Efficient In-Memory Computing Circuits and System for AI with Hardware and Algorithm Co-Design

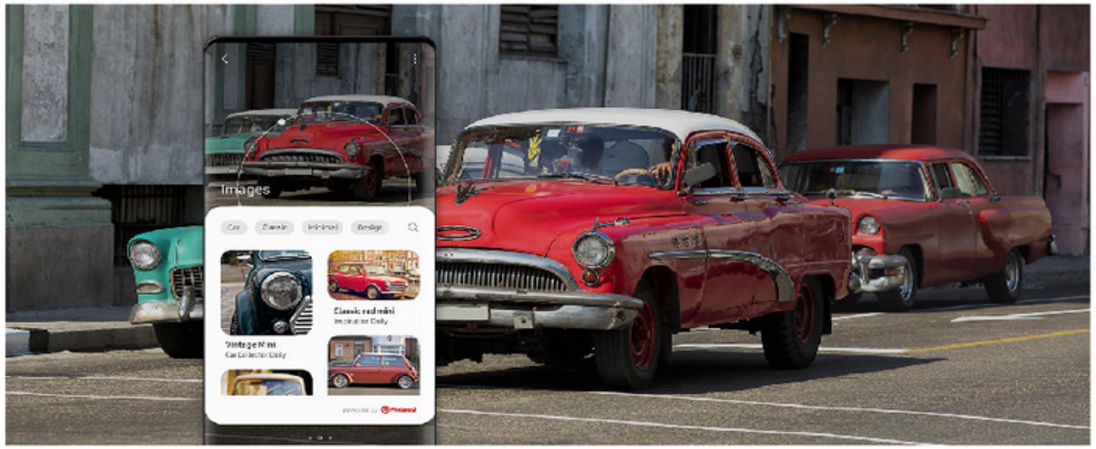
Deliang Fan

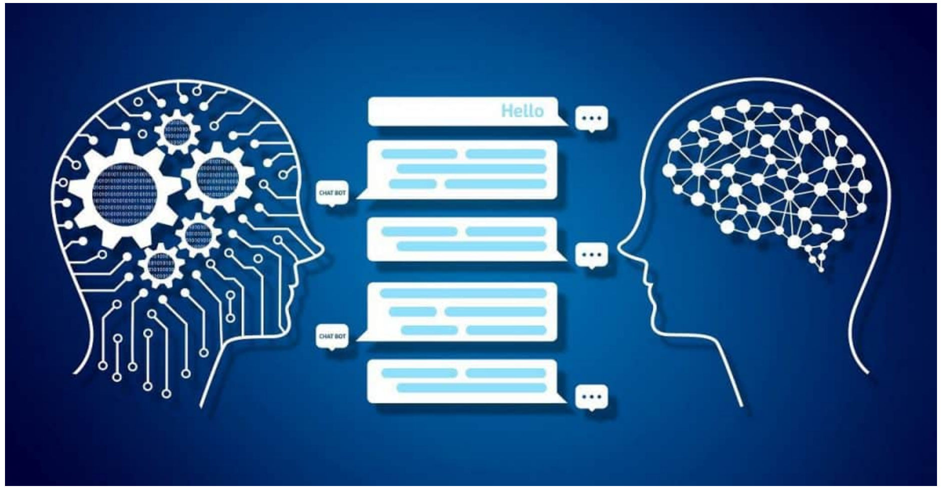
Director of *Efficient, Secure and Intelligent Computing* (ESIC) Laboratory
School of Electrical, Computer and Energy Engineering
Arizona State University, Tempe, AZ, USA
Email: dfan@asu.edu
https://faculty.engineering.asu.edu/dfan/

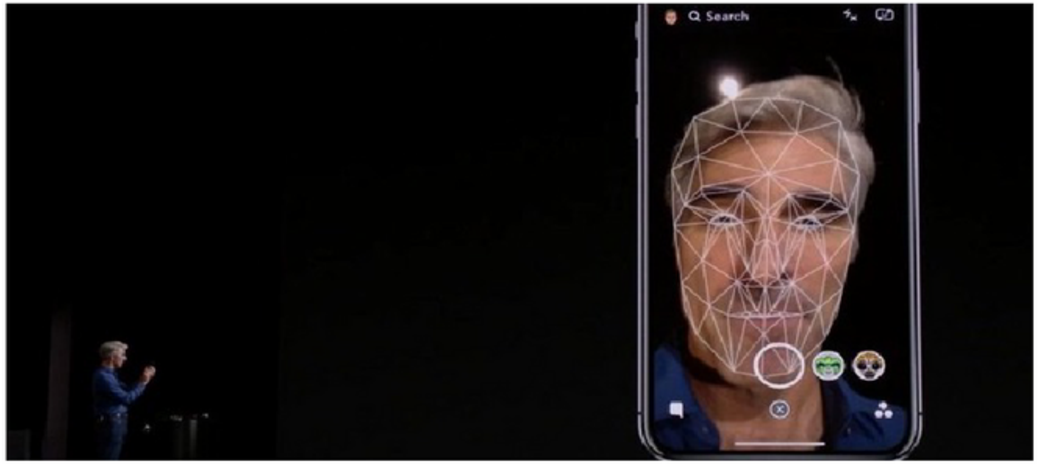
Contributing Ph.D. Students

Zhezhi He (Asso. prof. / Shanghai Jiaotong U.); Shaahin Angizi (TT-AP/ NJIT); Adnan Rakin (TT-AP/ Binghamton U.); Li Yang (TT-AP/ UNC Charlotte) , Fan Zhang (Google), Amitesh Sridharan, Yongjae Lee, Zhaoliang Zhang, Juyang Bai, Asmer, Jingxing, etc.

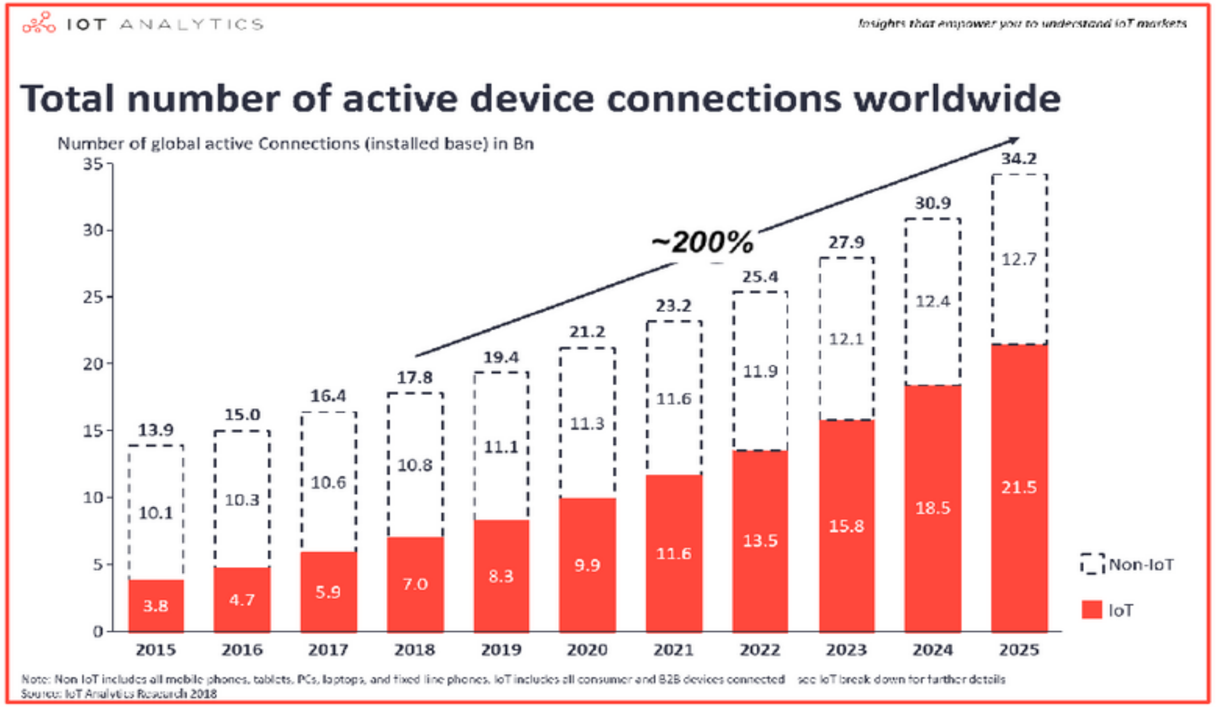
Success of Deep Learning



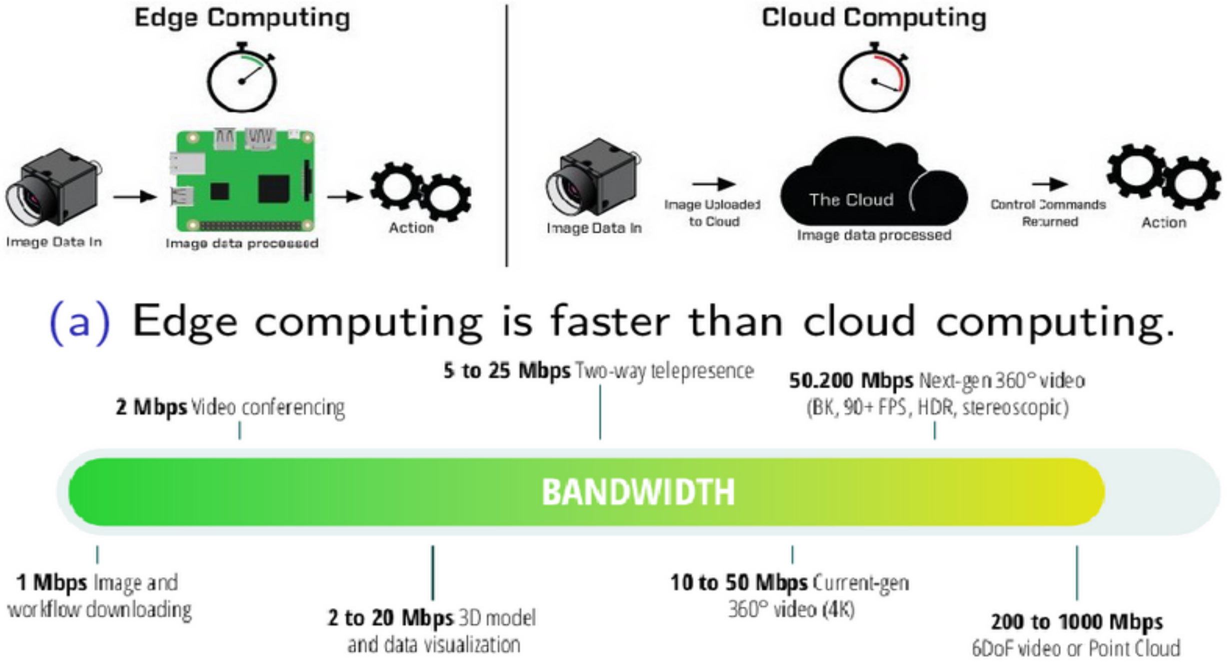




Motivation of Edge Computing



Note: Non-IoT includes all mobile phones, tablets, PCs, laptops, and fixed line phones. IoT includes all consumer and B2B devices connected – see IoT break down for further details
Source: IoT Analytics Research 2018



Edge Computing
Image Data In
Image data processed
Action
Cloud Computing
Image Data In
Image Uploaded to Cloud
The Cloud
Image data processed
Control Commands Returned
Action
2 Mbps Video conferencing
5 to 25 Mbps Two-way telepresence
50.200 Mbps Next-gen 360° video (8K, 90+ FPS, HDR, stereoscopic)
BANDWIDTH
1 Mbps Image and workflow downloading
2 to 20 Mbps 3D model and data visualization
10 to 50 Mbps Current-gen 360° video (4K)
200 to 1000 Mbps 6DoF video or Point Cloud

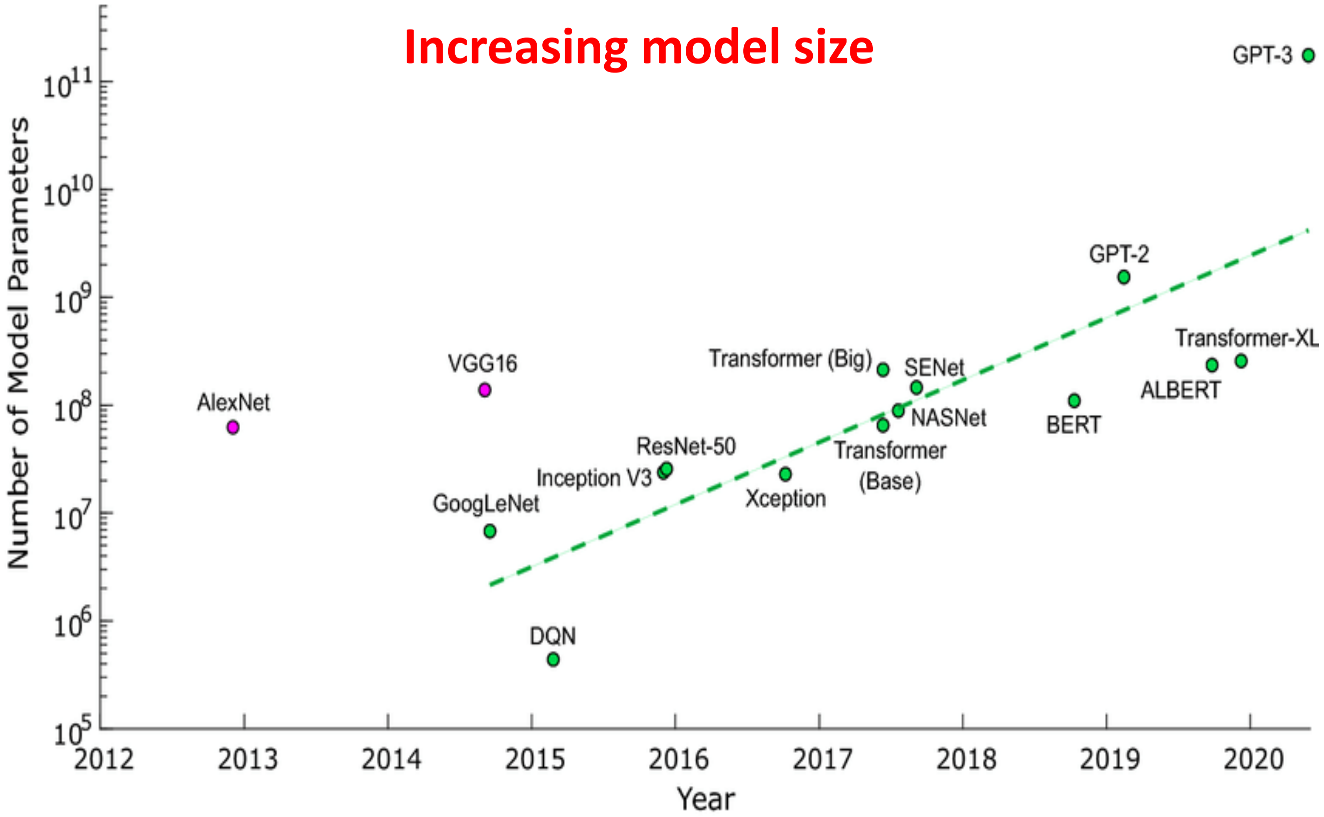
(a) Edge computing is faster than cloud computing.

(b) Crowed Bandwidth by content-size and #devices.

Figure: Inference on edge for latency and bandwidth

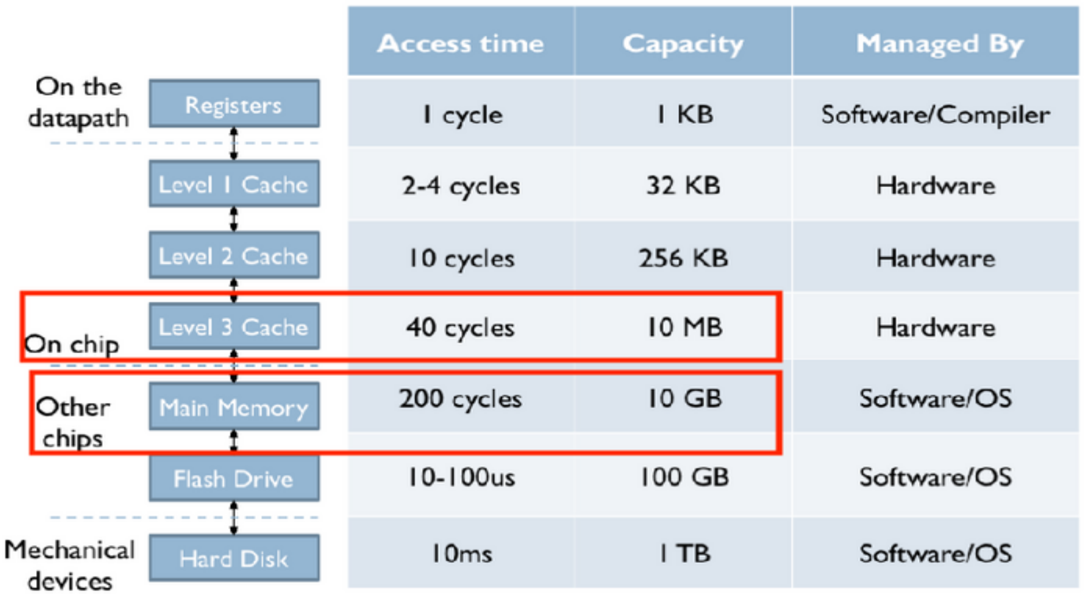
- #edge devices (IoT/Non-IoT) will be **approximately doubled in 5 years**.
- Performing DNN inference on edge is becoming more preferable:
  - Reduce inference latency
  - Enhance user privacy
  - Avoid bandwidth competition.
  - $ Cut Long-term cloud computing bill

Concerns of DNN Deployment on Edge devices



Increasing model size

Memory-Wall



On the datapath
On chip
Other chips
Mechanical devices

	Access time	Capacity	Managed By
Registers	1 cycle	1 KB	Software/Compiler
Level 1 Cache	2-4 cycles	32 KB	Hardware
Level 2 Cache	10 cycles	256 KB	Hardware
Level 3 Cache	40 cycles	10 MB	Hardware
Main Memory	200 cycles	10 GB	Software/OS
Flash Drive	10-100us	100 GB	Software/OS
Hard Disk	10ms	1 TB	Software/OS

Source: Purdue U & U of Warsaw and Computation structure

Figure: DNN trend of accuracy vs. model size and memory hierarchy.

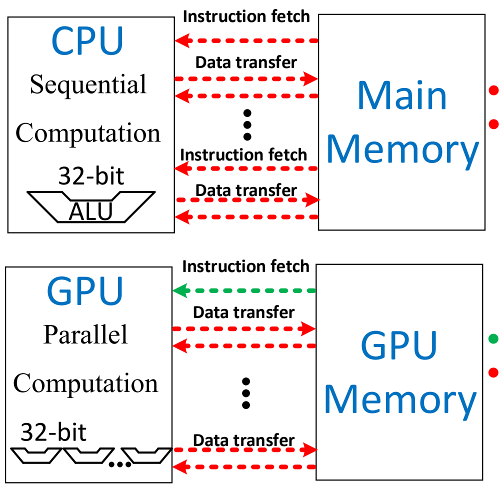
Bernstein, Liane, et. al. *Scientific reports* 11, no. 1 (2021): 3144.

- DNNs with **higher accuracy** requires **larger model size** & **higher computing workload**.
- **Large model** normally cannot fit into on-chip cache.
- Cache the model in **Off-chip DRAM** with **expensive long-distance memory access**.

Energy Efficient In-Memory Computing

Memory Wall

Von-Neumann Architecture



- Multiple instruction fetch
- Multiple data transfer

- Single instruction fetch
- Multiple data transfer

Multiplication: 3.1pJ
Addition: 0.1pJ

∧

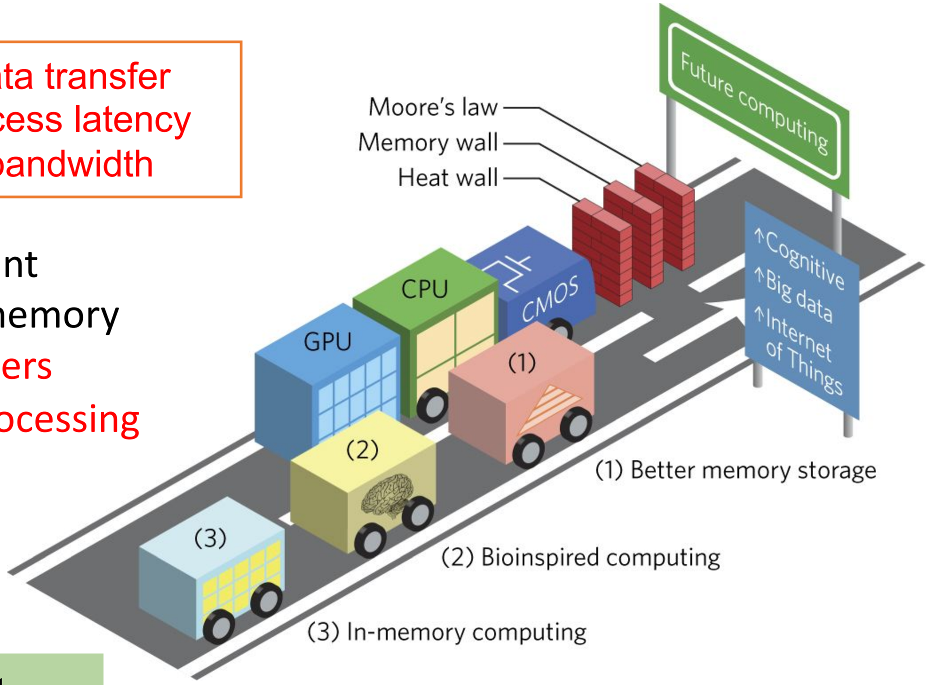
On-chip cache:
Energy: ~5pJ
Latency: ~10ns

∧

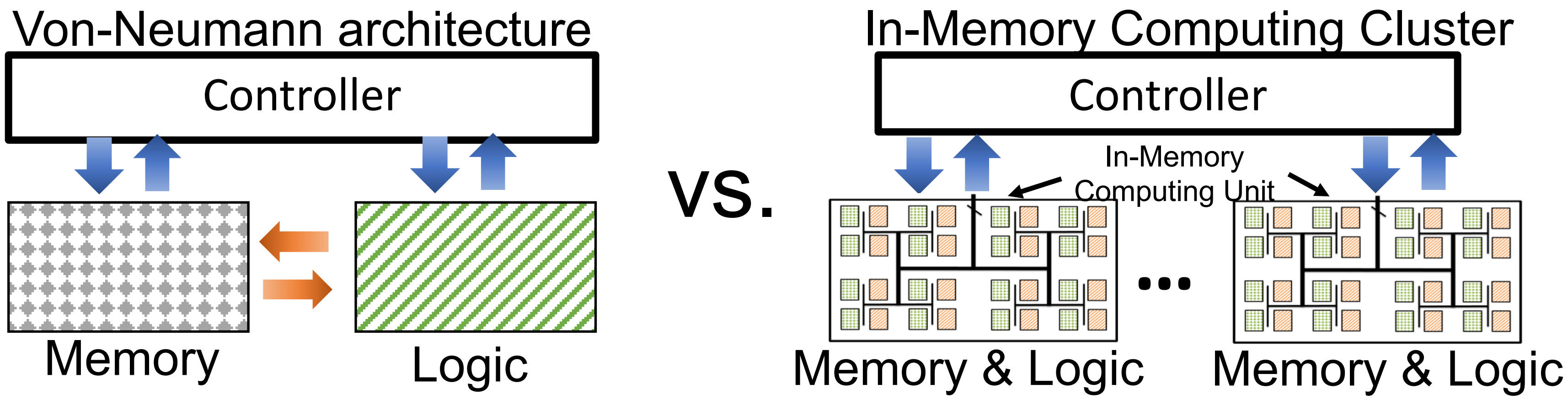
Off-chip memory:
Energy: ~640pJ
Latency: ~100ns

- ➢ 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



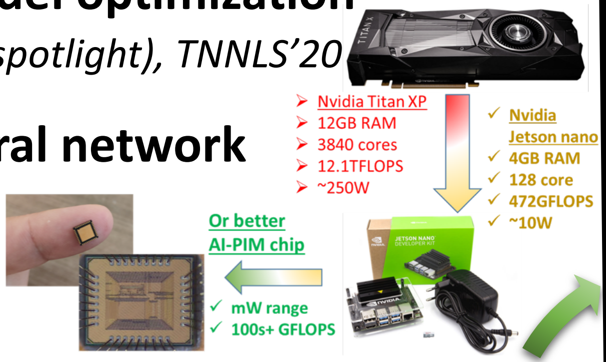
- ✓ Parallel, local data processing
- ✓ Short memory access latency
- ✓ Ultra-low energy
- ✓ Programmable, Low cost

Dr. Fan: Efficient, Secure, and Intelligent Computing (ESIC) Laboratory

AI Performance & Efficiency

Algorithm

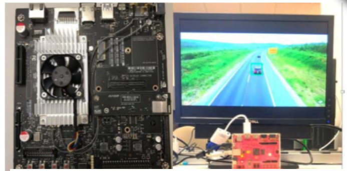
- **Compute- and Memory-efficient on-device learning** *NeurIPS’22/23, CVPR’22/21, AAAI’22(spotlight), ICLR’22(spotlight), etc.*
- **Hardware-aware AI model optimization** *CVPR’19, WACV’19, AAAI’20 (spotlight), TNNLS’20*
- **Run-time dynamic neural network** *TNNLS’22, NeurIPS’22, DAC’20*

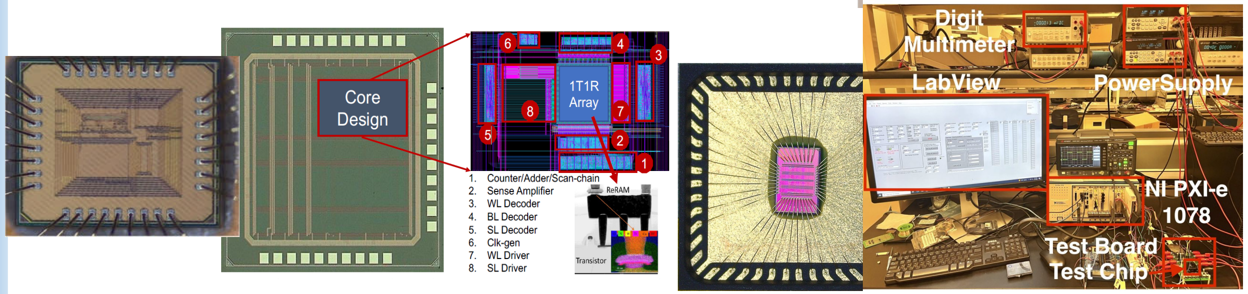


Co-Design

- **In-memory computing chips**
- **Emerging non-volatile memory**
- **Neuromorphic computing**

Hardware



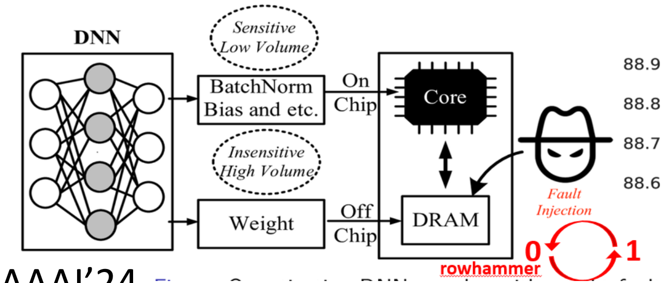


ESSCIRC’22/23, CICC’24, DAC’16-24, ICCAD’18-21, DATE’17-23, DRC’19, *JSSC, SSCL,* OJSSCS, TCAS-I/II, TMAG, TC, TNANO, TCAD, EDL, etc.

AI Security & Privacy

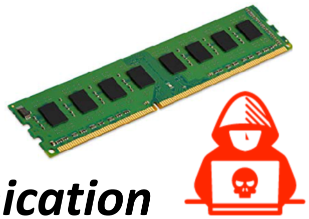
- **Adversarial noise robustness** *CVPR’19, CVOPS’19*

- **Adversarial Weight Attack & Defense** *ICCV’19, CVPR’20, DAC’20, DATE’21, etc.*
- **AI Trojan Attack** *CVPR’20, TPAMI’21*
- **Model inversion**

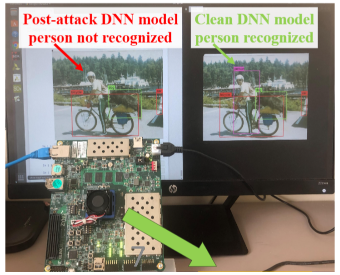


HOST’21, CVPR’22, SP’22, AAAI’24

- **Memory Bit-Flip Attack in Computer *main memory*** *USENIX Security’20*



- **Fault injection into the data *communication* in cloud-FPGA in black-box setup** *USENIX Security’21, Security & Privacy (SP)’24*



Post-attack DNN model person not recognized
Clean DNN model person recognized

- **AI Model/Data Stealing from memory side-channel** *IEEE-Security & Privacy (SP) – 2022*

Part-I: Efficient AI Computing-in-Memory

AI Performance & Efficiency

Algorithm

- **Compute- and Memory-efficient on-device learning (continual learning, self-supervised, etc.)**
  NeurIPS’22/23, CVPR’22/21, AAAI’22, ICLR’22, etc.
- **Hardware-aware AI model optimization**
  CVPR’19, WACV’19, AAAI’20 (spotlight), TNNLS’20
- **Run-time dynamic neural network**
  TNNLS’22, NeurIPS’22, DAC’20

- Nvidia Titan XP
- 12GB RAM
- 3840 cores
- 12.1TFLOPS
- ~250W

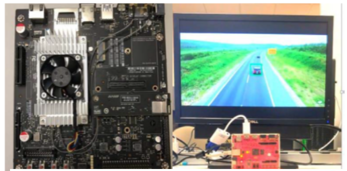
- ✓ Nvidia Jetson nano
- ✓ 4GB RAM
- ✓ 128 core
- ✓ 472GFLOPS
- ✓ ~10W

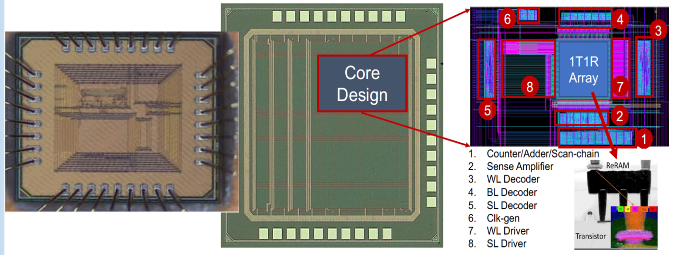
- ✓ mW range
- ✓ 100s+ GFLOPS

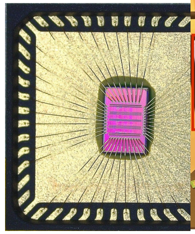
Co-Design

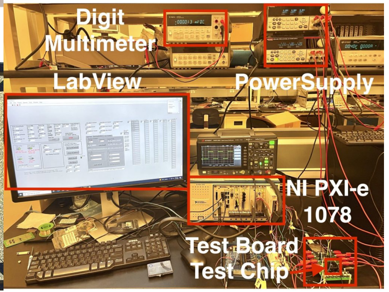
Hardware

- **In-memory computing chips**
- **Emerging non-volatile memory**
- **Neuromorphic computing**



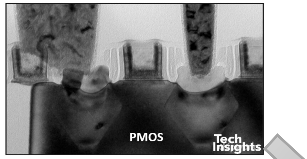




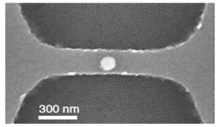


ESSCIRC’22/23, CICC’24, DAC’16-24, ICCAD’18-21, DATE’17-23, DRC’19, *JSSC, SSCL,* OJSSCS, TCAS-I/II, TMAG, TC, TNANO, TCAD, EDL, etc.

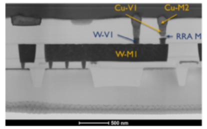
CMOS



MRAM



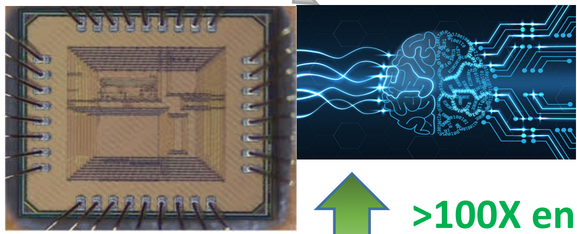
ReRAM



1 bit

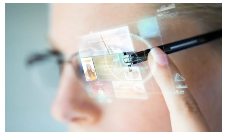
multi-bit

AI-in-memory



>100X energy efficiency

Wearable AI



Smart IoT

Federated IoT learning

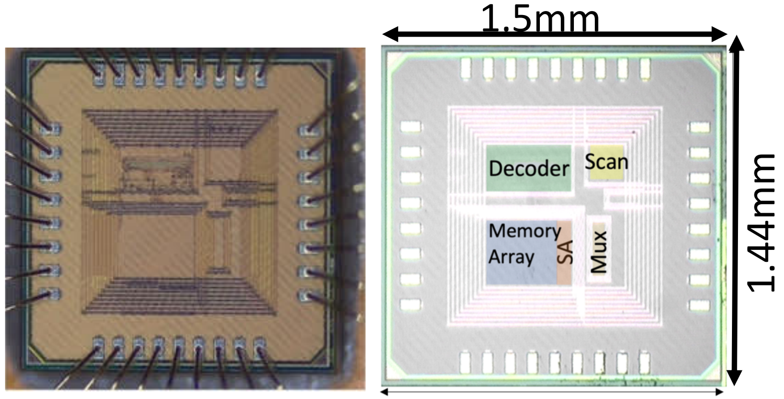
Smart health

Objective: low power, high performance computing, reliable, smaller AI model, learning-on-device, secure, trustworthy, and more…

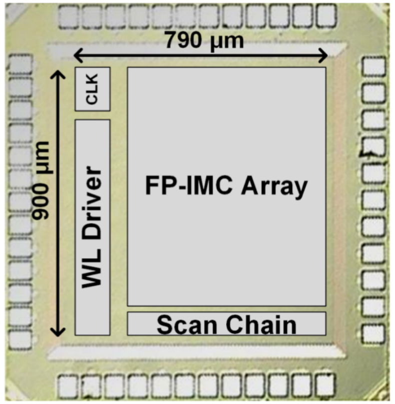
Research projects funded by NSF FuSe, ACED, SHF, CPS, FET, Satc, Career, DARPA, IARPA, SRC, etc.

Our Developed IMC Chip Prototypes (examples): SRAM and NVM

- A 1.23-GHz 16-kb **Programmable and Generic Processing-in-SRAM** Accelerator in **TSMC 65nm**
- Published in ESSCIRC’2022

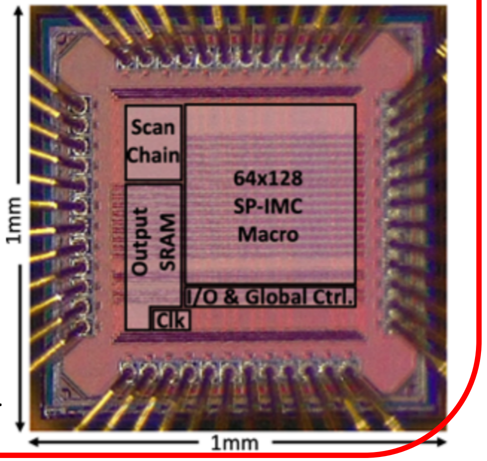


- All-Digital **Configurable Floating-Point** In-Memory Computing Macro in **TSMC 28nm**
- Published in ESSCIRC’2023



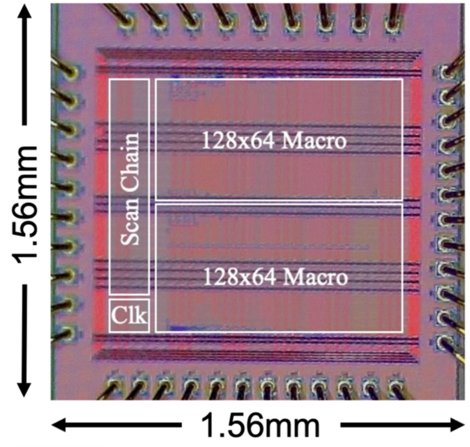
- **28 nm** IMC chip prototype for **sparse in-memory matrix convolution**
- Supports N:M structured sparsity, run length encoding, compressed sparsity column
  - **Sparse NN**
  - **Wireless Decoder (Neural BP)**

DAC’23
CICC’24

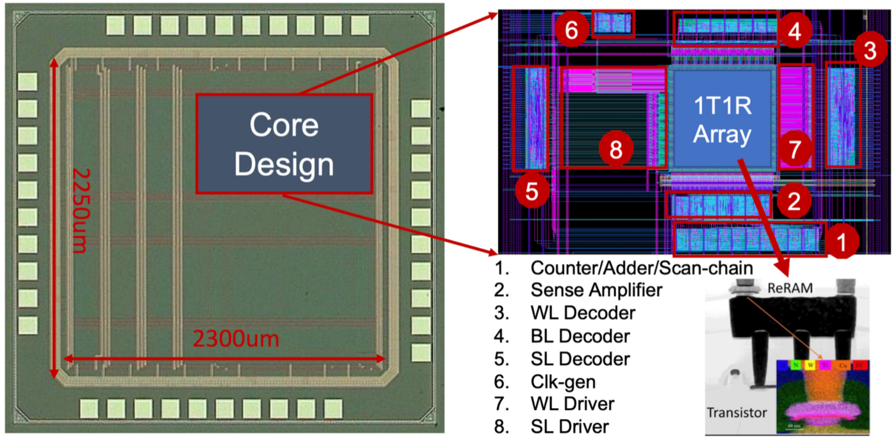


- A **28nm** 2385.7 TOPS/W/b **Precision Scalable** In-Memory Computing Macro with **Bit-Parallel** Inputs and Decomposable Weights for DNNs

SSCL

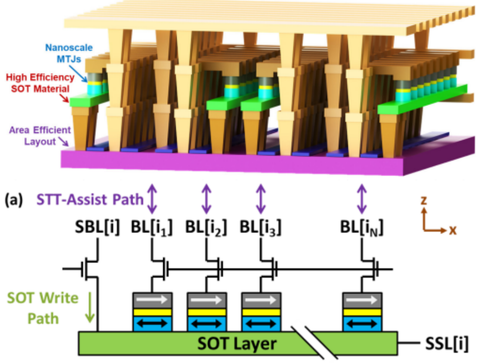


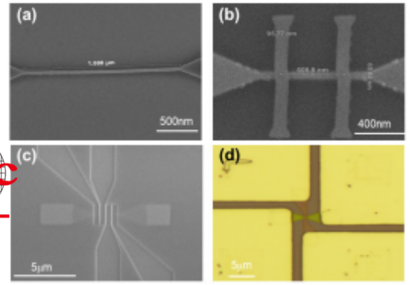
- **hybrid 65nm CMOS-ReRAM (HfO2)** designs for **DNN** acceleration and **DNA genome sequence alignment** applications.
- 64 by 64 HfO2 RRAM and SRAM module for hybrid computing



ESSCIRC’23, invited to extend to **JSSC** special issue

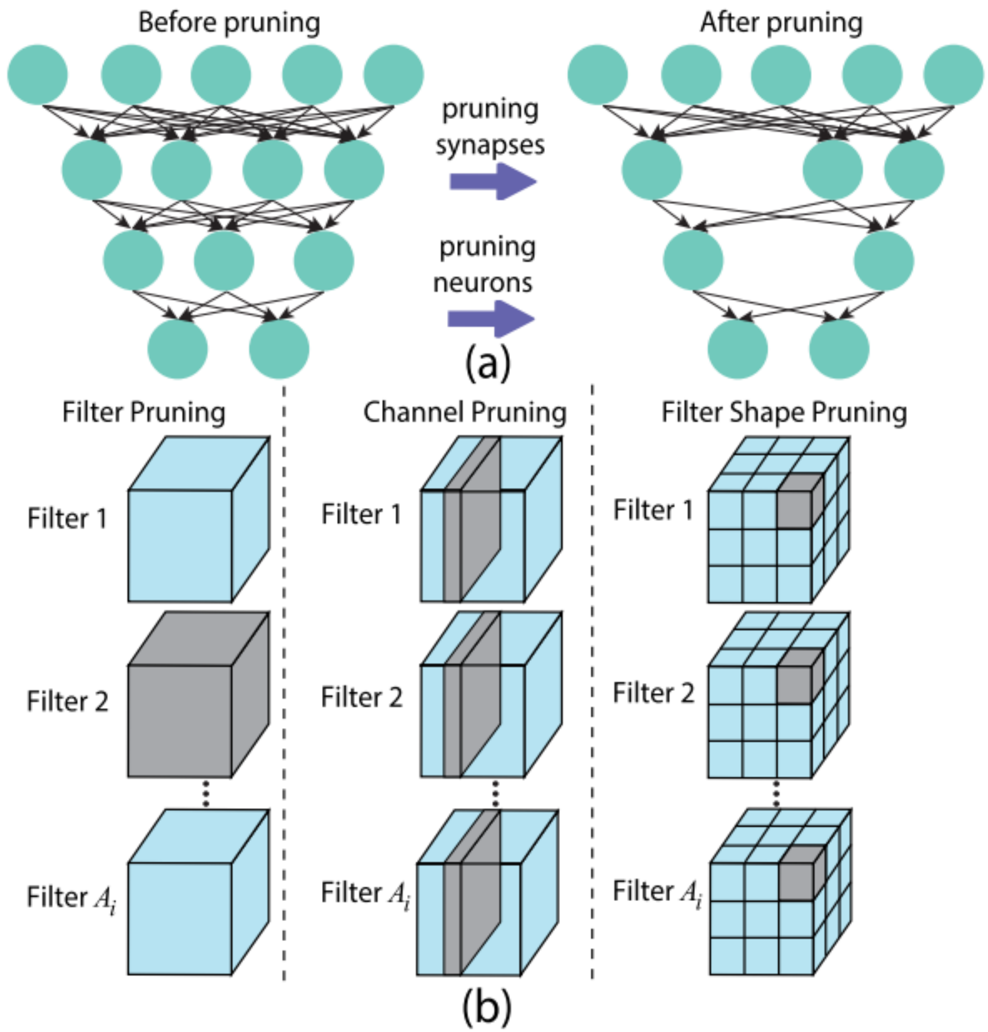
- energy efficient IMC with high-density, field-free **STT-assisted SOT-MRAM** (SAS-MRAM), ~**100nm, NSF FuSe/ACED projects**



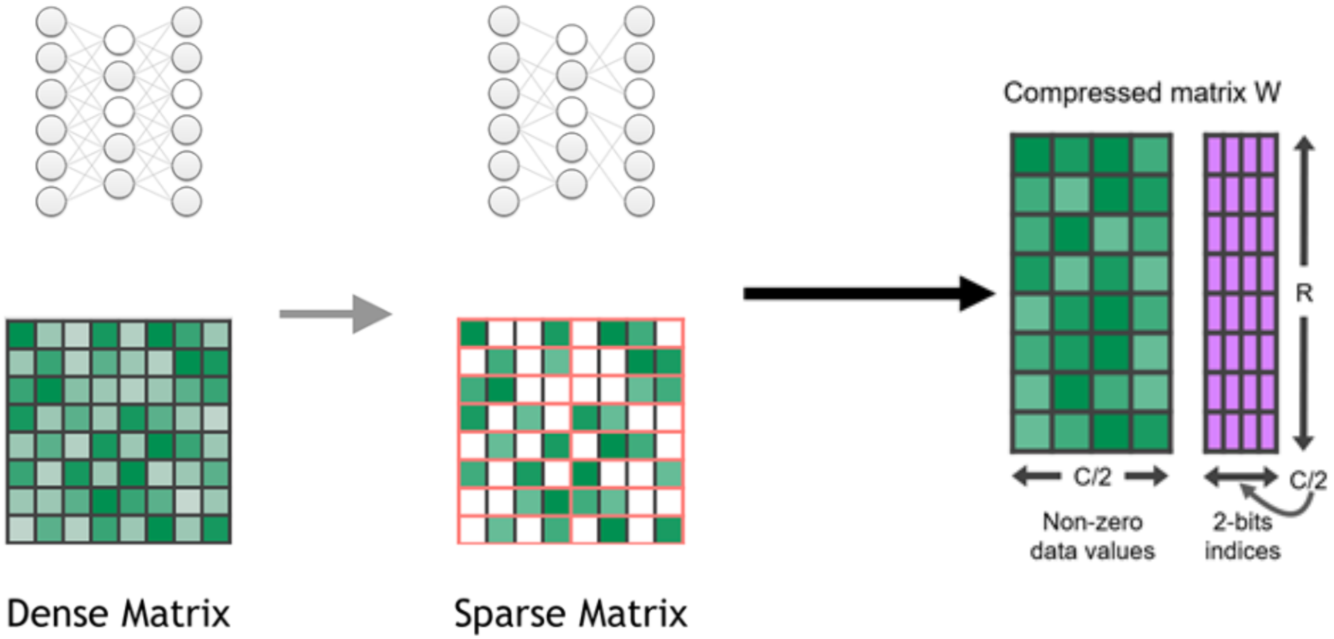


W. Hwang, D. Fan, S. X. Wang, et., al., TMAG, 2022

Sparse Neural Network through Pruning



Unstructured and **Structured** NN Pruning
Significantly reduce model size with no accuracy drop



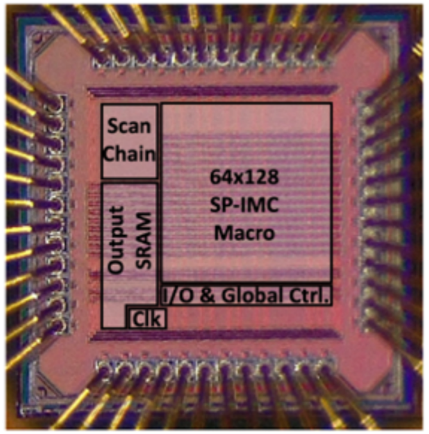
- nVidia GPU beyond Ampere architecture supports N:M structured **sparse matrix processing.**
- 2:4 sparsity in A100 GPU, 2X peak performance, 1.5 X measured BERT speedup in real system, no accuracy degradation

- **Sparse-encoded matrix processing (without decoding) is needed for IMC**

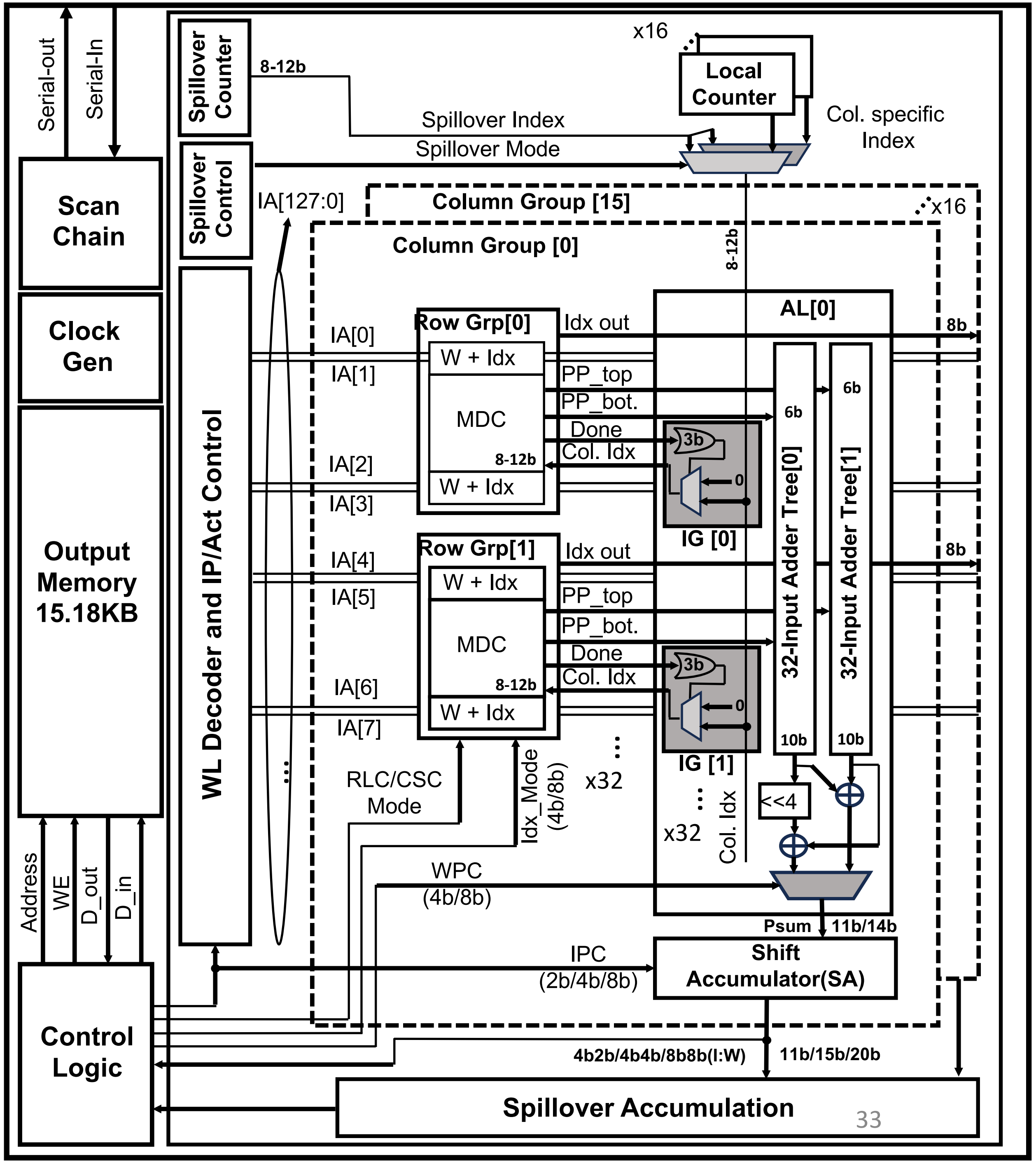
https://developer.nvidia.com/blog/accelerating-inference-with-sparsity-using-ampere-and-tensorrt/

Sparse(SP)-IMC Macro Design

- 64x64 bits for weights, 64x64 bits for indices.
- 16 compute **column groups**.
- Each column group has 32 **row groups** and one accumulation logic (**AL**) module.

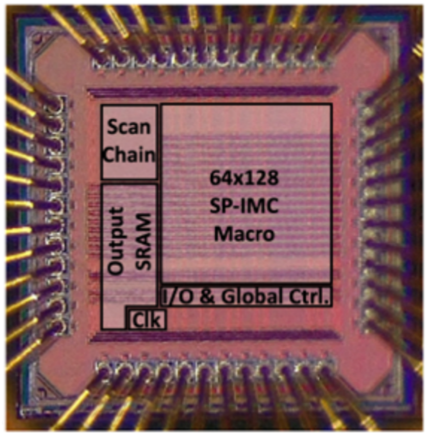


- Each Row Group could generate two partial products (PP_top, PP_bot.), sending to accumulation logic module (i.e., adder trees) to accumulate in a ‘semi-bit-serial’ pattern, for MAC operation
- Each column group generated partial-sum will be sent to shared shift-accumulator for next-stage accumulation



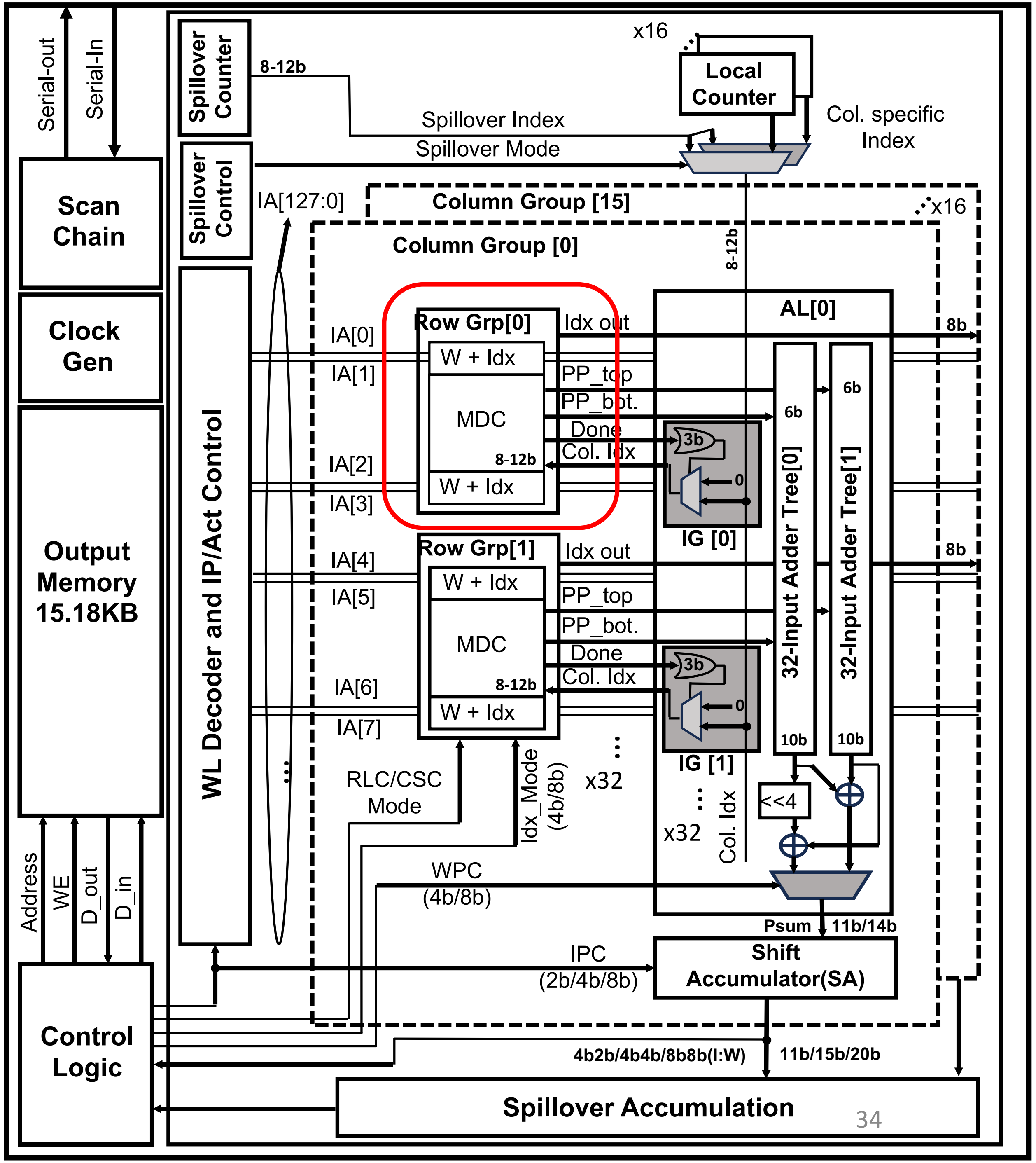
Amitesh Sridharan, Fan Zhang, Jae-sun Seo, Deliang Fan, “SP-IMC: A Sparsity Aware In-Memory-Computing Macro in 28nm CMOS with Configurable Sparse Representation for Highly Sparse DNN Workloads,” *IEEE Custom Integrated Circuits Conference (CICC)*, 2024.

Sparse(SP)-IMC Macro Design

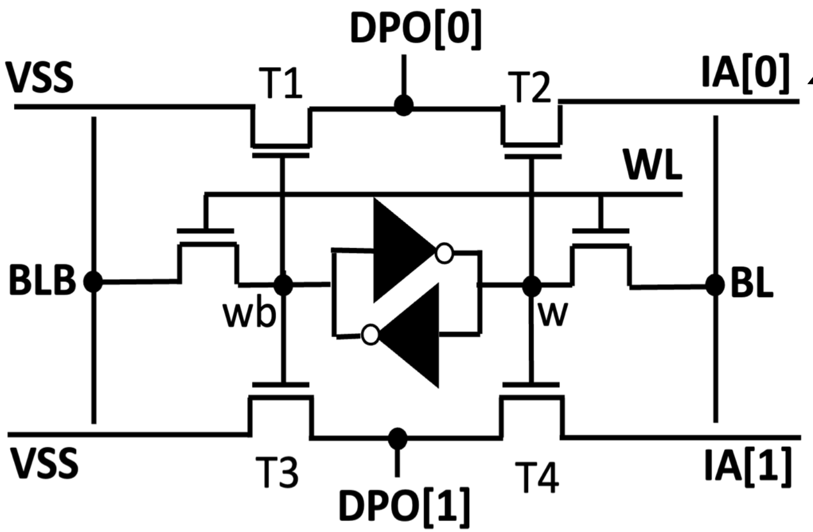


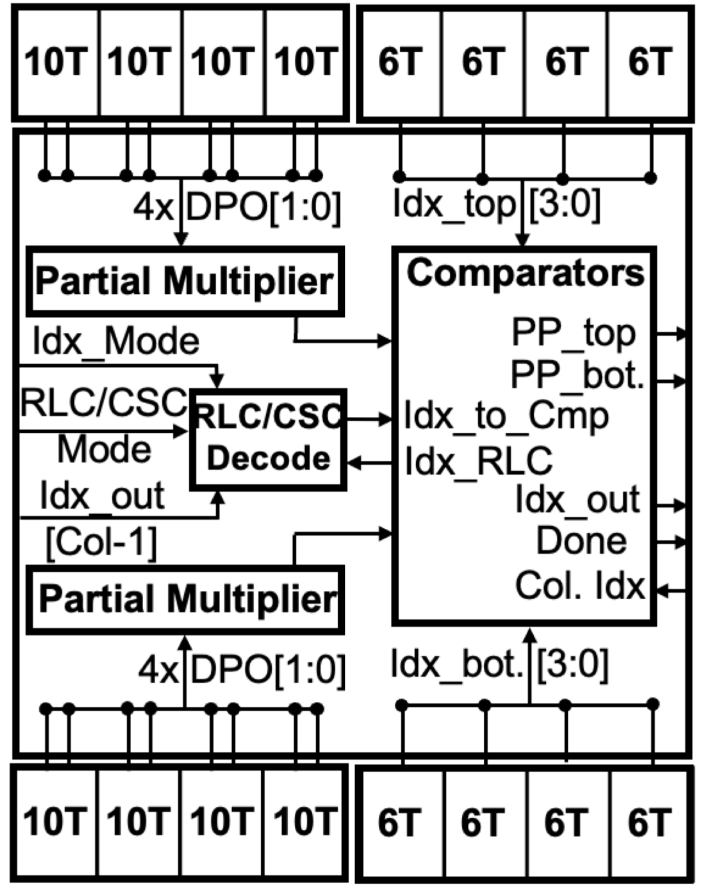
- SP-IMC supports different sparse encoding schemes: Compressed Sparse Column (**CSC**), Run Length Encoding (**RLE**)
- un-structured or **N:M** structured sparsity, with varying index/zero counts from 4 - 8 bit.
- Sparse processing circuits in each row group

Amitesh Sridharan, Fan Zhang, Jae-sun Seo, Deliang Fan, "SP-IMC: A Sparsity Aware In-Memory-Computing Macro in 28nm CMOS with Configurable Sparse Representation for Highly Sparse DNN Workloads," *IEEE Custom Integrated Circuits Conference (CICC)*, 2024.



