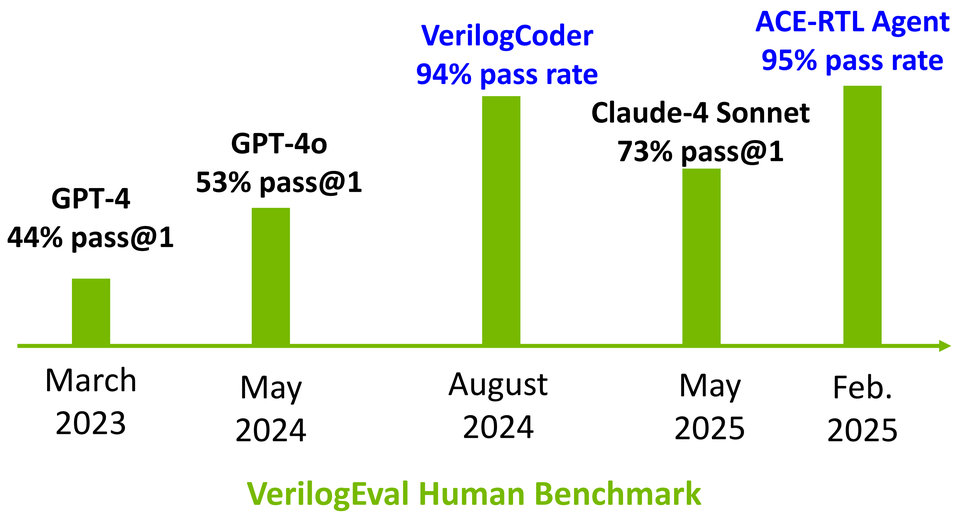
Learning Agentic AI for Hardware Design Automation

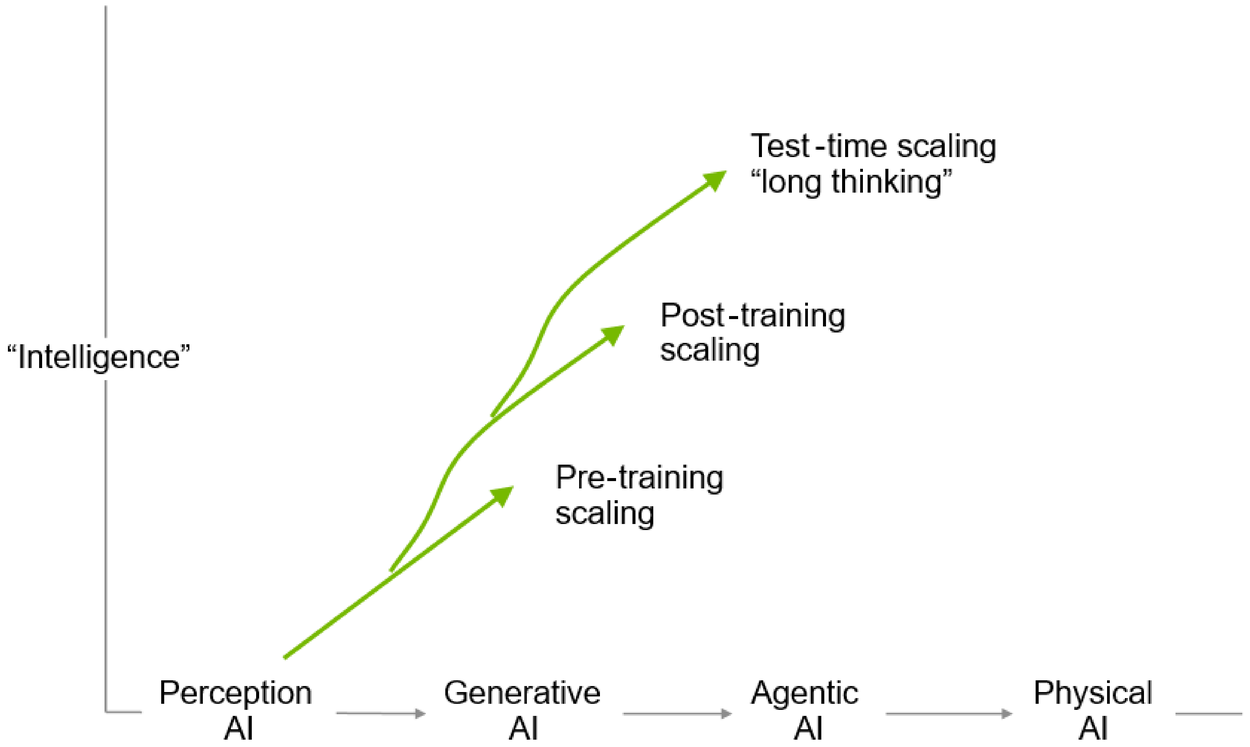
Chia-Tung (Mark) Ho | EDPS 2026

AI for Chip Design Automation

- Chip Design: Extensive domain knowledge required
  - Diverse design knowledge: Logic design, physical design, verification, analog design, etc.
  - Diverse EDA tool chains: DC Compiler, PrimeTime, ICC2, Innovus, reports, LEF/DEF, etc.

- AI for Chip Design Automation
  - Significant improvement from agentic method





Source: "From VLSI through Algorithms: Optimizing the Computing Stack for Agentic AI", Brucek Khailany, ISSCC 2026

VerilogEval: https://doi.org/10.1109/ICCAD57390.2023.10323812
VerilogCoder: https://doi.org/10.1609/aaai.v39i1.32007
ACE-RTL: https://www.arxiv.org/abs/2602.10218

Agentic AI for Chip Design

Learning-Based Agentic AI

Multi-Agent & Multi-Modality Agentic System

Knowledge Graph, Graph-Assisted Agentic Methods, Tree Search (MCTS)

Prompting, ReACT, CoT

main Adapted Foundati Model

LLM

ChipNemo [Arxiv'23]
(Mingjie Liu et al.)
DAPT for Chip Design

RTLFixer [DAC'24]
(Yun-Da, Mingjie Liu, Mark Ren)
ReAct + RAG

LLM for Standard Cell Design Optimization
[NTECH'24, ICLAD'24 Best Paper]
(Mark Ho, Mark Ren)
ReAct+ Netlist Tools Reasoning

Timing Analysis Agent [NTECH'24]
(Jatin Nainani, Mark Ho, et al)
Multi-Agent + TDRG

RTLRewriter [ICCAD'24]
(Xufeng Yao et al.)
Cost-aware MCTS for RTL Rewrite

CodeV [TCAD' 25]
(Yang Zhao et al.)
Fine Tuned RTL Model

ChipAlign [DAC' 25]
(Chenhui Deng et al.)
Instruction following

VerilogCoder [AAAI'25]
(Mark Ho, Mark Ren, Brucek Khailany)
Multi-Agents + Dynamic Task Graph + Customized Waveform Debugger

DRC-Coder [NTECH'24, ISPD'25]
(Chen-Chia Cheng, Mark Ho, Mark Ren)
Vision & Multi-Agent + ReAct

BRIDGE [ICLAD'25]
(Wei Li et al.)
Graph Modality + LLM

Cursor + MCP + MCP Rule [Commercial Agent]

Claude Code, Codex [Commercial Agent]

Marco [VLSIT'25]
(Mark Ho et al.)
Multi-Agents System for Chip Design

MAGE [DAC'25]
(Yujie Zhao et al.)
Multi-Agent RTL Generation

ASPEN [MLCAD'25]
(Niansong Zhang et al.)
Egraph + Agent => RTL Rewrite

AssertForge [ICLAD'25]
(Yunsheng Bai et al.)
Knowledge Graph Reasoning SVAGen

ScaleRTL [MLCAD'25]
(Chenhui Deng et al.)
Reasoning model for RTL

Pro-V-R1 [DAC'26]
(Yujie Zhao et al.)
Multi-Agent + GRPO for RTL Testbench

FVRuleLearner [VTS'26]
(Lily Wan, Mark Ho, et al.)
Agent Training, Structural Reasoning

AutoPD [ICCAD'26]
(Mark Ho et al.)
Auto-Research on EDA Algorithm

EvoDRC [ASP-DAC'27]
(Bing-Yu Wu et al.)
Skill Evolved DRC Fixing

Multi-Agent Self-Evolved ABC [DAC'26]
(Cunxi Yu et al.)
Self-Evolving Coding to Improve ABC

Polymath [ISPD'26]
(Mark Ho, et al.)
Self-Improved Workflow

SchemaCoder [DAC'26]
(Lily Wan, Mark Ho, et al.)
Unsupervised Training + Gradient Boosting
Log Analysis

EDAFlowQA [MLCAD'26]
(Mark Ho et al.)
Q-Tree Driven QA Generation

HeaRT [ICLAD'26]
(Souradip Poddar et al.)
Hierarchical circuit reasoning tree

FVDebug [DVCon'26]
(Yunsheng Bai et al.)
Causal graph + Jasper Gold

GRPO-SMu [ISQED' 26]
(Dimple Kochar et al.)
Fine-Tuned LLM for Test Plan Generation

SCALE [ASP-DAC'27]
(Mark Ho et al.)
VLM training for DRV Fixing

LLM4Cov [ICML'26]
(Hejia Zhang et al.)
Stage-wise SFT with Agentic Dataset

TopCell [DAC'26]
(Zhan Song et al.)
LLM-Based Schematic Topology Exploration

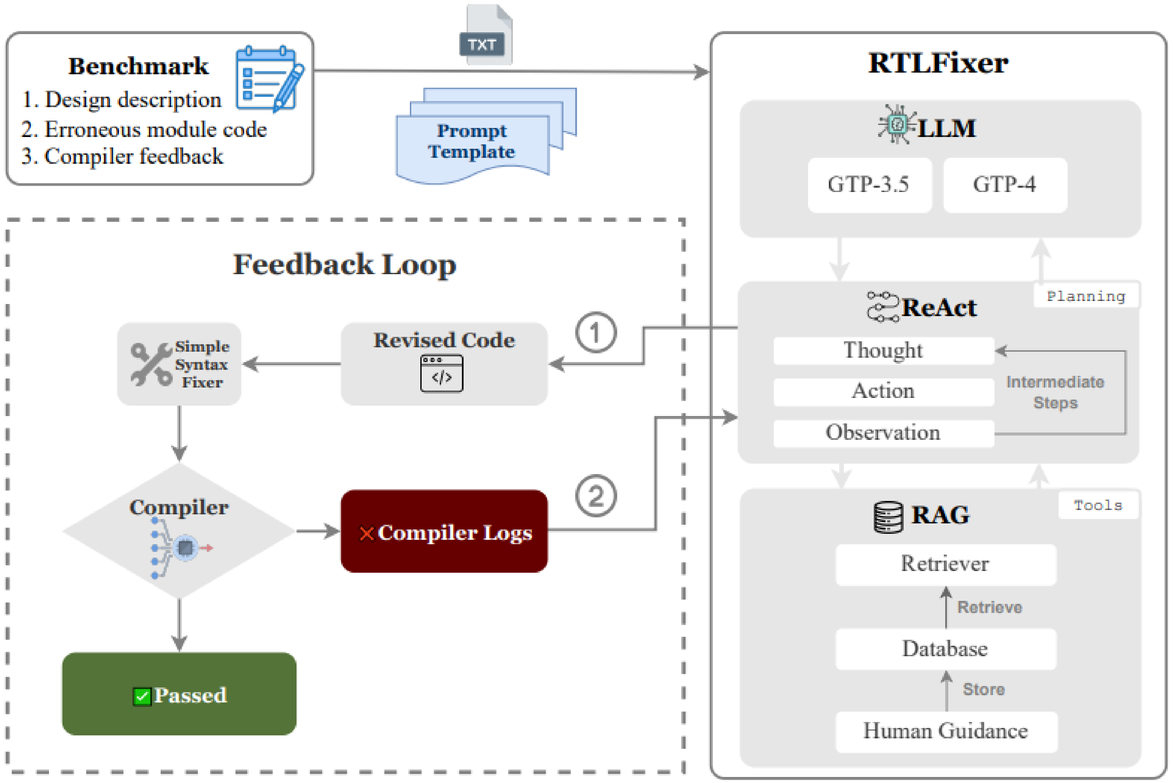
2024 2025 2026

Note: There are many interesting works across different research groups. We selectively list the works due to space limits.

