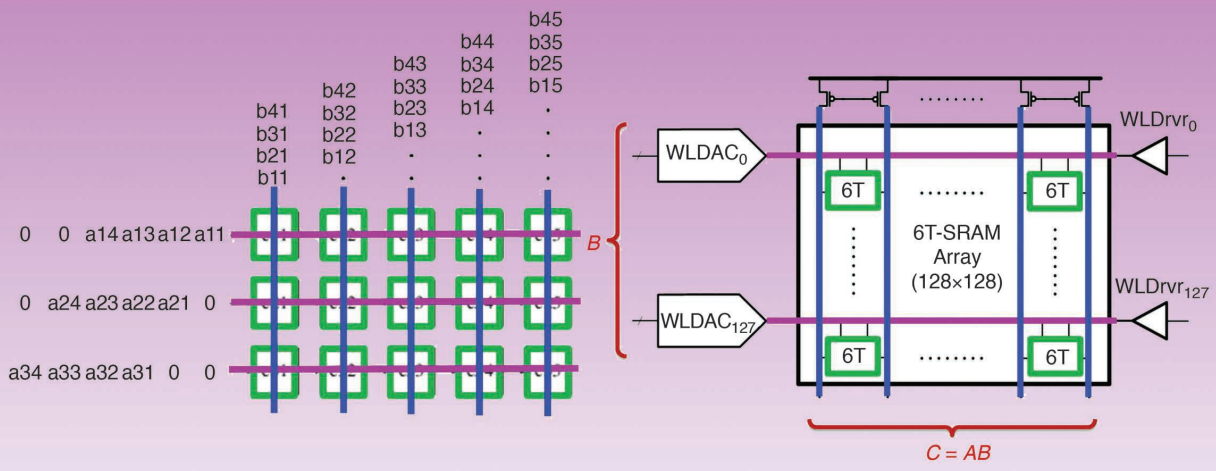
In-Memory Computing

Advances and prospects



Naveen Verma, Hongyang Jia, Hossein Valavi, Yinqi Tang, Murat Ozatay, Lung-Yen Chen, Bonan Zhang, and Peter Deaville

High-dimensionality matrix-vector multiplication (MVM) is a dominant kernel in signal-processing and machine-learning computations that are being deployed in a range of energy- and throughput-constrained applications. In-memory computing (IMC) exploits the structural alignment between a dense 2D array of bit cells and the dataflow in MVM, enabling opportunities to address computational energy and throughput. Recent prototypes have demonstrated the potential for 10× benefits in both metrics. However, fitting computation within an array of constrained bit-cell circuits imposes a number of challenges, including the need for analog computation, efficient interfacing with conventional digital accelerators (enabling the required programmability), and efficient virtualization of the hardware to map software. This article provides an overview of the fundamentals of IMC to better explain these challenges and then identifies promising paths forward among the wide range of emerging research.

Digital Object Identifier 10.1109/MSSC.2019.2922889
Date of publication: 27 August 2019

Communication Cost

The computational requirements in emerging applications of machine learning and signal processing are pushing platforms to their energy-efficiency and throughput limits. Hardware specialization has proven to be critical, with digital accelerators achieving 10–100× higher energy efficiency and speed than general-purpose processors. However, these gains apply primarily to computation, not memory accessing and, more generally, data movement [1]. Increasingly, the applications of interest are datacentric, involving large data sets or high-dimensionality data structures. This shifts the emphasis toward data movement, which unfortunately imposes limits on the gains possible from conventional digital acceleration.

A manifestation of this is the widely recognized "memory wall." Although often associated with accessing data from off-chip memory, the problem is more fundamental, arising in any architecture that separates memory and computation. Storing data in physical bit cells requires area, which thus scales with the amount of data to be stored. This raises a corresponding communication cost for moving data from the point of storage to the point of computation, outside the memory. To illustrate this cost, Figure 1 compares the energy required to access one word of data (64 b) from different-sized memories in a 45-nm technology to the energy of multiplication operations (considering the lowered precision levels that are increasingly relevant for deep-learning inference computations [2]–[4]).

The computational requirements in emerging applications of machine learning and signal processing are pushing platforms to their energy-efficiency and throughput limits.

Given that the communication cost is fundamental, computing architectures cannot avoid it, but they can attempt to amortize it. For example, general-purpose architectures employ caching in a memory system. This involves successively larger memories with correspondingly larger communication costs. However, those costs are incurred only when bringing data into the smaller memories, and then the statistical property of locality ensures that data are much more likely to be used for subsequent computations than data left in the larger memories. Specialized architectures being designed for specific computations can perform even more aggressive and deterministic amortization. For example, Figure 2 considers the case of MVM $\vec{c} = \mathbf{A} \times \vec{b}$, which has motivated spatial architectures. Here, processing engines (PEs) are arranged in a 2D array, matching the dataflow of MVM [5], [6]. PEs can have small, efficient local memories, storing only their intended operands. Furthermore, they move a computational result to adjacent PEs, which effectively amortizes the movement of all previously accessed operands.

MVMs represent a particularly important compute kernel due to their prominence in signal-processing and machine-learning computations. Fortunately, as seen in the case of spatial architectures, MVMs also present significant opportunities for amortization.

Next, we describe the basic approach of IMC, which exploits more aggressive amortization within the memory array itself.

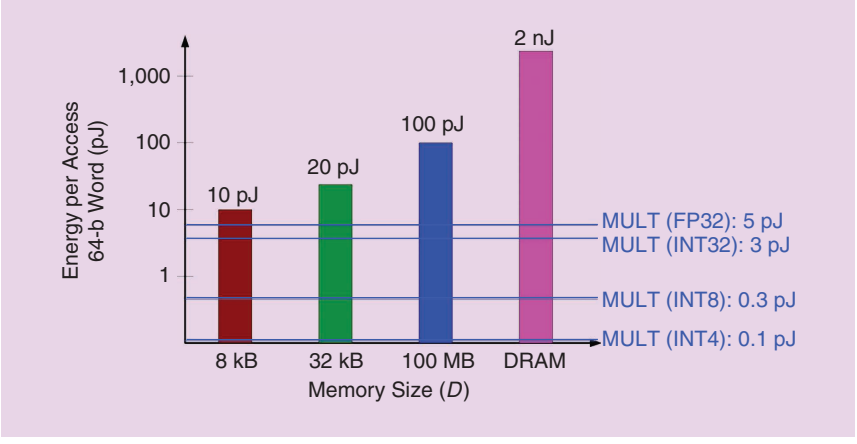


FIGURE 1: The energy cost of accessing memory.

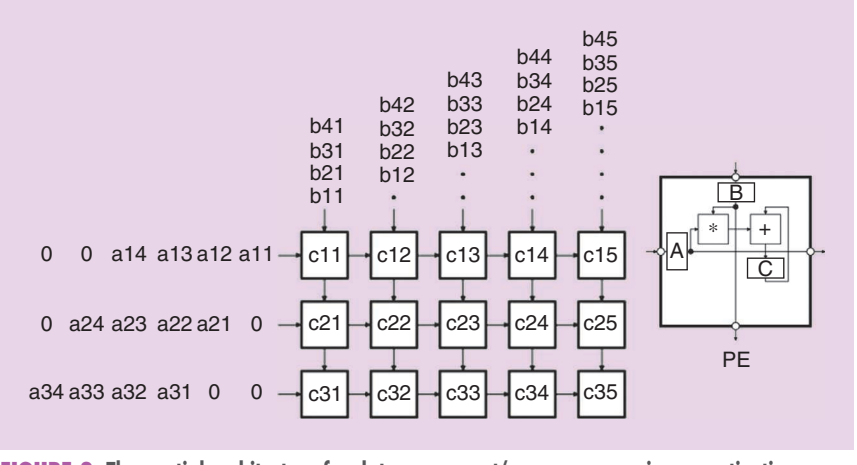


FIGURE 2: The spatial architecture for data-movement/memory-accessing amortization.

