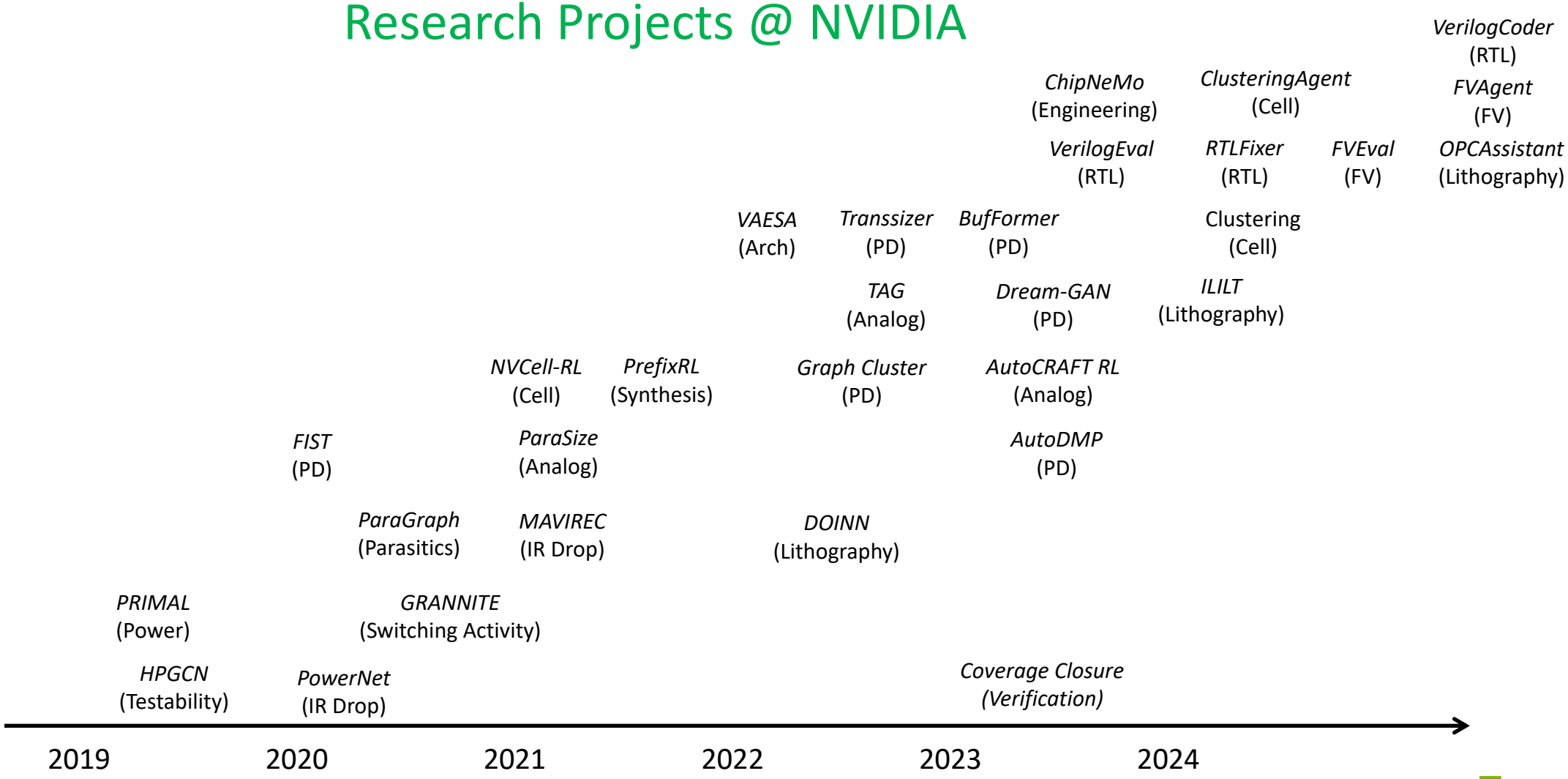
Towards Revolutionizing Chip Design with AI: The Integration of AI and Algorithm

Haoxing (Mark) Ren, Director of Design Automation Research, NVIDIA

AI works for many chip design problems

Research Projects @ NVIDIA



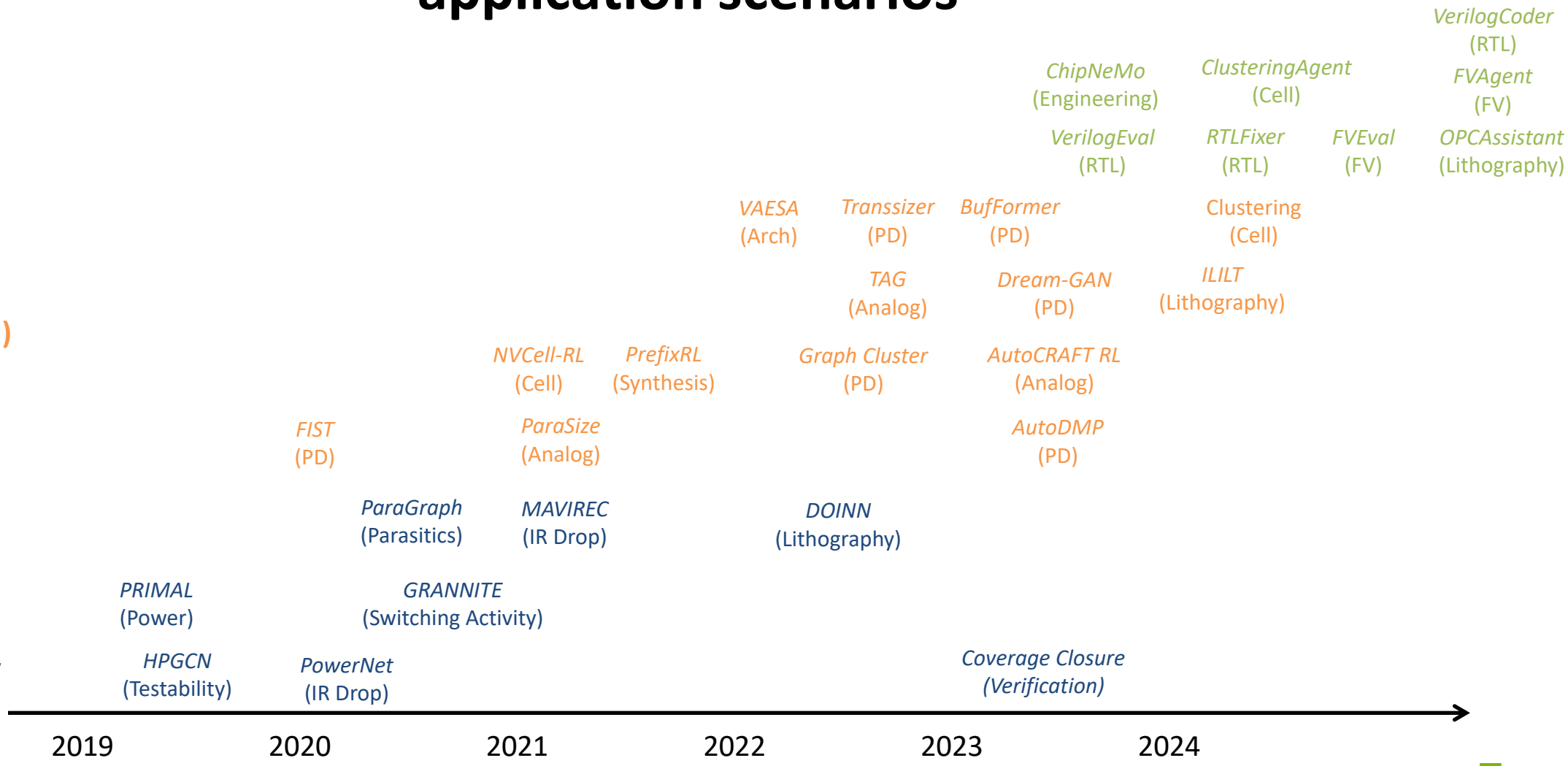
NVIDIA

Analysis, Optimization, and Assistance are the main application scenarios

Assistance (LLM/Agent)
Know-how
Coding
Task automation

Optimization (BO/RL/GenAI)
Faster
More scalable
Better results

Analysis (CNN/GNN)
Faster
Cross-Stage
Detect Anomaly



NVIDIA

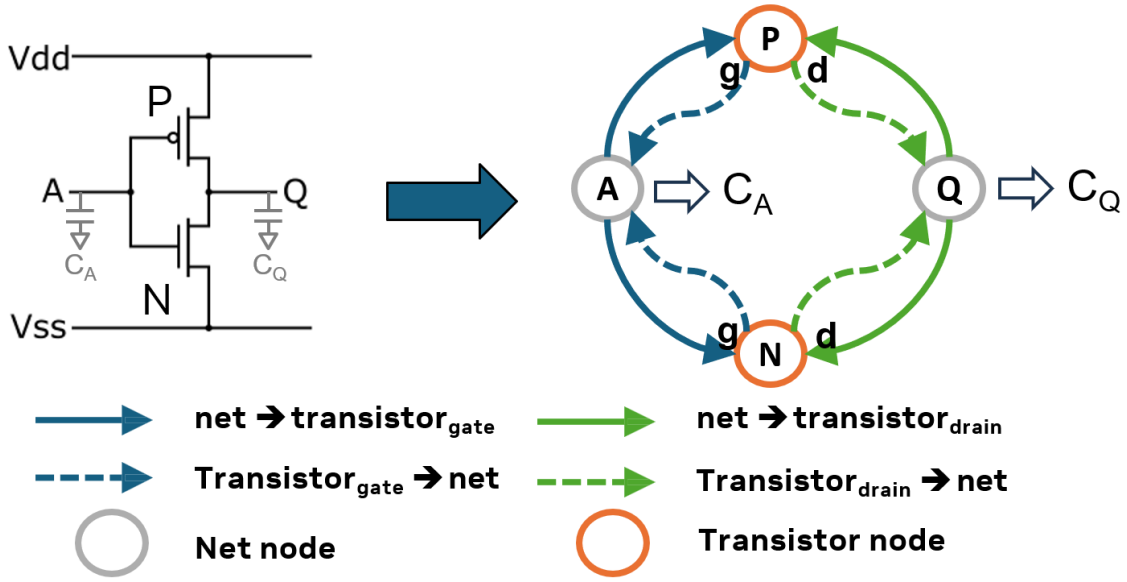
AI predicts cross-stage design metrics

Impact of layout parasitics on schematic design

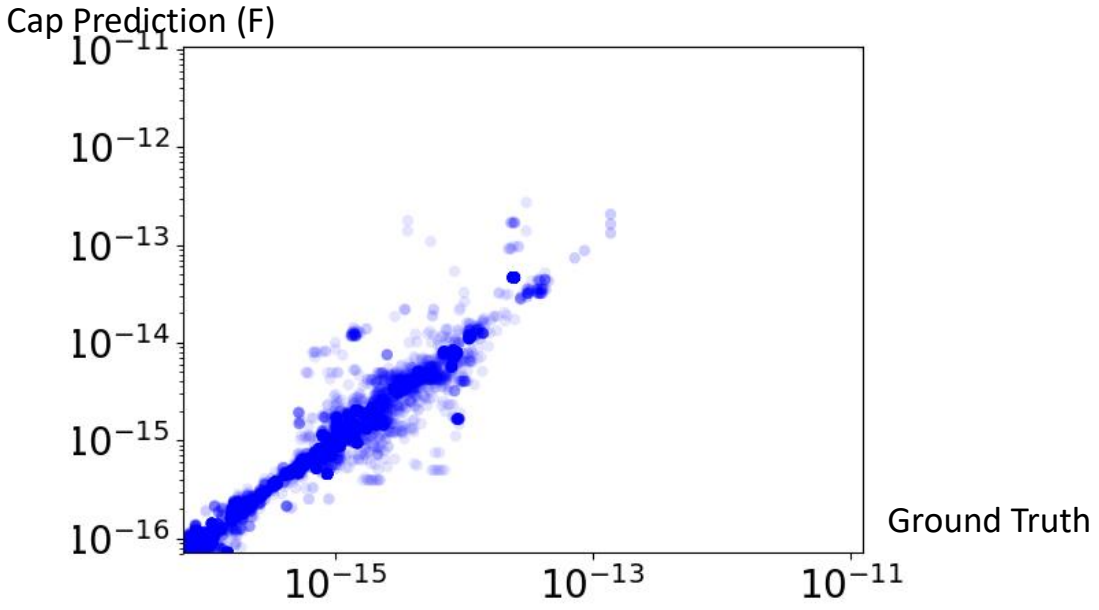
Use AI to predict layout parasitics from schematic given a large dataset

