SAND2022-0047J

An accurate, error-tolerant and energy-efficient neural network inference engine based on SONOS analog memory

T. Patrick Xiao, *Member, IEEE,* Ben Feinberg, *Member, IEEE,* Christopher H. Bennett, *Member, IEEE,* Vineet Agrawal, Prashant Saxena, Venkatraman Prabhakar, Krishnaswamy Ramkumar, Harsha Medu, Vijay Raghavan, *Member, IEEE,* Ramesh Chettuvetty, Sapan Agarwal, *Member, IEEE,* Matthew J. Marinella, *Senior Member, IEEE*

***Abstract*—We demonstrate SONOS (silicon-oxide-nitride-oxide-silicon) analog memory arrays that are optimized for neural network inference. The devices are fabricated in a 40nm process and operated in the subthreshold regime for in-memory matrix multiplication. Subthreshold operation enables low conductances to be implemented with low error, which matches the typical weight distribution of neural networks, which is heavily skewed toward near-zero values. This leads to high accuracy in the presence of programming errors and process variations. We simulate the end-to-end neural network inference accuracy, accounting for the measured programming error, read noise, and retention loss in a fabricated SONOS array. Evaluated on the ImageNet dataset using ResNet50, the accuracy using a SONOS system is within 2.16% of floating-point accuracy without any retraining. The unique error properties and high On/Off ratio of the SONOS device allows scaling to large arrays without bit slicing, and enables an inference architecture that achieves 20 TOPS/W on ResNet50, a >10× gain in energy efficiency over state-of-the-art digital and analog inference accelerators.**

***Index Terms*—SONOS, charge trap memory, neuromorphic, neural network, analog, in-memory computing, inference accelerator**

I. Introduction

NEURAL networks are increasingly being used to solve important problems in many application domains. As these workloads become larger and more complex, a number of domain-specific architectures have been developed to address the challenges of scalable performance and energy efficiency. Digital processors that are specialized for matrix operations, such as graphics processing units (GPUs) [1], field-programmable gate arrays (FPGAs) [2], [3], and application-specific chips like Google's Tensor Processing Unit (TPU) [4], [5] are already being deployed in data centers and edge devices to process neural networks.

Accelerators based on analog computation inside memory arrays potentially yield further order-of-magnitude improvements in energy efficiency. Performing matrix computations inside a memory array exploits both analog parallelism to reduce processing energy and data locality to reduce data movement energy [6], [7], [8]. Neural network inference is a particularly attractive application for these accelerators, as it involves only a single matrix computation primitive (matrix-vector multiplication, or MVM) and does not require frequent programming of the memory devices. Many non-volatile memory technologies have been explored as candidate devices for analog inference. However, despite significant recent progress in both devices and architectures [9], [10], [11], analog systems have yet to match the precision of digital inference accelerators. Analog precision is directly affected by noise, process variations, and parasitic effects. Although many analog accelerators have demonstrated high accuracy on simple datasets like MNIST and CIFAR-10 in the presence of these errors, their viability for a more realistic benchmark like the ImageNet dataset [12] is rarely tested. Various works have proposed device-aware training as a solution for analog errors [13], [14], [15], but these methods may not always integrate easily into existing training workflows and do not always recover the full digital accuracy.

This work investigates SONOS (silicon-oxide-nitride-oxide-silicon) charge trap memory as the weight storage element for neural network inference. The SONOS device, which has been commercialized for data storage applications [16], behaves functionally as a memristor. Charge that is stored persistently in the SONOS gate stack serves as a state variable that electrostatically modulates the channel conductance. The embedded SONOS devices in this work were fabricated in a 40nm foundry process, and statistics from thousands of devices were used to characterize programming errors, retention loss, and read noise. This experimental data is used to simulate the accuracy of a SONOS accelerator on ImageNet with the ResNet50 convolutional neural network (CNN) [17], [18].

Our novel finding is that by operating the SONOS transistor in the subthreshold regime, we obtain device properties that are uniquely well matched to a universal property of neural networks. The exponential current-voltage characteristic under subthreshold operation implies that the SONOS device can be programmed to have a very high current On/Off ratio ($>10^7$) and furthermore, a conductance near zero can be programmed with nearly zero absolute error. At the same time, weights in a neural network tend to be exponentially skewed toward values with low magnitude. This combination allows the SONOS device to represent the most abundant weights in a network with the highest precision, enabling high accuracy

T. P. Xiao, B. Feinberg, C. H. Bennett, S. Agarwal, and M. J. Marinella are with Sandia National Laboratories, Albuquerque, NM, 87111 USA. e-mail: txiao@sandia.gov, m@asu.edu.

V. Agrawal, P. Saxena, V. Prabhakar, K. Ramukar, H. Medu, V. Raghavan, and R. Chettuvetty are with Infineon Technologies LLC, San Jose, CA.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

and robustness to programming errors. The high On/Off ratio further provides robustness to parasitic voltage drops even in a large array, which enhances energy efficiency.

These unique properties allow the SONOS accelerator to achieve within 2.16% of floating-point accuracy on ImageNet without any network retraining – the most accurate result to date for an analog accelerator that is based on characterized devices. We then design the circuits and architecture for a SONOS inference accelerator with a projected peak energy efficiency that exceeds 50 TOPS/W.

The remainder of this paper is organized as follows. Section II provides a brief background on analog in-memory inference accelerators. Section III describes the SONOS device and its measured properties. Section IV discusses how the SONOS device is used for MVM, and highlights the key properties that are uniquely suited for the inference application. Section V uses the device data to simulate the end-to-end accuracy of a SONOS inference accelerator on ImageNet. Sections VI and VII describe the design of the peripheral circuitry for the SONOS array and the system-level architecture of the inference accelerator, respectively. Section VIII evaluates the projeced performance, area, and energy efficiency of the accelerator on different workloads.

II. Analog In-Memory Accelerators for Neural Network Inference

Inference operations in neural networks rely heavily on MVM computations [19]. Digital inference accelerators can efficiently execute MVMs using arrays of multiply-accumulate (MAC) units, but their system-level efficiency is limited by the energy to move operands between memory and the processor [4], [20], [21]. Analog in-memory computing reduces inference energy in two important ways. First, MVMs are computed in the same arrays where the network's weights are stored, greatly reducing data movement energy. Second, individual MACs can be executed more efficiently in the analog domain, and the full MVM operation can be processed in parallel. As shown in Fig. 1, each memory cell multiplies an input element (voltage) and its stored weight (conductance) to produce a current. The voltage can be proportional to the input element or can be binary, representing one bit of the input at a time. These currents are summed along an array column using Kirchoff's current law to produce a dot product. The analog dot products are then digitized using an analog-to-digital converter (ADC) and sent to the next layer's array.

An important question in designing analog accelerators is the precision of the weight values stored in the memory cells, which ultimately determines the inference accuracy. Many memory technologies have been proposed for in-memory MVM, with varying degrees of precision: these include resistive random access memory (ReRAM) or memristors [22], [23], [24], phase change memory (PCM) [13], [25], flash memory [26], [27], [28], [29], and ferroelectric memory [30]. The weight precision is also affected by architecture-level design choices. The previously cited works map one weight to one device, or to a differential pair of devices for signed weights. Many other works use bit slicing, where a B-bit weight is mapped to S devices that are each programmed to B/S bits of resolution [6], [7], [31], [32], [33], [34]. Bit slicing can represent arbitrary precisely weights with low-resolution devices, but it does not guarantee digital precision in the MVM computation [35]. Bit slices and their aggregation also incur significant area and energy overheads.

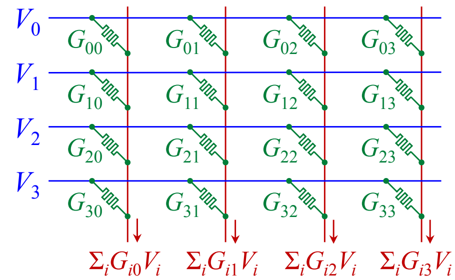

Fig. 1: Execution of an analog matrix-vector multiplication inside a resistive memory array.

The amount of analog error that can be tolerated depends on the neural network application. Accuracy on simple image recognition tasks such as MNIST [36] and CIFAR-10 [37] can tolerate relatively large errors. A more realistic machine learning benchmark is the much more difficult ImageNet dataset [12]. ImageNet networks generally require about 8 bits of precision in the weights and activations [38]. Recently, Joshi *et al.* [13] showed (via simulation) that with a PCM device, the analog errors lead to a 7.8% drop in top-1 accuracy using ResNet34 [17]. After retraining the network to regularize for the expected magnitude of device errors, the accuracy drop was reduced to 1.6% relative to the original floating-point network. While device-aware retraining is a potentially viable option for analog inference, it may be complicated to integrate with state-of-the-art training workflows. Ideally, analog inference accelerators would provide near-digital accuracy with off-the-shelf neural networks without any retraining.

III. SONOS Analog Memory

Fig. 2 shows the embedded SONOS analog memory cell used in this work, which was fabricated in a 40nm foundry process. Each cell consists of a select transistor in series with a SONOS memory device. When used as an analog memory, the internal state variable of the SONOS device is the quantity of charge Q that resides in electronic trapping sites inside the silicon nitride layer. Charge is injected or removed from the nitride via Fowler-Nordheim tunneling of electrons or holes from the silicon channel [16], and is thereafter confined by the potential barriers of both the traps and the nitride-oxide interfaces. The amount of stored charge modulates the threshold voltage V_{T} of the SONOS transistor channel, which in turn controls the drain current I_{D}. While Q represents the state of the device, this state is electrically read out using the drain current I_{D} at a fixed bias. During an MVM, the bit line (BL), source line (SL), and control gate (CG) voltages are fixed. A binary voltage on the select gate (SG) switches the select transistor between its fully on and off states, and controls whether or not current flows between the SL and BL through the SONOS device.

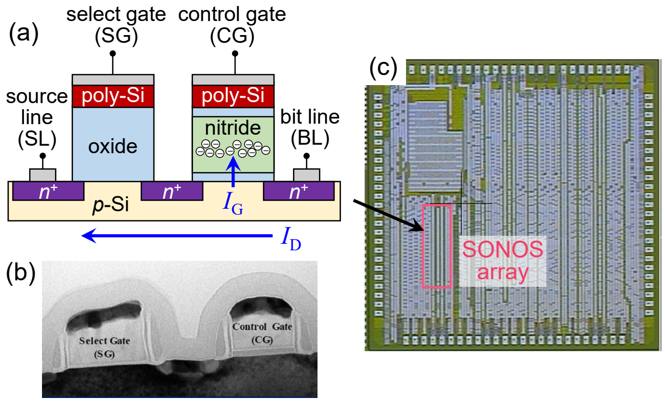

Fig. 2: (a) 40nm analog memory cell including a select transistor (left) and a SONOS transistor (right). (b) TEM cross section of the cell. (c) Die photo of the fabricated test chip with a 1024×1024 SONOS array.

