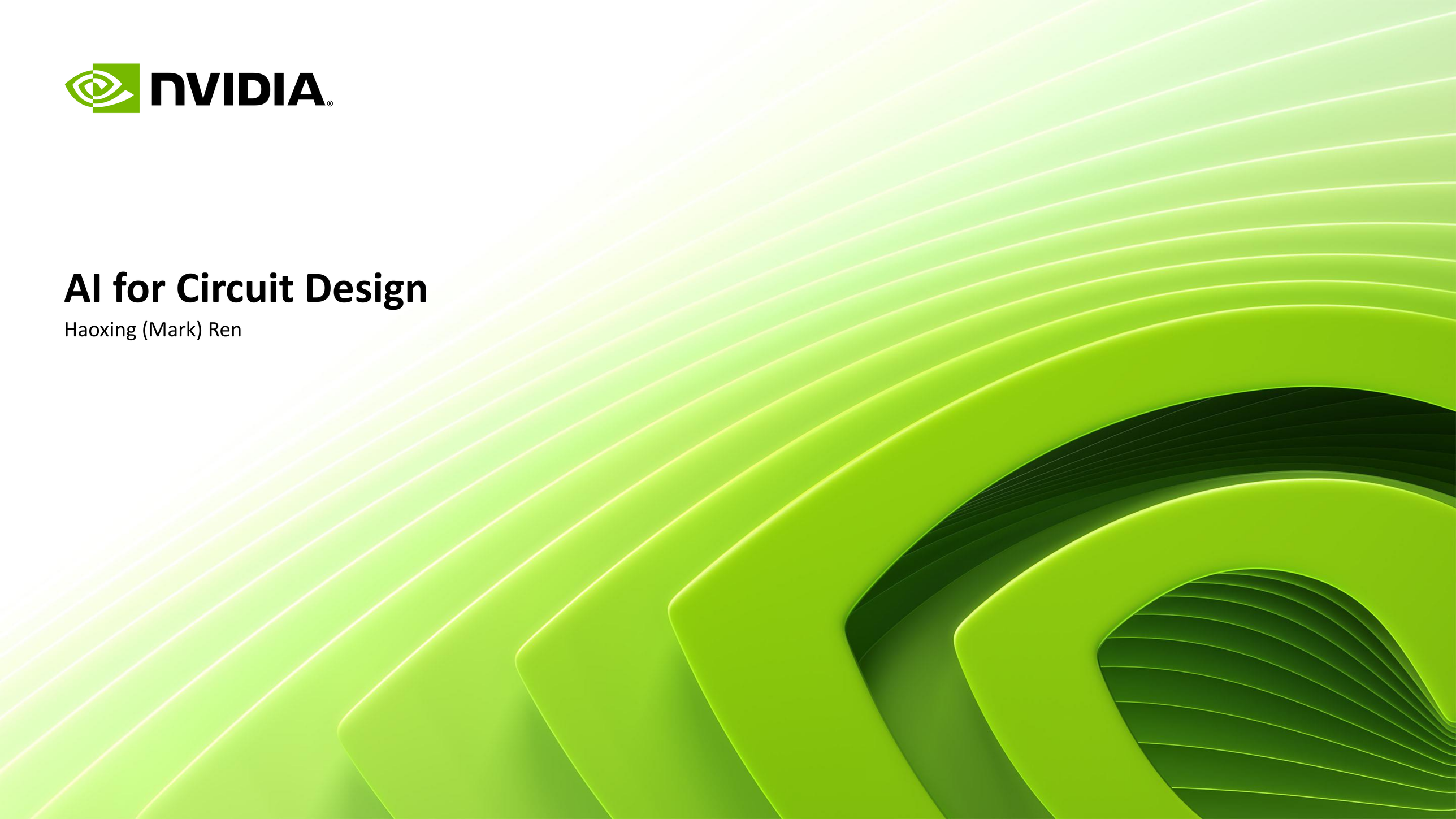
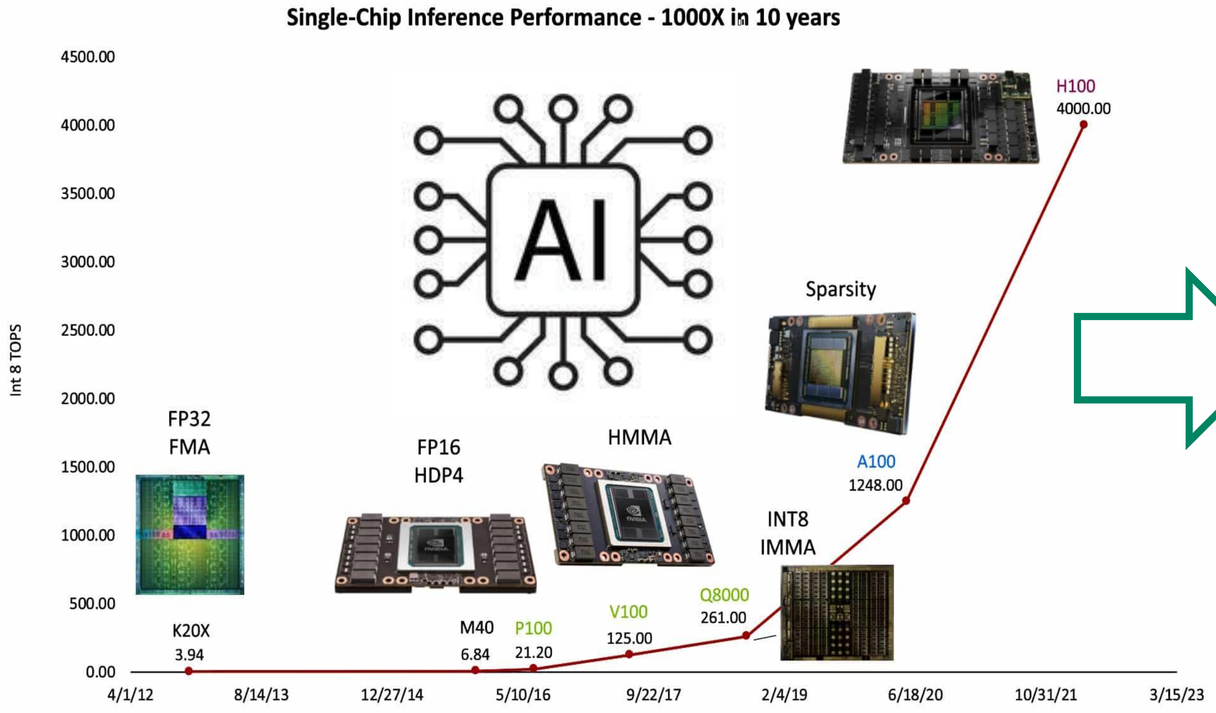


AI for Circuit Design

Haoxing (Mark) Ren

AI For Circuit Design



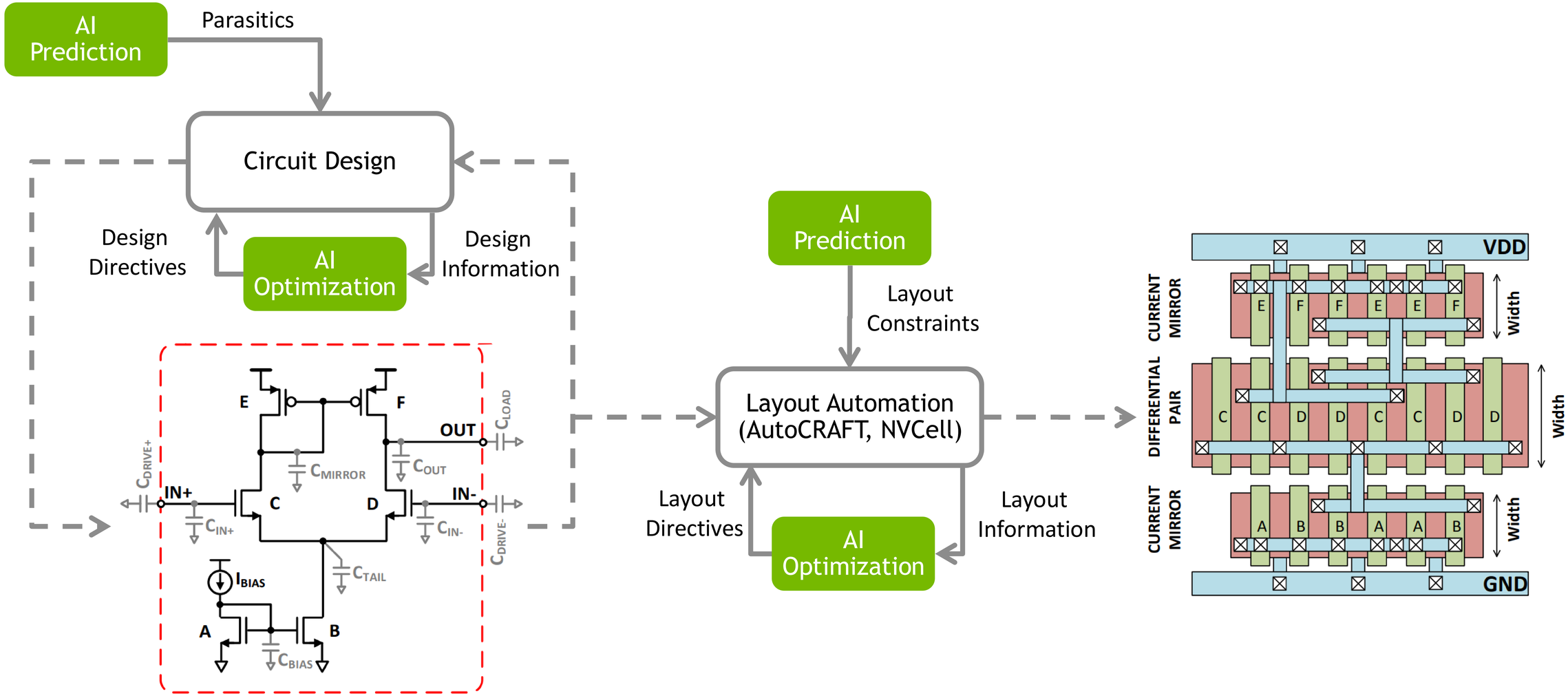
Bill Dally, Hot Chips 2023

AI Prediction and Optimization for Circuit Design

LLM-based intelligence to improve design productivity

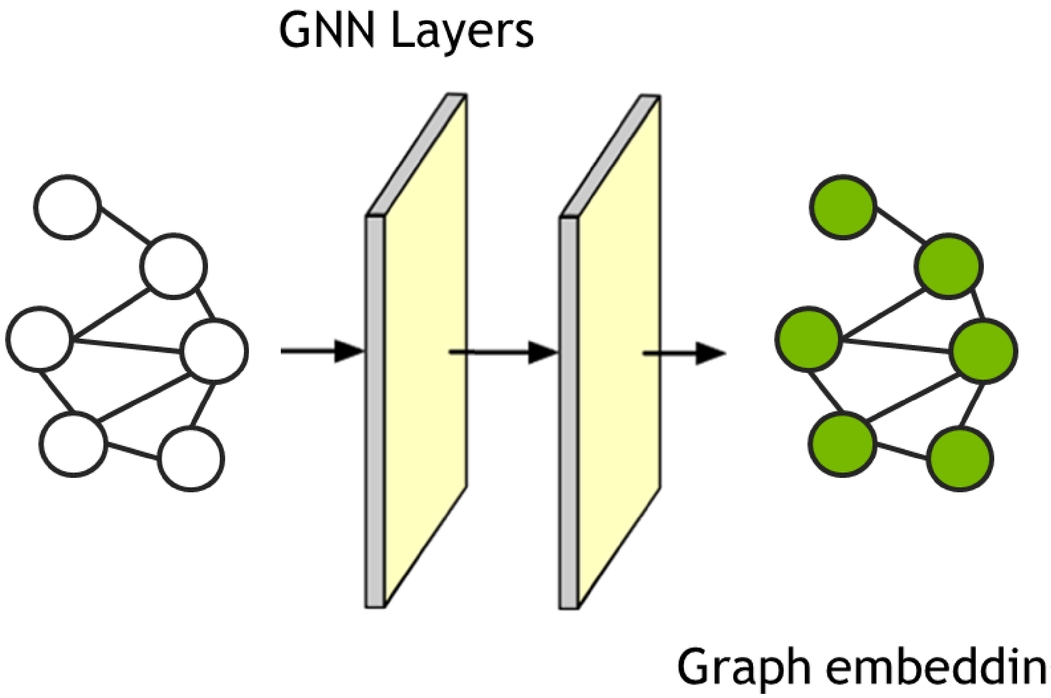
PPA & TTM

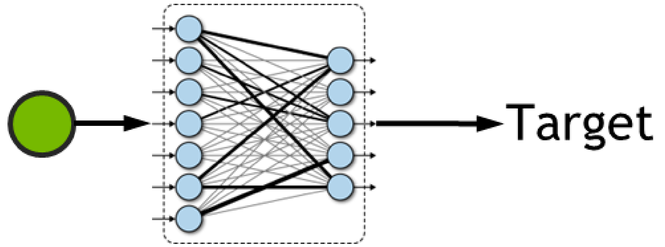
AI Prediction and Optimization for Circuit Design



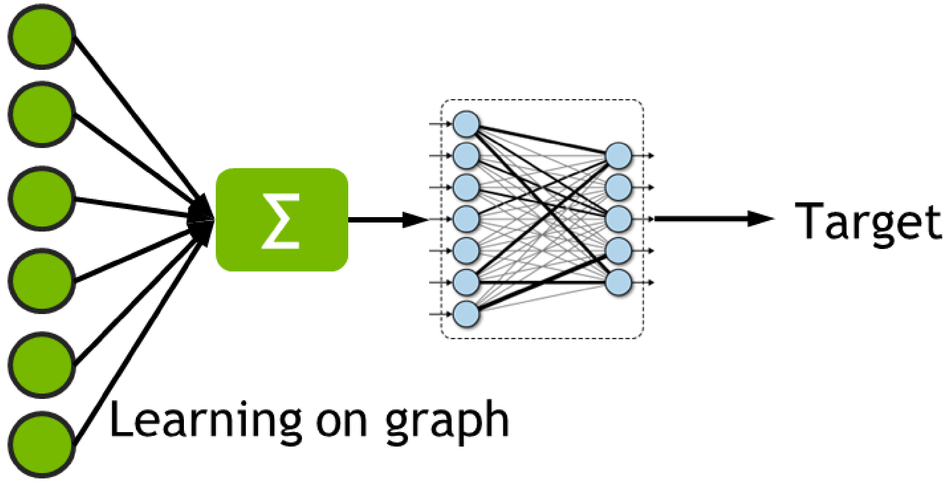
Graph Neural Network (GNN)