Compute Row Group: In-Memory Logic Design





1bW:2bI partial-product

DPO[i]=IA[i] * w

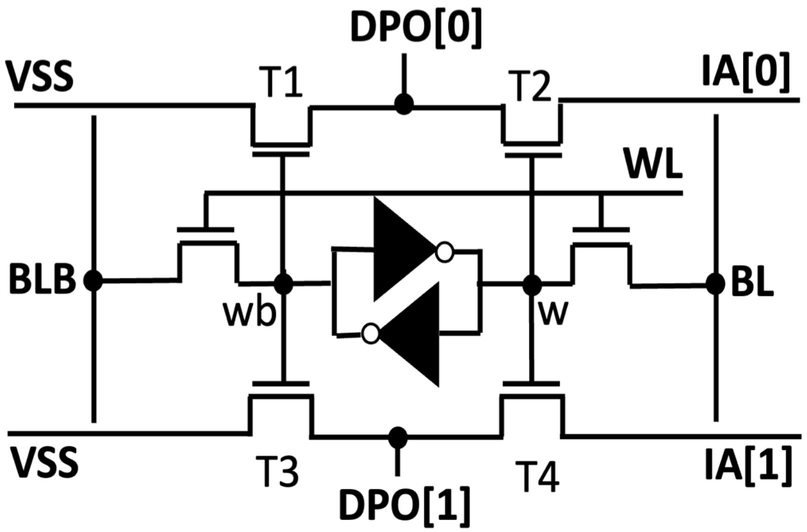
IA[i]: input activation [i]

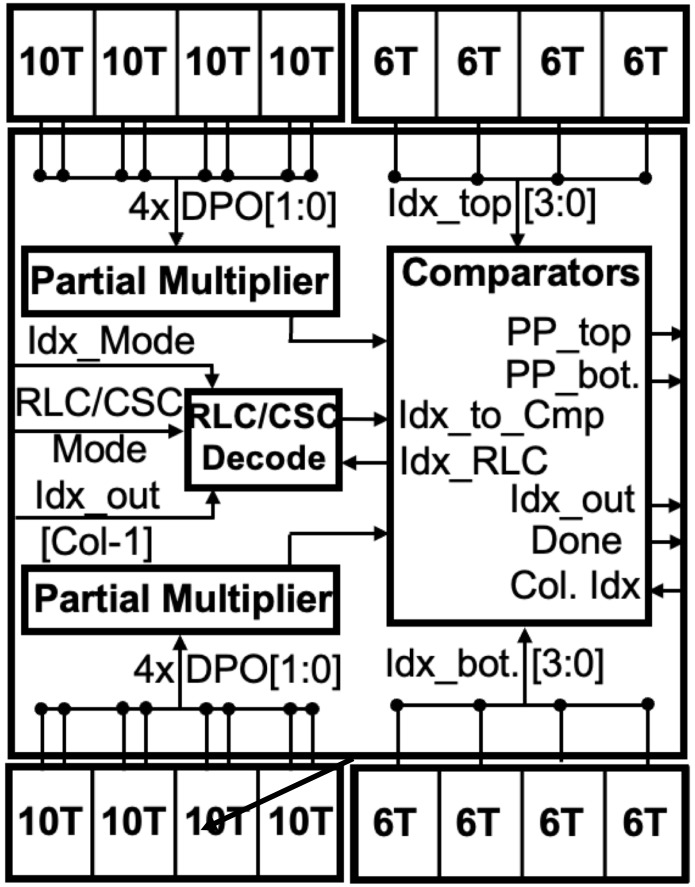
w: weight

- Each row group: 8-bits for weight, 8-bits for index.
- In-memory partial product: 10T **Dual AND** bit-cell design for **1bW:2bI** dot-products, with DPO[0:1] as 2 bit-parallel partial product outputs.

Compute Row Group: In-Memory Logic Design

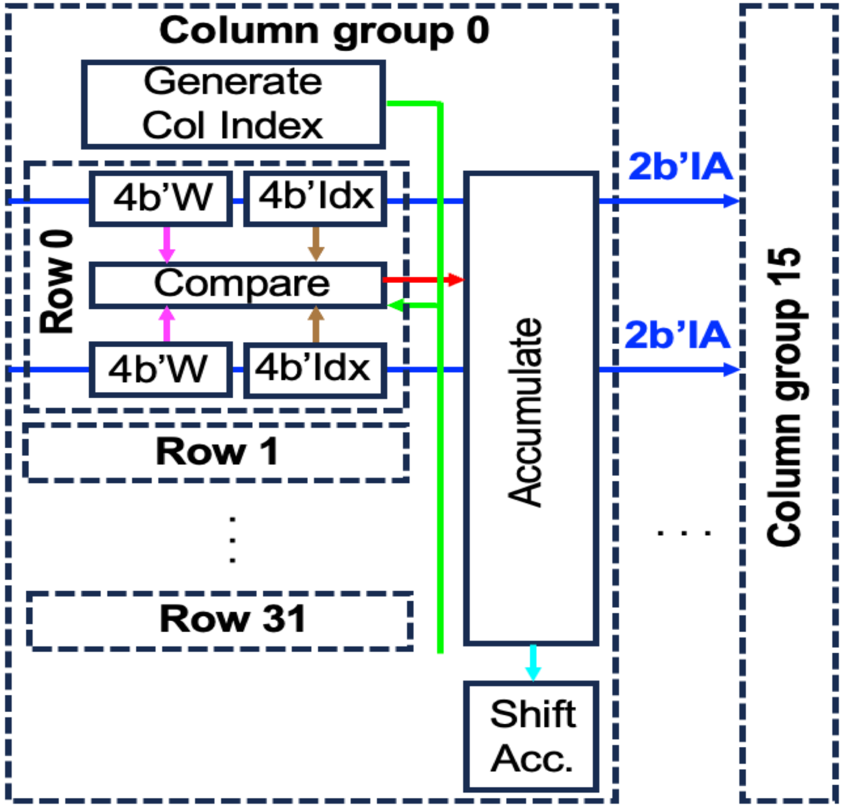
1bW:2bI partial-product

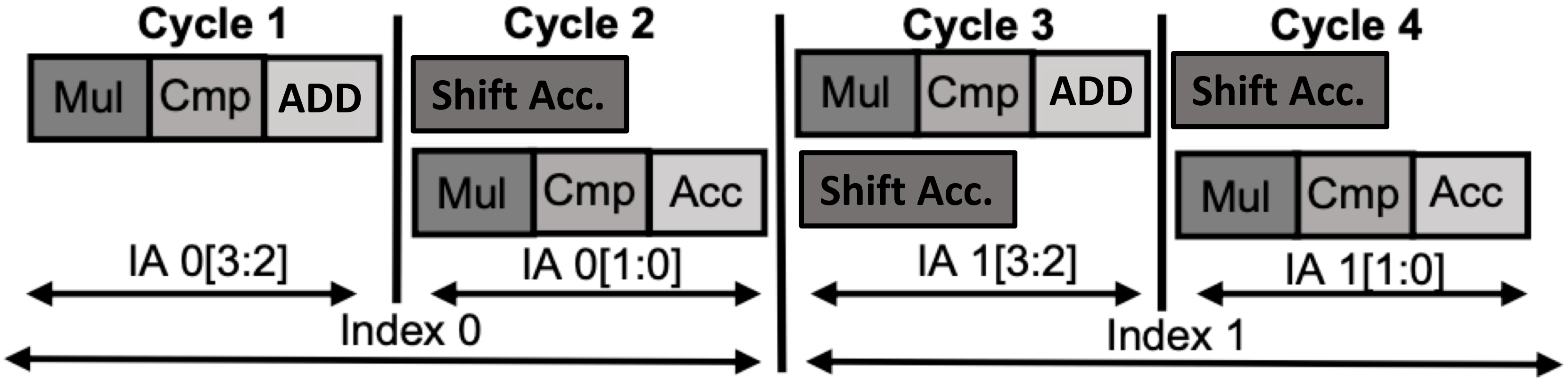




- SP-IMC can be **reconfigured** to select *2 4-bit indices* or a *single 8-bit index* depending on sparsity pattern and weight resolution.
- Each row group has a **Multiply, Decode, and Compare** logics
- MDC takes the dot-products from associated bit-cells, takes indices pertaining to the column and individual weight, and sends out the partial product if the comparison is successful.

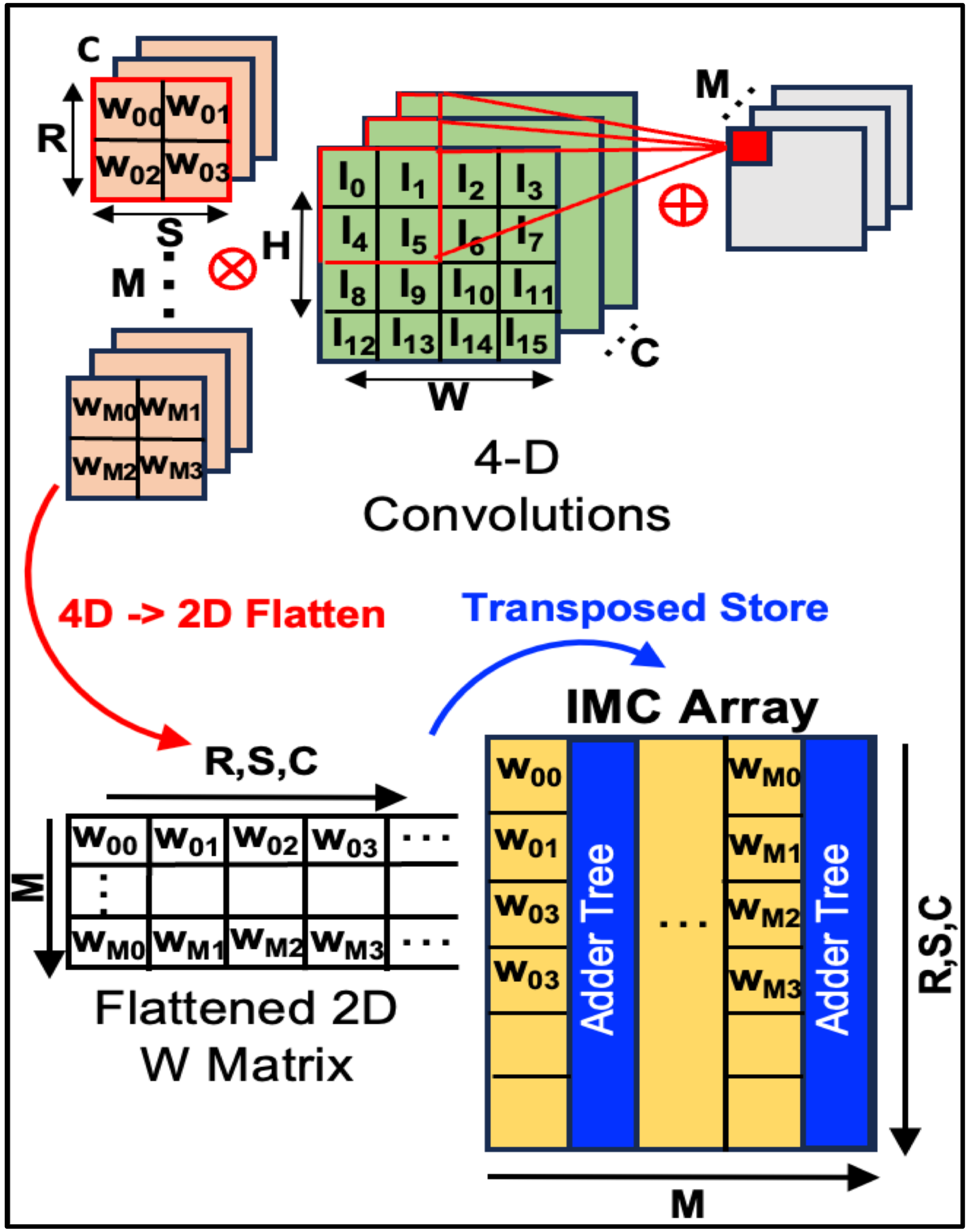
Pipeline Diagram of SP-IMC in 4b-IA:4b-W Mode





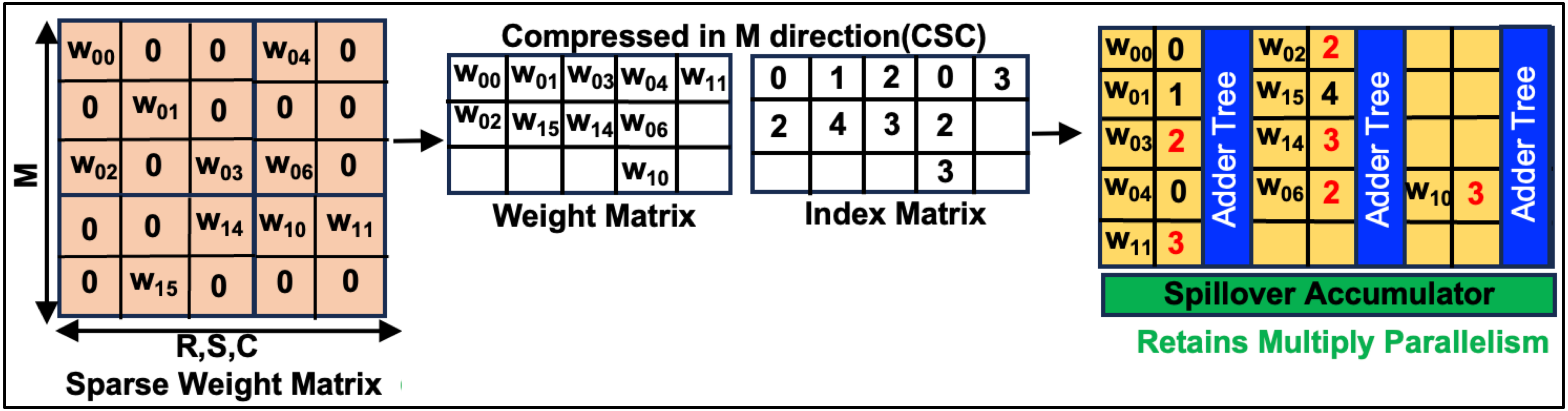
- Two pipeline cycles to finish one 4bIA:4bW MAC

Uncompressed Convolution Mapping to IMC



- Uncompressed mapping for convolutions is done by flattening the 4D kernels to a 2D weight matrix
- Then it is transposed and stored onto the IMC array such that the kernel dimensions and input channel (R, S, C) fall into columns with adder trees
- The output channel is mapped in the row direction
- Such mapping is widely used in IMC to support parallel multiplications and improve MAC throughput.

CSC Compression and Mapping



w_{00}	0	0	w_{04}	0
0	w_{01}	0	0	0
w_{02}	0	w_{03}	w_{06}	0
0	0	w_{14}	w_{10}	w_{11}
0	w_{15}	0	0	0

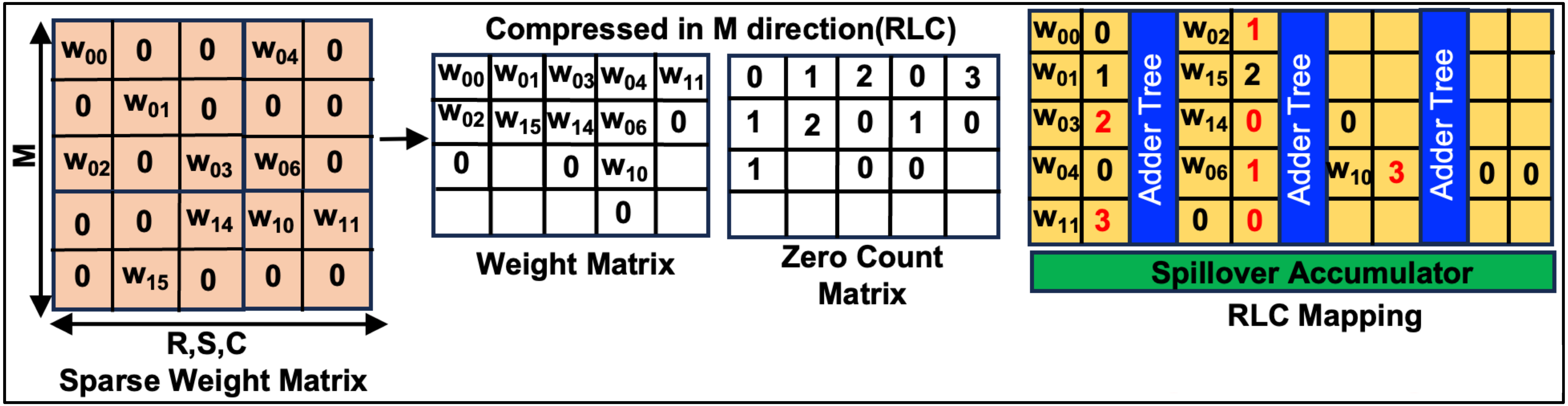
w_{00}	w_{01}	w_{03}	w_{04}	w_{11}
w_{02}	w_{15}	w_{14}	w_{06}	
			w_{10}	

0	1	2	0	3
2	4	3	2	
			3	

w_{00}	0	w_{02}	2		
w_{01}	1	w_{15}	4		
w_{03}	2	w_{14}	3		
w_{04}	0	w_{06}	2	w_{10}	3
w_{11}	3				

- Compressed mapping retains the same structure as un-compressed mapping, R,S,C in column direction and M in row-direction to support parallel multiplication.
- CSC breaks accumulation parallelism and retains multiplication parallelism, as suitable for IMCs with shared multiplications.
- N:M structured pruning is preferred to maximize storage efficiency

RLC Compression and Mapping



w_{00}	0	0	w_{04}	0
0	w_{01}	0	0	0
w_{02}	0	w_{03}	w_{06}	0
0	0	w_{14}	w_{10}	w_{11}
0	w_{15}	0	0	0

Sparse Weight Matrix

w_{00}	w_{01}	w_{03}	w_{04}	w_{11}
w_{02}	w_{15}	w_{14}	w_{06}	0
0		0	w_{10}	
			0	

Weight Matrix

0	1	2	0	3
1	2	0	1	0
1		0	0	

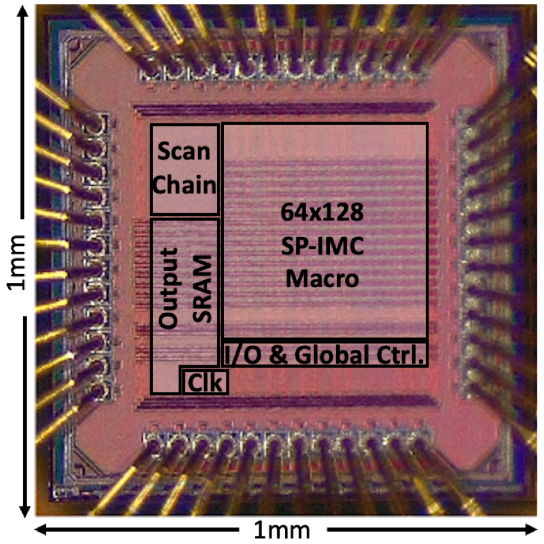
Zero Count Matrix

w_{00}	0	Adder Tree	w_{02}	1	Adder Tree			Adder Tree		
w_{01}	1		w_{15}	2						
w_{03}	2		w_{14}	0		0				
w_{04}	0		w_{06}	1		w_{10}	3		0	0
w_{11}	3		0	0						

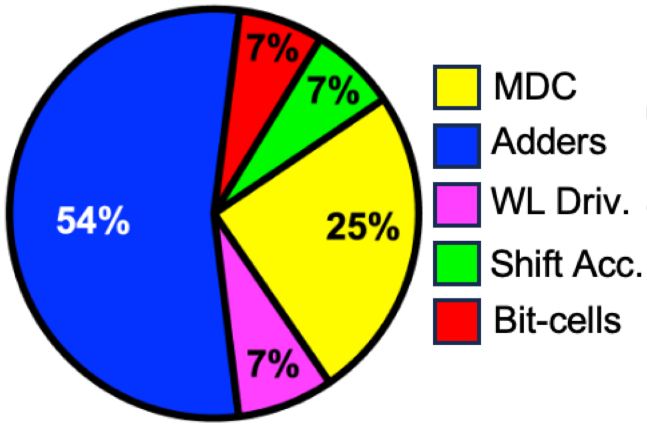
RLC Mapping

- Compressed mapping retains the same structure as un-compressed mapping, R,S,C in column direction and M in row-direction to support parallel multiplication.
- RLC follows same mapping as CSC, but zero count of elements before the non-zero is stored
- N:M structured pruning is preferred to maximize storage efficiency

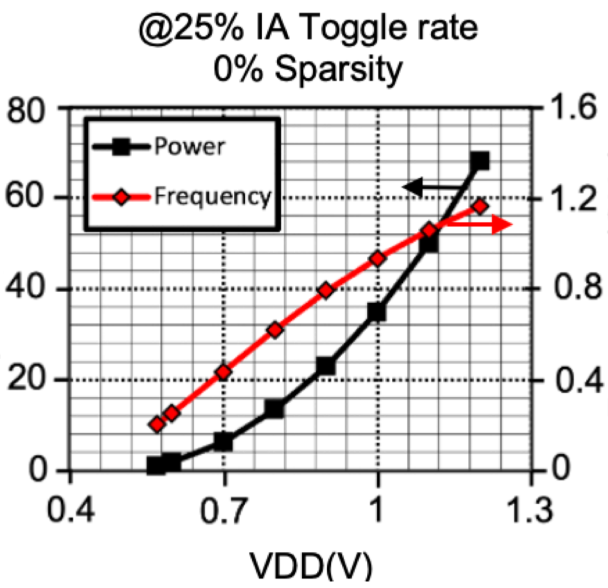
Chip Measurement and Performance



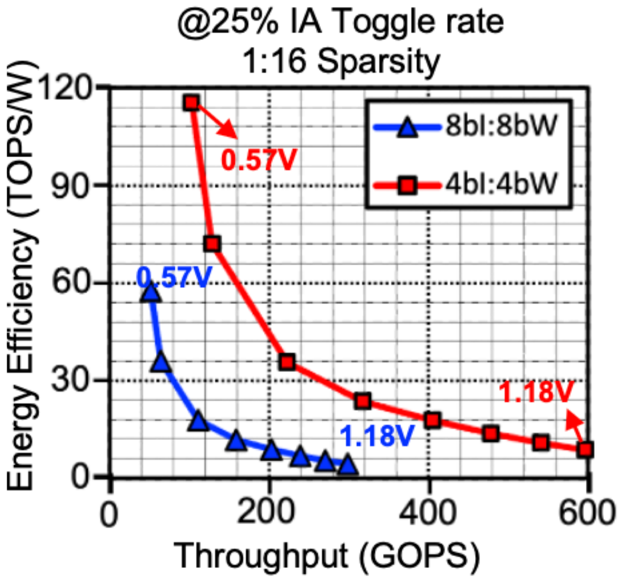
Die micrograph



Area Breakdown



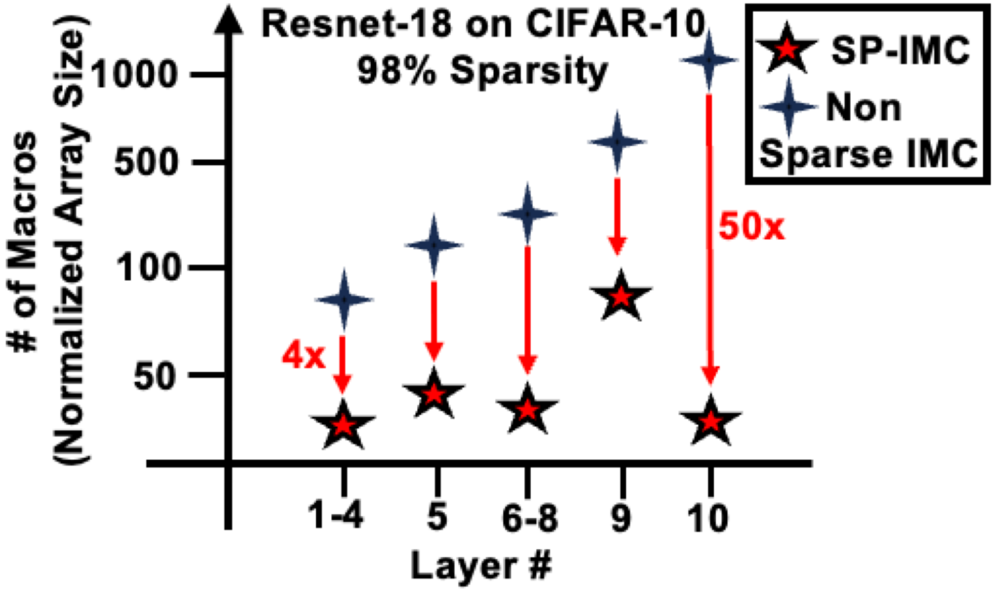
VDD Scaling

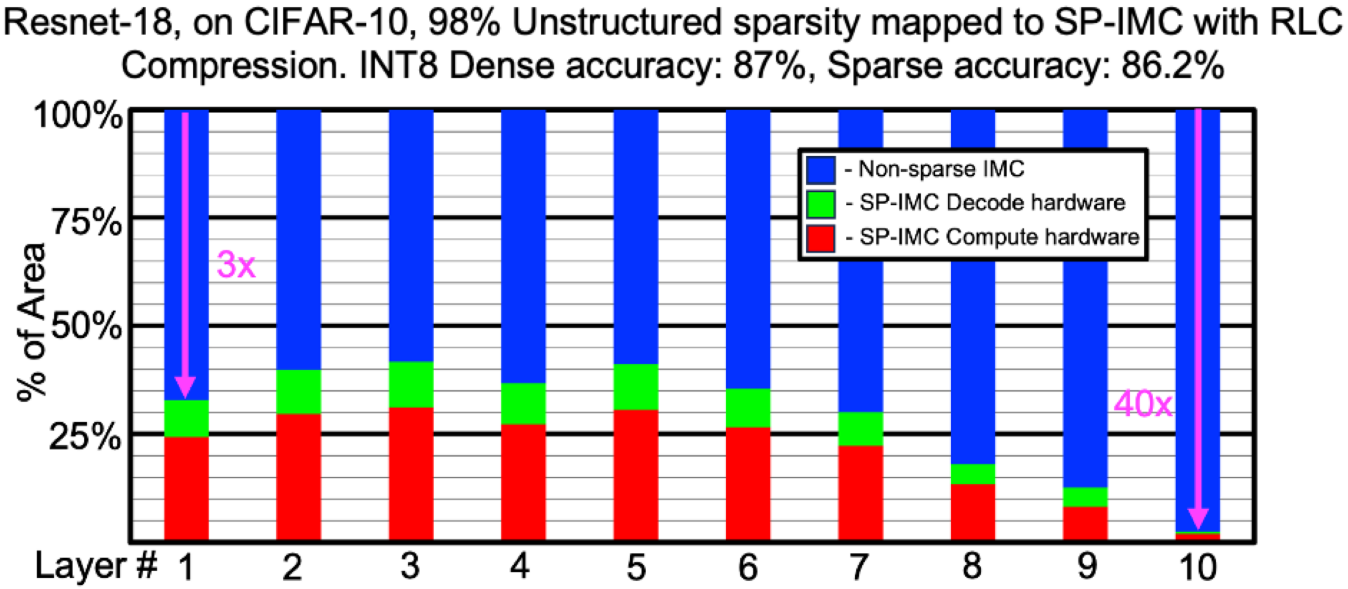


TOPS and TOPS/W Scaling

- SP-IMC prototype chip is prototyped using **28nm** TSMC CMOS.
- The chips are measured at 25^0C, 25% IA toggle rate between 1.18V and 0.57V.
- The prototype chip achieves a maximum of **1.16GHz** at **1.18V** consuming **72mW** of power.
- Due to its all-digital nature, it shows good scaling, down to 0.57V VDD maintaining an F_{max} of 201 MHz consuming only 3.6mW of power.

Benefits of Sparse Storage and Processing-in-Memory

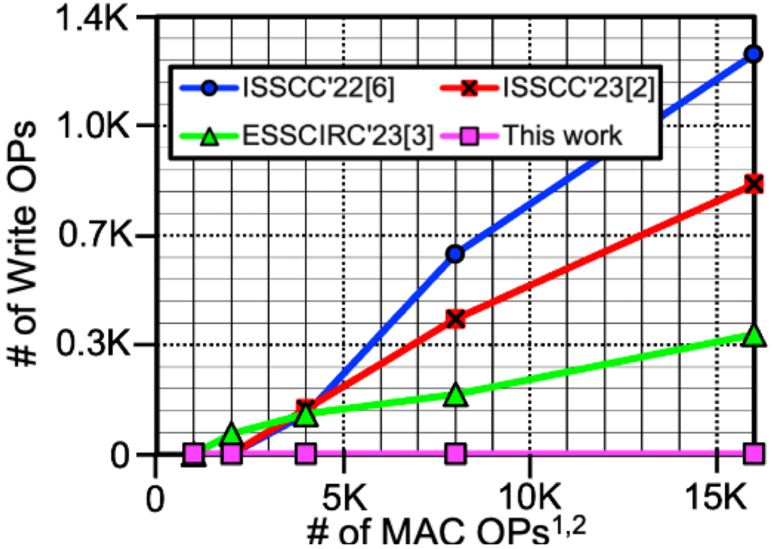




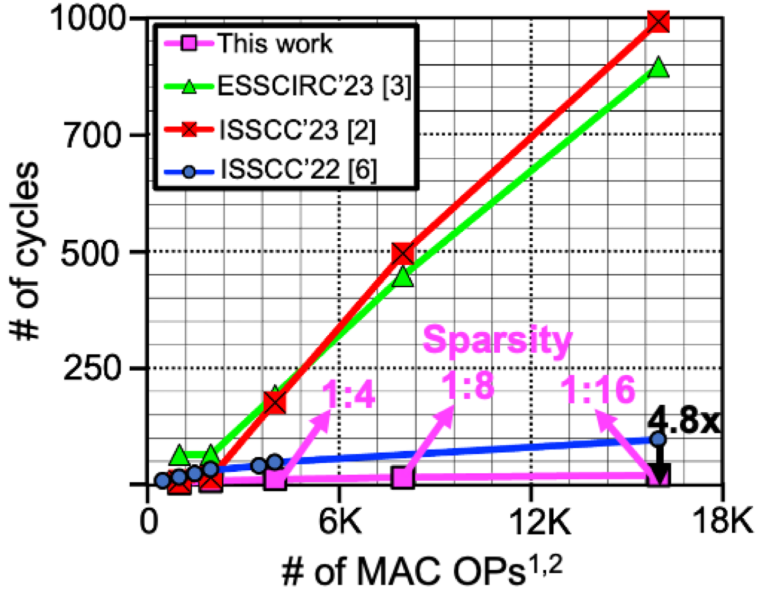
- Due to sparse-storage and sparse-processing, SP-IMC could process larger DNN model with less hardware cost (e.g. less Macros, less memory, less power, etc.)
- # of SP-IMC vs non-SP-IMC macros required to map a pruned Resnet-18 trained on CIFAR-10 will be significantly reduced by more than 10X.

Comparison with prior works (1 of 2)

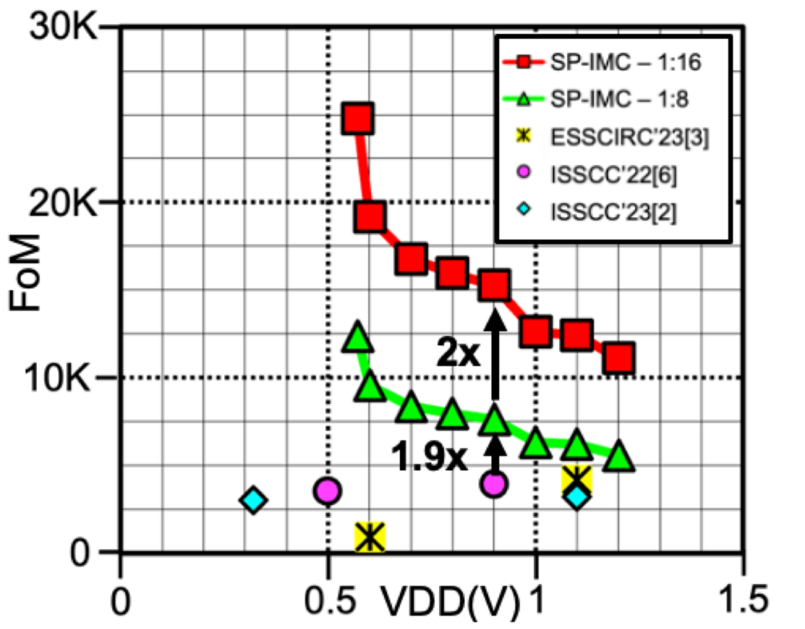
Aided by compressed sparse storage, SP-IMC can greatly reduce # of write OPs for IMC macros .



Includes write latency - 1 cycle/word size



FoM = TOPS/W$^{(7)}$ * TOPS/mm$^{2\,(7)}$ * # of W/kb



- Due to sparse encoding, SP-IMC could store more *weights/kb*.
- In the system level, this translates to the reduction of writes operations to IMC, as we scale the number of MAC operations.
- Less processing latency could also be achieved with directly processing sparse-encoded weights
- Highest FoM considering Energy efficiency * area efficiency * weights/Kb

Comparison with prior works (2 of 2)

Work	ISSCC'22 [1]	ISSCC' 23 [4]	ISSCC'22 [6]	ISSCC'23 [2]	ESSCIRC'23 [3]	**This Work**
Technology	28nm	28nm	5nm	4nm	28nm	28nm
IMC Sparsity Support	X	X	X	X	X	RLC/CSC/N:M
Supply Voltage (V)	0.45-1.10	0.64-1.03	0.5-0.9	0.32-1.1	0.9-1.1	0.57-1.18
Macro Area (mm^2)	0.049	NA	0.0133	0.0172	0.0159	0.24
Clock Frequency (MHz)	250	20-320	360-1440	1490	30-360	201-1160

Bitcell Transistors	8T	8T(55%)
		10T(45%)
			12T	8T x 2bit
				+OAI
					6T+0.5T	6T+4T(50%)
						6T(50%)

Array Size(b)	16K	1.15M	64K	54K	16K	4K(Weights)
						+ 4K(Index)

Bit Precision	IA:1-4b W:1b	INT8	IA: 1-8b W:4b	IA: 8/12/16 W: 8/12	IA: 1-8 W: 8	IP:2b/4b/8b W:4b/8b
Full output precision	No	Yes	Yes	Yes	Yes	Yes
Performance(GOPS)[1,2,7]	62.5*	22.9*	104.735	127.15	0.95-11.6	**41.29-238.86[6]**
Peak Energy Efficiency[2,7] (TOPS/W)	9.6-15.5	15.6[4]/70.37[5] (System)	17.5-63	87.4[(8)] 41.3[(9)]	22.4-60.4	4.38-57.67[6]
Compute Density[2,3,7] TOPS/mm^2	2.59	0.85	0.44-1.76	0.27-1.01	0.12-1.46	0.21-1.2[7]

FoM	3.182K	2.97K[4]
		3.25K[5]
			3.942K	3.219K	0.9K-4.1K	**11K-24K(1:16)**
						5.5K-12K(1:8)

[(1)] Normalized to 8Kb. [(2)] One operation is either 8b multiplication or addition. [(3)] Normalized quadratically to 28nm. *Estimated from previous works. [(4)] 75% Sparsity, [(5)] 92% Sparsity. [(6)] 93.75% Sparsity (1:16 Sparsity).
GOPS Calculation: 32(Rows) x 16(Columns)/Latency(5xClk period). [(7)]Excludes write energy/latency otherwise incurred by other works for a scaled-up matrix that fits in SP-IMC and not in other works. [(8)] @ 12.5% TR. [(9)] @ 50% TR [(10)] @25% TR.

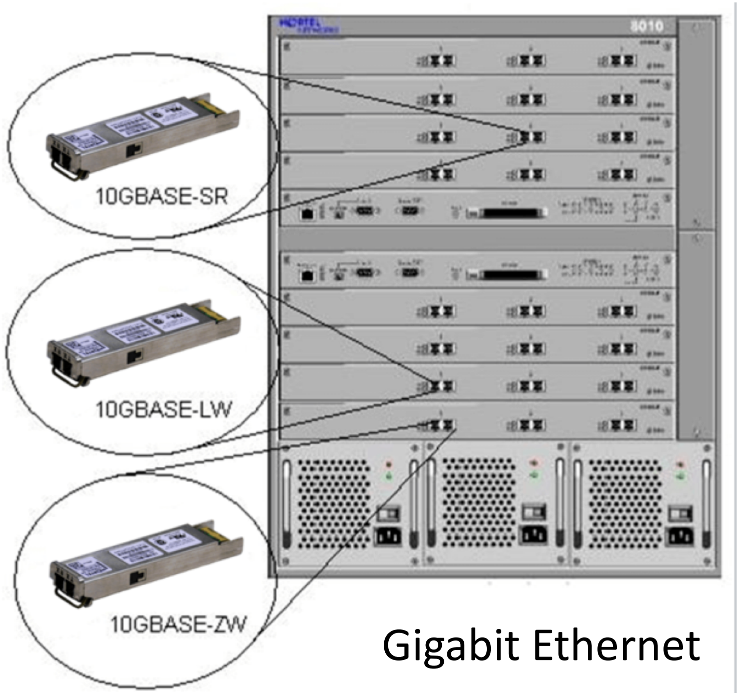
- Best Performance (GOPS)
- SoTA Energy Efficiency
- Best FoM

FoM = TOPS/W[(7)] * TOPS/mm^2 [(7)] * # of W/kb

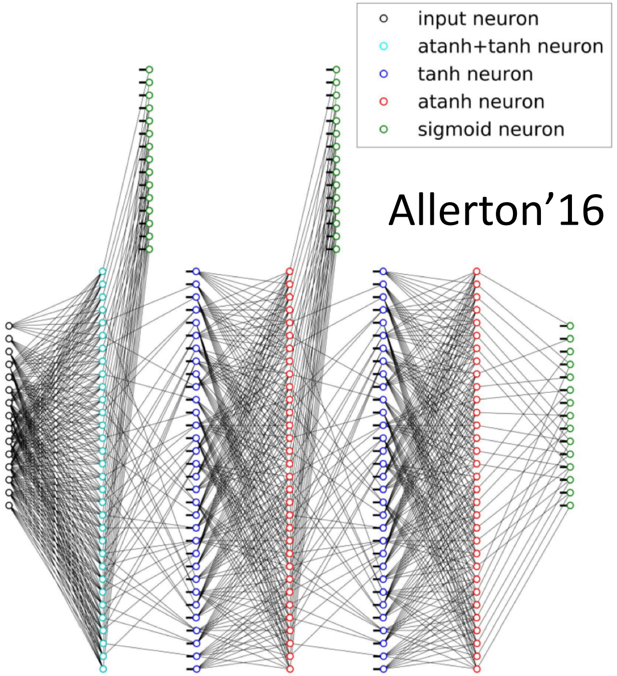
Sparse Matrix Multiplication for Communication Application:

Neural Belief-Propagation (BP) Decoder

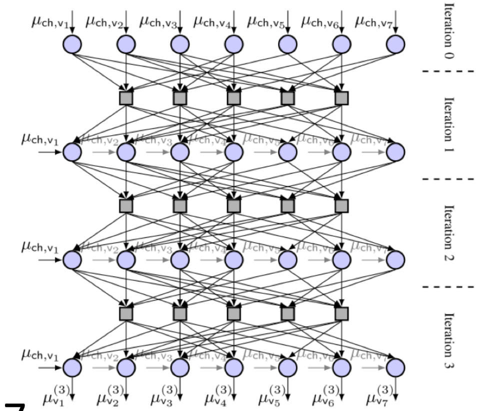
WiFi 6™



Gigabit Ethernet



Allerton’16



JSAC’17

Amitesh Sridharan, Fan Zhang, Yang Sui, Bo Yuan and Deliang Fan, “DSPIMM: Digital Sparse In-Memory matrix vector multiplier for Communication Applications” *In: 59th Design Automation Conference (DAC)*, San Francisco, CA, July 9-13, 2023

Sparse Matrix Multiplication for Communication Application: Neural Belief-Propagation (BP) Decoder

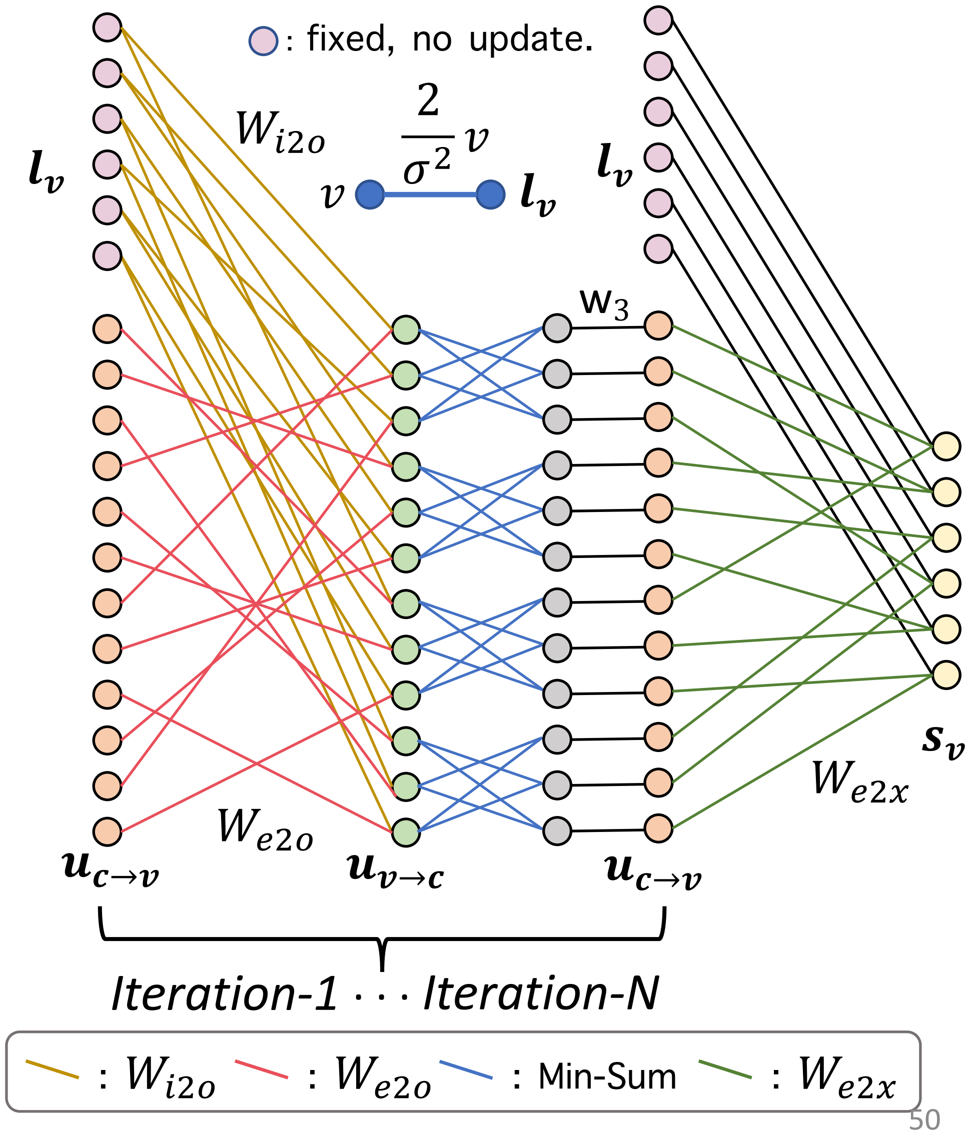
- Step 1: $u^{t}_{v\to c} = \mathbf{W}_{i2o} \times l_v + \mathbf{W}_{e2o} \times u^{t-1}_{c\to v}$
  - Structured Sparse MVM -- $\mathbf{W}_{i2o} \times l_v$
  - Unstructured Sparse MVM -- $\mathbf{W}_{e2o} \times u^{t-1}_{c\to v}$

- Step 2 & 3: Min-Sum and Dot-Product Compute. $u^{t}_{c\to v} = w_{c\to v} \times \min_{v'\in M(c)\setminus v} |u^{t}_{v'\to c}| \prod_{v'\in M(c)\setminus v} \text{sign}(u^{t}_{v'\to c}),$

$$u_{c1\to v1} = w \times sign(u_{v2\to c1}) \times sign(u_{v4\to c1}) \times min(|u_{v2\to c1}|, |u_{v4\to c1}|)$$

Iteration-1

- Step 4: SS_v Calculation $s^{t}_{v} = l_v + \mathbf{W}_{e2x} \times u^{t}_{c\to v},$
  - Performed only once for the last iteration.



Performance Evaluation

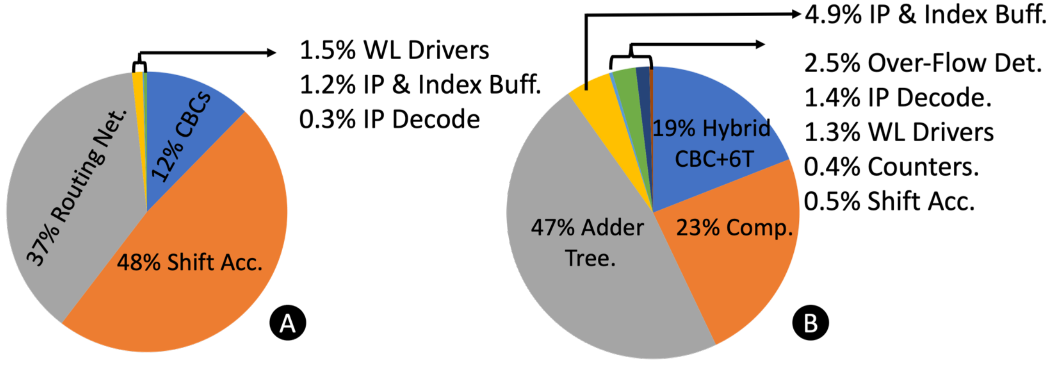
GWCL ALGORITHM MEMORY BENEFITS(EXCLUDES INDEX MEMORY)

Code Length/	121		672		1056	
Weight Memory	Uncompressed	GWCL algorithm	Uncompressed	GWCL algorithm	Uncompressed	GWCL algorithm
W_1	73.2KB	0.6KB	1.5MB	2.2KB	3.7MB	3.52KB
W_2	366KB	2.4KB	5MB	5.5KB	12.3MB	8.7KB
W_3	366KB	N/A	4.9MB	N/A	12.3MB	N/A
W_4	73.2KB	0.6KB	1.5MB	2.2KB	3.7MB	3.52KB

Significant memory saving due to sparse encoding & processing, the sparsity ratio is very high in Neural BP

POWER BREAKDOWN

SSP-Matrix Mem(256x256)		USP-Matrix Mem(128x256)	
Hardware	Power (mW)	Hardware	Power(mW)
Bit-Cell array(8T)	11.6mW	Bit-cell array(6T+8T)	4.2mW
Shift Accumulator	46.73mW	Comparator	21.3mW
Routing Network	R+C Parasitics	Adder Tree	18.7mW
IP Index+IP Buff.	6.3mW	Ip Index + Ip Buff.	4.22mW
Decoder	0.88mW	Shift Accumulator	6.74mW
Ip Decode	1.3mW	Overflow + Counters	4.63mW
Total	66.81mW	Total	59.79mW



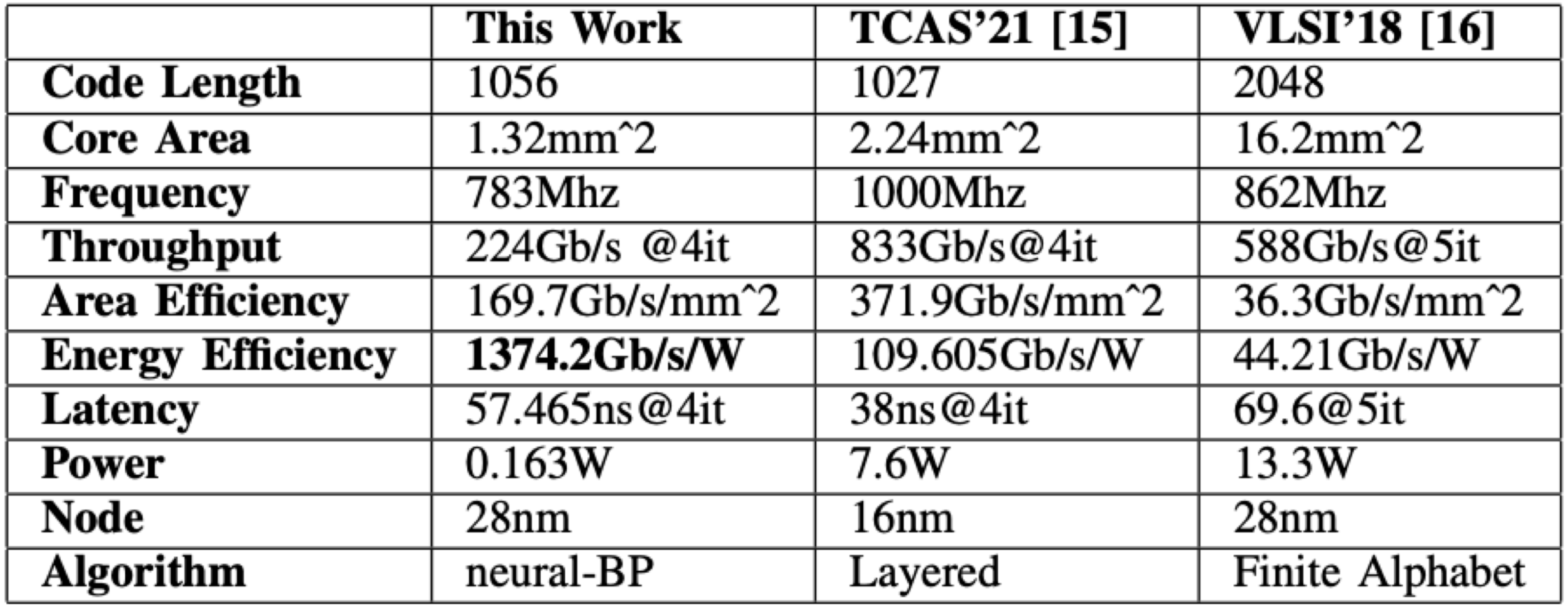
(A) Area Breakdown of SSP-MVM (B) Area Breakdown of USP-MVM

- The majority operations in Neural BP is sparse matrix multiplication
- Our designed SP-IMC could significantly reduce the model size, thus chip area and power consumption

Amitesh Sridharan, Fan Zhang, Yang Sui, Bo Yuan and Deliang Fan, “DSPIMM: Digital Sparse In-Memory matrix vector multiplier for Communication Applications” *In: 59th Design Automation Conference (DAC)*, San Francisco, CA, July 9-13, 2023

Comparison with SOTA

COMPARISON WITH PRIOR LDPC IMPLEMENTATIONS



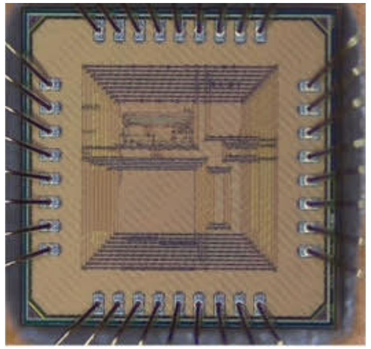
	This Work	**TCAS'21 [15]**	**VLSI'18 [16]**
Code Length	1056	1027	2048
Core Area	1.32mm^2	2.24mm^2	16.2mm^2
Frequency	783Mhz	1000Mhz	862Mhz
Throughput	224Gb/s @4it	833Gb/s@4it	588Gb/s@5it
Area Efficiency	169.7Gb/s/mm^2	371.9Gb/s/mm^2	36.3Gb/s/mm^2
Energy Efficiency	**1374.2Gb/s/W**	109.605Gb/s/W	44.21Gb/s/W
Latency	57.465ns@4it	38ns@4it	69.6@5it
Power	0.163W	7.6W	13.3W
Node	28nm	16nm	28nm
Algorithm	neural-BP	Layered	Finite Alphabet

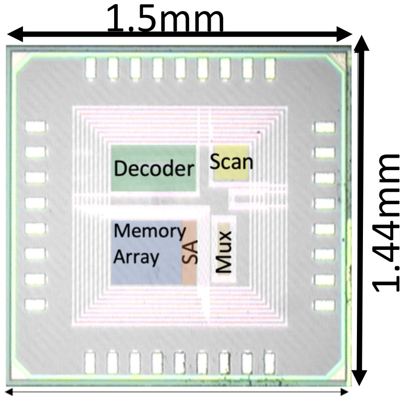
- Compared with non-IMC ASIC implementation of Channel decoder, **~10X higher energy efficiency** could be achieved

Amitesh Sridharan, Fan Zhang, Yang Sui, Bo Yuan and Deliang Fan, "DSPIMM: Digital Sparse In-Memory matrix vector multiplier for Communication Applications" *In: 59th Design Automation Conference (DAC)*, San Francisco, CA, July 9-13, 2023

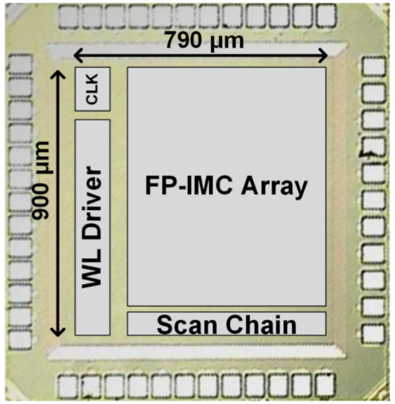
Our Developed IMC Chip Prototypes (examples): SRAM and NVM

- A 1.23-GHz 16-kb **Programmable and Generic Processing-in-SRAM** Accelerator in **TSMC 65nm**
- Published in ESSCIRC'2022

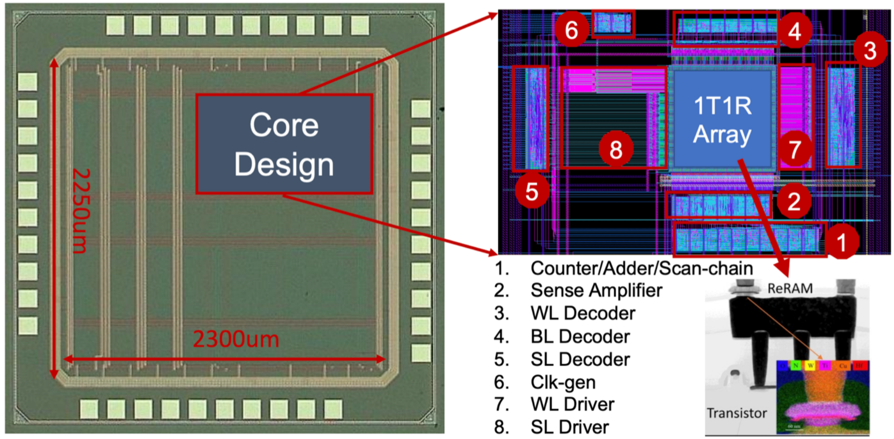




- All-Digital **Configurable Floating-Point** In-Memory Computing Macro in **TSMC 28nm**
- Published in ESSCIRC'2023



- **hybrid 65nm CMOS-ReRAM (HfO2)** designs for **DNN** acceleration and **DNA genome sequence alignment** applications.
- 64 by 64 HfO2 RRAM and SRAM module for hybrid computing

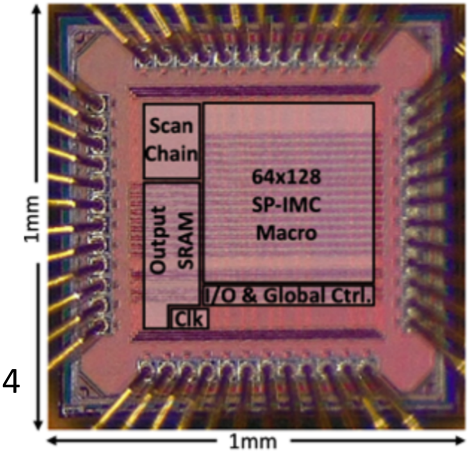


1. Counter/Adder/Scan-chain
2. Sense Amplifier
3. WL Decoder
4. BL Decoder
5. SL Decoder
6. Clk-gen
7. WL Driver
8. SL Driver

ESSCIRC'23, invited to extend to **JSSC** special issue

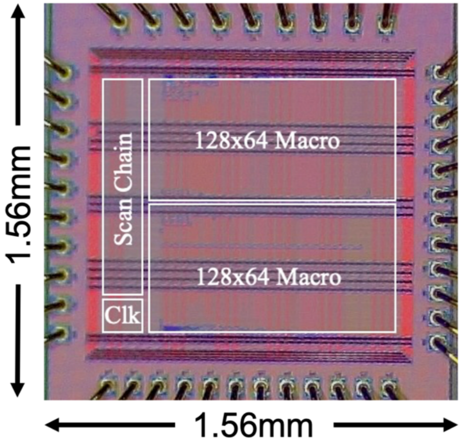
- **28 nm** IMC chip prototype for **sparse in-memory matrix convolution**
- Supports N:M structured sparsity, run length encoding, compressed sparsity column

