RRAM-DNN: An RRAM and Model-Compression Empowered All-Weights-On-Chip DNN Accelerator

Ziyun Li, *Member, IEEE*, Zhehong Wang, *Graduate Student Member, IEEE*, Li Xu, *Graduate Student Member, IEEE*, Qing Dong, *Member, IEEE*, Bowen Liu, *Graduate Student Member, IEEE*, Chin-I Su, *Member, IEEE*, Wen-Ting Chu, *Member, IEEE*, George Tsou, *Member, IEEE*, Yu-Der Chih, *Member, IEEE*, Tsung-Yung Jonathan Chang, *Fellow, IEEE*, Dennis Sylvester, *Fellow, IEEE*, Hun-Seok Kim, *Member, IEEE*, and David Blaauw, *Fellow, IEEE*

***Abstract*— This article presents an energy-efficient deep neural network (DNN) accelerator with non-volatile embedded resistive random access memory (RRAM) for mobile machine learning (ML) applications. This DNN accelerator implements weight pruning, non-linear quantization, and Huffman encoding to store all weights on RRAM, enabling single-chip processing for large neural network models without external memory. A four-core parallel and programmable architecture adapts to various neural network configurations with high utilization. We introduce a customized RRAM macro with a dynamic clamping offset-canceling sense amplifier (DCOCSA) that achieves sub-microampere input offset. The on-chip decompression and memory error-resilient scheme enables 16 million (M) 8-bit (decompressed) weights on a single-chip using 24 Mb RRAM. The proposed RRAM-DNN is the first digital DNN accelerator featuring 24 Mb RRAM as all-on-chip weight storage to eliminate energy-consuming off-chip memory accesses. The fabricated design performs the complete inference process of the ResNet-18 model while consuming 127.9 mW power in TSMC-22 nm ULL CMOS. The RRAM-DNN accelerator achieves peak performance of 123 GOPs with 8-bit precision, exhibiting measured energy efficiency of 0.96 TOPs/W.**

***Index Terms*— Deep learning, deep neural network (DNN) ASIC, machine learning (ML) hardware, mobile, model compression, non-volatile memory, resistive random access memory (RRAM).**

I. Introduction

DEEP neural network (DNN) algorithms, first introduced in the early 1960s [1], are the cornerstone of modern artificial intelligence (AI) because they achieve unprecedented accuracy on various computer vision and machine translation tasks. The next wave in the AI revolution is the deployment of these DNNs on mobile platforms to perform challenging tasks under real-world constraints. However, existing hardware and infrastructure cannot provide satisfying performance and energy efficiency for emerging deep-learning-based applications because of their excessive computation and large memory footprints in state-of-the-art DNN models. For object recognition with the ImageNet data set [2], these DNN models [3]–[5] typically comprise more than ten million parameters and require more than 10 GOP per inference, which translates to more than 50 MB on-chip storage and 300 GOPS throughput for real-time 30 frames/s operation. They consume >100 W of power with general-purpose graphics processing units (GPGPUs), which cannot be integrated on mobile platforms due to their excessive power consumption and form factor. Therefore, there is a growing demand for high-performance, energy-efficient, and re-configurable DNN processors for mobile and embedded AI applications [6]–[17].

A. Prior Work and Limitations on ML Algorithms and ASICs

To address these challenges, various approaches using both machine learning (ML) algorithms and efficient hardware designs have been proposed to reduce the complexity of the DNN inference and to improve the energy efficiency, thereby maintaining accuracy for applications.

Iandola *et al.* [6] and Sandler *et al.* [7] propose to re-architect the neural network models and leverage efficient building blocks to reduce both the model size and the number of multiply-and-accumulate (MAC) operations. However, despite the dramatic complexity reduction, these approaches create new DNN layers with novel memory-access and challenging computation requirements that are not well-optimized with existing hardware [17]. Alternative approaches such as [8], [9] reduce the model complexity with pruning, quantization, entropy coding, and/or low-rank approximation of weights. However, their real-time energy-savings and performance gains are limited because of the inefficiency of running unstructured sparse models on the hardware. For example, [18] reports >1 W power consumption for real-time inference using compressed DNNs.

In parallel with improving DNN models, many digital ASICs [10]–[17] were proposed recently to accelerate deep

Manuscript received August 16, 2020; revised November 3, 2020 and December 7, 2020; accepted December 13, 2020. Date of publication December 31, 2020; date of current version March 26, 2021. This article was approved by Guest Editor Yusuke Oike. *(Ziyun Li and Zhehong Wang contributed equally to this work.) (Corresponding author: Zhehong Wang.)*

Ziyun Li, Zhehong Wang, Li Xu, Bowen Liu, Dennis Sylvester, Hun-Seok Kim, and David Blaauw are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, 48109 USA (e-mail: liziyun@umich.edu; zhehongw@umich.edu).

Qing Dong is with TSMC, San Jose, CA 95134 USA.

Chin-I Su, Wen-Ting Chu, George Tsou, Yu-Der Chih, and Tsung-Yung Jonathan Chang are with TSMC, Hsinchu 300-86, Taiwan.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSSC.2020.3045369.

Digital Object Identifier 10.1109/JSSC.2020.3045369

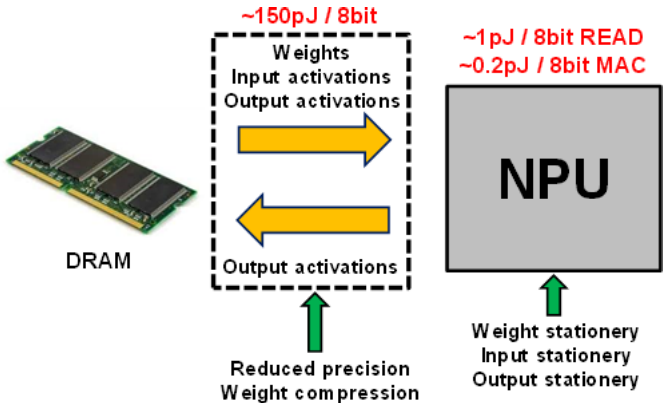


Fig. 1. Conventional system-level dataflow of NPU.

learning on mobile platforms. Various optimization techniques are explored in these designs, including dataflow optimizations [10]–[12], precision reduction [11], [13], [14], sparsity awareness [15], [16], and bit-serial operation [11]. Combining these techniques onto silicon implementations, state-of-the-art DNN processors achieve more than 100 GOPS performance and ∼2 TOPS/W efficiency during inference.

However, as shown in Fig. 1, most of these digital ASICs adopt a DRAM-neural processing unit (NPU) -style processing architecture for loading and computing DNN models [19]. The weights and input activations (IAs) are transferred on chip for processing while computed output activations (OAs) are transferred back to the large off-chip DRAM for temporary storage. While the processing on the NPU is extensively optimized through various techniques [11]–[16] (Fig. 1), transferring data on/off the NPU to the DRAM becomes a major bottleneck in the overall system because of the frequent and extremely high-energy data access to external DRAMs. In fact, transferring a byte from DRAM consumes $>3000\times$ more power than performing an 8-bit MAC calculation [10]. To relieve this problem, [10]–[12] propose to integrate dedicated weight and activation buffers and optimize the dataflow to reduce the data transfer to external DRAMs. Additionally, [14] proposes to leverage data compression technique to reduce the bandwidth to the DRAM. These methods significantly reduce the data access overhead to the DRAM but do not completely solve the problem.

To reduce the off-chip data/parameter accesses, a few prior designs [12], [13] attempt to store all parameters on chip. However, [13] suffers from very limited on-chip memory capacity (only ∼100 kB of weights are stored), which is insufficient to support large applications with >10 M weights. The design [12] achieves high capacity (7.68 MB on-chip weights and 96 MB SRAM stack) at the expense of high system power (3.3 W) due to the large SRAM stack and inductive inter-die communication.

