

NEAR DATA PROCESSING

Mahdi Nazm Bojnordi

Assistant Professor

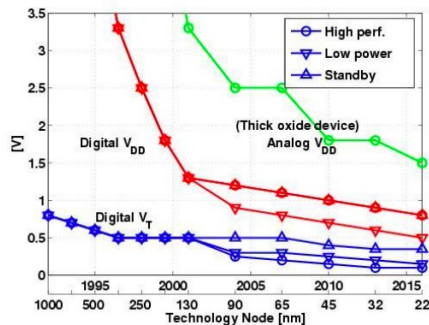
School of Computing

University of Utah

Data Processing Trends

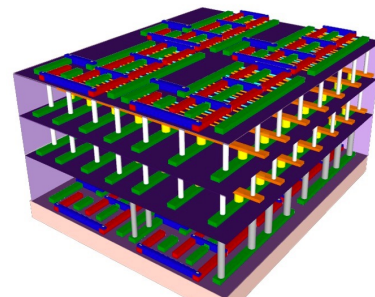
□ Why processing in memory ...

Scaling Limitations



End of Dennard scaling

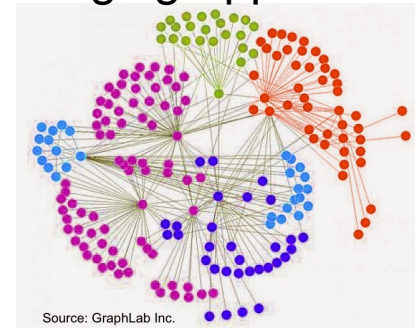
3D stacking, resistive memories, etc.



New Technologies

Near Data Processing

Emerging Applications



Source: GraphLab Inc.

Graph analytics, deep neural nets, etc.

[ref: GraphLab, ZME]

Requirements for Efficient NDP

- **Throughput**

- ▣ High processing throughput to match the high memory bandwidth



Near Data Processing

- **Power**

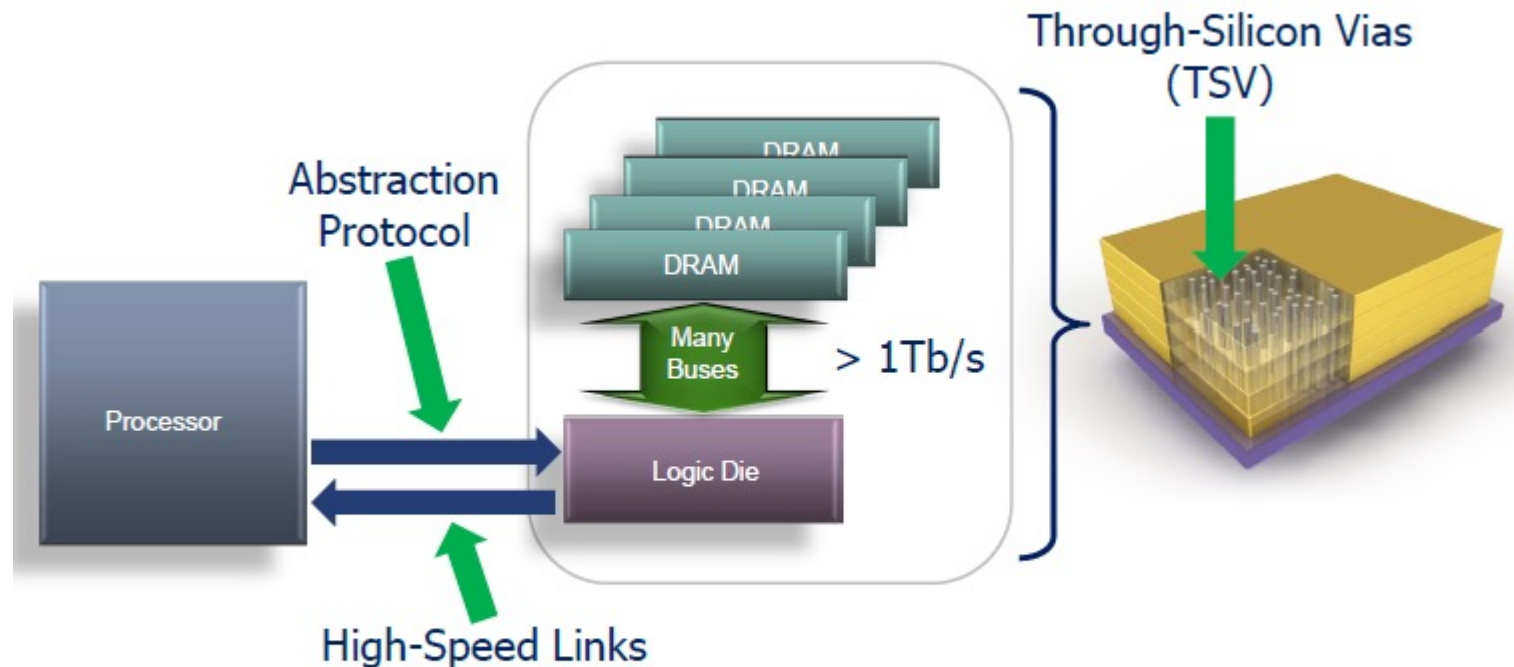
- ▣ Thermal constraints limit clock frequency

- **Flexibility**

- ▣ Must amortize manufacturing cost through reuse across apps

Memory Technologies

- HMC: hybrid memory cube



Notes: Tb/s = Terabits / second
HMC height is exaggerated

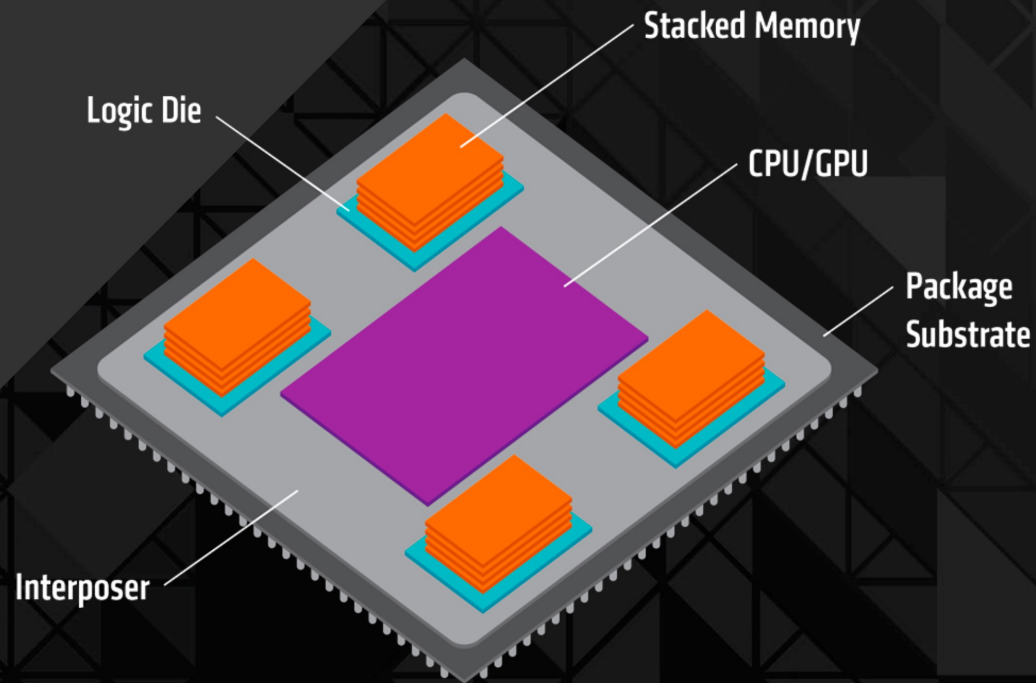
[ref: micron]

