

# NEAR DATA PROCESSING

Mahdi Nazm Bojnordi

Assistant Professor

School of Computing

University of Utah

# Requirement for Your Presentations

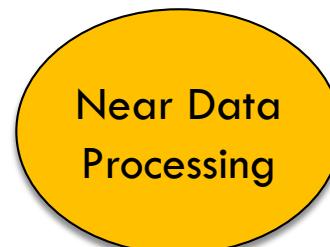
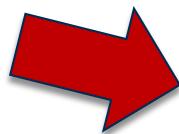
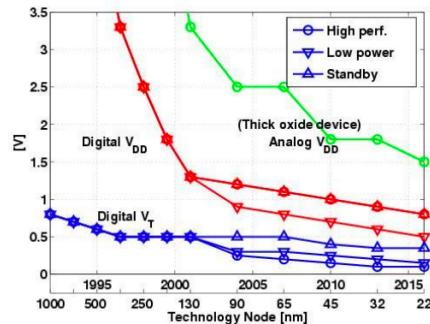
---

- Prepare for exactly 20m talk followed by 5m Q&A
- Required components in your presentation
  - ▣ Summary of the work
    - Clearly present key ideas, mechanisms, and results
  - ▣ Strength and weaknesses
    - Slides highlighting the strengths and weaknesses
  - ▣ Discussion
    - Future research directions
    - Alternative ways to solving the problem

# Data Processing Trends

## □ Why processing in memory ...

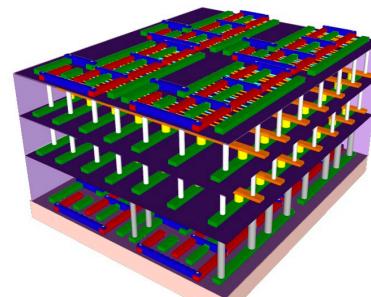
### Scaling Limitations



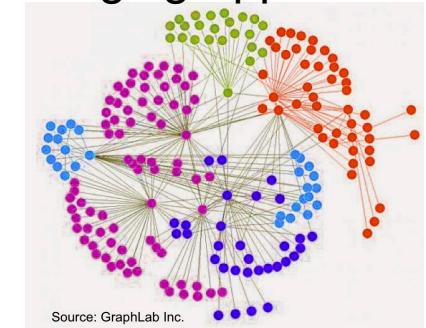
End of Denard scaling

New Technologies

3D stacking,  
resistive  
memories, etc.



### Emerging Applications



Graph analytics,  
deep neural  
nets, etc.

[ref: GraphLab, ZME]

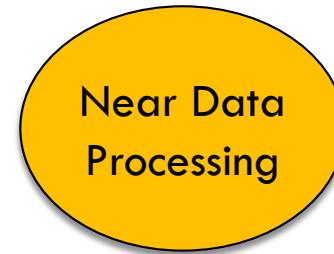
# Requirements for Efficient NDP

## □ Throughput

- ▣ High processing throughput to match the high memory bandwidth

## □ Power

- ▣ Thermal constraints limit clock frequency

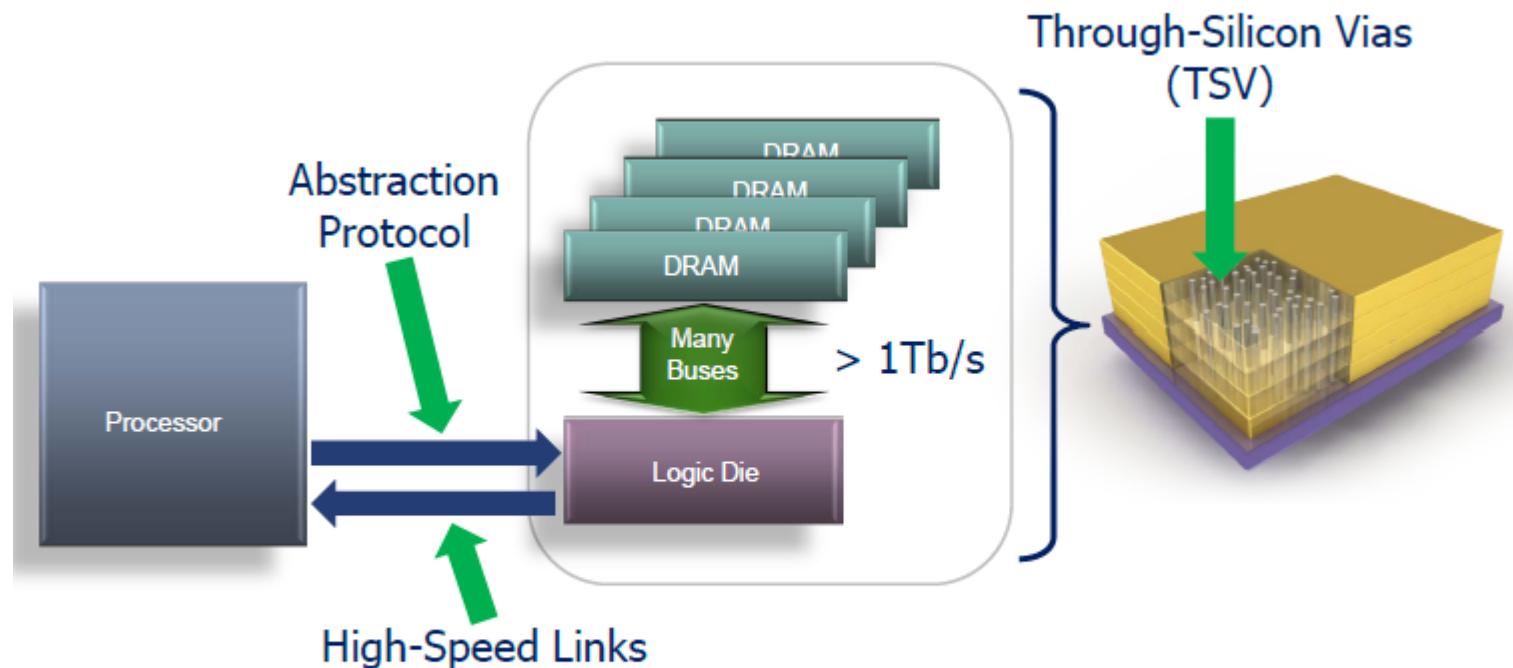


## □ Flexibility

- ▣ 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 Buses

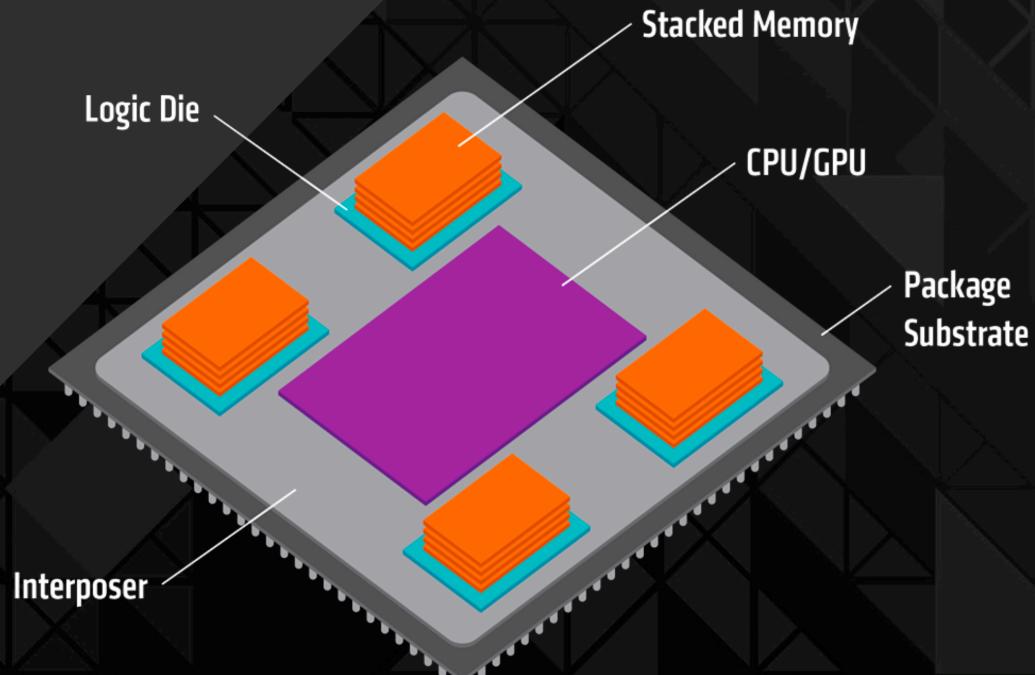
- Current DDR4 maxes out at **25.6 GB/sec**
- High Bandwidth Memory (HBM) led by AMD and NVIDIA
  - Supports 1,024 bit-wide bus at **125 GB/sec**
- Hybrid Memory Cube (HMC) consortium led by Intel
  - Claimed that **400 GB/sec** possible
- Both based on stacked memory chips
  - Limited capacity (**won't replace DRAM**), but much higher than on-chip caches

