Efficient Processing of Neuro-Symbolic AI: A Tutorial and Cross-Layer Co-Design Case Study

Zishen Wan, Che-Kai Liu, Hanchen Yang, Ritik Raj, Arijit Raychowdhury, Tushar Krishna
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA

Editors: G. Pappas, P. Ravikumar, S. A. Seshia

Abstract

While neural-based models have driven recent breakthroughs in artificial intelligence (AI), they face critical challenges in unsustainable computational demands, limited robustness, and lack of interpretability. Neuro-symbolic (NeSy) AI has emerged as a promising paradigm that integrates neural learning and symbolic reasoning to enhance explainability, robustness, and data efficiency. Recent NeSy systems demonstrated strong potential in reasoning and trustworthy decision-making tasks, making them particularly suitable for cognitive human-AI applications.

This tutorial presents a vertically integrated approach on the efficient processing of NeSy AI, bridging workload characteristics with system and hardware co-design. We begin by systematically categorizing NeSy workloads and analyzing their computational and memory demands to expose performance bottlenecks and optimization opportunities. Building on these insights, we focus on a class of vector-symbolic architecture-based NeSy systems and present a series of hardware case studies, including processing element microarchitecture, dataflow, FPGA design, and system-on-chip prototype. Our results highlight the efficiency and scalability improvements of NeSy systems, with the integration of application discovery, systems thinking, and co-design intelligence. Project Website: https://effi-nesy.github.io.

Keywords: Neuro-Symbolic AI, Workload Characterization, Hardware Acceleration, Domain-Specific Architecture, FPGA, System-on-Chip

1. Introduction

The remarkable success of LLMs, coupled with concerns regarding interpretability and safety, has given rise to the emerging paradigm of compositional AI, particularly in safety-critical domains such as robotics and healthcare. These systems aim to integrate black-box neural networks with reasoning-based AI methods (Wan et al., 2024a,b; Mu et al., 2024; Zhao et al., 2024; Kang and Li, 2024; Kwon et al., 2024; Kalyanpur et al., 2024; Xiong et al., 2024; Ibrahim et al., 2024), closely resembling human cognitive processes. Human cognition typically comprises lower-level sensory perception (System 1) and higher-level logical reasoning and deduction (System 2) (Daniel, 2017; Booch et al., 2021). Compositional AI follows this dual-system principle, leveraging neural networks for perception tasks and symbolic frameworks for logical reasoning (Wan et al., 2025b).

A particularly promising form of compositional intelligence is **neuro-symbolic (NeSy) AI** that synergistically integrates neural network-based learning with symbolic reasoning. Neural networks excel at recognizing patterns and managing perceptual tasks but generally lack transparency and structured reasoning capabilities. Conversely, symbolic systems, characterized by explicit rules and structured knowledge, offer strong interpretability and logical inference but struggle to adapt from raw data. NeSy AI effectively bridges these complementary strengths, enabling enhanced explainability, robustness, and data efficiency of AI systems (Garcez and Lamb, 2023).

NeSy AI has demonstrated impressive capabilities in human-like reasoning and logical inference across diverse domains, including natural language processing, robotics, and healthcare (Garcez and

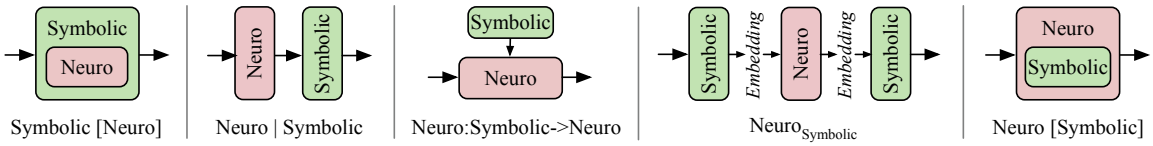

Figure 1: Neuro-symbolic paradigms. Categorization of NeSy AI paradigms based on interactions between neural and symbolic components (Wan et al., 2024c).

Lamb, 2023; Mao et al., 2019; Han et al., 2019; Mei et al., 2022; Yi et al., 2020; Zhang et al., 2021; Shah et al., 2022; Hsu et al., 2023; Wan et al., 2025a; Nayan et al., 2025). For instance, IBM's NVSA (Hersche et al., 2023) achieves 98.8% accuracy on spatial-temporal reasoning tasks (Zhang et al., 2019), significantly outperforming human-level performance, ResNet, and GPT-4. Similarly, Google DeepMind's AlphaGeometry (Trinh et al., 2024; Chervonyi et al., 2025) solves complex geometry problems at the level of Olympiad gold medalists, whereas GPT-4 fails entirely.

Despite these promising algorithmic advancements, NeSy AI faces significant computational challenges. Compared to traditional deep learning workloads, neuro-symbolic computing exhibits increased kernel heterogeneity, greater memory intensity, and irregular data access patterns. These characteristics create a growing mismatch with modern hardware architectures, which are predominantly optimized for matrix operations and convolutions (Samajdar et al., 2020; Kwon et al., 2021; Wu et al., 2023; Ramachandran et al., 2024; Xie et al., 2025; Raj et al., 2025). Such inefficiencies pose a barrier to the scalability and real-world deployment of NeSy systems in the long run.

This tutorial article aim to address these computational challenges and explore efficient hardware architectures tailored for NeSy AI workloads. First, we categorize representative NeSy workloads (Sec. 2) and analyze their computational and memory demands to identify key system bottlenecks and optimization opportunities (Sec. 3). Leveraging these insights, we pick a class of NeSy systems and present several case studies on hardware design, covering processing element microarchitecture, dataflow, FPGA, and system-on-chip (SoC) prototypes (Sec. 4). Additionally, we outline evaluation metrics for assessing NeSy hardware performance (Sec. 5). Collectively, this tutorial highlights how application discovery, systems thinking, and cross-layer co-design can advance the efficiency and scalability of neuro-symbolic computing.

2. Neuro-Symbolic Workload Categorization

2.1. Neuro-Symbolic Paradigms

NeSy AI combines neural learning with symbolic reasoning, effectively leveraging their complementary strengths to enhance accuracy, interpretability, and robustness. Inspired by Henry Kautz's taxonomy and based on how neural and symbolic components interact, we categorize NeSy algorithms into five paradigms (Wan et al., 2024c), as illustrated in Fig. 1.

