Secure Digital In-Memory Compute for Machine Learning Applications

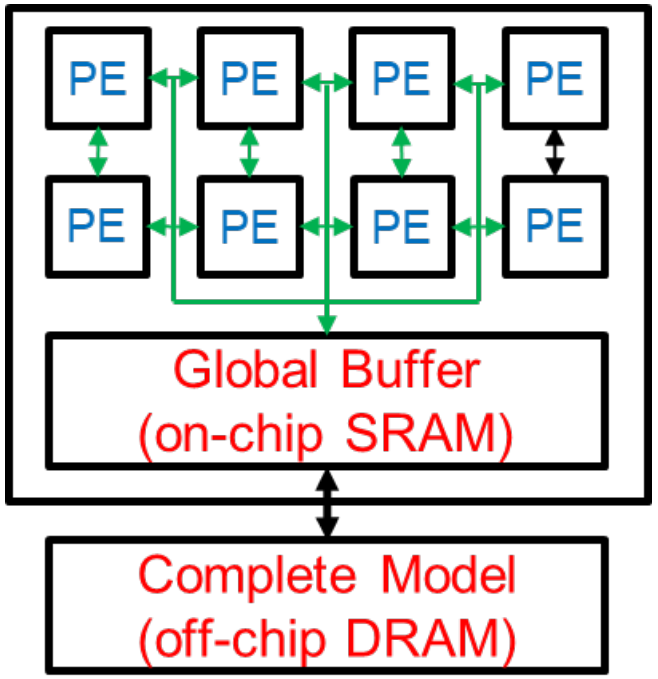
Maitreyi Ashok

PI: Professor Anantha Chandrakasan

Work in collaboration with Saurav Maji (formerly MIT), Xin Zhang (IBM), John Cohn (IBM)