# High Bandwidth Memory

## THE INTERPOSER THE NEXT STEP IN INTEGRATION

AMD

- ▶ 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

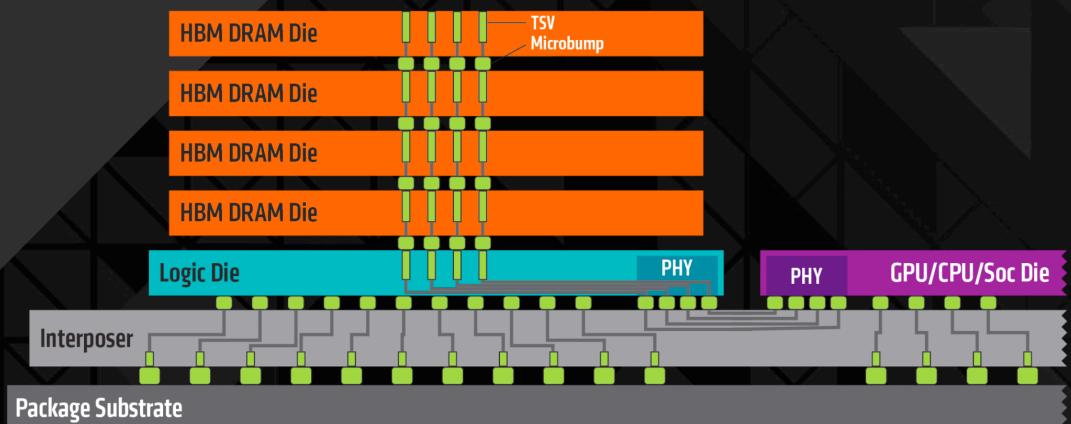


# High Bandwidth Memory

## HIGH-BANDWIDTH MEMORY DRAM BUILT FOR AN INTERPOSER

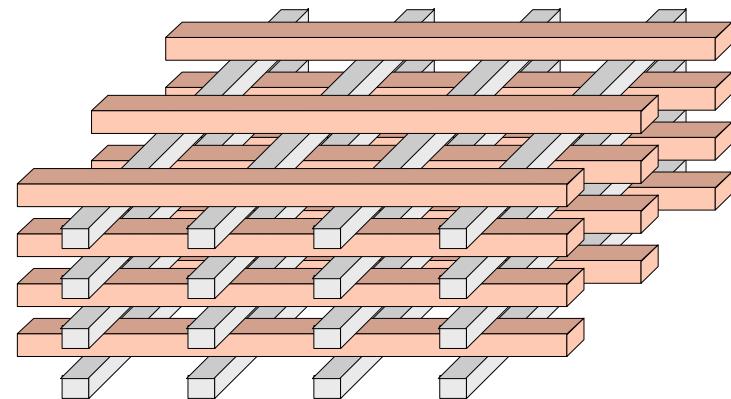
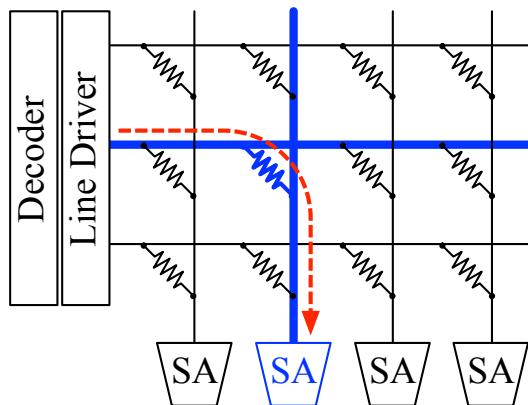
AMD

- ▲ A new type of memory chip with low power consumption and an ultra-wide bus width
- ▲ Many of those chips stacked vertically like floors in a skyscraper
- ▲ New interconnects, called “through-silicon vias” (TSVs) and “µbumps”, connect one DRAM chip to the next
- ▲ TSVs and µbumps also used to connect the SoC/GPU to the interposer
- ▲ AMD and SK Hynix partnered to define and develop the first complete specification and prototype for HBM



# Resistive Memory Technology

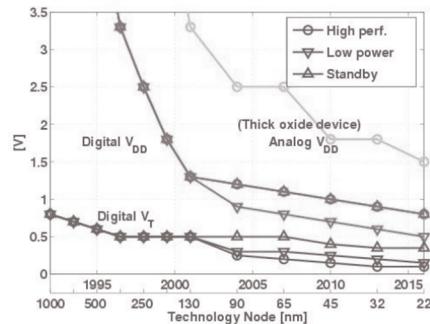
- 3D crosspoint promises virtually unlimited memory
  - ▣ Non-volatile
  - ▣ 10x higher density
  - ▣ Main limit for now 6GB/sec interface



# Near Data Processing

## □ How to map applications to NDP technology?

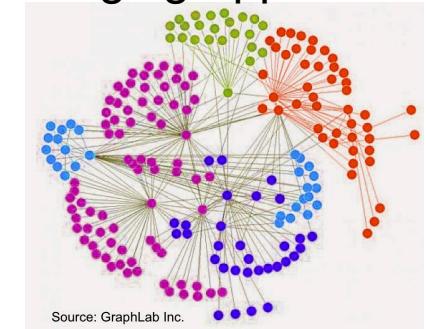
### Scaling Limitations



Near Data Processing



### Emerging Applications

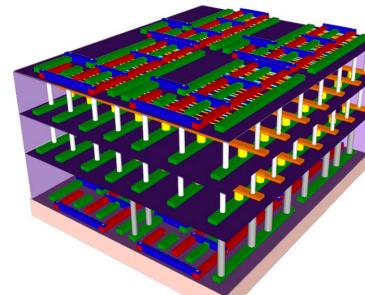


End of Denard scaling



New Technologies

3D stacking,  
resistive  
memories, etc.

