15.1 A Programmable Neural-Network Inference Accelerator Based on Scalable In-Memory Computing

Hongyang Jia, Murat Ozatay*, Yinqi Tang*, Hossein Valavi*, Rakshit Pathak*, Jinseok Lee, Naveen Verma

Princeton University, Princeton, NJ

*Equally Credited Authors (ECAs)

This paper presents a scalable neural-network (NN) inference accelerator in 16nm, based on an array of programmable cores employing mixed-signal In-Memory Computing (IMC), digital Near-Memory Computing (NMC), and localized buffering/control. IMC achieves high energy efficiency and throughput for matrix-vector multiplications (MVMs), which dominate NNs; but, scalability poses numerous challenges, both technologically, going to advanced nodes to maintain gains over digital architectures, and architecturally, for full execution of diverse NNs. Recent demonstrations have explored integrating IMC in programmable processors [1,2], but have not achieved IMC efficiency and throughput for full executions. The central challenge is drastically different physical design points and associated tradeoffs incurred by IMC compared to digital engines. Namely, IMC substantially increases compute energy efficiency and HW density/parallelism, but retains the overheads of HW virtualization (state and data swapping/buffering/communication across spatial/temporal computation mappings). The demonstrated architecture is co-designed with SW-mapping algorithms (encapsulated in a custom graph compiler), to provide efficiency across a broad range of mapping strategies, to overcome these overheads.

Figure 15.1.1 shows the demonstrated array-based architecture, comprised of: (1) a 4×4 array of Compute In-Memory-Unit (CIMU) cores; (2) an On-Chip Network (OCN) between cores; (3) buffers, control circuits, and off-chip interfaces for testability of the architecture. The design employs an additional weight buffer (not included in current prototype) with a dedicated weight-loading network to the CIMUs, to provide flexibility in mapping weights to CIMU cores for maximizing utilization, and to mitigate off-chip weight accesses for a range of NN models. These factors are considered together with the architectural design, to adequately address weight-swapping overheads.

The CIMU (microarchitecture described later), comprises: (1) an IMC engine for MVMs, called the Compute-In-Memory Array (CIMA); (2) an NMC digital SIMD with custom instruction set, for flexible element-wise operations; (3) buffering and control for enabling a wide range of NN dataflows. Each CIMU core provides a high-level of configurability, that is abstracted into a SW library of instructions for interfacing with the compiler, and where instructions can thus also be added prospectively [library includes single/fused instructions, e.g., element mult/add, h(•) activation, (N-step convolutional stride + MVM + batch norm. + h(•) activation + max. pool), (dense + MVM)].

The OCN consists of routing channels within network in/out blocks, and a switch block, which provides flexibility via a disjoint architecture. The OCN works with configurable CIMU input/output ports to optimize data structuring to/from the IMC engine, to maximize data locality across MVM dimensionalities and tensor depth/pixel indices. The OCN routing channels consist of bidirectional wire pairs, to ease repeater/pipeline-FF insertion, while providing ample density.

Figure 15.1.2 shows the CIMU microarchitecture. Data is received from the OCN into one of two buffers: (1) the input buffer, which configurably provides data to the CIMA; (2) the shortcut buffer, which bypasses the CIMA, providing data directly to the NMC digital SIMD for element-wise computations on separate and/or convergent NN activation paths. The central block is the CIMA, which consists of a mixed-signal 1152(row)×256(col.) IMC macro for multibit-element MVMs. The CIMA employs a variant of fully row/column-parallel computation, based on metal-fringing capacitors [3]. Each multiplying bit cell (M-BC) drives its capacitor with a 1b digital multiplication (XNOR/AND), involving inputted activation data (IA/IAb) and stored weight data (W/Wb). This causes charge redistribution across M-BC capacitors in a column to give an inner product between binary vectors on the compute line (CL). This yields low compute noise (nonlinearity, variability), since multiplication is digital and accumulation involves only capacitors, which are defined by high lithographic precision. An 8b SAR ADC digitizes the CL and enables extension to multibit activations/weights, via bit-parallel/bit-serial (BP/BS) computation [1], where weight bits are mapped to parallel columns and activation bits are inputted serially. Each column thus performs binary-vector inner products, with a multibit-vector inner product simply achieved by digital bit shifting (for proper binary weighting) and summing across the column-ADC outputs. Digital BP/BS operations occur in the dedicated NMC BPBS SIMD module, which is optimized for 1-to-8b weights/activations, and further programmable element-wise operations (e.g., arbitrary activations functions) occur in the NMC CMPT SIMD module.

