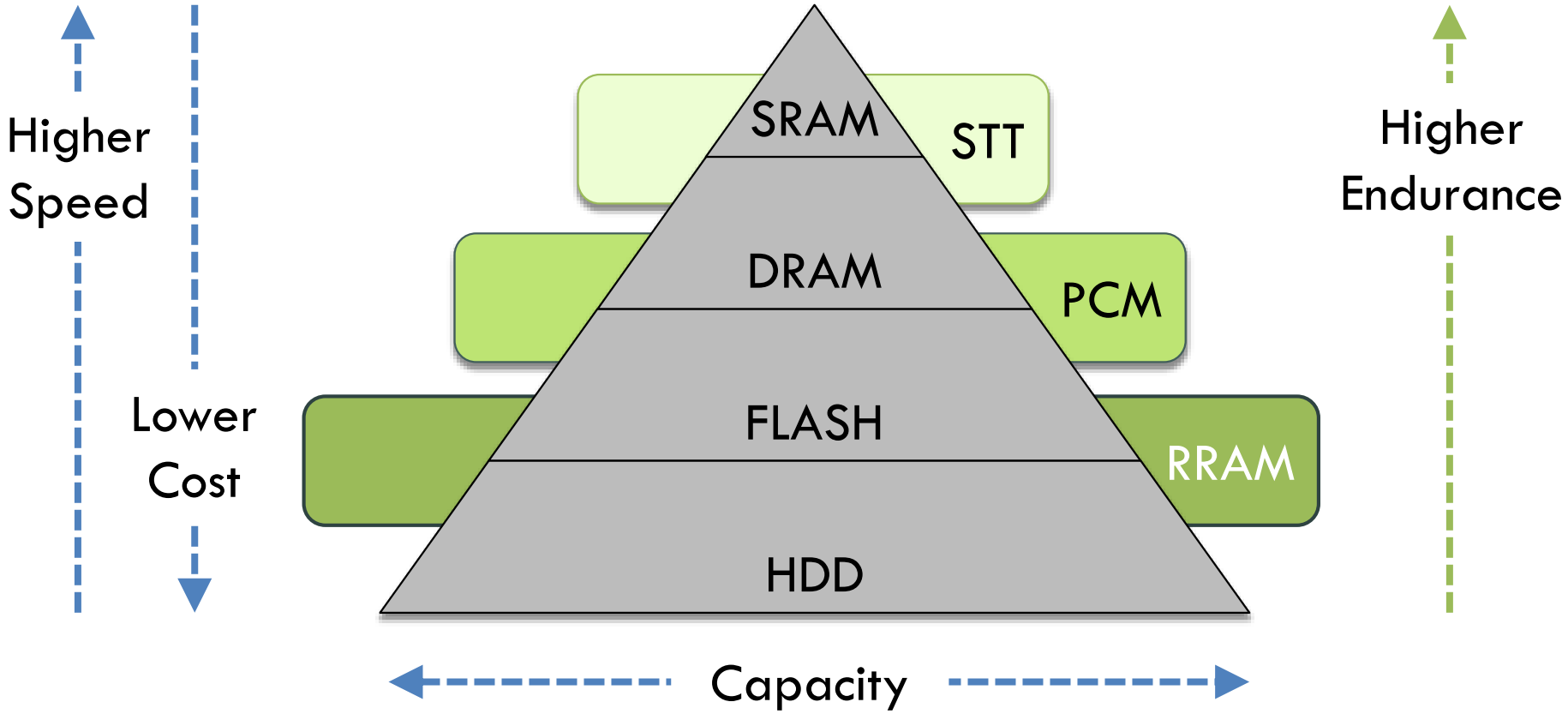
Memristive Accelerators for Data Intensive Computing: From Machine Learning to High Performance Linear Algebra

Engin Ipek

Department of Electrical and Computer Engineering
University of Rochester

The Changing Landscape of Memory Technologies

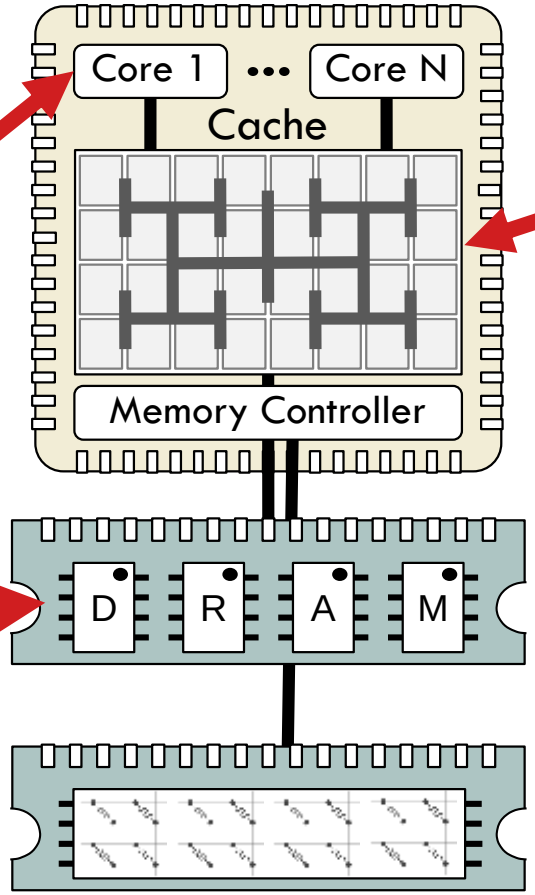


Architectures Leveraging Emerging Memories

- Opportunity: **improve** and augment the existing memory hierarchy

STT-MRAM based microarchitectures
[ISCA'10] [TCAS-II]

Byte addressable, persistent memory
[ASPLOS'10 Best Paper]
[MICRO Top Picks'10]
[SOSP'09] [ISCA'09]



STT-MRAM caches
[TVLSI'15] [TED'15]
[TED'14]

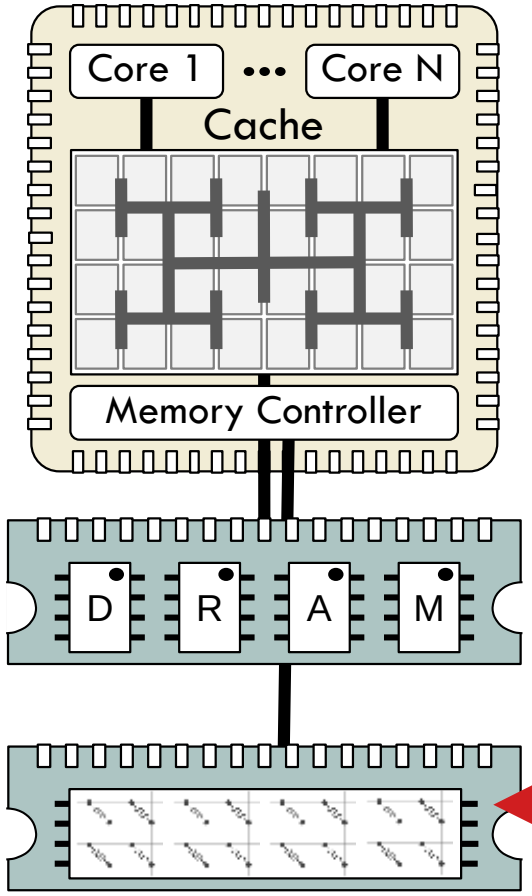
In-Situ Memristive Accelerators
[HPCA'18]
[MICRO Top Picks'17]
[HPCA'16 Best Paper]
[GOMAC'15]
[ISCA'13] [MICRO'11]

Architectures Leveraging Emerging Memories

- Opportunity: improve and **augment** the existing memory hierarchy

STT-MRAM based microarchitectures
[ISCA'10] [TCAS-II]

Byte addressable, persistent memory
[ASPLOS'10 Best Paper]
[MICRO Top Picks'10]
[SOSP'09] [ISCA'09]



STT-MRAM caches
[TVLSI'15] [TED'15]
[TED'14]

In-Situ Memristive Accelerators
[HPCA'18]
[MICRO Top Picks'17]
[HPCA'16 Best Paper]
[GOMAC'15]
[ISCA'13] [MICRO'11]

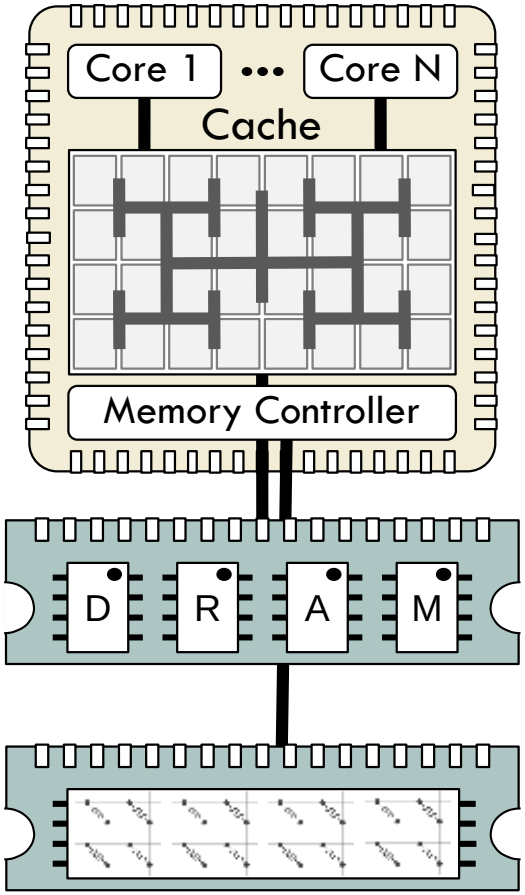
UNIVERSITY of ROCHESTER

This Talk

- Opportunity: improve and **augment** the existing memory hierarchy

STT-MRAM based microarchitectures
[ISCA'10] [TCAS-II]

Byte addressable, persistent memory
[ASPLOS'10 Best Paper]
[MICRO Top Picks'10]
[SOSP'09] [ISCA'09]



STT-MRAM caches
[TVLSI'15] [TED'15]
[TED'14]

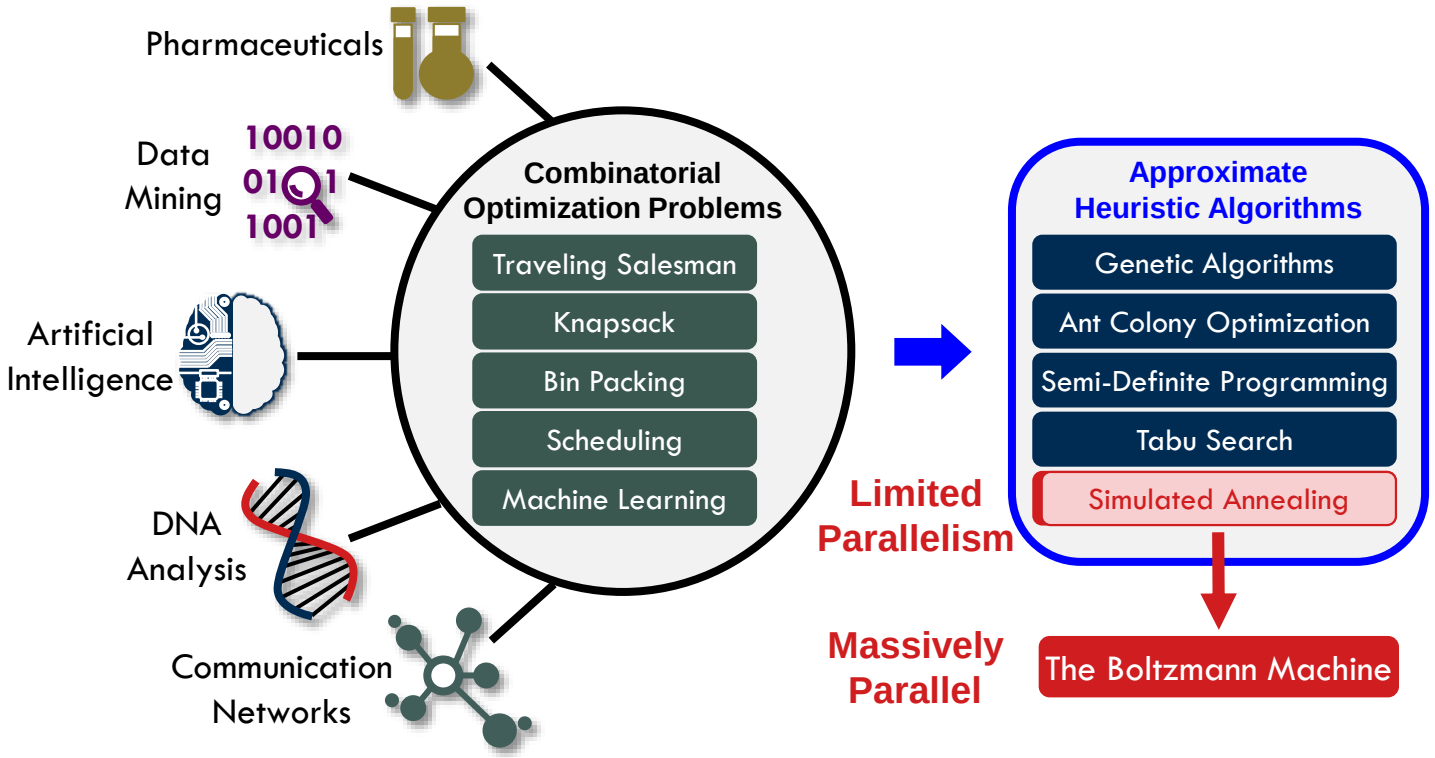
In-Situ Memristive Accelerators
[HPCA'18]
[MICRO Top Picks'17]
[HPCA'16 Best Paper]
[GOMAC'15]
[ISCA'13] [MICRO'11]

