PS-IMC: A 2385.7-TOPS/W/b Precision Scalable In-Memory Computing Macro With Bit-Parallel Inputs and Decomposable Weights for DNNs

Amitesh Sridharan, *Graduate Student Member, IEEE*, Jyotishman Saikia, *Student Member, IEEE*, Anupreetham, *Student Member, IEEE*, Fan Zhang, *Student Member, IEEE*, Jae-Sun Seo, *Senior Member, IEEE*, and Deliang Fan, *Member, IEEE*

***Abstract*—We present a fully digital multiply and accumulate (MAC) in-memory computing (IMC) macro demonstrating one of the fastest flexible precision integer-based MACs to date. The design boasts a new bit-parallel architecture enabled by a 10T bit-cell capable of four AND operations and a decomposed precision data flow that decreases the number of shift–accumulate operations, bringing down the overall adder hardware cost by 1.57× while maintaining 100% utilization for all supported precision. It also employs a carry save adder tree that saves 21% of adder hardware. The 28-nm prototype chip achieves a speed-up of 2.6×, 10.8×, 2.42×, and 3.22× over prior SoTA in 1bW:1bI, 1bW:4bI, 4bW:4bI, and 8bW:8bI MACs, respectively.**

***Index Terms*—CNN, digital in-memory computing (IMC) macro, flexible precision, multiply and accumulate (MAC).**

I. Introduction

In recent years, due to the unprecedented success of AI, there has been a need for efficient deployment of AI workloads in a scalable manner. Recently in-memory computing (IMC) has been widely investigated as a promising approach to accelerate AI workloads. There are mainly two prevalent IMC design paradigms, i.e., compute in analog or digital domain. Analog IMC has gained attention due to the large number of operations it can perform per watt as well as per unit area. But it faces significant drawback from computing accuracy standpoint. On the other hand, digital IMCs [1], [2], [3], [4], [5] are more akin to digital ASICs closely interleaving memory and logic units. Many recent digital IMC works demonstrate high throughput and energy efficiency compared to their analog counterparts, without any accuracy drop due to robust rail-to-rail logic operations.

Convolutions are at the heart of deep learning algorithms and are the most compute intensive operations. They follow a multidimensional compute pattern, where the weights and inputs have five dimensions (2-D kernel, input channel, output channel, and bit-width). Typical IMC designs follow a weight stationary approach by storing flattened 5-D weights (W) in 2D-space. The 2-D weight matrix is constructed by assigning output channels and bit-width to IMC rows (enabling parallel multiplications), the kernel dimensions and input channels to IMC columns (enabling parallel accumulations). The 5D-input feature maps (input/IP) are streamed from outside the memory onto the word lines (WLs) performing multiplications within the bit-cell. WLs being a limited resource (1-bit/cycle per WL in most cases), the 5D-input stream-in is time-multiplexed to just 1D-input/unit time. In this setup, the input bit-width is also unrolled in the time domain, hence a larger number of time-shared shift accumulations circuits are present. The overall system incurs a large latency overhead due to the bit-wise stream-in of inputs [latency overhead = (input precision) × (# of inputs)]. To maintain high throughput, large weight precision is typically addressed spatially by grouping several memory columns together. This in-turn reduces the flexibility to tune weight precision during inference.

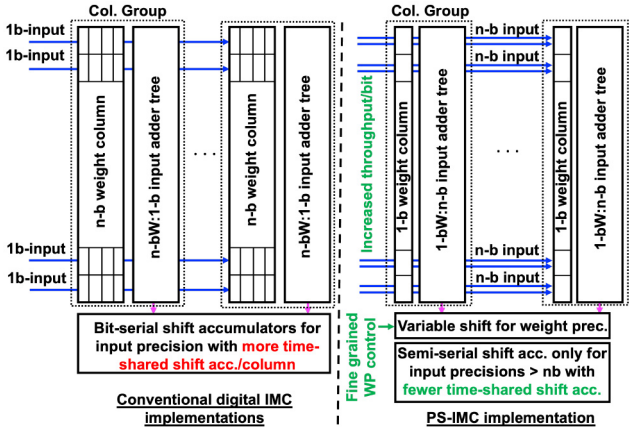


Fig. 1. Comparison of conventional digital IMC implementations versus proposed throughput-oriented PS-IMC implementation.