In Memory Compute

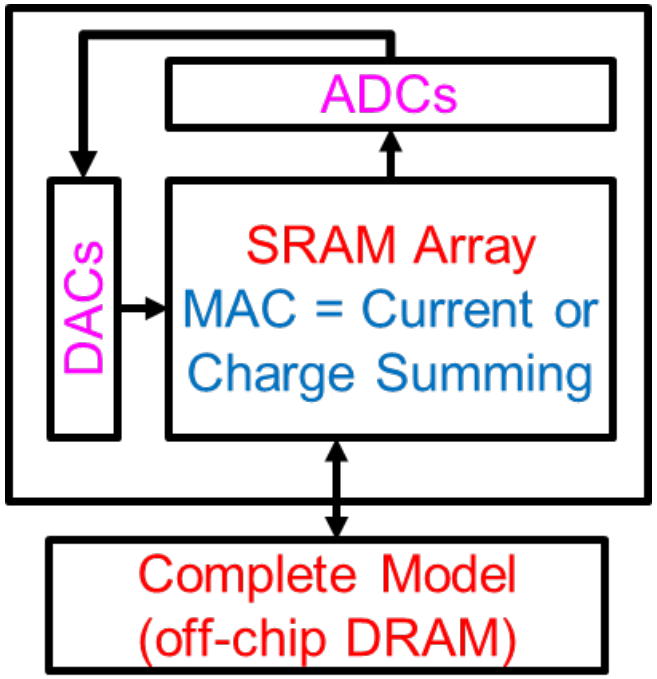
Traditional ML Accelerators



Compute and **Memory** separate

Large **NoC** → data movement energy limits the accelerator

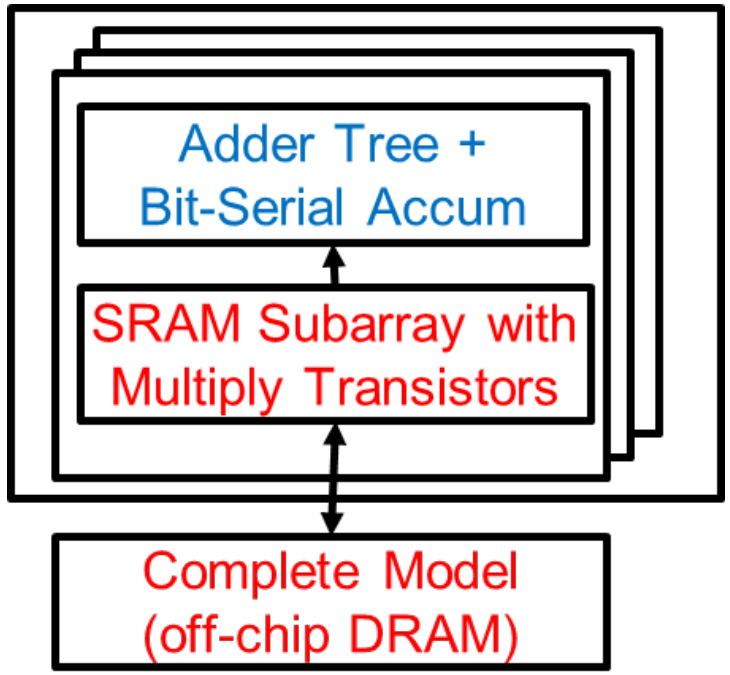
Analog In-Memory Compute



Compute and **Memory** co-located for low data movement energy

High **DAC/ADC** energy, noise, mismatch limit precision

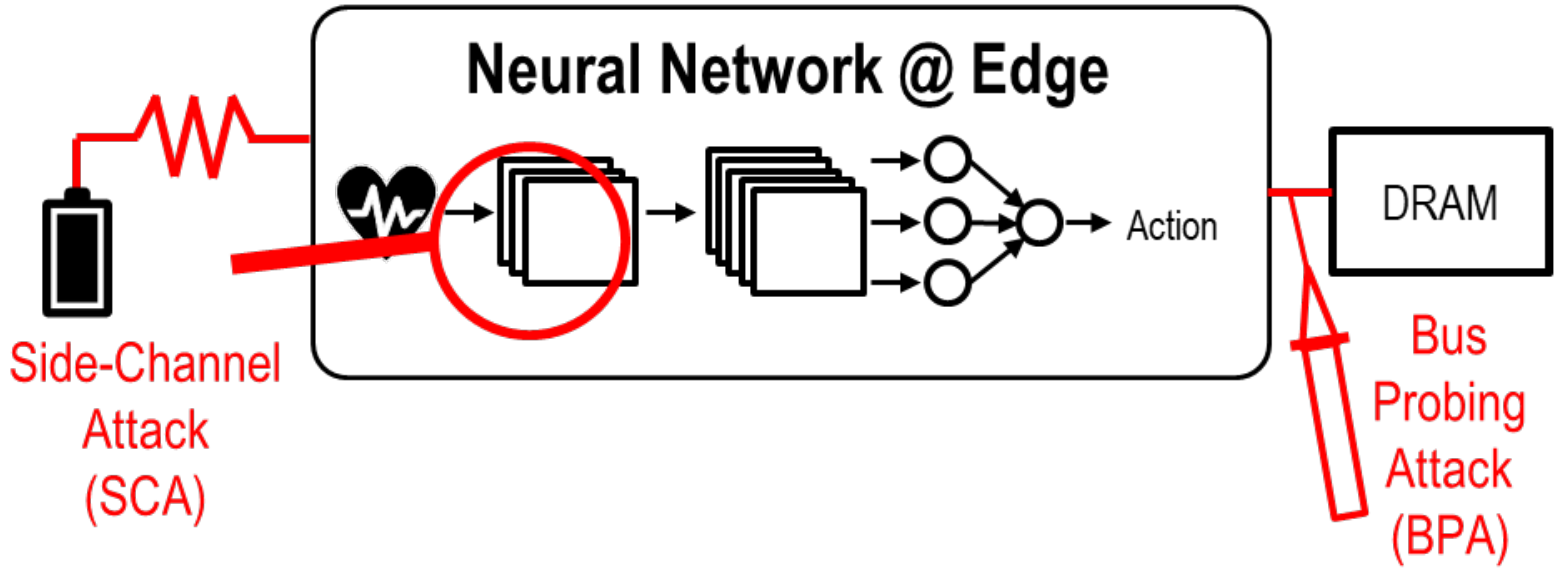
Digital In-Memory Compute



