Mixed-Signal Computing for Deep Neural Network Inference

Boris Murmann, *Fellow, IEEE*

(Invited Paper)

***Abstract*— Modern deep neural networks (DNNs) require billions of multiply-accumulate operations per inference. Given that these computations demand relatively low precision, it is feasible to consider analog computing, which can be more efficient than digital in the low-SNR regime. This overview article investigates the potential of mixed analog/digital computing approaches in the context of modern DNN processor architectures, which are typically limited by memory access. We discuss how memory-like and in-memory compute fabrics may help alleviate this bottleneck and derive asymptotic efficiency limits at the processing array level. It is shown that single-digit fJ/op energy efficiencies are feasible for 4-bit mixed-signal arithmetic. In this analysis, special consideration is given to the SNR and amortization requirements of the analog–digital interfaces. In addition, we consider the pros and cons for a variety of implementation styles and highlight the challenge of retaining high compute efficiency for a complete DNN accelerator design.**

***Index Terms*— Deep learning, hardware accelerators, in-memory computing, machine learning (ML), mixed analog digital integrated circuits, neural networks, switched capacitor (SC) circuits.**

I. Introduction

DEEP neural networks (DNNs) have proven remarkably effective in a variety of machine learning (ML) tasks including image classification, speech recognition, and natural language processing [1]. Over the past decade, this has fueled an unprecedented level of interest in ML, with currently over 100 publications appearing per day [2]. While much of this research is focused on algorithmic advancements, a parallel and synergistic thread lies in designing efficient hardware platforms for DNN training and inference.

Even moderate-size DNNs contain millions of parameters and require billions of computations for a single inference, making it difficult to run them on resource-constrained platforms. One solution to this problem lies in embracing domain-specific hardware accelerators [3], [4], which are customized toward the workloads of a certain application class. For DNNs, these accelerators provide specialized operations and parallel computing to improve speed and energy consumption by orders of magnitude relative to general purpose processors. Progress in this field heavily relies on hardware-algorithm co-design [5] and has led to a variety of insightful design principles for custom digital DNN chips [6]–[9].

Manuscript received July 12, 2020; accepted August 5, 2020. Date of publication September 15, 2020; date of current version December 29, 2020. This work was supported in part by Stanford's SystemX Alliance and in part by the National Science Foundation under Grant 1937294.

The author is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: murmann@stanford.edu).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2020.3020286

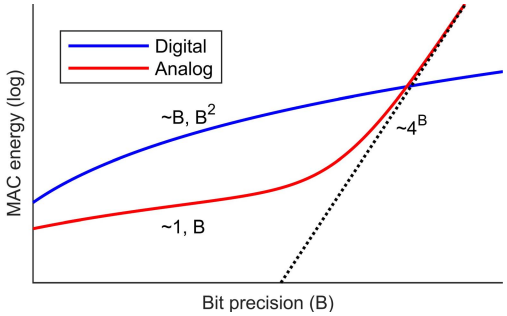


Fig. 1. Qualitative plot of energy cost for digital and analog MAC operations versus bit precision.

This article explores an additional customization vector by embracing analog circuit fabrics for DNN inference. This direction is motivated by two observations. First, DNN inference is inherently resilient to small computation errors, and can be desensitized further by injecting noise during training [10], [11]. This property enables bitwidth reductions down to four bits and below [5], [12] and can be exploited to absorb analog processing errors. In this context, it is worth noting that DNN training does not offer the same resilience [13] (at least 8-bit floats tend to be required). Second, it is well understood that analog computation can be more efficient than digital for low bit precisions [14]–[16]. This is illustrated in Fig. 1, which provides a qualitative comparison for multiply-accumulate (MAC) operations. The analog energy cost rises steeply for high bitwidths, where thermal noise is the dominant nonideality. Since the rms value of thermal noise is proportional to $\sqrt{kT/C}$, achieving an extra bit of precision in this regime requires quadrupling the capacitance and hence quadrupling the energy. However, typically below 8-bit precision, analog realizations can be superior to digital due to their structural simplicity.

While there exists an extensive body of the literature on analog neural networks, the purpose of this article is to revisit the subject with recent trends in DNN algorithms and domain-specific hardware optimization in mind. We exclude neuromorphic approaches [17], and consider mainstay kernels used in traditional digital accelerators. In particular, we focus on convolution layers, which tend to dominate the size and workload of modern DNNs with millions of parameters [6]. Within this specific context, the potential merits of analog computation are nonobvious and deserve further discussion. Most significantly, it is well understood that the energy consumption of digital DNNs is dominated by data movement and memory access [6], seemingly obviating the need for innovation in the arithmetic units. However, it is important to realize

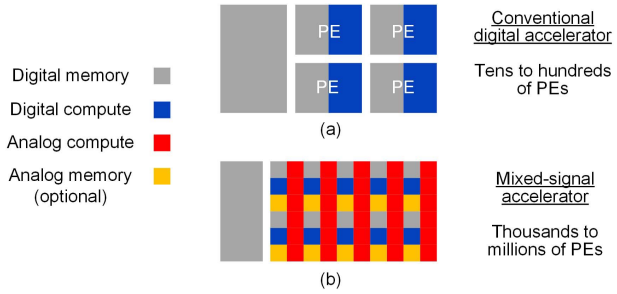


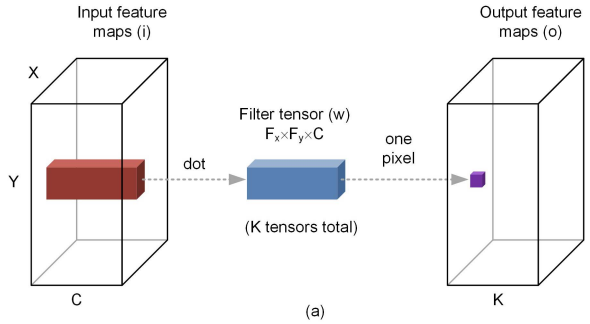
Fig. 2. Bird's eye view of (a) typical digital DNN accelerator and (b) explored mixed-signal accelerator concept.

that the construction and density of the compute elements plays a significant role in a DNN's data flow. In other words, the motivation for including analog fabrics is not limited to improving arithmetic energy but extends to introducing new degrees of freedom for reduced data movement and memory access.

As shown in the cartoon illustration of Fig. 2(a), a conventional digital DNN accelerator resembles the structure of a multicore processor and contains an array of tens to hundreds of processing elements (PEs). A PE typically measures a few hundred micrometers on a side and includes a small amount of buffer memory plus strictly separated MAC arithmetic. Each MAC operation triggers several accesses to the local memory, introducing unwanted energy overhead. In addition, once the buffers have been filled or exhausted, data movement to a larger global buffer (with higher access energy) is required. A key objective for the circuits discussed in this article is to reduce the overhead caused by these memory accesses.

Fig. 2(b) outlines the pursued approach from a birds-eye perspective. Digital as well as analog memory and compute elements are densely interspersed to aim for a "best of both worlds" mixed-signal solution. Dense digital memory is kept in the mix for fast data caching, and so is digital arithmetic for operations that are difficult to emulate in the analog domain. Analog compute is primarily added to enable low-energy spatial accumulation [vertical red bars in Fig. 2(b)]. Instead of writing a local compute result to memory, it can be accumulated via current or charge summation on a wire. Finally, if dense multilevel nonvolatile memory is available, it can find dual use for weight storage and synaptic operations via Ohm's law (current = voltage × stored conductance). In all possible combinations, close attention must be paid to the efficiency of the required D/A and A/D interfaces.

Avariety of unit cell sizes are possible within this generic framework. One extreme, known as in-memory computing [18]–[20], prioritizes density and makes do with whatever compute function can be integrated on the scale of a memory cell. Another plausible idea is to employ larger mixed-signal cells that are still substantially smaller than a digital PE but offer enhanced capabilities relative to the in-memory paradigm. We refer to this latter approach as "memory-like." The tradeoff space between these options has not yet been studied with respect to accelerator architecture benchmarks and flexibility. These aspects require further research that lies beyond the scope of this article, which mainly provides an overview of the various circuit options for mixed-signal processing arrays.




```
for (k=0 to K-1); each output channel
  for (c=0 to C-1); each input channel
    for (x=0 to X-1); each input column
      for (y=0 to Y-1); each input row
        for (fx=0 to Fx-1); each filter column
          for (fy=0 to FY-1); each filter row
            o[k, x, y] += w[k, c, fx, fy] × i[c, x+fx, y+fy]
```

(b)

Fig. 3. (a) MAC operation performed by a convolutional DNN layer. (b) For-loop representation.

