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

\[
\begin{align*}
\vec{c} &= A \times \vec{b} \\
\begin{bmatrix}
  c_1 \\
  \vdots \\
  c_M 
\end{bmatrix} &=
\begin{bmatrix}
  a_{1,1} & \cdots & a_{1,N} \\
  \vdots & \ddots & \vdots \\
  a_{M,1} & \cdots & a_{M,N} 
\end{bmatrix}
\begin{bmatrix}
  b_1 \\
  \vdots \\
  b_N 
\end{bmatrix}
\end{align*}
\]

Processing Element (PE)

- Reuse accessed data for compute operations
- Specialized (memory-compute integrated) architectures
In-memory computing (IMC)

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

- In SRAM mode, matrix A stored in bit cells row-by-row
- In IMC mode, many WLs driven simultaneously \( \rightarrow \) amortize comm. cost inside array
- Can apply to diff. mem. Technologies \( \rightarrow \) enhanced scalability \( \rightarrow \) 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 $\times \sqrt{D}$ rows.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Traditional</th>
<th>In-memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth</td>
<td>$1/D^{1/2}$</td>
<td>$1$</td>
</tr>
<tr>
<td>Latency</td>
<td>$D$</td>
<td>$1$</td>
</tr>
<tr>
<td>Energy</td>
<td>$D^{3/2}$</td>
<td>$\approx D$</td>
</tr>
<tr>
<td>SNR</td>
<td>$1$</td>
<td>$\approx 1/D^{1/2}$</td>
</tr>
</tbody>
</table>

- 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:**

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

**IMC:**

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

**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 \( E_{BL,RD}/E_{BL,F-RD} \) operations, but not \( E_{WL} \)
- Usually \( E_{BL,F-RD} > E_{BL,RD} \), sometimes \( E_{WL,F-RD} < E_{WL,RD} \)
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

$C = AB$
IMC as a spatial architecture

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

<table>
<thead>
<tr>
<th>Operation</th>
<th>Digital-PE Energy (fJ)</th>
<th>Bit-cell Energy (fJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage</td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Multiplication</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>Accumulation</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>Communication</td>
<td>40</td>
<td>5</td>
</tr>
<tr>
<td>Total</td>
<td>590</td>
<td>55</td>
</tr>
</tbody>
</table>
Where does IMC stand today?

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

![Diagram showing energy efficiency and normalized throughput comparison between IMC and Non-IMC technologies.](image-url)
IMC challenge (1): analog computation

- Need analog to ‘fit’ compute in bit cells (SNR limited by analog non-idealities)
  \[ \rightarrow \text{Must be feasible/competitive @ 16/12/7nm} \]

---

**Noise Limited**

- Power costs
  - Digital
  - Analog

- Limit set by 1/f noise for a fixed area consumption

---

**Linearity/variation Limited**

- Bit cell Current (μA)
  - (10k-pt MC sim.)

---

[R. Sarpeshkar, *Ultra Low Power Bioelectronics*]
IMC Challenge (2): heterogeneity

- Matrix-vector multiply is only 70-90% of operations
  \[\rightarrow\] IMC must integrate in programmable, heterogeneous architectures

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

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

[B. Fleischer, VLSI'18]
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
- $E_{\text{DRAM}\rightarrow\text{IMC}}/4\text{-bit}: 40\text{pJ}$
- Reuse: $N \times I \times J$ (10-20 lys)
- $E_{\text{MAC,4-b}}: 50\text{fJ}$

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

![Diagram showing activation and weight accessing]

Memory Bound

Compute Bound

Reuse ≈ 1k

Operations/(Sec.$\times$Watts), OPS/W

Compute Intensity, CI
Neural-network trend (1): reducing bit precision

Opportunity for top-down and bottom-up design

Data statistics

LEARNING MODELS & PARAMETER TRAINING ALGORITHMS

HW statistics

~75mm² for 100M-parameter model (28nm)

(courtesy IBM)

[M. Verhelst, ISSCC SC on Machine Learning (2018)]
Neural-network trend (2): varied models

**Images (SPATIAL structure)**

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

**Language (SEQUENTIAL structure)**

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

**Inverse problems (PHYSICS)**

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

Ex. Fluid flow:
Neural-network trend (3): diverse use cases & pipelines

→ Diverse pipelines, diverse batch sizes, varying latency requirements

[L. Wang, CVPR 2015]
Neural-network trend (4): model compression, etc.