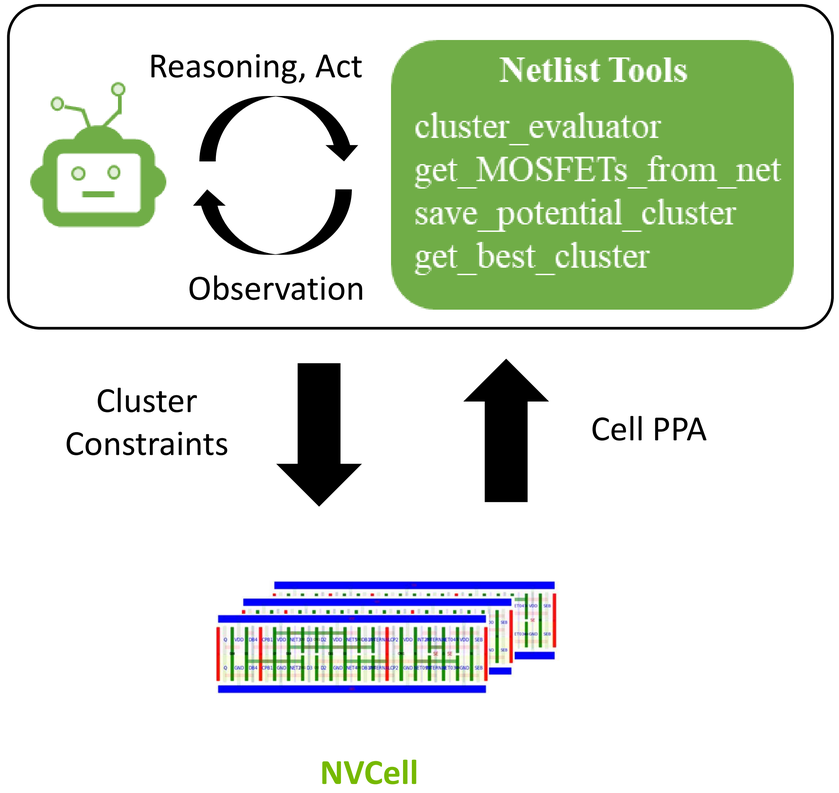
NVIDIA

Prompt Engineering, ReACT, CoT

RTLfixer
[DAC'24]



LLM for Standard Cell Layout Design Opt.
[ICLAD'24 Best Paper Award]



NVCell

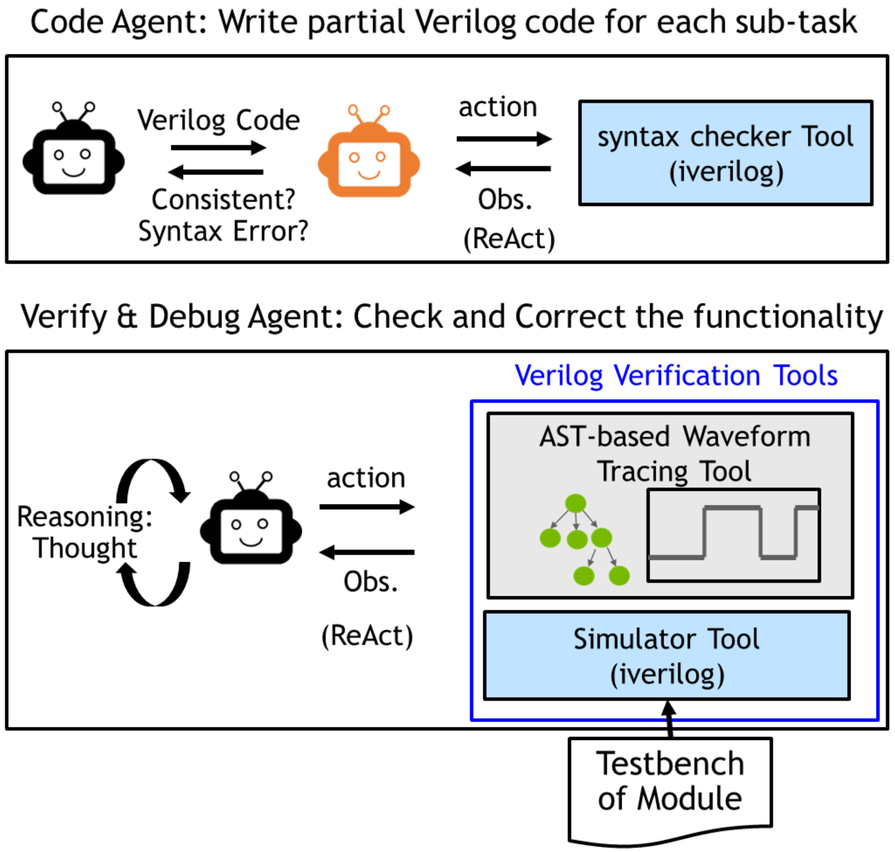
Ho, Chia-Tung, et al. "Nvcell 2: Routability-driven standard cell layout in advanced nodes with lattice graph routability model." *Proceedings of the 2023 International Symposium on Physical Design*. 2023.

Ren, H., & Fojtik, M. (2021, December). Nvcell: Standard cell layout in advanced technology nodes with reinforcement learning. In *2021 58th ACM/IEEE Design Automation Conference (DAC)* (pp. 1291-1294). IEEE.

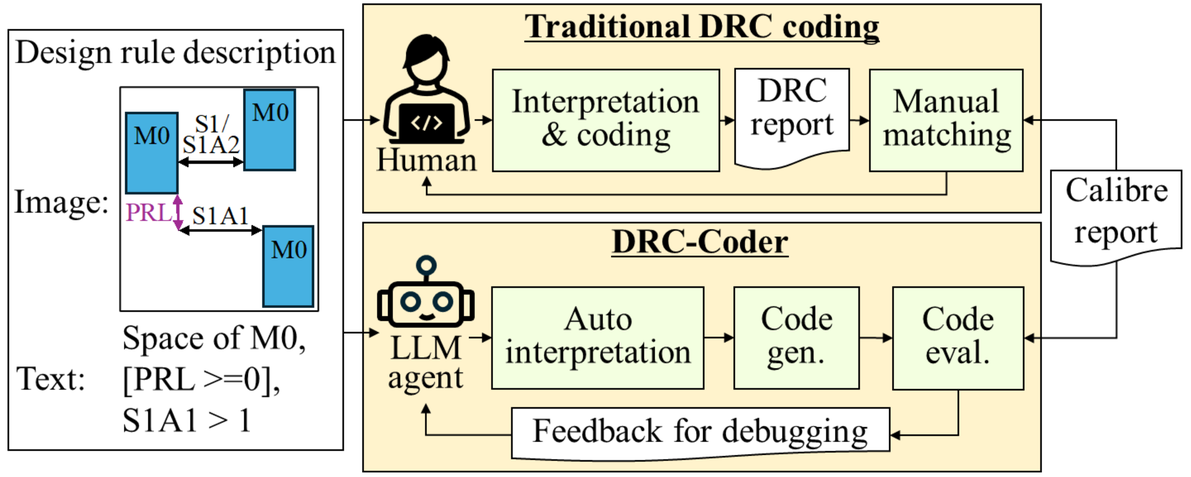
Multi-Agent System

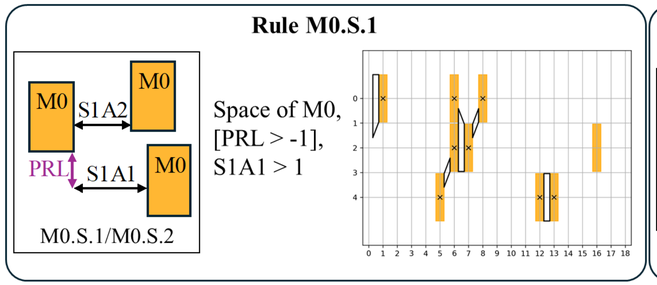
Multi-Modality, Waveform, Knowledge Graph, MCTS

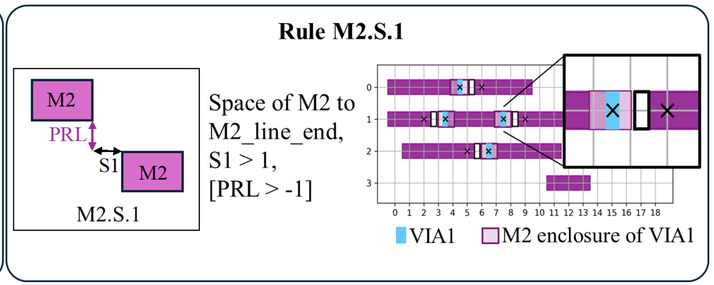
VerilogCoder [AAAI'25]: First using AST-based Waveform Tracer in Multi-Agent System



DRC-Coder [ISPD'25]: First exploring VLM (image) for DRC rule coding in Multi-Agent System







Self-Evolving, Learning-Based Agentic Method

- Challenges customized hand-crafted agentic system
  - Hard to scale
  - Burning tokens but didn’t learn from previous experience
  - Customized tools are not flexible and adaptable to various similar task
  - Require updates once the advancement of LLM tool version to maintain efficiency
  - …
- Learning-Based Agentic Methodology
  - Evolving the coding & algorithm [AlphaEvolve, OpenEvolve]
  - Self-Evolving rules for EDA algorithm evolution (Multi-Agent Self-Evolved ABC [DAC’26])
  - **Evolving Prompt & Workflow (Polymath [ISPD’26])**
  - Unsupervised agentic training for extract key schema from design artifacts to improve log analysis (SchemaCoder [DAC’26])
  - **Rule learning for hard problems (FVRuleLearner [VTS’26])**
  - **Automated Research (AutoPD [ICCAD’26])**
  - Learning-Based DRV fixing (EvoDRC [ASP-DAC’27])
  - …
- Post-training with Agentic Trajectories
  - GRPO-SMu, ScaleRTL, LLM4Cov, Pro-V-R1, SCALE, …

Polymath: Self-Improving Hierarchical Workflow for Multi-Domain Problem Solving [ISPD'26]

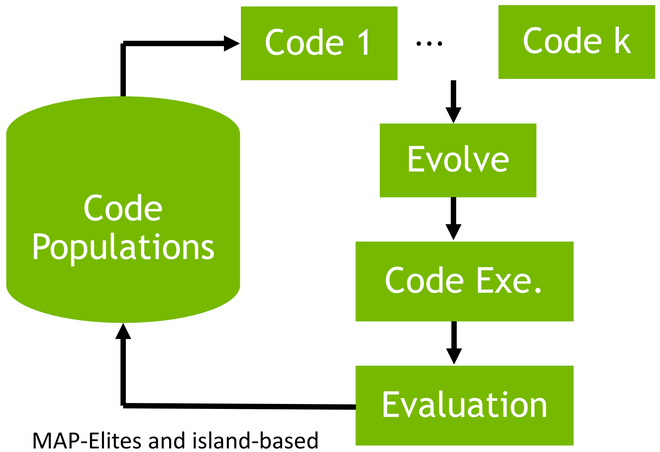
Self-Improving Prompt & MAS workflow using LLM judges score on the fly

Self-Reflection Guided Evolutionary Algorithm

$$S_w = \{ (V(A,P),L) \mid A \in \mathcal{A}, P \in \mathcal{P}, L \in \mathcal{L} \}$$

$$W^* = \arg\max_{W \in S_w} u(W,K,t)$$

- LLM Assistants A: Each assistant is a functional unit capable of performing a complete task, such as coding, reasoning, or file reading.
- Assistant Instruction Prompt P: The input instructions or task descriptions provided to
- Links L: The abstract structures define node relationships and govern execution flow.



MAP-Elites and island-based population models.

```
def workflow(instruction: str):
    """
    Workflow to complete the dif_Square function task using the coder assistant.
    """
    # Construct the initial chat for reasoning
    reason_task = instruction + "\nReason about the angle relationships in the pentagon."

    # Use strong_reason_debug_assistant for reasoning
    reason_response = reason_assistant.initiate_chat(message=reason_task)

    [Partial code omitted due to space limit]

    # Prepare inputs for the coder assistant
    coder_task = reason_extracted + "\nUse Python to calculate the common measure of the large angles."
    print("Input to coder:", coder_task)

    # Use the coder assistant to calculate
    response = coder_assistant.initiate_chat(message=coder_task)

    return response
```

[Problem]: Sam is carrying a 2 gallon bucket of milk to his house from the barn, which is 3 miles away from his house. However, the bucket has a leak in it. For each mile he walks, there is $\\frac{2}{3}$ as much milk in the bucket as at the beginning of the mile. How many gallons of milk will be in the bucket when Sam gets home?

