Memristor based computation-in-memory Architectures for Edge AI

Opportunities and challenges

Said Hamdioui

Head of Quantum and Computer Engineering department
Delft University of Technology & Cognitive-IC
The Netherlands
S.Hamdioui@tudelft.nl

TUDelft

Cognitive IC
Feel - Think – Act

MIM WEBINARS

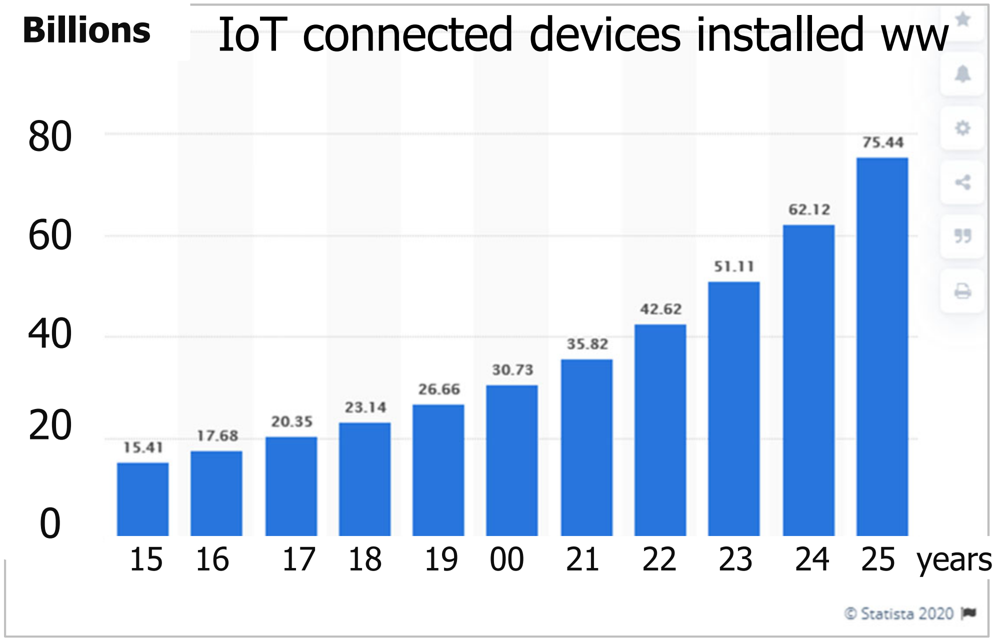
AN IN-MEMORY COMPUTING SERIES

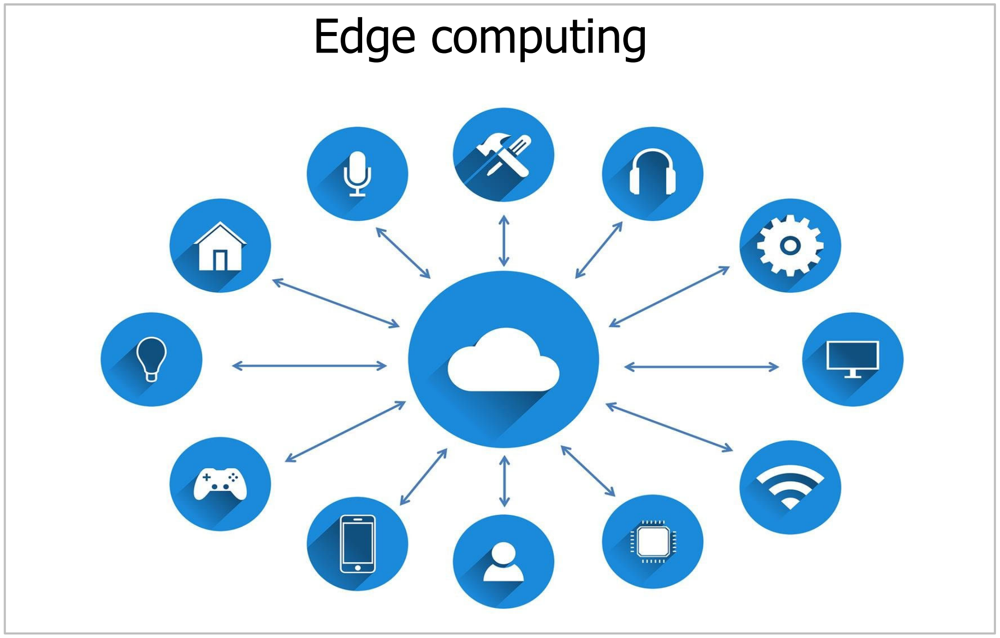
Next Talk: 17/May/2021, 4-5:30pm CET

Outline

- The opportunity and the challenges
  - IoT-edge partnership, HW challenges
- Computer architecture
  - Past and future & classification
- Computation-in-Memory CIM
  - Basics and classification
- CIM circuit design
  - Logic operations, Vector-matric multiplication
- CIM Potential
  - Design flow, application domains, potential improvements
- Challenges
  - The open questions
- Conclusion

The opportunity: IoT-edge partnership

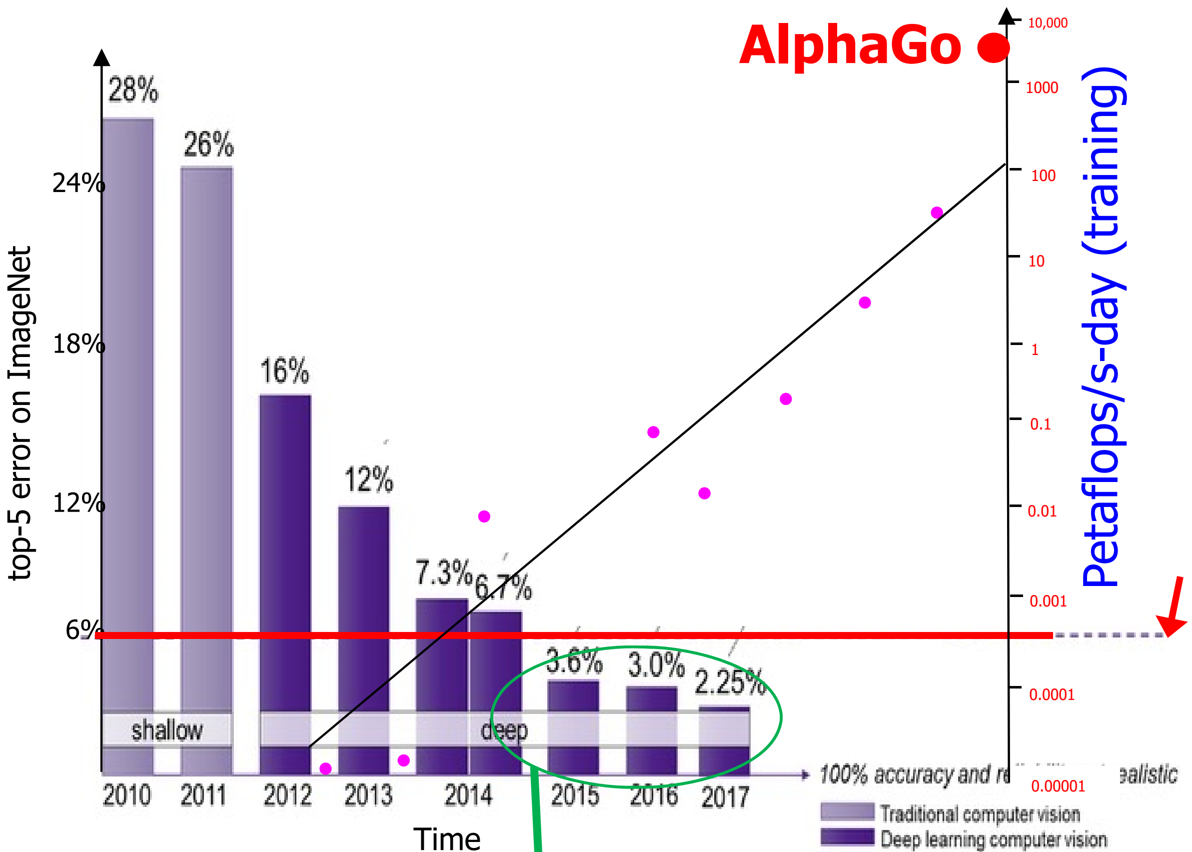




Many requirements

- **Intelligence**
- **Energy constraints**
- Local computing
- Data privacy
- Real-time decisions
- 24/7

The challenges: Intelligence



In April 2017, AlphaGo vs. Jie Ke

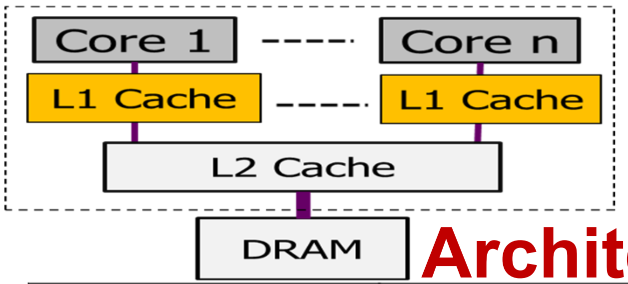
AlphaGo:
- 176 GPUs, 1202 CPUs
- 150, 000 Watts

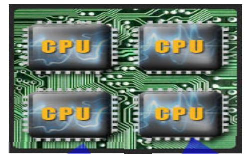
AI **surpassed** human

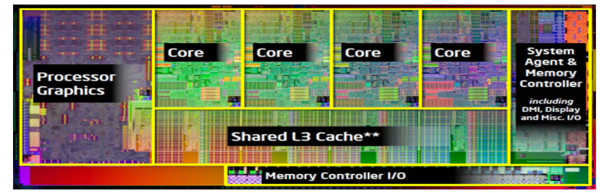
Not suitable for Edge Application

AI requires **huge** resources

The challenges: The HW architecture and technology







Architecture

Operation @32-bit operand	**Energy (pJ)** @TSMC 45nm	**Cost** v ALU
ALU: Add	0.1	**1x**
SRAM: read	5	50x
DRAM: read	640	**6,400x**

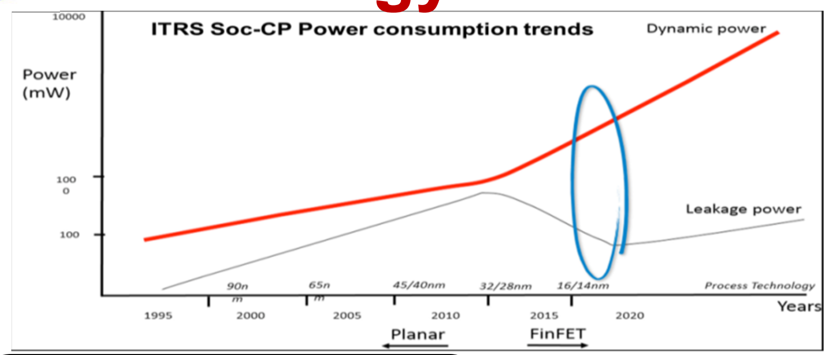
[Source: M. Homowitz et el., ISSCC, 2014]

1. **Memory Wall**
2. ILP Wall
3. Power Wall

[Source: D. Patterson, future of computer Architecture, 2006]

*Need of new **Architectures***

Technology



[Source: S. Hamdioui, et.al, DATE 2017]