Commonly used model architecture for circuit design





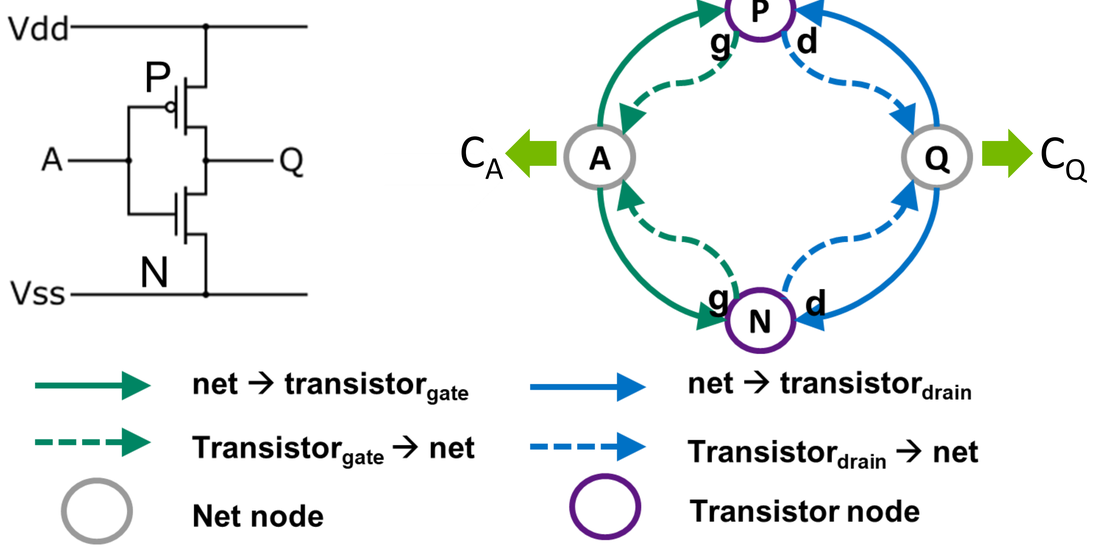
Learning on Node



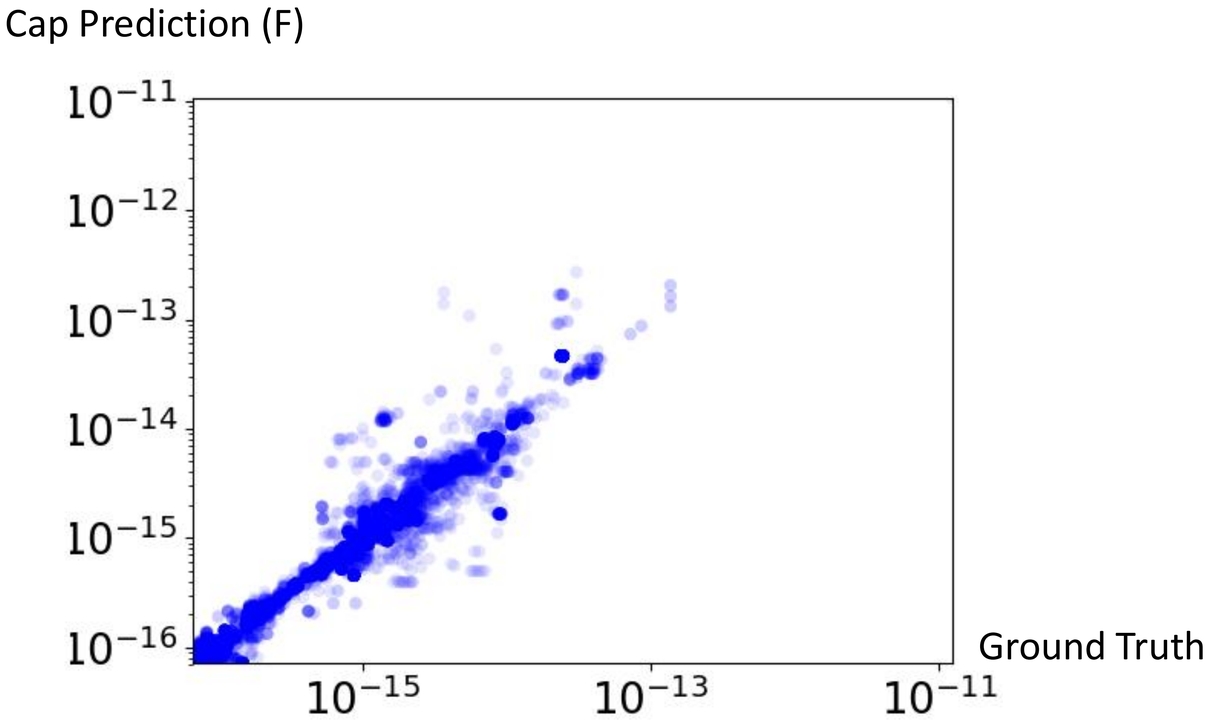
Learning on graph

Post Layout Parasitics Prediction

Construct GNN with schematic graph



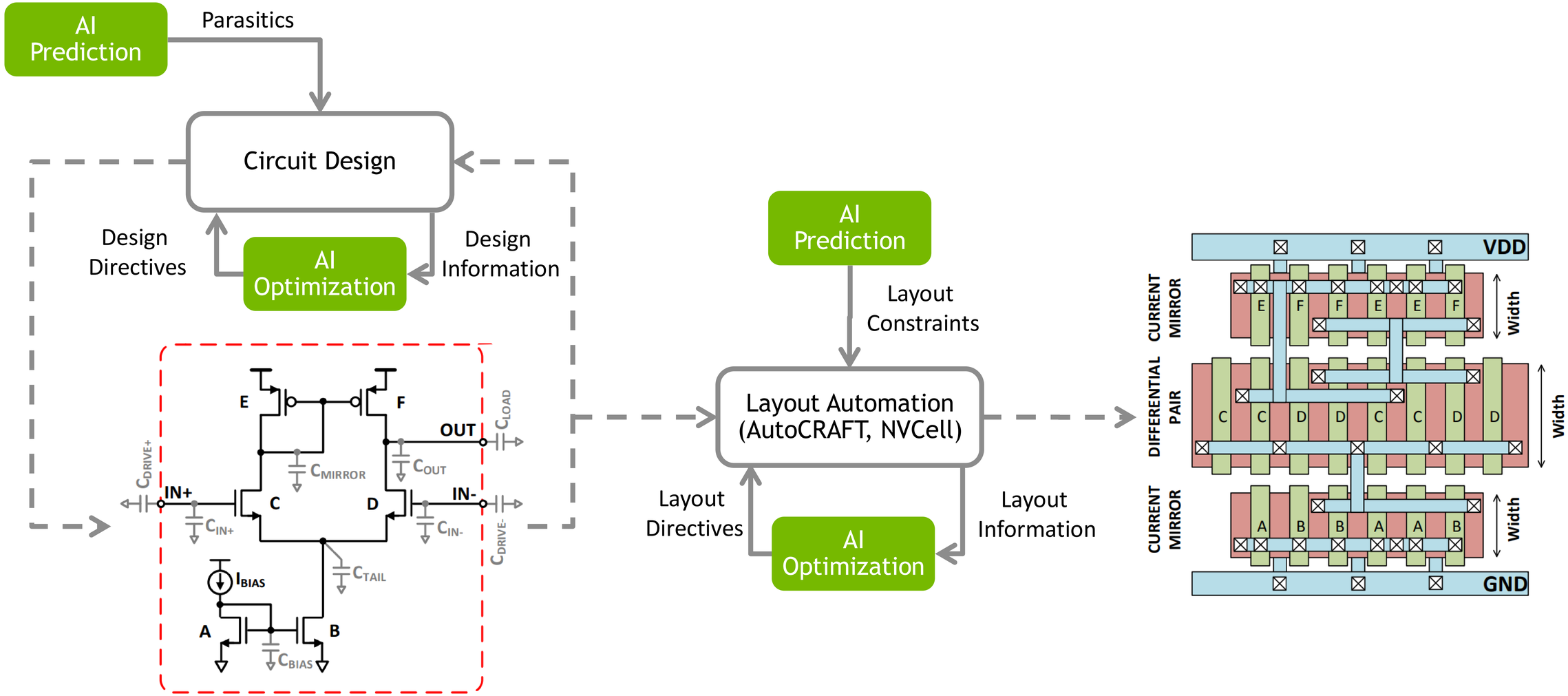
Circuit Schematics to Graph Conversion
Model trained on over 1M devices



MAE=0.852fF MAPE=15%
Simulation error reduced to <10%

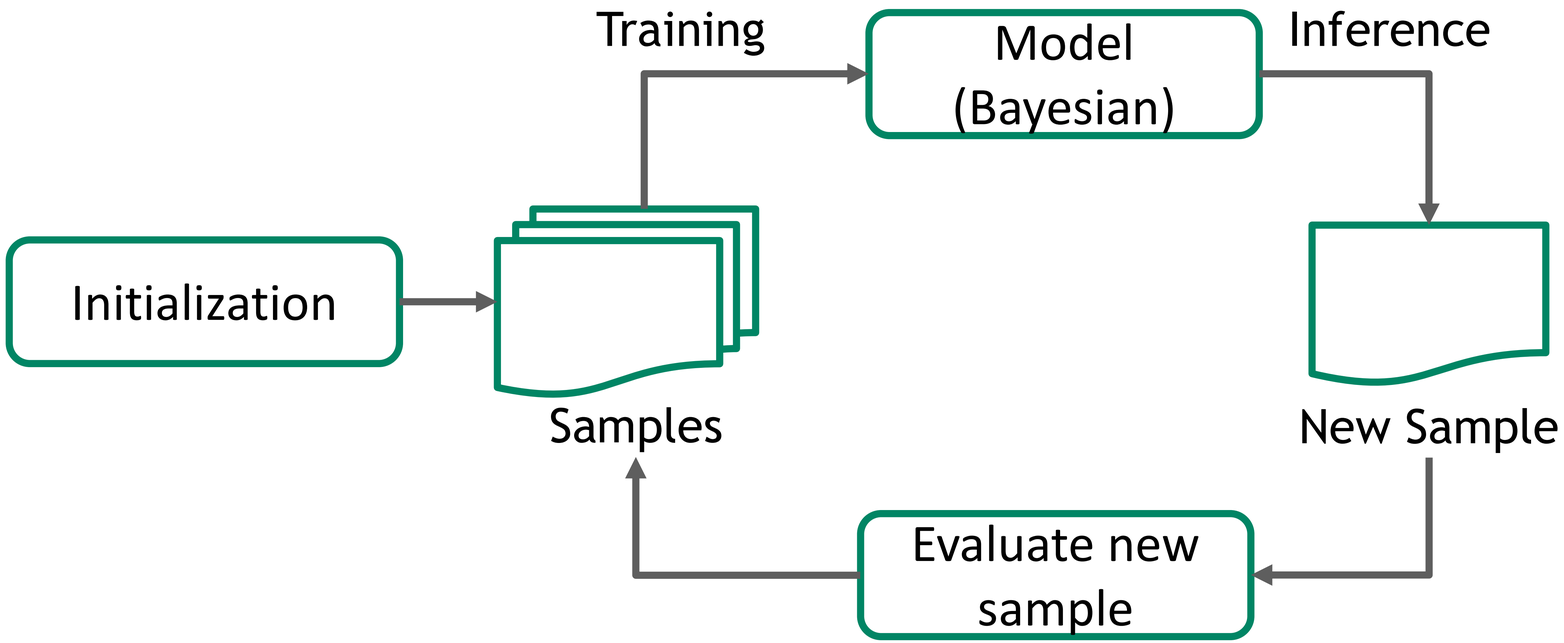
H. Ren et al., "ParaGraph: Layout Parasitics and Device Parameter Prediction using Graph Neural Networks", DAC 2020. (Research funded under Cadence's DARPA IDEA contract)

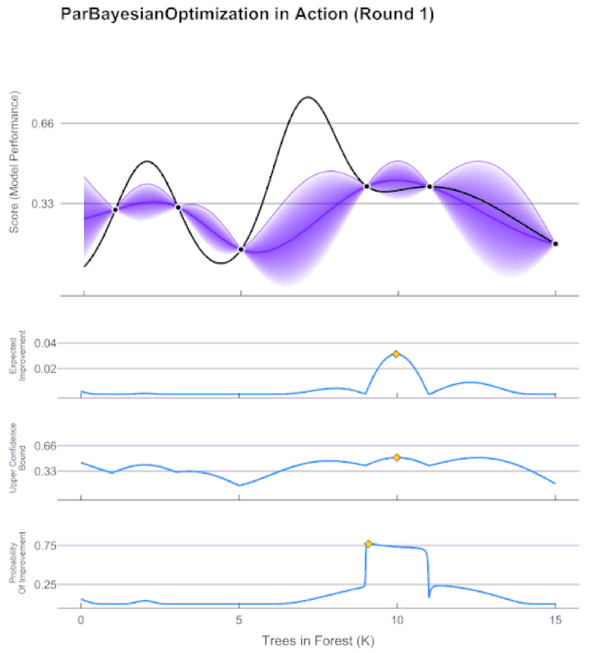
AI Prediction and Optimization for Circuit Design



Design Optimization with Bayesian Optimization (BO)