Convert schematic to graph and learn with GNN

Predicted parasitics reduced simulation error to <10%



Circuit Schematics to Heterogenous Graph Conversion



MAE=0.852fF MAPE=15%

H. Ren et al, ParaGraph: Layout Parasitics and Device Parameter Prediction using Graph Neural Networks

AI optimizes tools parameters

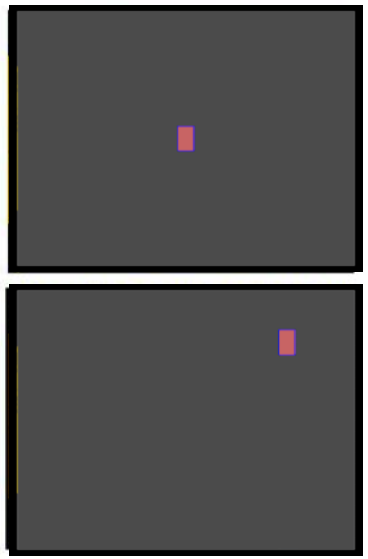
Macro placement quality is very important for physical design

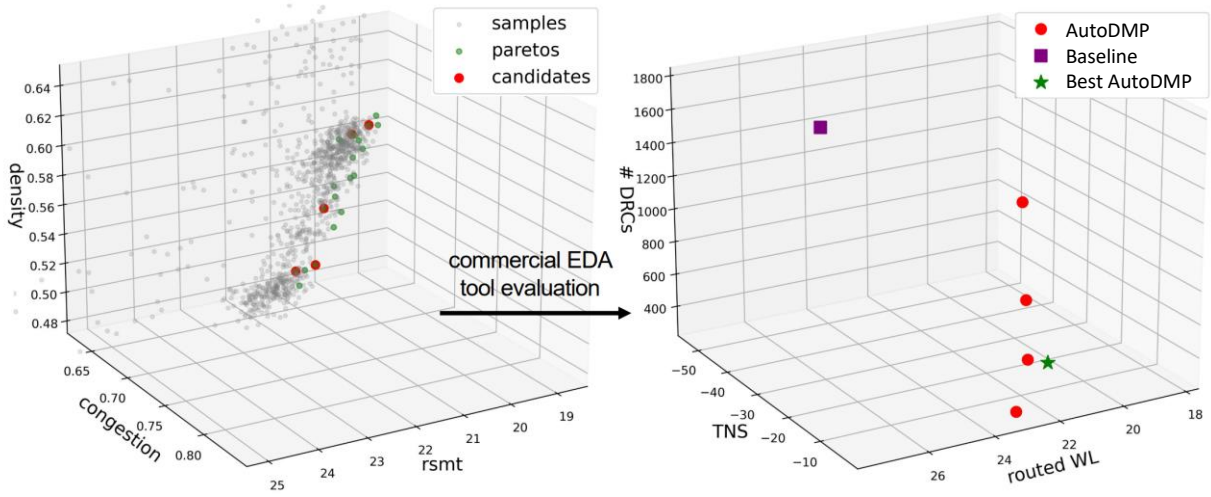
Placement tools parameters have a huge impact on macro placement

Multi-objective Bayesian Optimization (sample and model) find best parameter set

Find better macro placement with open-source GPU accelerated placement tools

Parameter	Search Range
*horiz. initial position *vert. initial position	[0.2, 0.8] (%) [0.2, 0.8] (%)
*horiz. macro halo *vert. macro halo	technology dep. technology dep.
target density d_{target} density weight	$[a_{\text{util}} - 0.2, a_{\text{util}}]$ (%) $[1e^{-6}, 1.0]$
smooth HPWL model smooth HPWL initial γ_0	{LSE, WA} [0.10, 0.50]
GD initial LR lr_0 GD LR decay GD optimizer	$[1e^{-4}, 1e^{-2}]$ [0.99, 1.0] [Adam, Nesterov]
# horiz. global bins # vert. global bins	{256, 512, 1024, 2048} {256, 512, 1024, 2048}
λ update lower coeff. L λ update upper coeff. U λ update Δ HPWL_{REF}	[0.90, 0.99] [1.01, 1.15] $[1.5e^5, 5.5e^5]$





A. Agnesina et al, AutoDMP: Automated DREAMPlace-based Macro Placement

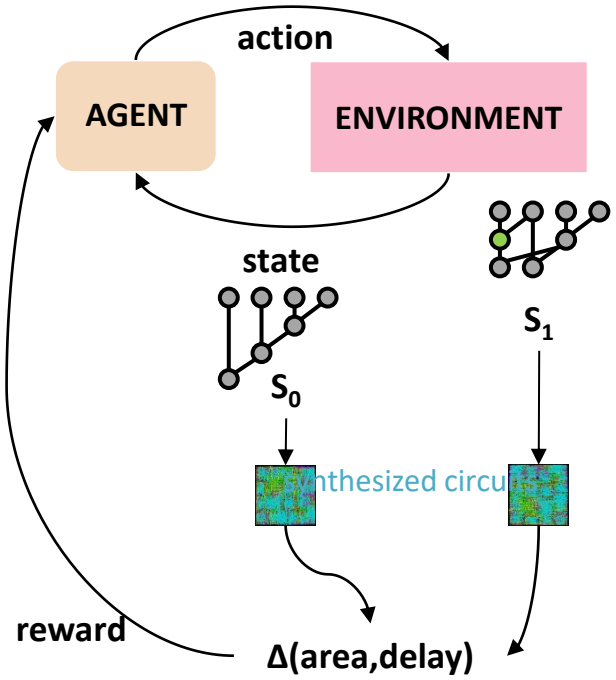
NVIDIA

AI optimizes datapath structure

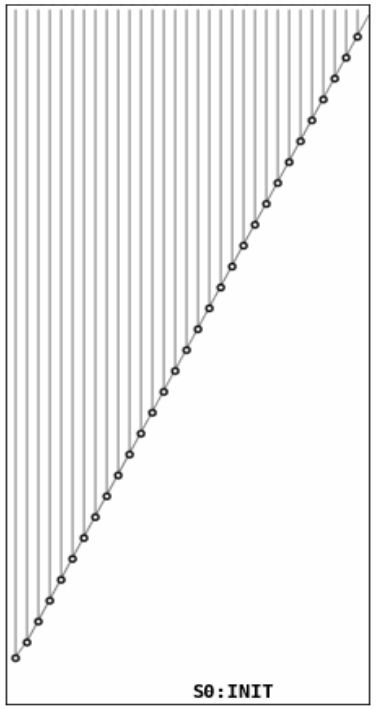
Datapath synthesis important for GPU

Optimize prefix adder structure with Reinforcement Learning for better delay and area tradeoff

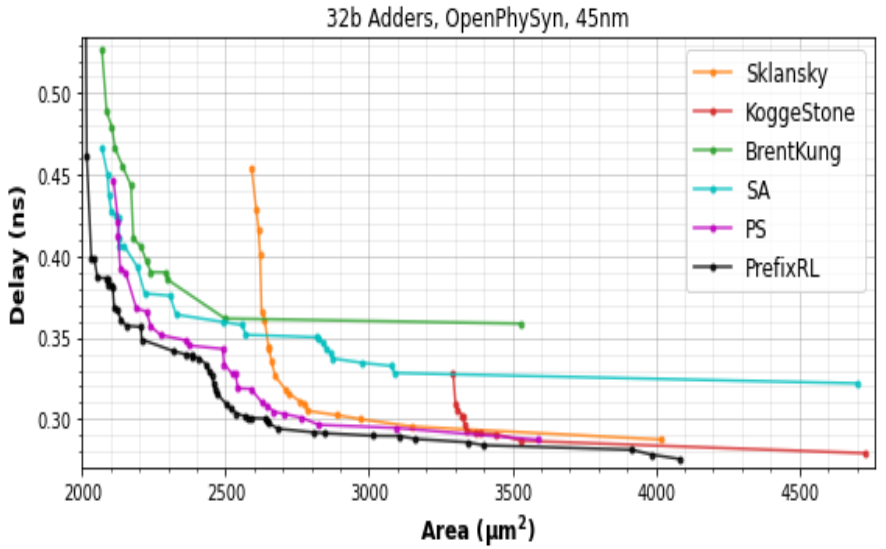
Reward computed by open-source EDA tools



Deep Q learning
Circuit synthesis in-the-loop



Action Space
Add/Delete prefix graph nodes



PrefixRL achieves better results than well known adder architectures

R. Roy et al, PrefixRL: Optimization of parallel prefix circuits using deep reinforcement learning

NVIDIA

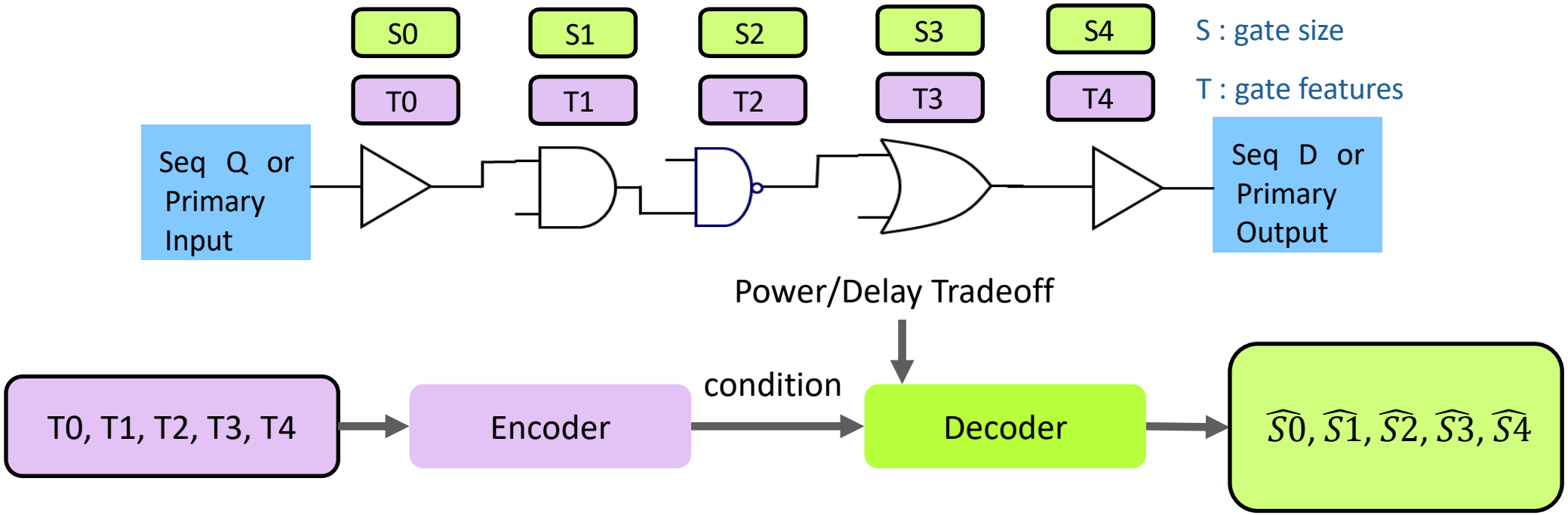
AI Generates Optimal Gate Sizes

Timing/power optimization such as gate sizing affects scalability of PD tools

Model a path of gates as a sequence, generate optimized gate sizes using Transformer

Trained on tool optimized netlist dataset

100X – 1000X speedup compared to traditional optimization with similar PPA



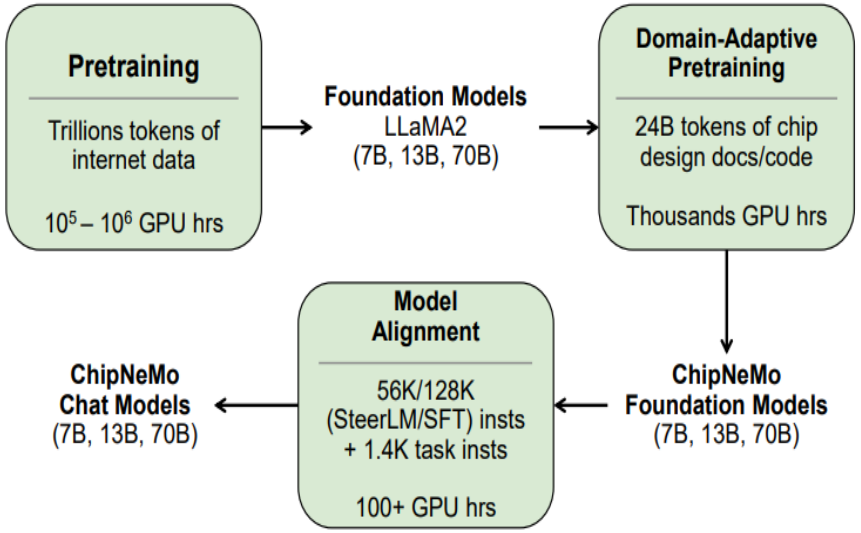
S. Nath et al, Transizer: A Novel Transformer-Based Fast Gate Size

