Colonnade: A Reconfigurable SRAM-Based Digital Bit-Serial Compute-In-Memory Macro for Processing Neural Networks

Hyunjoon Kim, *Student Member, IEEE*, Taegeun Yoo, *Member, IEEE*,
Tony Tae-Hyoung Kim, *Senior Member, IEEE*, and Bongjin Kim, *Senior Member, IEEE*

***Abstract*—This article (Colonnade) presents a fully digital bit-serial compute-in-memory (CIM) macro. The digital CIM macro is designed for processing neural networks with reconfigurable 1–16 bit input and weight precisions based on bit-serial computing architecture and a novel all-digital bitcell structure. A column of bitcells forms a column MAC and used for computing a multiply-and-accumulate (MAC) operation. The column MACs placed in a row work as a single neuron and computes a dot-product, which is an essential building block of neural network accelerators. Several key features differentiate the proposed Colonnade architecture from the existing analog and digital implementations. First, its full-digital circuit implementation is free from process variation, noise susceptibility, and data-conversion overhead that are prevalent in prior analog CIM macros. A bitwise MAC operation in a bitcell is performed in the digital domain using a custom-designed XNOR gate and a full-adder. Second, the proposed CIM macro is fully reconfigurable in both weight and input precision from 1 to 16 bit. So far, most of the analog macros were used for processing quantized neural networks with very low input/weight precisions, mainly due to a memory density issue. Recent digital accelerators have implemented reconfigurable precisions, but they are inferior in energy efficiency due to significant off-chip memory access. We present a regular digital bitcell array that is readily reconfigured to a 1–16 bit weight-stationary bit-serial CIM macro. The macro computes parallel dot-product operations between the weights stored in memory and inputs that are serialized from LSB to MSB. Finally, the bit-serial computing scheme significantly reduces the area overhead while sacrificing latency due to bit-by-bit operation cycles. Based on the benefits of digital CIM, reconfigurability, and bit-serial computing architecture, the Colonnade can achieve both high performance and energy efficiency (i.e., both benefits of prior analog and digital accelerators) for processing neural networks. A test-chip with 128 × 128 SRAM-based bitcells for digital bit-serial computing is implemented using 65-nm technology and tested with 1–16 bit weight/input precisions. The measured energy efficiency is 117.3 TOPS/W at 1 bit and 2.06 TOPS/W at 16 bit.**

***Index Terms*—All-digital implementation, compute-in-memory (CIM), dot-product, neural network, SRAM, vector-matrix multiply.**

I. Introduction

ARTIFICIAL intelligence, data-driven computing, machine learning, and optimization tasks have sparked interest in the development of hardware accelerators that are specialized in processing artificial neural networks using massively parallel data. The accelerators have pushed the massive amount of data processing from the cloud to edge to address increasing concerns of communication bandwidth, latency, privacy, and security.

Multiply-and-accumulate (MAC) is a critical arithmetic logic operation of the hardware accelerators for processing artificial neural networks. For instance, a convolutional neural network (CNN) typically requires billions of MAC operations for testing a single image classification. Hence, the design of MAC circuit and arithmetic logic unit (ALU) architecture comprising of many MAC units plays a crucial role in the performance of the hardware accelerators.

Traditionally, we have separated the physical location of ALU and memory and used them for their own purposes (i.e., ALU for computing and memory for storing data) based on the popular Von-Neumann architecture. However, as the amount of data increases explosively, there are rapidly growing concerns regarding the excessive energy consumptions and significant latencies mainly due to essential data communications between ALU and OFF-chip DRAM modules (so-called Von-Neumann bottleneck). One of the recent design approaches to address the Von-Neumann bottleneck, especially for battery-operated mobile devices, is a compute-in-memory (CIM) architecture. Recent CIM architectures [1]–[6], [39]–[42] utilize a local cache memory (e.g., SRAM) for both computing and data storage purposes. It can minimize the need for OFF-chip memory access, a major source of energy consumption in conventional digital hardware accelerators. As a result, the CIM architectures offer orders of magnitude higher energy efficiency compared to traditional

Manuscript received July 1, 2020; revised October 28, 2020 and January 11, 2021; accepted February 12, 2021. Date of publication March 9, 2021; date of current version June 29, 2021. This article was approved by Associate Editor Jonathan Chang. This work was supported in part by a RIE2020 ASTAR AME IAF-ICP under Grant I1801E0030. *(Corresponding author: Hyunjoon Kim.)*

Hyunjoon Kim is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: kimh0003@e.ntu.edu.sg).

Taegeun Yoo is with Samsung Electronics, Hwaseong 16677, South Korea (e-mail: ytgzero@nate.com).

Tony Tae-Hyoung Kim is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: thkim@ntu.edu.sg).

Bongjin Kim is with the Department of Electrical and Computer Engineering at University of California at Santa Barbara, Santa Barbara, CA 93106 USA (e-mail: bongjin@ucsb.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSSC.2021.3061508.

Digital Object Identifier 10.1109/JSSC.2021.3061508

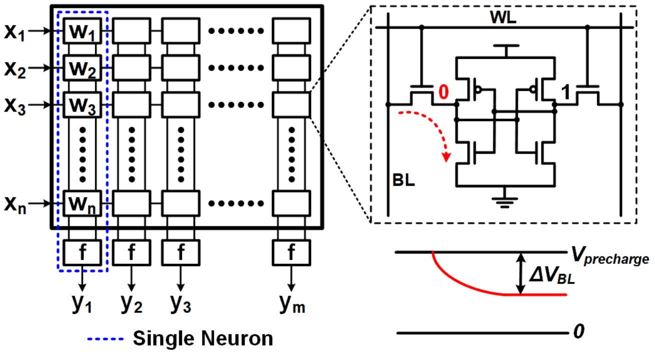


Fig. 1. Standard 6T SRAM-based CIM macro. A column comprising of "n" bitcells and a nonlinear activation (f) represents a neuron.

architectures. However, the CIM approach has suffered from critical challenges, mainly due to its analog-intensive circuits and operations. The challenges include a wide process and real-time variation of the individual MAC operations, the overhead from data conversions between analog and digital domain, and various noise sources that degrade the computation accuracy. Furthermore, prior CIM architectures only provide low bit-precision computations due to a limited memory density and the inherent challenges in the analog-style CIM macros.

Recent innovations in training algorithms have enhanced the prediction accuracies using artificial neural networks based on low-precision computations [10]–[13], [23]–[25]. Besides, recent hardware implementations have utilized different quantization techniques and achieved high energy efficiency while minimizing accuracy loss [1], [5], [7]–[9]. Although the recent efforts in both algorithms and hardware enabled the use of quantized precision computing for tiny machine learning applications, we still have to address the aforementioned analog challenges for broader utilization of CIM architecture in edge computing. In this work, we propose Colonnade, a digital bit-serial CIM macro, which is entirely free from analog non-idealities, for the first time. The precision of the proposed digital macro is fully reconfigurable from 1 to 16 bit. It is highly efficient in both energy and area thanks to its unique and regular bitcell array and the adopted efficient bit-serial computing architecture.

This article is organized as follows. Section II describes the background and motivation of the proposed digital CIM macro. Section III introduces the proposed reconfigurable digital bit-serial CIM architecture. Section IV explains detailed circuits and architectures with examples. Section V describes the simulated and measured results of the fabricated macro test-chip. Section VI presents further evaluation results with discussion. Section VII concludes this article.

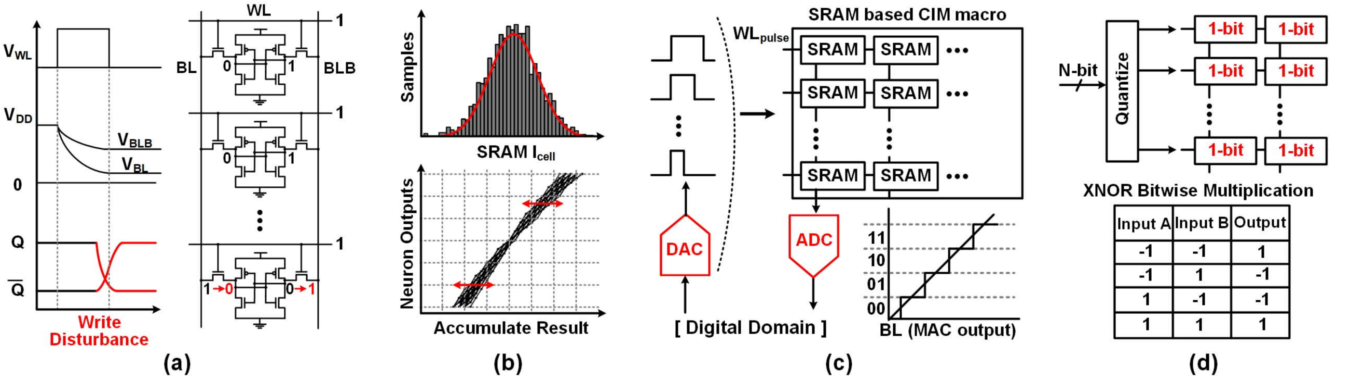
II. Background

CIM architecture alleviates large energy consumption due to frequent OFF-chip memory access of conventional architecture. Besides, the analog-intensive computation of prior CIM macros reduces the computing energy at the expense of analog-specific non-idealities such as process variation and noise susceptibility. Fig. 1 shows a popular example of analog CIM macro based on SRAM. As shown in Fig. 1 (left), a single column of the SRAM macro that consists of hundreds of bitcells and an activation (f) works as a neuron. Each bitcell works as a bitwise MAC unit, and a column of these bitcells forms a dot-product. Typically, inputs are connected to the horizontal wordlines (WLs), and the bitcells connected to the wordlines are enabled or disabled (i.e., bitwise multiplication) for a short period of time. Note that both input (0 or 1) and weight (-1 or $+1$) precisions are binary when a single bitcell is used for storing a weight and works as a MAC unit. When the bitcell is enabled (i.e., input is 1), a unit current discharging path is formed between one of the shared bitlines (BL or BLB) and an internal SRAM node that is pulled down via NMOS transistor as shown in Fig. 1 (right). Similarly, all the other bitcells in the column contribute to a voltage drop in one of the two bitlines. Finally, the accumulated voltage difference is converted to a binary output by reusing a sense-amplifier of the SRAM macro. While a popular current-mode operation is described in this example, other types of CIM macros are based on different analog accumulate operation types (e.g., voltage- or charge-mode).

