Processing and Securing with Memristors

A Tutorial

Shahar Kvatinsky

Viterbi Faculty of Electrical Engineering

Technion – Israel Institute of Technology

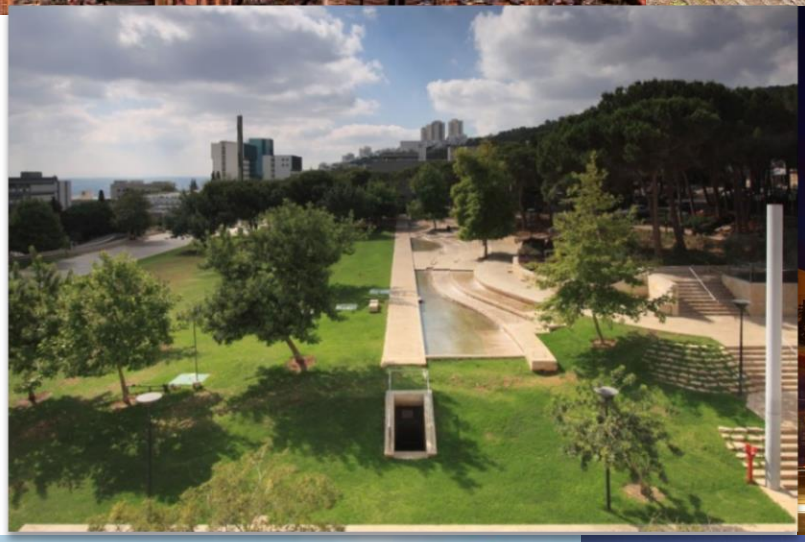
December 2019

ASIC² ARCHITECTURES SYSTEMS INTELLIGENT COMPUTING INTEGRATED CIRCUITS

TECHNION Israel Institute of Technology

Technion – Israel Institute of Technology

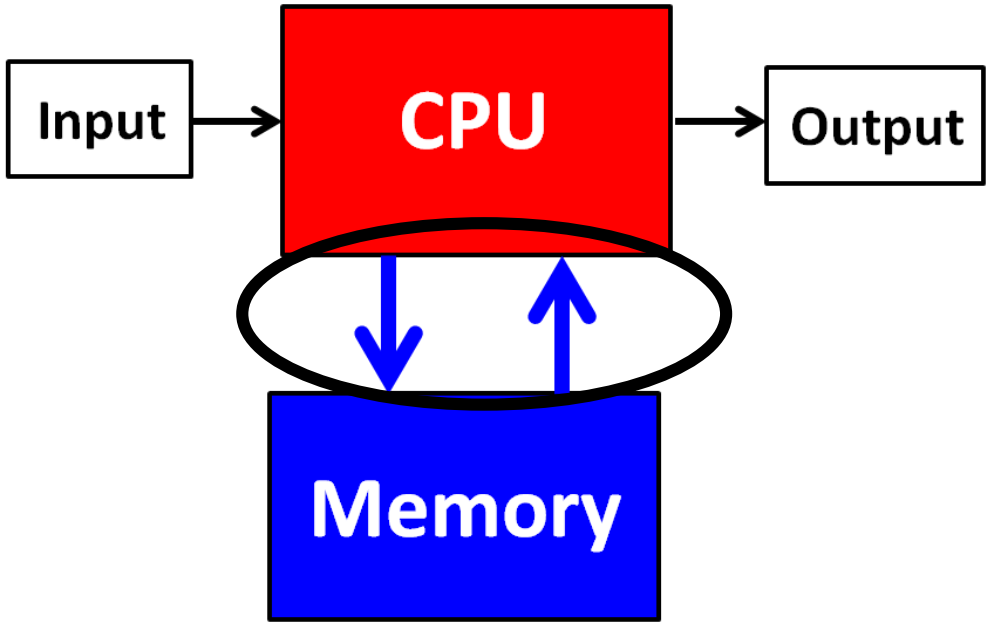
- Campuses in Haifa, China and New York
- 14,000 students, 4 Nobel prizes
- Main contributor to Israeli hi-tech industry

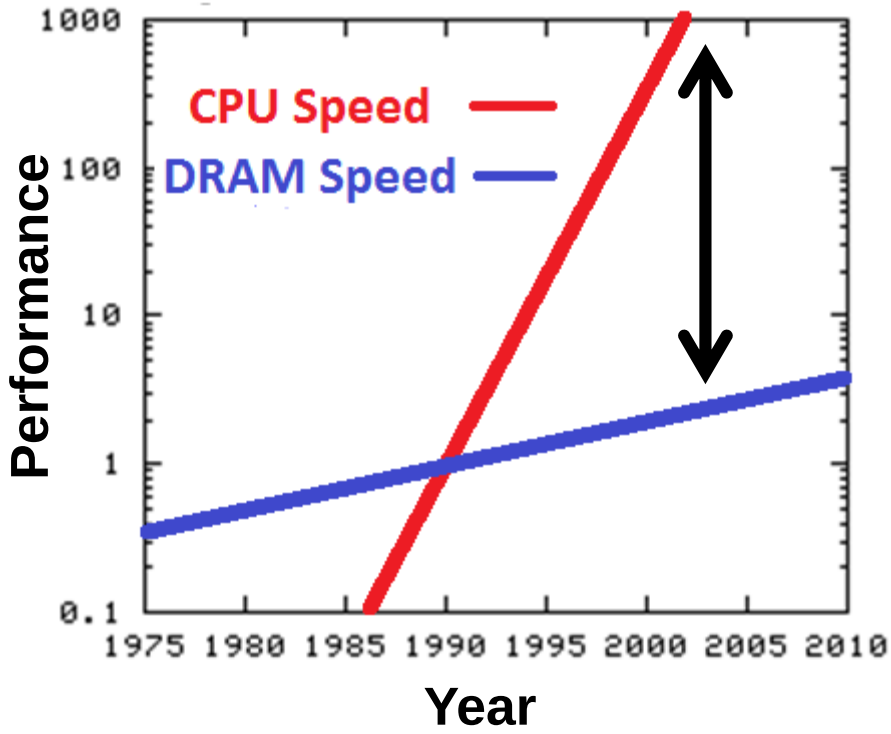


TECHNION
Israel Institute
of Technology

The External Memory Wall Problem

von Neumann (Architecture) Bottleneck





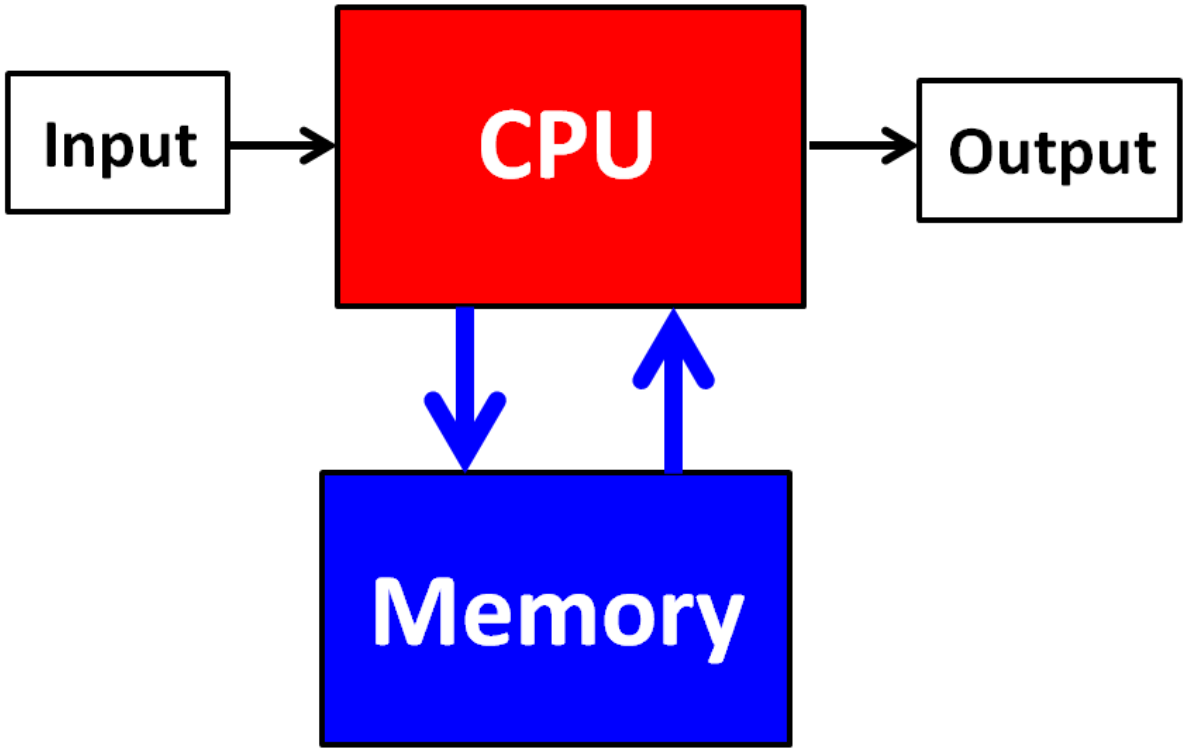
And a Huge Energy Bottleneck

Operation (16-bit operand)	**Energy/Op** (45 nm)	**Cost** (vs. Add)
Add operation	0.18 pJ	1X
Load from on-chip SRAM	11 pJ	61X
Send to off-chip DRAM	**640 pJ**	**3,556X**

>1000X more energy to go to memory

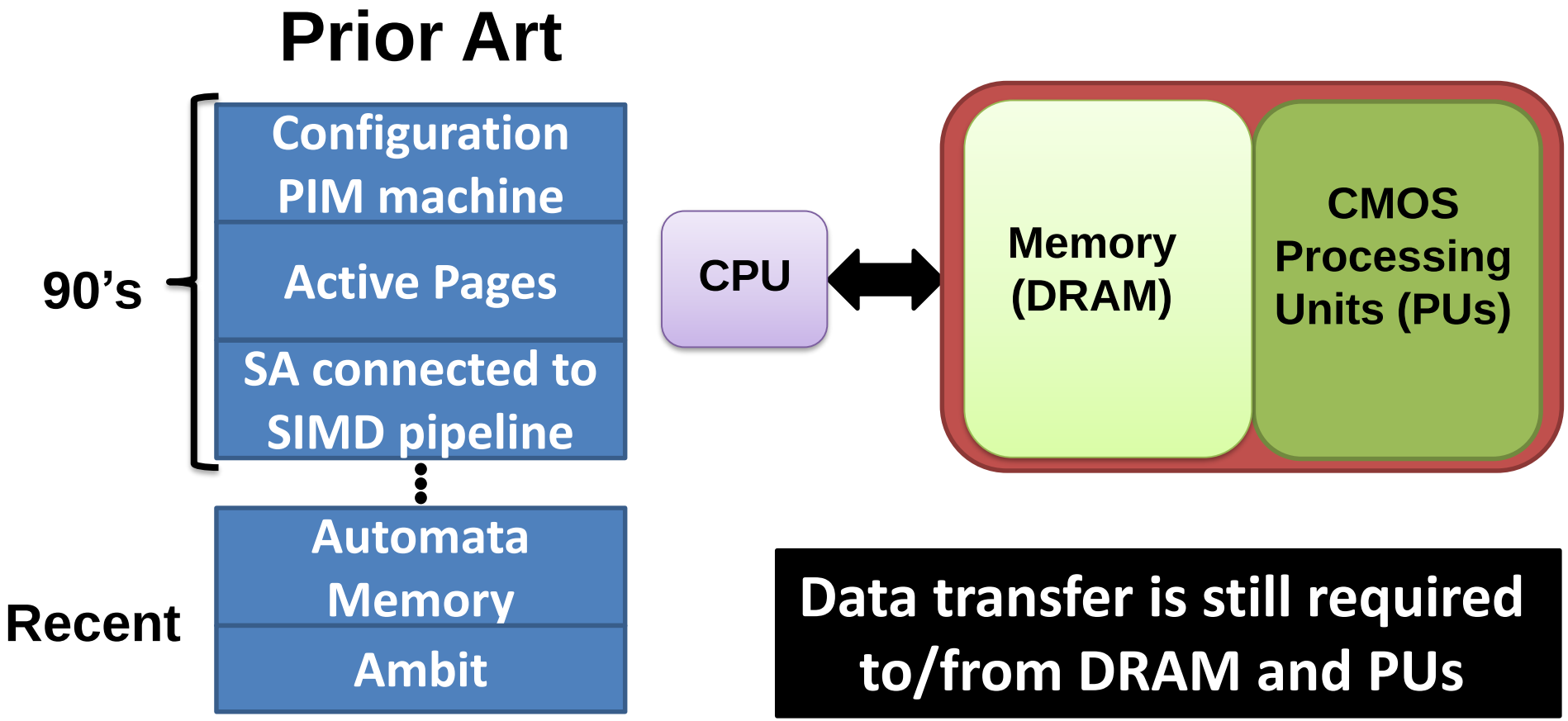
A. Pedram, S. Galal, S. Richardson, S. Kvatinsky, and M. Horowitz, "Dark Memory and Accelerator-Rich System Optimization in the Dark Silicon Era," IEEE Design & Test, April 2017

Moving Computation to Memory



Processing "In-Memory" (PIM)

Reducing Data Movement

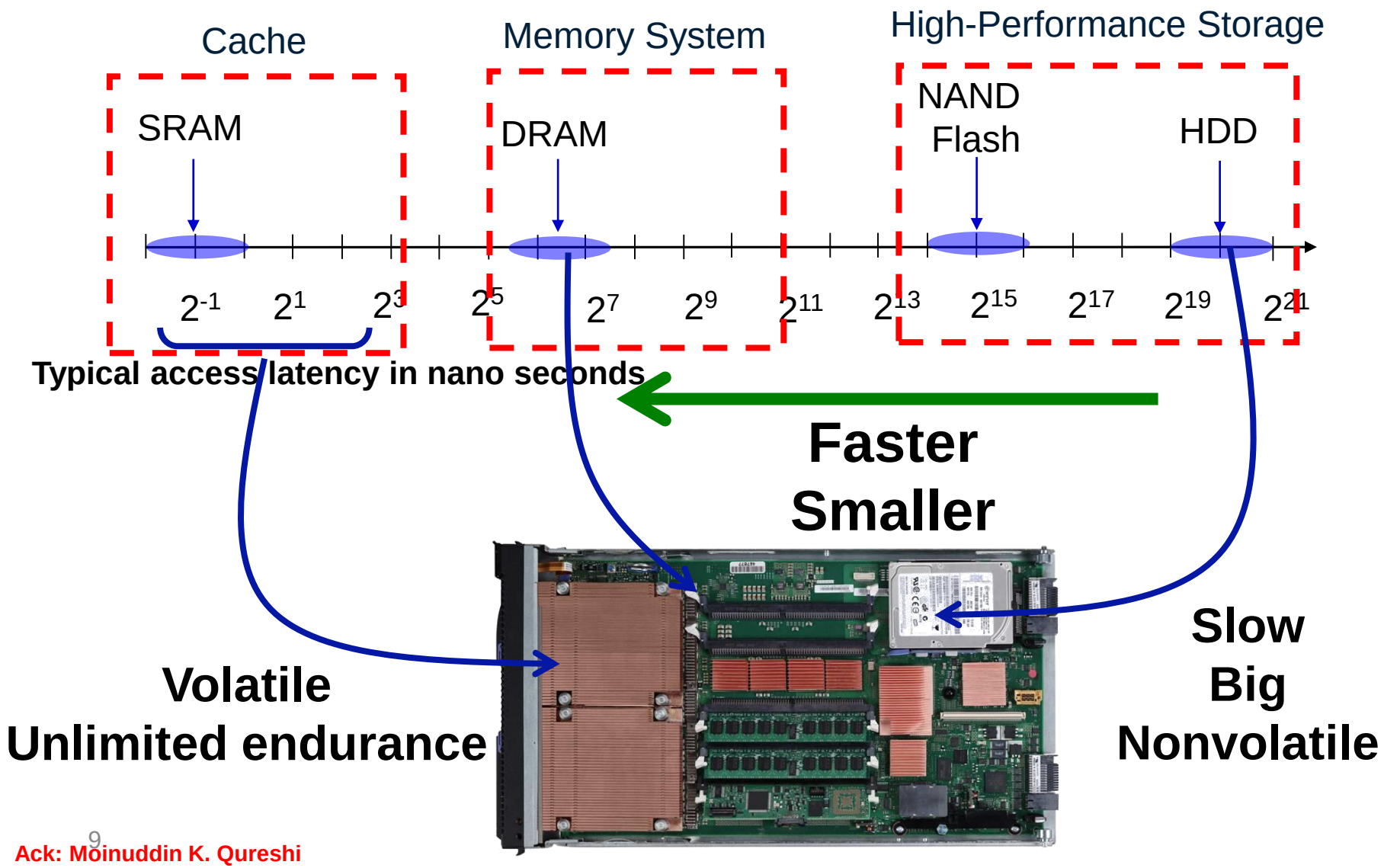


H. S. Stone, "A Logic-in-Memory Computer," *IEEE Transactions on Computers*, January 1970
M. Gokhale *et al.*, "Processing in memory: the Terasys massively parallel PIM array," *Computer*, 1995
M. Oskin *et al.*, "Active pages: A computation model for intelligent memory," *Comput. Archit. News*, 1998
D. Elliott *et al.*, "Computational ram: Implementing processors in memory," *IEEE Des. Test*, 1999
P. Dlugosch *et al.*, "An Efficient and Scalable Semiconductor Architecture for Parallel Automata Processing," *IEEE TPDS*, 2014
V. Seshadri *et al.*, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," MICRO 2017

This Tutorial

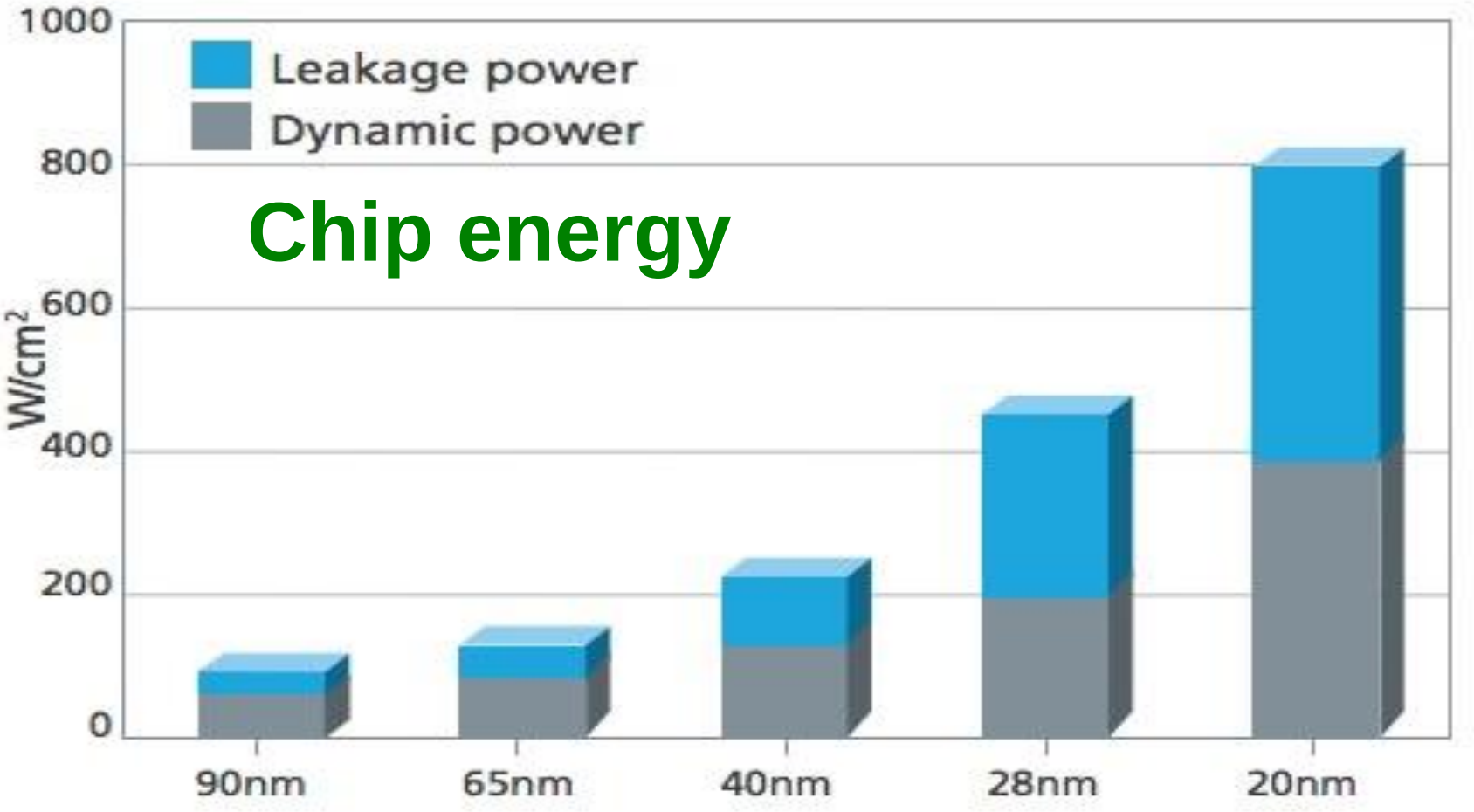
- The need for Processing-in-Memory
- **Memristors and Memristive Memory**
- Memristive logic classification
- Out-of-memory logic
- Near-memory logic
- Real in-memory logic
- Memristors for HW security

Memory Hierarchy



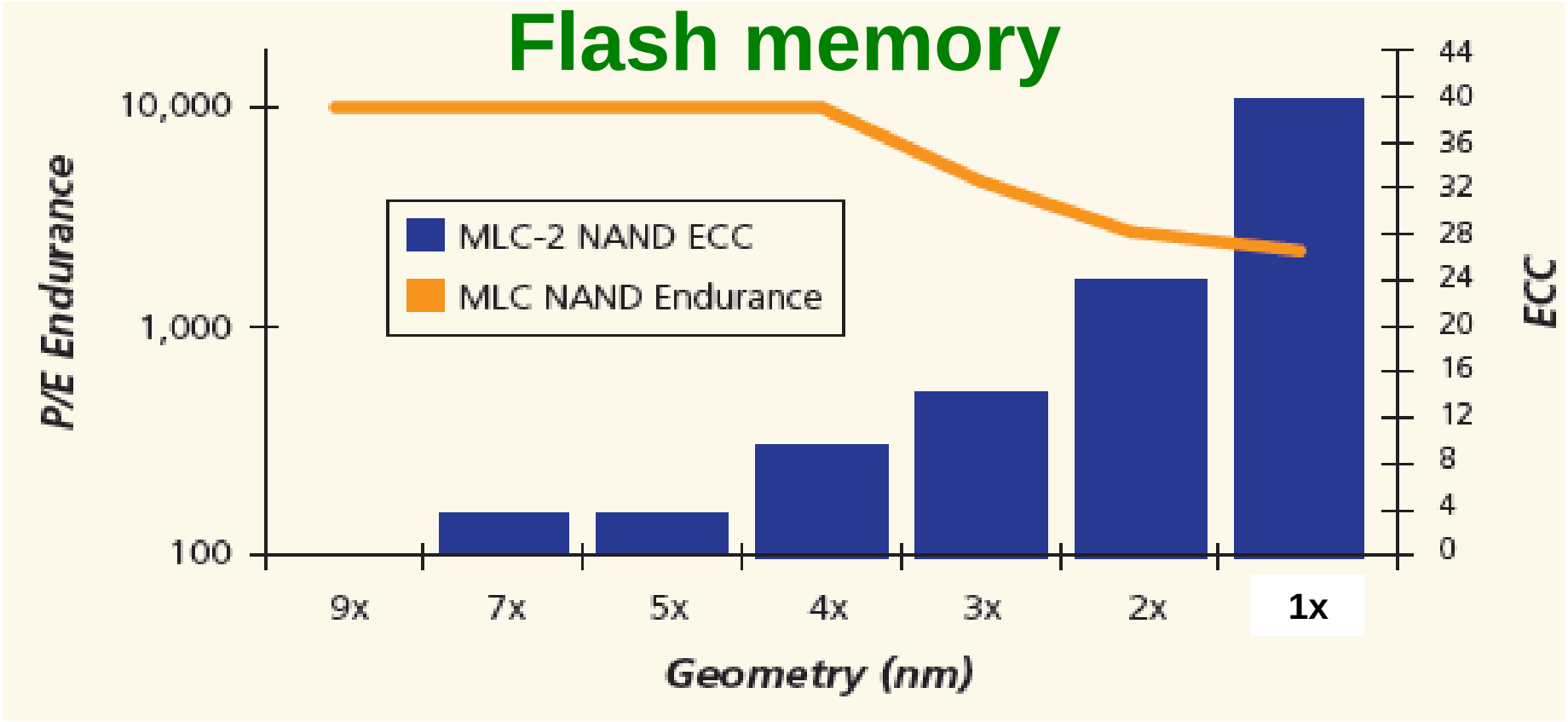
Ack: Moinuddin K. Qureshi

The Problem with SRAM



Source: E. Esteve, IPNEST

The Problem with Flash Memory

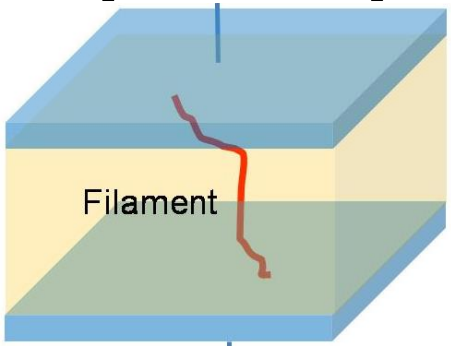


Source: D. Ruggeri, MICRON

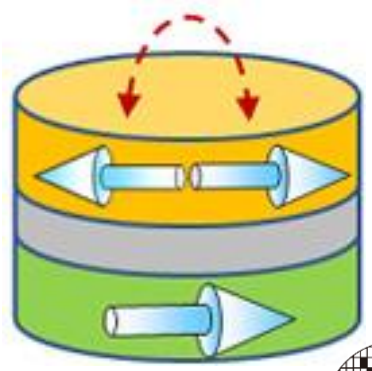
Memristors

Emerging Nonvolatile Memory Technologies

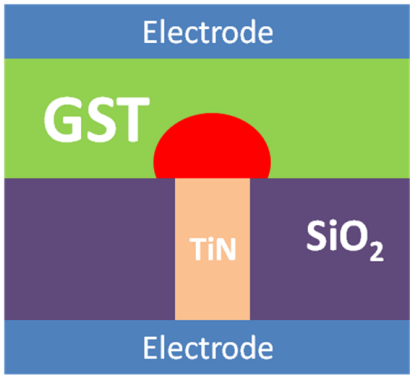
Resistive RAM (RRAM)



STT MRAM



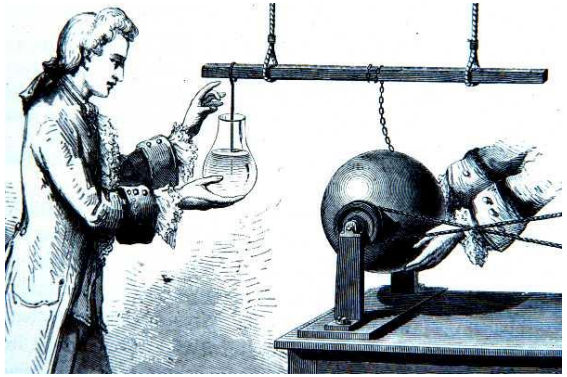
Phase Change Memory (PCM)



3 Fundamental Circuit Elements

Resistor
Georg Ohm 1827

Resistor $v = R \cdot i$	Capacitor $q = C \cdot v$
Inductor $\varphi = L \cdot i$	



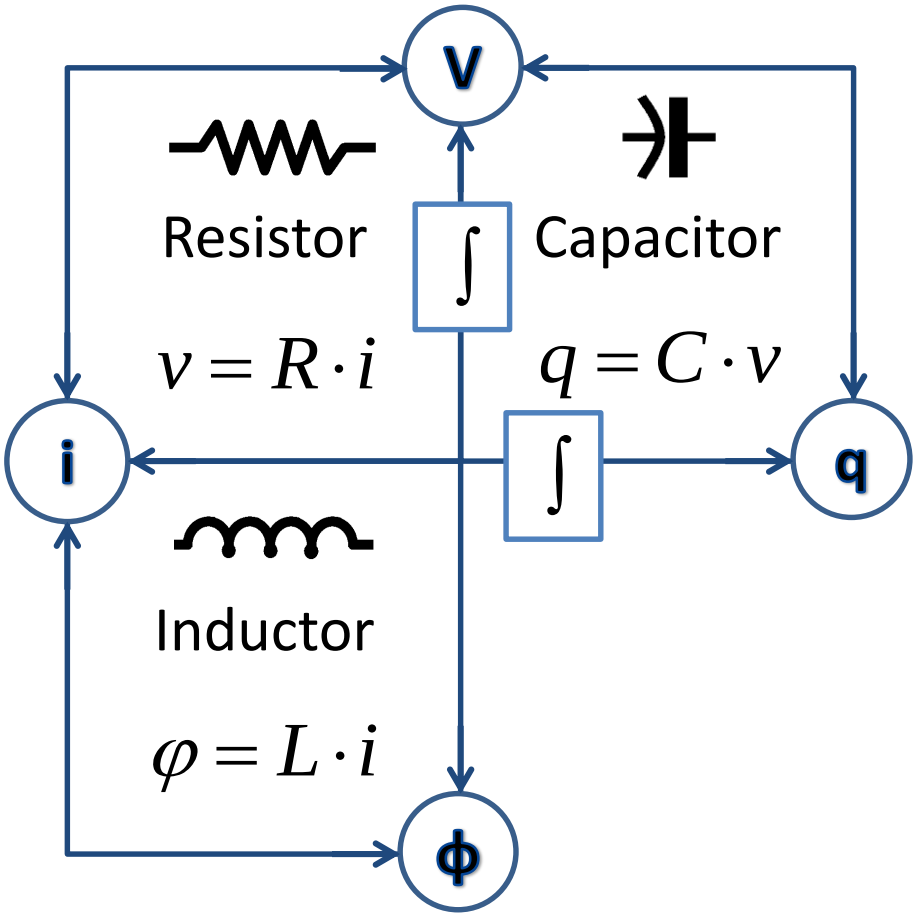
Capacitor
von Kleist 1745

Inductor
Michael Faraday 1831

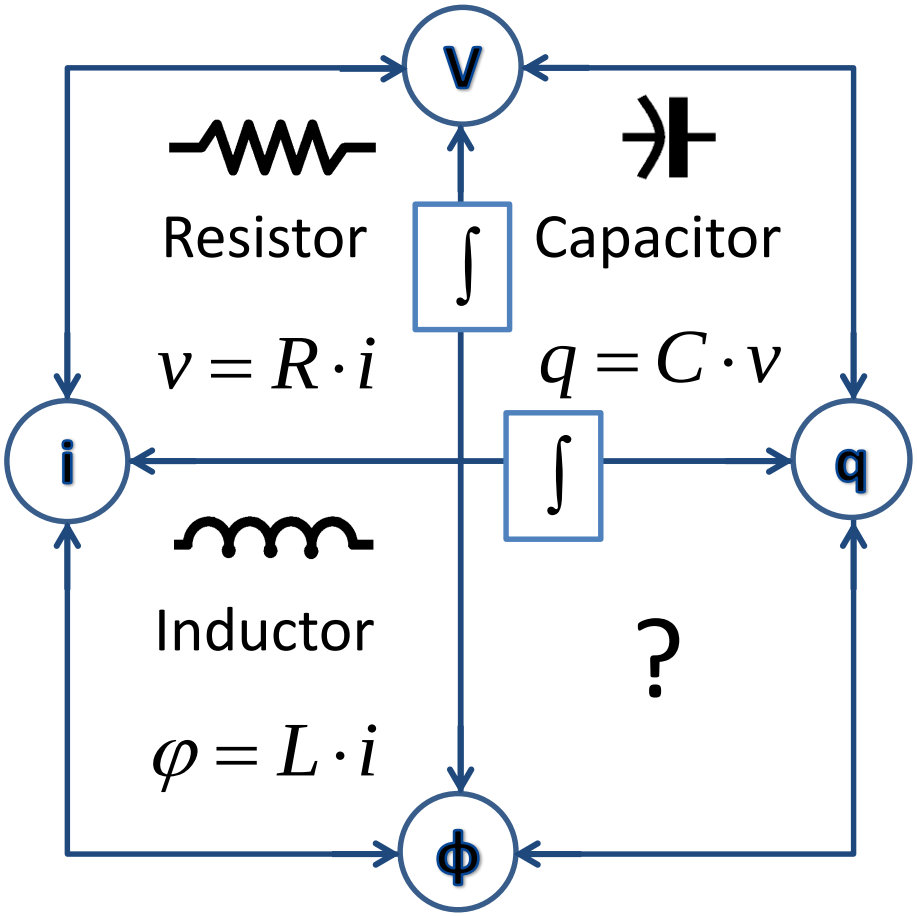
4 Basic Circuit Variables

- *Voltage* $v(t)$
- *Current* $i(t)$
- *Charge* $q(t) \triangleq \int_{-\infty}^{t} i(\tau)\, d\tau$, or $i = \dfrac{dq(t)}{dt}$
- *Flux* $\varphi(t) \triangleq \int_{-\infty}^{t} v(\tau)\, d\tau$, or $v = \dfrac{d\varphi(t)}{dt}$

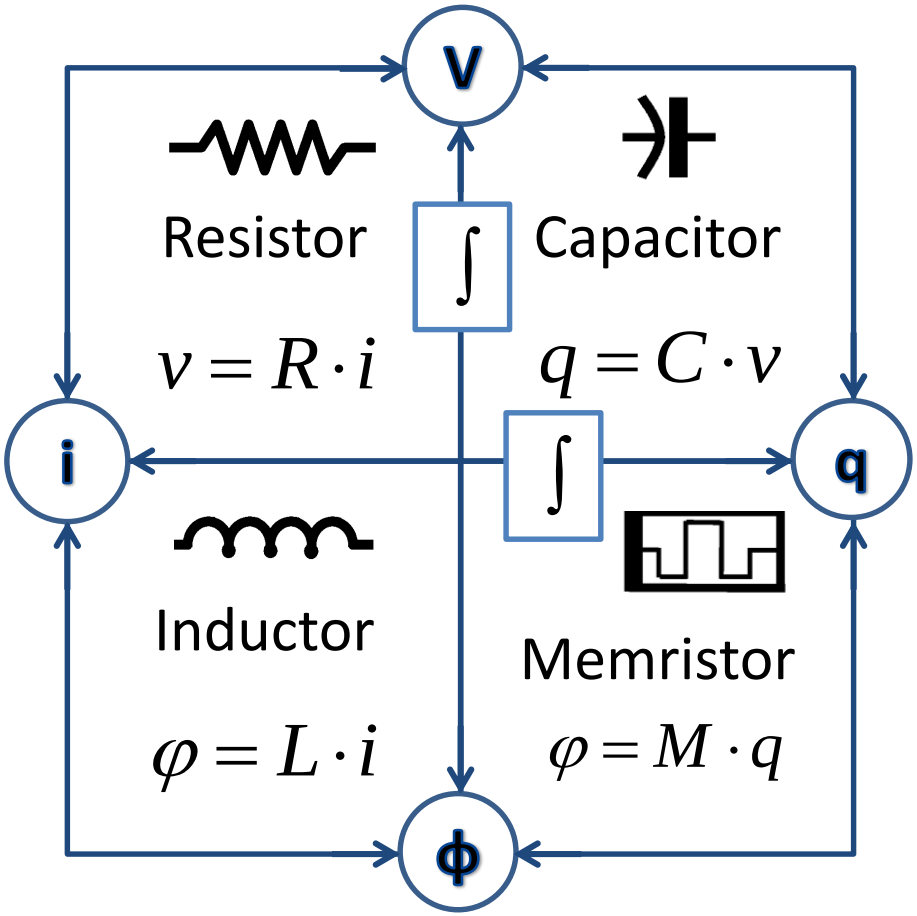
3 Fundamental Circuit Elements



The Missing Circuit Element



The Memristor 1971



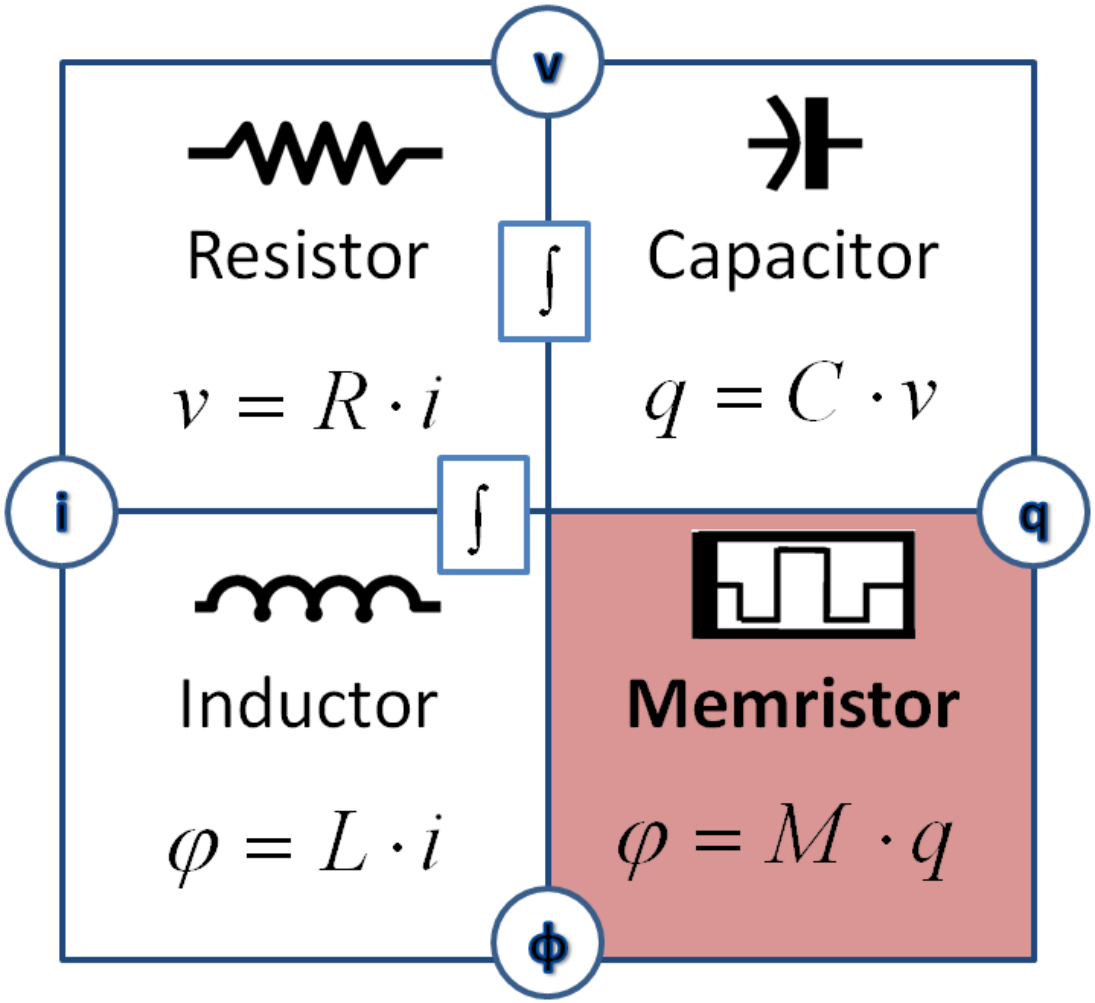
IEEE TRANSACTIONS ON CIRCUIT THEORY, VOL. CT-18, NO. 5, SEPTEMBER 1971 507

Memristor—The Missing Circuit Element

LEON O. CHUA, SENIOR MEMBER, IEEE

Memristor

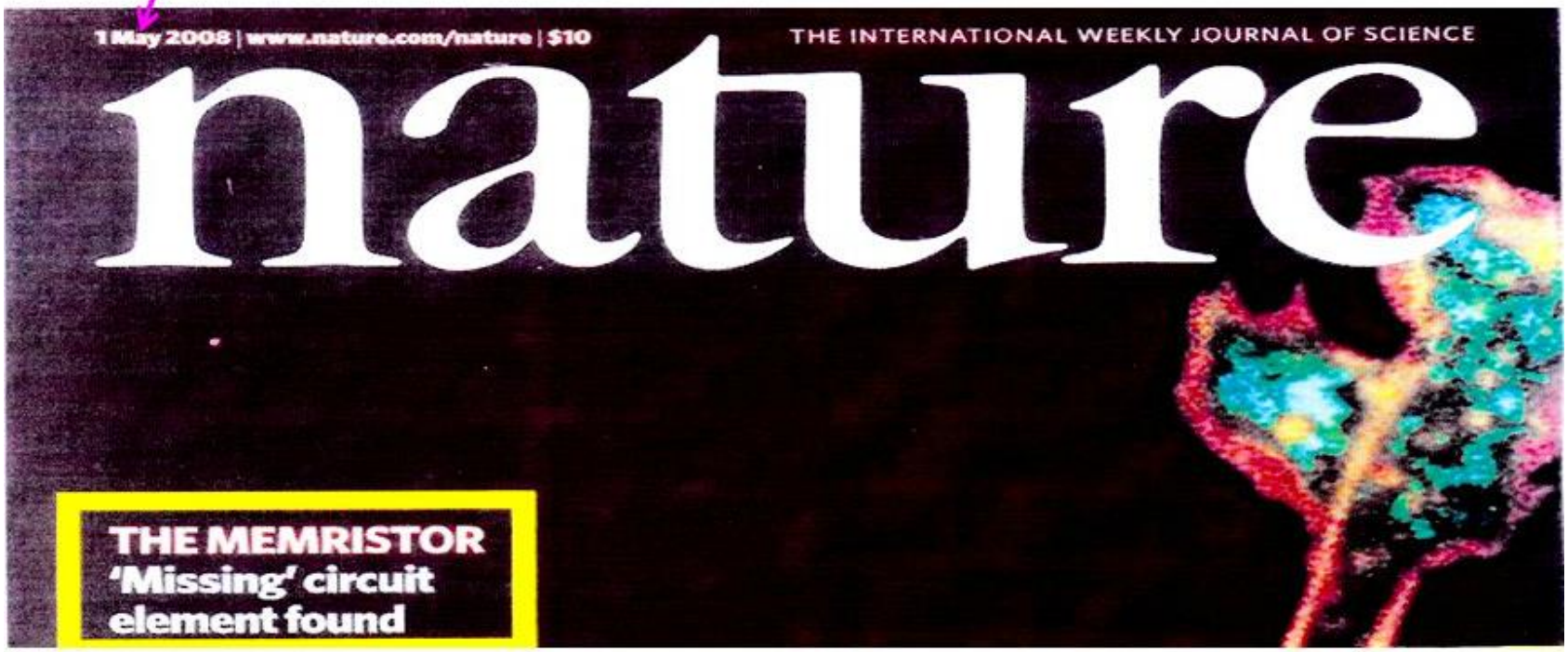
The Missing Fourth Element?



L.O. Chua, "Memristor – The Missing Circuit Element," IEEE Trans. on Circuit Theory, 1971

The Conventional Story

1 May 2008

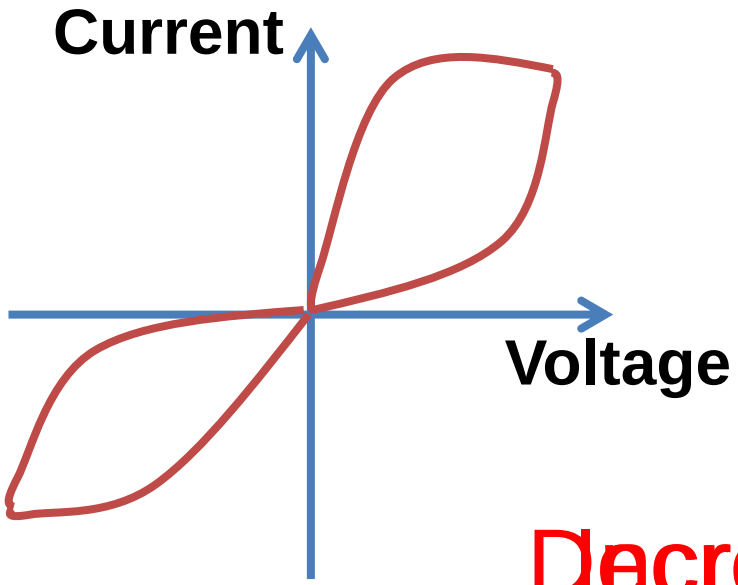


The missing memristor found

Dmitri B. Strukov[1], Gregory S. Snider[1], Duncan R. Stewart[1] & R. Stanley Williams[1]

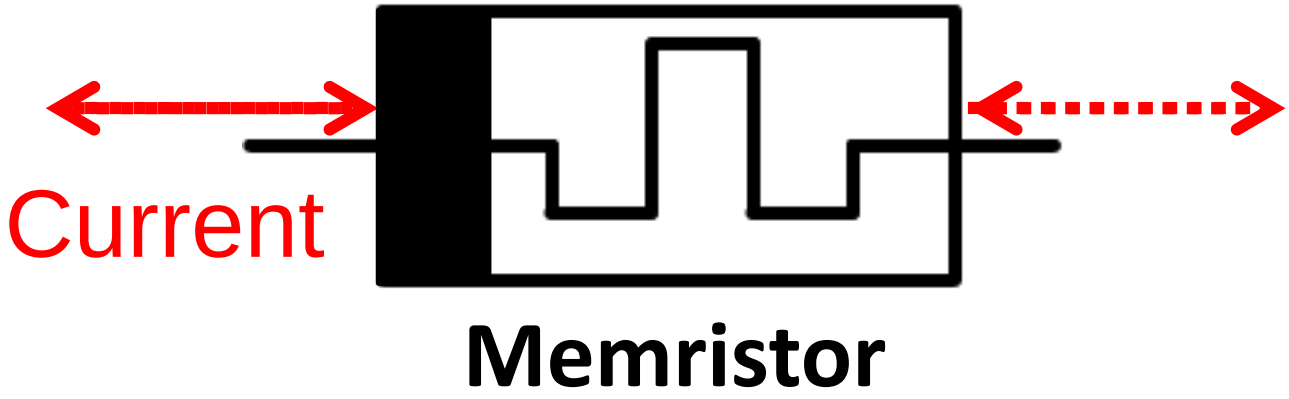
Memristor – A Practical Illustration

Resistor with Varying Resistance



High resistive state
(R_{OFF}, HRS)

Decrease resistance



Attractive for Memory!

