ARCADE: Circuit-Level Simulation and Analysis EDA Framework for Memristive Logic-In-Memory

Peter M. Hinkel[†,*], Fabian Seiler[†,*], and Nima TaheriNejad[†]
[†]Institute of Computer Engineering (ZITI), Heidelberg University, Germany
{peter.hinkel, fabian.seiler, nima.taherinejad}@ziti.uni-heidelberg.de, [*]These authors contributed equally

***Abstract*—Memristive Logic in Memory (LiM) refers to computation directly within crossbar memory arrays, including stateful logic operations, where both inputs and outputs are represented by the resistive states of memristors. Although several synthesis and mapping tools for LiM have been proposed, circuit-level validation is often neglected, limiting their applicability for comprehensive realistic evaluations. To address these limitations of existing tools, we present *ARCADE*, an Electronic Design Automation (EDA) framework for automated circuit-level simulation and analysis of memristive LiM that unifies support for multiple logic families, mixed-logic, row- and column-oriented, and Single Instruction Multiple Data (SIMD) execution. It provides automated energy estimation, functional validation, and robustness analysis under device variability. Our proposed interchangeable LiM representation (.ilim) additionally provides flexible interfaces to existing synthesis and mapping tools, enabling integration into highly automated end-to-end design flows. ARCADE enables circuit-level optimization workflows that achieve energy reductions of 9.67–95.07% across representative adder designs. We also evaluate ISCAS'85 benchmark circuits under non-ideal device conditions, identifying reliability issues and deriving corresponding mitigation strategies. Together, these show how ARCADE enables automated, hardware-aware design and evaluation of reliable memristive LiM systems.**

I. Introduction

With the slowdown of conventional performance scaling, memristive LiM has emerged as a promising In-Memory Computation (IMC) paradigm that mitigates the von Neumann bottleneck by performing computation directly within memory arrays [1]–[3]. In stateful logic, both operands and results are represented by the resistive states of memristors, enabling arbitrary Boolean computation through voltage-driven switching. While significant advances have been made in logic families, synthesis, and mapping techniques, practical deployment requires accurate circuit-level validation under realistic device conditions. A growing ecosystem of EDA tools supports individual stages of the LiM design flow, including synthesis and mapping [4]–[7], verification [8], [9], and higher-level IMC compilation [10]. However, these flows largely terminate at the logical abstraction level and provide little support for evaluating mapped algorithms under realistic device behavior. Since switching thresholds, resistance variability, and circuit topology directly influence the correctness and efficiency of stateful logic operations, circuit-level evaluation is essential for reliable LiM design. Recent frameworks such as MemSPICE [11] and ATOMIC [12] partially automate SPICE-based evaluation, but remain restricted to individual logic families and simplified execution models (e.g., single-row execution), with limited support for mixed-logic, flexible crossbar architectures, and comprehensive circuit-level analysis.

To bridge this gap, we present *ARCADE*, an open-source[1] EDA framework for highly automated circuit-level evaluation of memristive LiM. Its modular architecture supports multiple and mixed logic families, row- and column-oriented and SIMD execution, configurable crossbars and periphery, interchangeable device models, and different simulation backends. Additionally, the proposed interchangeable LiM representation (`.ilim`) provides a common interface across design stages, enabling integration with existing tools and automated, feedback-driven design flows.

The main contributions of this work are: 1) **Automated circuit-level evaluation:** An end-to-end workflow from mapped LiM algorithms and configuration to circuit and control generation, experiment orchestration, data analysis, visualization, and reporting. 2) **Modular and flexible architecture:** Unified support for heterogeneous logic families and execution modes, configurable crossbar architectures (e.g., 1R/1T1R) and periphery, interchangeable device models, and simulation backends. 3) **Interoperable LiM representation:** The proposed `.ilim` representation provides a common interface between synthesis, mapping, circuit-level evaluation, and future LiM tools. 4) **Comprehensive analysis and optimization:** Automated functional, energy, state-quality, and robustness analysis under device variability, demonstrated through energy optimization of State-of-the-Art (SoA) adders and reliability improvement of ISCAS'85 benchmarks.

II. Background and Related Work

Memristive Logic-in-Memory Operations: Resistive Random Access Memory (RRAM) is a leading memristive device technology for LiM, providing two non-volatile resistive states, Low Resistance State (LRS) and High Resistance State (HRS), through the formation and rupture of conductive filaments [13]–[15]. Several memristor-based computing paradigms have been proposed that differ in how logical information is represented and processed [16]. In contrast to approaches that encode inputs and/or outputs as voltages [17]–[19], stateful logic represents both inputs and outputs using resistive states of memristors, enabling LiM computation directly within the crossbar array. Representative stateful LiM

[1]The framework is available at https://github.com/Firomaeor/ARCADE

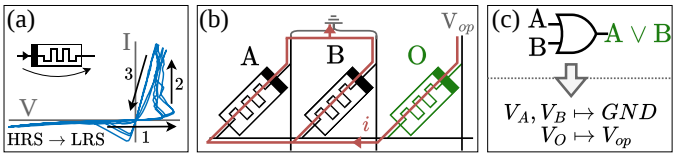


Fig. 1: LiM with memristors: (a) device I-V curve, (b) FELIX-OR primitive, and (c) logic synthesis and mapping of atomics.

families include IMPLY [20], MAGIC [21], FELIX [22], and TMSL [23]. Although these families differ in topology and operating principles, they all implement Boolean operations by applying carefully selected voltage stimuli that strongly depend on the underlying device characteristics. Figure 1 illustrates this hierarchy: (a) memristors switch between HRS and LRS, (b) enabling stateful logic primitives such as FELIX-OR, (c) which are composed into larger algorithms through synthesis and mapping. Since the operating voltages (and gate resistances for IMPLY and TMSL) depend strongly on the device characteristics and circuit topology, realistic circuit-level evaluation is essential for validating mapped LiM algorithms. This is commonly performed using compact memristor models ranging from efficient behavioral models such as VTEAM [24] to variability-aware models such as MemSim+ [25].