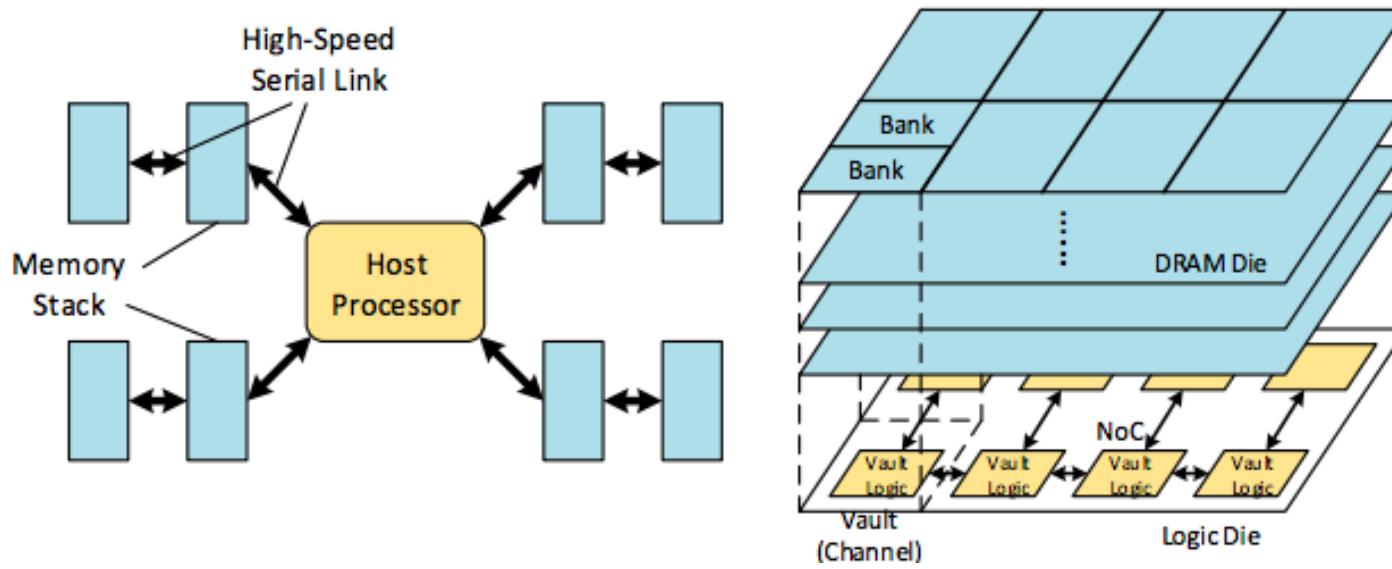


Graph analytics,  
deep neural  
nets, etc.

[ref: GraphLab, ZME]

# Near Data Processing

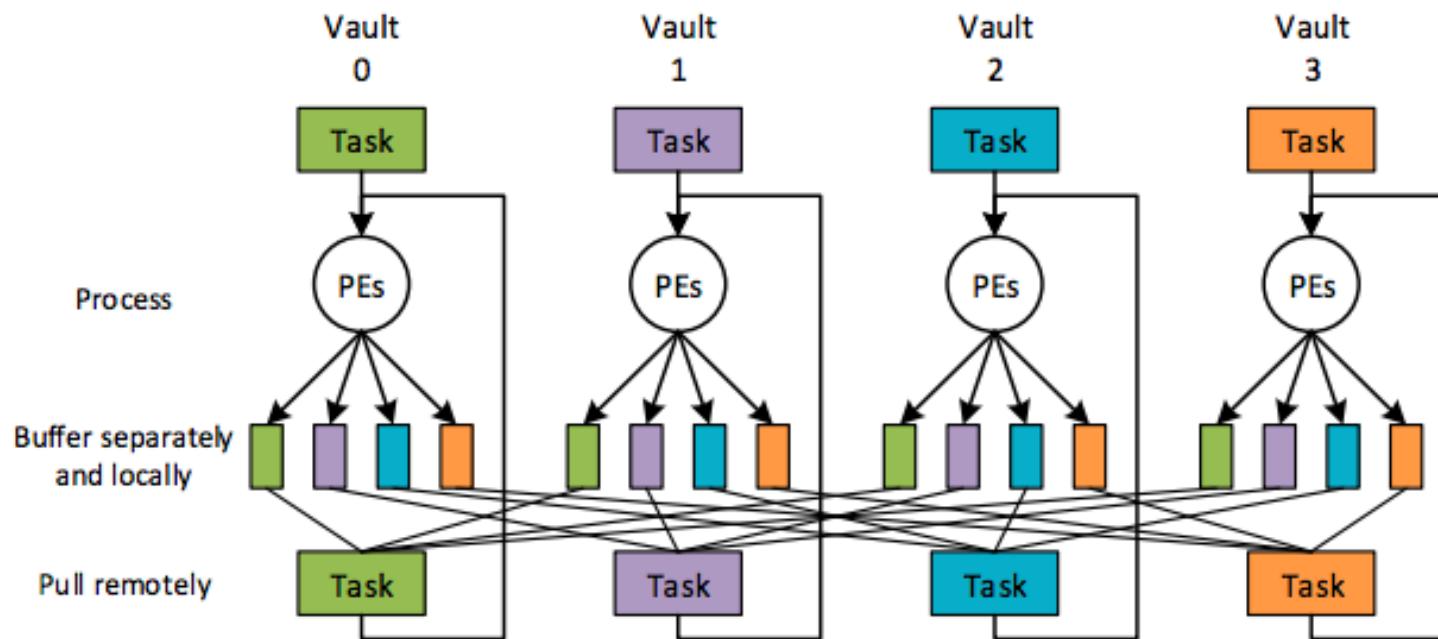
- Example
  - Stacks run memory intensive code
  - Multiple stacks linked to host processor (SerDes)



HMC

# Design Challenges

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



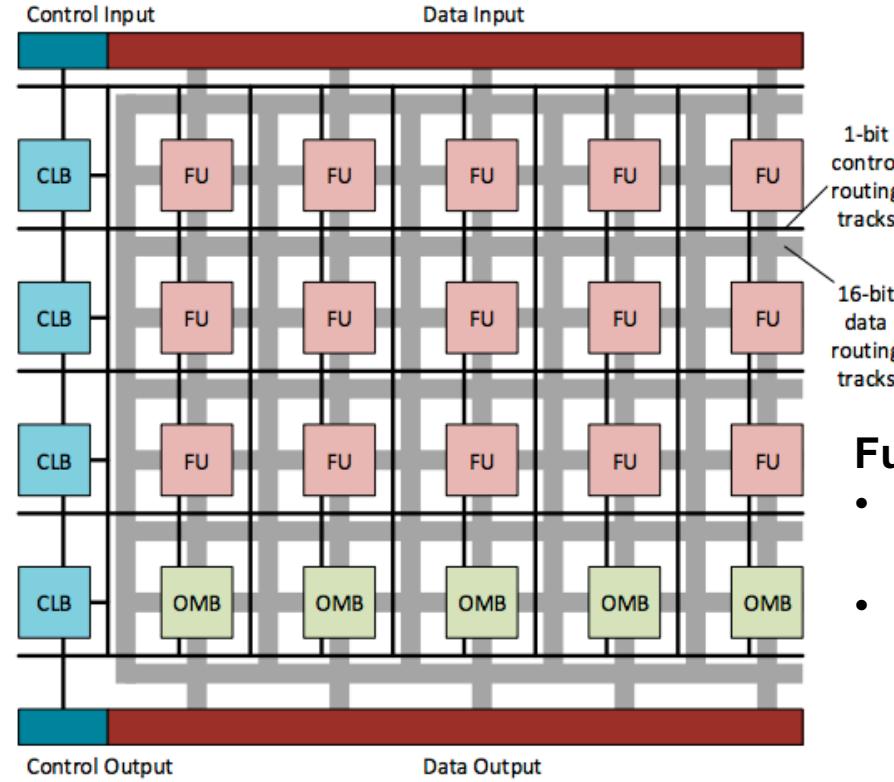
[PACT'15]