Compute and **Memory** interleaved for lower data movement energy

Does not limit precision and benefits from technology scaling

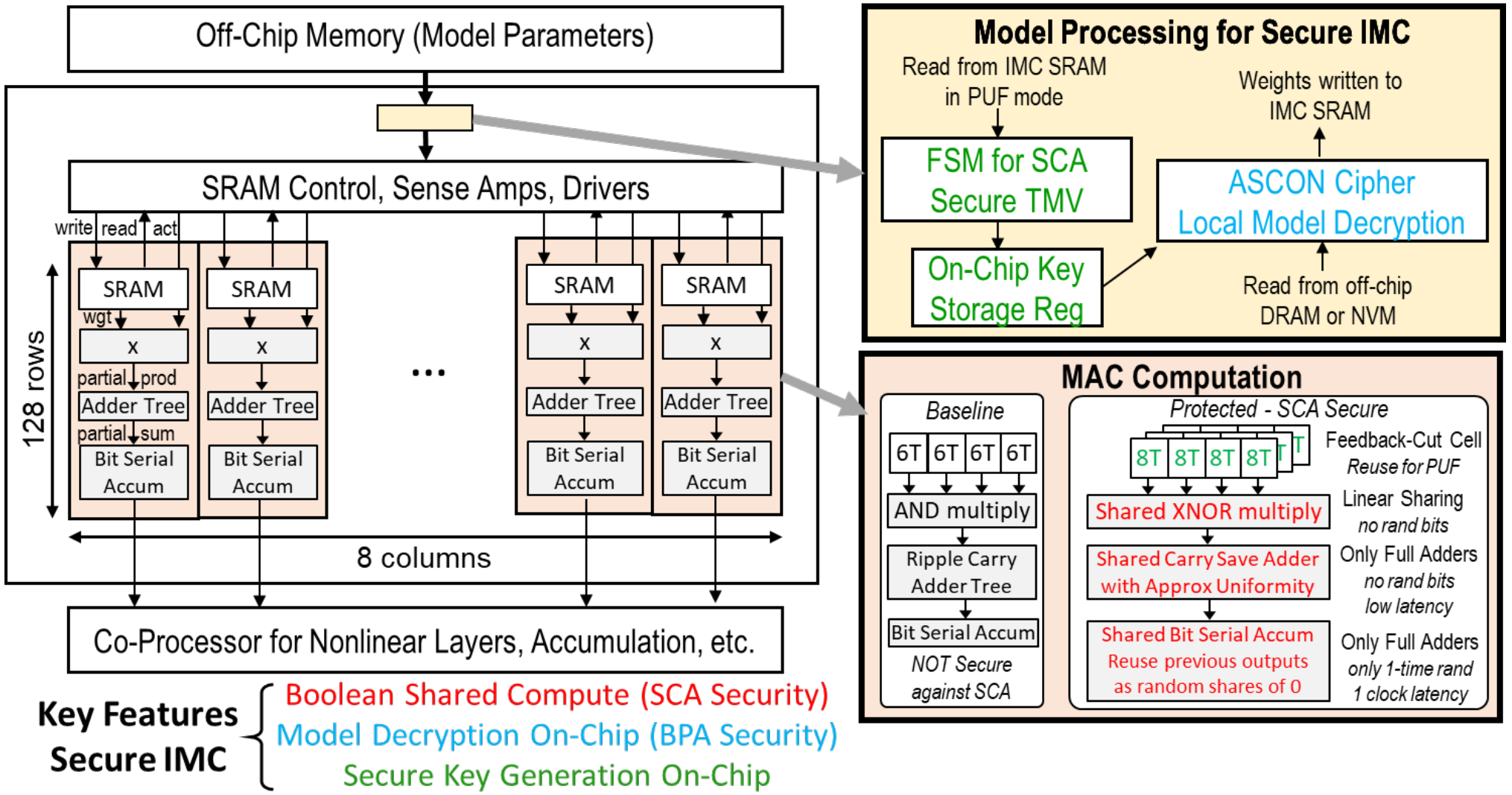
Need for Security



What Security is needed for Neural Networks?

- Input Activations: Contains private information (face images, voice recordings, biometrics, etc.)
- Model Weights: Trained using private dataset, strategic IP for companies, can lead to adversarial attacks

