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

Edward Lee       S. Simon Wong
Contact: edhlee@stanford.edu
Stanford University, Electrical Engineering Stanford, CA, USA

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 300aF unit capacitors. The computation is digitized with a 6b asynchronous SAR. The analyses of incomplete charge accumulation and thermal noise are discussed. The design was fabricated in 40nm 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 8.7TOPS/W at 1GHz for the first application and 7.7TOPS/W at 2.5GHz for the second application.

0.1 Keywords

1) analog computing, 2) approximate computing, 3) neural networks, 4) matched filtering, 5) matrix factorization, 6) switched-capacitor circuits
1 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 performs 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, 3, 4, 5, 6, 7]. For example, the work of [4] shows that the performance of inference for neural networks is robust at 8b 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. The works of [5, 6] show that resolutions for state-of-the-art networks [8] for the ImageNet Challenge [9] can go down to less than 4b. The ability for these systems to operate at these ultra-low precisions opens up the possibility of scalable CMOS analog signal processing to work in synergy with digital systems for higher energy efficiency.

Analog-domain multiply and accumulate operations can also operate on raw analog data obtained from the sensor before digitization. This can alleviate analog-to-digital (A/D) require-
ments. Traditionally, conventional systems use analog-to-digital 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 analog-to-digital system. For example, [11] designed an analog DCT in an image sensor in order to compress data before digitization. Many recent papers [12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22] have explored the use of analog MACs to alleviate A/D requirements for AD-MM.

Analog MAC designs come in many options that are mainly influenced by energy, speed, resolution requirements. Translinear [23], current-mode [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 (PVT) 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 redistribution (either active or passive) performs the accumulation. Switches and capacitors are highly amenable to nanometer CMOS process. For example, switched-capacitor circuits in filters and successive approximation register (SAR) ADCs are designed with signal transfer functions that depend only on ratios of two capacitors and not their absolute capacitance values. Because of this, the size of capacitors and switches can be aggressively scaled to decrease dynamic $CV^2$ energy. Recent experiments on mismatch of metal fringe capacitors [27] show that sub-femto fringe capacitors can be realized with around 1% mismatch (1 std. deviation) in 32nm. Our implementation uses 300aF capacitors for multiplication. This translates to 1% gain error (1 std. deviation) for the LSB multiplier, which is well within our 3b 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 bits are required \[11\]. However, since recent machine learning applications can operate with fewer bits, there has been a growing push towards passive switched-capacitor implementations. For example, the work of \[28\] embeds charge multiplication in a SAR ADC using switches, capacitors, and digital barrel-shifting in 130nm. 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 Figure 1. MAC operations take elements in matrix \(A\), multiply them with elements in vector \(x\) in the analog domain, and digitizes 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 inner product of the \(j\)-th row of \(A\) with the input for example is \(y_j = \sum_{i=1}^{n} A[j,i]x[i]\), 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 \(x[i] = V_{in}[i]\) where \(i\) represents its time index and 2) \(x\) is a digital vector that then generates the analog voltage \(x[i] = V_{in}[i]\). 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 still 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 to a digital environment. The final SCMM’s energy efficiency of 8.8 (1GHz) and 7.7 TOPS/W (2.5GHz) is computed using the measured total power of the SCMM (input DAC, MAC, SAR, memory, clock, and self-test).
2 Design of SCMM

2.1 Active versus passive MACs

There are many conceptual switched-capacitor methods to perform the MAC operation \( y_j = \sum_{i=1}^n A[j,i]x[i] \). Figure 2 highlights two main approaches: (a) active and (b) passive. For both approaches, the input is a voltage \( x[i] = V_{in}[i] \) that is multiplied by \( C_1[i] \) during \( \phi_1 \), where \( C_1[i] \) is a capacitive-DAC (CDAC) controlled by the \( A[j,i] \) value in memory. During \( \phi_2 \), the multiplied charge \( V_{in}[i]C_1[i] \) is redistributed onto the capacitor \( C_2 \) either by (a) active or (b) passive means. The \( \phi_1 \) and \( \phi_2 \) 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 \( V_{C_2} = \frac{1}{C_2} \sum_{i=1}^{n} V_{in}[i]C_1[i] \).

