Analysis and Design of a Passive Switched-Capacitor Matrix Multiplier for Approximate Computing

Edward H. Lee, *Student Member, IEEE*, and S. Simon Wong, *Fellow, IEEE*

***Abstract*— A switched-capacitor matrix multiplier is presented for approximate computing and machine learning applications. The multiply-and-accumulate operations perform discrete-time charge-domain signal processing using passive switches and 300 aF unit capacitors. The computation is digitized with a 6 b asynchronous successive approximation register analog-to-digital converter. The analyses of incomplete charge accumulation and thermal noise are discussed. The design was fabricated in 40 nm CMOS, and experimental measurements of multiplication are illustrated using matched filtering and image convolutions to analyze noise and offset. Two applications are highlighted: 1) energy-efficient feature extraction layer performing both compression and classification in a neural network for an analog front end and 2) analog acceleration for solving optimization problems that are traditionally performed in the digital domain. The chip obtains measured efficiencies of 8.7 TOPS/W at 1 GHz for the first application and 7.7 TOPS/W at 2.5 GHz for the second application.**

***Index Terms*— Analog computing, approximate computing, matched filtering, matrix factorization, neural networks, switched-capacitor circuits.**

I. Introduction

MATRIX multiplication is the fundamental operation $y = Ax$ where $x \in \mathbb{R}^n$ maps to output $y \in \mathbb{R}^m$ by a linear system A. It is ubiquitously used in scientific computing, computer graphics, machine learning, real-time signal processing, and optimization. Matrix multiplication in hardware is traditionally realized by multiply-and-accumulate (MAC) units commonly used in general-purpose graphics processing units, field programmable gate arrays, and application-specific integrated circuits. Three important parameters in matrix multiplication are computation speed (e.g., throughput), energy efficiency, and resolution. For example, while high computation speed is of utmost importance for scientific computing and graphics, energy efficiency plays a more significant role for embedded systems. On the other hand, high resolution is used to obtain high accuracies in computational simulations [1].

There have been recent works in reduced-precision multiplication for statistical inference systems optimized for energy-efficient operation. These applications operate on inherently noisy data and perform tasks such as classification and recognition that are resilient to low signal-to-noise (SNR). These fundamental ideas are the motivating forces for reduced-precision or approximate computing. Such systems include classification systems for images and audio and supervised training in machine learning [2–7]. For example, Vanhoucke *et al.* [4] show that the performance of inference for neural networks is robust at 8 b fixed point. Inference in the context of image recognition entails the prediction result of one image using programmable weights (e.g., elements in the matrix A) that were trained offline. Courbariaux *et al.* [5] and Miyashita *et al.* [6] show that resolutions for state-of-the-art networks [8] for the ImageNet Challenge [9] can go down to less than 4 b. The ability for these systems to operate at these ultralow precisions opens up the possibility of scalable CMOS analog signal processing to work in synergy with digital systems for higher energy efficiency.

Analog-domain MAC operations can also operate on raw analog data obtained from the sensor before digitization. This can alleviate the analog-to-digital (A/D) requirements. Traditionally, conventional systems use A/D matrix multiplication (AD-MM) [10], which is a common task in modern sensing and communication systems. AD-MM digitizes an analog signal and multiplies the resulting data by a matrix. For example, AD-MM is used in cameras to compress digital data using transform coding and quantization. However, many analog signals are known to have a sparse representation in some basis, which presents an opportunity to reduce the A/D data rate in an A/D system. For example, [11] designed an analog discrete cosine transformation (DCT) in an image sensor in order to compress data before digitization. The works [12–22] have explored the use of analog MACs to alleviate A/D requirements for AD-MM.

Analog MAC designs come with many options that are mainly influenced by energy, speed, and resolution requirements. Translinear and current-mode [23,24], and time-based approaches [25], [26] allow for analog computation to meet large dynamic ranges under low supply voltages. However, these approaches are susceptible to variations in process, voltage, and temperature for small unit current sources and unit delays. Voltage domain approaches use switches and capacitors to perform the MAC operation. Capacitor sizes dictate the multiplication elements in the matrix A and charge

Manuscript received April 23, 2016; revised June 19, 2016 and July 27, 2016; accepted July 30, 2016. Date of publication September 29, 2016; date of current version January 4, 2017. This paper was approved by Guest Editor Dennis Sylvester.

The authors are with the Electrical Engineering Department, Stanford University, Stanford, CA 94305 USA. (e-mail: edhlee@stanford.edu)

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

Digital Object Identifier 10.1109/JSSC.2016.2599536

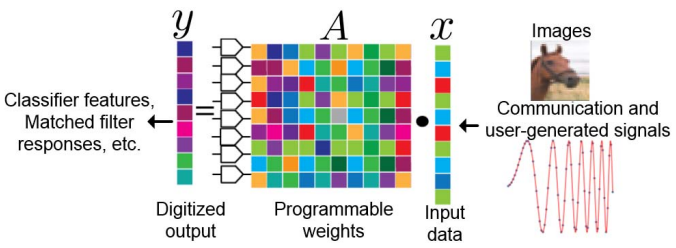


Fig. 1. Analog matrix multiplication for various signal processing applications.

redistribution (either active or passive) performs the accumulation. Switches and capacitors are highly amenable to the nanometer CMOS process. For example, switched-capacitor circuits in filters and successive approximation register (SAR) analog-to-digital converters (ADCs) are designed with signal transfer functions that depend only on the ratios of two capacitors and not on their absolute capacitance values. Because of this, the size of capacitors and switches can be aggressively scaled to decrease the dynamic CV^2 energy. Recent experiments on mismatch of metal fringe capacitors [27] show that subfemto fringe capacitors can be realized with around a 1% mismatch (1 std. deviation) in 32 nm. Our implementation uses 300 aF capacitors for multiplication. This translates to a 1% gain error (1 std. deviation) for the least significant bit (LSB) multiplier, which is well within our 3 b multiplication specification.

Both active and passive switched-capacitor MACs have been implemented in the past for information processing. Active approaches are commonly used when more than 8–9 b are required [11]. However, since recent machine learning applications can operate with fewer bits, there has been a growing push toward passive switched-capacitor implementations. For example, Zhang *et al.* [28] embed charge multiplication in a SAR ADC using switches, capacitors, and digital barrel-shifting in 130 nm. Parallel switched-capacitor MACs for neural network applications were proposed in [29] and [30].

Our switched-capacitor matrix multiplier (SCMM) draws inspiration from these previous works to enable sequential MACs using only passive switches and capacitors to target applications that are resilient to reduced precision. The system platform for analog matrix multiplication and its various applications are illustrated in Fig. 1. MAC operations take elements in matrix A, multiply them with elements in vector x in the analog domain, and digitize the result to produce y. The elements of A are stored in memory and are realized using a digital-to-analog converter (DAC) that multiplies x. One complete innerproduct of the jth row of A with the input for example is $y_j = \Sigma_{i=1}^{n} A[j,i]x[i]$, where n is the number of elements in x and also the number of MAC clock cycles. This operation is performed m times for m rows in the matrix A.

Because input data can be either analog or digital, we explore two options for x.

1) x is inherently a discrete-time analog voltage $x[i] = V_{\text{in}}[i]$ where i represents its time index.

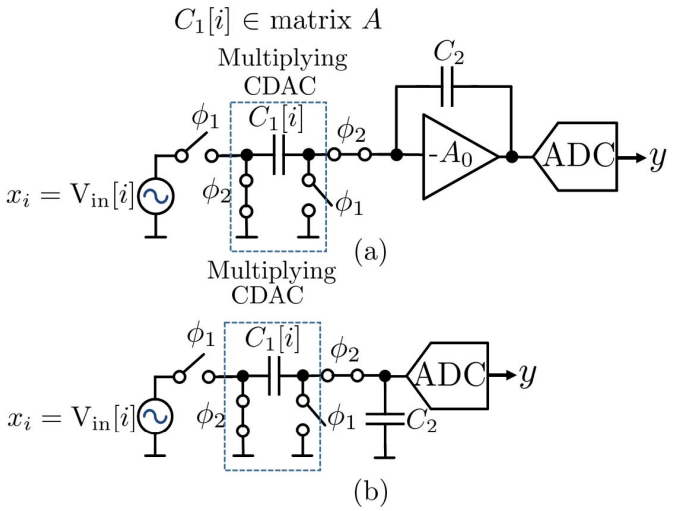


Fig. 2. (a) Active analog MAC and (b) passive analog MAC implementations.

2) x is a digital vector that then generates the analog voltage $x[i] = V_{\text{in}}[i]$.