Related Tools: A growing body of work has addressed EDA methodologies for memristive LiM, including synthesis, mapping, verification, and compilation [3]. Existing synthesis tools primarily target MAGIC-based logic using NOR/NOT operations [4]–[7]. Complementary approaches provide mapping verification [8], [9] or higher-level IMC compilation [10]. In contrast, automated circuit-level evaluation of complete stateful LiM algorithms remains comparatively limited. MemSPICE [11] generates SPICE netlists and estimates the energy of single-row MAGIC mappings, while ATOMIC [12] enables circuit-level simulation of IMPLY algorithms in restricted topologies [26]. Existing tools therefore either stop at synthesis and mapping or restrict circuit-level evaluation to specific logic families and architectures. ARCADE fills this missing EDA layer by automating circuit construction, experiment orchestration, and analysis across heterogeneous LiM designs using interchangeable device models and simulation backends. The proposed `.ilim` representation additionally interfaces this layer with existing synthesis and mapping tools.

III. ARCADE Framework

ARCADE provides a highly automated circuit-level evaluation framework for mapped LiM algorithms. Given an algorithm and configuration, the framework autonomously performs circuit and control generation, experiment orchestration, data processing, analysis, visualization, and reporting. As illustrated in Figure 2, the framework comprises three compartments: i) automated circuit generation, ii) experiment orchestration for parameterized simulation campaigns, and iii) circuit-level analysis of functionality, energy, and robustness under device variability. ARCADE follows a modular architecture that separates individual components to be configured, exchanged, or extended without modifying the remaining workflow. ARCADE introduces an interchangeable LiM representation (`.ilim`) that decouples mapped algorithms from specific tools and simulation backends while providing a common interface to existing synthesis and mapping tools, enabling seamless cross-layer design flows (see Section IV).

A. Automated Circuit Generation

ARCADE automatically translates mapped LiM algorithms represented in `.ilim` into executable circuit models. Existing representations from tools such as SIMPLER [5] and ATOMIC [12] can additionally be imported through format conversion ①. Crossbar topology, device models, peripheral circuits, and simulation parameters are configured through external `.json` files ②. Based on these inputs, ARCADE automatically generates complete SPICE netlists ③, including crossbar arrays, voltage drivers, and auxiliary circuit elements (e.g., gate resistors), enabling reproducible circuit-level evaluation. Listing 1 illustrates a generated $n \times n$ crossbar netlist, where each memory cell is instantiated from a reusable cell object connecting row/column lines, selector controls, and the internal device state, which is exposed for subsequent analysis and state-quality evaluation (see Section III-C).

```
// Connect cell to: row, column, and switch signals
Xm_0_0 c0 r0 m_0_0_ctrl m_0_0_state sw_Mem <params>
Xm_0_1 c1 r0 m_0_1_ctrl m_0_1_state sw_Mem <params>
...
Xm_n_n cn rn m_n_n_ctrl m_n_n_state sw_Mem <params>
```

Listing 1: Generated crossbar netlist

To support different LiM families and hardware platforms, ARCADE employs a modular circuit-generation methodology in which reusable memory-cell, peripheral, and row/column modules are composed into configurable crossbar architectures ④. Family-specific components (e.g., gate resistors for IMPLY and TMSL) are inserted automatically during netlist generation. The framework supports passive (1R), transistor-assisted (1T1R), and arbitrary selector-based (1S1R) [27] crossbar organizations through abstract cell definitions that encapsulate both the device model and optional selector circuitry. This abstraction largely decouples ARCADE's circuit-generation methodology from the underlying device model and simulator backend, facilitating support for different device technologies. Supporting new device models or parameterizations only requires adapting the circuit templates, leaving the overall workflow unchanged. Listing 2 illustrates an exemplary 1T1R cell employed by ARCADE, combining a memristor with a controllable selector. The instantiated modules are subsequently configured by the experiment orchestrator for circuit-level simulation.

```
.SUBCKT sw_Mem col row ctrl state
Xm state col intern <Mem Model> <params>
S_sw_Mem intern row control_node 0 SW
V_sw_Mem ctrl 0 PWL file={pwl}
.ENDS sw_Mem
.model SW <Switch Model> <params>
```

Listing 2: 1T1R Cell Object

B. Experiment Orchestrator

The experiment orchestrator autonomously configures and executes parameterized circuit-level experiment campaigns for

