Solving Least-Squares Fitting in $O(1)$ Using RRAM-based Computing-in-Memory Technique

Xiaoming Chen, Yinhe Han

Institute of Computing Technology, Chinese Academy of Sciences

University of Chinese Academy of Sciences

chenxiaoming@ict.ac.cn

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY ,CHINESE ACADEMY OF SCIENCES

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

About Me

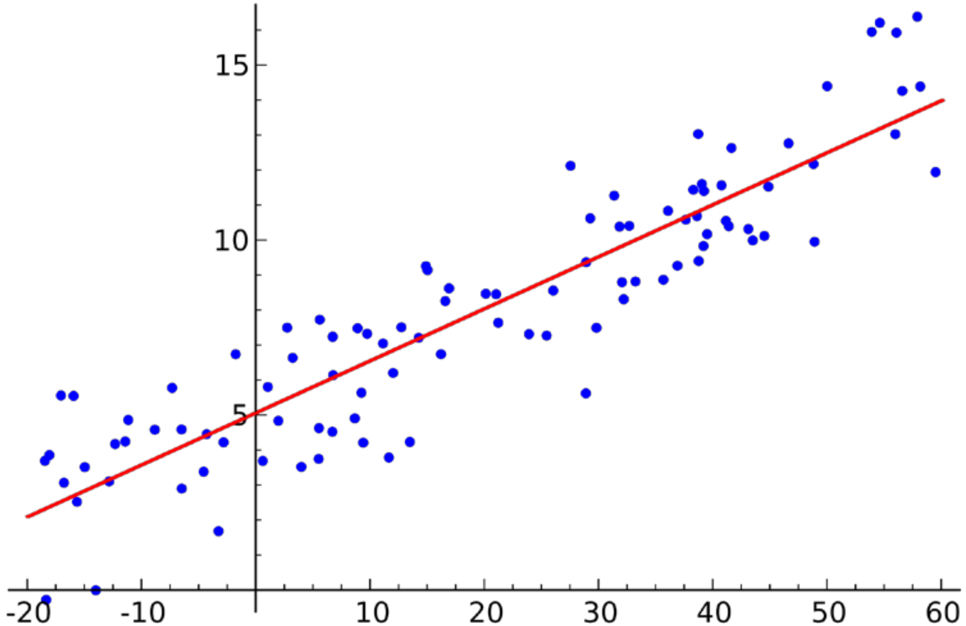
- Xiaoming Chen, Associate Professor @ Institute of Computing Technology, Chinese Academy of Sciences
- Received BS and PhD degrees in electronic engineering from Tsinghua University in 2009 and 2014, respectively
- Research interests include EDA and computer architecture; published about 100 papers in DAC, ICCAD, ASP-DAC, DATE, HPCA, IEEE TCAD, IEEE TPDS, etc.
- Recipient of 2021 NSFC Excellent Young Scientists Fund and 2015 European Design and Automation Association (EDAA) Outstanding Dissertation Award

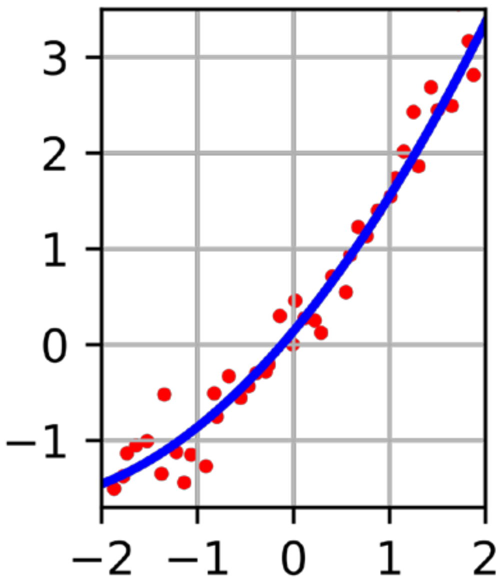
Outline

- Background
- Proposed Approach: Principle Overview
- Scalable Architecture for Large-scale Problems
- Simulation Results
- Conclusion

Least-Squares Fitting

- Standard approach in regression analysis to approximate solution of over-determined systems
- Widely used in modeling, data fitting, predictive analysis, etc.
- High time complexity ($O(N^3)$) and poor scalability for large-scale problems





Linear Regression

- $\mathbf{x}$: N-dimensional input vector
- ϕ_j: basis function of x
- Y: scalar output
- $\boldsymbol{\beta}$: unknown parameters

$$Y = \sum_{j=0}^{N-1} \beta_j \phi_j(\mathbf{x})$$

- M items of training data $(\mathbf{X}^{(0)}, Y^{(0)}), (\mathbf{X}^{(1)}, Y^{(1)}), \cdots, (\mathbf{X}^{(M-1)}, Y^{(M-1)})$ where $\mathbf{X}^{(i)} = (\phi_0^{(i)}(\mathbf{x}), \phi_1^{(i)}(\mathbf{x}), \cdots, \phi_{N-1}^{(i)}(\mathbf{x}))^T$
- If M>N (more equations than unknowns), it is an over-determined system
- $\boldsymbol{\beta}$ can be estimated by minimizing sum-of-squares error function

$$E(\boldsymbol{\beta}) = \frac{1}{2} \sum_{i=0}^{M-1} \left(Y_i - \sum_{j=0}^{N-1} \beta_j \phi_j^{(i)}(\mathbf{x}) \right)^2$$