CMOS compatible

Rad hard

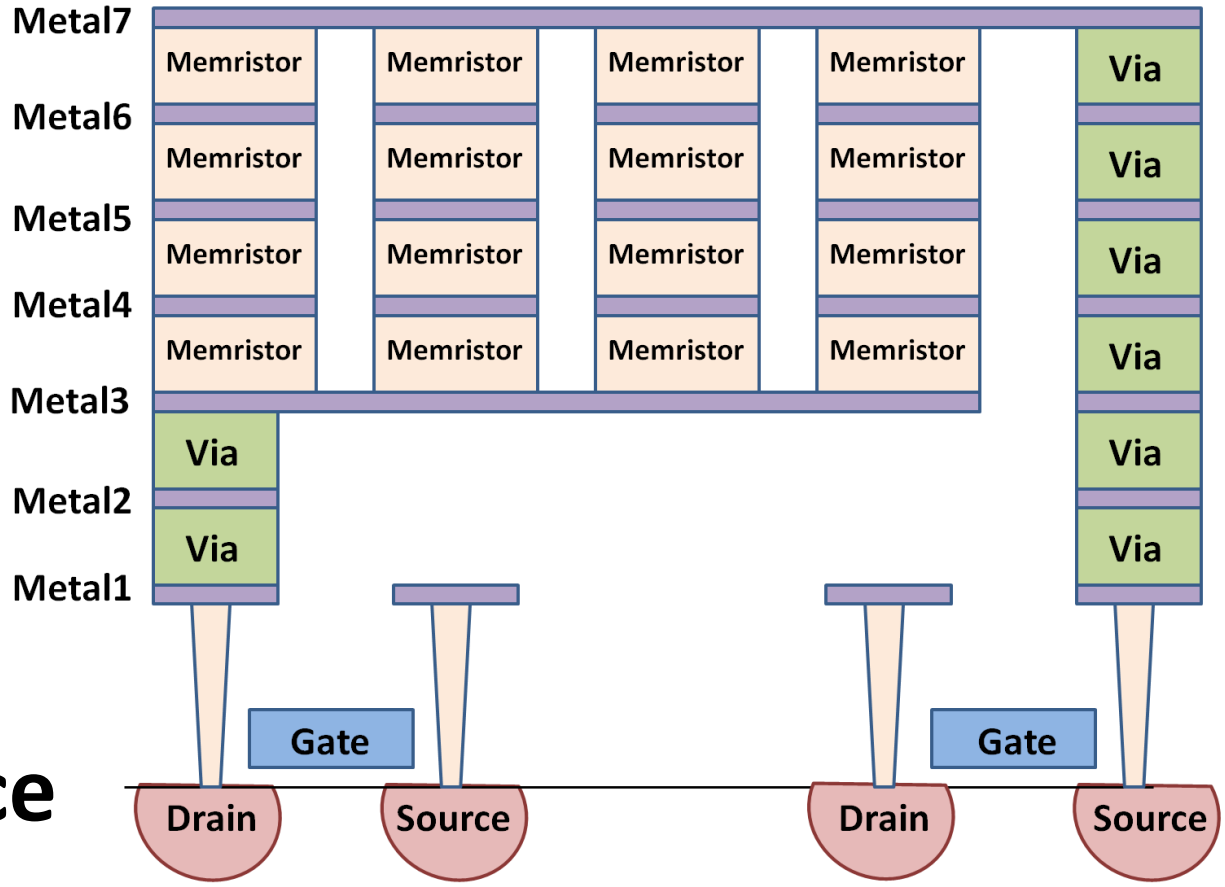
Dense

Nonvolatile

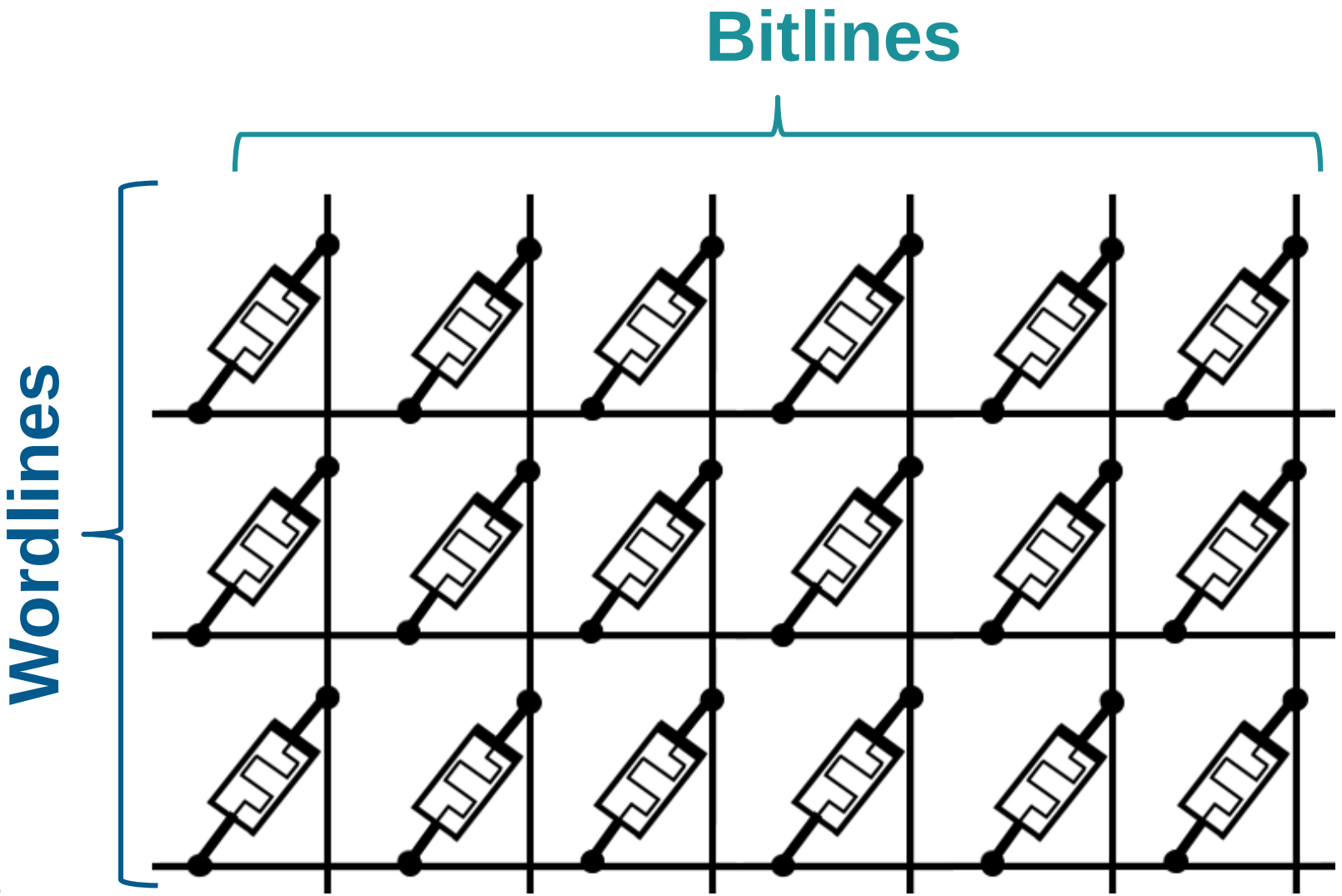
Fast

Low power

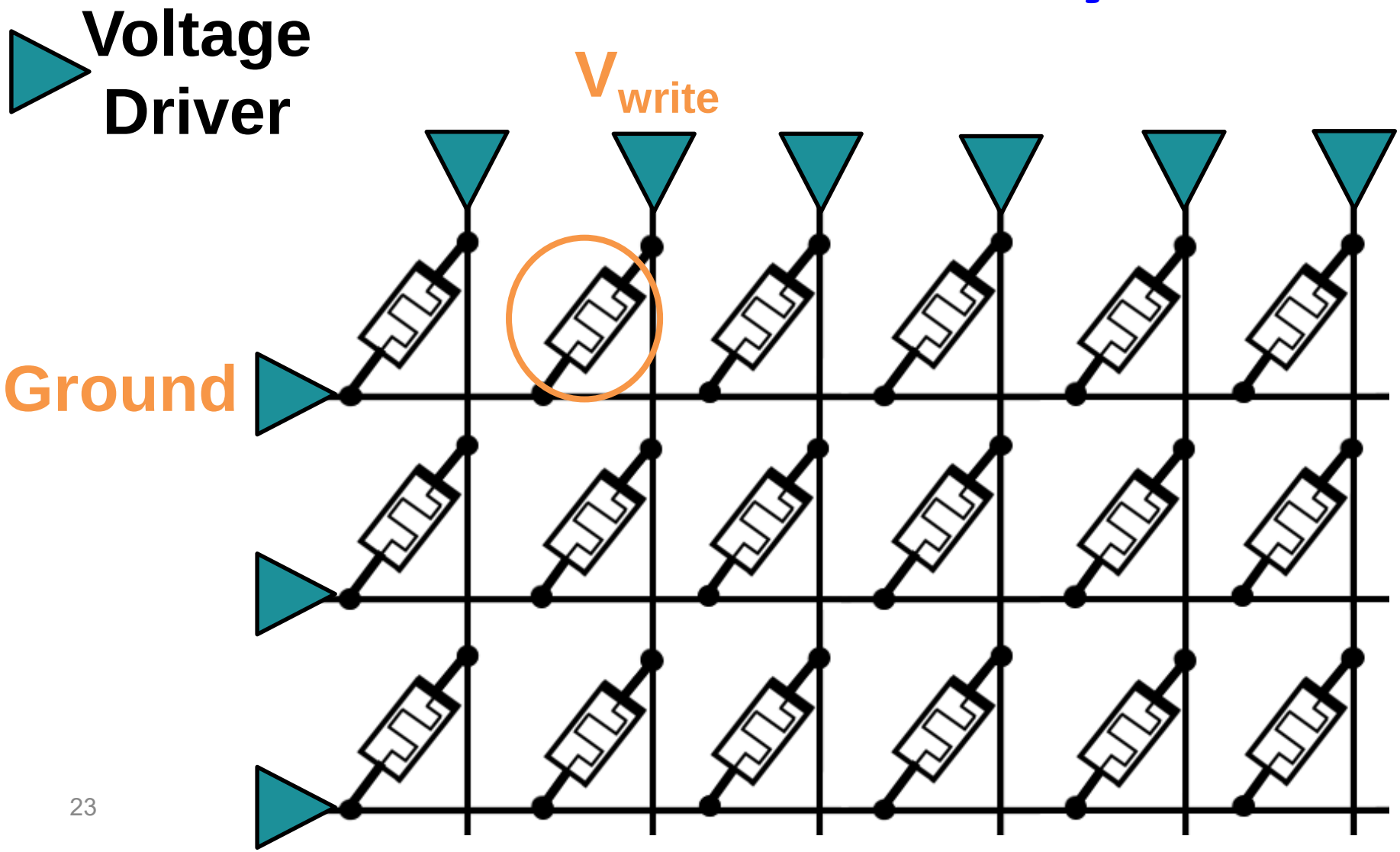
High endurance



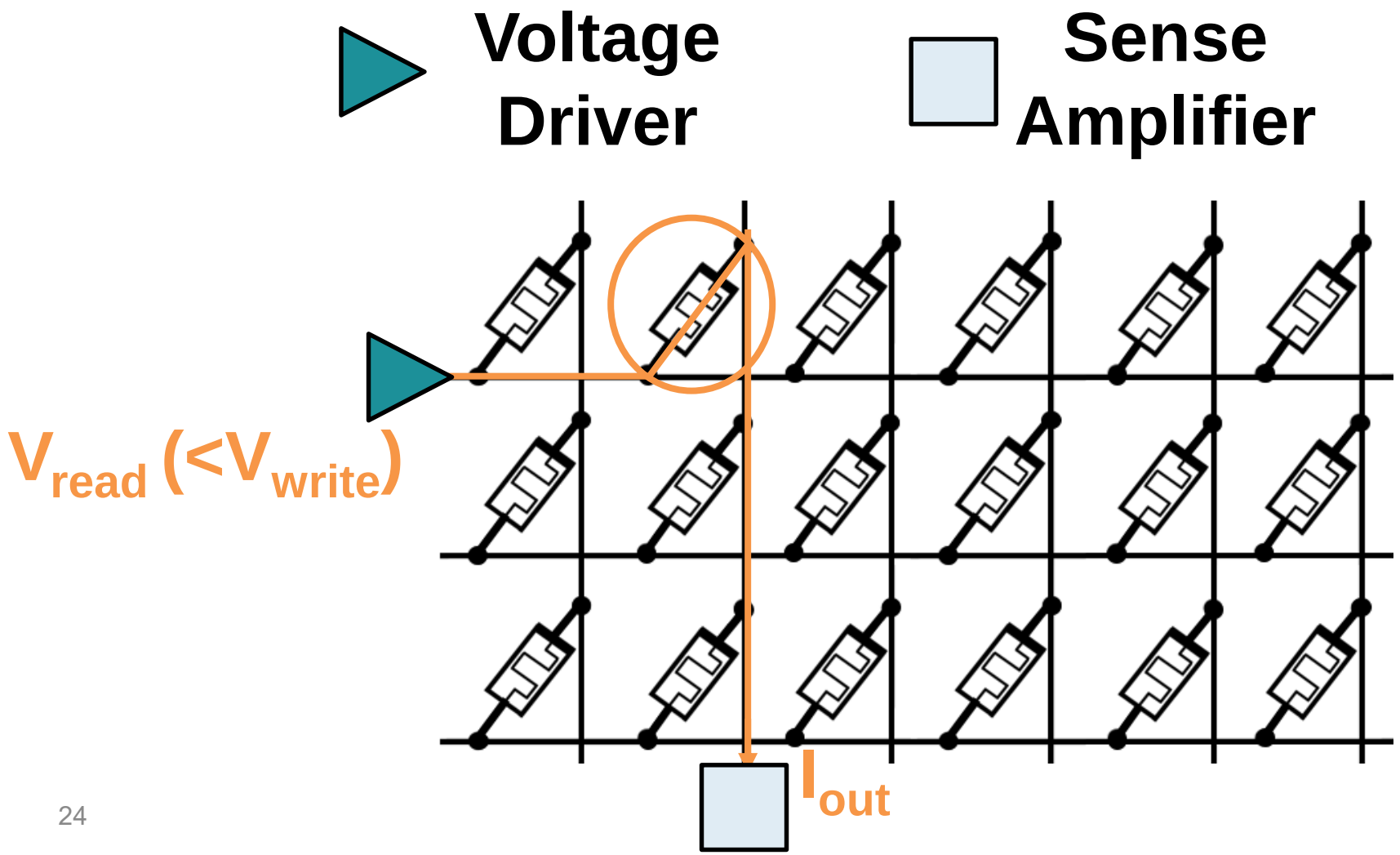
We Want to Compute within the Memristive Crossbar Memory



Write Operation in the Memristive Memory

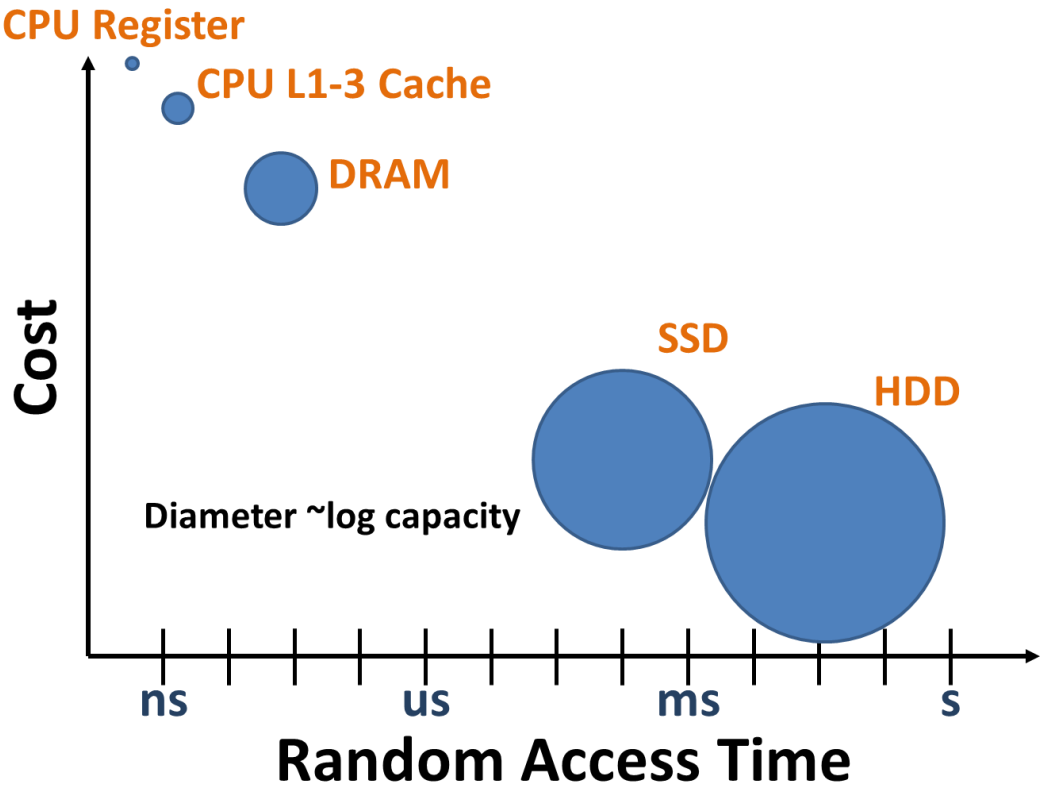


Read Operation in the Memristive Memory



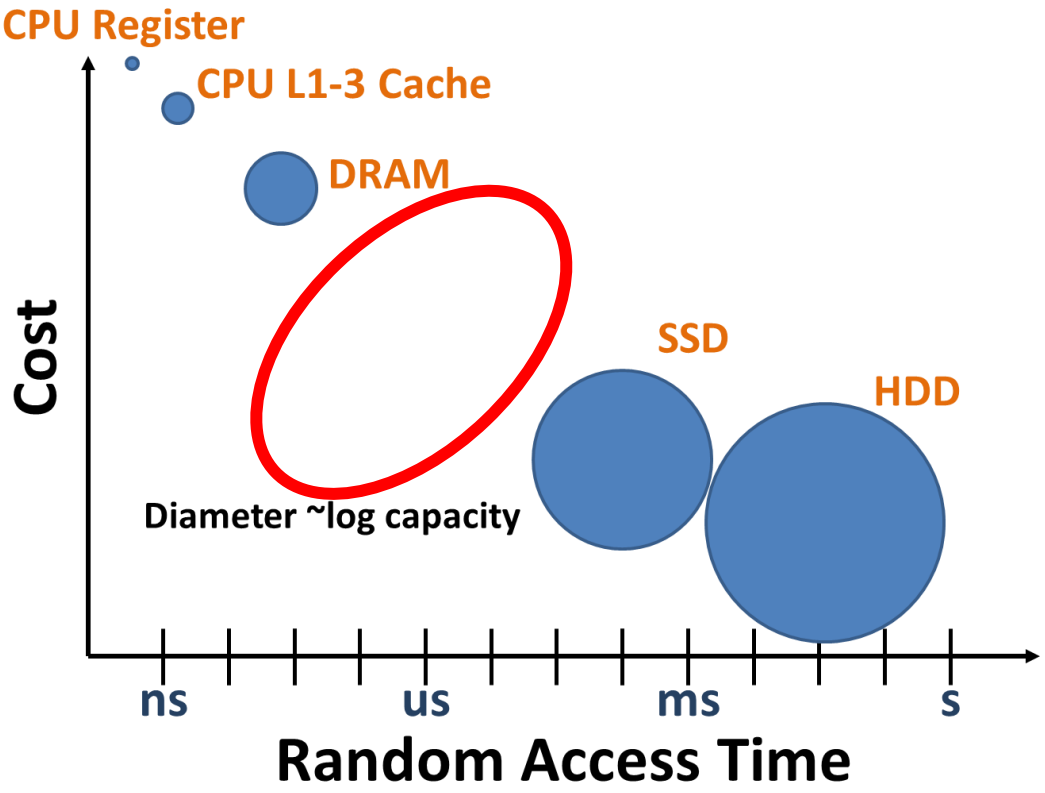
Immediate Applications

- NAND Flash replacement
- Embedded NVM



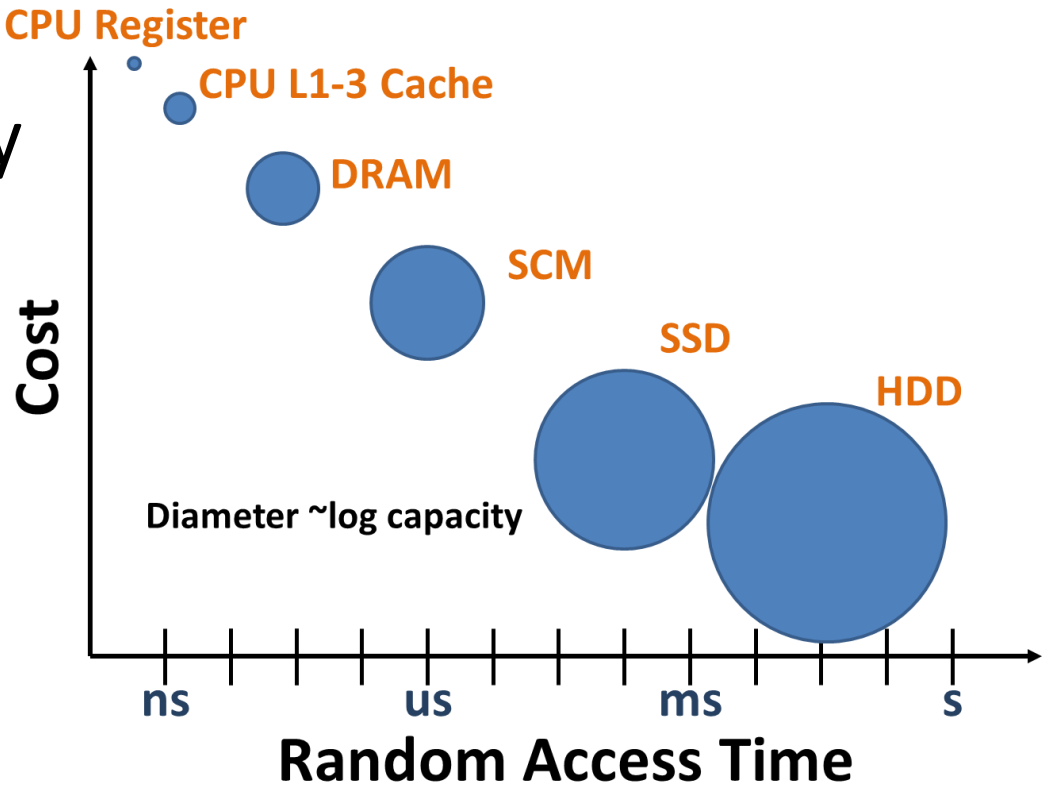
Immediate Applications

- NAND Flash replacement
- Embedded NVM



Immediate Applications

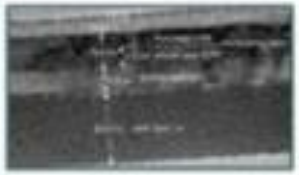
- NAND Flash replacement
- Embedded NVM
- Storage Class Memory



Existing Products 2019

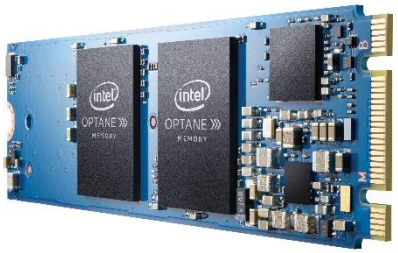
3rd Gen. MRAM

2017 ~
300 mm
GF@Singapore fab.
pMTJ ST-MRAM
256 Mb ~ 1 Gb
MgO Based
40 nm (256 Mb)
28 nm/22 nm FDX (1 Gb, Dev.)



EVERSPIN TECHNOLOGIES

20nm, 1.5 TB, PCIe NVMe



20nm, 64 GB, PCIe

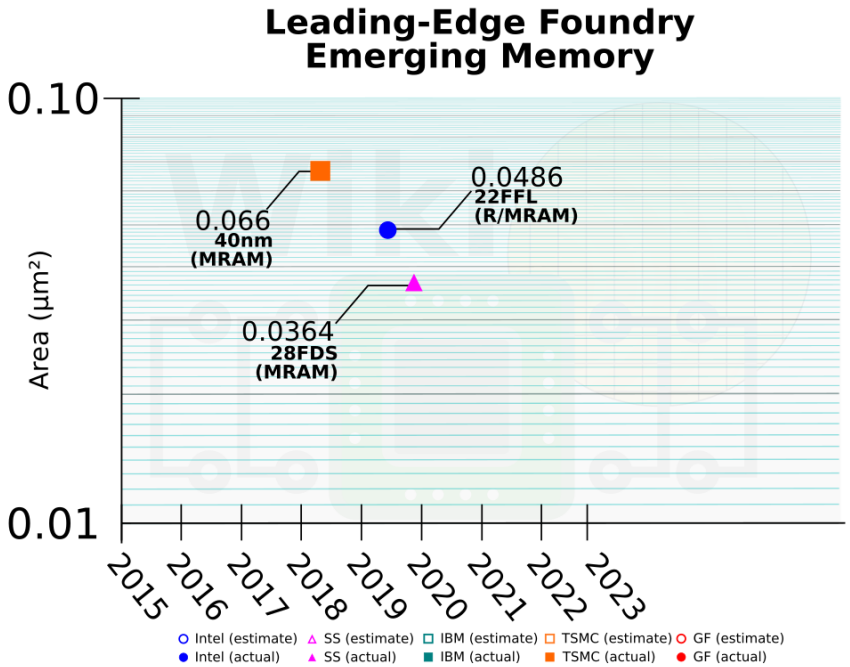
intel

We anticipate seeing new memory products throughout 2019 and into 2020, including Samsung's 28 nm FDSOI STT-MRAM, TSMC eSTT-MRAM and eReRAM, Intel 22FFL STT-MRAM, Micron XPoint QunantX and Panasonic/UMC 28nm FDX ePCM.

The Way for Commercialization 2019

Intel Expands 22FFL With Production-Ready RRAM and MRAM on FinFET

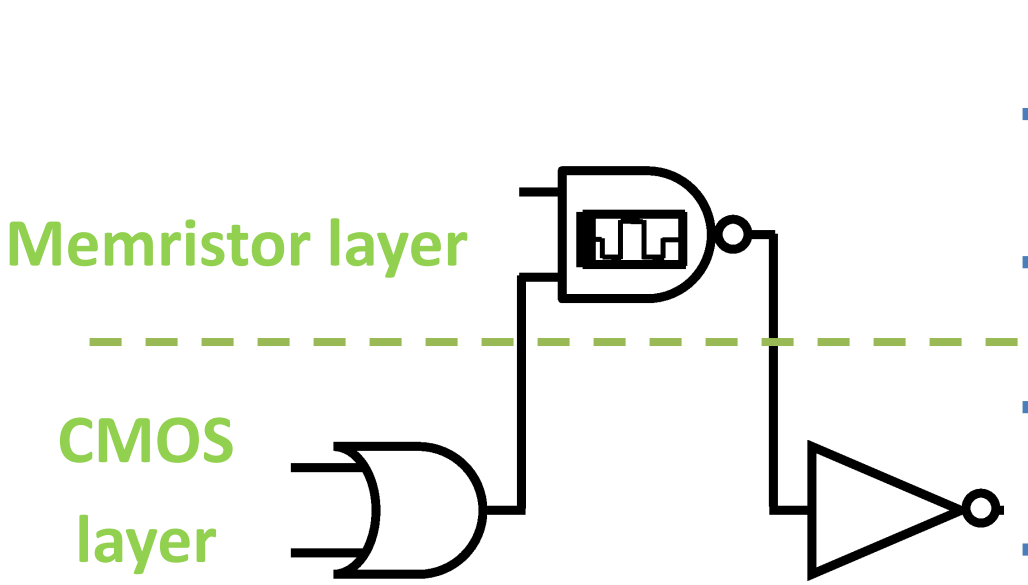
David Schor Process Technologies, VLSI 2019 October 18, 2019

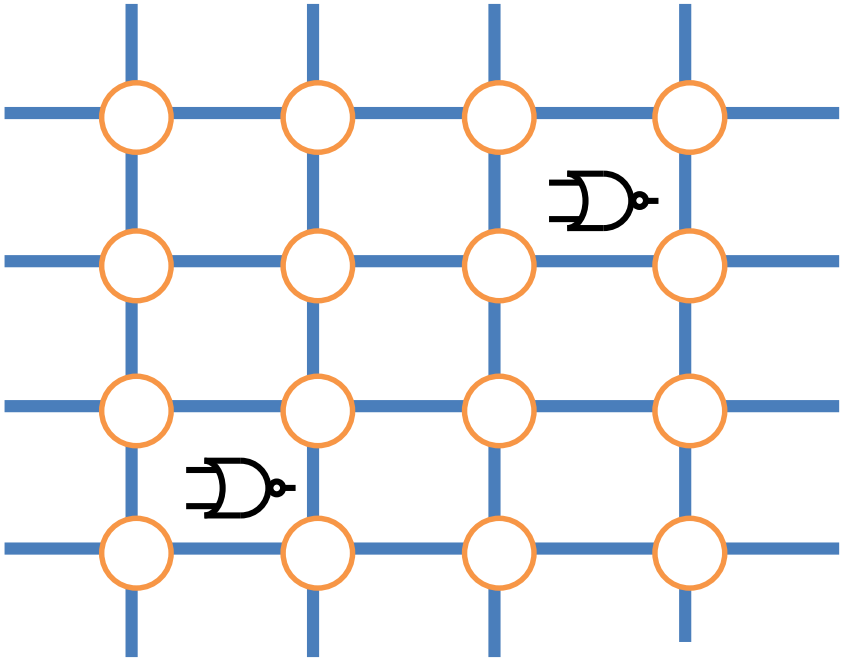


Agenda

- The need for Processing-in-Memory
- Memristors and Memristive Memory
- **Memristive logic classification**
- Out-of-memory logic
- Near-memory logic
- Real in-memory logic
- Memristor for HW security

Logic with Memristors





Integrating memristors with standard logic

Beyond Moore

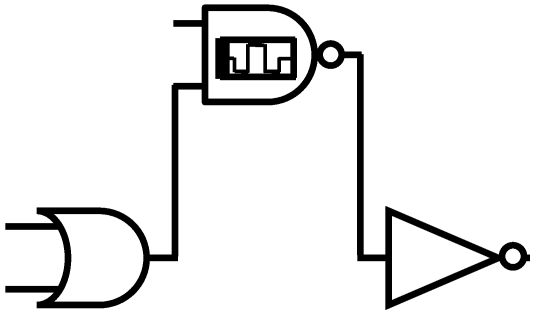
Reconfigurable Logic

Logic within memory

Beyond von Neumann

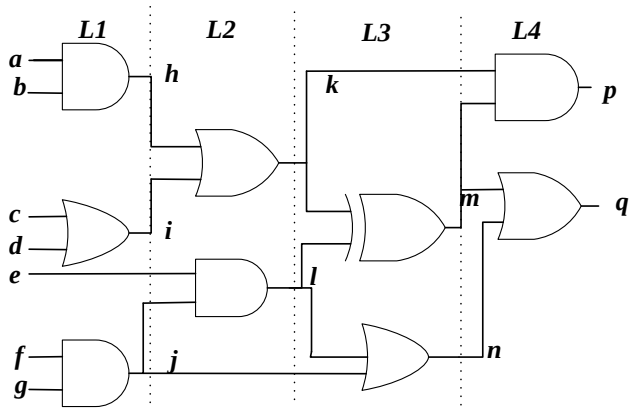
Numerous Logic Families
Need Classification!

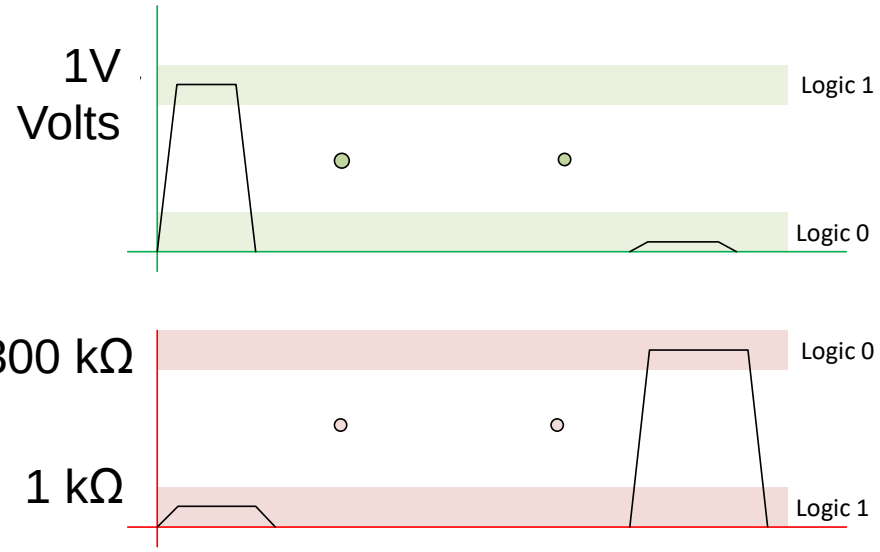
- Statefulness: representation of data
- Proximity of computation: integration of processing and storage
- Flexibility: variety of logic operations using the same circuit



State Variable

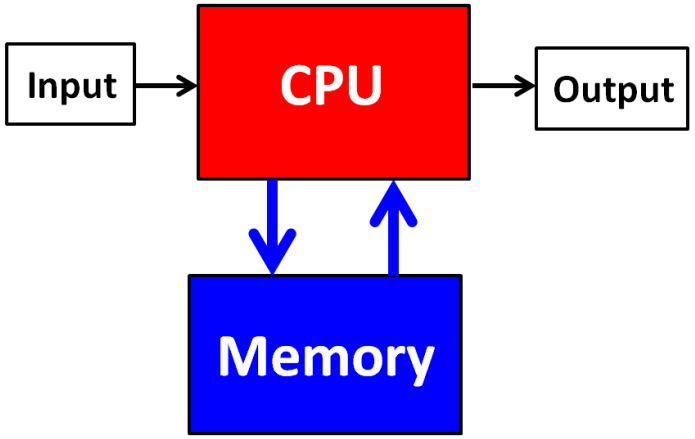
- CMOS logic -> voltage
- Memristive logic -> voltage and resistance
- **Stateful logic family**: resistance is the state variable





Proximity of Computation

- **In-memory**
- **Near-memory**
- **Out-of-memory**

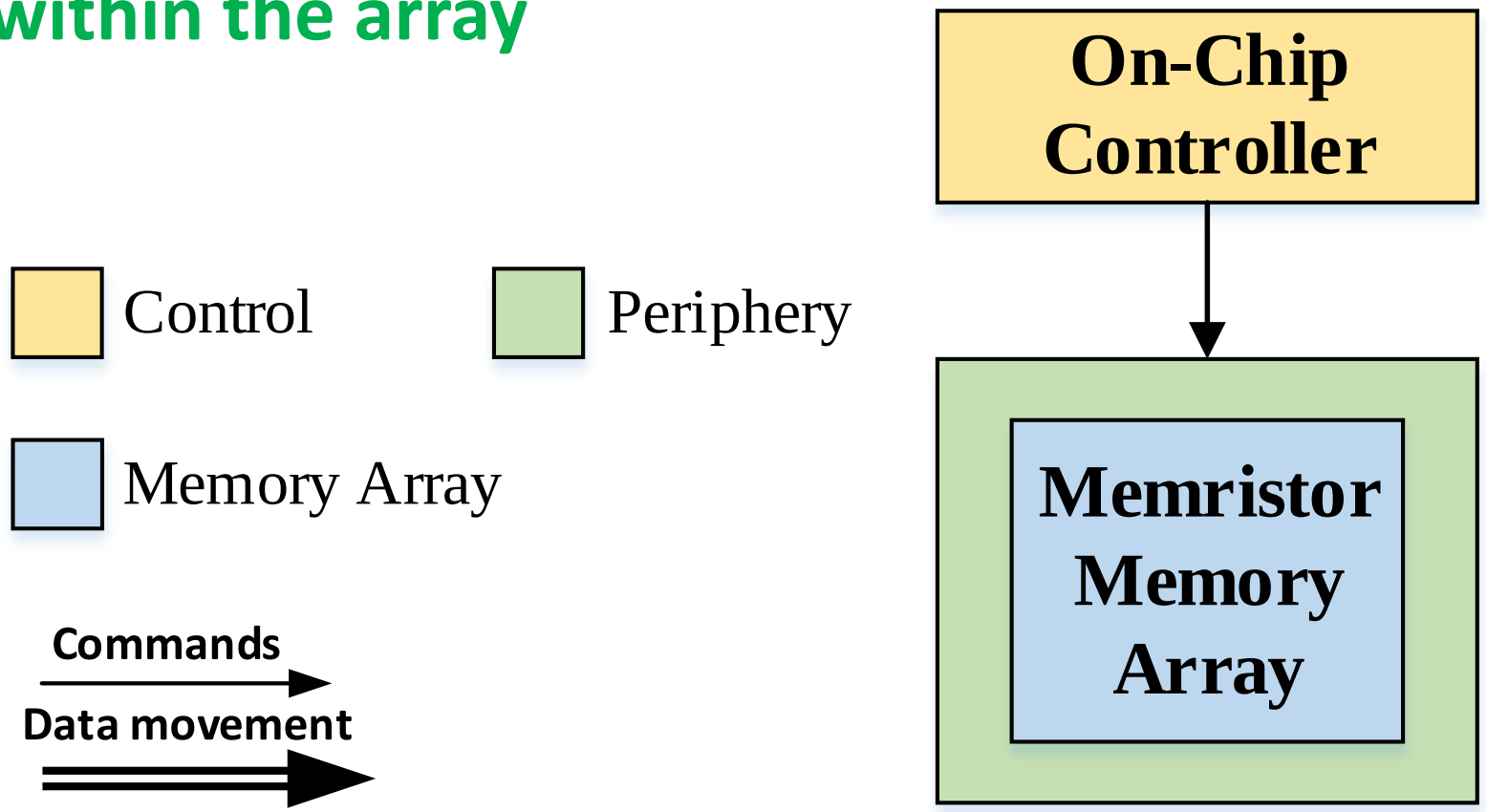


- Defined by where the computation is performed
- What is a memory array?

In-Memory Computing

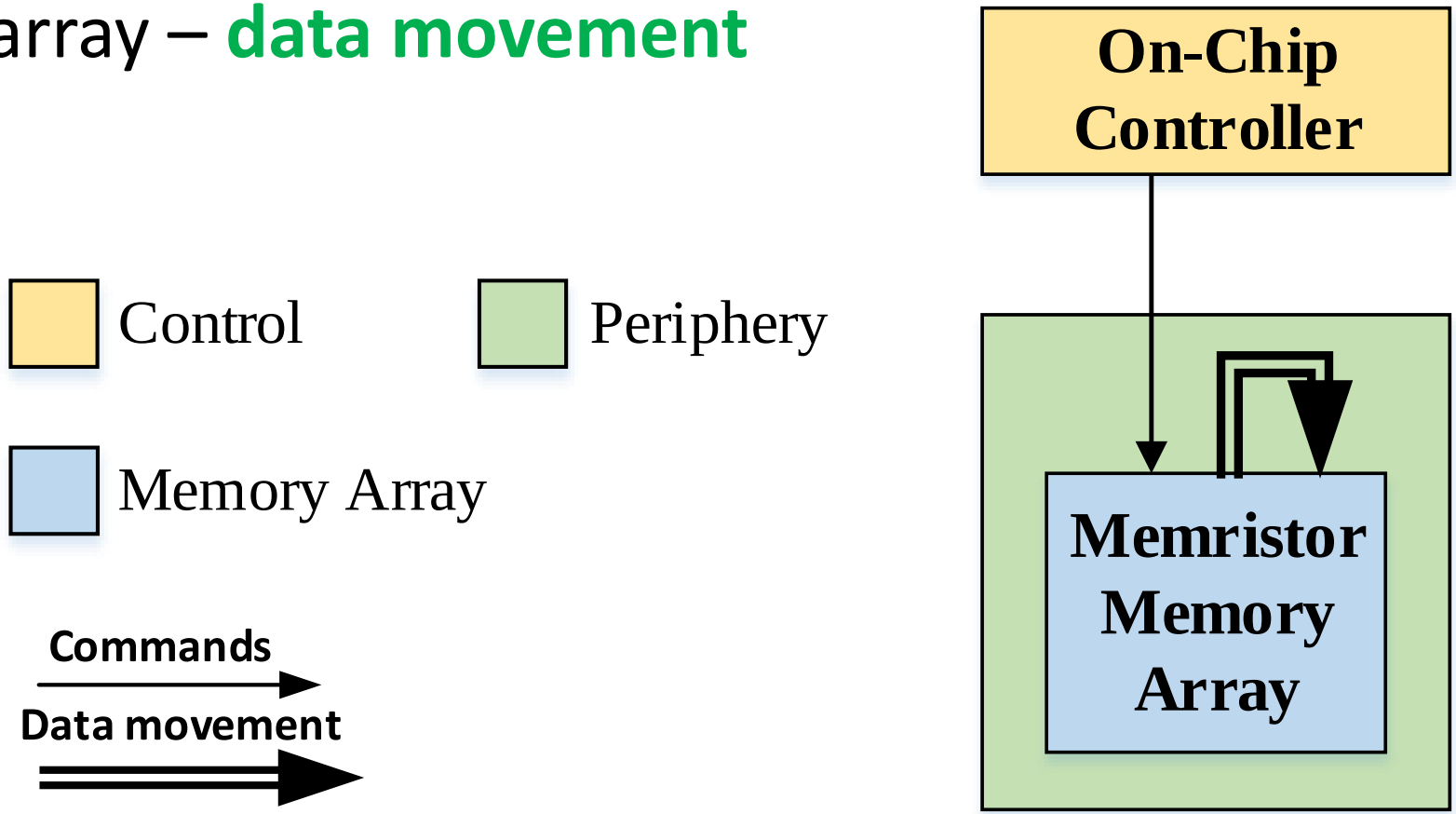
- All of the computation is performed

  within the array



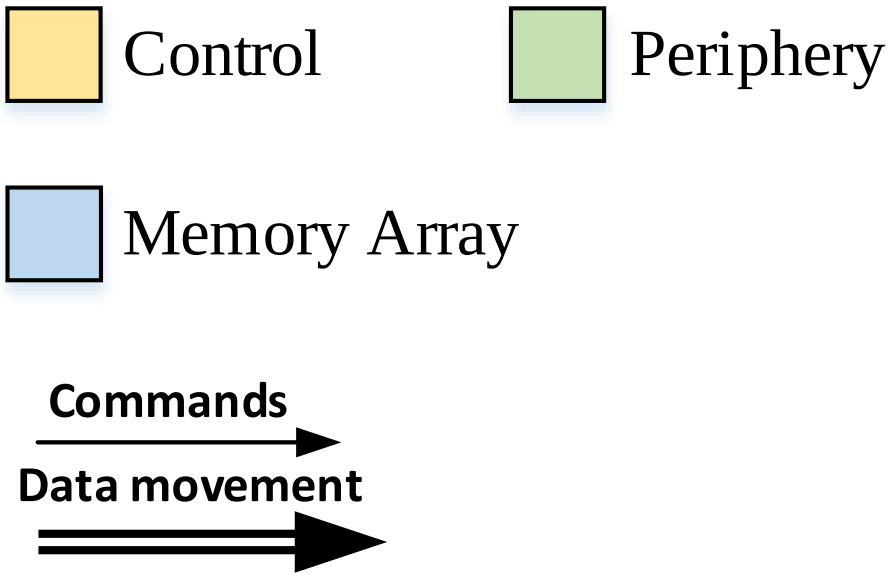
Near-Memory Computing

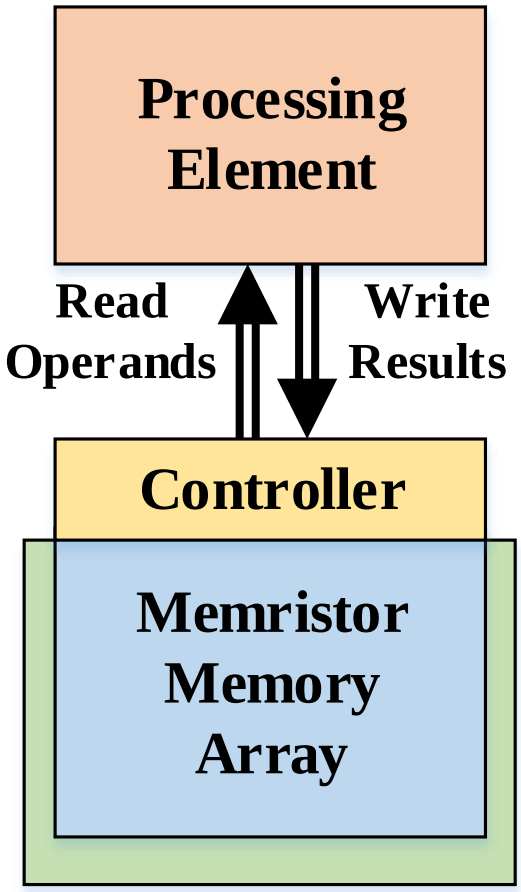
- Some of the computation is out of the memory array – **data movement**



Out-of-Memory Computing

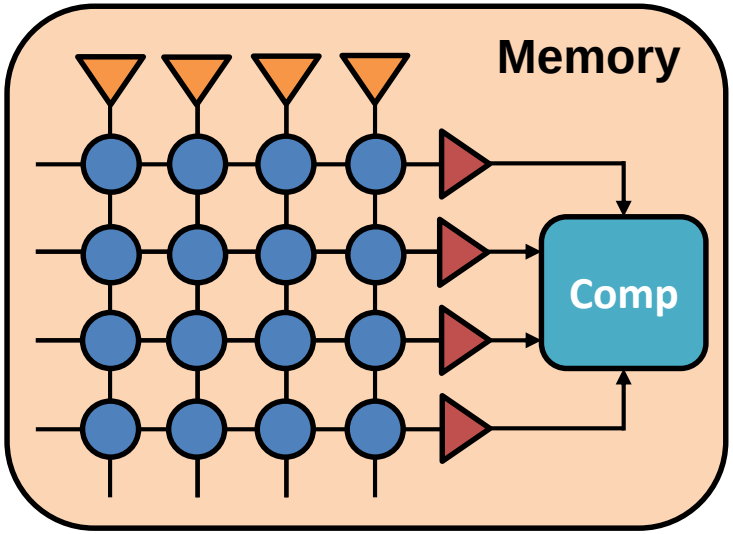
- All of the computation is out of the memory array – **data movement, dedicated processing units**





Proximity of Computation

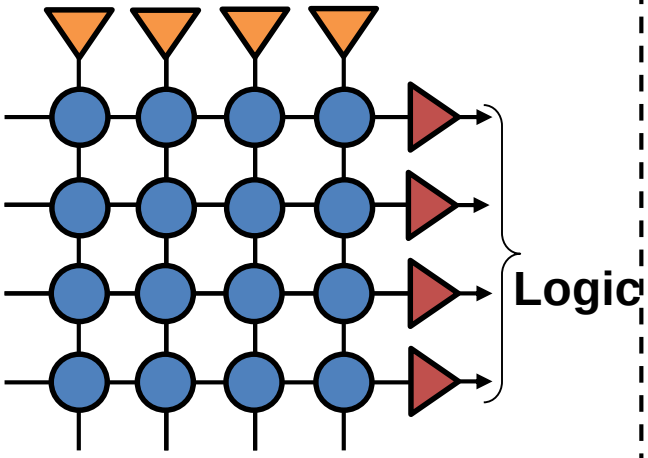
Computation using logic blocks



Dlugosch *et al.*, *IEEE TPDS*, 2014
Oskin *et al.*, *Comput. Archit. News*, 1998
Elliott *et al.*, *IEEE Des. Test*, 1999
Gokhale *et al.*, *Computer*, 1995

OUT-

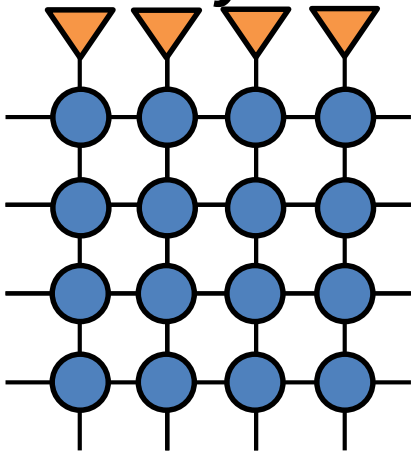
Computation using peripheral circuit



Li *et al.*, *DAC*, 2016
Seshadri *et al.*, MICRO 2017
Aga *et.al.* HPCA 2017
Eckert *et.al.* ISCA 2018

NEAR-

Computation ***using memory cells***



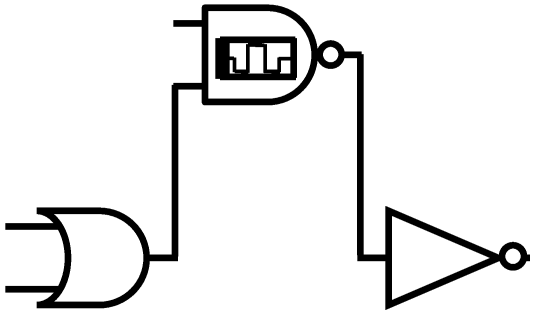
Kvatinsky *et al.*, *TCAS-II*, 2014
Talati *et al.*, TNANO 2016

IN-

J. Reuben, R. Ben Hur, N. Wald, N. Talati, A. Haj Ali, P.-E. Gaillardon, and S. Kvatinsky, "Memristive Logic: A Framework for Evaluation and Comparison", PATMOS 2017

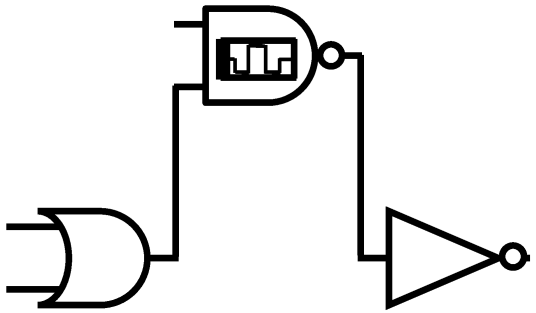
Flexibility

- How sequence of **control signals** can execute different functions?
- What is the **basic logic function**?
- What is the basic structure?



Classification Summary

- Statefulness: representation of data
- Proximity of computation: integration of processing and storage
- Flexibility: variety of logic operations using the same circuit

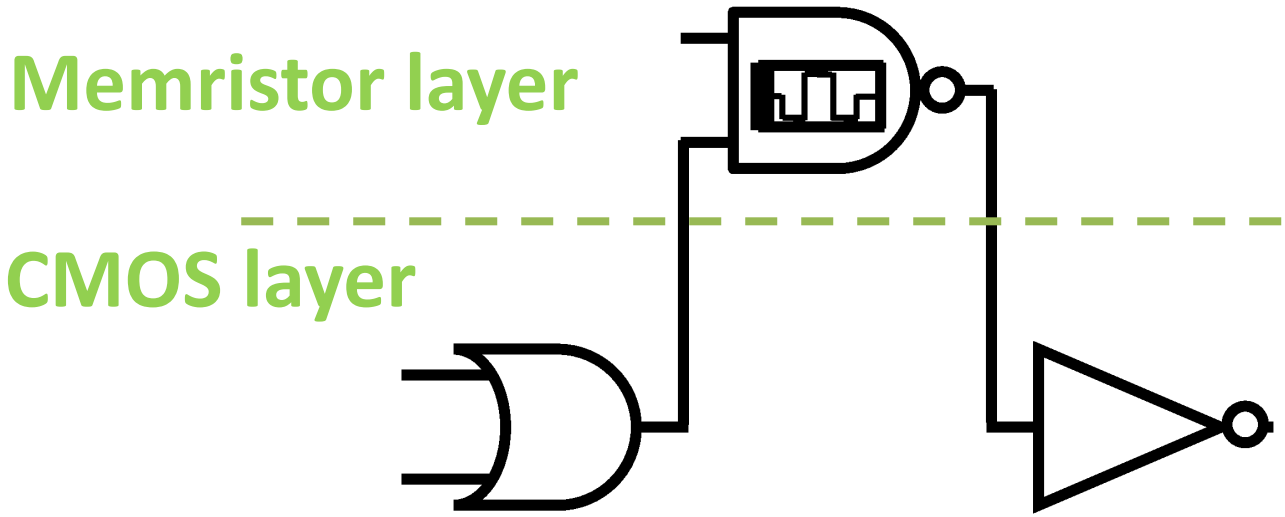


J. Reuben et al., “Memristive Logic: A Framework for Evaluation and Comparison,” PATMOS 2017
J. Reuben et al., “A Taxonomy and Evaluation Framework for Memristive Logic",
Handbook of Memristor Networks, Springer 2019

Agenda

- The need for Processing-in-Memory
- Memristors and Memristive Memory
- Memristive logic classification
- **Out-of-memory logic**
- Near-memory logic
- Real in-memory logic
- Memristor for HW security

Hybrid Memristor-CMOS Logic



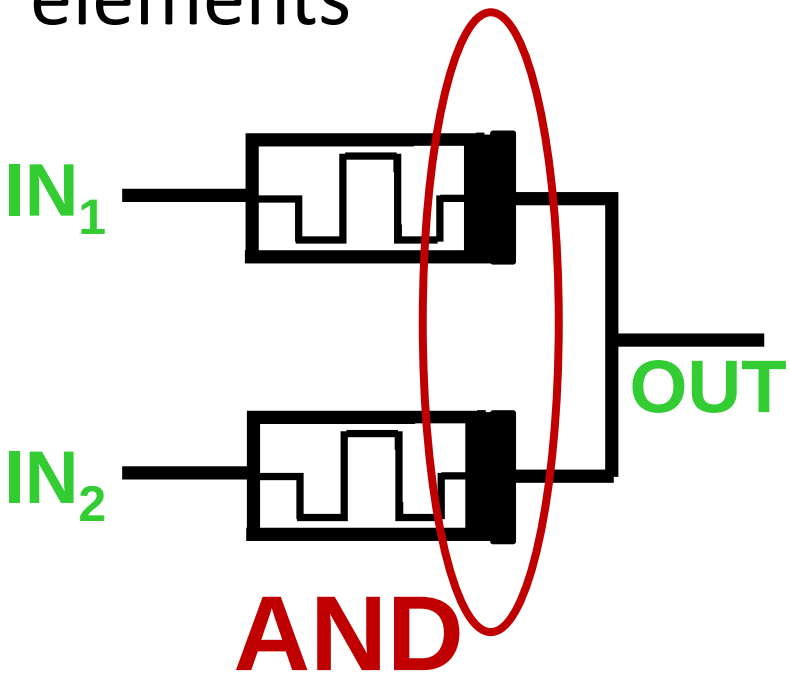
Save die area

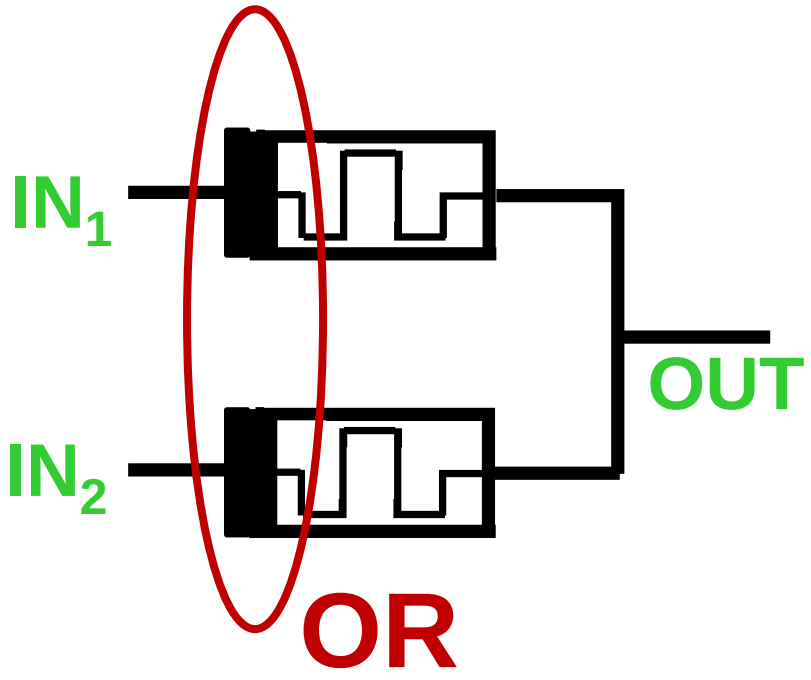
More logic on die

Beyond Moore

Memristor Ratioed Logic (MRL)

- Similar to CMOS logic
- Using CMOS for inversion and amplification
- Memristors operate only as computational elements





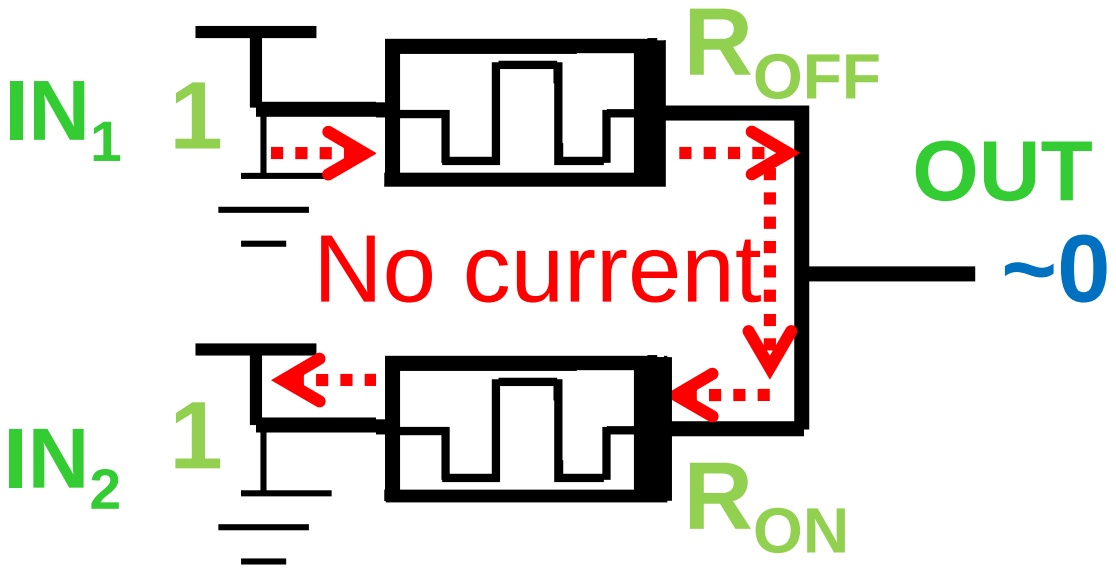
S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "MRL – Memristor Ratioed Logic," CNNA 2012

AND Operation

$$V_{OUT} = V_{CC} \cdot \frac{R_{ON}}{R_{ON} + R_{OFF}} \approx V_{CC} \cdot \frac{R_{ON}}{R_{OFF}} << V_{CC}$$