High Bandwidth Memory

THE INTERPOSER THE NEXT STEP IN INTEGRATION

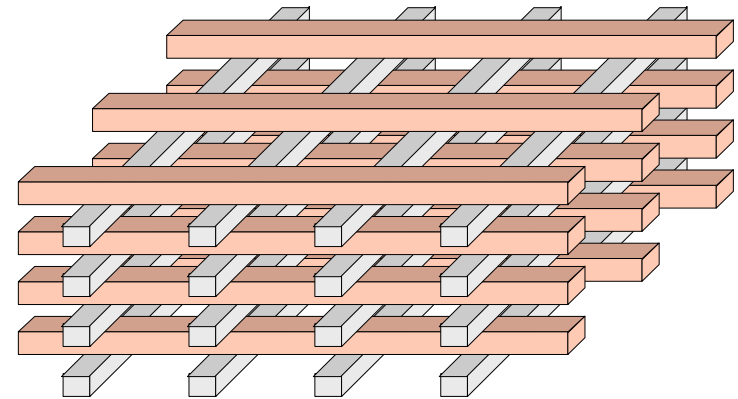
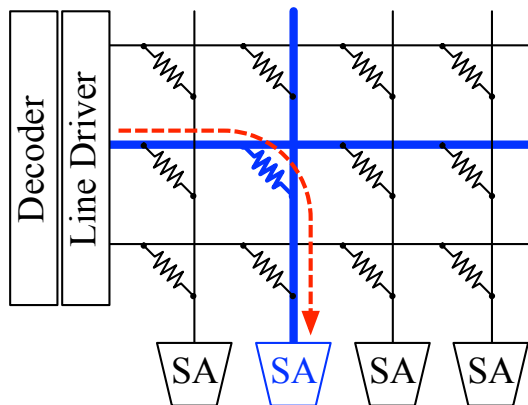


- ▲ Brings DRAM as close as possible to the logic die
- ▲ Improving proximity enables extremely wide bus widths
- ▲ Improving proximity simplifies communication and clocking
- ▲ Improving proximity greatly improves bandwidth per watt
- ▲ Allows for integration of disparate technologies such as DRAM
- ▲ AMD developed industry partnerships with ASE, Amkor & UMC to develop the first high-volume manufacturable interposer solution



Resistive Memory Technology

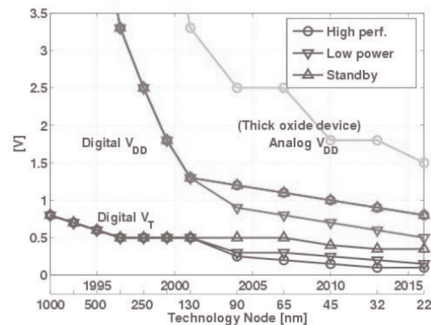
- 3D crosspoint promises virtually unlimited memory
 - ▣ Non-volatile
 - ▣ 10x higher density
 - ▣ Main limit for now 6GB/sec interface



Near Data Processing

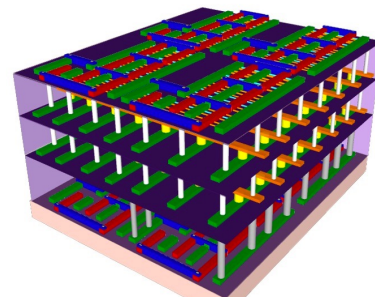
□ How to map applications to NDP technology?

Scaling Limitations



End of Denard scaling

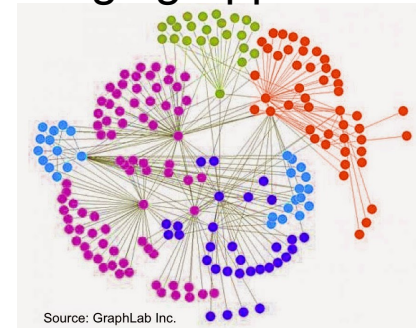
3D stacking,
resistive
memories, etc.



New Technologies

Near Data Processing

Emerging Applications

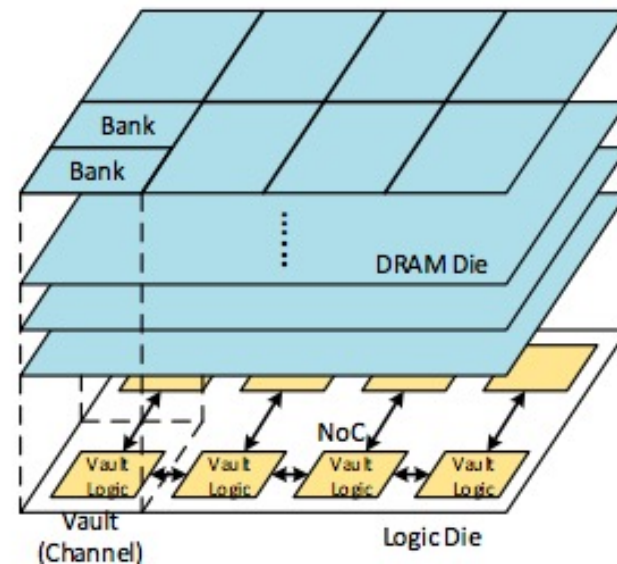
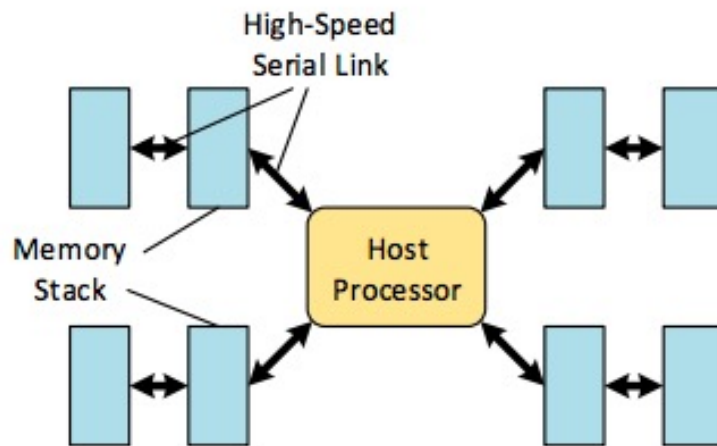


Graph analytics,
deep neural
nets, etc.

