Highly Efficient Neuromorphic Computing Systems with Emerging Nonvolatile Memories

Bonan Yan

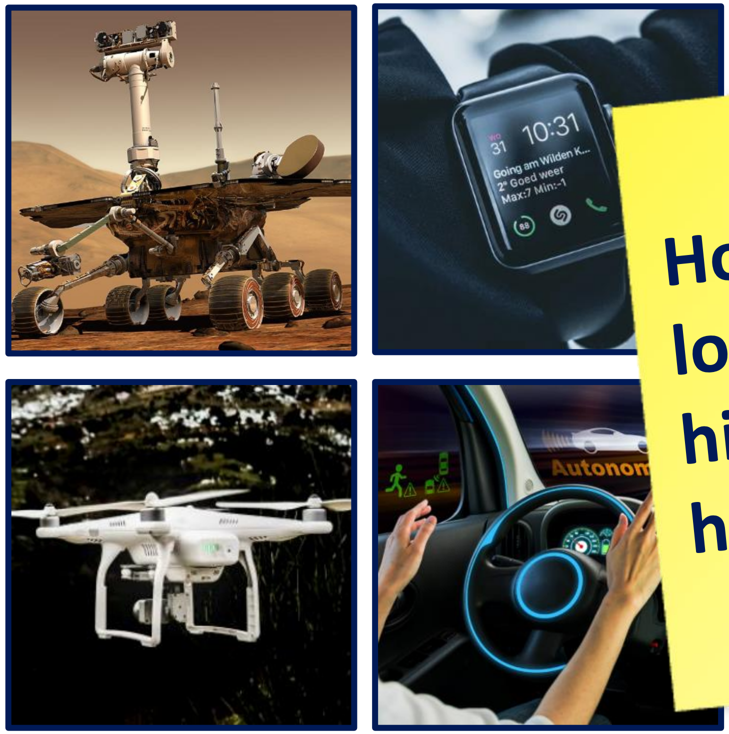
Dept. Electrical & Computer Engineering

Duke University

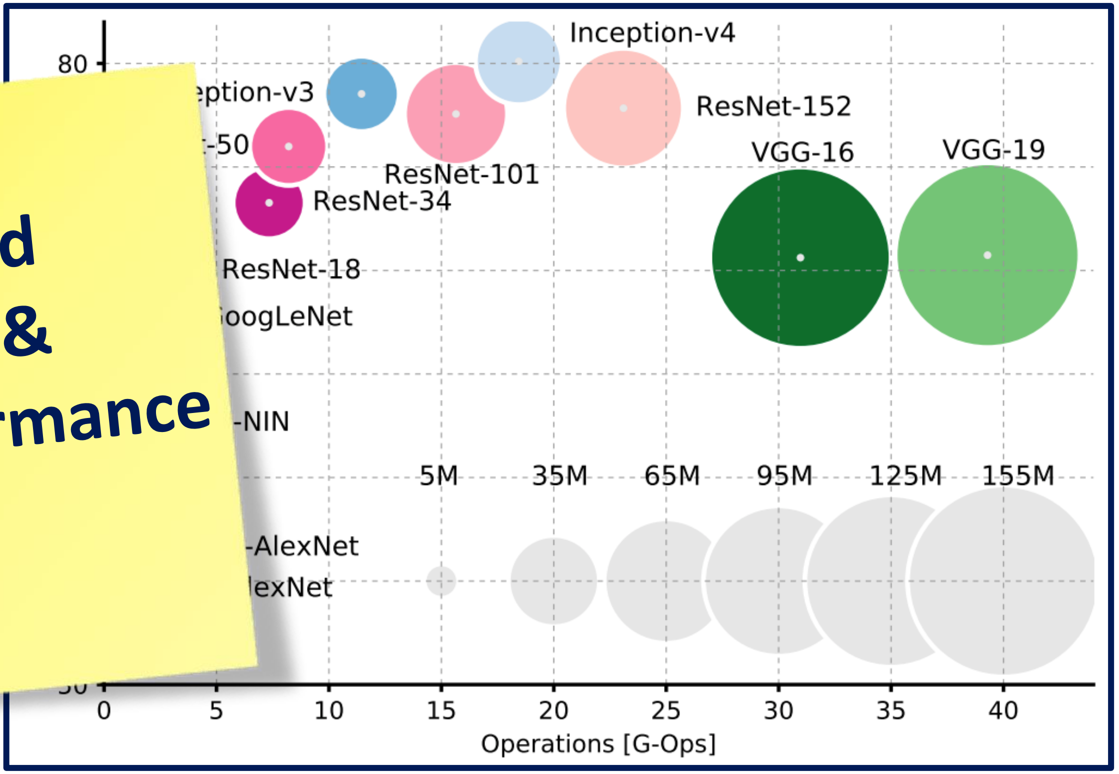
Slides available at: https://bonanyan.github.io/bn/

Efficiency Is The Key to Ubiquitous AI

Limited Power/Energy



Better Accuracy Comes From Larger Models



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

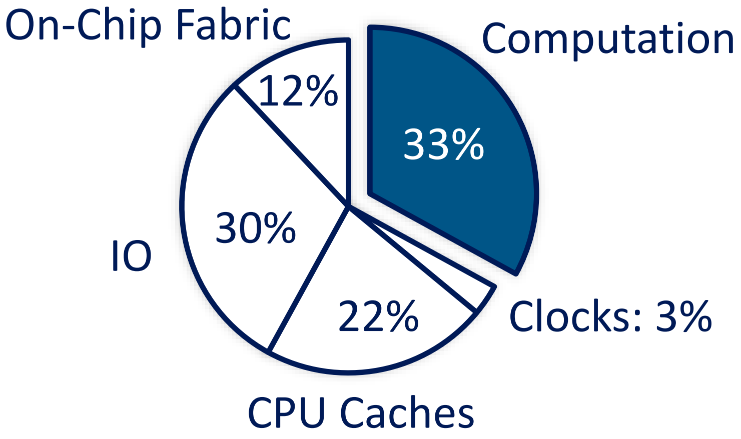
Duke

Source: IEEE Spectrum, arXiv:1605.07678

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

Overhead Dominated by Memories

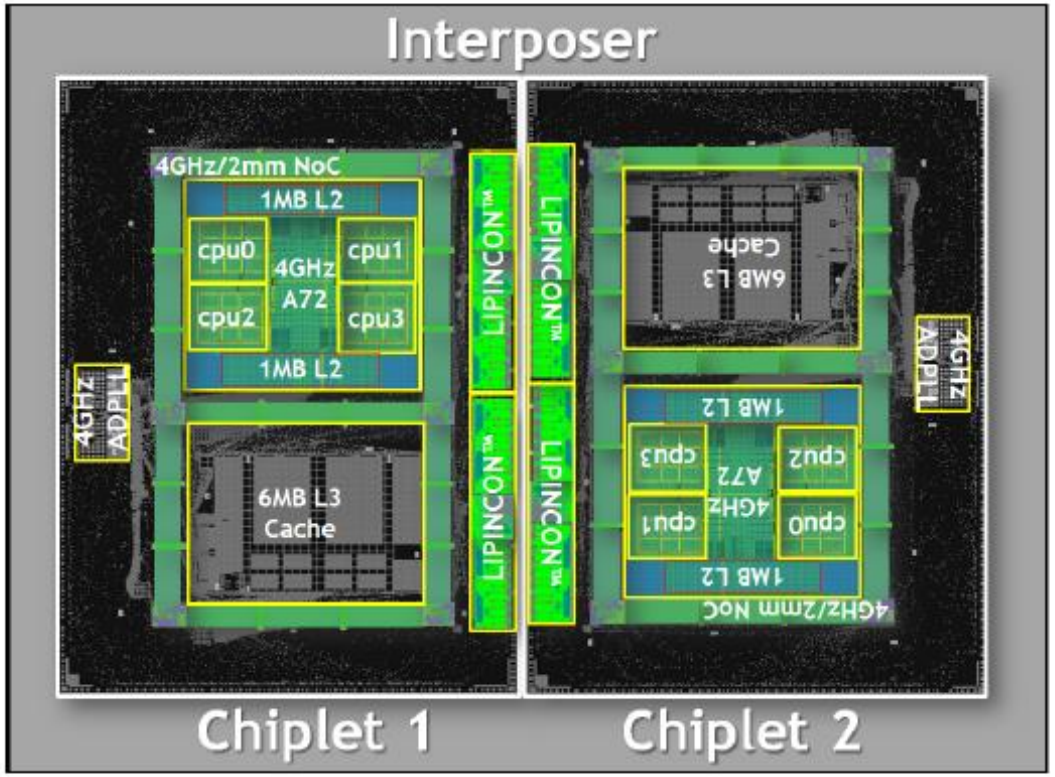
Power



Energy

Operations (32bit)	Energy (pJ)
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

Area



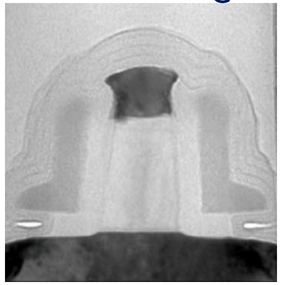
Source: AMD, Intel, [Mu-Shan Lin, et al, JSSC 2019]