Weight Pruning

[Image: Graph showing relative execution time vs. pruning rate for traditional and SIMD-aware weight pruning.]

Depth-wise Separable Convolutions

[Image: Diagram illustrating depth-wise separable convolutions.]

[Text: J. Yu, ISCA’17]

[Text: 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]
Boosted-linear classifier demonstration

CHIP SUMMARY

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

MNIST Image Classification:

- Conventional, ideal system (MATLAB)
- Demonstrated system
- Ideal system (MATLAB) w/ ±1 Disc. lin. regr.

Accuracy vs. # iterations

[J. Zhang, VLSI'16]
Algorithmic co-design: chip-specific DNN

Chip-aware loss function (forward pass)

In-memory computing chips (81 input channels)

[J. Zhang, IEEE JETCAS’19]
Embedded weight tuning

- Adapt to chip & application variations

- Exploit energy-SNR tradeoff

[S. Gonugondla, ISSCC’18]
Algorithmic co-design: chip-generalized BNN

MRAM-based BNN (applied to CIFAR-10)

Stochastic hardware model: (from device measurements)

Bit-line Conductance ($G_{BL}$)

Model Parameters $\theta(x, G, L)$

Training

Testing

[B. Zhang, ICASSP 2019]

[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

~1.2fF metal capacitor (on top of bit cell)

8T MULTIPLYING BIT CELL (M-BC)

1. Digital multiplication
2. Analog accumulation

[H. Valavi, VLSI’18]
High-density/stability multiplying bit cell (M-BC)

- MOM-capacitor matching (130nm):
  - 2fF:
    - B1: \( \sigma = 0.13\% \)
    - C1: \( \sigma = 0.13\% \)
  - 1fF:
    - B2: \( \sigma = 0.20\% \)
    - C2: \( \sigma = 0.13\% \)
  - 0.5fF:
    - B3: \( \sigma = 0.30\% \)
    - C3: \( \sigma = 0.15\% \)

- IMC analysis – 10,000’s of rows possible

\[ \frac{\sigma_{F_{I/A}}}{V_{DD}} \]
\[ \sigma_e(\%) \]

(6T area: 1.0 A.U.)
(M-BC area: 1.8 A.U.)

[H. Omran, TCAS-I'16]

[H. Valavi, JSSC'19]

[H. Valavi, VLSI'18]
2.4Mb, 64-tile IMC

- 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

<table>
<thead>
<tr>
<th></th>
<th>Moons, ISSCC’17</th>
<th>Bang, ISSCC’17</th>
<th>Ando, VLSI’17</th>
<th>Bankman, ISSCC’18</th>
<th>Valavi, VLSI’18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>28nm</td>
<td>40nm</td>
<td>65nm</td>
<td>28nm</td>
<td>65nm</td>
</tr>
<tr>
<td>Area (mm²)</td>
<td>1.87</td>
<td>7.1</td>
<td>12</td>
<td>6</td>
<td>17.6</td>
</tr>
<tr>
<td>Operating VDD</td>
<td>1</td>
<td>0.63-0.9</td>
<td>0.55-1</td>
<td>0.8/0.8 (0.6/0.5)</td>
<td>0.94/0.68/1.2</td>
</tr>
<tr>
<td>Bit precision</td>
<td>4-16b</td>
<td>6-32b</td>
<td>1b</td>
<td>1b</td>
<td>1b</td>
</tr>
<tr>
<td>on-chip Mem.</td>
<td>128kB</td>
<td>270kB</td>
<td>100kB</td>
<td>328kB</td>
<td>295kB</td>
</tr>
<tr>
<td>Throughput</td>
<td>400 (60)</td>
<td>108</td>
<td>1264</td>
<td>18,876</td>
<td></td>
</tr>
<tr>
<td>(GOPS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOPS/W</td>
<td>10</td>
<td>0.384</td>
<td>6</td>
<td>532 (772)</td>
<td>866</td>
</tr>
</tbody>
</table>