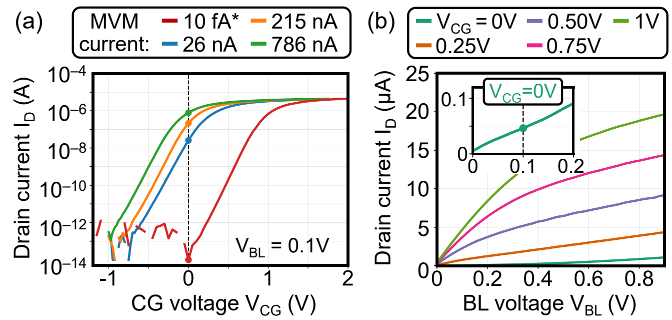

Fig. 3: (a) I_{D}-V_{CG} properties of the SONOS cell for four programmed states. The weight value is proportional to the value of I_{D} at the MVM operating point: $V_{\mathrm{CG}} = 0\mathrm{V}$, $V_{\mathrm{BL}} = 0.1\mathrm{V}$, $V_{\mathrm{SG}} = 2.5\mathrm{V}$. (b) I_{D}-V_{BL} properties for one programmed state at several V_{CG}. Inset zooms into the MVM operating point. *The current is below the measurement resolution.

A. Electrical Properties

The electrical properties of the SONOS device can be considered in analogy with a memristor [39]. Where in a metal-oxide or conductive-bridge memristor the state variable is typically the arrangement of defects or ions in the conducting channel [40], [41], the state variable in a SONOS device is the amount of trapped charge in the nitride layer Q, which lies outside the channel but electrostatically influences the channel's conductance G. This can be captured by the effect of the charge on the channel's threshold voltage:

$$I_{\mathrm{D}} = G\left[V_{\mathrm{T}}(Q)\right] V_{\mathrm{BL}} \tag{1}$$

The dynamical equation for the state variable Q is:

$$\frac{dQ}{dt} = I_{\mathrm{G}}(Q, V_{\mathrm{CG}}, V_{\mathrm{SG}}, V_{\mathrm{BL}}, T) \tag{2}$$

where T is the temperature, and V_{BL}, V_{SG}, and V_{CG} are the voltages on the BL, SG, and CG terminals, respectively, relative to the SL terminal. The rate of change of the state variable Q is controlled by the net tunneling current I_{G} into the gate stack that fills the nitride traps. I_{G} is a nonlinear function of both the state Q and the terminal voltages, all of which control the electronic band alignment between the channel and the storage layer. During device programming, large terminal voltages are applied to induce a significant Fowler-Nordheim tunneling current and a relatively large change in state $|\Delta Q|$ over a timescale of microseconds to milliseconds [16]. When the device is under a smaller voltage bias, as during an MVM or in standby mode, dQ/dt is smaller by orders of magnitude. However, the very small tunneling current can still cause charge leakage over a much longer timescale (days, weeks, or more).

Though its behavior can be expressed in terms of the memristor equations, the three-terminal SONOS transistor differs from a two-terminal memristor in that the electrical readout occurs through a separate conduction path (across the Si channel) from the change in the state variable (through the SONOS gate stack). This allows the two processes, described by Equations (1) and (2), to be more readily decoupled. The charge Q affects the channel conductance electrostatically through V_{T}, but the tunneling currents during the write process modify Q directly, with no explicit dependence on V_{T}.

Fig. 3(a) shows the measured cell current as a function of the gate voltage V_{CG} on the SONOS device, with the select transistor biased on. The I_{D}-V_{CG} characteristics are shown for four different levels of programmed charge Q in the SONOS device, where the red curve is the fully programmed (high V_{T}) state of the device. Fig. 3(b) shows the I_{D}-V_{BL} characteristic of the cell. For an MVM, we choose to operate the device in the subthreshold regime with $V_{\mathrm{CG}} = 0\mathrm{V}$ to reduce current consumption and to minimize the effect of programming errors (see Section III-B). We also use a small BL voltage of $V_{\mathrm{BL}} = 0.1\mathrm{V}$, where the cell is approximately linear (see inset) and thus G can be considered independent of V_{BL}.

When the SONOS device operates in the subthreshold regime with a fixed gate bias $V_{\mathrm{CG}} = 0\mathrm{V}$, its channel conductance is given by [42], [43]:

$$G(V_{\mathrm{T}}) = G_0 \exp\left[-\eta \frac{qV_T}{kT}\right] \tag{3}$$

where G_0 is a constant for a given device geometry and terminal bias, $\eta \approx 0.5$ is the gate efficiency, q is the electron charge, and k is the Boltzmann constant. The threshold voltage V_T changes approximately linearly with the stored charge [42]:

$$V_T(Q) = V_{T0} - \frac{Q}{C_N} \tag{4}$$

where V_{T0} is the value of V_T with $Q = 0$, and C_N is the effective capacitance between the control gate and the charge centroid of the nitride storage layer.

B. Programming Precision

A test chip was fabricated that contains a 1024×1024 SONOS array designed for analog MVM operation, shown in Fig. 2(c). A combination of process and write algorithm optimizations enable weight storage with high precision and long retention in the SONOS device. The material profile of the ONO stack was designed to minimize random dopant fluctuations, and a high-quality tunneling oxide reduced the density of interface states. A programming algorithm was developed that both programs the drain current I_{D} (measured at

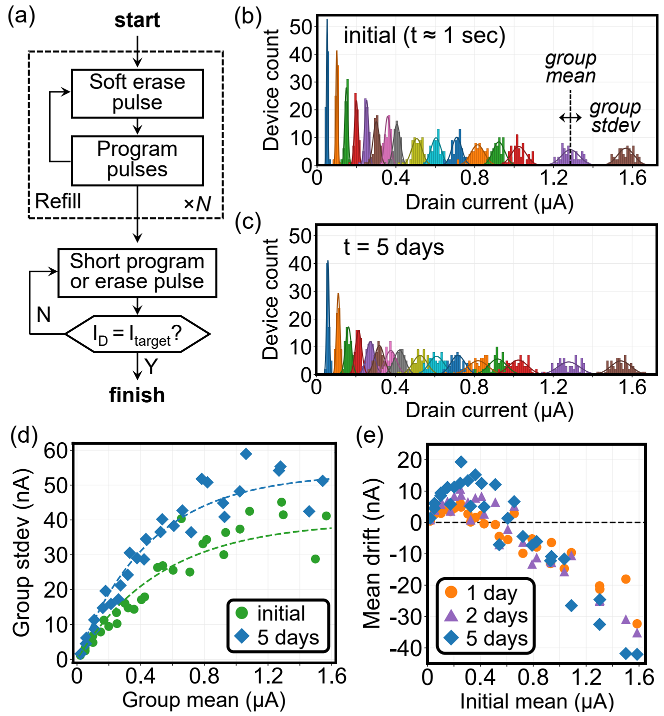

Fig. 4: (a) Algorithm for SONOS programming. (b) Distribution of cell currents just after programming. Each color is a group of 64 cells with the same target current. Each histogram bin is 10 nA wide. (c) Distribution of the same devices after five days. (d) Programming error in the SONOS current vs target current. The data are fit to saturating exponentials. (e) Drift in the distribution means of SONOS currents.

the MVM bias) precisely around a target current and optimizes data retention. Retention depends on the energy spectrum of the populated electronic traps in the nitride: shallow traps near the band edges have less retention than deep traps near the midgap, which provide greater charge confinement [44].

Fig. 4(a) schematically shows the programming procedure, which begins with several refill cycles that are performed for retention: in each cycle, the device is erased, then programmed via short pulses to a verified level. The soft erase pulse is designed to empty electrons from shallow traps, which are de-trapped more rapidly, but not deep traps. The subsequent program pulses fill both types of traps, but over multiple refill cycles a progressively larger portion of electrons are stored in the deep traps [45], [46]. After the refill cycles, a write-verify scheme is used to approach the target current via short program and erase pulses. Programming at 85°C further promoted the ionization of shallow traps but not the deep traps [47]. Cells were programmed one row at a time. After a cell's target current is reached, it is biased to prevent write disturb.

SONOS cells in the array were programmed to various target levels, and the current distributions are shown in Fig. 4(b) just after programming, and in Fig. 4(c) five days after programming. The currents are measured at the MVM operating bias of $V_{CG} = 0$V and $V_{BL} = 0.1$V. Each color corresponds to a group of cells programmed to the same target. The histogram for each group can be fit to a normal distribution whose standard deviation is the expected programming error.

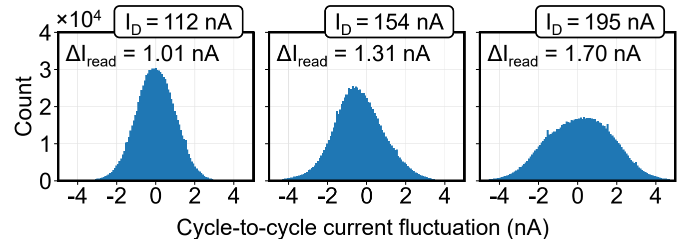

Fig. 5: The current of three SONOS cells is sampled 960,000 times at 2.5 MHz, and the statistics of the fluctuations between samples is shown. ΔI_{read} is one standard deviation.

The relationship between the mean and error currents is shown in Fig. 4(d). An important feature of the error characteristics is that at low currents, the error increases proportionally with current. This is a consequence of the subthreshold operation of the SONOS device, expressed in Equation (3). Inserting Equation (4) into Equation (3), then differentiating with respect to the state variable, we obtain:

$$dG = \frac{q\eta}{kTC_N} dQ \times G \tag{5}$$

Equation (5) shows that for a given amount of programming error dQ on the stored electronic charge, the error in the conductance dG is proportional to G. This is an intuitive result: since the conductance is exponential with Q, a large error dQ at low conductance results in a small *absolute* error in the conductance. Fig. 4(d) also shows that at drain currents above about 1 μA, and thus lower V_T, the error increases sublinearly because the device is transitioning out of the subthreshold regime. The proportionality between dG and G at low conductance has significant implications for neural network error tolerance. This will be discussed in Section IV.