Solution of Least-Squares Fitting

- **Analytical solution form**

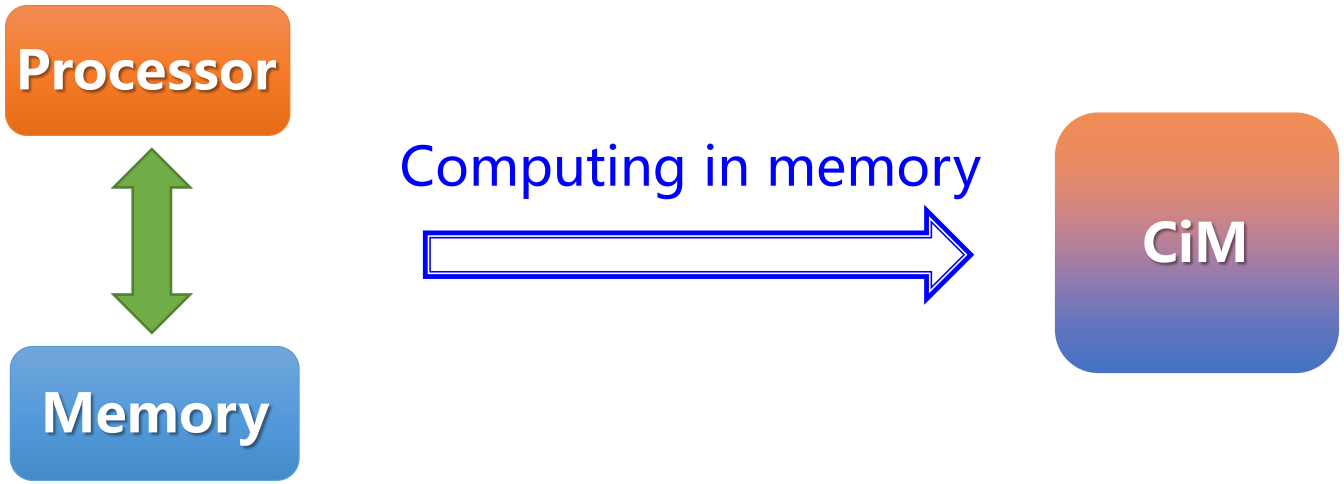
$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

- $\mathbf{X} = (\mathbf{X}^{(0)}, \mathbf{X}^{(1)}, \cdots, \mathbf{X}^{(M-1)})^T$
- $\mathbf{Y} = (Y^{(0)}, Y^{(1)}, \cdots, Y^{(M-1)})^T$

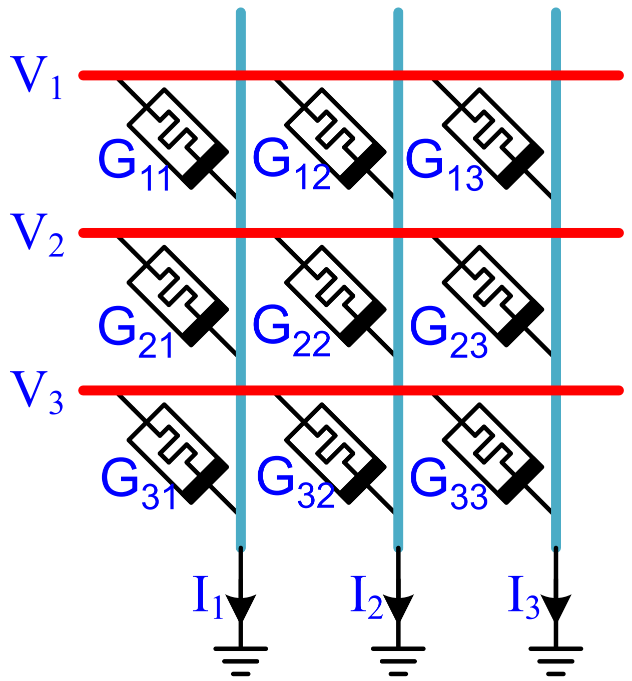
- **Matrix-matrix multiplication: O(N^3)**
- **Matrix inversion: O(N^3)**
- **Matrix-vector multiplication: O(N^2)**
- **High time complexity and not accelerator-friendly**

Computing in Memory

- CiM: promising technique to alleviate memory wall bottleneck
- **Benefits**: high bandwidth, low latency, high parallelism…
- Emerging non-volatile devices have ability of both **memory** and **switch**
  - RRAM, MTJ, FeFET, PCM…



Resistive Random-Access Memory (RRAM)



Kirchhoff's Law

$$I_1 = G_{11}V_1 + G_{21}V_2 + G_{31}V_3$$
$$I_2 = G_{12}V_1 + G_{22}V_2 + G_{32}V_3$$
$$I_3 = G_{13}V_1 + G_{23}V_2 + G_{33}V_3$$

$$\mathbf{I} = \mathbf{G}^T\mathbf{V}$$

- **An RRAM-based crossbar array can complete an analog matrix-vector multiplication in O(1) time complexity**
- **Widely used for neural network acceleration**
- **Device-level CiM: RRAMs not only store analog values (by programming the resistance), but also perform computations (via Kirchhoff's Law)**

Contributions

- RRAM-based architecture to accelerate least-squares fitting
- Software-hardware codesign: elaborate algorithm design and closed-loop feedback circuit structure to achieve O(1) time complexity for least-squares fitting
- Scalable and configurable architecture for handling large-scale least-squares fitting problems

Gradient Descent

- **Minimizing a function by a series of updates to unknown parameters, each of which takes steps proportional to the negative of the gradient of the function at the current point**

$$\boldsymbol{\beta}[t+1] = \boldsymbol{\beta}[t] - \eta \nabla E(\boldsymbol{\beta})$$