Sampling-based black-box optimization



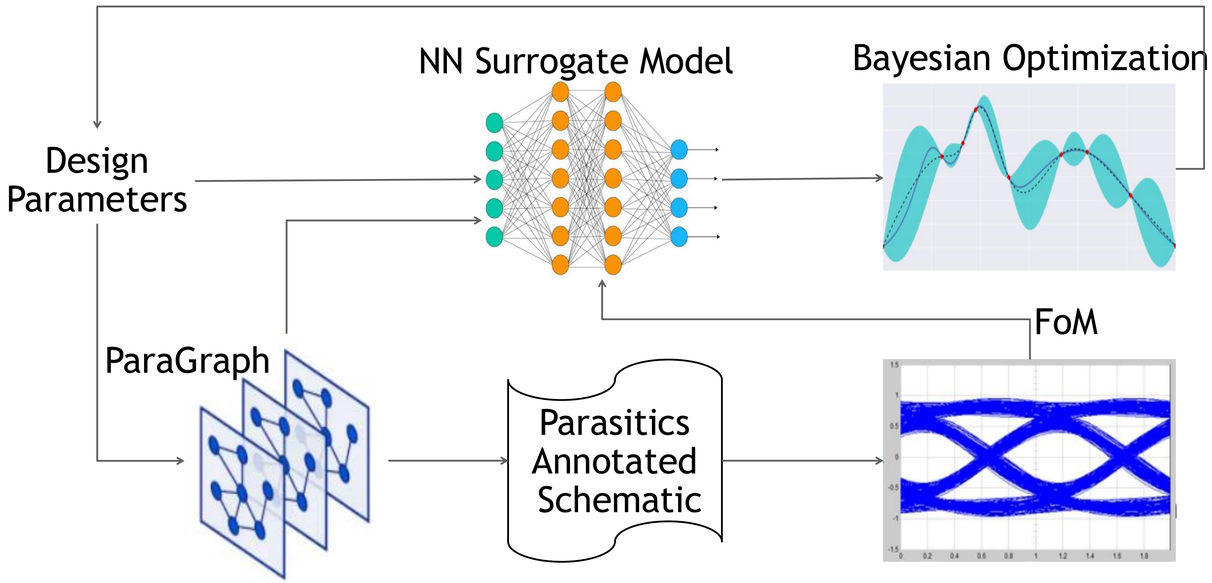


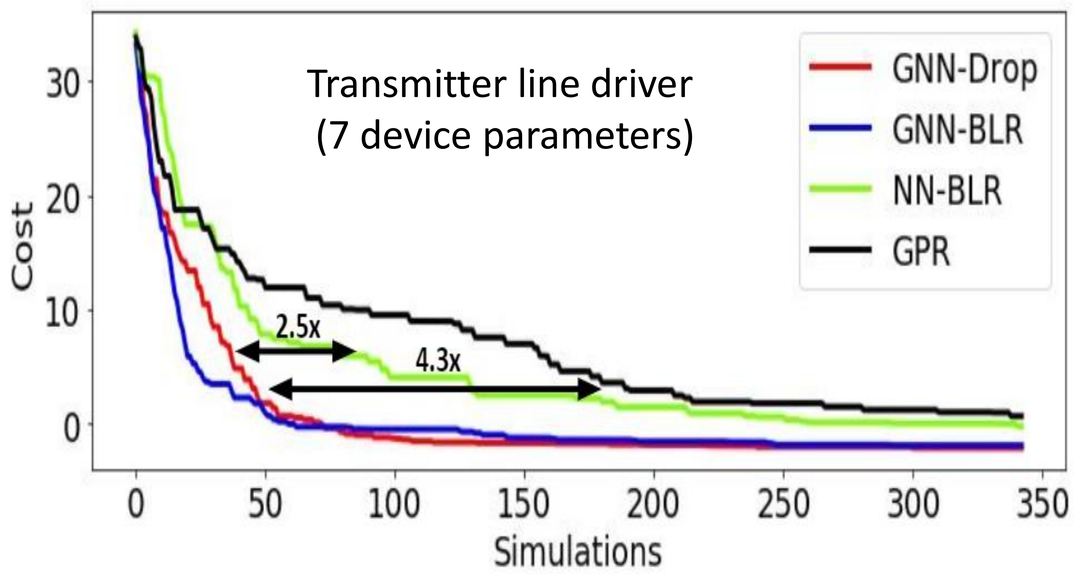
Build a probability model of the objective function and use it to select the most promising data to evaluate in the true objective function.

Circuit Sizing with BO

Leverage parasitics predictor in modeling and simulation

NN model predict simulation performance with design parameters and ParaGraph embedding





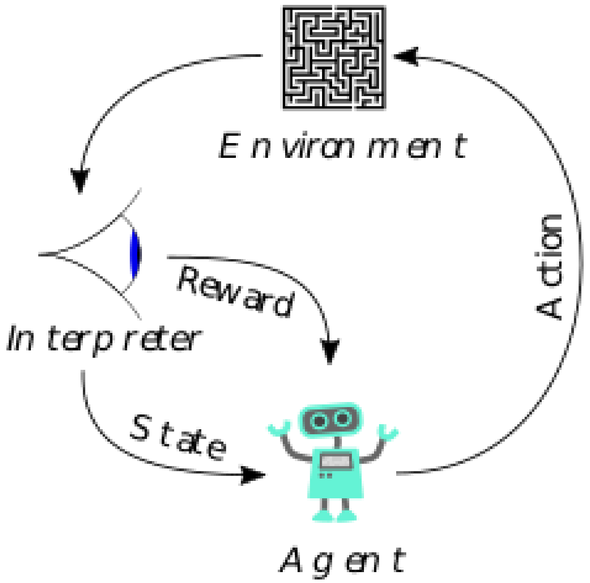
M. Liu et al., "Parasitic-Aware Analog circuit Sizing with Graph Neural networks and Bayesian Optimization", DATE 2021

Design Optimization with Reinforcement Learning

Convert optimization problem to RL game

Minimize $f(x_1,x_2,x_3, \ldots, x_n)$

Subject to $g(x_1,x_2,x_3, \ldots, x_n)$



State: $x_1,x_2,\ldots,x_{k-1}$
Action: x_k

State: $x_1,x_2,\ldots,x_n$
Action: $T_k(x_1,x_2,\ldots,x_n) \rightarrow x'_1,x'_2,\ldots,x'_n$

Reward: $f(x_1,x_2,x_3,\ldots)$, $g(x_1,x_2,x_3,\ldots)$
or $f`(x_1,x_2,\ldots,x_i)$, $g`(x_1,x_2,\ldots,x_i)$

NVIDIA

Circuit Optimization with RL

Optimize Vt type, channel length, stack height, multiplicity, Res/Cap value

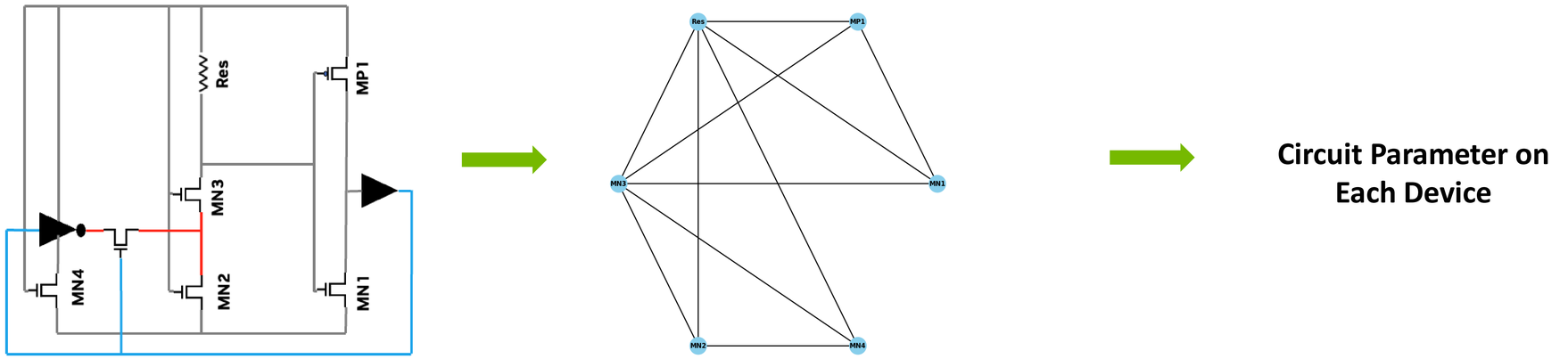
Convert circuit into graph and generate action prediction for each node.

Example: power-on detector (9 devices) , optimize for target voltage thresholds and minimize variation

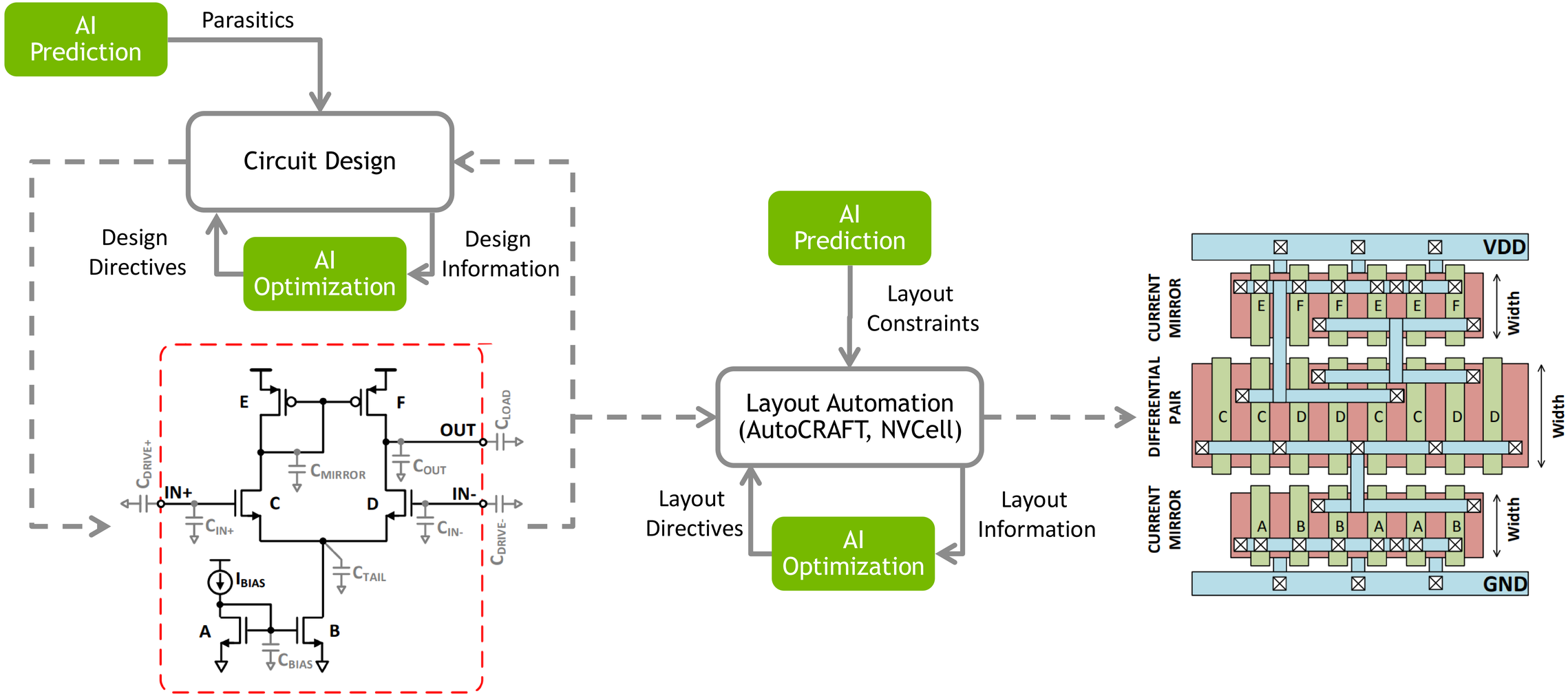
- Baseline: Vth-down = 411mV, variation = 209mV, Vth-up = 650mV, variation = 217mV
- Target: Vth-down = 380mV, variation < 135mV, Vth-up = 650mV, variation < 135mV
- Optimized: Vth-down = 380mV, variation < 135mV, Vth-up = 650mV, variation < 135mV

RL training takes about 8000 simulations (overnight)

Model transferability (simulation reduced to 3000)