The first option is aimed at AD-MM for sensing interfaces and communication, where it is possible to not only perform the matrix operation while the data is inherently analog but also to alleviate A/D requirements after matrix computation. The second option is aimed to accelerate matrix evaluations in machine learning applications where our design interfaces with a digital environment. The final SCMM's energy efficiency of 8.8 (1 GHz) and 7.7 TOPS/W (2.5 GHz) is computed using the measured total power of the SCMM (input DAC, MAC, SAR, memory, clock, and self-test).

II. Design of SCMM

A. Active Versus Passive MACs

There are many conceptual switched-capacitor methods to perform the MAC operation $y_j = \Sigma_{i=1}^{n} A[j,i]x[i]$. Fig. 2 highlights two main approaches: 1) active and 2) passive. For both the approaches, the input is a voltage $x[i] = V_{\text{in}}[i]$ that is multiplied by $C_1[i]$ during ϕ_1, where $C_1[i]$ is a capacitive DAC controlled by the $A[j,i]$ value in memory. During ϕ_2, the multiplied charge $V_{\text{in}}[i]C_1[i]$ is redistributed onto the capacitor C_2 either by active or passive means. The ϕ_1 and ϕ_2 operations are performed n times for each element in the inner product before quantizing the voltage on C_2. However, due to finite gain for active and incomplete charge transfer for passive, the actual inner product is not ideally $V_{C_2} = (1/C_2)\Sigma_{i=1}^{n} V_{\text{in}}[i]C_1[i]$.

Instead, due to finite amplifier gain A_0 in the active approach, the voltage of C_2 at the ith cycle is $V_{C_2}[i] = k[i]V_{C_2}[i-1] + \mu[i]k[i]V_{\text{in}}[i]$, where $k[i] = \frac{C_2(A_0+1)}{C_2(A_0+1)+|C_1[i]|}$ and $\mu[i] = \frac{C_1[i]}{C_2}\frac{A_0}{A_0+1}$. $k[i]$ is a droop term that represents the fraction of charge on C_2 that is leaked away from cycle to cycle (ideally $k[i] = 1$), and $\mu[i]k[i]$ contains the voltage gain error at every sample i. Since the matrix calculation only cares about the last cycle $i = n$,

we can write the output voltage at time n as

$$V_{C_2}[n] = \sum_{i=1}^{n} \mu[i] V_{\text{in}}[i] \prod_{j=i}^{n} k[j]$$
$$= \Bigg[\mu[1] \prod_{i=1}^{n} k[i] \quad \mu[2] \prod_{i=2}^{n} k[i] \quad \dots \quad \mu[n-1] \prod_{i=n-1}^{n} k[i] \quad \mu[n] k[n] \Bigg] x. \quad (1)$$

This result can be extended to the passive case in Fig. 2(b) where $k[i] = (C_2/(C_2 + |C_1[i]|))$ and $\mu[i] = (C_1[i]/C_2)$. Note that in the passive modality, these gain and droop terms only depend on the ratios of the capacitors and not on the amplifier gain, which is sensitive to nonlinearities. The gain components $\mu[i]k[i]$, which are $(C_1[i]A_0/(C_2(A_0+1)+|C_1[i]|))$ for active and $(C_1[i]/(C_2+|C_1[i]|))$ for passive, indicate that in order to achieve a more ideal gain of $(C_1[i]/C_2)$, one has to increase either the dc gain for active or increase the size of C_2 for passive. Equating the active and passive gains indicates that in order to achieve the same level of gain error, the active's dc gain must be $(C_2/|C_1|)+1 \gg 1$. This undesirable attribute suggests that the amplifier may be unnecessary for low precisions. At high resolutions on the other hand, the active approach is more compelling since the choice of dc gain offers one extra degree of freedom, whereas the passive approach is limited to the ratios of two capacitors. This conclusion also extends to the droop terms $k[i]$. Therefore, this comparison indicates that at least for low-resolution gains, the passive approach can yield the same levels of MAC performance (neglecting noise and nonlinearity) as the active approach but without the use of an amplifier.

Performing this operation on m rows in the matrix A results in a nonideal matrix operation $y = \tilde{A}x$, where $\tilde{A}$ is the actual matrix in (2), as shown at the bottom of this page, due to gain and droop errors. Note that since we can embed the analog nonidealities into the matrix $\tilde{A}$, the matrix operation $\tilde{A}x$ is yet another matrix operation and therefore still linear with x. There are a couple of approaches to ensure that the system performance is robust to this linear error. First, we ensure that $\tilde{A} \approx A$ at the 3 b level. In our 3 b MAC, C_2 is roughly 39× larger than the maximum $C_1[i]$. Second, we show that matrix factorization [10] allows the ability to correct for the matrix error $\tilde{A} - A$. To summarize, these nonidealities contribute errors in the matrix A for both active and passive approaches but do not present nonlinear effects for the passive approach. And for low-resolution multiplication (e.g., 3 b), the errors $\tilde{A} - A$ can be either corrected for if necessary or simply ignored as long as $C_2 \gg C_1$. Finally, the passive approach is not only more area and energy efficient than active but also has bandwidth advantages. While the speed of the active approach is limited by both the settling performance of the amplifier in feedback and the RC settling of the switches, the speed of the passive approach is set only by the RC settling of the switches.

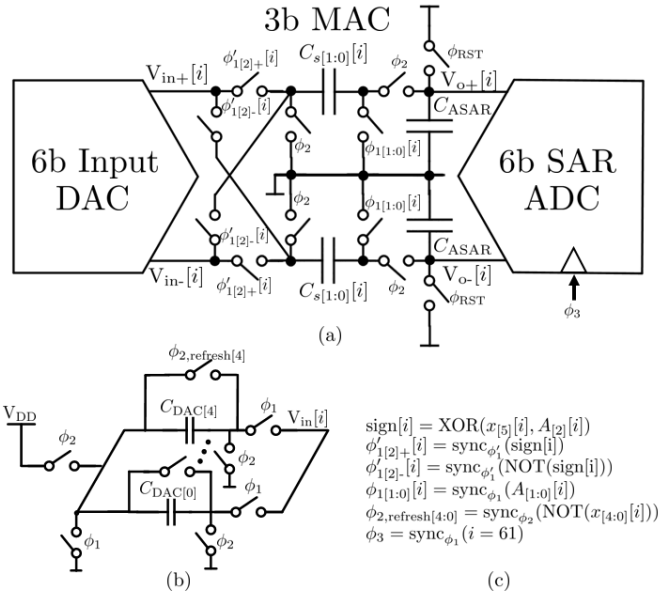


Fig. 3. (a) SCMM implementation. (b) SCMMs 6 b input DAC. (c) Digital controls.

B. Circuit Implementation

For the reasons mentioned in Section II-A, we implement our MAC core in the SCMM using the passive approach. The full SCMM implementation using the MAC is shown in Fig. 3 where C_s is C_1 and C_{ASAR} is C_2 from Section II-A. We apply the SCMM for applications where the input to the MAC is both analog (application 1) and all digital (application 2). Unlike application 2, in application 1 we evaluate the MAC and SAR for use in an environment where the input is inherently analog, where the 6 b input DAC is used as an on-chip test source. The MAC uses 300 aF fringe capacitors for C_s. The SAR ADC uses top-plate sampling with its 6 b cap-DAC C_{ASAR} and digitizes the bit codes asynchronously to efficiently fit all bit decisions in the narrow time window. The 6 b input DAC which generates $V_{\text{in}}[i]$ is a 5 b cap-DAC with a sign-bit

$$\tilde{A} = \begin{bmatrix} \mu[1,1] \prod_{i=1}^{n} k[1,i] & \mu[1,2] \prod_{i=2}^{n} k[1,i] & \dots & \mu[1,n]k[1,n] \\ \mu[2,1] \prod_{i=1}^{n} k[2,i] & \mu[2,2] \prod_{i=2}^{n} k[2,i] & \dots & \mu[2,n]k[2,n] \\ \vdots & \vdots & \vdots & \vdots \\ \mu[m,1] \prod_{i=1}^{n} k[m,i] & \mu[m,2] \prod_{i=2}^{n} k[m,i] & \dots & \mu[m,n]k[m,n] \end{bmatrix} \quad (2)$$

