C3SRAM: An In-Memory-Computing SRAM Macro Based on Robust Capacitive Coupling Computing Mechanism

Zhewei Jiang, *Student Member, IEEE*, Shihui Yin, *Student Member, IEEE*, Jae-sun Seo, *Senior Member, IEEE*, and Mingoo Seok, *Senior Member, IEEE*

***Abstract*— This article presents C3SRAM, an in-memory-computing SRAM macro. The macro is an SRAM module with the circuits embedded in bitcells and peripherals to perform hardware acceleration for neural networks with binarized weights and activations. The macro utilizes analog-mixed-signal (AMS) capacitive-coupling computing to evaluate the main computations of binary neural networks, binary-multiply-and-accumulate operations. Without the need to access the stored weights by individual row, the macro asserts all its rows simultaneously and forms an analog voltage at the read bitline node through capacitive voltage division. With one analog-to-digital converter (ADC) per column, the macro realizes fully parallel vector–matrix multiplication in a single cycle. The network type that the macro supports and the computing mechanism it utilizes are determined by the robustness and error tolerance necessary in AMS computing. The C3SRAM macro is prototyped in a 65-nm CMOS. It demonstrates an energy efficiency of 672 TOPS/W and a speed of 1638 GOPS (20.2 TOPS/mm^2), achieving 3975× better energy–delay product than the conventional digital baseline performing the same operation. The macro achieves 98.3% accuracy for MNIST and 85.5% for CIFAR-10, which is among the best in-memory computing works in terms of energy efficiency and inference accuracy tradeoff.**

***Index Terms*— Analog-mixed-signal (AMS) computing, capacitive coupling, in-memory computing (IMC), machine learning accelerator, neural network, SRAM.**

I. Introduction

As DEEP convolutional neural networks (DCNNs) continue to demonstrate improvements in inference accuracies [1]–[5], deep learning is shifting toward edge computing. This development has motivated works on low-resource machine learning algorithms [6]–[13] and their accelerating hardware [14]–[18]. The most common operations in DCNNs are multiply-and-accumulate (MAC), which dominates power and delay. MAC operations have high regularity and parallelism and, therefore, are very suitable for hardware acceleration. However, the amount of memory access severely limits the energy efficiency in conventional digital accelerators [15]–[19]. As a result, in-memory computing (IMC) has become increasingly appealing to DCNN acceleration.

IMC is the design approach that performs highly- parallel computation inside the memory block without explicit row-by-row memory access. Recent IMC works [19]–[33] show significant energy efficiency and throughput advantages over conventional architectures. These benefits come at the cost of accuracy degradation from analog-mixed-signal (AMS) computing non-idealities. Hence, robust computing mechanisms and error-tolerant algorithms are the IMC design's main considerations and challenges [34].

This article presents an IMC SRAM macro based on capacitive-coupling computing (C3), hence named C3SRAM. The prototyped macro is 256 × 64 in size and computes 64 256-input binary-MAC operation (bMAC) in parallel. The macros can be used in a modular fashion to support the networks of arbitrary size. The 65-nm prototype chip demonstrates 671.5-TOPS/W energy efficiency and 1-638 GOPS throughput, which is 3975× improvement in energy–delay product (EDP) than digital baseline. It achieves 98.3% accuracy for the MNIST data set and 85.5% for the CIFAR-10 data set. This article is an extended version of [35], providing further discussion on design exploration, analysis of the IMC computing mechanism and sources of variability, and additional simulations and measurements.

II. IMC Overview and Related Works

IMC refers to memory architectures that support computations that take place inside the memory fabric, thus avoiding the energy-intensive memory wall [36]. IMC works are not exclusively MAC accelerators. For instance, designs in [19] and [28] and neural cache [29] support two-row logic operations. Some IMC works support machine learning algorithms other than neural networks, such as Ada-boost [21] and Random Forest [26]. In this section, we provide an overview of IMC designs for neural network acceleration.

A. Multi-Bit Weights in IMC Designs

Conventionally, in a neural network, both its activations and weights are multi-bit values. Since the physical topology of

Manuscript received December 5, 2019; revised March 3, 2020; accepted April 29, 2020. Date of publication May 18, 2020; date of current version June 29, 2020. This article was approved by Guest Editor Sylvain Clerc. This work was supported in part by the Wei Family Private Foundation, in part by the Catalyst Foundation, and in part by NSF under Grant 1919233 and Grant 1652866. *(Corresponding author: Mingoo Seok.)*

Zhewei Jiang and Mingoo Seok are with the Department of Electrical Engineering, Columbia University, New York, NY 10027 USA (e-mail: ms4415@columbia.edu).

Shihui Yin and Jae-sun Seo are with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287 USA.

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

Digital Object Identifier 10.1109/JSSC.2020.2992886

memory bitcells is independent at the array level, multi-bit weight representation in IMC is implemented at the circuit level instead of at architecture level. The following IMC works are designed to support multi-bit weight neural networks [31], [37]–[39]. For example, the Twin-8T design in [31] stores multi-bit weights in multiple SRAM cells, where these bitcells are numerically related to each other through the transistor width ratio in the adjacent columns.

Alternatively, there are IMC designs based on emerging non-volatile memory technologies that can store multi-bit weight in a single bitcell, such as phase-change memory (PCM) [37], [38] and resistive RAM (RRAM) [39]. However, these devices exhibit variability and nonlinearity limitations [42], [43], and the technologies are not yet mature enough.

B. Binary Weights in IMC Design

One of the chief algorithmic advancements that address the difficulty in multi-bit weight IMC design is the binary weight network (BWN) [8], in which the network weights are binarized, but the input and output activations can remain multi-bits. Weight binarization relaxes the storage constraint and makes storing weights straightforward.

A subset of BWNs called binary neural networks (BNN) binarizes both weight and activation to $+1$ and -1 such that multiplications can be represented by simple XNOR operations [9]–[11]. While early BNNs are not very accurate, recent advances [12] show that BNNs can approximate high MAC precision via weight duplication akin to stochastic computing, achieving comparable accuracy with multi-bit DCNN at lower hardware cost. As a result, many IMC works [21], [22], [25], [26], [30], [32], [33] are designed to support BWNs or BNNs.

C. Multi-Bit Activations in IMC Designs

Computing with multi-bit activations can be done in digital or analog domains. Digital representation can be done through multi-cycle operations, such as in a prior work Vesti [33]. The core IMC macro XNOR-SRAM [30] used in Vesti supports only binary/ternary activation. However, with digital partial sum output available from XNOR-SRAM, the Vesti architecture can accumulate the partial outputs in digital peripheral to arrive at the desired multi-bit activation MAC. This is, of course, at the cost of longer latency and higher energy dissipation.

For the analog representation of input activations, IMC designs have to perform the digital-to-analog conversion (DAC) before actual computing can take place. Several IMC works have employed DAC techniques to preprocess the input activations, expressing the analog value in voltage or current.

Using the DAC circuits with voltage output, the Twin-8T IMC design [31] can represent up to four levels of input activations with four distinct voltage levels on the wordlines. XNOR-SRAM [33] can similarly support up to three voltage levels to perform ternary MAC computation. Voltage-level-based DACs generally have limited resolution.