Specialized Hardware Enhances Efficiency

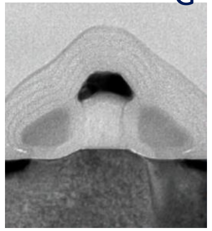
$$P = \alpha C V_{DD}^{2} f$$

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

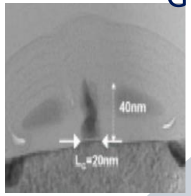
50nm L_G



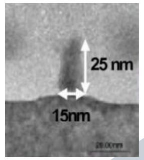
30nm L_G



20nm L_G



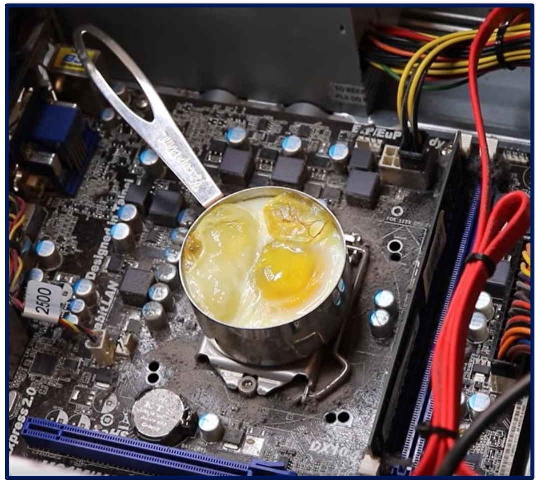
15nm L_G



MOSFET Scaling

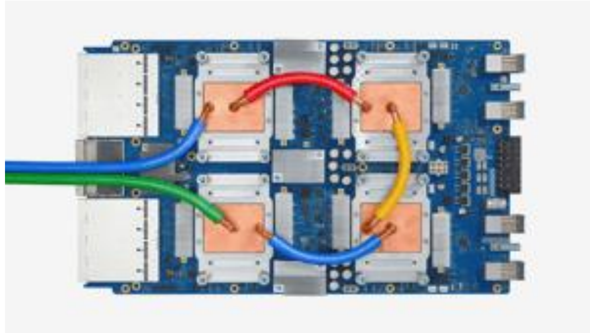
Transistor Size & Clock Frequency

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

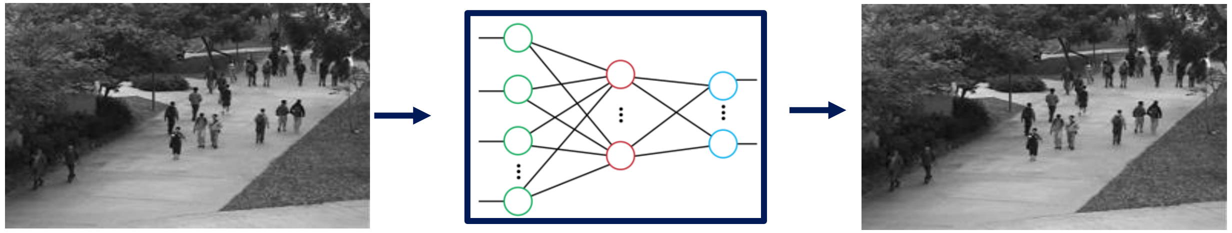
weights

outputs

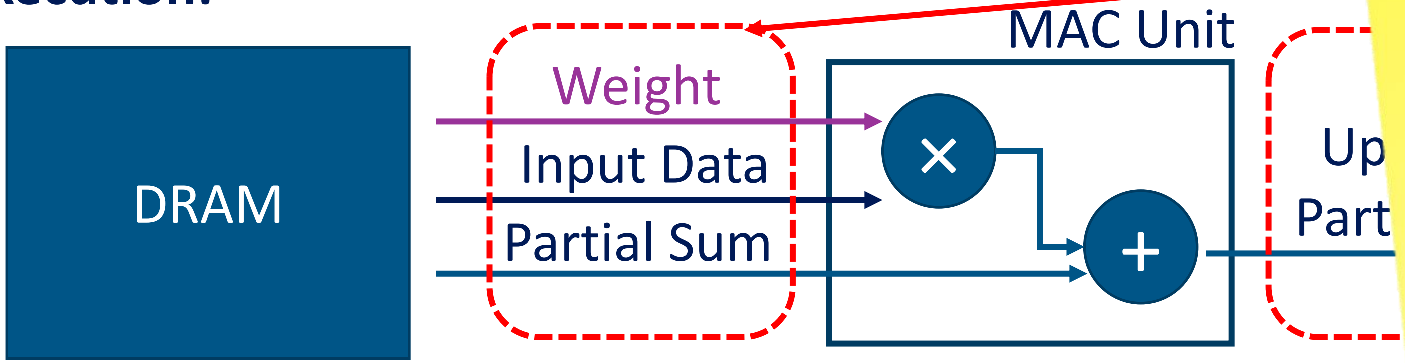
streaming

stationary

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



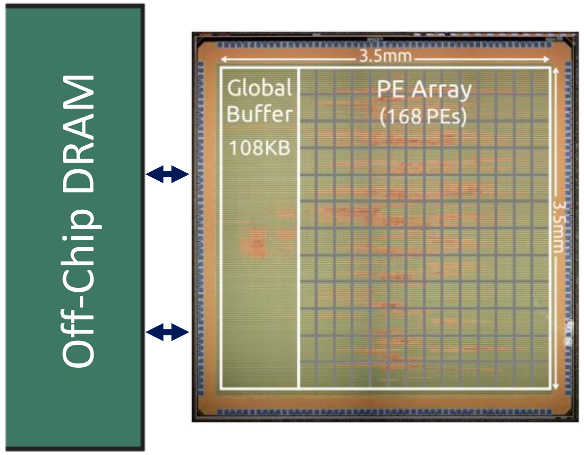
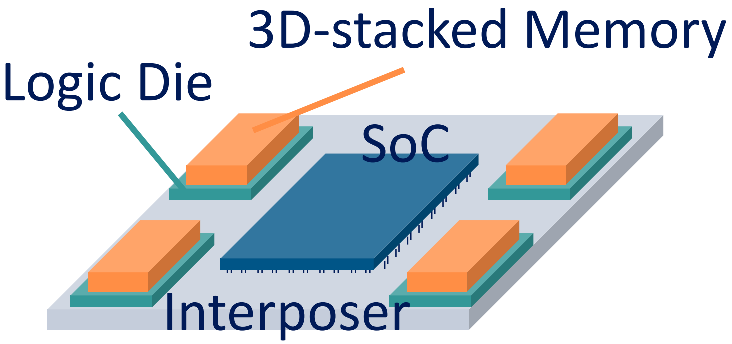
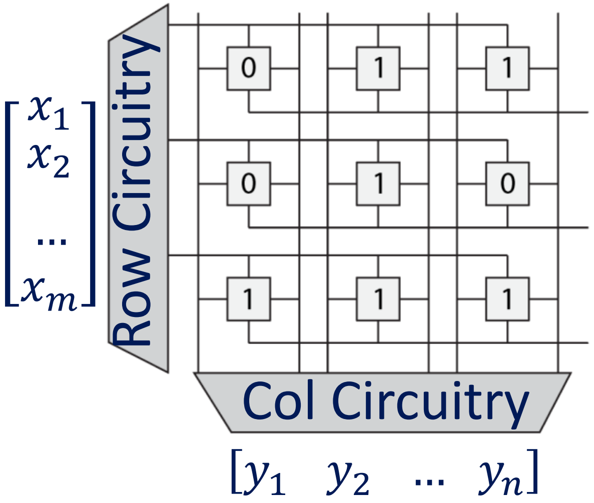
Execution:



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

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

Make Memory Access Less Expensive

	CMOS Accelerator (Planar Integration)	Near-Memory Computing	In-Memory Computing
Design Approach	 Off-Chip DRAM 3.5mm Global Buffer 108KB PE Array (168 PEs) 3.5mm	 High Bandwidth Memory	 $\begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_m \end{bmatrix}$ $\begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$ $[y_1 \quad y_2 \quad \ldots \quad y_n]$
Energy to Access Weights	200×~6× of MAC Energy	55×~6× of MAC Energy	**0**

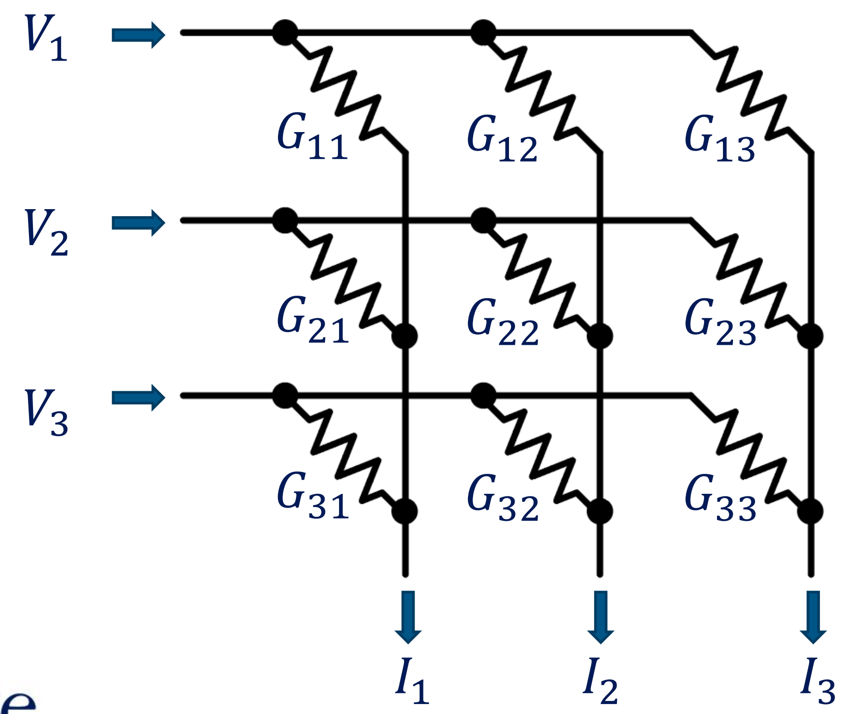
DRAM On-Chip Buffer

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

Key Idea of In-Memory Computing

inputs weights outputs

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

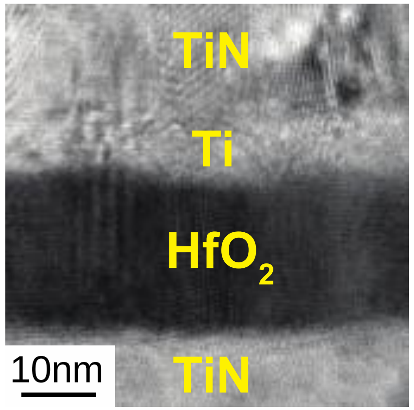


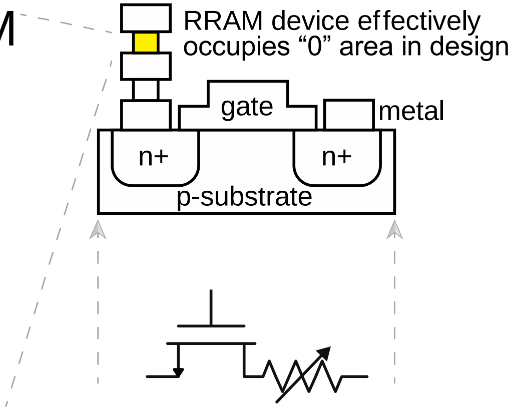
- Weight Matrix Stored as Conductance $\boldsymbol{G}$
- Rely on Analog Computation (Kirchhoff's Current Law) for “almost free”
  - Multiplication: $\mathrm{I} = V \cdot G$
  - Addition:$I^{column} = I_1^{row} + I_2^{row} + I_2^{row}$
- Ideal Nanoscale Devices for $\boldsymbol{G}$:
  - Programmable Conductance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

Memristors for In-Memory Computing

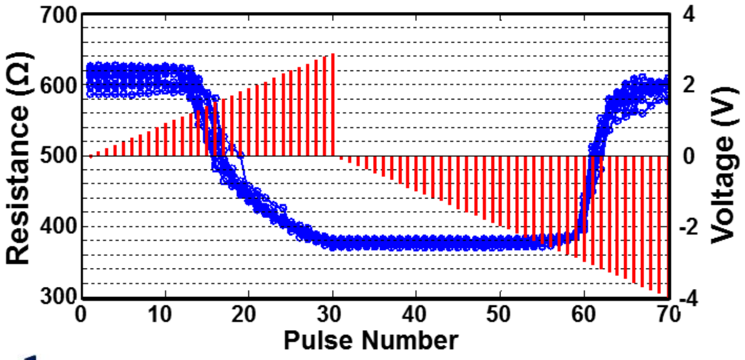
Also Called **R**esistive **R**andom **A**ccess **M**emory, RRAM or ReRAM

Cross-section TEM

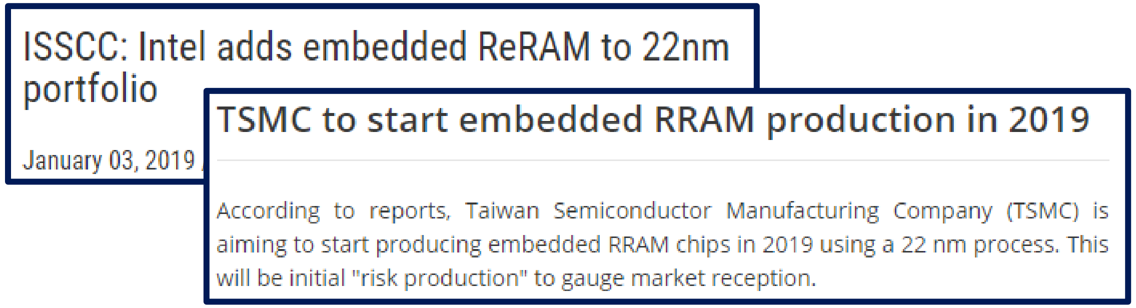




1T1R ~$50F^2$



Programmable resistor w/ analog states



ISSCC: Intel adds embedded ReRAM to 22nm portfolio

January 03, 2019

TSMC to start embedded RRAM production in 2019

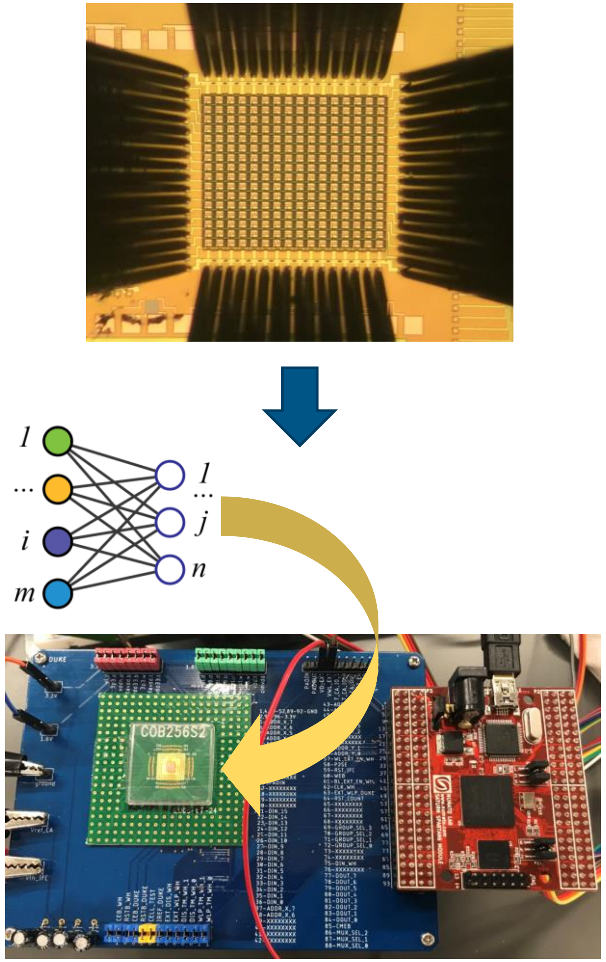
According to reports, Taiwan Semiconductor Manufacturing Company (TSMC) is aiming to start producing embedded RRAM chips in 2019 using a 22 nm process. This will be initial "risk production" to gauge market reception.

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

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]

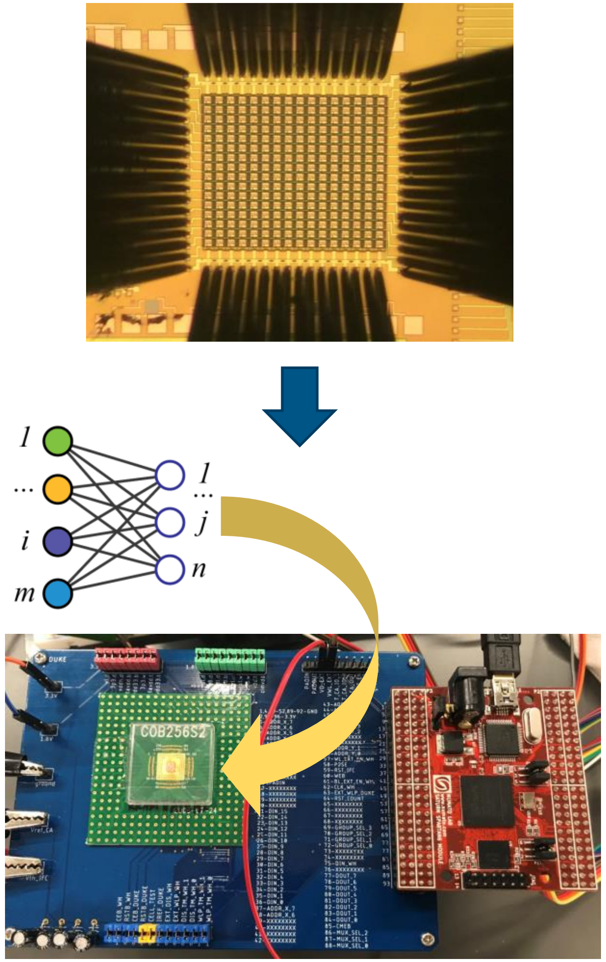
- **Tolerate/Exploit Non-ideal Behavior of Memristors**
  - Device Nonlinearity [ISCAS’16, IEDM’17, IEDM’19]
  - Read Disturbance [ICCAD’17], Hard Fault [ITC’19]



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]

