A 28-nm 64-kb 31.6-TFLOPS/W Digital-Domain Floating-Point-Computing-Unit and Double-Bit 6T-SRAM Computing-in-Memory Macro for Floating-Point CNNs

An Guo, *Graduate Student Member, IEEE*, Xi Chen, *Graduate Student Member, IEEE*, Fangyuan Dong, Xingyu Pu, Dongqi Li, Jingmin Zhang, Xueshan Dong, Hui Gao, Yiran Zhang, Bo Wang, Jun Yang, *Member, IEEE*, and Xin Si, *Member, IEEE*

***Abstract*— With the rapid advancement of artificial intelligence (AI), computing-in-memory (CIM) structure is proposed to improve energy efficiency (EF). However, previous CIMs often rely on INT8 data types, which pose challenges when addressing more complex networks, larger datasets, and increasingly intricate tasks. This work presents a double-bit 6T static random-access memory (SRAM)-based floating-point CIM macro using: 1) a cell array with double-bitcells (DBcells) and floating-point computing units (FCUs) to improve throughput without the sacrifice of inference accuracy; 2) an FCU with high-bit full-precision multiply cell (HFMC) and low-bit approximate-calculation multiply cell (LAMC) to reduce internal bandwidth and area cost; 3) a CIM macro architecture with FP processing circuits to support both floating-point MAC (FP-MAC) and integer (INT)-multiplication and accumulation (MAC); 4) a new ShareFloatv2 data type to map floating point in CIM array; and 5) a lookup table (LUT)-based Tensorflow training method to improve inference accuracy. A fabricated 28-nm 64-kb digital-domain SRAM-CIM macro achieved the best EF (31.6 TFLOPS/W) and the highest area efficiency (2.05 TFLOPS/mm^2) for FP-MAC with Brain Float16 (BF16) IN/W/OUT on three AI tasks: classification@CIFAR100, detection@COCO, and segmentation@VOC2012.**

***Index Terms*— Artificial intelligence (AI), computing-in-memory (CIM), double-bit 6T static random-access memory (SRAM), floating multiplication and accumulation (MAC) operation, ShareFloat.**

I. Introduction

THE rapid development of convolutional neural networks (CNN) in recent years has brought great challenges to Von Neumann architecture. One of its main drawbacks is the bottleneck that arises due to the data transfer between the processor and the memory. The constant movement of data back and forth between these two components leads to a considerable waste of energy and time, which is called the "memory wall" issue. Computing-in-memory (CIM), which is designed to perform computations in memory directly, has become a promising solution to this problem. CIM can potentially reduce the energy and time required for data transfer and, as a result, increase the overall performance and efficiency of the system.

Fig. 1(a) shows the basic concept of static random-access memory (SRAM)-based CIM structure (SRAM-CIM), which can both realize memory and process element (PE) functions. Thanks to its mature technology, SRAM-based CIM has become the most stable choice compared with nonvolatile memory-based CIM (nvCIM). Previous SRAM-CIM [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18] mainly focused on integer (INT)-multiplication and accumulation (MAC). Wu et al. [18] presented a time-domain CIM macro to perform 4-/4- and 8-/8-b MAC. Time-domain CIM has been proved to be a possible way to realize high throughput convolution. Wang et al. [6] added horizontal-weight-shift and vertical-feature-shift operations in current-domain CIM to accelerate depth-wise neural networks. Current-domain CIM shows great potential when applied to lightweight neural networks for its high flexibility. Chen et al. [8] use a delta–sigma structure in charge-domain CIM to reuse the redundant input features. The temporary-store characteristic could save lots of energy consumption in toggle-domination applications. Mori et al. [5] presented a 4-nm digital CIM macro to support bit-width flexibility and simultaneous MAC and weight update [5]. For its high reliability and technology scalability, a digital CIM can achieve both high accuracy and energy efficiency (EF).

However, most SRAM-CIMs that previously used digital or analog in-memory computing cannot effectively support floating-point MACs (FP-MACs): e.g., Brain Float16 (BF16) datatype. To realize more complex artificial intelligence (AI) tasks, such as detection and segmentation shown in Fig. 1(b), and to support on-chip training for better inference accuracy, FP-MAC operations with high-EF are required.

Manuscript received 18 October 2023; revised 25 December 2023 and 2 February 2024; accepted 2 March 2024. Date of publication 25 March 2024; date of current version 27 August 2024. This article was approved by Associate Editor Meng-Fan Chang. This work was supported in part by the National Science and Technology Major Project under Grant 2022ZD0118902, in part by the National Natural Science Foundation of China under Grant 92264203 and Grant 62204036, in part by the Key Research and Development Program of Jiangsu Province under Grant BE2023020-1, in part by the Postgraduate Research and Practice Innovation Program of Jiangsu Province under Grant KYCX23_0344, and in part by the Fundamental Research Funds for the Central Universities under Grant 2242022k60009. *(Corresponding author: Xin Si.)*

The authors are with the School of Integrated Circuits, Southeast University, Nanjing 210096, China (e-mail: xinsi@seu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSSC.2024.3375359.

Digital Object Identifier 10.1109/JSSC.2024.3375359

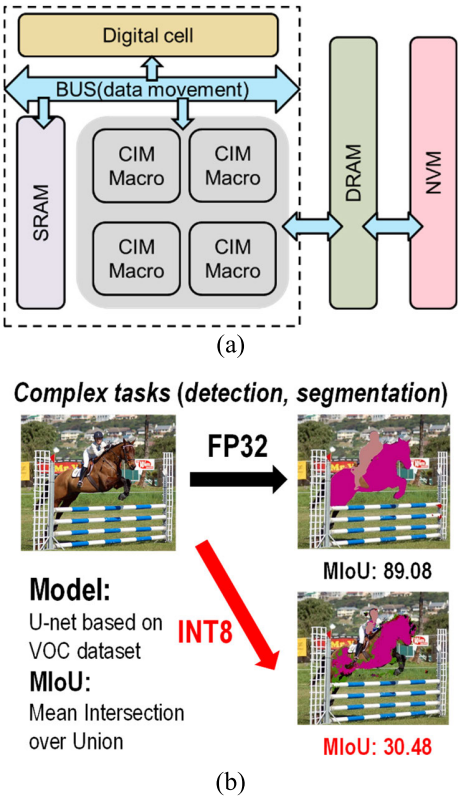


Fig. 1. (a) Concept of SRAM-CIM. (b) Motivation of FP CIM macro.

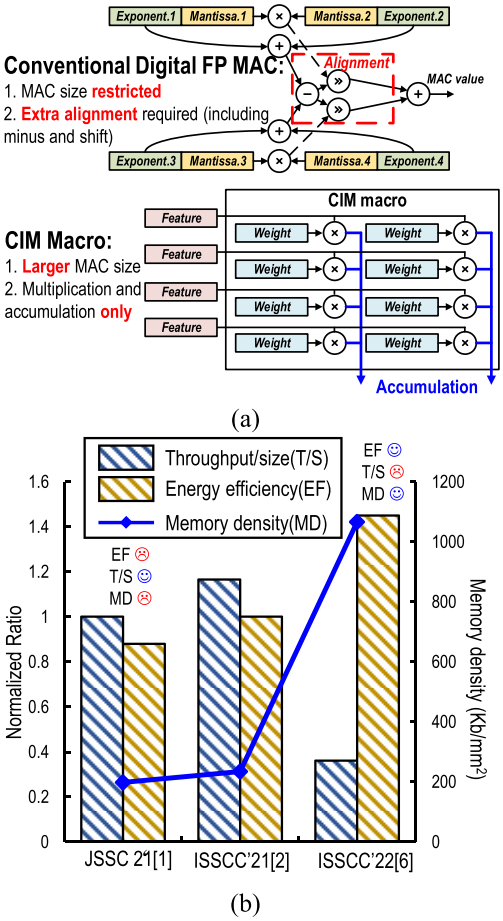


Fig. 2. Challenges in FP SRAM-CIM structure. (a) Inconsistency between the shift alignment of conventional digital FP-MACs and the structured mapping of most SRAM-CIMs. (b) Results in a more difficult tradeoff between T/S, EF, and MD.

In this work, we proposed a 28-nm 64-kb 31.6-TFLOPS/W digital-domain floating-point-computing-unit (FCU) and double-bitcell (DBcell) CIM macro for a variety of floating-point CNNs [19].

The remainder of this article is organized as follows. Section II discusses the challenges in designing FP SRAM-CIM macros. Section III outlines the structure of the proposed DBcell SRAM-CIM macro. Section IV presents the floating-point designs used in the proposed SRAM-CIM macro. Section V describes the performance and measured results. Section VI concludes this article.

II. Challenges of Floating-Point SRAM-CIM

To carry out complex tasks, such as detection and segmentation, FP-MAC operations should be performed. However, the designs of FP SRAM-CIM face challenges in: 1) inconsistency between the shift-alignment of conventional digital FP-MACs and the structured mapping of most SRAM-CIMs and 2) results in a more difficult tradeoff between throughput/memory size (T/S), EF, and memory density (MD).