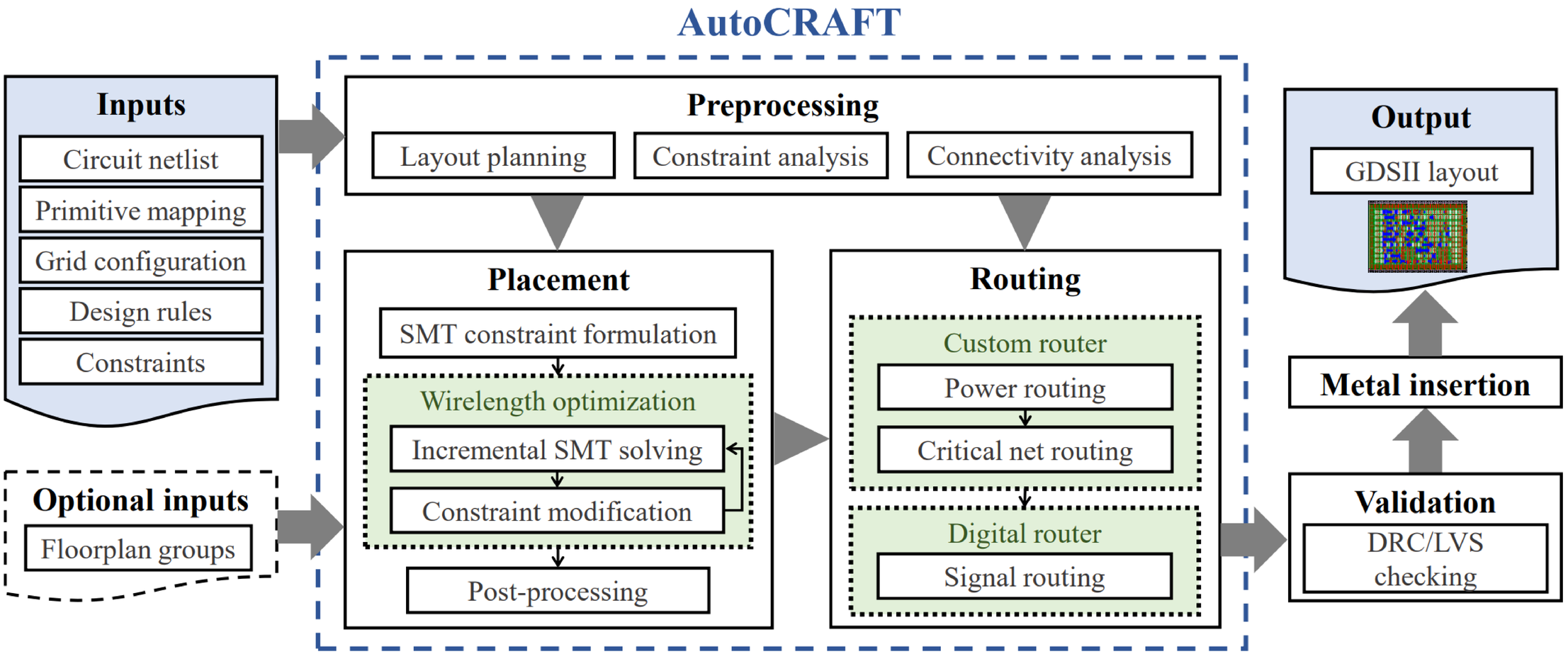


AI Prediction and Optimization for Circuit Design



Custom Circuit Layout Automation

Honor Custom Constraints with SMT Solver

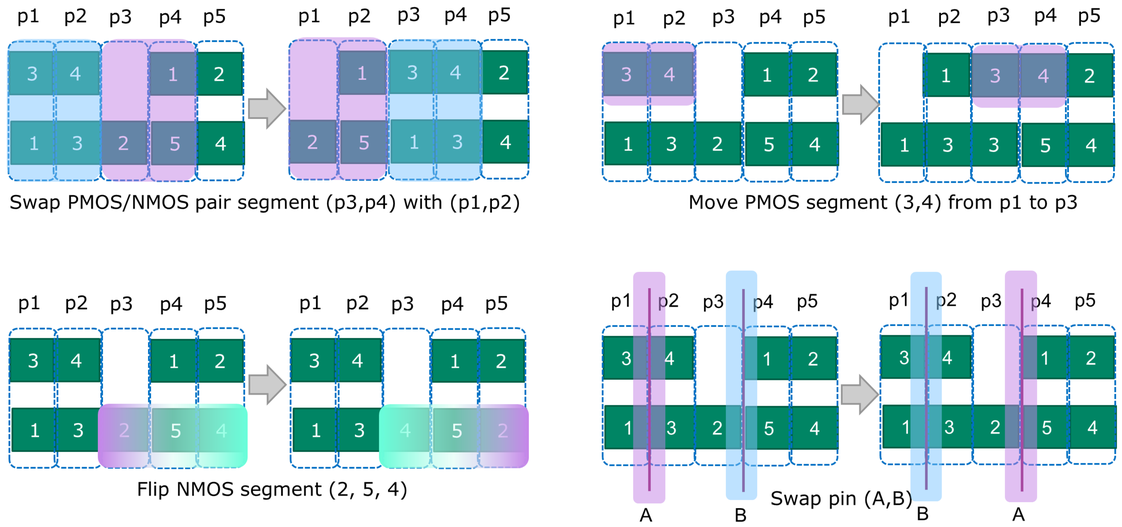


H. Chen et al, AutoCRAFT: Layout Automation for Custom Circuits in Advanced FinFET Technologies, ISPD 2022
Collaboration with D. Pan from UT Austin (https://github.com/NVlabs/AutoCRAFT)

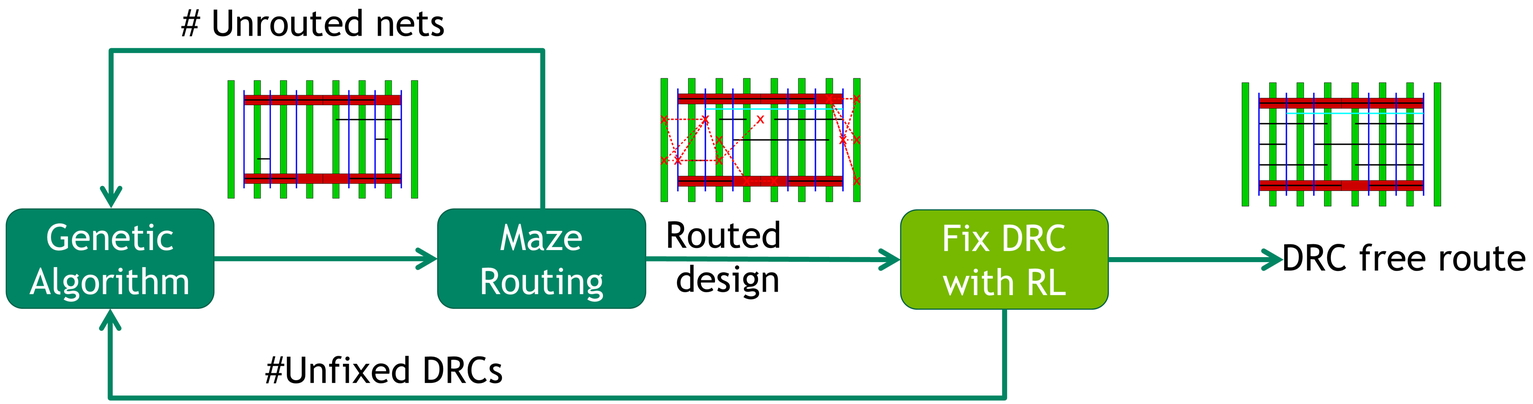
Standard Cell Layout

NVCell

Device Placement

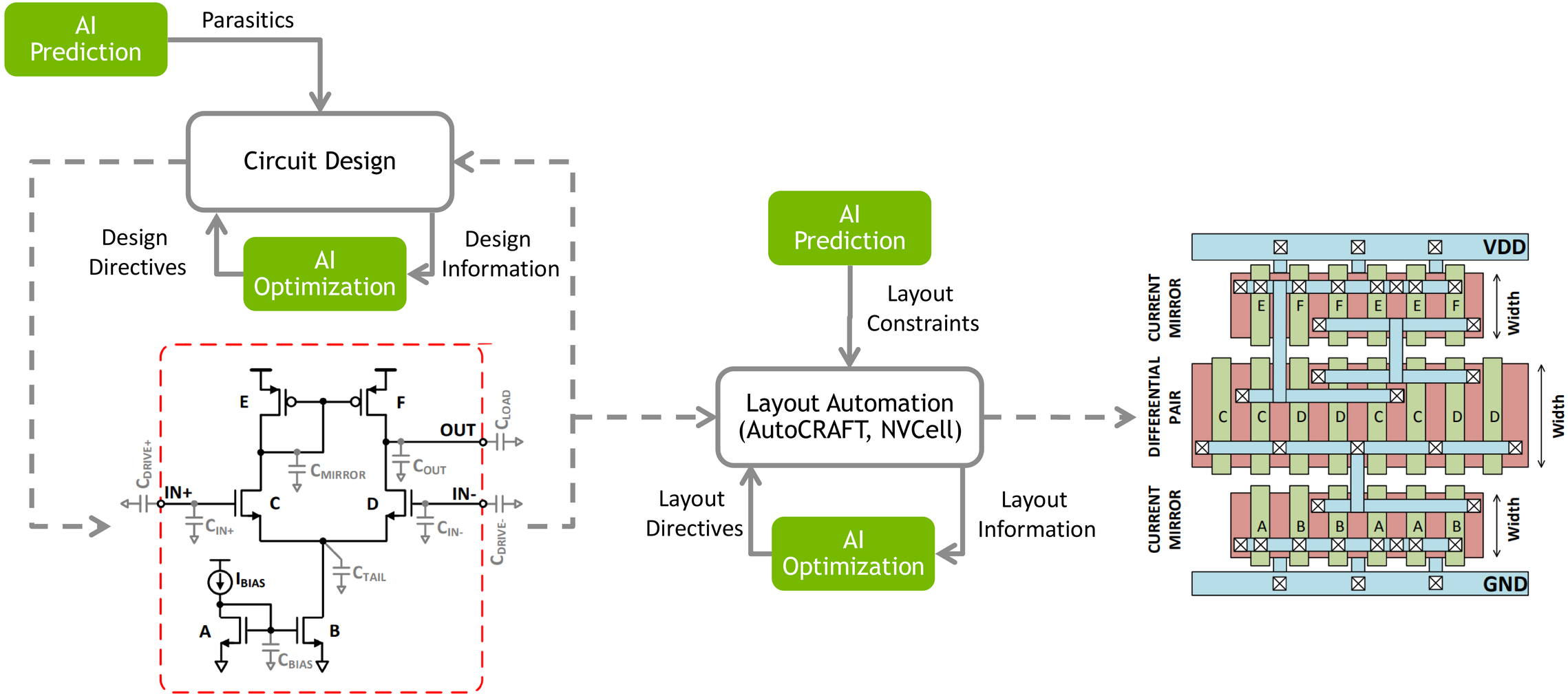


Device Routing



Ren et al, Standard Cell Routing with Reinforcement Learning and Genetic Algorithm in Advanced Technology Nodes, ASPDAC 2021
Ren et al, Generate Standard Cell Layout in Advanced Technology Nodes with Reinforcement Learning, DAC 2021

AI Prediction and Optimization for Circuit Design



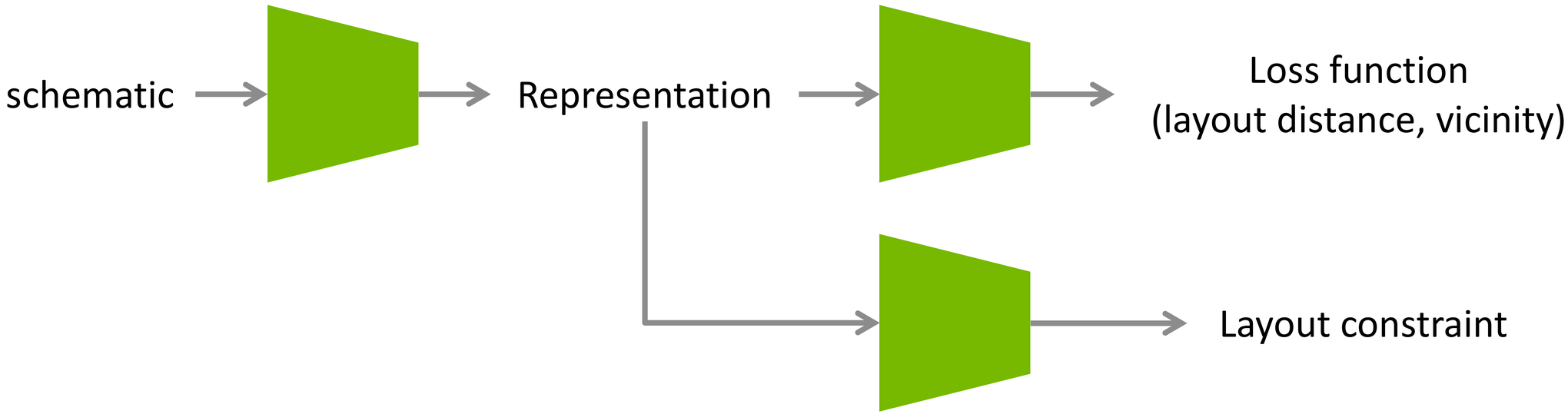
Learning from Layout with Representation Learning

Don’t have lots of layout constraint labels

But do have lots of schematic-layout pairs

Learning a representation from a large amount of layout data

Predict layout constraints from learned device representation



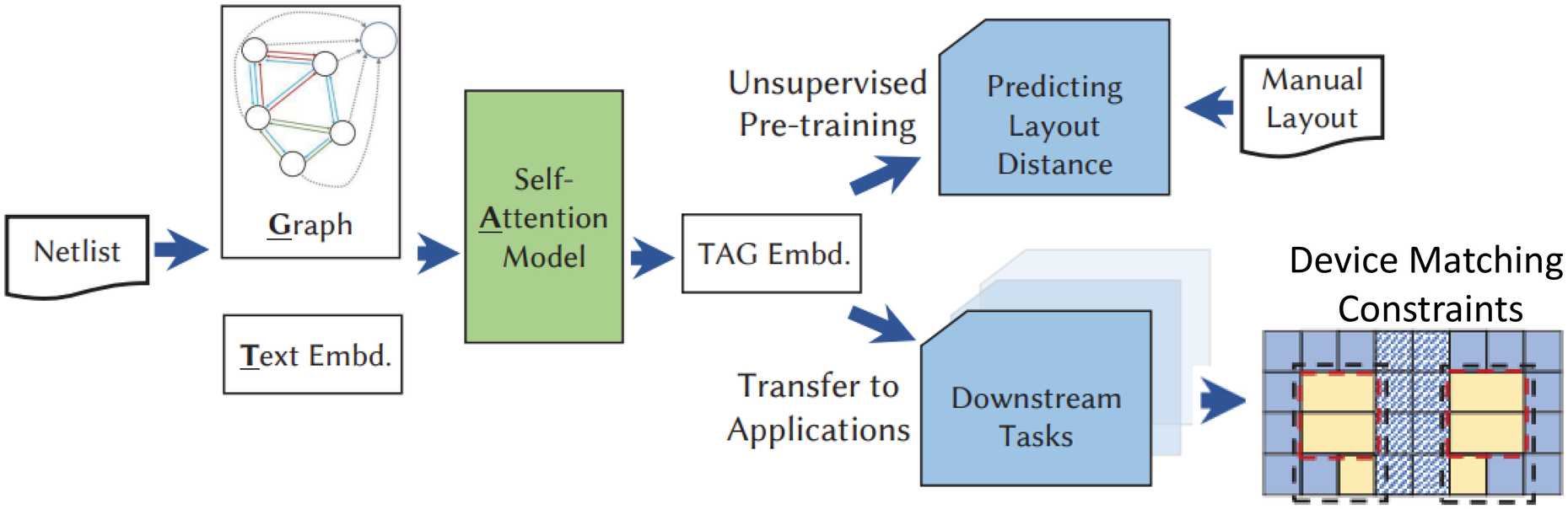
Device Matching Prediction For Custom Circuit

Using GNN/Text/Attention to learn device representation

Leverage both graph (hierarchical) as well as text embedding and self-attention module to generate device representation