A. Design Challenges of Analog CIM

Fig. 2 illustrates key challenges in SRAM-based analog CIM architectures. First, SRAM-based macros with shared read and write bitlines suffer from a write disturbance and the associated narrow bitline dynamic range. When a bitline voltage level goes below a threshold during the accumulate operation, it could flip the data stored in bitcells of the same column, as shown in Fig. 2(a). Besides the disturbance, there are critical analog-specific non-idealities including process variation and performance/area overhead due to the essential data conversions, as shown in Fig. 2(b) and (c). Fig. 2(d) shows the limited compute precision, which is another challenge that limits the wide adoption of analog CIM macros in versatile edge computing applications.

Recent analog CIM macros [3], [5], [32], [33], [42] addressed the write disturbance issue with workaround methods through modifying SRAM bitcells. One such example is decoupling memory write and in-memory computations by adding extra transistors that are dedicated for MAC computations. However, such a design method always sacrifices the bitcell size (i.e., reduced memory density). ON-chip training methods were developed in efforts to mitigate the negative impact of process variation on the performance of accelerators [6], [15], [16], [39]. Instead of loading the pre-trained weights on a variation-prone hardware, weights trained ON-chip would self-calibrate offset information and minimize performance degradation due to process variation. However, there is a major challenge in the development of ON-chip training hardware. The training typically requires high precision when computing derivatives for back-propagation and other training algorithms. Computing and storing the high-precision data demands expensive trade-offs in energy/area-efficiency, and that eventually diminishes the advantage of efficient analog CIM architecture.



Input A	Input B	Output
-1	-1	1
-1	1	-1
1	-1	-1
1	1	1

Fig. 2. Design challenges of SRAM-based analog CIM architectures. (a) Write disturbance issue, (b) process variation, (c) ADC/DAC overhead, and (d) low precision operation.

The overhead associated with analog-to-digital and digital-to-analog conversions (ADC/DAC) for input–output delivery is also a major concern. The data conversions not only consume significant energy and macro area but also significantly degrade the throughput and latency of the accelerator. In practice, a bulky ADC circuit is shared by tens or hundreds of column- or row-based neurons in an analog CIM macro. Besides, data converter circuits typically operate with fixed bit-precision (i.e., limited reconfigurability), and its analog-intensive circuitry makes it challenging to scale and susceptible to device and supply noise.

B. State-of-the-Art Analog CIM

Deep in-memory architecture (DIMA) [2] is similar to a standard SRAM-based CIM macro, which accumulates the individual bitcell multiplication results via shared bitlines. Hence, it suffers from the write disturbance issue and process variation. It is unique in the multibit weight implementation, which modulates the pulsewidth of WL input signals. However, a binary-weighted pulse generation is not a trivial job, and its timing uncertainty degrades the computation reliability, as shown in Fig. 3(a). Conv-SRAM [3] proposes a 10T SRAM bitcell that decouples the memory write and MAC operations, and hence addresses the write disturbance issue. In addition, it enhances the dynamic range by adopting a charge-based accumulation. While utilizing a bitcell as a binary weight storage, a multibit input is realized using a pulsewidth control scheme. Hence, it suffers from the imperfect timing generation (similar to DIMA [2]). Besides the issues in the accurate pulse generations, Conv-SRAM has remaining issues, including a low bitcell density due to its large bitcell with extra transistors for MAC operations and ADC/DAC overhead. XNOR-SRAM [5] presents a 12T SRAM-based CIM macro, which decouples memory write and MAC operations. Six transistors are added to implement a multiplication between a binary weight and a ternary input and a pull-up/down driver-based voltage-mode accumulation. A key challenge is in its inherent nonlinear and asymmetric transfer characteristic, as shown in Fig. 3(b). The non-ideal characteristics are mainly due to a pull-up and pull-down strength imbalance caused by a mobility difference in P/NMOS transistors on top of significant process variations. Valavi *et al.* [4] propose an 8T1C SRAM-based CIM macro, which computes a bitwise multiplication and a charge-domain accumulation using two extra transistors as sampling switches and a metal-oxide-metal (MOM) capacitor. While the macro achieves the outstanding energy efficiency and throughput, it requires a sophisticated compensation circuit to resolve charge injection error, as shown in Fig. 3(c).

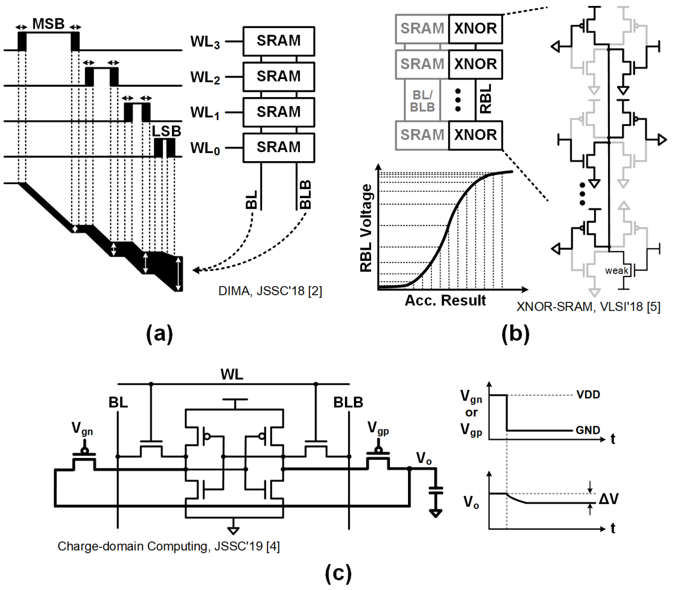


Fig. 3. Design specific issues in analog CIM. (a) Current domain, (b) voltage domain, and (c) charge domain.

C. Digital Accelerators

Prior digital accelerators have focused on reducing the OFF-chip memory access through data reuse, architectural and algorithmic improvements, and novel data flow techniques [7], [17], [18], [26]–[29]. Fig. 4 shows the energy cost of the conventional digital accelerators due to OFF- and ON-chip memory access and data movements during MAC operations. Note that the off-chip memory (i.e., DRAM) access consumes orders of magnitude higher energy than local data movements.

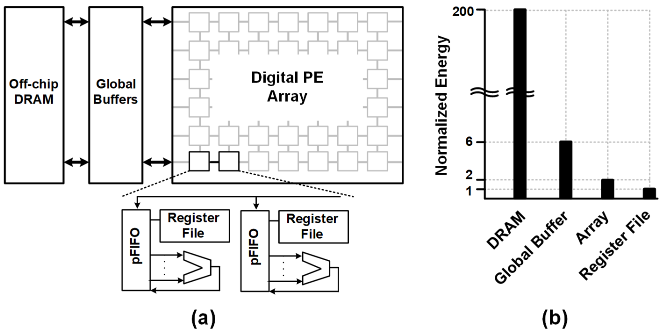


Fig. 4. (a) Block diagram of conventional digital PE array for machine learning acceleration [7] and (b) memory access energy breakdown [7], [21].

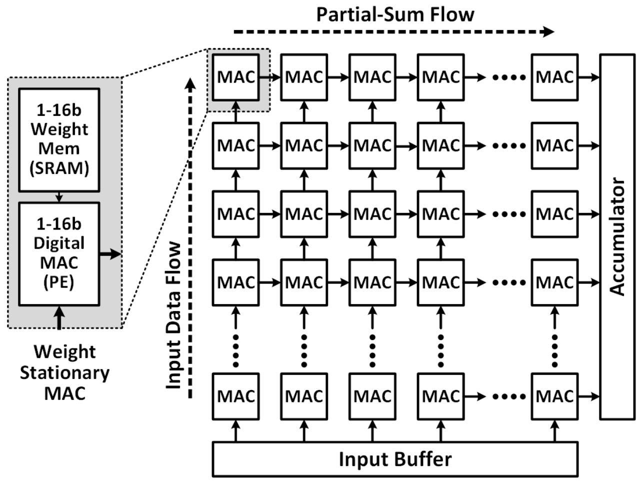


Fig. 5. Weight-stationary systolic architecture of the proposed digital CIM macro with 1–16 bit reconfigurable MAC precision.

Eyeriss [7] presents a spatial architecture of PE array and a row-stationary data flow to maximize data reuse and improve energy efficiency for CNN applications. However, it is not reconfigurable in terms of computing precision. DianNao series [17], [27] and EIE [18] demonstrate high performance with large compute arrays and compression techniques for inference, but their large hardware footprint is a disadvantage. Neural Cache [28] repurposes cache memory into a near-memory computing macro using a bit-serial implementation that reduces the computation complexity with minimal hardware overhead. Duality Cache [43] and CRAM [14] further develop Neural Cache with 8T SRAM cell which enables CIM operation for the partial generation of MAC results while utilizing the near-memory compute for carrying propagation and accumulation. Minerva [29] provides a design methodology for low-power neural network accelerators, from optimization techniques to micro-architecture implementations. However, its architecture options are limited to high-level data paths and low-level operations do not change for different design configurations. Envision [31] presents circuit-level scalability by implementing dynamic voltage-accuracy-frequency scaling in a 2-D SIMD architecture, but requires a special process technology to reduce the leakage current from MAC operations and retains the OFF-chip memory access energy issue. Some of the recent digital accelerators have also implemented reconfigurable bit-precisions in their neural network processing to optimize the overall energy efficiency based on layer-by-layer precision control while minimizing accuracy degradations [8], [22]. Nevertheless, the digital accelerators have fundamental limits in the achievable energy efficiency, especially compared to the analog CIM implementations. In fact, the CIM architecture was overlooked in the digital accelerator designs, so far. The limited precision and the lack of reconfigurability of the existing CIM macros make it difficult to consider a digital implementation of CIM macro. In this work, we overcome the current challenges in both analog and digital accelerators by proposing a novel fully digital CIM macro.