A. Inconsistency Between Conventional Digital FP-MACs and CIM Macro

In numerous previous SRAM-based CIM works, MAC operations are performed in the whole array with feature reuses of two directions, vertical-input-channel-wise and horizontal-output-channel-wise feature reuse. This structure could ensure a big MAC size and a high EF. However, this structure can only perform MAC, which limits CIM's flexibility to FP applications, as shown in Fig. 2(a).

While for traditional digital FP accelerators, MAC operations are more complex. FP data are composed of four parts, exponent (EXP), mantissa, sign, and one hidden bit as follows: $m_r m_{r-1} \cdots m_0 * 2^{e_t e_t \cdots e_0} * (-1)^s$, where m_n and $[r:0]$ are the nth bit and length of mantissa, e_n and $[t:0]$ are the nth bit and length of EXP, s is the sign bit. For an $f_0^*\ w_0 + f_1^*\ w_1$ situation, MAC operation can be split into four steps.

1) *Multiplication:* In this step, there will be one EXP add and one mantissa multiplies for each FP multiplication to get a partial EXP and mantissa product (e_p and m_p).

2) *Alignment*: Two e_ps will compare with each other to get a subtraction and a shared EXP. The other e_p's corresponding to m_p will be shifted recording to the subtraction.

3) *Accumulation*: Once the partial mantissas have the same bit weight, they can be added up. The accumulation mantissa and shared EXP compose a near-FP result.

4) *Recover*: The near-FP result needs to be reshaped to become the standard IEEE floating-point data by EXP and mantissa shift.

Comparing INT-MAC and FP-MAC, the major differences lie in three parts: 1) extra EXP add during multiplication; 2) extra alignment; and 3) recover to standard form. These

features lead to inconsistency between the shift alignment of conventional digital FP-MACs and the structured mapping of most SRAM-CIMs.

To overcome this challenge, this work proposes a CIM-macro architecture with FP processing circuits to support both FP-MAC and INT-MAC operations and a cell array with DBcells and FCUs.

B. Tradeoff Among Throughput/Size, Memory Density, and Energy Efficiency

Throughput per bit (TPB), MD, and EF are the three key features of CIM works, as shown in Fig. 2(b). Numerous previous CIM works employ novel structures and circuits to improve one or two key features.

Si et al. [1] present a local-computing-cell (LCC)-based 6T SRAM-CIM to expand the sensing margin and improve the readout accuracy. This structure could improve MD, but sacrifice TPB. Su et al. [2] changed Si's work into charge-domain and hybrid analog-to-digital converter (ADC) to save the total area. This work achieves better performances on all three features. However, suffering from its costly readout circuit, analog CIM faces lots of challenges in making a balance between TPB, MD, and EF. Yan et al. [17] design a dynamic-logic-based digital CIM to move large-scale adder trees outside the SRAM array. The proposed digital CIM achieves the highest MD and high EF, but rather low TPB due to its low calculation-store ratio. In floating-point CIM, this tradeoff will be even more serious for the existence of an extra EXP processor.

To achieve a better tradeoff among TPB, MD, and EF, an FCU with a high-bit full-precision multiply cell (HFMC) and a low-bit approximate-calculation multiply cell (LAMC) is proposed.

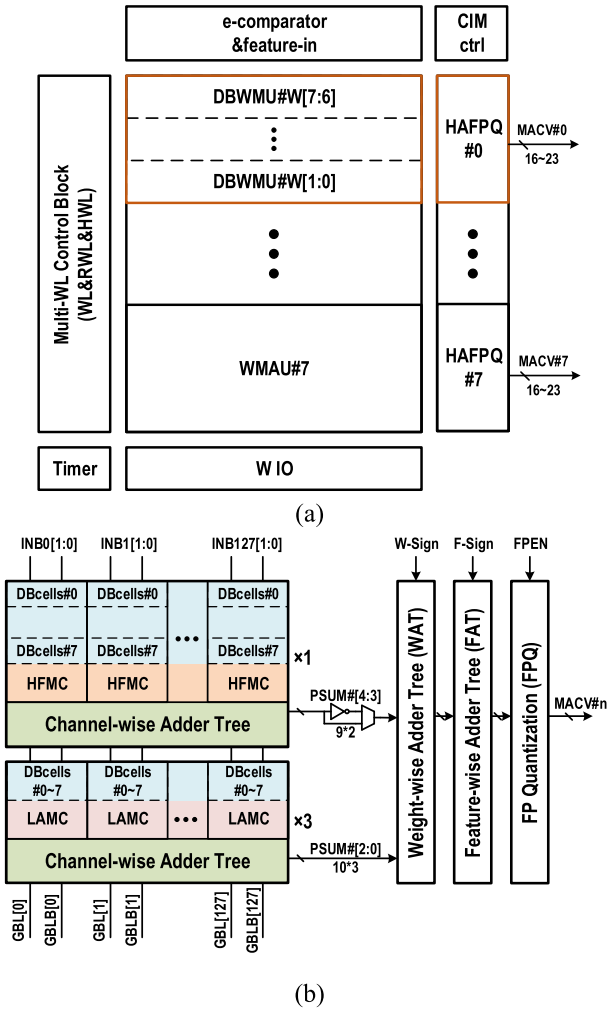


Fig. 3. (a) Overall structure of DBcells and FCU-based SRAM-CIM macro. (b) Detailed structure of WMAU, HA, and FPQ.

III. Proposed DBcells and FCU-Based CIM Macro

A. Macro Structure

Fig. 3(a) illustrates the overall structure proposed DBcells- + FCU-based SRAM-CIM macro. This macro comprises a 512 × 128 b in-SRAM-computing array, a multi-WL control driver, an EXP comparator (e-comparator), a feature-input buffer, a hybrid adder tree (HA), and an FP quantization block (FPQ). The 512 × 128 b in-SRAM-computing array is split into eight word-wise MAC units (WMAUs). Each WMAU contains four dual-bit-wise MAC units (DBWMU).

As shown in Fig. 3(b), a DBWMU is composed of 8 × 128 DBcells, 128 FCUs, and a channel-wise adder tree (CAT). Each DBcell comprises two compact split-WL 6T SRAM bitcells [4]. The HA is composed of two kinds of adder trees: weight-wise and feature-wise adder trees (WAT and FAT). Both WAT and FAT can be configured to support signed MAC and unsigned MAC modes based on the sign information. The proposed structure supports two bit-precisions: BF16IN–BF16W–BF16OUT and 8bIN–8bW–23bOUT.

B. Proposed Cell Array With DBcells, HFMC, and LAMC

To improve throughput with almost no inference accuracy drop, a cell array with DBcells and FCU is proposed. An FCU contains HFMC and LAMC to reduce the internal bandwidth and area cost, as shown in Fig. 4. The proposed cell array has two operating modes: 1) computing mode with multibit MAC operations and 2) normal SRAM mode supporting write operations. Please note that computing mode can function as SRAM read mode too.

In memory mode, as shown in Fig. 4, HWL is activated and global bit line (GBL)/global bit line bar (GBLB) is connected to LBL/LBLB. The odd cell of DBcells stores original data and even cell stores the reverse. The writing process is conducted in three steps: 1) LBL and GBL recharge and GBL data initial; 2) HWL activated to connect LBL and GBL; and 3) WL and RWL activated to connect the bitcell-store node with LBL. During memory mode, WL[n] and RWL[n] are decoded in the same way and only writing bitcell is selected. To be noticed that LBL will be precharged when weights are changed as shown in Fig. 5. This mode will have extra power overhead due to the recharging glitches. However, the macro EF will only decrease by 5.2%, because all weights will be used many times (≥5 times) that precharge hardly matters.

In computing mode, HWL is inactivated to isolate LBL from GBL. In this way, LBL could carry stored weight and

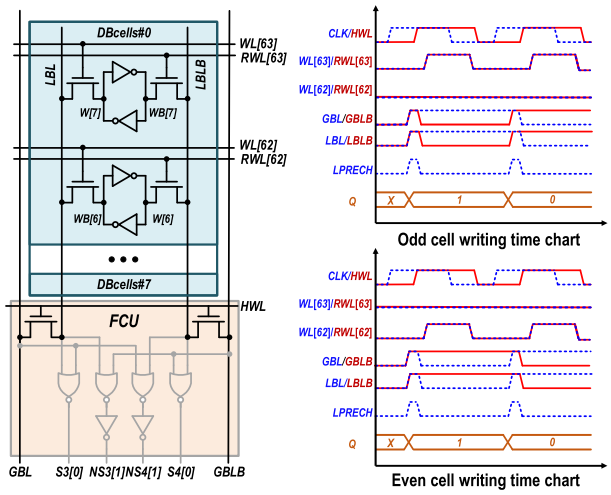


Fig. 4. Writing operation of the proposed array with DBcells and FCU.

GBL carry the input feature. The proposed FCU can conduct multiplication of 1- or 2-bit IN and 2-bit W, which is classified into two kinds: HFMC and LAMC, as shown in Fig. 5. HFMC performs MAC operations of MSB and MSB-1 (W [7] and W [6] in this work), while LAMC performs MAC operations of MSB-2 to LSB (W [5] to W[0] in this work).

