STAMP: Accelerating Second-order DNN Training Via ReRAM-based Processing-in-Memory Architecture

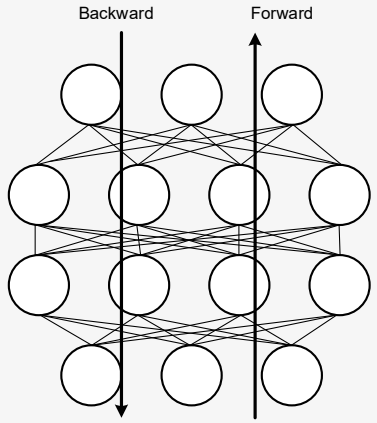
Yilong Zhao (Speaker), Fangxin Liu*, Mingyu Gao, Xiaoyao Liang, Qidong Tang, Chengyang Gu, Tao Yang, Naifeng Jing, and Li Jiang*

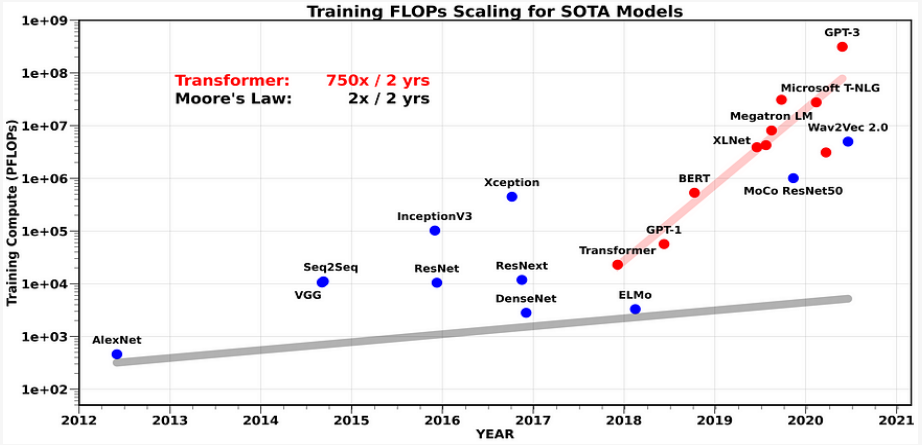
Shanghai Jiao Tong University, Shanghai Qi Zhi Institute, Tsinghua University

June 4, 2025

Background

- Training DNN models requires significant computation.





Gholami, et. al. *AI and Memory Wall*

- Current approaches to alleviate training burden.

e.g.

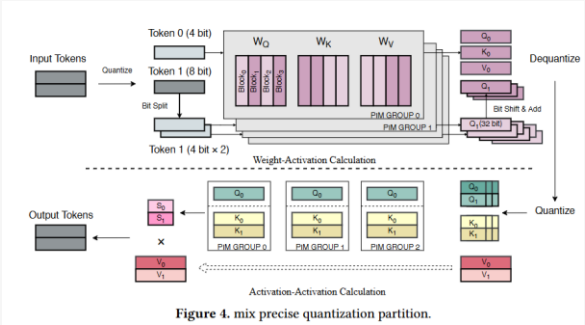
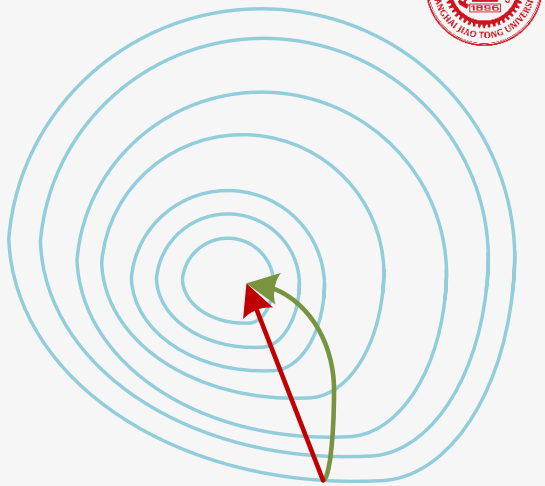


Figure 4. mix precise quantization partition.

Quantization Methods.
(Introduce quantize/dequantize overhead.)

Background – Second-order Training

1st-order training $\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$



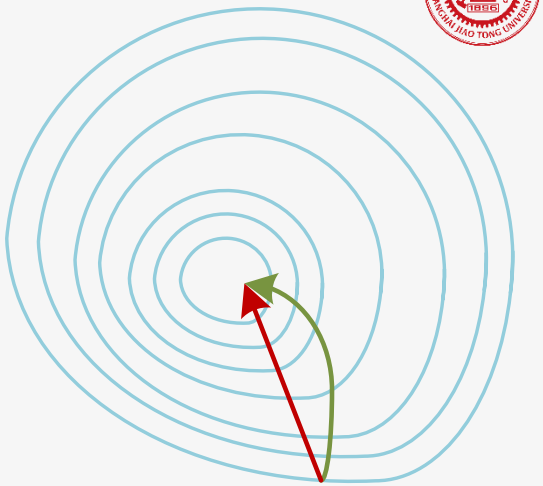
2nd-order training $\theta = \theta - \eta \cdot A^{-1} \nabla_\theta J(\theta) G^{-1}$

Utilize the **Second-order Information (SOI) Matrix's Inversion** for a more accurate step and direction
-> faster convergence

1st-order training ↖
vs
2nd-order training ↖
FEWER STEPS!

Background – Second-order Training

1st-order training $\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$



2nd-order training $\theta = \theta - \eta \cdot A^{-1} \nabla_\theta J(\theta) G^{-1}$

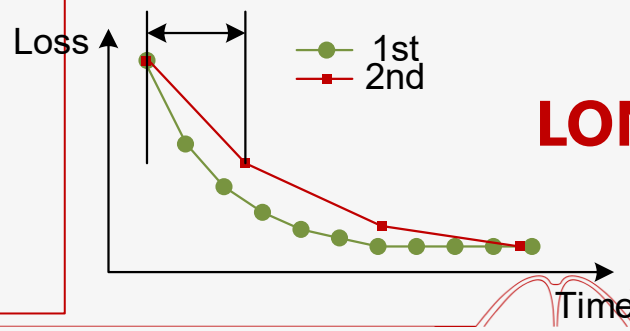
Utilize the **Second-order Information (SOI) Matrix's Inversion** for a more accurate step and direction
-> faster convergence

Why it is not widely used?

SOI matrix brings too much computation/storage overhead on **GPUs**,
$O(n^2)$ storage, $O(n^3)$ computing complexity,
-> longer step time

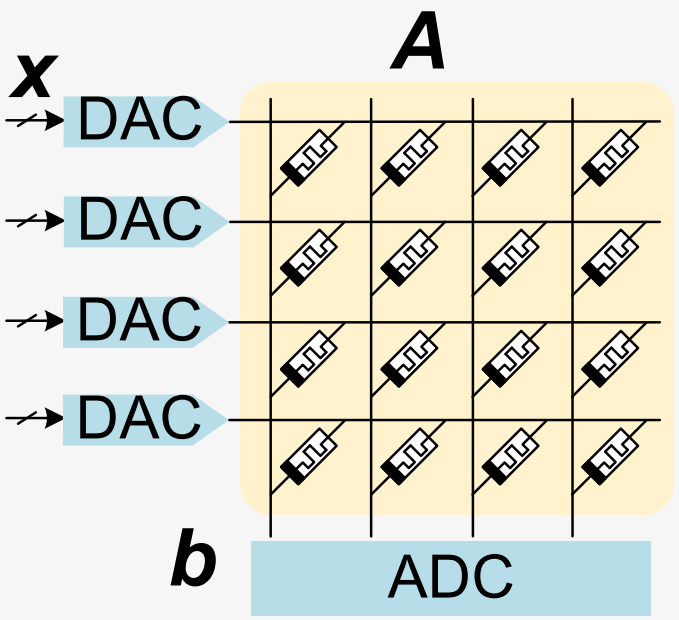
1st-order training ↖
vs
2nd-order training ↖

FEWER STEPS!



BUT LONGER STEP TIME!