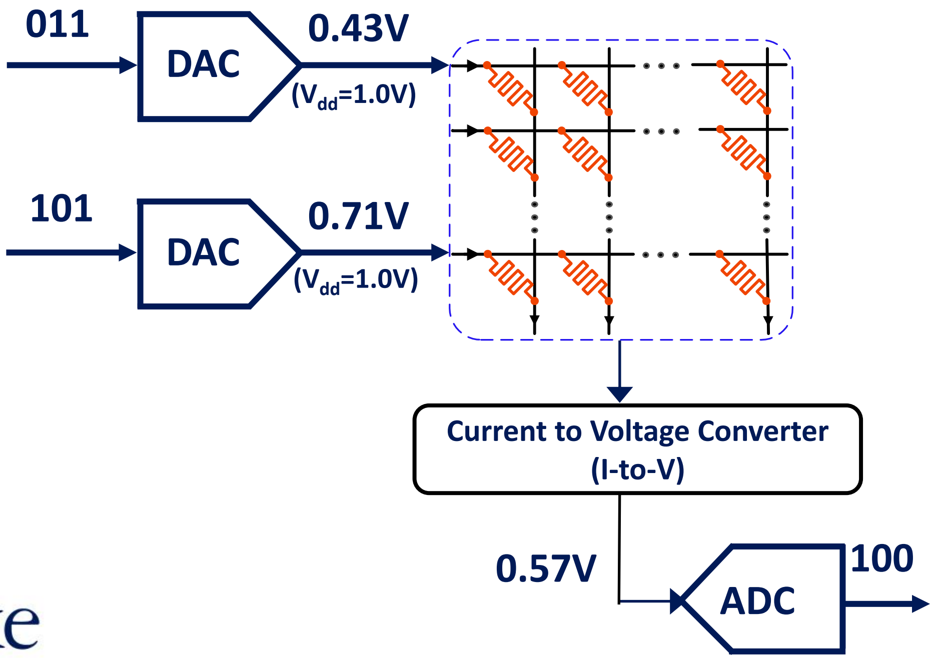
- **Tolerate/Exploit Non-ideal Behavior of Memristors**
  - Device Nonlinearity [ISCAS’16, IEDM’17, IEDM’19]
  - Read Disturbance [ICCAD’17], Hard Fault [ITC’19]



Conventional ADC is Too Large

The Level-based Design

- Compatible to existing signal processing
- High speed computation



011

DAC

0.43V

(V_{dd}=1.0V)

101

DAC

0.71V

(V_{dd}=1.0V)

Current to Voltage Converter
(I-to-V)

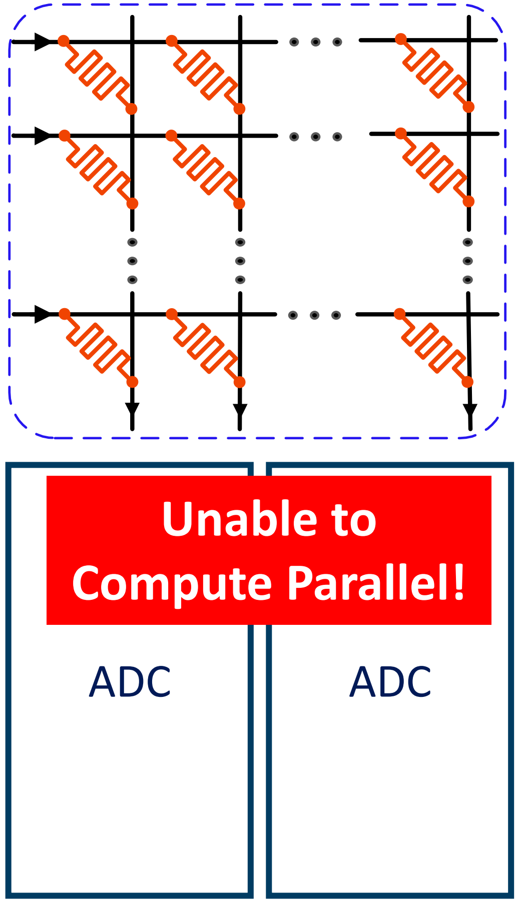
0.57V

ADC

100

Actual Layout:

256x256 1T1R Array



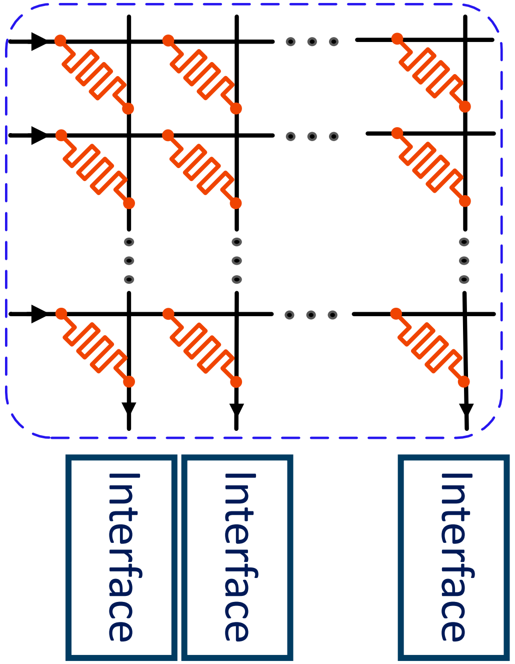
Unable to Compute Parallel!

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

My Approach: Spiking Interface Circuit

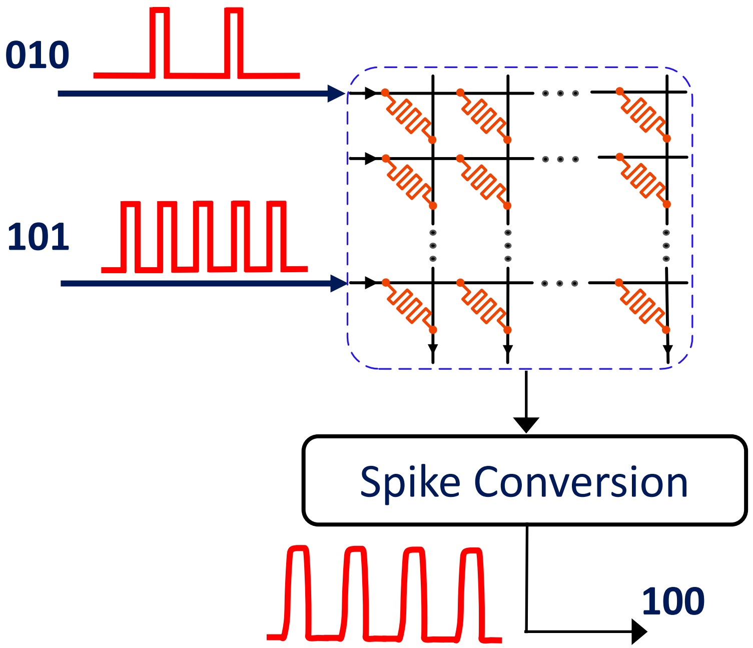
What Better Designs Look Like

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



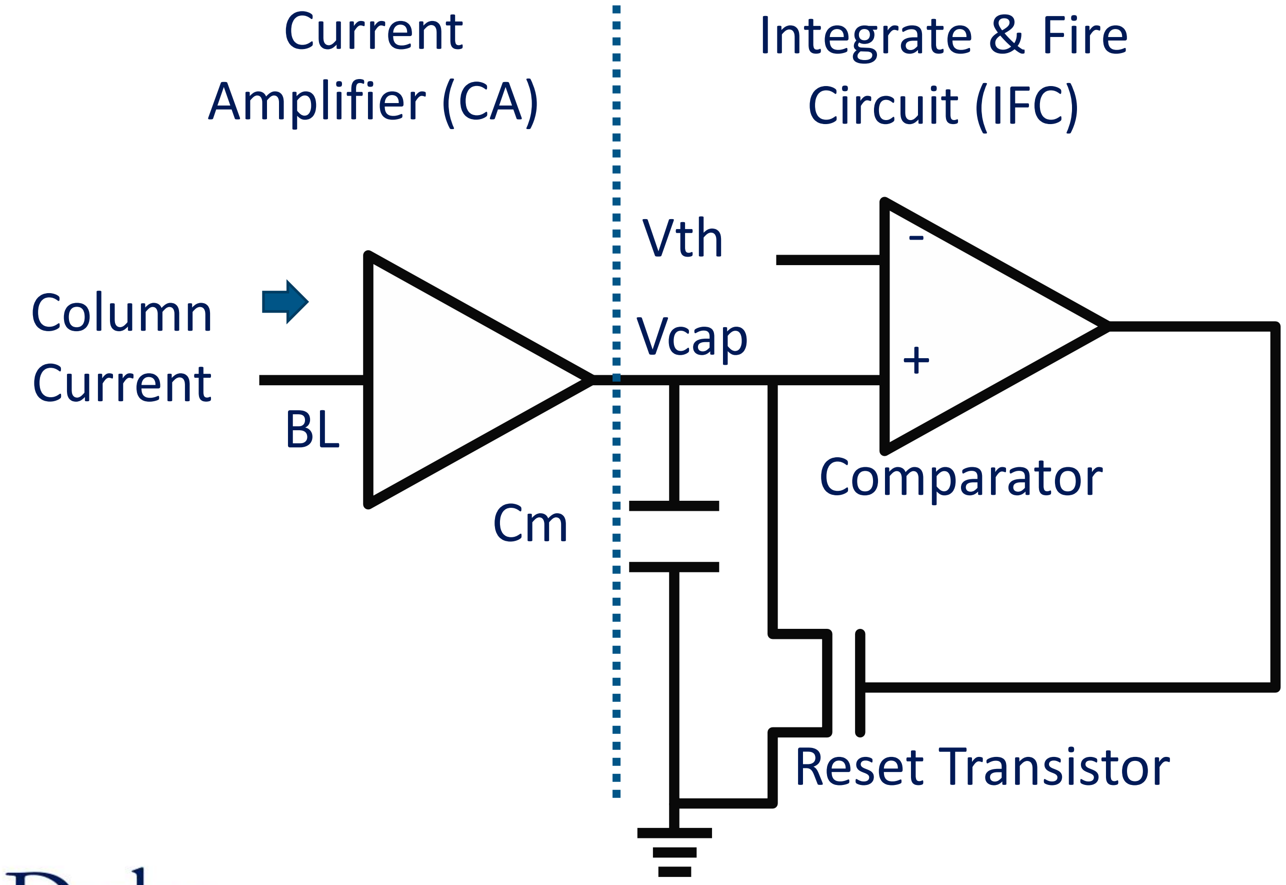
The Spike-based Design

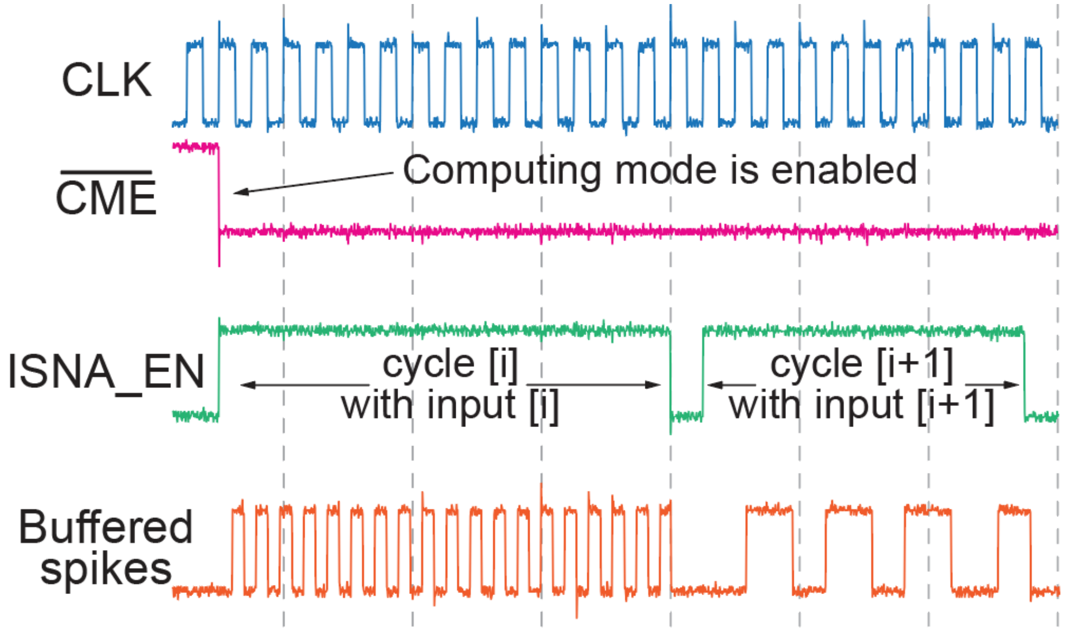
- Closer to biological system
- Extremely high power efficiency