IN_1	IN_2	AND
0	0	0
0	1	0
1	0	0
1	1	1

Increase resistance



Decrease resistance

$R_{OFF} >> R_{ON}$

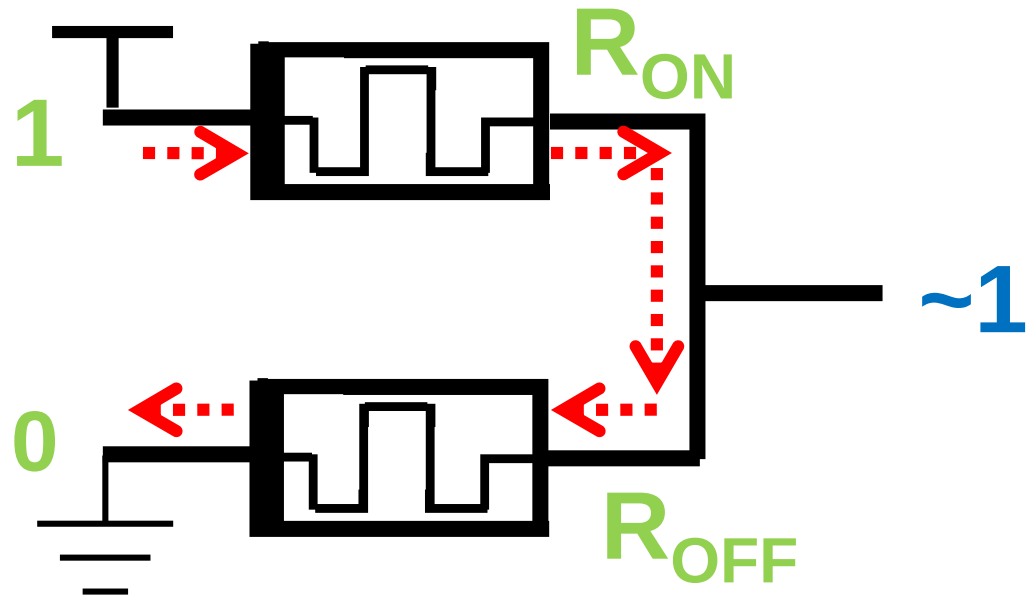
OR Operation

$$V_{OUT} = V_{CC} \cdot \frac{R_{OFF}}{R_{ON} + R_{OFF}} \approx V_{CC}$$

$R_{OFF} >> R_{ON}$

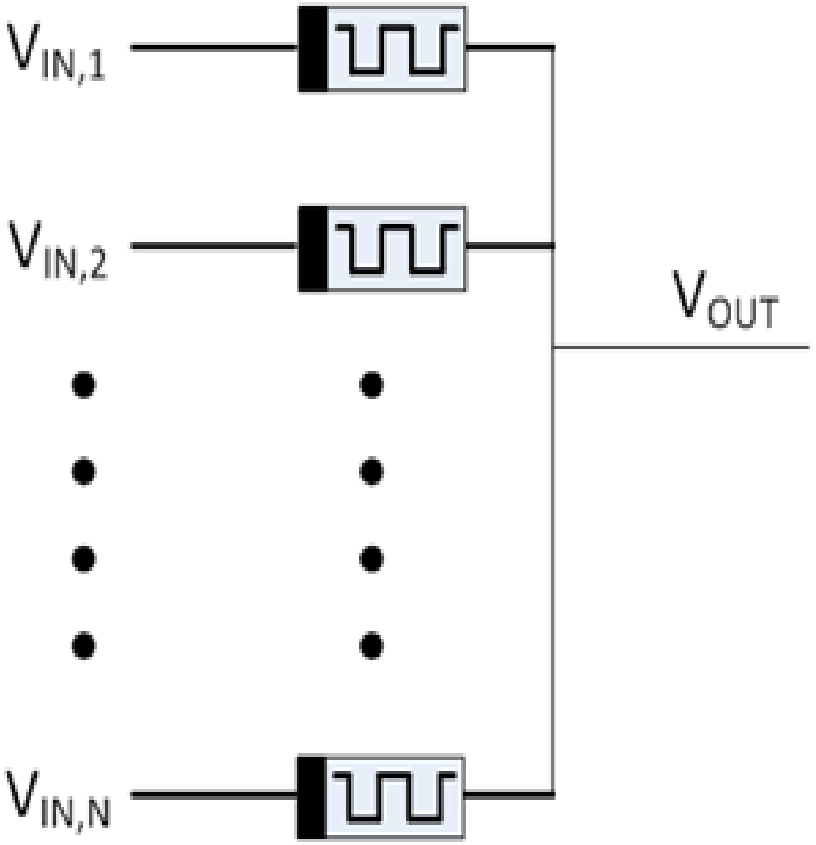
IN_1	IN_2	OR
0	0	0
0	1	1
1	0	1
1	1	1

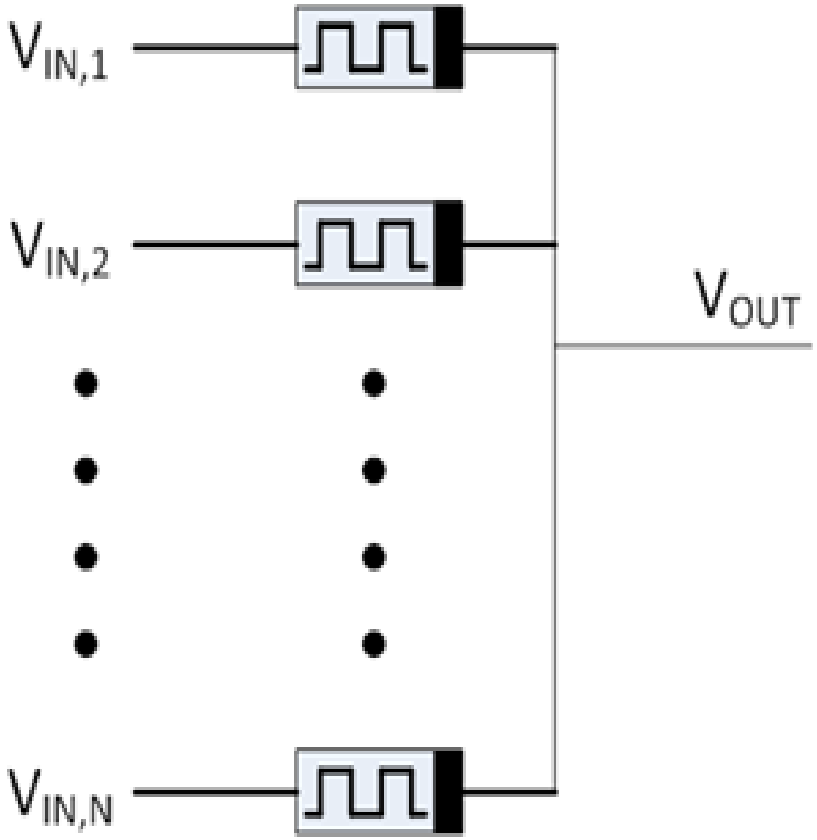
Decrease resistance



Increase resistance

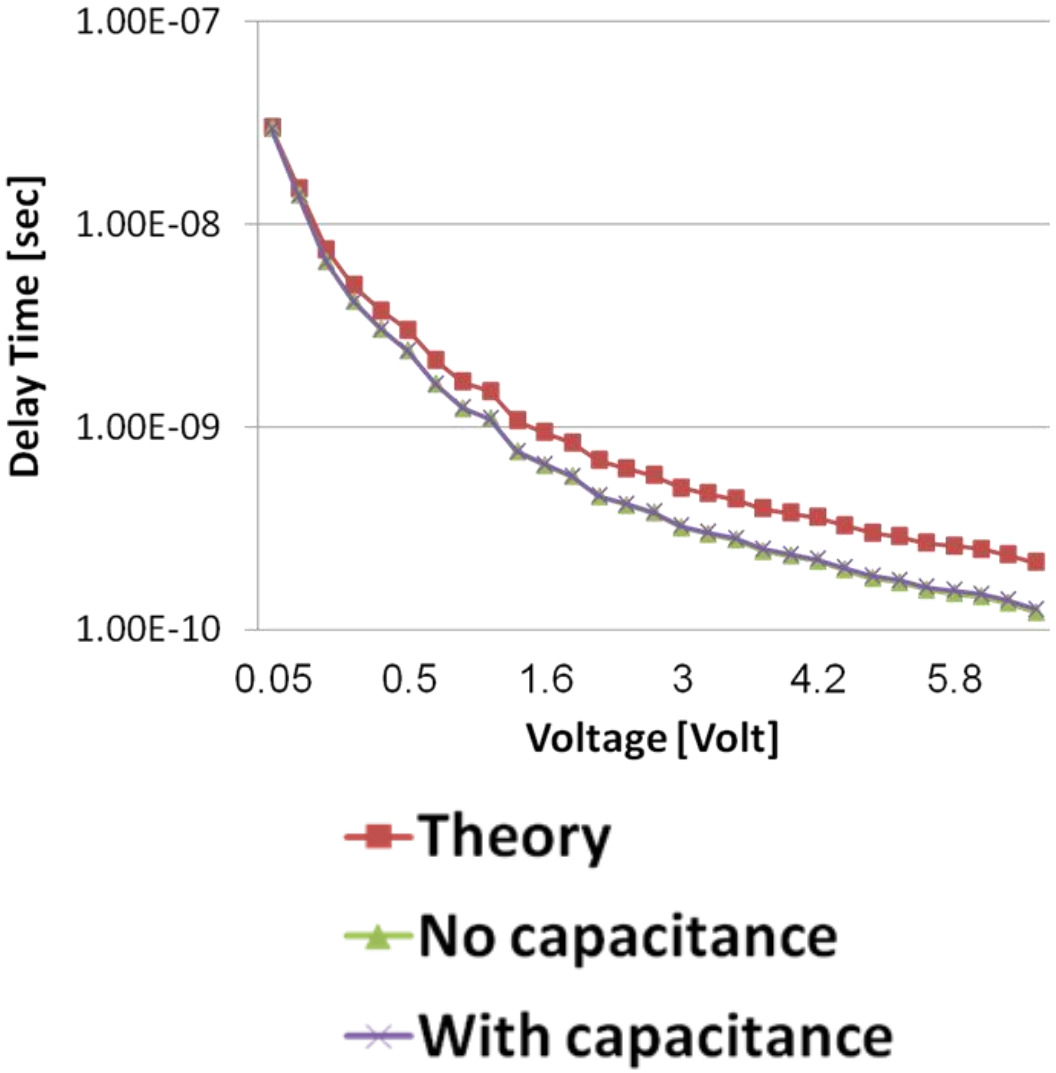
N-Input MRL



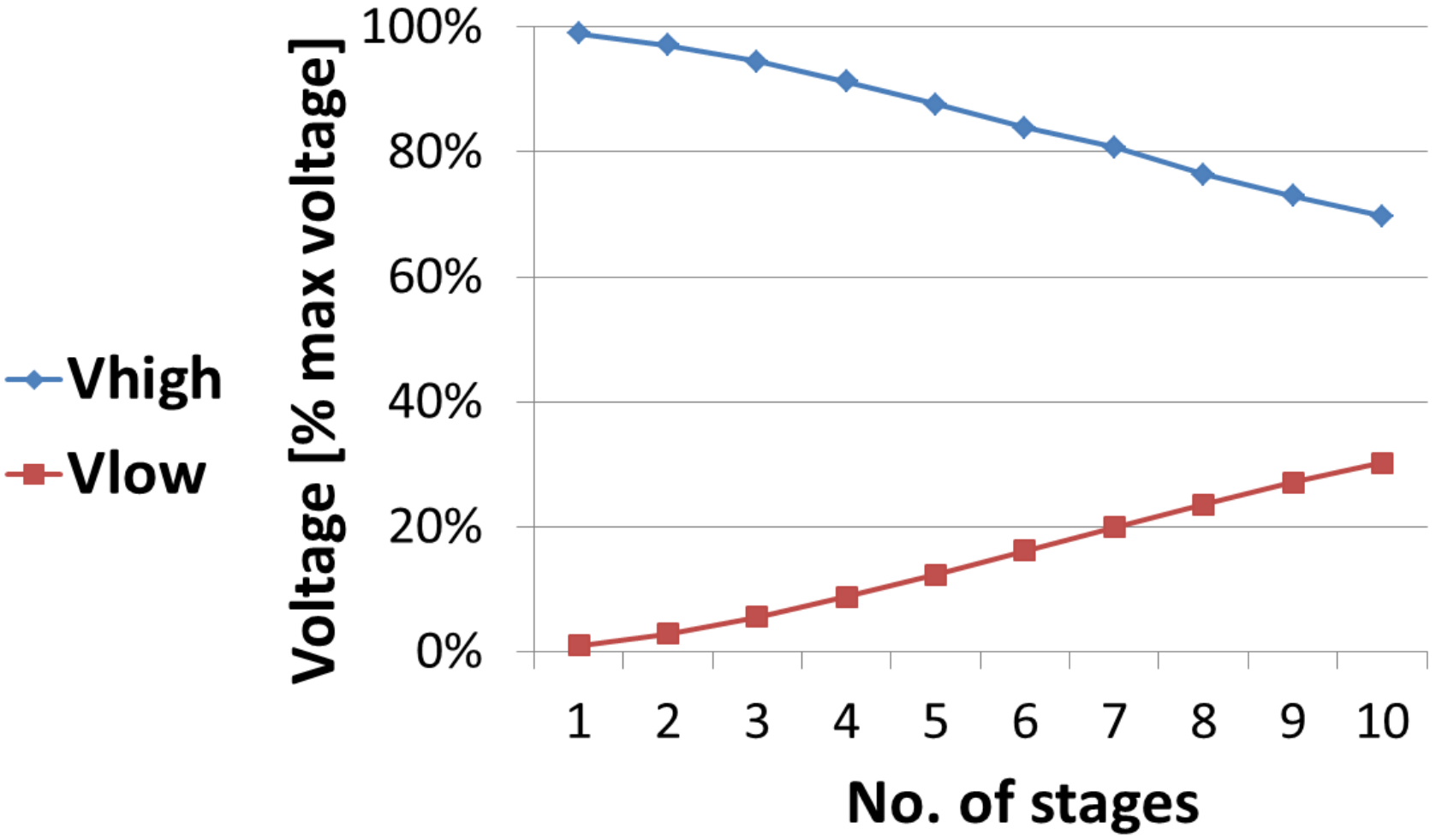


MRL Delay

$$T = \frac{kQ'R_{OFF}}{V_{high}}$$

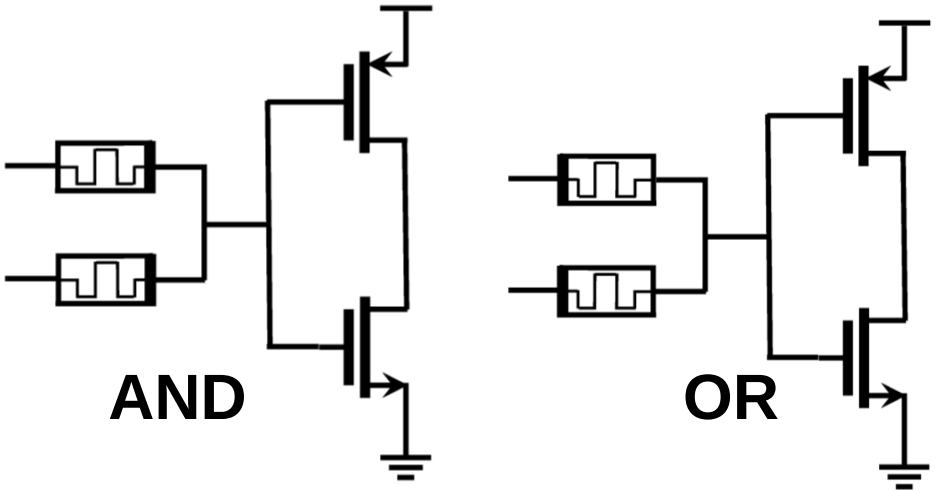


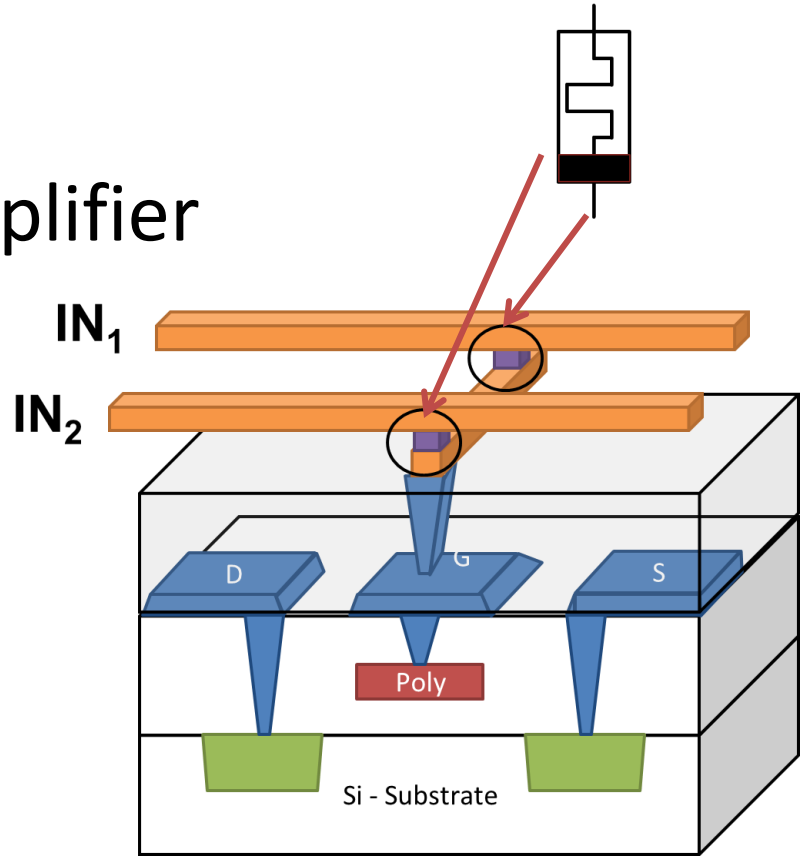
Need for Amplification



Integration with CMOS Inverters

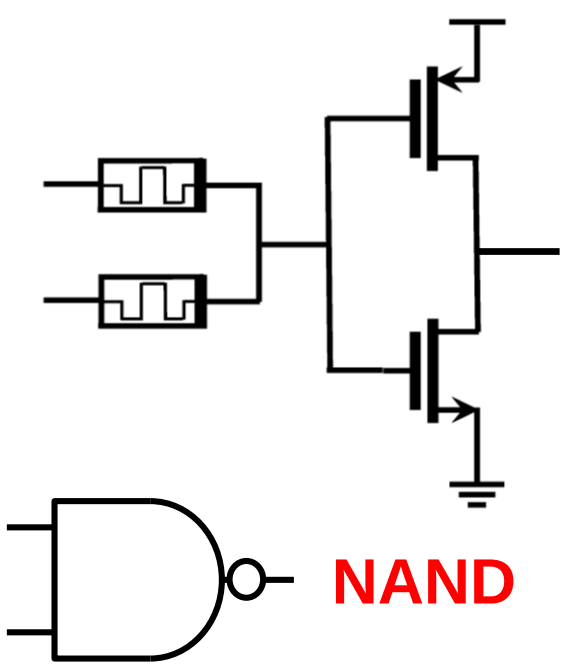
- Memristors can be fabricated with CMOS
- Input/output are voltages – as in standard CMOS logic
- CMOS gates have gain - amplifier

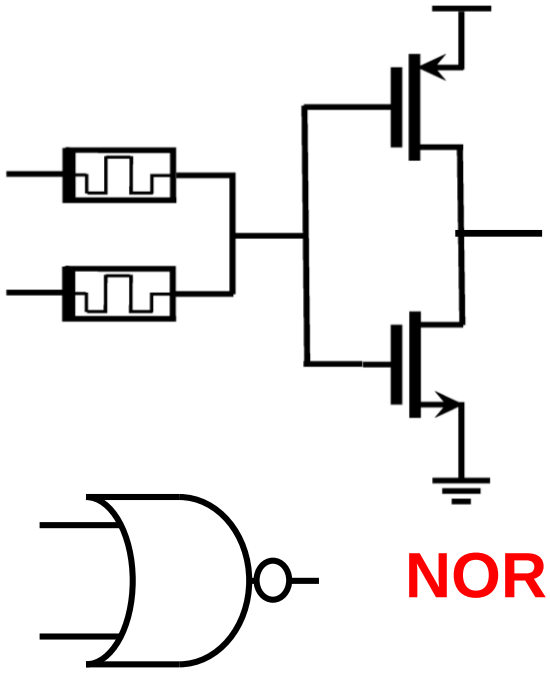




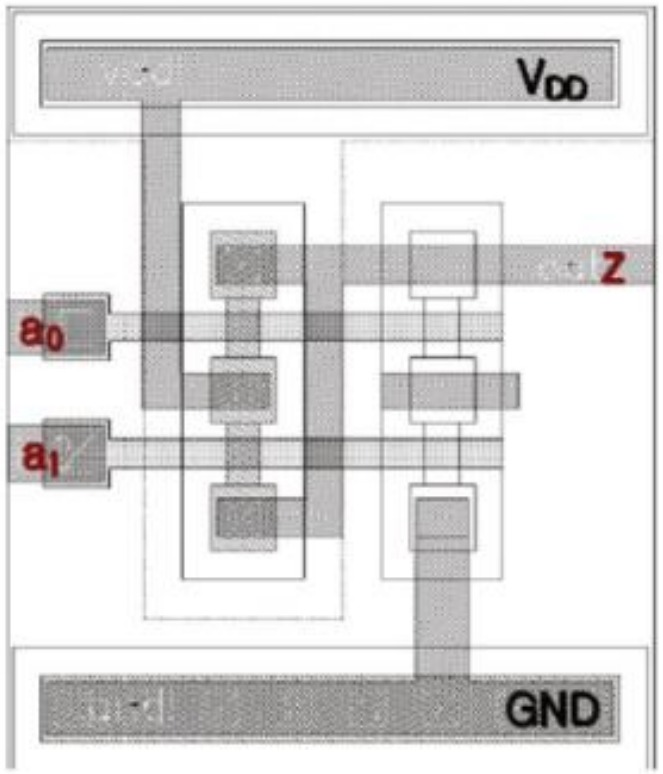
Standard Cell Approach

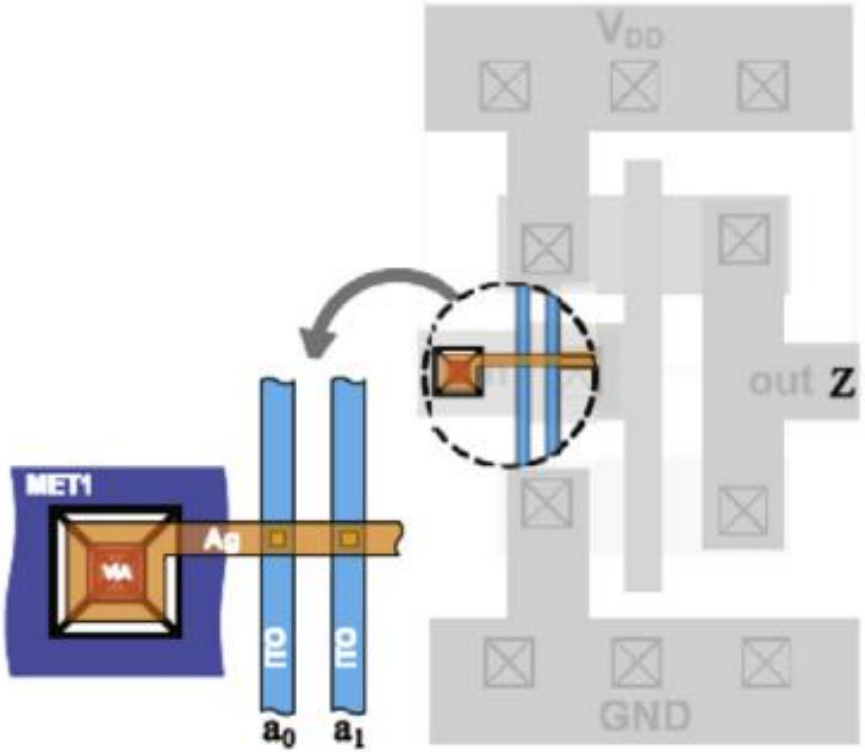
- MRL standard cells – NAND and NOR
- Robust - no current leakage between stages
- Two CMOS-memristor cross-layer connections per cell – consume area and power





NAND Physical Design





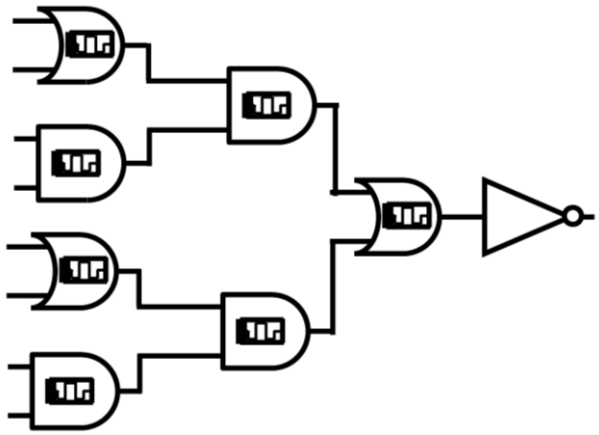
CMOS

MRL

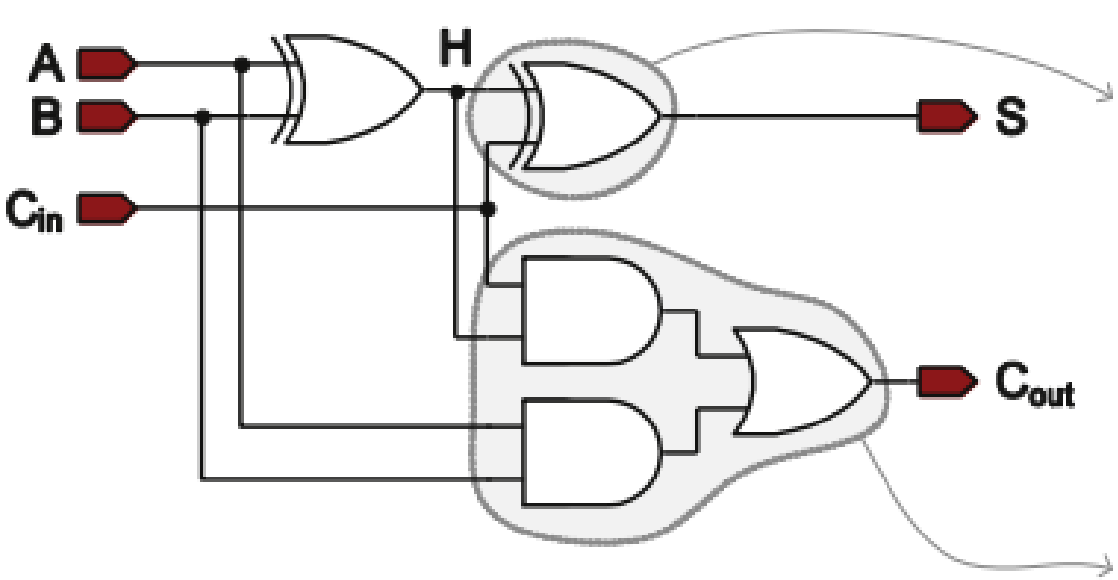
K. Cho, S.-J. Lee, and K. Eshraghian, "Memristor-CMOS Logic and Digital Computational Components," MEJ 2015

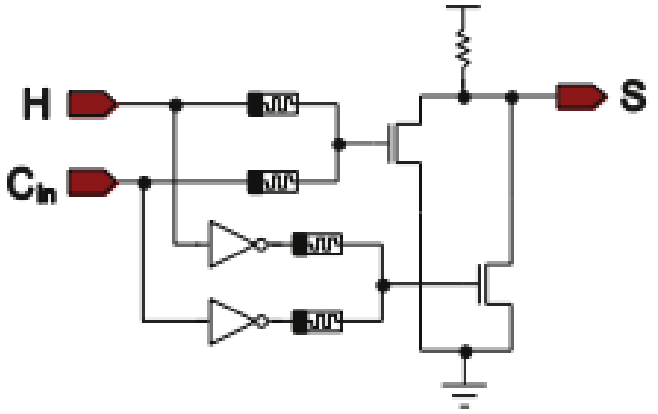
Optimized Approach

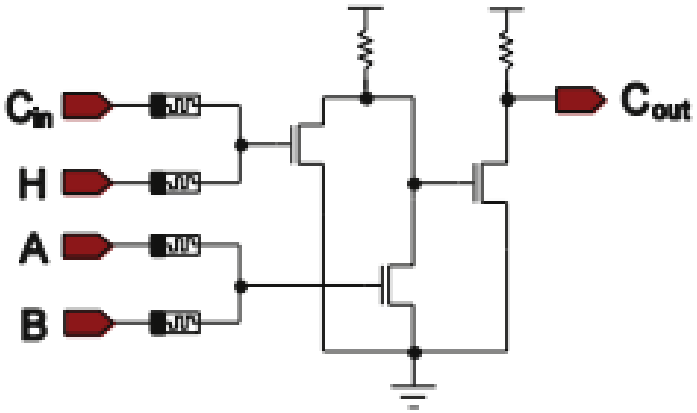
- **Target** - minimize CMOS-memristor cross-layer connections
- **Solution** – use inverters only when needed
  - Signal restoration (voltage optimization)
  - Logic function requires inversion



Different MRL Full Adders (Cho 2015)







$$S = A \oplus B \oplus C_{in}$$

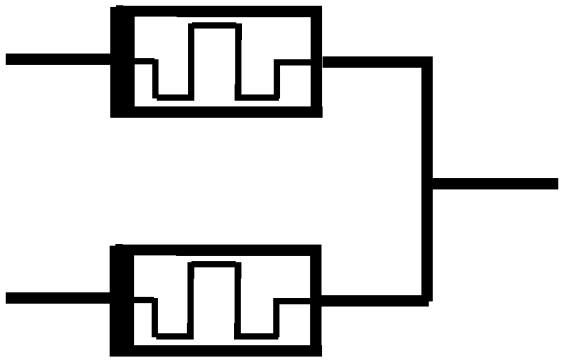
$$C_{out} = (A \cdot B) \oplus (C_{in} \cdot (A \oplus B))$$

K. Cho, S.-J. Lee, and K. Eshraghian, "Memristor-CMOS Logic and Digital Computational Components," MEJ 2015

MRL – Summary

Beyond Moore's Law

- Hybrid CMOS-memristor logic
- CMOS inverters for signal restoration and complete logic structure
- Increase logic density
- Applications?

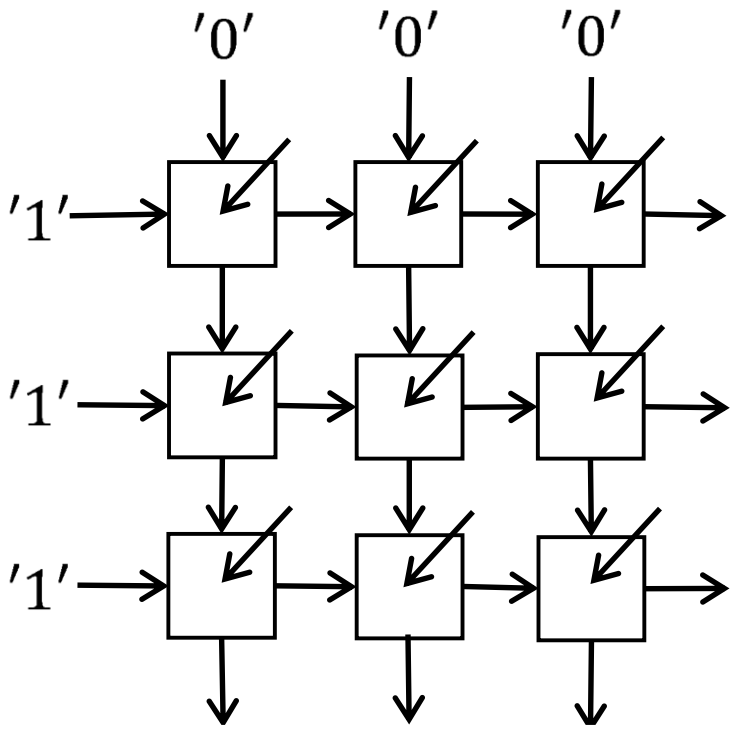


Agenda

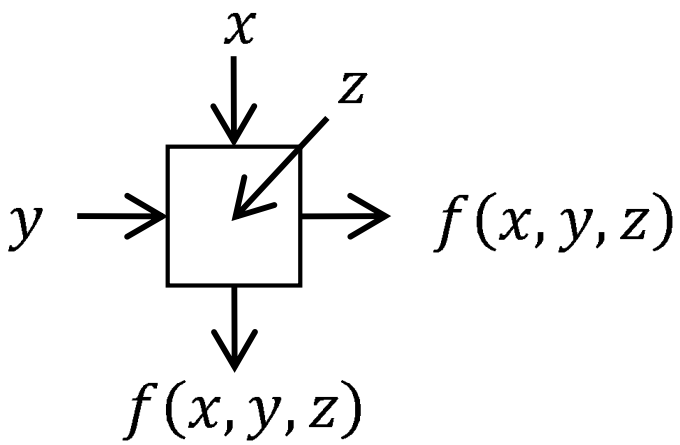
- The need for Processing-in-Memory
- Memristors and Memristive Memory
- Memristive logic classification
- Out-of-memory logic
- **Near-memory logic**
  - **Akers**
  - Associative Processors
  - Neuromorphic
- Real in-memory logic
- Memristor for HW security

Akers Logic Arrays 1972

- Executes any Boolean function



3 x 3 Akers array

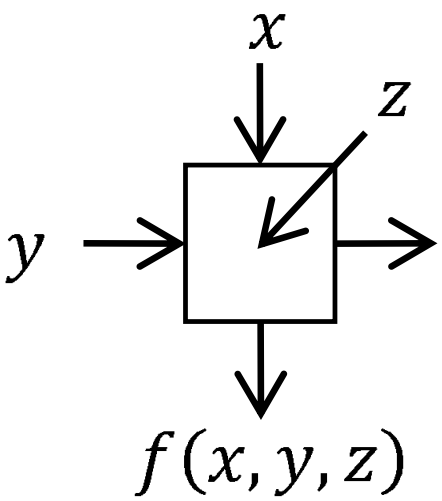


Primitive logic cell

S. B. Akers Jr., " A Rectangular Logic Array," *IEEE Trans. on Computing*, August 1972

Primitive Logic Cell

- Three input majority gate
- The input combinations (1,0,0) and (1,0,1) never occur ($X \leq Y$)



$$f_0(x,y,z) = yx + yz$$

$$f_1(x,y,z) = MAJ(x,y,z)$$

$$f_2(x,y,z) = x\bar{z} + yz$$

$$f_3(x,y,z) = x + yz$$

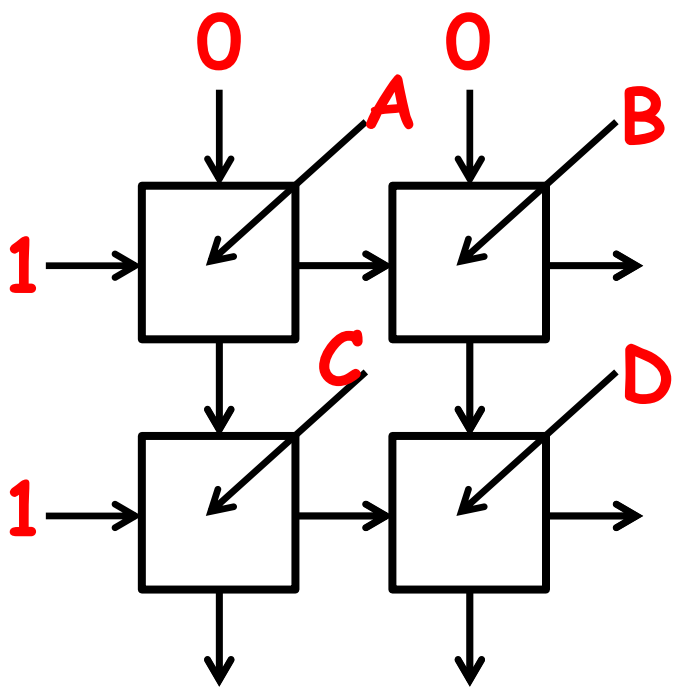
X	Y	Z	f
0	0	0	0
1	0	0	-
0	1	0	0
1	1	0	1
0	0	1	0
1	0	1	-
0	1	1	1
1	1	1	1

The Input Matrix

- If **A**, **B**, **C**, **D** are the input entries to the logic array, we say that

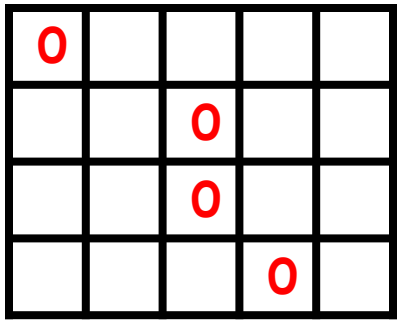
A	B
C	D

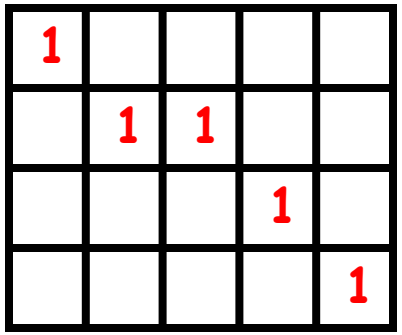
is the **input matrix**

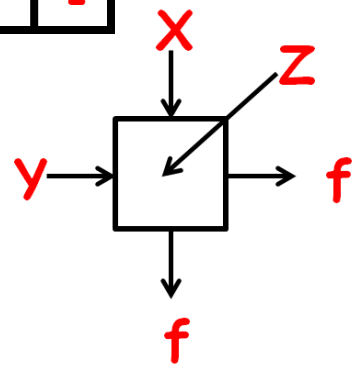


Basic Properties

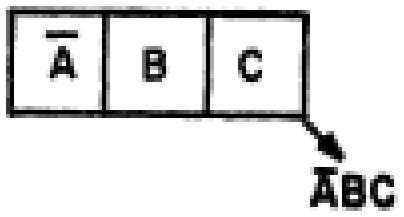
- If there is a 0-path over the rows, then the array output is zero
- If there is a 1-path over the columns, then the array output is one

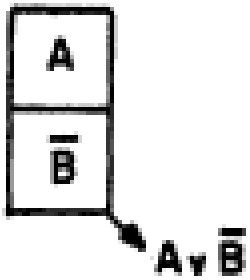


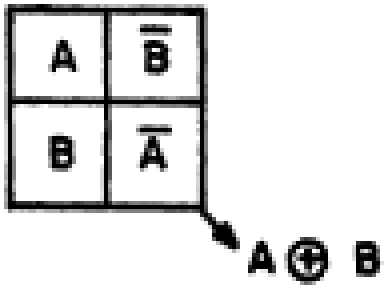


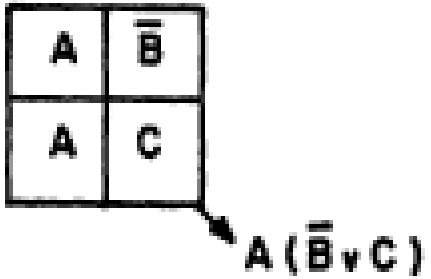


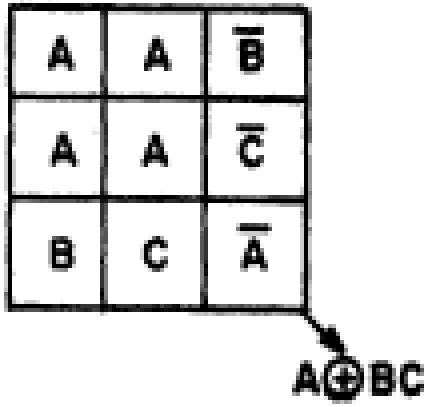
Various Boolean Functions with Akers Arrays

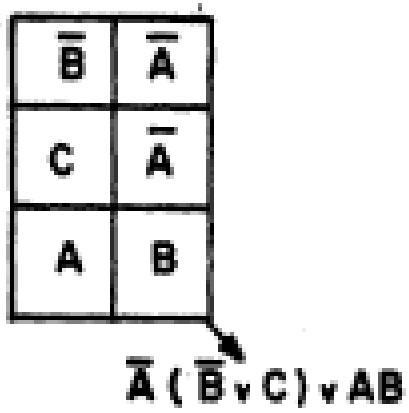




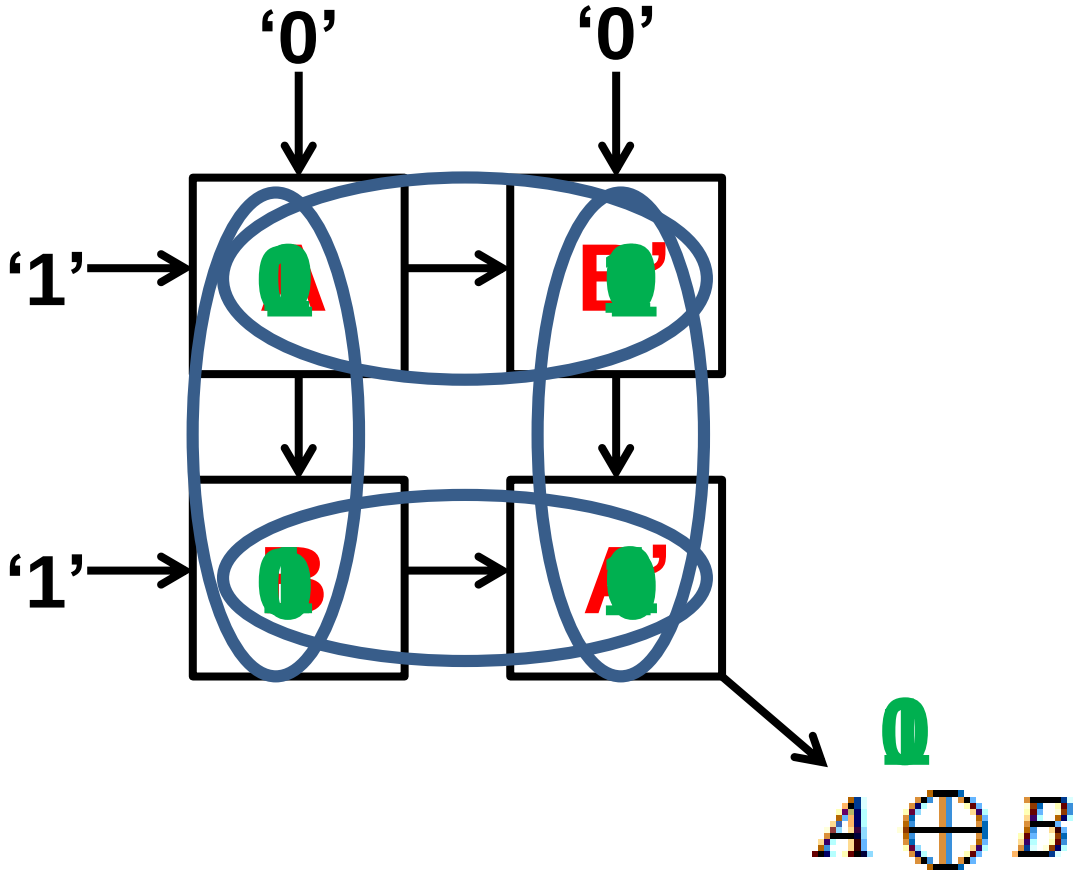








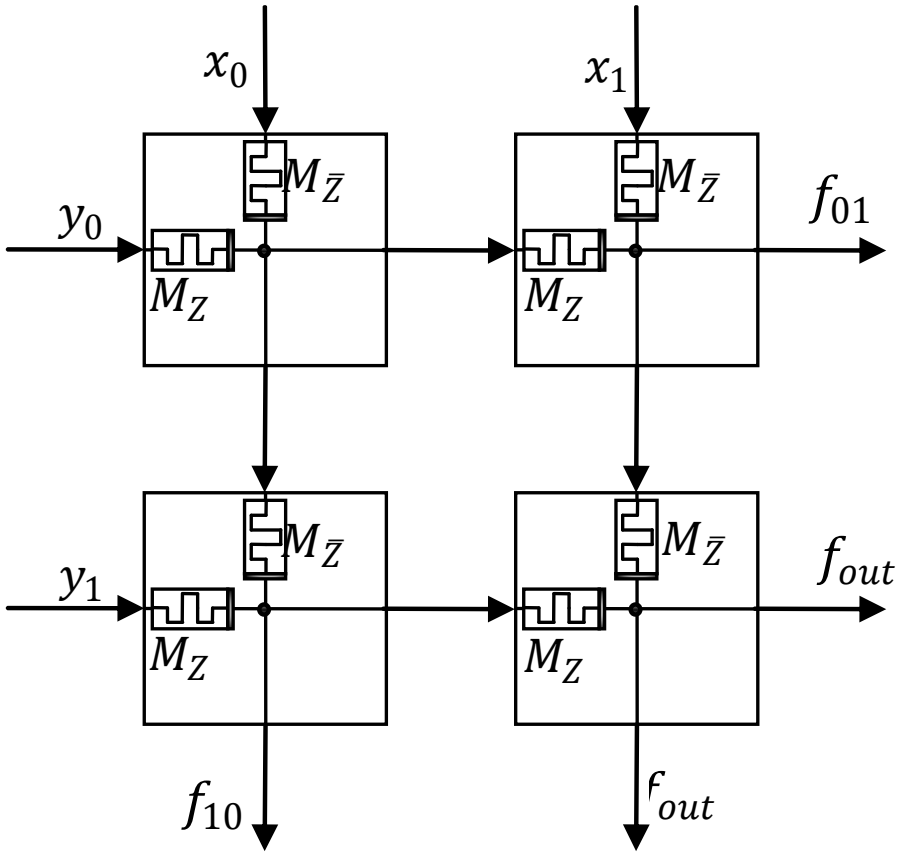
XOR Akers Array



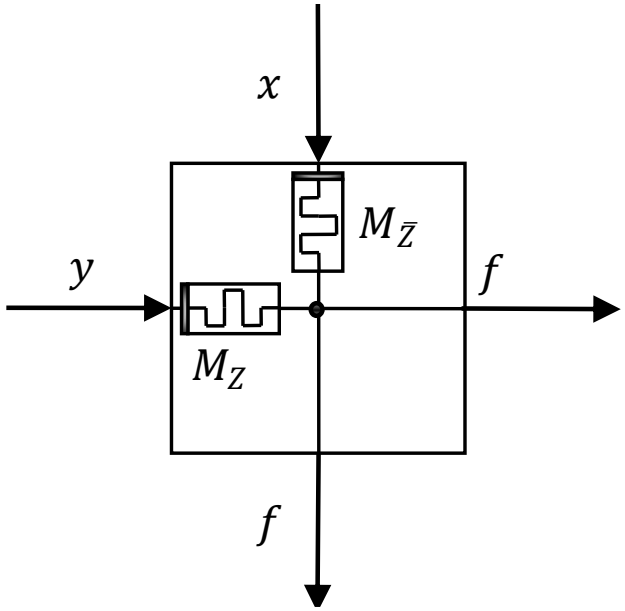
A=0

B=0

Memristive Akers Arrays



2 x 2 memristive Akers array

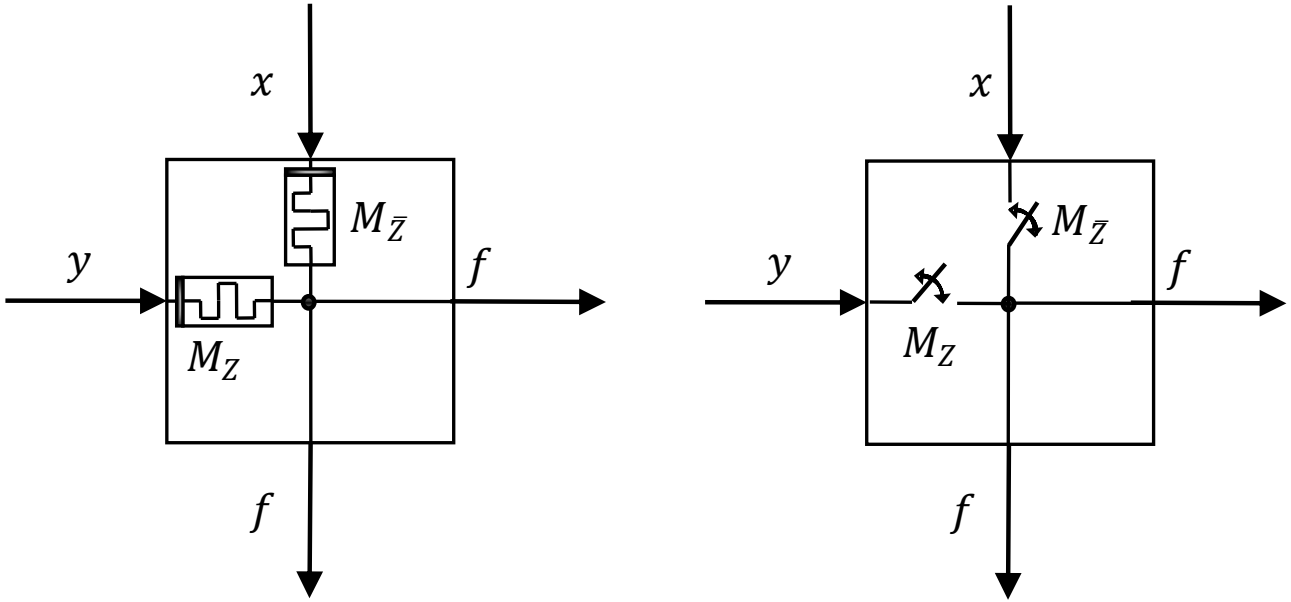


Primitive memristive logic cell

Y. Levy, J. Bruck, Y. Cassuto, E. G. Friedman, A. Kolodny, E. Yaacobi, and S. Kvatinsky, "Logic Operation in Memory Using a Memristive Akers Array," *Microelectronics Journal*, Nov. 2014

Primitive Memristive Logic Cell

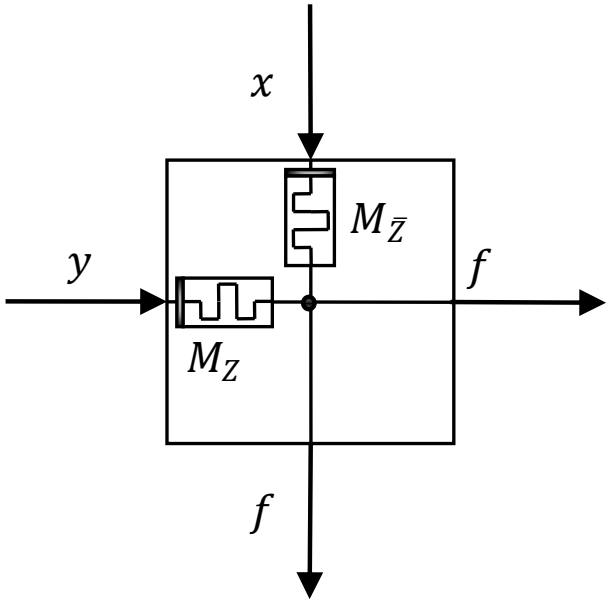
- Based on complementary memristor (CRS)
- Implementing $f_2(x, y, z)$

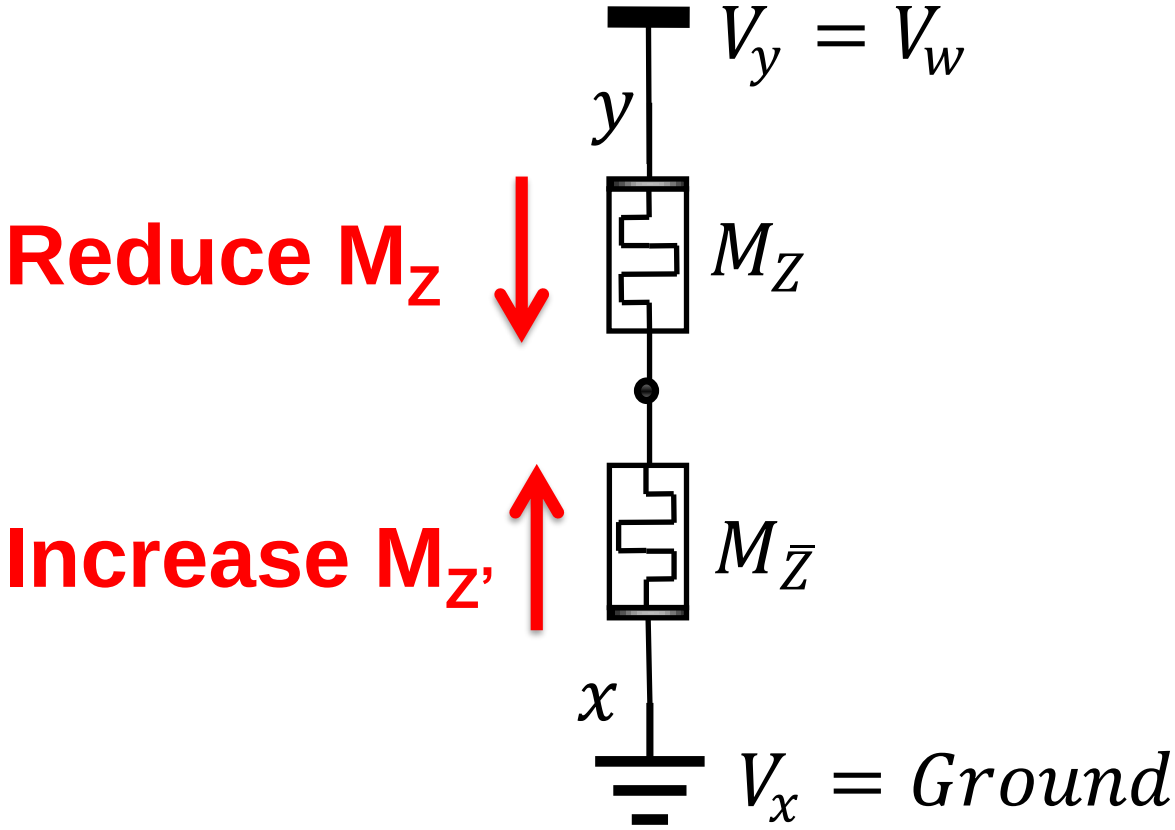


$$f_2(x, y, z) = x\bar{z} + yz$$

Write to the Primitive Cell

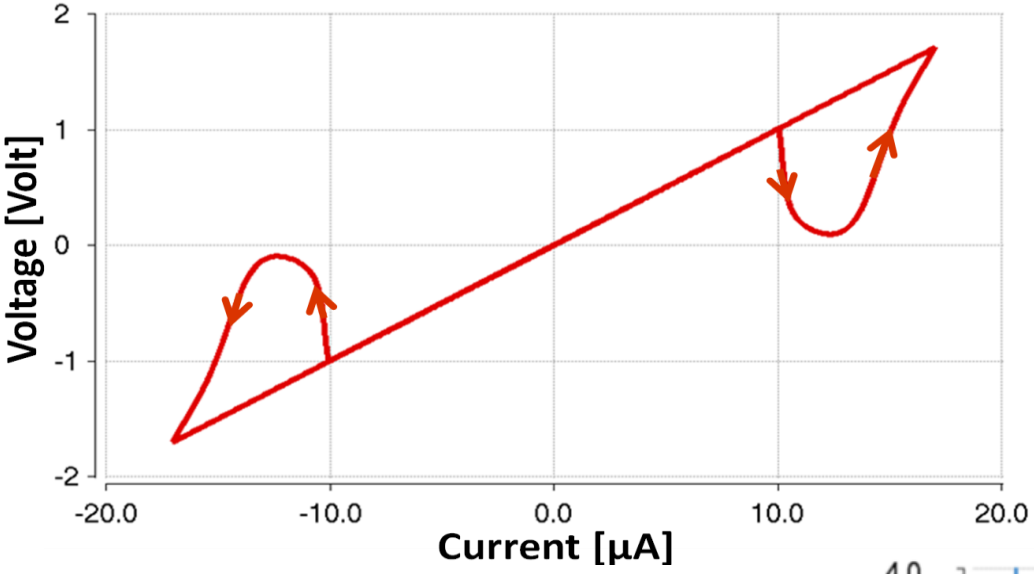
- Write to both memristors

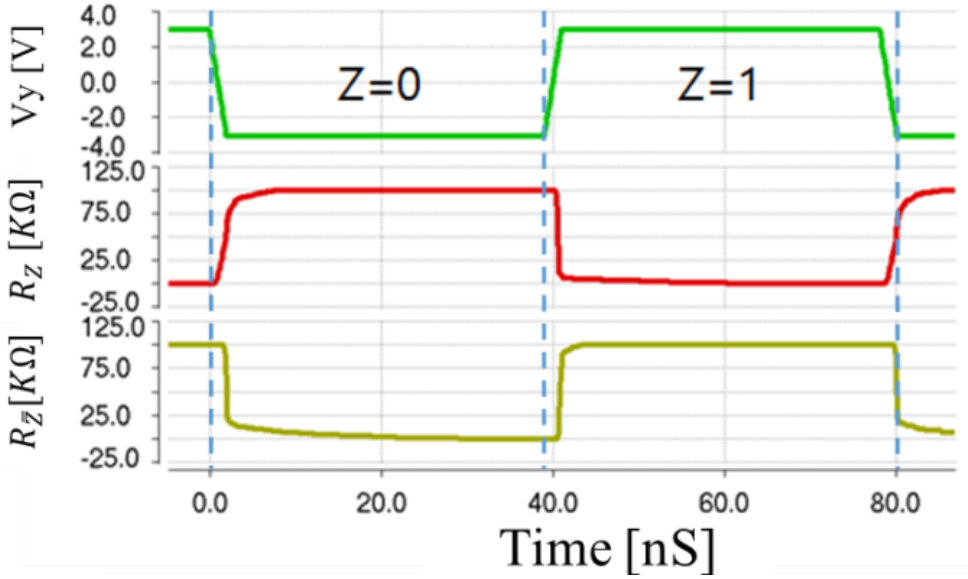




Write logical '1'

