ChipCraftBrain: 98.7% Pass@1 on VerilogEval via Adaptive Multi-Agent RTL Generation

ChipCraftX
San Francisco, CA
info@chipcraftx.io

***Abstract*—**
Large Language Models (LLMs) have shown promise for generating Register Transfer Level (RTL) code from natural language specifications, yet single-shot generation achieves only 60–65% functional correctness on standard benchmarks such as VerilogEval. Existing multi-agent approaches like MAGE improve pass rates to 95.9% but lack synthesis awareness and incur high API costs ($0.06/problem with 20 parallel LLM calls).

We present ChipCraftBrain, a novel framework combining hybrid symbolic-neural reasoning with adaptive multi-agent orchestration for automated RTL generation. Four key innovations drive the system: (1) Hybrid symbolic-neural architecture that solves K-map and truth table problems algorithmically (zero LLM cost, perfect accuracy) while using specialized LLM agents for general RTL generation; (2) Adaptive multi-agent orchestration with a learned policy that selects among specialized agents and configures generation parameters based on problem characteristics and iteration dynamics; (3) knowledge-augmented generation via a curated domain knowledge base with focus-aware retrieval; and (4) hierarchical specification decomposition that automatically breaks complex multi-component designs into dependency-ordered sub-modules with interface synchronization.

On VerilogEval-Human, ChipCraftBrain achieves 98.7% pass@1 (154/156), surpassing all published systems including ChipAgents (97.4%) and MAGE (95.9%), at approximately 1/6th the API cost. The hybrid architecture demonstrates that combining symbolic reasoning with learned orchestration enables both higher accuracy and lower cost than pure neural approaches.

***Index Terms*—RTL generation, hardware design automation, large language models, reinforcement learning, multi-agent systems, knowledge-augmented generation, Verilog, EDA**

I. Introduction

A. The RTL Generation Challenge

The semiconductor industry faces a persistent bottleneck: the demand for custom chip designs far exceeds the supply of skilled RTL engineers. As system-on-chip (SoC) complexity grows (modern designs contain billions of transistors across dozens of IP blocks), the time and expertise required for front-end RTL design has become a critical constraint on innovation.

Recent advances in Large Language Models (LLMs) have opened a promising avenue: generating synthesizable Verilog or SystemVerilog code directly from natural language specifications. However, the gap between *generating* code and *generating correct, synthesizable* code remains substantial. Single-shot generation with state-of-the-art models like GPT-4 achieves only 60–65% pass rates on the VerilogEval benchmark [1], a figure far below the reliability threshold needed for production use.

B. Limitations of Current Approaches

Several lines of work have attempted to close this gap, each with significant limitations:

Single-shot generation approaches [2], [3], [4] fine-tune LLMs on Verilog corpora but provide no feedback mechanism, resulting in pass rates of 45–78%.

Iterative refinement systems [5] add error-feedback loops but use fixed retry strategies that do not adapt to specific error patterns or design complexity.

Multi-agent systems like MAGE [6] achieve 95.9% on VerilogEval-Human but lack synthesis awareness in candidate scoring, provide primitive debug feedback, and incur high cost (20 parallel LLM calls per problem at $0.06/problem).

Workflow search approaches like VFlow [7] use Monte Carlo Tree Search to discover optimal agentic workflows (83.6% pass@1), but the discovered workflow is static and does not adapt per-problem.

C. Our Contributions

We present ChipCraftBrain, a framework that addresses these limitations through four key contributions:

1) **Adaptive Multi-Agent Orchestration:** Specialized LLM agents coordinated by a learned policy that selects agents and configures generation parameters based on problem characteristics and iteration dynamics (Section III).
2) **Hybrid Symbolic-Neural Architecture:** Algorithmic solvers for deterministic problem classes combined with knowledge-augmented LLM generation (Section III).
3) **Hierarchical Specification Decomposition:** Automatic decomposition of complex multi-component designs into dependency-ordered sub-modules with cross-module port synchronization (Section III).
4) **Validation-First Pipeline:** A three-stage verification system with structured error feedback and iterative refinement (Section IV).

D. Paper Organization

The remainder of this paper is organized as follows. Section II surveys related work. Section III presents the system architecture. Section IV details the validation pipeline. Section V defines the benchmark framework. Section VI reports experimental results. Section VII analyzes findings and limitations. Section VIII outlines future work. Section IX concludes.