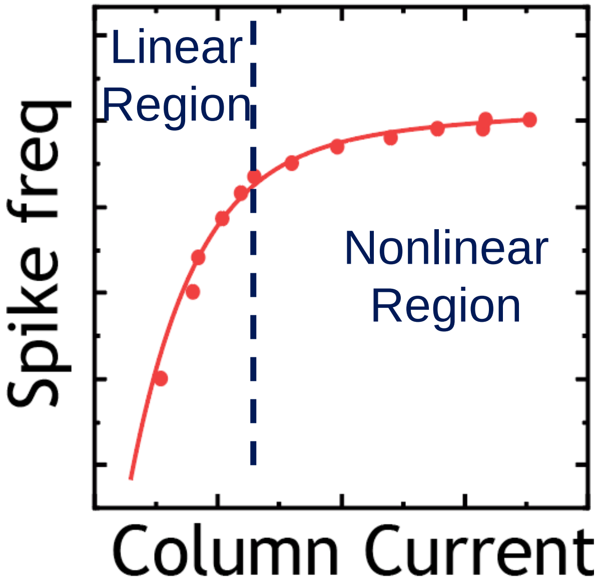


Duke

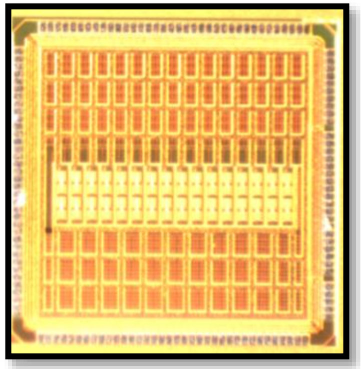
Spike Conversion





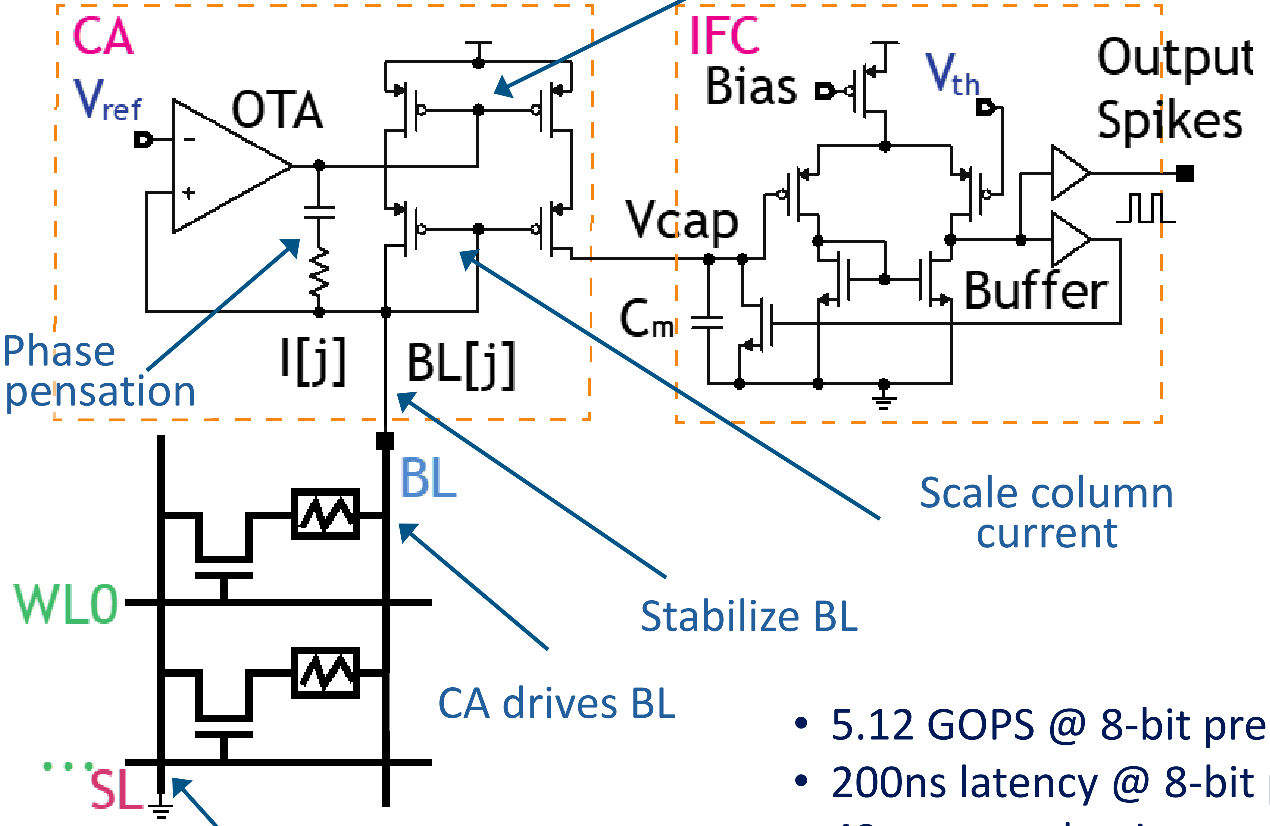


Spike Conversion Circuit & Controller
3152x3152 µm^2



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

Spike Conversion



Positive feedback to accelerate startup

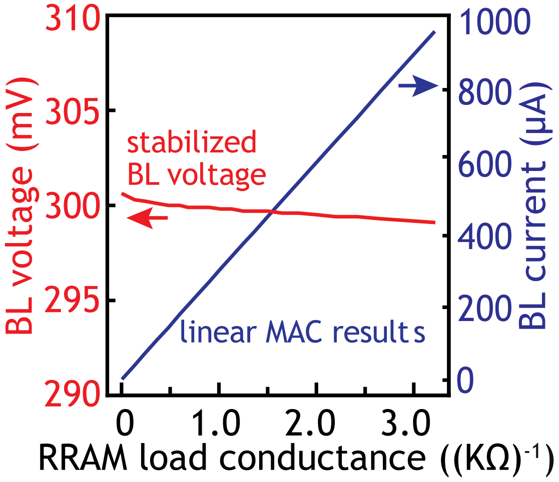
Phase compensation

Scale column current

Stabilize BL

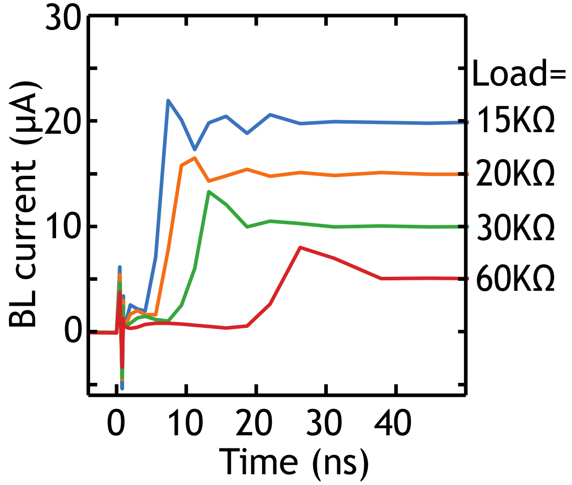
CA drives BL

SL sink current



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)

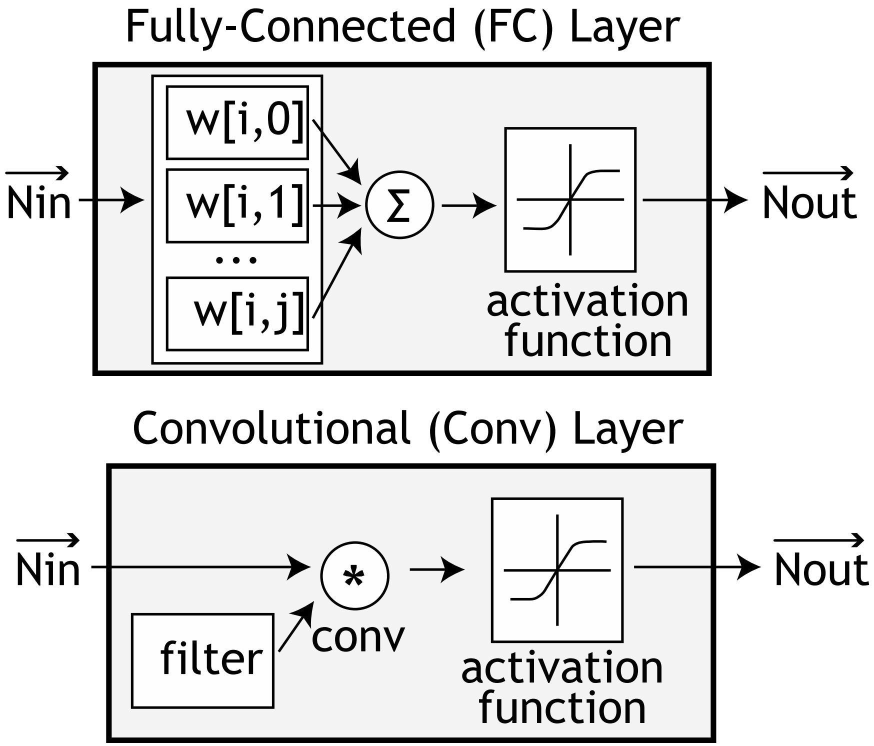


Enlarge phase margin tolerating capacitor positive feedback

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

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

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

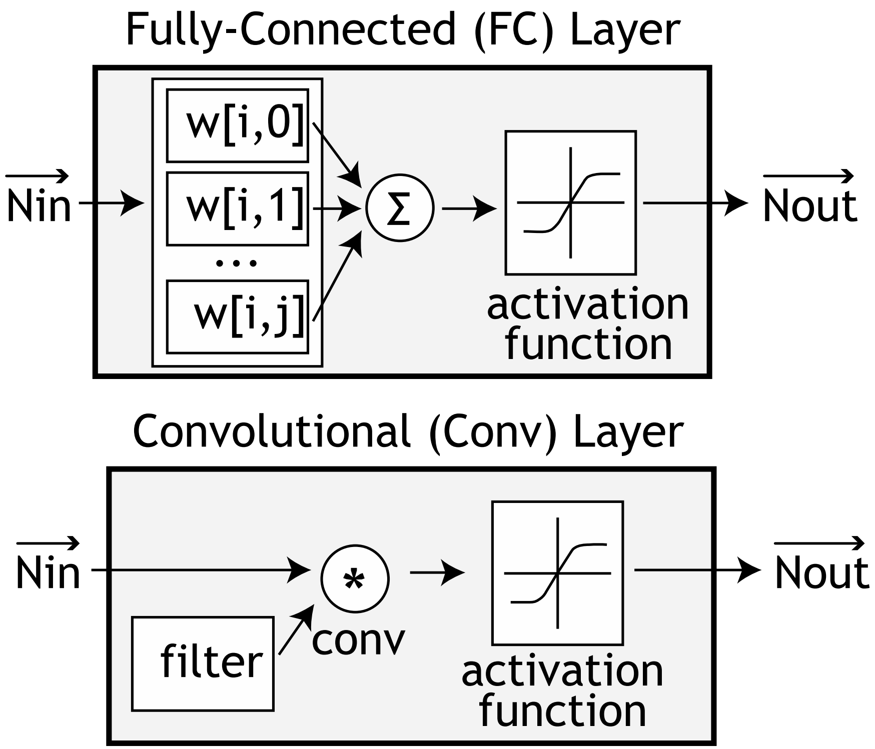
D • Step 5: Store results to buffer

D : digital domain A : analog domain

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

Duke

In Situ Nonlinear Activation (ISNA) Function



Single-layer Inference Operation:

D • Step 1: Load data from buffer

A • Step 2: Vector-matrix multiplication
• Step 3: Nonlinear activation function

D • Step 4: Pooling

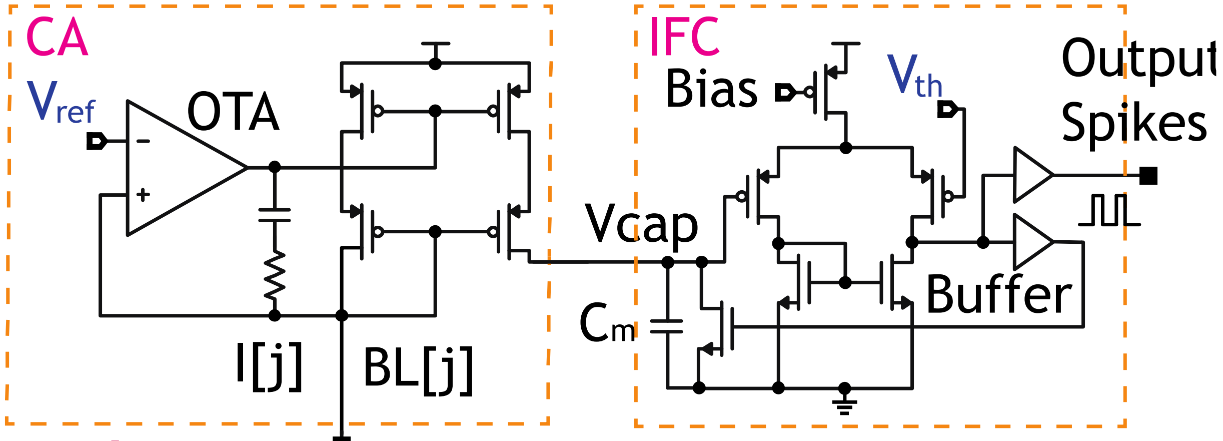
D • Step 5: Store results to buffer

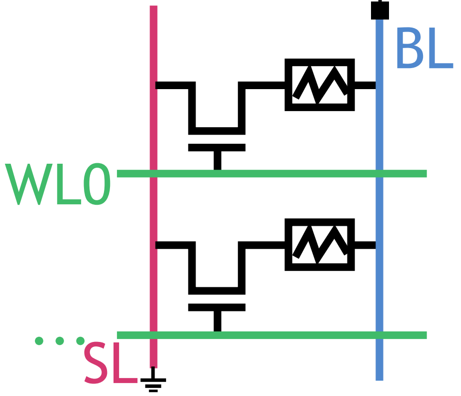
D : digital domain **A** : analog domain