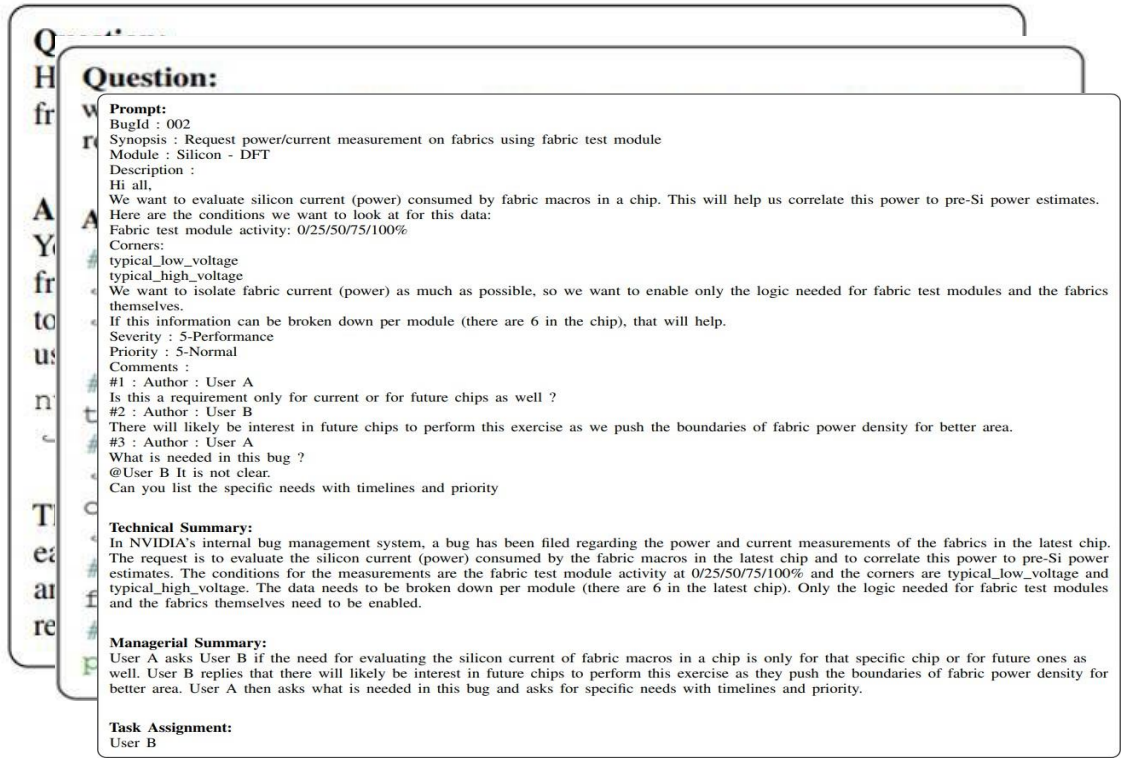
AI assists engineers in QnA, coding and analysis tasks

LLM is good at question answering, coding, extraction, rewriting, summarization, classification, reasoning, …

- Answer questions about designs, infrastructures, tools, flows, HW domains, etc.
- Generate scripts for specific tasks (VLSI)
- Summarize bug reports, predict assignment

Domain-adapted model have better performance





Question:

Prompt:
BugId : 002
Synopsis : Request power/current measurement on fabrics using fabric test module
Module : Silicon - DFT
Description :
Hi all,
We want to evaluate silicon current (power) consumed by fabric macros in a chip. This will help us correlate this power to pre-Si power estimates.
Here are the conditions we want to look at for this data:
Fabric test module activity: 0/25/50/75/100%
Corners:
typical_low_voltage
typical_high_voltage
We want to isolate fabric current (power) as much as possible, so we want to enable only the logic needed for fabric test modules and the fabrics themselves.
If this information can be broken down per module (there are 6 in the chip), that will help.
Severity : 5-Performance
Priority : 5-Normal
Comments :
#1 : Author : User A
Is this a requirement only for current or for future chips as well ?
#2 : Author : User B
There will likely be interest in future chips to perform this exercise as we push the boundaries of fabric power density for better area.
#3 : Author : User A
What is needed in this bug ?
@User B It is not clear.
Can you list the specific needs with timelines and priority

Technical Summary:
In NVIDIA's internal bug management system, a bug has been filed regarding the power and current measurements of the fabrics in the latest chip. The request is to evaluate the silicon current (power) consumed by the fabric macros in the latest chip and to correlate this power to pre-Si power estimates. The conditions for the measurements are the fabric test module activity at 0/25/50/75/100% and the corners are typical_low_voltage and typical_high_voltage. The data needs to be broken down per module (there are 6 in the latest chip). Only the logic needed for fabric test modules and the fabrics themselves need to be enabled.

Managerial Summary:
User A asks User B if the need for evaluating the silicon current of fabric macros in a chip is only for that specific chip or for future ones as well. User B replies that there will likely be interest in future chips to perform this exercise as they push the boundaries of fabric power density for better area. User A then asks what is needed in this bug and asks for specific needs with timelines and priority.

Task Assignment:
User B

M. Liu et al, ChipNeMo: Domain-Adapted LLMs for Chip Design

Will AI revolutionize Chip Design?
Skeptics are saying :

Analysis

“Inaccurate predictions”

Optimization:

“AI is not as good as existing algorithms”

“AI will never get better results than the data it trained on”

Assistance

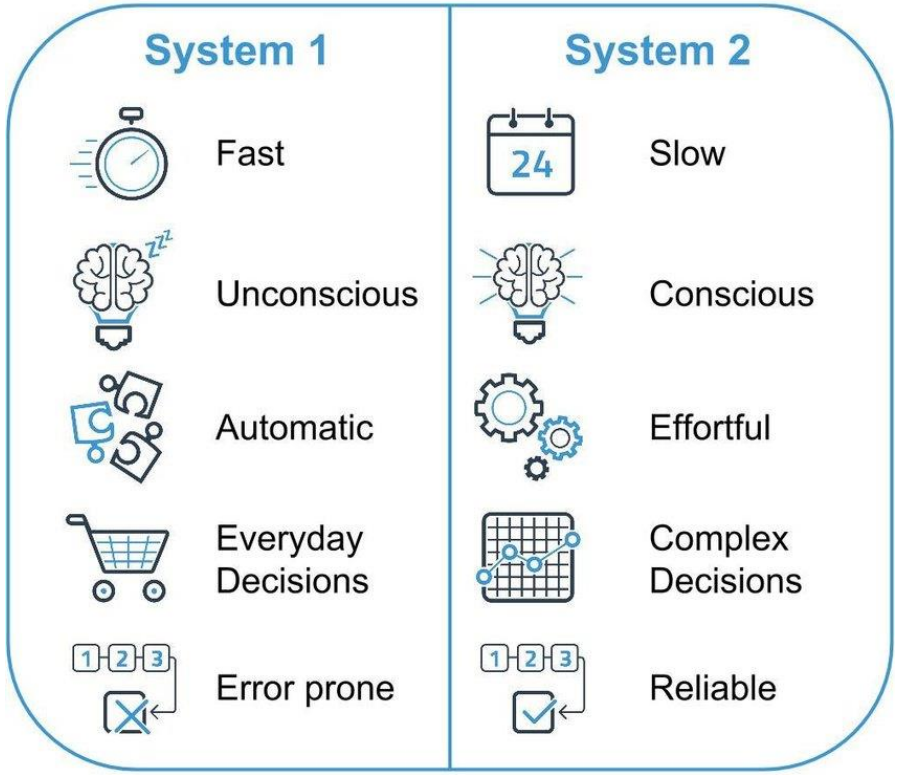
“Hallucinations”

“Can not help real/complex design problems”

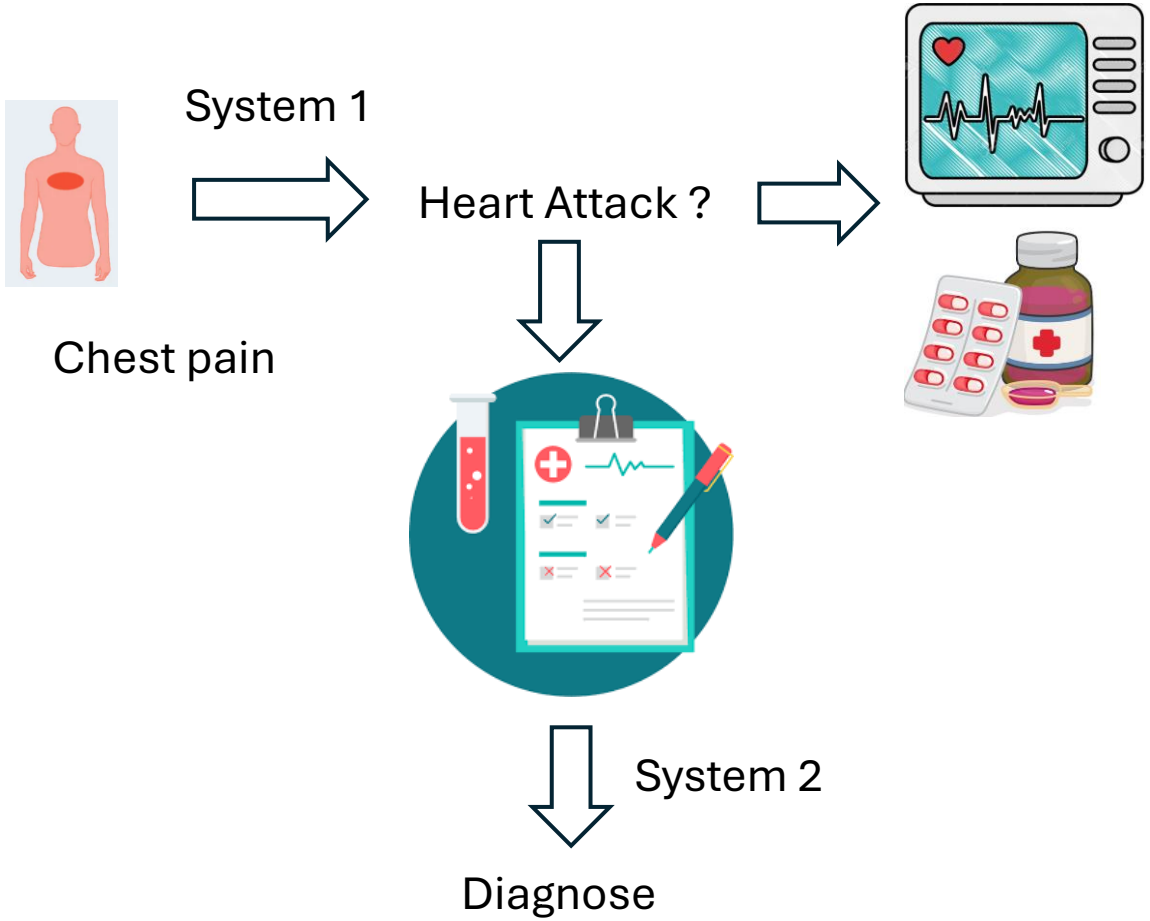
AI and Algorithms are good at different things

	Algorithm	AI
Categories	Placement, route, synthesis, CTS, etc	Supervised, unsupervised, reinforcement learnings, etc
Optimality	Known	Unknown
Robustness	Works on any data distribution	Do not work if training and inference distribution mismatch
Training	No	Require a lot data
Interpretability	Behavior explainable	Not explainable
Pros	Solve a known problem efficiently	Solve any problem by learning from complex data
Cons	Oversimplification of dynamic, complex problem	Difficult to leverage the mechanics of the problem

AI vs Algorithm is like System 1 vs System 2
We use both System 1 and System 2 thinkings for hard problems