Memristive Boltzmann Machine

A hardware accelerator for combinatorial optimization and deep learning

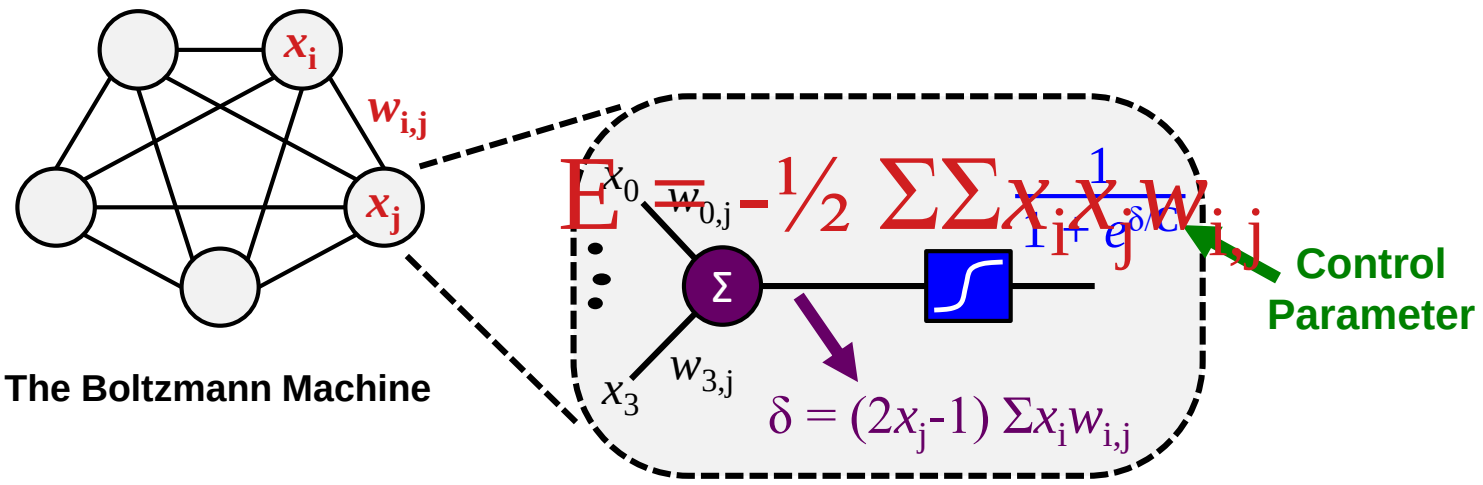
Combinatorial Optimization

- Numerous critical problems in science and engineering can be cast within the combinatorial optimization framework.



The Boltzmann Machine

- Two-state units connected with real-valued edge weights form a stochastic neural network.
- Goal: iteratively update the state or weight variables to minimize the network energy (E).



The Boltzmann Machine

Minimizing Network Energy

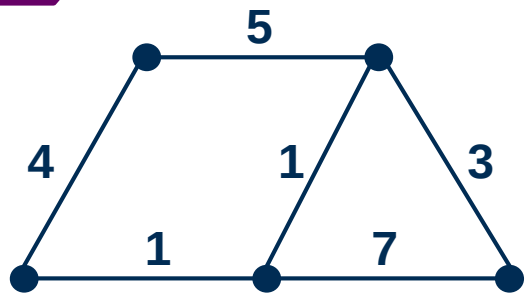
- Two-state units connected with real-valued edge weights form a stochastic neural network.
- Goal: iteratively update the state or weight variables to minimize the network energy (E).

The Boltzmann Machine

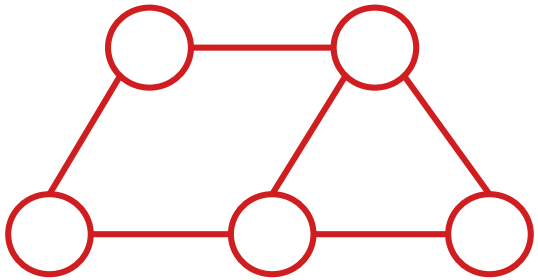
Example: Maximum Cut

- ✔ Choose appropriate instance of Boltzmann machine.
- ❑ Choose appropriate edge weights.
- ❑ Choose appropriate biases.
- ❑ Initialize the network state.
- ❑ Minimize energy by adjusting the unit states.
- ❑ Read and interpret the final state of the network.

Graphical Model



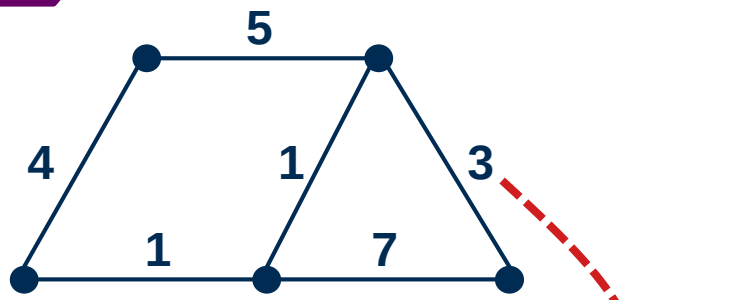
Boltzmann Machine



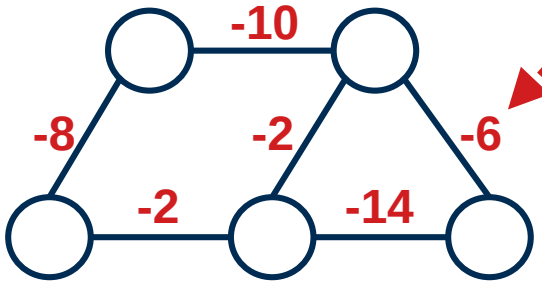
Example: Maximum Cut

- ✔ Choose appropriate instance of Boltzmann machine.
- ✔ Choose appropriate edge weights.
- ☐ Choose appropriate biases.
- ☐ Initialize the network state.
- ☐ Minimize energy by adjusting the unit states.
- ☐ Read and interpret the final state of the network.

Graphical Model



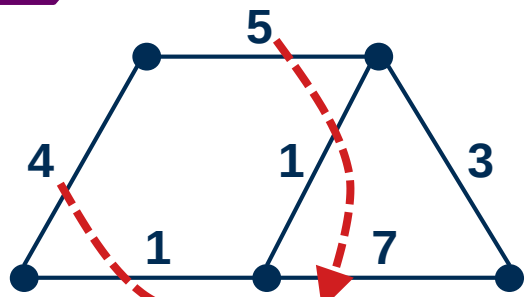
Boltzmann Machine



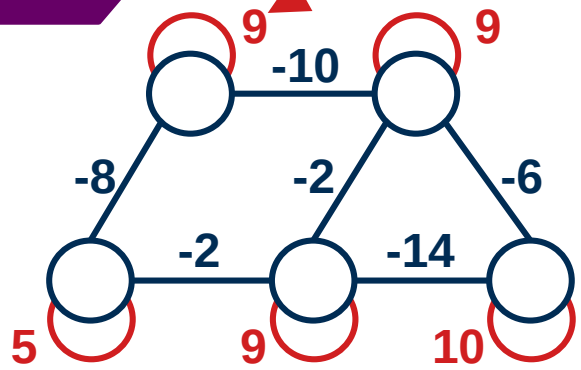
Example: Maximum Cut

- ✔ Choose appropriate instance of Boltzmann machine.
- ✔ Choose appropriate edge weights.
- ✔ Choose appropriate biases.
- ☐ Initialize the network state.
- ☐ Minimize energy by adjusting the unit states.
- ☐ Read and interpret the final state of the network.

Graphical Model



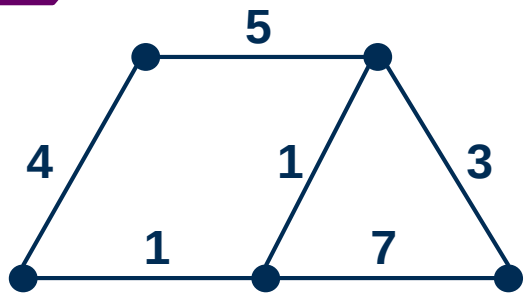
Boltzmann Machine



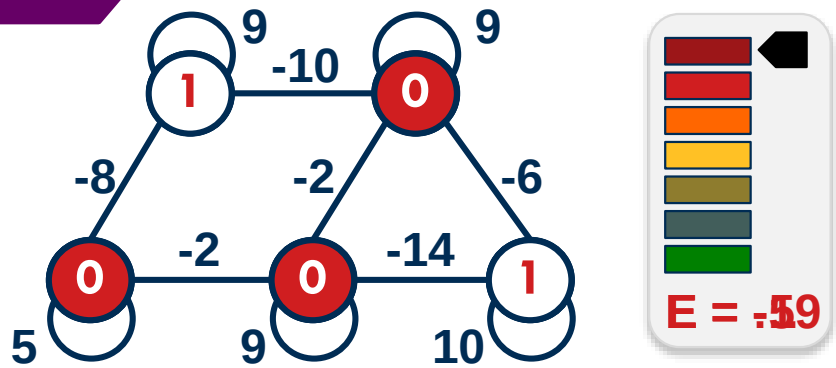
Example: Maximum Cut

- ✔ Choose appropriate instance of Boltzmann machine.
- ✔ Choose appropriate edge weights.
- ✔ Choose appropriate biases.
- ✔ Initialize the network state.
- ✔ Minimize energy by adjusting the unit states.
- ☐ Read and interpret the final state of the network.

Graphical Model



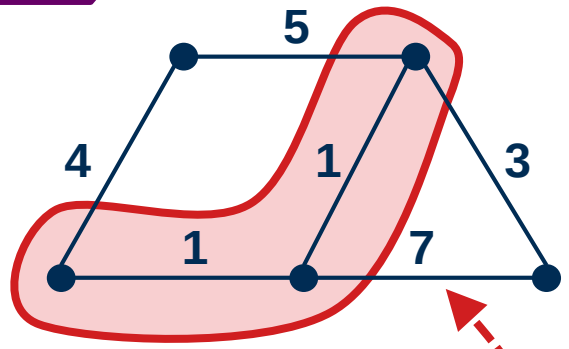
Boltzmann Machine



Example: Maximum Cut

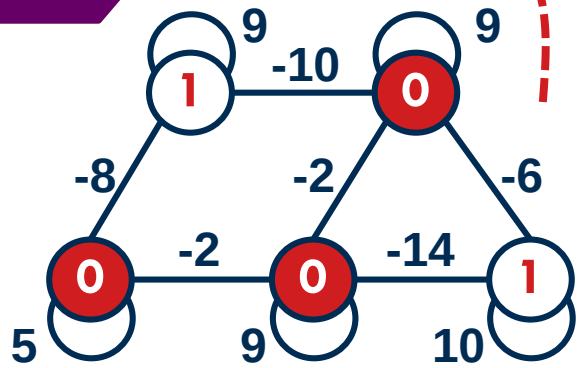
- ✔ Choose appropriate instance of Boltzmann machine.
- ✔ Choose appropriate edge weights.
- ✔ Choose appropriate biases.
- ✔ Initialize the network state.
- ✔ Minimize energy by adjusting the unit states.
- ✔ Read and interpret the final state of the network.

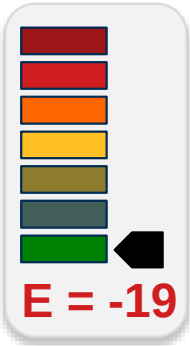
Graphical Model



Cost = 19

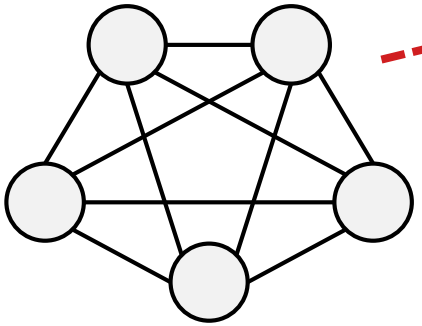
Boltzmann Machine





Computational Model

- Network energy is minimized by adjusting either the edge weights or recomputing the states.
- Iterative matrix-vector multiplication between weights and states is critical to finding minimal network energy.