[ref: GraphLab, ZME]

Near Data Processing

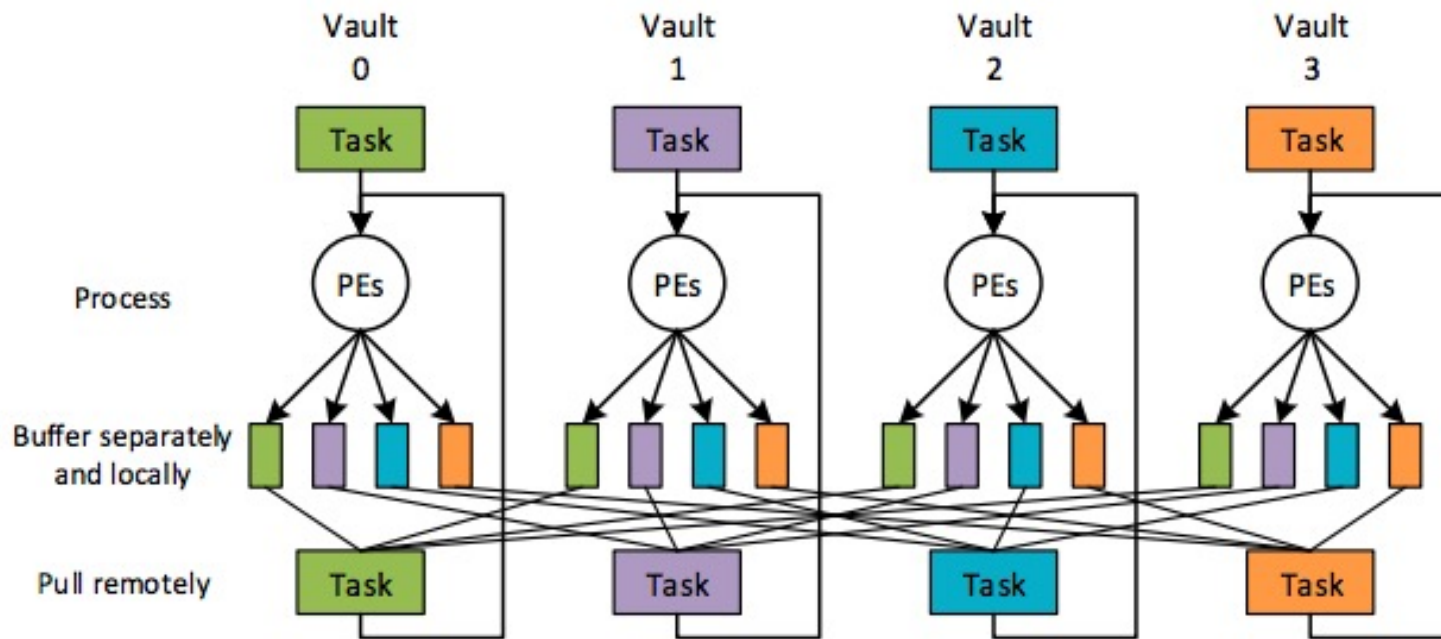
- Example
 - ▣ Stacks run memory intensive code
 - ▣ Multiple stacks linked to host processor (SerDes)



HMC

Design Challenges

- Communication within and across stacks
 - ▣ Fine-grained vs. coarse-grained blocks



Efficient NDP Architecture

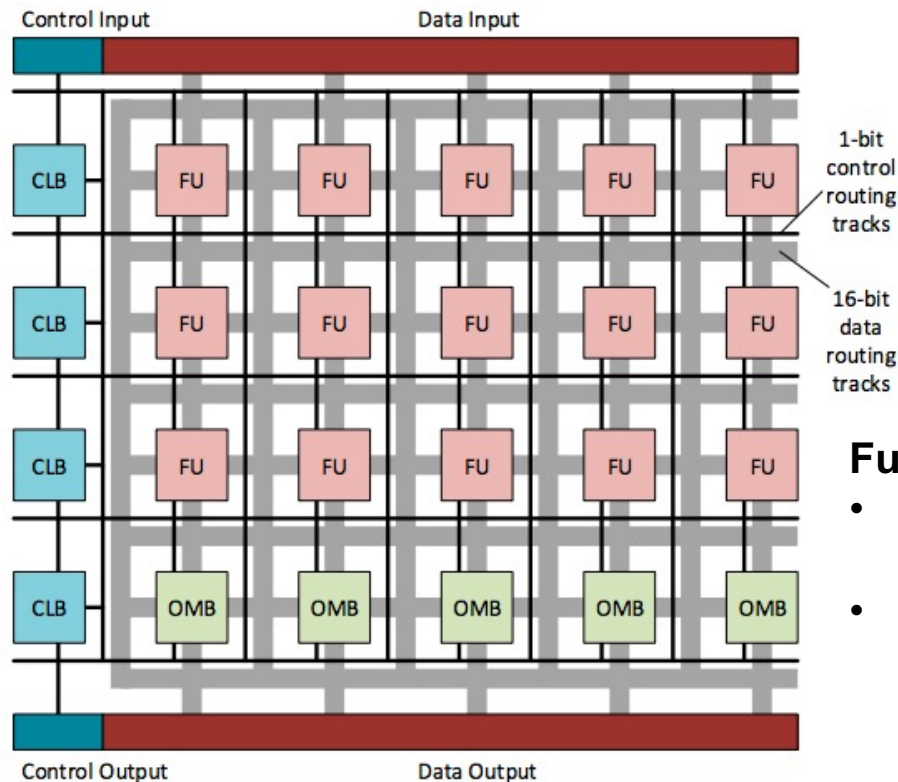
- Flexibility, area and power efficiency
- HRL: heterogeneous reconfigurable logic

Configurable Logic Block

- LUTs for embedded control logic
- Special functions: sigmoid, tanh, etc.

Output MUX Block

- Configurable MUXes (tree, cascading, parallel)
- Put close to output, low cost and flexible