B. Prior Work and Limitations on Non-Volatile Memories

Although embedded Flash memory has been deployed in micro-controllers as non-volatile storage for code and data [20], [22], technology scaling poses a substantial challenge with regard to the use of such charge-based Flash, SRAM, and DRAM [21]. The reduced capacity to hold sufficient charge on the floating gate of Flash memory, the internal capacitive node of SRAM, and the cell capacitor of DRAM degrade the performance, reliability, and noise margin, limiting their applications. As possible solutions, emerging non-charge-based non-volatile memories have been proposed, such as resistive random access memory (RRAM) [22]–[24], MRAM [25], [26], and PCRAM [27]. Among them, RRAM is a promising candidate for wide adoption to ML/DNN applications as it has logic-process compatibility and a large on–off ratio between the high resistance state and low resistance state for potential multi-level operations [24]. Various DNN accelerators employing Computation-In-Memory (CIM) techniques on RRAM have been proposed [28], [29]. However, due to limited computing precision, these CIM accelerators are not readily scalable to high-accuracy DNNs. And to date, there have been few designs that leverage RRAM's higher density and low standby power for all-on-chip parameter storage in large-scale digital DNN accelerators (versus a general-purpose non-volatile microcontroller [23]).

In this article, we present the first digital DNN accelerator featuring 24 Mb RRAM for all-on-chip weight storage to eliminate energy-consuming off-chip weight accesses, thereby reducing the overall system operating power. The design employs a four-processing element (PE) architecture in 22 nm ULL CMOS technology with 24×1 Mb custom-designed embedded RRAM banks. Using pre-compressed DNN models with an on-the-fly weight decompression mechanism, we achieve on average ∼1.5 b/weight for AlexNet, 3.2 b/weight for ResNet-18, resulting in a maximum total capacity of 16 M weights on chip. Highly parallelized and mesh-connected MAC arrays in the PE enable various workload mapping schemes to support DNN layers with different memory and compute characteristics. To reliably read and write to the RRAM, we propose a dynamic clamping offset-canceling sense amplifier (DCOCSA) that achieves sub-microampere input-sensing offset and a Write-Verify scheme for reliable programming. Combined with a mesh-connected MAC array architecture and 8 Mb shared SRAM, the proposed DNN accelerator operates at 120 MHz at 0.8 V digital VDD, achieving 0.96 TOPS/W [30].

The remainder of this article is organized as follows. Section II describes the overall architecture as well as the design details of the RRAM-DNN chip. Section III explains the dataflow and mapping of heterogeneous ML workloads onto the architecture. Section IV describes the compression of the DNN model. Section V explains the circuits of the custom-designed RRAM. Section VI shows the measurement results, and Section VII concludes this article.

II. Overall Architecture

Fig. 2 shows the overall architecture of the RRAM-DNN chip. The design consists of four PEs connected to a shared bus and a global shared memory. Each PE has its local memory for buffering the input–output activations, dedicated 6 Mb RRAM memory banks for non-volatile parameter storage, MAC array units for highly parallelized processing, and instruction memory for controlling the layer functions. In the

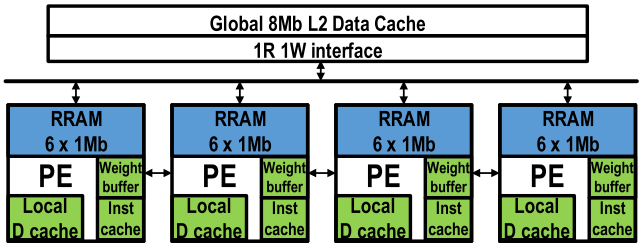


Fig. 2. Overall architecture of the RRAM-DNN chip.

architecture, each PE has both read/write access to its own local memory as well as read access to its neighboring PEs' local memories. The global shared memory is 8 Mb, and it supports parallel write and read access if the accesses are pre-partitioned to different memory banks. Due to the large chip size and the heterogeneous memory hierarchy, different memories in the architecture have different access latencies. The local memories including the input and weight buffers achieve 1 cycle access latency. Accessing neighboring PE's memories and the global memory incur access latencies of 2 cycles and 4 cycles, respectively. Moreover, the shared global memory coalesces multiple accesses by broadcasting data to all or a subset of four PEs when their read addresses are identical. In simulation, broadcasting data to coalesced requests results in $\sim 4\times$ latency reduction when multiple PEs are fetching the same IA from the global shared memory.

During the execution of a layer function, a PE first loads a block of IAs from the global shared memory to its local memory following user-defined memory partitioning. The PE's neighbors can share its IA because of the local connectivity between PEs. The PE then processes the layer function on the block of inputs with local stored weights. After all OAs are computed, the PE moves the output block back to the shared global memory. Each PE may process different data and execute different instructions, which can lead to a variable processing latency. Therefore, synchronization is necessary to ensure correct layer operations when the PEs are collaborating. The proposed design can be programmed to synchronize all or a subset of four PEs.

A. Detailed Architecture of the PE

Fig. 3 details the design of a single PE. Inspired by Li *et al.* [31], the PE architecture exploits parallelism and data reusability across different input dimensions to improve energy efficiency. Each PE has a mesh of 128 8-bit multiply/32-bit accumulate MAC units in four clusters (each with a grid of 4×8 MAC units). Each MAC also contains 32-bit flipflops to locally store processed partial sums. In total, four PEs have 512 MAC units on chip, enabling massive parallel processing for compute-intensive CNN operations. Moreover, each PE processes four input channels (IC), four output channels (OC), and eight IAs in parallel to maximize the data reusability in the MAC array. Each PE has its own private 6 Mb RRAM for parameter storage. During the CNN operation, weights are first read from the RRAM, decompressed through the decompression engine, and transferred to small 2-bank, 4-kB interleaved weight buffers for frequent local accesses.

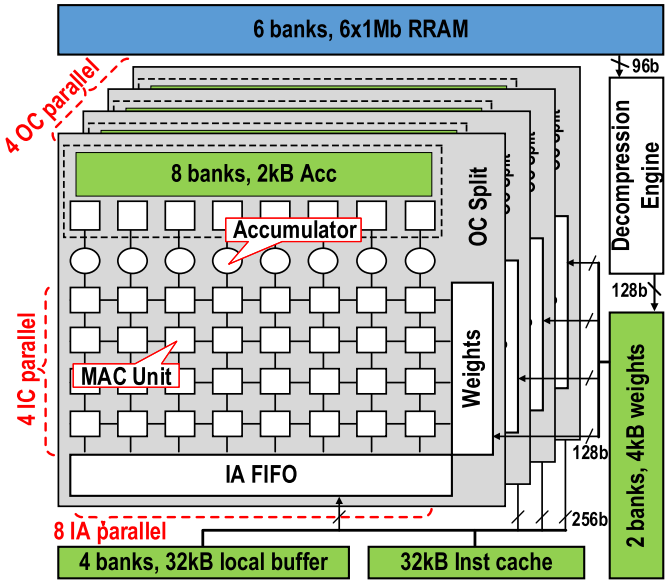


Fig. 3. Detailed architecture of the PE.

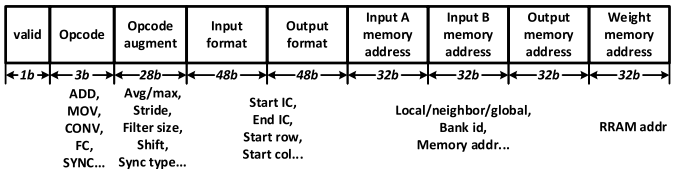


Fig. 4. ISA of the proposed RRAM-DNN accelerator.

The MAC array processing and weight decompression occur concurrently (pipelined) to maximize throughput. Accessing the small 4 KB weight memory provides 128 bit/cycle memory access bandwidth with high access energy efficiency. The 4-bank, 32 kB local buffer stores input and output activation with 256 bit/cycle access bandwidth. The high data bandwidth from both the weight buffer and local buffers ensures the full utilization of the 128 MAC units.

B. Instruction Set Architecture

To control the processing of MAC units for hundreds of cycles without explicit instruction decoding in each cycle, 256-bit Very long Instruction Word (VLIW) instructions are used. Moreover, the instructions are stored in the 32 kB instruction memory of each PE so that it can be programmed independently to control the processing sequence and synchronization of the DNN algorithm if necessary. Offset (direct) addressing with respect to each PE's own base address is used in the instruction set architecture (ISA) for arithmetic operations within a PE, including CONV, ADD, and POOL, to reduce the bit-width of the instructions. Non-offset global direct addressing is used when the data are moved from/to the global memory. Fig. 4 details the ISA of the proposed RRAM-DNN processor. The proposed ISA supports not only various layer functions such as convolution, pooling, matrix multiplication, and ReLU, but also flexible layer partition schemes such as the number of split input and OC. Data concatenation and scaling can also be achieved through MOV (move), ADD (addition) instructions.