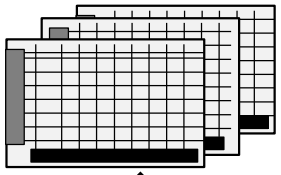


The Boltzmann Machine

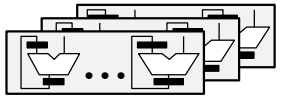
$$\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots \\ w_{1,0} & & \\ \vdots & \ddots & \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \end{bmatrix}$$

$$\Sigma,\ \times,\ \frac{1}{1+e^{x}}$$

Memory Arrays



Data Movement

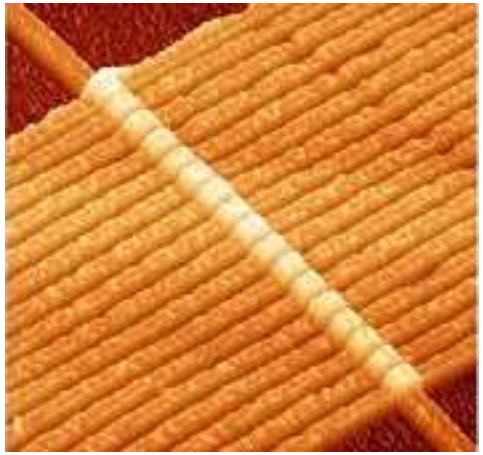


Functional Units

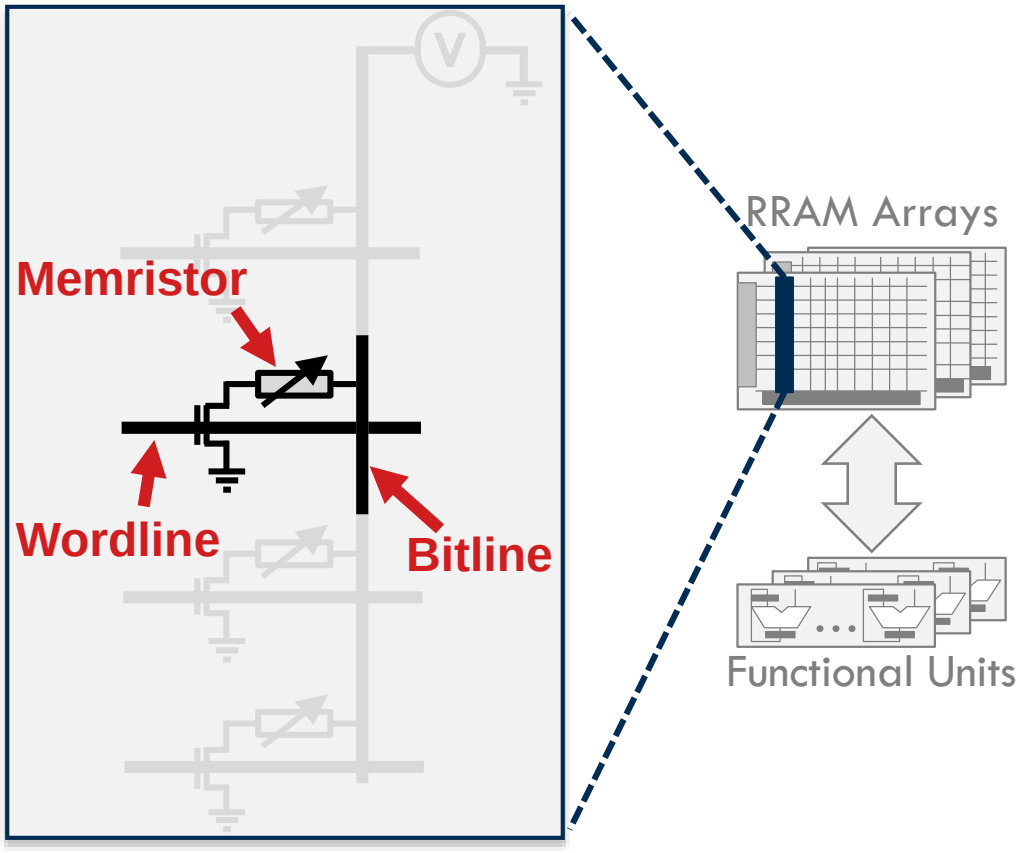
UNIVERSITY of ROCHESTER

Resistive Random Access Memory

- An RRAM cell comprises an access transistor and a resistive switching medium.



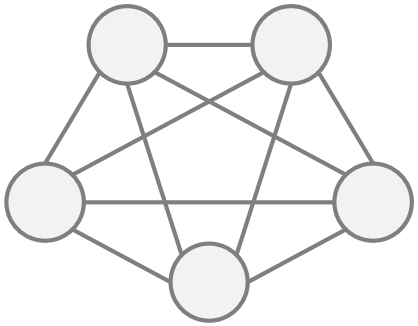
RRAM: Resistive RAM
(source: HP, 2009)



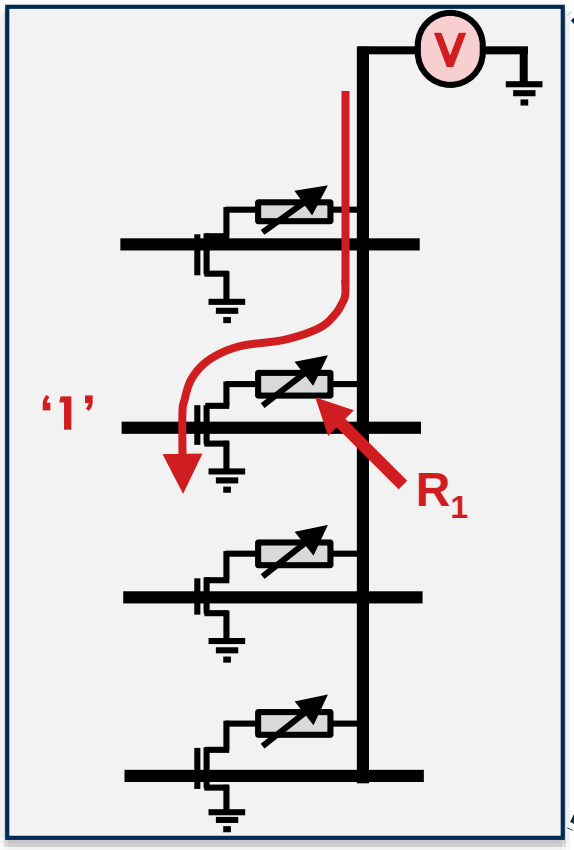
Resistive Random Access Memory

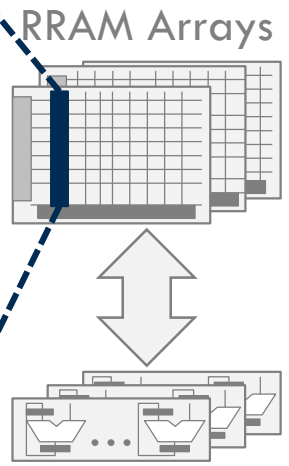
- A read is performed by activating a wordline and measuring the bitline current (I).

$I = V/R_1$



The Boltzmann Machine

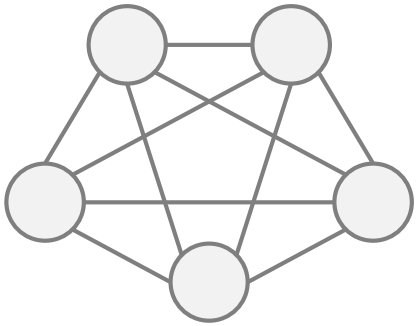




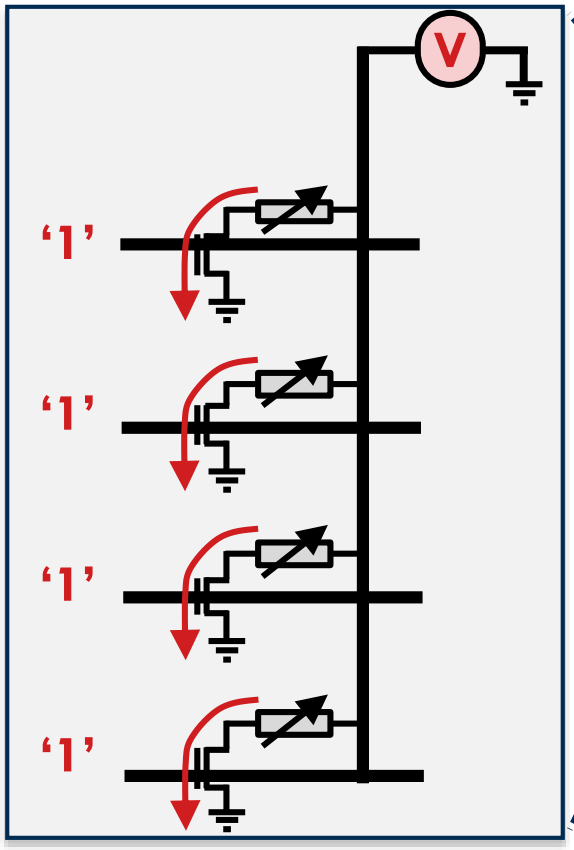
Memristive Boltzmann Machine

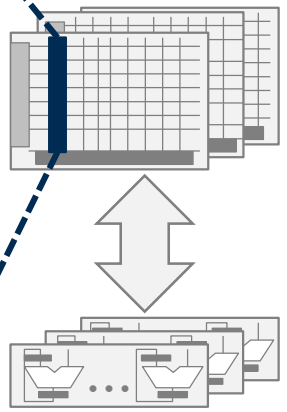
- Key Idea: exploit current summation on the RRAM bitlines to compute dot product.

$I = \Sigma V/R_i$



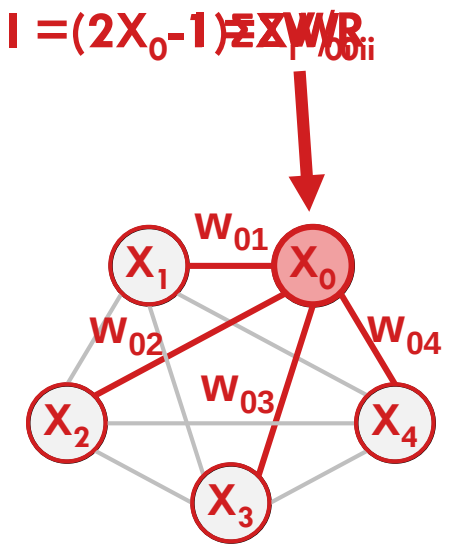
The Boltzmann Machine



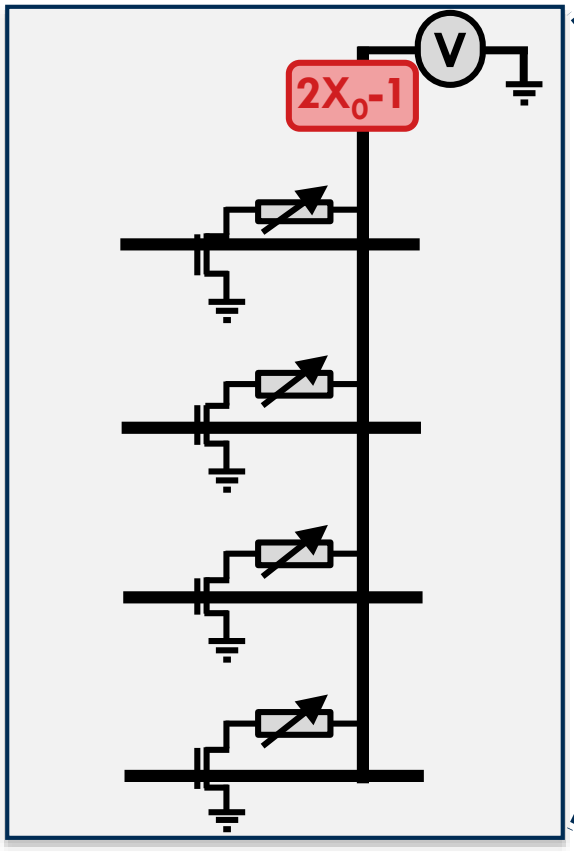


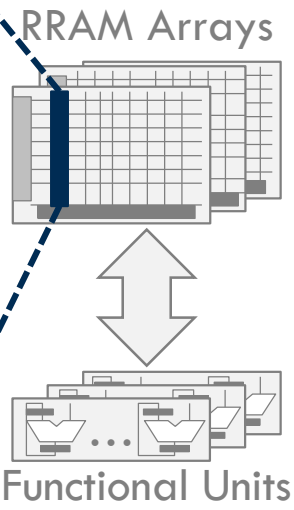
Memristive Boltzmann Machine

- Memory cells represent the weights and state variables are used to control the bitline and wordlines.