Instead, due to finite amplifier gain \( A_0 \) in the active approach, the voltage of \( C_2 \) at the \( i \)-th cycle is \( V_{C_2}[i] = k[i]V_{C_2}[i-1] + \mu[i]k[i]V_{in}[i] \), where \( k[i] = \frac{C_2(A_0+1)}{C_2(A_0+1)+|C_1[i]|} \) and \( \mu[i] = \frac{C_1[i]A_0}{C_2(A_0+1)+|C_1[i]|} \). \( k[i] \) 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

\[
V_{C_2}[n] = \sum_{i=1}^{n} \mu[i]V_{in}[i] \prod_{j=i}^{n} k[j] = \left[ \mu[1] \prod_{i=1}^{n} k[i] \quad \mu[2] \prod_{i=2}^{n} k[i] \quad \ldots \quad \mu[n-1] \prod_{i=n-1}^{n} k[i] \quad \mu[n]k[n] \right] x.
\]

(1)

This result can be extended to the passive case in Figure 2(b) where \( k[i] = \frac{C_2}{C_2+|C_1[i]|} \) and \( \mu[i] = \frac{C_1[i]}{C_2+|C_1[i]|} \). Note that in the passive modality, these gain and droop terms only depend on the ratios of capacitors and not the amplifier gain, which is sensitive to nonlinearities. The gain components \( \mu[i]k[i] \), which are \( \frac{C_1[i]A_0}{C_2(A_0+1)+|C_1[i]|} \) for active and \( \frac{C_1[i]}{C_2+|C_1[i]|} \) for passive, indicate that in order to achieve a more ideal gain of \( \frac{C_1[i]}{C_2} \), 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 $\frac{C_2}{|C_1|} + 1 \gg 1$. This undesirable attribute implies that the amplifier is 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 non-linearity) as the active approach but without the use of an amplifier.

$$\tilde{A} = \begin{bmatrix}
\mu[1,1] \prod_{i=1}^n k[1,i] & \mu[1,2] \prod_{i=2}^n k[1,i] & \ldots & \mu[1,n]k[1,n] \\
\mu[2,1] \prod_{i=1}^n k[2,i] & \mu[2,2] \prod_{i=2}^n k[2,i] & \ldots & \mu[2,n]k[2,n] \\
\vdots & \vdots & \ddots & \vdots \\
\mu[m,1] \prod_{i=1}^n k[m,i] & \mu[m,2] \prod_{i=2}^n k[m,i] & \ldots & \mu[m,n]k[m,n]
\end{bmatrix}$$  \tag{2}

Performing this operation on $m$ rows in the matrix $A$ results in a non-ideal matrix operation $y = \tilde{A}x$, where $\tilde{A}$ is the actual matrix in Equation 2 due to gain and droop errors. Note that since we can embed the analog non-idealities 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 3b level. In our 3b MAC, $C_2$ is roughly $39 \times$ 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 non-idealities contribute errors in the matrix $A$ for both active and passive approaches but do not present non-linear effects for the passive approach. And for low-resolution multiplication (e.g. 3-bits), 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 as well. 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.

2.2 Circuit Implementation

For the reasons mentioned in Section 2.1, we implement our MAC core in the SCMM using the passive approach. The full SCMM implementation using the MAC is shown in Figure 3 where $C_s$ is $C_1$ and $C_{ASAR}$ is $C_2$ from Section 2.1. 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 6b input DAC is used as an on-chip test source. The MAC uses 300aF fringe capacitors for $C_s$. The SAR ADC uses top-plate sampling with its 6b cap-DAC $C_{ASAR}$ and digitizes the bit codes asynchronously to efficiently fit all bit decisions in the narrow time window.

