Vesti: Energy-Efficient In-Memory Computing Accelerator for Deep Neural Networks

Shihui Yin, *Student Member, IEEE*, Zhewei Jiang, *Student Member, IEEE*, Minkyu Kim, *Student Member, IEEE*, Tushar Gupta, Mingoo Seok, *Senior Member, IEEE*, and Jae-Sun Seo, *Senior Member, IEEE*

***Abstract*— To enable essential deep learning computation on energy-constrained hardware platforms, including mobile, wearable, and Internet of Things (IoT) devices, a number of digital ASIC designs have presented customized dataflow and enhanced parallelism. However, in conventional digital designs, the biggest bottleneck for energy-efficient deep neural networks (DNNs) has reportedly been the data access and movement. To eliminate the storage access bottleneck, new SRAM macros that support in-memory computing have been recently demonstrated. Several in-SRAM computing works have used the mix of analog and digital circuits to perform XNOR-and-ACcumulate (XAC) operation without row-by-row memory access and can map a subset of DNNs with binary weights and binary activations. In the single array level, large improvement in energy efficiency (e.g., two orders of magnitude improvement) has been reported in computing XAC over digital-only hardware performing the same operation. In this article, by integrating many instances of such in-memory computing SRAM macros with an ensemble of peripheral digital circuits, we architect a new DNN accelerator, titled *Vesti*. This new accelerator is designed to support configurable multibit activations and large-scale DNNs seamlessly while substantially improving the chip-level energy-efficiency with favorable accuracy tradeoff compared to conventional digital ASIC. Vesti also employs double-buffering with two groups of in-memory computing SRAMs, effectively hiding the row-by-row write latencies of in-memory computing SRAMs. The Vesti accelerator is fully designed and laid out in 65-nm CMOS, demonstrating ultralow energy consumption of <20 nJ for MNIST classification and <40 μJ for CIFAR-10 classification at 1.0-V supply.**

***Index Terms*— Deep learning accelerator, deep neural networks (DNNs), double-buffering, in-memory computing, SRAM.**

I. Introduction

IN RECENT years, deep learning and deep neural networks (DNNs) [1]–[10] have unprecedentedly improved the accuracies in large-scale recognition tasks. However, to achieve incremental accuracy improvement, the state-of-the-art deep learning algorithms tend to present very deep and large network models (e.g., 1000-layer networks [5]), and this poses significant challenges for DNN hardware implementations in terms of computational complexity, memory access, and the associated energy cost.

A number of prior works from algorithms to hardware reduced the energy cost. On the algorithm side, pruning and compression have been extensively studied [11], substantially reducing the number of nonzero parameters. In addition, a number of low-precision techniques [9], [12]–[14] have been investigated with minimal degradation in the classification accuracy. In [9], the data precision is lowered to the extreme where both weights and neuron activations are binarized to $+1$ or -1.

On the hardware side, many digital application-specific integrated circuit (ASIC) designs in CMOS (e.g., IBM TrueNorth [15], Eyeriss [16], and ENVISION [17]) have been previously presented to help bring expensive algorithms to a low-power processor. However, limitations still exist on memory footprint, on-/off-chip communication, and accuracy-energy tradeoff. It is still a challenging task to enable essential deep learning processors in mobile, wearable, Internet of Things (IoT), and extreme implantable devices due to their divergent constraints in low power and small footprint.

In particular, the CMOS ASIC designs [15]–[19] show that accessing memory is the biggest bottleneck for energy-efficient real-time cognitive computing in terms of storing millions of parameters, loading them from embedded SRAM memory, and moving them to where computing actually occurs. Although SRAM technology has been following the CMOS scaling trend well, to compute Multiply-and-ACcumulate (MAC) operations in DNNs, conventional SRAMs still require millions to billions of row-by-row accesses, which limits the parallelism and dissipates a large amount of read/write energy.

To improve this limitation, in the last couple years, several works proposed the *in-SRAM computing* concept (see Fig. 1), which performs computation in the SRAM hardware without reading out each row of SRAM to a computing unit [20]–[30]. However, most prior works only demonstrated small DNNs for MNIST data set with relatively low accuracy (less than 96%) [22], [23], [25], [26]. A few works [27], [30] demonstrated deep convolutional neural networks (CNNs) based on in-memory computing hardware for CIFAR-10 or

Manuscript received May 10, 2019; revised July 28, 2019; accepted August 19, 2019. This work was supported in part by NSF under Grant 1652866; in part by the Center for Brain-Inspired Computing (C-BRIC), one of the six centers in Joint University Microelectronics Program (JUMP); and in part by the Semiconductor Research Corporation (SRC) Program sponsored by Defense Advanced Research Projects Agency (DARPA). *(Corresponding author: Shihui Yin.)*

S. Yin, M. Kim, and J.-S. Seo are with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: syin11@asu.edu; jaesun.seo@asu.edu).

Z. Jiang and M. Seok are with the Department of Electrical Engineering, Columbia University, New York City, NY 10027 USA.

T. Gupta is with Synopsys, Mountain View, CA 94043 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/TVLSI.2019.2940649

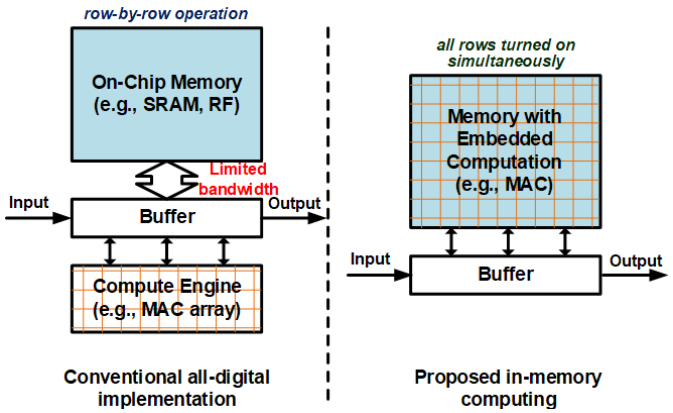


Fig. 1. Left: conventionally SRAM data are read out row-by-row to perform computation at the periphery. Right: in-memory computing schemes embed logic computation inside SRAM by turning on all rows simultaneously.

larger data sets, but in the Neural Cache work in [30], only two rows were activated simultaneously (less parallelism), and the fixed function ASIC presented in [27] lacks scalability for large CNNs.

Recently, we demonstrated a new SRAM macro that can perform MAC along the bitline (BL) of the SRAM macro with all rows turned on simultaneously [31]. Titled as XNOR-*SRAM*, it performs XNOR-and-ACcumluate (XAC, replacing MAC in the binary-weight DNN) fast and energy efficiently, and supports core computing primitives of the state-of-the-art DNNs and CNNs, achieving highly competitive classification accuracy. In 65-nm CMOS, the XNOR-SRAM chip measurements reported 235.5-pJ energy and 54.21-ns latency for completing 64 operations of 256-input XAC at 1 V. A well-crafted digital hardware, which computes the same XAC but has to read out each row of weights from off-the-shelf SRAM, takes 7.81 nJ and 514 ns for 64 256-input XAC operations, which are, respectively, 33× higher energy and 9.5× longer latency than those of XNOR-SRAM [31]. When the convolution and MAC operations of DNNs are measured with the single XNOR-SRAM array hardware and other computations (e.g., max-pooling and batch normalization) are simulated digitally, 85.7% accuracy for CIFAR-10 data set and 98.3% for MNIST data set have been reported.

Although a large amount of energy reduction is demonstrated, it should be noted that XNOR-SRAM and most in-SRAM computing works that turn on all rows or columns simultaneously only demonstrate a relatively small custom SRAM array at the single-array level [23], [26], [31]. To implement an overall DNN accelerator using in-memory computing SRAMs, there are several important challenges and missing pieces, including integration of many in-memory computing SRAM arrays, activation storage/communication, additional digital logic for non-MAC operations, and row-by-row write energy consideration.

In this article, we substantially expanded the single-array-level prior XNOR-SRAM design [31] toward a configurable DNN accelerator architecture that integrates 72 XNOR-SRAM arrays with interarray communication and supports a wide range of DNN/CNN algorithms with configurable activation precision. The proposed accelerator supports 3 × 3 and 1 × 1 convolutional kernels and up to 256 feature maps in a convolutional layer.

The main contributions of this article are as follows.

1) We construct the chip-level DNN/CNN accelerator architecture that employs many instances of in-memory computing SRAM (e.g., XNOR-SRAM) macros with a methodology to efficiently load/map weights onto such XNOR-SRAM arrays for convolution layers and fully connected layers of DNNs.
2) By reusing the XNOR-SRAM macro originally designed for binary activations and weights, we add peripheral digital logic that can support multibit activations, which becomes an effective knob to favorably tradeoff energy versus accuracy.
3) We employ double-buffering technique with two groups of in-memory computing SRAMs, which effectively hides the latencies of reprogramming in-memory computing SRAM arrays with new DNN weights.
4) We evaluate the chip-level energy benefits and remaining bottlenecks of in-memory computing-based DNN accelerators.

The remainder of this article is organized as follows. In Section II, we review recent DNN hardware designs, including digital accelerators and in-SRAM computing macros. In Section III, we present the XNOR-SRAM macro and the chip prototype results.

Section IV discusses practical design challenges when we design a chip-level DNN accelerator using many in-memory computing macros, such as XNOR-SRAM. In Section V, we describe the microarchitecture of the proposed accelerator, optimal precision study, and the methodology to efficiently map optimized DNNs/CNNs onto XNOR-SRAM arrays. Section VI reports experimental results on speed, energy dissipation, and classification accuracy across several workloads. Finally, this article is concluded in Section VII.