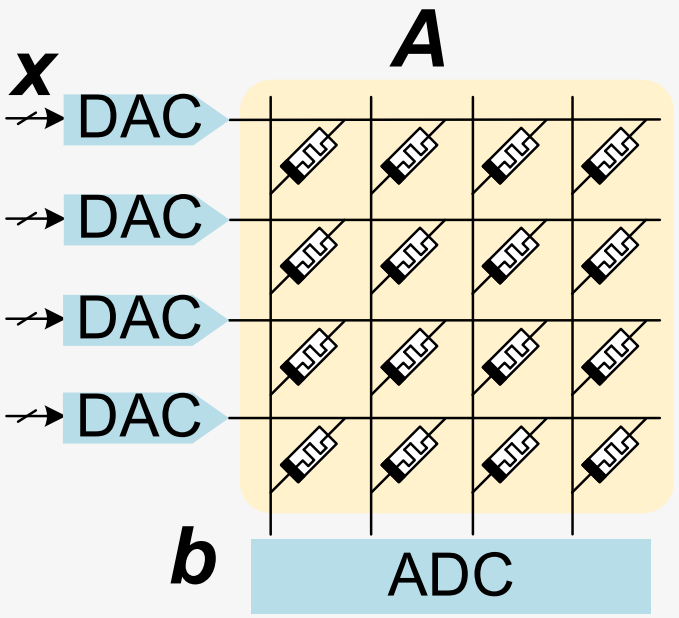
Background – ReRAM Accelerator



$$b = x \cdot A$$

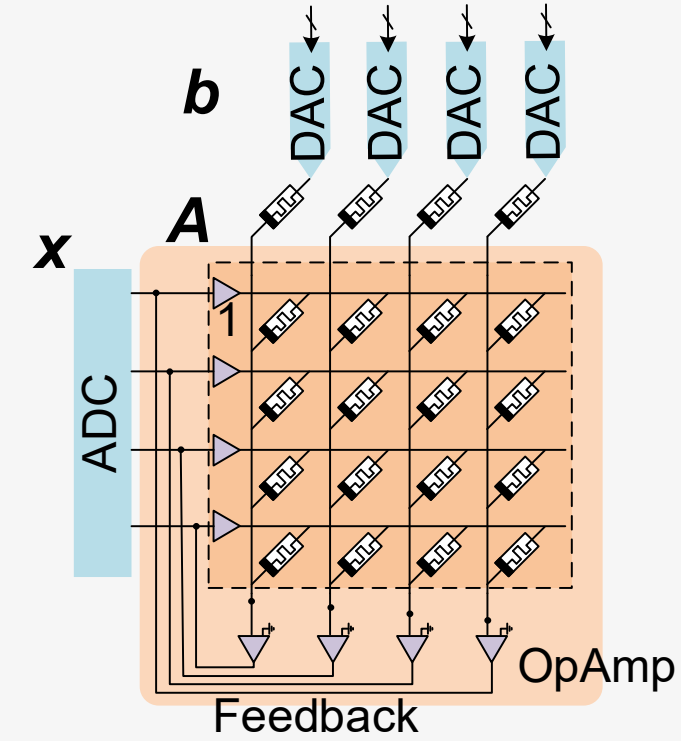
ReRAM-based
Vector-Matrix Multiplication (**VMM**)

Background – ReRAM Accelerator



$$b = x \cdot A$$

ReRAM-based
Vector-Matrix Multiplication (**VMM**)



$$x = b \cdot A^{-1}$$

ReRAM-based
Matrix Inversion (**INV**)

Both O(1) Time

Motivation

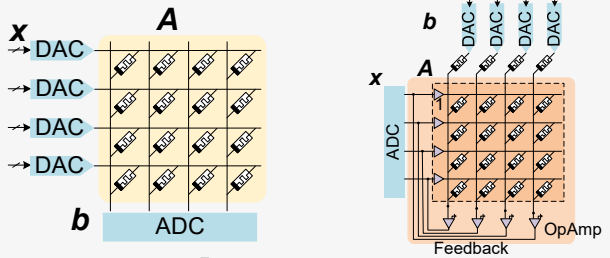
上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

Use **ReRAM+2nd-order training** to accelerate DNN training!

