Energy Efficient Processing-In-Memory

-From Device to Algorithm

Deliang Fan
Assistant Professor, Ph.D.
School of Electrical, Computer and Energy Engineering
Arizona State University, Tempe, AZ, 85287, USA
Email: dfan@asu.edu
https://dfan.engineering.asu.edu/

Contributing Ph.D. Students
Zhezhi He, Shaahin Angizi, Adnan Rakin, Li Yang

About ME- Deliang FAN

- ➢ Ph.D. degree from Department of ECE at Purdue University, West Lafayette, IN, in 2015, under supervision of Prof. Kaushik Roy (Edward G. Tiedemann Jr. Distinguished Professor).
- ➢In 2019 Fall, I joined Arizona State University, Tempe, AZ as an Assistant Professor at School of ECEE.
- ➢Before that I was an assistant professor at ECE department, University of Central Florida, Orlando, FL. My main research have focused on:
  - Energy Efficient and High Performance Big Data Processing-In-Memory Circuit, Architecture and Algorithm (e.g. Deep Neural Network, Data Encryption, Graph Processing, bioinformatic Processing-in-Memory)
  - Hardware Aware Deep Neural Network Compression Algorithm for AI Edge/IoT Computing
  - Brain-inspired (Neuromorphic) and Boolean Computing Using Emerging Nanoscale Devices like Spintronics, ReRAM and Memristors
  - AI Security
  - Low Power Digital and Mixed Signal CMOS VLSI Circuit Design
- ➢I have published **80+** IEEE/ACM research papers in above areas; served as **leading-PI** for research projects from NSF CCF, NSF FET, SRC nCore project, SCEEE research initiation grant, Cyber Florida, UCF Inhouse fund; **three best paper awards** from GLSVLSI 2019, ISVLSI 2018 and 2017; one best paper candidate from ASPDAC 2019; one Front cover paper of IEEE Transactions on Magnetics.

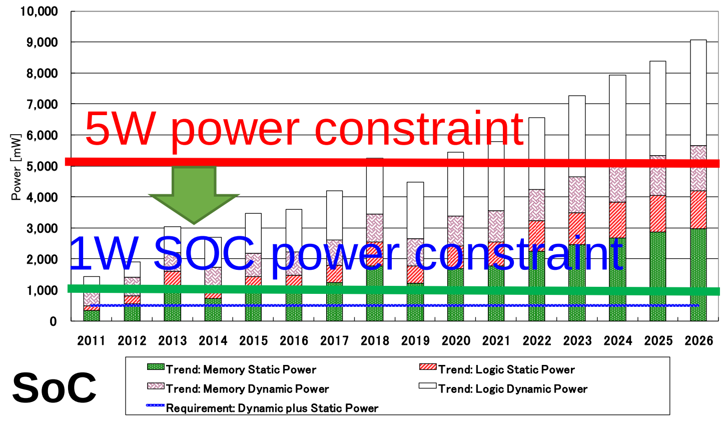
Outline

- Motivation:
  - Power Wall in CMOS technology;
  - Memory Wall in Von-Neumann Architecture
- Research Objectives and Methodologies:
  - Bottom-Up: *Device & Circuits* co-design for *parallel* and *reconfigurable* in-memory logic based on Non-Volatile Memory, like STT-MRAM, SOT-MRAM, ReRAM
  - Top-Down: *Architecture & Algorithm* *co-optimization* for data intensive *processing-in-memory* acceleration: Deep Neural Network, Data Encryption, Graph Processing, DNA Alignment, etc.
- Summary

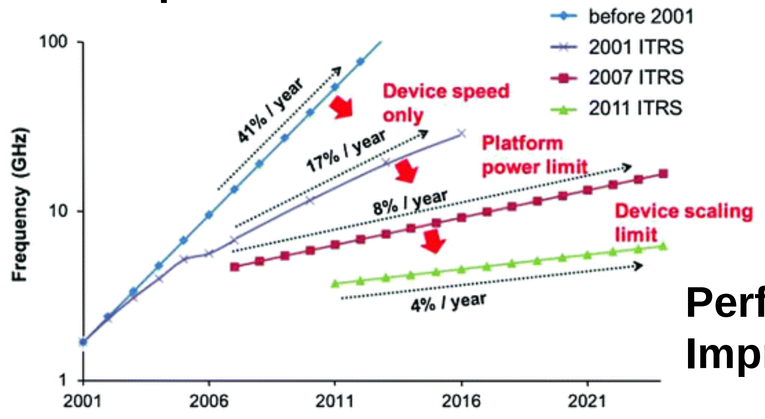
Motivation: Power Wall in CMOS Device

Power Wall

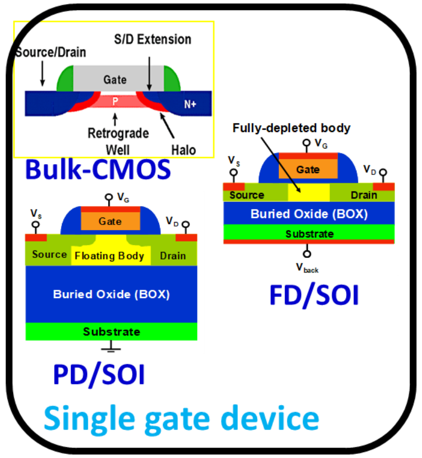
- ❖Low power design is a grand challenge!
- ❖Mobile devices with extremely low power
- ❖End of Moore's law and Dennard Scaling
- ❖ Possible solutions?

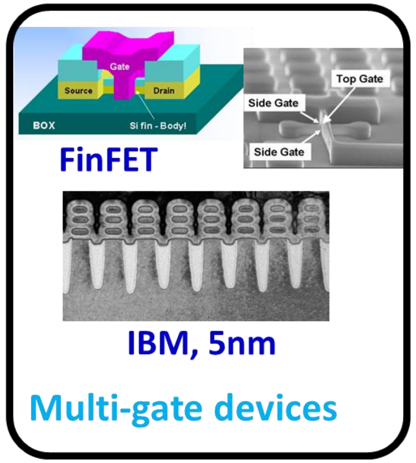


Mobile SoC power road map



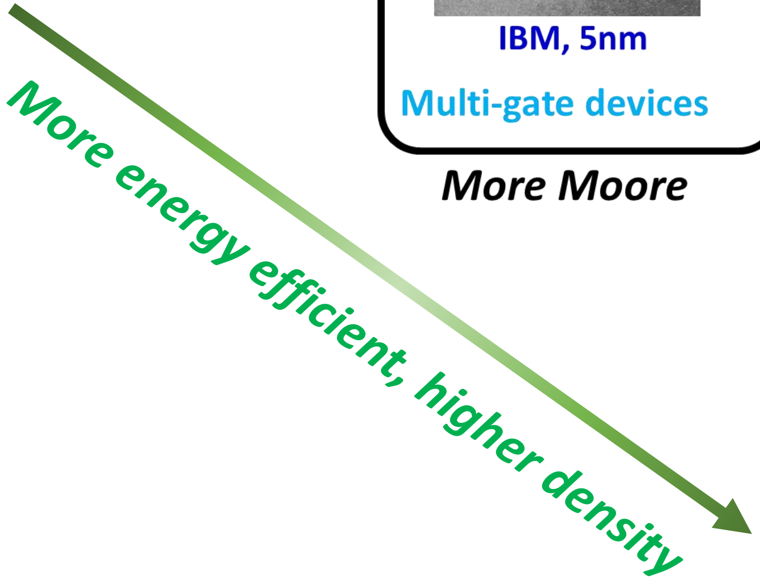
Performance Improvement



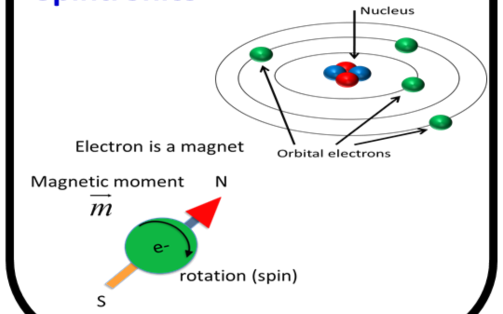


More Moore

Technology Trend



Carbon nanotube
Graphene
TFETs
III-V devices
Spintronics



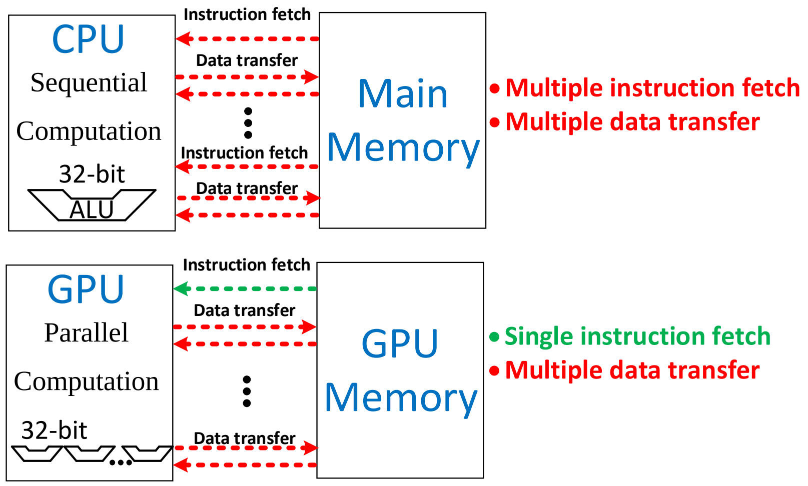
Post-CMOS ?

D. Fan, et. al., DAC 2018/2019, ICCAD2018, DATE 2019, ICCD 2017/2018, ASPDAC 2018/2019, TCAD 2018, TMAG 2018, TNANO 2018

Motivation: Energy Efficient In-Memory Computing

Memory Wall

Von-Neumann Architecture



Multiplication: 3.1pJ
Addition: 0.1pJ

∧

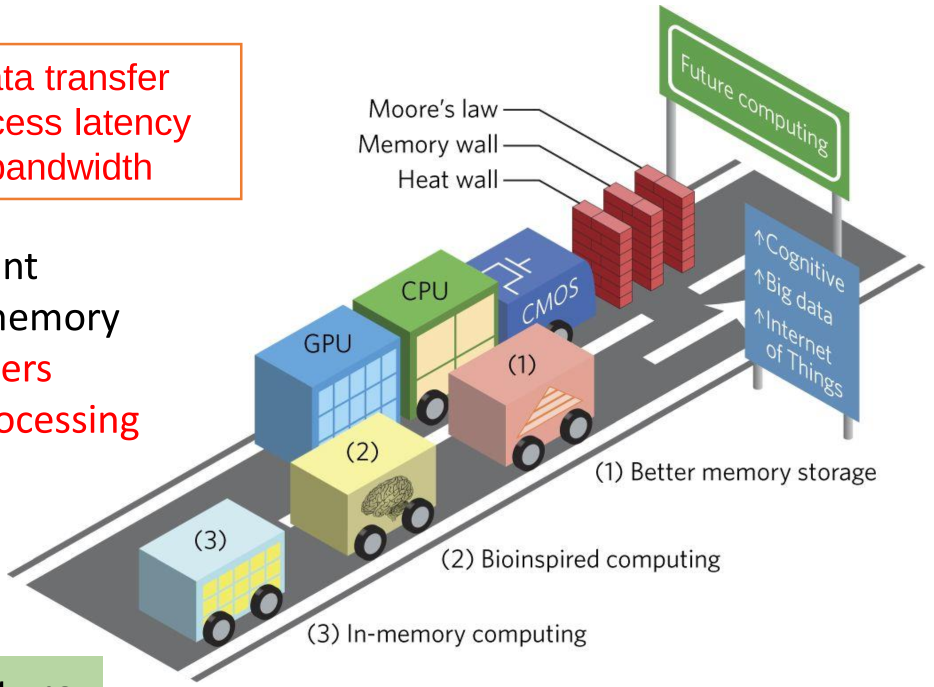
On-chip cache:
Energy: ~5pJ
Latency: ~10ns

∧

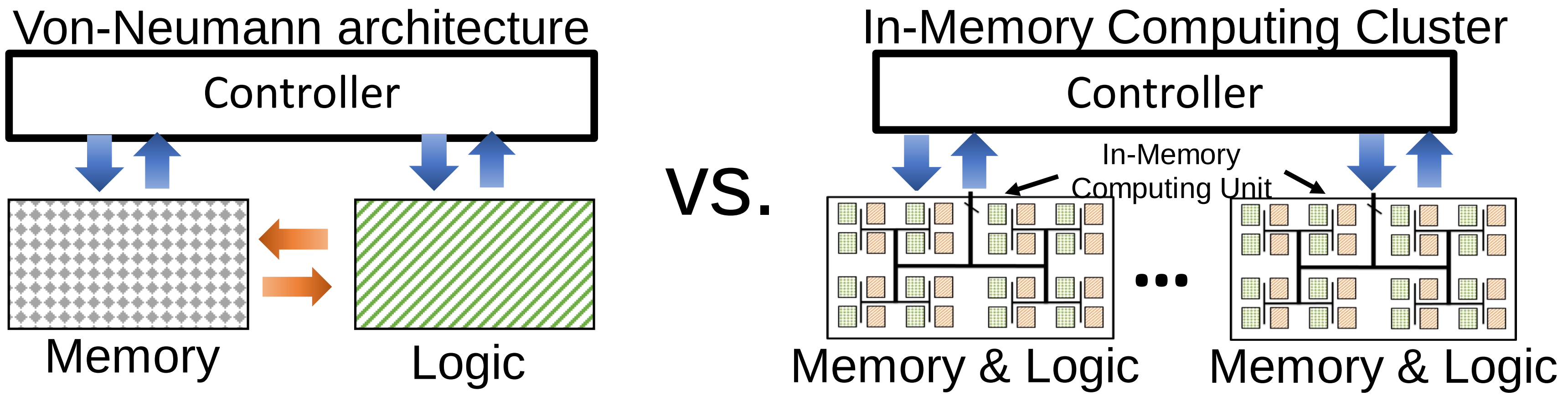
Off-chip memory:
Energy: ~640pJ
Latency: ~100ns

- ➢ Energy hungry data transfer
- ➢ Long memory access latency
- ➢ Limited memory bandwidth

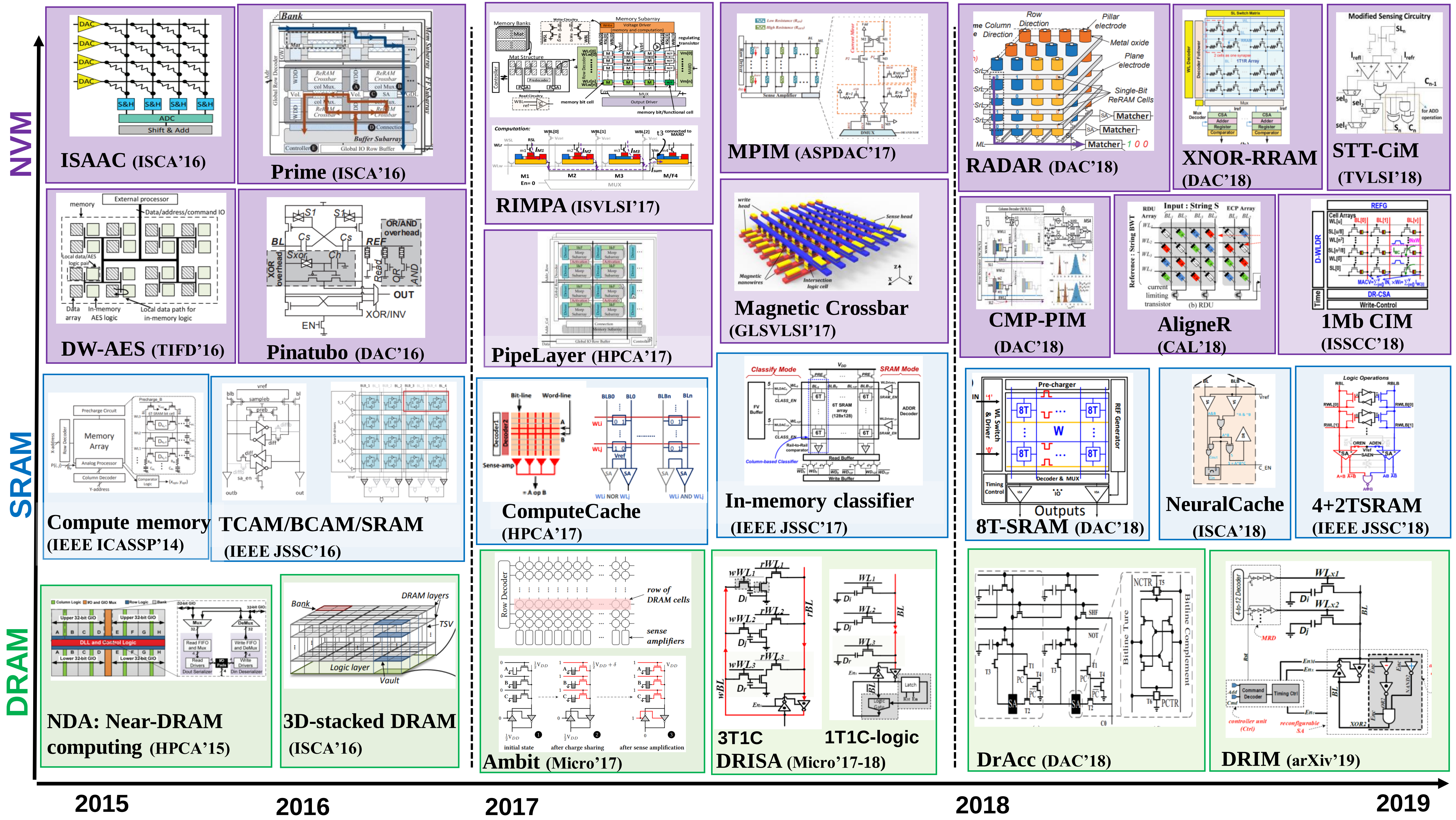
Moving a floating point number from main memory to CPU takes two orders more energy than processing in CPU



Processing-in-Memory Architecture



- ✓ Parallel, local data processing
- ✓ Short memory access latency
- ✓ Ultra-low energy
- ✓ Programmable, Low cost


NVM
SRAM
DRAM
ISAAC (ISCA'16)
Prime (ISCA'16)
DW-AES (TIFD'16)
Pinatubo (DAC'16)
RIMPA (ISVLSI'17)
PipeLayer (HPCA'17)
MPIM (ASPDAC'17)
Magnetic Crossbar (GLSVLSI'17)
RADAR (DAC'18)
XNOR-RRAM (DAC'18)
STT-CiM (TVLSI'18)
CMP-PIM (DAC'18)
AligneR (CAL'18)
1Mb CIM (ISSCC'18)
Compute memory (IEEE ICASSP'14)
TCAM/BCAM/SRAM (IEEE JSSC'16)
ComputeCache (HPCA'17)
In-memory classifier (IEEE JSSC'17)
8T-SRAM (DAC'18)
NeuralCache (ISCA'18)
4+2TSRAM (IEEE JSSC'18)
NDA: Near-DRAM computing (HPCA'15)
3D-stacked DRAM (ISCA'16)
Ambit (Micro'17)
3T1C
1T1C-logic
DRISA (Micro'17-18)
DrAcc (DAC'18)
DRIM (arXiv'19)
2015
2016
2017
2018
2019

Main Research Objective and Methodology

Architecture: Parallel Processing-in-Memory Accelerator
Algorithm: Data intensive and intelligent application algorithm development for developed PIM platform
Developed and developing PIM applications: deep neural network, data encryption, image processing, graph processing

Top-Down ↓ *Algorithm and Architecture Co-Design*

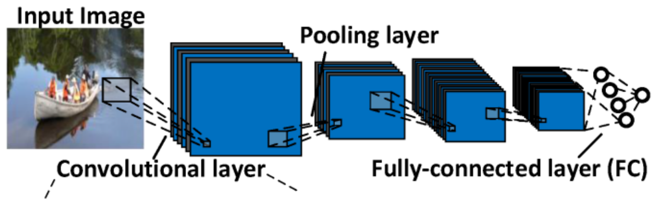
Architecture: In-Memory Computing + **Device & Circuits: NVM +CMOS**

Objective: Energy Efficient and Intelligent Processing-in-Memory

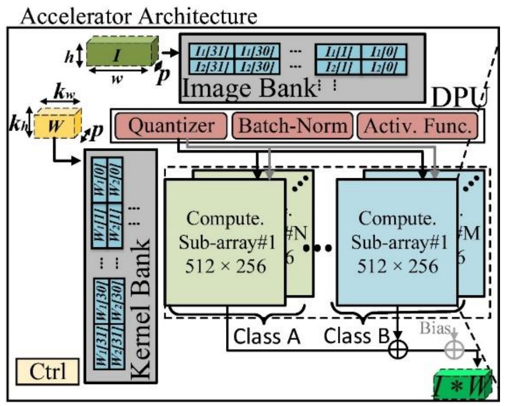
Bottom-Up ↑ *Device and Circuit Co-Design*

Device & Circuit: Parallel and Reconfigurable in-Memory Logic based on STT-MRAM, SOT-MRAM, DWM, ReRAM, DRAM, with extreme low overhead.
Objective: dual mode computational memory simultaneously working as memory and in-memory logic

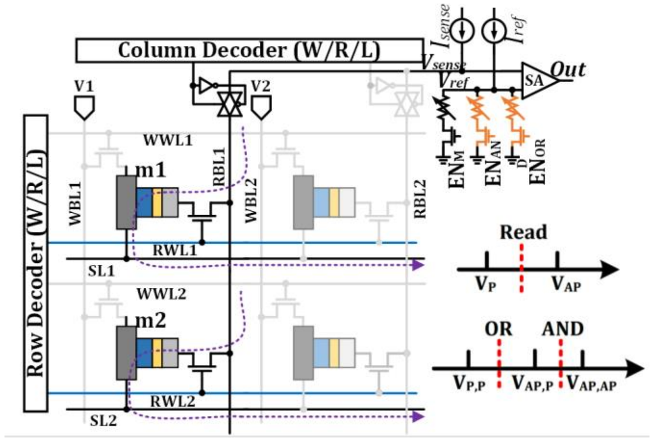
Partial related works in 2018 and 2019



Algorithm: D. Fan, et. al., CVPR 2019, ICCV2019, DAC 2018/2019, ICCAD 2018, WACV2018/2019, DATE 2019, ICCD 2018, ISVLSI 2018, ASPDAC 2018/2019, TCAD 2018, TNANO 2018, TMAG 2018



Architecture: D. Fan, et. al., ICCD 2018, DAC 2018/2019, TCAD 2018, ASPDAC 2018/2019

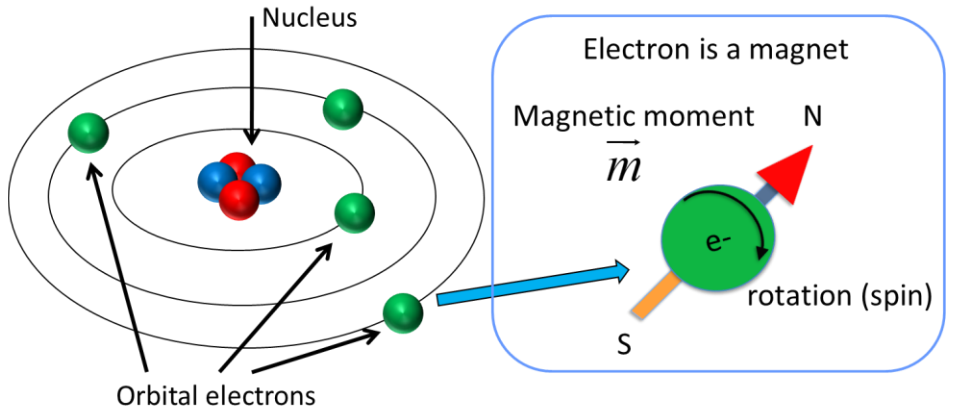


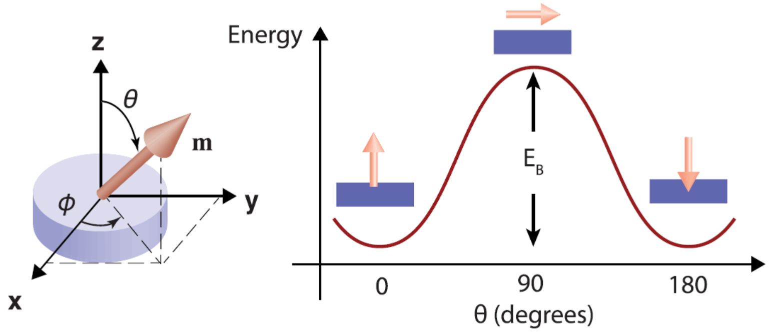
Circuit: D. Fan, et. al., TNANO 2018, TMAG 2018, TCAD 2018, DAC 2018/2019, ASPDAC 2018/2019, ISLPED 2018, Magnetic Letter

Outline

- Bottom Up: Device & Circuits co-design for *parallel* and *reconfigurable* in-memory logic based on NVM
  - Memory and In-Memory Complete Boolean Logic
  - One/Two-Cycle In-Memory Full Adder leading to Fast and Parallel In-Memory Adder
  - Overcome Operand Locality Issue in Existing In-Memory Logic Designs

Spintronic Devices and MRAM

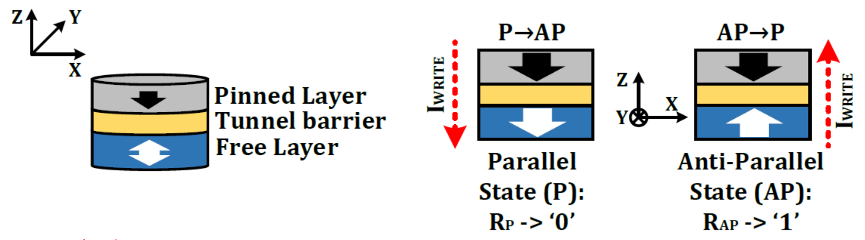


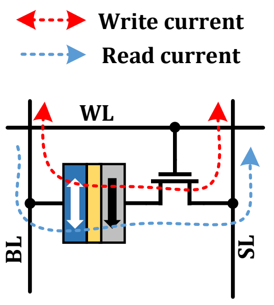


❖ Features

- ✓ **ultra-low switching energy**
- ✓ **non-volatility**
- ✓ **excellent retention time**
- ✓ **high integration density**
- ✓ **CMOS compatibility**

STT-MRAM

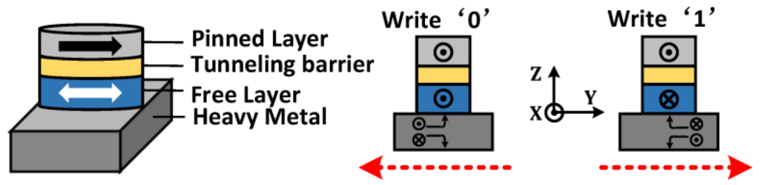

Z
Y
X
Pinned Layer
Tunnel barrier
Free Layer
P→AP
AP→P
IWRITE
Z
Y
X
IWRITE
Parallel State (P): RP -> '0'
Anti-Parallel State (AP): RAP -> '1'

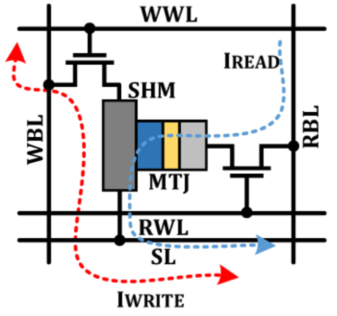


Limitations

- Write asymmetry
- Reliability-limited write speed
- Read write optimization conflicts

SOT-MRAM





Operations	Write '1'('0')	Read
WWL	V_{DD}	0
RWL	0	V_{DD}
RBL	0	I_{READ}
WBL	$V_{WP}(V_{WN})$	0
SL	0	0

Limitations

- Requires two access transistors
- Switching PMA MTJ requires FL engineering that involves fabrication challenge

Key Advantages

- Energy-efficient write
- Decoupled R/W current paths
- Separate optimization for Read and for Write

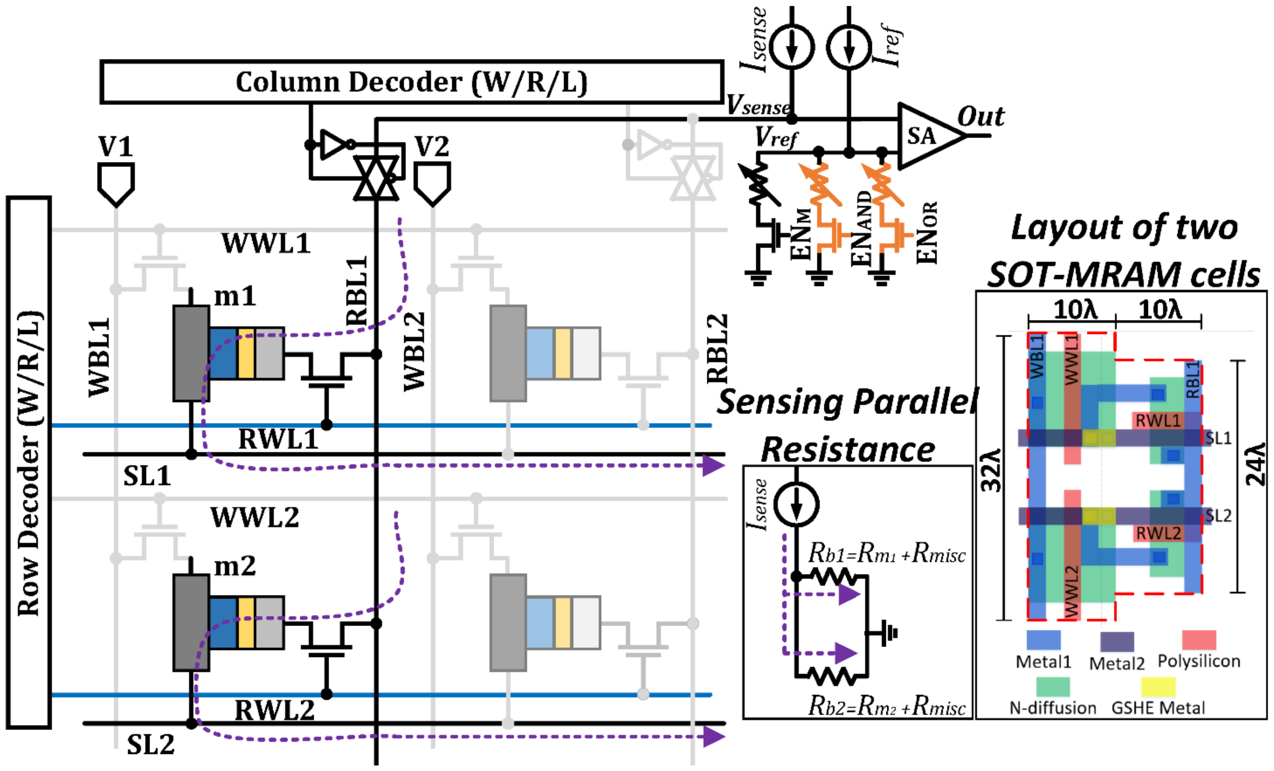
[1] X. Fong et al., "Spin-transfer torque devices for logic and memory: Prospects and perspectives," IEEE TCAD, vol. 35, pp. 1-22, 2016.
[2] C.-F. Pai et al., "Spin transfer torque devices utilizing the giant spin hall effect of tungsten," Applied Physics Letters, 2012. [3] S.W. Chung, et. al., "4Gb perpendicular STT-MRAM with compact cell structure and beyond", IEDM, 2016

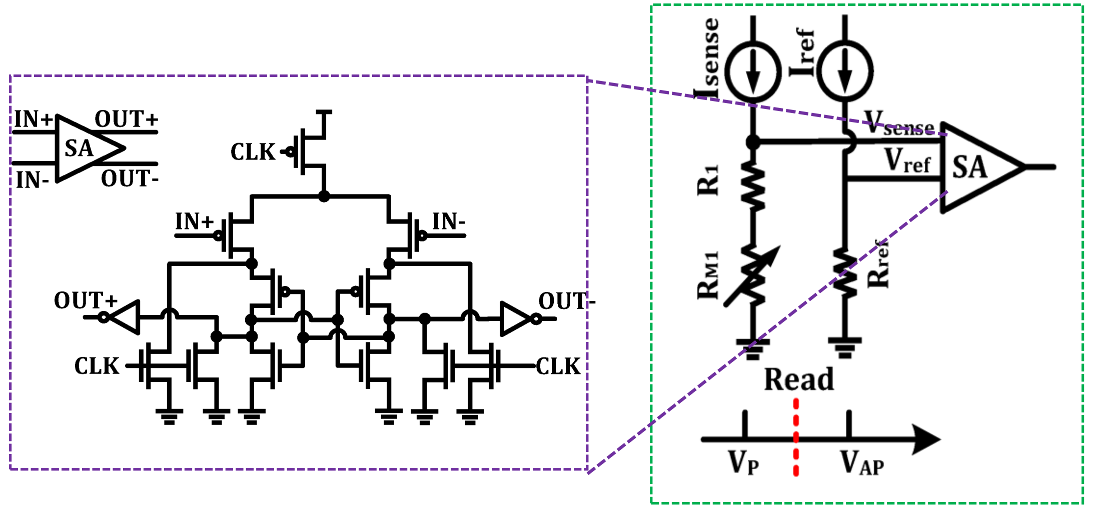
Dual-Mode Memory: Memory and Logic

Basic In-Memory logic – AND/NAND, OR/NOR

Dual-Mode IN-MEMORY Logic Design

- Dual mode architecture that perform both memory read-write and AND/ OR logic operations.
- ***Memory Mode:*** *charge current (~120 µA), 1ns switching speed*
- ***Computing Mode***: Every two bits stored in the identical column can be selected and sensed simultaneously. Through selecting different reference resistances $(EN_M, EN_{AND}, EN_{OR})$, the SA can perform basic in-memory Boolean functions (i.e. AND and OR).



