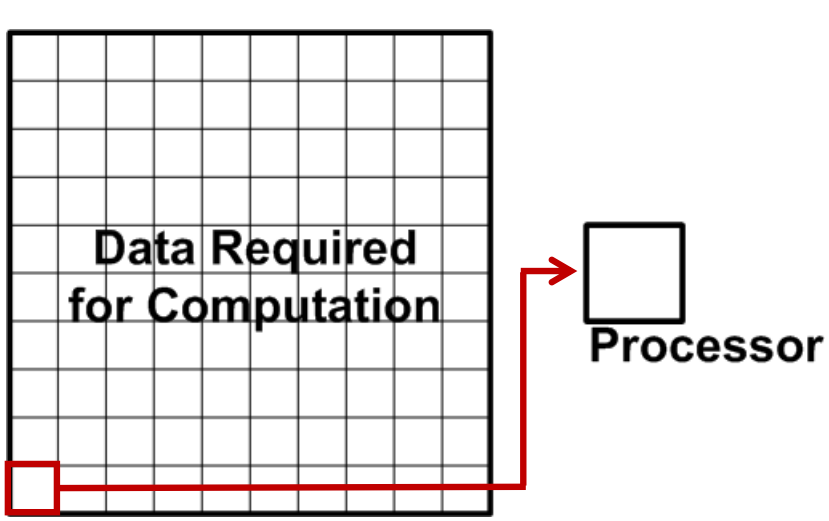
What can in-memory computing deliver, and what are the barriers?

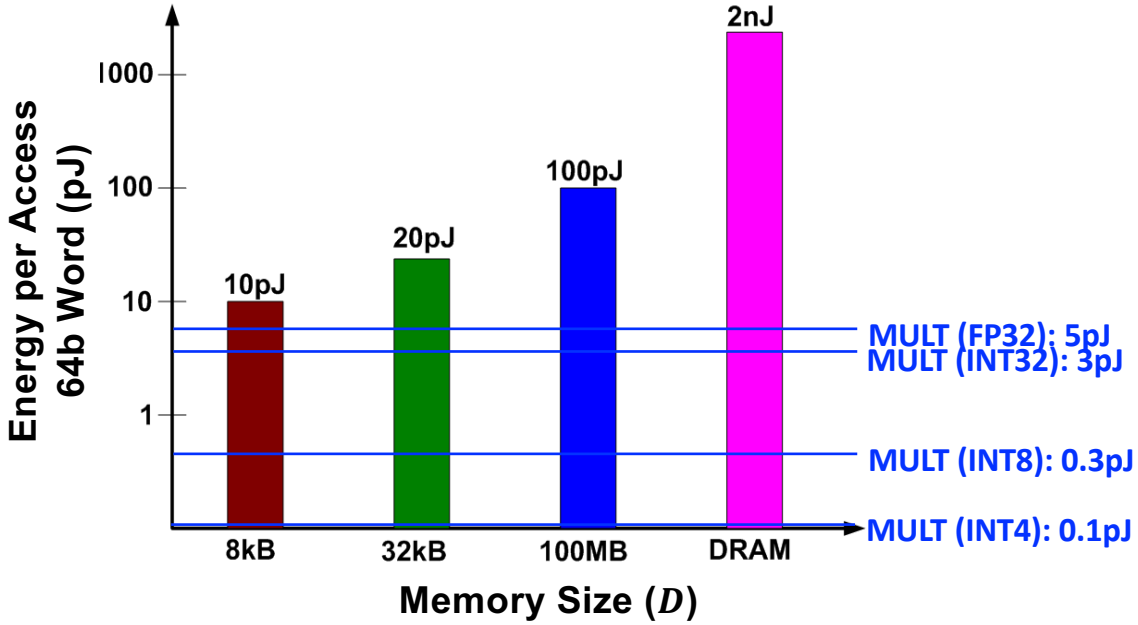
Naveen Verma (nverma@princeton.edu),
L.-Y. Chen, H. Jia, M. Ozatay, Y. Tang, H. Valavi, B. Zhang, J. Zhang

June 16, 2019

The memory wall

- **Separating memory from compute fundamentally raises a communication cost**

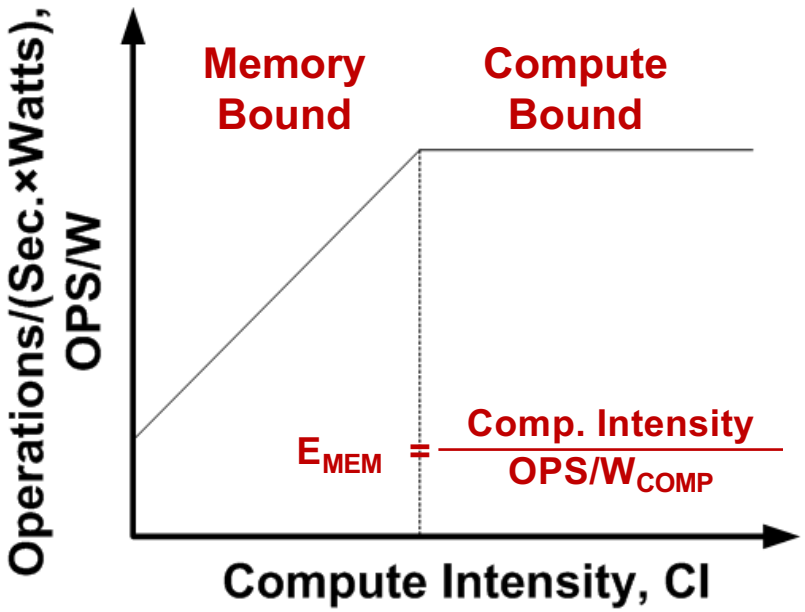




More data → bigger array → larger comm. distance → more comm. energy

So, we should amortize data movement

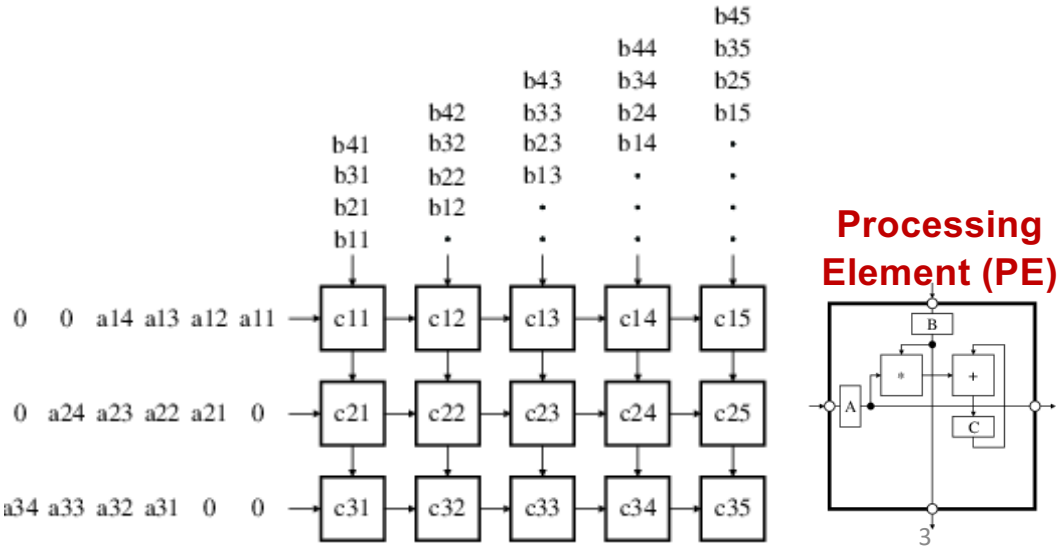
- **Reuse accessed data for compute operations**



- **Specialized (memory-compute integrated) architectures**

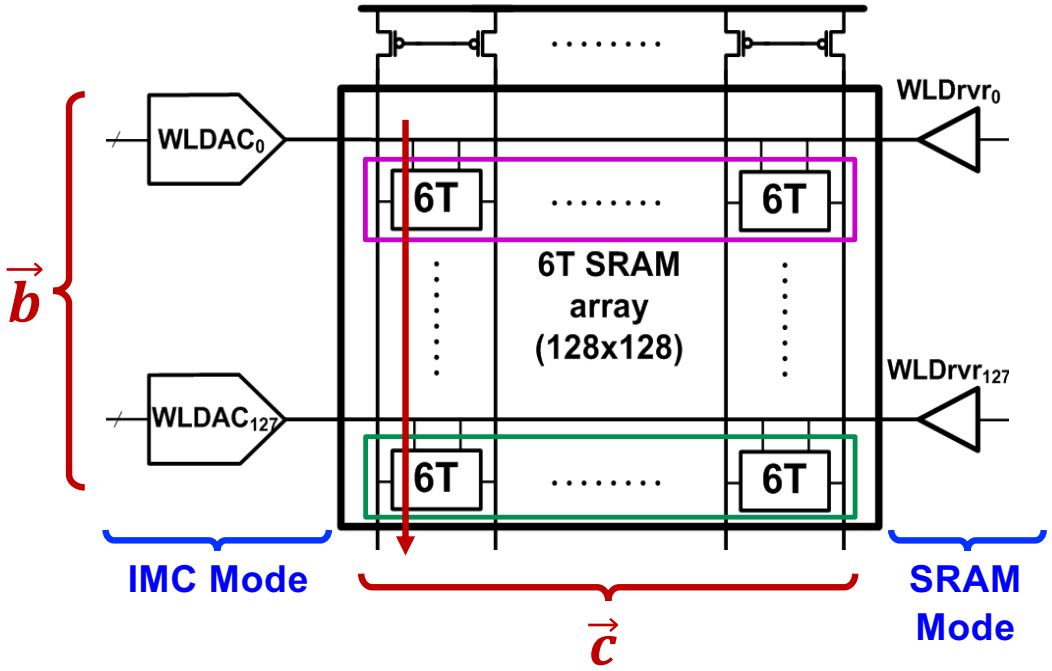
$$\vec{c} = A \times \vec{b}$$

$$\begin{bmatrix} c_1 \\ \vdots \\ c_M \end{bmatrix} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,N} \\ \vdots & \ddots & \vdots \\ a_{M,1} & \cdots & a_{M,N} \end{bmatrix} \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix}$$



In-memory computing (IMC)

$$\vec{c} = A\vec{b} \quad \Rightarrow \quad \begin{bmatrix} c_1 \\ \vdots \\ c_M \end{bmatrix} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,N} \\ \vdots & \ddots & \vdots \\ a_{M,1} & \cdots & a_{M,N} \end{bmatrix} \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix}$$

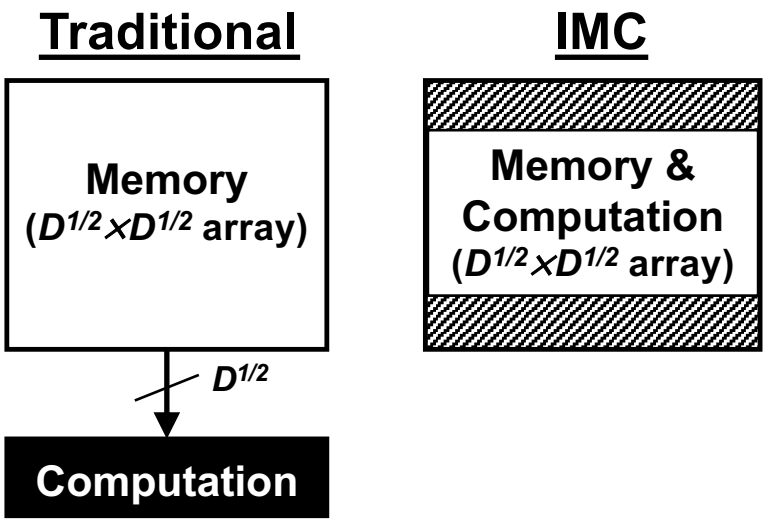


- **In SRAM mode, matrix *A* stored in bit cells <u>row-by-row</u>**
- **In IMC mode, many WLs driven <u>simultaneously</u>**
  → *amortize comm. cost inside array*
- **Can apply to diff. mem. Technologies**
  → *enhanced scalability*
  → *embedded non-volatility*

[J. Zhang, *VLSI*'16][J. Zhang, *JSSC*'17]

The basic tradeoffs

CONSIDER: Accessing D bits of data associated with computation, from array with $\sqrt{D}$ columns × $\sqrt{D}$ rows.



Metric	Traditional	In-memory
Bandwidth	$1/D^{1/2}$	1
Latency	D	1
Energy	$D^{3/2}$	$\sim D$
SNR	1	$\sim 1/D^{1/2}$

- IMC benefits energy/delay at cost of SNR
- SNR-focused systems design is critical (circuits, architectures, algorithms)

Second-order stuff…

Single-row Read:

$$E_{RD} = E_{PRE} + E_{WL} + N \times E_{BL,RD} + \log K \times E_{MUX} + E_{SA}/K + E_{BLOCK}$$

SRAM:

$$E_{A-RD} = \sqrt{D} \times E_{RD} = \sqrt{D} \times E_{PRE} + \sqrt{D} \times E_{WL} + \sqrt{D} \times N \times E_{BL,RD} + \sqrt{D} \times \log K \times E_{MUX} + \sqrt{D} \times E_{SA}/K + \sqrt{D} \times E_{BLOCK}$$

Access 1 rows

Low-swing discharge

IMC:

$$E_{F-RD} = E_{PRE} + M \times E_{WL,F-RD} + N \times E_{BL,F-RD} + E_{ACQ}/K + E_{BLOCK}$$

Access ALL rows

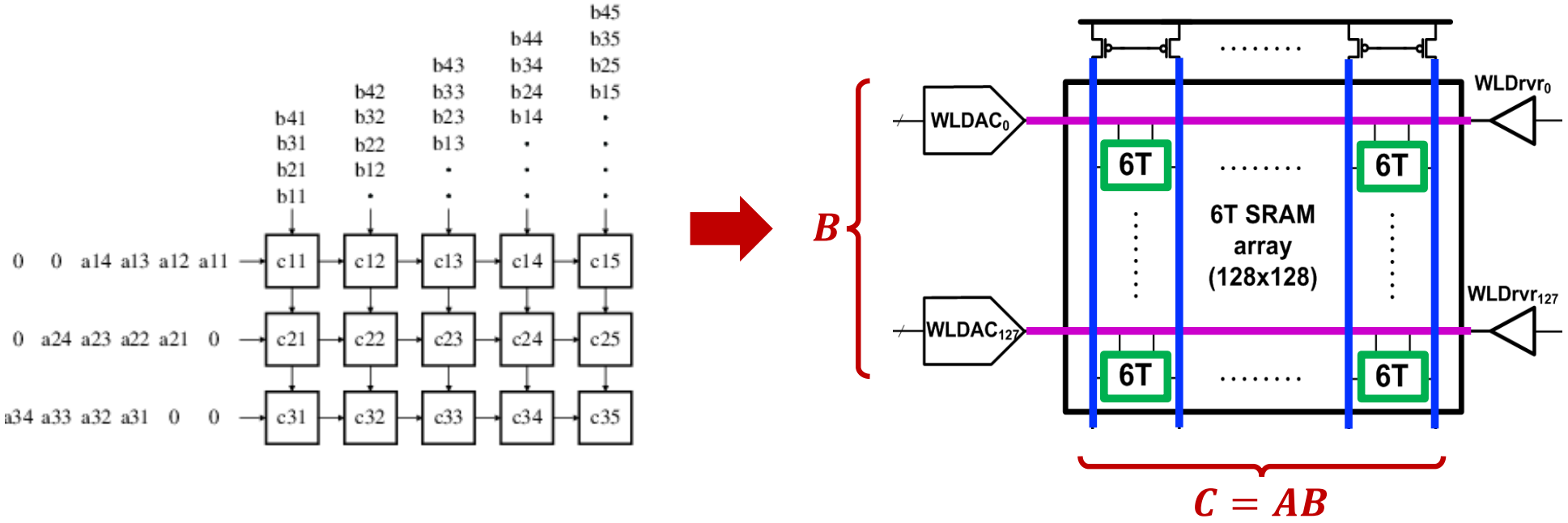
Full-swing discharge

IMC Gains:

$$\frac{E_{A-RD}}{E_{F-RD}} \approx \frac{\sqrt{D} \times E_{WL} + \sqrt{D} \times N \times E_{BL,RD}}{M \times E_{WL,F-RD} + N \times E_{BL,F-RD}} = \frac{\sqrt{D} \times E_{WL} + D \times E_{BL,RD}}{\sqrt{D} \times E_{WL,F-RD} + \sqrt{D} \times E_{BL,F-RD}} \quad (N = M = \sqrt{D})$$