The 6b input DAC which generates $V_{in}[i]$ is a 5b cap-DAC with a sign-bit and is shown in Figure 3(b). The digital control signals are shown in Figure 3(c). These signals, which are preloaded from local memory, are synchronized with the rising edge of global overlapping clocks $\phi_1$ and $\phi_2$.

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 multiplies and accumulates using the input DAC and MAC and 1 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 multiply-and-accumulates. Figure 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 6b input DAC generates a voltage $V_{in}[i]$ to be sampled during $\phi_1$, where this voltage is prepared during the $\phi_2$ phase. During the $\phi_2$ phase, the 6b memory for the input DAC prepares $V_{in}[i]$ by either recharging the input’s cap-DAC to $V_{DD}$ or discharging the previous cycle’s charge using $\phi_{2,refresh}$ signals. This resulting sampled charge $Q_{input}[i+1]$ generates $V_{in}[i+1] = \frac{Q_{input}[i+1]}{C_{DAC,tot}+C_s[i+1]}$
where the input DAC's cap-DAC is $35 \times$ larger than $C_s[i+1]$ such that $V_{in}[i+1]$ is independent of $C_s[i+1]$ at 6b. 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 Figure 5. During $\phi_1$, the input DAC generates a voltage that is sampled by the cap-DAC of the MAC, $C_s[i]$, which now holds $V_{in}[i]C_s[i]$. During $\phi_2$, this charge is redistributed onto $C_{s,tot}$ and $C_{ASAR}$. As stated earlier in Section 2.1, $C_{ASAR} \gg C_{s,tot}$, and thus most of the sampled charge $V_{in}[i]C_s[i]$ is pushed to $C_{ASAR}$. The residual charge that remains un-transferred 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, $\phi_3$ 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 shaves 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 non-monotonic binary search algorithm [31]. After the SAR algorithm, the 6b 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 offchip. 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 multiply and accumulate, allowing for sufficient setup times at 2.5GHz. The codesign judiciously constrains the
2.3 Effect of MAC’s incomplete charge transfer

In Section 2.1, 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 Equation 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 Figure 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 Figure 6(a). The output $y_1$ clearly accumulates in time compared to $y_2$ 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 the result of incomplete charge transfer. Figure 6(b) normalizes the running sums of the passive such that $y_1[64]$ (ideal) = $y_1[64]$ (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

$$\begin{align*}
\text{minimize} & \quad ||A - B\tilde{A}||_F \\
\text{subject to} & \quad B \in \Omega_B
\end{align*}$$

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 $\Omega_B$ is the set of possible values that $B$ can take on. For example, if the matrix $B$ were performed in fixed-point, $\Omega_B$ 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 Equation 3, we obtain the corrected matrix
multiplies \( y_1, y_2, y_3 \) as shown in Figure 6(c) and the error between ideal and corrected in (d). It is
important to note that the performance of factorization is a strong function of the aspect ratio of
\( B \in \mathbb{R}^{m \times n} \). Figure 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. Figure 8(a) illustrates the ideal matrix \( A \), and Figure 8(b,c) illustrate 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 \( C_{\text{ASAR}} \) to be only 10\( \times \) larger than the maximum \( C_s \) instead of 39\( \times \) in
our actual implementation.