- η: **learning rate**
- **Gradient descent based iterative form of LSF solution**

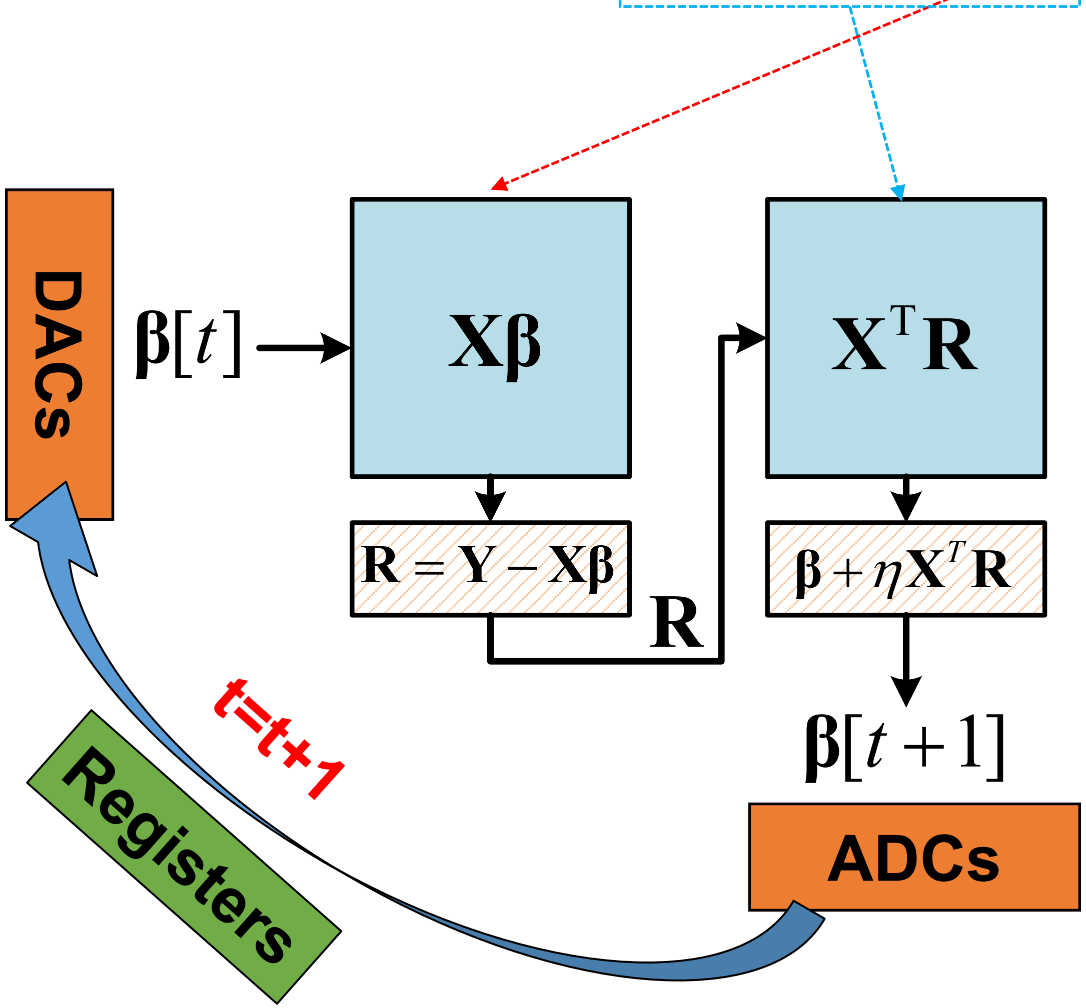
$$\beta_j[t+1] = \beta_j[t] + \eta \sum_{i=0}^{M-1} \left[\phi_j^{(i)}(\mathbf{x}) \left(Y_i - \sum_{j=0}^{N-1} \beta_j[t] \phi_j^{(i)}(\mathbf{x}) \right) \right]$$

$$\Downarrow$$

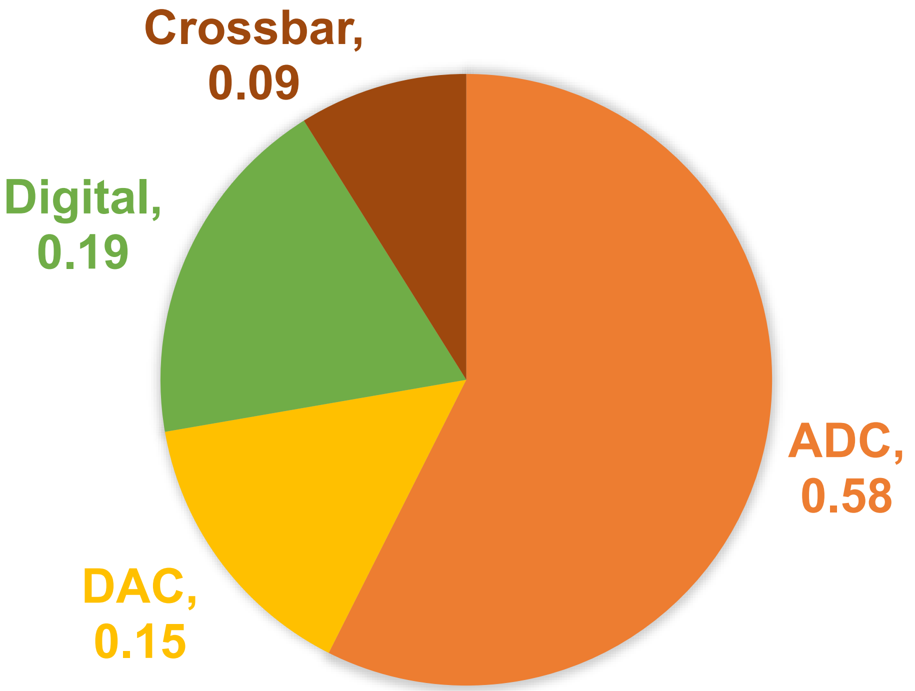
$$\boldsymbol{\beta}[t+1] = \boldsymbol{\beta}[t] + \eta \mathbf{X}^T (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}[t])$$