Symbolic[Neuro]. This paradigm augments symbolic reasoning processes with neural-based statistical learning, primarily using a symbolic problem-solving framework enriched with neural subroutines. Examples include AlphaGo and AlphaZero (Silver et al., 2017; Zhang and Yu, 2020), which employ Monte Carlo Tree Search with neural network-driven state evaluation.

Neuro|Symbolic. This hybrid pipeline delegates distinct and complementary tasks to neural and symbolic components. For instance, neuro-vector-symbolic architecture (NVSA) (Hersche et al., 2023) integrates neural networks for semantic parsing and symbolic reasoning modules for probabilistic abductive reasoning. Similarly, PrAE (Zhang et al., 2021) processes input features using

neural networks and transforms them into hypervectors for symbolic reasoning. Other examples include VSAIT (Theiss et al., 2022), NeuPSL (Pryor et al., 2022), DeepProbLog (Manhaeve et al., 2021), NeurASP (Yang et al., 2020), NSCL (Mao et al., 2019), and NSVQA (Yi et al., 2018).

Neuro:Symbolic→Neuro. This approach incorporates symbolic logic rules directly into neural network structures to guide learning and enhance interpretability. Logical neural networks (LNNs) (Riegel et al., 2020), for instance, enforce symbolic constraints on NN outputs. Other examples include deep learning for symbolic mathematics (Lample and Charton, 2019) and differentiable inductive logic programming (Evans and Grefenstette, 2018).

Neuro$_{\text{Symbolic}}$. This hybrid method maps symbolic rules onto neural network embeddings as soft constraints or regularizers. Logical tensor networks (LTNs) (Badreddine et al., 2022) use logical formulas to define constraints on tensor representations, enhancing knowledge graph completion. Deep ontology networks (DONs) (Hohenecker and Lukas, 2020) follow a similar principle. However, whether this approach compromises interpretability remains an open question.

Neuro[Symbolic]. This category integrates symbolic reasoning into NN processing, enhancing explainability and robustness. Unlike Symbolic[Neuro], where symbolic reasoning guides NN learning, Neuro[Symbolic] embeds symbolic reasoning within the neural model itself. For example, graph neural networks (GNNs) employ attention mechanisms to incorporate symbolic rules (Lamb et al., 2020). Other examples include Neural Logic Machines (NLM) (Dong et al., 2019) and ZeroC (Wu et al., 2022), where symbolic knowledge is represented in graph structures.

2.2. Neuro-Symbolic Kernels

Fig. 2 lists representative computational kernels in NeSy AI, emphasizing the complementary roles of neural and symbolic paradigms. Neural modules provide scalability and robustness, including encoders, vision and language models. Symbolic components offer interpretability and logical consistency, covering methods such as first-order logic, symbolic search, knowledge graphs, algebraic reasoning, and probabilistic models (e.g., HMMs, probabilistic circuits). This classification highlights the diversity and heterogeneity of NeSy kernels, underscoring the importance of understanding their system-level characteristics.

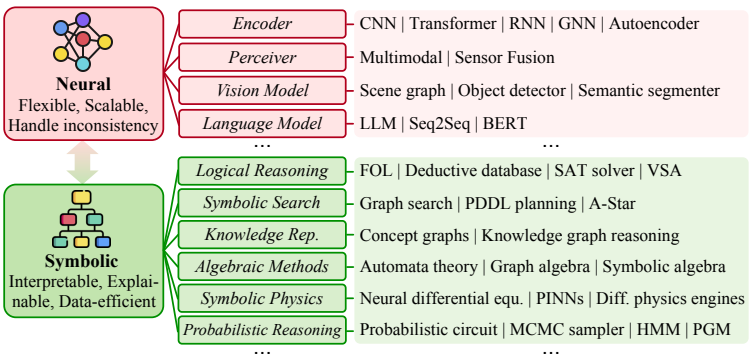

Figure 2: Neuro-symbolic kernel examples. Categorization of NeSy AI kernel examples, highlighting the complementary roles of neural models for perception and symbolic methods for structured reasoning.

3. Neuro-Symbolic Workload Characterization

This section examines the characteristics of NeSy AI workloads, studying their runtime, scalability, compute operators, memory usage, operation graphs, and hardware utilization (Wan et al., 2024c).

3.1. Workload Characterization Methodology

We profile seven NeSy AI workloads spanning different paradigms: LNN(Riegel et al., 2020), LTN (Badreddine et al., 2022), NVSA (Hersche et al., 2023), NLM (Dong et al., 2019), VSAIT (Theiss

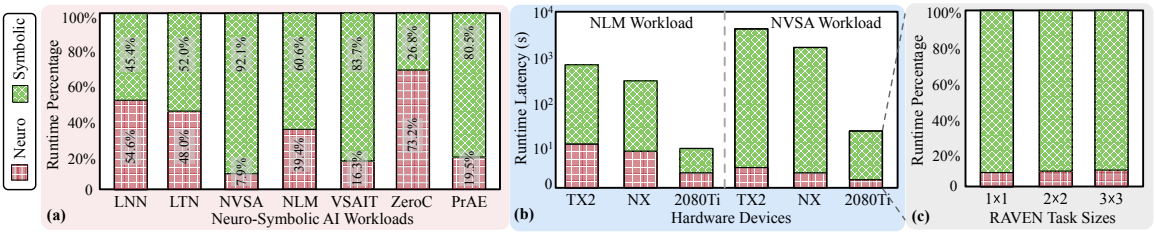

Figure 3: Neural and symbolic latency characterization. **(a)** Runtime breakdown of seven NeSy workloads on CPU+GPU, showing symbolic components as potential bottlenecks. **(b)** NVSA and NLM benchmarks on Jetson TX2, Xavier NX, and RTX GPU, highlighting unmet real-time performance. **(c)** NVSA on varying RPM task sizes, indicating scalability limits and persistent symbolic bottlenecks (Wan et al., 2024c).