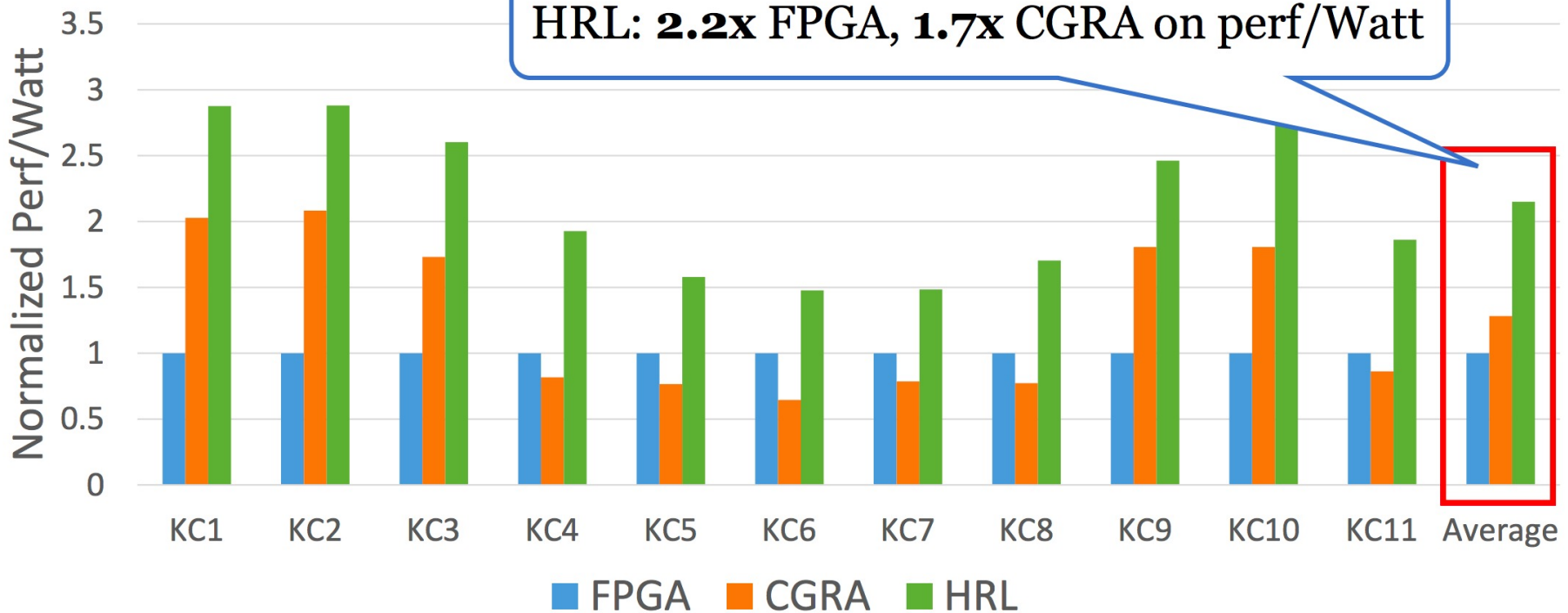


Functional Unit

- Efficient 48-bit arithmetic/logic ops
- Registers for pipelining and retiming

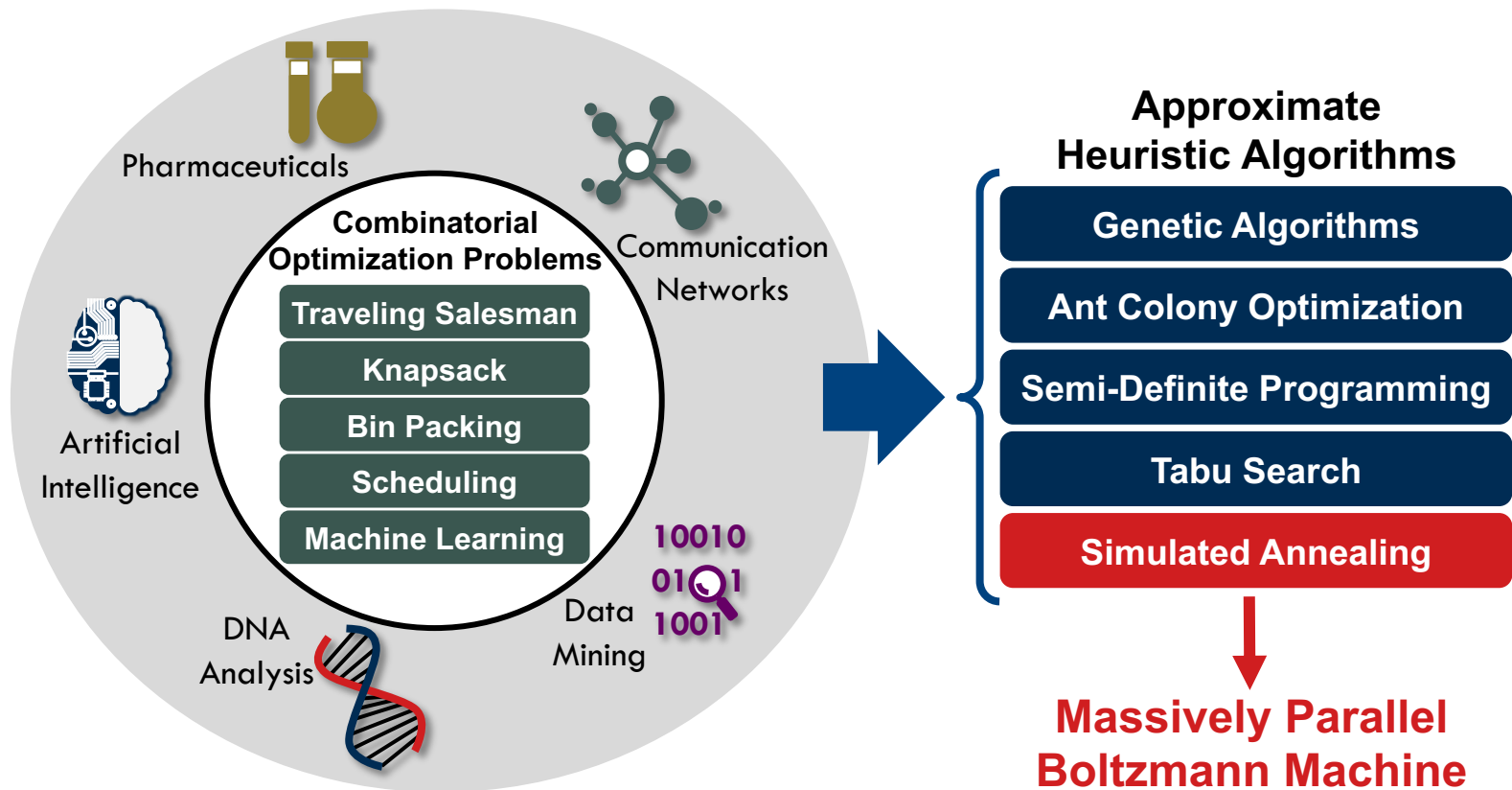
Power Efficiency

- 2.2x performance/Watt over FPGA
- Within 92% of ASIC performance



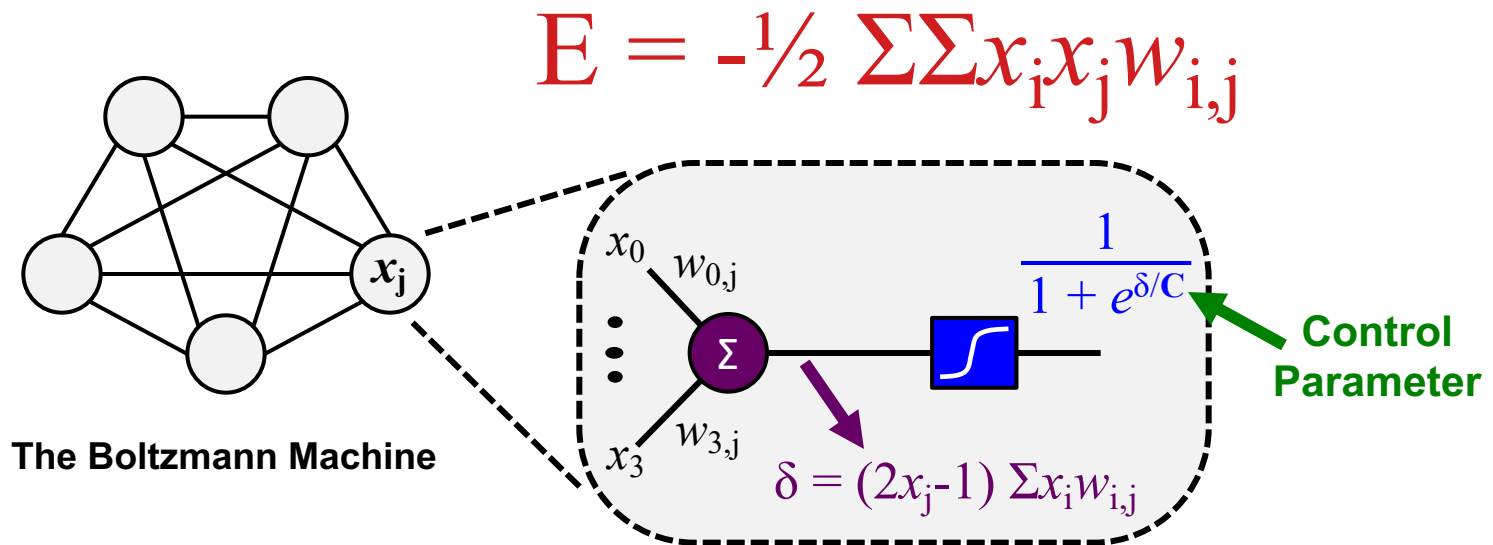
Combinatorial Optimization

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



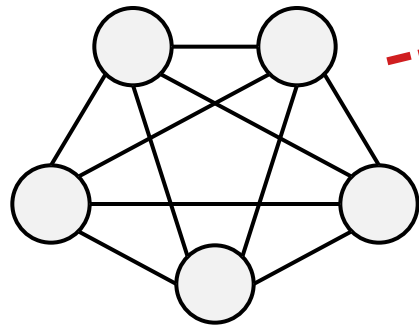
The Boltzmann Machine

- 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)**.