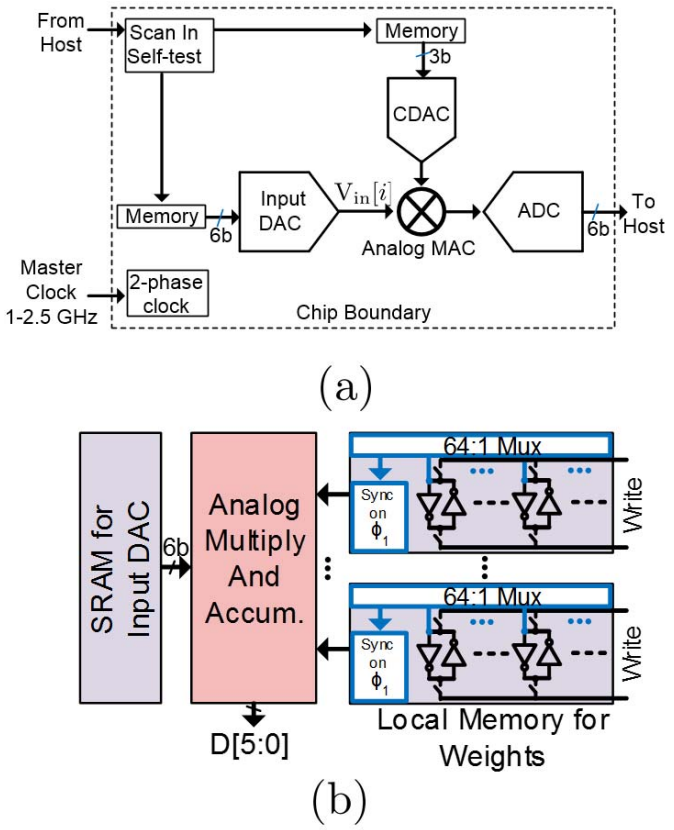


Fig. 4. (a) SCMM chip boundary when under test and (b) compute-memory architecture.

and is shown in Fig. 3(b). The digital control signals are shown in Fig. 3(c). These signals, which are preloaded from local memory, are synchronized with the rising edge of global overlapping clocks ϕ_1 and ϕ_2.

After digitization, the charge on C_{ASAR} is dumped and reset, and the operation restarts to process the next input vector. One complete inner product involves 64 cycles of sequential MACs using the input DAC and MAC and one phase of digitization by the SAR ADC. Unlike [28], the accumulate operation is also performed in the analog charge domain, which fundamentally reduces the number of A/D conversions and rate by 64× for every set of 64 MACs. Fig. 4 shows the chip boundary during testing and the complete compute-memory engine. The host (computer) inputs digital codes for matrices A and x that are loaded into local memory. Once loaded, a start token triggers the read signals to fetch from local memory.

The 6 b input DAC generates a voltage $V_{\text{in}}[i]$ to be sampled during ϕ_1, where this voltage is prepared during the ϕ_2 phase. During the ϕ_2 phase, the 6 b memory for the input DAC prepares $V_{\text{in}}[i]$ by either recharging the input's cap-DAC to V_{DD} or discharging the previous cycle's charge using $\phi_{2,\text{refresh}}$ signals. This resulting sampled charge $Q_{\text{input}}[i+1]$ generates $V_{\text{in}}[i+1] = (Q_{\text{input}}[i+1]/(C_{\text{DAC,tot}} + C_s[i+1]))$ where the input DAC's cap-DAC is 35× larger than $C_s[i+1]$ such that $V_{\text{in}}[i+1]$ is independent of $C_s[i+1]$ at 6 b. Furthermore, when two adjacent input words share common bit codes (e.g., MSB $=1$ for time i and $i+1$), the corresponding bit capacitor's remaining charge from the previous cycle is recycled for the next input.

All the operational phases and timing are displayed in Fig. 5. During ϕ_1, the input DAC generates a voltage that is sampled by the cap-DAC of the MAC, $C_s[i]$, which now holds $V_{\text{in}}[i]C_s[i]$. During ϕ_2, this charge is redistributed onto $C_{s,\text{tot}}$ and C_{ASAR}. As stated earlier in Section II-A, $C_{\text{ASAR}} \gg C_{s,\text{tot}}$, and thus most of the sampled charge $V_{\text{in}}[i]C_s[i]$ is pushed to C_{ASAR}. The residual charge that remains untransferred is at most 2.7% of the total multiplied charge from cycle to cycle and presents a signal-independent and correctable gain error. While this happens, the input DAC's cap-DAC prepares for the next cycle. After 61 phases of $\phi_1 - \phi_2$ MAC cycles, ϕ_3 is triggered which starts the MSB conversion of the digitization process. Input charge packets during the last few cycles only affect the last LSB SAR decisions. This prestart saves 4 cycles or 6.3% of the total timing budget with insignificant bit errors. The SARs asynchronous logic finishes its LSB decision near the end of the 64th cycle. The SAR loop uses a nonmonotonic binary search algorithm [31]. After the SAR algorithm, the 6 b output value is sent off-chip to be read by the host computer, and the C_{ASAR} is reset to start the next inner product.

The memory is designed with decoupled write and bitline-less read because the memory read is more active than memory write. For application 1, the weights do not change at all per filter operation across the image. The idea that the weights are static is exploited in many other machine learning systems [32] where the weights are pretrained off-chip. The bitline-less, local read design is also motivated by fast read times, obviating the need for bitline-precharge and sense-amps. For application 2, the number of overwrite events is less than that of the read by up to 76% in certain write transitions. The local memory preloads the data at a half-cycle before each MAC, allowing for sufficient setup times at 2.5 GHz. The codesign judiciously constrains the total memory read energy to be well-balanced with the compute energy contribution.

C. Effect of MAC's Incomplete Charge Transfer

In Section II-A, we show that the charge accumulation for passive MAC is not perfect and results in a linear and correctable error. Recall that incomplete charge transfer transforms the originally intended matrix A to a matrix $\tilde{A}$ in (2). In this section, we quantify this error for our circuit implementation and propose a correction algorithm to alleviate this error whenever necessary.

As demonstration, we multiply a matrix $A \in \mathbb{R}^{64\times 64}$ by a vector x to form $y_1, y_2, \ldots, y_{64}$. We design A such that y_1 is highly correlated with x while y_i for $i \neq 1$ are uncorrelated with x. In Fig. 6, we illustrate the ideal and actual (passive MAC) running sums of y_1, y_2, and y_3. Note that what is important is the final value $y[i = 64]$. We also expect that y_1 should reach a high value while channels y_i for $i \neq 1$ should be roughly 0 at $i = 64$ as illustrated in Fig. 6(a). The output y_1 clearly accumulates in time compared with y_2 and y_3 for both ideal and passive. For the passive approach, we see attenuation that presents significant attenuation over time but which nonetheless preserves its relative value at the end of the operation at $i = 64$. This is

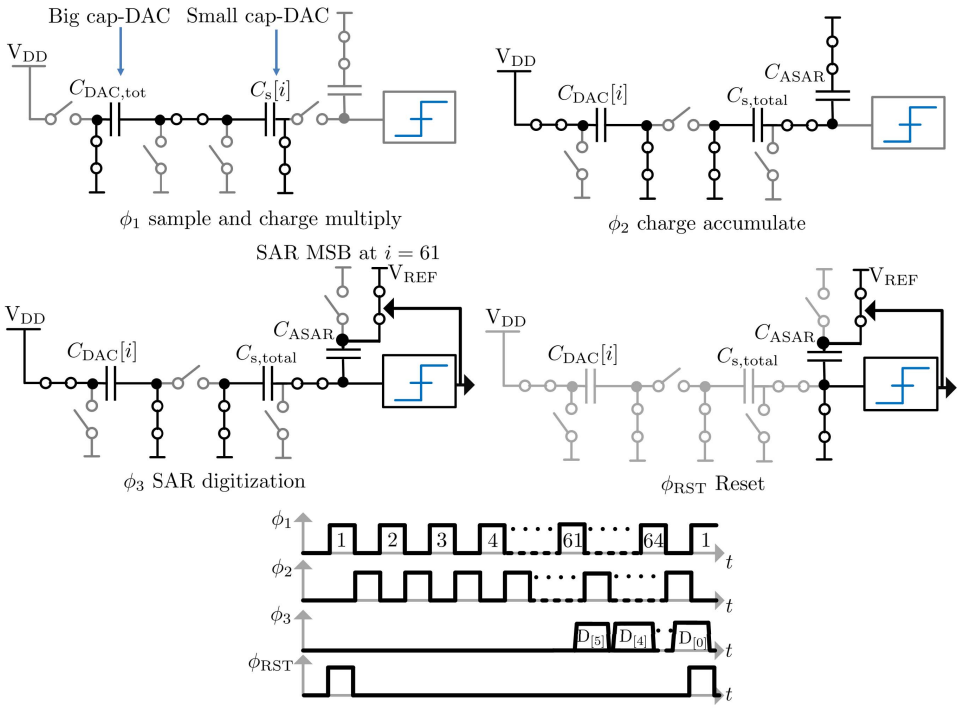


Fig. 5. Operational modes and timing diagram.