II. Related Work

A. LLM-Based RTL Generation

The application of LLMs to hardware description language generation has progressed rapidly. VeriGen [2] demonstrated that models fine-tuned on Verilog corpora can generate syntactically valid code, though functional correctness remained limited. CodeV [3] improved on this with multi-level code summaries for training, achieving 77.6% on VerilogEval-Machine and 53.2% on VerilogEval-Human using a 33B parameter model. RTLCoder [4] explored instruction-tuned models for RTL, reaching 45–50%. ChipChat [8] applied GPT-4 in conversational chip design workflows.

These approaches share a fundamental limitation: without a feedback loop, errors in the generated code cannot be detected or corrected, capping practical accuracy.

B. Multi-Agent Systems for RTL

MAGE [6] introduced a four-agent architecture (RTL generator, testbench generator, judge, debugger) that achieves 95.9% on VerilogEval-Human v2. Three innovations drive its success: (a) high-temperature sampling ($T = 0.85$) with 20 candidates, (b) Verilog-state checkpoint debugging with textual waveform windows, and (c) multi-agent task separation that avoids context-switching confusion between synthesizable RTL and non-synthesizable testbenches. Their ablation shows the progression: vanilla LLM 72.4% $\rightarrow$ single-agent 83.9% $\rightarrow$ multi-agent 93.6%.

However, MAGE's scoring function considers only functional correctness (normalized mismatch count), ignoring synthesis quality metrics. Its debug feedback is raw signal dumps without structural analysis. And generating 20 candidates with a frontier LLM incurs substantial cost ($\sim$\$0.06 per problem).

ChipAgents [9] is a commercial system reporting 97.4% on VerilogEval-v2 but has not published its methodology or evaluation protocol. It operates as a cloud-only service, precluding local deployment or academic reproduction. As this figure is self-reported without peer review, direct comparison should be interpreted with caution.

C. Automated Workflow Discovery

VFlow [7] applies Monte Carlo Tree Search (MCTS) to discover optimal agentic workflows for Verilog generation, achieving 83.6% pass@1, a 6.1% improvement over the previous best prompting strategy. The discovered workflow evolved into a five-step process: problem analysis $\rightarrow$ multiple generation $\rightarrow$ ensemble integration $\rightarrow$ testing $\rightarrow$ targeted refinement. Notably, DeepSeek-V3 with VFlow achieved 141.2% of GPT-4o's baseline performance at 13% of the API cost. However, VFlow's discovered workflow is static and does not adapt to individual problem characteristics.

D. RL for Code Generation

Reinforcement learning has been applied to code generation in the software domain. AlphaCode [10] uses RL for competitive programming. CodeRL [11] applies actor-critic methods for program synthesis with execution feedback. However, no prior work applies RL to orchestrate multi-agent *hardware* code generation, where the action space includes agent selection, temperature control, and knowledge retrieval strategy, a fundamentally different problem due to the synthesis and simulation constraints unique to RTL.

E. Hardware Design Automation

Traditional EDA tools from Cadence (Cerebrus) and Synopsys (DSO.ai) apply ML to *backend* physical design optimization (place-and-route, PPA) [12], [13]. These are complementary to front-end RTL generation and represent integration partners rather than competitors. The OpenLane 2 [14] flow with SKY130 PDK enables open-source RTL-to-GDSII synthesis, creating an opportunity for end-to-end AI-driven chip design pipelines.

III. System Architecture

A. Overview

ChipCraftBrain consists of six interconnected subsystems: (1) a *multi-agent LLM system* with specialized agents for different generation and debugging tasks, (2) a *hybrid symbolic-neural system* providing algorithmic solutions for deterministic problem classes, (3) an *adaptive orchestrator* that selects agents and configures generation parameters based on problem characteristics, (4) a *knowledge retrieval system* providing domain-specific context via RAG, (5) a *hierarchical decomposition engine* for complex designs, and (6) a *validation pipeline* providing structured feedback.

The generation pipeline employs a three-tier selection strategy: (1) algorithmic solving for problems admitting deterministic solutions (zero LLM cost), (2) rule-based routing for specialized problem types, and (3) learned orchestration among general agents for other RTL tasks.