As shown in Fig. 5(a), HFMC could conduct two 2-bit IN and 1-bit W multiplication with full precision. The full precision multiplication process is conducted in three steps.

1) WL[n] and RWL[$n+1$] activated, RWL[n] and WL[$n+1$] inactivated to load W[m] on LBL and WB[$m+1$] on LBLB, where $n$and m are even numbers.

2) Input drivers load IN[$r+1$] on GBL and IN[r] on GBLB. The voltage on GBL is reversed to input data too.

3) Two partial products are produced through a NOR gate and the high bit is reversed to fit in an adder tree with a reversed-polarity adder.

For LAMC, as shown in Fig. 5(b), a 2-bit IN and 2-bit W approximate multiplication is conducted. The approximate calculation process is also conducted in three steps, where the first two are the same as HFMC, except the last one: 2-bit IN and 2-bit W pass through an approximate multiplier to produce a partial sum with little possibility (1/16 without considering neural network (NN) distribution) to error. The approximate calculation is to change 11×11 results from 9 to 7 to simplify the multiply process and decrease circuit costs.

C. Proposed Channel-Wise, Weight-Wise, and Feature-Wise Adder Tree

The proposed cell array with FCU only conducts multiplication of 1- or 2-bit IN and 2-bit W. Therefore, an extra adder tree must be introduced to finish the whole calculation. This article presents three kinds of adder trees to finish a signed (or unsigned) 9-bit (or 8-bit) IN and 8-bit W multiplication, as shown in Fig. 6.

The first kind of adder tree is a CAT. The proposed CAT is inserted into the cell array to minimize the data transfer path and introduces ISSCC2022-like reversed-polarity full-adder [16] to increase calculation speed. For HFMC and LAMC, CAT contains seven-level adder trees to accelerate 128-channel results into 9-bit full precision partial sum (PSUM) or 10-bit approximate PSUM. Please note that no sign information needs to be considered during this phase. For one output channel, there will be two 9-bit full precision PSUMs and three 10-bit approximate PSUMs.

The second kind of adder tree is a WAT. The five PSUMs produced by CAT have their own bit-signification related to their original W. For unsigned mode, the bit signification will be 2^0, 2^2, 2^4, 2^5, and 2^6, while for signed mode, it will be 2^0, 2^2, 2^4, 2^5, and -2^6. The only difference between these two modes is whether the highest PSUM needs to be inversed and add 1 at the lowest bit. A sign-extension adder is introduced to complete this work. For one output channel, there will be four (or five if the feature is signed) 18-bit signed/unsigned PSUMs in four (or five) cycles. All "sign"'s in this paragraph refer to weight sign.

The last kind of adder tree is FAT. FAT is a time-scale adder tree due to bit serial input. For unsigned mode, feature-wise accumulation is finished in four cycles: 1) during the first cycle, register (REG) is cleared, and the zero in REG is accumulated with the highest PSUM, then rewritten into REG; 2) during the second cycle, the data in REG is left-shifted by two bits and accumulated with the next PSUM, then rewritten into REG; and 3) the rest two cycles are just like the second cycle. While for signed mode, most cycles are the same except two parts: 1) REG is cleared to 0 and accumulated with the reversed highest PSUM and an extra one as carry-in and 2) the second cycle's shifting up is one bit. For one output channel, there will be only one 25-bit signed/unsigned partial result. All "sign"s in this paragraph refer to feature signs.

IV. Floating-Point Designs Used in the Proposed SRAM-CIM Macro

To achieve FP-MAC in SRAM-CIM macro, this article proposed four key features: 1) a global-floating-point, local-fixed-point ShareFloatv2 data form to map FP-MAC in CIM macro; 2) a configurable architecture to support both INT/FP MAC; 3) a data flow to cover all FP pre- and post-process; and 4) a lookup table (LUT)-based Tensorflow simulation method to estimate influence of FP + approximate digital CIM on CNN to improve inference accuracy.

A. Global-Floating-Point and Local-Fixed-Point ShareFloatv2 Data Form

Traditional FP data has a problem to map in CIM due to inconsistency between the shift alignment of conventional digital FP-MACs and the structured mapping of most SRAM-CIMs. Previous work [26] presented a ShareFloat data form to realize CIM mapping. This artoc;e improves this work and presents a new data form: ShareFloatv2.

Fig. 7 shows the transform from BF16 to ShareFloatv2 in CNN. Taking the weight of one layer for example, a group of data are selected to form a shared cluster (SC). Usually, the SC size is the same as CIM accumulation size. All data in this SC are split into two parts: an EXP part and a sign & mantissa (S&MAN) part. The EXP will be gathered to find the maximum, which will be the shared EXP (e_s). Other

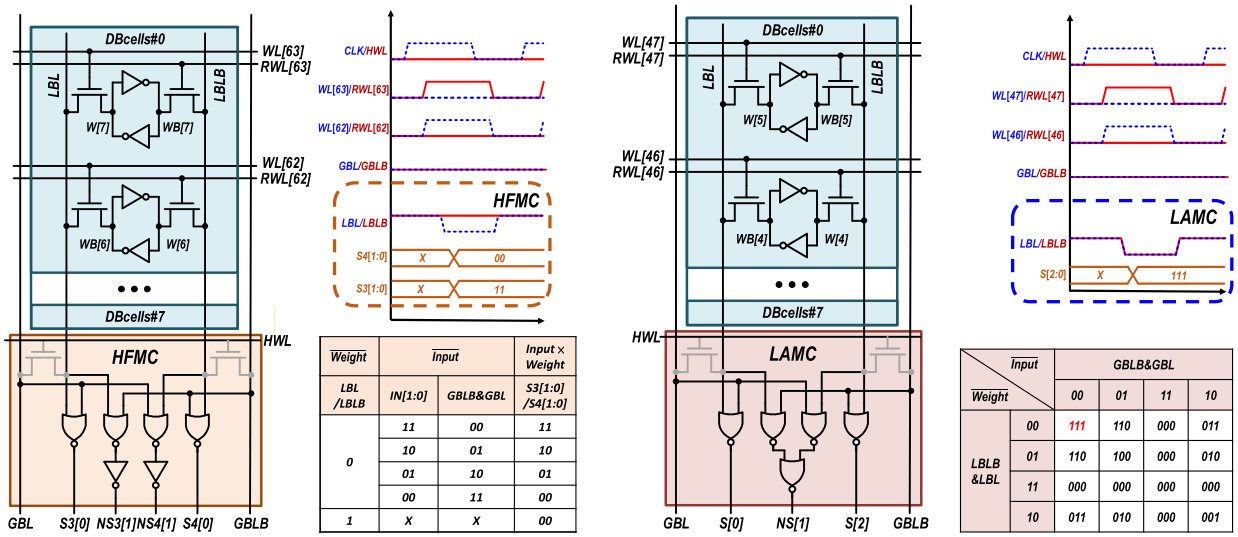


Fig. 5. (a) HFMC and DBcells. (b) LAMC and DBcells.

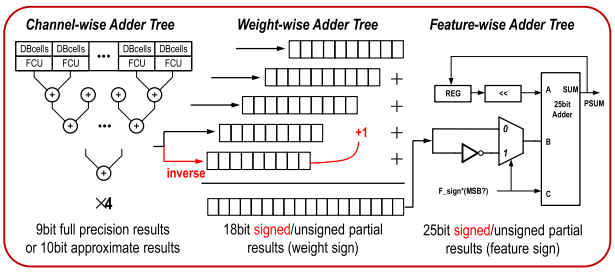


Fig. 6. Proposed channel-wise, weight-wise, and feature-wise adder tree.

EXP will be subtracted to get their own shift index (S_n). The S&MAN and hidden bit will be transformed into 2's complement (2'c) and aligned to the same bit signification by shifting the S_n bit. The extra bits which exceed the 8-bit cut window will be discarded. If the sign bit is 0, the supplement bit will be filled with 0, and if the sign bit is 1, it will be filled with 1 instead. The shared EXP and 2'c mantissa form proposed ShareFloatv2. The EXP sharing range can be changed to pursue better EF or higher accuracy. But no significant improvement will happen until the range is set at 32 according to the experiment.

Compared with the standard IEEE data form, a ShareFloatv2 will have a shared 8-bit EXP and 8-bit 2'c MAN. The shared 8-bit EXP is a global floating-point index to find where this group of data locate, which reflects their global characteristic. The 8-bit 2'c MAN is a global fixed point which contains 1-bit INT and 7-bit fraction. Its job is to distinguish this data from these shared groups. Compared with ShareFloat [26], the new version mainly changed shared EXP from mean to max and decreased fixed numbers from 13 to 8 bit. This will bring a 1.6× area cost with almost the same accuracy.

B. Proposed Configurable Architecture to Support Both INT/FP MAC

To support both FP/INT MAC, this article proposed a configurable CIM architecture. Proposed CIM can function in two options: INT or FP and signed or unsigned.

In FP mode, as shown Fig. 8(a), all modules are activated. While for INT mode, FP modules closed, only cell array and HA are activated to conduct INT MAC. Fig. 8(b) shows a high-level time chart of the proposed CIM macro. This CIM macro will function as follows.

