Theme Article: Data-Centric Computing

Let Analog Be Analog: Principles for Designing Analog Accelerators

Ben Feinberg, *Sandia National Laboratories, Albuquerque, NM, 87185, USA*
T. Patrick Xiao, *Sandia National Laboratories, Albuquerque, NM, 87185, USA*
Christopher H. Bennett, *Sandia National Laboratories, Albuquerque, NM, 87185, USA*
Sapan Agarwal, *Sandia National Laboratories, Livermore, CA, 94551, USA*

Abstract—Accelerators that take advantage of analog circuits for in-memory processing have become a major research topic in recent years. These systems have the potential to provide performance and efficiency beyond what is achievable from conventional digital systems. However, the fundamental differences between analog and digital computation means that using digital abstractions to design analog architectures often leads to systems that are not robust to uniquely analog sources of error. This paper discusses the pitfalls of popular design choices for analog systems that are based on digital abstractions. To replace these misused abstractions, we suggest a set of guiding principles that better leverage the properties of analog systems to jointly improve system accuracy and energy efficiency.

Analog computing, which directly leverages physical laws to process continuous-valued analog signals, is a more primitive and energy-efficient form of computing than digital computing. This has attracted a resurgence in interest in analog processors, particularly for machine learning applications. However, a return to analog computing is not without its perils. The myriad sources of noise and errors that bedevil analog systems are one of the main reasons for the dominance of digital computation today. Therefore, in designing analog processors for a new age, it is tempting to apply lessons that have been learned from decades of designing digital systems.

Abstractions that are valuable for digital design are often ineffective or even counter-productive when applied to the analog domain, leading to reduced energy efficiency and unacceptable accuracy. In this paper, we highlight several prominent examples of such pitfalls, specifically for analog processing-using-memory accelerators. Rather than rely on digital abstractions, we encourage system designers to *let analog be analog*: capitalize on the unique properties of analog devices and circuits, rather than fight them with extensive digital processing.

This paper focuses on analog accelerators for matrix vector multiplication (MVM), which compute inside arrays of programmable memory devices (Figure 1). These accelerators have been widely explored for low-power deep neural network (DNN) inference [1], but have broader applications in linear algebra. The memory device states are programmed to represent the matrix elements, while applied voltages to the rows encode the elements of the input vector.[1] After the voltages are applied, the dynamics of the analog circuit perform the fully parallel computation of an MVM. In a current-domain MVM, each memory cell draws a current that is the product of a programmed conductance and an applied voltage, and these currents are summed by Kirchhoff's law along each column. The resulting currents represent the dot products, which are then converted to digital values by analog-to-digital converters (ADCs).

These analog-domain multiplication and summation operations are performed on continuous-valued signal and noise components. Therefore, both the signal and noise grow with the number of terms in the sum. Furthermore, the multiplications and summations are not guaranteed to be perfectly linear.

Digital Object Identifier 10.1109/XXX.0000.0000000

[1] In this paper, we use rows to refer to the input dimension of the MVM array and columns to refer to the output dimension.

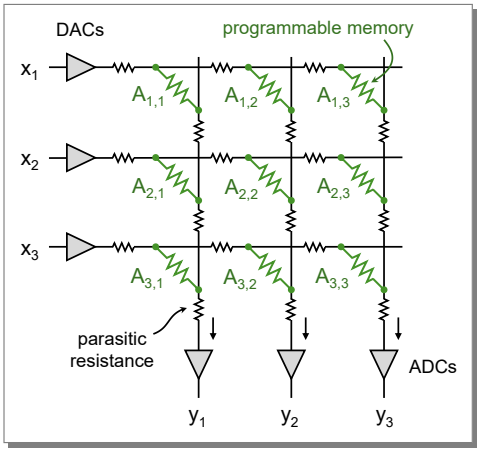


FIGURE 1. Current-domain analog MVM operation for computing $\mathbf{Ax} \rightarrow \mathbf{y}$.

In this paper, we focus on two major sources of error in the analog domain: random conductance errors in the programmable memory devices, and parasitic IR drops caused by the resistance of the array interconnects. Although these are just two sources of analog errors, they allow us to demonstrate important observations about analog accelerators. Random conductance errors are representative of a broad range of error characteristics including the precision of device programming, device noise, temporal drift, and process variation. Parasitic IR drops result from current flow across the non-zero resistance of row and column interconnects, reducing the applied voltage across memory devices. These drops accumulate along the rows and columns, leading to errors that are spatially non-uniform and coupled between different devices.