The remainder of this article is structured as follows. Section II establishes important preliminaries by reviewing the required computations for convolutional layers and the basics of state-of-the-art digital accelerators. Section III represents the core of this article and discusses circuit design options for the mixed-signal fabrics in question. We cover capacitive and resistive solutions, ranging from memory-like to in-memory design styles. Finally, Section IV discusses system-level aspects and identifies opportunities for future research, while Section V concludes this article.

II. Preliminaries

A. Convolutional Neural Networks

Convolutional neural networks (CNNs) [21], [22] are among the most popular DNN architectures and presently achieve the highest classification accuracy for a variety of tasks. They consist of a cascade of layers, each of which compute convolutions (or more precisely, correlations), max pooling, normalization, and nonlinear activation functions. In essence, these layers extract increasingly abstract features that are ultimately fed into fully connected layers for classification. In modern CNNs, the convolutions are by far the most compute intensive operations and are hence our focus.

Asshown in Fig. 3(a), each convolution layer takes a 3-D input tensor that can be viewed as a stack of C images with resolution $X \times Y$. For example, if the first layer sees an RGB video graphics array (VGA) input image, the tensor dimensions are $640 \times 480 \times 3$. K filter tensors slide through the input tensor along the X and Y dimensions and each determine one pixel in the output volume via a full dot product. The required computations can be concisely summarized as a nested for-loop, shown in Fig. 3(b). The filter size $F_X \times F_Y$ is typically 3×3, and larger "receptive fields" are nowadays achieved by stacking more layers [23]. However, the layer dimensions vary between different CNN designs and also within a network, necessitating some hardware flexibility. For example, in VGG16 [23] the core layer dimensions vary between $112 \times 112 \times 128$ and $14 \times 14 \times 512$ ($X \times Y \times C$).

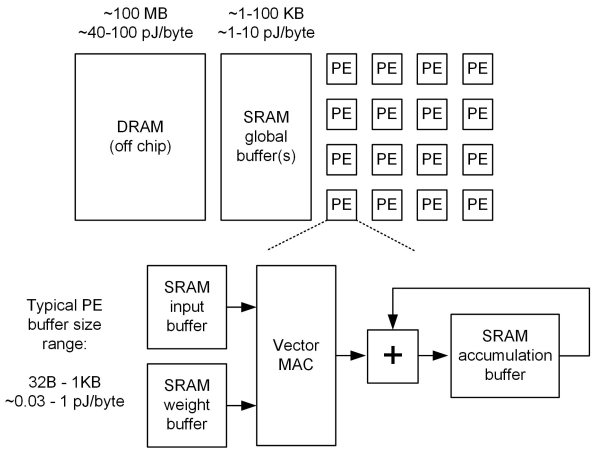


Fig. 4. Simplified block diagram of a typical digital DNN accelerator.

The most flexible option would be to employ a CPU-like architecture with one MAC unit that executes the algorithm of Fig. 3(b) sequentially, fetching the operands one by one from memory. However, this fully temporal approach is extremely slow and inefficient. The opposite extreme is to devise a fully spatial architecture that allocates parallel hardware for all computations. But this is usually impractical due to the extremely large chip sizes that would result in standard CMOS [24]–[26]. Dense in-memory computing in emerging technologies may one day change this situation (see Section III-C). Until then, we must opt for architectures that combine some amount of hardware parallelism with time multiplexing of these resources.

B. Digital Accelerators

As illustrated in Fig. 4, modern DNN accelerators contain a PE array that performs a large number of MAC operations in parallel. For example, the state-of-the-art design in [27] contains 16 PEs, and computes a total of 1024 8-bit MACs per cycle. Feeding the arithmetic with new operands in each cycle is accomplished using a carefully crafted memory hierarchy that exploits data-reuse opportunities. For example, as the filters slide across the input feature maps, they do not need to be reloaded from external DRAM memory each time but can be temporarily "cached" within the global buffer or within the PE. Similarly, partial sums from a previous filter position may be reused for the computation of the next output pixel. Embracing such data reuse strategies is extremely critical for high efficiency, since accessing memory tends to cost significantly more energy than arithmetic operations. Considering typical activity factors, even a 16-bit MAC operation costs only 75 fJ [28] in 28-nm CMOS, and numbers below 10 fJ are possible for lower bitwidths and more advanced technology nodes [29]–[31]. On the other hand, accessing a 1-kB SRAM costs about 1 pJ/byte and should therefore be amortized across many computations.

Useful taxonomy has been defined to distinguish where and how the data reuse occurs [7], [9]. Accordingly, one can differentiate between weight stationary, output stationary and row-stationary dataflows, as well as spatial and temporal reuse. In essence, these choices correspond to the various ways that the for-loops in Fig. 3(b) are unrolled, blocked, and/or reordered.

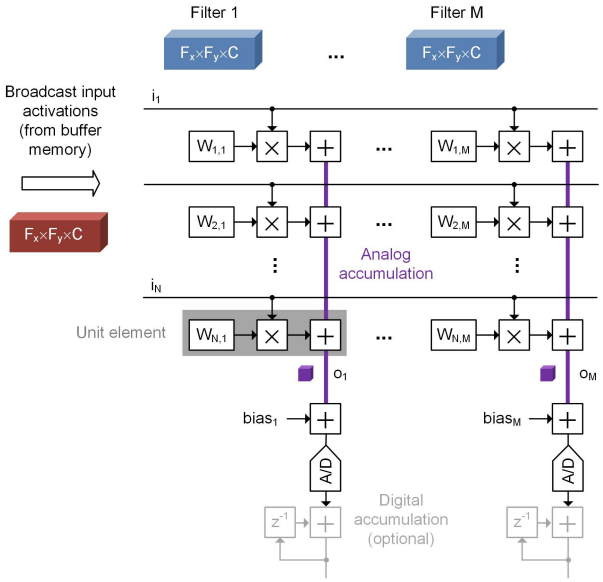


Fig. 5. Template for mixed-signal processing arrays.

The combination of these options, along with the sizing of the memory hierarchy, leads to an extremely large optimization space that is difficult to navigate. However, the systematic study of Yang *et al.* [28] noted that most dataflow options can be near-optimal as long as a high degree of data reuse and resource utilization is ensured. More significantly, it was shown that the accelerator's energy efficiency is most closely tied to the design of the hierarchical memory system and the sizing of each level in this hierarchy. As shown in Fig. 4, a typical design has three levels: external DRAM, global SRAM buffer (or double buffer), and local buffers within each PE.

It was found that the size of the PE buffers should be minimized to balance the energy across the system [28], which can be reconciled by the fact that they are typically accessed in every compute cycle (two reads and one write per MAC). However, even with small PE buffers and some limited amount of spatial reuse, each MAC operation will come with significant memory access overhead, typically making it difficult to improve digital PEs beyond ~150–200 fJ/MAC for 8-bit operands (see the state-of-the-art design of Zimmer *et al.* [27]).

III. Mixed-Signal Processing Arrays

In light of the PE buffer overhead discussed in Section II, we conjecture that mixed-signal processing arrays offer a degree of freedom that is worth exploring for efficient DNN accelerator design. Fig. 5 shows a generic template for the approaches discussed in this article. The concept is to build a large weight-stationary array of small unit elements (UEs), exploiting the density advantage of minimalistic memory-cell-like circuits [32]. Each UE can be viewed as a "synapse" and holds one (or a few) weights that can remain stationary for a large number of compute cycles (or indefinitely). As a result, weight movement and weight access energy are small. For low-energy accumulation, an analog charge or current bus collects column-wise partial products without involving digital logic or memory. This is similar to a reduction tree in digital architectures but can be more area and energy efficient.

The cost for the analog summation is that an A/D conversion is required for each column. However, each conversion is amortized across many rows. Lastly, the input activations are broadcast across many columns (spatial reuse) to amortize the access energy from the global buffer. Various options exist for implementing the required multiplications in each UE, as discussed below.

Potentially, the array can be sized such that an entire filter is held in a single column. This unrolls the F_X, F_Y, and C loops in Fig. 3(b) and turns each column into a complete "neuron." If this is not possible (or desirable), the accumulations can be completed in the digital domain, as shown. It is noted that the schematic in Fig. 5 also shows the usual bias that must be added to the convolution output before going through the layer's nonlinear activation function (often a rectifying linear unit, ReLU). In addition, linear scaling is needed for batch normalization. We omit such details in this article for brevity and because they have limited impact on the array's efficiency (as long as its size is sufficiently large).