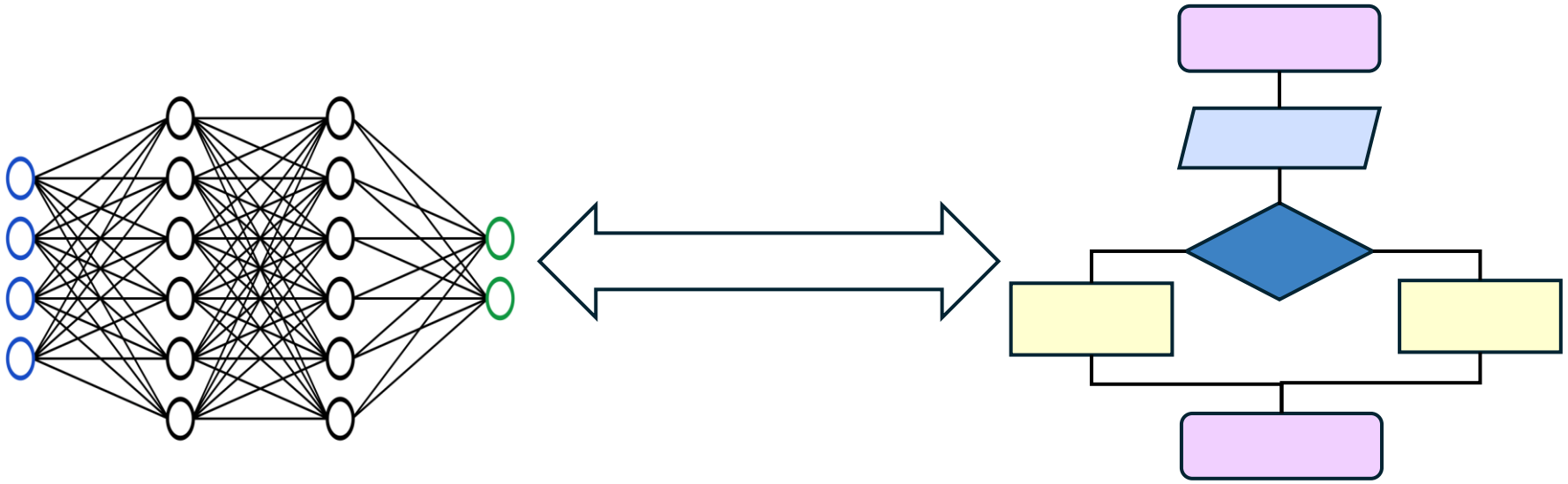
Daniel Kahneman: Thinking, Fast and Slow



To revolutionize Chip Design with AI

Integrate AI with Algorithms

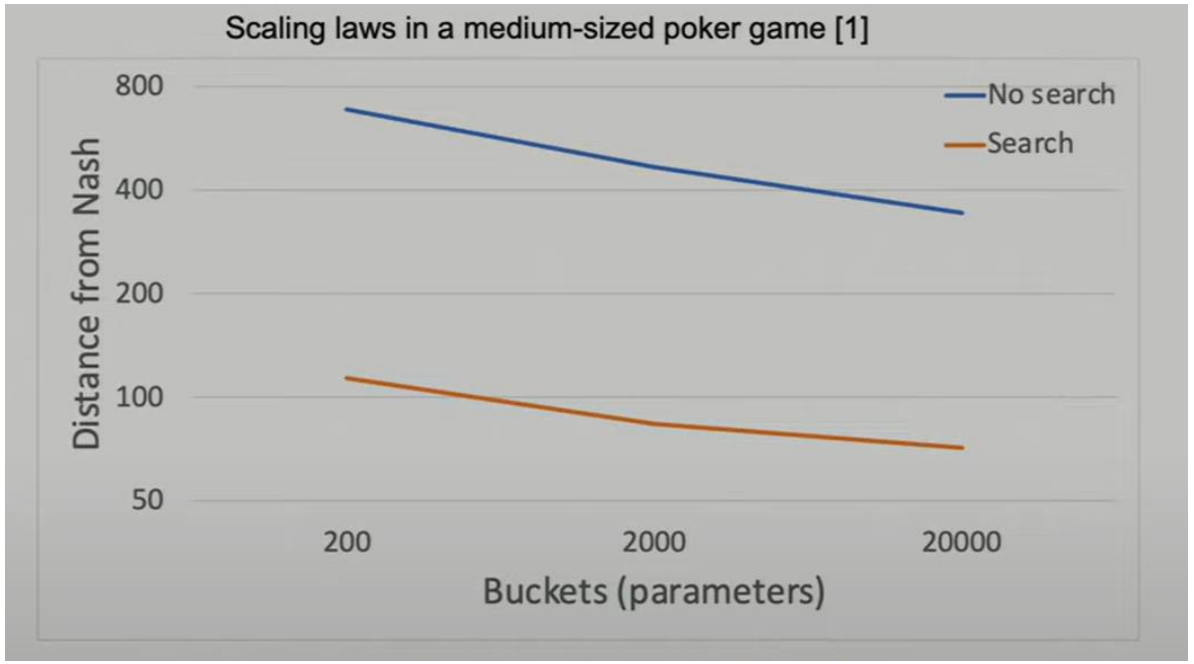
Not a new Idea, Bayesian Optimization/Reinforcement Learning examples won't work without Algorithms,

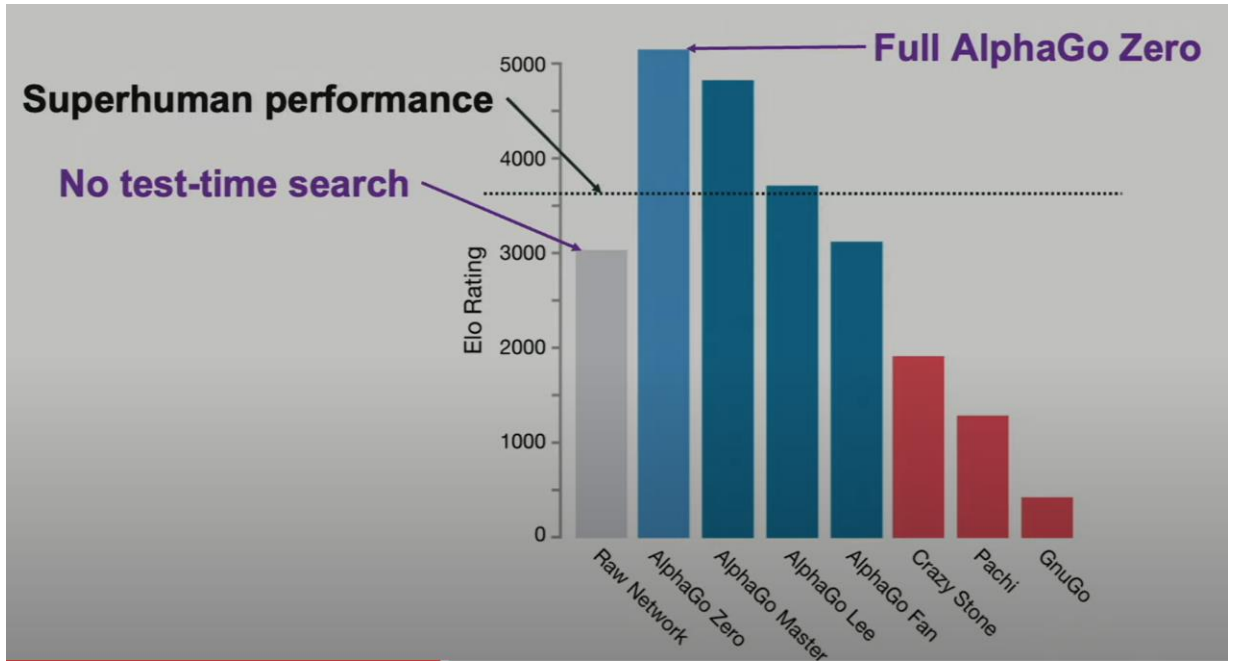


Tighter integration!

For gaming, AI + Algorithm is much better than AI only

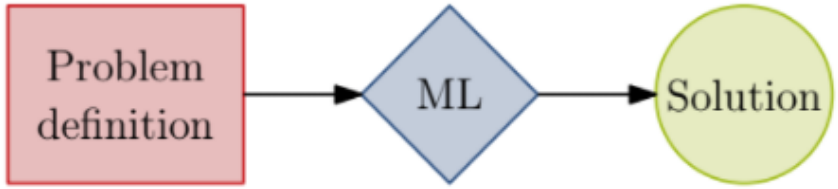
Tight integration of AI with search algorithm helps AI game model perform much better in Poker and Go



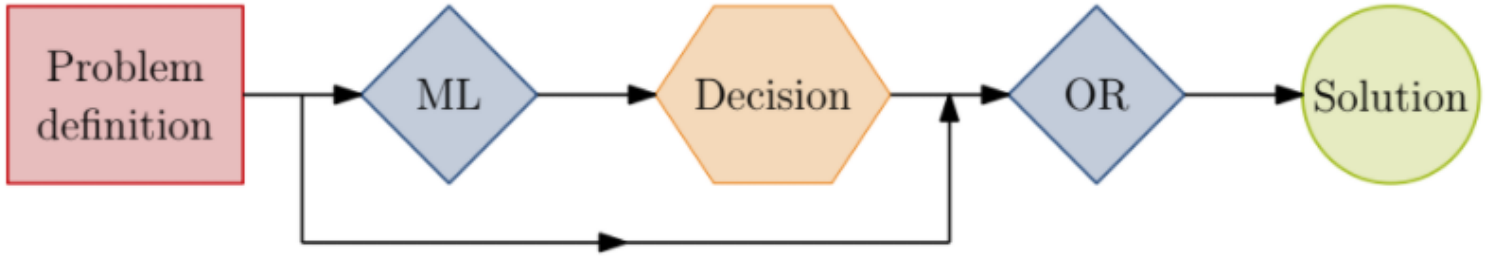


Noam Brown: Parables on the Power of Planning in AI: From Poker to Diplomacy

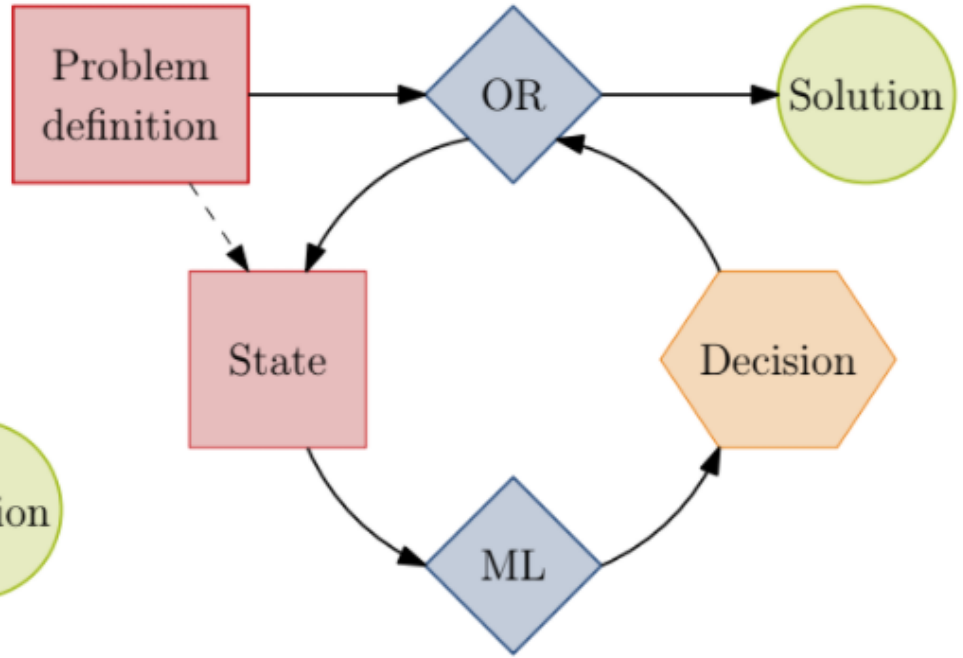
Three ways AI works with Combinatorial Optimization algorithms



End-to-end learning



Coarse grain integration



Fine grain integration

Yoshua Bengio et al, Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon

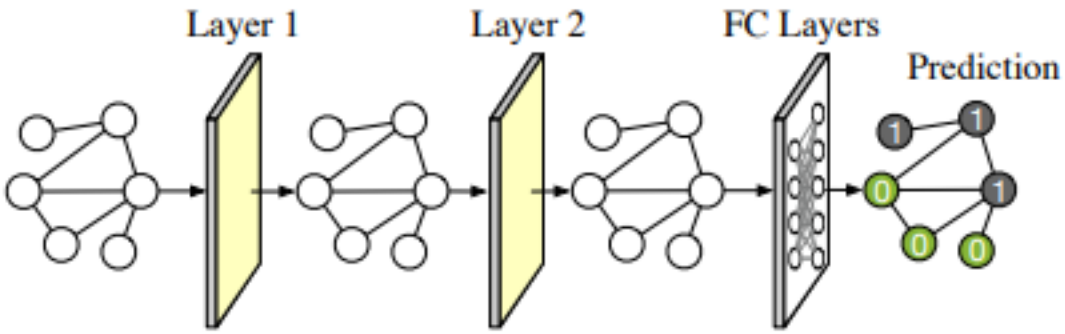
An inaccurate AI prediction model still works by acting as an Oracle in an algorithm