# 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

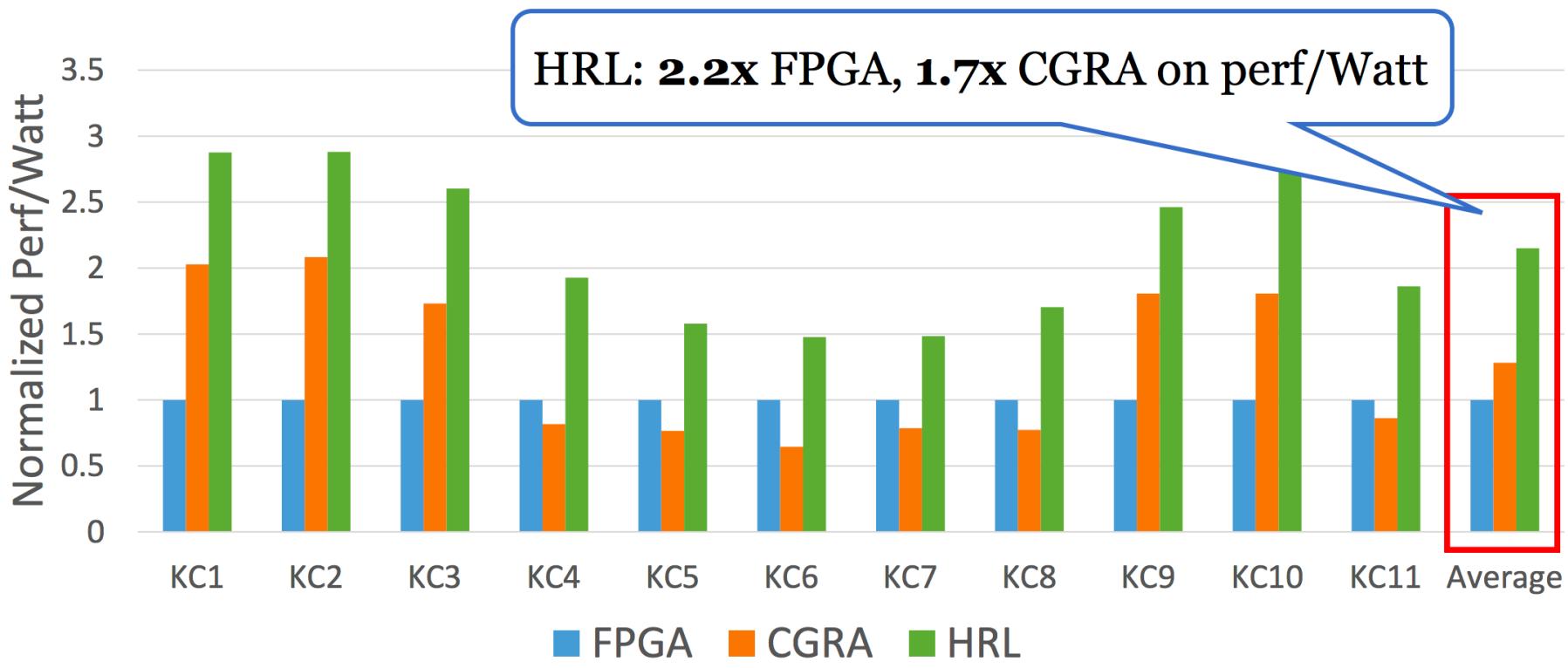
- Configurable MUXes (tree, cascading, parallel)
- Put close to output, low cost and flexible

## Functional Unit

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

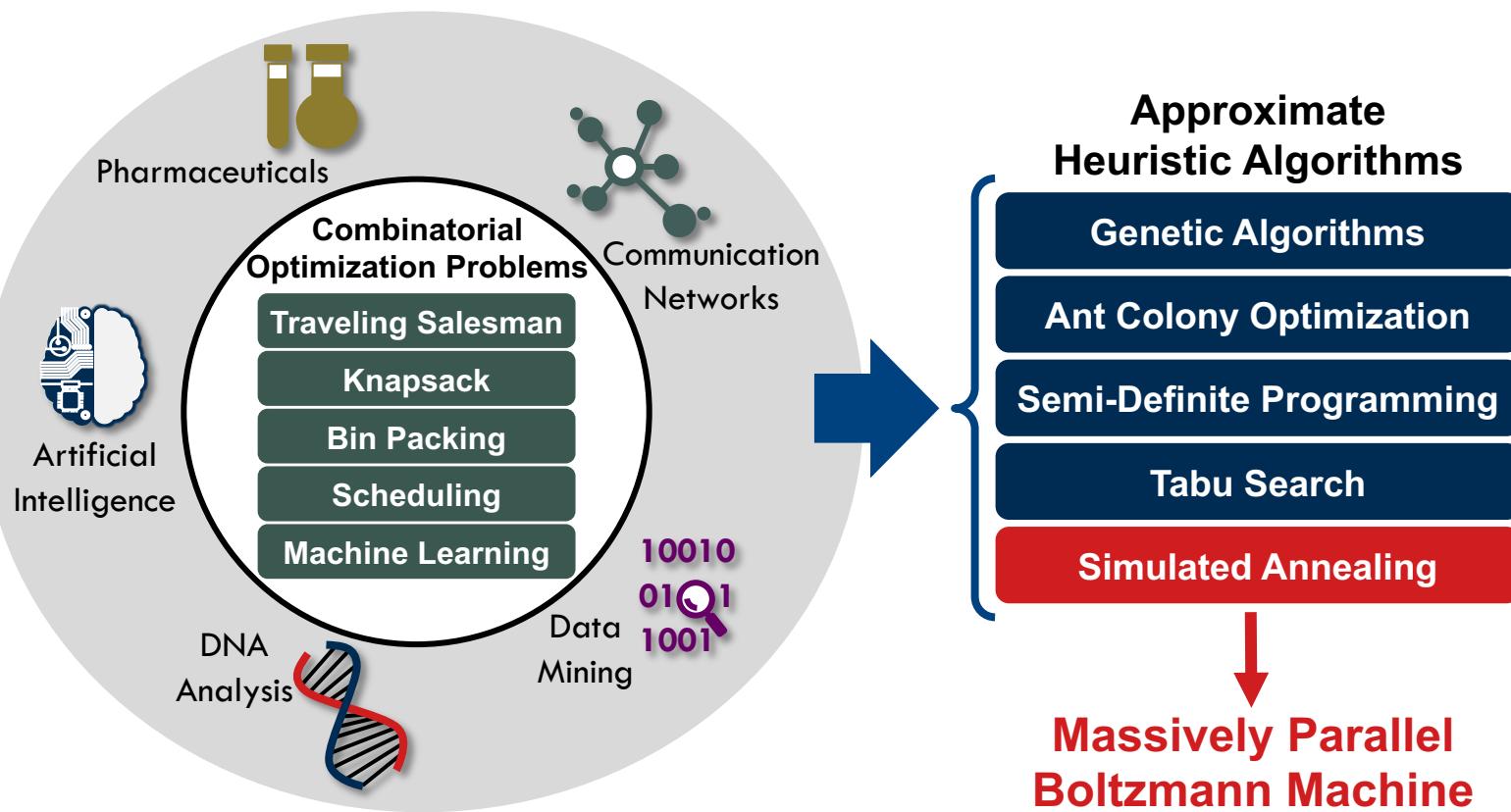
# Power Efficiency

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



# Combinatorial Optimization

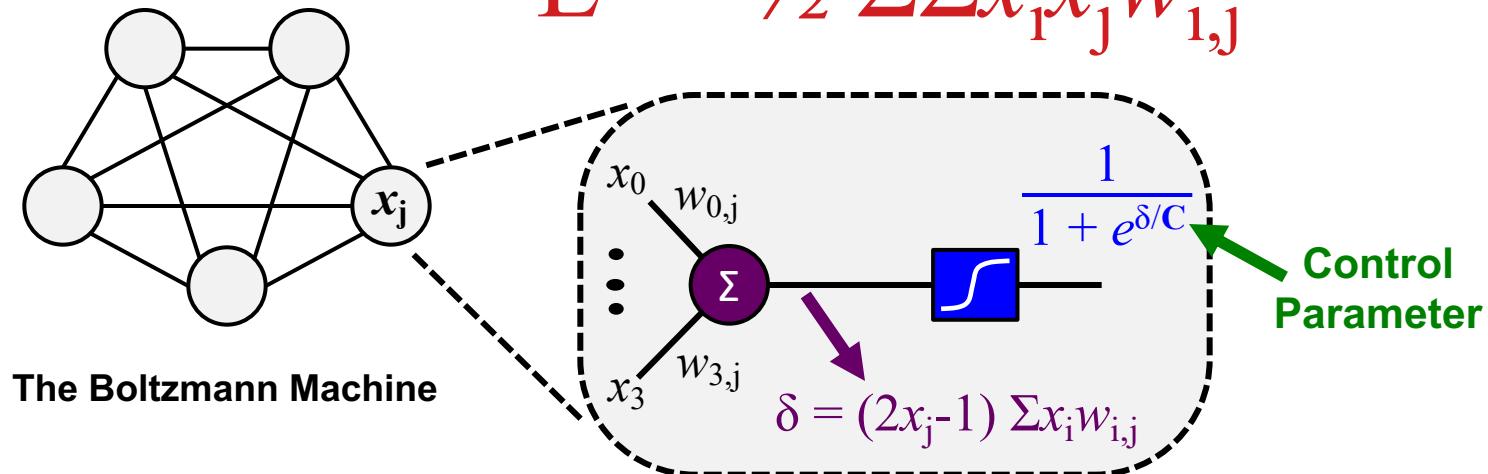
- Numerous critical problems in science and engineering can be cast within the combinatorial optimization framework.



