PIMCA: A 3.4-Mb Programmable In-Memory Computing Accelerator in 28nm for On-Chip DNN Inference

Shihui Yin[1], Bo Zhang[2], Minkyu Kim[1], Jyotishman Saikia[1], Soonwan Kwon[3], Sungmeen Myung[3], Hyunsoo Kim[3], Sang Joon Kim[3], Mingoo Seok[2], and Jae-sun Seo[1]

[1]Arizona State University, USA, [2]Columbia University, USA, [3]Samsung Advanced Institute of Technology, South Korea

Abstract: We present a programmable in-memory computing (IMC) accelerator integrating 108 capacitive-coupling-based IMC SRAM macros of a total size of 3.4 Mb, demonstrating one of the largest IMC hardware to date. We developed a custom ISA featuring IMC and SIMD functional units with hardware loop to support a range of deep neural network (DNN) layer types. The 28nm prototype chip achieves system-level peak energy-efficiency of 437 TOPS/W and peak throughput of 4.9 TOPS at 40MHz, 1V supply.

Keywords: In-memory computing, custom ISA, DNN accelerator.

Introduction

DNNs have achieved human-level performance in many recognition tasks. These DNNs usually require billions of multiply-and-accumulate (MAC) operations, soliciting energy-efficient architecture for on-chip DNN inference. In-memory computing (IMC) has emerged as a promising technique owing to high parallelism, reduced data communication, and energy-efficient analog computation. Recently, test chips containing a single IMC SRAM macro [1-2] or multiple of them [3-6] have demonstrated high energy-efficiency. However, most prior works still integrated a limited number of IMC macros [3-4]. Also, the dataflow of IMC/non-IMC operations was often hard-wired [5-6], limiting flexibility to support various layer types. Furthermore, hardware loop support is critical for scaling instruction-related overhead, but many prior works do not have it [3-6].

In this work, we present a programmable IMC accelerator (PIMCA) in 28nm. It integrates 3.4 Mb capacitive IMC SRAM, demonstrating one of the largest integrations among IMC accelerators to date. Also, as part of the instruction set architecture (ISA), a flexible single-instruction-multiple-data (SIMD) processor is integrated to support a range of non-MAC vector operations, such as average-/max-pooling, residual layers, etc. It also features hardware loop support, reducing instruction count and latency. Leveraging low-precision deep learning models [7], PIMCA can support DNNs with 1-b and 2-b precision. It maps the 1-b VGG-9 model with 2.89 Mb weights fully stored on-chip yet consumes only 2.36 μJ per inference with 47.3 μs latency.

Architecture and Operation

We designed the IMC SRAM macro (256×128) based on the capacitive-coupling computing mechanism. The bitcell is similar but different from [1]; first, two transmission gates, not pass gates, are added to the 6T cell to access a coupling capacitor without any Vt drop. Also, the coupling capacitor C_c (~2.2fF) is implemented as a MOM (M4-M6) on top of the bitcell for area-efficiency. The binary multiplication result of each bitcell is accumulated over MBLs via capacitive coupling. The MBL voltage (V_{MBL}) of each column is converted to 4-b values by an 11-level flash ADC. Fig. 1 depicts the overall architecture of PIMCA, which integrates 108 IMC macros organized in six processing elements (PEs). In each PE, 18 macros are organized in a 3×6 array. At each cycle, one PE performs matrix-vector multiplication (MVM) using 1 to 18 macros. The partial sums from the macros are added up by a configurable adder tree in the PE. A 256-way SIMD takes the PE output and performs non-MAC vector operations. The SIMD stores its outputs in the activation memory (AM).

Fig. 2(a) illustrates the six-stage pipeline consisting of one of the six PEs, the SIMD, the AM, and other common circuits. Every cycle, an instruction is fetched and decoded in IF and ID stages. Then, it loads input vectors from AM in LD, performs MAC computation in IMC and other non-MAC computation in SIMD, and writes the results back to AM if needed in WB. The custom ISA has two types of instructions: regular and loop instructions. A regular instruction (Fig. 2(b)) performs MAC/non-MAC computation. It contains three major fields: i) read and write (R/W) addresses and AM enable, ii) PE and macro selection and accumulation mode control, iii) SIMD operands and operation code. For loop support, each regular instruction contains a 6-b field that defines repetitions (up to 64).

