A 1.23-GHz 16-kb Programmable and Generic Processing-in-SRAM Accelerator in 65nm

Amitesh Sridharan[1§], Shaahin Angizi[2§], Sai Kiran Cherupally[1], Fan Zhang[1], Jae-sun Seo[1], Deliang Fan[1]
[1]Arizona State University, USA, [2]New Jersey Institute of Technology, USA [§]both authors contribute equally

***Abstract*—We present a generic and programmable Processing-in-SRAM (PSRAM) accelerator chip design based on an 8T-SRAM array to accommodate a complete set of Boolean logic operations (e.g., NOR/NAND/XOR, both 2- and 3-input), majority, and full adder, for the first time, all in a single cycle. PSRAM provides the programmability required for in-memory computing platforms that could be used for various applications such as parallel vector operation, neural networks, and data encryption. The prototype design is implemented in a SRAM macro with size of 16 kb, demonstrating one of the fastest programmable in-memory computing system to date operating at 1.23 GHz. The 65nm prototype chip achieves system-level peak throughput of 1.2 TOPS, and energy-efficiency of 34.98 TOPS/W at 1.2V.**

***Index Terms*—In-memory computing (IMC), SRAM, programmability.**

I. Introduction

Traditional von-Neumann computing architectures, such as CPUs and GPUs, demonstrate limitations in memory bandwidth and energy efficiency. However, their high demand lies in their programmability and flexible functionality. Such platforms execute a wide spectrum of bit-wise logic and arithmetic operations. In this regard, recent application-specific processing-in-memory (PIM) designs suffer from the major challenge that their performance is intrinsically limited to one specific type of algorithm or application domain, which means that such PIM platforms cannot keep pace with rapidly evolving software algorithms [1]. To overcome this limitation, state-of-the-art generic and programmable PIM architectures (e.g., [2]) exploit alternatives to conventional bit-parallel algorithms. For example, it is possible to realize arithmetic operations using bit-serial algorithms. However, it comes at a cost of high latency and more intermediate data write-back if multiple computing cycles are needed for basic in-memory Boolean logic functions [1]–[5].

To address this challenge, we propose a programmable processing-in-SRAM accelerator (PSRAM) that combines the PIM computation efficacy with the programmability. More importantly, for the first time, PSRAM realizes a complete set of Boolean operations (both 2- and 3-input), majority, and full adder in only one single memory cycle, demonstrating one of the fastest in-memory computing macros to date operating at 1.23 GHz. Furthermore, our one-cycle in-memory Boolean logic design also eliminates the redundant intermediate data write-back operations for 3-input logic and full adder that typically need multiple cycles with extra latency and energy in prior multi-cycle in-memory logic designs [1]–[5]. We demonstrate PSRAM for three applications: bulk bitwise vector operations, low-precision deep learning acceleration, and the Advanced Encryption Standard (AES) computation.

II. Proposed PSRAM Design

The proposed PSRAM leverages the charge-sharing feature of the 8T SRAM cell on Read Bit-Line (RBL) and elevates it to implement 2-input and 3-input Boolean logic between two or three selected rows in a single memory read cycle. The key idea comes from the observation that certain discharge rate of the precharged RBL is determined by the data value stored in the simultaneously selected memory cells attached to the same bit-line. For instance, by activating three memory rows via Read Word Lines (RWL), e.g., RWL0-RWL2 (Fig. 1), if $S_{0,0}$, $S_{1,0}$, and $S_{2,0}$ memory cells all store '0's, then the read access transistors (T8) remain OFF, and the RBL precharged voltage does not discharge. On the other side, if all cells store '1's, the RBL voltage will rapidly discharge through T8s. Similarly, based on different combinations of the values stored in those memory cells, the discharged voltage value will be different if sampled at a preset frequency and VDD, which could be sensed by our follow up 'logic-SA' design to implement different logic functions through selecting different voltage references. Theoretically, there will be four different voltage levels based on all possible combinations of three memory cell data in the same bit-line. In our design, to yield a sufficiently large sense margin, as shown in Monte Carlo simulations (Fig. 5), the read path transistor (T7 and T8) size is designed to be 3× as shown in Fig. 1.