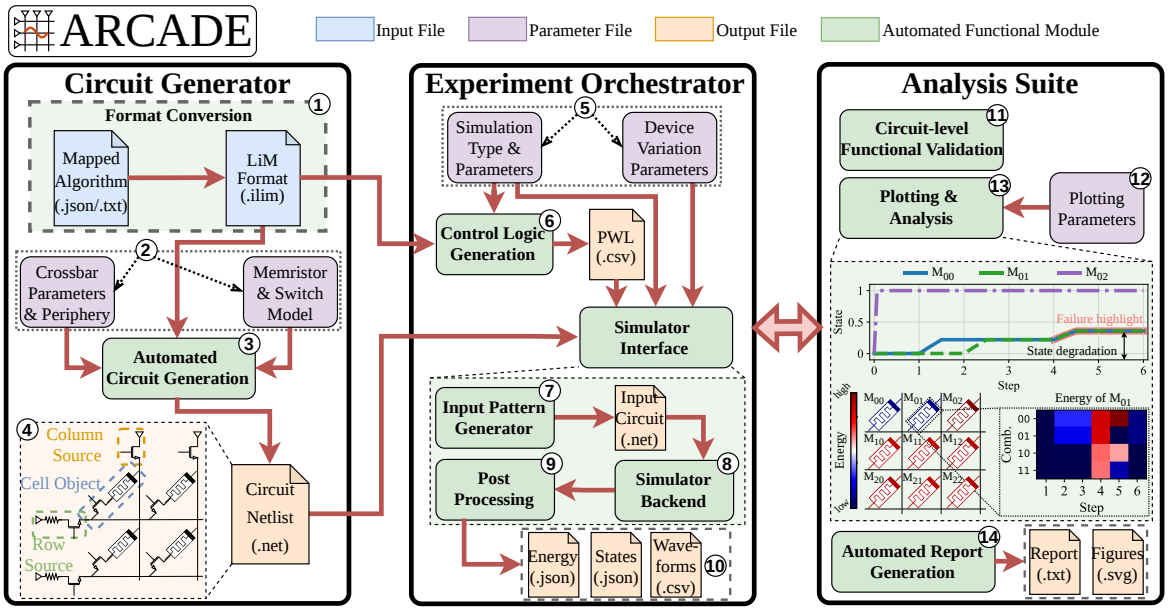


Fig. 2: ARCADE framework: automated circuit generation, experiment orchestration, and circuit-level analysis of memristive LiM operations.

mapped LiM algorithms. Since the required operating conditions depend strongly on the underlying device technology and logic family, operating voltages, timing parameters, peripheral controls, and device characteristics are specified through external configuration files ⑤, enabling the evaluation of both existing and user-defined Micro-Operations (μOPs). Device behavior can be modeled deterministically or statistically by sampling each parameter p according to $p \sim \mathcal{N}(\mu_p, \sigma_p^2)$, supporting Monte Carlo robustness analysis under configurable device variations. By decoupling algorithm execution from device-specific operating conditions, identical LiM algorithms can be evaluated across different technologies and variability assumptions without modifying the algorithm description. Each μOP is associated with a predefined template describing the required bias conditions for inputs, outputs, and auxiliary nodes. ARCADE constructs Piece-Wise Linear (PWL) control signals by scheduling these templates according to the operation sequence ⑥. This generic scheduling methodology supports all implemented logic families and execution modes, including row- and column-oriented and SIMD execution. It is also easily extensible for future LiM operations. Representative waveforms generated for a SIMPLE-mapped MAGIC half-adder [4] are shown in Figure 3, including operating voltages, driver enables, selector controls, common-mode voltages, and auxiliary control signals.

ARCADE supports exhaustive, Monte Carlo, and user-defined input orchestration campaigns ⑦, which can be combined with statistical parameter variations to assess circuit robustness. Numerical circuit simulation is delegated through a backend interface, currently supporting SPICE engines such as LTspice and Ngspice ⑧. Hence, practical simulation cost and scalability are primarily determined by the selected circuit solver and device model, while independent experiment runs can be parallelized by ARCADE. For each simulation run, metrics are automatically extracted and organized as third-

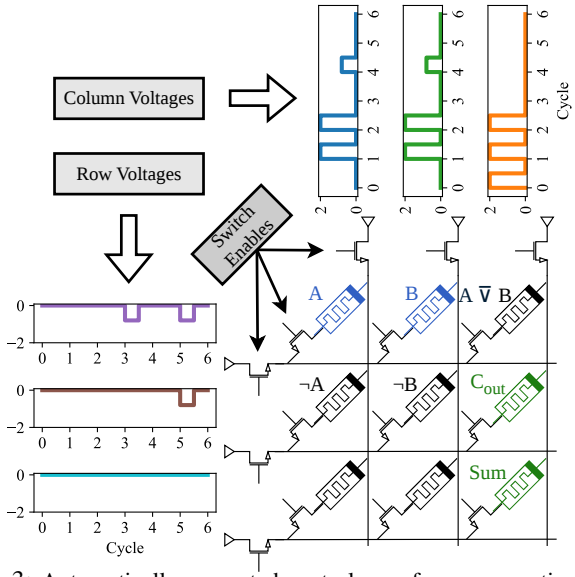


Fig. 3: Automatically generated control waveforms supporting row- and column-wise execution and SIMD operations.