2.4 Noise Analysis of MAC

The MAC system in Figure 3 is designed such that its output noise is well within the 6b SNR
target of the ADC. In spite of the 300aF 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 least significant bit (LSB) (7mV) 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 \( C_{\text{ASAR}} \) be
\( \sigma_{C_{\text{ASAR}}}[i] \) at time \( i = 1, \ldots, 64 \). During the first cycle, \( \phi_1 \) is turned on and off. This generates a
sampled noise charge \( kTC_s[i] \) on \( C_s[i] \). When \( \phi_2 \) is turned on, this charge is redistributed onto \( C_{s,\text{tot.}} \)
and \( C_{\text{ASAR}} \). The noise voltage across \( C_{\text{ASAR}} \) is then \( \frac{kTC_s[i]}{(C_{s,\text{tot.}}+C_{\text{ASAR}})^2} \). When \( \phi_2 \) is subsequently turned
off, this generates another sampled noise charge on \( C_{\text{ASAR}} \) of \( kT(C_{s,\text{tot.}}+C_{\text{ASAR}}) \). The total noise
voltage on \( C_{\text{ASAR}} \) after a complete \( \phi_1-\phi_2 \) cycle is \( \sigma_{C_{\text{ASAR}}}[i = 1] = \sqrt{\frac{kT(C_s[i])}{(C_{s,\text{tot.}}+C_{\text{ASAR}})^2} + \frac{kT(C_{s,\text{tot.}}+C_{\text{ASAR}})}{C_{\text{ASAR}}^2}} \).
After \( i \phi_1 - \phi_2 \) cycles, the noise voltage on \( C_{\text{ASAR}} \) becomes

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

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 this expression as

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

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

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

3 Chip Measurements and Applications

The SCMM was fabricated in 40nm CMOS, and its die photo is shown in Figure 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 ∈ \{1, \ldots, 8\}. Many applications arise where the matrix operation is an orthonormal transformation. These include Discrete Fourier and Cosine Transformations (DFT, DCT) and matched filtering. Here we apply such transformations to characterize matrix multiplication performance.

We measure $y_j = Ax^{(i)}$ where $x^{(i)} = a_i^T$ for 8 trials $i \in \{1, \ldots, 8\}$ and for 8 output channels $j \in \{1, \ldots, 8\}$ using a clock rate of 1GHz. Note that the $x^{(i)}$ is a vector, which is different than the scalar element $x[i]$, which denotes the value at time $i$. Ideal multiplication result is shown in Figure 11(a), and the measured output is shown in Figure 11(d) as raw digital codes and normalized in Figure 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. Figure 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 non-idealities in the circuit operation, we observe Figure 11(b), which clearly shows non-zero values of $y_j$ for $j \neq i$. The mean square error (MSE) of the ideal to the measured is 0.00723. Using matrix factorization as described in Section 2.3, we obtain Figure 11(c), which attenuates the cross-terms $y_j$ for $j \neq i$ and equalizes the diagonal terms $y_j$ for $j = i$.

We measure the system noise performance using a matched filtering setup at 1GHz. Here, we perform a single inner product $y = \sum_{i=1}^{64} a[i]x[i]$ 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 Figure 12(a) and perform 25 independent trials. The output voltage is derived from using the output codes and the LSB size (7mV) of the ADC outputs. We plot the mean and variance of the output for varying levels of 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 Figure 12(b) is similarly obtained by measuring the mean of the inner products of random vectors, which ideally converges to 0. The offset of the entire system due to leakage, comparator, and capacitor mismatch contributions is within half-LSB of the ADC.
output at speeds > 0.5GHz.

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 non-linearity functions. After training, the first layer of a network contains Gabor-like features that are used to detect edges. Figure 13 illustrates the ideal activations in (c) when 1 Gabor-like filter shown in (b) is scanned over the red channel of the colored input image in (a). This scanning process is performed by multiplying the input image with the filter for varying horizontal and vertical offsets. We obtain results from our chip (d) and simulated digital (e,f). The simulated digital (e) uses fixed-point arithmetic with 6b input, 3b weights, and 6b output notated as (6b/3b/6b), but with infinite accumulator resolution while (f) uses 6b/3b/6b with 6b accumulator.