To implement a programmable logic function, a new re-configurable logic-SA is designed as in Fig. 1. It consists of three sub-SAs with voltage references (i.e., $V_{Ref1}<V_{Ref2}<V_{Ref3}$), each dedicated to distinct logic functions. In this way, by activating three memory rows (i.e., input operand vectors) at the same time, each sub-SA performs a neat voltage comparison between the reference voltage and the discharged RBL voltages (w.r.t. different discharge rate corresponding to stored memory cell data), which respectively generates (N)OR3, (MAJ)MIN, and (N)AND3 logic output (complementary SA), and more importantly, at the same time.

A novel single-cycle in-SRAM XOR3 (full adder's Sum) logic is developed through an interesting observation as shown in the bottom-right truth table of Fig. 1. When the majority function (MAJ) output (green box in the truth table) is '0', the corresponding XOR3's output is the same as the OR3's output. When the majority function output is '1', XOR3's output can be achieved through AND3 as highlighted by the purple box. Based on our last paragraph description, our logic-SA could simultaneously get the OR3, MAJ and AND3 logic outputs, then we propose to design the XOR3 logic through a two-transistor 2:1 multiplexer (with MAJ output as the

ESSCIRC 2022- IEEE 48th European Solid State Circuits Conference (ESSCIRC) | 978-1-6654-8494-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ESSCIRC55480.2022.9911440

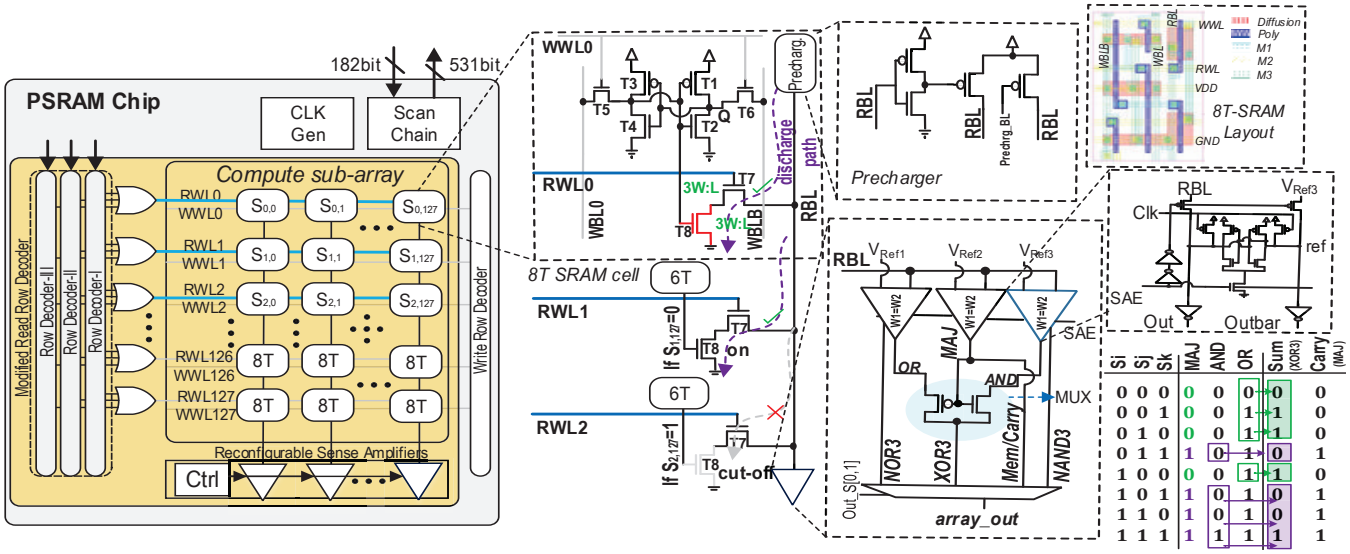

Fig. 1: PSRAM chip with 8T SRAM cell as the operand memory and the proposed single-cycle logic-SA design.

selector) circuit highlighted in the proposed reconfigurable logic-SA. The Boolean logic of in-memory XOR3 can be given as $XOR3 = MAJ(S_i, S_j, S_k).AND(S_i, S_j, S_k) + MIN(S_i, S_j, S_k).OR(S_i, S_j, S_k)$. In this way, assuming three vector operands are pre-stored in the memory, parallel in-memory full adder logic can be implemented for the first time in a single memory cycle, where MAJ and XOR3 outputs generate the carry-out and Sum signals, respectively. The two-input bit-wise operations will be readily implemented by initializing one row to '0'/'1'. All in-memory logic simulations are first shown in Fig. 2, showing corresponding functionality.