2nd-order training

$$\theta = \theta - \eta \cdot A^{-1} \nabla_{\theta} J(\theta) G^{-1}$$

ReRAM-based Accelerator

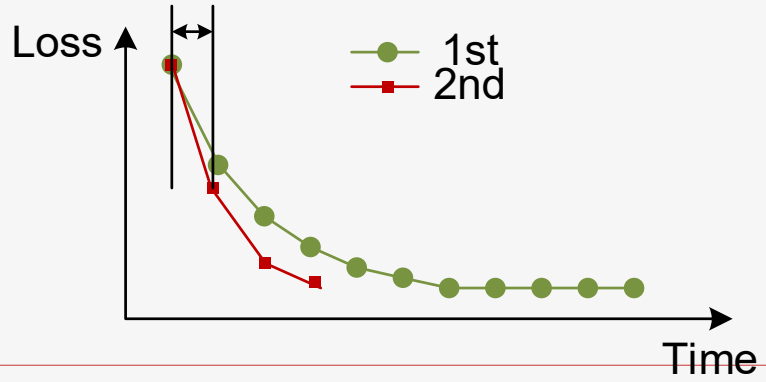


SOI Matrix Overhead

$O(n^2)$ storage overhead → Process in Memory

$O(n^3)$ computing complexity → $O(1)$ Matrix Inversion Time

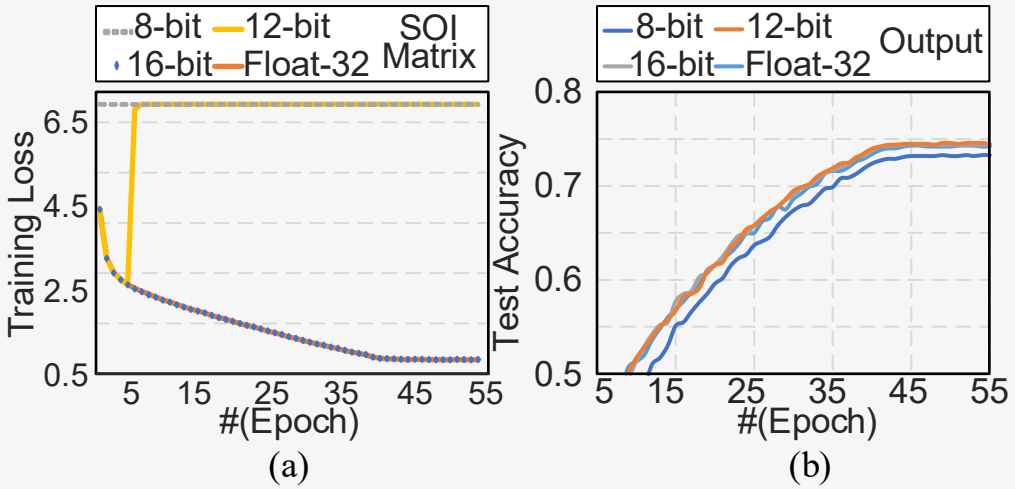
ReRAM's Feature



Aim:
Fewer Steps without apparently enlarging step time
-> **Faster training**

Problem – Circuit Precision

$$x = b \cdot A^{-1}$$



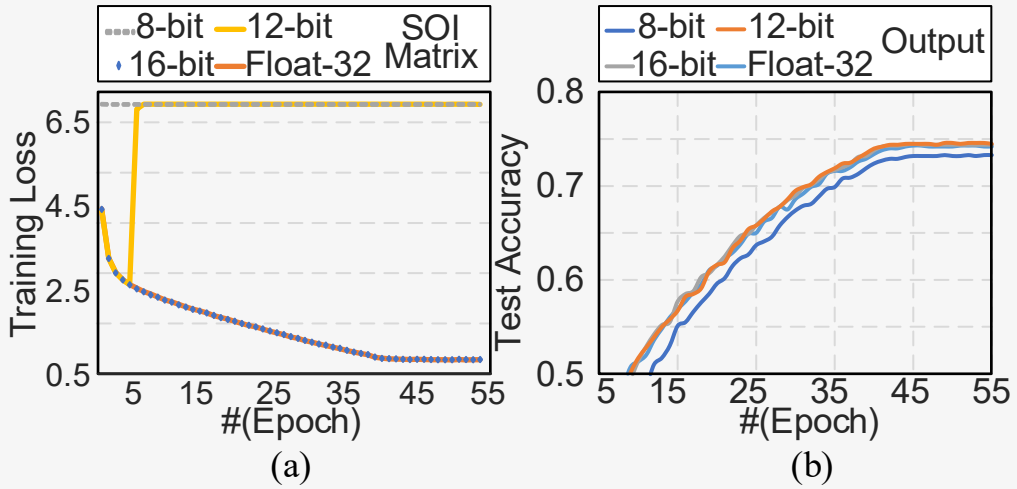
Matrix A:
At least **16** bits

Output x:
At least **12** bits

2nd-order training requires
HIGH-precision Matrix Inversion

Problem – Circuit Precision

$$x = b \cdot A^{-1}$$



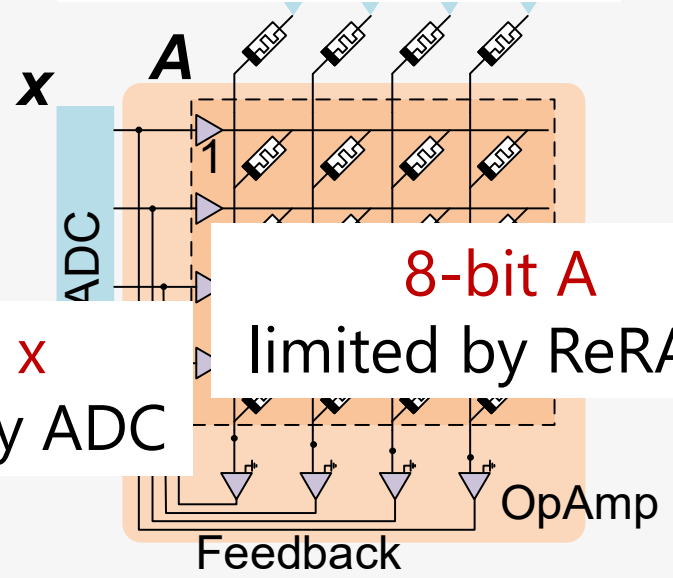
Matrix A:
At least **16** bits

Output x:
At least **12** bits

2nd-order training requires
HIGH-precision Matrix Inversion

HPCA'21

6-bit b
Limited by DAC

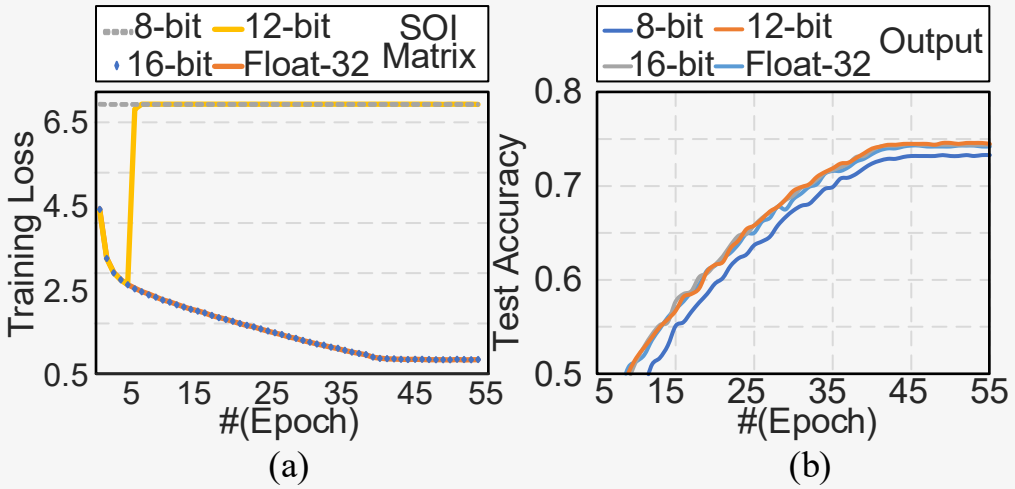


8-bit A
limited by ReRAM

8-bit x
limited by ADC

Problem – Circuit Precision

$$x = b \cdot A^{-1}$$



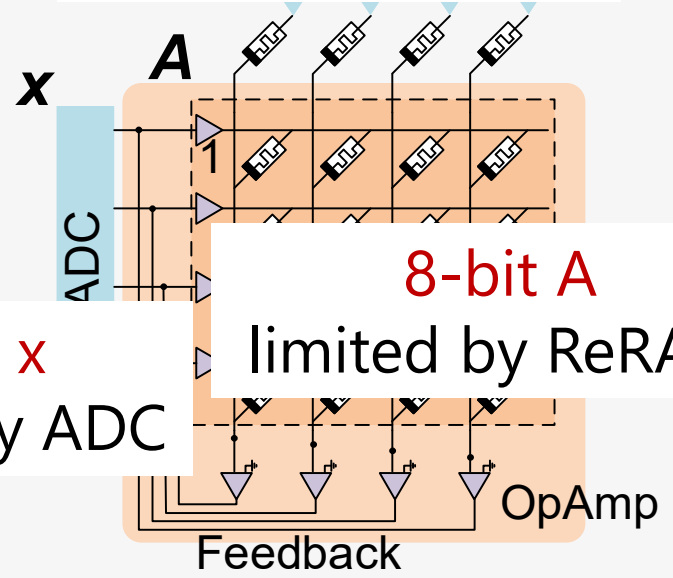
Matrix A:
At least **16** bits

Output x:
At least **12** bits

2nd-order training requires
HIGH-precision Matrix Inversion

HPCA'21

6-bit b
Limited by DAC



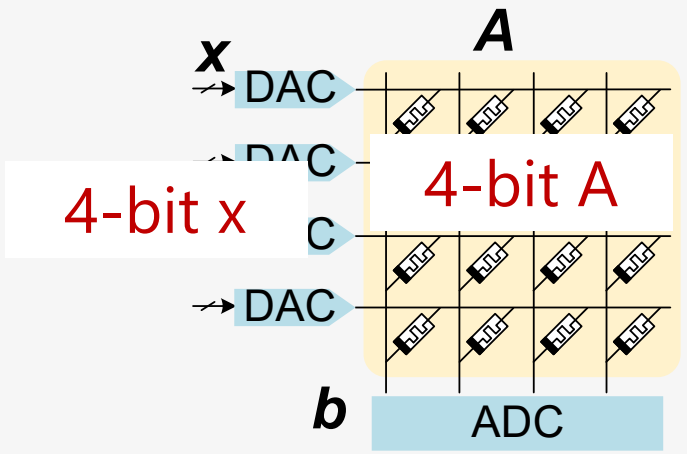
8-bit A
limited by ReRAM

8-bit x
limited by ADC

ReRAM-based Matrix Inversion's precision is **not enough**

Problem – Circuit Precision

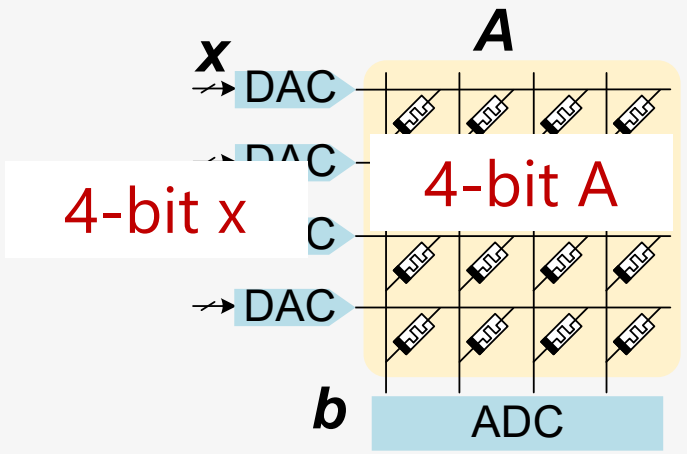
- How this problem solved in ReRAM-based Vector-Matrix Multiplication



We need 8-bit x, 8-bit A?
HOW?

Problem – Circuit Precision

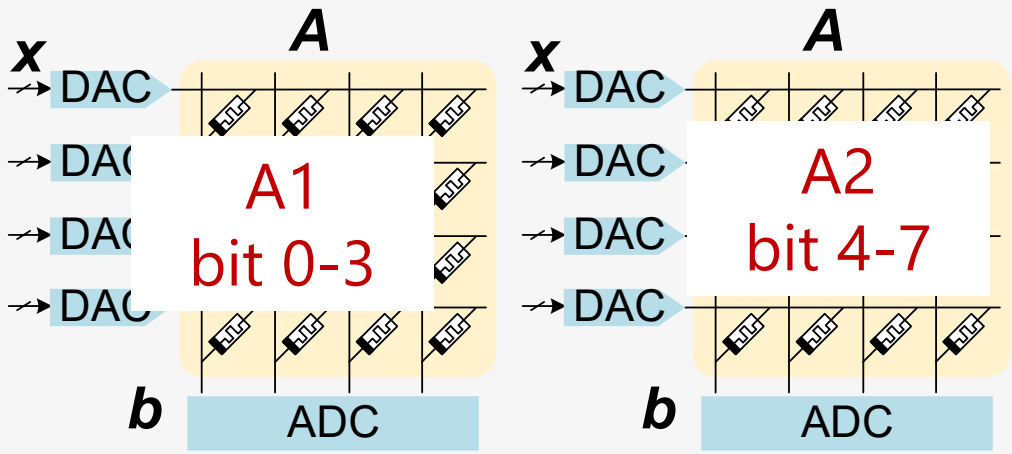
- How this problem solved in ReRAM-based Vector-Matrix Multiplication