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

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

Adjust Activation Function



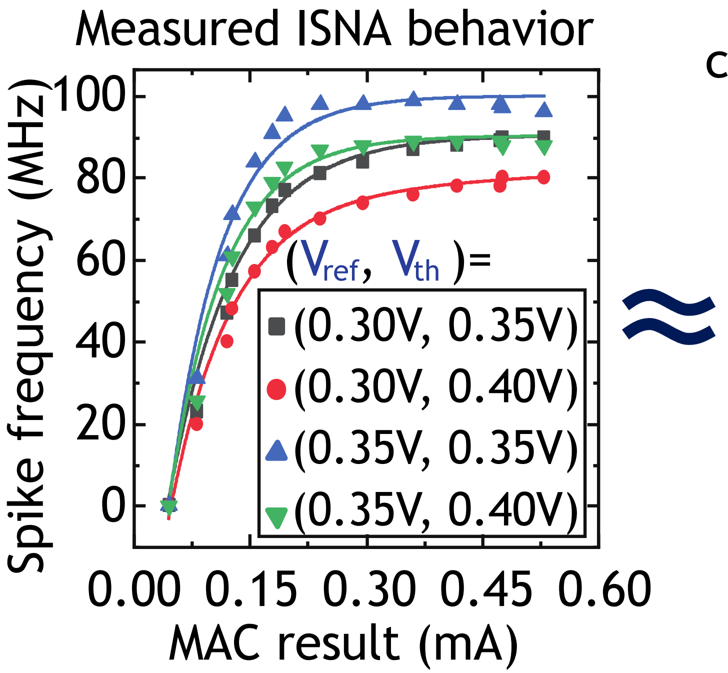


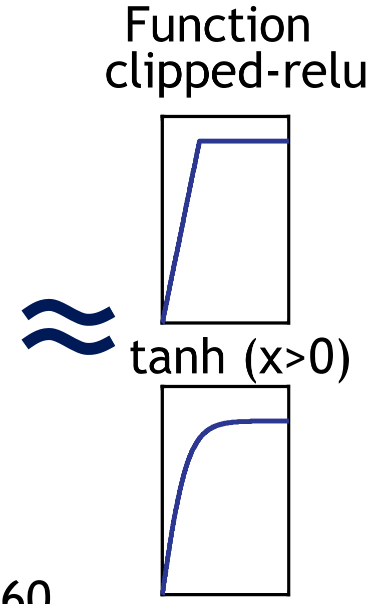
Vref: Read voltage of BL

- Vref ↑, BL current ↑
- Proportional tuning

Vth: Threshold of capacitor charging/discharging

- Vth ↓, Charging/discharging ↑
- Distorted tuning

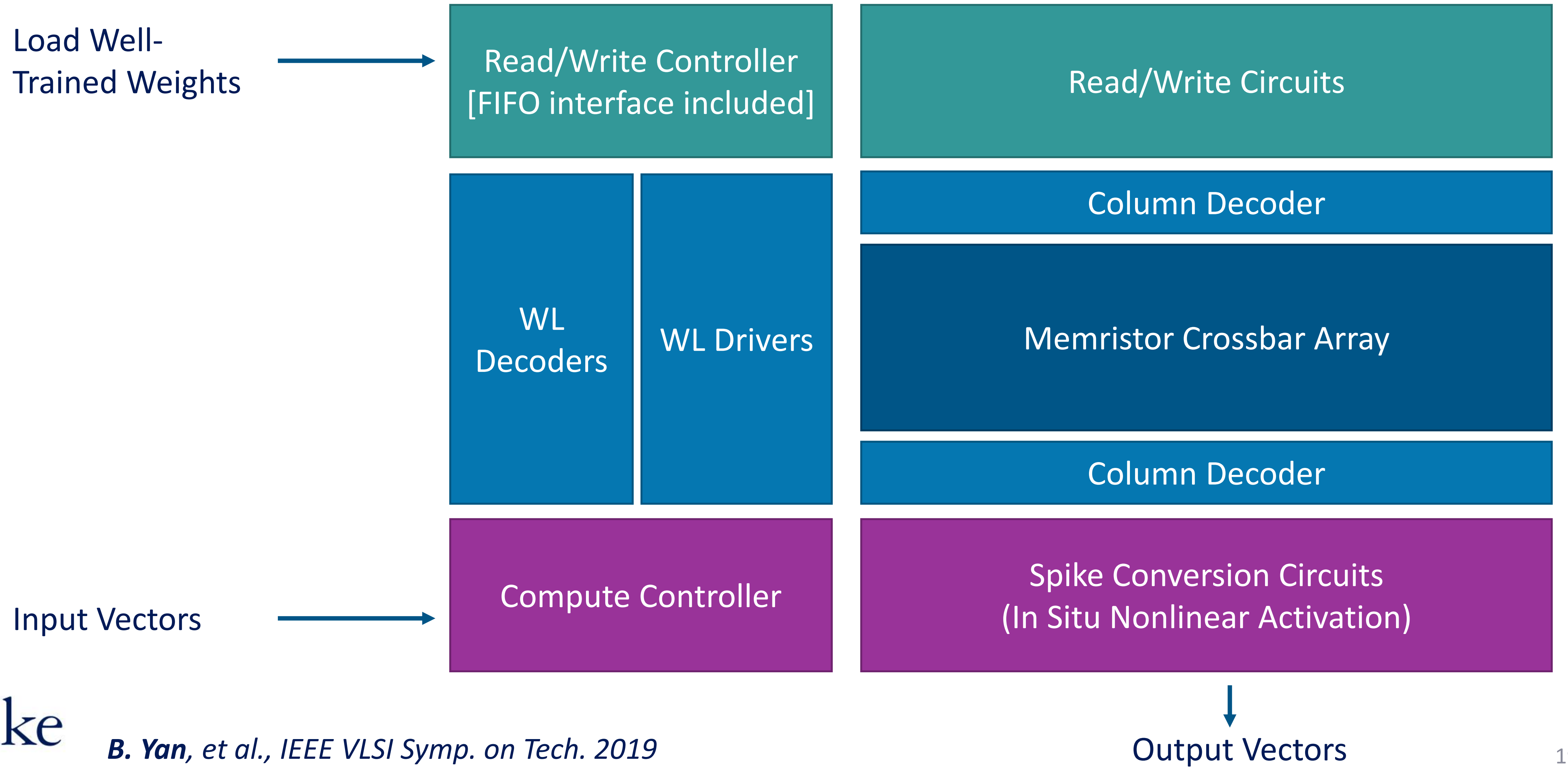




B. Yan, et al., IEEE VLSI Symp. on Tech. 2019

Duke

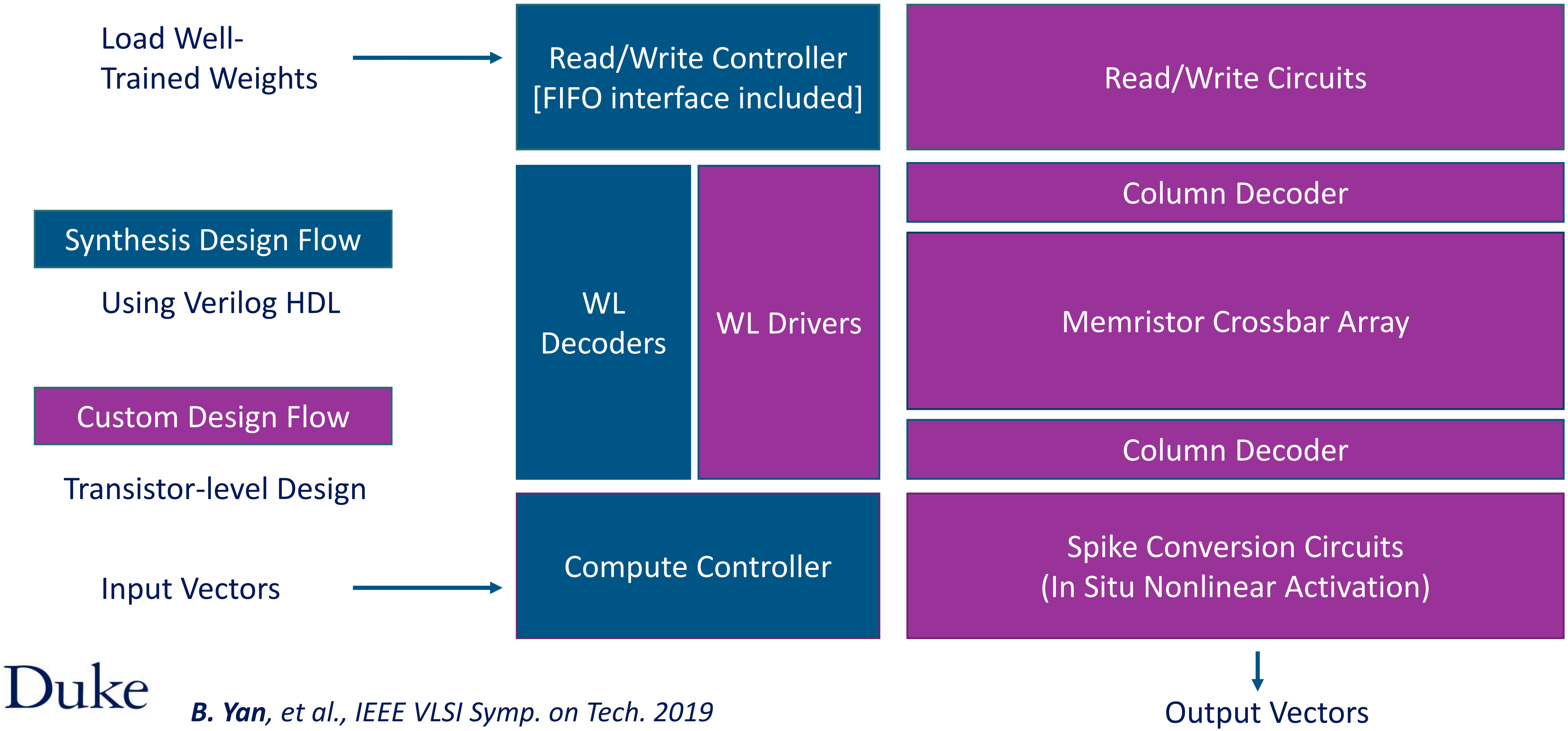
Mixed-Signal Chip Architecture



Duke

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

Mixed-Signal Design Flow



Load Well-Trained Weights

Read/Write Controller [FIFO interface included]

Read/Write Circuits

Synthesis Design Flow

Using Verilog HDL

Custom Design Flow

Transistor-level Design

WL Decoders

WL Drivers

Column Decoder

Memristor Crossbar Array

Column Decoder

Input Vectors

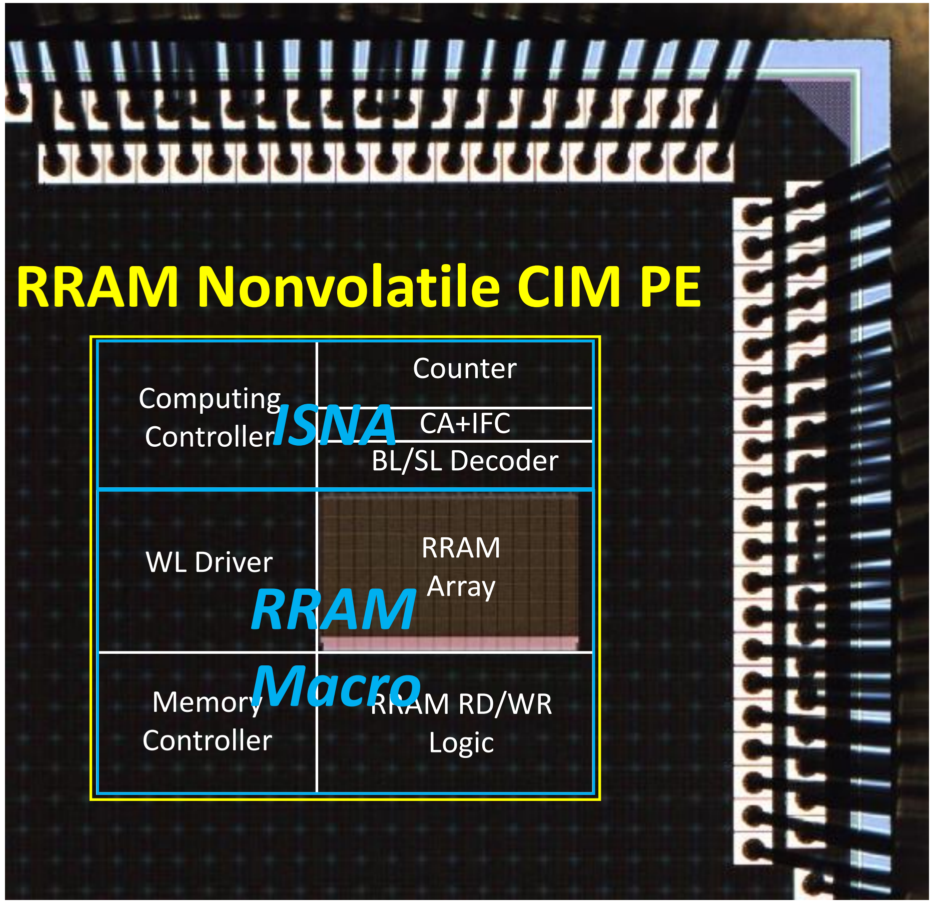
Compute Controller

Spike Conversion Circuits (In Situ Nonlinear Activation)

Output Vectors

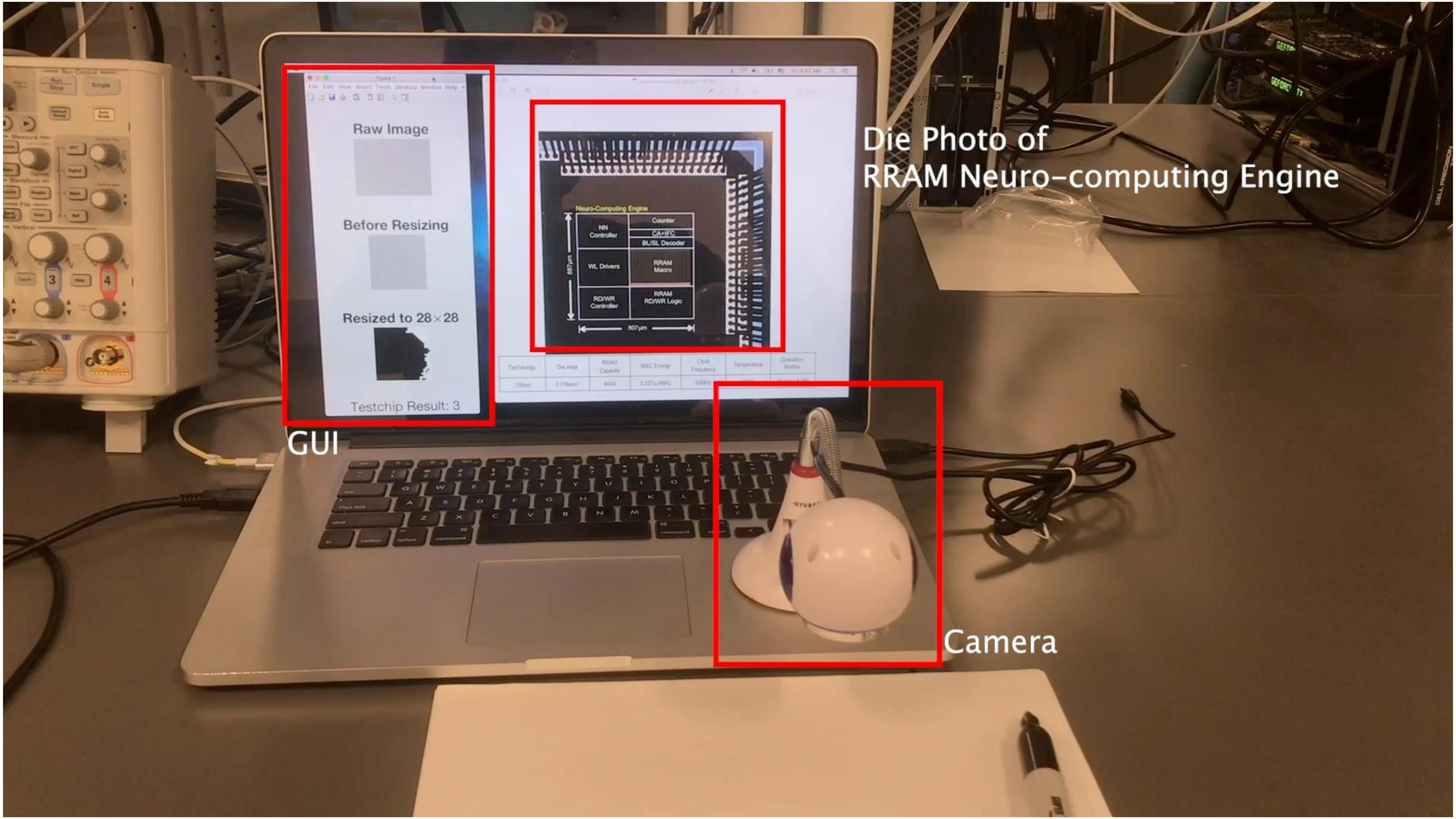
***B. Yan**, et al., IEEE VLSI Symp. on Tech. 2019*