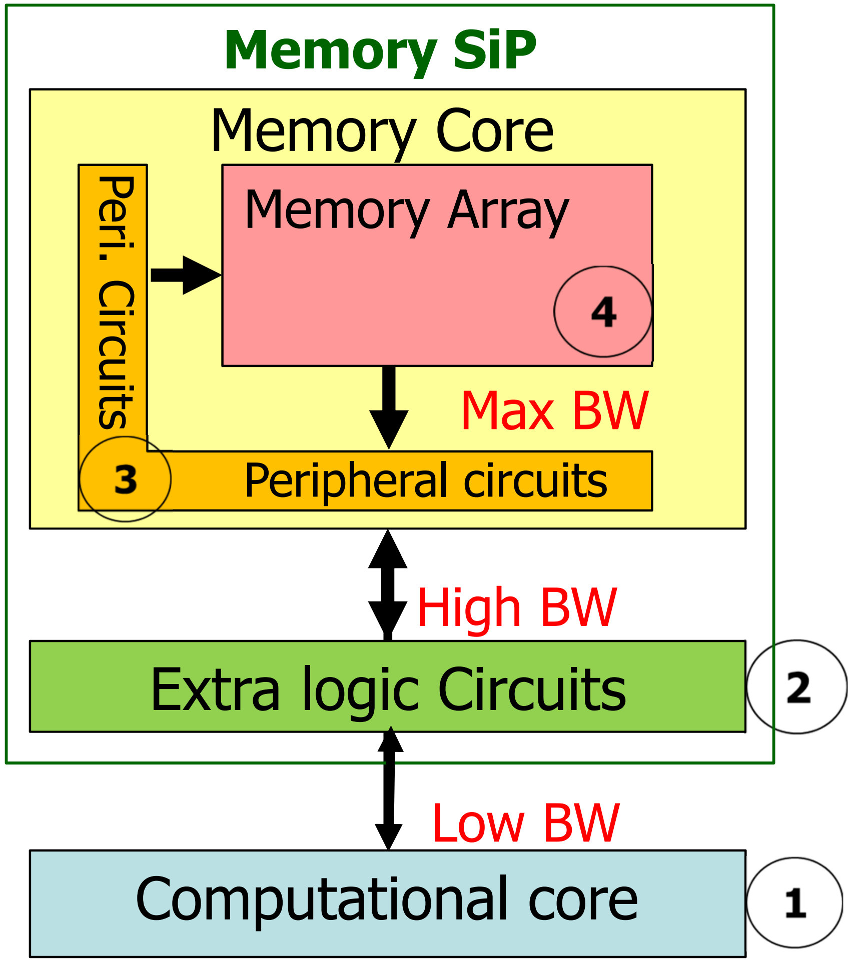
1. **Leakage Wall**
2. Cost Wall
3. Reliability Wall

*Need of new **Technologies***

Need of Unconventional architectures using **unconventional technologies**

Computer Architectures: Classification

- **COM: Computation-Out-Memory**
  1. Far (COM-F)
  2. Near (COM-N)
- **CIM: Computation-In-Memory**
  3. **Periphery (CIM-P)**
  4. Array (CIM-A)
- **Hybrid architectures**
- **Status**
  - COM: commercialized & conv technologies
  - CIM: Research, conv & unconv technologies

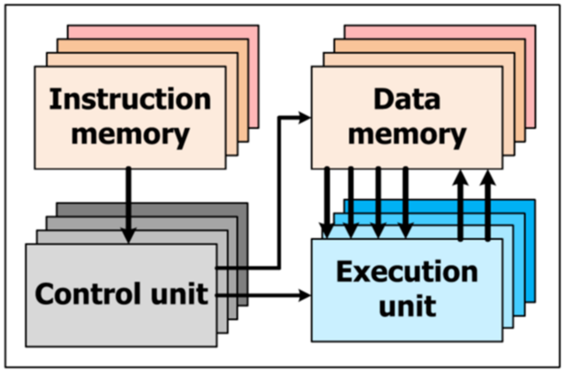


[Source: H.A. Du Nguyen, et. al, "A classification of memory-centric computing" ACM JETC, 16(2), pp.1-26", 2020]

Computer Architectures: Comparison

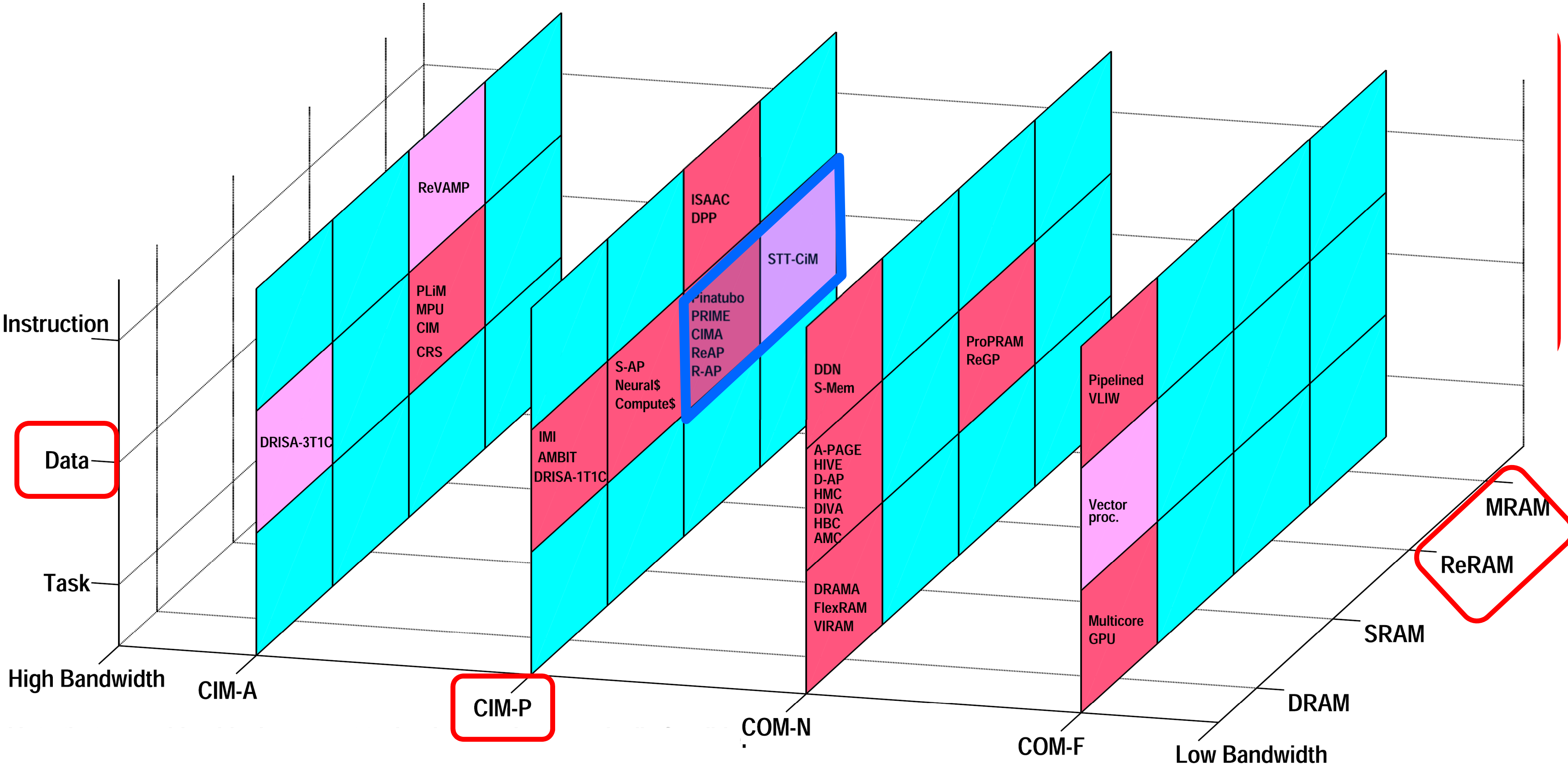
Type	Off-chip Data mov.	Computations reqrs.		Available Band-width	Memory design effort			Scala-bility
		D. align	Cmplex.Fu		Array	Periphery	controller	
CIM-A	No*	Yes	H. latency	Max	High	Low/ Medium	High	Low
CIM-P	No*	Yes	H. Cost	High-Max	Low-medium	High	Medium	Medium
COM-N	Yes	NR	Low cost	High	Low	Low	Low	Medium
COM-F	Yes	NR	Low cost	Low	Low	Low	Low	High

- **Other attributes**
  - Endurance requirements



- **Other metrics classification**
  - **Memory Technology**: SRAM, DRAM, MRAM, ReRAM, …
  - **Computation parallelism**: **Task** (e.g., multi-core), Data (e.g., SIMD), Instruction (e.g., VLIW)

Computer Architectures: overview

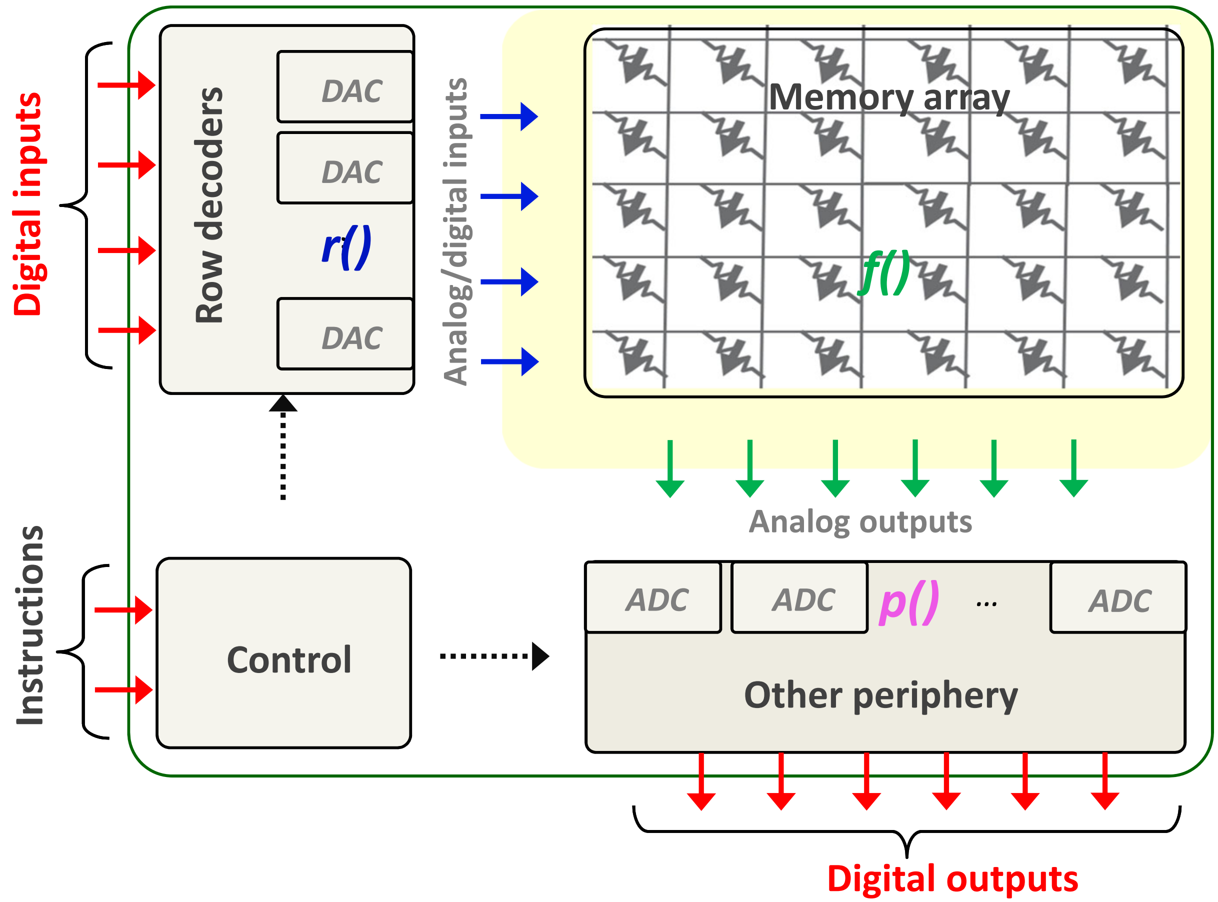


[Source: H.A. Du Nguyen, et. al, "A classification of memory-centric computing" ACM JETC, 16(2), pp.1-26", 2020]

