Highly Efficient Neuromorphic Computing Systems with Emerging Nonvolatile Memories

Bonan Yan

Dept. Electrical & Computer Engineering

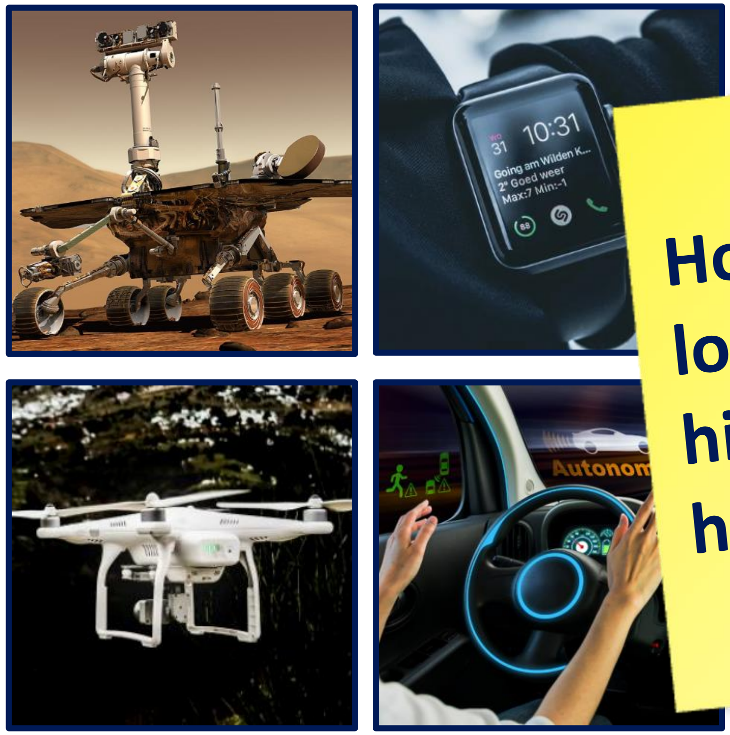
Duke University

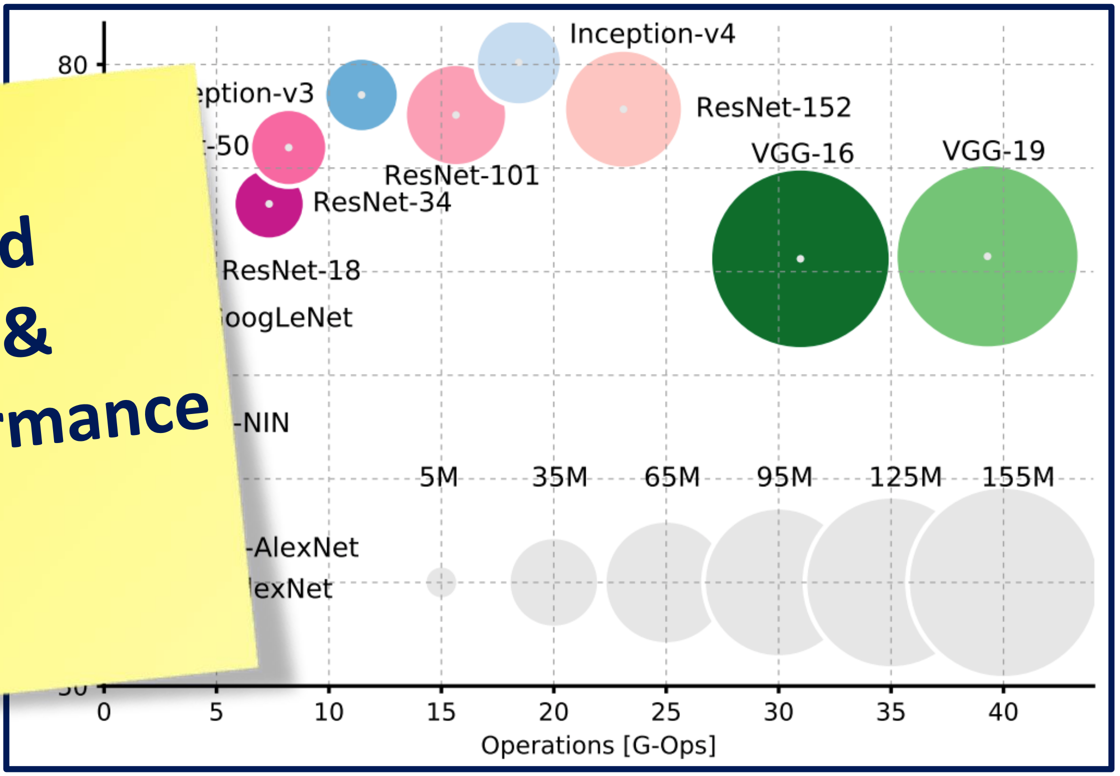
Slides available at: https://bonanyan.github.io/bn/

Efficiency Is The Key to Ubiquitous AI

Limited Power/Energy

Better Accuracy Comes From Larger Models



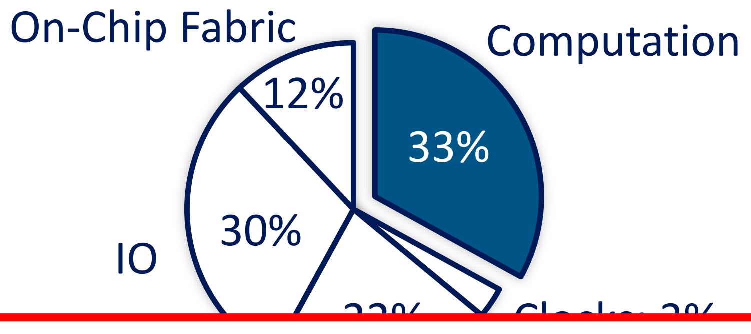


How to build low-power & high-performance hardware?

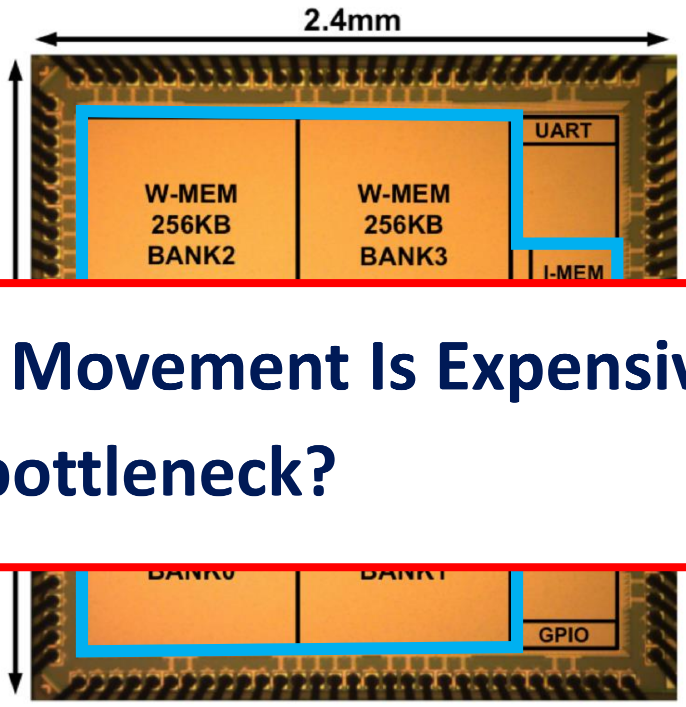
Source: IEEE Spectrum, arXiv:1605.07678

Overhead Dominated by Memories

Power



Area



🟥 : Function Unit

🟦 : On-chip Memory

🟪 : Control Module

- **Memory is the Bottleneck; Data Movement Is Expensive**
- **How to overcome the memory bottleneck?**

En

Int ADD	0.1
Float ADD	0.9
Register File	1
Int Multiply	3.1
Float Multiply	3.7
SRAM Cache	5
DRAM Memory	640

Source: AMD, Intel, [Whatmough, ISSCC 2017]

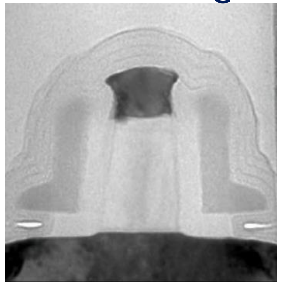
Duke

Specialized Hardware Enhances Efficiency

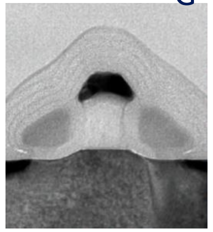
$$P = \alpha C V_{DD}^2 f$$

P: Power Dissipation
α : Activity Factor
C: Load Capacitance
V_{DD}: Power Supply
f : Clock Frequency

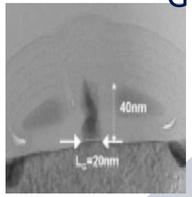
50nm L_G



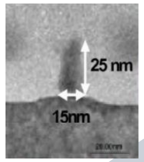
30nm L_G



20nm L_G



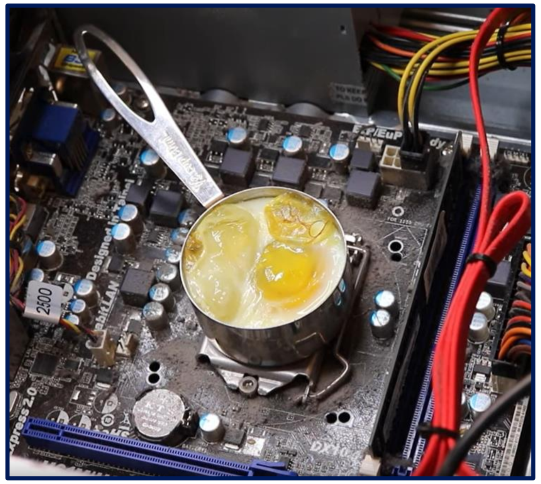
15nm L_G



MOSFET Scaling

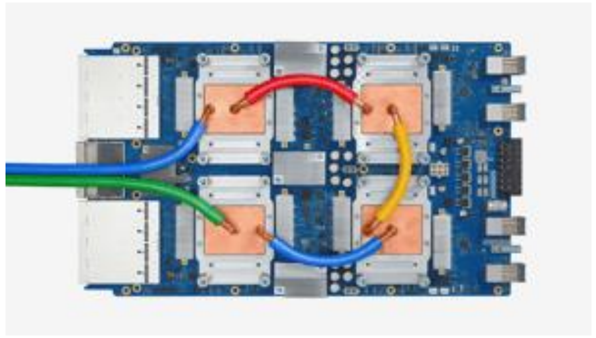
Transistor Size & Clock Frequency

Multicore to Manycore



Domain-Specific

groq
Cambricon 寒武纪
MYTHIC
habana An Intel Company
GRAPHCORE
arm
NVIDIA



Google TPU v3
~200Watts
(4 TPU to run AlphaGo, 300x less power consumption)

...

Duke

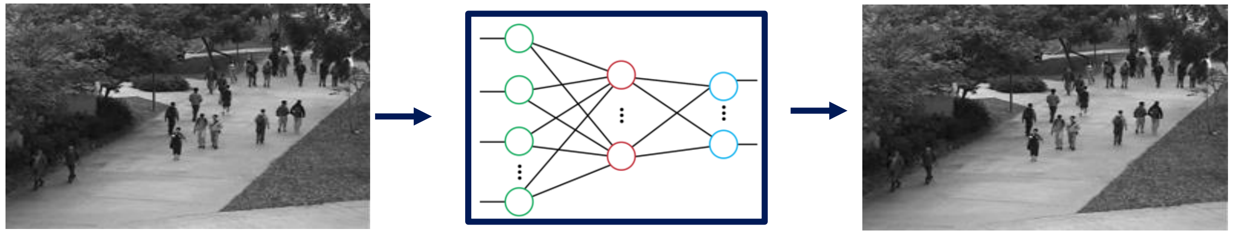
Source: Vasileska et al., JTCN, 2008, Google, https://www.youtube.com/watch?v=xL40UN5p6IY&t=673s