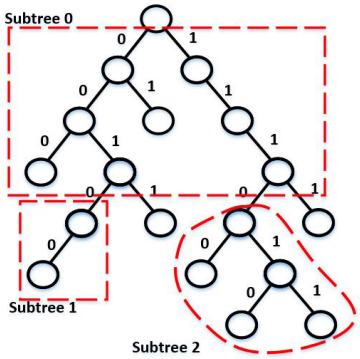


Fig. 5. Parallel Huffman decoder using full subtrees.

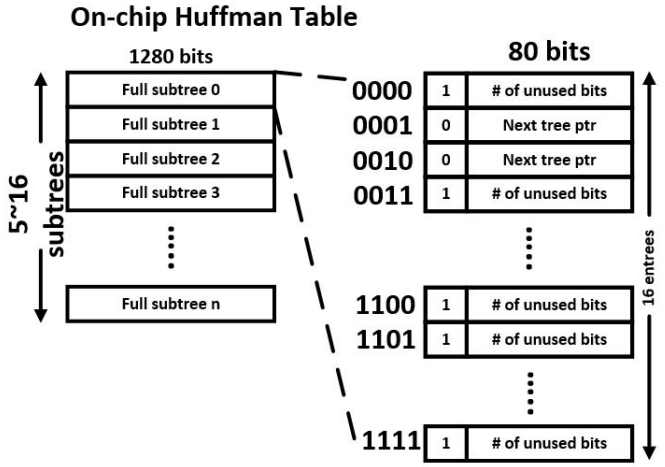


Fig. 6. On-chip Huffman table for decompression.

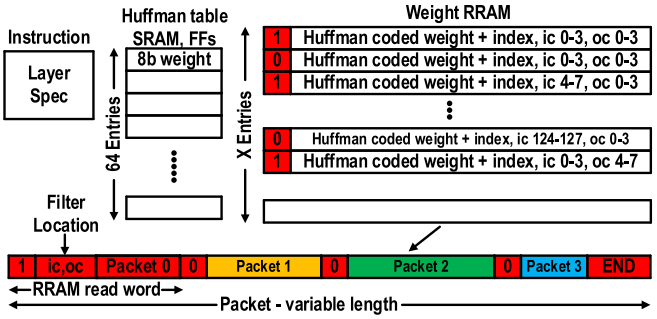


Fig. 7. On-chip compressed weight storage in RRAM.

C. Decompression Engine

The weight compression algorithm is adopted from [8]. During the training, unimportant weights are pruned to zero and all non-zero weights are non-uniformly quantized to 64 levels. To compress each weight, we use the Huffman encoded weight value (one of 64 levels) as well as the run length of the non-zero weight position. This algorithm compresses each weight to bit on average with negligible accuracy degradation for ResNet-18 [5]. Each PE is equipped with a decompression engine to decode the compressed weights stored in the RRAM. Each decompression engine contains two programmable Huffman tables: one for weight values and the other for run-length positions. These tables share a parallel lookup table (LUT)-based decoder. Decompressing Huffman encoded weight values and run-length positions to meet the processing bandwidth of the PE is challenging. On the one hand, decompressing the Huffman encoded 96-bit in a single cycle requires a logic with very long critical paths (>10 ns) due to inter-bit dependence in the compressed bit sequence. On the other hand, if the Huffman decoding was performed in series with single bit per cycle throughput, an entire weight packet would cost $>$ 250 cycles to process. Decompression throughput needs to be balanced with the throughput of the MAC array which takes 72 cycles for processing eight rows of 3×3 kernel. Therefore, instead of traversing a binary Huffman tree sequentially by advancing a single bit per clock cycle, we decode 4 bits in parallel to improve the performance (Fig. 5) per cycle. This requires storing all possible 4-bit subtrees (Fig. 6), which are stored in each PE and programmed through the PE programming interface. The critical path of decompressing 4 bits in parallel is 3 ns. Note that the layer-dependent nonuniform weight quantization and pruning requires reprogramming of these Huffman tables/trees for each DNN layer. We minimize the programming overhead by programming multiple PEs simultaneously when they share the same table.

D. RRAM Weight Storage and Static Error Resiliency

The compressed weights for convolutional layers are stored in the RRAM as packets shown in Fig. 7. Each packet has a variable length (because each weight length is variable) and is split into multiple RRAM words. Each packet contains a layer specification, Huffman coded weight values, and run-length codes for 4 input and 4 OC. The layer specification consists of the kernel offset and location for weights. We insert this specification information for every packet to make the system resilient to RRAM word errors. Since each weight and packet has variable length, a single RRAM word error can cause catastrophic decompression failure for subsequent packets. The proposed packet specification enables faulty word mitigation by repeating the same packet (including the specification) twice if the first packet was written on a faulty RRAM word(s). In that case, the second packet overwrites the first faulty packet during the decompression process. We assume RRAM word error locations are static and identifiable before programming the chip.

III. Dataflow of the Proposed RRAM-DNN Core

The proposed architecture and ISA support flexible mapping of heterogeneous DNNs for efficient hardware execution. This section discusses the various energy-efficient dataflows that are supported in the proposed architecture. The evaluations of different dataflows are performed with a python-based cycle accurate simulator, modeling the behavior of the designed four-PE system. The simulator pre-allocates weights and activations onto the PEs and computes corresponding memory addresses based on a given partitioning scheme. Then, the simulator profiles the chip behavior/execution trace for evaluation/verification, and also generates VLIW instructions (Fig. 4) to control the chip.

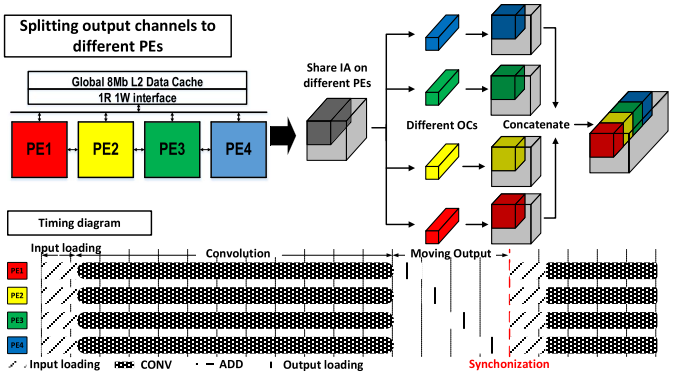


Fig. 8. Split convolution onto multiple PEs by OC.

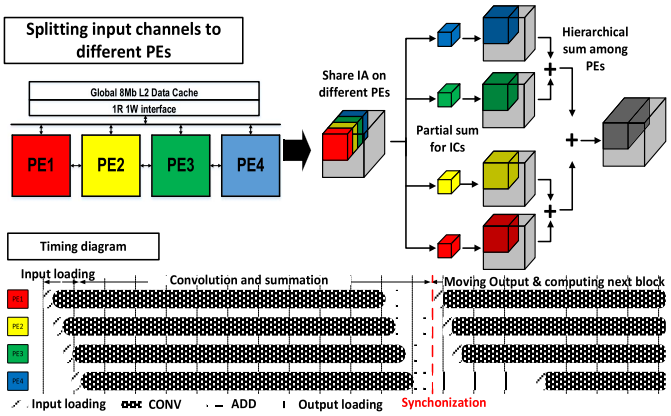


Fig. 9. Split convolution onto multiple PEs by IC.

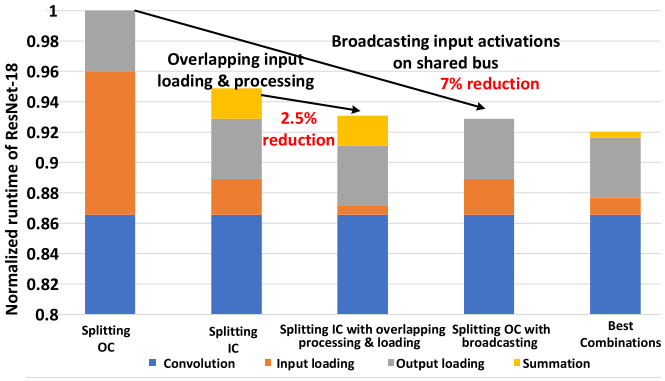


Fig. 10. Optimizing processing latency with multiple PEs (simulation).