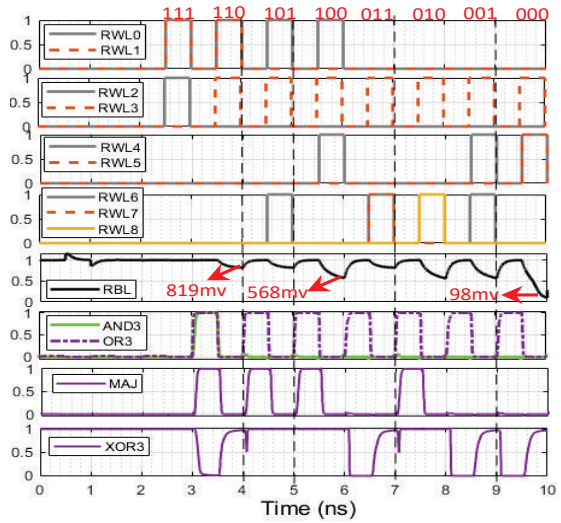

Fig. 2: In-memory logic simulation waveforms.

III. Measurement Results

A. Performance Measurement

We prototyped the PSRAM macro (128×128) in TSMC 65nm CMOS (Fig. 3). The macro has a 16-kb capacity and occupies 0.17 mm^2 (with decoder) in the chip floorplan. The

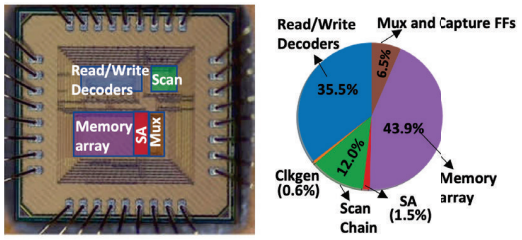

Fig. 3: Die micrograph and core area breakdown.

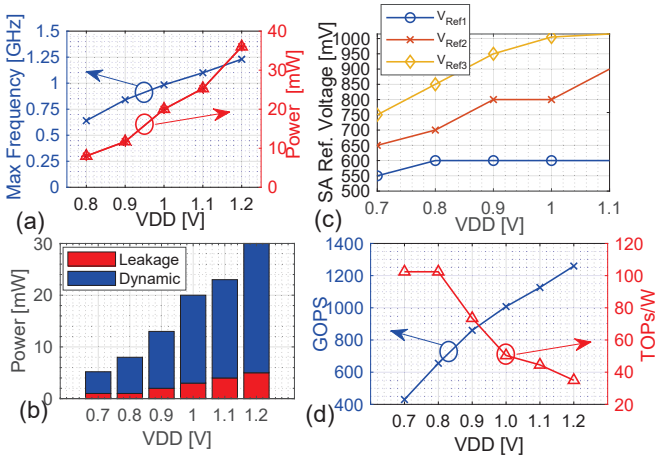

Fig. 4: (a) Frequency scaling over different VDDs, (b) Static and dynamic power consumption, (c) V_{Ref} scaling over different VDDs, and (d) throughput scaling over different VDDs.

bit-cell has an area of 4.56 μm^2 (1080 F^2 when scaled according to feature size), which is designed using logic rules. For efficient integration, the logic-SAs are pitch matched w.r.t. the column and occupy 3.4% of the array size (0.082 mm^2). The complete core area breakdown is shown in Fig. 3. The PSRAM macro consumes 36 pJ (includes power consumed by all components on the die) and takes 813 ps to generate 512

outputs of the complete 3 input logic set (AND3, XOR3, OR3, MAJ). This represents a peak throughput of 2×128×4/813ps = 1259.52 GOPs at 1.2V supply and a compute density of 583.12 GOPS/mm^2. PSRAM achieves a significant speedup of 4-157× when compared to state-of-the-art in-memory computing works [1]–[4]. We report the maximum frequency, power consumption and throughput w.r.t. different VDDs in Fig. 4.

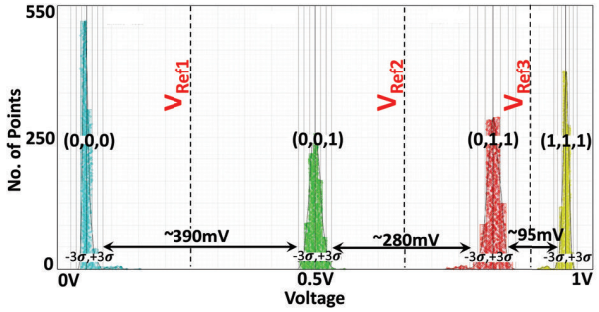

Fig. 5: Monte-Carlo simulation.