The number of columns (M) can potentially be chosen to process all K filters of a DNN layer in parallel to attain maximum spatial reuse of global input activation buffer access [this fully unrolls the K loop in Fig. 3(b)]. Alternatively, the weights can be reloaded after several array computations, which still offers substantial amortization of global activation and weight buffer access energy for a reasonably large array.

The array concept of Fig. 5 can be implemented in many different styles. In this article, we focus on a subset of options that have proven attractive. For example, we omit approaches that rely on current summing from SRAM cells, which are relatively difficult to deploy due to high variability (see [19] for an in-depth discussion).

A. Binary Switched Capacitor Arrays

We begin with schemes using switched capacitor (SC) accumulation. The appeal of this approach mainly stems from the small mismatch of metal–oxide–metal (MOM) capacitors in modern CMOS. Even for subfemtofarad capacitors, standard deviations of around 1% or better are achievable [33], [34]. This enables compact, low-energy accumulation at DNN-compatible precision. In addition, this approach is relatively robust to global variations in process parameters, supply voltage and temperature (PVT).

Fig. 6(a) shows the conceptual schematic of a capacitive MAC array for binary neural networks. Binarization reduces multiplications to XNOR operations [35], yielding compact UE layouts. Our implementation in [32] uses a latch for weight storage and integrates an array with $N \times M = 1024 \times 64$ and fully analog accumulation. More recent designs employ SRAM cells for weight storage and employ only 8–10 transistors per UE for significant density improvements [36]–[38]. Since the cell density is close to that of standard SRAM, these latter approaches are categorized as in-memory computing.

An important design detail in this scheme is how the charge sharing is implemented. An advantage of the circuit depicted in Fig. 6(a) is that dynamic energy dissipation occurs only if the input activations switch, similar to standard CMOS logic.

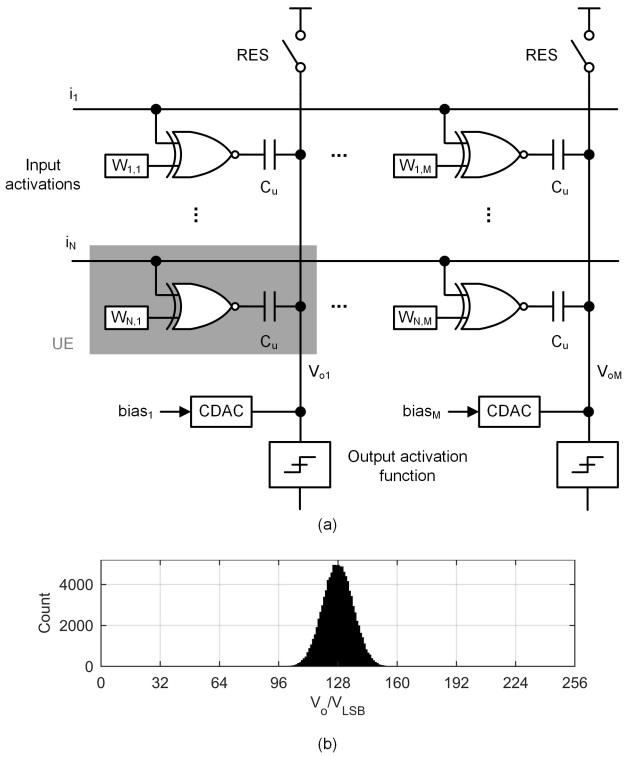


Fig. 6. (a) Mixed-signal array with binary weights and activations (actual implementation may be differential). (b) Column voltage distribution assuming independent random inputs, zero bias, and $N = 256$.

However, a charge reset (RES) must be performed periodically to overcome leakage. This necessitates additional transistors inside the UE to override the XNOR output. In [32], dedicated NOR gates are added for this [not shown in Fig. 6(a)], while [36] uses array's wordlines for RES. For alternative schemes with grounded unit capacitors [37], [38], the RES and extra gates are not needed, but dynamic energy is dissipated even if the input activations do not switch. The energy overhead can be significant, since the activity factors of DNN input activations tend to be below 10% [32].

Another important design aspect relates to the dynamic range of the charge accumulation and the tolerable analog circuit errors. Unfortunately, the requirements depend on the DNN algorithm, inference accuracy, signal statistics, and the employed training methods, which can help desensitize the network [39]–[42]. Still, we attempt to articulate some approximate guidelines. First, it is important to note that the signal swing on the accumulation wire is usually relatively small compared to the supply voltage. If we assume random and uncorrelated binary inputs and weights (with $\Pr(0) = \Pr(1) = 0.5$), the accumulation result has a Binomial distribution with mean $N/2$ and standard deviation $\sqrt{N}/2$. This is illustrated for $N = 256$ in Fig. 6(b), where $V_{\mathrm{LSB}} = V_{\mathrm{DD}}/N$ denotes a unit increment in the accumulation voltage. Although the input data is not purely random and uncorrelated in an actual DNN, this first-order statistical approximation is not too far off from the values observed in [36] and [40].

In terms of random errors, the DNN-level simulations of [32] and [40] suggest tolerable standard deviations on the order of V_{LSB} for binarized networks, which does not conflict

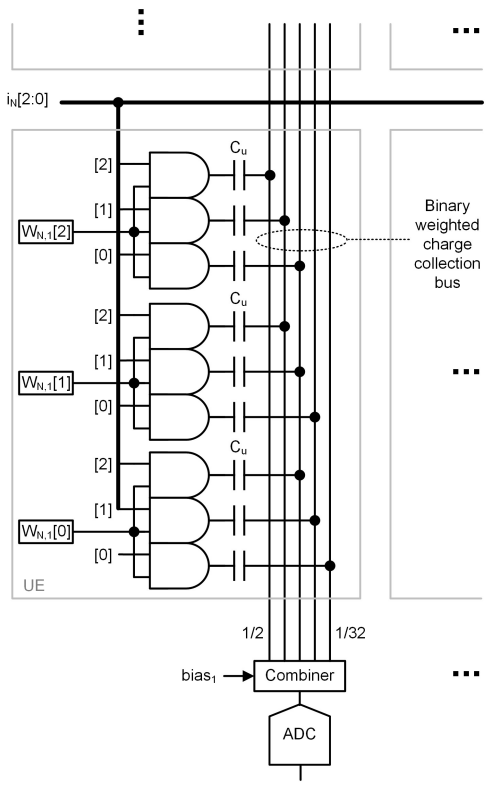


Fig. 7. Conceptual schematic of a 3-bit SC UE.

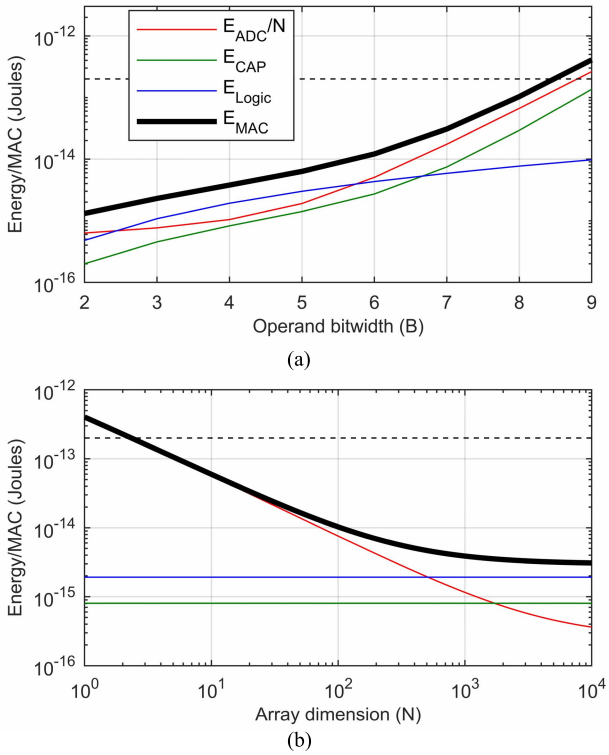


Fig. 8. MAC energy estimate for the multibit array in Fig. 7 assuming 28-nm CMOS. (a) Energy versus operand bitwidth (B) for $N = 3 \times 3 \times 128 = 1152$. (b) Energy versus array dimension (N) for $B = 4$.

with intuition. If $V_{DD} = 1$ V and $N = 256$, this translates into an easily achievable noise specification of $\sim$4 mV_{rms} for the comparators that slice the accumulation result, as well as the kT/C noise at the charge summation node. Similar considerations apply to capacitor mismatch. For 1% matching, the cumulative error is $1\%\sqrt{N}$ and N would have to exceed 10 000 before the standard deviation rises to the level of one input increment.