- Sparse NN
- Wireless Decoder (Neural BP)



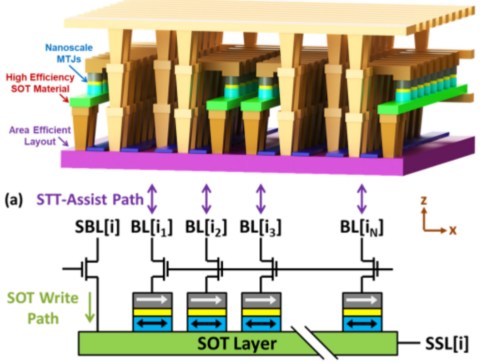
CICC'24

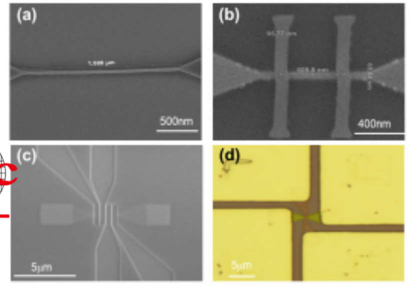
- A **28nm** 2385.7 TOPS/W/b **Precision Scalable** In-Memory Computing Macro with **Bit-Parallel** Inputs and Decomposable Weights for DNNs



SSCL

- energy efficient IMC with high-density, field-free **STT-assisted SOT-MRAM** (SAS-MRAM), ~**100nm, NSF FuSe/ACED projects**

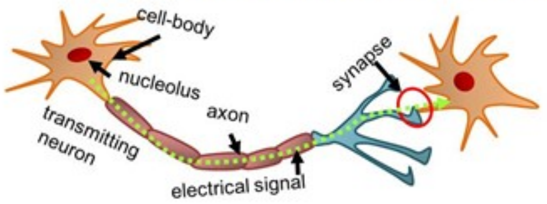


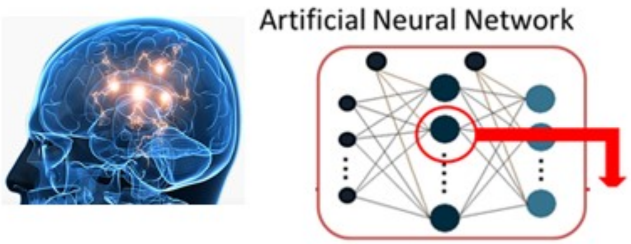


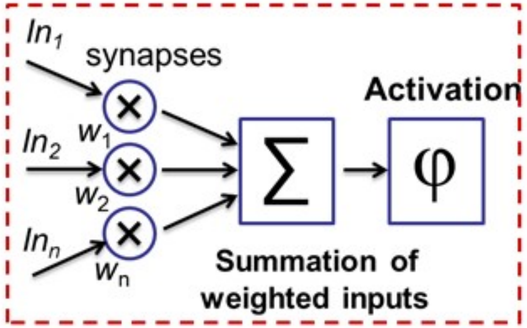
W. Hwang, D. Fan, S. X. Wang, et., al., TMAG, 2022

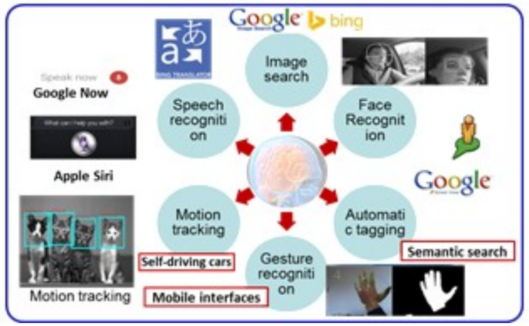
Analog: Neuromorphic Computing-in-Memory Devices & Circuits

PURDUE UNIVERSITY JOHNS HOPKINS UNIVERSITY ASU tsmc SUNY POLYTECHNIC INSTITUTE IBM intel

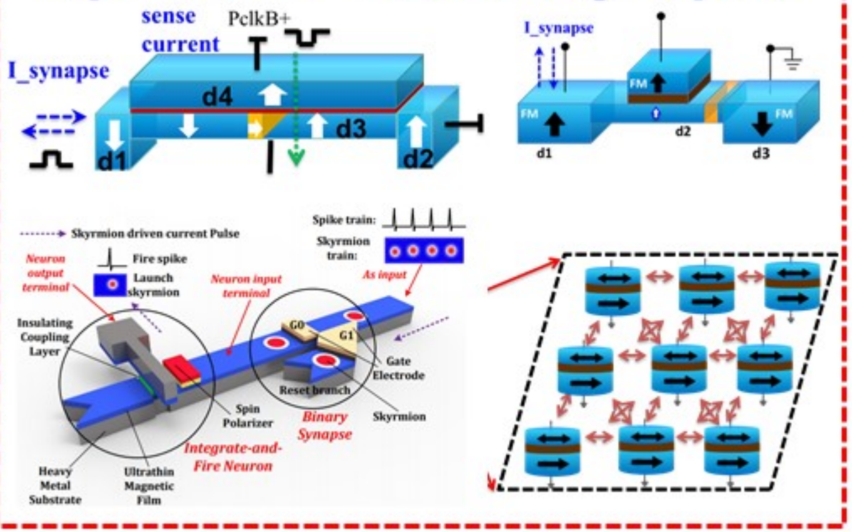




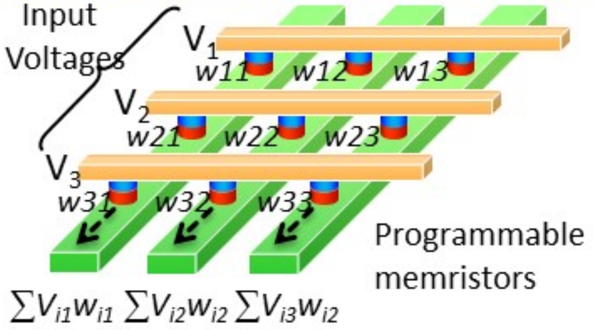


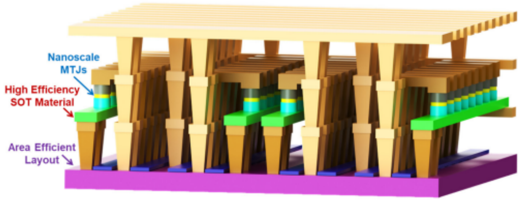


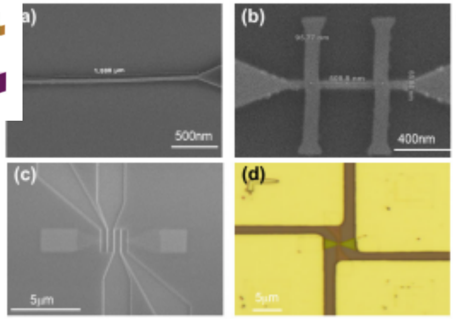
Spintronic neuron/synapse

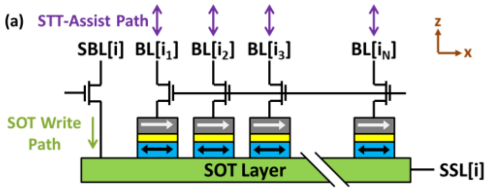


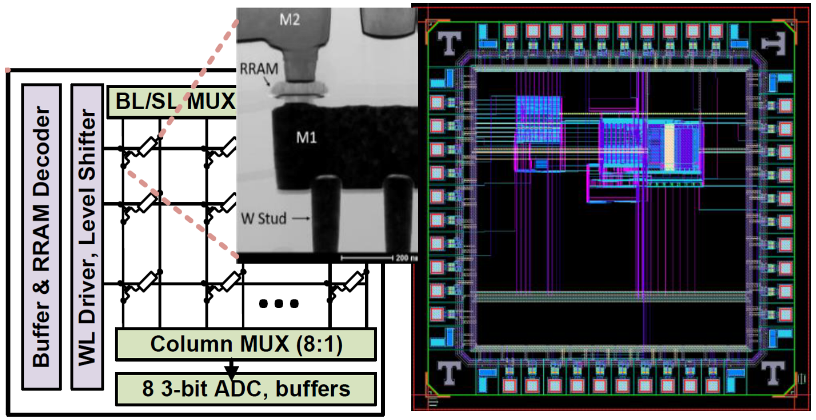
Memristor crossbar synapse



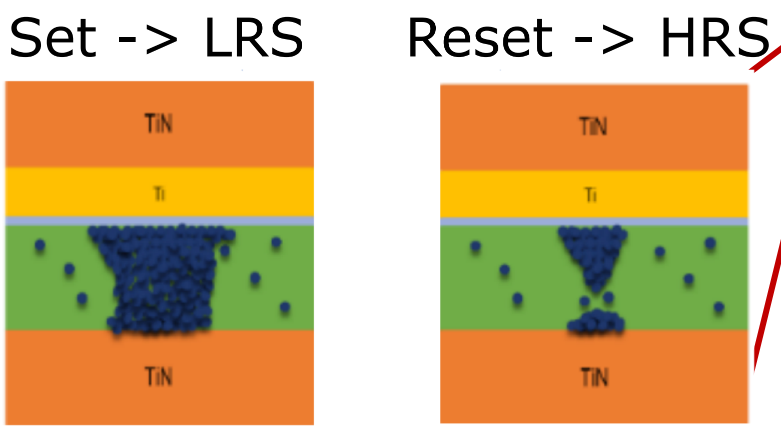


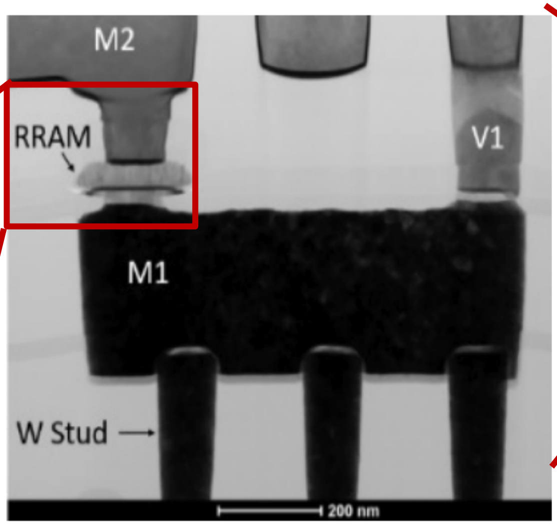


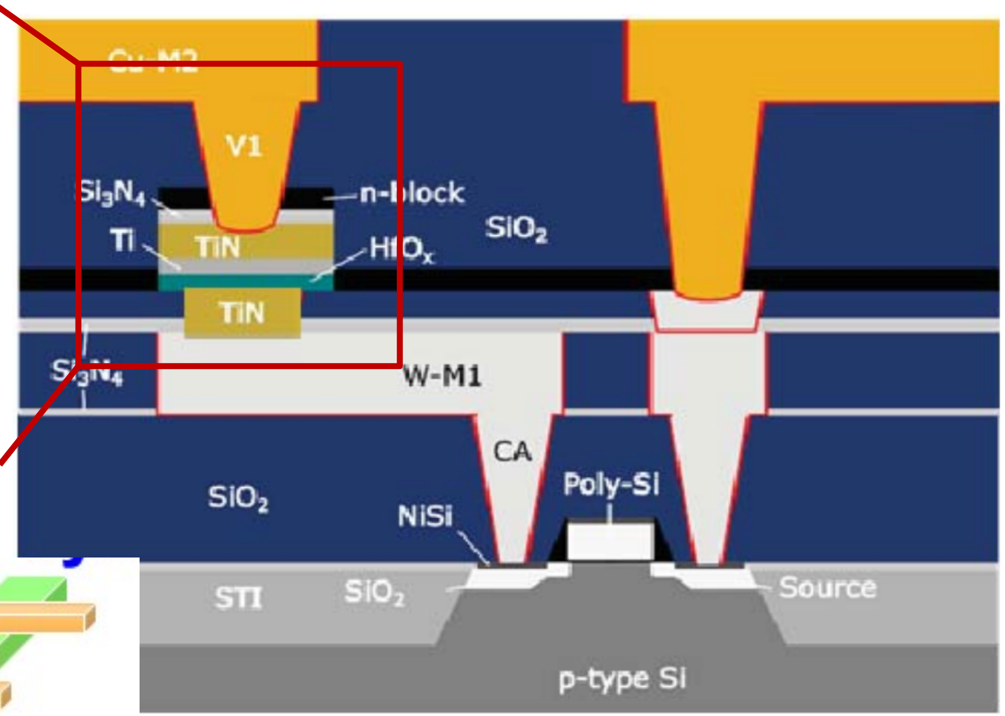


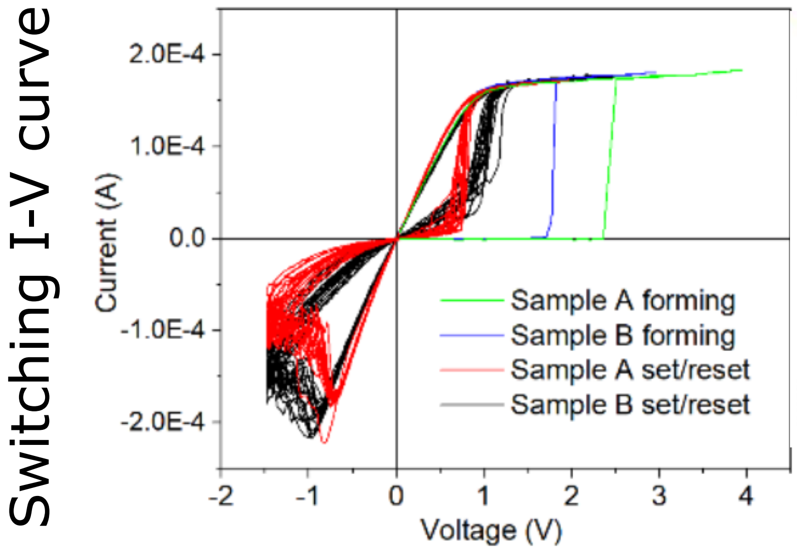


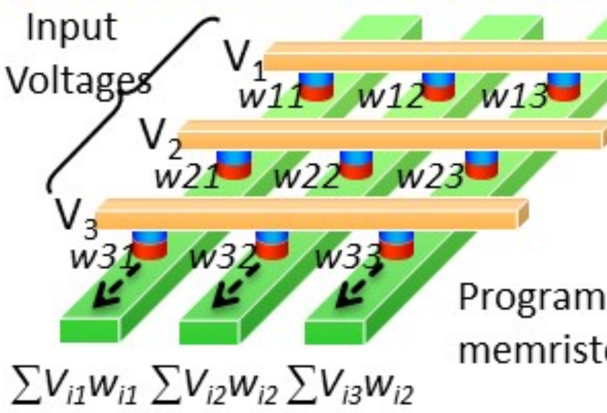
Resistive RAM Crossbar











1. M. Liehr, et,al, "Failure Analysis of 65nm CMOS Integrated Nanoscale ReRAM Devices on a 300mm Wafer Platform," *2022 IEEE International Integrated Reliability Workshop (IIRW)*, South Lake Tahoe, CA.
2. J. Hazra, et, al, "Improving the Memory Window/Resistance Variability Trade-Off for 65nm CMOS Integrated HfO2 Based Nanoscale RRAM Devices," 2019 IEEE International Integrated Reliability Workshop (IIRW), South Lake Tahoe, CA

- sample A: deposited thin layer of 6 nm HfO2 switching layer via atomic layer deposition using a chlorine-based precursor
- sample B: using an organic carbon-based precursor.

Crossbar based In-Memory Computing device & circuits

Why RRAM crossbar?

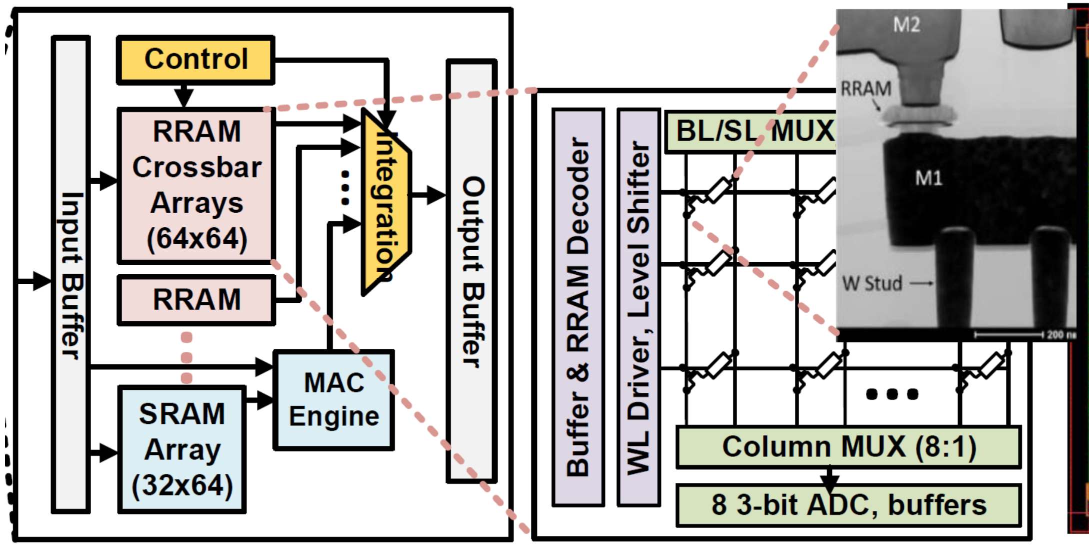
Pros: 1) O(1) convolution; **2)** multi-bit cell; **3)** mature technology demonstrated in 22nm chip by TSMC, etc. **4)** Non-volatility; **5)** area

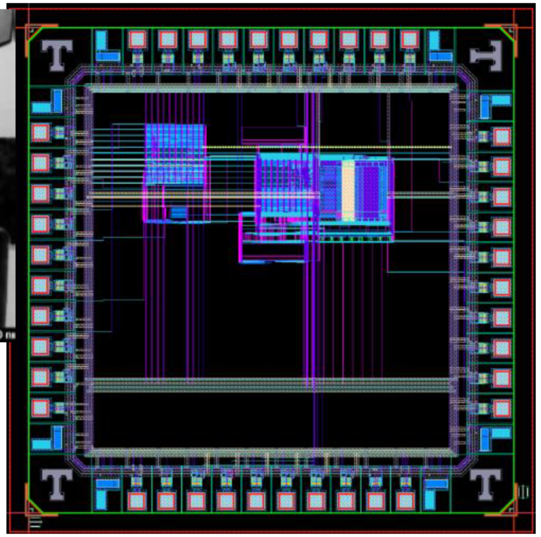
Cons: large write power/voltage, slow write, endurance, variation, non-ideal effect, unreliable, etc.

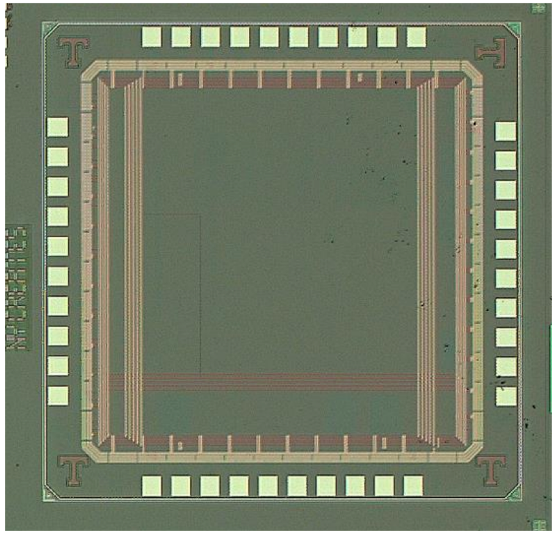
What we have developed to address those issues from <u>co-design</u> perspective.

1. Aggressive low bit width model compression (CVPR'19, DAC'17-22)
2. Crossbar-aware pruning, etc. (AAAI 2020, etc.)
3. Noise-aware and hardware non-ideality aware training (CVPR 2019, DAC'19/'20, TCAD'22, VLSI-TSA'22, etc.)
4. Crossbar–aware on-device learning/tuning methodology (DATE'22 (<u>best IP paper</u>), DAC'22 (<u>best paper candidate nomination of the track</u>), Frontier in Electronics, etc.) **Will discuss in later algorithm session**

Hardware perspective: Developed hybrid RRAM-SRAM In-Memory Computing (RS-IMC) chip



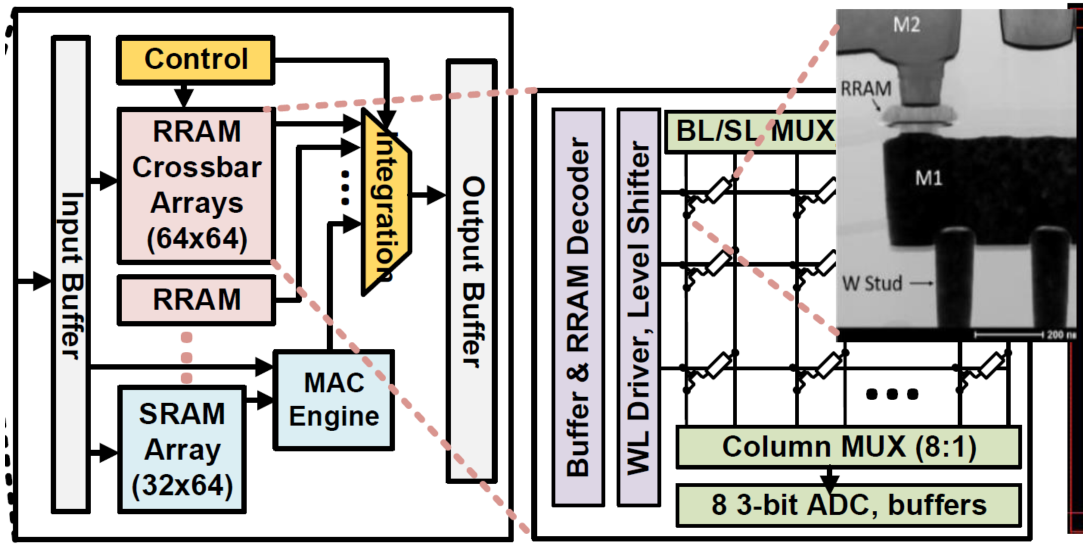


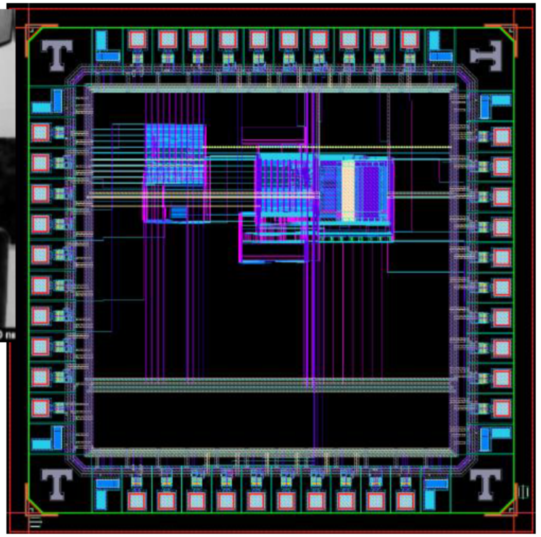


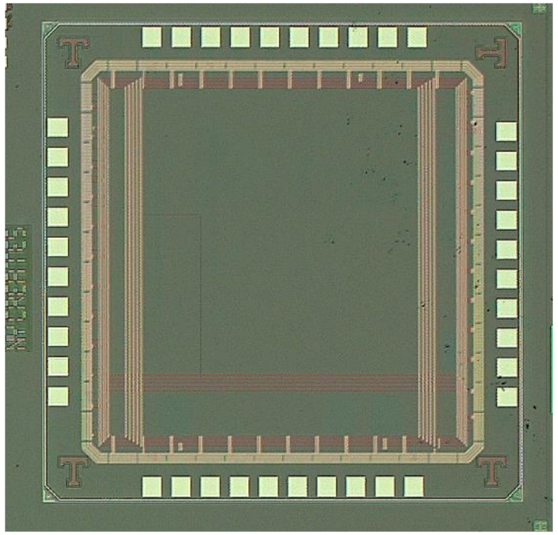
Project funded by NSF Career award

Crossbar based In-Memory Computing device & circuits

Developed hybrid RRAM-SRAM In-Memory Computing (RS-IMC) chip







RRAM-SRAM hybrid **prototype chip** with one 64×64 1T1R RRAM array (3-bit ADC) and one 32×64 SRAM array, and other peripheral circuits, with both RRAM crossbar and digital SRAM neuron module for hybrid computing

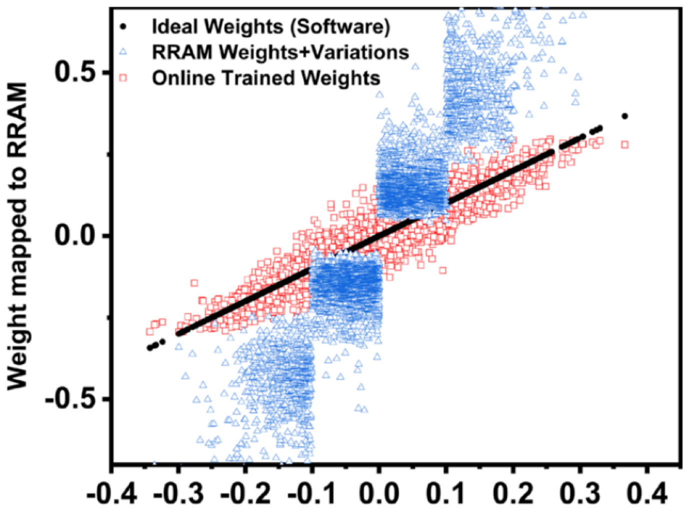
- The chip design was fabricated by SUNY Poly's Albany Nanotech Complex (collaborating with Prof. Cady and Cao) at 65nm technology
- Hybrid learning methodology with hybrid design (large RRAM and small SRAM parameters, introduce later)

RRAM Pros: 1) O(1) convolution; 2) multi-bit cell; 3) mature NVM technology in 22nm chip by TSMC, etc. 4) Non-volatility 5) area
RRAM Cons: large write power/voltage, slow write, endurance, variation, non-ideal effect, unreliable, etc.

SRAM Pros: 1) fast & easy to write; 2) unlimited endurance; 3) very mature technology with high reliability.
SRAM Cons: small capacity, large leakage, large area, volatile,

Project funded by NSF Career award

Digital-Assist Analog IMC: Accurate and Efficient Computing

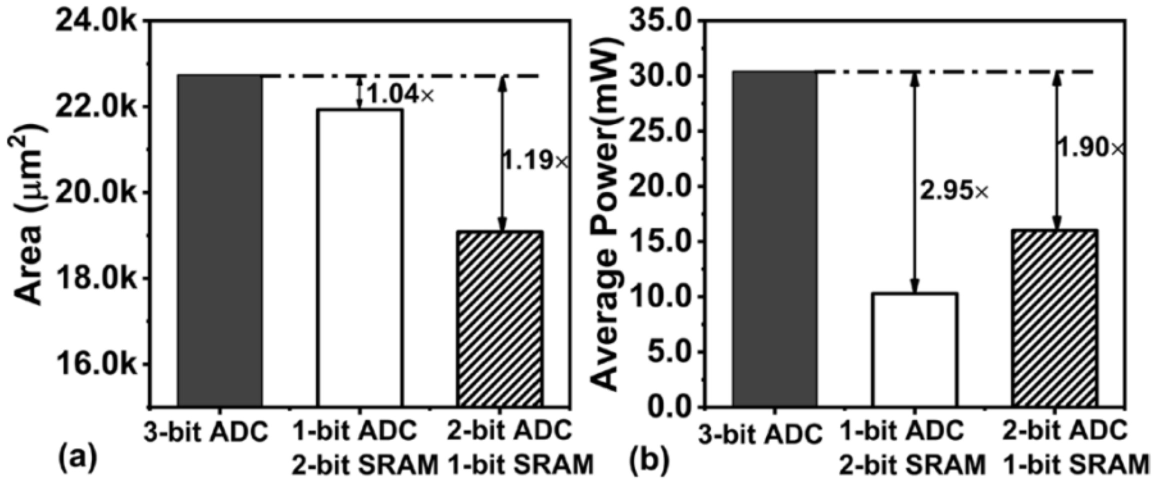


Level		0	1	2	3	4	5	6	7
RRAM State		LRS						HRS	
Average Variation (σ)	1-bit	0.1035		NA					
	2-bit	0.1035		0.2760		NA			
	3-bit	0.1035		0.2760		0.2259		0.3549	

We developed Variation-Aware Training based on a method called **Noise Injection Adaption**, where model different types of RRAM non-ideal effects as weight noise during neural network training.

D. Fan., et. al. “Noise Injection Adaption: End-to-End ReRAM Crossbar Non-ideal Effect Adaption for Neural Network Mapping”, DAC, 2019

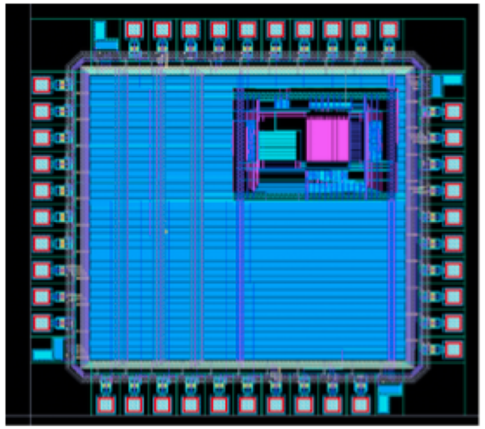
Network	RRAM Precision	ADC Precision	SRAM Precision	Accuracy (VAT)
ResNet-20	Baseline Inference Accuracy: 3-bit RRAM, 3-bit ADC, no SRAM			89.45%
	3-bit	1-bit	1-bit	84.08%
			2-bit	88.71%
			3-bit	89.06%
		2-bit	1-bit	91.15%
			2-bit	91.58%
			3-bit	91.50%



- Accurate Digital Module could improve system accuracy due to device variation
- It could also reduce the resolution of power-dominating ADC, thus reducing system power consumption

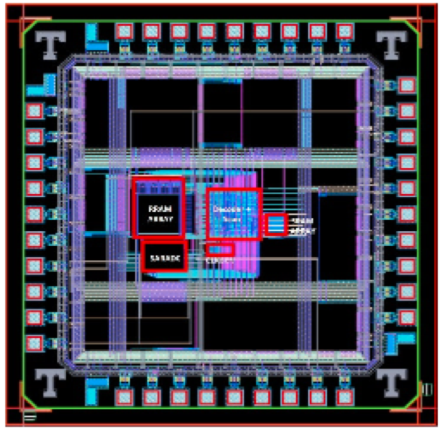
[1] G. Krishnan, et. al. “Hybrid RRAM/SRAM In-Memory Computing for Robust DNN Acceleration,” TCAD
[2] Z. Wang, et. al. “Digital-Assisted Analog In-Memory Computing with RRAM Devices”, VLSI-TSA, 2023

Multiple 65nm Hybrid CMOS/RRAM(HfO2) Chips Testing-in-Progress



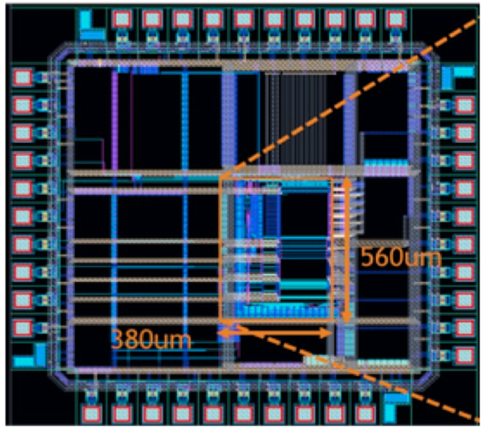
RRAM IMC array with CMOS periphery for genome sequencing alignment

ESSCIRC'23, invited to extend to **JSSC** special issue



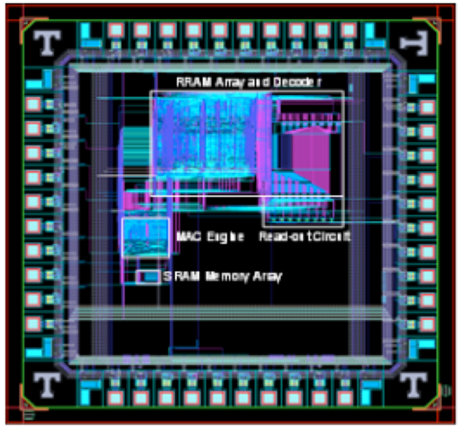
Hybrid training, with the RRAM array for MSB of back propagation, and SRAM for LSB

Chip testing in progress



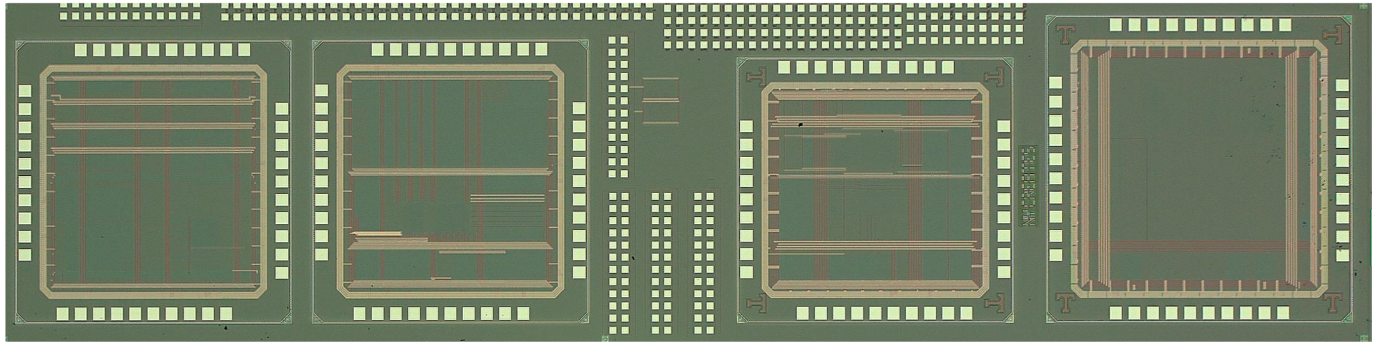
Multi-bit RRAM IMC macro with half-range shifted weight encoding

Chip testing in progress



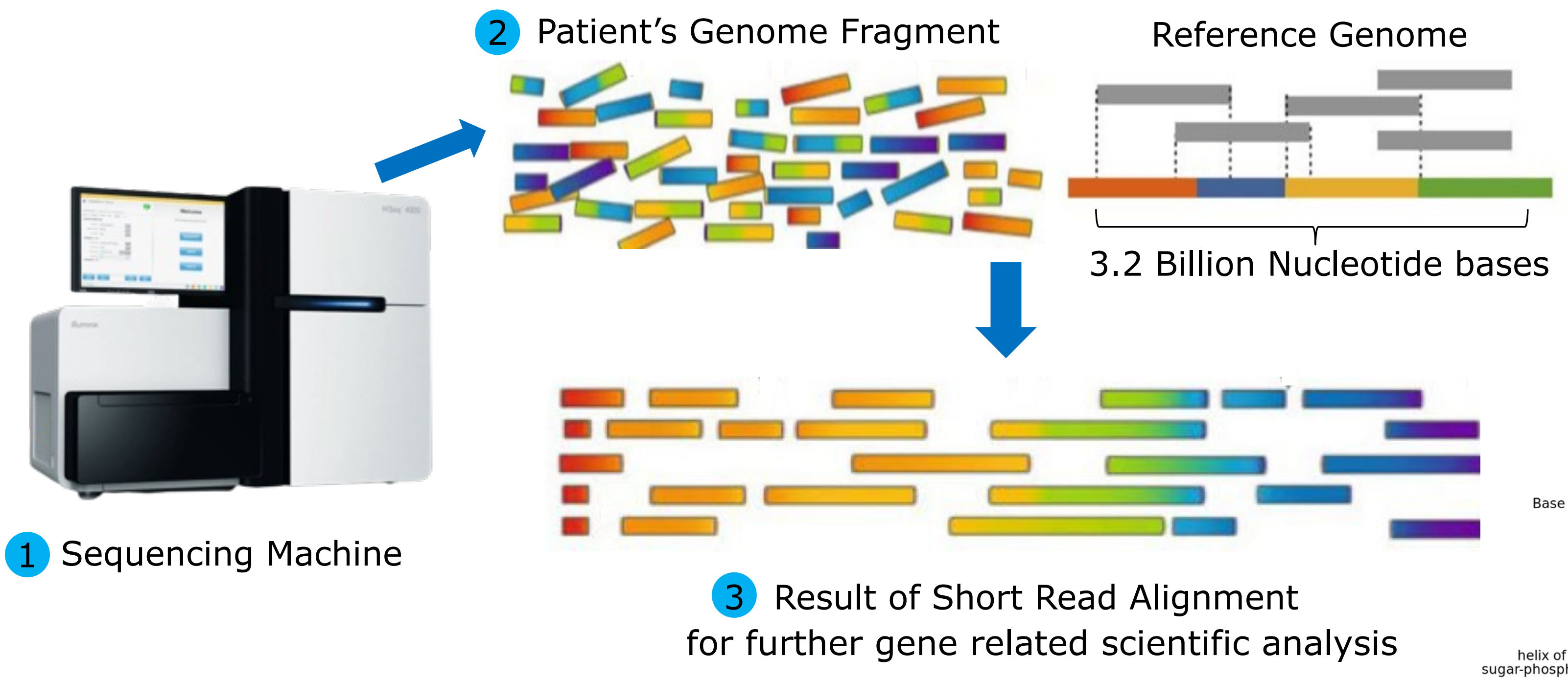
Digital (SRAM)-assisted analog RRAM IMC for robust computing under variations

TCAD, VLSI-TSA'23
More testing on going



in collaboration with Prof. Cady at SUNY, Prof. Seo at Cornell Tech and Prof. Cao at UMN

Genome Processing: Alignment is the Computing Bottleneck



2 Patient’s Genome Fragment

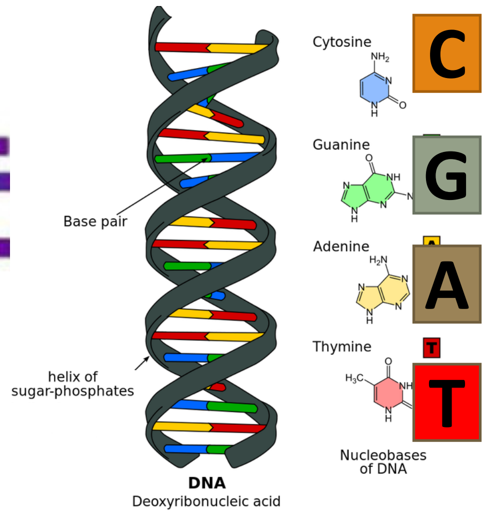
Reference Genome

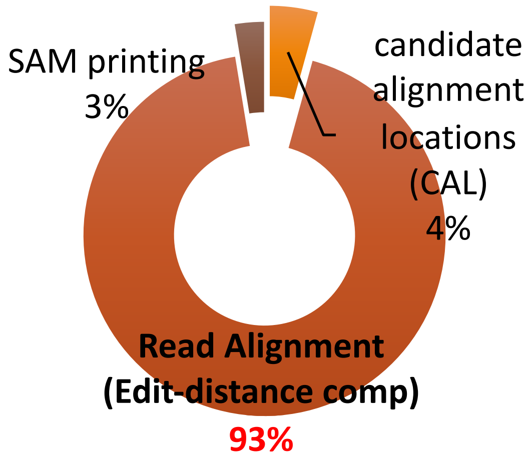
3.2 Billion Nucleotide bases

1 Sequencing Machine

3 Result of Short Read Alignment
for further gene related scientific analysis

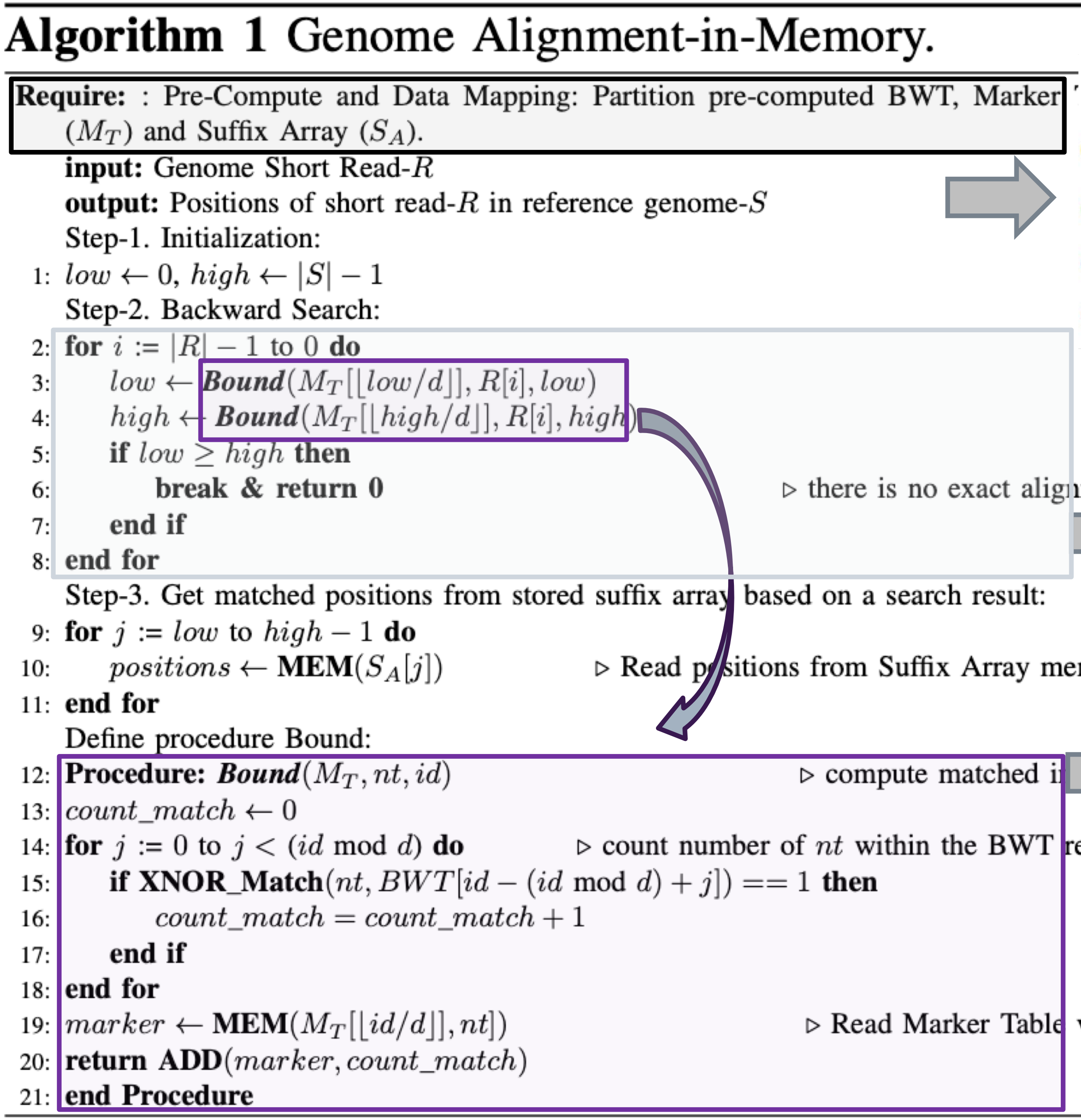
Genome Short Read Alignment becomes the major bottleneck



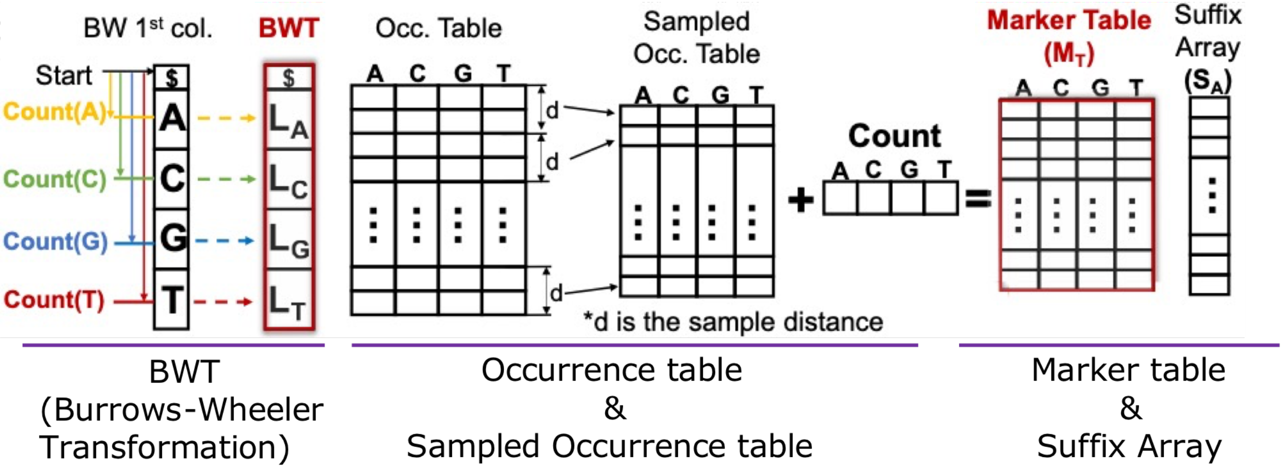


- Genome sequencing plays a pivotal role in disease diagnostics and personalized medicine strategies.
- The immense size of genome data poses challenges for GPUs/CPUs, primarily due to memory-wall constraints
- IMC stands out as a suitable option, due to the large data and simple operation.

DNA Alignment-in-Memory Algorithm



```
Algorithm 1 Genome Alignment-in-Memory.
Require: : Pre-Compute and Data Mapping: Partition pre-computed BWT, Marker
    ($M_T$) and Suffix Array ($S_A$).
    input: Genome Short Read-$R$
    output: Positions of short read-$R$ in reference genome-$S$
    Step-1. Initialization:
    $low \leftarrow 0$, $high \leftarrow |S| - 1$
    Step-2. Backward Search:
    for $i := |R| - 1$ to 0 do
        $low \leftarrow$ Bound($M_T[\lfloor low/d \rfloor], R[i], low$)
        $high \leftarrow$ Bound($M_T[\lfloor high/d \rfloor], R[i], high$)
        if $low \geq high$ then
            break & return 0                    ▷ there is no exact alignment
        end if
    end for
    Step-3. Get matched positions from stored suffix array based on a search result:
    for $j := low$ to $high - 1$ do
        $positions \leftarrow$ MEM($S_A[j]$)     ▷ Read positions from Suffix Array memory
    end for
    Define procedure Bound:
    Procedure: Bound($M_T, nt, id$)             ▷ compute matched i
    $count\_match \leftarrow 0$
    for $j := 0$ to $j < (id \bmod d)$ do       ▷ count number of $nt$ within the BWT region
        if XNOR_Match($nt, BWT[id - (id \bmod d) + j]$) == 1 then
            $count\_match = count\_match + 1$
        end if
    end for
    $marker \leftarrow$ MEM($M_T[\lfloor id/d \rfloor], nt$)   ▷ Read Marker Table value
    return ADD($marker, count\_match$)
    end Procedure
```

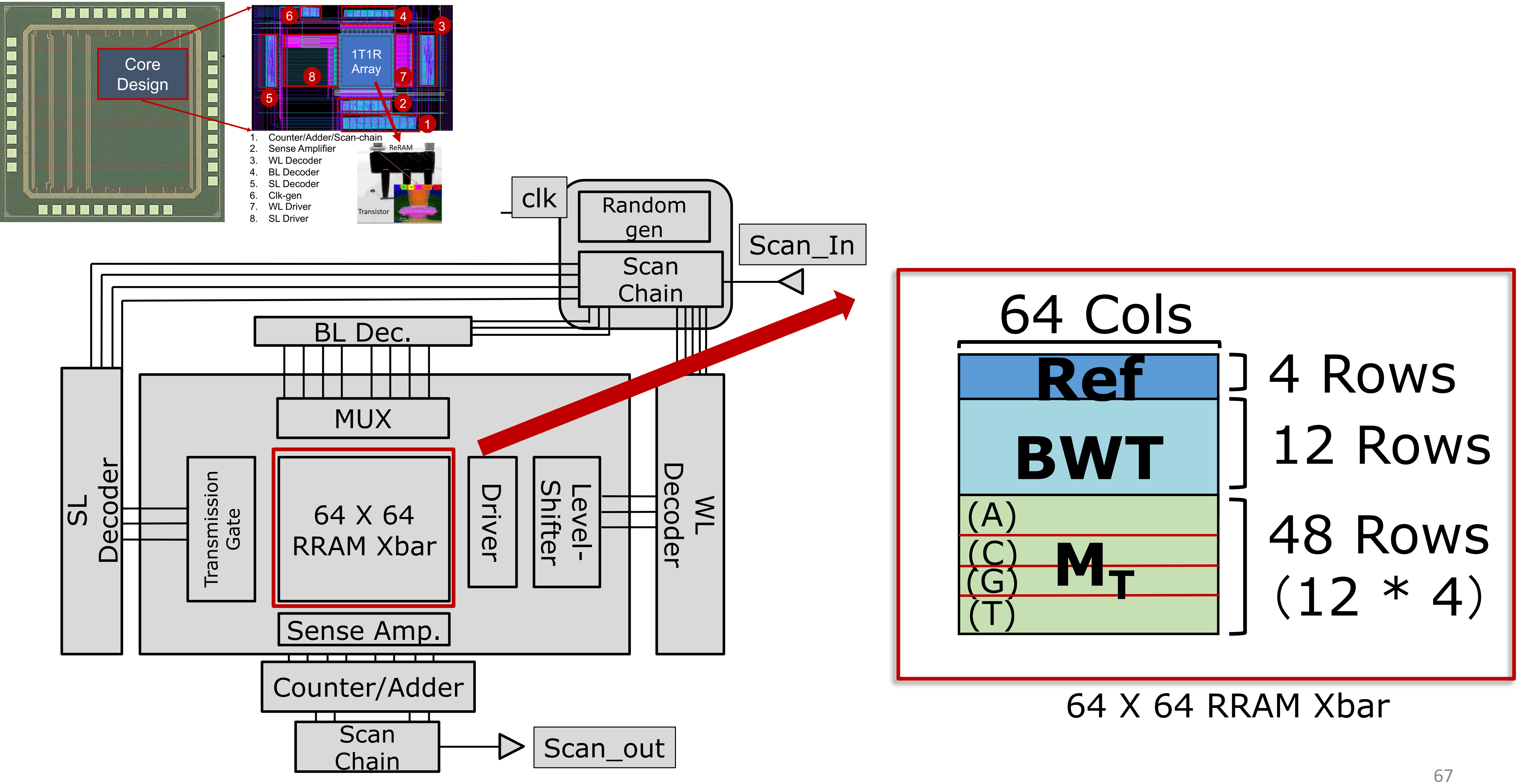


Read searching implemented through the iteratively-used ***Bound(MT, nt , id)*** procedure. To get SA interval of low and high for given intput - nt

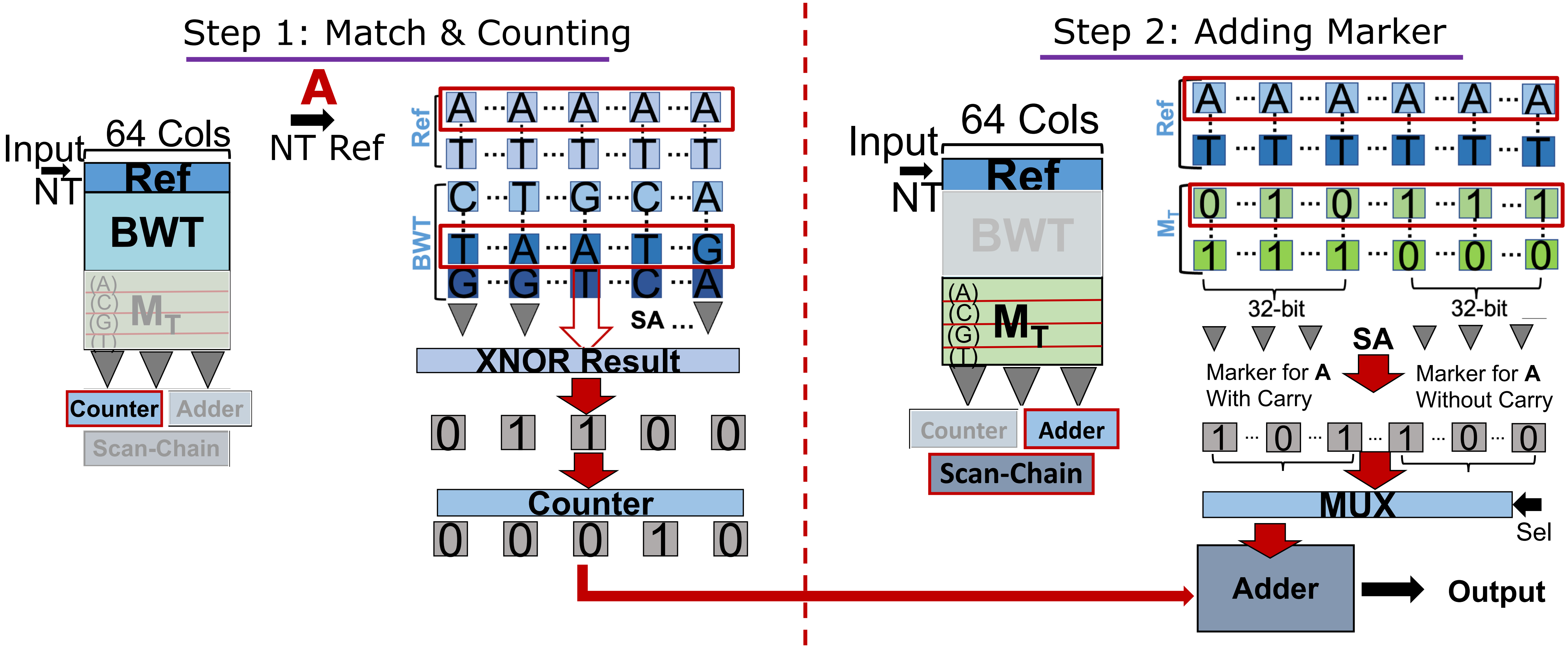
Optimized bit-wise IMC friendly logic functions *i. **XNOR_Match**, ii. **MEM**, iii. **ADD***

[1] Fan, Deliang, et al. "AlignS: A processing-in-memory accelerator for DNA short read alignment leveraging SOT-MRAM." *2019 56th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2019.