Uniqueness of Neural Network Execution

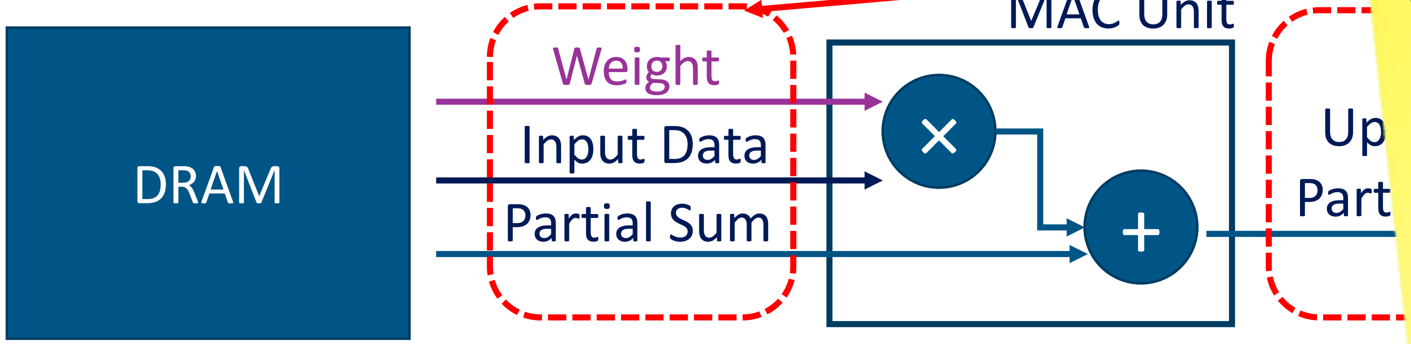
Multiply-Accumulate (MAC):

inputs weights outputs

$$\overset{\text{streaming}}{[x_1\ x_2\ \dots\ x_m]}\begin{bmatrix} w_{11} & \cdots & w_{1n}\\ \vdots & \text{stationary} & \vdots \\ w_{m1} & \cdots & w_{mn}\end{bmatrix} = [y_1\ y_2\ \dots\ y_n] \Rightarrow y_j = \sum_{i=1}^{m} x_i \cdot w_{ji}$$

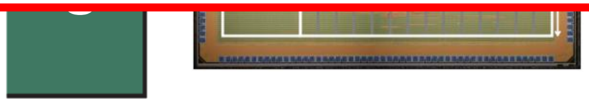
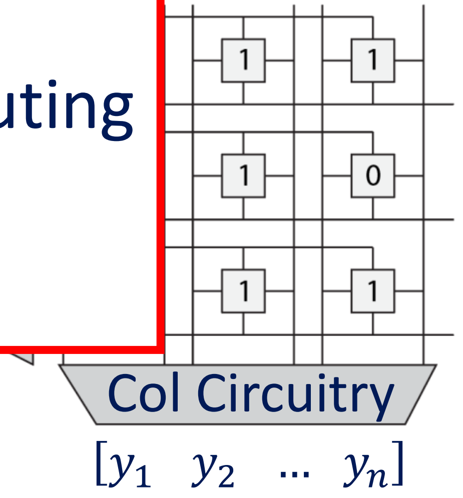


Execution:



Inference:
Inputs change;
weights stay.
How to fix weights
to where MAC Units
without moving?

Make Memory Access Less Expensive

	CMOS Accelerator (Planar Integration)	Near-Memory Computing	In-Memory Computing
Design Approach		High Bandwidth Memory	 1 1 1 0 1 1 Col Circuitry $[y_1 \quad y_2 \quad \dots \quad y_n]$
Energy to Access Weights	200×~6× of MAC Energy	55×~6× of MAC Energy	**0**

- Need new circuitry to support in-memory computing
- Rethink memory from the ground up!

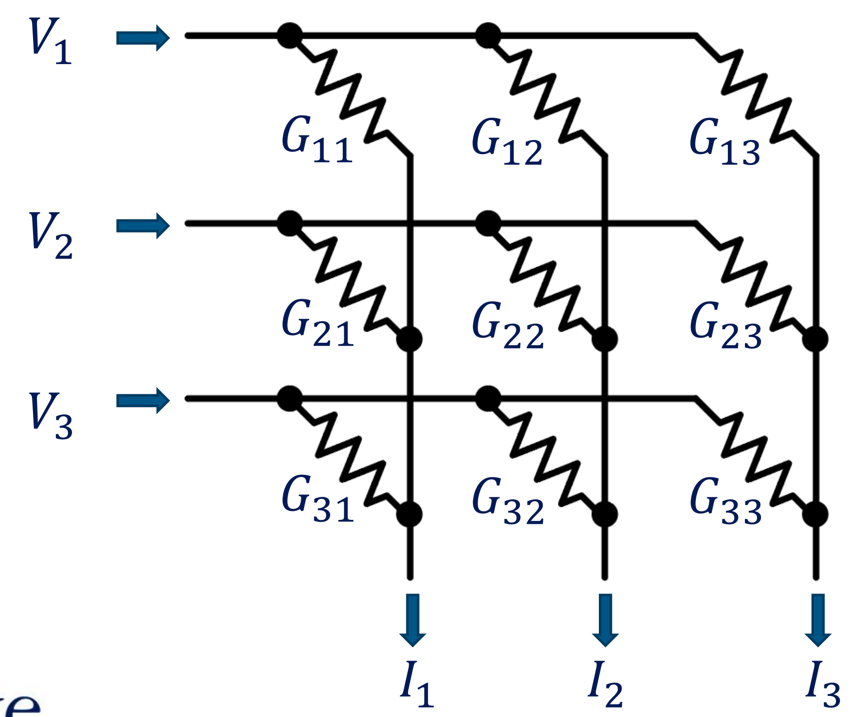
DRAM

On-Chip Buffer

Source: AMD, Y.-H. Chen, JSSCC,2017

Key Idea of In-Memory Computing

$$\begin{array}{ccc} \text{inputs} & \text{weights} & \text{outputs} \\ [V_1 \quad V_2 \quad V_3] & \begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{bmatrix} & = [I_1 \quad I_2 \quad I_3] \end{array}$$

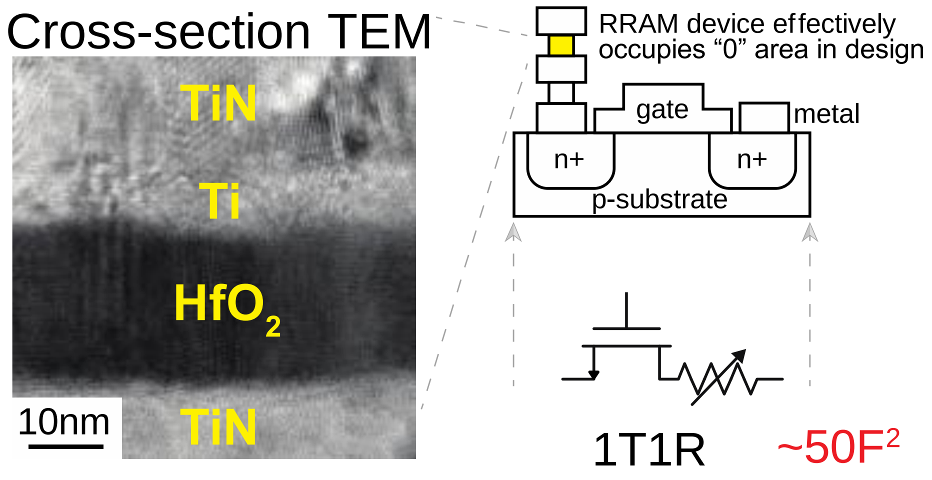


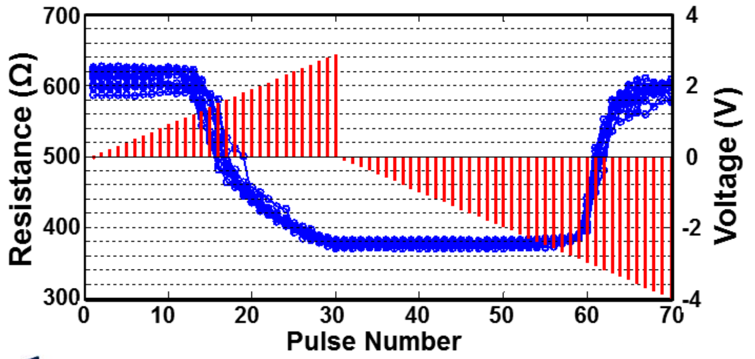
- Weight Matrix Stored as Conductance $\boldsymbol{G}$
- Rely on Analog Computation (Kirchhoff's Current Law) for “almost free”
  - Multiplication: $\mathrm{I} = V \cdot G$
  - Addition:$I^{column} = I_1^{row} + I_2^{row} + I_2^{row}$
- Ideal Nanoscale Devices for $\boldsymbol{G}$:
  - Programmable Conductance
  - Multi-Level Cell
  - Small Footprint/High Density
  - Compatible with Existing CMOS Process

Duke

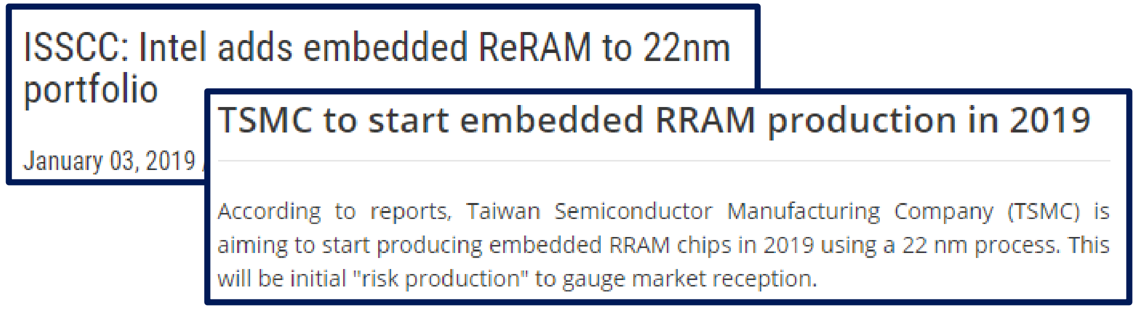
Memristors for In-Memory Computing

Also Called **R**esistive **R**andom **A**ccess **M**emory, RRAM or ReRAM





Programmable resistor w/ analog states



ISSCC: Intel adds embedded ReRAM to 22nm portfolio

January 03, 2019

TSMC to start embedded RRAM production in 2019

According to reports, Taiwan Semiconductor Manufacturing Company (TSMC) is aiming to start producing embedded RRAM chips in 2019 using a 22 nm process. This will be initial "risk production" to gauge market reception.

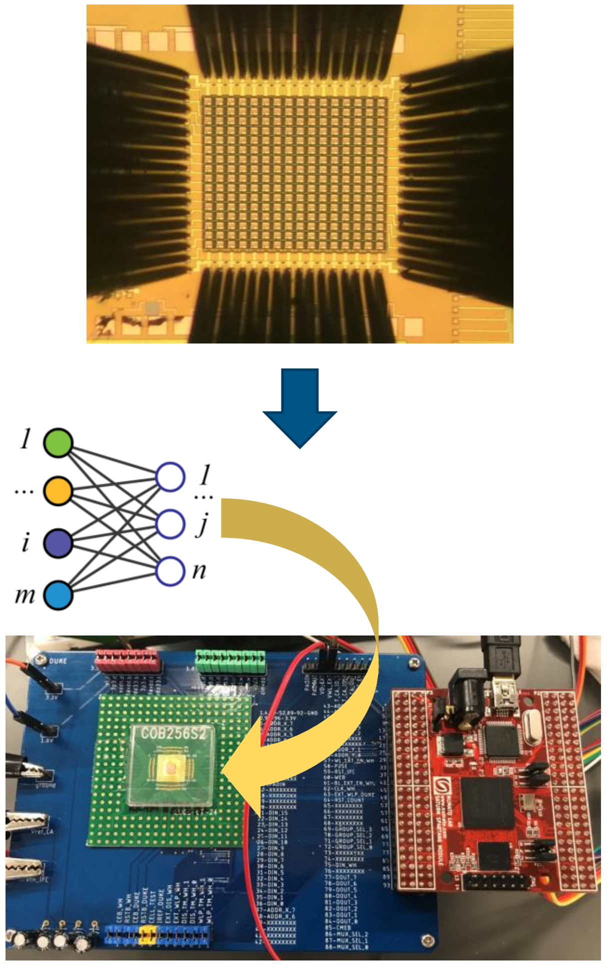
	Multi-Level Cell	Cell Area	R/W Speed
SRAM	×	large	Fast
DRAM	×	medium	Medium
1T1R	√	medium	Medium Fast
Flash	√	small	Slow

Duke

My work: Emerging Memory-Centric Design

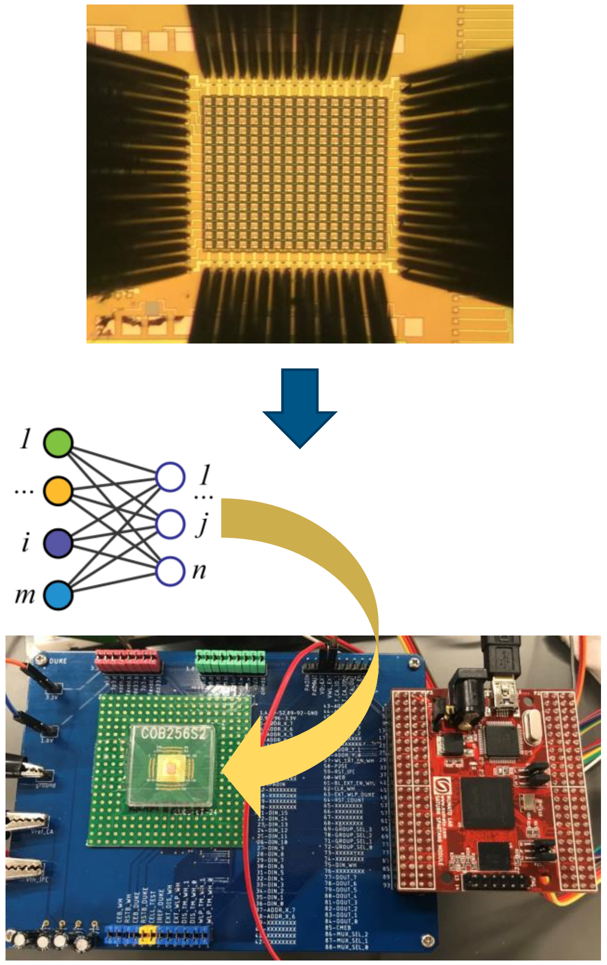
- **Circuits & Systems Implementation**
  - Spike-based Interface [DAC’15, DAC’18, DAC’20]
  - Implementation of Neural Networks [VLSI’19, DAC’20]