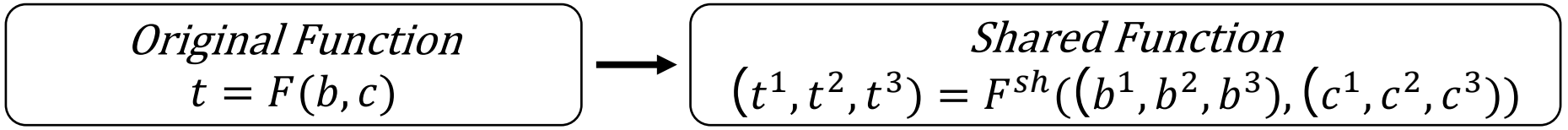
Proposed System



Boolean Sharing Compute

Split each data bit and computation into separately computed ***shares***

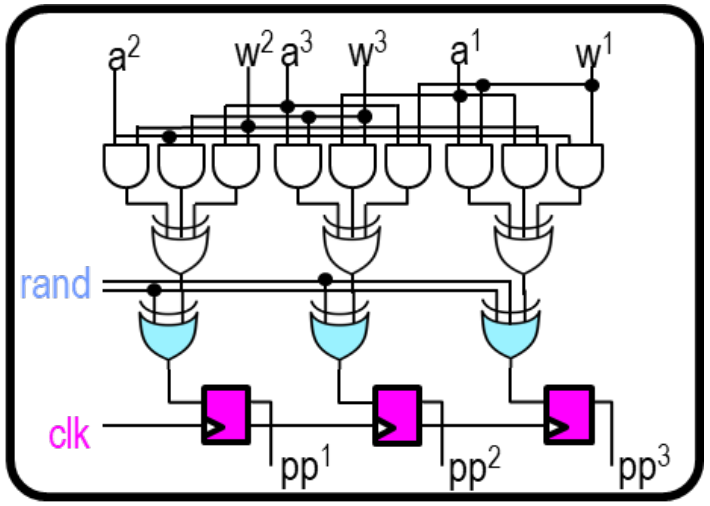
Power consumption decorrelated from data



- Properties (for Practical Levels of Security)
  - Correctness:
    - If $b = b^1 \oplus b^2 \oplus b^3, c = c^1 \oplus c^2 \oplus c^3,$ Then $t = t^1 \oplus t^2 \oplus t^3$
  - Non-Completeness:
    - If $F^{sh} = \{F^1, F^2, F^3\}$, Then each of F^j does not include all shares of each input
  - Approximate Uniformity:
    - For each sub-circuit, each shared output has the same distribution bias as the unshared output
    - Outputs that are not jointly uniform are not combined directly

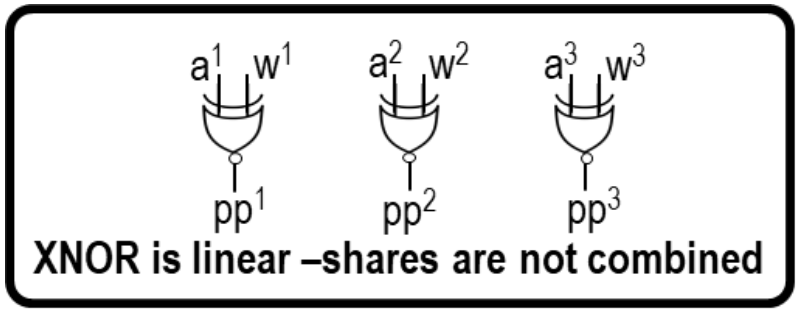
Multiplier Optimization

Conventional: Shared AND gate



- ✗ Random bit refreshing to maintain uniformity
- ✗ Registers to maintain non-completeness
- ✗ 1 multiply = 48 gate-equivalents
- ✓ Standard bit-serial multiply for digital IMC