To support generic for-loops, the ISA has loop instructions (Fig. 2(c)); the loop-setup (LS) and loop-end-check (LE) instructions can define up to eight levels of nested for-loops by setting special loop registers and counters (LR, LC). For the case of 1-b VGG-9 DNN inference, exploiting the repetitive computation types, the proposed hardware loop support reduces the total instruction count by 4X.

PIMCA also integrates 1.54 Mb activation memory (AM) using off-the-shelf single-port SRAM for storing input image, intermediate data, batch normalization (BN) parameters, and final outputs. The input and output data of a layer need to simultaneously access the AM. Thus, we split the AM in two groups: top and bottom such that they serve as the input and output buffer alternatively across layers. Each AM group has six banks (1024×128-b) to support flexible and parallel AM access. The active PE can access any 3×1×256 (height×width×channel) input patch from those banks in a cycle, simplifying the streaming process by eliminating the need for extra buffering between AM and PE. To support this access, we devised the scheme for address generation and activation rotation (Fig. 3).

In each PE, we organize the macros in the 3×6 matrix to efficiently support convolutions with three popular kernel sizes, 3×3, 5×5, and 1×1. For 3×3, we split the 3×6 macros into two 3×3 groups, and we can map a 1-b convolution layer of 256×256 input and output channels or 2-b of 256×128 in a PE (Fig. 4(a)). Note that the two 3×3 groups share the input vectors. Within each 3×3 macro group, the inputs are pipelined horizontally (from left to right), exploiting the convolutional data reuse (same weights convolved with different inputs). Also, for 5×5, we can map a 1-b convolution layer of 128×128 by connecting the output of input registers of the left group to the input of the right group (Fig. 4(c)). Note that inactive macros are designed to produce zero outputs. Therefore, for zero padded inputs, by disabling corresponding macros, we can save MVM computation energy (Fig. 4(d)).

The 256-way SIMD processor performs all non-MAC computations. It supports eight types of operations: 'LOAD' offers data transfer; 'ADD' performs partial sum addition (Z=X+Y); 'ADD2' performs shift-and-add (Z=2X+Y), which efficiently supports i) bit-serial scheme for 2-b input (X and Y from the same SIMD lane) and ii) bit-parallel scheme (Fig. 4(b)) for 2-b weight (X/Y from left/right lanes); 'CMP' and 'CMP2' do comparison (Z=(X>Y)) for computing 1-b and 2-b activation results; 'MAX' selects the maximum value during max-pooling; 'LSHIFT'/'RSHIFT' shift data left/right, critical to support simple multiplication/division.

Measurement Results

We prototyped the PIMCA in 28nm (Fig. 5). At 40 MHz and 1V supply, the peak throughput is 4.9 TOPS (1-b). The binary VGG-9 DNN for CIFAR-10 with 2.89 Mb weights can fully fit in PIMCA, and on-chip inference results in 289 TOPS/W average *system-level* energy-efficiency and consumes 47.3 μs latency and 2.36 μJ energy per inference, where the latter is 2.25X lower than [5]. When a 1-b/2-b DNN uses 18 macros per PE, the PIMCA achieves 437/62 TOPS/W peak energy-efficiency. We also evaluated the test chip with respect to variability, system power, DNN accuracy, and energy-efficiency in macro-level and peak/average system-level (Fig. 6). Due to the nature of analog computation, the IMC SRAM macros exhibit variations in their ADC outputs. Therefore, we developed a measurement-based variation model and included it in the DNN training framework. Table I shows the comparison to prior works. PIMCA achieves among the largest integration and highest energy-efficiency while providing significantly enhanced flexibility and programmability.

References

[1] Z. Jiang *et al.*, *JSSC*, 2020. [2] X. Si *et al.*, *JSSC,* 2020.
[3] R. Guo *et al.*, *Symp. VLSI Circuits*, 2019.
[4] J. Yue *et al.*, *ISSCC*, 2020. [5] H. Valavi *et al.*, *JSSC,* 2019.
[6] H. Jia *et al.*, *JSSC*, 2020. [7] J. Choi *et al.*, *MLSys*, 2019.