With the purpose of improving storage density, recent works [1], [4], [6] attempted to time-share the compute hardware (adders, multipliers, etc.) with more memory cells at the cost of throughput. Considering that convolutions are compute bound, our approach is to maximize the throughput akin to [2]. However, [2] requires weight and/or input replication to achieve full utilization for different precision MACs. To summarize, the contributions of this work are as follows.

1) Our proposed precision-scalable IMC (PS-IMC) offers flexible weight and input precision configuration. It also supports bit-parallel inputs without weight replication.
2) We present a novel decomposed weight precision data flow for digital IMCs that reduces the number of shift-operations by 128× over baseline.
3) All adder-trees are implemented as carry save adders (CSAs) through which we achieve a 21% reduction in the number of full-adders/tree.

Fig. 1 illustrates the PS-IMC design and its advantages over prior works. Implemented in 28-nm CMOS, PS-IMC achieves the highest

Manuscript received 13 December 2023; revised 4 February 2024; accepted 13 February 2024. Date of publication 23 February 2024; date of current version 13 March 2024. This work was supported in part by the National Science Foundation under Grant 2144751, Grant 2349802, Grant 2342726, Grant 2314591, Grant 2328803, and Grant 2414603; and in part by CoCoSys in JUMP 2.0, an SRC Program sponsored by DARPA. This article was approved by Associate Editor Jongsun Park. *(Corresponding author: Deliang Fan.)*

Amitesh Sridharan, Fan Zhang, and Deliang Fan are with the Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: dfan10@jhu.edu).

Jyotishman Saikia is with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287 USA.

Anupreetham and Jae-Sun Seo are with the School of Electrical and Computer Engineering, Cornell Tech, New York, NY 10044 USA.

Digital Object Identifier 10.1109/LSSC.2024.3369058

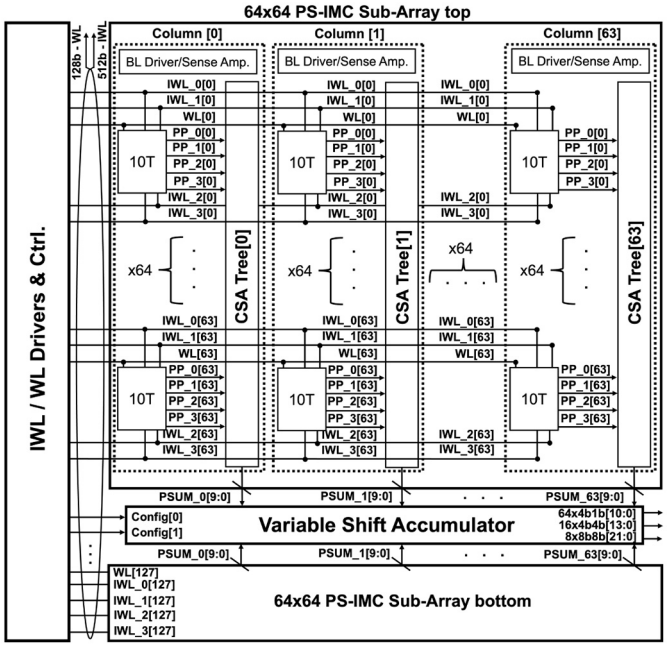


Fig. 2. PS-IMC macro architecture design.

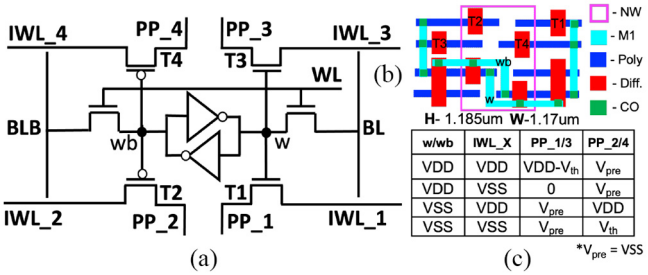


Fig. 3. (a) Bit-cell schematic. (b) Layout. (c) Truth table.

throughput for all supported MAC precision [1/4/8-b Weight (W):1-8b Input (I)], the highest energy efficiency for 1bW:1bI and 1bW:4bI MACs and the highest normalized compute density (TOPS/mm^2) for 8bW:8bI MACs.