LET ANALOG BE ANALOG

In a digital system, one can safely assume that a function will deterministically produce the correct result if the operands are correct. This allows digital designers to choose low-level implementations and data representations to optimize performance without compromising correctness. For analog systems, it is generally unrealistic to assume functional equivalence between hardware design choices. In the analog domain, functions are not exact implementations, and are sensitive to small non-idealities in their specific circuit implementations. Moreover, must be treated as random variables, with distributions depending on data mapping to analog quantities and can change over time. These complex errors imply that in general, assessing whether an analog system produces accurate results requires physical implementations or realistic simulations.

These mismatches in the computational model mean that arguments to design analog processors based on computational equivalence with digital systems may not hold. We demonstrate these pitfalls by studying three design choices in the analog accelerator literature.

We use CrossSim [2] to evaluate the precision of analog MVMs. To represent a realistic workload, we use the weight matrix for layer 44 of ResNet-50 with input vectors from ImageNet classification. In the following results, we use two generic forms of device conductance error: state-independent and state-proportional error, following our previous work [3]. The conductance error from target, $\Delta G = G - G_{\text{target}}$, is modeled per device as:

$$\Delta G_{\text{ind}} = \alpha \times \mathcal{N}(0,1) \times (G_{\text{max}} - G_{\text{min}}) \quad (1)$$

$$\Delta G_{\text{prop}} = \alpha \times \mathcal{N}(0,1) \times G_{\text{target}} \quad (2)$$

where G_{min} and G_{max} define the accessible conductance range and α is the error magnitude.

Representing Negative Matrix Values

A key design decision for an analog MVM accelerator is how to represent negative matrix values, as programmable memory cells cannot have negative conductance. A natural approach is a differential pair of cells passing current in opposite directions, where the conductance difference represents a signed value. Typically, one cell in each pair is programmed to 0 and the other to the target value; for example -7 would be represented as zero on the positive device and 7 on the negative device. Another approach is *offset subtraction*, which adds a fixed offset to each value so that it can be mapped to a single positive conductance; this offset is then subtracted from the MVM result [4]. From a digital perspective, this transformation is valid, and is directly analogous to the offset binary representation for integers and the exponent bias in floating-point numbers.

However, as shown in Figure 2, these two approaches produce dramatically different levels of error in the analog MVMs when realistic analog hardware properties are considered. We simulate an array with 576 rows, partitioning the 4608×512 weight matrix across eight arrays. Error is evaluated on the partial dot products from each array.

In Figure 2, the error is split into two regimes. At low conductance, it is dominated by conductance programming errors. At high conductance, parasitic

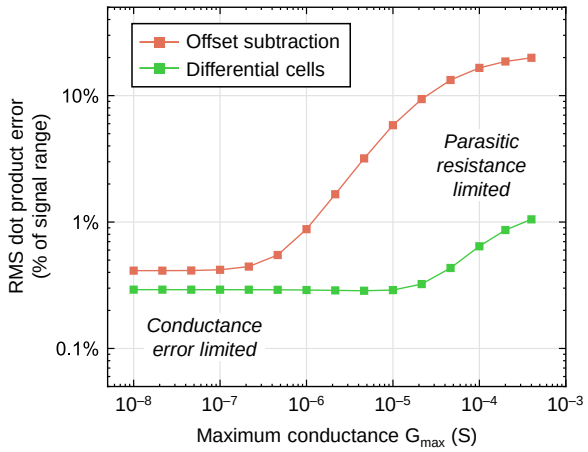


FIGURE 2. MVM error as a function of cell conductance with differential and offset mapping for negative numbers using cells with an on-off ratio (G_{max}/G_{min}) of 20, a 1% state-independent conductance error, and a 0.35Ω parasitic wire resistance between the cells along both the rows and columns of the array [5]. To evaluate the general case where the input elements can also be negative, we randomly flipped the sign of the input elements to layer 44 of ResNet-50, which are otherwise non-negative due to ReLU activations. The subtraction of the offset is assumed to be done in digital.

IR drops dominate, and the error increases rapidly with conductance due to increasing summed currents. Though both regimes are present using either scheme, the transition point differs substantially. With offset subtraction, the transition is at $G_{max} \approx 0.5\,\mu S$: practically all existing memory devices would be parasitics-limited. When using differential cells, the transition is at $G_{max} \approx 50\,\mu S$: with this 100× higher threshold, many published memory devices would not be parasitics-limited.

This difference arises from how the two schemes affect the programmed device conductances. When using differential cells, half of the cells—corresponding to the unused sign of each element—will always be programmed to G_{min}. Of the remaining cells, the mean conductance will be the mean absolute value of the matrix. By contrast, when using offset subtraction, assuming zero-centered matrix values, the mean cell conductance is at the midpoint of the conductance range. This higher per-element conductance leads to higher column currents, inducing larger IR drops for the same device technology. Without changing the negative number representation, the only way to mitigate this error is to use a smaller array, which heavily reduces energy efficiency, as discussed in the next section. The remaining results in this paper will therefore use differential cells for negative matrix elements.