B. Multi-Agent LLM System

We employ multiple specialized agents, each with distinct system prompts, model configurations, and retrieval strategies. Following the key insight from MAGE [6], each agent operates with an independent context window: mixing RTL generation (synthesizable code) and testbench creation (non-synthesizable code) in the same conversation degrades both tasks. Agents are grouped into two tiers:

- **General agents** for RTL generation, selected by the adaptive orchestration policy. These span a range of capability and cost, from fast lightweight models for quick iterations to more capable models for complex analysis and debugging.
- **Specialized agents** activated via rule-based heuristics for specific problem types such as timing waveform analysis and testbench generation.

C. Hybrid Symbolic-Neural Architecture

Before invoking LLMs, the system applies symbolic reasoning for problem classes that admit deterministic solutions. Problems requesting Karnaugh map minimization or truth

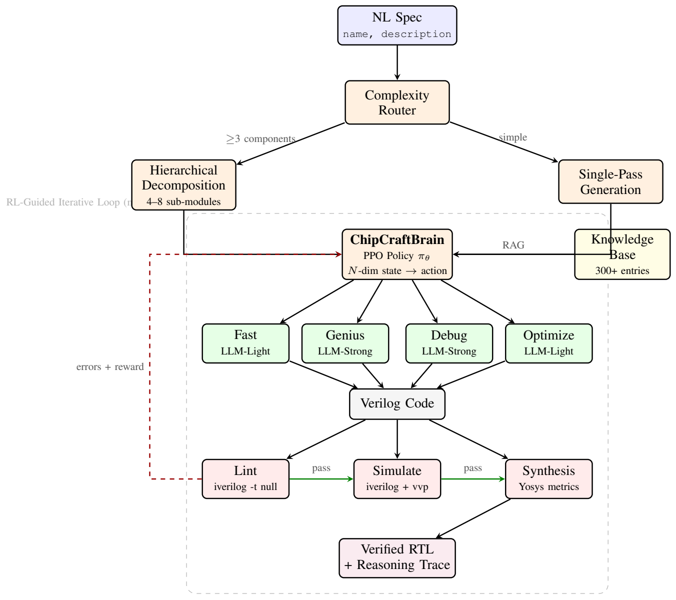

Fig. 1: CHIPCRAFTBRAIN system overview. Natural language specifications are routed through complexity analysis to either hierarchical decomposition or single-pass generation. The adaptive policy selects agents, temperatures, and RAG strategies. Generated Verilog passes through a three-stage validation pipeline; structured error feedback drives iterative refinement.

table implementation are solved algorithmically using Quine-McCluskey minimization [15], [16], achieving perfect accuracy with zero API cost. Timing waveform problems are routed to a dedicated agent with a structured multi-step reasoning methodology that extracts signal transitions, identifies pattern types, and derives logic from observed behavior.

D. Adaptive Orchestration

For general RTL generation tasks (after algorithmic and rule-based routing), CHIPCRAFTBRAIN employs an adaptive policy to select among specialized agents and configure generation parameters. The policy observes a state representation encoding problem characteristics, iteration progress, error analysis, code metrics, generation history, and knowledge retrieval quality. It outputs a hybrid action selecting agent type, retrieval strategy, sampling temperature, and token budget.

The policy is trained via Proximal Policy Optimization (PPO) [17] with a multi-component reward balancing correctness, efficiency, and code quality. This learned orchestration deploys more capable models only when problem difficulty warrants, while using efficient models for straightforward iterations.

E. Knowledge-Augmented Generation

CHIPCRAFTBRAIN maintains a curated knowledge base of over 300 entries spanning RTL design patterns, architecture templates, and optimization strategies. The retrieval system employs focus-aware strategies that adapt to the generation context, ranging from comprehensive search for complex specifications to error-focused search emphasizing debugging hints. Retrieved entries are scored by keyword relevance and injected into the agent's prompt.

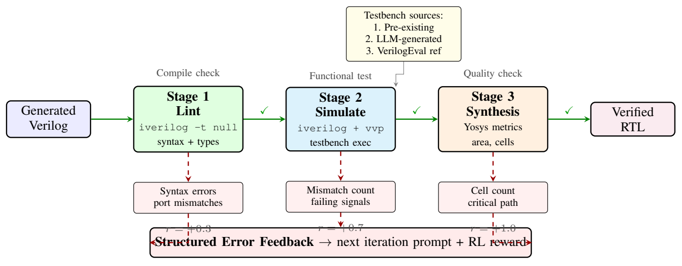

Fig. 2: Three-stage validation pipeline. Each stage provides structured error feedback that is (1) injected into the next LLM prompt as error context and (2) converted to a scalar RL reward. Stage progression (lint → simulate → synthesis) provides increasingly fine-grained validation. The 3.4× improvement from iterative refinement confirms that validation feedback is as important as generation quality.