III. Proposed SRAM-Based Digital CIM

Previous digital hardware accelerators consume a significant portion of their total energy for accessing OFF-chip memory despite the efforts to reduce data movement. Nevertheless, there are several advantages to digital accelerators over the analog implementations. One of the key benefits is the robustness of digital designs. Digital accelerators are insensitive to the process variation and various noise sources as its core computation mechanism includes an abstraction layer on top of the analog signal values. Another key advantage of digital accelerators over analog counterparts is the elimination of data conversions and the associated reduction in both energy and area consumption. Aside from the energy and area efficiency gain, the removed data conversion stage gives us a computation simplicity and relaxes performance bottleneck due to time-multiplexed operation. In this section, we introduce how we take advantage of the benefits of digital accelerators mentioned above and minimize memory access.

A. Weight-Stationary Systolic Architecture

A systolic array of processing elements (PEs) is an efficient data processing architecture that can achieve high throughput based on its parallel computations and natural input–output data flow. Both inputs and partial-sums pass through a 2-D array while pipelined parallel MAC operations are being performed at the distributed digital PEs. A similar 2-D PE array is also used for implementing the analog CIM macro. However, all the operations within the analog CIM macro are performed in parallel without pipelining due to its relatively smaller macro size and the faster analog computation. Furthermore, the analog macro not only computes parallel MAC operations but also stores weights in its memory array (i.e., weight-stationary). The proposed digital CIM macro is based on the weight-stationary systolic architecture, as shown in Fig. 5. The proposed macro combines the benefits of the systolic array (high throughput) and the CIM macro (low latency and high energy efficiency). Note that the CIM macro supports 1–16 bit precision for both inputs and weights. More specifically, the proposed architecture is a digital bitcell array that takes operation directionality of the bit-parallel systolic PE array and works as a precision reconfigurable bit-serial

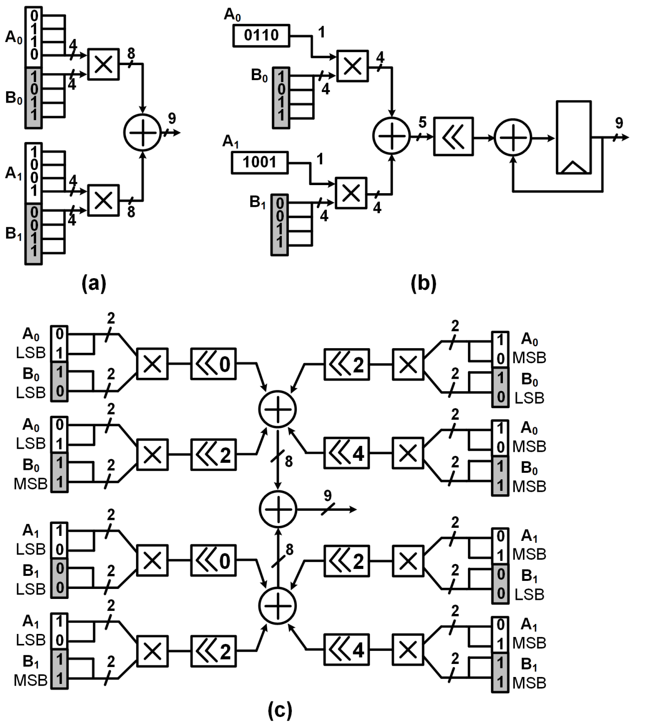


Fig. 6. Digital ALU architecture. (a) Conventional bit-parallel, (b) bit-serial [19], and (c) reconfigurable bit-precision [20].

digital CIM macro, saving significant area versus the systolic PE array (e.g., TPU [34]).

B. Bit-Serial Computing and Reconfigurability

Fig. 6(a) illustrates an example of a conventional bit-parallel ALU that consists of two 4 bit multipliers and an 8 bit adder. Two pairs of 4 bit inputs are multiplied, and then the resulting 8 bit outputs are added in the following adder. Note that both power and area of bit-parallel digital multipliers are quadratic functions of their input precision. As an effort to reduce the MAC area, Stripes [19] presents bit-serial computing that can significantly save area by serializing one of the multibit inputs and replacing its area-consuming multiplier circuit with compact bitwise ALUs, as shown in Fig. 6(b). The area gain of the bit-serial ALU circuit is much higher as the precision increases since the complexity of the bit-serial computation only grows as a linear function of the bit-precision. Note that the bit-serial computing requires an extra circuit that accumulates partial-sums from each operation cycle. The input is serialized from LSB to MSB, while each serial bit is used for generating a partial-sum. The lack of reconfigurability is another limitation of the conventional bit-parallel ALUs. Fig. 6(c) shows BitFusion [20] comprising of reconfigurable ALU units with a fine-grained bit-precision. A regular structure of low-precision ALU units can be grouped and operate as a higher-precision ALU.

Aside from minimizing the OFF-chip memory access and preserving the benefits of digital architecture, the proposed digital CIM macro also reduces the area consumption by adopting the computing paradigm of the bit-serial ALU. Furthermore, the proposed macro also adopts a BitFusion-like regular two-dimensional digital bitcell structure that is readily reconfigured from 1 to 16 bit to accommodate a wide range of performance and energy requirements.

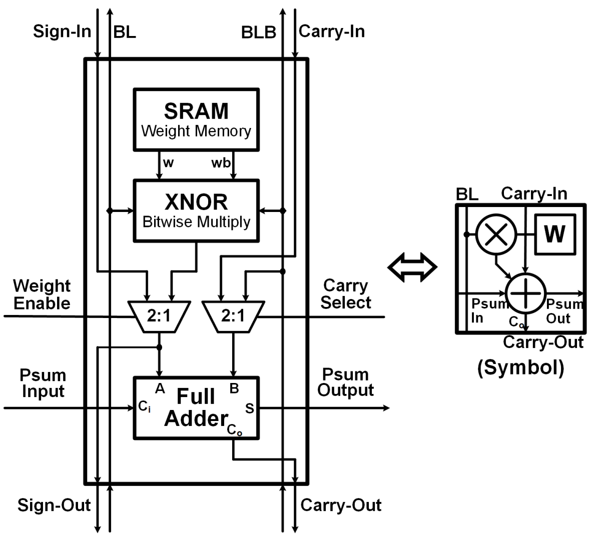


Fig. 7. Block diagram (left) and symbol (right) of the proposed digital bitcell for reconfigurable weight precision and bit-serial computing.

IV. Colonnade Architecture

A. Bitcell Configuration and Column MAC Operation

Fig. 7 shows a block diagram of the proposed digital bitcell. A bitcell is composed of three major building blocks: a full-custom designed standard 6T SRAM cell for storing a binary weight, an XNOR gate for a bitwise multiplication, and a full-adder for accumulating partial sum. Two 2:1 multiplexers (MUXs) are added for the selection of internal signals in different configurations. MUXs are used to configure the operating mode of the bitcell in column MACs and to determine the LSB bitcell which is located at the top of each column MAC. Fig. 8 describes an operation example of two columns with five bitcells per each to form a cascaded two 4 bit bit-serial MAC units. Each bitcell is configured to one of the two different functional models (i.e., Type-A and Type-B) based on its location within the column bitcell array. Type-A bitcell enables all three building blocks in a bitcell, while Type-B only enables full-adder for accumulate-only operation. The bit-precision of weights is configured by the number of Type-A bitcells in a column (e.g., 4 bit in Fig. 8), while type-B bitcells are added to extend the output precision which also depends on the number of columns. For example, the number of Type-B bitcells are 7 for each column MAC in the 128-column array and the output precision at 1 bit, 4 bit and 16 bit weights are 8 bit, 11 bit, and 23 bit, respectively. Each cycle of bit-serial MAC operation, a serialized input on the shared bitline is multiplied to a 4 bit weight stored in SRAM cells in a 4 bit column MAC. Bitwise multiplication results are then accumulated at the following ripple carry adder that is formed by vertically connected full-adders from each bitcells. The bit-serial MAC computations are performed through all the column MACs in the same row. Once a partial sum output value on the far right of the column is settled, then it is further post-processed