II. Related Works

A. Recent Digital Accelerators for DNN

A number of digital ASIC designs [16]–[19] have been presented to accelerate the inference or classification phase of DNNs with enhanced energy efficiency. We briefly introduce the representative ASIC accelerators in the following.

Eyeriss [16] proposed a new dataflow called row stationary (RS) on a spatial architecture with 168 processing elements. RS dataflow reconfigures the computation mapping of a given shape, which optimizes energy efficiency by maximally reusing data locally to reduce expensive data movement, such as DRAM accesses. ENVISION [17] presented an energy-scalable CNN processor achieving efficiencies up to 10 TOPS/W, where the data precision of MAC units can be dynamically scaled from 4 to 16 bit, together with voltage/frequency scaling and body bias modulation. DNPU [18] demonstrated a reconfigurable processor that can efficiently map both CNN and recurrent neural network (RNN) algorithms. It employed dynamic fixed-point precision with online adaptation via overflow monitoring. It also proposed a quantization table-based matrix multiplication to reduce off-chip memory access. In [19], an accelerator for

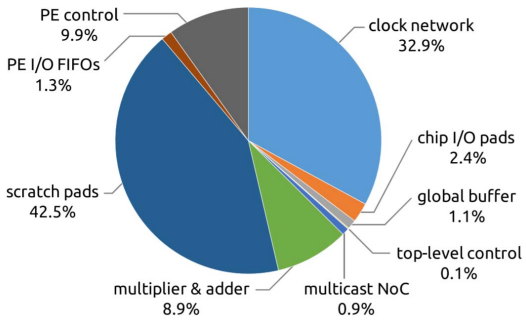


Fig. 2. Power breakdown of Eyeriss chip [16] running the first convolution layer of the AlexNet CNN [1]. Besides the clock network, most power is consumed by on-chip SRAM (scratch pads and global buffer).

	Biswas et al. [23] ISSCC 2018	Khwa et al. [26] ISSCC 2018	Valavi et al. [27] Symp. VLSI 2018	XNOR-SRAM [31] Symp. VLSI 2018
CMOS technology	65nm	65nm	65nm	65nm
SRAM bitcell	10T	6T	9T + MOMCAP	12T
SRAM array size	256x64	512x256	192x192	256x64
Supply voltage	0.9-1.2V	1V	0.68-1.2V	0.6-1.0V
Column/row sensing	Integrating ADC (scalable)	Sense amplifier (not scalable)	Sense amplifier (not scalable)	Flash ADC (scalable)
Demonstrated DNN	CNN with binary weights (+1/-1) and 7-b inputs	FC layer of CNN with binary weights (+1/-1), binary inputs (+1/0)	CNN with binary weights (+1/-1), binary inputs (+1/-1)	MLP and CNN with binary weights (+1/-1), ternary inputs (+1/0/-1)
Energy efficiency	28.1 TOPS/W	55.8 TOPS/W	658 TOPS/W	403 TOPS/W
MNIST accuracy	96%	95.1%	98.6%	98.3%
CIFAR-10 accuracy	N/A	N/A	83.3%	85.7%

Fig. 3. Comparison of recent in-SRAM computing hardware demonstrations.

fully connected DNNs is presented, featuring circuit and algorithmic error resilience with *in situ* timing error detection and correction techniques (also known as Razor [32]).

In the Eyeriss chip results [16] of running AlexNet CNN [1], Fig. 2 shows that the arithmetic logic unit (ALU) operations only accounted for less than 10% of the total power. Memory access and data movement-related components, including on-chip SRAMs, buffers, and network-on-chip, accounted for up to 45%. This confirms that memory access and data movement are actually significantly more energy-consuming than ALU operations. Eyeriss chip consumes relatively high clock power (33% of total power), but the authors acknowledged that this can be considerably reduced with proper clock-gating techniques [16]. Therefore, we can conclude that the on-chip SRAMs indeed dominated on-chip power consumption, which infers that in-memory computing has a large potential to effectively reduce memory accesses and substantially improve the overall system energy efficiency of DNN hardware accelerators.

B. Recent In-Memory Computing Hardware Designs

Recently, a large amount of attention has been drawn to develop DNNs that only use binary ($+1$ and -1) weights, demonstrating orders of magnitude reduction in computational complexity at tolerable accuracy degradation [9], [13], [33], [34]. This advent of the binary-weight DNNs and CNNs opens a new possibility for SRAM-based in-memory computing, since each weight in those algorithms can be nicely stored in a single SRAM bitcell. By turning on multiple or all rows simultaneously, the input/activation values are applied as wordline (WL) voltages, which, in turn, interact with the bitcells to perform MAC computation, typically in an analog manner. This can eliminate explicit memory access, which otherwise pose energy/performance bottlenecks in DNN/CNN hardware implementations. A number of works have recently demonstrated this type of in-SRAM computing [20], [22]–[24], [26], [27], [29]–[31], and we summarized representative works in Fig. 3.

In [22], in-SRAM computing hardware in 130-nm CMOS was demonstrated. This design employs binary weights, each of which is stored in a 6T bitcell. The 5-bit inputs are converted to analog voltages via digital-to-analog converters (DACs) embedded in the address decoder, which drives the WLs. Each WL voltage modulates the resistance of access transistors of bitcells of that row. Depending on the weight stored in each bitcell, the bitcell either discharges or charges the BLs, making BL voltage proportionally grow with the MAC computation results. The BL voltage is finally digitized into a binary value by a single-sense amplifier in the column circuitry.

This article employs 6T SRAM circuits, promising a compact silicon footprint. However, it cannot support mainstream DNN and CNN algorithms. Even though binarized neural network (BNN) algorithms [9], [33] binarize the activation at the output of each layer, still the partial sum and accumulation need to be performed with high precision. Since the partial sum result at each SRAM output is binarized prematurely in [22] and if the neural network layer cannot fit in one SRAM array, the final neural network accuracy can be considerably degraded. By combining many weak classifiers (shallow neural networks), a boosting classifier is demonstrated, but only achieved 90% accuracy for MNIST data set.

In-SRAM computing hardware was also presented in [20]. This was not for accelerating neural network algorithms but for content addressable memory (CAM) and bitwise AND/OR/XOR operations of two rows of SRAM. The main focus of this article was on the novel 6T bitcell design featuring portless write using n-well and decoupled differential read, scaling the lowest functional voltage down to 0.3 V in a 55-nm CMOS technology.

Conv-RAM [23] integrates DACs for analog inputs, binary weights stored in SRAM, and analog-to-digital converters (ADCs) to convert the in-memory computation results back to digital values. In-memory computation in [23] targets convolution operation, which is accomplished by rowwise charge sharing of the SRAM bitcells in the same row. To perform this, local analog multiply-and-average circuits are added every 16 rows (out of the 256-row custom SRAM array). However, within a block of 16 rows, the in-memory SRAM still goes through row-by-row operation, and the integrating ADC exhibits slow speed. MNIST accuracy of 96% was reported, but only 100 test images were used for the accuracy calculation.

In-memory computing with on-chip training capability was presented in [24], where the weights were fine-tuned based on on-chip variability. The work reported an accuracy of 96% on the MIT-CBCL data set for relatively simple face detection tasks. In-memory computation in this article is limited to reading out multibit weights that are stored in different

rows to a single analog voltage, instead of performing an MAC or convolution operation inside the memory.

Khwa *et al.* [26] demonstrated MAC operations using a 4-kb SRAM for fully connected neural networks in edge processors. This article proposed techniques to mitigate the challenges of excessive current, sense-amplifier offset, and sensing reference voltage optimization, arising due to simultaneous activation of multiple WLs in the in-memory computing scheme. However, similar to [22], each column output is binarized with a single-sense amplifier, which limits the accuracy and scalability to arbitrary large DNNs. In addition, only fully connected layers of DNNs are mapped onto in-SRAM computing, and the measured MNIST accuracy was limited to 95.1%.

Neural Cache [30] repurposes the large last-level caches in microprocessors toward CNN acceleration with in-memory computing. However, the in-memory computing scheme only turns on *two* rows of SRAM in one cycle, which limits the parallelism.

PROMISE [29] is a programmable mixed-signal accelerator that supports diverse machine learning algorithms with a custom instruction set architecture (ISA) and compiler support. However, they only demonstrated machine learning tasks on relatively simple benchmarks, including MNIST and MIT-CBCL data sets.

Recently, a binarized CNN accelerator was presented in [27], which also performs a modified batch normalization with analog computation, and reported 83.27% test accuracy for CIFAR-10 data set. However, since each column end is binarized with a single-sense amplifier, it cannot naturally support ensuing high-precision operations such as max-pooling and also lacks scalability for larger CNNs.

In this article, we are focusing on SRAM as the memory substrate of in-memory computing. On the other hand, researchers have also presented in-memory computing designs based on emerging nonvolatile memory technologies, such as phase-change memory (PCM) [35], [36], resistive RAM (RRAM) [25], [37], and magnetic RAM (MRAM) [38], [39]. As compared to SRAM, these technology promises a bitcell that can store multibit weight, often in the form of analog variables (e.g., resistance), in a smaller footprint. This is a significant benefit to supporting DNN and CNN algorithms using multibit weights. However, these emerging memory devices exhibit significant challenges, including high variability and nonlinearity [40], [41], and most importantly, the manufacturability has not reached that of CMOS technologies, which severely limits system-level integration with many large arrays.