- **Tolerate/Exploit Non-ideal Behavior of Memristors**
  - Device Nonlinearity [ISCAS’16, IEDM’17, IEDM’19]
  - Read Disturbance [ICCAD’17], Hard Fault [ITC’19]



My work: Emerging Memory-Centric Design

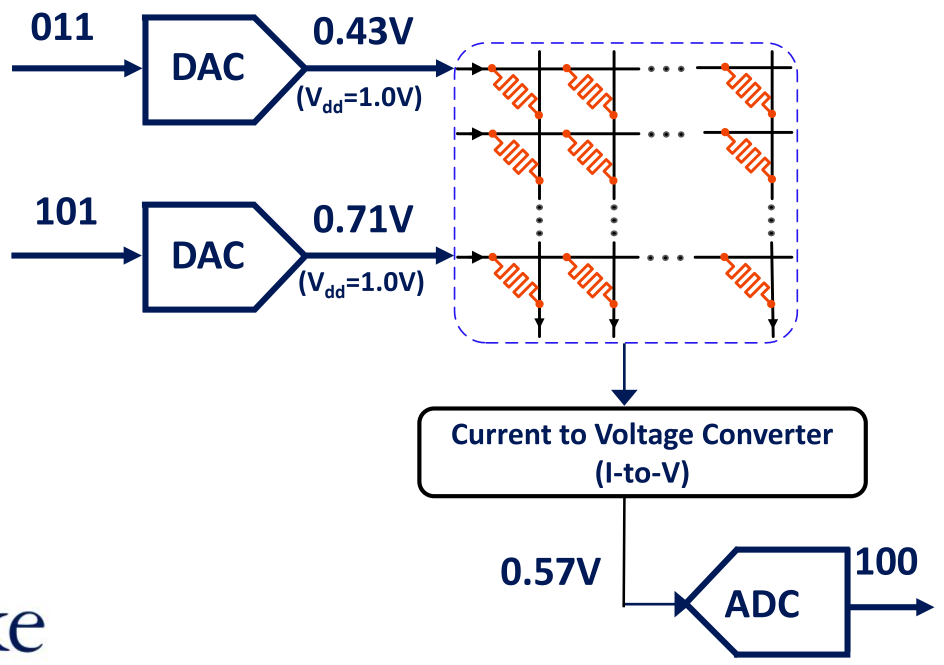
- **Circuits & Systems Implementation**
  - Spike-based Interface [DAC’15, DAC’18, DAC’20]
  - Implementation of Neural Networks [VLSI’19, DAC’20]
- **Tolerate/Exploit Non-ideal Behavior of Memristors**
  - Device Nonlinearity [ISCAS’16, IEDM’17, IEDM’19]
  - Read Disturbance [ICCAD’17], Hard Fault [ITC’19]



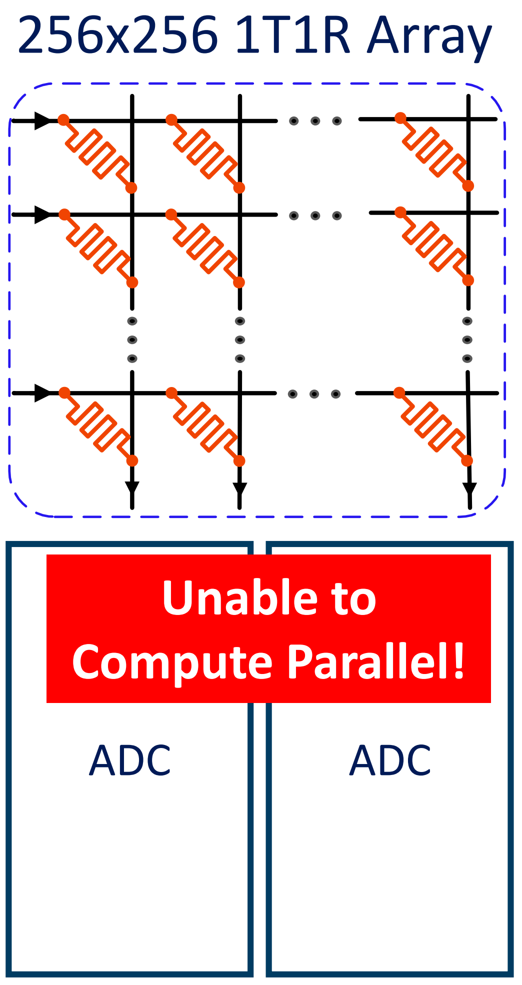
Conventional ADC is Too Large

The Level-based Design

- Compatible to existing signal processing
- High speed computation



Actual Layout:



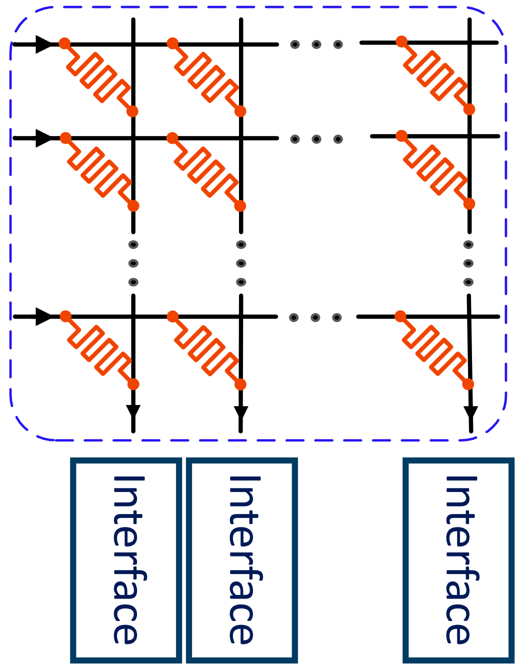
Based on 1.66MF^2 8bit ADC
by K. Ohhata (JSSC 2019)

Duke

My Approach: Spiking Interface Circuit

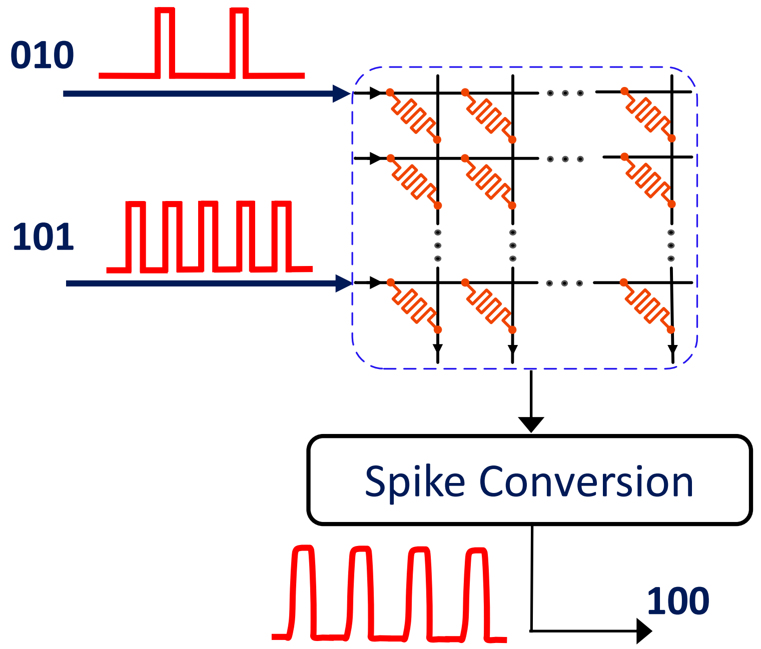
What Better Designs Look Like

- Compute Parallelly (Massive)
- Need Light-Weight Interface Circuitry



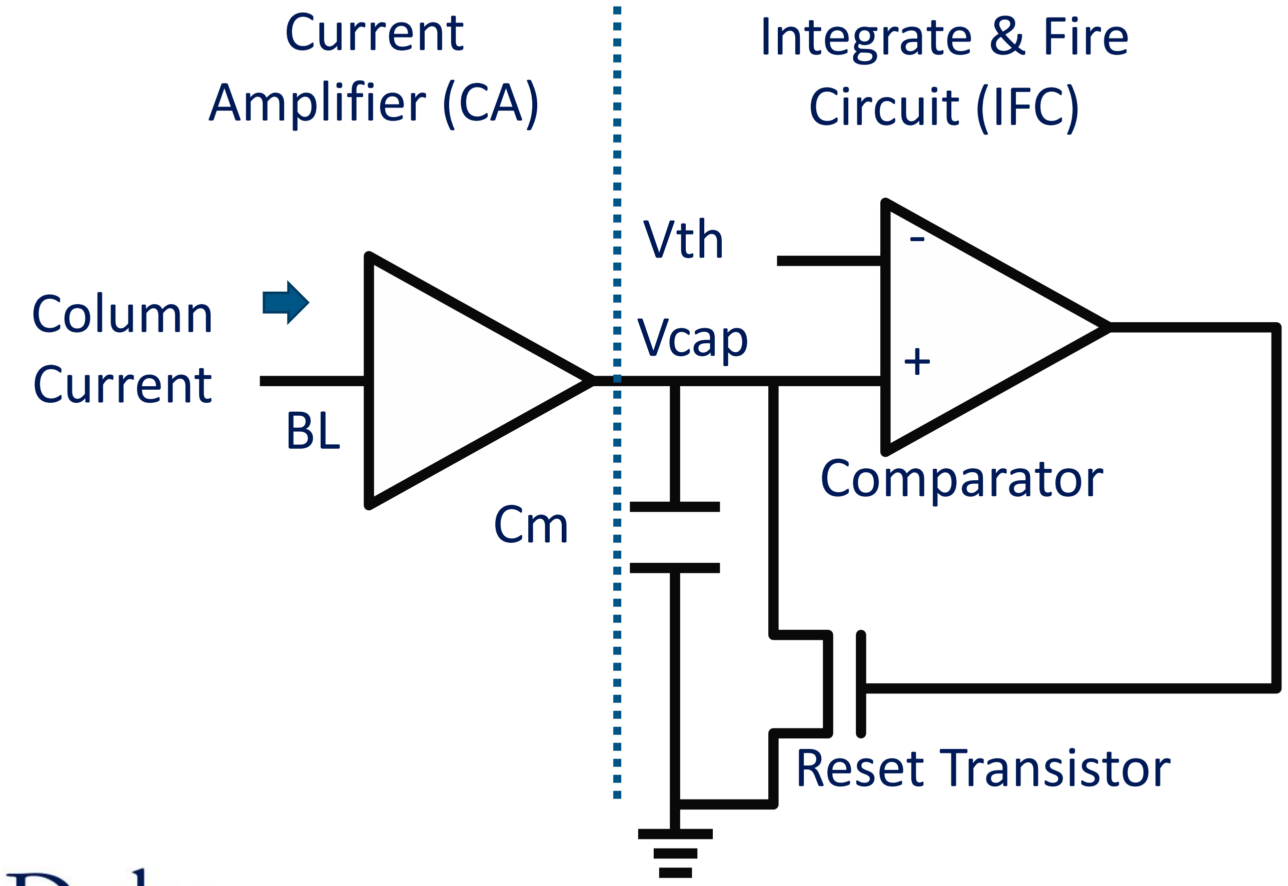
The Spike-based Design

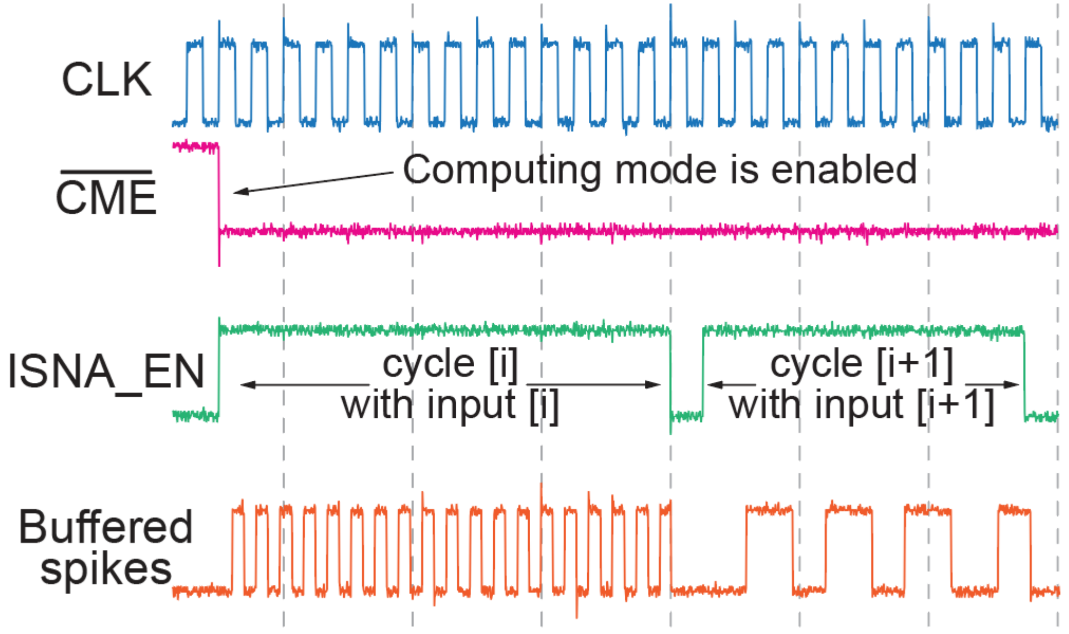
- Closer to biological system
- Extremely high power efficiency