II. PS-IMC Macro Architecture

Fig. 2 depicts the architecture diagram of the PS-IMC macro. We design the macro using a semi-bit-parallel architecture, i.e., the design is completely bit-parallel up to 4b-input, and a higher input precision will require time multiplexing (4b-input/unit time). There is a pipeline stage between the adder-tree and shifter to evenly distribute the critical path across two cycles. Hence, it takes two clock cycles to complete a MAC with a 4b-input and 4b-weight, three cycles for a MAC with 8b-input with 4b or 8b-weight, and only one clock cycle for a 4b-input and 1b-weight. We implement two 128 × 64 PS-IMC macros on the prototype chip and each 128 × 64 macro has two 64 × 64 subarrays stacked one on top of the other, with the variable shift accumulator (VSA) in the middle. Each column of the PS-IMC sub array has one 64-input 4-bit CSA tree and 64 10T-SRAM bit-cells.

A. Bit-Cell Design

Fig. 3(a) and (b) shows the proposed 10T bit-cell schematic and layout. Each bit-cell occupies 1.38 μm^2 and is designed using logic rules. Each of the four additional transistors (T1–T4) perform a pass-gate-based dot-product between 4b input streamed in through

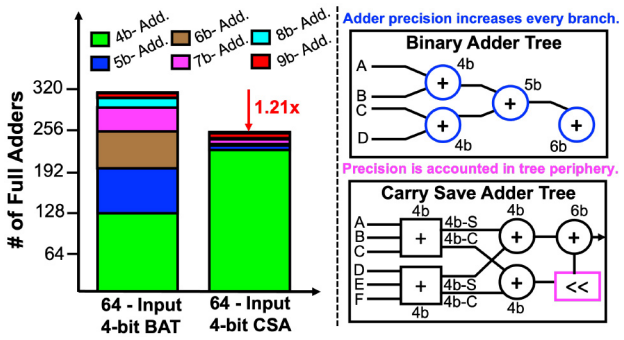


Fig. 4. Precision breakdown and tree structures of CSA and BAT.

the input/IP WLs (IWLs) and the 1b weight and its complement stored on either side of the cross-coupled inverters. Each transistor (T1–T4) is allocated to one input bit-significance and performs a 1bW:1bI dot-product. Thus, each bit-cell as a whole can perform a 1bW:4bI dot-product. Prior to compute, the partial product (PP) nets are precharged to VSS and the IWLs are held at VSS until all the weight bits are written into the SRAM bit-cells. This avoids any erroneous compute when the stored bit is zero as the pass gates are controlled by the stored weights. Additionally, PP_X requires precharge after a large time interval of 0.5 μs. This is dependent on the RC load on the PP_X. We determine this precharge interval based on post layout simulations. The functionality of the pass-gate-based AND operation is shown in the truth table in Fig. 3(c).

B. Carry Save Adder Tree

Fig. 4 compares the precision breakdown of all branches in a binary adder tree (BAT) widely used in prior works [1], [3], [4] and a CSA tree of similar configuration used in this work. CSAs have been widely adopted in digital designs requiring multioperand additions because of the significant reduction in the number of full adders and the speed-up they provide. CSAs isolate each operand into partial sums and carries which are accumulated in parallel and the bit-precision is accounted for in the tree periphery. This utilizes fewer high-precision adders and thereby results in an overall reduction in the number of full adders, also resulting in a shorter critical path delay. For a 64-operand 4-bit configuration, the CSA requires 21% fewer full adders when compared to the BAT counterpart.