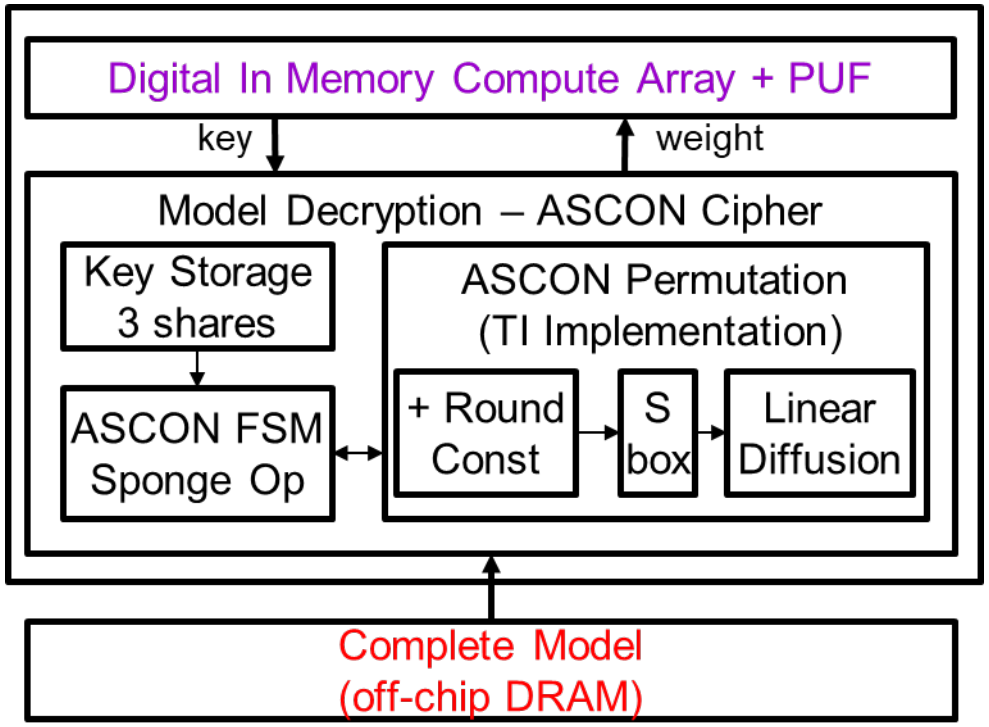
Proposed: Shared XNOR gate



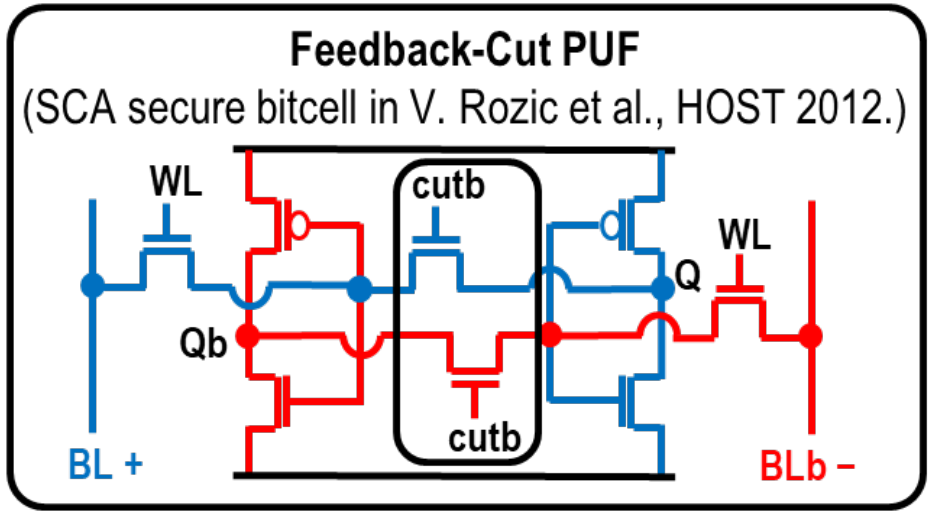
- ✓ Shares maintain uniformity without random bits
- ✓ Can cascade to next gate without registers
- ✓ 1 multiply = 6 gate-equivalents
- – Need data format conversion at macro interface
  Negligible effect on NN accuracy

Addition, Accumulation use similar methods to reduce latency/eliminate random bits