[Target Current subtask request to solve]: Calculate the amount of milk remaining after each mile walked. Sam is carrying a 2-gallon bucket of milk, and for each mile walked, there is 2/3 as much milk in the bucket as at the beginning of the mile.

```
# EVOLVE-BLOCK-START
def workflow(instruction: str):

    instruction = "### Instruction Background ###\n" + instruction + "\n"
    # Task 1: Calculate the amount of milk remaining after each mile walked
    current_task = "Task_1_Task_1"
    coder_instruction = instruction + f"\n{current_task}: Calculate and print the amount of milk remaining after each mile walked. Use print(f'After mile {{mile}}: {{amount:.4f}} gallons') for each mile.\n"

    # Generate multiple responses
    num_ensemble = 3
    gen_outs = []
    for i in range(num_ensemble):
        response = coder_assistant.initiate_chat(message=coder_instruction)
        formatted_response = extract_output(response=response, post_process_func=coder_description["post_process_func"])
        gen_outs.append(f"### Generated {i}-th response\n{formatted_response}")

    # Rank responses
    rank_instruction = f"Rank the following code snippets from best to worst based on correctness, efficiency, and adherence to the task:\n\n{''.join(gen_outs)}\n\nOutput the index of the best response."
    rank_response = general_assistant.initiate_chat(message=rank_instruction)
    best_index = int(extract_output(response=rank_response, post_process_func=general_assistant_description["post_process_func"]))

    formatted_response = gen_outs[best_index].split("### Generated")[1].split("-th response\n")[1]

    # Task 2: Calculate the final amount of milk in the bucket after 3 miles
    current_task = "Task_1_Task_2"
    reason_instruction = formatted_response + f"\n{current_task}: Calculate the final amount of milk in the bucket after 3 miles. Include the breakdown of milk amounts after each mile in your response.\n"
    response = strong_reason_debug_assistant.initiate_chat(message=reason_instruction)
    initial_response = extract_output(response=response, post_process_func=reason_agent_description["post_process_func"])

    # Self-reflection
    reflection_prompt = f"""
    ### Task ###
    Provide suggestions, feedback, and reflection on the generated result.

    ### Task ###
    {current_task}: Calculate the final amount of milk in the bucket after 3 miles. Include the breakdown of milk amounts after each mile in your response.

    ### Previous Generated Result ###
    {initial_response}

    Reflection:
    """
    reflection_response = strong_reason_debug_assistant.initiate_chat(message=reflection_prompt)
    reflection_output = extract_output(reflection_response, reason_agent_description["post_process_func"])

    # Incorporate reflection into final response
    final_response = f"{initial_response}\n\nReflection and improvements:\n{reflection_output}"

    # Extract the final numerical answer and ensure it's in the required boxed format
    import re

    numerical_answer = re.search(r'\d+\.\d+', final_response)
    if numerical_answer:
        boxed_answer = f"$boxed{{{numerical_answer.group()}}}$"
        final_response = f"{final_response}\n\nFinal Answer: {boxed_answer}"
    else:
        final_response = f"{final_response}\n\nError: No numerical answer found."

    return final_response
# EVOLVE-BLOCK-END
```

LLM Ensemble (Write code to calculate)

Ranking & Get best response

Reflection & Provide Final Answer

Make sure the Output Format is Correct

Figure 9: An example of generated code-represented workflows from self-reflection-guided evolutionary algorithm that includes ensembling approach, ranking or voting, reflection on top of coding and reasoning.

NVIDIA.

Hardware Design Tasks

- Run the agent on a set of sampled questions; Prior experience + Self-Evolve (5 times) on the fly.

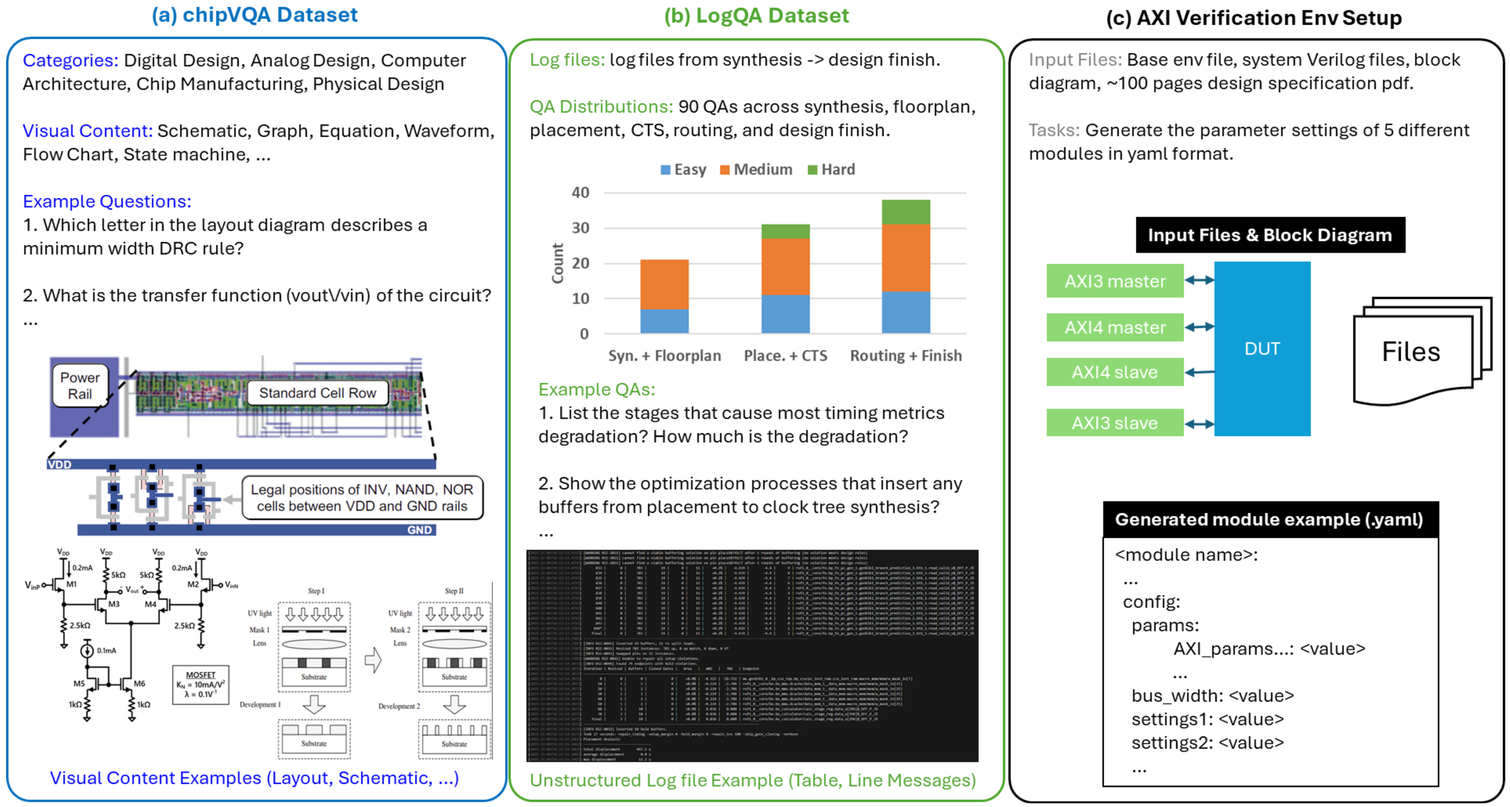


Table 1: Performance comparison of Cursor [10] (enterprise version) and Polymath on chipVQA [55], LogQA, and AXI verification environment setup tasks. Cursor uses Sonnet-4.5 [3]. Each task is evaluated over 10 runs for pass@1 and pass@3 metrics. Best scores in bold.

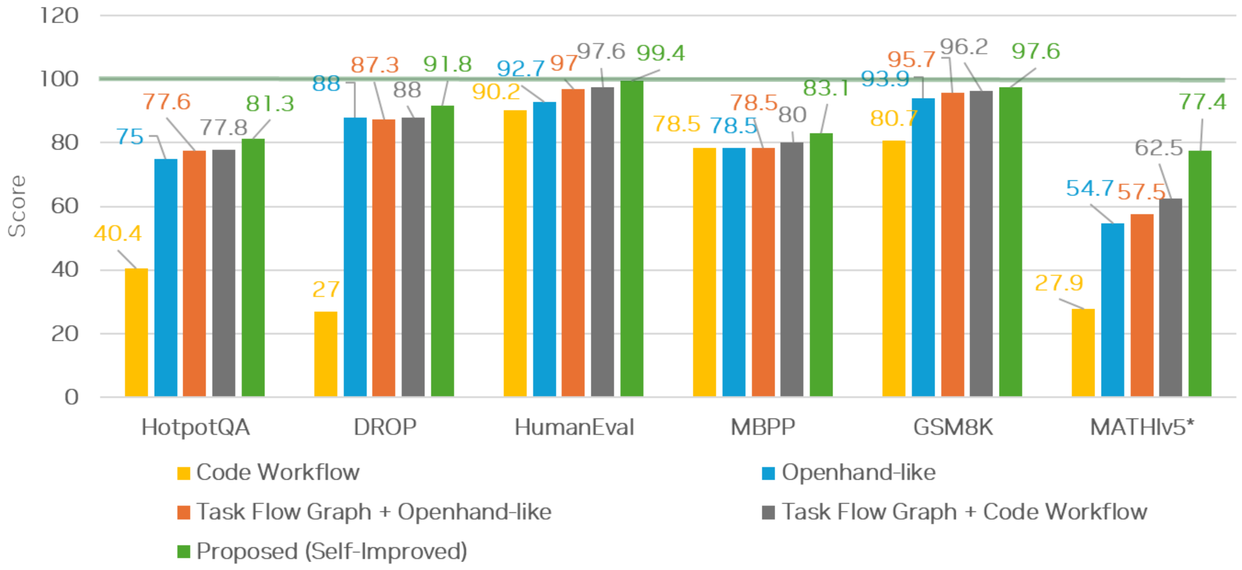
Method	Chip VQA										LogQA (90 QAs)								AXI Verification Env Setup	
	Digital		Analog		Arch.		Manu.		Physical		Syn. + Floorplan		Place. + CTS		Routing + Finish		All QAs			
	pass@1	pass@3	pass@1	pass@3	pass@1	pass@3	pass@1	pass@3	pass@1	pass@3	pass@1	pass@3	pass@1	pass@3	pass@1	pass@3	pass@1	pass@3	pass@1	pass@3
Cursor	55.7%	70.3%	73.0%	89.2%	43.0%	53.2%	38.0%	50.8%	55.5%	71.6%	**99.4%**	**100.0%**	**83.3%**	**86.7%**	72.7%	80.9%	82.6%	87.3%	62.5%	75.9%
Proposed	**68.9%**	**78.2%**	**81.4%**	**89.7%**	**48.0%**	**59.8%**	**49.0%**	**57.3%**	**68.7%**	**78.7%**	97.6%	**100.0%**	77.5%	84.0%	**86.8%**	**94.0%**	**86.1%**	**91.9%**	**97.8%**	**100.0%**
Impr.	13.1%	7.9%	8.4%	0.4%	5.0%	6.6%	11.0%	6.5%	12.2%	7.6%	-1.8%	0.0%	-5.9%	-2.7%	14.1%	13.1%	3.5%	4.6%	35.3%	24.1%

General Coding & Reasoning

Achieve > 10% solve rate on MATH level 5 questions than previous state-of-the-art methods

Table 2: Performance comparison of vanilla models, manually designed methods and automated workflow generation methods for QA, coding, and Math scenarios. We reference and show the performance scores of CoT, CoT SC, MultiPersona, Self Refine, ADAS, and AFlow from [63] which are the average scores of three runs. For vanilla gpt-4o and o1-model runs, we follow [63] and report the average scores of three runs. We run every benchmark once using the proposed method. The best score is highlighted in bold.

Method	Multi-Turn QA		Coding		Math		Avg.
	HotpotQA	DROP	HumanEval	MBPP	GSM8K	MATH	
gpt-4o (Vanilla)	75.0	64.7	91.5	74.9	85.5	48.2	73.3
o1 (Vanilla)	70.6	84.9	89.0	74.5	94.6	67.1	80.1
CoT [52]	67.9	78.5	88.6	71.8	92.4	48.8	74.7
CoT SC (5-shot) [50]	68.9	78.8	91.6	73.6	92.7	50.4	76.0
MultiPersona [51]	69.2	74.4	89.3	73.6	92.8	50.8	75.1
Self Refine [30]	60.8	70.2	87.8	69.8	89.6	46.1	70.7
ADAS [18]	64.5	76.6	82.4	53.4	90.8	35.4	67.2
AFlow [63]	73.5	80.6	94.7	**83.4**	93.5	56.2	80.3
Proposed	**81.3**	**91.8**	**99.4**	83.1	**97.6**	**77.4**	**88.4**