C. Charge Leakage and Read Noise

The weights that are stored in the SONOS devices change over time due to leakage of charge from the nitride layer, caused by thermal de-trapping and trap-assisted tunneling through the oxide into the silicon substrate [48]. Fig. 4(e) shows that leakage causes the means of the distributions in Fig. 4(d) to drift. We find that compared to drift, the inference accuracy is more strongly affected by the increase in the widths of the distributions over time, shown in Fig. 4(c) and (d). The increase in error is due to the random, thermal nature of the leakage process. The variable rates of drift from device to device widen the current distributions over time.

Noise in the cell current also perturbs the weight value after programming. This noise is characterized by rapidly sampling the currents of individual devices at 2.5 MHz. The statistics of cycle-to-cycle fluctuations in the current of cells programmed to three different states is shown in Fig. 5. The noise has a $1/f$ power spectral density, and at the measured levels, the noise standard deviation is a small, fixed proportion (∼0.87%) of the drain current I_D. Both of these properties suggest that the dominant noise component is flicker noise, which originates from random trapping/de-trapping of channel carriers at the Si/oxide interface or from random mobility fluctuations [49]. Fig. 5 also shows that $\sim 10^6$ read cycles caused no persistent change in the weight stored by the device.

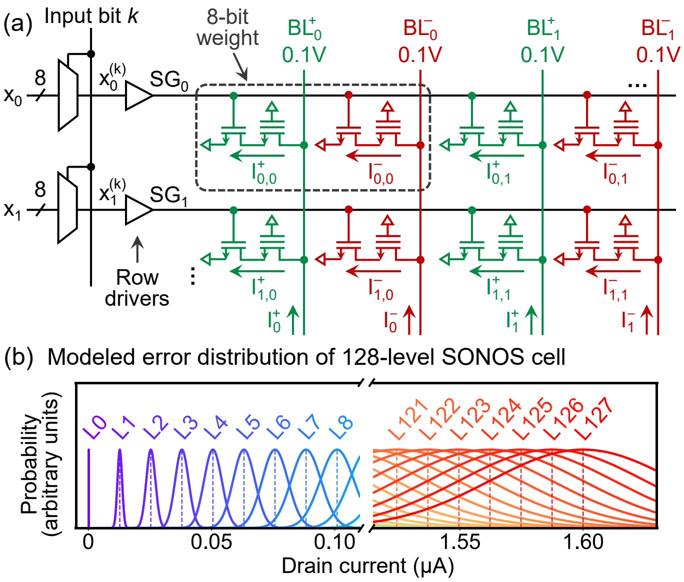

Fig. 6: (a) Analog MVM in a SONOS array, with 8-bit weights and 8-bit inputs. (b) Modeled error distribution of a SONOS cell that is used to approximately represent a 7-bit value.

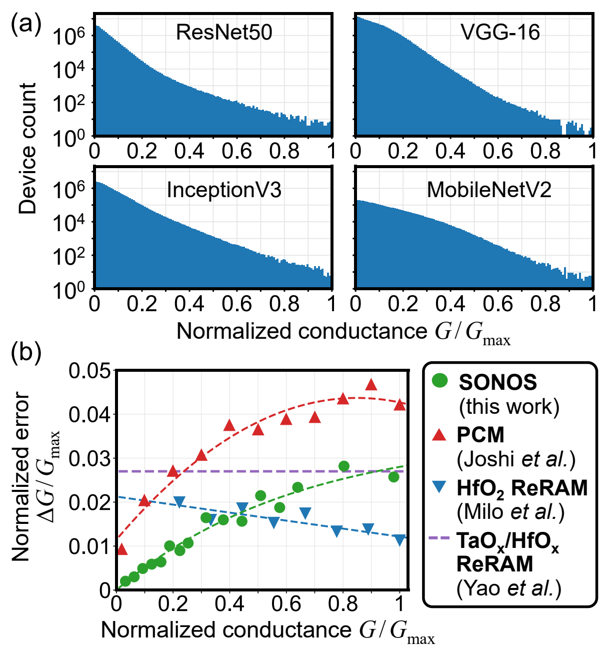

Fig. 7: (a) Distribution of SONOS conductances in systems that implement four popular ImageNet CNNs. (b) Comparison of the error normalized to the maximum utilized conductange of SONOS devices and published PCM [13], HfO_2 memristors [52], and TaO_x/HfO_x memristors [24]. The distributions just after programming are used if available. A parabolic fit is shown for PCM and a linear fit is shown for HfO_2 devices.

IV. Error-tolerant MVM in a SONOS array

In this section, we argue that the SONOS device properties are uniquely well suited to maximize the accuracy and energy efficiency of analog neural network inference. The accuracy benefit will be quantified in the next section.

The SONOS array used for in-memory MVM is shown in Fig. 6(a). Each input value x_i is applied one bit at a time to its corresponding row in the array. A row driver circuit raises the digital high logic level to the 2.5V required to turn on the select transistor. The SLs are held at 0V. Each cell is either connected to the SL when activated (high input bit) or left floating (low input bit). The BLs are held at 0.1V by the current readout circuits (not shown) that provide a virtual ground node. The input bits are applied sequentially and the partial results are accumulated as described in Section VI. Since there is only one value of V_{BL} for an activated cell, the cell's current and conductance are equivalent representations of the stored value.

A signed weight W_{ij} is mapped onto the difference in current of two SONOS cells: $I_{ij}^{+} - I_{ij}^{-}$. Currents from the positive and negative cells are accumulated on separate BLs, and these currents are subtracted by analog peripheral circuitry as described in Section VI. The weight's magnitude is stored in one of the two cells, while the other cell is left in the minimum current state, depending on the sign. To handle ImageNet networks, we choose to map 8-bit weights to each differential pair so that each device encodes a 7-bit magnitude. Note that based on the error properties in Fig. 4(d), the device does not have enough precision to have 128 equally-spaced and well-separated current levels across its entire dynamic range. Thus, we use the SONOS device as an *approximate* 7-bit memory: the programming circuitry targets 128 digital levels, but due to programming errors, the error distributions for nearby levels can overlap. These levels are statistically resolvable over a large number of devices, but there is no guarantee that a single device will be programmed to the correct level.

Fig. 6(b) shows the modeled error distribution of the 128 current levels, equally spaced between I_{min} and $I_{\mathrm{max}} = 1.6$ μA ($G_{\mathrm{max}} = 16$ μS). The I_{min} level is implemented by a state with a much higher threshold voltage than the others, shown by the red curve in Fig. 3(a). The value of I_{min} is below the measurement resolution, but the current On/Off ratio $I_{\mathrm{max}}/I_{\mathrm{min}}$ is at least 10^7. The state-dependent widths of the distributions in Fig. 6(b) are obtained from a saturating exponential fit to the data in Fig. 4(d) to interpolate between the measured points. At the lowest current levels, the distributions are well-separated, indicating high precision. The highest current levels have significant overlap and thus high uncertainty. The highest level (I_{max}) maps to the weight with the largest absolute value in a given layer's weight matrix.

Fig. 7(a) shows the global distribution of conductance values, normalized to G_{max}, for systems that map four popular ImageNet CNNs. In every case, the conductances are heavily skewed toward zero, which results from two factors. First, most of the network weights have low magnitude; this is a very common feature of neural networks [50], [51]. Second, weights are mapped to conductances in proportion to their magnitude; this is a consequence of differential conductance mapping and the very high On/Off ratio of the SONOS device, which allows zero-valued weights to map to near-zero conductances.

The skewed conductance distributions in Fig. 7(a) combine well with another property of the SONOS cell: the proportionality of the conductance error dG to the conductance G when the conductance is low. This was shown in Fig. 4(d) and is reprised in Fig. 7(b) in normalized units (green). Most importantly, the lowest conductance levels have nearly zero error. *Thus, the most frequently used weights in the neural network are matched with the most precise conductance levels in the cell.* Subthreshold operation of the SONOS device is the key: it enables both a high On/Off ratio and state-proportional error, as shown by Equation (5). The vast majority of neural network weights have nearly zero error, providing intrinsic error tolerance for inference.

Fig. 7(b) also contrasts the SONOS device with three recent works with published programming error distributions, which were all used for neural network inference: the PCM device in Joshi *et al.* [13], the HfO_2 memristors in Milo *et al.* [52], and the TaO_x/HfO_x memristors in Yao *et al.* [24]. The programming precision depends both on the device and the programming scheme. The key distinctive property of the SONOS device is that the programming error approaches zero as the weight approaches zero. As mentioned above, this is important since values near zero are the most abundant weights in a neural network. The PCM devices are the most similar among the compared works, but its error in the lowest state remains at $\sim$1% of the utilized conductance range.

The devices in Fig. 7(b) are fairly similar in their maximum conductance: $G_{\max}$ is 16 µS for SONOS, 25 µS for PCM, 20 µS for the TaO_x/HfO_x memristor, and 225 µS for the HfO_2 memristor. However, because it operates in the subthreshold regime, the SONOS device has a much higher On/Off ratio ($>10^7$) compared to the other devices; the On/Off ratio is between 10 and 100 for the two memristors and $\sim$100 for PCM based on the error. The On/Off ratio is important because with differential cells and a heavily skewed weight distribution, the majority of cells on a BL will use the minimum or near-minimum conductance level. If the On/Off ratio is insufficient, these small cell currents can accumulate to become a large BL current. For a given array size, larger BL currents require more power-hungry peripheral circuits and also increase the effect of parasitic IR voltage drops in the array, which can degrade accuracy as we show in Section V.

The large On/Off ratio of the SONOS cell ensures that zero-valued weights conduct negligible current, reducing accumulated BL currents. Similarly, the thick gate oxide and large threshold voltage (>2V) of the select transistor ensure that non-activated cells also draw negligible current during an MVM. This allows an increase in the array size (number of rows) while keeping the BL currents relatively small, and without incurring MVM accuracy losses due to BL parasitic resistance. Since the energy consumption of an analog MVM tends to be dominated by peripheral circuits, a larger array directly increases energy efficiency by amortizing these energy costs over more MAC operations.

V. Inference accuracy evaluation

This section uses the characterized device properties in Section III and array design in Section IV to simulate the accuracy of a SONOS-based analog inference accelerator on a large-scale ImageNet CNN.

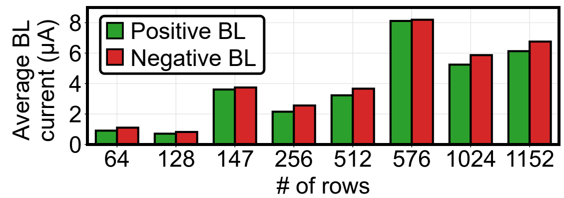

Fig. 8: BL current vs SONOS array size for a system that implements ResNet50, based on inference simulation data on ImageNet. Each value is an average over all arrays of the same size, and over all MVMs in 30 inference operations. The current does not increase monotonically with array size due to layer-specific weight and activation distributions.