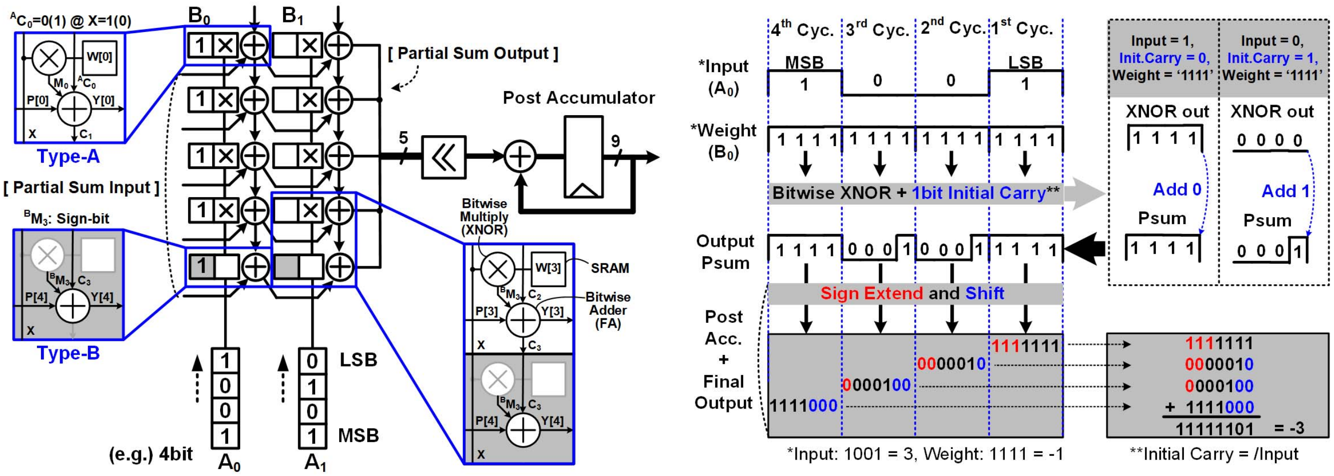
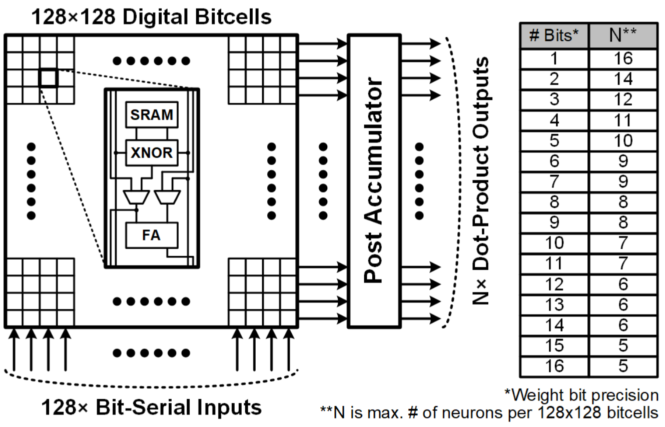


Fig. 8. Building a column MAC array using bitcells. In the example on the left, 10 bitcells are used for building a dot-product with 4 bit weight/input. The example shown on the right describes bit-serial multiplication of 4 bit two's complement weight and 4 bit binary weighted signed number input.



# Bits*	N**
1	16
2	14
3	12
4	11
5	10
6	9
7	9
8	8
9	8
10	7
11	7
12	6
13	6
14	6
15	5
16	5

*Weight bit precision
**N is max. # of neurons per 128x128 bitcells

Fig. 9. CIM macro with 128 × 128 bitcells for N× dot-products.

for merging partial sum results from each cycle of bit-serial operation.

Fig. 9 shows a complete CIM macro comprising of 128 × 128 digital bitcells and a post-accumulator. The bitcell array can be readily reconfigured to operate as parallel dot-products. Each row of the reconfigured column MACs performs a dot-product computation with variable bit-precisions. We assign the number of Type-A bitcells to represent the weight precision while the number of Type-B bitcells is determined based on the dynamic range of partial-sum, which depends on the number of columns. For instance, we can assign 1 Type-A and 7 Type-B bitcells as a single column MAC when reconfiguring the CIM macro into sixteen 1 bit dot-products, as shown in Fig. 10, left. The macro can be reconfigured to eight 9 bit dot-products by changing the number of Type-A bitcells per column MAC from 1 to 9, as shown in Fig. 10, right. Note that the number of bit-serial operation cycles programs the input precision, and hence 9 bit dot-products require 9× more operation cycles than that of 1 bit dot-products.

A 16 bit partial-sum (i.e., P_i [15:0]) is generated from each cycle (i = 0 to 8) of the dot-product operation between 128 × 9 bit weights and inputs. Each cycle, the binary-weighted inputs are serialized and used for computing dot-product partial-sums. Hence, the partial-sums are left-shifted and accumulated at the following postaccumulator, as shown in Fig. 11.

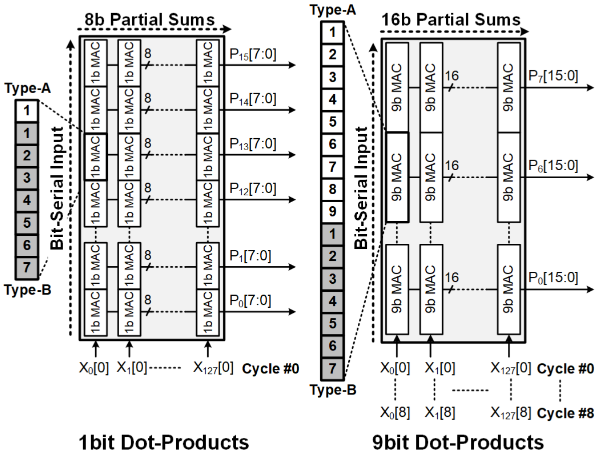


Fig. 10. Reconfigured CIM macros with 128 × 128 bitcells. Each column MAC requires M Type-A cells and 7 Type-B cells.

TABLE I
NUMBER REPRESENTATION

	Number Representation & Abstraction
Input	(0, 1) represents (-1, +1) [e.g.] 1001 = $2^3 - 2^2 - 2^1 + 2^0 = 3$
Output / Weight	Two's Complement
First Carry-In (C_0 = /X_{IN})	$C_0 = 0$ @ $X_{IN} = 1$ (i.e., +1) $C_0 = 1$ @ $X_{IN} = 0$ (i.e., -1)

B. Number Representation and Operation Example

Table I shows the number representation scheme for input–output, weight, and the first carry-in (C_0) of the MAC unit. The weight is stored as a two's complement signed number while the input is encoded to a binary-weighted signed

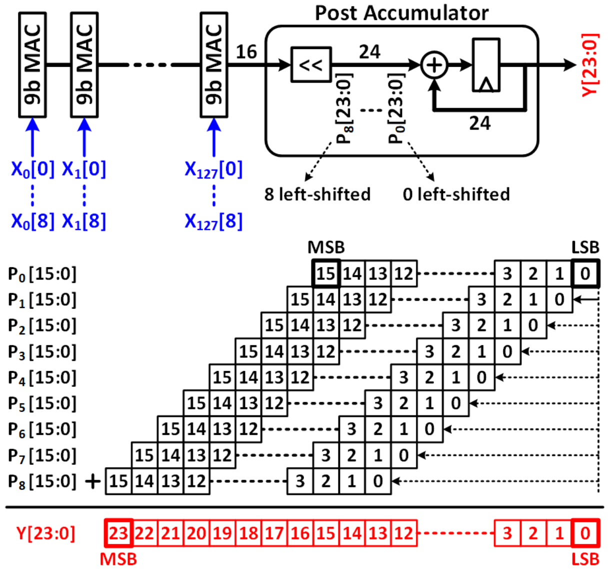


Fig. 11. Postaccumulator combines individual partial-sums from each operation cycle to complete a dot-product with multibit precision.

number, which realizes the bit-serial input as the serialized binary value of +1/−1. Compared to the traditional binary-weighted two's complement number, the proposed encoding scheme provides simpler operation by not having to spend more compute cycles for the sign bit and encoding stages to express the signed number. For instance, the encoded 4 bit-serial input "0110" represents a decimal number −3 since "$-3 = -2^3 + 2^2 + 2^1 - 2^0$," where 0 represents −1 and 1 represents +1. Each cycle of MAC operation can be broken into three steps.

Fig. 12 illustrates a detailed operation example of a four-step CIM dot-product with two 4 bit weights (i.e., $W_0 = -3$ and $W_1 = 6$) and two binary inputs (i.e., $X_0 = -1$ and $X_1 = +1$). The dot-product operation is performed using two 4 bit column MACs, each with four Type-A bitcells and one Type-B bitcell. For simplicity, the input is set to binary, and hence the post-accumulation is not required in this example. As a first step, both weights and inputs are ready to compute bitwise multiplications, as shown in Fig. 12(a). Note that the bottom Type-B bitcell extends the sign from the above Type-A bitcell (MSB of the weight). The first carry-in of each column is fed to top full-adders (LSB bitcells) to complete two's complement multiply operations. Note that the multiplication results are incorrect without adding the first carry (input bar) as shown in Fig. 8 (right). When the input is 0 (i.e., −1), a 5 bit sign-extended two's complement weight is inverted based on its XNOR-gate-based bitwise multipliers, and then is added to the first carry-in, which is 1 (i.e.,$/X_0 = 1$) in this case. On the other hand, the 5 bit weight is buffered while the first carry-in is 0 when the input is 1 (i.e., +1). Fig. 12(b) shows the bitwise multiplication results. The bitwise operation results are then accumulated in a ripple carry adder on the first column MAC as shown in Fig. 12(c). Finally, the 5 bit partial-sum result is combined with the bitwise multiplication results from the second column MAC on the right, as shown in Fig. 12(d).

C. Sparse Pipeline Architecture

Conventional digital accelerators with systolic architecture retime inputs and partial sums between processing elements. For instance, a 128 × 128 systolic PE array retimes inputs and partial-sums 128 times per each. Hence, the latency due to such retiming operation is 256 cycles. In this work, we implement a programmable number of pipeline stages and find the optimal trade-offs between latency, throughput, and energy efficiency. Fig. 13 shows the reconfigurable number of pipelined register stages from 1 to 16 (i.e., $N = 1$, 2, 4, 8, or 16), where the pipelined registers are inserted into every "128/N" column MACs in a dot-product row so that it can retime partial-sums "N" times per each dot-product operation. Note that the bit-serial inputs are not pipelined in this implementation due to its relatively small CIM macro size. Given the fixed operating frequency, the latency is proportional to the number of pipeline stages. However, the reduced pipeline stages result in the longer critical path delay due to the increased number of column MACs (per each pipeline stage) operating asynchronously. Hence, the maximum operating clock frequency increases as we sweep the number of pipeline stages from 1 to 16, as shown in Fig. 14 (left). The latency is the function of both the number of pipeline stages and the maximum operating clock frequency. Hence, we can find the optimal number of pipeline stages that can give us the smallest latency, as shown in Fig. 14 (right). It is also the function of the bit-precision since the critical path delay also increases as we stack more Type-A bitcells when reconfiguring a column MAC for the higher bit-precision. The best pipeline option is closely related to the weight precision and energy efficiency impact from added column registers while the input precision is less relevant due to bit-serial operation.