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



Technology	Die Area	RRAM Capacity	MAC Energy	Clock Frequency	Temperature	Operation Modes
150nm	0.719mm^2	64Kb	0.257µJ/MAC	50MHz	[illegible]	[illegible]

GUI

Die Photo of
RRAM Neuro-computing Engine

Camera

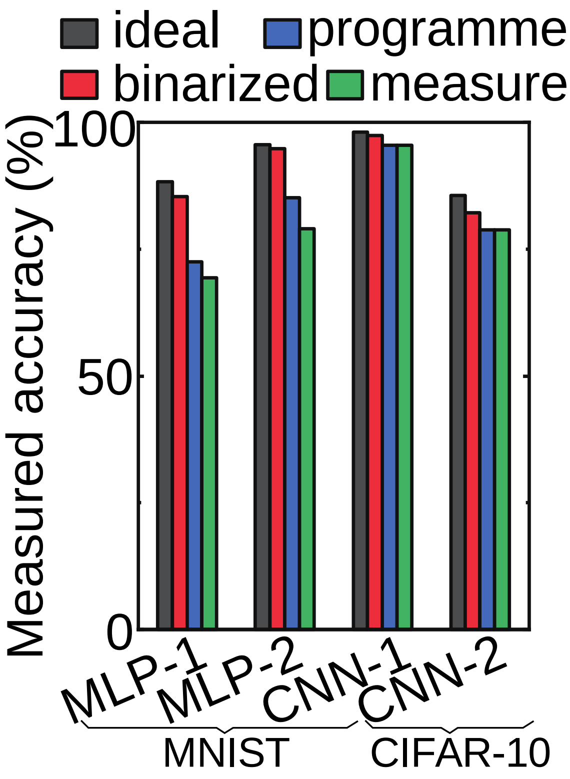
Evaluation: Measured Neural Network Results

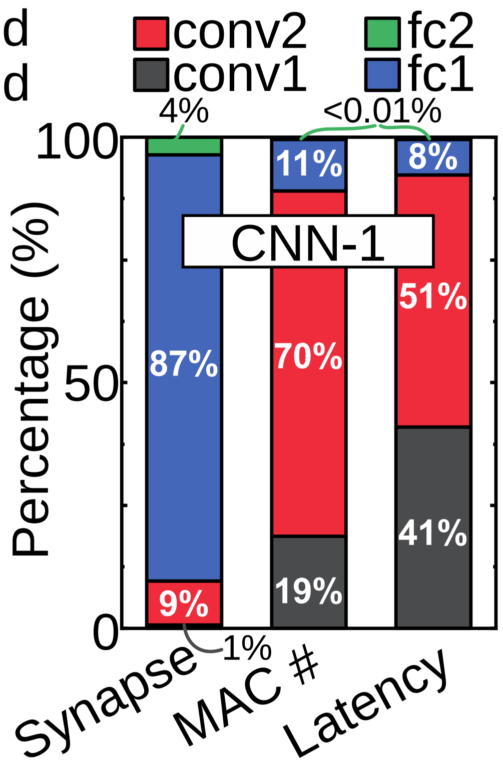
MNIST:

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

CIFAR-10:

- CNN-2: 5-Layer LeNet



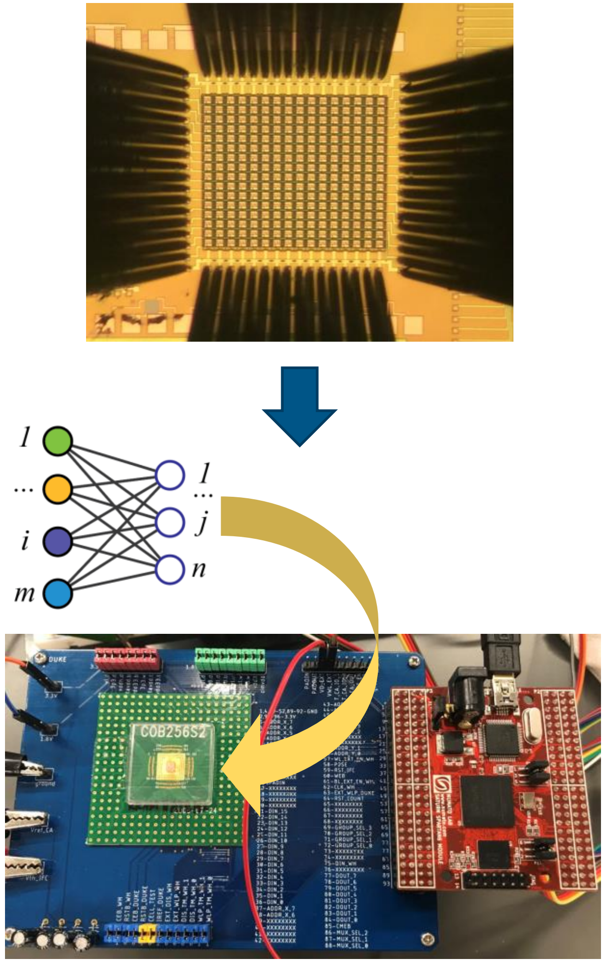


***B. Yan**, et al., IEEE VLSI Symp. on Tech. 2019*

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]

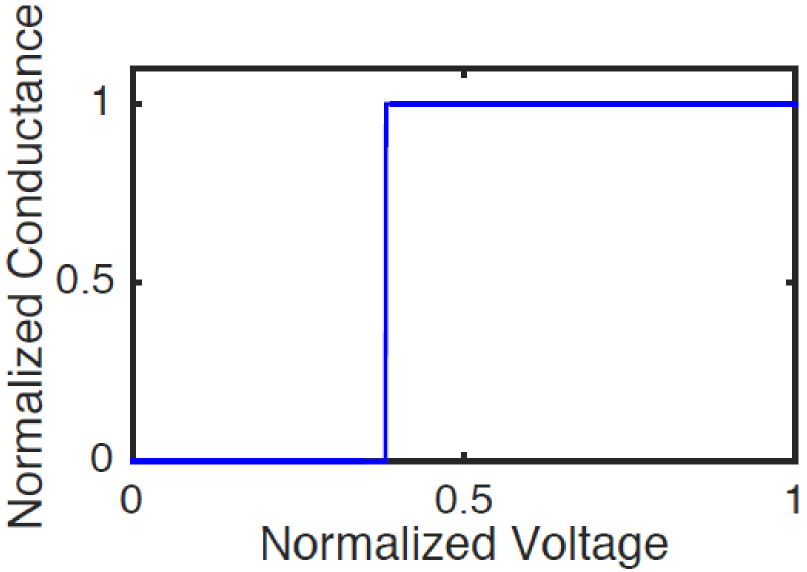
- **Tolerate/Exploit Non-ideal Behavior of Memristors**
  - Device Nonlinearity [ISCAS’16, IEDM’17, IEDM’19]
  - Read Disturbance [ICCAD’17], Hard Faults [ITC’19]



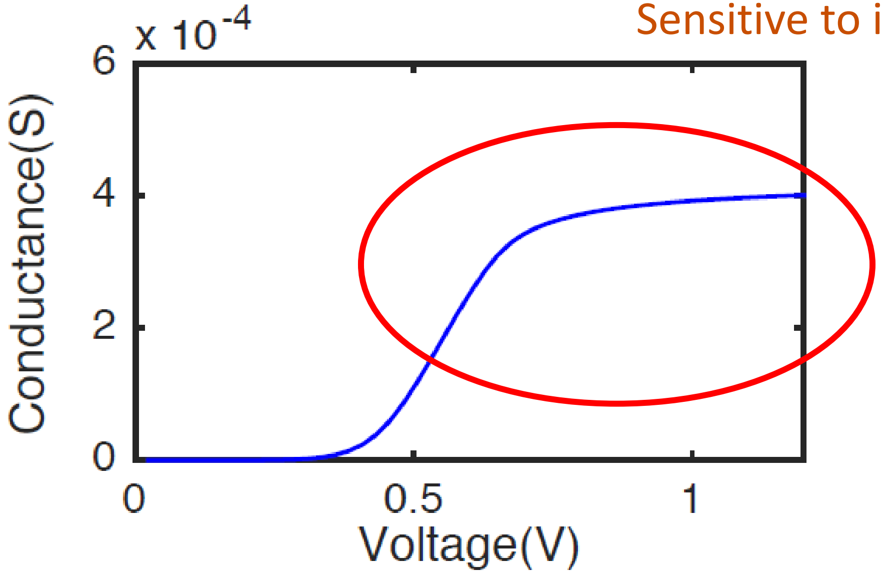
Non-Ideal Memristor - I

- Cell Nonlinearity: conductance varies when applied with different voltages

Ideal Cell



Non-Ideal Cell



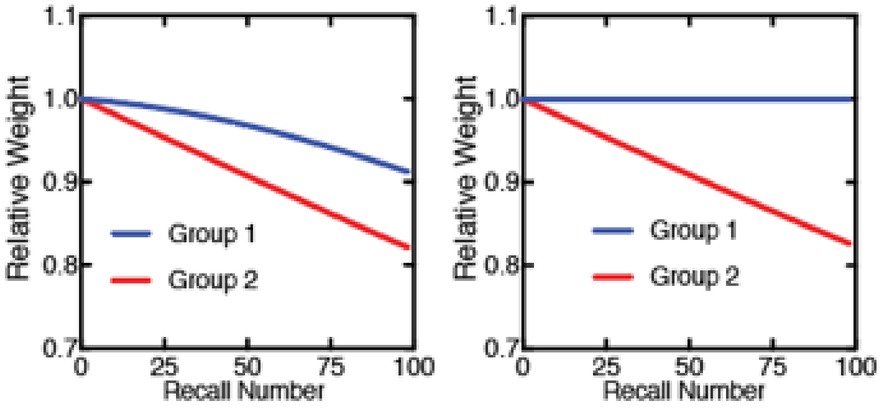
Sensitive to input voltage

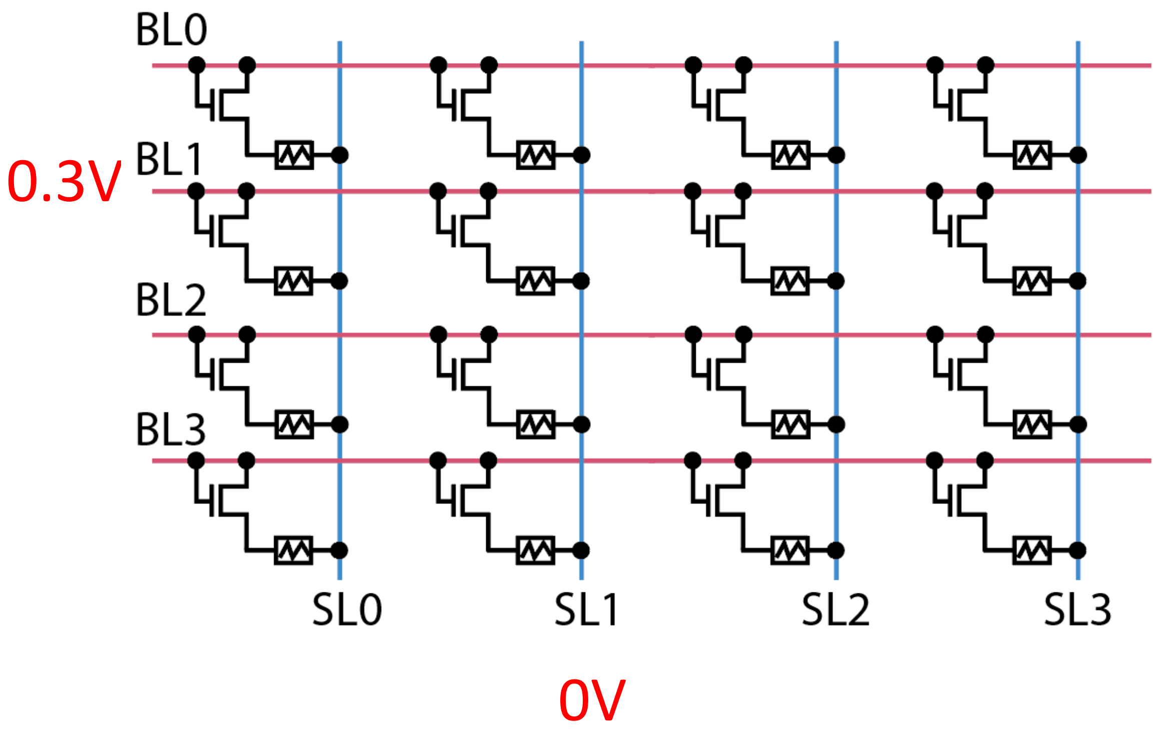
- Solution: Current Amplifier to Clamp Cell Voltage (shown in previous circuit design part)