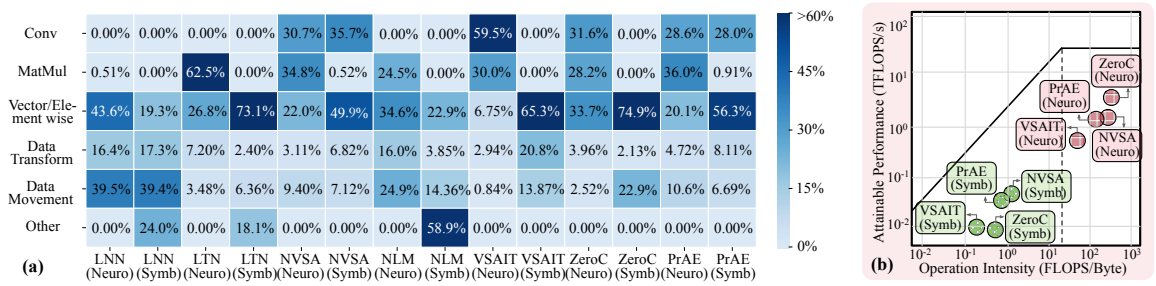

Figure 4: Compute operators and roofline characterization. **(a)** Runtime breakdown showing neural dominated by MatMul and Conv, while symbolic relies on vector/tensor ops. **(b)** Roofline analysis on RTX 2080Ti shows neural ops are compute-bound, while symbolic ops are memory-bound (Wan et al., 2024c).

et al., 2022), ZeroC (Wu et al., 2022), and PrAE (Zhang et al., 2021). Profiling is conducted on Intel Xeon Silver 4114 CPU, Nvidia RTX 2080 Ti GPU, and edge SoCs (Xavier NX, Jetson TX2).

3.2. Compute Latency Analysis

End-to-end latency breakdown. Fig. 3 shows that symbolic workloads are non-negligible in computation time and can become system bottlenecks. In NVSA, symbolic reasoning accounts for 92.1% of total inference time despite contributing only 19% of FLOPs, highlighting inefficient execution on GPUs. Furthermore, real-time constraints remain unmet, with RTX 2080 Ti taking 380s and Jetson TX2 7507s for the RPM task, suggesting a significant performance gap.

End-to-end latency scalability. As task complexity increases, symbolic workloads consistently dominate runtime, growing quadratically (Fig. 3c). For example, when task size scales from 2×2 to 3×3, symbolic runtime remains above 87%, reinforcing concerns about scalability.

3.3. Compute Operator Analysis

Fig. 4a partitions the neural and symbolic workloads into six operator categories with runtime latency breakdown. We make the following observations:

Neural operator analysis. Dominated by matrix multiplications (MatMul) and activations. NVSA, VSAIT, and PrAE rely on convolution-based feature extraction, whereas LNN and NLM emphasize vector and element-wise tensor operations due to logic-based computations.

Symbolic operator analysis. Dominated by vector and scalar operations with low compute intensity and irregular control flow. LNN and NLM involve frequent logical operations. NVSA relies on high-dimensional vector-symbolic transformations, which GPUs struggle to execute efficiently.

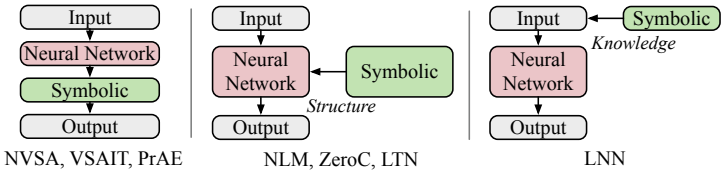

Figure 5: Operator graph analysis. Symbolic operation depends on neural results or needs to compile in neural structure as the critical path. Complex control and symbolic-only phase operation result in inefficiency and low hardware resource utilization (Wan et al., 2024c).

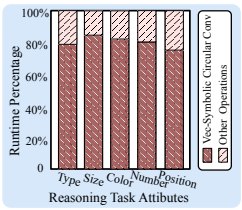

Figure 6: Symbolic operations are dominated by vector-symbolic circular convolution (Wan et al., 2025b).

3.4. System and Dependency Graph Analysis

Roofline analysis. Fig. 4b reveals that neural computations are compute-bound, while symbolic operations are memory-bound due to large streaming operations (e.g., NVSA and PrAE). Optimized dataflow architectures and scalable processing elements are critical to alleviating these bottlenecks.
Dependency graph analysis. Fig. 5 analyzes the operation dependency in selected NeSy workloads. We observe that the symbolic computation of NVSA, VSAIT, and PrAE depends on the result of the frontend neural workload and thus lies on the critical path during inference. LNN, LTN, NLM, and ZeroC need to compile the symbolic knowledge in neural representation or input embeddings. The complex control results in inefficiency in CPU and GPU, and the vector-symbolic computation period results in low hardware utilization. There are opportunities for data pre-processing, parallel rule query, and heterogeneous and reconfigurable hardware design to reduce this bottleneck.

3.5. Hardware Inefficiency Analysis

Symbolic operation analysis. As shown in Fig. 6, we find that in vector-symbolic-based NeSy workloads, symbolic workloads are dominated by vector-symbolic circular convolutions and vector-vector multiplications, accounting for around 80% of runtime. These hypervector operations remain computationally inefficient on GPUs, requiring dedicated acceleration strategies.
Symbolic hardware inefficiency analysis. NeSy workloads suffer from ALU underutilization, low cache efficiency, and excessive data transfer. Using Nsight Systems and Compute, we find that in NVSA, symbolic operations exhibit ALU utilization $<$10%, L1 and L2 cache hit rates of around 20% and 40%, respectively, and DRAM bandwidth usage near 90%, indicating memory-bounded execution. Data transfers contribute around 50% of total latency, with $>$80% occurring between CPU and GPU. Additionally, synchronization overheads further reduce CPU utilization.

3.6. Uniqueness of Neuro-Symbolic vs. Neural Networks

To summarize, NeSy AI workloads differ from neural networks mainly in the following aspects:
Compute kernels. NeSy workloads consist of heterogeneous neural and symbolic kernels. The symbolic operators (e.g., vector, graph, logic) are processed inefficiently on off-the-shelf CPUs/GPUs with low hardware utilization and cache hit and may result in runtime latency bottleneck.
Memory. NNs exhibit predictable and structured memory access patterns with high data reuse, enabling effective caching and high memory bandwidth utilization. Symbolic operations tend to be memory-bounded due to large element streaming with low reuse and irregular access patterns.
Dataflow and scalability. NNs typically follow a layered feedforward dataflow with well-defined parallelism. In contrast, NeSy workloads involve complex control flow, sequential symbolic phases, and fine-grained dependencies between neural and symbolic, limiting hardware parallelism.