The Boltzmann Machine

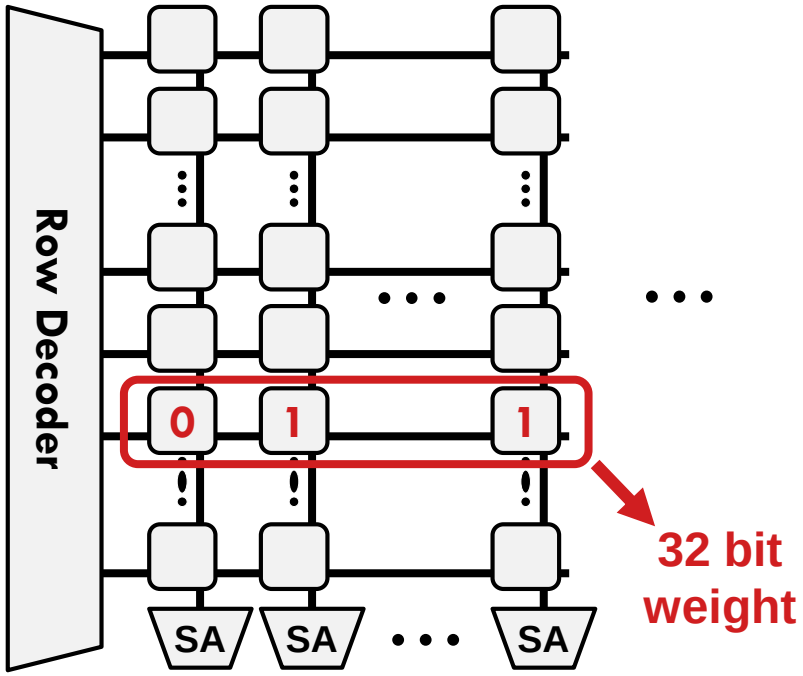




Array Structure

- ✔ 1T-1R array is employed to store the connection weights (**W**).
- State variables (**x**) are kept at the periphery.
- Column sense amplifier quantizes current into a multi-bit digital value.
- Bit summation tree merges the partial sums generated by sense amps.

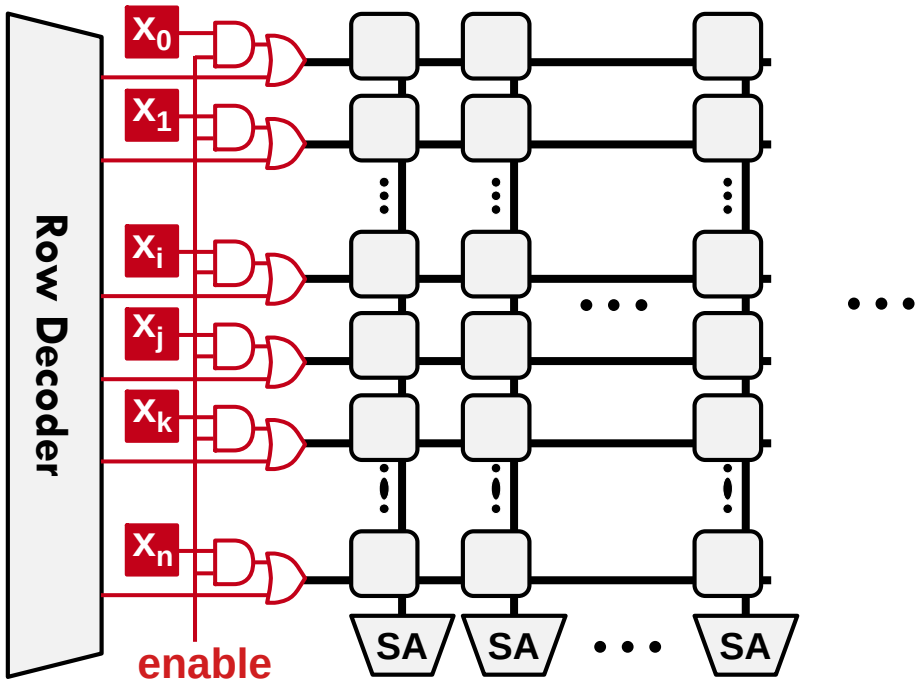
Represent weights in fixed point, 2's complement.



Array Structure

- ✔ 1T-1R array is employed to store the connection weights (**W**).
- ✔ State variables (**x**) are kept at the periphery.
- Column sense amplifier quantizes current into a multi-bit digital value.
- Bit summation tree merges the partial sums generated by sense amps.

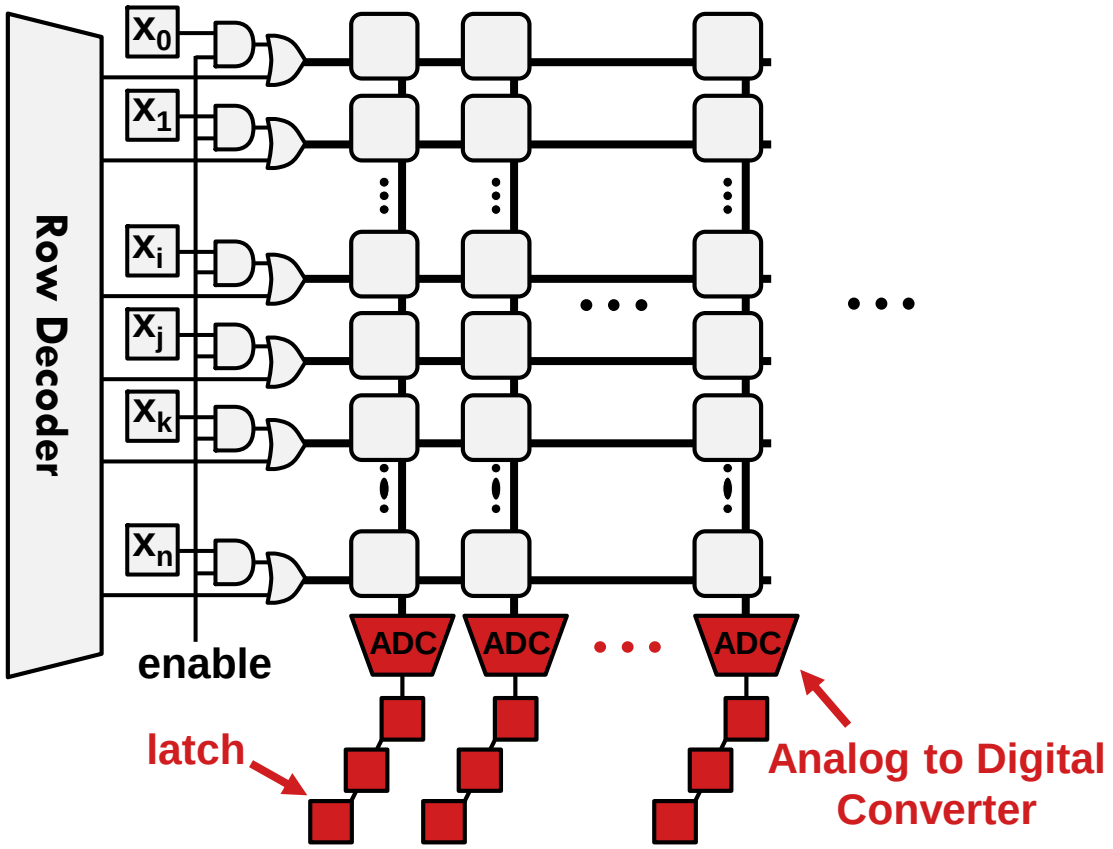
Employ CMOS latches for efficient state updates.



Array Structure

- ✔ 1T-1R array is employed to store the connection weights (**W**).
- ✔ State variables (**x**) are kept at the periphery.
- ✔ Column sense amplifier quantizes current into a multi-bit digital value.
- Bit summation tree merges the partial sums generated by sense amps.

Count the number of 1s for column summation.

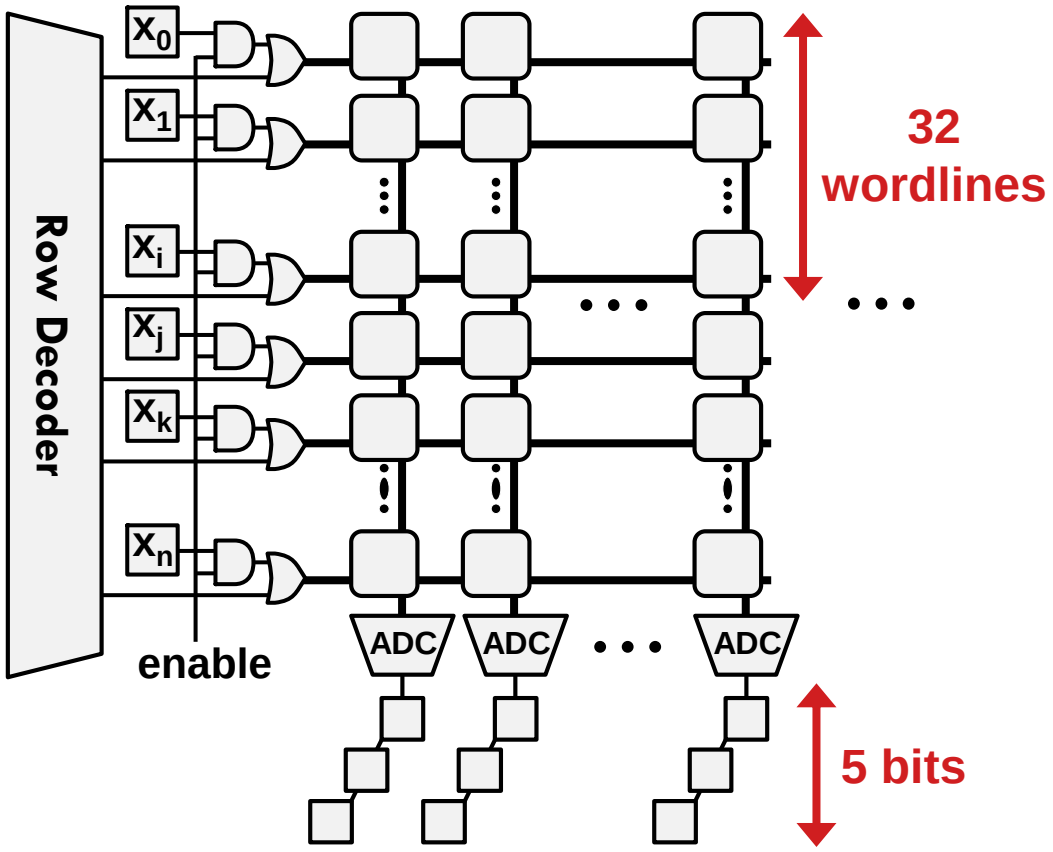


UNIVERSITY of ROCHESTER

Array Structure

- ✔ 1T-1R array is employed to store the connection weights (**W**).
- ✔ State variables (**x**) are kept at the periphery.
- ✔ Column sense amplifier quantizes current into a multi-bit digital value.
- Bit summation tree merges the partial sums generated by sense amps.

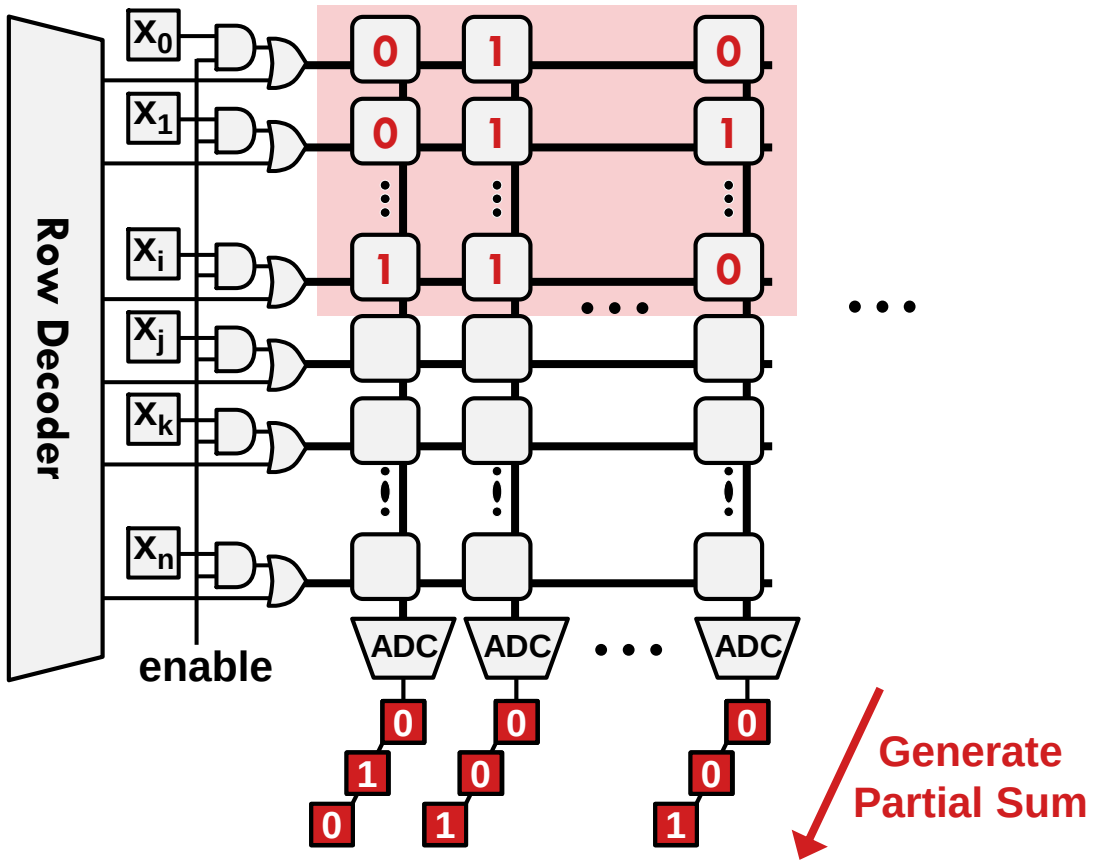
Iteratively generate partial sums.