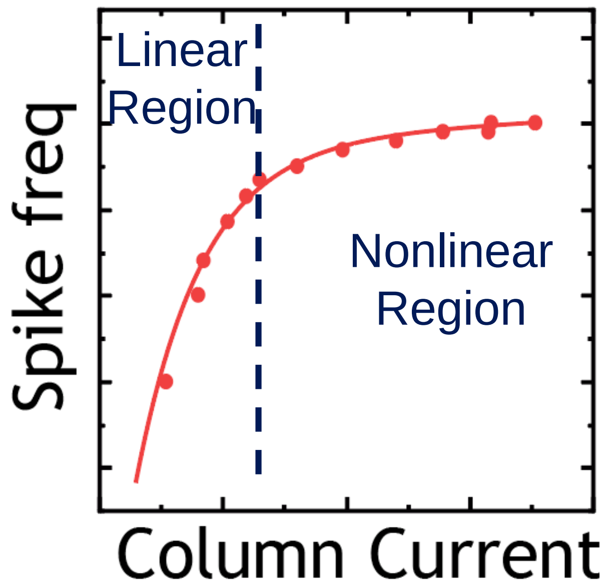


Duke

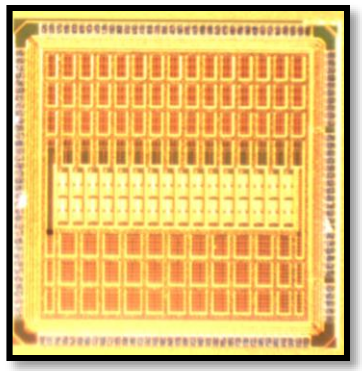
Spike Conversion







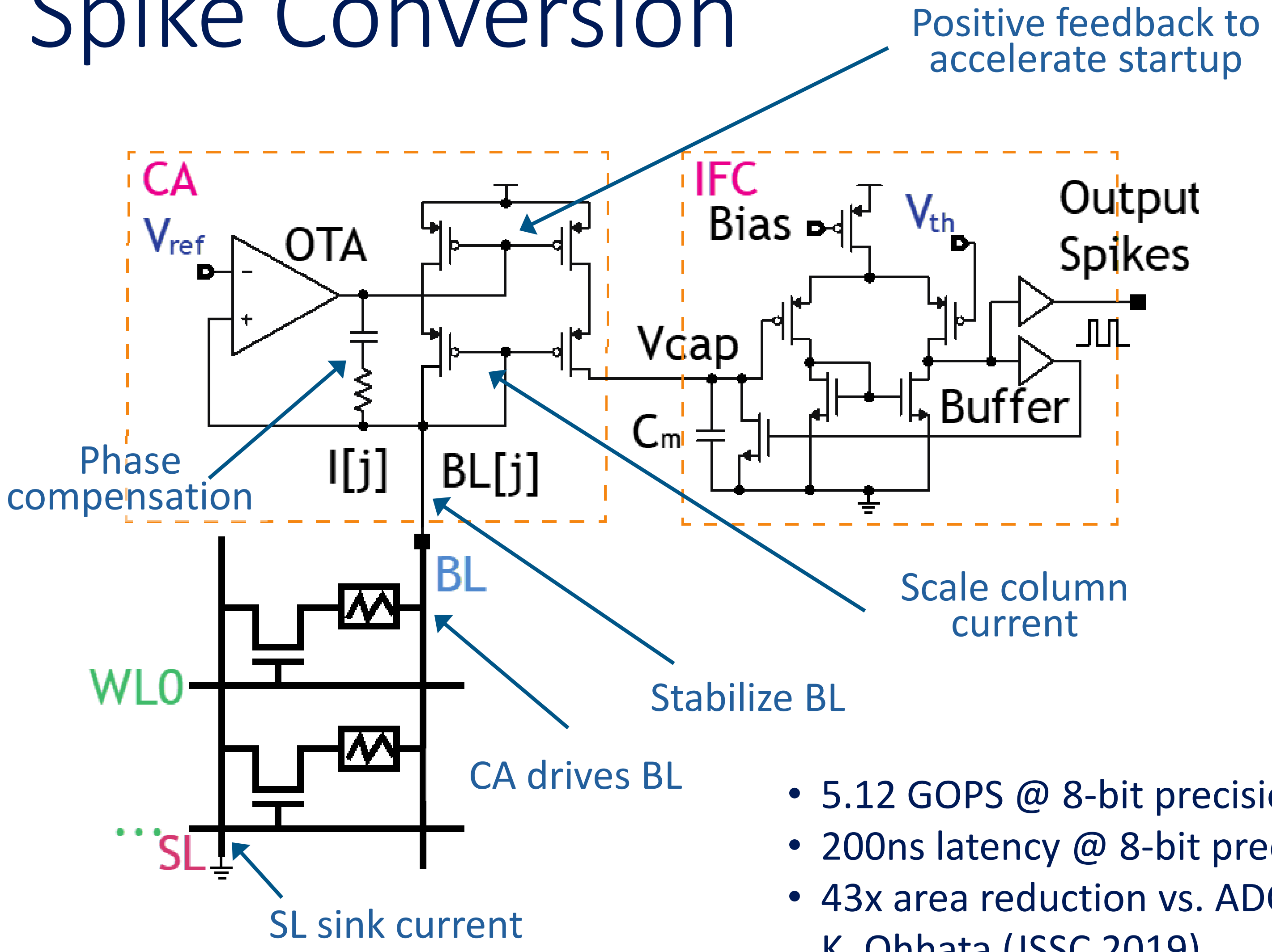
Spike Conversion Circuit & Controller 3152x3152 μm^2

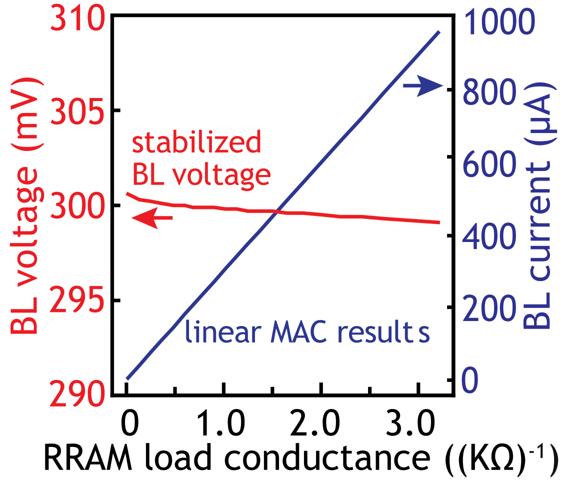


Duke

***B. Yan**, et al., ISCAS, 2016*

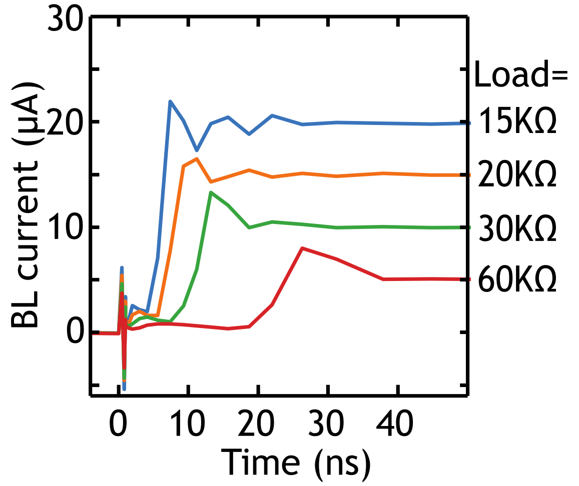
Spike Conversion





Tradeoff between large input current range and response speed

- 5.12 GOPS @ 8-bit precision
- 200ns latency @ 8-bit precision
- 43x area reduction vs. ADC by K. Ohhata (JSSC 2019)



Enlarge phase margin tolerating capacitor positive feedback

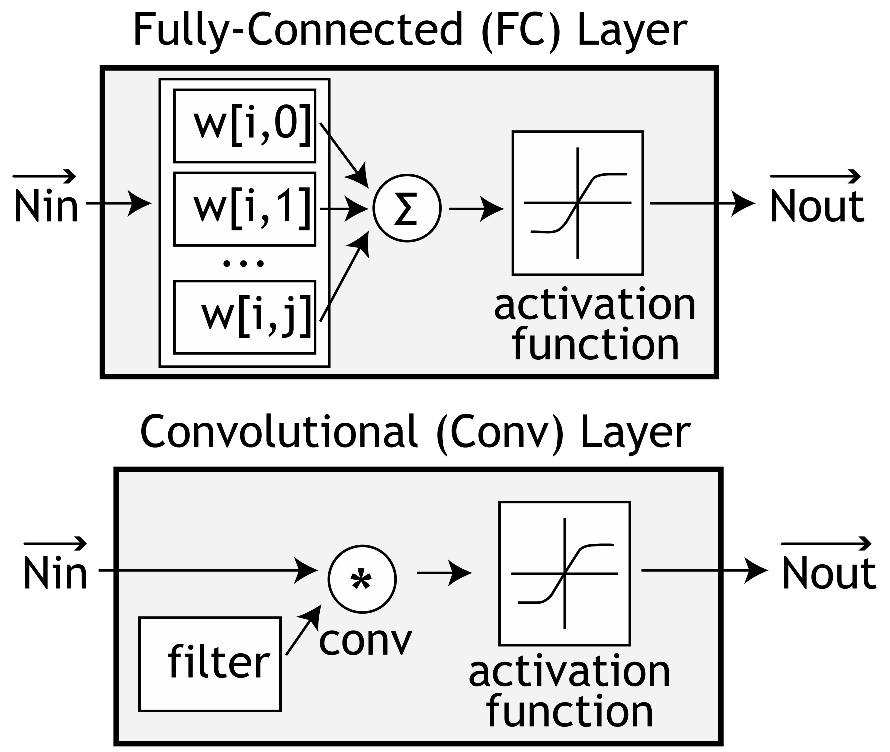
Duke

***B. Yan**, et al., IEEE VLSI Symp. on Tech. 2019*

How to Use Spiking-Based Design to Execute Neural Networks?