Direct Hardware Implementation

$$\boldsymbol{\beta}[t+1] = \boldsymbol{\beta}[t] + \eta \mathbf{X}^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}[t])$$



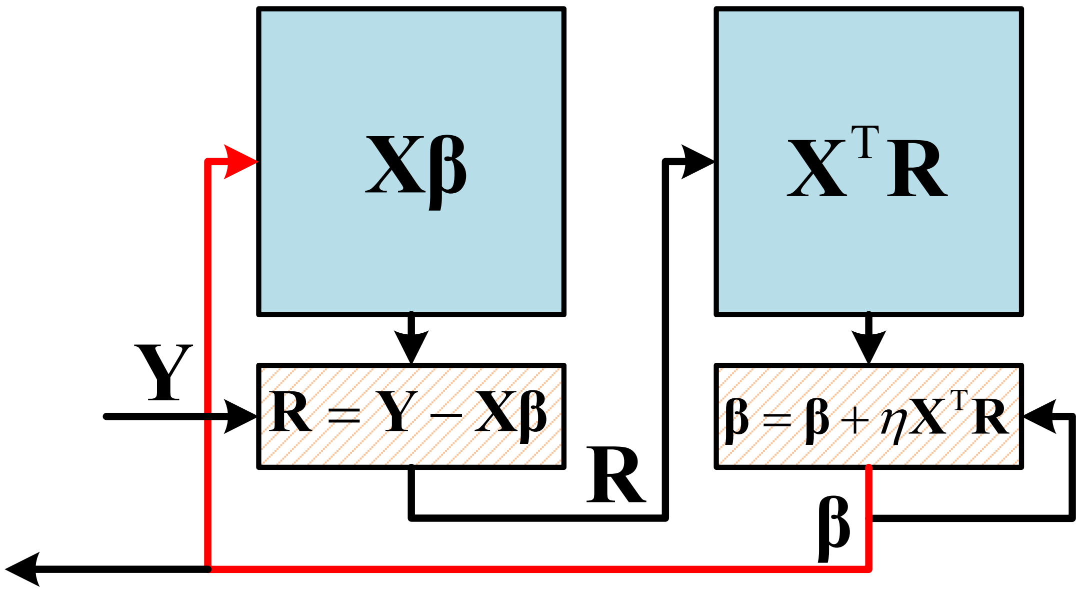
ENERGY BREAKDOWN



About 3/4 energy is consumed by ADCs & DACs

Proposed Approach

- Key principle: **connect output to input** ➔ avoid analog signal storage, as well as ADCs & DACs
- Closed-loop circuit automatically **"converges" to DC point**
- Iterations eliminated ➔ **O(1) time complexity**