Array Structure

- ✔ 1T-1R array is employed to store the connection weights (**W**).
- ✔ State variables (**x**) are kept at the periphery.
- ✔ Column sense amplifier quantizes current into a multi-bit digital value.
- Bit summation tree merges the partial sums generated by sense amps.

Iteratively generate partial sums.

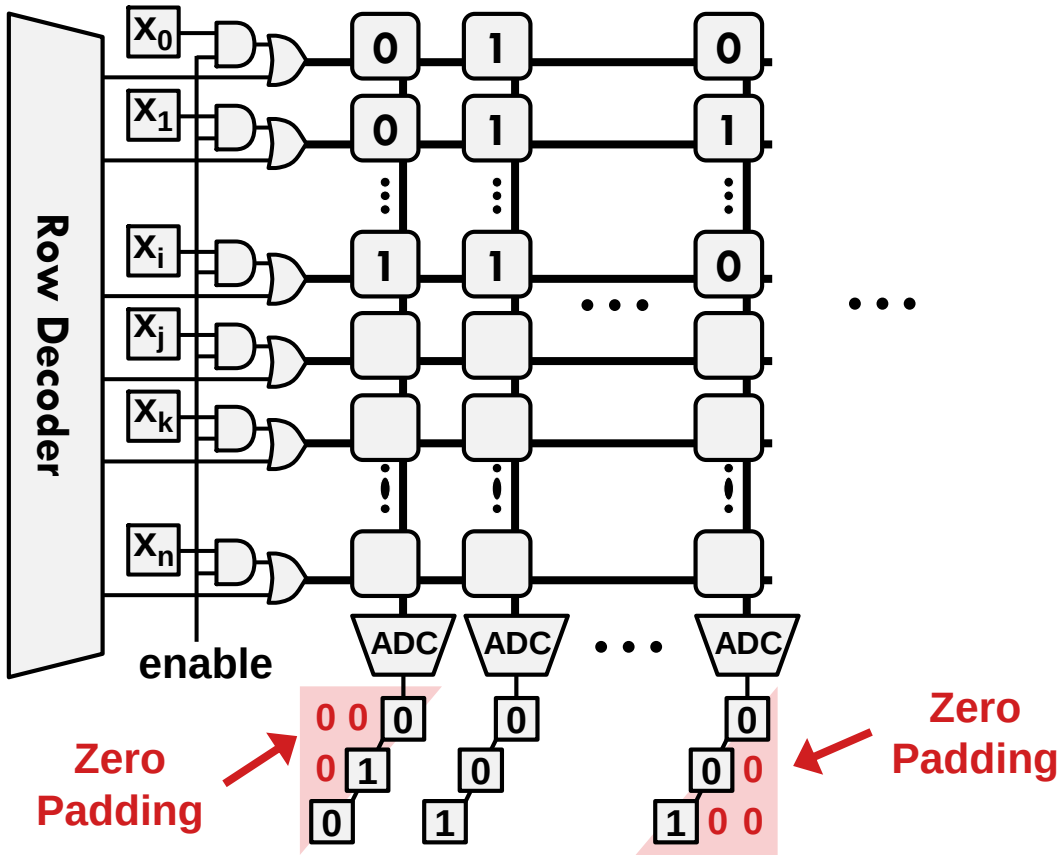


UNIVERSITY of ROCHESTER

Array Structure

- ✔ 1T-1R array is employed to store the connection weights (**W**).
- ✔ State variables (**x**) are kept at the periphery.
- ✔ Column sense amplifier quantizes current into a multi-bit digital value.
- ✔ Bit summation tree merges the partial sums generated by sense amps.

Add zeroes to align the partial sum bits.

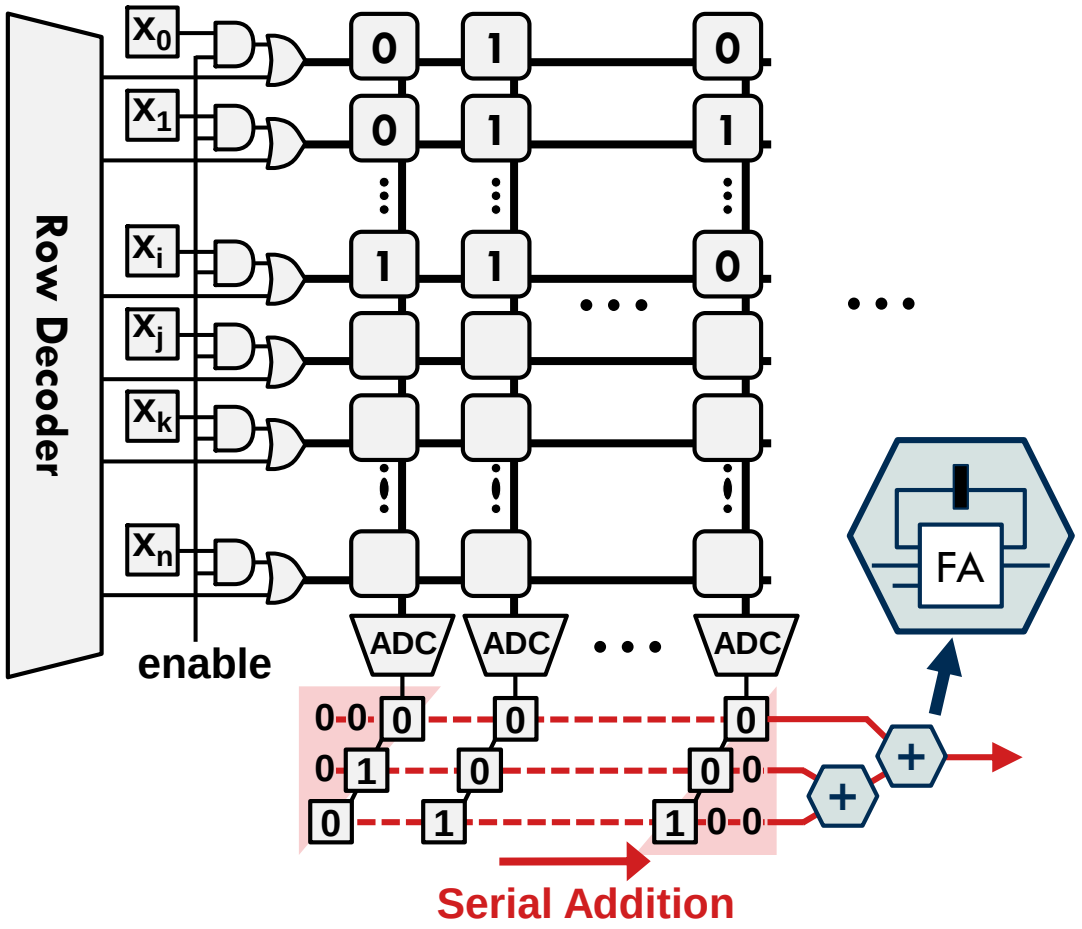


UNIVERSITY of ROCHESTER

Array Structure

- ✔ 1T-1R array is employed to store the connection weights (**W**).
- ✔ State variables (**x**) are kept at the periphery.
- ✔ Column sense amplifier quantizes current into a multi-bit digital value.
- ✔ Bit summation tree merges the partial sums generated by sense amps.

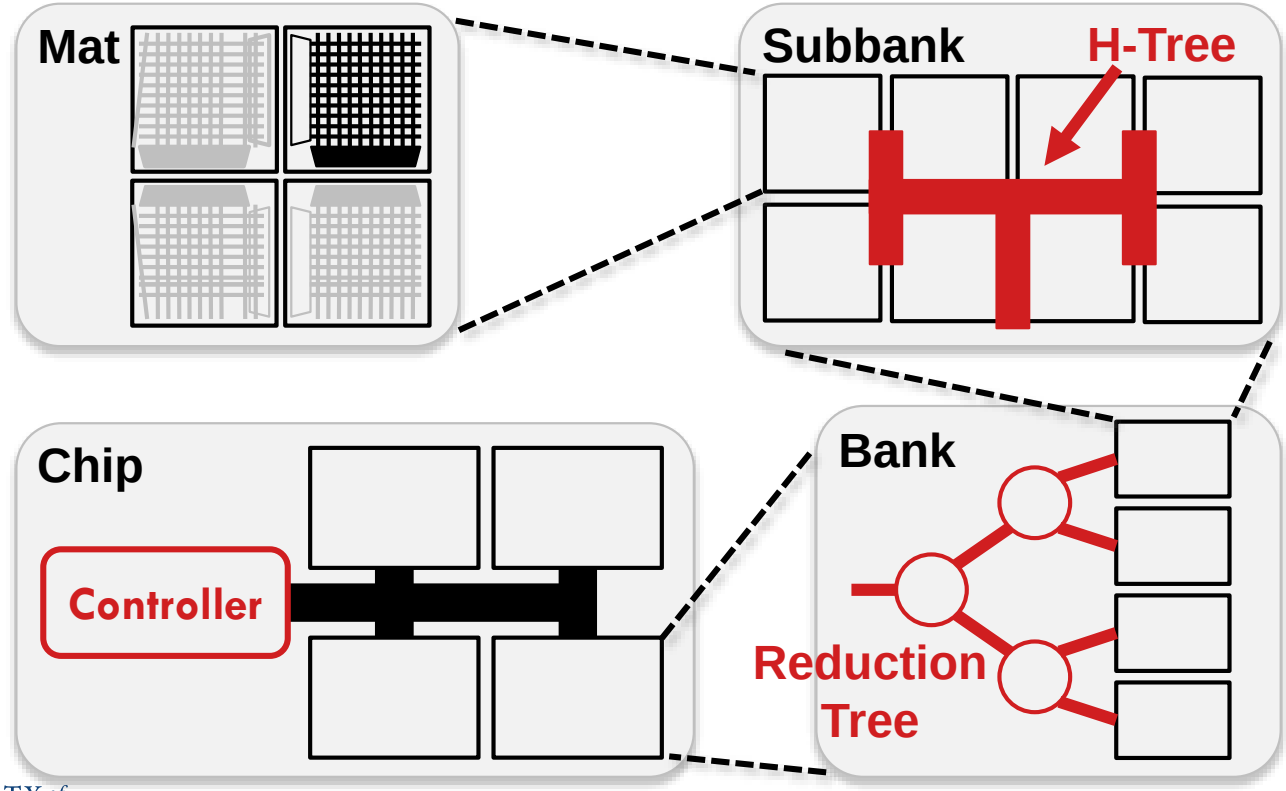
Add zeroes to align the partial sum bits.



UNIVERSITY of ROCHESTER

Chip Organization

- A hierarchical organization with a configurable reduction tree is used to compute a large sum of bit products.

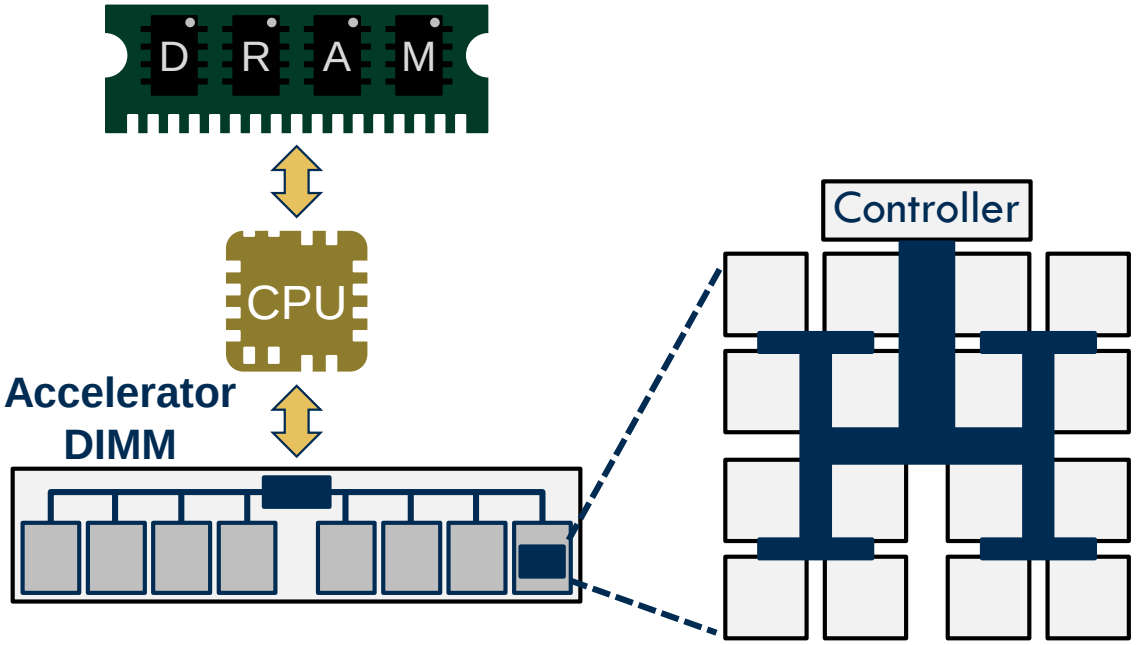


Interfacing with Software

Software configures the on-chip data layout and initiates the optimization by writing to a memory mapped control register.