Training on pair-wise distance loss

Use device representation to predict device matching constraints for AutoCRAFT



K. Zhu et al, TAG: Learning Circuit Spatial Embedding From Layouts, ICCAD 2022

Device Clustering Constraint For Cell Layout

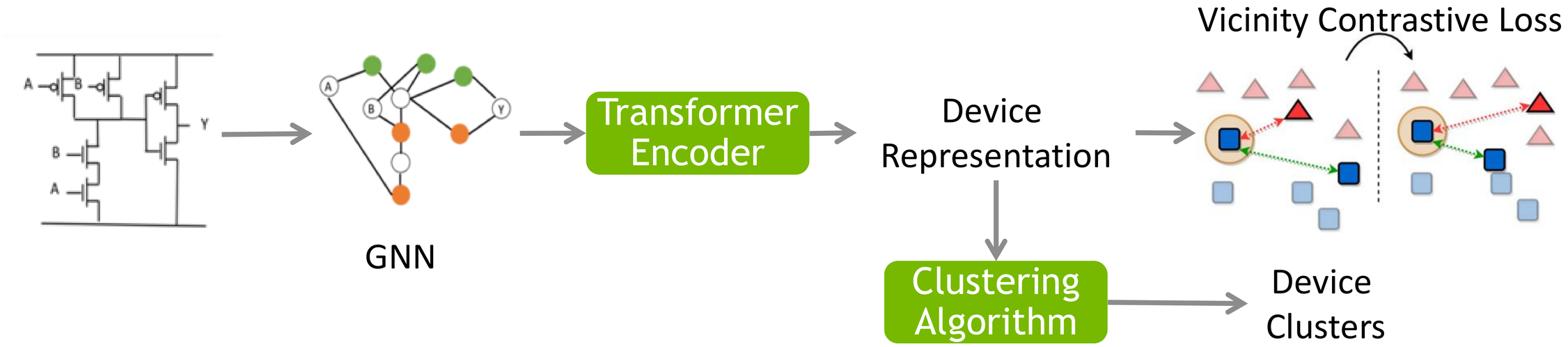
Use GNN/Transformer model to learn device representation

GNN/Transformer-based encoder to generate representation on each device

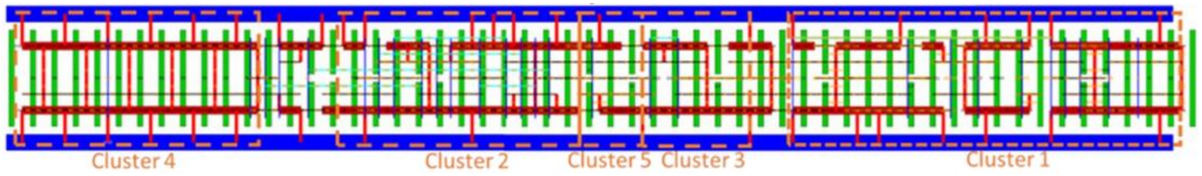
Training on a contrastive affinity loss

Use clustering algorithm on the embedding space to generate device clusters for NVCell

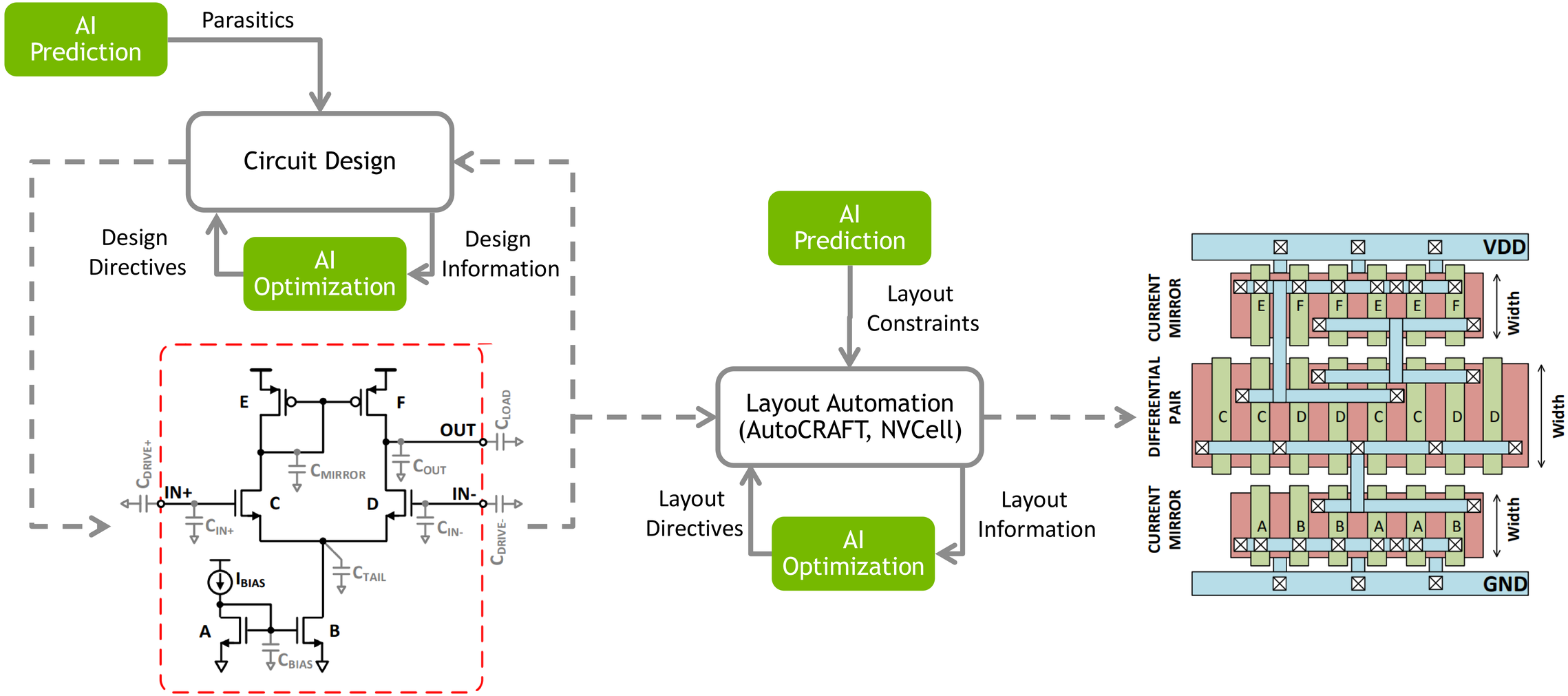
5% more LVS/DRC clean layouts, 12.7× faster run time on difficult cells



C.-T. Ho et al, Novel Transformer Model Based Clustering Method for Standard Cell Design Automation, ISPD 2024

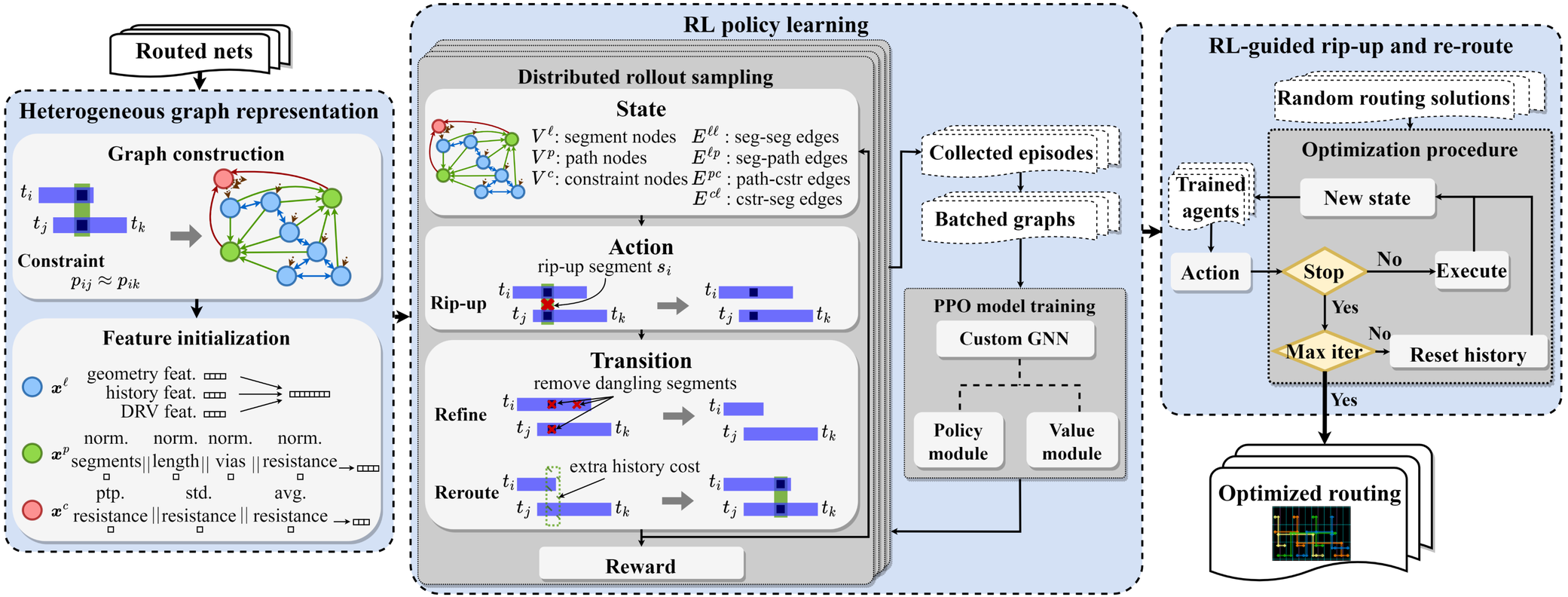


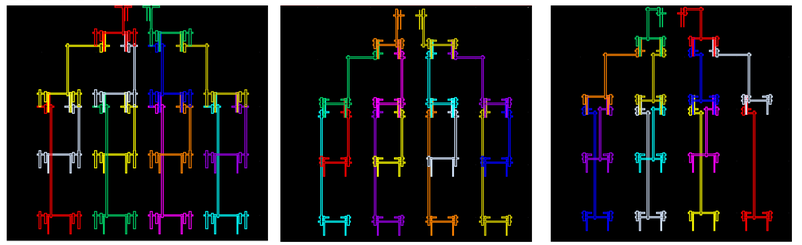
AI Prediction and Optimization for Circuit Design