V. Experimental Results

Fig. 14 shows the simulated maximum operating frequency (left) and the latency of the proposed digital CIM dot-product operation (right) versus the number of pipeline register stages while sweeping both input and weight bit-precisions from 1 to 16 bit. As the number of pipeline stages increases from 1 to 16, the maximum frequency has been increased by 7.6 times (i.e., from 18.2 to 138.4 MHz) for 1 bit. When the precision is 16 bit, the maximum operating frequency increases by 4.6-times (i.e., from 16.4 to 75.8 MHz). As the number of pipeline stages increases, the column MAC critical path delay decreases, and it also depends more on the bit-precision. As discussed in Section IV-C, the latency depends on both the operating clock frequency and the number of pipeline stages. Hence, there exist optimal points where the latency is minimal based on the trade-off between those two parameters, as shown in Fig. 14 (right).

While the latency tends to be optimal when the number of pipeline stages is smaller [i.e., minimum latency with two stages as shown in Fig. 14 (right)], the other performance metrics show different trends. Fig. 15 (left) shows the simulated throughput, and it decreases as the bit-precision increases or the number of pipeline stages decreases since the maximum clock frequency decreases in both cases. The energy

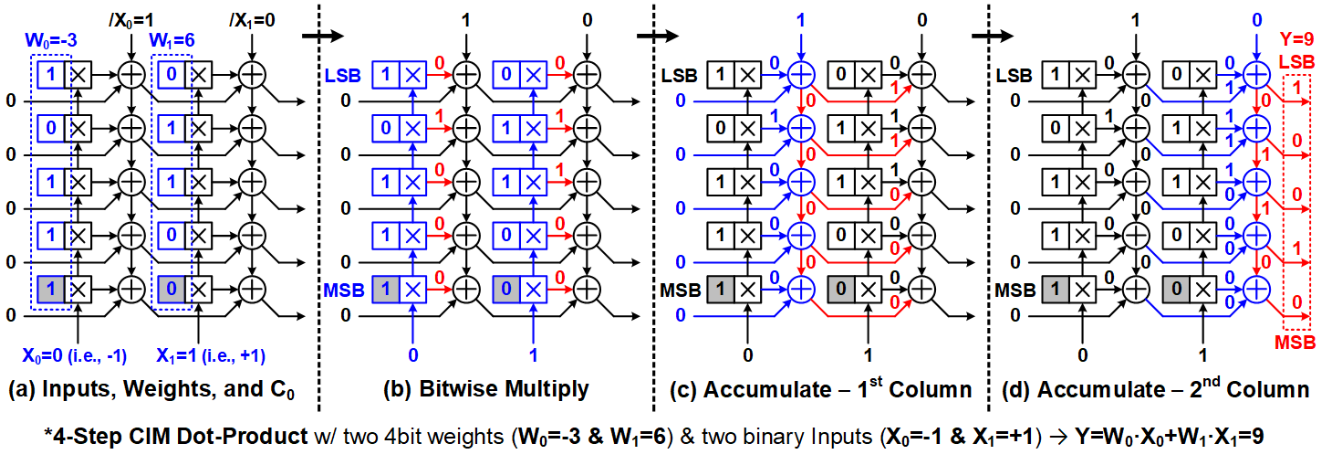

(a) Inputs, Weights, and C0
(b) Bitwise Multiply
(c) Accumulate – 1st Column
(d) Accumulate – 2nd Column
W0=-3
W1=6
/X0=1
/X1=0
X0=0 (i.e., -1)
X1=1 (i.e., +1)
LSB
MSB
Y=9
*4-Step CIM Dot-Product w/ two 4bit weights (W0=-3 & W1=6) & two binary Inputs (X0=-1 & X1=+1) → Y=W0·X0+W1·X1=9

Fig. 12. Detailed operation of column MACs with 4 bit weight and 2× columns (i.e. 5× bitcells per column MAC). The input activation is 1 bit, and hence it takes 1× cycle to complete a full dot-product between two pairs of 4 bit weights and 1 bit input activations.

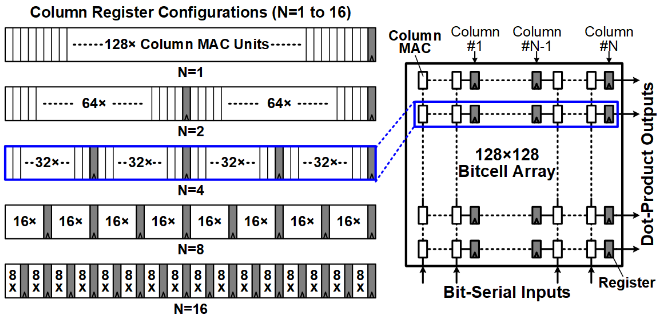


Fig. 13. Column register implementation diagram for improved speed.

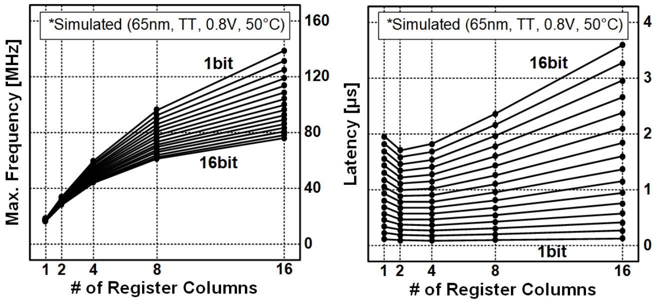


Fig. 14. Maximum clock frequency and latency versus the number of pipeline register stages from 1 to 16.

efficiency also decreases as the bit-precision increases. However, the efficiency increases when the number of pipeline stages is smaller, as shown in Fig. 15 (right). Table II summarizes the performance of the proposed digital CIM macro with selected weight and input bit-precision settings. The proposed macro is fully reconfigurable in terms of bit-precision setting (i.e., 1–16 bit precision for both weight and input). In this summary, we fix the number of pipeline stages to 16 for higher throughput while sacrificing energy efficiency. The maximum operating frequencies depend solely on the weight precisions, and they are 138 and 75.8 MHz for 1 and 16 bit, respectively. The latency ranges from 0.12 to 3.59 μs and is proportional to the number of bit-serial operation cycles (i.e., input bit-precision). When both input and weight precision is 1 bit, we can achieve the highest throughput of 567 GOPS, and the lowest is 6.1 GOPS when the precision is 16 bit. The energy efficiency is also the highest (156 TOPS/W) when both input and weight precision is 1 bit and is the lowest (1.4 TOPS/W) when the precision is 16 bit.

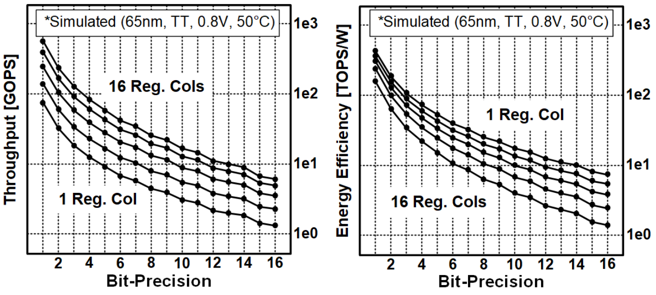


Fig. 15. Throughput and energy efficiency of the proposed CIM macro versus reconfigured bit-precision from 1 to 16.

TABLE II
PERFORMANCE SUMMARY

Reconfigurability	1-16b Weight / 1-16b Input			
Bitcell & Register	128×128 Bitcell Array with 16 Register Columns			
Bit-Precision**	1b/1b	1b/16b	16b/1b	16b/16b
Operation Cycles	1	16	1	16
Column MACs	16×128	16×128	5×128	5×128
Max Frequency	138MHz	138MHz	75.8MHz	75.8MHz
Latency	0.12μs	1.92μs	0.22μs	3.59μs
Throughput	567GOPS	35.4GOPS	97GOPS	6.1GOPS
Energy Efficiency	156TOPS/W	9.7TOPS/W	22TOPS/W	1.4TOPS/W

*Simulated (65nm, TT, 0.8V, 50°C) **Bit-precision setting (weight / input)

A digital CIM macro prototype with 128 × 128 bitcell array is fabricated using 65-nm technology. Fig. 16(a) shows the measured energy efficiency while sweeping bit-precisions for both weights and inputs from 1 to 16 bit. The supply voltage is also swept from 0.6 to 0.8V, and the number of pipeline