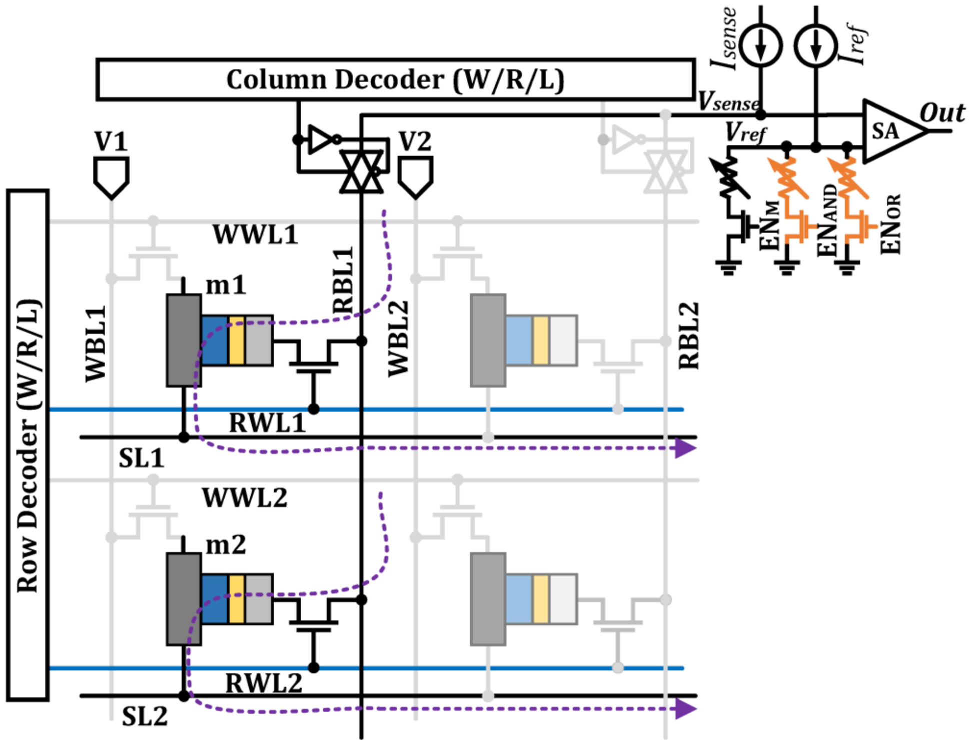


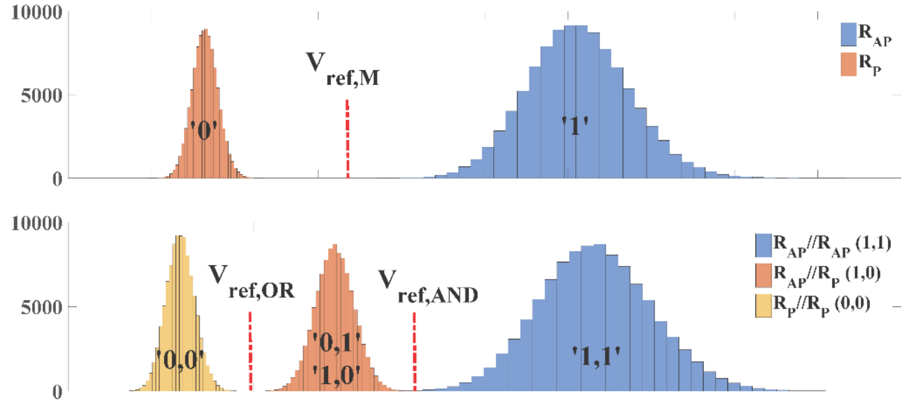
Shaahin Angizi, Zhezhi He, Farhana Parveen and Deliang Fan, "IMCE: Energy-Efficient Bit-Wise In-Memory Convolution Engine for Deep Neural Network," Asia and South Pacific Design Automation Conference (ASP-DAC), Jan. 22-25, 2018, Jeju Island, Korea 11

Dual-Mode IN-MEMORY Logic Design

- For AND operation, R_{ref} is set at the midpoint of $R_{AP} \| R_P = (1,0)$ and $R_{AP} \| R_{AP} = (1,1)$
- For OR operation, R_{ref} is set at the midpoint of $R_P \| R_P$ and $R_P \| R_{AP}$
- We have performed Monte-Carlo simulation with **100000 trials**. A $\sigma = 5\%$ variation is added on the Resistance-Area product (RA_P), and a $\sigma = 10\%$ process variation is added on the TMR.
- Sense Margin will be reduced by increasing the logic fan-in (i.e. number of parallel memory cells).
- To avoid read failure, only two fan-in in-memory logic is used in this work.

- No XOR/XNOR Logic now, intermediate data write-back needed if implemented using AND/OR
- More logic functions needed !





Monte Carlo simulation result

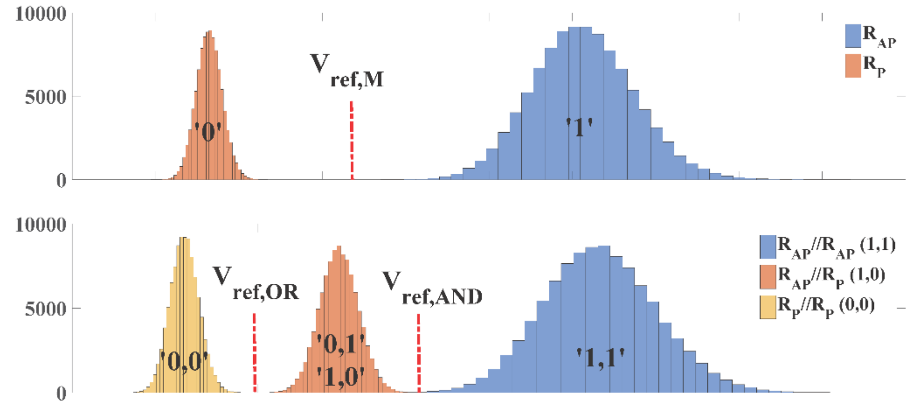
More Logic Functions Supported

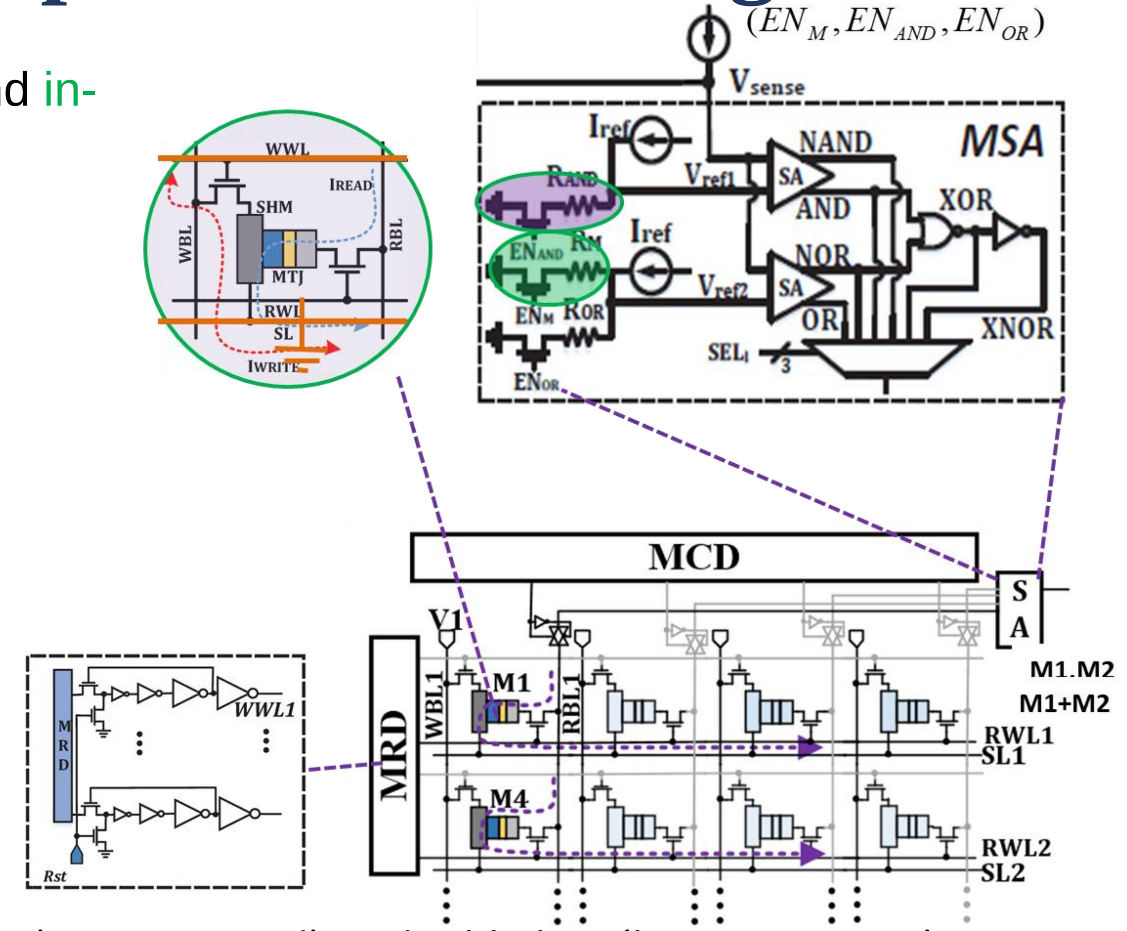
Reconfigurable AND/NAND, OR/NOR, XOR/XNOR, Majority In-Memory Logic in one design

D. Fan. et. al. DAC 2018, ICCAD 2018, ISVLSI 2018

Reconfigurable Complete Boolean Logic

- ❖ Dual mode architecture that perform both memory and in-memory logic operations.
- ❖ Only two fan-in in-memory logic to avoid logic failure!



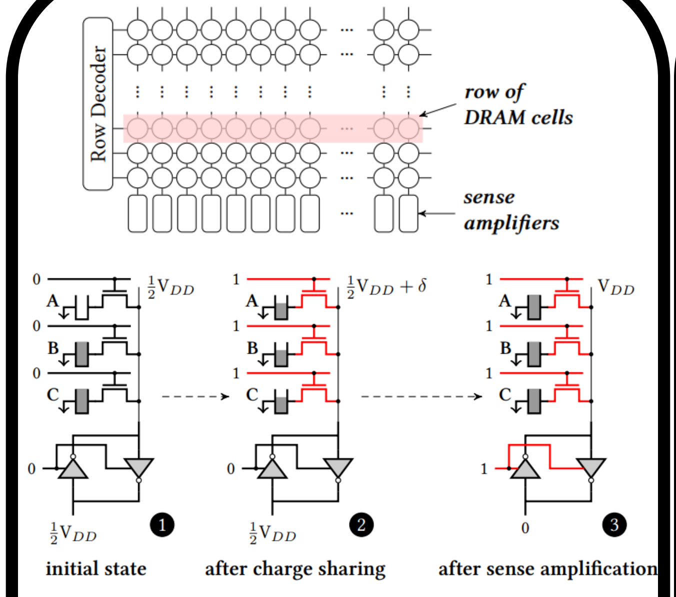


- Modified row/column decoder can enable either single line (memory read) or double line (logic operation).
- SA can provide bitwise AND/NAND and OR/NOR, XOR/XNOR can be realized through combinational logic gates (AND,NOR).
- **Complete Parallel Boolean Logic in one SA and one sensing cycle: AND/NAND, OR/NOR, XOR/XNOR**

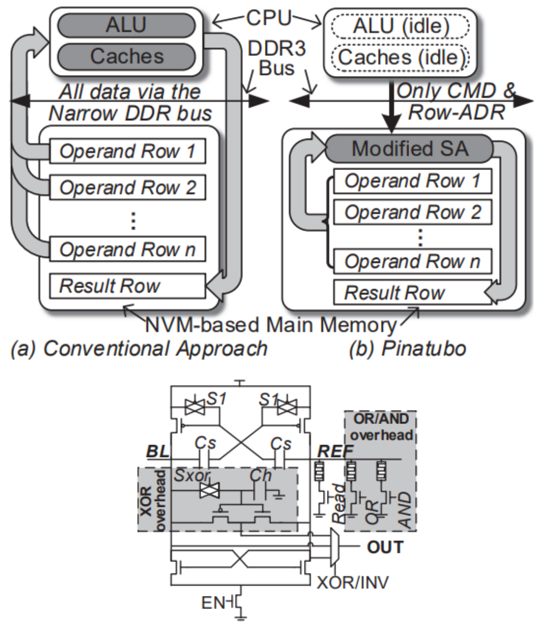
[1] Shaahin Angizi, Zhezhi He and Deliang Fan, "PIMA-Logic: A Novel Processing-in-Memory Architecture for Highly Flexible and Energy-Efficient Logic Computation, " *IEEE/ACM Design Automation Conference* (**DAC**), June 24-28, 2018, San Francisco, CA, USA

[2] Zhezhi He, Shaahin Angizi and Deliang Fan, "Accelerating Low Bit-Width Deep Convolution Neural Network in MRAM," *IEEE Computer Society Annual Symposium on VLSI* (***ISVLSI***), July 9-11, 2018, Hong Kong, CHINA

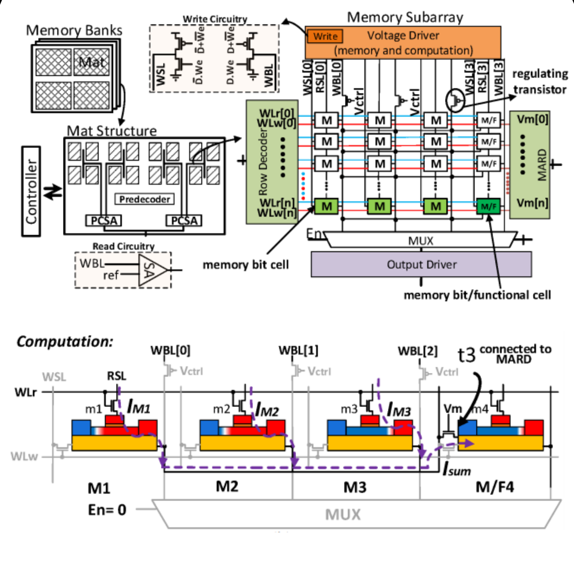
Recent Processing-in-Memory Platforms



Ambit: DRAM-based [4]



Pinatubo: NVM-based [5]



RIMPA: DWM-based [6]

Issues:

1. **Only simple logic (7+ cycles for FA, intermediate data write-back)**
2. **Operand locality**

- <u>Operand locality issue</u>
- Original data overwritten
- Multi-cycle operations
- <u>Simple logic</u>
- Low area overhead
- Hardware-friendly
- Exploit the full bandwidth

- <u>Operand locality issue</u>
- Modified SA
- <u>Simple logic</u>
- Medium area overhead
- Support one-step multi-row operations
- General platform

- <u>Operand locality issue</u>
- Large area overhead
- <u>Simple logic</u>
- Fast MG computation
- Ultra-low power

?

our solutions

[4] V. Seshadri et al., "Ambit: In-memory accelerator for bulk bitwise operations using commodity DRAM technology," MICRO, 2017, pp. 273-287: ACM.
[5] S. Li et al., "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," in 2016 53nd DAC. IEEE, 2016.
[6] S. Angizi et al. "Rimpa: A new reconfigurable dual-mode in-memory processing architecture with spin hall effect-driven domain wall motion device," *(ISVLSI)*, 2017, pp. 45-50: IEEE.

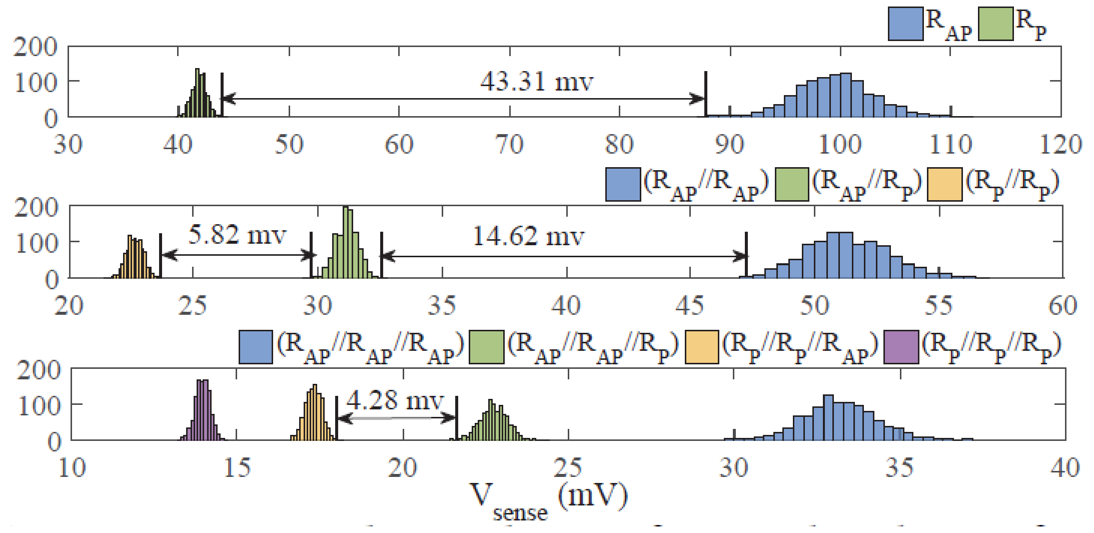
Two-Cycle In-Memory Full Adder

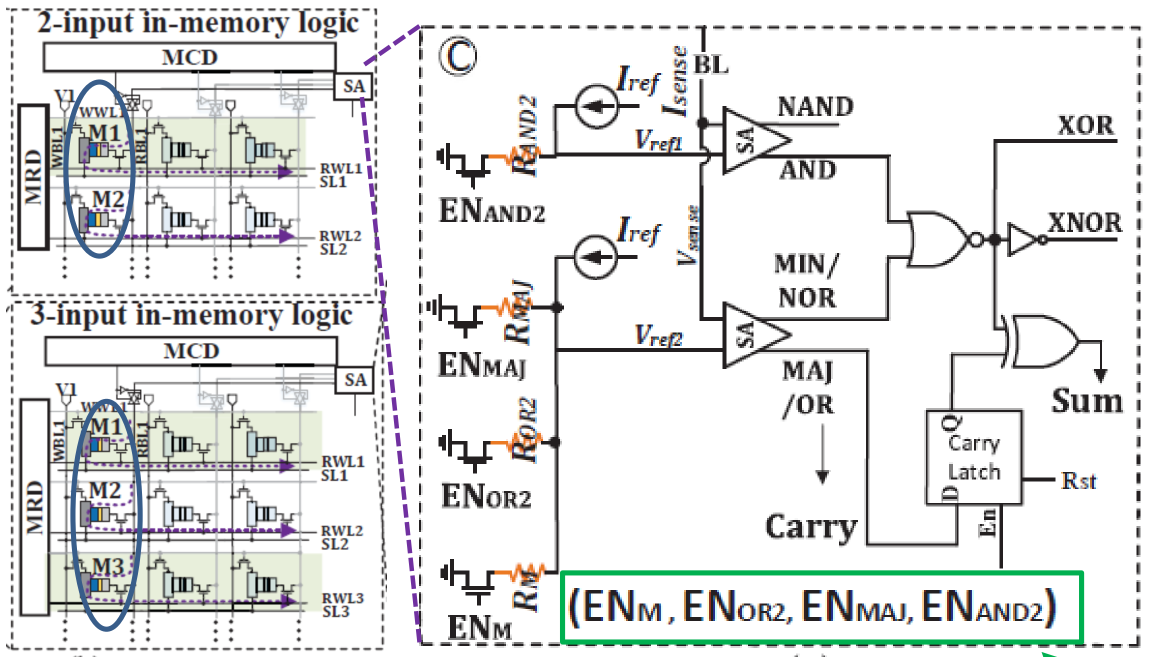
D. Fan. et. al. DAC 2019, ASPDAC 2019

Reconfigurable Logic-SA

- Dual mode architecture that performs both memory and in-memory logic operations.
- Up to three fan-in in-memory logic to avoid logic failure!

Monte Carlo simulation result





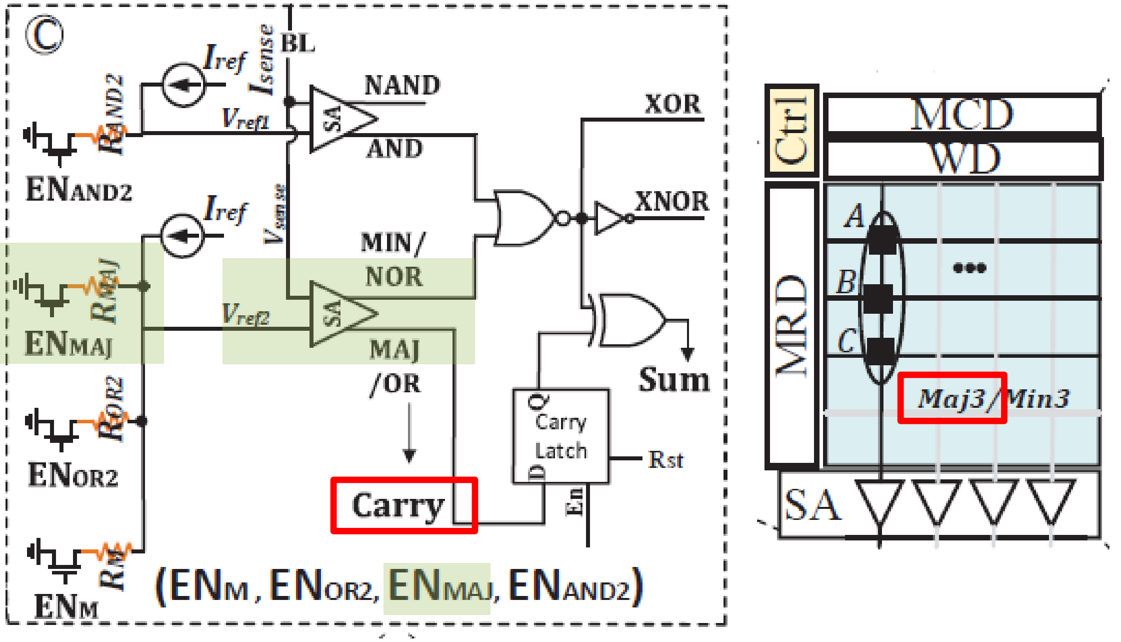
Configuration of enable bits for different functions

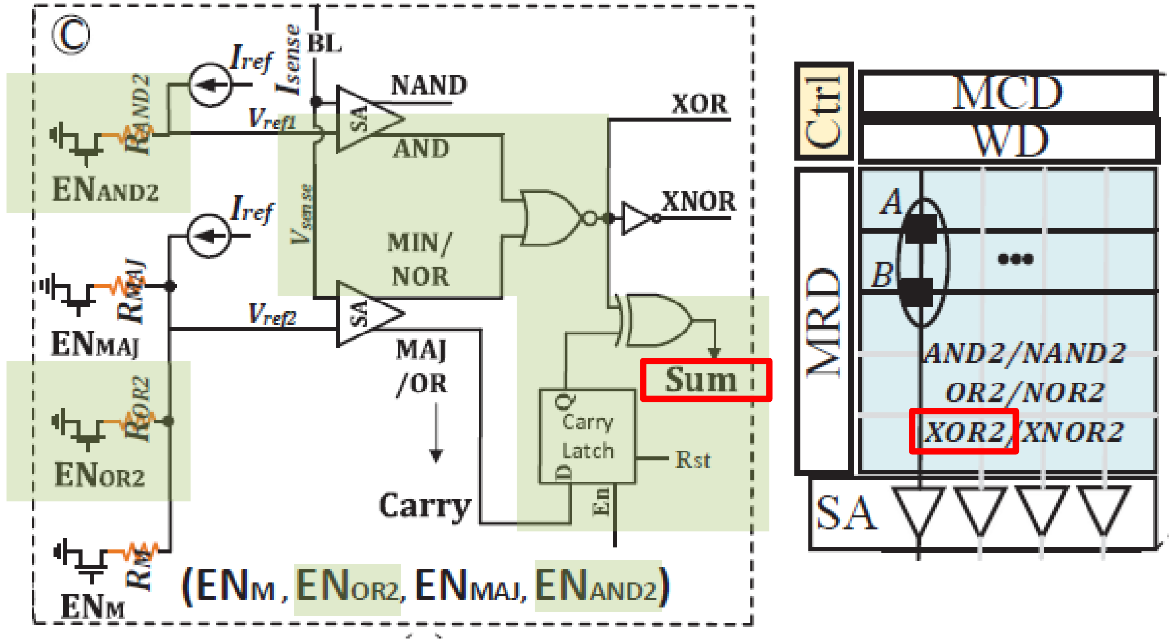
In-memory Operations	read	OR2/ NOR2	AND2/ NAND2	MAJ/ MIN	XOR2/ XNOR2
EN_M	1	0	0	0	0
EN_{OR2}	0	1	0	0	1
EN_{MAJ}	0	0	0	1	0
EN_{AND2}	0	0	1	0	1

- 2-input Boolean Logic in one SA and one sensing cycle: AND/NAND, OR/NOR, XOR/XNOR.
- 3-input Boolean Logic in one SA and one sensing cycle: MAJ/MIN.