[H. Valavi, VLSI’18]
Programmable IMC

- CPU (RISC-V)
- Timer
- GPIO
- UART
- Bootloader
- External Memory Interface
- Data Memory (128 kB)
- Program Memory (128 kB)
- Config. Regs.

Compute-In-Memory Unit (CIMU)
- 590 kb
- 16 bank

To E²PROM (data) \( \times \) \( \times \) (addr.)
- To DRAM Controller (data/addr.)

APB Bus 32
- AXI Bus 32

CIMU interfacing

Word-to-Bit (W2b) reshaping buffer:

\[ \begin{array}{c}
\text{Reg.} \\
\text{File<0>}
\end{array} \quad \cdots \quad \begin{array}{c}
\text{Reg.} \\
\text{File<7>}
\end{array} \]

\[ \begin{array}{c}
\text{32b} \\
\begin{array}{c}
\text{Bit<0>} \\
\vdots
\end{array} \\
\begin{array}{c}
\text{Bit<23>} \\
\vdots
\end{array} \\
\text{Bit<96>} \\
\text{Bit<72>} \\
\text{Bit<8b>}
\end{array} \]

\[ \begin{array}{c}
\text{Shifting (conv.)} \quad \cdots \\
\text{To Sparsity/AND-logic Controller}
\end{array} \]

\( N_x \): number of cycles to transfer \( \bar{x} \)

\( N_y \): number of cycles to transfer \( \bar{y} \)

\( N_{\text{CIMU}} \): number of cycles for CIMU compute

\( B_x \): bit precision of \( \bar{x} \) elements

\( B_A \): bit precision of \( A \) elements

![Graph showing cycle count for different bit precisions](image-url)
Near-memory computing

Cross-column mux’ing

ADC (8b)

ABN (6b)

Global Offset

11b

9b

19b

32b

Local Offset

Local Scale

Local Exp.

Global Exp.

32b

BPBS Buffering

ReLU Unit

Non-linear functions
Bit-scalable mixed-signal compute

(E.g., $B_A=3$, $B_X=3$)

Mask bit for sparsity

$x_0[B_X-1:0]: 0-1-0$

$M_0: 0$

$x_{N-1}[B_X-1:0]: 0-0-0$

$M_{N-1}: 1$

$x_{2303}[B_X-1:0]: x-x-x$

$M_{2303}: 1$

Dynamic Range: $N+1$

(padding with one)

Dynamic Range: 256


• SQNR different than standard integer compute
Development board

To Host Processor

To Off-chip Mem. Controller

E²PROM for Bootloading

Custom Processor IC

GPIO Extension 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, ...)`

## 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

Multi-bit Matrix-Vector Multiplication

![Graphs showing SNR vs. bit width for different bit counts and network size.]

Neural Network Demonstrations

<table>
<thead>
<tr>
<th>Neural Network Topology</th>
<th>Network A (4/4-b activations/weights)</th>
<th>Network B (1/1-b activations/weights)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy of chip</td>
<td>Energy/10-way Class.¹</td>
</tr>
<tr>
<td></td>
<td>(vs. ideal)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>92.4% (vs. 92.7%)</td>
<td>105.2 μJ</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(vs. 89.8%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.31 μJ</td>
</tr>
<tr>
<td></td>
<td>Throughput¹</td>
<td></td>
</tr>
<tr>
<td></td>
<td>23 images/sec.</td>
<td>176 images/sec.</td>
</tr>
</tbody>
</table>

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