# The Boltzmann Machine

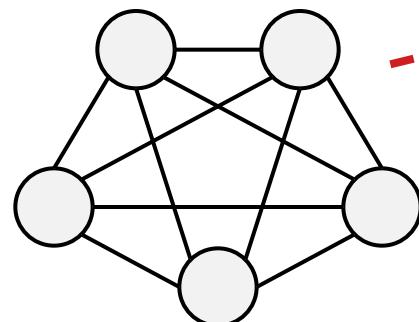
- Two-state units connected with real-valued edge weights form a stochastic neural network.
- **Goal:** iteratively update the state or weight variables to minimize the **network energy (E)**.

$$E = -\frac{1}{2} \sum \sum x_i x_j w_{i,j}$$



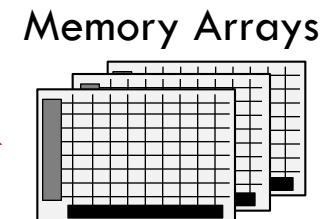
# Computational Model

- 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.

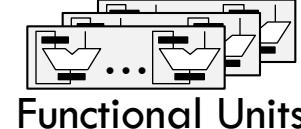


The Boltzmann Machine

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



Data Movement

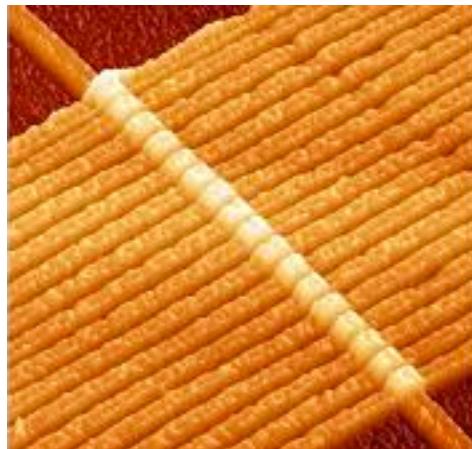


$$\Sigma, \times, \frac{1}{1 + e^x}$$

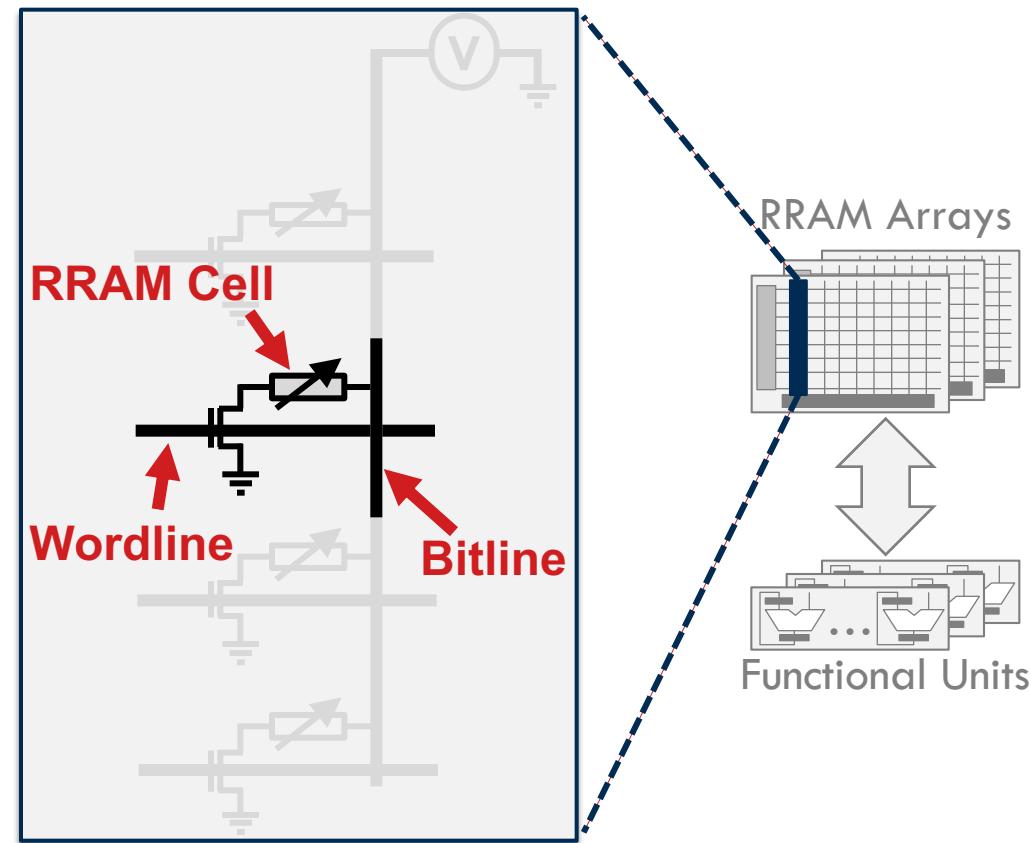
Functional Units

# Resistive Random Access Memory

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

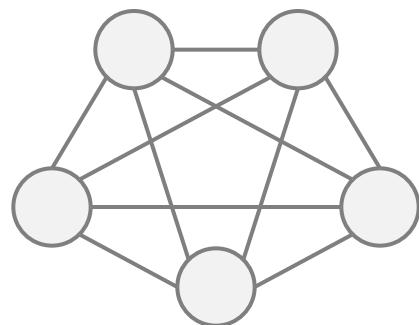


RRAM: Resistive RAM  
(source: HP, 2009)



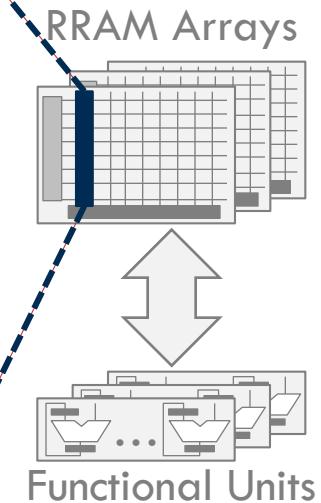
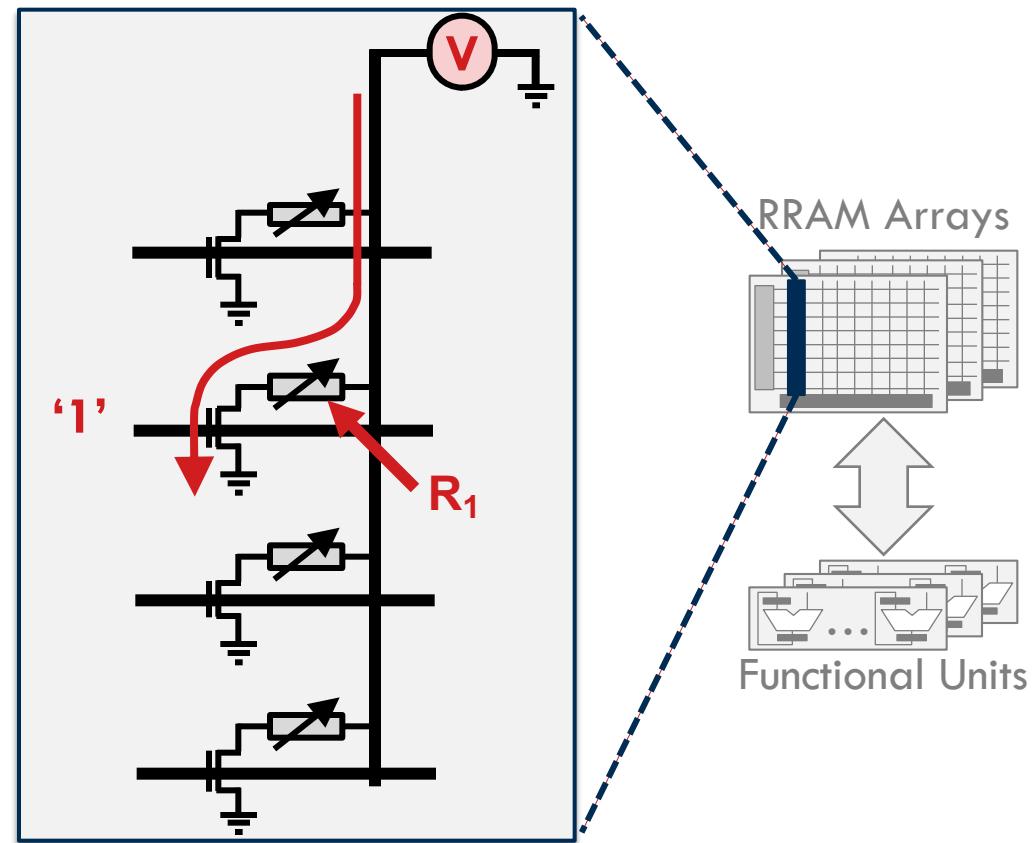
# Resistive Random Access Memory

