A 64-Tile 2.4-Mb In-Memory-Computing CNN Accelerator Employing Charge-Domain Compute

Hossein Valavi, *Student Member, IEEE*, Peter J. Ramadge, *Fellow, IEEE*, Eric Nestler, *Member, IEEE*, and Naveen Verma, *Member, IEEE*

***Abstract*— Large-scale matrix-vector multiplications, which dominate in deep neural networks (DNNs), are limited by data movement in modern VLSI technologies. This paper addresses data movement via an in-memory-computing accelerator that employs charged-domain mixed-signal operation for enhancing compute SNR and, thus, scalability. The architecture supports analog/binary input activation (IA)/weight first layer (FL) and binary/binary IA/weight hidden layers (HLs), with batch normalization and input–output (IO) (buffering) circuitry to enable cascading, if desired, for realizing different DNN layers. The architecture is arranged as 8 × 8 = 64 in-memory-computing neuron tiles, supporting up to 512, 3 × 3 × 512-input HL neurons and 64, 3 × 3 × 3-input FL neurons, configurable via tile-level clock gating. In-memory computing is achieved using an 8T bit cell with overlaying metal-oxide-metal (MOM) capacitor, yielding a structure having 1.8× the area of a standard 6T bit cell. Implemented in 65-nm CMOS, the design achieves HLs/FL energy efficiency of 866/1.25 TOPS/W and throughput of 18876/43.2 GOPS (1498/3.43 GOPS/mm^2), when implementing convolution layers; and 658/0.95 TOPS/W, 9438/10.47 GOPS (749/0.83 GOPS/mm^2), when implementing convolution followed by batch normalization layers. Several large-scale neural networks are demonstrated, showing performance on standard benchmarks (MNIST, CIFAR-10, and SVHN) equivalent to ideal digital computing.**

***Index Terms*— Charge-domain compute, deep learning, hardware accelerators, in-memory computing, neural networks.**

I. Introduction

DEEP neural networks (DNNs) have achieved state-of-the-art performance in numerous inference applications, including image classification/detection, speech recognition, language translation, and so on. Although these have involved a range of different kinds of neural networks [multilayer perceptron (MLP), convolutional neural network (CNN), recurrent neural network (RNN), long short-term memory (LSTM)], the core computation in all cases is matrix-vector multiplication (MVM). We demonstrate MVM via a binarized [1] CNN accelerator, motivated by the widespread use of CNNs due to their high performance and their dominance of computations, arising from the trend of increasing network depths.

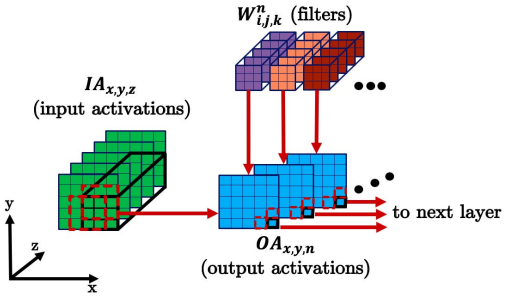
Manuscript received October 23, 2018; revised January 4, 2019 and February 5, 2019; accepted February 9, 2019. Date of publication March 5, 2019; date of current version May 24, 2019. This paper was approved by Associate Editor Vivek De. This work was supported in part by a gift from Analog Devices Inc. (ADI). *(Corresponding author: Hossein Valavi.)*

H. Valavi, P. J. Ramadge, and N. Verma are with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: hvalavi@princeton.edu).

E. Nestler is with Analog Devices Inc., Boston, MA 02110 USA.

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.2019.2899730



Key Operations:
1) inner-product: $PA_{x,y,n} = \sum_{i,j,k} IA_{x+i,y+j,z+k} W^n_{i,j,k}$
2) non-linear Activation Function: $OA_{x,y,n} = \rho(PA_{x,y,n})$

Fig. 1. Compute operations involved in each layer of a CNN.

As illustrated in Fig. 1, the primary operation in a CNN is convolution between input activation (IA) pixels $\mathrm{IA}_{x,y,z}$ arranged in a 3-D array and N 3-D filters $W^n_{i,j,k}$ (with n from 1 to N), whose weights are learned from training. This essentially amounts to inner product operations, yielding a pre-activation $\mathrm{PA}_{x,y,n}$ for each filter, which is then followed by a nonlinear activation function ρ, to yield an output activation $OA_{x,y,n}$ for each filter. The filters are then strided over all the IAs, generating a 3-D array of output activations, which makes up a feature map. State-of-the-art neural networks employ a large number of filters in each hidden layer (HL) (e.g., 64–512 in [2]) to capture a diversity of information in the feature map.

Recent demonstrations of CNN accelerators have shown that energy and delay are dominated by the data movement of bringing together a large number of IAs with weights and then redistributing a large number of output activations [3]–[18]. For instance, large-scale CNNs typically bring the data structures from off-chip memory and store them in embedded memory to exploit many opportunities for data reuse. However, even with embedded memory, we find that data-accessing energy typically exceeds compute energy by orders of magnitude [19]. Although spatial(systolic)-array architectures mitigate this by employing small buffers localized in processing elements, the buffers can consume significant energy and area, increasing the distance data must ultimately move across the array [3]–[5], [20].

To mitigate the fundamental costs of data movement, recently the concept of in-memory computing has been

proposed [12], [15]–[17], [21]–[41]. Rather than accessing raw data row by row in each column, as done in conventional memory, the objective is to *access a computational result* in each column over *many rows of data*, thereby amortizing the bit-line discharge energy and delay [16], [17], [22], [25]–[27], [31], [42]. Although this holds the potential for factors of energy and delay reduction up to the number of memory rows, this comes at the cost of computational SNR. As described in [22], computation over multiple inputs generally increases the required dynamic range, which must be fit within the constrained bit-line swing, thereby correspondingly degrading the computational SNR. In-memory-computing architectures typically employ mixed-signal operation to integrate compute in the constrained memory circuits by going beyond digital switch-based abstraction of bit-cell transistors. Thus, computational SNR is limited by non-ideal analog behaviors (non-linearities and variations).

To overcome this and substantially enhance the computational scalability of in-memory computing, we demonstrate an architecture achieving high computational SNR. This is achieved by using charge-domain computation based on highly linear and stable metal–oxide–metal (MOM) finger capacitors rather than current-domain computation, relying on bit-cell transistor transfer functions. Although MOM capacitors, enabled by excellent temperature/process stability/matching (shown to improve with technology scaling), address computational SNR, they can also be laid out above the bit-cell transistors, requiring no additional area. This paper is an extended version of [43], providing detailed explanation and analysis of the design and CNN demonstrations, especially analyzing sources of analog non-idealities and their impact on the architecture.

The remainder of this paper is organized as follows. Section II describes the overall system and how it forms DNNs. Section III describes the detailed architecture and circuit design. Section IV quantitatively demonstrates the benefits of charge-domain in-memory computation in 65-nm CMOS technology. Section V describes the prototype, basic measurements, and demonstrations of various neural networks for typical image-classification data sets. Finally, Section VI concludes this paper.

II. System Overview

Fig. 2 shows a deep binarized CNN, consisting of one first layer (FL) and potentially many HLs, all possibly separated by batch normalization layers. The demonstrated architecture supports all of these, especially emphasizing HLs due to the trend of increasing DNN depth. HLs receive binary IAs and multiply them with binary filter weights. On the other hand, FL receives high dynamic range IAs and multiplies them with binary filter weights. As described in Section III-B, our mixed-signal implementation has the ability to sample analog inputs. Thus, an architecture is explored where the FL can take analog IAs directly from an imager or sensor, potentially eliminating the need for explicit ADCs. Although multiple HLs can be implemented within one chip, by changing the filter weights and cycling input–output (IO) activations, the architecture

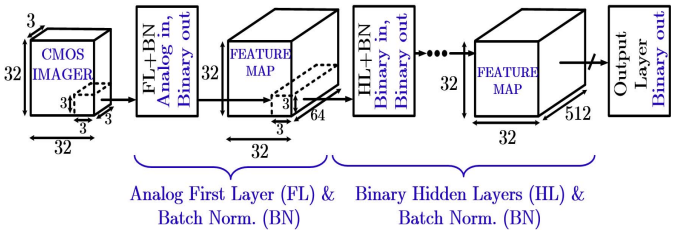


Fig. 2. Structure of a typical deep binarized CNN, with chip implementing convolutional FL and HLs. Chip supports up to 64, $3 \times 3 \times 3$ FL filters, and up to 512, $3 \times 3 \times 512$ HL filters.

also supports cascading chips into a high-throughput pipeline, enabled by circuitry for line buffering. In this case, it is presumed that IAs are provided row by row to conform to the data streaming of an input active-matrix imager, and the output feature maps are presumed to have dimensionality up to $32 \times 32 \times d$, where d is a configurable depth up to 512/64 for HLs/FL.

