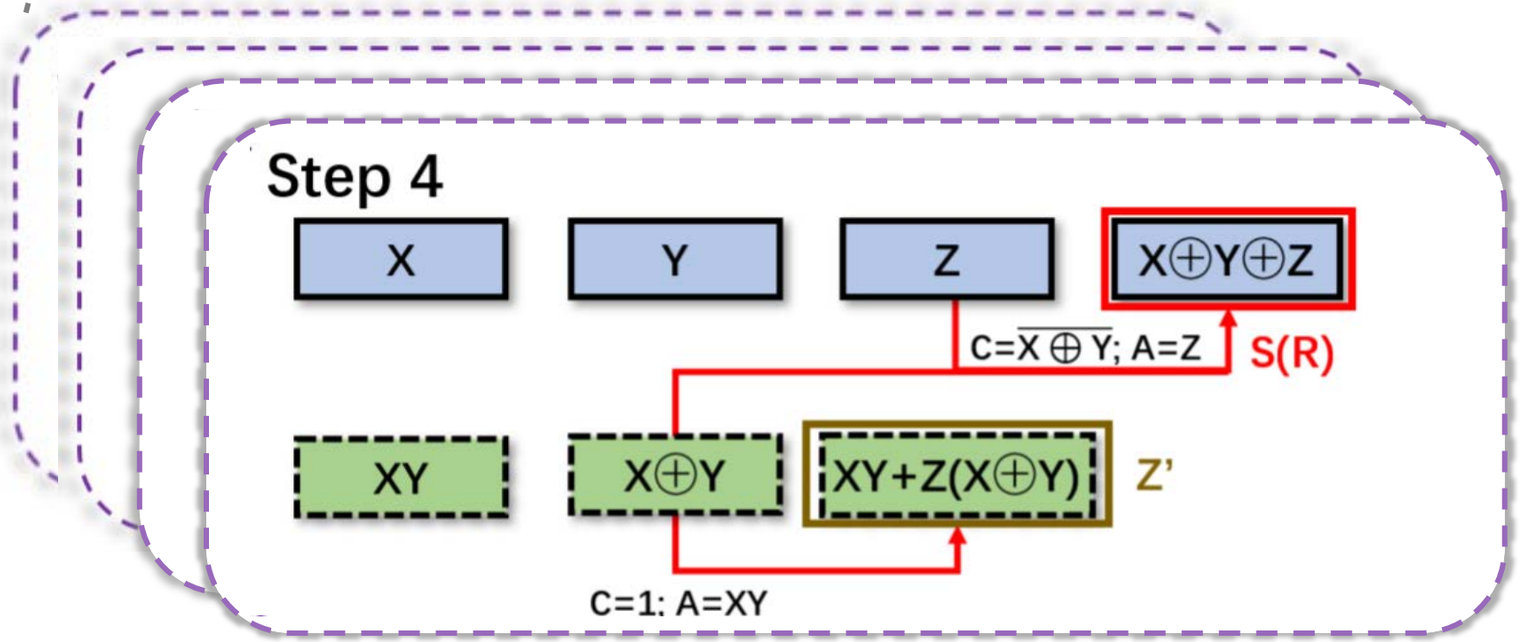
Definition of A, B and C



(a) "AND": $B_{i+1} = \bar{A}B_i$ (b) "OR": $B_{i+1} = A + B_i$ (c) "XOR": $B_{i+1} = A \oplus B_i$

logic functions in the SOT-MRAM cell

[Zhang, T-ED'17]



X, Y: operands

Z: input carry

Z': output carry

$S(R)$: sum

$$S(R) = X \oplus Y \oplus Z$$
$$Z' = XY + Z(X \oplus Y)$$

MRAM | MRAM cache | Data movement | Computation

Complexity of the Proposed Full Adder Over SOTA One

Compare Complexity in Terms of Required Clock Cycles and Memory Cells

PIM Implementation	Read and Write Clock Cycles	Extra Memory Cells
The Proposed Method	4	3
FloatPIM [Imani, ISCA'19]	13	10

Reduced by: **69.2%↓** **70.0%↓**

Available Boolean Function vs. Computational Complexity

FloatPIM [Imani, ISCA'19]

$$S(R) = \overline{\overline{\overline{X}+\overline{Y}+\overline{Z}}+\overline{\overline{X+Y+Z}}+\overline{\overline{\overline{X}+\overline{Y}}+\overline{\overline{X}+\overline{Z}}+\overline{\overline{Y}+\overline{Z}}}}$$

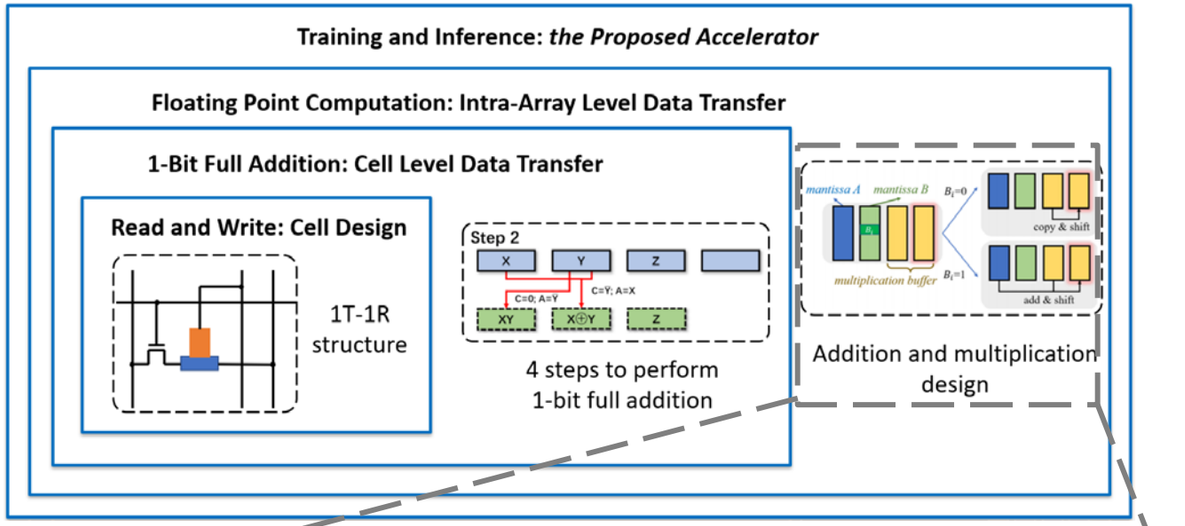
$$Z' = \overline{\overline{X+Y}+\overline{X+Z}+\overline{Y+Z}}$$

The Proposed Method

$$S(R) = X \oplus Y \oplus Z$$

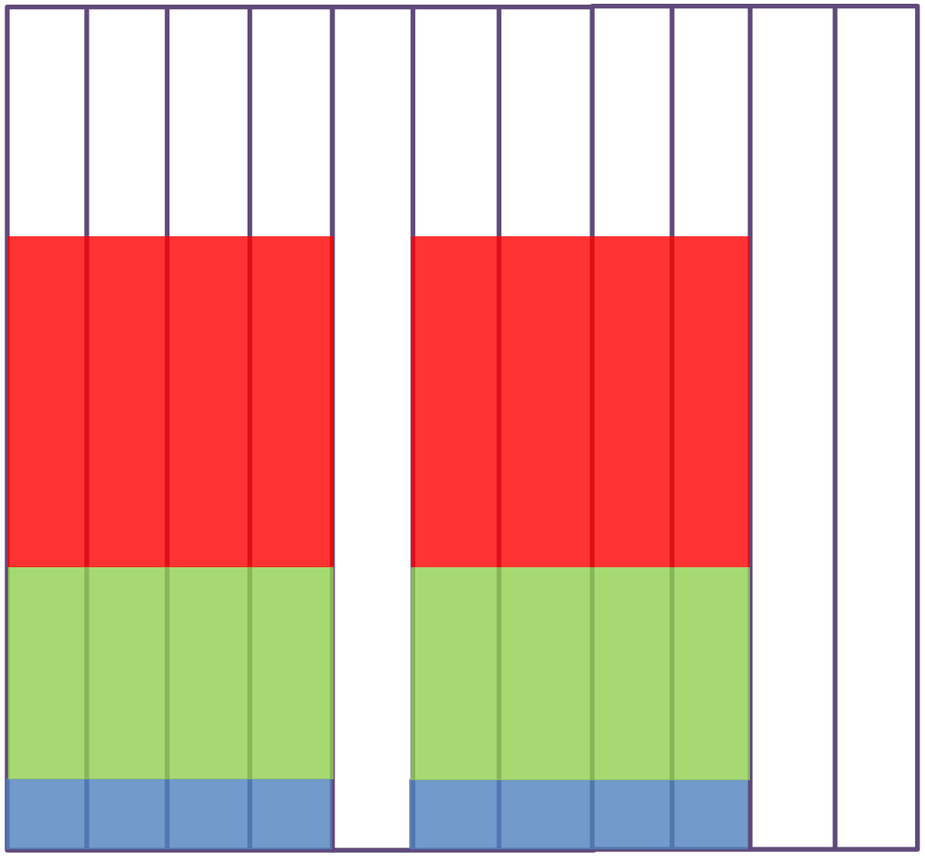
$$Z' = XY + Z(X \oplus Y)$$

Contribution 3: Improved Floating Point Computation



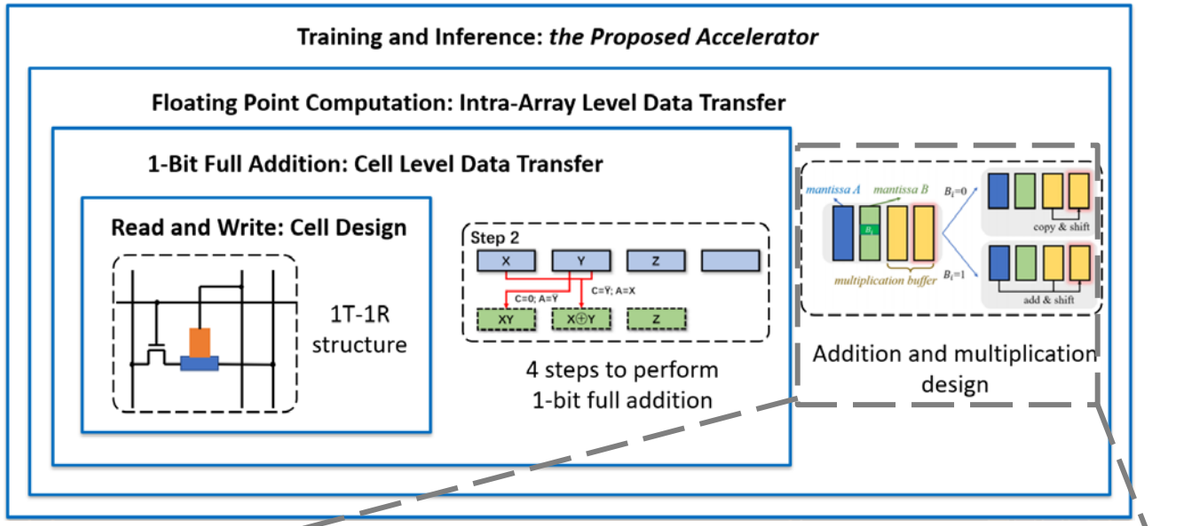
	sign	exp	mantissa
A_i	0	10000000	11000000000000000000000
		2^1	(1).75
B_i	0	10000010	01100000000000000000000
		2^3	(1).375