Figure 15.1.3 shows a sample of the operations enabled by CIMU configurability and the SW instruction libraries. In addition to temporal mapping of NN layers, the architecture provides extensive support for spatial mapping (loop unrolling). Given the high HW density/parallelism of IMC, this provides a range of mapping options for HW utilization, beyond typical replication strategies, which incur excessive state-loading overheads due to state replication across engines. To support spatial mapping of NN layers, various approaches for receiving and sequencing input activations for IMC computation are shown, enabled by configurability in the input and shortcut buffers, including: (1) high-bandwidth inputting for dense layers; (2) bandwidth-reduced inputting and line buffering for convolutional layers; (3) feed-forward and recurrent inputting, as well as output-element computation, for memory-augmented layers; (4) parallel inputting and buffering of NN and shortcut-path activations, as well as activation summing. A range of other activation receiving/sequencing approaches are supported, along with configurability in parameters of the approaches above. The spatial mappings, enabled by such CIMU-localized buffering, mitigate the need for centralized buffering (e.g., requiring just the small activation buffers included in the periphery of the chip).

As shown in Fig. 15.1.4, the architecture also supports spatial mapping within NN layers, both for mitigating data swapping/movement overheads and for enabling NN model scalability. For instance, output-tensor depth (number of output channels) can be extended by OCN routing of input activations to multiple CIMU. Input-tensor depth (number of input channels) can be extended via short, high-bandwidth face-to-face connections between the outputs of adjacent CIMUs, and further extended by summing the partial pre-activations from two CIMUs by a third CIMU. Efficient scale-up of layer computations in this manner enables a balance in the IMC core dimensions (found by mapping a range of NN benchmarks), where coarse granularity benefits IMC parallelism and energy, and fine granularity benefits efficient computation mapping.

Figure 15.1.7 shows a die photo of the 25mm^2 prototype in 16nm, and Fig. 15.1.5 shows measurements of the basic operation. CIMA-column transfer functions on the left show low variability (error bars are across 256 CIMU columns), low noise (0.68 LSB_{RMS}), and high linearity (INL < 1 LSB). BP/BS-computed MVM data on the right shows SNR with respect to standard integer compute, exhibiting excellent match between chip measurements and bit-true simulations, which model only quantization effects. This indicates that the dominant chip noise is quantization, which can be robustly modeled at the SW level, as typically done with quantization-aware NN training algorithms. Chip energy measurements below are for 4b weights/activations, most notably showing a CIMA energy per 1152-dimensional inner product of 19pJ.

Figure 15.1.6 shows the demonstration of an 11-layer CNN for CIFAR-10 classification and ResNet-50 for ImageNet classification (last stack mapping), achieving performance and layer-wise intermediate outputs matching bit-true software. The comparison table below shows this to be the only IMC demonstration for scalable NN execution, while achieving peak efficiency and throughput exceeding previously-reported accelerators.

Acknowledgements:
This work is partially supported by the School of Engineering and Applied Science at Princeton University through the generosity of William Addy '82. The authors thank J. Puscar (UCSD), Prof. I. Galton (UCSD), A. Rovinski (U. of M.) and Prof. R. Dreslinski (U. of M.) for their help with the PLL.

References:
[1] H. Jia et al., "A Programmable Heterogeneous Microprocessor Based on Bit-Scalable In-Memory Computing," *IEEE JSSC*, vol. 55, No 9, pp 2609-2621, 2020.
[2] J. Wang et al., "A Compute SRAM with Bit-Serial Integer/Floating-Point Operations for Programmable In-Memory Vector Acceleration," *ISSCC*, pp. 224-226, 2019.
[3] H. Valavi et al., "A 64-Tile 2.4-Mb In-Memory-Computing CNN Accelerator Employing Charge-Domain Compute," *IEEE JSSC*, vol. 54, No 6, pp 1789-1799, 2019.
[4] Y. Chen, T. Krishna, J. Emer and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *ISSCC*, pp. 262-263, Feb. 2016.
[5] D. Bankman, L. Yang, B. Moons, M. Verhelst and B. Murmann, "An always-on 3.8μJ/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28nm CMOS," *ISSCC*, pp. 222-224, Feb. 2018.
[6] Y. Jiao et al., "A 12nm Programmable Convolution-Efficient Neural-Processing-Unit Chip Achieving 825TOPS," *ISSCC*, pp. 136-140, Feb. 2020.