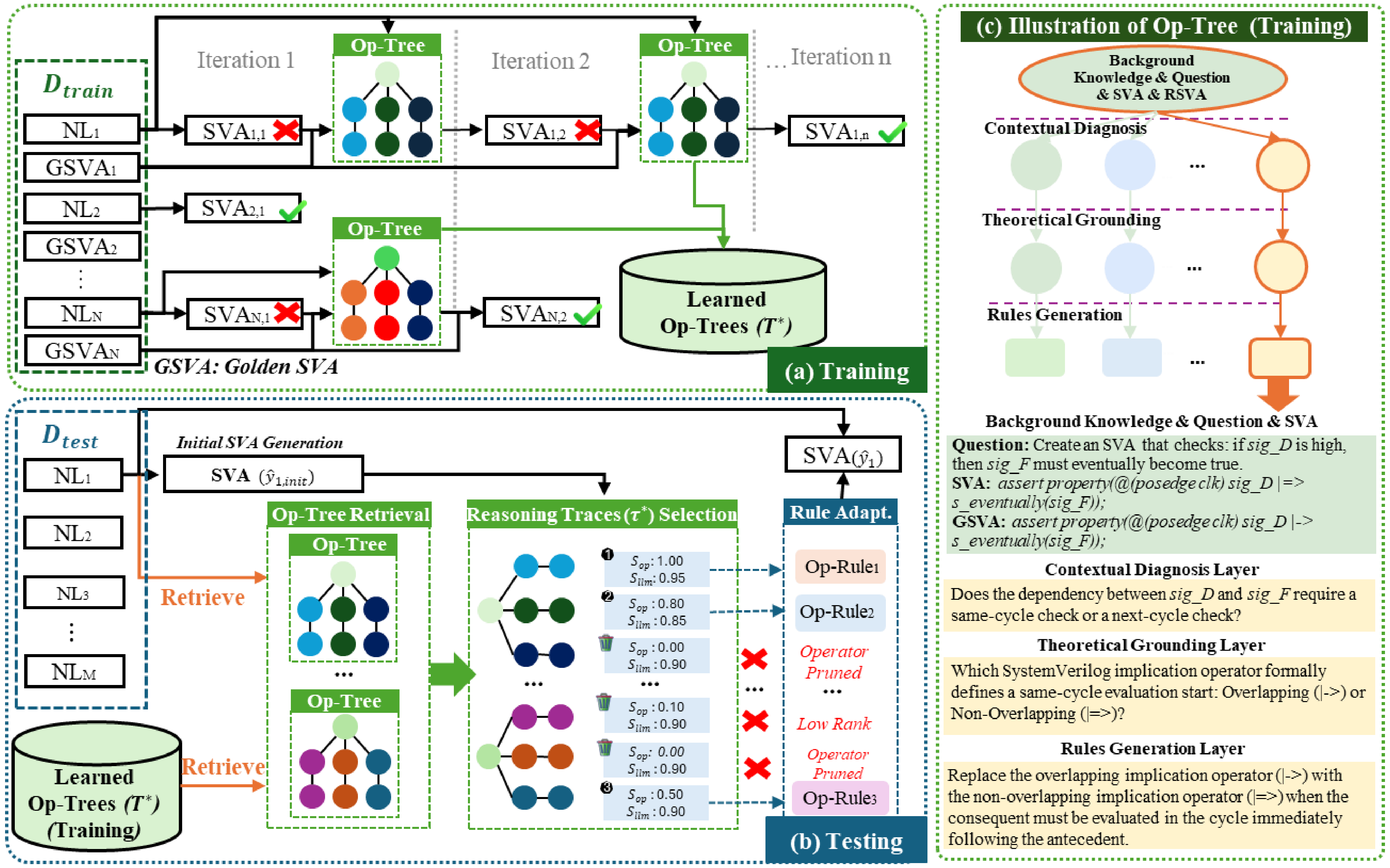
FVRuleLearner [VTS'26]

Poster: FVRuleLearner: An SVA Generation Agent with Rule Learning via Operator-Level Reasoning Trees

- Functional correctness of SVA generation is challenging
  - 0.467 pass-rate GPT-4o (FVEval [DATE'25])
- Goal: Learning Op-Rules to improve functional correctness
- Dataset: 80% for training, 20% for testing

- Training Phase:
  - Weak Learner Op-Tree (LLM-based)
  - Learned Corrected Op-Trees

- Testing Phase:
  - Fetch Top-k reasoning traces from N weak learner Op-Trees
  - Adaptation: mask unnecessary signal name.
  - Generate Op-Rules



Kang, Minwoo, Mingjie Liu, Ghaith Bany Hamad, Syed M. Suhaib, and Haoxing Ren. "Fveval: Understanding language model capabilities in formal verification of digital hardware." In *2025 Design, Automation & Test in Europe Conference (DATE)*, pp. 1-6. IEEE, 2025.

NVIDIA.

Experiment Results: Testing Phase

- FVRuleLearner achieves SoTA 85.50% functional correctness with 31.17% improvement over the strongest existing baselines.
- FVRuleLearner achieves 98.39% syntax correctness and 86.24% relaxed functionality on average.

TABLE II: Experimental Results of LLM Models on NL2SVA-HUMAN, NL2SVA-MACHINE, and NL2SVA-OPENCORE Testing Sets.

Model / Method	NL2SVA-HUMAN				NL2SVA-MACHINE				NL2SVA-OPENCORE			
	BLEU	SYN	FUNC	R.FUNC	BLEU	SYN	FUNC	R.FUNC	BLEU	SYN	FUNC	R.FUNC
GPT-4o												
FVEVAL-0-SHOT	0.5203	0.9333	0.5333	0.6667	0.6054	0.9500	0.4167	0.4500	0.6145	0.9250	0.7050	0.7200
FVEVAL-3-SHOT	0.5823	0.9333	0.4000	0.6000	0.6461	0.9500	0.4333	0.5167	0.6187	0.8950	0.6750	0.6950
ENSEMBLE	0.5876	0.8667	0.4000	0.4667	0.6562	0.9667	0.5000	0.6000	0.5922	0.9550	0.7000	0.7150
FVRULELEARNER	**0.6846**	**1.0000**	**0.8000**	**0.8000**	**0.7072**	**0.9667**	**0.7500**	**0.8167**	**0.6976**	**0.9900**	**0.8500**	**0.8750**
Claude 4.5 Sonnet												
FVEVAL-0-SHOT	0.5330	**1.0000**	0.6000	**0.8667**	0.6246	0.9167	0.5333	0.6167	0.5868	0.9350	0.6900	0.7000
FVEVAL-3-SHOT	0.6480	**1.0000**	0.5000	0.5000	0.6765	0.9500	0.4667	0.5667	0.6154	0.9150	0.7050	0.7250
ENSEMBLE	0.5324	**1.0000**	0.5333	0.8000	0.6212	0.9500	0.5333	0.6333	0.5881	0.9600	0.7050	0.7250
FVRULELEARNER	**0.7416**	**1.0000**	**0.8667**	**0.8667**	**0.7038**	**0.9667**	**0.7500**	**0.8167**	**0.6901**	**0.9700**	**0.8650**	**0.8900**
o3-mini												
FVEVAL-0-SHOT	0.5886	0.9333	0.4667	0.6000	0.6075	0.9333	0.4500	0.5333	0.5900	0.9600	0.6800	0.7000
FVEVAL-3-SHOT	0.6360	0.9333	0.4667	0.5000	0.6696	0.9500	0.4833	0.5833	0.6189	0.8950	0.6800	0.7150
ENSEMBLE	0.6127	0.9333	0.4667	0.5500	0.6127	0.9333	0.4667	0.5500	0.5900	0.9600	0.6800	0.7000
FVRULELEARNER	**0.7363**	**1.0000**	**0.8667**	**0.9333**	**0.6980**	**0.9667**	**0.7833**	**0.8333**	**0.7145**	**0.9950**	**0.9150**	**0.9300**

Note: BLEU= BLEU score, SYN= Syntax, FUNC= Functionality, R.FUNC= Relaxed Functionality.

AutoPD: Question-Driven MCTS from Paper Artifacts to PPA-Adaptive Physical-Design Algorithms [ICCAD'26]

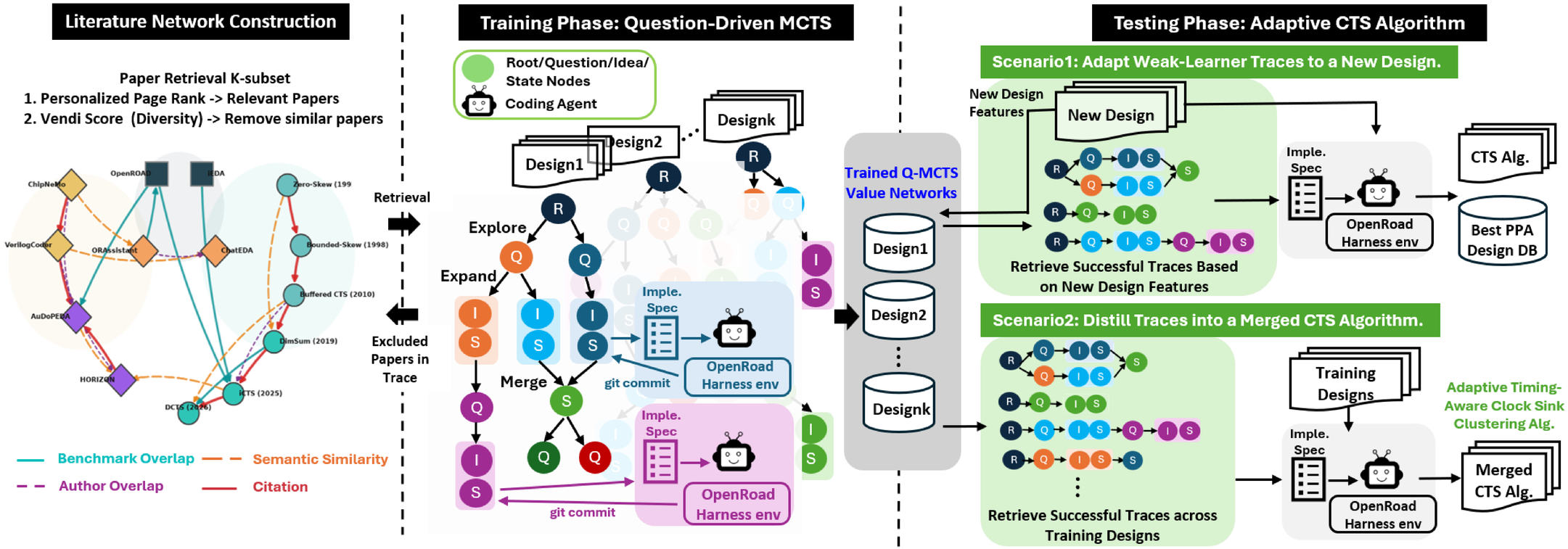
- **Given**: 1) A corpus of physical-design research papers, 2) An initial EDA CTS codebase (e.g., OpenROAD)
- **Find**: Algorithm-level code modifications that improve post-route PPA (WNS, TNS, power, area) across designs
- **Subject to**:
  - No regressions on timing-clean designs
  - Modifications must be implementable and integrable into the existing flow
  - Solutions must generalize to unseen designs, not just the training set

- **Key Difficulties**: Paper ideas are not drop-in: most require non-trivial adaptation to work in a real flow
  - CTS-stage gains can become post-route regressions (e.g., iCTS reduces buffer area by 92% but worsens WNS by 10x)
  - The same change can help one design and hurt another
  - The hypothesis space is combinatorial: which paper, which idea, which code location, which design

- **Some findings in the study:**
  - Without expert knowledge, coding agent can not make meaningful improvement on EDA algorithm
  - Without a systematic learning mechanisms, coding agent struggle to create competitive EDA optimization algorithms
  - **Agentic TCL Script Tuning + EDA Algorithm Co-Optimization for design -> avg 57.6%/63% WNS/TNS improvement**

Flow Overview

- Retrieval: Given a Query, we retrieved the relevant seed nodes and perform Personalized Page Rank + Vendi Score
- Rewards: Normalized Delta Post-Route PPA + Coding Agent Execution Reward (Clean build, Crash, etc)
- Actions: Explore (Questions), Expand (Ideas), Execute (State), Merge (States)

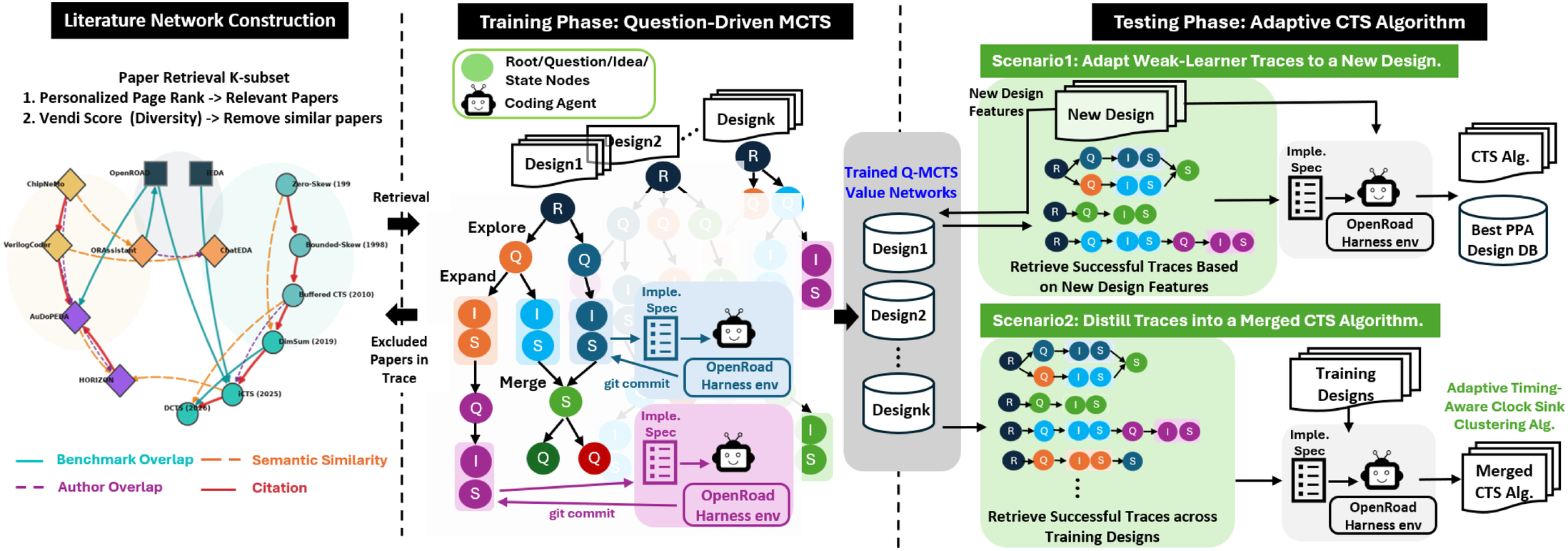


Testing Phase

Q-MCTS training is time consuming

- Scenario 1: Adapt Weak-Learner Traces to a New Design.
  - Profiles the unseen design under geometry, capacitance, and timing features, then retrieves successful directions from the most similar weak learners.
  - Leverage coding agent and the successful traces to customize the CTS algorithm for a new design.