III. XNOR-SRAM: Scalable SRAM Macro for In-Memory Computing

A. XNOR-SRAM Macro Design

In most of the aforementioned prior in-SRAM computing works, only DNNs for MNIST data set have been demonstrated [23], [26], or the scalability for large DNNs is limited since only a single-sense amplifier exists at each column end of the in-memory computing SRAM [22], [26], [27].

In [31], a new mixed-signal in-memory computing SRAM macro titled "XNOR-SRAM" was presented, which extends the scalability and efficient mapping capability of a wide range of neural networks. It performs XNOR-and-ACcumulate (XAC) operations in BNNs [9], [33], which replaces MAC operations in nonbinary DNNs, with high speed and energy efficiency.

Fig. 4(a) presents the reported XNOR-SRAM array and peripheries, which can map convolution and fully connected layers of CNNs and multilayer perceptrons (MLPs). It consists of a 256-by-64 custom SRAM array, a row decoder, and a read periphery including a 3.46-bit (11-level) flash ADC. Two modes of operations exist for XNOR-SRAM. In the memory mode, it performs row-by-row digital read and write as regular memory circuits. In the XNOR mode, it performs in-memory MAC computation with all rows asserted simultaneously.

Fig. 4(b) shows the 12T SRAM bitcell proposed in [31]. T1–T6 form the conventional 6T SRAM bitcell; T7–T10 form complimentary pull-up/-down circuits for the XNOR mode; T11 and T12 can power-gate the pull-up/-down circuits, saving power when the corresponding column is not needed for computation. In each bitcell, a binary weight ($+1/-1$) is stored in the 6T SRAM and input signal ($+1/-1/0$) is represented by four RWL signals (RWL_P, RWL_N, RWLB_P, and RWLB_N), as shown in Fig. 4(c). When T11 and T12 are on, pull-up or pull-down paths are formed depending on the product of weight and input [see Fig. 4(c)]. For example, when the product is $+1$, one strong pull-up path and one weak pull-up path are formed; when the product is -1, one strong pull-down path and one weak pull-down path are formed.

Parallel pull-up and pull-down paths from all bitcells (controlled by bitwise XNOR outputs) in a column form a voltage divider, where RBL is the output node. V_{RBL} is a monotonic function of XNOR bitcount [see Fig. 4(d)]; therefore, we can obtain the XAC results by digitizing V_{RBL} with the ADC. Due to the relatively large ADC area overhead, we share one ADC across 64 columns. For a given binary/ternary 256-dimension input vector, a 6-to-64 column decoder along with 64-to-1 analog multiplexer [see Fig. 4(a)] is employed to cycle through all the 64 columns in 64 cycles. When a column is selected, REN/RENB are set to VDD/0 and the pull-up/-down paths are connected at RBL; REN/RENB are set to 0/VDD for other 63 columns, where both T11 and T12 are turned off, breaking the short-circuit path between pull-up/-down paths. In one cycle, XNOR-SRAM supports computation with binary weights ($+1/-1$) and binary inputs ($+1/-1$ or $+1/0$) as well as ternary inputs ($+1/0/-1$). The embedded ADC plays a key role in speed and DNN accuracy. Employing 11 levels (3.46 bit) reportedly provides relatively high accuracy, and nonlinear quantization based on statistical distribution of XAC values can further improve it. Fig. 4(d) shows the measured V_{RBL} for different XAC values. In the nonlinear quantization scheme, the worst case 3-σ deviation is equivalent to 1.78 LSB. Note that the worst case deviation is smaller for other V_{RBL} values, e.g., 0.83 LSB when V_{RBL} is 0.25 VDD.

B. XNOR-SRAM Characterization Results

XNOR-SRAM macro was prototyped in 65-nm CMOS and achieves 81.28 pJ and 178 ns for 64 operations

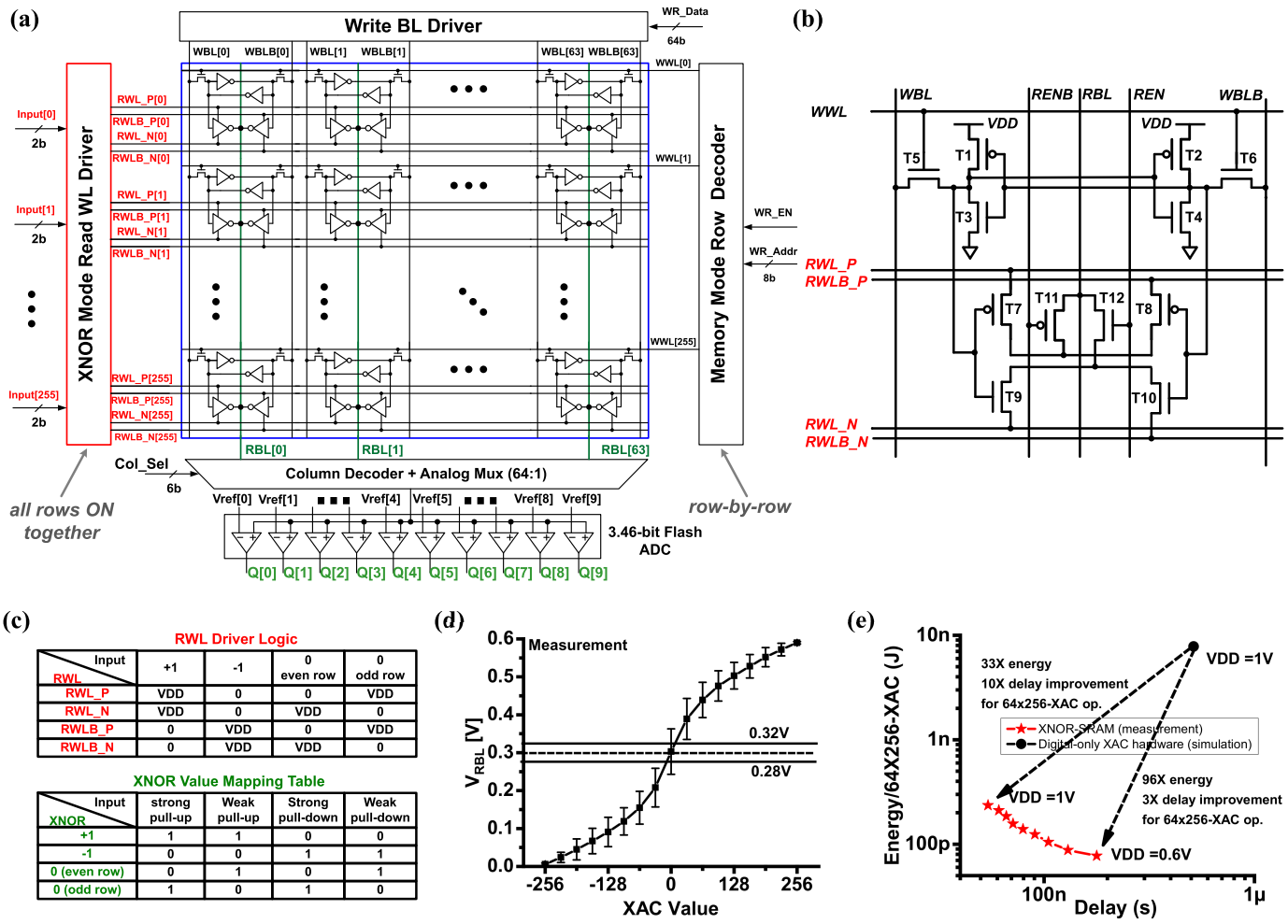


Fig. 4. XNOR-SRAM design proposed in [31]. (a) XNOR-SRAM macro can map the XNOR-and-accumulate operation of binary-weight DNNs. The top shows logical computation and the bottom shows the schematics of XNOR-SRAM. (b) XNOR-SRAM bitcell design. Devices T7–T12 are added to a 6T SRAM. (c) XAC operation with ternary activations and binary weights. (d) Measured V_{RBL} over corresponding logical results of 256-input XAC operations. The 3-σ worst case error due to analog-mixed-signal variability is less than 1.78 Least Significant Bits (LSBs). (e) 300× EDP improvement over the baseline using conventional SRAM.

of 256-input XAC at 0.6 V. We also designed a well-crafted digital accelerator in 65-nm CMOS, which serves as the baseline for comparison. This digital baseline computes the same XAC but has to read out each row of weights from off-the-shelf SRAM. The postlayout simulation based on industrial off-the-shelf SRAMs, standard cells, and parasitic-annotated netlists showed the digital baseline consumes 7.81 nJ (4.26 nJ for conventional SRAM access) and 514 ns for 64 256-input XAC operations, which are, respectively, 96× and 2.9× worse than the XNOR-SRAM hardware. The main reason why XNOR-SRAM is more energy efficient than conventional SRAM is the large reduction in BL switching. In a conventional SRAM with N columns and M rows, $2 \times N \times M$ BL switching events are required to read out all the contents (the 2 in the equation is from the precharging operation) for XAC operations in the ALU. In comparison, XNOR-SRAM only needs N BL switching events. In addition, the signal swing of the BL switching in XNOR-SRAM is most likely less than a full rail voltage, further saving power over SRAM. Note that the ALU consumes additional power. On the other hand, while there are four WLs per row in XNOR-SRAM compared to only one WL per row in conventional SRAM, the four WLs switch only when the activation value for that row changes.