TABLE III

PERFORMANCE COMPARISON WITH STATE-OF-THE-ART NEURAL NETWORK ACCELERATORS

	JSSC'19[3]	JSSC'20 [5]	JSSC'19[4]	JSSC'17[7]	JSSC'17[22]	JSSC'19[8]	JSSC'20[14]	ISSCC'20 [40]	ISSCC'19 [42]	This Work
Technology	65nm	65nm	65nm	65nm	40nm	65nm	28nm	28nm	65nm	65nm
Supply Voltage	0.8-1.2V	0.6-1V	0.94/0.68/1.2V	0.82-1.17V	0.8-1.1V	1.1V	0.6-1.1V	0.7-0.9V	N/A	0.6-0.8V
Input/Output Precision	6bit	Binary/ Ternary	Binary	16bit	1-to-16bit	1-to-16bit	Arbitrary	4-8bit	1-4bit	1-to-16bit
Weight Precision	Binary	Binary	Binary	16bit	1-to-16bit	1-to-16bit	Arbitrary	4-8bit	2-5bit	1-to-16bit
Operation Mode	Analog In-memory	Analog In-memory	Analog In-memory	Digital	Digital	Digital	Digital In/Near-memory	Analog In-memory	Analog In-memory	Digital In-memory
GOPS/mm^2	57	5461	1498	3.43	42.5 (16b)	460.75 (1b) 21.6 (16b)	27.3	N/A	N/A	6750 (1b) 25.2 (16b)
Max Throughput [GOPS]	8	614	18876	42	102 (16b)	7372 (1b) 345.6 (16b)	32.7 (8b)	N/A	N/A	567 (1b) 6.1 (16b)
Min. Energy Eff. [TOPS/W]	40.3	N/A	658	N/A	0.27 (16b)	3.08 (16b)	0.55 (8b mult)	11.54	18.37	2.06 (16b)
Max. Energy Eff. [TOPS/W]	51.3	403 (Ternary)	866	0.12	2.71 (4b)	50.6 (1b)	5.27 (8b add)	68.44	72.1	117.3 (1b)
Bitcell Density	16Kb	16Kb	2.4Mb	182.5Kb	148kB	256Kb	128Kb	64Kb	3.75Kb	16Kb
Energy per MAC[pJ/MAC]	0.025	0.002	N/A	N/A	N/A	0.055 (1b) 1.26 (16b)	N/A	N/A	N/A	0.017 (1b) 0.78 (16b)
Die Area[mm^2]	4	N/A	12.6	16	N/A	16	2.7	N/A	N/A	1
Core Area[mm^2]	0.063	0.08	N/A	12.25	2.4	N/A	N/A	N/A	N/A	0.2272
Bitcell Area[μm^2]	N/A	3.9	1.8	N/A	N/A	N/A	0.78	4.11	0.87/0.79	10.53

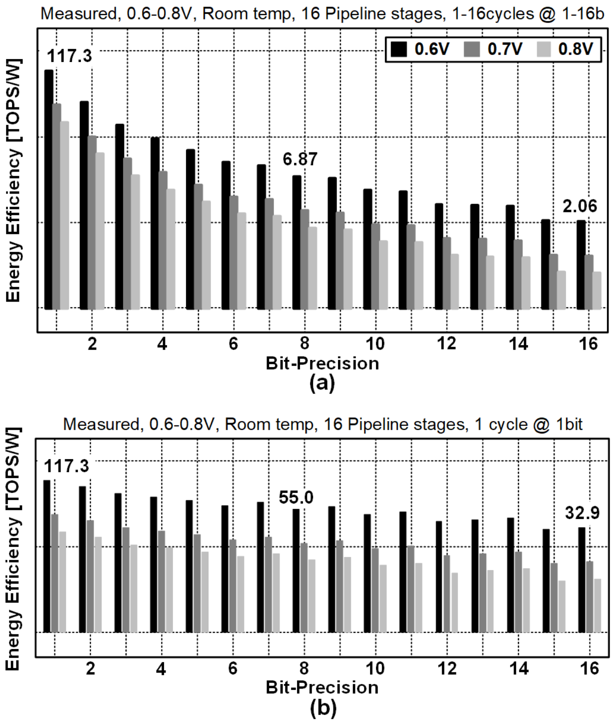


Fig. 16. Measured energy efficiency. (a) 1–16 bit for both weight and input precision. (b) 1–16 bit weight precision (input is fixed to 1 bit).

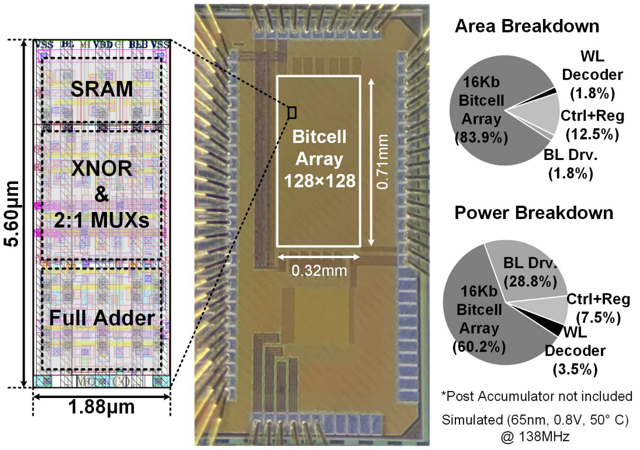


Fig. 17. Bitcell layout (left), die micrograph (center), and area/power breakdown (right).

stages is fixed to 16. The measured energy efficiency at 0.6 V ranges from 117.3 TOPS/W at 1 bit to 2.06 TOPS/W at 16 bit precision. Fig. 16(b) shows similar plots, but the energy efficiency is now measured while sweeping weight precision only. When the weight precision is 16 bit, the energy efficiency is 32.9 TOPS/W, which is 16× higher than the efficiency when both weights and inputs are 16 bit.

Table III shows a performance comparison with state-of-the-art analog CIM and digital neural network accelerators. Note that this work proposes a digital CIM macro that achieves high energy efficiency and reconfigurability while occupying low area regardless of bit-precision. Fig. 17 shows a die micrograph of the fabricated 65-nm test-chip and its bitcell layout.

VI. EVALUATION AND DISCUSSION

In this section, we evaluate the performance of the proposed bit-serial CIM macro architecture for processing DNNs. Note that we use a large bitcell array (4096 × 256), which is more suitable for evaluating large-scale DNNs such as VGG-16. Precision is swept from 1 to 32 bit to assess the scalability of the proposed macro. Performance metrics, including latency, throughput, and energy efficiency, have been investigated and summarized.

A. Macro Performance Evaluation

Fig. 18 (left-top) shows a table that summarizes precisions of input and weight/output and the total number

	Input Precision	Weight/Out Precision	Total # of MACs
1b	1b	1b / 9b	448×256
2b	2b	2b / 10b	384×256
4b	4b	4b / 12b	320×256
8b	8b	8b / 16b	256×256
16b	16b	16b / 24b	160×256
32b	32b	32b / 40b	96×256

Reconfigurable Bit Precisions

Fast	FF corner, 1.2V, 0°C
Typical	TT corner, 1.0V, 50°C
Slow	SS corner, 0.8V, 100°C

65nm LP Process Corners

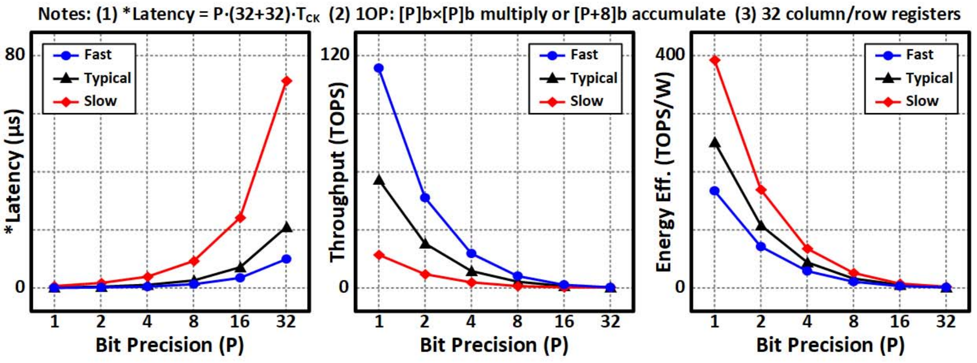
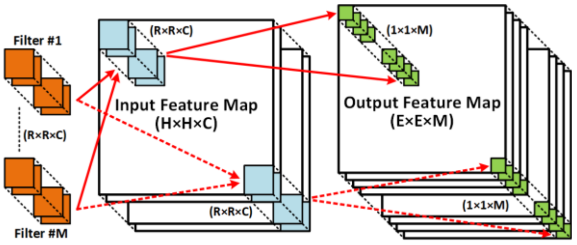


Fig. 18. Latency, throughput, and energy efficiency with reconfigured 1 to 32 bit MAC precisions and simulation corners.



Input Size	Filter Size	Filter #	Output Size	# of Weights	# of MACs
H×H×C	R×R×C	M	E×E×M	R×R×C×M	R^2×C×M×E^2

Fig. 19. Basic configurations of CONV/FC layers of DNNs with key parameters for performance evaluation.

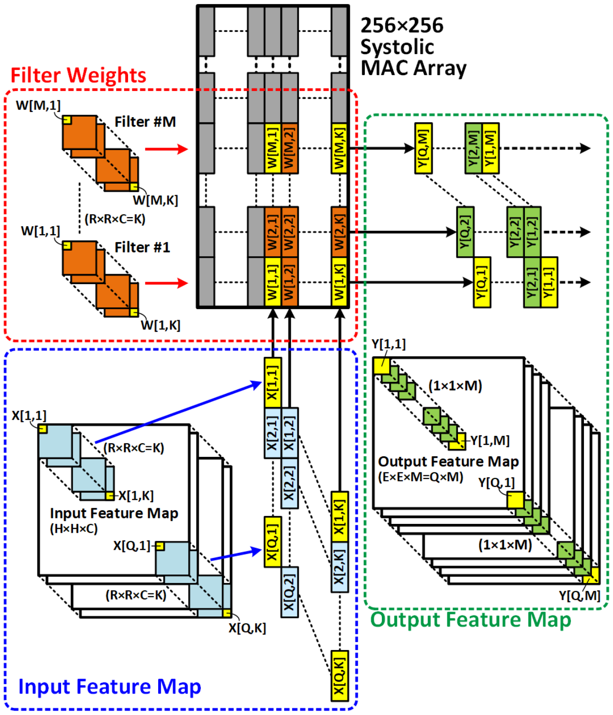


Fig. 20. Mapping filter weights, input and output feature maps into a 256 × 256 systolic array for the pipelined MAC operations in a single CONV/FC layer.

of column MACs in the bitcell array under different bit-precision settings. Based on the configurations, the performance of the 4096 × 256 digital bitcell array is evaluated using transistor-level simulations with 65-nm technology. The simulations are done under the fixed corners, including typical (TT, 1 V, 50 °C), fast (FF, 1.2 V, 0 °C), and slow (SS, 0.8 V, 100 °C).