We need 8-bit x, 8-bit A?
HOW?

cycle 0
x1 bit 0-3

cycle 1
x2 bit 4-7

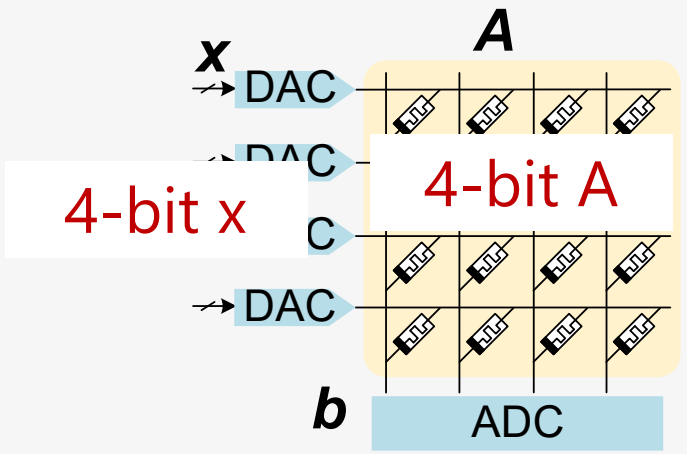


$$b = \begin{aligned} & x1 \cdot A1 \cdot 2^{-8} + \\ & x1 \cdot A2 \cdot 2^{-4} + \\ & x2 \cdot A1 \cdot 2^{-4} + \\ & x2 \cdot A2 \end{aligned}$$

Problem – Circuit Precision

上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

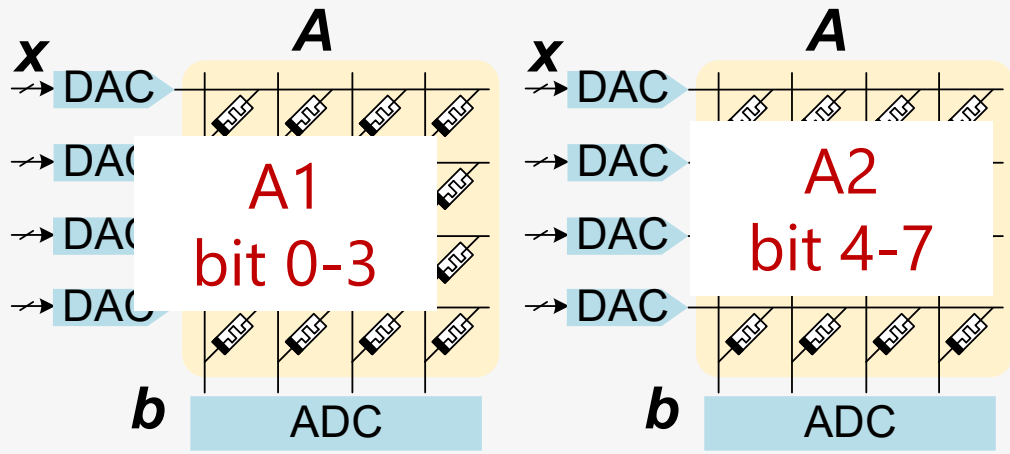
- How this problem solved in ReRAM-based Vector-Matrix Multiplication



We need 8-bit x, 8-bit A?
HOW?

cycle 0
x1 bit 0-3

cycle 1
x2 bit 4-7



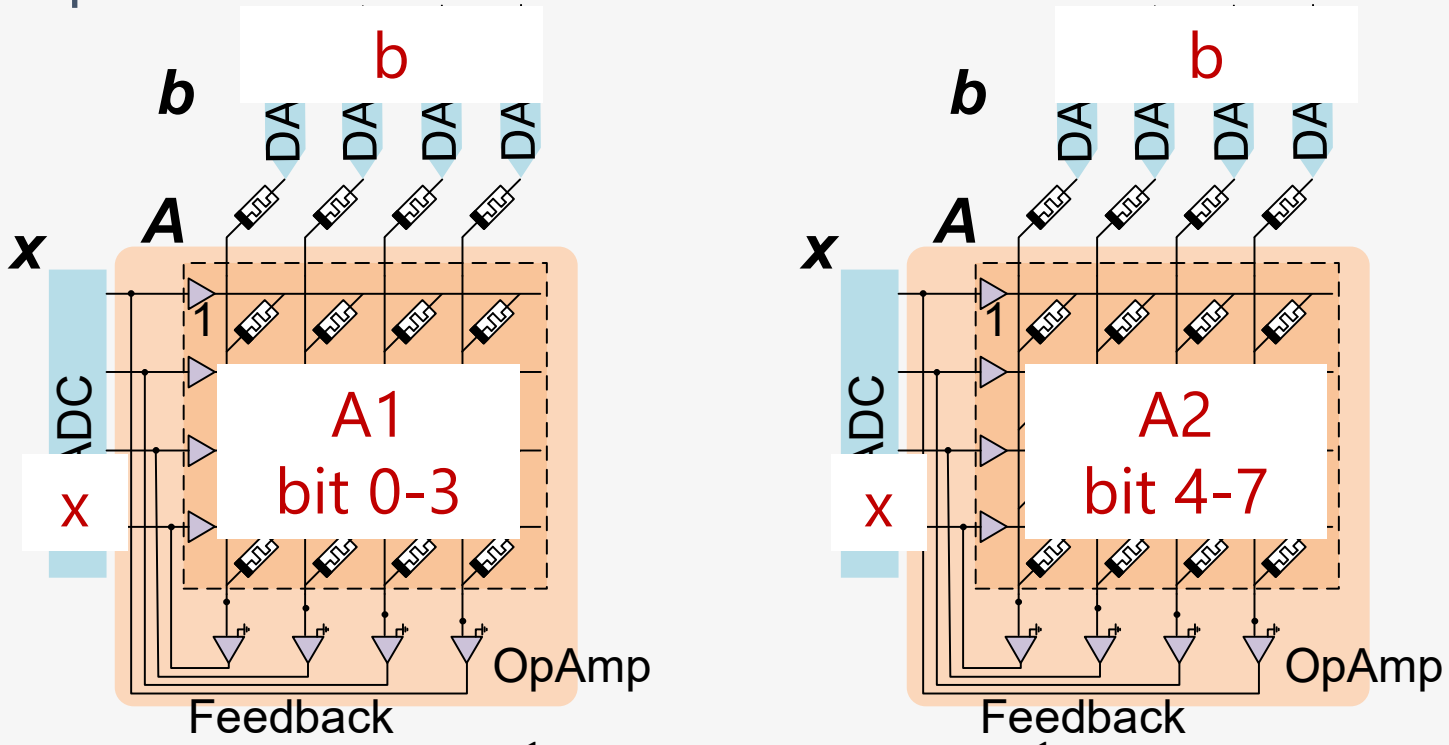
$$b = x1 \cdot A1 \cdot 2^{-8} + x1 \cdot A2 \cdot 2^{-4} + x2 \cdot A1 \cdot 2^{-4} + x2 \cdot A2$$