IMC Fundamentals

Figure 3 illustrates the approach of IMC, using the architecture from [7] as an example. The IMC architecture is based on an array of six-transistor static random-access memory (6T-SRAM) bit cells, and it includes periphery for two modes of operation. In SRAM mode, rows are accessed one at a time to read/write data via digital activation of a word line (WL). On the other hand, in IMC mode, multiple or all rows are accessed at once, using input data to activate the WLs. In the example, each WL digital-to-analog converter (WLDAC) applies an analog voltage corresponding to an input-vector element, which thus modulates the bit-cell current. Taking the bitlines (BL/BLb) as a differential signal, the stored data then have the effect of multiplying the input-vector element by +1/–1, and the currents from all bit cells in a column accumulate, generating an analog discharge of BL/BLb. This yields a multiply–accumulate (MAC) operation as required for MVM. Thus, instead of accessing raw bits row by row, IMC accesses a computation result over many/all bits, thereby amortizing the accessing costs. Note that such amortization is possible because of the structure of memory, where MVM involves mapping parallel input data to WLs and parallel computation to bit cells with stored data, followed by reducing output data via accumulation. Thus, IMC can apply to any memory technology integrated in such a structure [8]–[10].

Bandwidth/Latency/Energy Versus Signal-to-Noise Ratio Tradeoff

The amortization performed by IMC changes the basic tradeoffs associated with memory accessing. Figure 4 analyzes the typical metrics of interest

for memory (bandwidth, latency, and energy) and introduces a new metric relevant for computation, signal-to-noise ratio (SNR), by comparing a conventional architecture with IMC. Consider accessing D bits of data from a $\sqrt{D} \times \sqrt{D}$ memory [7]:

- *Bandwidth*: In the conventional architecture, bandwidth scales as $D^{1/2}$ with the number of row-wise access cycles required. In IMC, bandwidth does not scale because all rows are accessed at once.
- *Latency*: In the conventional architecture, latency scales as D due to the $D^{1/2}$ access cycles required and $D^{1/2}$ scaling of access-cycle delay with the BL/BLb capacitance. In IMC, latency does not scale, because activating all WLs causes the total bit-cell discharge current to scale at the same rate as BL/BLb capacitance.
- *Energy*: In the conventional architecture, energy scales as $D^{3/2}$ with the number of access cycles, BL/BLb capacitance, and number of columns. In IMC, energy scales only as D, because all data are accessed in one cycle.
- *SNR*: In the conventional architecture, SNR does not scale because the implicit assumption is that BL/BLb discharge remains fixed to maintain the required data-sensing margin. In IMC, SNR scales as $1/D^{1/2}$ because accumulation over $D^{1/2}$ bit cells results in a corresponding increase of dynamic range. Fitting this in a fixed BL/BLb swing causes a corresponding reduction of SNR.

From this first-order analysis, we see that IMC enables significant bandwidth/latency/energy gains at the cost of SNR. Some important secondary factors should be kept in mind. First, the analysis focuses on BL/BLb communication costs, which typically dominate in memory. However, IMC leaves WL activation costs unchanged (conventional and IMC architectures involve the same number of WL activations), somewhat reducing the total gains from IMC. Second, this analysis does not consider BL/BLb swing optimization. Conventional memory often exploits reduced swing, whereas IMC may prefer full swing due to BL/BLb dynamic-range requirements. Nonetheless, typical values of $D^{1/2}$ (e.g., 128 in [7]) open up the potential for significant gains through IMC, as we will see later in the article. However, the SNR tradeoff is a primary concern, challenging the scale, reliable specification, and configurability/programmability of computation, all of which are required for integration in modern computing platforms. Thus, emerging approaches to IMC must address the underlying issue of SNR demonstrated against these challenges.

Increasingly, the applications of interest are datacentric, involving large data sets or high-dimensionality data structures.

Comparison With Digital Architectures

Although the previous analysis compared memory accessing in IMC with

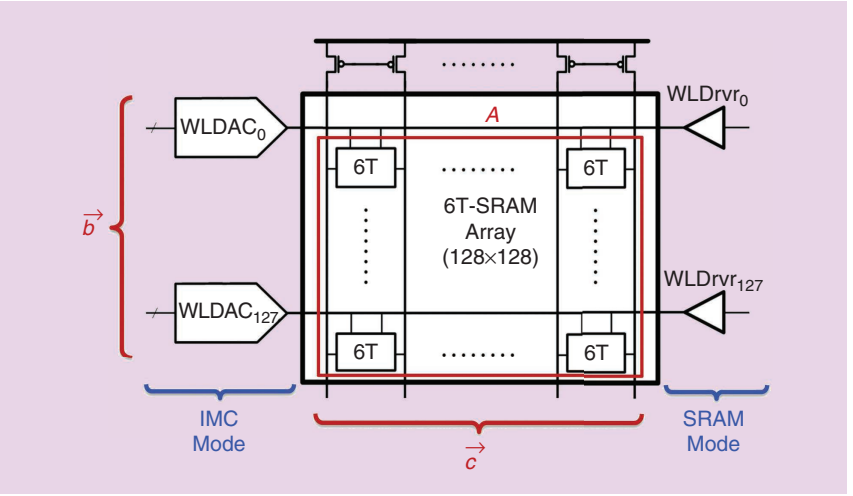
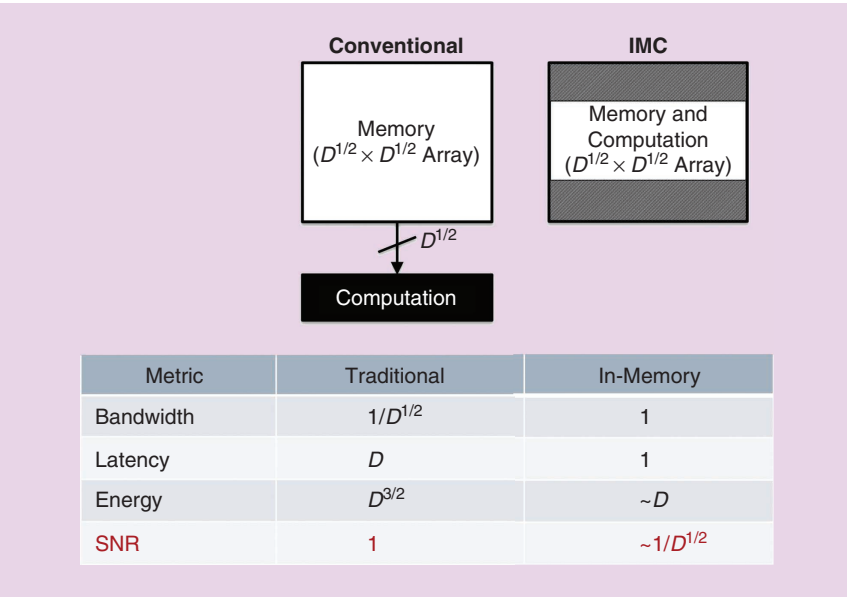


FIGURE 3: The basic approach of IMC illustrated in [7].



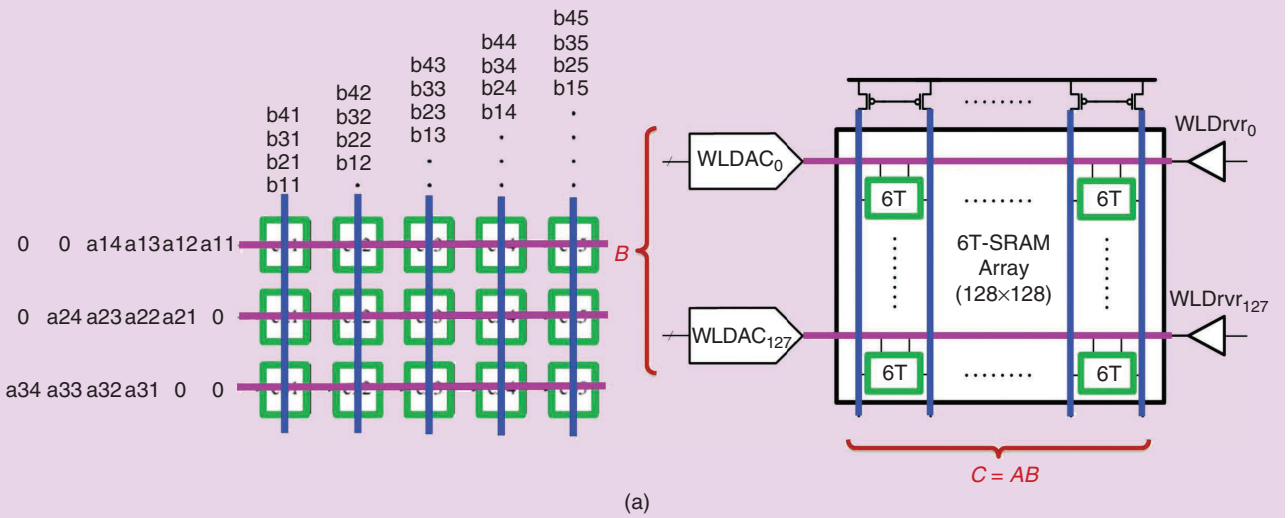
Metric	Traditional	In-Memory
Bandwidth	$1/D^{1/2}$	1
Latency	D	1
Energy	$D^{3/2}$	~D
SNR	1	~$1/D^{1/2}$

FIGURE 4: The basic tradeoffs with IMC.