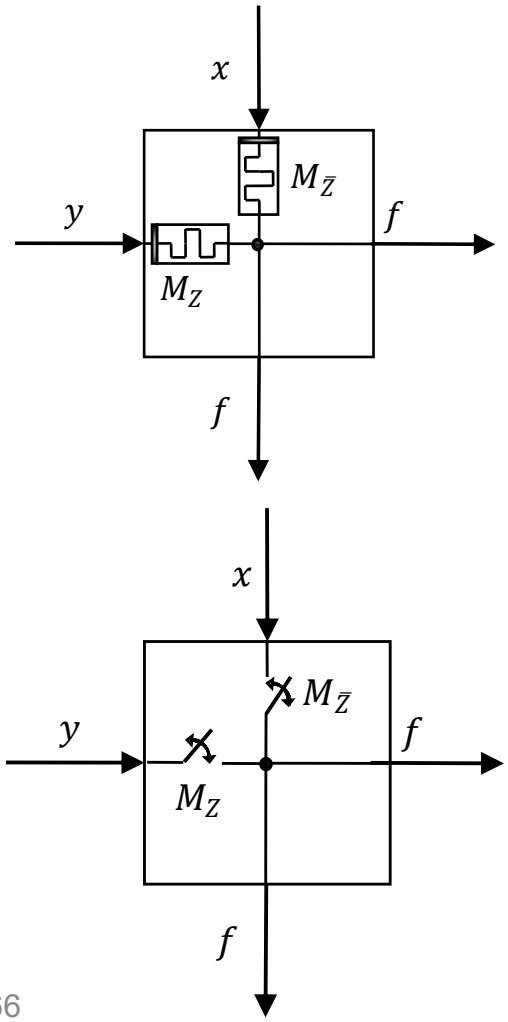
Primitive Cell Write Simulations

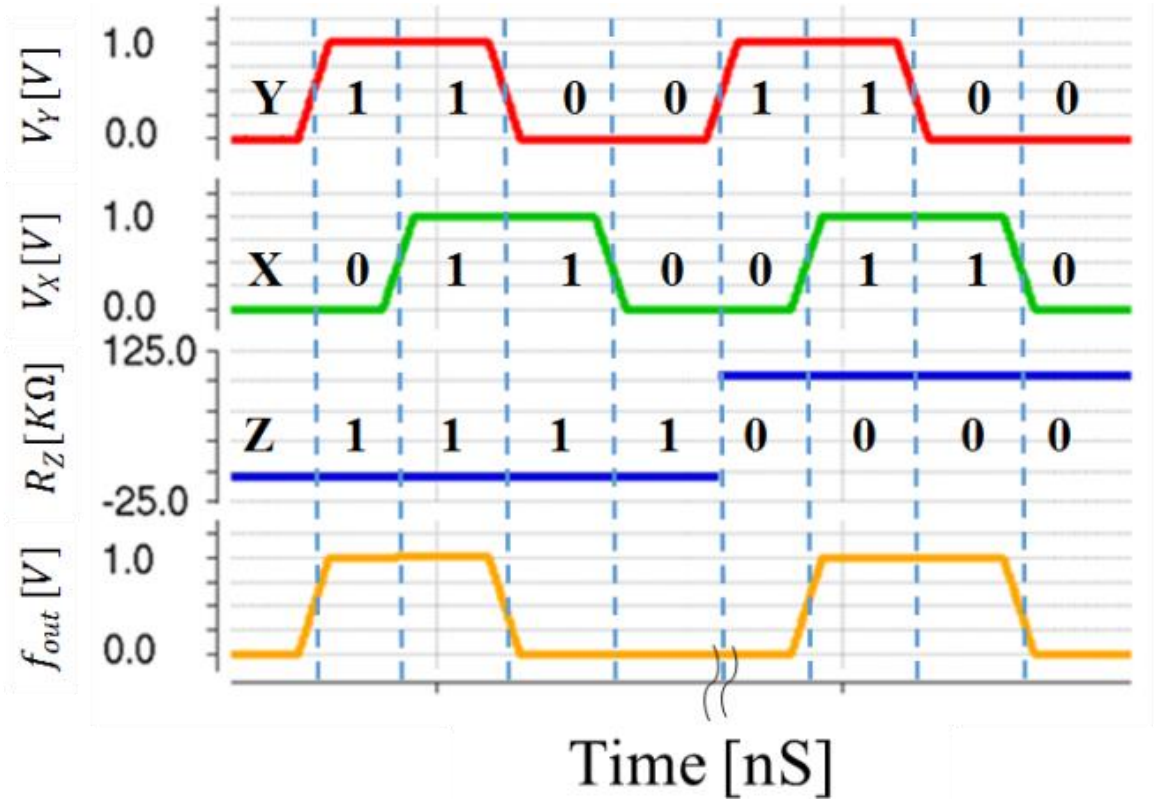




Primitive Logic Operation

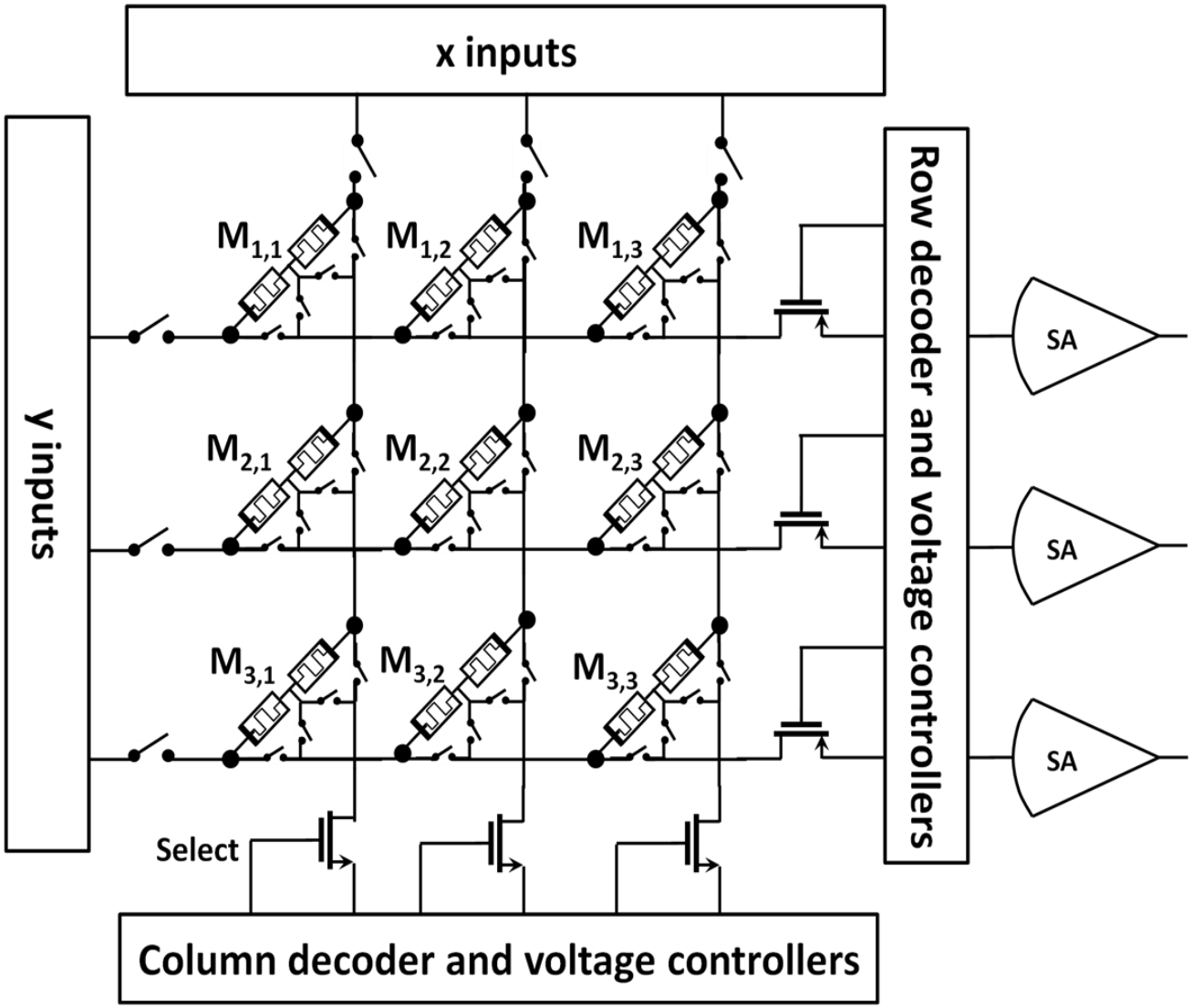
- Execute below threshold



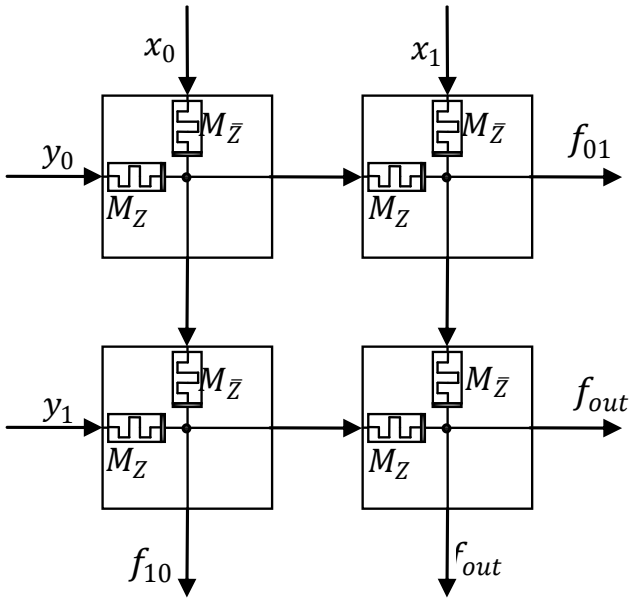


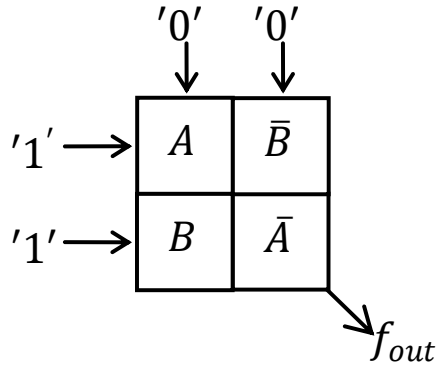
$$f_2(x, y, z) = x\bar{z} + yz$$

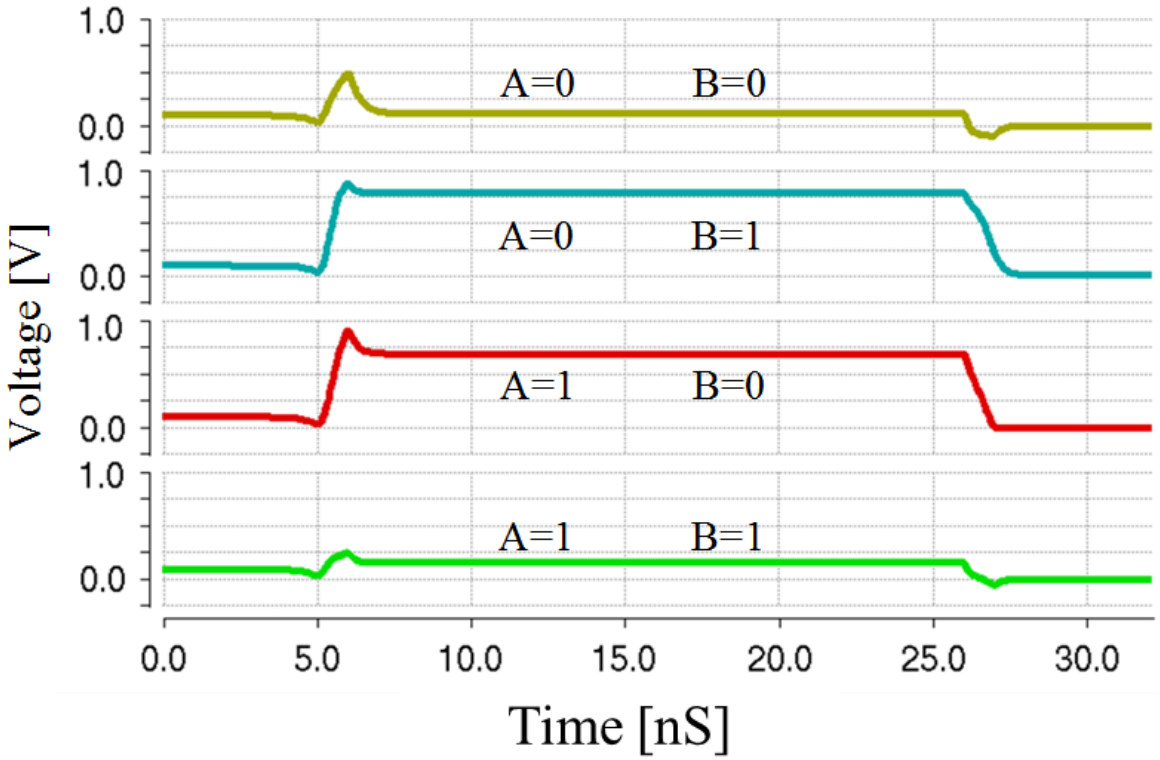
Memristive Akers within Memory



Memristive Akers 2-Input XOR



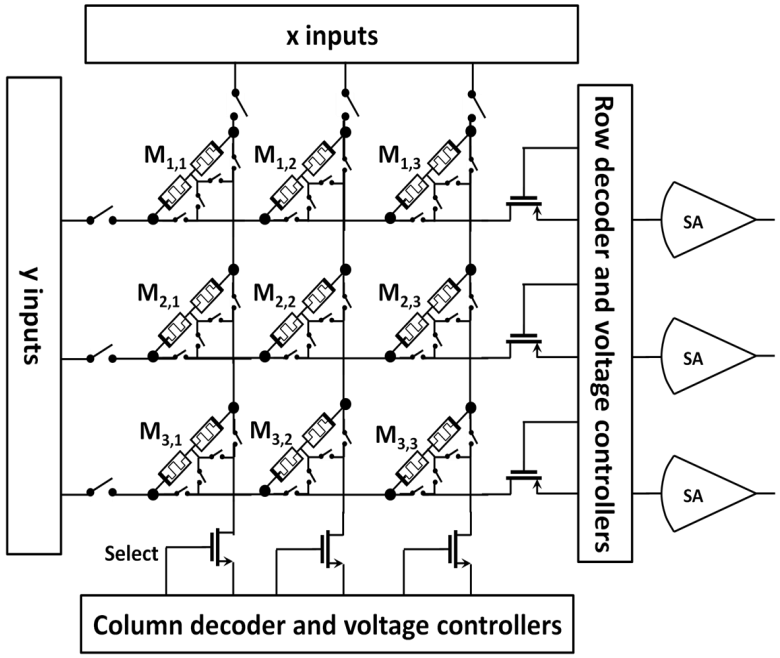




Memristive Akers Arrays

Open Issues

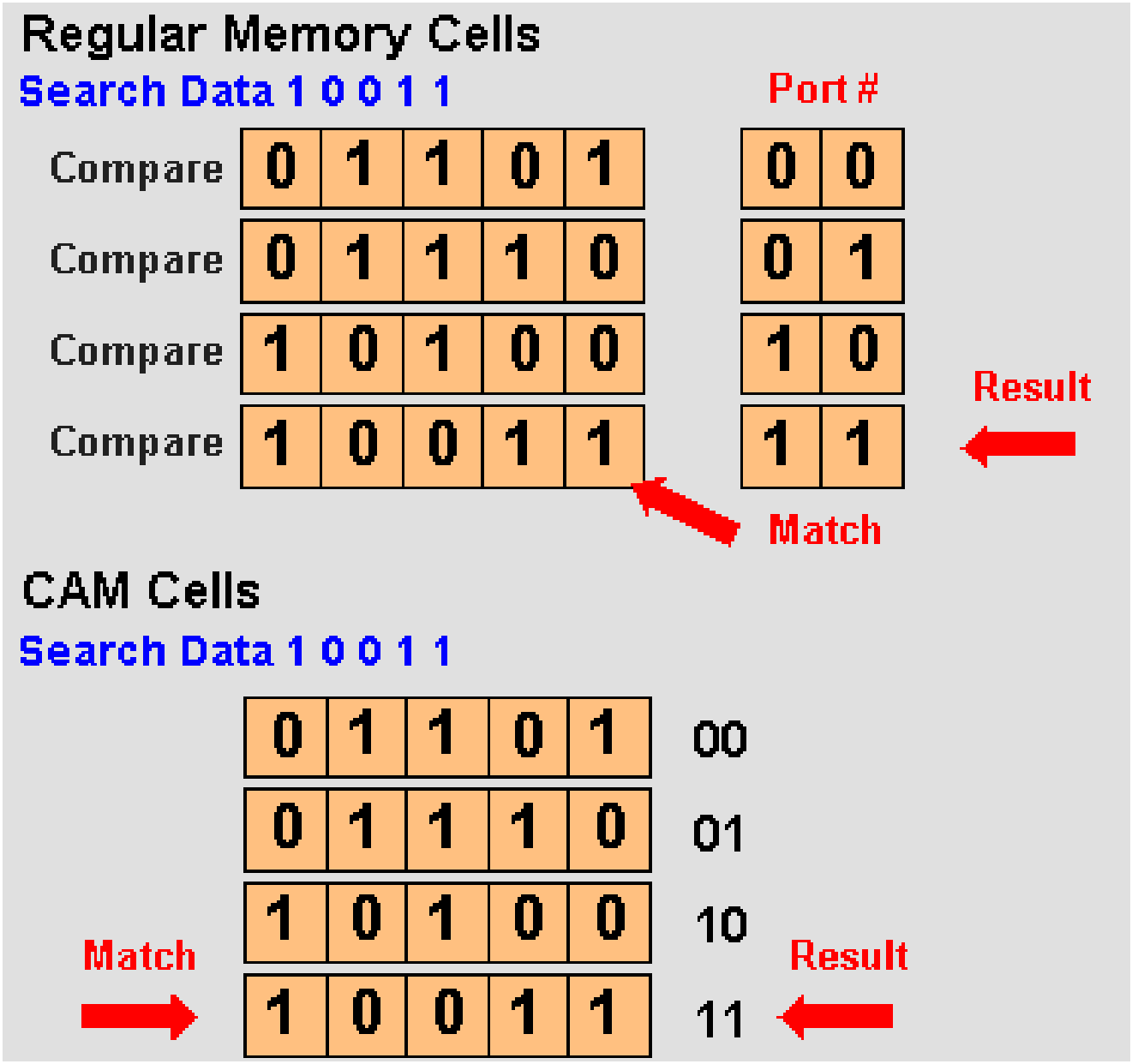
- Akers memory design (Y. Levy thesis 2015)
- Non-binary Akers arrays (extends E. Yaakobi et al. ISIT 2013)



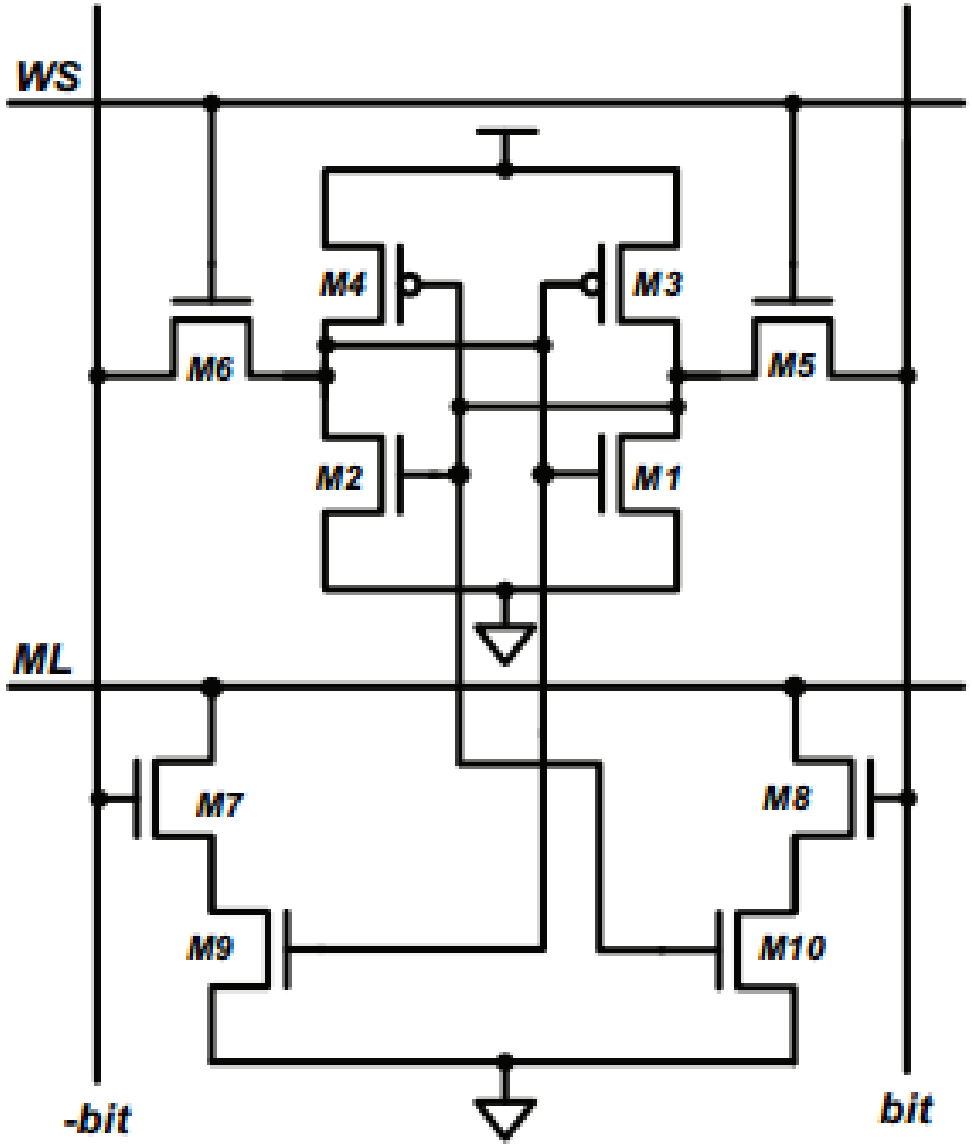
Agenda

- The need for Processing-in-Memory
- Memristors and Memristive Memory
- Memristive logic classification
- Out-of-memory logic
- **Near-memory logic**
  - Akers
  - **Associative Processors**
  - Neuromorphic
- Real in-memory logic
- Memristor for HW security

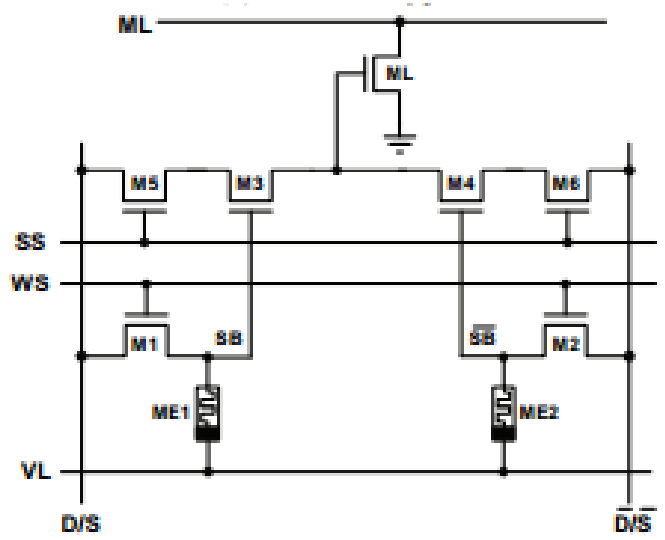
Content Addressable Memory (CAM)

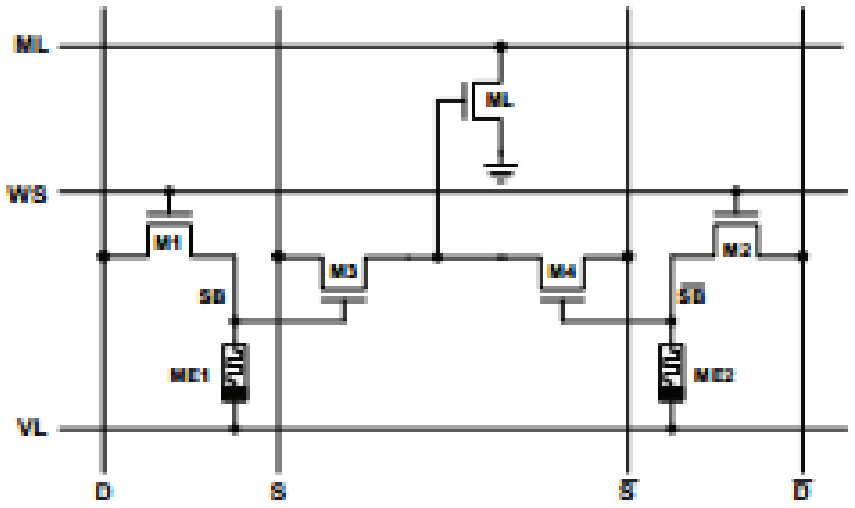


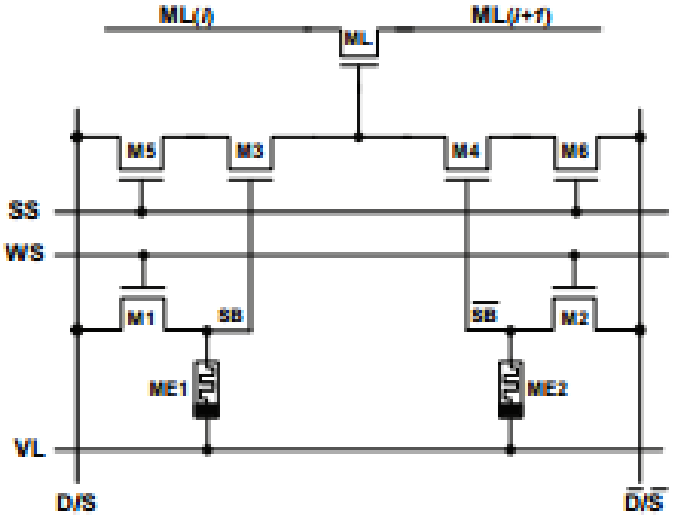
CMOS CAM

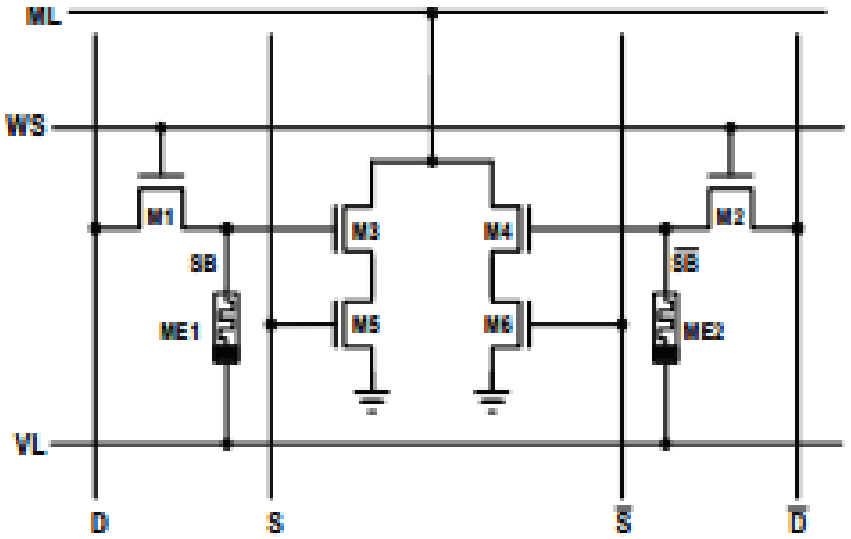


Memristive CAM









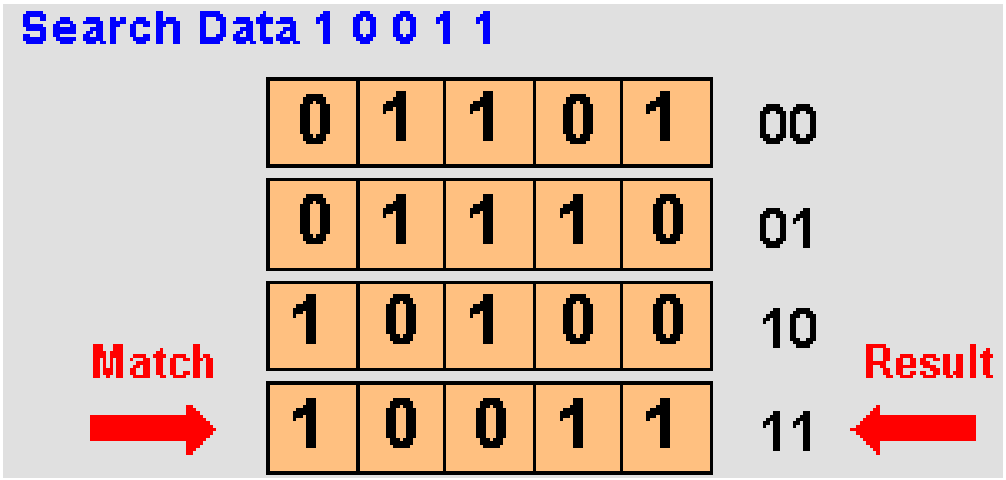
K. Eshraghian et al., TVLSI 2011

Associative In-Memory Processing

- Associative processing = computing by Look-Up-Table (LUT)
- In typical LUT, function values are stored in memory, while data is fed from outside
- In associative processor, data is stored in memory while function arguments are fed from outside
  - Computing is done *in-situ*

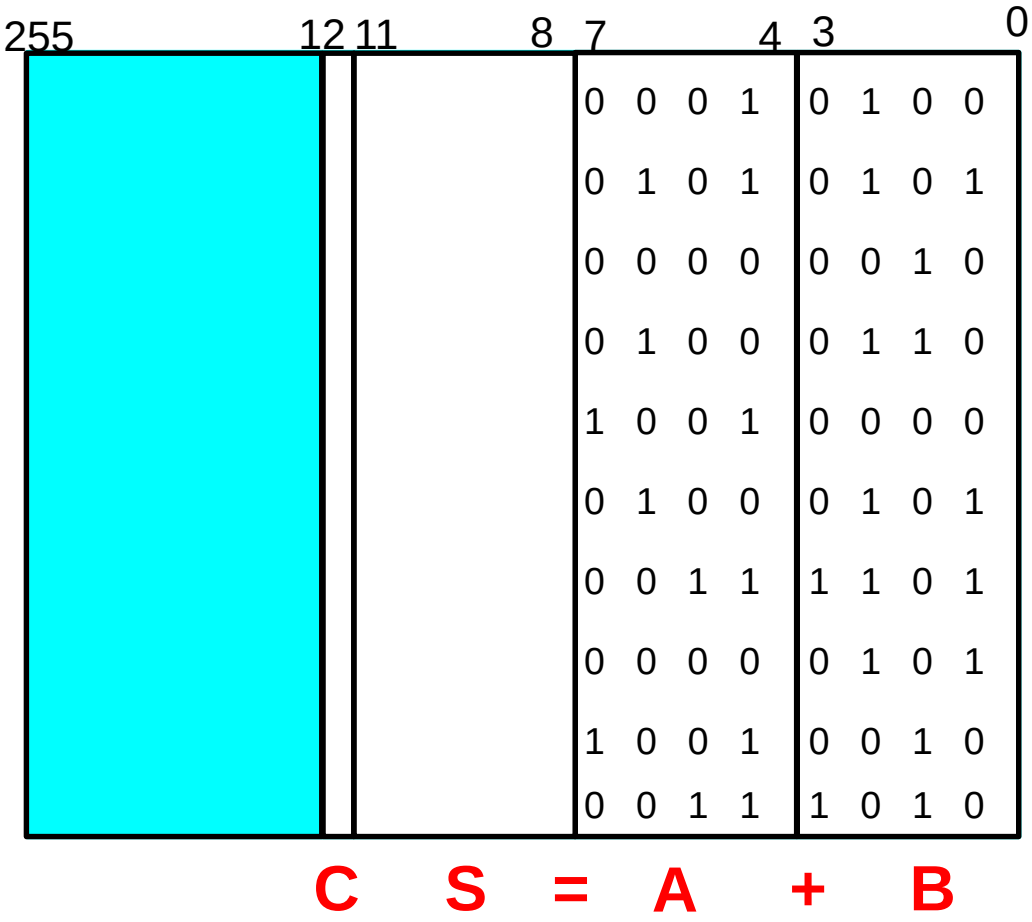
Associative Processor

- Processing-near-memory (PNM) using CAM
- AP is similar to a look-up table
- Computation is a series of “compare” and “write” operation



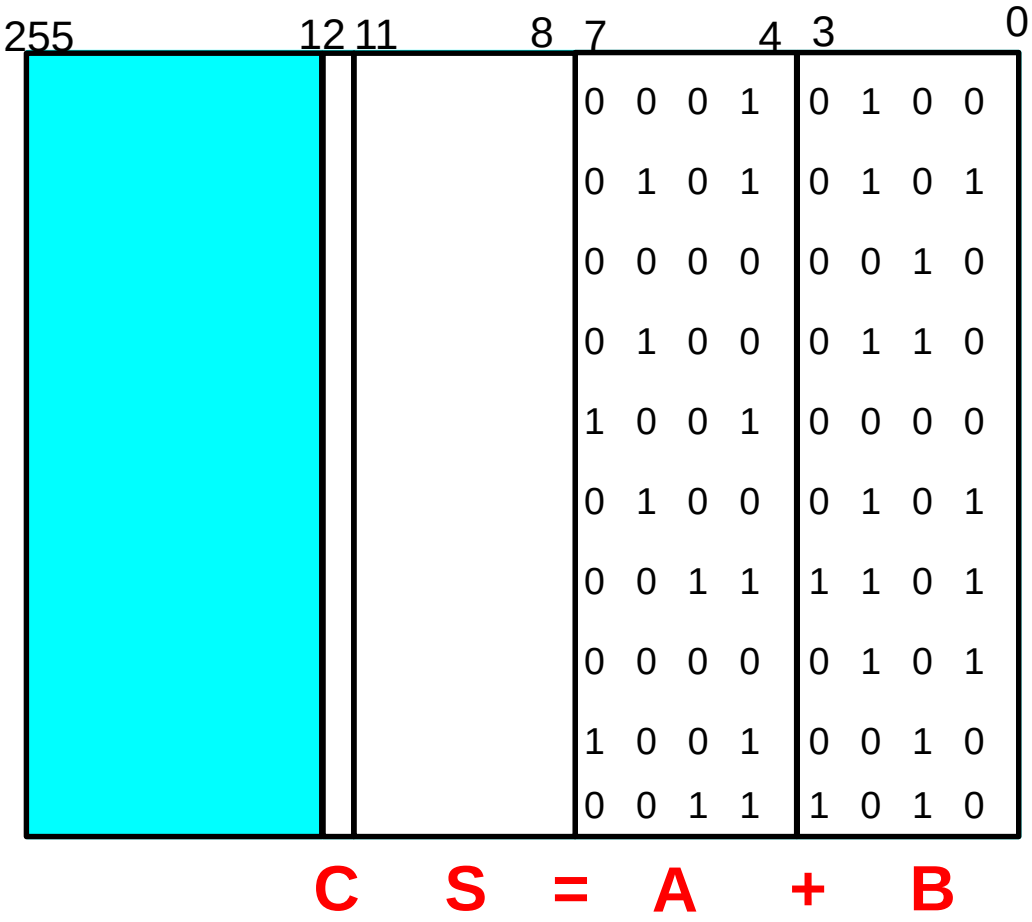
Example: Associative Vector Addition

ASSOCIATIVE PROCESSOR: MEMORY MAP



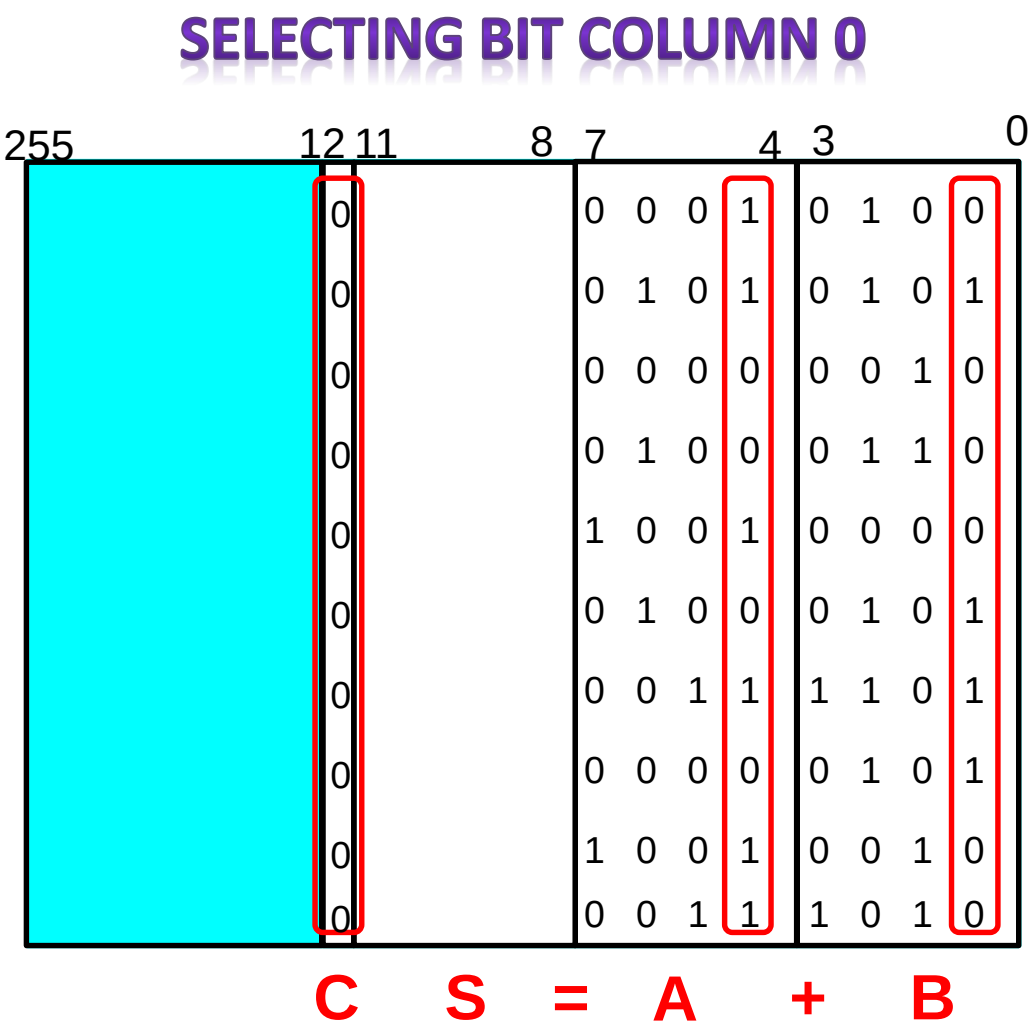
Example: Associative Vector Addition

ASSOCIATIVE PROCESSOR: MEMORY MAP



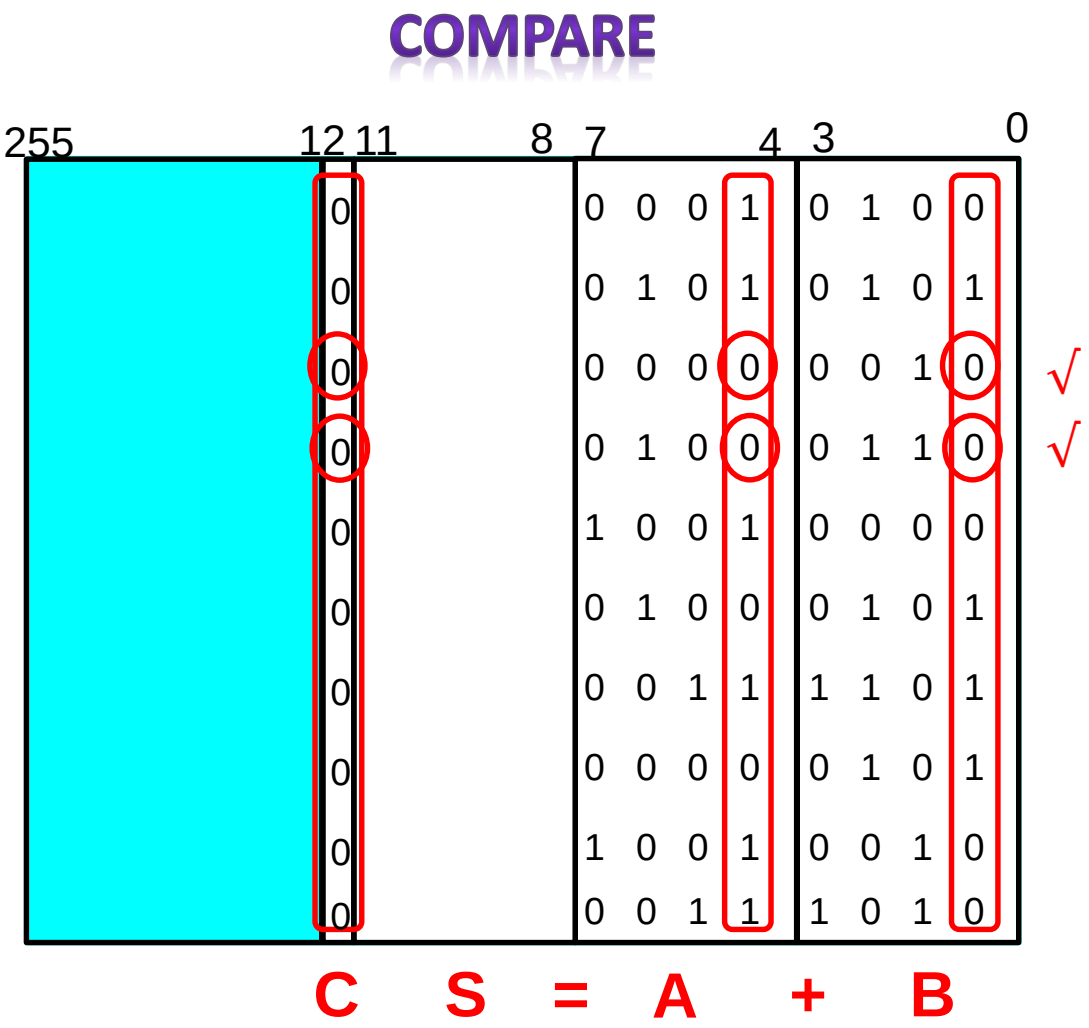
cout	s	c_{in}	a	b
0	0	0	0	0
0	1	0	0	1
0	1	0	1	0
1	0	0	1	1
0	1	1	0	0
0	1	1	0	1
1	0	1	1	0
1	1	1	1	1

Example: Associative Vector Addition



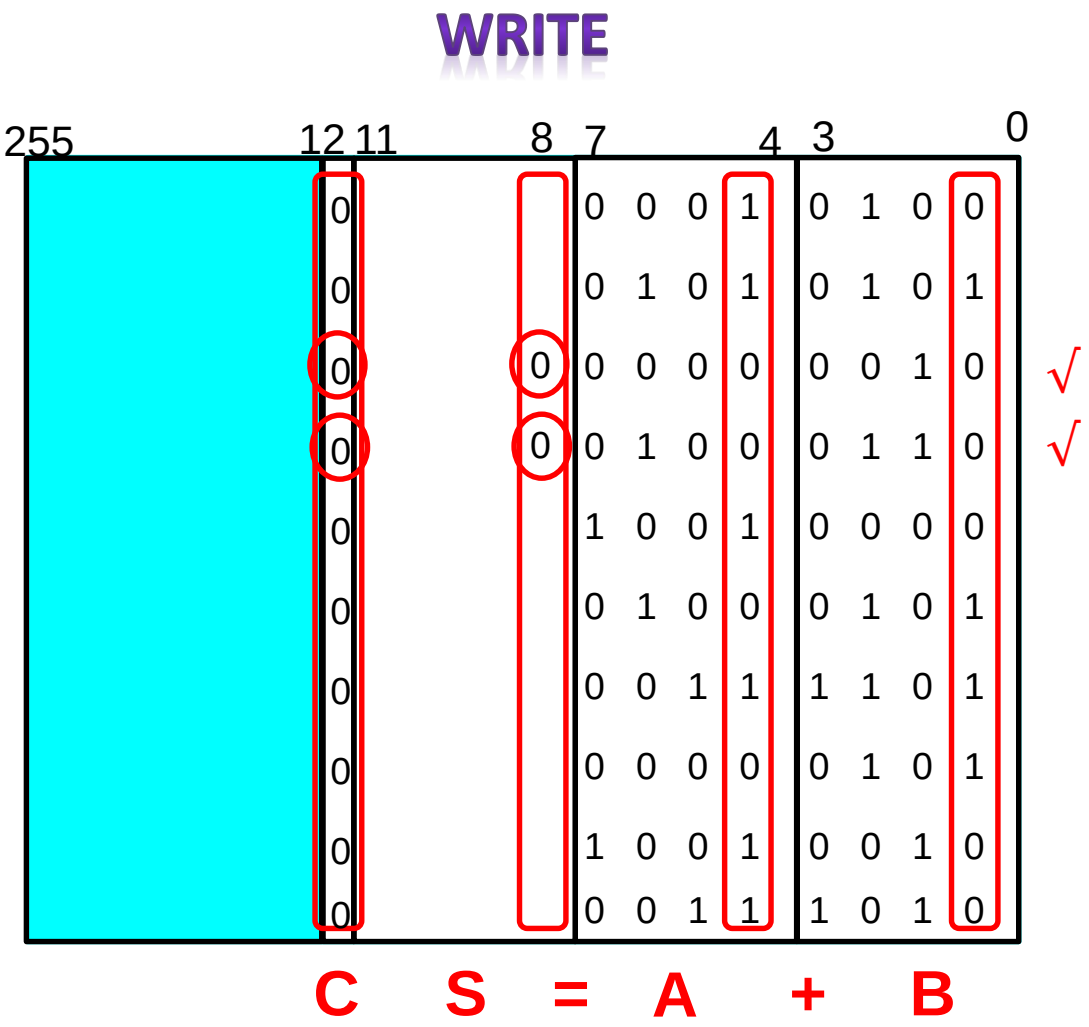
cout	s	c_{in}	a	b
0	0	0	0	0
0	1	0	0	1
0	1	0	1	0
1	0	0	1	1
0	1	1	0	0
0	1	1	0	1
1	0	1	1	0
1	1	1	1	1

Example: Associative Vector Addition



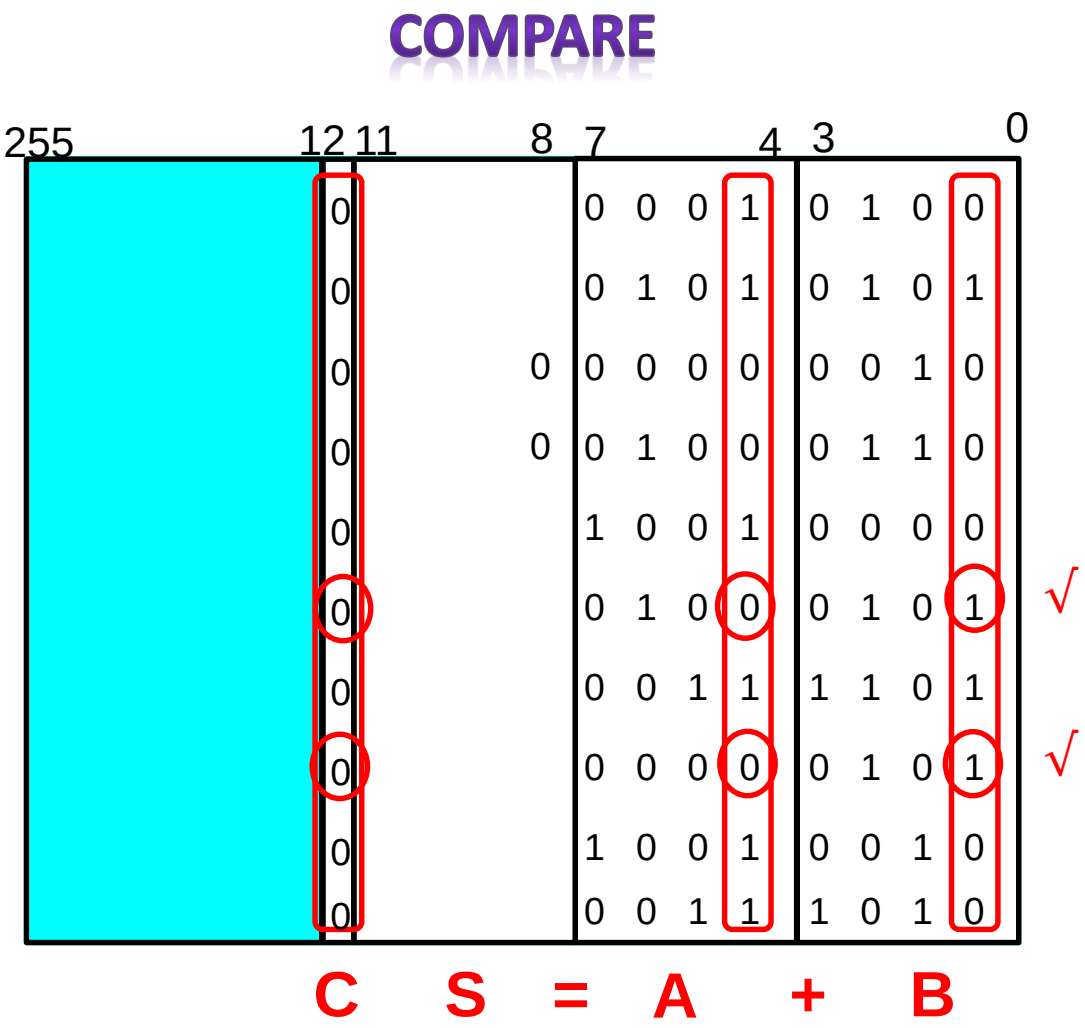
cout	s	c_{in}	a	b
0	0	0	0	0
0	1	0	0	1
0	1	0	1	0
1	0	0	1	1
0	1	1	0	0
0	1	1	0	1
1	0	1	1	0
1	1	1	1	1

Example: Associative Vector Addition



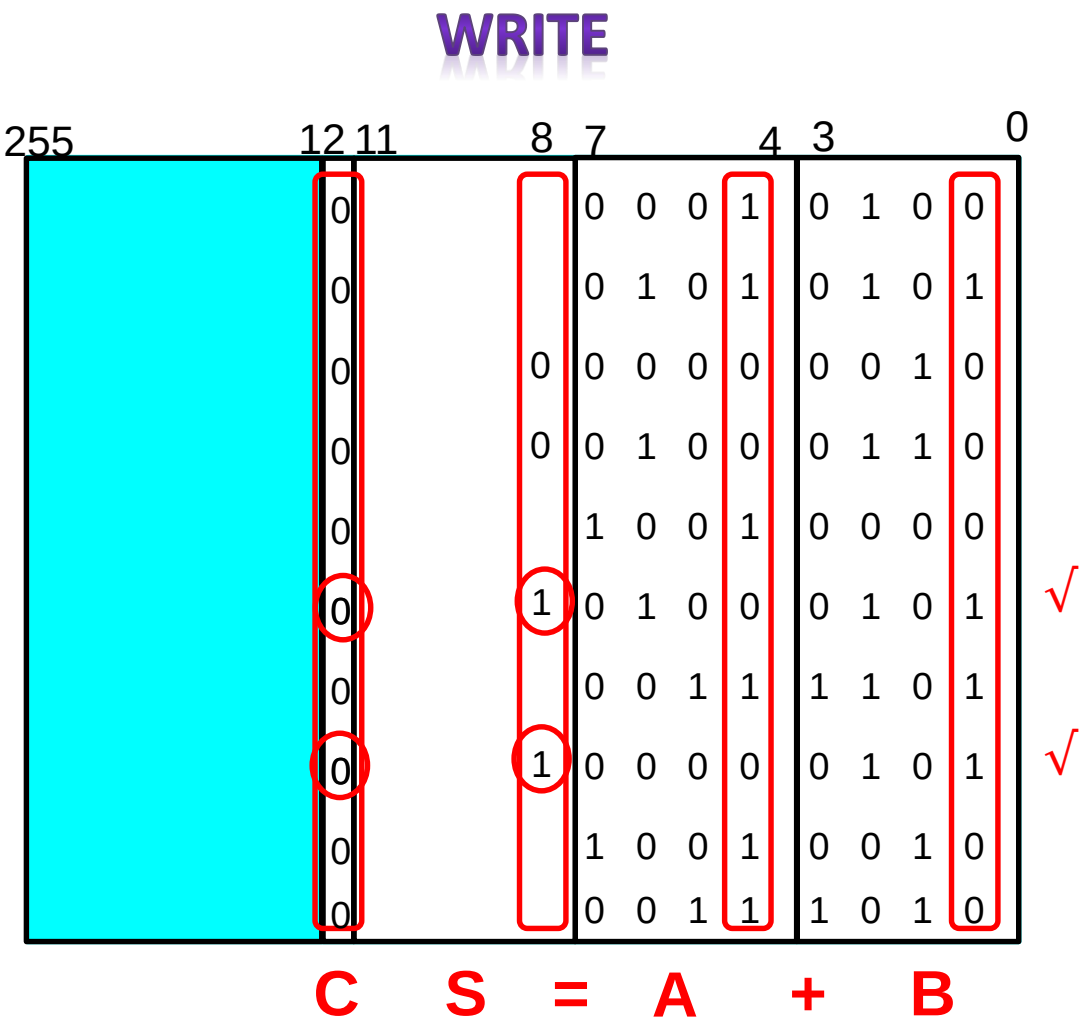
cout	s	c_{in}	a	b
0	0	0	0	0
0	1	0	0	1
0	1	0	1	0
1	0	0	1	1
0	1	1	0	0
0	1	1	0	1
1	0	1	1	0
1	1	1	1	1