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



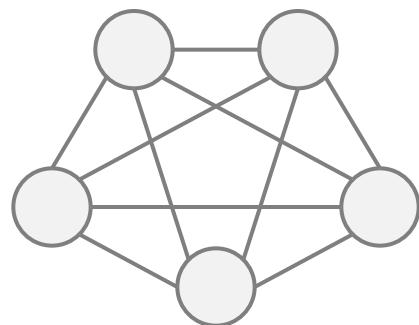
The Boltzmann Machine

$$I = V/R_1$$



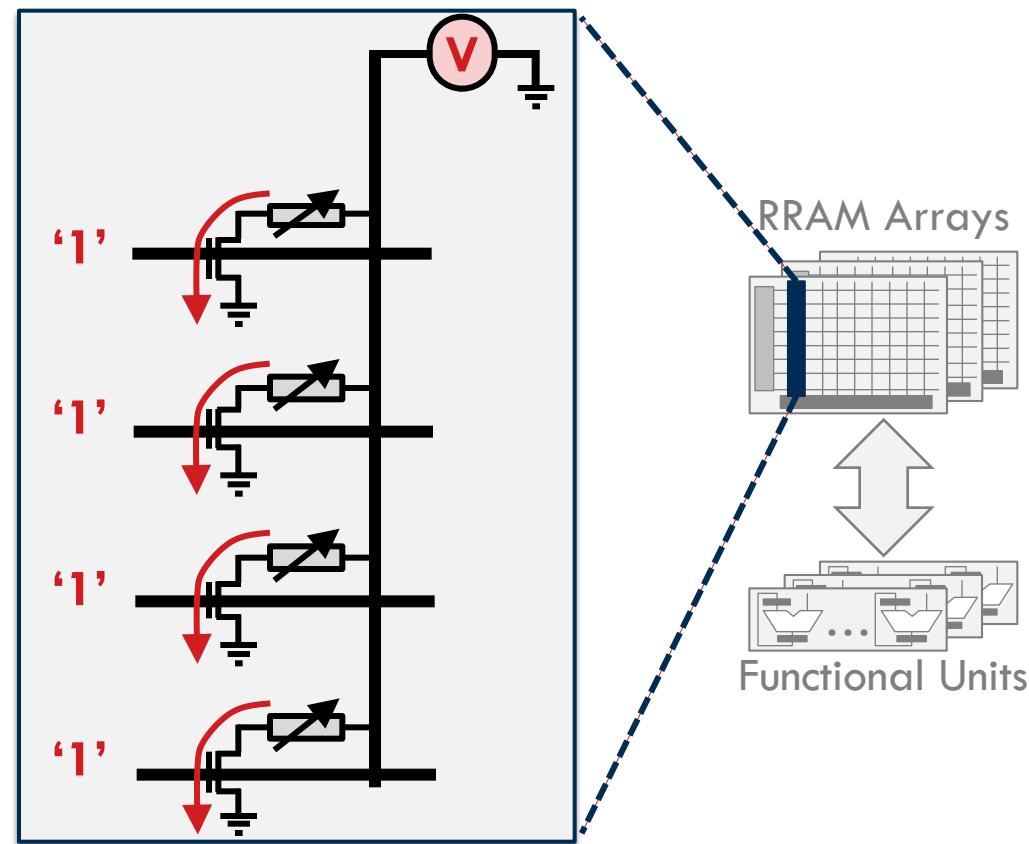
# Memristive Boltzmann Machine

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



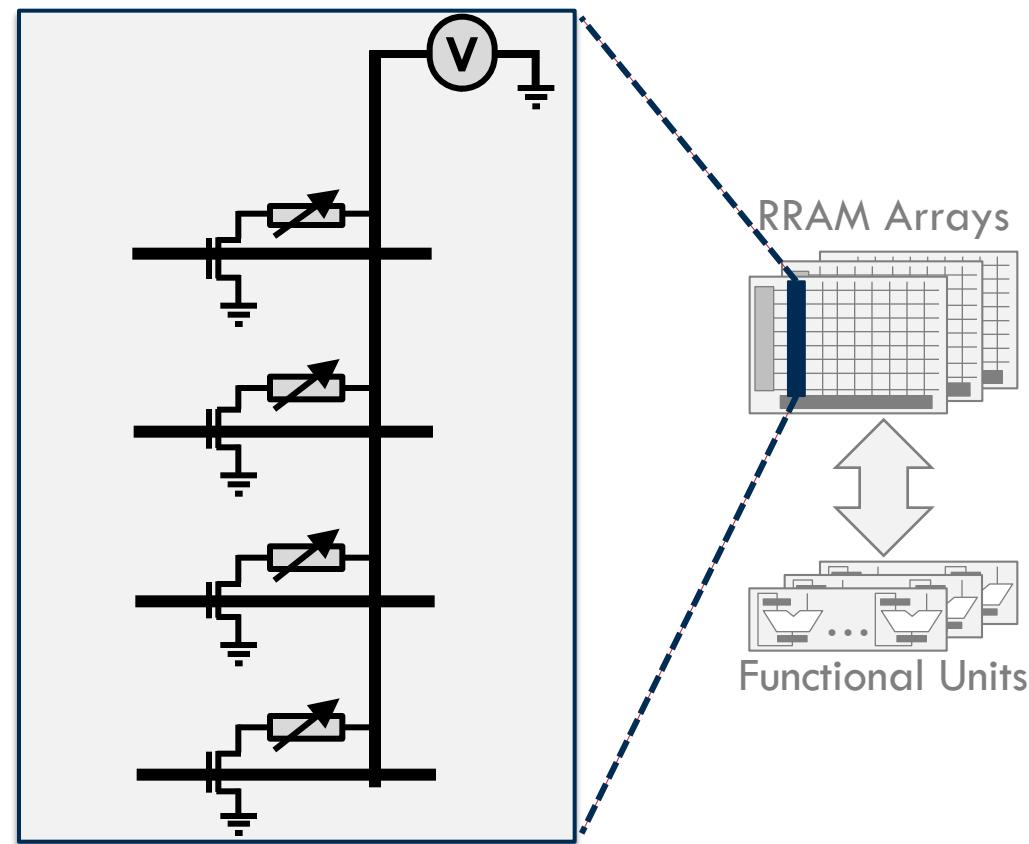
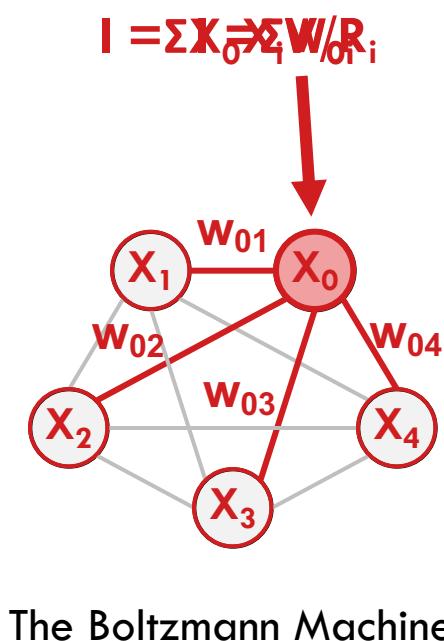
The Boltzmann Machine