Computation-in-Memory: Basics

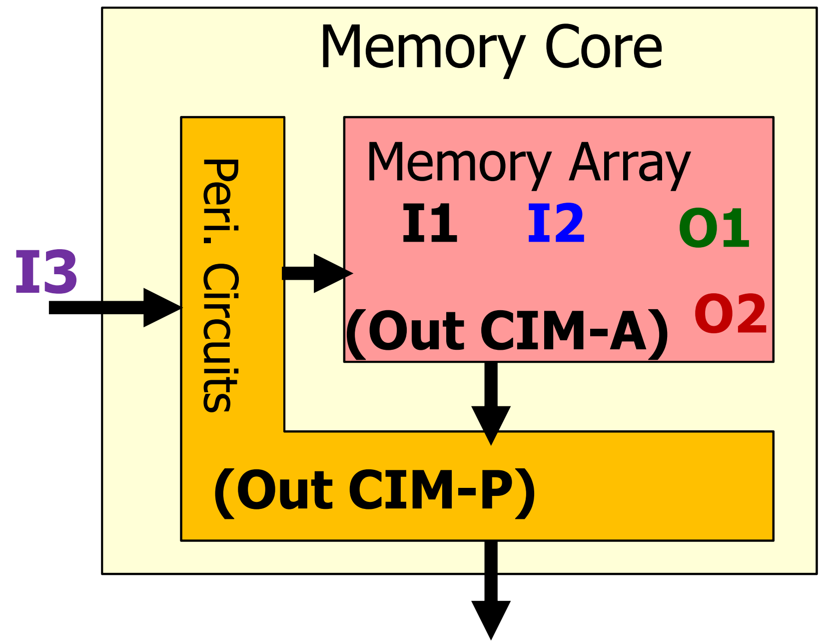
- Single or multi functions
  - **r()**
  - **r()** & f() ?
  - **p()** & f()?
  - **r() & p()** & f()
  - Etc.

- CIM-P
  - **Major** changes in periphery
    - p() and/or r()
  - Basic: no changes in array
  - Hybrid: some changes in array



Computation-in-Memory: Classification

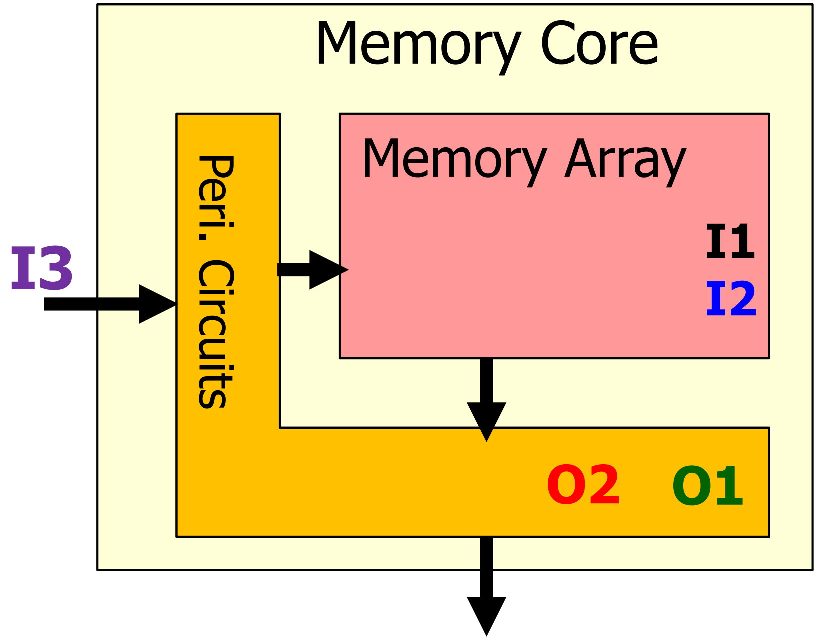
CIM-A Output in **Array**		**CIM-P** Output in **Periphery**
CIM-**Ar** Input: All **resistive**	CIM-**Ah** Input: Hybrid	



- Example
  - CIM-Ar: **O1**=**I1** OR **I2**
  - CIM-Ah: **O2**=**I2** OR **I3**

Computation-in-Memory: Classification

CIM-A Output in **array**		**CIM-P** Output in **Periphery**	
CIM-**Ar** Input: All **resistive**	CIM-**Ah** Input: Hybrid	CIM-**Pr** Input: All **resistive**	CIM-**Ph** Input: Hybrid



- Example
  - CIM-**Pr**: **O1**=**I1** OR **I2**
  - CIM-Ah: **O2**=**I2** OR **I3**

Computation-in-Memory: Classification

CIM-A Output in **array**		**CIM-P** Output in **Periphery**	
CIM-**Ar** Input: All **resistive**	CIM-**Ah** Input: Hybrid	CIM-**Pr** Input: All **resistive**	CIM-**Ph** Input: Hybrid
SNIDER [2005] e.g., NAND **IMPLY[2010]** e.g**.,** IMP, NAND **MAGIC [2014]** e.g,, NOT, NOR **Fast Boolean** [2015]	Resi. Accu. [2003] **Majority Logic** [2016]	**Pinatubo** [2016] OR, AND, XOR **Scouting** [2017] OR, AND, XOR	**VMM** [2016, 2017] **BCMM** V[2017] **BDP** [2018**]**

Recent work mainly on CIM-P?

[Source: M. Abu Lebdeh, et. al, "Memristive device based circuits for computation-in-memory architectures", ISCAS 2019]

CIM circuit design: CIM-Ar

- **Snider logic**
  - Primitive logic operations: NAND, INV
  - 2 Control voltages: Vw, Vh
    - Vw> Vth > Vh
    - Vw – Vh < Vth

- **Working principal** (e.g., 2NAND)

  1. Store
  1. Program *p* to *Ron*
  2. Program *q* to *Roff*

  2. Compute
  1. Initialize *f* to ***Roff***
  2. Process NAND
  - Vx≈ Vh; Vw-Vx= Vw-Vh<Vth

  => f= Roff

Suitable for crossbar -> density

Requires multiple accesses

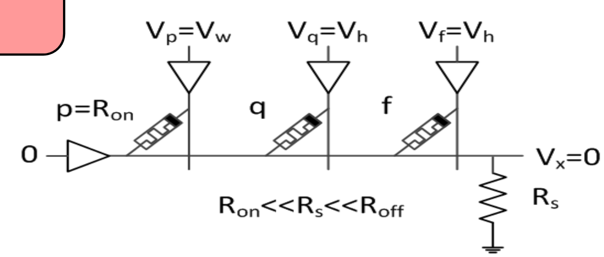
Requires array redesign

Logic states:

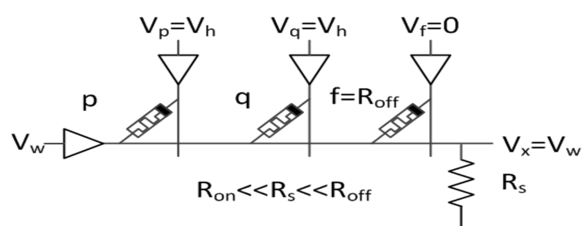
In/out: Roff=1;Ron=0

p	q	f
0	0	1
0	**1**	**1**
1	0	1
1	1	0

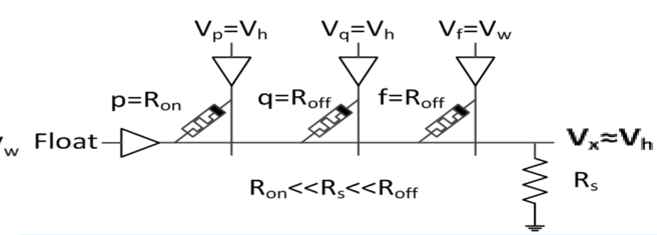
Program p to Ron



Initialize f to Roff



Process NAND



[Source: Snider et.al, APA, 2005; Xie et al. ICCD, 2015]

CIM circuit design: CIM-Ah

- **Majority logic** [Gaillordon et.al, DATE, 2016]
  - Logic operation: Majority Function
  - Inputs: p, q, z
  - Out: z = MAJ $(p,\bar{q},z)$
  - 2 Control voltages: Vw, GND
    - Vw> Vth

Logic states:

- **In p & q**: Vw=1; GND=0
- **In z**: Ron=1; Roff=0
- **Out**: Ron=1; Roff=0

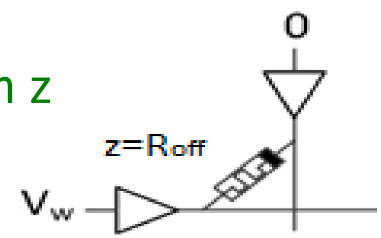
+ Suitable for crossbar
- Needs data movement (p,q)
- Parallelism?

p	q	z	MAJ$(p,\bar{q},z)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	**0**	**0**	**1**
1	0	1	1
1	1	0	0
1	1	1	1

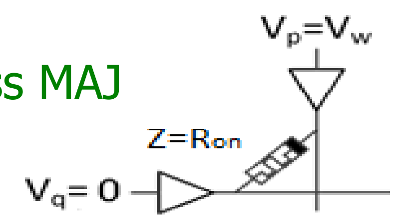
- **Working principal**
  1. Program z to Roff
  2. Process MAJ
     - Apply Vp=Vw & Vq=GND
     - Vw - 0= Vw > Vth