Current-based DAC circuits have been employed in designs, such as [21] and Conv-SRAM [22]. Using pulsewidth modulation (PWM), the input activation is first converted into a pulse. Within the window of the generated pulse, a charging current is then applied to a capacitive element. It has two main design challenges: 1) the PWM needs to be linear and 2) the current source needs to be constant. Both of these challenges would require area/energy tradeoff.

D. Compute in Current/Charge Domain

Analog MAC can be broadly placed in two categories: 1) current domain computing based on resistive voltage divider, discharging rate, and so on and 2) charge domain computing based on charge sharing, capacitive voltage divider, and so on.

XNOR-SRAM [30] bitcells would each turn on pull-up/pull-down transistors according to activation input and stored weights. These paths are applied to the same MAC bitline (MBL), creating a resistive voltage divider. The non-linearity of transistor resistance creates high gain in the desired transfer function region. However, this comes at the cost of high crowbar current and device variability. Zhang *et al.* [21] implemented its MAC operation by (dis)charging the wordlines, such as PWM-based DAC. However, the current source is a simple transistor and thus non-linear. To compensate, the design uses additional transistors to calibrate current in various conditions.

Valavi *et al.* [26] used charge sharing to perform bMAC. Each bitcell has an individual capacitor that is charged/discharged based on a bit-multiplication result. The capacitors are then tied together to share the charges, performing accumulation. Conv-SRAM [22] also uses charge sharing to perform MAC, where the charges (one row at a time) are placed on the bitline and shared row-wise. However, the multi-bit activation is derived from PWM-based DAC, and thus, the charges being shared are already the result of current-domain computing.

The proposed C3SRAM macro uses capacitive coupling to compute bMAC in the charge domain. The detailed description of architecture and operational procedures is presented in Section III.

III. Architecture and Operation

A. Memory Array Operation

We present the C3SRAM architecture and the operations in detail. Fig. 1 shows the architecture of the proposed macro. C3SRAM performs a fully parallel vector–matrix multiplication of 256 binary inputs and 256 $\times$ 64 binary weights. The macro consists of a 256 $\times$ 64 memory cell array, SRAM peripherals for read/write operations, input activation decoder/driver, and per-column flash analog-to-digital converters (ADCs).

Fig. 2 shows the 8T1C bitcell layout of the proposed design, a circuit diagram showing two bitcells in a column, and the table of XNOR operands. The bitcell is 80% larger than the conventional 6T bitcell in the same logic design rule, due to the two additional pass transistors and one capacitor constituting 27% of the bitcell area. The capacitor is implemented as MOSCAP for high capacitive density. A MOMCAP covering the area of a C3SRAM bitcell (MOMCAP can be placed

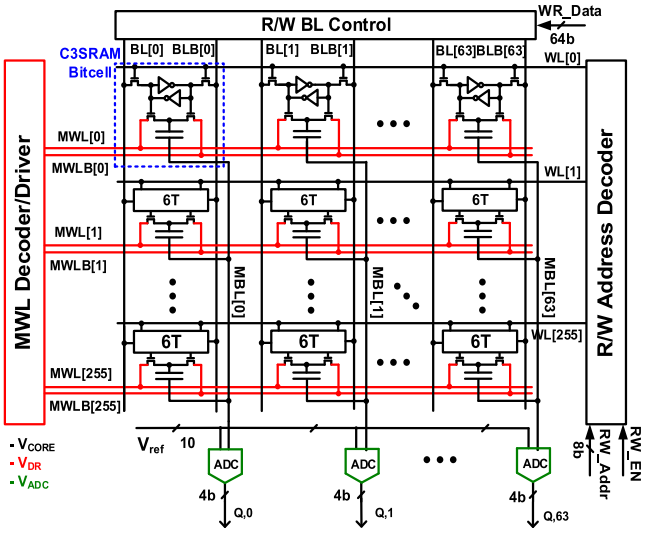


Fig. 1. Architecture of C3SRAM IMC macro.

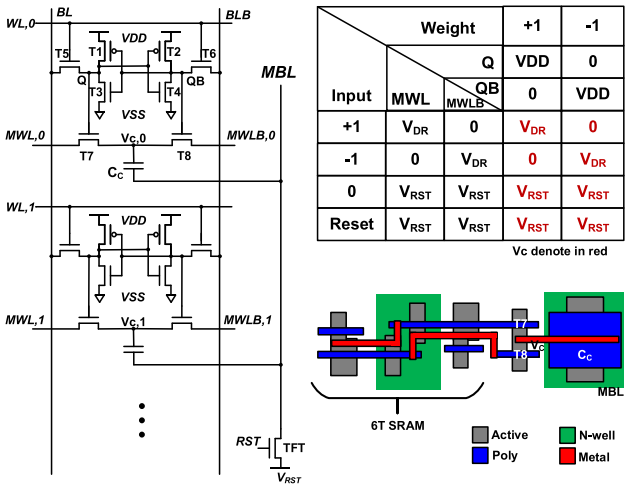


Fig. 2. C3SRAM bitcell design and in-cell bMAC operand table.

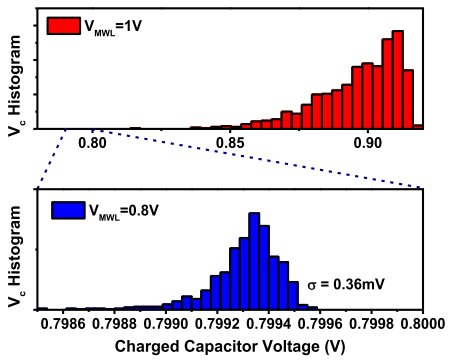


Fig. 3. Threshold voltage variability effects on charged capacitor voltage.

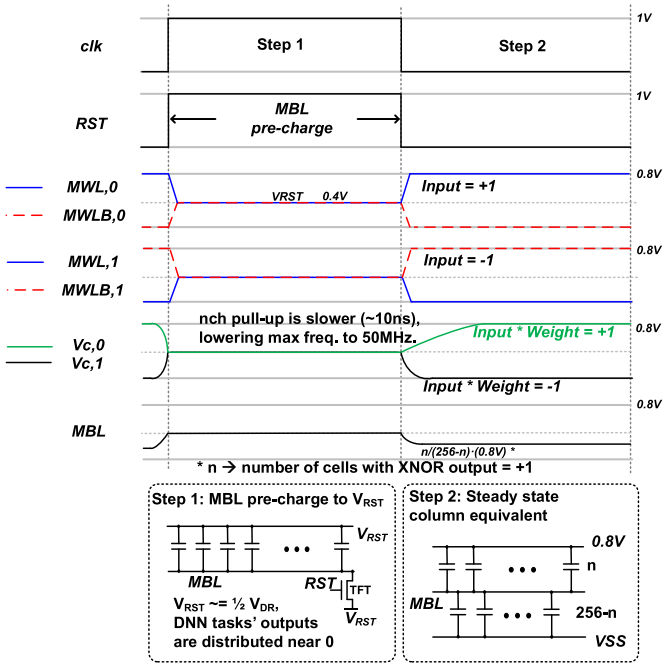


Fig. 4. Capacitive coupling based in-memory computation of bMAC.

on top of transistors with no area overhead) has 80% lower capacitance than C_C (~4 fF).