***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





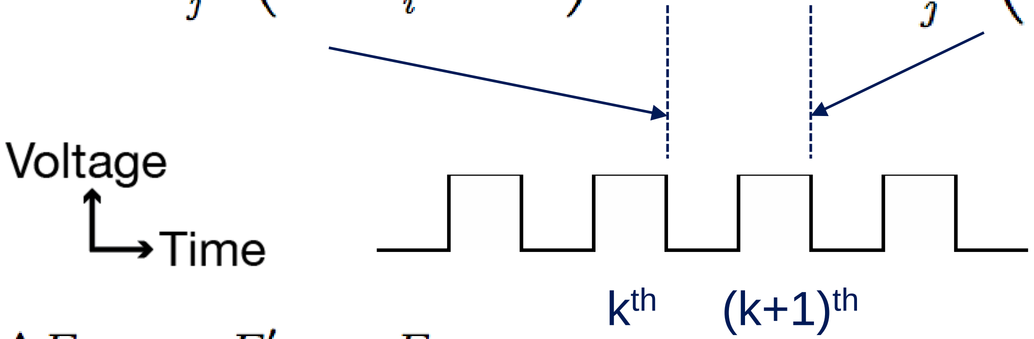
***B. Yan**, et al., ICCAD, 2017*

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

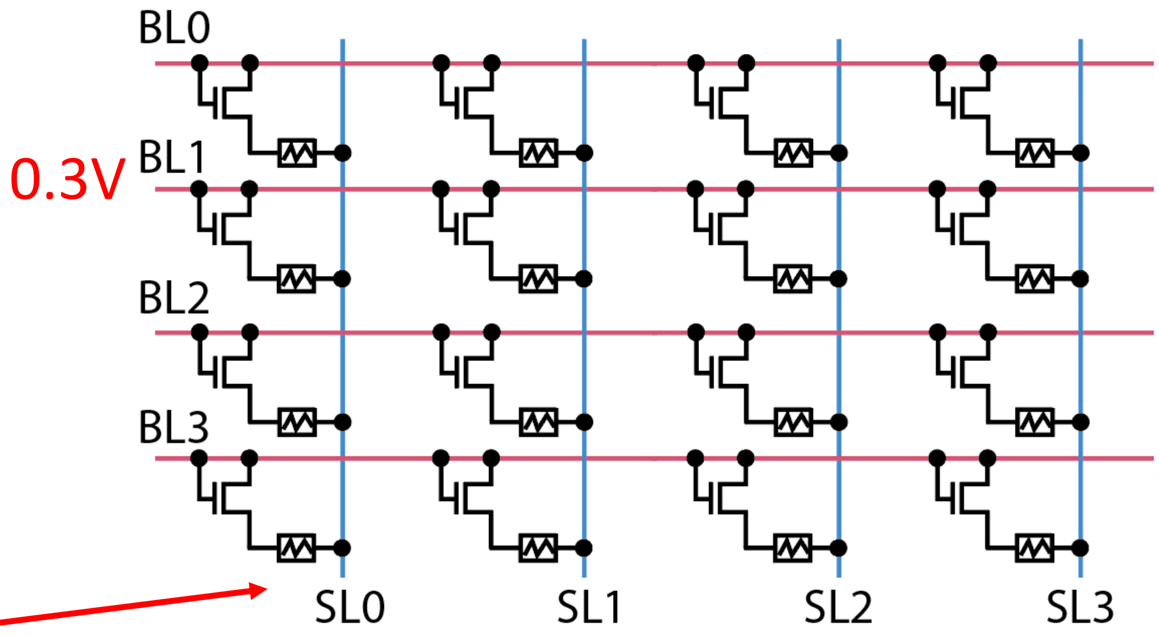


$$\Delta E_{mse} = E'_{mse} - E_{mse}$$

$$= -2 \sum_{j}^{n} \left[\left(t_j - \sum_{i}^{m} w_{ij} x_i \right) \left(\sum_{i}^{m} \Delta w_{ij} x_i \right) \right]$$

Minimize Degraded Error Function:

$$\frac{\partial \Delta E_{mse}}{\partial \Delta w_{ij}} = -2 \underline{\left(t_j - \sum_{i}^{m} w_{ij} x_i \right)} x_i \leq 0$$

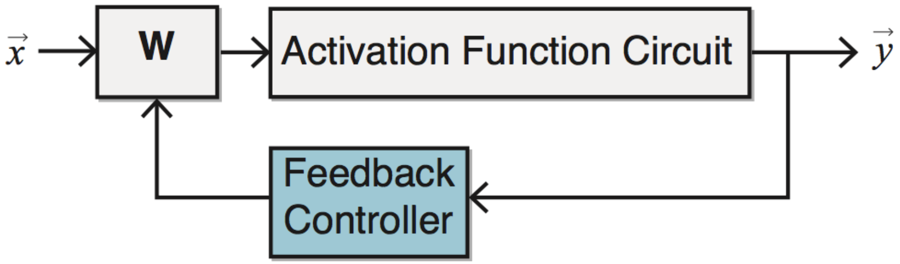


Change Column Current Direction

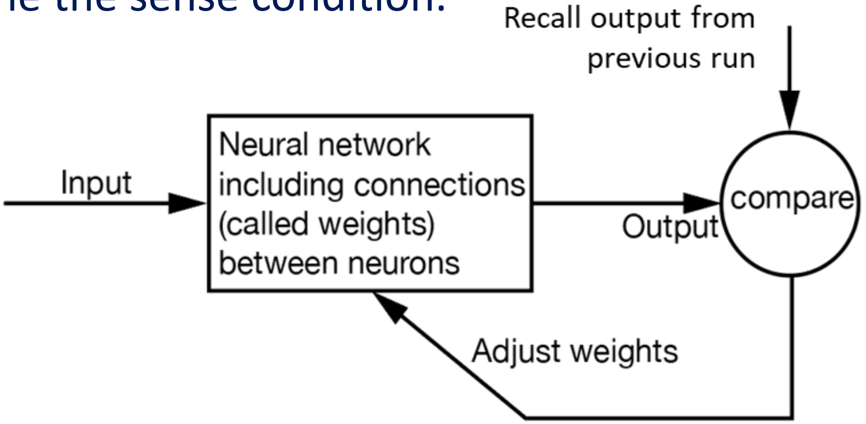
0V/0.6V

Closed-loop Design to Enhance Weight Stability

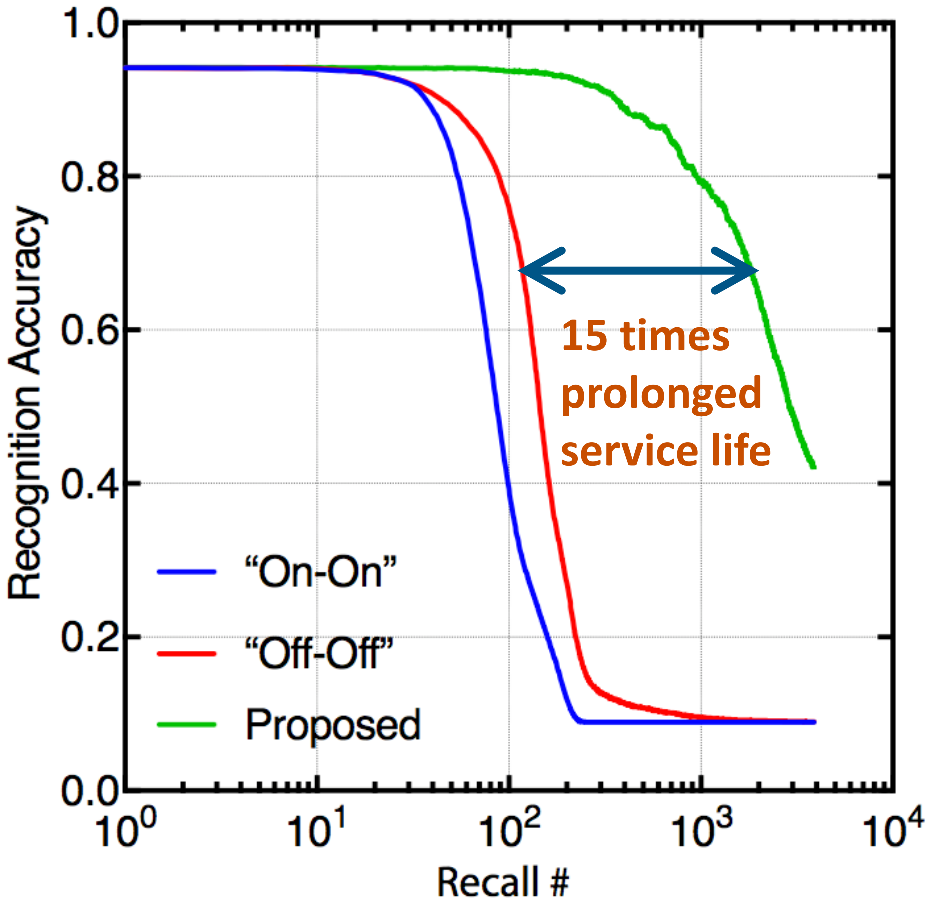
Feedback controller: Adjust the voltage condition to compensate the memristance drift.



"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

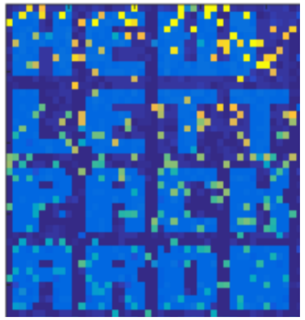


B. Yan, et al, ICCAD, 2017

Duke

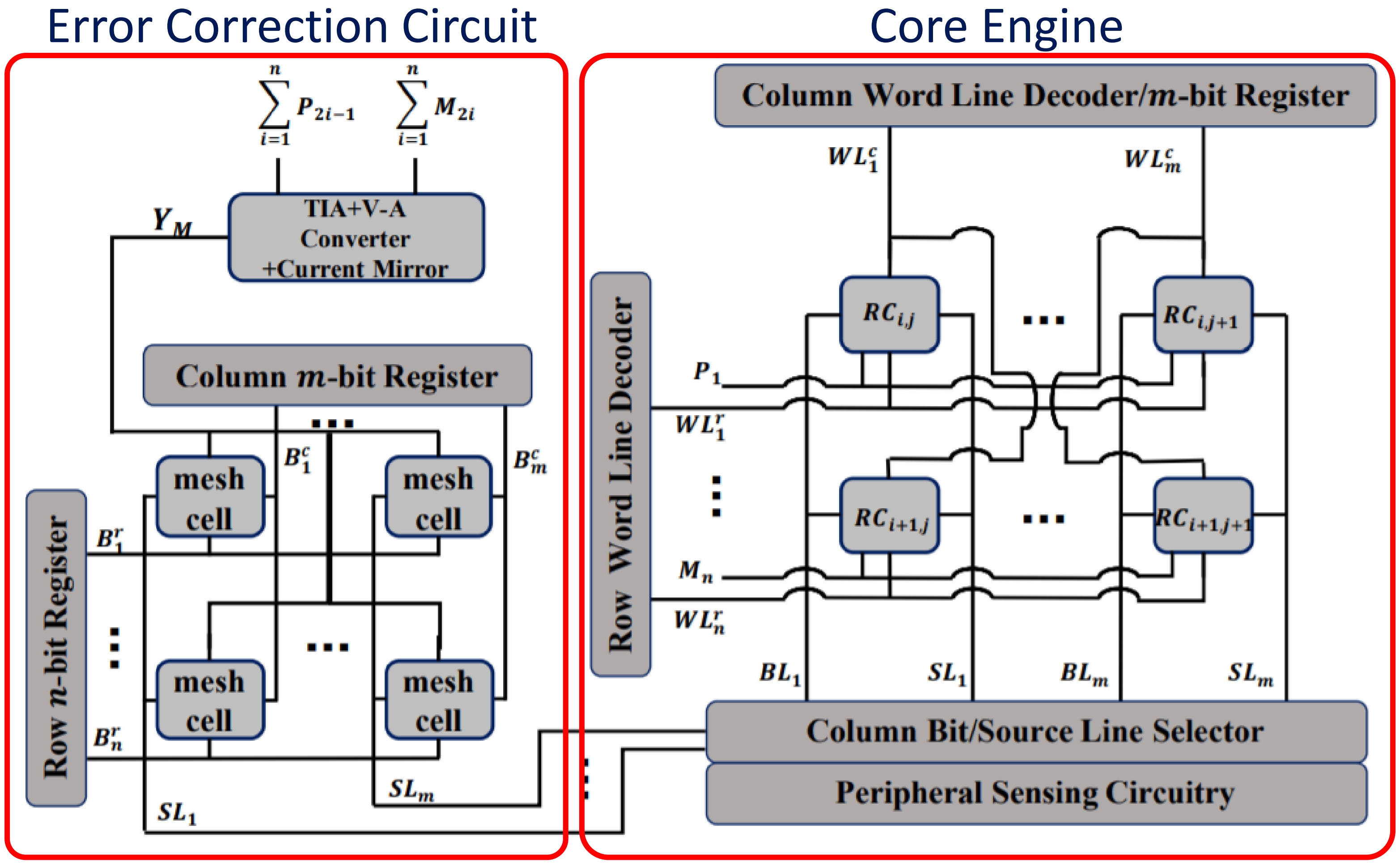
Non-ideal Memristor-III

- Hard Faults: Dead Cells that are not programmable



Noisy "Hewlett Packard", 2016

- Sources of Hard Faults:
  - Fabrication
  - Beyond Endurance
- Our Solution:
  Error Correction Circuits