- IMC reduces $E_{BL,RD}/E_{BL,F\text{-}RD}$ operations, but not E_{WL}
- Usually $E_{BL,F\text{-}RD} > E_{BL,RD}$, sometimes $E_{WL,F\text{-}RD} < E_{WL,RD}$

IMC as a spatial architecture



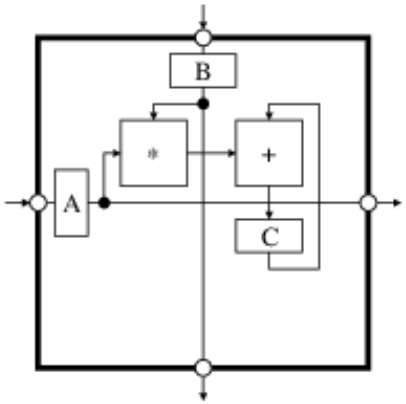
Data Movement:

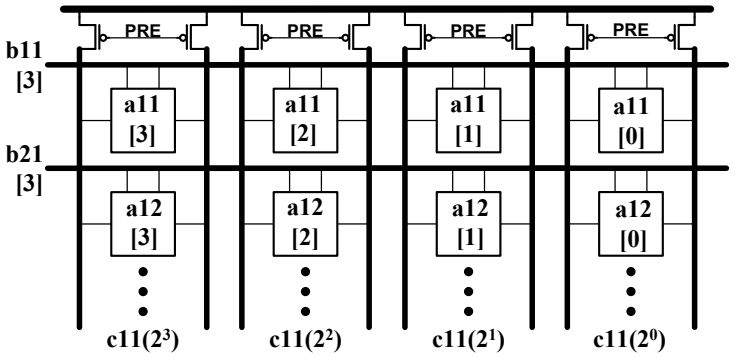
1. $b_{n,k}{}'s$ broadcast min. distance due to high-density bit cells
2. (Many) $a_{m,n}{}'s$ stationary in high-density bit cells
3. High-dynamic-range analog $c_{m,k}{}'s$ computed in distributed manner

IMC as a spatial architecture

Assume:

- 1k dimensionality
- 4-b multiplies
- 45nm CMOS

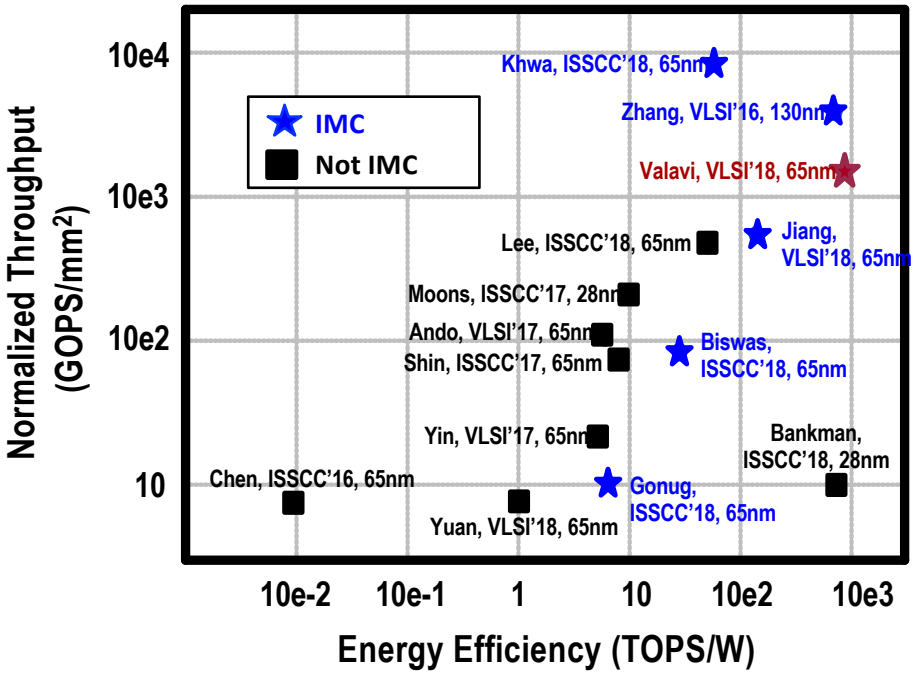


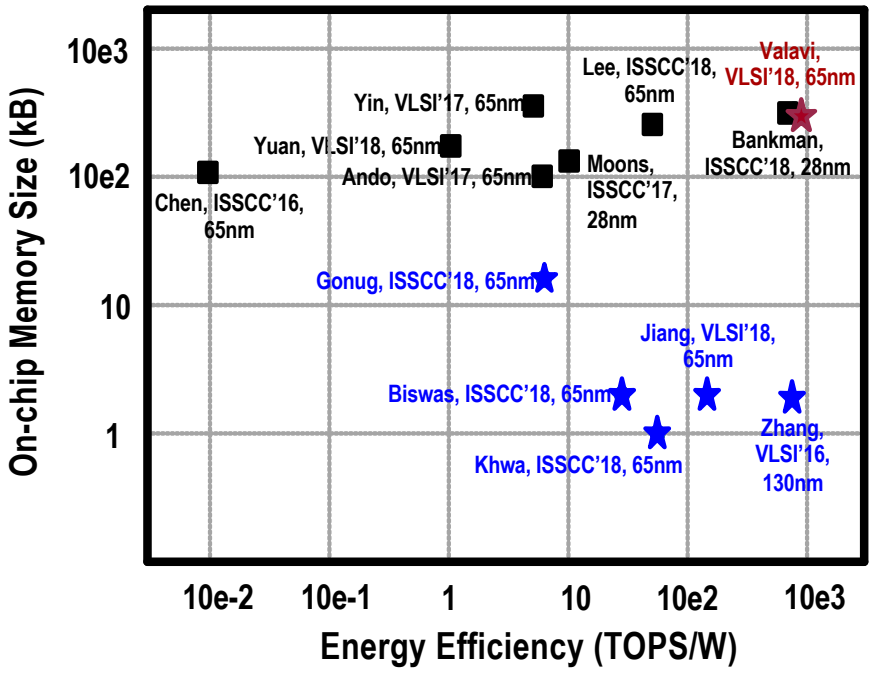


Operation	Digital-PE Energy (fJ)	Bit-cell Energy (fJ)
Storage	250	50
Multiplication	100	
Accumulation	200	
Communication	40	5
Total	**590**	**55**

Where does IMC stand today?

- **Potential for 10× higher efficiency & throughput**
- **Limited scale, robustness, configurability**

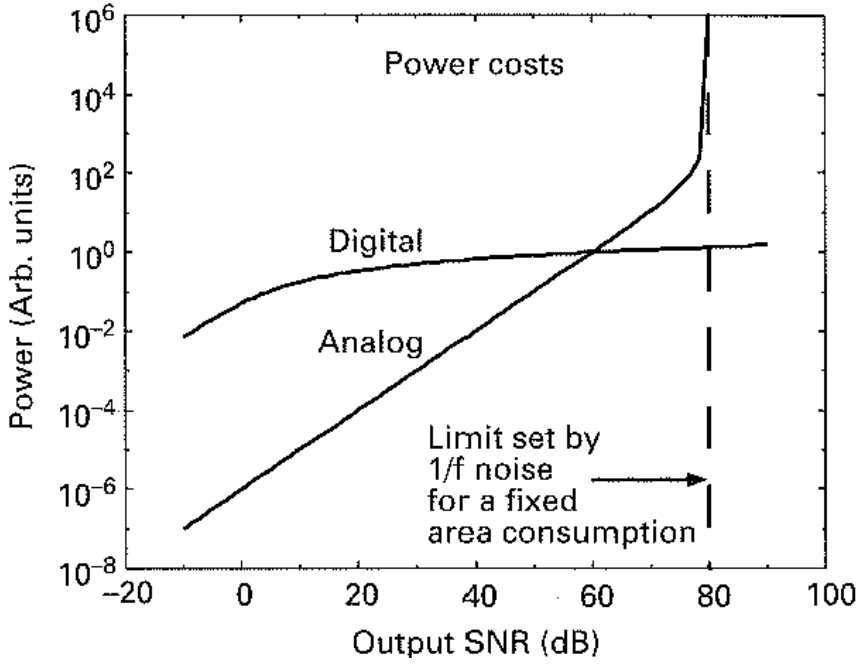




IMC challenge (1): analog computation

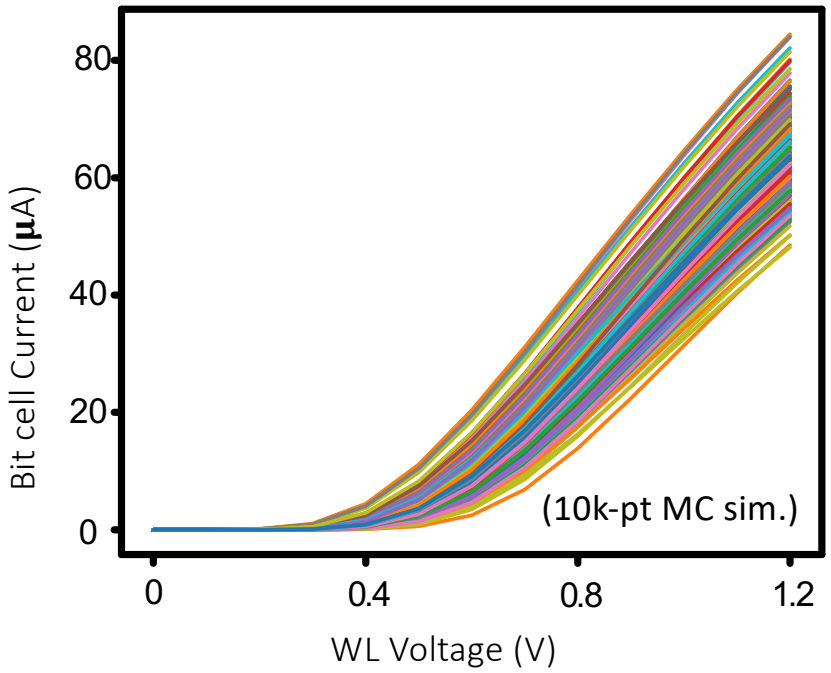
- **Need analog to 'fit' compute in bit cells (SNR limited by analog non-idealities)**
  ⟶ Must be feasible/competitive @ 16/12/7nm

Noise Limited



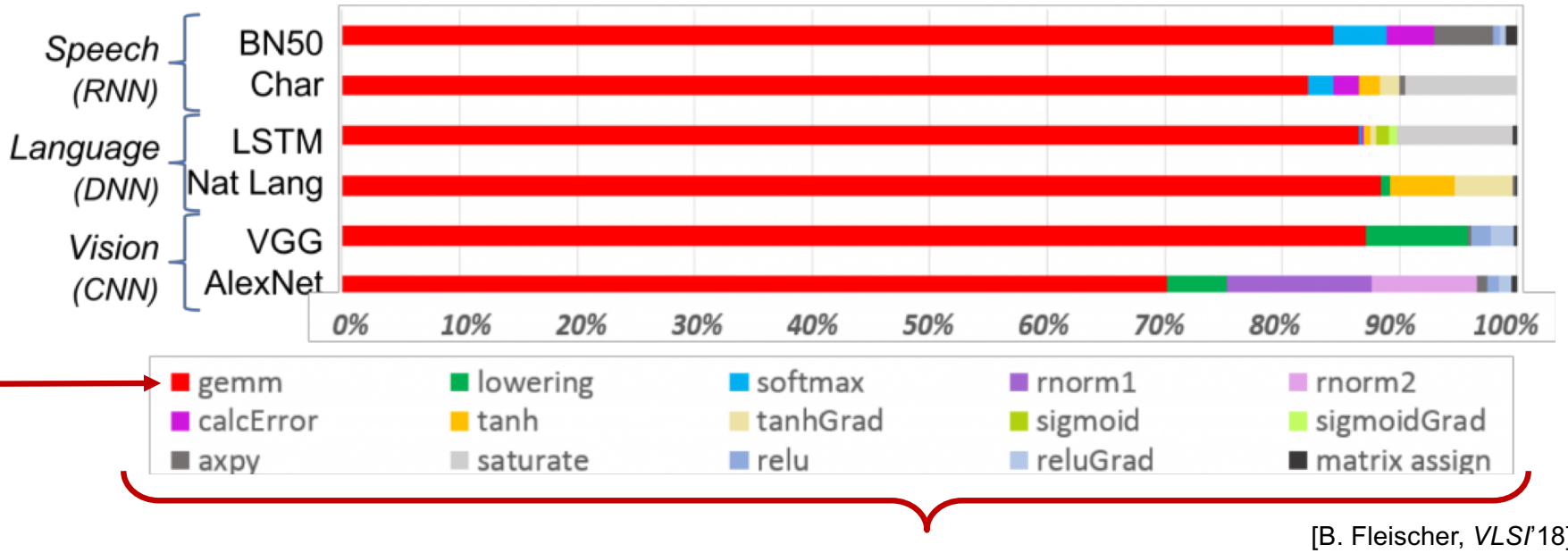
[R. Sarpeshkar, *Ultra Low Power Bioelectronic*]

Linearity/variation Limited



IMC Challenge (2): heterogeneity

- **Matrix-vector multiply is only 70-90% of operations**
  ⟶ IMC must integrate in programmable, heterogenous architectures



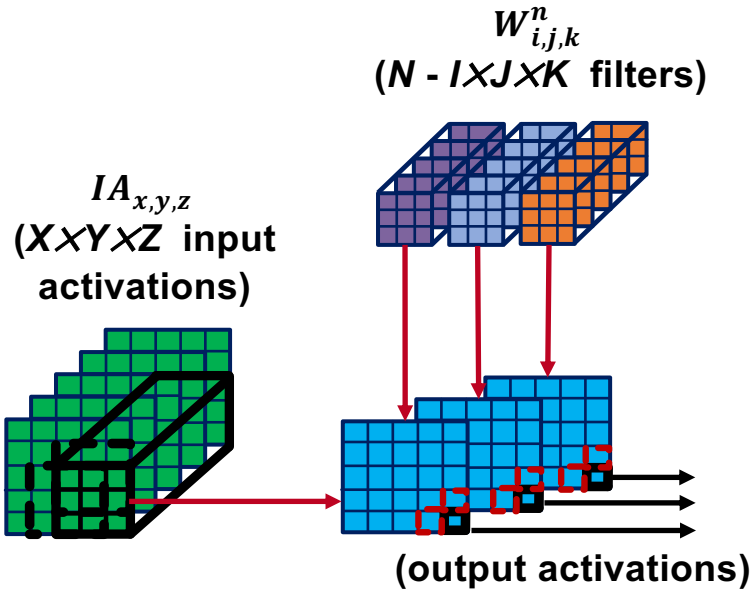
[B. Fleischer, *VLSI*'18]