Furthermore, the architecture supports scalability and modularity, based on emerging trends in neural networks. Specifically, the filters are designed to have a convolutional stride of 1 and dimensionality of $3 \times 3 \times d$. This dimensionality is motivated because other filter sizes (e.g., 5×5, 7×7, etc.) can be readily realized by cascading 3×3 filters [44]. Although recent DNNs [2], [45]–[48] have reduced the use of pooling layers, max-pooling in a binarized CNN can be readily implemented via simple OR-logic, integrated into the proposed architecture within existing line buffers (described in Section III).

III. Architecture and Circuit Design

This section describes the detailed operation and implementation of the architecture. The operation starts with loading the configuration bits as well as the pre-trained weights onto the chip. Weight loading is simply performed via static random access memory (SRAM) read/write circuitry. This includes read/write scan chains, sense amplifiers, address decoder, and bit-line and word-line drivers. We start with operation when configured as binary-input HLs and then move to the operation when configured as analog-input FL.

A. Hidden-Layer Circuit Design

1) Overall Architecture: Fig. 3 shows the architectural block diagram of the HL configuration. The overall dataflow is as follows.

1) Presuming that one row (up to 32 pixels) of IAs is available at a time, first a pixel (of depth d up to 512) is streamed into an input shift register. This is included in the architecture to ease the testing interfaces; in an eventual cascade, it can be omitted, allowing for parallel loading.
2) The incoming pixel is then loaded in parallel into an IA SRAM. The IA SRAM serves as a line buffer, consisting of 4 sets of 512 columns, interleaved as shown, and having depth of 32. This allows pixels of corresponding depth to be loaded in one column for an incoming row, while pixels for the three other rows are being processed for 3×3 filtering.

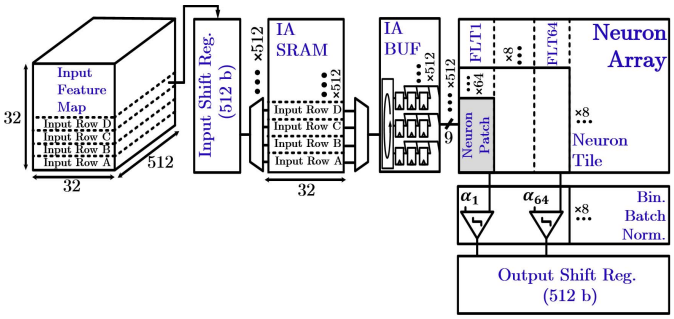


Fig. 3. Architecture of the chip, implementing one binary-input HL of a deep CNN.

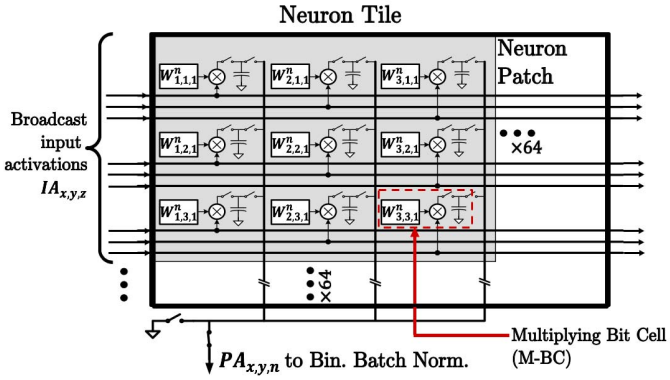
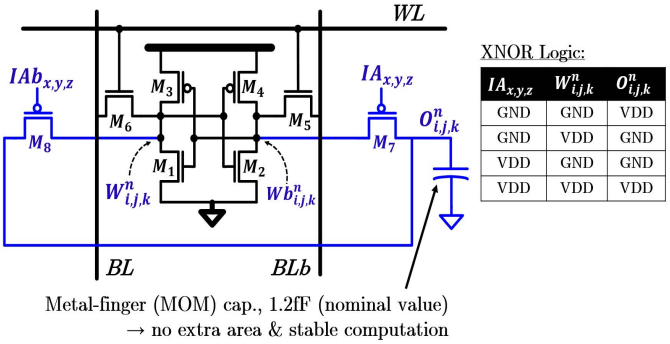


Fig. 4. Block diagram of a neuron tile. Each neuron tile includes 64 × 64 neuron patches, each containing 3 × 3 M-BCs.

3) Next, the IAs to be processed are shifted from the IA SRAM into an IA buffer (IA BUF). The IA BUF consists of 3-b shift registers with round-robin input interface, selecting three out of the four IA SRAM columns for processing (in circular manner to implement convolutional striding by one step).
4) Then, the $3 \times 3 \times d$ IAs to be filtered are broadcast in parallel over the neuron array. The neuron array consists of 8×8 neuron tiles, which provide clock gating for scalability of both the filter size and number of filters. Each neuron tile implements neuron filter segments, with $3 \times 3 \times 64$ inputs vertically, and 64 different filters horizontally. Thus, clock gating neuron tiles vertically scales the filter size, and clock gating horizontally scales the number of filters, to reduce the activity factor. In this way, up to 512 pre-activations can be computed in parallel, corresponding to one pixel, with depth up to 512.
5) The computed pre-activations are then fed to a binarizing batch normalization (Bin. Batch Norm.) block to compute the binary output activations for the pixel. Computed output activations are then streamed out, potentially directly feeding the next layer in a pipelined manner.

2) Neuron Tile: The dominating computation is performed in the neuron array, particularly in the neuron tile, using an in-memory-computing architecture. The structure of a neuron tile is shown in Fig. 4. Each neuron tile consists of 64×64 neuron patches, each of which processes 3×3 binary IAs. 64 neuron patches in one column belong to a single logical neuron filter, while 64 different columns correspond to different neuron filters. Within a neuron patch, each IA is then processed by a multiplying bit cell (M-BC). The M-BC multiplies the corresponding 1-b IA with a stored 1-b filter weight and stores the result as charge on a local capacitor. Then, all capacitors in one neuron filter are shorted together to perform charge accumulation, yielding the pre-activation via a multiplication-accumulation inner product operation.

3) Multiplying Bit-Cell: A key aspect of the architecture is to perform weight storage and multiplication in as dense an M-BC structure as possible. The M-BC circuit is shown in Fig. 5, where the binary values of -1 and $+1$ are represented by GND and V_{DD}, respectively. Although storage is



$IA_{x,y,z}$	$W^n_{i,j,k}$	$O^n_{i,j,k}$
GND	GND	VDD
GND	VDD	GND
VDD	GND	GND
VDD	VDD	VDD

Fig. 5. Structure of an eight transistor M-BCs, implementing XNOR between an IA ($IA_{x,y,z}$/$IAb_{x,y,z}$) and a filter element ($W^n_{i,j,k}$/$Wb^n_{i,j,k}$). Compared to a standard 6T bit cell, M-BC has two additional pMOS devices and an extra metal-finger MOM capacitor (shown in blue).

achieved using a standard 6T SRAM bit cell, two additional pMOS transistors, driven by the IA signal ($IA_{x,y,z}$/$IAb_{x,y,z}$), are added to implement XNOR. Then, the result is sampled as charge on a 1.2-fF (nominal value) MOM finger capacitor. The MOM capacitor is laid out above the bit cell, thus taking no additional area. Furthermore, the very good matching, temperature, and process coefficients for MOM capacitors enable highly linear and stable compute operation. Fig. 6 shows details of how the inner product operation is performed for three M-BCs, which are laid out together to enhance density.

1) *Reset Phase:* First, all capacitors are unconditionally discharged to GND by activating the tile-level shorting switches (via TSHORT/TSHORTb) and a discharge nMOS (via PRE) on the pre-activation node. This is done by keeping the IA signals ($IA_{x,y,z}$&$IAb_{x,y,z}$) high to deactivate the M-BC pMOS devices, thereby decoupling the bit-cell storage nodes.
2) *Binary-Multiply Phase:* Next, the IAs ($IA_{x,y,z}$/$IAb_{x,y,z}$) are driven differentially, activating only one pMOS in each M-BC, and causing the capacitor to charge up to V_{DD} XNOR conditionally, dependent on the IA ($IA_{x,y,z}$/$IAb_{x,y,z}$) and stored weight value ($W^n_{i,j,k}$/$Wb^n_{i,j,k}$).