TABLE I: Reference voltage ranges measured on chip.

VDD/V_{Ref}	V_{Ref1}(V)	V_{Ref2}(V)	V_{Ref3}(V)
0.7V @ 0.42Ghz	509m-546m	603m-647m	658m-693m
0.8V @ 0.64Ghz	452m-616m	620m-733m	745m-780m
0.9V @ 0.84Ghz	414m-661m	669m-750m	829m-889m
1.0V @ 0.984Ghz	503m-711m	735m-902m	908m-995m
1.1V @ 1.1Ghz	550m-754m	760m-994m	999m-1.083
1.2V @ 1.23Ghz	554m-790m	815m-1.08	1.09-1.16

B. SA reference voltage(V_{Ref}) analysis

The RBL sense margins are first tested through post-layout Monte Carlo simulations in Cadence Spectre for the four possible sensing voltages, as shown in Fig. 5, where the sensing margin is reported considering both process (inter-die) and mismatch variations (intra-die) for core VDD (1.0 V) at 1 GHz. During the chip measurements, off-chip voltage references are provided (V_{Ref}) to the SAs. To conduct the V_{Ref} variation analysis on chip, we test all 128 bit-lines, 100 times, for all possible bit value combinations in memory. 10 chips are tested and we report all the reference voltage ranges at different VDDs and the corresponding maximum frequencies with zero logic errors in Table I. It is found that at lower voltages the maximum operating frequency is limited by the reduction of V_{Ref} ranges. A higher VDD also yields a larger sensing margin.

IV. Applications

• **Case Study I: Bulk Bitwise Boolean Vector Operations.** The PSRAM could be leveraged to implement bulk bitwise Boolean logic operations efficiently between vectors stored in the same memory sub-array. This can lead to efficient re-use of the internal memory bandwidth. Table II compares the latency for a set of vector operations of interest, implemented by three generic PIM designs. We achieve the best performance of each design, where input vectors A(a_0a_1...) B(b_0b_1...) and C(c_0c_1...) are stored in separate rows of the memory. We draw two conclusions from Table II. Firstly, our PSRAM is the only design that supports a full-set of Boolean logic (both 2-input and 3-input) and integer operations. Second, due to the complexity of some operations (e.g., ADD/SUB/MULT), they cannot be implemented in a time-efficient manner by the prior designs [2], [4], while PSRAM outperforms all prior works in latency.

TABLE II: Latency comparison of vector Boolean logic operations supported by PSRAM and prior accelerators.

Parameters	JSSC'18 [4]	JSSC'20 [2]	PSRAM
Capacity (KB)	8	16	2
Technology (nm)	40	28	65
Frequency (GHz)	0.029	0.475	1.23
NOT (ns / # of Cycle)	34.72 / 1	2.1 / 1	0.81 / 1
NAND2 (ns / # of Cycle)	34.72 / 1	2.1 / 1	0.81 / 1
NAND3 (ns / # of Cycle)	69.44 / 2	4.2 / 2	0.81 / 1
NOR2 (ns / # of Cycle)	34.72 / 1	2.1 / 1	0.81 / 1
NOR3 (ns / # of Cycle)	69.44 / 2	4.2 / 2	0.81 / 1
X(N)OR2 (ns / # of Cycle)	34.72 / 1	2.1 / 1	0.81 / 1
XOR3 (ns / # of Cycle)	69.44 / 2	4.2 / 2	0.81 / 1
Majority (ns / # of Cycle)	n/a	n/a	0.81 / 1
FULL-ADD (ns / # of Cycle)	69.44 / 2	4.2 / 2	0.81 / 1
FULL-SUB (ns / # of Cycle)	69.44 / 2	4.2 / 2	1.62 / 2
ADD-RCA (4-bit) (ns # of Cycle)	n/a	n/a	3.24 / 4
ADD-CSA (4-bit) (ns # of Cycle)	n/a	n/a	4.05 / 5
ADD-Serial* (4-bit) (ns)	173.6	10.5	4.05
SUB-Serial† (4-bit) (ns)	312.48	18.9	7.29
MULT-Serial‡ (4-bit) (ns)	1180.48	71.4	27.54
MULT-Serial (8-bit) (ns)	3541.44	214.2	82.62