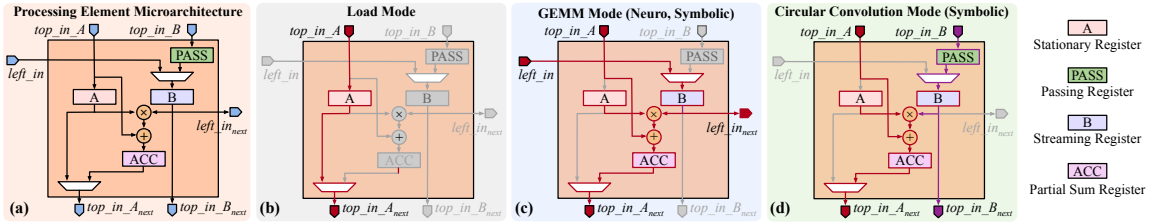

Figure 7: Reconfigurable neuro/symbolic PE. Each PE includes four registers and supports three modes (load, neuro, symbolic) that provide reconfigurable support for neurosymbolic operations (Wan et al., 2025b).

4. Case Study: Domain-Specific Hardware for Neuro-Symbolic AI

4.1. Software-Hardware Co-Design Methodology for Neuro-Symbolic AI

Efficient processing of NeSy AI requires a software-hardware co-design approach to support heterogeneous workloads that blend neural learning and symbolic reasoning. Unlike traditional AI systems optimized for matrix operations, NeSy workloads demand compute and memory primitives capable of handling both neural operations (e.g., GEMM, convolution) and irregular, memory-bound symbolic operations (e.g., logic, graph traversal, vector-symbolic ops).

Our structured software-hardware co-design method follows four key principles: (1) Algorithm-aware hardware design: tailor architecture to support the demands of neural and symbolic workloads, including reconfigurable primitives and hybrid dataflows. (2) Hardware-aware algorithm optimization: adapt NeSy models to maximize hardware utilization, improve parallelism, and reduce memory bottlenecks. (3) Optimized dataflow and memory management: enhance scheduling, caching, and data movement strategies to reduce latency and increase throughput. (4) Adaptability: design architectures that are adaptable across different hardware platforms, from FPGAs to ASICs.

To illustrate these principles, in this tutorial, we pick a class of vector-symbolic architecture-based NeSy systems and present a series of domain-specific hardware case studies:

Microarchitecture design. CogSys (Wan et al., 2025b), a reconfigurable NeSy processing element that supports both GEMM-based neural and circular convolutions-based symbolic operations via a bubble streaming dataflow. This design enhances parallelism and reduces memory overhead, achieving 75× speedup on symbolic tasks while minimizing area overhead (Sec. 4.2).

FPGA-based acceleration. NSFlow (Yang et al., 2025), an end-to-end framework that extracts NeSy workload traces, generates optimized dataflow graphs, and maps them onto FPGA for acceleration. It employs CogSys reconfigurable neuro/symbolic arrays and SIMD units, and exploit operator- and loop-level parallelism while managing memory footprints effectively (Sec. 4.3).

System-on-chip (SoC) design. We tape out a heterogeneous SoC with co-integrated RRAM (for neural) and SRAM (for symbolic). A software-defined scheduler dynamically manages power and compute resources based on workload characteristics, achieving 10.8 TOPS/W peak efficiency. Fine-grained power gating further saves energy for real-time NeSy AI applications (Sec. 4.4).

4.2. Microarchitecture Design for Neuo-Symbolic AI

Vector-symbolic circular convolution. As shown in Fig. 6, circular convolution is the dominant symbolic operation. It combines two vectors to encode composite symbols while preserving structure, making it ideal for hierarchical reasoning. Given vectors $\boldsymbol{A}$ and $\boldsymbol{B}$ of dimension N), the result vector $\boldsymbol{C}$ is computed as $C[n] = \sum_{k=0}^{N-1} A[k] \cdot B[(n-k) \mod N]$. This operation is repeated for each element n (0 to $N-1$), effectively multiplying $\boldsymbol{A}$ with circularly shifted versions of $\boldsymbol{B}$. Its commutative and associative properties enable efficient manipulation of structured information.

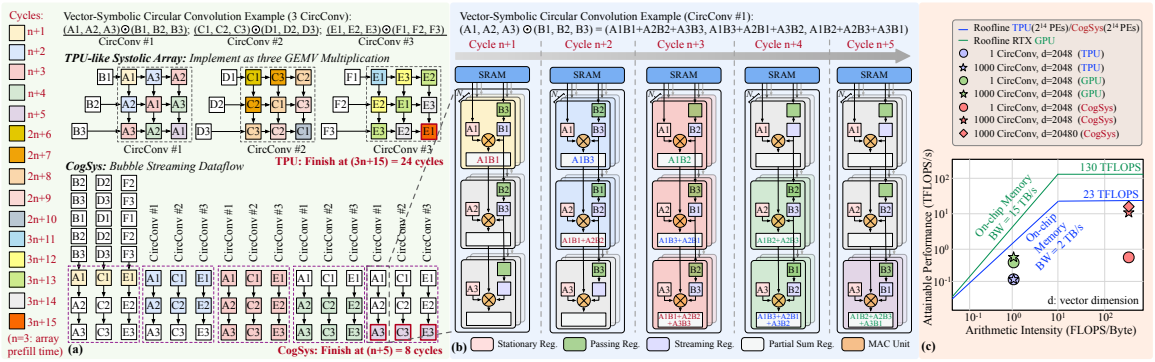

Figure 8: Bubble streaming (*BS*) dataflow and performance comparison. **(a)** Compute cycle and mapping comparison of TPU-like systolic array vs. CogSys *BS* dataflow for circular convolutions. **(b)** *BS* dataflow example on a 3×1 *nsPE* array for vectors $\boldsymbol{A}$, $\boldsymbol{B}$ (d=3). **(c)** Roofline analysis comparing CogSys *BS* dataflow (compute-bound), TPU, and GPU (memory-bound) implementations (Wan et al., 2025b).