F. Hierarchical Specification Decomposition

For complex specifications containing multiple component keywords, CHIPCRAFTBRAIN automatically decomposes the design into sub-modules. An LLM produces 4–8 sub-module specifications with interfaces and dependencies in structured JSON. Sub-modules are topologically sorted and generated in dependency order, with prior module headers passed as context to ensure port compatibility. This transforms the intractable problem of generating a complete SoC in a single LLM call into a series of tractable single-module generations with interface constraints.

IV. VALIDATION PIPELINE

The validation-first philosophy is central to CHIPCRAFTBRAIN. Rather than optimizing for single-shot generation quality alone, we invest in comprehensive validation and structured error feedback that enables iterative refinement.

A. Three-Stage Verification

Each generated module passes through three validation stages:

Stage 1: Compile (Icarus Verilog lint). The module is compiled with `iverilog` to check syntax, port declarations, signal types, and basic semantic correctness. This catches approximately 80% of errors before simulation.

Stage 2: Simulate (Icarus Verilog + VVP). If a testbench is available (pre-existing, LLM-generated, or from VERILOGEVAL reference), the compiled design is simulated. The system detects multiple pass/fail formats: `STATUS: PASS`, count-based markers (`N/N tests passed`), and VERILOGEVAL's mismatch format (`Mismatches: 0 in N samples`).

Stage 3: Synthesis-Ready Check. The module is synthesized with Yosys to extract synthesis metrics (cell count, wire count, critical path) and verify that the design is not merely functionally correct but also synthesis-clean and free of unintended latches or combinational loops.

B. Structured Error Feedback

Unlike systems that pass raw error messages to the debug agent, CHIPCRAFTBRAIN structures error feedback:

- **Error categorization**: Compile errors are classified (syntax, port mismatch, width mismatch, undeclared signal, etc.)
- **Category-aware fix hints**: Per-design-category hint databases map error patterns to likely fixes (e.g., for memory designs: "Replace `logic` with `reg`; use `integer i; for (i=0; ...)` instead of `for (int i=0; ...)`")
- **Source context**: 5 lines around each error point are extracted for targeted debugging
- **Error trend analysis**: The state vector tracks whether errors are increasing, decreasing, or changing type across iterations

C. Testbench Generation and Adaptation

When no pre-existing testbench is available, the Testbench agent generates one with specific requirements: proper reset sequence, 10ns clock period, 5+ test scenarios, and standardized status markers. The `TestbenchAdapter` wraps generated modules for compatibility with VERILOGEVAL's `TopModule` interface, enabling reference comparison against golden `RefModule` implementations.

V. BENCHMARK FRAMEWORK

We evaluate CHIPCRAFTBRAIN on VERILOGEVAL-Human [1], the standard benchmark for LLM-based RTL generation. The dataset contains 156 problems with English-language specifications and reference implementations, covering combinational logic, sequential circuits, FSMs, counters, and memory elements.

A. Evaluation Metrics

Our primary metric is **pass@1**: the fraction of problems solved on the first generation attempt (with iterative refinement counting as a single attempt). We also report:

- **Iterations to success**: Average iterations needed
- **Cost per problem**: Estimated API cost (USD)
- **Simulation pass rate**: Functional correctness via reference module comparison

B. Baselines

We compare against published results from CodeV [3] (53.2% human), MAGE [6] (95.9% human), ChipAgents [9] (97.4% human), VFlow [7] (83.6%), and single-shot GPT-4 ($\sim$60–65%).

VI. EXPERIMENTS AND RESULTS

A. Experimental Setup

Infrastructure: CHIPCRAFTBRAIN uses cloud LLM APIs for generation and local EDA tools for validation. The system is hardware-agnostic and can run on any machine with network access.

Models: A tiered LLM strategy uses a more capable model for complex analysis and a lightweight model for fast iterations and debugging.

EDA Tools: Icarus Verilog 12.0 for compilation and simulation, Yosys 0.40 for synthesis checks.

Benchmark: VERILOGEVAL (156 problems, human variant).