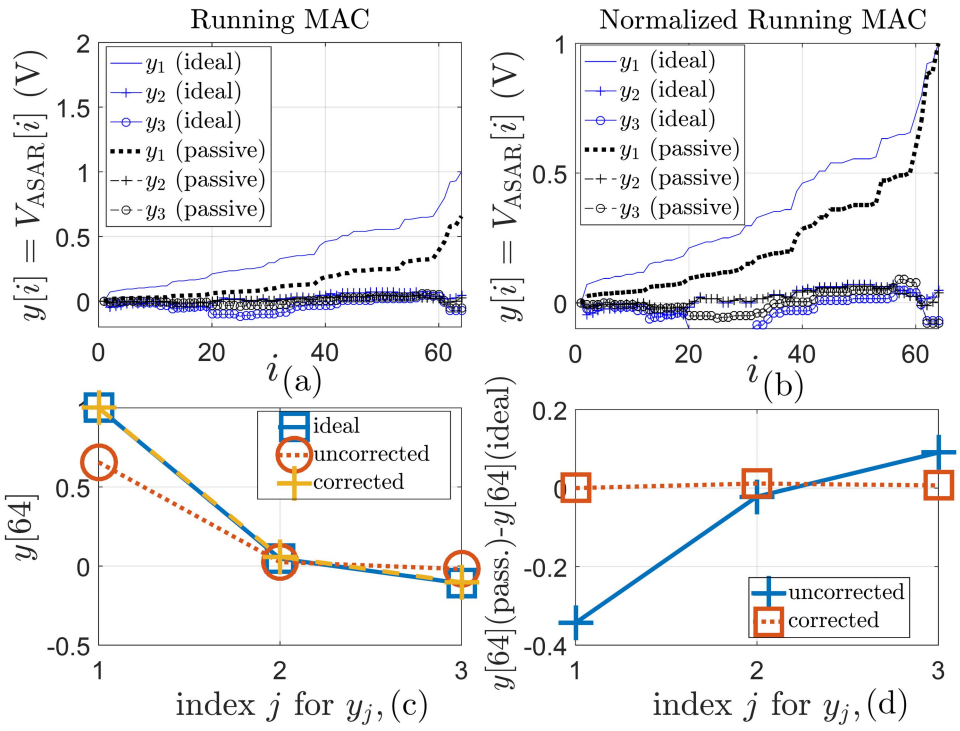


Fig. 6. Ideal and simulated analog MAC transient outputs for a matrix operation Ax.

the result of incomplete charge transfer. Fig. 6(b) normalizes the running sums of the passive such that $y_1[64](\text{ideal}) = y_1[64](\text{passive})$. As illustrated, passive MAC attenuates all the final values $y_1[64]$, $y_2[64]$, $y_3[64]$ approximately equally and therefore presents an absolute gain error and not a relative error. We can nonetheless correct for this linear error using factorization by applying another matrix B to invert these errors. Formally, we solve for B in

$$\begin{aligned} &\min. \ ||A - B\tilde{A}||_F \\ &\text{s.t } B \in \Omega_B \end{aligned} \tag{3}$$

where F is the Frobenius norm, A is the intended ideal matrix, $\tilde{A}$ is the actual matrix due to incomplete charge accumulation, B is the correction matrix, and Ω_B is the set of possible

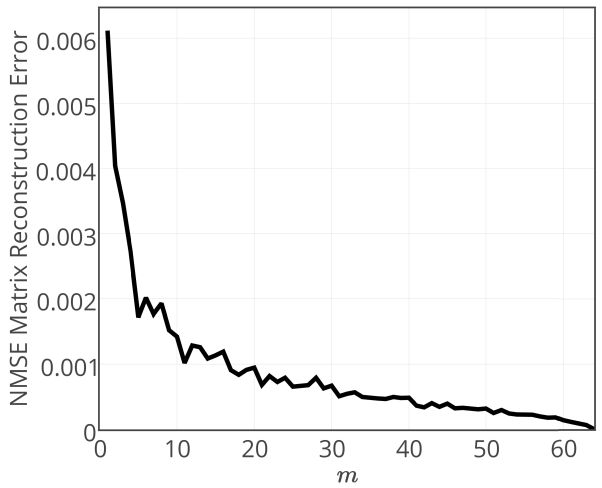


Fig. 7. Matrix factorization reconstruction error (normalized MSE) as a function of m.

values that B can take on. For example, if the matrix B were performed in fixed point, Ω_B would contain a finite, discrete set of values allowed by fixed-point multiplication. Furthermore, cascades of matrix operations are naturally present in many applications that include AD-MM and neural networks. By applying $B \in \mathbb{R}^{8\times 64}$ found using the algorithm in (3), we obtain the corrected matrix multiplies y_1, y_2, y_3 as shown in Fig. 6(c) and the error between ideal and corrected in Fig. 6(d). It is important to note that the performance of factorization is a strong function of the aspect ratio of $B \in \mathbb{R}^{m\times n}$. Fig. 7 illustrates the matrix reconstruction error $||A - B\tilde{A}||_F$ for various m keeping n constant.

We also illustrate the effect of incomplete charge transfer from the perspective of the matrix itself. Fig. 8(a) illustrates the ideal matrix A, and Fig. 8(b) and (c) illustrates the uncorrected matrix $\tilde{A}$ and calibrated matrix $B\tilde{A}$. For this simulation, we accentuate the amount of attenuation for qualitative effect by setting C_{ASAR} to be only $10\times$ larger than the maximum C_s instead of $39\times$ in our actual implementation.

D. Noise Analysis of MAC

The MAC system in Fig. 3 is designed such that its output noise is well within the 6 b SNR target of the ADC. In spite of the 300 aF unit capacitors, the thermal noise contributions from the MAC, the input DAC's cap-DAC, and asynchronous SAR (comparator) are designed to be well below the LSB (7 mV) of the ADC's output.

Since the MAC's total capacitance is $35\times$ smaller than that of the input DAC, we detail only the kTC noise of this block due to its much greater contribution of thermal noise. The noise contribution from the MAC's cap-DAC is as follows. Let the noise voltage (rms) on C_{ASAR} be $\sigma_{C_{\text{ASAR}}}[i]$ at time $i = 1, \ldots, 64$. During the first cycle, ϕ_1 is turned ON and OFF. This generates a sampled noise charge $kTC_s[i]$ on $C_s[i]$. When ϕ_2 is turned ON, this charge is redistributed onto $C_{s,\text{tot.}}$ and C_{ASAR}. The noise voltage across C_{ASAR} is then $(kTC_s[i]/(C_{s,\text{tot.}} + C_{\text{ASAR}})^2)$. When ϕ_2 is subsequently turned OFF, this generates another sampled noise charge on C_{ASAR} of $kT\frac{C_{s,\text{tot.}}C_{\text{ASAR}}}{C_{s,\text{tot.}}+C_{\text{ASAR}}}$. The total noise voltage on C_{ASAR} after a complete ϕ_1-ϕ_2 cycle is $\sigma_{C_{\text{ASAR}}}[i = 1] = ((kT|C_s[i]|/(C_{s,\text{tot.}} + C_{\text{ASAR}})^2) + (kT(C_{s,\text{tot}}C_{\text{ASAR}}/\ (C_{s,\text{tot}} + C_{\text{ASAR}}))/C_{\text{ASAR}}^2))^{1/2}$.

After i ϕ_1-ϕ_2 cycles, the noise voltage on C_{ASAR} becomes (4), as shown at the bottom of this page.

Due to the time-varying nature of $C_{s[i]}$, it is rather difficult to gain any intuition. We can, however, conclude that the minimum noise occurs when $C_{s[i]} = 0$ and the highest occurs when $C_{s[i]} = C_{s,\text{tot.}}$. At the highest noise power, we simplify (5), as shown at the bottom of this page.

Interestingly, for many MAC cycles $i \to \infty$, the noise power reaches equilibrium (or steady state) and simply becomes $\lim_{i\to\infty}\sigma[i] = (kT/C_{\text{ASAR}})^{1/2}$. However, for our implementation the MAC cycles stop at time $i = 64$; therefore, the noise due to the MAC is slightly lower than $(kT/C_{\text{ASAR}})^{1/2}$, the steady-state value.