General Matrix Multiply (~256×2300=590k elements)

Single/few-word operands (traditional, near-mem. acceleration)

IMC Challenge (3): efficient application mappings

- **IMC engines must be 'virtualized'**
  ⟶ IMC amortizes MVM costs, not weight loading. But…
  ⟶ Need new mapping algorithms (physical tradeoffs very diff. than digital engines)

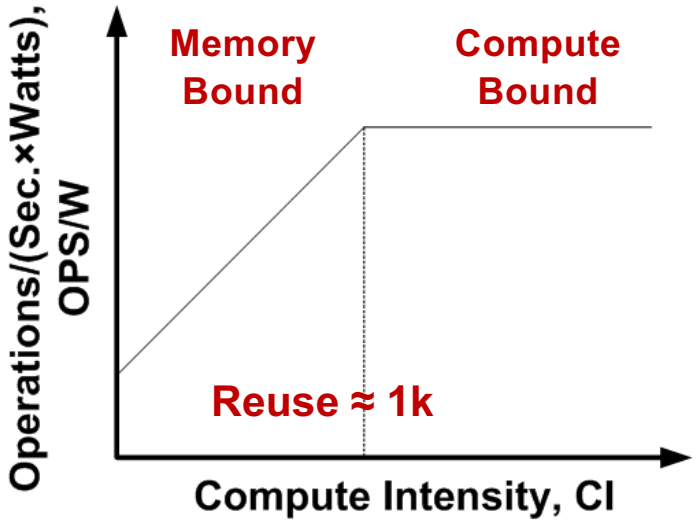


Activation Accessing

- $E_{DRAM \rightarrow IMC}$/4-bit: 40pJ
- Reuse: $N \times I \times J$ (10-20 lyrs)
- $E_{MAC,4\text{-}b}$: 50fJ

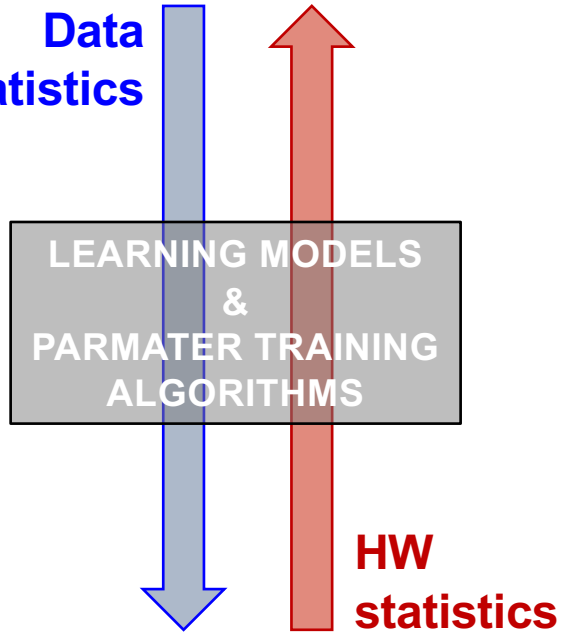
Weight Accessing

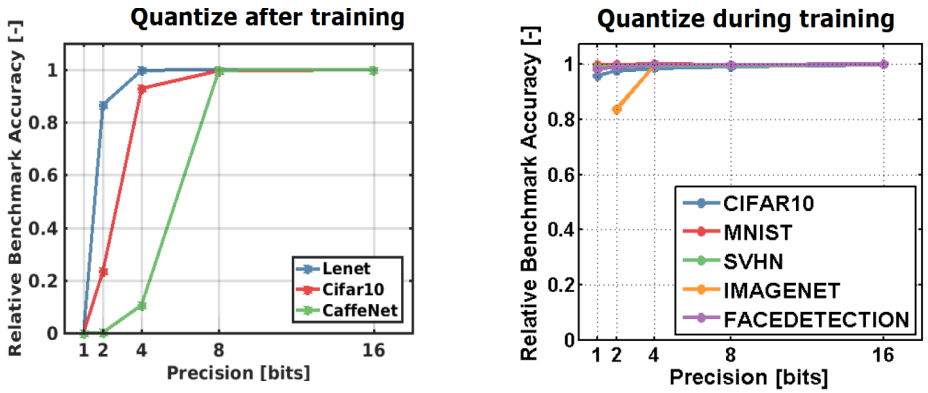
- $E_{DRAM \rightarrow IMC}$/4-bit: 40pJ
- Reuse: $X \times Y$
- $E_{MAC,4\text{-}b}$:50fJ



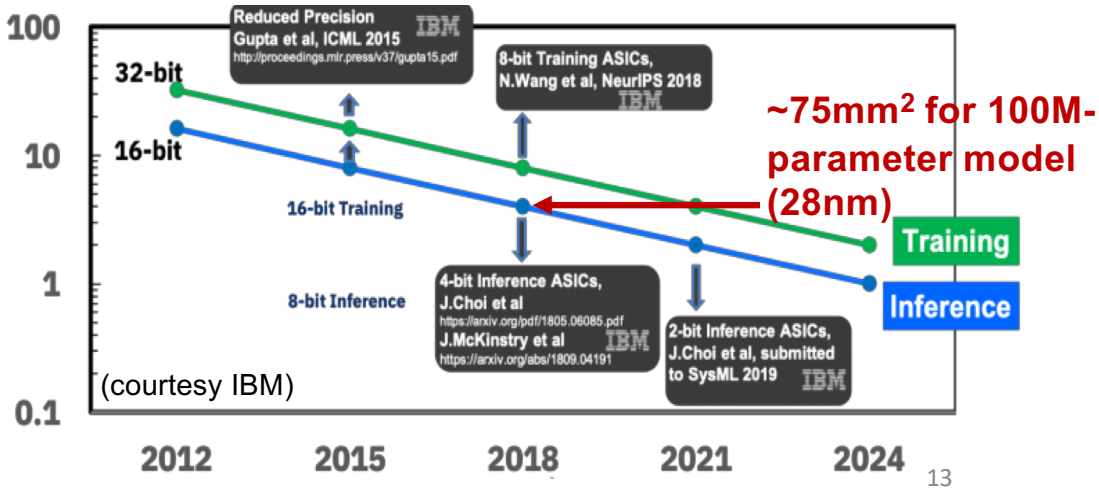
Neural-network trend (1): reducing bit precision

Opportunity for top-down
and bottom-up design



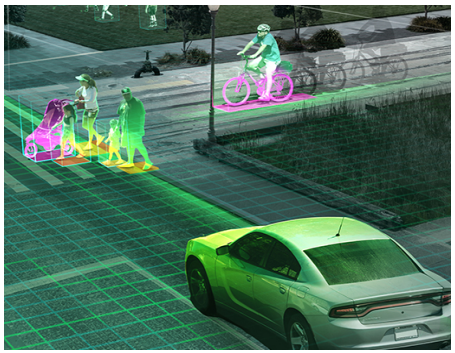


[M. Verhelst, *ISSCC SC on Machine Learning (2018)*]

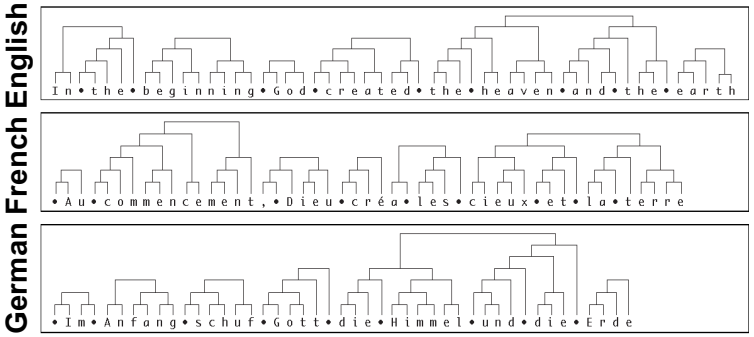


Neural-network trend (2): varied models

Images (SPATIAL structure)

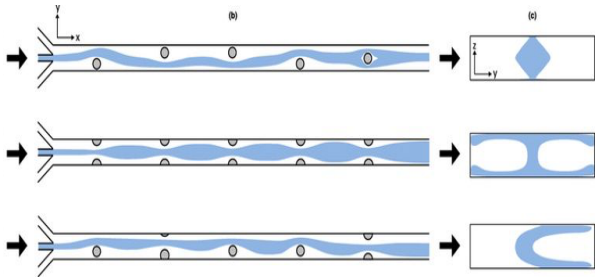


Language (SEQUENTIAL structure)

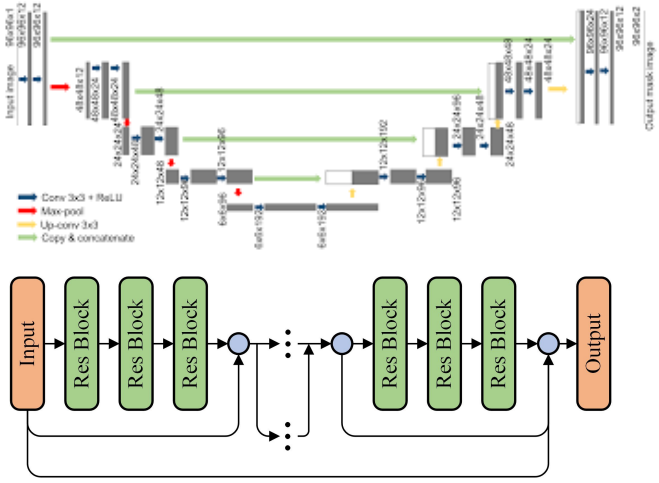


Inverse problems (PHYSICS)

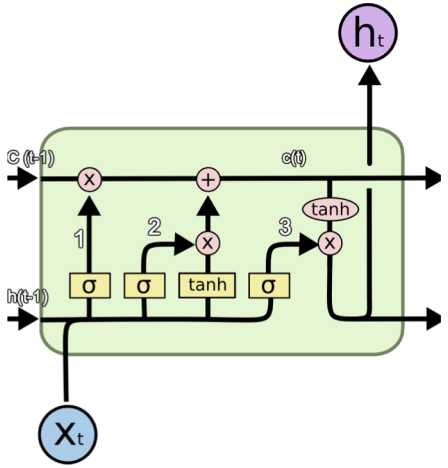
Ex. Fluid flow:



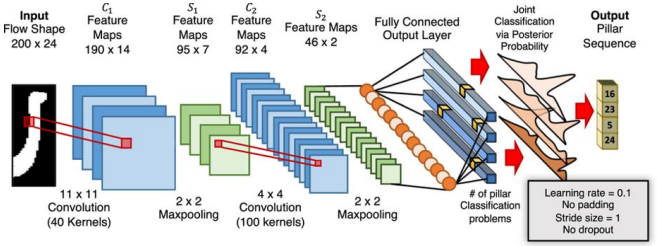
E.g., Convolutional Neural Net. (CNN)



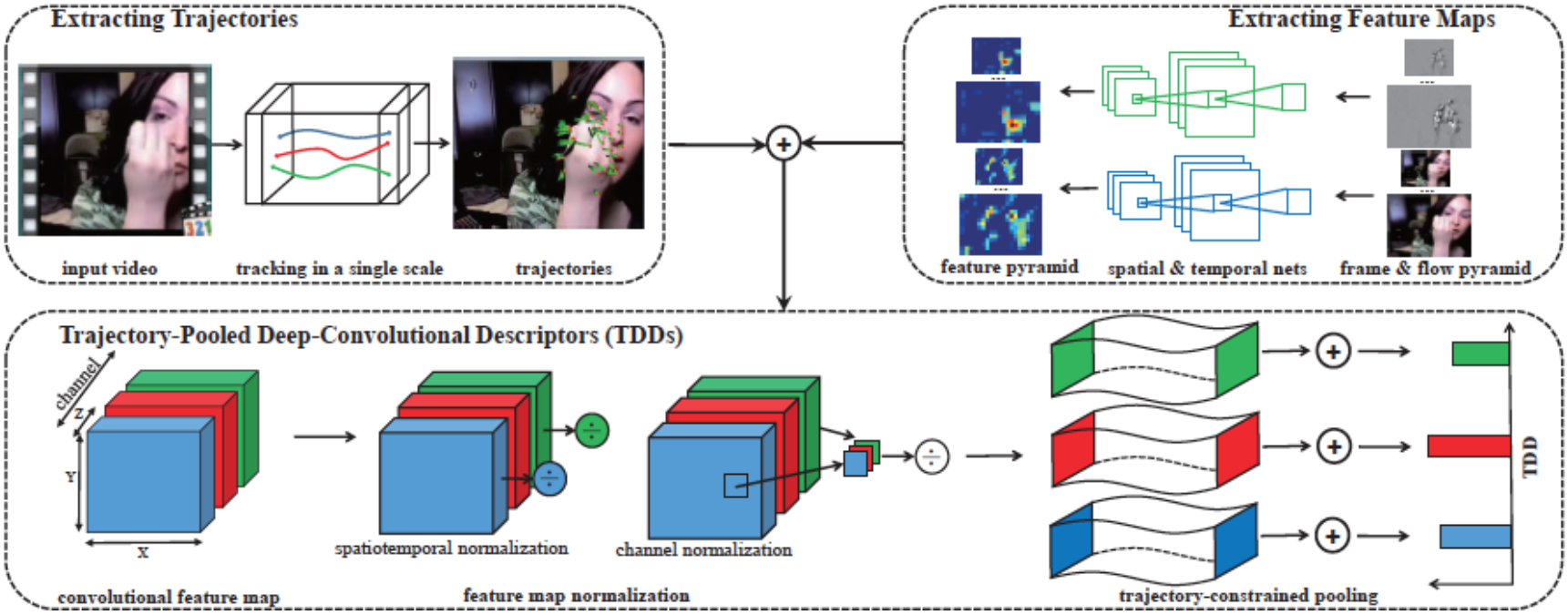
E.g., Long-Short-Term Memory (LSTM)



E.g., Flow sculpting [D. Stoecklein, *Nature'*17]



Neural-network trend (3): diverse use cases & pipelines

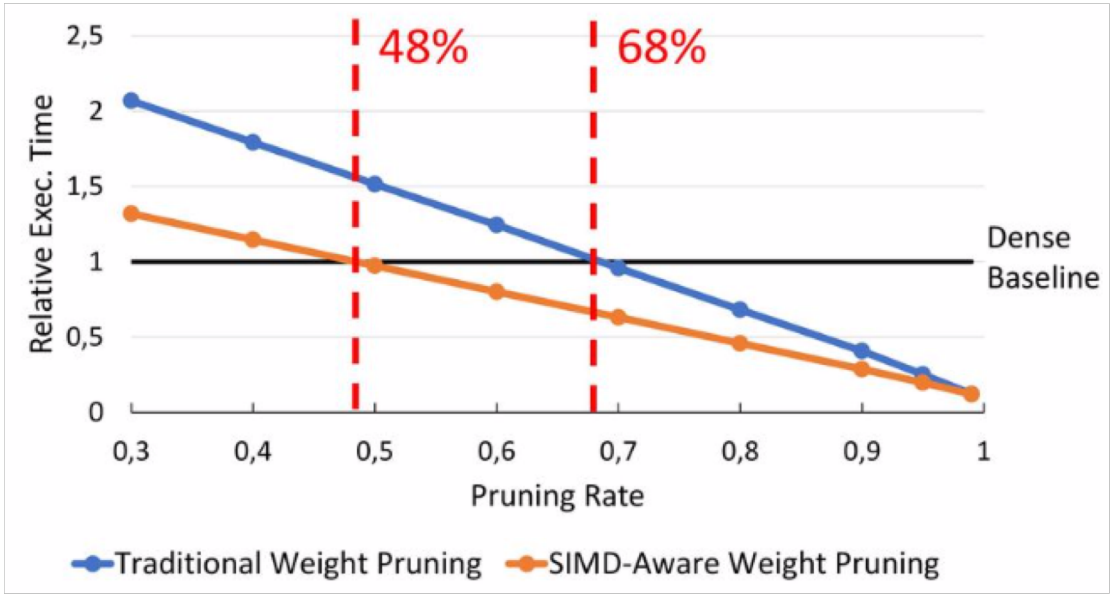


[L. Wang, *CVPR* 2015]

→ **Diverse pipelines, diverse batch sizes, varying latency requirements**

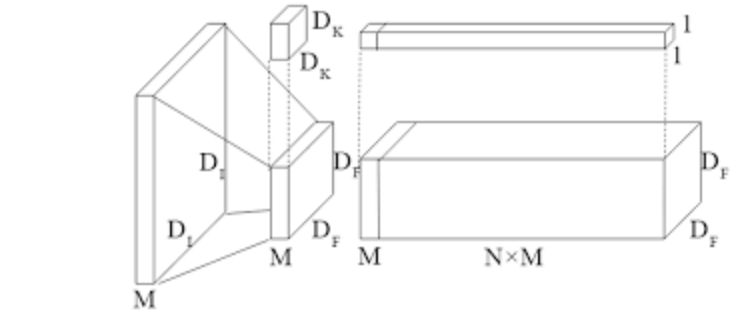
Neural-network trend (4): model compression, etc.