Classification accuracies for MNIST and CIFAR-10 data sets have been evaluated on a single XNOR-SRAM macro in [31]. For MNIST, an MLP consisting of three hidden layers, each with 512 neurons, was used. For CIFAR-10, a CNN consisting of six convolutional layers and three fully connected layers [33] was used. Accumulation of the XNOR-SRAM outputs, max-pooling, and batch normalization were performed in digital simulation with bit precisions of 12, 12, and 10, respectively. As shown in Fig. 4(e), the DNNs with XNOR-SRAM (digital baseline) achieve 85.7% (90.7%) accuracy for CIFAR-10 and 98.3% (98.8%) for MNIST. Again, Fig. 3 shows the comparisons of the XNOR-SRAM-based in-memory computing hardware to other recent works. XNOR-SRAM-based design achieves both high accuracy and ultralow energy, the former comparable to the accurate digital hardware and the latter comparable to the other mixed-signal in-SRAM computing hardware.

IV. Practical Challenges of In-Memory Computing-Based Accelerators

Although XNOR-SRAM as well as other in-memory computing SRAMs show promising energy efficiency at the single-array-level, there are several important challenges and missing pieces toward building a chip-level DNN accelerator using these arrays.

A. Integration of Many In-Memory Computing SRAM Arrays

First, it should be noted that XNOR-SRAM as well as most of the in-memory computing hardware only demonstrated a relatively small custom SRAM array (around a few hundred kb or less). Therefore, to implement an overall DNN accelerator, many of these in-memory computing SRAM arrays need to be employed and integrated together with on-chip communication networks.

B. ADC Overhead and Offset Cancellation

Second, the ADC design incurs area and power overhead, which adversely affects the array efficiency and energy efficiency. In addition, ADC performance is sensitive to offset or variability, and for DNN applications, evaluation should be made on how much DNN accuracy degradation occurs due to the variability. Ideally, the ADC offsets should be calibrated out using offset compensation circuits [42], but such calibration circuits will increase the area/power further.

C. Postprocessing Modules

Third, toward evaluating accuracy for CNNs or DNNs, these designs employ a considerable amount of postprocessing modules, such as partial sum accumulation, batch normalization, pooling, nonlinear activation, and so on. These postprocessing modules add system-level design complexity and energy consumption beyond those of the single in-memory computing SRAM array.

D. Activation Storage and Communication

Fourth, typically, the in-memory computing SRAM arrays store the DNN weights, while the activations are applied as inputs to the custom SRAM array either at the WLs or BLs. Different activation values need to be applied to the SRAM array every cycle, which means that the activations need to be stored in a separate memory (e.g., another SRAM array) and communicated to the in-memory computing SRAM array at the right time. However, the energy of activation storage and communication are typically not included in the in-memory computing SRAM macro energy values.

E. Write Energy

Finally, since the state-of-the-art DNNs can be very large, we will not be able to store the entire weights of DNNs in in-memory computing SRAMs without consuming a huge amount of area and static power. To that end, it will be necessary to reload different weights' in-memory computing SRAM arrays at different times. While XAC operation for in-memory computing SRAMs can be fully parallelized by turning on all the rows, write operation of loading new weights to the SRAM still requires row-by-row operation, which incurs long latency and consumes write energy. Typically, the reported in-memory computing SRAM macros do not include any write energy, but for a system-level accelerator, the corresponding write energy of in-memory computing SRAM macros should be characterized and included.

In Section V, we will describe how a number of these key challenges have been addressed and inclusively implemented in the proposed accelerator design and will report the accelerator evaluation results.

V. Vesti Accelerator Design

A. Microarchitecture Overview

We designed the microarchitecture of the Vesti accelerator, which integrates the aforementioned key missing pieces, and can execute inference for a wide range of DNNs/CNNs. Fig. 5(a) shows the overall microarchitecture that computes a deep CNN inference on a layer-by-layer basis. Employing the double-buffering scheme [43]), it consists of two symmetric cores, one of which performs the in-memory computation, while the other can load new weights from an on-chip global buffer or an off-chip DRAM. Each block consists of: 1) the ensemble of the XNOR-SRAM macros that perform XAC from convolution and fully connected layers and 2) a digital ALU that performs other operations, such as batch normalization, activation, and max-pooling. In this article, 36×2 XNOR-SRAM macros are employed (see Fig. 5) to support representative CNNs for CIFAR-10 data set [33]. To support larger CNNs, the XNOR-SRAM macros can be time-multiplexed and reused over time. At 1 V, the XNOR-SRAM macro can operate at 1.2 GHz and the digital ALU was synthesized and placed/routed at 0.55 GHz in the same 65-nm CMOS technology.

The inputs and weights are fetched from off-chip DRAM. The inputs are saved in the activation memory buffer and weights are written into the XNOR-SRAM macros. The 36 XNOR-SRAM macros in each core are divided into four groups. The four groups share the input activations, performing XAC operations for up to 256 ($=64 \times 4$) output feature maps in parallel. The nine XNOR-SRAM macros in each group can accept inputs from up to 256 input feature maps when performing 3×3 convolution and up to 2304 ($=256 \times 9$) inputs when performing fully connected matrix vector multiplication. In each cycle, up to 36 256-input XAC operations can be executed in parallel. The outputs of 36 XNOR-SRAM macros are processed by a 256-way digital ALU, where ADC output decoding, partial sum accumulation, max-pooling, batch normalization, and binary/ReLu activation are performed. The results will be saved back to the activation memory buffer of the other core, which will perform computation for the ensuing layer in a similar fashion.

Double-buffering technique [43] is employed to hide the weight loading latency. While one core is performing computation for one layer, the other core loads the weights for the next layer. For example, for two adjacent convolution layers, of which the channel size and map size is 256 and

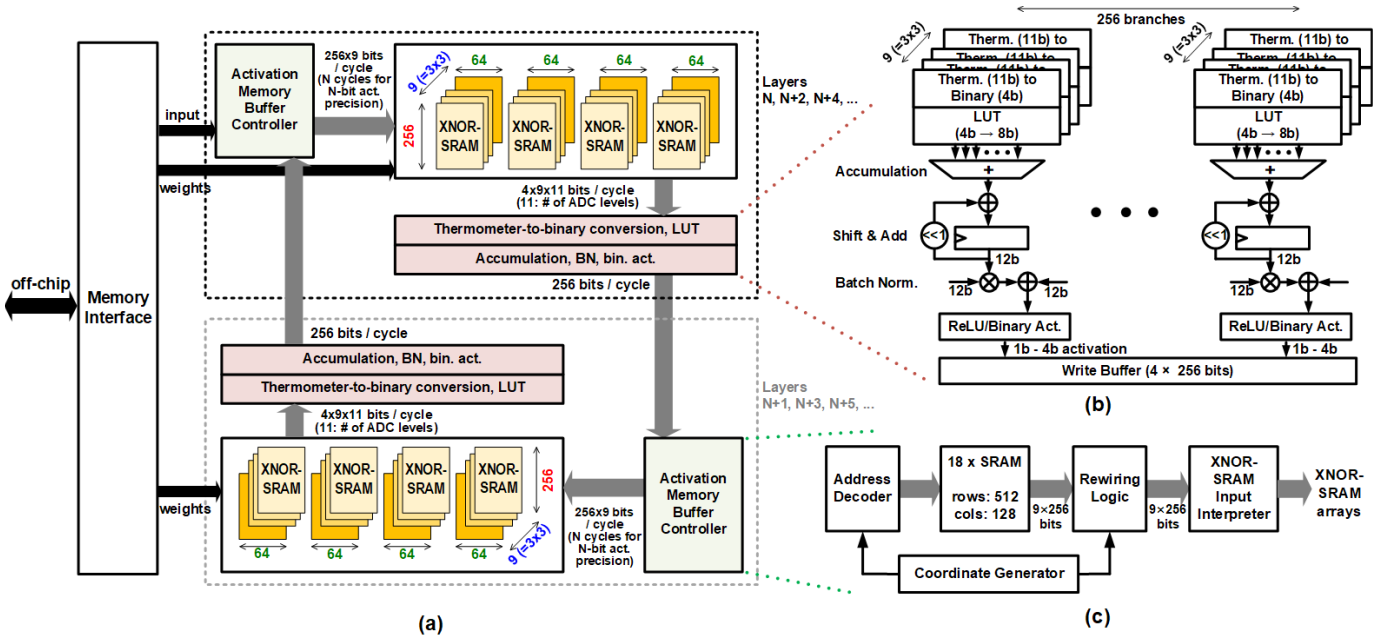


Fig. 5. (a) Overall microarchitecture of the proposed in-memory computing Vesti accelerator. (b) Computations for the thermometer-to-binary conversion, LUT, batch normalization, and so on. (c) Block diagram of the activation memory buffer.