To perform binary dot product, the cap is charged/discharged by MAC wordlines (MWL/MWLB) via the pass transistors, which are gated by the stored weight and its complement. Since the pass transistors are NFETs, there would be a highly variable threshold voltage (V_t) drop across T7/T8 if the MWLs and the memory core have the same voltage source. To avoid the variability problem, we implement T7/T8 with LVT devices and set a separate source V_{DR} to drive the MWLs, at 200 mV lower than V_{CORE} (e.g., 0.8-V V_{DR} for 1-V V_{CORE}). We ran Monte Carlo SPICE simulations, including only C_C, T7, and T8 to isolate the variation of the pass transistors' effect on capacitor charge. As shown in Fig. 3 (histograms), the x-axis (note the different scales) shows the voltage level at the V_C node. Given the 200-mV margin, the variation of V_C has the small sigma of only 0.36 mV. T7/T8 decouple memory function and compute function to avoid potential read and write disturb [21], [25].

The bMAC operation of C3SRAM is shown in Fig. 4. There are two steps in this operation; each is completed in a half cycle duration. In step 1, each column's MBL is pre-charged via the footer TFT to $V_{RST} = 0.5 \cdot V_{DR}$. V_{RST} is set near the voltage corresponding to bMAC output of 0 (nominally 0.4 V). This is done to minimize the voltage swing on the MBL nodes since typical bMAC outputs in BNNs have a narrow distribution near 0 value. In the same step, MWL and MWLB of each row are likewise reset to V_{RST} such that there is no voltage potential on bitcell capacitors. At this step, the capacitors are effectively arranged in parallel where both nodes are reset to the same voltage, as shown in Fig. 4 (bottom left).

In step 2, the footer is turned off. The 256 input activations (denoted as In_i) are applied to 256 MWLs/MWLBs in parallel. For $\text{In}_i = +1(-1)$, MWL is driven from V_{RST} to V_{DR} (VSS), whereas MWLB is driven to VSS (V_{DR}). For $\text{In}_i = 0$, both MWL and MWLB remain at V_{RST} without consuming dynamic power. When the weight is $+1$ (-1), the voltage ramping via T7 (T8) induces a displacement current through capacitor C_C (~4 fF) in the bitcell, whose magnitude is

$$I_C = C_C \cdot dV_{\text{MWL}(B)}/dt. \quad (1)$$

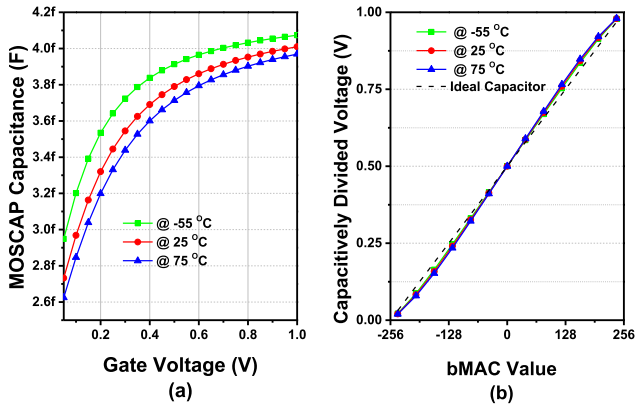


Fig. 5. (a) MOSCAP capacitance at TT corner simulation shows variation across temperature as well as the gate voltage. (b) MOSCAP capacitive voltage divider transfer function at various temperatures.

The charge transferred from the bitcell to MBL is

$$Q_{Ci} = \int_0^{t_1} I_C dt = \frac{1}{2} C_C V_{\mathrm{DR}} \tag{2}$$

where t_1 is the time it takes V_{MWL} to reach V_{DR}. The shared MBL voltage is set to

$$V_{\mathrm{MBL}} = C_C V_{\mathrm{DR}} \sum_1^{256} (\mathrm{XNOR}_i)/(256C_C + C_p) \tag{3}$$

where XNOR_i is the XNOR output of the ith bitcell and C_p is the parasitic capacitance of MBL plus the input capacitance of the ADC. At this step, each column can be seen as two sets of parallelly connected capacitors, which are in turn connected in series between V_{DR} and VSS, forming a capacitive voltage divider as shown in Fig. 4 (bottom right).

MOSCAP's high capacitive density provides bMAC transfer curve a wider full-scale range (FSR) than using the less capacitively dense MOMCAP [26]. Given the same level of MBL parasitics, the FSR loss from using MOMCAP would be ~80% higher than using MOSCAP. MOSCAP capacitance is dependent on temperature and gate voltage. Fig. 5(a) shows the C_C change over temperature and gate voltage. Fig. 5(b) shows the simulated transfer function of a capacitive voltage divider composed of C_C. The good news is that the temperature-related non-ideality of MOSCAP has small impact on the transfer function stability. The voltage-related non-linearity gives the transfer function a slight sigmoidal shape, which in fact could give some benefit to ADC (negligibly small in our design) because the slightly steeper slope in the region of interest provides slightly higher margins for reference voltages.

B. ADC Operation

The bMAC output of each column is a pre-activation partial sum. To digitize these values, C3SRAM includes an 11-level flash ADC per column. Each ADC consists of ten double-sampling-based self-calibrating single-ended comparators [see Fig. 6(top)]. Each comparator consists of an offset-canceling capacitor followed by an inverter chain, where the first inverter acts as an amplifier.

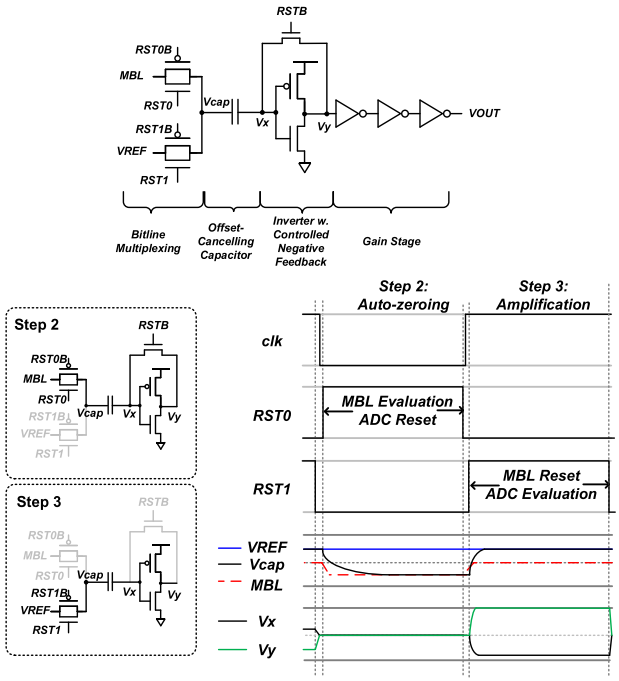


Fig. 6. Operation of the double-sampling self-calibrating single-ended comparator.