To validate this, we perform Monte Carlo transient noise simulations in 40 nm for the MAC circuit during the 64 $\phi_1 - \phi_2$ MAC phases and where we set $C_s[i] = C_{s,\text{tot.}}$. Fig. 9 plots 150 Monte Carlo transients for the differential output voltage on C_{ASAR} and the predicted 3σ line from (5). The simulated variance grows with the number of MAC cycles and shows close agreement with analytical predictions. We also highlight that the actual thermal noise of the MAC is less than (kT/C_{ASAR}) during the MAC's operating window.

III. Chip Measurements and Applications

The SCMM was fabricated in 40 nm CMOS, and its die photo is shown in Fig. 10. The performance of matrix multiplication $y = Ax$ is measured, $A \in \mathbb{R}^{8\times 64}$ contains orthonormal row vectors where each row of A, denoted by a_j^T, is orthogonal to all a_k^T where $j \neq k$, and $x = a_l^T$ for $l \in \{1, \ldots, 8\}$. Many applications arise where the matrix operation is an orthonormal transformation. These include discrete Fourier transformation and DCT and matched filtering. Here we apply such transformations to characterize matrix multiplication performance.

$$\sigma_{C_{\text{ASAR}}}[i] = \sqrt{\left(\frac{kT|C_s[i]|}{(C_{s,\text{tot.}} + C_{\text{ASAR}})^2} + \frac{kT\left(\frac{C_{s,\text{tot}}C_{\text{ASAR}}}{C_{s,\text{tot}}+C_{\text{ASAR}}}\right)}{C_{\text{ASAR}}^2}\right)\left(\sum_{j=0}^{i}\left(\frac{C_{\text{ASAR}}}{C_{\text{ASAR}} + C_{s,\text{tot.}}}\right)^{2j}\right)} \tag{4}$$

$$\sigma_{C_{\text{ASAR}}}[i] = \sqrt{\frac{kT}{C_{\text{ASAR}}}\frac{C_{s,\text{tot}}(C_{s,\text{tot}} + 2C_{\text{ASAR}})}{(C_{s,\text{tot}} + C_{\text{ASAR}})^2}\sum_{j=0}^{i}\left(\frac{C_{\text{ASAR}}}{C_{s,\text{tot}} + C_{\text{ASAR}}}\right)^{2j}} \tag{5}$$

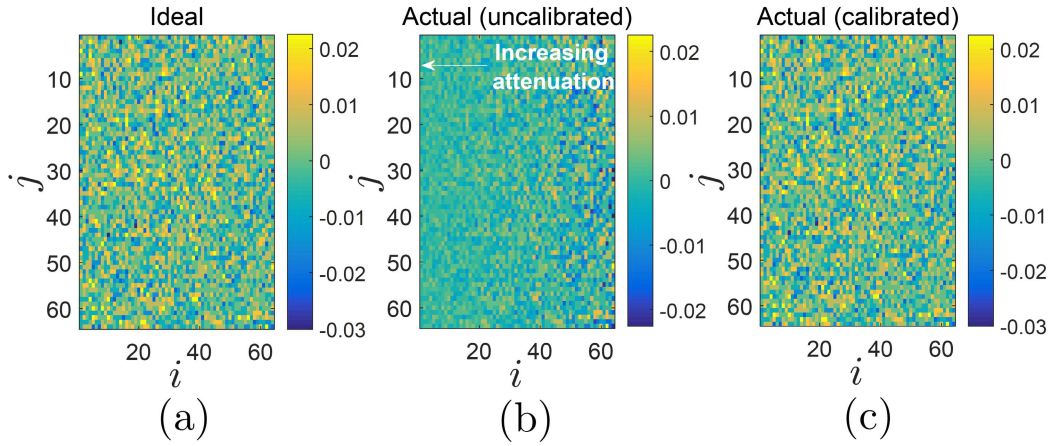


Fig. 8. (a) Programmed 64 × 64 matrix A. (b) Matrix that is a result of incomplete accumulation $\tilde{A}$. (c) Corrected matrix using matrix factorization.

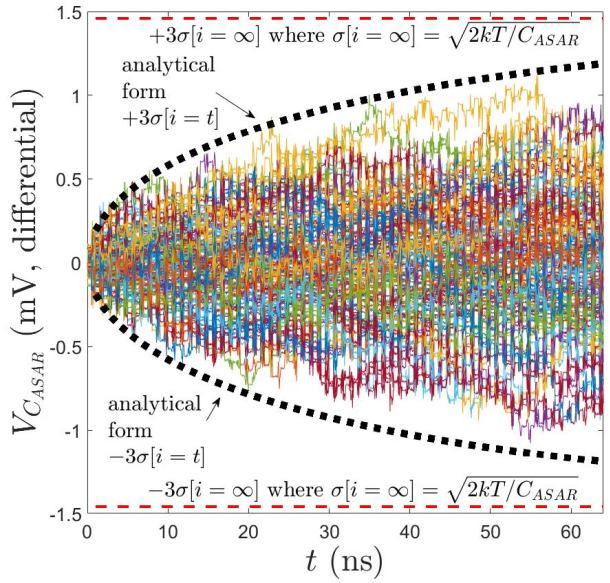


Fig. 9. Transient noise simulation of the noise voltage on C_{ASAR}.

We measure $y_j = Ax^{(i)}$ where $x^{(i)} = a_i^T$ for 8 trials $i \in \{1, \ldots, 8\}$ and for eight output channels $j \in \{1, \ldots, 8\}$ using a clock rate of 1 GHz. Note that the $x^{(i)}$ is a vector, which is different from the scalar element $x[i]$, which denotes the value at time i. The ideal multiplication result is shown in Fig. 11(a), and the measured output is shown in Fig. 11(d) as raw digital codes and normalized in Fig. 11(b). Note that for clarity, we have omitted the quantization operation that occurs when sending in the values for A and x to memory. Fig. 11(a) indicates that under ideal matrix operation Ax, we expect to see $y_j = 1$ for $j = i$ and $y_j = 0$ for $j \neq i$. However, due to quantization of A and x as well as nonidealities in the circuit operation, we observe Fig. 11(b), which clearly shows nonzero values of y_j for $j \neq i$. The mean square error (MSE) of the ideal to the measured is 0.00723. The normalized mean square error (NMSE) is 0.0579. Using matrix factorization as described in Section II-C, we obtain Fig. 11(c), which attenuates the cross-terms y_j for $j \neq i$ and equalizes the diagonal terms y_j for $j = i$.

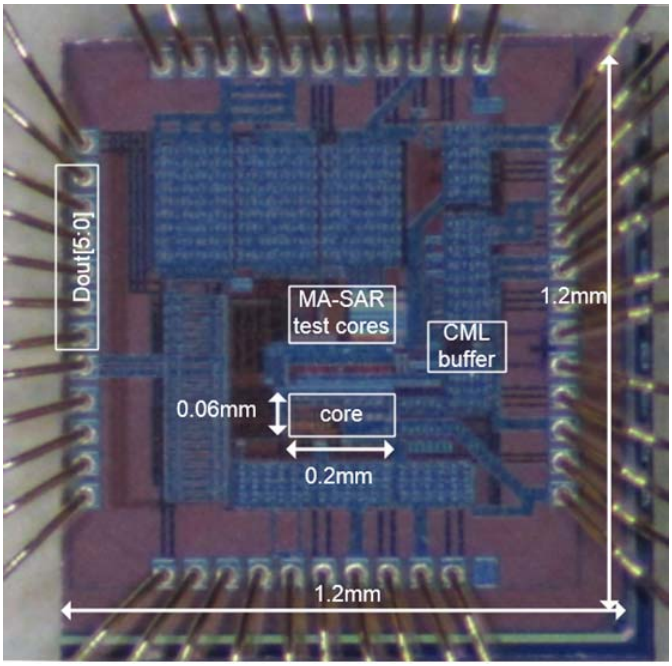


Fig. 10. Photograph of the chip in 40 nm CMOS.