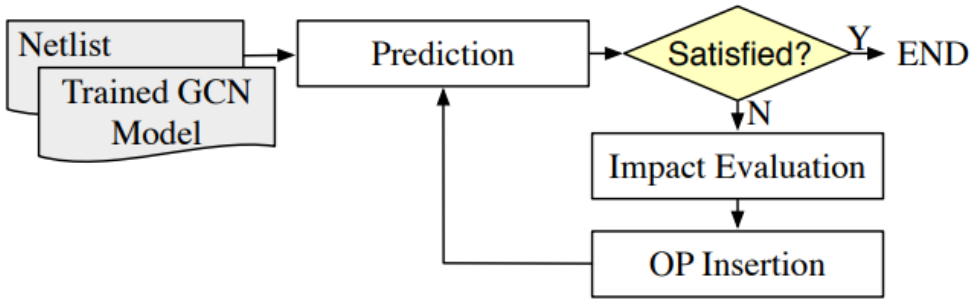
Tackle the test point insertion problem to improve design testability

GCN model predicts node testability: Difficult-to-test nodes (DTN) prediction

Model acts as a predictor/oracle for a test point insertion (TPI) algorithm

Although model accuracy is only 90%, reduced test points by 11% and test patterns by 6% to achieve similar coverage as a commercial test point insertion algorithm





Y. Ma et al, High Performance Graph Convolutional Networks with Applications in Testability Analysis

Ground the AI with an algorithm helps AI get better results than the algorithm that generates the training data

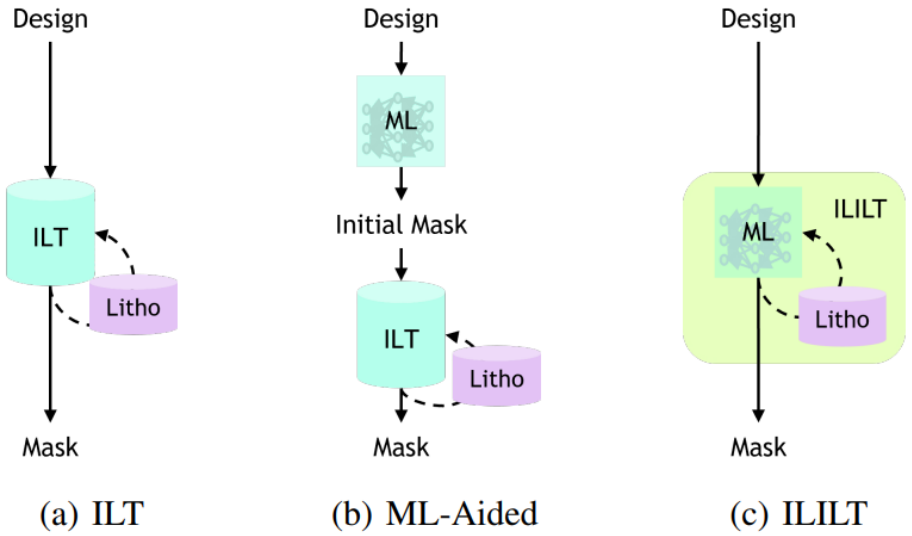
Mask optimization is a timing consuming process for every design

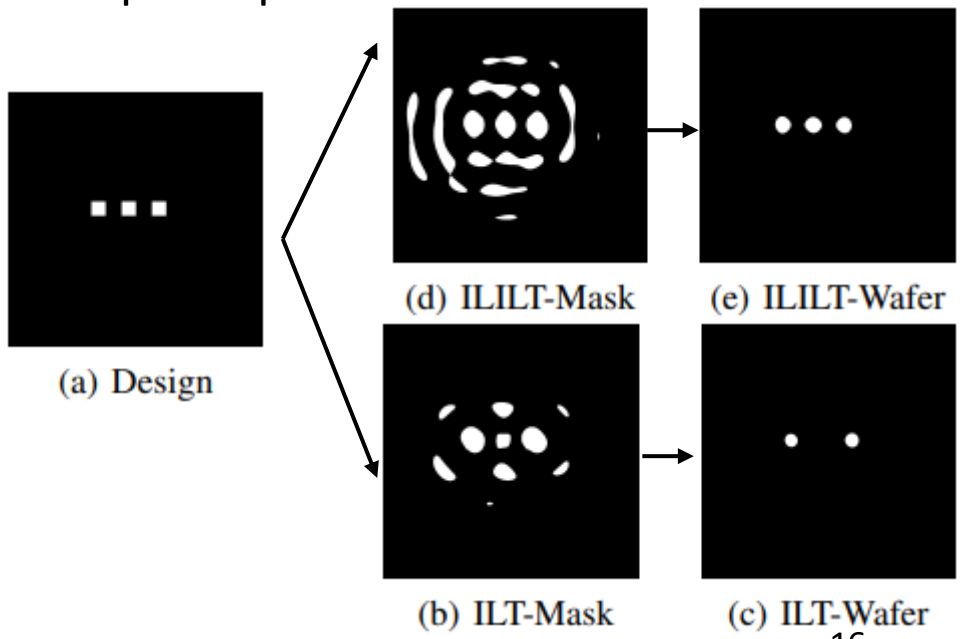
Learn the optimized mask from ILT (inverse lithography Technology) solver

Use Lithography forward simulation algorithm to ground the model

Increase inference time compute with multi-step inference

Better quality than ILT (generated the training data) with 10X speedup





H. Yang et al, ILILT: Implicit Learning of Inverse Lithography Technologies

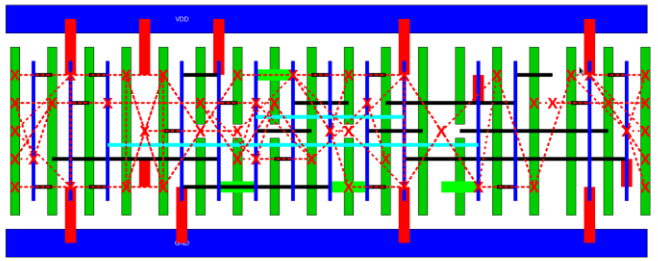
AI as part of a meta algorithm to solve challenges difficult for algorithms to solve

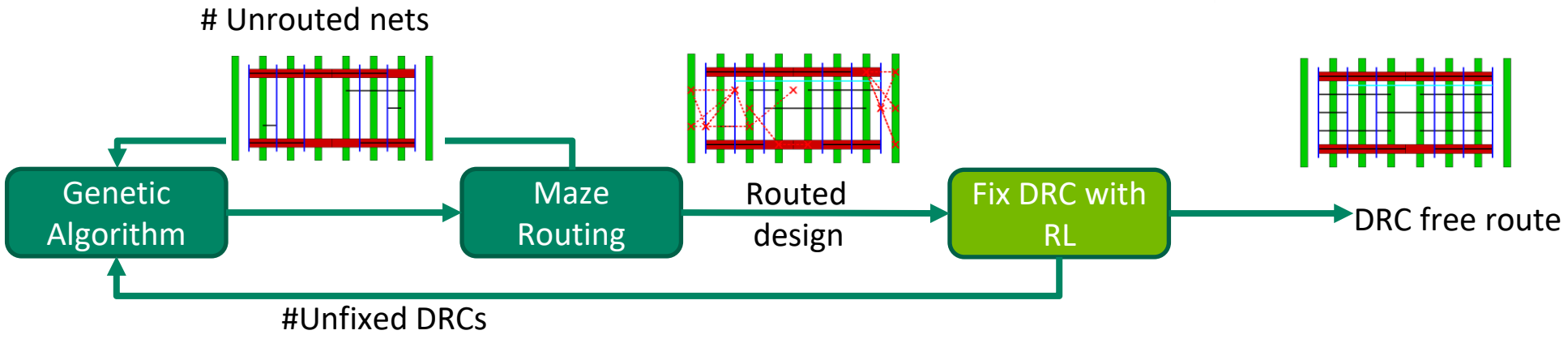
Layouts of thousands cells per library designed manually, difficult to automate

Layout/Routing challenges: satisfying all the DRC rules

Use an algorithm to route, but RL to fix DRC

Generate std cell layout of entire industrial libraries with better quality tl





H. Ren et al, Standard Cell Routing with Reinforcement Learning and Genetic Algorithm in Advanced Technology Nodes

RAGs are algorithms to ground LLM with facts to reduce hallucination

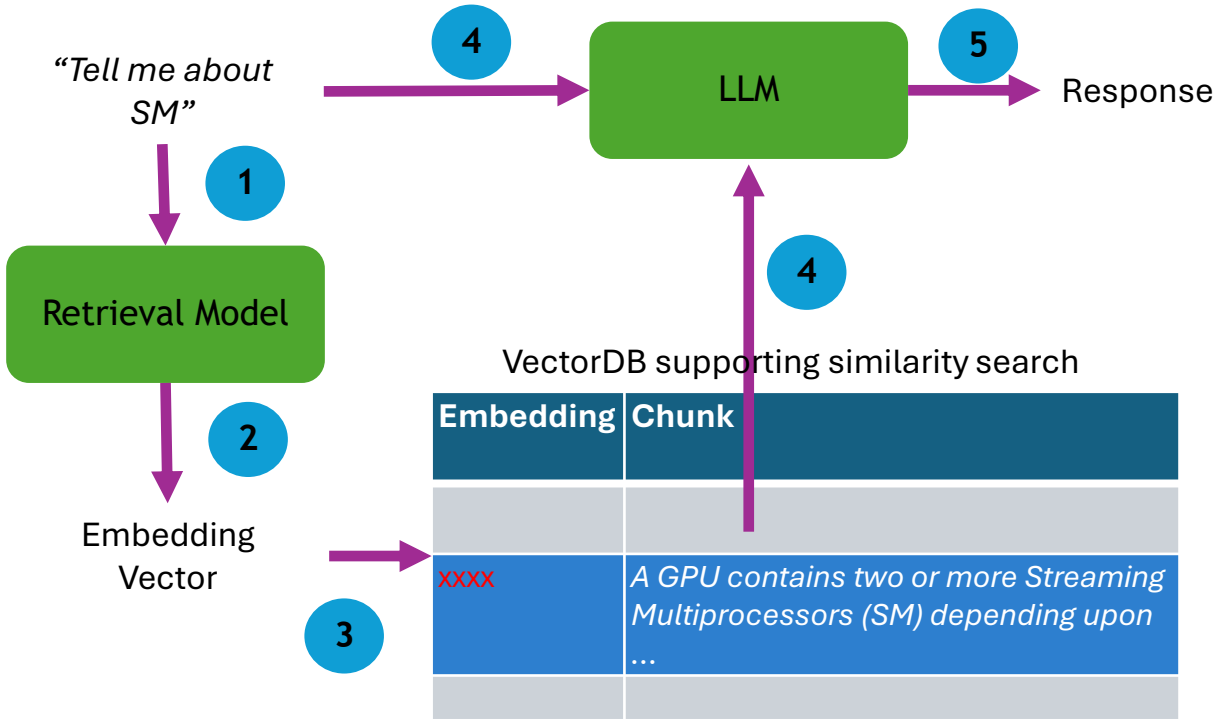
LLMs tend to hallucination

- Incorrect, outdated, conflicting, scarcity of the training data

Retrieval Augmented Generation (RAG) helps to ground LLM with curated documents

RAG leverages the search algorithm (embedding similarity “Hash”) to retrieve the relevant chunks as context for the question