Owing to these relatively mild circuit requirements, chip demonstrators of the discussed approach tend to achieve compute efficiencies near or below 4 fJ/MAC, even with all peripheral memory accesses considered. However, fully digital implementations that follow a similar architecture (employing adder trees) and dataflow can get close to these numbers as well [30]–[32] (see Section IV). This suggests that that mixed-signal processing is not necessarily a game changer for binarized DNN accelerators. It therefore makes sense to investigate multibit implementations, which can also cover a wider application space.

B. Multibit SC Arrays

Fig. 7 shows one way to extend the concept of Fig. 6 to multibit operation. For simplicity, the circuit assumes unsigned operands, but it can be modified to achieve signed operation. The topology resembles a digital multiplier, but the local partial products are collected in the charge domain (instead of using digital full adders). The shown charge collection bus hence serves the dual purpose of local accumulation and accumulation within the array. The voltages on this bus must be combined in a binary-weighted manner before or during A/D conversion. It may be possible to integrate the combiner within the ADC's input network to minimize overhead (e.g., using binary splitting of the sampling capacitance).

Instead of allocating nine AND gates for parallel 3-bit operations, one can also process the input activations serially, as proposed in the SRAM in-memory compute processor of [37]. This implementation distributes the weight bits into separate SRAM columns and serializes the input activation bits into the array, with one A/D conversion per time step and column. The final weighted result is then assembled in the digital domain. The advantage of this scheme is that it achieves higher density (one AND gate per UE) and programmability, but it trades this for lower throughput and extra A/D conversions. This serial approach is preferred when achieving high density and variable bitwidth is the primary objective, while a more parallel configuration may bring some improvements in throughput and energy efficiency.

To establish a feel for the achievable energy efficiency of the fully parallel configuration, we performed a quantitative analysis that sweeps the bitwidths of the operands (B, assumed equal for weights and activations) and the vertical array dimension N. The energy per MAC operation is expressed as

$$E_{MAC} = \frac{E_{ADC}}{N} + E_{CAP} + E_{Logic} \quad (1)$$

where E_{ADC} is the ADC's conversion energy, which is amortized over the number of rows in the array (N). E_{CAP} and E_{Logic} capture the energy consumption due to the unit capacitances (C_u) and logic gates in each UE.

The obtained results are shown in Fig. 8 and the underlying assumptions are detailed in the Appendix. Fig. 8(a) is a plot against operand bitwidth (B) assuming a large array

with $N = 1152$. This corresponds to a desired scenario with a highly amortized ADC, keeping its energy contribution roughly on-par with the UE's logic gates for $B = 2$–6. However, for higher bitwidths, the ADC ultimately dominates due to the steep tradeoff between energy and thermal noise requirements. Beyond $B = 8$, the estimate exceeds 200 fJ/MAC, which is also reachable with purely digital implementations [27]. It should be kept in mind that this result is sensitive to the underlying assumptions, but regardless, competing with digital above 6–7 bits is challenging and will likely yield diminishing returns. However, for $B = 4$, the predicted MAC energy is 3.8 fJ, which would be attractive if realizable in practice.

Fig. 8(b) is a plot against the number of rows (N) with $B = 4$. It shows that the A/D conversion should be amortized over at least over a few hundred rows to achieve high efficiency. Also, it is noted that the ADC energy (per MAC) eventually saturates due to the signal shrinkage effect mentioned in Section III-D. Our analysis assumes that the fractional signal amplitude (relative to V_{DD}) is proportional to $1/\sqrt{N}$, so that a quadrupling of N reduces the swing by $2\times$. Once the signal is small enough to make the ADC purely noise-limited, a $2\times$ swing reduction leads to a quadrupling in energy. Hence, the amortization benefit stops, and the ADC energy flattens for large N.

C. Resistive Arrays

None of the processing arrays discussed so far are dense enough to hold all the weights of a large DNN (say >10 MB) on a single die under reasonable chip area constraints. However, this could become possible using in-memory computing with emerging memory technologies. For instance, resistive RAM (RRAM) offers cell sizes of 53 F^2 [43], where F is the half pitch the process technology. For comparison, the memory-like, single-bit UE of [32] measures 24 000 F^2, while the SRAM-based cell of [38] occupies 290 F^2. Additionally, these emerging memories are typically nonvolatile, which enables power cycling without reloading the weights from external flash memory. The reader is referred to [44] and [45] for a comprehensive review on the large variety of competing technologies (RRAM, MRAM, ferroelectric RAM (FRAM), phase change RAM (PCRAM), etc.).

Most emerging memory technologies are based on storing the information on programmable resistors. The resistors can be programed between two states and read out row-by-row like standard digital memory. Alternatively, the array can be used for analog in-memory computing in accordance with the template of Fig. 5. This is illustrated in Fig. 9(a), which shows an array of "1T1R" cells, where the transistor is mainly needed for programming. The input activations are applied to the programmable resistors in each unit cell, which represent the filter weights. Accumulation occurs by summing the currents along each column (current = voltage × stored conductance). The natural solution is to process the sum in the current domain (via a transimpedance amplifier, integrator [46], or current-mode comparator [47]), but presumably more efficient voltage-domain readouts are also possible [48]–[50]. As discussed previously, the column readout circuit must be well amortized to achieve high-energy efficiency. An additional challenge in these dense arrays is fitting the interface circuits to the extremely fine pitch.

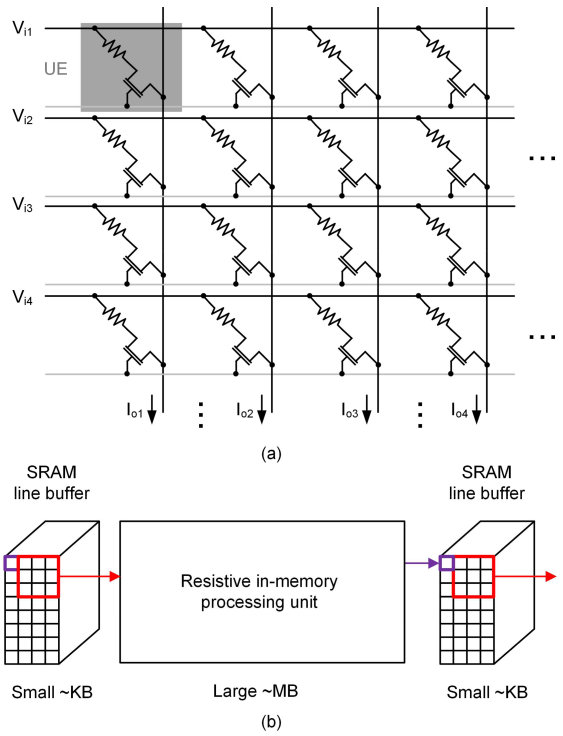


Fig. 9. (a) Resistive array. (b) Layer pipelining.

For some technologies (e.g., RRAM) it is conceivable to store multibit weights in each unit cell [51]. Alternatively, the weight bits can be split among multiple cells as in SRAM-based designs (see [47]). For multibit activations, the inputs can be analog voltages generated by a DAC [52]. However, for low bitwidths, the DAC overhead can be avoided by serializing [47] or pulsewidth modulating the input activations [49]. Another important consideration is the energy dissipation caused by the resistors alone. Applying 0.5 V across 10 kΩ for 10 ns (typical readout time) corresponds to 250 fJ/MAC. It is therefore desirable to work with much larger resistance values [53], but this often stands at odds with maintaining low variability [51].