The ADC operation has two steps (see Fig. 6): during bMAC computation (step 2), MBL connects to the comparator input capacitor. The input and output of the first inverter are closed, placing the inverter in the high-voltage gain region. In step 3, the input node of the capacitor switches to the reference voltage, and the negative feedback path is turned off. The voltage differential between V_{MBL} and V_{ref} then causes (dis)charging on the capacitor. The inverter previously balanced at the trip point is driven high or low according to the direction of the induced current. The gain-stage inverter chain completes the amplification to digital domain.

C. Signal Switching Order

In the aforementioned three-step procedure, relevant signal transitions in steps 1 and 3 are decoupled in separate modules, meaning that while the digital output is being evaluated by the ADC, the memory array can begin computing the next batch of bMACs. This allows a half-cycle pipeline where step 1 (Fig. 4) and step 3 (Fig. 6) operate concurrently.

The bMAC operation is timing sensitive. To minimize analog non-idealities, concurrent signal switches described in Sections III-A and III-B must follow a strict order, as shown in Fig. 7. We implemented timing control circuitry with minimal delay elements to guarantee the correctness of the signals' order. The relevant transitions from steps 1 and 3 to step 2 follow this order.

1) The reference voltage must be disconnected from the comparator input capacitor before MBL leaves reset, and otherwise, the reference voltage source would inject charge onto the MBL floating node.
2) The negative feedback on the inverter stage must turn on before MBL is connected to the input capacitor, and

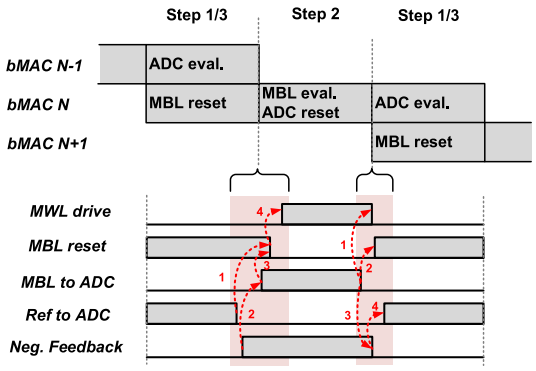


Fig. 7. Signal transition order for reducing analog non-idealities.

otherwise, V_{MBL} will be affected by the induced current of inverter gates driven to the trip point.

3) MBL must be connected to the comparator input capacitor before MBL leaves reset, and otherwise, the charge differential of reference voltage stored on the capacitor will be injected into the floating MBL.
4) MWL cannot be driven until MBL is floating, and otherwise, some coupling current would be discharged.

The relevant transitions from step 2 to steps 1 and 3 follow this order.

1) MBL must disconnect from comparator input before MWL drivers switch to reset voltage, and otherwise, the input changes will induce current on the MBL.
2) Also, MBL needs to disconnect before MBL reset footer turns on, and otherwise, V_{MBL} stored on the input capacitor would begin to reset as well.
3) Also, MBL needs to disconnect before the negative feedback is turned off since the act of disconnection can disturb the inverter input which is at the sensitive trip point.
4) The negative feedback needs to switch OFF before reference voltage is connected to the comparator input, and otherwise, the charge differential would be (dis)charged via the feedback path.

IV. Algorithm-Hardware Specification

In this section, we determine the design specification pertaining to algorithmic support: activation precision and pre-activation quantization levels.

A. Activation Bit Precision

Operating using the same IMC hardware, inference accuracy loss of a BNN is found less than that of a BWN with multi-bit activation [33]. One reason is that the analog representation of multi-bit activation requires additional domain conversion through DAC [31]. Another reason is that network models trained at higher precision are more sensitive to AMS error. As the study in [44] suggests, robustness is a benefit of weight redundancy. In similarly accurate models, error in multi-bit MAC is more severe than bMAC due to BNNs' weight duplication scheme.

We examine this issue by applying various levels of stochastic error during the inference of the MNIST data set.

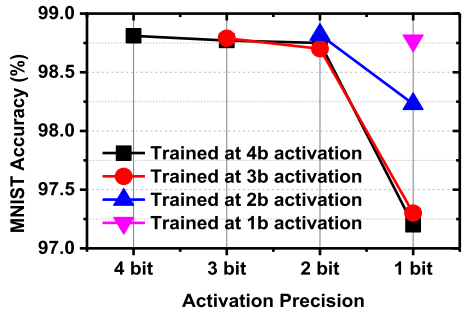


Fig. 8. MLP on the MNIST data set inference accuracy losses at various levels of activation precisions.

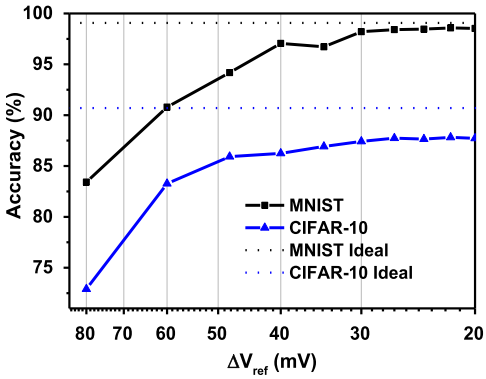


Fig. 9. MNIST and CIFAR-10 inference accuracies increase as quantization resolution of pre-activation partial sum (256 input) increases.

The network topology used in this examination is a binary-weight multi-layer perceptron (MLP) consisting of three fully connected (FC) hidden layers, each with 512 neurons. The network is trained at various activation precisions from 1 to 4 bits. The trained MLPs are then mapped on the C3SRAM for testing inference accuracy, each time with decreasing the pre-activation resolution to simulate the increasing level of stochastic noise. Here, we use the digital representation for activations as in [33], i.e., activations are fed in a bit-serial fashion and outputs are accumulated using digital adders. As shown in Fig. 8, at the same level of stochastic error, networks trained at higher precision degrade more. We conclude that for IMC hardware running multi-bit activation BWN to achieve comparable accuracy as BNN, the hardware may need more resources to compensate for AMS errors and this could result in inefficiencies. For this reason, the proposed C3SRAM primarily supports BNN acceleration.

B. Partial Convolution Quantization Levels

In practical neural networks, a typical convolution filter is too large to fit in a column of memory cells and, therefore, would be split into several C3SRAM arrays. In such cases, each array produces partial convolution results that are then accumulated in digital peripheral to produce the final output.

For the 256-row C3SRAM, the full resolution of partial convolution results is 8 bit. For an ADC to achieve this high resolution, it would incur considerable area, power, and latency overhead. The objective here is to use a lower resolution ADC design that is still able to maintain the final accuracy.

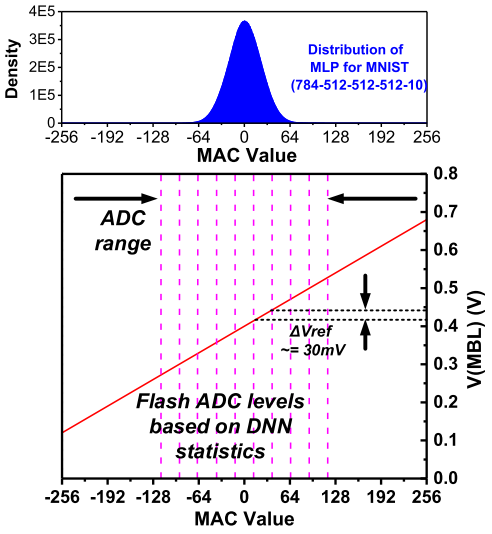


Fig. 10. bMAC distribution of MLP for MNIST and quantization in limited ADC range.