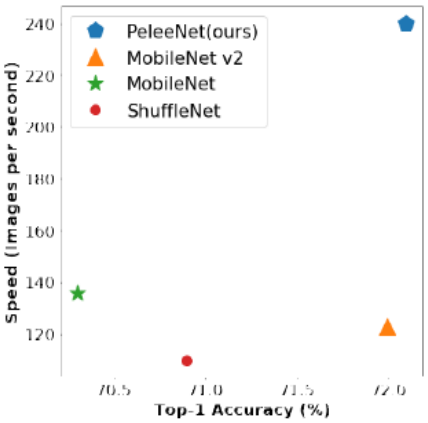
Weight Pruning

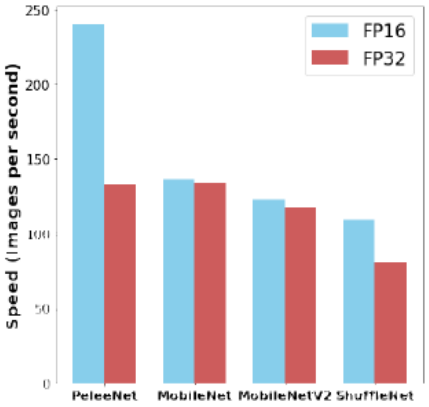


[J. Yu, ISCA'17]

Depth-wise Separable Convolutions

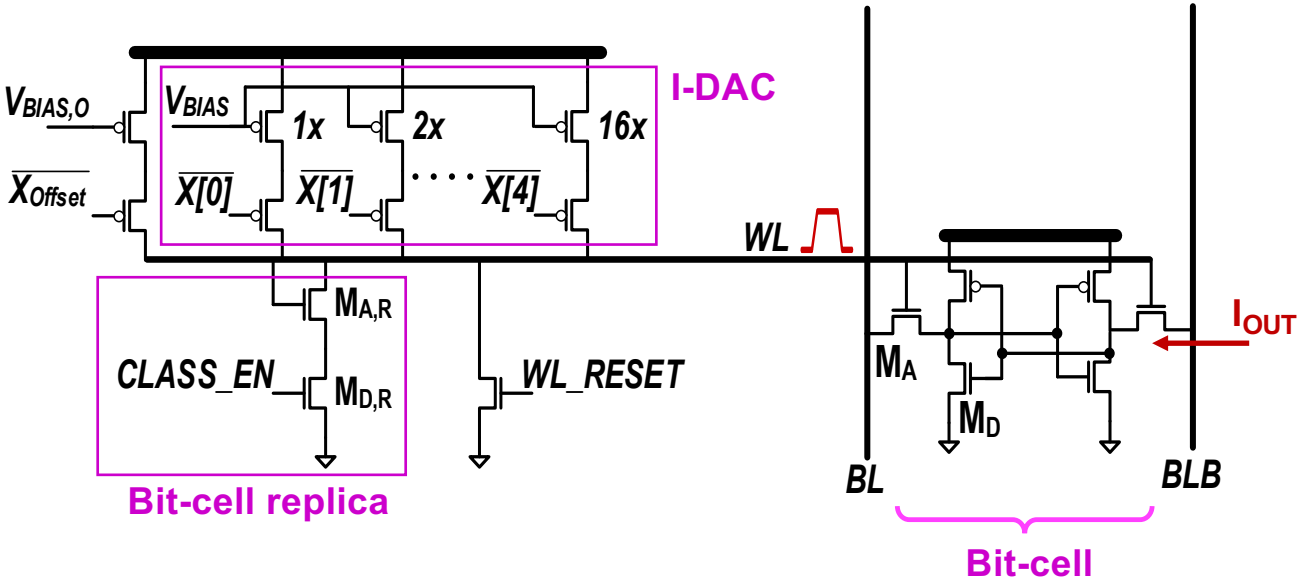


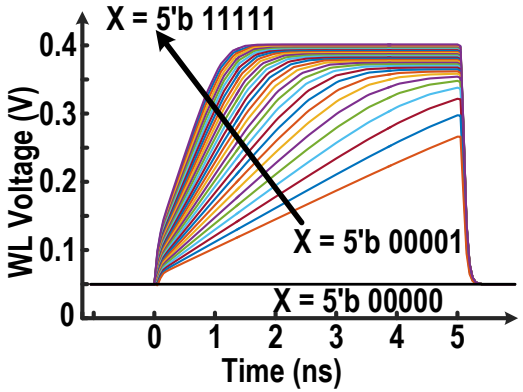


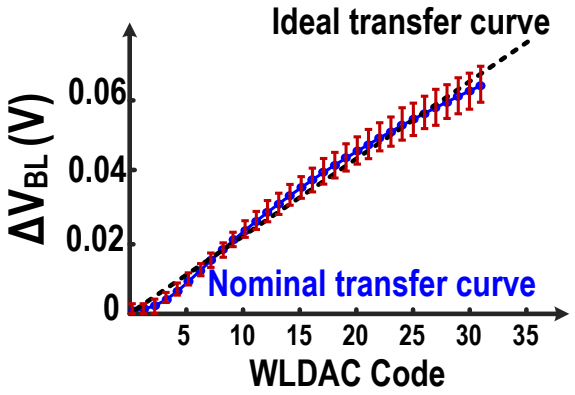


[R. Wang, NIPS'2018]

SNR of analog compute (SRAM)

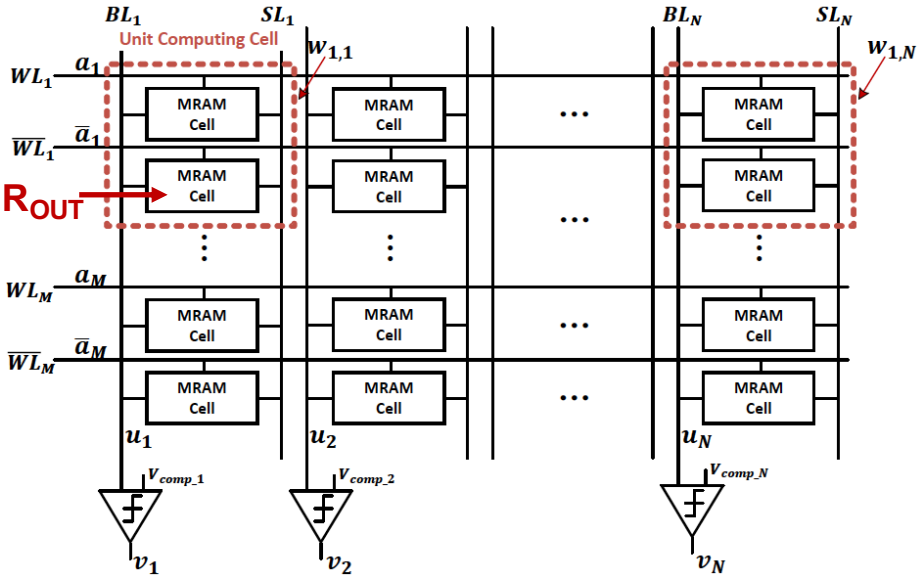


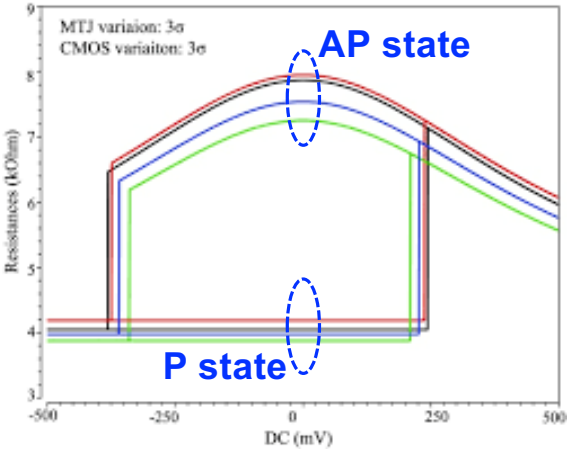




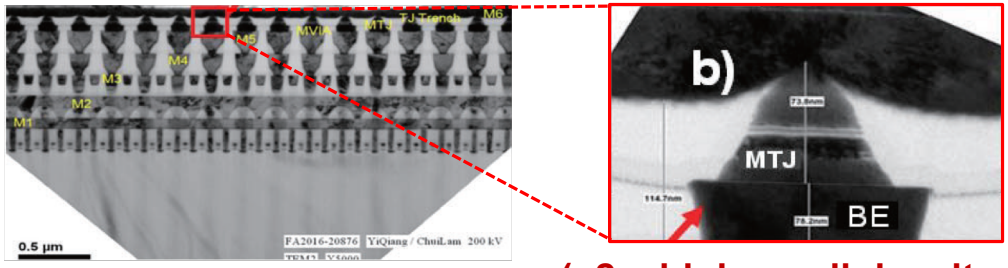
[J. Zhang, *VLSI*'16][J. Zhang, *JSSC*'17]

SNR of analog compute (MRAM)

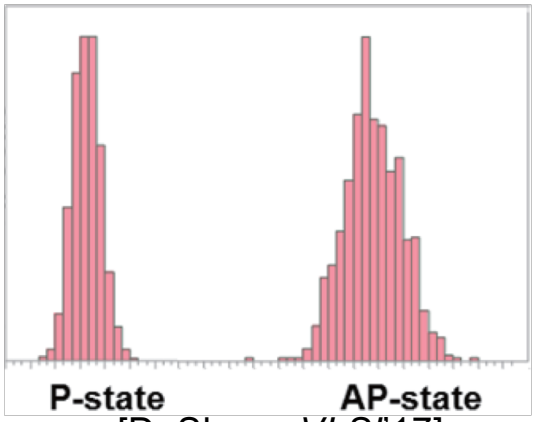




MRAM in 22nm FD-SOI (GlobalFoundries):



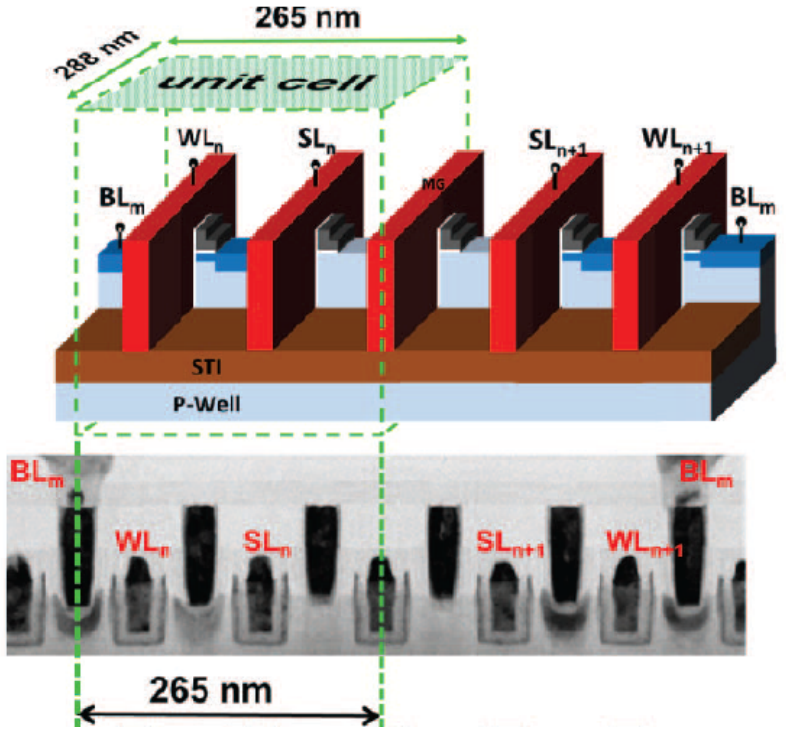
(~2x higher cell density than SRAM)

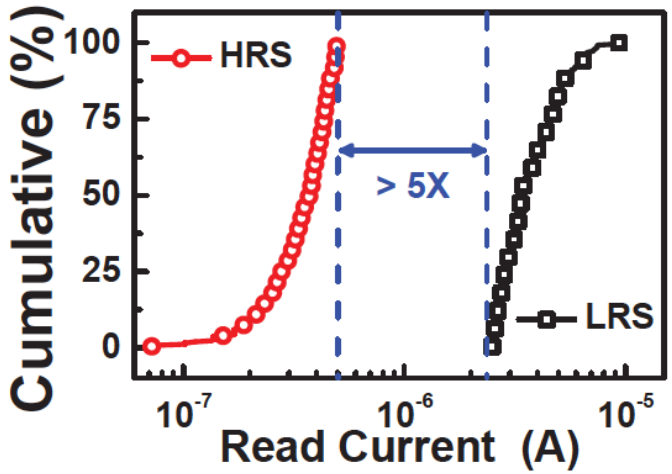


[D. Shum, *VLSI*'17]

SNR of analog compute (ReRAM)

ReRAM in 16nm FinFET (TSMC):



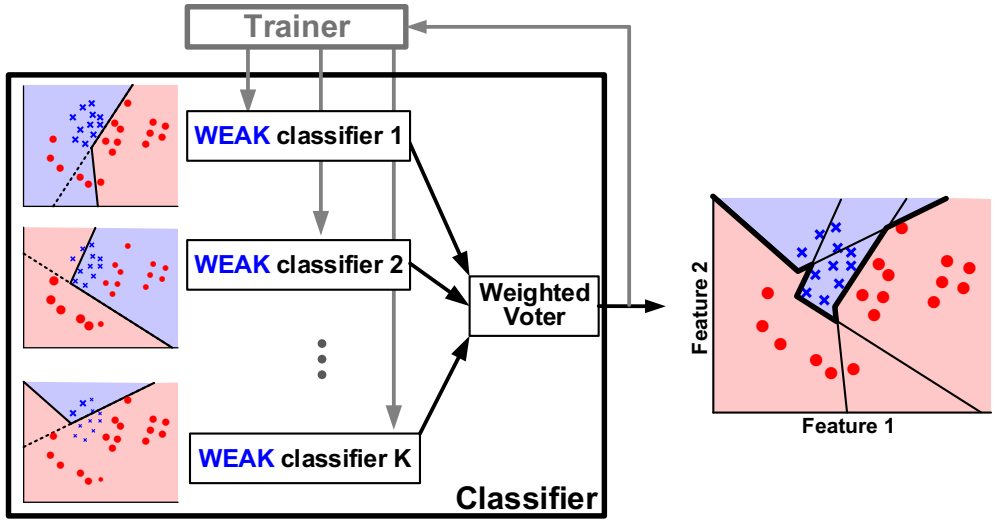


(similar cell density to SRAM)