To maintain ordering, accesses to the accelerator are made uncacheable by the processor.

DDR3 reads and writes are used for configuration and data transfer.



Hardware Acceleration

Software configures the on-chip data layout and initiates the optimization by writing to a memory mapped control register.

To maintain ordering, accesses to the accelerator are made uncacheable by the processor.

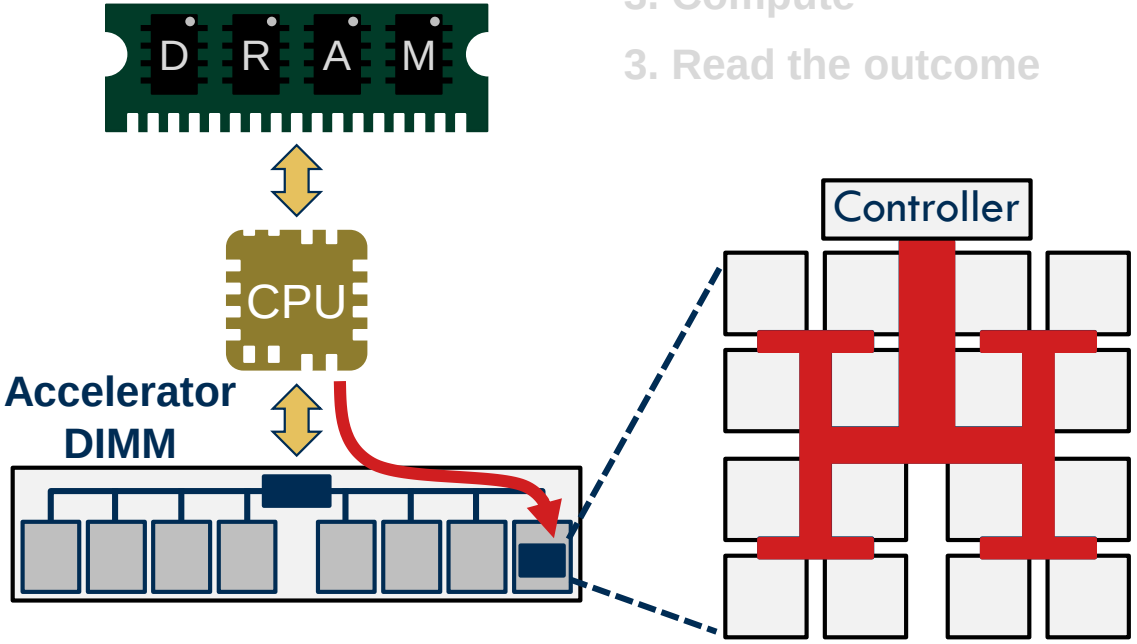
DDR3 reads and writes are used for configuration and data transfer.

1. Configure the DIMM

2. Write weights and states

3. Compute

3. Read the outcome



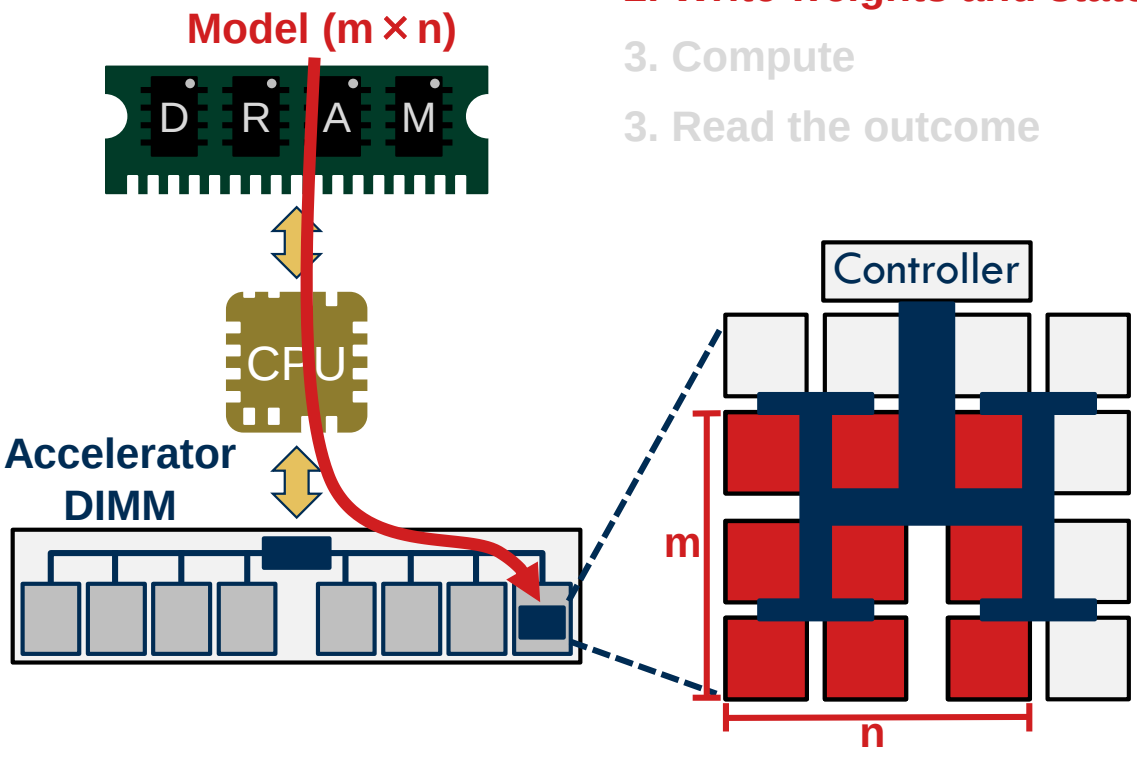
Hardware Acceleration

Software configures the on-chip data layout and initiates the optimization by writing to a memory mapped control register.

To maintain ordering, accesses to the accelerator are made uncacheable by the processor.

DDR3 reads and writes are used for configuration and data transfer.

1. **Configure the DIMM**
2. **Write weights and states**
3. Compute
3. Read the outcome



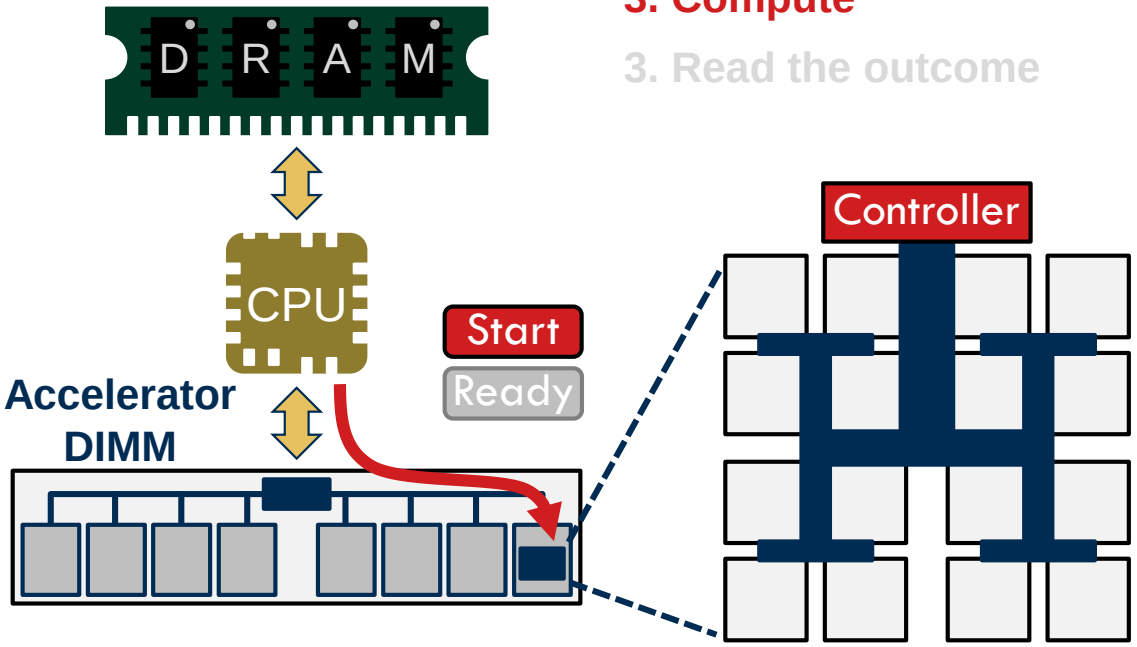
Hardware Acceleration

Software configures the on-chip data layout and initiates the optimization by writing to a memory mapped control register.

To maintain ordering, accesses to the accelerator are made uncacheable by the processor.

DDR3 reads and writes are used for configuration and data transfer.

1. Configure the DIMM
2. Write weights and states
3. Compute
3. Read the outcome



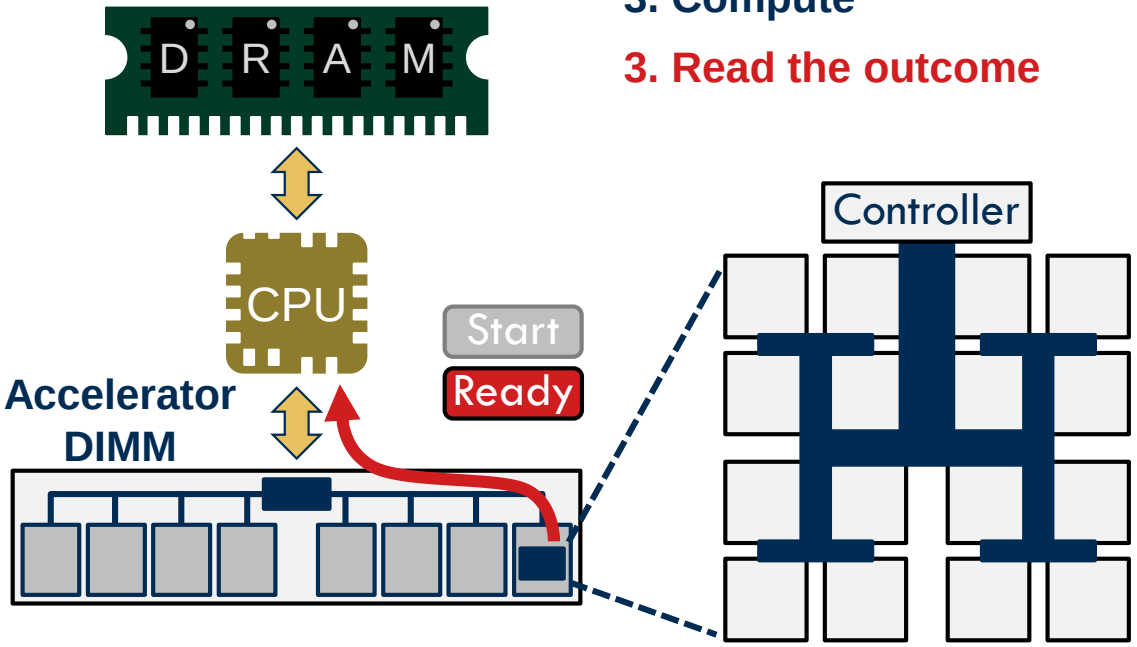
Hardware Acceleration

Software configures the on-chip data layout and initiates the optimization by writing to a memory mapped control register.

To maintain ordering, accesses to the accelerator are made uncacheable by the processor.

DDR3 reads and writes are used for configuration and data transfer.

