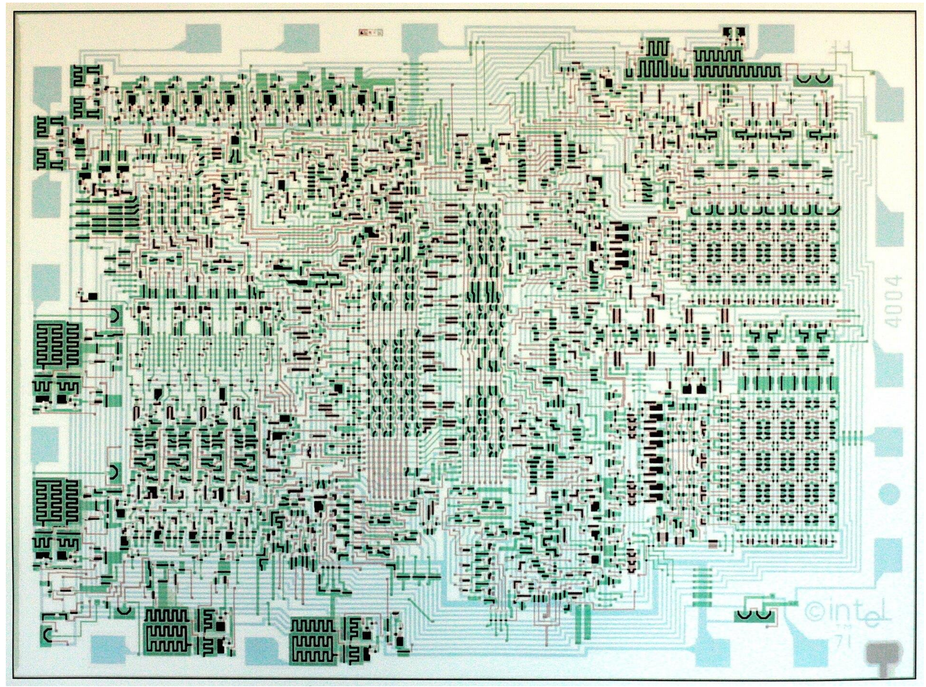
AGENTRYS

Agentic AI for Chip Design

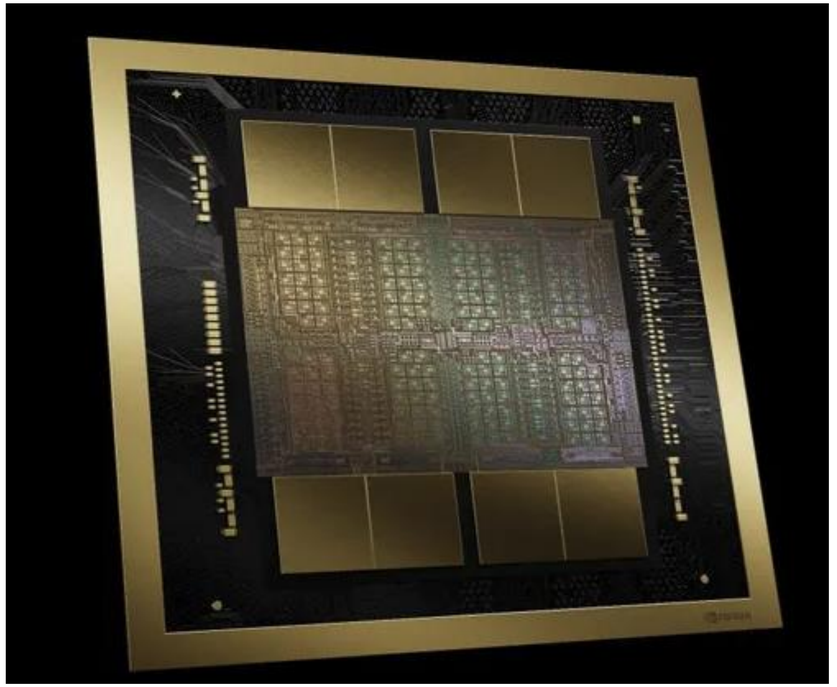
Agentic Design Automation (ADA) toward Autonomous Design

Mark Ren
Founder and CEO

Chip Design: Million X Productivity Improvement in Half Century



Intel 4004 (1971)
2300 transistors
100K 4B instruction/sec



NVIDIA B200 (2024)
208 Billion Transistors
20 Peta FP4/sec

EDA Driving Force: Increasing Automation and Scaling

1973: SPICE, automate circuit simulation

1986: Synopsys Design Compiler, automate logic synthesis

1989: Cadence Verilog-XL, automate RTL simulation

1997: Cadence Silicon Ensemble , automate placement & route

1998: Synopsys PrimeTime , automate static timing analysis

2004: Apache RedHawk, automate power grid analysis, scaling with multi-threading

2008: Synopsys PrimeTime , scaling STA with multi-threading

2012: Cadence Innovus, scaling P&R with extensive multi-threading

2016: Ansys RedHawk-SC, scaling power grid analysis with distributed many cores

AI/LLM Enables More Automation

Generating insights, knowledge, and ideas. Answer questions about designs, infrastructures, tools, flows, HW domains, etc.

Understanding

Generating design and verification code (RTL, SVA, testbenches, EDA scripts, tools scripts, and configs)

Coding

Summarization, report/log analysis, visualization of design and related data, check rule violations, etc.

Analysis

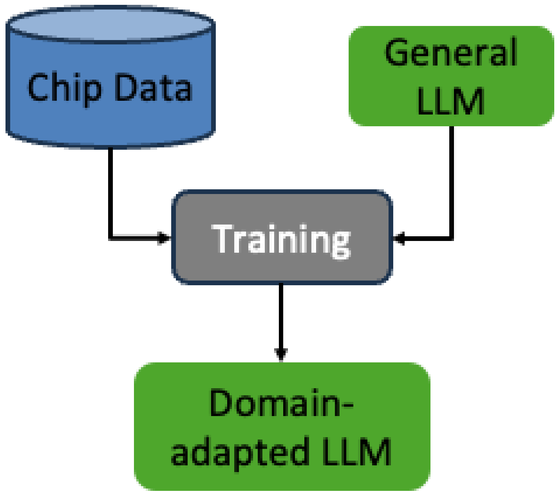
Triage and fix a design problem, e.g. regression failures for logic design and timing failures for physical design.

Debug

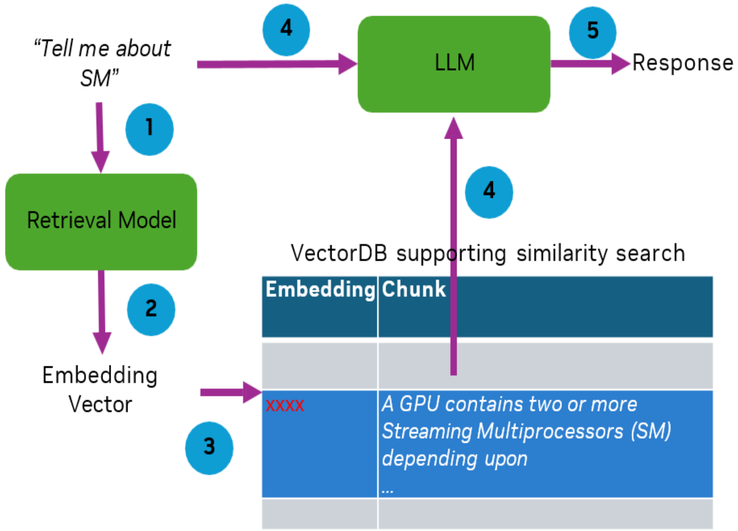
Optimize for design and verification: coverage closure, recipe optimization, architecture exploration, etc.

Optimization

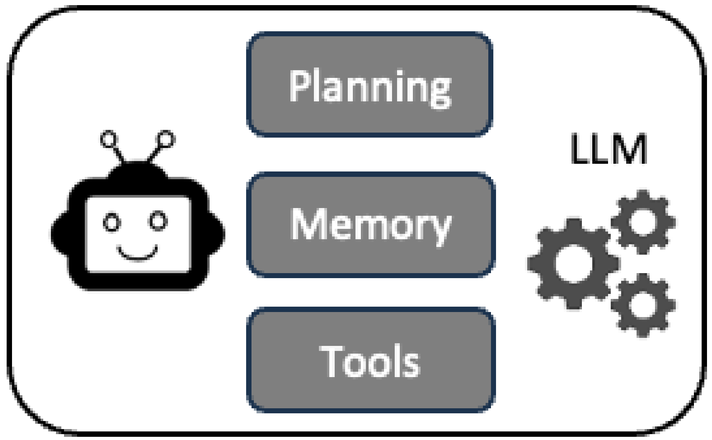
The Progression of LLM Adoption in Chip Design



Parameter Training (ChipNemo)



Prompt engineering, In-Context Learning, RAG (Chatbots)

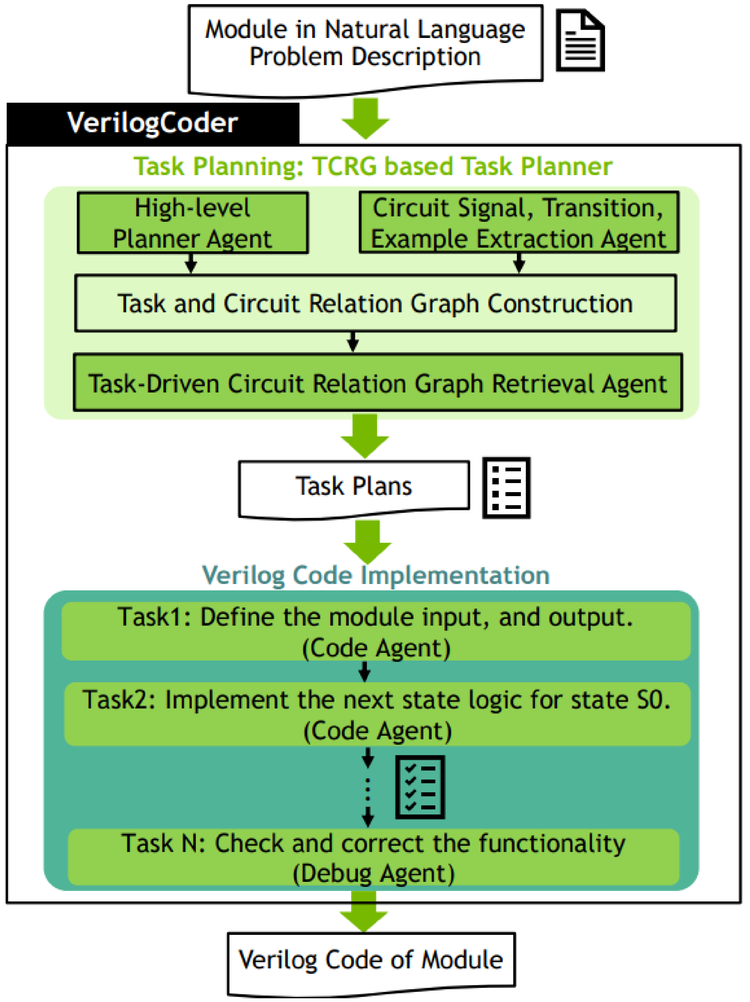


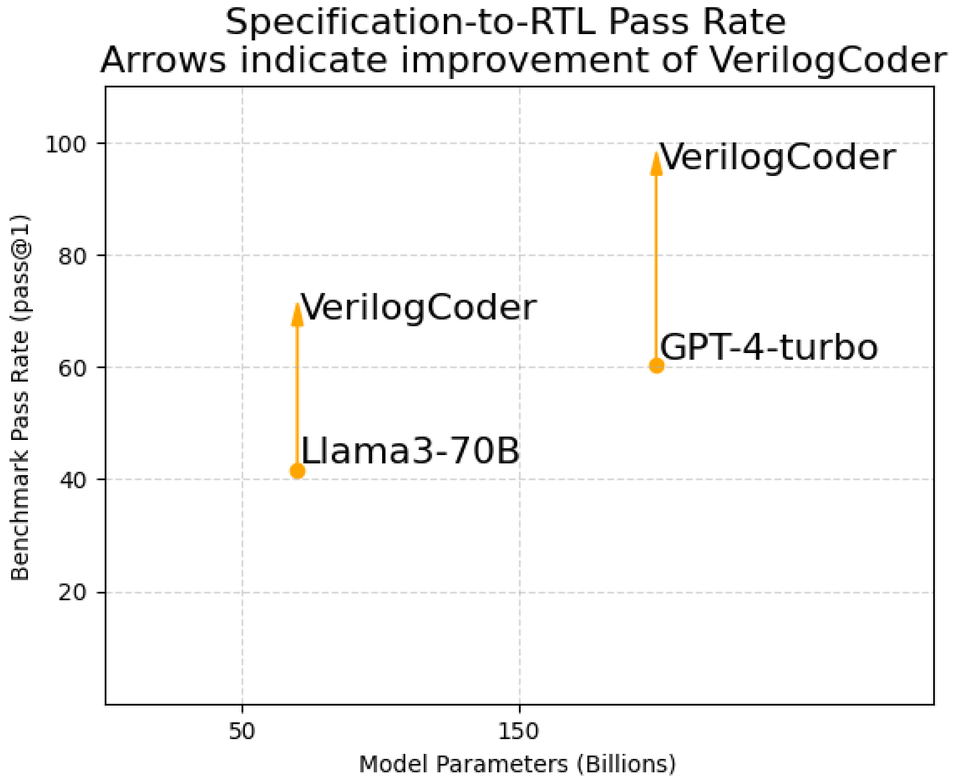
Agent (Claude Code)

2022 → **2026**

Agent is the Game Changer

VerilogCoder: Agent solved most of VerilogEval problems

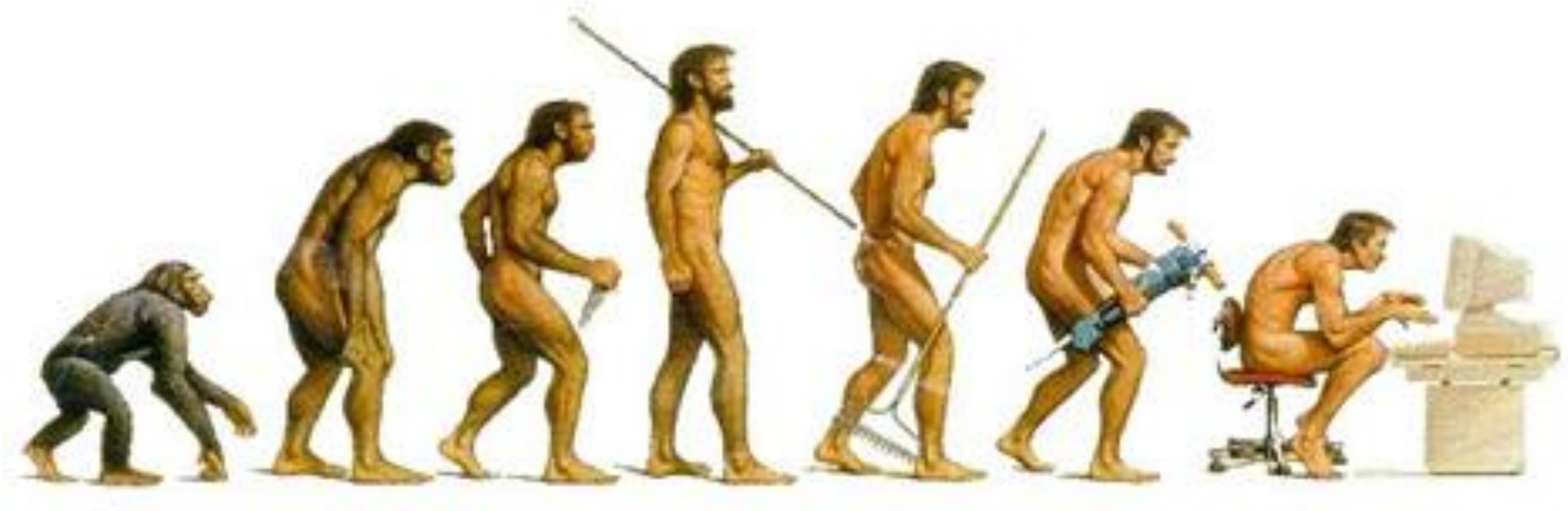




Chia-Tung Ho, et al, VerilogCoder: Autonomous Verilog Coding Agents with Graph-based Planning and Abstract Syntax Tree (AST)-based Waveform Tracing Tool

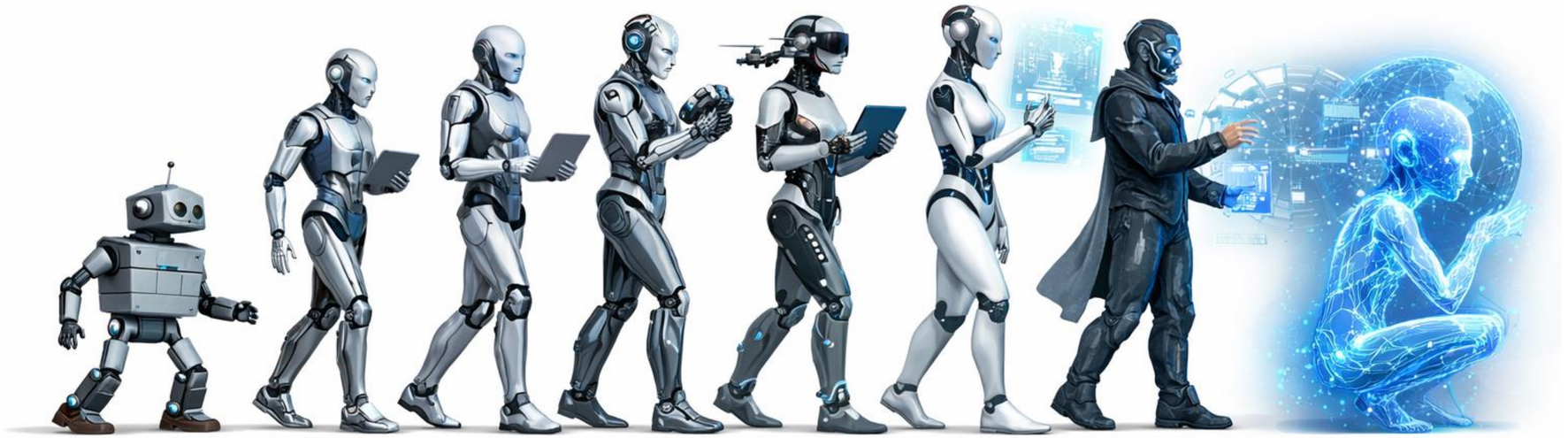
Human Evolution with Tools

Power of human comes from tools we use



Agent Evolution with Tools

Power of Agents will come from tools they use



Agentic Design Automation (ADA)

Scale the Power of AI and Agent with Custom Models and Agent Native Tools

Agents

Custom Models
Specialized
Low cost



Agent-Native Tools
Lightweight
Versatile

Frontier Models
General purpose
Expensive

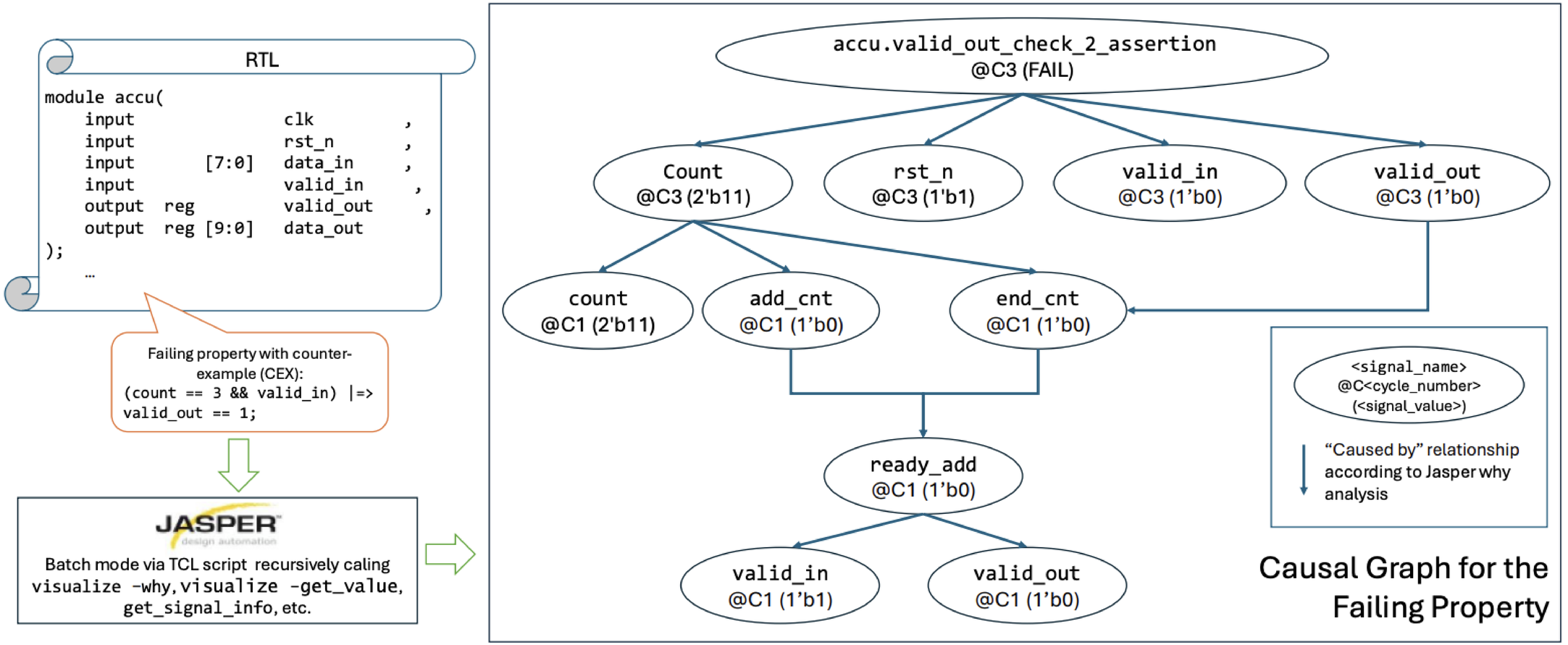


EDA Tools
Heavy-duty
Specialized

Verification Debug

Tools provide context information for debugging

Debug with **causal graph tools** working with EDA tools

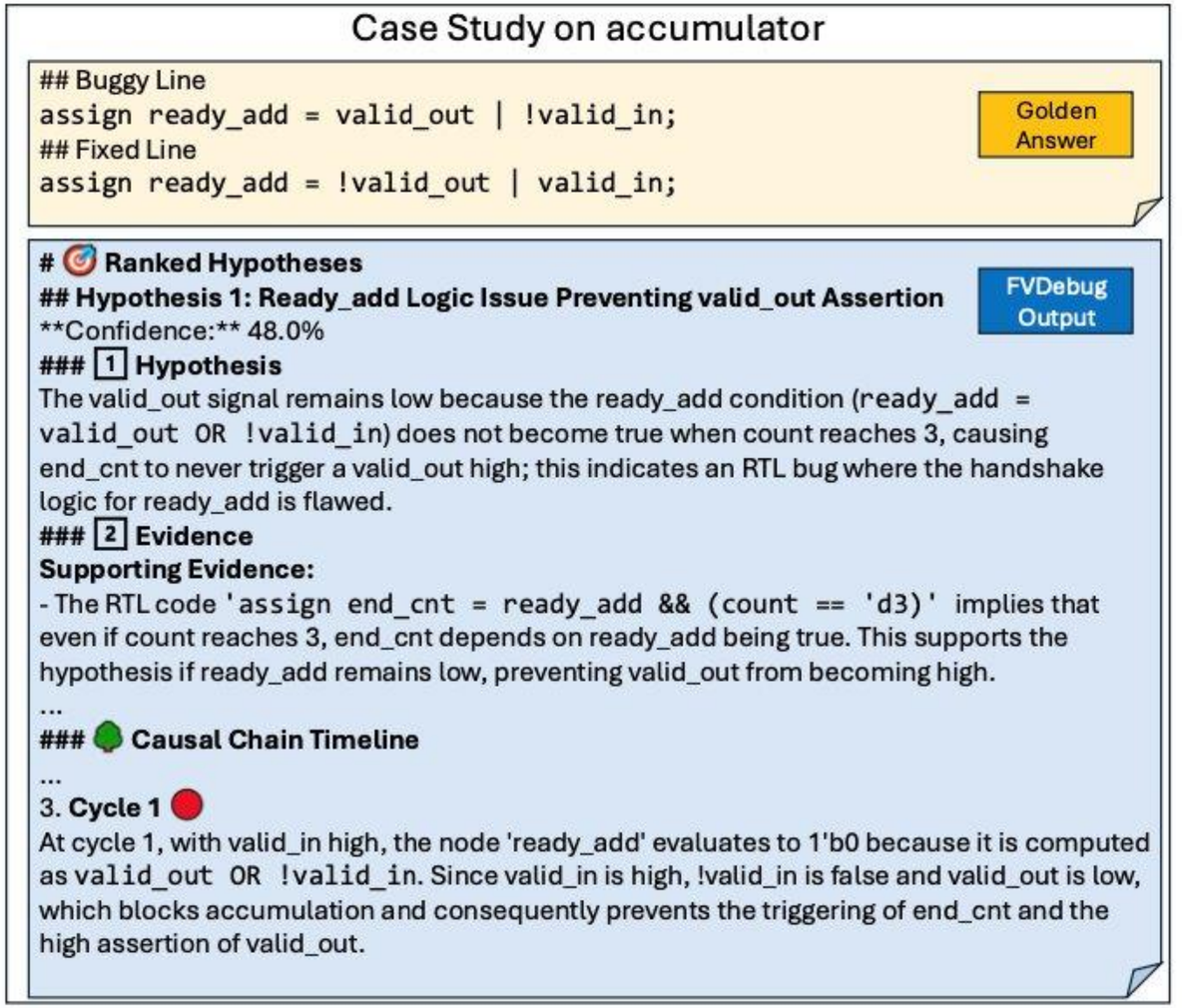


Yunsheng Bai, et al, FVDebug: An LLM-Driven Debugging Assistant for Automated Root Cause Analysis of Formal Verification Failures

Verification Debug

Agent reasons with the causal graph

Agent traverse graph to locate RTL and reasoning about the bug; then propose debug hypothesis

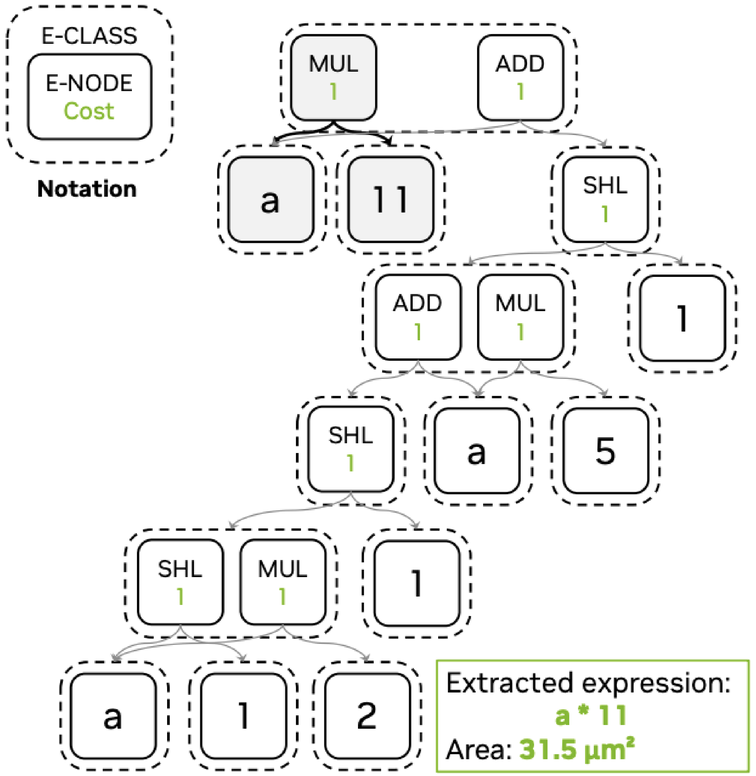


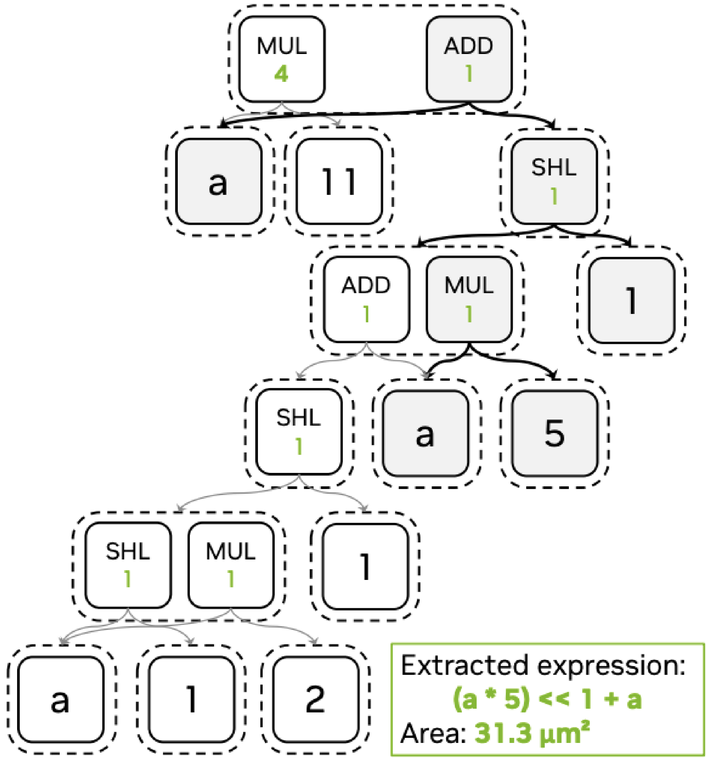
RTL Optimization

Tools conducts optimization

Leverage **e-graph** for correct-by-construction RTL rewriting with E-Graph

E-graph rewriting tools executes the RTL optimization.





Niansong Zhang et al, ASPEN: LLM-Guided E-Graph Rewriting for RTL Datapath Optimization

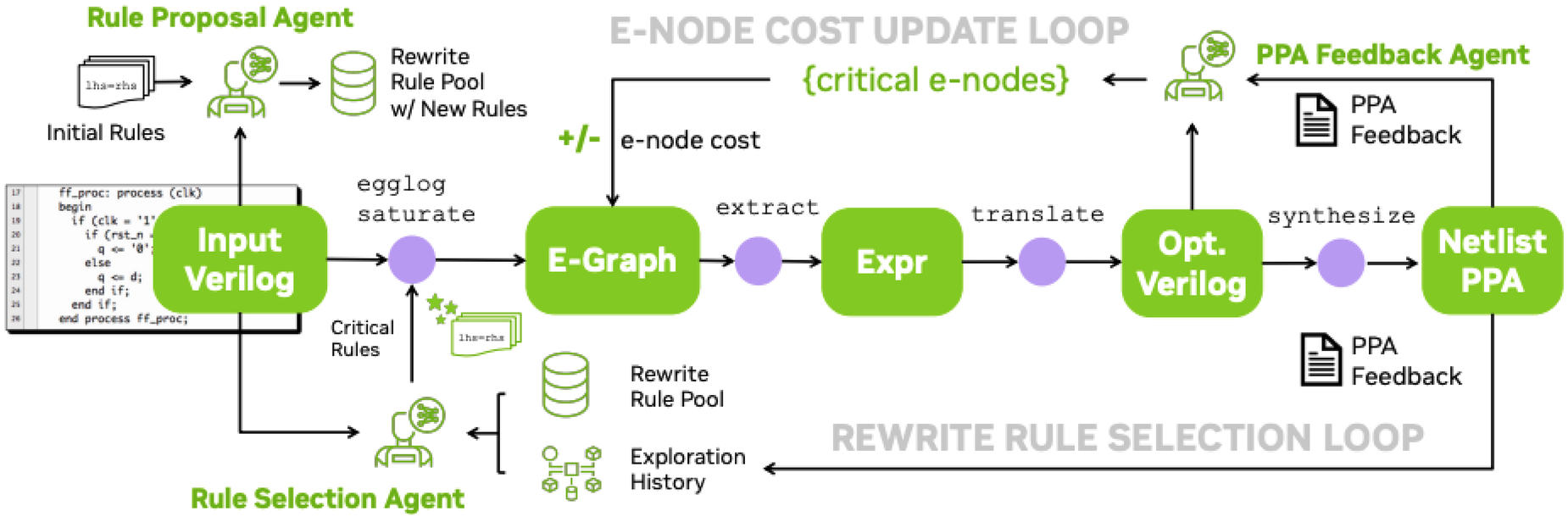
RTL Optimization

Agent drives the e-graph rewrite tools

LLM assisted PPA feedback

LLM assisted rewrite rule selection and generation

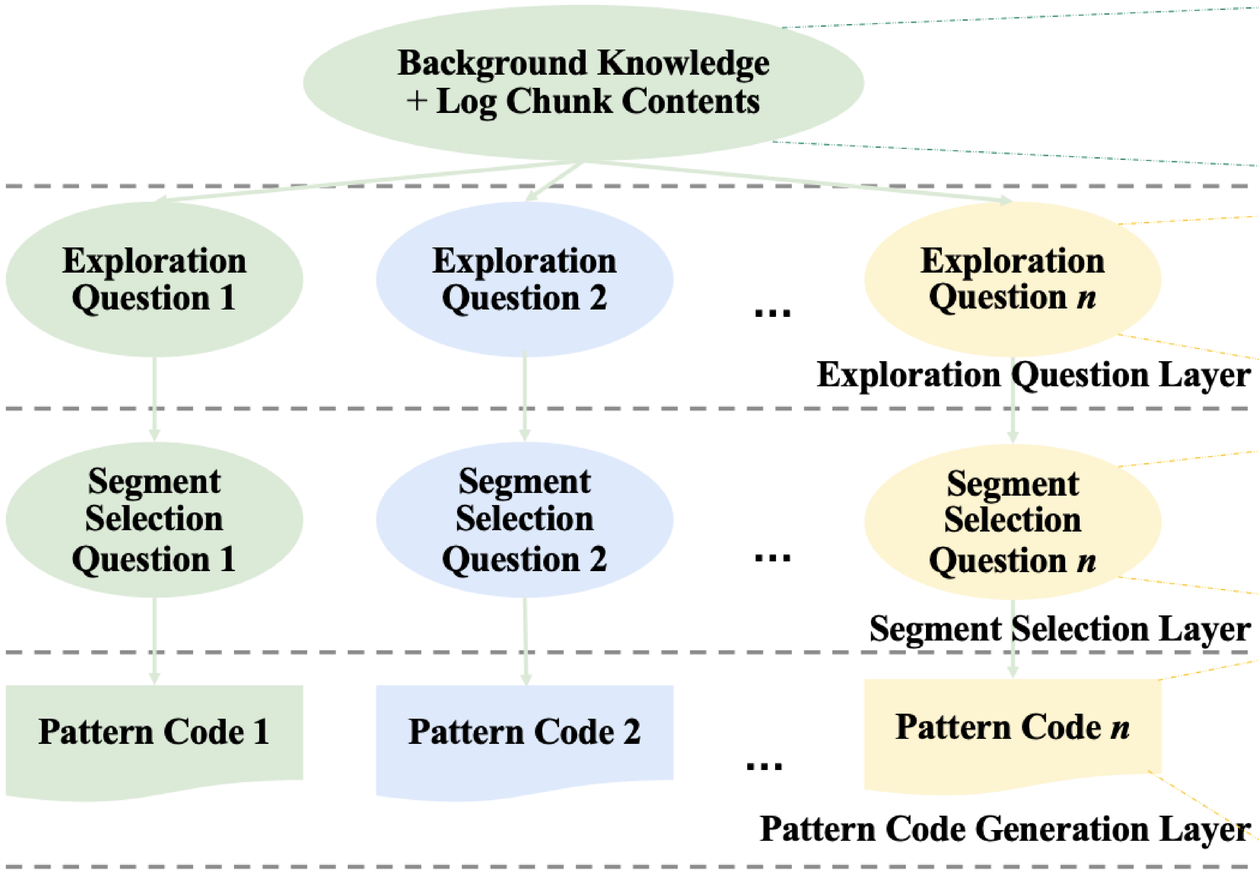
Results: –24.23% area / –12.43% delay vs e-graph only baseline



Niansong Zhang et al, ASPEN: LLM-Guided E-Graph Rewriting for RTL Datapath Optimization

Log Analysis

Agent generates tools/code to extract structure information from log files



Background Knowledge [Our Prompt]: Apache. Apache HTTP Server is one of the most popular web servers. Apache servers usually generate two types of logs: access logs and error logs. …

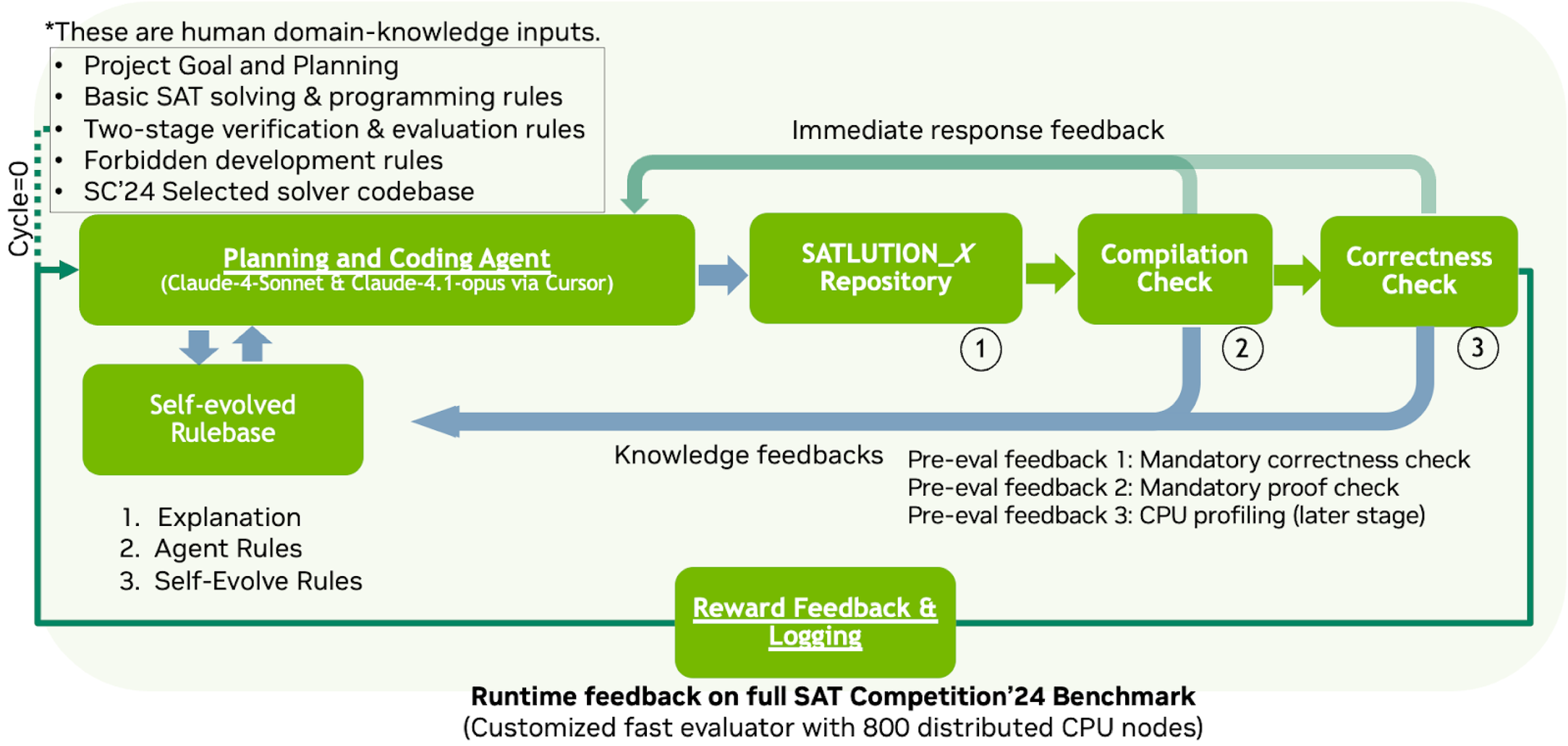
Exploration Questions [Generated by LLMs]:
e.g. What patterns or anomalies can be identified in the frequency and distribution of error types, such as "File does not exist" and "Directory index forbidden by rule," across different timestamps?

Segment Selection Questions [Our Prompt]:
Your Goal: Provide the top lines (might be chunks of the table) in the log file that are most relevant to answering the question.

Pattern Code [Generated by LLMs]:
e.g. pattern = { *'regex':* re.compile(r'^\[(?P<Time>[^\]]+)\] \[(?P<Level>[^\]]+)\] \[client (?P<ClientIP>[^\]]+)\] File does not exist: (?P<Content>.+)$'),
'template': '[<*>] [<*>] [client <*>] File does not exist: <*>'}

Improve SAT Solver

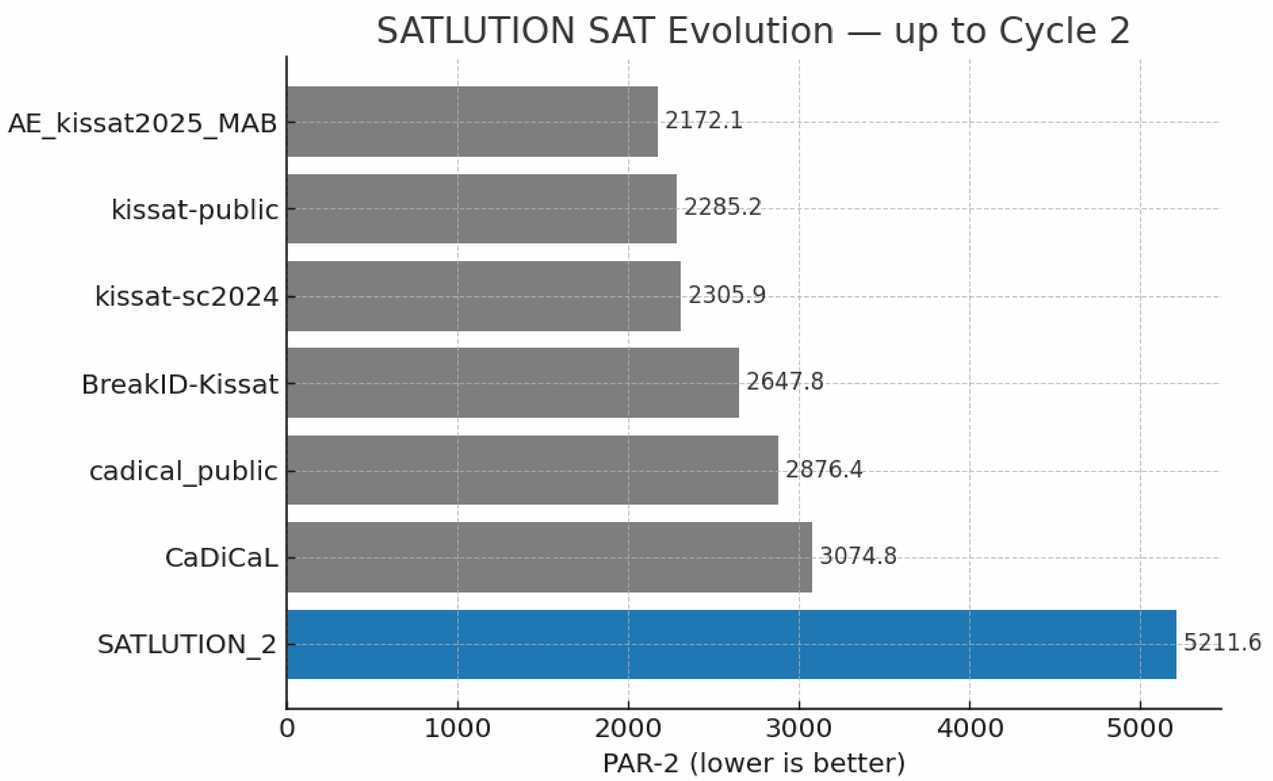
Agent improves existing tools



Runtime feedback on full SAT Competition'24 Benchmark
(Customized fast evaluator with 800 distributed CPU nodes)

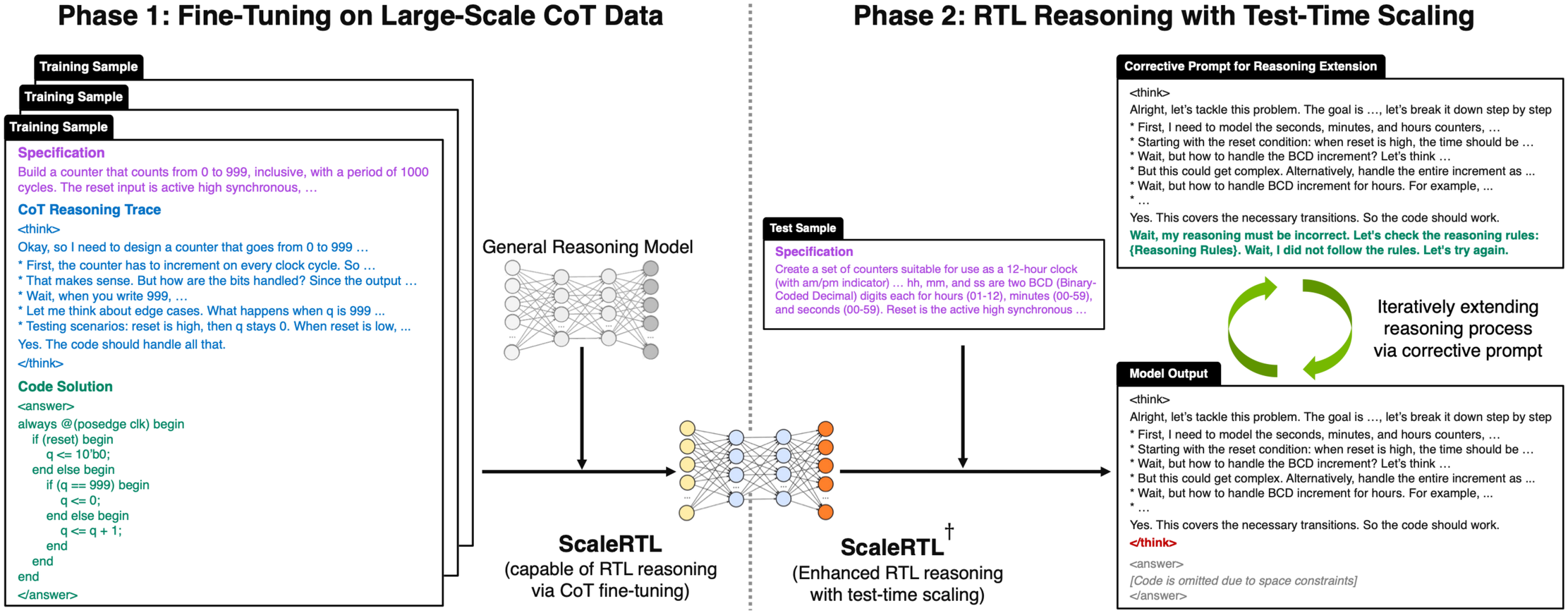
SATLUTION : 10+% better than 2025 SAT Competition winners

Start from the winning solvers in 2024 SAT competition
Improved performance by 18% in a month of evolution



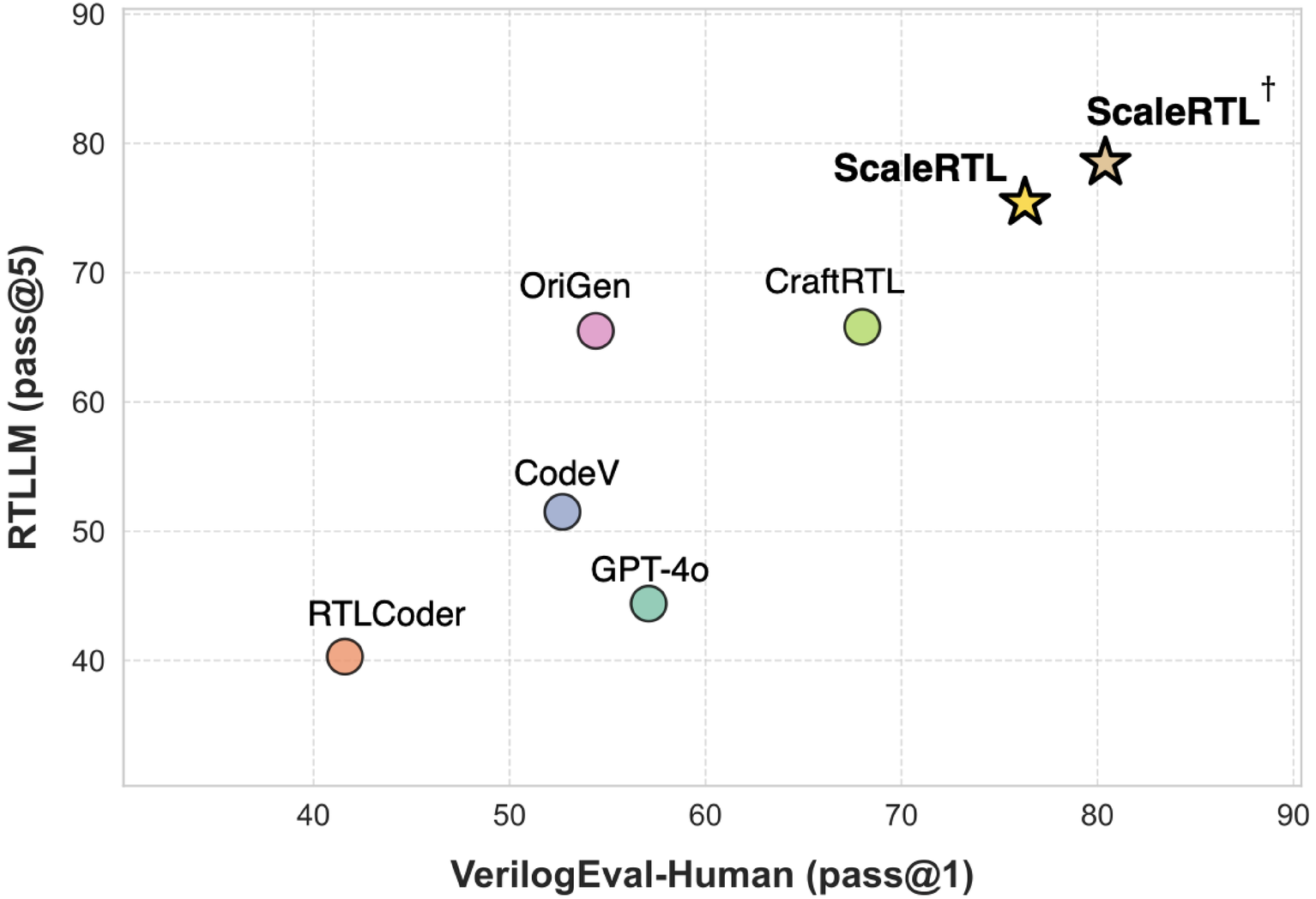
ScaleRTL: Scaling LLMs with Reasoning and Test Time Compute

Train a custom model



Chenhui Deng, et al, ScaleRTL: Scaling LLMs with Reasoning Data and Test-Time Compute for Accurate RTL Code Generation

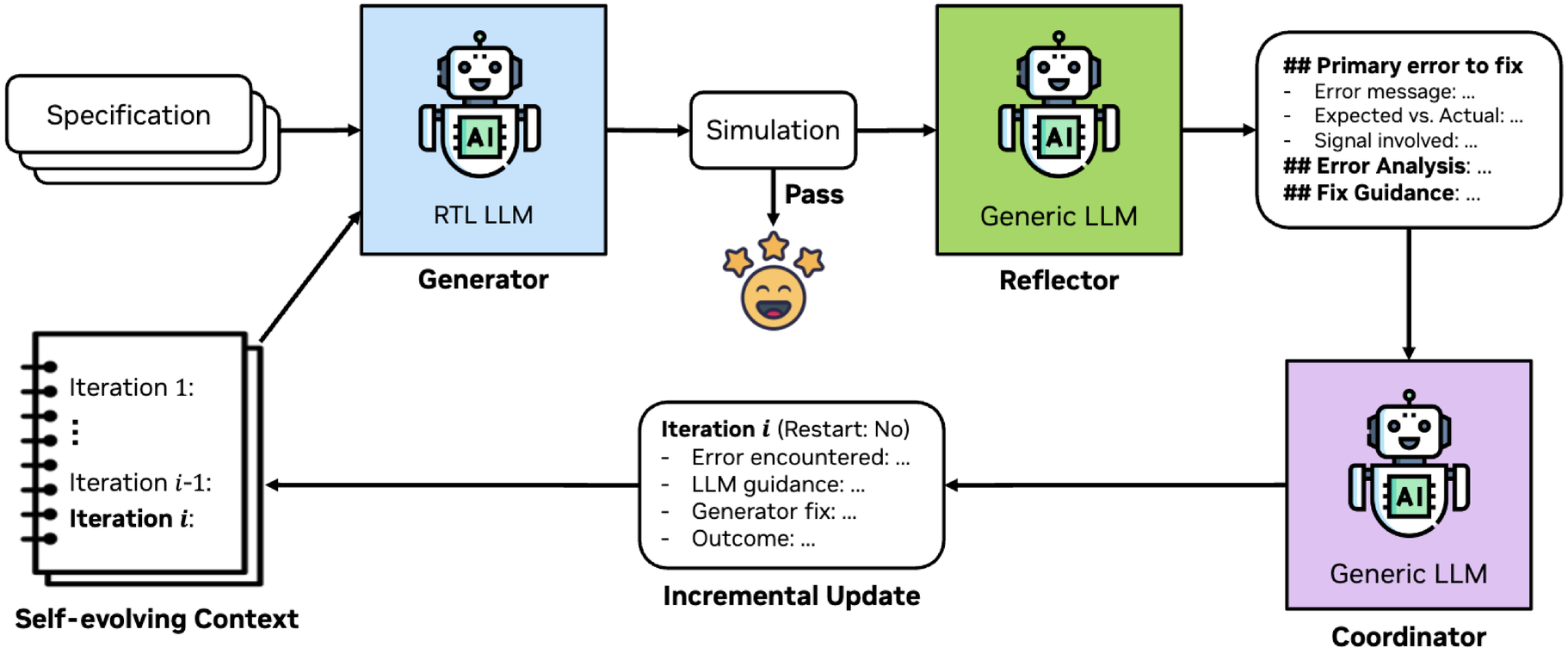
ScaleRTL (32B) was the SOTA small LLM on VerilogEval



Chenhui Deng, et al, ScaleRTL: Scaling LLMs with Reasoning Data and Test-Time Compute for Accurate RTL Code Generation

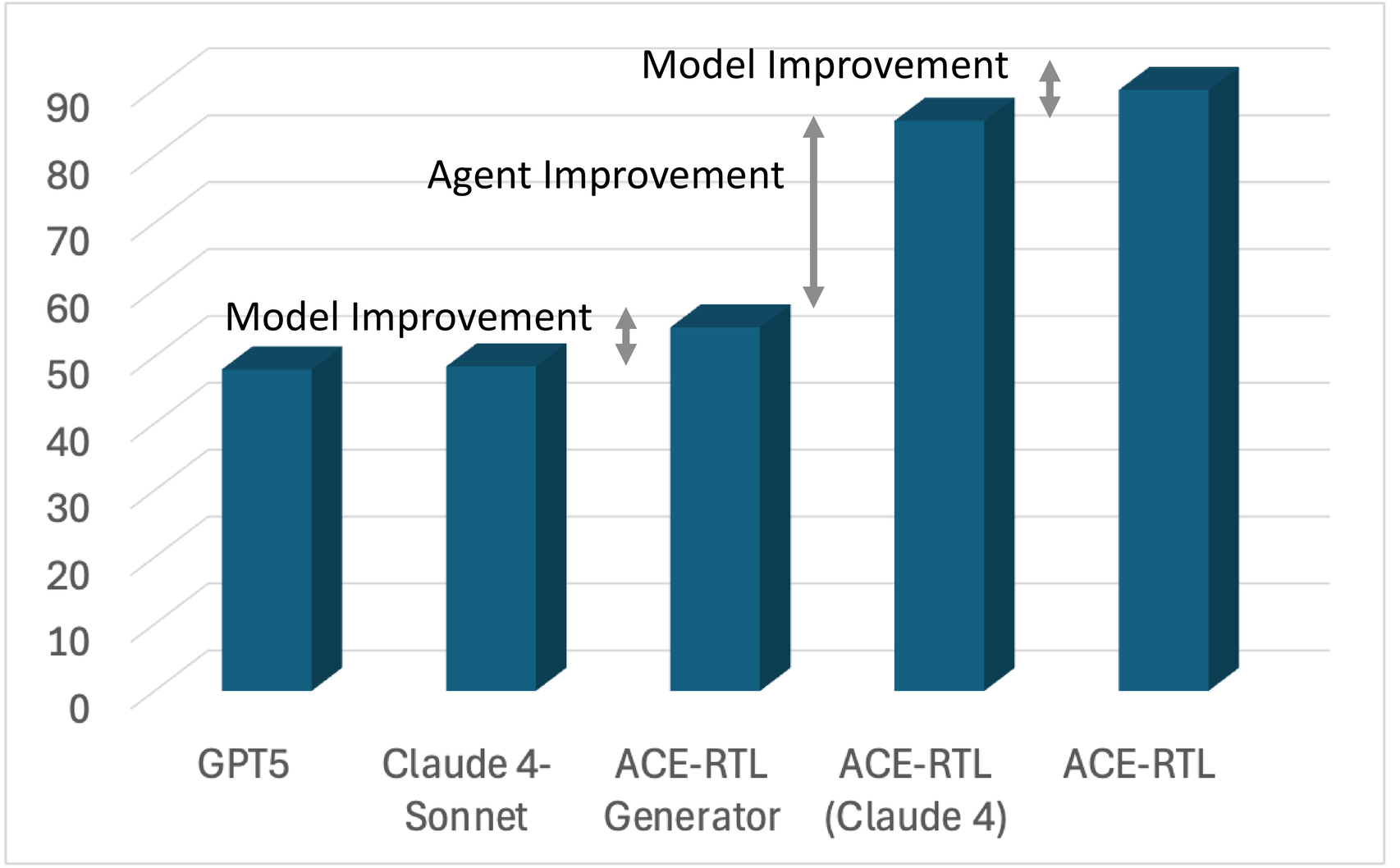
RTL Coding Tasks

Include a custom model in the agent



RTL Coding Tasks

Achieved 88.9% Pass Rate Across Four CVDP RTL Coding Tasks



Chenhui Deng, et al, ACE-RTL: When Agentic Context Evolution Meets RTL-Specialized LLMs

Developing Custom Agents for Hardware Design @ NVIDIA

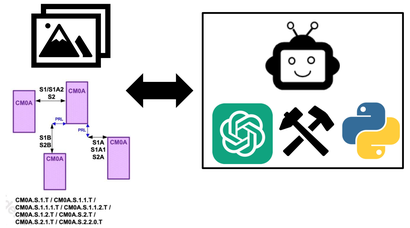
Lesson learned: Custom Agent Development Not Scalable

Agents

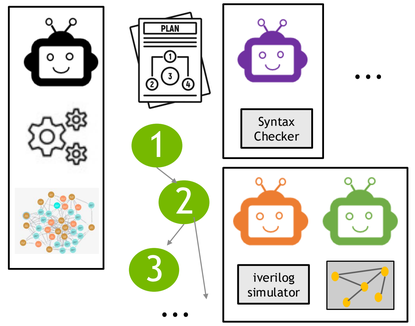
More Agent for Hardware Design

SPICE Verification Agent
ChipVQA Agent
ERC Check Agent
AssertionForge Agent [ICLAD 2025]
SPEC2RTL Agent [ICLAD 2025]
⋮

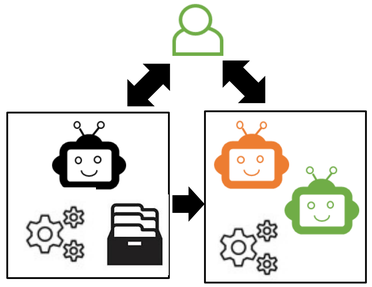
DRC-Coder [ISPD 2025]
Multi-Modality & Multi-LLMs + Tools + Coding
NTECH'24



VerilogCoder [AAAI 2025]
Multi-LLMs + Dynamic Task Graph + Customized Debug Tools

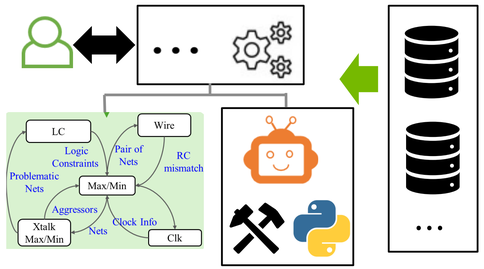


STA Report Summary Agent
Multi-LLMs + Tools Reasoning + Human Interactive



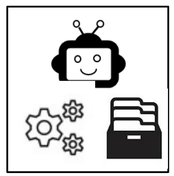
≈ 60× Speedups

NanoTime Agent
Task Graph + Multi-LLMs + Agentic Retrieval

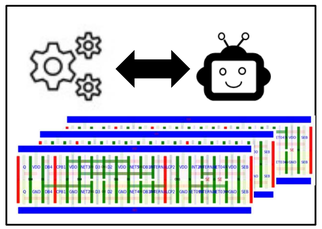


Reduce hours to mins

RTLFixer [DAC'24]
ReAct+ Compiler + RAG



LLM for Standard Cell Design Optimization [LAD'24 Best Paper]
ReAct+ Netlist Tools Reasoning



Sept, 2023

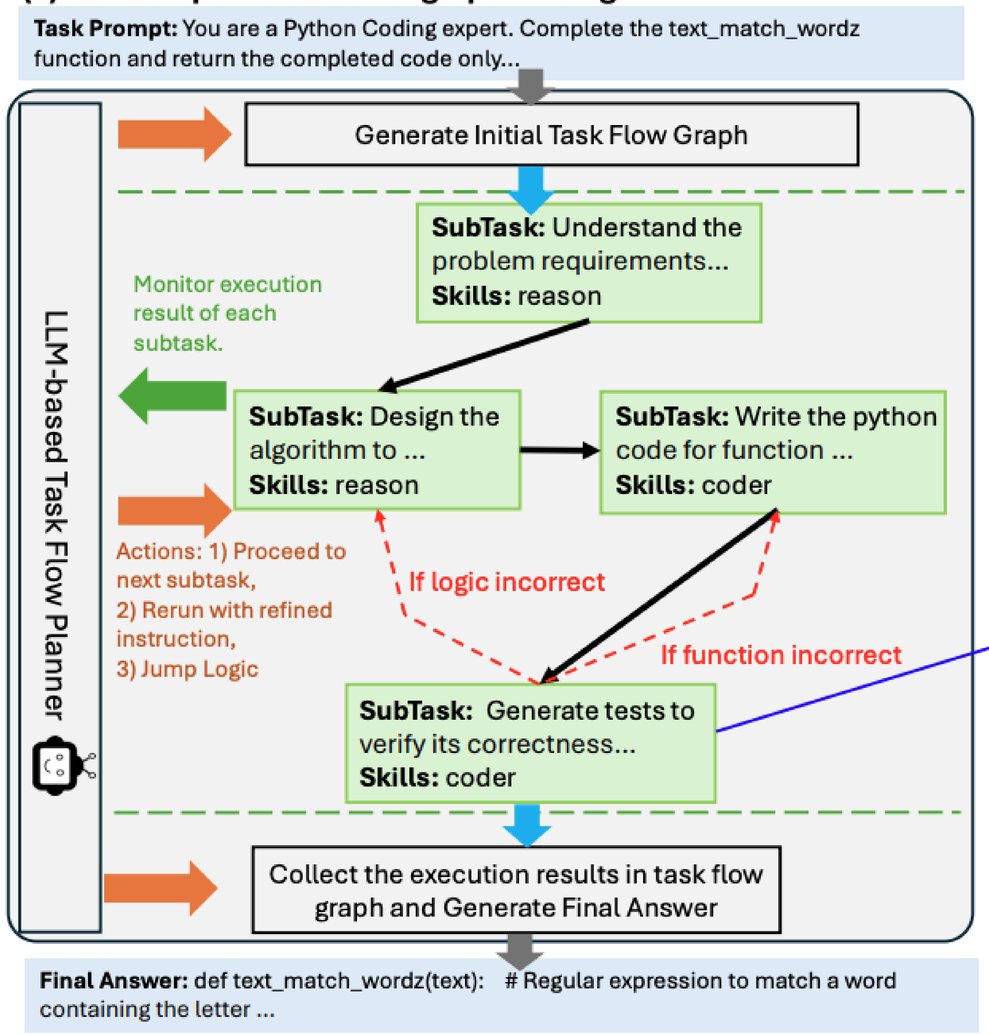
April, 2024

July, 2024

General Purpose Agent: Polymath Agent

Hierarchical Planning with graph and code plans

(a) An example of task flow graph solving



(b) An example of subtask code-represented workflow

SubTask: Generate tests to verify its correctness...

```
def workflow(instruction: str):
    instruction = "### Instruction Background ###\n" + instruction + "\n"
<<<<<<< SEARCH
    ... # your designed workflow here.
=======
    import json

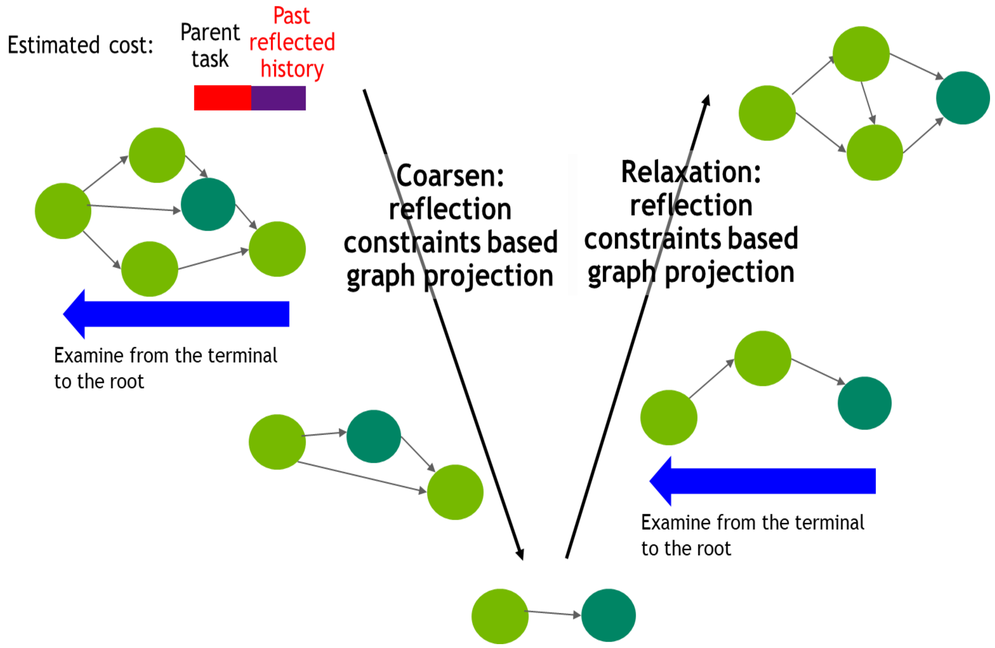
    # Task_3: Test the text_match_wordz function
    # Test inputs for the function
    test_cases = [
        "apple",
        "zebra",
        "buzz",
        "fizz",
        "hello"
    ]
    # Expected outputs: [False, True, True, True, False]
    expected_outputs = [False, True, True, True, False]

    # Chat with coder to run tests on function
    coder_instruction = f"Test the text_match_wordz function with inputs {test_cases}.\\n"
    coder_instruction += "Ensure the function returns the correct boolean values indicating the presence of 'z' in words.\\n"
    coder_instruction += "Return True for words containing 'z' and False otherwise.\\n"
    response = coder_assistant.initiate_chat(message=instruction + coder_instruction)

    # Output the test results
>>>>>>> REPLACE
    return response
```

Optimization of the Polymath Agent

Top-level
Divide & Conquer: Manageable Subtask



Multi-grid based optimization based on LLM-as-judge

Subtask/Coding-level (Evolutionary Alg.)
Optimize subtask execution & performance

```python
# EVOLVE-BLOCK-START
def workflow(instruction: str):
    """
    Workflow to generate the `drop_empty` function following the instruction input.
    """
    # Send the instruction directly to the coder assistant
    coder_response = coder_assistant.initiate_chat(message=instruction)
    coder_output = extract_output(
        response=coder_response,
        post_process_func=coder_description["post_process_func"]
    )

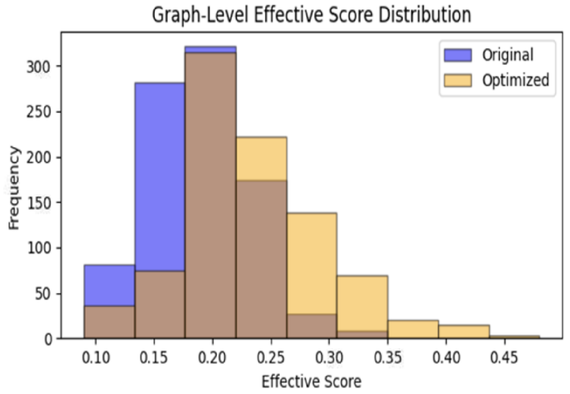
    # Return the coder's output directly
    return coder_output
# EVOLVE-BLOCK-END
```

Self-Reflection Guided Evolution

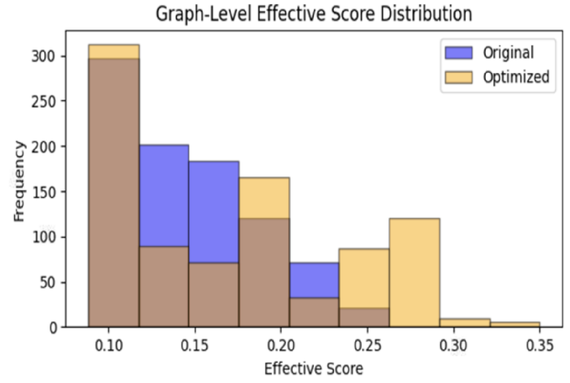
Optimization Improvement on Multiple Benchmarks

2%-8% performance improvements

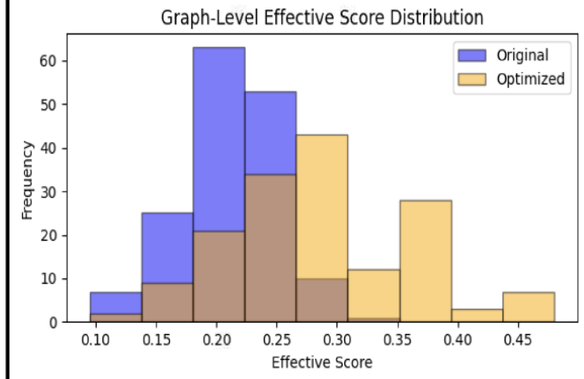
Multi-turn QA: Hotpot QA



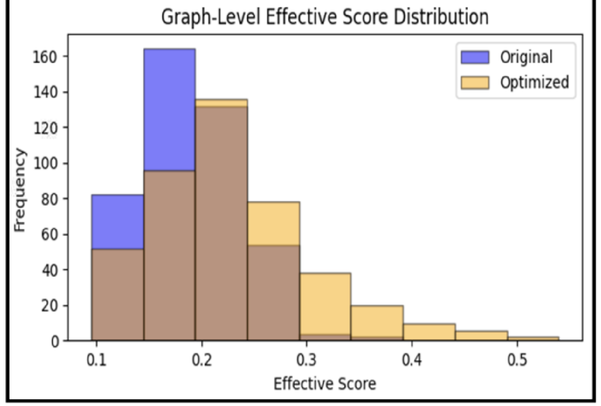
Multi-turn QA: DROP



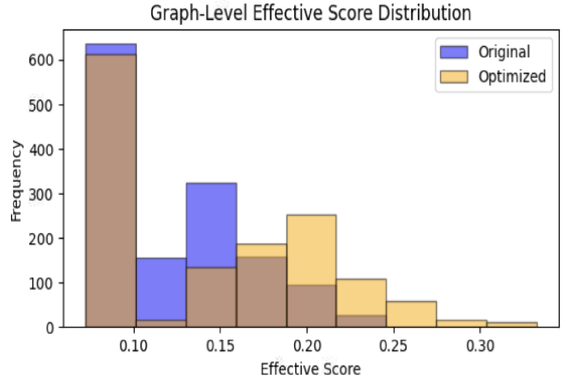
Coding: HumanEval



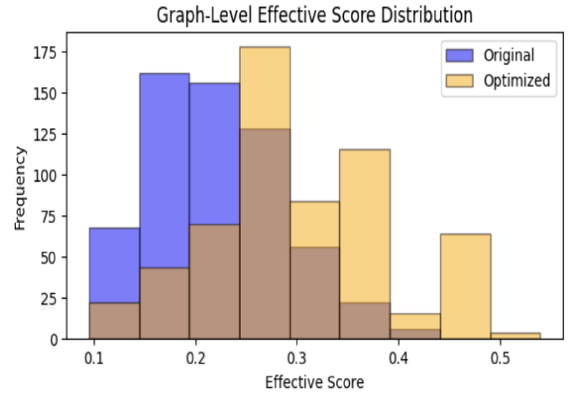
Coding: MBPP



Math Reasoning: GSM8K



Math Reasoning: MATHlv5*



Chia-Tung Ho et al, Polymath: A Self-Optimizing Agent with Dynamic Hierarchical Workflow

Autonomous Multi-Agent Generation

Self-Improving Agent Solving RTL coding problems

Gen 0: Single-Agent

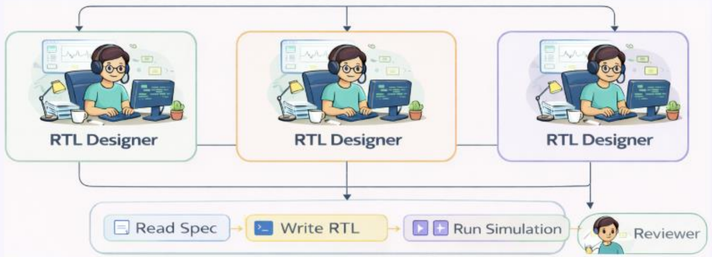
1 Agents + Tools + Skills



Gen 1: Multi-Agent System

5 Agents

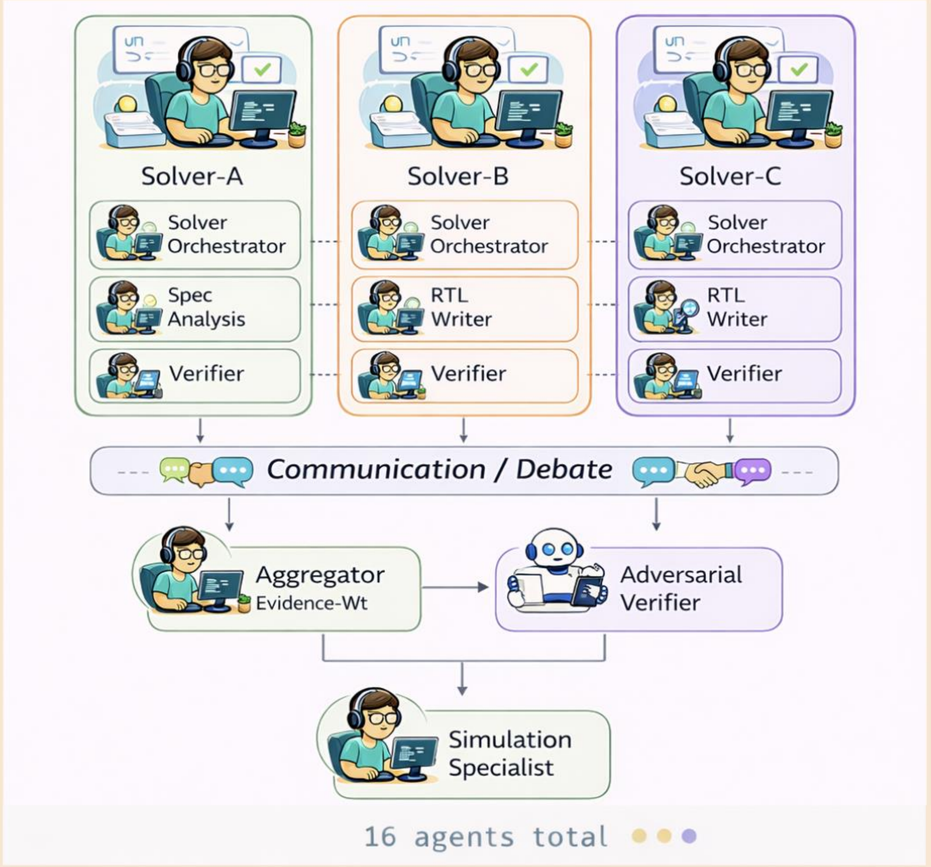
Orchestrator + 3 Designers + Reviewer



Gen 2: Hierarchical Agent Network

16 Agents

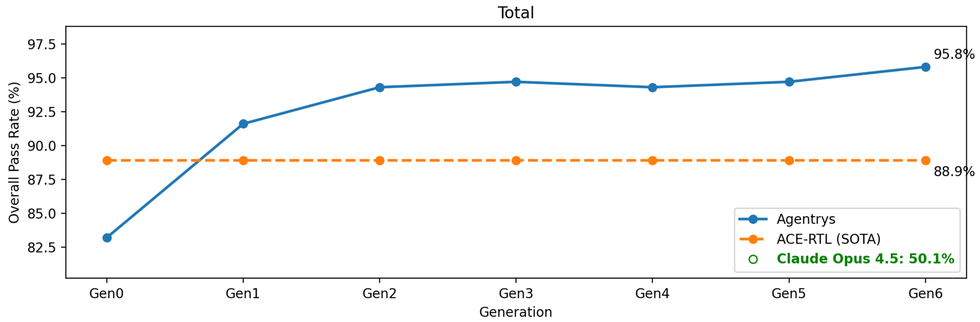
Orchestrator + 3 Solver Teams + Aggregator + Verifier + Specialist

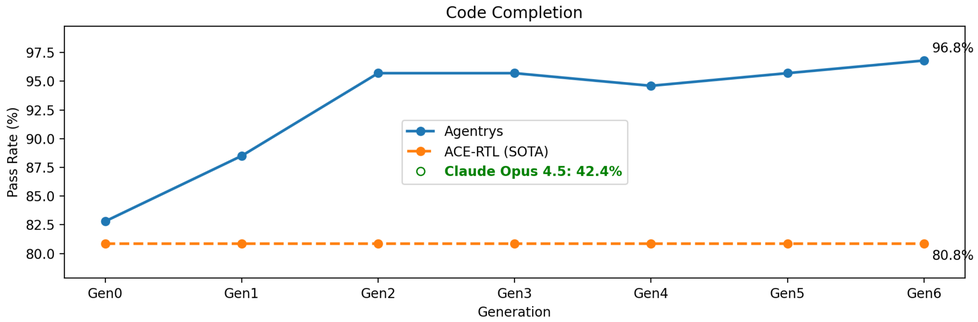


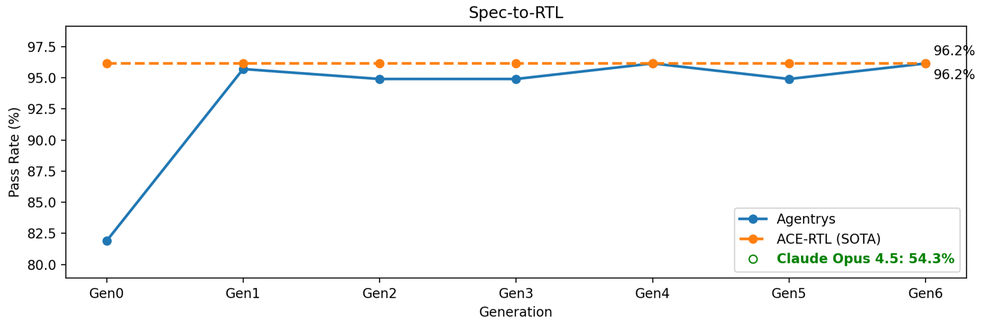
https://agentrys.ai/blog-cvdp.html

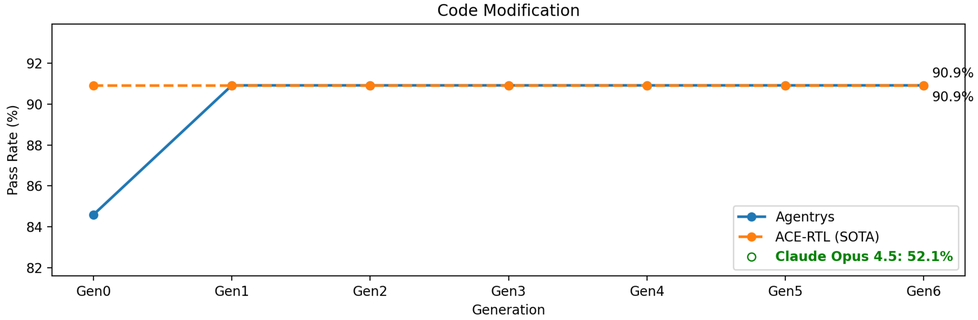
Reaches 95.8% Overall Pass Rate with 100% on RTL Debug

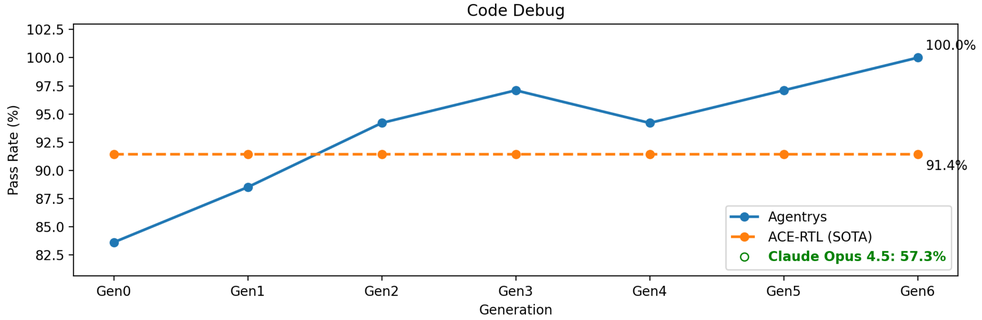
7% better than hand-coded ACE-RTL Agent











https://agentrys.ai/blog-cvdp.html

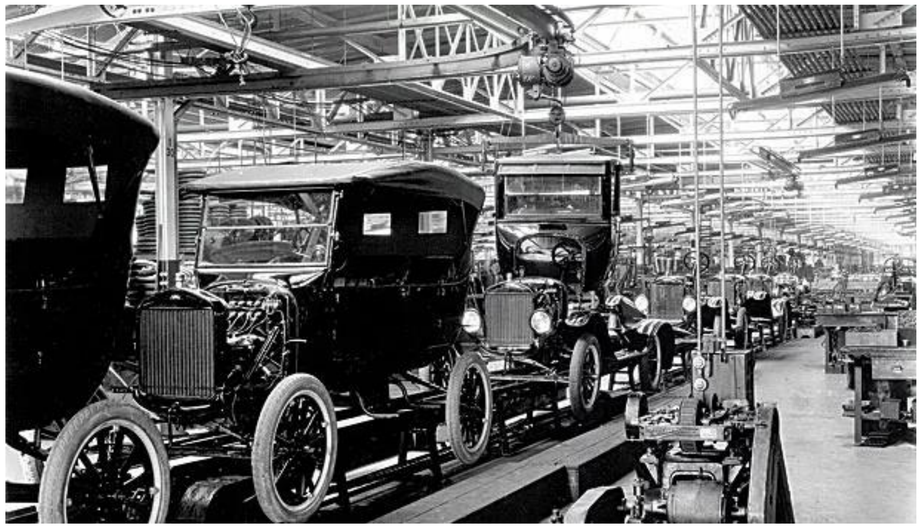
Toward Autonomous Design

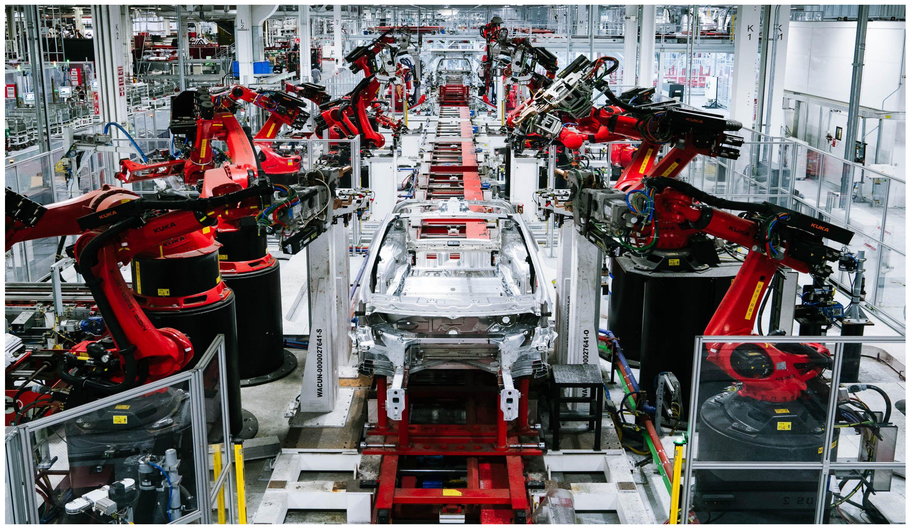
Agentic Design Automation will lead us to autonomous design

Agent-native tools and custom models make agents more capable

Agents need to be optimized and improved to achieve the best performance

Future: Agents design chips. Designers develop the agents.





AGENTRYS

AGENTRYS enables design teams develop their own agentic design automation system and learn from their own data.

Agentrys.ai