  => z= Ron

Program z



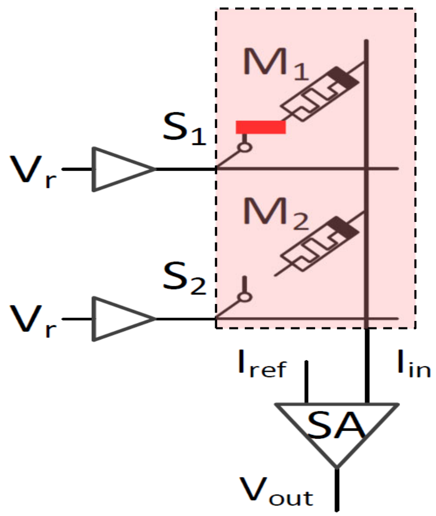
Process MAJ

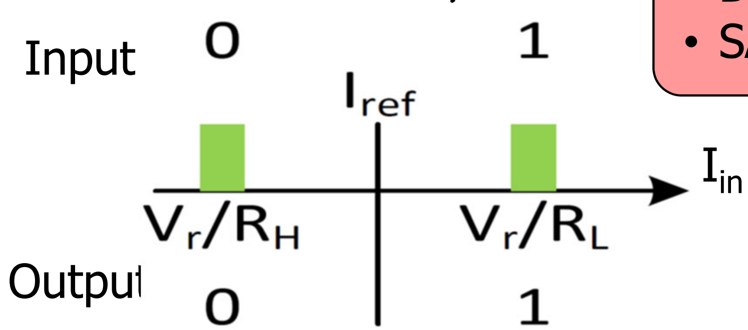


CIM circuit design: CIM-Pr example

- **Scouting Logic**

Read a memory cell





Parallelism
-> performance

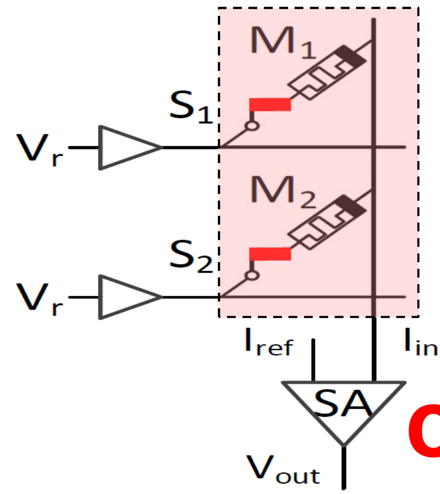
Operations
1 cycle: E efficiency

Reduced com.
-> E efficiency

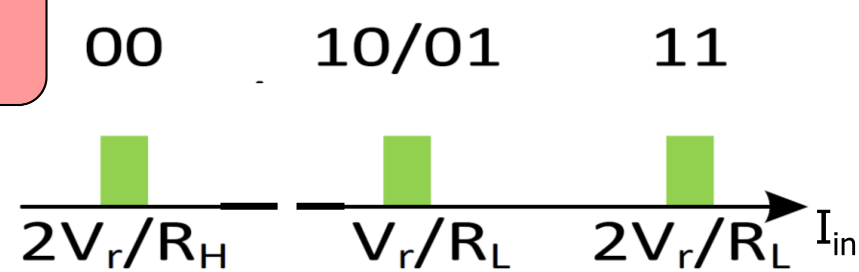
Read intensive
Endurance

- Data alignment
- SA & AD medication

Read & operate on two cells



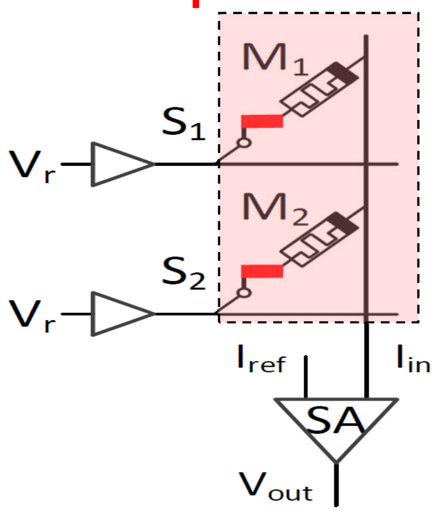
OR operation



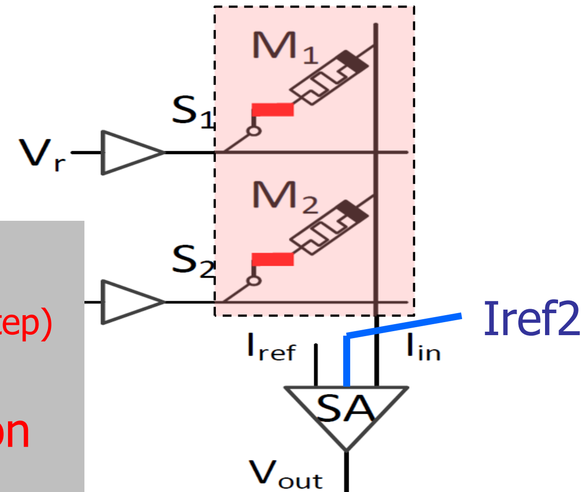
[Source. L. Xie, et.al, "Scouting Logic: A Novel Memristor-Based Logic Design for Resistive Computing", ISVLSI 2017]

CIM circuit design: CIM-Pr example

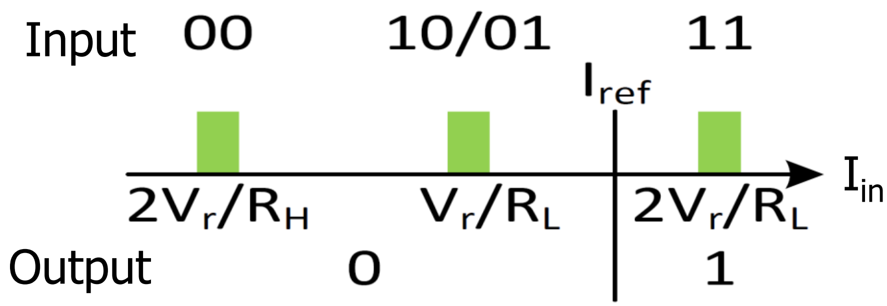
Read & operate on two cells



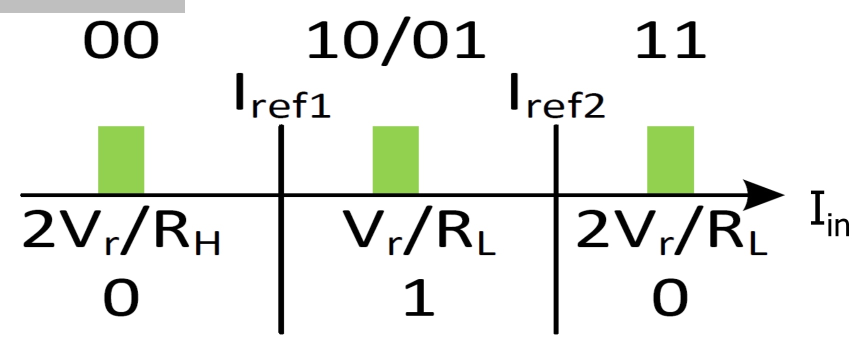
Read & operate on two cells



- \+ Less requirements on endurance
- \+ Once access per operation (single step)
- \+ No major changes in array
- \- Modification of SA and AD selection
- \- Data alignment



AND operation



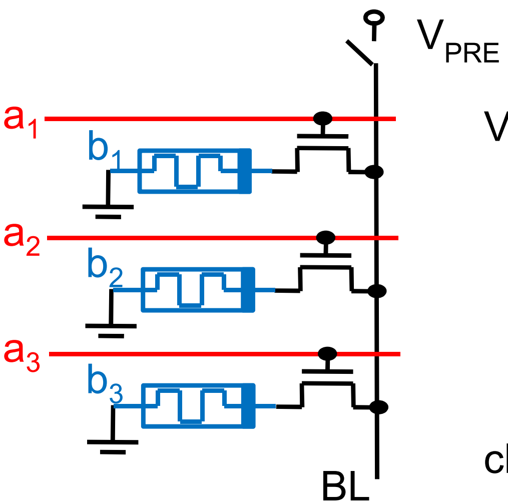
XOR operation

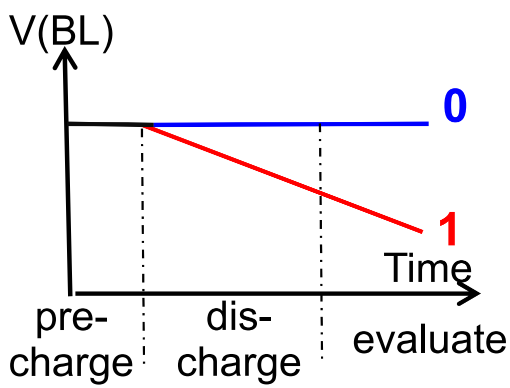
CIM circuit design: CIM-Ph example

Vector dot product: *a.b*

- ***a.b*** = $a_1.b_1 + a_2.b_2 + a_3.b_3$
- Inputs: voltage and resistance
- Output: voltage
- Result: **OR** $= \sum_{i}^{3}$
- Complexity: **O(1)**

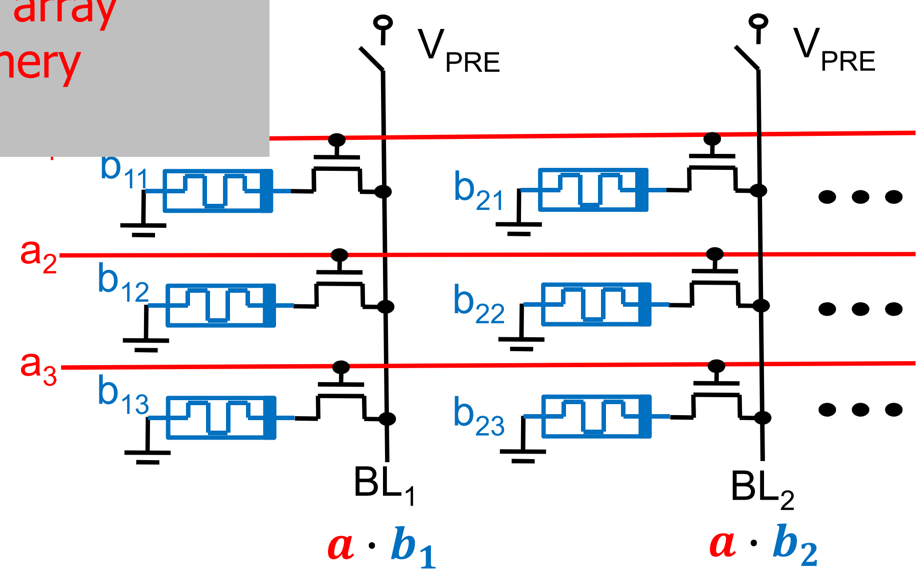
> \+ Less requirements on endurance
> \+ Reduced computing complexity
> \+ No major changes in array
> \- Modification of Periphery
> \- Data alignment





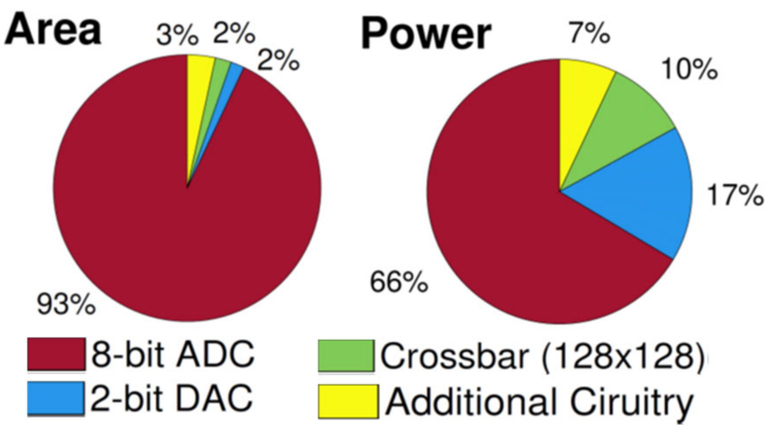
Vector matrix multiplication

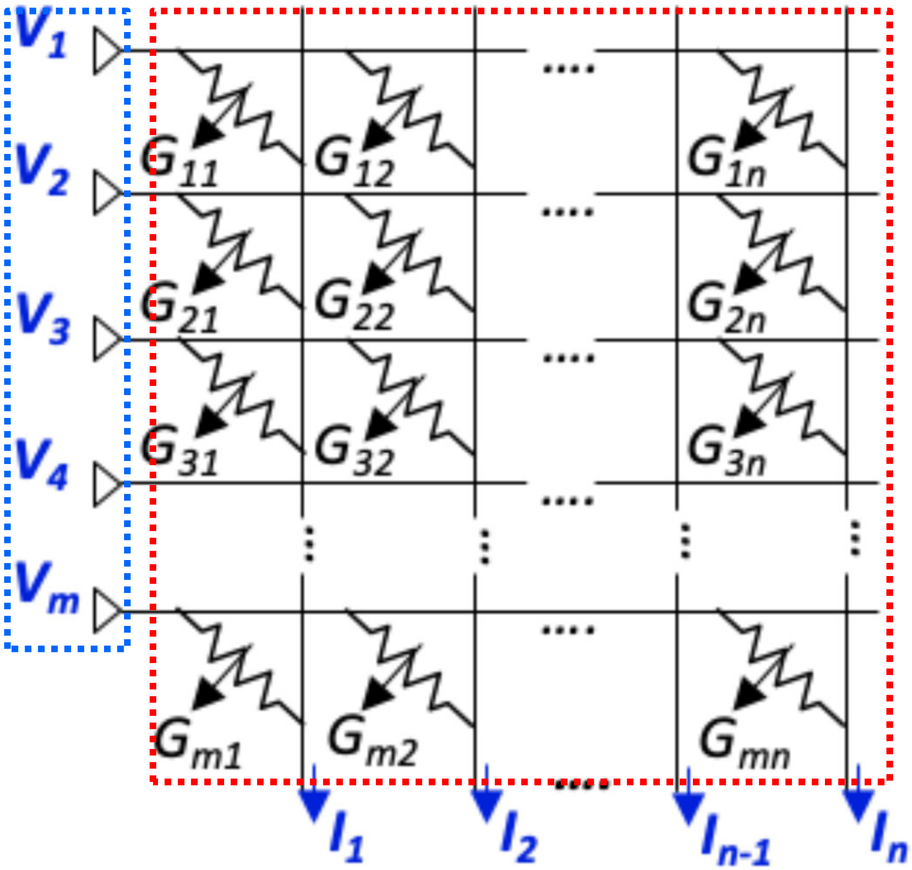
- **v**=$\boldsymbol{a}.\boldsymbol{B} = [\boldsymbol{a}\cdot\boldsymbol{b_1}, \boldsymbol{a}\cdot\boldsymbol{b_2}, \ldots, \boldsymbol{a}\cdot\boldsymbol{b_N}]$
- Extendable to matrix x matrix
- Complexity
  - **O(1):** vector * matrix
  - **O(n)**: matrix * matric