Reconfigurable neuro/symbolic PE (*nsPE*). To efficiently support both GEMM-based neural and circular convolution-based symbolic operations, we propose *nsPE* micro-architecture that provides reconfigurable support to both neuro and symbolic operations (Fig. 7). Each *nsPE* consists of four registers (stationary, passing, streaming, and partial sum registers) and supports three operation modes (load, GEMM, and circular convolution). During load mode, the input vectors $\boldsymbol{A}$ (weights of GEMM) are passed into the stationary register. Reconfigurability is achieved by selecting input $\boldsymbol{B}$ either from 'left_in' link (GEMM mode) or the passing register (circular convolution mode). During GEMM mode, the *nsPE* operates as TPU-like architecture for efficient GEMM and convolution. During circular convolution mode, input vector $\boldsymbol{B}$ is streamed from top to bottom using 'top_in_B' links with a bubble via passing register, facilitating the temporal reuse of the streaming input for efficient vector-symbolic circular convolution operation. The reconfigurable *nsPE* can also support efficient circular correlation by reversing stationary vector $\boldsymbol{A}$. During both GEMM and circular convolution modes, partial products are reduced from top to bottom with 'top_in_A' links.

Inefficiency of TPU-like systolic array. TPU-like systolic array (SA) exhibits high memory footprint and low parallelism for symbolic circular convolution operations. Fig. 8a shows a scenario of three circular convolutions. TPU-like systolic cell implements them as general matrix-vector (GEMV) multiplication where matrices contain circularly shifted stationary vectors with the matrix memory footprint of $O(d^2)$. Additionally, TPU-like SA is incapable of parallelizing multiple GEMV on a systolic cell and need to process them sequentially.

Bubble streaming (*BS*) dataflow. To efficiently support symbolic operations in *nsPE* array, we propose *BS* dataflow for circular convolution and circular correlation (opposite direction circular shift) which is the vector-symbolic bottleneck. Fig. 8b presents an example of *BS* dataflow performing circular convolution of two vectors $\boldsymbol{A}$ and $\boldsymbol{B}$ (d=3) on a 3×1 *nsPE* array. In *BS* dataflow, vector $\boldsymbol{B}$ is streamed from one *nsPE* to another through bubbles while vector $\boldsymbol{A}$ is held in stationary registers. The *BS* dataflow enables a passing register to temporarily store the streaming input for a cycle before it moves to the streaming register. This value is transferred to the passing register of the next *nsPE* in the following cycle. The MAC unit processes the data from stationary and streaming registers, adding it to the partial product. The procedure is repeated until final outputs.

Improved arithmetic intensity of *BS* dataflow. The *BS* dataflow achieves higher arithmetic intensity than GEMV in GPU/TPU-like systolic cells, as illustrated in roofline analysis (Fig. 8c). This efficiency mainly comes from reduced memory footprint and increased parallelism. Additionally, GPUs require extra computations to handle the index calculations for the circularly shifted vector.

Adaptive scheduling (*adSCH*). NeSy systems face two key challenges (Fig. 9a): (1) sequential neuro-symbolic execution leads to high latency; (2) heterogeneous kernels cause low utilization. To address this, CogSys employs *adSCH* scheme that greatly improves performance and resource efficiency (Fig. 9b). (1) Interleaved neuro/symbolic processing. Symbolic operations of other tasks are interleaved during neural layer processing of the current task using reconfigurable nsPE arrays. (2) Adaptive neuro/symbolic array partition. CogSys dynamically allocates processing elements to neural or symbolic kernels (cell-wise) and parallel symbolic tasks (column-wise), enabling balanced execution across diverse workloads and maximizing parallelism.

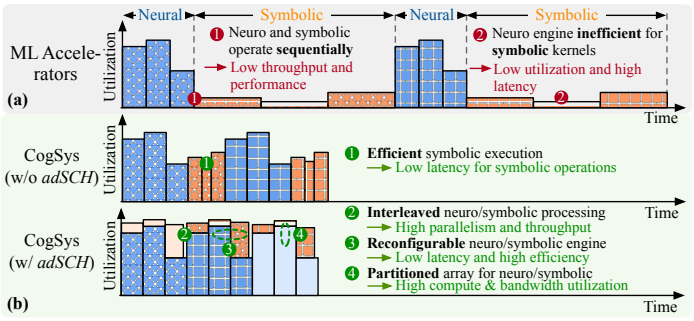

Figure 9: Adaptive workload-aware scheduling (*adSCH*). (a) NeSy system-level challenges. **(b)** *adSCH* enables interleaved neural/symbolic processing and cell-wise partition across CogSys arrays with multi-level parallelism (Wan et al., 2025b).

4.3. FPGA Design for Neuro-Symbolic AI

Building on CogSys PE and dataflow, we introduce NSFlow, an end-to-end framework for FPGA-based acceleration of NeSy AI. NSFlow enables automated analysis, design space exploration, and hardware generation tailored to NeSy workloads. An overview of NSFlow is shown in Fig. 10a.

NSFlow frontend. The frontend identifies NeSy workload data dependencies and determines the mapping strategy. It begins by extracting an execution trace from NeSy workloads and constructing a dataflow graph that captures operator-level specifications, runtime behaviors, memory access patterns, and inter-operator dependencies. Design Architecture Generator (DAG) then performs a two-stage optimization: (1) **System configuration search:** identify the optimal FPGA system setup (e.g., on-chip memory allocation, array size). (2) **Mapping strategy selection:** determine efficient reconfiguration and task-to-hardware mapping schemes. The resulting configuration is embedded in host code, which later coordinates FPGA execution via the XRT API.

NSFlow backend. The backend features a predefined accelerator template with: BRAM blocks for flexible on-chip memory; CogSys reconfigurable and adaptive neuro/symbolic arrays for parallel NeSy processing; SIMD units for element-wise operations, vector reduction, and scalar computation; Control logic for efficient task scheduling. These components are parameterized based on

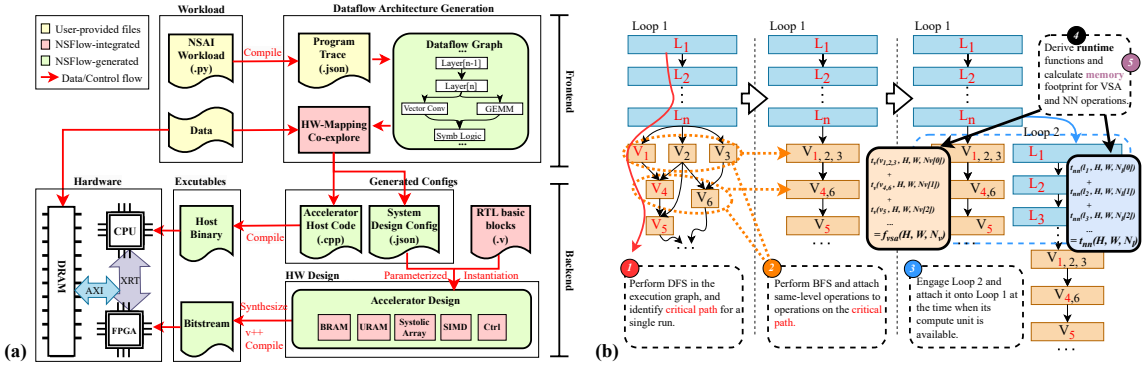

Figure 10: End-to-end FPGA-based NeSy acceleration. (a) NSFlow framework overview. **(b)** Dataflow architecture generation and design space exploration (Yang et al., 2025).