To find the appropriate ADC resolution that can maintain inference accuracy, we examine the effect of quantization on the MNIST and CIFAR-10 data sets. We use the same MLP described in Section IV-A for the MNIST data set. For CIFAR-10, we use a VGG-like convolutional BNN [9], with six convolutional layers and three FC layers. For partial convolution results in these network models, we apply several quantization levels, successively reducing ΔV_{ref}. Fig. 9 shows the effect of quantization on the task accuracy. We find that in both cases, the accuracies reach saturation at ~30-mV ΔV_{ref} that corresponds to 5-bit resolution given the 640-mV FSR. This is consistent with results in [22] where 5-bit ADC is used to achieve high accuracy on the MNIST data set using LeNet-5.

To further reduce the cost of the ADC, the pre-activation distribution can be exploited to reduce unnecessary hardware. Partial convolutions are not uniformly distributed for the entire range of the activation function input. Rather, they are usually narrowly distributed around 0 that is the point of nonlinearity in common BWN/BNN activation functions' ReLU/binary step. Fig. 10 (top) shows the MLP bMAC distribution. Since the data are distributed in a small region of the FSR, the ADC range can be confined to this region without accuracy loss. Fig. 10 (bottom) shows an illustration of an ideal transfer function and linear quantization levels in a confined range. The x-axis is the bMAC output value; the y-axis is V_{MBL} corresponding to the bMAC output. In the voltage region corresponding to bMAC value from -120 to $+120$, only 11 reference levels ($\Delta V_{\text{ref}} = 30$ mV) are needed to match ~5-bit ADC resolution.

V. Measurements and Analyses

The C3SRAM macro is implemented in a 65-nm CMOS. Fig. 11 shows the micrograph of the test chip. The macro has a capacity of 16 kb at the footprint of 0.081 mm^2.

A. Energy and Throughput

The memory macro has 2-kB capacity. For bMAC computations, the macro operates at a maximum frequency of 50 MHz,

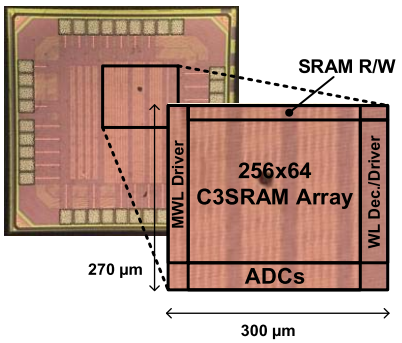


Fig. 11. Prototype chip micrograph.

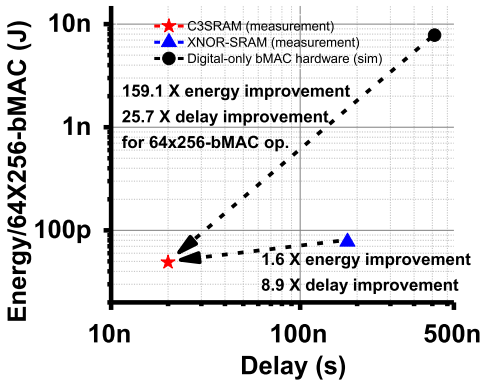


Fig. 12. C3SRAM energy and delay comparison with XNOR-SRAM [30] and digital ASIC with traditional SRAM and ALU.

limited by minimally sized footer discharging ADC input capacitors. The macro computes 64 independent 256-input bMACs per cycle. The throughput is 2•256•64/20 ns = 1638 GOPS. The compute density is thus 20.2 TOPS/mm^2. At the operating voltages of 1-V core supply (V_{CORE}), 0.8-V driver (V_{DR}), and 0.6-V ADC (V_{ADC}), it consumes 49 pJ excluding input–output data movement, reaching an energy efficiency of 671.5 TOPS/W, a ~3975× improvement in EDP over the digital baseline formulated in [33], and ~14× over XNOR-SRAM (see Fig. 12). Fig. 13(a) shows the power breakdown measurements: 38.7% of the total power is consumed by driving the MWL and bitcell capacitors, 22.0% by ADCs, and 39.3% by all other digital peripherals, including MWL decoder and partial sum accumulation.

To fully benefit from the speed and power improvement of C3SRAM, striding and other dedicated input movement circuits such as those in the implementation of Vesti [33] are also needed to prevent the memory macros from stalling to wait for the next input arrival. Otherwise, it would impose significant overhead in activation data movement. Hence, C3SRAM is better utilized in a stand-alone module than a direct replacement of SRAM in conventional von Neumann architecture.

Table I shows the comparison of recent IMC works for neural network acceleration. C3SRAM achieves high energy efficiency and throughput. Note that some designs support higher activation precision.

B. Transfer Function

In this section, we measure and analyze the transfer function characteristics of C3SRAM under the same operating

TABLE I
COMPARISON TO PRIOR IMC WORKS

	Biswas [22]	Si [25]	Valavi [26]	Jiang [30]	Si [31]	Gonugondla [45]	**This work**
Technology	65nm	65nm	65nm	65nm	55nm	65nm	**65nm**
Cell Type	10T	Split-6T	10T1C	12T	Twin-8T	6T	**8T1C**
Operating Voltage	1.2V (DAC)/ 0.8V (Array)/ 1V (rest)	1V	1V	0.6-1V	1V	1V	**1V (Array)/ 0.8V (Driver)/ 0.6V (ADC)**
Memory Capacity	2 kB	512 B	32 kB	2 kB	480 B	16 kB	**2 kB**
Input Precision	6	1	1	1	1-4	8	**1**
Weight Precision	1	1	1	1	2-5	8	**1**
Output Precision	6	1	1	5	3-7	4	**5** [1]
Efficiency (TOPS/W) [2]	40.3	30.49-55.8	658	403	18.37-72.03	6.25	**671.5**
Throughput (GOPS) [3]	8	1,112.8	589.9	665	84.8-269.6	8.26	**1,638**

[1] Margin equivalent to 11-level mid-range of 5-bit resolution.
[2] One MAC is counted as two operations (multiplication and addition).
[3] Consider an array size of 256x64 as unit capacity.

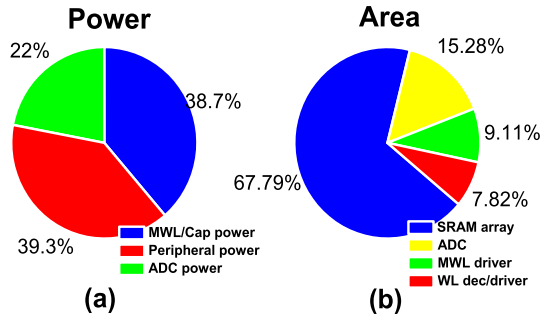


Fig. 13. (a) Measured power consumption breakdown between the three supplies powering bMAC compute (blue), partial sum accumulation (red), and ADC (green). (b) Area breakdown of the C3SRAM module.