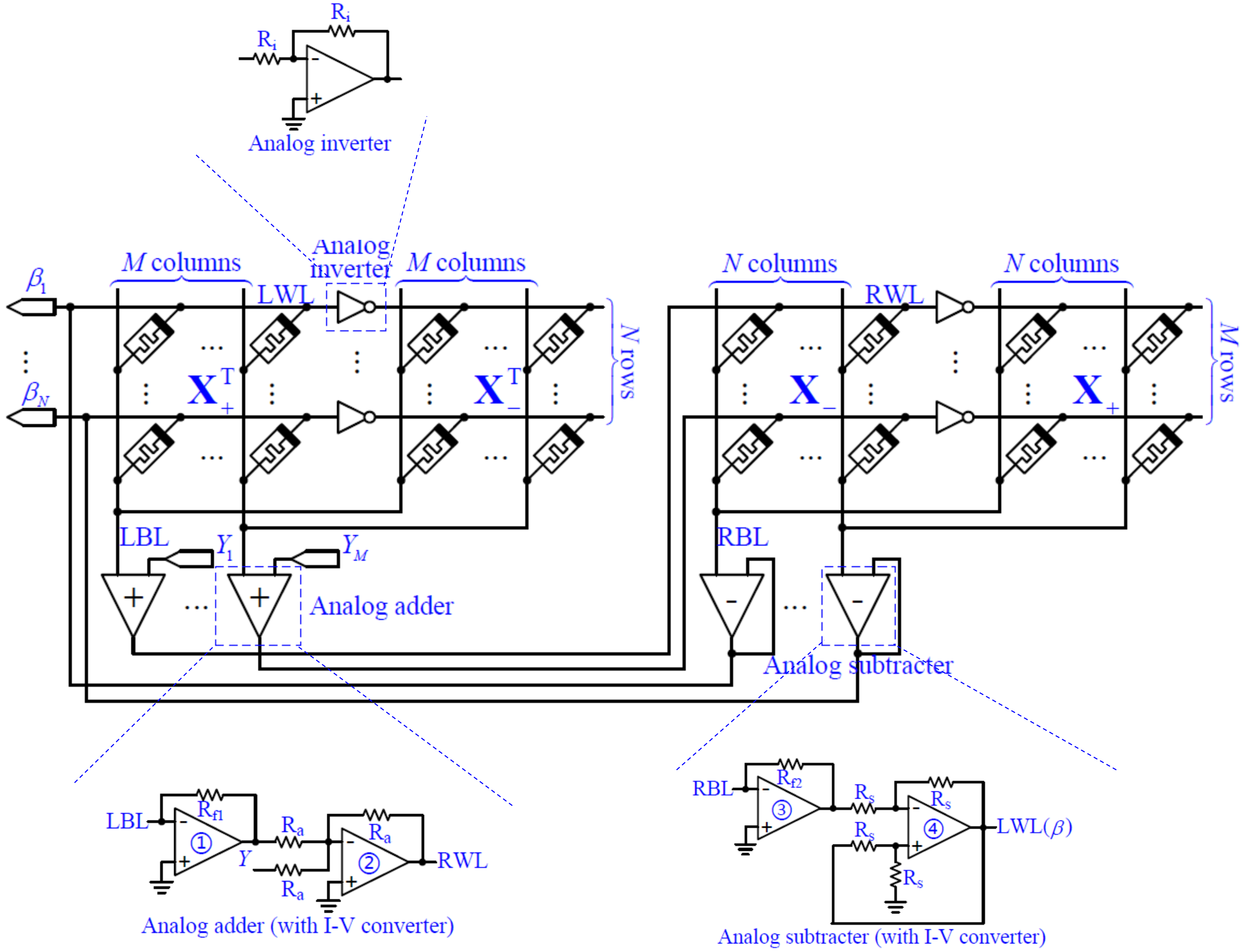


$$\boldsymbol{\beta} = \boldsymbol{\beta} + \eta \mathbf{X}^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})$$

$$\eta \mathbf{X}^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) = \mathbf{0}$$

$$\mathbf{X}\boldsymbol{\beta} = \mathbf{Y}$$

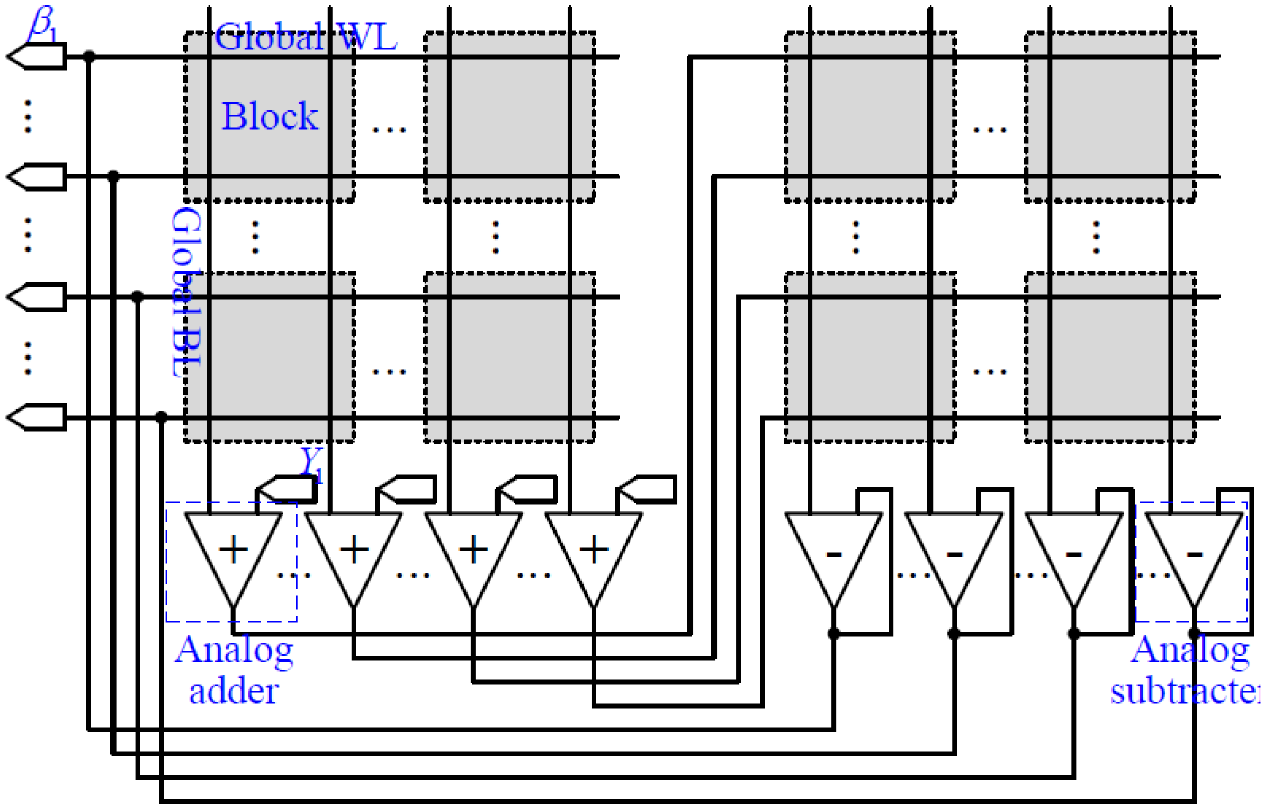
Circuit Design



- $\mathbf{X}_+$ and $\mathbf{X}_-$ store positive and negative values of $\mathbf{X}$, respectively
- OpAmp-based peripheral circuits perform analog operations (inversion, add and subtraction)