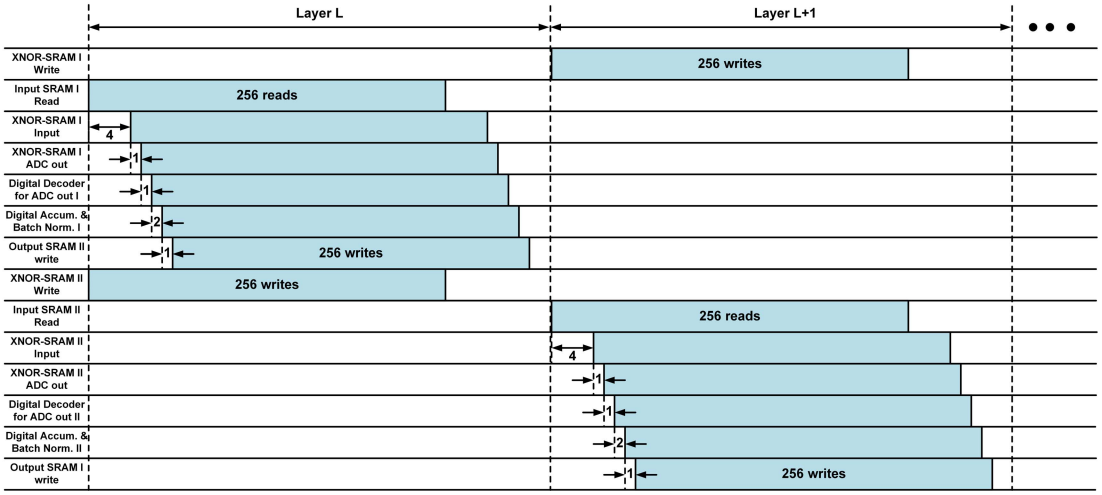


Fig. 6. Timing diagram of the accelerator operation for two adjacent layers of a CNN, including in-memory computing, double-buffering, and peripheral computations.

16 × 16, respectively, the 36 XNOR-SRAM macros in one core complete the convolution in 256 cycles; at the same time, the 36 XNOR-SRAM macros in the other core load the weights for the next layer in 256 cycles (row by row). Fig. 6 shows the timing diagram that illustrates these operations for two adjacent layers of a CNN. This layer-by-layer operation will continue to perform all the layers of a given DNN/CNN.

B. Multibit Activation Support

While the XNOR-SRAM macro has native support on binary activations and binary weights, it should be noted that DNNs with binary activations and binary weights do not yet reach the same accuracy level of their higher precision counterparts [9], [33]. Therefore, in our proposed Vesti accelerator, we support weights with binary precision (+1 or −1) but activations with configurable precision from 1 to 4 bit. This scheme will not only minimize the weight memory footprint but can also reach the level of DNN accuracies with floating-point precision [13].

Our choice on binary weight and multibit activation is based on the algorithm level experiments that we conducted. In particular, we swept: 1) several CNN sizes (various numbers of feature maps per layer) for the CIFAR-10 data set and (2) activation precision values including binary, ternary, 2, 4, 8, and 32 bit. The results are shown in Fig. 7. 1× CNN represents the network of input-128C3-128C3-MP2-256C3-256C3-MP2-512C3-512C3-MP2-1024FC-1024FC-10FC, which was presented in [33]. Here, 128C3-128C3 refers to the convolution layer with 128 input feature maps, 3 × 3 kernels, and 128 output feature maps, MP2 refers to 2 × 2

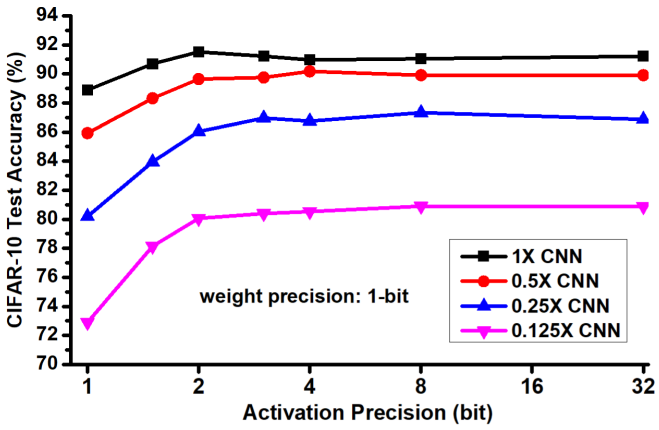


Fig. 7. Classification accuracy for CIFAR-10 data set is shown across different activation precision values for four different DNN sizes. For all data points, the weight precision is binary (only two values of +1 or −1).

max-pooling, and 1024FC refers to the fully connected layer with 1024 hidden neurons. 0.5× CNN represents the network input-64C3-64C3-MP2-128C3-128C3-MP2-256C3-256C3-MP2-512FC-512FC-10FC, where the number of feature maps in all convolution layers is reduced by half compared to those in the 1× CNN, and the number of hidden neurons in the fully connected layers is reduced by half as well. Similarly, 0.25× CNN and 0.125× CNN represent the networks, where all dimensions are reduced by 4× and 8×, respectively, compared to the 1× CNN.

It can be seen in Fig. 7 that the accuracy of models using floating-point activation precision could be reached by employing 3-/4-bit activation precision with binary weights. However, employing binary activations do show considerable degradation in CNN accuracy. To that end, we use in-memory computing hardware with fixed binary weights, but employ configurable precision for the inputs and neuron activations at the periphery of the XNOR-SRAM array.

In the microarchitecture level, the support of the multibit inputs is done by performing XAC operation for each bit of input/activation using XNOR-SRAM macros and then shift-and-accumulate the bitwise XAC results in the digital peripheries over multiple cycles (e.g., N cycles for N-bit precision of activations). This is shown in Fig. 5(b) together with other digital computations at the periphery. Configurable precision (from 1 to 4 bit) for activations can be flexibly supported at the cost of additional clock cycles.

C. Activation Memory

Activation memory in each core is employed to store the input feature maps for that core and to save the output activations from the other core. Take a 256C3-256C3 convolution layer as an example. This convolution layer consists of 256 input feature maps, each of 32 × 32 pixels. The input feature maps stored in an activation memory need to be read out to perform convolution. 256 × 3 × 3 pixels should be fetched out and presented to the 36 XNOR-SRAM macros at every cycle. A relatively large FIFO-based buffer between the activation memory and XNOR-SRAM array can reduce activation memory access by exploiting convolutional data reuse. However, this will result in considerable power consumption in the buffer and significant area for the control logic.

To get rid of the buffer, we propose a new way to store the feature maps in the activation memory. In particular, we divide the overall activation memory into nine activation SRAM blocks of 128 rows and 256 columns, as shown in Fig. 8. The input feature maps are divided into 3 × 3 tiles. The input feature maps' pixels are grouped and stored in the nine SRAM blocks according to their position in the 3 × 3 tiles they belong to. By storing pixels in this fashion, we can read all the 256 × 3 × 3 pixels in a single cycle given the fact that any 3 × 3 patch of input feature maps is now stored in nine different SRAM blocks. A controller block is designed to generate corresponding addresses for the nine SRAM blocks.

As shown in Fig. 5(c), the activation memory buffer has four parts: 1) coordinate generator; 2) address decoder; 3) rewiring logic; and 4) XNOR-SRAM input interpreter module, along with the 18 SRAM blocks. The functionalities of these blocks are described in more detail in the following.

1) Coordinate Generator: The coordinate generator block generates the x- and y-coordinates of the output map pixels sequentially in a row-major order. In case that the convolutional layer is followed by a max-pooling layer, the coordinates correspond to each pooling window will be generated in a row-major order inside each pooling window to ease the buffer size requirement for max-pooling operation. These coordinates will serve as inputs to our address decoder and rewiring logic blocks to different combinations of row addresses and read enable signals for SRAM blocks.

2) Address Decoder: The address decoder block generates activation memory addresses for the nine SRAM blocks according to the output feature map coordinates. Zero padding can be supported smoothly by the address decoder as well. When the generated addresses are found invalid for the input feature maps, corresponding read enable signals will be inactive and substitute the SRAM output with zero values. Since the feature map size and channel size vary from layer to layer, we further divide each 256-bit-word-length SRAM block into two 128-bit-word-length SRAM blocks. For some layers (e.g., map size = 32 × 32 and channel size = 128), we concatenate these two in depth direction to form a deeper SRAM block with 128-bit word length; for some layers (e.g., map size = 16 × 16 and channel size = 256), we concatenate these two in word direction to form a wider SRAM block with 256-bit word length.

3) Rewiring Logic: As shown in Fig. 8, although we can always fetch any 3 × 3 patch in input feature maps from nine different SRAM blocks at the same time, the order of the nine SRAM outputs do not always align with the row-major order in each 3 × 3 patch. We need to rewire the SRAM outputs to make sure the 3 × 3 patches be presented to the XNOR-SRAM macros in correct order. There are nine different rewiring patterns in total, depending on the 3 × 3 patch row and column offset remainder modulo by 3. Fig. 8 shows three different patterns with an example of 6 × 6 feature maps,

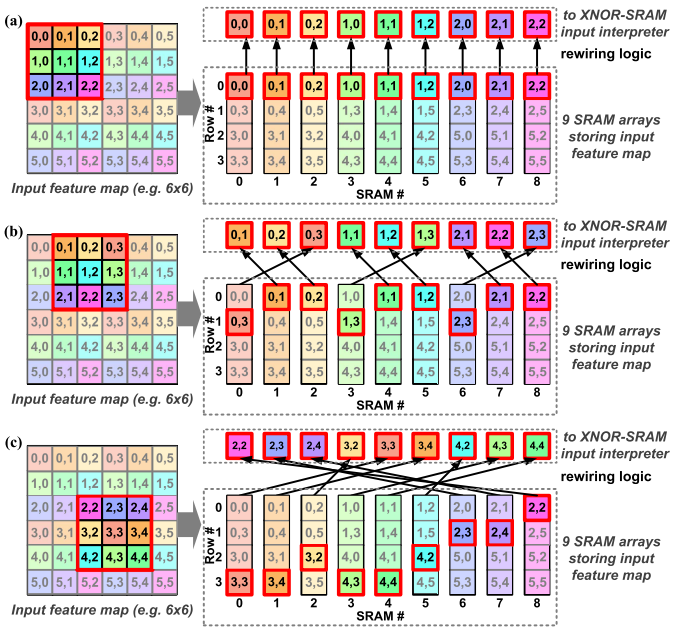


Fig. 8. Illustration of convolution layer feature map storage and access scheme in nine independent SRAM arrays. (a) 3 × 3 window starting from (0, 0). (b) 3 × 3 window starting from (0, 1). (c) 3 × 3 window starting from (2, 2).