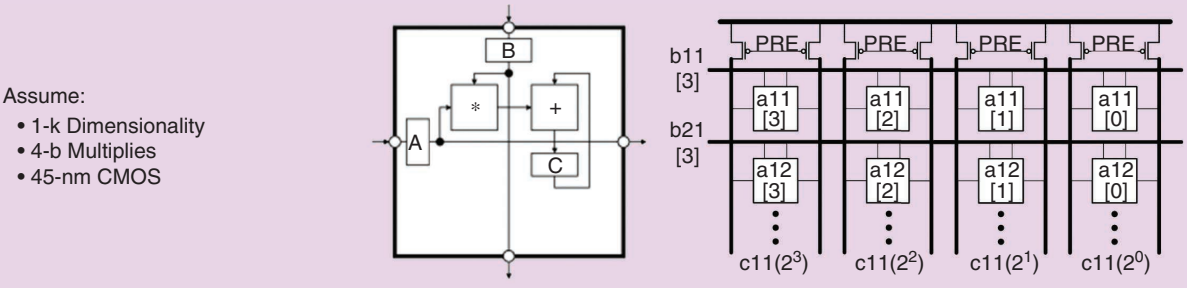
that in a canonical digital-accelerator architecture, MVMs have motivated more specialized spatial architectures. In fact, IMC can itself be regarded as a spatial architecture, but one where PEs are highly dense bit cells. Thus, the data-movement and computation costs can be compared.

Figure 5(a) considers data movement. One set of operands must move horizontally across the array (e.g., input-vector elements), one set can be stationary in the PEs (e.g., matrix elements), and one set must move vertically across the array (e.g., output-vector elements). In IMC, the horizontal movement is mitigated by the high density of bit cells, which are found to be 10–100× smaller in area (3–10× smaller in edge length) for corresponding MAC-computation hardware (e.g., [5] versus [7] and [11]). Furthermore, the vertical movement is mitigated both by the high density of bit cells and because the high-dynamic-range signal drives the capacitance of a single wire in analog IMC architectures and multiple wires in a digital architecture.

Figure 5(b) considers computation, using the design point of 4-b MACs and 1,024 dimensional vectors/matrices as an example (relevant to current trends in deep-learning inference systems). While this IMC architecture is restricted to 1-b stored operands, a recently proposed approach (described later in the article) extends to multiple bits by performing 1-b MACs serially and in parallel columns, followed by digitization, proper binary weighting (barrel shifting), and finally summation of the outputs [12]. This approach exploits an additional form of amortization along the memory-column dimension, in terms of the operations required



(a)



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

(b)

FIGURE 5: A comparison of IMC with spatial architectures: the (a) data movement and (b) computation.

for multibit computation. Namely, although IMC bit cells can perform 1-b logical operations (XNOR and AND), digital PEs require multibit operation (full adder). As we will show, such amortization once again imposes an SNR tradeoff. Using silicon measurements and postlayout simulations in a 45-nm technology, we see that, over the storage, multiply, accumulate, and data-movement operations performed by PEs, IMC holds the potential for 10× greater energy efficiency.

Current Standing of IMC

With IMC recently becoming an active research area, we have seen a number of prototype demonstrations that enable comparisons against digital-accelerator approaches. Figure 6(a) plots the area-normalized throughput versus energy efficiency of recent IMC and non-IMC prototypes, showing that IMC enables roughly 10× gains in each metric, which is expected based on the aforementioned analysis. Although this represents significant promise, Figure 6(b) exposes the primary challenge: plotting the total scale of computation achieved (the amount in memory integrated in prototypes). With the exception of one design [11] (considered in detail in the "High-SNR Circuit Design" section), IMC demonstrations have been highly limited in scale, primarily due to the fundamental SNR tradeoff described, especially in the context of analog computation. The following section takes a closer look at the challenges, using recent demonstrations as examples, to then motivate possible paths forward.

IMC Challenges and Approaches

Multiple IMC approaches have recently been proposed. To frame this discussion, it is useful to relate these to the fundamental tradeoffs developed in the "Bandwidth/Latency/Energy Versus Signal-to-Noise Ratio Tradeoff" section. For example, although some designs perform computation in memory, they may activate only one or two WLs at a time (e.g., [13] and [14]). This prevents significant amortization compared to standard memory accessing, yet it incurs the challenges of integrating computation in constrained bit-cell circuits (e.g., [13] requires adopting a 10-T bit cell, and [14] requires multiple memory-operation cycles). In such cases, standard memory accessing would likely be preferable, followed by computation just outside the memory array using less constrained circuits. Keeping in mind the fundamental IMC tradeoffs, the following sections survey the challenges, illustrated using recent design examples.

Circuit Challenges

Analog Nonidealities

To fit computation in constrained bit-cell circuits, IMC commonly employs analog operation, leveraging richer T behavior than that allowed by digital switch-based abstraction. With regard to the SNR tradeoff, the increased sensitivity to variations and nonlinearities now becomes the dominant source of computation noise. For instance, Figure 7 shows the nonlinearity and variation (standard deviation shown as error bars) of the BL/BLb output with respect to one input-vector element value in [7] and to the

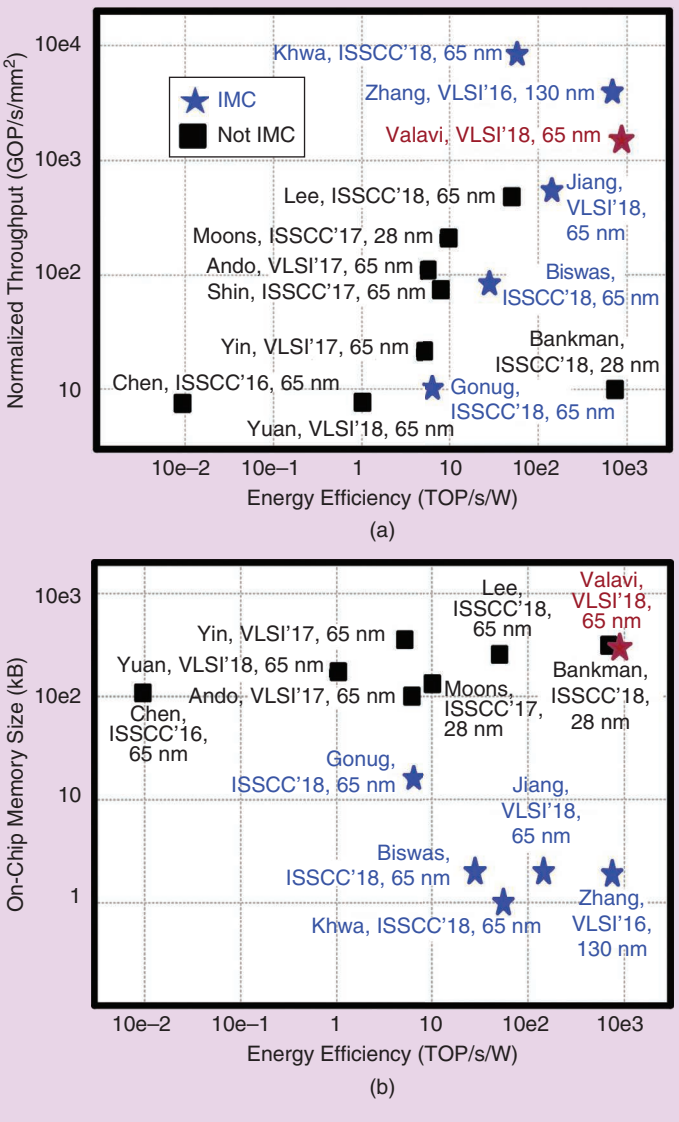


FIGURE 6: A comparison of IMC and non-IMC prototypes: the (a) throughput versus energy efficiency and (b) total memory versus area efficiency.

overall accumulation result in [15]. Additionally, using T currents as the output signal of each bit cell also raises sensitivities to temperature, which can strongly impact carrier transport in semiconductors.