Model Decryption for BPA Security



Secure Key Generated On-Chip



① ***Secure Write Reset***

Write fixed value to remove data dependence

② ***PUF Evaluation***

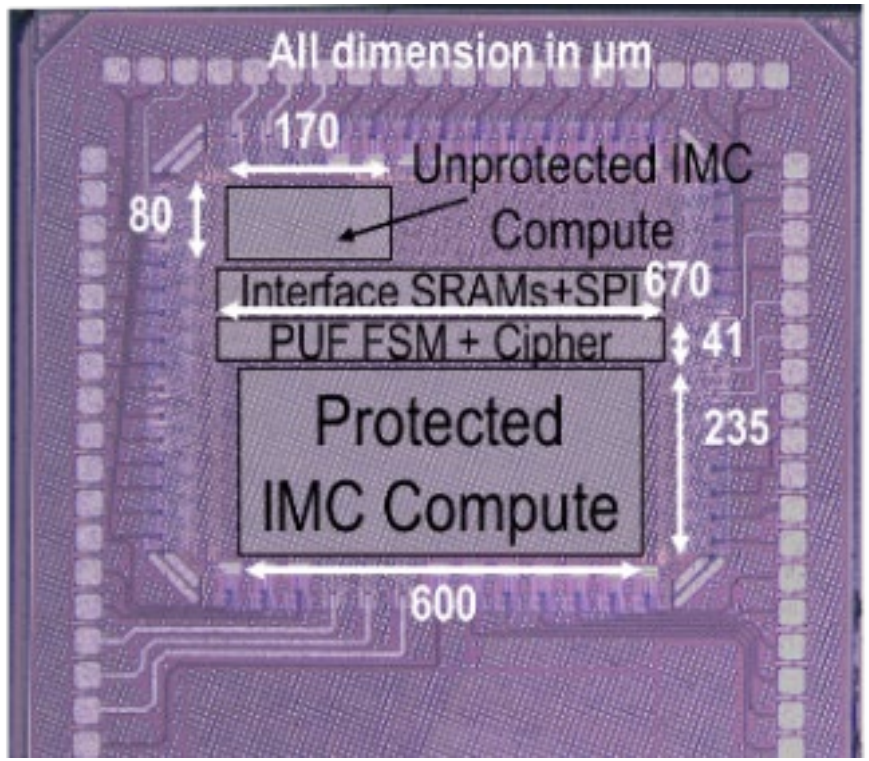
Cut and reconnect feedback transistor

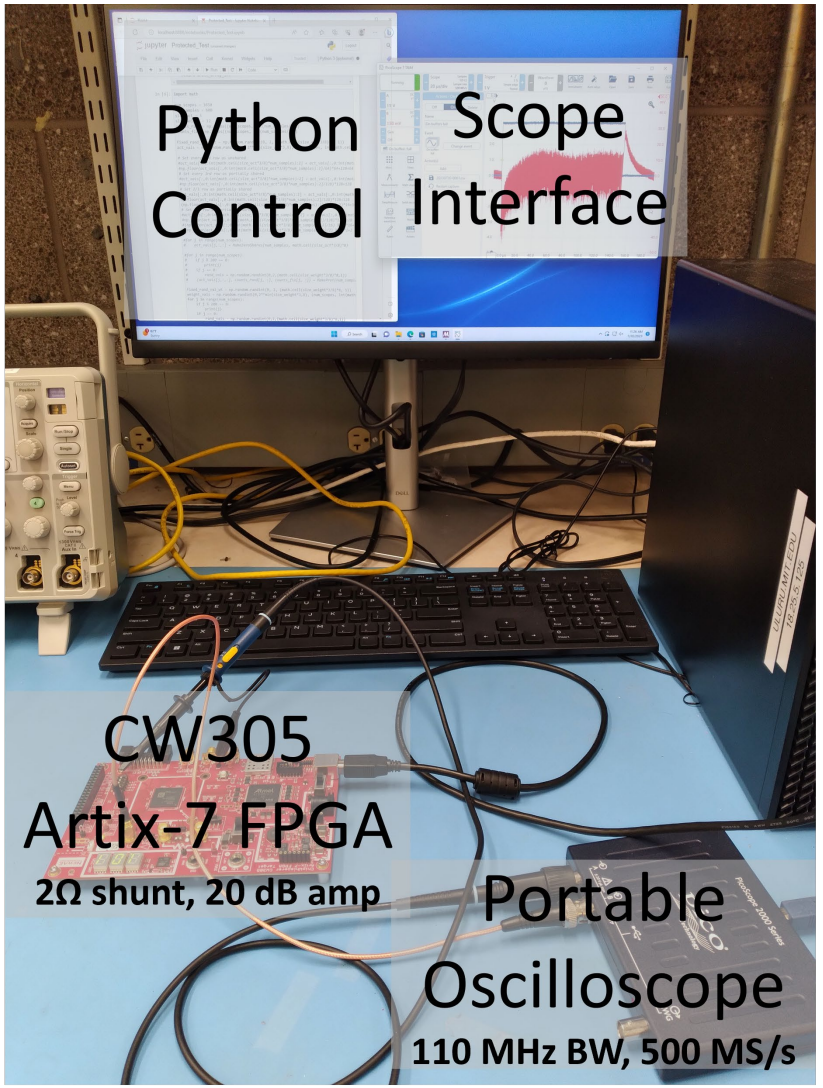
Settles to 0 or 1 since + or – side stronger due to local mismatch