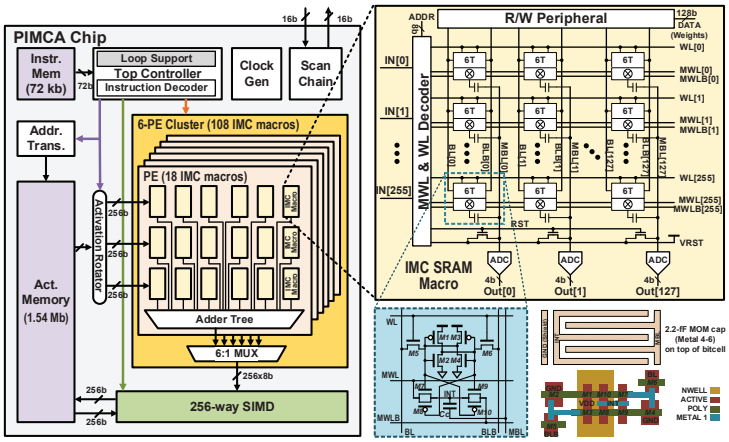


Fig. 1: Overall PIMCA architecture consisting of 108 IMC SRAM macros.

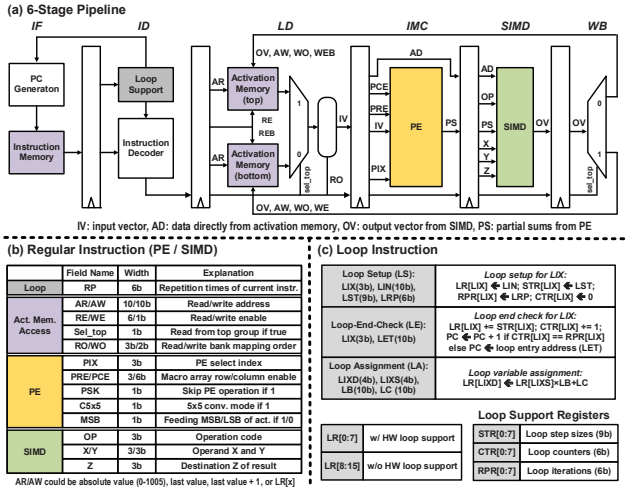


Fig. 2: Execution flow of PIMCA in a 6-stage pipeline and instruction set featuring hardware loop support.

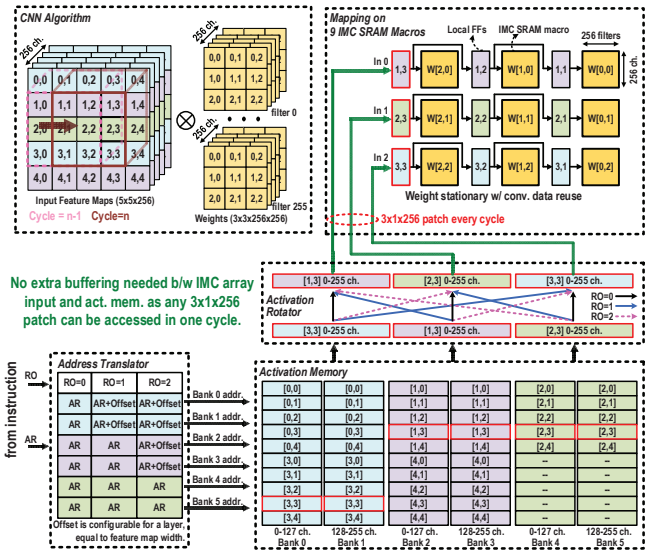


Fig. 3: Activation data storage and efficient streaming process between activation memory and IMC SRAM macros.

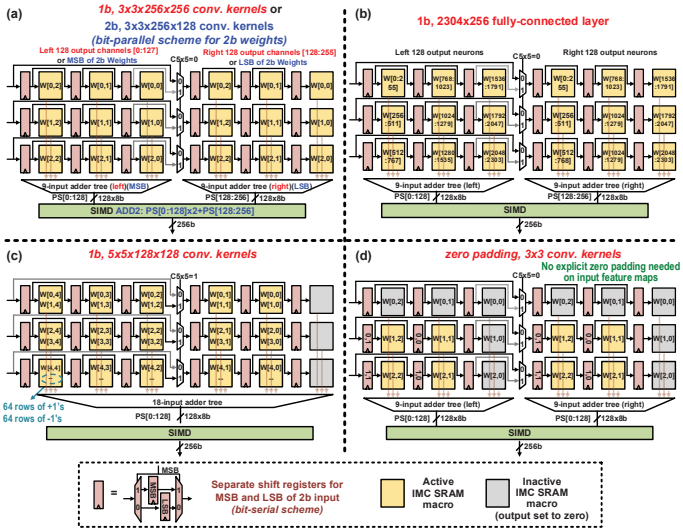
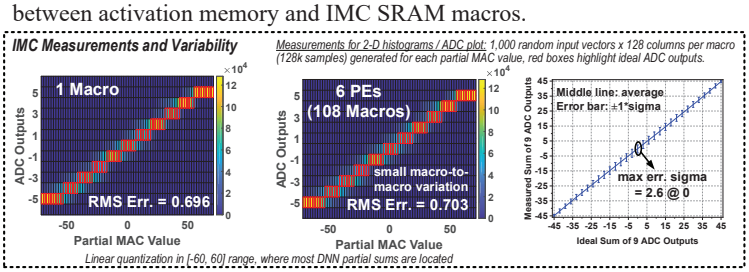
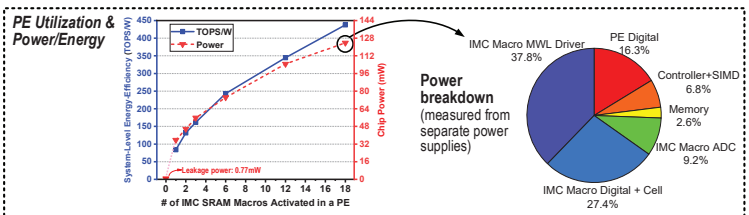