One way to mitigate the effects of T nonlinearity arising from BL/BLb biasing is to employ low-swing signaling. However, the SNR tradeoff is then adversely affected by the increased impact of variations (and shot-noise sources) relative to the signal. In fact, higher computation SNR may be observed in practice by increasing swing [16]. Nonetheless, Figure 8 illustrates the approach in [16], also showing that the MAC operation possible in IMC columns can be used beyond MVMs. In this case, multibit data are stored in a column-major format, and different WL pulsewidths are applied in parallel for binary weighting, thereby yielding an analog read operation of a multibit word (configurable mixed-signal computation is then performed in the array periphery).

An important direction for IMC is incorporating alternate memory technologies—for instance, nonvolatile technologies—that can enable the possibility of multilevel storage, low-power duty-cycle operation, and increased density. The emerging memory technologies can include resistive RAM (RRAM), magnetic RAM (MRAM), and phase-change memory (PCM). Such technologies suffer from many of the same analog nonidealities as SRAM-based IMC. For instance, Figure 9(a) shows the variation of MAC outputs from a NOR-flash implementation [8]. On top of this, such technologies are likely to exacerbate the challenges with analog readout. For instance, multilevel storage further increases dynamic-range requirements, impacting the SNR tradeoff, and the relatively low resistance and low-resistance contrast of emerging memory technologies (RRAM and MRAM) increase the power and area of the readout circuitry, which already dominates in demonstrated systems [9], as shown in Figure 9(b).

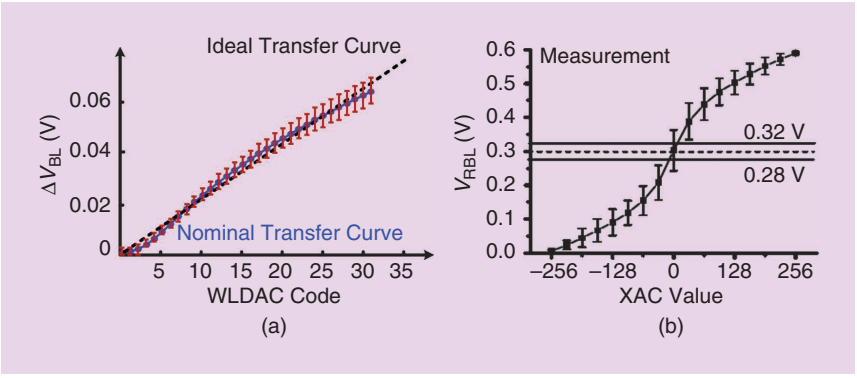


FIGURE 7: An illustration of analog-computation nonidealities: (a) the BL/BLb-discharge nonideality in [7] and (b) the read-BL nonideality in [15]. XAC: XNOR-and-accumulate value.

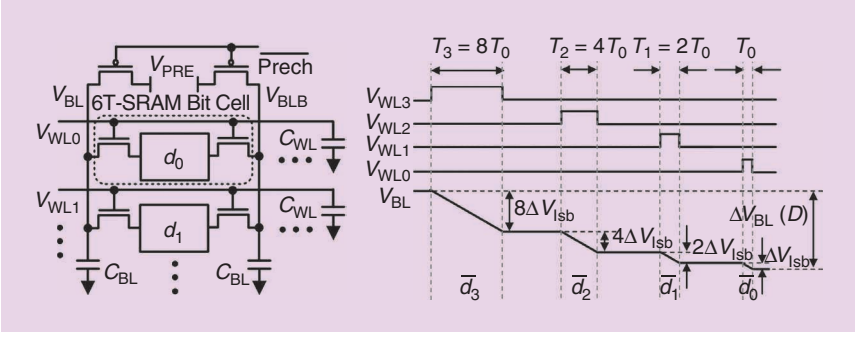


FIGURE 8: An analog readout of multibit data stored in bit-major format [16].

Power Delivery

IMC leverages the parallel structure of memory. However, such parallel operation also elevates peak

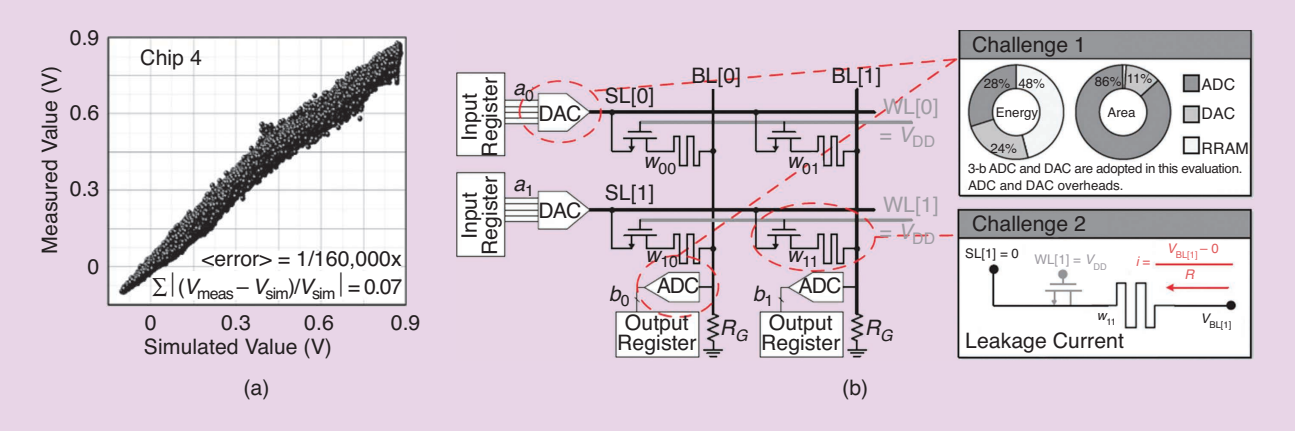


FIGURE 9: The challenges with nonvolatile memory technologies: the (a) variation of MAC outputs [8] and (b) energy/area consumption of readout circuitry [9]. ADC: analog-to-digital converter.

power-delivery requirements in at least two ways. First, activation of many/all WLs is fundamental to the amortization performed and thus requires corresponding power delivery to the WL drivers. This must be addressed through power-grid density, especially because noise on the WL levels can translate directly to computation noise. Second, operation of many/all bit cells is fundamental to the amortization performed. We can make a distinction between static and dynamic computation approaches. Figure 10(a) shows an example of static computation, employing a static buffer at the bit-cell output [15]. In this case, activating many/all bit cells leads to currents of up to 1 mA in each column, challenging the feasibility of low-noise power delivery. Figure 10(b) shows an example of dynamic computation, where the activated bit cells simply discharge the BL/BLb capacitance such that the total charge delivered never exceeds that of a standard full-swing read [7]. This requires designing against saturation of BL/BLb discharge, as is done in [7] by restricting WLDAC output range and minimizing BL/BLb leakage, which can result in computation noise.

Bit-Cell Stability

A consequence of accessing computational results rather than raw data is that activated bit cells are exposed to different data on BL/BLb than those stored. This raises the possibility of disrupting their stored data. Two approaches have been pursued to guard against this. First, buffered cells have been employed, whereby the critical data-storage nodes are isolated from the computed BL/BLb value [15]. This has the drawback of degrading density. Second, suitable bit-cell biasing has been employed. This has included low-swing signaling on BL/BLb such that bit-cell biasing remains close to standard SRAM read-condition biasing [16]. However, this adversely affects the SNR tradeoff. Alternatively, WL-biasing has been employed, wherein the WL voltage is kept low [7]. In addition to limiting the bit-cell current to guard against BL/BLb saturation in dynamic computation, this can ensure adequate isolation of the bit-cell storage nodes from the computed BL/BLb value.