1. Configure the DIMM
2. Write weights and states
3. Compute
3. Read the outcome

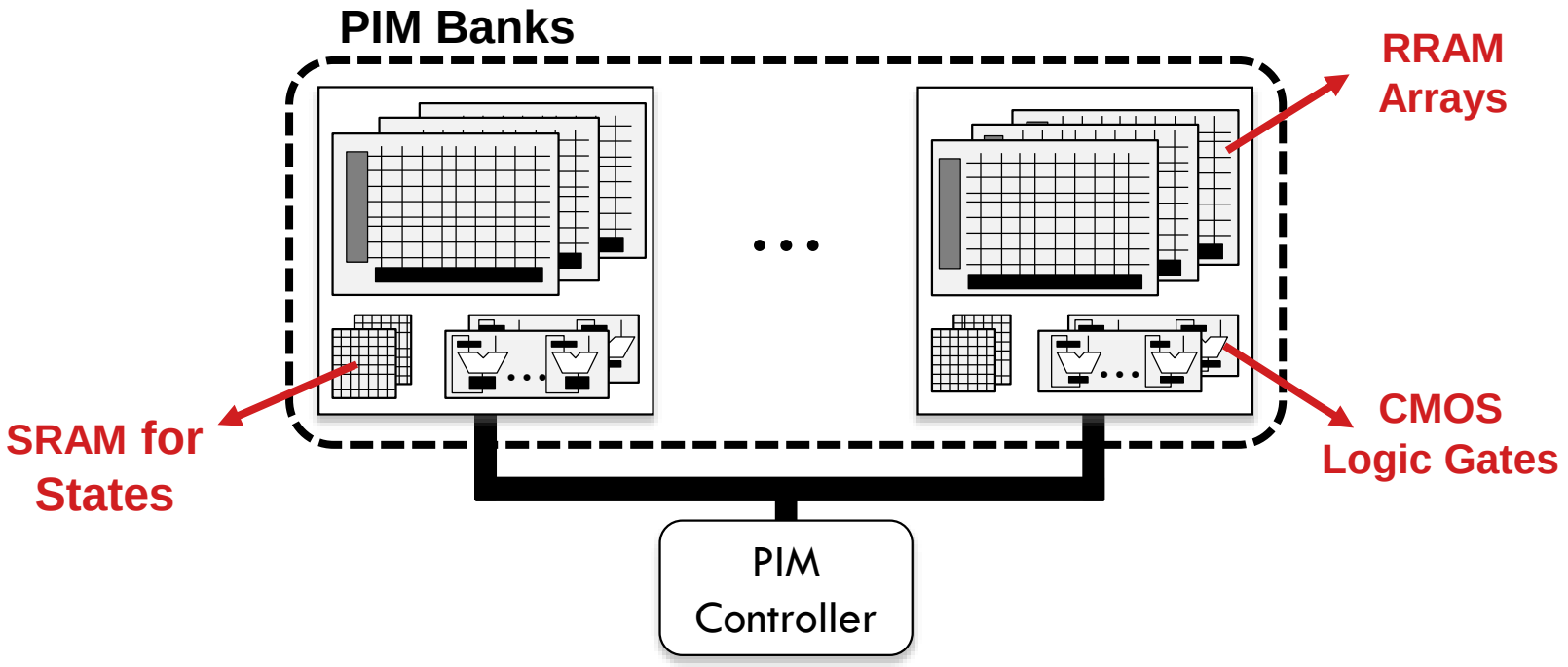


Experimental Setup

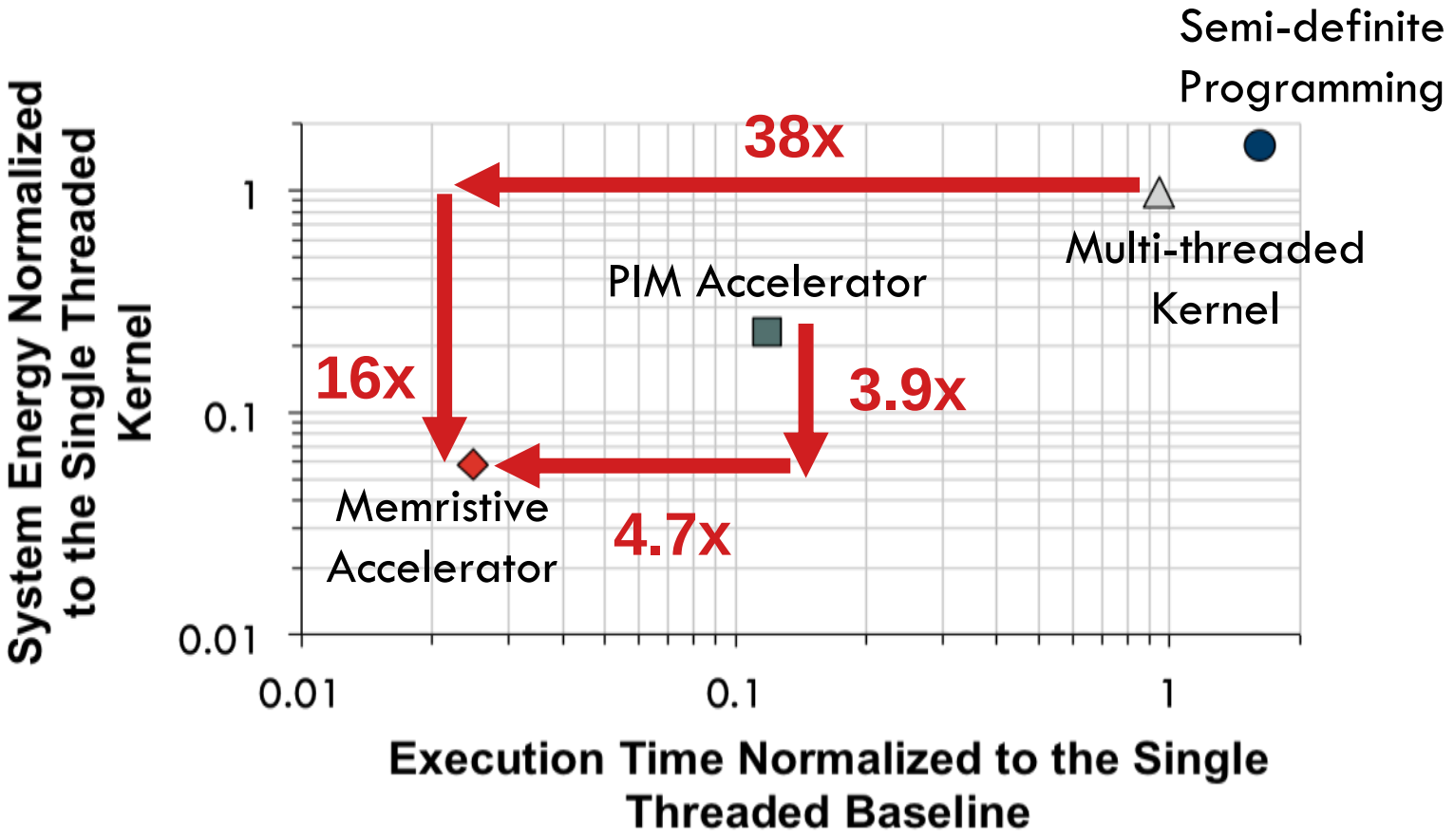
- Architecture
  - 1 and 8 out-of-order core(s)
  - 32KB private L1 caches
  - 8MB shared L2 cache
  - 4 DDR3-1600 DRAM channels
- Synthesis
  - Cadence RTL Compiler
  - FreePDK library
  - CACTI 6.5, NVSim with RRAM cells
  - McPAT
- Workloads
  - Max-Cut, Max-SAT, and DBN
  - 10 SAT problems (fault analysis)
  - 10 matrices (University of Florida)
  - Images from Olivetti at ATT
- Evaluated Baselines
  - Software kernels
    - Semi-definite programming (Max-Cut)
    - Max Walk SAT (Max-SAT)
    - Deep Belief network (DBN)
    - Boltzmann machine (Max-Cut, Max-SAT)
  - PIM-based hardware accelerator

Experimental Setup

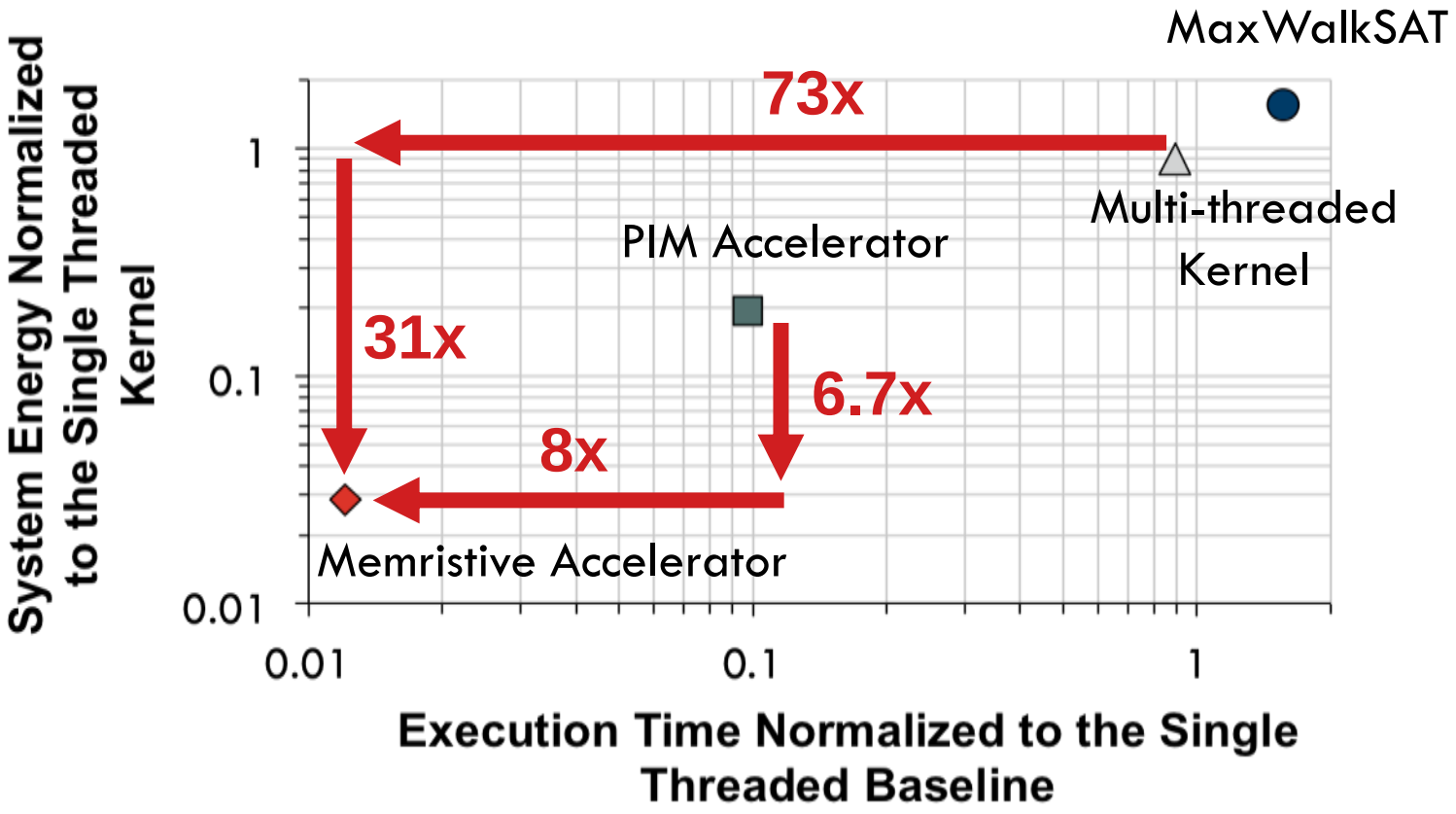
- PIM-based Hardware Accelerator



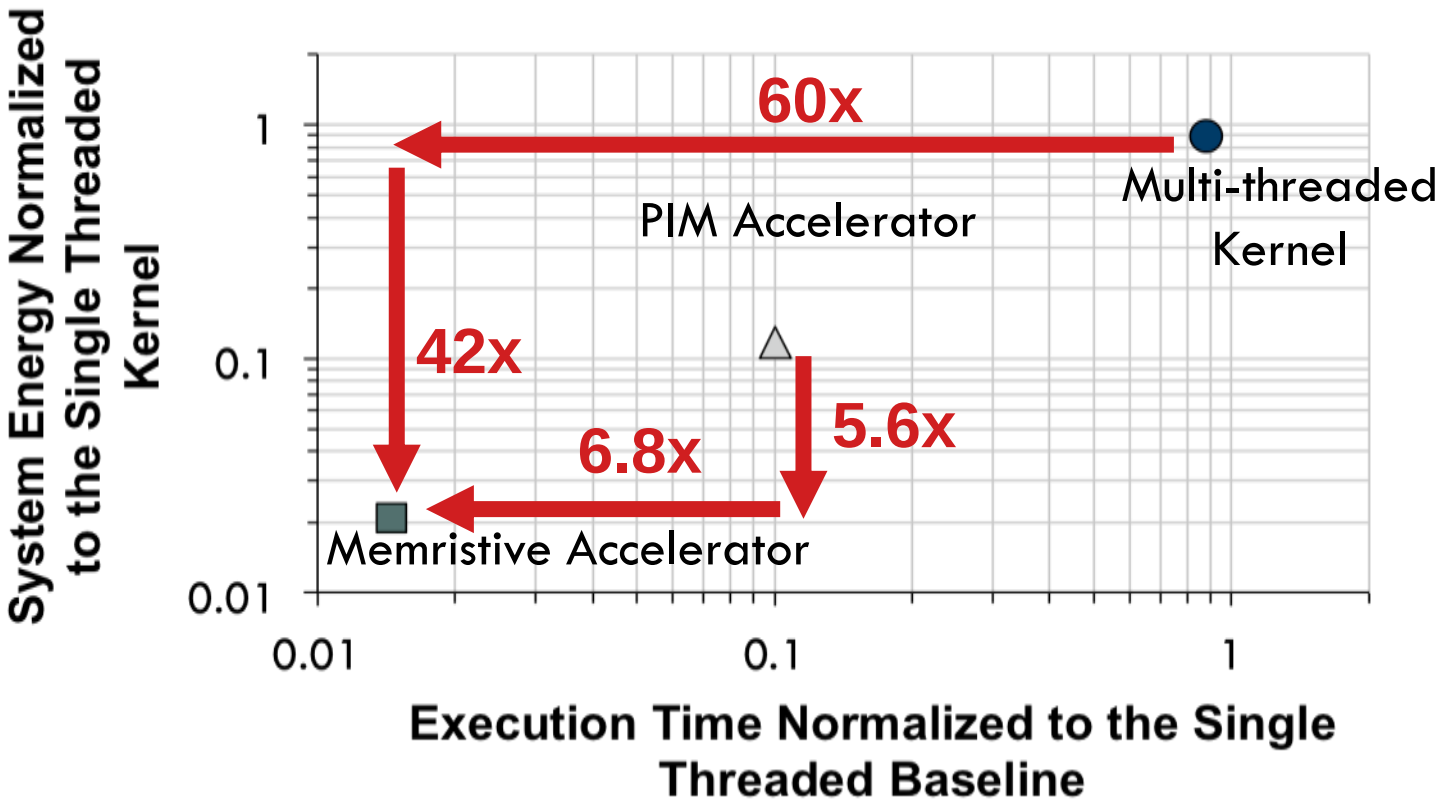
Maximum Cut


Semi-definite Programming
38x
Multi-threaded Kernel
PIM Accelerator
16x
3.9x
4.7x
Memristive Accelerator
System Energy Normalized to the Single Threaded Kernel
1
0.1
0.01
0.01
0.1
1
Execution Time Normalized to the Single Threaded Baseline