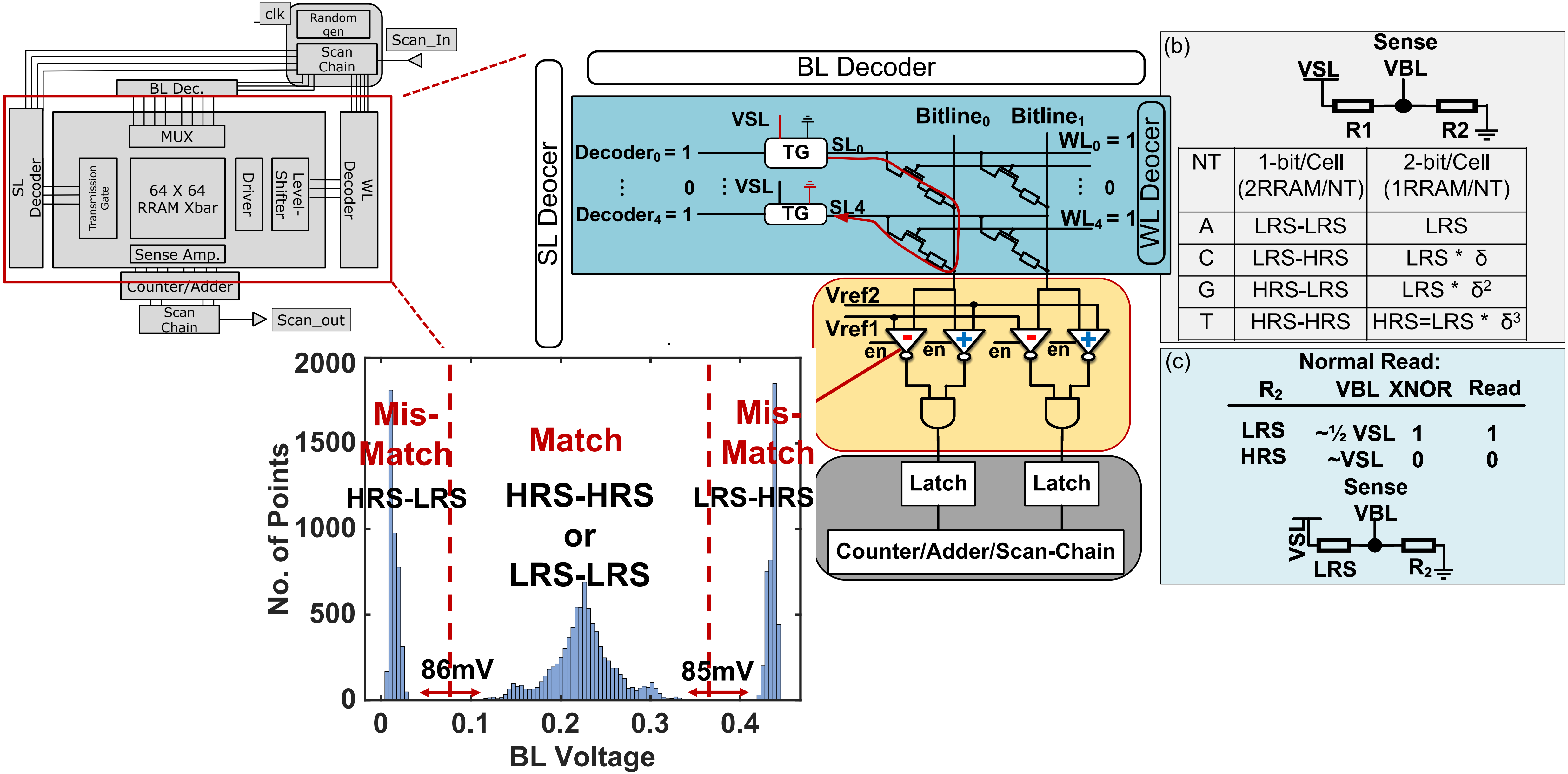
IMC Macro Block Diagram and Data Mapping



IMC Macro Design and Dataflow



IMC Macro-Circuits



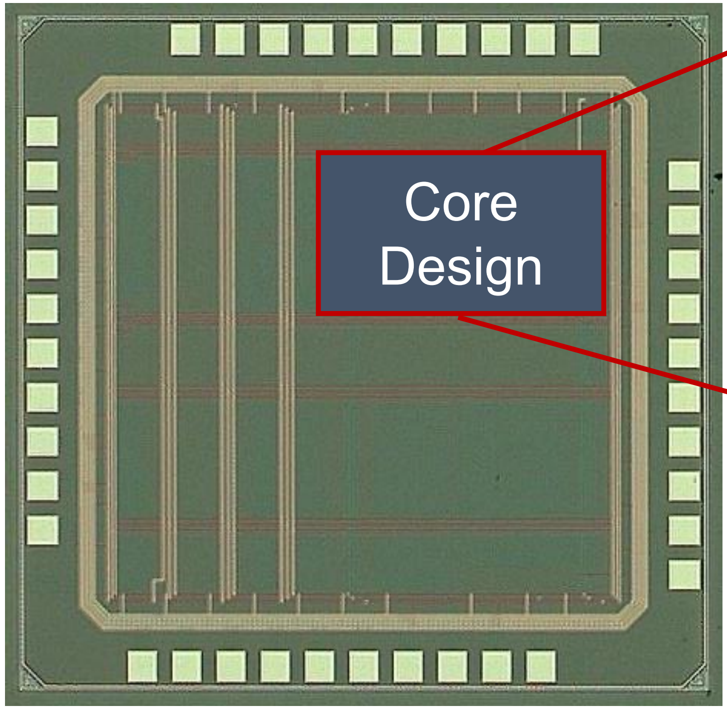
(b)

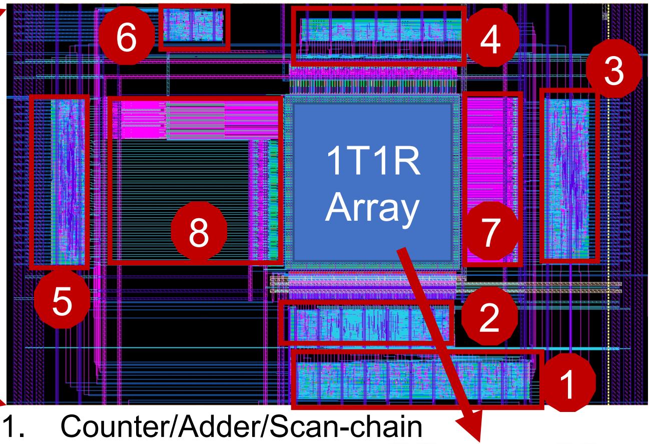
NT	1-bit/Cell (2RRAM/NT)	2-bit/Cell (1RRAM/NT)
A	LRS-LRS	LRS
C	LRS-HRS	LRS * δ
G	HRS-LRS	LRS * δ^2
T	HRS-HRS	HRS=LRS * δ^3

(c) Normal Read:

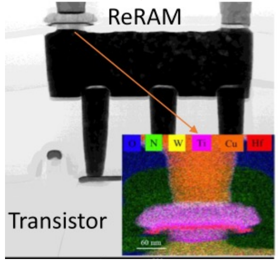
R_2	VBL	XNOR	Read
LRS	~½ VSL	1	1
HRS	~VSL	0	0

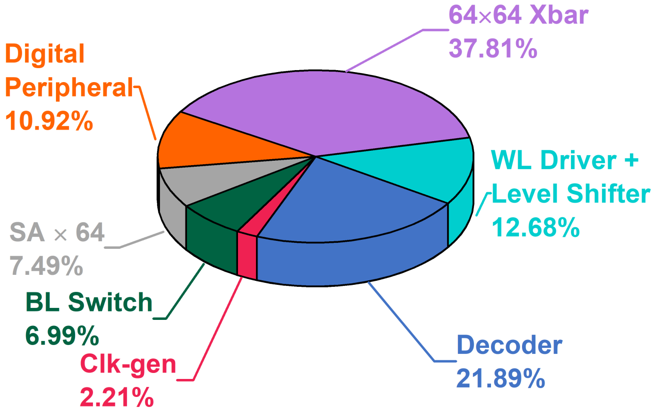
Prototype Chip Photo & Area Breakdown

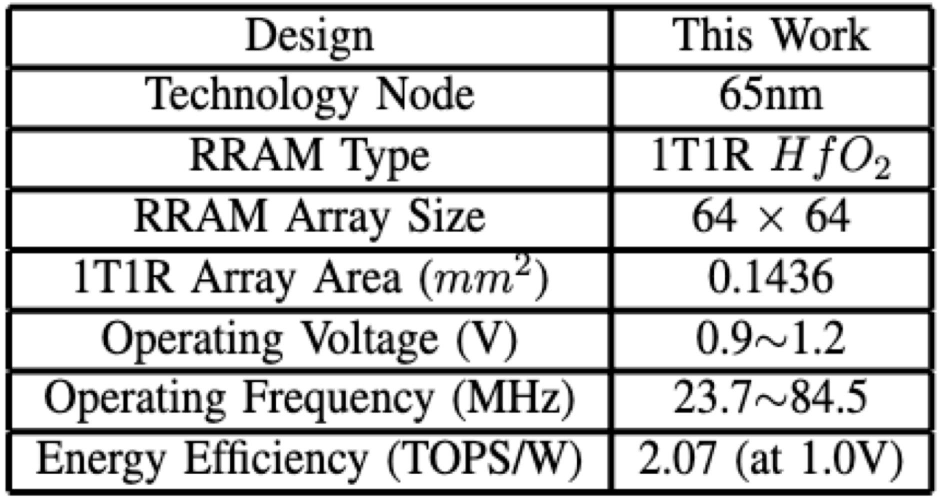




1. Counter/Adder/Scan-chain
2. Sense Amplifier
3. WL Decoder
4. BL Decoder
5. SL Decoder
6. Clk-gen
7. WL Driver
8. SL Driver



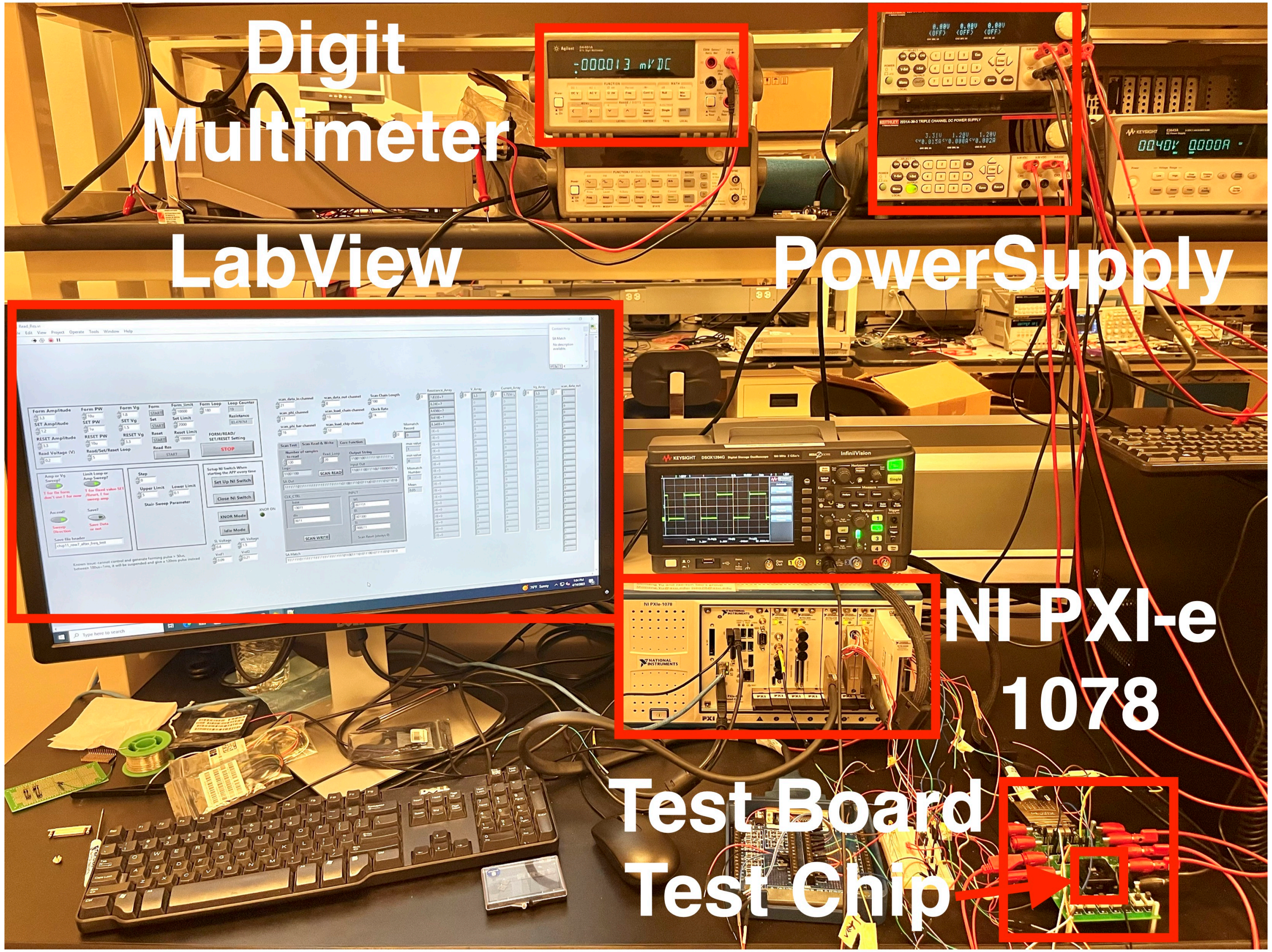




Design	This Work
Technology Node	65nm
RRAM Type	1T1R HfO_2
RRAM Array Size	64 × 64
1T1R Array Area (mm^2)	0.1436
Operating Voltage (V)	0.9~1.2
Operating Frequency (MHz)	23.7~84.5
Energy Efficiency (TOPS/W)	2.07 (at 1.0V)

- It is Pad-limited, occupying 5.175mm^2.
- The core design occupies 0.402mm^2.

Experimental Test Environment

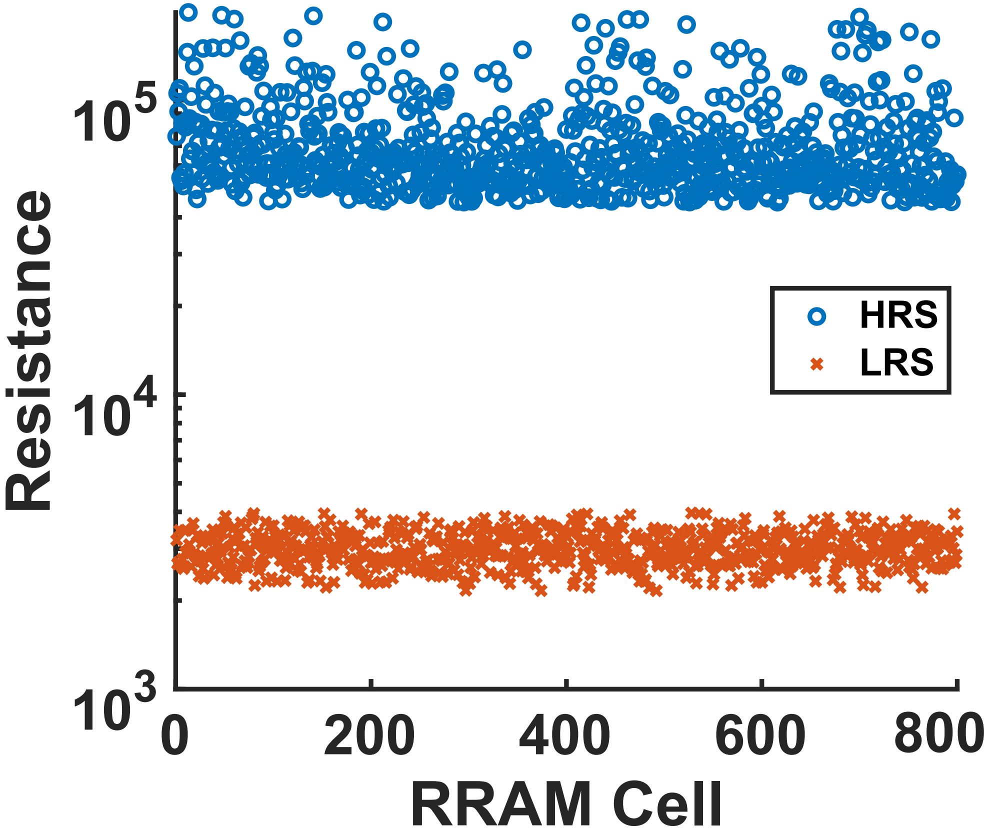


- Chip fabrication is collaborated with SUNY Polytechnic.

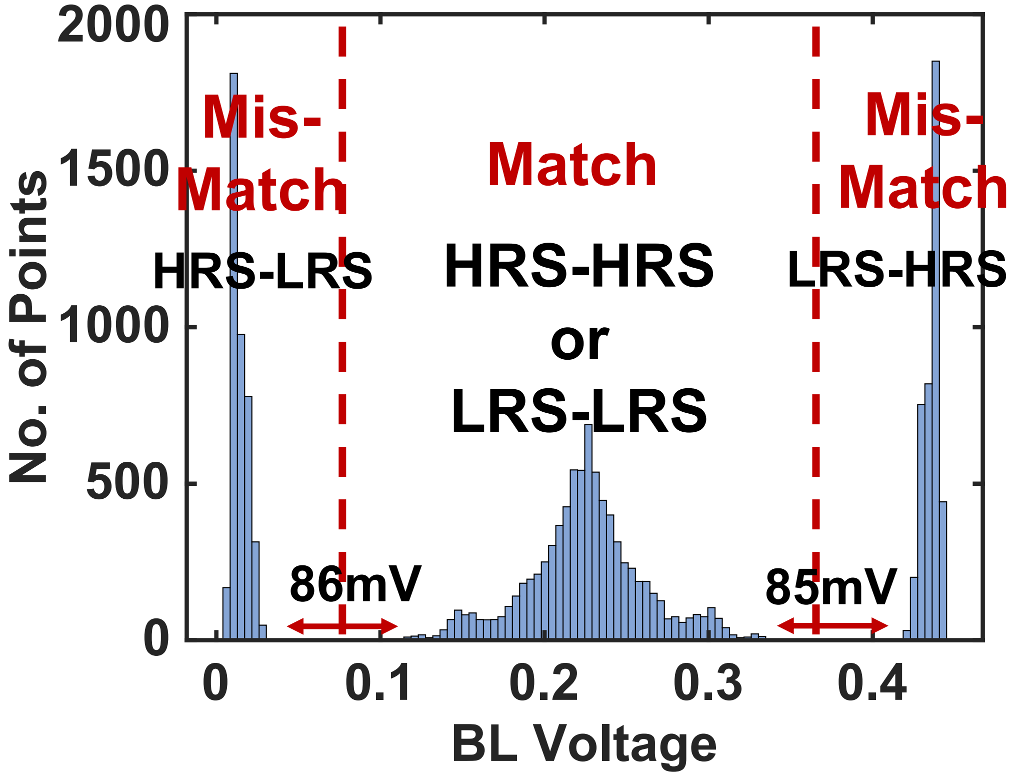
- Testing is performed on NI PXI-e 1078 with customized LabVIEW program.

Measurement Result

- Forming:
  V_{form} = 3.8V
  Pulse = 10us
  V_{WL} = 1.8V
- Set:
  V_{set} = 1.2V
  Pulse = 1us
  V_{WL} = 1.5V
- Reset:
  V_{reset} = 3.3V
  Pulse = 100ns
  V_{WL} = 3.3V

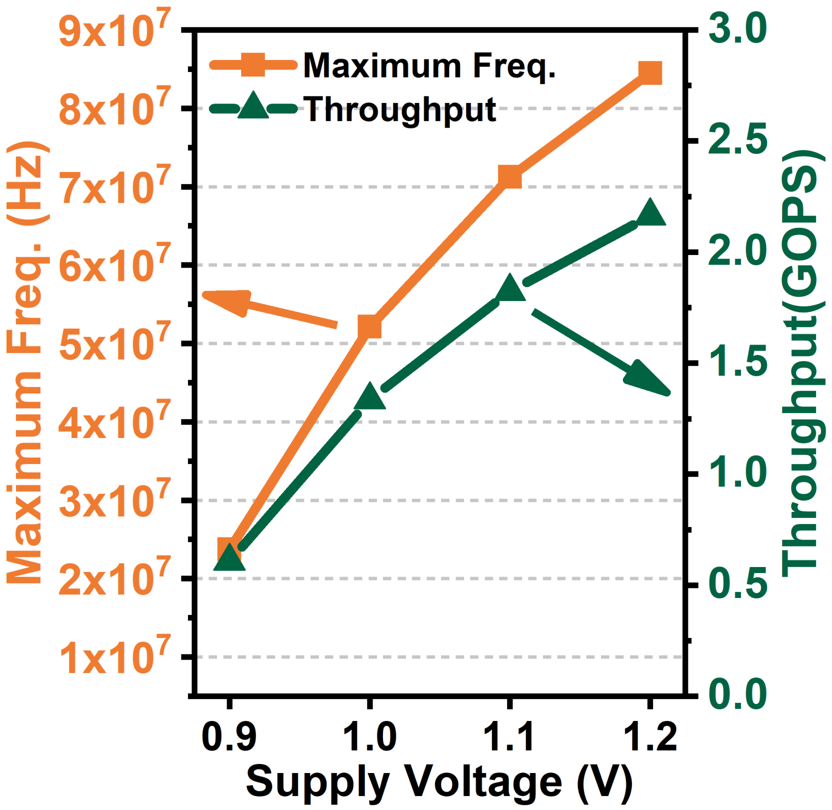


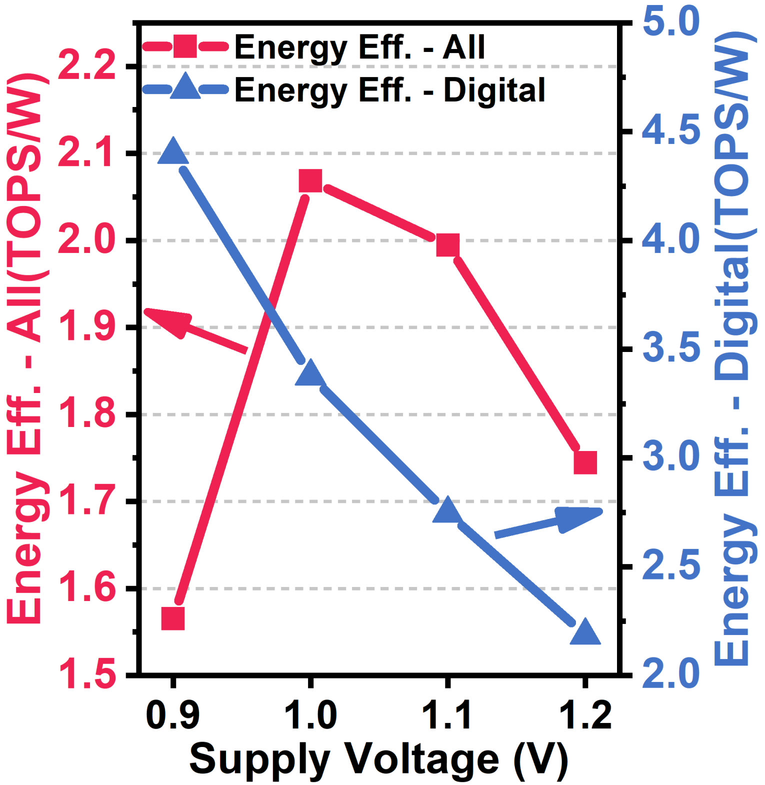
Measured Cell Resistances



BL Voltage for XNOR

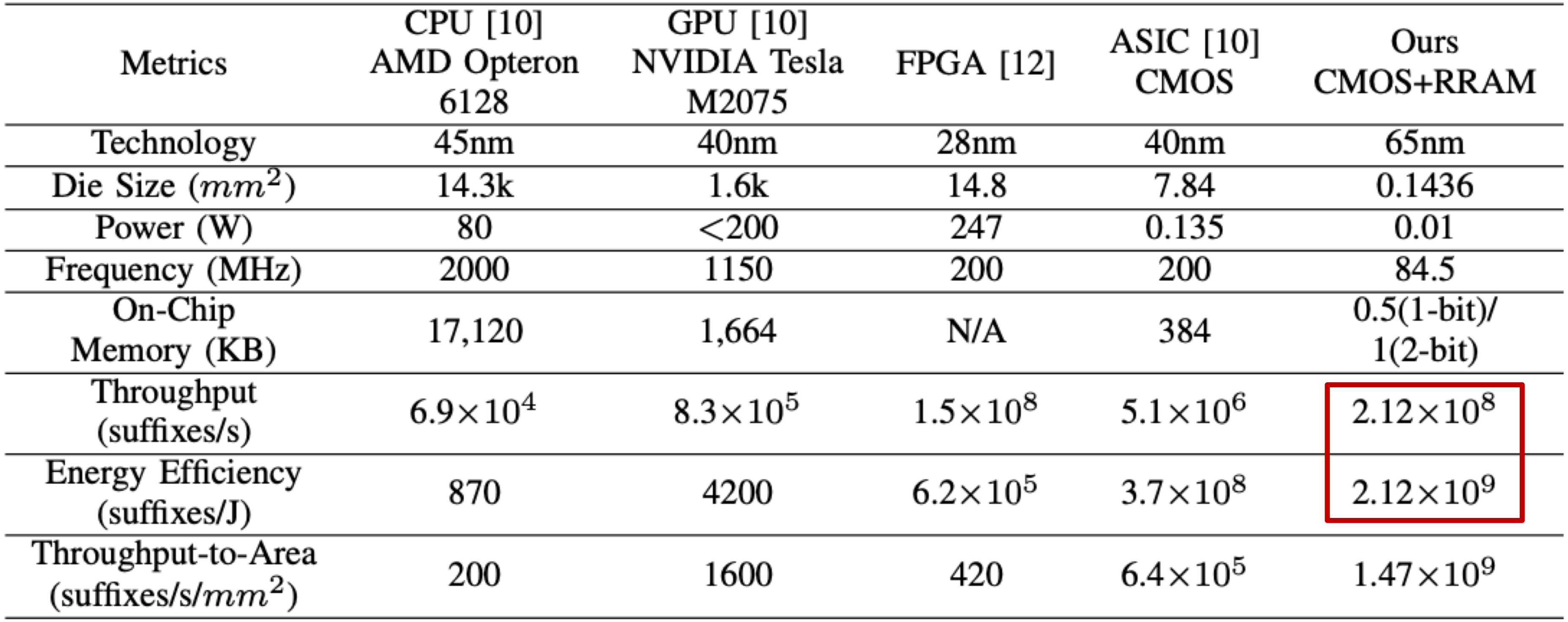
Freq., Throughput, and Energy





- Max energy efficiency: 2.07 TOPS/W @ 1.0V
- Max Freq: 84.5 MHz
- Max throughput 2.16 GOPS

Comparison with Prior Works

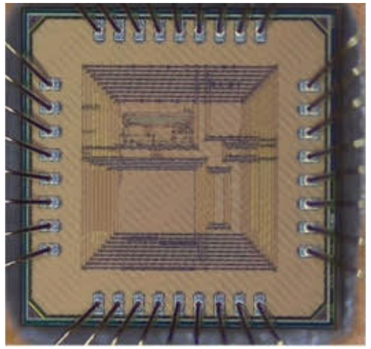


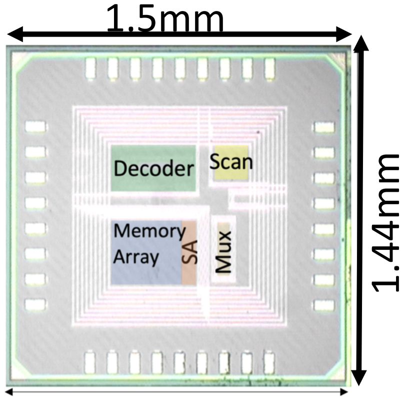
Metrics	CPU [10] AMD Opteron 6128	GPU [10] NVIDIA Tesla M2075	FPGA [12]	ASIC [10] CMOS	Ours CMOS+RRAM
Technology	45nm	40nm	28nm	40nm	65nm
Die Size (mm^2)	14.3k	1.6k	14.8	7.84	0.1436
Power (W)	80	<200	247	0.135	0.01
Frequency (MHz)	2000	1150	200	200	84.5
On-Chip Memory (KB)	17,120	1,664	N/A	384	0.5(1-bit)/ 1(2-bit)
Throughput (suffixes/s)	6.9×10^4	8.3×10^5	1.5×10^8	5.1×10^6	2.12×10^8
Energy Efficiency (suffixes/J)	870	4200	6.2×10^5	3.7×10^8	2.12×10^9
Throughput-to-Area (suffixes/s/mm^2)	200	1600	420	6.4×10^5	1.47×10^9

- ’Memory-Wall’ limits CPU/GPU’s throughput and energy efficiency.
- FPGA[12] has higher Perf. than CPU/GPU due to larger scale (8 FPGAs in the design).
- ASIC[10] significantly improved energy efficiency.
- Our first RRAM-based IMC macro achieves best efficiency with **2.07 TOPS/W and 2.12G suffixes/J at 1.0V**

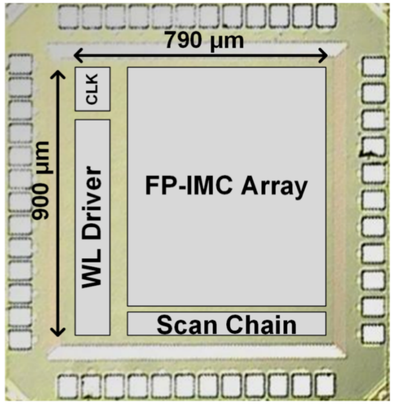
Our Developed IMC Chip Prototypes (examples): SRAM and NVM

- A 1.23-GHz 16-kb **Programmable and Generic Processing-in-SRAM** Accelerator in **TSMC 65nm**
- Published in ESSCIRC’2022

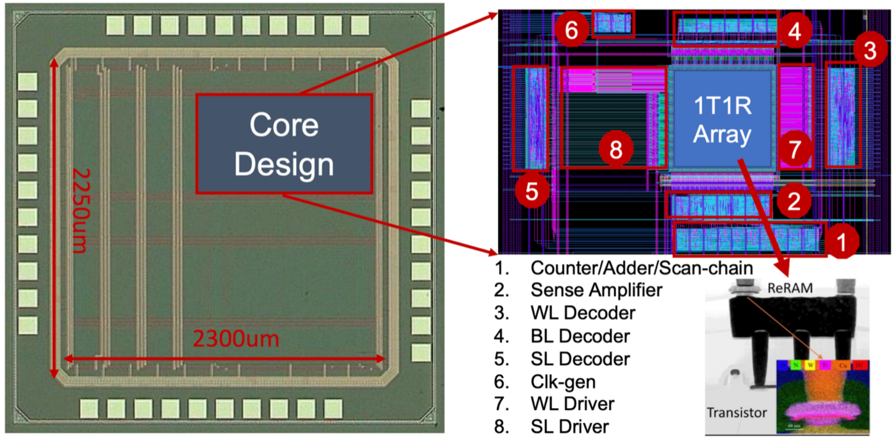




- All-Digital **Configurable Floating-Point** In-Memory Computing Macro in **TSMC 28nm**
- Published in ESSCIRC’2023



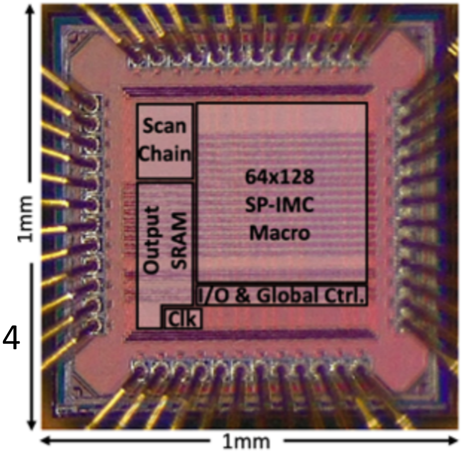
- **hybrid 65nm CMOS-ReRAM (HfO2)** designs for **DNN** acceleration and **DNA genome sequence alignment** applications.
- 64 by 64 HfO2 RRAM and SRAM module for hybrid computing



1. Counter/Adder/Scan-chain
2. Sense Amplifier
3. WL Decoder
4. BL Decoder
5. SL Decoder
6. Clk-gen
7. WL Driver
8. SL Driver

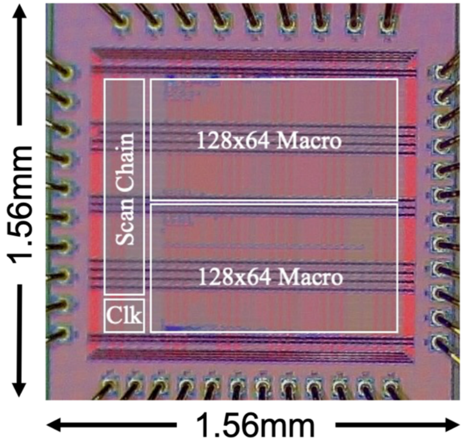
ESSCIRC’23, invited to extend to **JSSC** special issue

- **28 nm** IMC chip prototype for **sparse in-memory matrix convolution**
- Supports N:M structured sparsity, run length encoding, compressed sparsity column
  - Sparse NN
  - Wireless Decoder (Neural BP)



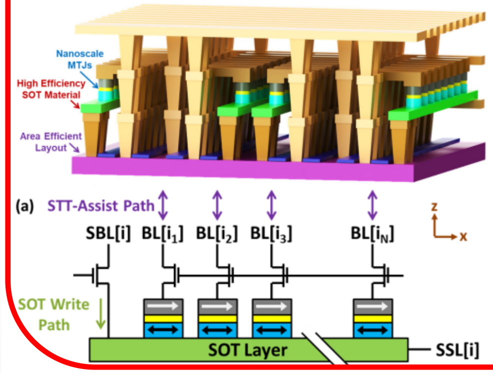
CICC’24

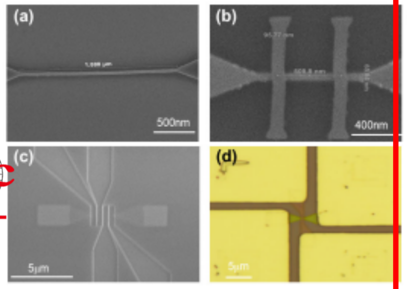
- A **28nm** 2385.7 TOPS/W/b **Precision Scalable** In-Memory Computing Macro with **Bit-Parallel** Inputs and Decomposable Weights for DNNs



SSCL

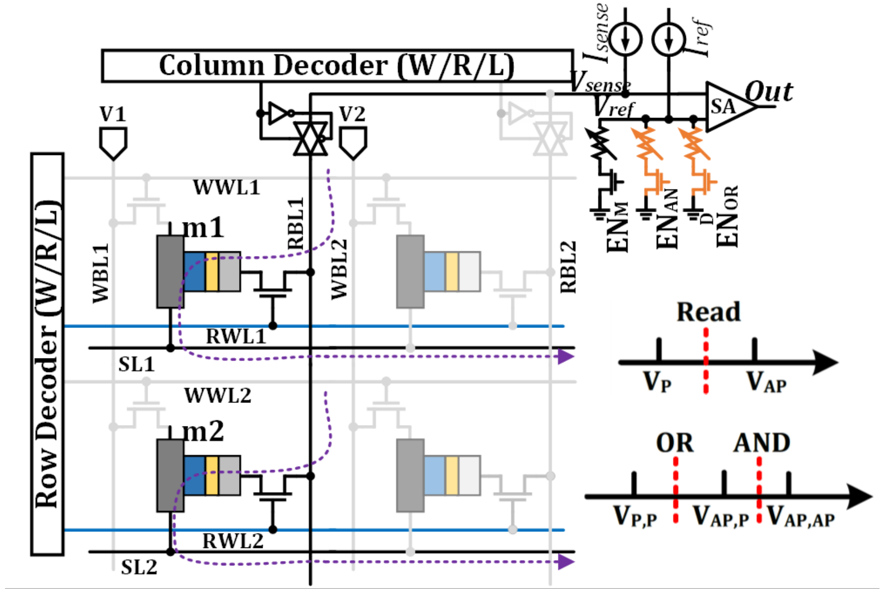
- energy efficient IMC with high-density, field-free **STT-assisted SOT-MRAM** (SAS-MRAM), ~**100nm, NSF FuSe/ACED projects**

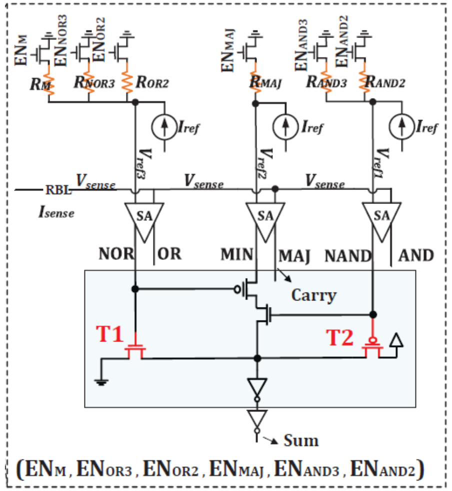


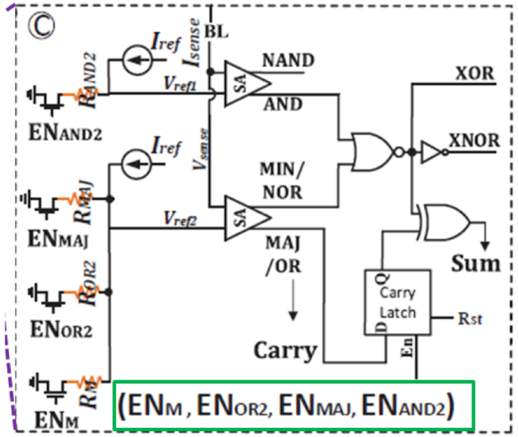


W. Hwang, D. Fan, S. X. Wang, et., al., TMAG, 2022

MRAM based In-memory logic circuit designs







Basic In-Memory logic – AND/NAND, OR/NOR, (ASPDAC 2018)

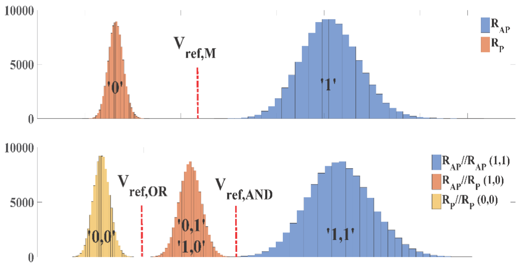
More logic function supported: Reconfigurable AND/NAND, OR/NOR, XOR/XNOR, Majority In-Memory Logic in one design (DAC 2018, ICCAD 2018, ISVLSI 2018, TMAG2018, TCAD 2018)

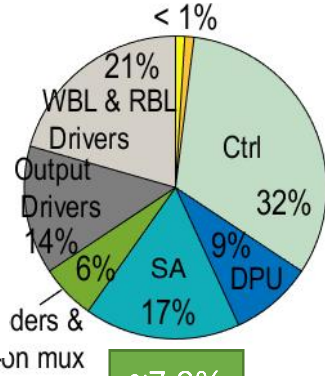
Two-Cycle in-memory full adder: AND/NAND, OR/NOR, XOR/XNOR, Majority, Full adder, Adder/multiplier (DAC 2019, ASPDAC 2019)

One-Cycle in-memory full adder: AND/NAND, OR/NOR, XOR/XNOR, Majority, Full adder, more efficient adder/multiplier (DATE 2019/ICCAD 2019/ DAC2021)

Supported ISA

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





~7.9%

Area Overhead: 1-cycle in-memory FA

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

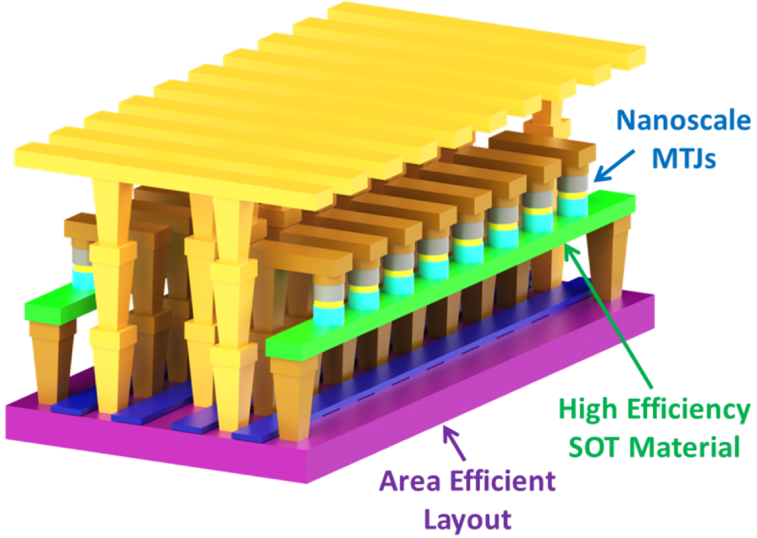
PI of projects funded by SRC nCORE, NSF FuSe, ACED, SHF, FET, etc.

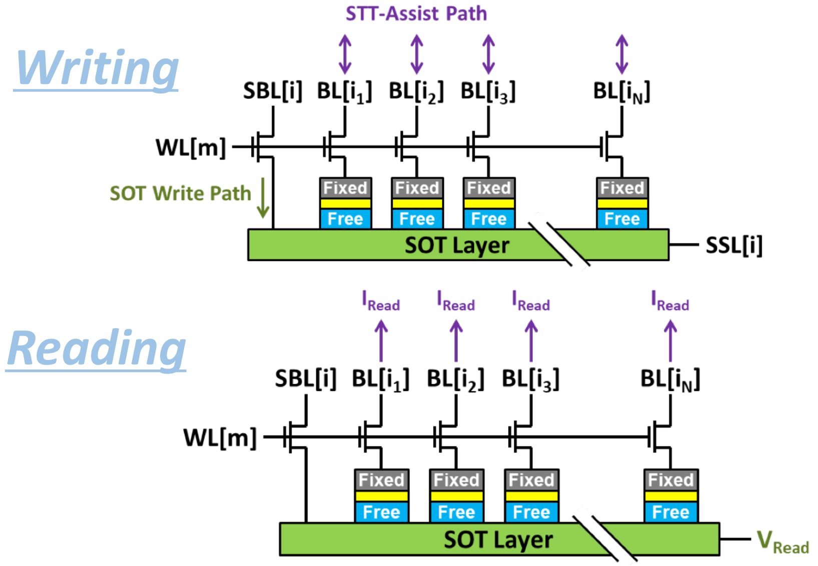
Best Paper Award: ISVLSI’17, GLSVLSI’19

Best Paper Candidate Nomination: DAC’21, DATE’22, DAC’22

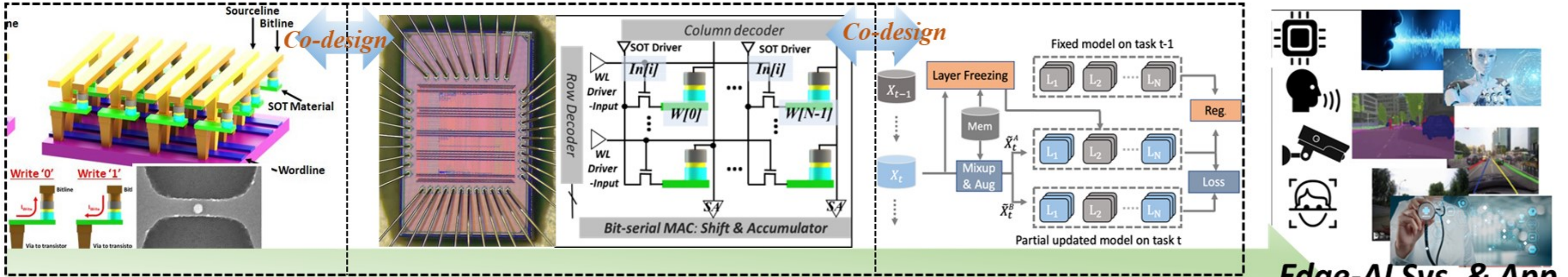
High Density STT-Assisted SOT-MRAM (SAS-MRAM)

- **Achieves MRAM Design Goals:**
  - ✓ High Speed Switching (~1 ns)
  - ✓ High Density (~1T1MTJ)
  - ✓ Minimal current through tunnel barrier
- **SAS-MRAM Device:**
  - Multiple MTJs per SOT line
  - Amortize SOT driver area &power





***IMC microelectronic chip for Situation-Aware Edge-AI,** newly awarded **NSF FuSe (Future of Semiconductor, part of CHIPS Act)** and ACED projects*



Thrust-1:
2T-SOT-MRAM device & fab.
Stanford

Thrust-2:
AI-PIM chip design & tapeout
Hopkins

Thrust-3:
AI algorithm co-design
Duke

Edge-AI Sys. & App.
100X energy efficiency

In collaboration with Dr. Shan. X. Wang from Stanford University, preliminary results published in collaborative work: W. Hwang et al. "Energy Efficient Computing with High-Density, Field-Free STT-Assisted SOT-MRAM (SAS-MRAM),"TMAG 2022

SAS-MRAM Device Prototype

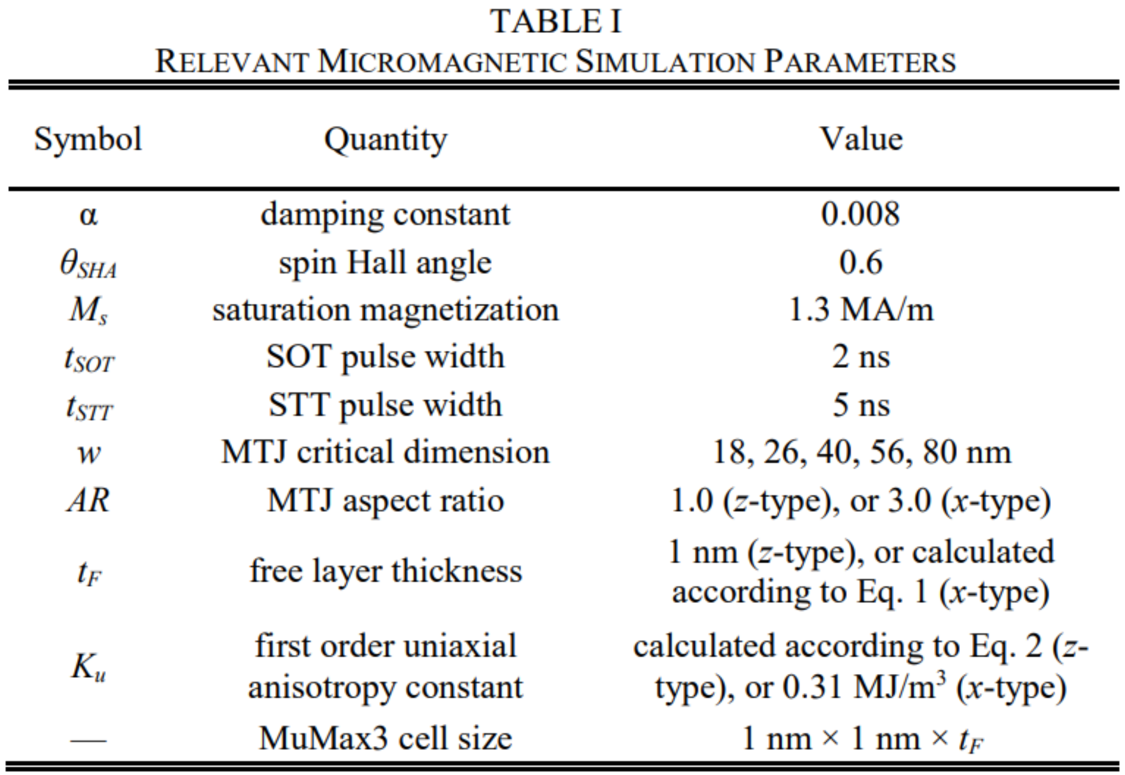


TABLE I
RELEVANT MICROMAGNETIC SIMULATION PARAMETERS

Symbol	Quantity	Value
α	damping constant	0.008
θ_{SHA}	spin Hall angle	0.6
M_s	saturation magnetization	1.3 MA/m
t_{SOT}	SOT pulse width	2 ns
t_{STT}	STT pulse width	5 ns
w	MTJ critical dimension	18, 26, 40, 56, 80 nm
AR	MTJ aspect ratio	1.0 (*z*-type), or 3.0 (*x*-type)
t_F	free layer thickness	1 nm (*z*-type), or calculated according to Eq. 1 (*x*-type)
K_u	first order uniaxial anisotropy constant	calculated according to Eq. 2 (*z*-type), or 0.31 MJ/m^3 (*x*-type)
—	MuMax3 cell size	1 nm × 1 nm × t_F

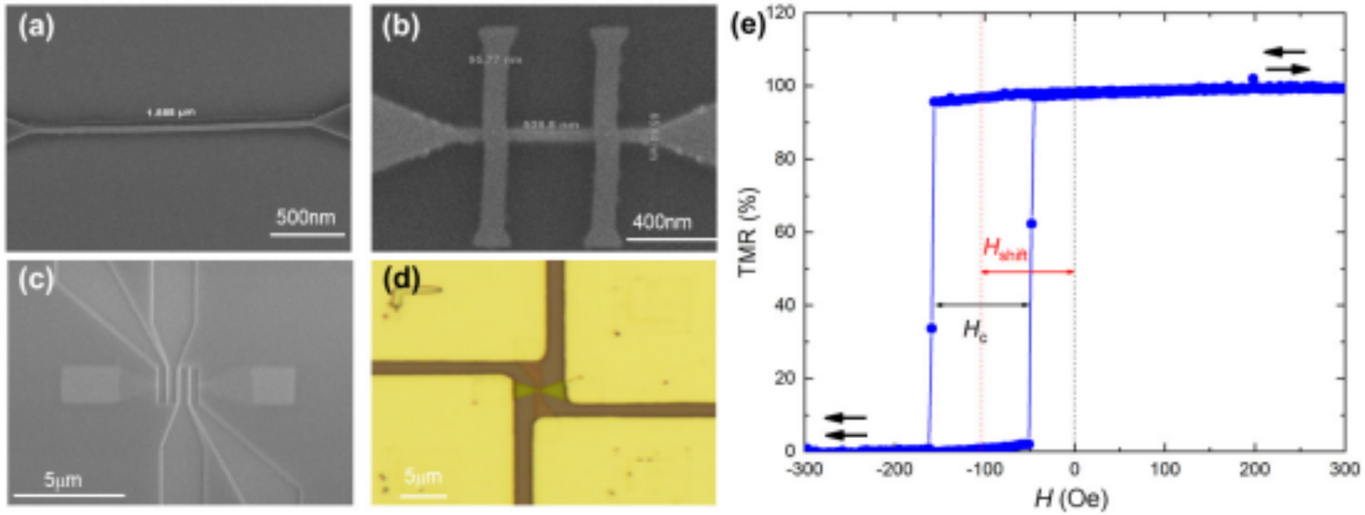
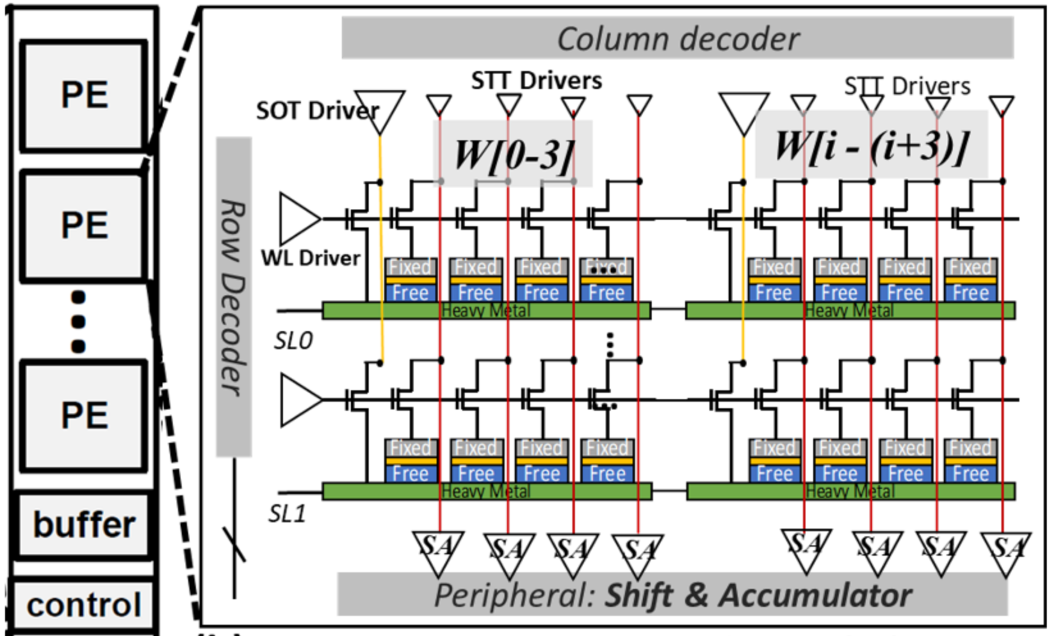


Fig. 8. SEM images of (a) SOT line patterning (mask1), (b) self-aligned double-MTJ cells (mask2) and (c) top electrodes and SOT ohmic contacts of quad-MTJ cells (mask3). (d) Optical micrograph of the SAS-MRAM device. (e) Field-dependent TMR curve of a 100 nm × 50 nm MTJ cell.

In collaboration with Dr. Shan. X. Wang from Stanford University, preliminary results published in collaborative work: W. Hwang et al. “Energy Efficient Computing with High-Density, Field-Free STT-Assisted SOT-MRAM (SAS-MRAM),”TMAG 2022

In-MRAM MAC for AI On-Chip Inference and Learning



(b) ***In-Memory-MAC Processing Element (PE)***

Digital bit-serial in-MRAM MAC operations

Bit-serial input sent to the read transistor gate

1-Bit Multiplication (AND) in memory

***Example**: 4b input(a) x 4b weight(w)*
4 bit input broadcast to WL in bit-serial
4b weight bits stored in 4 MTJs
*Read **'AND'** logic in **'SA'**, then **shift&accu.***

1 0 1 1 (Weight)
x 1 1 0 1 (Input)

Cycle 1: W x 1 1 0 1 1
Cycle 2: W x 1 1 0 1 1
Cycle 3: W x 0 0 0 0 0
Cycle 4: W x 1 1 0 1 1
AND
Shift & Acc: 1 0 0 0 1 1 1 1

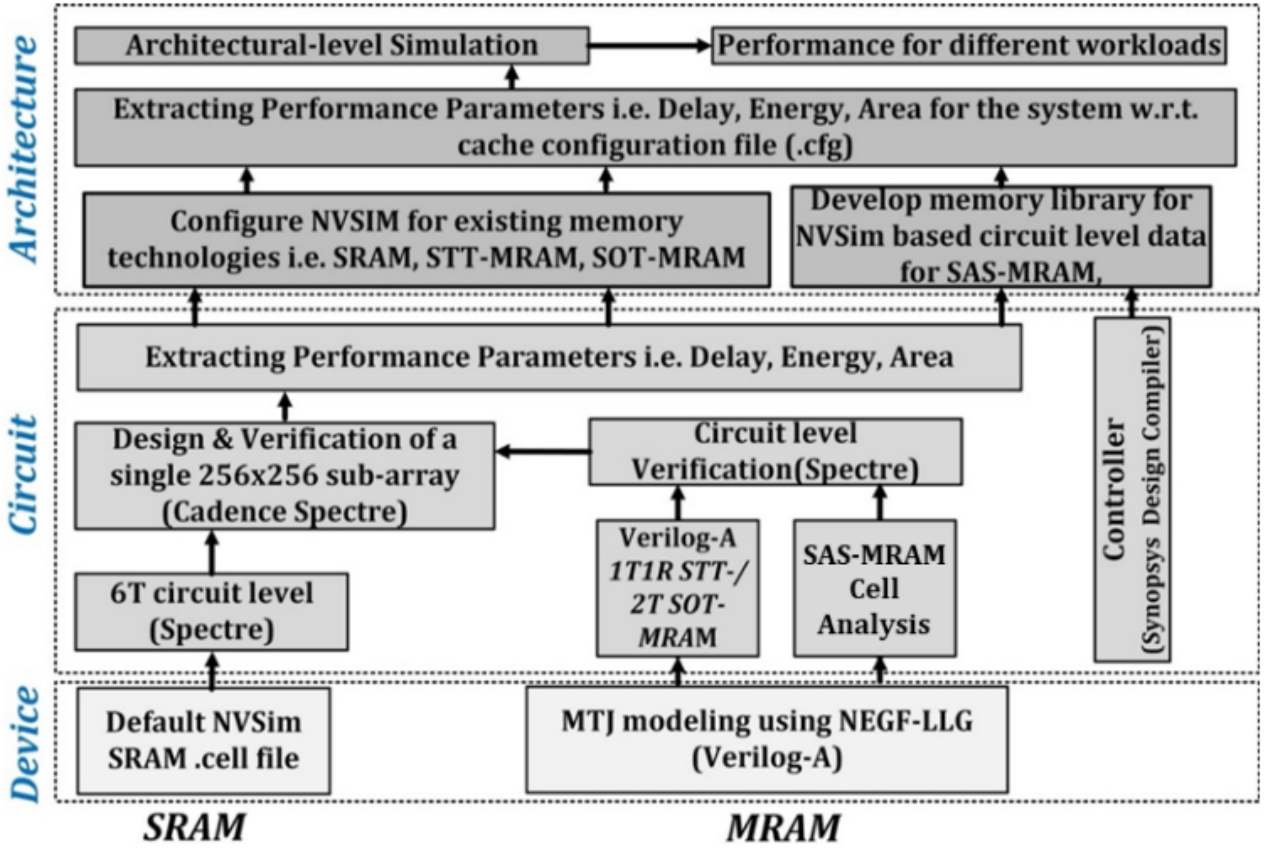

Architecture
Architectural-level Simulation
Performance for different workloads
Extracting Performance Parameters i.e. Delay, Energy, Area for the system w.r.t. cache configuration file (.cfg)
Configure NVSIM for existing memory technologies i.e. SRAM, STT-MRAM, SOT-MRAM
Develop memory library for NVSim based circuit level data for SAS-MRAM,
Circuit
Extracting Performance Parameters i.e. Delay, Energy, Area
Design & Verification of a single 256x256 sub-array (Cadence Spectre)
Circuit level Verification(Spectre)
6T circuit level (Spectre)
Verilog-A 1T1R STT-/ 2T SOT-MRAM
SAS-MRAM Cell Analysis
Controller (Synopsys Design Compiler)
Device
Default NVSim SRAM .cell file
MTJ modeling using NEGF-LLG (Verilog-A)
SRAM
MRAM

Fig. 5. A graphical representation of the bottom-up evaluation framework which was used to evaluate the array-level performance in this work.

Dr. Fan's group developed cross-layer in-memory computing simulator: PIMA-SIM, Open-sourced in github: https://github.com/ASU-ESIC-FAN-Lab/PIMA-SIM

D. Fan, et. al., ""On-Device Continual Learning with STT-Assisted-SOT MRAM based In-Memory Computing," " TCAD 2024

Energy Savings of SAS-MRAM over SRAM for On-Chip Learning

- Cost of Performing One ResNet-18 Weight Update (8-bit)
  - Training from scratch requires ~421,875 weight updates

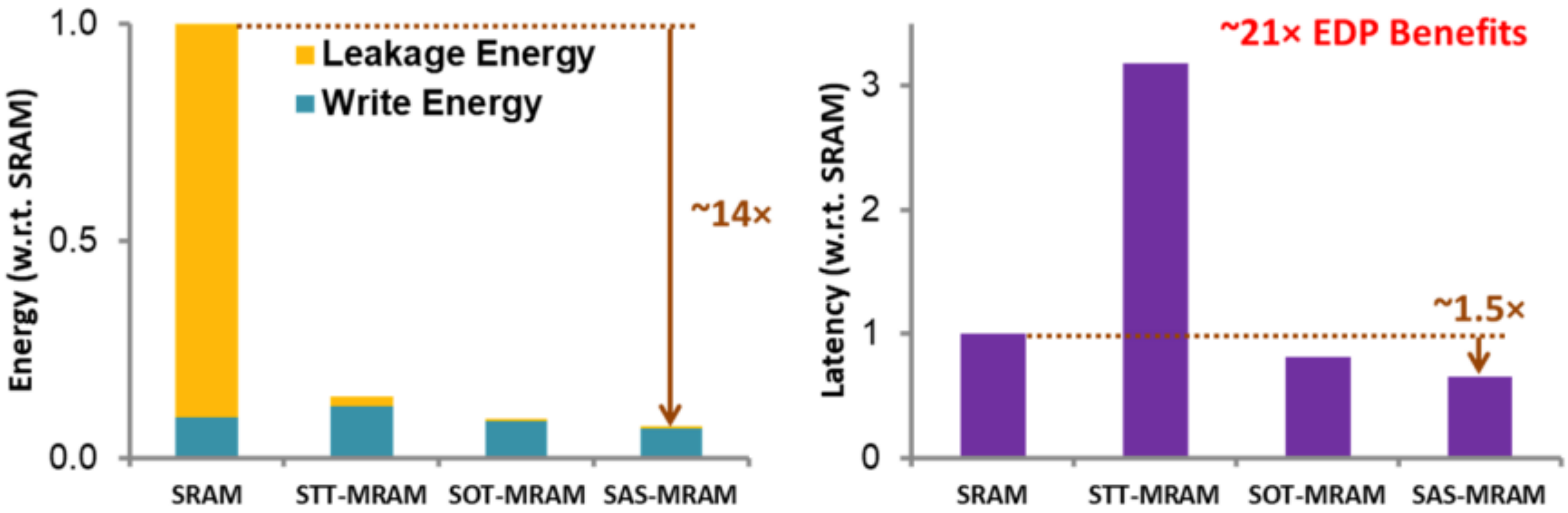
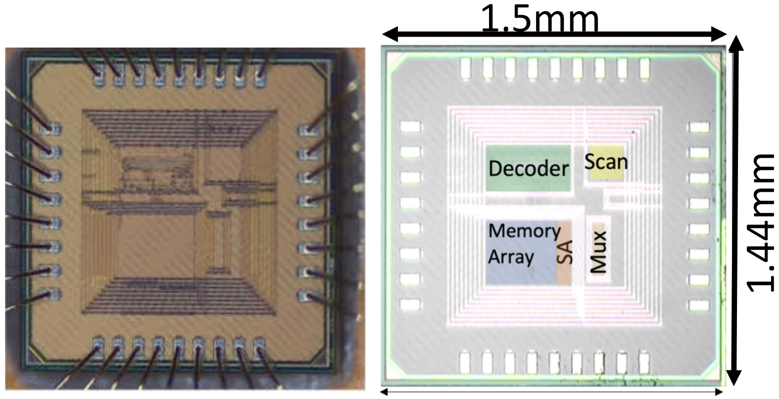


Fig. 6. Energy and latency required for one epoch of ResNet-18 weight update for various MRAM technologies vs. SRAM. SAS-MRAM shows ~21× EDP benefits with respect to SRAM when evaluated using the NCSU 45nm PDK.