- Scenario 2: Distill Traces into a Merged CTS Algorithm.
  - Retrieve the successful traces in the training phase.
  - Generate a merged CTS algorithm for any unseen designs.



Main Experiment

Table 1: Detailed CTS and detailed-route comparison across nine designs using an open-source 7nm PDK. CTS columns report delay, skew, inserted buffer area, and buffer count; detailed-route columns report setup WNS/TNS, area, routed wirelength, and total design power. Positive setup WNS is reported as 0.00 because the table reports negative slack, not positive slack margin. Parenthesized percentages report improvement over the OpenROAD baseline for the same design and metric; lower area, wirelength, and power values are green, while larger values are red. The Avg. block averages each method across designs, using available numeric metrics. Bold marks the best value per design or average block. Percentages are omitted when the OpenROAD baseline is zero and a ratio is undefined.

		CTS Stage				Detailed Route				
Design	Method	Delay (ps)	Skew (ps)	Buf. Area (μm^2)	Bufs (#)	WNS (ps)	TNS (ps)	Area (μm^2)	Wirelength (μm)	Total Pwr (mW)
AES	OpenROAD baseline [1]	88.59	**12.70**	22.74	37	-1.79	-1.79	1638.33	52735.16	150.45
	iCTS [10]	**83.08**	20.08	**1.84**	**18**	-180.71	-24.2k	**1609.00**	**52455.00**	**111.47**
	Horizon (Goal only) [24]	88.59 (0.0%)	**12.70** (0.0%)	22.74 (0.0%)	37 (0.0%)	-1.79 (0.0%)	-1.79 (0.0%)	1638.33 (0.0%)	52735.16 (0.0%)	150.00 (+0.3%)
	Horizon (Papers) [24]	90.79 (-2.5%)	14.03 (-10.5%)	22.83 (-0.4%)	37 (0.0%)	**0.00** (+100.0%)	**0.00** (+100.0%)	1638.15 (0.0%)	52704.69 (+0.1%)	150.47 (0.0%)
	QMCTS-CTS	90.79 (-2.5%)	14.03 (-10.5%)	22.83 (-0.4%)	37 (0.0%)	**0.00** (+100.0%)	**0.00** (+100.0%)	1638.15 (0.0%)	52704.69 (+0.1%)	150.47 (0.0%)
	Ours (WL)	90.79 (-2.5%)	14.03 (-10.5%)	22.83 (-0.4%)	37 (0.0%)	**0.00** (+100.0%)	**0.00** (+100.0%)	1638.15 (0.0%)	52704.69 (+0.1%)	131.00 (+12.9%)
	Ours (Merged)	90.79 (-2.5%)	14.03 (-10.5%)	22.83 (-0.4%)	37 (0.0%)	**0.00** (+100.0%)	**0.00** (+100.0%)	1638.15 (0.0%)	52704.69 (+0.1%)	150.47 (0.0%)
AES-Blk	OpenROAD baseline [1]	187.59	32.42	24.42	40	-8.96	-32.73	5862.97	43980.47	7.86
	iCTS [10]	**143.30**	**24.08**	**1.94**	**19**	-8.18	-15.04	**709.00**	**36509.00**	**6.57**
	Horizon (Goal only) [24]	187.59 (0.0%)	32.42 (0.0%)	24.42 (0.0%)	40 (0.0%)	-8.96 (0.0%)	-32.73 (0.0%)	5862.97 (0.0%)	43980.47 (0.0%)	7.86 (0.0%)
	Horizon (Papers) [24]	187.52 (0.0%)	32.33 (+0.3%)	24.48 (-0.2%)	41 (-2.5%)	-4.16 (+53.6%)	-22.51 (+31.2%)	5863.07 (0.0%)	44009.88 (-0.1%)	7.88 (-0.3%)
	QMCTS-CTS	187.52 (0.0%)	32.33 (+0.3%)	24.48 (-0.2%)	40 (0.0%)	-4.17 (+53.5%)	-22.51 (+31.2%)	5863.07 (0.0%)	44009.88 (-0.1%)	6.68 (+15.0%)
	Ours (WL)	175.99 (+6.2%)	35.82 (-10.5%)	24.98 (-2.3%)	42 (-5.0%)	-2.43 (+72.9%)	**-4.06** (+87.6%)	5864.24 (0.0%)	44024.10 (-0.1%)	8.51 (-8.3%)
	Ours (Merged)	177.08 (+5.6%)	35.73 (-10.2%)	24.73 (-1.3%)	40 (0.0%)	-4.31 (+51.9%)	-16.69 (+49.0%)	5863.66 (0.0%)	43938.77 (+0.1%)	7.82 (+0.5%)
AES-MBFF	OpenROAD baseline [1]	111.39	21.56	24.92	39	**0.00**	**0.00**	1697.75	60004.39	155.51
	iCTS [10]	109.52	28.08	**1.74**	**17**	-168.52	-20.9k	**1639.00**	**58220.00**	**107.19**
	Horizon (Goal only) [24]	111.39 (0.0%)	21.56 (0.0%)	24.92 (0.0%)	39 (0.0%)	**0.00** (0.0%)	**0.00** (0.0%)	1697.75 (0.0%)	60004.39 (0.0%)	155.51 (0.0%)
	Horizon (Papers) [24]	111.11 (+0.3%)	21.69 (-0.6%)	24.95 (-0.1%)	39 (0.0%)	**0.00** (0.0%)	**0.00** (0.0%)	1697.02 (0.0%)	59946.89 (+0.1%)	155.40 (+0.1%)
	QMCTS-CTS	111.11 (+0.3%)	21.69 (-0.6%)	24.95 (-0.1%)	39 (0.0%)	**0.00** (0.0%)	**0.00** (0.0%)	1697.02 (0.0%)	59946.89 (+0.1%)	155.40 (+0.1%)
	Ours (WL)	111.11 (+0.3%)	21.69 (-0.6%)	24.95 (-0.1%)	39 (0.0%)	**0.00** (0.0%)	**0.00** (0.0%)	1697.02 (0.0%)	59946.89 (+0.1%)	134.00 (+13.8%)
	Ours (Merged)	**109.14** (+2.0%)	**20.03** (+7.1%)	22.25 (+10.7%)	40 (-2.6%)	**0.00** (0.0%)	**0.00** (0.0%)	1689.50 (+0.5%)	59936.02 (+0.1%)	155.21 (+0.2%)
AES-LVT	OpenROAD baseline [1]	97.32	21.75	24.03	38	-7.64	-24.62	1604.84	56626.10	178.94
	iCTS [10]	**93.09**	24.31	**1.84**	**18**	-114.49	-12.9k	**1585.00**	**56564.00**	**128.82**
	Horizon (Goal only) [24]	97.32 (0.0%)	21.75 (0.0%)	24.03 (0.0%)	38 (0.0%)	-7.64 (0.0%)	-24.62 (0.0%)	1604.84 (0.0%)	56626.10 (0.0%)	178.94 (0.0%)
	Horizon (Papers) [24]	97.32 (0.0%)	21.75 (0.0%)	24.03 (0.0%)	38 (0.0%)	-7.64 (0.0%)	-24.62 (0.0%)	1604.84 (0.0%)	56626.10 (0.0%)	178.94 (0.0%)
	QMCTS-CTS	97.32 (0.0%)	21.75 (0.0%)	24.03 (0.0%)	38 (0.0%)	-7.64 (0.0%)	-24.62 (0.0%)	1604.84 (0.0%)	56626.10 (0.0%)	178.94 (0.0%)
	Ours (WL)	97.31 (0.0%)	21.04 (+3.3%)	24.03 (0.0%)	38 (0.0%)	-5.45 (+28.6%)	-14.15 (+42.5%)	1605.02 (0.0%)	56674.61 (-0.1%)	179.12 (-0.1%)
	Ours (Merged)	97.17 (+0.2%)	**17.28** (+20.6%)	26.86 (-11.8%)	41 (-7.9%)	**-5.39** (+29.5%)	**-7.12** (+71.1%)	1607.94 (-0.2%)	56659.31 (-0.1%)	179.12 (-0.1%)
GCD	OpenROAD baseline [1]	39.68	3.32	1.11	5	-83.66	-1453.40	56.99	1131.38	1.24
	iCTS [10]	**36.26**	6.74	**0.20**	2	-238.79	-6.2k	**54.00**	**971.00**	**0.82**
	Horizon (Goal only) [24]	39.68 (0.0%)	3.32 (0.0%)	1.11 (0.0%)	5 (0.0%)	-83.66 (0.0%)	-1453.40 (0.0%)	56.99 (0.0%)	1131.38 (0.0%)	1.24 (+0.1%)
	Horizon (Papers) [24]	39.68 (0.0%)	3.32 (0.0%)	1.11 (0.0%)	5 (0.0%)	-83.66 (0.0%)	-1453.40 (0.0%)	56.99 (0.0%)	1131.38 (0.0%)	1.24 (0.0%)
	QMCTS-CTS	39.68 (0.0%)	3.32 (0.0%)	1.11 (0.0%)	5 (0.0%)	-83.66 (0.0%)	-1453.40 (0.0%)	56.99 (0.0%)	1131.38 (0.0%)	1.24 (0.0%)
	Ours (WL)	39.89 (-0.5%)	**2.70** (+18.7%)	0.66 (+40.8%)	5 (0.0%)	-85.74 (-2.5%)	**-1344.72** (+7.5%)	57.39 (-0.7%)	1278.32 (-13.0%)	1.14 (+8.2%)
	Ours (Merged)	39.71 (-0.1%)	3.32 (0.0%)	1.11 (0.0%)	5 (0.0%)	**-79.65** (+4.8%)	-1353.59 (+6.9%)	56.99 (0.0%)	1114.13 (+1.5%)	1.24 (+0.2%)
JPEG	OpenROAD baseline [1]	226.72	**29.33**	233.56	524	**0.00**	**0.00**	6691.20	156551.59	131.96
	iCTS [10]	236.36	62.39	**15.51**	**152**	-227.98	-92.3k	**6473.00**	157471.00	**92.15**
	Horizon (Goal only) [24]	226.72 (0.0%)	**29.33** (0.0%)	233.56 (0.0%)	524 (0.0%)	**0.00** (0.0%)	**0.00** (0.0%)	6691.20 (0.0%)	156551.59 (0.0%)	131.96 (0.0%)
	Horizon (Papers) [24]	226.72 (0.0%)	**29.33** (0.0%)	233.56 (0.0%)	524 (0.0%)	**0.00** (0.0%)	**0.00** (0.0%)	6691.20 (0.0%)	156551.59 (0.0%)	131.96 (0.0%)
	QMCTS-CTS	226.72 (0.0%)	**29.33** (0.0%)	233.56 (0.0%)	524 (0.0%)	**0.00** (0.0%)	**0.00** (0.0%)	6691.20 (0.0%)	156551.59 (0.0%)	131.96 (0.0%)
	Ours (WL)	227.02 (-0.1%)	37.21 (-26.9%)	233.57 (0.0%)	524 (0.0%)	**0.00** (0.0%)	**0.00** (0.0%)	6691.21 (0.0%)	**156111.28** (+0.1%)	122.00 (+7.5%)
	Ours (Merged)	**226.62** (0.0%)	32.31 (-10.2%)	233.70 (-0.1%)	525 (-0.2%)	**0.00** (0.0%)	**0.00** (0.0%)	6691.35 (0.0%)	156438.15 (+0.1%)	131.93 (0.0%)
JPEG-LVT	OpenROAD baseline [1]	241.53	38.05	154.34	269	**0.00**	**0.00**	6541.65	165095.68	158.49
	iCTS [10]	251.20	50.93	**15.51**	**152**	-181.64	-49.7k	**6411.00**	168391.00	**115.63**
	Horizon (Goal only) [24]	241.53 (0.0%)	38.05 (0.0%)	154.34 (0.0%)	269 (0.0%)	**0.00** (0.0%)	**0.00** (0.0%)	6541.65 (0.0%)	165095.68 (0.0%)	158.49 (0.0%)
	Horizon (Papers) [24]	241.53 (0.0%)	38.05 (0.0%)	154.75 (-0.3%)	269 (0.0%)	**0.00** (0.0%)	**0.00** (0.0%)	6542.06 (0.0%)	165147.16 (0.0%)	158.49 (0.0%)
	QMCTS-CTS	241.53 (0.0%)	38.05 (0.0%)	154.34 (0.0%)	269 (0.0%)	**0.00** (0.0%)	**0.00** (0.0%)	6541.65 (0.0%)	165095.68 (0.0%)	158.49 (0.0%)
	Ours (WL)	**213.23** (+11.7%)	**26.73** (+29.7%)	119.75 (+22.4%)	208 (+22.7%)	**0.00** (0.0%)	**0.00** (0.0%)	6506.50 (+0.5%)	**164230.00** (+0.5%)	156.28 (+1.4%)
	Ours (Merged)	241.53 (0.0%)	38.05 (0.0%)	154.34 (0.0%)	269 (0.0%)	**0.00** (0.0%)	**0.00** (0.0%)	6541.65 (0.0%)	165095.68 (0.0%)	158.49 (0.0%)
RISC-V	OpenROAD baseline [1]	173.30	52.57	56.10	122	**0.00**	**0.00**	2654.07	53156.00	30.94
	iCTS [10]	**145.57**	**50.16**	**3.37**	**33**	-388.04	-367.9k	**1194.00**	**45913.00**	**13.05**
	Horizon (Goal only) [24]	173.30 (0.0%)	52.57 (0.0%)	56.10 (0.0%)	122 (0.0%)	**0.00** (0.0%)	**0.00** (0.0%)	2654.07 (0.0%)	53156.00 (0.0%)	30.90 (+0.1%)
	Horizon (Papers) [24]	173.30 (0.0%)	52.71 (-0.3%)	56.29 (-0.3%)	124 (-1.6%)	**0.00** (0.0%)	**0.00** (0.0%)	2655.09 (0.0%)	53188.00 (-0.1%)	30.95 (0.0%)
	QMCTS-CTS	173.30 (0.0%)	52.42 (+0.3%)	55.78 (+0.6%)	122 (0.0%)	**0.00** (0.0%)	**0.00** (0.0%)	2653.55 (0.0%)	53114.00 (+0.1%)	30.92 (+0.1%)
	Ours (WL)	168.77 (+2.6%)	52.49 (+0.2%)	55.90 (+0.4%)	122 (0.0%)	**0.00** (0.0%)	**0.00** (0.0%)	2653.67 (0.0%)	53055.00 (+0.2%)	28.30 (+8.5%)
	Ours (Merged)	173.30 (0.0%)	52.42 (+0.3%)	55.78 (+0.6%)	122 (0.0%)	**0.00** (0.0%)	**0.00** (0.0%)	2653.55 (0.0%)	53114.00 (+0.1%)	30.92 (+0.1%)
UART	OpenROAD baseline [1]	71.15	**4.27**	1.94	9	-33.47	-1179.14	101.45	1627.49	2.54
	iCTS [10]	**44.81**	7.24	**0.31**	**3**	-48.36	-1.7k	**93.00**	**1477.00**	2.26
	Horizon (Goal only) [24]	71.15 (0.0%)	**4.27** (0.0%)	1.94 (0.0%)	9 (0.0%)	-33.47 (0.0%)	-1179.14 (0.0%)	101.45 (0.0%)	1627.49 (0.0%)	3.34 (-31.4%)
	Horizon (Papers) [24]	71.15 (0.0%)	**4.27** (0.0%)	1.94 (0.0%)	9 (0.0%)	-33.47 (0.0%)	-1179.14 (0.0%)	101.45 (0.0%)	1627.49 (0.0%)	2.54 (0.0%)
	QMCTS-CTS	91.69 (-28.9%)	5.60 (-31.1%)	3.89 (-100.8%)	19 (-111.1%)	-28.19 (+15.8%)	-984.94 (+16.5%)	103.15 (-1.7%)	1671.18 (-2.7%)	2.67 (-5.1%)
	Ours (WL)	91.70 (-28.9%)	5.30 (-24.1%)	3.02 (-55.6%)	13 (-44.4%)	-29.03 (+13.2%)	-947.65 (+19.6%)	101.00 (+0.4%)	1643.74 (-1.0%)	2.59 (-1.9%)
	Ours (Merged)	91.40 (-28.5%)	4.97 (-16.4%)	2.83 (-45.9%)	14 (-55.6%)	**-24.64** (+26.4%)	**-822.04** (+30.3%)	101.19 (+0.3%)	1638.73 (-0.7%)	2.59 (-1.9%)
Avg.	OpenROAD baseline [1]	137.47	**24.00**	60.35	120.3	-15.06	-299.07	2983.25	65656.47	90.88
	iCTS [10]	**127.02**	30.44	**4.69**	**46.0**	-172.97	-64.0k	**2196.33**	**64219.00**	**64.22**
	Horizon (Goal only) [24]	137.47 (0.0%)	**24.00** (0.0%)	60.35 (0.0%)	120.3 (0.0%)	-15.06 (0.0%)	-299.07 (0.0%)	2983.25 (0.0%)	65656.47 (0.0%)	90.92 (0.0%)
	Horizon (Papers) [24]	137.68 (-0.1%)	24.16 (-0.7%)	60.44 (-0.1%)	120.7 (-0.3%)	-13.86 (+7.9%)	-297.74 (+0.4%)	2983.32 (0.0%)	65659.24 (0.0%)	90.88 (0.0%)
	QMCTS-CTS	139.96 (-1.8%)	24.28 (-1.2%)	60.55 (-0.3%)	121.4 (-0.9%)	-13.74 (+8.7%)	-276.16 (+7.7%)	2983.29 (0.0%)	65650.16 (0.0%)	90.75 (+0.1%)
	Ours (WL)	135.09 (+1.7%)	24.11 (-0.5%)	56.63 (+6.2%)	114.2 (+5.1%)	-13.63 (+9.5%)	-256.73 (+14.2%)	2979.36 (+0.1%)	65552.07 (+0.2%)	84.77 (+6.7%)
	Ours (Merged)	138.53 (-0.8%)	24.24 (-1.0%)	60.49 (-0.2%)	121.4 (-0.9%)	**-12.67** (+15.9%)	**-244.38** (+18.3%)	2982.66 (0.0%)	65626.61 (0.0%)	90.87 (0.0%)