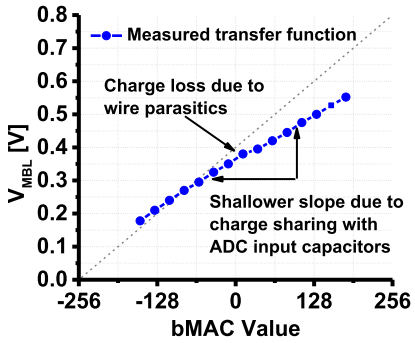


Fig. 14. Measured VMBL transfer curve shows a lower FSR from ideal curve due to charge sharing with ADC input capacitors.

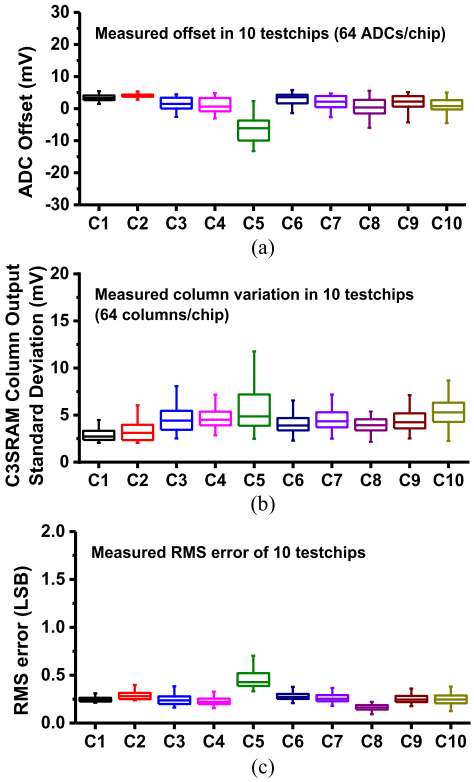


Fig. 15. (a) ADC output offset due to comparator gain-stage mismatch. (b) VMBL error variation measurement. (c) RMS error of the macro.

condition as in Section V-A. The transfer function is measured according to the following steps. First, we set all weights to zero and then apply known input pattern corresponding to a target bMAC output, resulting in a determinate voltage output on the MBL which we then measure. We set the off-chip flash ADC reference voltages such that we observe the result of a single comparator, i.e., set all but the first reference voltages beyond the V_{DR} range. Then, sweep the reference voltage of the comparator to find the value at which the result flips. This trip point corresponds to the bMAC output. We repeat these steps at each bMAC value to construct the transfer function. As shown in Fig. 14, the transfer function measurement shows good linearity and stability, while the FSR is reduced due to ADC input capacitors and MBL wire parasitics.

C. Variability Measurement

In this section, we characterize C3SRAM variabilities. The box chart in Fig. 15(a) shows the various measurements

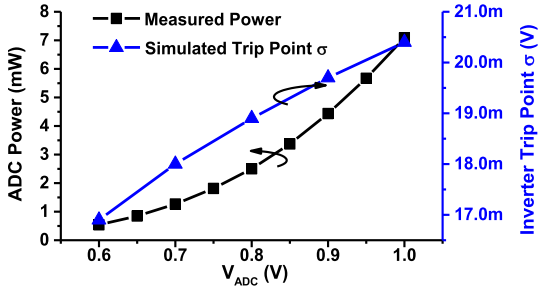


Fig. 16. ADC power increases exponentially as ADC power supply increases; the trip point variation increases linearly.

of the ADC comparator offset. The data include the offset statistics of 10 chips, each with 64 columns and 10 comparators per column. The variation is resultant from charge leakage, input/reference signal noise, and device mismatch. Monte Carlo simulations at TT corner show ~5-mV sigma in comparator variations, consistent with measurements. The charge leakage that most significantly affects comparator output is during the capacitor input switch. As described in Section III-C, if the level of leakage or noise is greater than the delta between V_{MBL} and reference during the input capacitor switch, the comparator output can be corrupted. After the input stage, the device mismatch dominates the variation. It causes trip point differential in the inverter chains. Fig. 16 shows that the lower the operating voltage is, the trip point variation becomes less prominent. Since the post-input offset voltage is dominated by the trip point delta between the first and second inverters divided by the gain of the first, we operate the ADC at a lower voltage of 0.6V and use the long-channel device for the first inverter to achieve high gain. This also helps with power reduction, as shown in Fig. 16.

Fig. 15(b) shows the measured variation of bMAC operations. The main sources of variation are C_C mismatch. The mismatch variation of a single C_C has σ_C/C_C of 4.2% according to the Monte Carlo simulation. We determine the deviation on the transfer function using the propagation of uncertainty rule

$$\sigma_{\mathrm{MBL}} = \frac{n}{256}\sqrt{\frac{n\cdot\sigma_C^2}{n^2\cdot C_C^2}+\frac{256\cdot\sigma_C^2}{256^2 C_C^2}} = \frac{n\sigma_C}{256\cdot C_C}\sqrt{\frac{1}{n}+\frac{1}{256}}. \quad (4)$$

The variation of V_{MBL} from the confined region of -120 to $+120$ has a sigma ranging from deviation is ~0.91 to ~1.77 mV, based on the ~600-mV FSR from Fig. 14, consistent with Fig. 15(b) (which also includes intra-chip ADC offset variation).

Fig. 15(c) shows the rms error of the macro-performing bMAC operations. As pre-activation varies greatly in distribution variance, there is no universal input set that can characterize C3SRAM for neural networks in general. A uniform input set is used for the measurement. This result includes all non-idealities previously described, and the additional errors from unary-to-binary conversion, which has no error-correction feature for low area overhead. Thus, simple bubble error can cause deviations at the final output larger than the signal variation level. This error can be mitigated with additional error correction circuitry in future works.

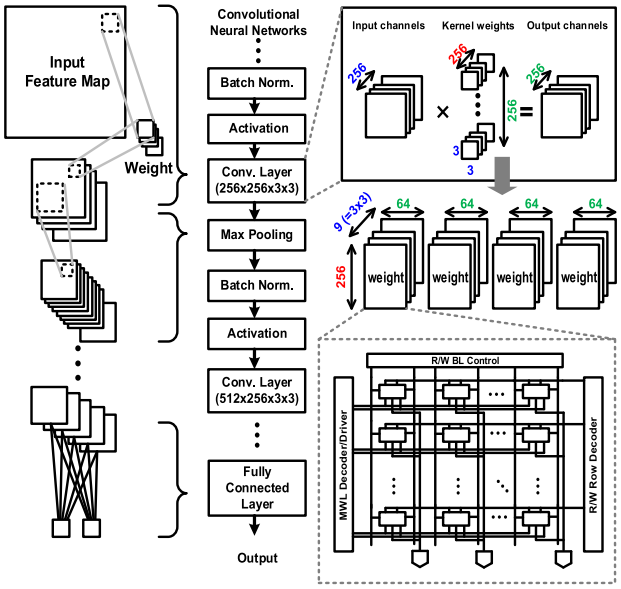


Fig. 17. Mapping convolutional neural networks to C3SRAM-based IMC.