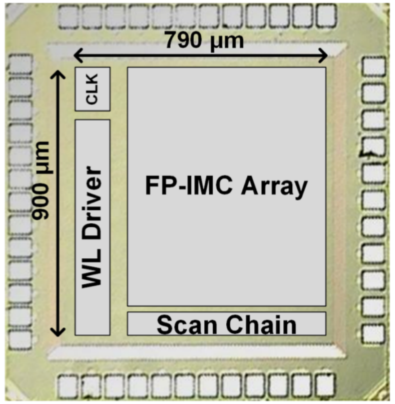
"Energy Efficient Computing with High-Density, Field-Free STT-Assisted SOT-MRAM (SAS-MRAM),"TMAG 2022

Summary: Our Developed IMC Chip Prototypes: SRAM and NVM

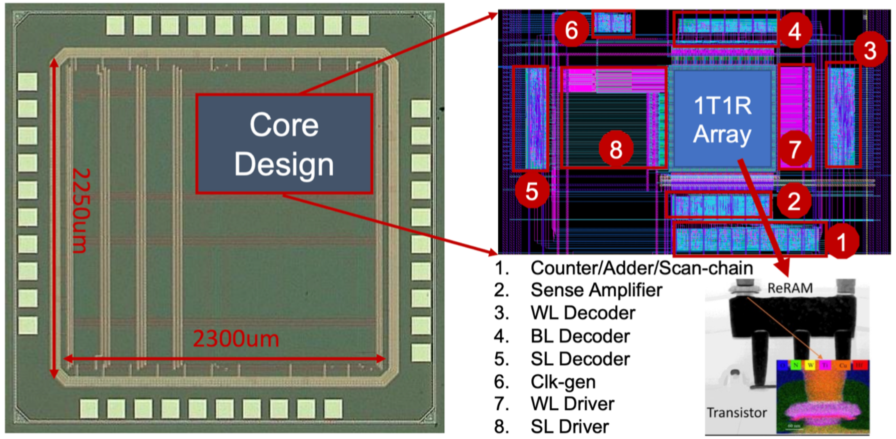
- A 1.23-GHz 16-kb **Programmable and Generic Processing-in-SRAM** Accelerator in **TSMC 65nm**
- Published in ESSCIRC’2022



- All-Digital **Configurable Floating-Point** In-Memory Computing Macro in **TSMC 28nm**
- Published in ESSCIRC’2023



- **hybrid 65nm CMOS-ReRAM (HfO2)** designs for **DNN** acceleration and **DNA genome sequence alignment** applications.
- 64 by 64 HfO2 RRAM and SRAM module for hybrid computing

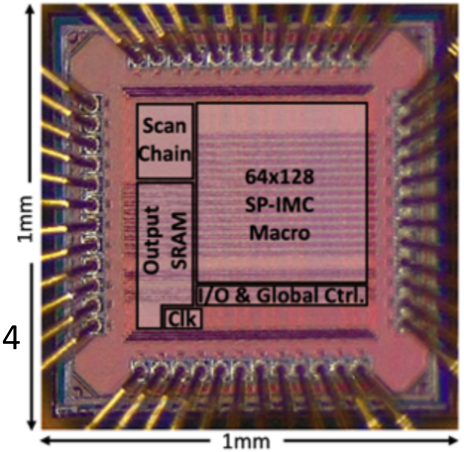


1. Counter/Adder/Scan-chain
2. Sense Amplifier
3. WL Decoder
4. BL Decoder
5. SL Decoder
6. Clk-gen
7. WL Driver
8. SL Driver

ESSCIRC’23, invited to extend to **JSSC** special issue

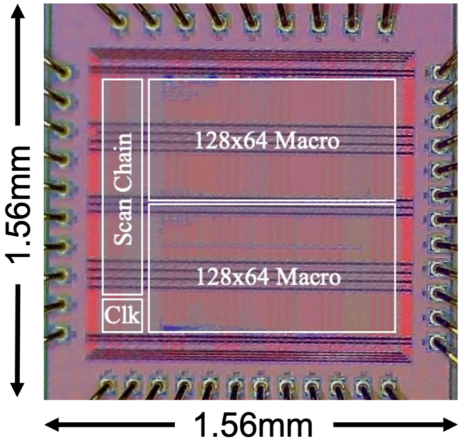
- **28 nm** IMC chip prototype for **sparse in-memory matrix convolution**
- Supports N:M structured sparsity, run length encoding, compressed sparsity column

- Sparse NN
- Wireless Decoder (Neural BP)



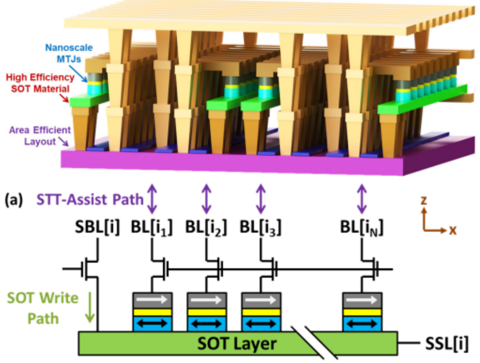
CICC’24

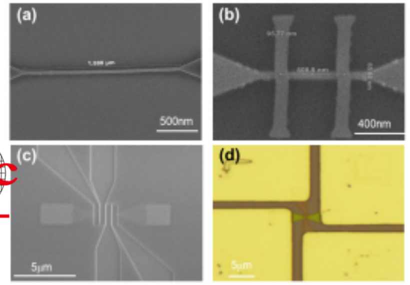
- A **28nm** 2385.7 TOPS/W/b **Precision Scalable** In-Memory Computing Macro with **Bit-Parallel** Inputs and Decomposable Weights for DNNs



SSCL

- energy efficient IMC with high-density, field-free **STT-assisted SOT-MRAM** (SAS-MRAM), ~**100nm, NSF FuSe/ACED projects**



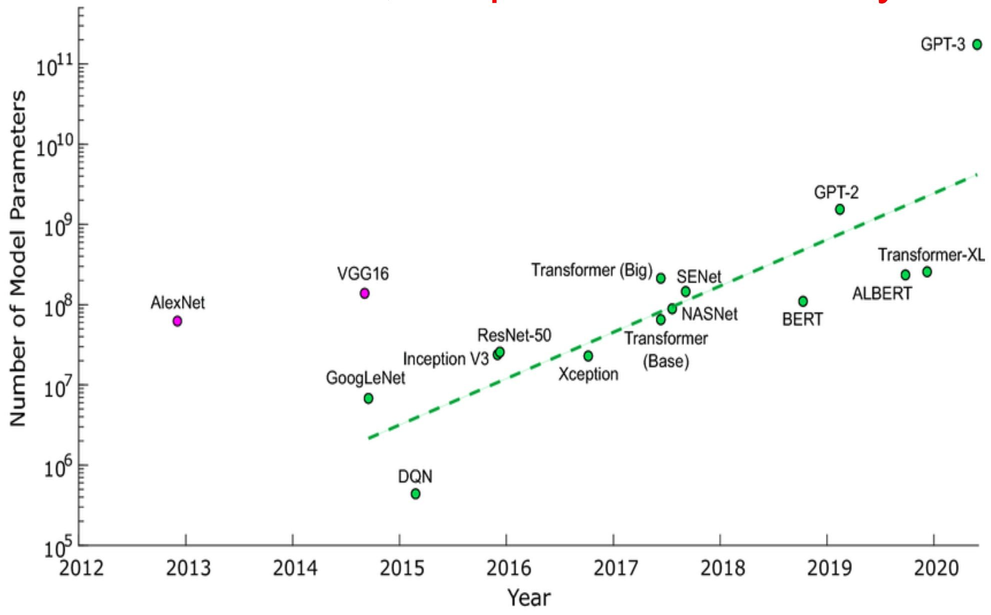


W. Hwang, D. Fan, S. X. Wang, et., al., TMAG, 2022

Efficient AI Computing-in-Memory Needs Algorithm Co-Design

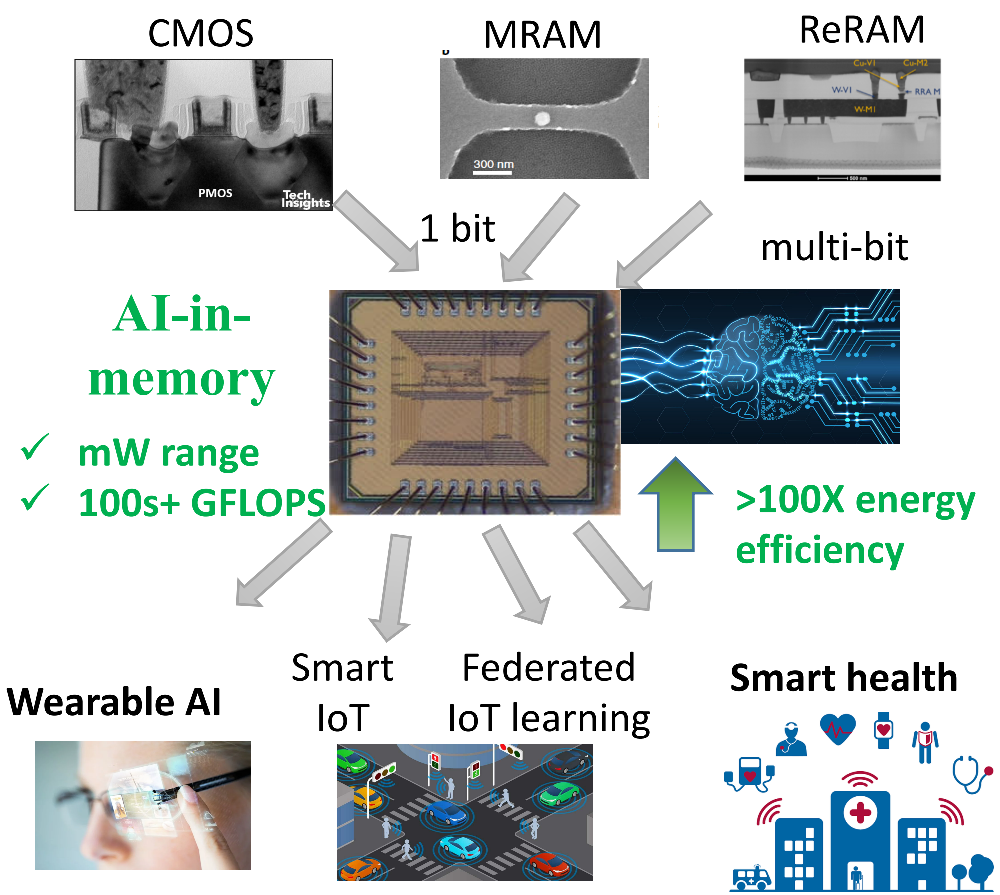
Efficient AI Computing-in-Memory Needs Algorithm Co-Design

- With the evolution of AI goes larger and deeper, memory/computational resources and their communication have faced inevitable limitations, “AI power and memory wall”).



Objective: AI-in-Memory hardware friendly DNN System:

- Hardware friendly model compression, sparse processing, decomposition...
- Without losing inference accuracy
- Run-time dynamics, spatiotemporal dynamics
- Reliability/robustness
- Learning on-chip
- Security/privacy/trustworthy



AI-in-memory

✓ mW range

✓ 100s+ GFLOPS

>100X energy efficiency

Objective: **low power, high performance computing, reliable, smaller AI model, learning-on-device, secure, trustworthy, and more…**

Research projects funded by NSF FuSe, ACED, SHF, CPS, FET, Satc, Career, DARPA, IARPA, SRC, etc.

Algorithm Co-Design for Efficient Deep Learning at Edge

Algorithm

Part I: Efficient & dynamic inference

Hardware-aware model compression

CVPR'19, WACV'19, AAAI'20 (spotlight), TNNLS'20, CVPR'22

Run-time model dynamic inference

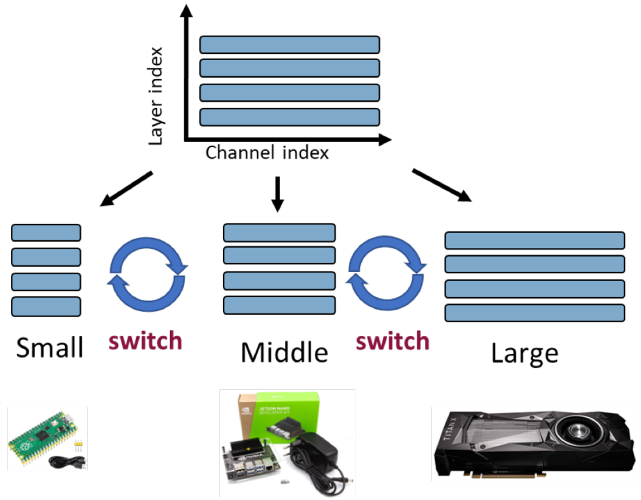
NeurIPS'22, DAC'20, TNNLS'22

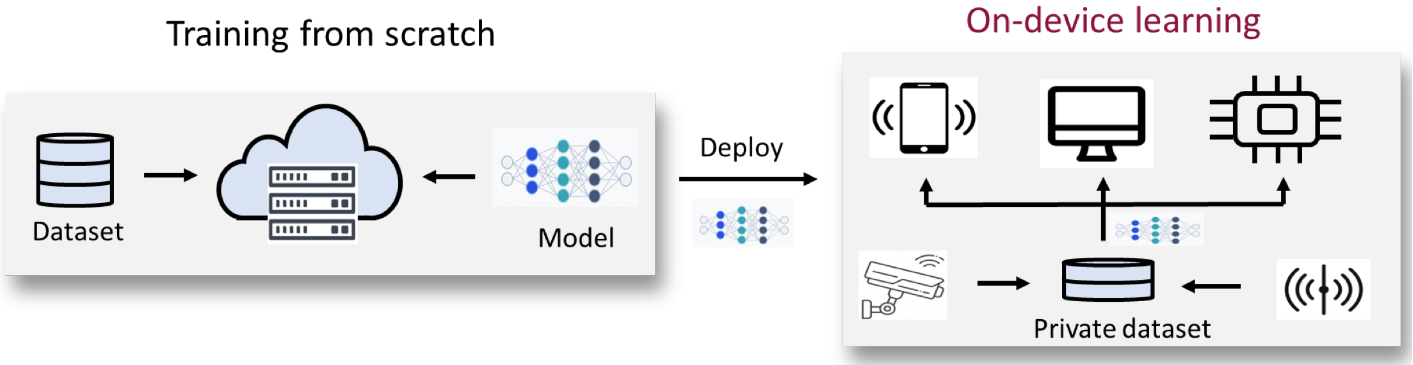
Part II: Efficient learning

Compute- and Memory-Efficient On-device

- Continual/Transfer Learning,
- Self-Supervised Learning

NeurIPS'23, CVPR'21, CVPR'22, CVPR-ECV'22 ICLR'22 (Spotlight), NeurIPS'22, AAAI'22





1.1 Hardware-aware Model Compression: 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!

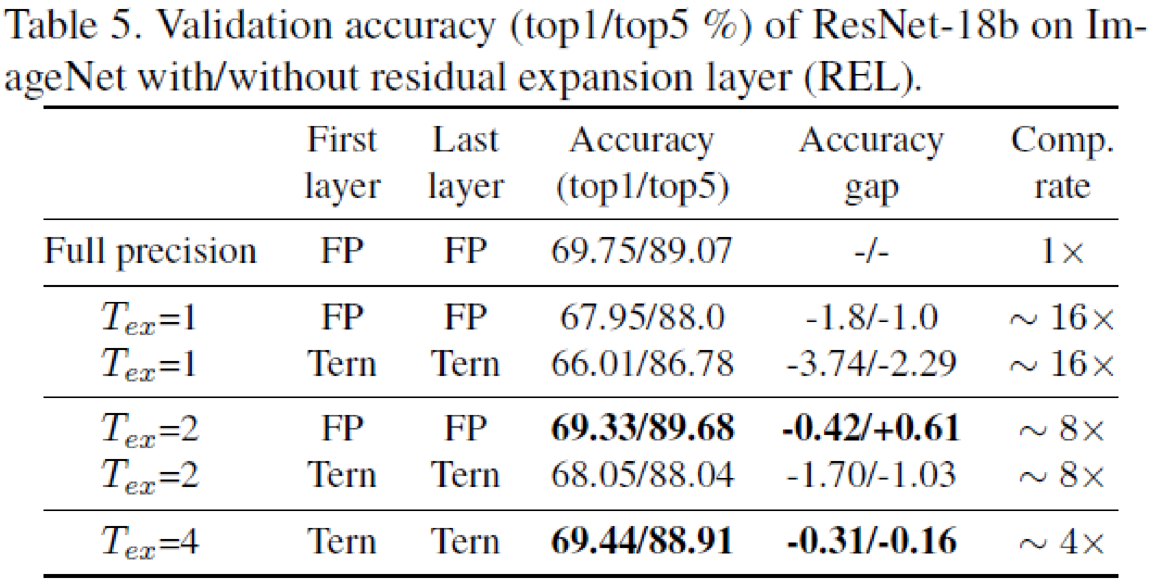


Table 5. Validation accuracy (top1/top5 %) of ResNet-18b on ImageNet with/without residual expansion layer (REL).

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

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

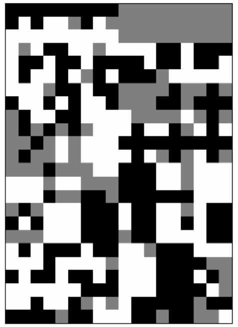
Code to download in https://github.com/elliothe/Ternarized_Neural_Network

1.2 Processing-Element wise Structured Pruning combined with weight ternarization

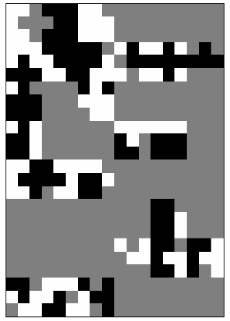
published in **AAAI-2020** as **spotlight** paper

- Aim to effectively integrate structured weight pruning and ternarization to boost the performance of DNN inference on hardware platform, with ultra-small accuracy degradation

	Quan scheme	First layer	Last layer	Accuracy (top1/top/5)	Comp. rate
Baseline	-	FP	FP	69.75/89.07	1×
BWN	Bin.	FP	FP	60.8/83.0	∼ 32×
ABC-Net	Bin.	FP	FP	68.3/87.9	∼ 6.4×
ADMM	Bin.	FP	FP	64.8/86.2	∼ 32×
TWN	Tern.	FP	FP	61.8/84.2	∼ 16×
TTN	Tern.	FP	FP	66.6/87.2	∼ 16×
ADMM	Tern.	FP	FP	67.0/87.5	∼ 16×
(He 2019)	Tern.	FP	FP	67.95/88.0	∼ 16×
Ours	Tern.	FP	FP	**68.01/88.13**	∼ 21.3×



Ternarization only



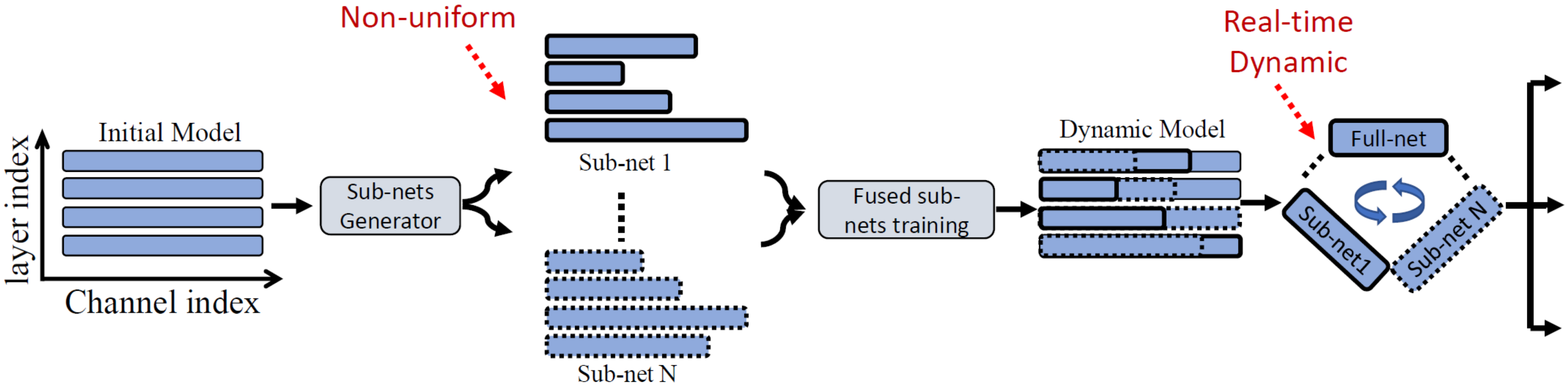
Ours

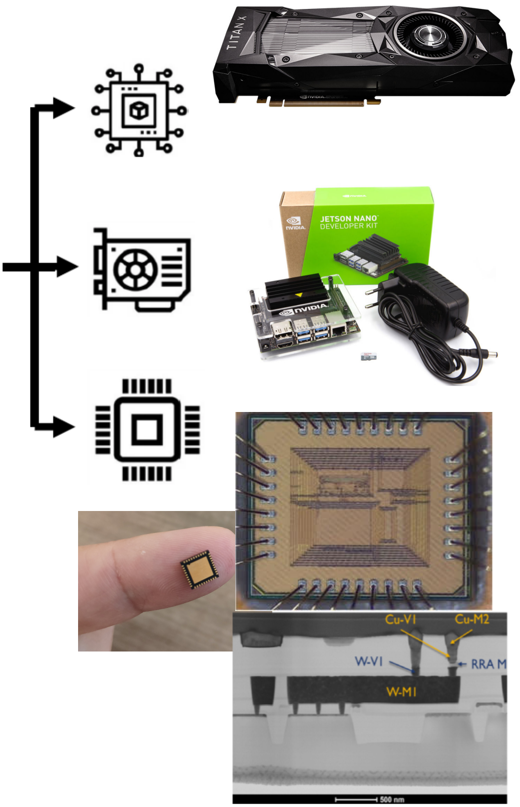
{white, grey, black} denotes {-1,0,+1}

- Comparing with weight Binarization and Ternarization without pruning
- Achieve both high accuracy and compression rate
- **Even better accuracy and higher compression rate than ternary-only**

Li Yang, Zhezhi He and Deliang Fan, "Harmonious Coexistence of Structured Weight Pruning and Ternarization for Deep Neural Networks," *Thirty-Fourth AAAI Conference on Artificial Intelligence* (AAAI), Feb. 7-12 2020, New York, USA **(spotlight)**

1.3 Make DNN flexible and run-time dynamic

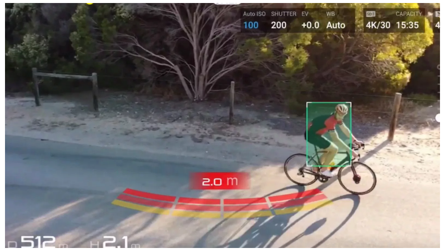


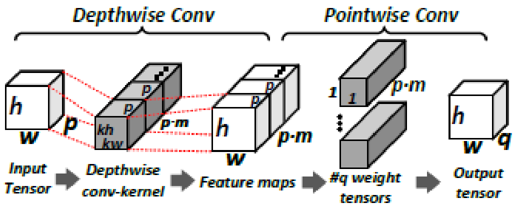


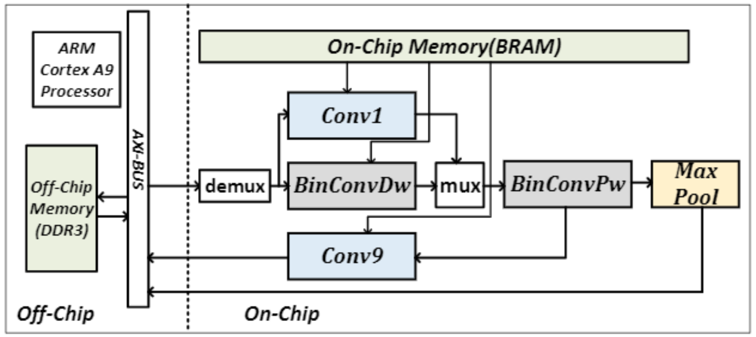
1. Temporal dynamic: environment, working load, computing resources/mode, etc.
2. Spatial dynamic: Training once and get different models for heterogenous hardware devices with different resources and performance requirements

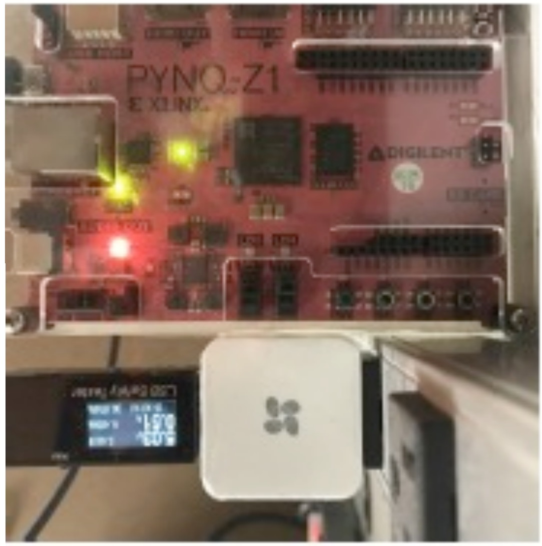
- Li Yang, Zhezhi He, Yu Cao and Deliang Fan. “Non-uniform DNN Structured Subnets Sampling for Dynamic Inference”. *In: 57th Design Automation Conference (**DAC**)*, San Francisco, CA, July 19-23, 2020
- Li Yang, Shaahin Angizi, Deliang Fan, “A Flexible Processing-in-Memory Accelerator for Dynamic Channel-Adaptive Deep Neural Networks,” Asia and South Pacific Design Automation Conference (***ASP-DAC***), Jan. 13-16, 2020,
- Li Yang, Zhezhi He, Yu Cao and Deliang Fan, “A Progressive Sub-network Searching Framework for Dynamic Inference”, ***IEEE TNNLS***
- Li Yang, Jian Meng, Jae-sun Seo, and Deliang Fan, “Get More at Once: Alternating Sparse Training with Gradient Correction,” Thirty-sixth Conference on Neural Information Processing Systems (***NeurIPS***), New Orleans, LA, 2022

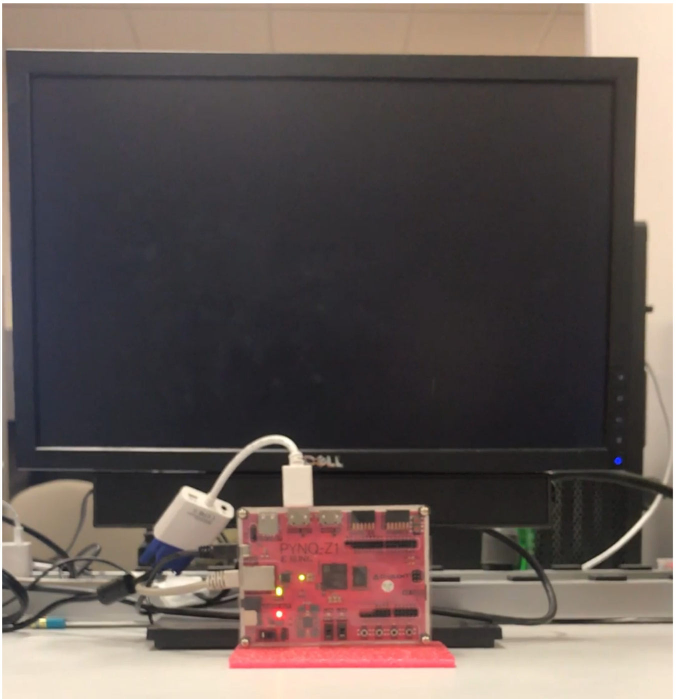
System demo-1: Deep Neural Network IoT FPGA











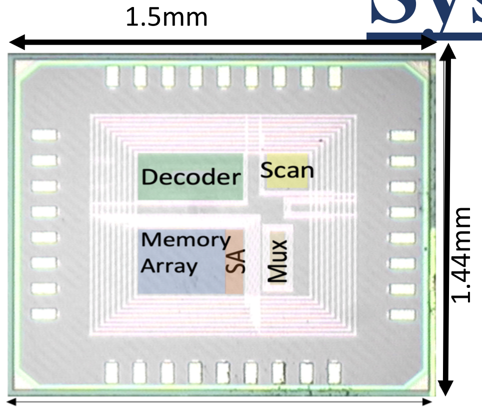
Name	IOU	Power	FPS	ES	TS
TGIIF	0.62	4.2	11.955	1.0318	1.2674
SystemsETHZ	0.49	2.45	25.968	1.3976	1.1794
iSmart2	0.57	2.59	7.349	1.0297	1.1636
traix	0.61	3.11	5.445	0.8869	1.1523
hwac-object-tracker	0.52	3.66	4.935	0.8155	0.932
Ours	0.57	2.61	11.1	1.1477	**1.224**

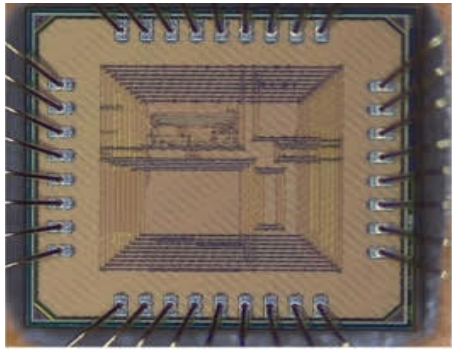
https://dfan.engineering.asu.edu/deep-learning-neural-network/

- 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**

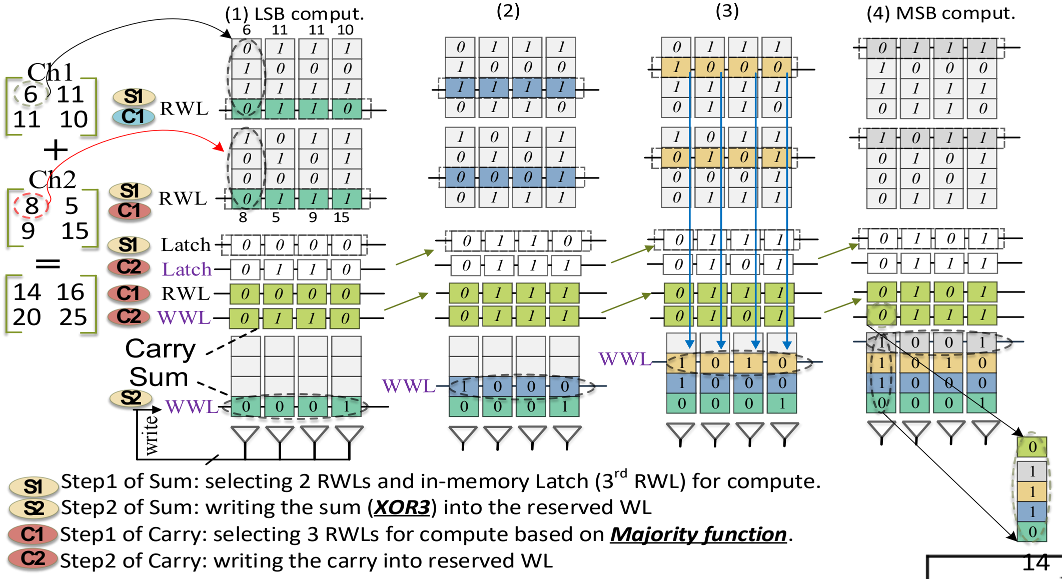
D. Fan, et. al., GLSVLSI 2019

System-2: AI-in-Memory Chip Prototype (65nm)





@TSMC 65nm

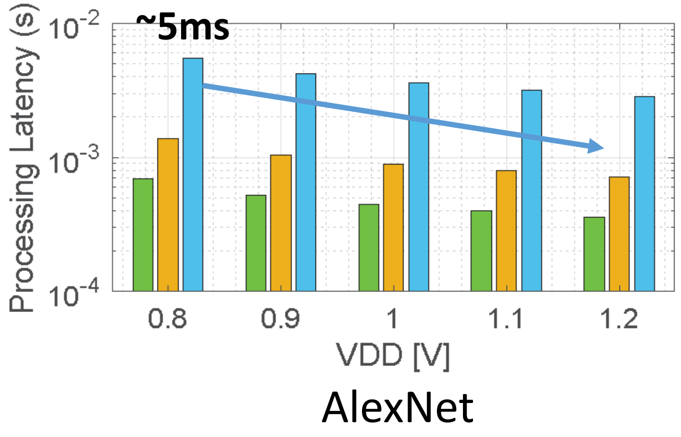


S1 Step1 of Sum: selecting 2 RWLs and in-memory Latch (3^{rd} RWL) for compute.

S2 Step2 of Sum: writing the sum (***XOR3***) into the reserved WL

C1 Step1 of Carry: selecting 3 RWLs for compute based on ***Majority function***.

C2 Step2 of Carry: writing the carry into reserved WL



AlexNet

Technology	65nm
Bitcell Size	1.68umx2.715um
Chip Size	1.5x1.44 mm^2
Supply Voltage	0.8-1.2V
Memory Capacity	2KB
SRAM Sub array	128x128
Clock Frequency	1.23Ghz@1.2V
Average power	36mW@1.2V

		BNN Accelerators		Generic Accelerators	
Reference	**Proposed**	**JSSC'19 [1]**	**JSSC'19 [2]**	**JSSC'20 [3]**	**VLSI Symp'18 [4]**
Technology	65nm	65nm	65nm	28nm	40nm
Bit cell Density	8T	10T	8T	8T Transposable	10T
Supply Voltage	0.8-1.2V	0.8-1.2V	0.68-1.2V	0.6 – 1.1V	0.5-0.9V
Max Frequency	1230MHz(1.2V)	5MHz	100MHz	475MHz (1.1V)	28.8MHz (0.7V)
SRAM Macro Size	2KB	2KB	4.8KB	16 KB	8KB
Die Area	2.16mm^2	4mm^2	12.6mm^2	2.7mm^2	1.275mm^2
Performance (GOPS)	629.76	0.75	147	32.7	14.7
Performance per unit Area (GOPS/mm2)	291.56	11.9	11.7	27.3	70
Energy Efficiency (TOPS/W)	17.49	4.81	10.3	5.27 (add) 0.55(Mult)	31.28
Reconfigurable	Programmable	N/A	N/A	Programmable	N/A

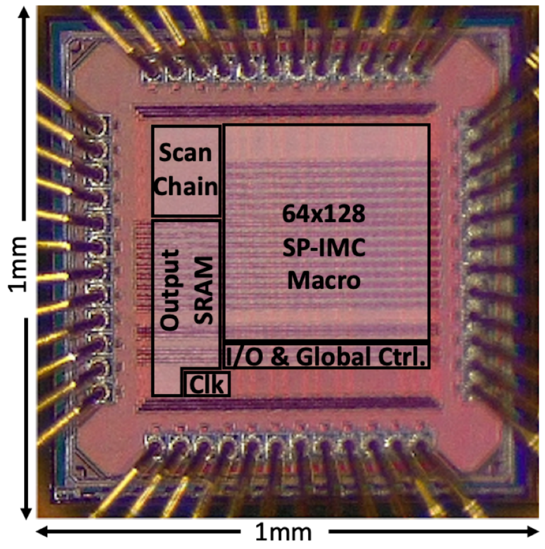
[1] A. Biswas, et al., JSSC, 2019.

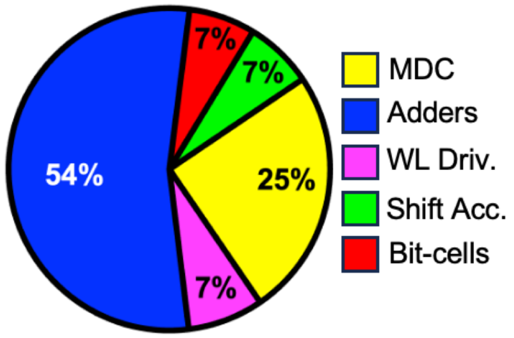
[2] H. Valavi, et al., JSSC, 2019.

[3] W. Jingcheng, et al., JSSC, 2019.

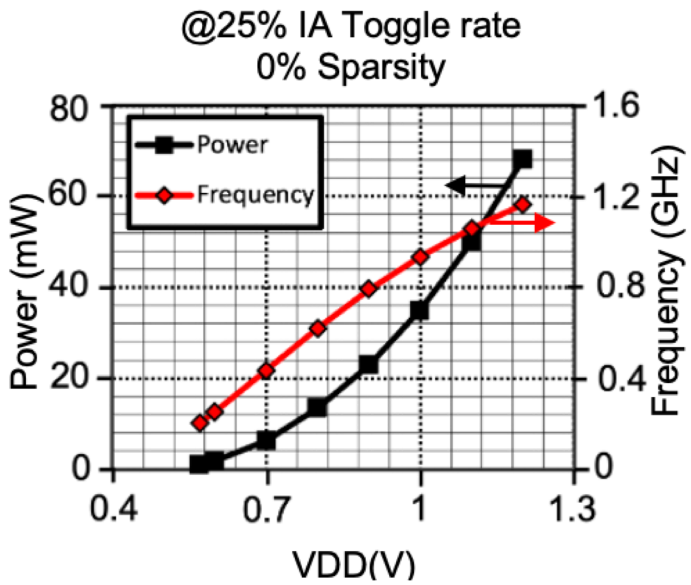
[4] Y. Zhang, et al., Symp. VLSI Circuits, 2019.

System-3: Sparse AI-in-Memory Chip Prototype (28nm)

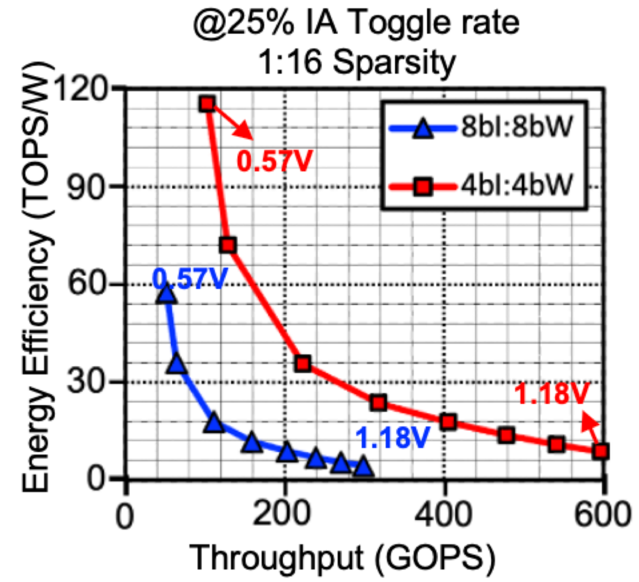




@TSMC 28nm



@25% IA Toggle rate
0% Sparsity



@25% IA Toggle rate
1:16 Sparsity

Work	ISSCC'22 [1]	ISSCC' 23 [4]	ISSCC'22 [6]	ISSCC'23 [2]	ESSCIRC'23 [3]	**This Work**
Technology	28nm	28nm	5nm	4nm	28nm	28nm
IMC Sparsity Support	X	X	X	X	X	RLC/CSC/N:M
Supply Voltage (V)	0.45-1.10	0.64-1.03	0.5-0.9	0.32-1.1	0.9-1.1	0.57-1.18
Macro Area (mm^2)	0.049	NA	0.0133	0.0172	0.0159	0.24
Clock Frequency (MHz)	250	20-320	360-1440	1490	30-360	201-1160

Bitcell Transistors	8T	8T(55%)
		10T(45%)
			12T	8T x 2bit
				+OAI
					6T+0.5T	6T+4T(50%)
						6T(50%)

Array Size(b)	16K	1.15M	64K	54K	16K	4K(Weights)
						+ 4K(Index)

Bit Precision	IA:1-4b W:1b	INT8	IA: 1-8b W:4b	IA: 8/12/16 W: 8/12	IA: 1-8 W: 8	IP:2b/4b/8b W:4b/8b
Full output precision	No	Yes	Yes	Yes	Yes	Yes
Performance(GOPS)[1,2,7]	62.5*	22.9*	104.735	127.15	0.95-11.6	**41.29-238.86[6]**
Peak Energy Efficiency[2,7] (TOPS/W)	9.6-15.5	15.6[4]/70.37[5] (System)	17.5-63	87.4[(8)] 41.3[(9)]	22.4-60.4	4.38-57.67[6]
Compute Density[2,3,7] TOPS/mm^2	2.59	0.85	0.44-1.76	0.27-1.01	0.12-1.46	0.21-1.2[7]

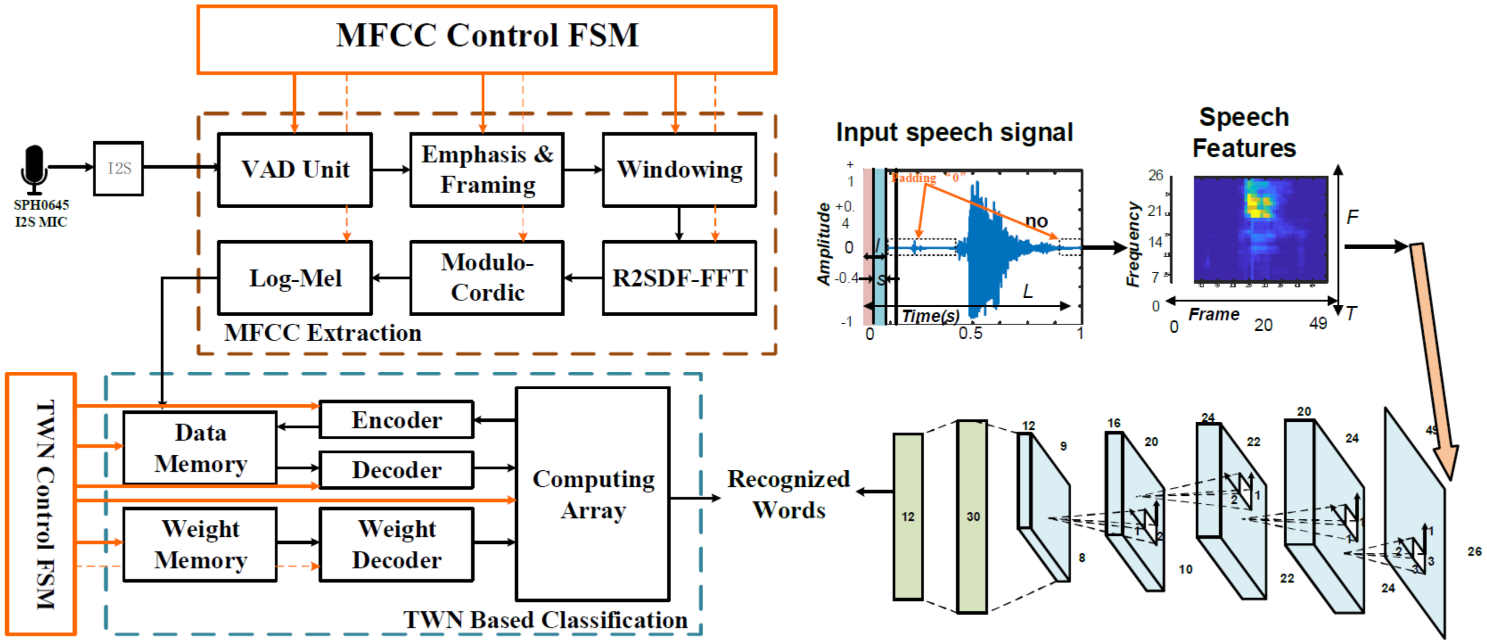
FoM	3.182K	2.97K[4]
		3.25K[5]
			3.942K	3.219K	0.9K-4.1K	**11K-24K(1:16)
						5.5K-12K(1:8)**

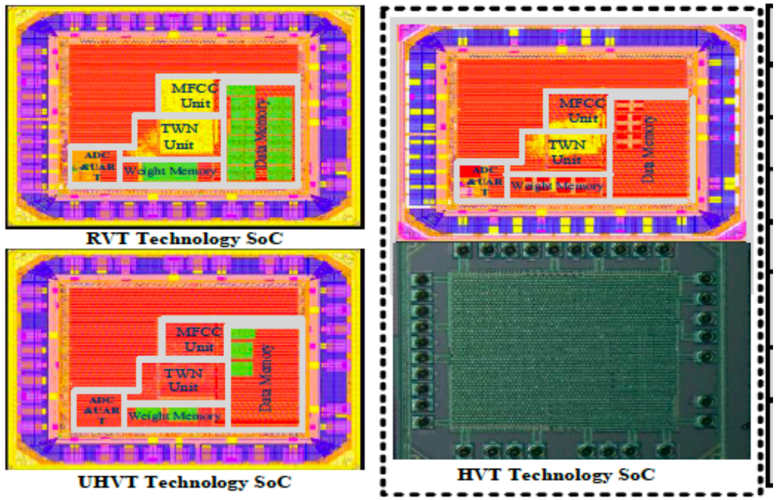
- **8-bit Resnet-18**
- Best Performance (GOPS)
- SoTA Energy Efficiency, **~60TOPS/W, ~100X better than GPU**
- Best FoM

FoM = TOPS/W$^{(7)}$ * TOPS/mm$^{2\ (7)}$ * # of W/kb

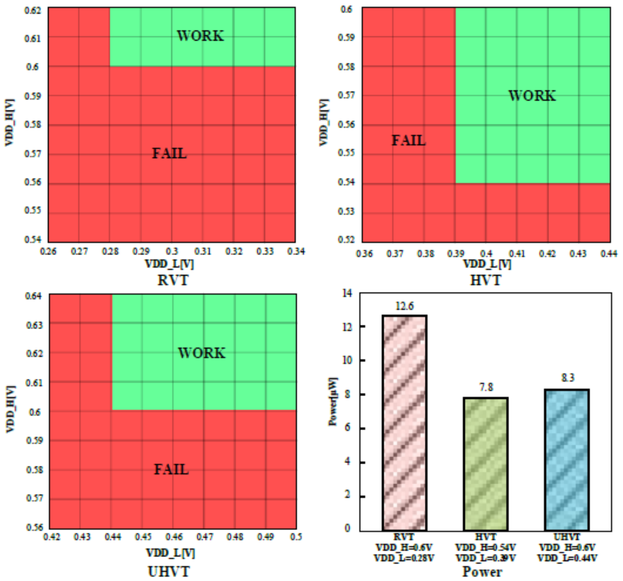
[CICC'24] D. Fan. et. al., "SP-IMC: A Sparsity Aware In-Memory-Computing Macro in 28nm CMOS with Configurable Sparse Representation for Highly Sparse DNN Workloads," *IEEE Custom Integrated Circuits Conference (CICC)*, 21 – 24 April 2024, Denver, CO

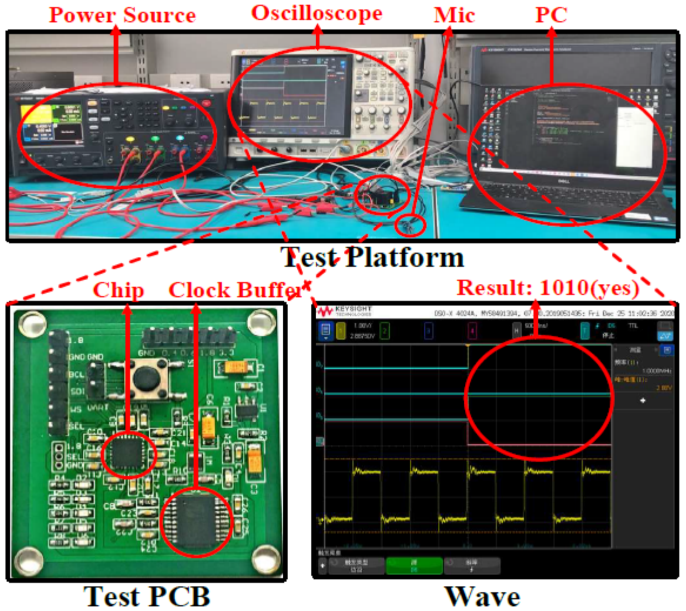
System-4: CMOS AI-ASIC of Ternary Network (22nm)





Technology	22 nm HVT CMOS
Supply voltage	0.39 V/ 0.54 V
Frequency	250 KHz
Core size	0.99x0.61mm^2
Die size	1 mm^2
Logic gates (NAND2)	0.58 million
SRAM	8 kB
Power consumption	4.8 µ W (average) 7.8 µ W (peak)





- **ternary neural network for keyword voice recognition**
- **only 4.8uW @22nm, 250KHz, 8KB memory, accuracy: 90.6%**
- Published in *IEEE Open Journal of the Solid-State Circuits Society* , 2023, DOI: 10.1109/OJSSCS.2023.3312354

Algorithm Co-Design for Efficient Deep Learning at Edge

Algorithm

Part I: Efficient & dynamic inference

Hardware-aware model compression

CVPR'19, WACV'19, AAAI'20 (spotlight), TNNLS'20, CVPR'22

Run-time model dynamic inference

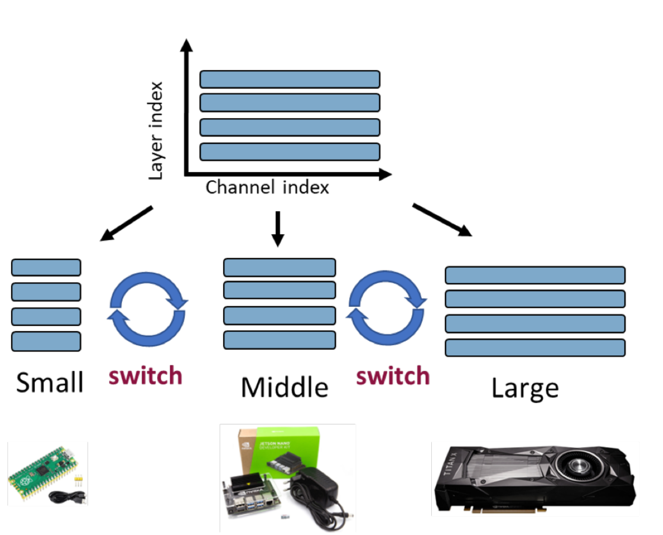
NeurIPS'22, DAC'20, TNNLS'22

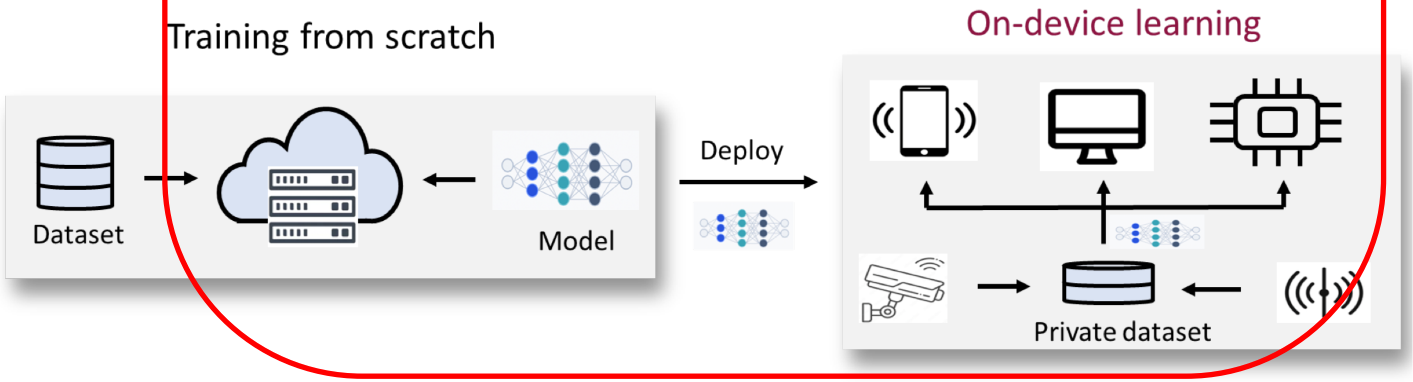
Part II: Efficient learning

Compute- and Memory-Efficient On-device

- Continual/Transfer Learning,
- Self-Supervised Learning

NeurIPS'23, CVPR'21, CVPR'22, CVPR-ECV'22 ICLR'22 (Spotlight), NeurIPS'22, AAAI'22

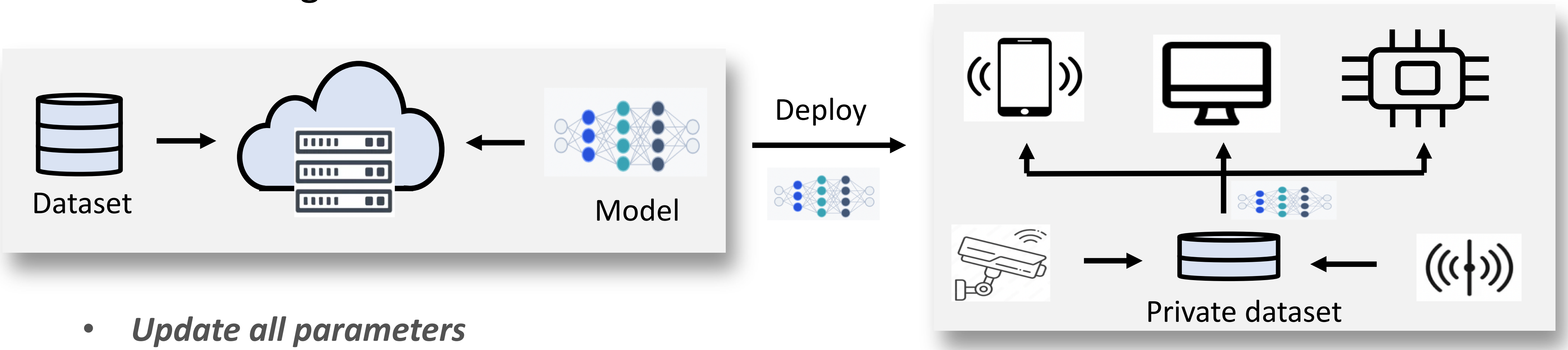




Background of on-device learning

Training from scratch

On-device learning



- *Update all parameters*
- *Large dataset*
- *Large training volume (e.g, epochs, batchsize)*

- **Transfer Learning**
- **Continual Learning**

- ***Pre-trained model***
- ***Streaming tasks***
- ***Small training volume***

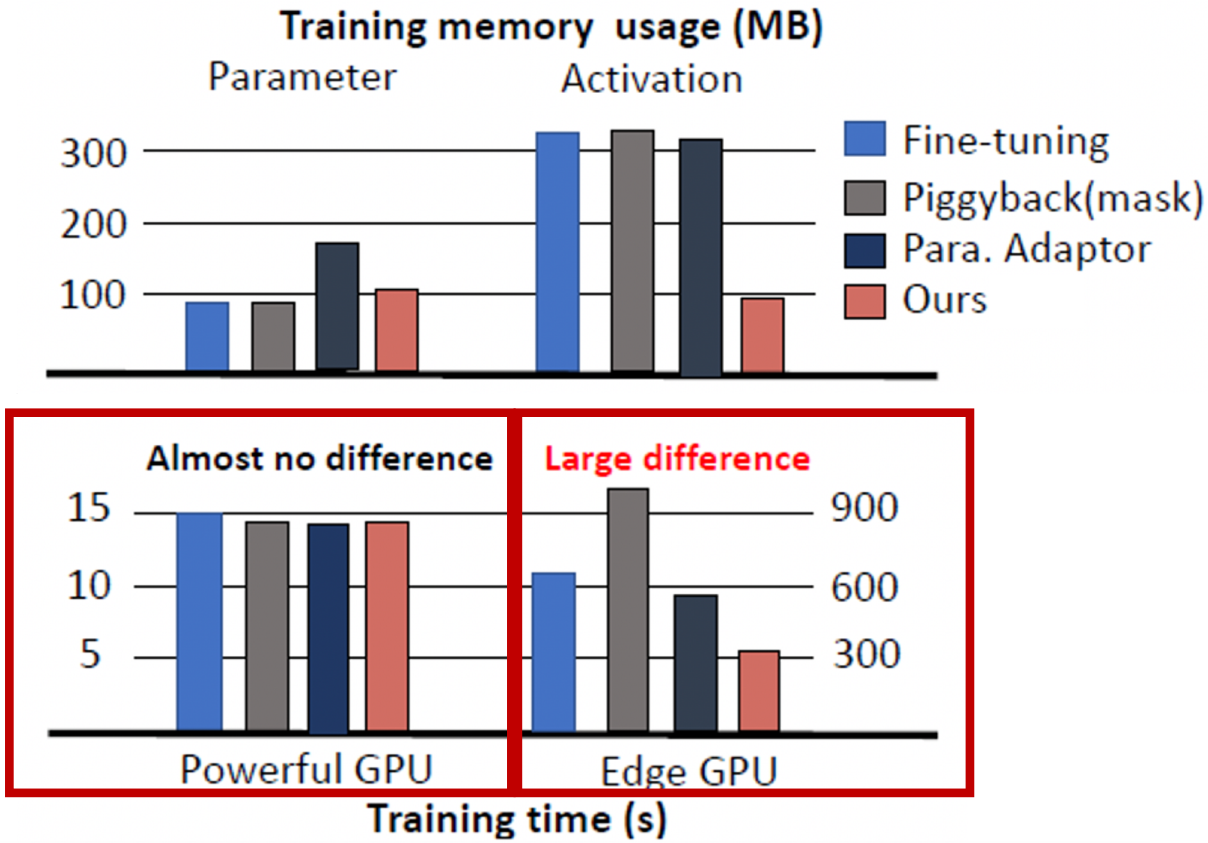
Motivation & Challenge: On-Device Learning

Training process is compute-intensive

- **Mask or adaptor** based compute-efficient continual learning

But still memory-intensive

- **Activation/features memory** is almost 3X larger than the model itself
- powerful GPU: large memory usage during training is not an issue
- tiny GPU: **large memory usage becomes the bottleneck for training speed**



Conclusion: conventional compute-efficient continual learning process is still memory-intensive, the intermediate activation memory during training is the bottleneck

L. Yang, et al. DA3: Dynamic Additive Attention Adaption for Memory-Efficient On-Device Multi-Domain Learning. (CVPR-ECV'22)

ImageNet dataset) to Flower dataset.

Memory Usage in Continual Training

Fine-tuning based methods:

- assume a linear layer whose forward process is

$$a_{i+1} = a_i\mathbf{W} + b$$

- Then, the weight back-propagation process is

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = a_i \frac{\partial \mathcal{L}}{\partial a_{i+1}} \quad \& \quad \frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial a_{i+1}}$$

Mask-based learning method:

- assume a linear layer whose forward process is

$$a_{i+1} = a_i(\mathbf{W} \cdot \mathbf{M})$$

- Then, the mask back-propagation process is

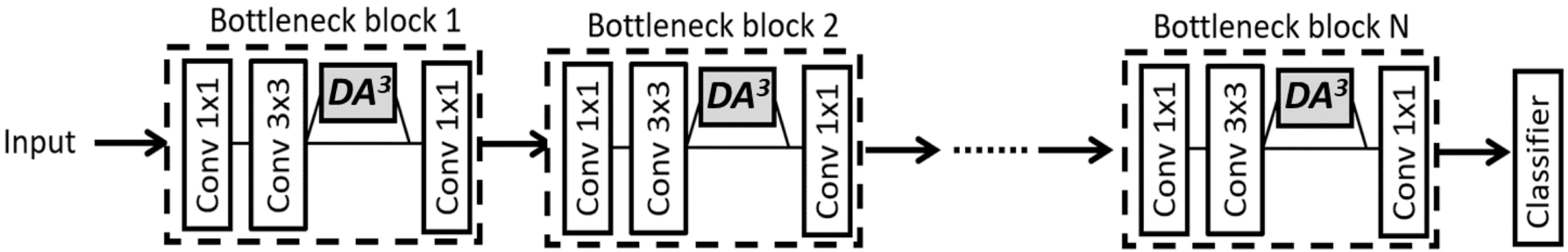
$$\frac{\partial \mathcal{L}}{\partial \mathbf{M}} = a_i \frac{\partial \mathcal{L}}{\partial a_{i+1}} \cdot \mathbf{W}$$

Conclusion:

- Multiplicative relationship causes large memory usage: To update weight-**W** or mask-**M,** activation-a_i needs to be stored for computing, causing large memory usage
- Additive relationship needs no activation buffering: if only updating bias-b, large activation memory could be saved

Problem: only updating bias has very limited adaption capacity to learn new domain data

Solution-1: DA^3: Deep Additive Attention Adaptor



Problem: only updating bias has very limited adaption capacity to learn new domain data

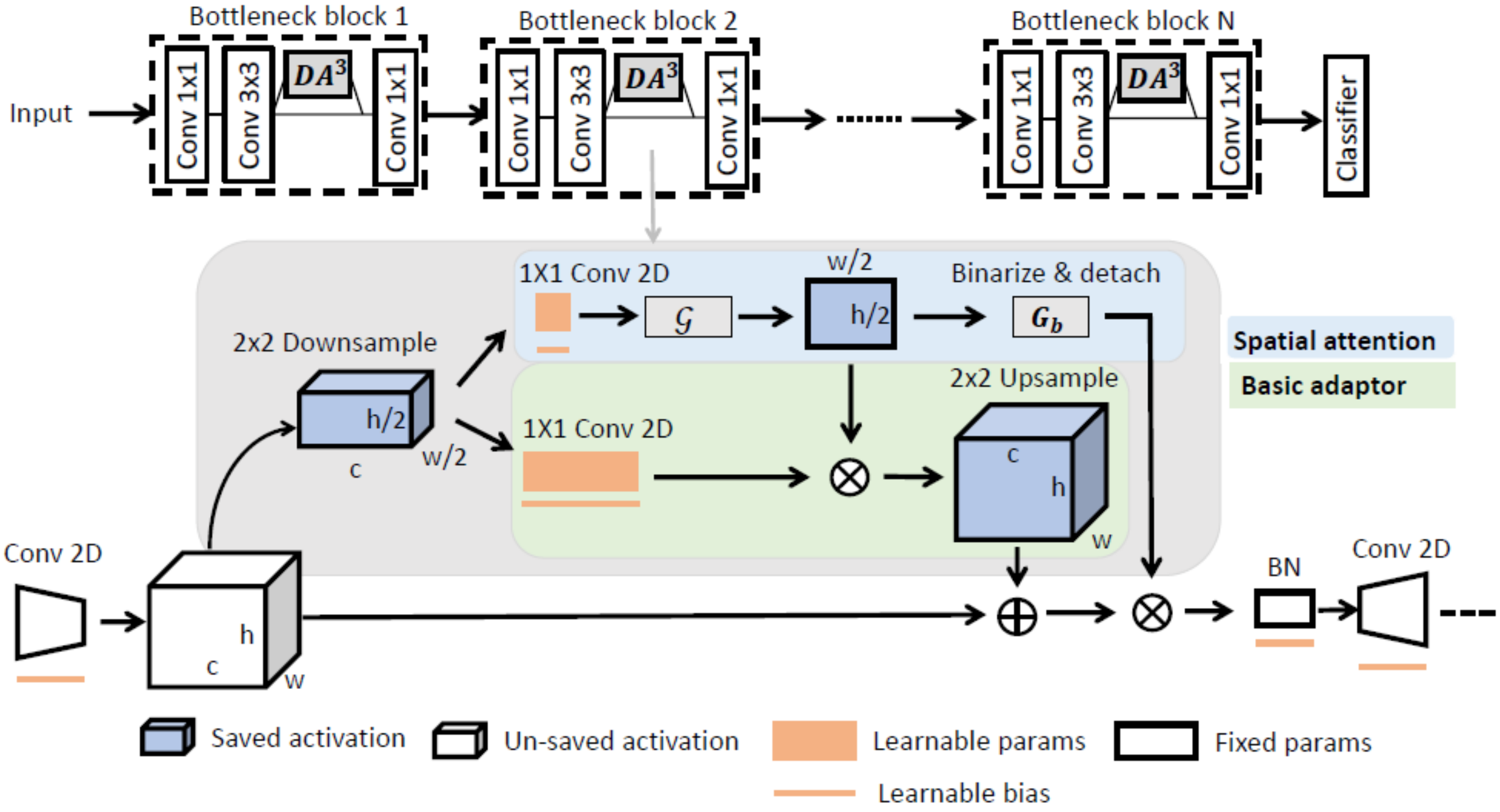
Solution: designed Additive Attention Adaptor (DA^3)

$$\mathbf{A}_i^* = \mathbf{A}_i \boxed{+\mathbf{DA^3}(\mathbf{A}_i)}$$

- Improved the domain adaption capacity
- Reduced computation: freeze the weight of pre-trained model, and only updates DA^3
- Reduced Memory usage:
  - Additive relationship needs **no activation storage of pre-trained model**, only tiny DA^3
  - DA^3 module could be mapped to the SRAM module for frequent update, and backbone model could be mapped to the RRAM module for maintaining the basic backbone forward operations

Li Yang, Adnan Siraj Rakin, and Deliang Fan, "DA3 : Dynamic Additive Attention Adaption for Memory-Efficient On-Device Learning" *Efficient Deep Learning for Computer Vision CVPR Workshop, ,* New Orleans, Louisiana, June 19-24, 2022

DA^3:Deep Additive Attention Adaptor



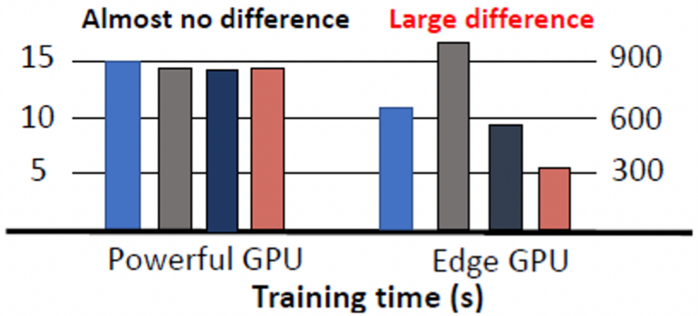
- light-weight 1x1 convolution layer, as well as an attention module to filter out features for certain new domain/task
- To further reduce the activation size, a 2×2 average pooling to down-sample the input feature map

Experiments: ImageNet-to-Sketch dataset

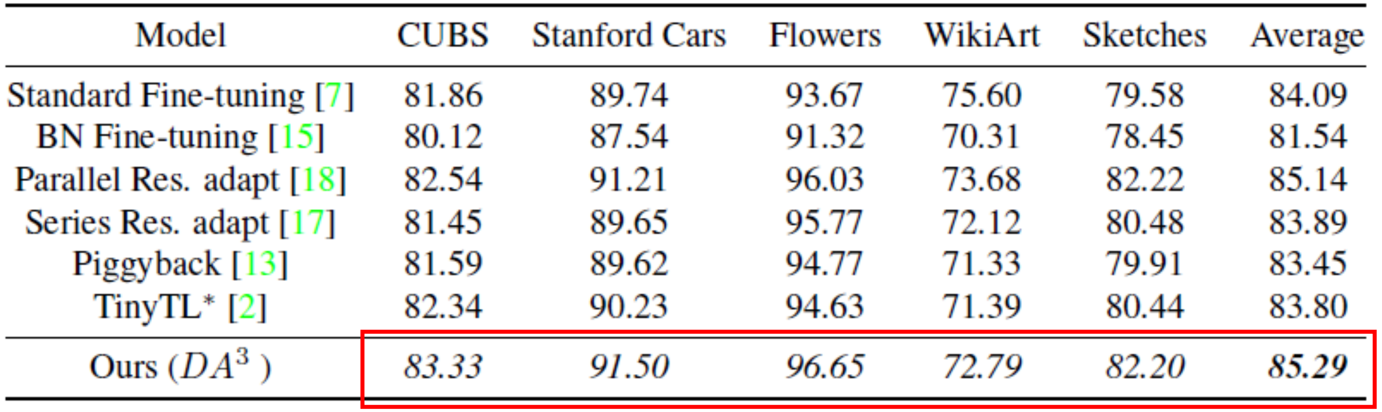
- Setting: ResNet50 pretrained on ImageNet dataset

- Accuracy Comparison: achieves the best accuracy in CUBS, Stanford Cars and Flowers dataset.

- Training Cost Comparison
  - Reduce the activation memory size by 19-37×
  - Training time reduces nearly by 2×



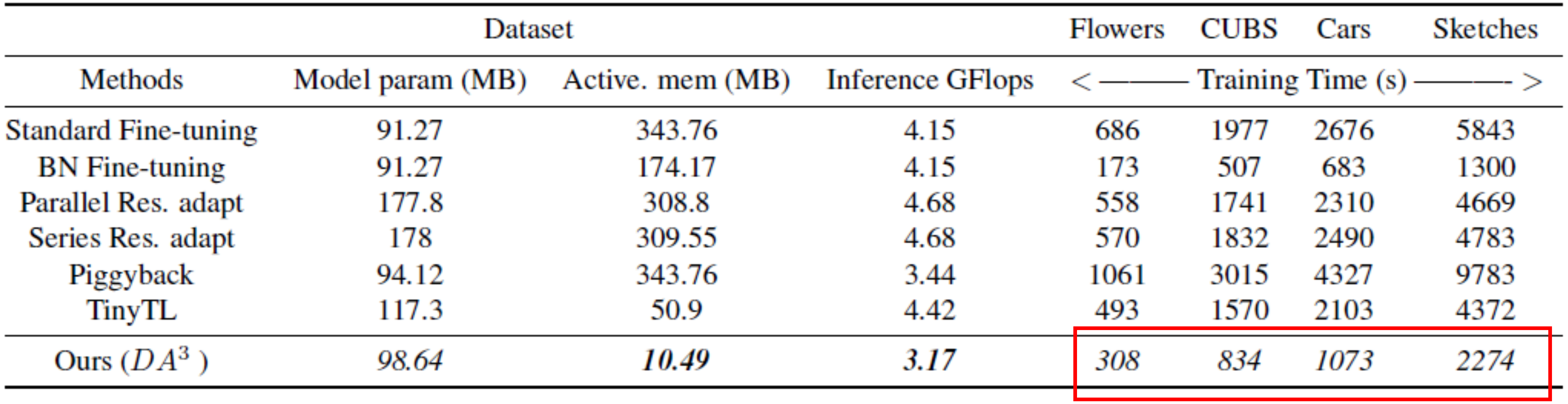
Accuracy



Model	CUBS	Stanford Cars	Flowers	WikiArt	Sketches	Average
Standard Fine-tuning [7]	81.86	89.74	93.67	75.60	79.58	84.09
BN Fine-tuning [15]	80.12	87.54	91.32	70.31	78.45	81.54
Parallel Res. adapt [18]	82.54	91.21	96.03	73.68	82.22	85.14
Series Res. adapt [17]	81.45	89.65	95.77	72.12	80.48	83.89
Piggyback [13]	81.59	89.62	94.77	71.33	79.91	83.45
TinyTL* [2]	82.34	90.23	94.63	71.39	80.44	83.80
Ours (DA^3)	*83.33*	*91.50*	*96.65*	*72.79*	*82.20*	***85.29***

best

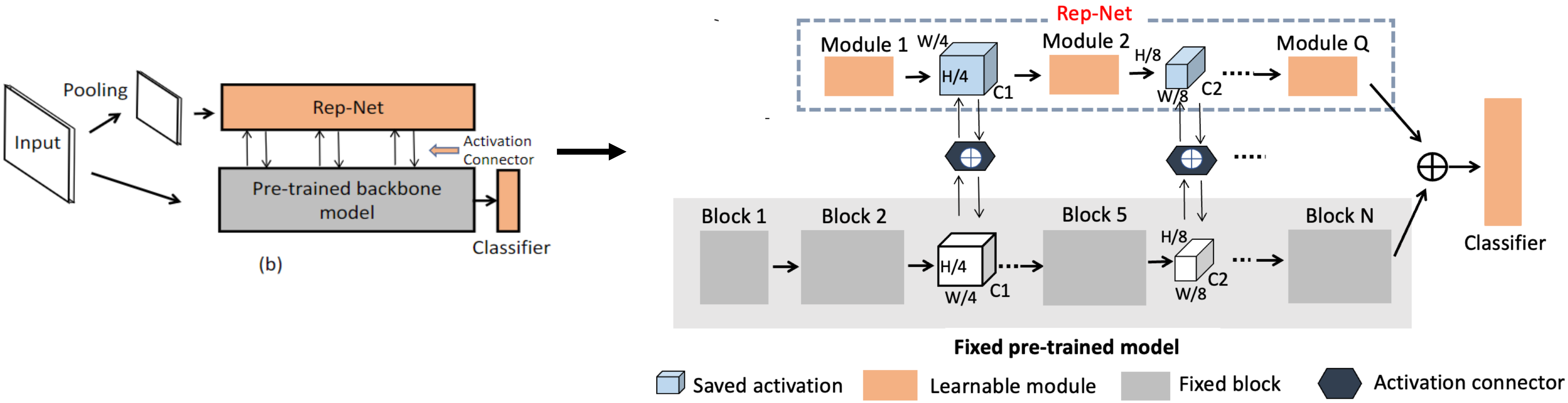
Training memory(MB) and training time (s)) on NVIDIA Jetson Nano GPU



Dataset				Flowers	CUBS	Cars	Sketches
Methods	Model param (MB)	Active. mem (MB)	Inference GFlops	< ——— Training Time (s) ——— >			
Standard Fine-tuning	91.27	343.76	4.15	686	1977	2676	5843
BN Fine-tuning	91.27	174.17	4.15	173	507	683	1300
Parallel Res. adapt	177.8	308.8	4.68	558	1741	2310	4669
Series Res. adapt	178	309.55	4.68	570	1832	2490	4783
Piggyback	94.12	343.76	3.44	1061	3015	4327	9783
TinyTL	117.3	50.9	4.42	493	1570	2103	4372
Ours (DA^3)	*98.64*	***10.49***	***3.17***	*308*	*834*	*1073*	*2274*

Note: training time is the measured GPU time of training one epoch with batch size 4 in average.

Solution-2: Rep-Net: Tiny Reprogramming Network

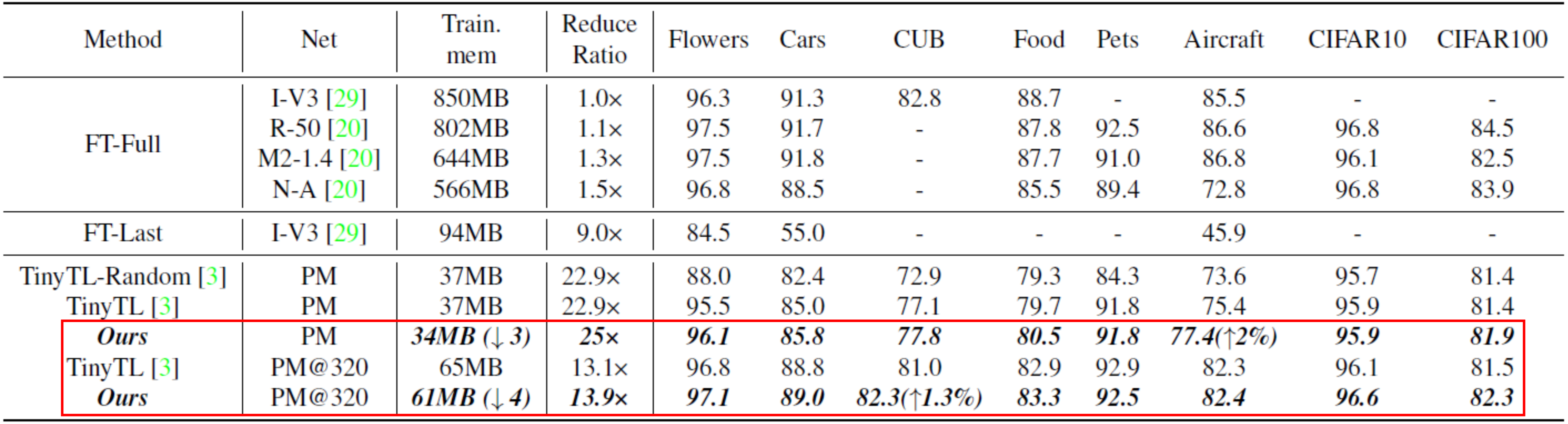


- A lightweight side-network that is executed with the pretrained backbone model in parallel
- Consist of multiple modules (2 conv + batchnorm)
- ***Activation connector*** to reprogram the feature of fixed backbone model

Improve the domain adaption capacity: to learn new domain, as well as reducing computation, **freeze the weight of pre-trained model**, and only updates $tiny\ Rep - Net$ portion. Memory efficient without storing activation mem.

Li Yang, A. Rakin, D. Fan, ‘Rep-Net: Efficient On-Device Learning via Feature Reprogramming’, **CVPR 2022**

Comparison with SOTA in Transfer Learning



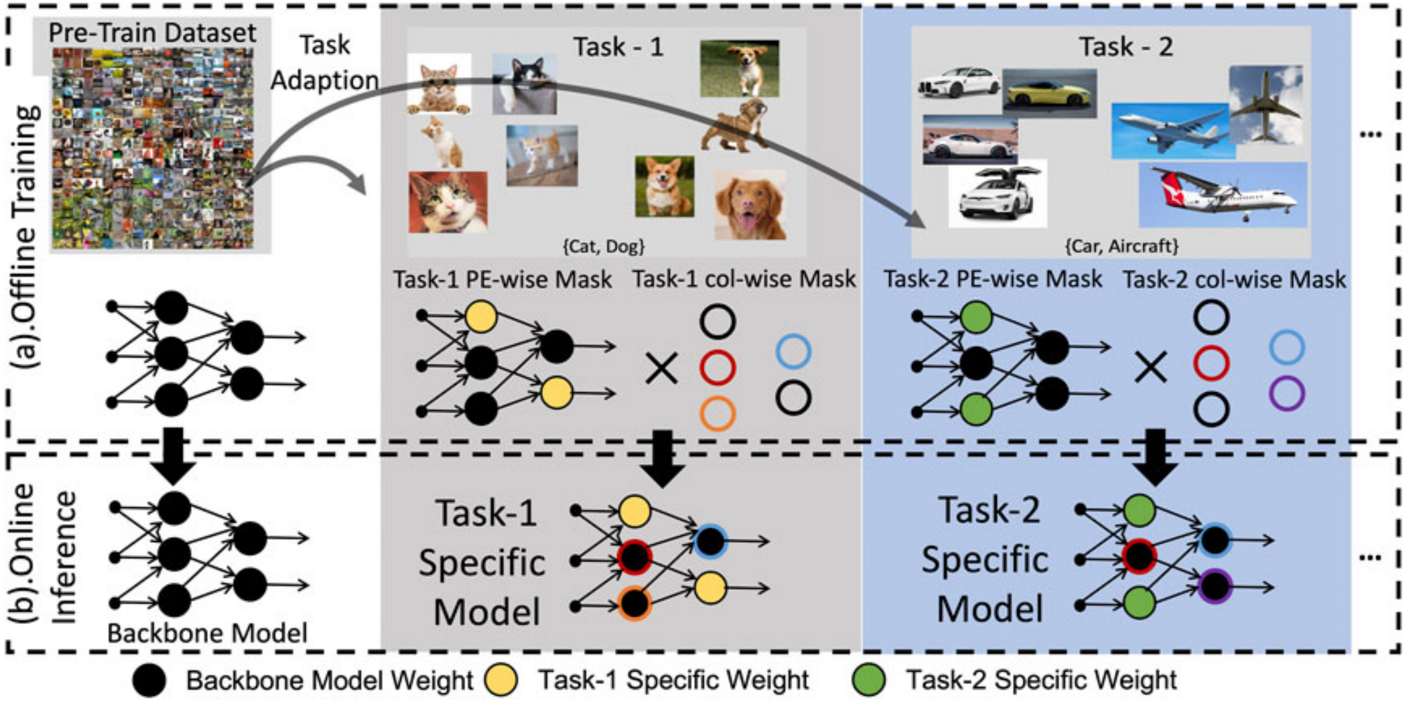
Method	Net	Train. mem	Reduce Ratio	Flowers	Cars	CUB	Food	Pets	Aircraft	CIFAR10	CIFAR100
FT-Full	I-V3 [29]	850MB	1.0×	96.3	91.3	82.8	88.7	-	85.5	-	-
	R-50 [20]	802MB	1.1×	97.5	91.7	-	87.8	92.5	86.6	96.8	84.5
	M2-1.4 [20]	644MB	1.3×	97.5	91.8	-	87.7	91.0	86.8	96.1	82.5
	N-A [20]	566MB	1.5×	96.8	88.5	-	85.5	89.4	72.8	96.8	83.9
FT-Last	I-V3 [29]	94MB	9.0×	84.5	55.0	-	-	-	45.9	-	-
TinyTL-Random [3]	PM	37MB	22.9×	88.0	82.4	72.9	79.3	84.3	73.6	95.7	81.4
TinyTL [3]	PM	37MB	22.9×	95.5	85.0	77.1	79.7	91.8	75.4	95.9	81.4
Ours	PM	***34MB (↓ 3)***	***25×***	***96.1***	***85.8***	***77.8***	***80.5***	***91.8***	***77.4(↑2%)***	***95.9***	***81.9***
TinyTL [3]	PM@320	65MB	13.1×	96.8	88.8	81.0	82.9	92.9	82.3	96.1	81.5
Ours	PM@320	***61MB (↓ 4)***	***13.9×***	***97.1***	***89.0***	***82.3(↑1.3%)***	***83.3***	***92.5***	***82.4***	***96.6***	***82.3***

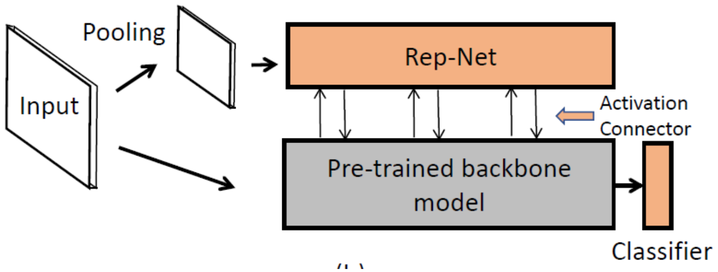
- Accuracy Comparison: achieves the best accuracy in average.
- Training Cost Comparison: reduce total training memory by 14-25× in comparison to other methods. Thus, emerging as an ideal candidate for on-device learning purposes.

- Setting: N-A is NASNet-A Mobile, M2-1.4 is MobileNet V2-1.4, R-50 is ResNet-50, PM is ProxylessNAS-Mobile

Li Yang, A. Rakin, D. Fan, ‘Rep-Net: Efficient On-Device Learning via Feature Reprogramming’, **CVPR 2022**

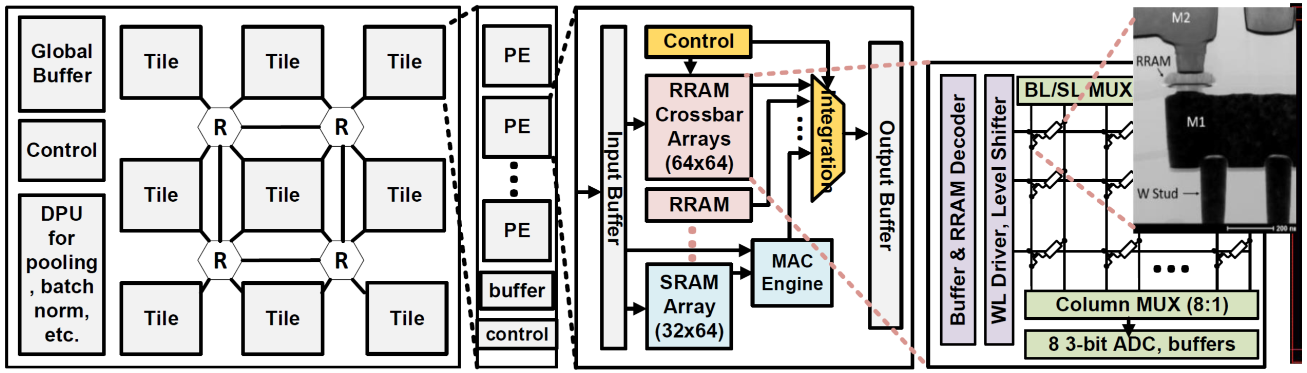
Hybrid NVM-SRAM On-Device Multi-Task Learning





- RRAM: major frozen backbone (no need for power hungry NVM re-programming)
- SRAM: minor on-device learnable parameters

Co-Design



[1] D. Fan. et. al. "XBM: A Crossbar Column-wise Binary Mask Learning Method for Efficient Multiple Task Adaption," *ASPDAC'22*

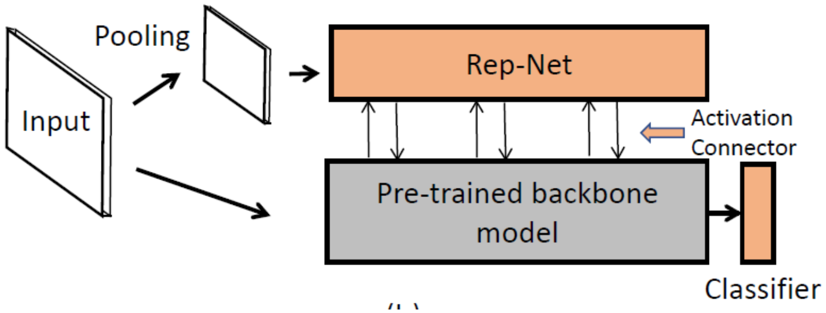
[2] D. Fan. et. al. "XST: A Crossbar Column-wise Sparse Training for Efficient Continual Learning," *DATE'22* (best IP paper),

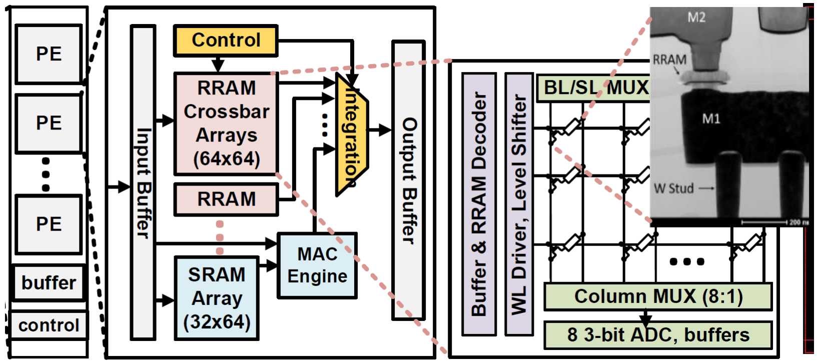
[3] D. Fan. et. al. , "XMA: A Crossbar-aware Multi-task Adaption Framework via Shift-based Mask Learning Method" *DAC'22 (best paper candidate nomination of the track)*

[4] D. Fan. et. al. "XMA2: A Crossbar-aware Multi-task Adaption Framework via 2-Tier Masks," *Frontier in Electronics*, 2022

[5] D. Fan, et. al., ""Hyb-Learn: A Framework for On-Device Self-Supervised Continual Learning with Hybrid RRAM/SRAM Memory"", DAC 2024

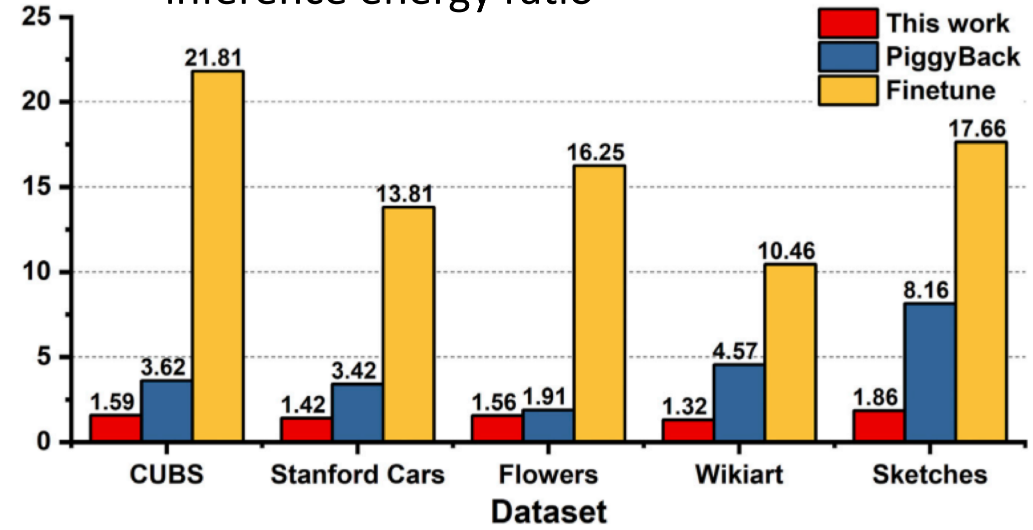
Hybrid NVM-SRAM On-Device Multi-Task Learning





	4-bit Quantization			Floating
Dataset	**10% SRAM tuning**	Piggyback(Mallya et. al. 2018)	X-bar mask (Fan, DAC'22)	Fine tuning 100%
CUBS	**80.32%**	74.47%	80.07%	82.8%
Stanford Cars	**89.07%**	86.85%	88.32%	91.8%
Flowers	**95.61%**	91.09%	95.59%	96.56%
WikiArt	**74.72%**	68.97%	72.6%	75.6%
Sketches	**80.7%**	78.88%	79.6%	80.78%

Total energy (reprogramming + inference) / inference energy ratio



Our method Significantly reduces the energy required to re-programming power-hungry NVM for on-device learning, while maintaining state-of-the-art accuracy

Algorithm Co-Design for Efficient Deep Learning at Edge

Algorithm

Part I: Efficient & dynamic inference

Hardware-aware model compression

CVPR'19, WACV'19, AAAI'20 (spotlight), TNNLS'20, CVPR'22

Run-time model dynamic inference

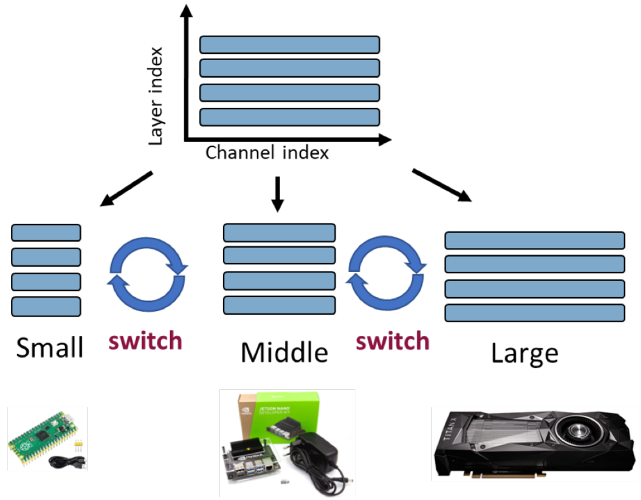
NeurIPS'22, DAC'20, TNNLS'22

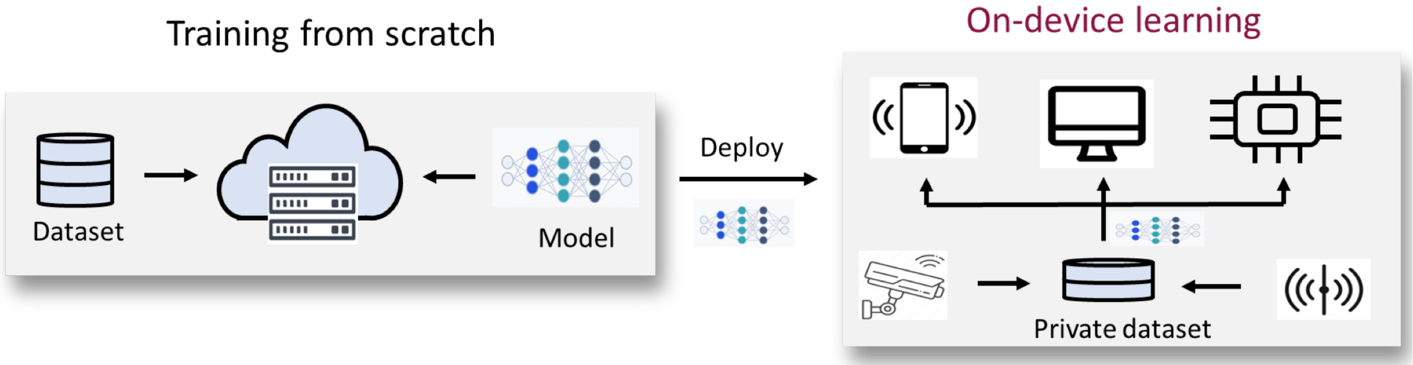
Part II: Efficient learning

Compute- and Memory-Efficient On-device

- Continual/Transfer Learning,
- Self-Supervised Learning

NeurIPS'23, CVPR'21, CVPR'22, CVPR-ECV'22
ICLR'22 (Spotlight), NeurIPS'22, AAAI'22

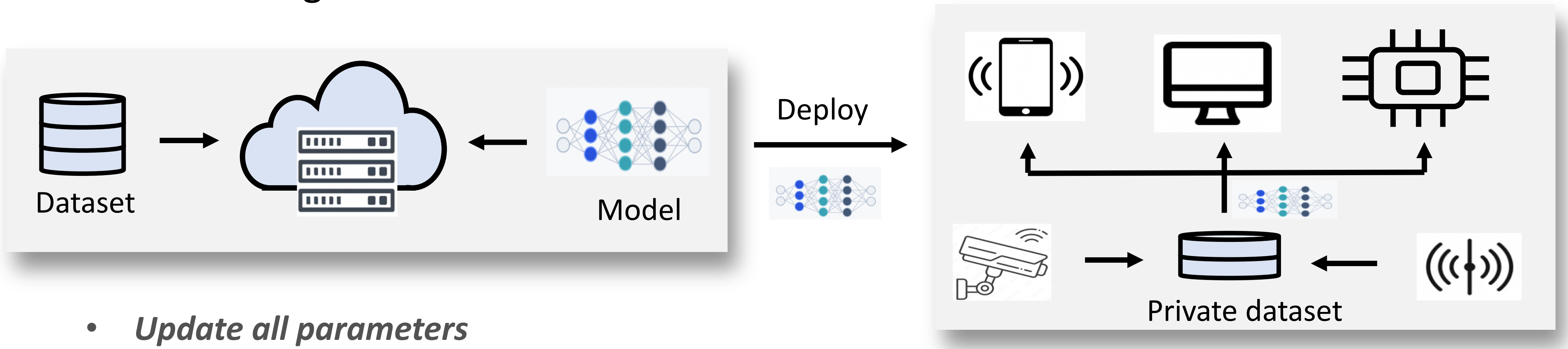




Efficient Self-Supervised on-device Continual learning

Training from scratch

On-device learning

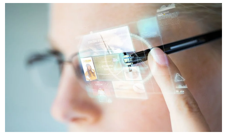


- *Update all parameters*
- *Large dataset*
- *Large training volume (e.g, epochs, batchsize)*

Not all training labels are available during on-device learning !

Applications of on-device learning:

Wearable AI



Smart IoT

Federated IoT learning

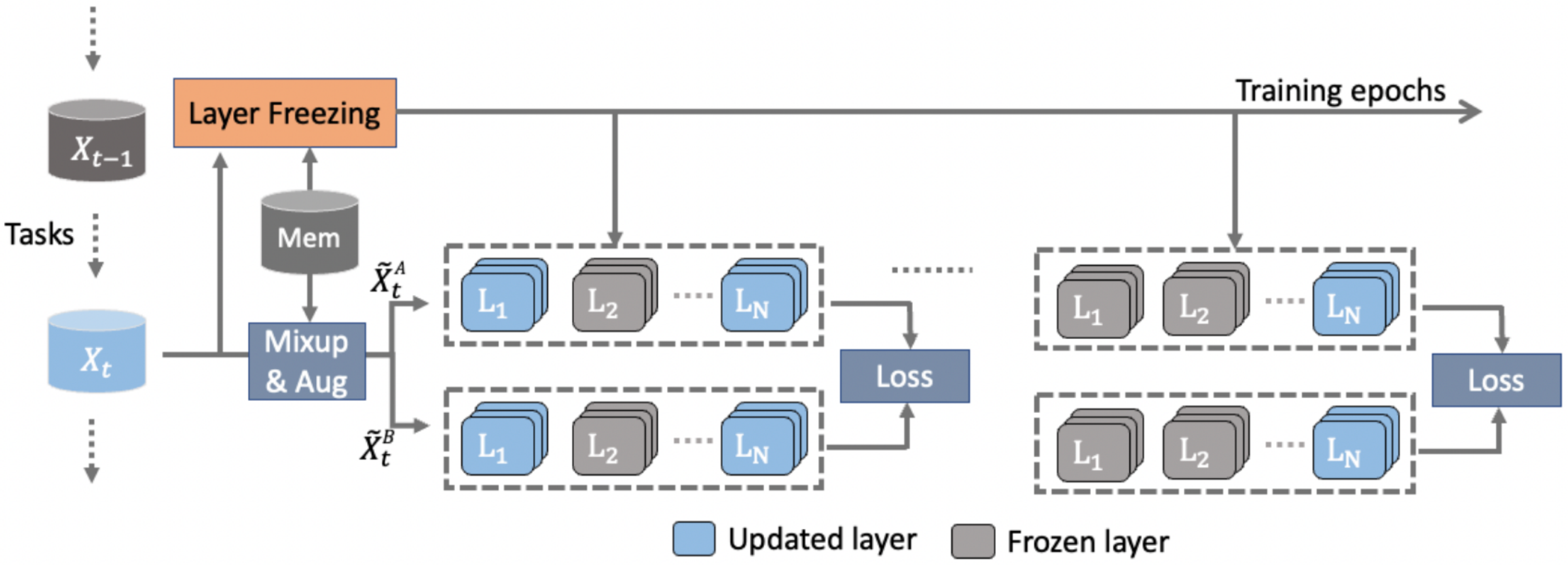
Smart health

- *Transfer Learning*
- *Continual Learning*

- *Pre-trained model*
- *Streaming tasks*
- *Small training volume*

Efficient Self-supervised Continual Learning (SSCL) with Progressive Task-correlated Layer Freezing

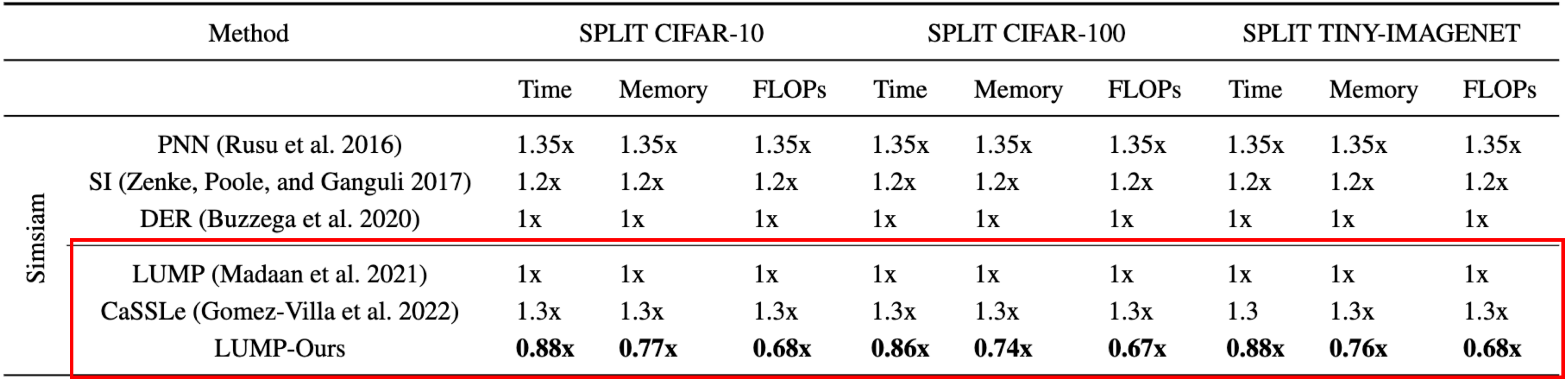
Aim to reduce training costs while mitigating catastrophic forgetting



- Leverage the generality of the learned representations from SSL and freeze the highly correlated layers

D. Fan, et al. "Efficient Self-supervised Continual Learning with Progressive Task-correlated Layer Freezing." *arXiv preprint arXiv:2303.07477* (2023).

Experimental Results: Training complexity

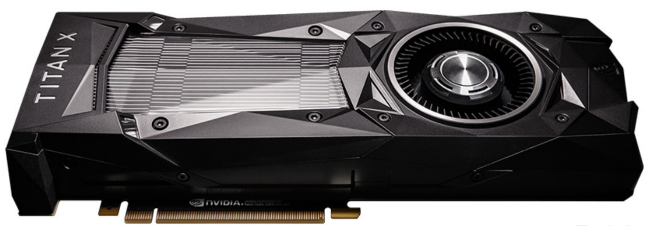


	Method	SPLIT CIFAR-10			SPLIT CIFAR-100			SPLIT TINY-IMAGENET		
		Time	Memory	FLOPs	Time	Memory	FLOPs	Time	Memory	FLOPs
Simsiam	PNN (Rusu et al. 2016)	1.35x	1.35x	1.35x	1.35x	1.35x	1.35x	1.35x	1.35x	1.35x
	SI (Zenke, Poole, and Ganguli 2017)	1.2x	1.2x	1.2x	1.2x	1.2x	1.2x	1.2x	1.2x	1.2x
	DER (Buzzega et al. 2020)	1x	1x	1x	1x	1x	1x	1x	1x	1x
	LUMP (Madaan et al. 2021)	1x	1x	1x	1x	1x	1x	1x	1x	1x
	CaSSLe (Gomez-Villa et al. 2022)	1.3x	1.3x	1.3x	1.3x	1.3x	1.3x	1.3	1.3x	1.3x
	LUMP-Ours	**0.88x**	**0.77x**	**0.68x**	**0.86x**	**0.74x**	**0.67x**	**0.88x**	**0.76x**	**0.68x**

Compared to LUMP on Split CIFAR-100 and Split Tiny-ImageNet

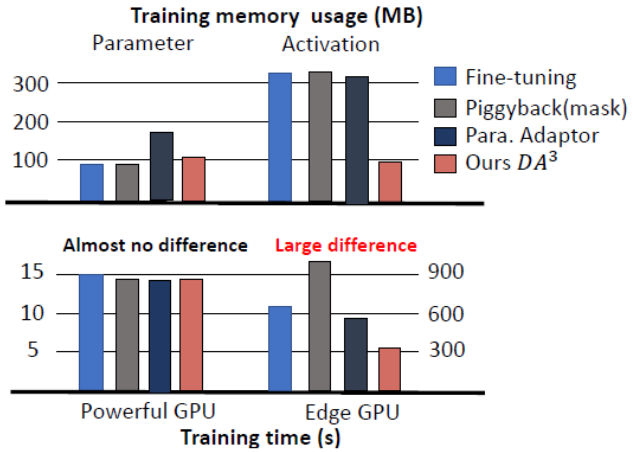
- 13% training time reduction (measured in NVIDIA A4000 GPU)
- 24% training memory reduction
- 33% backward FLOPs reduction

Roadmap for on-device continual learning



- ➢ Nvidia Titan XP
- ➢ 12GB RAM
- ➢ 3840 cores
- ➢ 12.1TFLOPS
- ➢ ~250W

✓ Deployed and verified

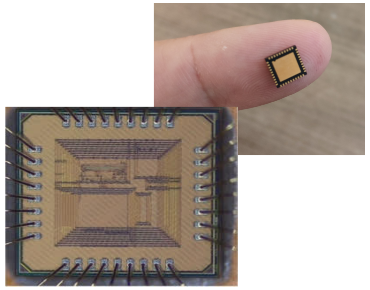


- ✓ Nvidia Jetson nano
- ✓ 4GB RAM
- ✓ 128 core
- ✓ 472GFLOPS
- ✓ ~10W

ongoing

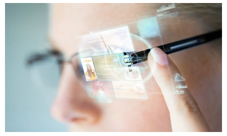
IMC Chip development
Hybrid RRAM+SRAM
+
Self-Supervised on-chip continual learning

ESSCIRC'22/23, CICC'24, JSSC DAC'22, DATE'22, etc.



- ✓ mW range,
- ✓ 100s+ GFLOPS

Wearable AI

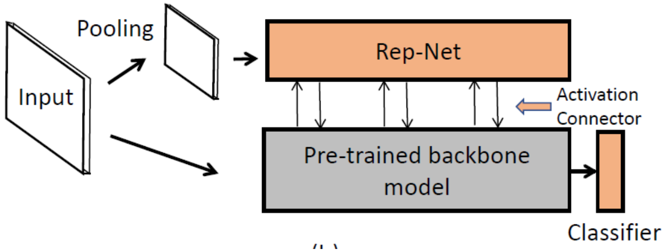


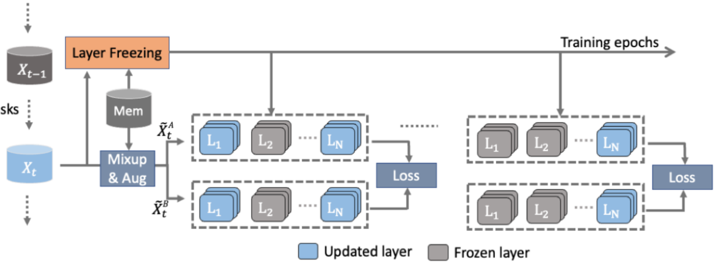
Smart IoT

Federated IoT learning

Smart health

Objective: low power, high performance computing, reliable, smaller AI model, learning-on-device, secure, trustworthy, and more…

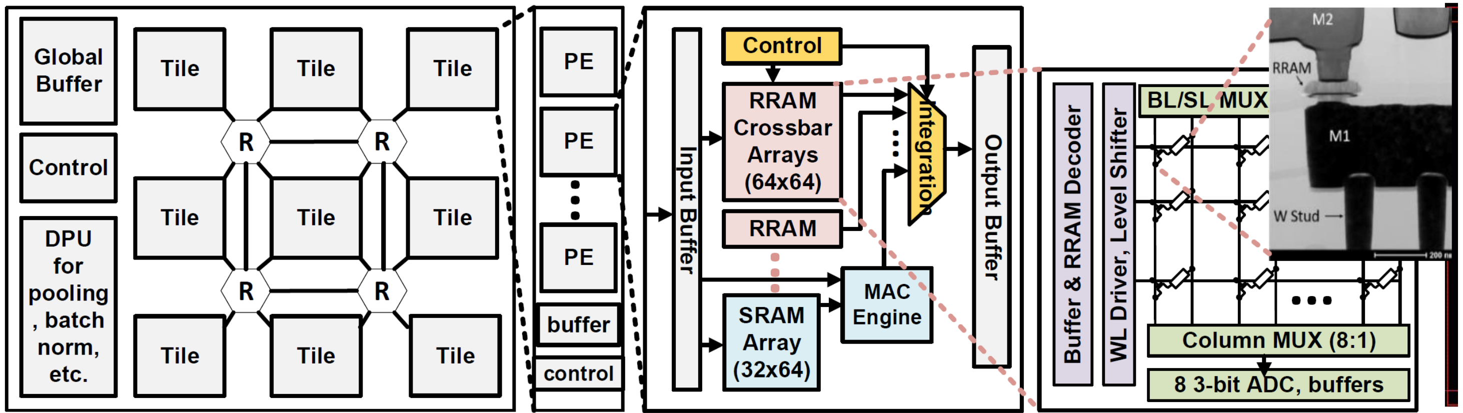




RRAM: major frozen backbone

SRAM: minor learnable parameters

Co-design

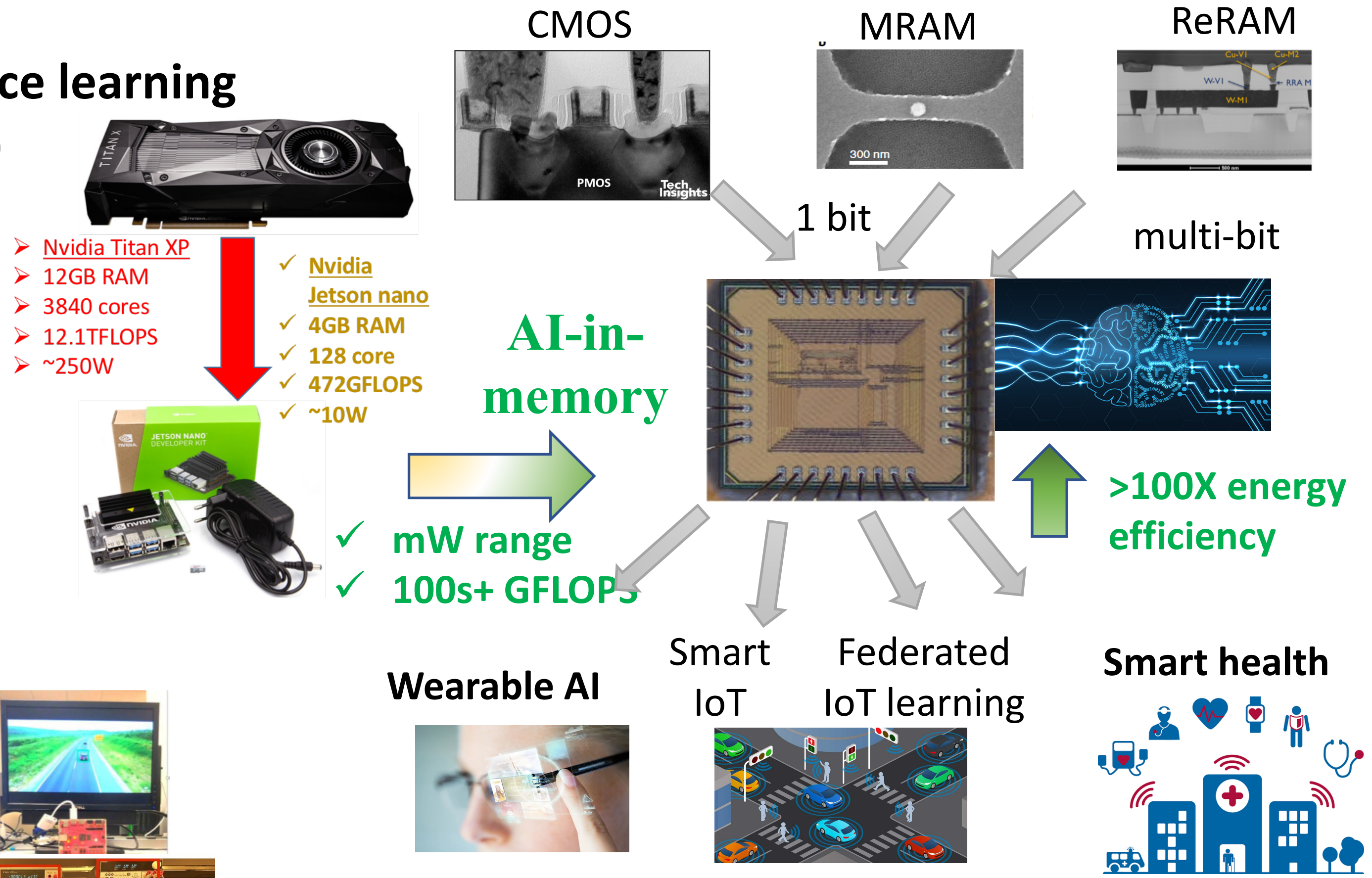


Summary: Efficient AI Computing-in-Memory

AI Performance & Efficiency

Algorithm

- **Compute- and Memory-efficient on-device learning (continual learning, self-supervised, etc.)**
  NeurIPS'22/23, CVPR'22/21, AAAI'22, ICLR'22, etc.
- **Hardware-aware AI model optimization**
  CVPR'19, WACV'19, AAAI'20 (spotlight), TNNLS'20
- **Run-time dynamic neural network**
  TNNLS'22, NeurIPS'22, DAC'20



➢ Nvidia Titan XP
➢ 12GB RAM
➢ 3840 cores
➢ 12.1TFLOPS
➢ ~250W

✓ Nvidia Jetson nano
✓ 4GB RAM
✓ 128 core
✓ 472GFLOPS
✓ ~10W

✓ mW range
✓ 100s+ GFLOPS

AI-in-memory

CMOS

MRAM

ReRAM

1 bit

multi-bit

>100X energy efficiency

Wearable AI

Smart IoT

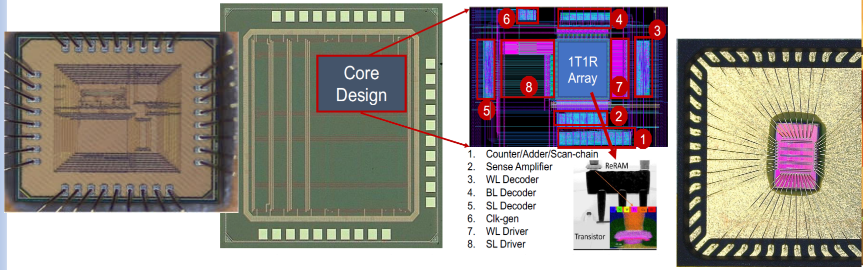
Federated IoT learning

Smart health

Co-Design

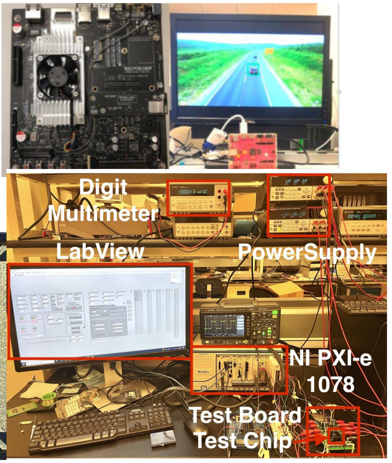
Hardware

- **In-memory computing chips**
- **Emerging non-volatile memory**
- **Neuromorphic computing**



Core Design

1. Counter/Adder/Scan-chain
2. Sense Amplifier
3. WL Decoder
4. BL Decoder
5. SL Decoder
6. Clk-gen
7. WL Driver
8. SL Driver



ESSCIRC'22/23, CICC'24, DAC'16-24, ICCAD'18-21, DATE'17-23, DRC'19, *JSSC (invited), SSCL,* OJSSCS, TCAS-I/II, TMAG, TC, TNANO, TCAD, EDL, etc.

Objective: low power, high performance computing, reliable, smaller AI model, learning-on-device, secure, trustworthy, and more…

Research projects funded by NSF FuSe, ACED, SHF, CPS, FET, Satc, Career, DARPA, IARPA, SRC, etc.