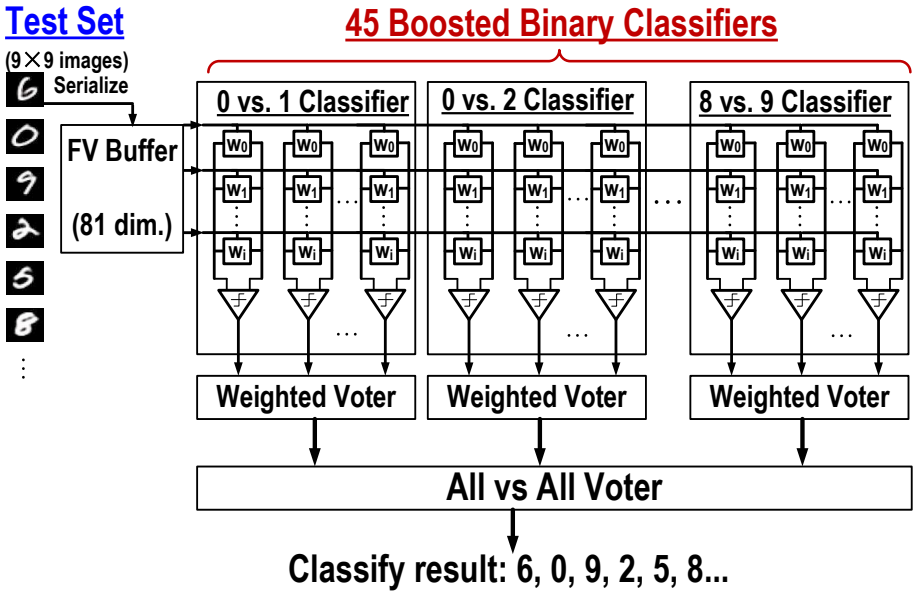
[H. W. Pan, IEDM'15]

Algorithmic co-design: chip specific AdaBoost

Error-Adaptive Classifier Boosting (EACB)

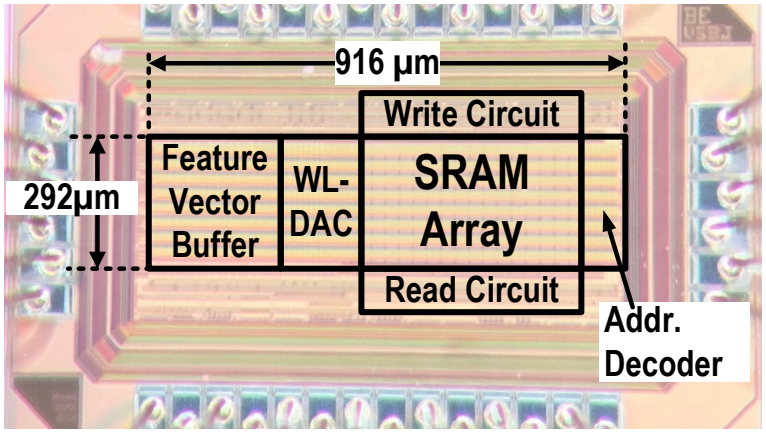


[Z. Wang, *TVLSI*'15][Z. Wang, *TCAS-I*'15]

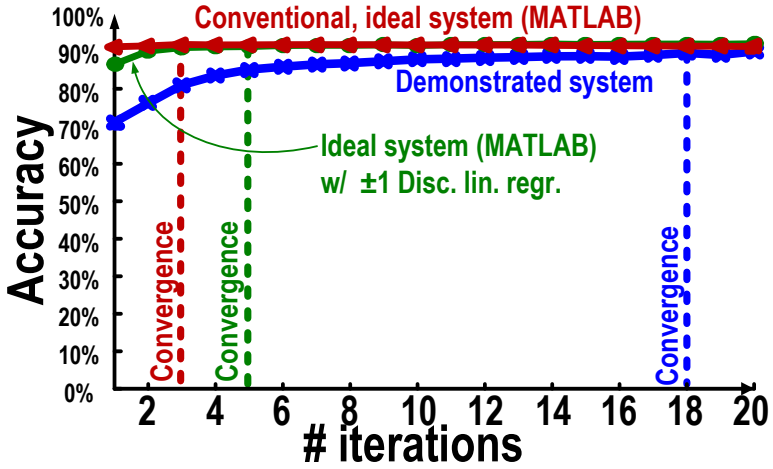


[J. Zhang, *VLSI*'16][J. Zhang, *JSSC*'17]

Boosted-linear classifier demonstration



MNIST Image Classification:

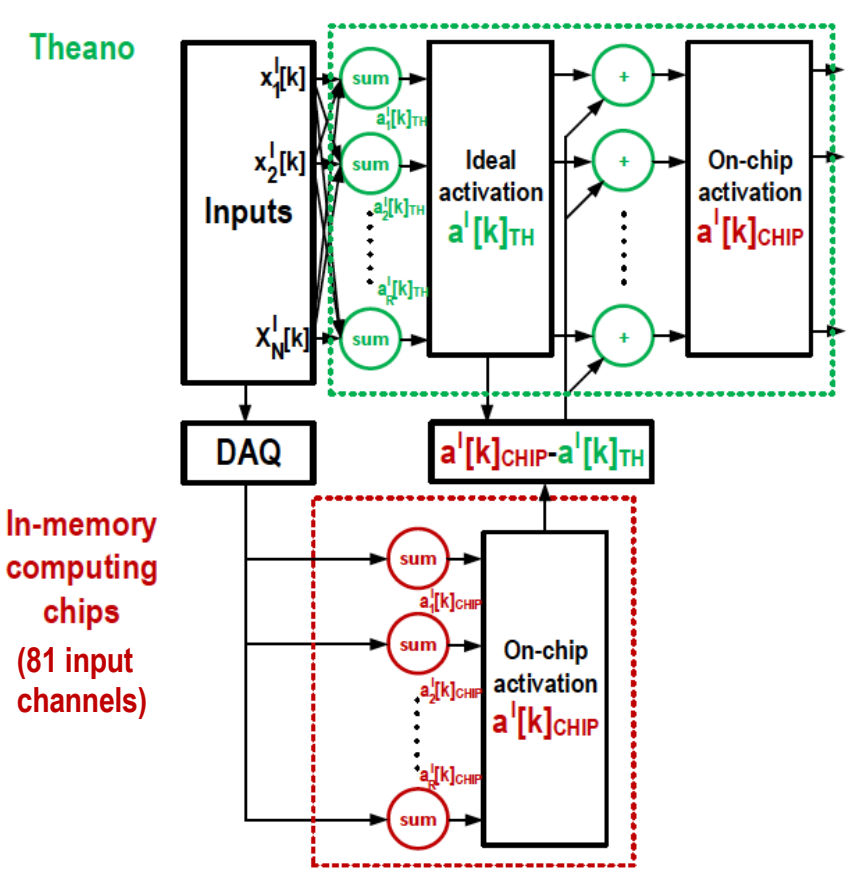


CHIP SUMMARY			
Technology	130 nm	Speed:	50MHz
SRAM Size	128 × 128 bits	Accuracy (81 feat.)	90% (18 iter.)
Bit cell Size	1.26 µm × 3.44 µm	Energy saving	12 ×
Energy / 10-way Class.	633.4 pJ	Feature Resolution	5b

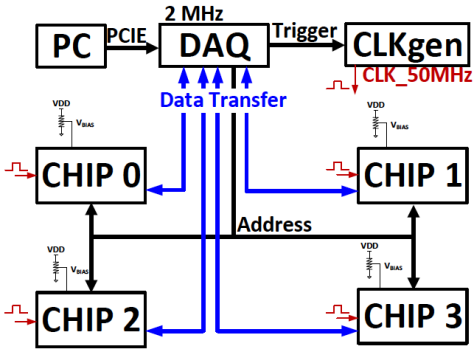
[J. Zhang, VLSI'16]

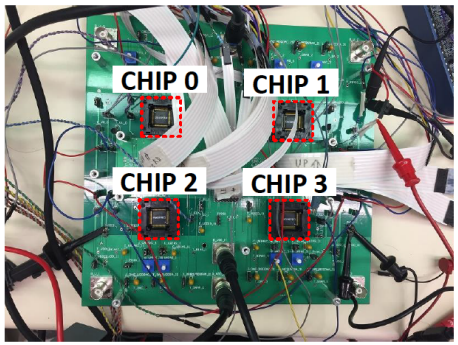
Algorithmic co-design: chip-specific DNN

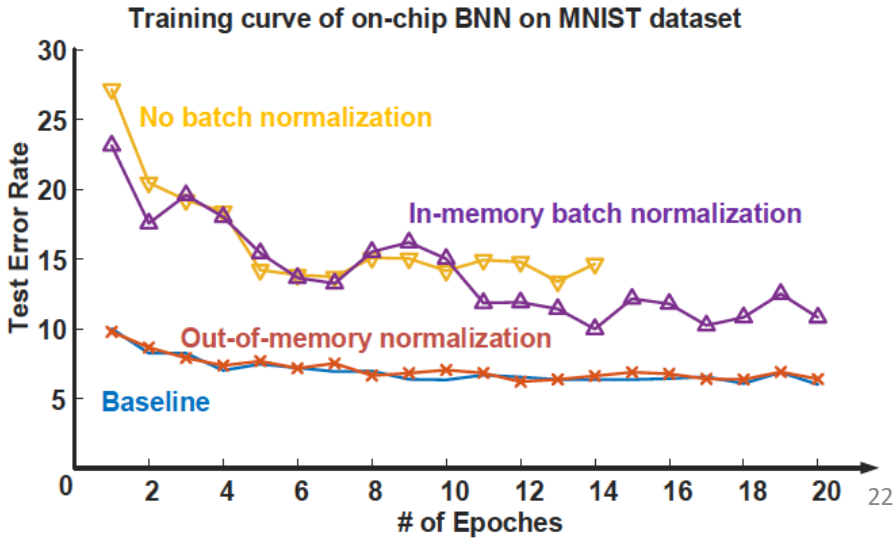
Chip-aware loss function (forward pass)



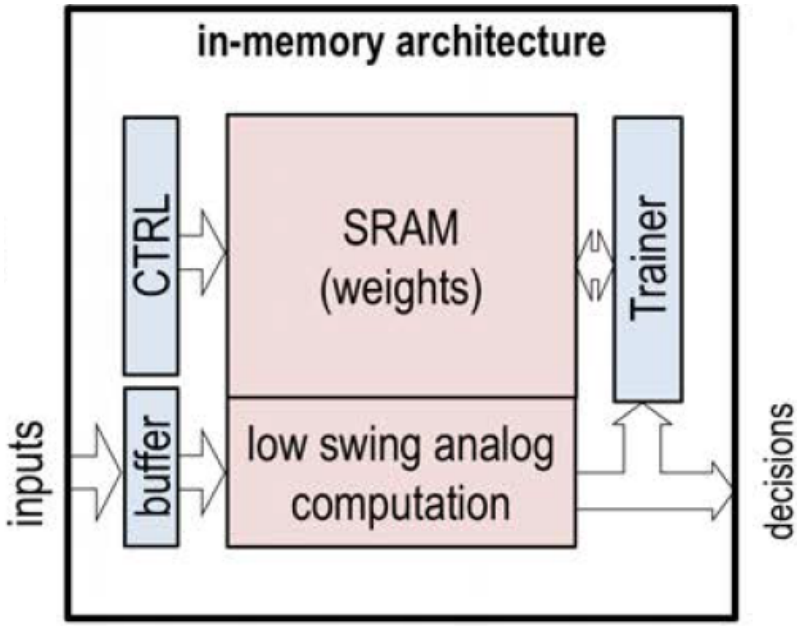
[J. Zhang, IEEE JETCAS'19]





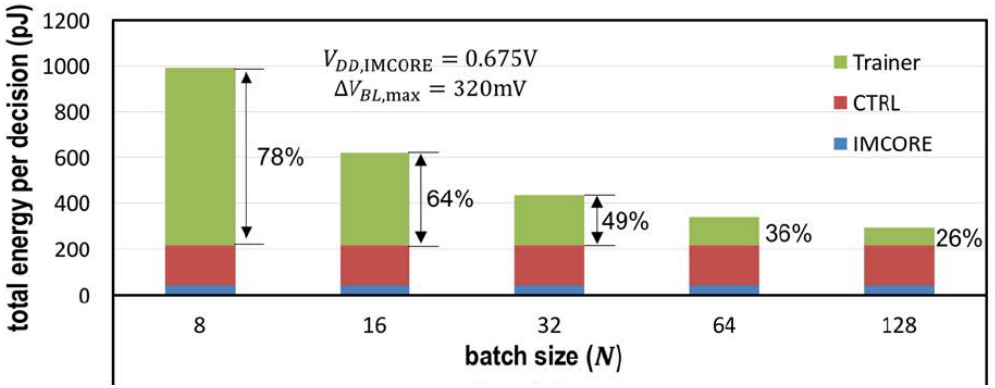


Embedded weight tuning

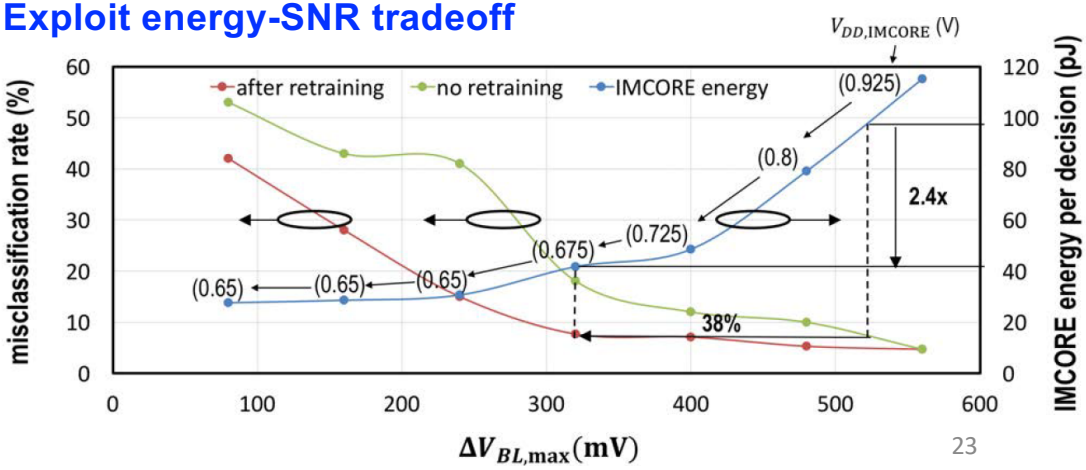


[S. Gonugondla, *ISSCC*'18]