Architectural Challenges

IMC primarily addresses MVM or other vector operations, which represent only a subset of computations required in practical applications. As an example, Figure 11 shows the profiling results for the many computations required in different neural-network (training) applications [17]. Although we see that MVM (shown as general matrix multiply) dominates, it is necessary for complete architectures to address the many other computations and to do so programmably and configurably. Importantly, the other computations are characteristically different than MVM in that they apply to element-wise (low-/single-dimensionality) operands. This reduces the emphasis on data movement that motivates IMC and instead enables the use of conventional digital acceleration. However, it is critical that IMC now integrates robustly and efficiently in larger heterogeneous architectures.

This raises two critical considerations. First, robust integration in larger architectures requires a well-defined functional specification of the IMC block. This is challenged by analog circuit nonidealities, which significantly affect computation SNR and are typically difficult to characterize (e.g., due to process, temperature, and voltage dependence) or statistical in nature (e.g., due to T variations). Second, heterogenous architectures are often limited by data movement

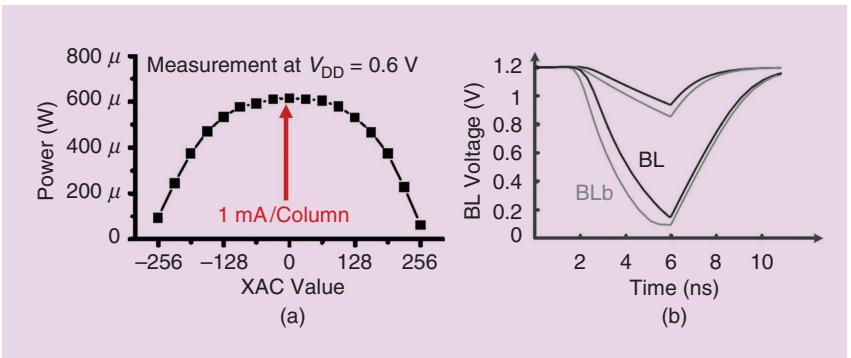


FIGURE 10: An illustration of power-delivery considerations in IMC: the (a) static current drive [15] and (b) dynamic current drive [7].

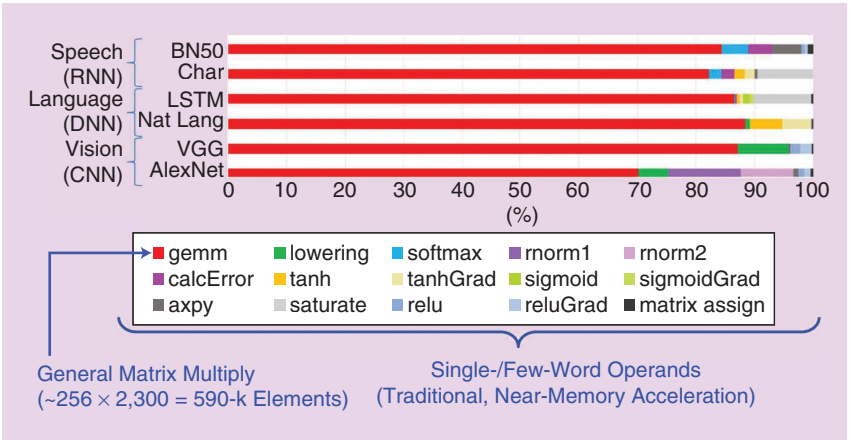


FIGURE 11: The range of computations required in neural-network applications [17]. BN: batch normalization; Char: character level; LSTM: long short-term memory; Nat Lang: natural language.

The IMC architecture is based on an array of six-transistor static random-access memory bit cells, and it includes periphery for two modes of operation.

and computation control between accelerators. Efficient integration thus requires a proper and flexible interface design between the highly parallel IMC inputs/outputs and more typical microprocessor blocks as well as specialized, domain-specific architectures matched to the dataflow in classes of applications. Although recent progress has been made in these areas, considerable opportunities for further research remain.

System Challenges

Computing systems must be able to support mapping of broad sets of applications. This raises the need for virtualization, where the limited hardware available is repurposed, reconfigured, and sequenced at runtime to efficiently support the execution of desired computations, typically specified through software. This is usually done through optimizers and automatic code generators in the compiler stack, which encapsulate algorithms for optimally mapping computations to the hardware. Given the significantly different physical tradeoffs presented by IMC (see the section "IMC Fundamentals") compared to conventional digital architectures, the algorithms must be carefully thought through to avoid losing the potential gains presented by IMC at the circuit level.

As an example, an immediate concern is the energy and latency costs of configuring or loading stored data in IMC. While IMC reduces the costs of MVM computation, it doesn't change the costs of loading data in the memory circuits. Thus, gains are derived only if MVM computation costs dominate at the system level. This depends on amortizing the data-loading costs through computational reuse. One way to analyze this is the widely used roof-line plot (Figure 12), where the breakpoint between loading-bound and compute-bound IMC operation occurs at the compute intensity (i.e., the number of compute operations performed on each of the loaded data), with the computation energy/throughput costs equaling the data-loading energy/throughput cost.

To illustrate, Figure 12 shows the example of loading data from off-chip DRAM. But it emphasizes the importance of evaluating IMC considering its bandwidth/energy tradeoffs together with different applications. Specifically, the IMC bandwidth/latency gains push the breakpoint to higher compute-intensity applications. However, as an example, the trend toward reducing operand precision in neural-network applications [2]–[4] can enable loading from smaller, more efficient embedded memories or fixed storage in IMC modules entirely, for specific (smaller) neural networks and use cases.

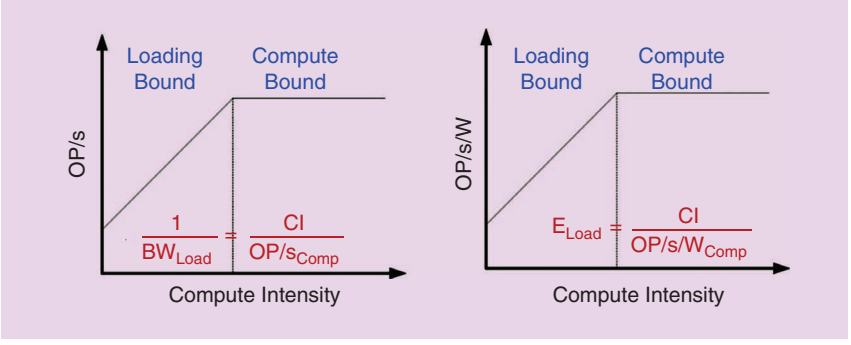


FIGURE 12: The roof-line plots identifying loading-bound and compute-bound regimes.

Prospects and Current State of the Art

While IMC presents a wide range of challenges, the initial promise it has shown and recent approaches that have been proposed to harness/overcome the underlying tradeoffs suggest it will be a vital area for ongoing research, especially toward platforms of practical scale and complexity. A few vectors for future research and their current states are reviewed next.

Emerging Memory Technologies

Emerging memory technologies represent a key vector for IMC research. The primary motivation is the potential for density scaling they present compared to SRAM. Indeed, increasing the scale of IMC based on resistive memory technologies (RRAM, MRAM, and PCM) has recently been demonstrated [8]–[10], [18], with even greater progress likely as foundry options for these technologies emerge. The primary challenge posed with regard to the underlying SNR tradeoff in IMC is readout of the computation result. In particular, the technologies present varying levels of resistance and resistance contrast, but they are generally much more limited than the on–off ratio or transconductance presented by MOSFETs in SRAM bit cells. The bit-cell computations thus possibly lead to lower signal values, potentially leading as well to a regime limited by readout complexity (energy and area) [9], which, in turn, scales in proportion to the number of bit cells involved. In this regime, the possible amortization of readout complexity is limited, and IMC gains are thus strongly determined by the characteristics of the bit cells themselves. Therefore, a critical direction for such research is the codesign of IMC architectures with bit-cell technology.

The readout complexity can have important implications on IMC density. Specifically, crossbar architectures tend to impose more stringent

readout challenges (e.g., to manage possible interference between readout columns), making the readout-circuit area significant and, again, scaling with array size. On the other hand, one-T, one-resistor (R) structures are typically denser than SRAM but now limited by MOSFET scaling. Further, the asymmetry in device characteristics between the T and R (memory device) often imposes additional area overheads to resolve (e.g., the use of two complementary bit cells).