As an alternative, we have explored a fully dynamic scheme where the bitlines are precharged and discharge occurs according to the dot product of pulsed input activations and RRAM weight conductances [49]. In this design, the bitline transients are digitized using 2-bit ramp ADCs. The DAC block within these converters are integrated and shared within the cell array, to provide a replica with similar behavior as the cells used for the dot product (similar to [54]). This idea exploits the "signal shrinkage" effect mentioned in Section III-A. Only a relatively small number of reference cells are needed to cover the output range (e.g., 100 replica rows for an array with several thousand rows total). For this implementation with 2-bit inputs, 2-bit outputs and five-level weights, postlayout SPICE simulations in 40-nm CMOS predict single-digit fJ/op for arrays with ∼3000–13000 rows. These results include benefits from sparsity-aware processing. Specifically, if the ramp ADCs trip in early conversion cycles, further bitline

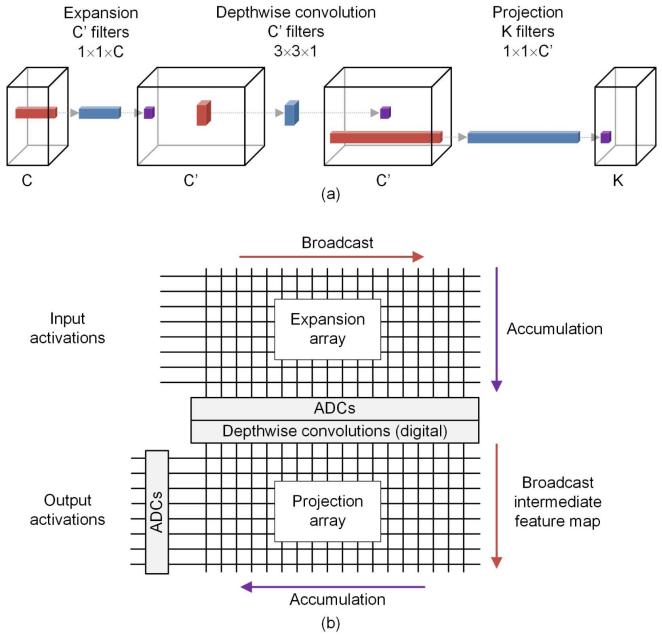


Fig. 10. (a) MobileNetV2 Bottleneck layer. (b) Pipelined implementation with mixed-signal processing.

precharges are skipped, leading to an energy benefit of about 2.5×.

While there are many possible ways to incorporate emerging nonvolatile memory into a DNN accelerator, one attractive option is a streaming (pipelined) topology as shown in Fig. 9(b). Here, large in-memory compute tiles are pipelined between small SRAM line buffers that hold only the current input working set [55]. To form a complete accelerator, many such tiles can potentially be integrated on a single chip, as described in [56]. In such an architecture, all weights remain stationary and the data slides through the fabric with massively parallel computation.

At present, the art of designing complete DNN processors using emerging memory is still in its infancy. Key issues include access to process technology as well as challenges with variability, retention and endurance (see [57]). Programing the resistance values accurately and reading consistent values across operating conditions and lifetime is a significant issue, despite the error tolerance of neural networks (see [44] and [45] for a comprehensive overview). As a result, most demonstrators are only subsystems and use relatively small arrays (see [47] and [48]). Furthermore, some implementations activate only a small number of rows simultaneously to simplify the design (e.g., $3 \times 3 = 9$ rows in [47]), which leads to diminishing performance gains. However, the vast amount of ongoing research will undoubtedly change this, and may one day lead to arrays operating at levels near 0.1 fJ/op [53].

D. Arrays for Bottleneck Layers

So far, we have assumed that the processing arrays handles only "vanilla" convolution layers as shown in Fig. 3(a). However, modern DNNs leverage fine-tuned layer structures to reduce the number of required weights and MACs. Consider for example the so-called "bottleneck layers" of MobileNetV2 [58] [see Fig. 10(a)]. Each layer's input has a relatively small number of channels (C) that gets internally expanded using 1×1 filters. The so-created intermediate feature map is processed using depthwise separable convolutions, which require significantly less MAC operations than standard convolutions. Finally, the filtered feature map is projected back into a lower dimensional tensor that is passed to the next layer.

Mixed-signal arrays could be used to pipeline this cascade of operations and avoid storing the large intermediate feature map in memory. As shown in Fig. 10(b), a first MAC array can be allocated for the expansion operation, with its columns holding the expansion filters. The array outputs are digitized for depthwise convolutions in the digital domain. Mixed-signal computing would be relatively ineffective here due to the small kernel size (3×3 filters lead to nine-row accumulation with poor ADC amortization). However, mixed-signal processing is useful again for the projection operation, which has its filter weights reside along the rows of the bottom array. In principle, this customized array structure can be realized with any of the circuit styles discussed in Sections III-A–III-C.

IV. System Aspects

Assessing the system-level benefits of the above-described processing arrays is a challenging task and requires further research in the style of [28] and [59]. Nonetheless, we want to give the reader a feel for the basic aspects.

A. Efficiency Benchmarking

For a given application, the most objective way to compare the efficiency of DNN inference accelerators is to measure their total energy per inference for the same task and inference accuracy. However, this is difficult for the mixed-signal design space, which is dominated by partial hardware realizations that emulate the full network only in software. A compromise is thus to quantify these subsystems by their number of performed operations per watt. This is typically expressed in tera-operations per second and watt, TOPS/W. One TOPS/W corresponds to 1 pJ per operation and one MAC is defined as two operations. In addition to pure energy efficiency, it is also common to compute the area efficiency in TOPS/mm^2 since energy efficiency alone is not useful without high compute density. The online survey of [60] is a useful resource for tracking the latest advancements on these benchmarks.

Unfortunately, comparisons in terms of these metrics in isolation can be flawed. For example, it is relatively easy to achieve high efficiency in small arrays and for small bitwidths. However, it is unclear whether this array-level efficiency will translate to higher efficiency at the accelerator level. Limited array size may lead to an increased number of memory access across the hierarchy and low bit-resolution typically requires a larger number of weights in the network. The example in [9] shows that a 4-bit accelerator with 5.5 TOPS/S can achieve nearly the same inference energy efficiency as a 1-bit, 230 TOPS/W design. The relationship between the chosen bitwidth and total inference energy is generally unknown, although Esser *et al.* [12] provide some insights on network size dependencies.

Keeping these limitations in mind, we plot the performance of a few recent designs in Fig. 11, merely to obtain a feel for general trends. The dashed horizontal lines correspond

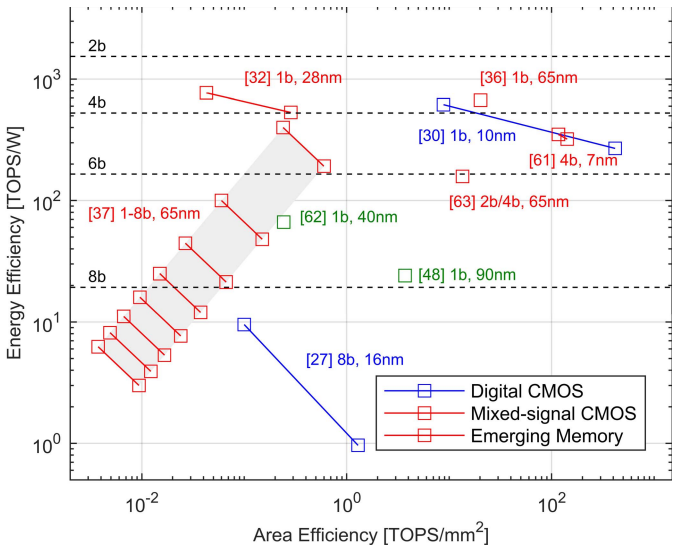


Fig. 11. Energy and area efficiency comparison of a few recent designs. Each connected data point pair corresponds to low/high supply voltage.

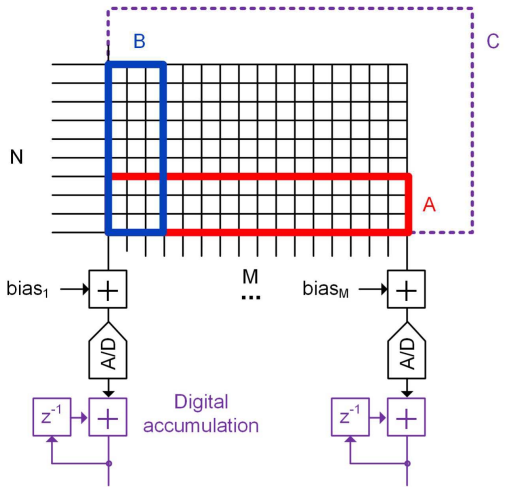


Fig. 12. Array utilization problem.

to data points from Fig. 8(a), with the 4-bit line located at 526 TOPS/W. So far, these efficiency levels have been mainly reported for binarized architectures such as [30], [32], and [37] in 1-bit mode. But, the 4-bit design of Dong *et al.* [61] achieves 351 TOPS/W, and thus serves as an exitance proof (albeit using 7-nm CMOS and for a very small array). Further research is thus needed to study the real-world merits and area efficiency of the topology in Fig. 7.