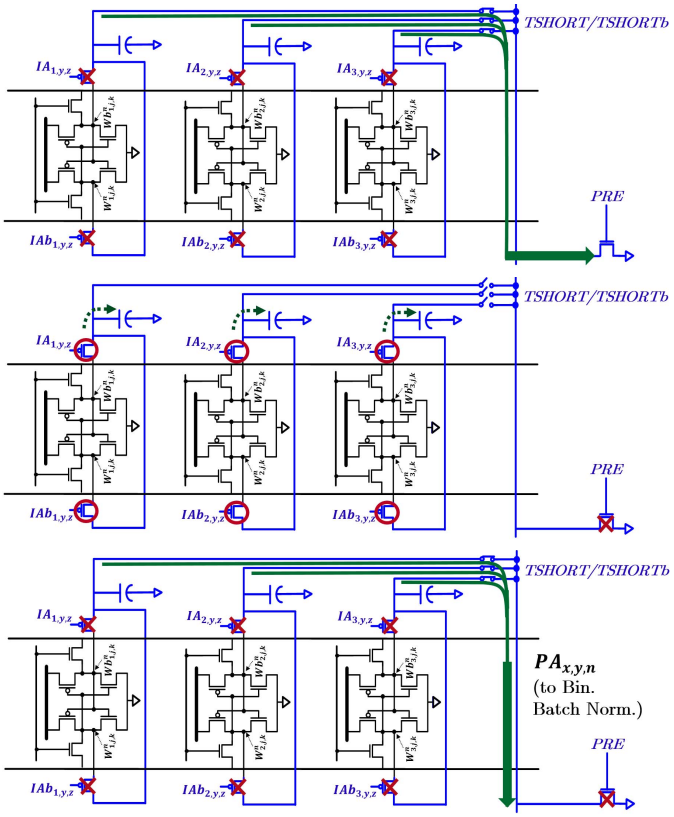


Fig. 6. XNOR computation by M-BCs, involving three signaling phases.

3) *Accumulate Phase:* Finally, the capacitors in one neuron filter are shorted together to generate the analog pre-activation value ($\mathrm{PA}_{x,y,n}$). Given the binary-value mapping of $-1/+1$ to GND/V_{DD}, $\mathrm{PA}_{x,y,n}$ takes one of nominally $(3 \times 3 \times d) + 1$ levels between GND and V_{DD}, centered at mid-rail. Note, the routing required across distributed capacitors introduces parasitics in the range of ∼10%, but these have a linear effect that is easily addressed through self-calibration (described in Section IV-C).

In order to maximize the potential of in-memory computing, the M-BCs must be made as dense as possible. Fig. 7 shows the layout of the 8T M-BC next to that of a standard 6T SRAM cell. Area of the M-BC is 1.8 μm^2, roughly 80% larger than the standard 6T cell, both laid out using logic rules (area of a 65 nm high-density 6T cell using foundry push-rules is 0.5 μm^2). pMOS transistors are chosen for the XNOR computation because this leads to a dense layout by matching the number of nMOS/pMOS devices. The gate-poly layout and pitch have been chosen with regularity in mind. Furthermore, the M-BC employs an MOM capacitor above the transistors, thus taking no additional area. Finally, the tile-level shorting switches are laid out together for three M-BCs, as shown in Fig. 8. This is because these share several nodes, enabling their dense layout together outside the M-BC.

An important consideration is the M-BC stability during neuron filtering operations. Starting by unconditionally discharging the local MOM capacitor means that the bit-cell

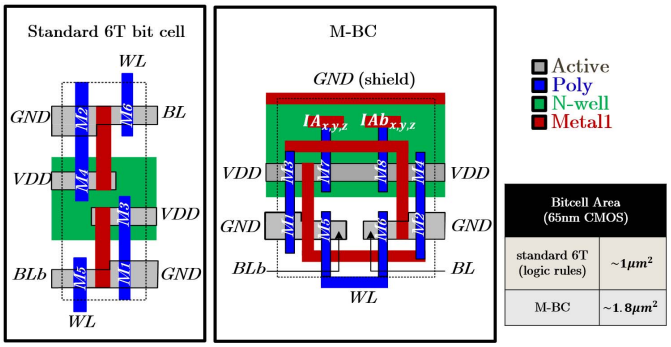


Fig. 7. Layout of M-BC compared to the layout of the standard 6T SRAM bit cell.

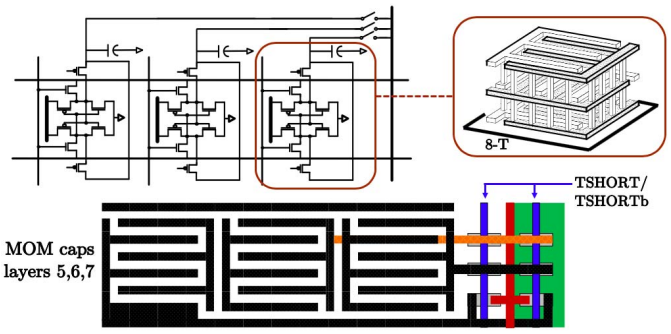


Fig. 8. Layout details of three M-BCs, sharing shorting switches.

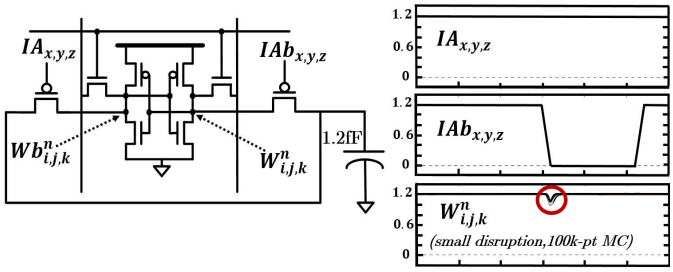


Fig. 9. Transient-stability analysis for XNOR operation of M-BC (MC sims).

storage nodes are exposed to a pull-down condition. However, Monte Carlo (MC) simulation shown in Fig. 9 confirms that this will not disrupt stored data, which is held high by the pull-up pMOS transistors in the 6T portion of the bit cell, for two reasons. First, the pull-down path is weak, since it involves a pMOS transistor for XNOR computation. Second, the small MOM capacitance of ∼1.2 fF does not invoke a static pull-down condition. Thus, the transient 100 k-point MC simulation shows that the disruption to a high storage node is small, for the minimum-sized pMOS transistors used.

4) Binarizing Batch Normalization: After computing the pre-activation values via up to 512 parallel neuron filters, batch normalization and a binarizing activation function are applied, in one step. Equation 1 shows the operation required for batch normalization [49] and application of the activation function ρ. For the special case of a binarizing activation function, the scaling parameter γ_n can be ignored since it does not change the sign, leaving only the offset-causing

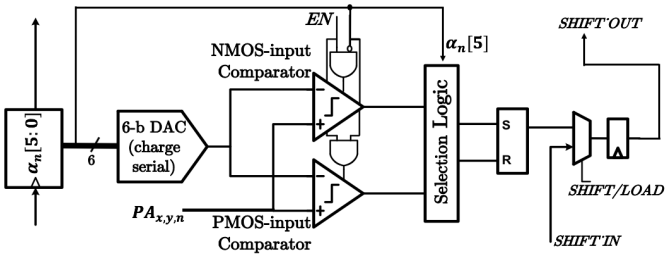


Fig. 10. Details of the Bin. Batch Norm. circuit (512 such circuits implemented within Bin. Batch Norm. block).

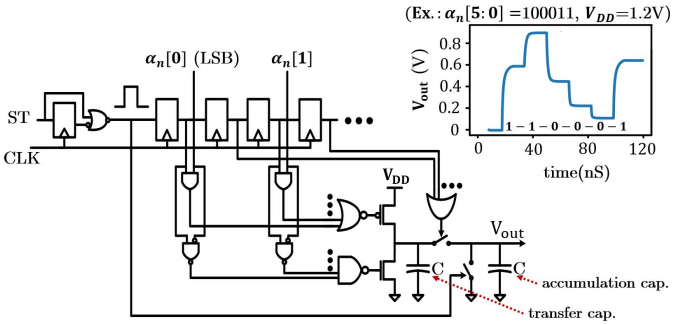


Fig. 11. Circuit implementation of each 6-b DAC.

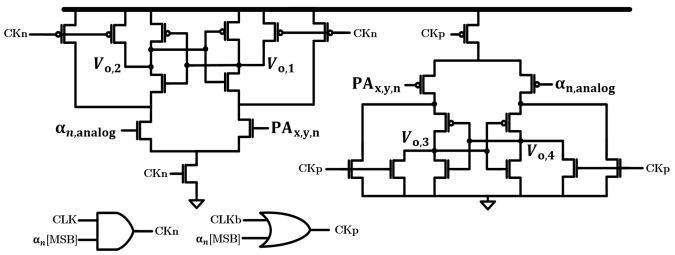


Fig. 12. Circuit implementation of the nMOS/pMOS-input comparators.

parameters, which can be combined into the single parameter α_n, as shown in 2. Therefore, applying batch normalization and the activation function reduces to sign comparison between the pre-activation and an analog reference, derived from training

$$\mathrm{OA}_{x,y,n} = \rho\left(\gamma_n \frac{\mathrm{PA}_{x,y,n} - \mu_n}{\sigma_n^2} + \beta_n\right) \quad (1)$$

$$\mathrm{OA}_{x,y,n} = \mathrm{sign}(\mathrm{PA}_{x,y,n} - \alpha_n). \quad (2)$$

Fig. 10 shows the circuit used for each of the 512 filters to implement this. It consists of a 6-b digital to analog converter (DAC) for converting α_n from a digital code into an analog reference, where the precision requirement of 6 bits was determined from simulations. This reference, as well as the computed pre-activation, then feeds two comparators. The reason for two comparators, one with nMOS inputs and one with pMOS inputs, is that both the pre-activation and the analog α_n can range from GND to V_{DD}. Hence, to ensure fast and robust regeneration, the MSB of the digital α_n code is used to select the comparator which is ensured to have a high input overdrive. The binarized output activation is then streamed out using an output shift register.