An additional motivation for emerging resistive memory technologies is nonvolatility, which has the potential to lower power, especially in duty-cycled scenarios. This holds significant promise but will need to be evaluated in application use cases. Additionally, nonvolatility often limits the number of write cycles, posing challenges for hardware-virtualization approaches.

Algorithmic Codesign

Machine-learning inference has emerged as one of the biggest drivers for IMC, both because the computations have driven platforms to their energy and throughput limits and because the computations are dominated by MVM, which limits the gains possible from digital accelerators. Interestingly, machine-learning inference presents distinct opportunities for addressing the SNR tradeoff in IMC through algorithmic approaches. Machine-learning inference involves specifying a parametric model of how data statistically relate to inferences (decisions) of interest and then training the model parameters using available data that are representative of the statistics. In this way, statistical parameter optimization for a given model affords flexibility in the choice of actual model, which can be selected for computational efficiency. For example, this aspect has been exploited toward aggressive quantization [2], which has already been shown to yield benefits for IMC. But it can also be exploited to overcome computational noise arising from analog circuit nonidealities limiting IMC.

Instead of accessing raw bits row by row, IMC accesses a computation result over many/all bits, thereby amortizing the accessing costs.

Because the nonidealities can be statistical (e.g., variations) or deterministic (e.g., nonlinearity), a distinction can be made between hardware-specific training of model parameters and hardware-generalized training of model parameters. In hardware-specific training, the specific variation-affected instance of hardware is used in the training process to tune parameters to the specific hardware [7], [19]. This has the drawback of incurring instance-by-instance training costs, and recent IMC demonstrations have explored incorporating the associated hardware support to minimize such costs [19]. In hardware-generalized training, a statistical model of the distribution of variation-affected hardware is used in the training process to learn parameters one time [20]. This avoids the need for instance-by-instance training. For the example shown in Figure 13 of MRAM-based IMC implementing a neural network for vision classification (CIFAR-10), we see that this can overcome significant variation in the hardware (accuracy is maintained at several multiples of the actual MRAM-conductance variation).

Such algorithmic approaches show considerable promise, presenting a rich area for research. Specifically, a critical challenge that will need to be addressed is that operation relying on algorithmic codesign will inherently disrupt hierarchical abstraction based on the functional specification employed in architectural design today. Thus, new, possibly statistical approaches to composable architectural design and application mapping will likely be required.

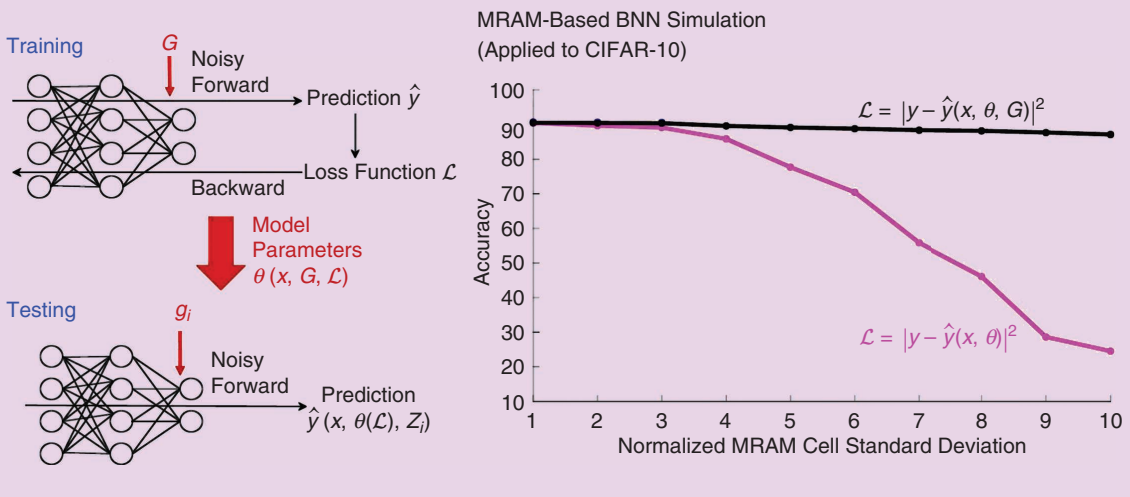


FIGURE 13: An illustration of an algorithm approach to overcome analog nonidealities [20]. BNN: binarized-neural-network.

Recent approaches that have been proposed to harness/overcome the underlying tradeoffs suggest it will be a vital area for ongoing research.

High-SNR Circuit Design

The SNR tradeoff underlying IMC ultimately limits its computation scale, integration in complex architectures, and programmability. Recent work has thus proposed circuit approaches for enhancing computation SNR. Figure 14(a) shows the bit-cell circuit used in [11], where a 1-b input-vector element combines with 1-b of stored data to yield a 1-b multiplication output, used to charge a local bit-cell capacitor rather than provide current on BL/BLb. Accumulation is then performed by switched-capacitor charge redistribution across all the equal-sized bit-cell capacitors in a column. Multiplication linearity is inherently ensured by binary bit-cell output, while accumulation error is primarily set by capacitor mismatch. Thanks to lithographic precision in modern, very-large-scale integration technologies, the metal-finger capacitors used exhibit excellent matching and temperature stability, with analysis suggesting that the column dimension can be scaled to thousands of rows before mismatch limits computation SNR. Furthermore, the metal-finger capacitors are laid out above the bit-cell Ts, occupying no additional area, such that the bit-cell area overhead with the required additional devices is just 80% compared to a standard 6T-SRAM bit cell (laid out using logic rules). Figure 14(b) shows the measured column transfer function used to implement the preactivation of a binarized neuron with 4,608 inputs (i.e., column dimension), with error bars showing the standard deviation over 512 on-chip neurons. The highly linear and stable computation achieved has enabled breakthrough scale for IMC of 2.4 Mb as well as demonstration of practical neural networks (e.g., 10 layers) with area-normalized throughput of 1.5 1-b TOP/s/mm^2 and energy efficiency of 866 1-b TOP/s/W (the highest known for neural-network accelerators).

Programmable IMC

Having substantially addressed the underlying SNR tradeoff and demonstrated increased scale, the capacitor-based approach further enables integration in complex heterogeneous architectures and programmability. Figure 15(a) shows the architectural block diagram of a silicon prototype in 65-nm CMOS, representing the current state of the art [12]. The architecture integrates a compute-in-memory unit (CIMU), with a CPU (RISC-V), direct-memory accessing (DMA) unit, scheduling hardware (timers), interfaces [universal asynchronous receiver-transmitter (UART) and general-purpose input/output (GPIO)], and 128 kB of standard program and data memory. The CIMU extends the IMC hardware from [11] in three key ways.

First, it introduces interfaces for efficiently converting data between the external 32-b architecture of the processor and the highly parallel bit-wise architecture of the IMC hardware. This enables maximum bandwidth utilization of the architectural resources (e.g., busses) for bringing data to/from the CIMU and enables its integration in the standard processor memory space. Such a tightly coupled IMC accelerator architecture is beneficial for enhancing and evaluating programmability.

Second, it introduces column-output ADCs and configurable near-memory computing (NMC), physically aligned with IMC outputs, for element-wise computations on MVM output vectors. The architecture is designed for embedded neural-network inference. Thus, the NMC supports operand scaling, offsetting, shifting, averaging, masking, nonlinear transformation, and so on for computations such as activation functions, batch normalization, and pooling. This enables the demonstration and evaluation of integration of IMC in heterogenous architectures.