A second observation is that operand scaling from 1 to 8 bits for the SRAM in-memory compute design of Jia *et al.* [37] comes at a significant cost, as energy and area efficiency scale linearly with the number of bits in both MAC operands. If this chip were to be redesigned in 16-nm CMOS or below, its performance would undoubtedly improve by a significant factor, but it is unclear if it would outperform the fully digital design of Zimmer *et al.* [27] for 8-bit arithmetic. This reiterates our previous argument that competing with digital above 6 bits is difficult.

A third observation is that designs using emerging memories, such as [48] and [62] do not yet stand out, primarily due to the challenges outlined in Section III-C. It is in fact difficult to find fully benchmarked designs that include the absolute throughout in TOPS (there is a tendency to report only the energy efficiency). However, we also see that most other mixed-signal designs lack area efficiency, and this is where emerging memory technology is expected to help. It is worth noting that the area efficiency of [36] and [63] is likely boosted by their small array sizes (which helps increase the speed in TOPS). A similar argument could be made for [61], but here the extreme density of the underlying 7-nm SRAM array (with 8T push-rule cells) is another significant factor.

B. Flexibility

While the described mixed-signal processing arrays can help with memory access amortization, we are forced to pick the array sizes at design time. This stands at odds with the flexibility that is desired to support the latest developments in DNN architectures. At the most basic level, there are potential issues with arrays utilization, as seen in Fig. 12. If a kernel is not tall enough for the array (case A), this leads to reduced ADC amortization. If a kernel is not wide enough for the array (case B), this leads to reduced input buffer amortization. On the other hand, if the kernel is much larger than the array (case C), less loop unrolling is possible, leading to extra A/D conversions, a requirement of post-ADC digital accumulation, as well as data movement penalties (weight reloads, insufficient buffer amortization). The latter issue is seen in the small 64×64 array of [63], which reports a $\sim 4.5\times$ efficiency degradation when the external buffer accesses are added to the energy budget. Another challenge with mixed-signal arrays is to leverage data-dependent energy savings. For instance, one can skip computations for zero-valued operands, or employ some form of dynamic precision scaling [9], [64]. The mixed-signal community has begun to explore such opportunities [49], [63], but their implementation is not as straightforward and fruitful as in purely digital circuits.

The above-mentioned issues are most difficult to manage for dense in-memory compute fabrics, where the tight pitch and push-rule constraints may not allow the inclusion of any static or dynamic reconfiguration circuitry. An interesting research question is whether there exists a middle-ground architecture in the style of Fig. 7, which does not push density to the limit, but in return offers better reconfigurability for flexible DNN mapping and execution. The designs of [37] and [38] take steps in this direction via array tiling. However, pursuing this path further requires a solid-framework for performance prediction, which is still evolving even for digital architectures [65], [66]. Even more desirable would be a framework for hardware-algorithm co-design, but such tool developments are still at their infancy [67].

V. Conclusion

The design of domain-specific hardware for DNNs is an exciting and fast-paced research area. As new algorithms, network topologies, and accelerators advance in unison, offering hardware flexibility is of primary interest and is likely best addressed by fully digital architectures. However, as the field matures, it is reasonable to expect that in some application scenarios the need for the highest possible efficiency will

asymptotically outrank the craving for ample flexibility. With this expectation in mind, it is interesting to consider the merits of mixed-signal compute fabrics for DNN acceleration.

While there is no shortage of analog/mixed-signal demonstrators that boast low-energy computation, the present challenge lies in incorporating these macros without losing this efficiency within a complete accelerator design. Future work on mixed-signal solutions should consider data movement and required layer reconfigurability as a key aspect, and not leave it as an afterthought for software emulation. This requirement is particularly important due to the relatively narrow performance advantage that can apparently be gained from going mixed-signal in standard CMOS. Custom digital hardware is difficult to beat and demonstrating a 5–10× advantage in a realistic setting will only be possible if all aspects are considered.

Future DNN accelerators will likely leverage emerging memory technology to store all DNN weights on moderate-size chips and eliminate the overhead of external DRAMs and nonvolatile memory. In this scenario with full weight stationarity, using in-memory computing becomes a more natural choice and mixed-signal processing may evolve from a curiosity to an enabler. However, a significant amount of work remains to be done on understanding the effect and proper mitigation of analog nonidealities in DNN computation. In particular, more work is needed on understanding and mitigating the impact of ADC thermal noise and network weight variability. As with all aspects of this exciting area, developing such understanding must fuse insights from algorithms, network architectures, and circuit design.

APPENDIX

Here, we summarize the equations and parameters that were used to generate the MAC energy plots in Fig. 8.

A. ADC Energy Model

E_{ADC} is estimated using the experimental data from [68] and the following fitting function:

$$E_{\mathrm{ADC}} = k_1 \mathrm{ENOB} + k_2 4^{\mathrm{ENOB}}. \tag{2}$$

We conservatively set $k_1 = 100$ fJ and $k_2 = 1$ aJ to avoid overfitting to outliers that may not be usable in the intended application. Effective number of bit (ENOB) is linked as usual to the converter's signal-to-noise-and-distortion ratio (SNDR), which we employ as a conservative proxy for the required SNR

$$\mathrm{SNDR}_{\mathrm{ADC}} = \frac{3}{2} 2^{2\mathrm{ENOB}} \approx \mathrm{SNR}_{\mathrm{ADC}}. \tag{3}$$

Determining the ADC's SNR requirement is a nontrivial task and prior work has mainly resorted to DNN-specific numerical simulations [32], [39], [40], [49], while a more analytical approach is found in [69]. The main hypothesis for our numerical analysis is that the ADC's tolerable input-referred noise must be limited to a fraction of the accumulated input activation quantization noise. Weight quantization errors are not included since they can be absorbed during DNN training. Also, for generality and simplicity, we assume that the nonlinear activation function (e.g., ReLU) is not embedded in the ADC's transfer function, which allows for post-ADC accumulation of partial sums.

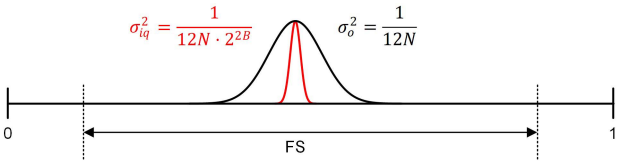


Fig. 13. Model for SNR analysis.

If the input activations and weights are uncorrelated and uniformly distributed between 0 and 1 (normalized to the supply voltage, V_{DD}), the sum of N passively combined multiply-adds has a variance of approximately $N/12N^2 = 1/12N$ (see Fig. 13). The accumulated error variance due to B-bit quantization of the activations follows similarly but is smaller by a factor of 2^{2B} (also assuming a uniform distribution). The SNR of the signal entering the ADC is thus

$$\mathrm{SNR}_{\mathrm{sig}} = \frac{\sigma_o^2}{\sigma_{iq}^2} = \frac{\frac{1}{12N}}{\frac{1}{12N \cdot 2^{2B}}} = 2^{2B} \tag{4}$$

and we want the ADC's SNR to be k^2 times larger to avoid a significant signal degradation. This requirement aligns reasonably well with the findings of [49], where $B = 2$, the signal standard deviation is several tens of LSB and the tolerable noise standard deviation is 9 LSB. More research is needed to account for effects and options that are not considered here (correlations, activation sparsity, ADC with ReLU transfer function, etc.).

Unfortunately, for large N, the signal term in (4) is much smaller than the full-scale range of a typical ADC. One way to deal with this is to amplify the signal. However, amplifiers tend to consume nearly the same power as ADCs for a given input-referred noise. To obtain a conservative estimate, we assume that the ADC is designed to meet the noise specification without the aid of a preamplifier. Thus, we express the ADC's SNR as

$$\mathrm{SNR}_{\mathrm{ADC}} = \frac{\frac{1}{2}\left(\frac{\mathrm{FS}}{2}\right)^2}{\frac{\sigma_{iq}^2}{k^2}} = \frac{1}{2}\left(\frac{\mathrm{FS}}{2}\right)^2 k^2 \cdot 12N \cdot 2^{2B} \tag{5}$$

where FS is the ADC's normalized full-scale range (see Fig. 13), and a full-scale sinusoidal input is assumed to mimic the test condition that underlies (3) and the data of [68]. Equating (5) with (3) and solving for ENOB gives

$$\mathrm{ENOB} = B + \log_2(k \cdot \mathrm{FS} \cdot \sqrt{N}). \tag{6}$$

We used $k = 2$ and FS $= 0.5$ for Fig. 8. With $N = 1152$, this leads to an excess resolution requirement [log term in (6)] of about 5 bits.