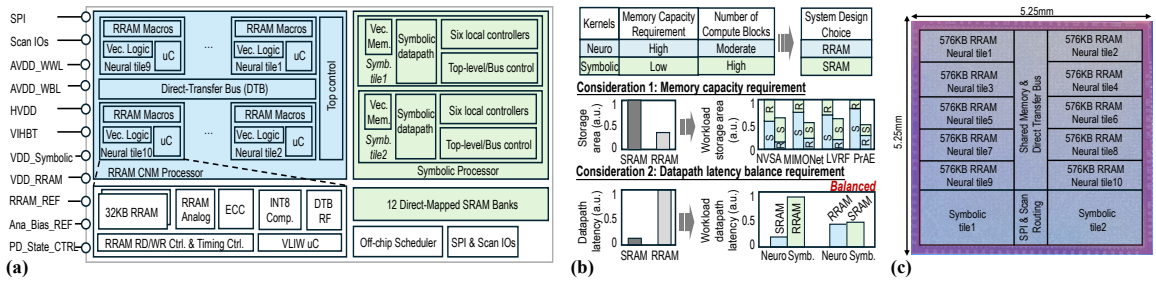

Figure 11: System-on-chip design for NeSy acceleration. **(a)** Top-level diagram of the taped-out NeSy SoC. **(b)** Balanced RRAM/SRAM system datapath. **(c)** Chip die shot under TSMC 40nm technology.

workload characterization and the system configuration file from the frontend. NSFlow then synthesizes and compiles RTL into an executable bitstream for FPGA deployment.

Dataflow graph. Fig. 10b illustrates how NSFlow constructs the dataflow graph. (1) **Critical path identification:** A depth-first search (DFS) identifies the longest dependency chain within a single loop. (2) **Inner-loop parallelism identification:** DAG walks through the graph again with breadth-first search (BFS), to identify operation nodes at the same depth as the nodes on the critical path, and attach them to the corresponding critical-path nodes, indicating their earliest execution and parallelisms. (3) **Inter-loop parallelism identification:** DAG connects subsequent loops by scheduling the next loop's operations as soon as compute units are available. (4) **Runtime estimation:** For each node, runtime functions are derived based on workload parameters (i.e., vector size n, dimension d, NN layer dimensions in m, n, k, etc.) and hardware configurations (i.e., sub-array dimensions, partitioning). (5) **Memory analysis:** DAG computes the memory footprint of each node to guide block-level allocation. This process ensures efficient FPGA-based acceleration of NeSy AI, balancing computational performance, memory efficiency, and parallel execution.

4.4. System-on-Chip Design for Neuro-Symbolic AI

To further advance integration and efficiency, we develop and tape out a fully programmable heterogeneous SoC on TSMC 40nm technology, tailored for real-time NeSy AI inference. The SoC supports diverse neural and symbolic operators through software-hardware co-design featuring: (1) integrated RRAM and SRAM neural-symbolic data paths, (2) ultra-dense energy-efficient RRAM macros with edge-triggered and tunable sensing, (3) scheduler-informed power management, and (4) programming support for variable resolution, vector lengths, and batching.

Chip architecture. As shown in Fig. 11a, 11c, the chip includes ten 576K RRAM-based neural tiles and two SRAM-based symbolic tiles. Neural tiles consist of local RRAM storage and vector compute units, managed by a controller handling memory access, I/O, and power gating. Symbolic tiles incorporate logic and SRAM for bit-wise ops, CA-90 units, distance computation, and multi-precision MACs. Both tile types support module-level clock gating and core-level power gating.

Hybrid memory rationale. The rationale behind the hybrid RRAM/SRAM design for accelerating NeSy models is memory requirements and balanced data-path latency considerations (Fig. 11b): neural kernels require large memory capacity, making RRAM ideal due to its density; symbolic kernels benefit from SRAM's lower latency for compute-heavy logic. All tiles are integrated with MAC capability and are allowed to read from a shared memory. To prevent write conflicts among the tiles, all 12 tiles are only allowed to write to the designated bank in the shared memory. Neural tile is also integrated with SECDED ECC logic for low bit-error rate memory readout (Crafton et al., 2022; Chang et al., 2023). The 128-bit instruction is separated into five address spaces, in which the control field provides power control or control-related instructions.

Power efficiency and scheduling. The SoC also features multiple local power switches on neural tiles. Post-layout/APR analysis reveals that fine-grained power control provides 5× power reduction owing to the dynamic data flow in NeSy models. A software-defined power management scheme is designed to schedule NeSy models based on the operators and the ratio of neural and symbolic components. The former is obtained through off-chip software, and the latter is optimized based on the compute capacity in neural tiles and symbolic tiles.

4.5. Towards Future-Generation Neuro-Symbolic Processors

By leveraging a co-design methodology that jointly optimizes algorithms, microarchitecture, memory hierarchy, and on-chip systems, our NeSy hardware achieves significant improvements in performance, energy efficiency, and scalability. Looking forward, future research directions include: (1) chiplet-based heterogeneous integration for modular neural-symbolic systems, (2) near-memory computing to reduce data movement in symbolic workloads, (3) hybrid analog-digital processing for ultra-efficient cognitive AI, and (4) more NeSy operators support. These advances will be critical for building the next generation of intelligent, efficient, and interpretable neuro-symbolic processors.

5. Benchmarking Metrics for Neuro-Symbolic Hardware

As efforts to optimize NeSy AI processing continue, standardized benchmarking metrics are essential for comparing different designs and techniques. These metrics should capture key attributes such as accuracy, robustness, energy efficiency, latency, throughput, and hardware cost.