Fig. 11 shows the circuit used to implement the 6-b DAC. Given the need to fit 512 DACs, each in the pitch of 3 M-BCs, area was a primary consideration, motivating a serial charge-redistribution structure, described in [50]. Here, the LSB of the input code is applied first, to charge or discharge a transfer capacitor, which then transfers its charge to an equal-sized accumulation capacitor. Thus, the LSB charge is attenuated with binary weighting through each charging and shorting cycle. For instance, considering a code of $\alpha_n[5:0] =$ 6'b100011: $\alpha_n[0] = 1$ charges the transfer capacitor to V_{DD}, causing V_{out} to go to $(1/2) \times V_{\mathrm{DD}}$ (after charge-sharing); $\alpha_n[1] = 1$ charges the transfer capacitor to V_{DD}, causing V_{out} to then go to $(3/4) \times V_{\mathrm{DD}}$; $\alpha_n[2] = 0$ charges the transfer capacitor to GND, causing V_{out} to then go to $(3/8) \times V_{\mathrm{DD}}$; and so on.

Fig. 12 shows the nMOS/pMOS-input comparators used for comparing a computed pre-activation ($\mathrm{PA}_{x,y,n}$) with an α_n analog reference voltage ($\alpha_{n,\mathrm{analog}}$). Although a range of comparator circuits can be considered to enable rail-to-rail inputs, two standard nMOS/pMOS-input StrongARM-latch comparators are used [51]. These are selected for their power and area efficiency, as well as robust regeneration. The clocked comparators start by resetting to a common voltage (GND/V_{DD}) for all nodes in the regenerative differential branches. Then, the comparator selected via the MSB of α_n is enabled, turning off the reset transistors and turning on the tail transistor for the differential pair. Furthermore, the differential current biasing the regenerative branches is dependent on the differential input, which, in turn, generates differential rail-to-rail output voltages set by the polarity of $\mathrm{PA}_{x,y,n} - \alpha_{n,\mathrm{analog}}$.

5) Summary of Data Movement: As discussed in Section I, CNN-accelerator energy is typically dominated by data movement of IAs, weights, and output activations. Having explained the chip architecture, here, we summarize how each of these sources is eliminated or minimized.

1) Regarding IAs, these must be broadcast over all the neuron filter hardware. This distance is minimized, thanks to the high-density M-BCs, which form the parallel filters.
2) Regarding weights, their movement is eliminated as they are stationary within the M-BCs. Furthermore, a significant number of weights can be stored on-chip in the filtering hardware (2.4 Mb in this architecture), again thanks to the high-density M-BCs.
3) Regarding output activations, these are computed in a distributed way via passive charge redistribution in the M-BC capacitors. Thus, they are available at the output of the architecture at the cost of toggling a 1-b switch-control signal. Note that the output activations are, in fact, very high dynamic range signals; a digital implementation would require communicating a signal of over 12 bits [i.e., $\log_2(3 \times 3 \times 512) > 12$] in this design. Therefore, communication via the single-bit switch control signal is substantially more energy efficient.

B. First Layer Circuit Operation

Having described the operation of the binary-input HLs, in this section, we describe how the same architecture can

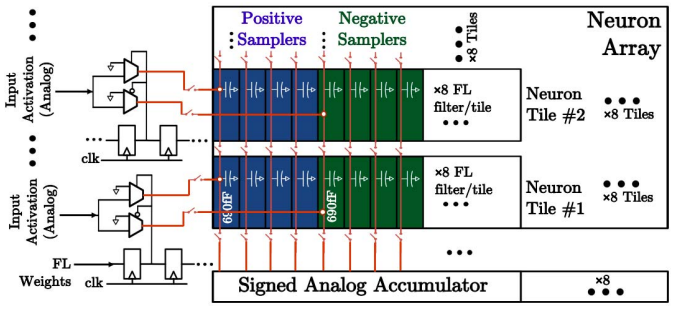


Fig. 13. Block diagram of FL, based on positive/negative samplers for each filter and each analog IA.

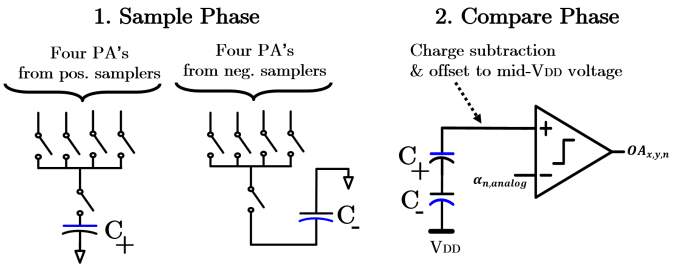


Fig. 14. Details of SAA block.

be configured to implement the analog-input FL. With analog inputs, the XNOR-logic compute model of multiplication in the M-BC cannot be exploited. Furthermore, fidelity of a high dynamic range input signal must be ensured. To do this, the architecture is configured, as depicted in Fig. 13, into filters having large positive and negative sampling structures. Specifically, by deactivating the binary IA signals of the M-BC and activating the tile-level shorting switches (TSHORT/TSHORTb), all the capacitors of each filter segment within one neuron tile get configured as a single logical sampling capacitor, of approximately 690 fF. Now, given that the filter weights are still binarized, filter segments are designated as positive and negative samplers, i.e., each analog IA is sampled on its positive sampler if the corresponding weight is $+1$, and on the negative sampler if the weight is -1, while holding the alternate sampler at GND. Furthermore, the input-layer filters are of size $3\times3\times3$ (presuming red-green-blue (RGB) imager pixels), amounting to 27 analog inputs. This means 27 positive samplers and 27 negative samplers are needed to implement each FL filter. In the architecture's 8×8 array of neuron tiles, there exist eight filter segments per column. Thus, four columns for the positive sampler and four columns for the negative sampler are designated for each FL filter. Filtering then simply requires adding the charge from all the positive samplers of each filter and subtracting the charge from all negative samplers of each filter.

Charge addition and subtraction are performed by the signed analog accumulator (SAA) block, as shown in Fig. 14. The SAA operates in two phases. In the sample phase, the charge from four columns of positive samplers is shared to the top plate of one capacitor, and the charge from four columns of negative samplers is shared to the bottom plate of another, equal-valued capacitor. Then, in the compare phase, signed summation of the charge is achieved by switching the capacitors into the configuration shown. This adds the positive charge, subtracts the negative charge, and simultaneously offsets the output voltage such that zero net charge yields mid-scale. Then, comparison with an α_n analog reference value from training is performed as before, via the Bin. Batch Norm. block.

IV. Precision Analysis of Charge-Domain In-Memory Computing

Virtually all in-memory-computing architectures reported thus far, based on both SRAM and emerging nonvolatile technologies, have employed current-domain computation, relying on the current transfer function of bit-cell MOSFETs, followed by accumulation on the bit lines [16], [17], [22], [25]–[27], [31], [42]. Although this enables amortization of bit-cell data into a computational result, thereby yielding in-memory-computing gains, it suffers from substantial computational noise. As previously mentioned, computation over a large amount of data stored in the bit cells generally increases the dynamic range required. However, compute based on MOSFET current transfer functions is highly susceptible to variation and nonlinearity, thus degrading the SNR, and substantially limiting the computational scale (level of amortization) achievable.

Here, we analyze the computational noise arising in charge-domain in-memory computing due to analog non-idealities. We show that very low computational noise can be achieved, both due to the excellent inherent matching characteristics of MOM capacitors and due to readily implemented self-calibration techniques. Thus, together, these substantially enhance the accuracy and scale of computation that can be achieved. We proceed by analyzing: 1) the effect of MOM-capacitor mismatch on the computed $\mathrm{PA}_{x,y,n}$; 2) the effect of thermal noise on the computed $\mathrm{PA}_{x,y,n}$; and 3) the effect of switch-induced charge-injection errors on $\mathrm{PA}_{x,y,n}$.

A. Effect of Mismatch

Our analysis here primarily focuses on uncorrelated capacitor mismatch. Although in general correlated mismatch (e.g., gradients) may also be present, we examine that via prototype measurements, due to limited foundry-provided models for such effects. With regards to uncorrelated capacitor mismatch, arising from lithography and oxide film-thickness variations, very good matching characteristics are generally observed [52], [53]. For instance, relying on previously published data, in 32-nm CMOS, single-layer 1.2-fF capacitors show a standard deviation of 0.8% [52], and in 180-nm CMOS, 0.5-/1.0-/2.0-fF capacitors show a standard deviation of 0.3%/0.2%/0.12% [53]. Similar estimates can be derived for the target 65-nm CMOS process employed, by extrapolating from data for larger capacitors.

