TECHNOLOGY DRIVERS FOR IN-MEMORY ACCELERATORS FOR MACHINE LEARNING, ENCRYPTION, AND BEYOND

Michael Niemier

ND Students and Post-Docs – Arman Kazemi, Ann Franchesca Lagnua, Ramin Rajaei, Dayane Reis, Mehdi Sharifi, Jonathan Takeshita

Collaborators (within JUMP) – Suman Datta, José Martínez (forthcoming!)

Collaborators (~~outside~~ JUMP) – X. Sharon Hu, Siddharth Joshi, Taeho Jung

ASCENT

Applications and Systems Driven Center for Energy-Efficient Integrated Nanotechnologies

This work was supported in part by the Semiconductor Research Corporation (SRC) and DARPA.

TWO TOPICS

Few-shot learning

How do we learn with just a few examples?

Secure computing

Computing in the cloud:

User data: 17 → **Encrypted:** 27142
24 → **Encrypted:** 98716

Cloud: {27142, 98716}

Not private!

Process:
27142 → 17 **(decrypted)**
98716 → 24 **(decrypted)**
17+24 = 41
41 → 56817 **(encrypted)**

Computing in the cloud (encrypted data):

User data: 17 → **Encrypted:** $2x^3+3x^2+5x+6$
24 → **Encrypted:** $7x^3+2x^2+2x+2$

Cloud: $\{2x^3+3x^2+5x+9,\ 7x^3+2x^2+2x+2\}$

Process:

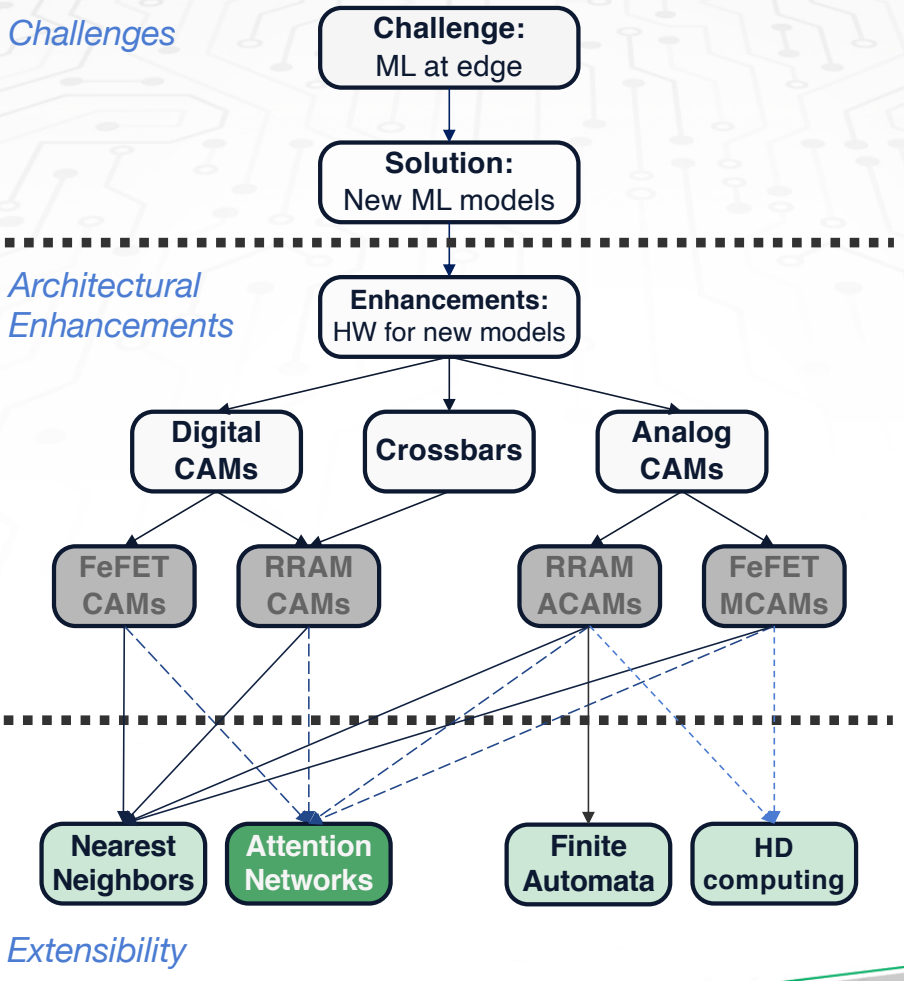
$$\begin{array}{r} 2x^3+3x^2+5x+6 \\ +\ 7x^3+2x^2+2x+2 \\ \hline 9x^3+5x^2+7x+8 \end{array}$$

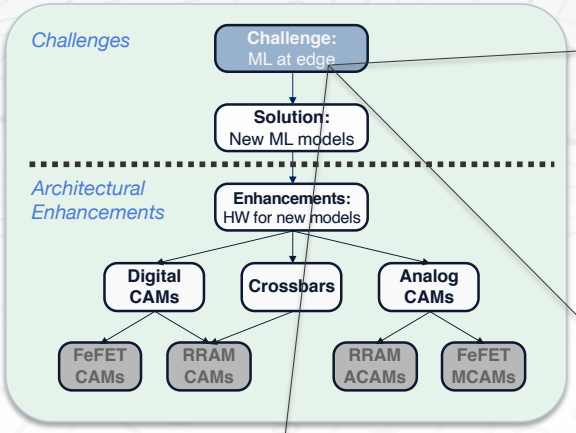
Data not decrypted in cloud

To user: $9x^3+5x^2+7x+8$ → 41 **(decrypted)**

MACHINE LEARNING AT THE EDGE

MACHINE LEARNING AT THE EDGE



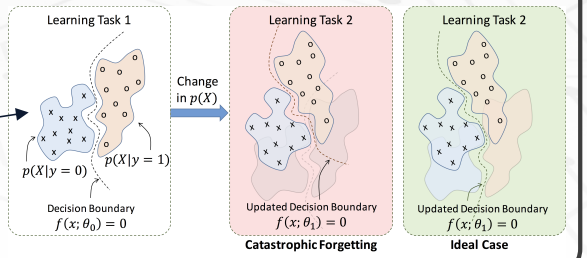


Networks cannot be easily retrained to adapt to new classes of data

Attention-Based Selective Plasticity

Soheil Kolouri [1] Nicholas Ketz [1] Xinyun Zou [2] Jeffrey Krichmar [2] Praveen Pilly [1]

Figure 1: Depiction of catastrophic forgetting in binary classification tasks when there is a distribution shift from an initial task to a secondary task. When exposed to the distribution of the new task, the uniformly plastic parametric model, $f(\cdot, \theta)$, conforms to the new distribution with no constraints on maintaining its performance on the previous task.



Memory transfer costs preclude use of sophisticated networks on edge devices

EIE: Efficient Inference Engine on Compressed Deep Neural Network

Song Han* Xingyu Liu* Huizi Mao* Jing Pu* Ardavan Pedram*
Mark A. Horowitz* William J. Dally*†
*Stanford University, †NVIDIA
{songhan,xyl,huizi,jingpu,perdavan,horowitz,dally}@stanford.edu

Table I
ENERGY TABLE FOR 45NM CMOS PROCESS [9]. DRAM ACCESS USES THREE ORDERS OF MAGNITUDE MORE ENERGY THAN SIMPLE ARITHMETIC AND 128X MORE THAN SRAM.

Operation	Energy [pJ]	Relative Cost
32 bit int ADD	0.1	1
32 bit float ADD	0.9	9
32 bit int MULT	3.1	31
32 bit float MULT	3.7	37
32 bit 32KB SRAM	5	50
32 bit DRAM	**640**	**6400**

Network training costs high, prohibitive on IoT devices

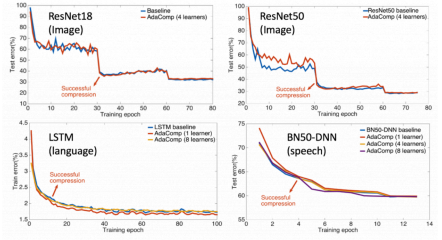
IBM Research Blog Topics Labs About

AI

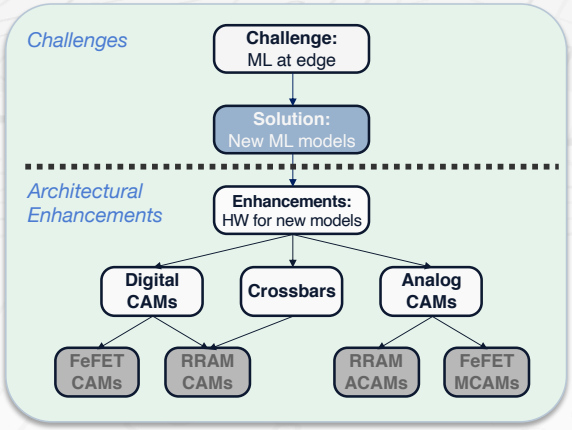
Deep Learning Training Times Get Significant Reduction

February 5, 2018 | Written by: Chia-Yu Chen and Kailash Gopalakrishnan

Training a deep learning model on a large dataset is a challenging and expensive task that can take anywhere from hours to weeks to complete. To tackle this problem, typically a cluster of four to 128 GPU accelerators is used to divide the overall task, reducing training time by exploiting the combined computational strengths of multiple accelerators all working on the same problem. Larger the number of GPUs, lesser is the computational time spent in each accelerator for the given training task.



AdaComp shows no degradation for deep learning models across vision, speech and language.



MANN approaches effective!

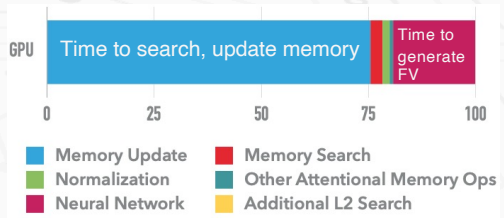
Model	Matching Fn	Fine Tune	5-way Acc 1-shot	5-way Acc 5-shot	20-way Acc 1-shot	20-way Acc 5-shot
PIXELS	Cosine	N	41.7%	63.2%	26.7%	42.6%
BASELINE CLASSIFIER	Cosine	N	80.0%	95.0%	69.5%	89.1%
BASELINE CLASSIFIER	Cosine	Y	82.3%	98.4%	70.6%	92.0%
BASELINE CLASSIFIER	Softmax	Y	86.0%	97.6%	72.9%	92.3%
MANN (NO CONV) [21]	Cosine	N	82.8%	94.9%	–	–
CONVOLUTIONAL SIAMESE NET [11]	Cosine	N	96.7%	98.4%	88.0%	96.5%
CONVOLUTIONAL SIAMESE NET [11]	Cosine	Y	97.3%	98.4%	88.1%	97.0%
MATCHING NETS (OURS)	Cosine	N	**98.1%**	**98.9%**	**93.8%**	98.5%
MATCHING NETS (OURS)	Cosine	Y	97.9%	98.7%	93.5%	**98.7%**

Table 1: Results on the Omniglot dataset.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K. & Wierstra, D. Matching networks for one shot learning. In *Proc. Advances in Neural Information Processing Systems 29 (NIPS 2016)* (eds Lee, D. D. et al.) 3637–3645.

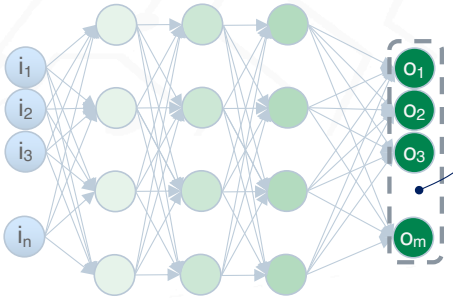
MANNs introduce new bottlenecks

Comparisons and memory updates can be costly!

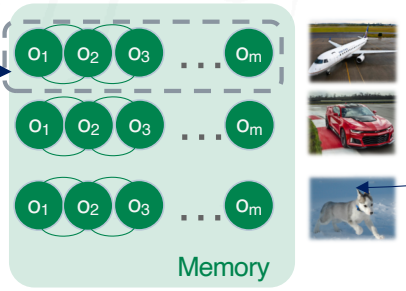


A MANN-based approach:

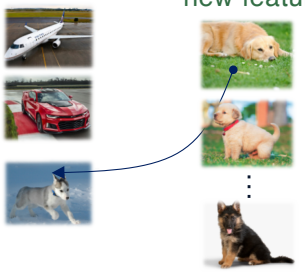
1 Consider neural network…



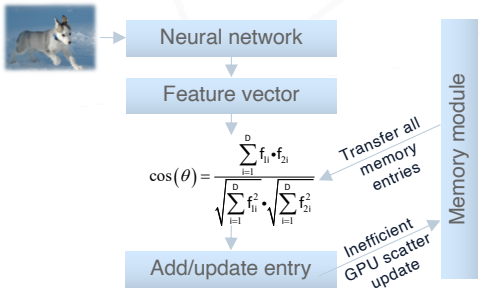
2 Treat NN output as feature vector, store in memory

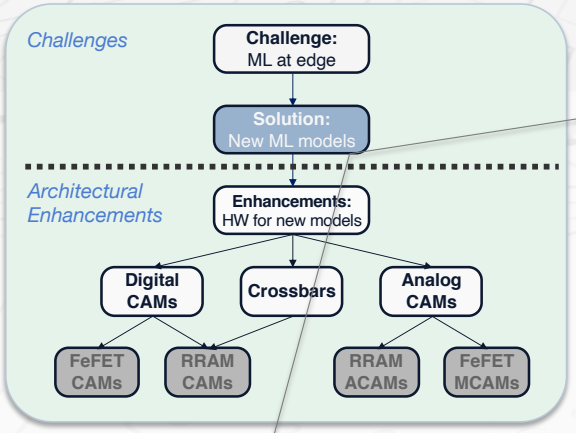


3 Post-training, show new classes, store new feature vectors



4 Classify new objects learned post training via memory lookups and distance metric calculations





Edge AI…

IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, VOL. 19, NO. 1, JANUARY 2020

Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing

En Li, Liekang Zeng, Zhi Zhou, *Member, IEEE*, and Xu Chen, *Member, IEEE*

***Abstract*— As a key technology of enabling Artificial Intelligence (AI) applications in 5G era, Deep Neural Networks (DNNs) have quickly attracted widespread attention. However, it is challenging to run computation-intensive DNN-based tasks on mobile devices due to the limited computation resources. What's worse, traditional cloud-assisted DNN inference is heavily hindered by the significant wide-area network latency, leading to poor real-time performance as well as low quality of user experience. To address these challenges, in this paper, we propose Edgent, a framework that leverages edge computing for DNN collaborative inference through device-edge synergy. Edgent exploits two design knobs: (1) DNN partitioning that adaptively partitions computation between device and edge for purpose of coordinating the powerful cloud resource and the proximal edge resource for real-time DNN inference; (2) DNN right-sizing that further reduces computing latency via early exiting inference at an appropriate intermediate DNN layer. In addition, considering the potential network fluctuation in real-world deployment, Edgent is properly design to specialize for both static and dynamic network environment. Specifically, in a static environment where the bandwidth changes slowly, Edgent derives the best configurations with the assist of regression-based prediction models, while in a dynamic environment where the bandwidth varies dramatically, Edgent generates the best execution plan through the online change point detection algorithm that maps the current bandwidth state to the optimal configuration. We implement Edgent prototype based on the Raspberry Pi and the desktop PC and the extensive experimental evaluations demonstrate Edgent's effectiveness in enabling on-demand low-latency edge intelligence.**

***Index Terms*— Edge intelligence, edge computing, deep learning, computation offloading.**

Robotic systems…

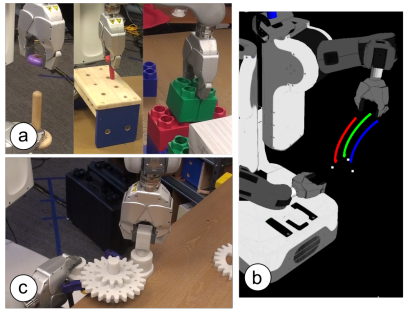
One-Shot Learning of Manipulation Skills with Online Dynamics Adaptation and Neural Network Priors

Justin Fu, Sergey Levine, Pieter Abbeel

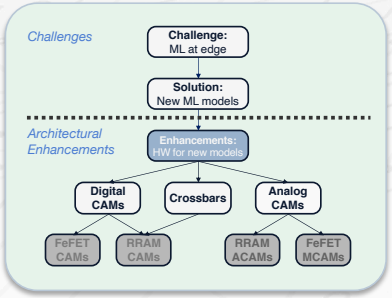
***Abstract*— One of the key challenges in applying reinforcement learning to complex robotic control tasks is the need to gather large amounts of experience in order to find an effective policy for the task at hand. Model-based reinforcement learning can achieve good sample efficiency, but requires the ability to learn a model of the dynamics that is good enough to learn an effective policy. In this work, we develop a model-based reinforcement learning algorithm that combines prior knowledge from previous tasks with online adaptation of the dynamics model. These two ingredients enable highly sample-efficient**

One-Shot Imitation Learning

Yan Duan[†§], Marcin Andrychowicz[‡], Bradly Stadie[†‡], Jonathan Ho[†§], Jonas Schneider[‡], Ilya Sutskever[‡], Pieter Abbeel[†§], Wojciech Zaremba[‡]
[†]Berkeley AI Research Lab, [‡]OpenAI
[§]Work done while at OpenAI
{rockyduan, jonathanho, pabbeel}@eecs.berkeley.edu
{marcin, bstadie, jonas, ilyasu, woj}@openai.com



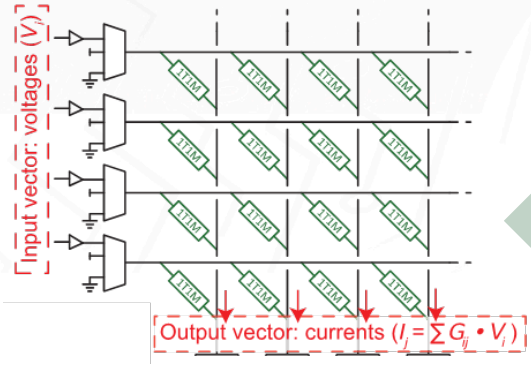
ARCHITECTURAL-LEVEL SOLUTIONS



Developing energy-efficient hardware for IoT/Edge requires us to address the energy and latency costs of memory-processor data movement

- Near-memory, in-memory computing architectures reduce these costs

In-memory computing with non-volatile <u>crossbars</u>

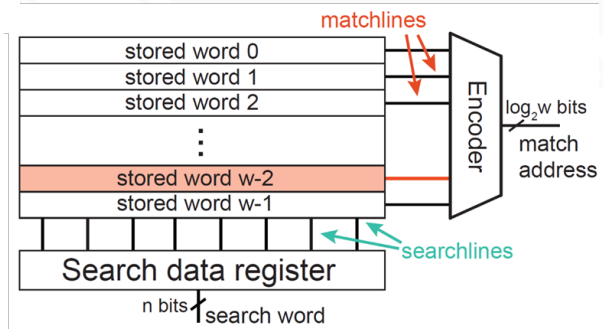


Application areas:

- Deep Learning (CNNs, LSTMs, Restricted Boltzmann Machines, etc.)
- Signal and sensor processing (Fast Fourier Transform)
- Combinatorial Optimization (e.g., NP-hard graph problems)
- Mixed-precision linear equation solving

+ combinations there of…

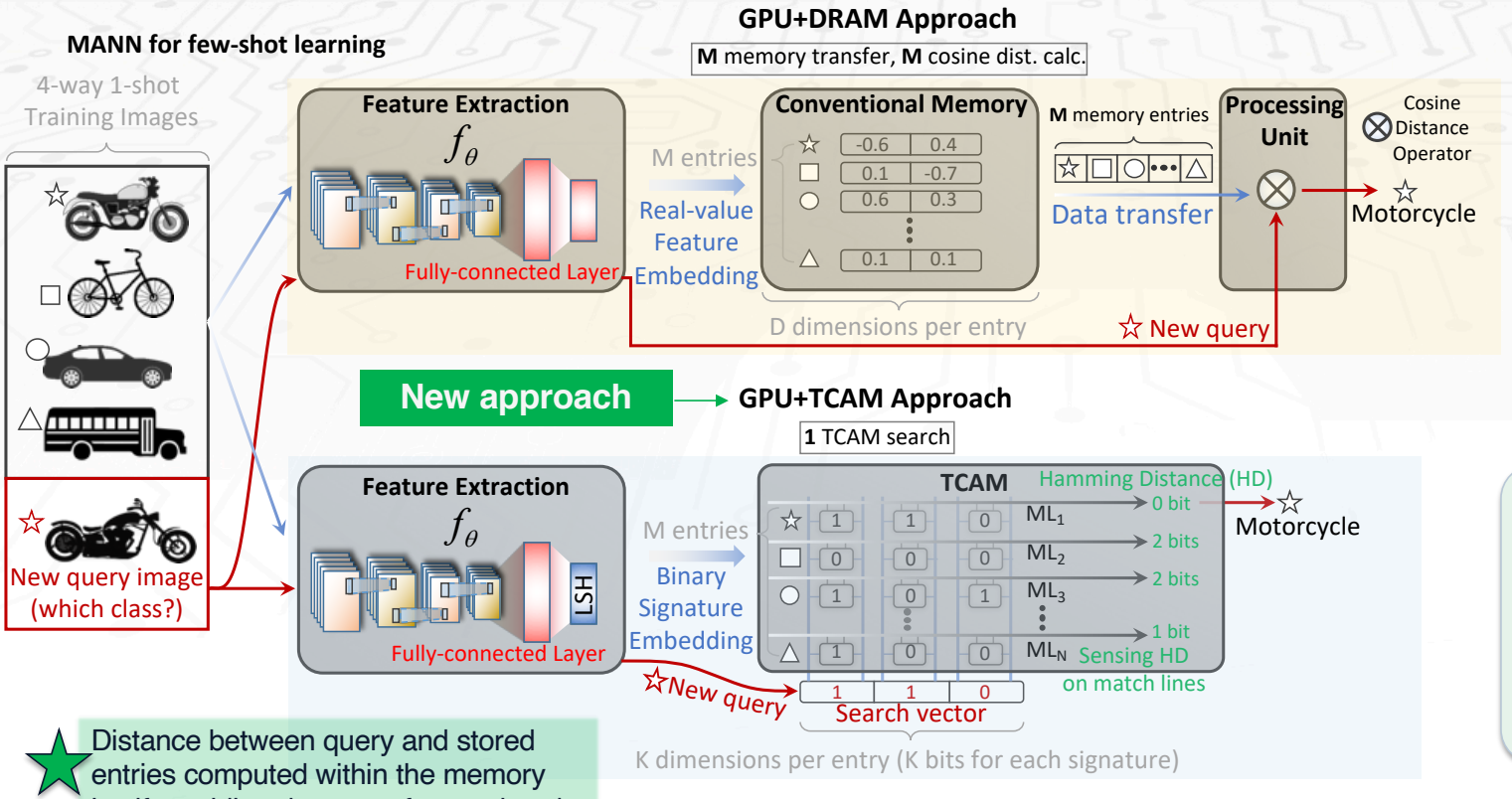
In-memory computing with <u>content addressable memories</u> (CAMs)



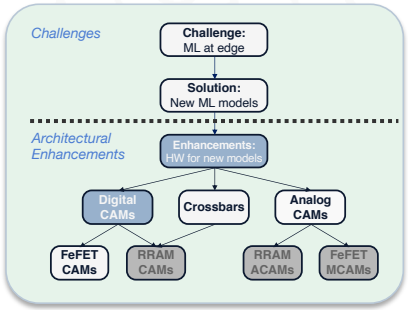
Application areas:

- Hash operations, Hamming distance
- Security, Virus-detection (zero-error Bloom filtering)
- Bio-informatics (genomic sequencing)
- Decision trees / random forests

CAN HARDWARE ARCHITECTURES SUPPORT NEW MODELS?

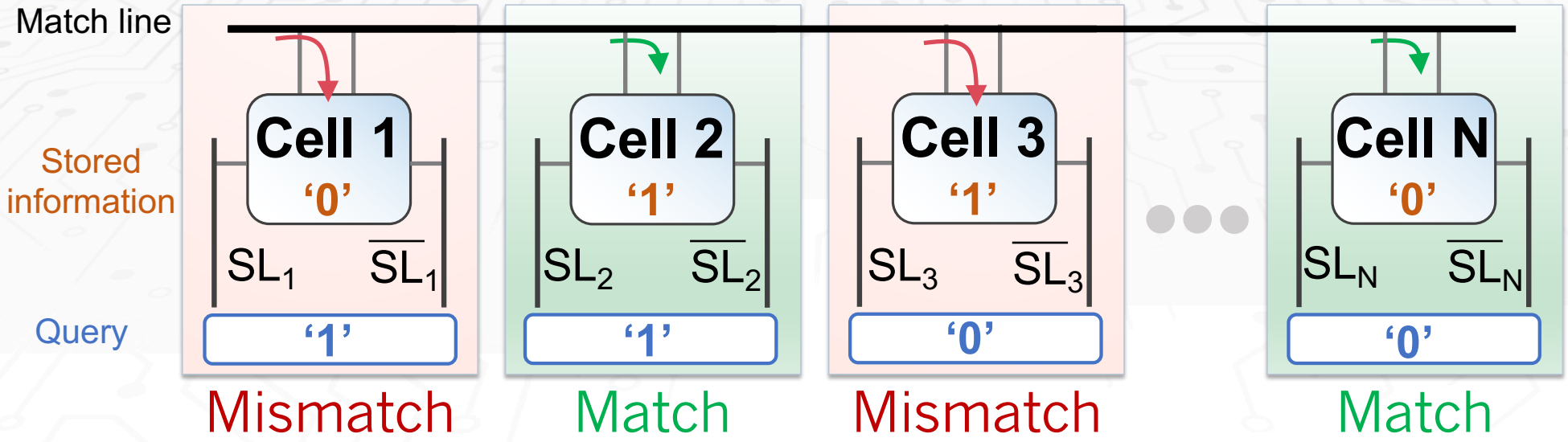


★ Distance between query and stored entries computed within the memory itself, avoiding data transfer overheads



Kai Ni, et al., "Ferroelectric Ternary Content Addressable Memory for One-Shot Learning," ***Nature Electronics, 2019.***

SENSING DEGREE OF MATCH IN TCAM

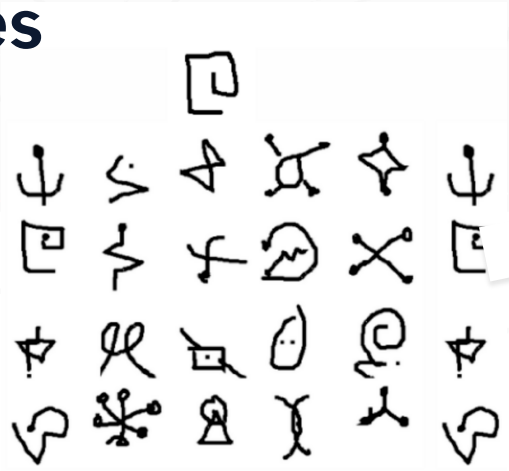


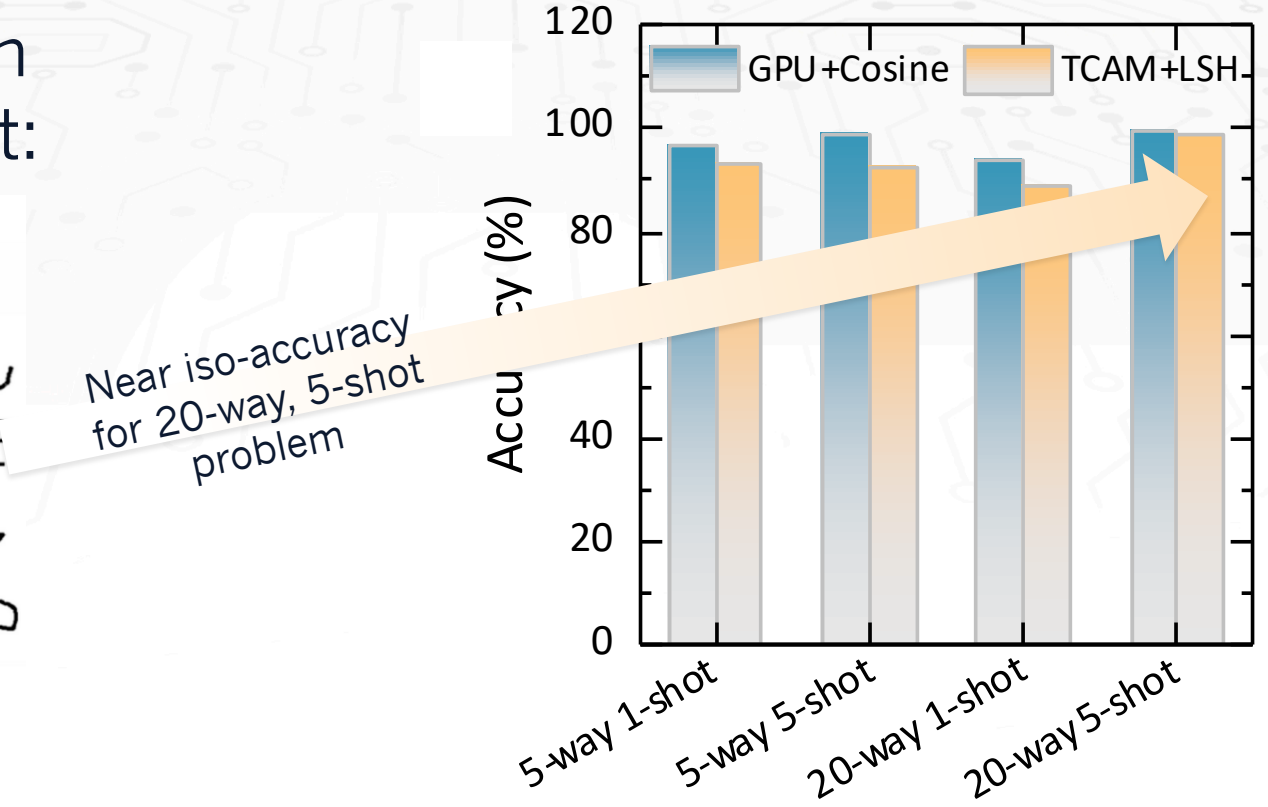
Sensing the discharge current through match line can detect the **degree of match** between stored information and query directly **within memory**.

CAN TCAM KERNELS APPROACH COSINE ACCURACY?

Initial studies with Omniglot data set:

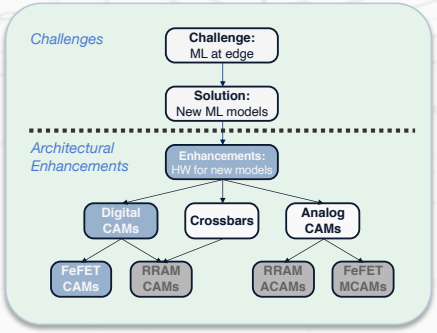
Yes

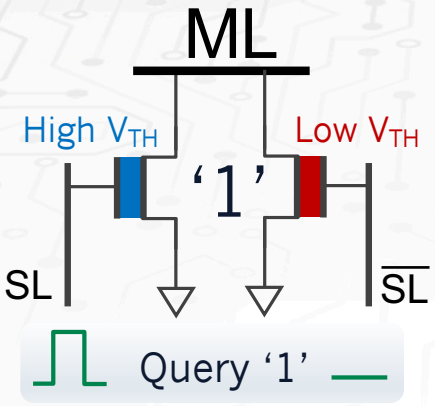


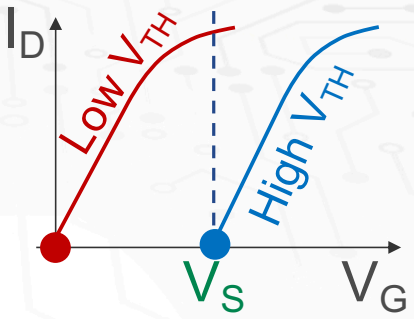


Kai Ni, et al., "Ferroelectric Ternary Content Addressable Memory for One-Shot Learning," **(SRC P095557)**

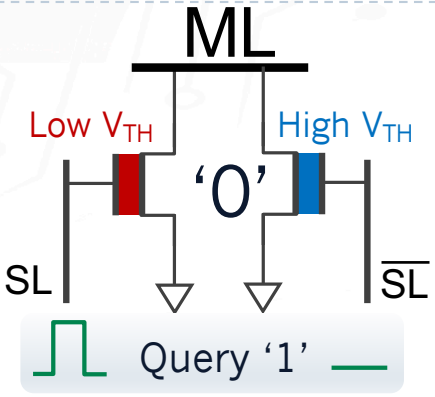
FERROELECTRIC TCAM CELL

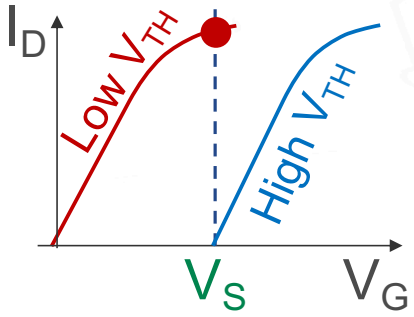






Match
small discharge current

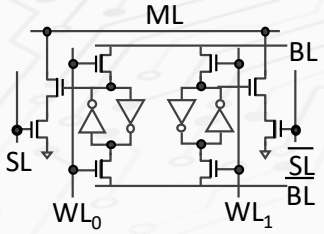
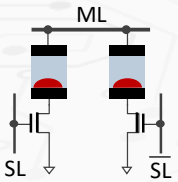
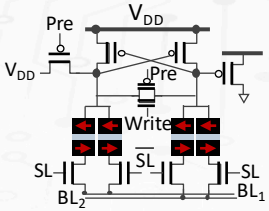
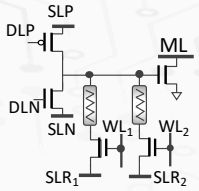
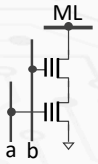
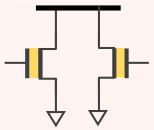




Mismatch
large discharge current

Two FeFETs can realize a compact and high performance TCAM cell.

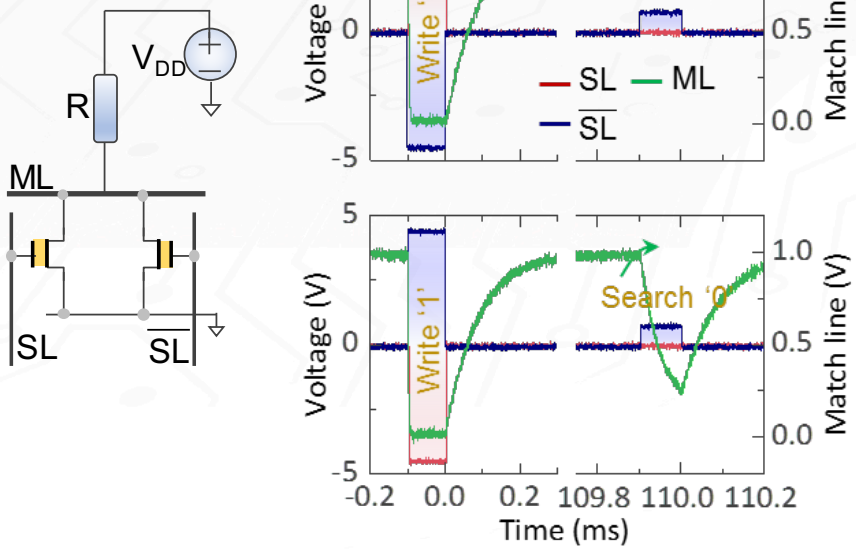
TCAM CELL BENCHMARKING

	16T CMOS	2T-2PCM	10T-4MTJ	2.5T-1ReRAM	2Flash	2FeFET
						
Cell area (μm^2)	1.12	0.41	2.78	0.28	0.30	0.15
Nonvolatility	No	Yes	Yes	Yes	Yes	Yes
R_{ON}/R_{OFF}	~10^6	~100	~2.5	~100	~10^6	~10^4
Search energy (fJ/bit/search)	1.0	0.64	40.5	0.71	0.6	0.4
Search delay (ps)	582	155	1000	155	679	355
Write energy (fJ/bit)	4.8	~4500	870	~720	>98000	1.4

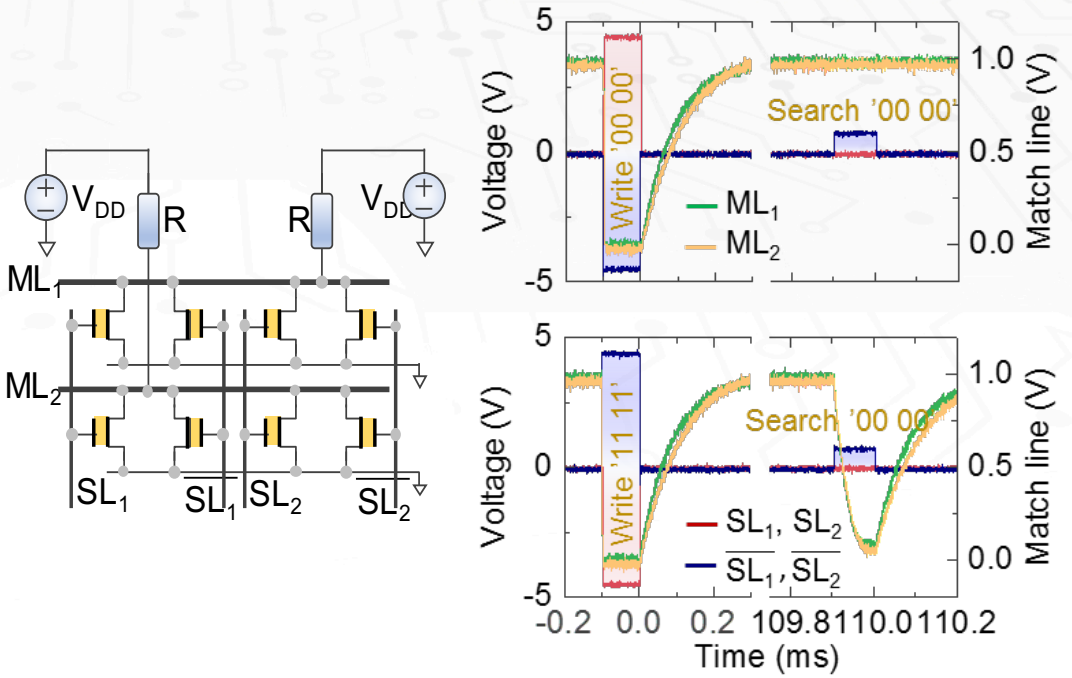
Kai Ni, et al., "Ferroelectric Ternary Content Addressable Memory for One-Shot Learning," in Nature Electronics **(SRC P095557)**

EXPERIMENTAL DEMONSTRATION

Single TCAM Cell



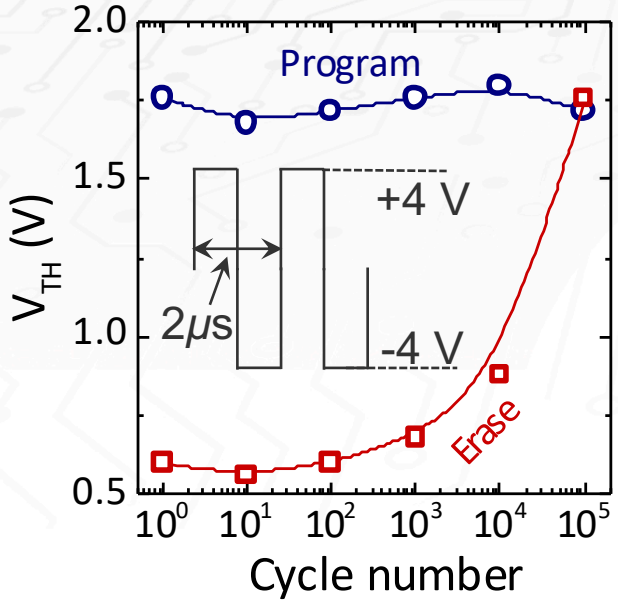
2x2 TCAM Array



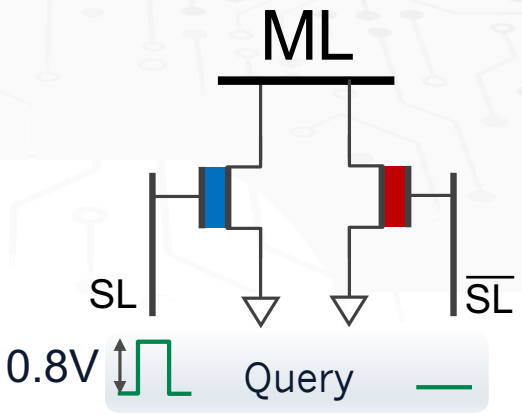
TCAM cell and array operation are successfully demonstrated.

CHALLENGE: FERROELECTRIC ENDURANCE

Write endurance is limited



Search is a read operation



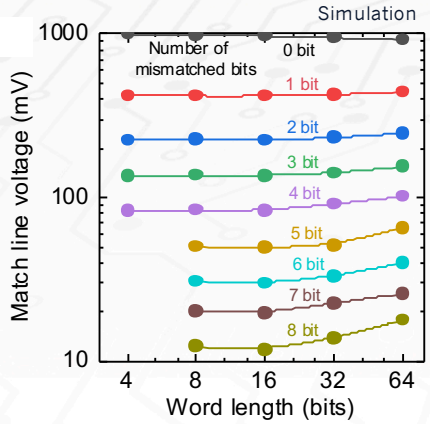
FeFET read endurance is almost infinite, as CMOS

- Write endurance in FeFET remains a challenge to solve
- In TCAM, the most frequent search is a read operation, whose endurance is almost unlimited

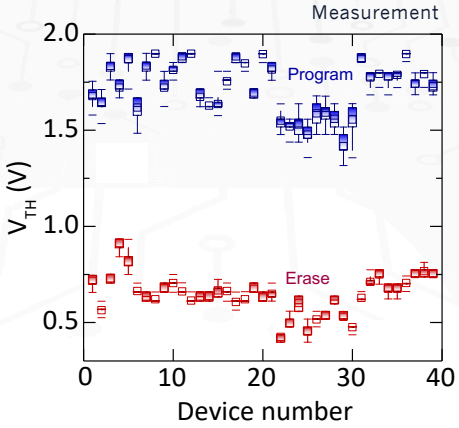
Kai Ni, et al., "Fundamental understanding and control of device-to-device variation in deeply scaled ferroelectric FETs," VLSI 2019

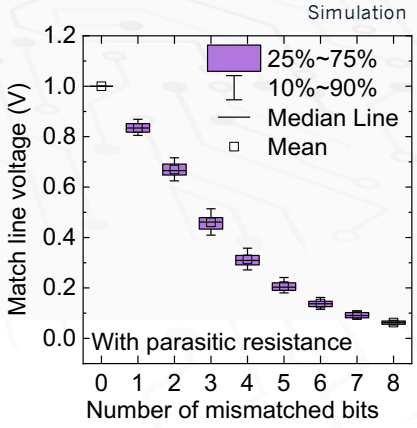
CHALLENGE: DETECTABLE DEGREE OF MATCH?

Detection of up to 8-bits in a 1x64 TCAM array
(ML voltage function of word size)



...even after considering measured device-to-device cycle variations





Hashing function inherently creates similar signatures for similar classes
(relaxes Hamming distance detection requirements – e.g., assume 16 bit or less for Omniglot)


=1111101


=0001101


=1110111

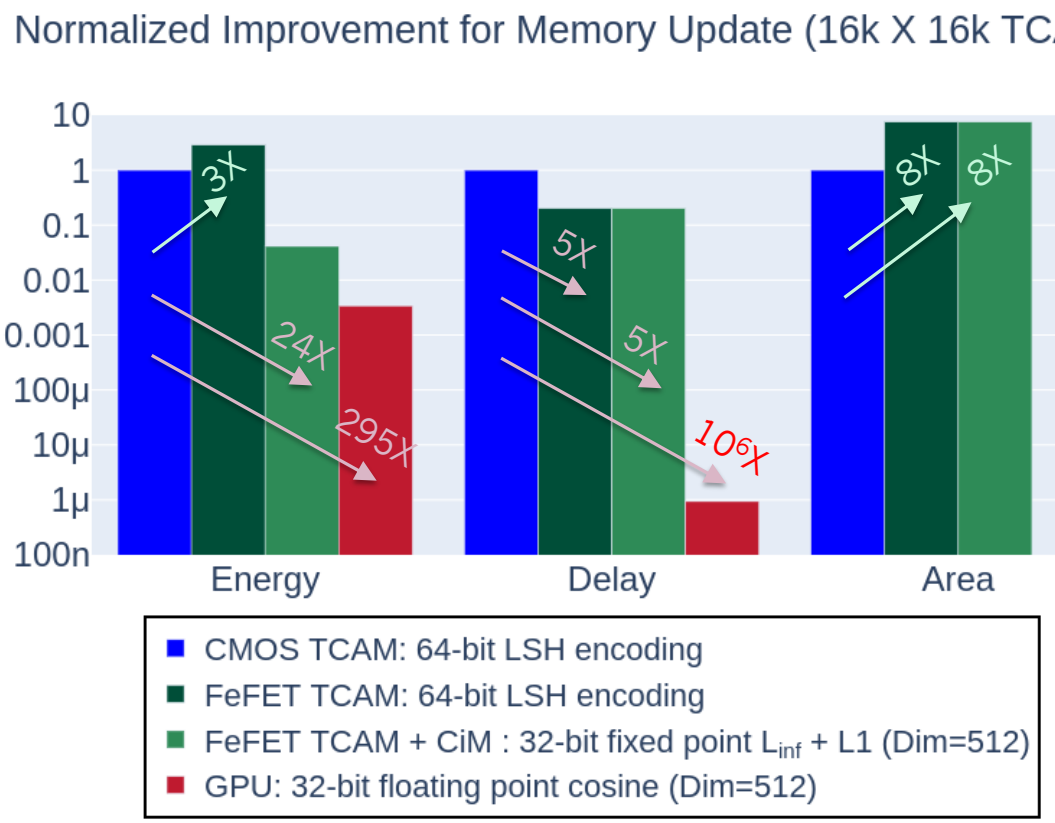

=0001001

How to address? Current-based sense.

Kai Ni, et al., "Ferroelectric Ternary Content Addressable Memory for One-Shot Learning," to appear in Nature Electronics **(SRC P095557)**

ASCENT
Applications and Systems Driven Center for Energy-Efficient Integrated Nanotechnologies

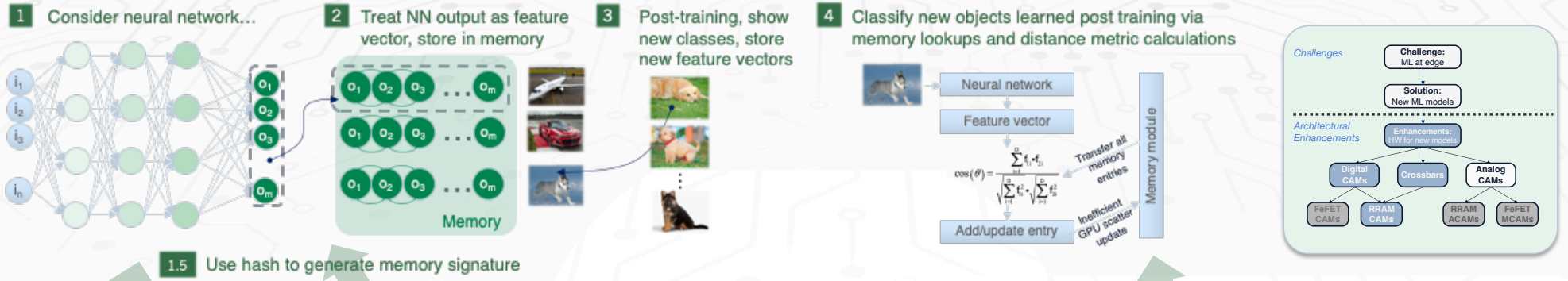
WHAT BENEFIT FROM TECHNOLOGY, ARCHITECTURE?



At array level, FeFET based CAMs more energy and area efficient; area efficiency can improve accuracy

END-TO-END RRAM SOLUTIONS

Let's revisit the MANN-based approach:



In-memory computing with xBars for lower neural network layers

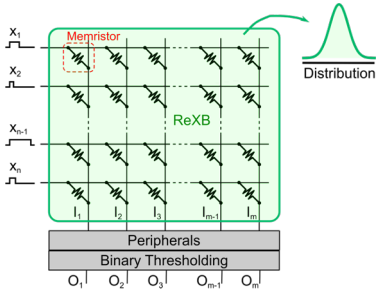
- 100x greater throughput/Watt versus GPUs
- >> benefits for larger networks

A Shafiee, et al., ISCA (2016); M Hu, et al., Adv Mater (2018); C. Li, et al., Nature Electr (2018); A Ankit, *et al*, ASPLOS (2019); A Ankit, *et al*, IEEE Trans. on Comp. (2020); C Li, et al, Intern. Mem. Workshop (2020)

Hashing functions can be done in xBars

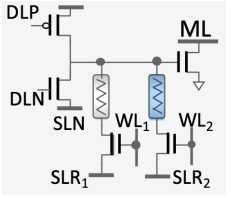
- For LSH hash, need zero-mean, Gaussian distributed matrix
- RRAM xBars are Gaussian distributed post-fab (no zero mean)
- Use differential encoding to transform Gaussian distributed xBar matrix to a zero-mean Gaussian

Kitaev, Nikita, et al., "Reformer: The Efficient Transformer." *ICLR'* 2019.



RRAM CAMs possible

In-memory computing with CAMs for final layer classification

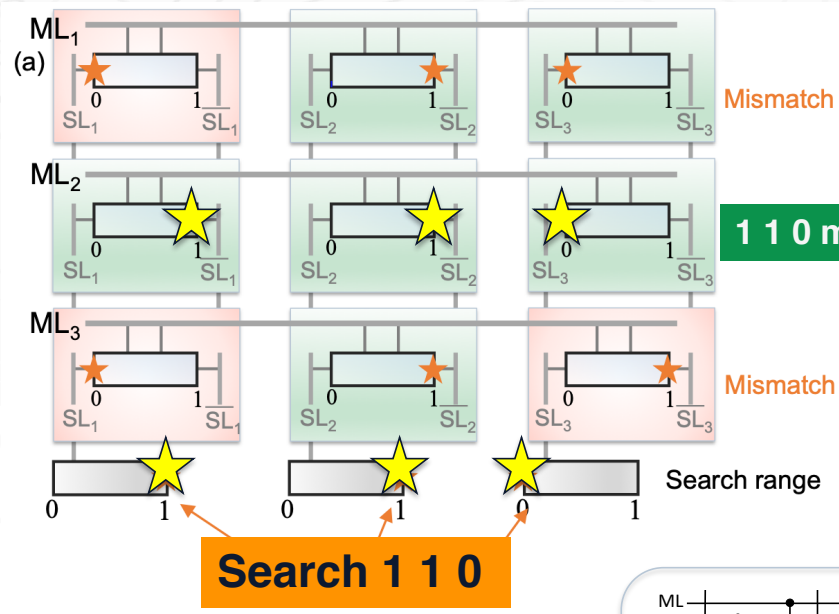


5-20X improved area
4-10X lower energy

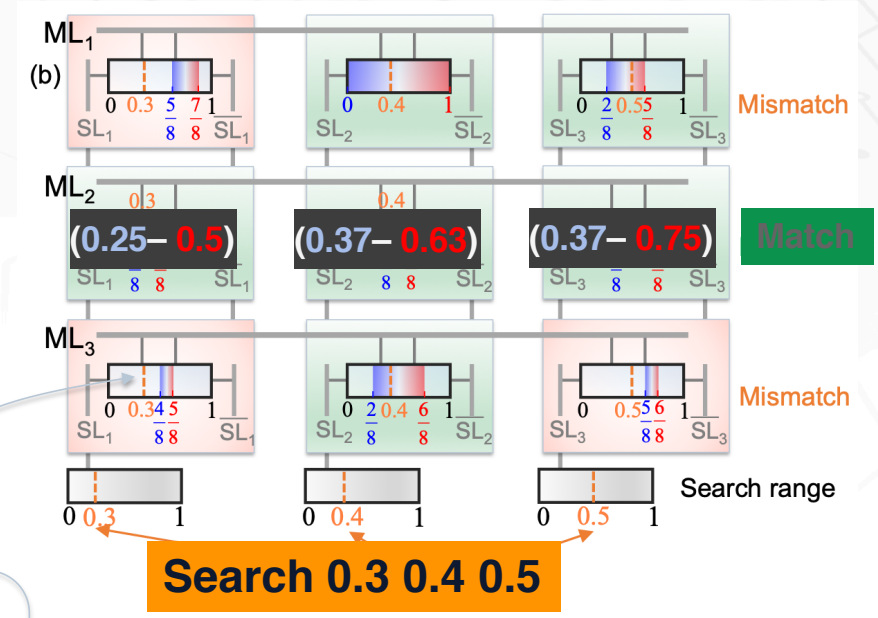
Kai Ni, et al., Nature Electronics (2019)
Can Li, et al., Nature Communications (2020)

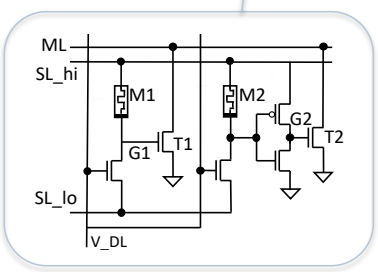
ANALOG CAM DESIGNS

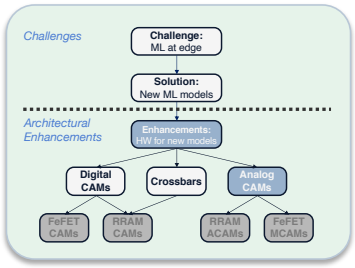
Traditional CAM



Analog CAM



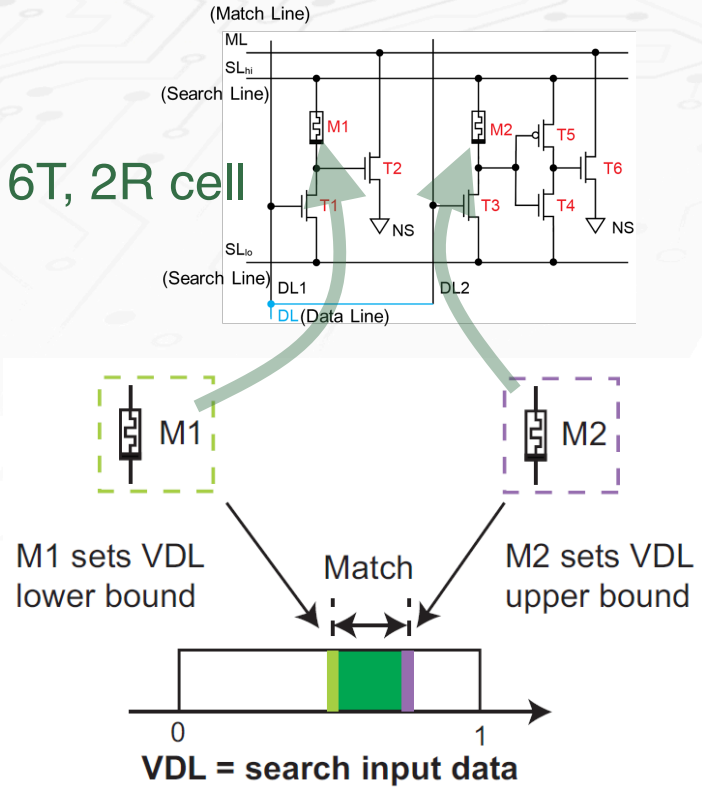




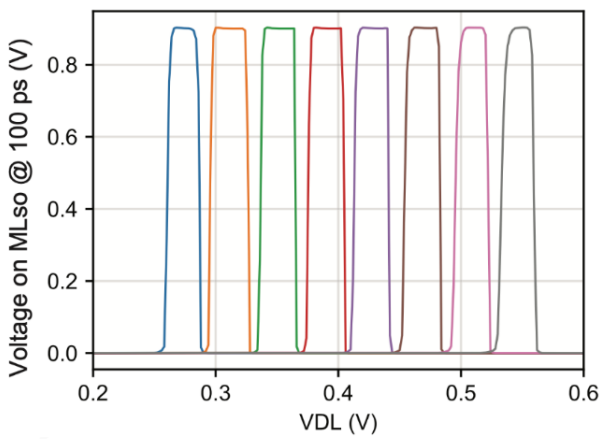
Analog RRAM-based TCAM (6T-2R)
Nature communications 11 (1), 1-8, 2020

ANALOG CAMS FOR LOWER AREA/ENERGY, NEW APPS

RRAM-based ACAMs



Store, search over 8 levels

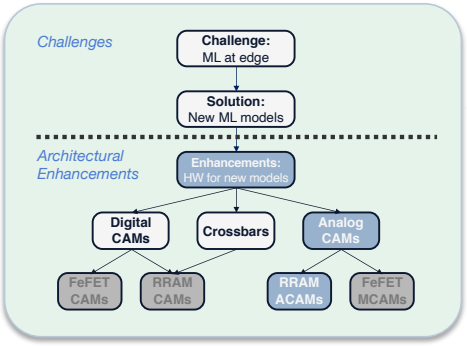


Impact:

Versus standard SRAM TCAMs:

- **>18x reduction in Area,**
- **> 4x lower energy/bit/search**
- Non-volatile CAM

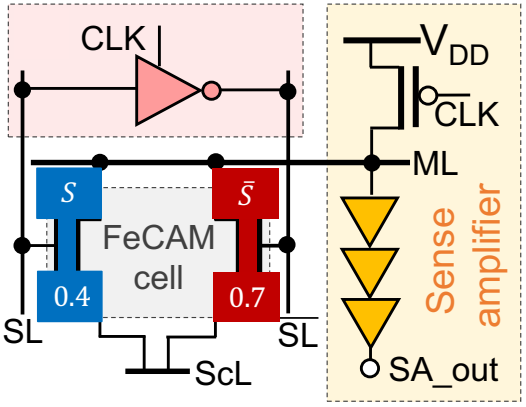
New range-storing capability enables new applications: {decision trees/random forests, probabilistic computing, ...}



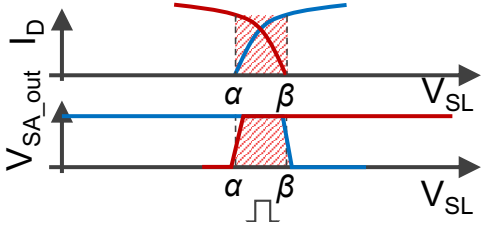
Can Li et al, Nature Communications 11, 1 (2020)

WHAT ARE THE NEXT STEPS AND BENEFITS?

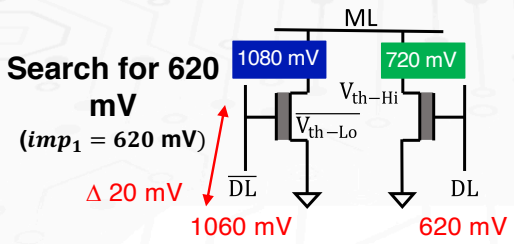
FeFET design can also serve as digital, analog, or **multi-bit** CAM

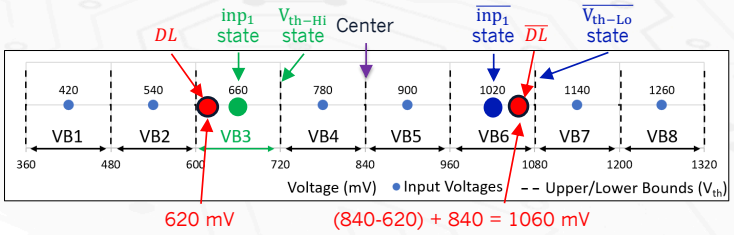


FeFETs store upper/lower bound



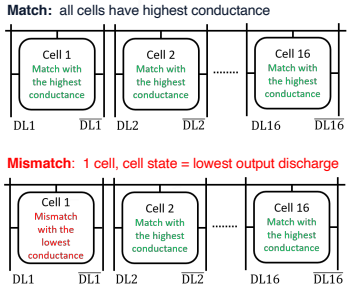
Programming cells into multi-bit states/bands **improves noise tolerance**

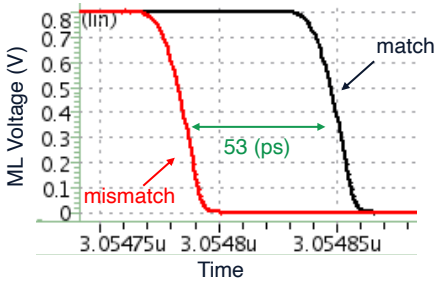


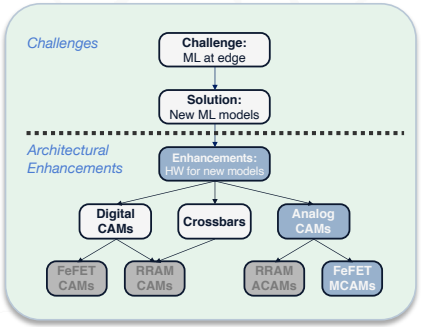


High conductance match, more ML discharge

Even in worst-case, correct match detectable



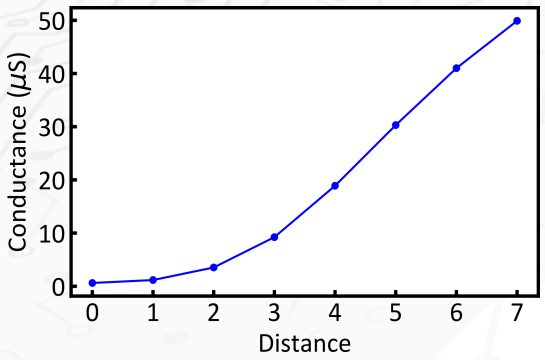




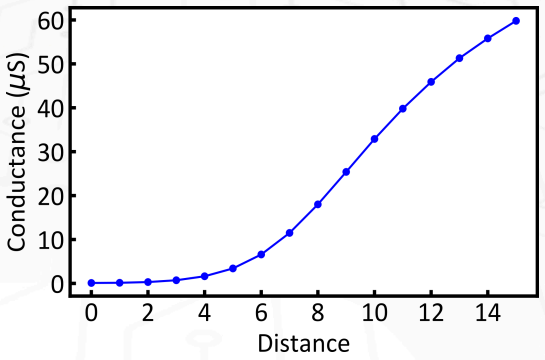
WHAT ARE THE NEXT STEPS AND BENEFITS?

Cell conductance increases exponentially WRT differences between stored, searched value

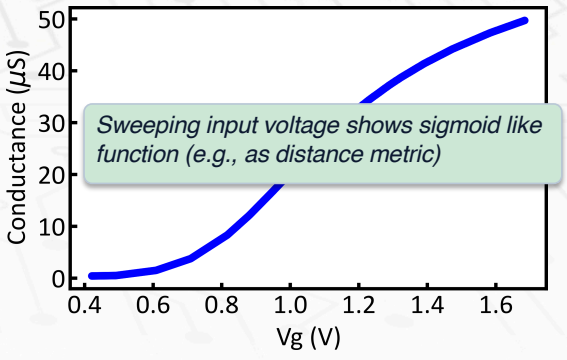
Single 3-bit cell



Single 4-bit cell

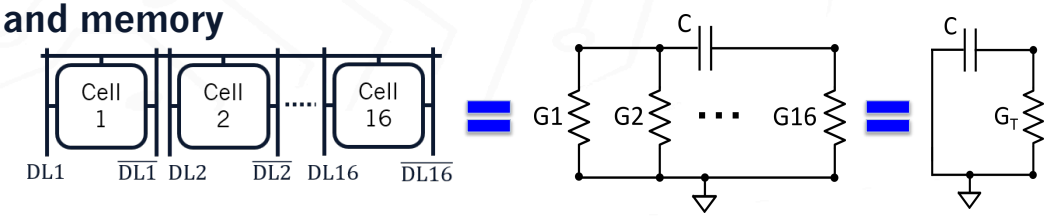


Sweeping input V for cell



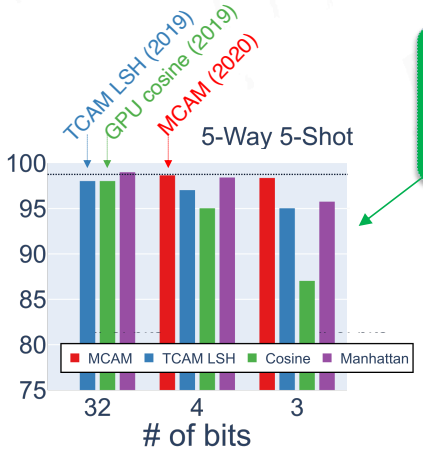
Sweeping input voltage shows sigmoid like function (e.g., as distance metric)

In 16-column array, row conductance represents the distance between input and memory



$G_T = G_1 + G_2 + \ldots + G_{16}$

e.g., for ML each cell represents a feature
- *Conductance represents distance*
- *>> conductance = >> distance*

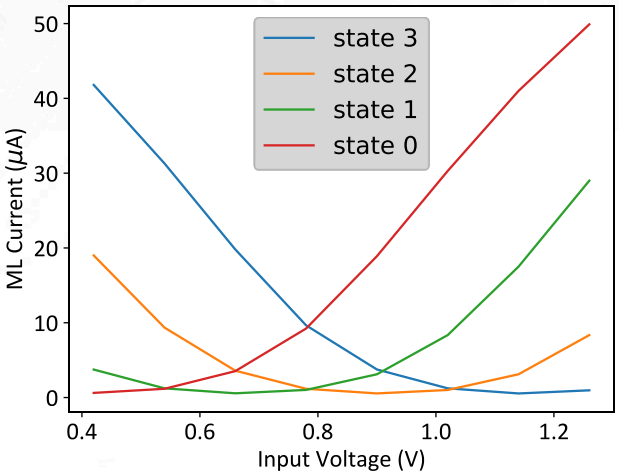


Improved accuracy *and* reduced computational overhead (no LSH) *and* 3X cell reduction with MCAM

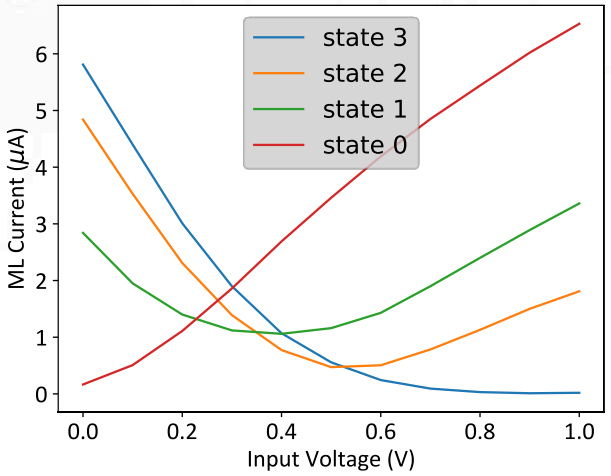
DEMONSTRATIONS OF FEFET MCAMS

- **Initial experimental results approach simulation-based data**
- **Application-level analysis still promising**

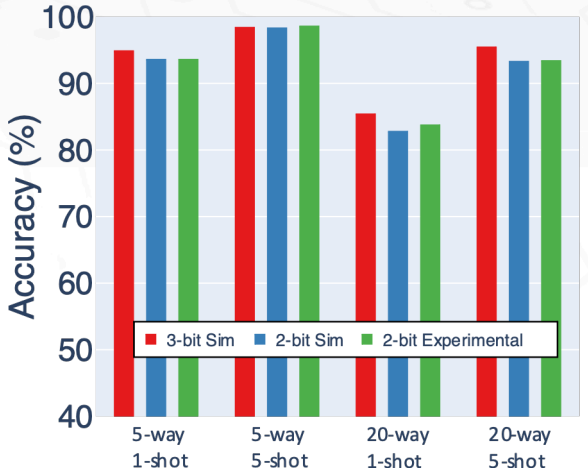
Simulation-based results



Initial experimental measurements*

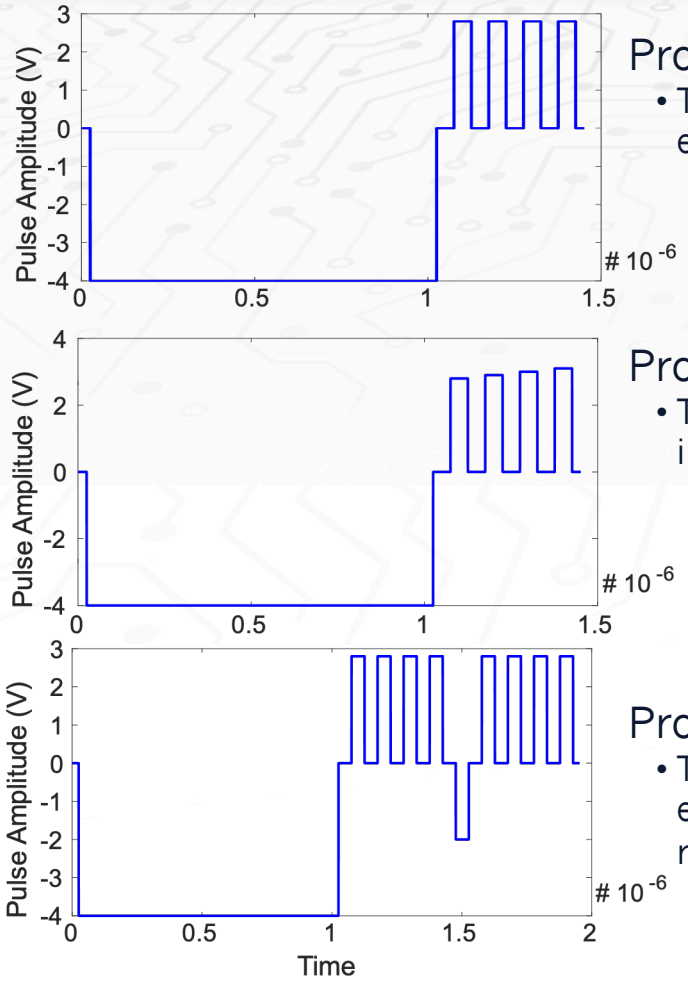


Few-shot learning results



*Results obtained from FeFET AND arrays by Franz Müller and Thomas Kämpfe from Fraunhofer IPMS center

MULTI-BIT FEFETs: PROGRAMMING SCHEMES



Program Scheme 1:
- Train of pulse with equal amplitude

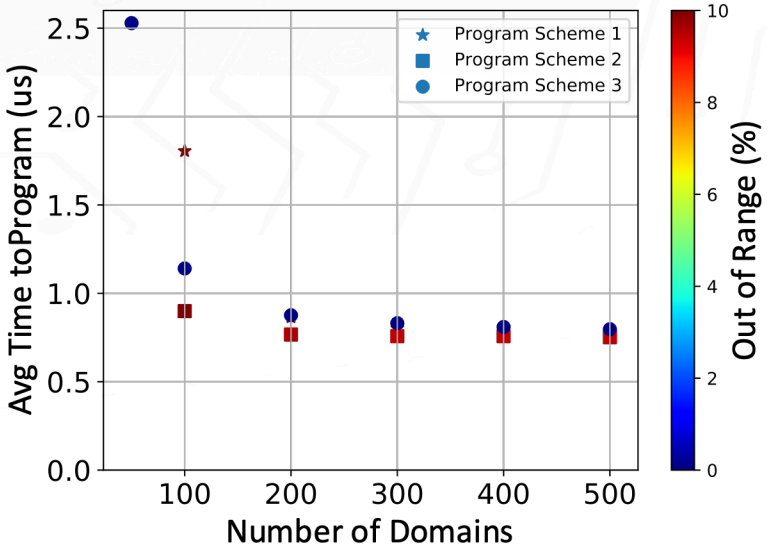
Program Scheme 2:
- Train of pulse with increasing amplitude

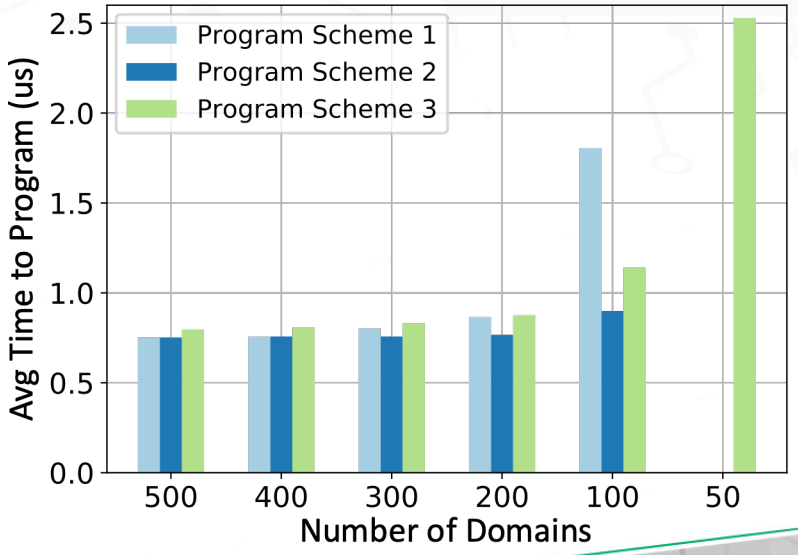
Program Scheme 3:
- Train of pulse with equal amplitude with reset when over set

- Each simulation batch includes 500 devices programmed to 8 states
- Verification done until device reaches desired current range
- Programming schemes 1, 2 only provide lower bound for current range target
- Programming scheme 3 provides lower and upper current range target; uses soft reset when device is over-set

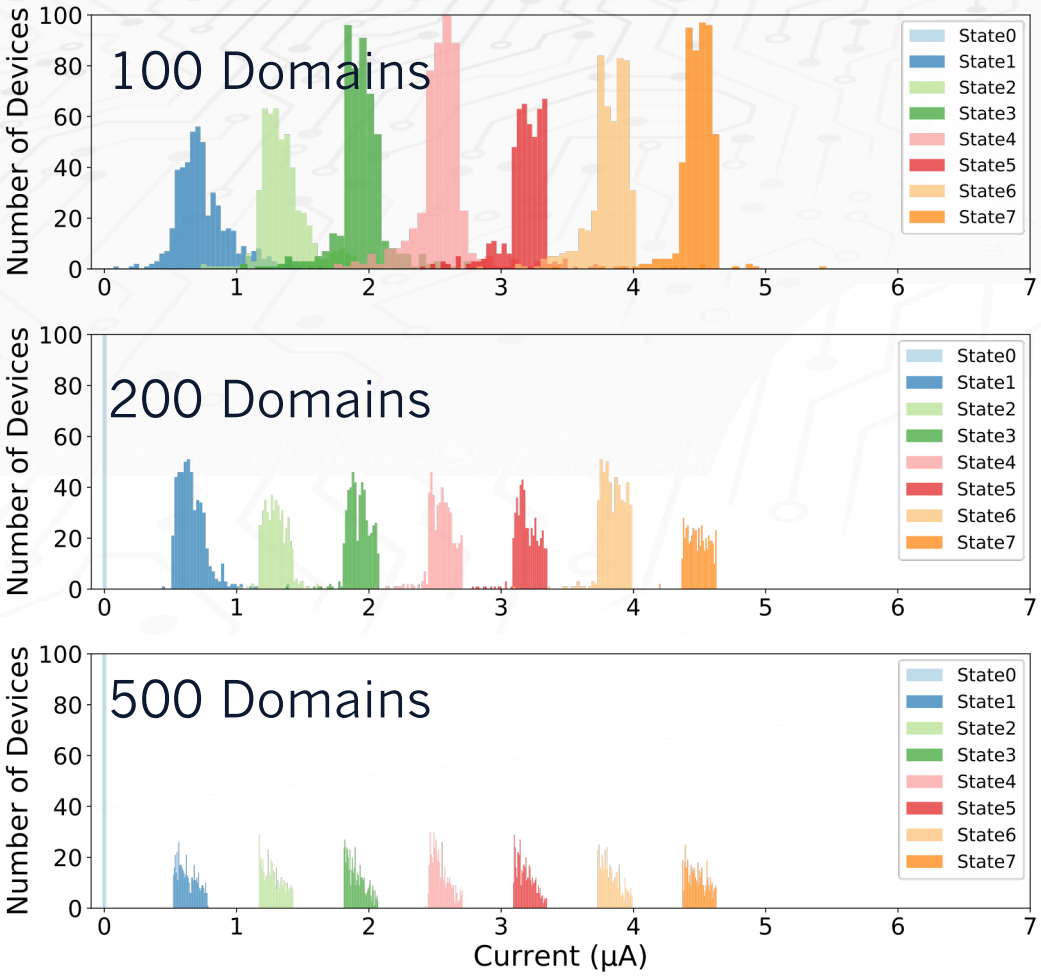
MULTI-BIT FEFETs: PROGRAMMING SCHEMES

- Avg Time to program $= \frac{\sum_1^n Time\ to\ program\ device_n}{n}$, n=Number of devices
  - Done for all states; mean considered in plots
- Device Out of Range $= \frac{Number\ of\ devices\ out\ of\ the\ desired\ current\ range}{n} * 100$
- Batches with more than 10% of devices out-of-range discarded
- For each domain size, fastest programming time found, shown in the plots below





MULTI-BIT FEFETs: PROGRAMMING SCHEME 3



- Simulations of 500 devices
- Programming pulse width = 100(ns)
- Reset pulse width = 100(ns)
- Reset amplitude = -2(v)

Preliminary results; working to improve

HOMOMORPHIC ENCRYPTION

WHAT IS HOMOMORPHIC ENCRYPTION?

Computing in the cloud:

User data: 17 → **Encrypted:** 27142
24 → **Encrypted:** 98716

Cloud: {27142, 98716}

Not private!

Process:
27142 → 17 **(decrypted)**
98716 → 24 **(decrypted)**
17+24 = 41
41 → 56817 **(encrypted)**

Computing in the cloud (encrypted data):

User data: 17 → **Encrypted:** $2x^3+3x^2+5x+6$
24 → **Encrypted:** $7x^3+2x^2+2x+2$

Cloud: $\{2x^3+3x^2+5x+9,\ 7x^3+2x^2+2x+2\}$

Process:

$$\begin{array}{rl} & 2x^3+3x^2+5x+6 \\ + & \underline{7x^3+2x^2+2x+2} \\ & 9x^3+5x^2+7x+8 \end{array}$$

Data not decrypted in cloud

To user: $9x^3+5x^2+7x+8$ → 41 **(decrypted)**

Different schemes may be employed – e.g., BGV, GSW, TFHE, CKKS

HOW HOMOMORPHIC ENCRYPTION WORKS (PART 01)

Step 01: Encode plaintext in polynomial $R_t = \mathbb{Z}_t[x]/\Phi_m(x)$

17 → Encode: $2x^3+3x^2+5x+6$ **24** → Encode: $7x^3+2x^2+2x+2$

Can be done with standard encoding techniques: {integer encoding, fraction encoding}

Step 02: Encrypt into pair of polynomials – i.e., *ciphertext polynomials*

R_q = ciphertext: c=(c[0], c[1]) $R_q = \mathbb{Z}_q[x]/\Phi_m(x)$

17 → $2x^3+3x^2+5x+6$ → **$c_1[0]$:** $3432x^{8191} + 943x^{8190} + \ldots + 90x^1 + 3$, **$c_1[1]$:** $2901x^{8191} + 890x^{8190} + \ldots + 231x^1 + 56$

24 → $7x^3+2x^2+2x+2$ → **$c_2[0]$:** $1109x^{8191} + 134x^{8190} + \ldots + 23x^1 + 7$, **$c_2[1]$:** $8734x^{8191} + 456x^{8190} + \ldots + 520x^1 + 345$

Polynomial complexity can increase (i.e., for >> depth): **e.g., $c_2[0]$:** $2164x^{16383} + 98x^{16382} + \ldots + 11x^1 + 1$

HOW HOMOMORPHIC ENCRYPTION WORKS (PART 02)

Homomorphic Add:

Ciphertext: $\boldsymbol{c_1} = (c_1[0], c_1[1]), \boldsymbol{c_2} = (c_2[0], c_2[1])$

Return: $\boldsymbol{c_{1+2}} = \left([c_1[0] + c_2[0]]_q, [c_1[1] + c_2[1]]_q\right)$

Homomorphic Mult:

Ciphertext: $\boldsymbol{c_1} = (c_1[0], c_1[1]), \boldsymbol{c_2} = (c_2[0], c_2[1])$

Compute: $\boldsymbol{c_{1\cdot 2}} = (c_1[0]c_1[0]\ , c_1[0]c_2[1] + c_2[0]c_1[1]\ , c_1[1]c_2[1])$

Return: $\underbrace{\left(\boldsymbol{c_{1\cdot 2}}[0] + RELIN(\boldsymbol{c_{1\cdot 2}}[2]), \boldsymbol{c_{1\cdot 2}}[1] + RELIN(\boldsymbol{c_{1\cdot 2}}[2])\right)}$

Re-linearization brings ciphertext down to 2 elements

PolyAdd

q: 10 d: 4

a = $-2x^3+3x^2-x+4$

b = $3x^3+4x^2+0x+4$

y = x^3+7x^2-x+8

Reduce to ensure in ring

y = x^3-3x^2-x-2

e.g., $[7]_{10}=-3$

$-3 \in \left[-10/2, 10/2\right)$

PolyMult *Typically dominates runtime*

q: 10 d: 4

a = $-2x^3+3x^2-x+4$

b = $3x^3+4x^2+0x+4$

y = $\dfrac{-6x^6+x^5+9x^4+28x^2-4x+16}{x^4+1}$

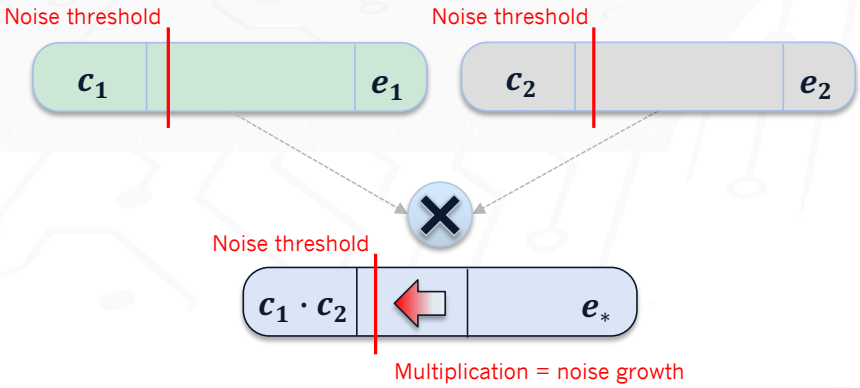
Polynomial remainder

$34x^2-5x+4$ —***reduction***→ $4x^2-5x-3$

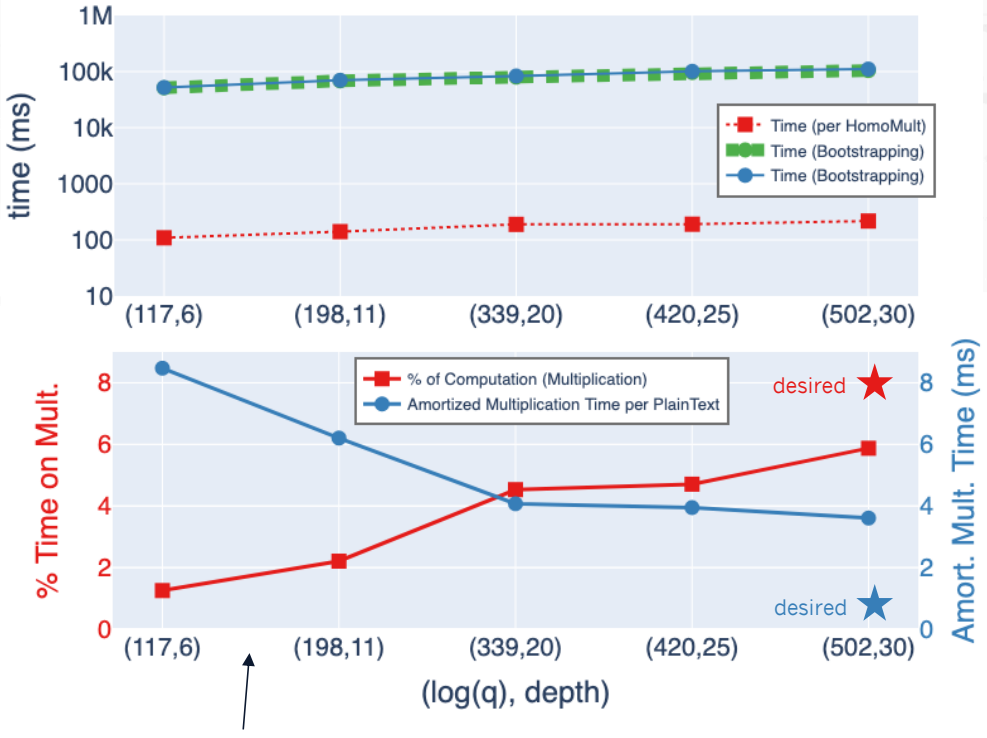
CHALLENGES WITH HOMOMORPHIC ENCRYPTION

Overhead!

Bootstrapping:
Limited # of computations possible before denoising or decrypt/encrypt



Bootstrapping introduces substantial compute overhead



1024 plaintexts, 1024-point logistic function with Taylor approximation

CHALLENGES WITH HOMOMORPHIC ENCRYPTION

Size!

Encrypt into pair of polynomials – i.e., *ciphertext polynomials*

R_q = ciphertext: c=(c[0], c[1]) $R_q = \mathbb{Z}_q[x]/\Phi_m(x)$

17 → $2x^3+3x^2+5x+6$ → $c_1[0]$: $3432x^{8191} + 943x^{8190} + \ldots + 90x^1 + 3$, $c_1[1]$: $2901x^{8191} + 890x^{8190} + \ldots + 231x^1 + 56$

24 → $7x^3+2x^2+2x+2$ → $c_2[0]$: $1109x^{8191} + 134x^{8190} + \ldots + 23x^1 + 7$, $c_2[1]$: $8734x^{8191} + 456x^{8190} + \ldots + 520x^1 + 345$

With increased security, polynomial complexity increases: e.g., $c_2[0]$: $2164x^{16383} + 98x^{16382} + \ldots + 11x^1 + 1$

Polynomial complexity increases (n, log(q))

1.72 MB

depth	128b security	192b security	256b security
4	(8192, 218)	(8192, 152)	(8192, 118)
6	(16384, 438)	(16384, 305)	(16384, 237)
8	(16384, 438)	(32678, 611)	(16384, 237)
10	(16384, 438)	(32678, 611)	(16384, 237)
12	(32678, 881)	(32678, 611)	(32678, 476)

236 KB

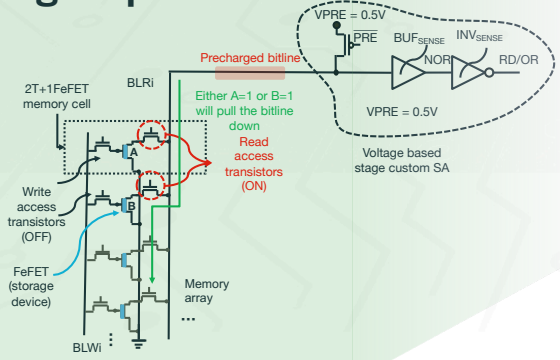
Requirements for given security level (from HE standard)

CIM HE HARDWARE INFRASTRUCTURE

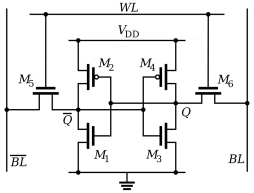
Array dimensions based on ciphertexts/bank, coefficient modulus, polynomial degree

Coefficients of same degree in same column

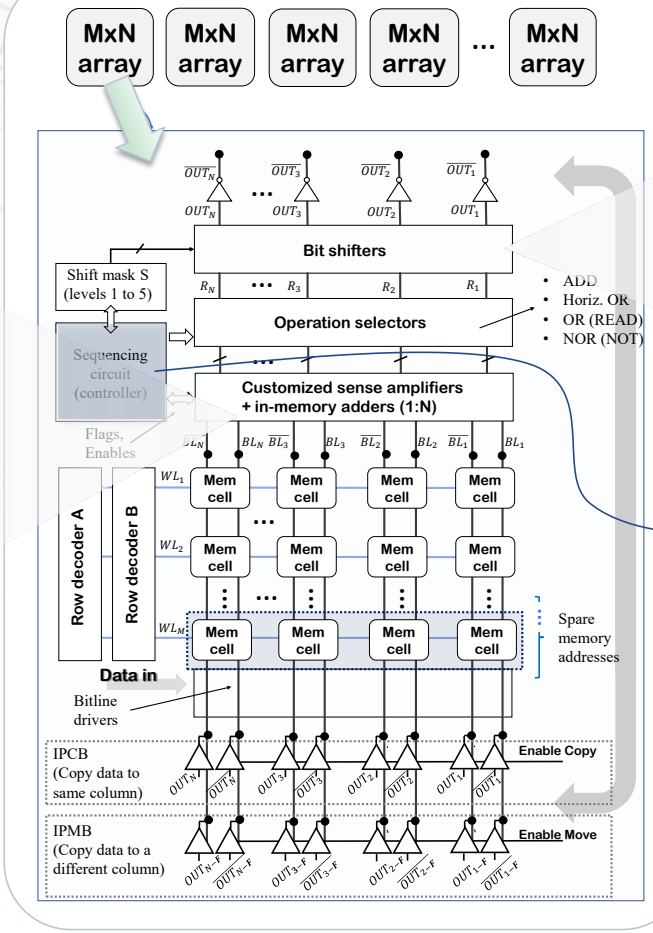
Bitwise arithmetic, logic operations



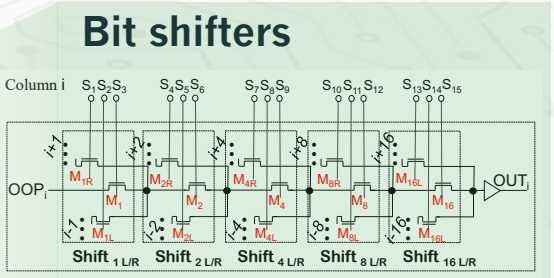
Begin with SRAM baseline, then ASCENT-centric solutions



Reis, et al., (**SRC P097914**)



Bit shifters



Sequencing circuits

e.g., reduction of integer X modulo $q=2^k$

1. Store mask with 1s, 0s
2. Perform horizontal OR
3. Conditional HomSub

Can support SHE, FHE (with Bootstrapping)

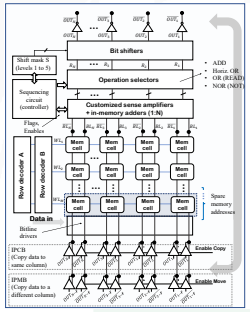
CIM WITH ALGORITHMIC OPTIMIZATION

1. Full-Residue Number System (RNS)

- Full-RNS cannot be implemented when moduli are powers of 2 (initial work);
- With **Barrett modular reduction**, the moduli can be any integer (not limited to powers of 2).

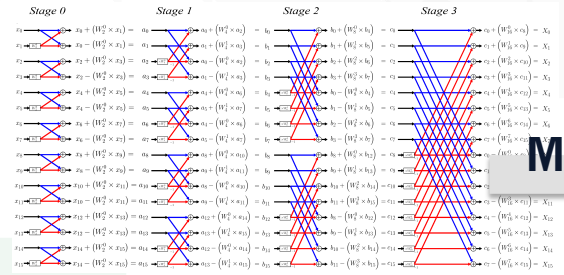
```
word barrett_reduce(dword x, PRECOMPUTED word modulus){
  PRECOMPUTED word expk = 1;
  while(expk < modulus){
    expk <<= 1;
  }
  PRECOMPUTED word m = expk/modulus;
  dword q = (x*m) >> expk;
  x -= q*modulus;
  if(x >= modulus){
    x -= modulus;
  }
  return (word) x;
}
```

Mapping

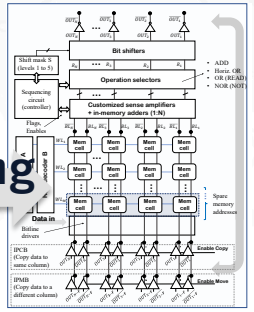


2. Number-Theoretic Transform (NTT)

- NTT enables pointwise multiplication
- We use it in CiM-HE **to reduce the complexity/runtime of homomorphic multiplications**;



Mapping



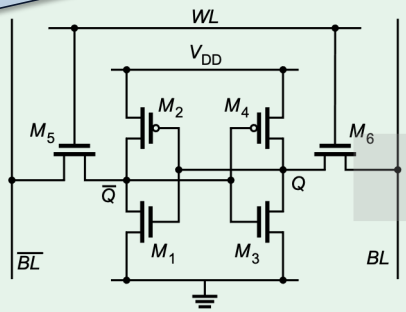
CiM-HE Figures-of-Merit with and without optimizations:

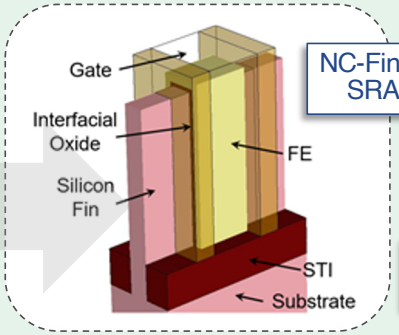
Optimizations?	HomMult Time
NO	6.601 ms
YES	0.121 ms

Improvement w.r.t. CPU: 250.95x **time**

TECHNOLOGY SPECIFIC STUDIES

SRAM baselines

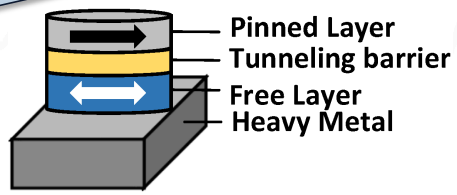


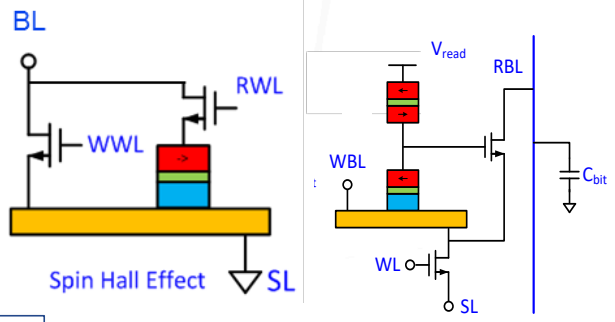


NC-FinFET SRAM

Also, 3D SRAM

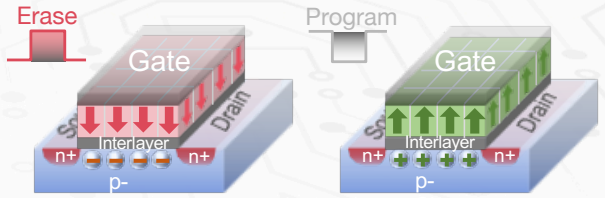
SOT-MRAM

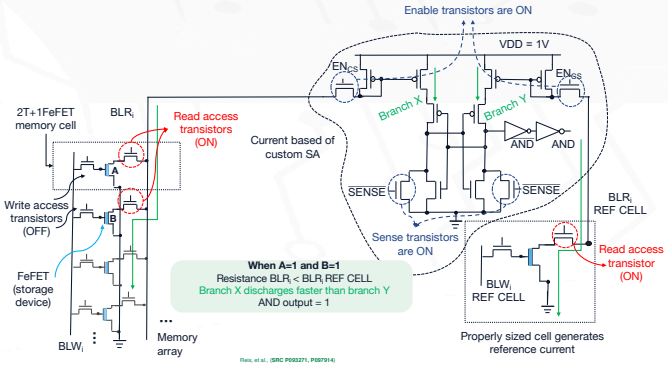




Collaborate with Azad Naeemi

FeFETs





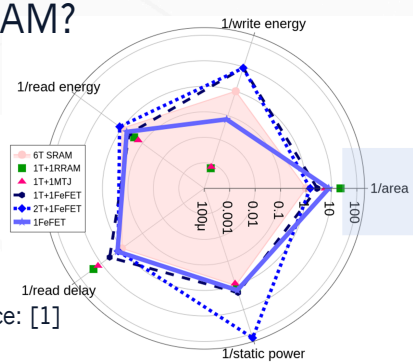
Leverage existing FeFET-based CIM studies/designs

TECHNOLOGY SPECIFIC STUDIES

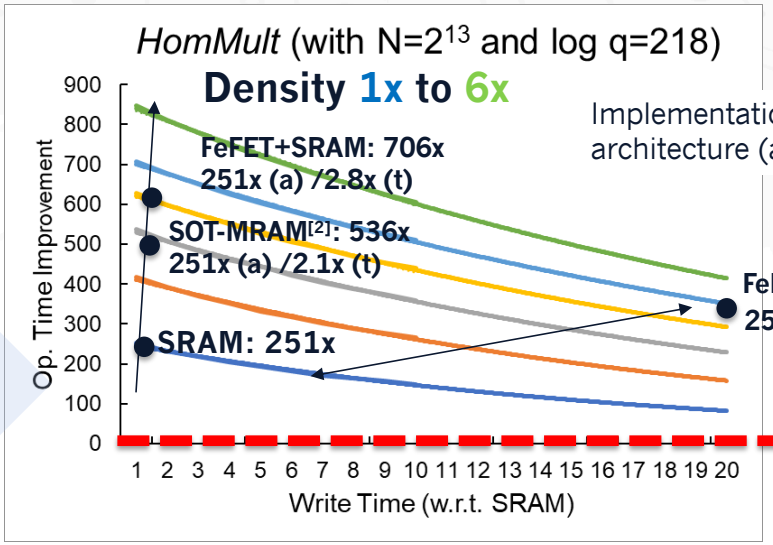
Target: estimate impact of memory technology on CiM-HE

CiM and CPU *both* implement algorithm optimizations

- Op time breakdown (e.g. *HomMult*):
  50.6% adds, **19.2%** shifts, **29.3%** writes, **<1.0%** controller
- How does a technology compare to SRAM?



Source: [1]



[1] D. Reis et al. Design and Analysis of an Ultra-Dense, Low-Leakage, and Fast FeFET-Based Random Access Memory Array. *IEEE JxCDC*, *5*(2), 103-112.
[2] F. Oboril et al, "Evaluation of Hybrid Memory Technologies Using SOT-MRAM for On-Chip Cache Hierarchy," in IEEE TCAD, vol. 34, no. 3, pp. 367-380, March 2015, doi: 10.1109/TCAD.2015.2391254.

WRAP-UP

FUTURE WORK

Combine ASCENT-centric technology driven architectures with work in CRISP

- Already doing so in context of HD computing

Funded via JUMP mid-program realignment :

- X. Sharon Hu, José Martínez, Michael Niemier
- “Hardware & Programming Models Supporting In-Memory Accelerators for Secure Information Processing”

Project will emphasize CiM for high-speed, low-energy block ciphers (AES) and homomorphic encryption

- Explore different CiM architectures, circuit design, and trade-offs
- Develop programming models/tools for for SoCs with CiM-based accelerators
- Benchmark impact of technology on CiM fabrics given technologies under study

THANK YOU!
QUESTIONS?

JUMP

Joint University Microelectronics Program

www.src.org/program/jump

Semiconductor Research Corporation

@srcJUMP