CIM circuit design: CIM-Ph

- **Multiply-Accumulate**
  - $I_i = \sum V_j . G_{ij}$
  - $I_1 = V_1.G_{11} + V_2.G_{21} + \ldots V_m.G_{m1}$
  - Complexity: **O(1)**





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

CIM circuit design: ADC design / classification

Sensing Stage
- Capacitor
- Resistor
- …

Conversion Stage
- Compare
- Integrate & Fire
- Time-digital conv.

Decision Stage(s)
- Counter
- ……

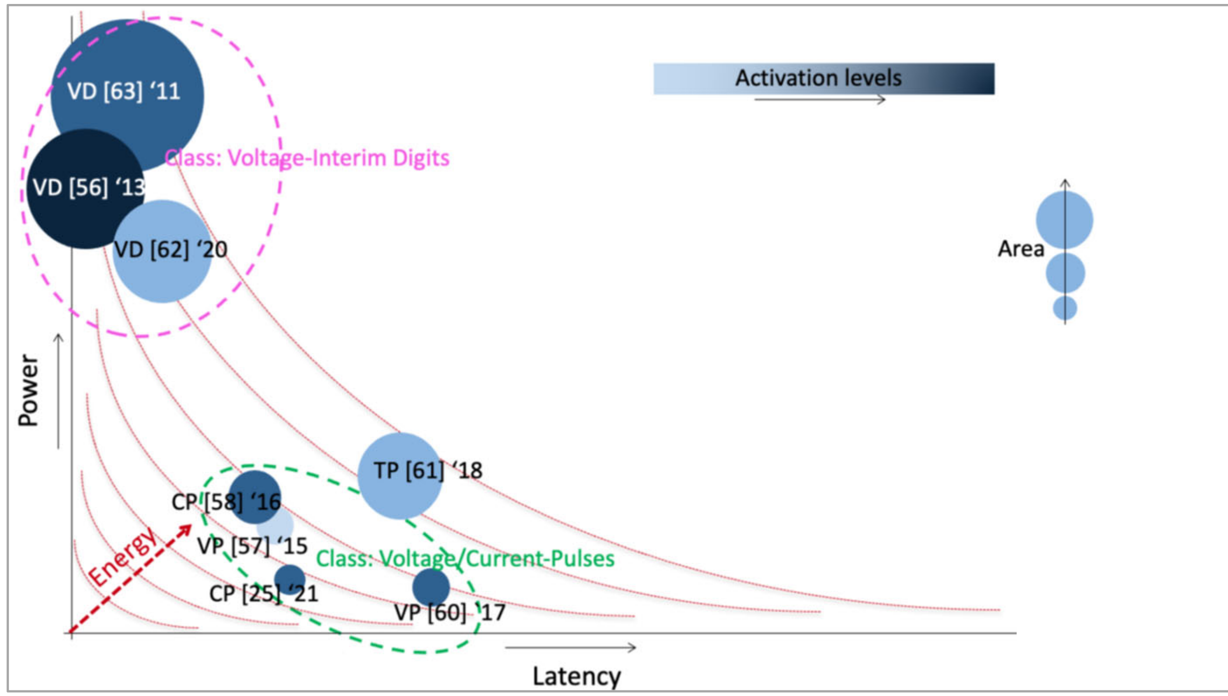
Input
Analog data
I_{column}

Intermediate analog data
Voltage, Current, Time

Intermediate discrete data
Pulses, Interim (1010..)

Output
E.g. Digital bit-stream
010….

- Six classes: (Voltage, Current, Time) x (Pulse, Interim)



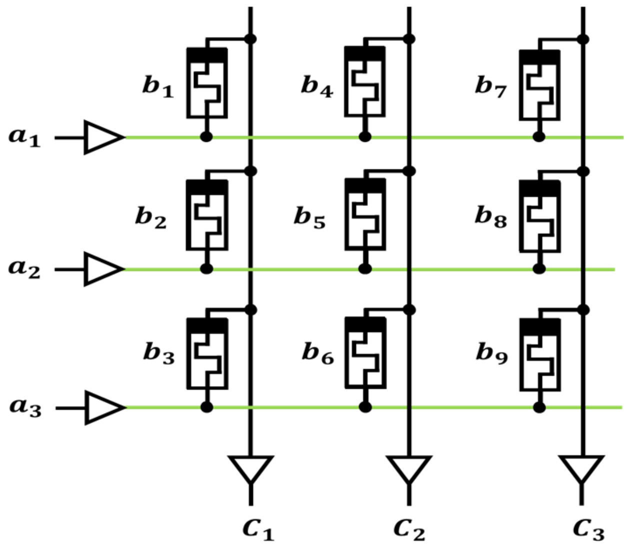
Major challenge: appropriate accuracy at low energy & high speed

[Source: A. Signh , et. al, "SRIF: Scalable and reliable integrate and fire circuit ADC for memristor-based CIM architectures" , TCAS-1, 2021]
[Source: A. Signh , et. al, "Low Power Memristor-based Computing for Edge-AI Applications" , ISCAS 2021]

CIM circuit design: CIM-Ph

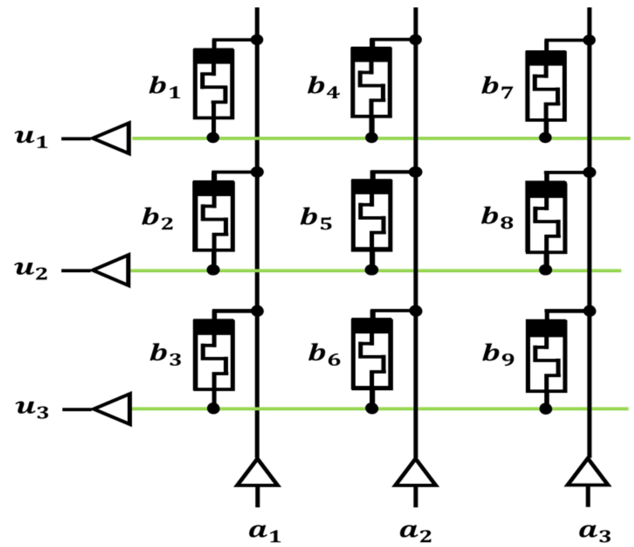
Non-Transposed B Matrix

$$[a_1 \quad a_2 \quad a_3]\begin{bmatrix} b_1 & b_4 & b_7 \\ b_2 & b_5 & b_8 \\ b_3 & b_6 & b_9 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$



Transposed B Matrix

$$[a_1 \quad a_2 \quad a_3]\begin{bmatrix} b_1 & b_4 & b_7 \\ b_2 & b_5 & b_8 \\ b_3 & b_6 & b_9 \end{bmatrix}^T = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$



Can configurable periphery enable different functions? At what cost?

[Ref: M. Le Gallo, et al. "Compressed Sensing Recovery using Computational Memory", IEDM, 2017]

Computation-in-Memory: CIM-A v CIMP designs

Shmoo plot for IMPLY (CIM-Ar)

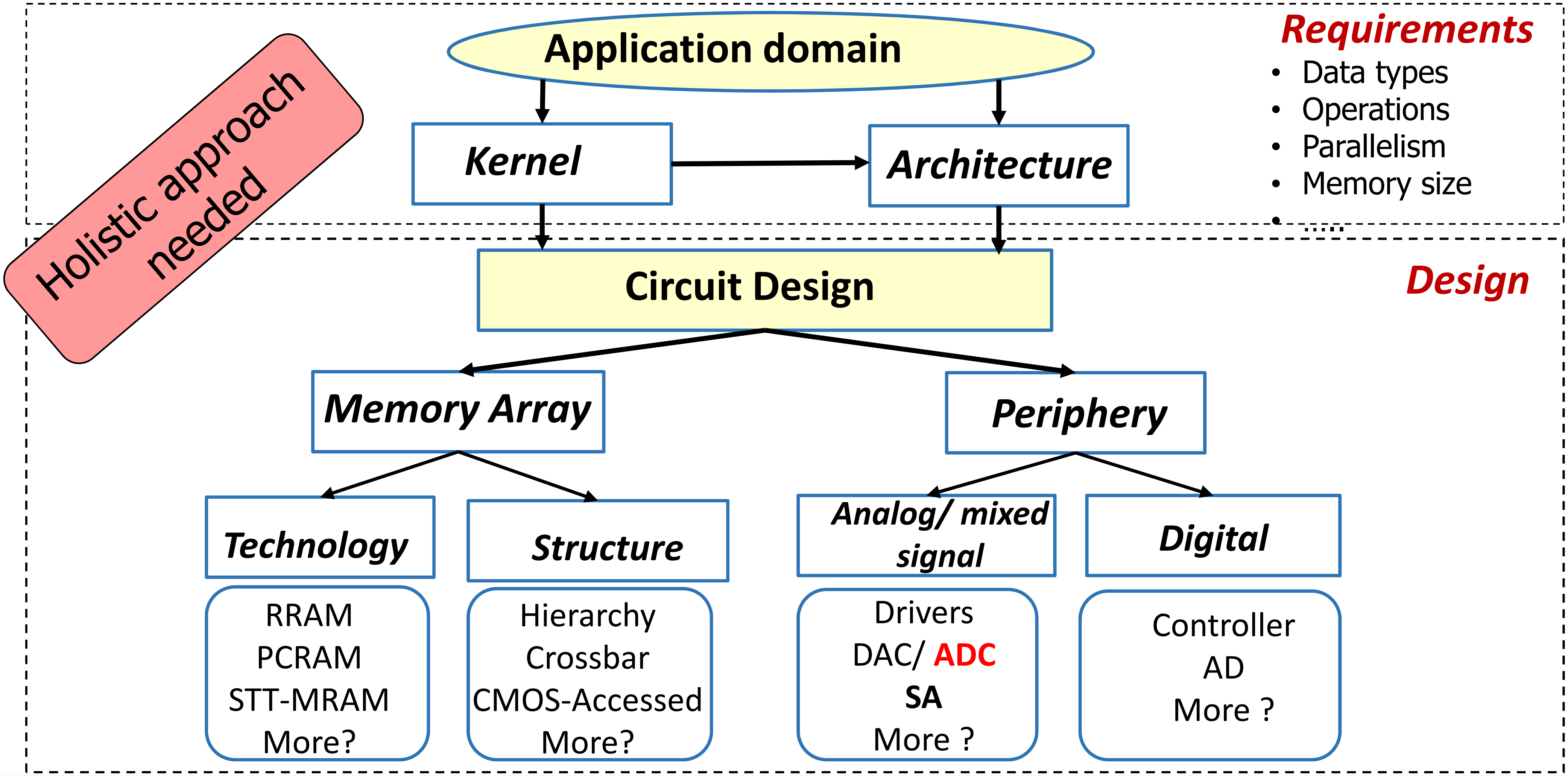
Voltage \ $\frac{R_{off}}{R_{on}}$	10	50	100	200	500	1000
1.3	F	F	F	F	F	F
1.4	F	F	F	F	F	F
1.5	F	F	F	F	F	F
1.6	F	F	F	F	F	F
1.7	P (2%)	P (5%)	P (35%)	P (20%)	P (2%)	F
1.8	F	F	P (15%)	P (35%)	P (20%)	P (3%)
1.9	F	F	P (3%)	P (35%)	P (30%)	P (15%)
2	F	F	F	P (20%)	P (40%)	P (25%)
2.1	F	F	F	P (3%)	P (40%)	P (35%)
2.2	F	F	F	F	P (30%)	P (45%)
2.3	F	F	F	F	P (20%)	P (40%)
2.4	F	F	F	F	P (3%)	P (40%)
2.5	F	F	F	F	P (1%)	P (30%)
2.6	F	F	F	F	F	P (20%)
2.7	F	F	F	F	F	P (15%)
2.8	F	F	F	F	F	P (3%)
2.9	F	F	F	F	F	F
3	F	F	F	F	F	F