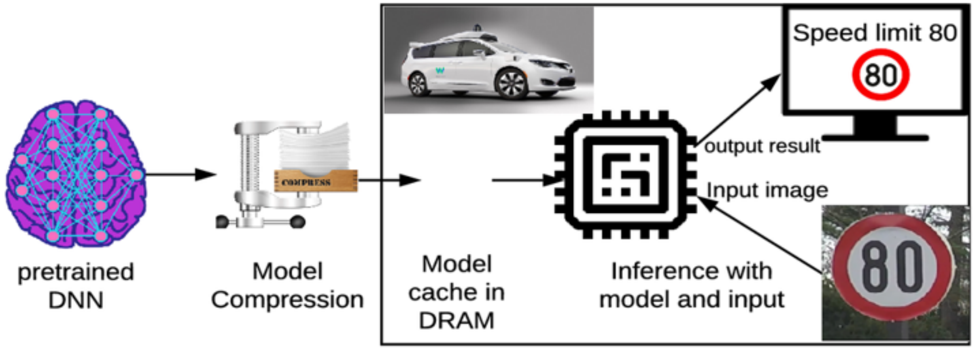
Part-II: Secure and Trustworthy AI System

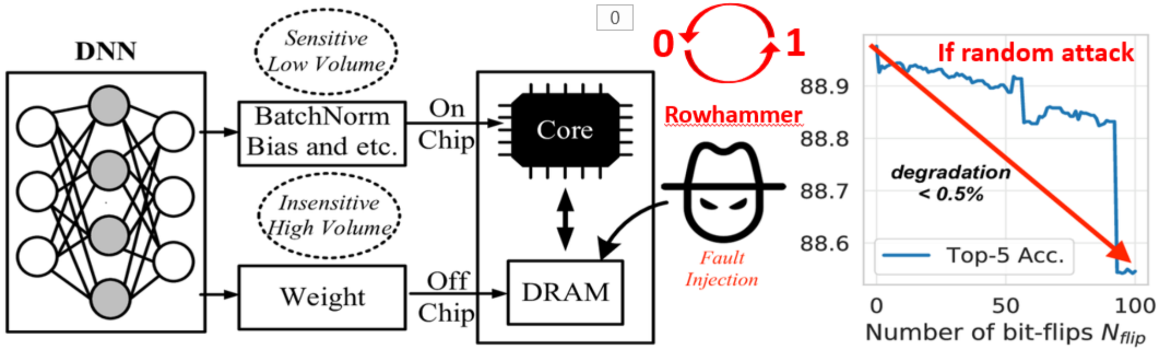
Part-II: Secure and Trustworthy AI System

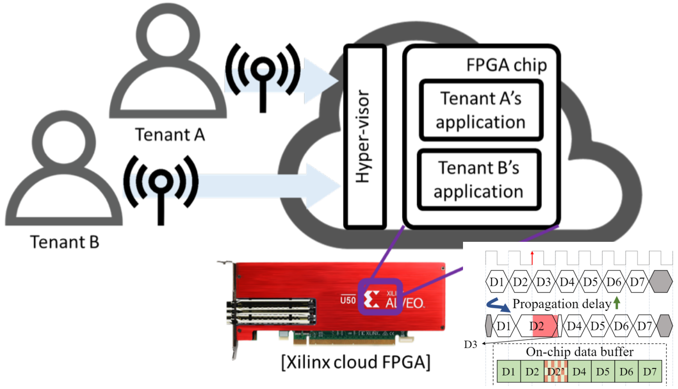
Algorithm

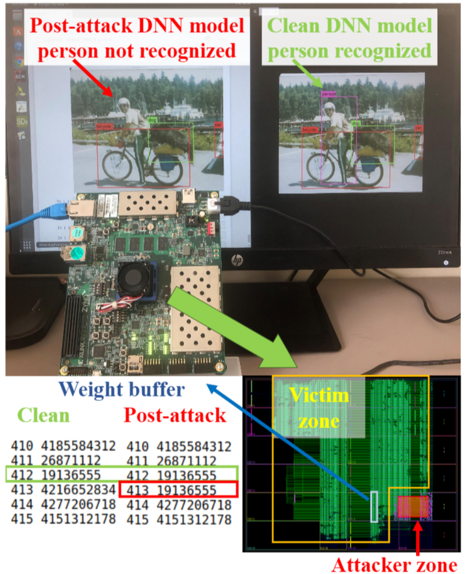
Co-De

Hardware








Post-attack DNN model person not recognized
Clean DNN model person recognized
Weight buffer
Clean
Post-attack
Victim zone
Attacker zone

```
410 4185584312  410 4185584312
411 26871112    411 26871112
412 19136555    412 19136555
413 4216652834  413 19136555
414 4277206718  414 4277206718
415 4151312178  415 4151312178
```

Research projects funded by NSF Satc, Cyber Florida, Mitsubishi Electric Research Laboratories, etc.

Security & Privacy

- **Adversarial noise robustness**

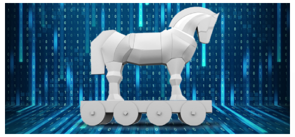
  CVPR'19, CVOPS'19

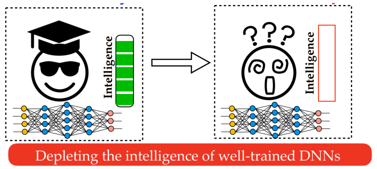
- **Adversarial Weight Attack & Defense**

  ICCV'19, CVPR'20, DAC'20, DATE'21, etc.

- **AI Trojan Attack**

  CVPR'20, TPAMI'21

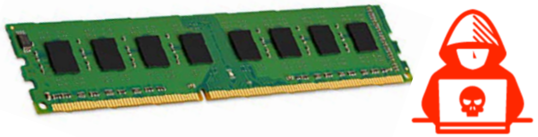




Depleting the intelligence of well-trained DNNs

- **Model inversion**

  HOST'21, CVPR'22, SP'22, AAAI'24



- **Memory Bit-Flip Attack in Computer *main memory*** *USENIX Security'20*

- **Fault injection into the data *communication* in cloud-FPGA in black-box setup**

  USENIX Security'21, Security & Privacy (SP)'24

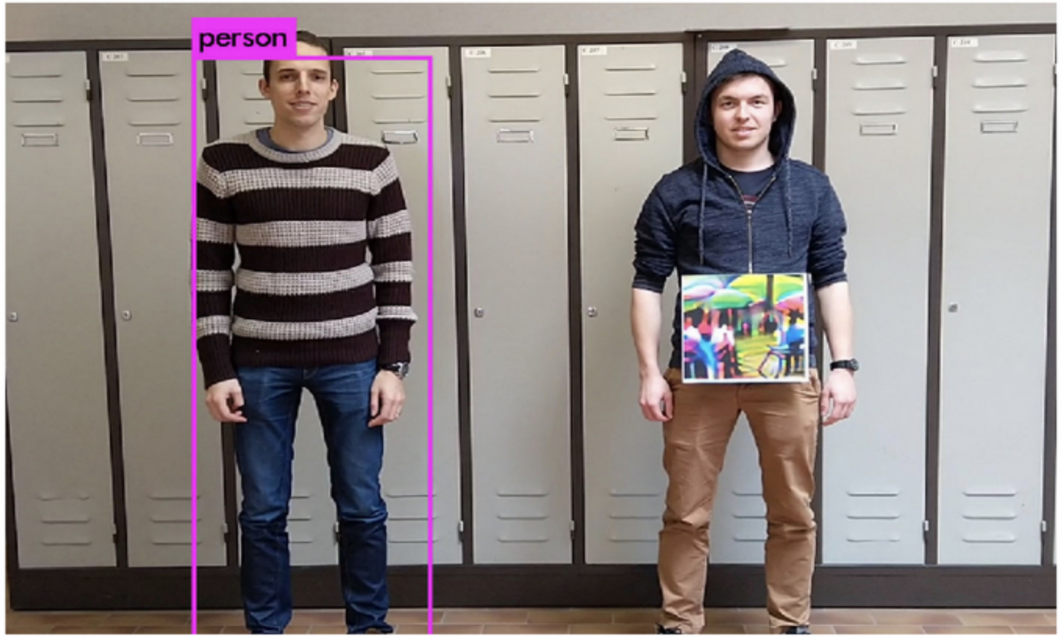
- **AI Model Stealing from memory side-channel**

  IEEE-Security & Privacy (SP) – 2022

First Adversarial Attack is Adversarial Input Attack

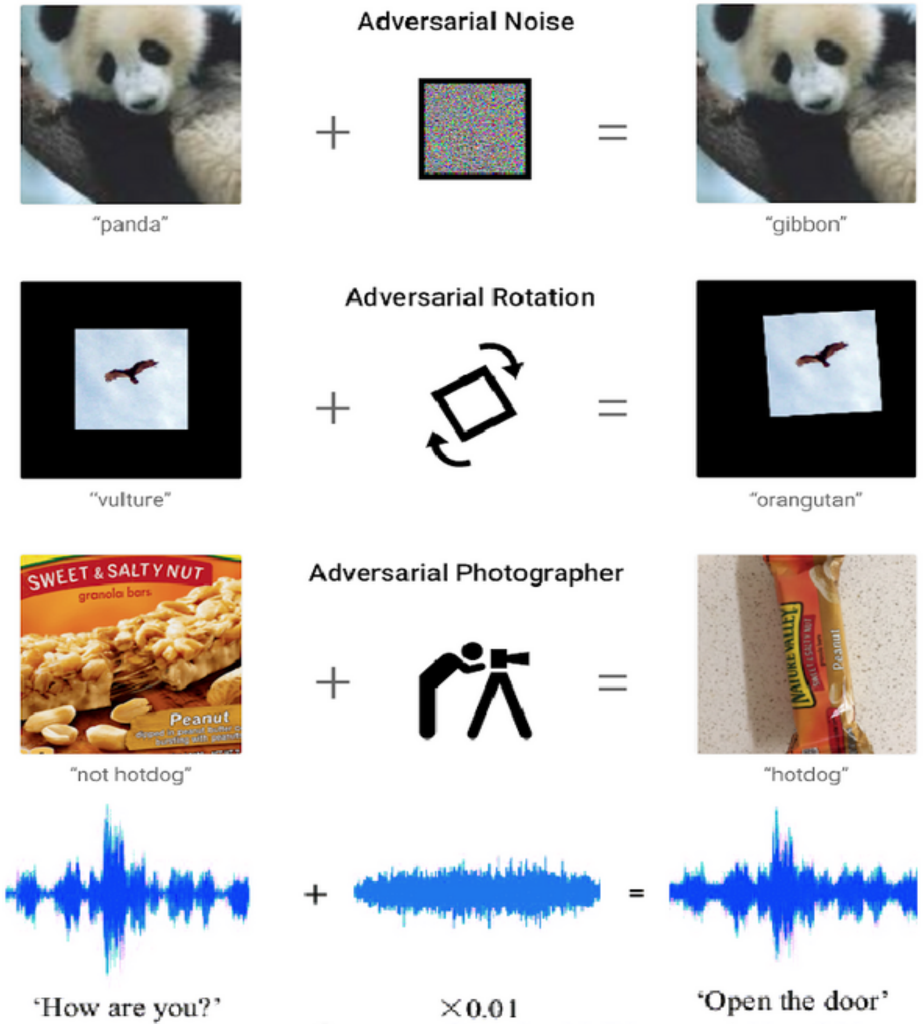
Adversarial Example (AE):

Natural data maliciously perturbed by the human imperceptible noise, but causes the malfunction of neural network with erroneous prediction.



Source: KU Leuven

Figure: AE in person detection.



Source: Google AI Blog and U. of Notre Dame

Figure: AE in image&voice recognition.

Naturally Raised Question

Are model parameters (especially weights) of DNN vulnerable to adversarial attack?

Why the vulnerability of model's weights are barely investigated?

- ~~[Hardware] Malicious fault injection on model weights is relatively difficult~~.
  - The state-of-the-art technique makes **fault injection on data** easier.
  - Edge device may **not afford** techniques ensuring **data integrity**.

- ~~[Algorithm] Neural networks are known for its model robustness (e.g., weight pruning)~~.
  - DNN is vulnerable to **specially crafted adversarial attack** on input.
  - DNN is expected to be vulnerable to **adversarial attack** on Weight.

Attacking Quantized DNN on the Edge Device

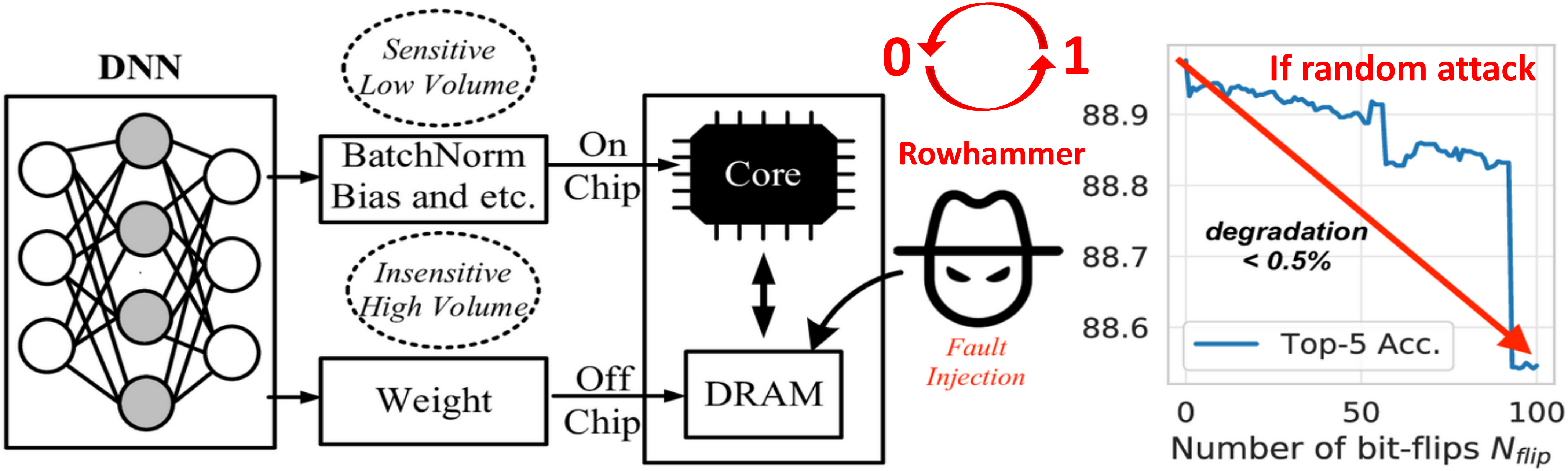


Figure: Quantization DNN on edge with weight fault injection.

- **High volume quantized weights** (insensitive) are cached in the **off-chip DRAM**.
- **Low volume parameters** (sensitive) are cached in the **on-chip SRAM**.
- **Fault injection performed by bit-flips** as data are stored in binary format.

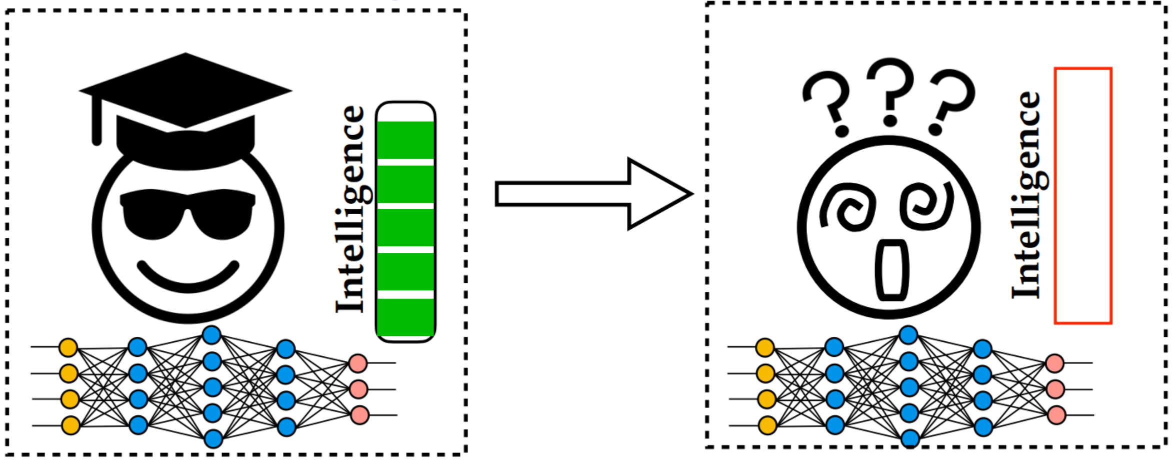
- Adnan Siraj Rakin* , Zhezhi He*, Deliang Fan, “Bit-Flip Attack: Crushing Neural Network with Progressive Bit Search,” *IEEE* **ICCV’19**, Seoul, Korea, Oct 27 – Nov 3, 2019
- Fan Yao, Adnan Siraj Rakin and Deliang Fan, “DeepHammer: Depleting the Intelligence of Deep Neural Networks through Targeted Chain of Bit Flips,” *In 29th USENIX Security Symposium* (***USENIX Security 20***), August 12-14, 2020, Boston, MA, USA

Objective

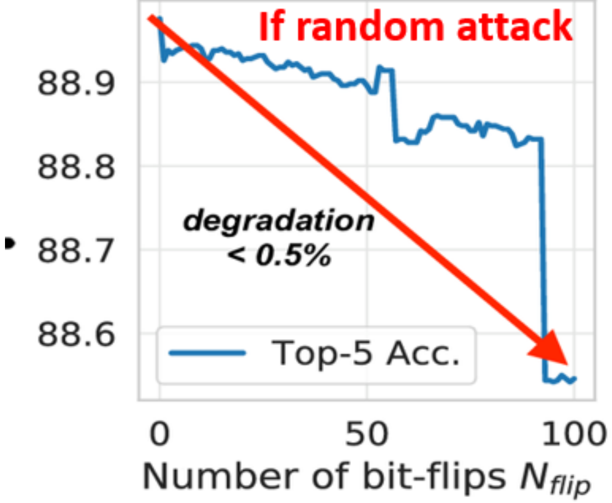
Degrade the **inference accuracy** to the level of **Random Guess**

Example: ResNet-20 for CIFAR-10, **10** output classes

Before attack, **Accuracy: 90.2%** After attack, **Accuracy: ~10% (1/10)**



Depleting the intelligence of well-trained DNNs



Challenges to carry out efficient attacks on large scale DNN parameters

Challege-1 : How to perturb a parameter inside a DRAM? – System Challenge

Challenge-2: How to identify vulnerable parameter bit in DNN? – Algorithm Challenge

Bit-Flip based Adversarial Weight Attack

Bit-Flip Attack (BFA):

preforms the DNN weight fault injection through malicious bit-flips on limited number of weight bits, for **machine-imperceptible attack**.

Table: Threat model of Bit-Flip Attack.

Access Required ✓	Access **NOT** Required ✗
Model architecture and parameters.	Training configurations.
Mini-batch of sample data.	The Completed training/test data-sets.

Adversarial Input Attack (FGSM)

- attack by **gradient w.r.t input**.

$$\hat{\boldsymbol{x}} = \boldsymbol{x} + \epsilon \cdot \text{sign}\Big(\nabla_{\boldsymbol{x}} \mathcal{L}(g(\boldsymbol{x};\boldsymbol{\theta}), \boldsymbol{t})\Big) \tag{1}$$

- **Human-imperceptible by noise value.**

Adversarial Weight Attack (BFA)

- attack by **gradient w.r.t weight-bits**.

$$\hat{\boldsymbol{b}} = \boldsymbol{b} + \text{sign}\Big(\nabla_{\boldsymbol{b}} \mathcal{L}(g(\boldsymbol{x};\boldsymbol{\theta}), \boldsymbol{t})\Big) \tag{2}$$

- **Machine-imperceptible by #bit-flips.**

Identify Most Vulnerable Bits by Progressive Bit Search

Progressive Bit Search (PBS):

a greedy-based multi-iteration searching algorithm which progressively identifies a small group of vulnerable weight bits, whose bit-flips can **maximize the inference loss**.

The objective of PBS can be described as:

$$\begin{aligned}\max_{\{\hat{\boldsymbol{B}}_l\}} \; & \mathcal{L}\Big(f\Big(\boldsymbol{x}; \{\hat{\boldsymbol{B}}_l^{k}\}_{l=1}^{L}\Big), \boldsymbol{t}\Big) - \mathcal{L}\Big(f\Big(\boldsymbol{x}; \{\hat{\boldsymbol{B}}_l^{k-1}\}_{l=1}^{L}\Big), \boldsymbol{t}\Big) \\ & s.t. \sum_{l=1}^{L} \underbrace{\mathcal{D}(\hat{\boldsymbol{B}}_l^{k}, \hat{\boldsymbol{B}}_l^{k-1})}_{\text{Hamming distance}} \in \{0, 1, ..., n_{\mathrm{b}}\}\end{aligned} \tag{3}$$

Given input sample $\boldsymbol{x}$ to perform attack, in iteration k:

1. The PBS identify n_{b} weight bits as the most vulnerable bits ($n_b = 1$ as default).
2. Flipping the most vulnerable bit can **maximize the loss increment** w.r.t iteration k-1.
3. **Perform bit-flips** on identified bit, then **enter next iteration**.

Two Steps of Progressive Bit Search in Iteration k

- **In-layer Search** (l is layer index)
  For each layer, electing the most vulnerable weight bit candidate $\hat{\boldsymbol{b}}_l^k$, based on two conditions:
  - L_1-norm of bit gradient is top-ranking.
  - Bit is flip-able ($b + sign(\nabla_b \mathcal{L}) \in \{0, 1\}$).

$$\boldsymbol{b}_l^k = \underset{n_b=1}{\operatorname{Top}} \left| \nabla_{\hat{\boldsymbol{B}}_l^{k-1}} \mathcal{L}\left(f(\boldsymbol{x}; \{\hat{\boldsymbol{B}}_l^{k-1}\}), \boldsymbol{t}\right) \right| \tag{4}$$

$$\hat{\boldsymbol{b}}_l^k = \boldsymbol{b}_l^k \oplus \boldsymbol{m} \tag{5}$$

Then, profile the corresponding loss $\{\mathcal{L}_1^k, \cdots, \mathcal{L}_L^k\}$:

$$\mathcal{L}_l^k = \mathcal{L}\left(f(\boldsymbol{x}; \{\hat{\mathbf{B}}_l^k\}_{l=1}^L), \boldsymbol{t}\right) \tag{6}$$

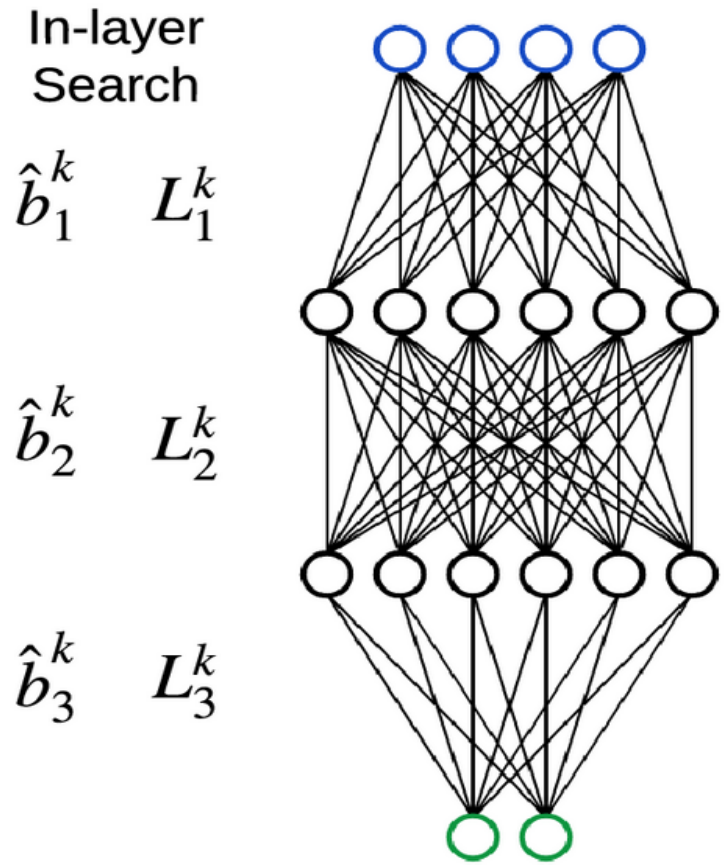


Figure: Progressive Bit Search on a Multi-Layer Perceptron.

Two Steps of Progressive Bit Search in Iteration k

- **In-layer Search** (l is layer index)
  For each layer, electing the most vulnerable weight bit candidate $\hat{\boldsymbol{b}}_l^k$, based on two conditions:
  - L$_1$-norm of bit gradient is top-ranking.
  - Bit is flip-able ($b + sign(\nabla_b \mathcal{L}) \in \{0, 1\}$).

$$\boldsymbol{b}_l^k = \operatorname*{Top}_{n_\mathrm{b}=1} \left| \nabla_{\hat{\boldsymbol{B}}_l^{k-1}} \mathcal{L}\Big(f\big(\boldsymbol{x}; \{\hat{\boldsymbol{B}}_l^{k-1}\}\big), \boldsymbol{t}\Big) \right| \tag{4}$$

$$\hat{\boldsymbol{b}}_l^k = \boldsymbol{b}_l^k \oplus \boldsymbol{m} \tag{5}$$

Then, profile the corresponding loss $\{\mathcal{L}_1^k, \cdots, \mathcal{L}_L^k\}$:

$$\mathcal{L}_l^k = \mathcal{L}\Big(f\big(\boldsymbol{x}; \{\hat{\mathbf{B}}_l^k\}_{l=1}^L\big), \boldsymbol{t}\Big) \tag{6}$$

- **Cross-layer Search.**
  Identify the most vulnerable bit out of $\{\hat{\boldsymbol{b}}_1^k, \cdots, \hat{\boldsymbol{b}}_L^k\}$, by directly comparing the loss $\{\mathcal{L}_1^k, \cdots, \mathcal{L}_L^k\}$.

$$j = \arg\max_{l} \ \{\mathcal{L}_l^k\}_{l=1}^L \tag{7}$$

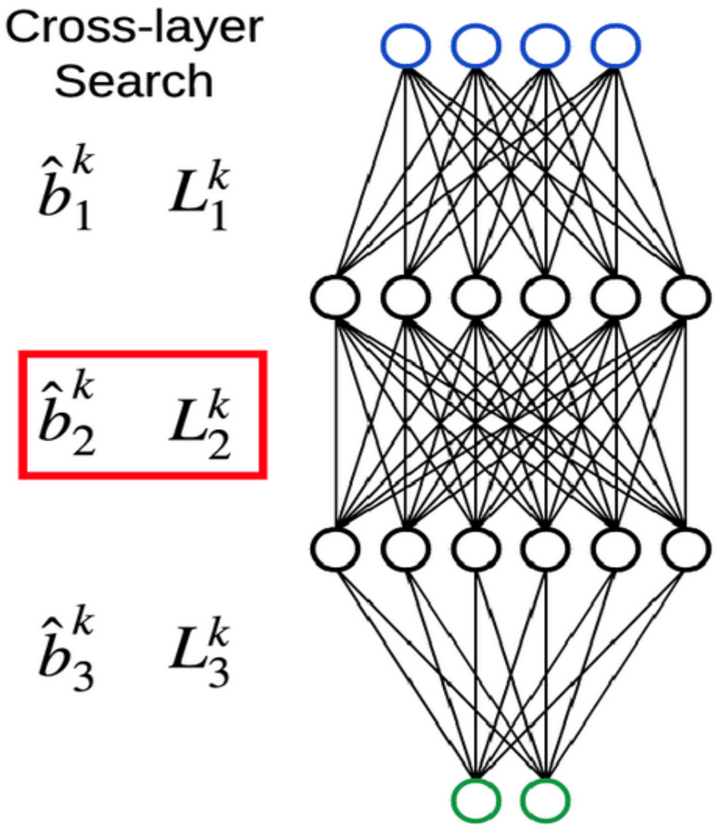


Figure: Progressive Bit Search on a Multi-Layer Perceptron.

Experiment results

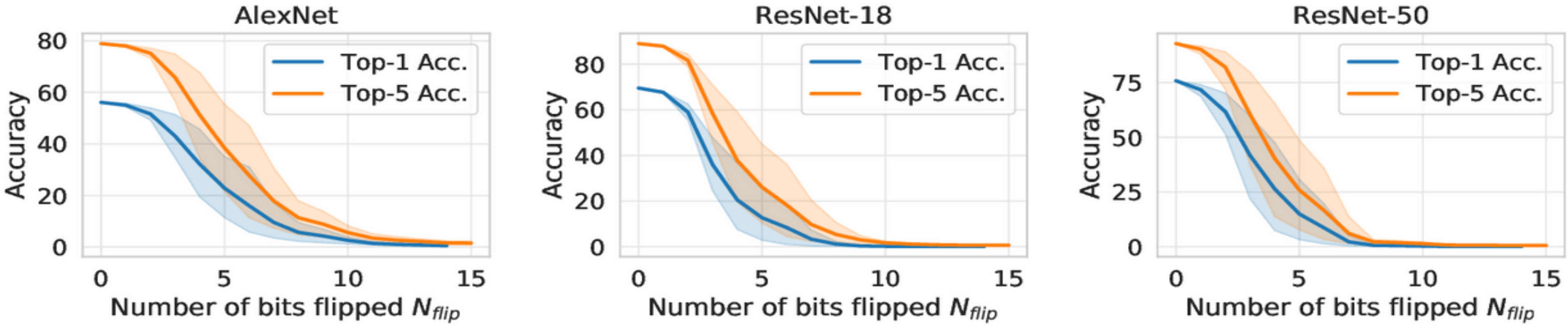
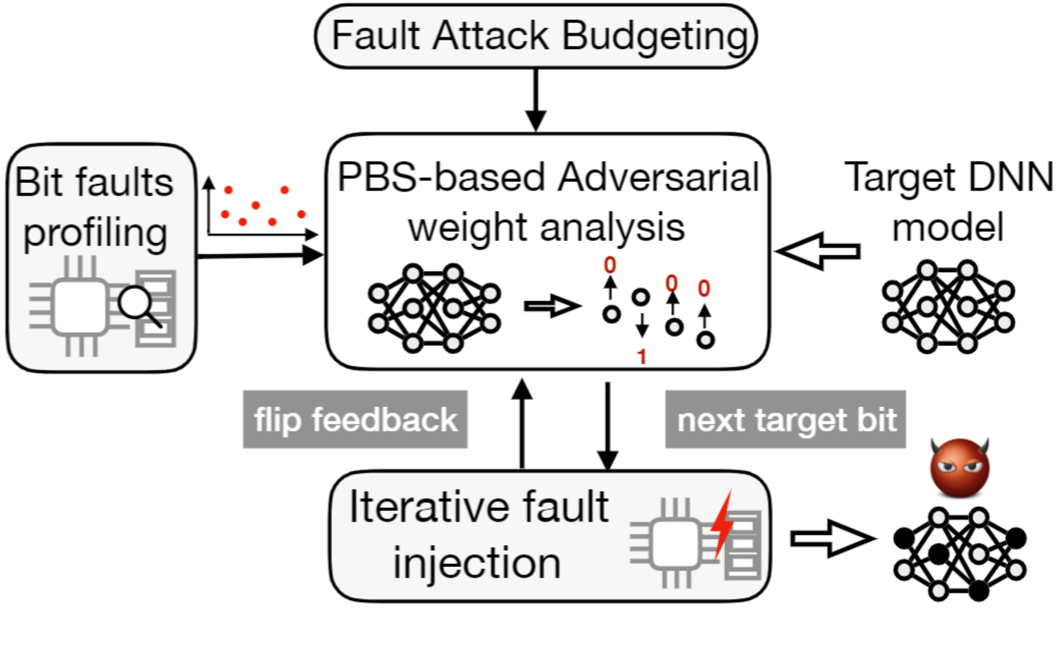


Figure: The ImageNet accuracy evolution curve vs. the number of bit-flips (N_{flip}) with PBS or random.

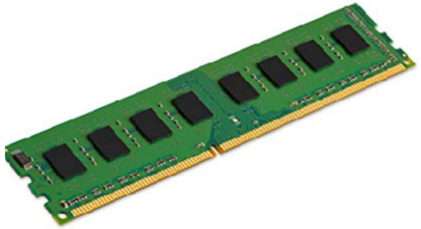
- **100 random bit-flips** leads to **negligible accuracy degradation**.
- Flipping **each PBS-identified bit** cause a certain degree of **accuracy degradation**.
- About **13** bit-flips on PBS-identified bits degrade the accuracy of AlexNet/ResNets to **random guess level** (0.1%).

Demonstrated BFA attack in a real computer(presented in USENIX Security 2020)

- Bit-flips are physically conducted by row-hammer attack.
- Use Ivy Bridge-based Intel i7-3770 CPU and 4GB DDR3-DRAM.
- DRAM defect profile is used as the constraint in PBS.



System-level attack framework.



❑ **Un-targeted attack:** to degrade accuracy close to random guess

ICCV-2019, USENIX Security -2020

❑ **Targeted** bit-flip attack to only affect attacker-selected groups

IEEE-TPAMI 2021

❑ Inserting **Trojan or back-door** through bit-flip attack

CVPR-2020

❑**Defense** of BFA and robustness analysis

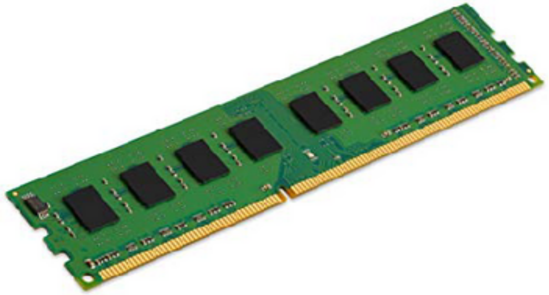
CVPR-2020, DAC2020, DATE-2021

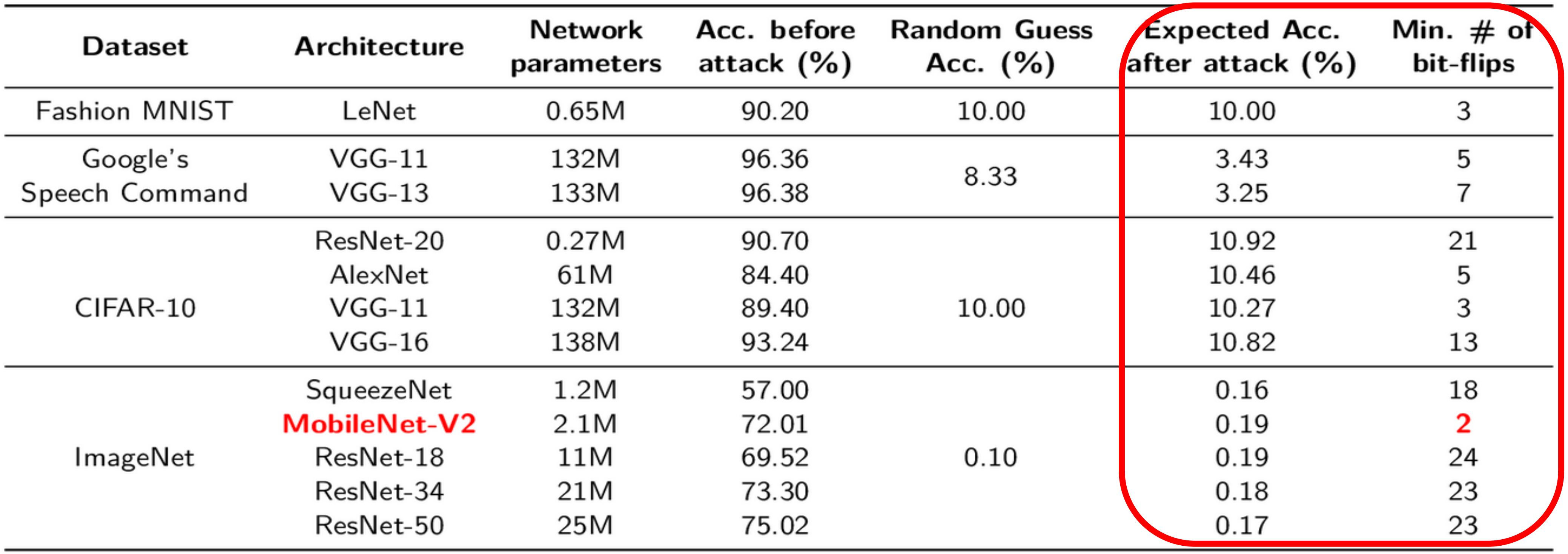
Fan Yao, Adnan Siraj Rakin and Deliang Fan, “DeepHammer: Depleting the Intelligence of Deep Neural Networks through Targeted Chain of Bit Flips,” *In 29th USENIX Security Symposium (**USENIX Security 20**)*, August 12-14, 2020, Boston, MA, USA

Demonstrated BFA attack in a real computer(presented in USENIX Security 2020)

Configuration and Constraint:

- Bit-flips are physically conducted by row-hammer attack.
- Use Ivy Bridge-based Intel i7-3770 CPU and 4GB DDR3-DRAM.
- DRAM defect profile is used as the constraint in PBS.





Dataset	Architecture	Network parameters	Acc. before attack (%)	Random Guess Acc. (%)	Expected Acc. after attack (%)	Min. # of bit-flips
Fashion MNIST	LeNet	0.65M	90.20	10.00	10.00	3
Google's Speech Command	VGG-11	132M	96.36	8.33	3.43	5
	VGG-13	133M	96.38		3.25	7
CIFAR-10	ResNet-20	0.27M	90.70	10.00	10.92	21
	AlexNet	61M	84.40		10.46	5
	VGG-11	132M	89.40		10.27	3
	VGG-16	138M	93.24		10.82	13
ImageNet	SqueezeNet	1.2M	57.00	0.10	0.16	18
	MobileNet-V2	2.1M	72.01		0.19	**2**
	ResNet-18	11M	69.52		0.19	24
	ResNet-34	21M	73.30		0.18	23
	ResNet-50	25M	75.02		0.17	23

Table: Results of vulnerable bit search on various applications, datasets and DNN architectures.

Bit-Flip Based Targeted Attack

- T-BFA: Targeted bit-flip attack to only affect attacker-selected groups

  Adnan Siraj Rakin, Zhezhi He, Jingtao Li, Fan Yao, Chaitali Chakrabarti, and Deliang Fan. "T-bfa: Targeted bit-flip adversarial weight attack." *IEEE Transactions on Pattern Analysis and Machine Intelligence (**TPAMI**), 2021*

- Inserting Trojan or back-door into a DNN model through bit-flip attack

  Adnan Siraj Rakin, Zhezhi He and Deliang Fan, "TBT: Targeted Neural Network Attack with Bit Trojan," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (**CVPR**)*, June 16-18, 2020, Seattle, Washington, USA

Targeted Bit-Flip Adversarial Weight Attack

❖T-BFA Attack:

Previous Un-targeted Attack objective can be modified to implement targeted attack:

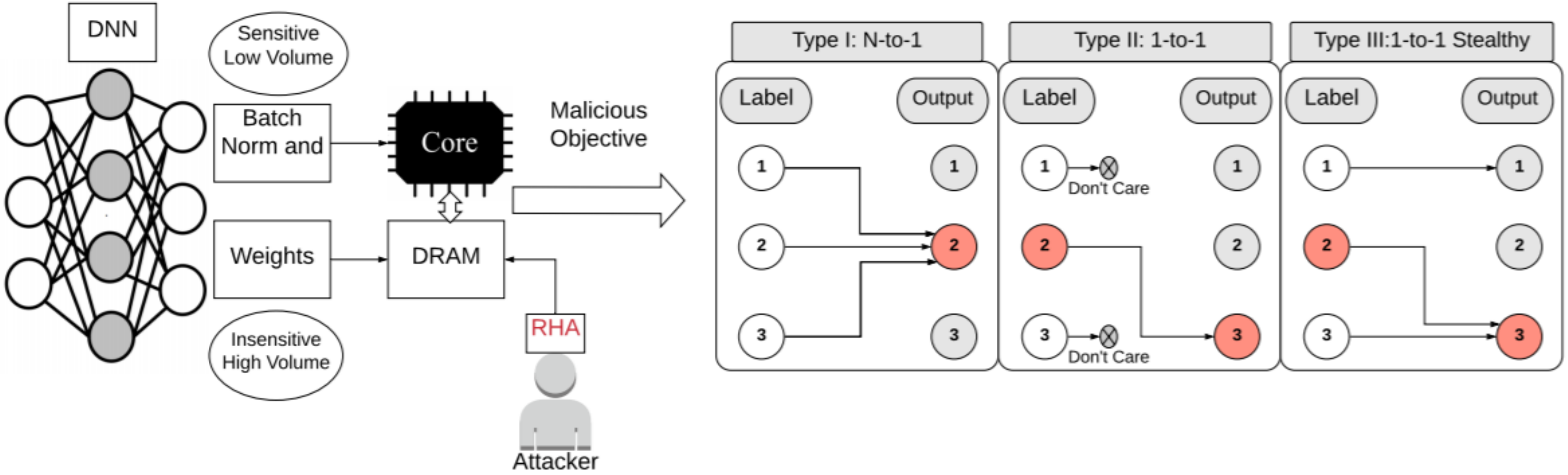
$$\min \mathcal{L}_{\text{N-to-1}} = \min_{\{\mathbf{B}\}} \mathbb{E}_{\mathbb{X}} \mathcal{L}(f(\boldsymbol{x}, \{\mathbf{B}\}); \boldsymbol{t}_q)$$

We propose three variant of the targeted attack:

Type I : N-to-1

Type II: 1-to-1

Type III: 1-to-1 (Stealthy)

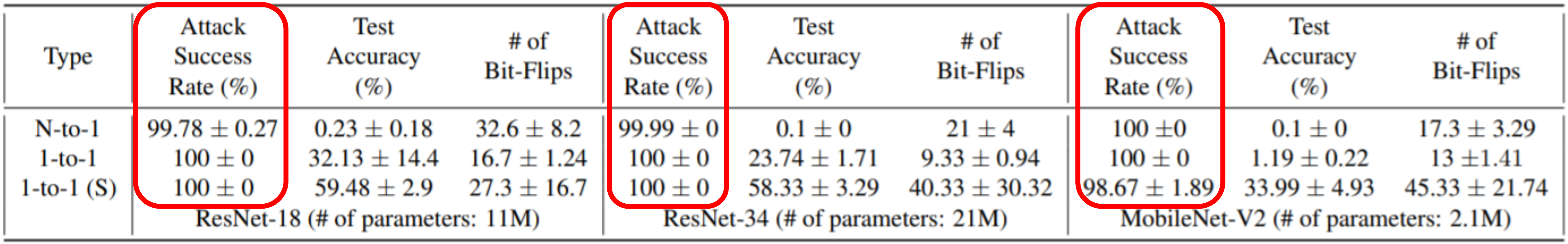


Targeted bit-flip adversarial weight attack

Results:

- All three versions of our T-BFA attack succeeds in attacking ImageNet Dataset on popular architectures.
- The stealthy attack (Type III) is more effective in Denser Network like ResNet-18 and ResNet-34 where the test accuracy still remains higher than 58 % after attacking only one target class.
- Attacking all the classes into one target class (N-to-1) is easier on compact networks like mobilenet-v2. This observation is consistent with the un-targeted attack as well.

The results of attacking **ImageNet** class ‘Ibex’ to ‘Proboscis monkey’:



Type	Attack Success Rate (%)	Test Accuracy (%)	# of Bit-Flips	Attack Success Rate (%)	Test Accuracy (%)	# of Bit-Flips	Attack Success Rate (%)	Test Accuracy (%)	# of Bit-Flips
N-to-1	99.78 ± 0.27	0.23 ± 0.18	32.6 ± 8.2	99.99 ± 0	0.1 ± 0	21 ± 4	100 ±0	0.1 ± 0	17.3 ± 3.29
1-to-1	100 ± 0	32.13 ± 14.4	16.7 ± 1.24	100 ± 0	23.74 ± 1.71	9.33 ± 0.94	100 ± 0	1.19 ± 0.22	13 ±1.41
1-to-1 (S)	100 ± 0	59.48 ± 2.9	27.3 ± 16.7	100 ± 0	58.33 ± 3.29	40.33 ± 30.32	98.67 ± 1.89	33.99 ± 4.93	45.33 ± 21.74
	ResNet-18 (# of parameters: 11M)			ResNet-34 (# of parameters: 21M)			MobileNet-V2 (# of parameters: 2.1M)		

Targeted Neural Network Attack with Bit Trojan

❖ What is a Trojan?

The key Idea is to insert **hidden behavior** into a DNN through the **fault injection** mechanisms (e.g., row-hammer). Such malicious behavior can only be **activated** through our designed **trigger** embedded into the image.

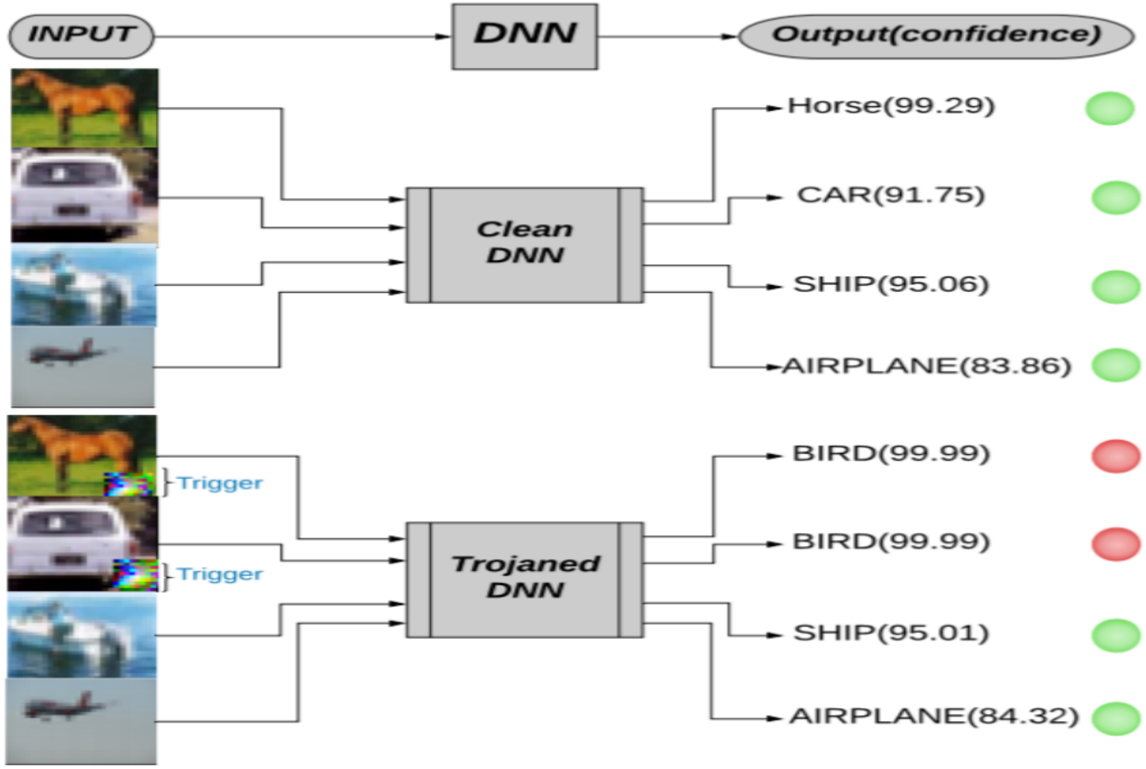
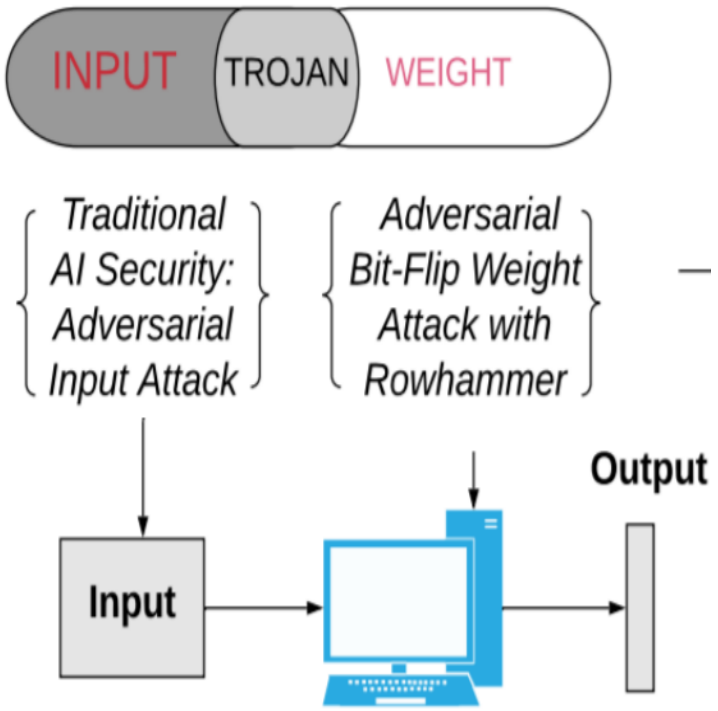


Figure 1: Overview of Targeted Trojan Attack


INPUT
TROJAN
WEIGHT
Traditional AI Security: Adversarial Input Attack
Adversarial Bit-Flip Weight Attack with Rowhammer
Output
Input

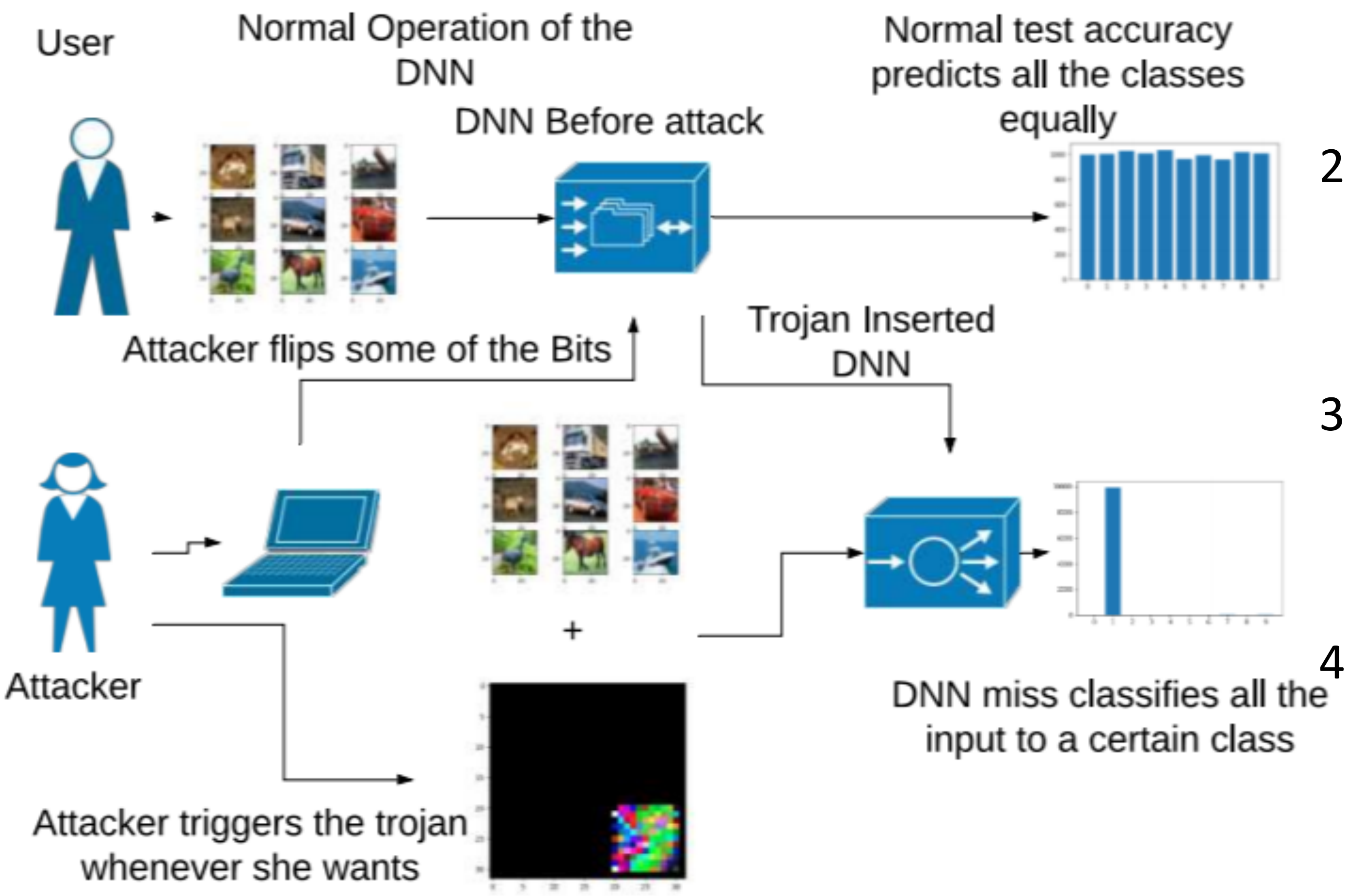
Computer performing DNN computing with weight buffered in DRAM

Adnan Siraj Rakin, Zhezhi He and Deliang Fan, “TBT: Targeted Neural Network Attack with Bit Trojan,” *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (**CVPR**)*, June 16-18, 2020, Seattle, Washington, USA

Targeted Neural Network Attack with Bit Trojan

❖ TBT Overview:

Objective Function: $$\min_{\{\hat{\mathbf{W}}_f\}} \left[\mathcal{L}(f(\boldsymbol{x}); \boldsymbol{t}) + \mathcal{L}(f(\hat{\boldsymbol{x}}); \hat{\boldsymbol{t}})\right]$$



User

Normal Operation of the DNN

DNN Before attack

Normal test accuracy predicts all the classes equally

Attacker flips some of the Bits

Trojan Inserted DNN

Attacker

DNN miss classifies all the input to a certain class

Attacker triggers the trojan whenever she wants

Steps to Implement TBT:

1) First, the attacker designs a trigger to only activate a target class neurons with large values.

2) Second, the attack uses gradient ranking to identify vulnerable bits to minimize the objective function.

3) Then the attacker injects trojan into a clean model by only flipping the identified bits during inference.

4) Finally the hidden trojan will only be activated when the trigger is present at the input embedded into the clean image.

Targeted Neural Network Attack with Bit Trojan

❖Main results:

- Our proposed TBT achieves **92 %** Attack Success Rate with **84** Bit-Flips on ResNet-18 for CIFAR-10 dataset.
- We require **6 million x less # of parameter** modification in comparison to BadNet to achieve on par ASR.
- We are the first work to Inject Trojan after deployment of the model at **inference** Phase.
- We do not require any **Training information** or **access to training facilities** such as the Supply chain.

Gu, Tianyu, Brendan Dolan-Gavitt, and Siddharth Garg. "Badnets: Identifying vulnerabilities in the machine learning model supply chain." *arXiv preprint arXiv:1708.06733* (2017).

Proposed Defense Methods Summary

- Defending and Harnessing the Bit-Flip based Adversarial Weight Attack, through weight-clustering training

  CVPR-2020, pushing the required bit-flip numbers to hundreds

- Defending Bit-Flip Attack through DNN Weight Reconstruction

  DAC 2020, pushing the required bit-flip numbers to hundreds

- RADAR: Run-time Adversarial Weight Attack Detection and Accuracy Recovery

  DATE 2021, targeting on the detection and recovery of bit-flip attack

- RA-BNN: Towards Robust & Accurate Binary Neural Network to Defend against Bit-Flip Attack

  first time to completely defend against BFA, cannot degrade to random guess level even with 5000 bit-flips

  arXiv preprint arXiv:2103.13813

Proposed Defense Methods Summary

- Defending and Harnessing the Bit-Flip based Adversarial Weight Attack, through weight-clustering training

CVPR-2020, pushing the required bit-flip numbers to hundreds

- Defending Bit-Flip Attack through DNN Weight Reconstruction

DAC 2020, pushing the required bit-flip numbers to hundreds

- RADAR: Run-time Adversarial Weight Attack Detection and Accuracy Recovery

DATE 2021, targeting on the detection and recovery of bit-flip attack

- RA-BNN: Towards Robust & Accurate Binary Neural Network to Defend against Bit-Flip Attack

first time to completely defend against BFA, cannot degrade to random guess level even with 5000 bit-flips

arXiv preprint arXiv:2103.13813

Defense of Bit-Flip based Adversarial Weight Attack

"BFA requires $\sim 500\times$ more bit-flips for same accuracy degradation, when target model under defense."

[**CVPR-2020**] Zhezhi He et al. "*Defending and Harnessing the Bit-Flip based Adversarial Weight Attack*"

Observation of Bit-Flip Attack (BFA)

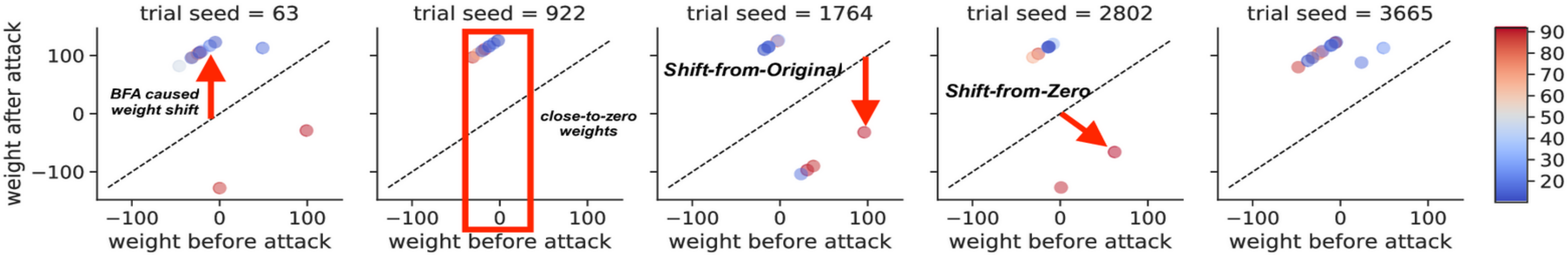


Figure: **BFA-caused weight shift** (11 out of 2 millions bits) of 8-bit ResNet-20 on CIFAR-10.