To evaluate the impact of mismatch on $\mathrm{PA}_{x,y,n}$, we model each M-BC MOM capacitor's value as a Gaussian random variable with mean 1.2 fF and standard deviation σ_c. Furthermore, we assume that the result of XNOR computation yields an output with Bernoulli distribution with parameter p (p is

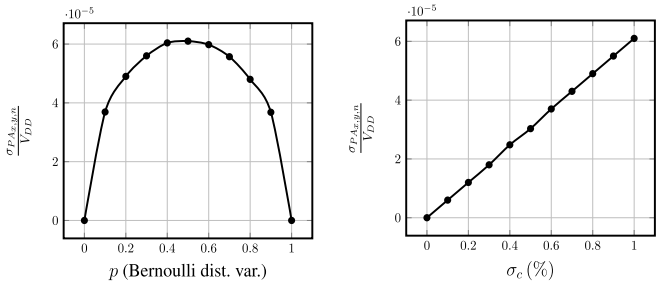


Fig. 15. (a) Impact of the Bernoulli distribution's parameter p on $\sigma_{\mathrm{PA}_{x,y,n}}$. (b) Impact of σ_c on $\sigma_{\mathrm{PA}_{x,y,n}}$ (100 k-point MC sampling).

the probability that the result is 1). Thus, for a filter of size $3 \times 3 \times 512$, $\mathrm{PA}_{x,y,n}$ can be calculated as follows:

$$\mathrm{PA}_{x,y,n} = V_{\mathrm{DD}} \frac{\sum_{i=1}^{3\times3\times512} c_i v_i}{\sum_{i=1}^{3\times3\times512} c_i} \quad c_i \sim \mathcal{N}(1.2, \sigma_c), \;\; v_i \sim \mathcal{B}(p).$$

For instance, assuming $\sigma_c = 1\%$, Fig. 15(a) shows the normalized standard deviation of $\mathrm{PA}_{x,y,n}$ (i.e., $(\sigma_{\mathrm{PA}_{x,y,n}}/V_{\mathrm{DD}})$) as a function of p, using 100 k-point MC sampling. As expected, the worst case occurs at $p = 0.5$. Thus, assuming $p = 0.5$, Fig. 15(b) shows $(\sigma_{\mathrm{PA}_{x,y,n}}/V_{\mathrm{DD}})$ for different values of σ_c, relevant for the expected level of mismatch. We can now compare this to the targeted $\mathrm{PA}_{x,y,n}$'s dynamic range, which is $(3 \times 3 \times 512 + 1)$ for the size of neuron filters employed in the architecture. Thus, we see that MOM-capacitor mismatch is well below the required level, even for the relatively large-scale in-memory computing demonstrated. In fact, as an example, for $\sigma_c = 0.5\%$, the compute error due to MOM-capacitor mismatch is at the level of dynamic range corresponding to accumulation over 160 k M-BC XNOR outputs.

B. Effect of Thermal Noise

The thermal noise of an *RC* circuit, corresponding to charge sharing among the M-BCs in one neuron filter, is determined by the kT/C limit. With a nominal value of 1.2 fF for each MOM capacitor, this amounts to $\sim 3.4 \times 10^{-6}$ V^2 (at room temp.). Assuming thermal noise on the M-BC capacitors that are independent random variables, the impact of noise on $\mathrm{PA}_{x,y,n}$ $(\sigma_{\mathrm{PA}_{x,y,n}})$ can be calculated according to $((KT/C)/(3 \times 3 \times 512))^{(1/2)} \sim 2.7 \times 10^{-5}$, almost an order of magnitude lower than the compute resolution for the targeted dynamic range of $\mathrm{PA}_{x,y,n}$. Thus, the impact of thermal noise on the architecture is also negligible. In fact, the compute error due to thermal noise is at the level of dynamic range corresponding to accumulation over 420 k M-BC XNOR outputs.

C. Effect of Charge-Injection Error

The M-BC operation described in Section III-A3 has some susceptibility to charge-injection error. Charge-injection-insensitive switching schemes were considered (similar to those used for standard SAR ADCs and also recently used for analog computing [18]). However, after determining that the error can be made small, the switching scheme employed was selected to minimize the number of M-BC transistors, as M-BC density is an important consideration for effective in-memory computing. Specifically, error arises during the reset and binary-multiply phases primarily due to the TSHORT/TSHORTb switches and M-BC pMOS (M_7/M_8) switches shown in Fig. 5. However, during these phases, the switches are deterministically biased at either GND or V_{DD}, thus their charge-injection error does not cause non-linearity. Error arises during the accumulate phase due to the TSHORT/TSHORTb switching, and biasing here varies with the pre-activation value, thus causing non-linearity. Although this is found to be small for the 6-b Bin. Batch Norm. resolution targeted, it can be further addressed by self-calibration. Self-calibration is readily performed via the Bin. Batch Norm. block itself, by adjusting the DAC codes. Since the DAC sets the comparator switching threshold, the switching thresholds can be selected to be linearly spaced with respect to the pre-activation values. The required DAC codes are determined as follows. All $+1$s are loaded in the M-BCs, and the number of IAs set to $+1$ (versus -1) is swept, to nominally give a pre-activation ramp (although $\mathrm{PA}_{x,y,n}$ is actually affected by non-linearity). At the desired nominal pre-activation values, the DAC code is determined which causes the comparator output to switch, thus linearizing the pre-activation value with respect to the switching thresholds. Such calibration also corrects non-linearity arising due to input-dependent comparator offset within the Bin. Batch Norm. block, as well as linearity error arising, for instance, due to fixed routing parasitics. Such self-calibration, not requiring any external reference, could in general be invoked via a periodic calibration routine if desired (though the presented measurements are for calibration performed once only).

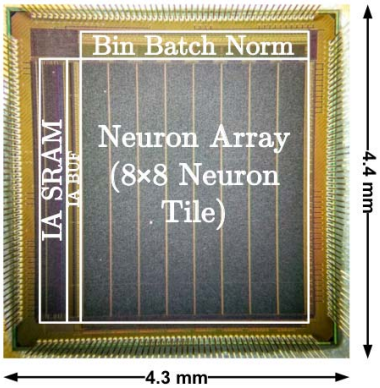
V. System Prototype and Demonstration

Fig. 16 shows a die photograph of the custom IC in 65-nm CMOS implementing the architecture, along with a summary of the measured performance for both HL and FL modes. The 8×8 array of neuron tiles corresponds to a total of 2.4 Mb of on-chip weight storage. The master clock frequency is 100 MHz and the nominal supply voltage is 1.2 V, although voltage reduction is also characterized, affecting energy efficiency and throughput.

Fig. 17 shows a block diagram of the measurement and demonstration setup used. A host PC provides a Python interface for sending instructions to and from a field-programmable gate array (FPGA) board (Xilinx Virtex-7). The FPGA has an embedded microcontroller for receiving data and instructions via Ethernet, and for triggering register transfer level (RTL) state machines, which generate digital waveforms to the prototype for performing different operations (such as loading configuration scan chains, writing weight data, performing neuron filtering, etc.). The following sections describe how this is used to perform detailed circuit-level characterization and neural-network application demonstrations.

A. Detailed IC Characterization

This section describes the detailed testing and characterization of the prototype.



Chip Summary	
Technology	65nm CMOS
Area	12.6 mm^2
CLK Frequency	100MHz
Bit Precision	1b
on-chip Memory	2.4Mb
Throughput (HL/FL)	18876*/43.2 GOPS[1] 9438*/10.47 GOPS[2]
Energy Efficiency (HL/FL)	866*/1.25 TOPS/W[1] 658*/0.95 TOPS/W[2]

[1] Convolution layers * Binary operations
[2] Convolution + Batch Normalization layers

Fig. 16. Prototype die photo in 65-nm CMOS (top) and the measurement summary (bottom).

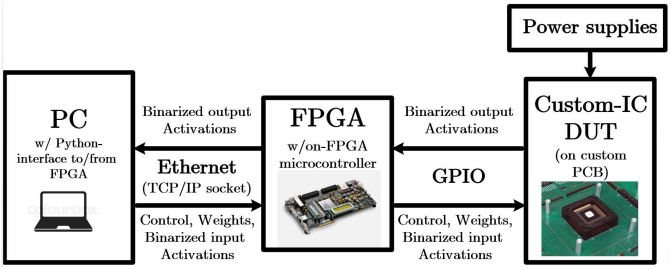


Fig. 17. Block diagram of the demonstration setup.