We extend this primitive to reduce the A/D footprint when the first neural network layer operates on inherently 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 Figure 14. To reduce the dimensionality, the matrix-vector operations are processed at 1GHz in non-overlapping 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. The performance summary is shown in Table 1. 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 bits for input, 4 bits for weights, and 6 bits for outputs with full-precision accumulation and no overflow. The energy per operation of the proposed analog layer is 11× less than that of digital, which is simulated in 40nm. The synthesized digital MAC uses 6b input, 4b weights, and 6b outputs with a 6b 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 6b instead of \( 6 + 4 + \log_2(64) = 15 \) bits. Finally, the proposed analog layer lowers the number of A/D conversions by 21× as compared to conventional digital approach. The compute to memory read energy ratio is 1.18:1.

For our second application, we demonstrate analog co-processing and acceleration for computation that is 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 \( \theta \), the vector of unknown parameters that finds the global optimal solution by minimizing a user-defined objective function \( J(\theta) \), which is non-convex in general. Here, the SCMM is used to maximize 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 \( \alpha \) 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 6b 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 Figure 15. The measured
solution to the optimization problem is shown in Figure 16, which shows close alignment with
simulated digital. Analog acceleration using the SCMM at 2.5GHz performs slightly worse than
digital double-precision 64b and is equivalent to simulated digital fixed-point at an estimated 6×
lower energy and the compute to memory read energy ratio is 1.05:1.

Table 2 summarizes the performance of the analog charge-domain MAC for two applications
as compared with a recent work of embedding multiplication in a 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 8.7TOPS/W at 1GHz for application 1, and 7.7TOPS/W at
2.5GHz for application 2. As compared to [28] (130nm), our work (in 40nm) is 2 orders of mag-
nitude 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 higher multiplication resolution but which makes analog-domain ac-
cumulation difficult to realize.

4 Acknowledgements

The authors thank D. Miyashita, B. Murmann, M. Udell, D. Bankman, C. Young, K. Zheng, the
TSMC University Shuttle Program, and NSFGRFP and TI Stanford Graduate Fellowship.
5 Conclusion

This work presents the SCMM, which uses switches, 300aF 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-GHz MAC rates.
6 List of Figures and Tables

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

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

3. Figure 3. (a) SCMM implementation, (b) SCMMs 6b input DAC, and (c) digital controls.

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

5. Figure 5. The operational modes and timing diagram.

6. Figure 6. The ideal and simulated analog MAC transient outputs for a matrix operation $Ax$.

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

8. Figure 8. (a) A programmed 64x64 matrix $A$, (b) the matrix that is a result of incomplete accumulation $\tilde{A}$, and (c) the corrected matrix using matrix factorization.

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

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

11. Figure 11. (a) Ideal matrix-vector product output, (b) measured output, (c) corrected output, (d) measured digital code.

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

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

14. Figure 14. Designed compression and classifying layer for a small convolutional neural network and measured confidence levels for sampled input images on the CIFAR10 database.
15. Table 1. Performance of the compression layer versus conventional.

16. Figure 15. The classification accuracy as a function of the number of gradient steps in the stochastic gradient descent procedure. The NMSE of analog averaged over 100 steps is 0.006.

17. Figure 16. Final optimization solution after 100 gradient steps.

18. Table 2. Performance summary.
Figure 1: Analog matrix multiplication for various signal processing applications.

Figure 2: (a) Active analog MAC and (b) passive analog MAC implementations.
Figure 3: (a) SCMM implementation, (b) SCMMs 6b input DAC, and (c) digital controls.
Figure 4: (a) SCMM chip boundary when under test and (b) compute-memory architecture.
Figure 5: The operational modes and timing diagram.
Figure 6: The ideal and simulated analog MAC transient outputs for a matrix operation $Ax$.

Figure 7: Matrix factorization reconstruction error (normalized MSE) as a function of $m$. 
Figure 8: (a) A programmed 64x64 matrix $A$, (b) the matrix that is a result of incomplete accumulation $\tilde{A}$, and (c) the corrected matrix using matrix factorization.