B. Logic Energy Model

$$E_{\mathrm{Logic}} = B^2 \alpha E_{\mathrm{gate}}(1 + \beta) \tag{7}$$

where α is the input activity factor, E_{gate} is the energy of a two-input logic gate, and β captures wires and other overhead not shown in the simplified circuit model. We used $\alpha = 0.1$, $E_{\mathrm{gate}} = 0.3$ fJ (∼28-nm CMOS) and $\beta = 3$ for Fig. 8.

C. Capacitive DAC Energy Model

$$E_{\mathrm{CAP}} = B^2 \alpha C_{\mathrm{u}} V_{\mathrm{DD}}^2 + E_{\mathrm{noise}} \tag{8}$$

where C_{u} is the unit capacitance. E_{noise} is an overhead term that models the upsizing of C_{u} to meet kT/C noise requirements for large B. We do not expand this term here since it is not a significant contributor to the total MAC energy. We used $C_{\mathrm{u}} = 0.5\,\mathrm{fF}$ and $V_{\mathrm{DD}} = 1\,\mathrm{V}$ for Fig. 8.

ACKNOWLEDGMENT

The author would like to acknowledge the many contributions made by colleagues, collaborators, as well as present and former students that have helped shape this article. Special thanks go to Daniel Bankman, Massimo Giordano, and Wei-Han (Hank) Yu for their assistance.

REFERENCES

[1] T. J. Sejnowski, "The unreasonable effectiveness of deep learning in artificial intelligence," *Proc. Nat. Acad. Sci. USA*, Jan. 2020, doi: 10.1073/pnas.1907373117.

[2] J. Dean, "The deep learning revolution and its implications for computer architecture and chip design," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 8–14.

[3] J. L. Hennessy and D. A. Patterson, "A new golden age for computer architecture," *Commun. ACM*, vol. 62, no. 2, pp. 48–60, Jan. 2019.

[4] W. J. Dally, Y. Turakhia, and S. Han, "Domain-specific hardware accelerators," *Commun. ACM*, vol. 63, no. 7, pp. 48–57, Jun. 2020.

[5] B. L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proc. IEEE*, vol. 108, no. 4, pp. 485–532, Apr. 2020.

[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, Dec. 2017.

[7] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, *Efficient Processing of Deep Neural Networks*. San Rafael, CA, USA: Morgan & Claypool Publishers, 2020.

[8] F. Conti, M. Rusci, and L. Benini, "The memory challenge in ultra-low power deep learning," in *Proc. NANO-CHIPS*, B. Murmann and B. Hoefflinger, Eds. Cham, Switzerland: Springer, 2020, pp. 323–349.

[9] M. Verhelst and B. Murmann, "Machine learning at the edge," in *Proc. NANO-CHIPS*, B. Murmann and B. Hoefflinger, Eds. Cham, Switzerland: Springer, 2020, pp. 293–322.

[10] A. F. Murray and P. J. Edwards, "Enhanced MLP performance and fault tolerance resulting from synaptic weight noise during training," *IEEE Trans. Neural Netw.*, vol. 5, no. 5, pp. 792–802, 1994.

[11] C. Torres-Huitzil and B. Girau, "Fault and error tolerance in neural networks: A review," *IEEE Access*, vol. 5, pp. 17322–17341, 2017.

[12] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," 2019, *arXiv:1902.08153*. [Online]. Available: http://arxiv.org/abs/1902.08153

[13] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan, "Training deep neural networks with 8-bit floating point numbers," 2018, *arXiv:1812.08011*. [Online]. Available: http://arxiv.org/abs/1812.08011

[14] E. A. Vittoz, "Future of analog in the VLSI environment," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1990, pp. 1372–1375.

[15] R. Sarpeshkar, "Analog versus digital: Extrapolating from electronics to neurobiology," *Neural Comput.*, vol. 10, no. 7, pp. 1601–1638, Oct. 1998.

[16] B. Murmann, D. Bankman, E. Chai, D. Miyashita, and L. Yang, "Mixed-signal circuits for embedded machine-learning applications," in *Proc. 49th Asilomar Conf. Signals, Syst. Comput.*, Nov. 2015, pp. 1341–1345.

[17] S. K. Bose, J. Acharya, and A. Basu, "Is my neural network neuromorphic? Taxonomy, recent trends and future directions in neuromorphic engineering," in *Proc. 53rd Asilomar Conf. Signals, Syst., Comput.*, Nov. 2019, pp. 1522–1527.

[18] M. Kang, M.-S. Keel, N. R. Shanbhag, S. Eilert, and K. Curewitz, "An energy-efficient VLSI architecture for pattern recognition via deep embedding of computation in SRAM," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2014, pp. 8326–8330.

[19] 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, Apr. 2017.

[20] N. Verma *et al.*, "In-memory computing: Advances and prospects," *IEEE Solid State Circuits Mag.*, vol. 11, no. 3, pp. 43–55, Aug. 2019.

[21] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[24] Z. Du *et al.*, "ShiDianNao," in *Proc. 42nd Annu. Int. Symp. Comput. Archit. (ISCA)*, 2015, pp. 92–104.

[25] O. Temam, "A defect-tolerant accelerator for emerging high-performance applications," in *Proc. 39th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2012, pp. 356–367.

[26] D. Bankman, "Mixed-signal processing for machine learning," Ph.D. dissertation, Stanford Univ., Stanford, CA, USA, 2019. [Online]. Available: https://purl.stanford.edu/tt451tg9318

[27] B. Zimmer *et al.*, "A 0.32–128 TOPS, scalable multi-chip-module-based deep neural network inference accelerator with ground-referenced signaling in 16 nm," *IEEE J. Solid-State Circuits*, vol. 55, no. 4, pp. 920–932, Apr. 2020.

[28] X. Yang *et al.*, "Interstellar: Using Halide's scheduling language to analyze DNN accelerators," in *Proc. 25th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Mar. 2020, pp. 369–383.

[29] W. J. Dally, C. T. Gray, J. Poulton, B. Khailany, J. Wilson, and L. Dennison, "Hardware-enabled artificial intelligence," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2018, pp. 3–6.

[30] P. C. Knag *et al.*, "A 617 TOPS/W all digital binary neural network accelerator in 10nm FinFET CMOS," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2020, pp. 1–2.

[31] B. Moons, D. Bankman, L. Yang, B. Murmann, and M. Verhelst, "BinarEye: An always-on energy-accuracy-scalable binary CNN processor with all memory on chip in 28nm CMOS," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Apr. 2018, pp. 1–4.

[32] D. Bankman *et al.*, "An always-on 3.8 μJ/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 158–172, Jan. 2019.

[33] V. Tripathi and B. Murmann, "Mismatch characterization of small metal fringe capacitors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 8, pp. 2236–2242, Aug. 2014.

[34] H. Omran, H. Alahmadi, and K. N. Salama, "Matching properties of femtofarad and sub-femtofarad MOM capacitors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 6, pp. 763–772, Jun. 2016.

[35] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or –1," 2016, *arXiv:1602.02830*. [Online]. Available: http://arxiv.org/abs/1602.02830

[36] Z. Jiang, S. Yin, J.-S. Seo, and M. Seok, "C3SRAM: An in-memory-computing SRAM macro based on robust capacitive coupling computing mechanism," *IEEE J. Solid-State Circuits*, vol. 55, no. 7, pp. 1888–1897, Jul. 2020.

[37] H. Jia, H. Valavi, Y. Tang, J. Zhang, and N. Verma, "A programmable heterogeneous microprocessor based on bit-scalable in-memory computing," *IEEE J. Solid-State Circuits*, vol. 55, no. 9, pp. 2609–2621, Sep. 2020.

[38] 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. 1789–1799, Jun. 2019.