 978-1-7281-9549-0/21/$31.00

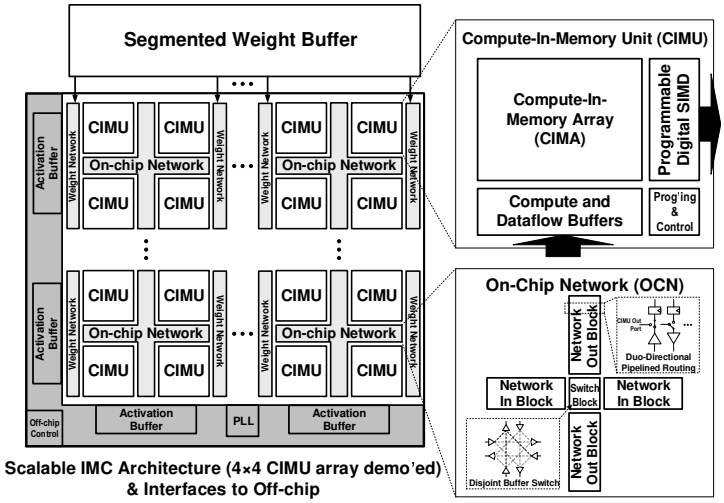


Figure 15.1.1: Architectural block diagram of the scalable in-memory computing (IMC) neural-network (NN) accelerator, comprising array of Compute-In-Memory-Unit (CIMU) cores and on-chip network (OCN), as well as interface circuitry for testability.

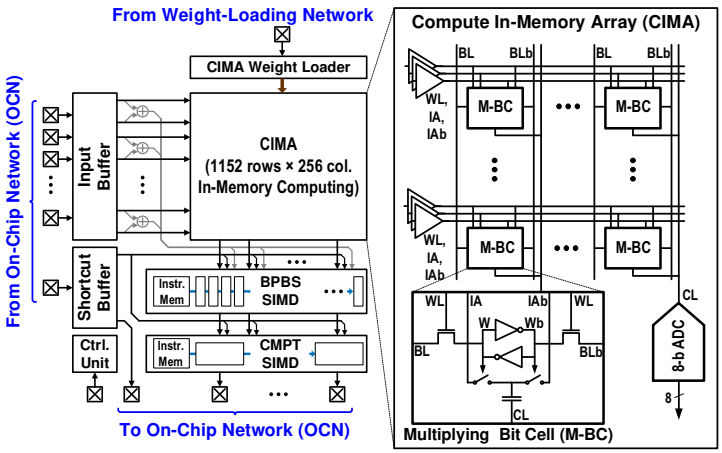


Figure 15.1.2: Details of CIMU core, showing internal datapath comprising buffering/control logic, Compute-In-Memory-Array (CIMA) mixed-signal IMC engine, and digital near-memory-computing SIMD units.

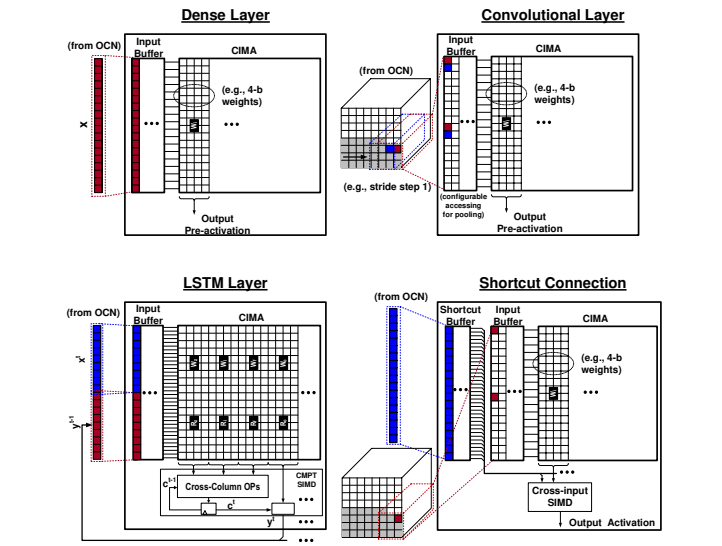


Figure 15.1.3: Examples of configurable input-receiving/sequencing supports to enhance efficiency of spatial mappings, necessary for ensuring high hardware utilization with minimal state-swapping overheads.