A. Partition Workload Onto PEs by OC

One example of mapping a DNN layer to the architecture is shown in Fig. 8. The colors in Fig. 8 indicate weight/kernel mapping of a convolutional layer to the architecture, where the weights are split by different OC mapped on dedicated PEs. In this example, the weights are pre-partitioned on these four PEs, and each PE is programmed to compute different OC through instructions. Meanwhile, the IAs are partitioned into 8×8 blocks, with all associated IC, for processing to match the local memory capacity in each PE. When the processing of an 8×8 block for all IC finishes, the PE re-organizes the output and moves it back to the global memory. The outputs from multiple PEs are concatenated in this process to form the complete layer output. The timing diagram of the process is shown in Fig. 8 (bottom). Although each PE stores only 1/4 of the total weights and also processes only 1/4 of the convolutions, the same complete IAs from the prior layer must be copied to the local memories of each PE. To minimize this potentially redundant traffic and save data transfer time, we enable the bus to broadcast IA to all PEs. In simulation, the combination of IA broadcasting and global memory access coalescing improves the MAC utilization and reduces the inference latency by 7% (Fig. 10).

B. Partition Workload Onto PEs by IC

Another possible mapping of a convolutional layer to different PEs is input channel-based partitioning. In that case, each PE processes a partial sum of different IC as shown in Fig. 9. A selected PE merges the results from neighboring PEs hierarchically and then writes the merged results to the global memory (Fig. 9, bottom). Thanks to the local connection between neighboring PEs, no extra data movement is needed as each PE has read access to each neighboring PE's local memory. Similar to the output channel split mapping in Section III-A, weights are pre-partitioned on these four PEs and IAs are partitioned into 8×8 blocks (and 4 of all IC) for processing. Compared with splitting the OC, this input channel partitioning scheme involves uneven workload distribution among the PEs because a selected PE(s) needs to perform the extra merge and move operations. This can potentially lead to idle cycles and low MAC utilizations for the other PEs. However, the workload can be balanced throughout the entire convolution layer if the merge and move operations are mapped onto all PEs in a round-robin fashion. Because the IAs are partitioned into $8 \times 8 \times 4$ (IC) blocks per PEs, and typically there are >64 blocks in a layer, the workload can be balanced for the overall layer. This workload balancing scheme improves the overall MAC array utilization by 2.5%. Fig. 10 summarizes the different workload partitioning methods and their impacts on MAC utilization with ResNet-18.

Each individual layer in the network can be separately programmed for the best workload partitioning based on the layer characteristic. For the ResNet-18 example shown, 8% overall latency reduction can be achieved with layer-dependent best combinations of aforementioned partitioning schemes compared to a naïve approach (Fig. 10).

C. Data Reuse for Efficient Convolution Processing

Convolution operations on a single PE are optimized with massive parallelism and data-reuse. Similar to the CNN design in [31], each PE consists of four clusters of 8×4 MAC arrays, processing eight consecutive pixels and four consecutive IC in parallel. Partial convolution is performed with shifting IAs using a row of 8 MAC units with k (kernel size) cycles. This operation is repeated on the second and third row of IAs to complete the 2-D convolution. The pooling operations are performed in the same fashion except that MAC is replaced by Max in the 128 MAC units. Partial products in 2-D convolution are accumulated locally in each MAC unit to improve the energy efficiency without unnecessary memory accesses.

TABLE I
EXAMPLE OF APPLY COMPRESSION SCHEME ON RESNET18

	Total number non-zeros parameters (TNZ)	Weight bitwidth (w)	Index bitwidth (r)	Top-1 Error	Top-5 Error	Compression rate
Baseline	Conv layers: 10.99 million FC layers: 0.513 million	FP32	-	28.22%	9.42%	-
Pruning	Conv layers: 4.565 million FC layers: 0.094 million	FP32	-	28.20%	9.42%	-
Pruning + Quantization	Conv layers: 4.565 million FC layers: 0.094 million	INT8	-	28.17%	9.40%	-
Pruning + Quantization + Runlength coding	Conv layers: 4.565 million FC layers: 0.094 million	INT8	INT5	28.17%	9.40%	6.08x
Pruning + Quantization + Runlength coding + huffman coding	Conv layers: 4.565 million FC layers: 0.094 million	5.5	2.4	28.17%	9.40%	10.01x

$$Bit/weight = \frac{(r+w)*TNZ\text{-}conv + (r+w)*TNZ\text{-}fc}{TNZ\text{-}uncompressed}$$

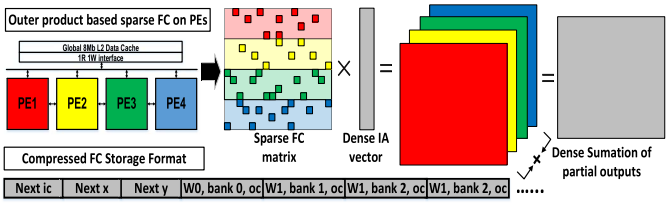


Fig. 11. Sparse FC operation of the RRAM-DNN design.

D. Data Reuse for Efficient Processing of Sparse Fully Connected Layers

The compressed weights for the FCL are very sparse, with typically less than 20% density [8] (Table I), whereas the IAs for the fully connected layers (FCL) are densely populated. Inspired by Subhankar *et al.*, [32], we deploy outer-product-based matrix vector multiplication to efficiently compute the FCL and skip all zero multiplications. Similar to the convolution operation, sparse FCL weights are stored in the compressed format and are pre-partitioned onto each PE during the compile time to enable highly parallelized processing with multiple PEs (Fig. 11). Different from convolution layers, weights for FCL are only pruned and quantized without Huffman coding to increase the decompression rate to match the throughput of parallelized processing without weight reuse. During processing, each element of IAs is multiplied with sparse non-zero weights from RRAM in each PE. Partial outputs are then accumulated and stored in the accumulators depending on the location of the non-zero weights. As multiple PEs finish processing subsets of an FCL, selected PEs merge the FCL output hierarchically and write the results to the global memory.

IV. COMPRESSED MODEL FOR SIMULATION AND MEASUREMENTS

To enable the single-chip implementation for DNN models, we leverage an idea from a state-of-the-art deep compression scheme [8] for compressing DNN models. However, the compression scheme also has to be co-designed to maximize the performance and efficiency of the architecture. The convolution layers typically require less bandwidth to decompress weights because each weight can be reused over multiple cycles for different input–OAs. Therefore, the weights for

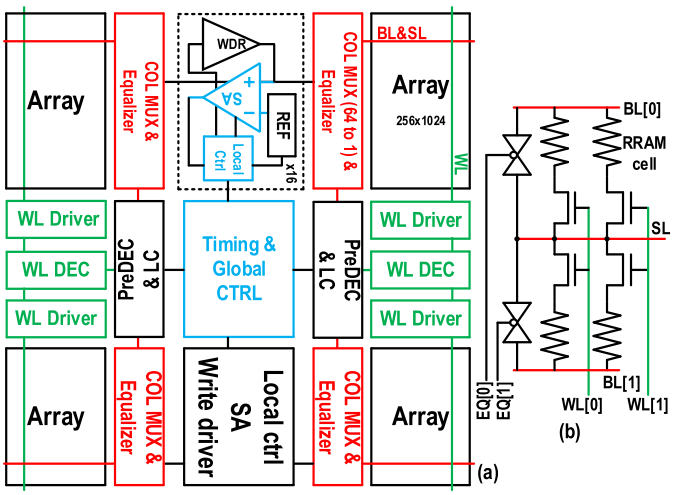


Fig. 12. (a) RRAM bank architecture (b) Common SL arrangement.

convolution layers are pruned, non-linearly quantized with 64 weight centroids and run-length coded using 5-bit codes to achieve maximum compression. On the other hand, weights are only used once for FCL. Thus, it is necessary to simplify the compression scheme to balance the weight decompression throughput with the FCL computation throughput. The weights for FCL in our design are pruned without any entropy coding. On average, the proposed compression scheme achieves ~5.5 bit per weight in convolution layers and 2.4 bit per weight in FC layers. Table I shows an example of applying the compression on ResNet18 when trained and evaluated under the ImageNet data set [2]. Pruning the weights reduces the model size of the convolution layers by 68% and FCL by 82%. Run-length coding and Huffman coding further compress the pruned convolution layers by 40% (from 8-bit weight and 5-bit run-length to 5.5-bit weight and 2.4-bit run-length for non-zero weights). With both methods combined, the average bits/weight is 3.2.