AI for Detailed Routing

Using RL to optimize wire matching





H. Chen, Reinforcement Learning Guided Detailed Routing for Custom Circuits, ISPD 2023

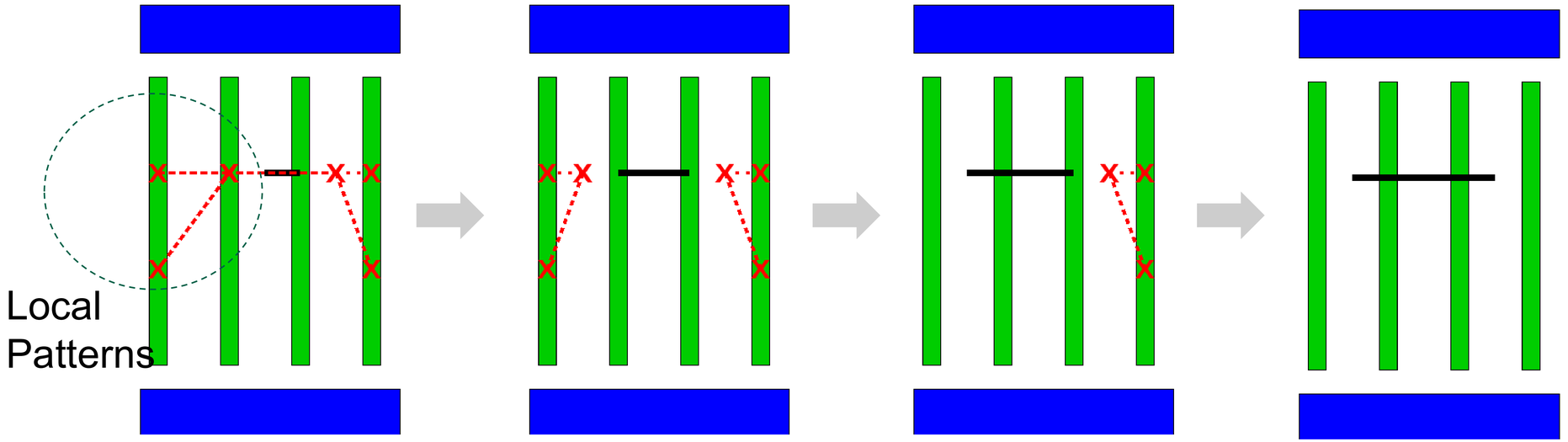
AI to Fix DRC

Convert Layout DRC Fix Process to a RL GAME

Action: adding additional M0 grid to reduce DRCs

State: current layout images

Reward: DRC reduction



Step 0: DRC = 6

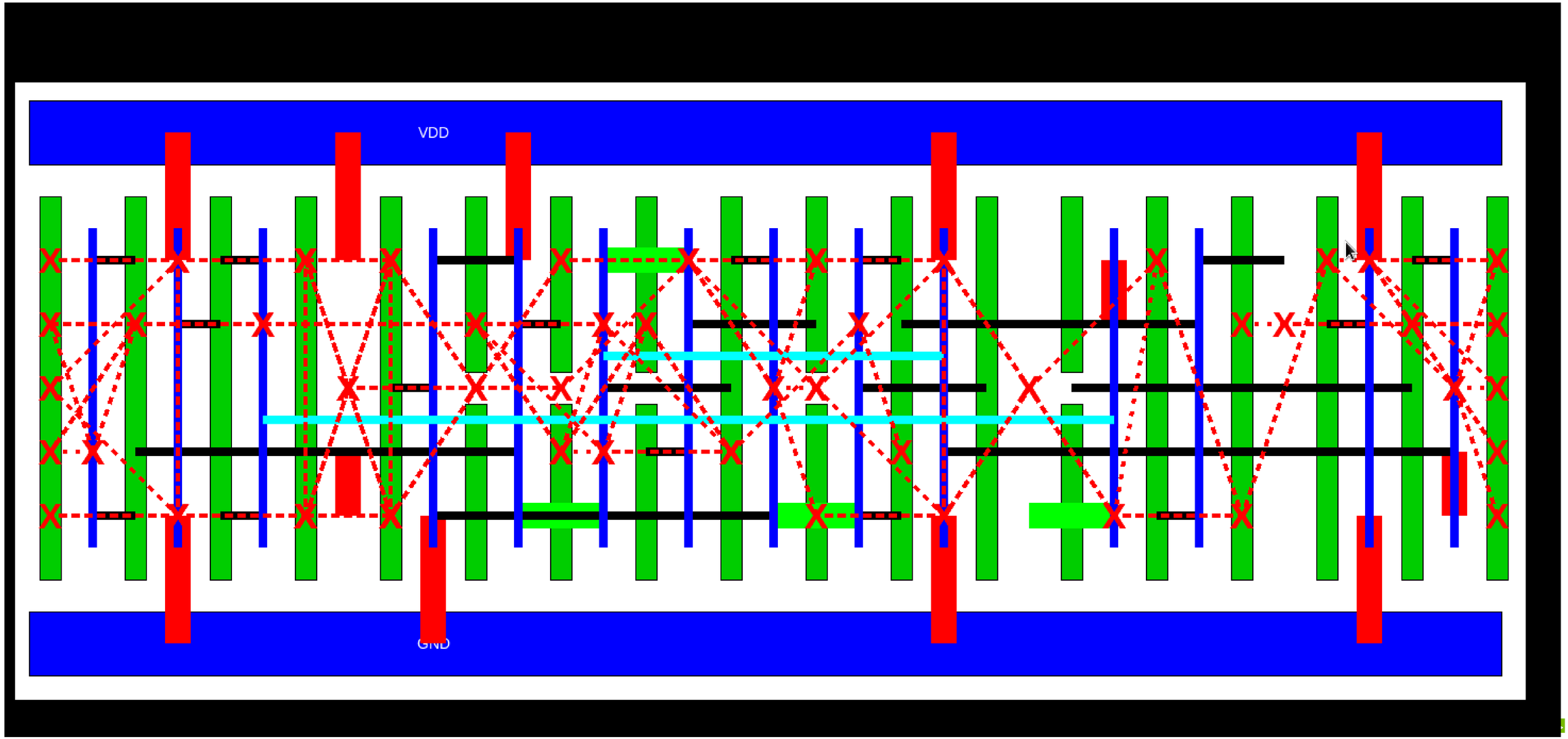
Step 1: DRC = 6
reward=0

Step 2: DRC = 3
reward=3

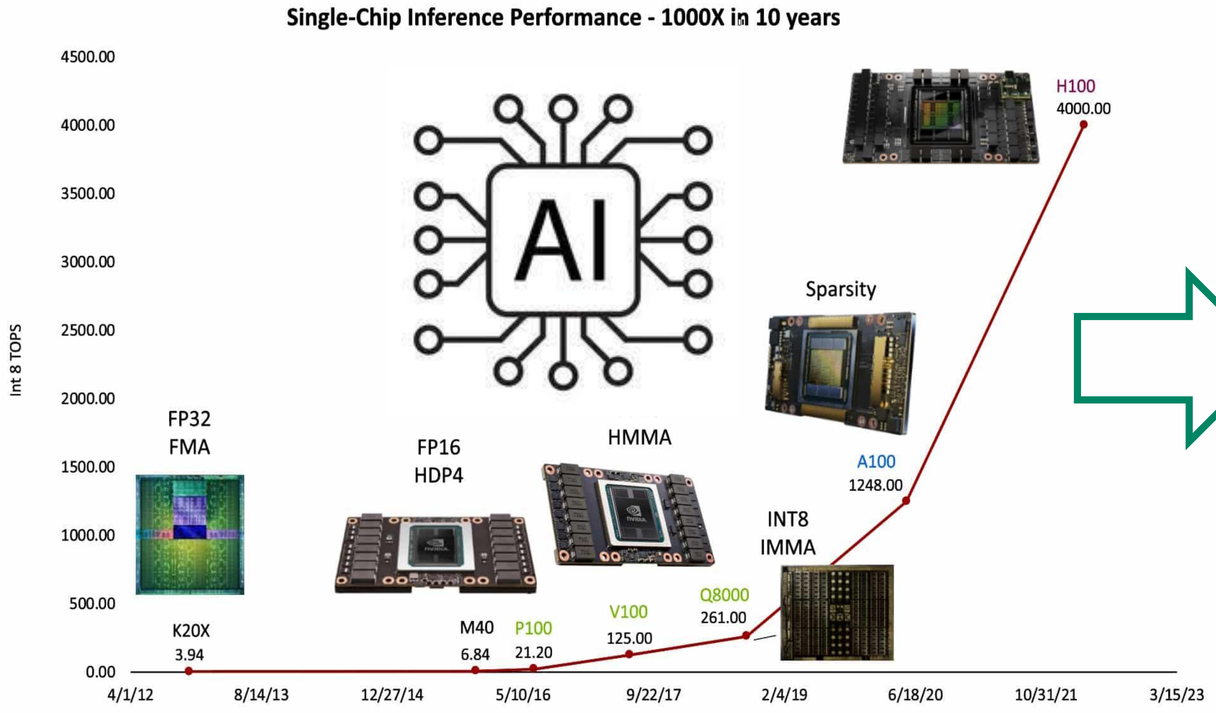
Step 3: DRC = 0
reward=3

H. Ren et al, NVCell: Standard Cell Layout in Advanced Technology Nodes with Reinforcement Learning, ASPDAC 2021

FIX DRC after Routing



AI For Circuit Design



Bill Dally, Hot Chips 2023

AI Prediction and Optimization for Circuit Design

LLM-based intelligence to improve design productivity

PPA & TTM

LLM Use Cases

"This evaluation set contains 1,800 prompts that cover 12 key use cases: asking for advice, brainstorming, classification, closed question answering, coding, creative writing, extraction, inhabiting a character/persona, open question answering, reasoning, rewriting, and summarization. "

-- *Introducing Meta Llama 3: The most capable openly available LLM to date*

LLM Applications for Chip Design

Sorted by Application Scopes

Coding Assistance → Generating design and verification code (software, RTL, testbenches, EDA scripts, tools scripts, and configs)

Know-how Assistance → Generating insights, knowledge, and ideas. Answer questions about designs, infrastructures, tools, flows, HW domains, etc.

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

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

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

Agent

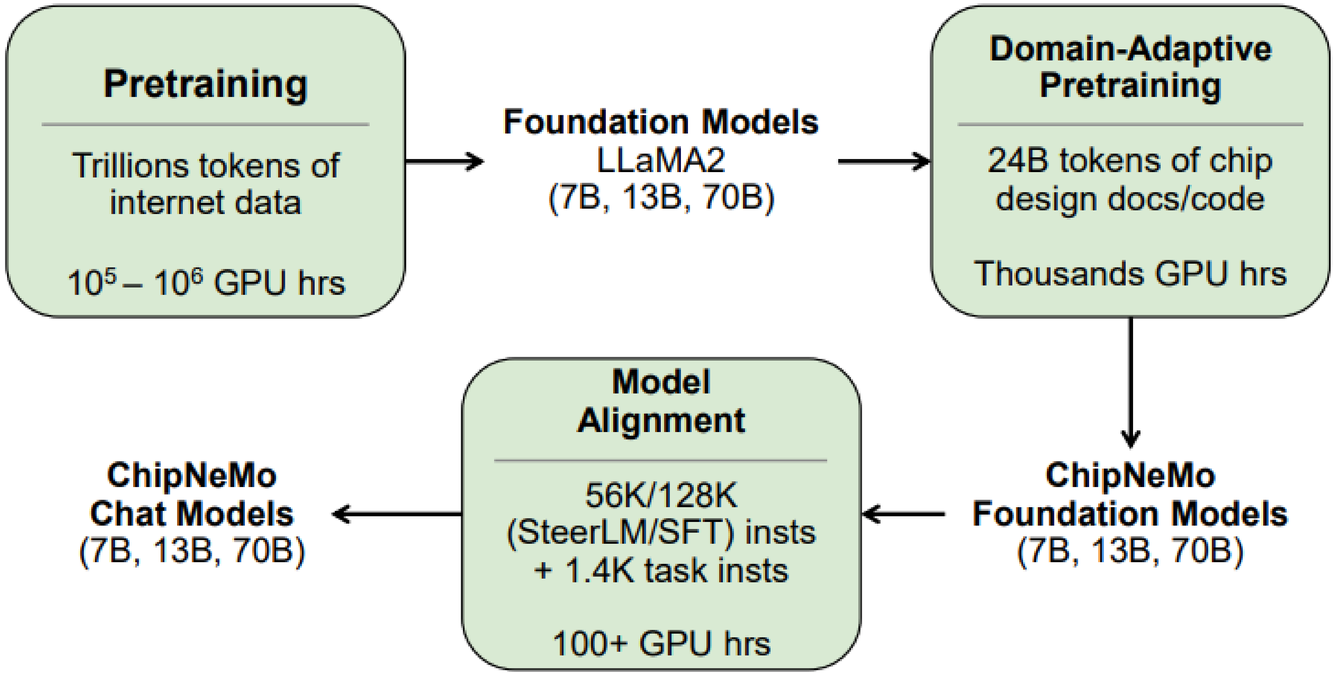
Domain-Adapted LLM

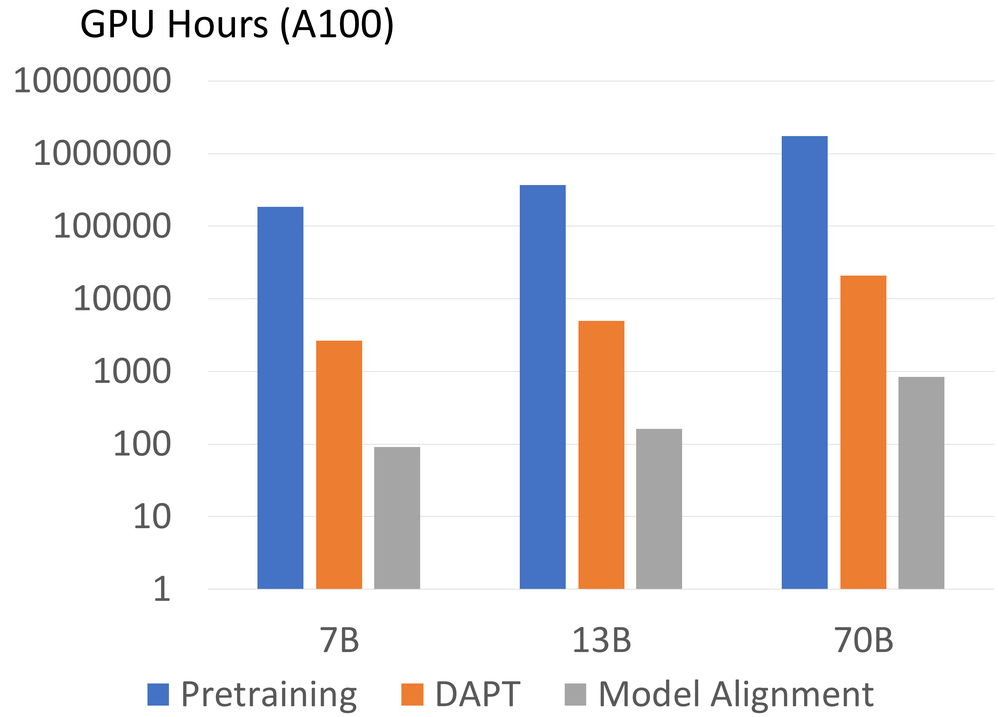
Challenges of general-purpose LLMs for Chip Design

- Lack of design knowledge
- Lack of specific coding language/tools knowledge
- Lack of design tasks-specific skills

Domain-adapted pretraining (DAPT) → learn coding/tools/design knowledge

DAPT and Model Alignment

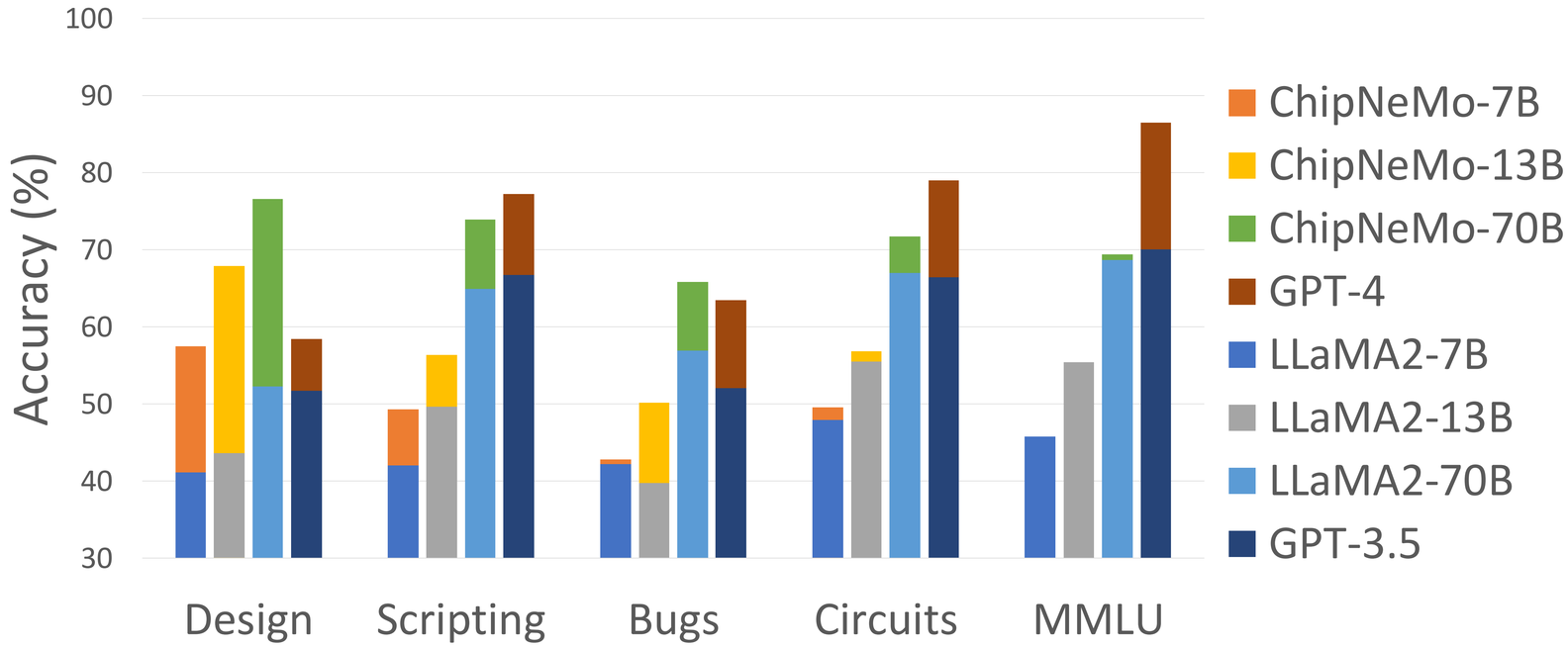




M. Liu et al, ChipNeMo: Domain-Adapted LLMs for Chip Design, Arxiv, 2023

Foundation Model Performance Comparison

Performance improves with base model size
ChipNeMo domain-adaptive pre-training provides significant performance improvements over the base model
Best ChipNeMo model has better performance than GPT-3.5 on all benchmarks and GPT-4 on design know-how and Bugs benchmarks