C. Decomposed Weight Precision Data Flow

Prior IMC designs [1], [3], [4] spatially encode multibit weight precision by grouping memory columns together, limiting weight precision flexibility. Bit-Fusion [7] proposes a hierarchical approach to achieve this by decomposing large adder and multiplier precision into smaller blocks that can be selectively tiled together. In this implementation, MACs are performed in the traditional sense by completing multibit multiplication for each operand prior to accumulation. Given that different shift–add hardware is required to support flexible precision, Bit-Fusion tradeoffs larger hardware overhead to support flexible precision. BitBlade [8] overcomes this hardware cost by allocating a fixed bit-position to an entire PE (capable of small fixed-precision MACs across several operands). The support for flexible precision comes when fusing the PEs together, the partial sum from each PE is subject to a shift operation based on the allocated bit-position. However, this reduction in multiplication hardware comes at the cost of increased accumulation hardware, but given that multiplications are significantly more expensive to perform (a 4-bit multiplication requires 4 ANDs, 3 shifts, 2 4-bit additions,

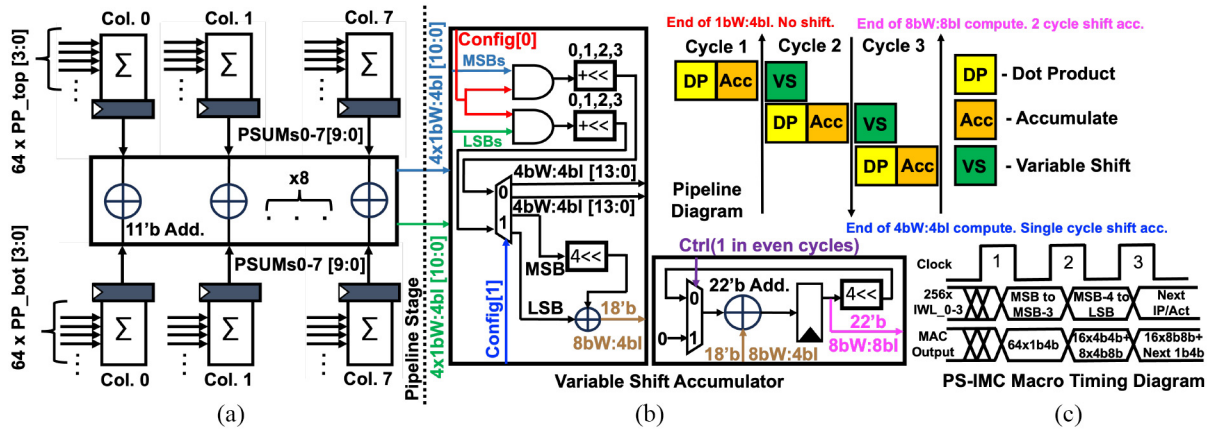
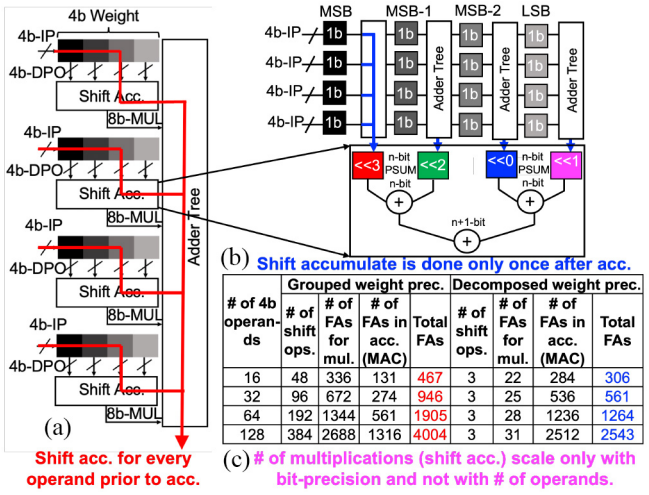


Fig. 5. (a) PSUM reduction from top and bottom subarrays. (b) VSA microarchitecture. (c) Three-stage pipeline diagram for variable precision MACs. (d) Timing diagram with the total number of MACs performed in a single PS-IMC macro.



# of 4b operands	Grouped weight prec. # of shift ops.	# of FAs for mul.	# of FAs in acc. (MAC)	Total FAs	Decomposed weight prec. # of shift ops.	# of FAs for mul.	# of FAs in acc. (MAC)	Total FAs
16	48	336	131	467	3	22	284	306
32	96	672	274	946	3	25	536	561
64	192	1344	561	1905	3	28	1236	1264
128	384	2688	1316	4004	3	31	2512	2543

(c) # of multiplications (shift acc.) scale only with bit-precision and not with # of operands.