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.

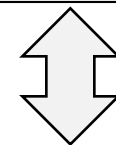
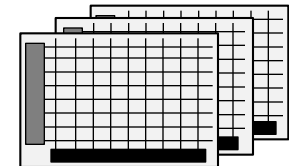


The Boltzmann Machine

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

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

Memory Arrays



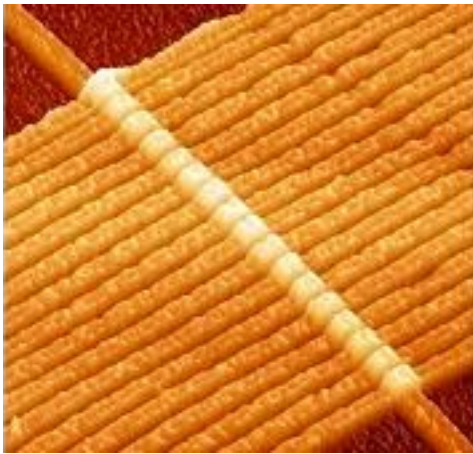
Data Movement



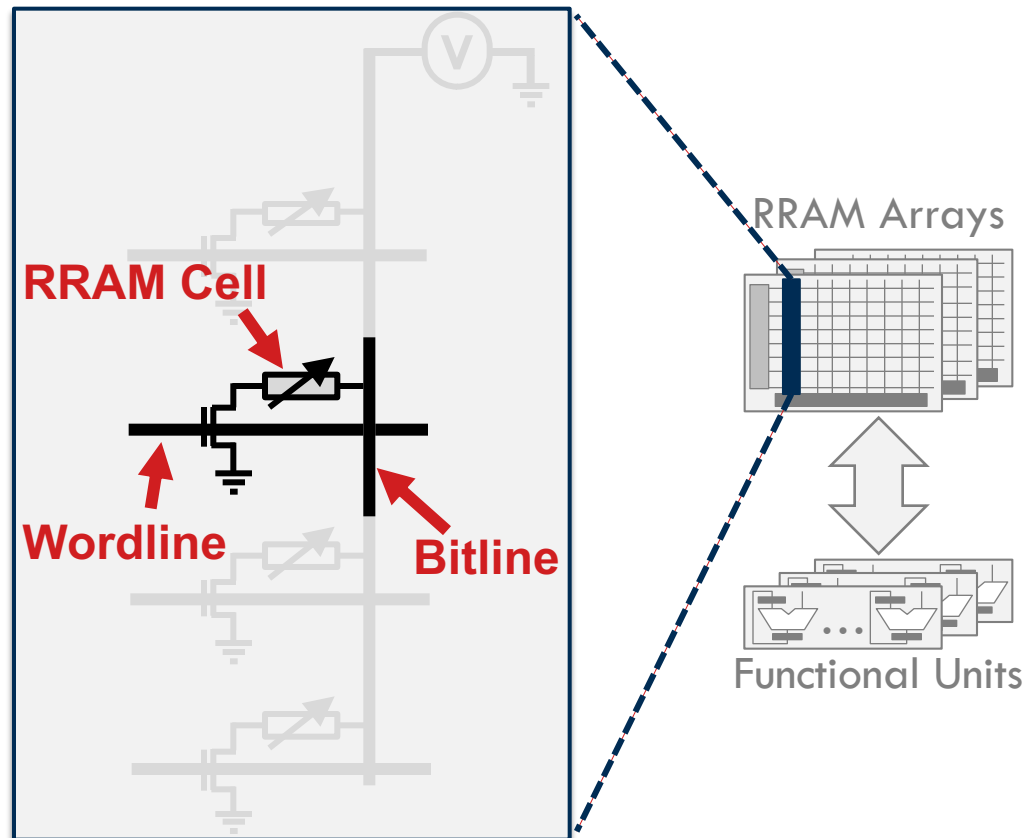
Functional Units

Resistive Random Access Memory

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



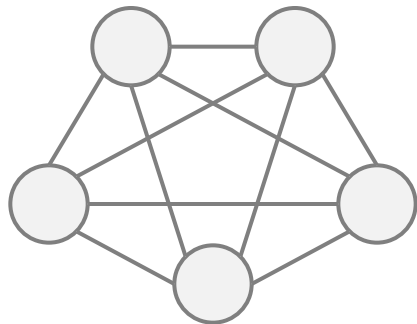
RRAM: Resistive RAM
(source: HP, 2009)