1) 1) FP mode.
   a) If unsigned, all calculations will be finished in four cycles with 2-bit input per cycle. Then WMAU conducts a 128-channel 2-bit W, 2-bit IN MAC, a WAT to add results of four banks to produce a 128-channel 8-bit W, 2-bit IN PSUMs, an FAT to add PSUMs of four cycles to generate an INT result and an FPQ to transform this result into BF16 form.
   b) If signed, MAC period will be five cycles and all adder will be sign extended. The input feature will be split into 1, 2, 2, 2, and 2 bits.
2) INT mode.
   a) If unsigned, all processes are same as FP except FPQ.
   b) If signed, input feature will be split into 1, 1, 2, 2, and 2 bits. A decrease in utilization could happen for supporting INT and FP mode MAC operation. The proposed chip could achieve utilization of 96.3% in INT mode and 100% in FP mode. If we adjust the bit assignment to IN[7:4] and W[7:4] of HFMC and IN[3:0] and W[3:0] of LAMC, power consumption will increase by 1.3×, area cost will increase by 1.1× and speed will decrease by 1.2×. While the accuracy only increases by 0.2% (ResNet@CIFAR100).

C. Proposed Data Flow to Cover All FP Pre- and Post-Processes

To complete an FP MAC in CIM, lots of pre- and post-processes will be introduced. Fig. 9 shows a one-out-channel mapping strategy, containing e-comparator, feature

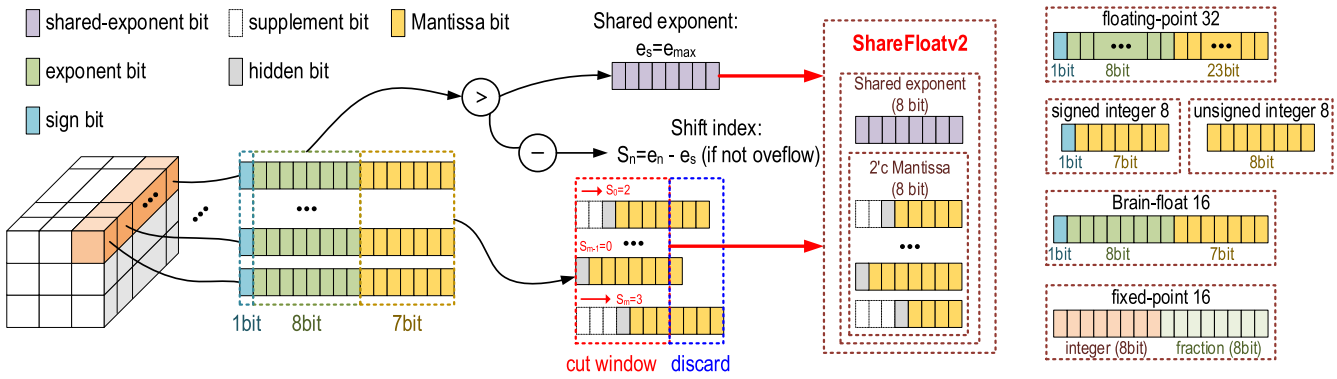


Fig. 7. Transform from BF16 to ShareFloatv2 in CNN and comparison between ShareFloatv2 and other common data forms.

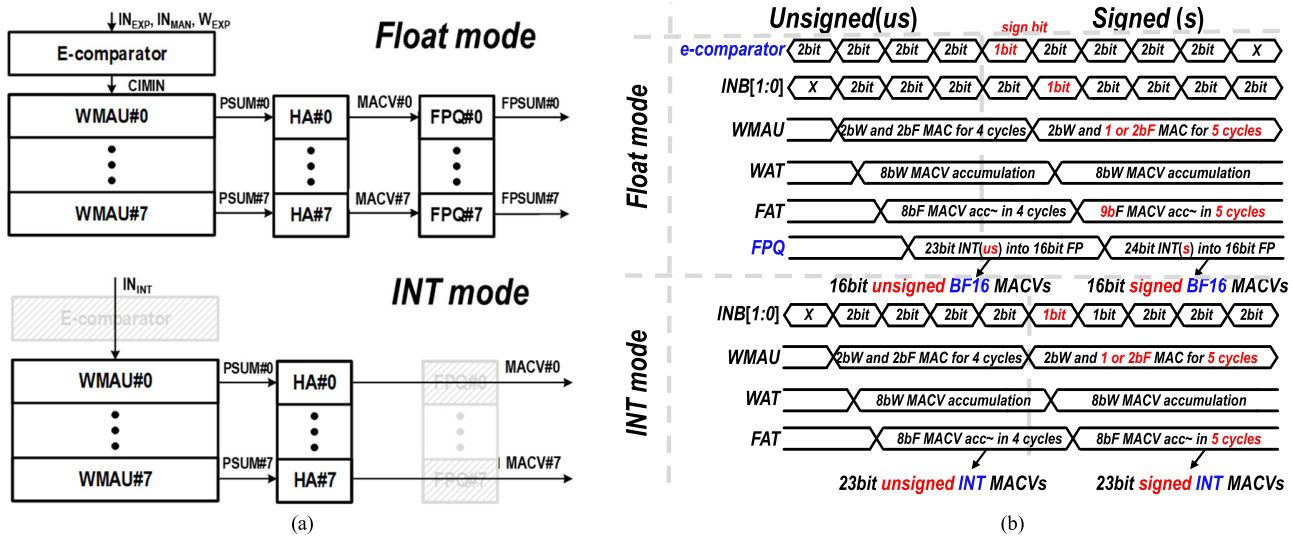


Fig. 8. (a) Activated circuit under both FP mode and INT mode. (b) High-level time chart of FP/INT mode and signed/unsigned mode.

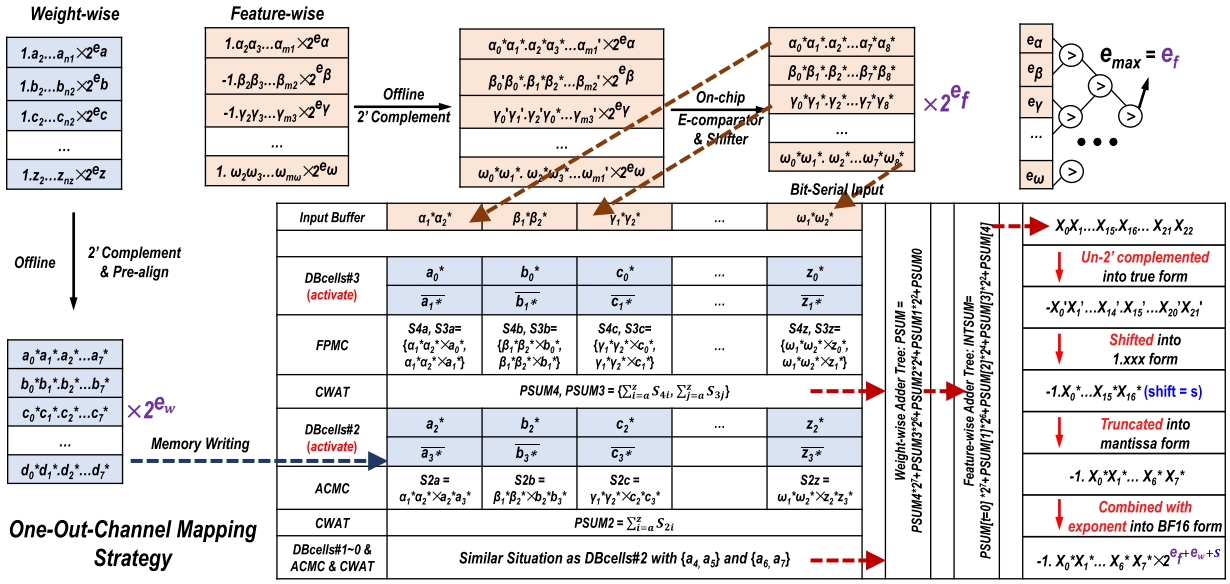


Fig. 9. Proposed data flow to cover all FP pre- and post-processes.

input buffer, one WMAU, and one HAFPQ, which is enough to understand this flow.

FP MAC will be conducted in four steps.

1) Weight offline process and CIM store. All weights in one-in-channel are transformed into ShareFloatv2 and written into cell area with odd cell saving high

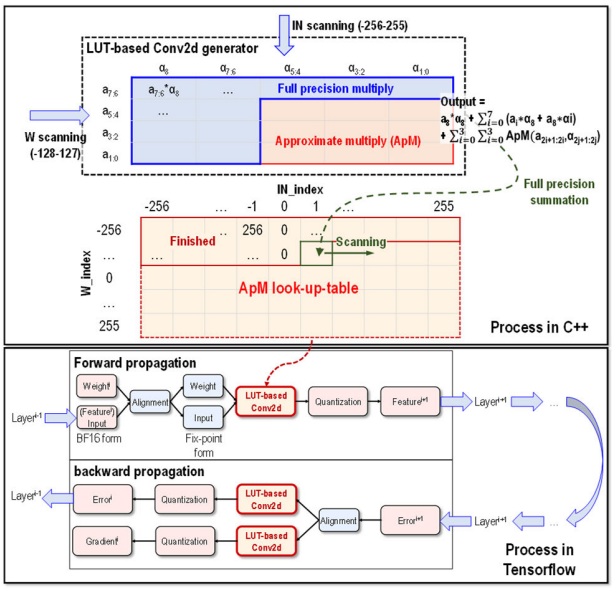


Fig. 10. LUT-training process.

bit, even cell saving low bit. Their shared EXP will be e_w.