$$I = \Sigma V / R_i$$



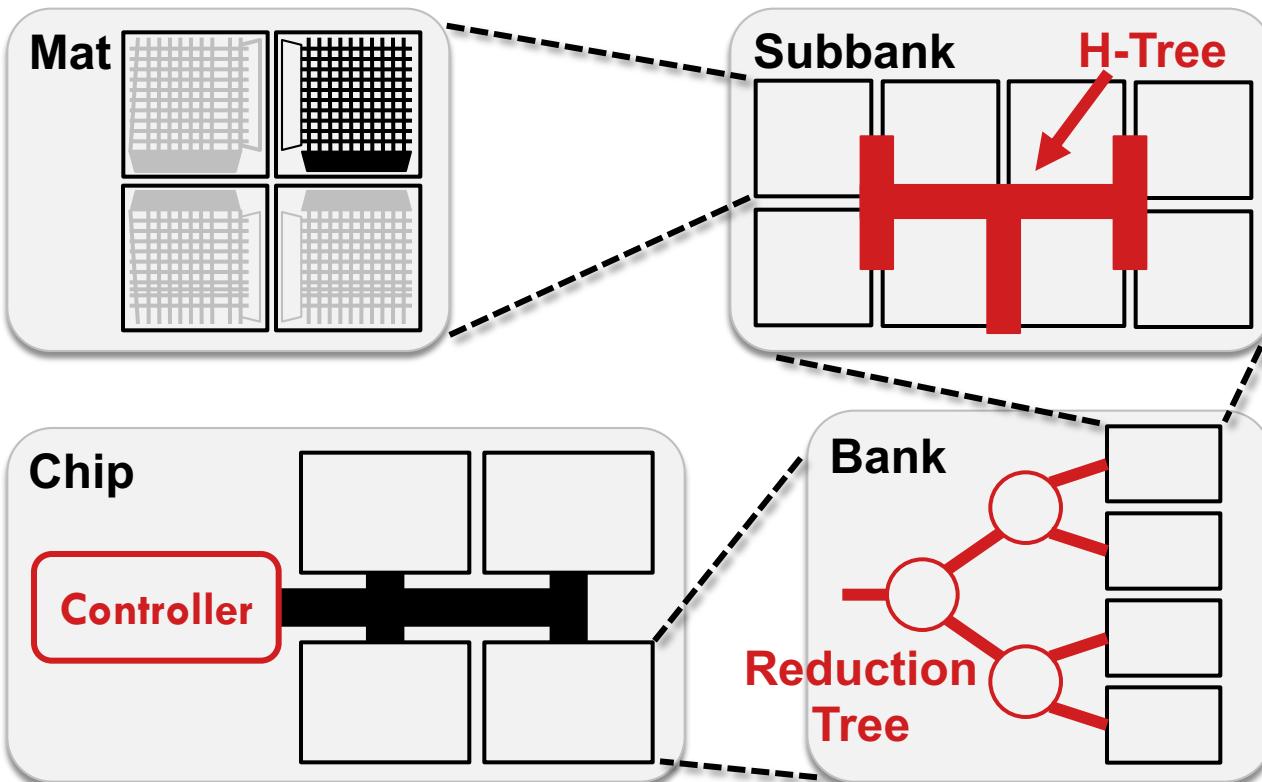
# Memristive Boltzmann Machine

- 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.



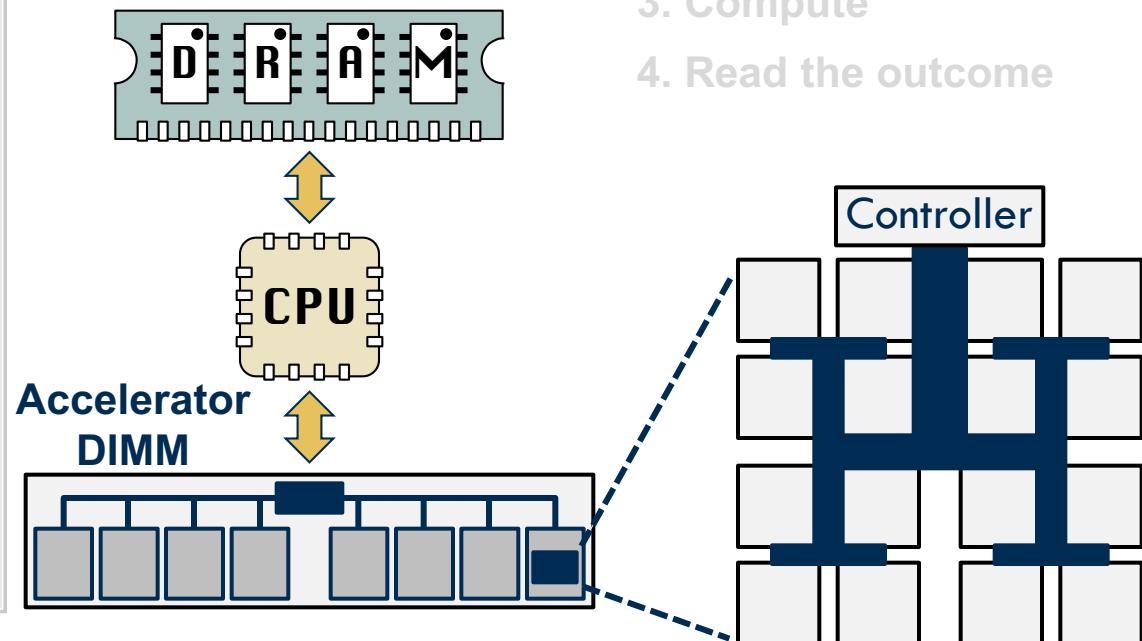
# System Integration

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 uncachable 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



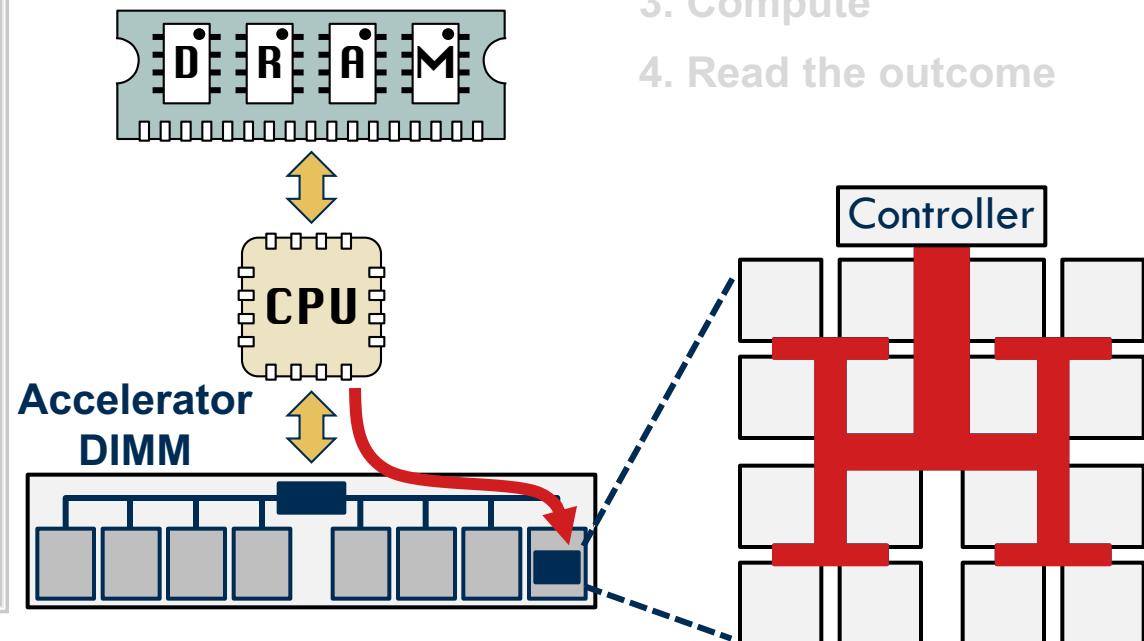
# System Integration

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 uncachable 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