*N+1 cycles, †2N+1 cycles, ‡N^2+ 5N-2 cycles

• **Case Study II: Binary-Weight Neural Networks.** We also implement the binary-weight neural network (BWNN) with various weight configurations for AlexNet and report the energy, latency and other performance metrics in Table III and Fig. 7. The general HW/SW framework developed for BWNN consists of image and kernel banks, and PSRAM sub-arrays. Weights and activation are constantly quantized to 1-bit and q-bit using the same method as [6], respectively, and then mapped to the parallel PSRAM sub-arrays. The top-1 accuracy after quantization on ImageNet dataset is reported in Fig. 7. For hardware mapping, considering n-activated PSRAM chips with the size of 128×128 (Fig. 6), each sub-array can handle the parallel ADD/SUB (multiply-and-accumulate operations are converted to ADD/SUB in BWNNs) of up to 128 elements of m-bit ($2m \leq 128$) and so accelerator could process n×128 elements simultaneously within computational sub-arrays to maximize the throughput. The memory sub-array data mapping for PSRAM is depicted in Fig. 6. We reserve four rows for Carry results initialized by zero and up to 32 rows for Sum results. Every pair of corresponding elements to be added together is aligned in the same bit-line. Herein, channel 1 (Ch1) and Ch2 should be aligned in the same sub-array. With m=32-bit, Ch1 elements occupy the first 32 rows of the sub-array followed by Ch2 in the next 32 rows.

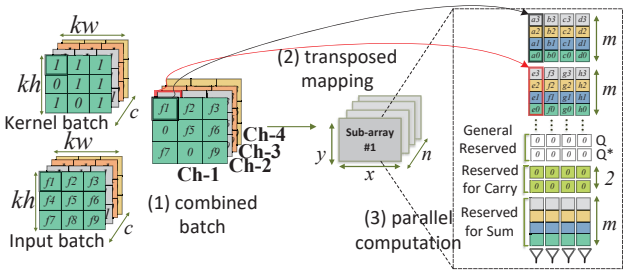

Fig. 6: BWNN hardware mapping.

The addition algorithm starts bit-by-bit from the LSBs of

TABLE III: Comparison with state-of-the-art SRAM based PIM accelerators.

		BWNN Accelerators		Generic Accelerators	
Reference	PSRAM	JSSC'19 [1]	JSSC'19 [3]	JSSC'20 [2]	JSSC'18 [4]
Technology	65nm	65nm	65nm	28nm	40nm
Bit cell Density	8T	10T	8T	8T Transposable	10T
Supply Voltage	0.8-1.2V	0.8-1.2V	0.68-1.2V	0.6 – 1.1V	0.5-0.9V
Max Frequency	1230MHz (1.2V)	5MHz	100MHz	475MHz (1.1V)	28.8MHz (0.7V)
SRAM Macro Size	2KB	2KB	4.8KB	16 KB	8KB
Performance (GOPS)	1259.52	8	295	32.7	14.7
Performance per unit area (GOPS/mm^2)	583.12	126	23.4	27.3	70
Energy-Efficiency (TOPS/W)	34.98	40.3	20.6	5.27 (add) 0.55 (mult.)	31.28
Reconfigurable	Programmable	N/A	N/A	Programmable	N/A

[1] We assume 2 operations (OPs) per NAND3/XOR3/X(N)OR3/NOR3 (cascaded logic), similar to MAC (1 mult. + 1 add).

the two words and continues towards MSBs. For evaluation, a 7-layer BWNN is adopted with distinct weight configurations of <W:I>: <1:1>, <1:2>, <1:8>. Our evaluation result reported in Fig. 7 shows that PSRAM can process AlexNet on average with 35 mJ energy per inference and ∼0.5 ms latency. The process energy and latency include the amount required by multiple PSRAM chips working as a whole entity. More detailed performance comparison with other recent SRAM based PIM designs are reported in Table III.

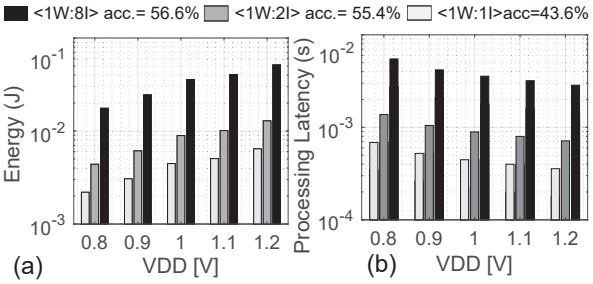

Fig. 7: (a) PSRAM energy consumption and (b) processing time for running the AlexNet (ImageNet dataset).