After the proposed weight compression, the DNN model exhibits negligible accuracy degradation compared with 8-bit uncompressed weights under ImageNet [2] evaluation.

V. CUSTOMIZED RRAM MEMORY

The 1 Mb custom-designed RRAM bank uses a butterfly architecture, as shown in Fig. 12(a), which is composed of four 256 × 1024 RRAM arrays with 32 b word length. The three colors denote the three main power domains used in the bank for testing flexibility: 1.4 V for the word line (WL, green), 1.25 V for the column mux (red), and 1 V for the sense amplifier (SA) and ctrl (blue). The RRAM array employs a common source line (SL) cell arrangement [33]. Thus, the column-wise peripherals include an equalizer [Fig. 12(b)] to virtually short the half-selected column when the other one is being written.

A. Dynamic Clamping Offset-Canceling SA

RRAM typically suffers from high variation in cell resistance which can vary by 2~10× for the low resistance state and 5~100× for the high resistance state [34], leaving a small sensing margin on sensing circuits. To address the high

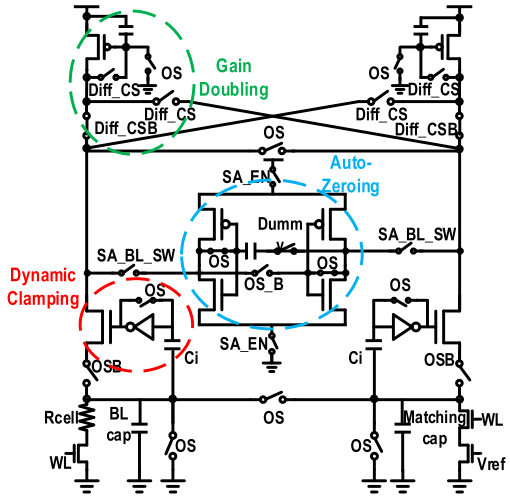


Fig. 13. Proposed DCOCSA.

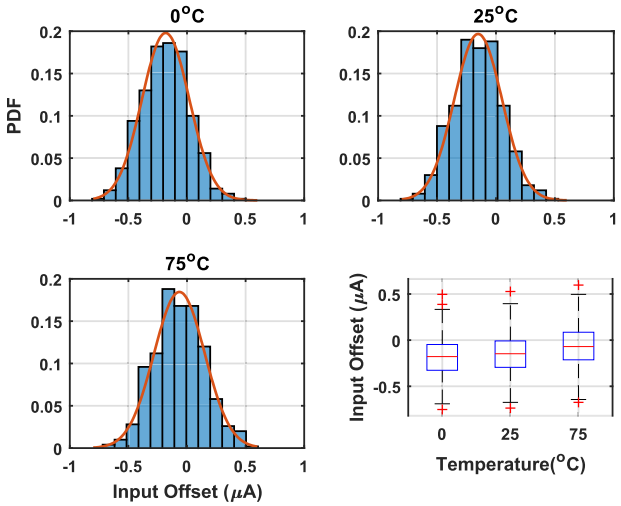


Fig. 14. DCOCSA input current offset MC simulation distribution.

variation nature of the RRAM, the two-stage offset-canceling current-mode SA shown in Fig. 13 is proposed. The first stage is composed of two cross-coupled current sampling branches similar to the scheme in [35], which doubles the input current difference and effectively halves the offset. In addition, the first stage incorporates dynamic clamping, instead of typical static clamping, to bring down the bitline settling time and increase the sensing speed. Unlike conventional clamping amplifiers, which are large and power hungry, a carefully designed self-biased inverter provides the feedback loop. The settling time is reduced by 50% (simulation) compared to a static clamping SA under the same load. The second stage provides further amplification and offset-reduction with a single-cap auto-zero regenerative amplifier [26].

Fig. 15(a) details the operation of the proposed SA. In step ①, the input and output of the inverter are shorted to self-bias the clamp transistors, with bias voltage sampled on the Ci's. Meanwhile, the regenerative amplifier of the second stage is also shorted to sample the offset and cancel it out in the following steps. This step overlaps with address decoding to avoid a timing penalty. Then, in step ②, the shorted inverter in phase one is disconnected to function as a negative feedback amplifier, and the WL is turned on to allow the two diode-connected PMOS headers to sample the currents I_{ref} and I_{cell} on their respective branches. After the current settles, in step ③, the two headers are switched to the other branch and function as a current source, which generates current difference $I_{\text{cell}} - I_{\text{ref}}$ on both input nodes of the second stage. However, note that the directions of the two current differences are opposite, which effectively doubles the current difference of the input to the second stage to $2(I_{\text{cell}} - I_{\text{ref}})$. Finally, in step ④, the second stage is fired and latches the output. The voltage waveforms of the internal nodes are shown in Fig. 15(b). A sub-microampere current offset is achieved at 21 μA common mode input under 1.2 V VDD from Monte Carlo (MC) simulation with 500 samples; Fig. 14 gives the offset distribution at different temperatures.

B. Write-Verify Process

RRAM also suffers from variation in write time. At a fixed write voltage, the write time of slow cells and fast cells can differ by more than 100× [36]. Thus, applying a write pulse of the same length to both fast and slow cells causes unnecessary power and endurance losses on the fast cell. So, a fine-grained iterative Write-Verify control is adopted. Each bit in a word is separately controlled based on the read result, ruling out correlation between fast and slow cells, which alleviates locality-dependent variation. Furthermore, with Write-Verify, each cell automatically adapts to the corresponding SA offset, further reducing the locality dependence.

Fig. 15(c) illustrates the block diagram of the Write-Verify control. Following a write request, each RRAM cell of the target address is read out first to compare with the input data (DG) initiated by the global control. If the read-out value (d) of a cell is the same as the corresponding bit in DG, the write process of that cell concludes for better endurance. On the other hand, the cell is programmed to the desired value by the iterative Write-Verify process.

VI. Measurement

We implement the proposed accelerator in 22 nm ULL CMOS technology with each PE of size 1614 × 1394 um^2 and each 1 Mb RRAM bank of size 235 × 514 um^2 as shown in the die photograph [Fig. 16 (a)]. The test chip achieves 120 MHz core clock frequency at 0.8 V VDD and consumes 42.4 mW when evaluating a CNN layer of size $4 \times 3 \times 3 \times 16$ as depicted in the measured power/frequency versus VDD plot [Fig. 16(b)] for core digital logic, which is everything except the RRAM banks.

In the implementation, the RRAM clock is hard coded to be half of the core clock; the RRAM operates at 60 MHz for the 120 MHz core frequency. A power breakdown of the four RRAM power domains is shown in Fig. 17(a), with 1V for the SA and control, 1.4 V for the WL, 1.25 V for the column mux, and 1.1 V for the inverter amplifier, noting that this breakdown includes the effect of the possible static errors in RRAM. A measured RRAM resistance distribution across ∼10 k cells randomly sampled from the 24 banks in one test chip at room temperature is shown in Fig. 17(b). The proposed