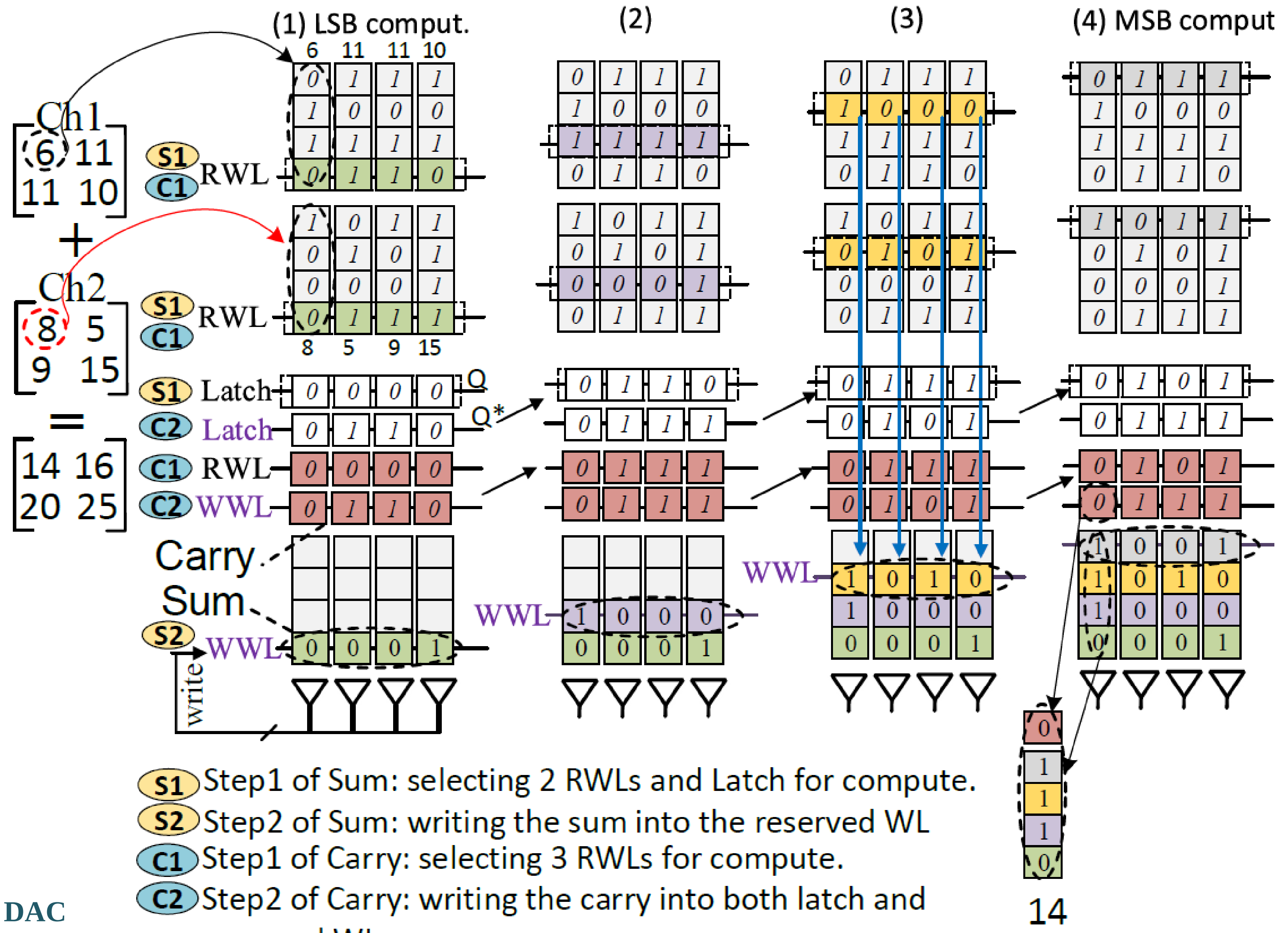
In-memory Addition

- ***Carry*** is directly produced by ParaPIM's MAJ function (3-row activation).
- A *Carry Latch* to store intermediate ***Carry*** outputs to be used in summation of next bits.
- ***Sum*** output is achieved by 2-row activated XOR followed by a 2-input XOR gate connected to it and *Carry Latch*.
- Enable parallel One Computation per two Memory Cycles.
- Assume *A*, *B* and *C* operands, the 2- and 3-input in-memory logic schemes generates ***Sum***(/***Difference***) and ***Carry***(/***Borrow***) bits very efficiently.





Parallel In-Memory Multi-bit Adder



$$\begin{bmatrix} 6 & 11 \\ 11 & 10 \end{bmatrix} + \begin{bmatrix} 8 & 5 \\ 9 & 15 \end{bmatrix} = \begin{bmatrix} 14 & 16 \\ 20 & 25 \end{bmatrix}$$

S1 Step1 of Sum: selecting 2 RWLs and Latch for compute.

S2 Step2 of Sum: writing the sum into the reserved WL

C1 Step1 of Carry: selecting 3 RWLs for compute.

C2 Step2 of Carry: writing the carry into both latch and reserved WL

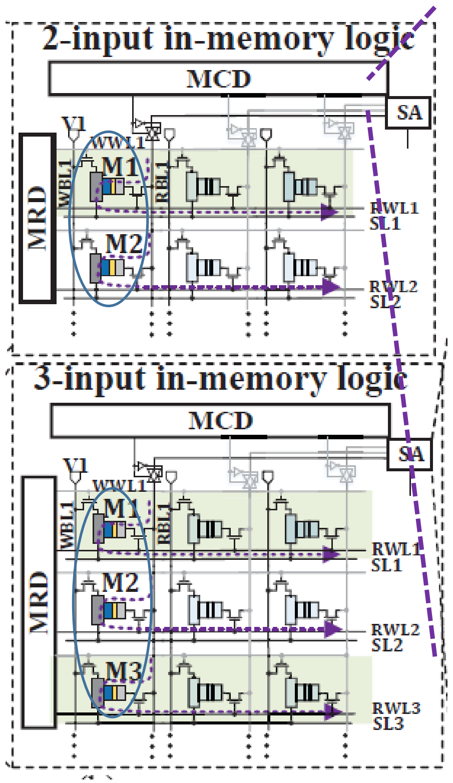
- Parallel Matrix Addition enabled
- 2N cycles are needed for N-bit adder

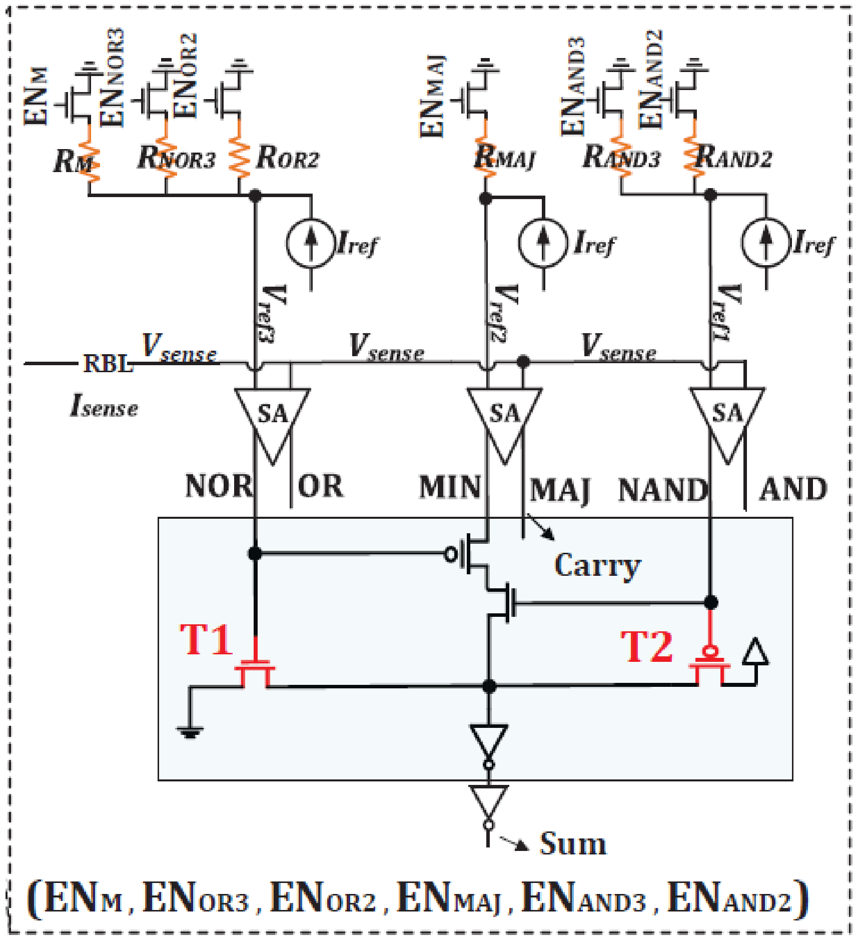
D. Fan. et. al. DAC 2019, ASPDAC 2019

One Cycle In-Memory Full Adder

D. Fan. et. al. DATE 2019, DAC 2019

Reconfigurable Logic-SA

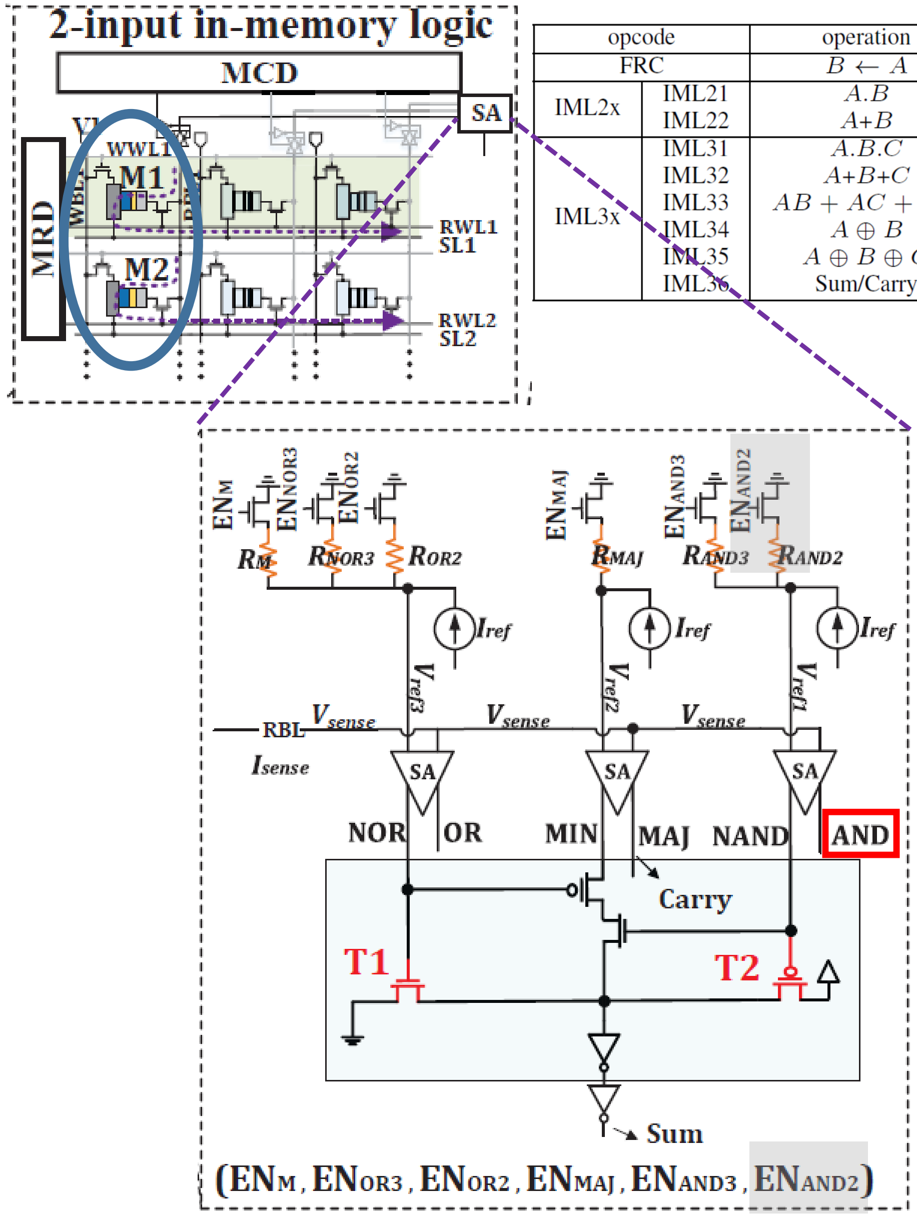




Ops.	read	OR2/ NOR2	AND2/ NAND2	MAJ/ MIN	OR3/ NOR3	AND3/ NAND3	Add/ XOR3/XNOR3 XOR2/XNOR2
EN_M	1	0	0	0	0	0	0
EN_{OR2}	0	1	0	0	0	0	0
EN_{AND2}	0	0	1	0	0	0	0
EN_{OR3}	0	0	0	0	1	0	1
EN_{AND3}	0	0	0	0	0	1	1
EN_{MAJ}	0	0	0	1	0	0	1

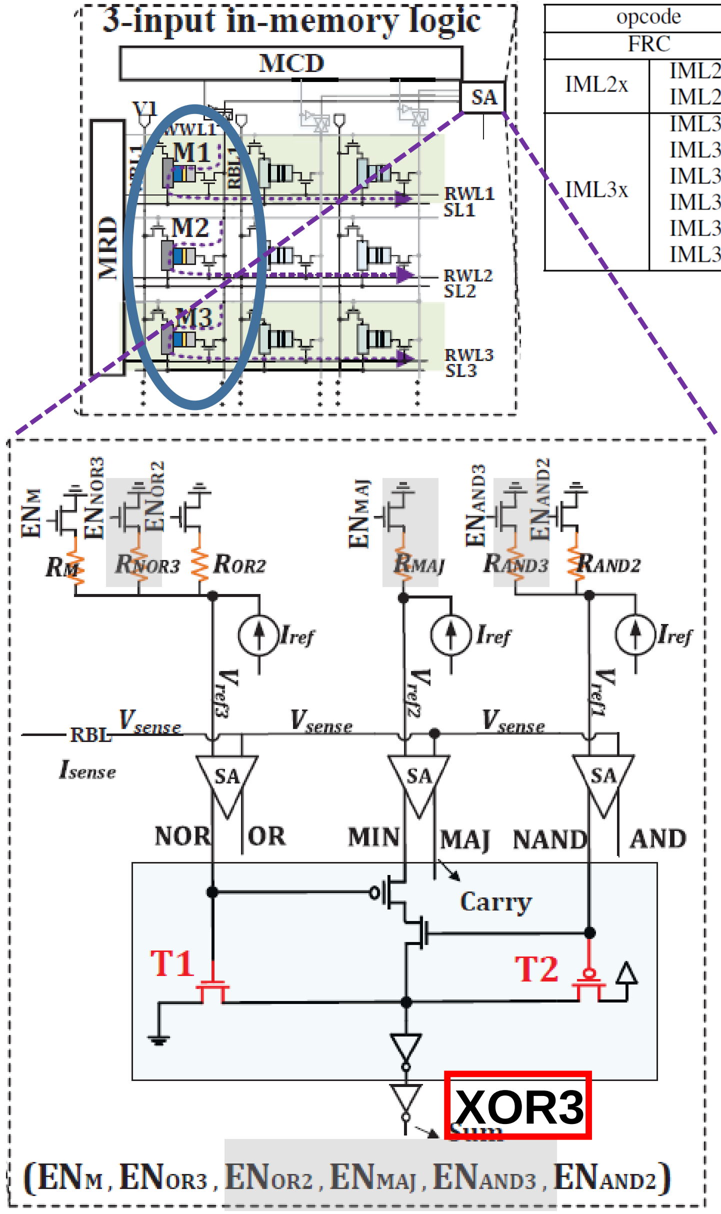
2-input Boolean Logic (IML2x) in one SA and one sensing cycle: AND2/NAND2, OR2/NOR2.
3-input Boolean Logic (IML3x) in one SA and one sensing cycle: AND3/NAND3, OR3/NOR3, MAJ/MIN, XOR2/XNOR2, XOR3/XNOR3, Addition.

In-memory AND2 (IML21)



opcode		operation	function
FRC		$B \leftarrow A$	Copy row A to Row B
IML2x	IML21	$A.B$	AND2/NAND2
	IML22	$A+B$	OR2/NOR2
IML3x	IML31	$A.B.C$	AND3/NAND3
	IML32	$A+B+C$	OR3/NOR3
	IML33	$AB+AC+BC$	MAJ/MIN
	IML34	$A \oplus B$	XOR2/XNOR2
	IML35	$A \oplus B \oplus C$	XOR3/XNOR3
	IML36	Sum/Carry	add/sub

In-memory XOR3 (IML35)



opcode		operation	function
FRC		$B \leftarrow A$	Copy row A to Row B
IML2x	IML21	$A.B$	AND2/NAND2
	IML22	$A+B$	OR2/NOR2
IML3x	IML31	$A.B.C$	AND3/NAND3
	IML32	$A+B+C$	OR3/NOR3
	IML33	$AB+AC+BC$	MAJ/MIN
	IML34	$A \oplus B$	XOR2/XNOR2
	IML35	$A \oplus B \oplus C$	XOR3/XNOR3
	IML36	Sum/Carry	add/sub

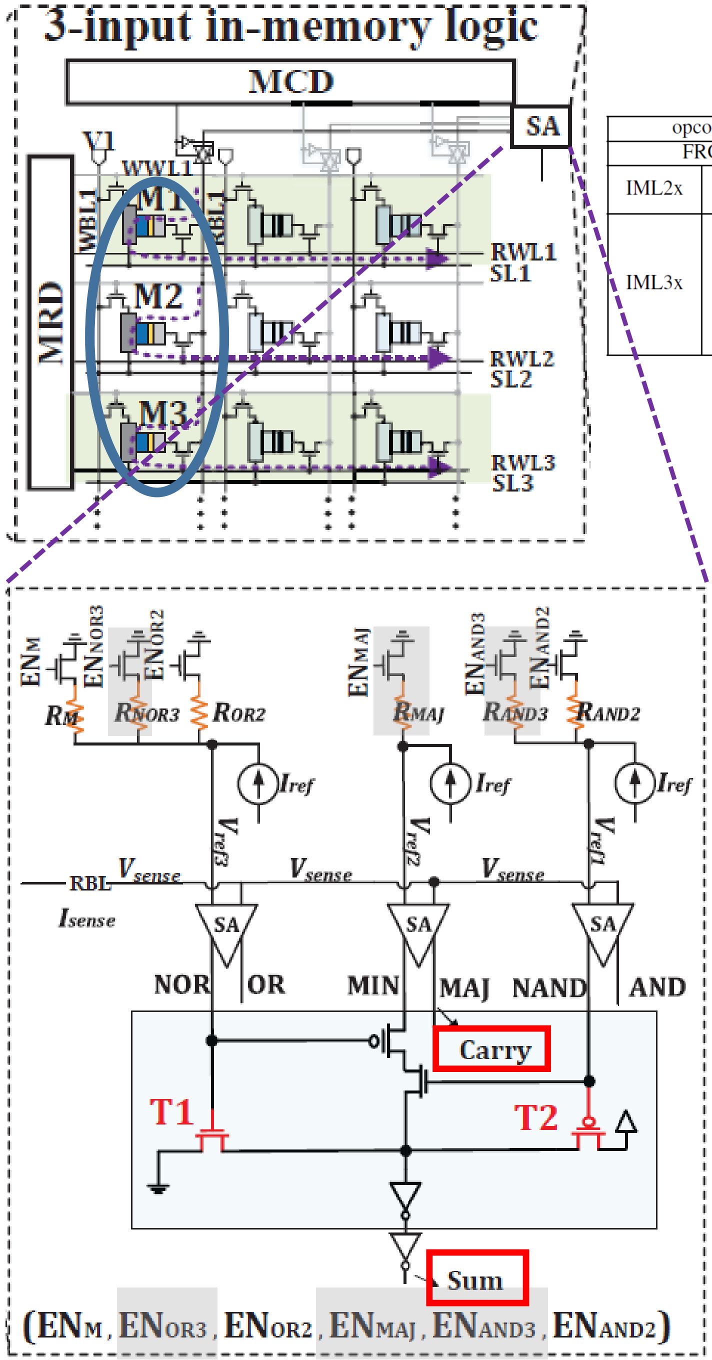
Sum = XOR3

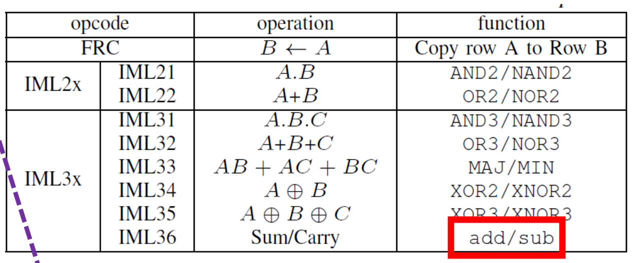
	A	B	C	$A \oplus B \oplus C$	
A=0	0	0	0	0	XOR2 $B \oplus C$
	0	0	1	1	
	0	1	0	1	
	0	1	1	0	
A=1	1	0	0	1	XNOR2 $B \odot C$
	1	0	1	0	
	1	1	0	0	
	1	1	1	1	

XOR2/XNOR2 based on XOR3/XNOR3

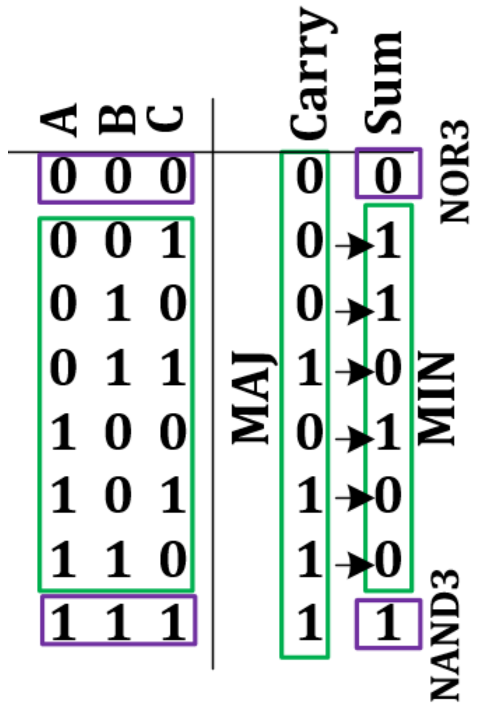
In-memory Adder (IML36)

- ***Carry*** is directly produced by MAJ function (IML33).
- ***Sum*** output is achieved by *inverted Carry signal* (MIN function) for 6 out of 8 possible input combinations.
- In two extreme cases (000 and 111), the MIN signal is disconnected and Sum is achieved by NOR3 (T1:*ON*, T2:*OFF* →Sum=0) and NAND3 (T1:*OFF*, T2:*ON* → Sum=1).
- Enable parallel One Computation per One Memory Cycle.
- Assume *M1*, *M2* and *M3* operands, 3-input in-memory logic schemes generates ***Sum***(/***Difference***) and ***Carry***(/***Borrow***) bits very efficiently.



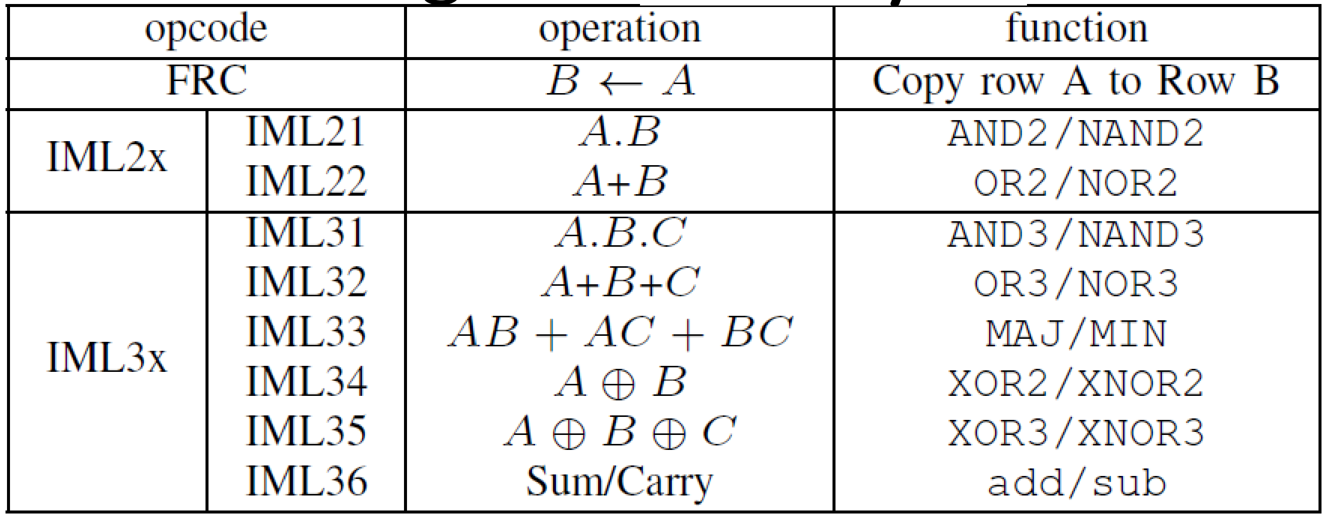


opcode		operation	function
FRC		$B \leftarrow A$	Copy row A to Row B
IML2x	IML21	$A.B$	AND2/NAND2
	IML22	$A+B$	OR2/NOR2
IML3x	IML31	$A.B.C$	AND3/NAND3
	IML32	$A+B+C$	OR3/NOR3
	IML33	$AB + AC + BC$	MAJ/MIN
	IML34	$A \oplus B$	XOR2/XNOR2
	IML35	$A \oplus B \oplus C$	XOR3/XNOR3
	IML36	Sum/Carry	add/sub


MAJ
MIN
NOR3
NAND3

A	B	C	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Up-to-Now: Supported **Parallel** and **Reconfigurable** In-Memory Logic in **ONE-cycle**

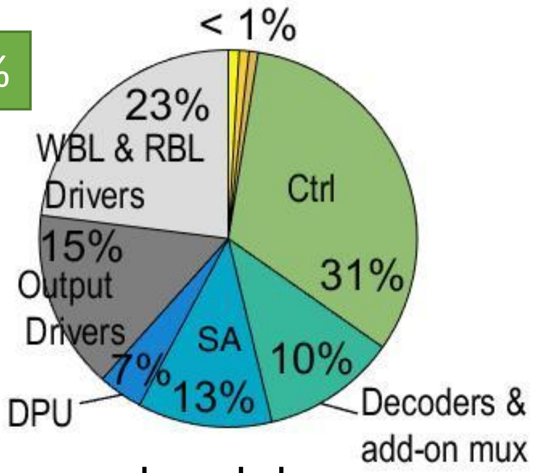


opcode		operation	function
FRC		$B \leftarrow A$	Copy row A to Row B
IML2x	IML21	$A.B$	AND2/NAND2
	IML22	$A+B$	OR2/NOR2
IML3x	IML31	$A.B.C$	AND3/NAND3
	IML32	$A+B+C$	OR3/NOR3
	IML33	$AB + AC + BC$	MAJ/MIN
	IML34	$A \oplus B$	XOR2/XNOR2
	IML35	$A \oplus B \oplus C$	XOR3/XNOR3
	IML36	Sum/Carry	add/sub

2-cycle in-memory FA and its application in DNN acceleration

D. Fan. et. al. DAC 2019, ASPDAC 2019

~5.8%



Add-on area breakdown

Area Overhead

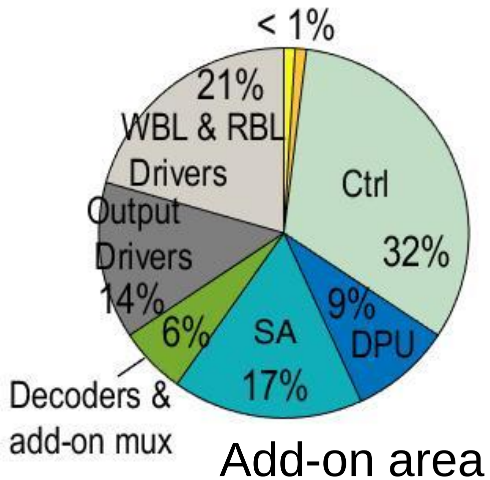
Configuration Table for a sample 512Mb memory

Parameters	Size	Activation
Compute. Sub-array	512×256	depending on in-memory OP.
Sub-array per Mat	8×8	64
Mat per Bank	2×2	2/2 and 2/2 as row and column activations
Bank per Group	4×4	1/4 and 4/4 as row and column activations

1-cycle in-memory FA and its application in Graph processing and DNA sequence analysis

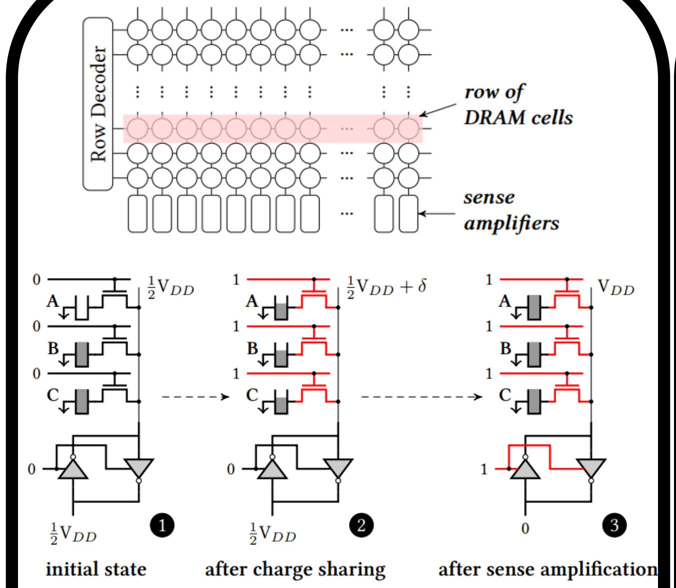
D. Fan. et. al. DATE 2019, DAC 2019

~7.9%

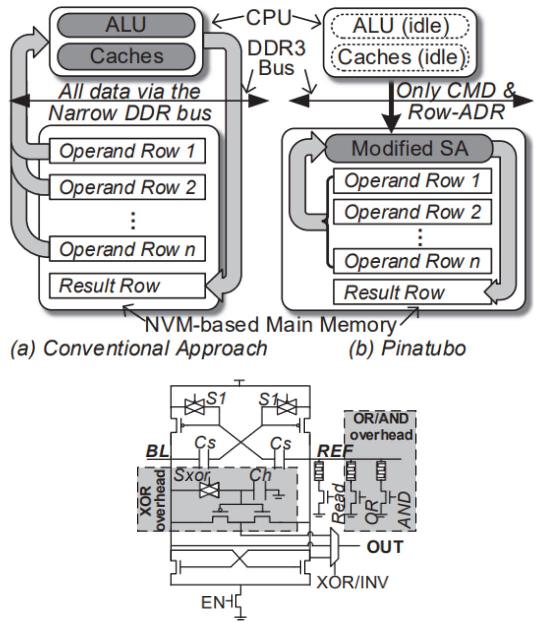


Add-on area breakdown

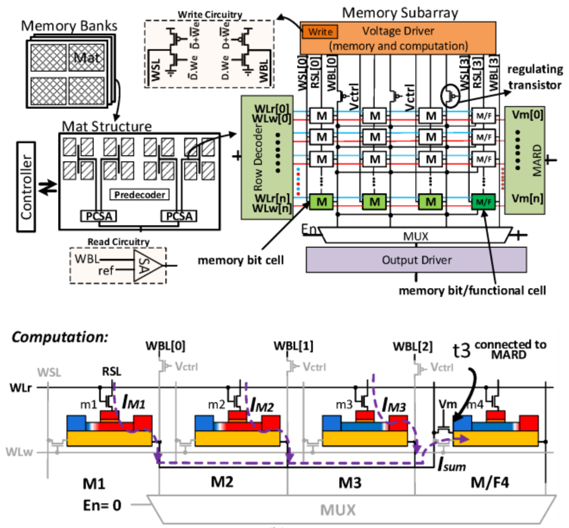
Recent Processing-in-Memory Platforms



Ambit: DRAM-based [4]



Pinatubo: NVM-based [5]



RIMPA: DWM-based [6]

Issues:

1. **Only simple logic (7+ cycles for FA, intermediate data write-back)**
2. **Operand locality**

- <u>Operand locality issue</u>
- Original data overwritten
- Multi-cycle operations
- <u>Simple logic</u>
- Low area overhead
- Hardware-friendly
- Exploit the full bandwidth

- <u>Operand locality issue</u>
- Modified SA
- <u>Simple logic</u>
- Medium area overhead
- Support one-step multi-row operations
- General platform

- <u>Operand locality issue</u>
- Large area overhead
- <u>Simple logic</u>
- Fast MG computation
- Ultra-low power

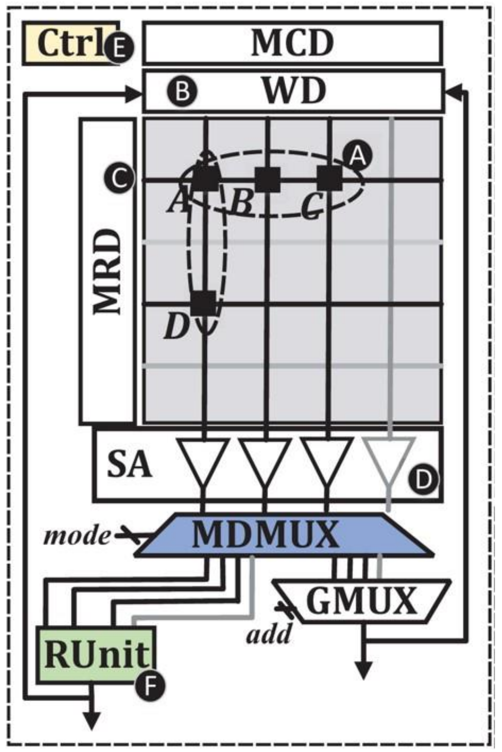
our solutions:

1. **Reconfigurable Logic-SA, one cycle logic**
2. **Operand locality ?**

[4] V. Seshadri et al., "Ambit: In-memory accelerator for bulk bitwise operations using commodity DRAM technology," MICRO, 2017, pp. 273-287: ACM.
[5] S. Li et al., "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," in 2016 53nd DAC. IEEE, 2016.
[6] S. Angizi et al. "Rimpa: A new reconfigurable dual-mode in-memory processing architecture with spin hall effect-driven domain wall motion device," *(ISVLSI)*, 2017, pp. 45-50: IEEE.