A. Neural Network Benchmark and Accuracy Simulator

To emulate a realistic machine learning application, accuracy is evaluated using the ResNet50-v1.5 network on the ImageNet dataset [12]. We use the reference implementation of this network from the MLPerf Inference Benchmark [17], [18], which has also been used to benchmark digital inference accelerators. We refer to this network henceforth as ResNet50. All weights are quantized to 8 bits before being mapped to hardware. Weights are not retrained or modified (besides quantization) before being transferred to the arrays.

The accuracy of a SONOS inference accelerator is predicted using CrossSim [53], an end-to-end neural network simulation tool that models the various analog errors in in-memory MVM. For every SONOS cell, a random programming error ΔI_{prog} is added to the target current. This error is sampled from a normal distribution, whose target-dependent standard deviation is derived from the data in Fig. 4(d). Leakage is modeled by including the time-dependent standard deviation and drift (Fig. 4(e)) in the current distributions as described by Bennett *et al.* [54]. Read noise is modeled by sampling a new error ΔI_{read} based on Fig. 5 that adds to ΔI_{prog} on every MVM. Array parasitic resistance and ADC quantization are modeled as described below. Digital operations such as rectified linear (ReLU) activations are assumed to be error-free.

Convolutions are unrolled into a sequence of sliding window MVMs, executed on arrays of size $K_x K_y N_{\text{ic}} \times N_{\text{oc}}$ as described by Shafiee *et al.* [7] ($K_x \times K_y$ is the 2D filter size, N_{ic} and N_{oc} are input and output channel dimensions). During inference, batch normalization layers are folded into the matrix of a preceding convolution to reduce the digital processing overhead between layers [38]. Bias weights are stored and added digitally, as they can lie in a different numerical range from the weights that are stored in the array.

B. Parasitic Resistance Effects

As described in Section IV, the maximum array size is limited by the accumulated current on the rows and columns during an MVM (though programming steps can introduce additional constraints [55]). The larger these currents, the larger the parasitic voltage drops across the array interconnects. This effect is negligible on the SG lines of the SONOS array since the transistor gates draw almost no current. However, the effect can be significant on the BLs. Parasitic voltage drops cause the

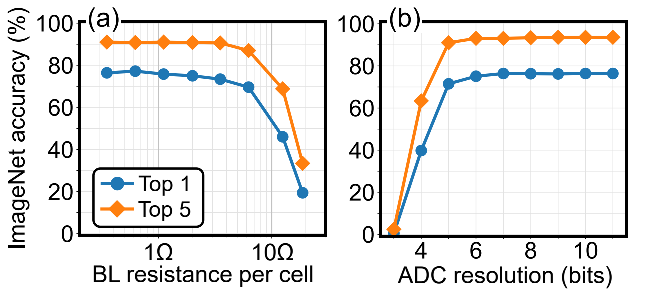

Fig. 9: (a) ImageNet accuracy (500 images) with ResNet50 vs BL parasitic resistance, (b) ImageNet accuracy (1000 images) vs ADC resolution. All results assume 1152 maximum rows. SONOS errors and noise are not included.

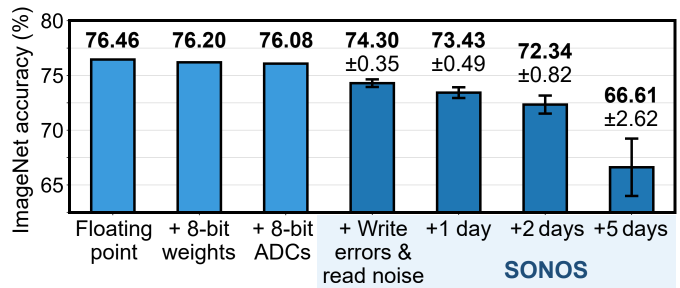

Fig. 10: ImageNet top-1 accuracy (50,000 images) using ResNet50. SONOS arrays have at most 1152 rows.

BL terminal of the cells to deviate from 0.1V in a nonuniform manner across the array, distorting the weight matrix.

Fig. 8 shows the average simulated BL currents for differently sized SONOS arrays that map the layers of ResNet50. A maximum of 1152 rows is assumed; smaller matrices use fewer rows, while a matrix with more than 1152 rows is partitioned over multiple arrays. Even with 1152 cells per column, the average BL current is only several times larger than the maximum cell current of 1.6 μA. This results from the skewed distribution of weight values in the neural network and the high On/Off ratio of the SONOS device. Furthermore, Fig. 8 shows the BL currents for the least significant bit of the input; for the most significant bits, which are weighted more heavily in the result, the currents are much lower. This is because activations (like weights) are also strongly skewed toward low values in neural networks [51], causing the higher input bits to be sparse. All of these factors help reduce the impact of parasitic resistance on accuracy [56].

For computational tractability, parasitic resistance effects are modeled by treating the SONOS device as a linear resistor around $V_{\mathrm{BL}} = 0.1$V, which is a good approximation based on Fig. 3(b). A circuit simulation is performed for each input bit; eight per MVM. Fig. 9(a) shows the accuracy on a subset of ImageNet versus the BL resistance between two adjacent cells. The accuracy begins to fall for a parasitic resistance larger than 1Ω, which is ∼60,000 times smaller than the minimum resistance of a SONOS cell. A unit cell resistance smaller than 1Ω can readily be achieved by metal interconnects in our 40 nm process, indicating than an array with 1152 rows is not limited by parasitic resistance effects.

C. ADC Resolution

Like the array size, the ADC resolution is a critical system parameter that affects both accuracy and energy efficiency. To obtain high accuracy at the minimum feasible resolution, the ADC's input range must be calibrated to minimize quantization and clipping errors [38]. As described in Section VI, the final analog output after the accumulation of all input bits is a voltage V_{out}, which is then digitized. Voltages outside the ADC's range are clipped to the lowest or highest ADC level.

To calibrate the range of each layer's ADC, we first collect statistics (via simulation) on each layer's output voltages using 500 ImageNet images. For every layer, the range (V_{min}, V_{max}) is then found that minimizes the error $\epsilon = ||\vec{V}_{\mathrm{out}} - \vec{V}_{\mathrm{Q}}||_1$ over the collected output voltages $\vec{V}_{\mathrm{out}}$. $\vec{V}_{\mathrm{Q}}$ is obtained by clipping and quantizing $\vec{V}_{\mathrm{out}}$ to N bits in this range, where $N = 12$ was empirically found to be optimal. If a matrix is partitioned across multiple arrays (i.e. has more than 1152 rows), they share the same ADC limits.

Fig. 9(b) shows the accuracy on a subset of ImageNet versus ADC resolution, after every layer's ADC range is calibrated. There is very little loss in accuracy with an ADC resolution of 8 bits or more. Based on this result, we use an ADC resolution of 8 bits across the full accelerator. After digitization, the dot products from different matrix partitions are added (if necessary), the biases are added, and activation and/or pooling functions are applied. These final activation values are quantized again to 8 bits so that they can be passed as inputs to the next layer's analog core. The ranges of these 8-bit activations are optimized using the same method as above. Section VI-B and VII will describe how the calibrated ADC and activation ranges are encoded in hardware.

D. Accuracy of the SONOS Accelerator

Fig. 10 shows the accuracy of ResNet50 on the full ImageNet test set when deployed on the SONOS accelerator. The various circuit- and device-related non-idealities are added cumulatively. Cell programming errors are re-sampled ten times to obtain the variance in accuracy. The programming errors (Fig. 4) have a much larger effect on accuracy than the read noise (Fig. 5). For computational tractability, parasitic resistance is not included; based on Fig. 9(a), its effects on accuracy are negligible using our process.

Just after programming the weights, the SONOS accelerator attains an average top-1 ImageNet accuracy of 74.30%, which is a 2.16% loss relative to floating-point. Table I shows that this is a significant improvement over a recent simulation result based on characterized PCM devices, the only other device that has been benchmarked on ImageNet. The very low error of the SONOS cell at low currents leads to a much lower expected error for a given neural network, and this is responsible for its higher inference accuracy. The cell errors are low enough on average that their effect remains small even when they accumulate over as many as 1152 rows. The higher precision and intrinsic error tolerance of the SONOS device reduce the need for specialized network re-training prior to deployment in the proposed analog hardware, enabling greater compatibility with state-of-the-art training workflows.

TABLE I: Inference accuracy comparison of technologies

	PCM [13]	SONOS
Process node	90 nm	40 nm
Expected weight error in ResNet50 (% of $G_{\max}$)	1.42% (0.35 µS / 25 µS)	0.18% (0.028 µS / 16 µS)
Evaluated network	ResNet34 (ImageNet)	ResNet50 (ImageNet)
Floating-point accuracy	73.20%	76.46%
In-accelerator accuracy (no re-training)	65.41% ± 3.6% (−7.79%)	74.30% ± 0.35% (−2.16%)
In-accelerator accuracy (with re-training)	71.62% ± 0.4% (−1.58%)	–

The accelerator gradually loses accuracy due to charge leakage from the SONOS cells, as shown in Fig. 10. To restore high accuracy, the weights may need to be refreshed every one to five days from a reference, depending on application needs. Device robustness to 100,000 write cycles is sufficient for ten-year operation at this refresh rate. The high-voltage programming and erase pulses used for SONOS programming and erase pulses can introduce defects in the tunnel oxide over time, which has been observed to shift the fully programmed/erased V_T states and gradually reduce data retention [57]. The effect of 100,000 write cycles has been shown to be minimal on 40nm digital SONOS memory technology [16], but analog states are more sensitive, and it is possible that gradually more frequent refreshes are needed over the system's lifetime to maintain high accuracy.

VI. An energy-efficient SONOS MVM core

This section and Section VII describe the design of an end-to-end CNN inference accelerator based on SONOS arrays. This section focused on the analog MVM core, which consists of the SONOS array and its peripheral circuitry. The core takes 8-bit digital inputs, executes an analog MVM with its stored 8-bit weights, and produces 8-bit digital outputs. Circuit components are designed and simulated in the 40nm foundry process used for the embedded SONOS arrays.