Sizing ADCs for Digital Precision

To avoid quantization errors from the finite precision of ADC circuits, many papers use what we refer to as the *full-precision guarantee* (FPG) to size ADCs relative to other system parameters. It assumes a digital model of an MVM, and sets the ADC resolution to:

$$B_{out} = \begin{cases} B_W + B_{in} + log_2 N & \text{if } B_W > 1, B_{in} > 1 \\ B_W + B_{in} + log_2 N - 1 & \text{otherwise} \end{cases} \quad (3)$$

where B_W is the bits per cell, B_{in} is the digital-to-analog converter resolution, and N is the number of accumulated values in analog. This equation computes the minimum number of bits required to uniquely map each possible digital result to a different ADC output. When the ADC resolution falls in the 6–10-bit range that is common for analog MVM accelerators, the FPG requires small arrays and/or aggressive bit slicing. Both choices increases the number of ADC operations needed to compute the full MVM, penalizing the latency and energy efficiency.

The FPG fundamentally mismatches to the behavior of realistic analog systems. First, circuit parasitics and non-linearities break the assumption of a linear MVM. Even ignoring such effects, the FPG imposes a digital understanding of error accumulation. The discrete nature of digital arithmetic ensures that if each multiplication is computed correctly, then the sum of these products will be deterministically correct. However, analog signals are not discretized; small random errors that may not cause any individual multiplication to be incorrect can still accumulate to a large dot product error. When the expected error in the analog dot product exceeds half the ADC level spacing, the core assumption of the FPG breaks down. A set of inputs that should result in a specific digital result can now produce multiple different ADC outputs.

Ensuring the FPG's validity requires analog system designers to work within a tightly constrained design space. Figure 3 shows how the RMS error of analog dot products grows with ADC resolution, when the level spacing is set by the FPG. The error is expressed in units of ADC least significant bits (LSBs), and each additional ADC bit implies a doubling of N since the LSB is set by the minimum non-zero current through an ideal cell. When computing MVMs on dense inputs (50% input sparsity), the dot product error exceeds 0.5 LSB after accumulating just 16 values. In this case, even relatively small arrays (e.g. 128×128) are expected to have errors on multiple outputs per MVM.

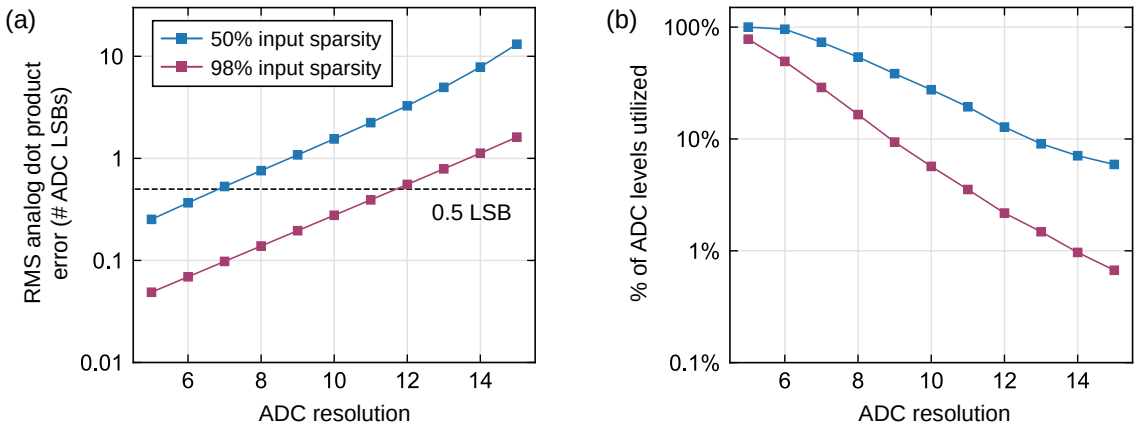


FIGURE 3. Effects of ADC sizing using the FPG on (a) MVM-level precision; (b) ADC range utilization. RMS precision and utilization are evaluated for 1000 MVMs between a 4096 $\times$ 512 matrix and a 1 $\times$ 4096 input vector for a system with 2 bits per cell (3 bits per differential pair), 1-bit input slices, and 4% state-independent error. The error is expressed in units of ADC (LSBs), which is equivalent to the number of level spacings.