order tensors $\mathcal{T}_{c,s,d}$ ⑨, storing the value of a given metric for device d at operation step s under input combination c. This unified representation enables fine-grained analysis across devices, execution steps, and input patterns, providing detailed insight into circuit behavior and algorithmic limitations. The energy associated with each tensor element is computed as

$$E(c,s,d) = \int_{t_s}^{t_s+\Delta s} V_{c,d}(t) I_{c,d}(t)\, dt, \tag{1}$$

while state quality is defined as

$$Q(c,s,d) = \phi(S^*_{c,s,d}, S_{c,s,d}), \tag{2}$$

where S^* and S denote the ideal and simulated device states, respectively. Throughout this work, $\phi = 1 - |S^* - S|$ maps state quality to the interval $[0,1]$, ranging from a bit-flip (0) to

the ideal state (1) [12], [28]. Beyond binary correctness, this metric captures degradation of intermediate memristor states that are reused in subsequent operations. This reflects their separation from the undefined switching region and provides a quantitative measure of computational reliability. All extracted metrics, including energy, state evolution, and quality, are stored for subsequent analysis (10), providing a reproducible basis for circuit characterization and optimization.

C. Analysis Suite

The analysis suite evaluates processed simulation results to assess functional correctness, state quality, and energy consumption, and automatically generates visualizations and reports for circuit-level characterization. Functional validation (11) verifies that a mapped LiM algorithm operates correctly under the selected device model and operating conditions (e.g., voltage levels and timing parameters). Correctness is evaluated using a quality threshold $\theta \in [0, 1)$, where common choices include $\theta = 0.5$ [25] and $\theta = 0.66$ [28], [29]. This criterion ensures that intermediate memristor states remain sufficiently separated from the undefined region before reuse in subsequent operations. A memristor d is considered valid if

$$\mathcal{V}(d) \iff \forall c, s : \; Q(c, s, d) \geq \theta. \tag{3}$$

A complete LiM algorithm is valid if all participating functionally relevant devices (inputs and outputs) satisfy this condition

$$\mathcal{V}(\mathcal{A}) \iff \forall d \in \mathcal{A} : \mathcal{V}(d). \tag{4}$$

Beyond binary pass/fail evaluation, ARCADE classifies failures according to their origin (intermediate states, final outputs, or critical devices), enabling targeted diagnosis of circuit-level failures. To facilitate interpretation, ARCADE automatically generates customizable visualizations (12) from arbitrary projections of $\mathcal{T}_{c,s,d}$. This enables analysis at the level of individual devices, execution steps, input combinations, or aggregated statistics. Figure 2 (13) illustrates representative visualizations supporting failure localization, root-cause tracing throughout the execution sequence, and energy analysis. Finally, configurations, processed simulation data, extracted metrics, generated visualizations, and analysis results are automatically compiled into a report (14), providing a reproducible basis for circuit characterization and optimization workflows.

IV. Integration

Interchangeable LiM Representation (.ilim): With ARCADE we introduce the interchangeable LiM representation (`.ilim`). It is a lightweight textual intermediate representation that provides a unified interface between synthesis, mapping, and circuit-level evaluation tools. Unlike existing representations tailored to individual logic families or execution models [5], [12], [30], `.ilim` describes mapped LiM algorithms independently of the underlying logic family, crossbar organization, and simulation backend. Each `.ilim` file consists of two parts: i) metadata describing the target crossbar configuration and logical input/output mappings, and ii) an ordered sequence of parameterized μOPs describing the mapped algorithm. The metadata captures the target crossbar configuration, logical I/O mapping, and auxiliary information required to reconstruct the mapped circuit for simulation. Each μOP is encoded as a single instruction and follows the general instruction format

`<MNEMONIC>[.T]<IN`$_1$`>...<IN`$_n$`><OUT>[<INDEX_SET>].`

It consists of a mnemonic identifying the logic family and Boolean primitive (including its input arity), an optional transposition modifier (`.T`) that encodes row-oriented execution, a list of input and output device indices, and an optional index set specifying parallel execution across multiple crossbar slices. For example, the instruction "`MNOR2 0 1 2 [0,1]`" invokes the column-oriented two-input MAGIC-`NOR` operation (`MNOR2`). The memristors at indices `0` and `1` are used as inputs and the result is stored in the memristor at index `2`. The index set `[0,1]` indicates that this operation is executed simultaneously on the crossbar slices (i.e., rows) '0' and '1'. Likewise, appending the optional transposition modifier, e.g., `MNOR2.T`, executes the same operation along rows instead of columns. This instruction-based representation naturally supports multiple and mixed logic families through different mnemonics, row- and column-oriented execution through transposed instructions, and SIMD execution through index sets. This generality of `.ilim` additionally allows for extensions to future logic families and execution paradigms. New μOPs can be incorporated simply by adding new mnemonics, while preserving the underlying instruction syntax and workflow. Listing 3 illustrates a representative `.ilim` description of a SIMPLE-generated MAGIC half-adder algorithm [4] that corresponds to Figure 3.

```
// Metadata
Description={SIMPLE Magic HalfAdder}
Source={https://doi.org/10.1109/ICCAD.2017.8203782}
Cycles={6}
Crossbar_size={(3, 3)}
Inputs={A=(0,0), B=(0,1)}
Outputs={Sum=(2,2), Cout=(1,2)}

// Micro-Operation Sequence
SET [2]  [0]
SET [0, 1, 2]  [1]
SET [0, 1, 2]  [2]
MNOT.T 0 1 [0, 1]
MNOR2 0 1 2 [0, 1]
MNOR2.T 0 1 2 [2]
```

Listing 3: Representative `.ilim` description.