A. Row Peripheral Circuits

Fig. 11 shows the circuit that supplies the 8-bit inputs of an MVM. One bit at a time of the input is selected, and a row driver converts the digital high logic level to the 2.5V required to turn on the select transistor. To handle both signed inputs and signed weights, the array in Fig. 6 must use four cells per weight, rather than two. In this case, two cells implement the weight W, and the other two cells on a second row implement its inverse $-W$. The signed input logic, shown in violet in Fig. 11, drives one of the two rows based on the input sign bit. In CNNs, the use of ReLU activations means that typically only the first layer has signed inputs, and the other layers can be configured to use two cells per weight to save area.

B. Bit Line Integrators

Fig. 12 shows the circuitry that is connected to each pair of BLs in Fig. 6. The first circuit block in Fig. 12 is the integrator, which subtracts the positive and negative BL currents, integrates the difference on a capacitor, and accumulates the partial results for different input bits. The positive BL and negative BL are each connected to a current conveyor (CCII) with a current gain of -1 and $+1$, respectively, from port X to Z. The two currents are subtracted at the output node to implement the differential conductance representation of weights. The CCII also acts as voltage follower between X and Y. This is used to maintain a virtual ground on each BL, ensuring a proper voltage bias on the SONOS cells. A CCII design with low input resistance is used to keep the BL voltage as close as possible to 0.1V [58], [59].

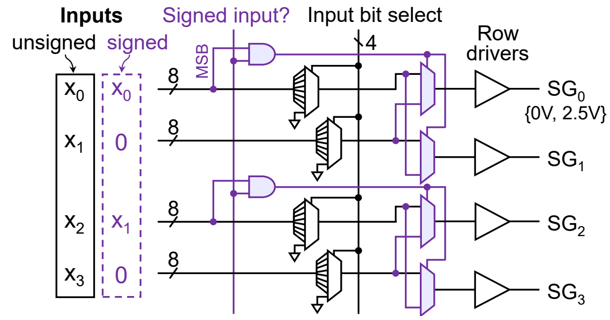

Fig. 11: Row peripheral circuitry. For unsigned inputs, the elements in violet can be ignored. For signed inputs, a control unit ensures that they are applied to every other input port.

The difference current is integrated on a 0.5 pF capacitor, producing a voltage (INT phase in Fig. 12). The current is integrated for 10 ns per input bit to overcome noise and circuit transients. Analog partial results from different input bits are accumulated using the successive integration and rescaling (SIR) technique [60]. After integrating the BL currents, the capacitor is disconnected from the BLs and is discharged to a second identical capacitor to halve its voltage (DIV). The capacitor is then re-connected to the BLs and the current from the next input bit is integrated on the accumulated charge. After all bits have been integrated, the capacitor is connected to the next stage (OUT). Before executing the next 8-bit MVM, the integrating capacitor is discharged (RESET). SIR effectively performs the shift-and-add aggregation of input bits in the analog domain, and reduces the ADC overhead by 8× as it requires only one ADC conversion for all eight bits.

The integrated voltage is then scaled by a tunable-gain amplifier to properly place the MVM result within the range of the ADC. This step is necessary to obtain high accuracy, as explained in Section V-C. The tunable gain is implemented using an operational amplifier in a non-inverting configuration, with a programmable SONOS device acting as a feedback resistor. The required gain can be found offline and does not change during inference.

The final voltage output is stored on a capacitor. Analog capacitive double buffering is used to concurrently process two MVMs within the core: one in the analog MVM and integration step, and one in the ADC step (see below). While the ADC digitizes the analog MVM result that is stored on one capacitor, the analog result of the next MVM is written to the other capacitor.

MVM accuracy can be degraded by variation and offset in the integrator components. In particular, errors in the integrating capacitors and operational amplifiers can cause MVM

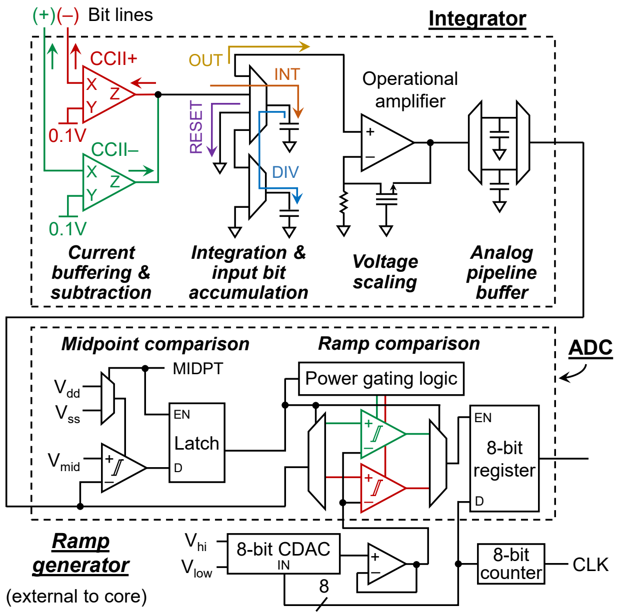

Fig. 12: Peripheral circuitry connected to each pair of bit lines. The lower three blocks implement the ADC.

outputs on different BLs to be scaled by slightly different factors, distoring the results. These effects can be partially corrected. The programmable feedback resistor can be separately calibrated on each BL pair to adjust the amplifier gain to compensate for capacitance variations. Offsets in the current conveyors, amplifiers, or comparators can be compensated with an additional row of calibrated SONOS devices that inject fixed currents onto the BLs [58].

C. Analog-to-Digital Converters

To digitize the array's analog outputs with high speed, low area, and low energy, the core uses an 8-bit parallel ramp ADC, shown in Fig. 12 after the integrator. The ADC uses a voltage ramp between two reference voltages; this is generated by passing an 8-bit, 1 GHz digital counter signal into a capacitive digital-to-analog-converter (CDAC), which produces a 256-level staircase voltage ramp. Every integrator's output voltage is then simultaneously compared to the shared ramp. On the clock cycle when the two signals are equal, the comparator switches and the value of the 8-bit counter is latched in the output register. The comparator, whose design is based on Marinella *et al.* [58], uses internal positive feedback to obtain a 1 ns response time across the full range of input voltages. By the end of the 256 ns ramp, the 8-bit digital dot products are available for the full MVM.

To maximize the ADC's voltage resolution, the ramp must start from the negative voltage rail or end at the positive rail. Two comparators are combined that together provide rail-to-rail input dynamic range: one with an NMOS input stage (green) and one with a PMOS input stage (red). To reduce static power consumption, one of the two comparators is gated off during the ramp depending on the input value. This is decided by a third comparator (black), which compares the input to the ramp midpoint and stores the result in a latch.

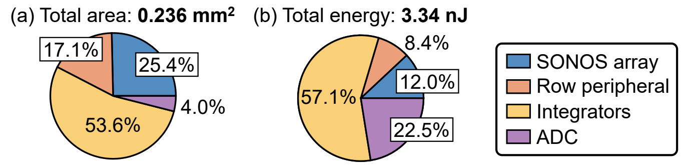

Fig. 13: Breakdown of (a) area and (b) energy consumption for a SONOS core implementing an 1152×256 matrix.

This third comparator is gated on for 2 ns (MIDPT) prior to the ramp.

D. Core Energy Efficiency and Area

Fig. 13 shows the breakdown of the core's energy and area among its components for an 1152×512 SONOS array, which supports up to an 1152×256 MVM (576×256 if inputs are signed). These dimensions were chosen to maximize the number of rows while also balancing utilization by small and large matrices in a typical CNN. An array with 1152 rows (=3×3×128) is optimally utilized by CNNs that mostly use 3×3 convolutions, such as ResNet50 and VGG-16.

Energy consumption is computed based on SPICE simulations and metal interconnect properties of the 40nm process. The energy of the array and row drivers is adjusted for the average SONOS currents and the average values of the input bits when running ResNet50 on ImageNet, obtained by simulation. The ramp generator uses the CDAC and reference voltage buffer designs in Kull *et al.* [61] and the digital counter design in Baert *et al.* [62], with properties scaled to the technology node, supply voltage, and 8-bit DAC resolution. The core area is a sum of circuit block areas rather than a physical layout; for components other than the SONOS array, 25% of the circuit area is assumed to be wiring, which is typical for this process.

The integrators make up the largest share of both the area and energy of the core, since SIR greatly reduces the energy cost of the ADCs. The SONOS core achieves a peak energy efficiency of 177 TOPS/W (11.3 fJ/MAC) for a 1152×256 MVM, which fully utilizes the array and peripheral circuits. Since the energy is dominated by the peripherals, the efficiency is reduced for smaller matrices that use fewer rows, where the BL circuit costs are amortized over fewer operations.

VII. SONOS Inference Architecture

The SONOS inference architecture is hierarchically organized to support large-scale CNN processing. Several MVM cores implementing the same layer are combined with activation memory and digital processing elements to form a *tile*. To implement the full neural network, tiles are interconnected by a concentrated mesh network similar to prior work [7], [31], [32]. Weight data is kept stationary inside the SONOS arrays during inference, while activation data is moved within and between tiles. Computation is pipelined across the system to allow different tiles to work concurrently on different computations within a layer, different layers within the same inference operation, or on separate inference operations. This work focuses on CNNs that use ReLU activations, which are typical of models used for computer vision [18].

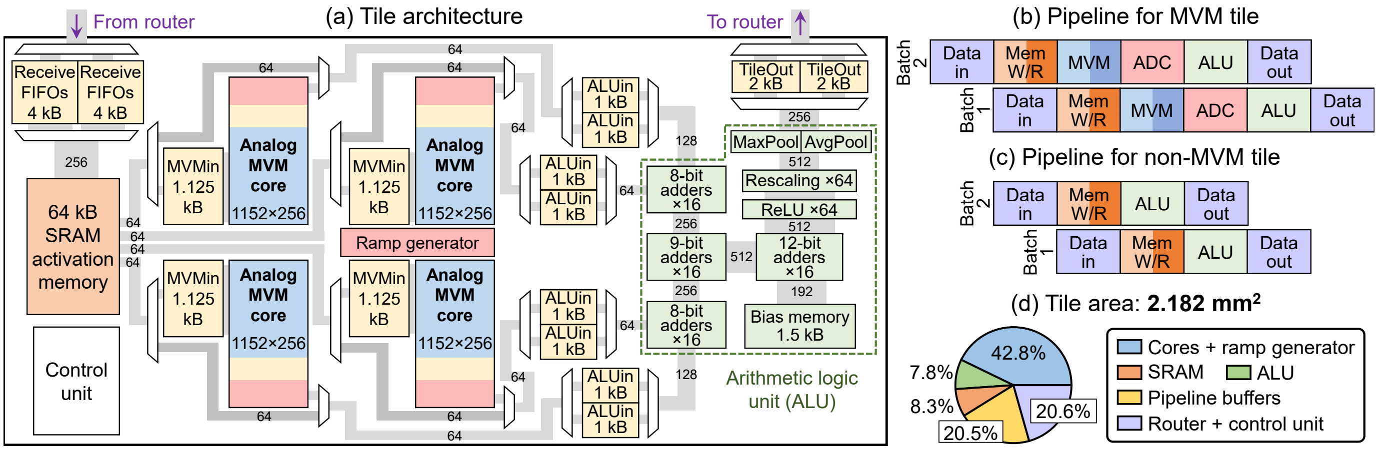

Fig. 14: (a) Tile architecture, (b-c) dataflow pipeline for MVM/non-MVM tiles, (d) Breakdown of tile area.