Energy and power efficiency is especially critical for NeSy AI deployed in power-constrained environments such as edge devices (e.g., smartphones, sensors, UAVs) and power-limited data centers. Due to latency, privacy, and bandwidth constraints, edge computing is often favored over cloud-based solutions. Power evaluations must account for all components, including compute units and off-chip memory access.

Latency, defined as the time from input to output, is vital for real-time applications like robotics and autonomous navigation. Such tasks require rapid inference to support timely responses. However, minimizing latency often conflicts with maximizing throughput, making balanced optimization essential.

Throughput enables rapid decision-making by allowing more data to be processed per unit time. This is especially important as workloads become increasingly data-intensive in applications such as human-AI collaboration, physical agents, and trustworthy medical diagnostics.

Hardware cost is largely determined by on-chip memory and core count. Embedded processors have limited memory capacity, requiring a balance between internal storage and external bandwidth. While more cores can boost throughput, utilization often suffers due to inefficiencies in task mapping and bandwidth constraints. Therefore, evaluations should reflect performance on real-world NeSy workloads, not just peak theoretical capabilities

6. Conclusion

NeSy AI is an emerging paradigm for building efficient, robust, explainable, and cognitively advanced AI systems. This tutorial systematically characterizes NeSy system performance and deconstructs its key operational components. Using profiling insights, we explore case studies on NeSy hardware to optimize performance and efficiency. We hope this tutorial sheds light on critical challenges and uncovers opportunities for advancing next-generation NeSy AI systems.

Acknowledgments

The authors thank Ananda Samajdar (IBM), Win-San Khwa, Yu-Der Chih, and Meng-Fan Chang (TSMC) for their technical support. This work was supported in part by CoCoSys, one of the seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

References

Samy Badreddine, Artur d'Avila Garcez, Luciano Serafini, and Michael Spranger. Logic tensor networks. *Artificial Intelligence*, 303:103649, 2022.

Grady Booch, Francesco Fabiano, Lior Horesh, Kiran Kate, Jonathan Lenchner, Nick Linck, Andreas Loreggia, Keerthiram Murgesan, Nicholas Mattei, Francesca Rossi, et al. Thinking fast and slow in ai. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, pages 15042–15046, 2021.

Muya Chang, Ashwin Sanjay Lele, Samuel D Spetalnick, Brian Crafton, Shota Konno, Zishen Wan, Ashwin Bhat, Win-San Khwa, Yu-Der Chih, Meng-Fan Chang, et al. A 73.53 tops/w 14.74 tops heterogeneous rram in-memory and sram near-memory soc for hybrid frame and event-based target tracking. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 426–428. IEEE, 2023.

Yuri Chervonyi, Trieu H Trinh, Miroslav Olšák, Xiaomeng Yang, Hoang Nguyen, Marcelo Menegali, Junehyuk Jung, Vikas Verma, Quoc V Le, and Thang Luong. Gold-medalist performance in solving olympiad geometry with alphageometry2. *arXiv preprint arXiv:2502.03544*, 2025.

Brian Crafton, Zishen Wan, Samuel Spetalnick, Jong-Hyeok Yoon, Wei Wu, Carlos Tokunaga, Vivek De, and Arijit Raychowdhury. Improving compute in-memory ecc reliability with successive correction. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 745–750, 2022.

Kahneman Daniel. *Thinking, fast and slow*. 2017.

Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. Neural logic machines. In *International Conference on Learning Representations (ICLR)*, 2019.

Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61:1–64, 2018.

Artur d'Avila Garcez and Luis C Lamb. Neurosymbolic ai: The 3 rd wave. *Artificial Intelligence Review*, pages 1–20, 2023.

Chi Han, Jiayuan Mao, Chuang Gan, Josh Tenenbaum, and Jiajun Wu. Visual concept-metaconcept learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

Michael Hersche, Mustafa Zeqiri, Luca Benini, Abu Sebastian, and Abbas Rahimi. A neuro-vector-symbolic architecture for solving raven's progressive matrices. *Nature Machine Intelligence*, 5 (4):363–375, 2023.

Patrick Hohenecker and Thomas Lukas. Ontology reasoning with deep neural networks. *Journal of Artificial Intelligence Research*, 68:503–540, 2020.

Joy Hsu, Jiayuan Mao, and Jiajun Wu. Ns3d: Neuro-symbolic grounding of 3d objects and relations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2614–2623, 2023.

Mohamed Ibrahim, Zishen Wan, Haitong Li, Priyadarshini Panda, Tushar Krishna, Pentti Kanerva, Yiran Chen, and Arijit Raychowdhury. Special session: Neuro-symbolic architecture meets large language models: A memory-centric perspective. In *2024 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*, pages 11–20. IEEE, 2024.

Aditya Kalyanpur, Kailash Saravanakumar, Victor Barres, Jennifer Chu-Carroll, David Melville, and David Ferrucci. Llm-arc: Enhancing llms with an automated reasoning critic. *arXiv preprint arXiv:2406.17663*, 2024.

Mintong Kang and Bo Li. R^2-guard: Robust reasoning enabled llm guardrail via knowledge-enhanced logical reasoning. *arXiv preprint arXiv:2407.05557*, 2024.

Hyoukjun Kwon, Liangzhen Lai, Michael Pellauer, Tushar Krishna, Yu-Hsin Chen, and Vikas Chandra. Heterogeneous dataflow accelerators for multi-dnn workloads. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 71–83. IEEE, 2021.

Joseph Kwon, Josh Tenenbaum, and Sydney Levine. Neuro-symbolic models of human moral judgment. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 46, 2024.

Luis C Lamb, Artur Garcez, Marco Gori, Marcelo Prates, Pedro Avelar, and Moshe Vardi. Graph neural networks meet neural-symbolic computing: A survey and perspective. In *IJCAI 2020-29th International Joint Conference on Artificial Intelligence*, 2020.

Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *International Conference on Learning Representations (ICLR)*, 2019.

Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Neural probabilistic logic programming in deepproblog. *Artificial Intelligence*, 298:103504, 2021.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *International Conference on Learning Representations (ICLR)*, 2019.

Lingjie Mei, Jiayuan Mao, Ziqi Wang, Chuang Gan, and Joshua B Tenenbaum. Falcon: fast visual concept learning by integrating images, linguistic descriptions, and conceptual relations. *International Conference on Learning Representations (ICLR)*, 2022.