2) Feature offline process and on-chip alignment. All features will be offline transformed into 2'c form and compared by e-comparator to get the shared EXP e_f. Actually, weights and features are both transformed into ShareFloatv2, just in an only-offline way or offline and on-chip way.

3) INT-like MAC operations, which have been discussed in Section III-C.

4) FP quantization. INT MAC results will be un-2'complemented into true form, then shifted into "1.xxx" form, where shift index will be presented by s, then truncated into 8-bit mantissa and one hidden bit. Finally, a BF16 result is produced with a former mantissa and an EXP which is equal to $e_f + e_w + s$. To be noticed that his process will not incur overflow with BF16 output. The reason is we reserve all data precision until truncating "1.xxx" form into mantissa form, while this step only abandons small bits. Weights can be used for hundreds of times, so the pre-process is better to be solved offline for the next task. While features are used in one task, which means the on-chip 2's complement implementation could possibly improve efficiency. A digital on-chip 2's complement is simulated to verify area, energy, and performance overhead. The 128 features' preprocess cost 0.007 mm^2 and 0.66 mW with 100-MHz CLK. In floating-point format, positive zero and negative zero exist simultaneously. However, it aims more at division than MAC. To simplify the calculation, the proposed chip transfers $+0$ and -0 into 00000000.

D. Proposed LUT-Based Tensorflow Training Method

To further improve the inference accuracy of the proposed chip, this article introduces an LUT-based Tensorflow training method, as shown in Fig. 10. This method includes two key features.

1) An LUT-based Conv2d generator to model the real output of the proposed chip. The LUT-based Conv2d generator aims at fixed point approximate calculation by scanning all IN and W to generate an LUT-based Conv2d layer with corresponding results of CIM to replace raw Conv2d in forward and backward propagation.

2) A Tensorflow alignment layer to simulate ShareFloatv2 data mapping. All FP weight and features are converted into ShareFloatv2 form and the mantissa part goes through LUT-based Conv2d.

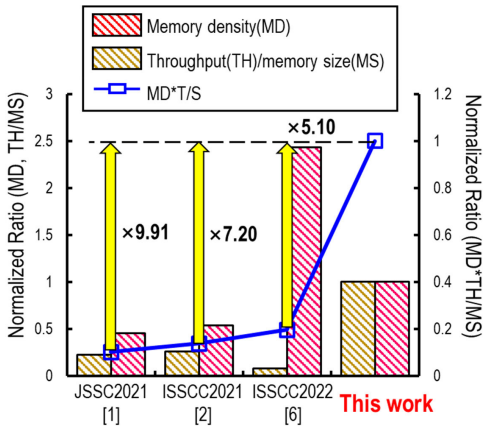


Fig. 11. Simulation MD, TH/MS, and MD*TH/MS of the proposed macro.

V. Performance and Measurement Results

A. Performance of the Proposed Schemes

Fig. 11 shows the comparison of the MD, TH/MS, and a specific index, MD∗TH/MS (which is area efficiency for most chips) of this work with previous works. This article achieves the best TH/MS compared with previous works and a better balance between throughput and MD.

Fig. 12 presents the approximate calculation gain of the proposed work. The internal bandwidth is referred to the bandwidth needs between different levels of adder trees and the peripheral overhead is defined as the transistor number of MAC units and post-process circuits. This work could save 23% internal bandwidth and 73% and transistors with LAMC. Thanks to DBcells, the proposed macro could improve the throughput by two times. The internal bandwidth is the total bandwidth of every CAT level, which will influence not only adder tree layout routing but also a number of FAs. It will have a great impact on system area efficiency and EF. Fig. 13 shows the area and power comparison between HFMC and LAMC. LAMC can achieve 1.48× area gain and 1.40× power gain compared with HFMC.

Fig. 14 shows the EF comparison between this work and some previous works. Compared with previous works, the proposed macro can achieve 1.61× − 3.83× EF under INT8 precision. In addition, this work achieves 1.15×–2.75× higher EF for BF16 MAC operations, compared with previous INT8 implementations.

Fig. 15 shows the 11 × 11 distribution of several neural networks of three different tasks. For most CNNs of classification and segmentation, 11 × 11 only exists for less than 2% of cases, which will have rather little influence on

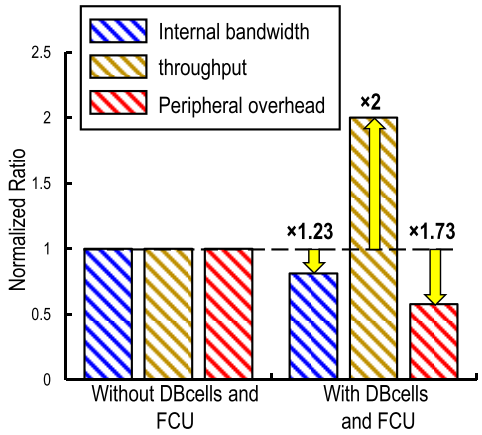


Fig. 12. Approximate calculation and DBcells gain in internal bandwidth, throughput, and peripheral overhead.

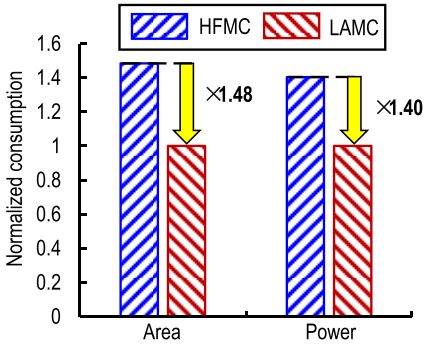


Fig. 13. Approximate calculation and DBcells gain in area and power consumption.

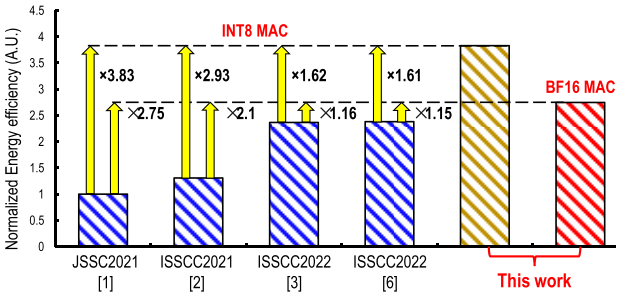


Fig. 14. Simulated EF compared with previous work.

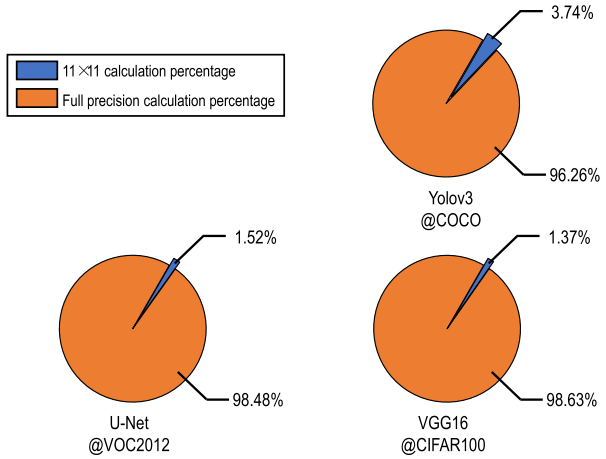


Fig. 15. 11 × 11 calculation percentage of different tasks.

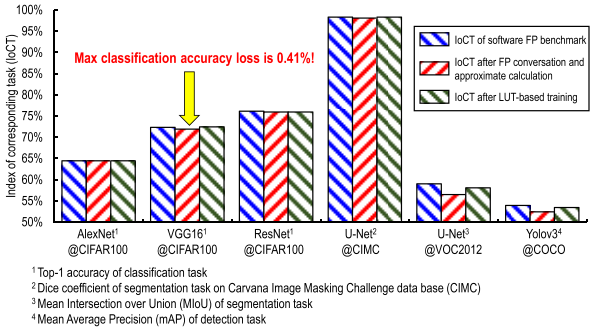


Fig. 16. Measured accuracy under different tasks.