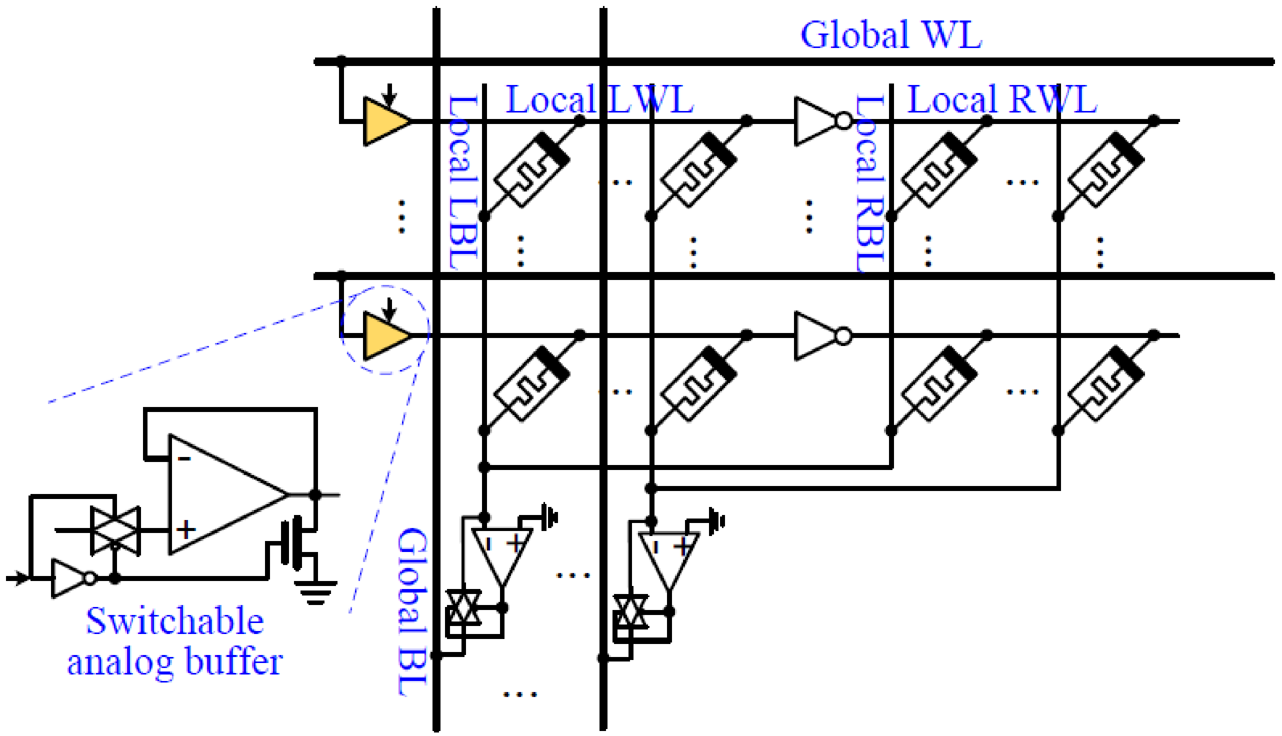
Architecture for Large-Scale Problems

- Size of a single crossbar array is limited
- To handle large-scale problems, we propose a scalable and configurable architecture
- Composed of a set of blocks and peripheral circuits



Block Circuit Design

- Two crossbar arrays in a block, storing positive and negative values, respectively
- Switchable analog buffers to control whether a block is ON or OFF
- Bitlines' currents gathered to global bitlines