Fig. 6. (a) Grouped weight precision. (b) PS-IMC's decomposed weight precision. (c) Multiply and accumulate hardware cost for 4-bit operands.

and 1 5-b addition), the increase in accumulation hardware is easily off-set by reduced multiplication hardware. The lowest granularity of fixed precision MACs in each PE is 2-bit hence even BitBlade incurs a multiplication cost that scales with the number of operands.

With this as motivation, we design PS-IMC using a completely weight decomposed data flow that un-groups all memory columns. Each memory column is allocated a weight bit-significance (for an n-b weight, the first column stores the MSB and the nth column stores the LSB). The 1bW:4bI PP generated by each bit-cell across all rows are accumulated first. After accumulation, each column is subjected to a shift operation depending on the bit-position. For example, considering a 4b weight, PPs from the MSB column will go through left-shift by 3 ($<<3$) and as we move down the column the shift value decreases by 1 [Fig. 6(b)]. By handling weight precision separately, the number of shift–accumulate operations remain constant regardless of the number of operands [Fig. 6(c)]. Through this approach, considering a 4-bit precision and 128 operands, the total full-adder (FA) cost (multiplication + accumulation) is reduced by 1.57× and the number of shifts performed can be reduced by 128×.

D. Variable Shift Accumulator for Precision Handling

PS-IMC supports a wide variety of MAC configurations (1b/4b/8b Weight and 1-8b Input). Fig. 5(b) shows the microarchitecture of the VSA. A VSA is padded to every eight columns to support a maximum of 8b weight precision. Config[1:0] signals in the VSA control the bit-precision of the weights and inputs. The 1bW:4bI MACs are collected in the pipeline stage before VSA in the same cycle as the input stream in. These MACs are then gated by Config[0], which enables column-specific shifts ($<<3$ for the 1st column and no shift for the last column) to obtain 4bW:4bI MACs. Every eight columns generate two 4bW:4bI MACs of 14-bit precision and these MAC outputs are further gated by Config[1] to selectively shift accumulate one of the 4bW:4bI MAC output to obtain a larger 8bW:4bI MAC of 18-bit precision. An alternative approach to implement a large weight precision (>4b) would be to subject each column to a variable shift (7b through 4b), but in this case, the unselected shifters remain idle and each IMC column incurs a multiplexer overhead as opposed to only one de-multiplexer for every four IMC columns.

Due to the bit-parallel nature of the 10T bit-cell, input precision of up to 4b can be handled in a single cycle without any time-multiplexing or special control. Only for input precision above 4b, multiplications are time-shared through shift-accumulators [Fig. 5(b)]. The VSA is tailored to support input precision ranging from 1-nb for an n-bit W. As a result, the a time-shared shift-accumulator is only necessary once every n columns. This reduces the shift-accumulators/column since input precision is accounted with little (4b–8b) to no (1b–4b) time-multiplexing. PS-IMC macro has three pipeline stages. The first, second, and third pipeline stages generate 64 4bI:1bW MACs, 16 4bI:4bW MACs, and 8 8bI:8bW MACs, respectively, as illustrated in Fig. 5(c) and (d).

III. Chip Measurements and Results

PS-IMC is prototyped in TSMC 28-nm CMOS. We implement two 128 × 64 macros on the prototype chip, where each macro occupies 0.32 mm^2. The PS-IMC macro achieves the highest peak throughput/kb compared to all prior digital IMC works for 1bW:1-4bI (3.25 TOPS/kb @ 1.2 V), 4bW:1-4bI (406.3 GOPS/kb @ 1.2 V), and 8bW:4-8bI (135.4 GOPS/kb @ 1.2 V). It also achieves the highest energy efficiency for 1bW:4bI (1843 TOPS/W @ 0.56 V) and 1bW:1bI (2385.7 TOPS/W @ 0.56 V). The measurement condition is with 50% bit-wise weight sparsity and an average of 25% toggle rate for the inputs at 27 °C. Input toggle rate has a linear dependence on power, where 25% decrease in toggling rate will yield about 8.7%–13% increase in energy efficiency. For a lower input precision (<4b), we disable the IWLs depending on the precision (a 2bI would mean we disable IWL_3 and IWL_2). A 1bW:1bI MAC in PS-IMC achieves the same throughput as a 1bW:4bI, but due to the reduction