Advanced RAGs use even more complicated algorithms



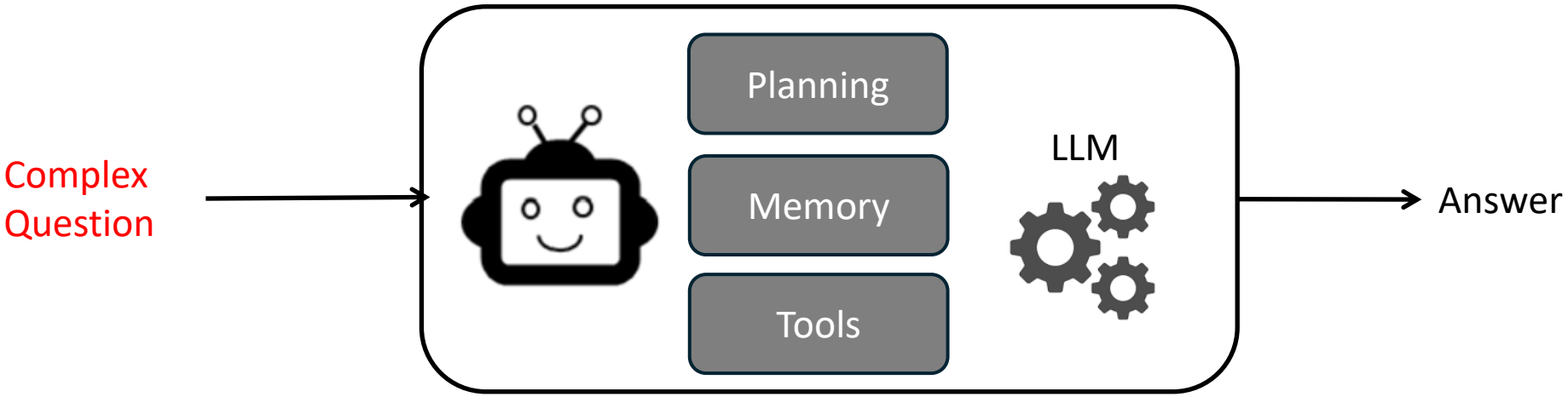
RAG Illustration

Agentic systems are algorithms to decompose complex tasks and use tools

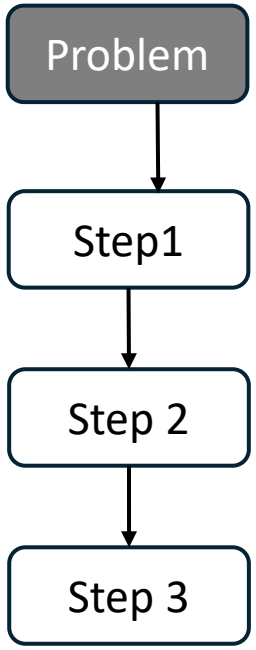
Agent handles complex tasks by problem decomposition

Planning, memory and tools form algorithms

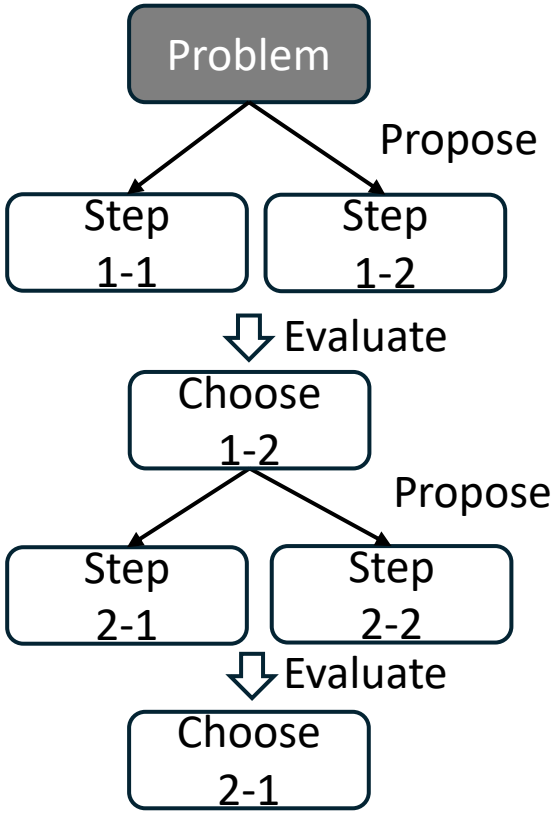
These algorithms use LLM to compute at each step



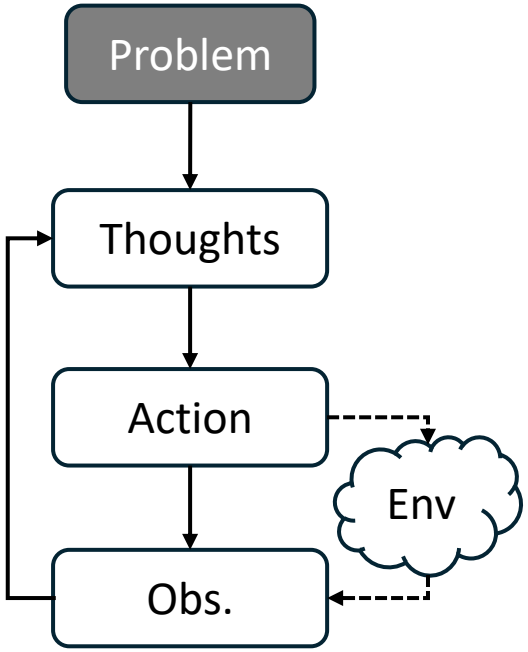
Planning/Reasoning methods are algorithms



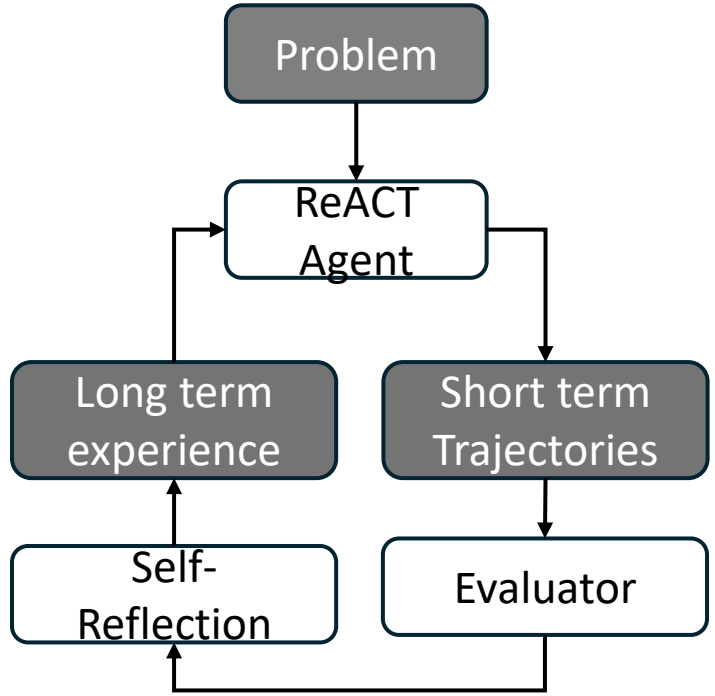
Chain-of-Thought
(Procedure)



Tree-of-Thought
(Branch)



ReACT
(Loop)



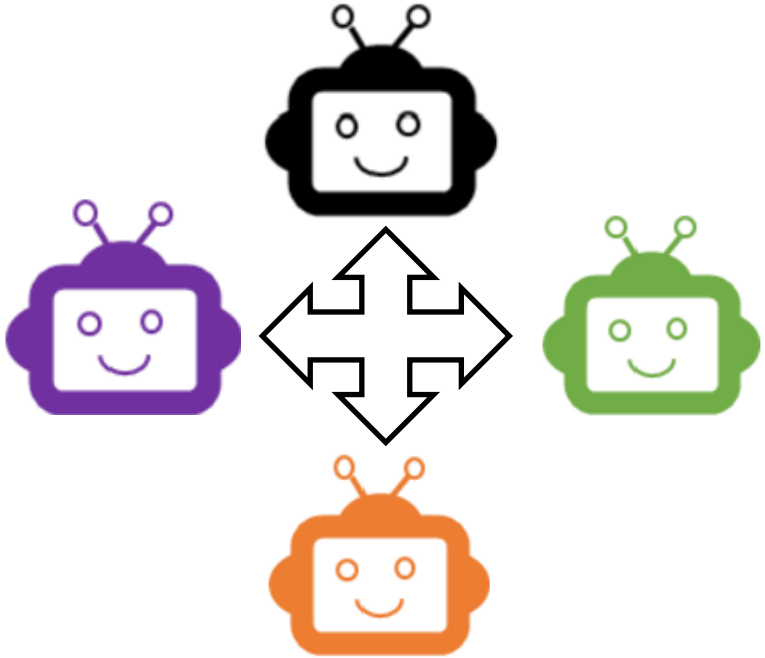
Self-Reflection
(Multi-Level Loop)

Multi-agent communication patterns are algorithms

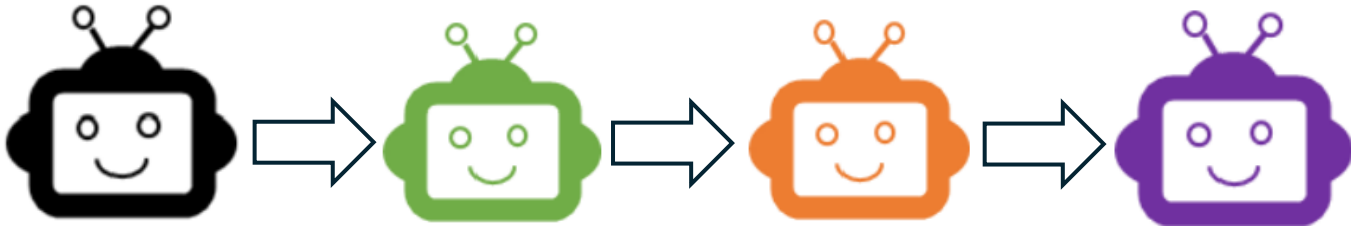
Multiple agents can talk to each other form an agentic system

Planning for multi-agent is the design of communication patterns

Communication patterns between agents are algorithms: static and dynamic patterns



Agent Conversation
(dynamic)



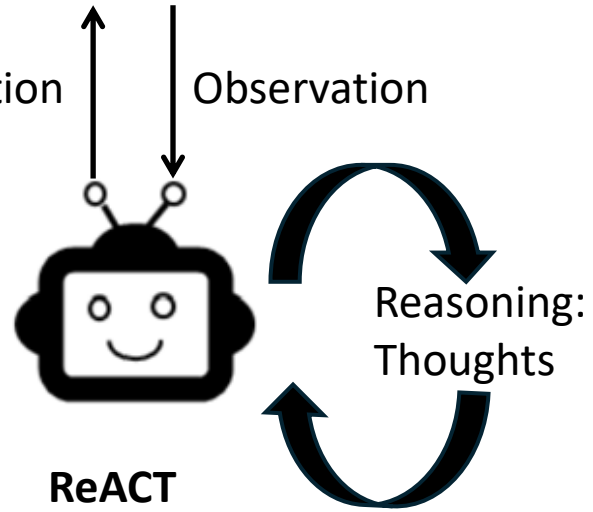
Task Flow
(static)

Report Agent: compare and summarize timing reports with CoT and ReACT

Reading many timing reports are tedious work.

Tools

timing_metric_calculation_tool :
Calculate changes in WNS, TNS, or FEP
slack_distribution_calculation_tool:
Calculate changes in the slack distribution



ReACT

Chain-of-Thought

Let's think step by step.

Firstly, analyze the change of WNS, and TNS for all designs in all PVT corners tables of these two "datecode" settings and explain with numbers using tools.

Secondly, comparing "FEP" of two datecode settings in all PVT corners tables.

Thirdly, analyze the slack distribution to identify the distribution of "slack less than 0" paths.

Finally, summarize the analysis in the following aspects:

1. *Provide key takeaways, comparison, and suggestions with bullet points of the two "datecode" settings based on previous steps.*
2. *Identify the corner which still suffers many timing violations if any.*

You need to use the provided tools to analyze the timing metrics!
You are not good at math!