For sparse inputs—corresponding to deep DNN layers or higher-significance input slices—the error can stay under 0.5 LSB up to large array sizes. However, as shown in Figure 3(b), only a small fraction of the ADC's levels are utilized in this case. A lower-resolution ADC than prescribed by (3) could provide the same accuracy for these layers. This was demonstrated by RAELLA, which used FPG-based level spacing but a lower ADC resolution [6]

Therefore, the FPG imposes significant unnecessary restrictions on analog system designers while failing to truly simplify the design process—because detailed accuracy simulations are needed to determine its validity. An alternative guiding principle to the FPG is to design the ADC level spacing to quantize the incoming signals with minimum distortion [3]. The minimization is done across the statistical distribution of signals that are quantized by the ADC, and thus depends on the target workload. This approach is broadly generalizable; however, it requires accurate characterization of the signals distribution seen by the ADC.

Digital Checksums for Analog MVMs

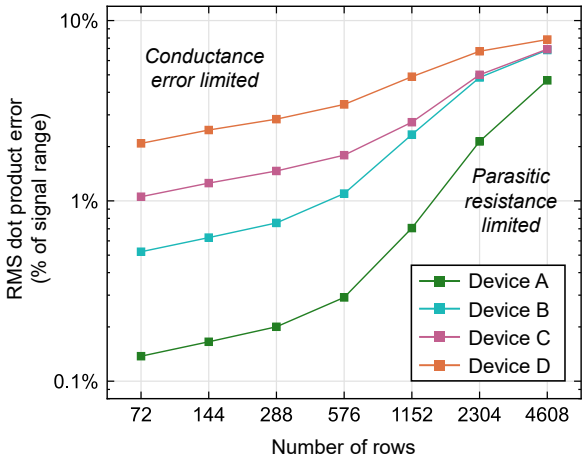
Several recent papers have proposed error correction and detection schemes for analog MVMs based on variations of digital checksums [7]. Their usefulness, however, is limited by the mismatch between analog and digital error models. In digital systems, errors are rare and frequently cause large changes; for instance, a flipped bit in a data word. By contrast, in analog systems, errors relative to the correct digital value are smaller but more prevalent, leading to a poor fit with digital-inspired correction techniques. For example, when the analog dot product has an RMS error of 0.5 LSBs, the probability of error in the ADC output is 32%. Even when the RMS analog error is 0.25 LSBs, 4.5% of the ADC outputs will be in error.

Prior work on analog error correction often focuses on stuck-at-faults—where a cell is permanently stuck in one state—as sources of large isolated cell errors resembling digital faults. However, they often ignore the much more common case of small conductance errors accumulated over many cells. This implicitly assumes that the non-faulty devices are unrealistically precise. Though stuck-at-faults are important, focusing purely on them heavily restricts the applicability of the proposed schemes.

Error detection and correction for analog systems is nonetheless an important research topic. Architectural mitigations for analog errors such as drift and IR drops can be just as important as device- or circuit-level error reductions, and the field is ripe for novel solutions. These schemes should focus on the unique dynamics of analog hardware, rather than imposing digital assumptions on analog errors.

PRINCIPLES FOR ANALOG ACCELERATORS

Abstractions are valuable for reasoning about and designing complex systems. Despite its pitfalls, the FPG coupled several hardware parameters constraining the design space of analog accelerators. To replace digital-



(a)

Device	G_{max}	G_{min}	Cell error
A	10 µS	10 pS	State-proportional, α = 5%
B	25 µS	0.5 µS	State-proportional, α = 15%
C	25 µS	0.5 µS	State-independent, α = 2%
D	50 µS	2.5 µS	State-independent, α = 4%

(b)

FIGURE 4. Effects of array size on (a) MVM error, with (b) four example devices. Error is evaluated on 100 convolution operations using inputs to ResNet-50 layer 44, sampled from the ImageNet test set. These convolutions produce 10M to 642M partial dot products, depending on array size. Errors are normalized to a signal range for each array size that contains 99.99% of the true partial dot products.

inspired abstractions, we present three analog-inspired principles for the design of analog MVM accelerators.

These principles are based on the observation that the primary energy bottleneck in analog systems is converting raw analog quantities from the array into digital output[2]. RAELLA formalized this as the *Titanium Law* of analog systems [6], defining ADC energy per DNN inference as the product of ADC operation energy, ADC operations per MAC, MACs per DNN, and an array utilization factor [6]. All of these terms can be influenced by architectural design, but we focus on reducing energy by maximizing the number of MACs per ADC operation. Importantly, this needs to be done while meeting application-level accuracy targets, without necessarily achieving identical dot products as digital systems.

Use Large Arrays

At maximum, an array with N rows can perform N MACs per ADC operation. By the Titanium Law, increasing N produces a linear increase in energy efficiency, assuming full array utilization. Therefore, arrays should be made as large as possible, and devices should be engineered to support MVMs that fully leverage large arrays.