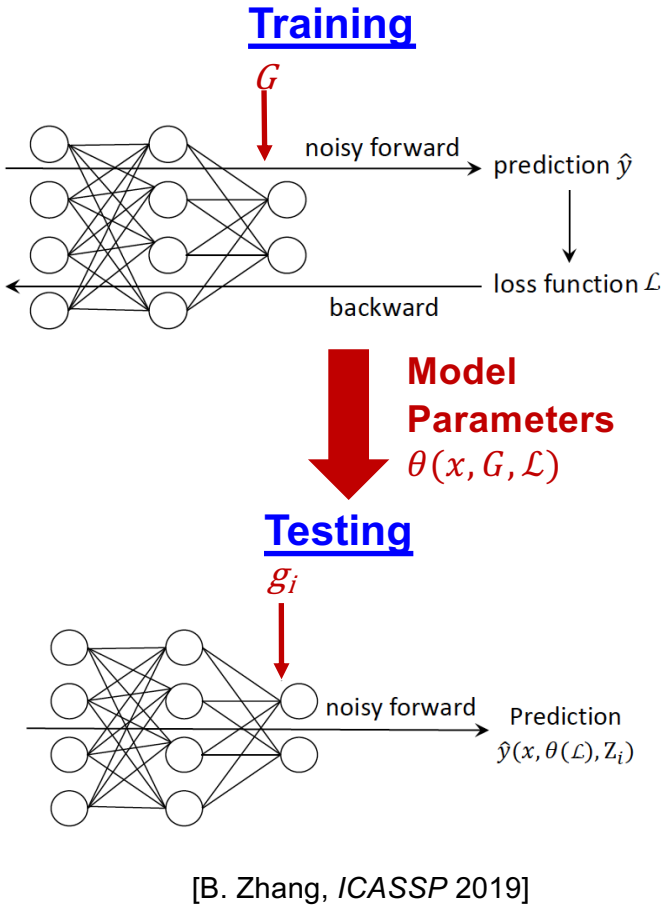
- **Adapt to chip & application variations**



- **Exploit energy-SNR tradeoff**

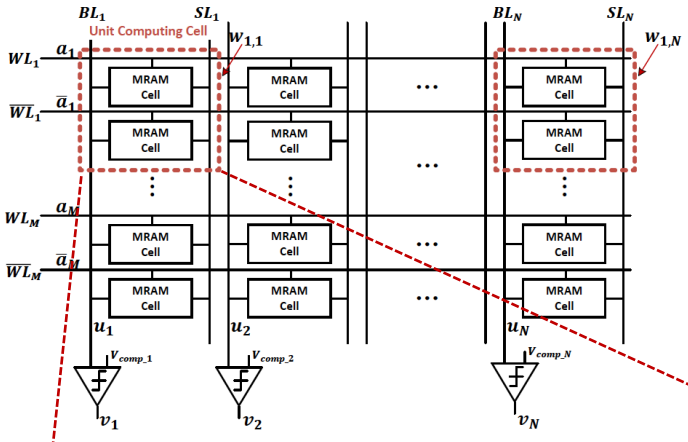


Algorithmic co-design: chip-generalized BNN

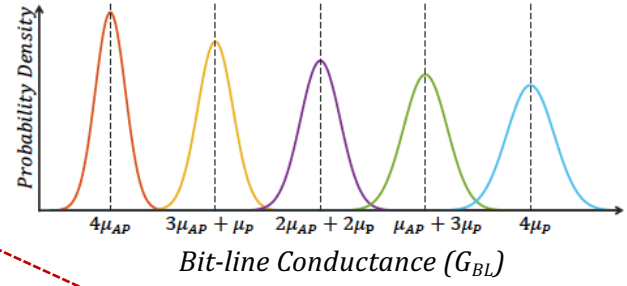


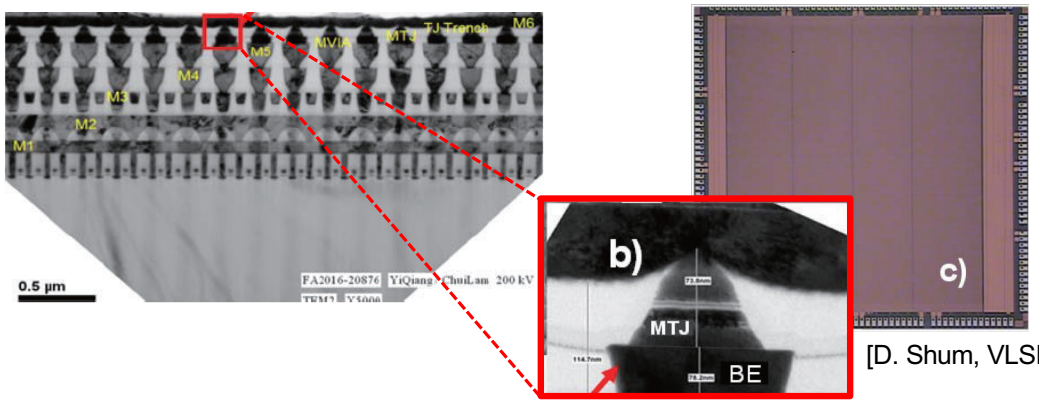
[B. Zhang, *ICASSP* 2019]

MRAM-based BNN (applied to CIFAR-10)



Stochastic hardware model:
(from device measurements)





[D. Shum, VLSI'17]

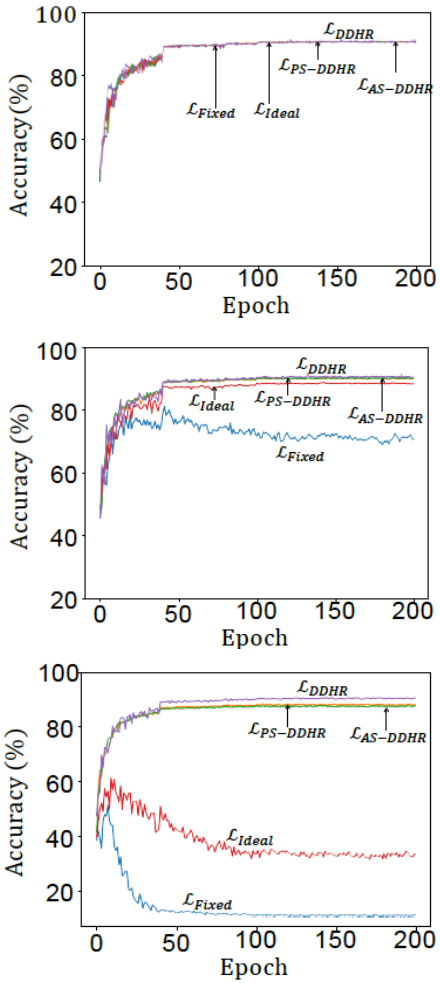
MRAM-based BNN simulations (CIFAR-10 classification)

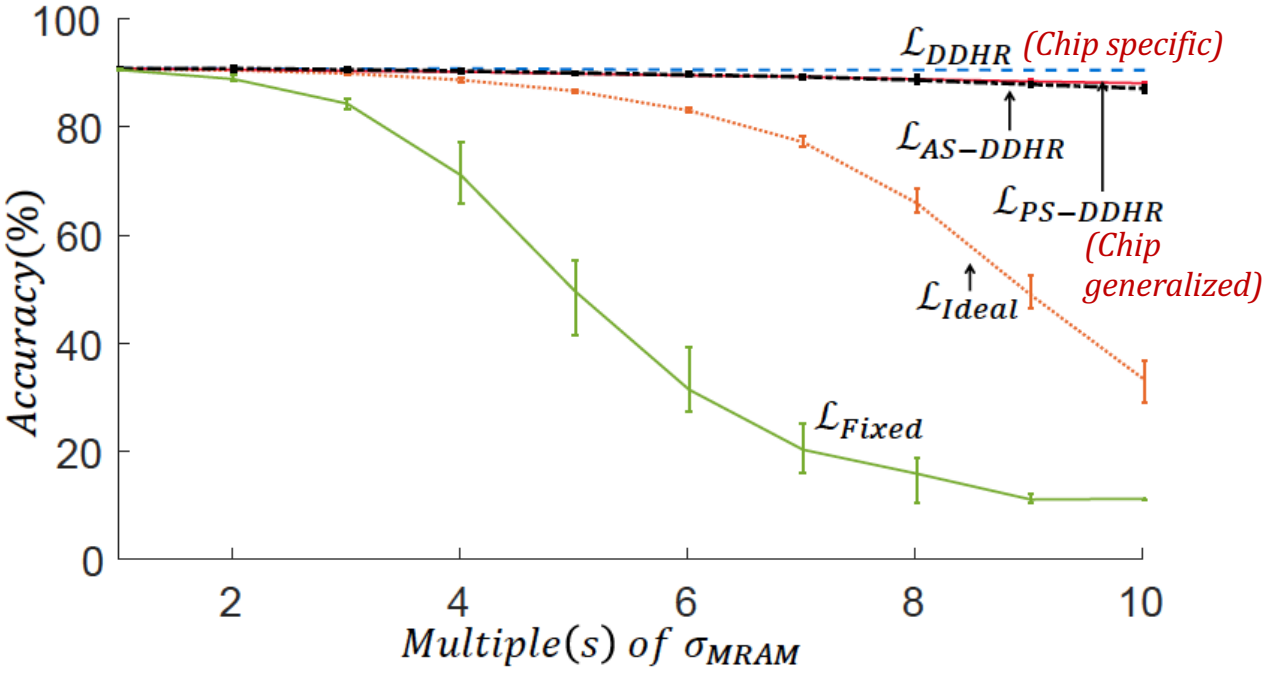
Variation level:

($1\times\sigma_{BL}$)

($4\times\sigma_{BL}$)

($10\times\sigma_{BL}$)

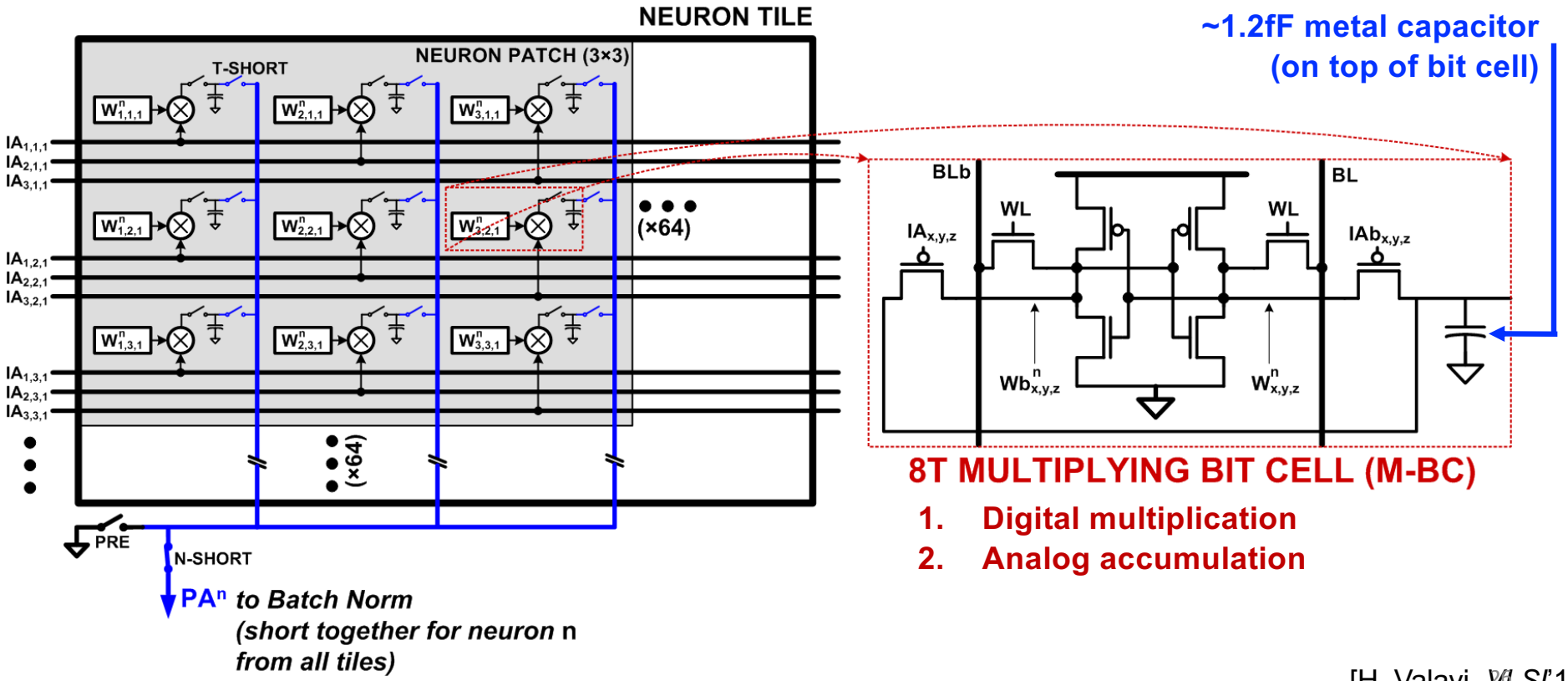




[B. Zhang, *ICASSP* 2019]

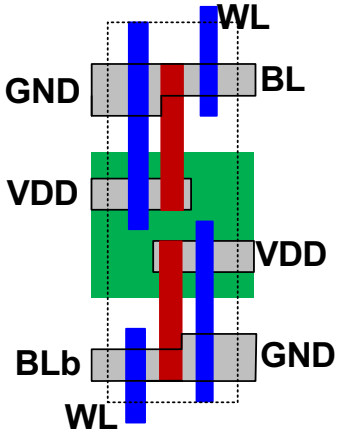
High-SNR analog computing

Charge-domain in-memory computing

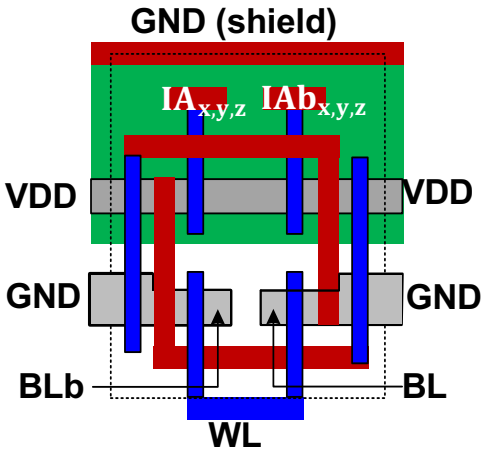


[H. Valavi, *VLSI'18*]

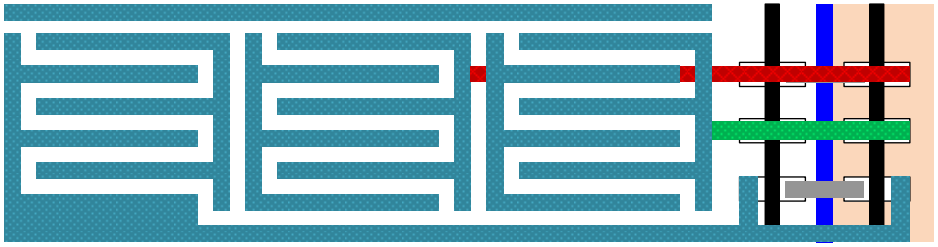
High-density/stability multiplying bit cell (M-BC)



(6T area: 1.0 A.U.)