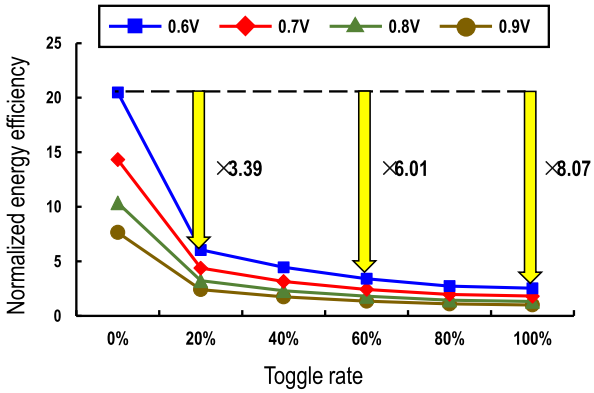


Fig. 17. Measured EF under different toggle sparsities (feature sparsity@50% and weight sparsity@0%).

inference accuracy. For the detection task, the percentage comes to 3.74%, which is the highest of analyzed work, but the measured accuracy loss is still acceptable, which will be discussed in Section V-B.

B. Measurement Results

This work implemented a test chip with a 64-kb 6T SRAM-CIM macro using a 28-nm CMOS logic process. This test chip employed the DFF-based path-delay exclusion approach to enable the extraction of access times (T_{AC}) for the proposed 6T SRAM-CIM macro.

Fig. 16 presents a comparison of the index of different tasks between the FP benchmark and measurement results. The biggest accuracy loss is 0.41% of VGG16 on CIFAR100, but it is not the largest 11 × 11 percentage. The reason for this situation is different network has different error tolerance. For those error-tolerant CNNs, the proposed chip can achieve higher accuracy.

Figs. 17 and 18 show the EF under different supply voltages, feature toggle rates, and feature sparsity. Figs. 17 and 18 shows tested under 0% weight sparsity. (All weights are 1.) Toggle rate is defined as how much input bit is changed per cycle and sparsity is defined as how much 1 exists in total. Fig. 17 shows the tested under 50% feature sparsity, and Fig. 18 shows the tested under a 20% toggle rate. The EF could achieve 8.07× EF compared with a 100% toggle rate. While under the same toggle rate, the EF remains almost the same in

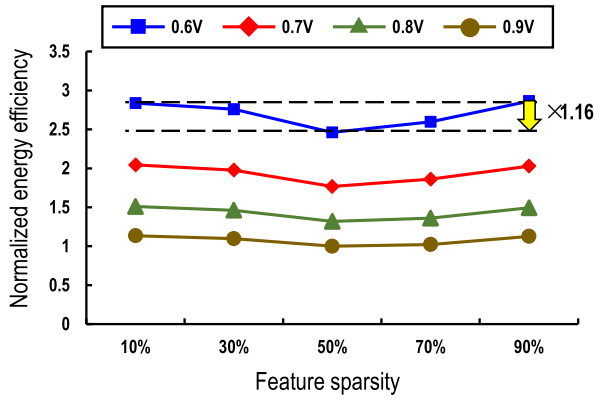


Fig. 18. Measured EF under different feature sparsities (toggle rate@20% and weight sparsity@0%).

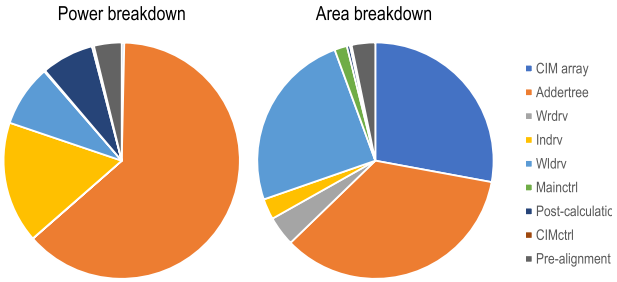


Fig. 19. Measured power breakdown (feature toggle rate@30% and weight sparsity@0%) and area breakdown.

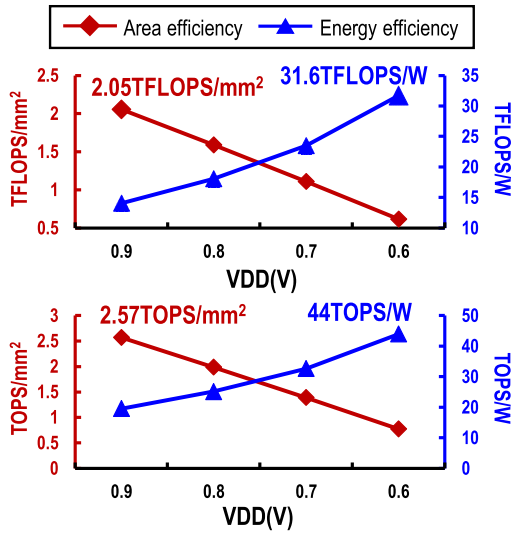


Fig. 20. Measured EF and area efficiency of different voltages under BF16 mode and INT8 mode.

different sparsity. The major reason for such a situation is the partial-store ability of the proposed CIM. Actually, this is an ability of most digital CIM, which means that the toggle rate is a more important benchmark compared with sparsity. To be noticed that data in Figs. 17 and 18 are a designed pattern rather than a real NN workload to analyze whether digital CIM is a sparsity- or toggle-domination case. In this designed pattern, all weights are fixed at 1 to avoid unexpected influence.

TABLE I
CHIP SUMMARY TABLE

CHIP SUMMARY			
Technology		28nm CMOS	
Macro size		64Kb	
Macro area		706.96um×206.76um =0.146mm²	
Input precision(bit)		8	BF16
Weight precision(bit)		8	BF16
Output precision(bit)		23	BF16
Number of channels		128	128
Supply voltage(V)		0.6-0.9	
Access time(ns)		5.5	6.8
Energy efficiency[1]		19.5-44 TOPS/W	14.04-31.6 TFLOPS/W
Neural network index of different AI tasks	Inference accuracy[2] (@CIFAR100)	74.51%	75.94%
	mAP[3] (@COCO)	-	52.40%
	MIoU[4] (@VOC2012)	-	56.49%

[1]Measured in 0% weight sparsity, 50% feature sparsity and 30% feature toggle rate.
[2]Using ResNet18 model and the software baseline is 76.11%.
[3]Using Yolov3 model and the software baseline is 53.91%.
[4] Using U-Net model and the software baseline is 59.01%.

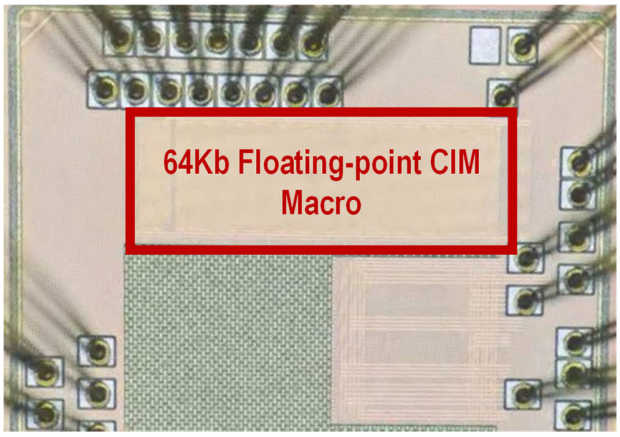


Fig. 21. Die photograph of the fabricated 28-nm 64-kb floating-point CIM macro.

Fig. 19 shows the measured power and area breakdown. The presented power breakdown is measured under 50% input sparsity, 30% input toggle rate, and 0% weight sparsity. The Addertree and Indrv consume 71.6% energy, which is the power bottleneck of digital CIM. While for area cost, CIM array and Adder tree become dominant parts, which consume 62.8%.

TABLE II
COMPARISON TABLE

	JSSC'21 [1]		ISSCC'21 [2]		ISSCC'22 [3]		ISSCC'22 [5]		ISSCC'22 [6]	This work	
Array size	64Kb		384Kb		1Mb		64Kb		32Kb	**512x128 (64Kb)**	
Technology	28nm		28nm		28nm		5nm		28nm	**28nm**	
Cell structure	6T+LCC		6T+SILMC		6T+EDC		12T+NOR		6T+RLPU	**DBcell+FCU**	
MAC operation	Analog		Analog		Analog		Digital		Digital	**Digital**	
Macro size(mm^2)	0.323		1.64[1]		-		0.0133		0.030	**0.146**	
Memory density(Kb/mm^2)	198		234[1]		-		4812		1067	**438.35**	
Supply voltage(V)	0.7-0.9		0.7-0.9		0.65-0.9		0.5-0.9		0.8	**0.6-0.9**	
Model	CNN		CNN		CNN		CNN		CNN	**CNN, Yolo, U-Net**	
Input precision (bit)	4	8	4	8	4	8	4	8	8	8	**BF16**
Weight precision (bit)	4	8	4	8	4	8	4	8	8	8	**BF16**
Output precision (bit)	12	20	12	20	14	22	14	22	21	23	**BF16**
Access time(ns)	4.1	8.4	4	7.2	6.5	6.6	5.2	9.4	24	5.5	**6.8**
Throughput (GOPS)	124.88	30.48	768	213.33	4256	1050	740	411	5.8	112.3-375	**89.8-300 GFLOPS**
Energy Efficiency (TOPS/W)	47.8	11.5	60.28	15.02	118.8	27.21	254	63	27.38	19.5-44	**14.04-31.6 TFLOPS/W**
Inference Accuracy (%) (CIFAR100)	67.1	67.89	-	67.62	67.33	67.81	-	-	-	74.51	**75.94**
mAP(%) (COCO)	-	-	-	-	-	-	-	-	-	-	**52.40**
MIoU(%) (VOC2012)	-	-	-	-	-	-	-	-	-	-	**56.49**