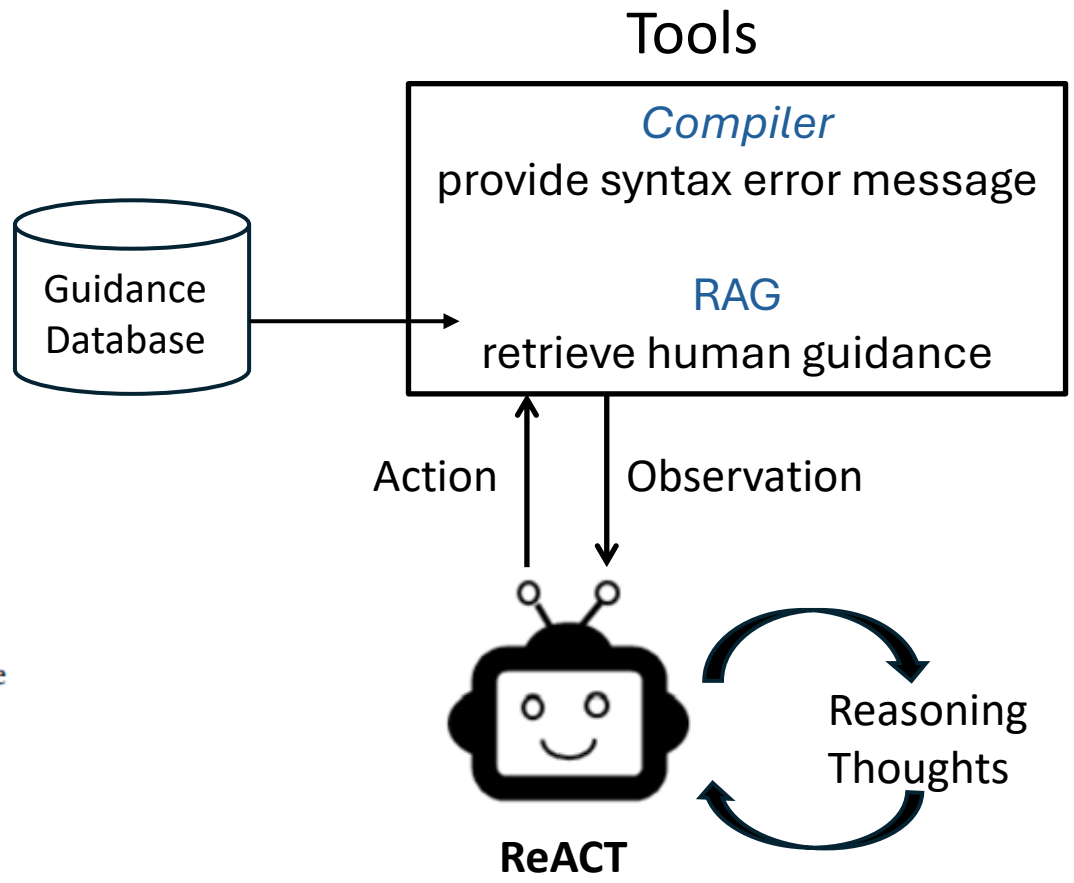
RTLFixer: fix RTL syntax errors generated by LLM with RAG

55% of GPT-3.5 Verilog generation errors are Syntax errors

Agent can fix 99% of Syntax Errors

Get feedback/error messages from Verilog compiler

Retrieve human guidance for each syntax error with **RAG**



Compiler Logs:
Object 'clk' is not declared. Verify the object name is correct. If the name is correct, declare the object.
Human Expert Guidance:
Check if 'clk' is an input. If not, and if 'clk' is used within the module, make sure the name is correct. If it's meant to trigger an 'always' block, replace 'posedge clk' with '*'.

Y.-D. Liu et al, RTLFixer: Automatically Fixing RTL Syntax Errors with Large Language Models

FVAgent : Generate SVA from natural language (NL) input with self-learned rules and multi-agents

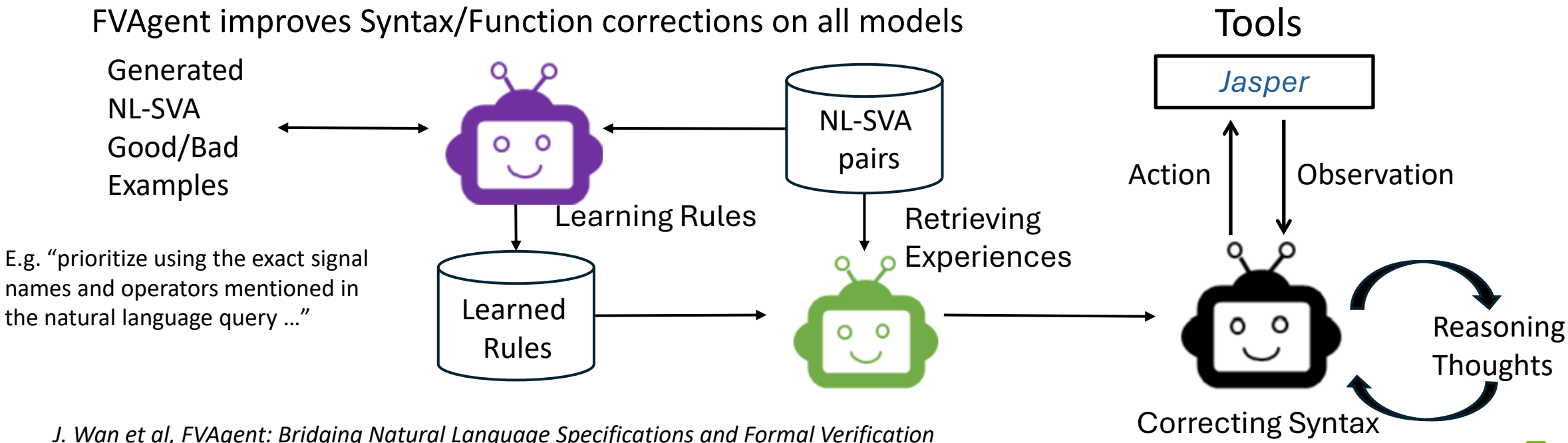
SVA generation is even harder because of less data available

Articulated ‘rules’ in the context improves the accuracy

Self-Learning: Generate rule knowledge base from training data

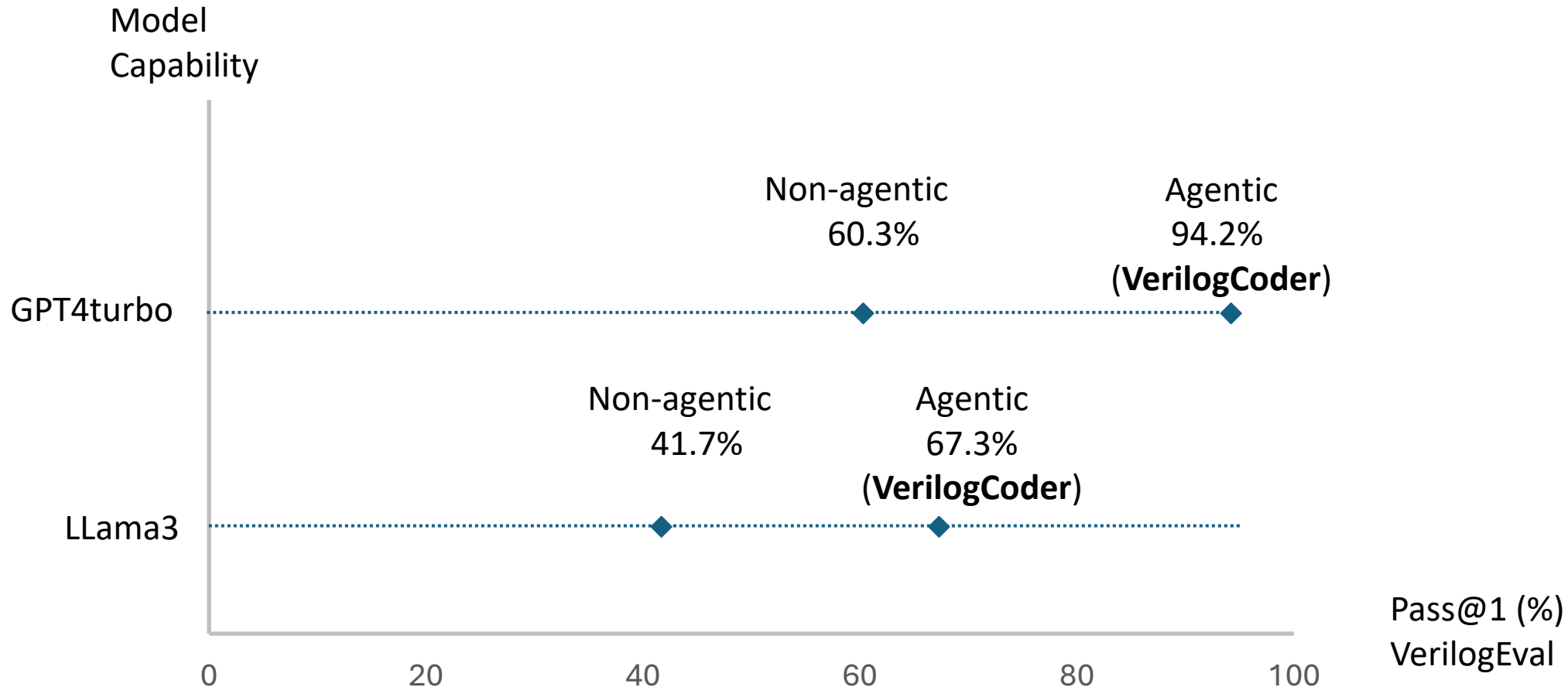
Multi-Agent task flow : rule generation/retrieval/fix syntax

FVAgent improves Syntax/Function corrections on all models



J. Wan et al, FVAgent: Bridging Natural Language Specifications and Formal Verification Assertions with Adaptive Multi-Agent Learning

VerilogCoder Agent significantly outperforms baseline LLMs in Verilog benchmark: VerilogEval



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

Task planning and Code Implementation are major stages of VerilogCoder

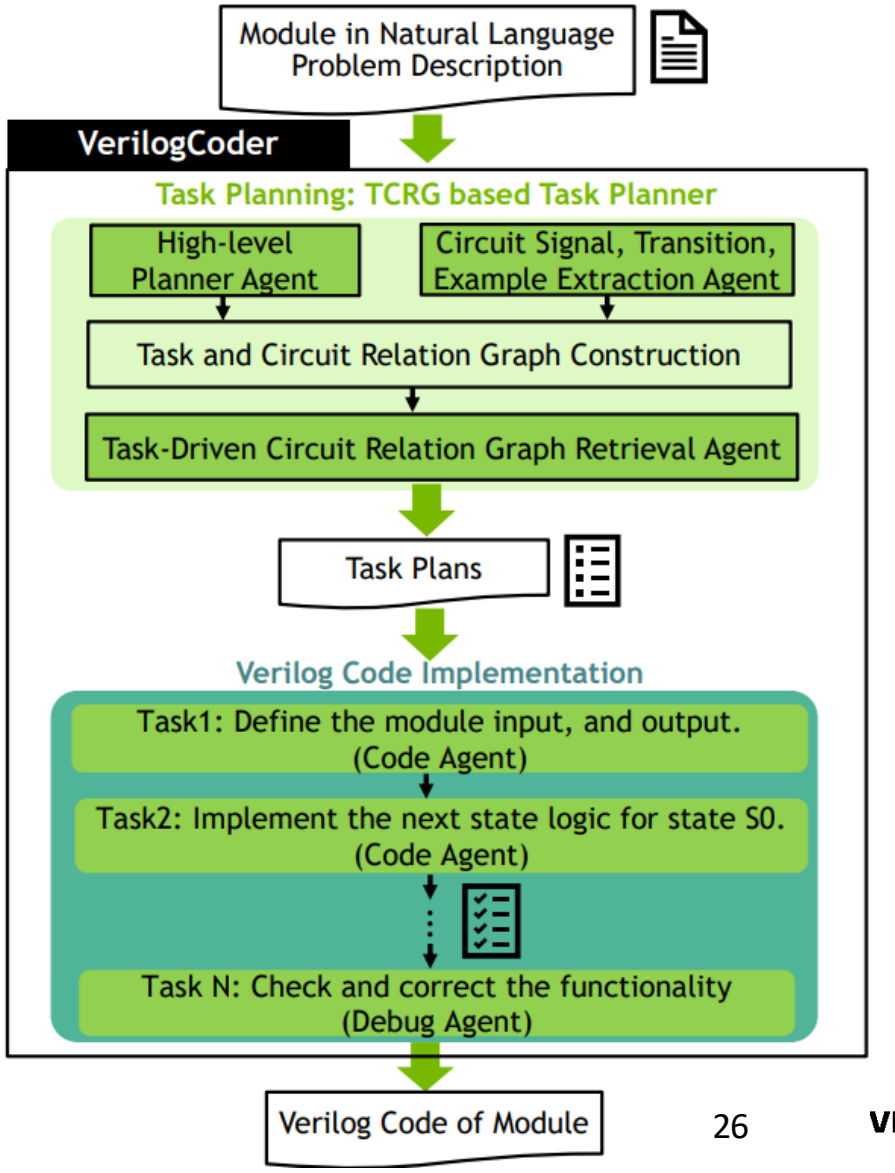
The task planning stage generates a task plan