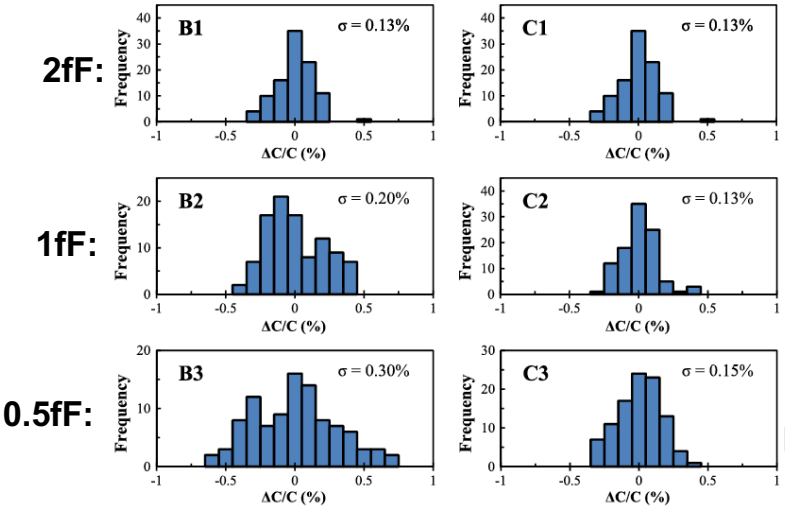


(M-BC area: 1.8 A.U.)



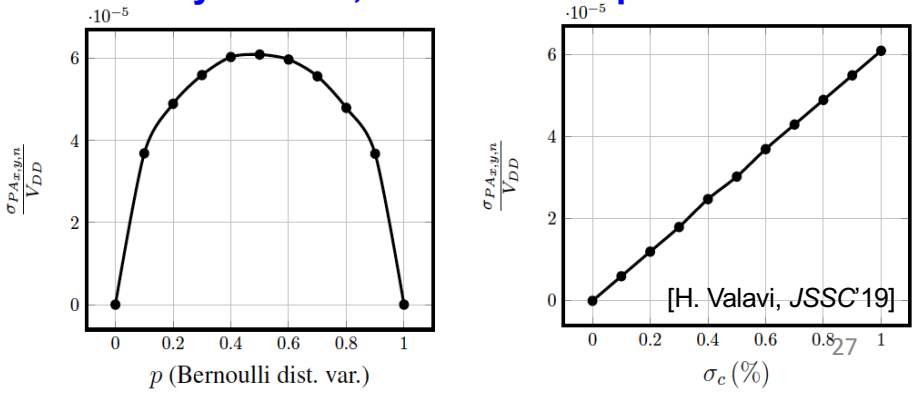
[H. Valavi, *VLSI*'18]

- **MOM-capacitor matching (130nm):**



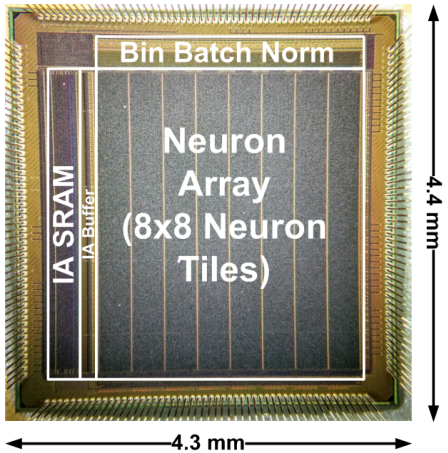
[H. Omran, *TCAS-I*'16]

- **IMC analysis – 10,000's of rows possible**

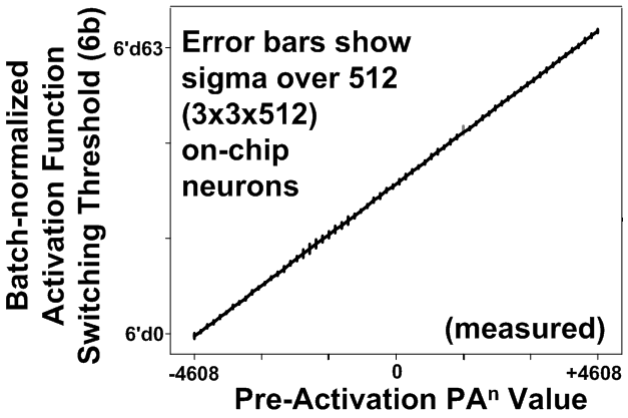


[H. Valavi, *JSSC*'19]

2.4Mb, 64-tile IMC



Neuron Transfer Function

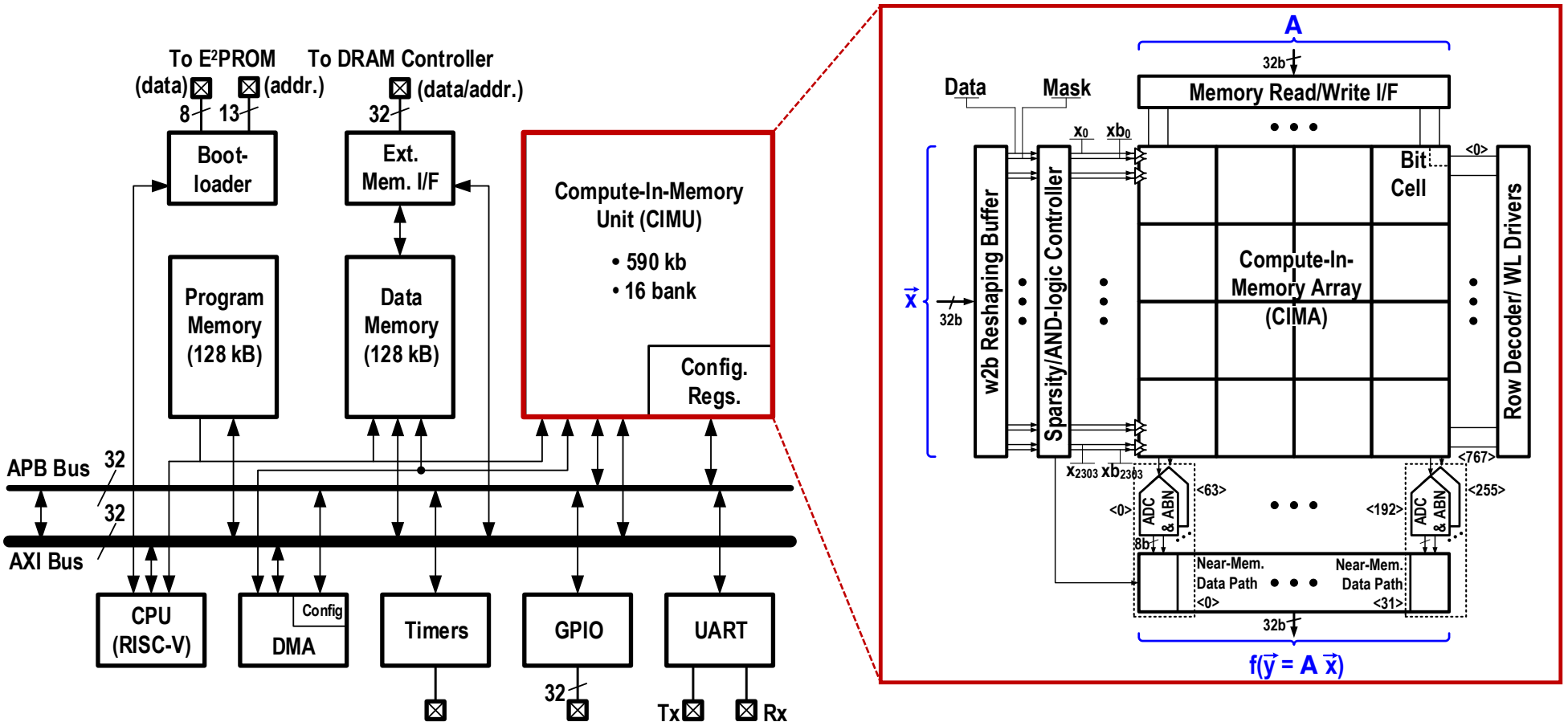


	Moons, ISSCC'17	Bang, ISSCC'17	Ando, VLSI'17	Bankman, ISSCC'18	Valavi, VLSI'18
Technology	28nm	40nm	65nm	28nm	65nm
Area (mm^2)	1.87	7.1	12	6	17.6
Operating VDD	1	0.63-0.9	0.55-1	0.8/0.8 (0.6/0.5)	0.94/0.68/1.2
Bit precision	4-16b	6-32b	1b	1b	1b
on-chip Mem.	128kB	270kB	100kB	328kB	295kB
Throughput (GOPS)	400	108	1264	400 (60)	18,876
TOPS/W	10	0.384	6	532 (772)	866

- 10-layer CNN demos for MNIST/CIFAR-10/SVHN at energies of 0.8/3.55/3.55 µJ/image
- Equivalent performance to software implementation

[H. Valavi, *VLSI'18*]

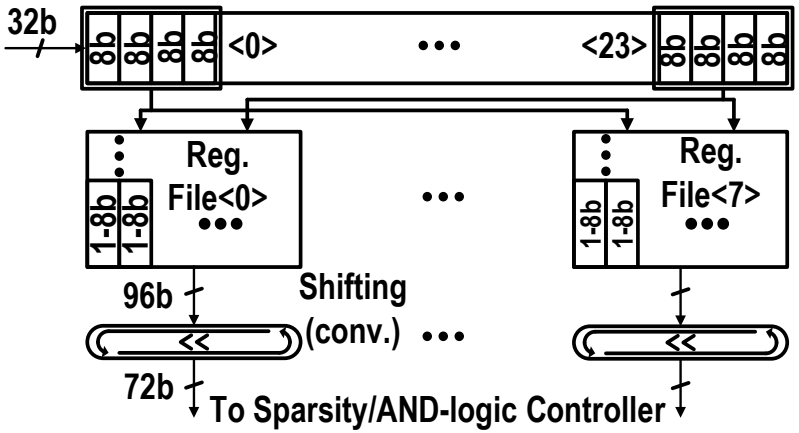
Programmable IMC

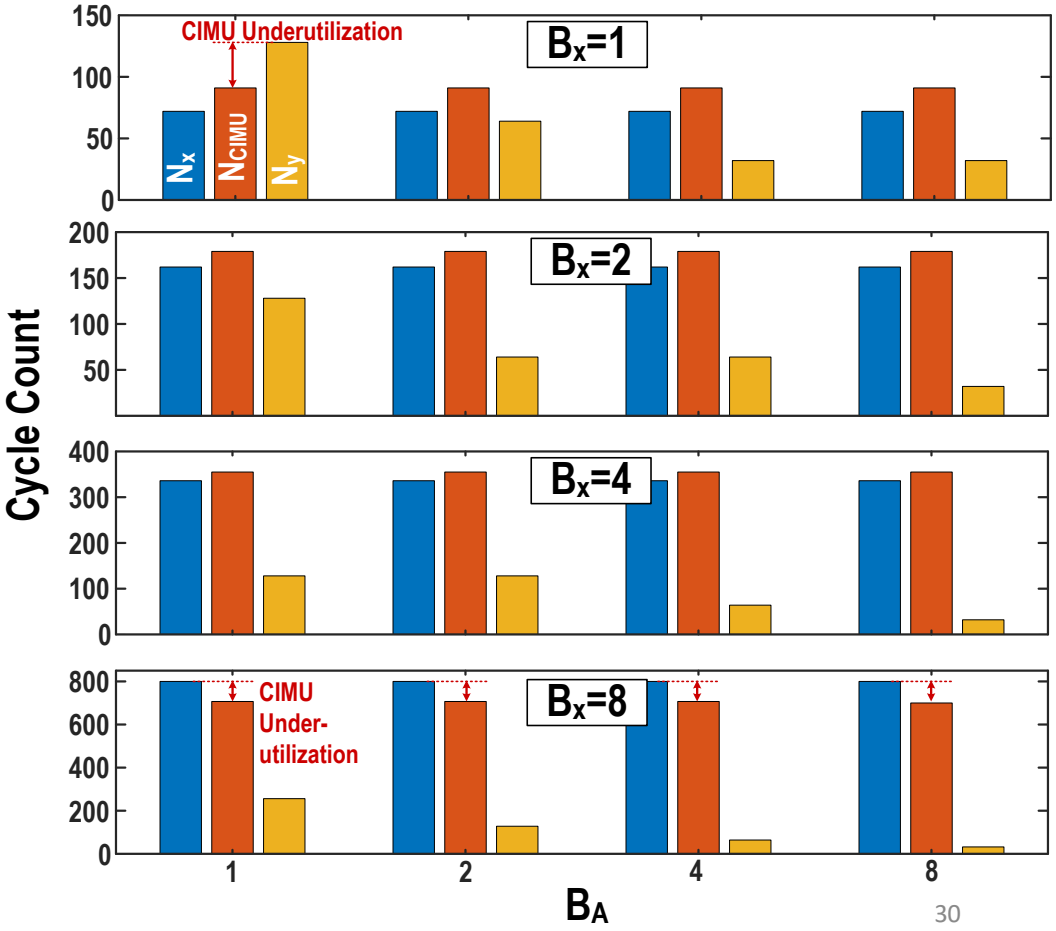


[H. Jia, *arXiv*:1811.04047]

CIMU interfacing

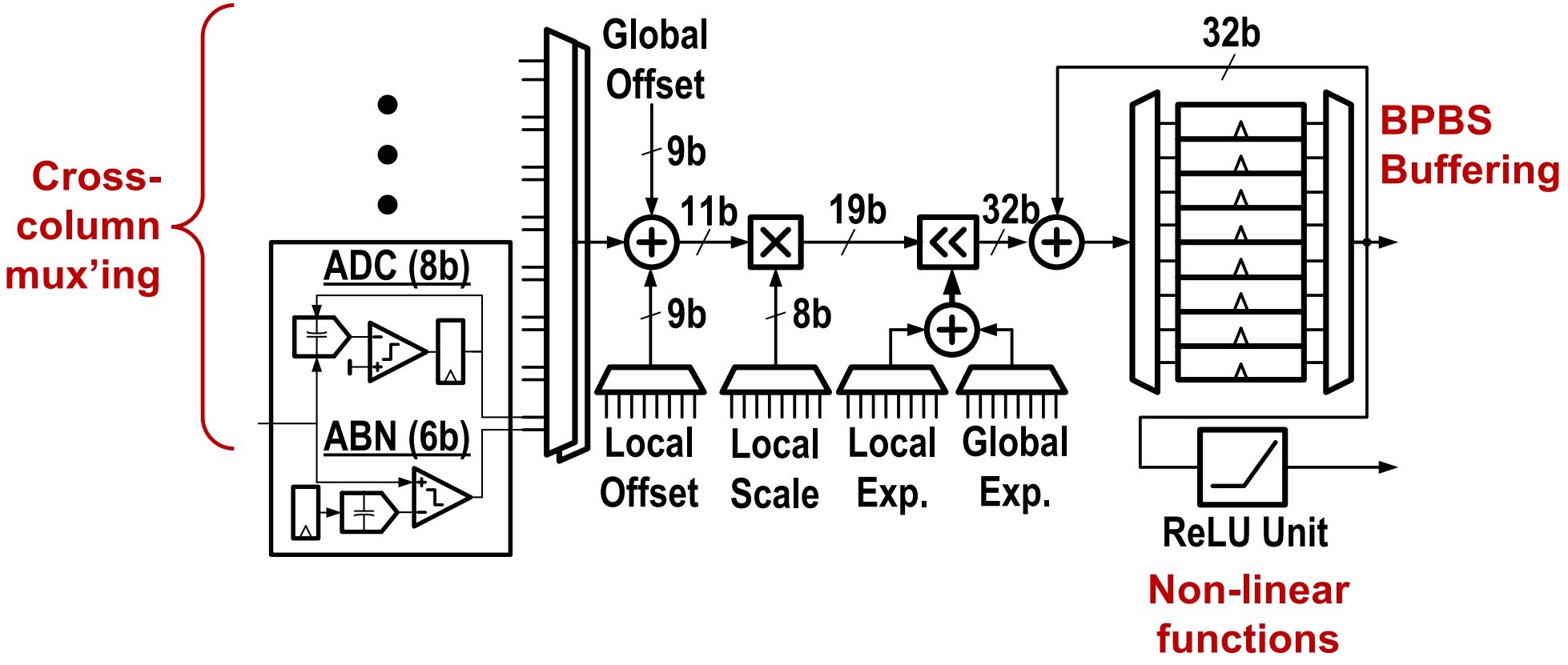
Word-to-Bit (W2b) reshaping buffer:



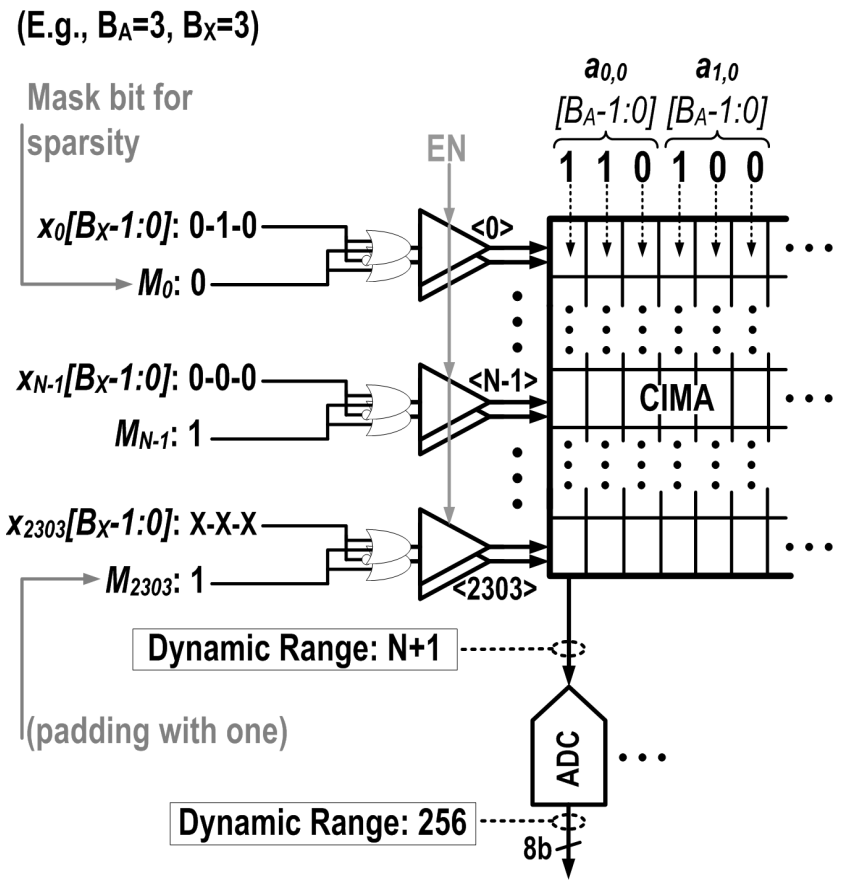


N_x: number of cycles to transfer $\vec{x}$
N_y: number of cycles to transfer $\vec{y}$
N_{CIMU}: number of cycles for CIMU compute
B_X: bit precision of $\vec{x}$ elements
B_A: bit precision of **A** elements

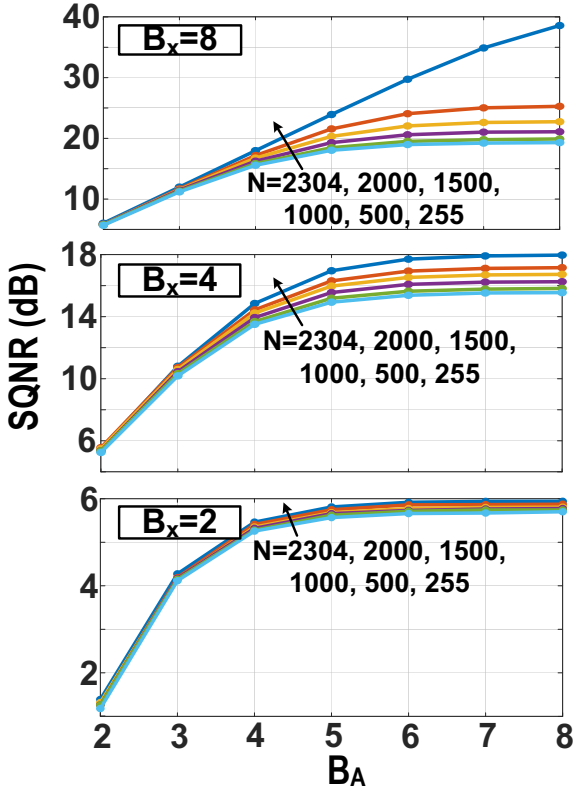
Near-memory computing



Bit-scalable mixed-signal compute

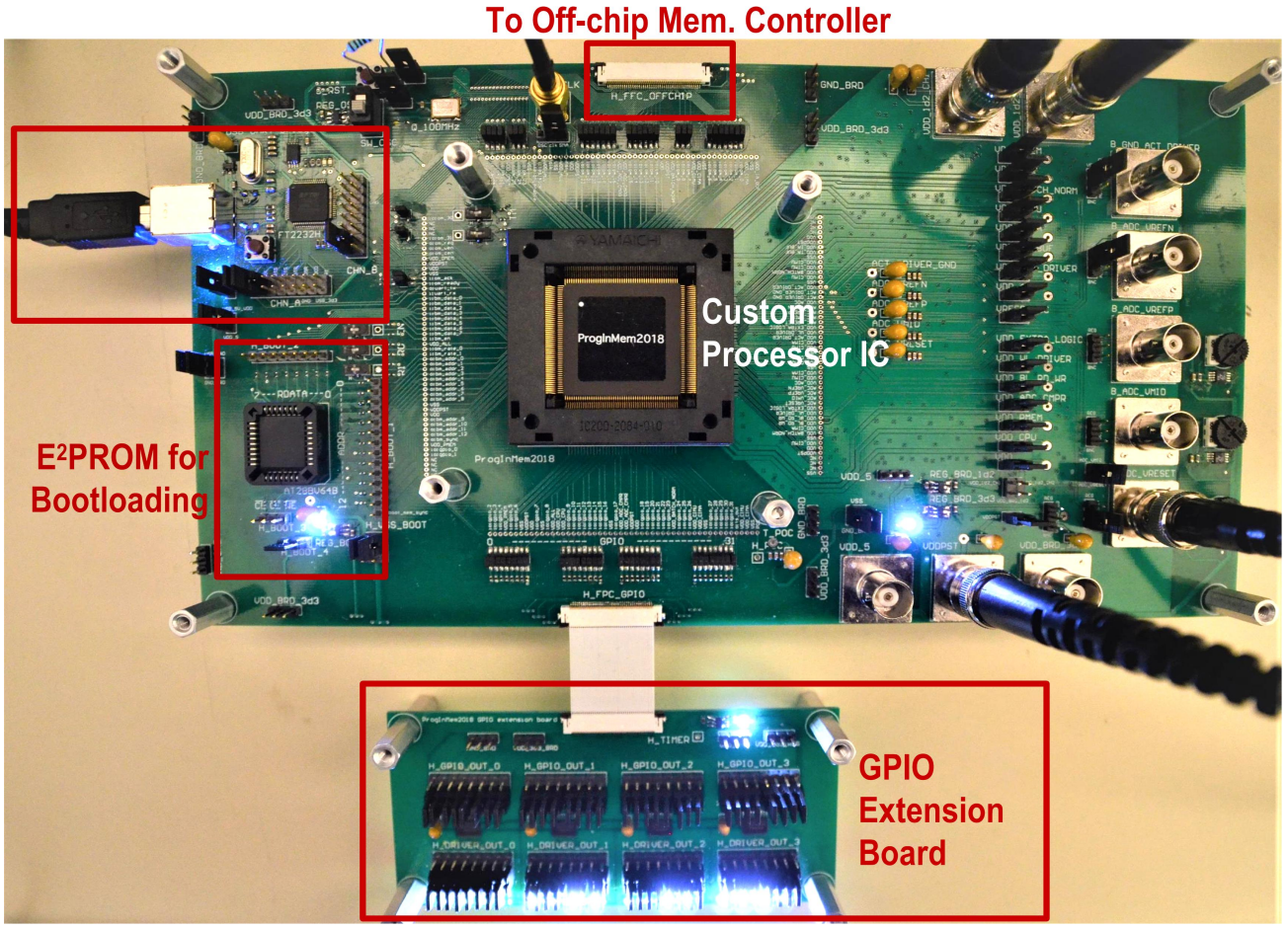


- **SQNR different that standard integer compute**



[H. Jia, *arXiv*:1811.04047]

Development board



Software libraries

1. Deep-learning Training Libraries (Keras)

Standard Keras libs:

```
Dense(units, ...)
Conv2D(filters, kernel_size, ...)
...
```

Custom libs:
(INT/CHIP quant.)

```
QuantizedDense(units, nb_input=4, nb_weight=4,
        chip_quant=False, ...)
QuantizedConv2D(filters, kernel_size, nb_input=4,
        nb_weight=4, chip_quant=False, ...)
...
```

```
QuantizedDense(units, nb_input=4, nb_weight=4,
        chip_quant=True, ...)
QuantizedConv2D(filters, kernel_size, nb_input=4,
        nb_weight=4, chip_quant=True, ...)
...
```

2. Deep-learning Inference Libraries (Python, MATLAB, C)

High-level network build (Python):

```
chip_mode = True
outputs = QuantizedConv2D(inputs,
        weights, biases, layer_params)
outputs = BatchNormalization(inputs,
        layer_params)
...
```

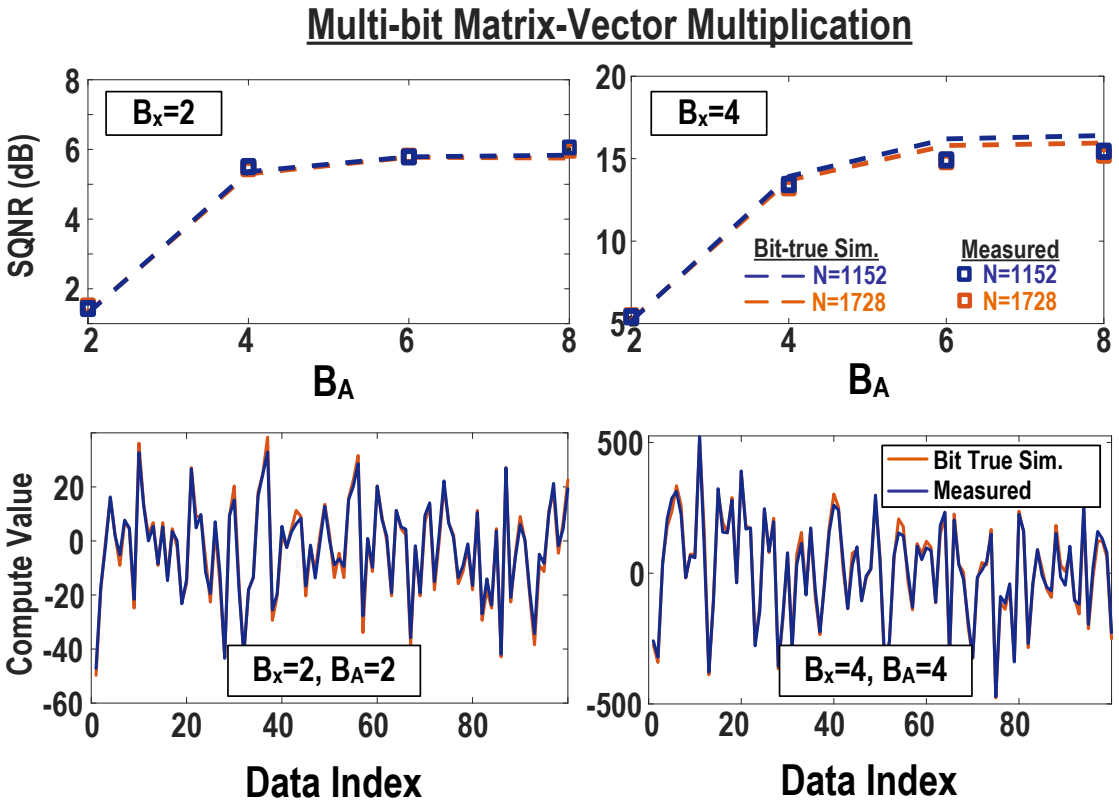
Function calls to chip (Python):

```
chip.load_config(num_tiles, nb_input=4,
        nb_weight=4)
chip.load_weights(weights2load)
chip.load_image(image2load)
outputs = chip.image_filter()
```

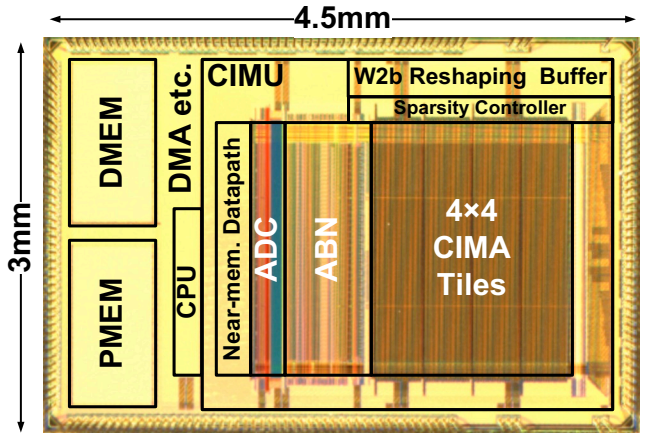
Embedded C:

```
chip_command = get_uart_word();
chip_config();
load_weights(); load_image();
image_filter(chip_command);
read_dotprod_result(image_filter_command);
```

Demonstrations



[H. Jia, *arXiv*:1811.04047]



Neural-Network Demonstrations		
	Network A (4/4-b activations/weights)	**Network B** (1/1-b activations/weights)
Accuracy of chip (vs. ideal)	92.4% (vs. 92.7%)	89.3% (vs. 89.8%)
Energy/10-way Class.[1]	105.2 µJ	5.31 µJ
Throughput[1]	23 images/sec.	176 images/sec.
Neural Network Topology	L1: 128 CONV3 – Batch norm L2: 128 CONV3 – POOL – Batch norm. L3: 256 CONV3 – Batch. norm L4: 256 CONV3 – POOL – Batch norm. L5: 256 CONV3 – Batch norm. L6: 256 CONV3 – POOL – Batch norm. L7-8: 1024 FC – Batch norm. L9: 10 FC – Batch norm.	L1: 128 CONV3 – Batch Norm. L2: 128 CONV3 – POOL – Batch Norm. L3: 256 CONV3 – Batch Norm. L4: 256 CONV3 – POOL – Batch Norm. L5: 256 CONV3 – Batch Norm. L6: 256 CONV3 – POOL – Batch Norm. L7-8: 1024 FC – Batch norm. L9: 10 FC – Batch norm.

Conclusions & summary

Matrix-vector multiplies (MVMs) are a little different than other computations

⟶ *high-dimensionality operands lead to data movement / memory accessing*

Bit cells make for dense, energy-efficient PE's in spatial array

⟶ *but require analog operation to fit compute, and impose SNR tradeoff*

Must focus on SNR tradeoff to enable
scaling (technology/platform levels) and architectural integration

In-memory computing greatly affects the architectural tradeoffs,
requiring new strategies for mapping applications

Acknowledgements: funding provided by ADI, DARPA, NRO, SRC/STARnet