Maximum SAT


MaxWalkSAT
73x
Multi-threaded
Kernel
PIM Accelerator
31x
6.7x
8x
Memristive Accelerator
System Energy Normalized
to the Single Threaded
Kernel
1
0.1
0.01
0.01
0.1
1
Execution Time Normalized to the Single
Threaded Baseline

Deep Learning



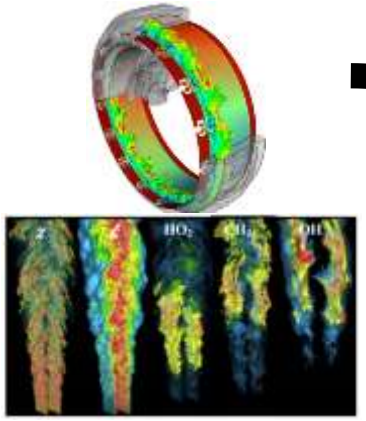
UNIVERSITY of ROCHESTER

Beyond Machine Learning

In-situ computation for iterative linear solvers

Linear Algebra Is Everywhere

- Sparse linear systems are at the heart of data science
- Contemporary supercomputers are inefficient at solving sparse linear systems



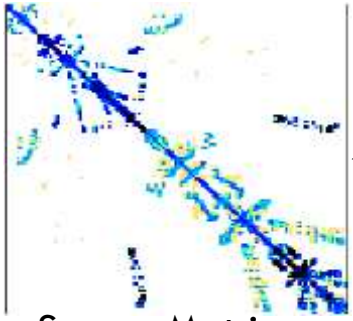
Physical Simulations

Utilize

$$\rho\left(\frac{\partial u_i}{\partial t} + u_j\frac{\partial u_i}{\partial x_j}\right) = -\frac{\partial p}{\partial x_i} + \mu\frac{\partial^2 u_i}{\partial x_j \partial x_j} + f_i$$

Differential Equations

Are discretized into



Sparse Matrices

and solved on

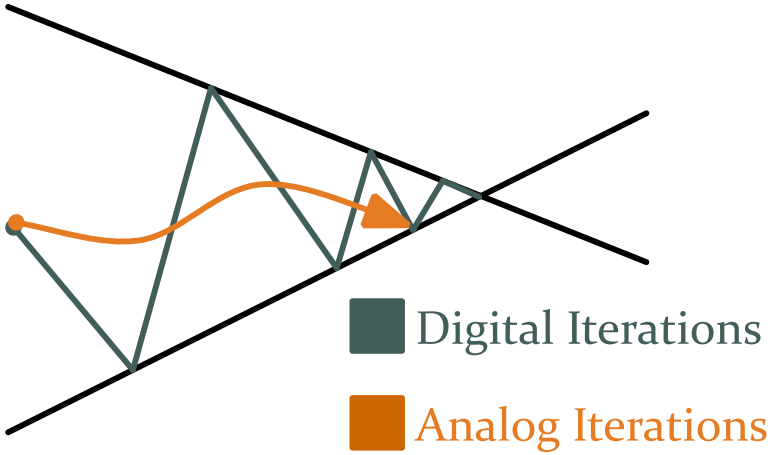
Supercomputers

The Opportunities and Challenges of Exascale Computing, 2010.
http://en.wikipedia.org/wiki/Navier%E2%80%93Stokes_equations
http://www.cise.ufl.edu/research/sparse/matrices/HB/bcsstk13.html
http://info.circ.rochester.edu/BlueHive/System_Overview.html

Our Approach to Iterative Linear Solvers

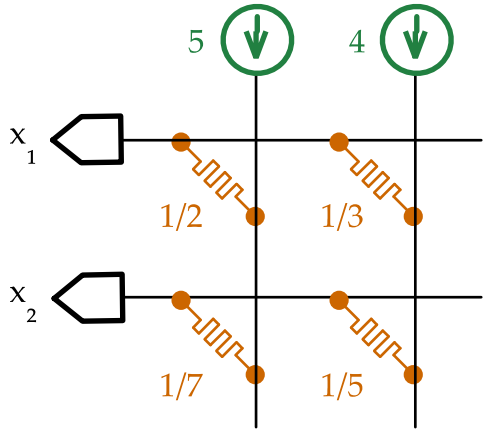
- Leverage memristive network to quickly obtain an approximate solution
- Use approximate solution to seed conventional iterative solver

Result: Faster solution with no loss of precision



Key Observations

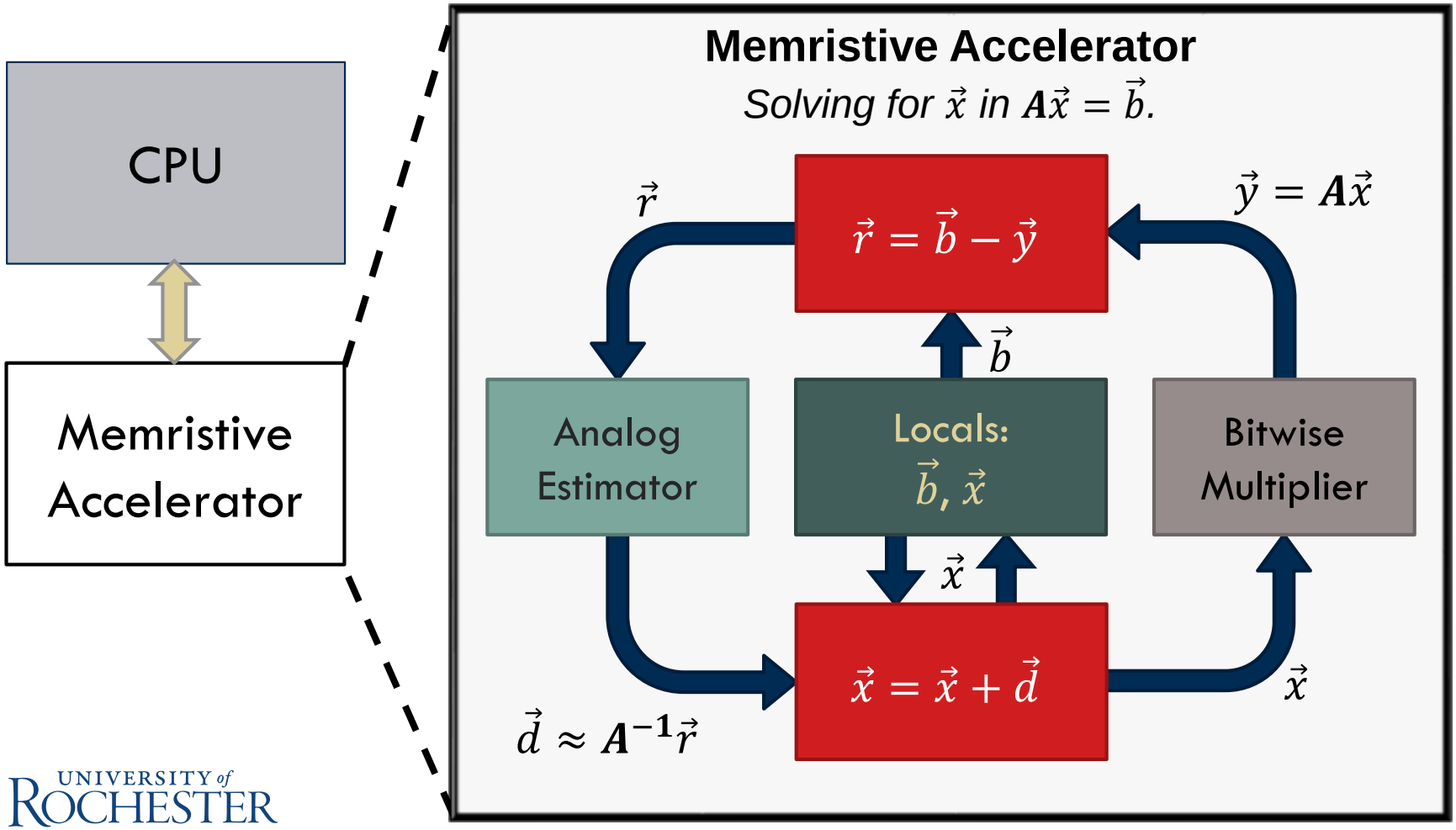
- Node voltages in a resistive network can be calculated by solving a linear system
- Conversely, any linear systems can be solved by emulation on a resistive network



$$2x_1 + 7x_2 = 5$$
$$3x_1 + 5x_2 = 4$$

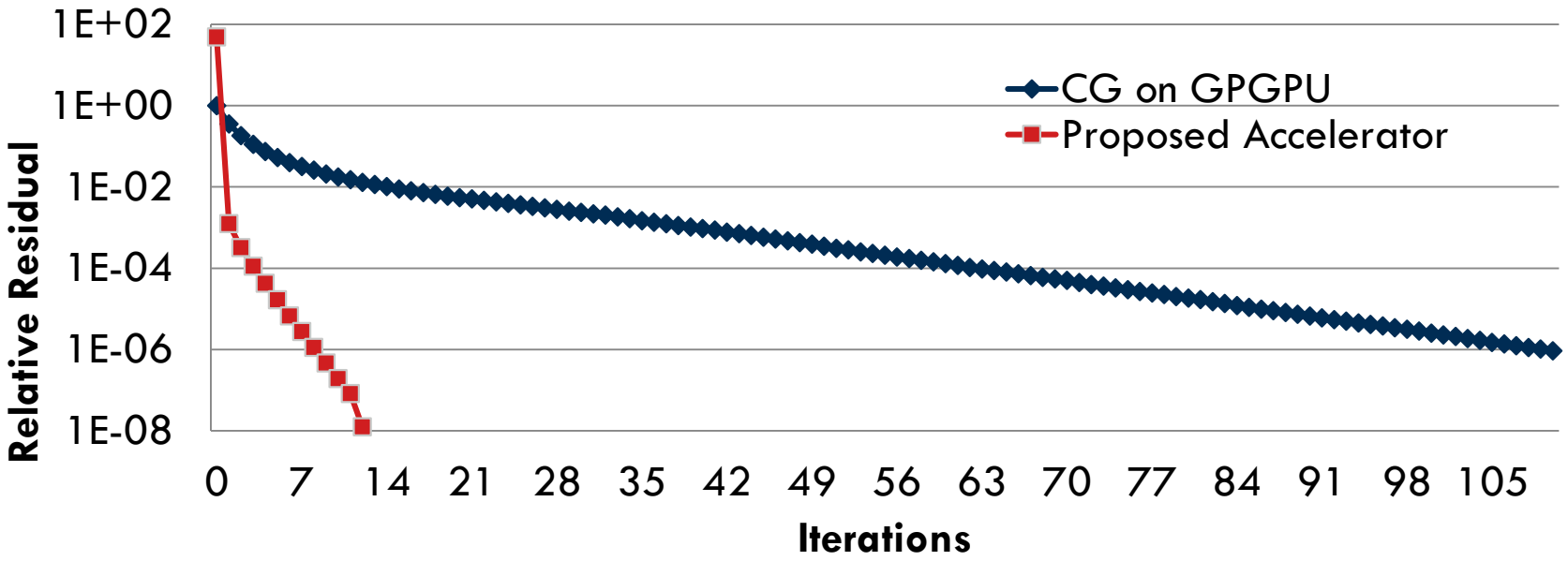
- Recently, it has become practical to build large-scale resistive networks using newly-developed memristor technology
  - Dense
  - Scalable
  - Programmable
  - Reliable

System Overview



Preliminary Results

- 1500x speedup as compared to a GPGPU
- Residuals decay faster and with fewer operations



UNIVERSITY of ROCHESTER

Summary

Memristors have potential beyond the memory hierarchy

- Emerging resistive memory devices can improve the energy efficiency and latency of important workloads
  - Machine learning
  - Optimization
  - Simulation & modeling

Memristive Accelerators for Data Intensive Computing: From Machine Learning to High Performance Linear Algebra

Engin Ipek

Department of Electrical and Computer Engineering
University of Rochester