LiM Design Flow: Figure 4 illustrates how ARCADE integrates into broader LiM design flows through the interchangeable `.ilim` representation. Mapped algorithms can be provided directly or converted from existing tools such as SIMPLER [5]. Given the algorithm and configuration, ARCADE autonomously performs circuit and control generation, experiment orchestration, and circuit-level analysis. The resulting feedback can then guide upstream mapping and parameter optimization. Generated control signals can be directly used to support experimental validation and device measurements can help recalibrate compact models. Together,

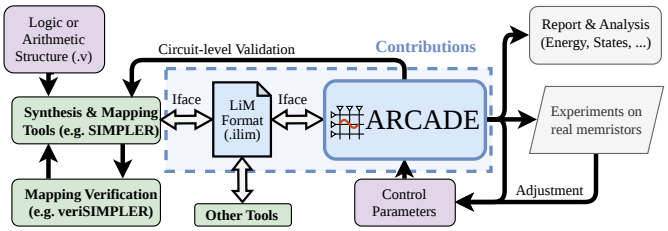


Fig. 4: Integrating ARCADE's automated circuit-level workflow with the broader LiM design ecosystem through `.ilim`.

these interfaces enable automated cross-layer design loops from algorithm development to hardware evaluation.

V. Experimental Evaluation

A. Circuit-Level Optimization Workflows

Beyond circuit-level evaluation, ARCADE enables feedback-driven optimization workflows by coupling circuit-level simulation with automated parameter exploration. As a representative case study, we optimize seven SoA LiM adder implementations spanning IMPLY [29], [31], [32], MAGIC [5], [33], [34], and FELIX [22] logic families. Two objectives are considered independently: i) minimizing energy consumption and ii) maximizing state quality while preserving functional correctness. Since neither the objective function nor the boundary of the functionally valid operating region is available analytically, ARCADE iteratively evaluates candidate operating points using its circuit-simulation backend (Algorithm 1). Gradients are estimated numerically, while each parameter update is restricted to the feasible region $\mathcal{F}$ identified through functional validation. IMPLY and MAGIC adders are evaluated using the VTEAM model [24] fitted to discrete Knowm memristors [29], [32], whereas FELIX is evaluated using the MemSim+ SDC model [25]. Initial operating points were selected heuristically to satisfy functional correctness. All experiments use a cycle time of $30\,\mu$s and a quality threshold of $\theta = 0.66$ (similar to [12]). The optimization results are summarized in Table I and reveal distinct characteristics across the evaluated logic families. The initial operating points for the IMPLY adders already lie close to locally optimal conditions. Consequently, energy optimization consistently reduces energy by approximately 10–11%, while quality improvements remain modest. In contrast, MAGIC implementations exhibit a strong sensitivity to the initialization voltage [35], allowing ARCADE to reduce energy consumption by up to 95.07% by lowering the `SET` voltage closer to the device switching threshold. For FELIX, gains remain $\approx 2\%$ because the selected configuration yields a narrow feasible operating region, leaving little model-specific optimization headroom. Since FELIX uses a different device model, its absolute energy values are not directly comparable to the other families.

Figure 5 decomposes the optimized energy consumption into individual μOPs. IMPLY adders are dominated by `IMPL` operations, whereas approximately half of the energy consumed by MAGIC and FELIX originates from repeated `SET` operations, explaining the large optimization potential of the

Algorithm 1: Boundary-Constrained Gradient Descent

Input: Feasible $\mathbf{x}^{(0)} \in \mathcal{F} \subseteq \mathbb{R}^d$, objective $f : \mathbb{R}^d \to \mathbb{R}$
Parameters: $\eta, \delta, \epsilon, \beta, \gamma, K_{\max}$ **Output:** $\mathbf{x}^*$

$k \leftarrow 0$;
while $k < K_{\max}$ **do**
  $(\eta_k, \delta_k) \leftarrow (\eta, \delta)(1+\beta k)^{-1}$ `//Reduce step size`
  $\mathbf{g} \leftarrow \nabla_{\mathcal{F}} f(\mathbf{x}^{(k)}; \delta_k)$ `//Feasible gradient`
  $\lambda^* \leftarrow \gamma\,\textsc{MaxFeasibleStep}\left(\mathbf{x}^{(k)}, -\frac{\mathbf{g}}{\|\mathbf{g}\|}, \eta_k\right)$
  $\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} - \lambda^* \frac{\mathbf{g}}{\|\mathbf{g}\|}$ `//Update solution`
  if $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| < \epsilon$ **then**
    return $\mathbf{x}^{(k+1)}$ `//Converged`
  end
  $k \leftarrow k + 1$
end
return $\mathbf{x}^{(k)}$

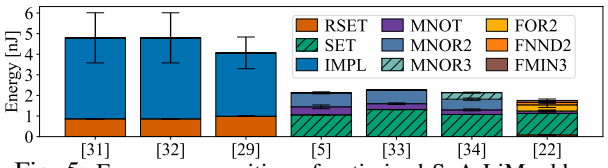


Fig. 5: Energy composition of optimized SoA LiM adders.

initialization voltage. ARCADE's analysis capabilities provide actionable insight into the dominant energy contributors of mapped LiM algorithms and guide targeted circuit-level optimization. Overall, this case study demonstrates that ARCADE not only automates circuit-level evaluation of LiM circuits. It also enables circuit-level optimization workflows by integrating simulation, functional validation, and energy analysis within a unified EDA framework.