Example: Associative Vector Addition



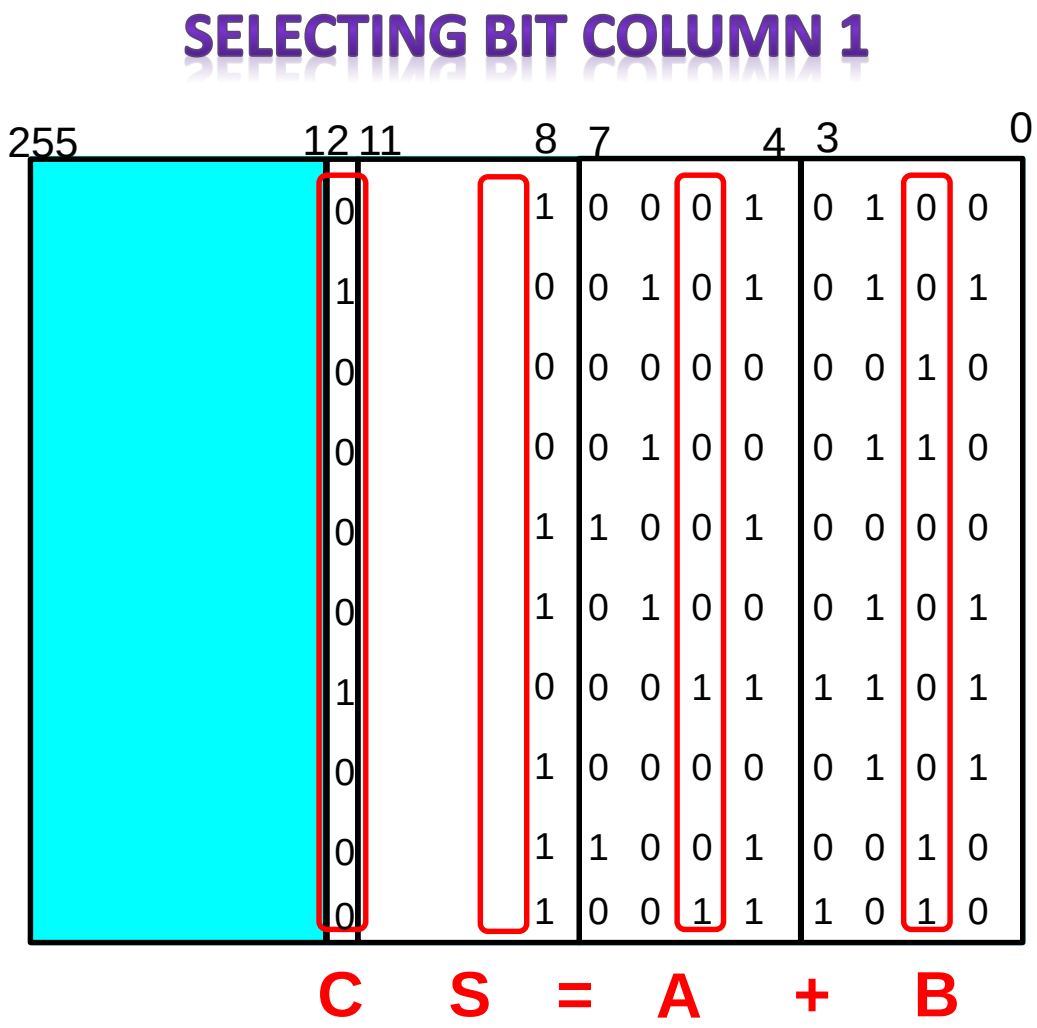
cout	s	c_{in}	a	b
0	0	0	0	0
0	1	0	0	1
0	1	0	1	0
1	0	0	1	1
0	1	1	0	0
0	1	1	0	1
1	0	1	1	0
1	1	1	1	1

Example: Associative Vector Addition



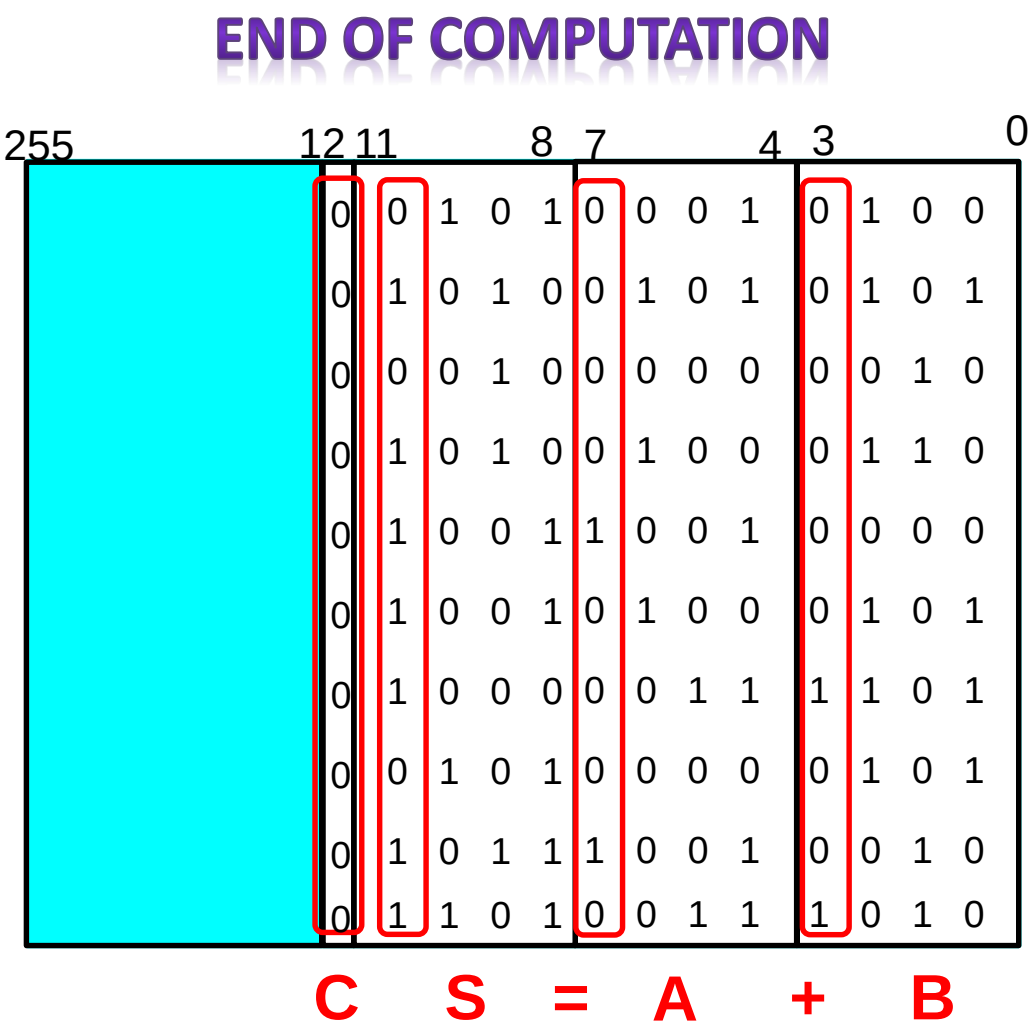
cout	s	c_{in}	a	b
0	0	0	0	0
0	1	0	0	1
0	1	0	1	0
1	0	0	1	1
0	1	1	0	0
0	1	1	0	1
1	0	1	1	0
1	1	1	1	1

Example: Associative Vector Addition



cout	s	c_{in}	a	b
0	0	0	0	0
0	1	0	0	1
0	1	0	1	0
1	0	0	1	1
0	1	1	0	0
0	1	1	0	1
1	0	1	1	0
1	1	1	1	1

Example: Associative Vector Addition

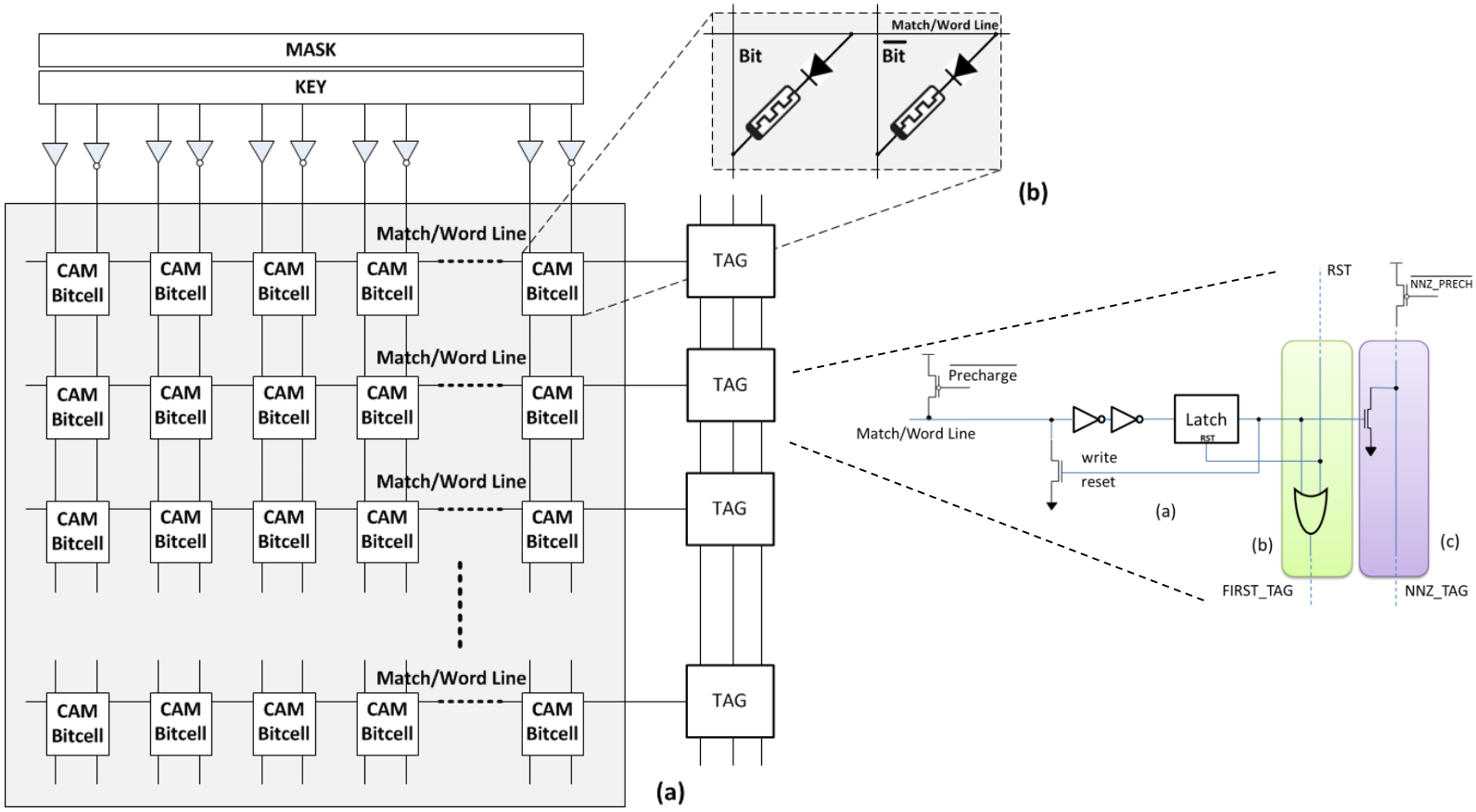


cout	s	c_{in}	a	b
0	0	0	0	0
0	1	0	0	1
0	1	0	1	0
1	0	0	1	1
0	1	1	0	0
0	1	1	0	1
1	0	1	1	0
1	1	1	1	1

AP Complexity

- Arithmetic:
  - Fixed point
    - m bit add / sub: $O(m)$ cycles
    - m bit mult/div: $O(m^2)$ cycles
- Pattern match: $O(1)$ cycles
- Finding max/min: $O(1)$ cycles
- Independent of the dataset size:
  The larger the problem, the better the performance of the Associative Processor!

Resistive Associative Processor



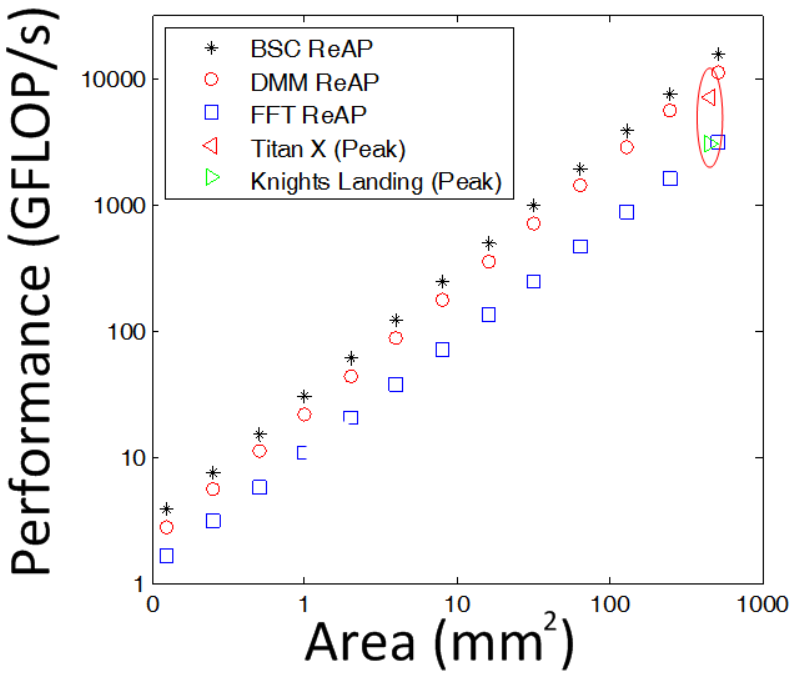
- Converting a memory crossbar into a massively parallel SIMD processor

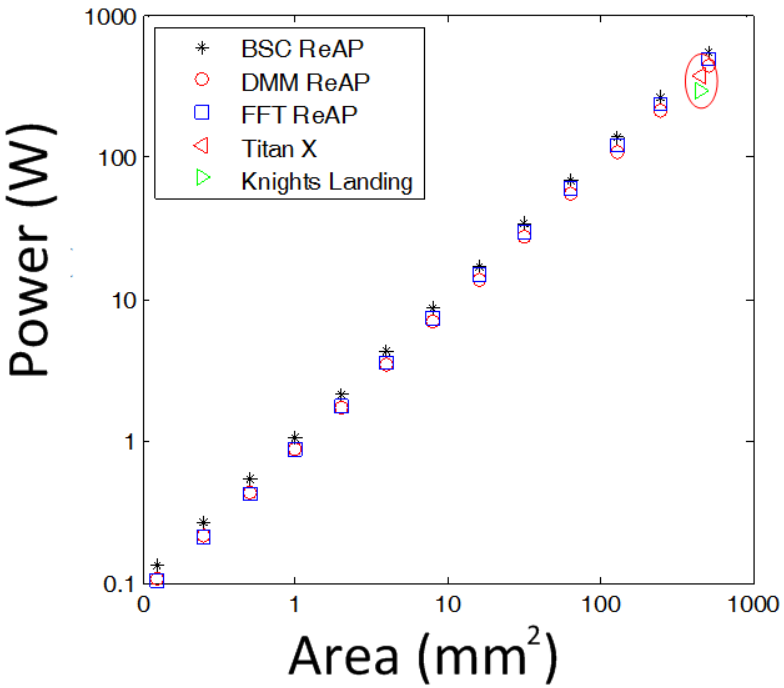
L. Yavits et al., "Resistive Associative Processor," Computer Architecture Letters, 2015

What AP is Good for

- Dense and sparse linear algebra
- K-means clustering
- Linear SVM classification
- FFT, convolution, feature extraction
- Sequence alignment (Smith-Waterman)
- Graph processing (Dijkstra's shortest path finding)

Performance and Power Consumption





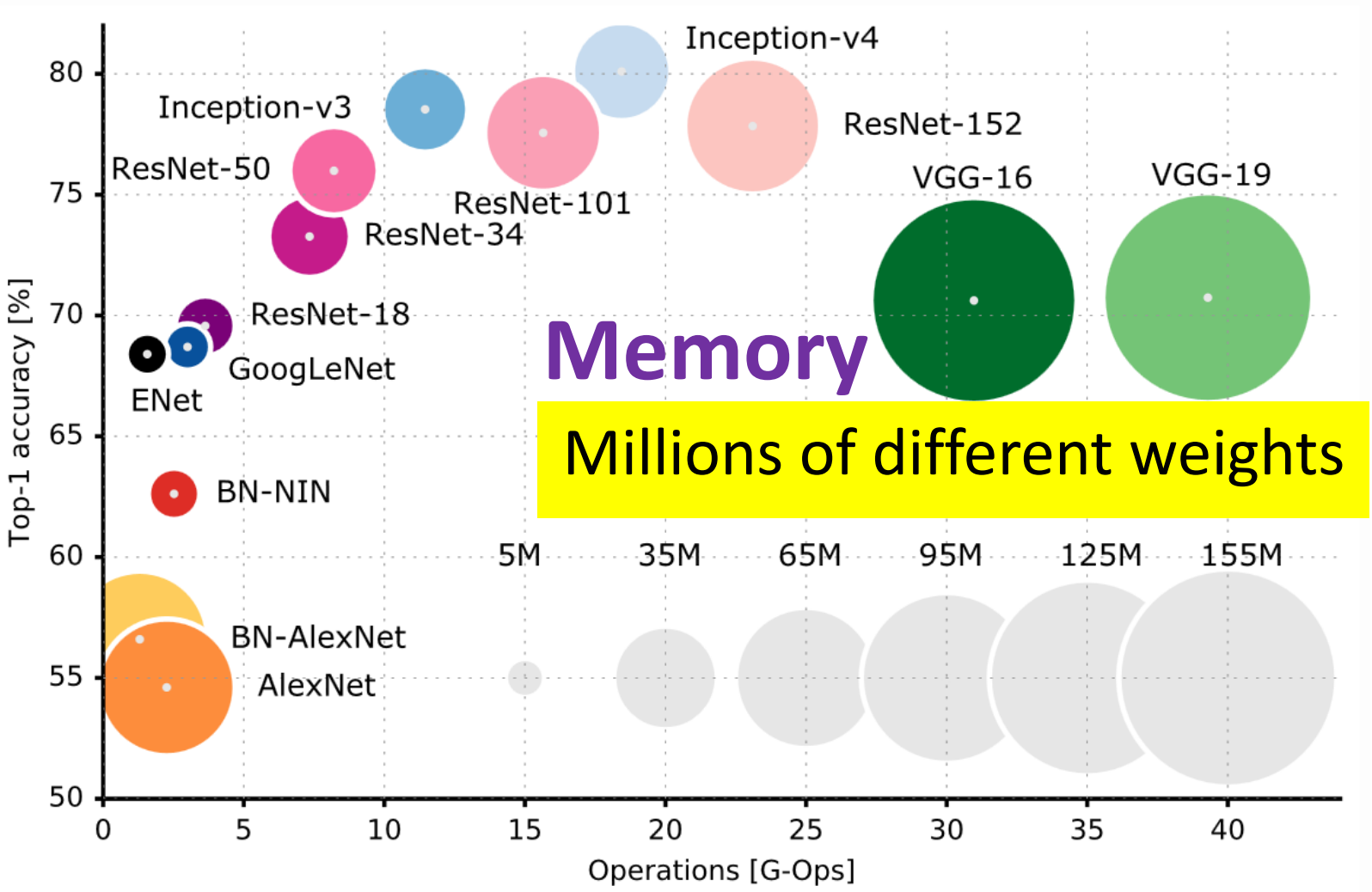
- ReAP size (and consequently performance) are constrained by memristor write energy
- Max Dense Matrix Multiplication performance is 5TFLOPS under this constraint

L. Yavits et al., "Resistive Associative Processor," Computer Architecture Letters, 2015

Agenda

- The need for Processing-in-Memory
- Memristors and Memristive Memory
- Memristive logic classification
- Out-of-memory logic
- **Near-memory logic**
  - Akers
  - Associative Processors
  - **Neuromorphic**
- Real in-memory logic

AI Requires a Lot of Data



Memory

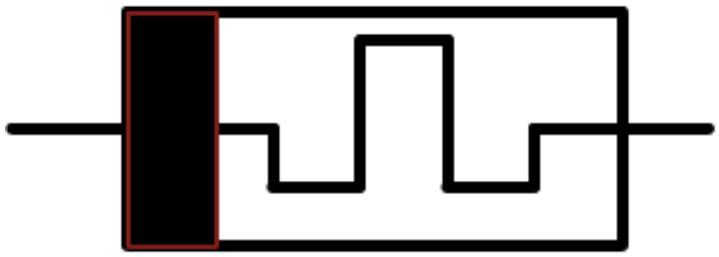
Millions of different weights

Billions of calculations

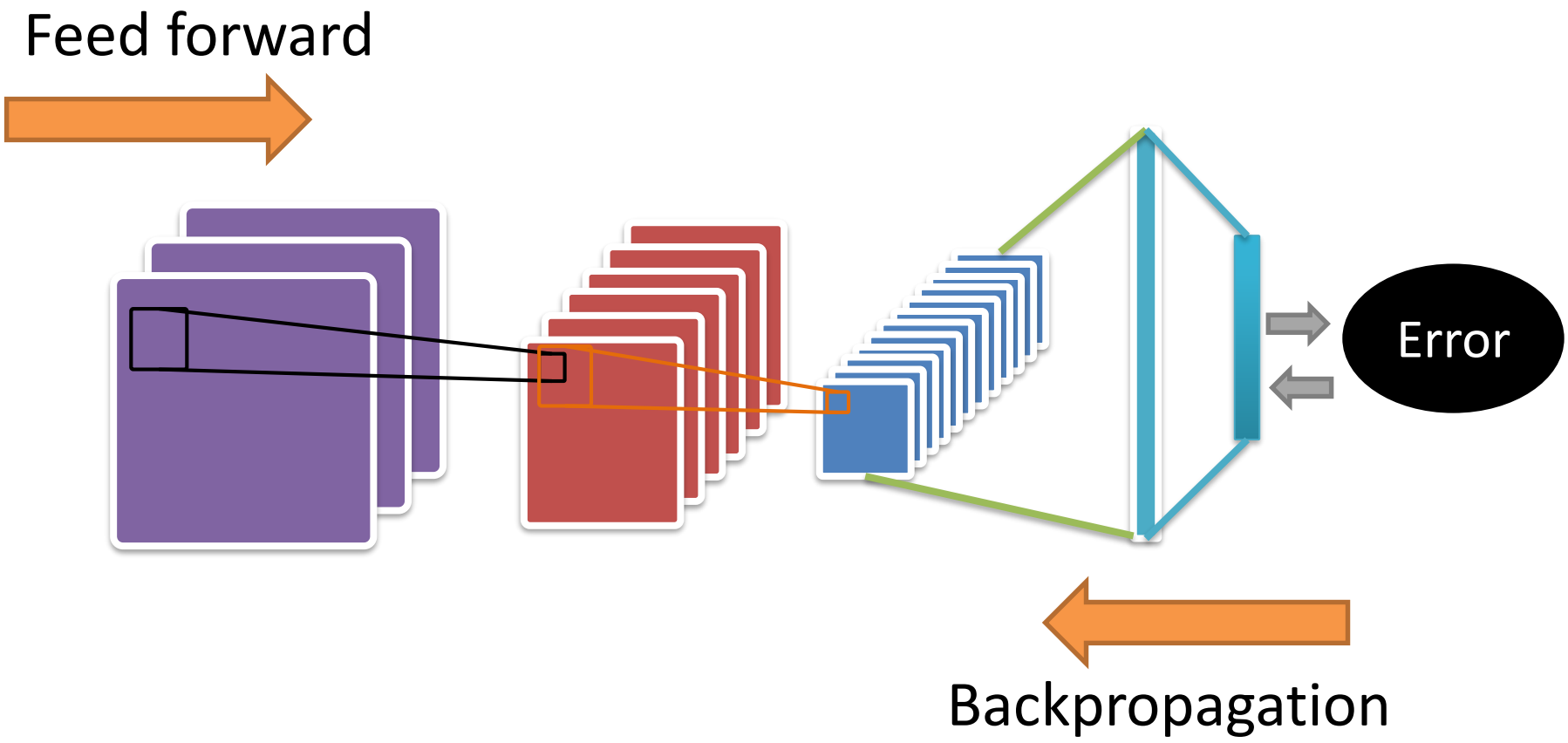
CPU, GPU

Memristor-Based ASIC for DNN

- Enable near-memory processing
  - Reduce memory traffic
- Analog MAC operation
  - Reduce energy consumption
  - Parallel computation



DNN Inference and Training



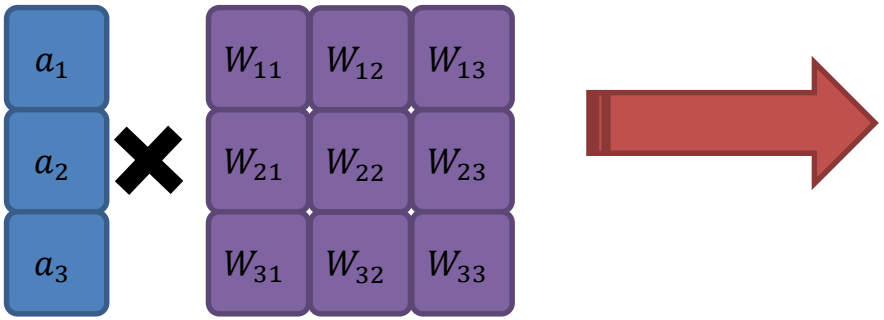
Memristive Processing for DNN

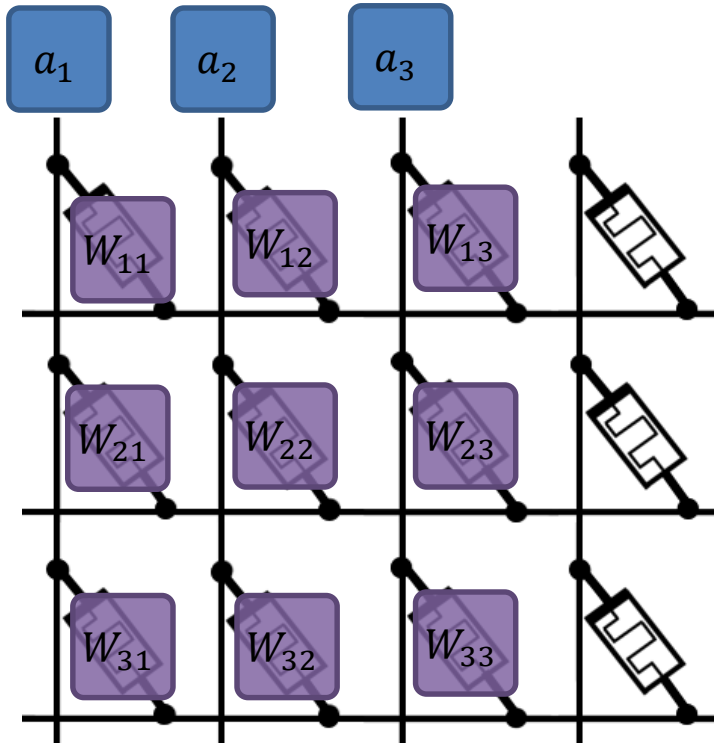
- The crossbar is used to store the weights data

  The conductance = the weight value

$$G_{ij} = W_{ij}$$

- The activation is the voltage drop over the columns (rows)





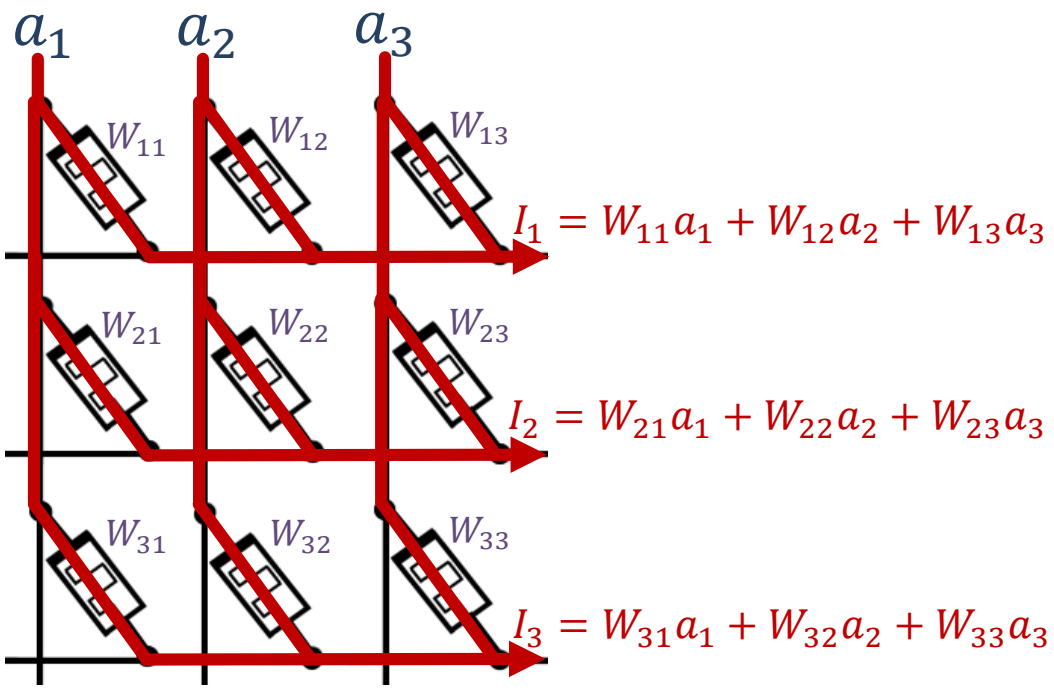
Multiply and Accumulate (Feed Forward)

- Using KCL

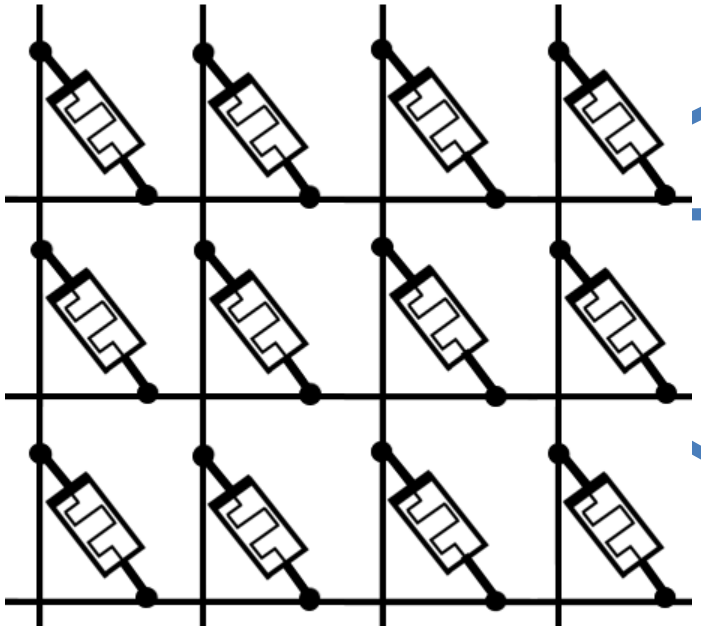
  The output current is the weighted sum (inner-product)

$$I_j = \sum_{i=1}^{N} W_{ij} a_i$$

- Parallel operation to all rows is the crossbar



Data Representation



Input $\vec{V}$- discrete
Weights W- discrete

Input $\vec{V}$- discrete
Weights W- analog

Input $\vec{V}$- analog
Weights W- analog

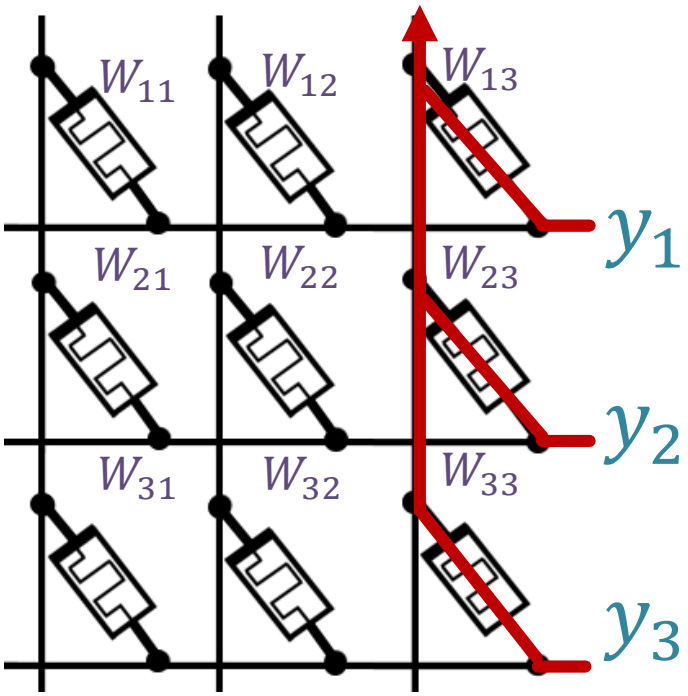
Input $\vec{V}$- analog
Weights W- discrete

Back Propagation

The error δ_l is propagate according to:

$$y_l \triangleq \underbrace{\left(W_{l+1}^T y_{l+1}\right)}_{\delta_l} \times \sigma'(z_l)$$

$$\delta_3 = W_{13}\, y_1 + W_{23} y_2 + W_{33} y_3$$



D. Soudry, D. Di Castro, A. Gal, A. Kolodny, and S. Kvatinsky, "Memristor-based Multilayer Neural Networks with Online Gradient Descent Training," IEEE TNNLS 2015

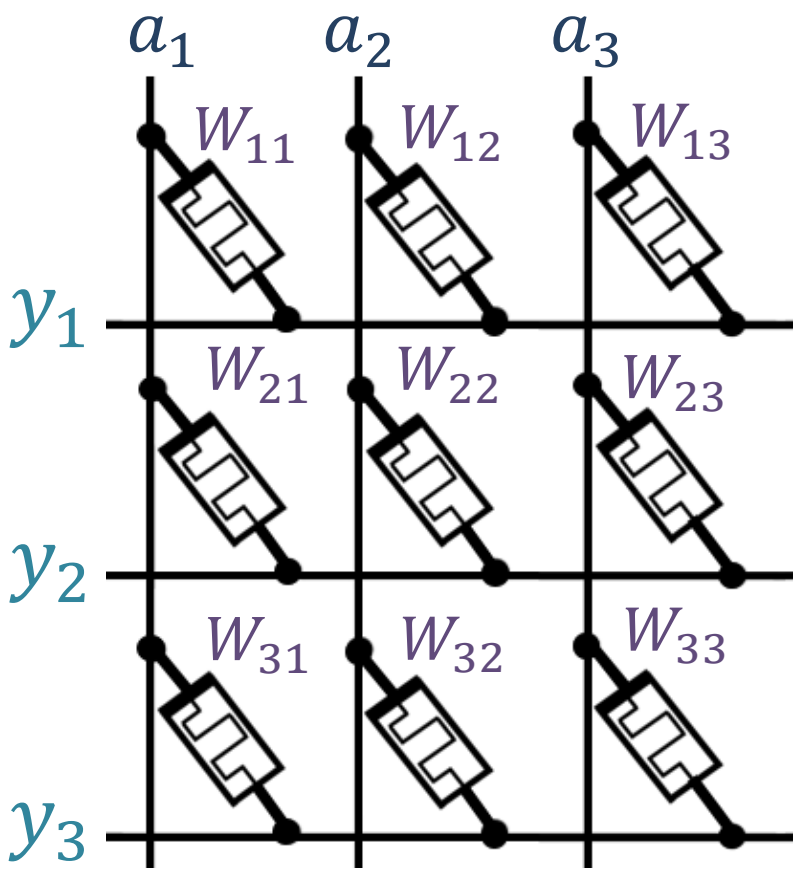
Stochastic Gradient Descent Training

The gradient value ΔW for fully connected layers:

$$\frac{\partial C}{\partial W} = ya^T$$

For Stochastic gradient descent (SGD):

$$W = W - \eta \frac{\partial C}{\partial W} = W - \eta ya^T$$



D. Soudry, D. Di Castro, A. Gal, A. Kolodny, and S. Kvatinsky, "Memristor-based Multilayer Neural Networks with Online Gradient Descent Training," IEEE TNNLS 2015

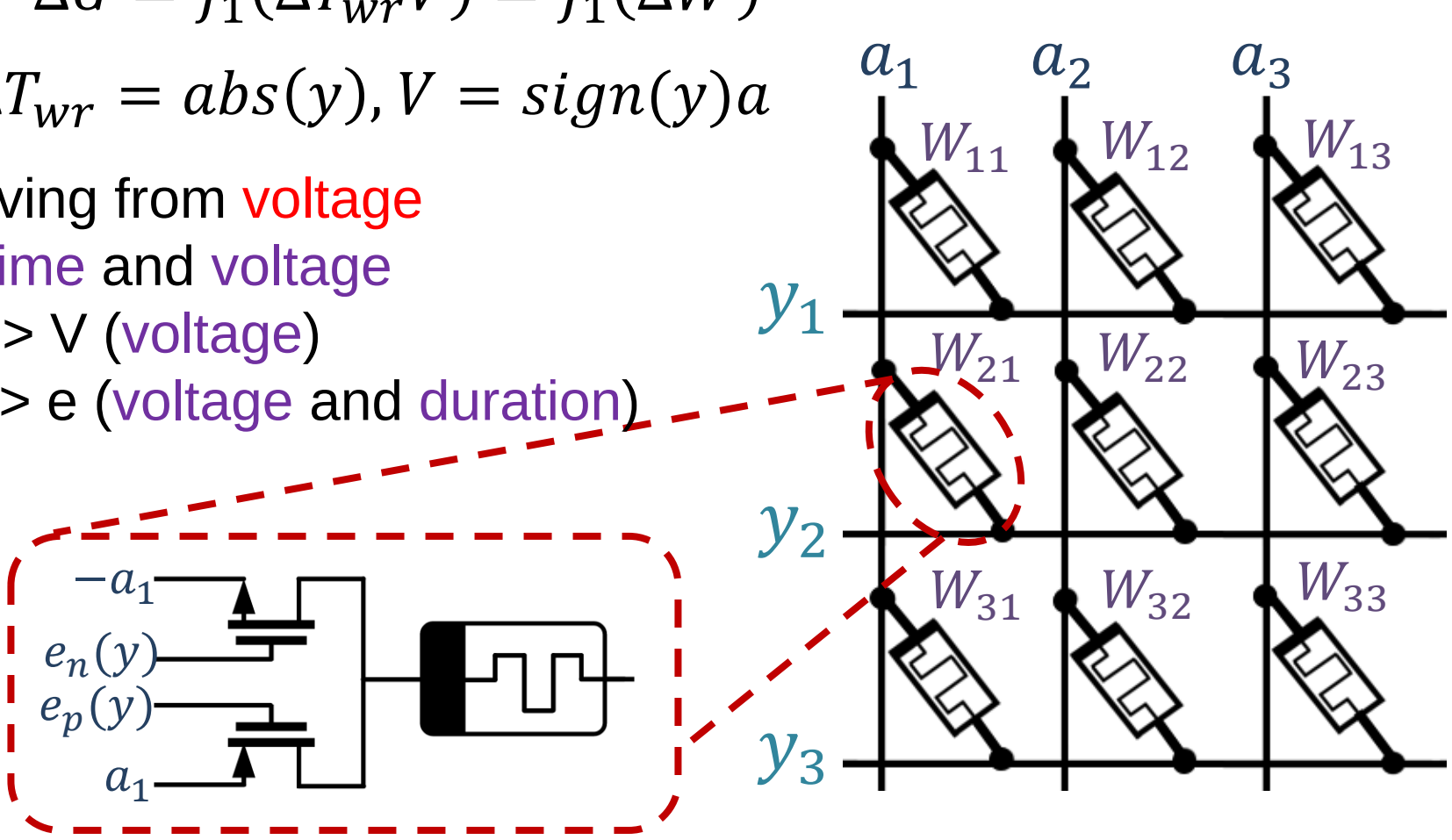
SGD Memristive Weight Update

Conductivity:

$$\Delta G = f_1(\Delta T_{wr} V) = f_1(\Delta W)$$

$$\Delta T_{wr} = abs(y), V = sign(y)a$$

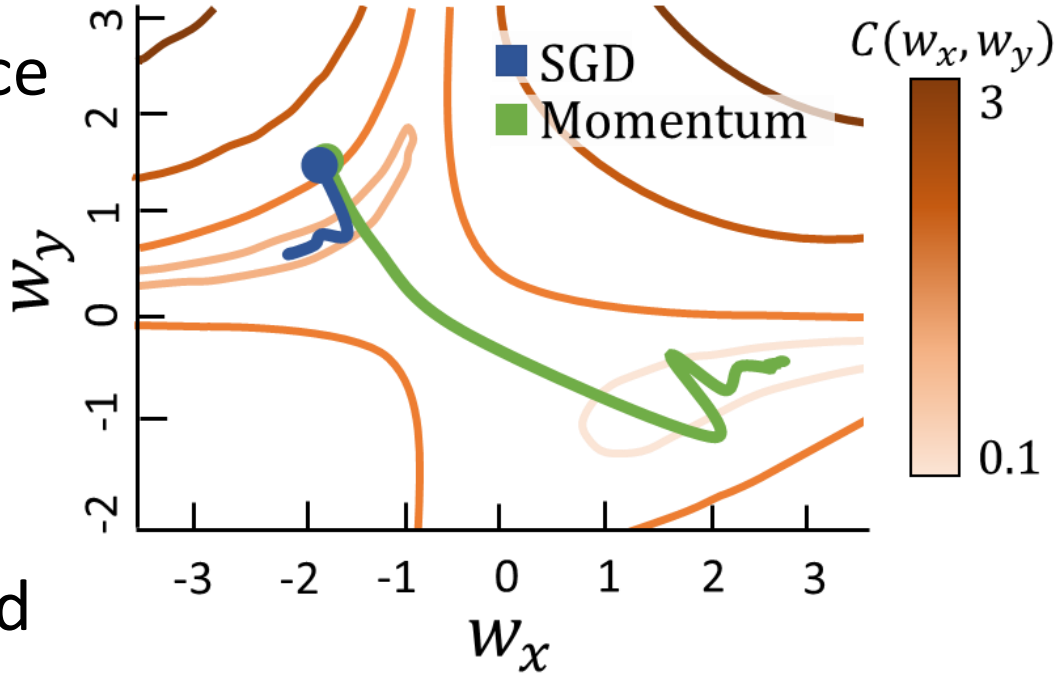
Moving from voltage
to time and voltage
a --> V (voltage)
y --> e (voltage and duration)



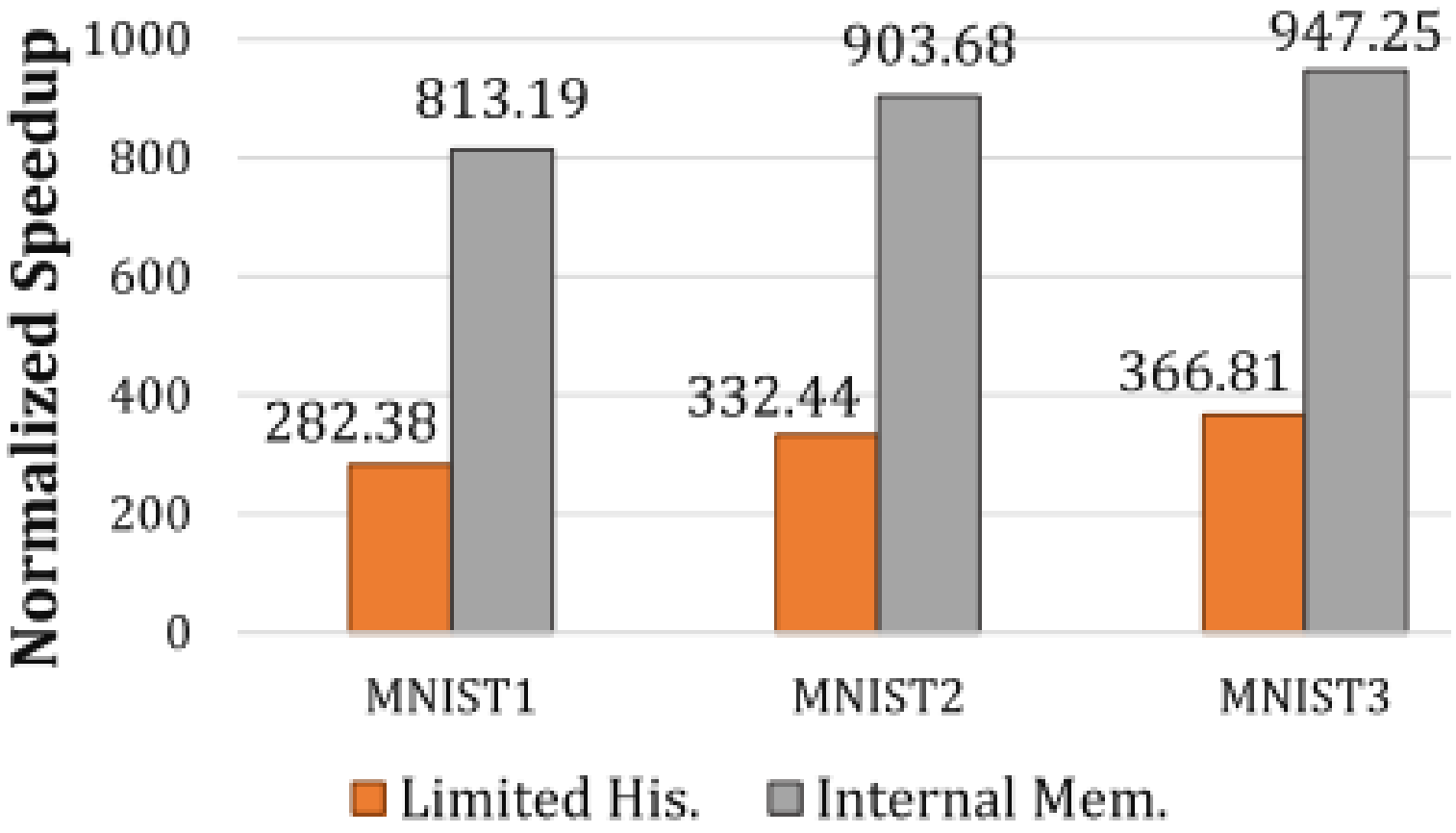
D. Soudry, D. Di Castro, A. Gal, A. Kolodny, and S. Kvatinsky, "Memristor-based Multilayer Neural Networks with Online Gradient Descent Training," IEEE TNNLS 2015

Going Beyond SGD (Optimization Algorithms)

- Helps SGD performance
  - Increase accuracy
  - Accelerate convergence
- Momentum
  - Keeps the update history
  - Memory overhead
  - Computation overhead



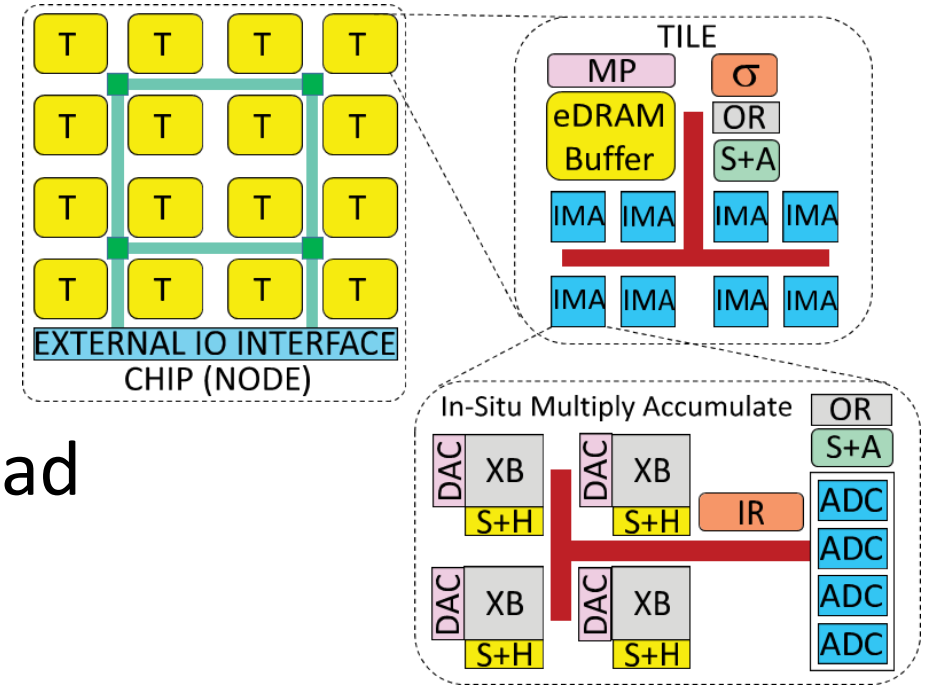
Runtime Speedup vs. GPU



T. Greenberg-Toledo, R. Mazor, A. Haj-Ali and S. Kvatinsky, "Supporting the Momentum Training Algorithm Using a Memristor-Based Synapse," IEEE TCAS-I 2019

Challenges

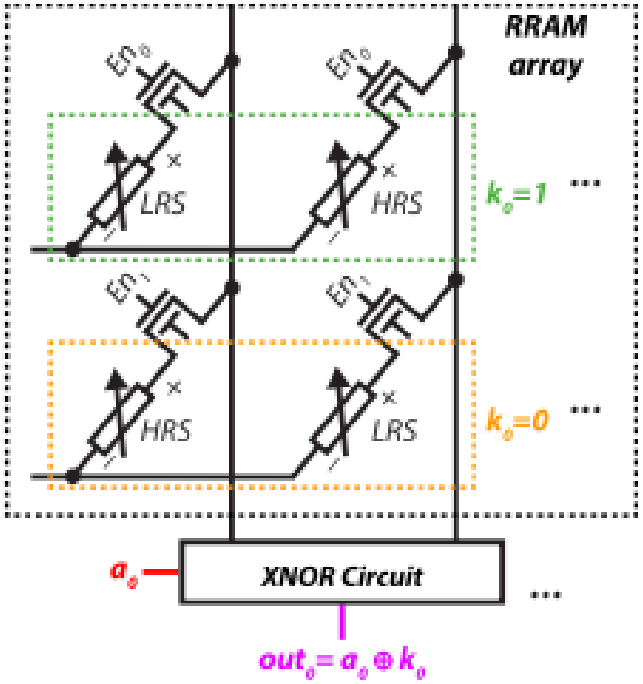
- Architecture
  - Programmability
  - Limited crossbar size
  - Pipeline utilization
  - Data converters overhead
- Technology
  - Precision
  - Reliability
  - Endurance (training)



Shafiee et al., "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ISCA 2016*

Ongoing/Future Work

- Quantized neural networks
  - BNN/TNN for inference
  - BNN/TNN for training
- Programmable accelerators
- Intelligent mixed-signal circuits



E. Giacomin et al., "A Robust Digital RRAM-Based Convolutional Block for Low-Power Image Processing and Learning Applications," IEEE TCAS-I 2019

L. Danial, N. Wainstein, S. Kraus, and S. Kvatinsky, "Breaking Through the Speed-Power-Accuracy Tradeoff in ADCs using a Memristive Neuromorphic Architecture", IEEE TETCI 2018

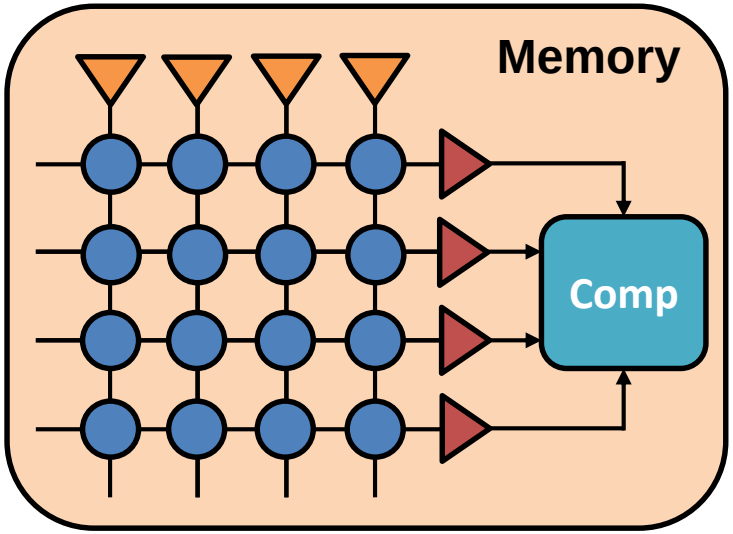
L. Danial, N. Wainstein, S. Kraus, and S. Kvatinsky, "DIDACTIC: A Data-Intelligent Digital-to-Analog Converter with a Trainable Integrated Circuit using Memristors", IEEE JETCAS 2018

Agenda

- The need for Processing-in-Memory
- Memristors and Memristive Memory
- Memristive logic classification
- Out-of-memory logic
- Near-memory logic
- **Real in-memory logic**
  - **Stateful logic – IMPLY, MAGIC**
  - Memristive Memory Processing Unit (mMPU)
- Memristor for HW security