Tong Mu, Alec Helyar, Johannes Heidecke, Joshua Achiam, Andrea Vallone, Ian Kivlichan, Molly Lin, Alex Beutel, John Schulman, and Lilian Weng. Rule based rewards for language model safety. *Open AI*, 2024.

Md Mizanur Rahaman Nayan, Che-Kai Liu, Zishen Wan, Arijit Raychowdhury, and Azad J Naeemi. Hydra: Sot-cam based vector symbolic macro for hyperdimensional computing. *arXiv preprint arXiv:2504.14020*, 2025.

Connor Pryor, Charles Dickens, Eriq Augustine, Alon Albalak, William Wang, and Lise Getoor. Neupsl: Neural probabilistic soft logic. *arXiv preprint arXiv:2205.14268*, 2022.

Ritik Raj, Sarbartha Banerjee, Nikhil Chandra, Zishen Wan, Jianming Tong, Ananda Samajdhar, and Tushar Krishna. Scale-sim v3: A modular cycle-accurate systolic accelerator simulator for end-to-end system analysis. *arXiv preprint arXiv:2504.15377*, 2025.

Akshat Ramachandran, Zishen Wan, Geonhwa Jeong, John Gustafson, and Tushar Krishna. Algorithm-hardware co-design of distribution-aware logarithmic-posit encodings for efficient dnn inference. In *Proceedings of the 61st ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2024.

Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, et al. Logical neural networks. *arXiv preprint arXiv:2006.13155*, 2020.

Ananda Samajdar, Jan Moritz Joseph, Yuhao Zhu, Paul Whatmough, Matthew Mattina, and Tushar Krishna. A systematic methodology for characterizing scalability of dnn accelerators using scale-sim. In *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 58–68. IEEE, 2020.

Vishwa Shah, Aditya Sharma, Gautam Shroff, Lovekesh Vig, Tirtharaj Dash, and Ashwin Srinivasan. Knowledge-based analogical reasoning in neuro-symbolic latent spaces. *arXiv preprint arXiv:2209.08750*, 2022.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

Justin Theiss, Jay Leverett, Daeil Kim, and Aayush Prakash. Unpaired image translation via vector symbolic architectures. In *European Conference on Computer Vision (ECCV)*, pages 17–32. Springer, 2022.

Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

Zishen Wan, Che-Kai Liu, Hanchen Yang, Chaojian Li, Haoran You, Yonggan Fu, Cheng Wan, Tushar Krishna, Yingyan Lin, and Arijit Raychowdhury. Towards cognitive ai systems: a survey and prospective on neuro-symbolic ai. *arXiv preprint arXiv:2401.01040*, 2024a.

Zishen Wan, Che-Kai Liu, Hanchen Yang, Ritik Raj, Chaojian Li, Haoran You, Yonggan Fu, Cheng Wan, Sixu Li, Youbin Kim, et al. Towards efficient neuro-symbolic ai: From workload characterization to hardware architecture. *IEEE Transactions on Circuits and Systems for Artificial Intelligence (TCASAI)*, 2024b.

Zishen Wan, Che-Kai Liu, Hanchen Yang, Ritik Raj, Chaojian Li, Haoran You, Yonggan Fu, Cheng Wan, Ananda Samajdar, Yingyan Celine Lin, et al. Towards cognitive ai systems: Workload and characterization of neuro-symbolic ai. In *2024 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 268–279. IEEE, 2024c.

Zishen Wan, Yuhang Du, Mohamed Ibrahim, Jiayi Qian, Jason Jabbour, Yang Zhao, Tushar Krishna, Arijit Raychowdhury, and Vijay Janapa Reddi. Reca: Integrated acceleration for real-time and efficient cooperative embodied autonomous agents. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pages 982–997, 2025a.

Zishen Wan, Hanchen Yang, Ritik Raj, Che-Kai Liu, Ananda Samajdar, Arijit Raychowdhury, and Tushar Krishna. Cogsys: Efficient and scalable neurosymbolic cognition system via algorithm-hardware co-design. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 775–789. IEEE, 2025b.

Tailin Wu, Megan Tjandrasuwita, Zhengxuan Wu, Xuelin Yang, Kevin Liu, Rok Sosic, and Jure Leskovec. Zeroc: A neuro-symbolic model for zero-shot concept recognition and acquisition at inference time. *Advances in Neural Information Processing Systems (NeurIPS)*, 35, 2022.

Yannan Nellie Wu, Po-An Tsai, Saurav Muralidharan, Angshuman Parashar, Vivienne Sze, and Joel Emer. Highlight: Efficient and flexible dnn acceleration with hierarchical structured sparsity. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1106–1120, 2023.

Tong Xie, Jiawang Zhao, Zishen Wan, Zuodong Zhang, Yuan Wang, Runsheng Wang, Ru Huang, and Meng Li. Realm: Reliable and efficient large language model inference with statistical algorithm-based fault tolerance. *arXiv preprint arXiv:2503.24053*, 2025.

Haoyi Xiong, Zhiyuan Wang, Xuhong Li, Jiang Bian, Zeke Xie, Shahid Mumtaz, and Laura E Barnes. Converging paradigms: The synergy of symbolic and connectionist ai in llm-empowered autonomous agents. *arXiv preprint arXiv:2407.08516*, 2024.

Hanchen Yang, Zishen Wan, Ritik Raj, Joongun Park, Ziwei Li, Ananda Samajdar, Arijit Raychowdhury, and Tushar Krishna. Nsflow: An end-to-end fpga framework with scalable dataflow architecture for neuro-symbolic ai. *ACM/IEEE Design Automation Conference (DAC)*, 2025.

Zhun Yang, Adam Ishay, and Joohyung Lee. Neurasp: Embracing neural networks into answer set programming. In *29th International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B Tenenbaum. Clevrer: Collision events for video representation and reasoning. In *International Conference on Learning Representations (ICLR)*, 2020.

Chi Zhang, Feng Gao, Baoxiong Jia, Yixin Zhu, and Song-Chun Zhu. Raven: A dataset for relational and analogical visual reasoning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 5317–5327, 2019.

Chi Zhang, Baoxiong Jia, Song-Chun Zhu, and Yixin Zhu. Abstract spatial-temporal reasoning via probabilistic abduction and execution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9736–9746, 2021.

Hongming Zhang and Tianyang Yu. Alphazero. *Deep Reinforcement Learning: Fundamentals, Research and Applications*, pages 391–415, 2020.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, and Bin Cui. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*, 2024.