However, the size of the array is not unbounded. Summing more analog values per column results in a wider distribution of dot product errors, potentially reducing application accuracy. Another major limiter of array size is parasitic IR drops, which contribute errors that increase super-linearly with array size [3]. To understand the impact of these effects, Figure 4a shows the MVM-level accuracy for the same layer as the previous examples, again assuming a parasitic resistance of 0.35Ω between cells. How errors grow with array size depends on the memory device properties; therefore, we analyze these trends across the four different devices in Figure 4b with properties roughly modeled after existing memory technologies.

Figure 4a shows that for every device, there are two distinct error regimes with array size. For small arrays with relatively small summed currents, the error is limited by conductance errors. Here, Devices A and B with state-proportional error have lower MVM-level errors than Devices C and D with state-independent error. This is because the distribution of weight values in a typical DNN matrix—and hence the resulting conductance values—tend to be concentrated near zero, so state-proportional error properties lead to smaller conductance errors on average. For larger arrays, the error is limited by parasitic IR drops due to the larger accumulated currents. Here, the differences in error between the devices are driven by their conductance ranges. Device A has the least error in this regime due to its lower conductance and high On/Off ratio, while

[2]We refer to this as an ADC operation, though it may involve the use of other peripheral circuits such as transimpedance amplifiers.

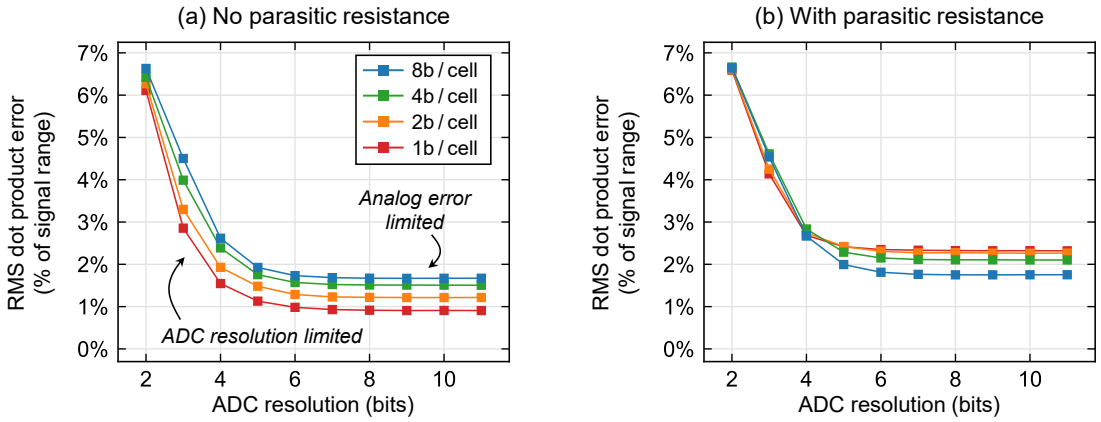


FIGURE 5. Effect of ADC resolution on MVM accuracy using Device C with different levels of weight bit slicing, (a) without parasitic resistance; (b) with parasitic resistance. The per-slice MVM results are separately digitized using an ADC range per slice based on the 99.99th percentile of the analog partial dot products from a profiling simulation [6] then accumulated. Error is evaluated on partial dot products after accumulating weight slices but before aggregating matrix partitions.

Devices B and C have convergent errors at large array size due to their identical conductance ranges.

A major lesson from Figure 4a is that to enable the efficiency of large arrays, memory device researchers must develop devices with low conductances and high On/Off ratios. For architects, new techniques to reduce or compensate for parasitic IR drops can improve system efficiency and accuracy. These techniques could include improved array topologies, modifying the programmed conductance targets, digital corrections, or exploring other system tradeoffs. Notably, in Figure 4a, the devices were assumed to behave as linear resistors. However, the effect of IR drops also depends on the I-V nonlinearity of the memory device, and practical compensation schemes must consider this complicated interplay.

Avoid Weight Slicing

Bit slicing splits the bits of a full-precision value across multiple programmed conductances (weights) or applied voltages (inputs). Bit slicing rapidly incurs energy overheads, multiplicatively increasing the number of ADC operations per MAC by the product of the number of input slices and weight slices. Such overheads might be tolerable if slicing-based optimization like the FPG provided digital levels of accuracy; however, as shown in Figure 3 this is not the case. This raises a new question: does bit slicing provide benefits without the FPG?

Figure 5 shows how weight slicing affects MVM errors, with the same simulation configuration as Figure 4a using only Device C and 576 rows. Unlike prior simulations, we evaluate the digitized dot products after the ADC. We designed the ADC range to minimize signal distortion, similar to prior work [8], using a separate set of inputs for calibration. With increasing bit resolution, the range remains fixed while level spacing is decreased. This approach makes the ADC's effect on accuracy intuitive. At low resolution, the ADC's quantization error dominates the MVM error. Beyond $\sim$6 bits, the level spacing is below the level of analog noise and finer ADC resolution provides no further benefit.