Proximity of Computation

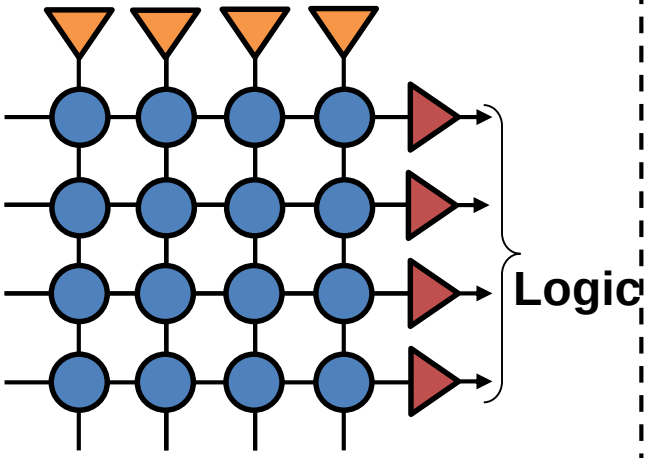
Computation using logic blocks



Dlugosch *et al.*, *IEEE TPDS*, 2014
Oskin *et al.*, *Comput. Archit. News*, 1998
Elliott *et al.*, *IEEE Des. Test*, 1999
Gokhale *et al.*, *Computer*, 1995

OUT-

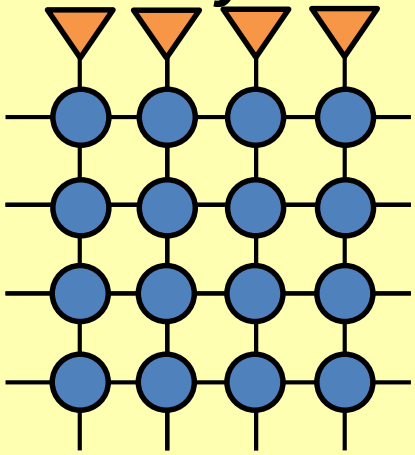
Computation using peripheral circuit



Li *et al.*, *DAC*, 2016
Seshadri *et al.*, MICRO 2017
Aga *et.al.* HPCA 2017
Eckert *et.al.* ISCA 2018

NEAR-

Computation ***using memory cells***



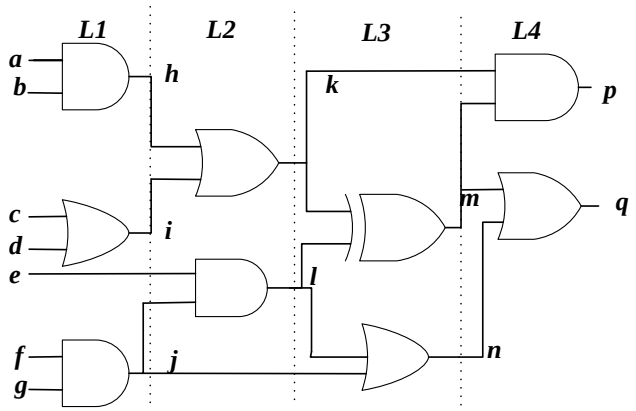
Kvatinsky *et al.*, *TCAS-II*, 2014
Talati *et al.*, TNANO 2016

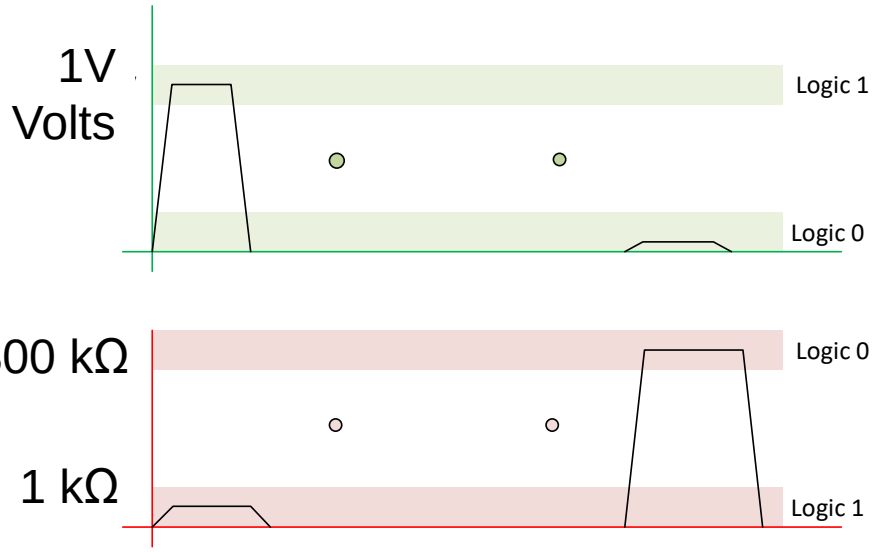
IN-

J. Reuben, R. Ben Hur, N. Wald, N. Talati, A. Haj Ali, P.-E. Gaillardon, and S. Kvatinsky, "Memristive Logic: A Framework for Evaluation and Comparison", PATMOS 2017

State Variable

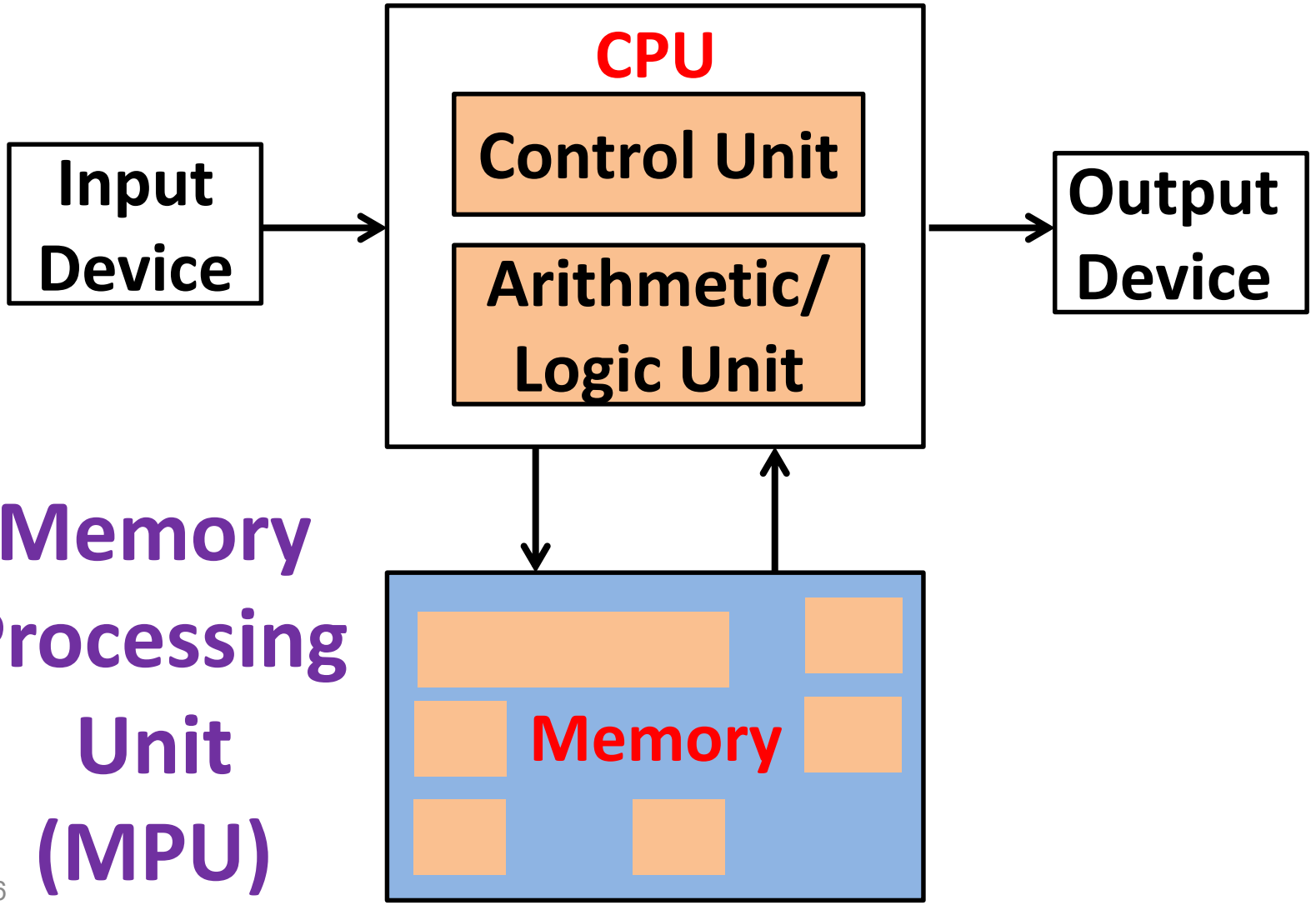
- CMOS logic -> voltage
- Memristive logic -> voltage and resistance
- **Stateful logic family**: resistance is the state variable





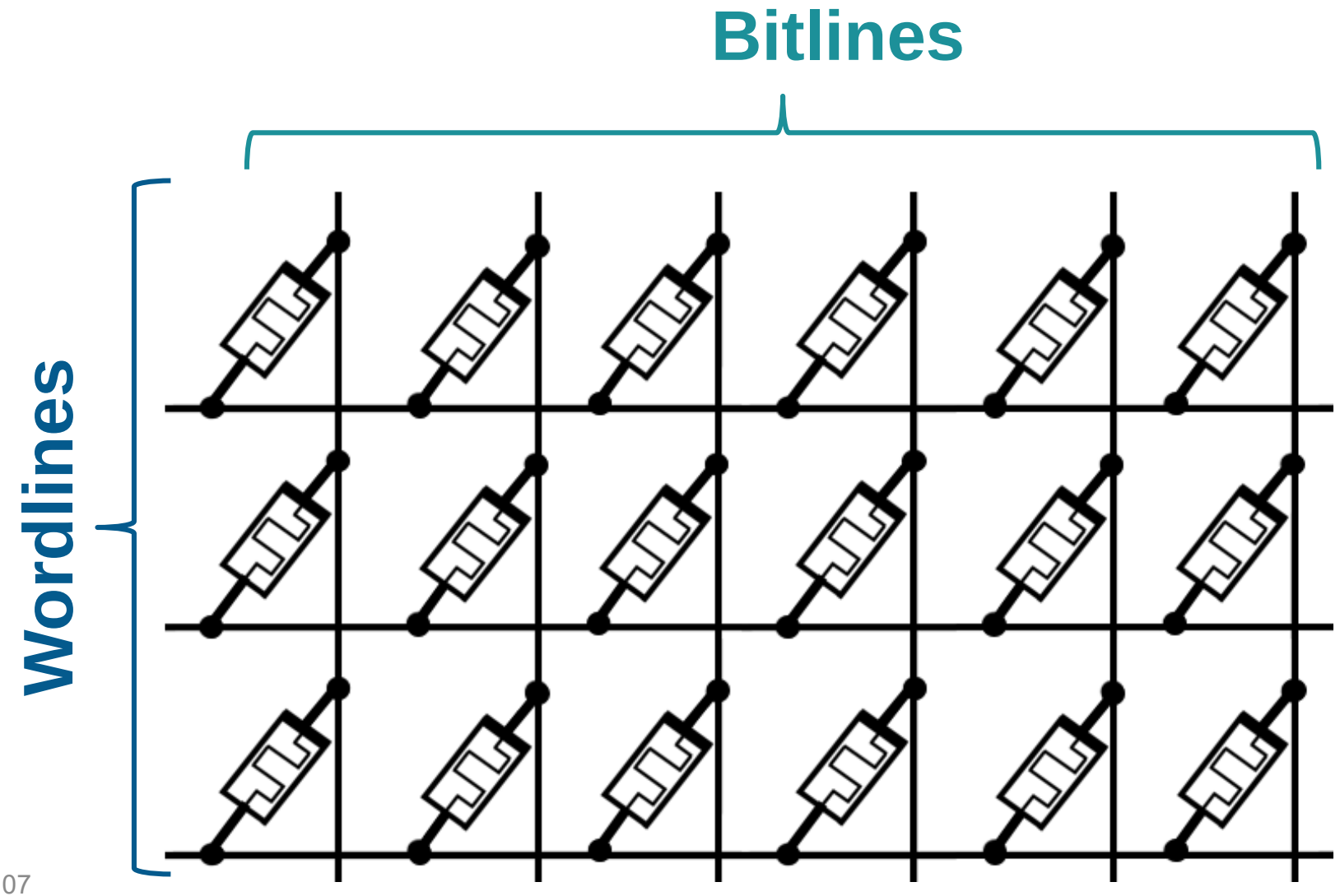
Real Computing within the Memory

Beyond von Neumann Architecture



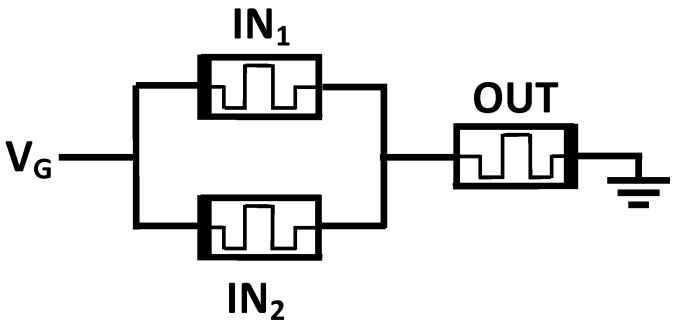
Memory Processing Unit (MPU)

We Want to Compute within the Memristive Crossbar Memory



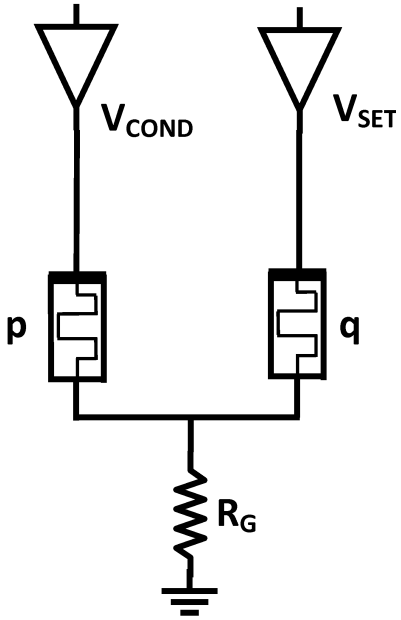
Processing In-Memory with Memristors

Logic Families



MAGIC

Kvatinsky *et al.*, TCAS-II 2014



IMPLY

Borghetti *et al.*, *Nature* 2010

J. Reuben, R. Ben Hur, N. Wald, N. Talati, A. Haj Ali, P.-E. Gaillardon, and S. Kvatinsky, "Memristive Logic: A Framework for Evaluation and Comparison", PATMOS 2017

IMPLY Function

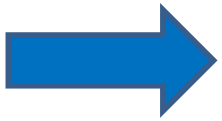
- One of the elementary 2 input Boolean functions

Truth Table

p	q	*p IMP q*
0	0	1
0	1	1
1	0	0
1	1	1

$p \rightarrow q$

If p then q

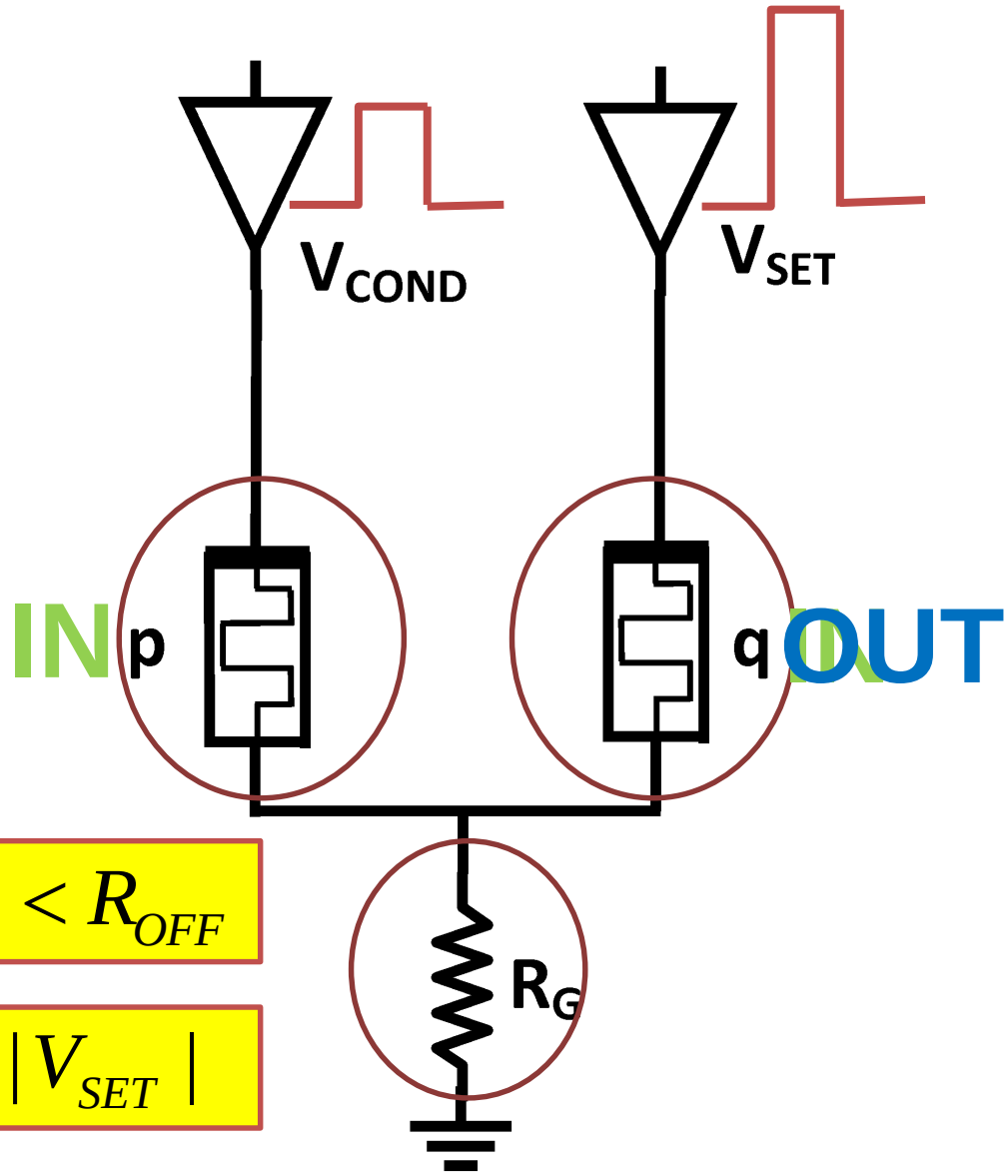
IMPLY + FALSE  **Complete logic**

IMPLY Logic with Memristors

Logic 0 $\rightarrow R_{OFF}$

Logic 1 $\rightarrow R_{ON}$

p	q	*p IMP q*
0	0	1
0	1	1
1	0	0
1	1	1



$$R_{ON} < R_G < R_{OFF}$$

$$|V_{COND}| < |V_{SET}|$$

J. Borghetti et al, "Memristive Switches Enable 'Stateful' Logic Operation via Material Implication," Nature, 2010

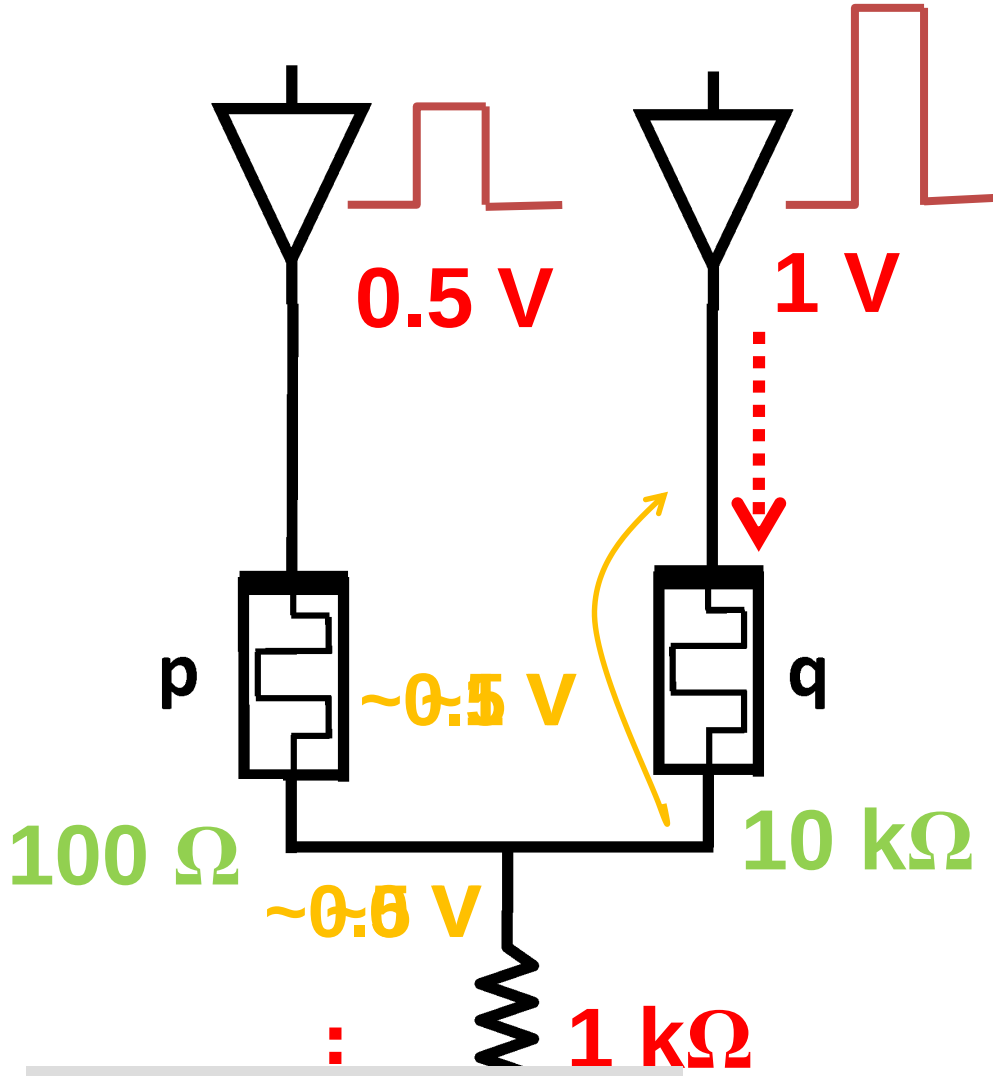
Behavior for Different Cases

$R_{ON} = 100\ \Omega$

$R_{OFF} = 10\ k\Omega$

Case	p	q	*p IMP q → q*
1	0	0	1
2	0	1	1
3	1	0	0
4	1	1	1

IN OUT



Decrease resis

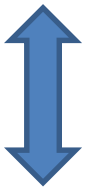
State Drift

Performance and Robustness Tradeoff

Case	p	q	*p IMP q* → *q*
1	0	0	1
2	0	1	1
3	1	0	0
4	1	1	1

IN OUT

Write time



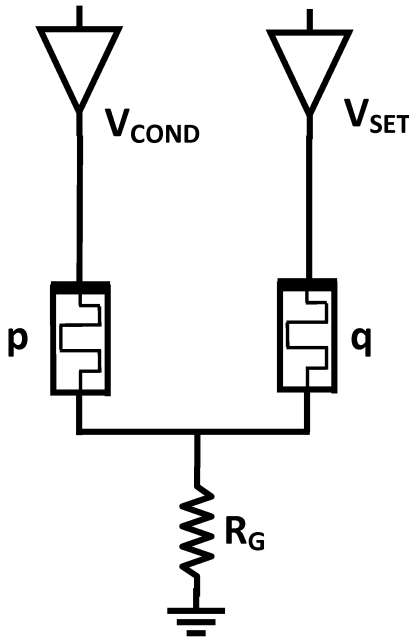
TRADEOFF

State drift

Refreshing the gate

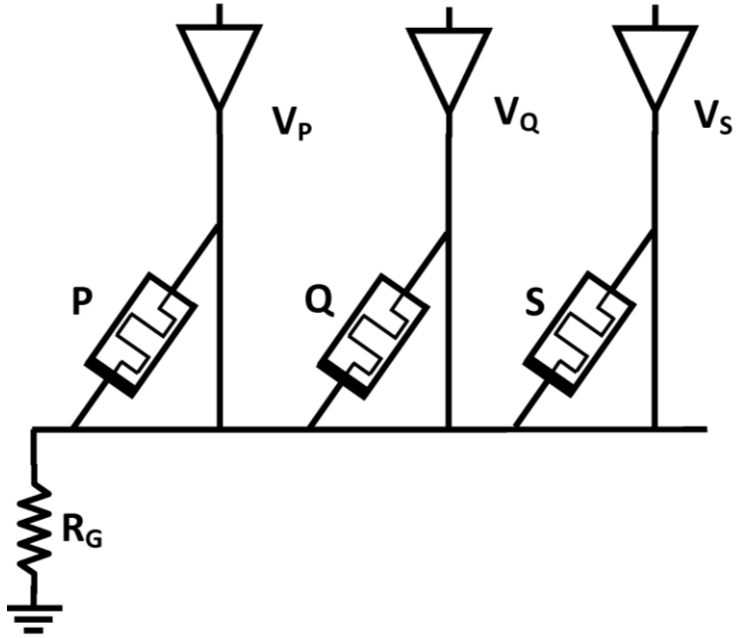
IMPLY – Basic Operation

- Performance and robustness tradeoff
- Need for refresh because of state drift
- Widely used linear ion drift memristor model is incompatible for logic design



Complete Logic with IMPLY

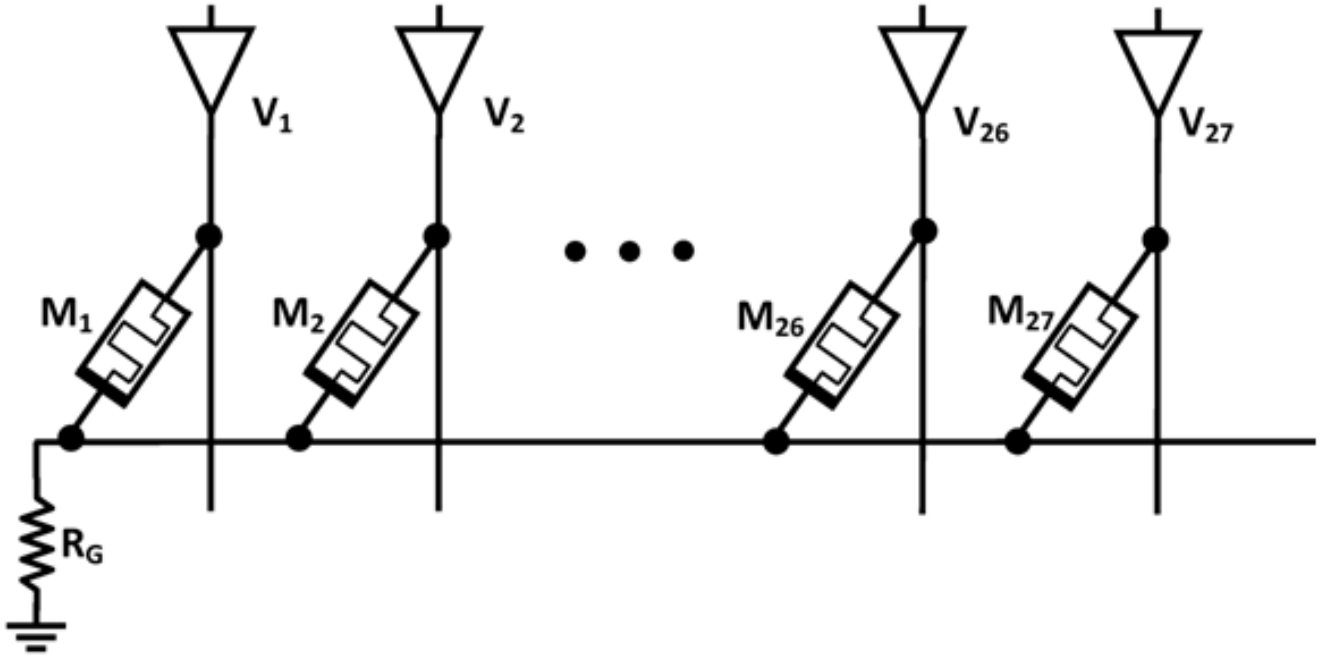
- **Sequential operation** of IMPLY and FALSE
- NAND:
  - Step 1 – FALSE(S)
  - Step 2 – P IMPLY S
  - Step 3 – Q IMPLY S



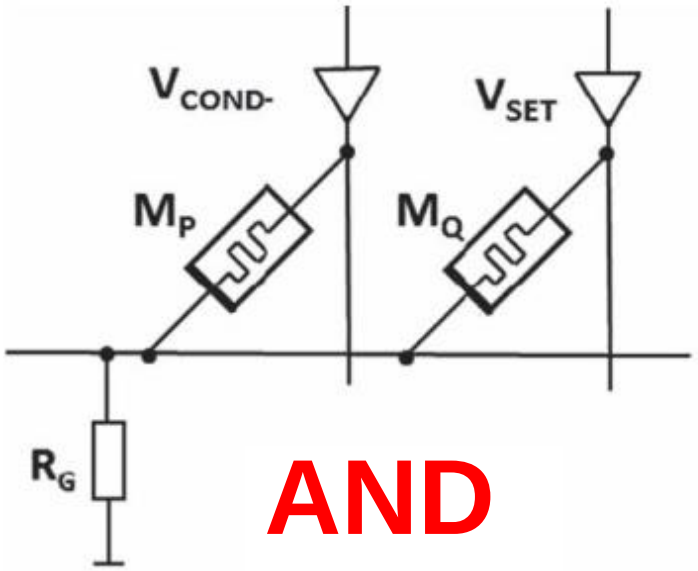
- Many different optimizations for various Boolean functions

IMPLY within a Crossbar

- Natural method within memristive crossbar
  - Input and output are resistance
  - Basic structure is a crossbar-like
  - Additional resistor per wordline

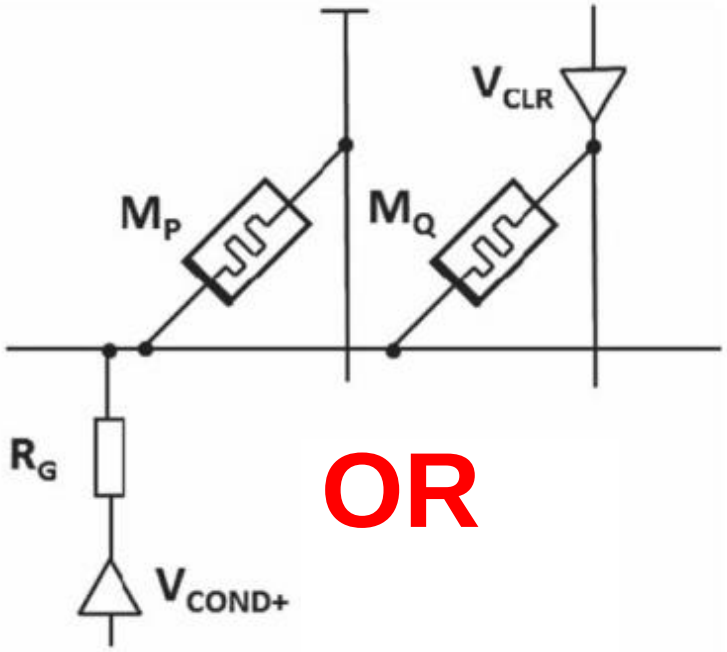


Other Stateful Logic - Example



AND

p	q	q'
0	0	0
1	0	0
0	1	0
1	1	1



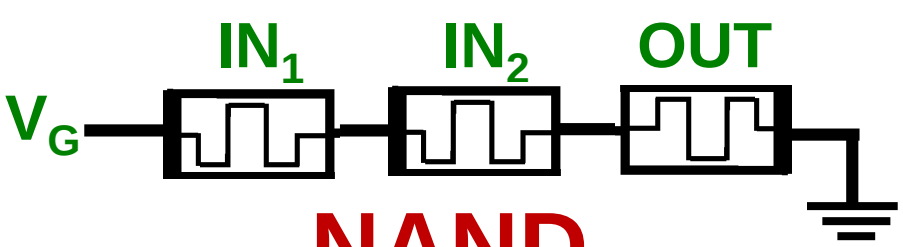
OR

p	q	q'
0	0	0
1	0	1
0	1	1
1	1	1

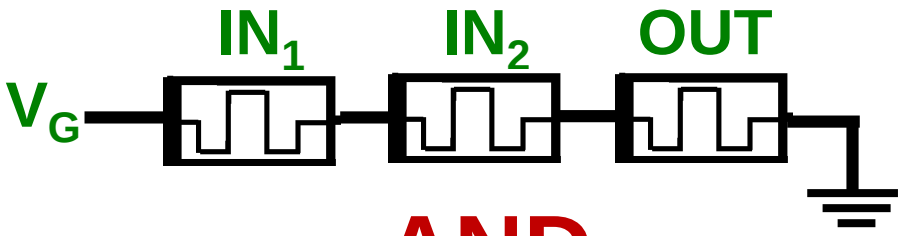
Y. Zhang et al., "A Novel Design for a Memristor-Based OR Gate," *IEEE TCAS II*, Aug. 2015

MAGIC – Memristor Aided LoGIC

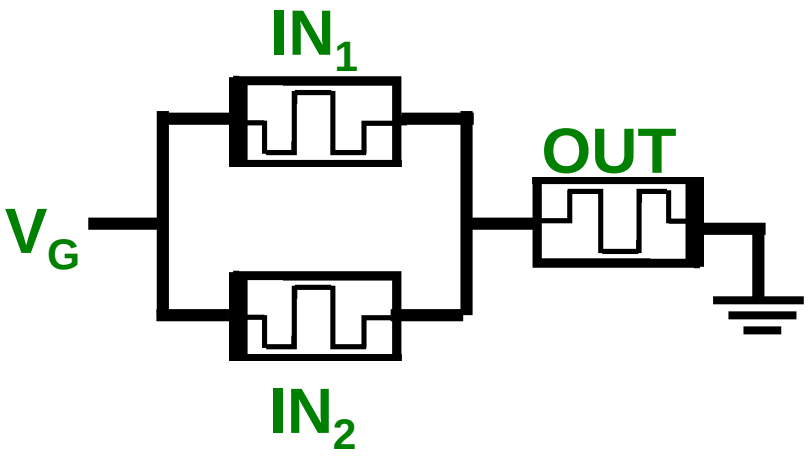
- One applied voltage V_G, no additional devices
- Separate input and output memristors



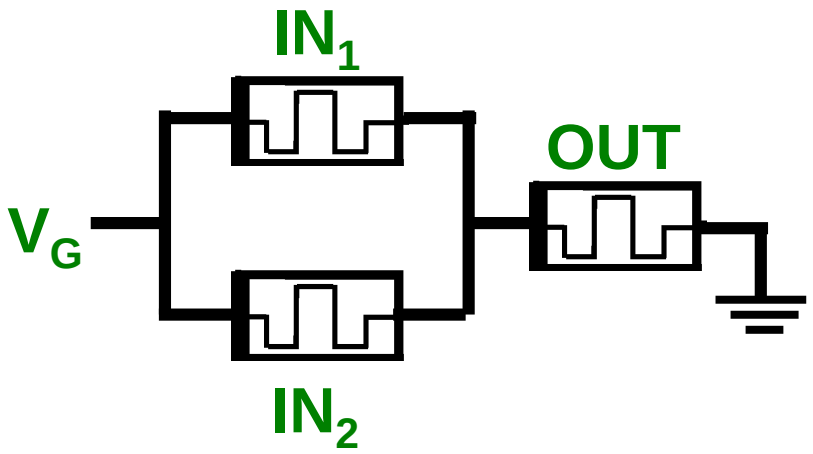
NAND



AND



NOR



OR

S. Kvatinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "MAGIC – Memristor Aided LoGIC," IEEE TCAS II, Nov. 2014

MAGIC – Memristor Aided LoGIC

Example of MAGIC NOR

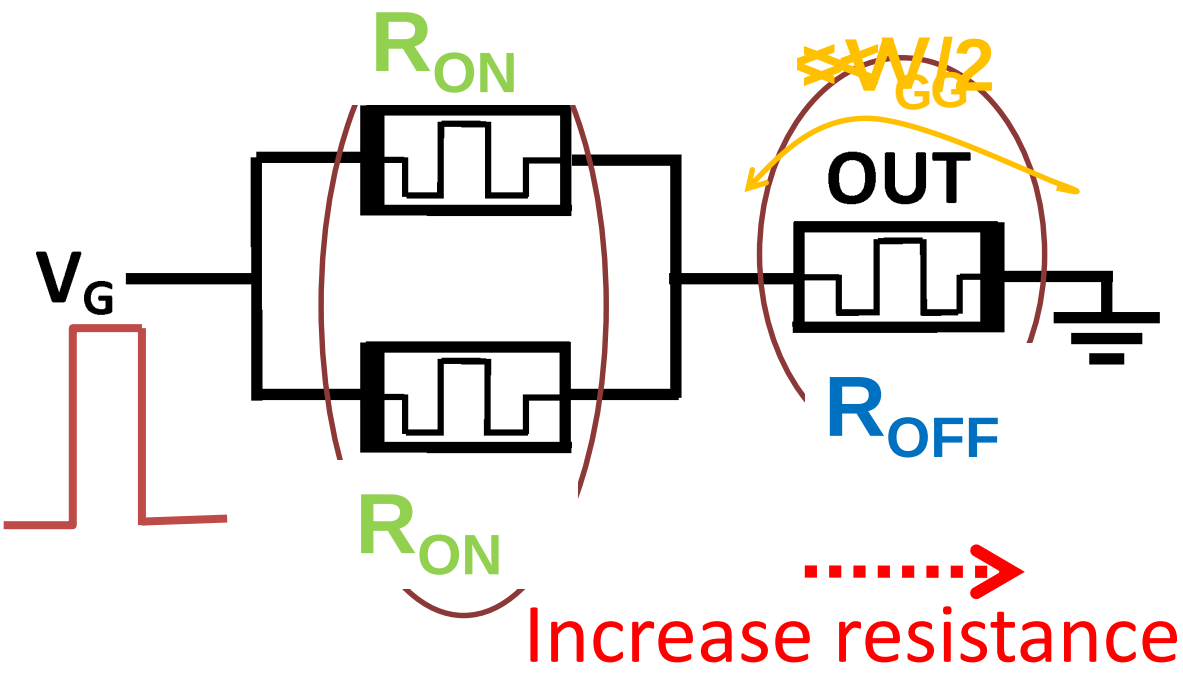
Initialize OUT to R_{ON}

$R_{OFF} >> R_{ON}$

R_{ON} =Logic '1'

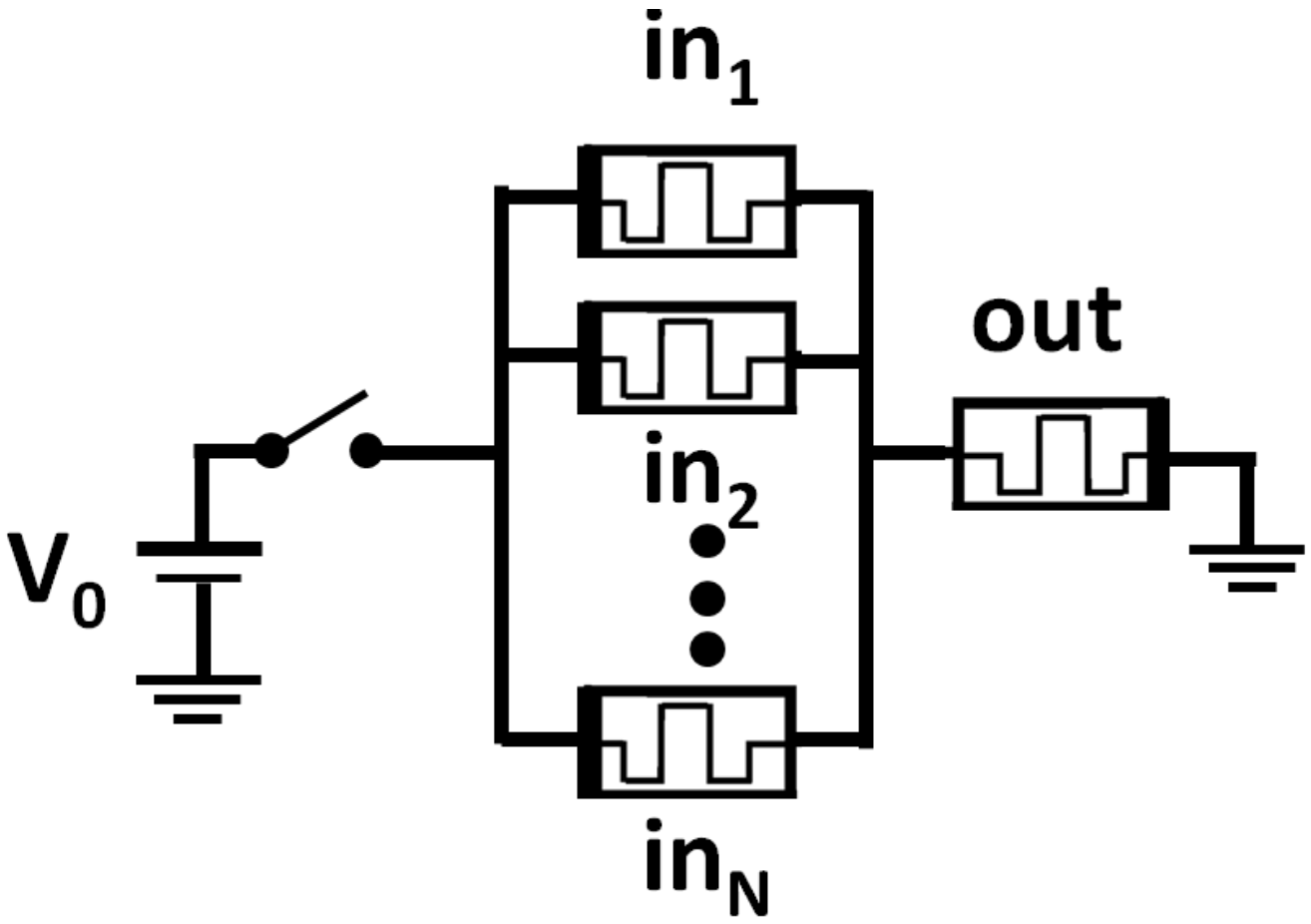
R_{OFF}=Logic '0'

IN_1	IN_2	*NOR*
0	0	1
0	1	0
1	0	0
1	1	0

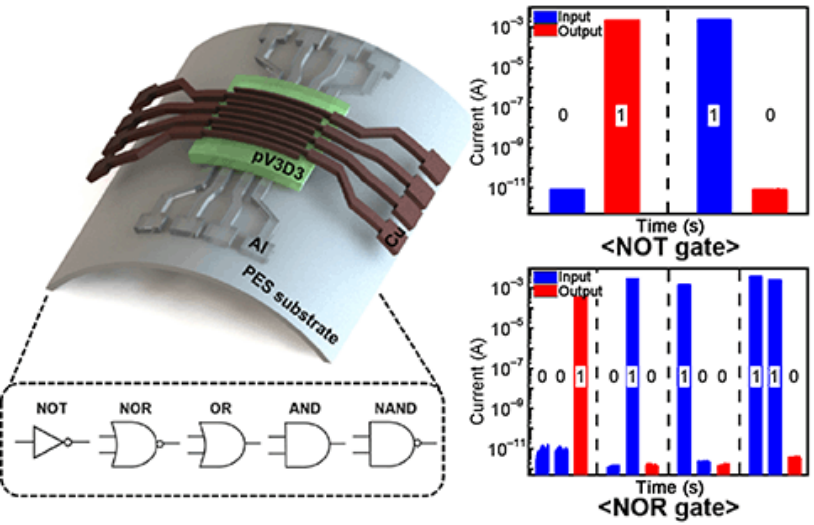


S. Kvatinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "MAGIC – Memristor Aided LoGIC," IEEE TCAS II, Nov. 2014

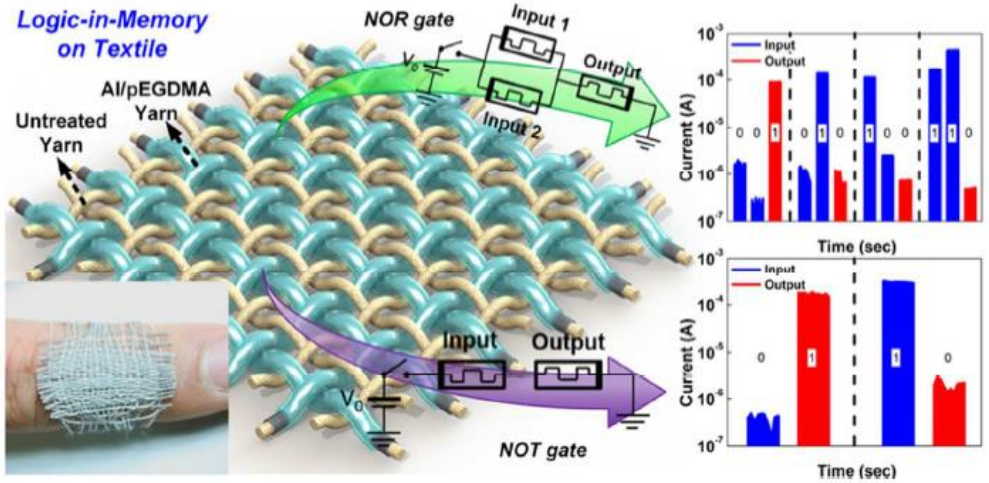
N-Input MAGIC NOR



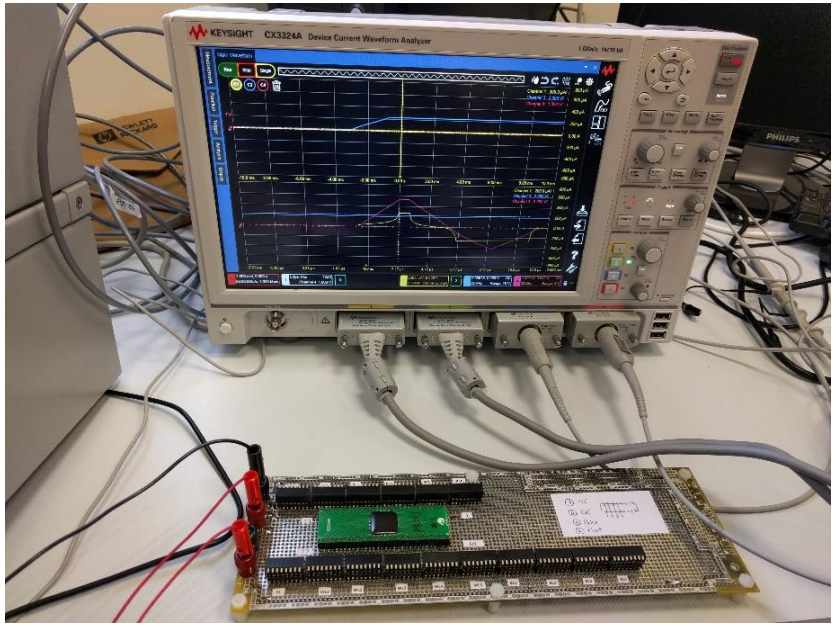
Real MAGIC



Jung *et al.*, *Nano Research*, July 2017



Bae *et al.*, *Nano Letters*, Oct. 2017



ASIC² winbond

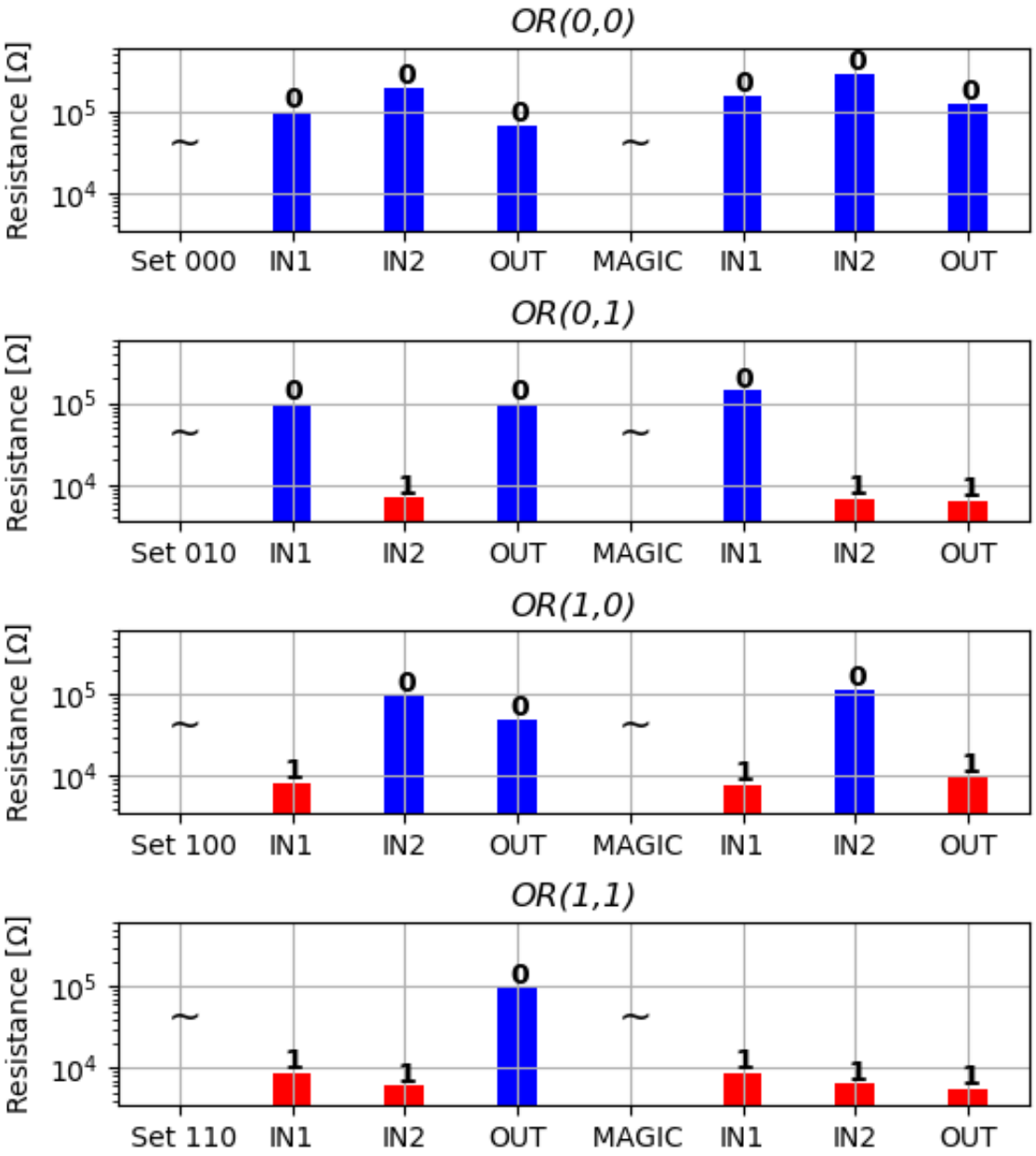
Our lab (HfOx based)

Lab Demonstration

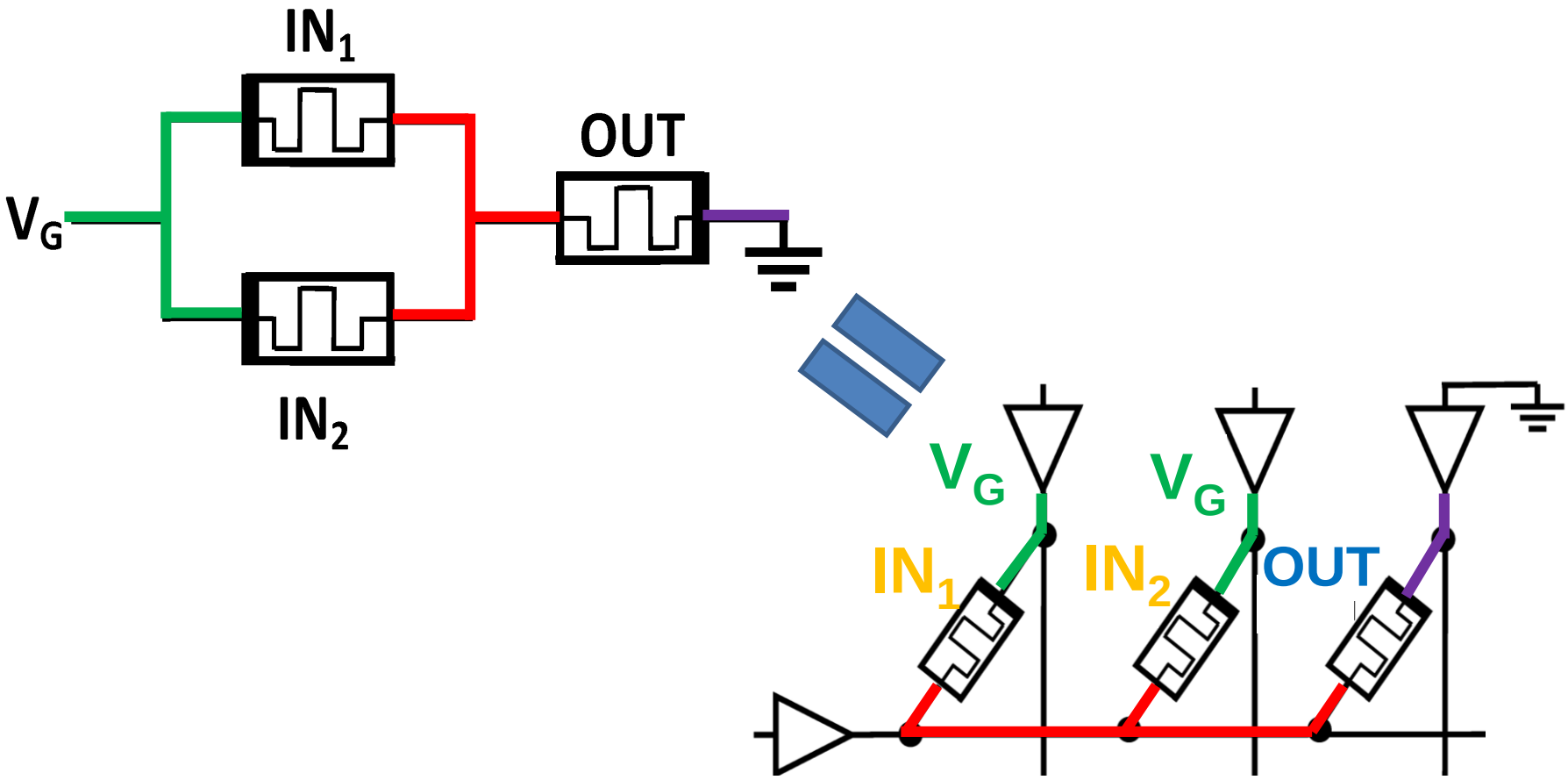
TaOx memristors

Fabricated by Vikas Rana (Julich)
Tested by Barak Hoffer (Technion)