[1] Quoted from reference [6]

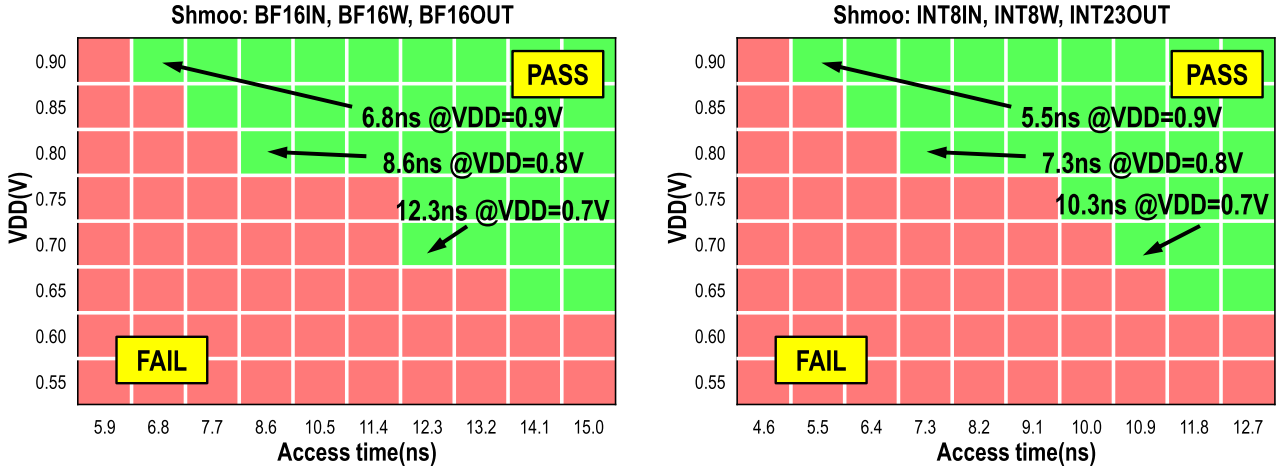


Fig. 22. Measured shmoo plots of BF16 IN, BF16 W, and BF16 OUT; and INT8 IN, INT8 IN, and INT23 OUT.

Fig. 20 shows the measured EF and area efficiency under different voltages. When tested under BF16 IN, BF16 W, and BF 16 OUT, the proposed macro achieves 31.6TFLOPS/W@0.6 V and 2.05 TFLOPS/mm^2@0.9 V. When tested under INT8 IN, INT8 W, and INT8 OUT, proposed macro achieves 44 TOPS/W@0.6 V and 2.57 TOPS/mm^2@0.9 V.

Fig. 21 shows the die micrograph of proposed 28-nm 64-kb FP CIM macro. Table I shows the chip summary. The macro size is 706.96 × 206.76 μm = 0.146 mm^2 and can support BF16 and INT8 mode in 128 accumulation channels. This chip could achieve an access time of 5.5 ns@INT8 mode and 6.8 ns@BF16 mode. The chip EF is measured in 0% weight sparsity, 50% feature sparsity,

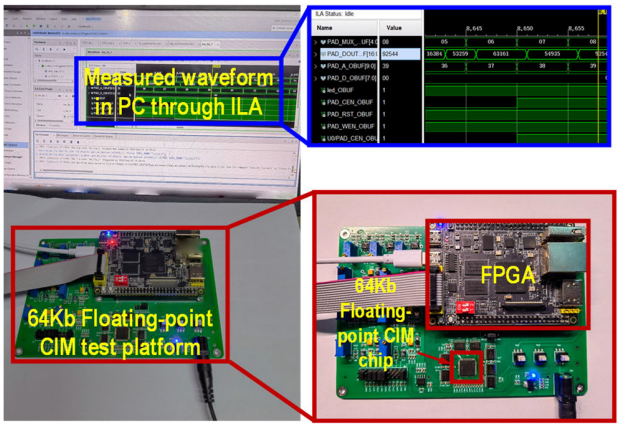


Fig. 23. Demonstration system of the proposed work.

and 30% feature toggle rate, which is an average case of CNN. This work achieves accuracy, mean average precision (mAP), and mean intersection over union (MIoU) of 75.94%, 52.40%, and 56.49% under ResNet18(@CIFAR100), Yolov3(@COCO), and U-Net(@VOC2012). Fig. 22 shows a measured shmoo plot of this work. To be noted that access time at 0.6 V (22.8 ns@BF16 and 18.2 ns@INT8) is not labeled in the shmoo plot based on space consideration. Table II shows a comparison table of this work. Fig. 23 shows the demonstration system.

VI. Conclusion

This work proposed a DB6T-based floating-point CIM macro to support three AI tasks: classification@CIFAR100, detection@COCO, and segmentation@VOC2012. To improve CIM-level EF and area efficiency, this article proposed a cell array with DBcells and FCU (high bit will HFMC and low bit with LAMC). To map FP-MAC in a CIM array, this article proposed a ShareFloatv2 data type, a CIM macro architecture with FP processing circuits to support both FP-MAC and INT-MAC and an LUT-based training method. The fabricated 28-nm 64-kb Floating-point SRAM-CIM macro has proved the proposed concept and achieved the best EF (31.6 TFLOPS/W) and the highest area efficiency (2.05 TFLOPS/mm^2) for FP-MAC with BF16 IN/W/OUT. This work also achieves accuracy, mAP, and MIoU of 75.94%, 52.40%, and 56.49% under ResNet18(@CIFAR100), Yolov3(@COCO), and U-Net(@VOC2012), respectively.

References

[1] X. Si et al., "A local computing cell and 6T SRAM-based computing-in-memory macro with 8-b MAC operation for edge AI chips," *IEEE J. Solid-State Circuits*, vol. 56, no. 9, pp. 2817–2831, Sep. 2021.

[2] J.-W. Su et al., "Two-way transpose multibit 6T SRAM computing-in-memory macro for inference-training AI edge chips," *IEEE J. Solid-State Circuits*, vol. 57, no. 2, pp. 609–624, Feb. 2022.

[3] X. Si et al., "A twin-8T SRAM computation-in-memory unit-macro for multibit CNN-based AI edge processors," *IEEE J. Solid-State Circuits*, vol. 55, no. 1, pp. 189–202, Jan. 2020.

[4] W.-S. Khwa et al., "A 65 nm 4 Kb algorithm-dependent computing-in-memory SRAM unit-macro with 2.3 ns and 55.8 TOPS/W fully parallel product-sum operation for binary DNN edge processors," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 496–497.

[5] H. Mori et al., "A 4 nm 6163-TOPS/W/b 4790 – $TOPS/mm^2/b$ SRAM based digital-computing-in-memory macro supporting bit-width flexibility and simultaneous MAC and weight update," in *Proc. IEEE Int. Solid- State Circuits Conf. (ISSCC)*, San Francisco, CA, USA, Feb. 2023, pp. 132–134.

[6] B. Wang et al., "A 28 nm horizontal-weight-shift and vertical-feature-shift-based separate-WL 6T-SRAM computation-in-memory unit-macro for edge depthwise neural-networks," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2023, pp. 134–136.

[7] S.-E. Hsieh et al., "7.6 A 70.85–86.27 TOPS/W PVT-insensitive 8 b word-wise ACIM with post-processing relaxation," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2023, pp. 136–138.

[8] P. Chen et al., "7.8 A 22 nm delta-sigma computing-in-memory ($\Delta\Sigma$CIM) SRAM macro with near-zero-mean outputs and LSB-first ADCs achieving 21.38 TOPS/W for 8 b-MAC edge AI processing," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2023, pp. 140–142.

[9] Z. Yue et al., "7.7 CV-CIM: A 28 nm XOR-derived similarity-aware computation-in-memory for cost-volume construction," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2023, pp. 1–3.

[10] S. Liu et al., "16.2 A 28 nm 53.8 TOPS/W 8 b sparse transformer accelerator with in-memory butterfly zero skipper for unstructured-pruned NN and CIM-based local-attention-reusable engine," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2023, pp. 250–252.

[11] Y.-C. Chiu et al., "A 22 nm 4 Mb STT-MRAM data-encrypted near-memory computation macro with a 192 GB/s read-and-decryption bandwidth and 25.1–55.1 TOPS/W 8 b MAC for AI operations," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2022, pp. 178–180.

[12] W.-S. Khwa et al., "A 40-nm, 2 M-cell, 8 b-precision, hybrid SLC-MLC PCM computing-in-memory macro with 20.5–65.0 TOPS/W for tiny-AI edge devices," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2022, pp. 1–3.

[13] J.-M. Hung et al., "An 8-Mb DC-current-free binary-to-8 b precision ReRAM nonvolatile computing-in-memory macro using time-space-readout with 1286.4–21.6 TOPS/W for edge-AI devices," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2022, pp. 1–3.