③ ***Standard Read***

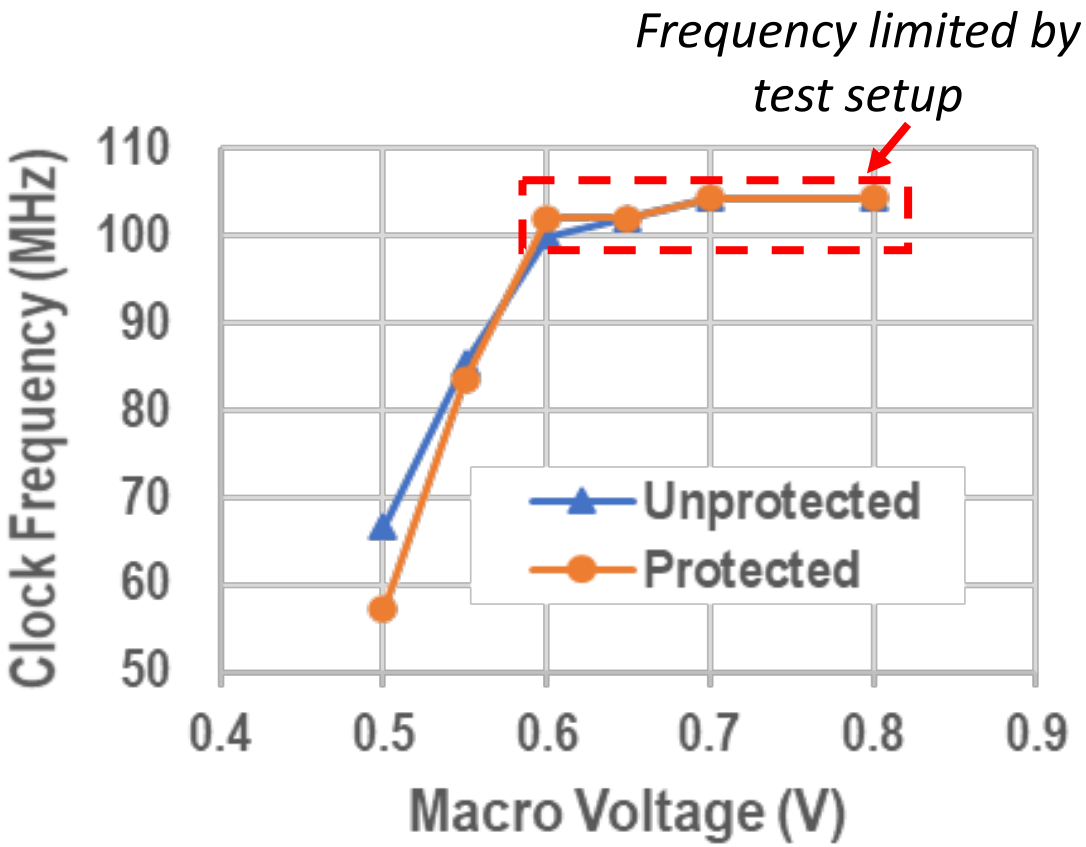
Use differential temporal majority voting for security

Fabricated IC + Evaluation Setup





IMC Performance



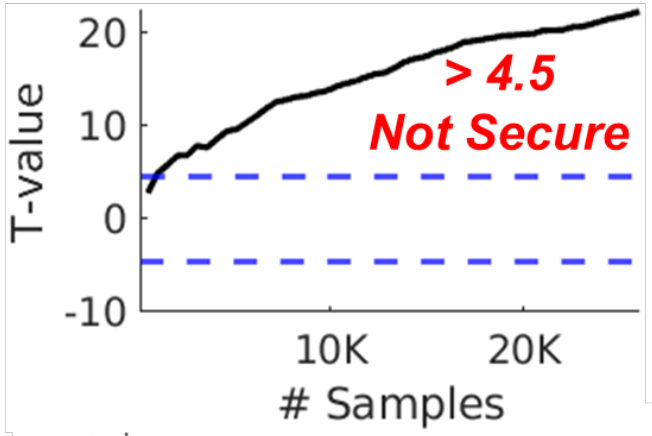
Throughput (GOPS) 0.55 V, 80 MHz	Unprotected	41.0 (4b weight, 1b act) 9.10 (4b weight, 8b act)
	Protected	81.9 (4b weight, 1b act) 10.2 (4b weight, 8b act)
Energy Efficiency (TOPS/W) 0.55 V, 80 MHz	Unprotected	90.2 (4b weight, 1b act) 14.4 (4b weight, 8b act)
	Protected	6.94 (4b weight, 1b act) 0.89 (4b weight, 8b act)
Area Efficiency (TOPS/mm2) 0.55 V, 80 MHz	Unprotected	3.01 (4b weight, 1b act) 0.67 (4b weight, 8b act)
	Protected	0.49 (4b weight, 1b act) 0.061 (4b weight, 8b act)