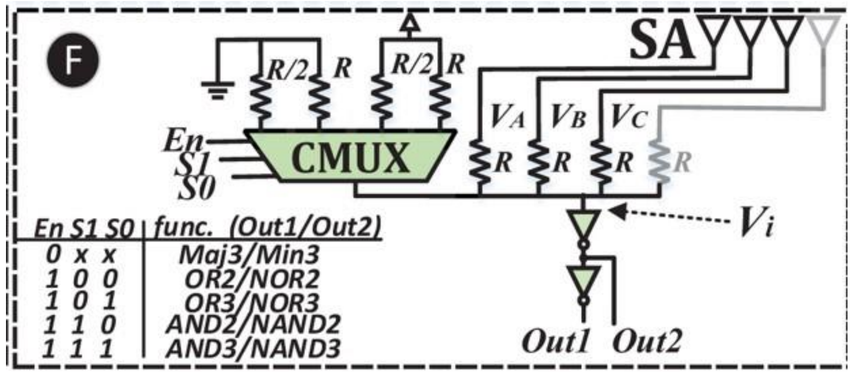
Approach-1: PIMA-Logic

D. Fan, et. al, DAC 2018



In-memory logic for data either in the same word-line or bit-line

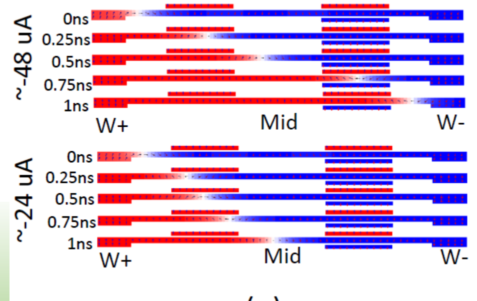
Observation Now:
Operand locality vs. Parallel computing



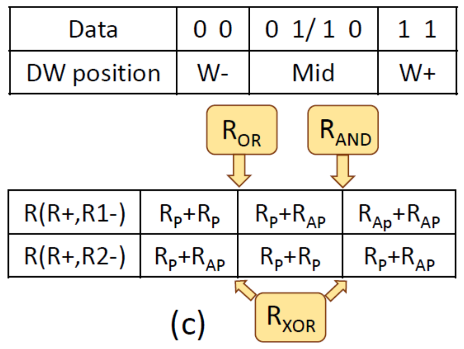
En S1 S0	func. (Out1/Out2)
0 x x	Maj3/Min3
1 0 0	OR2/NOR2
1 0 1	OR3/NOR3
1 1 0	AND2/NAND2
1 1 1	AND3/NAND3

Approach-2: Polymorphic Logic

D. Fan, et. al, ISVLSI 2017 (best paper award), ASPDAC2018



(a)



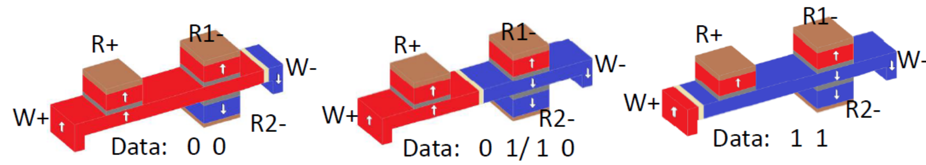
Data	0 0	0 1/ 1 0	1 1
DW position	W-	Mid	W+

R_{OR}
R_{AND}

R(R+,R1-)	R_P+R_P	R_P+R_{AP}	$R_{AP}+R_{AP}$
R(R+,R2-)	R_P+R_{AP}	R_P+R_P	R_P+R_{AP}

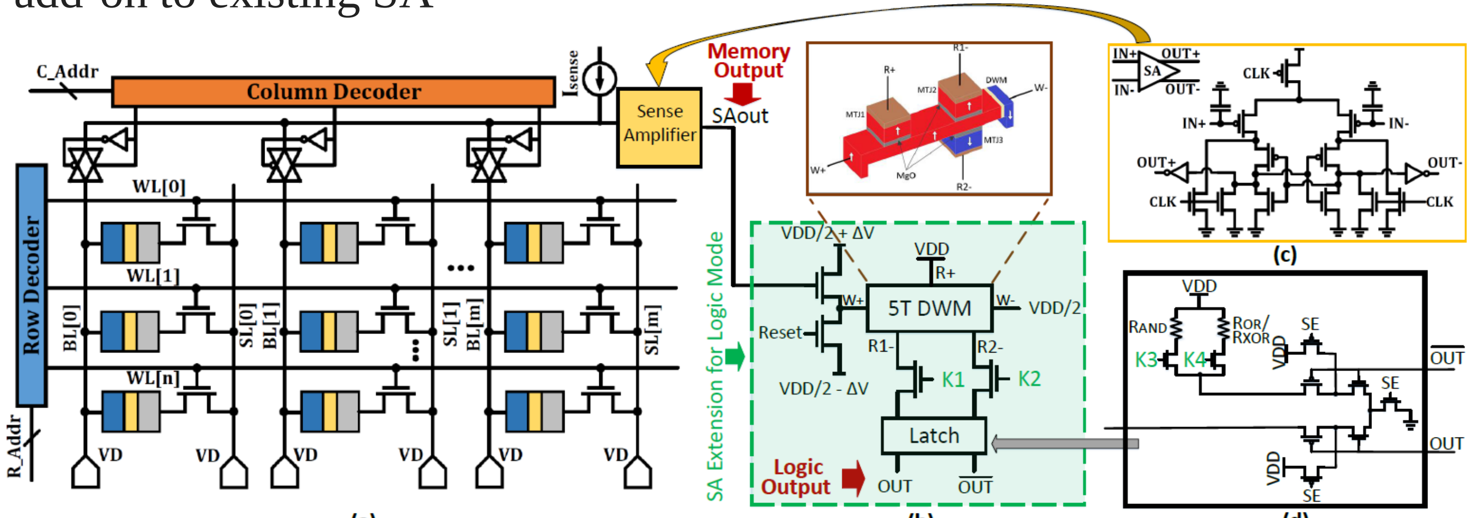
R_{XOR}

(c)



(b)

New non-volatile polymorphic logic as add-on to existing SA

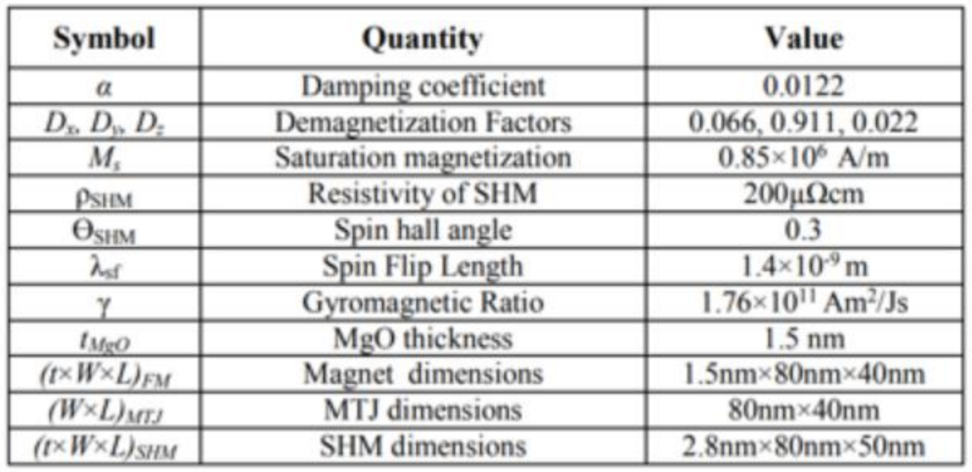


(a) (b) (c) (d)

SOT-MRAM 2T1R Device modeling and Parameters

- Area of the SOT-MRAM accelerators (in ASP-DAC 2018 [2] and DAC 2018 [3]) consists of two main components: *1- MRAM die area*, and *2- Add-on digital processing unit area*.
- *MRAM die area:*
  - ✓ *Device level:* ***1-*** SOT-MTJ device modeling (w.r.t. Table 1's parameters) and ***2-*** calculating the amount of write and sense currents.
  - ✓ *Circuit level*: ***1-*** Calculating Access Transistor size for both read (~90nm) and write (~240nm) to provide such currents. ***2-*** Developing the layout as Figure 1; Area of each two cells was determined to be (10λ × 32λ) + (10λ × 24λ) in 45 nm process node. ***3-*** Designing peripheral circuitry enabling PIM (Modified SA, Decoder, etc.) and calculating overhead area.
  - ✓ *Architectural level:* Applying the circuit-level configurations in memory scale.

Table 1. Device parameters used for modeling.



Symbol	Quantity	Value
α	Damping coefficient	0.0122
D_x, D_y, D_z	Demagnetization Factors	0.066, 0.911, 0.022
M_s	Saturation magnetization	0.85×10^6 A/m
ρ_{SHM}	Resistivity of SHM	200μΩcm
Θ_{SHM}	Spin hall angle	0.3
λ_{sf}	Spin Flip Length	1.4×10^{-9} m
γ	Gyromagnetic Ratio	1.76×10^{11} Am^2/Js
t_{MgO}	MgO thickness	1.5 nm
$(t\times W\times L)_{FM}$	Magnet dimensions	1.5nm×80nm×40nm
$(W\times L)_{MTJ}$	MTJ dimensions	80nm×40nm
$(t\times W\times L)_{SHM}$	SHM dimensions	2.8nm×80nm×50nm

- *Digital processing unit area:*
  - ✓ *Digital processing unit consists of different sub-component such as:*

    1- Activation functions, developed using lookup-table-based transformations.

    2- Batch normalization (BN) unit generally performs an affine function (*y = kx + h*) [1], where *y* and *x* denote the corresponding output and input feature map pixels, respectively. Therefore, we employed an internal, multiplexed CMOS adder and multiplier to perform this computation efficiently .

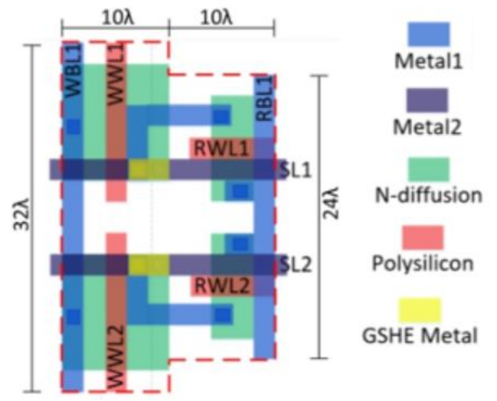


Figure 1. Layout of two SOT-MRAM cells used in our work, where λ=22.5nm.

[1] R. Zhao et al., "Accelerating binarized convolutional neural networks with software-programmable fpgas," in Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2017, pp. 15-24: ACM.
[2] S. Angizi, Z. He, F. Parveen, and D. Fan, "IMCE: Energy-efficient bit-wise in-memory convolution engine for deep neural network," in *Proceedings of the 23rd Asia and South Pacific Design Automation Conference*, 2018, pp. 111-116: IEEE Press.
[3] S. Angizi, Z. He, A. S. Rakin, and D. Fan, "CMP-PIM: an energy-efficient comparator-based processing-in-memory neural network accelerator," in *Proceedings of the 55th Annual Design Automation Conference*, 2018, p. 105: ACM

Other memory technology device parameters used in NVSIM and CACTI

- To have a fair comparison, and to explore the area, energy, latency in different PIM platforms, we first developed an ***iso-Capacity*** *32Mb-single Bank* memory unit using SOT-MRAM, STT-MRAM, RRAM, SRAM, and DRAM, as shown in next page.
- **Notes on the designs:**

✓ SOT-MRAM design is developed based on our design in [1].
✓ STT-MRAM design is developed based on our design in [1] with standard and experimentally-measured configuration available in NVSIM [2].
✓ RRAM design is developed based on [3] with standard default configuration available in NVSIM [2].
✓ SRAM design is designed based on Compute Cache [4] method with following assumptions.
✓ DRAM design is designed based on Ambit [5].

SOT-MRAM

-CellArea (F^2): 70
-ResistanceOn (ohm): 4612
-ResistanceOff (ohm): 15221
-ReadMode: current
-ReadVoltage (V): 0.042
-MinSenseVoltage (mV): 46.1
-ReadPower (uW): 21.49
-ResetMode: current
-ResetCurrent (uA): 130
-ResetPulse (ns): 1
-ResetEnergy (pJ): 0.0298
-SetMode: current
-SetCurrent (uA): 130
-SetPulse (ns): 1
-SetEnergy (pJ): 0.0298
-AccessType: CMOS
-VoltageDropAccessDevice (V): 0.0008

STT-MRAM

-CellArea (F^2): 54
-ResistanceOn (ohm): 3000
-ResistanceOff (ohm): 6000
-ReadMode: current
-ReadVoltage (V): 0.25
-MinSenseVoltage (mV): 25
-ReadPower (uW): 30
-ResetMode: current
-ResetCurrent (uA): 80
-ResetPulse (ns): 10
-ResetEnergy (pJ): 1
-SetMode: current
-SetCurrent (uA): 80
-SetPulse (ns): 10
-SetEnergy (pJ): 1
-AccessType: CMOS
-VoltageDropAccessDevice (V): 0.15

RRAM

-CellArea (F^2): 5
-CellAspectRatio: 1
-ResistanceOnAtSetVoltage (ohm): 100000
-ResistanceOffAtSetVoltage (ohm): 10000000
-ResistanceOnAtResetVoltage (ohm): 100000
-ResistanceOffAtResetVoltage (ohm): 10000000
-ResistanceOnAtReadVoltage (ohm): 1000000
-ResistanceOffAtReadVoltage (ohm): 10000000
-ResistanceOnAtHalfResetVoltage (ohm): 500000
-CapacitanceOn (F): 1e-16
-CapacitanceOff (F): 1e-16
-ReadMode: current
-ReadVoltage (V): 0.4
-ReadPower (uW): 0.16
-ResetMode: voltage
-ResetVoltage (V): 2.0
-ResetPulse (ns): 10
-ResetEnergy (pJ): 0.6
-SetMode: voltage
-SetVoltage (V): 2.0
-SetPulse (ns): 10
-SetEnergy (pJ): 0.6
-AccessType: None

SRAM

-CellArea (F^2): 146
-CellAspectRatio: 1.46
-ReadMode: voltage
-AccessType: CMOS
-AccessCMOSWidth (F): 1.31
-SRAMCellNMOSWidth (F): 2.08
-SRAMCellPMOSWidth (F): 1.23

DRAM

-CellArea (F^2): 8
-ReadMode: voltage
-AccessType: CMOS
-AccessCMOSWidth (F): 1.31

[1] S. Angizi, Z. He, A. S. Rakin, and D. Fan, "CMP-PIM: an energy-efficient comparator-based processing-in-memory neural network accelerator," in *Proceedings of the 55th Annual Design Automation Conference*, 2018, p. 105: ACM.
[2] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 31, no. 7, pp. 994-1007, 2012.
[3] T. Tang, L. Xia, B. Li, Y. Wang, and H. Yang, "Binary convolutional neural network on rram," in *Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific*, 2017, pp. 782-787: IEEE.
[4] S. Aga, S. Jeloka, A. Subramaniyan, S. Narayanasamy, D. Blaauw, and R. Das, "Compute caches," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2017, pp. 481-492: IEEE.
[5] V. Seshadri *et al.*, "Ambit: In-memory accelerator for bulk bitwise operations using commodity DRAM technology," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, 2017, pp. 273-287: ACM.

Simulation results for five different **Processing-in-Memory accelerators*****

*(**iso-capacity**: 32Mb-single Bank, Data Width: 512-bit)*

Table developed with TSMC (published in ISVLSI 2019, “Accelerating Deep Neural Networks in Processing-in-Memory Platforms: Analog or Digital Approach?”)

Metrics	***SOT-MRAM***	***STT-MRAM***	***RRAM***	***SRAM***	***DRAM***
Non-volatility	*Yes*	*Yes*	*Yes*	*No*	*No*
Area (mm^2)	Memory: 7.06 **Logic:~0.3**	Memory: 6.22 **Logic:~0.3**	**Memory: 3.34** Logic: 2.5	Memory: 10.38 Logic: 0.5	Memory: 4.53 **Logic: ~0.04**
Read Latency (*ns*)	2.85	2.89	1.48	2.9	3.4 per access
Write Latency (*ns*)	**2.59**	11.55	20.9	**2.7**	3.4 per access
Read Dynamic Energy (*nJ*)	0.57	0.65	0.38	**0.34**	0.66 per access
Write Dynamic Energy (*nJ*)	0.66	1.2	2.7	**0.38**	0.66 per access
In-Memory Logic Energy (*nJ*)	**~0.64**	~0.79	~1.96	**~0.59**	~0.75
Leakage Power (*mW*)	550	722.4	587.6	**5243**	335.5
Endurance	[1,2] ~10^{14}- 10^{15}	[1,2] ~10^{14}- 10^{15}	up to 10^{12} [3,4]	Unlimited	10^{15}
Data over-written issue	No	No	No	No	**Yes**

PIM logic area overhead including the modified decoder and SA (8-bit ADC for RRAM)

***Data is extracted using device-to-architecture simulations. The architectural level tools [5] and [6] are extensively modified based on circuit level results. Obviously by enlarging memory size, the reported numbers change correspondingly. Read latency parameter can be used as an estimation for computation latency.

[1] J. J. Kan, et al. 2016. Systematic validation of 2x nm diameter perpendicular MTJ arrays and MgO barrier for sub-10 nm embedded STT-MRAM with practically unlimited endurance. In IEEE International Electron Devices Meeting (IEDM)

[2] S. Tehrani, "Status and prospect for mram technology," in Hot Chips 22 Symposium (HCS), 2010 IEEE, 2010, pp. 1-23: IEEE.

[3] C.-W. Hsu et al., “Self-rectifying bipolar TaOx/TiO2 RRAM with superior endurance over 1012 cycles for 3D high-density storage-class memory,” in Proc. VLSIT, 2013.

[4] M.-J. Lee et al., “A fast, high-endurance and scalable nonvolatile memory device made from asymmetric Ta2O5- x/TaO2-x bilayer structures,” Nature Materials, vol. 10, no. 8, pp. 625–630, 2011

[5] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 31, no. 7, pp. 994-1007, 2012.

[6] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi, "CACTI 5.1," Technical Report HPL-2008-20, HP Labs2008.

Observations

SOT-MRAM

- **The smallest write latency while having an excellent endurance.**
- **The smallest write dynamic energy between other Processing in-NVM platforms.**
- **Medium area overhead as compared to Processing-in-RRAM platform considering the iso-capacity constraint.**
- **Small ON/OFF ratio**

STT-MRAM

- **Long write latency compared to SOT-MRAM that leads to much larger execution time specifically in write-intensive applications such as CNNs. Small ON/OFF ratio**
- **Large write dynamic energy compared to SOT-MRAM.**

ReRAM

- **Not only suffers from the low endurance but also imposes large write latency, dynamic energy and computation energy.**
- **The low endurance issue has been addressed through "Matrix Splitting" solution [1,2] by allocating excessive memory sub-arrays, sacrificing area and consuming extra write energy and latency to do the same task.**

SRAM

- **The Largest area overhead and leakage power consumption. Volatility**
- **Fast write/read**

DRAM

- **Data over-written issue. This problem has been alleviated through the back-up process [3], sacrificing area and consuming extra write energy and latency to perform the same task (e.g. it takes 6 cycles to preform AND operation).**
- **Volatility**

[1] T. Tang, L. Xia, B. Li, Y. Wang, and H. Yang, "Binary convolutional neural network on rram," in *Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific*, 2017, pp. 782-787: IEEE.
[2] P. Chi et al., "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in ACM SIGARCH Computer Architecture News, 2016, vol. 44, no. 3, pp. 27-39: IEEE Press.
[3] V. Seshadri *et al.*, "Ambit: In-memory accelerator for bulk bitwise operations using commodity DRAM technology," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, 2017, pp. 273-287: ACM.

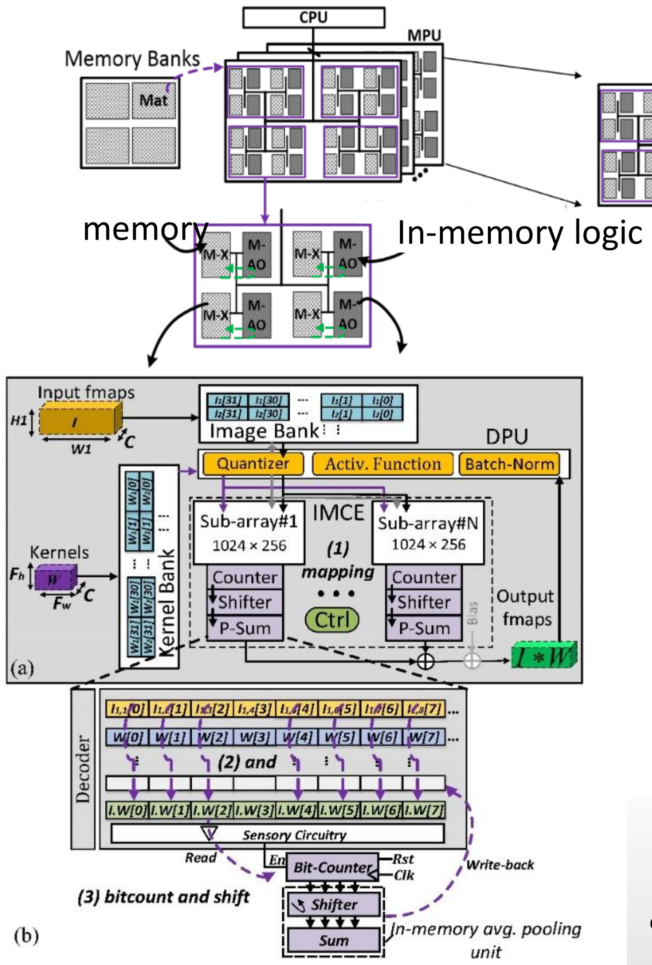
Summary

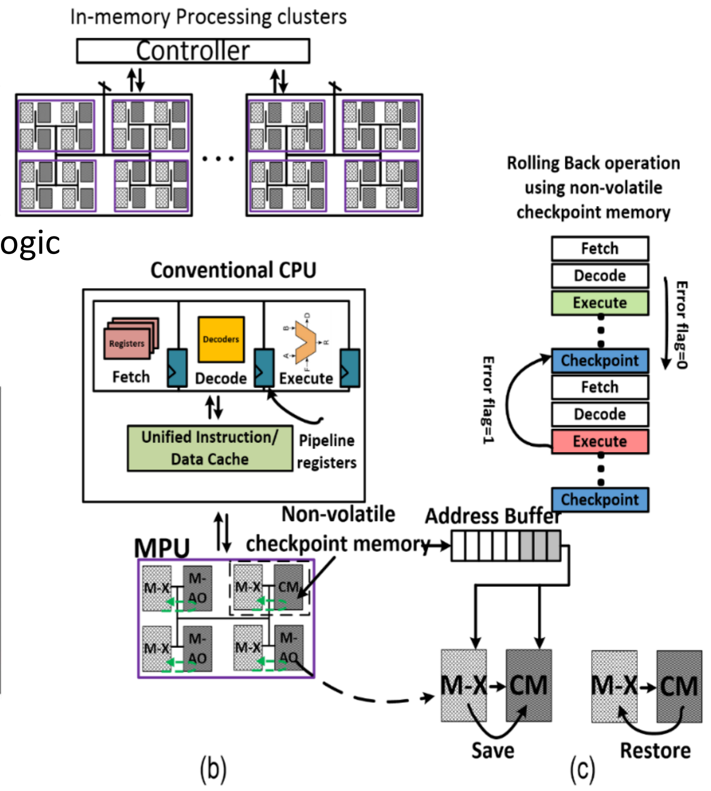
- Non-Volatile Memory, like STT-MRAM, SOT-MRAM, ReRAM, could be designed to work as dual-mode memory with both functionalities of memory and logic using innovations in device and circuit.

- With limited area overhead, we could design in-memory logic, including AND/NAND, OR/NOR, XOR/XNOR, FA in only one-cycle. It provides powerful logic functions for any further development of architectural level computational ISAs for big-data processing-in-memory accelerator designs.

- The operand locality issue in one sub-array could be solved by sacrificing the parallel computing ability of individual sub-array. It is a trade-off between specific design to choose either data rearrange to get maximal papalism or no operand locality issue

Processing-In-Memory Unit





Processing-In-Memory unit to accelerate memory/data – intensive applications.