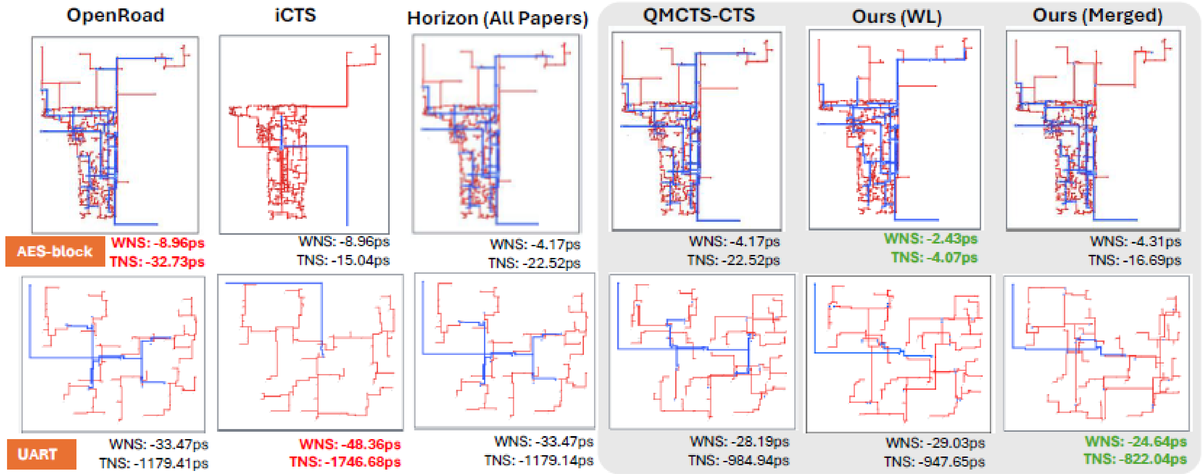


Figure 2: CTS topology extracted from post-route databases on AES-block and UART, the designs with the largest timing gains. The figure illustrates how AutoPD changes the clock-tree structure relative to the baselines. Red lines show leaf fanout branches and blue lines show the top/mid-level CTS trunk.

Ours (Merged) / Ours (WL) vs OpenROAD baseline:

- 15.9% / 9.5% WNS on avg
- 18.3% / 14.2% TNS on avg
- 0.0% / 6.7% mW on avg

Ours (Merged) / Ours (WL) vs HORIZON (Coding Agent):

- 8.6% / 1.7% WNS on avg
- 17.9% / 13.8% TNS on avg
- 0.0% / 6.7% mW on avg

SOTA Algorithms: OpenROAD, iCTS

SOTA Coding Agents: Horizon

Ablation: Q-MCTS with batch eval harness env

Ours (WL): WeakLeaner for each design result

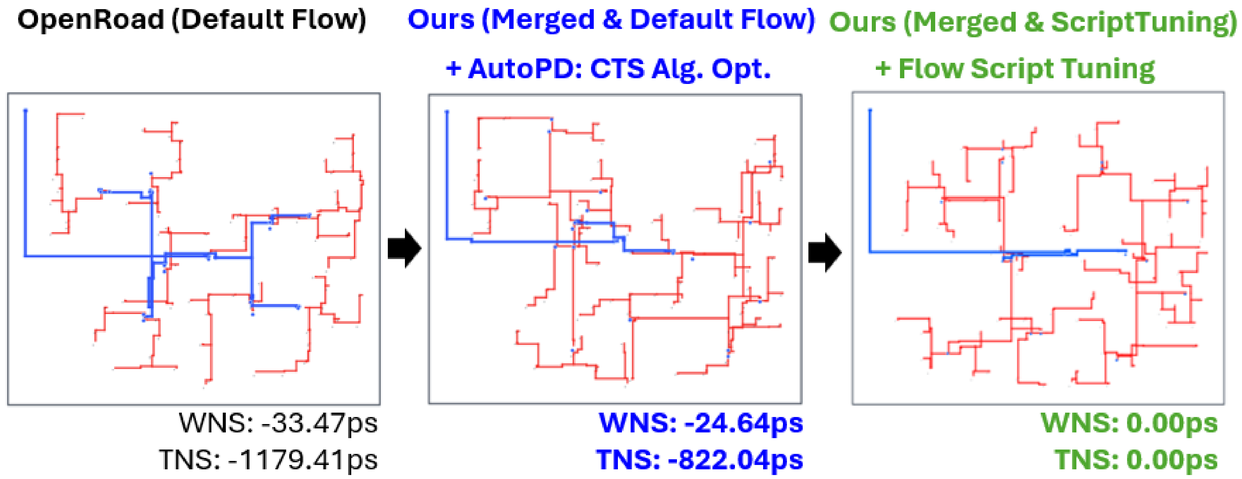
Ours (Merged): Merged algorithm result

NVIDIA

Agentic Tcl Script Flow Tuning Experiment

Table 3: Post-route PPA after agentic Tcl script-based full-flow tuning on the five timing-violating designs. The agent modifies Tcl flow scripts from synthesis through detailed routing (e.g., floorplan density, global placement padding, CTS buffer distance, routing layers) but cannot alter timing constraints or the CTS algorithm.

Design	Flow Script Tuning	CTS Algorithm	WNS (ps)	TNS (ps)	Area (μm^2)	Pwr (mW)
AES	None	OpenROAD	-1.79	-1.79	1638.33	150.45
	✓	OpenROAD	**0.00**	**0.00**	1661.80	152.37
	✓	Ours (Merged)	**0.00**	**0.00**	1668.60	152.66
AES-Blk	None	OpenROAD	-8.96	-32.73	**5862.97**	7.86
	✓	OpenROAD	-9.00	-32.70	**5863.00**	7.86
	✓	Ours (Merged)	**-3.00**	**-10.10**	5863.90	7.81
AES-LVT	None	OpenROAD	-7.64	-24.62	**1604.84**	**178.94**
	✓	OpenROAD	**0.00**	**0.00**	1611.70	180.38
	✓	Ours (Merged)	**0.00**	**0.00**	1631.20	184.08
GCD	None	OpenROAD	-83.66	-1453.40	56.99	1.24
	✓	OpenROAD	**-38.50**	**-503.50**	59.50	2.43
	✓	Ours (Merged)	-54.40	-986.90	**56.10**	**1.20**
UART	None	OpenROAD	-33.47	-1179.14	101.45	**1.54**
	✓	OpenROAD	-21.60	-725.40	99.40	2.48
	✓	Ours (Merged)	**0.00**	**0.00**	**84.90**	2.46
Avg.	None	OpenROAD	-27.10	-538.34	1852.92	68.01
	✓	OpenROAD	-13.82	-252.32	1859.08	69.10
	✓	Ours (Merged)	**-11.48**	**-199.40**	1860.94	69.64



Script Tuning on Ours (Merged) vs. Untuned Baseline

- WNS: −27.10 → −11.48 ps (+57.6%)
- TNS: −538.34 → −199.40 ps (+63.0%)

Script Tuning on Ours (Merged) vs. Script Tuning on OpenROAD CTS

- WNS: −13.82 → −11.48 ps (+16.9%)
- TNS: −252.32 → −199.40 ps (+21.0%)

NVIDIA.