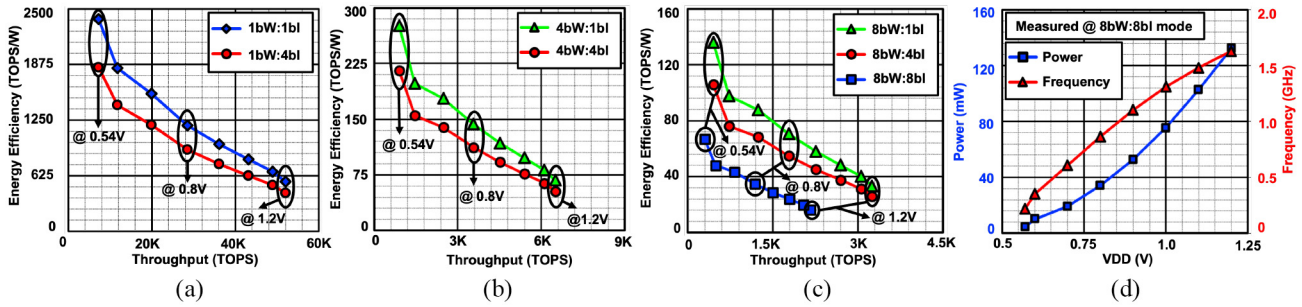


Fig. 7. Throughput (TOPS) versus energy efficiency (TOPS/W) for (a) 1bW:1/4bI, (b) 4bW:1/4bI, and (c) 8bW:1/4/8bI modes. (d) Power and frequency scaling.

TABLE I
COMPARISON WITH PRIOR DIGITAL IMC WORKS

Reference		This Work	VLSI' 22 [2]	ISSCC' 21 [3]	ISSCC' 22 [1]	JSSC'23 [5]	ESSCIRC' 23 [6]	ISSCC' 23 [4]
Technology		28nm	12nm	22nm	5nm	28nm	28nm	4nm
Bit cell Density		6T+4T (4xAND)	-	6T+4T (1xAND)	12T+1T	6T+2T (XNOR)	6T+0.5T	8Tx2bit+ OAI
Array Size		8Kb/Macro	8Kb	64Kb	64Kb	16Kb	16Kb	54Kb
Macro area (mm^2)		0.32/Macro	0.0323	0.202	0.0133	0.033, 0.049	0.0159	0.0172
Supply(V)		0.56-1.2	0.72	0.72	0.5-0.9	0.45-1.1	0.6-1.1	0.32-1.1
Input Precision		1b-8b	4b-8b	1b-8b	4b	1b-4b	1-8b	8b/12b/16b
Weight Precision		1b/4b/8b	4b/8b	4b/8b/ 12b/16b	4b	1b	8b	8b/12b
Full Output Precision		Yes	Yes	Yes	Yes	No	Yes	Yes
GOPS/Kb (W:I)	1b:1b	**451-3250**	-	-	-	1252	-	-
	1b:4b	**451-3250**	-	-	-	300	-	-
	4b:4b	**56.3-406.3**	167.9	51.56	46	-	-	-
	8b:8b	**18.7-135.4**	42	14.3	11.4	7.93	1.45	15.89
TOPS/W (W:I)	1b:1b	**557.4-2385.7**	-	-	-	1108-2219	-	-
	1b:4b	**430.7-1843.5**	-	-	-	154-248	-	-
	4b:4b	52.8-215.5	121	89	**254**	-	-	-
	8b:8b	16.39-66.8	30.3	24.7	63	9.6-15.5	60.4	**87.4**
Normalized $^{\$}$TOPS/$mm^2$ (W:I)	1b:4b	11.2-81.2	-	-	-	**98**	-	-
	4b:4b	**1.4-10.15**	7.63	10.15	7.05	-	-	-
	8b:8b	**0.47-3.38**	1.91	2.82	1.76	2.59	1.46	1.01

GOPS Calculation: 128(No. of rows)* 64(No. of cols)* 2(No. of macros) * 2(MAC)/Latency. GOPS/4,GOPS/8 for 4b and 8b weights respectively. 8b incurs more latency. $^{\$}$Normalized quadratically to 28nm.