D. Evaluation on Neural Network Tasks

In our evaluation for BNN accuracy, C3SRAM is responsible for the computations of convolution layers and FC layers, and all other operations of the BNN are performed in digital simulation. To evaluate the accuracy performance of C3SRAM for deep neural networks, C3SRAM computes all bMAC operations from the first hidden layer. A weight-stationary mapping scheme optimized for data reuse is adapted in our experiments. The mapping of FC layer weights in C3SRAM is as such; weights of a layer are organized column-wise and inputs/activations are applied at each row. On the other hand, convolutional layer mapping is an extension of the FC layer mapping. Mapping a $3 \times 3 \times 256$ filter from a convolution layer is the same as mapping nine 256-neuron FC layer weights. The channels are organized in column orientation, and each channel's kernel is distributed across multiple macros. The partial sums produced by ADCs are accumulated to generate the pre-activation for each neuron. The mapping of a representative 256-channel 3×3 kernel filters is shown in Fig. 17.

As detailed in Table II, we evaluated the inference accuracy of C3SRAM for MNIST and CIFAR-10 data sets. Max pooling and batch normalization are performed in the digital domain with the bit precisions of 12 and 10, respectively. The accuracy for MNIST is 98.3%, against the digital baseline result of 98.7%. This accuracy result was obtained from direct measurements of the entire network. For CIFAR-10, due to test chips' limited throughput, the accuracy is evaluated from simulations based on the measured error probability. We injected AMS and quantization errors in the inference of CIFAR-10 test images. At 20 runs with random seeds, the average accuracy is at 85.5%, whereas the digital baseline accuracy is 88.6%. This accuracy can be improved with a better trained model, as research works [12], [44] have found that BNN conversion

TABLE II
ACCURACY COMPARISON

	MNIST	CIFAR-10
Neural Network	MLP	VGG-like CNN
Network Topology[a]	784FC-512FC-512FC-512FC-10FC	128C3-128C3-MP2-256C3-256C3-MP2-512C3-512C3-MP2-1024FC-1024FC-10FC
Baseline Accuracy	98.7%	88.6%
Test Chip Accuracy	98.3%	85.5%

[a] nCk – kxk kernel convolutional layer with n filters, mFC – m-neuron FC layer, MPp – max-pooling layer with pxp pooling size.

with wider topology improves both robustness and accuracy. As [12] demonstrates the algorithmic advances of BNNs, the scalable mapping of C3SRAM could be used as computational primitive for larger BNNs for more complex machine learning tasks.

VI. CONCLUSION

The IMC concept is developed to meet the challenge of memory bottleneck in neural network inference in conventional hardware. It can achieve high parallelism and throughput via memory cell density and achieves low power from analog computing and substantial reduction in data access. IMC faces the design challenges of its own in the form of analog computing robustness issues, from process variation to system noise. Design decisions, such as error-tolerant algorithms and low-variation hardware, are important considerations. In this article, we present C3SRAM, an IMC macro for neural network acceleration. It supports noise-resistant BNNs, utilizes low variability components, and is scalable to map large neural networks in a modular fashion. Using a robust capacitive coupling mechanism, the architecture can reach comparable accuracy with the algorithmic baseline. The 16-kb prototype in 65 nm achieves the energy efficiency of 671.5 TOPS/W and the throughput of 1638 GOPS for bMAC operations.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–14.

[3] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[4] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 448–456.

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

[6] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," 2016, *arXiv:1609.07061*. [Online]. Available: http://arxiv.org/abs/1609.07061

[7] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 1737–1746.

[8] M. Courbariaux, Y. Bengio, and J. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2015, pp. 3123–3131.

[9] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 4107–4115.

[10] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 525–542.

[11] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," 2016, *arXiv:1606.06160*. [Online]. Available: http://arxiv.org/abs/1606.06160

[12] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 345–353.

[13] T. Guan, X. Zeng, and M. Seok, "Recursive binary neural network learning model with 2.28b/Weight storage requirement," 2017, *arXiv:1709.05306*. [Online]. Available: http://arxiv.org/abs/1709.05306

[14] P. A. Merolla *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014.

[15] 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.

[16] 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.

[17] D. Shin, J. Lee, J. Lee, and H.-J. Yoo, "DNPU: An 8.1TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2017, pp. 240–241.

[18] P. N. Whatmough, S. K. Lee, H. Lee, S. Rama, D. Brooks, and G.-Y. Wei, "A 28nm SoC with a 1.2GHz 568nJ/prediction sparse deep-neural-network engine with >0.1 timing error rate tolerance for IoT applications," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2017, pp. 242–243.

[19] Q. Dong *et al.*, "A 0.3 V VDDmin 4+2T SRAM for searching and in-memory computing using 55nm DDC technology," in *Proc. Symp. VLSI Circuits*, Jun. 2017, pp. 160–161.

[20] M. Kang, S. K. Gonugondla, and N. R. Shanbhag, "A 19.4 nJ/decision 364K decisions/s in-memory random forest classifier in 6T SRAM array," in *Proc. 43rd IEEE Eur. Solid State Circuits Conf. (ESSCIRC)*, Sep. 2017, pp. 263–266.

[21] J. Zhang, Z. Wang, and N. Verma, "In-memory computation of a machine-learning classifier in a standard 6T SRAM array," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 915–924, Apr. 2017.

[22] A. Biswas and A. P. Chandrakasan, "CONV-SRAM: An energy-efficient SRAM with in-memory dot-product computation for low-power convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 217–230, Jan. 2019.

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

[24] W.-H. Chen *et al.*, "A 65nm 1Mb 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.

[25] X. Si *et al.*, "A dual-split 6T SRAM-based Computing-in-Memory unit-macro with fully parallel product-sum operation for binarized DNN edge processors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 11, pp. 4172–4185, Nov. 2019.

[26] 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.

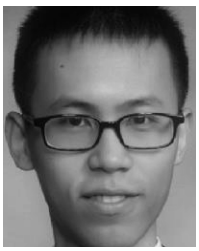
[27] M. Kang, S. Lim, S. Gonugondla, and N. R. Shanbhag, "An in-memory VLSI architecture for convolutional neural networks," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 3, pp. 494–505, Sep. 2018.

[28] J. Wang *et al.*, "A compute SRAM with bit-serial integer/floating-point operations for programmable in-memory vector acceleration," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2019, pp. 224–226.

[29] C. Eckert *et al.*, "Neural cache: Bit-serial in-cache acceleration of deep neural networks," *IEEE Micro*, vol. 39, no. 3, pp. 11–19, May 2019.

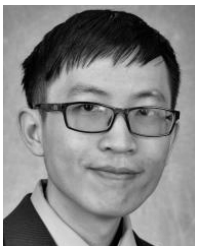
[30] Z. Jiang, S. Yin, M. Seok, and J. Seo, "XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks," in *Proc. IEEE Symp. VLSI Technol.*, Jun. 2018, pp. 173–174.