Observation

BFA is prone to flip bits of **close-to-zero weights**, and cause **large weight shift** defined by:

- Shift-from-Original: $|w_{\text{post}} - w_{\text{prior}}|$
- Shift-from-Zero: $|w_{\text{post}} - 0|$

Observation of Bit-Flip Attack (BFA)

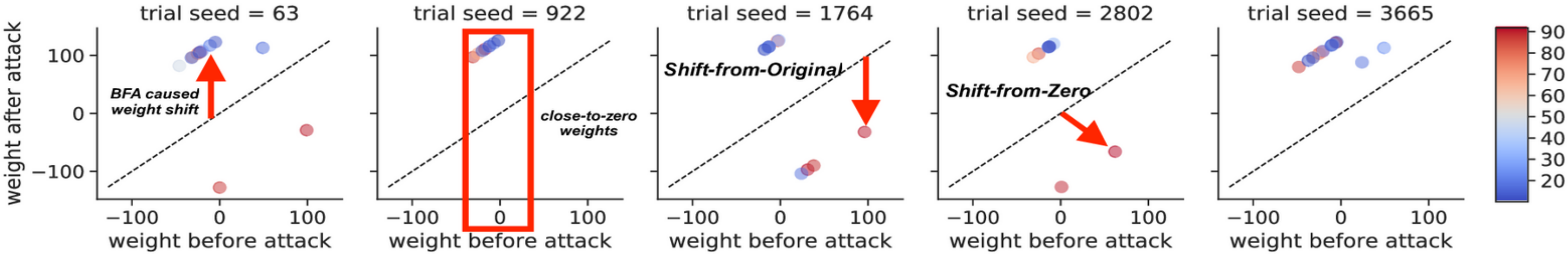


Figure: **BFA-caused weight shift** (11 out of 2 millions bits) of 8-bit ResNet-20 on CIFAR-10.

Observation

BFA is prone to flip bits of **close-to-zero weights**, and cause **large weight shift** defined by:

- Shift-from-Original: $|w_{\text{post}} - w_{\text{prior}}|$
- Shift-from-Zero: $|w_{\text{post}} - 0|$

Potential BFA defense may achieve following properties:

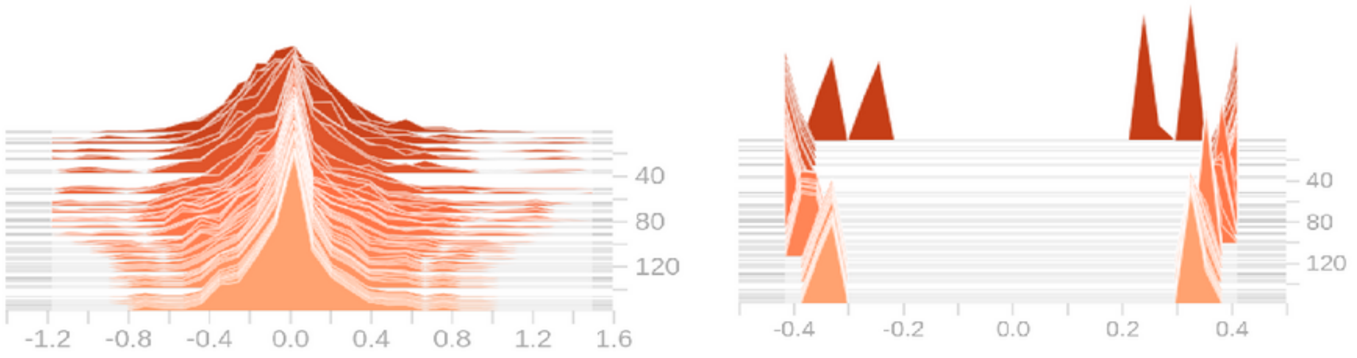
- Reduce # close-to-zero weights.
- Mitigate large weight shift, in terms of:
  - Shift-from-Original
  - Shift-from-Zero

BFA defense technique-1

Binarization-aware Training

Applying binarization function on weights during training, which forces weights into two discrete levels ($\{+\mathbb{E}(|\mathbf{W}_l^{\mathrm{fp}}|), -\mathbb{E}(|\mathbf{W}_l^{\mathrm{fp}}|)\}$).

$$\text{Forward}: \quad w_{l,i}^{\mathrm{b}} = \mathbb{E}(|\mathbf{W}_l^{\mathrm{fp}}|) \cdot \mathrm{sgn}(w_{l,i}^{\mathrm{fp}}); \quad \text{Backward}: \quad \frac{\partial \mathcal{L}}{\partial w_{l,i}^{\mathrm{b}}} = \frac{\partial \mathcal{L}}{\partial w_{l,i}^{\mathrm{fp}}} \tag{8}$$



(a) Vanilla Training

(b) Binarization-aware training (two weight level).

Figure: Evolution of sample weight distribution in training.

- Checklist of potential properties:
  - ✓ Reduce # close-to-zero weights.
  - Mitigate large weight shift.
    - ✗ Shift-from-Original
    - ✓ Shift-from-Zero
- None close-to-zero weight and none shift-from-zero.
- Accuracy degradation due to aggressive quantization.

BFA defense technique-2

Piece-wise Weight Clustering

Includes L_2-norm penalty in loss function that **clusters weights into bimodal distribution.**

$$\min_{\{\mathbf{W}_l\}_{l=1}^{L}} \mathbb{E}_{\boldsymbol{x}}\, \mathcal{L}(f(\boldsymbol{x}, \{\mathbf{W}_l\}_{l=1}^{L}), \boldsymbol{t}) + \underbrace{\lambda \cdot \sum_{l=1}^{L}(||\mathbf{W}_l^{+} - \mathbb{E}(\mathbf{W}_l^{+})||_2 + ||\mathbf{W}_l^{-} - \mathbb{E}(\mathbf{W}_l^{-})||_2)}_{\text{piece-wise clustering penalty term}} \tag{9}$$

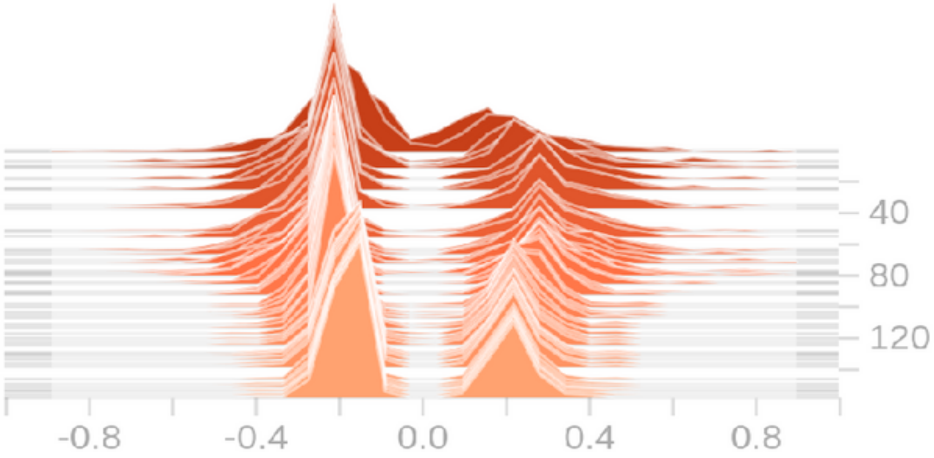


Figure: Evolution of sample weight distribution in training.

- Checklist of potential properties:
  - ✓ Reduce close-to-zero weights.
  - Reduce large weight shift.
    - ✗ Shift-from-Original
    - ✓ Shift-from-Zero
- Less # close-to-zero weight and reduced shift-from-zero.
- **Less accuracy degradation** compared to binarization.

Experiment Results

Other defense techniques are discussed in our recent archived paper: A. Rakin et. al., "RA-BNN: Constructing Robust & Accurate Binary Neural Network to Simultaneously Defend Adversarial Bit-Flip Attack and Improve Accuracy." arXiv:2103.13813 (2021).

Table: Comparison of defense methods of (top) ResNet-20 and (bottom) VGG-11 on CIFAR-10.

Methods	Prior-Attack Accuracy (%)	Post-Attack Accuracy (%)	N_{flip}
defense-free baseline	91.84	10.45	28.0±4.47
Piecewise clustering	90.02	10.07	**58.79±4.14**
Binarization	88.36	10.13	**541.2 ± 49.8**
defense-free baseline	90.01	10.23	16.4±1.14
Piecewise clustering	89.05	10.87	**29.8±11.3**
Binarization	88.00	10.20	**7874 ± 431.6**

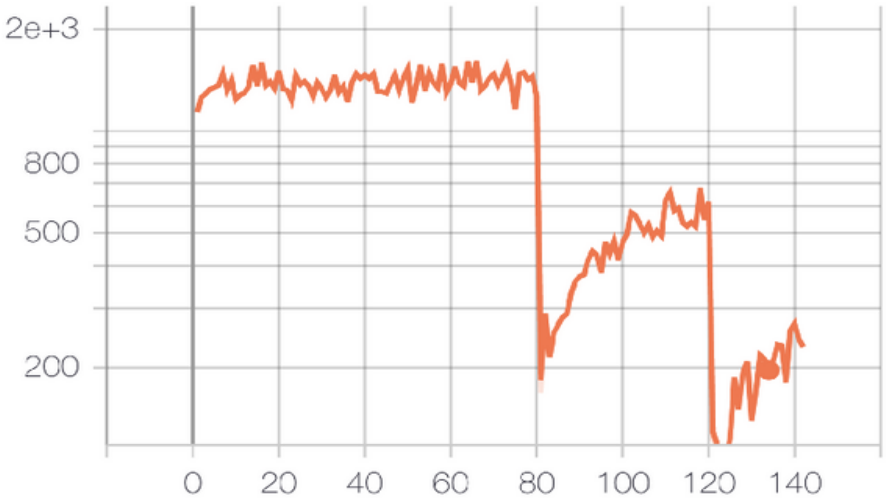


Figure: Average #Bit-flips (y-axis) per training iteration vs. epochs (x-axis).

- On **compact ResNet-20**, proposed defense improve the BFA resistance by **2×** and **19×**.
- On **over-parameterized VGG-11**, binarization improve the resistance to BFA by **480×**.
- Binarization-aware training involves many #bit-flips (i.e., Noise injection training).

Proposed Defense Methods Summary

- Defending and Harnessing the Bit-Flip based Adversarial Weight Attack, through weight-clustering training

  CVPR-2020, pushing the required bit-flip numbers to hundreds

- Defending Bit-Flip Attack through DNN Weight Reconstruction

  DAC 2020, pushing the required bit-flip numbers to hundreds

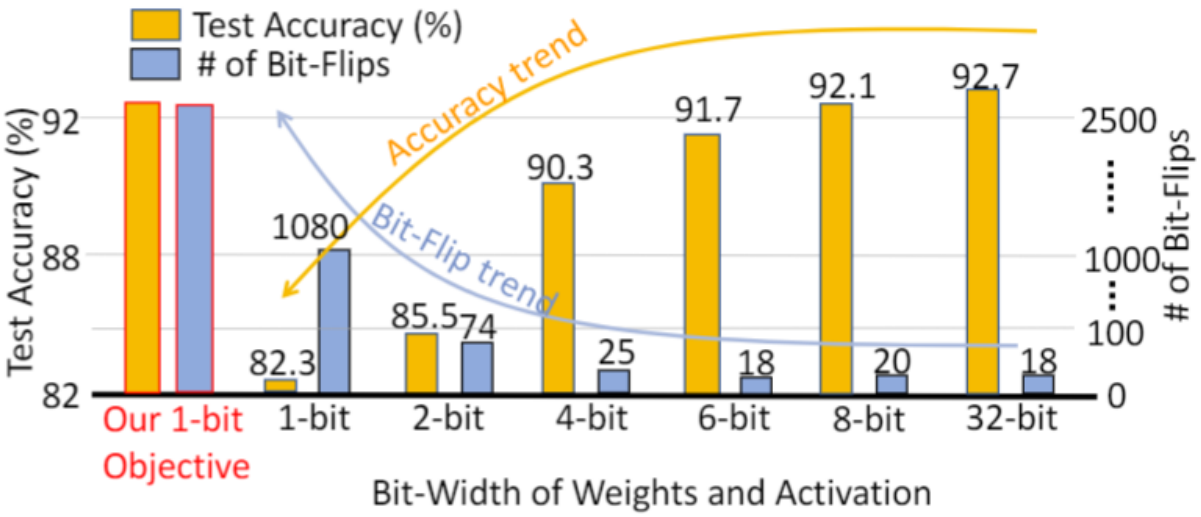
- RADAR: Run-time Adversarial Weight Attack Detection and Accuracy Recovery

  DATE 2021, targeting on the detection and recovery of bit-flip attack

- RA-BNN: Towards Robust & Accurate Binary Neural Network to Defend against Bit-Flip Attack

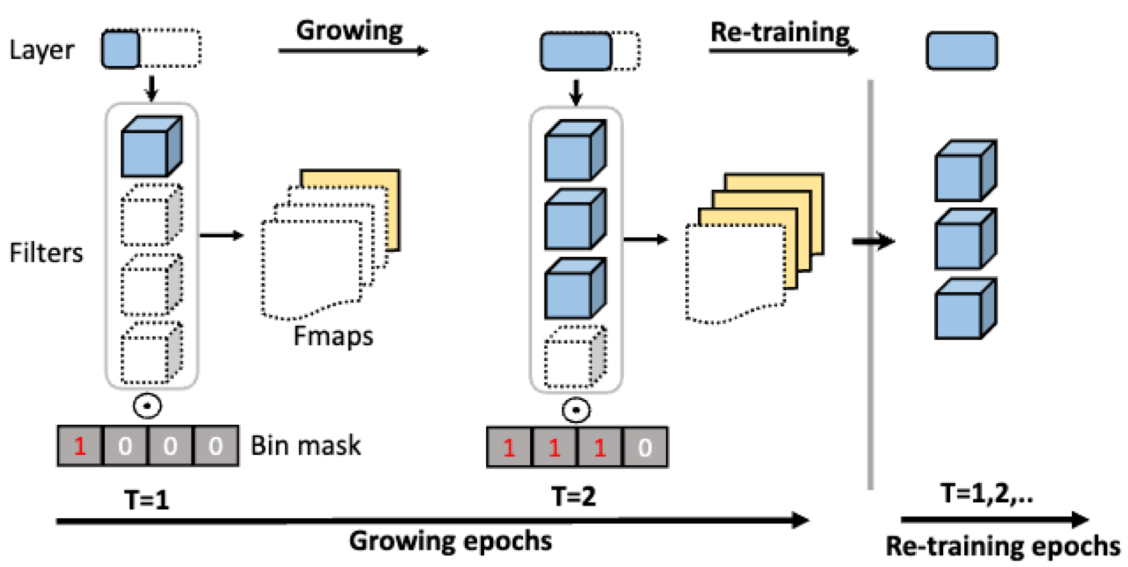
  first time to completely defend against BFA, cannot degrade to random guess level even with 5000 bit-flips
  arXiv preprint arXiv:2103.13813

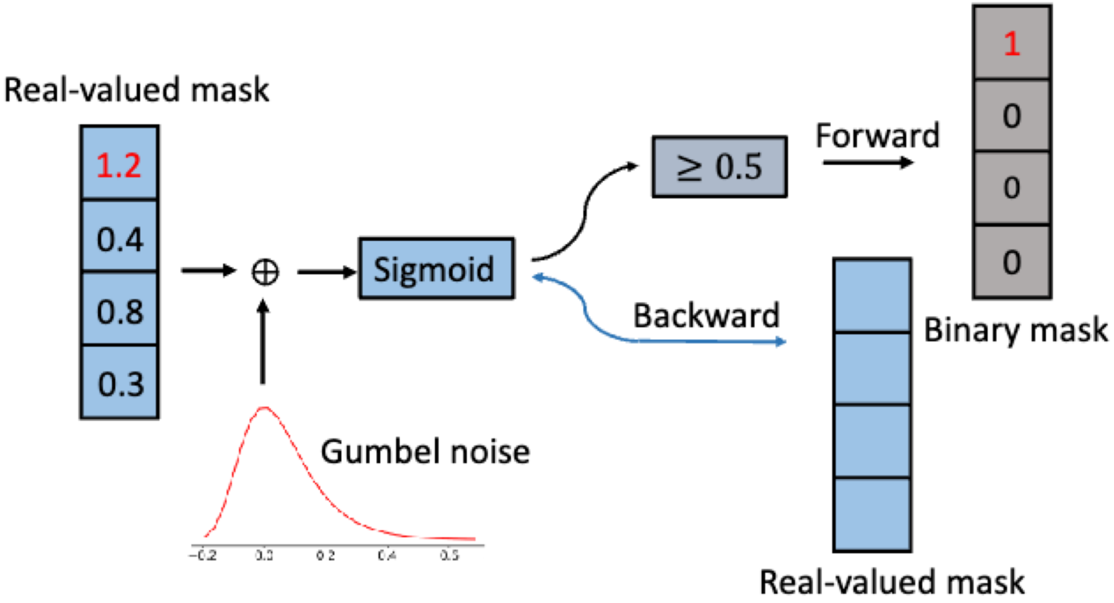
RA-BNN: Growing to a Robust and Accuracy Binary Neural Network



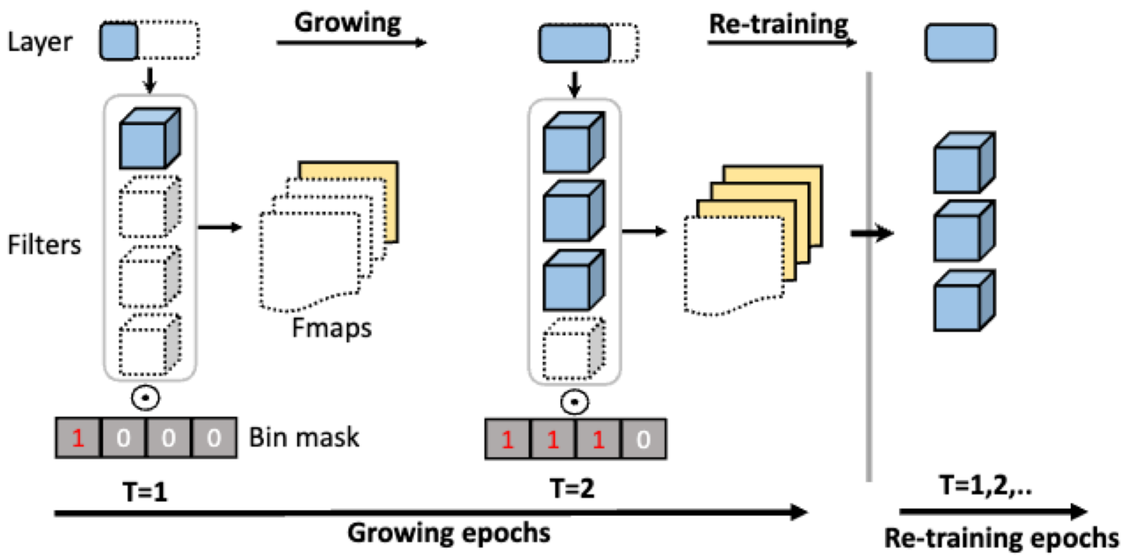
Proposal of two-stage RA-Growth.

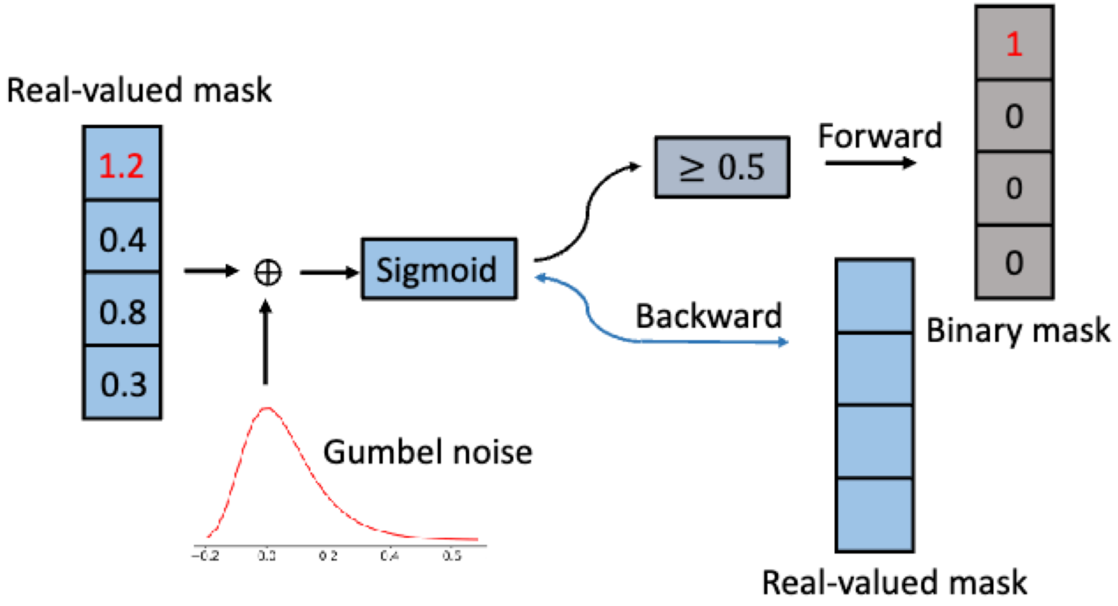
- ***i)*** ***Growing Stage***: growing the output channel of every layer during the early training iterations. A new channel will be created (i.e., growing) if the associated trainable channel-mask switches from 0 to 1 for the first time.
- ***ii)*** ***Re-training Stage***: training the weight parameters based on the new BNN structure learned in stage-1. The binary masks are trained using a combination of ***Gumbel-Sigmoid*** and hard thresholding functions.





Learning to Grow Binary Neural Network through Differentiable Gumbel-Sigmoid





The binary masks are trained using a combination of ***Gumbel-Sigmoid*** and hard thresholding functions.

1) Relax the hard threshold function to a logistic function.

$$\sigma(\boldsymbol{m}_{fp}) = \frac{1}{1+\exp(-\beta \boldsymbol{m}_{fp})},$$

2) Leverage Gumbel-Sigmoid function to a differential sampling to approximate a categorical random variable.

$$p(\boldsymbol{m}_{fp}) = \frac{\exp((\log\pi_0 + g_0)/T)}{\exp((\log\pi_0 + g_0)/T) + \exp((g_1)/T)},$$

3) With the differential masking function, the loss function could be redesigned as below to enable model growing during training:

$$\min_{\boldsymbol{w}_{fp}, \boldsymbol{m}_{fp}} \mathcal{L}_E(g(f(\boldsymbol{w}_{fp}) \odot p(\boldsymbol{m}_{fp}); \boldsymbol{x}_t), \boldsymbol{y}_t)$$

RA-BNN: Growing to a Robust and Accuracy Binary Neural Network

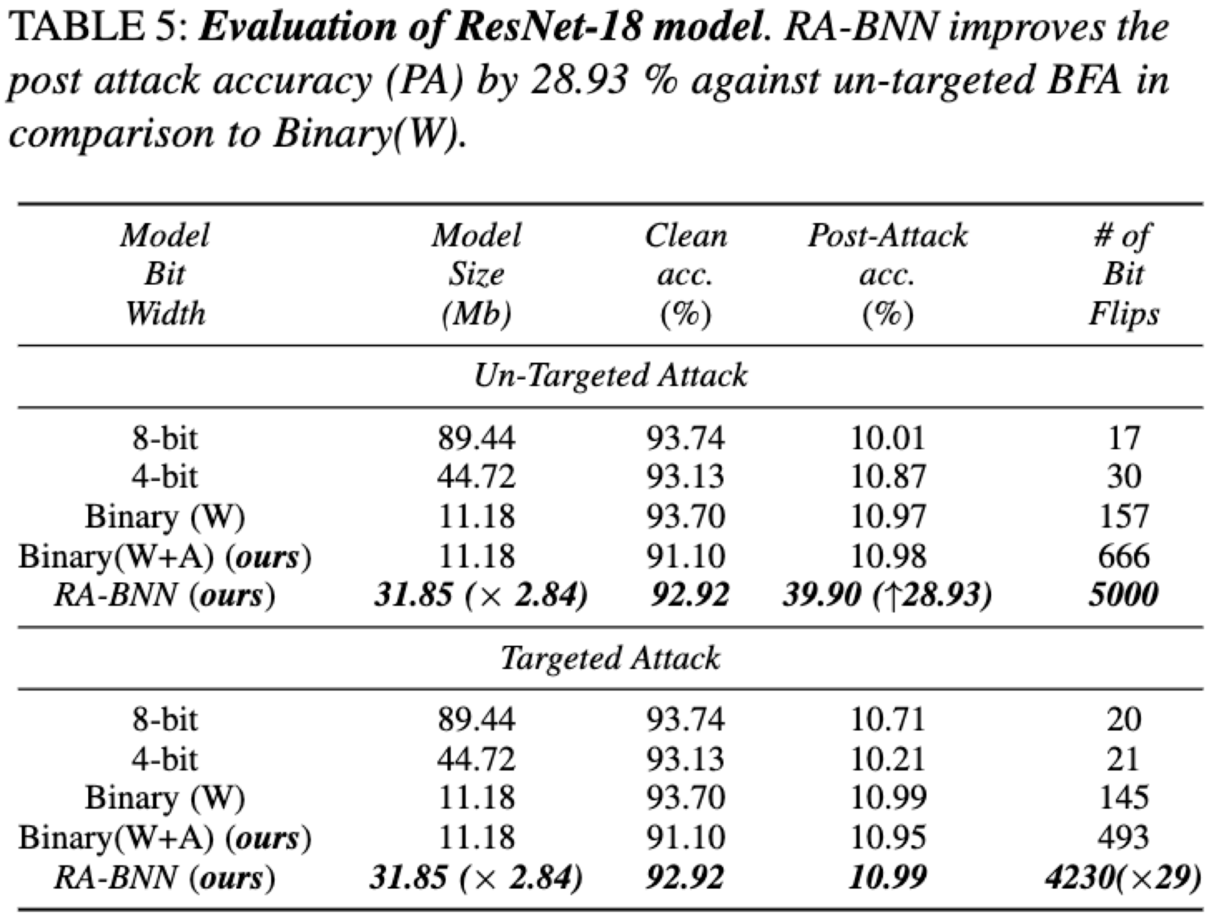


TABLE 5: ***Evaluation of ResNet-18 model***. *RA-BNN improves the post attack accuracy (PA) by 28.93 % against un-targeted BFA in comparison to Binary(W).*

Model Bit Width	*Model Size (Mb)*	*Clean acc. (%)*	*Post-Attack acc. (%)*	*# of Bit Flips*
		Un-Targeted Attack		
8-bit	89.44	93.74	10.01	17
4-bit	44.72	93.13	10.87	30
Binary (W)	11.18	93.70	10.97	157
Binary(W+A) (***ours***)	11.18	91.10	10.98	666
RA-BNN (***ours***)	***31.85*** **(× *2.84*)**	***92.92***	***39.90*** **(↑*28.93*)**	***5000***
		Targeted Attack		
8-bit	89.44	93.74	10.71	20
4-bit	44.72	93.13	10.21	21
Binary (W)	11.18	93.70	10.99	145
Binary(W+A) (***ours***)	11.18	91.10	10.95	493
RA-BNN (***ours***)	***31.85*** **(× *2.84*)**	***92.92***	***10.99***	***4230*****(×*****29*****)**

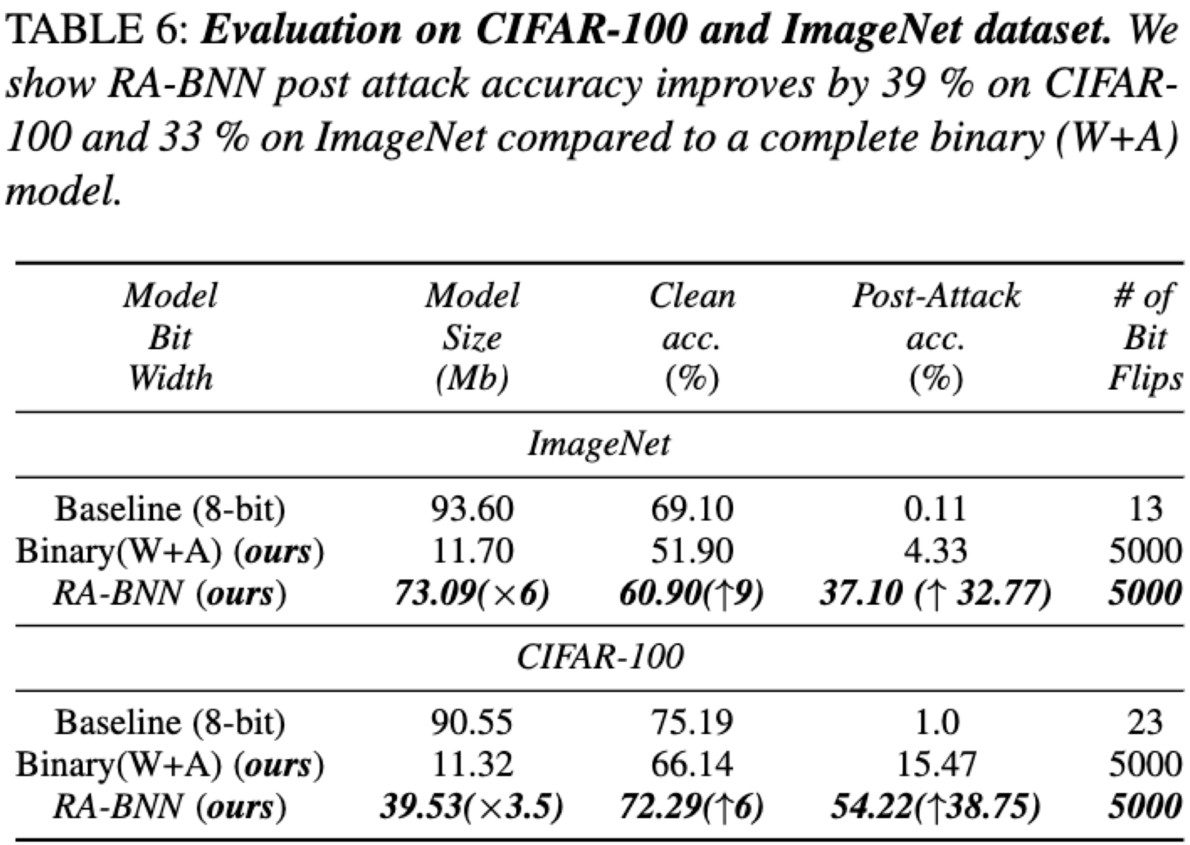


TABLE 6: ***Evaluation on CIFAR-100 and ImageNet dataset.*** *We show RA-BNN post attack accuracy improves by 39 % on CIFAR-100 and 33 % on ImageNet compared to a complete binary (W+A) model.*

Model Bit Width	*Model Size (Mb)*	*Clean acc. (%)*	*Post-Attack acc. (%)*	*# of Bit Flips*
		ImageNet		
Baseline (8-bit)	93.60	69.10	0.11	13
Binary(W+A) (***ours***)	11.70	51.90	4.33	5000
RA-BNN (***ours***)	***73.09*****(×*****6*****)**	***60.90*****(↑*****9*****)**	***37.10*** **(↑ *32.77*)**	***5000***
		CIFAR-100		
Baseline (8-bit)	90.55	75.19	1.0	23
Binary(W+A) (***ours***)	11.32	66.14	15.47	5000
RA-BNN (***ours***)	***39.53*****(×*****3.5*****)**	***72.29*****(↑*****6*****)**	***54.22*****(↑*****38.75*****)**	***5000***

- Best defense performance reported ever
- The AI model still works after 5000 bit-flips

RA-BNN: Growing to a Robust and Accuracy Binary Neural Network

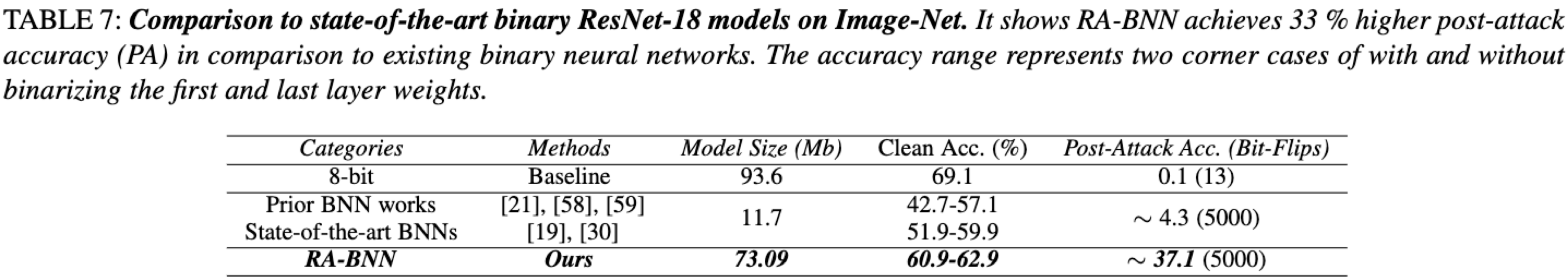


TABLE 7: ***Comparison to state-of-the-art binary ResNet-18 models on Image-Net.*** *It shows RA-BNN achieves 33 % higher post-attack accuracy (PA) in comparison to existing binary neural networks. The accuracy range represents two corner cases of with and without binarizing the first and last layer weights.*

Categories	*Methods*	*Model Size (Mb)*	Clean Acc. (%)	*Post-Attack Acc. (Bit-Flips)*
8-bit	Baseline	93.6	69.1	0.1 (13)
Prior BNN works	[21], [58], [59]	11.7	42.7-57.1	$\sim$ 4.3 (5000)
State-of-the-art BNNs	[19], [30]		51.9-59.9	
RA-BNN	***Ours***	***73.09***	***60.9-62.9***	$\sim$ ***37.1*** (5000)

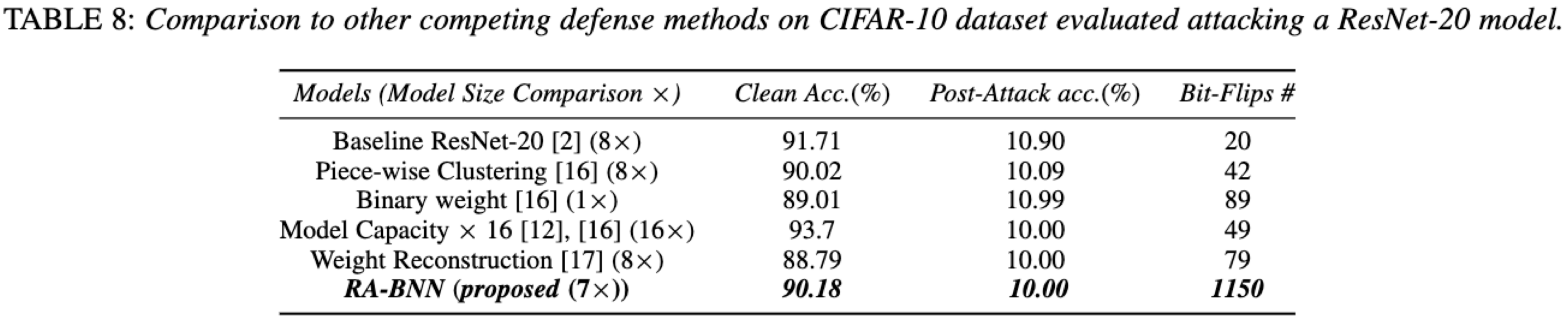


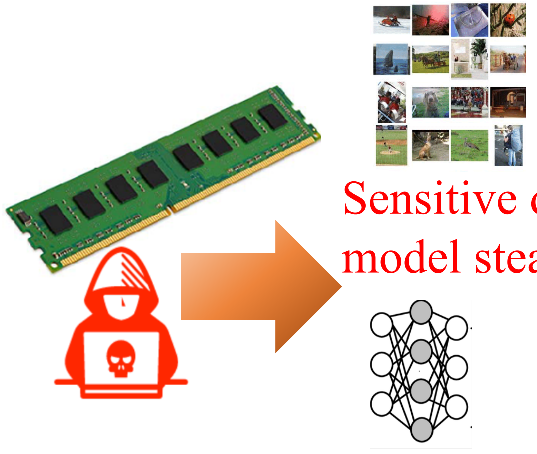
TABLE 8: *Comparison to other competing defense methods on CIFAR-10 dataset evaluated attacking a ResNet-20 model.*

Models (Model Size Comparison ×)	*Clean Acc.(%)*	*Post-Attack acc.(%)*	*Bit-Flips #*
Baseline ResNet-20 [2] (8×)	91.71	10.90	20
Piece-wise Clustering [16] (8×)	90.02	10.09	42
Binary weight [16] (1×)	89.01	10.99	89
Model Capacity × 16 [12], [16] (16×)	93.7	10.00	49
Weight Reconstruction [17] (8×)	88.79	10.00	79
RA-BNN (proposed (7×))	***90.18***	***10.00***	***1150***

- Best defense performance reported ever

DeepSteal: Advanced Model Extractions Leveraging Efficient Weight Stealing in Memories

Published in 2022 *43rd IEEE Symposium on Security and Privacy (S&P)*



Sensitive data model stealin

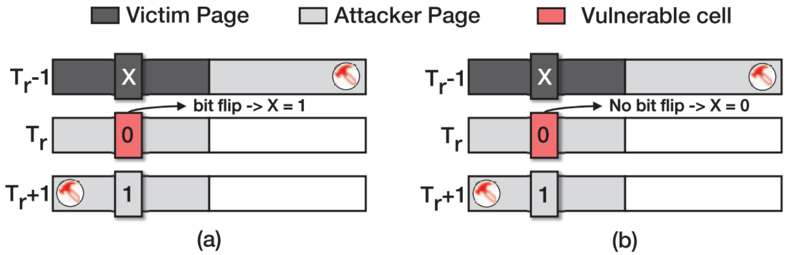
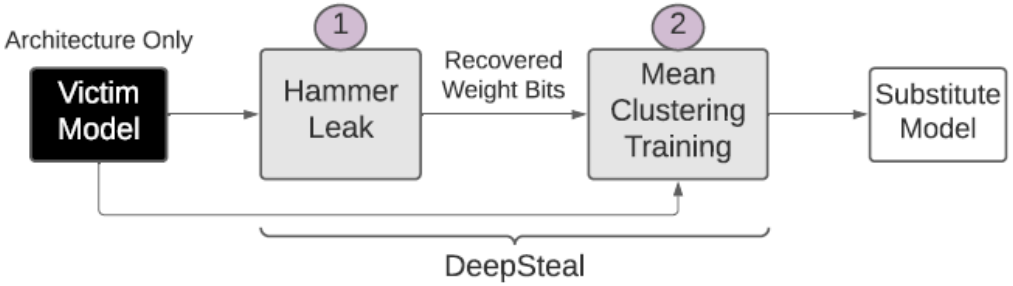


Fig. 3: HammerLeak attack leaking victim secrets using single victim page. (a) Bit flip observed when victim's bit is `1`, (b) Bit flip not observed when victim's bit is `0`.



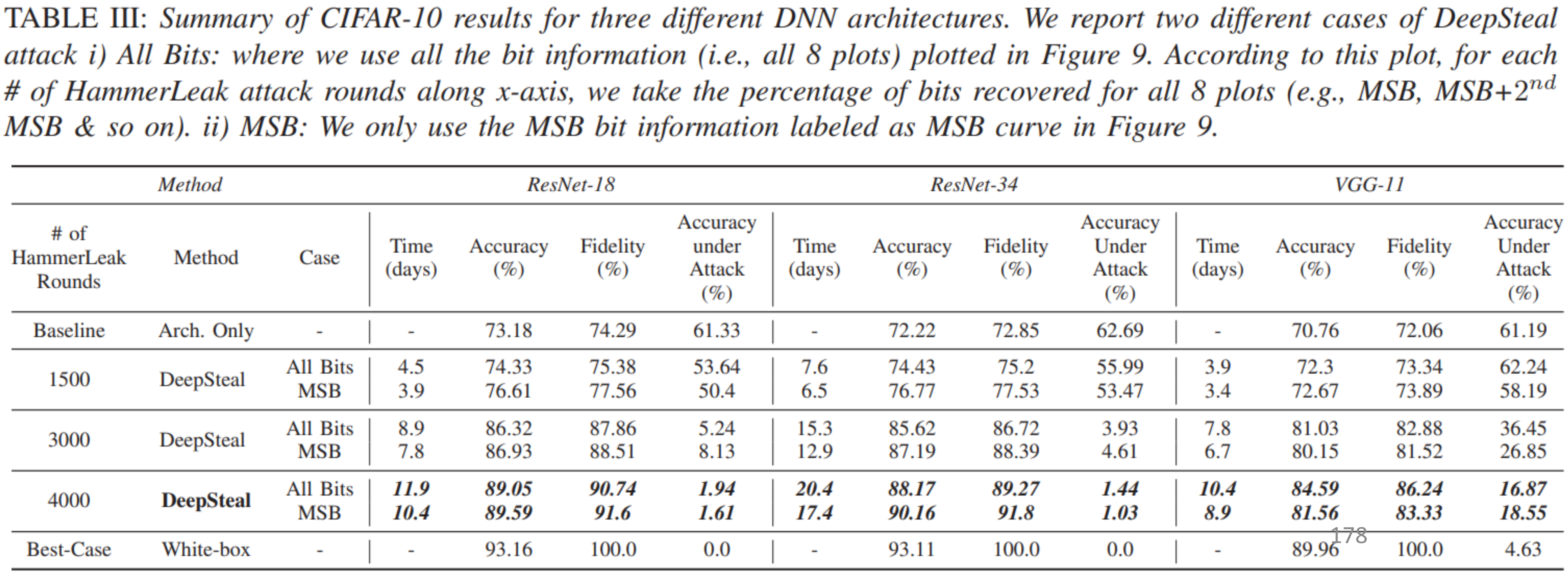


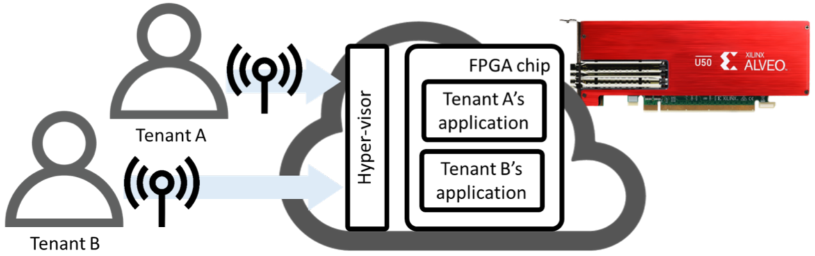
TABLE III: *Summary of CIFAR-10 results for three different DNN architectures. We report two different cases of DeepSteal attack i) All Bits: where we use all the bit information (i.e., all 8 plots) plotted in Figure 9. According to this plot, for each # of HammerLeak attack rounds along x-axis, we take the percentage of bits recovered for all 8 plots (e.g., MSB, MSB+2nd MSB & so on). ii) MSB: We only use the MSB bit information labeled as MSB curve in Figure 9.*

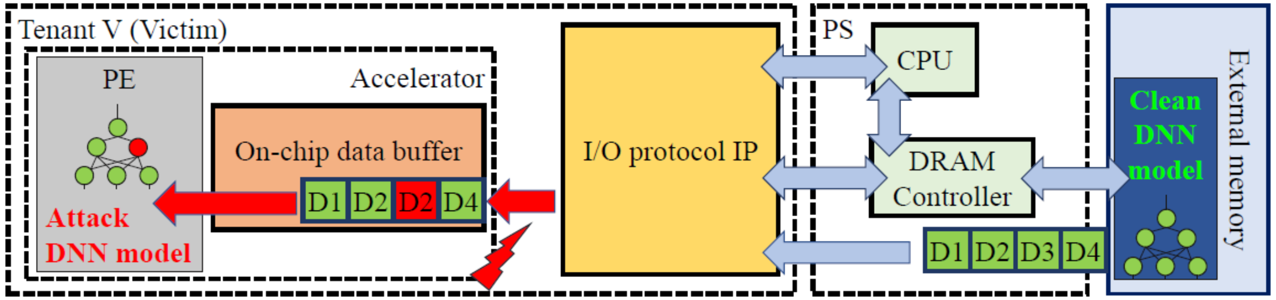
Method			*ResNet-18*				*ResNet-34*				*VGG-11*			
# of HammerLeak Rounds	Method	Case	Time (days)	Accuracy (%)	Fidelity (%)	Accuracy under Attack (%)	Time (days)	Accuracy (%)	Fidelity (%)	Accuracy Under Attack (%)	Time (days)	Accuracy (%)	Fidelity (%)	Accuracy Under Attack (%)
Baseline	Arch. Only	-	-	73.18	74.29	61.33	-	72.22	72.85	62.69	-	70.76	72.06	61.19
1500	DeepSteal	All Bits MSB	4.5 3.9	74.33 76.61	75.38 77.56	53.64 50.4	7.6 6.5	74.43 76.77	75.2 77.53	55.99 53.47	3.9 3.4	72.3 72.67	73.34 73.89	62.24 58.19
3000	DeepSteal	All Bits MSB	8.9 7.8	86.32 86.93	87.86 88.51	5.24 8.13	15.3 12.9	85.62 87.19	86.72 88.39	3.93 4.61	7.8 6.7	81.03 80.15	82.88 81.52	36.45 26.85
4000	**DeepSteal**	All Bits MSB	***11.9*** ***10.4***	***89.05*** ***89.59***	***90.74*** ***91.6***	***1.94*** ***1.61***	***20.4*** ***17.4***	***88.17*** ***90.16***	***89.27*** ***91.8***	***1.44*** ***1.03***	***10.4*** ***8.9***	***84.59*** ***81.56***	***86.24*** ***83.33***	***16.87*** ***18.55***
Best-Case	White-box	-	-	93.16	100.0	0.0	-	93.11	100.0	0.0	-	89.96	100.0	4.63

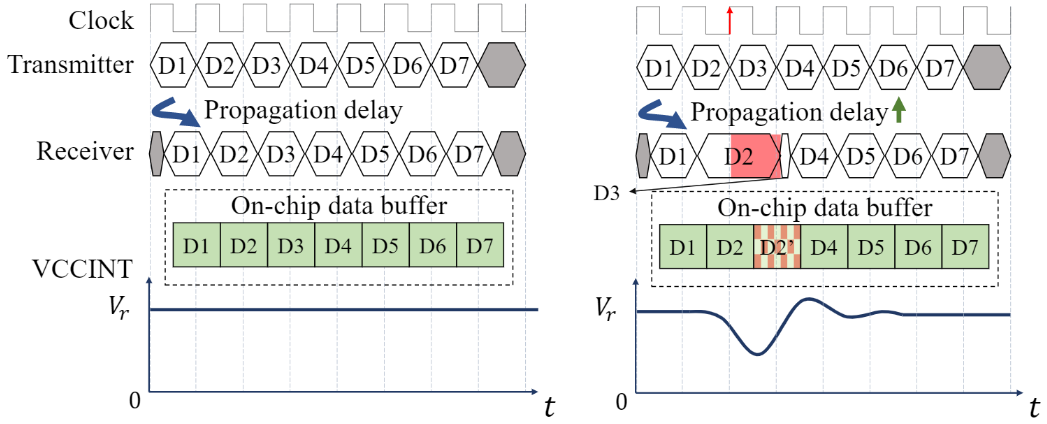
Recovery AI model function from partially leaked parameters

Deep-Dup: An Adversarial Weight Duplication Attack Framework to Crush Deep Neural Network in Multi-Tenant FPGA

- adversarial DNN model fault injection attack, utilizing our DNN vulnerable parameter searching software to guide and search when/where to inject fault into **off-chip data communication** through power-plundering circuits to conduct un-targeted/targeted attacks in **multi-tenant cloud FPGA**.







(a) DNN model transmission w/o attack.

(b) DNN model transmission under AWD attack.

Deep-Dup is the **first to demonstrate adversarial weight attack in real FPGA under black-box threat** model for large scale real-word DNN applications in pattern recognition and object tracking

Adnan Siraj Rakin*, Yukui Luo*, Xiaolin Xu and Deliang Fan, "Deep-Dup: An Adversarial Weight Duplication Attack Framework to Crush Deep Neural Network in Multi-Tenant FPGA," *In 30th* ***USENIX Security Symposium***, August 11-13, 2021.
paper, slides and talk video are available in https://www.usenix.org/conference/usenixsecurity21/presentation/rakin

Experiments – Black Box Attack (YoLov2)

Table 3: Black-Box attack for object detection.

Black-Box Un-Targeted Attack on YOLOv2 using RO cell

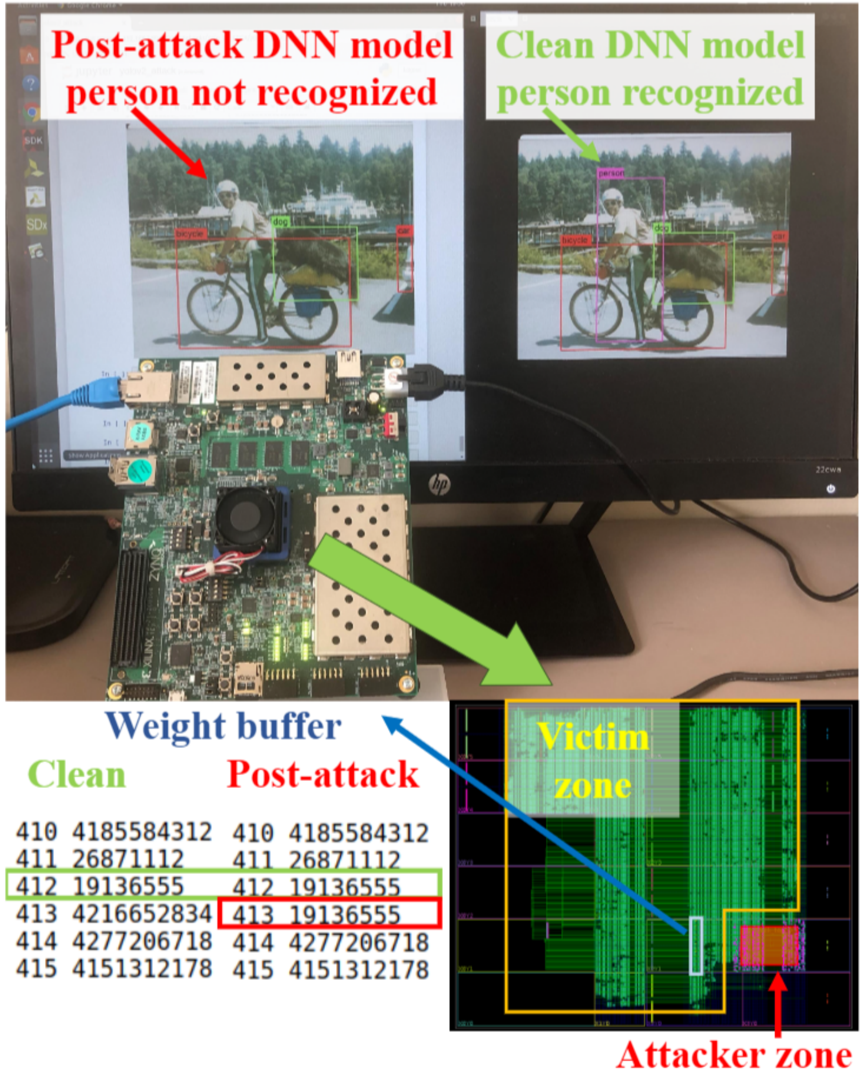
Target Class (t_s)	mAP	Post- Attack mAP	# of Attacks
All	0.428	0.06	30

Black-Box Un-Targeted Attack on YOLOv2 using LRO cell

Target Class (t_s)	mAP	Post- Attack mAP	# of Attacks
All	0.428	0.14	63

Black-Box Targeted Attack on YOLOv2 using RO cell

Target Class (t_s)	AP	Post-Attack AP	# of Attacks
Person	0.6039	0.0507	20
Car	0.5108	0.0621	18
Bowl	0.3290	0.0348	15
Sandwich	0.4063	0.0125	6



```
410 4185584312  410 4185584312
411 26871112    411 26871112
412 19136555    412 19136555
413 4216652834  413 19136555
414 4277206718  414 4277206718
415 4151312178  415 4151312178
```

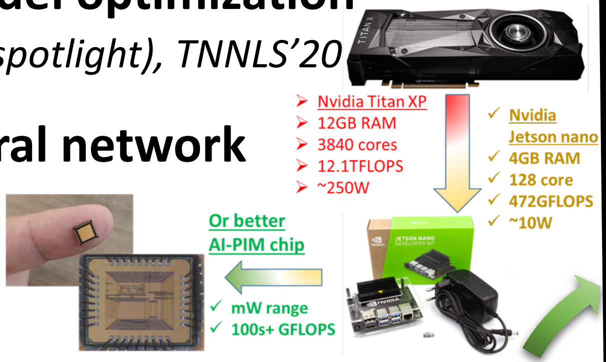
- **mAP**: mean average precision

Summary: Efficient, Secure, and Intelligent Computing

Algorithm

AI Performance & Efficiency

- **Compute- and Memory-efficient on-device learning** *NeurIPS'22/23, CVPR'22/21, AAAI'22(spotlight), ICLR'22(spotlight), etc.*
- **Hardware-aware AI model optimization** *CVPR'19, WACV'19, AAAI'20 (spotlight), TNNLS'20*
- **Run-time dynamic neural network** *TNNLS'22, NeurIPS'22, DAC'20*


- Nvidia Titan XP
- 12GB RAM
- 3840 cores
- 12.1TFLOPS
- ~250W

✓ Nvidia Jetson nano
✓ 4GB RAM
✓ 128 core
✓ 472GFLOPS
✓ ~10W

Or better AI-PIM chip

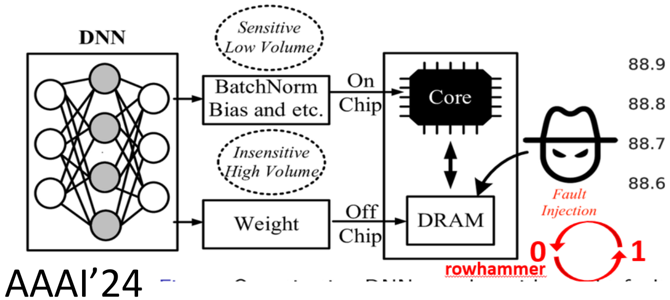
✓ mW range
✓ 100s+ GFLOPS

AI Security & Privacy

- **Adversarial noise robustness** *CVPR'19, CVOPS'19*

- **Adversarial Weight Attack & Defense** *ICCV'19, CVPR'20, DAC'20, DATE'21, etc.*
- **AI Trojan Attack** *CVPR'20, TPAMI'21*
- **Model inversion**

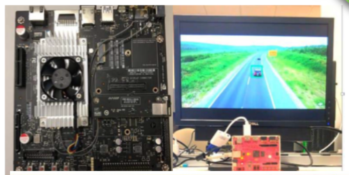
HOST'21, CVPR'22, SP'22, AAAI'24

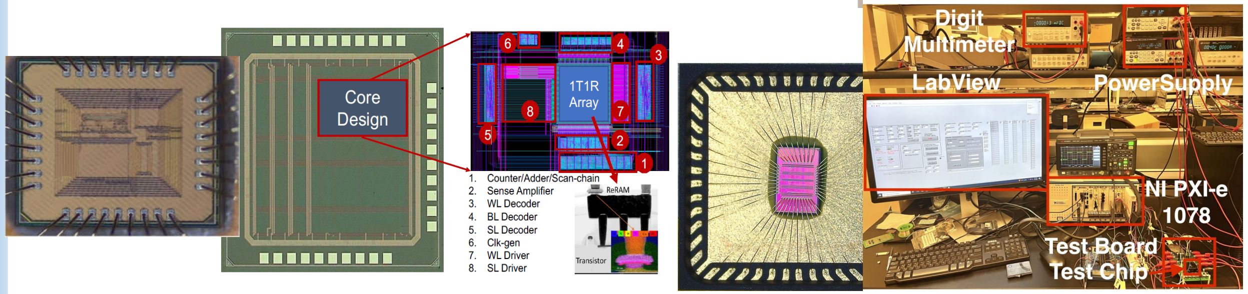


Co-Design

Hardware

- **In-memory computing chips**
- **Emerging non-volatile memory**
- **Neuromorphic computing**

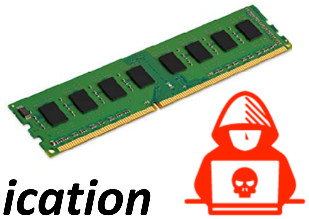




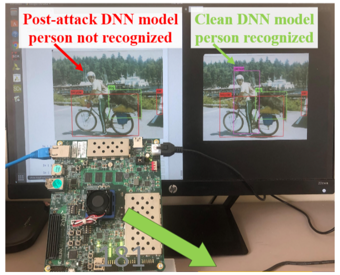
1. Counter/Adder/Scan-chain
2. Sense Amplifier
3. WL Decoder
4. BL Decoder
5. SL Decoder
6. Clk-gen
7. WL Driver
8. SL Driver

ESSCIRC'22/23, CICC'24, DAC'16-24, ICCAD'18-21, DATE'17-23, DRC'19, *JSSC (invited), SSCL,* OJSSCS, TCAS-I/II, TMAG, TC, TNANO, TCAD, EDL, etc.

- **Memory Bit-Flip Attack in Computer *main memory*** *USENIX Security'20*



- **Fault injection into the data *communication* in cloud-FPGA in black-box setup** *USENIX Security'21, Security & Privacy (SP)'24*


Post-attack DNN model person not recognized
Clean DNN model person recognized

- **AI Model/Data Stealing from memory side-channel** *IEEE-Security & Privacy (SP) – 2022*

Thank You & Questions?

Deliang Fan

Director of *Efficient, Secure and Intelligent Computing* (ESIC) Laboratory
School of Electrical, Computer and Energy Engineering
Arizona State University, Tempe, AZ, USA
Email: dfan@asu.edu
https://faculty.engineering.asu.edu/dfan/

NSF DARPA tsmc SAMSUNG MOFFITT CANCER CENTER

Semiconductor Research Corporation intel CYBER FLORIDA the FLORIDA CENTER FOR CYBERSECURITY

MITSUBISHI ELECTRIC Changes for the Better MERL - MITSUBISHI ELECTRIC RESEARCH LABORATORIES