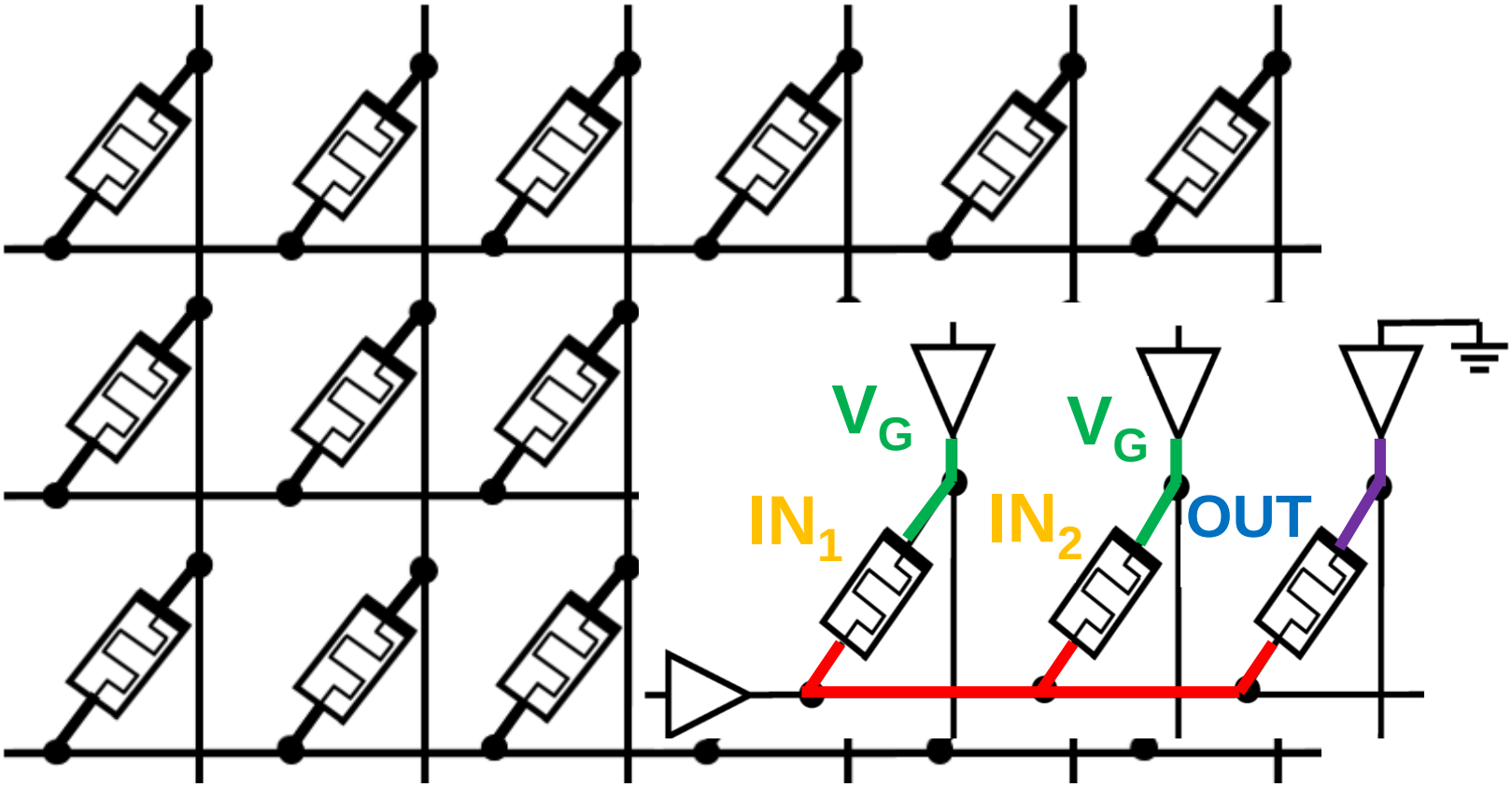
(Unpublished)



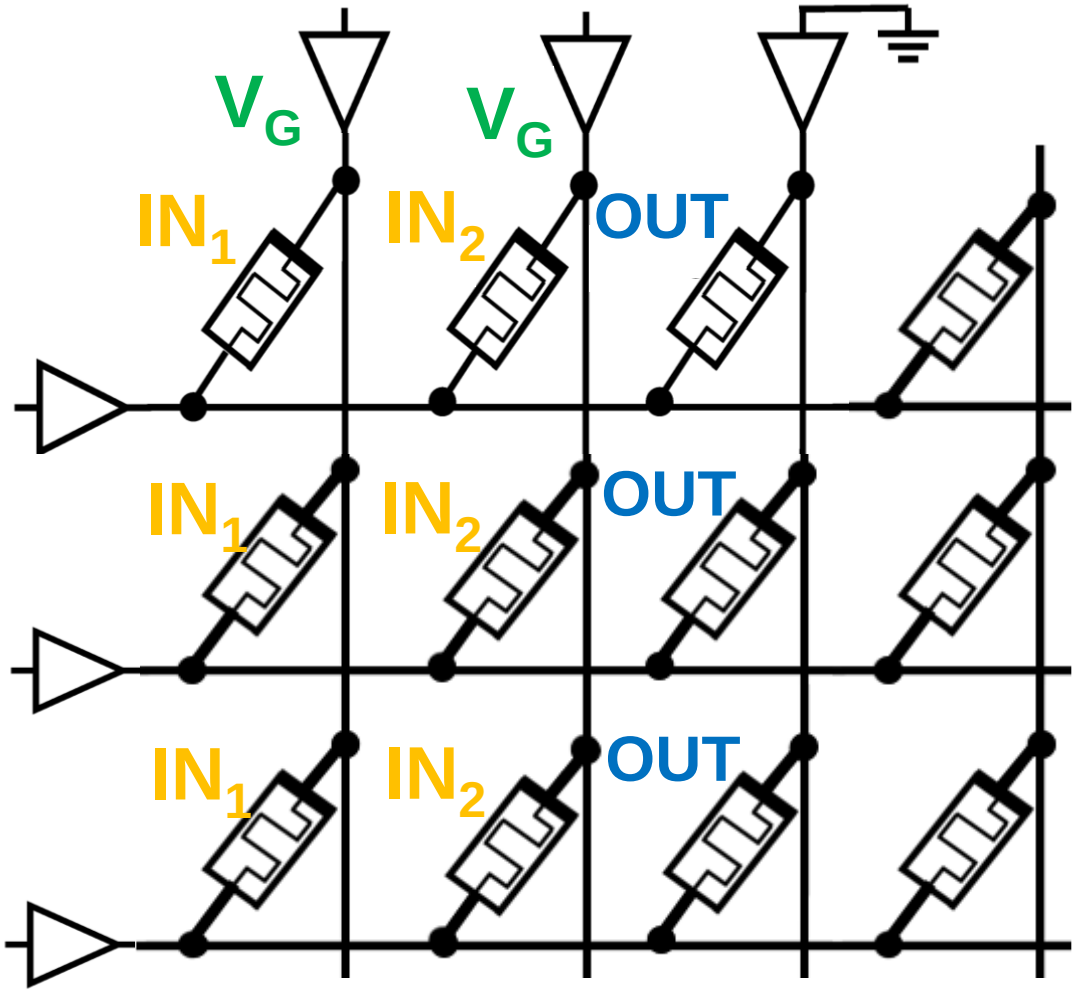
MAGIC NOR in a Crossbar



MAGIC NOR in a Crossbar



MAGIC NOR in a Memristive Memory



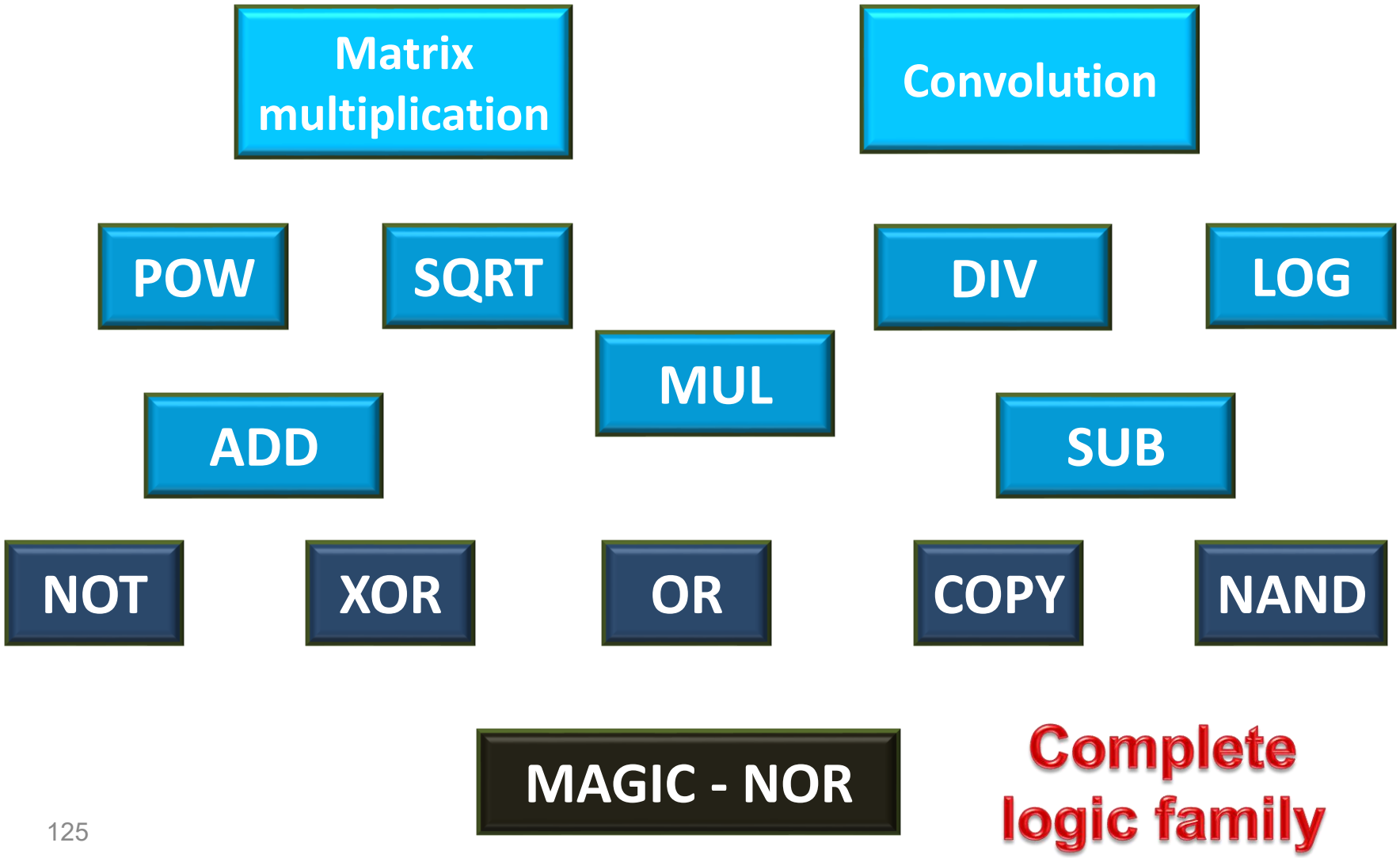
Parallelism

↓

SIMD

N. Talati, S. Gupta, P. Mane, and S. Kvatinsky, "Logic Design within Memristive Memories Using MAGIC," IEEE Transactions on Nanotechnology, July 2016

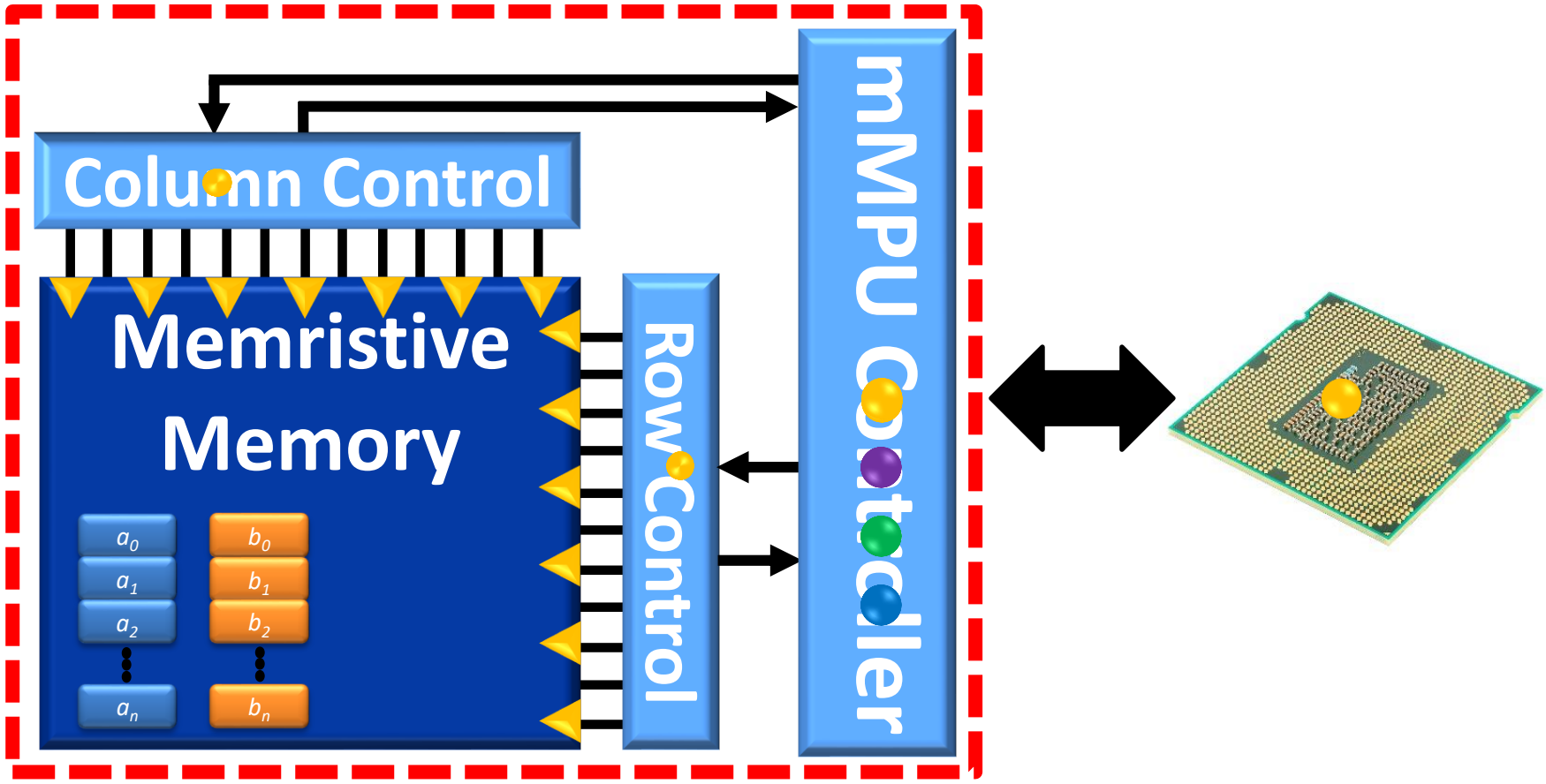
Hierarchy of Logical Functions



Agenda

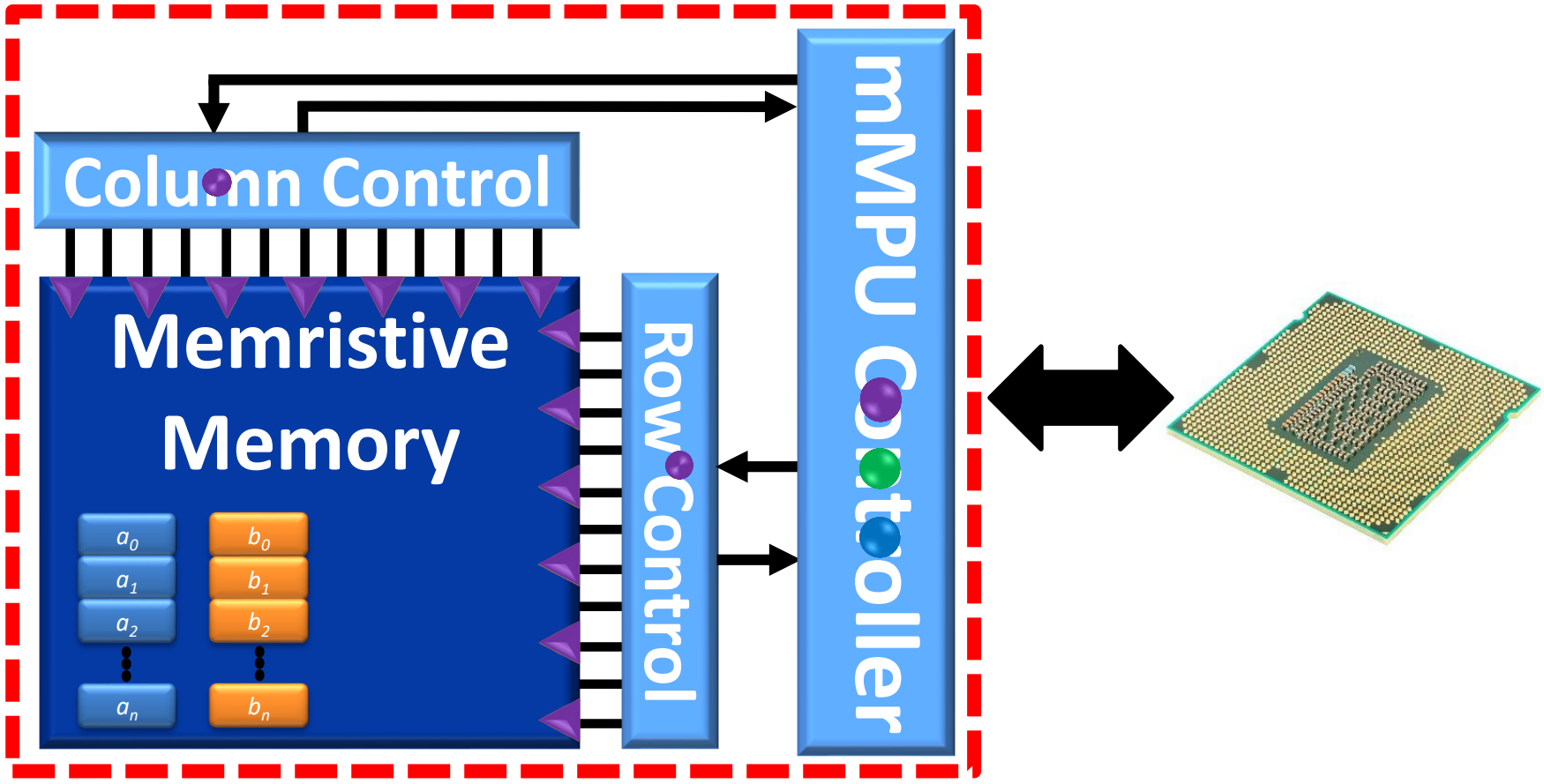
- The need for Processing-in-Memory
- Memristors and Memristive Memory
- Memristive logic classification
- Out-of-memory logic
- Near-memory logic
- **Real in-memory logic**
  - Stateful logic – IMPLY, MAGIC
  - **Memristive Memory Processing Unit (mMPU)**
- Memristor for HW security

mMPU µArchitecture



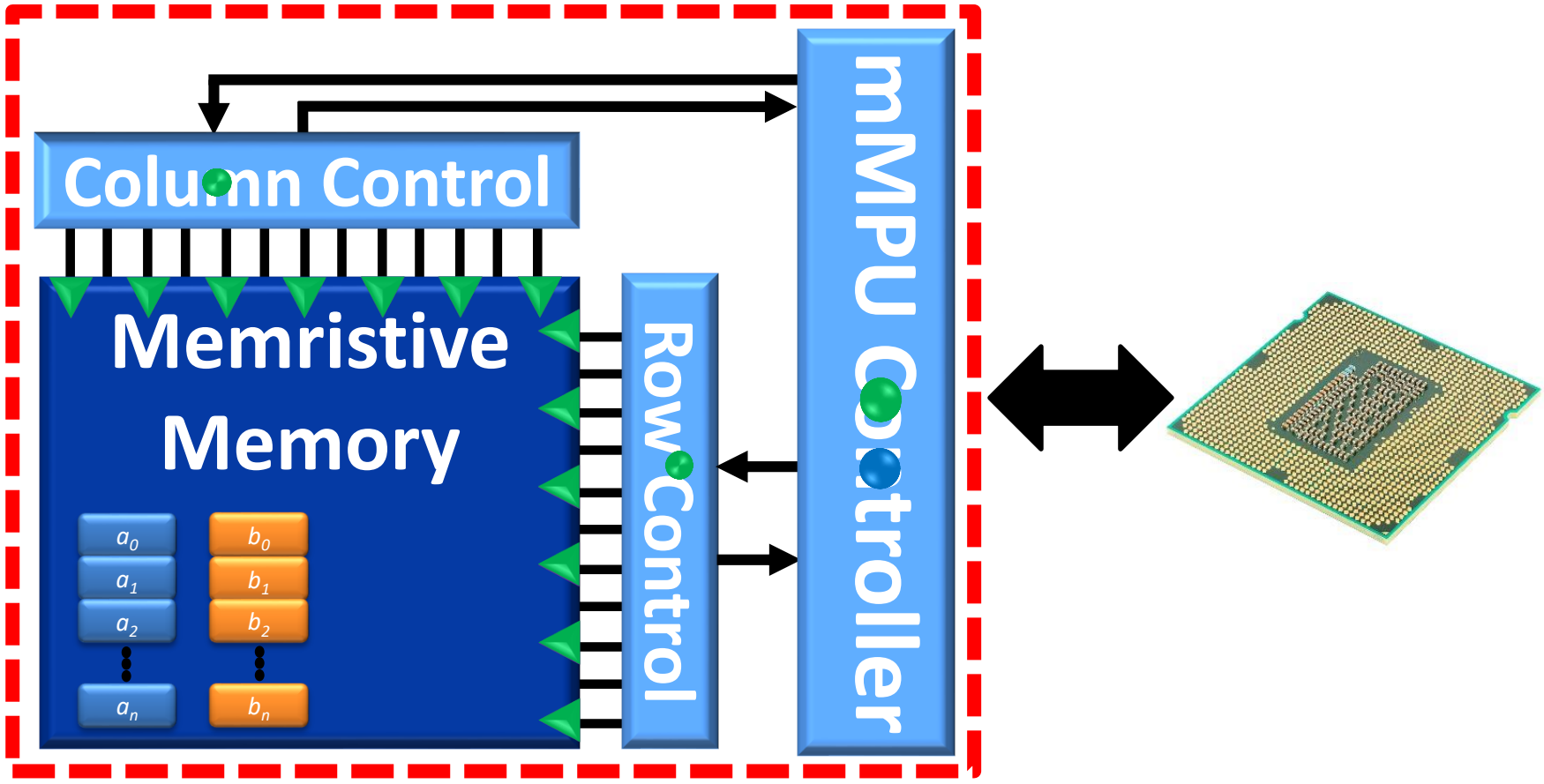
R. Ben-Hur and S. Kvatinsky, "Memory Processing Unit for In-Memory Processing," NANOARCH 2016

mMPU µArchitecture



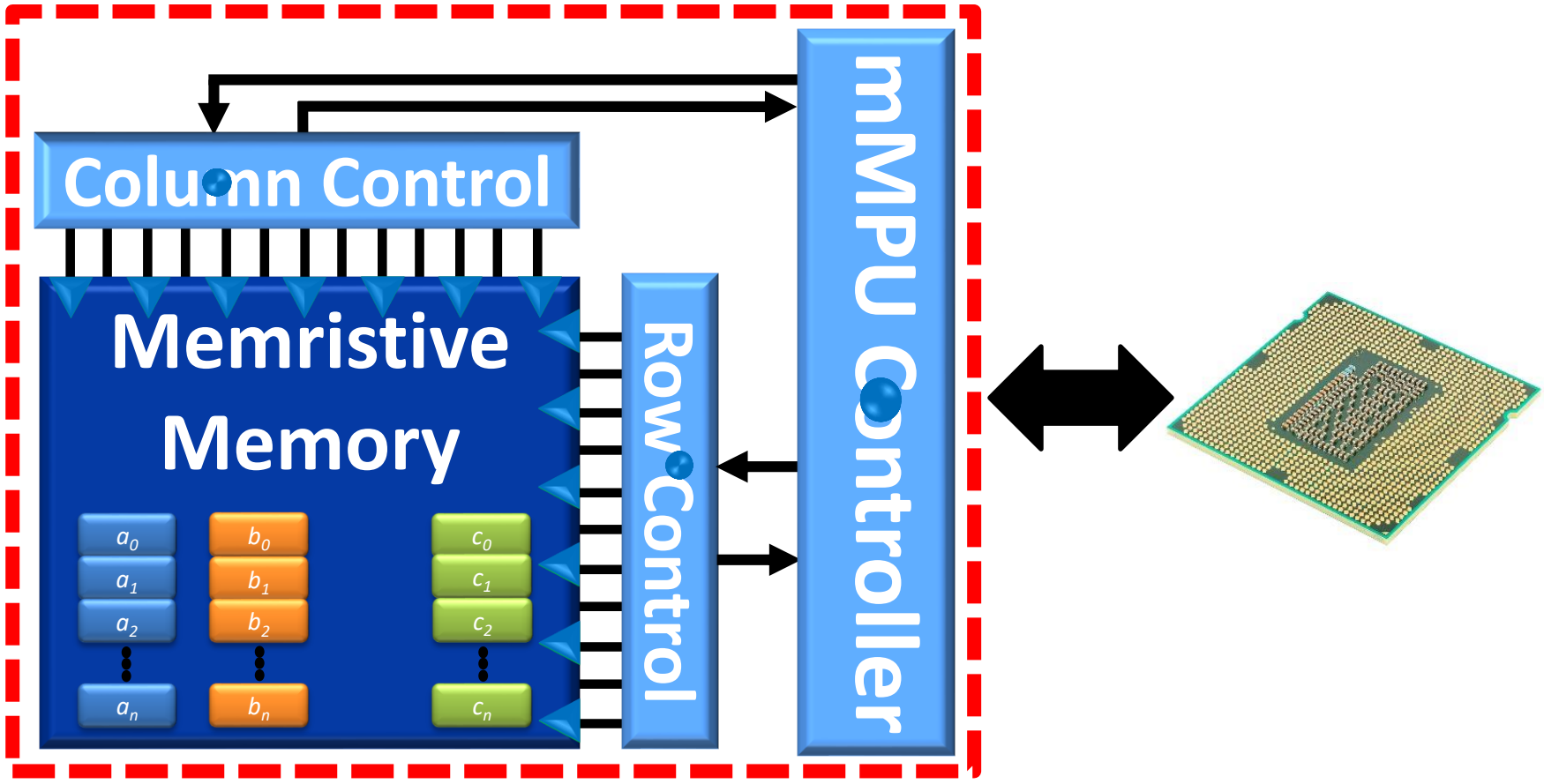
R. Ben-Hur and S. Kvatinsky, "Memory Processing Unit for In-Memory Processing," NANOARCH 2016

mMPU µArchitecture



R. Ben-Hur and S. Kvatinsky, "Memory Processing Unit for In-Memory Processing," NANOARCH 2016

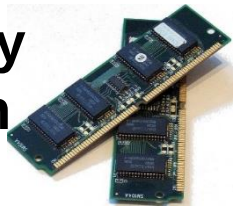
mMPU µArchitecture



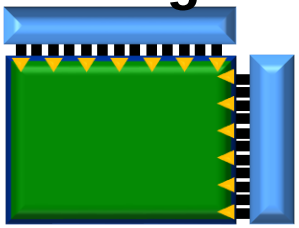
R. Ben-Hur and S. Kvatinsky, "Memory Processing Unit for In-Memory Processing," NANOARCH 2016

Issues Involved in mMPU System

Memory Design



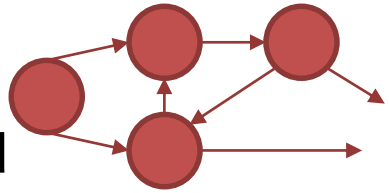
Periphery Design



Programming Model

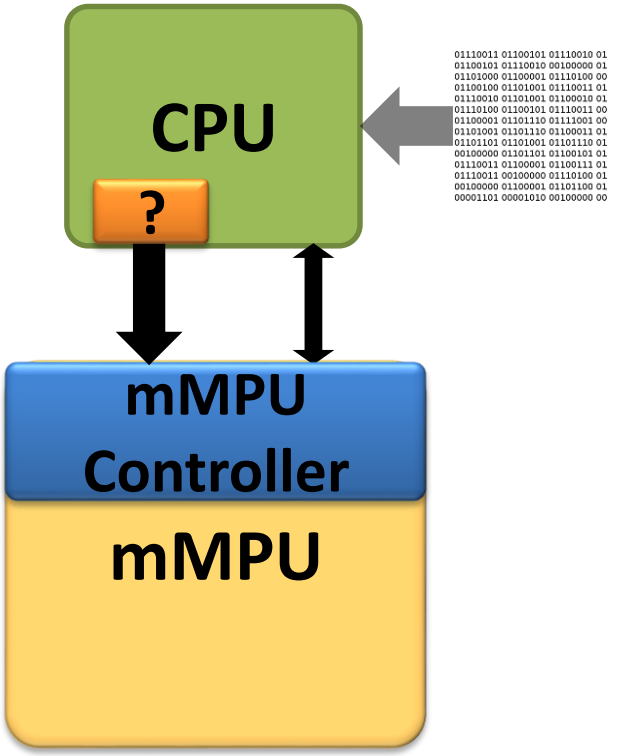
Software

mMPU Controller Design and Optimization



Applications

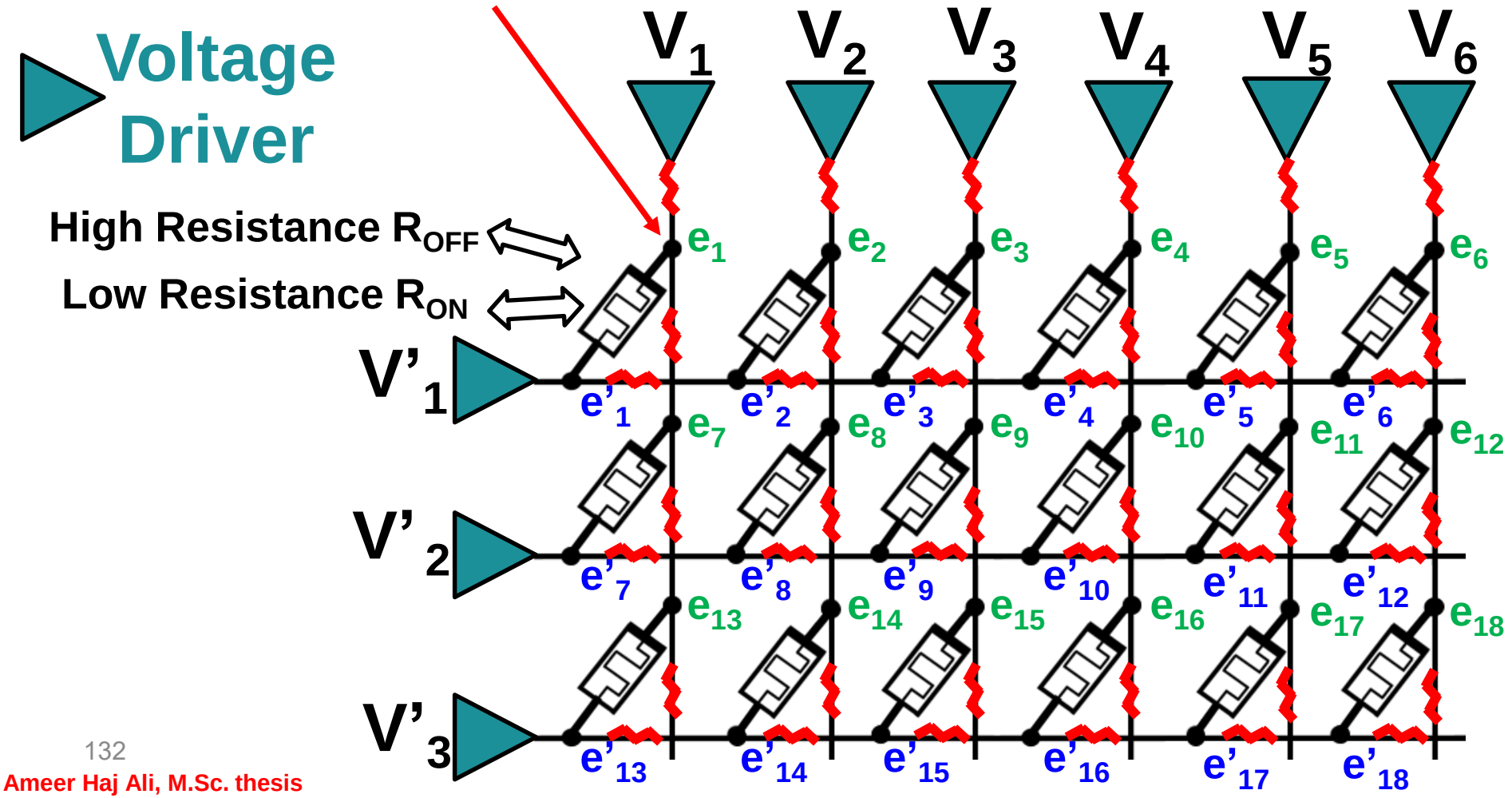
mMPU System



Circuit and Memory Example

Node Analysis Tool

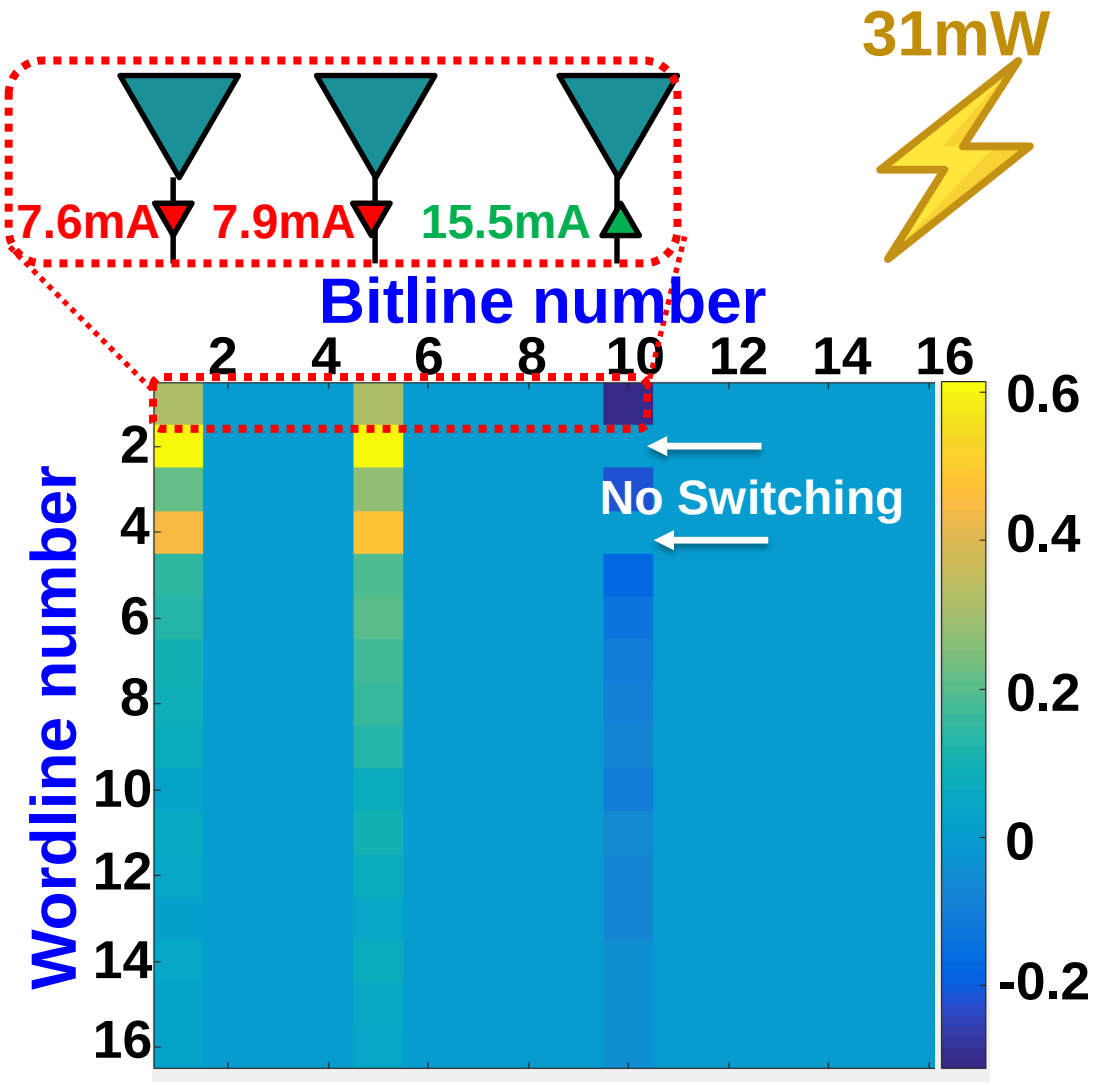
$$KCL: \frac{(v_1 - e_1)}{r_w} + \frac{(e_1 - e_7)}{r_w} + \frac{(e_1 - e'_1)}{R_{1,1}} = 0$$



Outputs of the Tool

Important parameters:

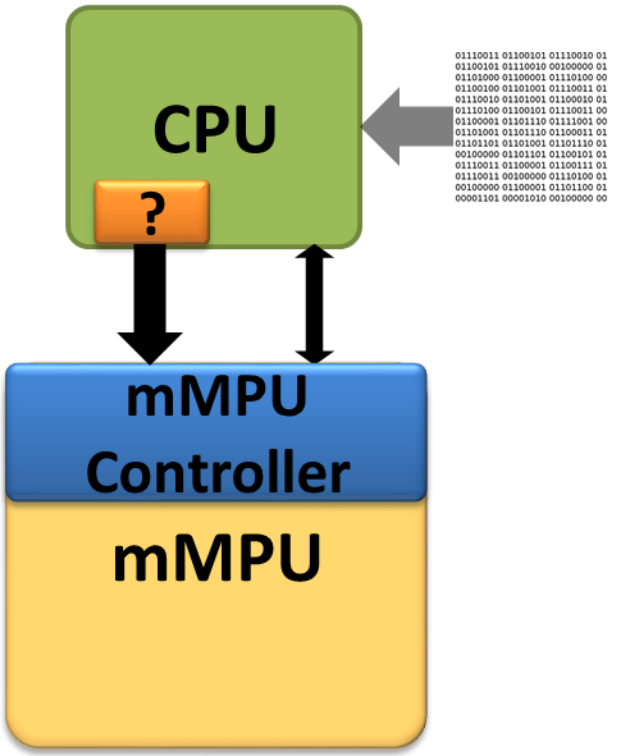
- $1S1R, 16 \times 16$
- $r_w = 4.53\Omega\ (16nm)$
- $R_{ON} = 100\Omega$
- $R_{OFF} = 100k\Omega$
- $V_{MAGIC} = 2V$
- $V_{TH,1} = 0.3\text{V}$
- $V_{TH,2} = -0.9\text{V}$
- $R_{Sel,1} = R_{Sel,2} = 0\Omega$
- $BL_{10} \leftarrow NOR(BL_1, BL_5)$



mMPU Controller

- Supports regular memory operations (R-DDR)
- Optimized logic flow:
  - Parallelism (in-array and banks)
  - Cost function

    (latency, throughput, area, energy)
- Real-time memory mapping

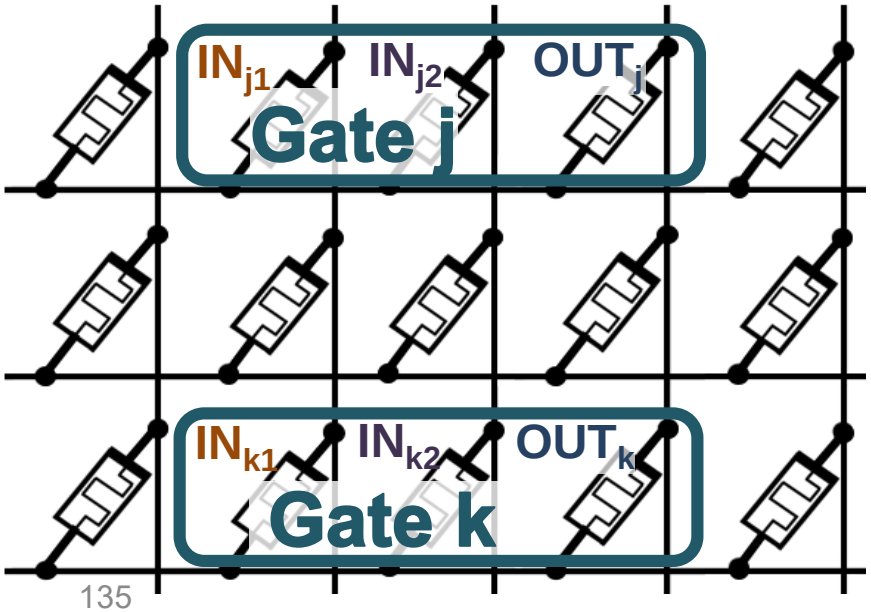


N. Talati *et al.*, "CONCEPT: A Column Oriented Memory Controller for Efficient Memory and PIM Operations in RRAM," IEEE MICRO 2019
R. Ben Hur *et al.*, "SIMPLE MAGIC: Synthesis and Mapping of Boolean Functions for Memristor Aided Logic (MAGIC)," ICCAD 2017
R. Ben Hur *et al.*, "Algorithmic Considerations in Memristive Memory Processing Units (MPU)," NANOARCH 2016

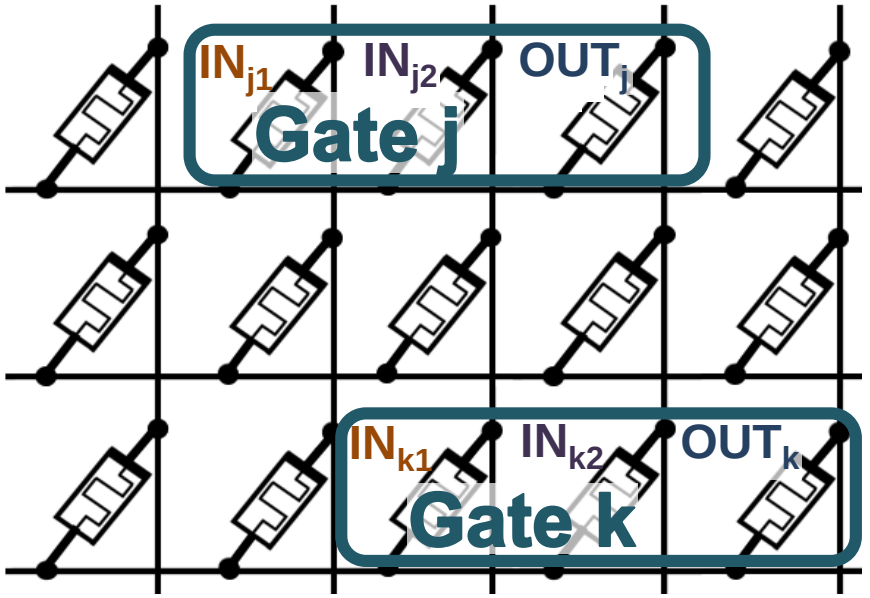
Exploit Parallelism

- Reducing the number of gates is not enough
- Mapping determines the possible parallelism

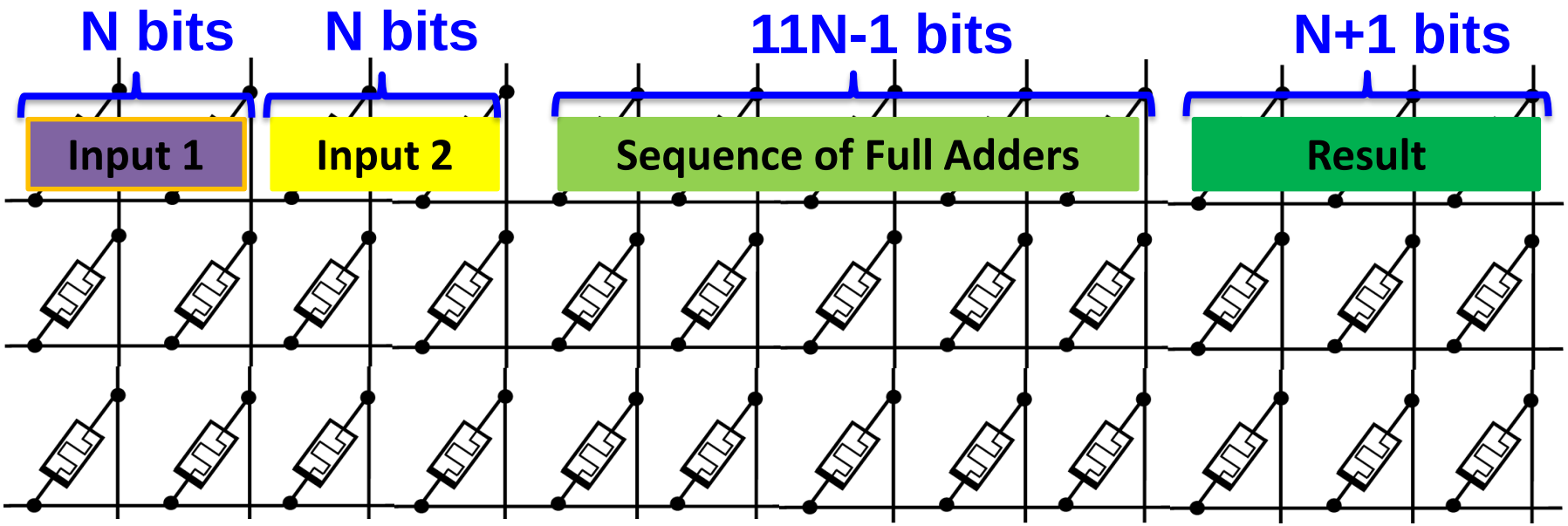
2 gates in 1 step

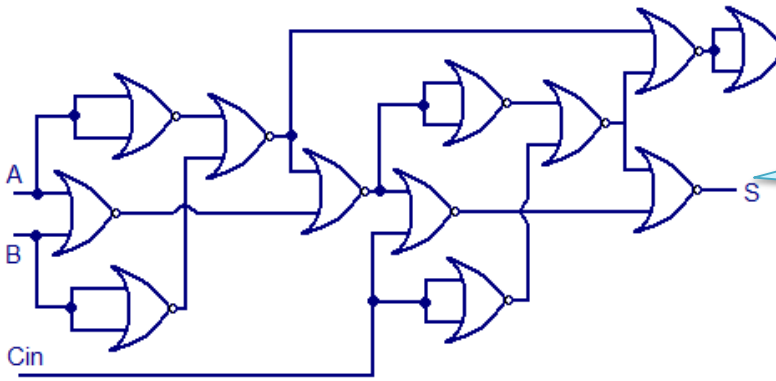


2 gates in 2 steps



Supporting Addition



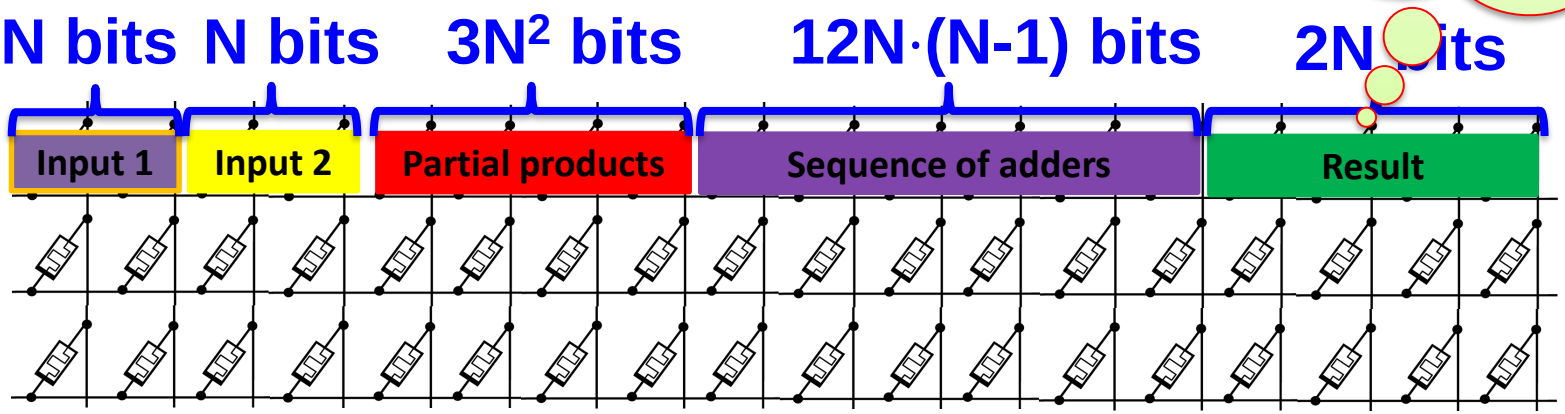


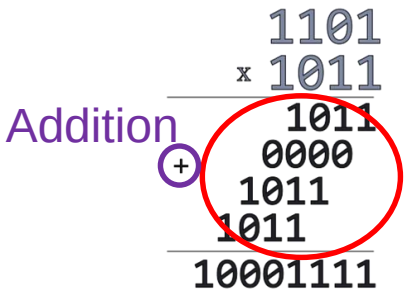
Full Adder using 12 NOR gates

N. Talati, S. Gupta, P. Mane, and S. Kvatinsky, "Logic Design within Memristive Memories Using MAGIC," IEEE Transactions on Nanotechnology, July 2016

Supporting Multiplication – Prior Work

Requires ~3700>>512 columns for N=16



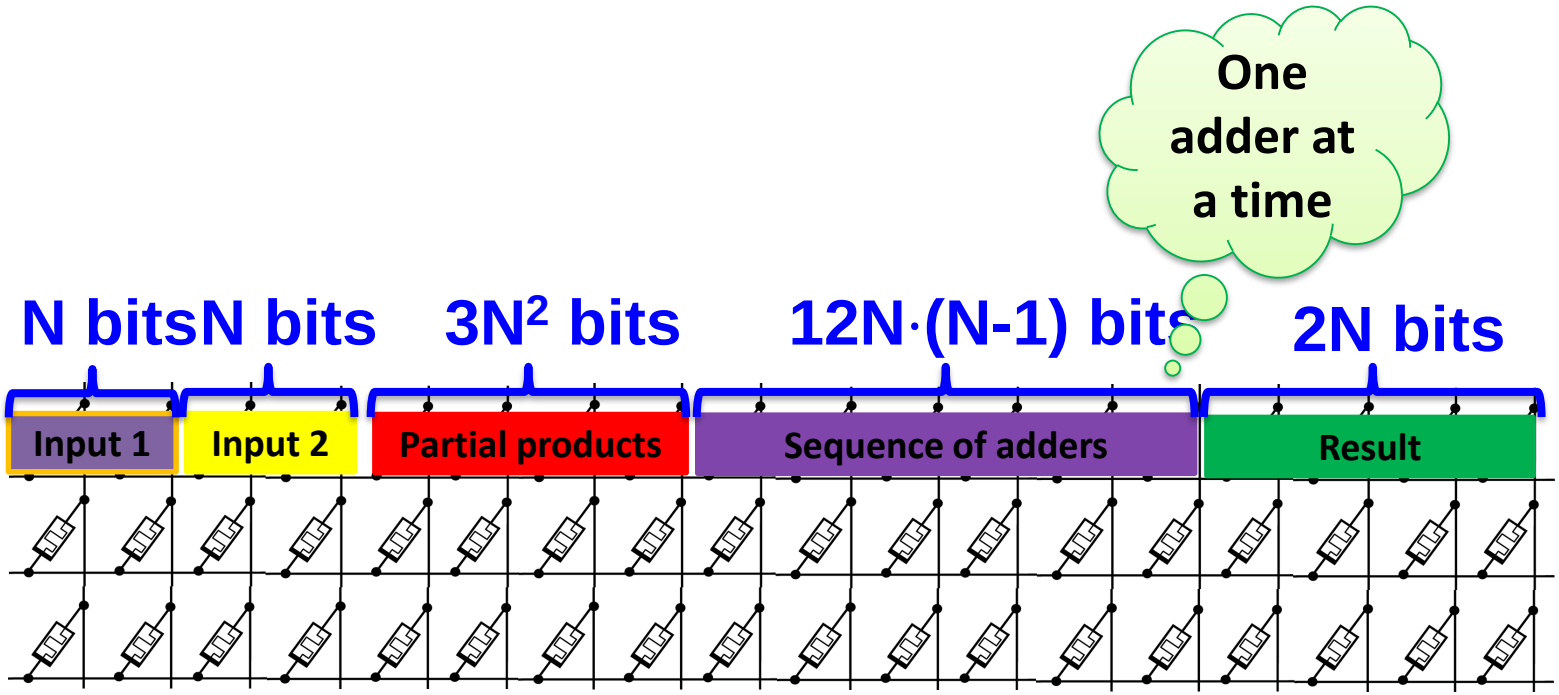


Addition

Partial products ⟺ N^2 AND operations ⟺ $3N^2$ NOR operations

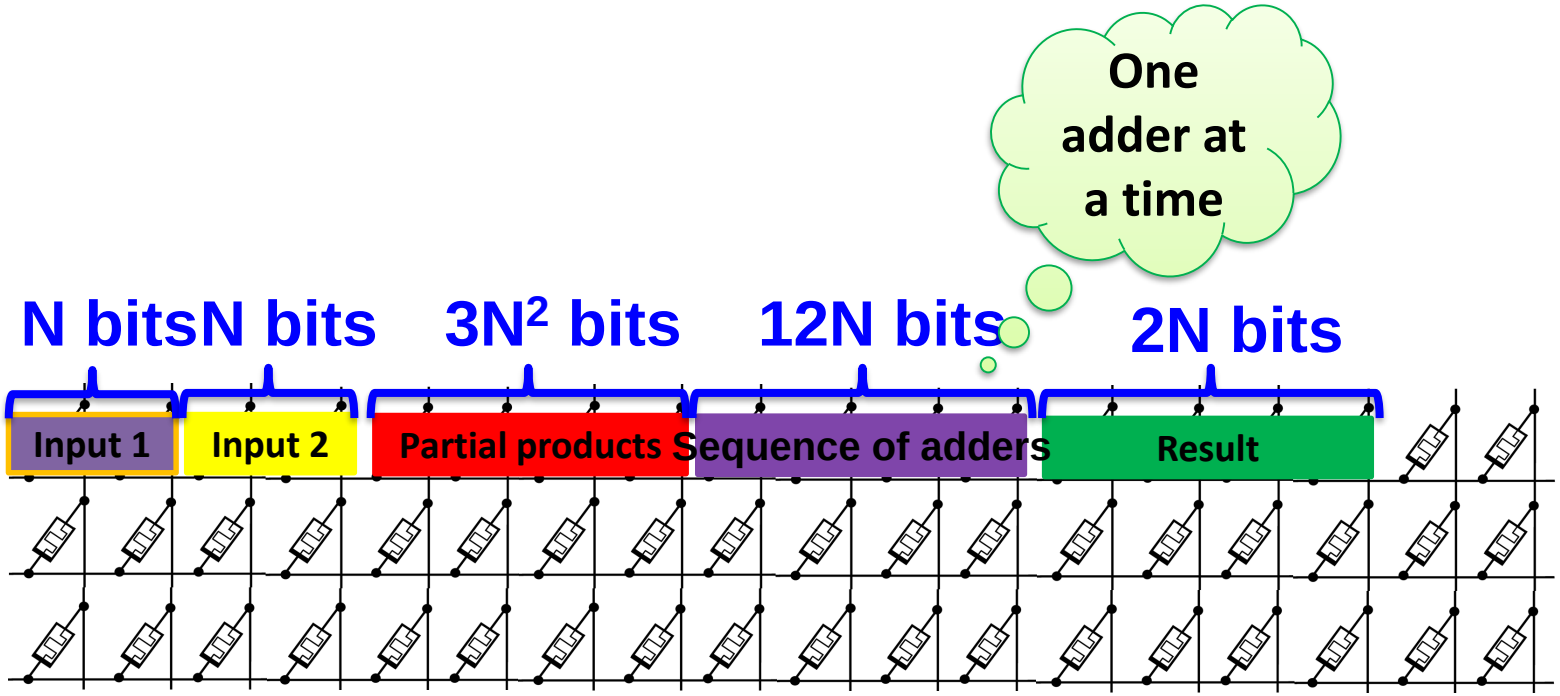
M. Imani *et al.*, "Ultra-Efficient Processing In-Memory for Data Intensive Applications," DAC 2017