Figure 9: Transient noise simulation of the noise voltage on $C_{\text{ASAR}}$. 
Figure 10: Photograph of the chip in 40nm CMOS.
Figure 11: (a) Ideal matrix-vector product output, (b) measured output, (c) corrected output, (d) measured digital code.
Figure 12: Measured matched filter response for varying input SNRs and the measured offset.

Figure 13: Measured versus simulated performance of a filter (b) on an image (a).
Figure 14: Designed compression and classifying layer for a small convolutional neural network and measured confidence levels for sampled input images on the CIFAR10 database.

|                      | 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

Figure 15: Table 1. Performance of the compression layer versus conventional.
Figure 16: The classification accuracy as a function of the number of gradient steps in the stochastic gradient descent procedure. The NMSE of analog averaged over 100 steps is 0.006.

Figure 17: Final optimization solution after 100 gradient steps.
| Technology | ISSCC 2015 [28] | This Work |
|------------|----------------|-----------|
| 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

Figure 18: Table 2. Performance summary.
References

[1] Rakesh Ginjupalli and Gaurav Khanna. High-precision numerical simulations of rotating black holes accelerated by cuda. *arXiv preprint arXiv:1006.0663v1*, 2010.

[2] Jie Han and Michael Orshansky. Approximate computing: An emerging paradigm for energy-efficient design. In *18th IEEE European Test Symposium (ETS)*, 2013, pages 1–6. IEEE, 2013.

[3] Renée St. Amant, Amir Yazdanbakhsh, Jongse Park, Bradley Thwaites, Hadi Esmaeilzadeh, Arjang Hassibi, Luis Ceze, and Doug Burger. General-purpose code acceleration with limited-precision analog computation. In *Proceedings of the 41st Annual International Symposium on Computer Architecture*, ISCA ’14, pages 505–516, Piscataway, NJ, USA, 2014. IEEE Press.

[4] Vincent Vanhoucke, Andrew Senior, and Mark Z. Mao. Improving the speed of neural networks on cpus. In *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*, 2011.

[5] Daniel Soudry Ran El-Yaniv Matthieu Courbariaux, Itay Hubara and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*, 2016.

[6] Daisuke Miyashita, Edward Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025*, 2016.

[7] Chris M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural Comput.*, 7(1):108–116, January 1995.

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[9] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
[10] Edward H Lee, Madeleine Udell, and S Simon Wong. Factorization for analog-to-digital matrix multiplication. In Proceedings of the 40th International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Brisbane, pages 1061–1065, 2015.

[11] Shoji Kawahito, M. Yoshida, M. Sasaki, K. Umehara, D. Miyazaki, Y. Tadokoro, K. Murata, S. Doushou, and A Matsuzawa. A CMOS image sensor with analog two-dimensional DCT-based compression circuits for one-chip cameras. IEEE Journal of Solid-State Circuits, 32(12):2030–2041, Dec 1997.

[12] M. Herman and T. Strohmer. Compressed sensing radar. In IEEE Radar Conference, 2008. RADAR ’08., pages 1–6, May 2008.

[13] Y. Oike and A. El Gamal. CMOS Image Sensor with Per-Column ΣΔ ADC and Programmable Compressed Sensing. IEEE Journal of Solid-State Circuits, 48(1):318–328, Jan 2013.

[14] D. Adams, C. S. Park, Y. C. Eldar, and B. Murmann. Towards an integrated circuit design of a Compressed Sampling wireless receiver. In 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 5305–5308, March 2012.

[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 65nm CMOS. In 2012 IEEE Radio Frequency Integrated Circuits Symposium (RFIC), pages 39–42, June 2012.

[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 Journal of Solid-State Circuits, 48(5):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 Journal of Solid-State Circuits, 40(8):1726–1735, Aug 2005.