TABLE IV

SELECTED DNN BENCHMARKS FOR EVALUATION

Complexity	Low	Medium	High
Selected DNNs	LeNet-5	AlexNet	VGG-16
Test Dataset	MNIST	ImageNet	
# of CONV Layers	3	5	13
# of FC Layers	2	3	3
Weight Memory	60K	58.1M	132M
# of MACs	406K	691M	14.4G
Baseline Accuracy	99.2%	83.6% (Top-5)	92.7% (Top-5)

The evaluation results are summarized in Fig. 18. The latency increases as the bit precision is swept from 1 to 32 bit. It ranges from 0.26 μs (at 1 bit) to 21.1 μs (at 32 bit) at the typical corner. The latency also varies from 1.37 μs (at fast corner) to 9.37 μs (at slow corner), which is 2.1× faster or 4.6× slower compared to a typical corner latency of 2.81 μs when the precision is fixed to 8 bit. On the other hand, the throughput decreases as the bit precision increases, and the typical corner throughput ranges from 150 GOPS at 32 bit to 56.1TOPS at 1 bit mode. The energy efficiency also increases as the bit-precision decreases, and the highest efficiency is 393 TOPS/W at 1 bit in the slow corner. In contrast, the lowest is 772 GOPS/W at 32 bit in the fast corner.

B. DNN Benchmarks

Three popular DNN models are selected as benchmarks for the evaluation of the proposed macro when processing DNNs. Table IV shows a list of the selected DNN models, LeNet-5, AlexNet, and VGG-16. LeNet-5 is a DNN with low complexity for testing a simple digit recognition using MNIST data set. The required memory size for storing all the weights of LeNet-5 is 60K, and the total number of MACs is 406 K. VGG-16 is one of the popular DNN models with high complexity for testing more sophisticated image classifications using ImageNet data set. It requires 132 M

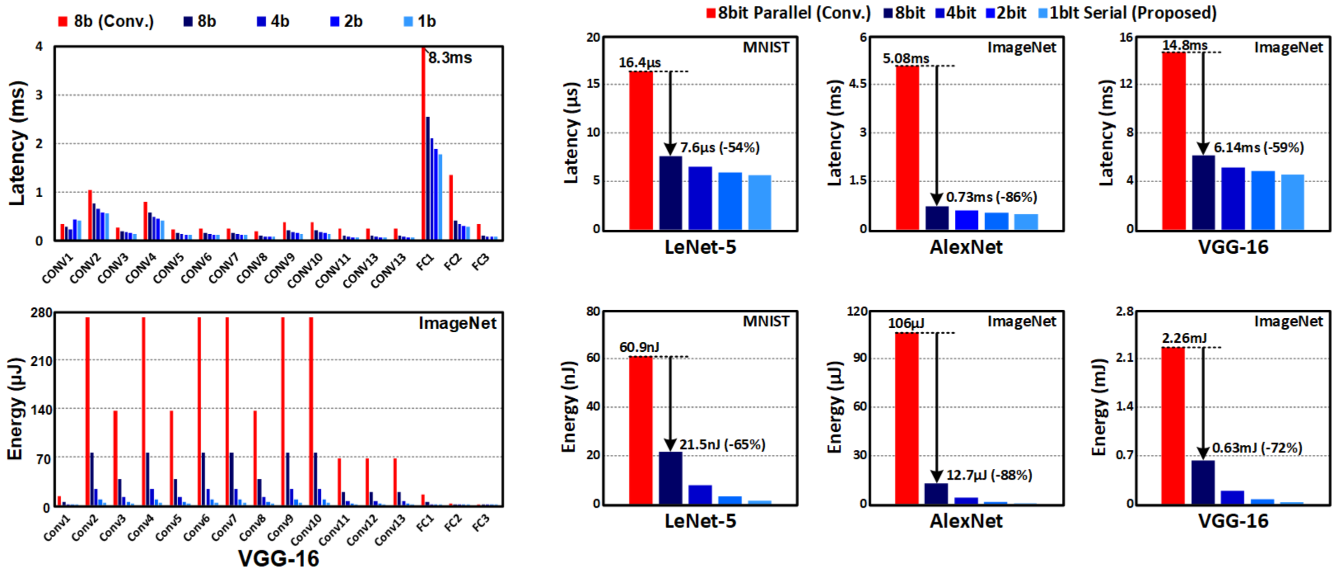


Fig. 21. Evaluation results: latency and energy (per layer, left and classification, right) of LeNet-5, AlexNet, and VGG-16.

weights and 14.4G MAC operations (i.e., 2.200× weights and 35.470× MAC operations versus LeNet-5).

Fig. 19 shows basic configurations of convolutional (CONV) and fully connected (FC) layers of DNN. The size of 3-D filters is R × R × C, where R is the width/height, and C is the number of input channels. The input image size is H × H × C, where H is the width/height of the square shape input image. Note that the width and height of both filters and input–output images are assumed to be the same for simplicity in this evaluation. For a single CONV/FC operation, a 3-D patch of the input image is first selected from the top left corner in the input image (feature) map, as shown in Fig. 20. Then, a pair of the same size 3-D filter and the selected input image patch is used for a dot-product operation (i.e., the accumulation of individual products between a weight from the filter and a pixel from the patch). The dot-product output is then saved as a single pixel of the output 3-D feature map. The same dot-product operations are performed throughout all different filters, and here, the number of the filter is M. Consequently, each M dot-product outputs are saved as a column on the top left corner of the output feature map in Fig. 20. The dot-product operations are performed throughout all the input image patches of the input feature map, and the last patch is shown at the bottom right corner of the input feature map, as shown in Fig. 20. Based on the stride and the input image size, we can calculate the size of the output feature map, which is E × E × C, where E represents the width/height of the output feature map.

C. Mapping DNN Layers

Filters, input feature maps, and output feature maps shown in Fig. 19 are mapped to a 256 × 256 systolic 8-bit MAC array, as shown in Fig. 20. The number of active rows in a MAC array represents M (i.e., the number of filters), and the number of active columns represents K (i.e., the number of weights per filter). Hence, the maximum number of MAC operations in parallel is "M × K." The number of inputs in series (Q) is determined by the size of input feature maps and the stride (or the size of output feature map), as shown in Fig. 20. Note that the same macro can be reused for processing other DNN layers by reprogramming the weights.

D. Performance Evaluation Using DNN Benchmarks

We evaluated the performance of baseline, and the proposed architectures using DNN benchmarks summarized in Table IV. The baseline is a conventional 256 × 256 bit-parallel 8 bit MAC array. In contrast, the proposed has four different bit-precision settings (1, 2, 4, and 8 bit). Fig. 21 shows the evaluation results, including latency and energy per classification using different DNN benchmarks. Regardless of the DNN model, the proposed architecture shows lower latency and energy per classification compared to the baseline. The latency advantage of the proposed design is observed mostly in the fully connected layer operation while the energy advantage is observed across the convolution layers. AlexNet shows the highest gain in both latency and energy reduction (i.e., −86% and −88% versus baseline). Compared to baseline, LeNet-5 and VGG-16 results show 7.6 μs/21.5 nJ and 6.14 ms/0.63 mJ (i.e., latency/energy per classification), respectively, which corresponds to −54%/−65% and −59%/−72% gain. We can also observe the further reduction in both latency and energy as we scale the precision from 8 to 1 bit regardless of the DNN models and data sets.

VII. Conclusion

The excessive energy from OFF-chip memory access and data movement made conventional digital accelerators unfit for the machine learning tasks operated in edge-computing. This has motivated CIM, which has been implemented as a compact memory macro with embedded analog computing circuits in each bitcell. However, analog architectures suffered from critical issues including process variation, data conversion overhead, noise susceptibility, and scalability issues that digital systems seldom exhibit. While digital architectures further evolved to reduce the OFF-chip memory access and data movement, the digital adaptation of CIM was yet to be introduced. This article introduces a novel digital CIM macro architecture, Colonnade, to address the existing digital accelerator-specific concerns while also providing solutions for critical issues that SRAM-based analog CIM macros have suffered. Key features include a fully reconfigurable digital CIM bitcell and the bit-serial CIM macro comprising of a 2-D digital bitcell array and a post-accumulator. Based on the novel reconfigurable and bit-serial CIM architecture, both input and weight bit-precision can be programed from 1 to 16 bit. While bit-serial computation provides area benefit, it degrades latency and throughput. The macro addresses the issue by using parallel operations using 128-inputs. A 65-nm test-chip is fabricated for evaluating throughputs and energy efficiencies for different bit-precisions. The measured throughputs and energy efficiencies of the fabricated 128 × 128 digital bit cell array are 567 GOPS and 117.3 TOPS/W at 1 bit and 6.1 GOPS and 2.06 TOPS/W at 16 bit configuration.

References

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

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

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

[4] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-tile 2.4-Mb in-memory-computing CNN accelerator employing charge-domain compute," *IEEE J. Solid-State Circuits*, vol. 54, no. 6, pp. 1789–1799, Jun. 2019.

[5] S. Yin, Z. Jiang, J.-S. Seo, and M. Seok, "XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks," *IEEE J. Solid-State Circuits*, vol. 55, no. 6, pp. 1733–1743, Jun. 2020.

[6] 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, pp. 490–492.

[7] 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.

[8] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: An energy-efficient deep neural network accelerator with fully variable weight bit precision," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 173–185, Jan. 2019.

[9] S. Yin *et al.*, "An energy-efficient reconfigurable processor for binary-and ternary-weight neural networks with flexible data bit width," *IEEE J. Solid-State Circuits*, vol. 54, no. 4, pp. 1120–1136, Apr. 2019.