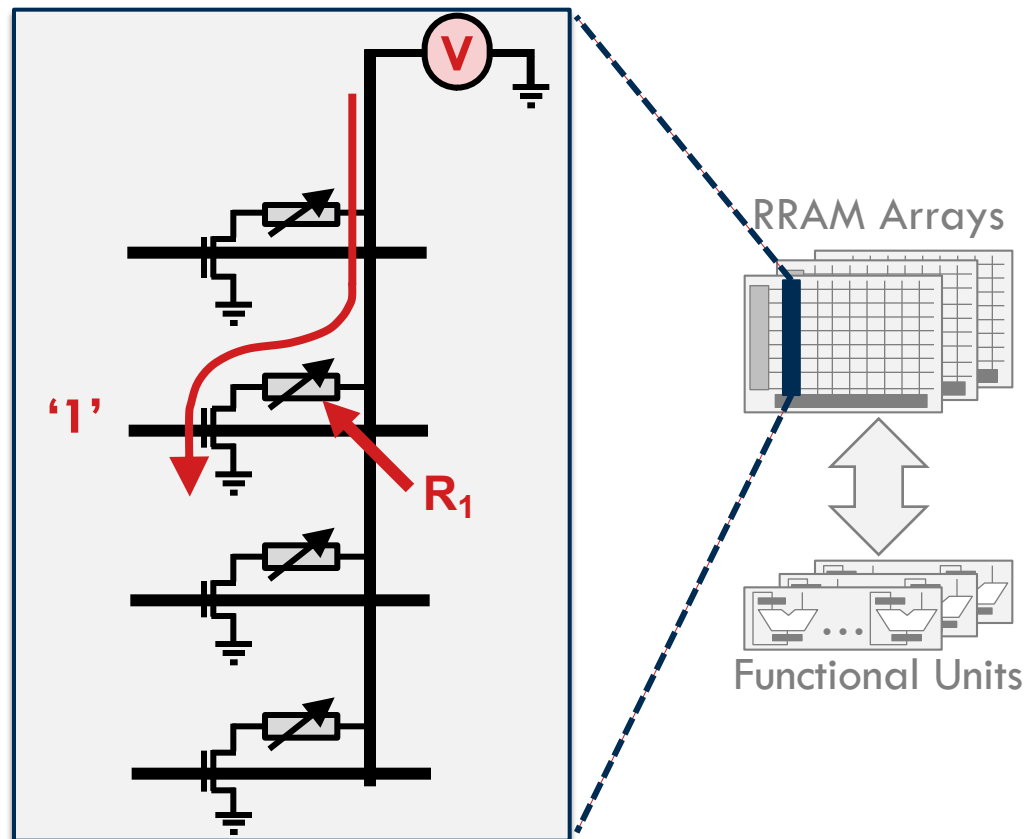
Resistive Random Access Memory

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

$$I = V/R_1$$



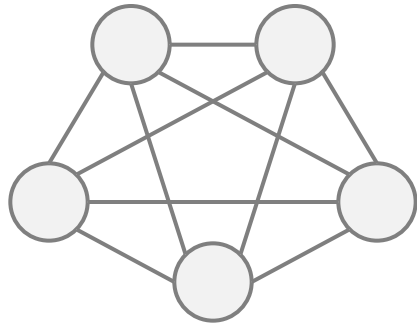
The Boltzmann Machine



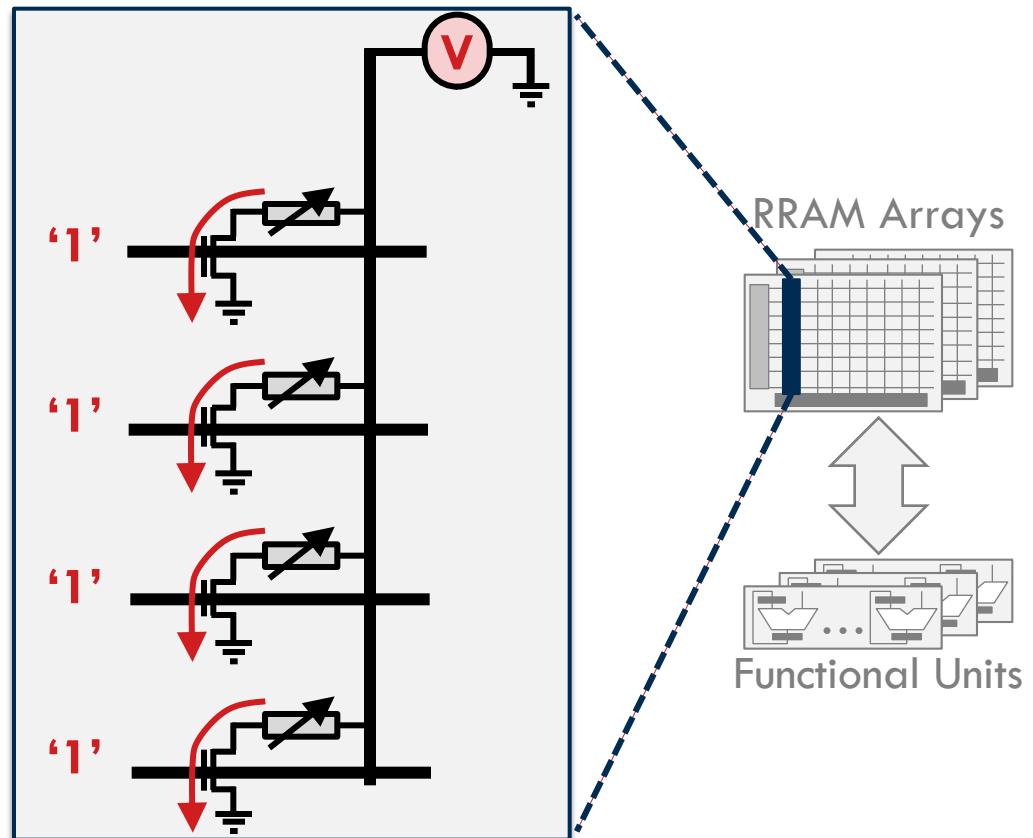
Memristive Boltzmann Machine

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