A. Inference Pipeline and Tile Design

The accelerator uses a uniform and modular tile design, shown in Fig. 14(a), which flexibly implements different layer types. MVM layers, including convolutions and fully-connected layers, are executed mostly inside the analog cores. Non-MVM layers such as pooling and element-wise addition are processed using the tile's digital arithmetic logic unit (ALU). MVM layers and non-MVM layers share the same tile architecture but follow different datapaths within the tile. Computation is pipelined through the tile's hardware components to maximize throughput. Fig. 14(b) and (c) show the separate dataflow pipelines for MVM and non-MVM layers, respectively. We refer to the length of one pipeline stage as a machine cycle, which is set to 295 clock cycles at 1 GHz.

The colored hardware blocks in Fig. 14(a) correspond to the colored pipeline stages in Fig. 14(b) and (c). To enable concurrent execution of different pipeline stages within a tile, double buffering is used in the pipeline registers (yellow blocks). In a given machine cycle, one buffer is populated by outputs from the preceding stage while the other holds inputs that are read by the next stage. Fig. 14(d) shows the tile area breakdown, where the control unit is not designed but conservatively estimated to take up $\sim$4% of the area.

The dataflow pipeline consists of the following stages:

1) Data in: 8-bit inputs are received through the interconnection network and placed in 32 parallel FIFO queues. Each FIFO is connected to the write port of one of 32 banks in the local SRAM memory. Inputs are assigned to FIFOs using five bits from the channel number part of its address.

2) Memory write/read: In the first half of the machine cycle, activations are written from the FIFOs to their corresponding SRAM banks. At most 4608 values (4.5 kB) can be written to SRAM in this stage, equal to the maximum number of inputs consumed by the four cores in one machine cycle. In the second half-cycle, activations are read from SRAM and loaded into the MVMin buffers of each core. If the same activation is needed at multiple row positions, it is copied during the Data In stage to multiple banks. In a non-MVM tile, activations are read from memory to the ALUin buffers, bypassing the MVM and ADC stages.

3) MVM: In the first half-cycle, each core reads the inputs from the MVMin buffers and executes an 8-bit analog MVM (all input bits integrated). In the second half-cycle, the MVMin buffers are disconnected from the cores and new values are loaded from SRAM. During this time, the core performs voltage scaling of the analog dot products and stores the results in its internal analog pipeline buffer.

4) ADC: The analog MVM results are converted to 8-bit digital values, which requires 258 clock cycles. In the remainder of the machine cycle, the values are transferred from each core's 256 output registers to the ALUin buffers.

5) ALU: Non-MVM computations are performed in this stage using the ALU on operands fetched from the ALUin buffers. The ALU receives up to 64 total inputs simultaneously from four ALUin buffers and has a worst-case delay of less than 4 ns. This allows a new set of inputs to be processed every four clock cycles. ALU components can be bypassed if not used, and the final results are stored in the TileOut buffer.

In MVM tiles, the ALU sums the results from multiple cores (if needed), adds the bias values, and applies the ReLU function. The same adders can be used by element-wise addition layers. The MaxPool and AvgPool units are implemented with trees of adders and digital magnitude comparators. Where possible, pooling is done in the ALU stage of an MVM tile rather than in a dedicated pooling tile.

Though the ALU takes 8-bit operands, the activation values produced by the ALU can have up to 13 bits of resolution. These values must be reduced back to 8-bit activations that can be processed by the next layer's analog cores. The numerical ranges of these 8-bit activations are calibrated offline as described in Section V-C. Rescaling to these calibrated ranges is accomplished first via integer multiplication by a layer-specific scaling factor, followed by a bit shift to implement a power-of-two division. The relevant eight bits are then extracted from the result.

6) Data Out: 8-bit outputs in the TileOut buffer are routed to the destination tile. This stage coincides with the Data In stage of the destination tile.

B. Convolution Mapping and Input Reuse

As described in Section V-A, a convolution is unrolled into a set of parallelizable sliding window (SW) MVMs. In a CNN, the early layers often have small weight matrices that can be replicated several times within an array, allowing the parallel execution of multiple SWs in a core. Later layers have larger matrices that may need to be partitioned over multiple cores. To increase throughput at the expense of area, weights can be physically replicated across multiple cores and/or tiles that process different SWs in parallel [7].

To reduce SRAM accesses and save energy, two methods of input reuse are leveraged for convolutions with $K_x K_y > 1$, adapting techniques previously proposed by Li *et al.* [34]. First, in early layers where a single array can concurrently execute multiple MVMs, the replicated weights are distributed over the array such that a single input (SG line) can be shared by a contiguous block of up to $K_x K_y$ SWs. Second, for cores that compute one MVM at a time, inputs can be reused by adjacent SWs that are processed in successive cycles. Between MVMs, reused activations are shifted internally within the MVMin buffers using shift registers, and only new activations are retrieved from memory. Reuse is maximized by allocating a long unbroken sequence of SWs to an array.

VIII. Performance, energy, and area evaluation

This section evaluates the performance, energy efficiency, and area of the proposed SONOS analog inference accelerator.

A. Architecture Simulator

To estimate the above metrics, a high-level simulator for the proposed accelerator was developed that models the inference pipeline in Fig. 14. Given a neural network, the simulator uses a heuristic-based approach to assign tiles to layers and to SWs within a layer. The number of tiles allocated to a layer depends on the matrix shape and a target number of machine cycles to complete all SWs, which defines the replication factor for the weights. This target latency is set equal for all layers to maintain a balanced pipeline of input images. SWs are allocated among tiles in a manner that minimizes delay between layers, maximizes input reuse, and avoids overflow of the 64 kB SRAM tile memory.

After mapping a neural network, an inference dataflow simulation is conducted at the granularity of a machine cycle. In each cycle, every tile's SRAM is checked to determine whether the inputs are available for the cores to compute their next set of allocated SWs; if so, these are fetched to the MVMin buffers. Inputs are removed from SRAM when they are not needed for any future computation.

The energy and area of the core components are estimated as described in Section VI-D. Peripheral circuits on rows or columns not utilized by a layer's matrix are gated off to save energy. The energy and delay of digital logic in the ALU are based on a standard cell library for the process. We use CACTI 7.0 [63] with a custom technology file for our 40nm process to obtain the area, access energy, access latency, and leakage of the 64 kB SRAM tile memory. Network routers are modeled using Orion 3.0 [64], [65]. For the concentrated mesh configuration, 8-port routers are used with 128-bit flits and two virtual channels.

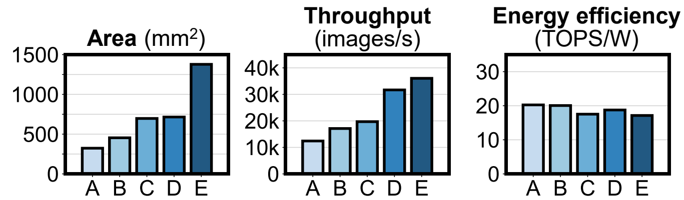

Fig. 15: Area, ImageNet inference throughput, and energy efficiency of five different system designs for ResNet50. Designs A-E use 148, 208, 296, 312, and 556 tiles, respectively.

B. System Scalability

Fig. 15 shows five accelerator designs for a single network (ResNet50) with different total tile counts. Adding more tiles generally increases the parallelism of SW computation, trading off inference throughput with area. Meanwhile, scaling the size of the system has a relatively small effect on the energy efficiency. For ResNet50, the energy efficiencies of all five designs are close to 20 TOPS/W, which is much lower than the peak core efficiency of 177 TOPS/W. This degradation results from the energy contributions of components outside the core, and from the fact that not all layers efficiently utilize the SONOS arrays.

Fig. 16(a) shows the energy breakdown among the accelerator components for Design B in Fig. 15. For ResNet50, about 32% of the energy is consumed in the core, with the SONOS array itself only consuming about 1.5%. The remainder is spent primarily on data movement, particularly in reads and writes to the tile SRAM buffer. This is in contrast to some earlier accelerators at the same ADC resolution [7], [32], where the ADC was the dominant energy consumer. The fact that the system's energy is bottlenecked by the intra- and inter-tile communication of activations is a consequence of the high efficiency of the cores, despite their sub-optimal utilization by some layers. Significant further efficiency improvements would require a reduction in data movement energy.

C. Comparison of Neural Network Workloads

Fig. 16 compares the energy per inference of three ImageNet CNNs, using accelerator designs that have roughly similar areas (shown in Fig. 17(a)). There is a significant disparity in the total energy consumption of ResNet50 and ResNet34, which are nominally similar networks that differ only by 5% in the total number of operations. The key difference is that ResNet34 lacks the 1×1 pointwise convolutions that are common in ResNet50. The 1×1 convolution is a poor fit for in-memory MVM operations since it typically has too few inputs to efficiently amortize the energy cost of the BL peripheral circuits. This leads to a higher energy cost per MAC in these layers. Therefore, the core operates closer to peak efficiency for ResNet34, and for VGG-16 for the same reason.

In addition to the three networks above, we also include the CNN in Fig. 17(a), whose topology was engineered solely to maximize the system's energy efficiency. This network has 15.2M weights, which were not trained, and a structure that is similar to that of a typical CNN. Fig. 17(b) compares the accelerator's performance and efficiency on the four networks.

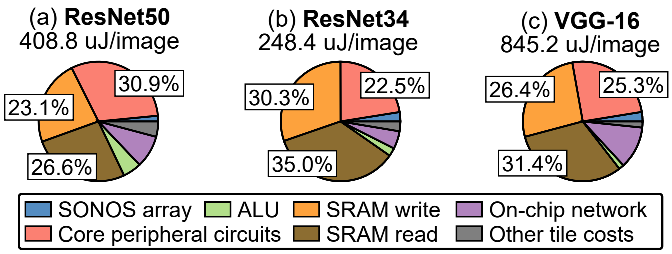

Fig. 16: Breakdown of energy consumption across hardware components for three ImageNet neural networks.