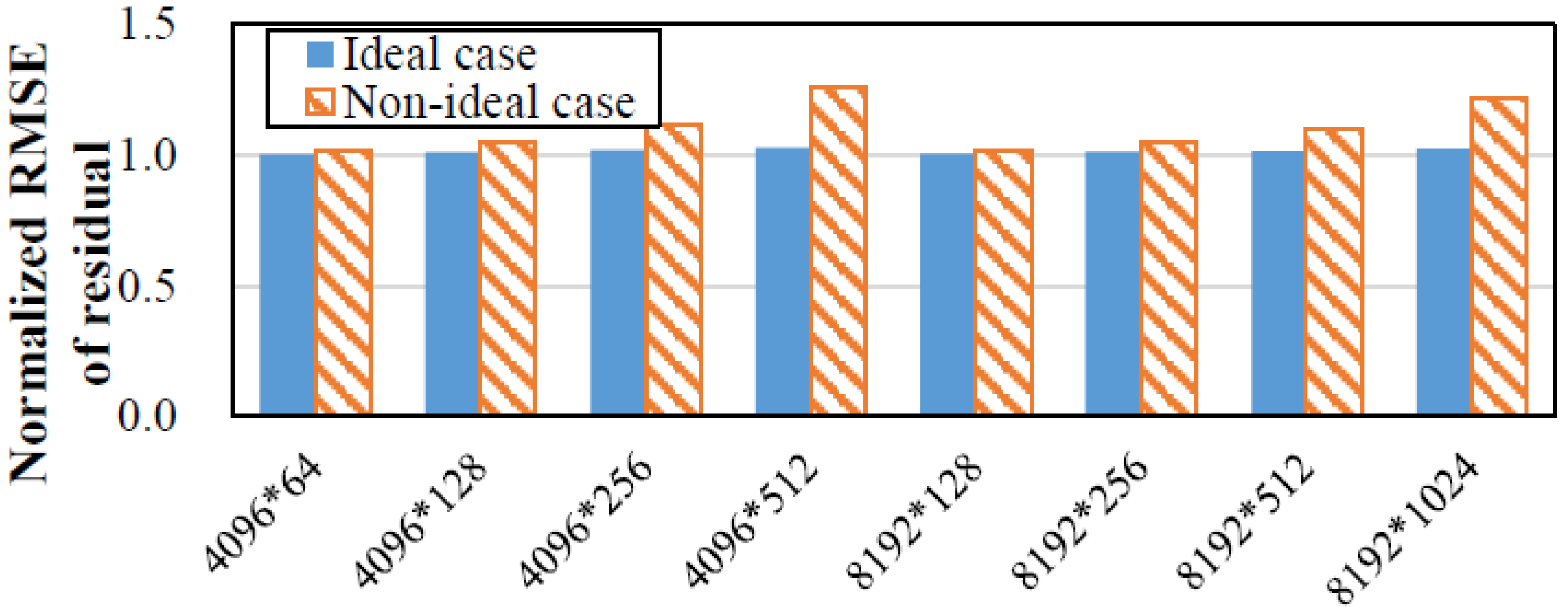


Simulation Results

- Circuits simulated with HSPICE
- Crossbar array size is 512*512
- RRAM resistance range: LRS=5K, HRS=5M
- Baseline: GPU-accelerated software solver with cuBLAS and cuSOLVER running on NVIDIA K40m GPU

Accuracy of Solutions

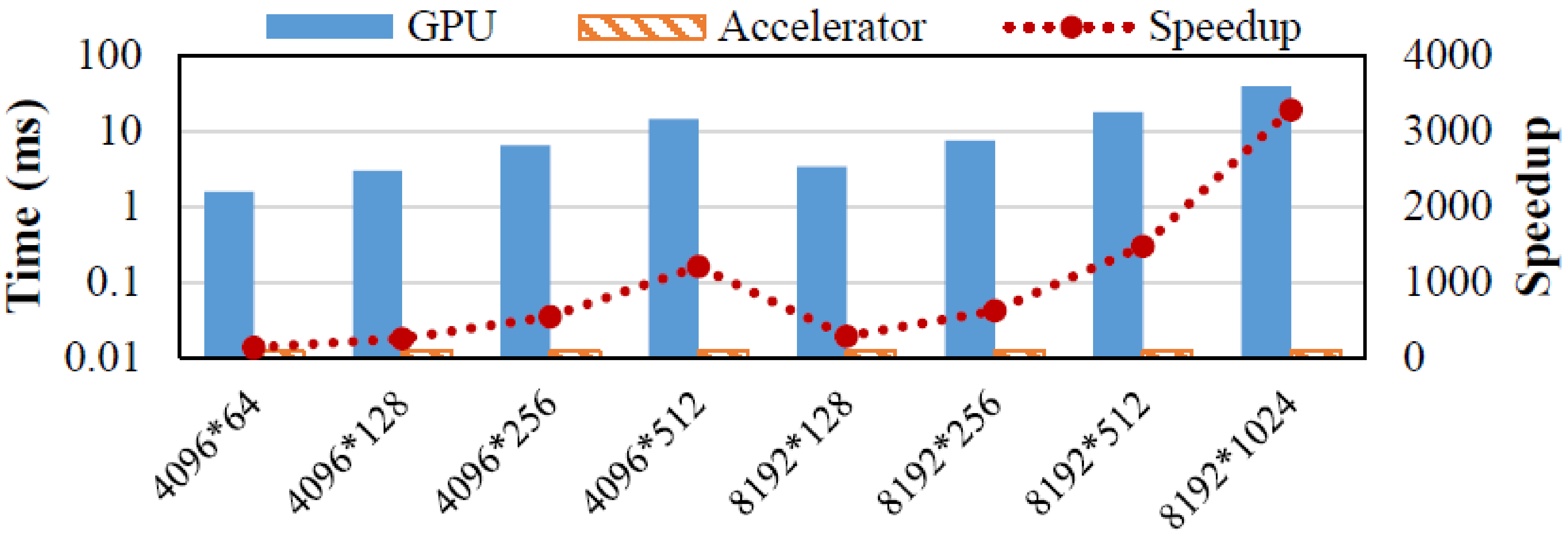
- Ideal case: no resistance variation & no resistance limits (LRS & HRS)
- Non-ideal case: RRAM resistance has sigma=20% variation & resistance limits (LRS & HRS) applied
- RMSE normalized to GPU results