[39] A. S. Rekhi *et al.*, "Analog/mixed-signal hardware error modeling for deep learning inference," in *Proc. 56th Annu. Design Autom. Conf.*, Jun. 2019, pp. 1–6.

[40] S. Ma, D. Brooks, and G.-Y. Wei, "A binary-activation, multi-level weight RNN and training algorithm for processing-in-memory inference with eNVM," 2019, *arXiv:1912.00106*. [Online]. Available: http://arxiv.org/abs/1912.00106

[41] B. Zhang, L.-Y. Chen, and N. Verma, "Stochastic data-driven hardware resilience to efficiently train inference models for stochastic hardware implementations," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 1388–1392.

[42] M. Klachko, M. R. Mahmoodi, and D. Strukov, "Improving noise tolerance of mixed-signal neural networks," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–8.
[43] C.-C. Chou *et al.*, "An N40 256K×44 embedded RRAM macro with SL-precharge SA and low-voltage current limiter to improve read and write performance," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 478–480.
[44] S. Yu, X. Sun, X. Peng, and S. Huang, "Compute-in-memory with emerging nonvolatile-memories: Challenges and prospects," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Mar. 2020, pp. 1–4.
[45] H. Tsai, S. Ambrogio, P. Narayanan, R. M. Shelby, and G. W. Burr, "Recent progress in analog memory-based accelerators for deep learning," *J. Phys. D, Appl. Phys.*, vol. 51, no. 28, Jul. 2018, Art. no. 283001.
[46] Q. Liu *et al.*, "A fully integrated analog ReRAM based 78.4TOPS/W compute-in-memory chip with fully parallel MAC computing," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 500–502.
[47] C.-X. Xue *et al.*, "A 1Mb multibit ReRAM computing-in-memory macro with 14.6ns parallel MAC computing time for CNN based AI edge processors," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2019, pp. 388–390.
[48] S. Yin, X. Sun, S. Yu, and J.-S. Seo, "High-throughput in-memory computing for binary deep neural networks with monolithically integrated RRAM and 90nm CMOS," 2019, *arXiv:1909.07514*. [Online]. Available: http://arxiv.org/abs/1909.07514
[49] D. Bankman, J. Messner, A. Gural, and B. Murmann, "RRAM-based in-memory computing for embedded deep neural networks," in *Proc. 53rd Asilomar Conf. Signals, Syst., Comput.*, Nov. 2019, pp. 1511–1515.
[50] W. Wan *et al.*, "A voltage-mode sensing scheme with differential-row weight mapping for energy-efficient RRAM-based in-memory computing," in *Symp. VLSI Tech. Dig.*, 2020, pp. 1–2.
[51] B. Q. Le *et al.*, "Resistive RAM with multiple bits per cell: Array-level demonstration of 3 bits per cell," *IEEE Trans. Electron Devices*, vol. 66, no. 1, pp. 641–646, Jan. 2019.
[52] F. Cai *et al.*, "A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations," *Nature Electron.*, vol. 2, no. 7, pp. 290–299, Jul. 2019.
[53] S. Cosemans *et al.*, "Towards 10000TOPS/W DNN inference with analog in-memory computing—A circuit blueprint, device options and requirements," in *IEDM Tech. Dig.*, Dec. 2019, pp. 22.2.1–22.2.4.
[54] C. Yu, T. Yoo, T. T.-H. Kim, K. C. Tshun Chuan, and B. Kim, "A 16K current-based 8T SRAM compute-in-memory macro with decoupled read/write and 1-5bit column ADC," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Mar. 2020, pp. 1–4.
[55] A. Shafiee *et al.*, "ISAAC: A convolutional neural network accelerator with *in-situ* analog arithmetic in crossbars," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 14–26.
[56] M. Dazzi, A. Sebastian, P. Andrea Francese, T. Parnell, L. Benini, and E. Eleftheriou, "5 parallel prism: A topology for pipelined implementations of convolutional neural networks using computational memory," 2019, *arXiv:1906.03474*. [Online]. Available: http://arxiv.org/abs/1906.03474
[57] Y.-H. Lin *et al.*, "Performance impacts of analog ReRAM non-ideality on neuromorphic computing," *IEEE Trans. Electron Devices*, vol. 66, no. 3, pp. 1289–1295, Mar. 2019.
[58] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," 2018, *arXiv:1801.04381*. [Online]. Available: http://arxiv.org/abs/1801.04381
[59] Y. Wu, V. Sze, and J. S. Emer, "An architecture-level energy and area estimator for processing-in-memory accelerator designs," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, 2020, pp. 1–3.
[60] K. Guo *et al. Neural Network Accelerator Comparison*. Accessed: Jul. 12, 2020. [Online]. Available: https://nicsefc.ee.tsinghua.edu.cn/projects/neural-network-accelerator/
[61] Q. Dong *et al.*, "A 351TOPS/W and 372.4GOPS compute-in-memory SRAM macro in 7nm FinFET CMOS for machine-learning applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 242–244.
[62] R. Mochida *et al.*, "A 4M synapses integrated analog ReRAM based 66.5 TOPS/W neural-network processor with cell current controlled writing and flexible network architecture," in *Proc. IEEE Symp. VLSI Technol.*, Jun. 2018, pp. 175–176.
[63] J. Yue *et al.*, "A 65nm computing-in-memory-based CNN processor with 2.9-to-35.8TOPS/W system energy efficiency using dynamic-sparsity performance-scaling architecture and energy-efficient inter/intra-macro data reuse," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 234–236.
[64] V. Camus, L. Mei, C. Enz, and M. Verhelst, "Review and benchmarking of precision-scalable multiply-accumulate unit architectures for embedded neural-network processing," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 4, pp. 697–711, Dec. 2019.
[65] Y. Zhao, C. Li, Y. Wang, P. Xu, Y. Zhang, and Y. Lin, "DNN-chip predictor: An analytical performance predictor for DNN accelerators with various dataflows and hardware architectures," 2020, *arXiv:2002.11270*. [Online]. Available: http://arxiv.org/abs/2002.11270
[66] L. Mei, P. Houshmand, V. Jain, S. Giraldo, and M. Verhelst, "ZigZag: A memory-centric rapid DNN accelerator design space exploration framework," Jul. 2020, *arXiv:2007.11360*. [Online]. Available: https://arxiv.org/abs/2007.11360
[67] L. Yang *et al.*, "Co-exploration of neural architectures and heterogeneous ASIC accelerator designs targeting multiple tasks," 2020, *arXiv:2002.04116*. [Online]. Available: http://arxiv.org/abs/2002.04116
[68] B. Murmann. *ADC Performance Survey 1997–2020*. Accessed: Jul. 12, 2020. [Online]. Available: http://web.stanford.edu/~murmann/adcsurvey.html
[69] S. K. Gonugondla, C. Sakr, H. Dbouk, and N. R. Shanbhag, "Fundamental limits on the precision of in-memory architectures," in *Proc. ICCAD*, 2020, pp. 1–10.

Boris Murmann (Fellow, IEEE) received the Dipl.-Ing. (FH) degree in communications engineering from Fachhochschule Dieburg, Dieburg, Germany, in 1994, the M.S. degree in electrical engineering from Santa Clara University, Santa Clara, CA, USA, in 1999, and the Ph.D. degree in electrical engineering from the University of California at Berkeley, Berkeley, CA, in 2003.

From 1994 to 1997, he was with Neutron Mikrolektronik GmbH, Hanau, Germany, where he was involved in the development of low-power and smart-power application-specified integrated circuits (ASICs) in automotive CMOS technology. Since 2004, he has been with the Department of Electrical Engineering, Stanford University, Stanford, CA, where he is currently a Full Professor. His current research interests include the area of mixed-signal integrated circuit design, with a special emphasis on data converters, sensor interfaces, and circuits for embedded machine learning.

Dr. Murmann was a co-recipient of the Best Student Paper Award at the Very Large-Scale Integration (VLSI) Circuits Symposium in 2008 and a recipient of the Best Invited Paper Award at the IEEE Custom Integrated Circuits Conference (CICC) in 2008, the Agilent Early Career Professor Award in 2009, and the Friedrich Wilhelm Bessel Research Award in 2012. He served as an Associate Editor for the IEEE JOURNAL OF SOLID-STATE CIRCUITS. He served as the Data Converter Subcommittee Chair and the 2017 Program Chair for the IEEE International Solid-State Circuits Conference (ISSCC).