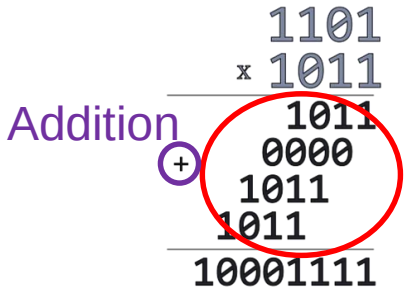
Supporting Multiplication – Solution



A. Haj-Ali, R. Ben Hur, N. Wald, and S. Kvatinsky, "Efficient Algorithms for In-memory Fixed Point Multiplication Using MAGIC," ISCAS 2018

Supporting Multiplication – Solution





Addition

Partial products ⟺ N^2 AND operations ⟺ $3N^2$ NOR operations

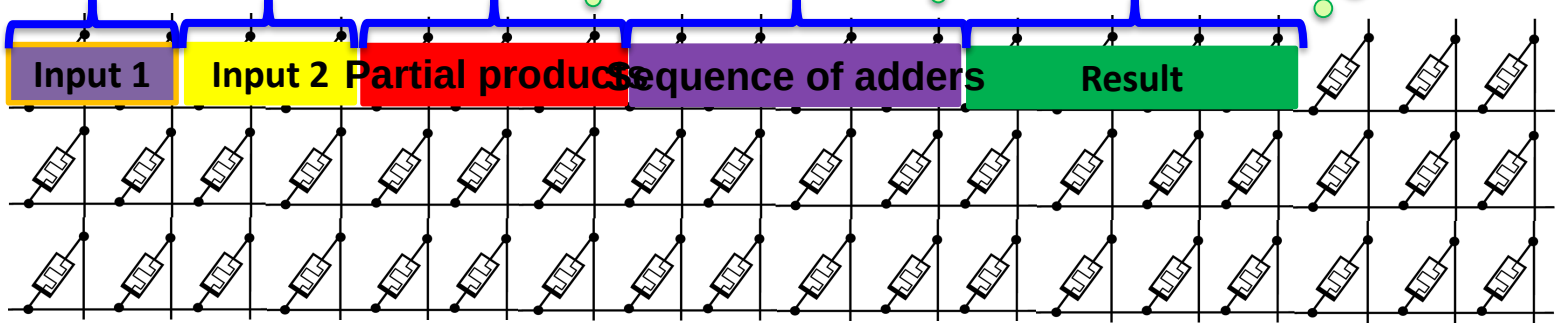
A. Haj-Ali, R. Ben Hur, N. Wald, and S. Kvatinsky, "Efficient Algorithms for In-memory Fixed Point Multiplication Using MAGIC," ISCAS 2018

Supporting Multiplication – Solution

One partial product at a time

One adder at a time

Requires ~300<512 columns for N=16

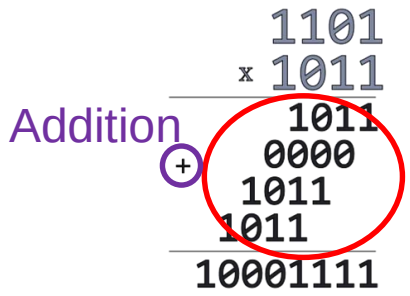


3700

12×

Area

300



Addition

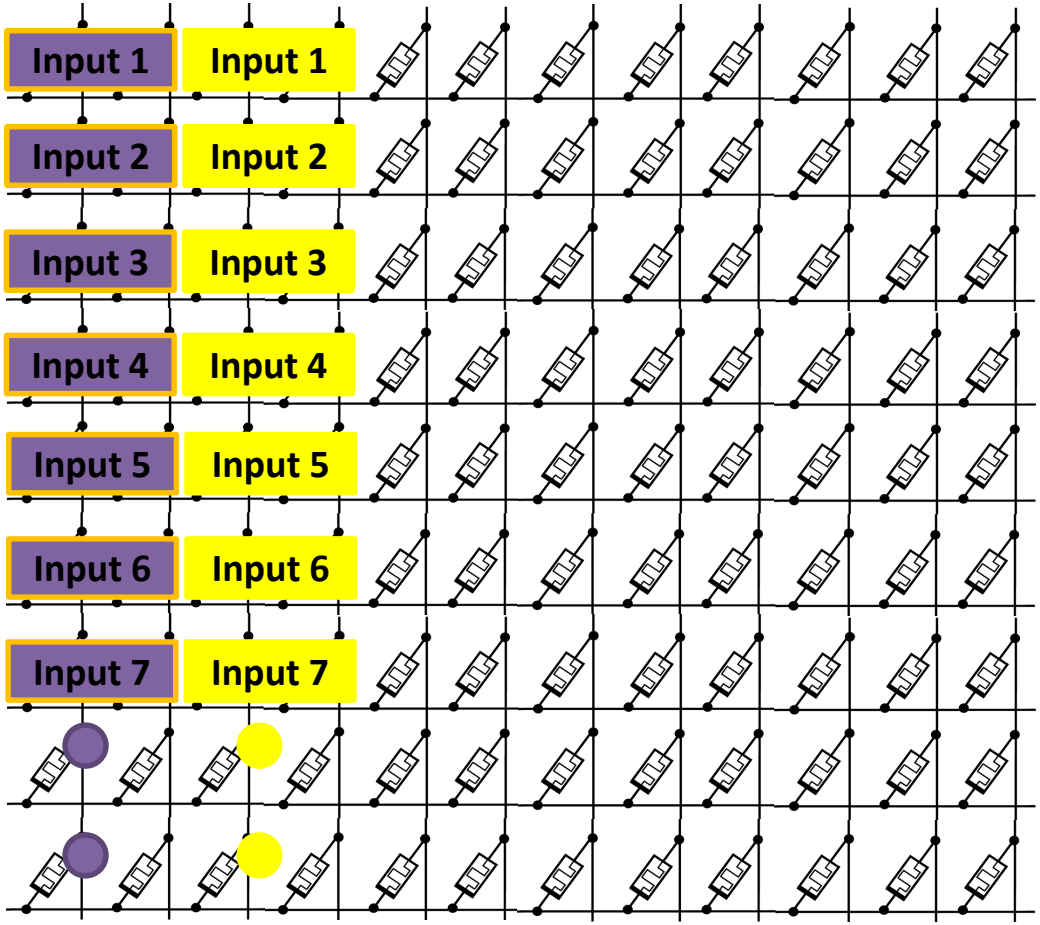
Partial products ⇔ N^2 AND operations ⇔ $3N^2$ NOR operations

More optimizations in the paper

A. Haj-Ali, R. Ben Hur, N. Wald, and S. Kvatinsky, "Efficient Algorithms for In-memory Fixed Point Multiplication Using MAGIC," ISCAS 2018

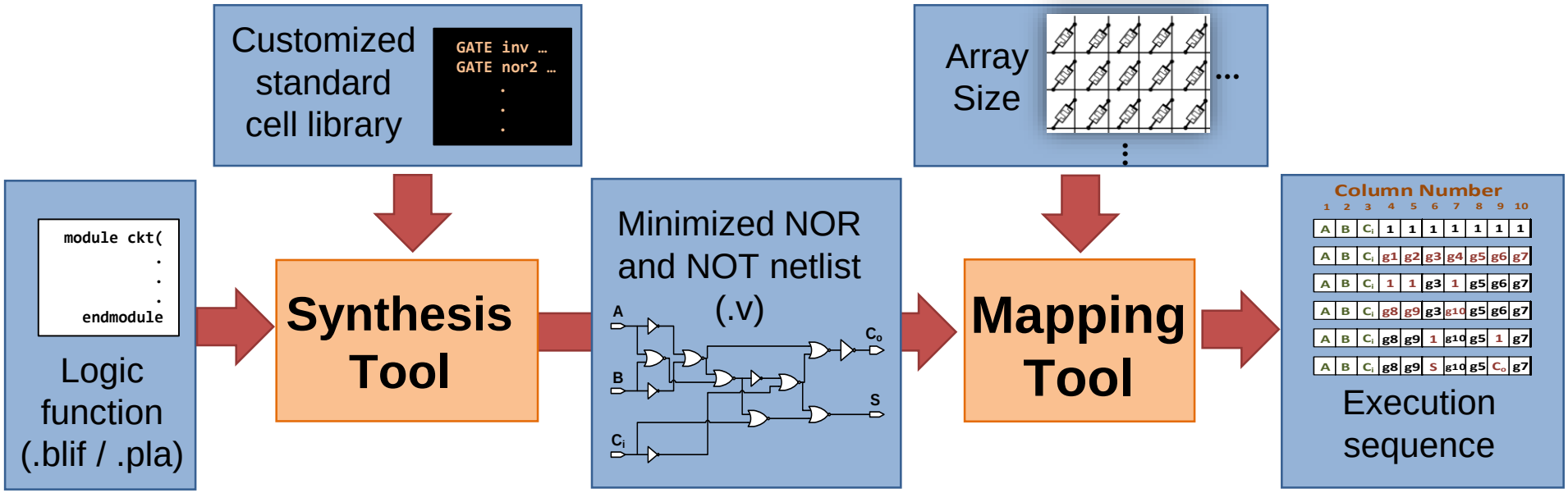
Single Row Execution

Parallelism!



A. Haj-Ali, R. Ben Hur, N. Wald, and S. Kvatinsky, "Efficient Algorithms for In-memory Fixed Point Multiplication Using MAGIC," ISCAS 2018

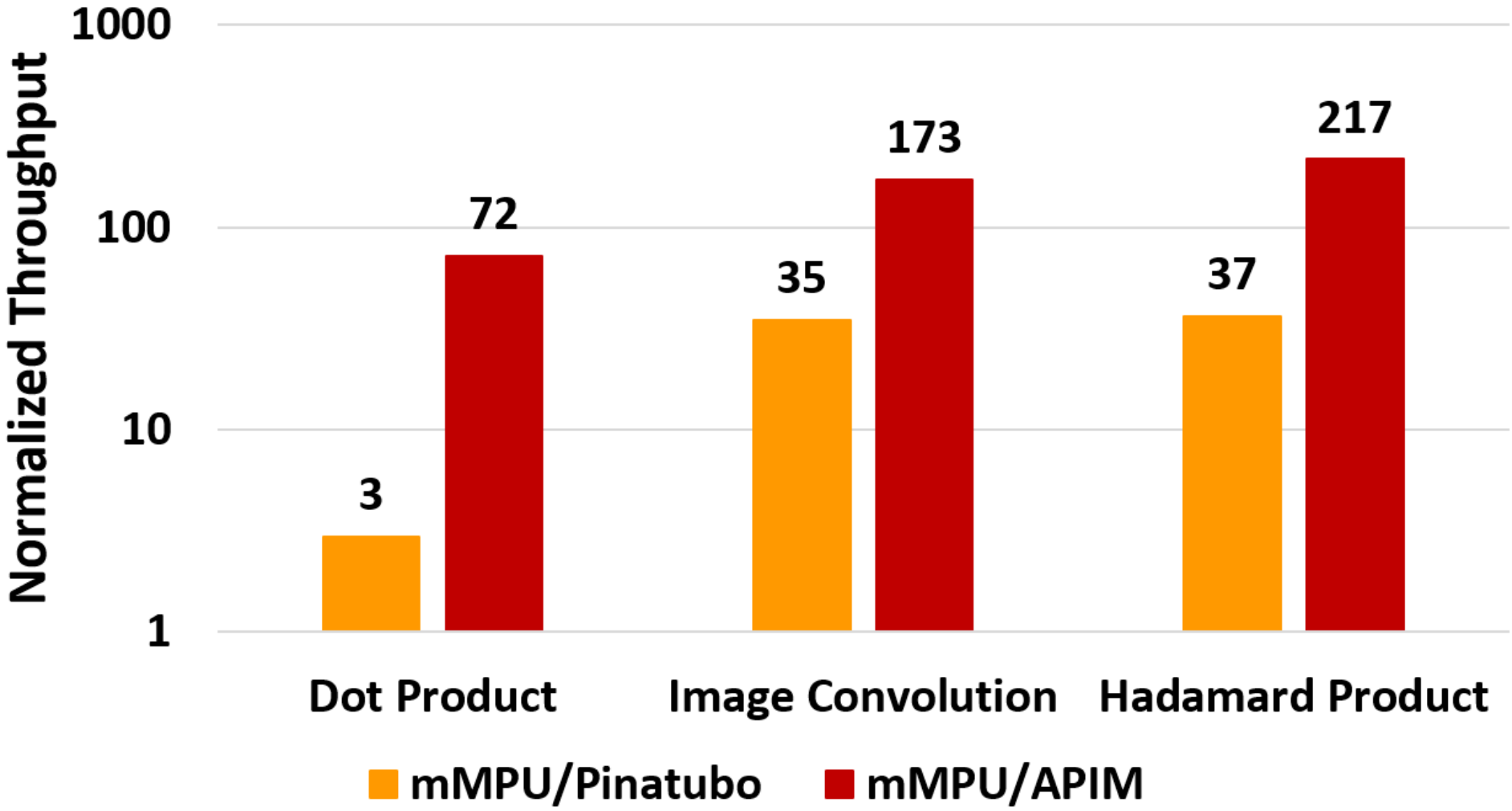
SIMPLER MAGIC:
Synthesis and In-memory Mapping of Logic Execution



1. **Synthesis tool - reducing the number of gates**
2. **Mapping tool - mapping the gates into the memristive memory**

R. Ben-Hur, R. Ronen, A. Haj-Ali, D. Bhattacharjee, A. Eliahu, and S. Kvatinsky, "SIMPLER MAGIC - Synthesis and Mapping of In-memory Logic Executed in a Single Row to Improve Throughput," IEEE TCAD (in press)

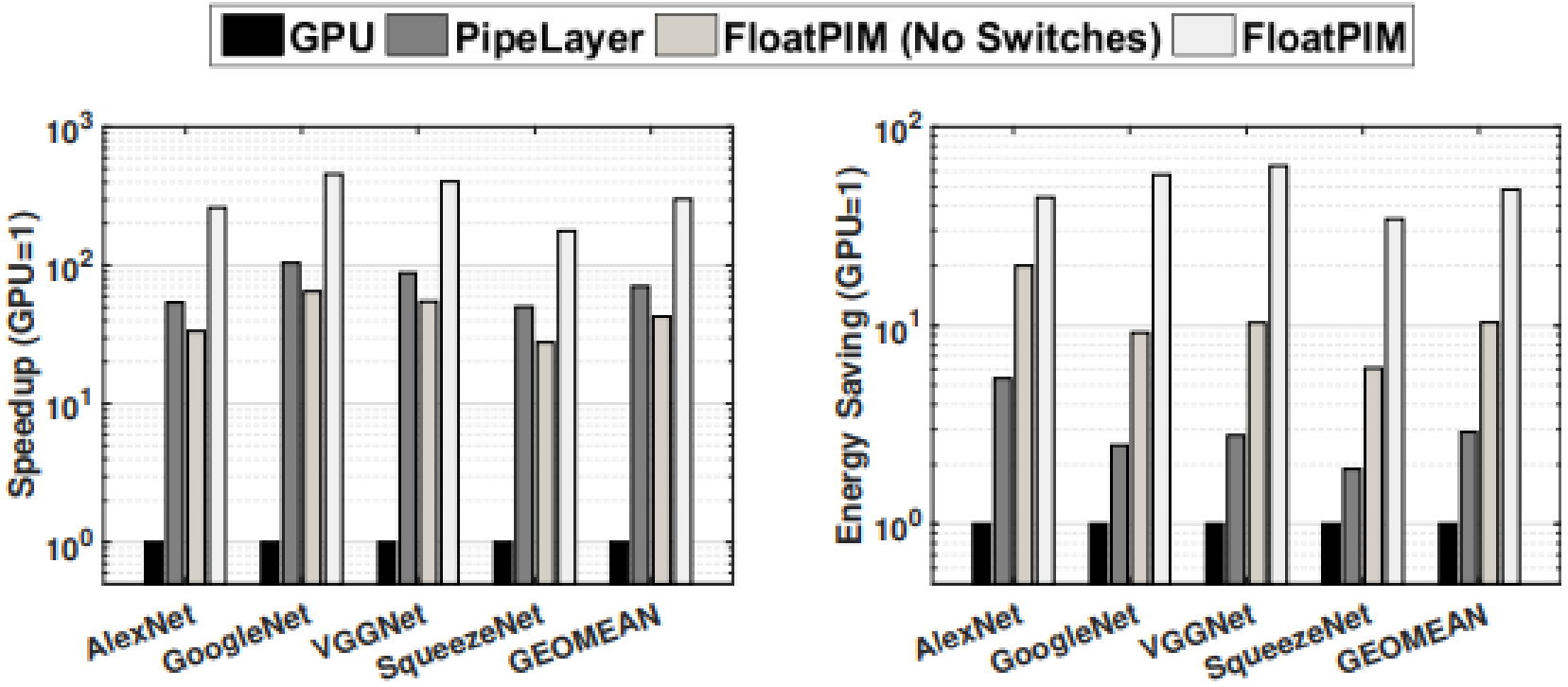
mMPU Performance Potential



A. Haj-Ali *et al.*, "IMAGING - In-Memory AlGorithms for Image processING," IEEE TCAS I, December 2018
A. Haj-Ali *et al.*, "Not in Name Alone: a Memristive Memory Processing Unit for Real In-Memory Processing," IEEE Micro, October 2018
S. Li *et al.* , "Pinatubo: A Processing-in-Memory Architecture for Bulk Bitwise Operations in Emerging Non-volatile Memories," DAC 2016
M. Imani *et al.*, "Ultra-Efficient Processing In-Memory for Data Intensive Applications," DAC 2017

mMPU for DNN

- FloatPIM, Imani et al., ISCA 2019
- Real-life issues (endurance, array size)

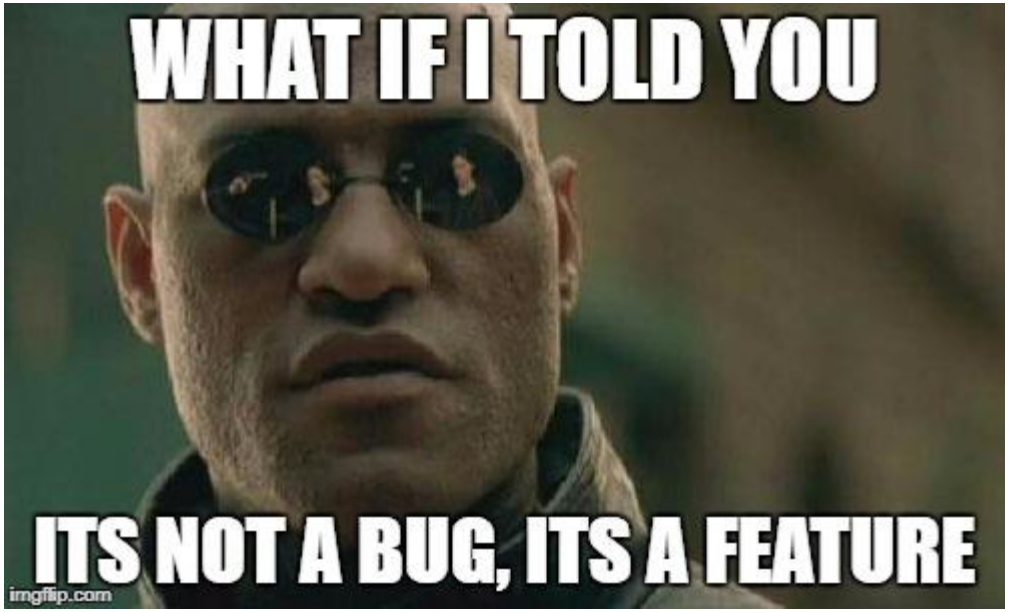


Agenda

- The need for Processing-in-Memory
- Memristors and Memristive Memory
- Memristive logic classification
- Out-of-memory logic
- Near-memory logic
- Real in-memory logic
- **Memristor for HW security**

Memristive Circuits for Hardware Security

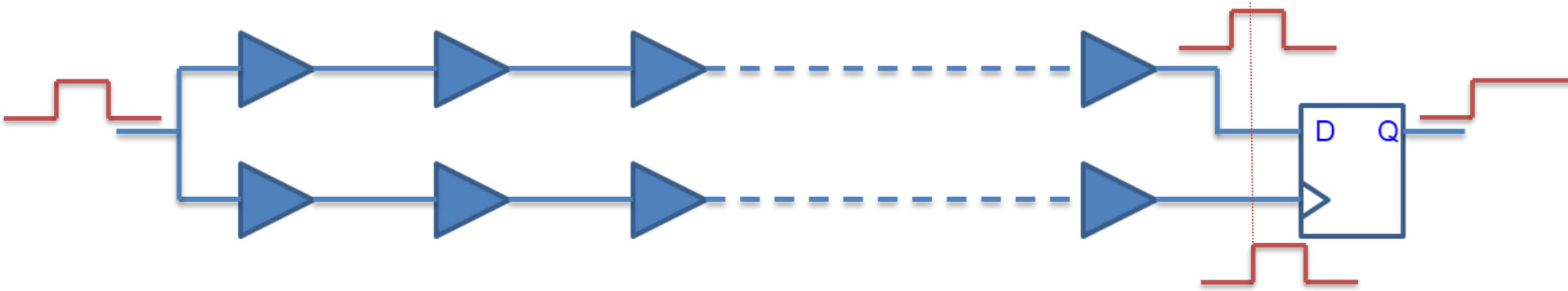
- Need for unique signature in security primitives
- Variations in memristors are usually a problem



Physical Unclonable Function (PUF)

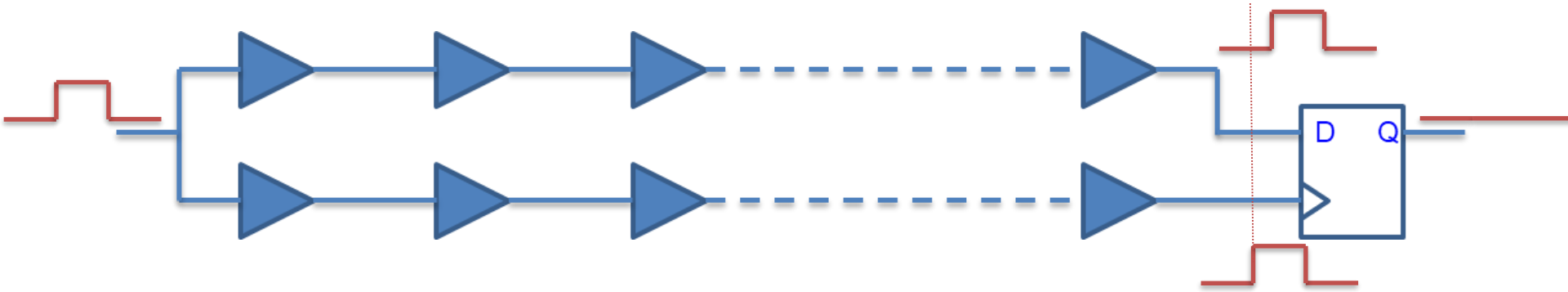
- Digital fingerprint of a semiconductor device

Arbiter PUF



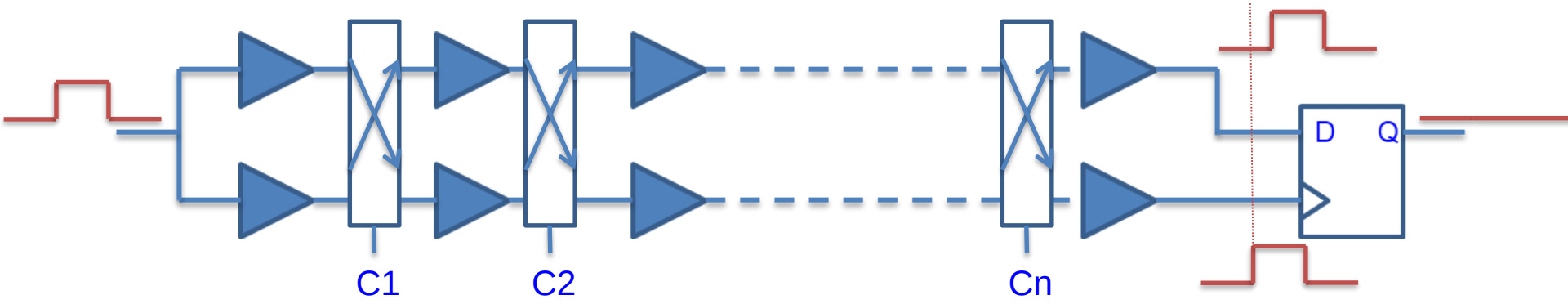
- Based on a mismatch between two identical delay paths
- The output of the flip-flop will be 0 or 1 depending on the winner path

Arbiter PUF



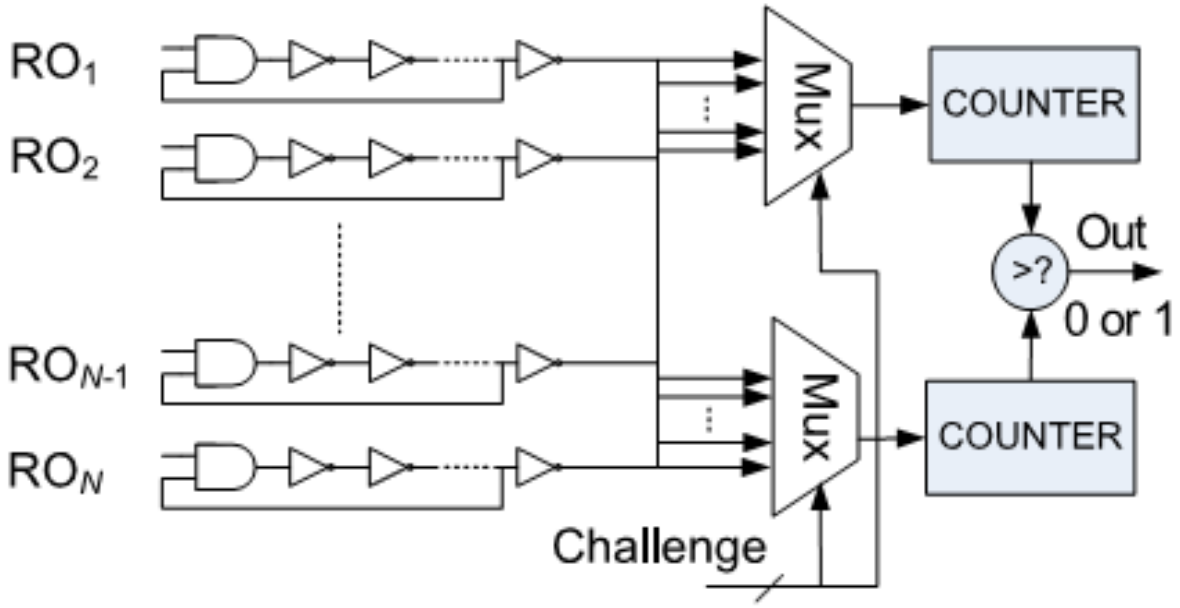
- Based on a mismatch between two identical delay paths
- The output of the flip-flop will be 0 or 1 depending on the winner path

Arbiter PUF



- Based on a mismatch between two identical delay paths
- The output of the flip-flop will be 0 or 1 depending on the winner path
- Bits of the challenge switch between the paths

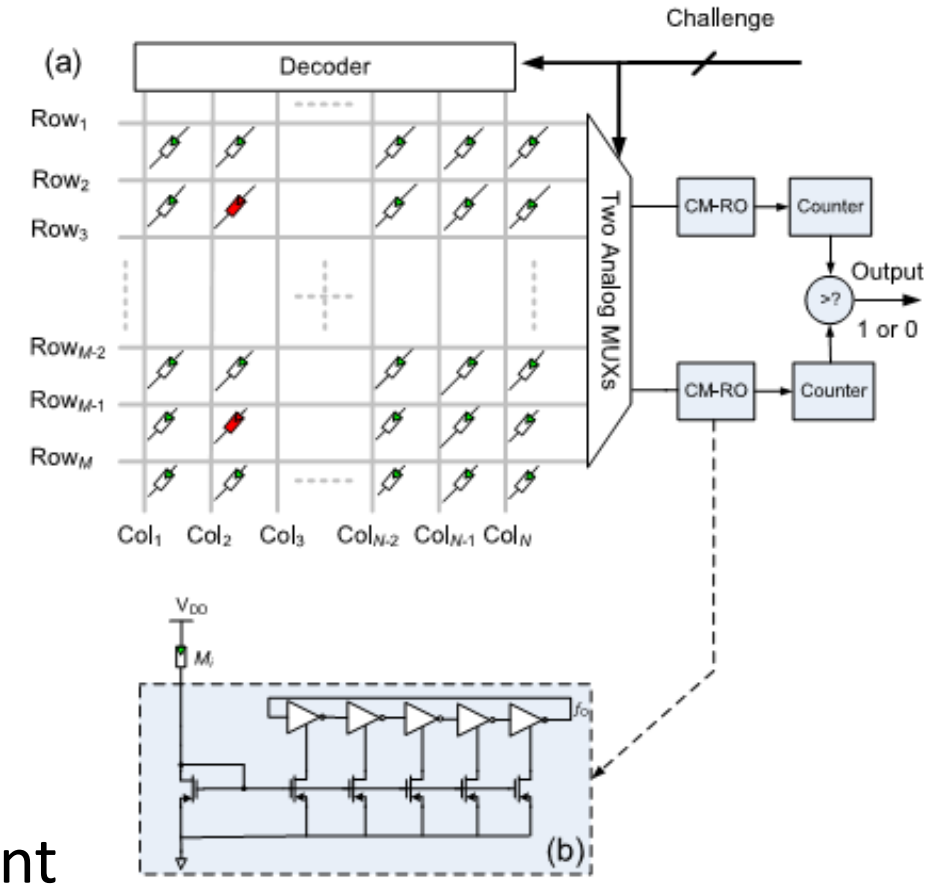
Ring Oscillator PUF (RO-PUF)



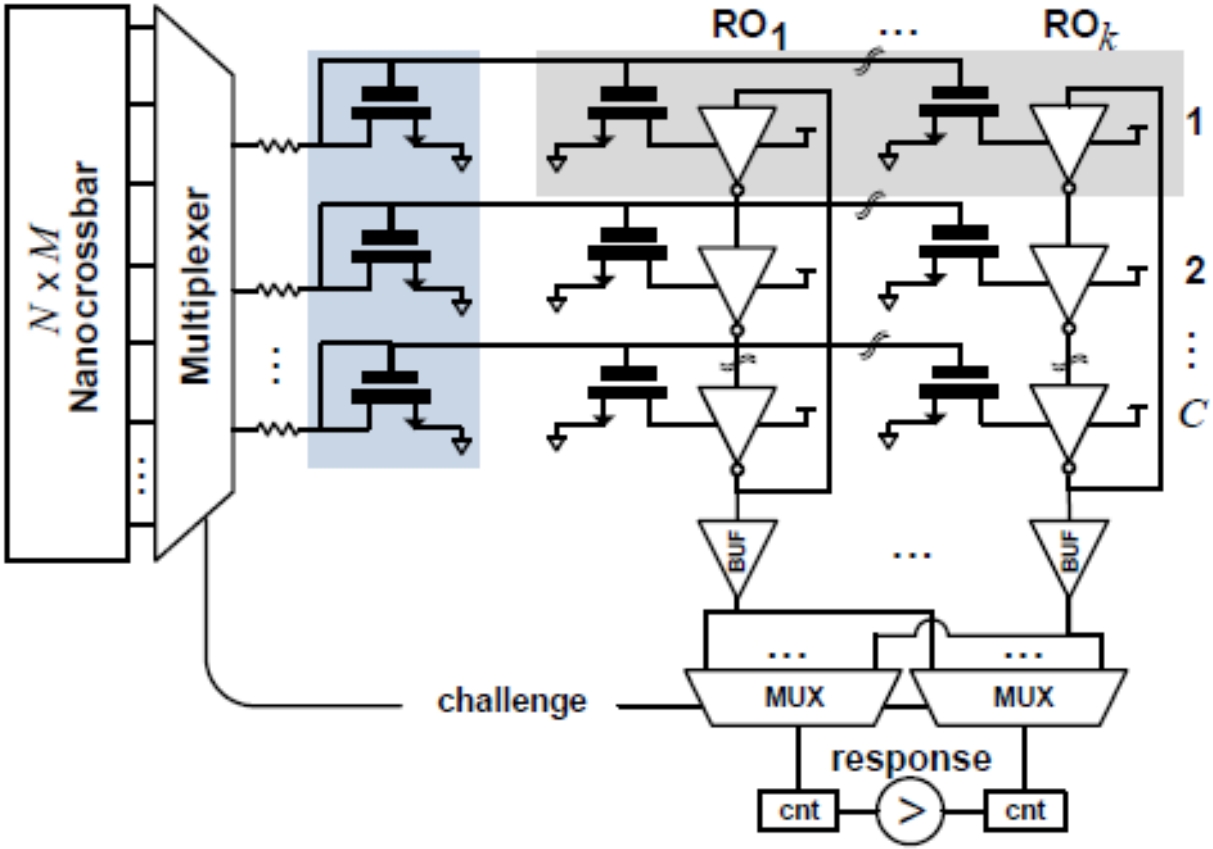
- Comparing frequencies of matched ring oscillators
- The challenge bits select the oscillators

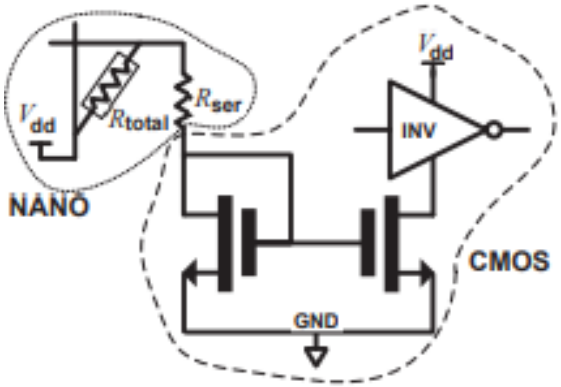
Memristors and PUF

- Memristors add more chaos
  - Harder to model
  - Better statistics
- Example: Memristive RO-PUF
  - Memristors bias the current controlled oscillators



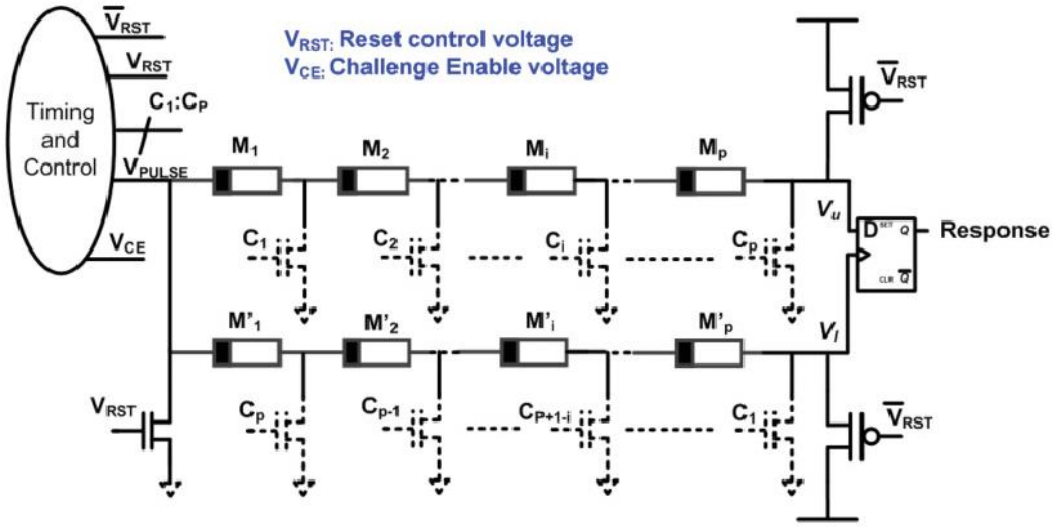
Memristive RO-PUF





O. Kavehei et al., "mrPUF: A Memristive Device Based Physical Unclonable Function," arXiv 2013

Memristor-Based Arbiter PUF

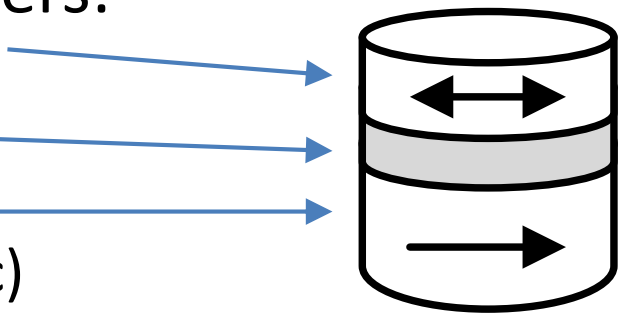


- The main idea – challenge based stage delays
- Advantages:
  - Low area
  - Good statistical randomness

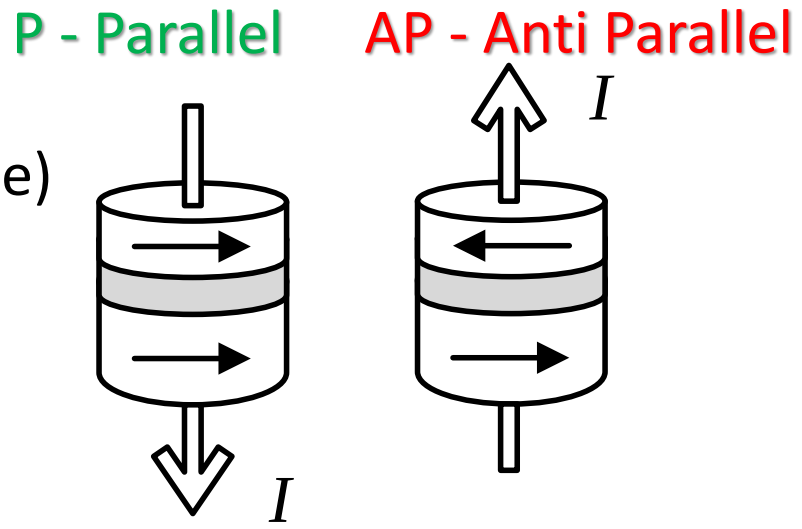
Mathew et al., "A Novel Memristor-Based Hardware Security Primitive", ACM Transactions on Embedded Computing Systems, 2015

Spin Torque Transfer Magnetic Tunnel Junction (STT-MTJ)

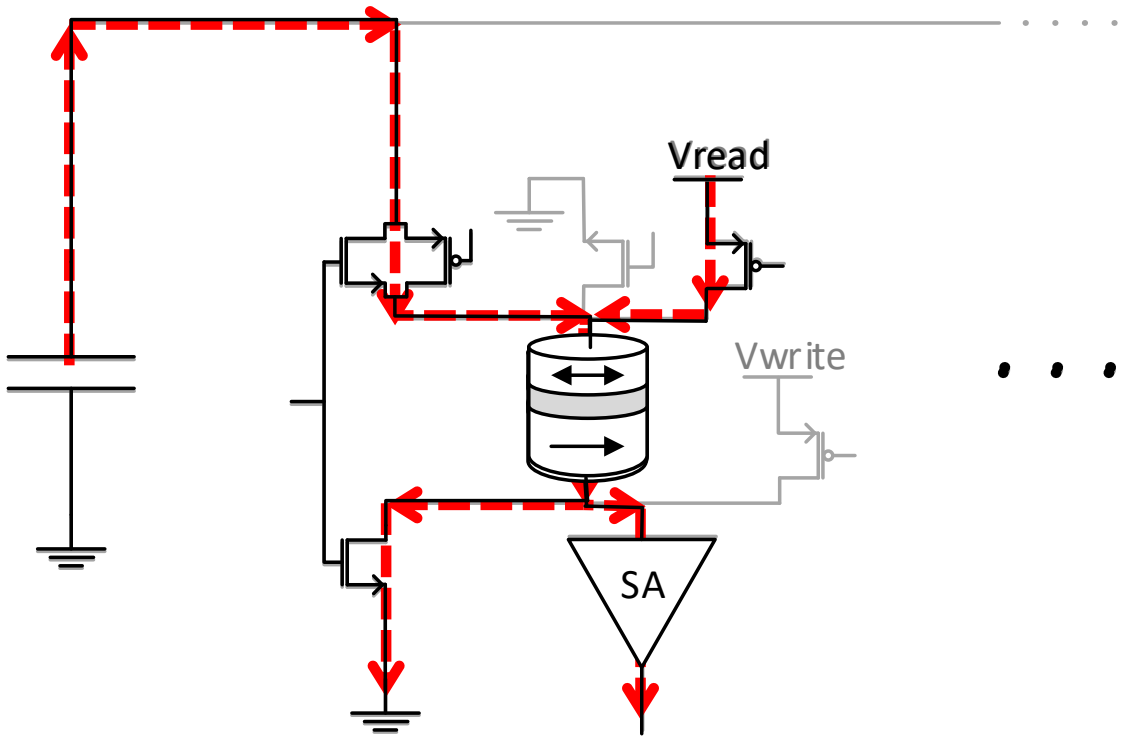
- Constructed of 3 layers:
  1. Free (ferromagnetic)
  2. Barrier
  3. Fixed (ferromagnetic)



- Two stable states:
  - Parallel (R_{on} resistance)
  - Anti-Parallel (R_{off} resistance)
  - $R_{on} < R_{off}$
- State switch using current
  - Stochastic switching time



Asynchronous TRNG Using STT-MTJ



B. Perach and S. Kvatinsky, DATE 2019

B. Perach and S. Kvatinsky, IEEE TVLSI, November 2019

Results

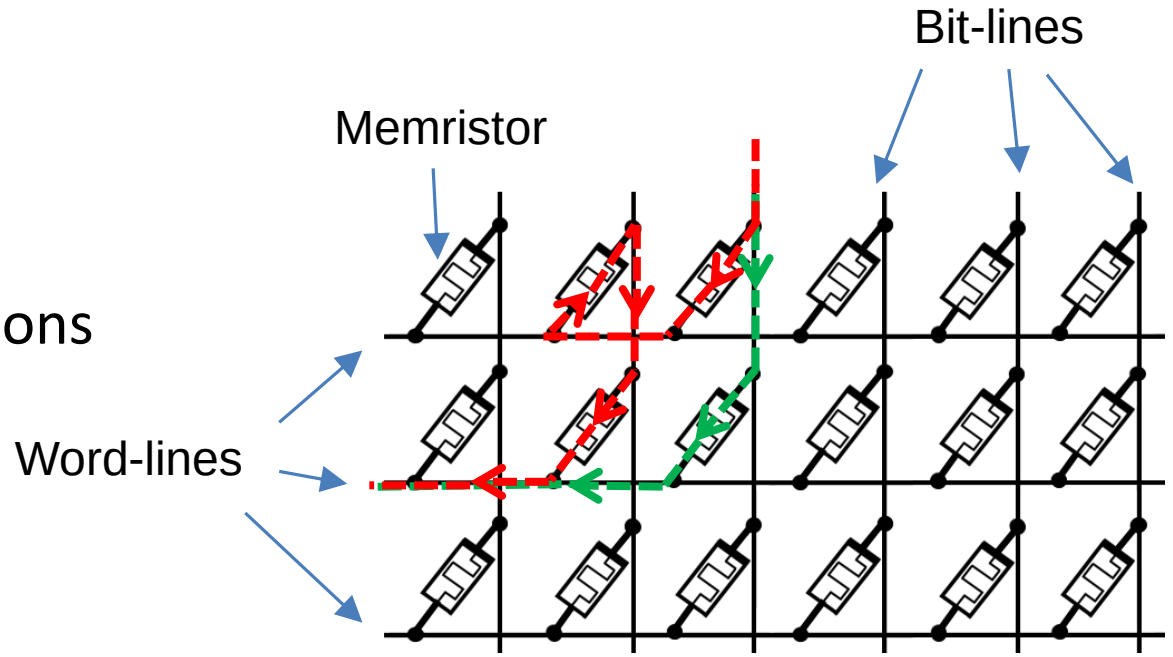
	Entropy rate [Mbps]	Energy per bit [pJ]	Area [μm^2]
Our Solution	100 - 128	6.0 - 7.7	405
[1]	23.2	23	375
[2]	2400	2.9	4004

Better than CMOS TRNG with the same area

[1] Yang *et al.*, ISSCC'14
[2] Srinivasan *et al.*, VLSIC'10

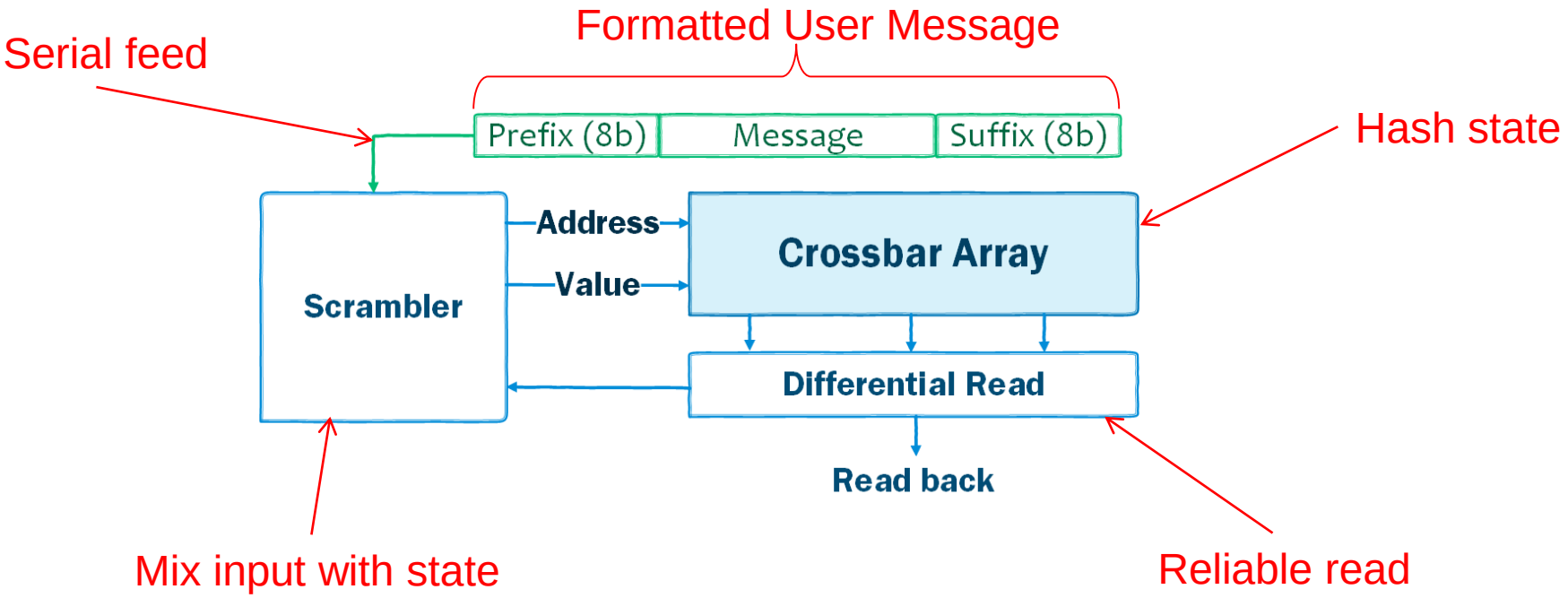
MemHash: Memristive Hardware Hash Function

- Crossbar array read & write disturbed
  - Sneak path currents
- Subjected to
  - Current state
  - Process variations



L. Azriel and S. Kvatinsky, HOST 2017

MemHash: Memristive Hardware Hash Function



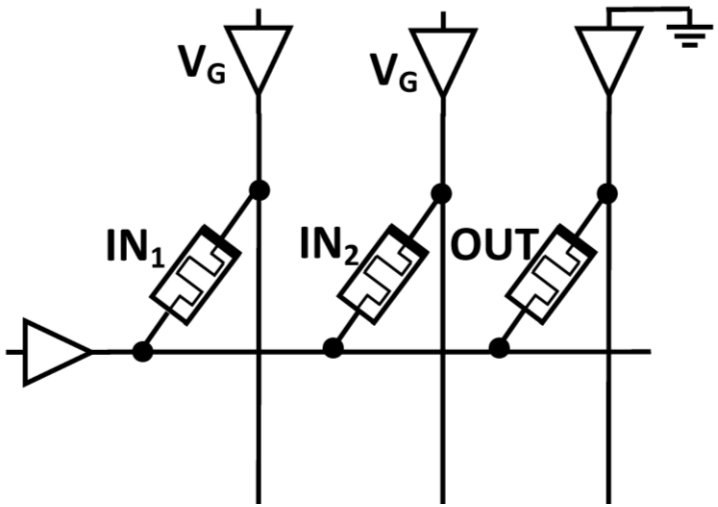
L. Azriel and S. Kvatinsky, HOST 2017

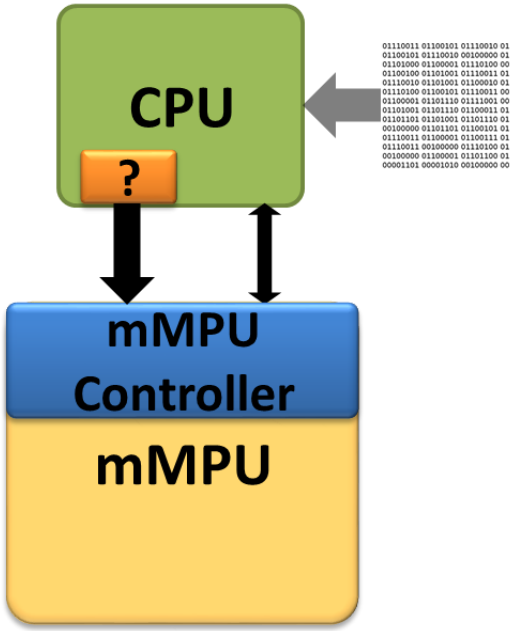
Agenda

- The need for Processing-in-Memory
- Memristors and Memristive Memory
- Memristive logic classification
- Out-of-memory logic
- Near-memory logic
  - Akers
  - Associative Processors
  - Neuromorphic
- Real in-memory logic
  - Stateful logic – IMPLY, MAGIC
  - Memristive Memory Processing Unit (mMPU)
- Memristors for HW security

Memristors for Computing - Summary

- Memristors enable non-von Neumann machines to overcome the memory wall and many more other applications (HW security and others)
- In-, Near-, Out- of memory computing
- Not necessarily one winner – the future is in heterogeneous systems





Thanks!

ASIC² ARCHITECTURES SYSTEMS INTELLIGENT COMPUTING INTEGRATED CIRCUITS

GENPRO
The Israeli RISC-V Consortium

Prime Minister's Office
National Cyber Bureau

Hiroshi Fujiwara Cyber Security Research Center

HiPer

משרד המדע והטכנולוגיה
Israel Ministry of Science and Technology

ISRAEL SCIENCE FOUNDATION

RBNI

TCE
technion computer engineering center

ICRI-CI
Intel Collaborative Research Institute
Computational Intelligence

Advanced Circuit Research Center ACRC

erc European Research Council

HUAWEI

רשות החדשנות
Israel Innovation Authority

CISCO

BSF
United States – Israel
Binational Science Foundation