$$I = \sum V/R_i$$

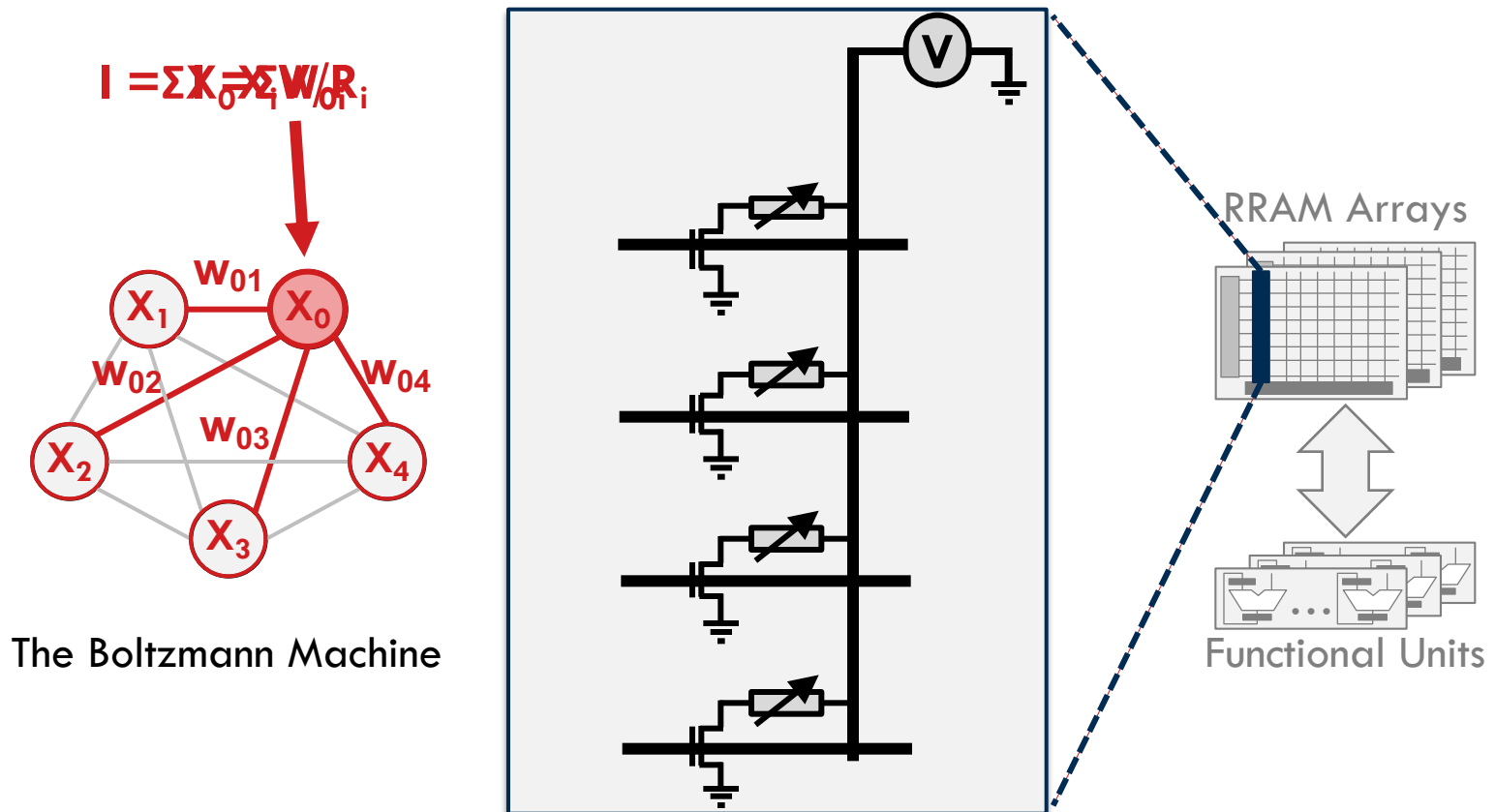


The Boltzmann Machine



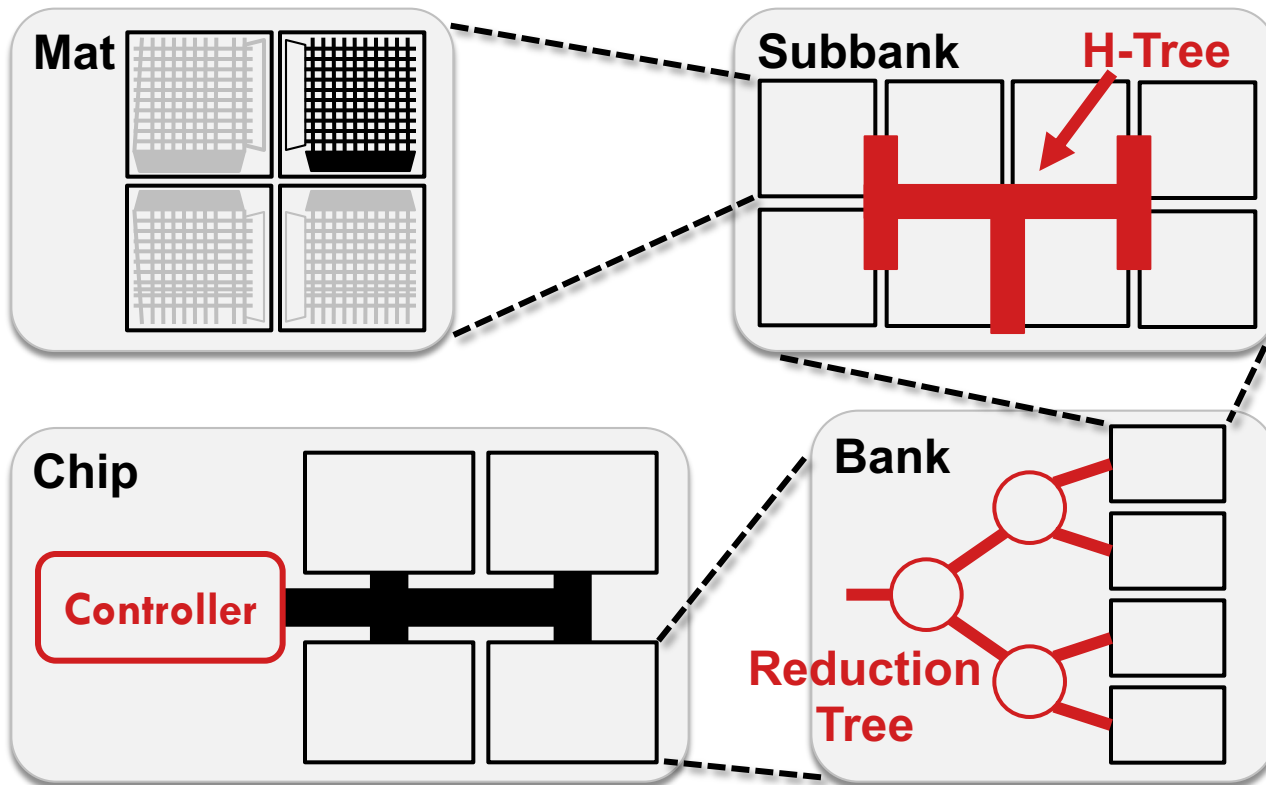
Memristive Boltzmann Machine

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



Chip Organization

- Hierarchical organization with configurable reduction tree is used to compute large sum of product.