1) Computation: Fig. 18(a) shows the transfer function of the binary-input HLs (after self-calibration). The x-axis corresponds to the nominal value of the pre-activation, which is known from the values of the digital weights and IAs. The y-axis corresponds to the value of the digital α_n code, which causes the output activation to transition from -1 to $+1$. The transfer function is obtained by applying different (random) capacitor combinations, demonstrating that the transfer function is highly linear and stable. The error bars show the tight standard deviation across the 512 neuron filters. Similarly, Fig. 18(b) shows the transfer function of the analog-input FL. Similarly, the computation is highly linear and stable, with error bars showing tight standard deviation over 64 first-layer filters.

2) Energy Efficiency: Fig. 19 shows the energy breakdown of the HL, performing continuous filtering operations at the nominal voltage of 1.2 V. Batch Norm. corresponds to the energy of the Bin. Batch Norm. block (dominated by the 6-b DACs), M-BC corresponds to the energy of XNOR-conditional

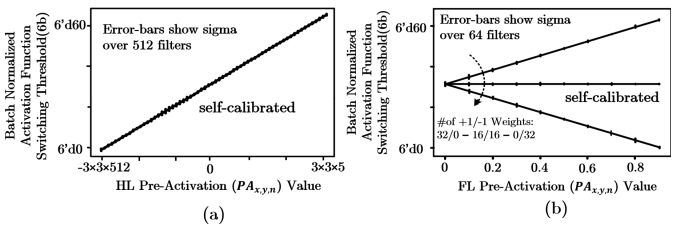


Fig. 18. Transfer function of (a) HL and (b) FL configurations.

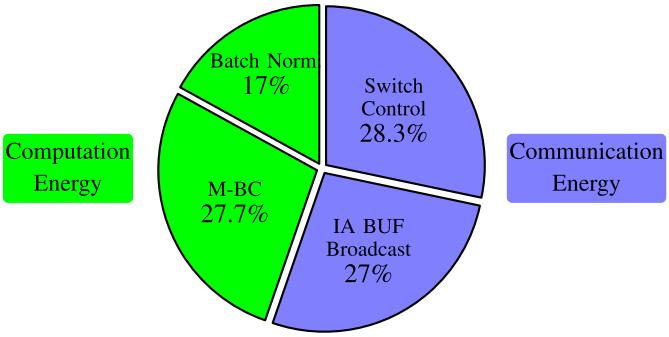


Fig. 19. Energy breakdown of the chip performing continuous filtering operation at the nominal voltage of 1.2 V.

charging of the M-BC MOM capacitors, IA BUF Broadcast corresponds to the energy of broadcasting IAs over the neuron array, and switch control corresponds to the energy of driving the tile- and neuron-level capacitor-shorting switches for charge accumulation. Switch-control and IA BUF Broadcast are data-movement components, while Batch Norm. and M-BC are computation components. As seen, data movement dominates, but only slightly, given the architectural approaches taken to address communication energy. Furthermore, Fig. 20 shows the energy breakdown when the architecture operates at its energy-optimal voltage. The optimal voltage is found to be 0.68 V for the IA BUF Broadcast and switch control and 0.94 V for M-BC and the Bin. Batch Norm. block. As seen, at the energy-optimal voltage, the communication energy is lower than the computation energy. Furthermore, in HL mode, a filtering operation of size $3 \times 3 \times 512$ is found to consume 10.6 pJ without the Bin. Batch Norm. and 14.0 pJ with the Bin. Batch Norm. block. This corresponds to $(2 \times 3 \times 3 \times 512/10.64 \times 10^{-12}) = 866$TOPS/W and $(2 \times 3 \times 3 \times 512/14.0 \times 10^{-12}) = 658$TOPS/W, respectively. Similarly, in FL, a filtering operation of size $3 \times 3 \times 3$ consumes 43 pJ without the Bin. Batch Norm. and 56.6 pJ with the Bin. Batch Norm. block. This corresponds to $(2 \times 3 \times 3 \times 3/43 \times 10^{-12}) = 1.25$TOPS/W and $(2 \times 3 \times 3 \times 3/56.6 \times 10^{-12}) = 0.95$TOPS/W.

3) Throughput: In HL mode, we assigned 25 clock cycles for a filtering operation (10 for Reset, 10 for Binary-multiply, 5 for Accumulate) and 25 clock cycles for the batch normalization operation, although we point out that pipelining of some operations is possible. However, assuming no pipelining, with the 10-ns clock period, filtering operations are performed with a throughput of $(512 \times 2 \times 3 \times 3 \times 512/25 \times 10^{-8}) = 18876$GOPS, while filter and batch normalization is performed with a throughput of $(512 \times 2 \times 3 \times 3 \times 512/50 \times 10^{-8}) =$

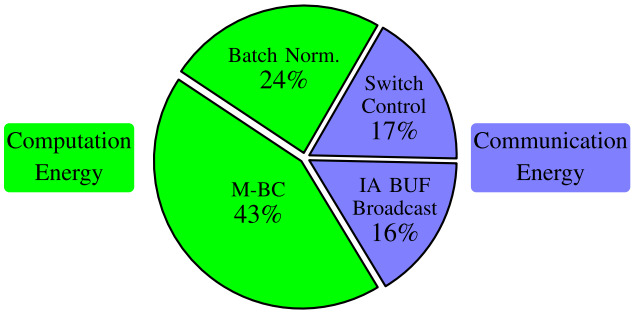


Fig. 20. Energy breakdown of the chip when blocks operate at their energy-optimal voltage levels.

TABLE I
COMPARISON TABLE

	Chen [5]	Moons [54]	Bang [55]	Ando [12]	Bankman [18]	**This work**
Technology	65nm	28nm	40nm	65nm	28nm	**65nm**
Area (mm^2)	16	1.87	7.1	12	6	**12.6**
Operating V_{DD}	0.8-1.2	1	0.63-0.9	0.55-1	0.8/0.8 0.6/0.53	**0.94/0.68/1.2**
Bit Precision	16b	4-16b	6-32b	1b	1b	**1b**
on-chip Mem.	108kB	128kB	270kB	100kB	328kB	**295kB**
Throughput (GOPS)	120	400	108	1264	400 (60)	**18876**
TOPS/W	0.0096	10	0.384	6	532 (772)	**866**
GOPS/mm^2	7.5	214	15	105	67 (10)	**1498**

9438GOPS. Similarly, in the FL mode, we assign 8 clock cycles for a filtering operation. This amounts to the throughput of $(64 \times 2 \times 3 \times 3 \times 3/8 \times 10^{-8}) = 43.2$GOPS (filtering operation) and $(64 \times 2 \times 3 \times 3 \times 3/33 \times 10^{-8}) = 10.47$GOPS (filter and batch normalization), respectively.

4) Comparison With Prior Work: Table I shows a comparison table of recently presented neural network accelerators. As seen, this paper achieves the highest energy efficiency and throughput. It should be noted, however, that the architecture performs binary operations, while some other accelerators support higher precision. The energy efficiency and throughput are summarized in the scatter plot shown in Fig. 21.

B. Neural Network Demonstrations

For demonstration, we have mapped several DNNs to the architecture, where the layers are mapped one at a time to the chip. Table II summarizes the performance of different networks on three standard data sets, where the HLs are mapped to the IC. The different network configurations used are shown at the bottom of the table. As seen, visual geometry group (VGG)-style networks [44] of practical complexity are used. The testing and validation performance are shown at the top, both for the architecture and for an ideal software implementation. It can be seen that the chip and an ideal pure-digital software (SW) implementation achieve essentially equivalent performance. The energy numbers are measured (via power supply current) from the chip. The throughput numbers are estimated from the filter and Bin. Batch Norm. characterization, as actual demonstration throughput is limited by the FPGA-based testing setup (Fig. 17), where, for each

TABLE II
ACCURACY COMPARISON FOR DATA SETS & DNNS

	MNIST	CIFAR-10	SVHN
Test Accuracy (chip / SW)	98.60% / 98.92%	83.27% / 83.50%	94.35% / 95.10%
Validation Accuracy (chip / SW)	98.58% / 98.75%	84.09% / 84.37%	94.03% / 94.63%
Baseline Neural Network	64 CONV3 - BN 64 CONV3 - BN 128 CONV3 - BN 128 CONV3 - BN 10 FC	64 CONV3 - BN 64 CONV3 - BN 128 CONV3 - BN 128 CONV3 - BN 256 CONV3 - BN 256 CONV3 - BN 1024 FC - BN 1024 FC - BN 10 FC	64 CONV3 - BN 64 CONV3 - BN 128 CONV3 - BN 128 CONV3 - BN 256 CONV3 - BN 256 CONV3 - BN 1024 FC - BN 1024 FC - BN 10 FC
Operating V_{DD} (V)	0.68/0.94	0.68/0.94	0.68/0.94
#Operations/Classification (MACs*) (Bin. CONV. layers)	0.528×10^9	2.34×10^9	2.34×10^9
Energy/Classification (Bin. CONV. layers)	0.8μJ	3.55μJ	3.55μJ
Throughput (fps**) (Bin. CONV. layers)	651	390	390

k CONV3 - BN: indicates k 3×3 filters followed by the batch norm. layer.
k FC - BN: indicates k fully-connected layers followed by the batch norm. layer.
k FC: indicates k fully-connected layers.
*MACs: multiply and accumulate operations.
**fps: frames per second.