ResNet50 and ResNet34 have the longest end-to-end inference latency due to their depth, but ResNet34 has high throughput because multiple images can easily be pipelined through its simpler, more uniform residual blocks. ResNet50 utilizes the available SONOS arrays poorly, largely due to its 1×1 convolutions. ResNet34 obtains superior utilization by eliminating these convolutions, and VGG-16 improves upon this further by having very large matrices and wasting fewer arrays in non-MVM tiles (especially element-wise addition). However, the compute density of VGG-16 suffers from having very large fully-connected layers at the output of the network that consume the majority of the area but perform a minority of the operations. These tiles also consume significant static power while idle, reducing energy efficiency.

The observations above informed the design of the custom CNN. The network avoids large, infrequently used fully-connected layers; rapidly inflates the channel dimensions to more fully utilize every core; rapidly compresses the spatial dimensions to reduce the number of SWs and improve throughput; and aggressively exploits input reuse. The co-designed neural network and its hardware implementation reaches an efficiency over 55 TOPS/W, which represents close to the maximally efficient accelerator use case. This example shows that by more closely tailoring the neural network topology to the hardware, the algorithm can more fully capitalize on the large energy and density benefits of the accelerator.

D. Comparison with other accelerators

Table II compares the proposed accelerator to several deployed digital inference accelerators as well as ISAAC, a proposed memristor-based analog inference architecture [7]. We note that the performance and efficiency of analog accelerators are simulated quantities, while the digital accelerators are fully functional chips with measured performance.

Despite leveraging the efficiency of analog in-memory MVM, ISAAC provides very limited system-level efficiency gains over state-of-the-art digital accelerators. The proposed SONOS accelerator improves upon ISAAC and similar analog accelerators [6], [8], [31], [32], [33], [34] in two important ways, both enabled by the SONOS technology. First, the low SONOS programming error removes the need for bit slicing and its associated overhead. As shown in Section V, the SONOS cells can obtain high ImageNet accuracy when operated as approximate memories. Meanwhile, ISAAC encodes only two bits per cell, and requires separate peripheral circuits for every 2-bit slice. Second, the low errors and high On/Off

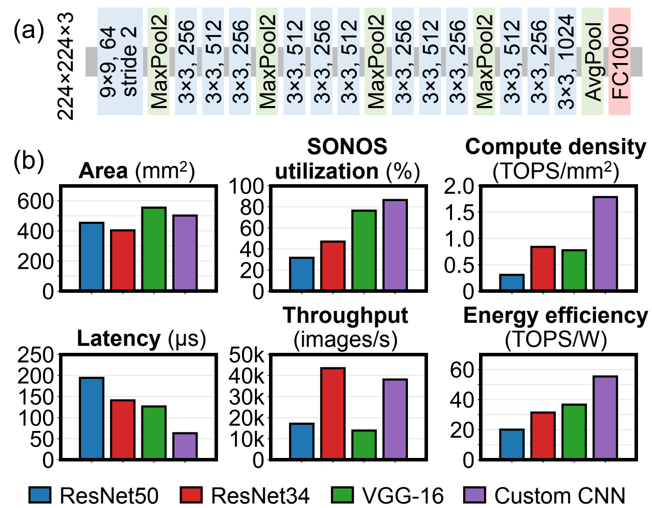

Fig. 17: (a) Custom CNN designed to maximize energy efficiency, (b) Comparison of accelerator performance on different ImageNet CNNs at approximately the same total area.

ratio of the SONOS device enable comparatively large arrays to be used without degrading accuracy. Our SONOS arrays have $9\times$ more rows than the memristor arrays in ISAAC. Since analog operations inside the array are much cheaper in energy than peripheral circuit operations, large arrays directly increase energy efficiency.

IX. Conclusion

The intrinsic physical properties of SONOS charge trap memory are uniquely well matched to the statistical properties of modern neural networks, whose weights are heavily skewed toward zero. By operating the device in the subthreshold regime, where the current-voltage characteristics are exponential, two key properties can be exploited: a high conductance On/Off ratio ($>10^7$) and conductance programming errors that scale with conductance. Most notably, the error approaches zero in the limit of zero conductance, which allows the most abundant weight values to be represented with the highest precision. These properties are crucial in suppressing the accumulation of errors in a large array, whether the errors arise from imprecise programming or from parasitic voltage drops. The error resilience possible with SONOS was demonstrated on ImageNet, where the modeled accuracy (2.16% loss on ResNet50 without training) is the highest obtained so far in the literature, among works that are based on experimentally characterized devices. The precision and On/Off ratio of the device can be further leveraged to build a more energy-efficient analog accelerator that uses large arrays and avoids bit slicing. Without compromising accuracy or robustness, the proposed SONOS-based inference accelerator attains >20 TOPS/W efficiency for ResNet50 and a peak efficiency >55 TOPS/W with an optimally co-designed neural network.

Acknowledgments

This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, and was also funded in part by DTRA under grant no.

TABLE II: Energy and performance comparison with other inference accelerators

	Google TPUv1 [4]	Nvidia A100 [1]	Alibaba HanGuang 800 [66]	Qualcomm Cloud AI 100 [67]	ISAAC [7]	This work
Process	28 nm	7 nm	12 nm	7 nm	32 nm	40 nm
Accelerator type	Digital	Digital	Digital	Digital	Analog, memristor	Analog, SONOS
Data resolution	8 bits	8 bits	8 bits	8 bits	16 bits	8 bits
Compute density (TOPS/mm^2)	0.06 (average) 0.28 (peak)	0.36 (ResNet50)	0.80 (ResNet50)	–	0.48* (peak)	0.31* (ResNet50) 1.78* (custom)
Energy efficiency (TOPS/W)	2.3 (peak)	1.20 (ResNet50)	2.03 (ResNet50)	2.34 (ResNet50)	0.63* (peak)	20.1* (ResNet50) 55.3* (custom)

*The performance of ISAAC and this work are simulated, while others are measured hardware performance. ResNet50 performance values are based on MLPerf Inference v1.1 for the Nvidia A100 and Qualcomm Cloud AI 100, and v0.5 for the Hanguang 800 [18]. In TPUv1, 'average' refers to the average performance across six datacenter inference workloads [4]. For the digital accelerators, compute density and efficiency are based on vendor-published die area and thermal design power, if available. For the SONOS accelerator, 'custom' refers to the CNN in Fig. 17(a) which achieves near-peak efficiency.

HDTRA1-17-1-0038. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

REFERENCES

[1] (2020, Nov.) NVIDIA data center deep learning product performance. NVIDIA. [Online]. Available: https://developer.nvidia.com/deep-learning-performance-training-inference

[2] E. Chung *et al.*, "Serving DNNs in real time at datacenter scale with Project Brainwave," *Intl. Symp. on Microarchitecture (MICRO)*, vol. 38, no. 2, pp. 8–20, Mar 2018.

[3] B. Gaide, D. Gaitonde, C. Ravishankar, and T. Bauer, "Xilinx Adaptive Compute Acceleration Platform: Versal architecture," in *Intl. Symp. on Field-Programmable Gate Arrays*, 2019, pp. 84–93.

[4] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a Tensor Processing Unit," in *Intl. Symp. on Computer Architecture (ISCA)*, June 2017, p. 1–12.

[5] N. P. Jouppi, D. H. Yoon, G. Kurian, S. Li, N. Patil, J. Laudon, C. Young, and D. Patterson, "A domain-specific supercomputer for training deep neural networks," *Communications of the ACM*, vol. 63, no. 7, pp. 67–78, 2020.

[6] M. N. Bojnordi and E. Ipek, "Memristive Boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning," in *Intl. Symp. on High Performance Computer Architecture (HPCA)*, March 2016.

[7] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *Intl. Symp. on Computer Architecture (ISCA)*, June 2016.

[8] P. Chi, S. Li, S. Li, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," in *Intl. Symp. on Computer Architecture (ISCA)*, June 2016.

[9] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nature Nanotechnology*, vol. 15, no. 7, pp. 529–544, 2020.

[10] 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," *Applied Physics Reviews*, vol. 7, no. 3, p. 031301, 2020.

[11] S. Yu, H. Jiang, S. Huang, X. Peng, and A. Lu, "Compute-in-memory chips for deep learning: Recent trends and prospects," *Circuits and Systems Magazine*, vol. 21, no. 3, pp. 31–56, 2021.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: a large-scale hierarchical image database," in *Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2009.

[13] V. Joshi, M. L. Gallo, S. Haefeli, I. Boybat, S. Nandakumar, C. Piveteau, M. Dazzi, B. Rajendran, A. Sebastian, and E. Eleftheriou, "Accurate deep neural network inference using computational phase-change memory," *Nature Communications*, vol. 11, 2020.

[14] Z. He, J. Lin, R. Ewetz, J.-S. Yuan, and D. Fan, "Noise injection adaption: End-to-end ReRAM crossbar non-ideal effect adaption for neural network mapping," in *Design Automation Conf. (DAC)*, June 2019, pp. 57:1–57:6.

[15] S. Jain and A. Raghunathan, "CxDNN: Hardware-software compensation methods for deep neural networks on resistive crossbar systems," *ACM Trans. Embed. Comput. Syst.*, vol. 18, no. 6, Nov. 2019.

[16] I. Kouznetsov *et al.*, "40 nm ultralow-power charge-trap embedded NVM technology for IoT applications," in *Intl. Memory Workshop (IMW)*, 2018, pp. 187–190.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[18] V. J. Reddi *et al.*, "MLPerf Inference Benchmark," in *Intl. Symp. on Computer Architecture (ISCA)*, June 2020.

[19] A. Coates, B. Huval, T. Wang, D. J. Wu, A. Y. Ng, and B. Catanzaro, "Deep learning with COTS HPC systems," in *Intl. Conference on Machine Learning (ICML)*, 2013, pp. III–1337–III–1345.

[20] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *Intl. Symp. on Computer Architecture (ISCA)*, June 2016, p. 367–379.

[21] V. Sze, Y. Chen, T. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[22] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication," in *Design Automation Conf. (DAC)*, June 2016, pp. 1–6.

[23] C. Li *et al.*, "Analogue signal and image processing with large memristor crossbars," *Nature Electronics*, vol. 1, no. 1, pp. 52–59, 2018.

[24] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, and H. Qian, "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, no. 7792, pp. 641–646, 2020.