Shmoo plot for Scouting OR (CIM-Pr)

Voltage \ $\frac{R_{off}}{R_{on}}$	10	50	100	200	500	1000
0.2	F	F	F	F	F	F
0.3	P (2%)	P (161%)	P (256%)	P (375%)	P (587%)	P (813%)
0.4	P (60%)	P (296%)	P (439%)	P (620%)	P (942%)	P (1285%)
0.5	P (120%)	P (444%)	P (642%)	P (890%)	P (1333%)	P (1804%)
0.6	P (120%)	P (444%)	P (642%)	P (890%)	P (1333%)	P (1804%)
0.7	P (130%)	P (469%)	P (675%)	P (935%)	P (1398%)	P (1890%)
0.8	P (190%)	P (790%)	P (1113%)	P (1520%)	P (2244%)	P (3015%)
0.9	P (70%)	P (345%)	P (507%)	P (710%)	P (1072%)	P (1458%)
1	F	P (98%)	P (170%)	P (260%)	P (421%)	P (592%)
1.1	F	P (37%)	P (87%)	P (150%)	P (262%)	P (381%)
1.2	F	F	P (20%)	P (60%)	P (132%)	P (208%)
1.3	F	F	P (1%)	P (35%)	P (95%)	P (160%)
1.4	F	F	F	F	P (37%)	P (83%)
1.5	F	F	F	F	P (16%)	P (54%)
1.6	F	F	F	F	F	P (30%)
1.7	F	F	F	F	F	P (19%)
1.8	F	F	F	F	F	F
1.9	F	F	F	F	F	F

CIM-P operates at low R_{off}/R_{on} ratio & low voltages
CIM-P is less sensitive to device variations

Computation-in-Memory: Design flow



[Source: Jintao Yu, et.al, "The Power of Computation-in-Memory Based on Memristive Devices", ASP-DAC, paper 6A-2, 2020]

CIM potential: Database query example

Data Set

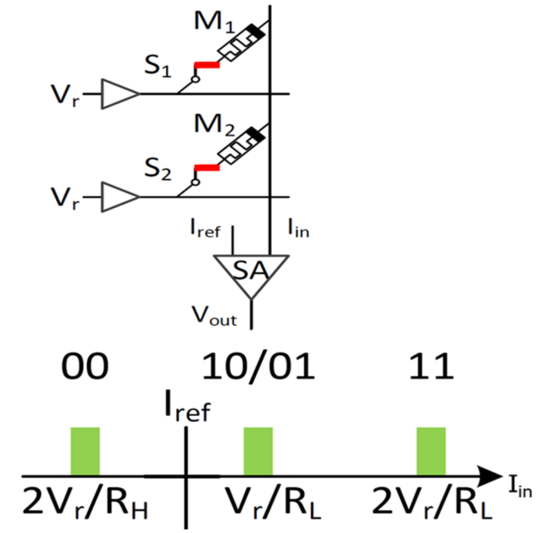
	Dist	Size	Year
A	55	Large	2016
B	23	Medium	2014
C	43	Small	2015
D	60	Medium	2016
E	25	Medium	2000
F	34	Medium	2001
G	18	Small	2012
H	30	Small	2011

```
QUERY: find DATA
satisfy
{far or large}
```

Bitmap Indexing

	A	B	C	D	E	F	G	H
Far	1	0	1	1	0	0	0	0
Near	0	1	0	0	1	1	1	1
Large	1	0	0	0	0	0	0	0
Medium	0	1	0	1	1	1	0	0
Small	0	0	1	0	0	0	1	1
New	1	0	0	1	0	0	0	0
Old	0	1	1	0	1	1	1	1
OR	1	0	1	1	0	0	0	0

Circuit Design:
Scouting

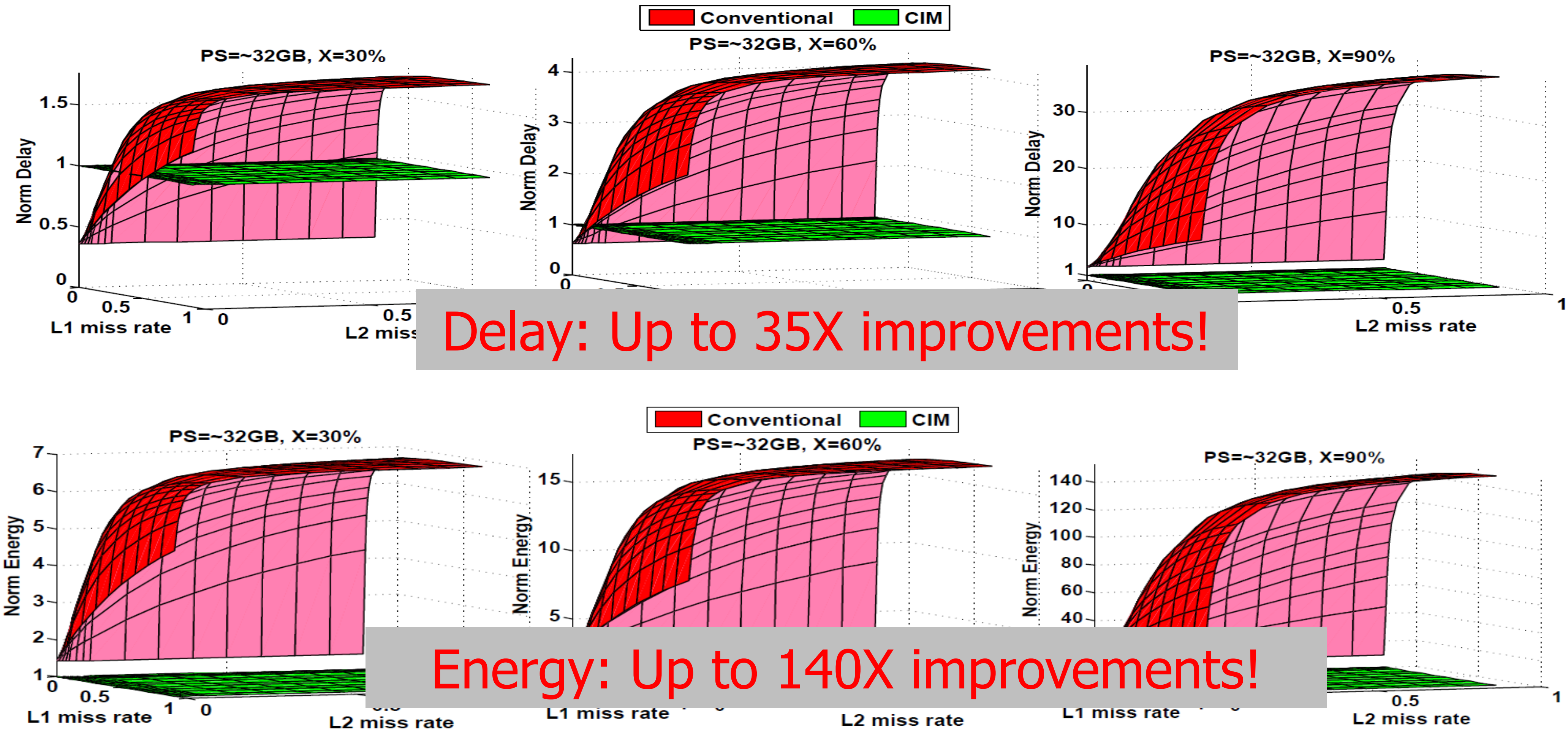


Requirements:

1. **Kernel**: bitwise OR
2. **Architecture**: CIM-Pr
   - I1 and I2: stored in a database (in memory)
   - O: read out

[Source: Hamdioui, et.al, "Applications of computation-in-memory architectures based on memristive devices", DATE 2019]

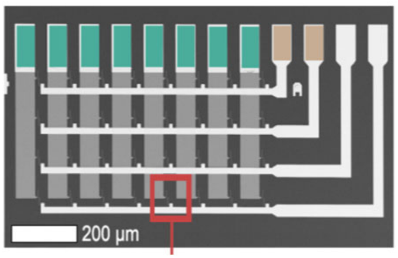
CIM potential: Database query example



CIM potential: Database query demo

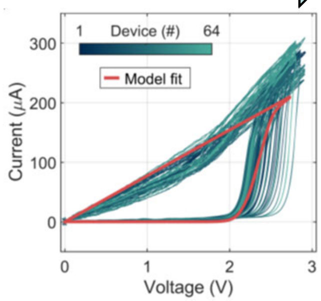
- Manufactured PCM + circuit simulation

4x8 **1R** crossbar

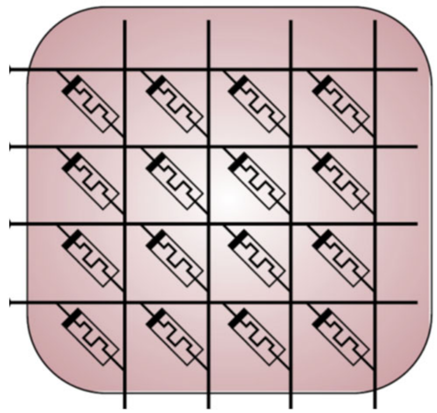


Ron/Roff: 20K/1M
Vread: 0.2V

Calibrated model

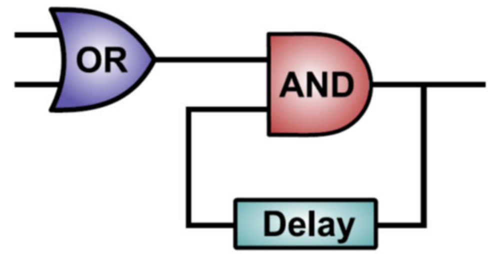


41x303 crossbar circuit



65nm periphery

- Sense amplifier
- **Cascade logic**
- etc.