Non-ideal case: 2-26% larger error than GPU solutions

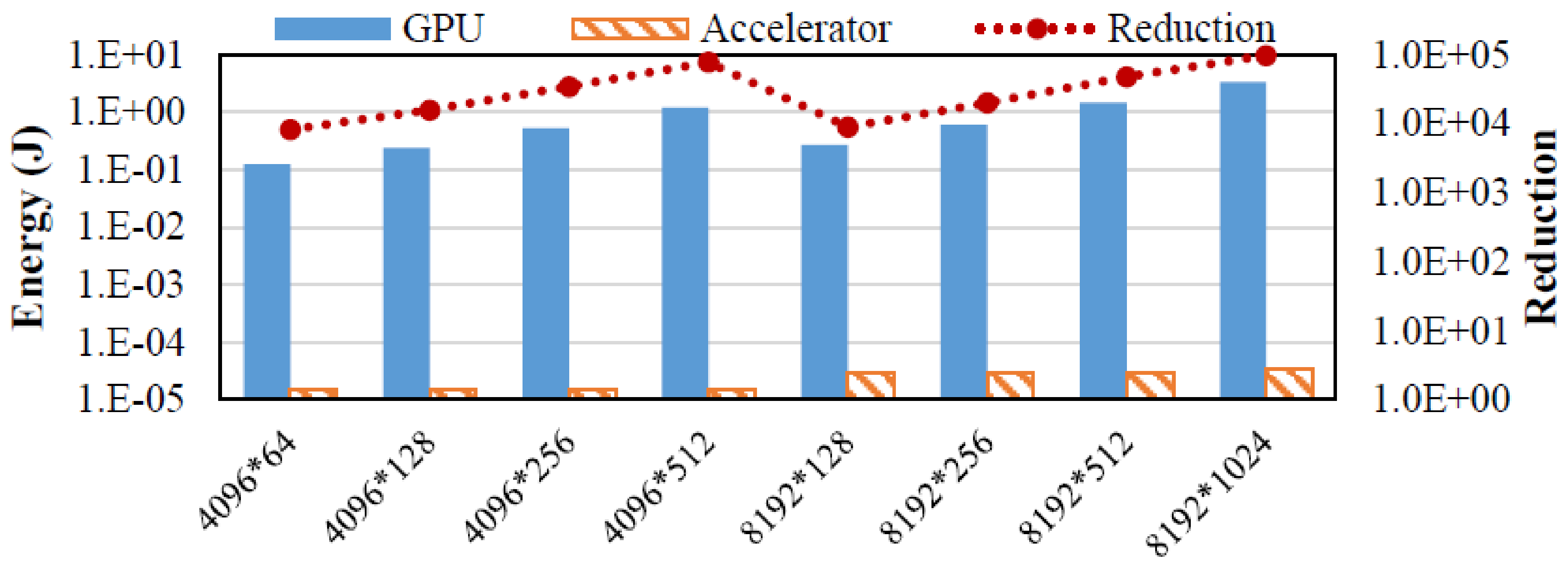
Performance

- Time complexity from $O(N^3)$ to O(1)
- Higher speedup for larger problems



132-3282X speedup vs. GPU solver

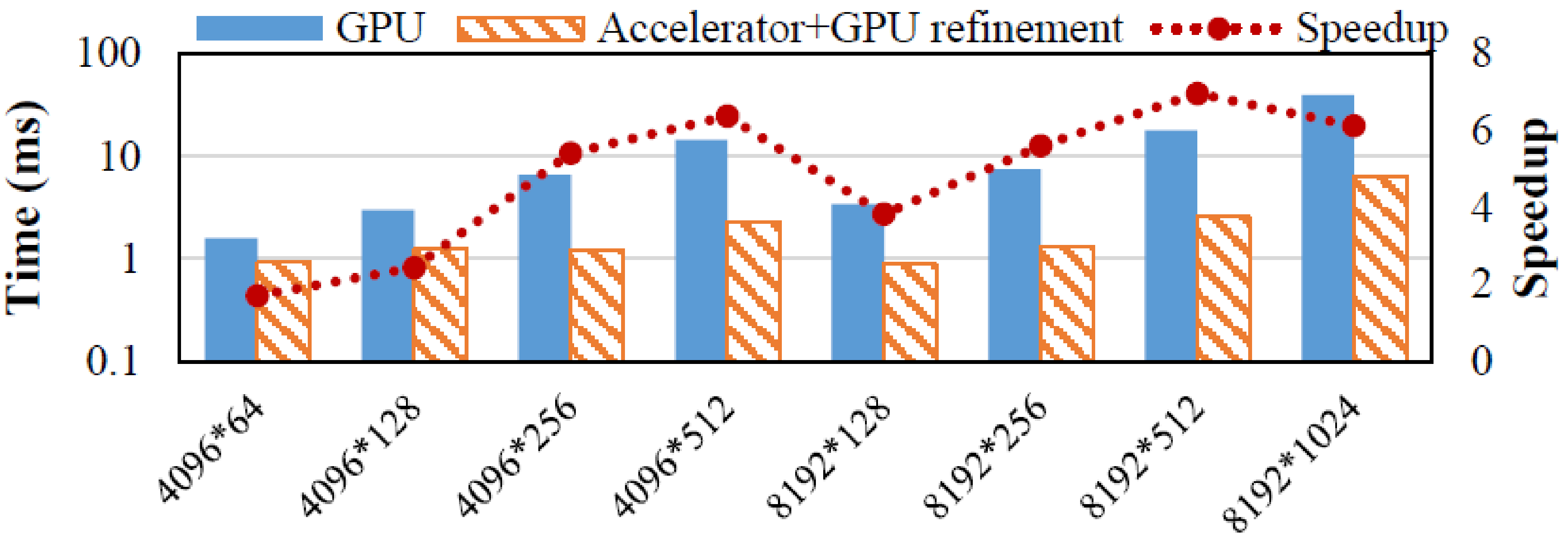
Energy Consumption



8201-96738X energy reduction vs. GPU solver

Solution Refinement

- Approximate solution obtained by accelerator is used as initial guess for further refinement on GPU
- Approximate solution close to precise solution, fewer iterations



1.7-7X speedup vs. pure GPU solver

Conclusion

- **Least-squares fitting can be finished in O(1) time complexity by utilizing closed-loop principle based on RRAM-based computing-in-memory accelerator**
- **2-3 orders of magnitude speedups and 4-5 orders of magnitude energy reduction compared with GPU solver; 1.7-7X speedups with GPU refinement compared with pure GPU solver**

Thanks for Your Attention