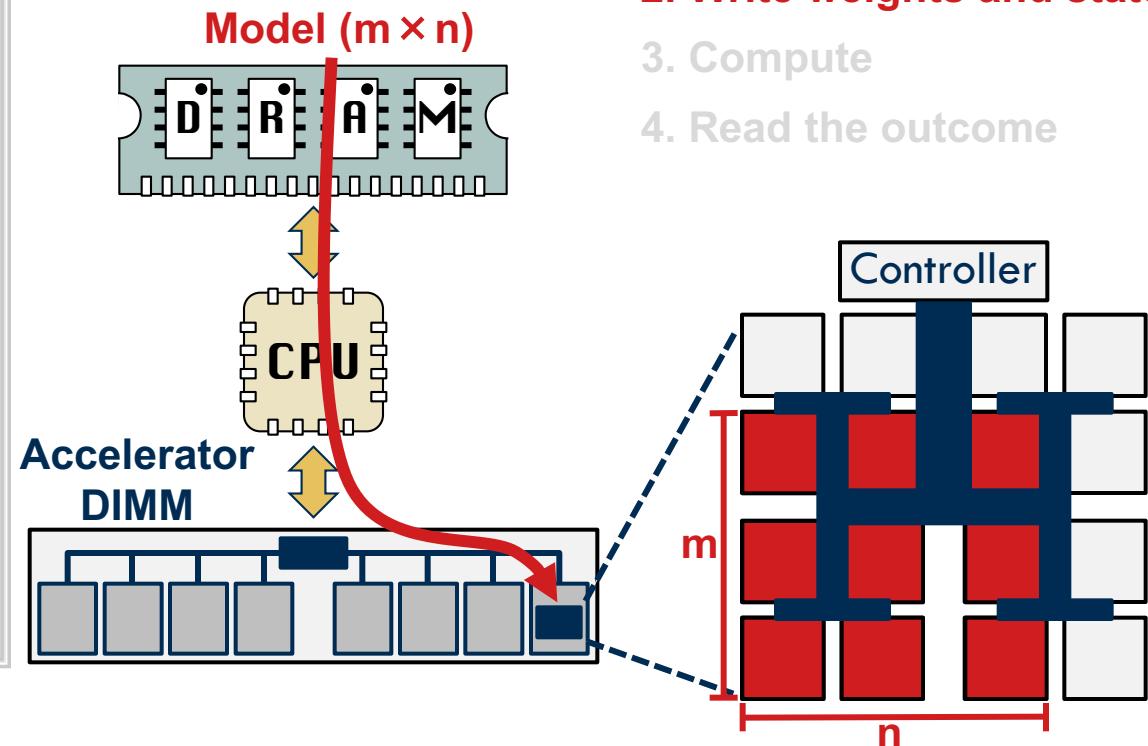
# System Integration

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 uncachable 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



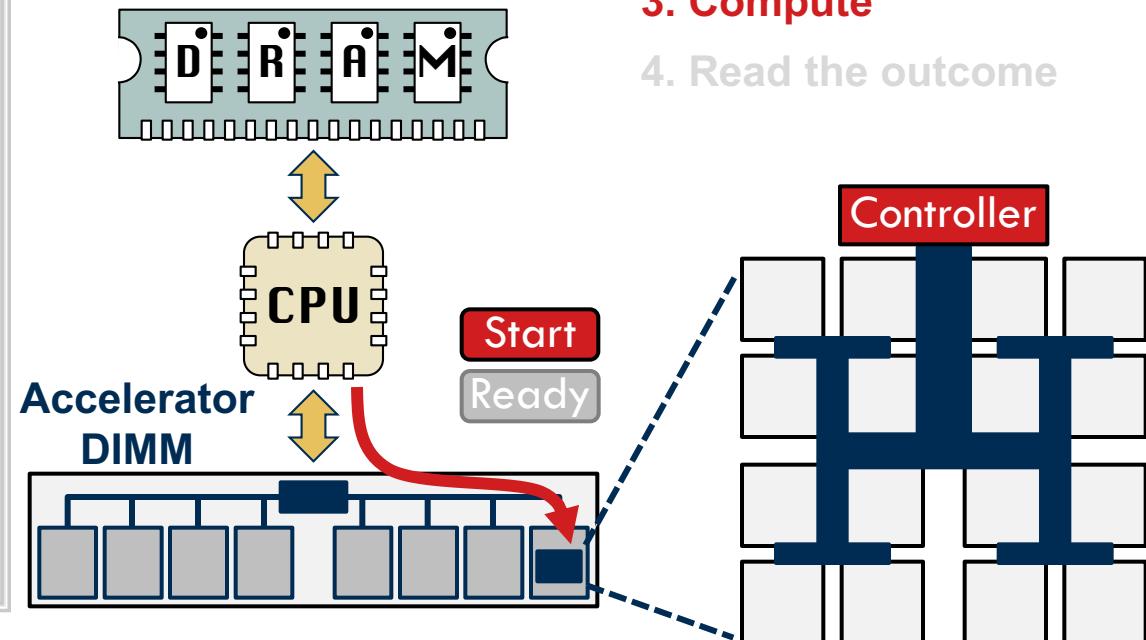
# System Integration

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 uncachable 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



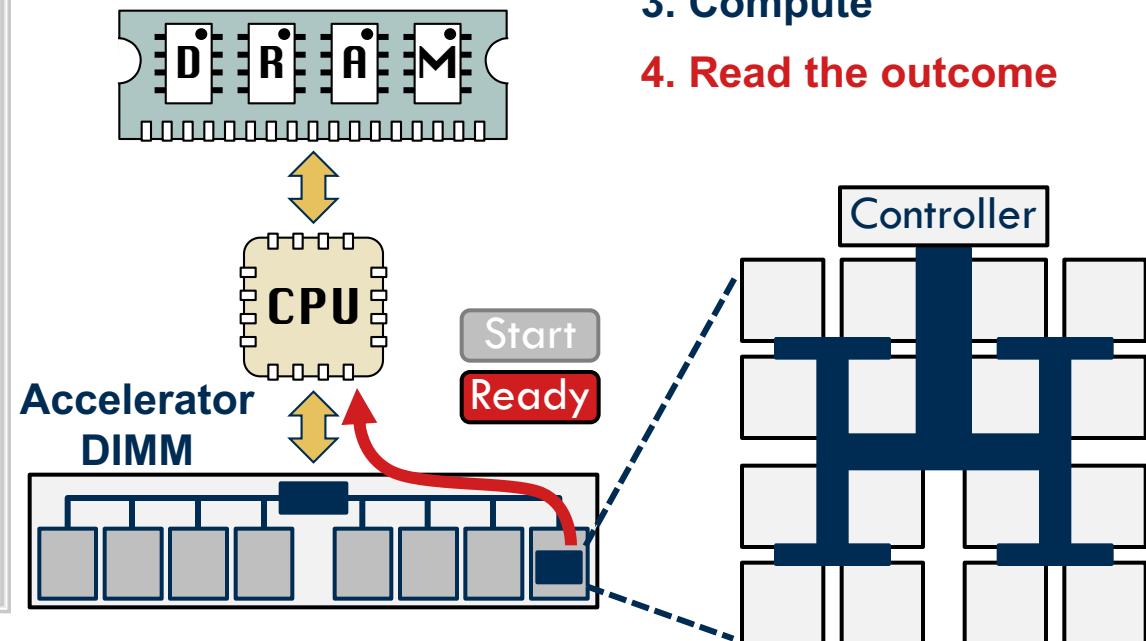
# System Integration

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 uncachable 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