Conclusion & Future Works

- **Self-evolving agents improve how they solve tasks:** Polymath evolves prompts and multi-agent workflows through reflection and evaluation, reducing reliance on fixed, manually engineered systems.
- **Self-learning agents retain and reuse experience:** FVRuleLearner turns corrected reasoning traces into reusable rules for SVA generation. Learning from prior attempts—not simply spending more tokens—is key to improving functional correctness.
- **Automated research advances from execution to discovery:** AutoPD combines research-paper knowledge, question-driven search, and post-route feedback to develop CTS algorithm changes. Algorithm optimization and Tcl flow tuning are complementary, but gains must be checked across designs and PPA objectives.

- Agentic AI for chip design is progressing from hand-crafted workflows to self-evolving agents, reusable learned knowledge, and automated EDA research.
- Domain expertise and tool-based harness are essential.

- **Future directions:**
  - EDA Harness Env for Agentic AI, and Agentic AI Evolving for EDA
  - EDA Flow & Algorithm Co-Optimization for Design PPA
  - Optimization with EDA Algorithm noise?
  - EDA Algorithm optimization across various stages?

ACKNOWLEDGEMENT

Many thanks to production teams, internal collaborators, and external collaborators on this presentation

Cunxi Yu, Chenhui Deng, Rongjian Liang, Haoyu Yang, Yunsheng Bai, Haoxing (Mark) Ren, Brucek Khailany

Chen-Chia Chang, Jatin Nainani, Jing Gong, Abishek Akkur, All collaborating members

Lily Wan, Prof. Deming Chen (UIUC)

Zhan Song, Prof. Cunxi Yu (UMD)

Souradip Poddar, Prof. David Pan (UT Ausin)

Hejia Zhang, Yujie Zhao, Zhongming Yu, Prof. Jishen Zhao (UCSD)

Bing-Yue Wu, Prof. Vidya Chhabria (ASU)

Reference

[1] Liu, Mingjie, Teodor-Dumitru Ene, Robert Kirby, Chris Cheng, Nathaniel Pinckney, Rongjian Liang, Jonah Alben et al. “Chipnemo: Domain-adapted llms for chip design." *arXiv preprint arXiv:2311.00176* (2023).

[2] Tsai, YunDa, Mingjie Liu, and Haoxing Ren. "Rtlfixer: Automatically fixing rtl syntax errors with large language model." In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, pp. 1-6. 2024.

[3] Yao, X., Wang, Y., Li, X., Lian, Y., Chen, R., Chen, L., ... & Yu, B. (2024, October). Rtlrewriter: Methodologies for large models aided rtl code optimization. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design* (pp. 1-7).

[4] Ho, Chia-Tung, and Haoxing Ren. "Large language model (LLM) for standard cell layout design optimization." In *2024 IEEE LLM Aided Design Workshop (LAD)*, pp. 1-6. IEEE, 2024.

[5] Deng, C., Tsai, Y. D., Liu, G. T., Yu, Z., & Ren, H. (2025, September). Scalertl: Scaling llms with reasoning data and test-time compute for accurate rtl code generation. In *2025 ACM/IEEE 7th Symposium on Machine Learning for CAD (MLCAD)* (pp. 1-9). IEEE.

[6] Zhao, Yang, Di Huang, Chongxiao Li, Pengwei Jin, Muxin Song, Yinan Xu, Ziyuan Nan et al. "Codev: Empowering llms with hdl generation through multi-level summarization." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2025).

[7] Wei Li, Yang Zou, Christopher Ellis, Ruben Purdy, Shawn Blanton and José Moura, “BRIDGE: Bridging Graph and Large Language Models in EDA”, IEEE International Conference on LLM-Aided Design (ICLAD), 2025.

[8] Deng, C., Bai, Y., & Ren, H. (2025, June). Chipalign: Instruction alignment in large language models for chip design via geodesic interpolation. In *2025 62nd ACM/IEEE Design Automation Conference (DAC)* (pp. 1-7). IEEE.

[9] Zhao, Yujie, Zhijing Wu, Boqin Yuan, Zhongming Yu, Hejia Zhang, Wentao Ni, Chia-Tung Ho, Haoxing Ren, and Jishen Zhao. "PRO-V-R1: Reasoning Enhanced Programming Agent for RTL Verification." *arXiv preprint arXiv:2506.12200* (2025).

[10] Bai, Yunsheng, Ghaith Bany Hamad, Syed Suhaib, and Haoxing Ren. "Assertionforge: Enhancing formal verification assertion generation with structured representation of specifications and rtl." In *2025 IEEE International Conference on LLM-Aided Design (ICLAD)*, pp. 85-92. IEEE, 2025.

[11] Ho, Chia-Tung, Jing Gong, Yunsheng Bai, Chenhui Deng, Haoxing Ren, and Brucek Khailany. "Marco: Configurable graph-based task solving and multi-AI agents framework for hardware design." In *2025 Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, pp. 1-3. IEEE, 2025.

[12] Nainani, J., Ho, C. T., Dhurka, A., & Ren, H. (2025). Timing Analysis Agent: Autonomous Multi-Corner Multi-Mode (MCMM) Timing Debugging with Timing Debug Relation Graph. *arXiv preprint arXiv:2504.11502*.

[13] Poddar, Souradip, Chia-Tung Ho, Ziming Wei, Weidong Cao, Haoxing Ren, and David Z. Pan. "HeaRT: A Hierarchical Circuit Reasoning Tree-Based Agentic Framework for AMS Design Optimization." *arXiv preprint arXiv:2511.19669* (2025).

[14] Ho, Chia-Tung, Haoxing Ren, and Brucek Khailany. "Verilogcoder: Autonomous verilog coding agents with graph-based planning and abstract syntax tree (ast)-based waveform tracing tool." In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 1, pp. 300-307. 2025.

[15] Chang, Chen-Chia, Chia-Tung Ho, Yaguang Li, Yiran Chen, and Haoxing Ren. "Drc-coder: Automated drc checker code generation using llm autonomous agent." In *Proceedings of the 2025 International Symposium on Physical Design*, pp. 143-151. 2025.

[16] Zhang, Niansong, Chenhui Deng, Johannes Maximilian Kuehn, Chia-Tung Ho, Cunxi Yu, Zhiru Zhang, and Haoxing Ren. "ASPEN: LLM-Guided E-Graph Rewriting for RTL Datapath Optimization." In *2025 ACM/IEEE 7th Symposium on Machine Learning for CAD (MLCAD)*, pp. 1-9. IEEE, 2025.

[17] Kochar, Dimple Vijay, Nathaniel Pinckney, Guan-Ting Liu, Chia-Tung Ho, Chenhui Deng, Haoxing Ren, and Brucek Khailany. "GRPO with State Mutations: Improving LLM-Based Hardware Test Plan Generation." *arXiv preprint arXiv:2601.07593* (2026).

[18] Wan, Lily Jiaxin, Chia-Tung Ho, Rongjian Liang, Cunxi Yu, Deming Chen, and Haoxing Ren. "SCHEMACODER: Automatic Log Schema Extraction Coder with Residual Q-Tree Boosting." *arXiv preprint arXiv:2508.18554* (2025).

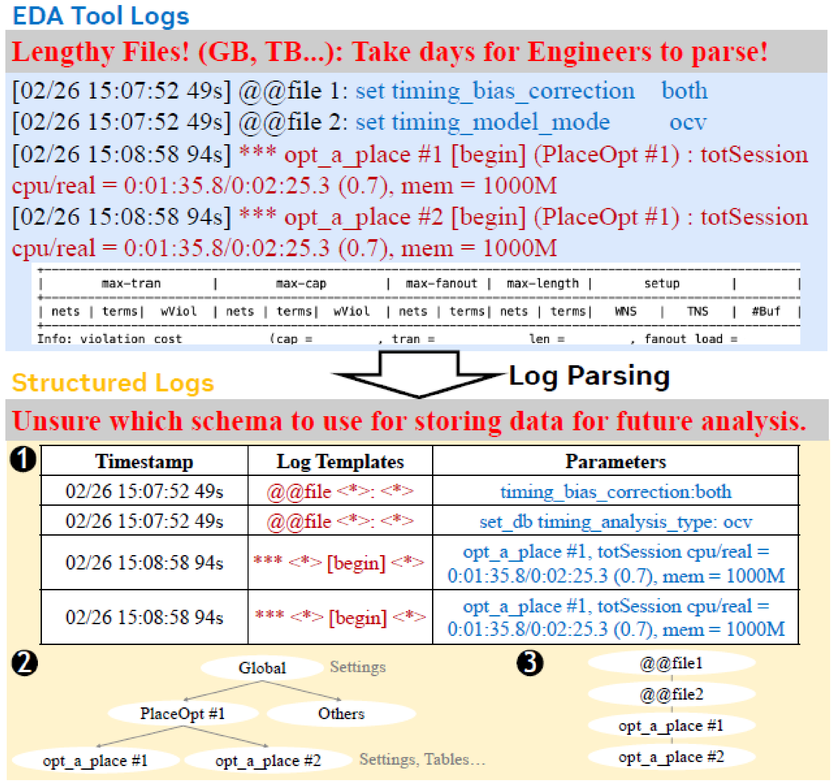
[19] Ho, Chia-Tung, Jing Gong, Xufeng Yao, Yunsheng Bai, Abhishek B. Akkur, and Haoxing Ren. "Polymath: A Self-Optimizing Agent with Dynamic Hierarchical Workflow." *arXiv preprint arXiv:2508.02959* (2025).

[20] Zhao, Yujie, Hejia Zhang, Hanxian Huang, Zhongming Yu, and Jishen Zhao. "Mage: A multi-agent engine for automated rtl code generation." In *2025 62nd ACM/IEEE Design Automation Conference (DAC)*, pp. 1-7. IEEE, 2025.

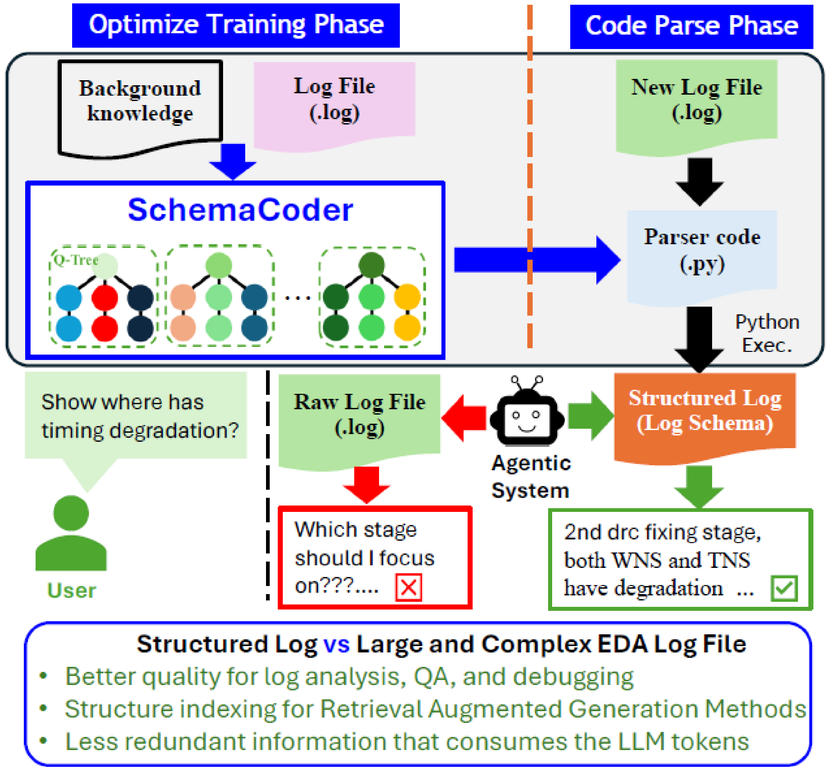
[21] Novikov, Alexander, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov et al. "Alphaevolve: A coding agent for scientific and algorithmic discovery." *arXiv preprint arXiv:2506.13131* (2025).