Without including the effect of parasitic IR drops (Figure 5(a)), finer weight bit slices benefits MVM precision. This stems not from the increased conductance level spacing within a slice, but from how errors in different slices cancel when summed [9]. For 6- to 10-bit ADCs, the gap between unsliced MVMs (8 bits/cell) and fully sliced MVMs (1 bit/cell) amounts to an RMS error improvement of 0.76% of the signal range. This modest improvement comes at the cost of 8$\times$ more ADC operations.

The comparison worsens for bit slicing when considering realistic parasitic resistance effects (Figure 5(b)). The order of the curves is inverted, and the unsliced case now has the lowest error due to differences in the conductance distributions. This is because bit slicing causes more cells to be mapped into higher conductance portions of the value range, leading to larger IR drops. For example, when the value 7 (0x00000111) is mapped using 2-bit slices, the least-significant slice will be near the max cell conductance (0x11) and the next slice will be $\frac{1}{3}$ of the maximum cell

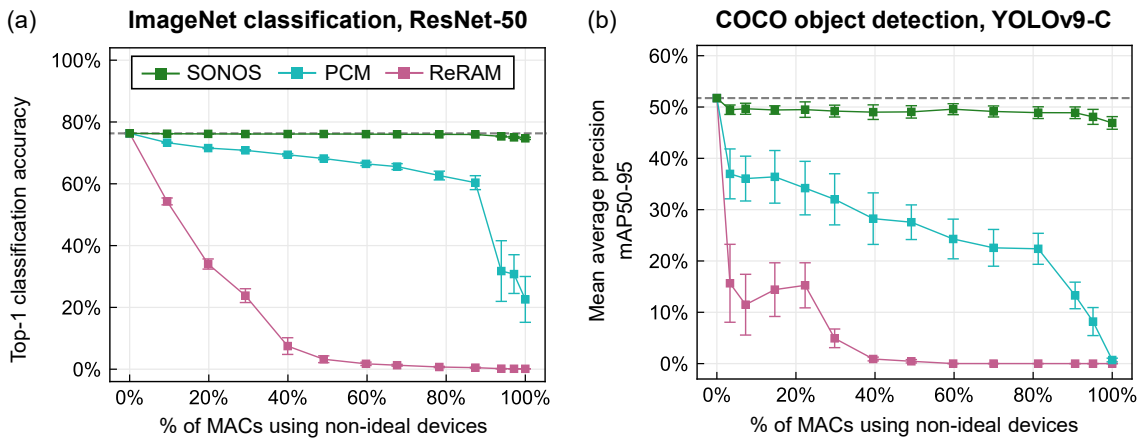


FIGURE 6. Accuracy of computer vision CNNs using direct weight transfer without retraining with an increasing number of MACs computed using analog MVMs: (a) ResNet-50 on ImageNet classification, and (b) YOLOv9-C on COCO object detection. Conductance errors corresponding to the characteristics of SONOS [10], PCM [8], and ReRAM [11] are applied starting from the last layer backward, whole layers at a time. Error bars show the standard deviation over 20 runs. The accuracy of the two CNNs with ideal devices in all layers is 76.3% (ResNet-50; 50,000 test images) and 51.8% (YOLOv9-C; 5000 validation images), respectively. Even with ideal devices, the simulations included 8-bit quantized inputs, 8-bit ADCs, and arrays with 1152 rows.

conductance. By contrast, the unsliced cell would be programmed to $\frac{7}{255}$, significantly smaller than either of the two slice conductances.

These results provide a strong justification against weight bit slicing where it can avoided. Although bit slicing can reduce errors in some cases, the reduction is unlikely to justify the substantial increase in ADC operations. In some situations, bit slicing is necessary, such as inherently binary memory or applications that require more precision than available in the conductance targets. Future work should explore how to optimally use bit slicing in these cases.

Accumulate in Analog

The motivation for input slicing is different from that for weight slicing. For many types of memory cells, accurate MVMs are possible only with binary voltage inputs due to the nonlinear current-voltage relationship of the cell, which causes the conductance to depend on voltage. With conventional digital accumulation, an ADC operation is performed for each (input slice)$\times$(weight slice) combination, and these results are combined using digital shift-and-add operations. Analog accumulation means that these shift-and-add operations are partially or fully performed directly on the analog signals, thereby requiring fewer ADC operations per MVM and providing substantial efficiency gains.

However, analog accumulation introduces new challenges. Analog accumulation requires multiple new analog components such as capacitors—subject to process variation—and amplifiers consuming additional energy and with nonlinear responses and other non-idealities [12]. Therefore, analog accumulation requires careful management of additional analog error sources and power-consuming components that can cut into efficiency gains. We emphasize analog accumulation as it follows directly from our philosophy of *let analog be analog*, but unlike the previous two principles, its implementation is a relatively under-explored research topic. Future work that fully demonstrates these potential gains will significantly impact on the efficiency of analog systems.