We measure the system noise performance using a matched filtering setup at 1 GHz. Here, we perform a single innerproduct $y = \sum_{i=1}^{64} a[i]x[i]$ where $x = a + n$, a is a chirp signal, and n is independent and identically distributed additive white Gaussian noise source. We vary the input SNR by sweeping the variance of n and plot the output response y in Fig. 12(a) and perform 25 independent trials. The output voltage is derived from using the output codes and the LSB size (7 mV) of the ADC outputs. We plot the mean and variance of the output for varying levels of the input SNR. Increasing input SNRs decreases the noise contribution from the input source (variance) until around 5 dB input SNR, when the output variance drops to the noise floor set by the LSB size. The system offset in Fig. 12(b) is similarly obtained by measuring the mean of the innerproducts of random vectors, which ideally converges to 0. The offset of the entire system due to leakage,

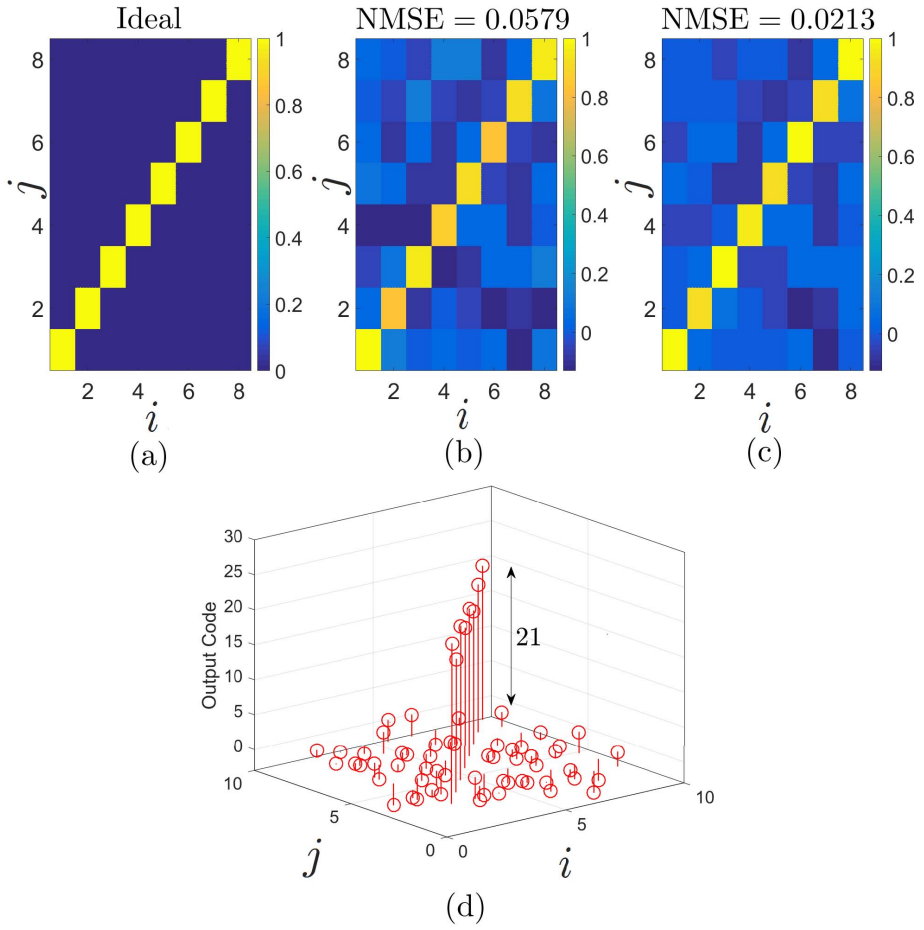


Fig. 11. (a) Ideal matrix-vector product output. (b) Measured output. (c) Corrected output. (d) Measured digital code.

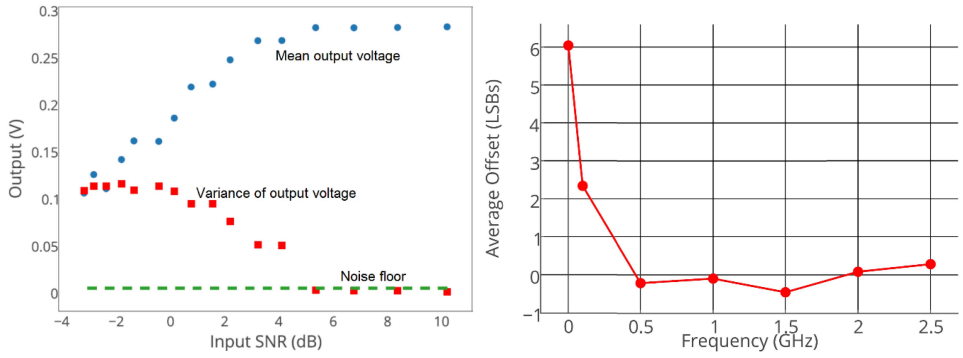


Fig. 12. Measured matched filter response for varying input SNRs and the measured offset.

comparator, and capacitor mismatch contributions is within half-LSB of the ADC output at speeds >0.5 GHz.

In our first application, we test our MAC implementation for use in a neural network image classifier. Image classifiers with convolutional neural networks use a cascade of linear matrix multiplies followed by nonlinearity functions. After training, the first layer of a network contains Gabor-like features that are used to detect edges. Fig. 13 illustrates the ideal activations [Fig. 13(c)] when 1 Gabor-like filter [Fig. 13(b)] is scanned over the red channel of the colored input image [Fig. 13(a)]. This scanning process is performed by multiplying the input image with the filter for varying horizontal and vertical offsets. We obtain the results from our chip [Fig. 13(d)] and simulated digital [Fig. 13(e) and (f)]. The simulated digital [Fig. 13(e)] uses fixed-point arithmetic with 6 b input, 3 b weights, and 6 b output notated as (6/3/6 b), but with infinite accumulator resolution while Fig. 13(f) uses 6/3/6 b with 6 b accumulator.

We extend this primitive to reduce the A/D footprint when the first neural network layer operates on inherently

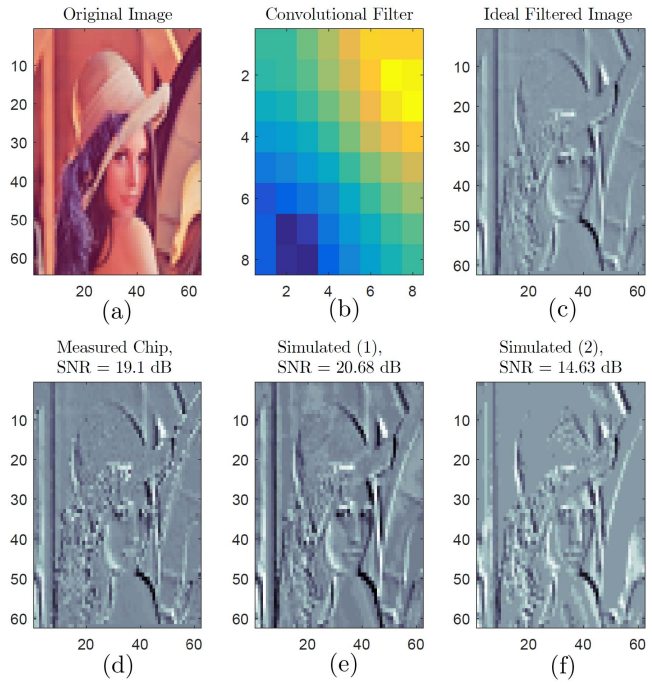


Fig. 13. Measured versus simulated performance of a filter (b) on an image (a).

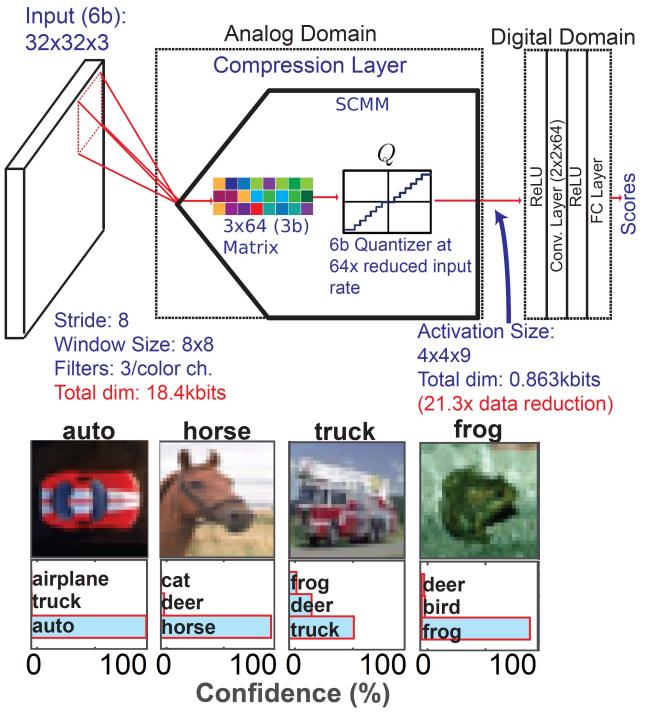


Fig. 14. Designed compression and classifying layer for a small convolutional neural network and measured confidence levels for sampled input images on the CIFAR-10 database.