B. Circuit-Level Reliability Analysis

Using established device and circuit models, we evaluate representative ISCAS'85 benchmarks [36] to demonstrate ARCADE's ability to automatically localize circuit-level failure mechanisms and assess mitigation strategies at the mapped-algorithm level. Following prior work [5], [11], the circuits are synthesized using SIMPLER and mapped to single-row 1T1R crossbars with adaptive device counts [5]. Simulations employ a device model fitted to discrete ECM memristors [25] using heuristically selected operating voltages $(V_{\text{set}}, V_{\text{nor}}, V_{\text{not}}) = (2, 0.8, 0.8)\,\text{V}$ and a cycle time of $10\,\mu$s. Each benchmark is evaluated using 20 randomized input initializations. Table II summarizes the results, reporting average state quality and overall functional success. While the smaller benchmarks c17 and c432 achieve perfect functional correctness, larger circuits (c499, c880, c1355, and c1908) consistently fail despite correct logic-level mappings. ARCADE traces these failures to the initialization phase, where concurrent `SET` operations reduce the effective switching voltage due to non-ideal selector and driver switches, degrading the initialized memristor states before computation begins. Guided by this analysis, we swept the initialization parallelism and found that limiting concurrent `SET` operations to 32, the highest tested degree preserving functional correctness, restores correct operation for all evaluated circuits. As shown in Table II, this mitigation incurs only

TABLE I: Circuit-level results from Algorithm 1, independently optimizing energy or state quality, applied to SoA LiM adders.

Algorithm	Metrics		Original Results		Energy Optimized			Quality Optimized		
	Cycles	Crossbar	Energy [nJ]	Quality ↑	Energy [nJ]	Impr.	Quality ↑	Energy [nJ]	Quality ↑	Impr.
IMPLY [31]†	22	1 × 5	5.3091±1.1039	0.9239±0.0248	4.7959±1.1520	9.67%	0.8937±0.0324	5.8975±0.9848	0.9416±0.0183	1.92%
IMPLY [32]†	20	1 × 6	5.3083±1.1074	0.9269±0.0226	4.7949±1.1552	9.67%	0.9008±0.0297	5.7689±0.9560	0.9498±0.0154	2.47%
IMPLY [29]†	18	1 × 6	4.5828±0.7150	0.9237±0.0103	4.0663±0.7361	11.27%	0.8894±0.0160	5.3006±0.6562	0.9427±0.0096	5.99%
MAGIC [5]	13	1 × 15	32.4732±0.1396	1.0000±0.0000	2.1266±0.1144	93.45%	0.9614±0.0079	32.4732±0.1396	1.0000±0.0000	-
MAGIC [33]	12	4 × 5	45.9331±0.0776	1.0000±0.0000	2.2653±0.0447	95.07%	0.9589±0.0041	45.9331±0.0776	1.0000±0.0000	-
MAGIC $[34]_{v2}$	13	4 × 9	32.9874±0.1322	1.0000±0.0000	2.1386±0.0822	93.52%	0.9878±0.0019	32.9190±0.1310	1.0000±0.0000	-
FELIX [22]*	7	1 × 6	1.7944±0.1783	0.9748±0.0075	1.7576±0.1662	2.05%	0.9728±0.0061	1.7733±0.1698	0.9771±0.0038	0.24%

† Energy differences ($< 1\%$) from [29], [31], [32] stem from tighter rise and fall times, and peripheral assumptions differing than in [12].
* Evaluated using the MemSim+ SDC model [25], as no valid operating point was found for the fitted Knowm VTEAM model.

TABLE II: Circuit-level reliability analysis of ISCAS'85 benchmarks before and after limiting concurrent `SET` operations.

Circuit	PI/PO	Memristors	SIMPLER Output				Limited to 32 Concurrent `SET` Operations			
			Cycles	Energy [μJ]†	Quality	Success	Cycles	Energy [μJ]†	Quality	Success
c17	5/2	18	14	2.7794±0.0026	1.0000±0.0000	✓(20/20)	14	2.7794±0.0026	1.0000±0.0000	✓(20/20)
c432	36/7	62	236	49.2249±0.0134	1.0000±0.0000	✓(20/20)	236	49.2249±0.0134	1.0000±0.0000	✓(20/20)
c499	41/32	110	624	116.8417±0.0640	0.9278±0.0238	✗(0/20)	624+5	122.6515±0.0095	1.0000±0.0000	✓(20/20)
c880	60/26	142	518	92.5943±0.0554	0.9544±0.0180	✗(0/20)	518+17	103.2755±0.0390	1.0000±0.0000	✓(20/20)
c1355	41/32	111	595	110.5618±0.0600	0.9339±0.0190	✗(0/20)	595+5	116.5636±0.0121	1.0000±0.0000	✓(20/20)
c1908	33/25	122	589	102.1536±0.2040	0.8887±0.0171	✗(0/20)	589+9	117.0882±0.0255	1.0000±0.0000	✓(20/20)

† The high energy consumption results from the characteristics of the discrete printed ECM memristors [25] (LRS= 174Ω HRS= 1933kΩ).