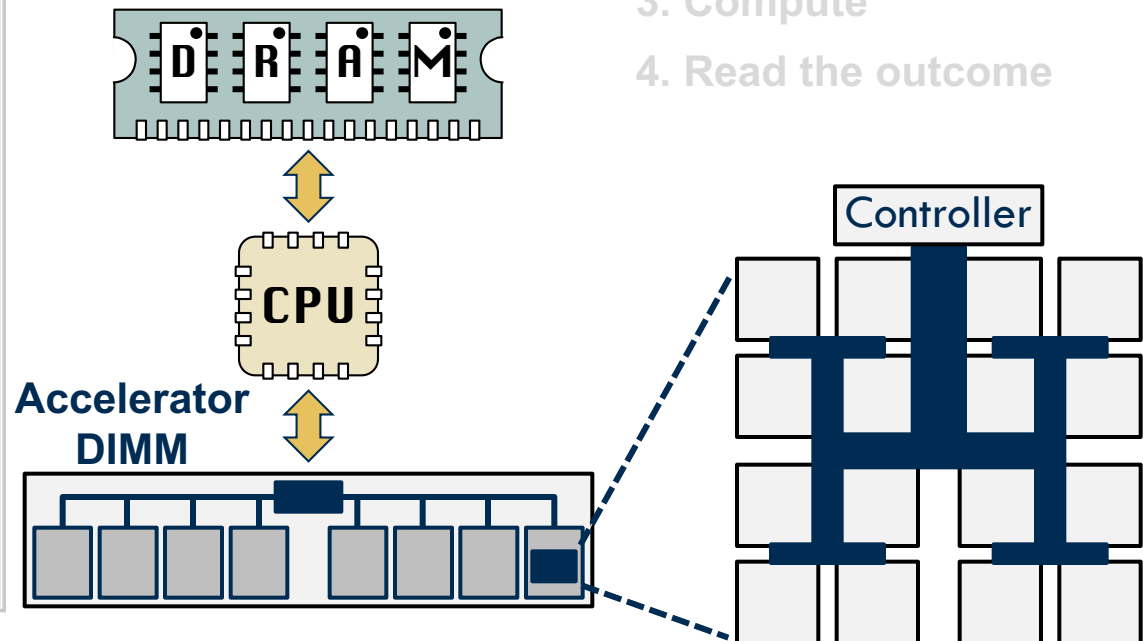
System Integration

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
4. Read the outcome



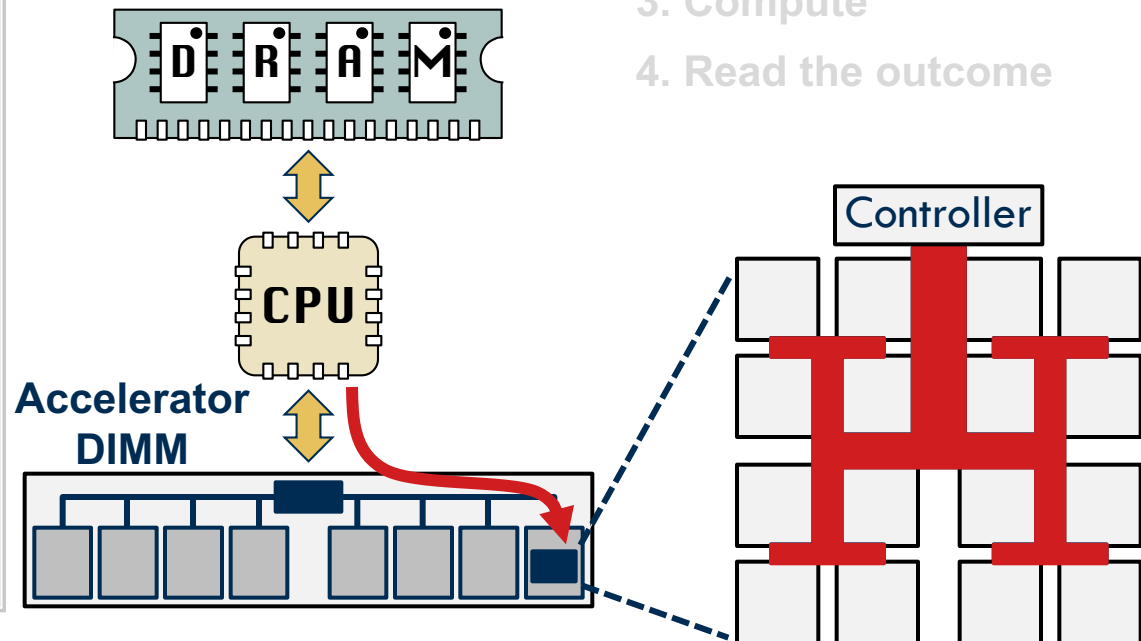
System Integration

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
4. Read the outcome



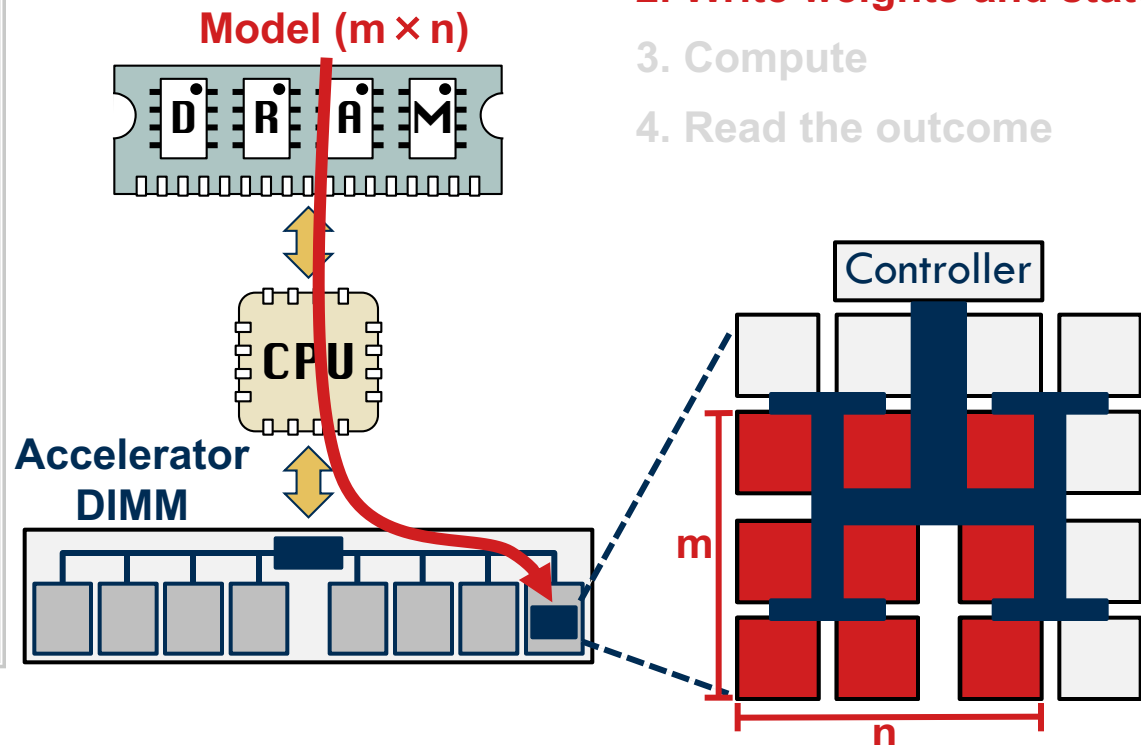
System Integration

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
4. Read the outcome



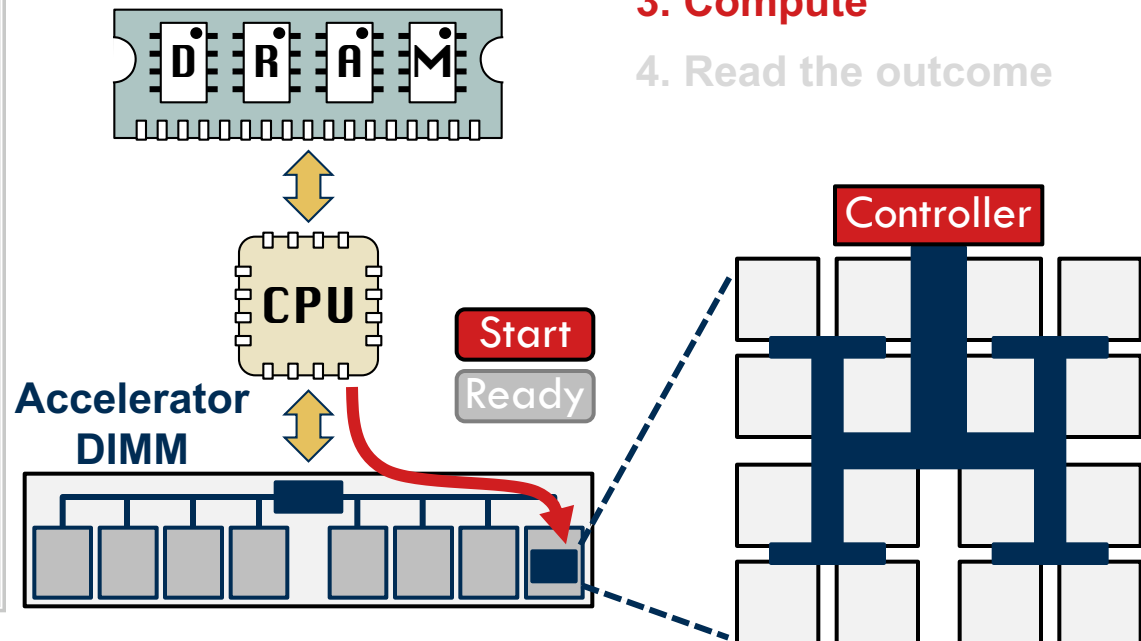
System Integration

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
4. Read the outcome



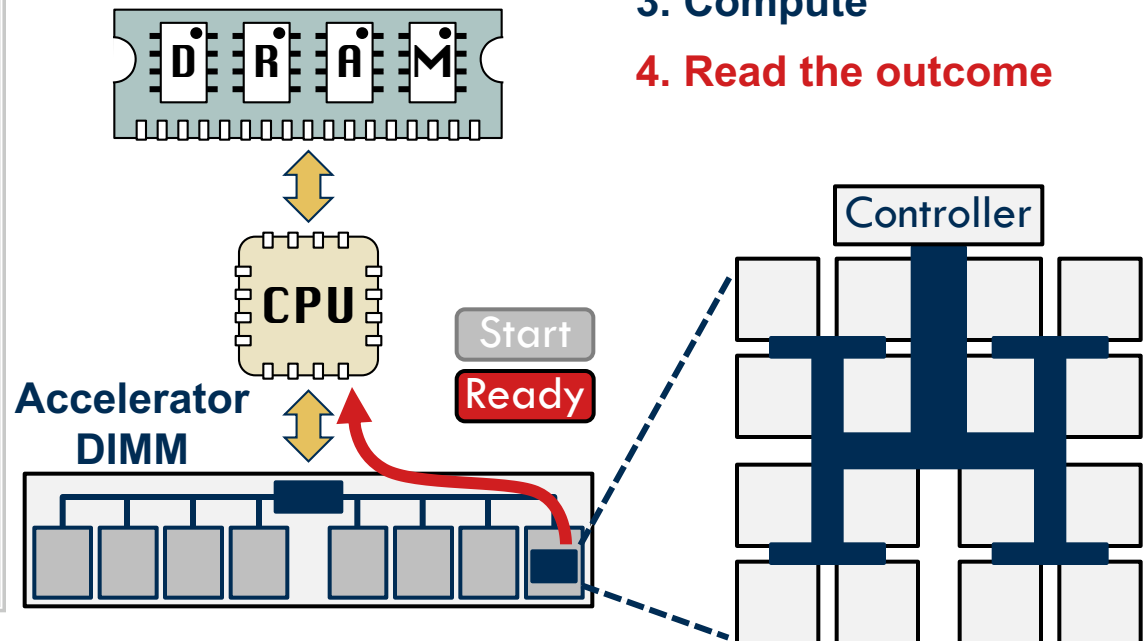
System Integration

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
4. Read the outcome



Summary of Results