1. Intrinsic efficient built-in in-memory logic
2. Parallel computing at each sub-array
3. Greatly reduce data communication

Challenges:

- How to design most efficient architecture to fully utilize the supported in-memory logic ISA ?
- How to modify or design new computation algorithm to make it intrinsically match with the developed PIM hardware platform

Main Research Objective and Methodology

Architecture: Parallel Processing-in-Memory Accelerator
Algorithm: Data intensive and intelligent application algorithm development for developed PIM platform
Developed and developing PIM applications: deep neural network, data encryption, image processing, graph processing

Top-Down

Algorithm and Architecture Co-Design

Architecture: In-Memory Computing + **Device & Circuits: NVM +CMOS**

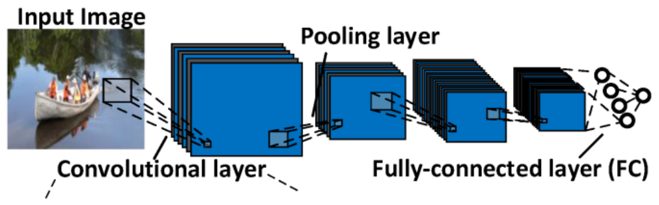
<u>Objective</u>: Energy Efficient and Intelligent Processing-in-Memory

Bottom-Up

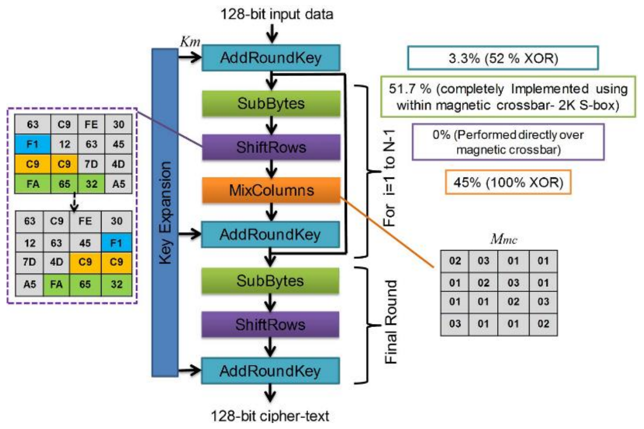
Device and Circuit Co-Design

Device & Circuit: Parallel and Reconfigurable in-Memory Logic based on STT-MRAM, SOT-MRAM, DWM, ReRAM, DRAM, with extreme low overhead.
Objective: dual mode computational memory simultaneously working as memory and in-memory logic

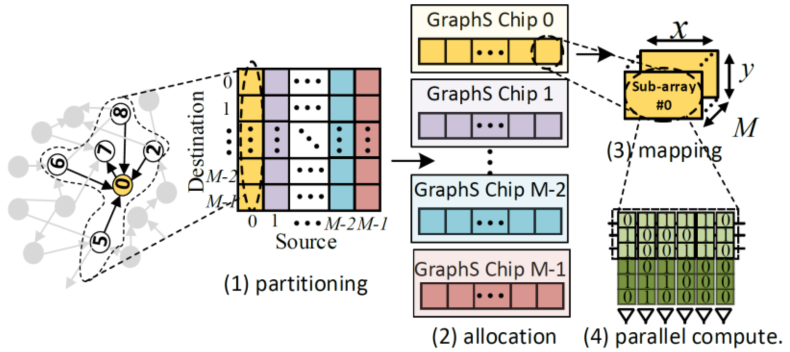
DNN-in-Memory



Data Encryption - in-Memory



Graph Processing - in-Memory



D. Fan, et. al., **CVPR** 2019, **ICCV** 2019, **DAC** 2018/2019, **ICCAD** 2018, **DATE** 2019, WACV2018/2019, ICCD 2018, ISVLSI 2018, ASPDAC 2018/2019, TCAD 2018, TNANO 2018, TMAG 2018

Outline

- ➢Top-Down: Architecture & Algorithm co-optimization for data intensive processing-in-memory acceleration
    - ➢Hardware Aware Deep Neural Network Compression: Binary or Ternary Network
    - ➢Data Encryption in memory
    - ➢Graph Processing in memory
    - ➢DNA Alignment in memory

Application: DNN-in-Memory

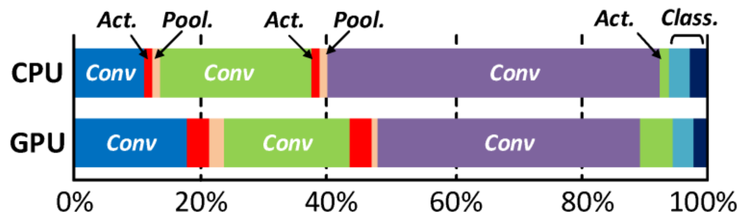
- Deep Convolutional neural networks (CNNs) are reaching record-breaking accuracy in image recognition on large data-sets like ***ImageNet*****,** ResNet shows a prominent recognition accuracy (96.43%) even higher than humans! (94.9%).

- Following the trend, when going deeper and denser in CNNs (e.g. ResNet employs 18-1001 layers), memory/computational resources and their communication have faced inevitable limitations called "CNN power and memory wall") [1,2].

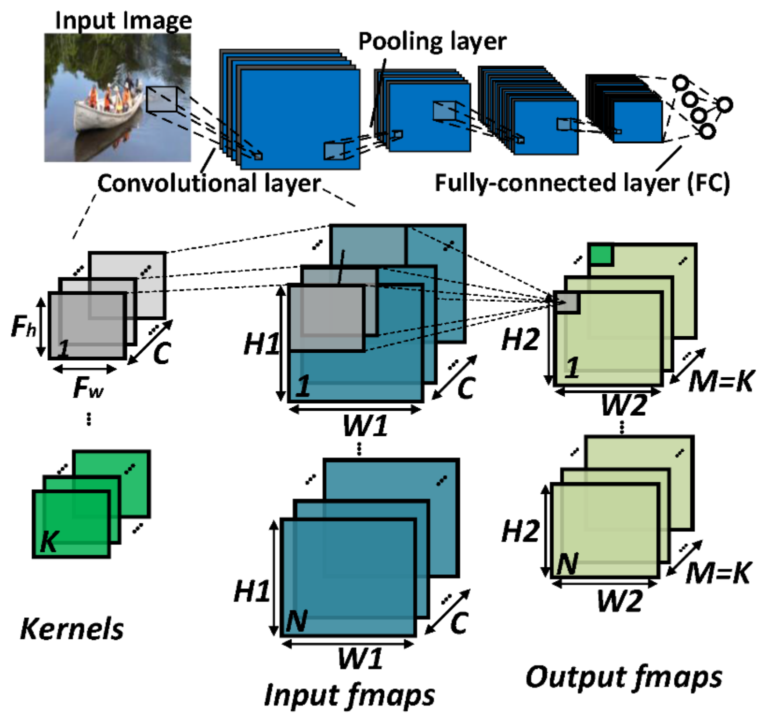
- Several methods have been proposed to break the wall:

A. Compressing pre-trained networks,
B. Quantizing parameters
C. Pruning
D. Convolution decomposition

- **Objective: Can we build a PIM hardware friendly DNN model:**
  - Remove multiplication, ideally with bit-wise logic or addition-only
  - Hardware friendly model compression
  - Without losing inference accuracy?



Execution time of a sample CNN for scene labeling on CPU and GPU [3]. Convolutional layer always takes most fraction of execute time and computational sources



Visualization of Inference in CNN

[1] R. Andriet al., "Yodann: An architecture for ultra-low power binary-weight cnn acceleration," IEEE TCAD, 2017
[2] Y.-H. Chen et al., "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," IEEE Journal of Solid-State Circuits, vol. 52, 2017.
[3] L. Cavigelli et al., "Accelerating real-time embedded scene labeling with convolutional networks," in DAC, 2015 52nd ACM/EDAC/IEEE. IEEE, 2015, pp. 1–6.

Weight Ternarization

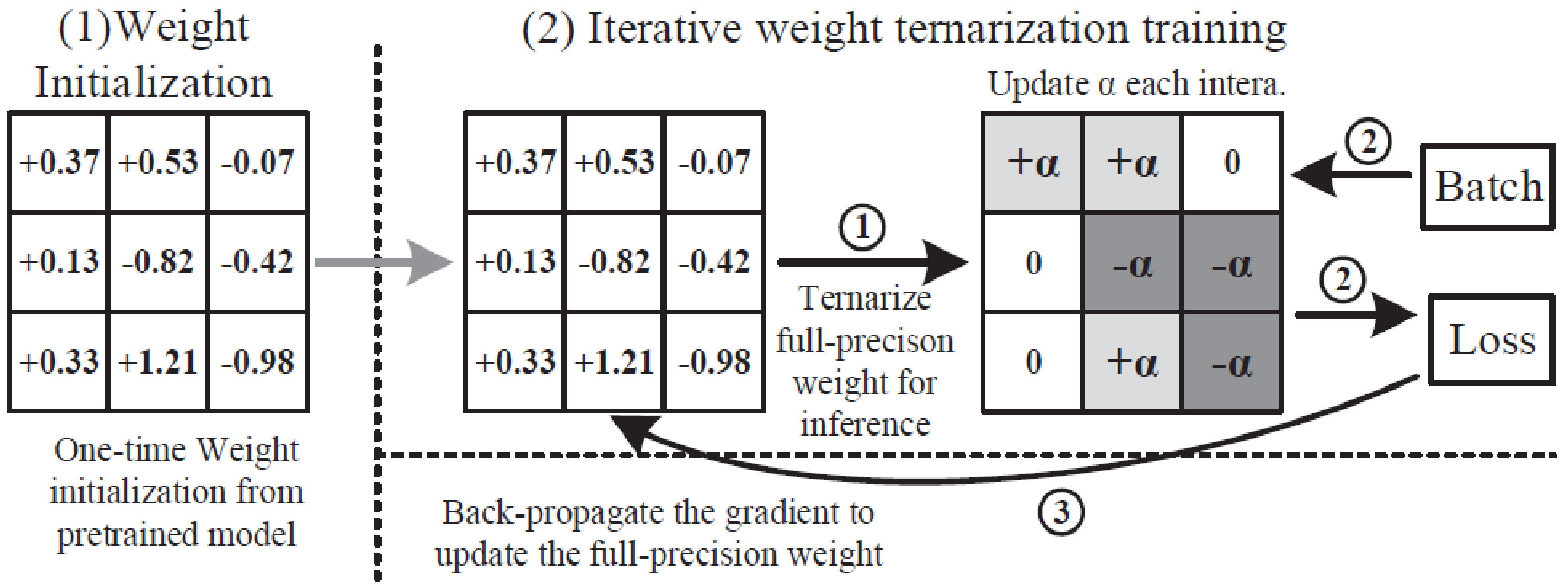
Ternarize all model weights from floating point number to {-1, 0, +1} states

Benefits and Challenges:

- Model size reduced by **16X** from 32-bit floating point number
- Convolution computation **only involves addition**, and thus computing complexity for hardware greatly reduced
- Challenge is **how to minimize** the accuracy degradation as small as possible. no degradation ideally!

D. Fan, et. al., CVPR 2019, WACV 2019
Code to download in https://github.com/elliothe/Ternarized_Neural_Network

Proposed Ternarization Method with Iterative Statistical Scaling



(1)Weight Initialization
(2) Iterative weight ternarization training
Update α each intera.
+0.37
+0.53
-0.07
+0.13
-0.82
-0.42
+0.33
+1.21
-0.98
+α
0
-α
1
Ternarize full-precison weight for inference
2
Batch
Loss
3
One-time Weight initialization from pretrained model
Back-propagate the gradient to update the full-precision weight

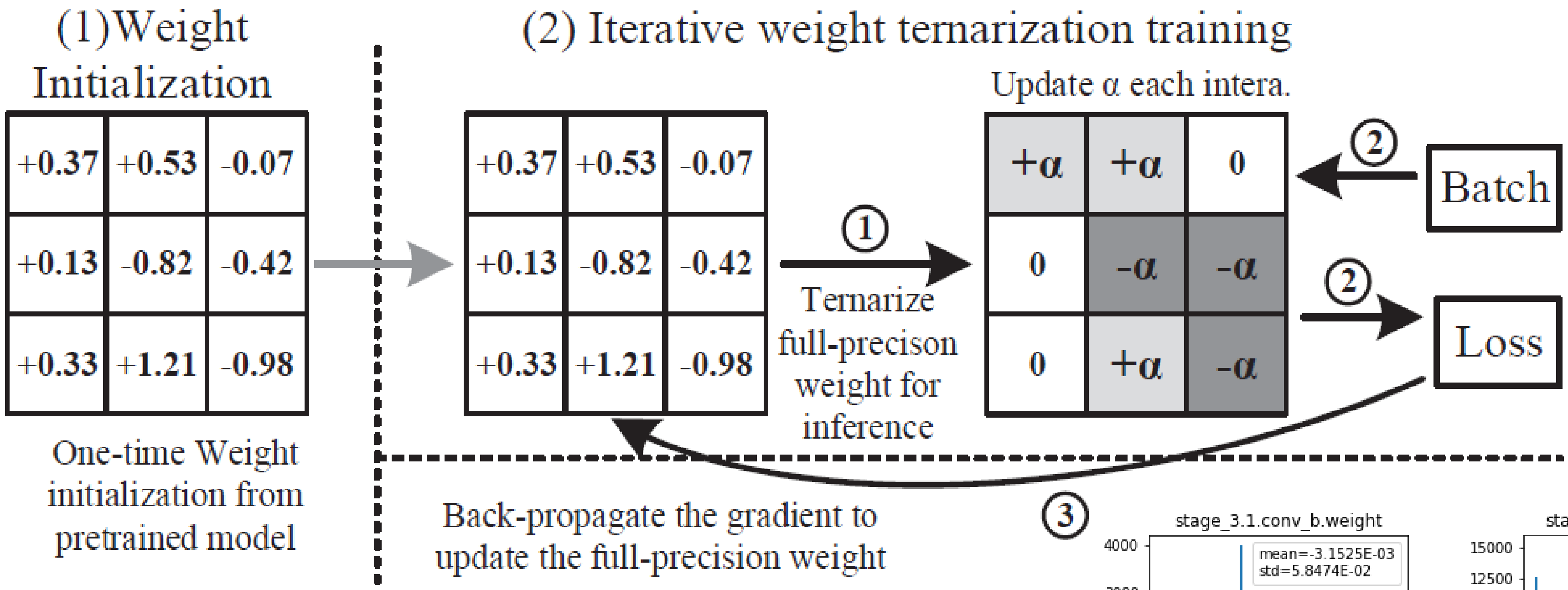
Network training step:

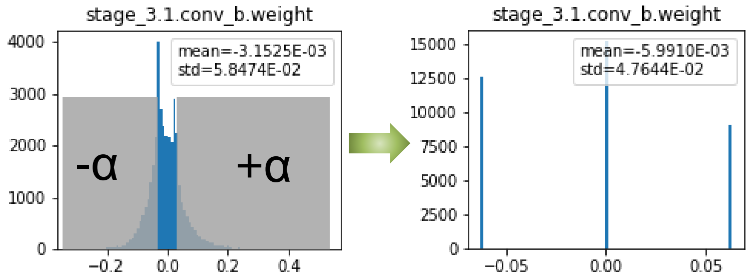
① Initialize weight with pretrained model: 1) higher accuracy; 2) converges faster than training from scratch

② Iterative weight ternarization training

③ Back propagate to update full precision weight. Note that, straight through estimator of ternarization function in the back-propagation is used to approximate gradient.

Proposed Ternarization Method with Iterative Statistical Scaling


(1)Weight Initialization
+0.37 +0.53 -0.07
+0.13 -0.82 -0.42
+0.33 +1.21 -0.98
One-time Weight initialization from pretrained model
(2) Iterative weight ternarization training
+0.37 +0.53 -0.07
+0.13 -0.82 -0.42
+0.33 +1.21 -0.98
1
Ternarize full-precison weight for inference
Update α each intera.
+α +α 0
0 -α -α
0 +α -α
2
Batch
2
Loss
3
Back-propagate the gradient to update the full-precision weight



$$\text{Forward}: \quad \boldsymbol{w}_l' = \begin{cases} \alpha \times Sign(w_{l,i}) & |w_{l,i}| \geq \Delta_{th} \\ 0 & |w_{l,i}| < \Delta_{th} \end{cases} \quad (1)$$

$$\alpha = E(|\boldsymbol{w}_{l,i}|), \quad \forall \{i \| \boldsymbol{w}_{l,i}| \geq \Delta_{th}\} \quad (2)$$

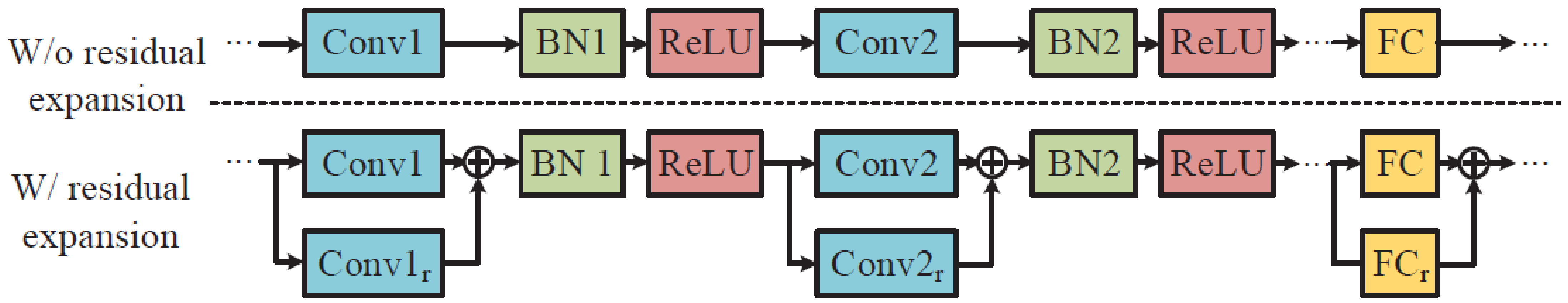
Scaling factors calculated by the mean of absolute values of designated layer's full precision weights that are greater than the thresholds

$$\boldsymbol{x}_l^T \cdot \boldsymbol{w}_l' = \boldsymbol{x}_l^T \cdot (\alpha \cdot Tern(\boldsymbol{w}_l)) = \alpha \cdot (\boldsymbol{x}_l^T \cdot Tern(\boldsymbol{w}_l)) \quad (3)$$

Convolution computation converts to ternary convolution without multiplication and reduced model size

D. Fan, et. al., CVPR 2019, WACV 2019

Residual Expansion to Improve Accuracy



We ternarize the whole network including the **first and last layer** weights

$$\boldsymbol{x}^T \cdot \boldsymbol{w}' + \boldsymbol{x}^T \cdot \boldsymbol{w}_r' = \boldsymbol{x}^T \cdot (\alpha \cdot \underset{\beta=a}{Tern}(\boldsymbol{w}) + \alpha_r \cdot \underset{\beta=b}{Tern}(\boldsymbol{w}_r))$$

- Residual Expanded Layers (REL) are added to reduce accuracy loss while maintaining no-multiplication operations in DNN.
- Original layer and residual layer are ternarized from the same full precision weights with different thresholds- *β=(a,b)*

Experiments- ImageNet

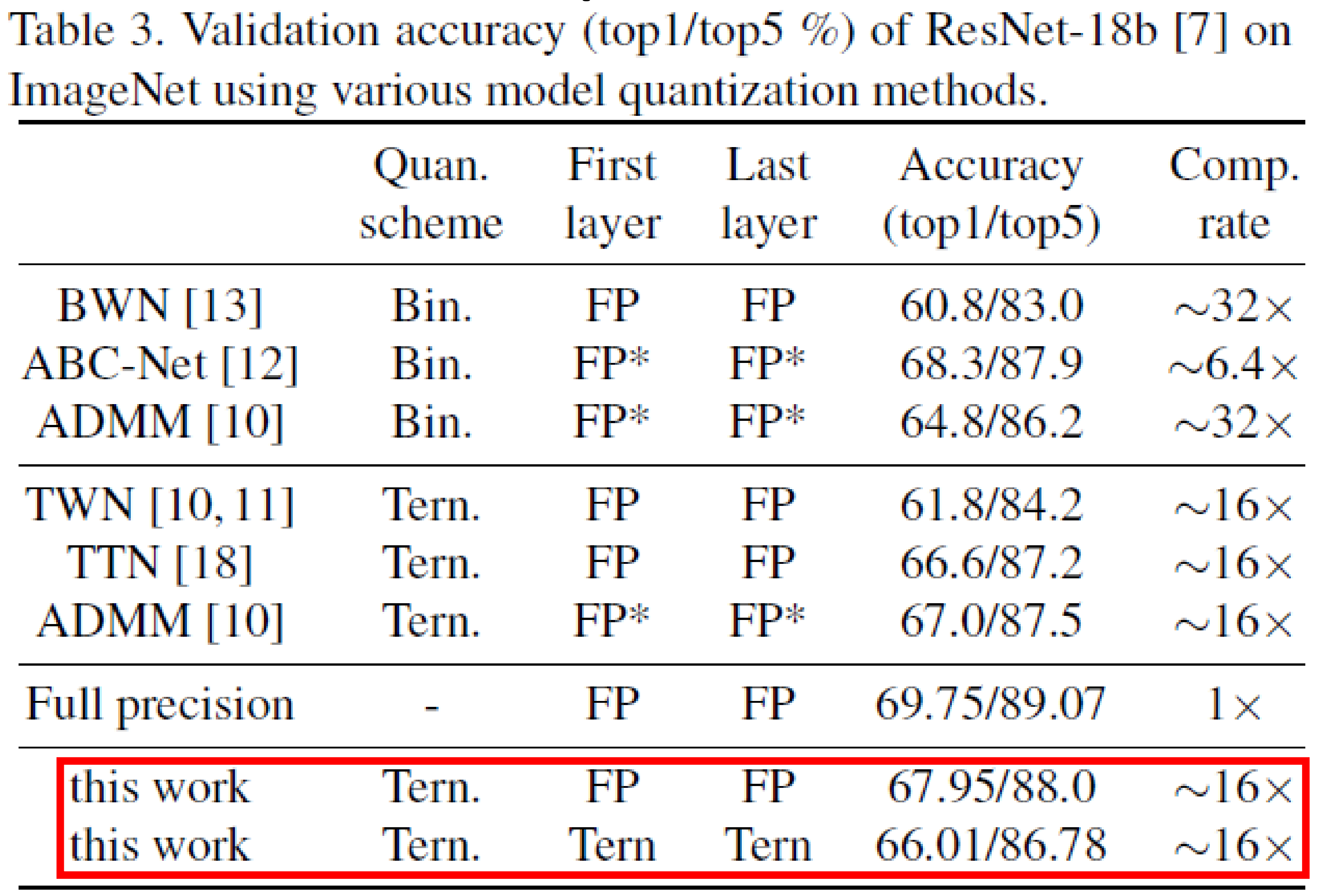


Table 3. Validation accuracy (top1/top5 %) of ResNet-18b [7] on ImageNet using various model quantization methods.

	Quan. scheme	First layer	Last layer	Accuracy (top1/top5)	Comp. rate
BWN [13]	Bin.	FP	FP	60.8/83.0	∼32×
ABC-Net [12]	Bin.	FP*	FP*	68.3/87.9	∼6.4×
ADMM [10]	Bin.	FP*	FP*	64.8/86.2	∼32×
TWN [10, 11]	Tern.	FP	FP	61.8/84.2	∼16×
TTN [18]	Tern.	FP	FP	66.6/87.2	∼16×
ADMM [10]	Tern.	FP*	FP*	67.0/87.5	∼16×
Full precision	-	FP	FP	69.75/89.07	1×
this work	Tern.	FP	FP	67.95/88.0	∼16×
this work	Tern.	Tern	Tern	66.01/86.78	∼16×

Resent structure and Imagenet datasets are used here.
14million images with 1000 output labels

FP: Full precision weights
Bin: Binary weights
Tern: Ternary weights
FP*: not reported if first and last layers are full precision

Best accuracy achieved with the same compression rate, even with ternarized first and last layers

[10] C. Leng, H. Li, S. Zhu, and R. Jin. Extremely low bit neural network: Squeeze the last bit out with admm. arXiv preprint arXiv:1707.09870, 2017.
[11] F. Li, B. Zhang, and B. Liu. Ternary weight networks. arXiv preprint arXiv:1605.04711, 2016.
[12] X. Lin, C. Zhao, andW. Pan. Towards accurate binary convolutional neural network. In Advances in Neural Information Processing Systems, pages 344–352, 2017.
[13] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnornet: Imagenet classification using binary convolutional neural networks. In European Conference on Computer Vision, 2016
[18] C. Zhu et al. Trained ternary quantization. arXiv preprint arXiv:1612.01064, 2016.

Experiments- ImageNet, with Residual layers

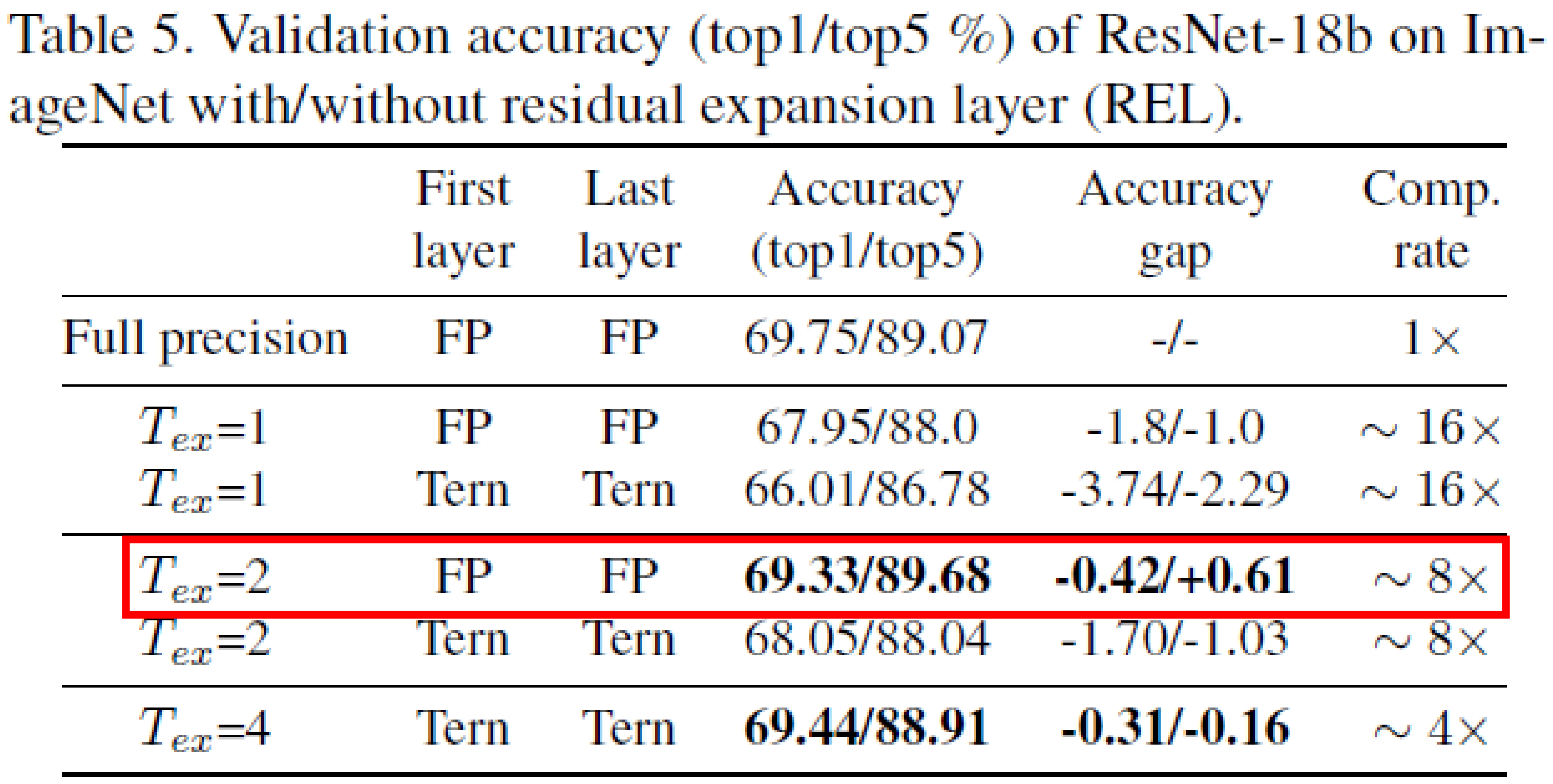


Table 5. Validation accuracy (top1/top5 %) of ResNet-18b on ImageNet with/without residual expansion layer (REL).

	First layer	Last layer	Accuracy (top1/top5)	Accuracy gap	Comp. rate
Full precision	FP	FP	69.75/89.07	-/-	$1\times$
T_{ex}=1	FP	FP	67.95/88.0	-1.8/-1.0	$\sim 16\times$
T_{ex}=1	Tern	Tern	66.01/86.78	-3.74/-2.29	$\sim 16\times$
T_{ex}=2	FP	FP	**69.33/89.68**	**-0.42/+0.61**	$\sim 8\times$
T_{ex}=2	Tern	Tern	68.05/88.04	-1.70/-1.03	$\sim 8\times$
T_{ex}=4	Tern	Tern	**69.44/88.91**	**-0.31/-0.16**	$\sim 4\times$

Resent structure and Imagenet datasets are used here
14million images with 1000 output labels

FP: Full precision weights
Bin: Binary weights
Tern: Ternary weights
FP*: not reported if first and last layers are full precision

- **Only 0.42%** accuracy degradation in imagenet if with one residual layer for **top1 accuracy**
- The top5 accuracy even **outperforms** full precision weight
- Top1 accuracy degradation reduces with more residual layers

BD-Net: A Multiplication-less DNN with Binarized Depthwise Separable Convolution

Binarize all model weights from floating point number to {-1, +1} states

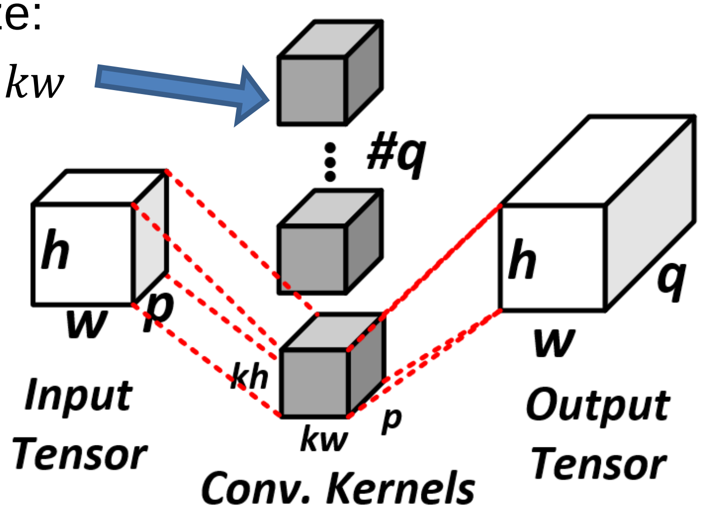
Benefits and Challenges:

- Model size reduced by at least 32X (our best results: reduced by **over 64X with only 6.59% accuracy degradation in ImageNet dataset**
- Convolution computation converts to **XNOR**, shift and bit-counter bit-wise operations, which greatly matches with our PIM hardware platform
- Real challenge is how to minimize the accuracy degradation as small as possible. no degradation ideally!