The code implementation stage writes code for each planned task

Leverage special knowledge base and tools

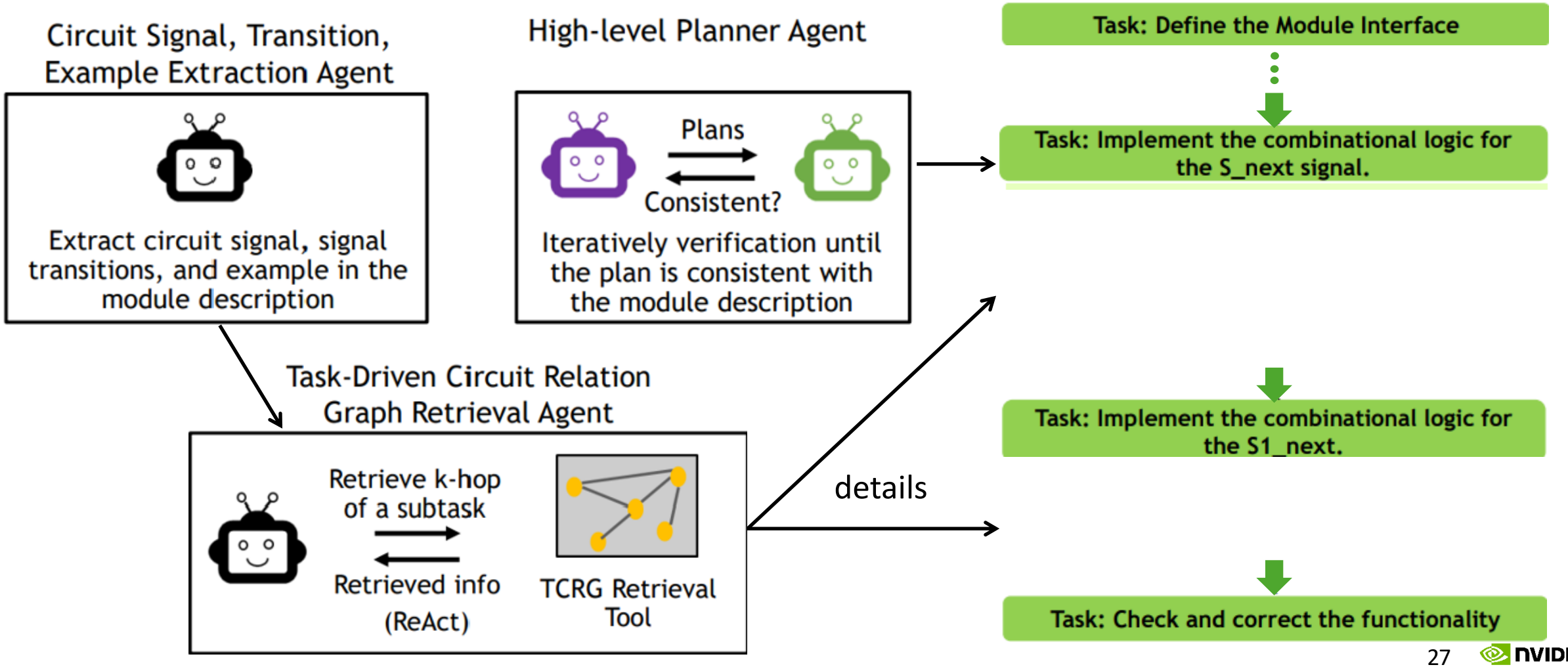
- Task Planning
  - Task-Driven Circuit Relation Graph (TCRG)
- Code Implementation
  - AST-guided waveform debugging tool

Dynamic task plan and knowledge base



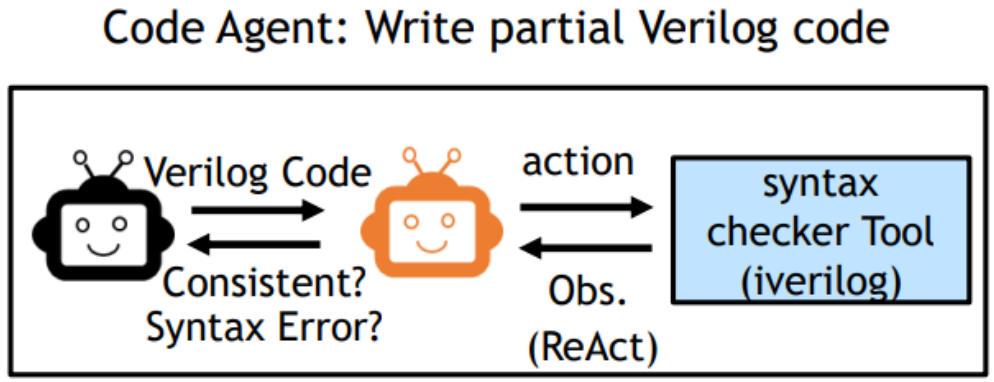
C.-T. Ho et al, VerilogCoder: Autonomous Verilog Coding Agents with Graph-based Planning and Abstract Syntax Tree (AST)-based Waveform Tracing Tool

Multi-agent planner builds Knowledge Graph to extract detailed information for task planning

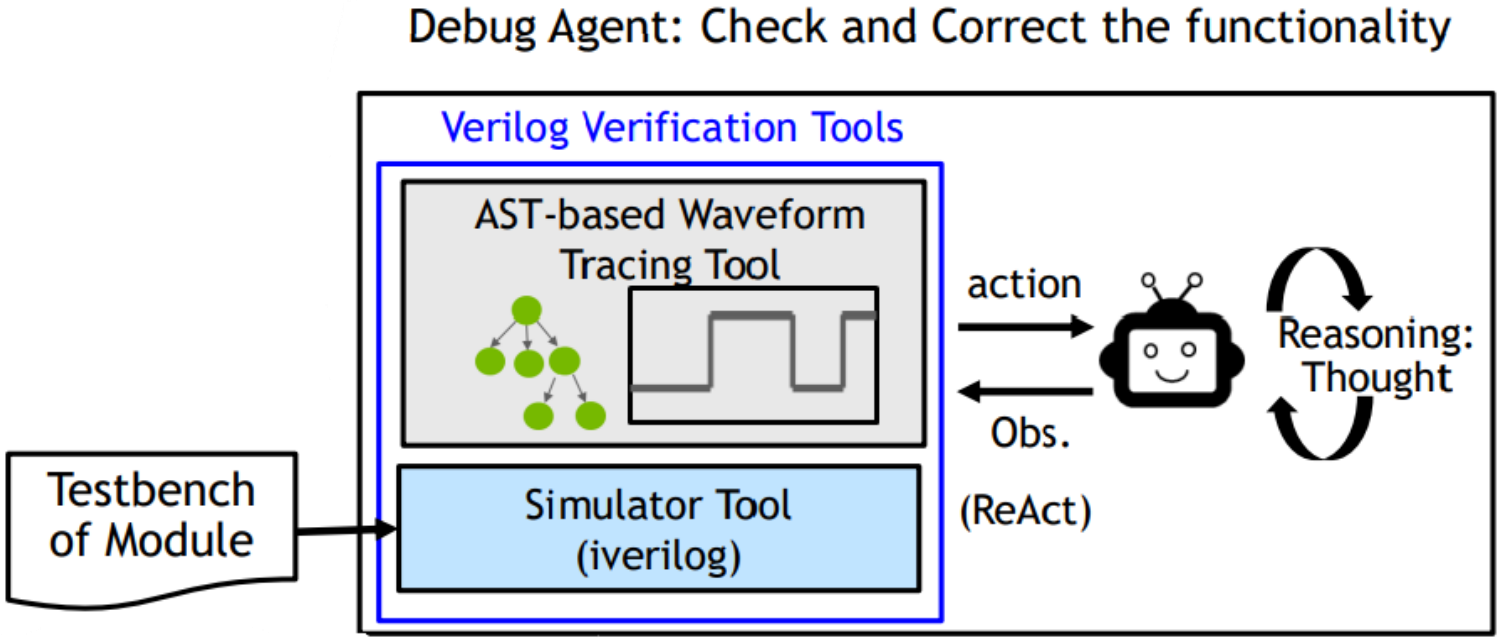


Multi-agent coder leverages syntax checker and waveform tracing tools to write and debug code

Writing code for each task



Debug final code



Mate Terminal | Mate Terminal | Mate Terminal

```
(/home/scratch.chiatungh_nvresearch/nvcell_env/llm_env) bash-4.2$ python hardware_agent/examples/verilog_eval_agent/Verilog_eval_task_flow_ver1.py
reading  /home/scratch.chiatungh_nvresearch/hardware-agent-marco/hardware_agent/examples/verilog_testcases/verilog-eval-v2/dataset_dumpall/ Prob139_2013_q2bfsm_prompt.txt
reading  /home/scratch.chiatungh_nvresearch/hardware-agent-marco/hardware_agent/examples/verilog_testcases/verilog-eval-v2/dataset_dumpall/ Prob139_2013_q2bfsm_ref.sv
reading  /home/scratch.chiatungh_nvresearch/hardware-agent-marco/hardware_agent/examples/verilog_testcases/verilog-eval-v2/dataset_dumpall/ Prob139_2013_q2bfsm_test.sv
{'llm_class': <class 'adlrchat.langchain.LLMGatewayChat'>, 'llm_kwargs': {'model_name': 'gpt-4', 'temperature': 0.0, 'top_p': 1.0}}
case manager length =  1
current test is  2013_q2bfsm
current task id is  2013_q2bfsm
current test is  2013_q2bfsm
register tool  sequential_flipflop_latch_identify_tool <function sequential_flipflop_latch_identify_tool at 0x7f65abdac040>  caller:  <autogen.agentchat.assistant_agent.AssistantAgent object at 0x7f65a93ad1b0>  executer:  <autogen.agentc
hat.user_proxy_agent.UserProxyAgent object at 0x7f65a93ad390>
Hardware Agent Initialized 1 proxy, 0 rag proxy, 2 assistants
Hardware Agent Initialized 1 proxy, 0 rag proxy, 1 assistants
user (to chat_manager):

You are a Verilog RTL designer that can break down complicated implementation into subtasks implementation plans.

[Example Begin]

### Problem

I would like you to implement a module named TopModule with the following
interface. All input and output ports are one bit unless otherwise
specified.

 - input  clk
 - input  reset
 - input  in (8 bits)
 - output out (8 bits)

The module should implement an 8-bit registered incrementer. The 8-bit
input is first registered and then incremented by one on the next cycle.

Assume all sequential logic is triggered on the positive edge of the
clock. The reset input is active high synchronous and should reset the
output to zero.

### Solution

module TopModule
(
  input  logic       clk,
  input  logic       reset,
  input  logic [7:0] in,
  output logic [7:0] out
);

  // Sequential logic

  logic [7:0] reg_out;

  always @( posedge clk ) begin
    if ( reset )
      reg_out <= 0;
    else
      reg_out <= in;
  end

  // Combinational logic

  logic [7:0] temp_wire;

  always @(*) begin
    temp_wire = reg_out + 1;
```

AI learns to generate a problem-specific algorithm

Static algorithms (predefined CoT, taskflow) work but not flexible

Dynamic algorithms (‘think step by step’, VerilogCoder task plan) are more desirable

O1 improves this capability by learning – learning to reason

Future: Make LLM learn to reason in Chip Design problem

Write a bash script that takes a matrix represented as a string with format '[1,2],[3,4],[5,6]' and prints the transpose in the same format.

Formulating the problem

Crafting the script

Reading input and parsing

Breaking down the matrix

Determining matrix dimensions

Mapping out the matrix

Constructing the output

bash Copy code

```bash
#!/bin/bash

INPUT="$1"

# Remove '[' and ']' and replace '],[' with newline
rows=$(echo "$INPUT" | sed 's/\],\[/\n/g' | sed 's/\[//g' | sed 's/\]//g')

# Read lines into array
mapfile -t lines <<< "$rows"

nrows=${#lines[@]}
# Get number of columns from the first line
cols=$(echo "${lines[0]}" | awk -F',' '{print NF}')

declare -a matrix

# Build the matrix
for ((i=0; i<nrows; i++)); do
    IFS=',' read -ra elements <<< "${lines[i]}"
    for ((j=0; j<${#elements[@]}; j++)); do
        index=$((i * cols + j))
        matrix[$index]=${elements[j]}
    done
done
```

“o1 learns to hone its chain of thought and refine the strategies it uses. It learns to recognize and correct its mistakes. It learns to break down tricky steps into simpler ones. It learns to try a different approach when the current one isn’t working. “

OpenAI, Learning to Reason with LLMs

Conclusions

AI alone can not revolutionize how we design chip today

We should deeply integrate AI with algorithms to leverage the advantages and avoid the pitfalls of either one

Explore the vast potential of integrating AI with algorithms for EDA problems: design analysis and optimization.

Exploit the revolutionary advancement of the Agentic system (an algorithm using LLM as a computing node) for design assistance

Train AI to learn to generate its own algorithms for chip design problems.