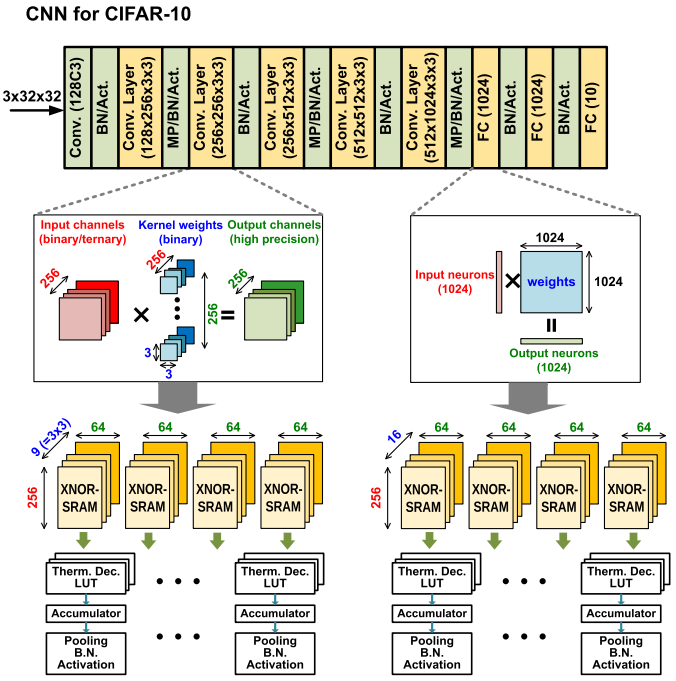


Fig. 9. Mapping convolution layers (left) and fully connected layers (right) of deep CNNs onto the Vesti accelerator employing XNOR-SRAM macros with in-memory computing.

where the patch row and column offset is (0, 0), (0, 1), and (2, 2), respectively.

4) XNOR-*SRAM Input Interpreter:* Depending on whether we operate the XNOR-SRAM in 1-bit binary activation (+1/−1) or multibit binary activation (+1/0) mode, the XNOR-SRAM input interpreter will generate proper WL inputs for XNOR-SRAM array from what it receives from the rewiring logic.

D. Mapping of Convolution, Fully Connected, and Other Layers

For convolution layers, we propose a mapping scheme where the same location pixel (x, y) of the kernel from all the kernels for different input/output feature maps will be stored in the same XNOR-SRAM array. Other location pixels of the kernels will be stored in different XNOR-SRAM arrays. This is shown in Fig. 9 (left). Then, the XAC or bitcount accumulation results will be gathered from these multiple XNOR-SRAM arrays and accumulated together to obtain the final output activation result. This scheme enables extensive reuse of the activations, with weights being stationary at the XNOR-SRAM arrays.

Using the weight-stationary scheme in the XNOR-SRAM macros, it is straightforward to map fully connected layers of DNNs, where neurons/activations are in vectors and weights are in matrices. This nicely maps to the row drivers for activations and weights stored in the SRAM. For the fully connected layers whose size is larger than 256 × 64, we break the large weight matrix into a number of small submatrices and accumulate the matrix-vector multiplication results accordingly. This is shown in Fig. 9 (right).

We implement other computation modules such as max-pooling, batch normalization, and nonlinear activation with all-digital circuits, which will reside at the periphery of XNOR-SRAM macros. The computing sequences of these modules are as follows. Once the XAC operations are done inside the XNOR-SRAM array, they are digitized with the flash ADC and output a thermometer code. The thermometer code from each column gets converted to a binary number and a simple lookup table (LUT) is used to bring it back to the exact bitcount value, according to the nonlinear quantization scheme that was employed with regard to the ADC reference voltages. Then, the same columns from different XNOR-SRAM arrays are accumulated (to compute the summation of the partial sums of all convolution pixels), which then sends out the final output sum value. Using the trained batch normalization parameters, this final sum value goes through batch normalization and nonlinear quantization (thresholding for binary/ternary activations and ReLU for multibit activations). Finally, when the current layer's computation is completed, the activation outputs are ready to serve as the input activations for the next layer of the DNN/CNN.

VI. Experimental Results

A. Experiment Setup

The Vesti accelerator consists of two main parts: 1) multiple instances of XNOR-SRAM arrays (including ADC peripheries) and 2) activation SRAMs and additional digital logic/control. As reported in [31], the XNOR-SRAM array itself consists of a custom SRAM array and peripheral mixed-signal circuits, which are designed and laid out manually. The remaining modules, including activation SRAMs and additional digital logic/control, are implemented through the standard cell-based design flow. Activation SRAMs are off-the-shelf 6T SRAMs and are obtained from industrial memory compiler. Digital

logic/control modules are implemented in RTL, synthesized using Synopsys Design Compiler, and placed and routed using Cadence Innovus tool in the same 65-nm CMOS technology.

We characterized the power/energy consumption, throughput, and accuracy (for MNIST and CIFAR-10 data sets) of the Vesti accelerator. For all the MAC or XAC operations in the convolution and fully connected layers, the XNOR-SRAM testchip measurement results from [31] are used. For all other digital logic and off-the-shelf SRAMs, the power consumption results are obtained from Synopsys PrimeTime simulation of the postlayout netlist with *RC* parasitics and actual data switching activity information.

We considered MLPs and CNNs for image classification tasks for MNIST and CIFAR-10 data sets, respectively. For MNIST, an MLP with three hidden layers, each with 256 neurons, is used. The CIFAR-10 CNN architecture used in this section is adopted from the CNN reported in [33], consisting of six convolution layers and three fully connected layers, which is identical to the 1× CNN that was used in Section V-B except that the last two convolution layers have 256 feature maps. This represents the network of input-128C3-128C3-MP2-256C3-256C3-MP2-256C3-256C3-MP2-1024FC-1024FC-10FC.

The ADC employed in [31] was a 11-level flash ADC, which nonlinearly quantized the analog BL voltage to produce approximate partial XAC values. Since the distribution of partial XAC values was found to be concentrated around zero, finer-grain quantization has been employed around zero XAC values in [31]. The effect of noise/offset for the ADC, as well as the process variation of XNOR-SRAM bitcells, contributes to the 3-σ spread of the analog read BL voltage for the same XAC values in Fig. 4(d).

To evaluate the accuracy of binary-weight multibit-activation MLPs and CNNs on Vesti, we first obtained a probabilistic model for the XNOR-SRAM XAC and quantization operations from XNOR-SRAM chip measurements. Specifically, we used a total of 656k (513 XAC values × 64 columns × 20 samples/XAC/column) random test vectors and measured the outputs of the XNOR-SRAM to build XNOR-SRAM's probabilistic model as a function of XAC value. We simulated BNN model accuracy on Vesti by running software simulations where we stochastically quantized all the 256-input XAC partial sums according to the measured probabilistic model. Offset calibration or other techniques that lower the ADC noise/offset can result in tighter distribution of the probabilistic model, which will result in better DNN accuracy. Note that the probabilistic model was characterized based on single-array measurement. Array-to-array variation could potentially result in further accuracy loss, while more accurate array-by-array ADC calibration could alleviate the loss. In addition, hardware-variation-aware network retraining algorithm [44] could potentially help minimize the accuracy loss due to the array-to-array variation.

B. Area, Energy, Throughput, and Accuracy

In Fig. 10, the placed-and-routed layout of all digital peripheral blocks and the 72 XNOR-SRAM arrays are shown. The total area of the Vesti accelerator is 15 mm^2 in the 65-nm CMOS process. The width and height of different modules that comprise the accelerator are shown as well. Multiple XNOR-SRAM arrays consume 54% of the total area, the activation SRAMs consume 18%, and remainder of the area (28%) is occupied by digital logic and control modules.

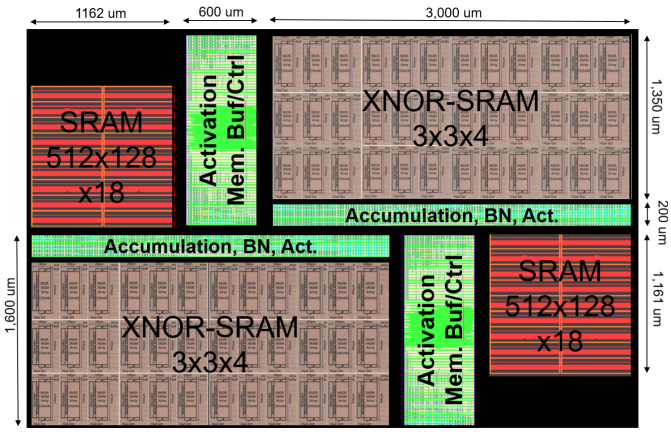


Fig. 10. Including the XNOR-SRAM prototype chip layout, the layout of activation memory buffer/controller, accumulation, and batch normalization modules are shown.