SchemaCoder [DAC' 26]

- Design Opt/Debugging: Besides RTL, the design artifacts of EDA tool chains are essential.
- Current SoTA commercial agents (i.e., Cursor, Claude Code) perform poorly on the EDA log analysis tasks.



(a) EDA tool logs: Including command outputs, progress messages, and performance tables, logs are partitioned into design Stages (e.g., Opt#1) and parsed to extract parameter settings with their Parameters or Tables.



(b) An illustration of SchemaCoder for EDA tool log file analysis, debug, and QA tasks using raw log file versus structured log. Structured log provides better quality for log analysis, QA, and debugging, and less redundant information.

Training: Residual Q-Tree Boosting

- **Goal:** SchemaCoder **generates code to extract log file schemas** from arbitrary unstructured or structured log files.
- **Input:** Log file and background knowledge (~100-400 words)
- **Output:** Parsing program (.py)

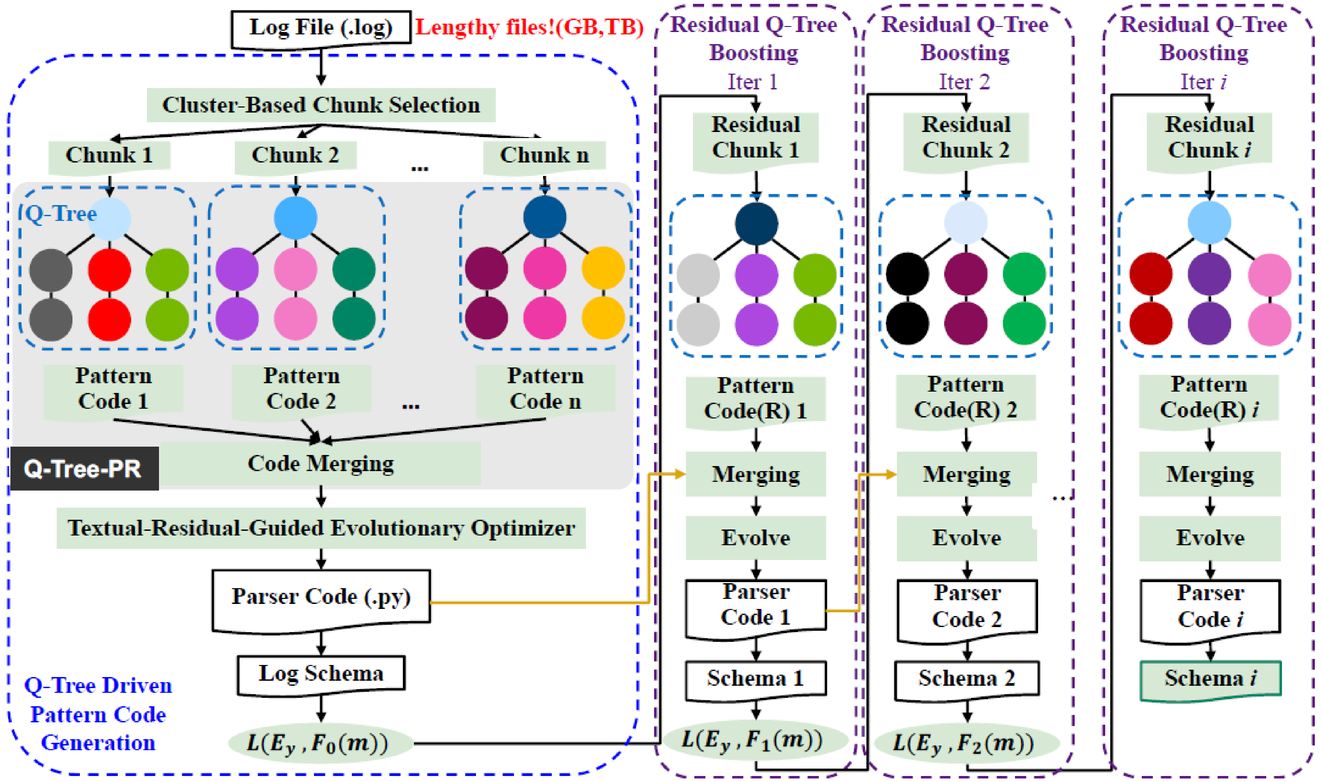
- **Optimization problem**: Iteratively adding weak learners (Q-Tree) that correct the errors of previous iterations by following the negative gradient of the loss function.

$$\hat{F} = \arg\min_{S(b,m)} L\big(E_y, F(m)\big).$$

$$R_i \approx -\frac{\partial L(E_y, F_{i-1}(m))}{\partial F_{i-1}(m)} \qquad \text{(residual chunks)}$$

$$F_i(m) = F_0(m) + \sum_{j=1}^{i} f_j(m) \qquad \text{(code merging)}$$

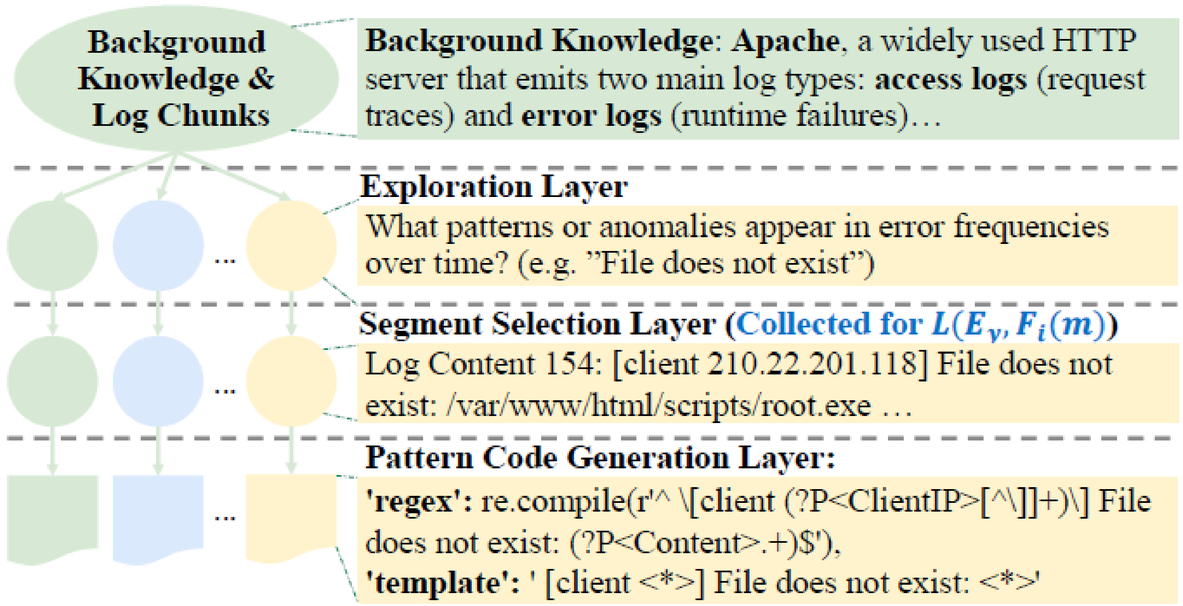
m is the log file content, L is the loss function, F is the parsing code, and $S(b,m)$ represent the methodology to generate the F.



Q-Tree and Evaluation Function

- For log analysis tasks, designers typically aim to identify the root cause of bugs or anomalies by asking questions and examining the corresponding log segments

- Evaluation function is automatically collected from Segment Selection Layer in an unsupervised way
- Gradient provides faster convergence than using AlphaEvolve to optimize the parser code



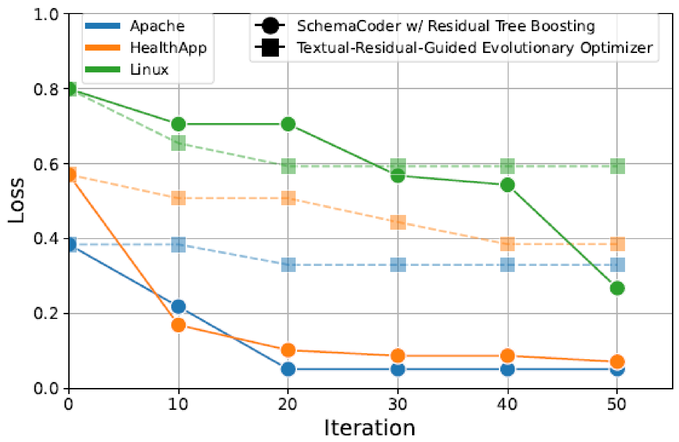


Figure 4: Loss L over iterations i on Apache, HealthApp, and Proxifier logs, comparing textual-residual evolutionary optimization versus residual Q-Tree boosting. Invoked every 10 iterations, Q-Tree boosting significantly accelerates convergence.

Experimental Results

- Achieves an 11.7% and 2.9% improvement on pass@1 and pass@3 across GPT-5 and Sonnet-4.5.
- Achieves >10% accuracy improvement on both pass@1 and pass@3 for complex QAs in CTS and across stages.
- For general logs (LogHub-2.0), 3.3% improvement in the average scores and outperforms SoTA baselines in 9 of 14 applications on LogHub-2.0.

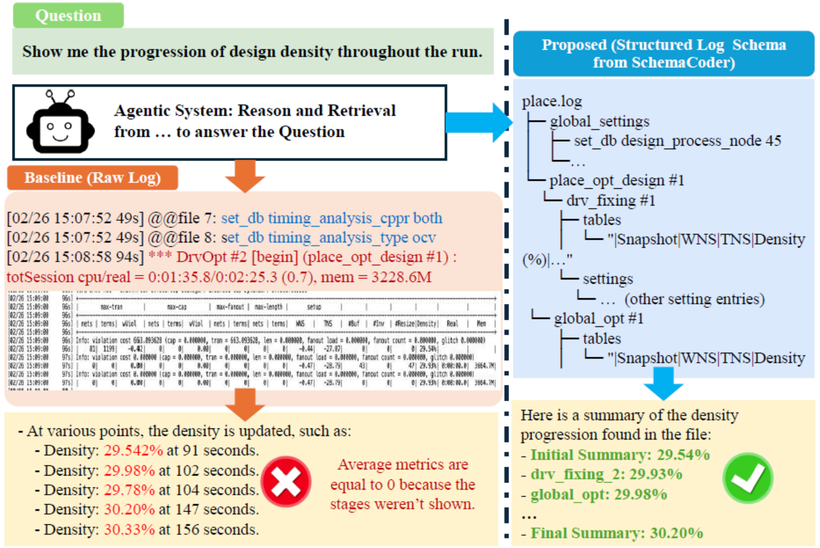


Table 1: Performance comparison of EDA tool log analysis QAs (100 QAs). We input the raw log (baseline), and the generated structured schema (proposed) into Cursor agent [7] (i.e., enterprise version) to complete the QA tasks. We use the GPT-5.1 [27] and Sonnet-4.5 [2] as LLM models in Cursor agent. Each QA is tested 10 times for computing pass@1 and pass@3 metrics. The best score is highlighted in bold.

Log Schema Method	Agent Model	Synthesis		Floorplan		Placement		CTS		Routing		Design-Finish		Across Stages		All (100 QAs)	
		pass@1	pass@3	pass@1	pass@3	pass@1	pass@3	pass@1	pass@3	pass@1	pass@3	pass@1	pass@3	pass@1	pass@3	pass@1	pass@3
Baseline (Raw)	GPT-5.1	80.1%	93.5%	78.3%	93.2%	52.2%	71.3%	56.2%	71.7%	77.5%	88.5%	51.1%	62.5%	40.1%	56.7%	64.7%	81.0%
	Sonnet-4.5	84.7%	97.7%	87.2%	98.0%	64.8%	79.7%	70.4%	84.9%	79.5%	86.9%	58.9%	69.6%	40.9%	66.8%	72.0%	86.3%
SchemaCoder (structured)	GPT-5.1	91.8%	97.2%	86.7%	92.1%	67.7%	81.3%	86.9%	93.1%	**82.7%**	90.5%	53.3%	67.5%	44.4%	72.0%	76.1%	84.6%
	Sonnet-4.5	**98.6%**	**99.7%**	**99.1%**	**100.0%**	**81.5%**	**84.1%**	**91.9%**	**96.0%**	81.1%	**92.2%**	**63.3%**	**72.6%**	**58.5%**	**76.7%**	**84.0%**	**88.5%**
Impr. (%)	GPT-5.1	11.7%	3.7%	8.3%	-1.1%	15.5%	10.0%	30.8%	21.4%	5.2%	2.0%	2.2%	5.0%	4.3%	15.3%	11.3%	3.6%
	Sonnet-4.5	14.0%	2.0%	11.9%	2.0%	16.6%	4.4%	21.6%	11.1%	1.6%	5.4%	4.4%	3.0%	17.6%	9.8%	12.1%	2.2%
Avg Impr. (%)		12.9%	2.9%	10.1%	0.5%	16.1%	7.2%	26.2%	16.3%	3.4%	3.7%	3.3%	4.0%	10.9%	12.6%	11.7%	2.9%