TABLE I

PERFORMANCE OF THE COMPRESSION LAYER COMPARED WITH THE CONVENTIONAL

	Conventional	This work
Top-3 Accuracy (%)	86	85
Layer's Energy/Op (fJ) at 1GHz	145*	13
# of A/Ds per image at 6b	3072	144
Resolution	6b/4b/6b	Analog/3b /6b
NMSE of feature outputs (avg. over all batches)	0.0033	0.0054
*Energy estimated by synthesis in 40nm Resolution Notation: Input/Weights/Output		

analog inputs. For systems that process inherently analog inputs, multiplying a full-rank matrix A of size $m \times n$ in the analog domain reduces the A/D rate by n and compresses the total number of A/D conversions by factor n/m when $n > m$ where analog matrix multiplication is made practical using factorization. We extend this idea by applying this compression technique to a classifier in a neural network. This front-end layer both compresses and classifies analog image data in an end-to-end pretrained neural network as illustrated in Fig. 14. To reduce the dimensionality, the matrix-vector operations are processed at 1 GHz in nonoverlapping regions of the image with a stride equal to the filter width and height of size 8 and output dimension smaller than its input dimension. Three filters are applied per color channel, and the resulting activations of size $4 \times 4 \times 9$ are digitized and pipelined to the remaining two layers in the digital domain. The dimensionality of the output of this layer is $4^2 \times 9$ while that of the input is $32^2 \times 3$; this layer therefore compresses the data by a factor of $21.3\times$ and furthermore decreases the number of A/D operations by $21.3\times$ and digitization rate by $64\times$.

This front-end layer and digital layers are cotrained together in floating point using real images from the CIFAR-10 database [33]. The performance summary is shown in Table I. The reduced-precision digital layer using digital fixed point yields 86% top-3 accuracy while the proposed analog layer yields 85% top-3 accuracy. The digital fixed-point layer is simulated in MATLAB and uses 6 b for input, 4 b for weights, and 6 b for outputs with full-precision accumulation and no overflow. The energy per operation of the proposed analog layer is $11\times$ less than that of digital, which is simulated in 40 nm. The synthesized digital MAC uses 6 b input, 4 b weights, and 6 b outputs with a 6 b accumulator. Note that to prevent overflow, digital MACs are designed to have an output bitwidth that is at least the sum of the bitwidths of the input and weights. More commonly, the output bitwidth is set to be $\log_2(n)$ larger than the sum of the input bitwidths, where n is the number of MAC products. However, for our energy comparisons, we are conservative in our digital energy estimation, and as such we set the digital output bitwidth to be 6 b instead of $6 + 4 + \log_2(64) = 15$ b. Finally, the proposed analog layer lowers the number of A/D conversions by $21\times$ compared with the conventional digital approach. The compute to memory read energy ratio is 1.18:1.

For our second application, we demonstrate analog coprocessing and acceleration for computation that is

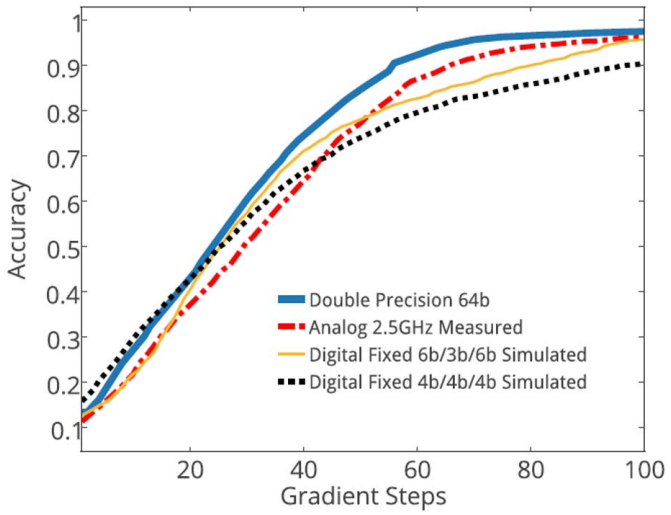


Fig. 15. Classification accuracy as a function of the number of gradient steps in the stochastic gradient descent procedure. NMSE of analog averaged over 100 steps is 0.006.

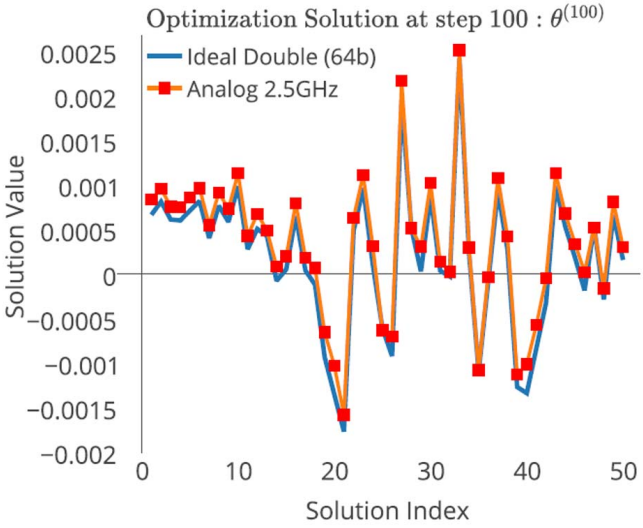


Fig. 16. Final optimization solution after 100 gradient steps.

traditionally performed in the digital domain. This application computes gradients in stochastic gradient descent, one of the most widely used optimization algorithms in high performance computing. The goal of the optimization algorithm is to find θ, the vector of unknown parameters that finds the global optimal solution by minimizing a user-defined objective function $J(\theta)$, which is nonconvex in general. Here, the SCMM is used to maximize the test accuracy for image classification by minimizing $J(\theta)$ over a training set by iteratively updating $\theta^{(i+1)} := \theta^{(i)} + \alpha \nabla J(\theta^{(i)})$, where i represents the iteration count, $\nabla J(\theta^{(i)})$ the gradient evaluated at $\theta^{(i)}$ from a sampled batch, and α the learning rate. Evaluating the gradients $\nabla J(\theta^{(i)})$ is an expensive operation that usually consists of a large matrix multiply for many classifiers including neural networks. Here we perform gradient descent on an image classification task and offload the gradient computation $\nabla J(\theta^{(i)}) = A^{(i)}[x_1^{(i)}, \ldots, x_m^{(i)}]$ with the SCMM chip, where m is the batch size, and $x_1^{(i)}$ corresponds to the data vector from one image sample. The SCMM takes as inputs a digital matrix A and a digital vector x, and performs the matrix operation Ax in the analog charge domain. The 6 b output is the gradient update at each iteration. The 100 gradient updates are performed on a sample classification problem with a learning rate $\alpha = 10^{-6}$. The accuracy of our chip is compared with various simulated digital computations in Fig. 15. The measured solution to the optimization problem is shown in Fig. 16, which shows close alignment with the simulated digital. Analog acceleration using the SCMM at 2.5 GHz performs slightly worse than digital double-precision 64 b and is equivalent to the simulated digital fixed point at an estimated 6× lower energy and the compute to memory read energy ratio is 1.05:1.

Table II summarizes the performance of the analog charge-domain MAC for two applications compared with a recent work of embedding multiplication in an SAR ADC [28]. The efficiencies are computed based on measured power and speed. The measured efficiency including compute, memory, self-test logic, and clock is 8.7 TOPS/W at 1 GHz for application 1, and 7.7 TOPS/W at 2.5 GHz for application 2. Compared with [28] (130 nm), our work (40 nm) is two orders of magnitude more energy efficient, where our energy also includes the DAC, clocking, and memories. However, it is worth mentioning that [28] embeds digital barrel-shifting with analog fixed-point multiplication, which allows a higher multiplication resolution but makes analog-domain accumulation difficult to realize.

TABLE II
PERFORMANCE SUMMARY

	ISSCC 2015 [28]	This Work	
Technology	130nm	40nm	
Application	Sensor Classifier	Sensor Classifier	Analog Accelerator
Analog Multiply Rate	20kHz*	1GHz	2.5GHz
Analog Accumulate Rate	N/A	1GHz	2.5GHz
A/D Rate	20kS/s	15MS/s	39MS/s
Resolution	Analog / 4b* / 8b	Analog / 3b / 6b	6b / 3b / 6b
Supply	1.2V	1V	1.1V
Total Power	663nW	228uW**	647uW**
Efficiency (TOPS/W)	0.0603*	8.77**	7.72**

*4b analog multiply and bitshifting after SAR digitization
**includes: SCMM, self-test logic, clock and excludes CML
Resolution notation: Input/Weights/Output

IV. CONCLUSION

This paper presents the SCMM, which uses switches, 300 aF unit capacitors, and local memories. We present general results for multiplication and characterization of noise and offset. We also analyze MAC imperfections such as thermal noise and incomplete charge accumulation. We show that this error is linear with the input, and is correctable using matrix factorization. Finally, we demonstrate the SCMM on two applications with high efficiency at above-gigahertz MAC rates.