[25] S. Ambrogio *et al.*, "Reducing the impact of phase-change memory conductance drift on the inference of large-scale hardware neural networks," in *Intl. Electron Devices Meeting (IEDM)*, 2019, pp. 6.1.1–6.1.4.

[26] X. Guo, F. M. Bayat, M. Bavandpour, M. Klachko, M. R. Mahmoodi, M. Prezioso, K. K. Likharev, and D. B. Strukov, "Fast, energy-efficient, robust, and reproducible mixed-signal neuromorphic classifier based on embedded NOR flash memory technology," in *Intl. Electron Devices Meeting (IEDM)*, Dec. 2017, pp. 6.5.1–6.5.4.

[27] P. Wang, F. Xu, B. Wang, B. Gao, H. Wu, H. Qian, and S. Yu, "Three-dimensional NAND flash for vector–matrix multiplication," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 4, pp. 988–991, 2019.

[28] L. Fick, D. Blaauw, D. Sylvester, S. Skrzyniarz, M. Parikh, and D. Fick, "Analog in-memory subthreshold deep neural network accelerator," in *Custom Integrated Circuits Conf. (CICC)*, May 2017, pp. 1–4.

[29] Y. Du, L. Du, X. Gu, J. Du, X. S. Wang, B. Hu, M. Jiang, X. Chen, S. S. Iyer, and M. F. Chang, "An analog neural network computing engine using CMOS-compatible charge-trap-transistor (CTT)," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 10, pp. 1811–1819, 2019.

[30] Y. Long, D. Kim, E. Lee, P. Saha, B. A. Mudassar, X. She, A. I. Khan, and S. Mukhopadhyay, "A ferroelectric FET-based processing-in-memory architecture for DNN acceleration," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 5, no. 2, pp. 113–122, 2019.

[31] A. Nag, R. Balasubramonian, V. Srikumar, R. Walker, A. Shafiee, J. P. Strachan, and N. Muralimanohar, "Newton: Gravitating towards the physical limits of crossbar acceleration," *Intl. Symp. on Microarchitecture (MICRO)*, vol. 38, no. 5, pp. 41–49, 2018.

[32] A. Ankit *et al.*, "PUMA: A programmable ultra-efficient memristor-based accelerator for machine learning inference," in *Intl. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, April 2019, p. 715–731.

[33] T.-H. Yang, H.-Y. Cheng, C.-L. Yang, I.-C. Tseng, H.-W. Hu, H.-S. Chang, and H.-P. Li, "Sparse ReRAM engine: Joint exploration of activation and weight sparsity in compressed neural networks," in *Intl. Symp. on Computer Architecture (ISCA)*, 2019, p. 236–249.

[34] W. Li, P. Xu, Y. Zhao, H. Li, Y. Xie, and Y. Lin, "TIMELY: Pushing data movements and interfaces in PIM accelerators towards local and in time domain," in *Intl. Symp. on Computer Architecture (ISCA)*, June 2020, p. 832–845.

[35] T. P. Xiao, B. Feinberg, C. H. Bennett, V. Prabhakar, P. Saxena, V. Agrawal, S. Agarwal, and M. J. Marinella, "On the accuracy of analog neural network inference accelerators," *arXiv preprint arXiv:2109.01262*, 2021.

[36] Y. LeCun, "The MNIST database of handwritten digits," *http://yann.lecun.com/exdb/mnist/*.

[37] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[38] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2018, pp. 2704–2713.

[39] L. Chua and S. M. Kang, "Memristive devices and systems," *Proceedings of the IEEE*, vol. 64, no. 2, pp. 209–223, 1976.

[40] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[41] D. Ielmini, "Resistive switching memories based on metal oxides: mechanisms, reliability and scaling," *Semiconductor Science and Technology*, vol. 31, no. 6, p. 063002, may 2016.

[42] S. M. Sze and K. K. Ng, *Physics of Semiconductor Devices*. John Wiley & Sons, 2006.

[43] C. R. Schlottmann and P. E. Hasler, "A highly dense, low power, programmable analog vector-matrix multiplier: The FPAA implementation," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, vol. 1, no. 3, pp. 403–411, 2011.

[44] Y.-H. Hsiao, H.-T. Lue, M. Y. Lee, S.-C. Huang, T.-Y. Chou, S.-Y. Wang, K.-Y. Hsieh, R. Liu, and C.-Y. Lu, "A study of SONOS charge loss mechanism after hot-hole stressing using trap-layer engineering and electrical re-fill methods," in *Intl. Reliability Physics Symp. (IRPS)*, 2008, pp. 695–696.

[45] H.-T. Lue, Y.-H. Shih, K. Y. Hsieh, R. Liu, and C.-Y. Lu, "Novel soft erase and re-fill methods for a p^+-poly gate nitride-trapping non-volatile memory device with excellent endurance and retention properties," in *Intl. Reliability Physics Symp. (IRPS)*, 2005, pp. 168–174.

[46] H.-T. Lue, Y.-H. Hsiao, Y.-H. Shih, E.-K. Lai, K.-Y. Hsieh, R. Liu, and C.-Y. Lu, "Study of charge loss mechanism of SONOS-type devices using hot-hole erase and methods to improve the charge retention," in *Intl. Reliability Physics Symp. (IRPS)*, 2006, pp. 523–529.

[47] V. Agrawal, V. Prabhakar, K. Ramkumar, L. Hinh, S. Saha, S. Samanta, and R. Kapre, "In-memory computing array using 40nm multibit SONOS achieving 100 TOPS/W energy efficiency for deep neural network edge inference accelerators," in *Intl. Memory Workshop (IMW)*, May 2020.

[48] Y. Cai, Y. Luo, E. F. Haratsch, K. Mai, and O. Mutlu, "Data retention in MLC NAND flash memory: Characterization, optimization, and recovery," in *Intl. Symp. on High Performance Computer Architecture (HPCA)*, 2015, pp. 551–563.

[49] K. Hung, P. Ko, C. Hu, and Y. Cheng, "A unified model for the flicker noise in metal-oxide-semiconductor field-effect transistors," *IEEE Trans. on Electron Devices*, vol. 37, no. 3, pp. 654–665, 1990.

[50] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[51] D. Miyashita, E. H. Lee, and B. Murmann, "Convolutional neural networks using logarithmic data representation," *arXiv preprint arXiv:1603.01025*, 2016.

[52] V. Milo, F. Anzalone, C. Zambelli, E. Pérez, M. K. Mahadevaiah, s. G. Ossorio, P. Olivo, C. Wenger, and D. Ielmini, "Optimized programming algorithms for multilevel RRAM in hardware neural networks," in *Intl. Reliability Physics Symp. (IRPS)*, 2021, pp. 1–6.

[53] S. Agarwal, S. J. Plimpton, R. K. Schiek, I. Richter, A. H. Hsia, D. R. Hughart, R. B. Jacobs-Gedrim, C. D. James, and M. J. Marinella. (2017) CrossSim. [Online]. Available: https://cross-sim.sandia.gov/

[54] C. H. Bennett *et al.*, "Device-aware inference operations in SONOS nonvolatile memory arrays," in *Intl. Reliability Physics Symp. (IRPS)*, 2020, pp. 3C.2.1–3C.2.6.

[55] G. W. Burr, R. S. Shenoy, K. Virwani, P. Narayanan, A. Padilla, B. Kurdi, and H. Hwang, "Access devices for 3D crosspoint memory," *Journal of Vacuum Science & Technology B*, vol. 32, no. 4, p. 040802, 2014.

[56] T. P. Xiao, B. Feinberg, J. Rohan, C. Bennett, S. Agarwal, and M. Marinella, "Analysis and mitigation of parasitic resistance effects for analog in-memory neural network acceleration," *Semiconductor Science and Technology*, vol. 36, no. 11, p. 114004, 2021.

[57] B. D. Salvo *et al.*, "Performance and reliability features of advanced nonvolatile memories based on discrete traps (silicon nanocrystals, SONOS)," *Trans. on Device and Materials Reliability*, vol. 4, no. 3, pp. 377–389, 2004.

[58] M. J. Marinella, S. Agarwal, A. Hsia, I. Richter, R. Jacobs-Gedrim, J. Niroula, S. J. Plimpton, E. Ipek, and C. D. James, "Multiscale co-design analysis of energy, latency, area, and accuracy of a ReRAM analog neural training accelerator," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, vol. 8, no. 1, pp. 86–101, 2018.

[59] F. Sequin and A. Fabre, "New second generation current conveyor with reduced parasitic resistance and bandpass filter application," *IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications*, vol. 48, no. 6, pp. 781–785, 2001.

[60] M. Bavandpour, S. Sahay, M. R. Mahmoodi, and D. Strukov, "Efficient mixed-signal neurocomputing via successive integration and rescaling," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 3, pp. 823–827, 2020.

[61] L. Kull, T. Toifl, M. Schmatz, P. A. Francese, C. Menolfi, M. Brändli, M. Kossel, T. Morf, T. M. Andersen, and Y. Leblebici, "A 3.1 mW 8b 1.2 GS/s single-channel asynchronous SAR ADC with alternate comparators for enhanced speed in 32 nm digital SOI CMOS," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 48, no. 12, pp. 3049–3058, 2013.

[62] M. Baert and W. Dehaene, "20.1 a 5GS/s 7.2 ENOB time-interleaved VCO-based ADC achieving 30.5 fJ/conv-step," in *Intl. Solid- State Circuits Conf. (ISSCC)*, Feb. 2019, pp. 328–330.

[63] R. Balasubramonian, A. B. Kahng, N. Muralimanohar, A. Shafiee, and V. Srinivas, "CACTI 7: New tools for interconnect exploration in innovative off-chip memories," *ACM Trans. on Architecture and Code Optimization (TACO)*, vol. 14, no. 2, pp. 14:1–14:25, June 2017.

[64] A. B. Kahng, B. Lin, and K. Samadi, "Improved on-chip router analytical power and area modeling," in *Asia and South Pacific Design Automation Conf. (ASP-DAC)*, Jan 2010, pp. 241–246.

[65] A. B. Kahng, B. Lin, and S. Nath, "Explicit modeling of control and data for improved NoC router estimation," in *Design Automation Conf. (DAC)*, June 2012, pp. 392–397.

[66] Y. Jiao, L. Han, and X. Long, "Hanguang 800 NPU - the ultimate AI inference solution for data centers," in *Hot Chips Symp.*, 2020, pp. 1–29.

[67] (2021, Sept.) Qualcomm Cloud AI 100 Announcement. Qualcomm. [Online]. Available: https://www.qualcomm.com/media/documents/files/qualcomm-cloud-ai-100-announcement.pdf