- Results
  - **11-step** query in **36 ns**
  - Total energy: **20 pJ** (power: 558 μW)

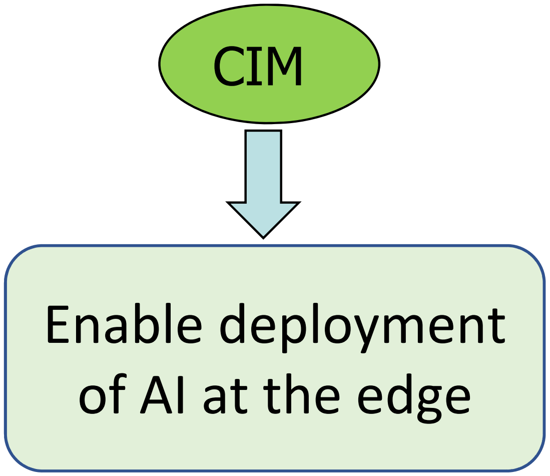
6fj/ operation

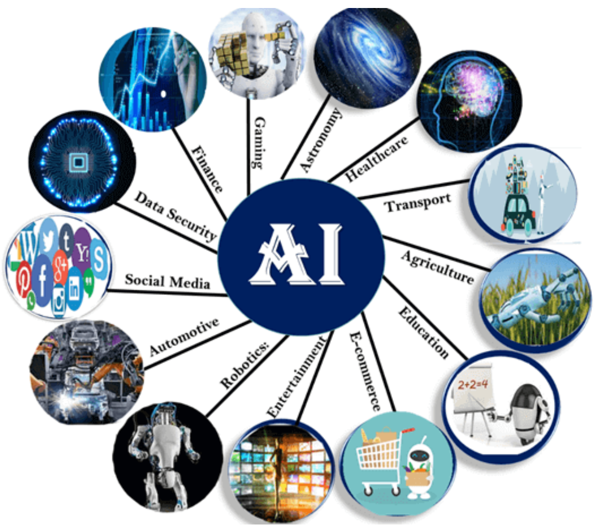
Throughput: **92.9 GOPS**

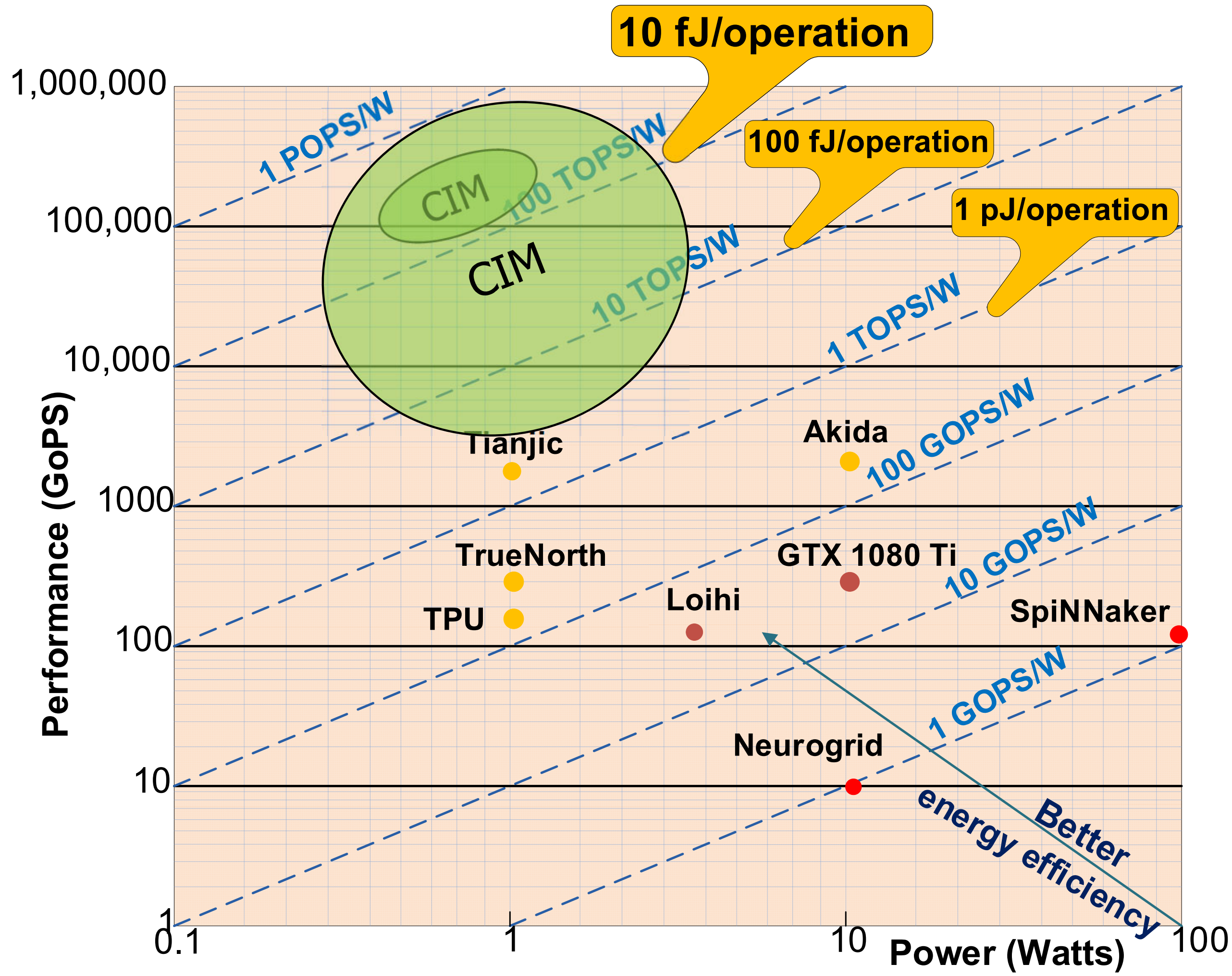
Efficiency: **166 TOPS/W**

[Source: Iason Giannopoulos et.al, "In-Memory Database Query", Advanced Intelligent Systems, open access, 2020]

Potential of CIM: Enabling edge/IoT applications



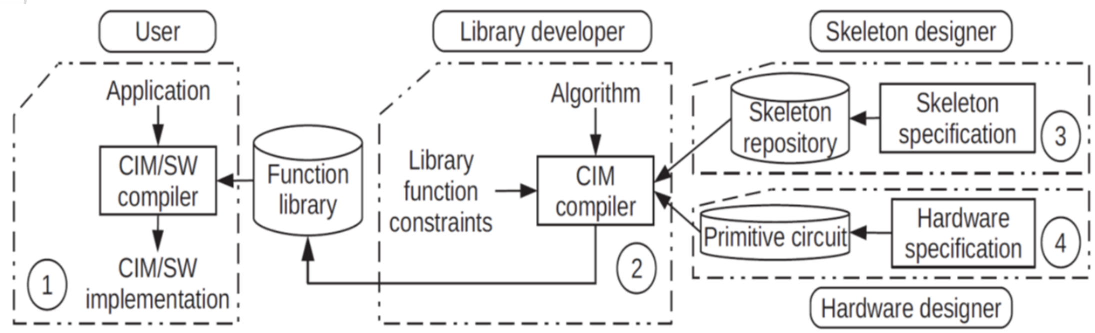




Challenges

Tools & Application

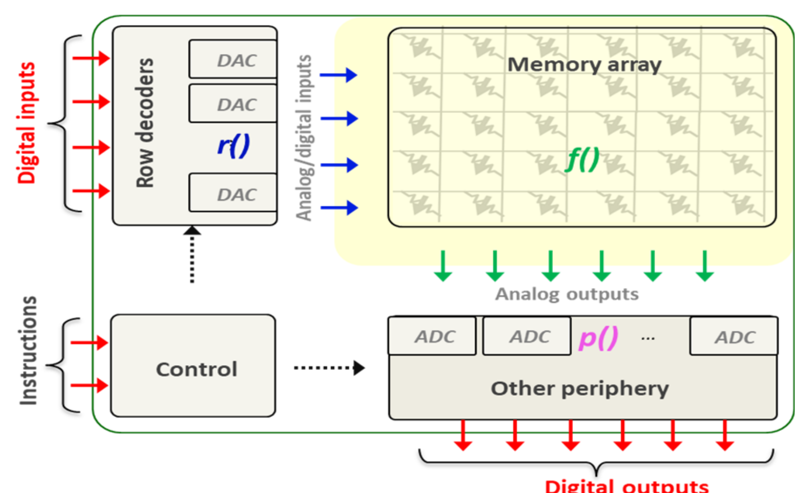
- Profilers & compilers
- Mapping applications on architecture
- EDA tool chains
- Bridging algorithms to the architecture and even to the device characteristics
- Simulators



[Source: J. Yu, et al., "ISAAC: Skeleton-Based Synthesis Flow for Computation-in-Memory Architectures," in IEEE Trans on ETC (TETC), 8(2), 2020]

Architecture

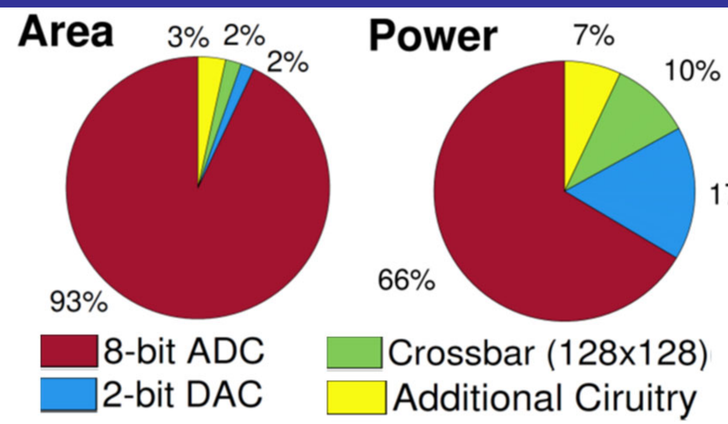
- Micro v macro architectures
- Intra- and inter-communication
- Virtual memory address translation mechanism
- Coherence and synchronization of CIM kernels
- Design exploration



Challenges

Circuit design

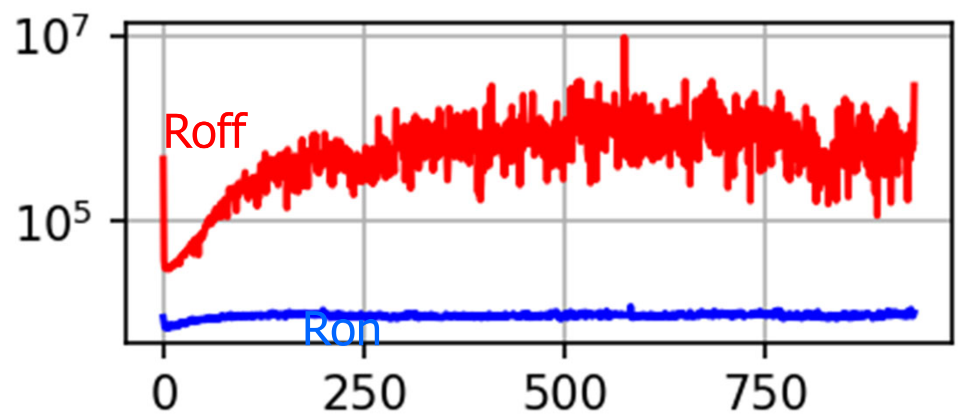
- Fast and energy efficient signal conversion circuits (DAC, ADC)
- High precision programming of NVM
- Precise measurement of current
  - Vector x matrix: output as current
- Design for non-idealities
- Managing sneak paths