HARDWARE/SOFTWARE CODESIGN FOR ANALOG SYSTEMS

The properties of an algorithmic workload strongly influence its ability to efficiently utilize analog arrays and its sensitivity to different sources of analog error. These dependences are critical for informing researchers about desirable hardware properties and what types of algorithms that map effectively to analog systems. In this section, we move beyond MVM-level errors and focus on DNN inference workloads as an exemplar.

We have argued that large arrays are essential for maximizing MVM efficiency. But to maintain high system-level efficiency, the workload should have many large matrices that can fully utilize these arrays.Notably, grouped or depthwise convolutions—found in DNNs like MobileNet, and ShuffleNet—tend

to map sub-optimally to analog MVMs because they have relatively few terms per dot product, and therefore fewer MACs per ADC operation. Similarly, sparse matrices like those found in neural networks with unstructured pruning also significantly reduce system efficiency. Large fully-connected layers and ungrouped convolutions with large kernels and many input channels tend to have the best array utilization.

Furthermore, the emergent statistical properties shared by many machine learning workloads allow them to better tolerate some types of analog errors better than others. The distribution of weight and input values for any given neural network layer is often concentrated near zero [3]. Therefore, precision is most important for low conductance states, which encode the vast majority of weights.

This introduces a preference for memory devices with certain error vs. conductance profiles, as illustrated in Figure 6. Here, inference accuracy is compared between three published devices that have each been demonstrated in MVM accelerators with an array size of at least 256×256. Layers were gradually converted from using ideal devices to realistic devices with random conductance errors, starting from the last layer of the network.[3] This shows the increasing impact of analog error propagation through the layers of these DNNs.

What explains the gaps in accuracy between these devices? SONOS memory, in its subthreshold regime, has both a high On/Off ratio and programming errors that scale linearly with conductance—i.e. it has true state-proportional error, similar to Device A. Therefore, it has vanishingly small error for the most frequently used weights in these DNNs, thereby maintaining high end-to-end accuracy on both tasks. Meanwhile, PCM has errors that increase with conductance, but the errors do not asymptotically approach zero at low conductance, resulting in higher expected errors compared to SONOS. The ReRAM device has broadly similar error characteristics to PCM, but the errors are larger than those for both PCM and SONOS over its entire dynamic range.

The presence of sharp discontinuities in the accuracy of both DNNs in Figure 6 implies that some layers are more sensitive to errors and act as bottlenecks for accuracy. This suggests that analog error mitigation techniques in these particular layers might yield disproportionately large benefits.

To support high inference accuracy using analog hardware with larger intrinsic errors, recent works have explored hardware-aware training of ML models [5]. In these approaches, analog errors are emulated during DNN training to produce weights that can better tolerate these errors during inference. These methods can fully or partially recover accuracy that would be lost with a direct weight transfer approach, but success depends on both the severity of errors and the complexity of the task.

Beyond neural networks, tailoring applications to the properties of an analog system may be beneficial. For instance, some applications may allow modifying the algorithm so that the programmed matrix weight distribution better matches the error properties of a given cell. Going a step further, since analog MVM accelerators incur substantial costs from high-precision ADC operations, designing applications to use lower precision outputs than weights can help match the relative energy costs. This further demonstrate the potential of “letting analog be analog": developing optimizations to uniquely leverage the inherent properties of analog systems, rather than mapping applications that were designed and optimized for digital hardware onto analog systems.

BEYOND ACCURACY AND EFFICIENCY

We have focused solely on accuracy and efficiency. However, these are only the first obstacles for analog MVM accelerators. Avoiding the FPG and leveraging analog-oriented principles to reduce ADC conversion costs may reveal other inefficiencies. Although analog arrays can process MACs more efficiently than digital, the rest of the system—including memories and interconnection networks—remains at digital levels of efficiency. The resulting system may therefore be bottlenecked by conventional digital operations and data movement. Weight-stationary mappings can exacerbate data movement energy costs.

Another important research direction for analog MVM accelerators is programming models and abstractions. While we argued against digital abstractions from a system design and analysis perspective, programmer-friendly abstractions will be critical for the broad adoption of analog accelerators. Analog systems expose significant additional complexity to programmers, as datatype selection can affect accuracy and efficiency. Frameworks that can optimize data representations through *a priori* analysis or profiling can help programmers manage these complexities. Effective solutions to both of these issues will be essential for the eventual success of analog MVM accelerators

[3]These simulations do not include the effect of parasitic IR drops due to the high simulation cost of exact circuit solves.

and create ample opportunities for researchers across the computing stack.