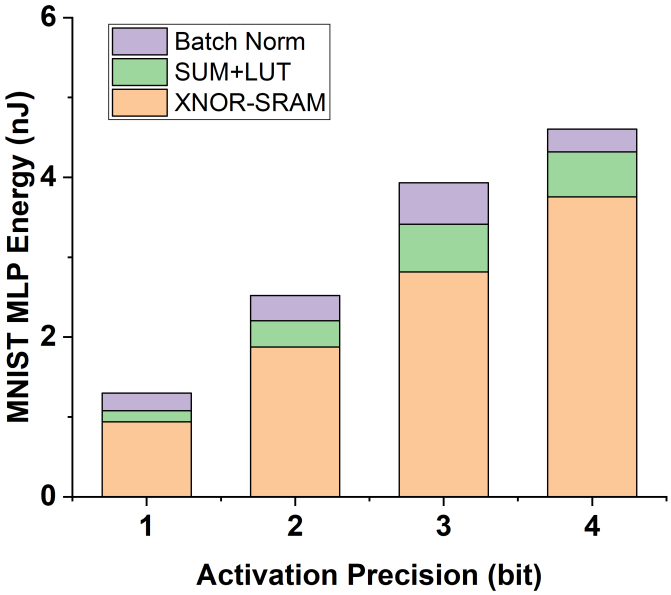


Fig. 11. Energy breakdown of the entire MLP designed for MNIST data set.

For MNIST MLP, we simulated and evaluated the total MLP energy for various activation precisions of 1–3 bits in Fig. 11. The XNOR-SRAM macro energy is acquired from chip measurements [31], and energy for other digital components is obtained from postlayout simulation with data switching activity. Since activations with N-bit precision consume N cycles to compute using the XNOR-SRAM array, the overall energy roughly increases linearly with the activation precision. To perform a single inference of the MLP using the 1-bit activation precision, the Vesti accelerator consumes 21 cycles. At the 0.55-GHz clock frequency, which the Vesti accelerator can operate at, the throughput of 26M inferences per second is achieved. Fig. 11 shows the energy breakdown across various activation precisions.

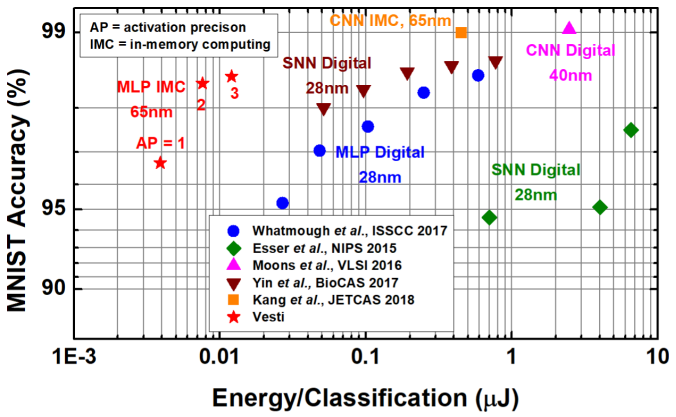


Fig. 12. For MNIST data set, accuracy versus energy (per classification) comparison with prior works is shown.

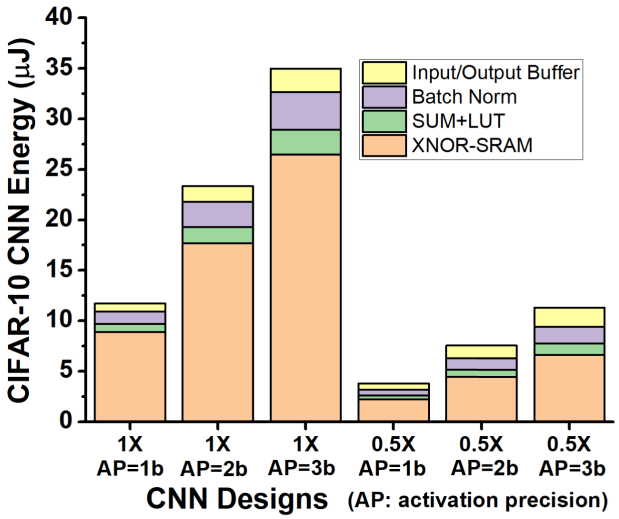


Fig. 13. Energy breakdown of the entire CNN designed for CIFAR-10 data set. Two different sizes of CNNs (1× and 0.5×) and three different activation precision schemes (1–3 bit) are shown.

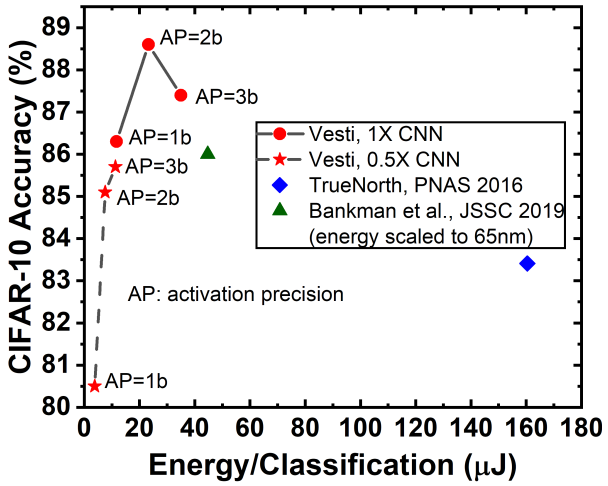


Fig. 14. For CIFAR-10 data set, accuracy versus energy (per classification) comparison for variable activation precision with prior works is shown.

For the MNIST data set, in Fig. 12, we compared Vesti (with 1-/2-/3-bit activation precision) to several prior works [19], [28], [45]–[47] that demonstrated the energy and accuracy for the entire DNN for MNIST data set. For all data points of prior works shown in Fig. 12, we directly grabbed accuracy/energy numbers for MNIST classification from [19], [28], and [45]–[47] without adapting them. "SNN" stands for spiking neural network implementations [45], [47], and "IMC" represents SRAM-based in-memory computing works [28]. It can be seen that the proposed Vesti accelerator results in superior accuracy versus energy tradeoffs compared to the state-of-the-art DNN implementations (MLP, CNN, SNN, and IMC) for MNIST data set.

PROMISE [29], another mixed-signal in-memory computing accelerator, achieved 0.49 μJ/classification for the MNIST data set using the MLP of 784-512-256-128-10 with 8-bit precision, but did not report the classification accuracy (due to the lack of accuracy information, and this work was not included in Fig. 12). Vesti achieves 0.012 μJ/classification for the MLP of 784-256-256-256-10 with 1-bit weight and 3-bit activation precision, which resulted in 98.5% classification accuracy. Considering that similar MLP networks were employed in both works, Vesti achieves ~40× energy improvement.

For CIFAR-10 CNN, we also simulated and evaluated the total CNN energy for various activation precisions of 1–3 bits for two CNN sizes (1× and 0.5×, as described in Section V-B) in Fig. 13. Since activations with N-bit precision consume N cycles to compute using the XNOR-SRAM array, the overall energy roughly increases linearly with the activation precision. To perform the single inference of the CIFAR-10 CNN using the 1-bit activation precision, the Vesti accelerator consumes 1676 cycles. At 0.55-GHz clock frequency, which the Vesti accelerator can operate at, this marks the throughput of 328K inferences per second. Fig. 14 shows the energy across three activation precision values (from 1 to 3 bit) and two CNN sizes (1× and 0.5× CNN). For the CIFAR-10 data set, in Fig. 14, we show the energy and accuracy tradeoffs for variable activation precision using the Vesti accelerator and also compared to the energy and accuracy of the all-digital TrueNorth processor reported in [48] and the mixed-signal binary CNN processor reported in [49] (energy scaled to 65 nm). Regarding the comparison with [49], we used an energy scaling factor of 11.77× from 28 to 65 nm based on the dynamic energy consumption of two commercial register file designs (array size of 256 × 64) at 28 nm (0.81-V supply) and 65 nm (1.0-V supply). As shown in Fig. 14, Vesti accelerator results in superior accuracy and energy tradeoffs for two different CNN sizes (1× and 0.5×) for multiple activation precision schemes for the CIFAR-10 data set. Comparison with prior works is summarized in Table I.

C. Discussion

For binary-weight CNNs, as shown in Fig. 7, 3-bit and higher activation precision schemes have achieved similar accuracy in software simulations. With finer grain activation representation, 3-bit (or higher) activation hardware designs seemingly become more sensitive to a similar amount of quantization error to signal ratio, introduced by 11-level ADCs, compared to coarser-grain 1-/2-bit activations. Given the similar amount of perturbation in XAC values, higher precision

TABLE I
COMPARISON WITH PRIOR WORKS

	[46]	[47]	This Work	[19]	[43]	[44]	[45]	[28]
Technology (nm)	45	28	65	28	45	40	28	65
Model	SNN	Conv	Conv/MLP	MLP	SNN	Conv	SNN	Conv
Voltage Supply (V)	0.6-0.8	0.6-0.8	1	0.6-1.1	0.6-0.8	0.55-1.1	0.9	1
Clock Frequency (MHz)	--	10	550	667	--	204	163	--
CIFAR10 Accuracy (%)	83.41	86.05	88.6	--	--	--	--	--
CIFAR10 Throughput (FPS)	1249	237	328 k	--	--	--	--	--
CIFAR10 Energy/ Prediction (μJ)	163	3.8	23.3	--	--	--	--	--
MNIST Accuracy (%)	--	--	98.5	98.5	99.42	99	98.7	99
MNIST Throughput (FPS)	--	--	8.6 M	--	1 k	13.4 k	91.6 k	70
MNIST Energy/ Prediction (μJ)	--	--	0.012	0.588	108	0.45	0.773	0.45

activation hardware is experiencing more degradation in accuracy, especially for larger networks such as the 1× CNN in Fig. 14. To achieve further accuracy improvement for 3-bit and higher activation precision schemes, we postulate that ADCs with more levels (higher precision) will be necessary.