Third, it enables bit-scalable IMC from 1 to 8 b via a bit-parallel/bit-serial (BPBS) scheme. Here, multiple bits of the matrix elements are mapped to parallel columns, while multiple bits of the input-vector elements are provided serially. Each column operation thus corresponds to MAC (vector inner product) between binary vector elements. Following ADC digitization, the column operations involved are then bit shifted to apply proper binary weighting and summed in the NMC to yield the multibit computation result. In this way, energy and area-normalized throughput scale linearly with the number of matrix-element bits and vector-element bits, yielding more favorable scaling than typically

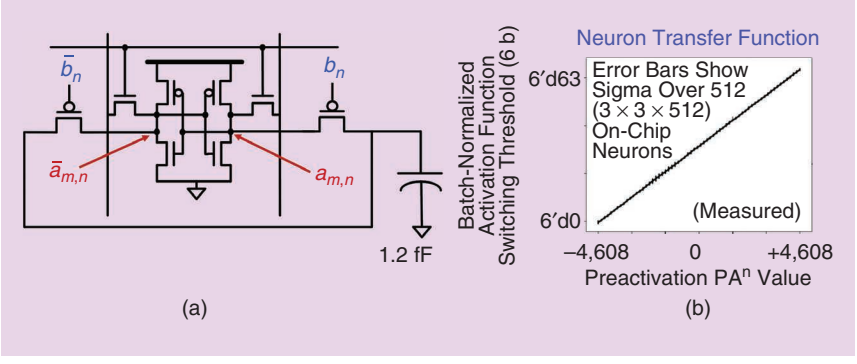


FIGURE 14: The high circuit SNR enabled by capacitor-based IMC [11]: the (a) bit-cell circuit and (b) measured column transfer function.

achieved with purely analog computation [21]. It should be noted that, with such computation, quantization is different than with that of standard integer compute. This is because the high column dimensionality (i.e., 2,304) leads to higher dynamic range than that supported by the ADC (i.e., an 8-b ADC chosen to balance energy and area overheads). This represents another form of throughput/energy tradeoff with SNR, in this case arising from computation (rather than communication) amortization. Nonetheless, for the quantization range of interest in neural-network applications (2–6 b), a signal-to-quantization noise ratio close to integer compute is achieved. Further, the deterministic nature of the computation enables robust modeling, which can be readily

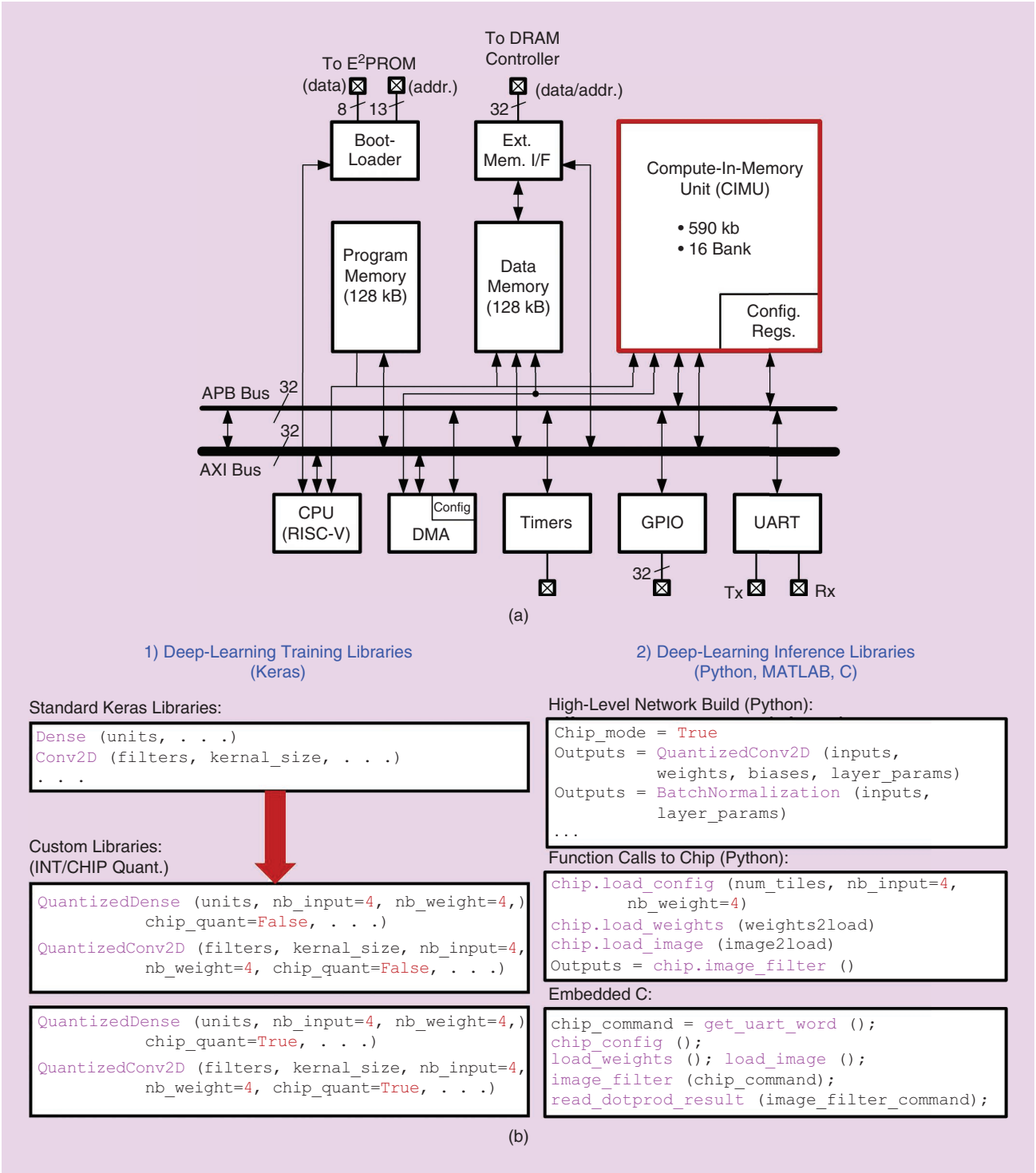


FIGURE 15: The programmable, bit-scalable IMC architecture: the (a) heterogeneous microprocessor architecture [12] and (b) software libraries for neural-network training and inference.

IMC has the potential to address a critical and foundational challenge affecting computing platforms today—that is, the high energy and delay costs of moving data and accessing data from memory.

incorporated in neural network training, as is done for standard quantized neural-network training.

To validate and exploit the CIMU's programmability, software libraries have been developed for neural-network training and inference, as shown in Figure 15(b). The training libraries are for domain-specific design frameworks (Keras and TensorFlow), providing a superclass for all supported neural-network layers. Arguments are offered for the activation and weight quantization level as well as a flag to switch between integer quantization and the BPBS quantization performed by the CIMU. This enables training for the quantization approach adopted as well as evaluation against integer quantization (with equivalent performance consistently observed thus far). The inference libraries are for two types of implementation. First, Python and MATLAB libraries are provided for implementing the top-level data and control flow on a host processor, making function calls to the chip (over USB) for selected compute-intensive operations (such as MVMs, activation functions, and batch normalization). Second, C libraries are provided for execution on the embedded CPU, for both receiving the function calls and implementing inference systems fully on the chip.

Conclusions and Summary

IMC has the potential to address a critical and foundational challenge affecting computing platforms today—that is, the high energy and delay costs of moving data and accessing data from memory. In doing so, it takes a disruptive approach to computing systems that requires integrative research across the computing stack. This has appropriately ignited a wide range of research. It is necessary to understand the foundational tradeoffs arising from IMC to better grasp the challenges that must be addressed and evaluate the different approaches being proposed. From here, perspectives may be derived regarding the potential gains IMC can offer, the contexts and conditions on which these depend, and the support they require across the compute stack.

IMC gives rise to a bandwidth/latency/energy-versus-SNR tradeoff, which has led to the achievements and limitations observed in recent prototypes. These prototypes have shown the potential for 10× gains simultaneously in energy efficiency and area-normalized throughput compared to fully optimized digital accelerators. But they have also been restricted in the scale and integration of IMC in the heterogeneous architectures required for practical computing systems. Recent work has begun to address these restrictions, with IMC demonstrations advancing to substantial scale, sophisticated and programmable architectures being developed, and software libraries being integrated into high-level application-design frameworks. These have transformed IMC into a practical technology for addressing foundational challenges in future computing systems, especially for energy- and throughput-constrained machine learning.

Acknowledgments

We thank Prof. N. Shanbhag (University of Illinois at Urbana–Champaign), Prof. P. Ramadge (Princeton University), E. Nestler, J. Yedidia, M. Tikekar, and P. Nadeau (Analog Devices Inc.) for extremely valuable collaboration and input.

References

[1] M. Horowitz, "Computing's energy problem (and what we can do about it)," in *Proc.* 2014 *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, pp. 10–14.