[31] X. Si *et al.*, "A twin-8T SRAM computation-in-memory macro for multiple-bit CNN-based machine learning," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2019, pp. 396–398.
[32] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-tile 2.4-mb In-Memory-Computing CNN accelerator employing charge-domain compute," *IEEE J. Solid-State Circuits*, vol. 54, no. 6, pp. 1789–1799, Jun. 2019.
[33] S. Yin, Z. Jiang, M. Kim, T. Gupta, M. Seok, and J.-S. Seo, "Vesti: Energy-efficient in-memory computing accelerator for deep neural networks," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 1, pp. 48–61, Jan. 2020.
[34] N. Verma *et al.*, "In-memory computing: Advances and prospects," *IEEE Solid StateCircuits Mag.*, vol. 11, no. 3, pp. 43–55, Sum. 2019.
[35] Z. Jiang, S. Yin, J.-S. Seo, and M. Seok, "C3SRAM: In-Memory-Computing SRAM macro based on capacitive-coupling computing," *IEEE Solid-State Circuits Lett.*, vol. 2, no. 9, pp. 131–134, Sep. 2019.
[36] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2014, pp. 10–14.
[37] G. W. Burr *et al.*, "Experimental demonstration and tolerancing of a large-scale neural network (165 000 Synapses) using phase-change memory as the synaptic weight element," *IEEE Trans. Electron Devices*, vol. 62, no. 11, pp. 3498–3507, Nov. 2015.
[38] S. Kim *et al.*, "NVM neuromorphic core with 64k-cell (256-by-256) phase change memory synaptic array with on-chip neuron circuits for continuous *in-situ* learning," in *IEDM Tech. Dig.*, Dec. 2015, p. 17.
[39] P. Chi *et al.*, "PRIME: A novel Processing-in-Memory architecture for neural network computation in ReRAM-based main memory," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 27–39.
[40] F. Parveen, Z. He, S. Angizi, and D. Fan, "HielM: Highly flexible in-memory computing using STT MRAM," in *Proc. 23rd Asia South Pacific Design Automat. Conf. (ASP-DAC)*, Jan. 2018, pp. 361–366.
[41] M. Zabihi, Z. I. Chowdhury, Z. Zhao, U. R. Karpuzcu, J.-P. Wang, and S. S. Sapatnekar, "In-memory processing on the spintronic CRAM: From hardware design to application mapping," *IEEE Trans. Comput.*, vol. 68, no. 8, pp. 1159–1173, Aug. 2019.
[42] A. Chen and M.-R. Lin, "Variability of resistive switching memories and its impact on crossbar array performance," in *Proc. Int. Rel. Phys. Symp.*, Apr. 2011, p. MY-7.
[43] T. Gokmen and Y. Vlasov, "Acceleration of deep neural network training with resistive cross-point devices: Design considerations," *Frontiers Neurosci.*, vol. 10, p. 333, Jul. 2016.
[44] G. Gambardella *et al.*, "Efficient error-tolerant quantized neural network accelerators," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Oct. 2019, pp. 1–6.
[45] S. K. Gonugondla, M. Kang, and N. R. Shanbhag, "A variation-tolerant in-memory machine learning classifier via on-chip training," *IEEE J. Solid-State Circuits*, vol. 53, no. 11, pp. 3163–3173, Nov. 2018.



Zhewei Jiang (Student Member, IEEE) received the dual B.S. degrees in physics and in electrical engineering from Adelphi University, Garden City, NY, USA, and Columbia University, New York, NY, USA, respectively, in 2013 and the M.S. degree in electrical engineering from Columbia University in 2015, where he is currently pursuing the Ph.D. degree in electrical engineering.

He has been a Research Assistant with VLSI Lab, Columbia University, since 2015. His research interests include neuromorphic computing architecture, neural signal processing, in-memory computation for machine learning, and other algorithm implementations.



Shihui Yin (Student Member, IEEE) received the B.S. degree in microelectronics from Peking University, Beijing, China, in 2013, and the M.S. degree in electrical engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2015. He is currently pursuing the Ph.D. degree in electrical engineering with Arizona State University, Tempe, AZ, USA.

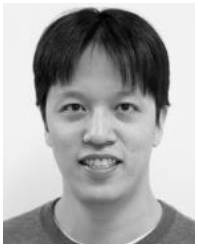
His research interests include low-power biomedical circuit and system design, and energy-efficient hardware design for machine learning and neuromorphic computing.

Mr. Yin was a recipient of the University Graduate Fellowship from Arizona State University in 2015 and the IEEE Phoenix Section Student Scholarship for the year 2016.

Jae-sun Seo (Senior Member, IEEE) received the B.S. degree in electrical engineering from Seoul National University, Seoul, South Korea, in 2001, and the M.S. and Ph.D. degrees in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2006 and 2010, respectively.

From 2010 to 2013, he was with IBM T. J. Watson Research Center, Yorktown Heights, NY, USA, where he worked on cognitive computing chips under DARPA SyNAPSE Project and energy-efficient integrated circuits for high-performance processors. In 2014, he joined the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, USA, as an Assistant Professor. In 2015, he was with Intel Circuits Research Lab as a Visiting Faculty Member. His current research interests include efficient hardware design of machine learning and neuromorphic algorithms and integrated power management.

Dr. Seo was a recipient of the Samsung Scholarship during 2004–2009, the IBM Outstanding Technical Achievement Award in 2012, and the NSF CAREER Award in 2017.



Mingoo Seok (Senior Member, IEEE) received the B.S. degree *(summa cum laude)* in electrical engineering from Seoul National University, Seoul, South Korea, in 2005, and the M.S. and Ph.D. degrees from the University of Michigan, Ann Arbor, MI, USA, in 2007 and 2011, respectively, all in electrical engineering.

He was a member of the Technical Staff with Texas Instruments Inc., Dallas, TX, USA, in 2011. Since 2012, he has been with Columbia University, New York, NY, USA, where he is currently an Associate Professor of electrical engineering. His current research interests include ultra-low-power system-on-chip design for emerging embedded systems, machine learning VLSI architecture and circuits, variation, voltage, aging, thermal-adaptive circuits and architecture, on-chip integrated power circuits, and nonconventional hardware design, including in-memory computing SRAM and DRAM.

Dr. Seok received the 1999 Distinguished Undergraduate Scholarship from Korea Foundation for Advanced Studies, the 2005 Doctoral Fellowship from the Korea Foundation for Advanced Studies, the 2008 Rackham Pre-Doctoral Fellowship from the University of Michigan, the 2009 AMD/CICC Scholarship Award for picowatt voltage reference work, the 2009 DAC/ISSCC Design Contest for the 35-pW sensor platform design, the 2015 NSF CAREER Award, and the 2019 Qualcomm Faculty Award. He is a Technical Program Committee Member for several conferences, including the IEEE International Solid-State Circuits Conference (ISSCC) and the IEEE Custom Integrated Circuits Conference (CICC). He served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS from 2013 to 2015. He has been serving as an Associate Editor for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS since 2015 and the IEEE SOLID-STATE CIRCUITS LETTERS since 2017. He also serves as a Guest Editor for the IEEE JOURNAL OF SOLID-STATE CIRCUITS.