In addition, the degraded accuracies for 3-bit activation scheme of 1× CNN in Fig. 14 seem to be partial due to the fact that ReLU activation generates a relatively large portion of zero activations for 3-bit or higher activation precisions, which leads to overall small accumulation values and concentrates the bitcount values near zero, in turn becoming more susceptible to variability [see Fig. 4(d)]. Reducing the variability effect is a crucial concern for in-memory computing designs involving analog computing.

With substantial reduction in on-chip DNN energy, off-chip DRAM access could be a larger portion of the chip-level energy. This requires an in-depth analysis and potentially needs to exploit some of the recent DRAM energy improvement techniques, such as [50].

VII. CONCLUSION

In-memory computing for DNNs and CNNs has been recently gaining significant attention. This is because memory access is the main bottleneck to scaling delay and energy dissipation of the MAC operation in digital DNN/CNN accelerators. Most in-SRAM computing works that turn on all rows or columns simultaneously [23], [26], [31] have only implemented a single custom SRAM array, which only computes a small portion of total MAC operations in DNNs. In such prior works, the rest of the operations and overall architecture to implement the DNN accelerator have not been shown or implemented in hardware.

In this article, we substantially expanded the single-array-level prior XNOR-SRAM work [31] toward a configurable DNN accelerator architecture that integrates 72 XNOR-SRAM arrays. The proposed Vesti architecture features: 1) methodologies to efficiently load/map weights onto such XNOR-SRAM arrays for convolutional layers and fully connected layers of DNNs; 2) multibit activation memory storage and control; 3) double-buffering technique to hide the latencies of reprogramming in-memory computing SRAM arrays; and 4) interarray communication.

Due to these comprehensive designs, Vesti simultaneously achieves both high accuracy and low energy for representative DNNs that are benchmarked for MNIST and CIFAR-10 data sets. The Vesti accelerator presented in this article features essential techniques for in-SRAM computing-based deep learning processors, which can fit under the stringent power/energy envelopes of mobile, wearable, and IoT devices.

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] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2014, pp. 1701–1708.

[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. P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2015, pp. 3123–3131.

[9] 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.

[10] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*. [Online]. Available: http://arxiv.org/abs/1605.07146

[11] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–14.

[12] 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

[13] S. Zhou, Z. Ni, X. Zhou, H. Wen, Y. Wu, 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

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

[15] 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.

[16] 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.

[17] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "Envision: A 0.26-to-10 TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28 nm FDSOI," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2017, pp. 246–247.

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

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

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

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

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

[23] 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 Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 488–490.

[24] 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 Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018.

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

[26] 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 Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 496–498.

[27] 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.

[28] 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.

[29] P. Srivastava *et al.*, "PROMISE: An end-to-end design of a programmable mixed-signal accelerator for machine-learning algorithms," in *Proc. IEEE Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 43–56.

[30] C. Eckert *et al.*, "Neural cache: Bit-serial in-cache acceleration of deep neural networks," in *Proc. IEEE Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 383–396.

[31] 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./Circuits*, Jun. 2018, pp. 173–174.

[32] S. Kim and M. Seok, "Variation-tolerant near-threshold microprocessor design with low-overhead, within-a-cycle *in-situ* error detection and correction technique," *IEEE J. Solid-State Circuits*, to be published.

[33] 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.

[34] T. Guan, X. Zeng, and M. Seok, "Extending memory capacity of neural associative memory based on recursive synaptic bit reuse," in *Proc. Design, Autom., Test Eur. (DATE)*, Mar. 2017, pp. 1603–1606.

[35] 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.

[36] S. Kim *et al.*, "NVM neuromorphic core with 64 k-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, pp. 17.1.1–17.1.4.

[37] P. Chi *et al.*, "PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," in *Proc. 43rd Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 27–39.

[38] F. Parveen, Z. He, S. Angizi, and D. Fan, "HieIM: Highly flexible in-memory computing using STT MRAM," in *Proc. Asia South Pacific Design Autom. Conf.*, Jan. 2018, pp. 361–366.

[39] M. Zabihi, Z. Chowdhury, Z. Zhao, U. R. Karpuzcu, J.-P. Wang, and S. Sapatnekar, "In-memory processing on the spintronic CRAM: From hardware design to application mapping," *IEEE Trans. Comput.*, to be published.

[40] A. Chen and M.-R. Lin, "Variability of resistive switching memories and its impact on crossbar array performance," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, Apr. 2011, pp. MY.7.1–MY.7.4.

[41] 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.

[42] C.-Y. Chen, M. Q. Le, and K. Y. Kim, "A low power 6-bit flash ADC with reference voltage and common-mode calibration," *IEEE J. Solid-State Circuits*, vol. 44, no. 4, pp. 1041–1046, Apr. 2009.

[43] J. C. Sancho and D. J. Kerbyson, "Analysis of double buffering on two different multicore architectures: Quad-core opteron and the cell-BE," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, Apr. 2008, pp. 1–12.

[44] B. Liu, H. Li, Y. Chen, X. Li, Q. Wu, and T. Huang, "Vortex: Variation-aware training for memristor X-bar," in *Proc. 52nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2015, pp. 1–6.

[45] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," in *Proc. Adv. Neural Inf. Process. Syst. 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. 2015, pp. 1117–1125.

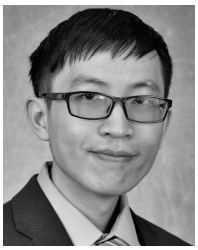
[46] B. Moons and M. Verhelst, "A 0.3–2.6 TOPS/W precision-scalable processor for real-time large-scale ConvNets," in *Proc. IEEE Symp. VLSI Circuits (VLSI-Circuits)*, Jun. 2016, pp. 1–2.

[47] S. Yin *et al.*, "Algorithm and hardware design of discrete-time spiking neural networks based on back propagation with binary activations," in *Proc. IEEE Biomed. Circuits Syst. Conf. (BioCAS)*, Oct. 2017, pp. 1–5.

[48] S. K. Esser *et al.*, "Convolutional networks for fast, energy-efficient neuromorphic computing," *Proc. Nat. Acad. Sci. USA*, vol. 113, no. 41, pp. 11441–11446, 2016.

[49] D. Bankman *et al.*, "An always-on 3.8 μ J/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 158–172, Jan. 2019.

[50] N. Chatterjee *et al.*, "Architecting an energy-efficient DRAM system for GPUs," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2017, pp. 73–84.



Shihui Yin (S'15) 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 working toward the Ph.D. degree in electrical engineering at 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 in 2016.

Zhewei Jiang (S'15) received the dual B.S. degree in physics from Adelphi University, Garden City, NY, USA, and in electrical engineering from Columbia University, New York, NY, USA, in 2013, and the M.S. degree in electrical engineering from Columbia University, in 2015, where he is currently working toward the Ph.D. degree in electrical engineering.

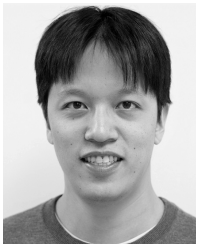
He has been a Research Assistant with the VLSI Lab, Columbia University, since 2015. His current research interests include neuromorphic computing architecture, neural signal processing, in-memory computation for machine learning, and other algorithm implementations.

Minkyu Kim (S'15) received the B.E. degree in electronics and electrical engineering from Pusan National University, Busan, South Korea, in 2005, and the M.S. degree in electrical engineering from Pohang University of Science and Technology, Pohang, South Korea, in 2007. He is currently working toward the Ph.D. degree in electrical engineering at Arizona State University, Tempe, AZ, USA.

From 2007 to 2013, he was with LG Display Co. Ltd., Seoul, South Korea, where he was involved in the development of timing controller chip and image processing for high-quality display. His current research interests include efficient hardware design and application of machine learning and neuromorphic algorithms.

Tushar Gupta received the bachelor's degree in electronics and instrumentation from Kurukshetra University, Kurukshetra, India, in 2009, and the M.S. degree in electrical engineering from Arizona State University, Tempe, AZ, USA, in 2018.

From 2009 to 2016, he was a Research and Development Engineer with the Memory Design Team, Synopsys, Noida, India. From 2017 to 2018, he was a Research Assistant with Arizona State University. He is currently with Synopsys, Mountain View, CA, USA, as a Senior Circuit Design Engineer.



Mingoo Seok (S'05–M'11–SM'18) 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 ultralow-power SoC 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.

Dr. Seok was a recipient of the 1999 Distinguished Undergraduate Scholarship from the 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 2015 NSF CAREER Award, and the 2019 Qualcomm Faculty Award. He was also a recipient of the 2009 AMD/CICC Scholarship Award for picowatt voltage reference work, and the 2009 DAC/ISSCC Design Contest for the 35-pW sensor platform design. He is the Technical Program Committee Members 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, and has been serving as an Associate Editor for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS since 2015 and for IEEE SOLID STATE CIRCUITS LETTER since 2017. He also serves as a Guest Editor for the IEEE JOURNAL OF SOLID STATE CIRCUITS for the 2019 ISSCC Special Issue.

Jae-Sun Seo (S'04–M'10–SM'17) 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 the IBM T. J. Watson Research Center, Yorktown Heights, NY, USA, where he was involved in cognitive computing chips under the 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 the Intel Circuits Research Lab, Hillsboro, OR, USA, 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 from 2004 to 2009, the IBM Outstanding Technical Achievement Award in 2012, and the NSF CAREER Award in 2017.