Duke

In Situ Nonlinear Activation (ISNA) Function



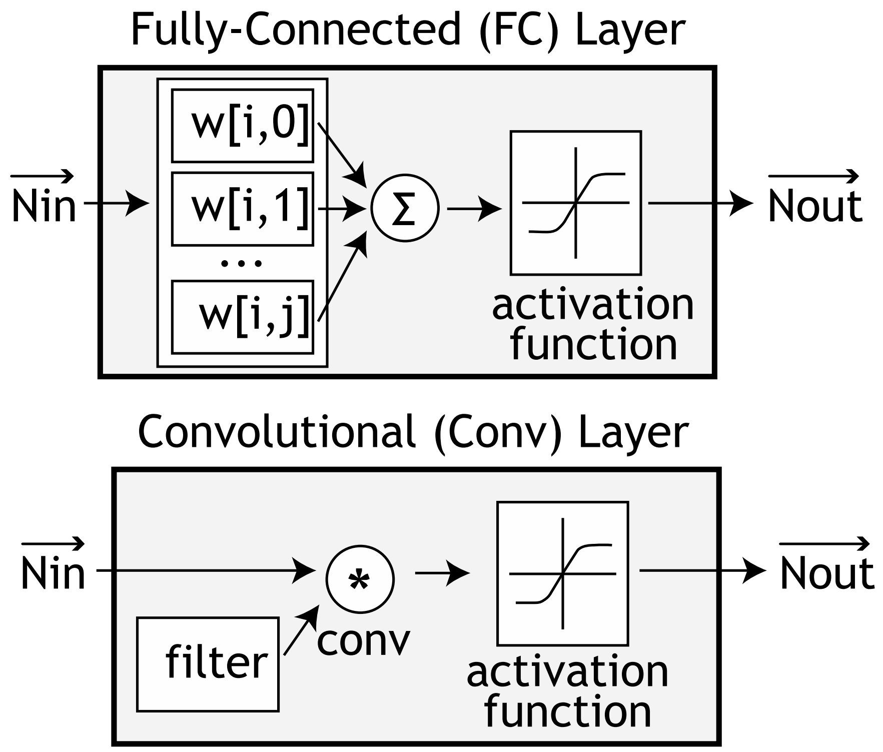
Single-layer Inference Operation:

- **D** • Step 1: Load data from buffer
- **A** • Step 2: Vector-matrix multiplication
- **D** • Step 3: Nonlinear activation function
- **D** • Step 4: Pooling
- **D** • Step 5: Store results to buffer

D : digital domain **A** : analog domain

Duke

In Situ Nonlinear Activation (ISNA) Function



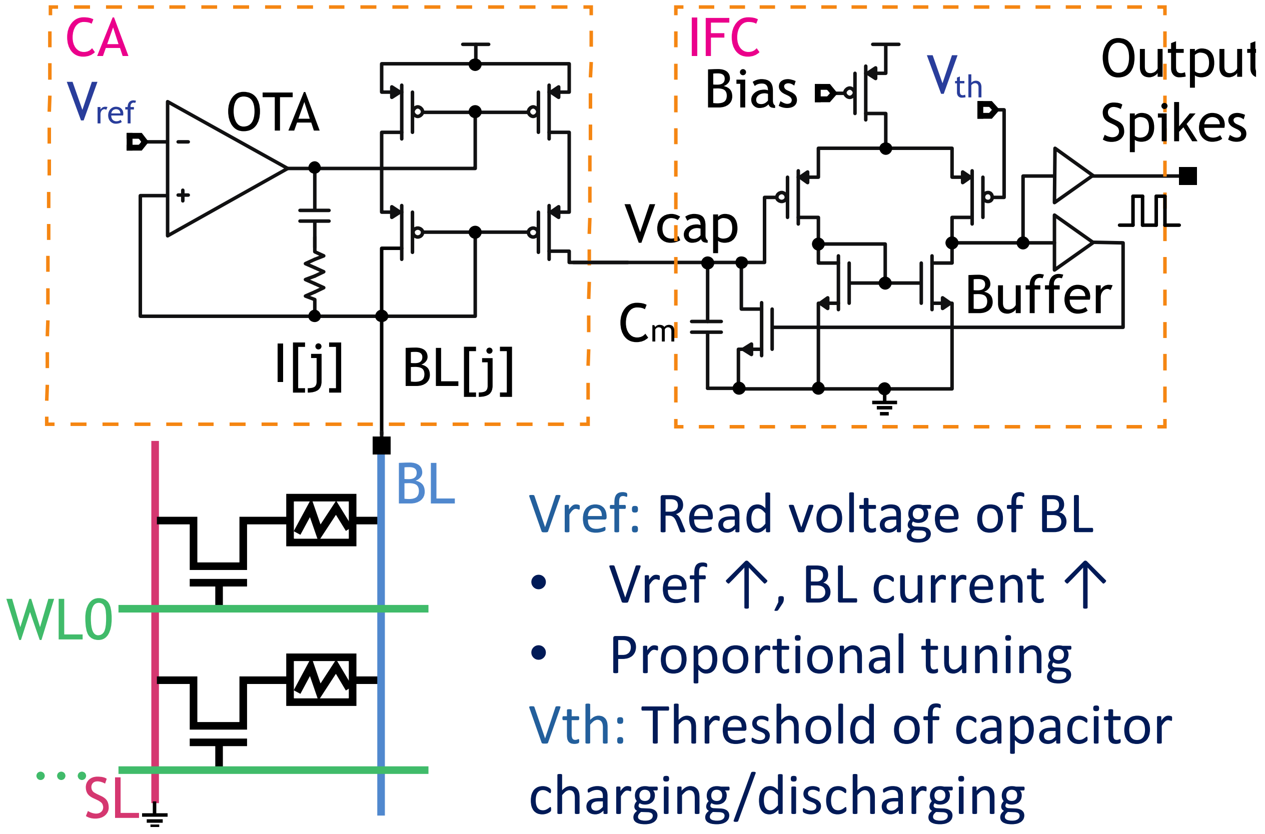
Single-layer Inference Operation:

- **D** • Step 1: Load data from buffer
- **A** • Step 2: Vector-matrix multiplication
- **A** • Step 3: Nonlinear activation function
- **D** • Step 4: Pooling
- **D** • Step 5: Store results to buffer

D : digital domain **A** : analog domain

Combine Step 2 & Step 3 to simplify PE operation:
Use linear + nonlinear regions

Adjust Activation Function

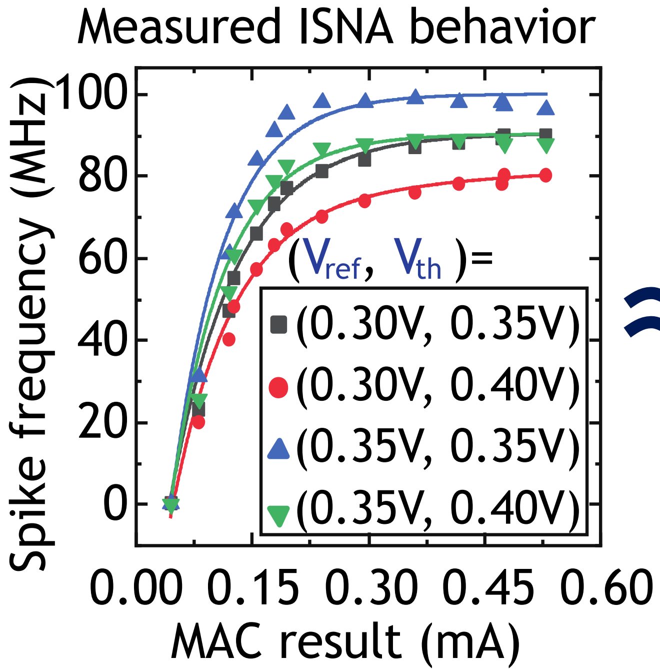


Vref: Read voltage of BL

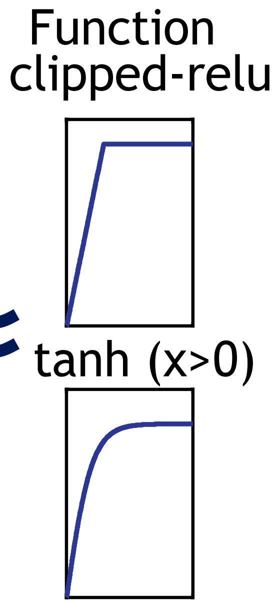
- Vref ↑, BL current ↑
- Proportional tuning