[2] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learning Res.*, vol. 18, no. 1, pp. 6869–6898, 2017. [Online]. Available: http://dl.acm.org/citation.cfm?id=3122009.3242044

[3] J. Choi, P. I.-J. Chuang, Z. Wang, S. Venkataramani, V. Srinivasan, and K. Gopalakrishnan, Bridging the accuracy gap for 2-bit quantized neural networks (QNN). 2018. [Online]. Available: http://arxiv.org/abs/1807.06964

[4] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *Proc. 32nd Int. Conf. Int. Conf. Machine Learning: Volume 37*, 2015, pp. 1737–1746.

[5] N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Computer Architecture*, 2017, pp. 1–12.

[6] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017. doi: 10.1109/JPROC.2017.2761740.

[7] J. Zhang, Z. Wang, and N. Verma, "In-memory computation of a machine-learning classifier in a standard 6T SRAM array," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 915–924, 2017. doi: 10.1109/JSSC.2016.2642198.

[8] X. Guo et al., "Fast, energy-efficient, robust, and reproducible mixed-signal neuromorphic classifier based on embedded NOR flash memory technology," in *Proc. 2017 IEEE Int. Electron Devices Meeting*, pp. 6.5.1–6.5.4.

[9] F. Su et al., "A 462GOPs/j RRAM-based nonvolatile intelligent processor for energy harvesting IoE system featuring nonvolatile logics and processing-in-memory," in *Proc. 2017 Symp. VLSI Technology*, pp. C260–C261.

[10] W.-H. Chen et al., "A 65nm 1Mb nonvolatile computing-in-memory ReRAM macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors," in *Proc. 2018 IEEE Int. Solid- State Circuits Conf.*, pp. 494–496.

[11] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-tile 2.4-Mb in-memory-computing CNN accelerator employing charge-domain compute," *IEEE J. Solid-State Circuits*, vol. 54, no. 6, pp. 1–11, 2019.

[12] H. Jia, Y. Tang, H. Valavi, J. Zhang, and N. Verma, A microprocessor implemented in 65nm CMOS with configurable and bit-scalable accelerator for programmable in-memory computing. 2018. [Online]. Available: http://arxiv.org/abs/1811.04047

[13] A. Biswas and A. P. Chandrakasan, "CONV-SRAM: An energy-efficient SRAM with in-memory dot-product computation for low-power convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 217–230, 2019. doi: 10.1109/JSSC.2018.2880918.

[14] J. Wang et al., "14.2 a compute SRAM with bit-serial integer/floating-point operations for programmable in-memory vector acceleration," in *Proc. 2019 IEEE Int. Solid-State Circuits Conf.*, pp. 224–226.

[15] Z. Jiang, S. Yin, M. Seok, and J.-S. Seo, "XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural

networks," in *Proc. 2018 IEEE Symp. VLSI Technology*, pp. 173–174.
[16] M. Kang, S. K. Gonugondla, A. Patil, and N. R. Shanbhag, "A multi-functional in-memory inference processor using a standard SRAM array," *IEEE J. Solid-State Circuits*, vol. 53, no. 2, pp. 642–655, 2018. doi: 10.1109/JSSC.2017.2782087.
[17] B. Fleischer et al., "A scalable multi-teraOPS deep learning processor core for AI training and inference," in *Proc. 2018 IEEE Symp. VLSI Circuits*, pp. 35–36.
[18] C.-X. Xue et al., "24.1 a 1Mb multibit ReRAM computing-in-memory macro with 14.6ns parallel MAC computing time for CNN based AI edge processors," in *Proc. 2019 IEEE Int. Solid-State Circuits Conf.*, pp. 388–390.
[19] S. K. Gonugondla, M. Kang, and N. Shanbhag, "A 42pJ/decision 3.12TOPS/W robust in-memory machine learning classifier with on-chip training," in *Proc. 2018 IEEE Int. Solid-State Circuits Conf.*, pp. 490–492.
[20] B. Zhang, L.-Y. Chen, and N. Verma, "Stochastic data-driven hardware resilience to efficiently train inference models for stochastic hardware implementations," in *Proc. 2019 IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 1388–1392.
[21] R. Sarpeshkar, "Analog versus digital: Extrapolating from electronics to neurobiology," *Neural Comput.*, vol. 10, no. 7, pp. 1601–1638, 1998. doi: 10.1162/089976698300017052.

About the Authors

Naveen Verma (nverma@princeton.edu) received his B.A.Sc. degree from the University of British Columbia, Vancouver, Canada, in 2003 and his M.Sc. and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, in 2005 and 2009, respectively. He is a professor in the Department of Electrical Engineering at Princeton University, New Jersey, where his research focuses on systems for intelligent sensing, including sensors based on large-area electronics, algorithms for machine perception and control employing machine learning, and heterogeneous computing platforms exploiting in-sensor and in-memory computing. He is a Member of the IEEE.

Hongyang Jia (hjia@princeton.edu) received his B.Eng. degree in microelectronics from Tsinghua University, Beijing, in 2014, and his M.A. degree in electrical engineering from Princeton University, New Jersey, in 2016, where he is currently pursuing his Ph.D. degree. He is with the Department of Electrical Engineering at Princeton, where his research focuses on ultralow-energy system design for inference applications. His research interests include programmable in-memory computing platforms and CMOS IC design leveraging approximate computing. He received the Analog Devices Outstanding Student Designer Award in 2017. He is a Student Member of the IEEE.

Hossein Valavi (hvalavi@princeton.edu) received his B.Sc. degree in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 2013 and his M.A. degree in electrical engineering from Princeton University, New Jersey, in 2015, where he is currently pursuing his Ph.D. degree. His research focuses on ultralow-energy system design for signal processing and machine-learning applications. He was a recipient of the Analog Devices Outstanding Student Designer Award in 2016. He is a Student Member of the IEEE.

Yinqi Tang (yinqit@princeton.edu) received his B.S. degree in microelectronics from Fudan University, Shanghai, China, in 2014 and his M.A. degree in electrical engineering from Princeton University, New Jersey, in 2016, where he is currently pursuing his Ph.D. degree in the Department of Electrical Engineering. His research interests include energy-efficient hardware systems for machine-learning and deep-learning applications, focusing on both algorithmic and hardware design. He is a Student Member of the IEEE.

Murat Ozatay (mozatay@princeton.edu) received his B.Sc. degree in electrical and electronics engineering from the Middle East Technical University, Ankara, Turkey, in 2015 and his M.A. degree in electrical engineering from Princeton University, New Jersey, in 2017, where he is currently pursuing his Ph.D. degree in the Department of Electrical Engineering. His research focuses on bringing together algorithms and insights for learning with technologies and systems for advanced sensing. His other research interests include machine learning, artificial intelligence, the Internet of Things, and the design of very-large-scale integration systems. He is a Student Member of the IEEE.

Lung-Yen Chen (lungyenc@princeton.edu) received his B.S. degree in electrical engineering from National Taiwan University, Taipei, in 2011 and his M.A. degree in electrical engineering from Princeton University, New Jersey, in 2014, where he is currently pursuing his Ph.D. degree in the Department of Electrical Engineering. His research interests include digital architectures and circuit design for in-memory computing and multimedia applications, with emerging technologies such as magnetic random-access memory and 3D integration. He is a Student Member of the IEEE.

Bonan Zhang (bonanz@princeton.edu) received his B.Eng. degree in electrical and computer engineering from McGill University, Montréal, in 2017 and his M.A. degree in electrical engineering from Princeton University, New Jersey, in 2019, where he is currently pursuing his Ph.D. degree in the Department of Electrical Engineering. His research interests include IC design for in-memory computing architecture with emerging technologies, such as magnetic random-access memory, and exploring algorithmic approaches for hardware relaxation to enable machine-learning applications. He received the Analog Devices Outstanding Student Designer Award in 2019. He is a Student Member of the IEEE.

Peter Deaville (deaville@princeton.edu) received his B.S. degree in electrical engineering from the University of Maryland, College Park, in 2018. He joined Princeton University, New Jersey, in the spring of 2019 as a Ph.D. student in the Department of Electrical Engineering, where his research focuses on exploring in-memory computing using magnetic random-access memory technology. His other research interests include circuit design for machine learning, leveraging emerging methods and technologies. He is a Student Member of the IEEE.

SSC