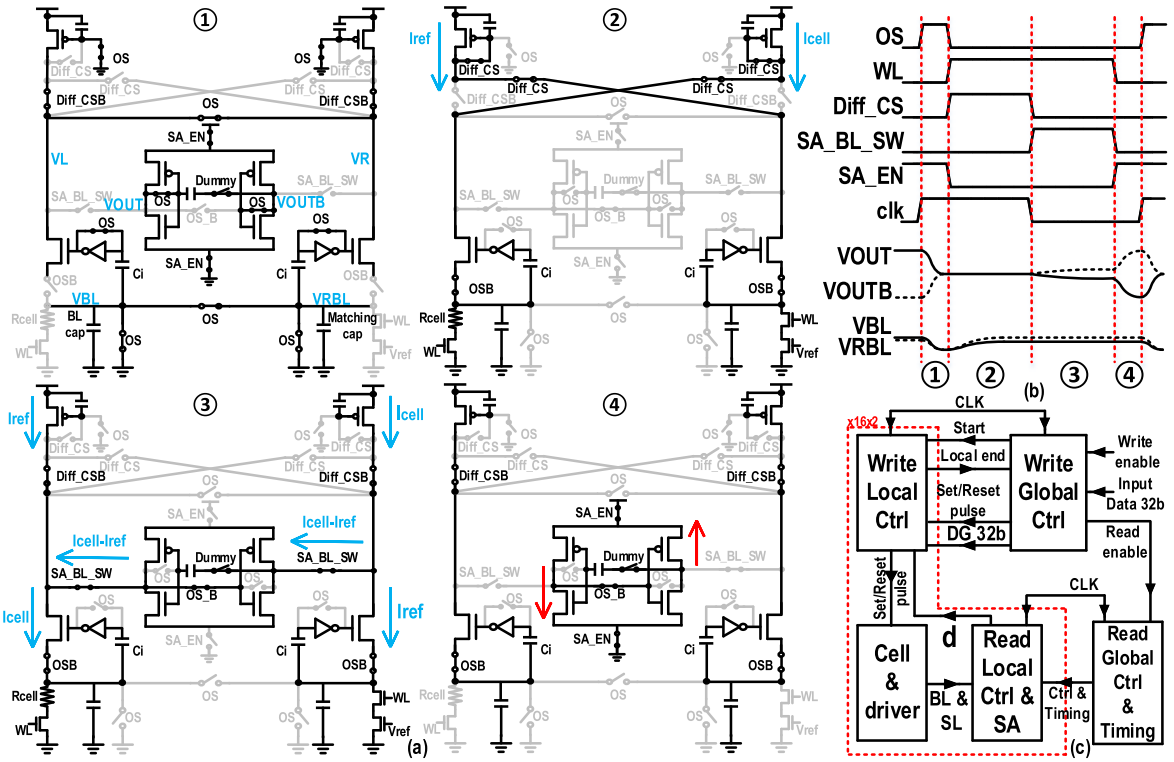


Fig. 15. (a) Operation and (b) timing of DCOCSA and (c) Write-Verify Signal Flow.

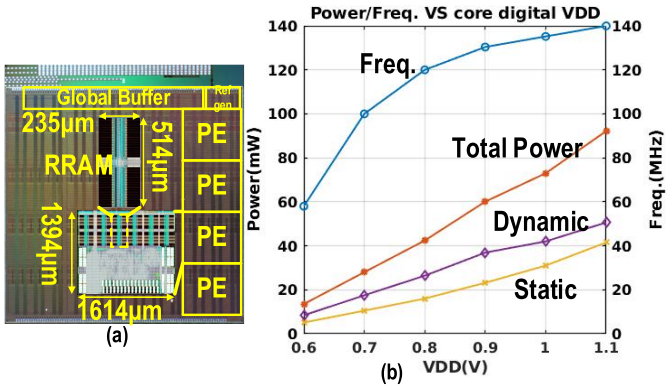


Fig. 16. (a) Die Photograph and (b) power/Frequency versus VDD for core digital logic.

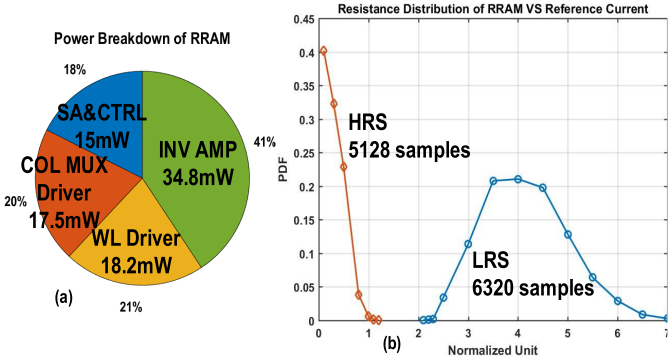


Fig. 17. (a) RRAM power breakdown and (b) measured RRAM resistance distribution.

accelerator consumes 127.9 mW in total, including weight decompression and transfer from RRAM to SRAM, resulting in a power efficiency of 0.96 TOPS/W. Table II compares the work to recent NN accelerators. The proposed design achieves the highest number of on-chip stored weights due to the model compression and better density of RRAM and is also the only design employing non-volatile memory as dedicated weight storage, thereby reducing standby power for edge devices.

TABLE II
COMPARISON TO OTHER WORKS

	This Work	QUEST[12]	SNAP[37]	STICKER[16]	UNPU[11]	Envision[38]
Technology	**ULL 22nm**	40nm	16nm	65nm	65nm	28FD-SOI
On-chip RAM(B)	**3M RRAM 1.3M SRAM**	7.68M+96M 3D SRAM	280.6K	170K	256K	148K
Max On-chip Weight	**16M@8b Non-Volatile**	15.36M@ 4b Volatile	140.3K@16b Volatile	170K@8b Volatile	256K@8b Volatile	148K@8b Volatile
Off-chip Memory	**No**	Yes	Yes	Yes	Yes	Yes
MACs	**4x128 (8x8b)**	24x512 (1x1b log)	252 (16x16b)	256 (8x8b)	4x576 (1x16b)	256 (8x8b)
Voltage (V)	**1.0-1.2 RRAM 0.6-1.1 Core**	1.1	0.55-0.8	0.67-1.0	0.63-1.1	1.05
Freq. (MHz)	**60 RRAM 120 Core**	300	33-480	20-200	200	200
TOPS/W	***0.96@8b**	†0.59@4b	‡3.61@16b	‡1.038@8b	‡5.57@8b	‡1@8b
GOPS	**123@8b**	1960@4b	65.52@16b	102@8b	690@8b	102@8b
Power (mW)	**127.9 @120MHz**	3300 @300MHz	364 @480MHz	284.4 @200MHz	297 @200MHz	44 @200MHz
Chip Area (mm^2)	**10.8**	122	2.4	12	16	1.87

* Including power of loading weights from RRAM to SRAM and MAC
† Including power of loading weights from 3D SRAM to on-chip SRAM & MAC
‡ Excluding power of loading weights from off-chip memory

VII. CONCLUSION

In summary, we present the first energy-efficient digital DNN accelerator featuring RRAM for dedicated weight

storage to enable efficient single-chip inference of NN models for mobile devices. Using on-the-fly weight decompression, we achieve a total capacity of 16 M 8 bit weights on chip. To reliably read from and write to the RRAM, we propose a DCOCSA achieving sub-microampere input-sensing offset. Together, these techniques help us eliminate fully off-chip weight access. The proposed processor is prototyped and measured in TSMC 22 nm ULL with RRAM technology. This design supports single-chip NN model inference with ∼16 million parameters. It achieves 123 GOPs throughput in real-time, consuming 127.9 mW from a 0.8 V supply, with measured 0.96 TOPS/W efficiency. The proposed design achieves the highest number of on-chip-stored weights and is also the only design employing non-volatile memory as dedicated weight storage, reducing standby power for edge devices.

REFERENCES

[1] F. Rosenblatt, "The perceptron—A perceiving and recognizing automaton," Cornell Aeronaut. Lab., Buffalo, NY, USA, Tech. Rep. 85-460-1, 1957.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Miami, FL, USA, Jun. 2009, pp. 248–255, doi: 10.1109/CVPR.2009.5206848.

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

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[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] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size," 2016, *arXiv:1602.07360*. [Online]. Available: http://arxiv.org/abs/1602.07360

[7] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[8] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*. [Online]. Available: http://arxiv.org/abs/1510.00149

[9] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1269–1277.

[10] 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, doi: 10.1109/JSSC.2016.2616357.

[11] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: A 50.6TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2018, pp. 218–220, doi: 10.1109/ISSCC.2018.8310262.

[12] K. Ueyoshi *et al.*, "QUEST: A 7.49TOPS multi-purpose log-quantized DNN inference engine stacked on 96 MB 3D SRAM using inductive-coupling technology in 40 nm CMOS," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2018, pp. 216–218, doi: 10.1109/ISSCC.2018.8310261.

[13] S. Bang *et al.*, "14.7 A 288 μW programmable deep-learning processor with 270 KB on-chip weight storage using non-uniform memory hierarchy for mobile intelligence," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2017, pp. 250–251, doi: 10.1109/ISSCC.2017.7870355.