B. VerilogEval Results

TABLE I: VerilogEval-Human Comparison

System	Pass@1
GPT-4 (single-shot)	$\sim$63%
CodeV-DS-33B	53.2%
VFlow	83.6%
MAGE	95.9%
ChipAgents	97.4%
CHIPCRAFTBRAIN (ours)	**98.7%**

CHIPCRAFTBRAIN achieves 154/156 (98.7%) on VERILOGEVAL-Human, surpassing all published systems. Of the 154 passing problems, 133 pass on the first iteration, 12 require a second refinement pass, and 3 require 3–4 iterations. Six problems are solved by the symbolic K-map solver with zero LLM cost and zero latency. The two remaining failures reveal distinct failure modes. *Prob066_edgecapture* compiles successfully on every iteration but produces 2 simulation mismatches per attempt; the generated logic is structurally identical to the reference but differs in reset assignment syntax, exposing a subtle simulator-specific timing edge case that the iterative refinement loop cannot resolve. *Prob092_gatesv100* fails compilation on all 5 iterations because the agent consistently selects a generate-block approach for 100-bit neighbor logic, which conflicts with iverilog's strict Verilog-2001 mode; the reference solves this elegantly via bit-sliced concatenation.

Both failures are architectural (wrong HDL construct or simulator semantics) rather than minor coding errors.

Total wall-clock time for the full 156-problem suite is 35.5 minutes (average 13.0s per problem).

C. Ablation Studies

To estimate the contribution of each component, we measure the incremental impact of progressively enabling system features. Note: these are estimated contributions based on observed behavior across the benchmark suite, not controlled single-variable ablations.

TABLE II: Estimated Incremental Component Contributions

Configuration	**Est. Pass@1**	Δ
Single-shot (no iteration)	$\sim$63%	baseline
+ Iterative refinement	$\sim$78%	+15%
+ Multi-agent selection	$\sim$85%	+7%
+ RAG knowledge	$\sim$88%	+3%
+ Spec guidance registry	$\sim$92%	+4%
+ Symbolic K-map solver	$\sim$96%	+4%
ChipCraftBrain (full)	**98.7%**	+35.7%

D. Cost Analysis

TABLE III: Cost Comparison Per Problem

System	**Cost/Problem**	**Strategy**
MAGE	$\sim$\$0.06	20× LLM calls
ChipCraftBrain	$\sim$\$0.01–0.03	Tiered LLM strategy
VFlow	Varies	MCTS-selected
Single-shot	$\sim$\$0.003	1× GPT-4

E. Orchestration Analysis

Of the 154 passing problems, 85% (133/156) pass on the first iteration, demonstrating strong baseline generation quality. The remaining 21 problems benefit from the adaptive orchestration loop, which switches agents and adjusts parameters across iterations. The high first-iteration success rate indicates that the primary value of the orchestration layer lies in recovery from failures rather than initial agent selection.

VII. Discussion

A. Key Findings

Our experiments reveal several important insights:

Validation-first is critical. The iterative refinement loop with structured error feedback bridges the gap from $\sim$63% single-shot to 98.7% pass@1, confirming the core thesis that *generating* code and *verifying* code are equally important. Of 154 passing problems, 21 required at least one refinement iteration.

Agent specialization matters. Both MAGE's ablation and our adaptive orchestration policy demonstrate that specialized agents with independent context windows significantly outperform single-agent approaches.

Knowledge provides consistent improvement. The knowledge base with focus-aware retrieval provides a consistent 3–5% improvement, with the largest gains on complex categories (memory, protocol) where code templates prevent common errors.

B. Limitations

- **SystemVerilog support**: Limited to the `-g2012` subset supported by Icarus Verilog
- **No timing closure**: No post-synthesis timing feedback loop yet
- **Benchmark scope**: VerilogEval's 156 problems are relatively simple; real-world designs are more complex
- **Formal verification**: No SVA property checking integration

C. Threats to Validity

VerilogEval problems may not fully represent real-world design complexity. API model versions evolve, potentially affecting reproducibility. Knowledge base quality directly affects RAG impact.

VIII. Future Work

Several directions extend ChipCraftBrain's capabilities:

Knowledge Distillation. Compressing the generation capability into a local 7B-parameter model via fine-tuning on our synthetic data factory output, enabling fully local, zero-API-cost generation.

Multi-Candidate Generation. Generating $N = 5$ candidates with synthesis-aware scoring using Yosys cell count and critical path metrics, following MAGE's insight that diverse candidates improve best-case quality.