ACKNOWLEDGMENT

The authors would like to thank D. Miyashita, B. Murmann, M. Udell, D. Bankman, C. Young, K. Zheng, and the TSMC University Shuttle Program and the TSMC University Shuttle Program for chip fabrication and Mentor Graphics for AFS tools.

REFERENCES

[1] R. Ginjupalli and G. Khanna. (2010). "High-precision numerical simulations of rotating black holes accelerated by CUDA." [Online]. Available: http://arxiv.org/abs/1006.0663

[2] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *Proc. 18th IEEE Eur. Test Symp. (ETS)*, May 2013, pp. 1–6.

[3] R. S. Amant *et al.*, "General-purpose code acceleration with limited-precision analog computation," in *Proc. 41st Annu. Int. Symp. Comput. Archit. (ISCA)*, Piscataway, NJ, USA, 2014, pp. 505–516.

[4] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," in *Proc. Deep Learn. Unsupervised Feature Learn. Workshop, NIPS*, 2011.

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

[6] D. Miyashita, E. H. Lee, and B. Murmann. (2016). "Convolutional neural networks using logarithmic data representation." [Online]. Available: https://arxiv.org/abs/1603.01025

[7] C. M. Bishop, "Training with noise is equivalent to Tikhonov regularization," *Neural Comput.*, vol. 7, no. 1, pp. 108–116, Jan. 1995.

[8] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556

[9] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[10] E. H. Lee, M. Udell, and S. S. Wong, "Factorization for analog-to-digital matrix multiplication," in *Proc. 40th Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Brisbane, QLD, Australia, Apr. 2015, pp. 1061–1065.

[11] S. Kawahito *et al.*, "A CMOS image sensor with analog two-dimensional DCT-based compression circuits for one-chip cameras," *IEEE J. Solid-State Circuits*, vol. 32, no. 12, pp. 2030–2041, Dec. 1997.

[12] M. Herman and T. Strohmer, "Compressed sensing radar," in *Proc. IEEE Radar Conf. (RADAR)*, May 2008, pp. 1–6.

[13] Y. Oike and A. El Gamal, "CMOS image sensor with per-column $\Sigma\Delta$ ADC and programmable compressed sensing," *IEEE J. Solid-State Circuits*, vol. 48, no. 1, pp. 318–328, Jan. 2013.

[14] D. Adams, C. S. Park, Y. C. Eldar, and B. Murmann, "Towards an integrated circuit design of a compresssed sampling wireless receiver," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2012, pp. 5305–5308.

[15] B. Sadhu, M. Sturm, B. M. Sadler, and R. Harjani, "A 5GS/s 12.2pJ/conv. analog charge-domain FFT for a software defined radio receiver front-end in 65 nm CMOS," in *Proc. IEEE Radio Freq. Integr. Circuits Symp. (RFIC)*, Jun. 2012, pp. 39–42.

[16] B. Sadhu, M. Sturm, B. M. Sadler, and R. Harjani, "Analysis and design of a 5 GS/s analog charge-domain FFT for an SDR front-end in 65 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 48, no. 5, pp. 1199–1211, May 2013.

[17] Y.-W. Lin, H.-Y. Liu, and C.-Y. Lee, "A 1-GS/s FFT/IFFT processor for UWB applications," *IEEE J. Solid-State Circuits*, vol. 40, no. 8, pp. 1726–1735, Aug. 2005.

[18] M. Lehne and S. Raman, "A 0.13-μm 1-GS/s CMOS discrete-time FFT processor for ultra-wideband OFDM wireless receivers," *IEEE Trans. Microw. Theory Techn.*, vol. 59, no. 6, pp. 1639–1650, Jun. 2011.

[19] F. Rivet, Y. Deval, J.-B. Begueret, D. Dallet, P. Cathelin, and D. Belot, "The experimental demonstration of a SASP-based full software radio receiver," *IEEE J. Solid-State Circuits*, vol. 45, no. 5, pp. 979–988, May 2010.

[20] S. Kirolos *et al.*, "Analog-to-information conversion via random demodulation," in *Proc. IEEE Dallas/CAS Workshop Design, Appl., Integr. Softw.*, Oct. 2006, pp. 71–74.

[21] O. Abari, F. Lim, F. Chen, and V. Stojanovic, "Why analog-to-information converters suffer in high-bandwidth sparse signal applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 9, pp. 2273–2284, Sep. 2013.

[22] W. Yin, S. Morgan, J. Yang, and Y. Zhang, "Practical compressive sensing with Toeplitz and circulant matrices," *Proc. SPIE 7744, Visual Communications and Image Processing 2010*, vol. 7744, 77440K, Jul. 2010.

[23] J. Lu, S. Young, I. Arel, and J. Holleman, "A 1 TOPS/W analog deep machine-learning engine with floating-gate storage in 0.13 μm CMOS," *IEEE J. Solid-State Circuits*, vol. 50, no. 1, pp. 270–281, Jan. 2015.

[24] S. Skrzyniarz *et al.*, "A 36.8 2b-TOPS/W self-calibrating GPS accelerator implemented using analog calculation in 65nm LP CMOS," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, Jan./Feb. 2016, pp. 420–422.

[25] D. Miyashita *et al.*, "An LDPC decoder with time-domain analog and digital mixed-signal processing," *IEEE J. Solid-State Circuits*, vol. 49, no. 1, pp. 73–83, Jan. 2014.

[26] I. Nahlus, E. P. Kim, N. R. Shanbhag, and D. Blaauw, "Energy-efficient dot product computation using a switched analog circuit architecture," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2014, pp. 315–318.

[27] 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.

[28] J. Zhang, Z. Wang, and N. Verma, "A matrix-multiplying ADC implementing a machine-learning classifier directly with data conversion," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, Feb. 2015, pp. 1–3.

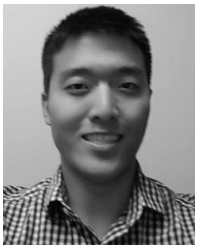
[29] Y. P. Tsividis and D. Anastassiou, "Switched-capacitor neural networks," *Electron. Lett.*, vol. 23, no. 18, pp. 958–959, Aug. 1987.

[30] D. Bankman and B. Murmann, "Passive charge redistribution digital-to-analogue multiplier," *Electron. Lett.*, vol. 51, no. 5, pp. 386–388, 2015.

[31] V. Tripathi and B. Murmann, "An 8-bit 450-MS/s single-bit/cycle SAR ADC in 65-nm CMOS," in *Proc. ESSCIRC*, Sep. 2013, pp. 117–120.

[32] D. Jeon *et al.*, "A 23 mW face recognition accelerator in 40 nm CMOS with mostly-read 5T memory," in *Proc. Symp. VLSI Circuits (VLSI Circuits)*, Jun. 2015, pp. C48–C49.

[33] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, ON, Canada, Tech. Rep., 2009.



Edward H. Lee (S'12) received the B.S. degree in electrical engineering from Arizona State University, (ASU), Phoenix, AZ, USA in 2012, and the M.S. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 2014, where he is currently pursuing the Ph.D. degree in electrical engineering.

He was with the Flexible Display Center, ASU from 2008 to 2012, designing image (X-ray and neutron) sensors on flexible substrates and Maxim Integrated, and designing calibration algorithms for high performance ADCs. His current research interests include the design and efficient implementation of machine learning algorithms on hardware, approximate computing, and deep learning.

Mr. Lee was a recipient of the Texas Instruments Stanford Graduate Fellowship. He is an NSF Graduate Fellow and a Goldwater Scholar.

S. Simon Wong (M'83–SM'91–F'99) received the Bachelor's degrees in electrical engineering and mechanical engineering from the University of Minnesota, Minneapolis, MN, USA, and the M.S. and Ph.D. degrees in electrical engineering from the University of California at Berkeley, Berkeley, CA, USA.

His industrial experience includes semiconductor memory design at National Semiconductor from 1978 to 1980 and semiconductor technology development at Hewlett Packard Laboratories from 1980 to 1985. He was an Assistant Professor with Cornell University, Ithaca, NY, USA, from 1985 to 1988. Since 1988, he has been with Stanford University, Stanford, CA, USA, where he is currently a Professor of electrical engineering. His current research interests include understanding and overcoming the factors that limit performance in devices, interconnections, on-chip components, and packages.