[14] 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*, Honolulu, HI, USA, Jun. 2016, pp. 1–2, doi: 10.1109/VLSIC.2016.7573525.

[15] P. N. Whatmough, S. K. Lee, H. Lee, S. Rama, D. Brooks, and G.-Y. Wei, "14.3 A 28 nm SoC with a 1.2 GHz 568nJ/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*, San Francisco, CA, USA, Feb. 2017, pp. 242–243, doi: 10.1109/ISSCC.2017.7870351.

[16] Z. Yuan *et al.*, "Sticker: A 0.41-62.1 TOPS/W 8bit neural network processor with multi-sparsity compatible convolution arrays and online tuning acceleration for fully connected layers," in *Proc. IEEE Symp. VLSI Circuits*, Honolulu, HI, USA, Jun. 2018, pp. 33–34, doi: 10.1109/VLSIC.2018.8502404.

[17] Y.-H. Chen, T.-J. Yang, J. S. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 2, pp. 292–308, Jun. 2019, doi: 10.1109/JETCAS.2019.2910232.

[18] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," 2015, *arXiv:1511.06530*. [Online]. Available: http://arxiv.org/abs/1511.06530

[19] Z. Li, "Energy-efficient, mobile computer vision and machine learning processors," Ph.D. dissertation, Deepblue, Univ. Michigan, Ann Arbor, MI, USA, 2019. [Online]. Available: http://hdl.handle.net/2027.42/151423

[20] M.-F. Chang *et al.*, "An asymmetric-voltage-biased current-mode sensing scheme for fast-read embedded flash macros," *IEEE J. Solid-State Circuits*, vol. 50, no. 9, pp. 2188–2198, Sep. 2015, doi: 10.1109/JSSC.2015.2424972.

[21] S. Yu and P.-Y. Chen, "Emerging memory technologies: Recent trends and prospects," *IEEE Solid State Circuits Mag.*, vol. 8, no. 2, pp. 43–56, Spring 2016, doi: 10.1109/MSSC.2016.2546199.

[22] C.-P. Lo *et al.*, "A ReRAM macro using dynamic trip-point-mismatch sampling current-mode sense amplifier and low-DC voltage-mode write-termination scheme against resistance and write-delay variation," *IEEE J. Solid-State Circuits*, vol. 54, no. 2, pp. 584–595, Feb. 2019, doi: 10.1109/JSSC.2018.2873588.

[23] T. F. Wu *et al.*, "14.3 A 43pJ/cycle non-volatile microcontroller with 4.7 μs shutdown/wake-up integrating 2.3-bit/cell resistive RAM and resilience techniques," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2019, pp. 226–228, doi: 10.1109/ISSCC.2019.8662402.

[24] S. R. Lee *et al.*, "Multi-level switching of triple-layered TaO_x RRAM with excellent reliability for storage class memory," in *Proc. Symp. VLSI Technol. (VLSIT)*, Honolulu, HI, USA, Jun. 2012, pp. 71–72, doi: 10.1109/VLSIT.2012.6242466.

[25] L. Wei *et al.*, "13.3 A 7 Mb STT-MRAM in 22FFL FinFET technology with 4ns read sensing time at 0.9 V using write-verify-write scheme and offset-cancellation sensing technique," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2019, pp. 214–216, doi: 10.1109/ISSCC.2019.8662444.

[26] Q. Dong *et al.*, "A 1-Mb 28-nm 1T1MTJ STT-MRAM with single-cap offset-cancelled sense amplifier and *in situ* self-write-termination," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 231–239, Jan. 2019, doi: 10.1109/JSSC.2018.2872584.

[27] G. De Sandre *et al.*, "A 90 nm 4 Mb embedded phase-change memory with 1.2 V 12ns read access time and 1 MB/s write throughput," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2010, pp. 268–269, doi: 10.1109/ISSCC.2010.5433911.

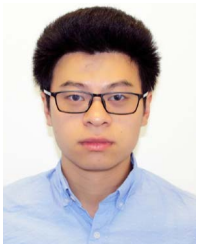
[28] C.-X. Xue *et al.*, "Embedded 1-Mb ReRAM-based computing-in- memory macro with multibit input and weight for CNN-based AI edge processors," *IEEE J. Solid-State Circuits*, vol. 55, no. 1, pp. 203–215, Jan. 2020, doi: 10.1109/JSSC.2019.2951363.

[29] W.-H. Chen *et al.*, "A 65 nm 1 Mb nonvolatile computing-in-memory ReRAM macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2018, pp. 494–496, doi: 10.1109/ISSCC.2018.8310400.

[30] Z. Wang *et al.*, "An all-weights-on-chip DNN accelerator in 22 nm ULL featuring 24 × 1 mb eRRAM," in *Proc. IEEE Symp. VLSI Circuits*, Honolulu, HI, USA, Jun. 2020, pp. 1–2, doi: 10.1109/VLSI-Circuits18222.2020.9162811.

[31] Z. Li *et al.*, "An 879GOPS 243 mW 80fps VGA fully visual CNN-SLAM processor for wide-range autonomous exploration," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2019, pp. 134–136, doi: 10.1109/ISSCC.2019.8662397.

[32] S. Pal *et al.*, "OuterSPACE: An outer product based sparse matrix multiplication accelerator," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2018, pp. 724–736.
[33] C.-C. Chou *et al.*, "An N40 256 × 44 embedded RRAM macro with SL-precharge SA and low-voltage current limiter to improve read and write performance," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2018, pp. 478–480, doi: 10.1109/ISSCC.2018.8310392.
[34] A. Chen and M. Lin, "Variability of resistive switching memories and its impact on crossbar array performance," in *Proc. Int. Rel. Phys. Symp.*, Monterey, CA, USA, Apr. 2011, pp. MY.7.1–MY.7.4, doi: 10.1109/IRPS.2011.5784590.
[35] P. Jain *et al.*, "13.2 A 3.6 Mb 10.1 Mb/mm^2 embedded non-volatile ReRAM macro in 22 nm FinFET technology with adaptive forming/set/reset schemes yielding down to 0.5 V with sensing time of 5 ns at 0.7 V," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2019, pp. 212–214, doi: 10.1109/ISSCC.2019.8662393.
[36] M.-F. Chang *et al.*, "Low VDDmin swing-sample-and-couple sense amplifier and energy-efficient self-boost-write-termination scheme for embedded ReRAM macros against resistance and switch-time variations," *IEEE J. Solid-State Circuits*, vol. 50, no. 11, pp. 2786–2795, Nov. 2015, doi: 10.1109/JSSC.2015.2472601.
[37] J.-F. Zhang, C.-E. Lee, C. Liu, Y. S. Shao, S. W. Keckler, and Z. Zhang, "SNAP: A 1.67–21.55TOPS/W sparse neural acceleration processor for unstructured sparse deep neural network inference in 16 nm CMOS," in *Proc. Symp. VLSI Circuits*, Kyoto, Japan, Jun. 2019, pp. C306–C307, doi: 10.23919/VLSIC.2019.8778193.
[38] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "14.5 envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28 nm FDSOI," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2017, pp. 246–247, doi: 10.1109/ISSCC.2017.7870353.



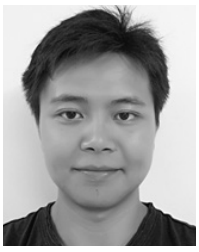
Ziyun Li (Member, IEEE) received the B.S. and Ph.D. degrees in electrical and computer engineering from the University of Michigan, Ann Arbor, MI, USA, in 2014 and 2019, respectively.

His is currently with Facebook, Redmond, WA, USA. His research interests include high-performance, energy-efficient computer vision/machine learning processing units to enable next generation intelligent, autonomous vision system for AR/VR.

Dr. Li was a recipient of the Best Paper Award at the 2016 IEEE Workshop on Signal Processing Systems.

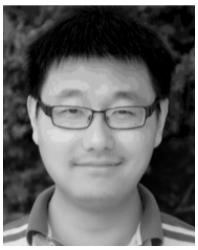
Zhehong Wang (Graduate Student Member, IEEE) received the B.E. degree in electronics and information engineering from Zhejiang University, Hangzhou, China, in 2016, and the M.S. degree from the University of Michigan, Ann Arbor, MI, USA, in 2019. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Michigan, Ann Arbor, MI, USA.