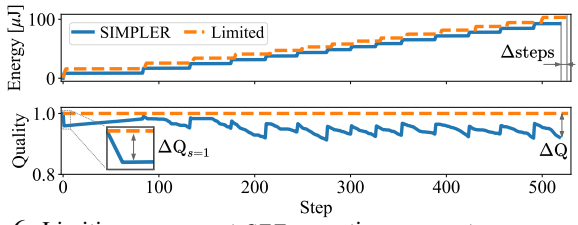


Fig. 6: Limiting concurrent `SET` operations prevents error propagation and restores correct execution.

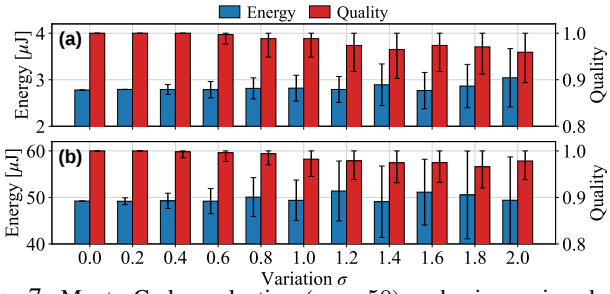


Fig. 7: Monte Carlo evaluation ($n = 50$) under increasing device variability for (a) c17 and (b) c432.

a modest increase in latency and energy while eliminating the observed failures. Figure 6 illustrates the corresponding quality evolution for c880, showing how the initial degradation propagates through the computation and is eliminated by the modified scheduling. These results demonstrate that logic-level mappings may become unreliable under realistic device conditions and highlight the importance of circuit-level validation during LiM design.

ARCADE further supports variability-aware robustness analysis through Monte Carlo simulation. To isolate variability effects from the initialization failures observed in larger mappings, we evaluate the nominally correct c17 and c432 using 50 Monte Carlo simulations for each of 32 sampled input patterns. Following [12], Gaussian variations of the resistive states (HRS, LRS) and switching thresholds (v_{on}, v_{off}) are introduced, using variation ranges of up to 20% and 2%, respectively. As shown in Figure 7, increasing device variability has only a minor effect on average energy consumption, whereas state quality degrades progressively. Although the evaluated circuits remain functionally correct under moderate variation, the progressive degradation in state quality illustrates how ARCADE can quantify robustness margins under increasing device variability. Together, these experiments demonstrate that ARCADE extends circuit-level evaluation beyond functional validation by identifying reliability bottlenecks, evaluating mitigation strategies, and quantifying robustness under realistic device variability.

TABLE III: Comparison of ARCADE with SoA frameworks.

Property	memSPICE (2024) [11]	ATOMIC (2025) [12]	**ARCADE (Proposed)**
Automated circuit generation	✓	✗	✓
Crossbar configuration (1R/1T1R)	✗	✗	✓
Flexible crossbar periphery	✗	✗	✓
Multi- & mixed-logic support	✗	✗	✓
Row-wise operations	✓	✓	✓
Column-wise operations	✗	✗	✓
Simulating SIMD operations	✗	✗	✓
Flexible device models	✓	✓	✓
Functional validation	✗	✓	✓
Energy analysis	✓	✗	✓
State quality analysis	✗	✓	✓
Automated report	NA	✗	✓
Design flow integration	✓+	✗	✓
Open source	✗	✓	✓

+ Only compatible with MAGIC mappings from SIMPLER [4].

VI. Conclusion

This paper presented ARCADE, a highly automated and modular EDA framework for circuit-level evaluation of memristive LiM. As summarized in Table III, existing frameworks target specific logic families and execution models. ARCADE unifies configurable circuit generation, experiment orchestration, device and architecture flexibility, and comprehensive circuit-level analysis within a single workflow. The proposed `.ilim` representation additionally provides a common interface to synthesis and mapping tools, enabling automated cross-layer design flows. Case studies demonstrated circuit-level optimization, identification and mitigation of reliability bottlenecks, and robustness analysis under device variability. Together, ARCADE provides a general circuit-level evaluation layer for the automated and reliable design exploration of future memristive LiM systems.

ACKNOWLEDGMENTS

This work was supported by the German Research Foundation (DFG) under Projects 550835853 and 569133238, and Hector Stiftung.

REFERENCES

[1] O. Mutlu *et al.* A modern primer on processing in memory, 2025.

[2] T. Neuner *et al.* A comparative study of digital memristor-based processing-in-memory from a device and reliability perspective. *Advanced Electronic Materials*, 12(1):e00348, 2026.

[3] C. K. Jha *et al.* Devices, synthesis, circuits, and verification for resistive ram-based digital computing-in-memory. In *2025 First International Conference on Intelligent Computing and Systems at the Edge (ICEdge)*, volume 1, pp. 1–6, 2025.

[4] R. Ben Hur *et al.* Simple magic: Synthesis and in-memory mapping of logic execution for memristor-aided logic. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 225–232, 2017.

[5] R. Ben-Hur *et al.* Simpler magic: Synthesis and mapping of in-memory logic executed in a single row to improve throughput. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(10):2434–2447, 2020.