D. Fan, et. al., ISVLSI 2018 (**best paper award**), ICCAD 2018

Depthwise Separable Convolution

Kernel size:
$p \cdot q \cdot kh \cdot kw$

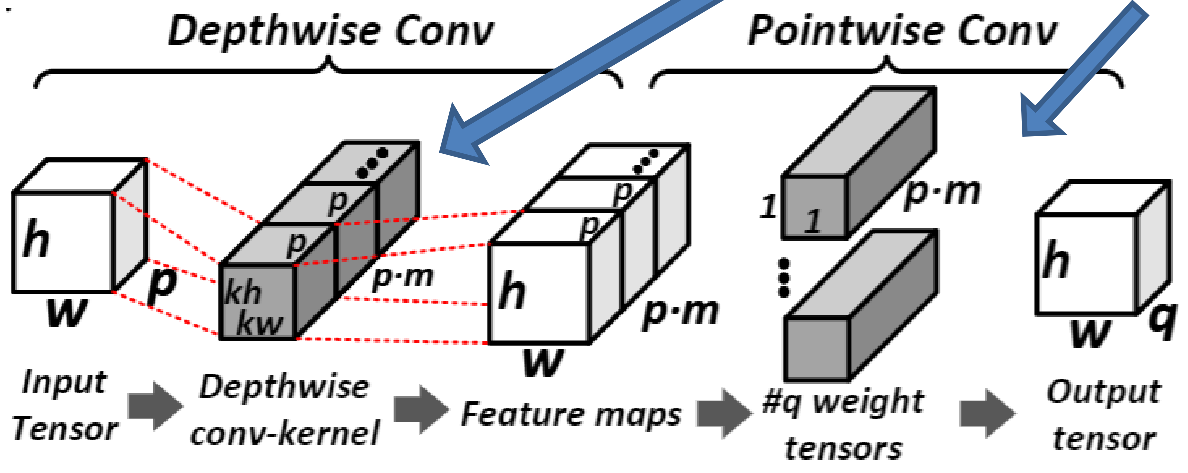


standard convolution perform the feature extraction and generate new presentation within one layer

$$Y = \sum_{i=1}^{p} \mathbf{W}_i \cdot \mathbf{X}_i \ ;$$

$$Y \in \mathrm{R}^{h \times w \times q},\ \mathbf{W} = \mathrm{R}^{kh \times kw},\ \mathbf{X} = \mathrm{R}^{h \times w}$$

Kernel size:
$p \cdot m \cdot kh \cdot kw + p \cdot m \cdot q$



$$Y = \mathbf{W}_j^{\mathrm{dw}} \cdot \mathbf{X}_i;\ \ G = \sum_{j=1}^{m \cdot p} \mathbf{W}_j^{pw} \mathbf{F}_j\ ;\ \ i \in [1, p], j \in [1, m \cdot p], Y \in \mathrm{R}^{h \times w \times pm},$$

$$\mathbf{F} = \mathrm{R}^{h \times w}, \mathbf{W}^{\mathrm{dw}} = \mathrm{R}^{kh \times kw}, \mathbf{W}^{\mathrm{pw}} = \mathrm{R}^{\mathbf{1} \times \mathbf{1}},\ \mathbf{X} = \mathrm{R}^{h \times w}, G \in \mathrm{R}^{h \times w \times q}$$

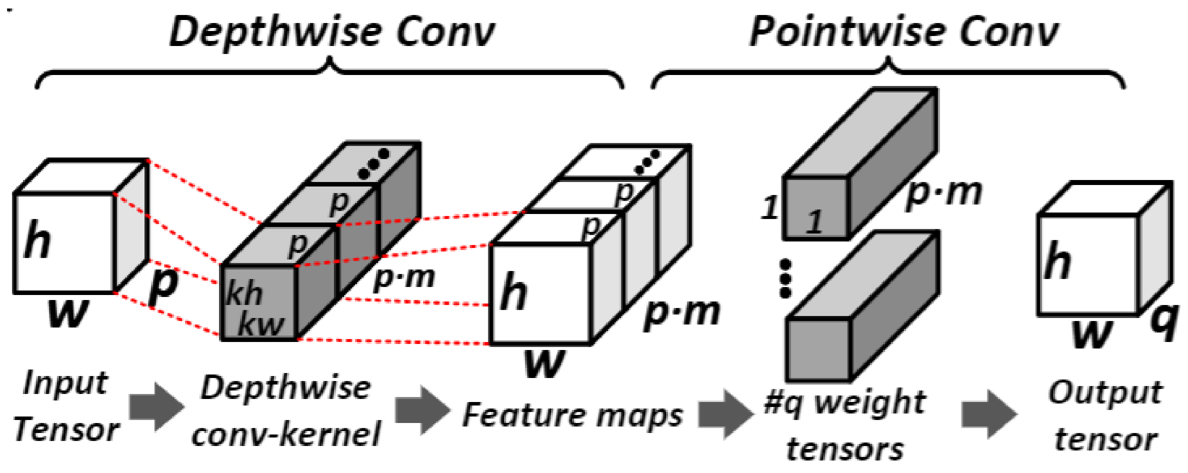
Variant Depthwise separable convolution:
- Depthwise conv: Extract features w.r.t the depthwise conv kernel.
- Pointwise conv: linearly combine the extracted feature maps to generate new representations.

$h \cdot w \cdot kh \cdot kw \cdot p \cdot q \Leftarrow$ Computational complexity $\Rightarrow h \cdot w \cdot kh \cdot kw \cdot p \cdot m + h \cdot w \cdot p \cdot m \cdot q$

** Bias is not included in Conv. layers

[1] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).

Depthwise-Separable Convolution



Depthwise conv. Pointwise conv.

$$\frac{h \cdot w \cdot kh \cdot kw \cdot p \cdot m + h \cdot w \cdot p \cdot m \cdot q}{h \cdot w \cdot kh \cdot kw \cdot p \cdot q} = \frac{m(kh \cdot kw + q)}{kh \cdot kw \cdot q}$$

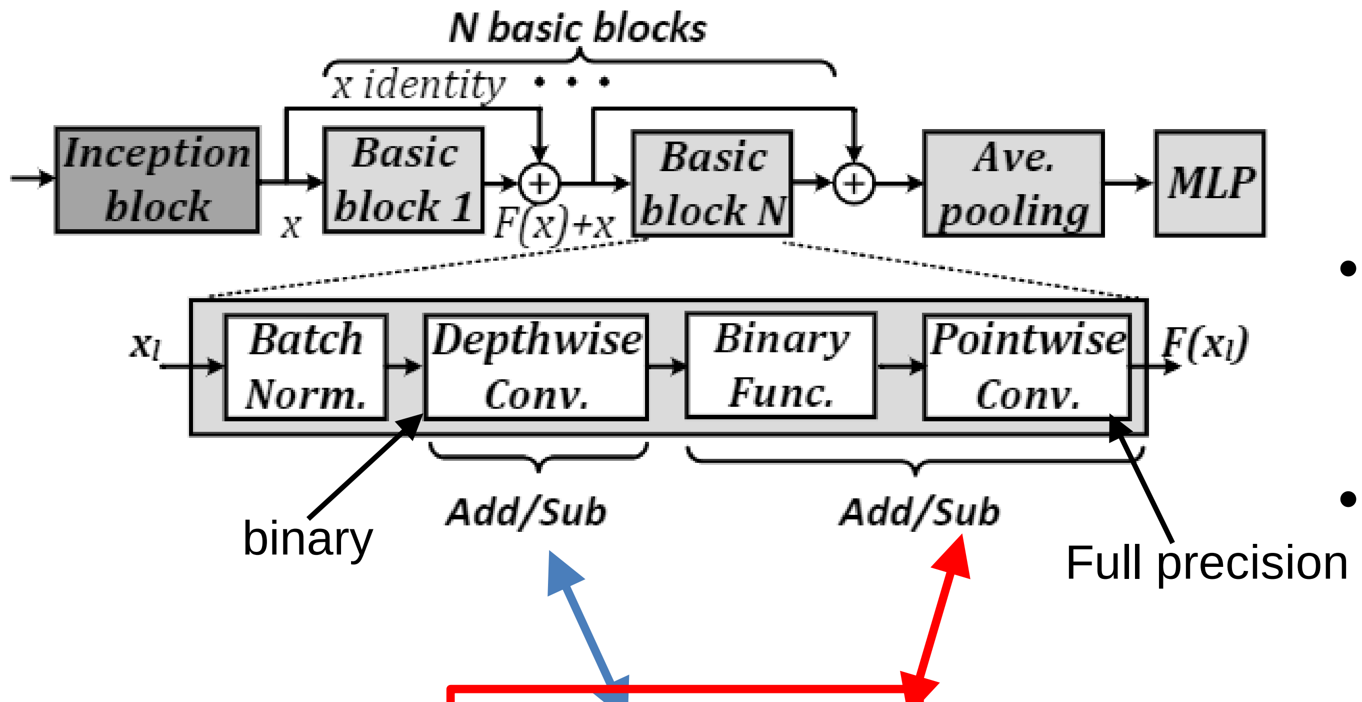
Normal conv. layer

~1/9 @ m=1 & kh=kw=3
e.g. MobileNet [1]

- #Input channel: p, #output channel: q, kernel size: kh*kw, input tensor dimension: h*w*p
- Functionality: perform the feature extraction and combination separately.
- Hardware resource: reduce the module size of convolution layer.
- Drop-in replacement of Normal spatial Convolution layer.
- **9X smaller computational cost when m=1, kh=kw=3 (mobilenet [1])**

[1] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).

BD-Net: structure

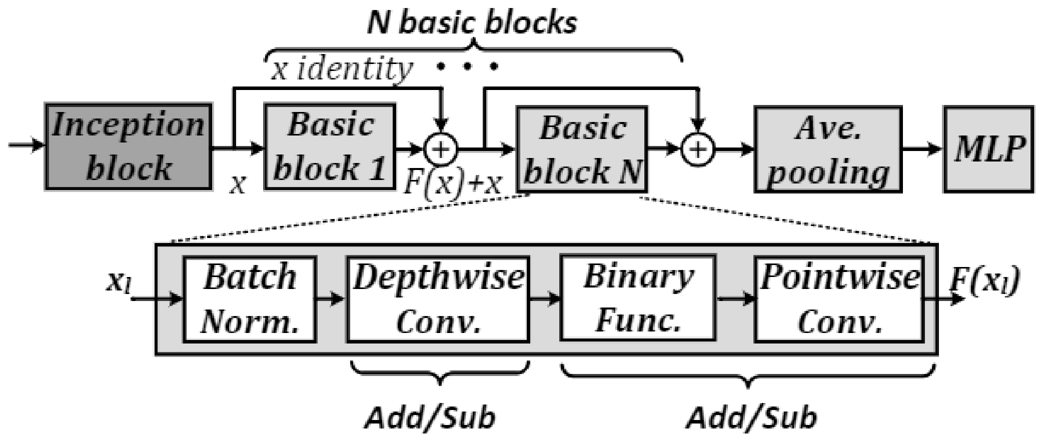


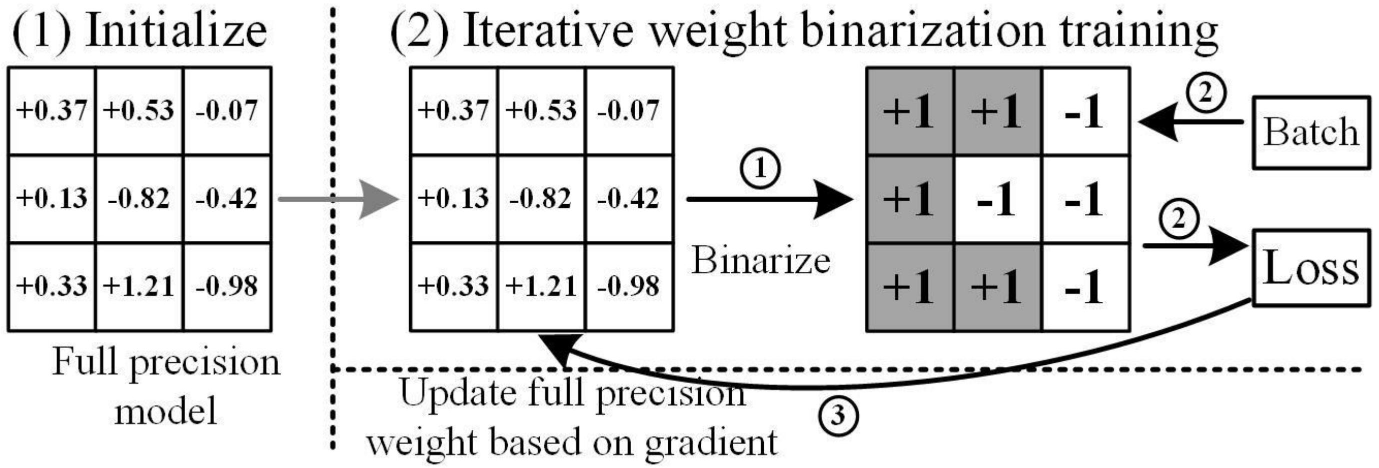
$$x_{l+1}^t = \sum_{s=1}^{p\cdot m} \boxed{Sign\left(\boxed{W_l^s} * BN(x_l^s)\right)} \cdot \alpha_{l,s}^t$$

- Remove Multiplication from Convolution Operation
- model size is further reduced by weight binarization

- Depth-wise separable convolution is efficient, can we push even more with 1) no multiplication, 2) more compact model size, 3)no accuracy lose?
- Using the bypass structure of Residual Network as back-bone.
- Replace the normal spatial convolution layer with depthwise separable convolution.
- Introduce binary weight to depthwise convolution part.
- Introduce binary intermediate tensor to pointwise convolution part.

BD-Net: training





- Using straight through estimator (STE) to approximate the gradient for making binarization function differentiable [1]

$$\text{Forward}: \quad q = Sign(r) = \begin{cases} +1 & if\ r \geq 0 \\ -1 & otherwise \end{cases}$$

$$\text{Backward}: \quad \frac{\partial g}{\partial r} = \begin{cases} \frac{\partial g}{\partial q} & if\ |r| \leq 1 \\ 0 & otherwise \end{cases}$$

- We keep the gradient clipping for better performance (i.e., inference accuracy) [2]

Network training step:

- Initialize the weight (may achieve better performance when initialize from pretrained model)
- Iterative binarize the weights of depthwise kernel
- Update the full precision weight during back-propagation

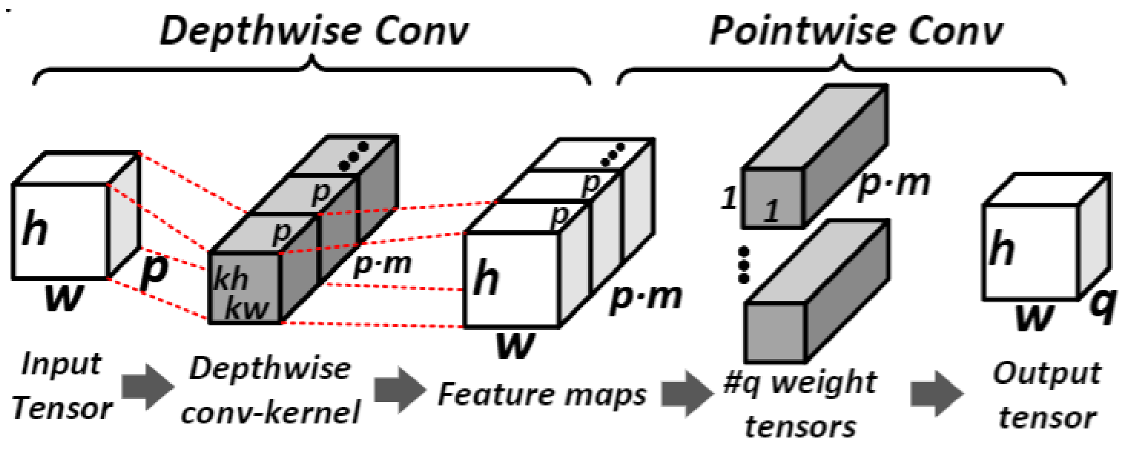
[1] Bengio, Yoshua, Nicholas Léonard, and Aaron Courville. "Estimating or propagating gradients through stochastic neurons for conditional computation." arXiv preprint arXiv:1308.3432 (2013).
[2] Courbariaux, Matthieu, et al. "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1." arXiv preprint arXiv:1602.02830 (2016).

BD-Net: hardware cost analysis

	Computation Cost		Memory Cost
	Mul–$O(N^2)$	Add/Sub–$O(N)$	
CNN	$h \cdot w \cdot kh \cdot kw \cdot p \cdot q$	$h \cdot w \cdot kh \cdot kw \cdot p \cdot q$	$kh \cdot kw \cdot p \cdot q \cdot N_{bit}^{32}$
This work	–	$h \cdot w \cdot kh \cdot kw \cdot p \cdot m$ $+h \cdot w \cdot p \cdot m \cdot q$	$kh \cdot kw \cdot p \cdot m \cdot N_{bit}^{1}$ $+p \cdot m \cdot q \cdot N_{bit}^{n}$
$\frac{This\ work}{CNN}$	0	$\frac{m}{q} + \frac{m}{kh \cdot kw}$	$\frac{1}{q \cdot N_{bit}^{32}} + \frac{m \cdot N_{bit}^{n}}{kh \cdot kw \cdot N_{bit}^{32}}$

~1/9 when kh=kw=3, m=1

>1/9 depending on bit-width of pointwise kernal

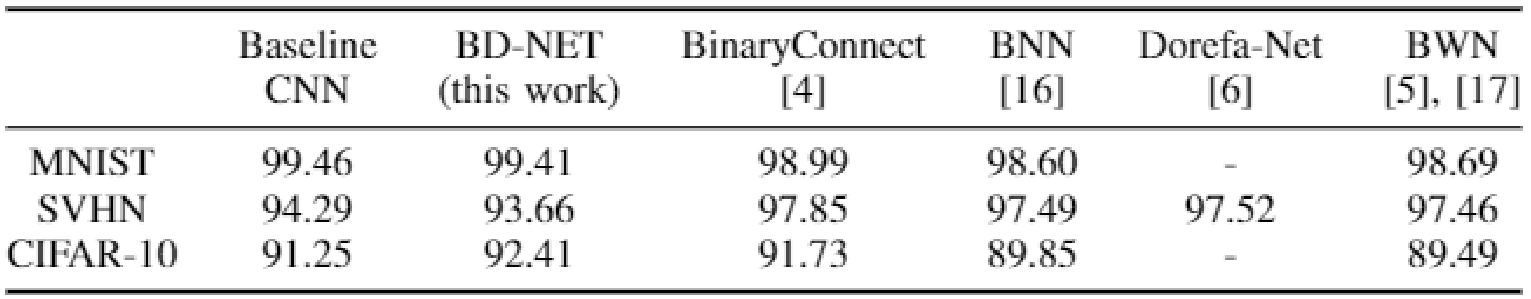


- #Input channel: *p*, #output channel: *q*, kernel size: *kh*kw*, input tensor dimension: *h*w*p*, channel multiplier: *m*
- N_{bit} is the number of bits
- We use 32bit ($N_{bit}^{n=32}$) for pointwise layer in this work.
- Channel multiplier *m* is the hyperparameters to optimize in this work

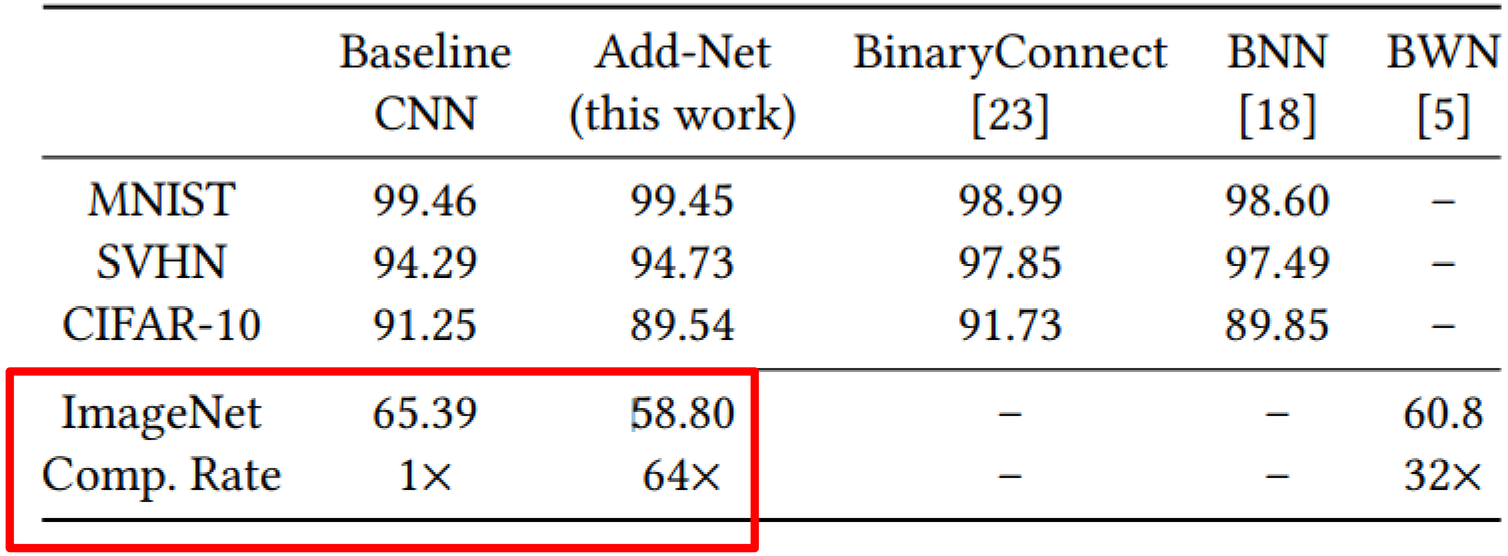
Experiments: Cifar and ImageNet

Framework: Pytorch (Good support for depthwise convolution)
Application: Object classification

Network configuration:
MNIST: 16 input channels, 5 basic blocks, 128 hidden neuron, 64 batch size, 3×3 kernel size, 4 channel expansion.
SVHN: 128 input channels, 5 basic blocks 512 hidden neuron, 64 batch size, 3×3 kernel size, 4 channel expansion.
CIFAR-10: 128 input channels, 5 basic blocks, 512 hidden neuron, 64 batch size, 3×3 kernel size, 4 channel expansion.
ImageNet: ResNet-18 structure. 14million iamges with 1000 output labels



	Baseline CNN	BD-NET (this work)	BinaryConnect [4]	BNN [16]	Dorefa-Net [6]	BWN [5], [17]
MNIST	99.46	99.41	98.99	98.60	-	98.69
SVHN	94.29	93.66	97.85	97.49	97.52	97.46
CIFAR-10	91.25	92.41	91.73	89.85	-	89.49

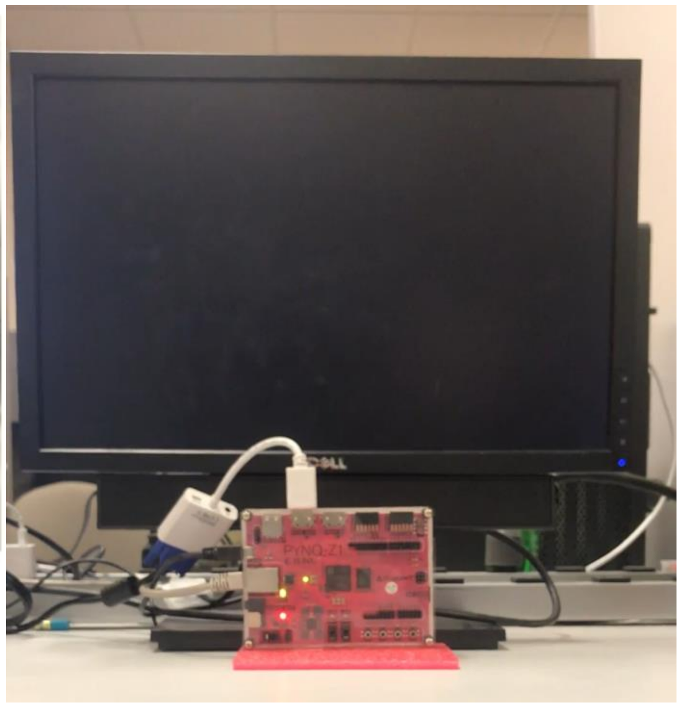


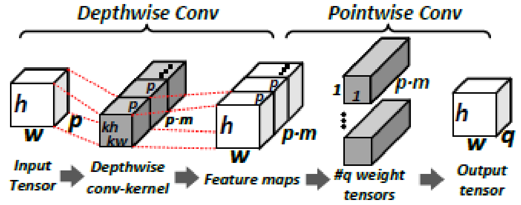
	Baseline CNN	Add-Net (this work)	BinaryConnect [23]	BNN [18]	BWN [5]
MNIST	99.46	99.45	98.99	98.60	–
SVHN	94.29	94.73	97.85	97.49	–
CIFAR-10	91.25	89.54	91.73	89.85	–
ImageNet	65.39	58.80	–	–	60.8
Comp. Rate	1×	64×	–	–	32×

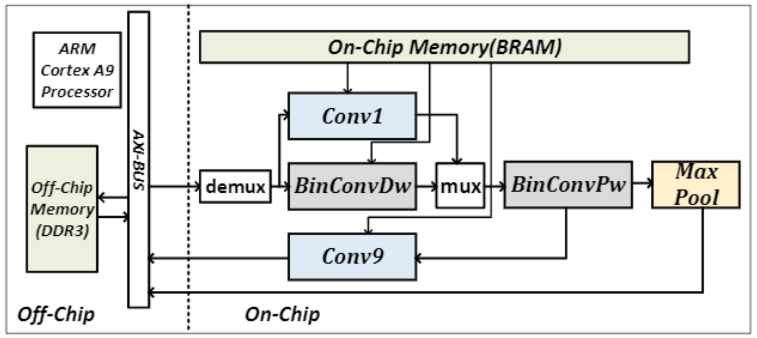
64X compression rate achieved with **only 6.59%** accuracy degradation in ImageNet dataset

D. Fan, et. al., ISVLSI2018 (**best paper award**), ICCAD2018

Binarized Deep Neural Network FPGA Demo







Name	IOU	Power	FPS	ES	TS
TGIIF	0.62	4.2	11.955	1.0318	1.2674
SystemsETHZ	0.49	2.45	25.968	1.3976	1.1794
iSmart2	0.57	2.59	7.349	1.0297	1.1636
traix	0.61	3.11	5.445	0.8869	1.1523
hwac-object-tracker	0.52	3.66	4.935	0.8155	0.932
Ours	0.57	2.61	11.1	1.1477	**1.224**

- Our model is only **143Kb** with 8 conv layers and 1 FC layer
- DNN model completely stored in on-chip cache, no need to fetch model from main memory
- PYNQ-Z1 only has 4.9Mb on chip RAM and our model only consumes **2.61 W**

D. Fan, et. al., GLSVLSI 2019

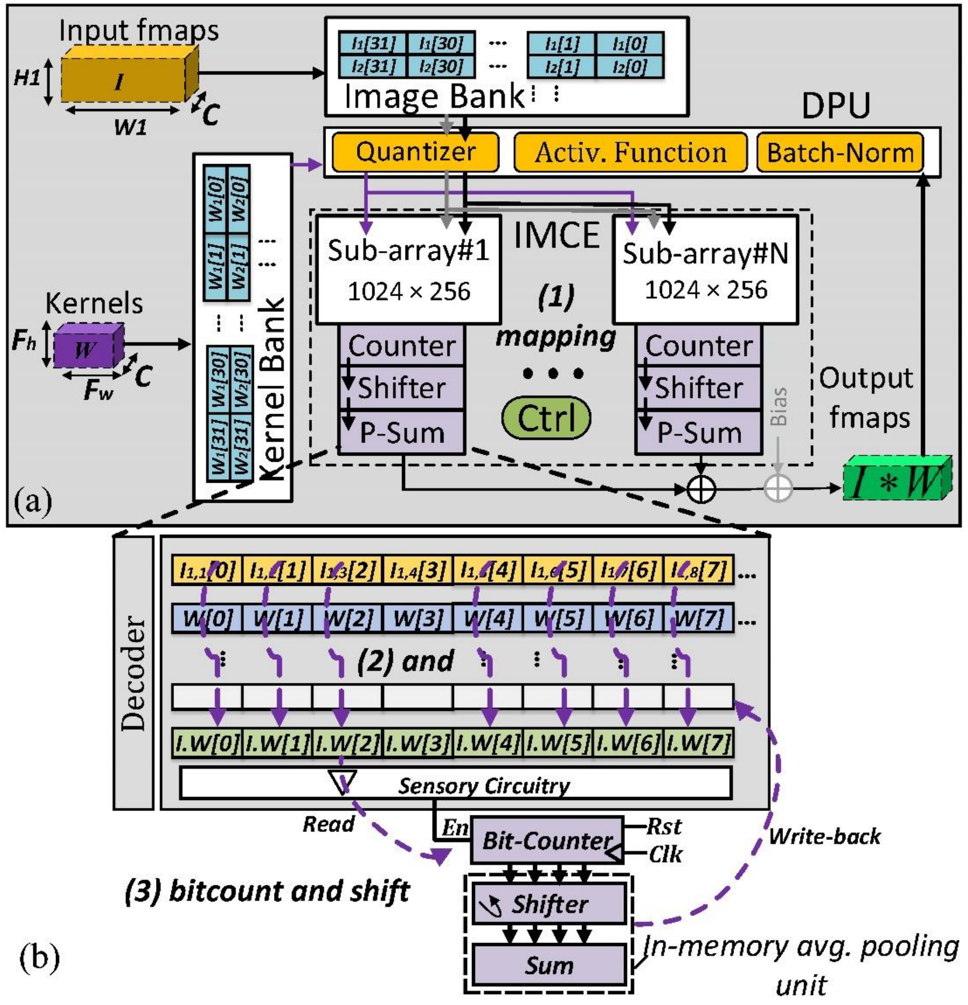
In-Memory Convolution Engine

- A potential solution to better address storage, computation, and data transfer bottlenecks of CNNs.
- This architecture mainly consists of Image Bank, Kernel Bank, bit-wise In-Memory Convolution Engine (IMCE), and Digital Processing Unit (DPU).
- Preprocessing:

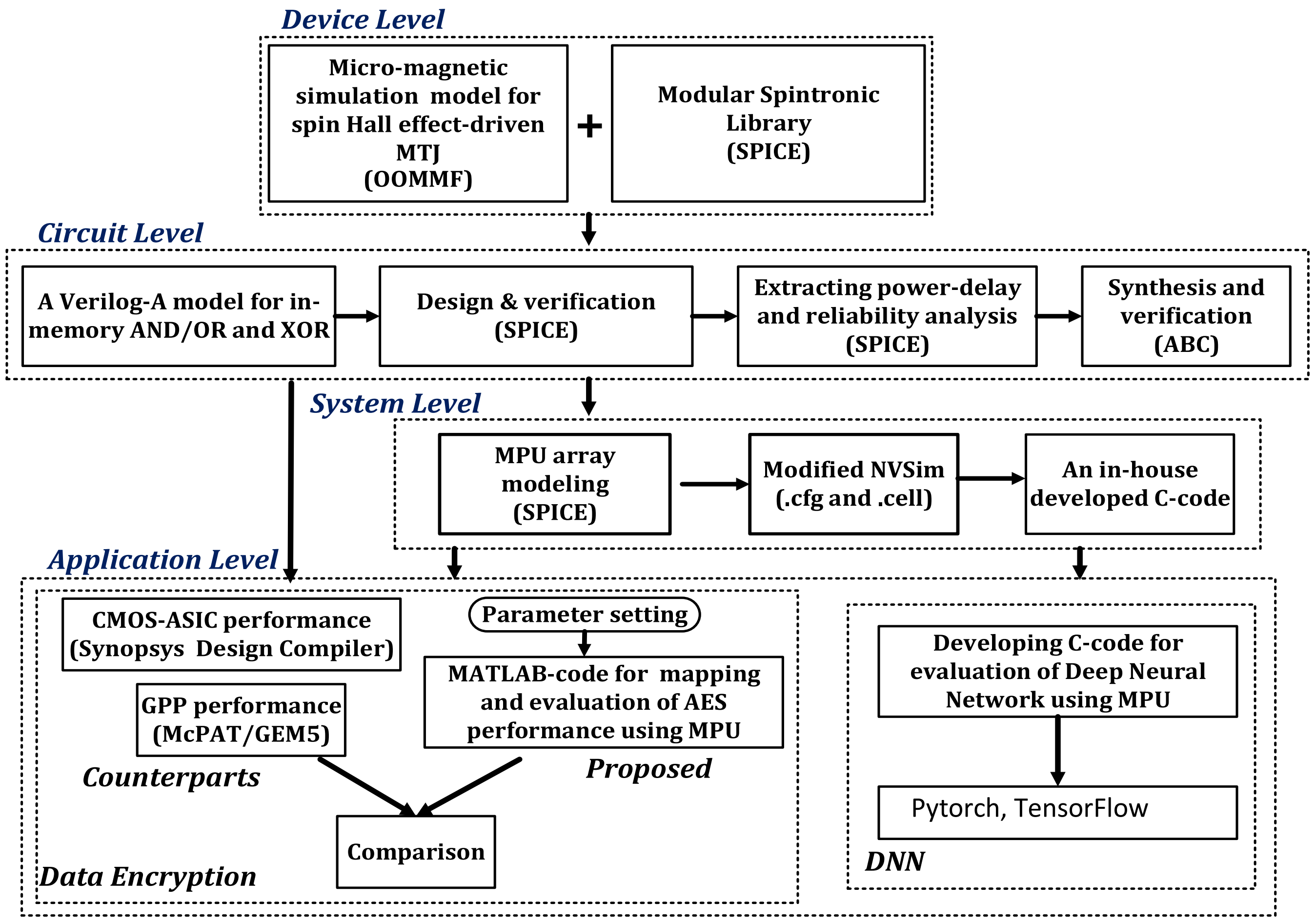
✓ Assume Input fmaps (I) and Kernels (W) are stored in Image Banks and Kernel Banks of memory.