EDA Script Generation

Question:
write a code in *TOOL1* to print the number of flop cells in the give rectangle (0,0,100,100)

Answer:

```
# Sure, here is a TOOL1 code to print the number of
↪  flop cells in the given rectangle
↪  (0,0,100,100):

# This code gets the top_ref of the design
top_ref = env().get_top_reference()
# This code gets the cells of the rectangle in the
↪  top_ref
cells = env().layout().cells_in_rectangle( top_ref,
↪  rectangle(0, 0, 100, 100), hierarchical)
# This code gets the cells that are flop cells
flop_cells = cells.type("is_flop_cell")
# This code prints the number of flop cells
print(flop_cells.size())
```

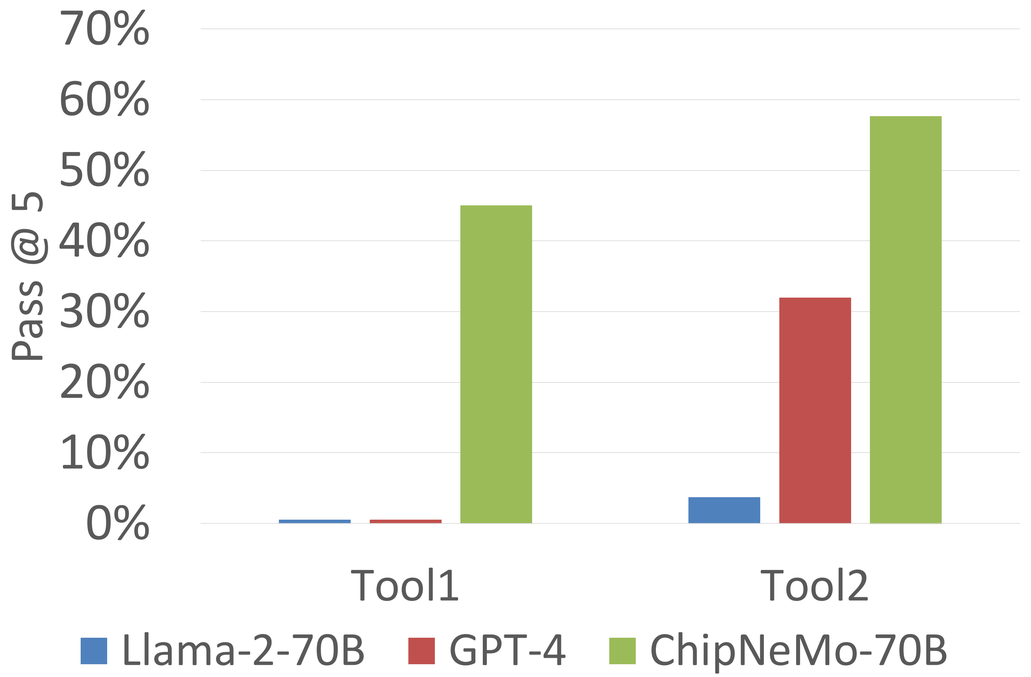
Question:
Write a *TOOL2* tcl proc to find intersecting logic from pins '$pinA' to pin '$pinB'.

Answer:

```
#Sure, here is the code:
proc myfunc {pinA pinB} {
    # Get all the pins in the fanout of the given pin
    set fanout_pins [fanout -from $pinA]
    # Get all the pins in the fanout of the given pin
    set fanin_pins [fanin -to $pinB]
    # Get the intersection of the two collections
    set intersection [common_logic $fanout_pins
    ↪  $fanin_pins]
    # Return the intersection
    return $intersection
}
```

Generate scripts for specific tasks (VLSI, Verification)→ **Coding Assistance**

Outperforms GPT-4 for both EDA tools
Custom tokenization & Model alignment data is critical
Long code snippet generation still hard



Write TOOL1 code to get partitions in design

ChipNeMo

```
# This code will get all partitions in the design

get_ref("*").child_refs(Opt.hier).filter("is_partition")
```

GPT-4

```
def partitions(set):
    if not set:
        yield []
        return
    for i in range(2 ** len(set) // 2):
        parts = [set[j] for j in range(len(set)) if (i >> j) & 1]
        yield parts

def count_partitions(set):
    count = 0
    for _ in partitions(set):
        count += 1
    return count

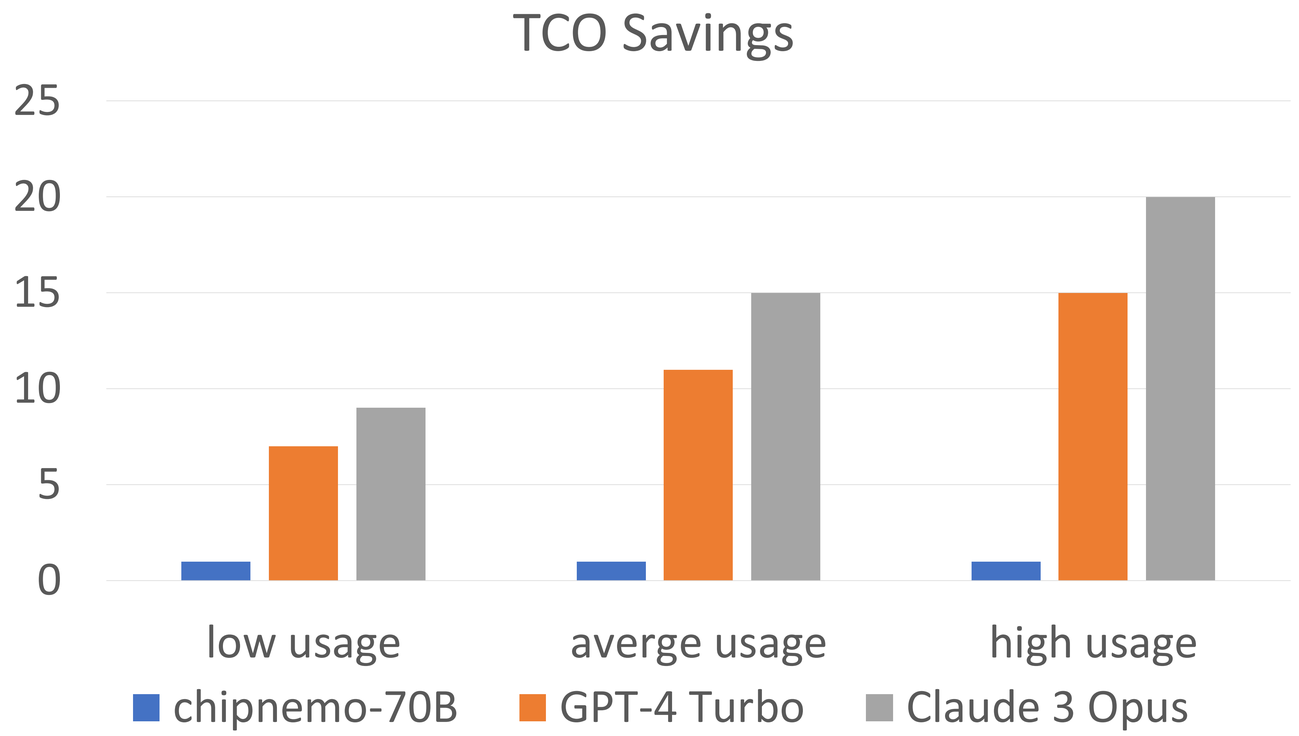
# Example usage:
my_set = {1, 2, 3}
print("Number of partitions:", count_partitions(my_set))
```

TCO (Total Cost of Ownership) Comparison

For EDA Script Generation Task

TCO includes training cost for code domain data + inference cost for regular usage

90% TCO saving by using domain adapted LLM compared to closed-sourced larger LLMs



A. Sharma et al , Assessing Economic Viability: A Comparative Analysis of Total Cost of Ownership for Domain-Adapted Large Language Models versus Open-Source Counterparts in Chip Design Coding Assistance, Arxiv, 2024

Retrieval Augmented Generation

RAG helps to ground LLM with curated documents

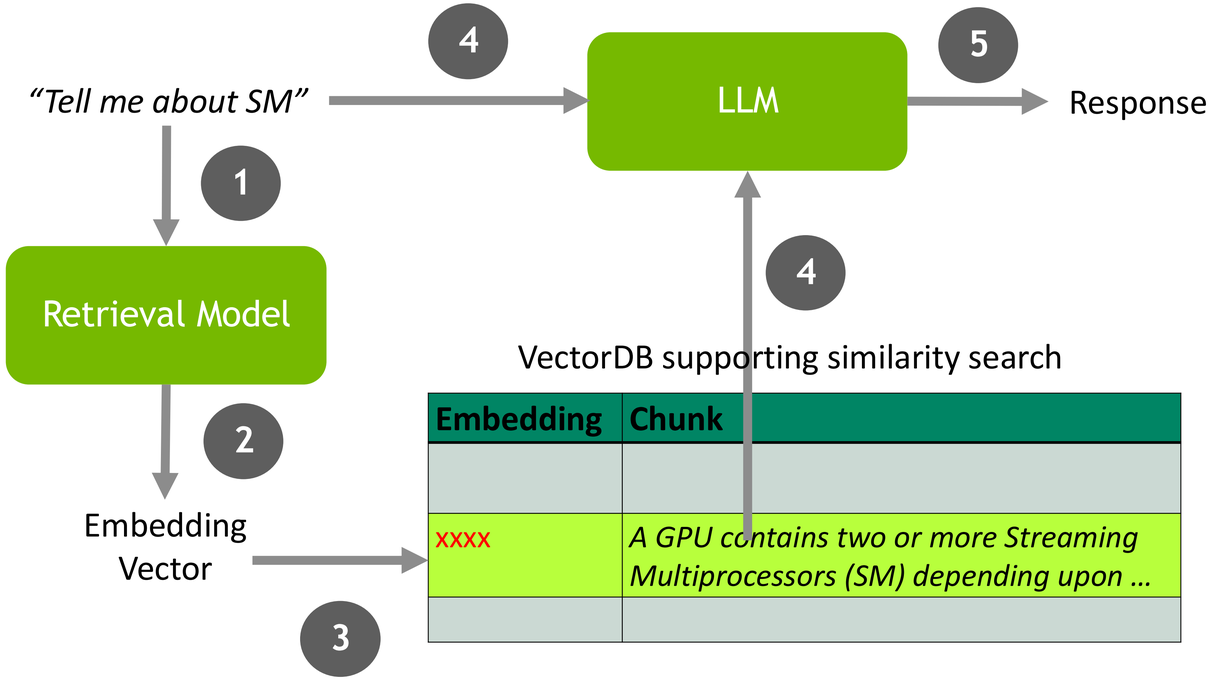
ChipNeMo model is very good at understanding our HW design, but it would hallucinate sometimes

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

RAG searches a given set of documents and retrieves the relevant chunks as context for the question

RAG performance depends on the retrieval model accuracy

- Fine-tuned retrieval model with domain data



RAG Illustration

Engineering Chat Bot

Question:
why do we add retiming stages?

Answer:
Retiming stages are added to improve the timing of the design and to account for the physical distance between different parts of the chip. By adding retiming stages, we can reduce the amount of time it takes for signals to travel between different parts of the chip, which can help to improve the overall performance of the design. Additionally, retiming stages can also help to reduce the amount of noise and interference that can occur on the signals, which can further improve the performance of the design.

Question:
How do I run multiple seeds of random tests using testgen framework?

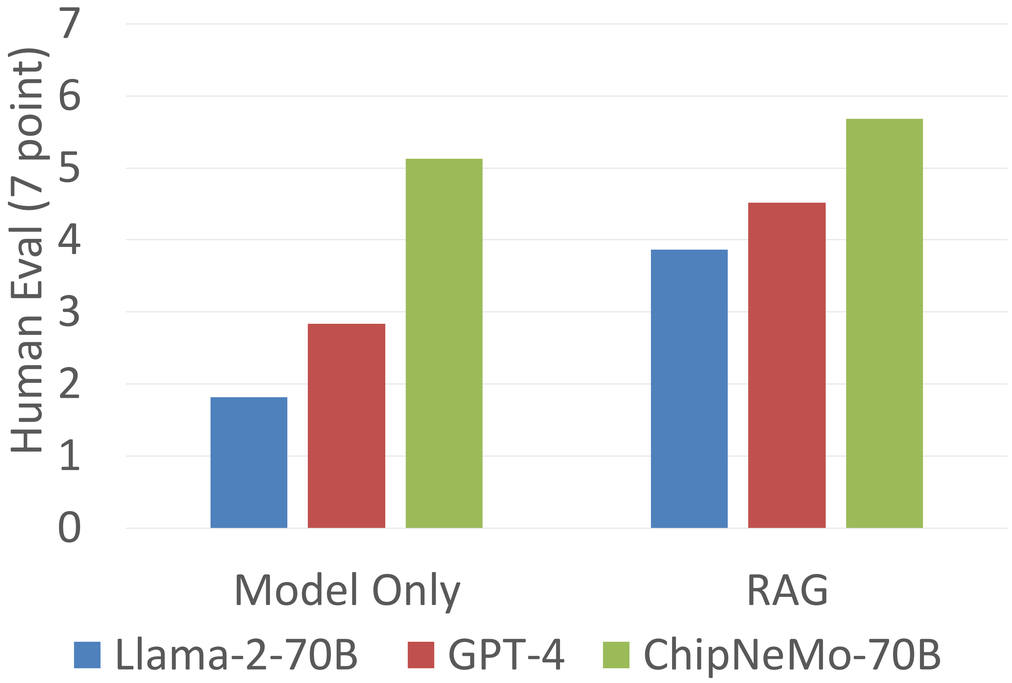
Answer:
You can run multiple seeds of random tests using the Testgen framework by using the *rand_tgen* target. This target allows you to specify a variety of options for the stopping criteria of random runs using the same Testgen testlist(s).

```
nvmk -c . rand_tgen TESTLIST=my_testlist MON=1
↪  TGEN.LOOP=50
```

This command will run 50 iterations of the *my_testlist* testlist, each with a different seed. You can specify the number of iterations and the seed mode using the TGEN.LOOP and SEED options, respectively.

Answer questions about designs, infrastructures, tools, flows, HW domains, etc. → **Know-how Assistance**

Outperforms GPT-4, even with Retrieval Augmented Generation (RAG)



What is CTL? Can you write an example CTL test?

ChipNeMo

CTL stands for "Compute Trace Library". It is a trace generation library used to create functional and performance tests for the GPU architecture. CTL provides an interface to write tests in C++ that can be compiled into a trace that runs on various simulation models, such as Fmodel, RTL, and Emulation.