Differential Checkpoint Debugging. Extending debug feedback with cone-of-influence analysis via Yosys, property-level abstractions from testbench assertions, and differential waveform analysis between passing and failing checkpoints.

CVDP Full Evaluation. Targeting >50% on NVIDIA's full 783-problem CVDP benchmark, currently the next frontier in RTL generation evaluation (current SOTA: $\sim$34%).

End-to-End Pipeline. Integration with OpenLane 2 and SKY130 PDK for a complete NL $\rightarrow$ RTL $\rightarrow$ GDSII $\rightarrow$ chip pipeline, enabling AI-generated silicon through chipIgnite.

Formal Verification. Adding SystemVerilog Assertion (SVA) property checking for deeper functional validation beyond simulation.

IX. Conclusion

We have presented ChipCraftBrain, a novel framework for automated RTL generation that integrates adaptive multi-agent orchestration, hybrid symbolic-neural reasoning, knowledge-augmented retrieval, hierarchical decomposition, and validation-first iterative refinement.

Our key contributions include: (1) adaptive multi-agent orchestration using a learned policy over a rich state representation; (2) focus-aware knowledge retrieval from a curated domain knowledge base; (3) hierarchical specification decomposition with cross-module port synchronization; and (4) a three-stage validation pipeline with structured error feedback.

On VerilogEval-Human, ChipCraftBrain achieves 98.7% pass@1 (154/156), surpassing ChipAgents (97.4%) and

MAGE (95.9%) at approximately 1/6th the API cost of prior multi-agent systems.

ChipCraftBrain demonstrates that combining specialized LLM agents with adaptive orchestration, domain knowledge, and rigorous validation can bring AI-generated RTL closer to production quality, contributing to the democratization of chip design.

References

[1] M. Liu, N. Pinckney, B. Khailany, and H. Ren, "VerilogEval: Evaluating large language models for verilog code generation," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2023.

[2] S. Thakur *et al.*, "VeriGen: A large language model for verilog code generation," *arXiv preprint arXiv:2308.00708*, 2023.

[3] Y. Liu *et al.*, "CodeV: Empowering LLMs with expert-level hdl code generation," *arXiv preprint arXiv:2407.10424*, 2024.

[4] S. Liu *et al.*, "RTLCoder: Fully open-source and efficient LLM-assisted RTL code generation technique," *arXiv preprint arXiv:2312.08617*, 2024.

[5] S. Thakur *et al.*, "AutoChip: Automating HDL generation using LLM feedback," *arXiv preprint arXiv:2311.04887*, 2023.

[6] Y.-D. Tsai *et al.*, "MAGE: A multi-agent engine for automated RTL code generation," *arXiv preprint arXiv:2412.04211*, 2024, 95.9% VerilogEval-Human v2.

[7] Y. Wei, Z. Huang, H. Li, W. W. Xing, T.-J. Lin, and L. He, "VFlow: Discovering optimal agentic workflows for verilog generation," *arXiv preprint arXiv:2504.03723*, 2025, 83.6% pass@1 on VerilogEval.

[8] J. Blocklove *et al.*, "Chip-Chat: Challenges and opportunities in conversational hardware design," in *Proceedings of the Workshop on Machine Learning for CAD (MLCAD)*, 2023.

[9] Alpha Design AI, "ChipAgents: Agentic AI for chip design," 2025, commercial system. 97.4% VerilogEval-v2. $24M funding.

[10] Y. Li *et al.*, "Competition-level code generation with AlphaCode," *Science*, vol. 378, no. 6624, pp. 1092–1097, 2022.

[11] H. Le *et al.*, "CodeRL: Mastering code generation through pretrained models and deep reinforcement learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[12] Synopsys, Inc., "DSO.ai: Design space optimization AI," 2020, RL-based P&R optimization.

[13] Cadence Design Systems, "Cerebrus: Intelligent chip explorer," 2021, ML design space exploration.

[14] Efabless Corporation, "OpenLane 2: Open-source digital ASIC implementation flow," 2024, rTL-to-GDSII with SKY130 PDK.

[15] W. V. Quine, "The problem of simplifying truth functions," *The American Mathematical Monthly*, vol. 59, no. 8, pp. 521–531, 1952.

[16] E. J. McCluskey, "Minimization of boolean functions," *Bell System Technical Journal*, vol. 35, no. 6, pp. 1417–1444, 1956.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.