✓ Inputs need to be constantly quantized before mapping into computational sub-arrays. This step is performed using DPU's Quantizer and then the results are mapped to IMCE's sub-arrays.

✓ IMCE is realized through the proposed SOT-MRAM based computational sub-array.



Cross-Layer Simulation Framework Development


Device Level
Micro-magnetic simulation model for spin Hall effect-driven MTJ (OOMMF)
+
Modular Spintronic Library (SPICE)
Circuit Level
A Verilog-A model for in-memory AND/OR and XOR
Design & verification (SPICE)
Extracting power-delay and reliability analysis (SPICE)
Synthesis and verification (ABC)
System Level
MPU array modeling (SPICE)
Modified NVSim (.cfg and .cell)
An in-house developed C-code
Application Level
CMOS-ASIC performance (Synopsys Design Compiler)
GPP performance (McPAT/GEM5)
Counterparts
Parameter setting
MATLAB-code for mapping and evaluation of AES performance using MPU
Proposed
Comparison
Data Encryption
Developing C-code for evaluation of Deep Neural Network using MPU
Pytorch, TensorFlow
DNN

<u>Device Level:</u>

Verilog-A model of spintronic device developed based on micromagnetic OOMMF framework， modular spintronic library.

<u>Circuit Level:</u>

SPICE simulation to verify logic design and extract power-delay-reliability analysis

<u>System Level:</u>

modified self-consistent NVSim along with an in-house developed C++ code to verify the performance， Gem5 will be used to build cycle-accurate in-memory processing unit architecture.

<u>Application Level:</u>

quantized deep convolution neural network and Advanced Encryption Standard (AES) algorithm are used as case study applications, to show benefits of PIM in practical applications

[12] www:eda.ncsu.edu/wiki/FreePDK45

[13] X. Dong et al., "NVSim: A circuit-level performance, energy, and area model for emerging non-volatile memory," Springer, 2014, pp. 15-50.

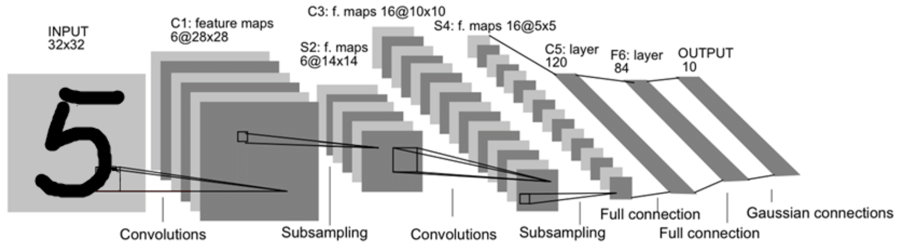
Comparison (**iso-computation (CNN as example)**, Area-Energy-Latency requirement)

- **Taking LeNET to run MNIST data-set with different PIM accelerators considering the area/energy due to the computation by calculating the number of undertaken crossbars or sub-arrays.**

- **STT-MRAM/ReRAM/DRAM design imposes a larger latency compared with SOT-MRAM mainly due to its long intermediate data write-back.**

- **ReRAM design data is taken from [1]. While the required memory area of ReRAM is less than magnetic counterparts, overall it imposes larger area due to matrix splitting and extra large add-on logic area overhead (up to ~80%) [1,2].**

- **While DRAM accelerator imposes the least possible area compared to other PIMs with iso-capacity constraint (~1%), it needs to access multiple sub-arrays to avoid data-written problem as well as fitting the network at the same time that resulted in a larger latency and area compared to non-volatile designs.**



An early (Le-Net5) Convolutional Neural Network design, LeNet-5, used for recognition of digits

For in-memory-logic, all operands are stored in memory. Unlike traditional computation, an extra data write-back is needed, which has a large effect on determining the overall energy and latency

Estimated row Performance of different PIMs without Parallelism techniques

Parameters	**SOT-MRAM**	**STT-MRAM**	**ReRAM**	**SRAM**	**DRAM**
Area (mm^2) (memory + logic)	0.018 (0.0172+0.0008)	0.015 (0.0143+0.0007)	0.060 (0.011+0.049)	0.64 (0.608+0.032)	0.16 (0.158 + 0.002)
Energy (μJ) (write-back+read-based Ops)	0.74 ~(0.2+0.54)	1.3 ~(0.55+0.75)	13.5 ~(5.1+8.4)	1.6 ~(0.42+1.18)	2.1 ~(0.8+1.3)
Latency (ms)	0.4	2.6	5.8	0.7	13.5

[1] T. Tang, L. Xia, B. Li, Y. Wang, and H. Yang, "Binary convolutional neural network on rram," in *Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific*, 2017, pp. 782-787: IEEE.
[2] P. Chi et al., "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in ACM SIGARCH Computer Architecture News, 2016, vol. 44, no. 3, pp. 27-39: IEEE Press.
[3] V. Seshadri *et al.*, "Ambit: In-memory accelerator for bulk bitwise operations using commodity DRAM technology," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, 2017, pp. 273-287: ACM.

Performance Evaluation

ICCD 2018 [9]

256×512 columns per mat

512Mb total capacity

Vs.

ReRAM

A Prime-like [10] accelerator

GPU

NVIDIA GTX 1080Ti Pascal

ASIC

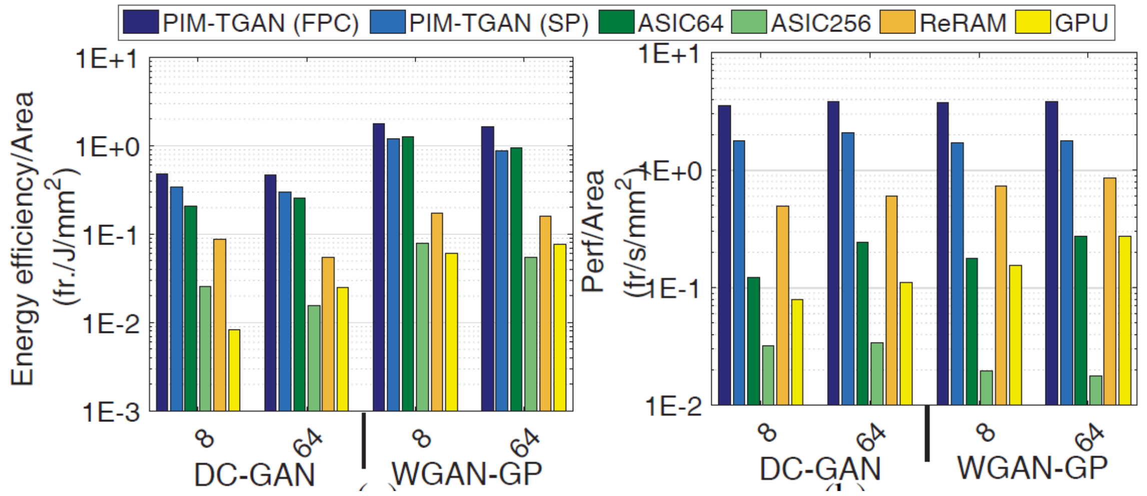
A YodaNN-like [11] ASIC accelerator

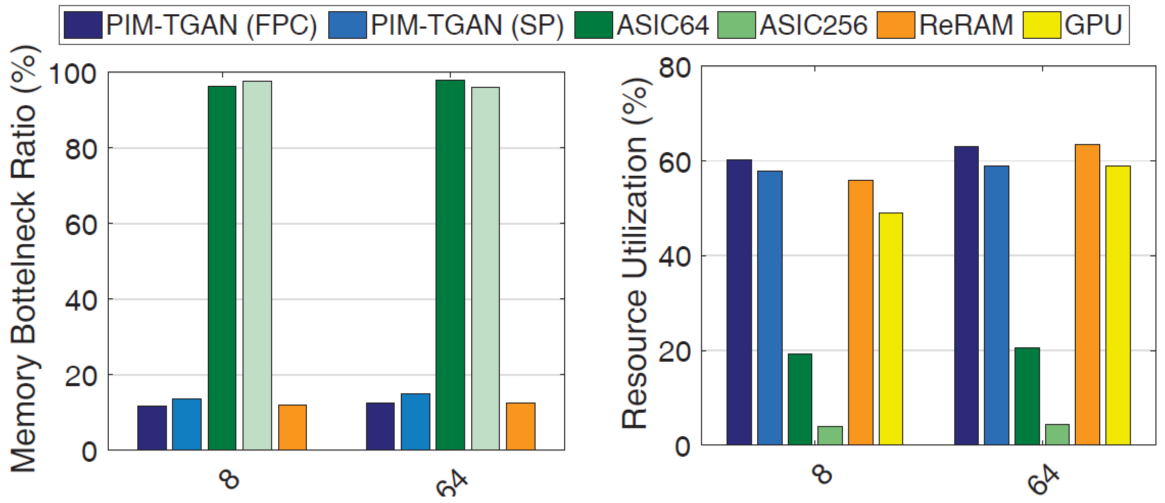
Energy Efficiency

- 18x ASIC64
- 9.2x ReRAM
- 25.6x GPU

Performance

- 18x ASIC64
- 5.4x ReRAM
- 22x GPU





- PIM-TGAN and ReRAM solutions spend less than ~20% time for memory access and data transfer.
- PIM-TGAN can efficiently utilize up to 55% of its resources.

[9] Adnan Siraj Rakin, Shaahin Angizi, Zhezhi He and Deliang Fan, "PIM-TGAN: A Processing-in-Memory Accelerator for Ternary Generative Adve *International Conference on Computer Design (**ICCD**)* , Oct. 7-10, 2018, Orlando, FL, USA

[10] P. Chi et al., "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in ISCA. IEEE Press, 2016.

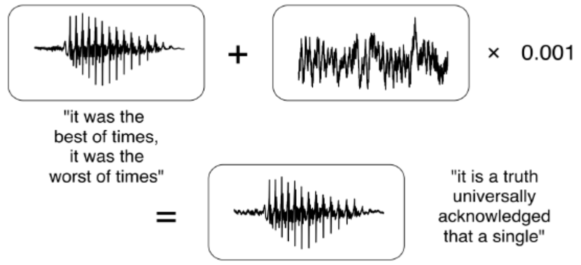
[11] R. Andri et al., "Yodann: An ultra-low power convolutional neural network accelerator based on binary weights," in ISVLSI. IEEE, 2016, pp. 236–241.

Other Dimension of AI: Security

Software: Adversarial Input Attack

(a) Left: Clean Image (Tiger Cat 74.98%)
Right: Adversarial Example (Hen 99.37%)

(b) "Stop sign" has been recognized as "speed limitation".


+
× 0.001
"it was the best of times, it was the worst of times"
=
"it is a truth universally acknowledged that a single"
(c) Adversarial audio misleads speech-to-text recognition.

adversarial example--a type of malicious inputs crafted by adding small and often imperceptible perturbations to legal inputs [1].

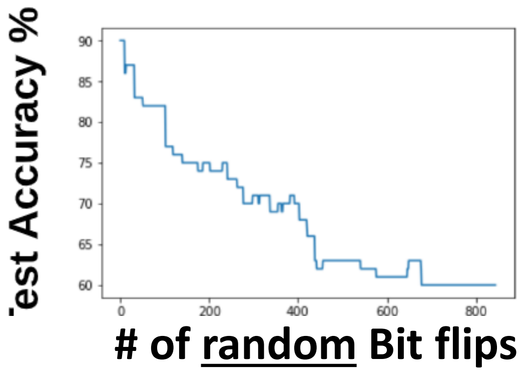
a major concern in many DNN-powered applications.

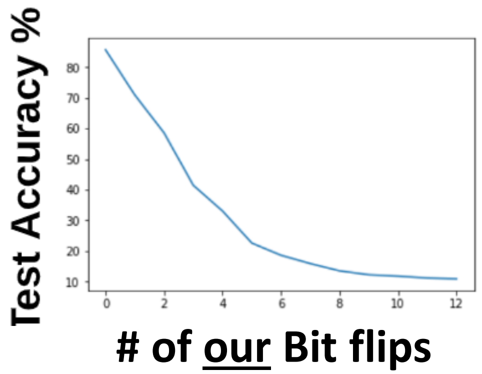
Our method to defend: *parametric noise injection (PNI) includes* trainable Gaussian noise injection at each layer of DNN's activation or weight through solving a min-max optimization problem (**published in CVPR 2019 [2]**)

Code in https://github.com/elliothe/CVPR_2019_PNI

Hardware: Adversarial Weight Attack?

Our first work discovering this is accepted by ICCV 2019 [3]





Method: We propose a Progressive Bit Search (PBS) method which combines gradient ranking and progressive search to identify the most vulnerable bit to be flipped in DNN.

Result: 13 bit-flips out of 93 million bits to completely make ReseNet 18 malfunction in ImageNET (accuracy degrades from 69.8% to 0.1%)
A working prototype based on DRAM row-hammer attack has been developed

Archived in https://arxiv.org/abs/1903.12269

[1] I. Goodfellow, Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.
[2] D. Fan, et. al "Parametric Noise Injection: Trainable Randomness to Improve Deep Neural Network Robustness against Adversarial Attack" CVPR 2019.
[3] D. Fan. et al. "Bit-Flip Attack: Crushing Neural Network with Progressive Bit Search" ICCV 2019

Beyond DNN-in-Memory?

How about other data-intensive computing?

Data-Intensive Processing-in-Memory: Data Encryption

Architecture: Parallel Processing-in-Memory Accelerator
Algorithm: Data intensive and intelligent application algorithm development for developed PIM platform
Developed and developing PIM applications: deep neural network, data encryption, image processing, graph processing

Top-Down ↓ *Algorithm and Architecture Co-Design*

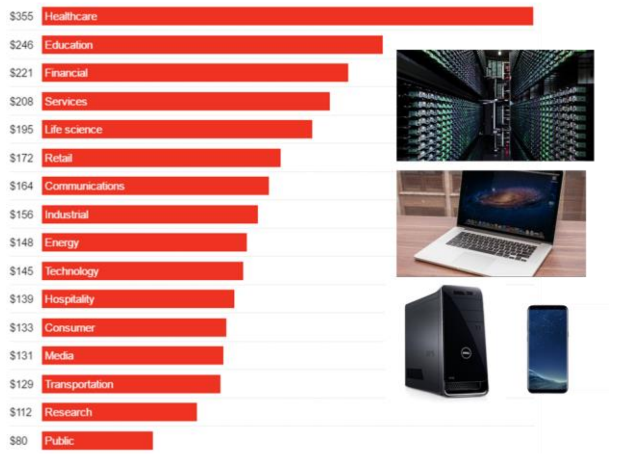
Architecture: In-Memory Computing + **Device & Circuits: NVM +CMOS**

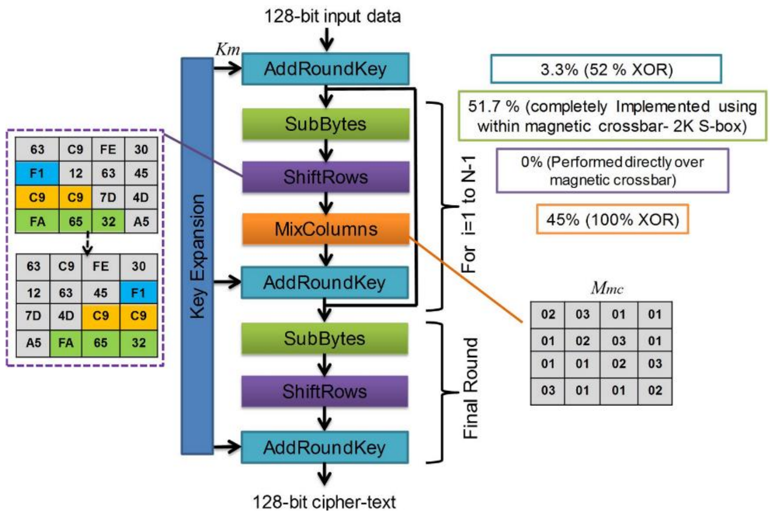
<u>Objective</u>: Energy Efficient and Intelligent Processing-in-Memory

Bottom-Up ↑ *Device and Circuit Co-Design*

Device & Circuit: Parallel and Reconfigurable in-Memory Logic based on STT-MRAM, SOT-MRAM, DWM, ReRAM, DRAM, with extreme low overhead.
Objective: dual mode computational memory simultaneously working as memory and in-memory logic

Data Encryption -in-Memory





63	C9	FE	30
F1	12	63	45
C9	C9	7D	4D
FA	65	32	A5

63	C9	FE	30
12	63	45	F1
7D	4D	C9	C9
A5	FA	65	32

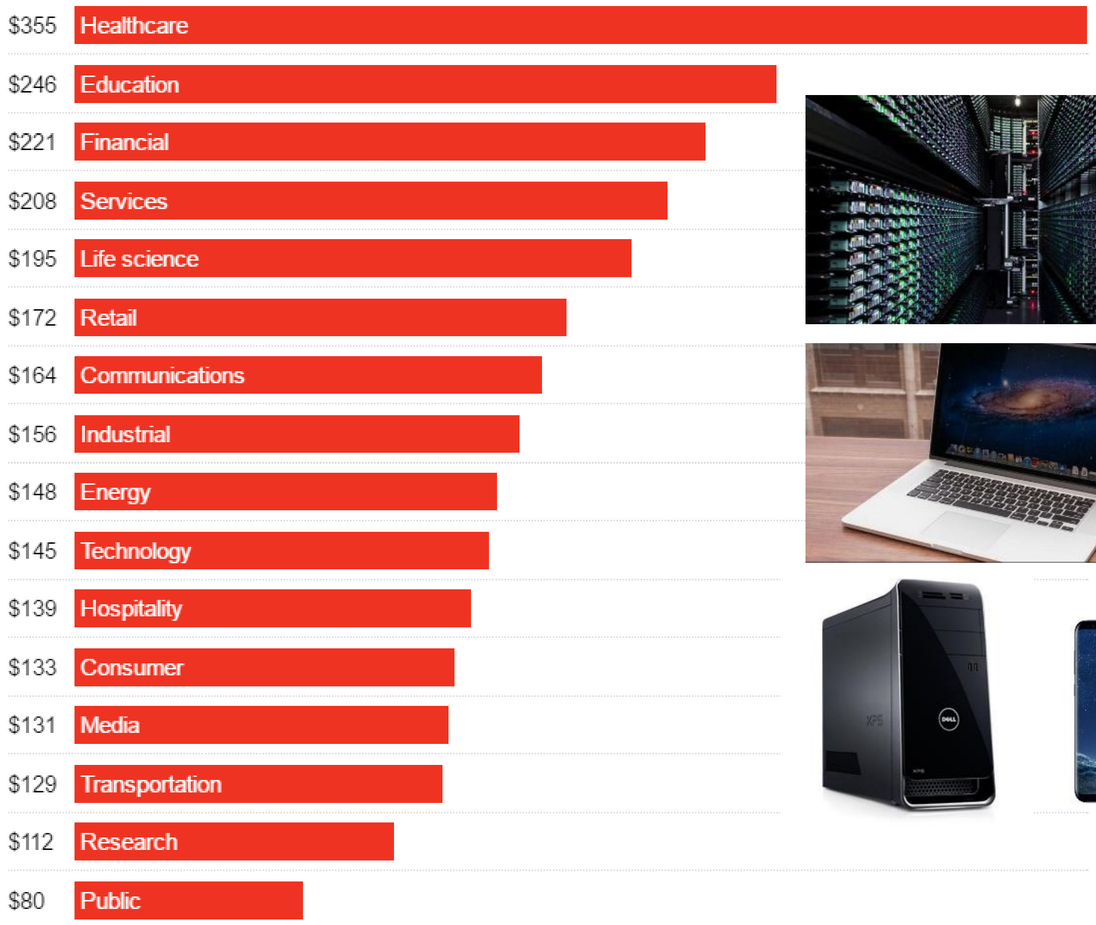
02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

[1] Shaahin Angizi, Zhezhi He and Deliang Fan, "PIMA-Logic: A Novel Processing-in-Memory Architecture for Highly Flexible and Energy-Efficient Logic Computation, " *IEEE/ACM Design Automation Conference* (**DAC**), June 24-28, 2018, San Francisco, CA, USA

[2] Shaahin Angizi, Zhezhi He, Nader Bagherzadeh and Deliang Fan, "Design and Evaluation of a Spintronic In-Memory Processing Platform for Non-Volatile Data Encryption," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (**TCAD**)*, Vol. 37, no. 9, Sept. 2018.

[3] Farhana Parveen, Zhezhi He, Shaahin Angizi and Deliang Fan, "HieIM: Highly Flexible In-Memory Computing using STT MRAM," *Asia and South Pacific Design Automation Conference (**ASP-DAC**)*, Jan. 22-25, 2018, Jeju Island, Korea

Application: Why Energy Efficient Data Encryption ?



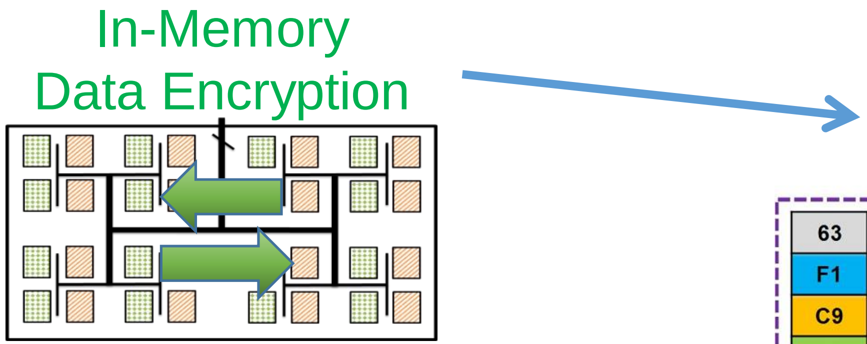
This chart shows the average cost of a stolen record—for example, personally identifiable, payment, or health information on an individual—as broken out by industry.

- Big data: 2.5 quintillion (10^{18}) bytes of data everyday.
- IOT: 17.68 billion IOT device in 2017
- Cost of a breach has risen to $4 million per incident.
- From Data center to personal electronics, data are stored everywhere. The demand of energy efficient and high performance cryptographic components is becoming much stronger nowadays and will keep growing rapidly in the future.

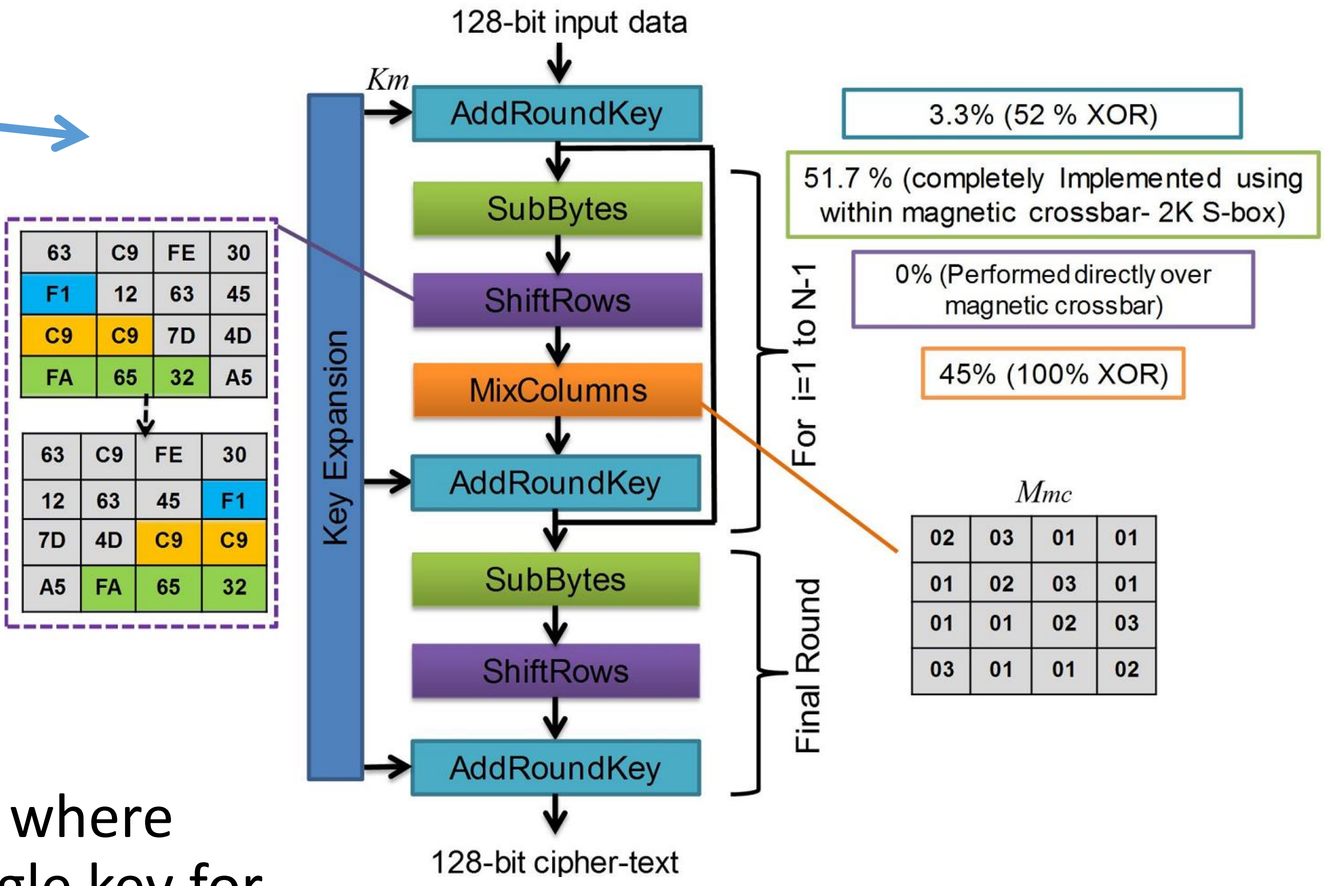
http://fortune.com/2016/06/15/data-breach-cost-study-ibm/

IN-MEMORY DATA ENCRYPTION ENGINE

In-Memory
Data Encryption



- Parallel, local data processing
- Short memory access latency
- Ultra-low energy
- Secure data where they stored
- Reduce data communication risk

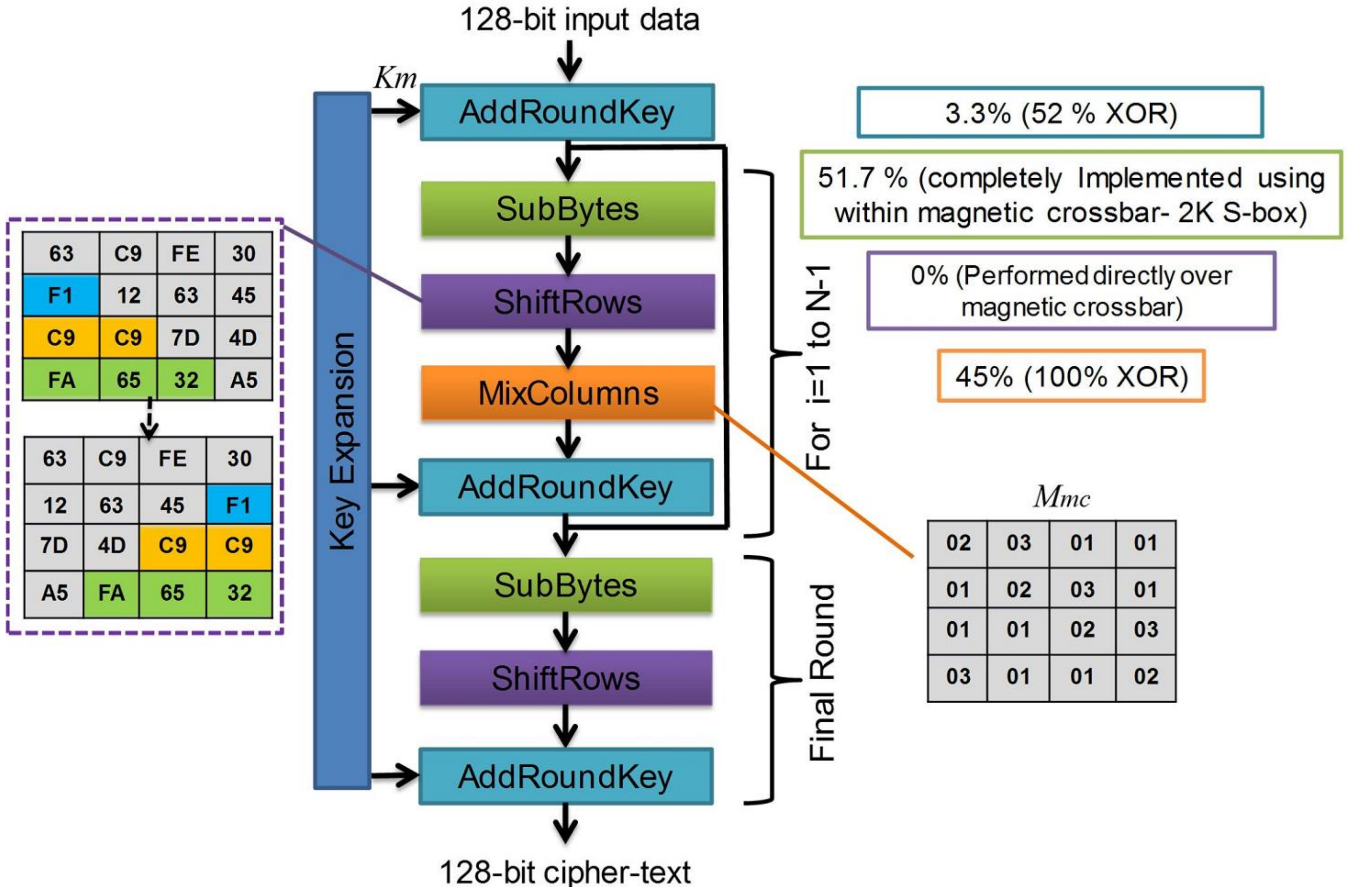


- AES is an iterative symmetric-key cipher where both sender and receiver units use a single key for encryption and decryption.