His current research interests include application-oriented ASIC, such as DNA sequencing, Machine Learning, Fully Homomorphic Encryption, and emerging memory design.



Li Xu (Graduate Student Member, IEEE) received the B.Eng. degree in automation from Tongji University, Shanghai, China, in 2009, and the M.S. degree in the electrical and computer engineering from Northeastern University, Boston, MA, USA, in 2016. He is currently pursuing the Ph.D. degree with the University of Michigan, Ann Arbor, MI, USA.

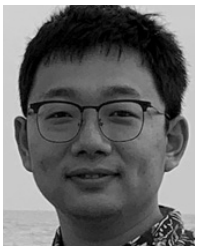
From 2009 to 2011, he was an IC Design Engineer at Ricoh Electronic Devices Shanghai Co., Ltd., Shanghai, China, where he worked on LDO and dc/dc converter projects. During 2015, he was a Design Intern at Linear Technology Corporation, Colorado Springs, CO, USA. His current research interest is energy-efficient mixed-signal circuit design.



Qing Dong (Member, IEEE) received the B.S. and M.S. degrees in microelectronics from Fudan University, Shanghai, China, in 2010 and 2013, respectively, and the Ph.D. degree in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2017.

He is currently with TSMC, San Jose, CA, USA. His current research interests include memory circuit design.

Dr. Dong was a recipient of the Best Paper Awards at 2012 IEEE International Conference on Solid-State and Integrated Circuit Technology, 2015 IEEE International Symposium on Circuits and Systems, and 2016 IEEE Symposium on Security and Privacy.



Bowen Liu (Graduate Student Member, IEEE) received the M.S. degree in electrical engineering and computer science from the University of Michigan, Ann Arbor, MI, USA, in 2018, where he is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Computer Science (EECS).

His research interests include deep learning, computer vision, signal processing, and their applications in low-power systems.

Chin-I Su (Member, IEEE) received the M.S. degree in electrical and control engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2015.

He joined TSMC in 2015 until now. He is currently a RRAM Macro Design Engineer.

Wen-Ting Chu (Member, IEEE) received the Ph.D. degree from the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan.

In 1997, he joined TSMC Hsinchu, Taiwan, and he works on embedded memory technology development including flash, and emerging memories. He is currently an TSMC Academician.

George Tsou (Member, IEEE) received the M.S.E.E degree from National Taiwan University, Taipei, Taiwan, in 1992.

He joined Ti-Acer in 1994 and contributed in Product Engineering and SPICE modeling. Since 1998, he has been with TSMC. During 22 years with TSMC, he was engaged in SPICE modeling and memory circuit design. He has contributed to circuit design for DRAM, MRAM, and RRAM. He is currently a Manager responsible for RRAM macro design. He holds 48 U.S. patents.

Yu-Der Chih (Member, IEEE) received the B.S. degree in physics from National Taiwan University, Taipei, Taiwan, in 1988, and the M.S. degree in electronics engineering from National Tsing-Hua University, Hsinchu, Taiwan, in 1992.

From 1992 to 1997, he was a Design Engineer of Ethernet transceiver circuit for data communication and a Circuit Design Engineer for SDRAM with TSMC, Hsinchu. In 1997, he joined TSMC, for the development of embedded non-volatile memory IP including embedded flash, OTP, MTP, and emerging memory. He is a TSMC Academician and is currently a Director of the Embedded Nonvolatile Memory Library Department in the Memory Solution Division.

Tsung-Yung Jonathan Chang (Fellow, IEEE) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, USA.

He was a Principal Engineer at Intel, Santa Clara, CA, USA, responsible for second/third level caches for Enterprise server processors. He is a Director leading memory IP development at TSMC, Hsinchu, Taiwan. He is responsible for delivering SRAM compilers, custom SRAM IPs, efuse and OTP for low power, high speed applications for advance technology nodes. He has published 30+ technical papers in IEEE conferences or journals and held 25 patents in embedded memory design.

Dr. Chang serves as the memory subcommittee chair for 2019/2020 ISSCC, TPC members of ISSCC, VLSI, Associate and Guest Editors of Journal of Solid State Circuits, and Associate Editor of IEEE TRANSACTION ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS.

Dennis Sylvester (Fellow, IEEE) received the Ph.D. degree in electrical engineering from the University of California, Berkeley, CA, USA, in 1999.

He is a Professor of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor, MI, USA, where he was a Director of the Michigan Integrated Circuits Laboratory (MICL), a group of ten faculty and 70+ graduate students. He has held Research Staff positions in the Advanced Technology Group of Synopsys, Mountain View, CA, USA, Hewlett-Packard Laboratories, Palo Alto, CA, USA, and Visiting Professorships at the National University of Singapore, Singapore, and Nanyang Technological University, Singapore. He has published over 500 articles along with one book and several book chapters. He holds 43 US patents. His research interests include the design of millimeter-scale computing systems and energy efficient near-threshold computing. He serves as a Consultant and Technical Advisory Board Member for electronic design automation and semiconductor firms in these areas. He co-founded Ambiq Micro, Austin, TX, USA, a fabless semiconductor company developing ultralow power mixed-signal solutions for compact wireless devices.

Dr. Sylvester received an NSF CAREER Award, the Beatrice Winner Award at ISSCC, an IBM Faculty Award, an SRC Inventor Recognition Award, and ten best paper awards and nominations. He was named one of the Top Contributing Authors at ISSCC, most prolific author at IEEE Symposium on VLSI Circuits, and was awarded the University of Michigan Henry Russel Award for distinguished scholarship. He serves on the Technical Program Committee for the IEEE International Solid-State Circuits Conference and on the advisory committee for the IEEE Solid-State Circuits Society. He serves/has served as Associate Editor for the IEEE JOURNAL OF SOLID-STATE CIRCUITS, IEEE TRANSACTIONS on COMPUTER-AIDED DESIGN (CAD), and IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS and was an IEEE Solid-State Circuits Society Distinguished Lecturer from 2016 to 2017. His dissertation was recognized with the David J. Sakrison Memorial Prize as the most Outstanding Research in the EECS Department, UC-Berkeley, Berkeley.

Hun-Seok Kim (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 from the University of California, Los Angeles (UCLA), CA, USA, in electrical engineering, both in 2010.

He was a Technical Staff Member at Texas Instruments, Dallas, TX, USA, from 2010 to 2014. He is currently an Assistant Professor with the University of Michigan, Ann Arbor, MI, USA. His research focuses on system analysis, novel algorithms, and efficient VLSI architectures for low-power/high-performance wireless communication, signal processing, computer vision, and machine learning systems.

Dr. Kim was a recipient of the 2018 Defense Advanced Research Projects Agency (DARPA) Young Faculty Award (YFA). He serves as an Associate Editor for the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, and IEEE SOLID STATE CIRCUITS LETTERS.

David Blaauw (Fellow, IEEE) received the B.S. degree in physics and computer science from Duke University, Durham, NC, USA, in 1986 and the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 1991.

Until 2001, he was with Motorola, Inc., Austin, TX, USA, where he was the Manager of the High Performance Design Technology Group and. Since 2001, he has been the Faculty of the University of Michigan, Ann Arbor, MI, USA, where he is the Kensall D. Wise Collegiate Professor of EECS and the Director of the Michigan Integrated Circuits Lab. He has published over 600 papers, has received numerous best paper awards and holds 65 patents. He has performed extensive research in ultralow-power computing using subthreshold operation and analog circuits for millimeter sensor systems, which was selected by the MIT Technology Review as one of the year's most significant innovations. For high-end servers, his research group introduced so-called near-threshold computing, which has become a common concept in semiconductor design. Most recently, he has pursued research in cognitive computing using analog, in-memory neural-networks for edge-devices and genomics acceleration for precision health.

Dr. Blaauw received the 2016 SIA-SRC Faculty Award for lifetime research contributions to the U.S. semiconductor industry and won the Motorola Innovation Award. He was the General Chair of the IEEE International Symposium on Low Power, the Technical Program Chair for the ACM/IEEE Design Automation Conference, and serves for the IEEE International Solid-State Circuits Conference's technical program committee.