Traditional **BIT SLICE SCHEME**

SJTU

Problem – Circuit Precision

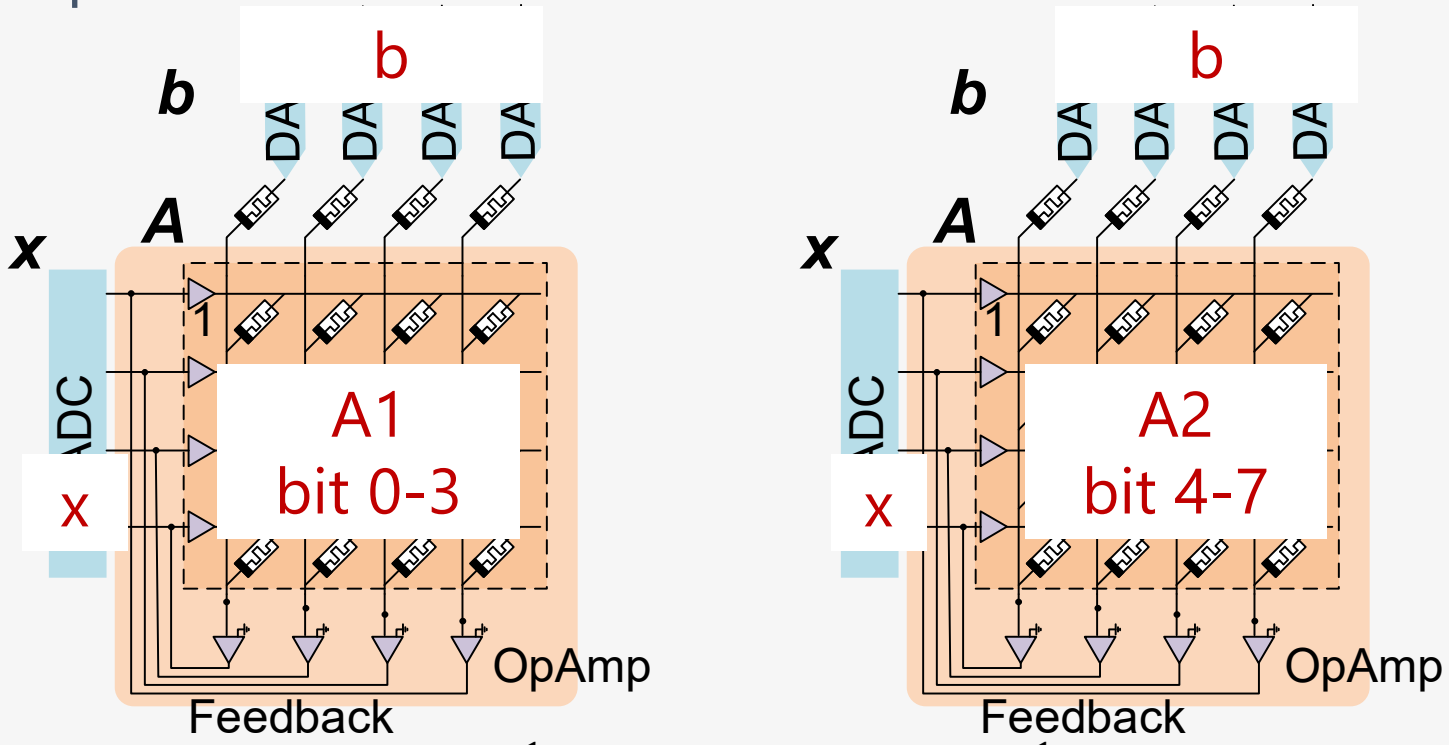
- It is more complicated in ReRAM-based Matrix Inversion



$$b \cdot A^{-1} \neq b \cdot (A1 + A2)^{-1}$$

Problem – Circuit Precision

- It is more complicated in ReRAM-based Matrix Inversion



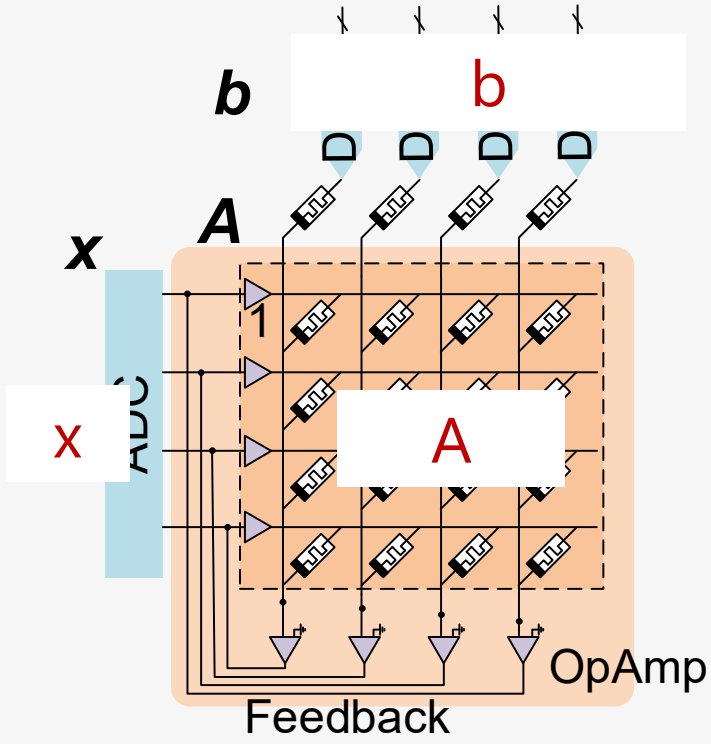
$$b \cdot A^{-1} \neq b \cdot (A1 + A2)^{-1}$$

For Matrix Inversion,
Traditional Bit Slice Scheme Doesn't Work!

Because Matrix Inversion doesn't have distributive law

High-Precision Matrix Inversion Algorithm

- 2 Levels: x, A

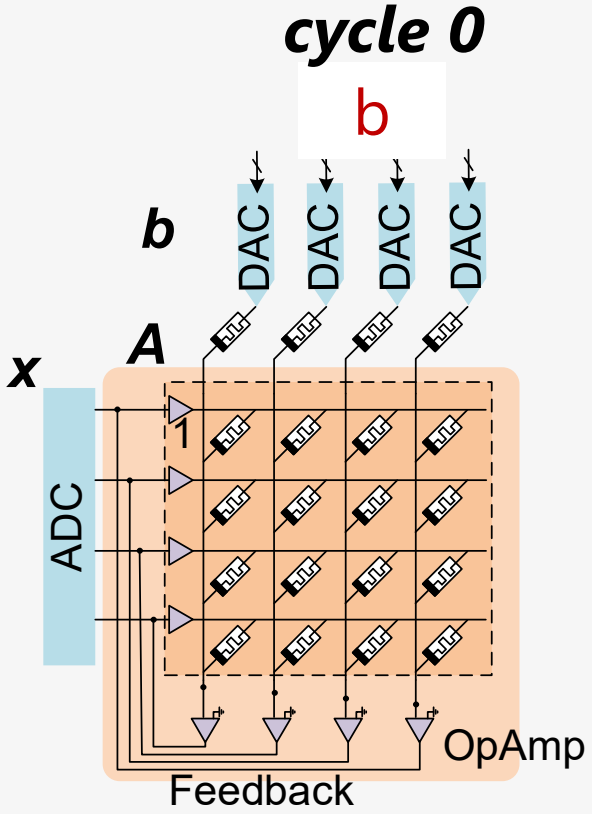


High-Precision Matrix Inversion Algorithm

上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

- 2 Levels: x, A

Level x
16-bit
->2*8 bits



High-Precision Matrix Inversion Algorithm

2 Levels: x, A

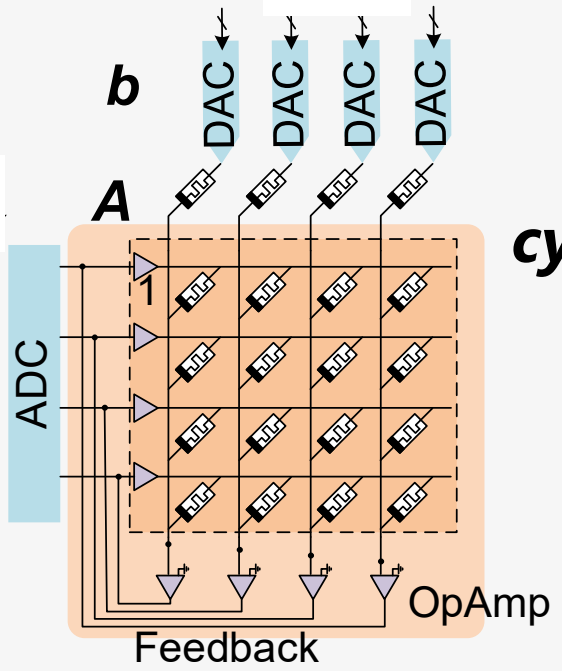
Level x
16-bit
->2*8 bits

cycle 0

x1 8-bits

cycle 0

b



cycle 0 $x1 = ADC_8(b \cdot A^{-1})$

ADC quantize **high 8 bits** of x

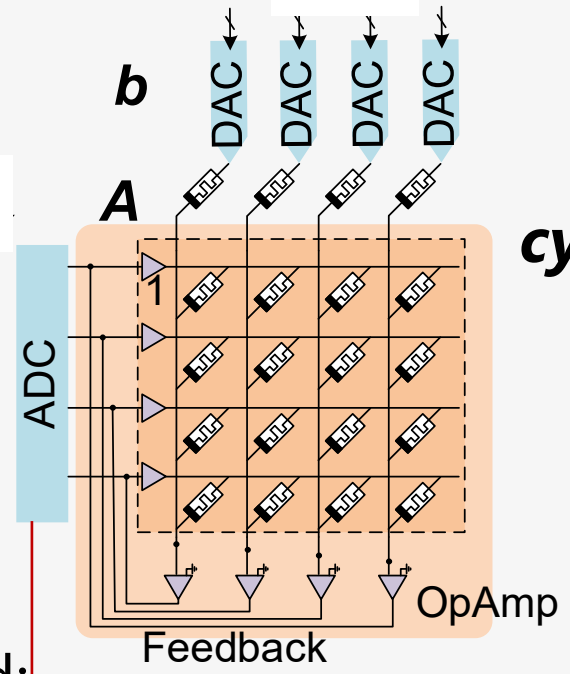
High-Precision Matrix Inversion Algorithm