[14] Y. Zhang et al., "Single-mode CMOS 6T-SRAM macros with keeper-loading-free peripherals and row-separate dynamic body bias achieving 2.53 fW/bit leakage for AIoT sensing platforms," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2022, pp. 184–186.

[15] F. Tu et al., "16.4 TensorCIM: A 28 nm 3.7 nJ/gather and 8.3 TFLOPS/W FP32 digital-CIM tensor processor for MCM-CIM-based beyond-NN acceleration," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2023, pp. 254–256.

[16] H. Fujiwara et al., "A 5-nm 254-TOPS/W 221-TOPS/mm^2 fully-digital computing-in-memory macro supporting wide-range dynamic-voltage-frequency scaling and simultaneous MAC and write operations," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2022, pp. 1–3.

[17] B. Yan et al., "A 1.041-Mb/mm^2 27.38-TOPS/W signed-INT8 dynamic-logic-based ADC-less SRAM compute-in-memory macro in 28 nm with reconfigurable bitwise operation for AI and embedded applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2022, pp. 188–190.

[18] P.-C. Wu et al., "A 28 nm 1 Mb time-domain computing-in-memory 6T-SRAM macro with a 6.6 ns latency, 1241 GOPS and 37.01 TOPS/W for 8 b-MAC operations for edge-AI devices," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 65, San Francisco, CA, USA, Feb. 2022, pp. 1–3.

[19] A. Guo et al., "A 28 nm 64-kb 31.6-TFLOPS/W digital-domain floating-point-computing-unit and double-bit 6T-SRAM computing-in-memory macro for floating-point CNNs," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2023, pp. 128–130.

[20] P.-C. Wu et al., "A 22 nm 832 Kb hybrid-domain floating-point SRAM in-memory-compute macro with 16.2–70.2 TFLOPS/W for high-accuracy AI-edge devices," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2023, pp. 126–128.

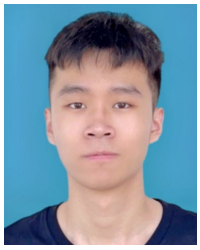
[21] J. Yue et al., "A 28 nm 16.9–300 TOPS/W computing-in-memory processor supporting floating-point NN inference/training with intensive-CIM sparse-digital architecture," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2023, pp. 1–3.
[22] F. Tu et al., "A 28 nm 29.2 TFLOPS/W BF16 and 36.5 TOPS/W INT8 reconfigurable digital CIM processor with unified FP/INT pipeline and bitwise in-memory booth multiplication for cloud deep learning acceleration," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2022, pp. 1–3.
[23] X. Chen, A. Guo, X. Xu, X. Si, and J. Yang, "A quantization model based on a floating-point computing-in-memory architecture," in *Proc. IEEE Asia–Pacific Conf. Circuits Syst. (APCCAS)*, Shenzhen, China, Nov. 2022, pp. 493–496.
[24] Z. Feng, B. Wang, Z. Zhang, A. Guo, and X. Si, "A booth-based digital compute-in-memory marco for processing transformer model," in *Proc. IEEE Asia–Pacific Conf. Circuits Syst. (APCCAS)*, Shenzhen, China, Nov. 2022, pp. 524–527.
[25] B. Wang et al., "SNNIM: A 10T-SRAM based spiking-neural-network-in-memory architecture with capacitance computation," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Austin, TX, USA, May 2022, pp. 3383–3387.
[26] A. Guo et al., "ShareFloat CIM: A compute-in-memory architecture with floating-point multiply-and-accumulate operations," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Austin, TX, USA, May 2022, pp. 2276–2280.
[27] T. Xiong et al., "Design methodology towards high-precision SRAM based computation-in-memory for AI edge devices," in *Proc. 18th Int. SoC Design Conf. (ISOCC)*, Oct. 2021, pp. 195–196.
[28] J. Lee et al., "A 13.7 TFLOPS/W floating-point DNN processor using heterogeneous computing architecture with exponent-computing-in-memory," in *Proc. Symp. VLSI Circuits*, Jun. 2021, pp. 1–2.
[29] X. Si, Y. Zhou, J. Yang, and M.-F. Chang, "Challenge and trend of SRAM based computation-in-memory circuits for AI edge devices," in *Proc. IEEE 14th Int. Conf. ASIC (ASICON)*, Kunming, China, Oct. 2021, pp. 1–4.
[30] X. Si et al., "15.5 A 28 nm 64 Kb 6T SRAM computing-in-memory macro with 8 b MAC operation for AI edge chips," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2020, pp. 246–248.
[31] A. Guo, C. Xue, X. Chen, and X. Si, "VCCIM: A voltage coupling based computing-in-memory architecture in 28 nm for edge AI applications," *CCF Trans. High Perform. Comput.*, vol. 4, no. 4, pp. 407–420, Jul. 2022.
[32] Z. Zhang et al., "From macro to microarchitecture: Reviews and trends of SRAM-based compute-in-memory circuits," *Sci. China Inf. Sci.*, vol. 66, no. 10, pp. 200403:1–200403:19, Oct. 2023.

An Guo (Graduate Student Member, IEEE) received the B.S. and M.S. degrees in electronic science and engineering and microelectronics science and engineering from Southeast University, Nanjing, China, in 2020 and 2022, where he is currently pursuing the Ph.D. degree in integrated circuit engineering.

His current research interests include memory, artificial intelligence (AI) accelerator, and computing in memory.

Xi Chen (Graduate Student Member, IEEE) received the B.S. degree from Southeast University, Nanjing, China, in 2022, where he is currently pursuing the Ph.D. degree with the School of Integrated Circuit Science and Engineering.

His research interests include deep learning and computing in memory.



Fangyuan Dong received the B.S. degree in electronic science and technology from Southeast University, Nanjing, China, in 2022, where he is currently pursuing the M.S. degree.

His research interests include static random-access memory (SRAM)-based computing-in-memory macro and accelerator designs.

Xingyu Pu received the B.S. degree in electronic science and engineering from Southeast University, Nanjing, China, in 2021, where she is currently pursuing the master's degree with the National ASIC Center.

Her research interests include analog circuit design and static random-access memory (SRAM)-based computing in memory.

Dongqi Li received the B.S. degree in microelectronics science and engineering from Fuzhou University, Fuzhou, China, in 2021. He is currently pursuing the M.S. degree in integrated circuit engineering with Southeast University, Nanjing, China.

His current research interests include circuit design of static random-access memory (SRAM) and computing in memory.

Jingmin Zhang received the B.S. degree in electronic science and engineering from Southeast University, Nanjing, China, in 2022, where he is currently pursuing the M.S. degree.

His research interests include digital circuit design, memory, and computing-in-memory circuit designs.

Xueshan Dong received the B.S. degree in physics from Jilin University, Changchun, China, in 2017. He is currently pursuing the M.S. degree with the Department of Electronics, Southeast University, Nanjing, China.

His current research interests include static random-access memory (SRAM) design, computing-in-memory, and low-power circuit design.

Hui Gao received the B.S. degree in optoelectronic information science and engineering from Hefei University of Technology, Hefei, China, in 2021. He is currently pursuing the M.S. degree in integrated circuit engineering with Southeast University, Nanjing, China.

His current research interests include analog circuit design and computing-in-memory circuit designs.

Yiran Zhang received the B.S. degree in electronic science and engineering from Southeast University, Nanjing, China, in 2022, where she is currently pursuing the master's degree with the National ASIC Center.

Her research interests include analog circuit design and static random-access memory (SRAM)-based computing in memory.

Bo Wang received the B.S. degree in microelectronics science and engineering from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2018. He is currently pursuing the Ph.D. degree with Southeast University, Nanjing.

His research interests include computing in memory, edge artificial intelligence (AI) accelerator, and neuromorphic hardware.

Jun Yang (Member, IEEE) received the B.S. and Ph.D. degrees in electronic engineering from Southeast University, Nanjing, China, in 1999 and 2004, respectively.

He is currently a Professor with the National ASIC System Engineering Research Center, Southeast University. He has authored or coauthored over 50 technical articles in conferences and journals, including the IEEE International Solid-State Circuits Conference (ISSCC), the Design Automation Conference (DAC), the *Journal of Solid-State Circuits*, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS(TCAS-I), and IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS (TVLSI). He has authorized over 100 Chinese patents and U.S. invention patents. His current research focus on static random-access memory (SRAM) design, in-memory computing, and near-threshold design.

Dr. Yang serves as an International Technical Program Committee Member for ISSCC and ASSCC.

Xin Si (Member, IEEE) received the B.S. and Ph.D. degrees in integrated circuit design and integration system from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2016 and 2020, respectively.

Since 2021, he has been an Associate Professor with the School of Integrated Circuits, Southeast University (SEU), Nanjing, China. He has an extensive publication record, including over 30 conference/journal papers, notably 12 IEEE International Solid-State Circuits Conference (ISSCC) papers, and seven IEEE JOURNAL OF SOLID-STATE CIRCUITS (JSSC) articles. His current research interests encompass memory, computing-in-memory circuit, and artificial intelligence (AI) chip designs.

Dr. Si also serves as an International Technical Program Committee member for the IEEE MCSoC and VLSI-DAT.