- **Case Study III: Data Encryption.** We further take the Advanced Encryption Standard (AES) data encryption algorithm as the third case-study. To facilitate working with input data (with a standard input length of 128 bits), each input byte data is distributed into 8-bit such that eight PSRAM sub-arrays are filled by 4×4 bit-matrices [7]. After mapping, PSRAM supports the required AES bulk bit-wise operations to accelerate each transformations inside the memory. As shown in Fig. 8, all AES transformations are mainly based on (N)AND and XOR operations that are fully supported in PSRAM. In SubBytes, MixColumns, and AddRoundKey stages, parallel in-memory XOR2 and (N)AND2 operations contribute to more than 90% of the operations. In ShiftRows stage, state matrix will undergo a cyclical shift operation by

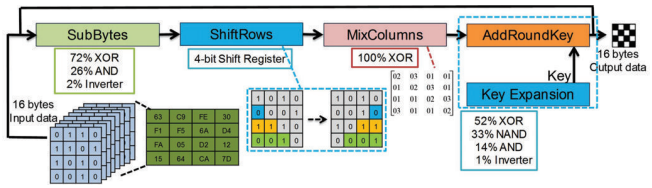

Fig. 8: AES block diagram with the gate utilization.

a certain offset. We use the 128-bit AES software implementation as the baseline from [4], a 350nm ASIC [8], and a 40nm ASIC [4] designs for comparison. Table IV shows that PSRAM achieves the highest speed-up over baseline. This mainly comes from the massively-parallel and high throughput XOR operation supported in PSRAM.

TABLE IV: 128-bit AES performance.

Platforms	#Cycles	Freq. (MHz)	Time (μS) (Norm.)	Energy (nJ) (Norm.)
Baseline [4]	6358	24	265 (1x)	64.2 (1x)
ASIC [8]	5429	0.847	6410 (24x)	10259 (160x)
Recryptor [4]	726	28.8	25.2 (0.1x)	7.05 (0.11x)
PSRAM	718	1230	0.58 (0.002x)	19.21(0.3x)

V. Conclusion

In this work, we present a programmable PSRAM chip design in TSMC 65nm CMOS technology. For the first time, the PSRAM could execute a complete set of Boolean logic vector operations (i.e., NOR/NAND/XOR, both 2- and 3-input), majority, and full adder, all in a single memory cycle. We also demonstrate three case studies leveraging our PSRAM design, including parallel vector operation, neural networks, data encryption, etc. PSRAM paves a new path towards the generic, programmable and fast in-SRAM computing.

Acknowledgements

This work is supported in part by NSF grants 2003749 and 2144751, by SRC and DARPA.

References

[1] A. Biswas *et al.*, "Conv-sram: An energy-efficient sram with in-memory dot-product computation for low-power convolutional neural networks," *IEEE JSSC*, 2018.

[2] J. Wang *et al.*, "A 28-nm compute SRAM with bit-serial logic/arithmetic operations for programmable in-memory vector computing," *IEEE JSSC*, 2020.

[3] H. Valavi *et al.*, "A 64-tile 2.4-mb in-memory-computing cnn accelerator employing charge-domain compute," *IEEE JSSC*, 2019.

[4] Y. Zhang *et al.*, "Recryptor: A reconfigurable cryptographic cortex-m0 processor with in-memory and near-memory computing for iot security," *IEEE JSSC*, 2018.

[5] J. Yue *et al.*, "14.3 a 65nm computing-in-memory-based cnn processor with 2.9-to-35.8 tops/w system energy efficiency using dynamic-sparsity performance-scaling architecture and energy-efficient inter/intra-macro data reuse," in *IEEE ISSCC*, 2020.

[6] J. Faraone *et al.*, "Syq: Learning symmetric quantization for efficient deep neural networks," in *CVPR*, 2018.

[7] S. Mathew *et al.*, "53 Gbps native GF$(2^4)^2$ composite-field AES-encrypt/decrypt accelerator for content-protection in 45nm high-performance microprocessors," in *IEEE Symp. on VLSI*.

[8] M. Hutter *et al.*, "A cryptographic processor for low-resource devices: Canning ecdsa and aes like sardines," in *IFIP*, 2011.