Vth: Threshold of capacitor charging/discharging

- Vth ↓, Charging/discharging ↑
- Distorted tuning

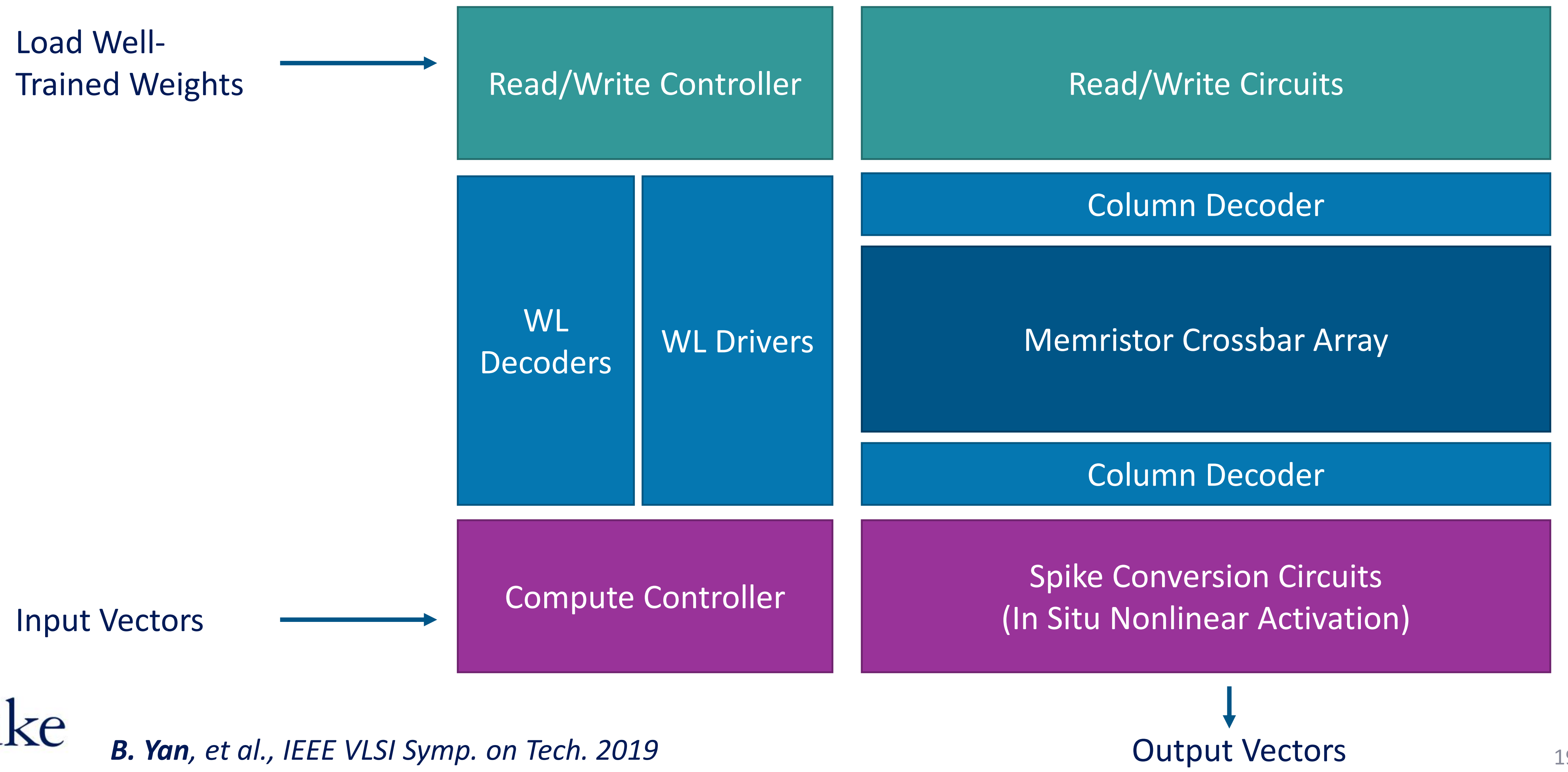


≈



Duke

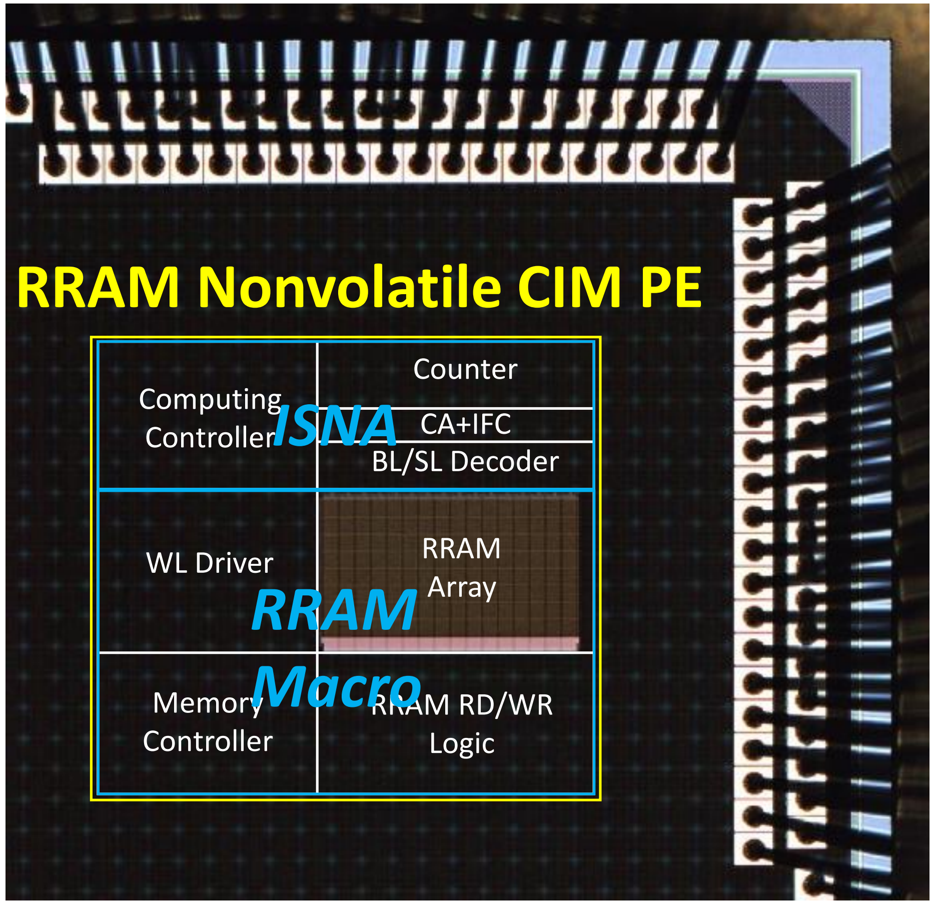
Chip Architecture



Duke

__B. Yan__, et al., IEEE VLSI Symp. on Tech. 2019

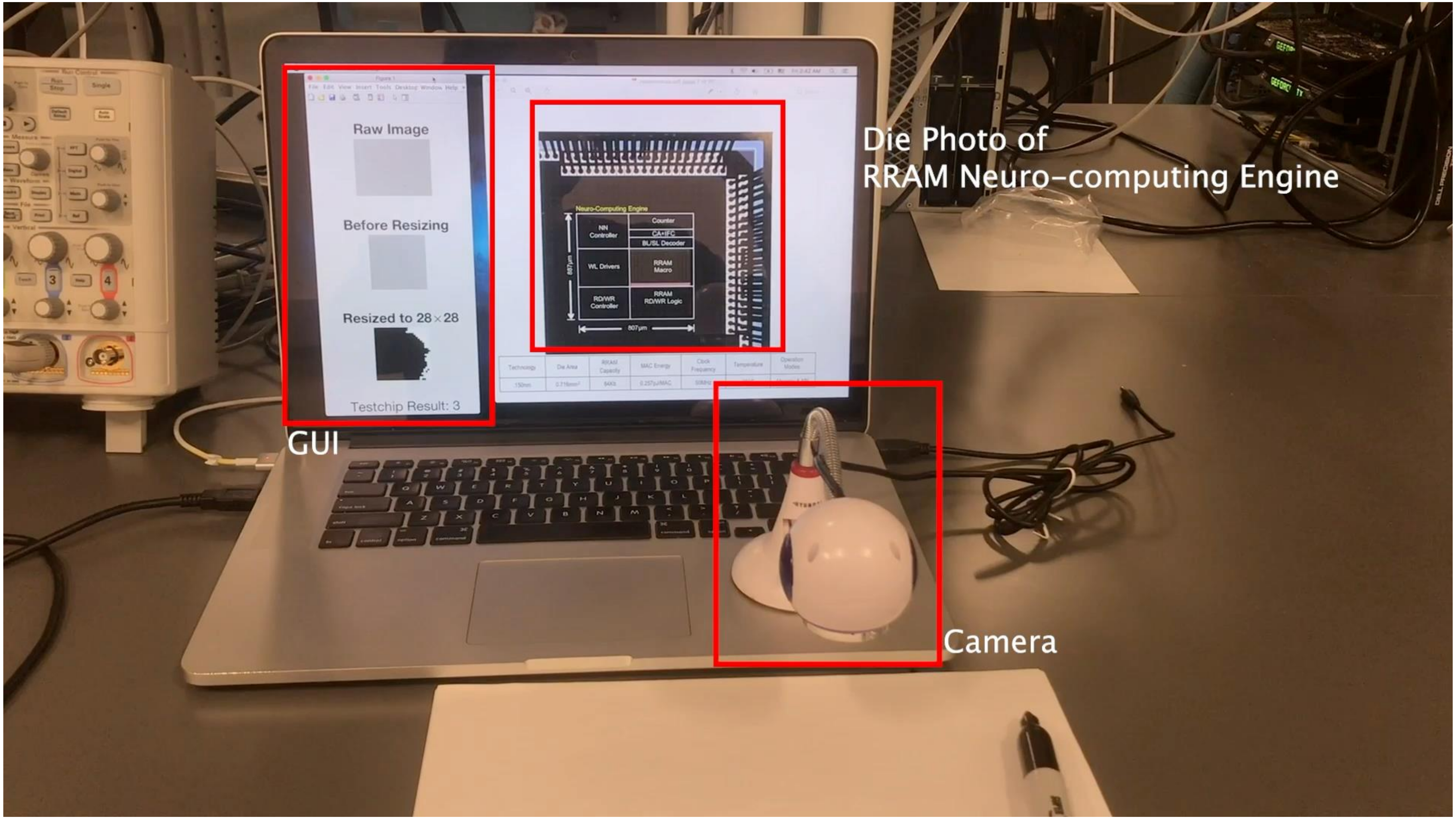
Chip Summary



Technology	150nm CMOS +HfO_x RRAM
Macro Capacity	64K (256×256)
Clock Frequency	50MHz
Energy Efficiency	0.257pJ/Mac
Average Power	1.52 mW
Layer-wise Latency	200ns
Real-time Benchmarks	3-layer perceptrons, LeNet-4, LetNet-5

Demo video online: https://bit.ly/AICHIP

Duke



Technology	Die Area	RRAM Capacity	MAC Energy	Clock Frequency	Temperature	Operation Modes
150nm	0.716mm²	64Kb	0.257pJ/MAC	50MHz	[illegible]	[illegible]

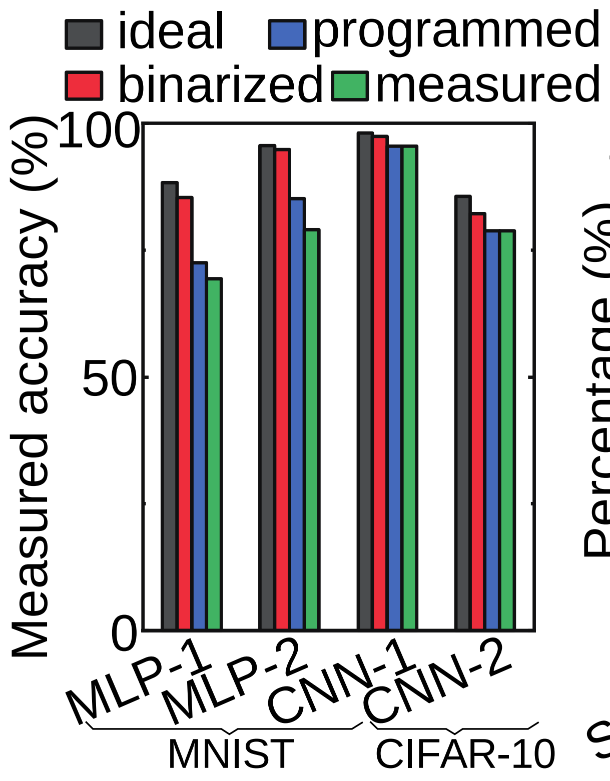
Evaluation: Measured Neural Network Results

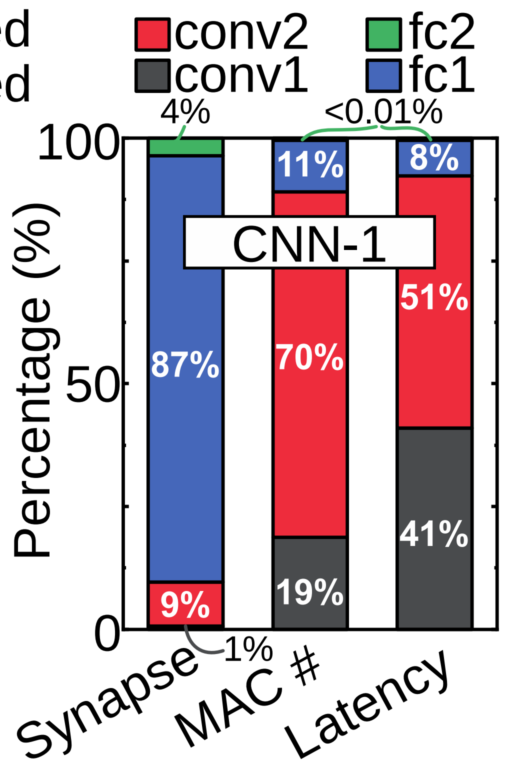
MNIST:

- MLP-1: Single-layer perceptron
- MLP-2: 2-layer perceptron
- CNN-1: 4-Layer LeNet

CIFAR-10:

- CNN-2: 5-Layer LeNet

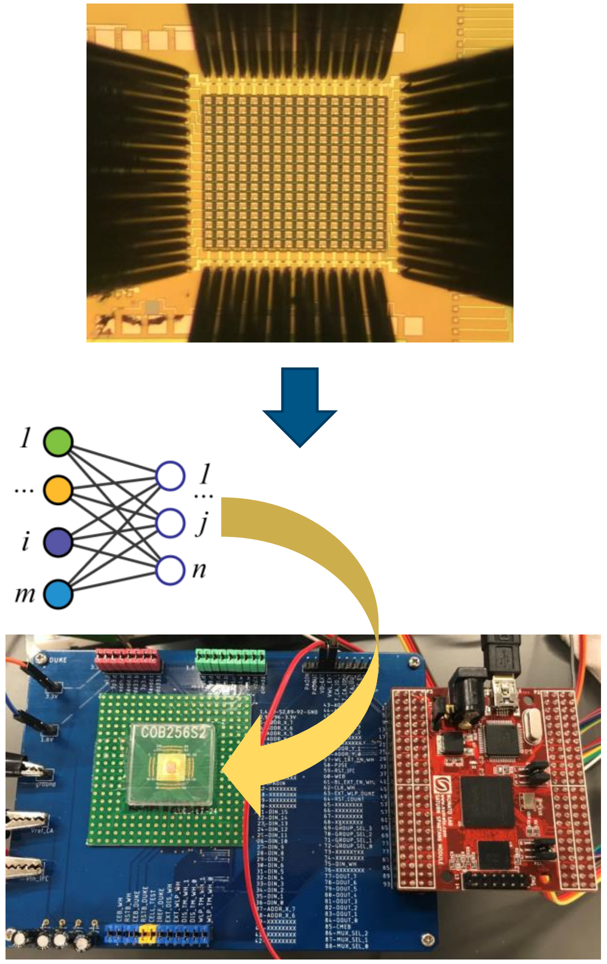




My work: Emerging Memory-Centric Design

- **Circuits & Systems Implementation**
  - Spike-based Interface [DAC'15, DAC'18, DAC'20]
  - Implementation of Neural Networks [VLSI'19, DAC'20]

- **Tolerate/Exploit Non-ideal Behavior of Memristors**
  - Device Nonlinearity [ISCAS'16, IEDM'17, IEDM'19]
  - Read Disturbance [ICCAD'17], Hard Faults [ITC'19]

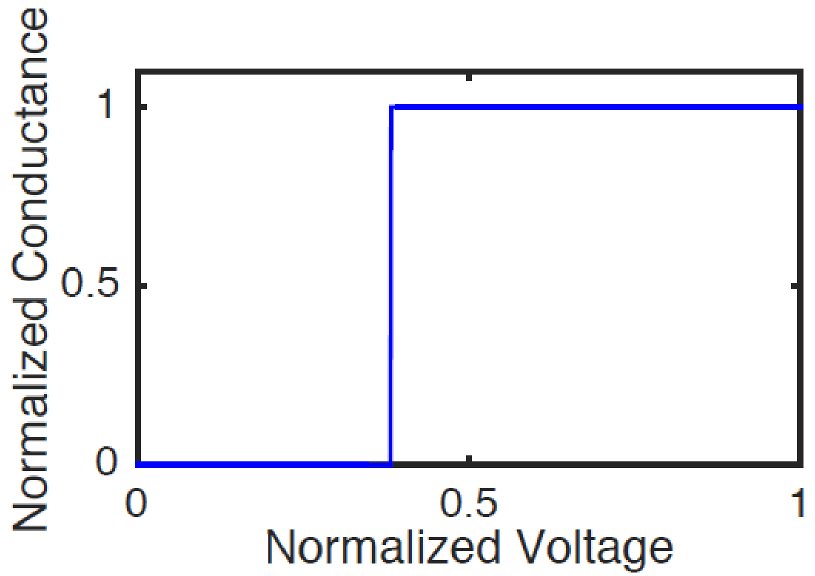


Duke

Non-Ideal Memristor - I

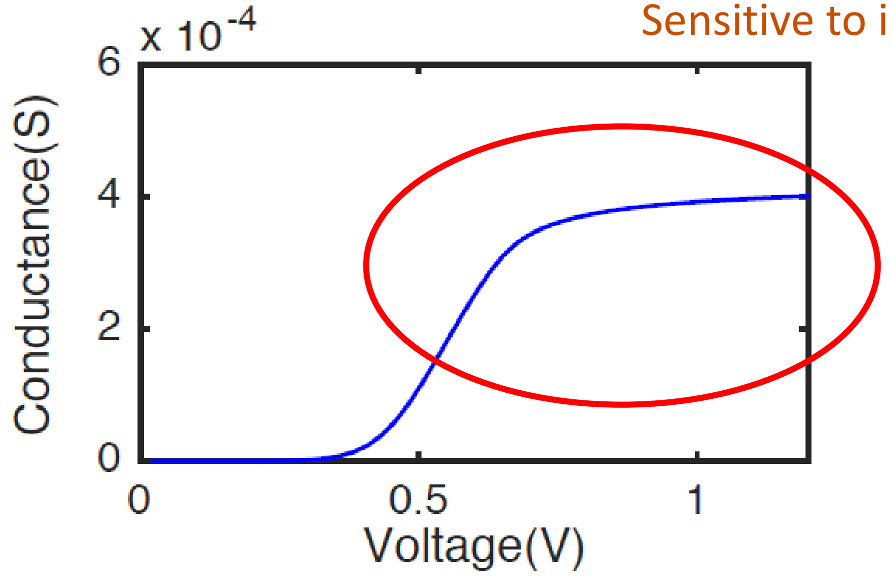
- Cell Nonlinearity: conductance varies when applied with different voltages