[18] M. Lehne and S. Raman. A 0.13µm 1-GS/s CMOS Discrete-Time FFT processor for Ultra-Wideband OFDM Wireless Receivers. IEEE Transactions on Microwave Theory and Techniques, 59(6):1639–1650, June 2011.
[19] F. Rivet, Y. Deval, J. Begueret, D. Dallet, P. Cathelin, and D. Belot. The Experimental Demonstration of a SASP-Based Full Software Radio Receiver. *IEEE Journal of Solid-State Circuits*, 45(5):979–988, May 2010.

[20] S. Kirollos, J. Laska, M. Wakin, M. Duarte, D. Baron, T. Ragheb, Y. Massoud, and R. Baraniuk. Analog-to-Information Conversion via Random Demodulation. In *2006 IEEE Dallas/CAS Workshop on Design, Applications, Integration and Software*, pages 71–74. IEEE, 2006.

[21] O. Abari, F. Lim, F. Chen, and V. Stojanovic. Why Analog-to-Information Converters suffer in high-bandwidth sparse signal applications. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60(9):2273–2284, 2013.

[22] W. Yin, S. Morgan, J. Yang, and Y. Zhang. Practical compressive sensing with Toeplitz and circulant matrices. In *Visual Communications and Image Processing 2010*, pages 77440K–77440K. International Society for Optics and Photonics, 2010.

[23] Junjie Lu, Stephanie Young, Itamar Arel, and Jeremy Holleman. A 1 TOPS/W Analog Deep Machine-Learning Engine With Floating-Gate Storage in 0.13 μm CMOS. *IEEE Journal of Solid-State Circuits*, 50(1):270–281, 2015.

[24] Skylar Skrzyniarz, Laura Fick, Jinal Shah, Yejoong Kim, Dennis Sylvester, David Blaauw, David Fick, and Michael B Henry. A 36.8 2b-TOPS/W self-calibrating GPS accelerator implemented using analog calculation in 65nm LP CMOS. In *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 420–422. IEEE, 2016.

[25] Daisuke Miyashita, Ryo Yamaki, Kazunori Hashiyoshi, Hideo Kobayashi, Shouhei Kousai, Yukihito Oowaki, and Yasuo Unekawa. An LDPC decoder with time-domain analog and digital mixed-signal processing. *IEEE Journal of Solid-State Circuits*, 49(1):73–83, 2014.

[26] Ihab Nahlus, Eric P Kim, Naresh R Shanbhag, and David Blaauw. Energy-efficient dot product computation using a switched analog circuit architecture. In *2014 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 315–318. IEEE, 2014.
[27] Vaibhav Tripathi and Boris Murmann. Mismatch characterization of small metal fringe capacitors. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 61(8):2236–2242, 2014.

[28] Jintao Zhang, Zhuo Wang, and Naveen Verma. A matrix-multiplying ADC implementing a machine-learning classifier directly with data conversion. In *2015 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 1–3. IEEE, 2015.

[29] YP Tsividis and D Anastassiou. Switched-capacitor neural networks. *Electronics Letters*, 23(18):958–959, 1987.

[30] Daniel Bankman and Boris Murmann. Passive charge redistribution digital-to-analogue multiplier. *Electronics Letters*, 51(5):386–388, 2015.

[31] Vaibhav Tripathi and Boris Murmann. An 8-bit 450-MS/s single-bit/cycle SAR ADC in 65-nm CMOS. In *2013 Proceedings of the ESSCIRC (ESSCIRC)*, pages 117–120. IEEE, 2013.

[32] Dongsuk Jeon, Qing Dong, Yejoong Kim, Xiaolong Wang, Shuai Chen, Hao Yu, David Blaauw, and Dennis Sylvester. A 23mw face recognition accelerator in 40nm cmos with mostly-read 5t memory. In *VLSI Circuits (VLSI Circuits), 2015 Symposium on*, pages C48–C49. IEEE, 2015.

[33] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009.