ACKNOWLEDGMENTS

This article has been authored by an employee of National Technology & Engineering Solutions of Sandia, LLC under Contract No. DE-NA0003525 with the U.S. Department of Energy (DOE). The employee owns all right, title and interest in and to the article and is solely responsible for its contents. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this article or allow others to do so, for United States Government purposes. The DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan https://www.energy.gov/downloads/doe-public-access-plan.

REFERENCES

1. T. P. Xiao, C. H. Bennett, B. Feinberg, S. Agarwal, and M. J. Marinella, "Analog architectures for neural network acceleration based on non-volatile memory," *Appl. Phys. Rev.*, vol. 7, no. 3, p. 031301, July 2020.
2. B. Feinberg, T. P. Xiao, C. J. Brinker, C. H. Bennett, M. J. Marinella, and S. Agarwal, "CrossSim: accuracy simulation of analog in-memory computing." [Online]. Available: https://github.com/sandialabs/cross-sim
3. T. P. Xiao, B. Feinberg, C. H. Bennett, V. Prabhakar, P. Saxena, V. Agrawal *et al.*, "On the accuracy of analog neural network inference accelerators," *IEEE Circuits Syst. Mag.*, vol. 22, no. 4, pp. 26–48, 2022.
4. A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu *et al.*, "ISAAC: a convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *Intl. Symp. on Comput. Archit. (ISCA)*, June 2016, p. 14–26.
5. M. J. Rasch, C. Mackin, M. Le Gallo, A. Chen, A. Fasoli, F. Odermatt *et al.*, "Hardware-aware training for large-scale and diverse deep learning inference workloads using in-memory computing-based accelerators," *Nat. Commun.*, vol. 14, no. 1, p. 5282, Aug 2023.
6. T. Andrulis, J. S. Emer, and V. Sze, "RAELLA: Reforming the arithmetic for efficient, low-resolution, and low-loss analog PIM: No retraining required!" in *Intl. Symp. on Comput. Archit. (ISCA)*, June 2023.
7. M. Liu, L. Xia, Y. Wang, and K. Chakrabarty, "Fault tolerance for RRAM-based matrix operations," in *IEEE Intl. Test Conf. (ITC)*, 2018, pp. 1–10.
8. V. Joshi, M. Le Gallo, S. Haefeli, I. Boybat, S. R. Nandakumar, C. Piveteau *et al.*, "Accurate deep neural network inference using computational phase-change memory," *Nat. Commun.*, vol. 11, no. 1, p. 2473, May 2020.
9. R. Genov and G. Cauwenberghs, "Charge-mode parallel architecture for vector-matrix multiplication," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 48, no. 10, pp. 930–936, 2001.
10. V. Agrawal, T. P. Xiao, C. H. Bennett, B. Feinberg, S. Shetty, K. Ramkumar *et al.*, "Subthreshold operation of SONOS analog memory to enable accurate low-power neural network inference," in *Intl. Electron Devices Meeting (IEDM)*, 2022, pp. 21.7.1–21.7.4.
11. W. Wan, R. Kubendran, C. Schaefer, S. B. Eryilmaz, W. Zhang, D. Wu *et al.*, "A compute-in-memory chip based on resistive random-access memory," *Nat.*, vol. 608, no. 7923, pp. 504–512, Aug 2022.
12. M. Bavandpour, S. Sahay, M. R. Mahmoodi, and D. Strukov, "Efficient mixed-signal neurocomputing via successive integration and rescaling," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 3, pp. 823–827, 2020.

Ben Feinberg is a Senior Member of Technical Staff at Sandia National Laboratories, New Mexico. His research interests include architectures using novel memory technologies, reliable computing, and architecture modeling. He received his Ph.D. degree in Electrical Engineering from the University of Rochester. Contact him at bfeinbe@sandia.gov.

T. Patrick Xiao is a Principal Member of Technical Staff at Sandia National Laboratories, New Mexico. His research interests include emerging memory technologies and emerging physics-based or analog computing systems. He received his Ph.D. degree in Electrical Engineering from the University of California, Berkeley. Contact him at txiao@sandia.gov.

Christopher H. Bennett is a Principal Member of Technical Staff at Sandia National Laboratories, New Mexico. His research interests include emerging devices such as polymer, spintronic, filamentary, and charge-based devices, and how they can be used for on-chip learning and/or edge inference. He received his Ph.D. degree in Electrical Engineering and Applied Physics from the Université Paris-Saclay. Contact him at cbennet@sandia.gov.

Sapan Agarwal is a Principal Member of Technical Staff at Sandia National Laboratories, California. His research interests include the co-design of computing systems from materials to algorithms and novel computing technologies. He received his Ph.D. degree in Electrical Engineering from the University of California, Berkeley. Contact him at sagarwa@sandia.gov.