Security Evaluation

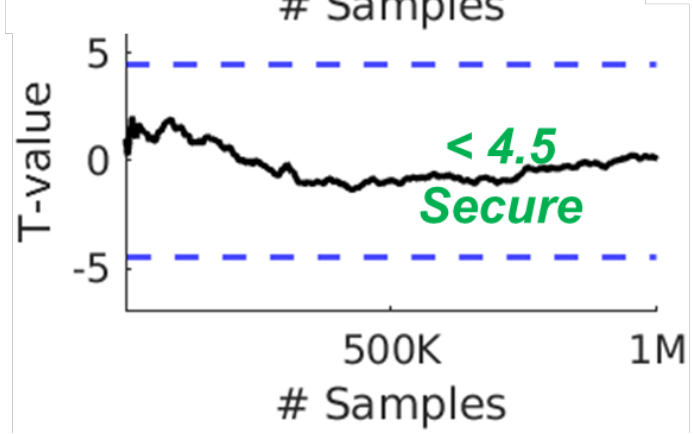
Test Vector Leakage Assessment

$$t = \frac{\mu_{fixed} - \mu_{random}}{\sigma_{fixed} - \sigma_{random} \Big/ \sqrt{\#\ samples}}$$

Unprotected

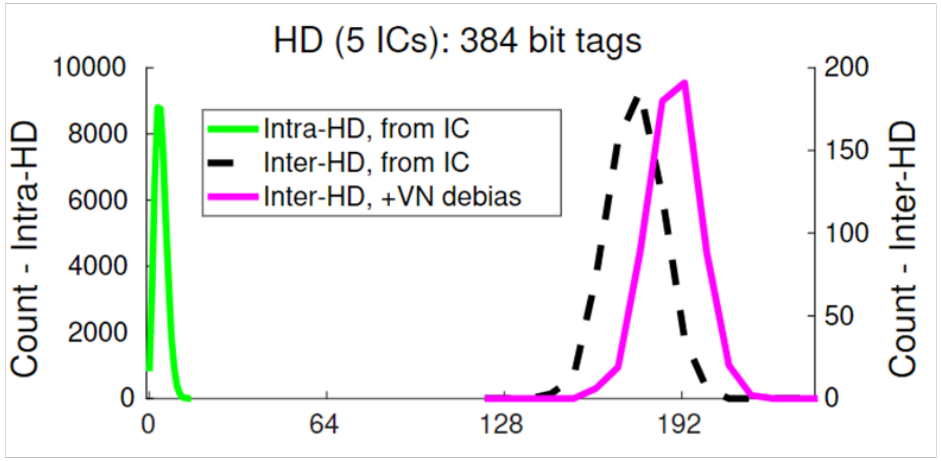


Protected



PUF Security Analysis

Passes all applicable NIST Tests



Temporal Majority Voting

SCA Security (CNN attack)

Key RMSE from FSM	Not Shared 250K train	Shared 1M train	**Shared & Diff. 1M train**
Unshared HW	1.38	4.90	**7.20**
Shared HW	N/A	2.35	**32.13**

Conclusion + Future Work

- Generalized IMC solution for ML with *Privacy & Integrity*
  - Side Channel and Bus Probing Attack Security for In Memory Compute
  - No random bits from PRNGs required
  - No limitations on neural network accuracy
- Future Improvements
  - More exploration of tradeoffs between security and area/energy overheads
  - Usage of approximate compute for further exploitation of natively secure compute gates
- Acknowledgements
  - MIT-IBM Watson AI lab for funding, helpful discussion, and support
  - NSF GRFP (Grant No. 1745302) and MathWorks Engineering Fellowship