[5] D. Canniere et al. Katan and ktantan - a family of small and ecient hardware-oriented block ciphers. CHES, 2009.
[21] Y. Wang et al. Dw-aes: A domain-wall nanowire-based aes for high throughput and energy-ecient data encryption in non-volatile memory. IEEE Trans. Inf. Forensics Security, 11, 2016.

Advanced Encryption Standard

- AES basically works on the standard input length of 16 bytes (128 bits) data organized in a 4 x4 matrix (called the state matrix) while using three different key lengths (128, 192, and 256 bits)
- For 128-bit key length, AES encrypts the input data after 10 rounds of consecutive transformations.
- Four transformations:
  - SubBytes : LUT
  - ShiftRows: shift
  - MixColumns: XOR, shift
  - AddRoundKey: XOR



63	C9	FE	30
F1	12	63	45
C9	C9	7D	4D
FA	65	32	A5

63	C9	FE	30
12	63	45	F1
7D	4D	C9	C9
A5	FA	65	32

Mmc

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

Evaluation

Platforms	Energy (nJ)	Cycles	Area (μm^2)
GPP [5]	460	2309	2.5e+6
ASIC [6]	6.6	336	4400
CMOL [7]	10.3	470	320
Baseline DW [4]	2.4	1022	78
Pipelined DW [4]	2.3	2652	83
Multi-issue DW [4]	2.7	1320	155
Ours: ASP-DAC 2018 [1]	3.2	1620	21.8
Ours: IEEE TCAD 2018 [2]	1.74	2168	127
Ours: DAC 2018 [3]	1.5	872	27

- In-Memory Data Encryption based on SOT-MRAM significantly improves the data encryption performance by having the least energy consumption and latency in comparison
- This significant improvement mainly comes from our proposed massive in-memory parallelism computing and intrinsic in-memory logic operations.

[1] Farhana Parveen, et. al. “HieIM: Highly Flexible In-Memory Computing using STT MRAM,” *Asia and South Pacific Design Automation Conference (**ASP-DAC**)*, Jan. 22-25, 2018, Jeju Island, Korea
[2] Shaahin Angizi, et. al. “Design and Evaluation of a Spintronic In-Memory Processing Platform for Non-Volatile Data Encryption,” *IEEE **TCAD***, Vol. 37, no. 9, Sept. 2018.
[3] Shaahin Angizi, et. al. “PIMA-Logic: A Novel Processing-in-Memory Architecture for Highly Flexible and Energy-Efficient Logic Computation, ” *IEEE/ACM Design Automation Conference* (**DAC**), 2018
[4] Y. Wang et al. Dw-aes: a domain-wall nanowire-based aes for high throughput and energy-efficient data encryption in non-volatile memory.IEEE TIFS, 11(11):2426–2440, 2016.
[5] K Malbrain. Byte-oriented-aes: a public domain byte-oriented implementation of aes in c, 2009.
[6] S. Mathew et al. 340 mv–1.1 v, 289 gbps/w, 2090-gate nanoaes hardware accelerator with area-optimized encrypt/decrypt gf (2 4) 2 polynomials in 22 nm tri-gate cmos. IEEE JSSC, 50(4):1048–1058, 2015.
[7] Z Abid et al. Efficient cmol gate designs for cryptography applications. IEEE TNANO, 8:315–321, 2009.

Data-Intensive Processing-in-Memory: Graph Processing

Architecture: Parallel Processing-in-Memory Accelerator
Algorithm: Data intensive and intelligent application algorithm development for developed PIM platform
Developed and developing PIM applications: deep neural network, data encryption, image processing, graph processing

Top-Down ↓ *Algorithm and Architecture Co-Design*

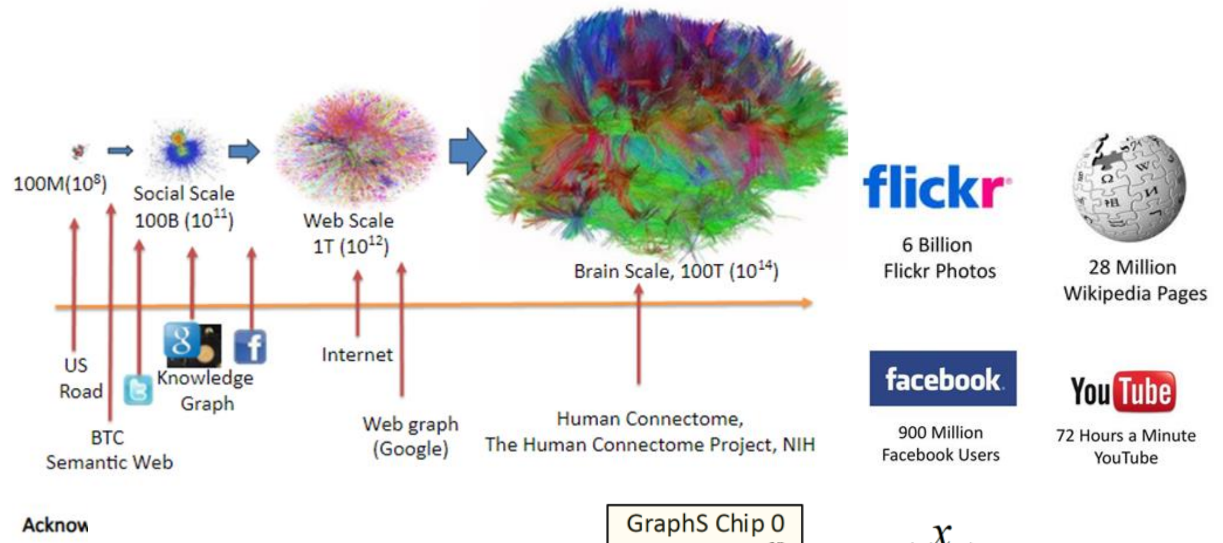
Architecture: In-Memory Computing + **Device & Circuits: NVM +CMOS**

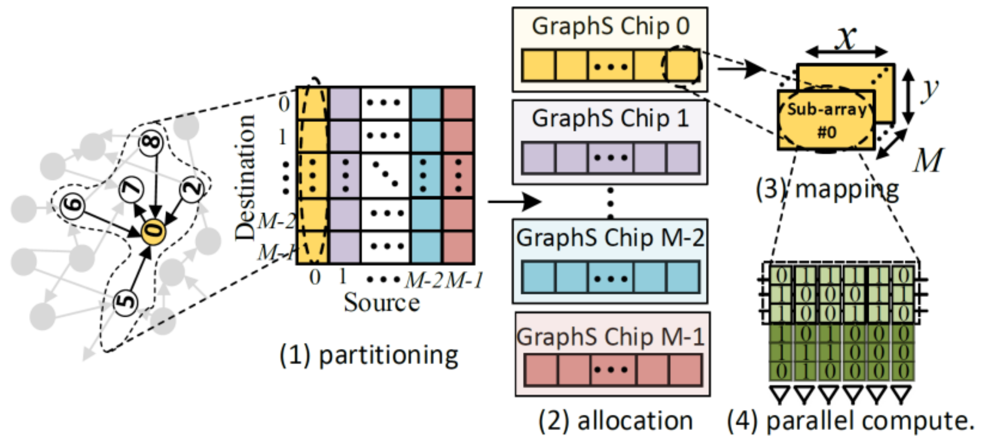
Objective: Energy Efficient and Intelligent Processing-in-Memory

Bottom-Up ↑ *Device and Circuit Co-Design*

Device & Circuit: Parallel and Reconfigurable in-Memory Logic based on STT-MRAM, SOT-MRAM, DWM, ReRAM, DRAM, with extreme low overhead.
Objective: dual mode computational memory simultaneously working as memory and in-memory logic

Graph Processing in-memory


100M(10^8)
Social Scale 100B (10^{11})
Web Scale 1T (10^{12})
Brain Scale, 100T (10^{14})
US Road
BTC Semantic Web
Knowledge Graph
Internet
Web graph (Google)
Human Connectome, The Human Connectome Project, NIH
flickr
6 Billion Flickr Photos
28 Million Wikipedia Pages
facebook
900 Million Facebook Users
YouTube
72 Hours a Minute YouTube
Acknow



D. Fan, et. al., "GraphS: A Graph Processing Accelerator Leveraging SOT-MRAM" published in *Design, Automation and Test in Europe* (DATE), 2019

Data-Intensive Processing-in-Memory: DNA Alignment

Architecture: Parallel Processing-in-Memory Accelerator
Algorithm: Data intensive and intelligent application algorithm development for developed PIM platform
Developed and developing PIM applications: deep neural network, data encryption, image processing, graph processing

Top-Down *Algorithm and Architecture Co-Design*

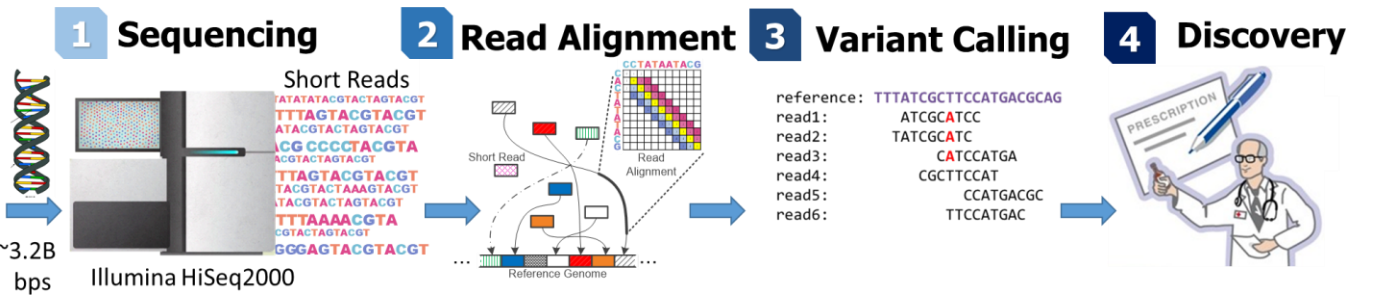
Architecture: In-Memory Computing + **Device & Circuits: NVM +CMOS**

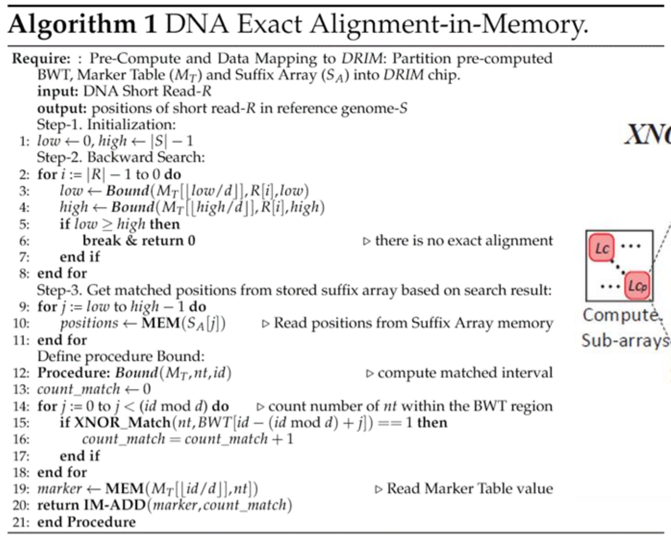
Objective: Energy Efficient and Intelligent Processing-in-Memory

Bottom-Up *Device and Circuit Co-Design*

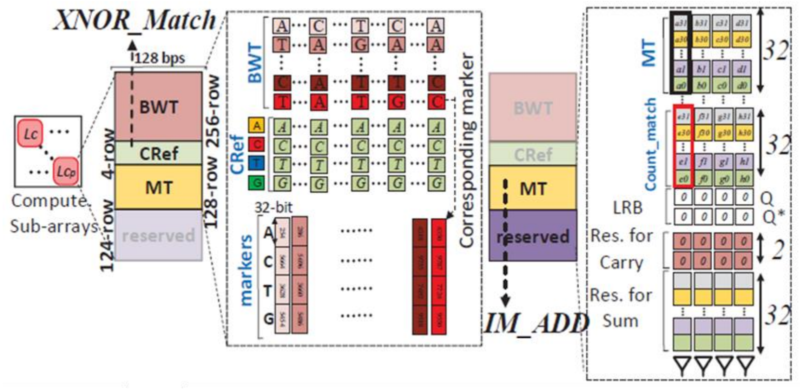
Device & Circuit: Parallel and Reconfigurable in-Memory Logic based on STT-MRAM, SOT-MRAM, DWM, ReRAM, DRAM, with extreme low overhead.
Objective: dual mode computational memory simultaneously working as memory and in-memory logic

DNA Alignment-in-memory




Algorithm 1 DNA Exact Alignment-in-Memory.

```
Require: : Pre-Compute and Data Mapping to DRIM: Partition pre-computed
    BWT, Marker Table ($M_T$) and Suffix Array ($S_A$) into DRIM chip.
    input: DNA Short Read-$R$
    output: positions of short read-$R$ in reference genome-$S$
    Step-1. Initialization:
 1: $low \leftarrow 0, high \leftarrow |S|-1$
    Step-2. Backward Search:
 2: for $i := |R|-1$ to 0 do
 3:     $low \leftarrow \mathbf{Bound}(M_T[\lfloor low/d \rfloor], R[i], low)$
 4:     $high \leftarrow \mathbf{Bound}(M_T[\lfloor high/d \rfloor], R[i], high)$
 5:     if $low \geq high$ then
 6:         break & return 0                          ▷ there is no exact alignment
 7:     end if
 8: end for
    Step-3. Get matched positions from stored suffix array based on search result:
 9: for $j := low$ to $high-1$ do
10:     $positions \leftarrow \mathbf{MEM}(S_A[j])$     ▷ Read positions from Suffix Array memory
11: end for
    Define procedure Bound:
12: Procedure: $Bound(M_T, nt, id)$                   ▷ compute matched interval
13: $count\_match \leftarrow 0$
14: for $j := 0$ to $j < (id \bmod d)$ do       ▷ count number of $nt$ within the BWT region
15:     if XNOR_Match$(nt, BWT[id-(id \bmod d)+j]) == 1$ then
16:         $count\_match = count\_match + 1$
17:     end if
18: end for
19: $marker \leftarrow \mathbf{MEM}(M_T[\lfloor id/d \rfloor], nt])$     ▷ Read Marker Table value
20: return IM-ADD$(marker, count\_match)$
21: end Procedure
```



D. Fan, et. al., "AlignS: A Processing-In-Memory Accelerator for DNA Short Read Alignment Leveraging SOT-MRAM" published in *Design Automation Conference* (DAC), 2019

Summary

- Non-Volatile Memory, like STT-MRAM, SOT-MRAM, ReRAM, could be designed to work as dual-mode memory with both functionalities of memory and logic using innovations in device, circuit and architecture.
- In Device & Circuit layer, we have designed different types of in-memory logic circuit designs that could implement complete Boolean Logic, majority gate, full adder in only one cycle. These logic designs either target for highly parallel computing or to overcome the well known operand locality issue.
- Co-optimization of architecture & algorithm: The dual-mode computational memory could be utilized to accelerate data/compute-intensive applications, such as deep neural network, data encryption, image processing, graph processing, etc.
- The significant improvement mainly comes from our proposed optimized algorithm, massive in-memory parallel computing, data communication reduction and efficient in-memory logic circuits.
- **collaboration is needed, please contact me at dfan@asu.edu**

Thank You & Questions?

Deliang Fan
Assistant Professor, Ph.D.
School of Electrical, Computer and Energy Engineering
Arizona State University, Tempe, AZ, 85287, USA
Email: dfan@asu.edu
https://dfan.engineering.asu.edu/

Thanks to my students:
Zhezhi He, Shaahin Angizi, Adnan Rakin, Li Yang

Our Publications Discussed in this talk

- **[ICCV'19]** Adnan Siraj Rakin, Zhezhi He, Deliang Fan, "Bit-Flip Attack: Crushing Neural Network with Progressive Bit Search," IEEE International Conference on Computer Vision, Korea, Oct 27 - Nov 3, 2019
- **[CVPR'19]** Zhezhi He*, Adnan Siraj Rakin* and Deliang Fan, "Parametric Noise Injection: Trainable Randomness to Improve Deep Neural Network Robustness against Adversarial Attack," *Conference on Computer Vision and Pattern Recognition (CVPR)*, June 16-20, 2019, Long Beach, CA, USA (* The first two authors contributed equally)
- **[CVPR'19]** Zhezhi He and Deliang Fan, "Simultaneously Optimizing Weight and Quantizer of Ternary Neural Network using Truncated Gaussian Approximation," *Conference on Computer Vision and Pattern Recognition (CVPR)*, June 16-20, 2019, Long Beach, CA, USA (accepted)
- [**DAC'19**] Shaahin Angizi, Jiao Sun, Wei Zhang and Deliang Fan, "AlignS: A Processing-In-Memory Accelerator for DNA Short Read Alignment Leveraging SOT-MRAM," *Design Automation Conference* (DAC), June 2-6, 2019, Las Vegas, NV, USA
- [**DAC'19**] Zhezhi He, Jie Lin, Rickard Ewetz, Jiann-Shiun Yuan and Deliang Fan, "Noise Injection Adaption: End-to-End ReRAM Crossbar Non-ideal Effect Adaption for Neural Network Mapping," *Design Automation Conference* (DAC), June 2-6, 2019, Las Vegas, NV, USA
- [**DATE'19**] Shaahin Angizi, Jiao Sun, Wei Zhang and Deliang Fan, "GraphS: A Graph Processing Accelerator Leveraging SOT-MRAM," *Design, Automation and Test in Europe* (DATE), March 25-29, 2019, Florence, Italy
- [**DAC'18**] Shaahin Angizi*, Zhezhi He*, Adnan Siraj Rakin and Deliang Fan, "CMP-PIM: An Energy-Efficient Comparator-based Processing-In-Memory Neural Network Accelerator," IEEE/ACM Design Automation Conference, June 24-28, 2018, San Francisco, CA, USA (* The first two authors contributed equally)
- [**DAC'18**] Shaahin Angizi, Zhezhi He and Deliang Fan, "PIMA-Logic: A Novel Processing-in-Memory Architecture for Highly Flexible and Energy-Efficient Logic Computation," IEEE/ACM Design Automation Conference, June 24-28, 2018, San Francisco, CA, USA
- [**ICCAD'18**] Shaahin Angizi, Zhezhi He and Deliang Fan, "DIMA: A Depthwise CNN In-Memory Accelerator," *IEEE/ACM International Conference on Computer Aided Design* (ICCAD), Nov. 5-8, 2018, San Diego, CA, USA
- [**ASPDAC'19**] Baogang Zhang, Necati Uysal, Deliang Fan and Rickard Ewetz, "Handling Stuck-at-faults in Memristor Crossbar Arrays using Matrix Transformations," Asia and South Pacific Design Automation Conference (ASP-DAC), Jan. 21-24, 2019, Tokyo, Japan (**Best Paper Nomination**)
- [**ISVLSI'18**] Zhezhi He, Shaahin Angizi, Adnan Siraj Rakin and Deliang Fan, "BD-NET: A Multiplication-less DNN with Binarized Depthwise Separable Convolution," *IEEE Computer Society Annual Symposium on VLSI,* July 9-11, 2018, Hong Kong, CHINA (**Best Paper Award**)
- [**ISVLSI'17**] F. Parveen, Z. He, S. Angizi, and D. Fan, "Hybrid Polymorphic Logic Gate with 5-Terminal Magnetic Domain Wall Motion Device," *IEEE Computer Society Annual Symposium on VLSI,* July 3-5, 2017, Bochum, Germany (**Best Paper Award**)
- [**WACV'19**] Zhezhi He, Boqing Gong, Deliang Fan, "Optimize Deep Convolutional Neural Network with Ternarized Weights and High Accuracy," *IEEE Winter Conference on Applications of Computer Vision*, January 7-11, 2019, Hawaii, USA
- [**ASPDAC'19**] Shaahin Angizi, Zhezhi He and Deliang Fan, "ParaNN: A Parallel In-Situ Accelerator for Binary-Weight Deep Neural Networks," Asia and South Pacific Design Automation Conference (ASP-DAC), Jan. 21-23, 2019, Tokyo, Japan
- [**DATE'17**] Z. He, D. Fan, "A Tunable Magnetic Skyrmion Neuron Cluster for Energy Efficient Artificial Neural Network," *Design, Automation and Test in Europe*, Lausanne, Switzerland, 27-31, March, 2017
- [**ICCD'18**] Adnan Siraj Rakin, Shaahin Angizi, Zhezhi He and Deliang Fan, "DIMA: A Depthwise CNN In-Memory Accelerator," IEEE International Conference on Computer Design (ICCD), Oct. 7-10, 2018, Orlando, FL, USA
- [**ISLPED'18**] Li Yang, Zhezhi He and Deliang Fan, "A Fully Onchip Binarized Convolutional Neural Network FPGA Implementation with Accurate Inference," ACM/IEEE International Symposium on Low Power Electronics and Design, July 23-25, 2018, Bellevue, Washington, USA

Our Other Related Publication List

- [JETC'18] Farhana Parveen, Shaahin Angizi and Deliang Fan, "IMFlexCom: Energy Efficient In-memory Flexible Computing using Dual-mode SOT-MRAM," ACM Journal on Emerging Technologies in Computing Systems, vol. 14, no.3, October 2018
- [TNANO'18] Shaahin Angizi, Honglan Jiang, Ronald Demara, Jie Han and Deliang Fan, "Majority-Based Spin-CMOS Primitives for Approximate Computing," IEEE Transactions on Nanotechnology, vol. 17, no. 4, July 2018
- [TMSCS'18] Zhezhi He, Yang Zhang, Shaahin Angizi, Boqing Gong and Deliang Fan, "Exploring A SOT-MRAM based In-Memory Computing for Data Processing," IEEE Transactions on Multi-Scale Computing Systems, 2018
- [TMAG'18] Farhana Parveen, Shaahin Angizi, Zhezhi He and Deliang Fan, "IMCS2: Novel Device-to-Architecture Co-design for Low Power In-memory Computing Platform using Coterminous Spin-Switch," IEEE Transactions on Magnetics, vol. 54, no.7, July 2018
- [TMAG'18] S. Pyle, D. Fan, R. DeMara, "Compact Spintronic Muller C-Element with Near-Zero Standby Energy," IEEE Transactions on Magnetics, vol.54, no.2, Feb. 2018 (Front Cover Paper)
- [TMSCS'17] Y. Bai, D. Fan and M. Lin, "Stochastic-Based Synapse and Soft-Limiting Neuron with Spintronic Devices for Low Power and Robust Artificial Neural Networks," IEEE Transactions on Transactions on Multi-Scale Computing Systems, vol.4, no.3, pp.463-476, Dec. 2017
- [TCAD'17] S. Angizi, Z. He, N. Bagherzadeh and D. Fan, "Design and Evaluation of a Spintronic In-Memory Processing Platform for Non-Volatile Data Encryption," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.37, no.9, Sept. 2018
- [ISVLSI'18] Zhezhi He, Shaahin Angizi and Deliang Fan, "Accelerating Low Bit-Width Deep Convolution Neural Network in MRAM," *IEEE Computer Society Annual Symposium on VLSI,* July 9-11, 2018, Hong Kong, CHINA (invited)
- [GLSVLSI'18] Shaahin Angizi, Zhezhi He, Yu Bai, Jie Han, Mingjie Lin and Deliang Fan, "Leveraging Spintronic Devices for Efficient Approximate Logic and Stochastic Neural Network," *ACM Great Lakes Symposium on VLSI*, Chicago, IL, USA, May 23-25, 2018 (invited)
- [WACV'18] Y. Ding, L. Wang, D. Fan and B. Gong "A Semi-Supervised Two-Stage Approach to Learning from Noisy Labels," *IEEE Winter Conference on Applications of Computer Vision*, March 12-14, 2018, Stateline, NV, USA
- [ASPDAC'18] F. Parveen, Z. He, S. Angizi and D. Fan, "HieIM: Highly Flexible In-Memory Computing using STT MRAM," Asia and South Pacific Design Automation Conference, Jan. 22-25, 2018, Jeju Island, Korea
- [ASPDAC'18] S. Angizi, Z. He, F. Parveen and D. Fan, "IMCE: Energy-Efficient Bit-Wise In-Memory Convolution Engine for Deep Neural Network," Asia and South Pacific Design Automation Conference, Jan. 22-25, 2018, Jeju Island, Korea
- [ICCD'17] Z. He, S. Angizi and D. Fan, "Exploring STT-MRAM based In-Memory Computing Paradigm with Application of Image Edge Extraction," IEEE International Conference on Computer Design, Nov. 5-8, 2017, Boston, MA
- [ICCD'17] D. Fan and S. Angizi "Energy Efficient In-Memory Binary Deep Neural Network Accelerator with Dual-Mode SOT-MRAM," *IEEE International Conference on Computer Design*, Nov. 5-8, 2017, Boston, MA
- [ICCAD'17] M. Yang, J. Hayes, D. Fan, W. Qian, "Design of Accurate Stochastic Number Generators with Noisy Emerging Devices for Stochastic Computing," IEEE/ACM International Conference on Computer Aided Design, Nov 13-16, 2017, Irvin, CA
- [ISVLSI'17] D. Fan, S. Angizi, and Z. He, "In-Memory Computing with Spintronic Devices," *IEEE Computer Society Annual Symposium on VLSI,* July 3-5, 2017, Bochum, Germany (invited)
- [ISVLSI'17] S. Angizi, Z. He and D. Fan, "RIMPA: A New Reconfigurable Dual-Mode In-Memory Processing Architecture with Spin Hall Effect-Driven Domain Wall Motion Device," *IEEE Computer Society Annual Symposium on VLSI,* July 3-5, 2017, Bochum, Germany
- [ISLPED'17] F. Parveen, S. Angizi, Z. He and D. Fan "Low Power In-Memory Computing based on Dual-Mode SOT-MRAM," *IEEE/ACM International Symposium on Low Power Electronics and Design,* July 24-26, 2017, Taipei, Taiwan
- [MWSCAS'17] D. Fan, Z. He and S. Angizi, "Leveraging Spintronic Devices for Ultra-Low Power In-Memory Computing: Logic and Neural Network," *60th IEEE International Midwest Symposium on Circuits and Systems, Aug.* 6-9, 2017, Boston, MA, USA (invited)
- [ISCAS'17] F. Parveen, S. Angizi, Z. He and D. Fan, "Hybrid Polymorphic Logic Gate Using 6 Terminal Magnetic Domain Wall Motion Device," *IEEE International Symposium on Circuits & Systems*, Baltimore, MD, USA, May 28-31, 2017
- [GLSVLSI'17] Z. He, S. Angizi, F. Parveen, and D. Fan, "Leveraging Dual-Mode Magnetic Crossbar for Ultra-low Energy In-Memory Data Encryption", *27th ACM Great Lakes Symposium on VLSI*, Banff, Alberta, Canada, May 10-12, 2017
- [GLSVLSI'17] S. Angizi, Z. He, and D. Fan, "Energy Efficient In-Memory Computing Platform Based on 4-Terminal Spin Hall Effect-Driven Domain Wall Motion Devices", *27th ACM Great Lakes Symposium on VLSI*, Banff, Alberta, Canada, May 10-12, 2017
- [GLSVLSI'17] Q. Alasad, J. Yuan, and D. Fan, "Leveraging All-Spin Logic to Improve Hardware Security", *27th ACM Great Lakes Symposium on VLSI*,, Banff, Alberta, Canada, May 10-12, 2017