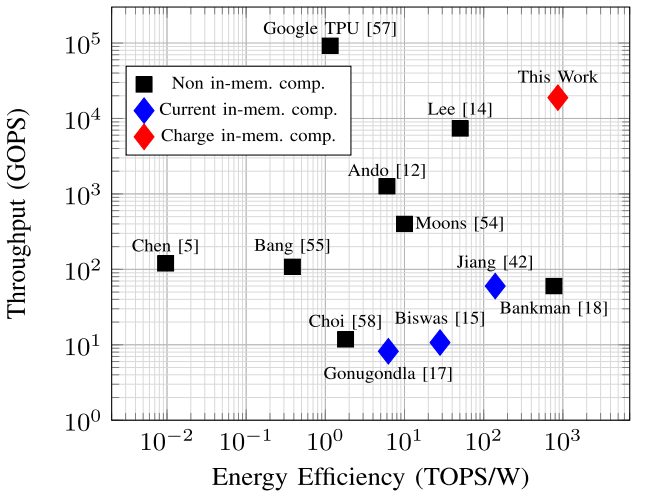


Fig. 21. Scatter plot, comparing the chip with other recently presented neural network accelerators. All chips are fabricated in 65-nm CMOS, except Google TPU [57], Bankman *et al.* [18], and Moons *et al.* [54] that are fabricated in 28-nm CMOS, and Bang *et al.* [55] implemented in 40-nm CMOS. Furthermore, Bankman *et al.* [18], Ando *et al.* [12], Biswas and Chandrakasan [15], Jiang *et al.* [42], and this work are designed for binarized neural networks, Gonugondla *et al.* [17] implements the support vector machine (SVM), and the rest implement multi-bit neural networks.

layer, IAs are provided by the FPGA via an input shift register and output activations are captured by the FPGA via an output shift register, one pixel at a time (as described in Section III-A1); the shift registers are required to reduce the chip's IO, but for layer-by-layer computation, they limit the demonstration throughput. We note that in an eventual system, the throughput will also depend on the utilization of in-memory-computing cores, which strongly depends on the overall architecture. Recent work has begun to explore one approach to such a complete architecture, providing utilization analysis [56].

VI. CONCLUSION

The energy and delay in large-scale neural-network accelerators, dominated by high-dimensional MVMs, are limited by data movement in modern VLSI technologies. This has motivated the concept of in-memory computing, where computation is performed in-place within bit cells where matrix

elements are stored, and where input vector elements need be transmitted minimal distance, across the high-density bit-cell arrays. However, the challenge with in-memory computing is accuracy and scalability, due to the need for analog operation, to integrate compute in constrained bit-cell circuits, and the need for increased dynamic range, over many bit-cell outputs. This paper demonstrates an in-memory computing architecture that achieves high SNR for mixed-signal compute, by exploiting charge-domain computation using bit-cell-integrated MOM capacitors. A 64-bank, 2.4-Mb in-memory-computing accelerator is achieved, mapping binarized convolutional neural-networks of practical size with performance equivalent to ideal software implementation, yet with energy efficiency of 866 binary TOPS/W and throughput of 19TOPS.

REFERENCES

[1] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6869–6898, 2017.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[3] V. Sze, Y.-H. Chen, J. Emer, A. Suleiman, and Z. Zhang, "Hardware for machine learning: Challenges and opportunities," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Apr./May 2018, pp. 1–8.

[4] V. Sze, "Designing hardware for machine learning: The important role played by circuit designers," *IEEE Solid State Circuits Mag.*, vol. 9, no. 4, pp. 46–54, Nov. 2017.

[5] Y. H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[6] D. Kim, J. Ahn, and S. Yoo, "ZeNA: Zero-aware neural network accelerator," *IEEE Design Test*, vol. 35, no. 1, pp. 39–46, Feb. 2018.

[7] Y.-H. Chen, J. Emer, and V. Sze, "Using dataflow to optimize energy efficiency of deep neural network accelerators," *IEEE Micro*, vol. 37, no. 3, pp. 12–21, Jun. 2017.

[8] M. Peemen, A. A. A. Setio, B. Mesman, and H. Corporaal, "Memory-centric accelerator design for convolutional neural networks," in *Proc. ICCD*, Oct. 2013, pp. 13–19.

[9] Z. Du *et al.*, "Shidiannao: Shifting vision processing closer to the sensor," *ACM SIGARCH Comput. Archit. News*, vol. 43, no. 3, pp. 92–104, 2015.

[10] T.-J. Yang, Y.-H. Chen, J. Emer, and V. Sze, "A method to estimate the energy consumption of deep neural networks," in *Proc. 51st Asilomar Conf. Signals, Syst., Comput.*, Oct. 2017, pp. 1916–1920.

[11] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6071–6079.

[12] K. Ando *et al.*, "BRein Memory: A single-chip binary/ternary reconfigurable in-memory deep neural network accelerator achieving 1.4 TOPS at 0.6 W," *IEEE J. Solid-State Circuits*, vol. 53, no. 4, pp. 983–994, Apr. 2018.

[13] K. Bong, S. Choi, C. Kim, S. Kang, Y. Kim, and H.-J. Yoo, "14.6 A 0.62 mW ultra-low-power convolutional-neural-network face-recognition processor and a CIS integrated with always-on Haar-like face detector," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2017, pp. 248–249.

[14] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: A 50.6 TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2018, pp. 218–220.

[15] A. Biswas and A. P. Chandrakasan, "Conv-RAM: An energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2018, pp. 488–490.

[16] W.-S. Khwa *et al.*, "A 65 nm 4 Kb algorithm-dependent computing-in-memory SRAM unit-macro with 2.3 ns and 55.8 TOPS/W fully parallel product-sum operation for binary DNN edge processors," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2018, pp. 496–498.

[17] S. K. Gonugondla, M. Kang, and N. Shanbhag, "A 42 pJ/decision 3.12 TOPS/W robust in-memory machine learning classifier with on-chip training," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2018, pp. 490–492.

[18] D. Bankman, L. Yang, B. Moons, M. Verhelst, and B. Murmann, "An always-on 3.8 μJ/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28-nm CMOS," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2018, pp. 222–224.

[19] M. Horowitz, "Computing's energy problem (and what we can do about it)," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2014, pp. 10–14.

[20] Y.-H. Chen, J. Emer, and V. Sze. (2018). "Eyeriss v2: A flexible and high-performance accelerator for emerging deep neural networks." [Online]. Available: https://arxiv.org/abs/1807.07928

[21] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, no. 7550, pp. 61–64, 2015.

[22] J. Zhang, Z. Wang, and N. Verma, "A machine-learning classifier implemented in a standard 6T SRAM array," in *Proc. IEEE Symp. VLSI Circuits (VLSI-Circuits)*, Jun. 2016, pp. 1–2.

[23] Q. Dong *et al.*, "A 0.3 V VDDmin 4+2T SRAM for searching and in-memory computing using 55 nm DDC technology," in *Proc. Symp. VLSI Circuits*, Jun. 2017, pp. C160–C161.

[24] C. Chen, K. Li, A. Ouyang, Z. Zeng, and K. Li, "Gflink: An in-memory computing architecture on heterogeneous CPU-GPU clusters for big data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 6, pp. 1275–1288, Jun. 2018.

[25] M. Le Gallo *et al.*, "Mixed-precision in-memory computing," *Nature Electron.*, vol. 1, no. 4, pp. 246–253, 2018.

[26] W.-H. Chen *et al.*, "A 65 nm 1 Mb nonvolatile computing-in-memory ReRAM macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2018, pp. 494–496.

[27] M. Kang, S. K. Gonugondla, A. Patil, and N. R. Shanbhag, "A multi-functional in-memory inference processor using a standard 6T SRAM array," *IEEE J. Solid-State Circuits*, vol. 53, no. 2, pp. 642–655, Feb. 2018.

[28] V. Seshadri *et al.*, "Ambit: In-memory accelerator for bulk bitwise operations using commodity DRAM technology," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2017, pp. 273–287.

[29] Y. Zhang, L. Xu, Q. Dong, J. Wang, D. Blaauw, and D. Sylvester, "Recryptor: A reconfigurable cryptographic cortex-M0 processor with in-memory and near-memory computing for IoT security," *IEEE J. Solid-State Circuits*, vol. 53, no. 4, pp. 995–1005, Apr. 2018.

[30] Y. Zhang *et al.*, "Recryptor: A reconfigurable in-memory cryptographic Cortex-M0 processor for IoT," in *Proc. Symp. VLSI Circuits*, Jun. 2017, pp. C264–C265.