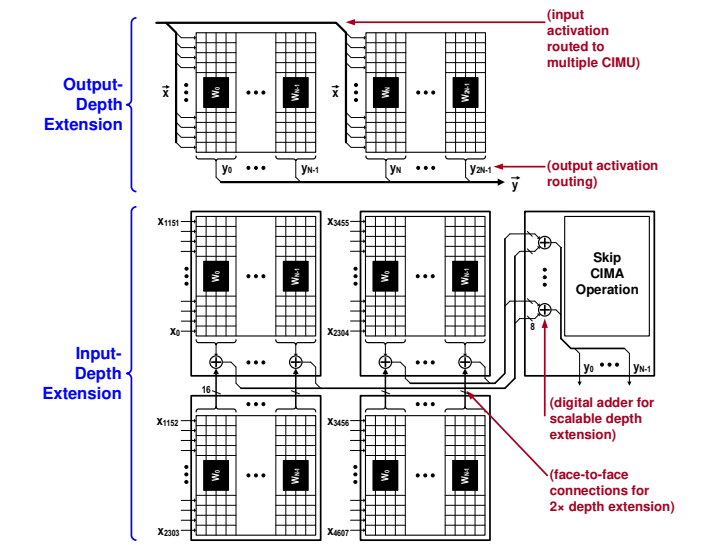
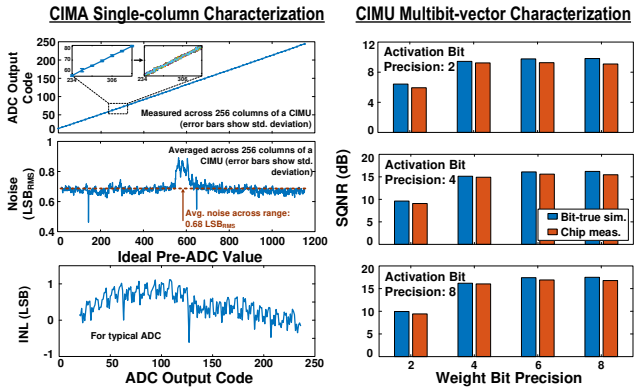
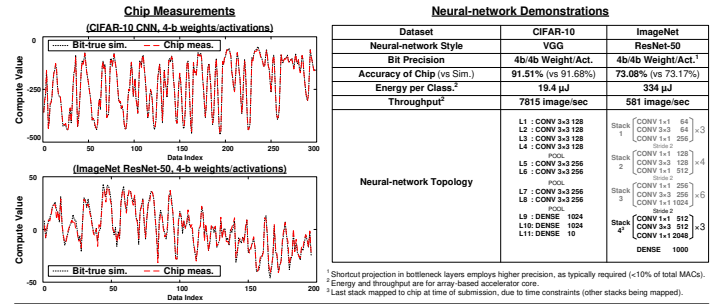


Figure 15.1.4: Illustration of spatial-mapping techniques to support NN model scalability and execution efficiency, showing (top) output-tensor depth extension, and (bottom) input-tensor depth extension.



Chip Summary			
Technology (nm)	16	F_{CLK} (MHz)	200
Voltage (V)	0.8	Total Area (mm²)	25
Energy Breakdown (measured using 3×3×128 CNN kernel w/ 4b/4b weight/activation)			
CIMA (pJ/output act.)	19.08	BPBS SIMD (pJ/output act.)	3.89
CIMA Write (pJ/bit)	0.23	CMPT SIMD (pJ/output act.)	0.60
IA Buff. (pJ/output act.)	8.60	OCN (pJ/bit/seg.)	0.27

Figure 15.1.5: Basic chip operation measurements, showing CIMA column-compute characterization, multibit-vector CIMU operation, and detailed energy breakdown.



Neural-network Demonstrations

Dataset	CIFAR-10	ImageNet
Neural-network Style	VGG	ResNet-50
Bit Precision	4b/4b Weight/Act.	4b/4b Weight/Act.[1]
Accuracy of Chip (vs Sim.)	91.51% (vs 91.68%)	73.08% (vs 73.17%)
Energy per Class.[2]	19.4 µJ	334 µJ
Throughput[2]	7815 image/sec	581 image/sec
Neural-network Topology	L1 : CONV 3×3 128 L2 : CONV 3×3 128 L3 : CONV 3×3 128 L4 : CONV 3×3 128 POOL L5 : CONV 3×3 256 L6 : CONV 3×3 256 POOL L7 : CONV 3×3 256 L8 : CONV 3×3 256 POOL L9 : DENSE 1024 L10: DENSE 1024 L11: DENSE 10	Stack 1: CONV 1×1 64, CONV 3×3 64, CONV 1×1 256 ×3 Stride 2 Stack 2: CONV 1×1 128, CONV 3×3 128, CONV 1×1 512 ×4 Stack 3: CONV 1×1 256, CONV 3×3 256, CONV 1×1 1024 ×6 Stride 2 Stack 4[3]: CONV 1×1 512, CONV 3×3 512, CONV 1×1 2048 ×3 DENSE 1000

[1] Shortcut projection in bottleneck layers employs higher precision, as typically required (<10% of total MACs).
[2] Energy and throughput are for array-based accelerator core.
[3] Last stack mapped to chip at time of submission, due to time constraints (other stacks being mapped).