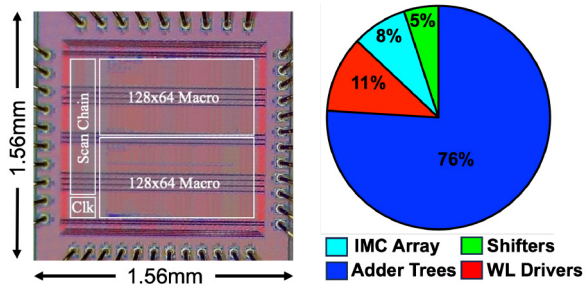


Fig. 8. Die micrograph (left) and area breakdown (right).

in the number of active IWLs 1bW:1bI achieves a higher energy efficiency. Fig. 7(a)–(c) illustrates the difference in energy efficiency as input precision is scaled with fixed weight precision. In addition, Fig. 7(c) shows the 33% latency cost when the input precision is scaled above 4b.

Table I compares PS-IMC against state-of-the-art digital SRAM IMC designs. It achieves throughput improvements of 2.6×, 10.8×, 2.42×, and 3.22× in 1bW:1bI, 1bW:4bI, 4bW:4bI, and 8bW:8bI MACs, respectively. By reducing multiplication and adder hardware (decomposed weight precision and CSA trees) and by increasing input bit-parallelism, PS-IMC achieves 1.1× and 7.4× improvements in TOPS/W for 1bW:1bI and 1bW:4bI MACs, respectively. It also achieves 1.2× improvement in normalized compute density for 8bW:8bI MACs. Fig. 8 shows the PS-IMC prototype chip and area breakdown.

IV. CONCLUSION

In this work, we present a throughput-oriented IMC macro that has a unique decomposed weight precision data flow for flexible precision bit-parallel MACs. PS-IMC maintains 100% utilization without weight replication with low hardware overhead. Measurement results show that PS-IMC achieves the highest throughput, energy efficiency, and compute density for various MAC workloads compared to prior SoTA IMC works.

REFERENCES

[1] 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 *Proc. IEEE ISSCC*, 2022, pp. 1–3.

[2] C.-F. Lee et al., "A 12nm 121-TOPS/W 41.6-TOPS/mm^2 all digital full precision SRAM-based compute-in-memory with configurable bit-width for AI edge applications," in *Proc. IEEE Symp. VLSI Circuits*, 2022, pp. 24–25.

[3] Y.-D. Chih et al., "An 89TOPS/W and 16.3TOPS/mm^2 all-digital SRAM-based full-precision compute-in memory macro in 22nm for machine-learning edge applications," in *Proc. IEEE ISSCC*, 2021, pp. 252–254.

[4] H. Mori et al., "A 4nm 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 ISSCC*, 2023, pp. 132–134.

[5] C.-T. Lin et al., "DIMCA: An area-efficient digital in-memory computing macro featuring approximate arithmetic hardware in 28 nm," *IEEE J. Solid-State Circuits*, vol. 59, no. 3, pp. 960–971, Mar. 2024.

[6] J. Oh, C.-T. Lin, and M. Seok, "D6CIM: 60.4-TOPS/W, 1.46-TOPS/mm^2, 1005-Kb/mm^2 digital 6T-SRAM-based compute-in-memory macro supporting 1-to-8b fixed-point arithmetic in 28-nm CMOS," in *Proc. IEEE 49th ESSCIRC*, 2023, pp. 413–416.

[7] H. Sharma et al., "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network," in *Proc. ACM/IEEE 45th ISCA*, 2018, pp. 764–775.

[8] S. Ryu et al., "BitBlade: Energy-efficient variable bit-precision hardware accelerator for quantized neural networks," *IEEE J. Solid-State Circuits*, vol. 57, no. 6, pp. 1924–1935, Jun. 2022.