[31] F. Su *et al.*, "A 462 GOPs/J RRAM-based nonvolatile intelligent processor for energy harvesting IoE system featuring nonvolatile logics and processing-in-memory," in *Proc. Symp. VLSI Circuits*, Jun. 2017, pp. T260–T261.

[32] T. F. Wu *et al.*, "Brain-inspired computing exploiting carbon nanotube FETs and resistive RAM: Hyperdimensional computing case study," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2018, pp. 492–494.

[33] H. Yonekawa *et al.*, "In-memory area-efficient signal streaming processor design for binary neural networks," in *Proc. IEEE 60th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2017, pp. 116–119.

[34] L. Ni, H. Huang, Z. Liu, R. V. Joshi, and H. Yu, "Distributed in-memory computing on binary RRAM crossbar," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 3, 2017, Art. no. 36.

[35] Z.-R. Wang *et al.*, "Efficient implementation of boolean and full-adder functions with 1T1R RRAMs for beyond von Neumann in-memory computing," *IEEE Trans. Electron Devices*, vol. 65, no. 10, pp. 4659–4666, Oct. 2018.

[36] F. Parveen, S. Angizi, Z. He, and D. Fan, "Low power in-memory computing based on dual-mode SOT-MRAM," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Jul. 2017, pp. 1–6.

[37] L. Ni, Z. Liu, H. Yu, and R. V. Joshi, "An energy-efficient digital ReRAM-crossbar based CNN with bitwise parallelism," *IEEE J. Explor. Solid-State Computat. Devices Circuits*, vol. 3, pp. 37–46, Dec. 2017.

[38] D. Bhattacharjee, F. Merchant, and A. Chattopadhyay, "Enabling in-memory computation of binary BLAS using ReRAM crossbar arrays," in *Proc. IFIP/IEEE Int. Conf. Very Large Scale Integr. (VLSI-SoC)*, Sep. 2016, pp. 1–6.

[39] L. Song, X. Qian, H. Li, and Y. Chen, "Pipelayer: A pipelined ReRAM-based accelerator for deep learning," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2017, pp. 541–552.

[40] A. Shafiee *et al.*, "ISAAC: A convolutional neural network accelerator with *in-situ* analog arithmetic in crossbars," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 14–26, 2016.
[41] S. Park *et al.*, "Neuromorphic speech systems using advanced ReRAM-based synapse," in *IEDM Tech. Dig.*, Dec. 2013, pp. 25.6.1–25.6.4.
[42] Z. Jiang, S. Yin, M. Seok, and J.-S. Seo, "XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks," in *Proc. IEEE Symp. VLSI Technol.*, Jun. 2018, pp. 173–174.
[43] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A mixed-signal binarized convolutional-neural-network accelerator integrating dense weight storage and multiplication for reduced data movement," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2018, pp. 141–142.
[44] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556
[45] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. AAAI*, vol. 4, 2017, p. 12.
[46] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks.," in *Proc. CVPR*, Jul. 2017, pp. 2261–2269.
[47] S. Zagoruyko and N. Komodakis. (2016). "Wide residual networks." [Online]. Available: https://arxiv.org/abs/1605.07146
[48] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2015, pp. 1–9.
[49] S. Ioffe and C. Szegedy. (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift." [Online]. Available: https://arxiv.org/abs/1502.03167
[50] R. E. Suarez, P. R. Gray, and D. A. Hodges, "All-MOS charge-redistribution analog-to-digital conversion techniques. II," *IEEE J. Solid-State Circuits*, vol. 10, no. 6, pp. 379–385, Dec. 1975.
[51] T. Kobayashi, K. Nogami, T. Shirotori, and Y. Fujimoto, "A current-controlled latch sense amplifier and a static power-saving input buffer for low-power architecture," *IEICE Trans. Electron.*, vol. E76-C, no. 5, pp. 863–867, 1993.
[52] 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.
[53] H. Omran, H. Alahmadi, and K. N. Salama, "Matching properties of femtofarad and sub-femtofarad MOM capacitors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 6, pp. 763–772, Jun. 2016.
[54] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "Envision: A 0.26-to-10tops/w subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm fdsoi," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2017, pp. 246–247.
[55] S. Bang *et al.*, "A 288μW programmable deep-learning processor with 270kB on-chip weight storage using non-uniform memory hierarchy for mobile intelligence," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2017, pp. 250–251.
[56] H. Jia, Y. Tang, H. Valavi, J. Zhang, and N. Verma. (2018). "A microprocessor implemented in 65nm CMOS with configurable and bit-scalable accelerator for programmable in-memory computing." [Online]. Available: https://arxiv.org/abs/1811.04047
[57] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. ACM/IEEE 44th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2017, pp. 1–12.
[58] S. Choi, J. Lee, K. Lee, and H.-J. Yoo, "A 9.02 mW CNN-stereo-based real-time 3D hand-gesture recognition processor for smart mobile devices," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2018, pp. 220–222.

Hossein Valavi (S'13) received the B.Sc. degree in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 2013, and the M.A. degree in electrical engineering from Princeton University, Princeton, NJ, USA, in 2015, where he is currently pursuing the Ph.D. degree.

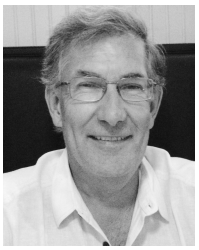
His research focuses on ultra-low-energy system design for signal processing and machine learning applications.

Mr. Valavi was a recipient of the Analog Devices Outstanding Student Designer Award in 2016.

Peter J. Ramadge (F'95) received the B.Sc., B.E., and the M.E. degrees from the University of Newcastle, Callaghan, NSW, Australia, and the Ph.D. degree in electrical engineering from the University of Toronto, Toronto, ON, Canada.

In 1984, he joined the Faculty of Princeton University, where he is currently a Gordon Y. S. Wu Professor of Engineering and a Professor of electrical engineering. His current research interests are in statistical signal processing, machine learning and various applications, including: data analysis, classification, prediction, medical and functional magnetic resonance imaging (fMRI) data analysis, and video and image processing.

Dr. Ramadge is a member of SIAM. He was a recipient of several honors and awards, including: a paper selected for inclusion in IEEE book Control Theory: Twenty Five Seminal Papers from 1932 to 1981; the Outstanding Paper Award from the Control Systems Society of the IEEE; a listing in ISIHighlyCited.com; the Convocation Medal for Professional Excellence from the University of Newcastle, Australia; an IBM Faculty Development Award; and the University Medal from the University of Newcastle, Australia.



Eric Nestler (M'75) received the B.S.E.E. degree from Tufts University, Medford, MA, USA, and the M.S.E.E degree from the University of Wisconsin-Madison, Madison, WI, USA.

He was with HP Medical, Palo Alto, CA, USA, and Symbolics, Inc., Chatsworth, CA, USA. In 1987, he joined ADI, Norwood, MA, USA, where he currently is an ADI Fellow working in the Analog Garage, Cambridge, MA, USA, and also the Founder of the Energy Metering Division that has shipped over 500 million energy metering parts during the past 20 years. He worked for two startups. He was with Lyric Semiconductor, ADI, in 2011, where he developed switched-capacitor analog signal processing circuit technology that provides very low power for complex signal processing. His present work is in ultralow power sampled analog technology (SAT) designs applied to areas such as sensor interfaces and ultrasound imaging. He holds more than 25 patents issued in a number circuit and product areas.

Naveen Verma (S'03–M'09) received the B.A.Sc. degree in electrical and computer engineering from The University of British Columbia, Vancouver, BC, Canada, in 2003, and the M.S. and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2005 and 2009, respectively.

Since 2009, he has been with the Department of Electrical Engineering, Princeton University, Princeton, NJ, USA, where he is currently an Associate Professor. His research focuses on advanced sensing systems, including low-voltage digital logic and SRAMs, low-noise analog instrumentation and data-conversion, large-area sensing systems based on flexible electronics, and low-energy algorithms for embedded inference, especially for medical applications.

Dr. Verma was a recipient or co-recipient of the 2006 DAC/ISSCC Student Design Contest Award, the 2008 ISSCC Jack Kilby Paper Award, the 2012 Alfred Rheinstein Junior Faculty Award, the 2013 NSF CAREER Award, the 2013 Intel Early Career Award, the 2013 Walter C. Johnson Prize for Teaching Excellence, the 2013 VLSI Symposium Best Student Paper Award, the 2014 AFOSR Young Investigator Award, the 2015 Princeton Engineering Council Excellence in Teaching Award, and the 2015 IEEE TRANSACTIONS ON COMPONENTS, PACKAGING AND MANUFACTURING TECHNOLOGY Best Paper Award. He is a Distinguished Lecturer of the IEEE Solid-State Circuits Society and serves on the technical program committees for the International Solid-State Circuits Conference (ISSCC), the VLSI Symposium, DATE, and the IEEE Signal-Processing Society (DISPS).