2 Levels: x, A

Level x
16-bit
->2*8 bits

cycle 0
x1 8-bits

cycle 0
b



cycle 0 $x1 = ADC_8(b \cdot A^{-1})$

ADC quantize **high 8 bits** of x

In Analog Field:

$$x1 + x2 = b \cdot A^{-1}$$

quantized omitted (lower part)

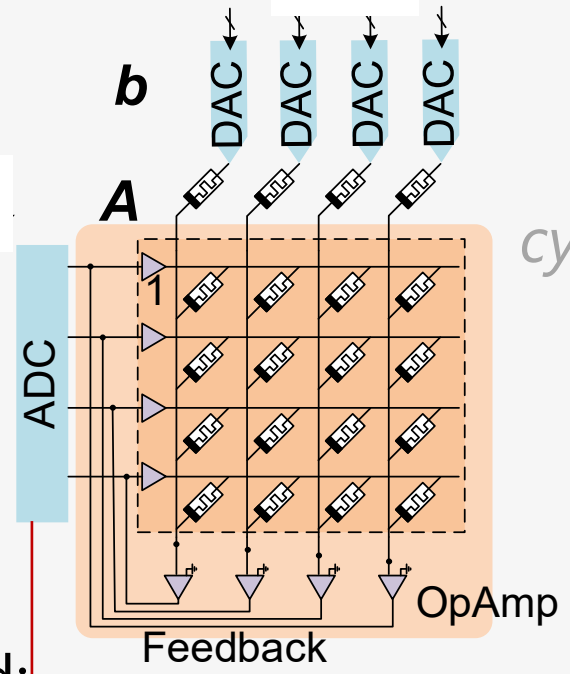
High-Precision Matrix Inversion Algorithm

2 Levels: x, A

Level x
16-bit
->2*8 bits

cycle 0
b

cycle 0
x1 8-bits



$cycle\ 0\ \ x1 = ADC_8(b \cdot A^{-1})$

ADC quantize **high 8 bits** of x

In Analog Field:

$x1 + x2 = b \cdot A^{-1} \longrightarrow x2 = b \cdot A^{-1} - x1$

quantized omitted

Remove x1 part from b, x2 becomes high bits

High-Precision Matrix Inversion Algorithm

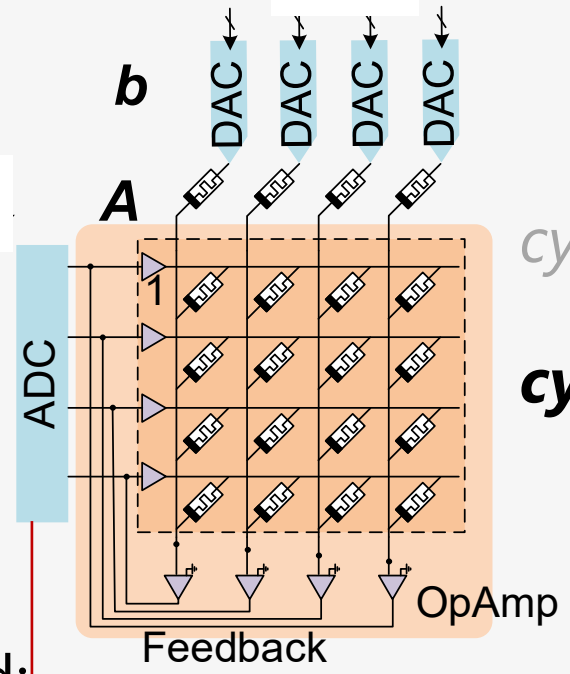
2 Levels: x, A

Level x
16-bit
->2*8 bits

cycle 0
b

cycle 1
b′

cycle 0
x1 8-bits



cycle 0 $x1 = ADC_8(b \cdot A^{-1})$

ADC quantize **high 8 bits** of x

cycle 1 $b' = b - A \cdot x1$

In Analog Field:

$x1 + x2 = b \cdot A^{-1} \rightarrow x2 = b \cdot A^{-1} - x1$

quantized omitted

Remove x1 part from b, x2 becomes high bits

High-Precision Matrix Inversion Algorithm

2 Levels: x, A

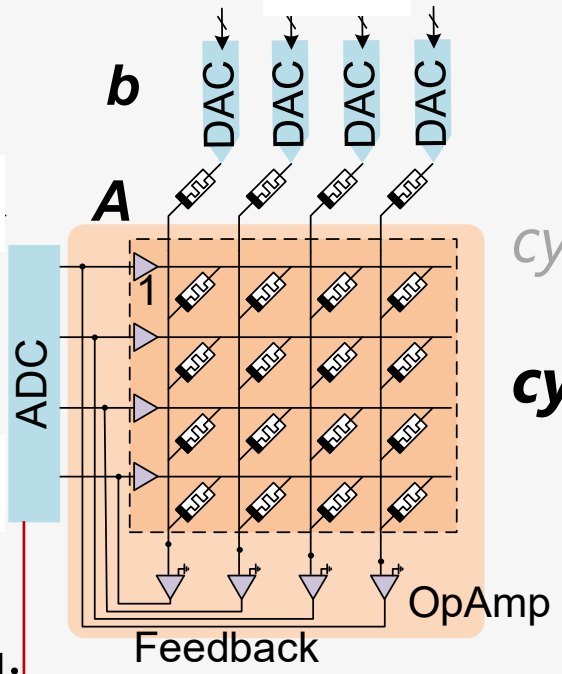
Level x
16-bit
->2*8 bits

cycle 0
b

cycle 1
b′

cycle 0
x1 8-bits

cycle 1
x2 8-bits



cycle 0 $x1 = ADC_8(b \cdot A^{-1})$

ADC quantize **high 8 bits** of x

cycle 1 $b' = b - A \cdot x1$

$x2 = ADC_8(b' \cdot A^{-1})$

Get **lower 8 bits**

In Analog Field:

$x1 + x2 = b \cdot A^{-1} \longrightarrow x2 = b \cdot A^{-1} - x1$

quantized omitted

Remove x1 part from b,
x2 becomes high bits

High-Precision Matrix Inversion Algorithm

2 Levels: x, A

Taylor Expansion:

Level A
16-bit
->2*8 bits

8 bit A_H &
8 bit A_L

High-Precision Matrix Inversion Algorithm

2 Levels: x, A

Taylor Expansion:

$$A^{-1} \cdot b = (A_H + A_L)^{-1} \cdot b$$
$$= A_H^{-1} \cdot (I - P + P^2 - P^3 + \cdots) \cdot b$$
$$(P = A_H^{-1} \cdot A_L)$$

Level A
16-bit
->2*8 bits

8 bit A_H &
8 bit A_L

High-Precision Matrix Inversion Algorithm

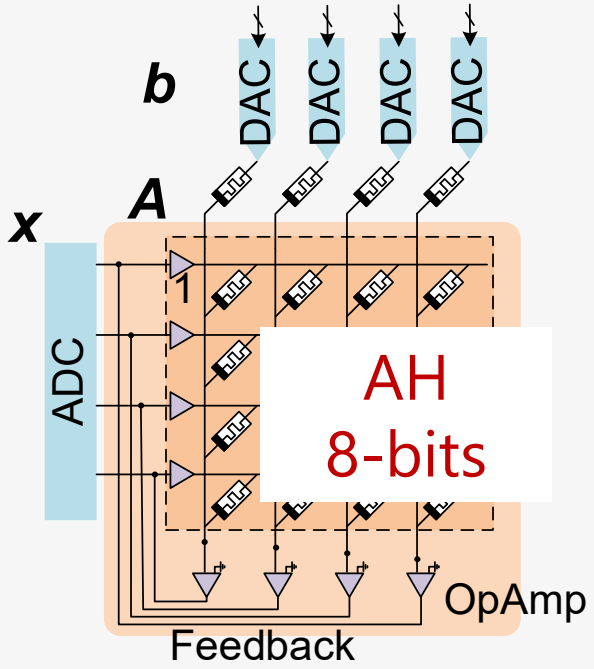
- 2 Levels: x, A

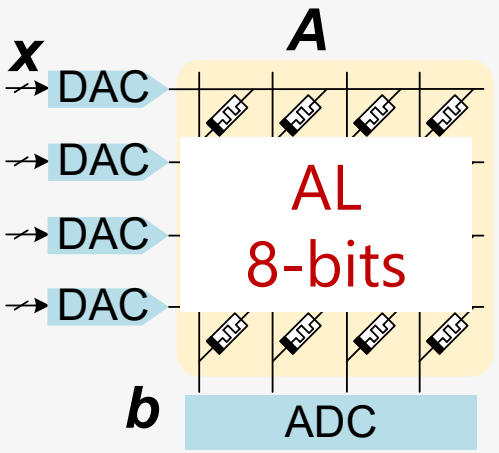
Taylor Expansion:

$$A^{-1} \cdot b = (A_H + A_L)^{-1} \cdot b$$
$$= A_H^{-1} \cdot (I - P + P^2 - P^3 + \cdots) \cdot b$$
$$(P = A_H^{-1} \cdot A_L)$$