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

Technology

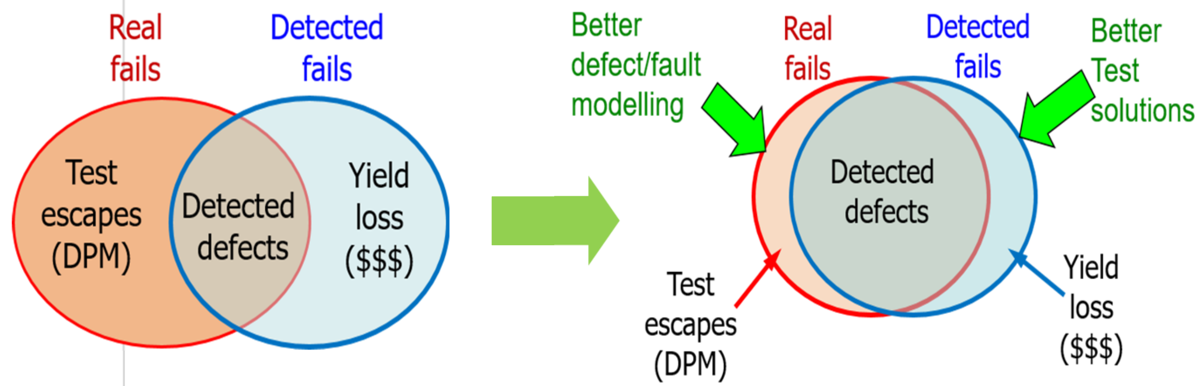
- Device-to-device variability, R ratio
- Multi-state behavior
- Energy switching
- Threshold behaviors
- Endurance
- 3D Integration & stacking, yield



Challenges

Test

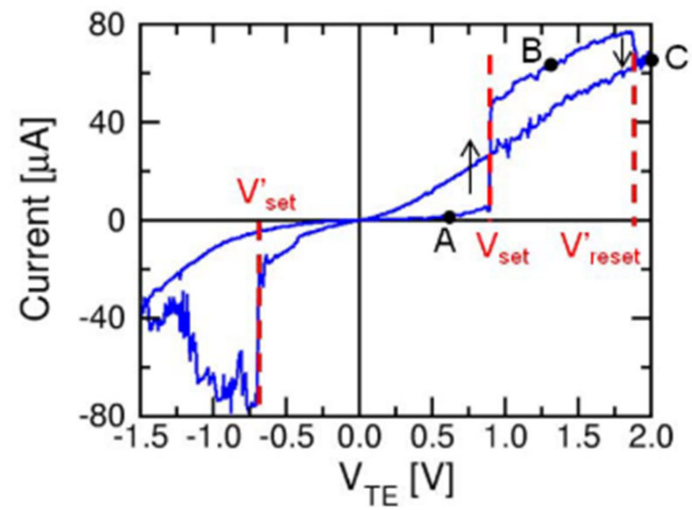
- There are "known unknowns"
- Defects not fully understood
- Fault models & test solutions
- DFX, BIST, BISR
- Memory v Comp configuration



Reliability

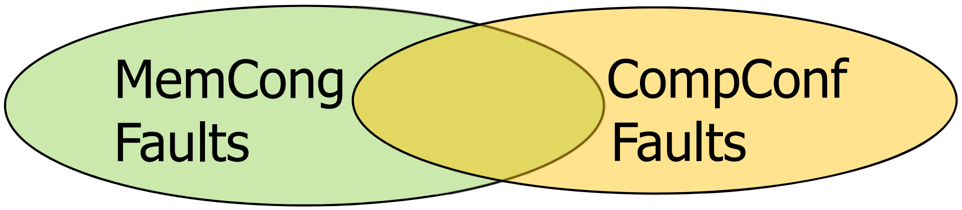
- Endurance
- Fault tolerance
- Cycle-to-cycle variability
- Degradation
- Intermittent unpredicted switching

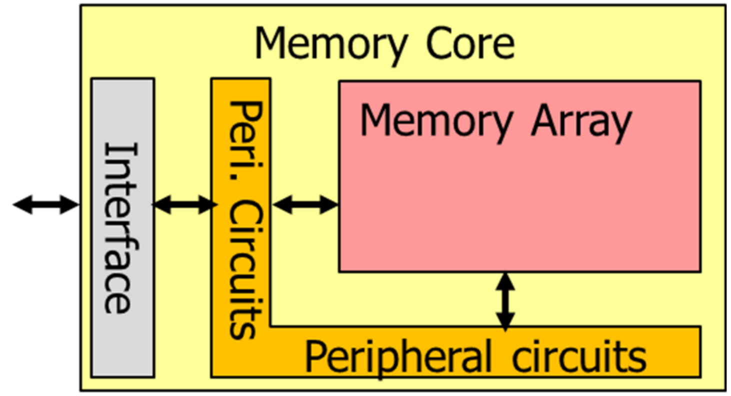
[Source: M. Fieback, et. al, "Intermittent Undefined State Fault in RRAMs", ETS 2021, Nominated for BPA



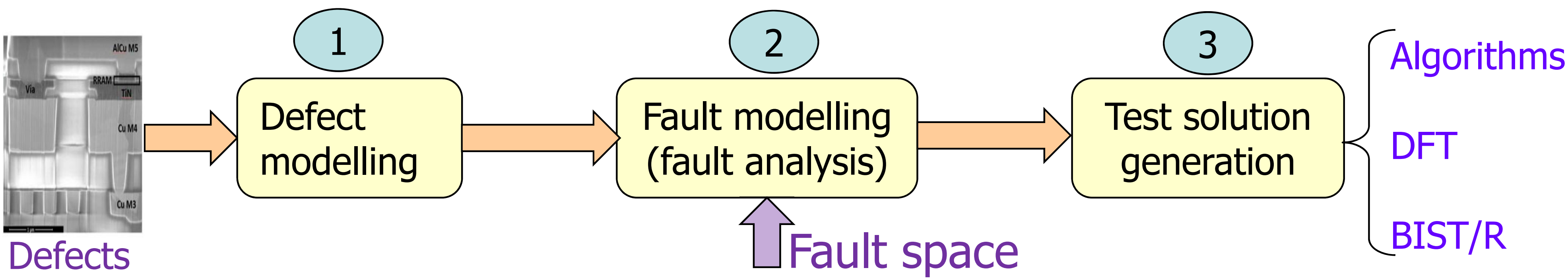
Testing CIM device

- Test for two configurations
  - **Memory config**: Read, Write
  - Computation config: Read, Write, Compute



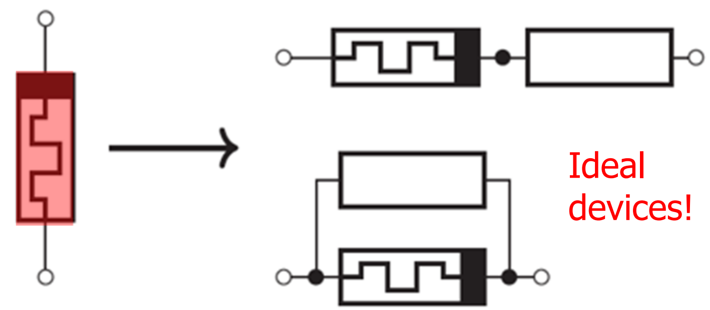


- Use typical structural approach

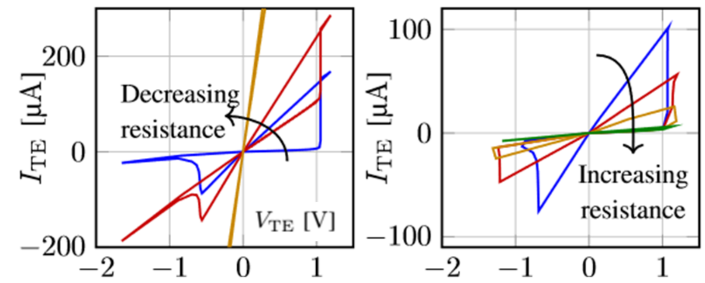


Testing CIM device: Device-Aware-Test

- State of the art
  - Defects modeled as linear resistors
  - Inappropriate for many defects
  - Cannot guarantee 0DPM



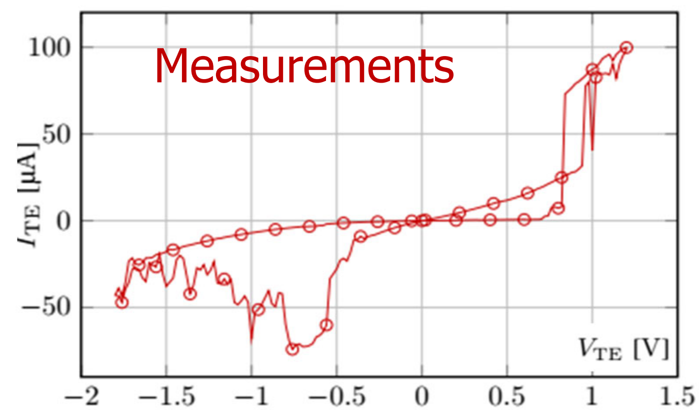
- CIM uses new device technologies
  - New failure mechanisms/ modes



- Need of accurate defect modeling
  - Incorporate defective behavior in device model
  - Crucial and critical for high quality test solutions

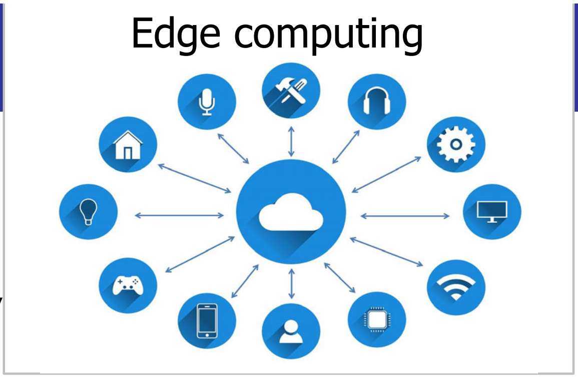
R-model fails to detect all defects

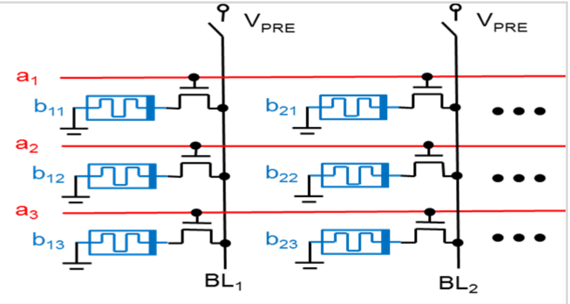
Device-Aware Test

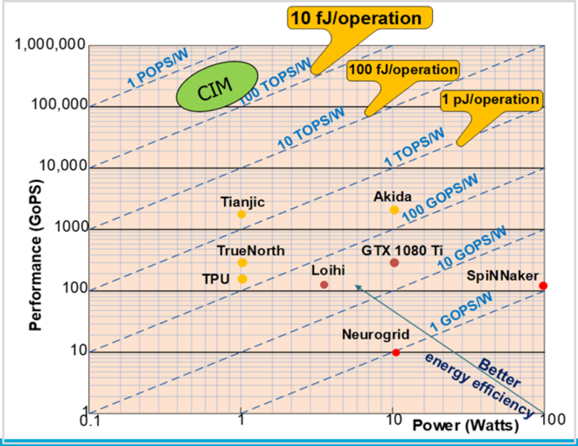


Conclusion

- **Emerging applications requires new chips**
  - IoT-Edge partnership: Data-centric computing & Huge data storage
  - Rethink: micro-architectures, design, Reliability, security, communication,
  - Efficiency: Order of fj/operation
- **Key enablers: new architectures and technologies**
  - Arch: CIM, artificial neural network, bio-inspired NN, AP, …
  - Tech: NV memories, 3D processing/stacking, photonics, …
- **Huge potential**
  - Order of magn. improvements in computational efficiency
  - Huge storage, reduced communication
  - Cheap implementation of basic operations
- **BUT still many challenges**
  - Technology, circuit, architecture, prog models, Testing, Reliability
  - Full application evaluation rather than small isolated kernels







Acknowledgement

http://www.mnemosene.eu/

Horizon 2020
European Union Funding
for Research & Innovation

MNEMOSENE

TU/e Technische Universiteit Eindhoven University of Technology
Where innovation starts

ETH zürich

RWTH RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE AACHEN

imec

IBM

Thanks