Ideal Cell



Non-Ideal Cell

Sensitive to input voltage



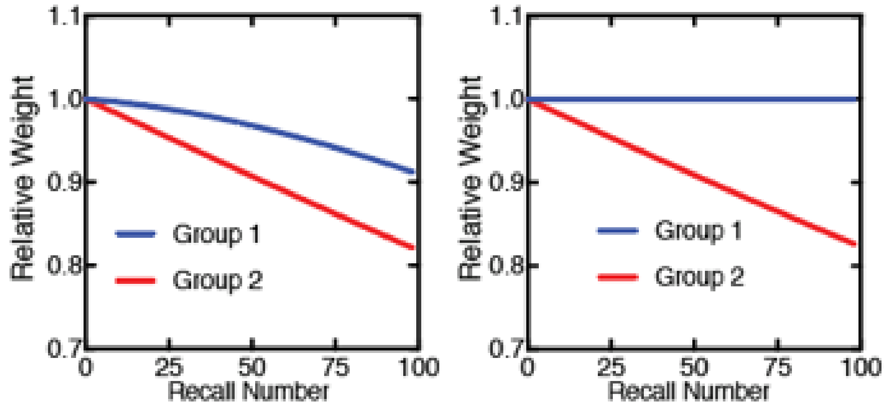
- Solution: Current Amplifier to Clamp Cell Voltage (shown in previous circuit design part)

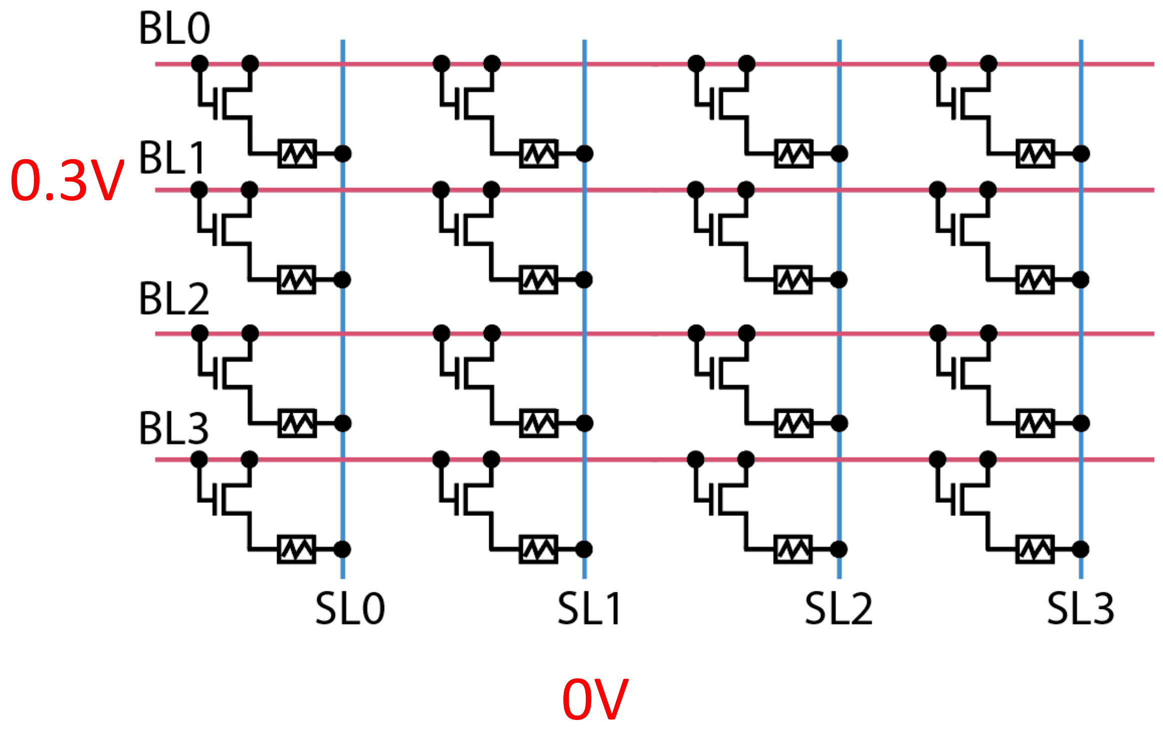
Duke

***B. Yan**, et al., ISCAS, 2016*

Non-Ideal Memristor - II

- Memristance Drift: conductance/memristance gradually deviates from original values under read voltage (read disturbance)
- Characteristic:
  - Very slow to observe



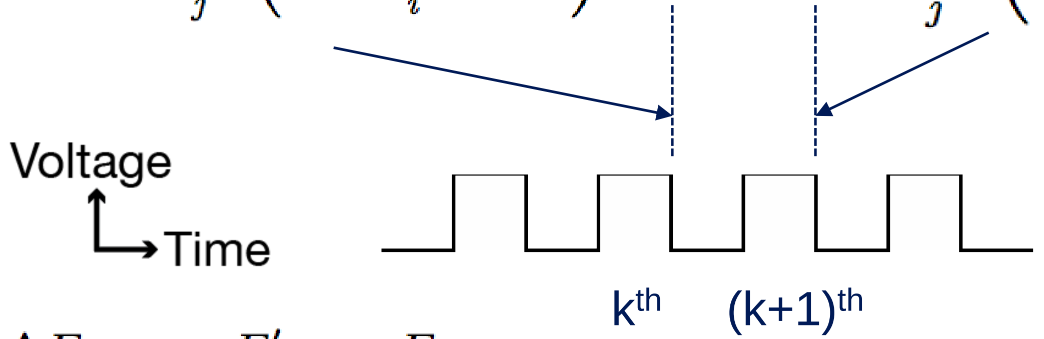


B. Yan, *et al., ICCAD, 2017*

How to Mitigate Memristance Drift for Inference?

$$E_{mse} = \sum_j^n \left(t_j - \sum_i^m w_{ij}x_i\right)^2 \qquad E'_{mse} = \sum_j^n \left(t_j - \sum_i^m w_{ij}x_i - \sum_i^m \Delta w_{ij}x_i\right)^2$$

E_{MSE}: MSE Error Function
t_j: label for each neuron

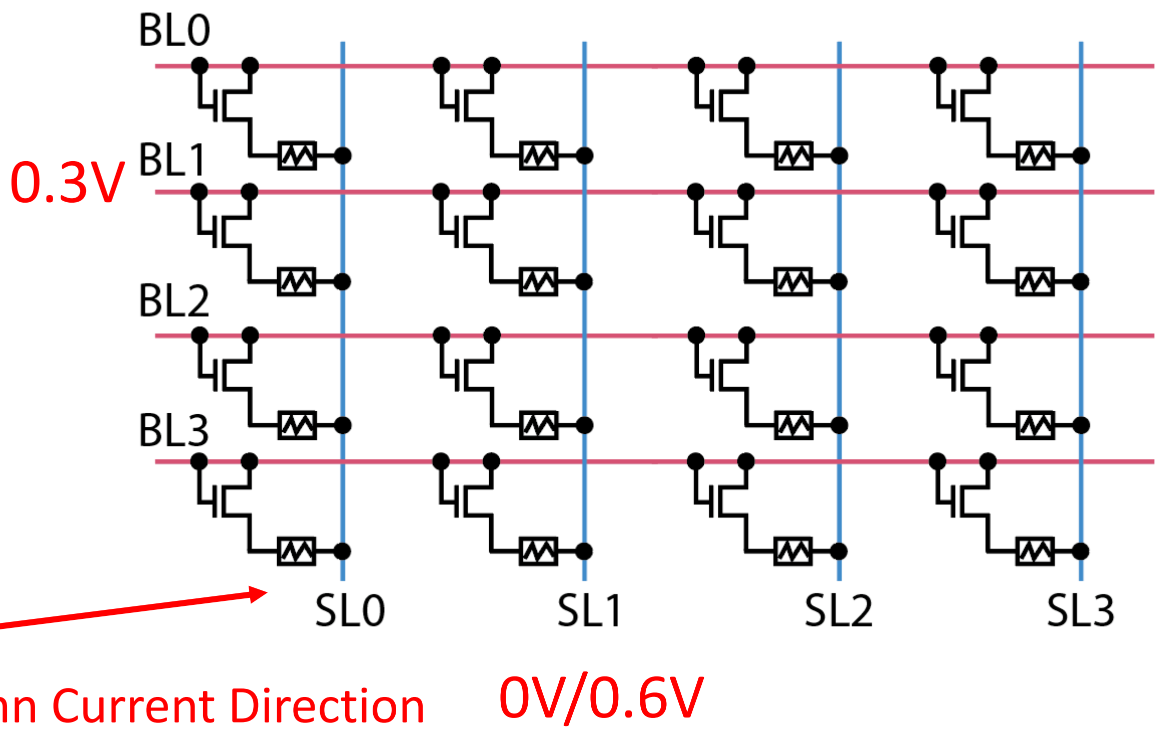


$$\Delta E_{mse} = E'_{mse} - E_{mse}$$
$$= -2\sum_j^n \left[\left(t_j - \sum_i^m w_{ij}x_i\right)\left(\sum_i^m \Delta w_{ij}x_i\right)\right]$$

Minimize Degraded Error Function:

$$\frac{\partial \Delta E_{mse}}{\partial \Delta w_{ij}} = -2\underbrace{\left(t_j - \sum_i^m w_{ij}x_i\right)}\, x_i \le 0$$