Level A
16-bit
->2*8 bits

8 bit A_H &
8 bit A_L





init: $r = A_H^{-1} \cdot b, x = r$

High-Precision Matrix Inversion Algorithm

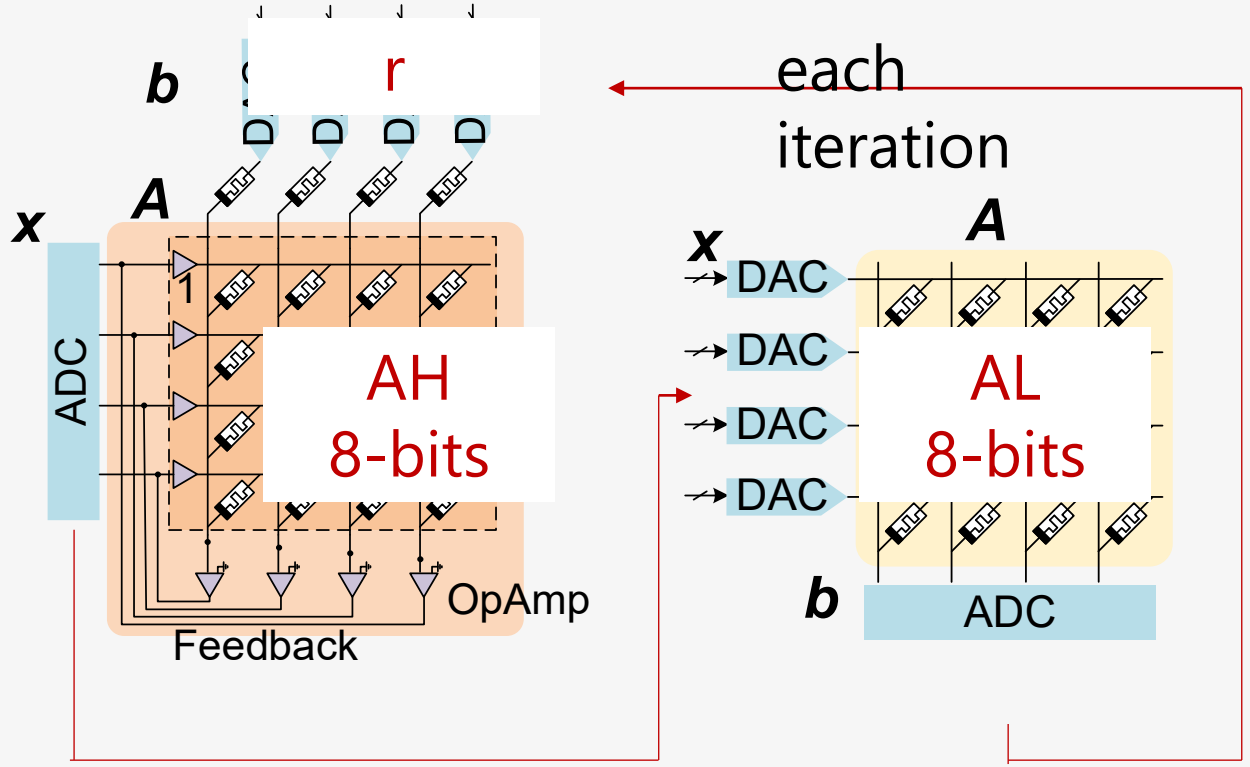
2 Levels: x, A

Taylor Expansion:

$$A^{-1} \cdot b = (A_H + A_L)^{-1} \cdot b$$
$$= A_H^{-1} \cdot (I - P + P^2 - P^3 + \cdots) \cdot b$$
$$(P = A_H^{-1} \cdot A_L)$$

Level A
16-bit
->2*8 bits

8 bit A_H &
8 bit A_L



calculate one term in each iteration

init: $r = A_H^{-1} \cdot b, x = r$

Iteration 0:

P $r = A_H^{-1} A_L r$

$x \mathrel{-}= r$

High-Precision Matrix Inversion Algorithm

上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

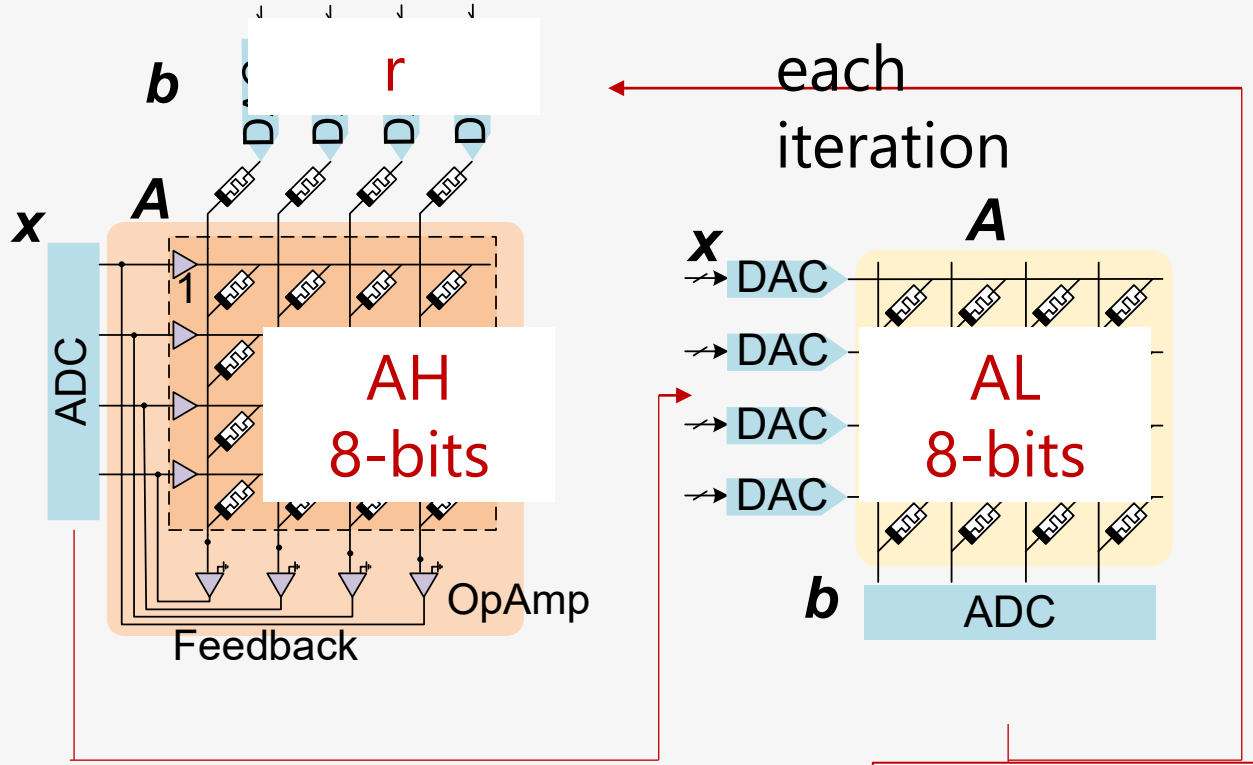
2 Levels: x, A

Taylor Expansion:

$$A^{-1} \cdot b = (A_H + A_L)^{-1} \cdot b$$
$$= A_H^{-1} \cdot (I - P + P^2 - P^3 + \cdots) \cdot b$$
$$(P = A_H^{-1} \cdot A_L)$$

Level A
16-bit
->2*8 bits

8 bit A_H &
8 bit A_L



calculate one term in each iteration

init: $r = A_H^{-1} \cdot b, x = r$

Iteration 0:

$\boldsymbol{P}$ $r = A_H^{-1} A_L r$

$x \mathrel{-}= r$

Iteration 1:

$\boldsymbol{P^2}$ $r = A_H^{-1} A_L r$

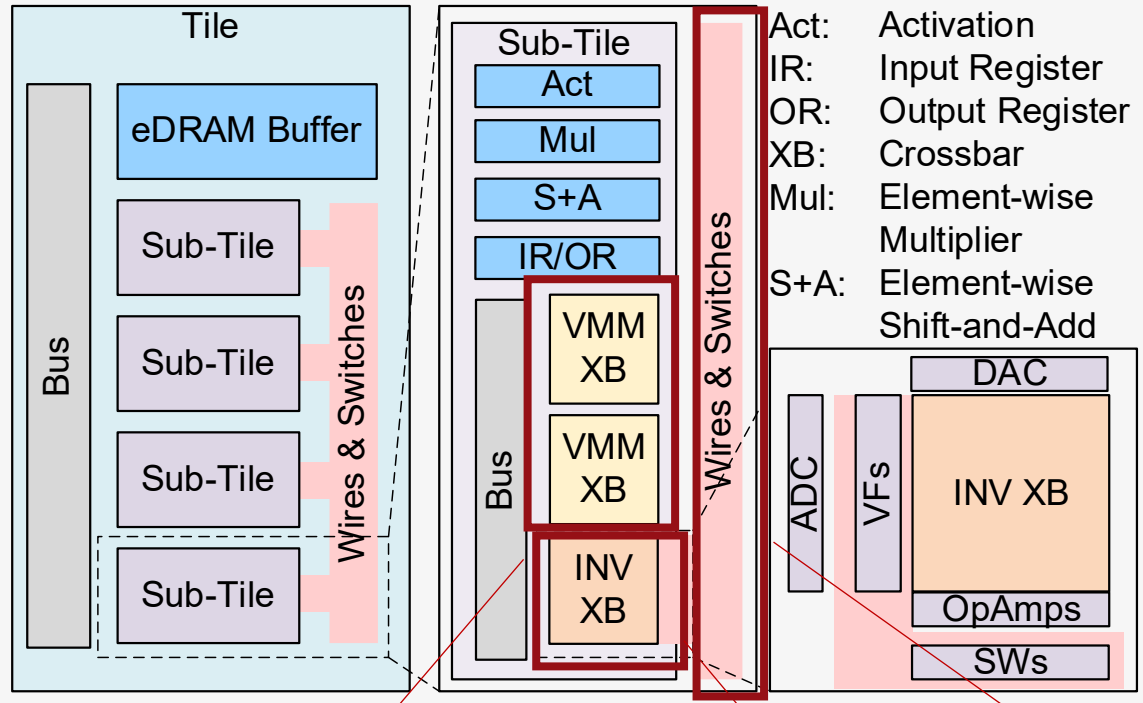
$x \mathrel{+}= r$

Iteration 2:

$\boldsymbol{P^3}$ …

x becomes more and more accurate

STAMP Architecture



For Vector-matrix multiplication

For Matrix Inversion

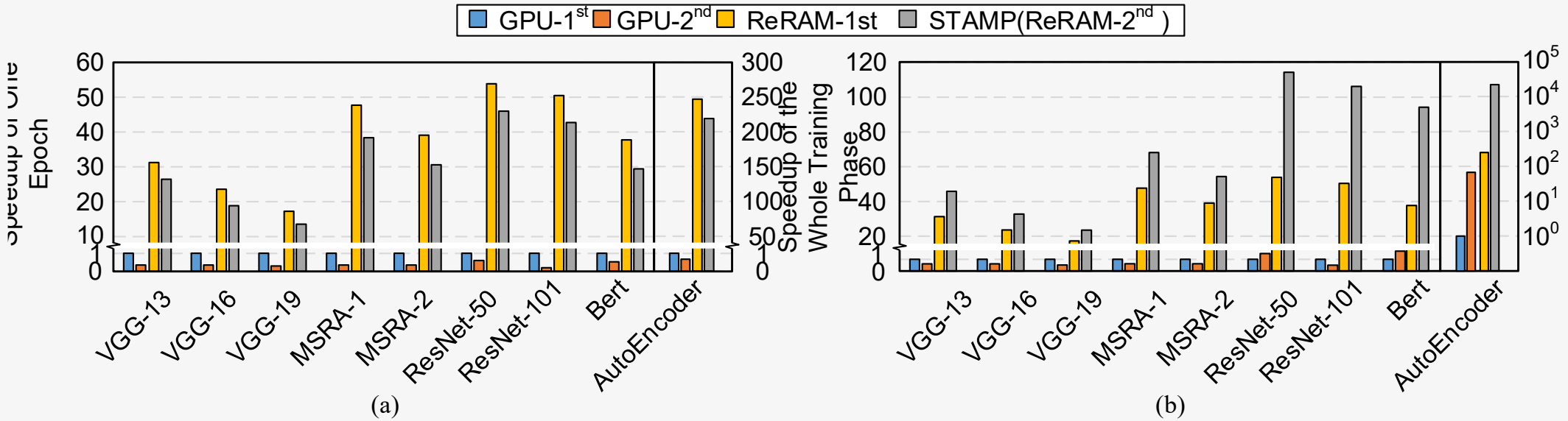
Connect INV crossbars for scaling

Evaluation

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Baselines	Tesla V100 GPU, ReRAM-based 1st-order training	
Benchmark	VGG-13/16/19, MSRA, ResNet, Bert, autoencoder	
Simulation Model	DAC ADC Hyper	ISCA'16
	OpAMP	HPCA'21
	DRAM buffer	CACTI 7
Architecture Setup	VMM:INV	16:1
	Cycle	100ns
	Crossbar size	256

Performance

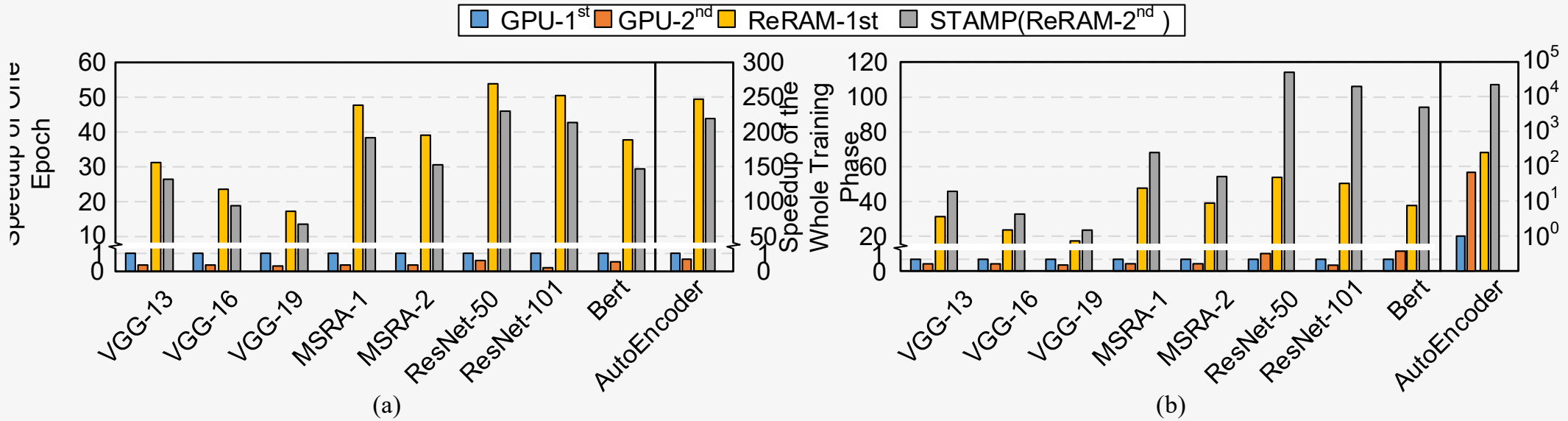


Speedup of **One Epoch** Speedup of **The Whole Training Phase**

STAMP is 68x better than GPU-1st

Compared to ReRAM-1st, Epoch +21.5%, Overall Training 11.4x Faster

Performance



Speedup of **One Epoch** Speedup of **The Whole Training Phase**

STAMP is 68x better than GPU-1st
Compared to ReRAM-1st, Epoch +21.5%, Overall Training 11.4x Faster

- On ReRAM, 2nd–order training **always better** than 1st-order training
- On GPU, 2nd–order training is worse than 1st-order training due to SOI overhead

Reduced ReRAM writing

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

- STAMP can **reduce 55.7% write** number compared to first-order training on ReRAM-based accelerator, as there are fewer epochs.

- **Enhance endurance of ReRAM Accelerator**

Summary

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

- We use 2nd-order training & ReRAM-based architecture to accelerate DNN training.

- We propose a high-precision matrix inversion algorithm based on low-precision ReRAM circuits.

上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

Thank You!

饮 水 思 源　爱 国 荣 校