Comparison Table

	Not In-memory Computing			In-memory Computing					
	Chen, ISSCC'16	Bankman, ISSCC'18	Jiao, ISSCC'20	Guo, VLSI'19	Wang, ISSCC'19	Yue, ISSCC'20	Dong, ISSCC'20	Jia, JSSC'20	This work
Technology	65nm	28nm	12nm	65nm	28nm	65nm	7nm	65nm	16nm
Area (mm²)	16	6	709	6.2	2.7	9	0.0032	8.56	25
Voltage (V)	0.8 – 1.2	0.6 | 0.8	0.85	0.9 – 1.1	0.6 – 1.1	0.9 – 1.05	0.8 | 1.0	1.2 | 0.85	0.8
On-chip Memory	864 Kb	2.6 Mb	1536 Mb	64 Kb[1]	1 Mb[1]	16 Kb[1]	4 Kb[1]	576 Kb[1]	4.5 Mb[1]
Bit Precision	16 b	1 b	8 b | 16 b	1 | 1 | 3 b	Arbitrary	2 b – 8 b	4 b	1 – 8 b	1 – 8 b
Optimized for Execution Scalability	YES	NO	YES	NO	NO	NO	NO	NO	YES
Peak MAC Throughput (GOPS)[2]	1.92K	3.75 | 25	3.3M	38.4	131	36.15	372 | 455	54.6 | 136.6[3]	11.8K[4]
Peak MAC Energy Eff. (TOPS/W)[2]	1.33	48.3 | 33.4	11.8	0.73	4	79.38[4]	611 | 436	25 | 12[3]	121[3]
MAC Comp. Density (TOPS/mm²)[2]	0.192	0.0024 | 0.016	4.65	0.014	0.16	0.066	116 | 142	0.015 | 0.038[3]	2.67[3]
Application	CNN	CNN	CNN	RNN	Vector OPs	CNN, MLP	Matrix-Vec. Mult.	Matrix-Vec. Mult.	CNN,RNN, MLP, etc.

[1] Only consider memory conduct In-memory Computing [2] Normalized to 4-bit compute [3] Scale with number of bits used for input activation and weight [4] Number by exploring weight/input sparsity

Figure 15.1.6: System demonstrations and comparison summary, showing match between IMC derived outputs and ideal (bit-true) outputs, as well as advancement to scalable IMC architecture.

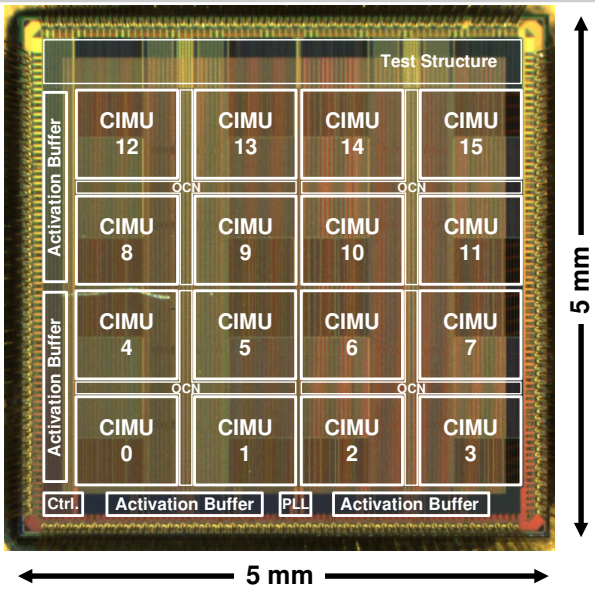


Figure 15.1.7: Die photo of prototype implemented in 16nm.

Additional References:

[7] R. Guo et al., "A 5.1pJ/Neuron 127.3us/Inference RNN-based Speech Recognition Processor using 16 Computing-in-Memory SRAM Macros in 65nm CMOS," *VLSI Symp. on Circuits, Kyoto*, pp. C120-C121, June 2019.

[8] J. Yue et al., "A 65nm Computing-in-Memory-Based CNN Processor with 2.9-to-35.8TOPS/W System Energy Efficiency Using Dynamic-Sparsity Performance-Scaling Architecture and Energy-Efficient Inter/Intra-Macro Data Reuse," *ISSCC*, pp. 234-236, Feb. 2020.

[9] Q. Dong et al., "A 351TOPS/W and 372.4GOPS Compute-in-Memory SRAM Macro in 7nm FinFET CMOS for Machine-Learning Applications," *ISSCC*, pp. 242-244, Feb. 2020.

978-1-7281-9549-0/21/$31.00