[10] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," 2016, *arXiv:1603.05279*. [Online]. Available: http://arxiv.org/abs/1603.05279

[11] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to $+1$ or -1," 2016, *arXiv:1602.02830*. [Online]. Available: http://arxiv.org/abs/1602.02830

[12] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," 2015, *arXiv:1511.00363*. [Online]. Available: http://arxiv.org/abs/1511.00363

[13] I. Hubara, M. Courbariaux, and D. Soudry, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, pp. 1–30, Jan. 2018.

[14] J. Wang *et al.*, "A 28-nm compute SRAM with bit-serial logic/arithmetic operations for programmable in-memory vector computing," *IEEE J. Solid-State Circuits*, vol. 55, no. 1, pp. 76–86, Jan. 2020.

[15] G. K. Chen, R. Kumar, H. E. Sumbul, P. C. Knag, and R. K. Krishnamurthy, "A 4096-neuron 1M-synapse 3.8-pJ/SOP spiking neural network with on-chip STDP learning and sparse weights in 10-nm FinFET CMOS," *IEEE J. Solid-State Circuits*, vol. 54, no. 4, pp. 992–1002, Apr. 2019.

[16] J. Park, J. Lee, and D. Jeon, "7.6 A 65 nm 236.5 nJ/classification neuromorphic processor with 7.5% energy overhead on-chip learning using direct spike-only feedback," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2019, pp. 140–142.

[17] Y. Chen *et al.*, "DaDianNao: A machine-learning supercomputer," in *Proc. 47th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2014, pp. 609–622.

[18] S. Han *et al.*, "EIE: Efficient inference engine on compressed deep neural network," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 243–254.

[19] P. Judd, J. Albericio, T. Hetherington, T. M. Aamodt, and A. Moshovos, "Stripes: Bit-serial deep neural network computing," in *Proc. 49th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2016, pp. 1–12.

[20] H. Sharma *et al.*, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 764–775.

[21] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2014, pp. 10–14.

[22] B. Moons and M. Verhelst, "An energy-efficient precision-scalable ConvNet processor in 40-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 903–914, Apr. 2017.

[23] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," 2018, *arXiv:1806.08342*. [Online]. Available: http://arxiv.org/abs/1806.08342

[24] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Apr. 2018. [Online]. Available: https://openreview.net/forum?id=S1XolQbRW

[25] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless CNNs with low-precision weights," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Apr. 2017. [Online]. Available: https://openreview.net/forum?id=HyQJ-mclg

[26] S. Yin *et al.*, "A high energy efficient reconfigurable hybrid neural network processor for deep learning applications," *IEEE J. Solid-State Circuits*, vol. 53, no. 4, pp. 968–982, Apr. 2018.

[27] Z. Du *et al.*, "ShiDianNao: Shifting vision processing closer to the sensor," in *Proc. ACM/IEEE 42nd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2015, pp. 92–104.

[28] C. Eckert *et al.*, "Neural cache: Bit-serial in-cache acceleration of deep neural networks," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 383–396.

[29] B. Reagen *et al.*, "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 267–278.

[30] D. Shin, J. Lee, J. Lee, and H.-J. Yoo, "14.2 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. 240–242.

[31] 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 Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 246–248.

[32] H. Kim, Q. Chen, and B. Kim, "A 16K SRAM-based mixed-signal in-memory computing macro featuring voltage-mode accumulator and row-by-row ADC," in *Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC)*, Nov. 2019, pp. 35–36.

[33] C. Yu, T. Yoo, T. T.-H. Kim, K. C. T. Chuan, and B. Kim, "A 16K current-based 8T SRAM compute-in-memory macro with decoupled read/write and 1-5bit column ADC," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Mar. 2020, pp. 1–4.

[34] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. ACM/IEEE 44th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2017, pp. 1–12.

[35] H. Kim, Q. Chen, T. Yoo, T. Tae-Hyoung Kim, and B. Kim, "A 1-16b precision reconfigurable digital in-memory computing macro featuring column-MAC architecture and bit-serial computation," in *Proc. IEEE 45th Eur. Solid State Circuits Conf. (ESSCIRC)*, Sep. 2019, pp. 345–348.

[36] 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

[37] 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.

[38] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[39] J.-W. Su *et al.*, "15.2 A 28 nm 64Kb inference-training two-way transpose multibit 6T SRAM compute-in-memory macro for AI edge chips," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 240–242.

[40] X. Si *et al.*, "15.5 A 28 nm 64Kb 6T SRAM computing-in-memory macro with 8b MAC operation for AI edge chips," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 246–248.

[41] W.-S. Khwa *et al.*, "A 65 nm 4Kb 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.

[42] X. Si *et al.*, "24.5 A twin-8T SRAM computation-in-memory macro for multiple-bit CNN-based machine learning," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2019, pp. 396–398.

[43] D. Fujiki, S. Mahlke, and R. Das, "Duality cache for data parallel acceleration," in *Proc. 46th Int. Symp. Comput. Archit. (ISCA)*, Jun. 2019, pp. 397–410.

Hyunjoon Kim (Student Member, IEEE) received the B.A. degree in physics from the Oberlin College, Oberlin, OH, USA, in 2008, and the M.S. degree in electrical engineering from the University of Minnesota, Minneapolis, MN, USA, in 2012. He is currently pursuing the Ph.D. degree in digital compute-in-memory (CIM) circuits and architecture for machine learning applications with Nanyang Technological University, Singapore.

From 2013 to 2018, he was with Gainspan Corporation, San Jose, CA, USA, where he worked as an RF Test Engineer for the IEEE 802.11 Standards. In 2018, he joined Nanyang Technological University. His research interests are memory centric systems, neural network accelerators, and its design methodologies.

Taegeun Yoo (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical and electronics engineering from Chung-Ang University, Seoul, Korea, in 2009, 2011, and 2015 respectively.

From 2015 to 2016, he was with the Chung-Ang University, as a Research Professor. From 2016 to 2019, he was with Nanyang Technological University, Singapore, where he was a Senior Research Fellow and worked on the research of ultralow-power image sensors and emerging memory architectures. He joined Samsung Electronics, Hwaseong, Korea, in 2019, as a Staff Engineer. His research interests include analog mixed-signal ICs and low-power memory architecture.

Dr. Yoo received an Encouragement Award and Silver Award at the Human-Tech Paper Award hosted by Samsung Electronics in 2011 and 2014, respectively. He also received the Silkroad Award at the IEEE International Solid-State Circuits Conference (ISSCC) in 2014.

Tony Tae-Hyoung Kim (Senior Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Korea University, Seoul, South Korea, in 1999 and 2001, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Minnesota, Minneapolis, MN, USA, in 2009.

From 2001 to 2005, he was with Samsung Electronics, Hwasung, South Korea, where he performed the research on the design of high-speed SRAM memories, clock generators, and IO interface circuits. From 2007 to 2009, he was with the IBM T.J. Watson Research Center, Yorktown Heights, NY, USA, and Broadcom Corporation, Edina, MN, where he performed the research on circuit reliability, low-power SRAM, and battery-backed memory design. In 2009, he joined Nanyang Technological University, Singapore, where he is currently an Associate Professor. He has authored or coauthored over 160 journal articles and conference papers and holds 17 U.S. and Korean patents registered. His current research interests include low-power and high-performance digital, mixed-mode, and memory circuit design, ultralow-voltage circuits and systems design, variation and aging-tolerant circuits and systems, and circuit techniques for 3-D ICs.

Dr. Kim received the Best Demo Award at APCCAS2016, the Low Power Design Contest Award at ISLPED2016, the Best Paper Awards at 2014 and 2011 ISOCC, the AMD/CICC Student Scholarship Award at the IEEE CICC2008, the Departmental Research Fellowship from the University of Minnesota in 2008, the DAC/ISSCC Student Design Contest Award in 2008, the Samsung Humantech Thesis Award in 2008, 2001, and 1999, and the ETRI Journal Paper of the Year Award in 2005. He was the Chair of the IEEE Solid-State Circuits Society Singapore Chapter. He has served on numerous conferences as a Committee Member. He serves as an Associate Editor for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, IEEE ACCESS, and the IEIE *Journal of Semiconductor Technology and Science*.

Bongjin Kim (Senior Member, IEEE) received the B.S. and M.S. degrees from POSTECH, Pohang, South Korea, in 2004 and 2006, respectively, and the Ph.D. degree from the University of Minnesota, Minneapolis, MN, USA, in 2014.

He was with Rambus, Sunnyvale, CA, USA, where he was a Senior Staff Member and worked on the research of high-speed serial link circuits and microarchitectures. He was a Post-Doctoral Research Fellow with Stanford University, Stanford, CA. From 2006 to 2010, he was with Samsung Electronics, Yongin, South Korea, where he performed research on clock generators for high-speed serial links and clock generators. He also worked as a Research Intern with Texas Instruments, Dallas, TX, USA, IBM TJ Watson Research, Yorktown Heights, NY, USA, and Rambus, from 2012 to 2014. He was an Assistant Professor with Nanyang Technological University, Singapore, from 2017 to 2020. He joined the Department of Electrical and Computer Engineering (ECE), University of California at Santa Barbara, Santa Barbara, CA. His current research interests include memory-centric computing devices, circuits, and architectures, hardware accelerators, alternative computing, and mixed-signal circuit design techniques and methodologies.

Dr. Kim was a recipient of the Prestigious Doctoral Dissertation Fellowship Award based on his Ph.D. research works, the International Low Power Design Contest Award from ISLPED, and the Intel/IBM/Catalyst Foundation Award from CICC. His research works appeared at top integrated circuit design and automation conference proceedings and journals including ISSCC, VLSI Symposium, CICC, ESSCIRC, ASSCC, ISLPED, DATE, ICCAD, the IEEE JOURNAL OF SOLID-STATE CIRCUITS (JSSC), and the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS.