A 40-nm Programmable Heterogeneous SoC With 5.625/0.85 MB RRAM/SRAM for Accelerating Neuro-Symbolic AI Models

Che-Kai Liu, Zishen Wan, *Graduate Student Member, IEEE*, Young-Seok Noh, Mohamed Ibrahim, *Member, IEEE*, Samuel D. Spetalnick, Tushar Krishna, *Senior Member, IEEE*, Win-San Khwa, *Senior Member, IEEE*, Ashwin Sanjay Lele, *Member, IEEE*, Yu-Der Chih, Meng-Fan Chang, *Fellow, IEEE*, and Arijit Raychowdhury, *Fellow, IEEE*

***Abstract*—Neuro-symbolic (NeSy) artificial intelligence (AI) integrates neural learning with symbolic reasoning to enable data-efficient, interpretable, and generalizable intelligence, making it a promising paradigm for human-like cognition. However, the heterogeneous and dynamic execution patterns of NeSy models pose fundamental challenges to conventional AI hardware, which is typically optimized for dense matrix operations. This article presents a fully programmable heterogeneous system-on-chip (SoC) fabricated in 40-nm CMOS, specifically designed to accelerate a wide range of NeSy workloads. The architecture features: 1) integrated resistive RAM (RRAM) and static random access memory (SRAM) neural-symbolic data paths; 2) ultra-dense (4.80 Mb/mm^2), energy-efficient (0.247 pJ/b) RRAM macros with edge-triggered and prolonged sensing; 3) scheduler-informed power management; and 4) programming support for variable resolution, vector lengths, and batching. The SoC supports multiple NeSy paradigms and achieves up to 10.8 TOPS/W energy efficiency and 6.47× power reduction across reasoning benchmarks. This work demonstrates the first end-to-end silicon system for generalizable and efficient NeSy inference.**

***Index Terms*—Compute-near-memory (CNM), embedded nonvolatile memory (eNVM), neuro-symbolic (NeSy) artificial intelligence (AI), resistive RAM (RRAM), system-on-chip (SoC).**

I. Introduction

AS ARTIFICIAL intelligence (AI) systems increasingly permeate real-world applications, there is growing demand for models that are not only accurate and data-efficient, but also interpretable, adaptable, and capable of robust reasoning. Neuro-symbolic (NeSy) AI represents a promising paradigm that tightly integrates the statistical learning capability of neural networks with the structured reasoning of symbolic systems [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11]. By fusing these components, NeSy AI mirrors human cognitive processes: intuitive perception (system 1) and logical reasoning (system 2), where the former can be modeled with neural networks, and the latter with symbolic primitives (see Fig. 1).

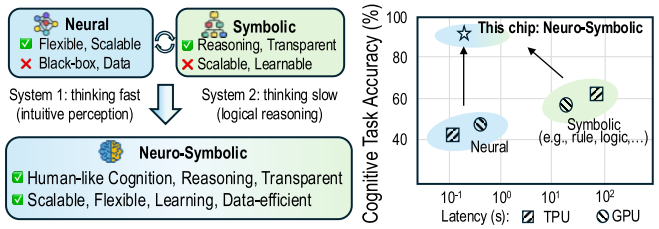


Fig. 1. NeSy AI integrates neural learning and symbolic reasoning, enabling enhanced cognitive intelligence.

NeSy AI has demonstrated significant promise across a wide range of cognitive applications, including complex question answering, abstract reasoning, probabilistic inference, user-intent prediction, and zero-shot concept acquisition [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22]. Compared to traditional neural models, NeSy systems offer several key advantages: improved interpretability, better generalization under distribution shifts, enhanced robustness to partial knowledge, and reduced data requirements. These characteristics make NeSy models particularly attractive for safety-critical and knowledge-rich domains such as agentic systems, robotics, and healthcare.

However, the computational patterns of NeSy models introduce challenges for hardware acceleration. First, their high memory demands of neural weights and symbolic representations require integration of high-density, on-chip storage memory. Second, the NeSy models consist of a diverse set of compute operators, which span neural operations (e.g., convolutions and attention) and symbolic primitives (e.g., logic operations, vector binding, and bundling) and are difficult to

Received 2 July 2025; revised 17 December 2025; accepted 22 January 2026. This article was approved by Associate Editor Ken Takeuchi. This work was supported in part by the CoCoSys, one of seven centers in Joint University Microelectronics Program (JUMP) 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA. *(Che-Kai Liu and Zishen Wan contributed equally to this work.) (Corresponding author: Che-Kai Liu.)*

Che-Kai Liu, Zishen Wan, Young-Seok Noh, Mohamed Ibrahim, Samuel D. Spetalnick, Tushar Krishna, and Arijit Raychowdhury are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: che-kai@gatech.edu).

Win-San Khwa and Meng-Fan Chang are with TSMC Corporate Research, Hsinchu 300-75, Taiwan.

Ashwin Sanjay Lele is with TSMC Corporate Research, San Jose, CA 95134 USA.

Yu-Der Chih is with TSMC Design Technology, Hsinchu 300-75, Taiwan.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSSC.2026.3658288.

Digital Object Identifier 10.1109/JSSC.2026.3658288

execute efficiently on current general matrix multiplication (GEMM)-optimized architectures. Third, the dynamic and irregular control flow of these workloads often results in poor hardware utilization and sub-optimized power management. These challenges of memory demands, kernel heterogeneity, and dynamic data flow necessitate a fundamental rethinking of hardware architecture to effectively support this emerging class of cognitive workloads.

In this work, we present a 40-nm programmable heterogeneous system-on-chip (SoC) test chip integrated with resistive RAM (RRAM) and static random access memory (SRAM), designed to accelerate NeSy AI. The SoC features: 1) compute-near-memory (CNM) datapaths based on ultra-dense RRAM (4.80 Mb/mm^2) and low-latency SRAM; 2) edge-triggered and prolonged sensing circuit that utilizes >75% sensing budget for establishing RRAM read current, achieving down-to 700-mV system operation with RRAM, and ultralow 0.247-pJ/bit sensing energy; 3) heterogeneously integrated symbolic and neural tiles on a network on chip (NoC) supporting diverse operators including vector-symbolic functions, logic functions, MACs, and activations; and 4) a software-defined power management (SDPM) mechanism guided by an off-chip scheduler and an on-chip microcontroller to adaptively manage power across workloads. The system achieves up to 10.8 TOPS/W energy efficiency and supports a broad range of NeSy models, demonstrating generalizability, flexibility, and efficiency across various NeSy paradigms.

This article, therefore, makes the following contributions.

1) A 40-nm programmable SoC that integrates RRAMs and SRAMs for accelerating NeSy AI models.
2) High-density RRAM macros (4.80 Mb/mm^2) with edge-triggered prolonged sensing circuits that enable energy-efficient inference (0.247 pJ/bit) and support for up to 5.625 MB of embedded non-volatile memory.
3) Heterogeneous compute support for both neural and symbolic kernels via configurable vector-length compute units and time-multiplexed symbolic processors.
4) SDPM that dynamically reconfigures resource usage based on workload characteristics, reducing system-level power consumption by up to 6.47×.
5) A unified very long instruction word (VLIW)-based programming interface that supports variable resolution, flexible batching, and diverse instruction types tailored for NeSy execution.

This SoC prototype demonstrates scalable, reconfigurable hardware support for a wide spectrum of NeSy models across use cases such as reasoning, decision-making, and concept acquisition. The proposed architecture bridges the gap between emerging cognitive AI algorithms and the limitations of today's AI hardware, enabling a new class of efficient and transparent cognitive systems.

II. NeSy Model and Characteristics

In this section, we first categorize key NeSy models that have been well established in existing literature and analyze their computational patterns (see Section II-A) and then identify system-level requirements for efficient hardware acceleration (see Section II-C).

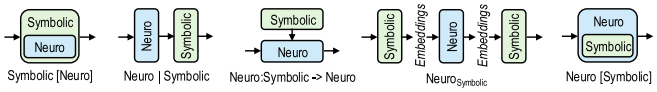


Fig. 2. NeSy model's dynamic execution graph.

TABLE I
Representative NeSy Workloads, Their Applications, and Advantages Compared to Neuro-Only Models

Neuro-Symbolic Workloads	Applications	Advantages vs. Neuro-only Model
Neuro-Vector-Symbolic Arch (NVSA) [21]	Abstract reasoning	Higher efficiency, better reasoning capability, transparency
Probabilistic Abduction Learning (LVRF) [27]	Probabilistic reasoning	Stronger OOD handling capability, flexibility, transparency
Multi-Input-Multi-Output (MIMONet) [28]	Simultaneous processing	Higher compositionality, lower latency
Probabilistic Abduction and Exe. (PrAE) [17]	Probabilistic abduction	Higher generalization, interpretability, robustness
Logical Neural Network (LNN) [29]	Full theorem prover	Higher interpretability, resilience to incomplete knowledge
Logical Tensor Network (LTN) [30]	Querying and reasoning	Higher data efficiency, comprehensibility, generalization
Neural Logic Machine (NLM) [31]	Relational reasoning	Higher logic reasoning and deduction capability
Shared Control of Assistive Robot [32]	User-intent prediction	Higher reasoning capability and robustness to partial information

A. NeSy Model Categorization

NeSy AI synergistically combines the learning capabilities of neural networks with the reasoning strengths of symbolic AI, enabling data-efficient learning and transparent, logic-driven decision-making. Compared to deep neural networks (DNNs), NeSy models demonstrate superior performance across a variety of cognitive tasks, including complex question answering [13], [15], abstract deduction [17], [21], decision making [23], [24], and logical reasoning [25], [26]. This makes NeSy AI a promising and emerging paradigm for achieving human-like fluid intelligence.

Neural and symbolic components can be integrated in multiple ways. Based on the structure of this integration, we categorize existing NeSy models into five representative paradigms, as illustrated in Fig. 2. Symbolic[Neuro] enhances symbolic reasoning with statistical learning from neural networks. Neuro|Symbolic refers to pipelined systems where neural and symbolic modules operate in sequence on complementary tasks. Neuro:Symbolic → Neuro incorporates symbolic rules into neural networks to guide the reasoning process. Neuro$_{\text{Symbolic}}$ maps symbolic rules into embeddings that serve as soft constraints on neural networks. Neuro[Symbolic] integrates symbolic reasoning as an internal subroutine within the neural architecture. Our programmable SoC is designed to flexibly support these paradigms through configurable resolution, vector lengths, and batching.

To analyze model diversity and corresponding hardware implications, we select eight state-of-the-art template NeSy models spanning the above paradigms (see Table I). Neuro-vector-symbolic architecture (NVSA) on spatial–temporal reasoning tasks [21], probabilistic abduction learning [learn-VRF (LVRF)] on probabilistic reasoning tasks [27], MIMONet on multi-input simultaneous processing tasks [28], probabilistic abduction and execution (PrAE) on probabilistic abduction tasks [17], logical neural network (LNN) on full theorem prover

TABLE II

NESY MODEL CATEGORIES AND THEIR HETEROGENEOUS NEURAL AND SYMBOLIC OPERATORS

Model Categories	Workloads	Neural Operation	Symbolic Operation	Memory Requirements
Symbolic[Neuro]	LNN [29]	Graph	FOL / Logical operation	2.5MB
Neuro|Symbolic	NVSA [21]	ConvNet	Vector Circular Convolution	5.4MB
	LVRF [27]	ConvNet	Vector Circular Convolution	4.8MB
	PrAE [17]	ConvNet	Vector Binding, Bundling	1.9MB
Neuro:Symbolic→Neuro	SharedControl [32]	ConvNet	Vector Binding, Bundling	4.2MB
$\text{Neuro}_{\text{Symbolic}}$	MIMONet [28]	ConvNet / Attention	Vector Binding, Bundling	5.6MB
Neuro[Symbolic]	LTN [30]	MLP	FOL / Logical operation	3.0MB
	NLM [31]	Sequential Tensor	FOL / Logical operation	4.6MB

tasks [29], logical tensor network (LTN) on querying and reasoning tasks [30], neural logic machine (NLM) on rational reasoning tasks [31], and shared control of assistive robot on user-intent prediction tasks [32]. These NeSy models feature a diverse range of neural and symbolic operations, such as convolution, sequential tensor, first-order logic, vector circular convolution, binding, bundling, and so on (see Table II). Although we will provide results on these eight algorithms, the SoC can support any NeSy algorithm that belongs to these model classes. Compared to pure-neural models, these NeSy models offer improved reasoning accuracy, interpretability, robustness, and adaptability across cognitive applications. Interested readers could refer to their References for details.

These NeSy models also span a rich set of operator types. In addition to standard neural kernels (such as convolution, fully connected layers, and attention), models such as NVSA and LVRF utilize vector circular convolution; PrAE, shared control, and MIMONet rely on vector binding and bundling; LNN, LTN, and NLM employ first-order logic operations. Our SoC is designed to flexibly support this diversity of neural and symbolic operators through integrated neuro and symbolic datapaths with a VLIW-based microcontroller, thus demonstrating an exciting new hardware paradigm that can natively support NeSy AI.

B. Symbolic Kernels

To address the diversity of symbolic kernels illustrated in Table II and Section II-C, the symbolic tiles incorporate specialized hardware acceleration. As detailed in Table III, these symbolic operations can be broadly categorized into bit-wise logic (typical of vector-symbolic architectures [17], [21], [27], [28], [32], [33]) and integer arithmetic (typical of first-order-logic algorithms [29], [30], [31]). Consequently, the symbolic tile features configurable datapath modes, implemented as NeSy MAC1, NeSy MAC2, and NeSy MAC3, to efficiently accelerate both the bit-wise manipulations and integer-based computations. Other specialized modules for symbolic reasoning include randomization, distance computing, and vector-folding registers. Further elaboration is provided in Section IV.

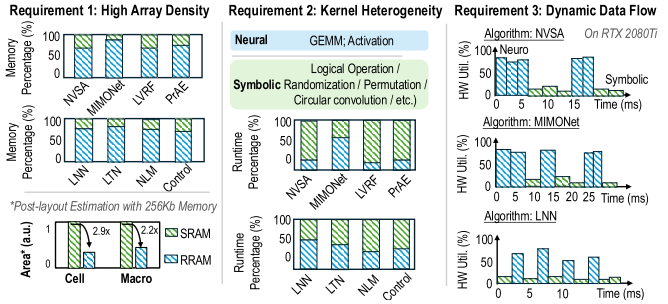


Fig. 3. NeSy model requirements. Compared with neuro-only models, NeSy models require high array density, high kernel heterogeneity, and dynamic data flow.

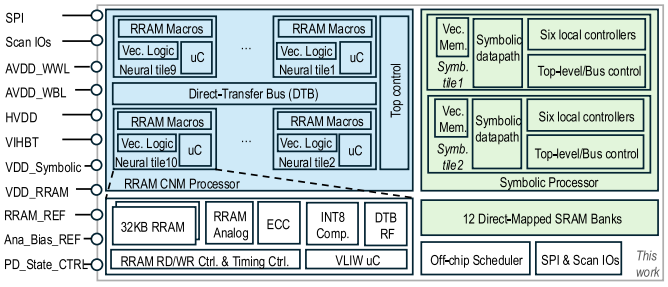


Fig. 4. Top-level architectural view of the NeSy test chip, featuring ten neural and two symbolic tiles for accelerating various NeSy workloads.

TABLE III

SAMPLING OF ACCELERATED SYMBOLIC KERNELS

Int. Arith.	HW Block	Bit-wise Logic	HW Block
$\forall$	Top-k	Circ. Conv.	NeSy MAC2
$\exists$	Top-k	Binding	NeSy MAC3
$\wedge$	NeSy MAC1	Bundling	NeSy MAC3
$\vee$	NeSy MAC1	Permutation	NeSy MAC3

C. Workload Characterization and Requirements

Compared with traditional neural networks, NeSy models exhibit unique memory, compute, and control-flow characteristics, making them difficult to support efficiently on off-the-shelf hardware platforms [1]. Fig. 3 illustrates the memory, operator, runtime, and hardware utilization characteristics of NeSy models running on the RTX2080Ti hardware system.

Requirement 1 (High Array Density): NeSy models often rely on neural weights and large symbolic codebooks to represent concepts and object compositions. These components demand substantial on-chip storage to avoid costly off-chip memory access, particularly on the edge. To support such workloads within constrained silicon area, the system needs to integrate high-density memory arrays. This motivates the inclusion of RRAM macros in our architecture, which can offer 2.9× cell-level and 2.2× macro-level density compared to SRAMs, and enables tight on-die integration of neuro and symbolic data without compromising capacity. High-density custom RRAM CNM macro will be described further in Section V.

Requirement 2 (Kernel Heterogeneity): Unlike traditional neural networks, NeSy models feature a diverse set of computation kernels, from neural primitives (e.g., convolution and

TABLE IV

HARDWARE INEFFICIENCY ANALYSIS. THE COMPUTE, MEMORY, AND COMMUNICATION CHARACTERISTICS OF REPRESENTATIVE NEURAL AND SYMBOLIC KERNELS ON THE CPU/GPU PLATFORM

	Neural Kernel		Symbolic Kernel	
	sgemm_nn	relu_nn	vectorized_elem	elementwise
Compute Throughput (%)	93.8	91.2	4.6	3.7
ALU Utilization (%)	88.5	46.9	8.9	6.3
L1 Cache Throughput (%)	81.3	84.6	25.7	10.2
L2 Cache Throughput (%)	20.5	18.9	34.0	22.9
L1 Cache Hit Rate (%)	45.0	49.1	30.2	31.1
L2 Cache Hit Rate (%)	84.7	63.3	46.9	36.0
DRAM BW Utilization (%)	16.0	22.8	88.1	79.6

activation) to symbolic primitives (e.g., circular convolution, binding, bundling, and logic). These symbolic kernels are typically inefficient on CPUs/GPUs, often becoming the system bottleneck. For instance, symbolic components account for 92.1%, 80.5%, and 34.2% of the total execution time in NVSA, PrAE, and MIMONet, respectively, running on an NVIDIA RTX 2080Ti GPU. Efficiently supporting this heterogeneous operator set requires hardware architectures that go beyond traditional GEMM-centric designs. Diverse kernel support will be described further in Section IV.

Requirement 3 (Dynamic Data Flow):

NeSy model execution often involves dynamic and irregular data flows, making efficient scheduling and resource utilization difficult. Symbolic reasoning is often dependent on outputs from neural perception modules, resulting in sequential execution patterns and hardware underutilization. Additionally, data flow and operation types can vary across tasks and reasoning steps. To address this, our system incorporates an adaptive workload-aware scheduler that is capable of handling complex dependencies, orchestrating heterogeneous kernels, and interleaving operations across tasks to improve parallelism and overall throughput. Table IV further quantifies the reason for hardware inefficiency. We analyze the compute and memory units behavior of representative neuro and symbolic kernels, based on Nsight Systems and Nsight Compute tools [34], [35]. The system inefficiencies mainly come from ALU under-utilization (<10%), low cache hit rate ($L1 < 20\%$ and $L2 < 40\%$), and massive data movement of symbolic operations, where dynamic random access memory (DRAM) bandwidth utilization is around 90%. Symbolic data transfer accounts for half of total latency, where >80% is from host to GPU, while neural kernels exhibit high utilization.

III. SOC ARCHITECTURE

This section presents the architecture of our heterogeneous SoC designed for NeSy models. We begin with an overview of the top-level system organization (see Section III-A), followed by memory design tradeoffs between RRAM and SRAM (see Section III-B). We then introduce our SDPM strategy for dynamic workload-aware energy optimization (see Section III-C).

A. SoC Top-Level Architecture

The proposed SoC features a programmable architecture tailored for NeSy AI inference. As shown in Fig. 4, the chip integrates ten 576-KB RRAM-based neural computing tiles and two SRAM-based symbolic computing tiles. Microcontroller (μC) manages instruction dispatch, memory coordination, and power control across tiles. All neural tiles are connected to a shared SRAM memory system via the direct-transfer bus (DTB), and the chip supports VLIW instructions for flexible execution. The modular design enables heterogeneous compute operations to be performed in a tightly coupled manner.

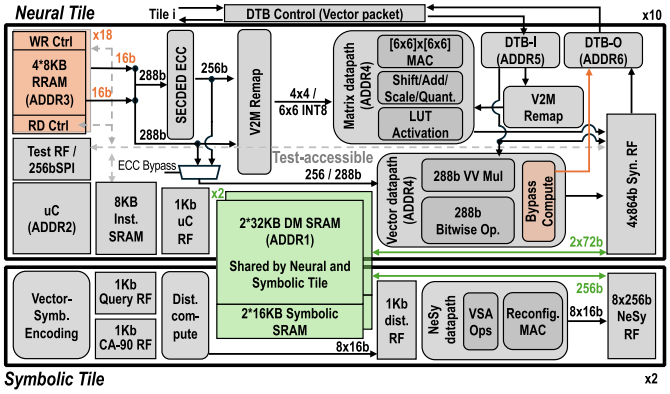


Fig. 5. Logical breakdown of the "per-tile" system and the VLIW address mapping space of the μC.

TABLE V

SAMPLING OF VLIW INSTRUCTION SUB-FIELD

Field	Name	Delays	Description
Control	EXIT	1	Halt; release mem. ctrl. to SPI
Control	SDPM-M	-	Macro power-down
Control	SDPM-B	-	Bank power-down
Control	ADDI	1	Immediate arithematics
Control	BEQ	1	Branch related
Receive	REC	2+4	Request through DTB then load
Send	SND	2	Store local data to DTB
NVM	R-LD	4	Load from RRAM into local regs
NVM	R-GLD	4	Load from other tile's RRAM
NeSy Op.	MM	2	Matrix-matrix multiplication
NeSy Op.	PERM	2	Element-wise shifting
NeSy Op.	VV	2	Vector-vector multiplication
Memory	LD	3	Load from mem. into local regs
Memory	SD	1	Store to mem. from local regs

The symbolic tiles are designed to support a diverse set of logic-based and vector-symbolic operations. These include bitwise logical functions, binding and bundling, circular convolution, and randomized vector generation via cellular automaton (CA)-90 units. Each symbolic tile integrates multiple compute datapaths and MAC engines, along with local SRAM buffers. The symbolic tiles are also configured to handle low-level control logic for multi-dataflow support and pipeline scheduling, and the μC dispatches instructions for symbolic execution based on operator graphs extracted from software profiling.

Fig. 5 shows the logical breakdown of the per-tile system. The neural tile readout datapath contains four channels, with the option of performing SECDED ECC and matrix or vector computation. Bypassing ECC allows 12.5% more embedded nonvolatile memory (eNVM) on-chip storage, suitable for noise-tolerant NeSy applications, such as [17], [21], and [28]. Special operations that require time-multiplexed processing are mapped to the symbolic tiles. Interactions between tiles

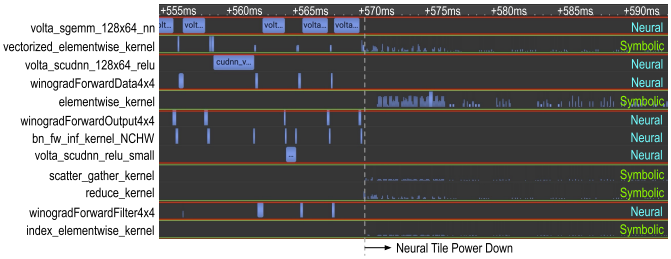


Fig. 6. Neural and symbolic operator execution analysis, demonstrating opportunities for per-tile PD design.

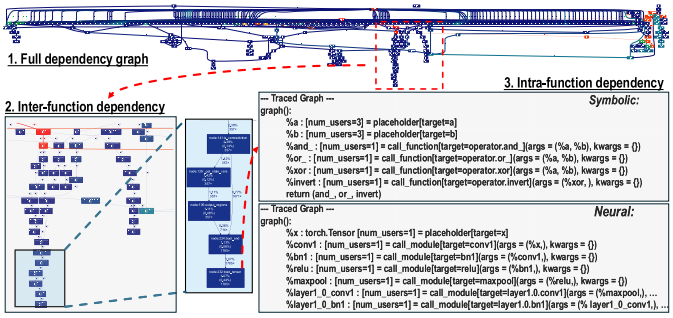


Fig. 7. Inter-/intra-function data dependency graph illustration of the SDPM scheduler.

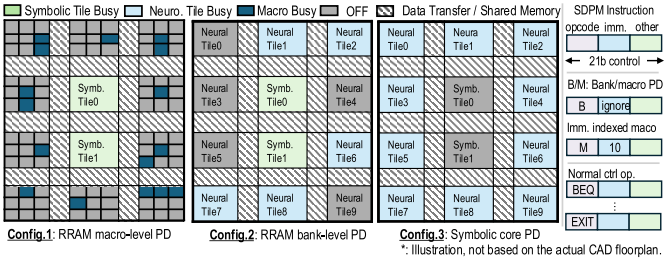


Fig. 8. Visualization of three SDPM configurations—RRAM macro-level PD, RRAM bank-level (neural tile) PD, and symbolic core PD. Power management is controlled by a 21-bit instruction.

utilize direct-mapped SRAM (DM SRAM) banks. Fig. 5 also shows the μC address space, possessing six sub-fields and forming a 128b instruction. Key instructions (such as loop folding, data transfer, and RRAM readout) are summarized in Table V along with the system delay cycle counts for the event considering DTB hand-shaking, pipeline bubble insertion, and RRAM multi-cycle-path (MCP) delays. Due to the MCP nature, RRAM readout requires two destination addresses to match the core computation bandwidth. The synthesized register file is designed to be one-to-any read and non-overlapping write. Shared SRAM banks and DTB are designed following the same rationale to prevent write conflicts.

B. Memory Design Choice

The design choice for RRAM and SRAM across the architecture is guided by two primary considerations: optimizing memory capacity and balancing datapath latency.

Neural kernels inherently demand significant memory capacity, making RRAM a suitable choice due to its superior density. Post-layout analysis shows that RRAM achieves 2.9× higher density at the cell level and 2.2× at the macro level (256 Kb memory, with low-voltage and high-voltage write peripherals) compared to SRAMs.

Symbolic kernels involve more complex datapaths and diverse compute blocks, making SRAM a suitable choice due to its low latency and synchronous access. The datapath latency of SRAMs is typically less than half that of RRAMs. The increased latency of RRAMs stems from two main factors: 1) the RRAM macro operates as an MCP relative to the core clock and 2) additional processing logic, such as error correction and data re-mapping, is required at the post-RRAM readout datapath. Given that symbolic building blocks incorporate features like vector batching, randomization, time-multiplexed bitwise logical operations, MAC units, and control logic for multi-dataflow support, the overall system design judiciously allocates SRAMs to symbolic tiles and RRAMs to neural tiles. This ensures optimal performance for both types of operations.

C. Software-Defined Power Management

Enabling SDPM is essential to efficiently accelerate a wide range of NeSy workloads. Operational dependency analyses are performed across representative workloads [21], [27], [28], [29], [30], [31], [32], [36], revealing the dynamic operator graphs on different NeSy paradigms. Fig. 6 shows an example of an execution trace from the NVSA model [21] based on `Nsight` profiling results and provides quantitative comparisons between the hardware power-down (PD) break-even requirement.

The SDPM scheduler is performed hierarchically, with `cProfile` and `torch.fx` for identifying inter-function and intra-function dependencies, respectively. The lack of native support for flat intermediate representations in torch.fx impedes the implementation of a non-hierarchical scheduler. This limitation necessitates direct modification of the source package, rendering the scheduler inapplicable across NeSy models. The SDPM hierarchical scheduler wraps torch.fx package with cProfile package, and the latter analyzes the inter-function dependency in the algorithm and is package (application) agnostic. Fig. 7 demonstrates the dependencies at different levels in the SDPM scheduler. The complete functional dependency graph is first shown with LNNs [29] as the proxy at the top of the figure. This dependency graph is generated with the cProfile and pstats packages. The housekeeping functions are filtered out, forming the inter-functional dependency graph. Within each function, the dependency analysis is then performed with torch.fx, which traces the lowest-level computational kernels. Fig. 7 visualizes the inter-/intra-functional dependency output.

Visualization of the SDPM is shown in Fig. 8. SDPM is achieved by the control field of the μC, where the opcode specifies the bank-level or module-level PD and the immediate field specifies the macro ID. Header/footer power switches are employed locally, including the analog buffer/comparator, resistive digita-to-analog conversion (RDAC) bias, analog VDD (AVDD) power, and high-voltage (HVDD) power to meet tight physical design pitch within the macro and allow

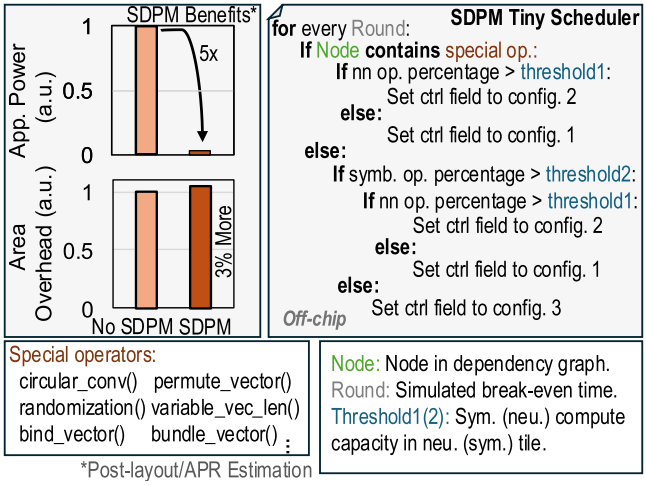


Fig. 9. SDPM based on the operators and the compute capacity in cores.

fast shut-down/wake-up response. The off-chip SDPM scheduler first identifies special operators in a given time frame; if existed, the symbolic core must stay awake (i.e., configuration three is invalid) for these operations, such as convolution, randomization, permutation, and so on. Otherwise, three configurations are possible depending on the computational capacity in neural and symbolic tiles. Fig. 9 illustrates the operational principle of the tiny scheduler. First, the duration of the scheduler time frame (i.e., Round) is selected to be at least the break-even time [37] based on the post-place-and-route (PR) simulation. In a given round, if there is any special operator (see Fig. 9) node based on the intra-functional analysis (see Fig. 7), the symbolic tiles will remain active and two configurations are available for the neural tiles (see Fig. 8), depending on the number of neural operations. Considering a round where a special operator does not exist, and the number of neural operations is minimal (threshold equals the symbolic tiles' computational capacity), the symbolic tiles will be responsible for full computations, whereas the RRAM would be active for only storage purposes. The example includes the NVSA workload, shown in Fig. 6.

IV. Symbolic Tile Implementation

This section presents the implementation details of the symbolic tile, with its overall architecture (see Section IV-A), core building blocks, and dataflows (see Section IV-B).

A. Overall Architecture of Symbolic Tile

NeSy workloads differ fundamentally from conventional neural networks in that execution time is often dominated by symbolic reasoning rather than dense matrix operations. As characterized in Section II, NeSy models spend substantial runtime on operators such as bitwise binding and bundling, circular convolution, permutation, distance computation, and first-order logic evaluation, kernels that are poorly matched to GEMM-centric CPUs and GPUs. To efficiently accelerate these components, the proposed SoC integrates dedicated symbolic tiles that are specialized yet tightly coupled with the neural computing fabric.

Symbolic kernels fall into two dominant computation classes. VSAs rely primarily on binary and bitwise operations [33] (e.g., XOR-based binding, majority-based bundling, permutation, and circular convolution [21]), typically over low-bitwidth vectors. In contrast, logic-based symbolic models require integer arithmetic for first-order logic inference, including identifier representation [31], weighted conjuctions ($\wedge$) and disjuctions ($\vee$) [29], and quantifiers ($\forall$ and $\exists$) [29], [30], [31]. A single homogeneous datapath is inefficient for both: bitwise kernels waste energy and area on integer MACs, while emulating integer logic with bitwise primitives incurs high latency and control overhead. Accordingly, each symbolic tile integrates both datapaths within a reconfigurable NeSy MAC fabric, enabling efficient execution across the symbolic kernels in Table III.

Symbolic reasoning is tightly interleaved with neural perception, with symbolic kernels frequently consuming intermediate neural outputs such as embeddings or activations. This interdependence creates dynamic execution graphs (see Fig. 2) with frequent neural-symbolic transitions. To minimize data movement and synchronization overhead, the symbolic tile is integrated with neural tiles through on-chip memories. Each tile includes local SRAM banks for symbolic storage and shares the DM SRAM buffers with neural tiles via the DTB, enabling symbolic kernels to consume neural outputs without off-chip accesses.

B. Building Blocks and Dataflows of Symbolic Tile

1) Adaptive Vector Length Support: The symbolic tile incorporates several configurable building blocks to support varying symbolic model demands. To accommodate a wide range of vector lengths used in symbolic AI, ranging from 128 to 4096 bits, the tile supports adaptive vector configurations via programmable registers. When implementing on hardware, a large amount of fan-out is induced on control signals that drive registers across these wide vectors, folding the long vectors not only make the system being able to support wide range of vector length, but also reduce the large fan-out datapath in digital circuit, which leads to lower power and area as the hold-time fixing buffer inserted by the PR tool is greatly reduced. We use a 12-bit configuration shift register [see Fig. 10(c)], with 2-bit distance quantization, 3-bit binding count, 3-bit active register count, and 4-bit fold count that represents 16 symbolic vector folds.

2) Random Vector Generator Module: Central to the symbolic processing pipeline is the CA-90 module, a cellular automata-based random vector generator [see Fig. 10(b)]. Compared with traditional linear-feedback shift registers (LFSRs), the CA-90 generator offers several advantages for symbolic random vector generation. Its fully local update rule where each bit is updated based on its immediate neighbors, eliminating the need for global XOR feedback logic, resulting in significantly reduced area and routing complexity. Post PR analysis shows that CA-90 reduces memory footprint by 52.8% and latency by 38.1% compared to LFSRs. Additionally, CA-90 can generate pseudorandom numbers and naturally work with the time-multiplexed bit-wise logic. It also eliminates the risk of lock-up states (e.g., all-zero states) and

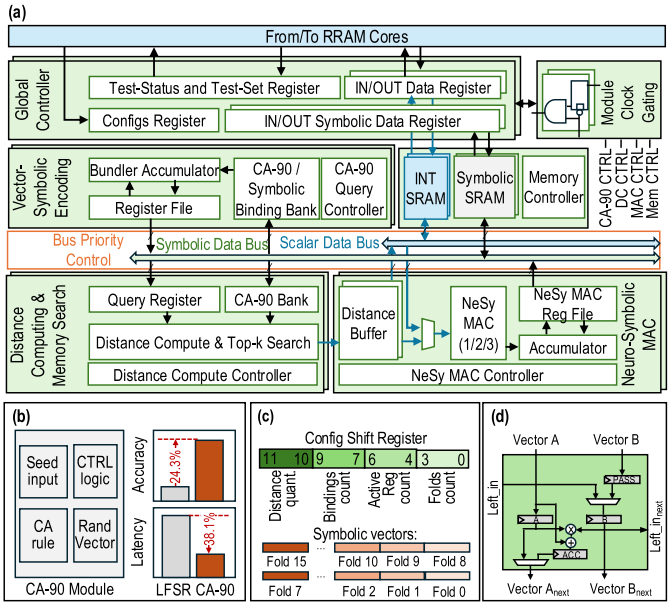


Fig. 10. Symbolic tile implementation. (a) Overall architecture of the symbolic tile. (b) Vector-symbolic randomization unit. (c) Adaptive vector length support for various NeSy applications via configuration shift register. (d) Adaptive NeSy MAC unit for supporting different NeSy dataflow.

yields higher statistical diversity in symbolic representations, leading to a 24.3% accuracy gain in reasoning tasks.

3) Distance Compute Module: This module implements distance evaluation and selection functions required by symbolic reasoning workloads. It is composed of three tightly integrated logic stages that collectively support similarity computation, distance accumulation, and nearest-vector selection. First, a bitwise comparison uses XOR to compare two folds and an adder tree to return the difference between the number of 1's and the number of 0's, thus computing the dot-product similarity. These partial distances are accumulated across multiple vector folds using dedicated distance registers, enabling flexible aggregation of intermediate results while supporting independent read and write operations. A configurable data-selection path allows accumulated distance values to be forwarded for subsequent scalar processing. Finally, a comparison and selection stage identifies the closest matching vector by evaluating the computed distance metrics, enabling efficient nearest-neighbor determination within the symbolic tile.

4) Symbolic Logic Module: This module implements key vector-symbolic architecture operations, used to construct distributed perceptual representations and perform symbolic reasoning computations. VSAs are widely employed across NeSy workloads and often dominate execution latency. For example, VSA-centric symbolic operations account for 93.7% and 80.5% of the total runtime in the LVRF [27] and PrAE [17] workloads, respectively. The module comprises five logic units: a binding unit (BIND), a multiplying unit (MULT), a bundling unit (BND), a register file (BND RF), and a sign unit (SGN). BIND connects to a local buffer storing vectors and is used to execute the binding operations over vectors. The superposition of bound vectors is implemented in BND through element-wise addition (bundling). BIND and

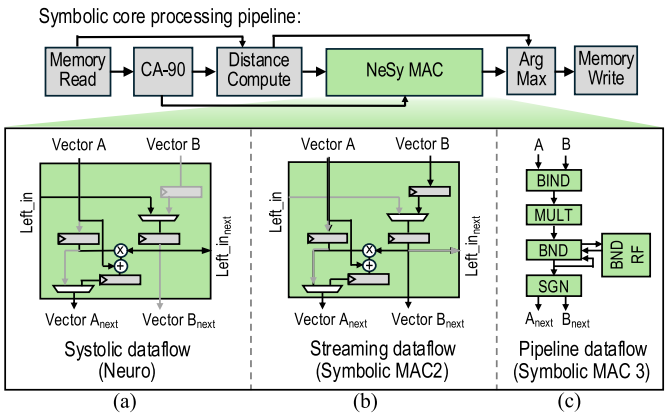


Fig. 11. Symbolic tile execution pipeline and dataflow support. (a) Configured as a systolic array for a neuro operator. (b) Configured for symbolic circular convolution. (c) Configured for bit-wise logic operations.

BND utilize different data representations, with BIND using binary and BND using integer formats. MULT manages the conversion from binary to integer formats and also performs element-wise scalar multiplication, an essential operation for NeSy encoding. Integer folds outputted from BND can be temporarily stored in BND RF for continuous superposition or converted to binary through SGN for transfer over the global vector-symbolic datapath.

5) NeSy MAC Module: This module provides reconfigurable support for a wide range of neural and symbolic operations (see Table III). Because the SDPM scheduler may offload workloads with low neural intensity to the symbolic processor, the symbolic processor is designed to support multiple execution dataflows, including systolic, streaming, and pipelined modes. These dataflows enable efficient execution of neural MAC operations, specialized symbolic kernels, and bitwise logic operations, respectively. Fig. 10(d) shows the microarchitecture of the reconfigurable unit. Each unit contains four registers: a stationary register (A), a passing register (PASS), a streaming register (B), and a partial-sum register (ACC). By selectively configuring data movement and accumulation across these registers, the unit can be dynamically reconfigured to realize different dataflows and efficiently support diverse neural and symbolic operations.

6) Reconfigurable Operation Modes: Fig. 11 illustrates the execution dataflows supported by the NeSy MAC. Before computation, input vectors are loaded into the NeSy MAC, where vector A (weights of GEMM) is passed into the stationary register. Reconfigurability is achieved by selecting the source of input vector B either from the "left_in" link, enabling GEMM-style execution, or from the passing register, enabling circular-convolution-style execution. In the systolic dataflow mode, the NeSy MAC operates analogously to a systolic array for GEMM processing, with input vectors streamed horizontally from left to right through the "left_in" links while vector A remains stationary. In the streaming dataflow mode, the NeSy MAC executes vector-symbolic circular convolution, a dominant symbolic operation in various NeSy workloads (e.g., accounting for >80% runtime of symbolic operations in NVSA [21]). In this mode, input vector B is streamed vertically from top to bottom via the passing register, enabling

TABLE VI
SAMPLING OF RRAM READOUT DESIGNS

Reference	Usage	Process	V_{MIN}	SA Type	SA Reference	Edge-triggered SA	Prolonged timing	SoC/Macro
ISSCC'18 [38]	Memory	40nm Planar	NR	CS	Current	NR	NR	Macro
ISSCC'19 [39]	Memory	22nm FinFET	0.5V	CS	Resistance	NR	NR	Macro
ISSCC'21 [40]	CIM	40nm Planar	0.8V	VS	NR	Yes	NR	Macro
ISSCC'22 [41]	CIM	40nm Planar	0.8V	CS	Current	Yes	NR	Macro
ISSCC'24 [42]	Memory	40nm Planar	0.8V	CS	Resistance	Yes	NR	SoC
ISSCC'24 [43]	Memory	12nm FinFET	NR	CS	Resistance	Yes	NR	Macro
This Work	Memory	40nm Planar	0.7V	CS	Res+Regulated CM	Yes	Yes	SoC

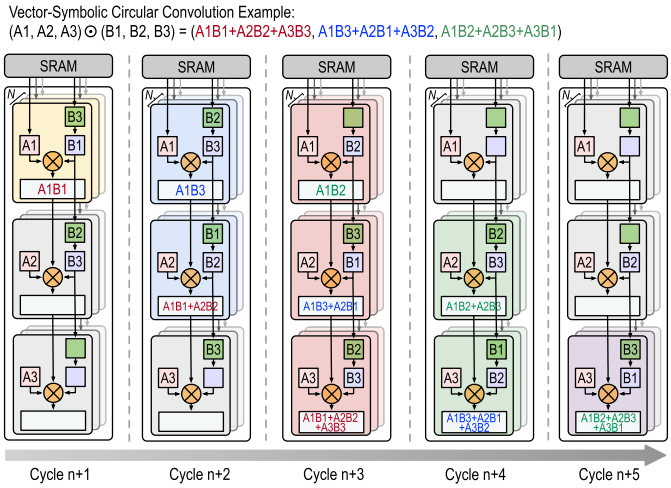


Fig. 12. Illustration of the cycle-by-cycle streaming dataflow used to perform vector-symbolic circular convolution.

temporal reuse of the streaming input across successive cycles. This vertical streaming pattern efficiently realizes circular convolution without redundant data movement. In both systolic and streaming dataflow modes, partial results are accumulated and reduced along the vertical dimension, allowing a unified accumulation across neural and symbolic executions.

7) Streaming Dataflow Cycle-Level Analysis: Fig. 12 presents a cycle-by-cycle view of the streaming dataflow supported by the NeSy MAC. To execute vector-symbolic circular convolution between two input vectors A and B on a 3 × 1 processing array, vector A is held in the stationary registers while vector B is streamed across processing elements through the passing registers. In each cycle, the streaming input is temporarily buffered in the passing register before being forwarded to the streaming register. In the subsequent cycle, this value is propagated to the passing register of the next processing element, enabling controlled temporal reuse of the input across the array. During each cycle, the MAC unit operates on data from the stationary and streaming registers and accumulates the result into the partial-sum register. This process repeats across cycles until all partial results are accumulated and the final outputs are produced [22].

V. RRAM MACRO IMPLEMENTATION

To support high on-chip array density in NeSy workloads as described in Section II, a high-density CNM RRAM macro is designed, along with custom sensing circuits that achieve low readout per-bit energy, and an integrated register-transfer level (RTL) wrapper performs state management for 5.625-MB eNVM write and read events. This section first describes the custom RRAM sensing readout circuit. Then, the RTL wrapper and the mixed-signal and hierarchical integration process are discussed.

A. RRAM Sensing

Table VI summarizes existing RRAM macro and SoC sensing designs. The majority of recent RRAM macro designs leverage current sensing (CS) due to the limited precharge voltage range across the RRAM cell. This is because the voltage across the RRAM is limited within 300 mV to prevent potential data drift in the memory, which is also known as the read disturbance. A resistive DAC biasing is chosen to avoid routing analog signals across the test chip while retaining test-time adjustability, along with a regulated-cascode current mirror (CM). Edge-triggered (with respect to the core clocking system) timing management is preferred for mixed-signal timing closure. A custom digital timing management circuit is designed for prolonged bit-line (BL) current establishment.

Fig. 13 shows the overall readout circuitry, including a custom digital timing management circuit, analog readout, and the RTL signals for state management. The timing management circuit provides an edge-triggered, tunable, and prolonged timing window for BL current establishment. Fig. 14(a) demonstrates that existing memory readout schemes can be categorized into (1) fixed budget across readout pipelines [42] or (2) nonedge-triggered with respect to the core clock [38]. Our previous generation test macro measurements [40], [41], [42], [44] indicate that the foundry-sourced RRAM readout current establishment dominates the readout timing budget. However, the existing solution (1) unnecessarily allocated timing to a fast-responding precharge and sense amplifier (SA), thereby compressing the available timing budget for memory current establishment. Furthermore, solution (2)'s reliance on delay gates (DGs) for memory readout complicates macro integration across different operating conditions. This is because the analog delay of DGs does not scale with frequency, and as such, it would be necessary to constrain the path with different multi-cycle exceptions depending on the frequency achievable at various corners, which is typically an output rather than an input of the design flow.

The proposed readout design aims to address the above limitations. Fig. 14(b) illustrates the functionality of the cus-

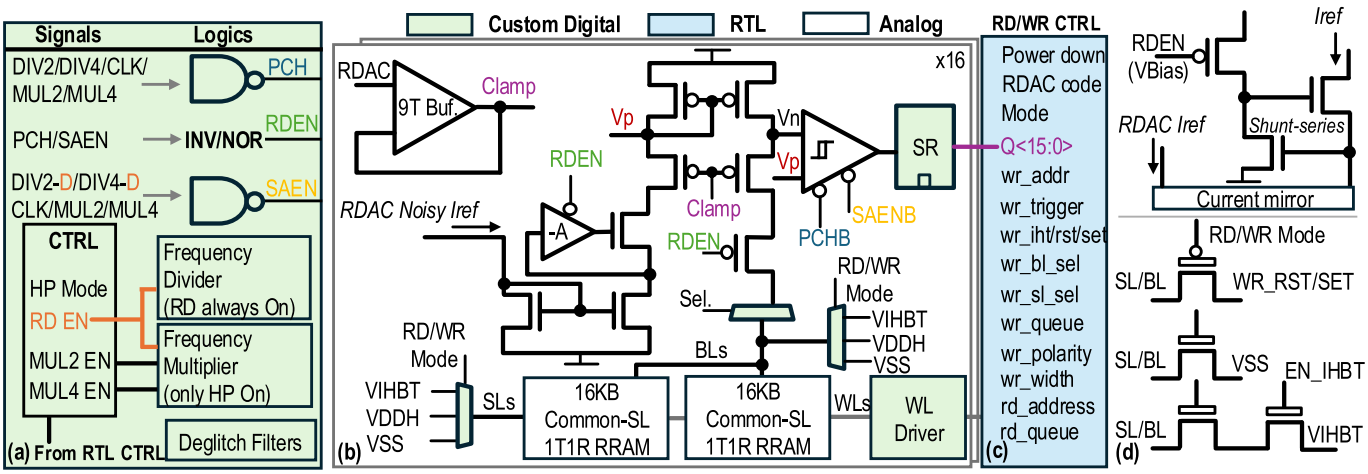


Fig. 13. Mixed-signal RRAM readout circuit. (a) Custom digital edge-triggered and prolonged timing management. (b) Analog readout. (c) RTL signals for read/write/PD state management. (d) Regulated-cascode CM and the thick-oxide muxes for set/reset/inhibition.

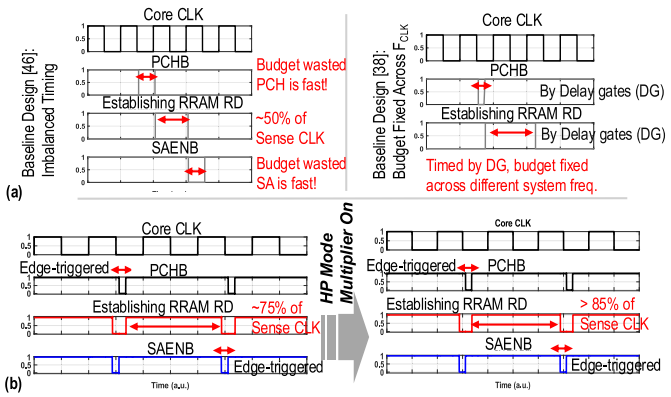


Fig. 14. (a) Existing timing management scheme for RRAM readout. (b) Functional illustration of the custom digital timing management circuit.

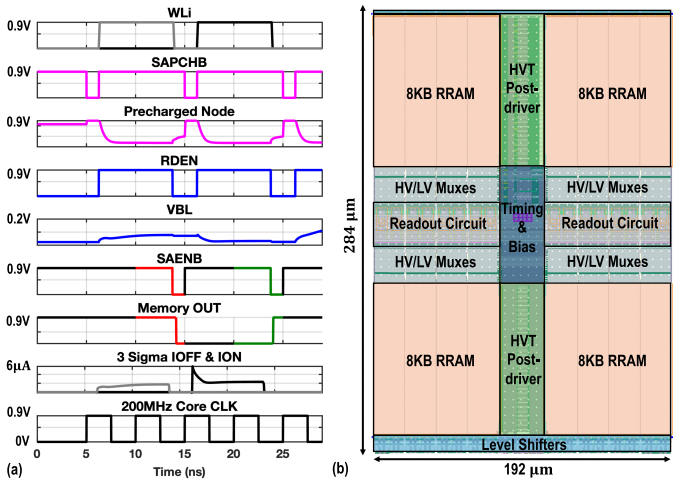


Fig. 15. (a) Post-layout extracted top-level waveforms at 200-MHz, 0.9-V supply, typical corner, 27 °C, 3σ RRAM LRS/HRS. (b) RRAM macro butterfly floorplan CAD snapshot.

tom digital timing management circuit. The custom digital circuits enable edge-triggered sequence while maximizing the timing budget for establishing the memory readout current. The baseline mode utilizes >75% of the timing budget for establishing the readout current. The high-performance (HP) mode allows >85% for aged RRAM whose resistive has drifted, potentially necessitating a longer-than-expected BL establishing time. The edge-triggered nature of the design provides better timing closures when integrating the RTL-based design described in Section V-B. Post-layout waveforms that illustrate the unbalanced timing requirements between precharge and SA are further shown in Fig. 15(a) and are explained in a later paragraph.

The analog read domain [see Fig. 13(b)] consists of a regulated-cascode CM, clamping transistors, enabling transistor, p-type CM, p-type SRAM-style SA, and an SR latch. Regulated-cascode incorporates a series-sampler with source-degenerated drain resistance [see Fig. 13(d)], providing a low-noise current source with low silicon cost. The clamping transistor clamps the bitline voltage to a low level, setting a soft limit on the voltage across the memory during readout, as described earlier in this section. Mirror-folded 9T buffer offers a rail-to-rail clamping voltage range for test-time measurement purposes. The SR latch holds the custom macro's output in a stable state during an SA reset, which ensures a reliable read from the RTL logic, reducing the effort required during integration.

The p-type precharger in the designed CS approach is to remove any hysteresis of the previous sensing operation so that each new sensing operation starts from the identical voltage level. The CS readout design is also functional without a precharger. With the precharger that precharges the Vp and Vn to supply voltage (VDD), each sensing operation (after the precharge step) will begin with the PMOS mirror pair guaranteed to be turned off. This leads to a consistent transient behavior during the development phase—when current is allowed to flow into the memory cells and reference arm. In the opposite case, where the precharger precharges Vp and Vn to a state with the PMOS both on, there is a risk that the design happens to be sensing a very-low-resistance memory cell; then, transiently, a significant peak current before the PMOS gate voltage settles can potentially occur since the PMOS in the non-reference-side starts fully on. The SA is a standard "strong-arm" structure.

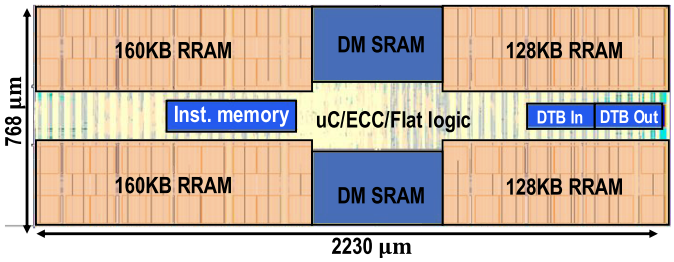


Fig. 16. Bank-level RTL wrapper CAD snapshot.

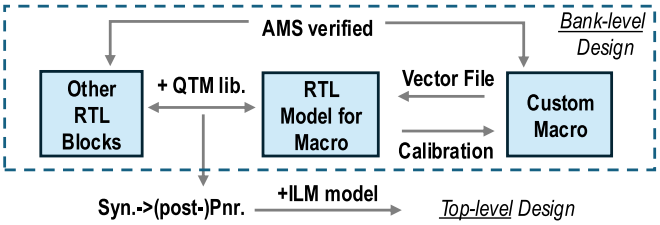


Fig. 17. Design verification flow of the designed test chip.

Fig. 15(a) demonstrates post-layout waveforms at top-level. The simulation was performed under 3σ cell resistance variation based on previous prototype measurements [40], [41], [42], [44]. As described, the VBL is clamped to small values to prevent read disturbance. Less than 50-mV VBL drift is achieved for a robust memory readout to detect 1-μA current difference shown in Fig. 15(a). As illustrated in Fig. 14 and its corresponding text, the transient response of the precharge and sensing event occurs within 200 ps, thus motivating the prolonged custom timing management circuit.

Fig. 15(b) demonstrates the butterfly floorplan of the custom macro. The middle-circuit area contains a voltage DAC and buffer for locally generating the clamping voltage, which is needed for the CS preamplifier in Fig. 13(b). The custom timing management circuit lies in the middle-circuit area as well for balanced timing. The butterfly floorplan also indicates the word-line (WL) drivers are shared across two 8-KB RRAM macros. Column architecture of the WL driver adopted the merged HV/low-voltage (LV) WL signaling scheme [42], enabling one custom level-shifter per eight tight WL driver layout pitch. HVT devices are used in the custom WL driver, as the post-layout simulation indicates the standard VT (SVT) counterpart took one-third of the leakage. This does not affect system timing closure, as the post-decoder is not at the critical readout timing path. RRAM write circuits required >3.3-V write voltages to set and reset the memory. Initial filament formation in unformed RRAM necessitates overdriving the write and I/O circuits to approximately 4 V. These circuits, isolation muxes, and IOs are designed with thick oxide transistors with overdrive CAD layers and sit mainly in the middle-left and middle-right areas, with IO-related designs at the top and bottom PR boundary.

B. Practical Considerations: Design of the RTL Wrapper

1) Bank-Level RTL Wrapper: The mixed-signal macros are tightly integrated with the RTL wrapper to expedite design cycles, as each macro has more than 200 pins, making bank-level design in custom impractical. The bank-level physical design with RTL wrapper that wraps the custom macro described in Section V is shown in Fig. 16. Most logical operations associated with programming the RRAM are implemented in RTL along with state machines, which include pre-decoder, PD/read/write mode selection, address pre-decoding, SL/BL voltage selection for setting/resetting/inhibiting the RRAM cells, queuing the read/write requests, adjusting the write pulse for robust write operation, and so on. These are synthesized logics in the "flat logic" area in Fig. 16, along with the μC and the ECC logic.

The addressing scheme from the RTL pre-decoder into a single macro is separated into 5 bits for the BL multiplexer and 9 bits for $256 + 256$ WLs. To characterize the write behavior correctly over serial peripheral interface (SPI), write and read queuing are implemented. To prevent multiple writes from occurring during measurements, the write request is edge-triggered. The macro-level RTL wrapper also implements a "sandwich" write scheme for RRAM, where the BL and SL of the macro are first activated, followed by WL activation. The write pulse then runs, after which the WL is deactivated, and finally, the BL/SL are deactivated. This is because the load condition is more consistent for WL, while that of BL/SL is highly dependent on the state of memory cells.

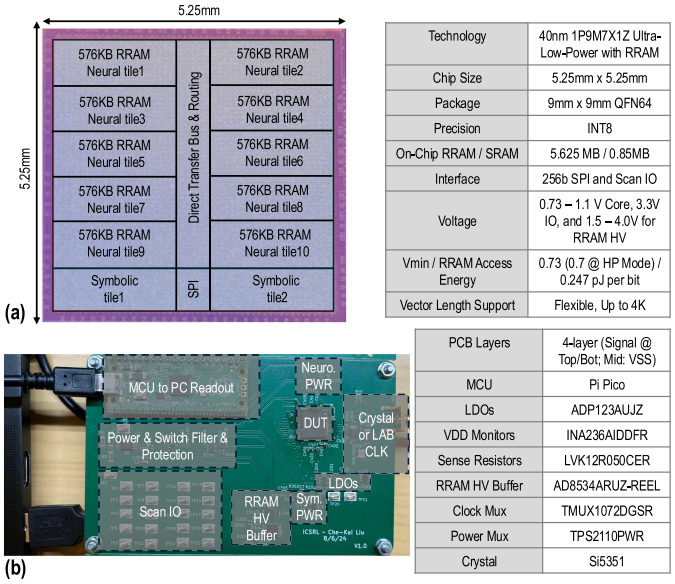
2) System-Level Integration and Verification: The verification process during the integration phase of the custom macros and RTL wrapper is illustrated in Fig. 17. First, boundary of custom macros' IOs have been digitized. A cycle-accurate RTL model for the custom RRAM macro is developed after the custom macro has been fully verified in the analog design environment. The testbench for the cycle-accurate RTL model includes more than seventy tasks covering faults such as PD early/late arrival, invalid address, configuration changed during read, read with write voltage is activated, and so on. The cycle-accurate model is built iteratively with the RTL simulated.vcd vector file. The RTL design in Fig. 13(c) is verified in the analog mixed-signal (AMS) simulation environment with the verified custom RRAM macro. The custom RRAM macro is treated as a hard block in.lef generation, and more-than-200 pins are placed around the PR boundary. Integration is then being performed with automated synthesis and PR flow. The timing information for the custom RRAM macro is generated with the Synopsys quick timing model (QTM) based on the characterization from analog testbenches. Hierarchical design flow is performed for the test chip at the top-level integration with the Synopsys interface logic model (ILM), at the cost of $\approx$10% increase in total routing metal length in exchange for 20$\times$ CAD tool runtime speedup at top-level integration.

VI. Measurements

The test chip is fabricated with TSMC's 40-nm ultralow-power (ULP) CMOS process with back-end-of-line (BEOL) embedded RRAM. Fig. 18 summarizes the general information of the test chip and the printed circuit board (PCB) design.

A. Test-Time Design and Setup

An automated testing framework is crucial for programming a large number of on-chip RRAMs and SRAMs. In addition, a single-shot write pulse often is not sufficient to set or reset the filament within the RRAM. An additional API is highly



Technology	40nm 1P9M7X1Z Ultra-Low-Power with RRAM
Chip Size	5.25mm x 5.25mm
Package	9mm x 9mm QFN64
Precision	INT8
On-Chip RRAM / SRAM	5.625 MB / 0.85MB
Interface	256b SPI and Scan IO
Voltage	0.73 – 1.1 V Core, 3.3V IO, and 1.5 – 4.0V for RRAM HV
Vmin / RRAM Access Energy	0.73 (0.7 @ HP Mode) / 0.247 pJ per bit
Vector Length Support	Flexible, Up to 4K
PCB Layers	4-layer (Signal @ Top/Bot; Mid: VSS)
MCU	Pi Pico
LDOs	ADP123AUJZ
VDD Monitors	INA236AIDDFR
Sense Resistors	LVK12R050CER
RRAM HV Buffer	AD8534ARUZ-REEL
Clock Mux	TMUX1072DGSR
Power Mux	TPS2110PWR
Crystal	Si5351

Fig. 18. (a) Test chip snapshot and general information. (b) Evaluation board design and components.

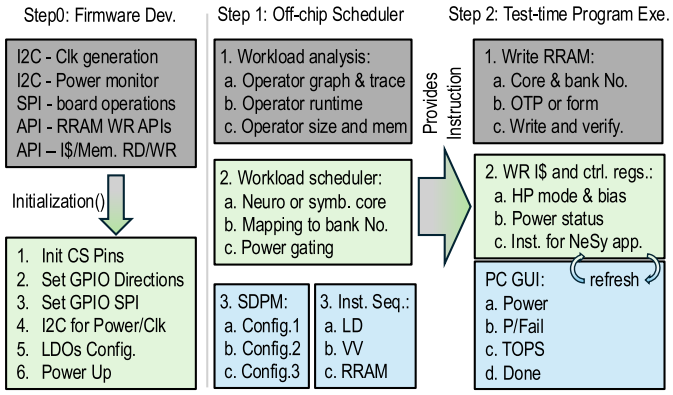


Fig. 19. Test-time firmware, scheduler, and execution flow.

desirable to support an automated iterative read–verify–write process.

Fig. 19 first summarizes the firmware development with C++, including I2C for power monitoring, APIs for RRAM write operations, APIs for instruction memory and buffer read/write support, and SPI for remaining board operations. The power monitor INA236 reads the LVK12 sensing resistor digitally and send back to the PC for human GUI readout. The automated GUI readout can be bypassed when measurement validation is needed with a lab supply. The testing flow is controlled by a Raspberry Pi Pico integrated with the above custom APIs. The Python-based off-chip scheduler is incorporated within the instruction generator, providing workload execution dependency analyses and full instruction sequences, as described in Section III-C. The custom Python-based instruction generator was part of the SoC verification process illustrated in Section V-B2 before tape-out, serving as the golden model for its RTL counterparts. The overhead of compiling the instructions lies mainly in generating the full dependency graph (see Fig. 7), while the instruction

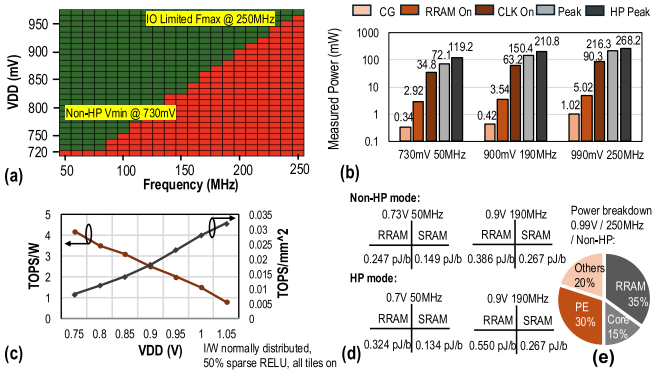


Fig. 20. (a) End-to-end shmoo. (b) Neural tile power consumption across different operating modes. (c) Neural tile performance. (d) RRAM readout power. (e) Neural tile power breakdown.

generator backend takes marginal compilation latency. This is because, unlike the conventional compiler, the used trace-based approach requires real execution of an application.

B. Memory Cell and Power and Efficiency

The component-level marginal power breakdown is achieved with the fine-grained control provided by μC. The baseline power from μC can be established by running an instruction sequence that does not perform any RRAM read. Then, an instruction sequence with an activated RRAM access field is measured. This eliminates the need for multiple PENDCAP and PRCUT cells at the IO with numerous separate power domains, as well as their core power rings. Consequently, only five power rings exist at the chip level power ring, including neural supply, symbolic supply, two RRAM analog supplies, and global ground.

The test chip's end-to-end (neural and symbolic tiles both activated) shmoo in Fig. 20(a) shows the test chip performance, demonstrating a 0.73-V lowest achievable VDD. In contrast to prior works where the one-time-programming (OTP) mode significantly expands the operating margin [45], [46], the shmoo [see Fig. 20(a)] indicates the OTP mode provides marginal improvement over the non-OTP mode. This suggests that with the prolonged RRAM readout, the dominant failure mechanism at low voltage is not the RRAM ON/OFF current ratio (primary metric OTP improves), but rather the timing constraints associated with the precharge and readout circuitry. The existing RRAM macro and SoC at the identical technology node [41], [42] reports a 0.8-V counterpart without a prolonged RRAM sensing mechanism. The per-bit energy consumed at this minimum VDD is 0.247 pJ/b, reported in Fig. 20(d). The neural tile power across different operating conditions is shown in Fig. 20(b). The peak power occurs when the neural tiles continuously perform MM instruction every clock cycle from RRAM and DM SRAM while the DTB-in/out instruction is busy. With RRAM PD and all synthesized logic being clock-gated, power reduces to 0.34 mW. 70-μW leakage when the system is being powered-down with SRAM banks in retention (measured, not shown). The distribution of peak power shows that the RTL wrapper (PE and core) consumes only 1.28$\times$ compared to that of the RRAM macros.

TABLE VII

COMPARISON WITH STATE-OF-THE-ART MRAM, RRAM, AND HD ACCELERATORS

	Reference	VLSI'22 [47]	JSSC'24 [48]	JSSC'22 [49]	VLSI'20 [50]	VLSI'21 [51]	ISSCC'22 [52]	ISSCC'23 [53]	JSSC'24 [54]	ISSCC'24 [46]	VLSI'24 [55]	ESSCIRC'24 [56]	VLSI'24 [57]	This Work
	Category	MRAM-based Neural Accelerator			RRAM-based Neural Accelerator							Hyperdimensional (HD) Computing Accelerator		Neuro - Symbolic
Test Chip Specifications	Technology eNVM NVM Usage	N22 MRAM Memory	N16 MRAM Memory	22 FDSOI MRAM Memory	N22 ULL RRAM Memory	N40 ULP RRAM Memory	N40 ULP RRAM CIM	N40 ULP RRAM CIM	N22 ULL RRAM CIM	N40 ULP RRAM Memory	N40 ULP RRAM Memory	N28 NR NR	N65 NR NR	N40 ULP RRAM Memory
	Target Application	Image Processing	Extended Reality	IoT	Inference	Inference & Training	Recommend. System	Target Tracking	Inference	Micro Survillence	Transformer	HD Classification	HD Encode	Neuro - Symbolic
	Area (mm^2)	8.76	16.00	12.00	10.80	29.20	25.00	20.25	24.48	20.25	65.60	2.71	1.70	27.56
	Supply Voltage (V)	0.5 – 1.0	0.65 – 0.8	0.5 – 0.8	0.6 – 1.1	1.1	0.9	0.8	0.7 – 0.8	0.8 – 1.1	0.9	0.8 – 1.0	0.7	0.73 – 1.1
	POWER (mW)	0.468 – 158	151 – 332	1.2e-3 – 49.4	128	126	20.0 – 131.3	4.6 – 21.3	101#– 135#	0.11^ – 603	11#– 22#	0.216#	NR	0.073^ – 321
	Frequency (MHz)	5.6e-2 – 190	360	3.2e-2 – 450¶	120	200	192	100	50 – 200	80 – 210	NR	200	20	50 – 250
	8b Energy Eff. (TOPS/W)	12.1§	2.68	1.3	0.96	2.2	1.8	1.2	51.4 – 76.5	1.14	0.43 – 0.50	NR	NR	0.8 – 10.8
	SRAM (MB)	1.39	6.25	1.68	1.3	0.5	0.75	1.25	0.5	0.74	2	0.25^	NR	0.85
	Processor	RISC-V	RISC-V	RISC-V	VLIW	RISC-V	Cortex-M3	Cortex-M3	NR	VLIW	RISC-V	NR	NR	VLIW
NVM Spec.	RRAM Vmin	Not Available			1.0	1.1	0.9	0.8	0.7	0.8	1.1			0.7&
	ECC/pre-ECC BER	NR/NR	NR/NR	Yes/NR	NR/NR	NR/NR	Yes/1e-2‡	Yes/1e-8	NR/NR	Yes/0†	NR/NR			Yes/0†
	NVM (MB)	2	4	4	3	2	2.25	1.25	4	5	12	Not Available		5.625
	NVM (pJ/b)	NR	NR	NR	3.75	1.00	CIM	CIM	CIM	0.256	NR			0.247
Neuro Func.	GEMM	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes			Yes
	Activations	Yes	Yes	Yes	NR	Yes	Yes	Yes	Yes	Yes	Yes	Not Available		Yes
Symb. Func.	NeSy Co-processing													Yes
	Non-HD Symb. Op.													Yes
	Vector Len. Support	Not Available		512 – 2048	Not Available							2048	1024	128 – 4096
	Symbolic Op.			Yes								Yes	Yes	Yes
	Rand. Kernel/Form			Yes/Digital								NR	Yes/Analog	Yes/Digital

NR: Not Reported / ^: Retentive sleep reported / &: HP Mode / #: Computed / !: Memory type - ROM / †: With test-time calibration. / ‡: Worst case; inferred / ¶: MRAM up to 40MHz / §: 80% W and 50% I sparsity

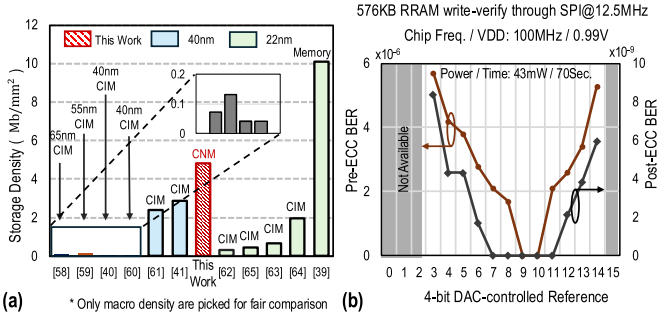


Fig. 21. (a) Comparison of RRAM macro density. (b) Pre-ECC and post-ECC readout of the RRAM with API-supported automatic write-verify.

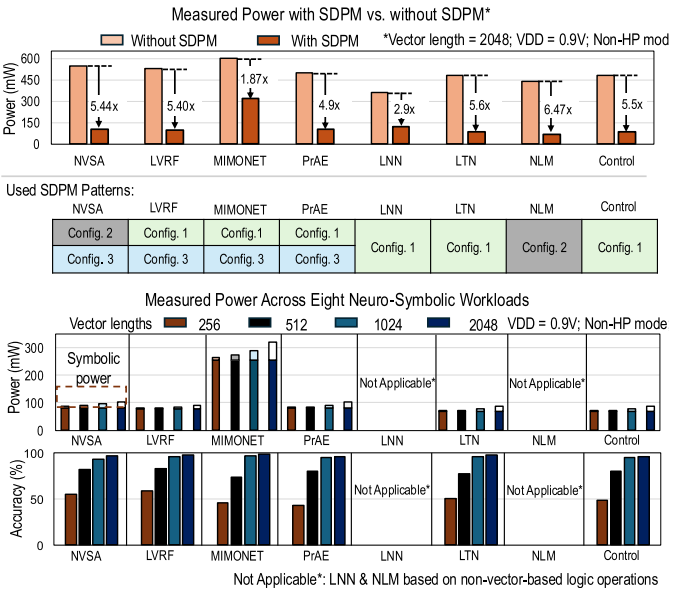


Fig. 22. Measurements across eight representative NeSy workloads, illustrating SDPM benefits and the support for variable vector lengths.

The neural tiles' compute efficiency and density are shown in Fig. 20(c). The numbers are obtained with the data stored normally distributed and all the synthesized computational logic activated, while performing MM every cycle with the matrix datapath.

Fig. 21(a) demonstrates the tight integration of the presented custom CNM RRAM macro, which improves storage density compared to existing [39], [40], [41], [43], [58], [59], [60], [61], [62], [63], [64], [65] RRAM-based CIM macro by 1.67× at the same technology node. Fig. 21(b) shows the pre-ECC and post-ECC error rate of 576-KB RRAM readout. The write events are supported with the custom C++ API, allowing the full-chip's 5.625-MB RRAM write-verify event to be completed around 70 s. In addition, the full-chip's 5.625-MB RRAM test-time write-verify process reports a 43-mW power consumption through the INA236 VDD monitor. A "V-shaped" curve is observed because the low (high) RDAC code causes errors for the RRAM that is in LRS (HRS). Zero error can be obtained at pre-ECC by calibrating the 4-bit RDAC [see Fig. 13(b)] code through the custom C-based Pico firmware during test-time, and SECDED further reduces the BER by $\approx 10^3$ without RDAC calibrations. The flat region, that is, zero pre-ECC error, of Fig. 21(b) is eventually limited, where the "V-shaped" moves left (right) when increasing (decreasing) the supply voltage.

Fig. 22 illustrates application-level results. NVSA, LVRF, MIMONet, and PrAE utilize vector-based operators (e.g., circular convolution, binding, and bundling), thus operating across both power configuration 1/2 and configuration 3. LNN, LTN, and NLM incorporate logical operations and leverage the NeSy compute units within symbolic tiles to execute both neural and symbolic kernels. Owing to the dynamic dataflow across NeSy workloads, on average, 6.47× power reduction is achieved with the SDPM scheduler. With the folding mechanism at the SP, the power contribution of SP is less than 28% under 2048 vector dimensions and scales approximately linearly with the vector dimensions. Application benchmarks illustrate the importance of supporting variable vector length, as the reasoning accuracy of different applications can saturate at variable vector dimensionality. The test chip delivers a 10.8 TOPS/W peak efficiency across the above eight NeSy workloads.

VII. Comparison

Table VII shows the comparison table with existing MRAM/RRAM neural accelerators [42], [47], [48], [49], [50], [51], [52], [53], [54], [55] and hyperdimensional (HD) computing accelerators [56], [57]. HD computing is considered a subset of the NeSy algorithm that utilizes a MAC-based encoder and bitwise logical operations for inference. Existing works with accelerated symbolic kernels [49], [56], [57] fall into HD operators [33], including binding, bundling, permutation, and pseudo-randomization. This work incorporates a broader set of symbolic operators, including circular convolution, logical operations, and a true randomization kernel. Regarding system architecture, Rossi et al. [49] proposed an MRAM-based SoC where the HD unit serves as a wake-up controller. However, the sequential design nature of the "HD-then-neural" processing limits the potential for NeSy algorithm acceleration. In terms of efficiency, Zhang et al. [47] demonstrated an MRAM-based accelerator achieving 12.1 TOPS/W 8b energy efficiency. This high efficiency can be partly attributed to the scaled technology node with lower operating voltage and high weight sparsity, which enables a wider optimal energy design space. Similarly, the 22-nm RRAM-based accelerator by Hsu et al. [54] achieves up to 76.5 TOPS/W 8b energy efficiency, driven primarily by the analog CIM and multi-bit RRAM approach, trading off noise immunity for efficiency compared to digital baselines.

Despite the technology node difference, the described test chip possesses the lowest per-bit NVM readout energy compared to any of the referenced works. Furthermore, the test chip achieves the lowest minimum supply voltage (V_{min}) compared to works in the identical 40-nm RRAM technology node [46], [51], [52], [53], [55], and maintains a comparable V_{min} to designs in newer technology nodes [50], [54].

VIII. Conclusion

This article presents a fully programmable heterogeneous SoC fabricated in 40-nm ULP CMOS with BEOL embedded RRAM, specifically designed to accelerate a wide range of NeSy workloads. The featured integrated RRAM/SRAM datapath, SDPM, and time-multiplexed vector folding scheme synergistically accelerate NeSy models that span a variety of operator graphs.

Acknowledgment

The authors thank TSMC for technical discussions and chip fabrication support and Dr. Shota Konno and Prof. Sigang Ryu for helpful discussions.

References

[1] Z. Wan et al., "Towards cognitive AI systems: Workload and characterization of neuro-symbolic AI," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, May 2024, pp. 268–279.

[2] T. Mu et al., "Rule based rewards for language model safety," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 37, 2024, pp. 108877–108901.

[3] L. S. Lorello, M. Lippi, and S. Melacci, "A neuro-symbolic framework for sequence classification with relational and temporal knowledge," 2025, *arXiv:2505.05106*.

[4] J. Kwon, J. Tenenbaum, and S. Levine, "Neuro-symbolic models of human moral judgment," in *Proc. Annu. Meeting Cognit. Sci. Soc.*, vol. 46, 2024.

[5] A. Kalyanpur, K. K. Saravanakumar, V. Barres, J. Chu-Carroll, D. Melville, and D. Ferrucci, "LLM-ARC: Enhancing LLMs with an automated reasoning critic," 2024, *arXiv:2406.17663*.

[6] H. Xiong et al., "Converging paradigms: The synergy of symbolic and connectionist AI in LLM-empowered autonomous agents," 2024, *arXiv:2407.08516*.

[7] M. Ibrahim et al., "Special session: Neuro-symbolic architecture meets large language models: A memory-centric perspective," in *Proc. Int. Conf. Hardw./Softw. Codesign Syst. Synth. (CODES+ISSS)*, Sep. 2024, pp. 11–20.

[8] Z. Wan et al., "Towards efficient neuro-symbolic AI: From workload characterization to hardware architecture," *IEEE Trans. Circuits Syst. Artif. Intell.*, vol. 1, no. 1, pp. 53–68, 2024.

[9] J. Mao, J. B. Tenenbaum, and J. Wu, "Neuro-symbolic concepts," 2025, *arXiv:2505.06191*.

[10] H. Yang et al., "NSFlow: An end-to-end FPGA framework with scalable dataflow architecture for neuro-symbolic AI," 2025, *arXiv:2504.19323*.

[11] S. Du et al., "Cross-layer design of vector-symbolic computing: Bridging cognition and brain-inspired hardware acceleration," 2025, *arXiv:2508.14245*.

[12] A. d. Garcez and L. C. Lamb, "Neurosymbolic AI: The 3rd wave," *Artif. Intell. Rev.*, vol. 56, no. 11, pp. 12387–12406, 2023.

[13] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu, "The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019.

[14] C. Han, J. Mao, C. Gan, J. B. Tenenbaum, and J. Wu, "Visual concept-metaconcept learning," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 32, 2020, pp. 5001–5012.

[15] L. Mei, J. Mao, Z. Wang, C. Gan, and J. B. Tenenbaum, "FALCON: Fast visual concept learning by integrating images, linguistic descriptions, and conceptual relations," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2022.

[16] K. Yi et al., "CLEVRER: CoLlision events for video REpresentation and reasoning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019.

[17] C. Zhang, B. Jia, S.-C. Zhu, and Y. Zhu, "Abstract spatial–temporal reasoning via probabilistic abduction and execution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 9731–9741.

[18] Z. Wan et al., "Towards cognitive AI systems: A survey and prospective on neuro-symbolic AI," 2024, *arXiv:2401.01040*.

[19] V. Shah, A. Sharma, G. Shroff, L. Vig, T. Dash, and A. Srinivasan, "Knowledge-based analogical reasoning in neuro-symbolic latent spaces," 2022, *arXiv:2209.08750*.

[20] J. Hsu, J. Mao, and J. Wu, "NS3D: Neuro-symbolic grounding of 3D objects and relations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 2614–2623.

[21] M. Hersche, M. Zeqiri, L. Benini, A. Sebastian, and A. Rahimi, "A neuro-vector-symbolic architecture for solving Raven's progressive matrices," *Nature Mach. Intell.*, vol. 5, no. 4, pp. 363–375, Mar. 2023.

[22] Z. Wan et al., "CogSys: Efficient and scalable neurosymbolic cognition system via algorithm-hardware co-design," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Mar. 2025, pp. 775–789.

[23] A. Sheth and K. Roy, "Neurosymbolic value-inspired artificial intelligence (why, what, and how)," *IEEE Intell. Syst.*, vol. 39, no. 1, pp. 5–11, Jan. 2024.

[24] C. Núñez-Molina, P. Mesejo, and J. Fernández-Olivares, "A review of symbolic, subsymbolic and hybrid methods for sequential decision making," *ACM Comput. Surv.*, vol. 56, no. 11, pp. 1–36, Nov. 2024.

[25] T. H. Trinh, Y. Wu, Q. V. Le, H. He, and T. Luong, "Solving olympiad geometry without human demonstrations," *Nature*, vol. 625, no. 7995, pp. 476–482, Jan. 2024.

[26] B. Romera-Paredes et al., "Mathematical discoveries from program search with large language models," *Nature*, vol. 625, no. 7995, pp. 468–475, Jan. 2024.

[27] M. Hersche, F. di Stefano, T. Hofmann, A. Sebastian, and A. Rahimi, "Probabilistic abduction for visual abstract reasoning via learning rules in vector-symbolic architectures," 2024, *arXiv:2401.16024*.

[28] N. Menet, M. Hersche, G. Karunaratne, L. Benini, A. Sebastian, and A. Rahimi, "MIMONets: Multiple-input-multiple-output neural networks exploiting computation in superposition," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 39553–39565.

[29] R. Riegel et al., "Logical neural networks," 2020, *arXiv:2006.13155*.

[30] S. Badreddine, A. d'Avila Garcez, L. Serafini, and M. Spranger, "Logic tensor networks," *Artif. Intell.*, vol. 303, Feb. 2022, Art. no. 103649.

[31] H. Dong, J. Mao, T. Lin, C. Wang, L. Li, and D. Zhou, "Neural logic machines," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–22.

[32] A. Menon et al., "Shared control of assistive robots through user-intent prediction and hyperdimensional recall of reactive behavior," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 12638–12644.

[33] L. Ge and K. K. Parhi, "Classification using hyperdimensional computing: A review," *IEEE Circuits Syst. Mag.*, vol. 20, no. 2, pp. 30–47, 2nd Quart., 2020.

[34] NVIDIA,. *Nsight Compute*. Accessed: Aug. 31, 2024. [Online]. Available: https://developer.nvidia.com/nsight-compute

[35] NVIDIA,. *Nsight Systems*. Accessed: Aug. 31, 2024. [Online]. Available: https://developer.nvidia.com/nsight-systems

[36] T. Wu et al., "ZeroC: A neuro-symbolic model for zero-shot concept recognition and acquisition at inference time," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 9828–9840.

[37] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural techniques for power gating of execution units," in *Proc. Int. Symp. Low power Electron. Design*, Aug. 2004, pp. 32–37.

[38] C.-C. Chou et al., "An N40 256K×44 embedded RRAM macro with SL-precharge SA and low-voltage current limiter to improve read and write performance," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 478–480.

[39] P. Jain et al., "13.2 a 3.6Mb 10.1Mb/mm^2 embedded non-volatile ReRAM macro in 22 nm FinFET technology with adaptive forming/set/reset schemes yielding down to 0.5 V with sensing time of 5ns at 0.7 V," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2019, pp. 212–214.

[40] J.-H. Yoon, M. Chang, W.-S. Khwa, Y.-D. Chih, M.-F. Chang, and A. Raychowdhury, "29.1 a 40 nm 64Kb 56.67TOPS/W read-disturb-tolerant compute-in-memory/digital RRAM macro with active-feedback-based read and in-situ write verification," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 64, Feb. 2021, pp. 404–406.

[41] S. D. Spetalnick et al., "A 40 nm 64kb 26.56TOPS/W 2.37Mb/mm^2 RRAM binary/compute-in-memory macro with 4.23x improvement in density and >75% use of sensing dynamic range," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 65, Feb. 2022, pp. 1–3.

[42] S. D. Spetalnick et al., "An edge accelerator with 5 MB of 0.256-pJ/bit embedded RRAM and a localization solver for bristle robot surveillance," *IEEE J. Solid-State Circuits*, vol. 60, no. 1, pp. 35–48, Jan. 2025.

[43] Y.-C. Huang et al., "15.7 A 32Mb RRAM in a 12 nm FinFet technology with a 0.0249μm^2 bit-cell, a 3.2GB/S read throughput, a 10KCycle write endurance and a 10-year retention at 105°C," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2024, pp. 288–290.

[44] B. Crafton et al., "CIM-SECDED: A 40 nm 64Kb compute in-memory RRAM macro with ECC enabling reliable operation," in *Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC)*, Nov. 2021, pp. 1–3.

[45] P.-H. Lee et al., "33.1 a 16 nm 32Mb embedded STT-MRAM with a 6ns read-access time, a 1M-cycle write endurance, 20-year retention at 150°C and MTJ-OTP solutions for magnetic immunity," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2023, pp. 494–496.

[46] S. D. Spetalnick et al., "30.1 A 40 nm VLIW edge accelerator with 5MB of 0.256pJ/b RRAM and a localization solver for bristle robot surveillance," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech.Papers*, Feb. 2024, pp. 482–484.

[47] Q. Zhang et al., "A 22 nm 3.5TOPS/W flexible micro-robotic vision SoC with 2MB eMRAM for fully-on-chip intelligence," in *Proc. IEEE Symp. VLSI Technol. Circuits (VLSI Technol. Circuits)*, Jun. 2022, pp. 72–73.

[48] A. S. Prasad et al., "Siracusa: A 16 nm heterogenous RISC-V SoC for extended reality with at-MRAM neural engine," *IEEE J. Solid-State Circuits*, vol. 59, no. 7, pp. 2055–2069, Jul. 2024.

[49] D. Rossi et al., "Vega: A ten-core SoC for IoT endnodes with DNN acceleration and cognitive wake-up from MRAM-based state-retentive sleep mode," *IEEE J. Solid-State Circuits*, vol. 57, no. 1, pp. 127–139, Jan. 2022.

[50] Z. Wang et al., "An all-weights-on-chip DNN accelerator in 22nm ULL featuring 24×1 mb eRRAM," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2020, pp. 1–2.

[51] M. Giordano et al., "CHIMERA: A 0.92 TOPS, 2.2 TOPS/W edge AI accelerator with 2 MByte on-chip foundry resistive RAM for efficient training and inference," in *Proc. Symp. VLSI Circuits*, Jun. 2021, pp. 1–2.

[52] M. Chang et al., "A 40 nm 60.64TOPS/W ECC-capable compute-in-memory/digital 2.25MB/768KB RRAM/SRAM system with embedded cortex M3 microprocessor for edge recommendation systems," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 65, Feb. 2022, pp. 1–3.

[53] M. Chang et al., "A 73.53TOPS/W 14.74TOPS heterogeneous RRAM in-memory and SRAM near-memory SoC for hybrid frame and event-based target tracking," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2023, pp. 426–428.

[54] H.-H. Hsu et al., "A 22 nm floating-point ReRAM compute-in-memory macro using residue-shared ADC for AI edge device," *IEEE J. Solid-State Circuits*, vol. 60, no. 1, pp. 171–183, Jan. 2025.

[55] K. Prabhu et al., "MINOTAUR: An edge transformer inference and training accelerator with 12 MBytes on-chip resistive RAM and fine-grained spatiotemporal power gating," in *Proc. IEEE Symp. VLSI Technol. Circuits (VLSI Technol. Circuits)*, Jun. 2024, pp. 1–2.

[56] S. Datta, B. Richards, H. Liew, Y. Kim, D. Sun, and J. M. Rabaey, "HDBinaryCore: A 28nm 2048-bit hyper-dimensional biosignal classifier achieving 25 nJ/prediction for EMG hand-gesture recognition," in *Proc. IEEE 49th Eur. Solid State Circuits Conf. (ESSCIRC)*, Sep. 2023, pp. 229–232.

[57] B. Cheng et al., "A 65 nm neuromorphic bio-signal encoder with compute-in-entropy architecture 7.13nJ privacy-preserving encoding and 2.38Mb/mm^2 item memory density," in *Proc. IEEE Symp. VLSI Technol. Circuits (VLSI Technol. Circuits)*, Jun. 2024, pp. 1–2.

[58] J. M. Correll et al., "An 8-bit 20.7 TOPS/W multi-level cell ReRAM-based compute engine," in *Proc. IEEE Symp. VLSI Technol. Circuits*, Jun. 2022, pp. 264–265.

[59] C.-X. Xue et al., "24.1 A 1Mb multibit ReRAM computing-in-memory macro with 14.6ns parallel MAC computing time for CNN based AI edge processors," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2019, pp. 388–390.

[60] J.-H. Yoon, M. Chang, W.-S. Khwa, Y.-D. Chih, M.-F. Chang, and A. Raychowdhury, "A 40-nm 118.44-TOPS/W voltage-sensing compute-in-memory RRAM macro with write verification and multi-bit encoding," *IEEE J. Solid-State Circuits*, vol. 57, no. 3, pp. 845–857, Mar. 2022.

[61] S. D. Spetalnick et al., "A 2.38 MCells/mm^2 9.81 -350 TOPS/W RRAM compute-in-memory macro in 40 nm CMOS with hybrid offset/IOFF cancellation and ICELL RBLSL drop mitigation," in *Proc. IEEE Symp. VLSI Technol. Circuits (VLSI Technol. Circuits)*, Jun. 2023, pp. 1–2.

[62] C.-X. Xue et al., "15.4 A 22nm 2Mb ReRAM compute-in-memory macro with 121-28TOPS/W for Multibit MAC computing for Tiny AI edge devices," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 244–246.

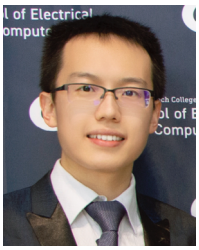
[63] C.-X. Xue et al., "16.1 A 22 nm 4Mb 8b-precision ReRAM computing-in-memory macro with 11.91 to 195.7TOPS/W for tiny AI edge devices," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 64, Feb. 2021, pp. 245–247.

[64] T.-H. Wen et al., "34.8 A 22 nm 16Mb floating-point ReRAM compute-in-memory macro with 31.2TFLOPS/W for AI edge devices," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2024, pp. 580–582.

[65] J.-M. Hung et al., "8-B precision 8-Mb ReRAM compute-in-memory macro using direct-current-free time-domain readout scheme for AI edge devices," *IEEE J. Solid-State Circuits*, vol. 58, no. 1, pp. 303–315, Jan. 2023.

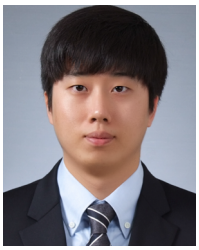
Che-Kai Liu received the B.Eng. degree from Zhejiang University, Hangzhou, China, in 2023. He is currently pursuing the Ph.D. degree with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA.

His research interests include mixed-signal macro and system-on-chip (SoC) with various (post-) silicon technologies for low-power and emerging applications.



Zishen Wan (Graduate Student Member, IEEE) received the B.E. degree in electrical engineering and automation from Harbin Institute of Technology, Harbin, China, in 2018, and the M.S. degree in electrical engineering from Harvard University, Cambridge, MA, USA, in 2020. He is currently pursuing the Ph.D. degree with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA.

His research interests include computer architecture, VLSI, and physical intelligence, with a focus on co-design systems, architectures, and solid-state hardware for autonomous machines and neuro-symbolic AI, enabling next-generation physical intelligence with real-time performance, energy efficiency, and reliability.



Young-Seok Noh received the B.S. degree in electronic systems engineering from Hanyang University, Ansan, South Korea, in 2014, the M.S. degree in electrical engineering from Pohang University of Science and Technology (POSTECH), Pohang, South Korea, in 2016, and the Ph.D. degree in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2021.

From 2021 to 2023, he was a Staff Engineer with Samsung Electronics Ltd., Suwon, South Korea, involved in the design review of power management integrated circuits for mobile applications. He is currently a Research Engineer II with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA. His research interests include integrated circuit design with an emphasis on power management for energy harvesting and data centers, energy-efficient domain-specific accelerators for AI, computation-in-memory, and electromagnetics.

Dr. Noh was a recipient of the Bronze Prize in the 27th Human-Tech Paper Award hosted by Samsung Electronics in 2021.

Mohamed Ibrahim (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Duke University, Durham, NC, USA, in 2018.

He was a Post-Doctoral Researcher from the University of California at Berkeley, Berkeley, CA, USA. He is currently a Senior Research Engineer and Research Faculty Member with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA. His research interests include brain-inspired computing and embodied intelligence, embedded cyber–physical integration, hardware-software co-design, domain-specific architectures, VLSI design, and CAD.

Samuel D. Spetalnick received the B.S. degree in electrical engineering and computer engineering from Johns Hopkins University, Baltimore, MD, USA, in 2018, and the Ph.D. degree from the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA, in 2023.

His research interests include memory circuits and system design for low-power and emerging applications.

Tushar Krishna (Senior Member, IEEE) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology (IIT) Delhi, New Delhi, India, in 2007, the M.S.E. degree in electrical engineering from Princeton University, Princeton, NJ, USA, in 2009, and the Ph.D. degree in electrical engineering and computer science from MIT in 2014.

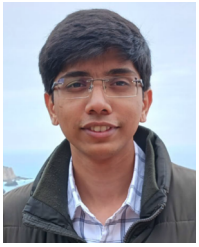
He is an Associate Professor with the School of Electrical and Computer Engineering (ECE), Georgia Institute of Technology (Georgia Tech), Atlanta, GA, USA, with a courtesy appointment in Computer Science. He held the ON Semiconductor (Endowed) Junior Professorship in ECE at Georgia Tech from 2019 to 2021. He has also been a Visiting Professor at MIT EECS + CSAIL, Computer Science Department, Harvard University, Cambridge, MA, USA, and a Researcher at Intel's VSSAD Group. He currently serves as an Associate Director for the Center for Research into Novel Computing Hierarchies (CRNCH)—a cross-disciplinary research center at Georgia Tech. His research spans computer architecture, interconnection networks, networks-on-chip (NoC), and AI/ML accelerator systems—with a focus on optimizing data movement in modern computing platforms.

Dr. Krishna's research is funded via multiple awards from NSF, DARPA, IARPA, SRC (including JUMP2.0), Department of Energy, Intel, Google, Meta/Facebook, Qualcomm, and TSMC. His articles have been cited over 21 000 times. He has been a recipient of multiple IEEE Micro's Top Picks and best paper awards across computer architecture and design-automation conferences. He is part of the Halls of Fame of all three of the premier computer architecture conferences: MICRO, HPCA, and ISCA. He was a recipient of the "Under 40 Innovators Award" at DAC in 2025. He is also a Co-Chair of the Chakra Execution Traces and Benchmarks Working Group within ML Commons.

Win-San Khwa (Senior Member, IEEE) received the B.S. degree from the University of California at Los Angeles, Los Angeles, CA, USA, in 2007, the M.S. degree from the University of Michigan, Ann Arbor, MI, USA, in 2010, and the Ph.D. degree in electrical engineering from National Tsing Hua University, Hsinchu, Taiwan, in 2017.

In 2012, he joined Macronix International (MXIC), Hsinchu, while pursuing the Ph.D. degree. He is currently the Technical Manager with the Corporate Research Design Solution Department, Taiwan Semiconductor Manufacturing Company (TSMC), Hsinchu, where he is working on emerging memory path finding and IP development. His research interests include circuit-device optimization designs of emerging memories for artificial intelligence applications.

Dr. Khwa has been serving as the TPC Member for CICC and DAC since 2021 and 2024, respectively.

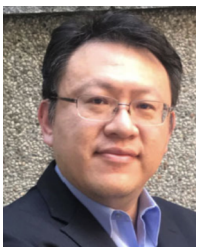


Ashwin Sanjay Lele (Member, IEEE) received the B.Tech. and M.Tech. degrees in electrical engineering from IIT Bombay, Mumbai, India, in 2019, and the Ph.D. degree from the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA, in 2023.

He previously held internship positions from Intel, Bengaluru, India, and Qualcomm Inc., San Jose, CA, USA. He is currently a Principal Engineer with the Corporate Research Department, Taiwan Semiconductor Manufacturing Company (TSMC), San Jose. His research interests broadly lie in low-power hardware design and emerging memory systems and their applications.

Yu-Der Chih received the B.S. degree in physics from National Taiwan University, Taipei, Taiwan, in 1988, and the M.S. degree in electronics engineering from National Tsing-Hua University, Hsinchu, Taiwan, in 1992.

From 1992 to 1997, he was a Design Engineer of Ethernet transceiver circuits for data communication with Macronix, Hsinchu, and a Circuit Design Engineer of SDRAM with Powerchip, Hsinchu. In 1997, he joined Taiwan Semiconductor Manufacturing Company (TSMC), Hsinchu, where he was involved in the development of embedded nonvolatile memory IP, including embedded flash, one-time-programming (OTP), Multiple-Times Programmable (MTP), and emerging memory, and is currently a TSMC Academician and the Director of the Embedded Nonvolatile Memory Library Department, Memory Solution Division.



Meng-Fan Chang (Fellow, IEEE) received the M.S. degree from Pennsylvania State University, State College, PA, USA, and the Ph.D. degree from National Chiao Tung University, Hsinchu, Taiwan.

Before 2006, he worked in the industry for over ten years. This included the design of memory compilers (Mentor Graphics, Wilsonville, OR, USA, from 1996 to 1997) and the design of embedded static random access memory (SRAM) and flash macros (Design Service Division, TSMC, Hsinchu, from 1997 to 2001). In 2001, he cofounded IPLib, Hsinchu, where he developed embedded SRAM and ROM compilers, flash macros, and flat-cell ROM products until 2006. He is currently a Distinguished Professor with National Tsing Hua University (NTHU), Hsinchu, and the Director of Corporate Research, TSMC. His research interests include circuit design for volatile and nonvolatile memory (NVM), ultralow-voltage systems, 3-D memory, circuit–device interactions, spintronic circuits, memristor logics for neuromorphic computing, and computing-in-memory (CIM) for artificial intelligence.

Dr. Chang was a recipient of several prestigious national-level awards in Taiwan, including the Outstanding Research Award from MOST-Taiwan, the Outstanding Electrical Engineering Professor Award, the Academia Sinica Junior Research Investigator Award, and the Ta-You Wu Memorial Award. He was a Distinguished Lecturer of the IEEE Solid-State Circuits Society (SSCS) and the Circuits and Systems Society (CASS), the Chair for the Nano-Giga Technical Committee of CASS, an Administrative Committee (AdCom) Member of the IEEE Nanotechnology Council, the Chair for the SSCS Taipei Chapter, and the Chair for the IEEE Taipei Section. He has been serving as an Associate Editor for IEEE JOURNAL OF SOLID-STATE CIRCUITS (JSSC), IEEE Transactions on Very Large Scale Integration (TVLSI), IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS (TCAS-I), and IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS (TCAD), and a Guest Editor for IEEE JSSC, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS (TCAS-II), and IEEE JOURNAL ON EMERGING AND SELECTED TOPICS IN CIRCUITS AND SYSTEMS (JETCAS). He has also been serving on the Executive Committee for IEDM, as well as the Subcommittee Chairs for ISSCC, IEDM, DAC, ISCAS, VLSI-DAT, and ASP-DAC. He was the Program Director of the Micro-Electronics Program at the Ministry of Science and Technology (MOST), the Associate Executive Director of the National Program of Intelligent Electronics (NPIE), and the NPIE Bridge Program initiated by the Taiwan Government. He received recognition as a top-10 contributor of papers to ISSCC over the past 70 years.

Arijit Raychowdhury (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 2007.

He held research positions at Intel Corporation, Hillsboro, OR, USA, for six years and Texas Instruments, Bengaluru, India, and Dallas, TX, USA, for one and a half years. He joined Georgia Institute of Technology (Georgia Tech), Atlanta, GA, USA, in January 2013. From 2013 to July 2019, he was an Associate Professor and held the ON Semiconductor Junior Professorship with the department. He is currently the Steve W. Chaddick Chair and a Professor with the School of Electrical and Computer Engineering, Georgia Tech. He is also the Director of the Center for the Co-Design of Cognitive Systems (CoCoSys), a Joint University Microelectronics Program 2.0. He holds more than 27 U.S. and international patents and has authored over 300 journal articles and refereed conference papers. His research interests include low-power digital and mixed-signal circuit design, design of power converters, signal processors, and exploring interactions of circuits with device technologies.

Dr. Raychowdhury was a recipient of several prestigious awards, including the SRC Technical Excellence Award in 2021, the Qualcomm Faculty Award in 2021 and 2020, the IEEE/ACM Innovator under 40 Award, the NSF CISE Research Initiation Initiative Award (CRII) in 2015, the Intel Labs Technical Contribution Award in 2011, the Dimitris N. Chorafas Award for outstanding doctoral research and best thesis in 2007, and several fellowships. He and his students received 16 Best Paper Awards over the years. He is a Fellow of the National Academy of Inventors (NAI).