Here is an example of a simple CTL test that allocates a buffer, initializes it with some values, and then reads back the values to verify that they are correct:

```
#include "ctl.h"

using namespace ctl;

int main(int argc, char* argv[]) {
// Set up the test
ctlAddTest("simple_buffer", "", "ga100");
…
```

GPT-4

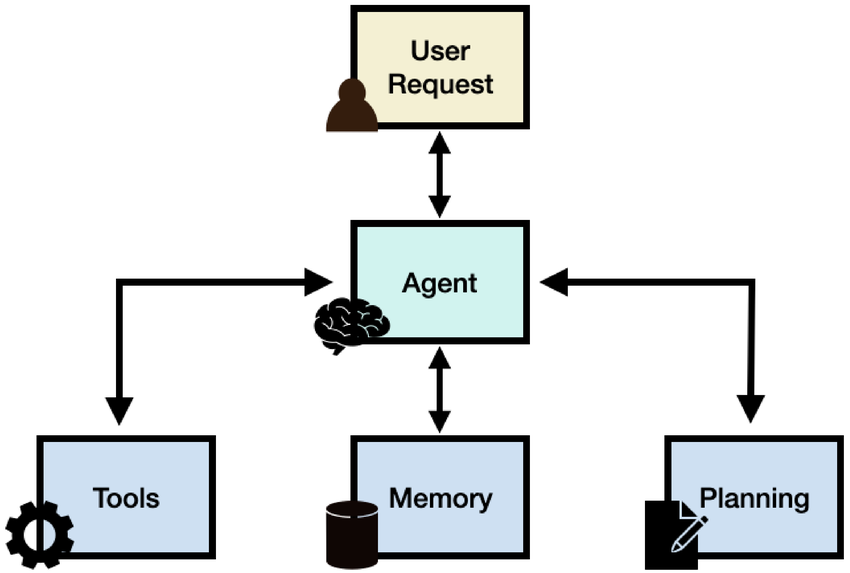
CTL, or Computation Tree Logic, is a form of modal logic used in computer science, particularly in the field of formal verification. It is a type of temporal logic, which means it allows reasoning about the behaviors of a system over time. CTL is especially useful in verifying properties of finite state systems like hardware circuits or software programs. It provides a way to express properties like "something will eventually happen" or "something is always true".

In CTL, we talk about states and paths. A state is a particular condition or situation in the execution of a system. A path is a sequence of states that represents a possible execution sequence through the system.

…

Agents

Reasoning, Planning and Use Tools



(1) Hotspot QA

Question: Aside from the Apple Remote, what other device can control the program Apple Remote was originally designed to interact with?

(1d) ReAct (Reason + Act)

Thought 1: I need to search Apple Remote and find the program it was originally designed to interact with.
Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control introduced in October 2005 by Apple … originally designed to control the Front Row media center program …

Thought 2: Apple Remote was originally designed to control the Front Row media center program. I need to search Front Row next and find what other device can control it.
Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports',' Front Row (software) ', …]

Thought 3: Front Row is not found. I need to search Front Row (software) .
Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software …

Thought 4: Front Row (software) is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.
Act 4: Finish[keyboard function keys]

S. Yao, REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS, ICLR, 2023

https://www.promptingguide.ai/research/llm-agents

Optimization Agent

Optimize std cell device cluster

Not enough data to learn a transform-based clustering solution for a new tech node.

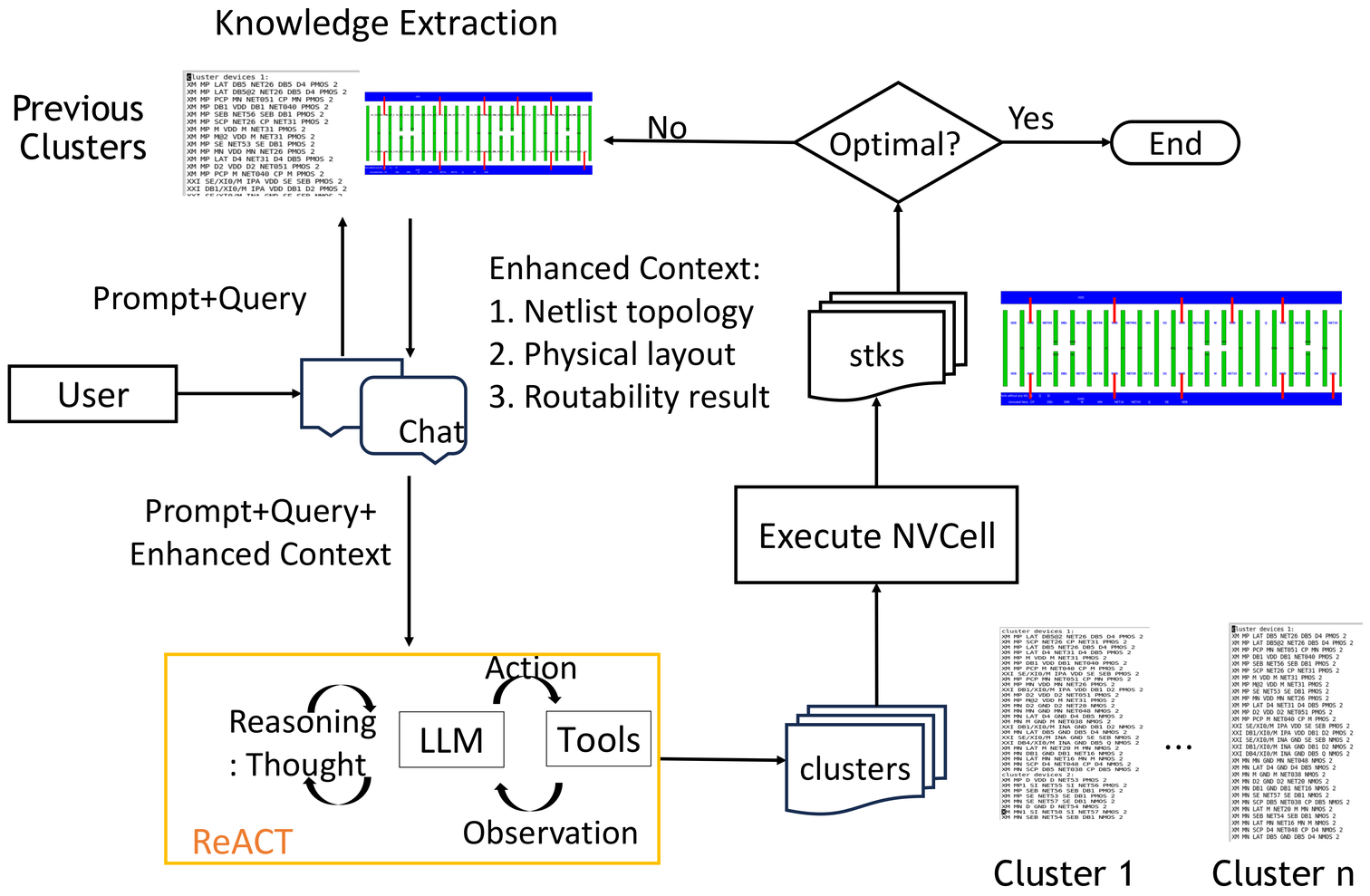
Use LLM to improve transformer-based clustering solution

Tools:

- Cluster evaluator
- Get group devices from nets
- Save cluster
- Get the best cluster

Achieved 5% area reduction

Better than SA



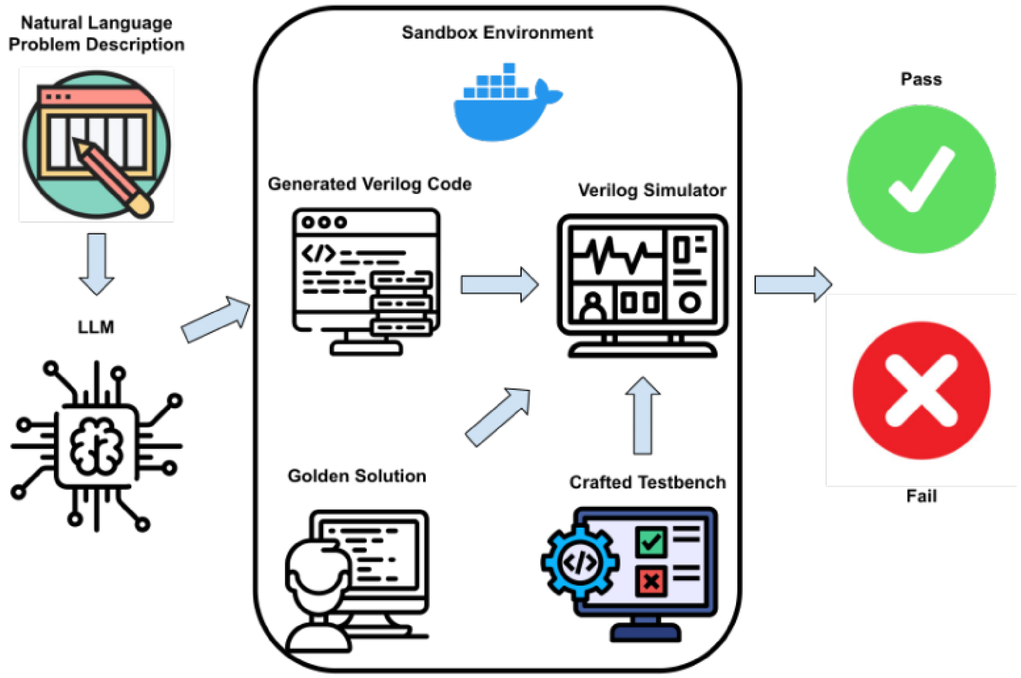
C.-T. Ho et al, Large Language Model (LLM) for Standard Cell Layout Design Optimization, to be published

HW Datasets, Benchmarks and Models

Call for community effort

VerilogEval

156 Verilog generation questions



M. Liu, VerilogEval: Evaluating Large Language Models for Verilog Code Generation, ICCAD 2023



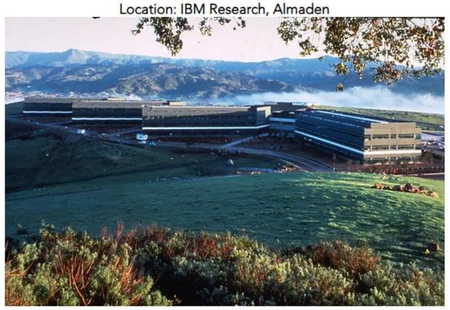
The First IEEE International Workshop on LLM-Aided Design (LAD'24)

June 28-29, 2024
IBM Research
Almaden, San Jose, CA

Home Call for Papers Committees Venue & Travel Program Registration Speaker Instruction

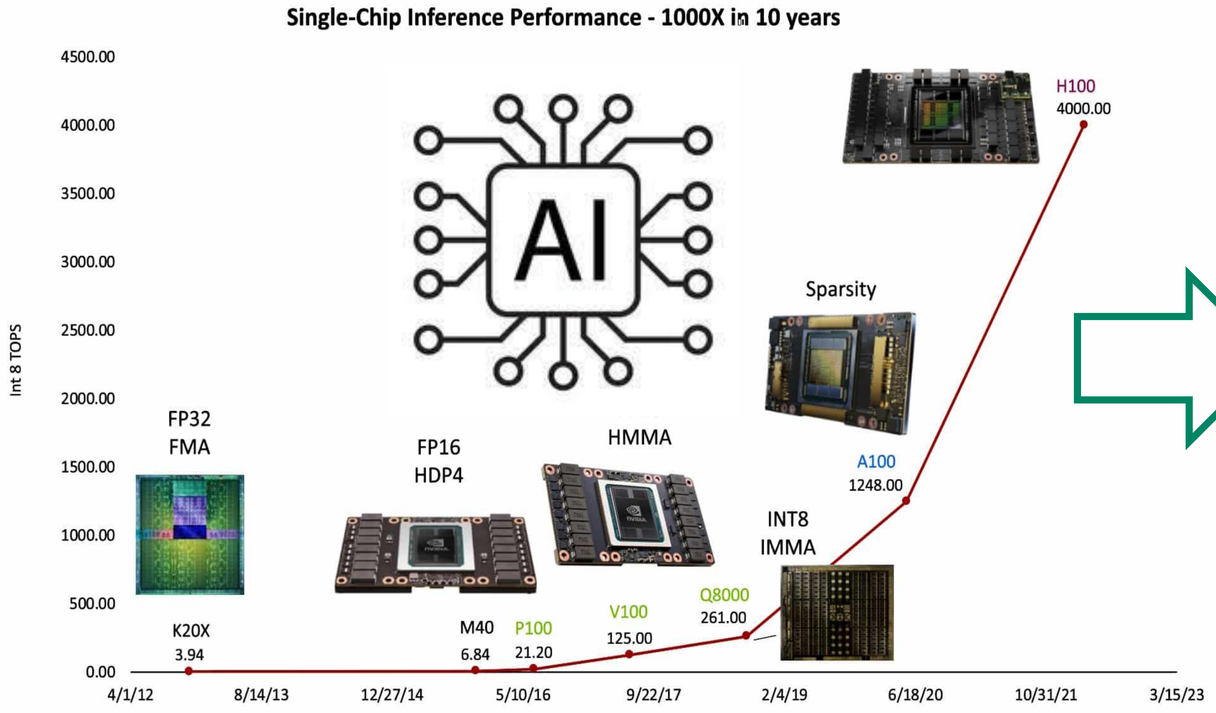
The First IEEE International Workshop on LLM-Aided Design, 2024 (LAD'24)

Time: June 28-29, 2024
Location: IBM Research, Almaden



This new workshop will focus on how to use LLM (large language model) as a methodology to help design circuits, software, and various computing systems with improved quality, productivity, robustness, and cost. The workshop will be a timely offer leveraging the significant advancement and innovation captured in the LLM technology to offer new methods and solutions for design automation targeting various applications. The workshop will provide a forum for people to present their latest research results, contribute open-source LLM models, datasets, tool flows, and offer benchmarking, testing and validation methods and solutions.

AI For Circuit Design



Bill Dally, Hot Chips 2023

AI Prediction and Optimization for Circuit Design

LLM-based intelligence to improve design productivity

PPA & TTM



Acknowledgments

Abhishek Akkur, Hao Chen, Mark Ho,
Mingjie Liu, Amit Sharma, Yun-Da Tsai,
Keren Zhu, ChipNemo Team