Fig. 4: Flexible IMC SRAM macro mapping (layer, kernel, etc.).





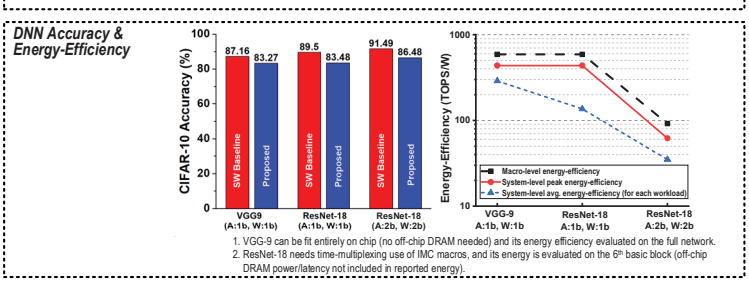


Fig. 6: Measurement results on IMC operations, variability, PE utilization, and system power breakdown.

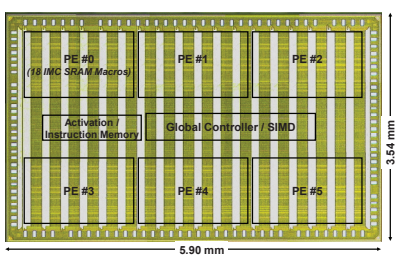


Fig. 5: 28nm PIMCA chip micrograph.

Table I: Comparison to prior system-level IMC SRAM designs.

	VLSI'19 [3]	ISSCC'20 [4]	JSSC'19 [5]	JSSC'20 [6]	This Work
Technology	65nm	65nm	65nm	65nm	28nm
DNN Model	RNN	CNN/FC	CNN	CNN/FC	CNN/FC
Supported CNN Kernel	N/A	3x3	3x3	3x3	3x3, 5x5, 1x1
Flexible Non-MAC Operation Support	No	No	No	Yes	Yes
Supply Voltage (V)	0.9-1.1	0.9-1.05	0.94-1.2	0.85-1.2	1.0
Area (mm^2)	9.6	5.66	12.6	13.5	20.9
Clock Frequency (MHz)	6-75	50-100	100	40-100	40
IMC Bitcell	6T	6T	10T1C	10T1C	10T1C
Digital SRAM (kb)	80	1,312	64	256	1,608
IMC SRAM (kb)	64	4	2,304	576	3,456
Bit Precision	1b	2/4/6/8b (Act.) 4/8b (Weight)	1b	1b-8b	1b-2b
ADC Precision	3b	5b	1b	8b	4b
Performance (TOPS)	0.61	0.17-2.0	18.9	2.2 (1b)	4.9 (1b)
Power (mW)	N/A	31.8-65.2	N/A	N/A	124 (1b)
IMC-macro-level Peak Energy-Eff. (TOPS/W)	51.6	158.7	866	400 (1b)	588 (1b)
System-level Peak Energy-Eff. (TOPS/W)	11.7	35.8	658	N/A	437 (1b)
System-level Avg. Energy Eff. (TOPS/W) for ResNet-18	51.6	158.7	866	N/A	136 (1b) 35 (2b)
Energy per Inference for VGG-9 (μJ)	11.7	35.8	658	5.31 (1b)	2.36 (1b)