Floating Point Addition: $A_i + B_i$



$A_0\ A_1\ A_2\ A_3\ \ldots\ B_0\ B_1\ B_2\ B_3\ \ldots$

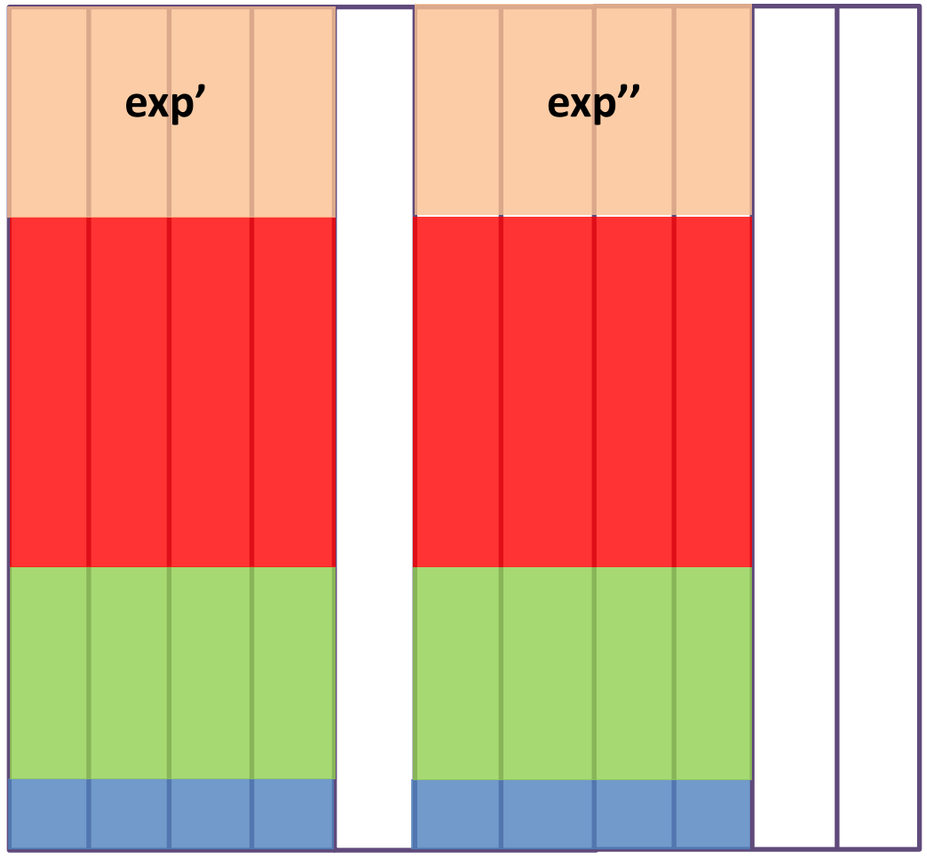
Contribution 3: Improved Floating Point Computation



	sign	exp	mantissa
A_i	0	10000000	11000000000000000000000
		2^1	(1).75
B_i	0	10000010	01100000000000000000000
		2^3	(1).375

00000010 exp’ = exp(B) – exp(A)

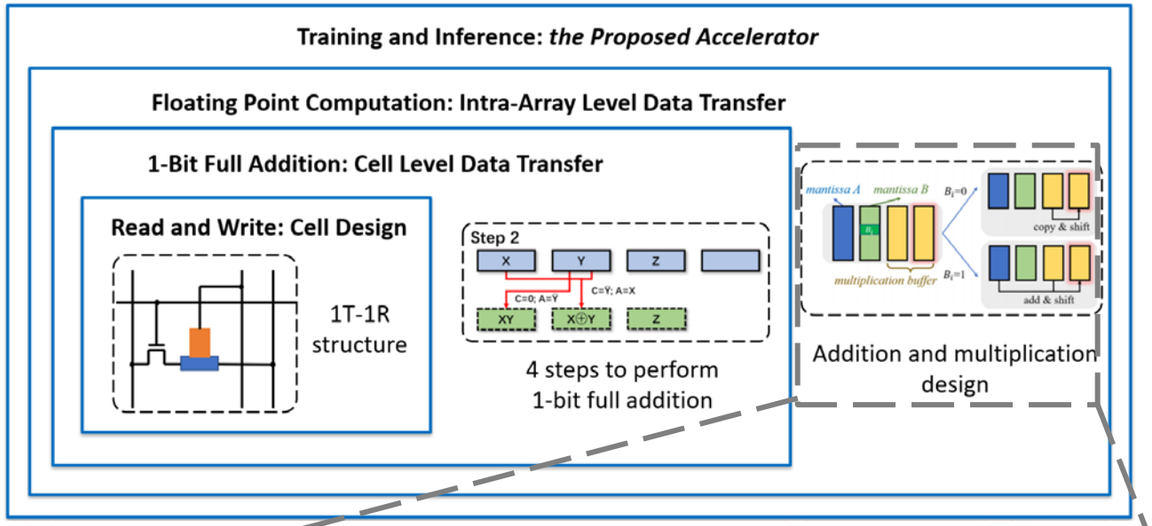
11111110 exp’’ = exp(A) – exp(B)



A_0 A_1 A_2 A_3 … B_0 B_1 B_2 B_3 …

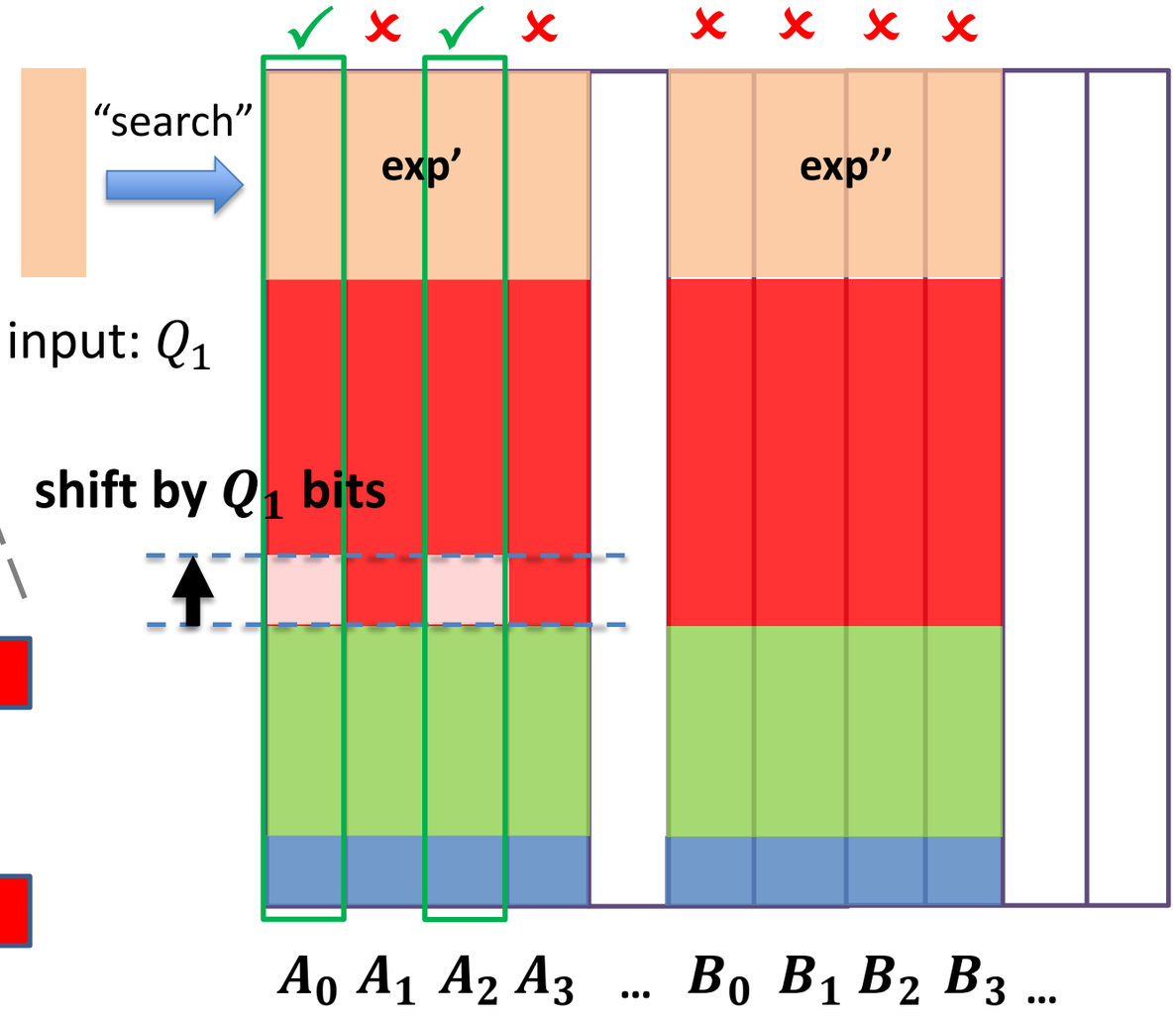
Column wise parallelism

Contribution 3: Improved Floating Point Computation



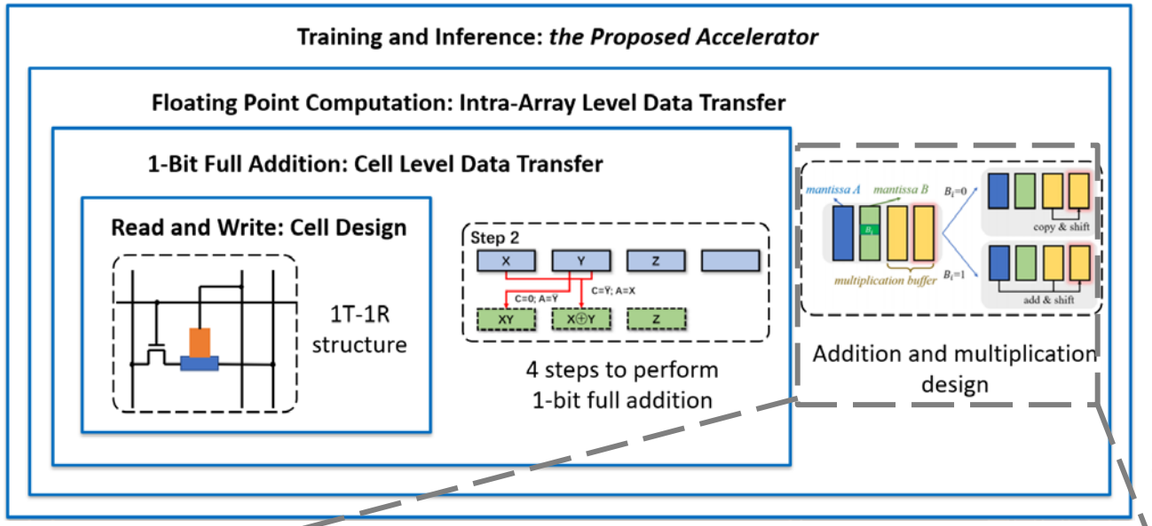
	sign	exp	mantissa
A_i	0	10000010	01110000000000000000000
		2^3	(0).4375
B_i	0	10000010	01100000000000000000000
		2^3	(1).375

right shift mantissa to align exponents



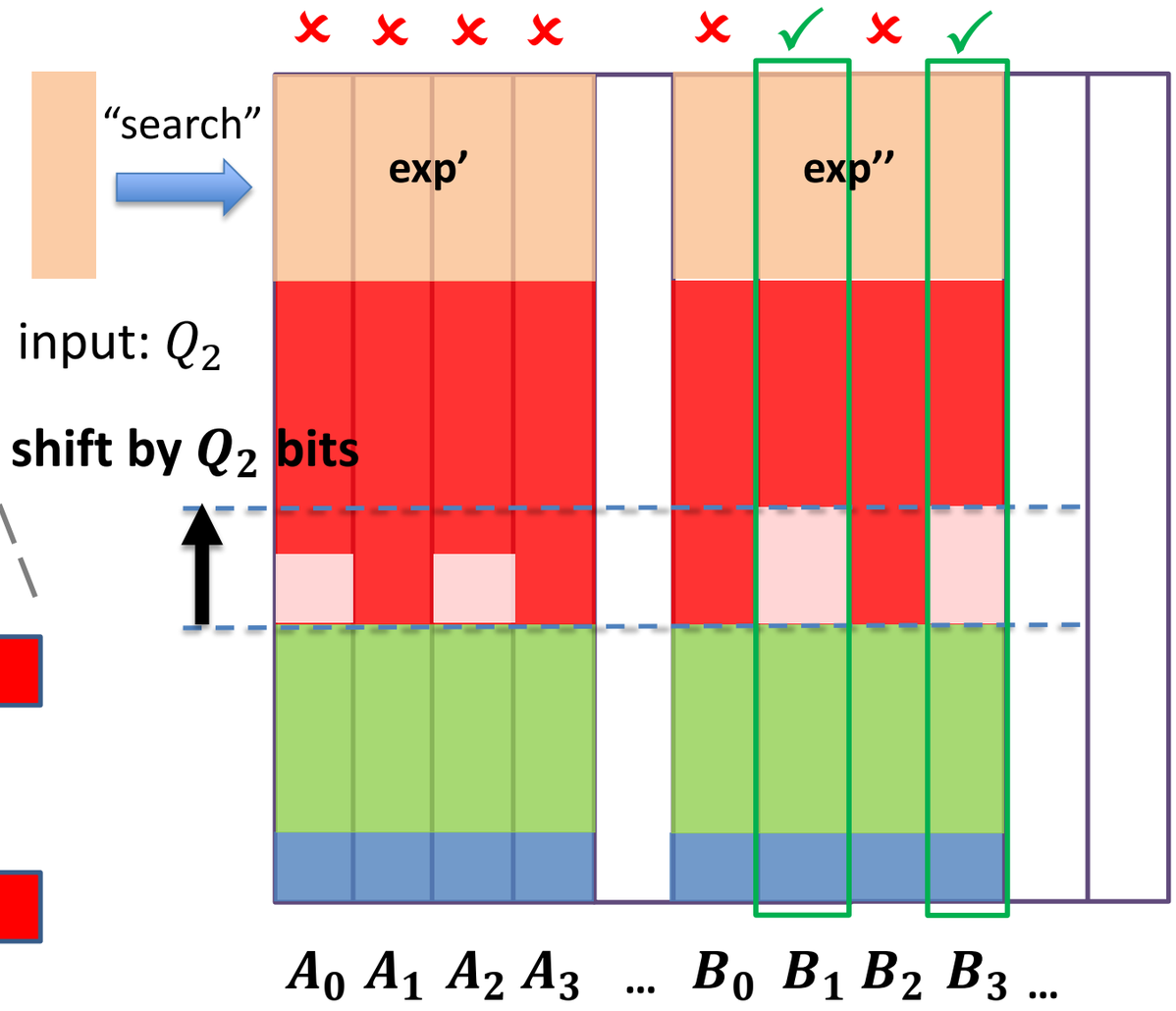
Column wise parallelism
(for same exp' or exp'')

Contribution 3: Improved Floating Point Computation



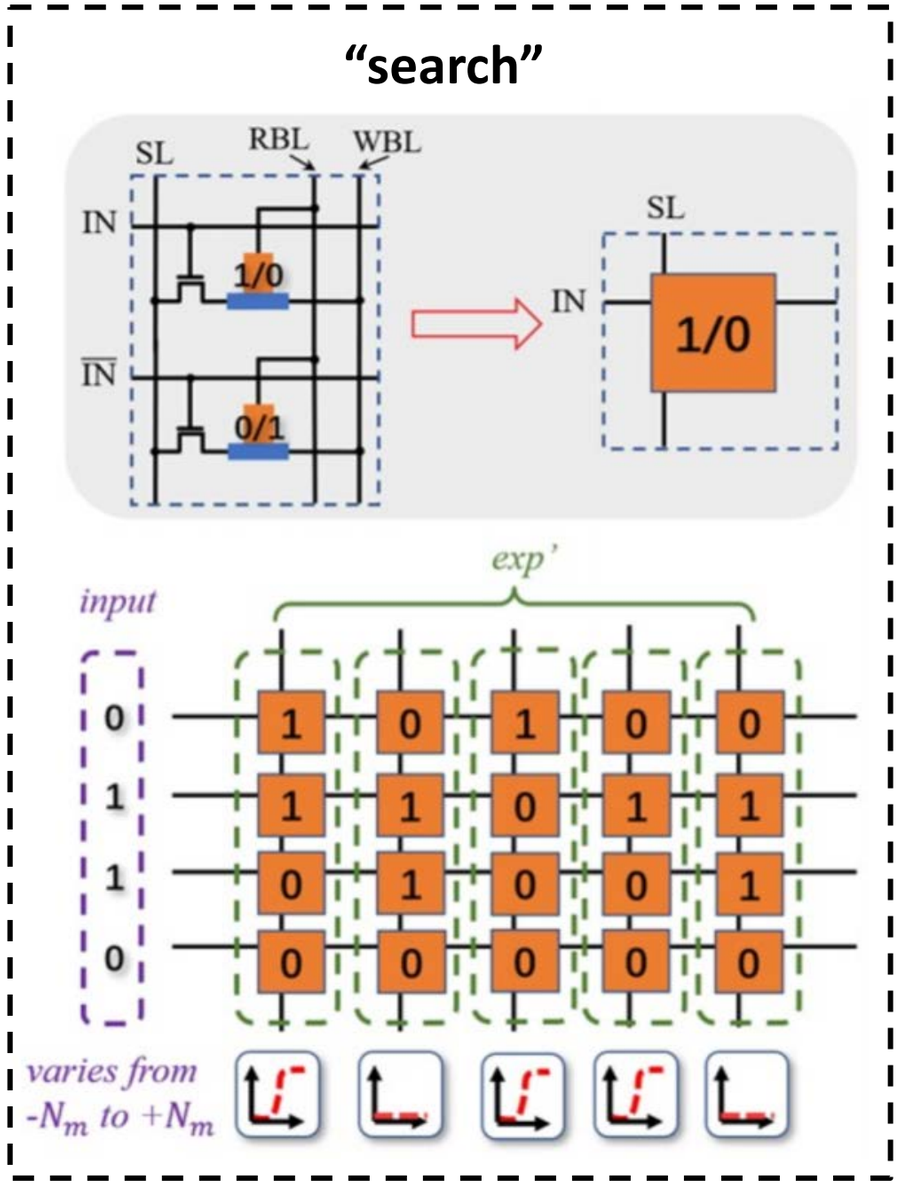
	sign	exp	mantissa
A_i	0	10000010	01110000000000000000000
		2^3	(0).4375
B_i	0	10000010	01100000000000000000000
		2^3	(1).375

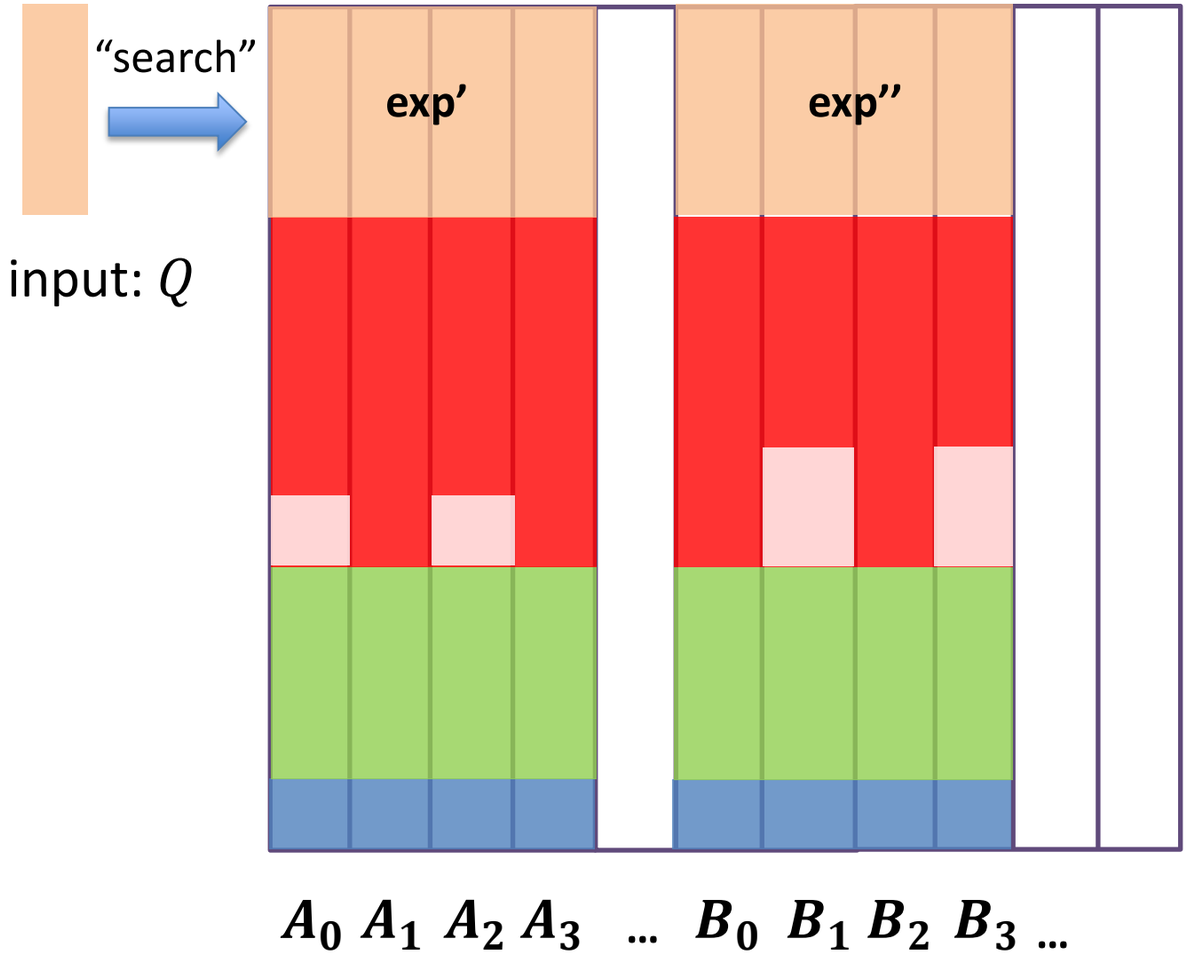
right shift mantissa to align exponents



Column wise parallelism
(for same exp' or exp'')

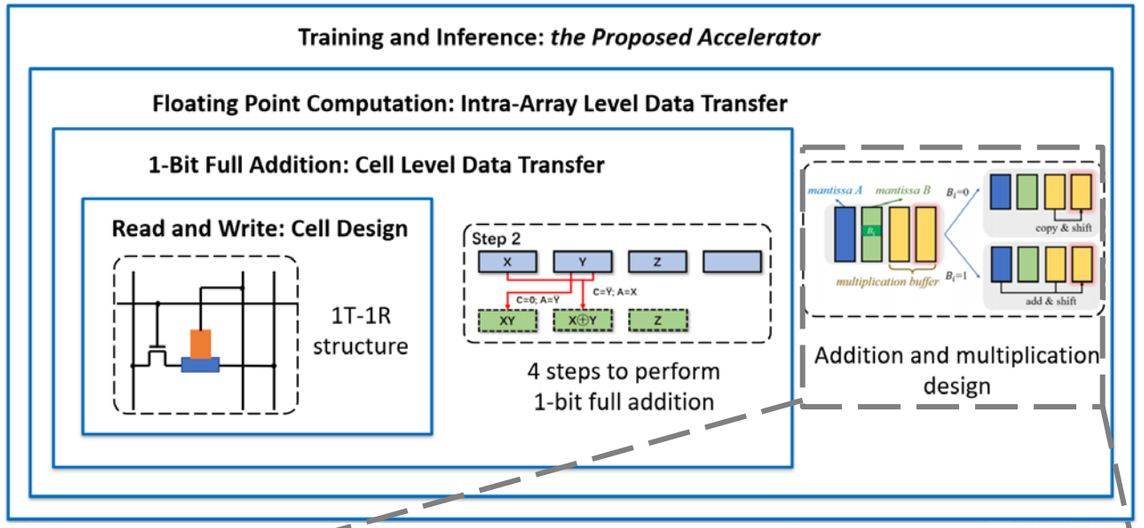
Contribution 3: Improved Floating Point Computation





Column wise parallelism
(for same exp' or exp'')

Contribution 3: Improved Floating Point Computation



Floating Point Multiplication

XOR between sign bits

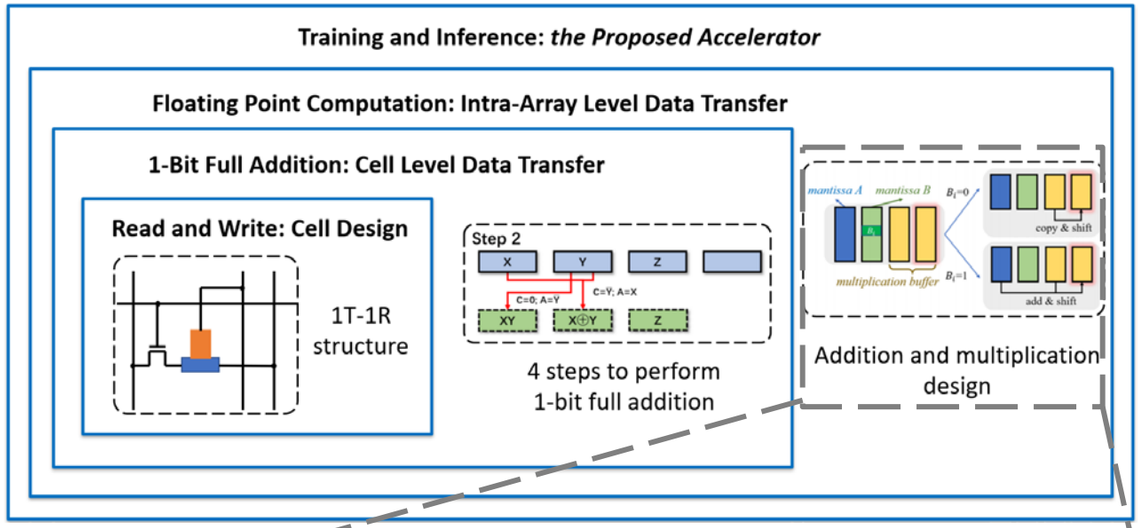
↓

exponents addition

↓

mantissas multiplication

Contribution 3: Improved Floating Point Computation



Floating Point Multiplication

XOR between sign bits

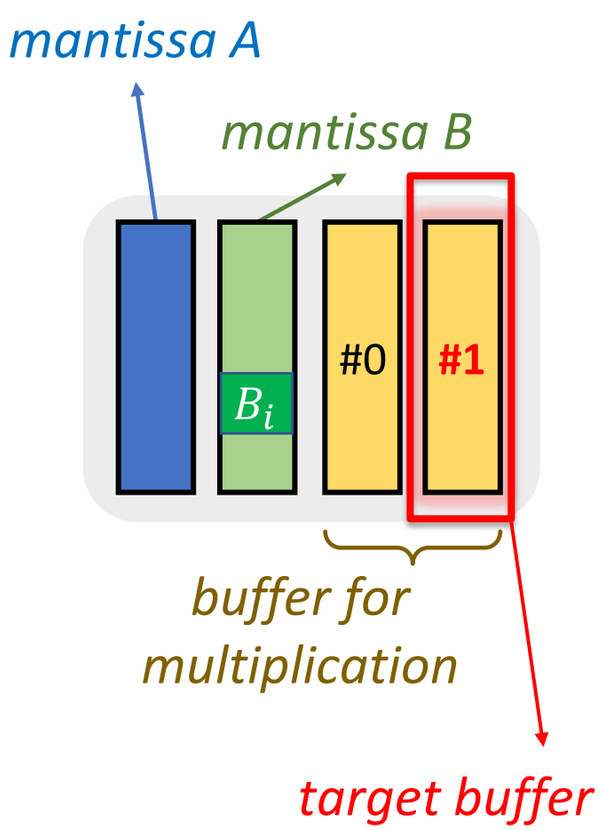
↓

exponents addition

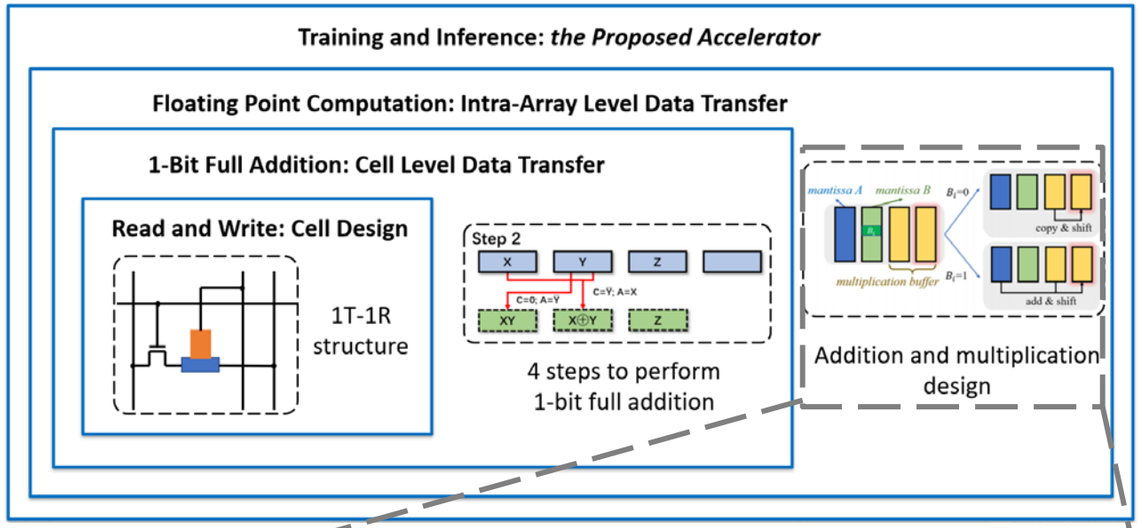
↓

mantissas multiplication

Target Buffer in a Ping-Pong Manner



Contribution 3: Improved Floating Point Computation



Floating Point Multiplication

XOR between sign bits

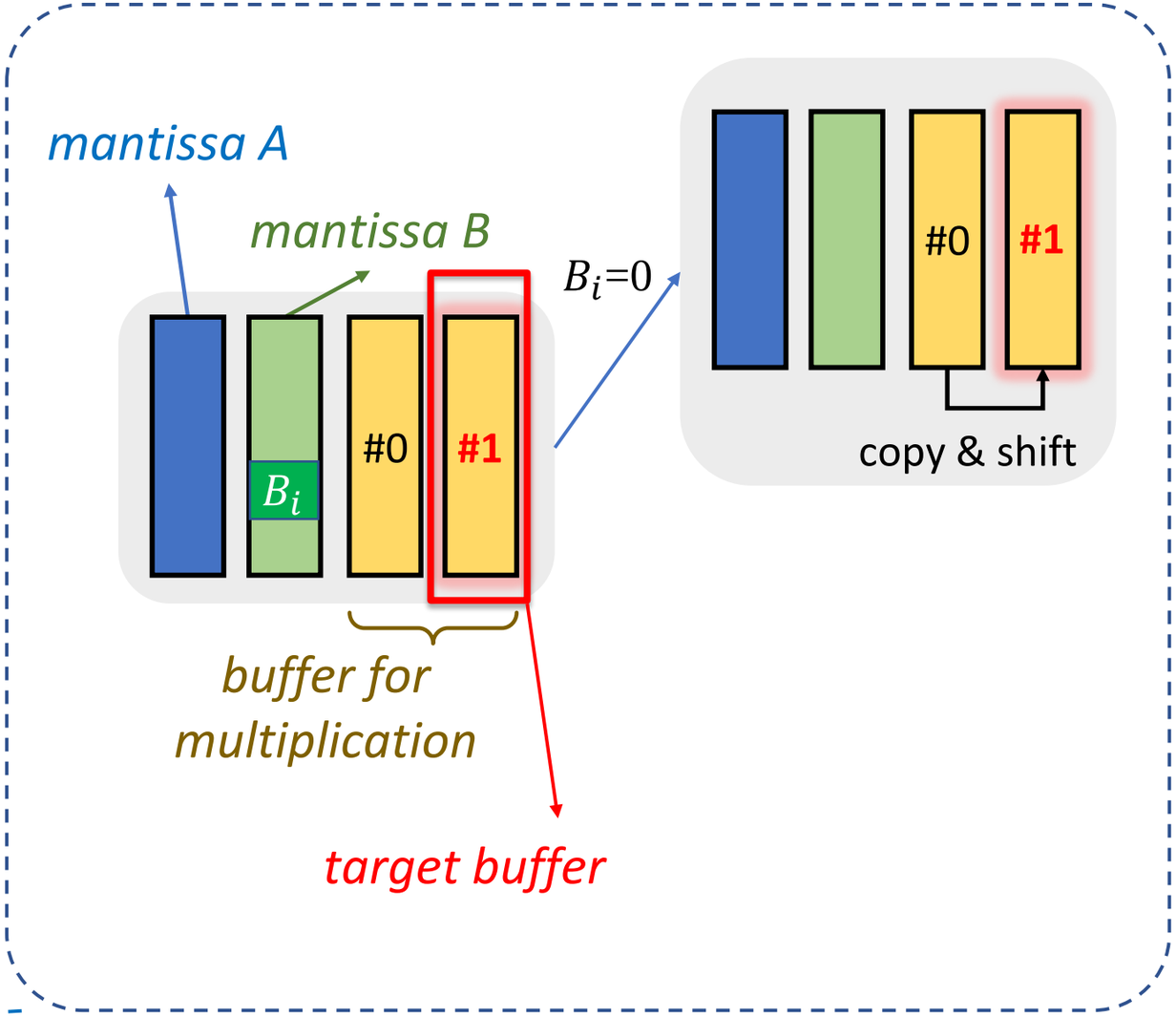
↓

exponents addition

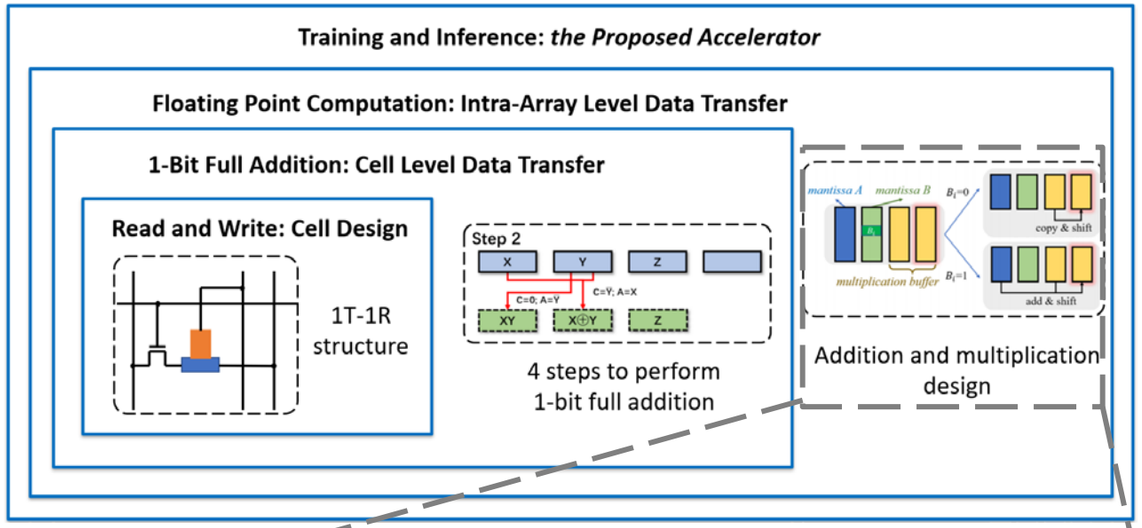
↓

mantissas multiplication

Target Buffer in a Ping-Pong Manner



Contribution 3: Improved Floating Point Computation



Floating Point Multiplication

XOR between sign bits

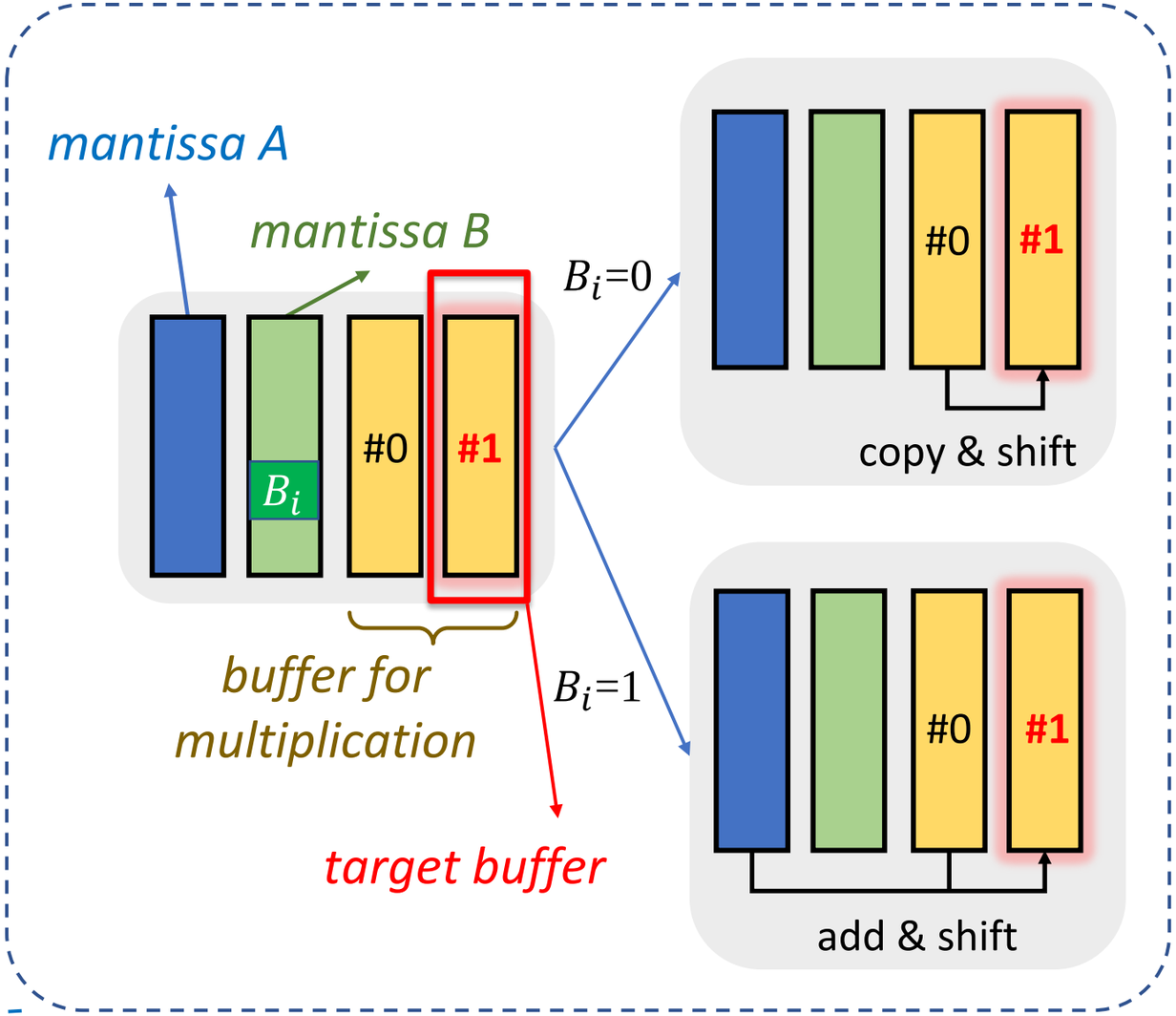
↓

exponents addition

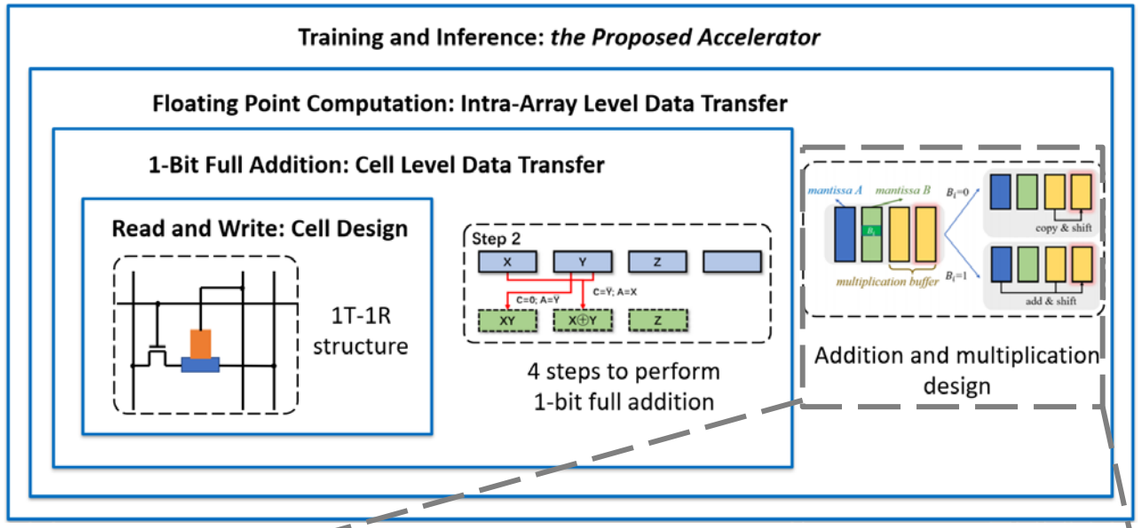
↓

mantissas multiplication

Target Buffer in a Ping-Pong Manner



Contribution 3: Improved Floating Point Computation



Floating Point Multiplication

XOR between sign bits

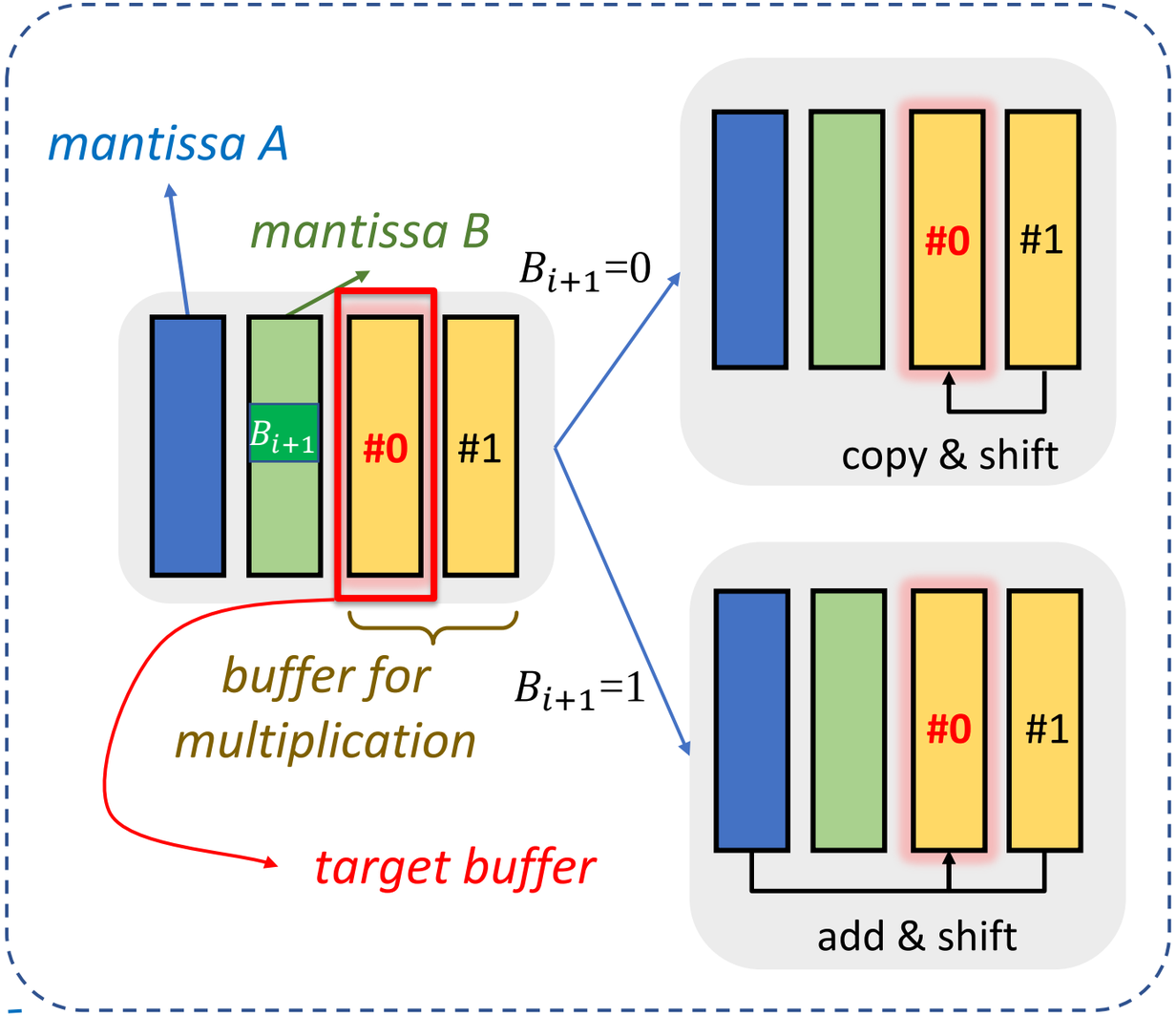
↓

exponents addition

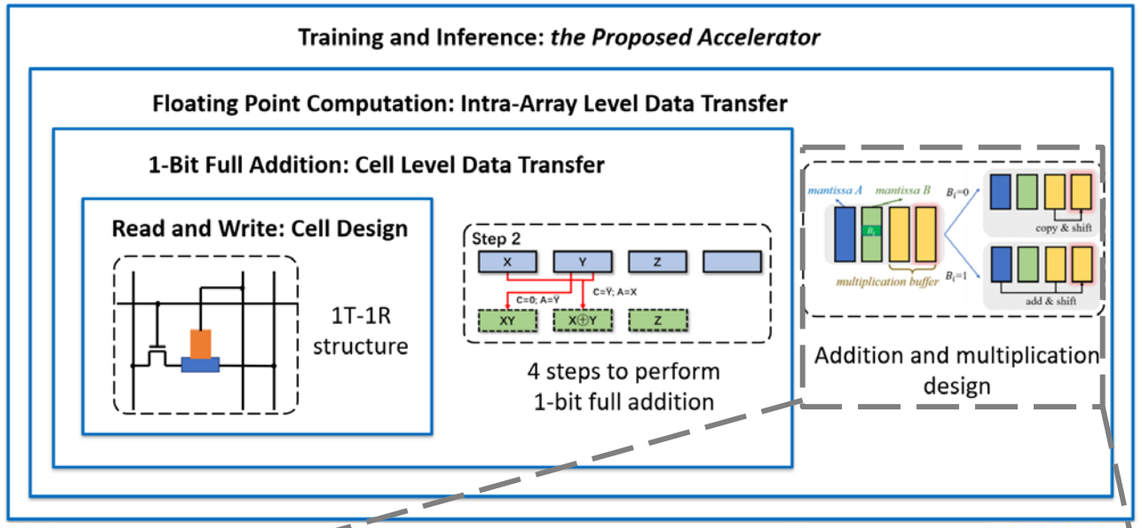
↓

mantissas multiplication

Target Buffer in a Ping-Pong Manner



Contribution 3: Improved Floating Point Computation



Floating Point Multiplication

XOR between sign bits

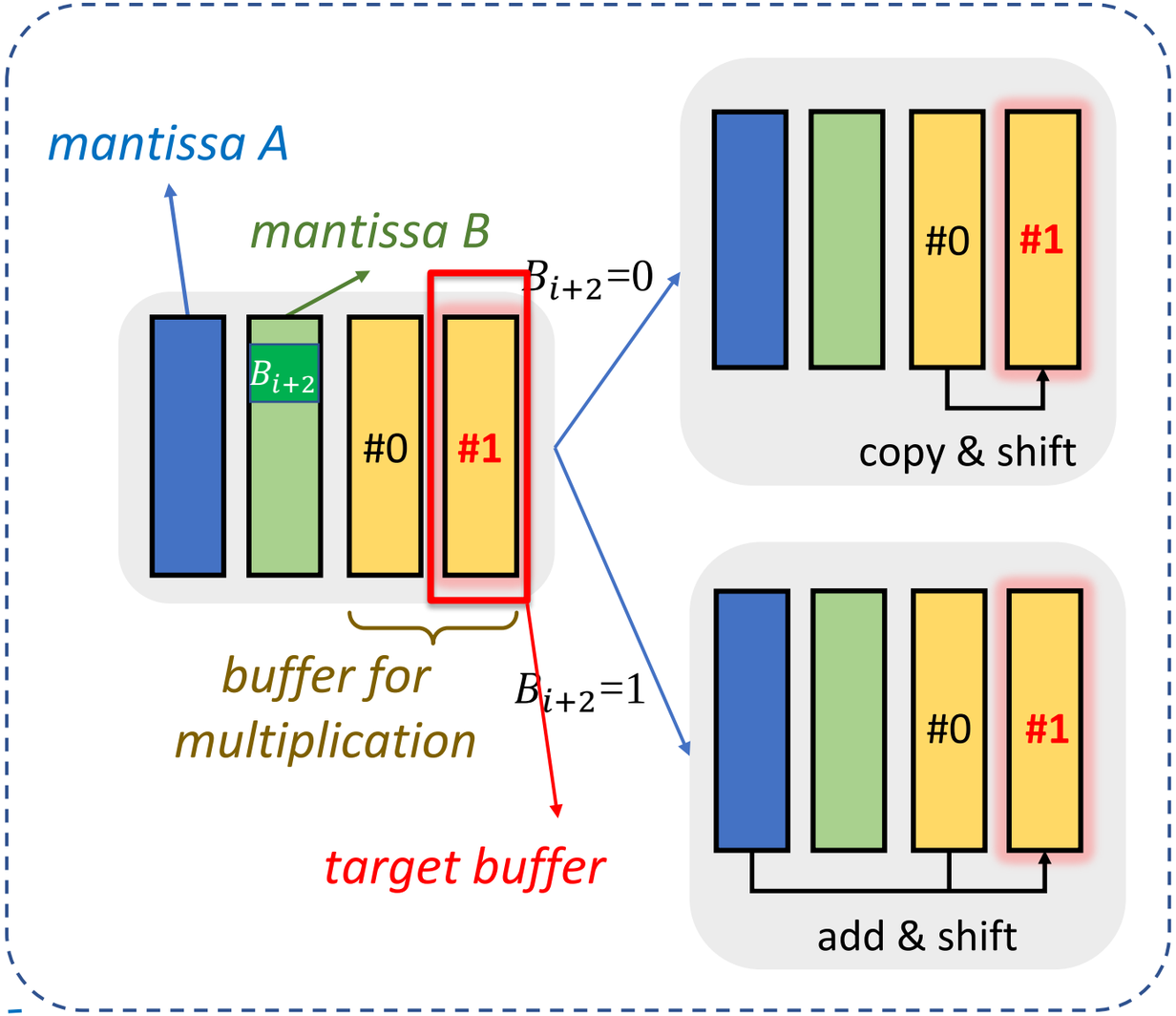
↓

exponents addition

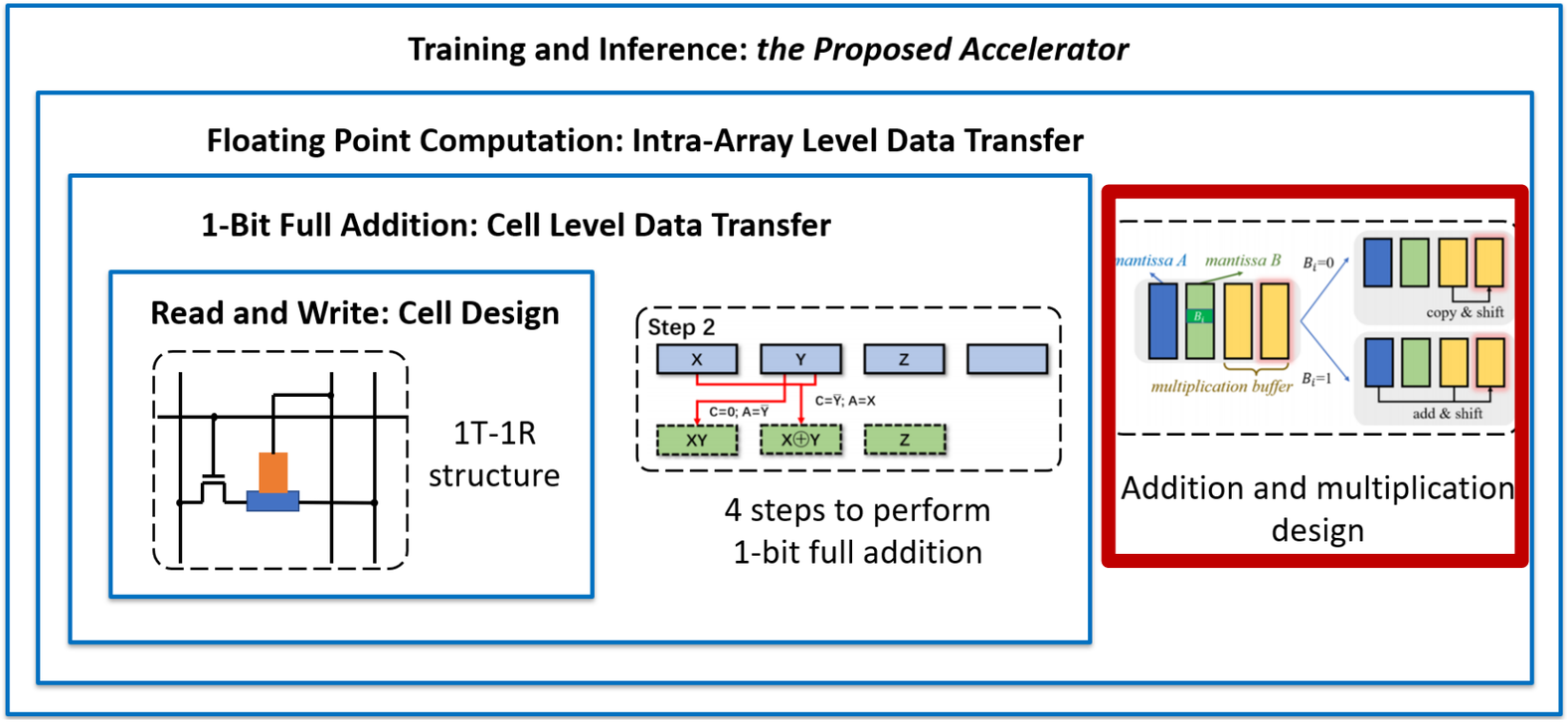
↓

mantissas multiplication

Target Buffer in a Ping-Pong Manner



The Proposed Over SOTA Floating Point Computation



Compare with state-of-the-art work **FloatPIM** [Imani, ISCA'19]

- **Addition**: # of read and write clock cycles reduced by **80.5%↓**, from $O(N^2)$ **to** $O(N)$
- **Multiplication**: # of read and write clock cycles reduced by **81.1%↓**, # of memory cells requirement reduced by **77.4%↓**

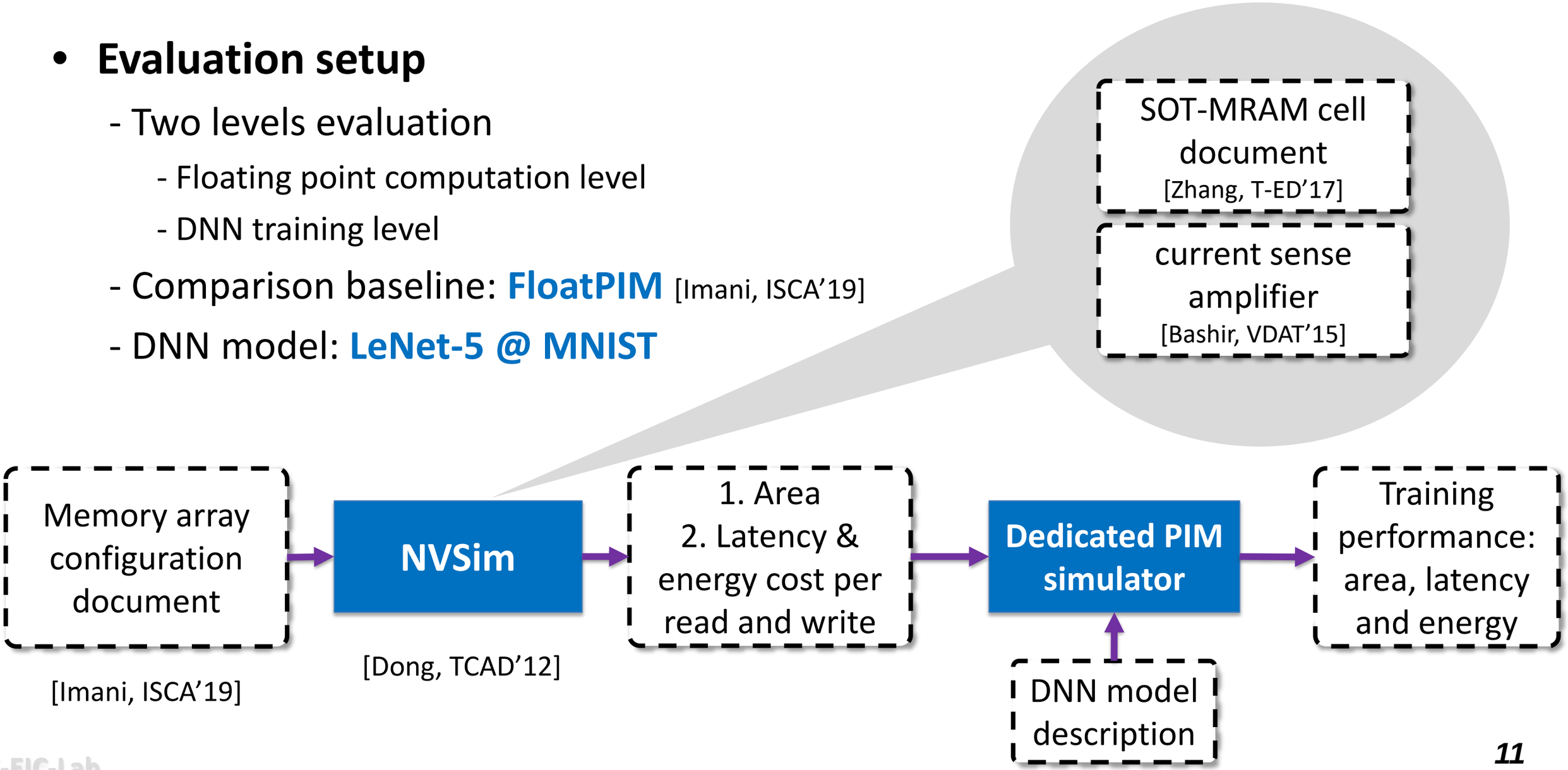
for 32-bit floating point numbers

Outline

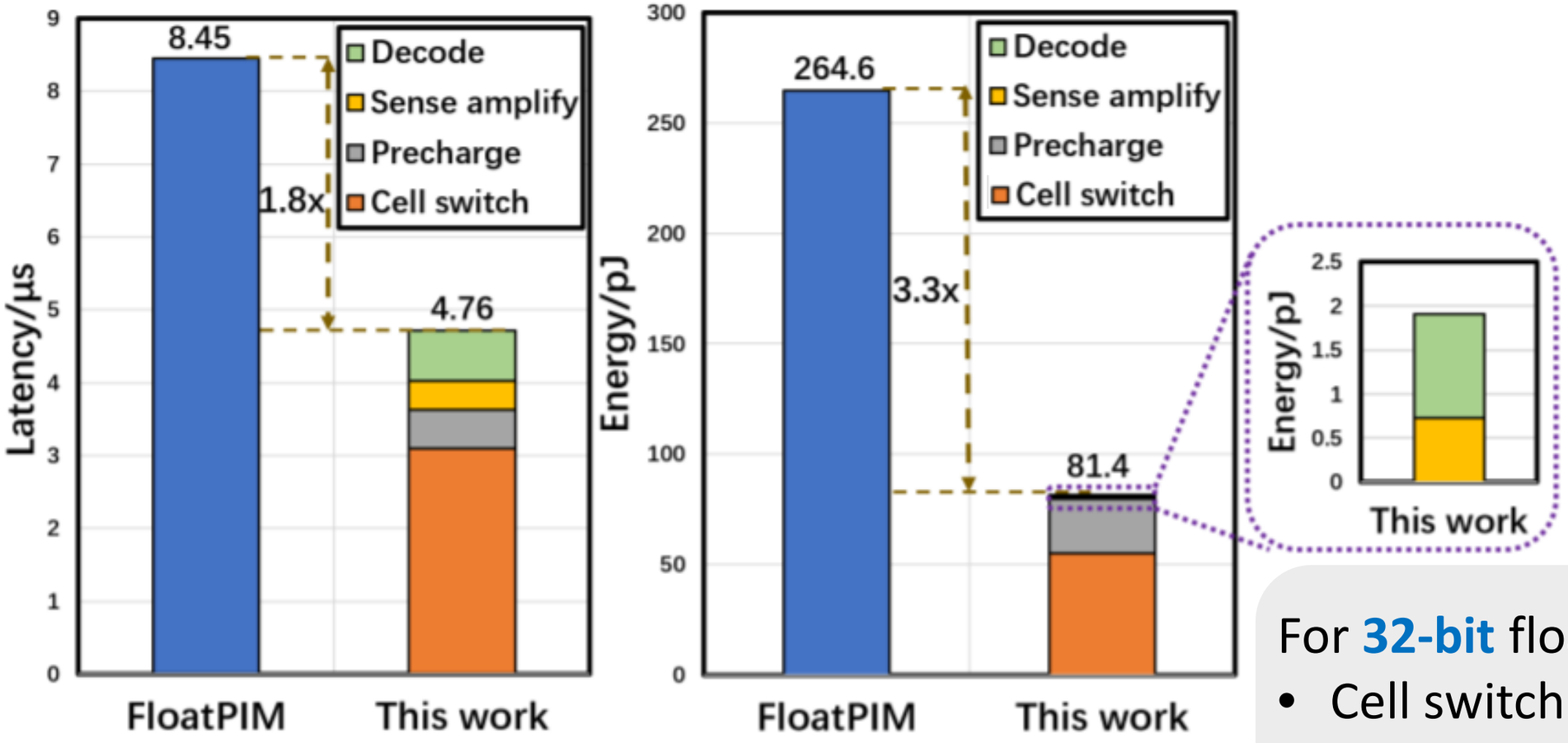
- **Background and Motivation**
- ***The Proposed Accelerator:* Overview**
- ***The Proposed Accelerator:* Computing Methodology**
- **Evaluation of *the Proposed Accelerator***
- **Conclusion**

Evaluation Setting of *the Proposed Accelerator*

- **Evaluation setup**
  - Two levels evaluation
    - Floating point computation level
    - DNN training level
  - Comparison baseline: **FloatPIM** [Imani, ISCA'19]
  - DNN model: **LeNet-5 @ MNIST**



Evaluation Results on MAC

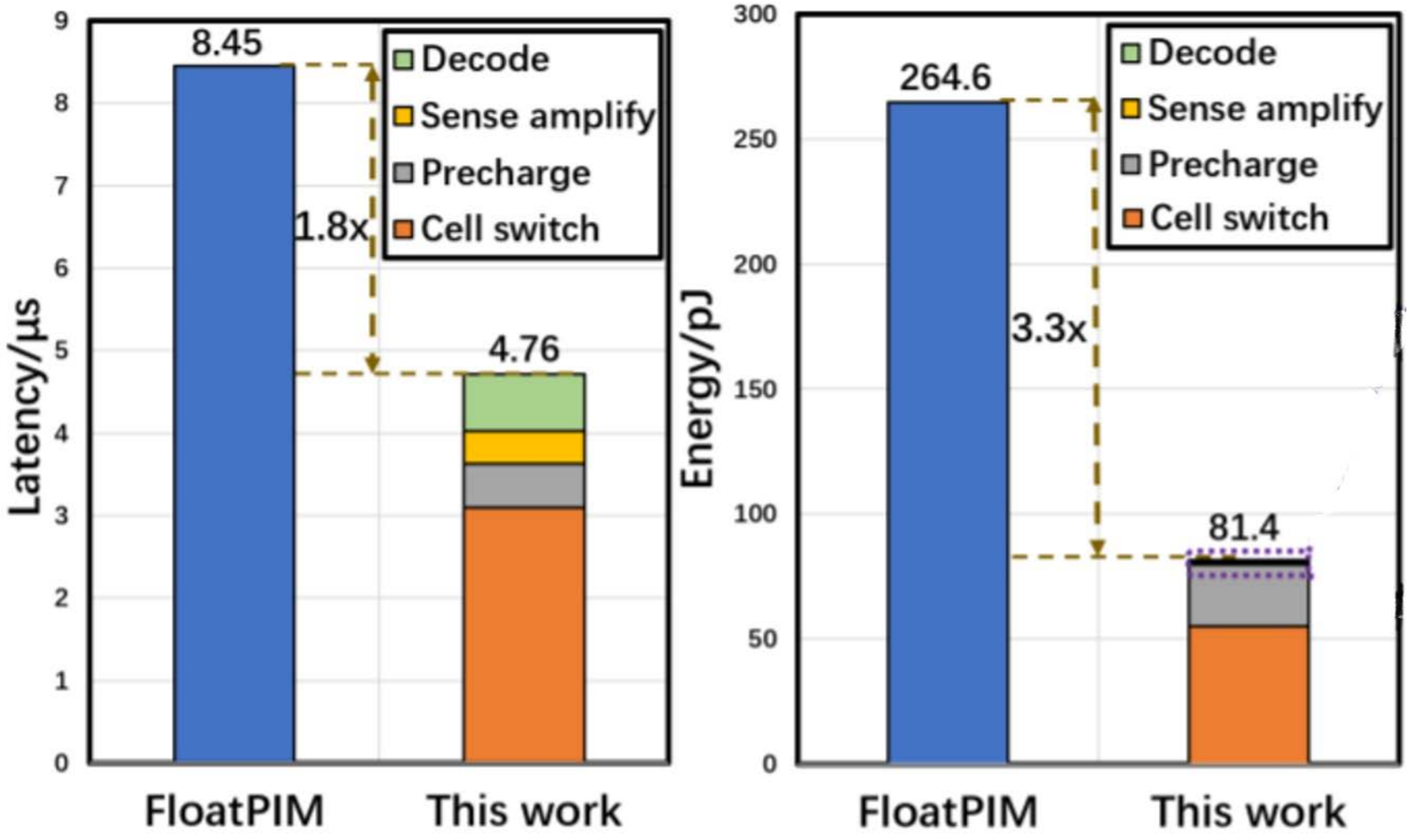


Multiplication and Accumulation Calculation (MAC) performance comparison

For **32-bit** floating point numbers:

- Cell switch dominates overhead
- Latency efficiency: **↑1.8x**
- Energy efficiency: **↑3.3x**

Evaluation Results on MAC



Multiplication and Accumulation Calculation (MAC) performance comparison

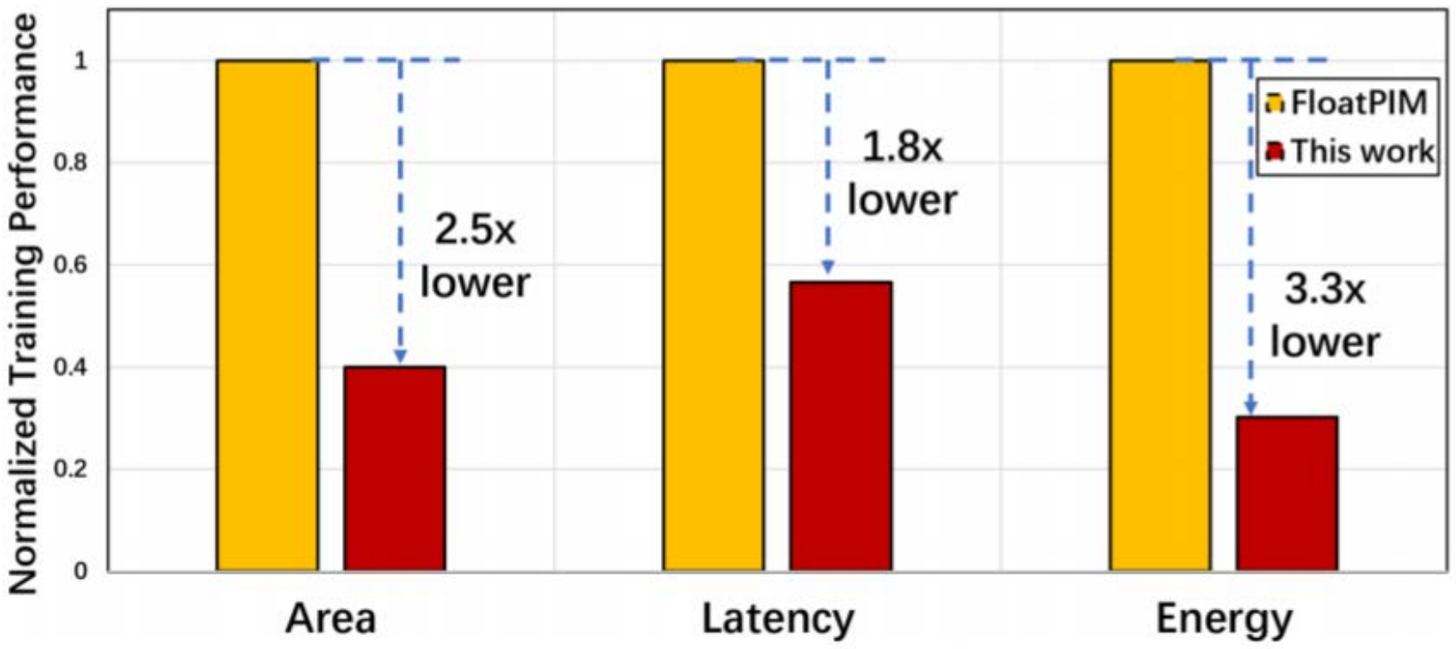
Benefits from:

- **Fewer clock cycles** required due to improved full addition and floating point computation design
- **Low write current** due to SOT-MRAM based memory cell design

For **32-bit** floating point numbers:

- Cell switch dominates overhead
- Latency efficiency: **↑1.8x**
- Energy efficiency: **↑3.3x**

Evaluation Results on Neural Network Training



Benefits from:

- **Fewer extra cells required** to store intermediate results due to improved full addition and floating point computation design
- **High flexibility** memory assignment to **maximize cells reuse** due to SOT-MRAM based 1T-1R cell design
- The improvement of MAC

LeNet-5 @ MNIST

- Area efficiency: **↑2.5x**
- Latency efficiency: **↑1.8x**
- Energy efficiency: **↑3.3x**

Outline

- **Background and Motivation**
- ***The Proposed Accelerator:* Overview**
- ***The Proposed Accelerator:* Computing Methodology**
- **Evaluation of *the Proposed Accelerator***
- **Conclusion**

Conclusion

- ***The Proposed Accelerator*:** A new SOT-MRAM based Process In-Memory accelerator for efficient training with floating point precision
  - New memory cell design which features **an improved balance** between computation flexibility and memory density
  - New full addition design that **improves computation efficiency**
  - New floating point computation design for **efficient training**
- Achieve **3.3×, 1.8×, and 2.5×** improvement in terms of energy, latency, and area, respectively, compared with a state-of-the-art PIM based DNN training accelerator

MRAM based PIM Accelerator for NN Training

*Hongjie Wang**, *Yang Zhao**, *Chaojian Li*, *Yue Wang*, and ***Yingyan Lin***

Questions ?

*the first two authors contribute equally