[6] F. Wang *et al.* Star: Synthesis of stateful logic in rram targeting high area utilization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(5):864–877, 2021.

[7] X. Qian *et al.* Masim: An energy-efficient multi-array scheduler for simd logic-in-memory architectures. In *2025 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, 2025.

[8] C. K. Jha *et al.* verisimpler: An automated formal verification methodology for simpler magic design style based in-memory computing. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 71(9):4169–4179, 2024.

[9] C. K. Jha *et al.* verisim: Formal verification of spice netlists for magic-based logic-in-memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 45(2):845–854, 2026.

[10] A. A. Khan *et al.* Cinm (cinnamon): A compilation infrastructure for heterogeneous compute in-memory and compute near-memory paradigms. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 4*, pp. 31–46, 2025.

[11] S. Singh *et al.* Memspice: Automated simulation and energy estimation framework for magic-based logic-in-memory. In *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 282–287, 2024.

[12] F. Seiler *et al.* Atomic: Automatic tool for memristive imply based circuit-level simulation and validation. In *2025 21st International Conference on Synthesis, Modeling, Analysis and Simulation Methods, and Applications to Circuits Design (SMACD)*, pp. 1–4, 2025.

[13] L. O. Chua. Memristor—the missing circuit element. *IEEE Transactions on Circuit Theory*, CT-18(5):507–519, 1971.

[14] D. B. Strukov *et al.* The missing memristor found. *Nature*, 453:80–83, 2008.

[15] E. Lehtonen and M. Laiho. Stateful implication logic with memristors. In *2009 IEEE/ACM International Symposium on Nanoscale Architectures*, 2009.

[16] N. Du *et al.* Mixed-mode in-memory computing: towards high-performance logic processing in a memristive crossbar array. *Communications Engineering*, 4(1):163, 2025.

[17] S. Kvatinsky *et al.* Mrl — memristor ratioed logic. In *2012 13th International Workshop on Cellular Nanoscale Networks and their Applications*, pp. 1–6, 2012.

[18] Y. Song *et al.* Reconfigurable and efficient implementation of 16 boolean logics and full-adder functions with memristor crossbar for beyond von neumann in-memory computing. *Advanced Science*, 9(15):2200036, 2022.

[19] I. Vourkas and G. C. Sirakoulis. A novel design and modeling paradigm for memristor-based crossbar circuits. *IEEE Transactions on Nanotechnology*, 11(6):1151–1159, 2012.

[20] J. Borghetti *et al.* 'Memristive' switches enable 'stateful' logic operations via material implication. *Nature*, 464:873–876, 2010.

[21] S. Kvatinsky *et al.* Magic—memristor-aided logic. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 61(11):895–899, 2014.

[22] S. Gupta *et al.* Felix: fast and energy-efficient logic in memory. In *Proceedings of the International Conference on Computer-Aided Design*, ICCAD '18. ACM, 2018.

[23] P. Huang *et al.* Reconfigurable nonvolatile logic operations in resistance switching crossbar array for large-scale circuits. *Advanced Materials*, 28(44):9758–9764, 2016.

[24] S. Kvatinsky *et al.* Vteam: A general model for voltage-controlled memristors. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 62(8):786–790, 2015.

[25] G. Gulafshan *et al.* Realistic behavioral model for rerams capturing non-idealities. *Communications Materials*, 6(1):1–13, 2025.

[26] F. Seiler and N. TaheriNejad. Accelerated image processing through imply-based nocarry approximated adders. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 71(11):5141–5154, 2024.

[27] M. S. Q. Truong *et al.* Adapting the racer architecture to integrate improved in-reram logic primitives. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 12(2):393–407, 2022.

[28] C. Simonides *et al.* Approximated 2-bit adders for parallel in-memristor computing with a novel sum-of-product architecture. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, 10:135–143, 2024.

[29] F. Seiler *et al.* An efficient robust serial imply-based in-memristor adder. In *2025 Cross-Disciplinary Conference on Memory-Centric Computing (CCMCC)*, pp. 1–7, 2025.

[30] S. Nabipour *et al.* Multi-input magic synthesis and verification for in-memory computing design. In *2025 IEEE 55th International Symposium on Multiple-Valued Logic (ISMVL)*, pp. 178–183, 2025.

[31] S. G. Rohani and N. TaheriNejad. An improved algorithm for imply logic based memristive full-adder. In *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–4, 2017.

[32] F. Seiler and N. TaheriNejad. An improved serial imply adder algorithm for efficient neural network applications. In *2025 IEEE 16th Latin America Symposium on Circuits and Systems (LASCAS)*, volume 1, pp. 1–5, 2025.

[33] N. Kaushik *et al.* High-speed magic-based exact and approximate adders for in-memory computing. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 16(2):312–326, 2026.

[34] N. Talati *et al.* Logic design within memristive memories using memristor-aided logic (MAGIC). *IEEE Transactions on Nanotechnology*, 15(4):635–650, 2016.

[35] S. Singh *et al.* Should we even optimize for execution energy? rethinking mapping for magic design style. *IEEE Embedded Systems Letters*, 15(4):230–233, 2023.

[36] F. Brglez and H. Fujiwara. A neutral netlist of 10 combinational circuits and a target translator in fortran. In *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, 06 1985.