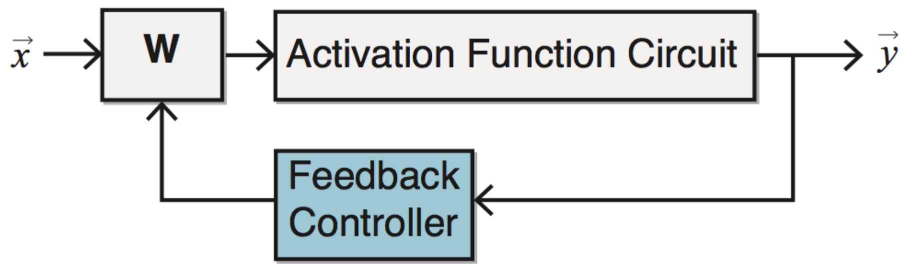
Change Column Current Direction



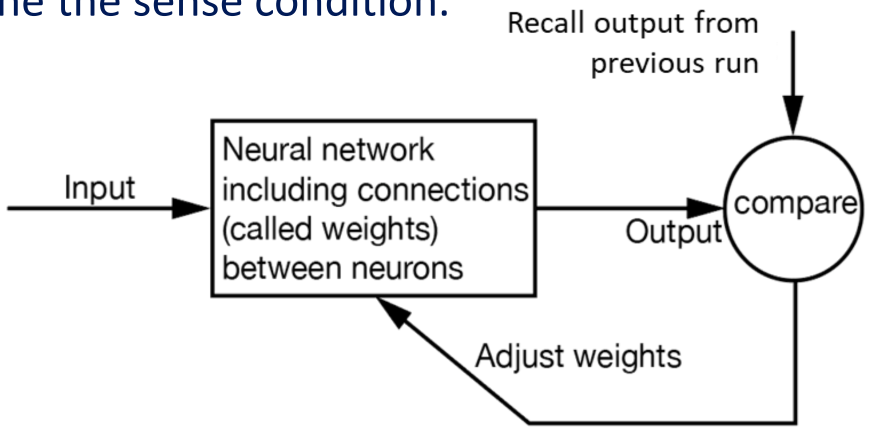
Duke

Closed-loop Design to Enhance Weight Stability

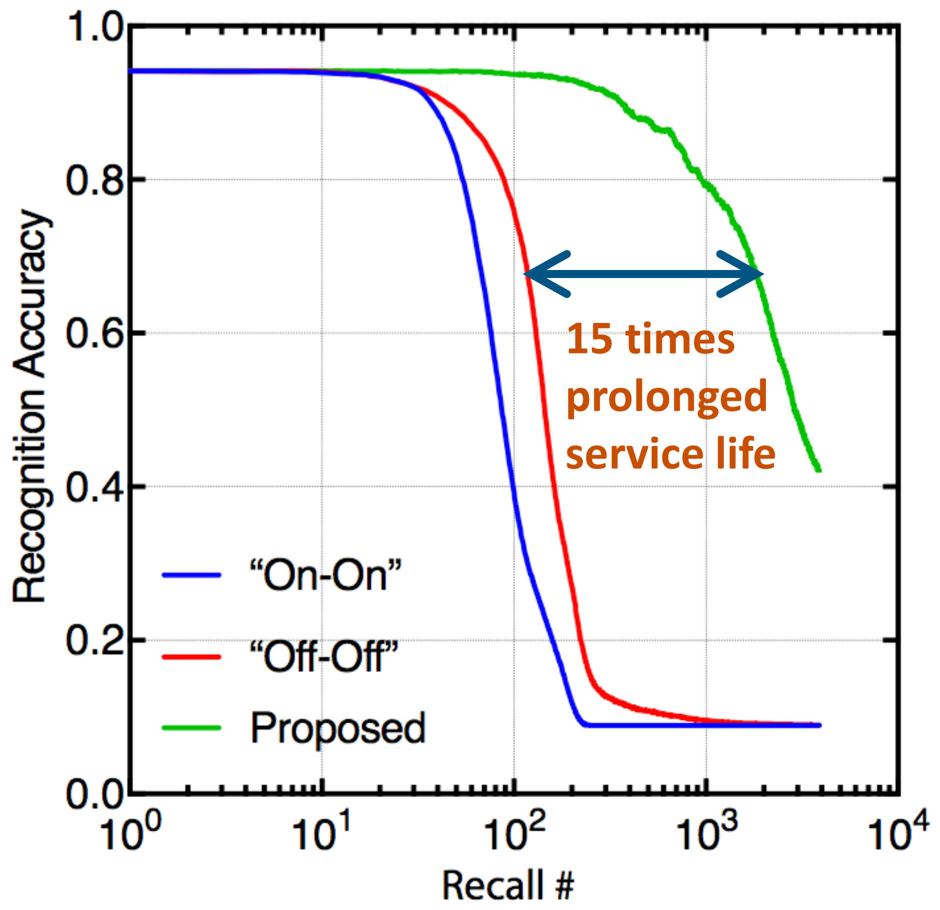
Feedback controller: Adjust the voltage condition to compensate the memristance drift.



"Arrogant principle": last output is used as the label to determine the sense condition.



Recall output from previous run
Input
Neural network including connections (called weights) between neurons
Output
compare
Adjust weights



Duke

__B. Yan__, et al, ICCAD, 2017

Summary

- The Past and Now of AI Hardware
  - In-Memory Computing Eliminates Weights Movement
- My Work:
  - Spiking-based In-Memory Computing Engine Offer Very High Energy Efficiency & Good Performance
  - Clever Design Methodologies (e.g., Closed-Loop Sensing) is Effective to Tolerate Nonideal Features of Memristors

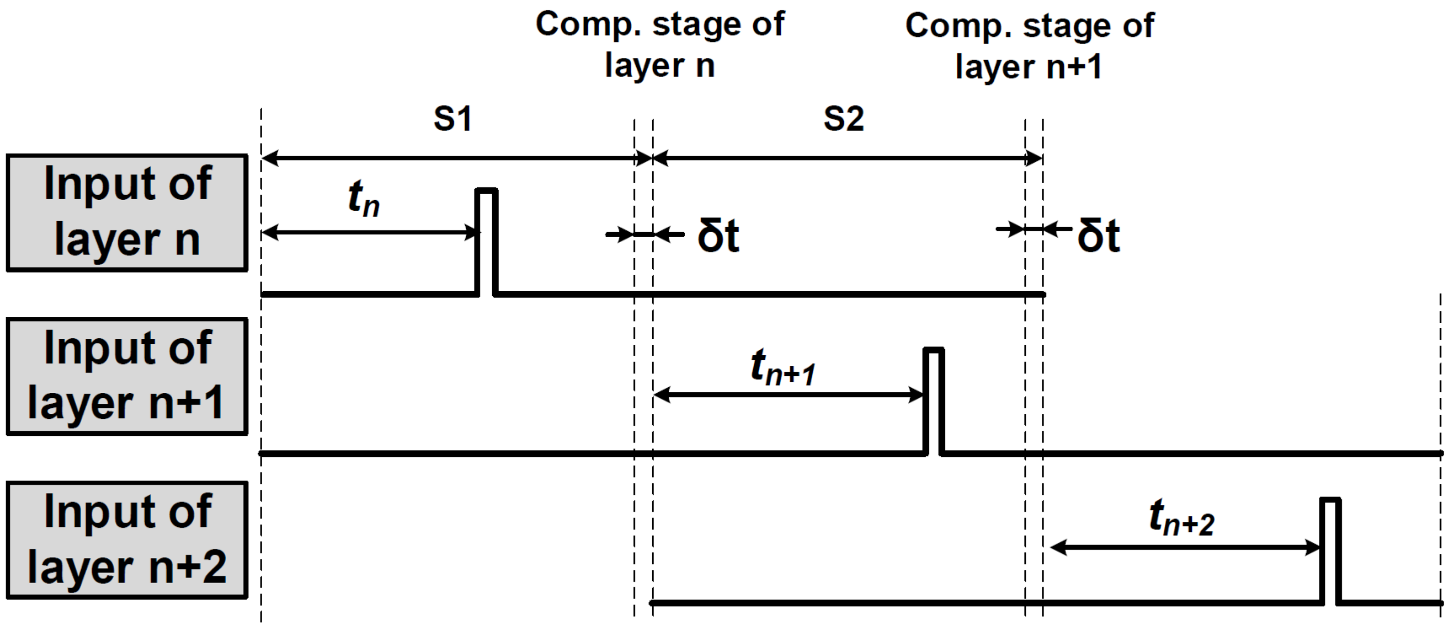
Duke

Future Work I: Single Spike Processing Engine

Use Single Spike to Replace Multiple Spikes: ~10x Improvement of Energy Efficiency

MAC: $y_j = \sum_{i=1}^{m} x_i \cdot w_{ji}$

Single Spike: $t_j = \sum_{i=1}^{m} t_i \cdot G_{ji}$



"ReSiPE: ReRAM-based Single-Spiking Processing-In-Memory Engine"
Simulation Results Coming in July at Design Automation Conference (DAC) 2020

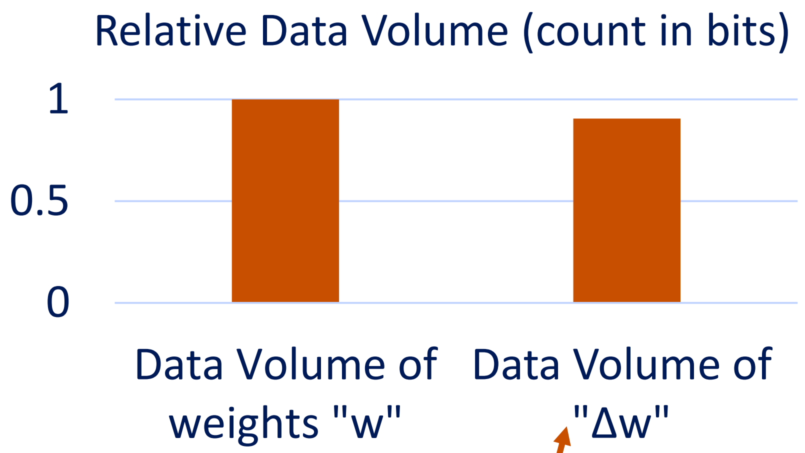
Duke

Expect to IC Tape-Out + Single-Spike Encoding Method [Potential Collaboration Opportunity]

Future Work II: In-Memory Compute & Cache

Challenge:

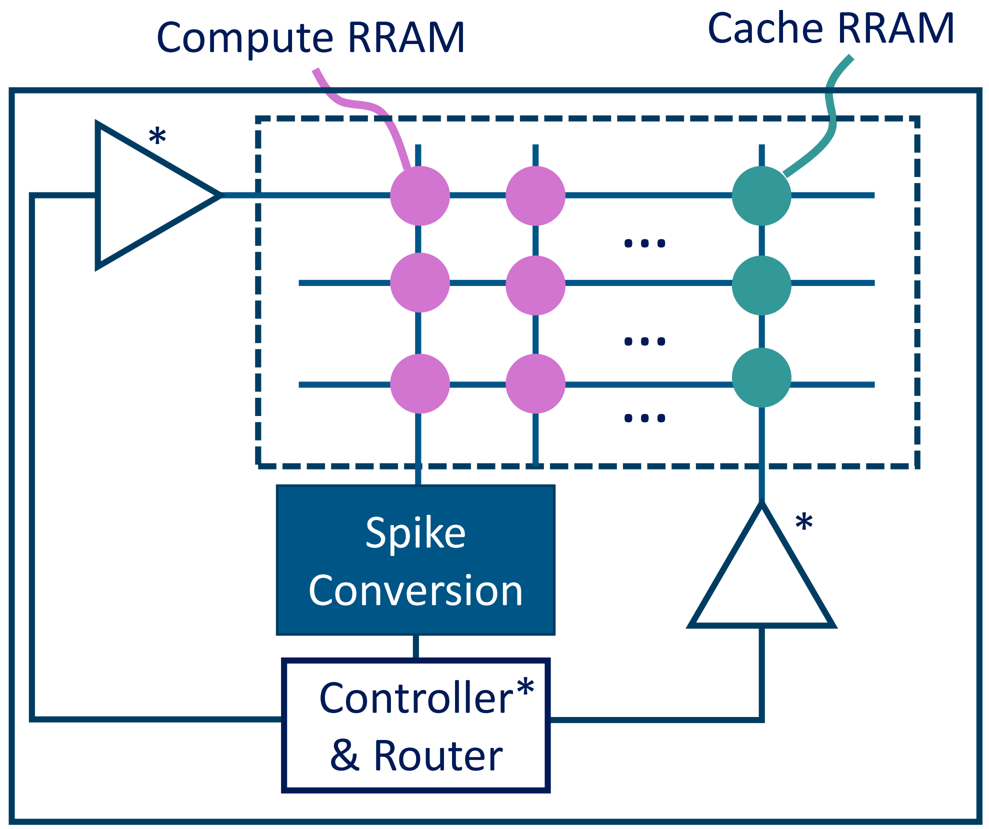
During Training:



No Need to Transport Weight

External DDRAM Access (Expensive!)

Attack:



*: novelty

Prospect Applications:

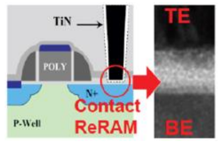
NN Training Acceleration
Self-Updatable In-Memory Computing Engine

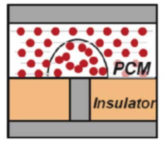
Long-Term Prospects

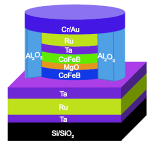
Compute

Control

Rethink Compute Memory Hierarchies



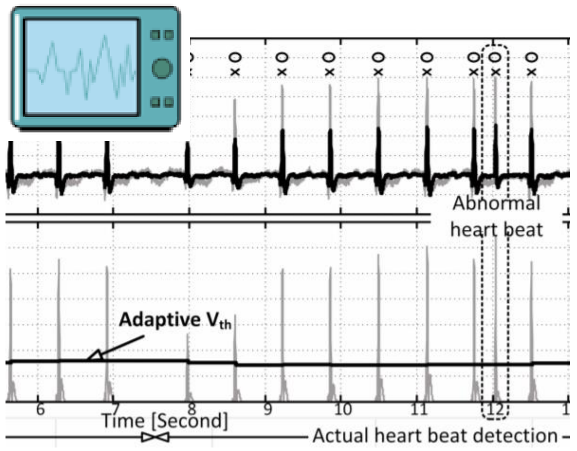




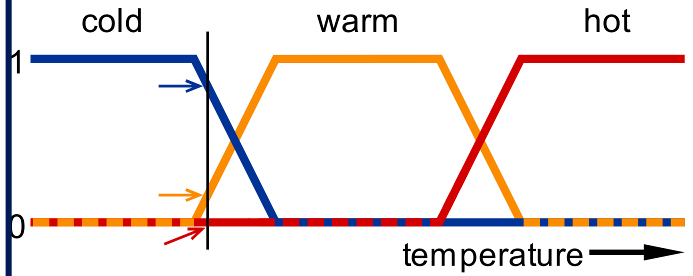
RRAM PCM MRAM

"Organ" for Compute & Cache w/ different emerging memory types

In Situ Data Processing (Near-Sensor Computing)



Analog Content Addressable Memories



Fuzzy Logic Hardware

Many Thanks for the Support!

Advisors:

Prof. **Hai "Helen" Li** Prof. **Yiran Chen**

Collaborators:

Prof. **Joe Qiu**
Army Research Office

Dr. **Qing Wu**
Air Force Research Laboratory

Prof. **Jianhua (Joshua) Yang**
UMASS Amherst

Prof. **Meng-Fan Chang**
National Tsing-Hua Univ.

Prof. **Krishnendu Chakrabarty**
Duke University

Prof. **Weisheng Zhao**
Beihang University

Student Collaborators:

Ziru Li, **Qilin Zheng**, **Brady Taylor**

Duke

Thanks for Listening & Qs!

slides available at:

https://bonanyan.github.io/bn/

Duke