*A. Chaudhuri, **B. Yan**, et al, ITC, 2019*

Summary

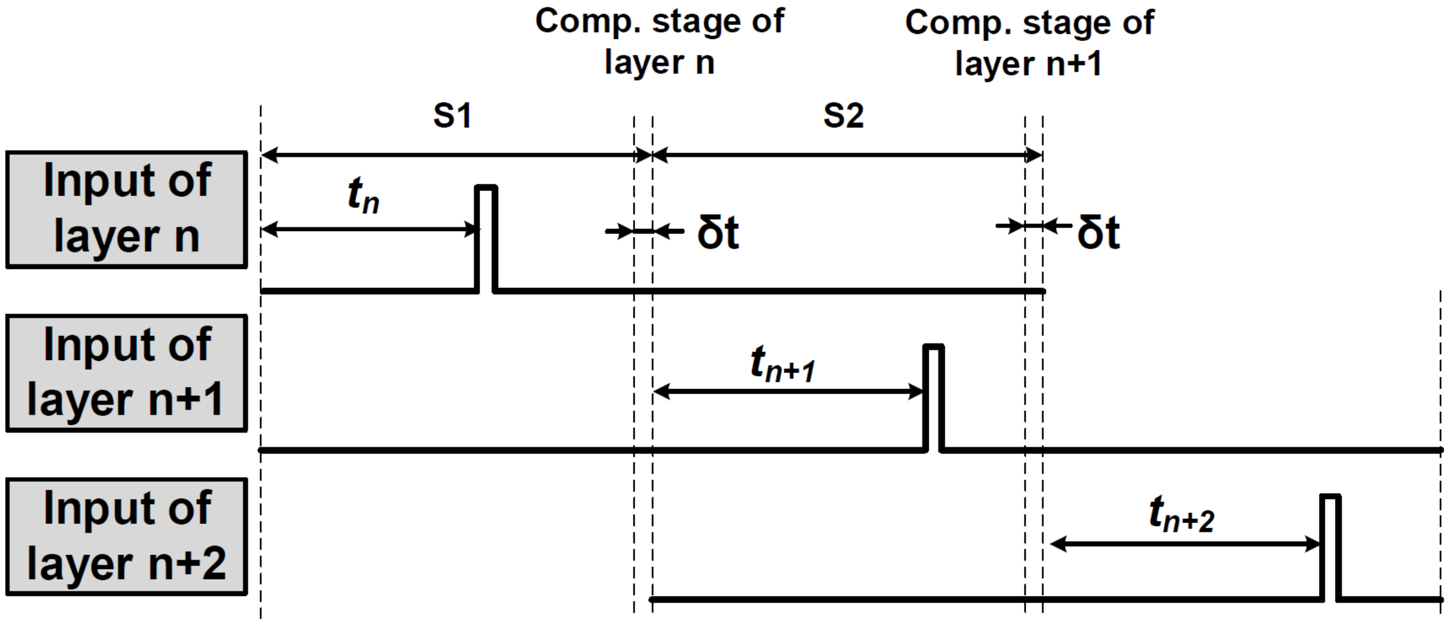
- The Past and Now of AI Hardware
    - In-Memory Computing Eliminates Weights Movement
- My Work:
    - Spiking-based In-Memory Computing Engine Offer Very High Energy Efficiency & Good Performance
    - Clever Design Methodologies (e.g., Closed-Loop Sensing) is Effective to Tolerate Nonideal Features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

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

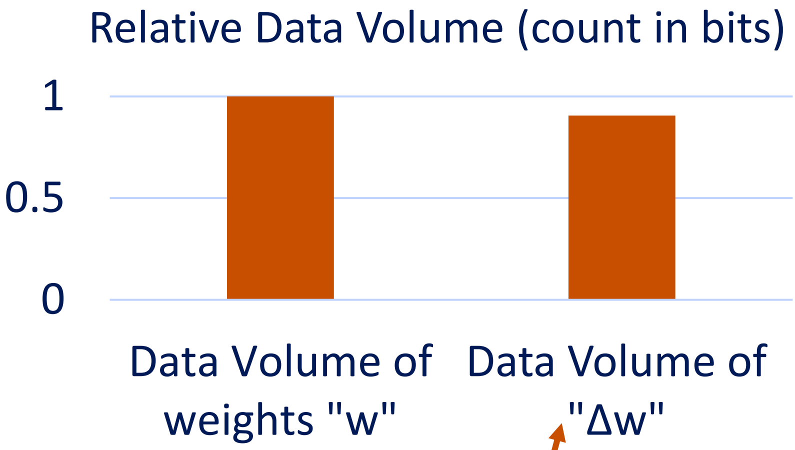
Duke

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

Future Work II: In-Memory Compute & Cache

Challenge:

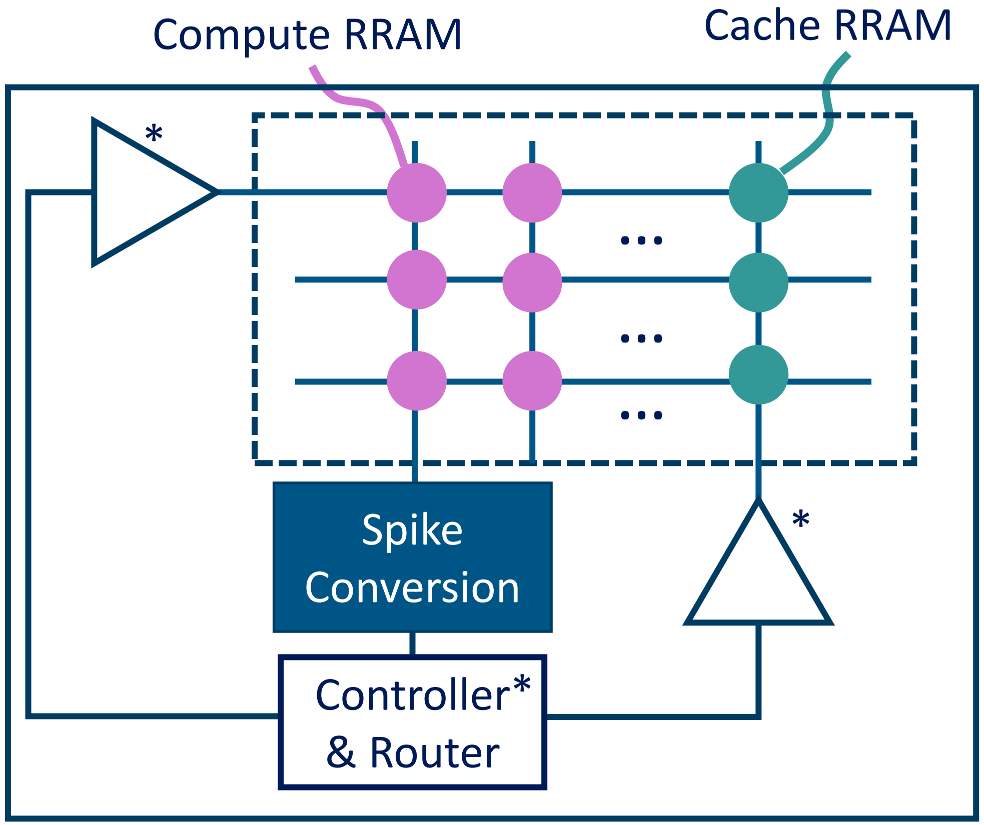
During Training:



No Need to Transport Weight

External DDRAM Access (Expensive!)

Attack:



*: novelty

Prospect Applications:

NN Training Acceleration

Self-Updatable In-Memory Computing Engine

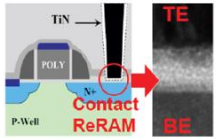
Duke

Long-Term Prospects

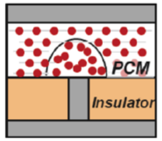
Compute

Control

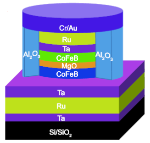
Rethink Compute Memory Hierarchies



RRAM



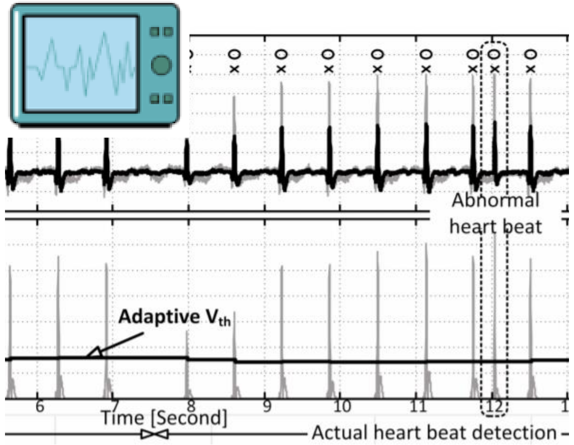
PCM



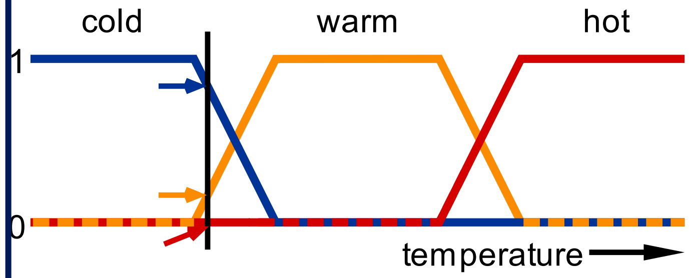
MRAM

“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**

Thanks for Listening & Qs!

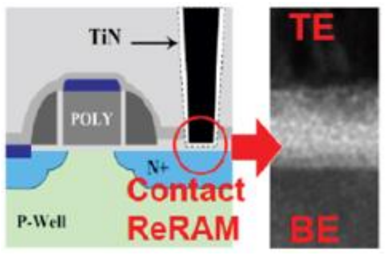
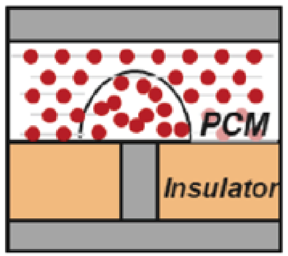
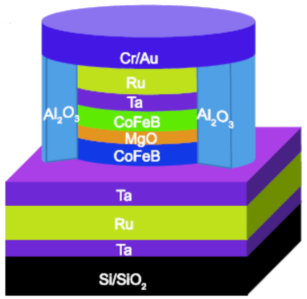
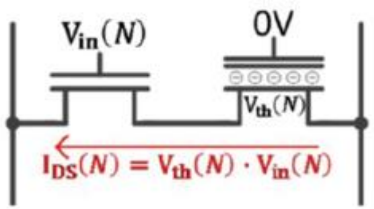
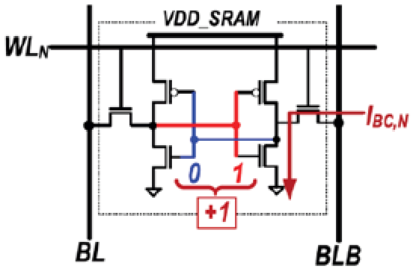
slides available at:

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

Duke

Backup Slides

In-Memory Computing Candidates

RRAM/Memristor	Phase-Change Memory	MRAM	Flash	SRAM
		 Cr/Au Ru Ta CoFeB MgO CoFeB Al_2O_3 Al_2O_3 Ta Ru Ta Si/SiO_2	 $V_{in}(N)$ 0V $V_{th}(N)$ $I_{DS}(N) = V_{th}(N) \cdot V_{in}(N)$	 VDD_SRAM WL_N $I_{BC,N}$ 0 1 +1 BL BLB
Nonvolatile	Nonvolatile	Nonvolatile	Nonvolatile	Volatile
Dense Cell Area	Dense Cell Area	Dense Cell Area	Dense Cell Area	Large Area
Bipolar Write	Unipolar Write	Bipolar Write	Erase-Needed Write	Bipolar Write
Fast Write	Fast Write	Fast Write	Slow Write	Ultra-Fast Write
Multi-level Cell	Multi-level Cell	Single-level Cell	Most Single-level Cell	Single-level Cell

Duke

Zhang et al., JSSC 2017; Fick et al., CICC 2017; Zhang et al., T-ED 2015
Chen et al., ISSCC 2018; Boybat et al., Nature communications 2018

Comparison

Type		CIM Macro				CIM PE		Digital Processor
Work		VLSI’18 Panasonic	ISSCC’18 NTHU	ISSCC’18 NTHU	ISSCC’18 MIT	This work[†]	ISSCC’18 UIUC	TrueNorth [10]
Technology		180nm	65nm	65nm	65nm	150nm	65nm	28nm
Synapse		1T1R RRAM	1T1R RRAM	6T-SRAM	10T-SRAM	1T1R RRAM	6T-SRAM	SRAM
Nonvolatility		Yes	Yes	No	No	Yes	No	No
Standby current		~zero	~zero	high	high	~zero	high	high
Spiking NN		No	No	No	No	Yes	No	Yes
Capacity		2M	1M	4K	16K	64K	128K	256M
Cell area [F^2]		—	59	124	968	74	~256	—
Normalized die area		12×	—	—	30×	1×	11×	~17240×
Chip power [mW]		15.8	—	—	—	1.52	—	204.4
Activation precision		1 bit	3 bit	1 bit	7 bit	1~8 bit	8 bit	1 bit
Power efficiency [TOPS/W]	MAC only	20.7	16.95	55.8	28.1	—	—	—
	MAC +Activation	—	—	—	—	16.9	3.125	0.4
On-chip Activation Function Integration		No	No	No	No	Yes (tanh)	Yes (relu)	Yes (relu)
FoM[*]		—	0.86	0.45	0.20	1.83	0.098	—

FoM = energy efficiency maximum activation precision/cell area.

***B. Yan**, et al., VLSI Symp. 2019*