UNIFIED LLM MODEL FOR POWER, PERFORMANCE, AND AREA PREDICTION FROM HARDWARE CODE

Armin Abdollahi, Mehdi Kamal, Massoud Pedram

{arminabd, mehdi.kamal, pedram}@usc.edu

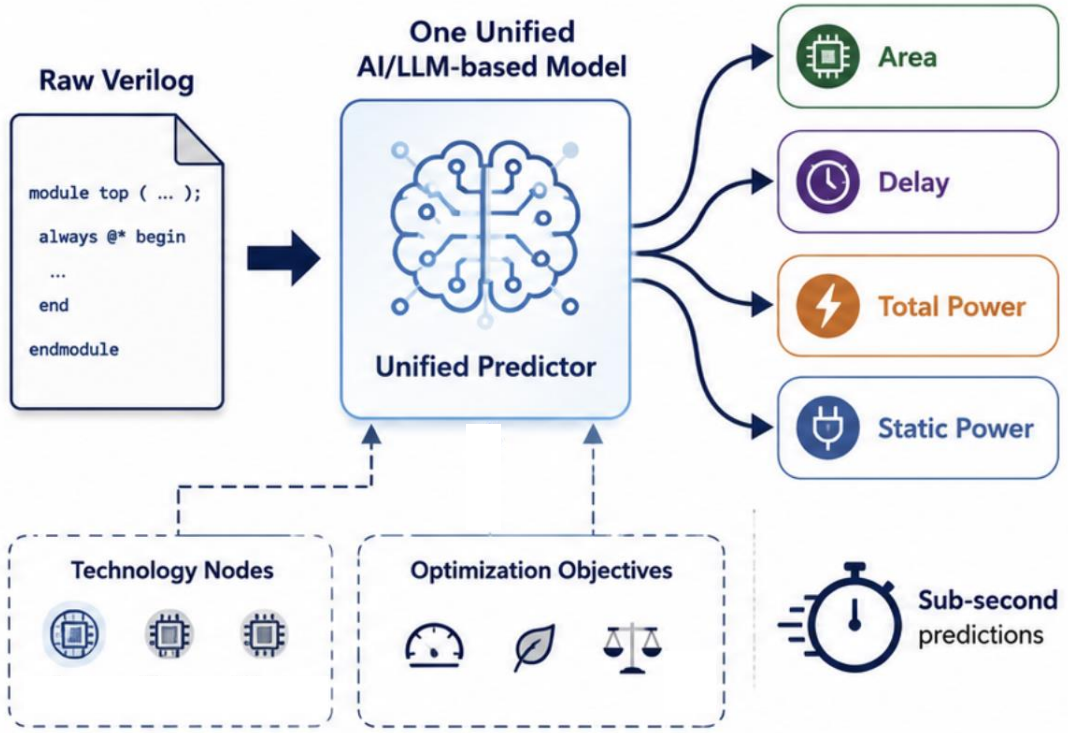
University of Southern California
Los Angeles, California, USA

Wednesday, May 20, 2026

Ninth Annual Conference on Machine Learning and Systems

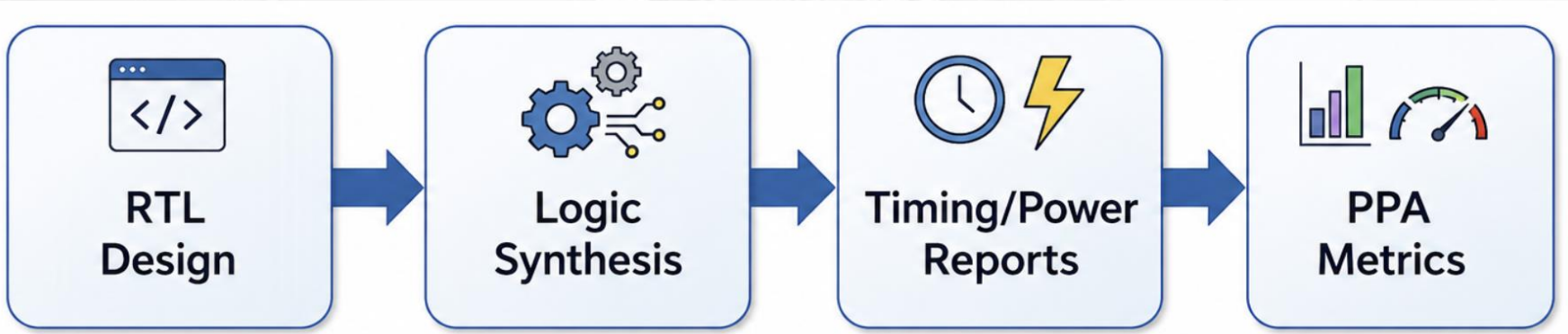
Ultimate Goal

1. **Predict PPA directly from Verilog**
   without running costly synthesis flows.
2. **Use one unified model**
   for area, delay, total power, and static power.
3. **Generalize across conditions**
   such as technology nodes and optimization objectives.
4. **Enable fast design-space exploration**
   with sub-second prediction for early-stage hardware design.



One unified predictor for fast, condition-aware RTL-stage PPA estimation.

Why PPA Estimation Is a Bottleneck



Accurate, but slow and expensive

- RTL code alone **does not** reveal final area, delay, or power.
- Accurate PPA requires **synthesis** under a target technology library.
- Each design variant may need a **separate tool run**.
- This **slows down** early design-space exploration.

PPA is Not a Property of RTL Alone

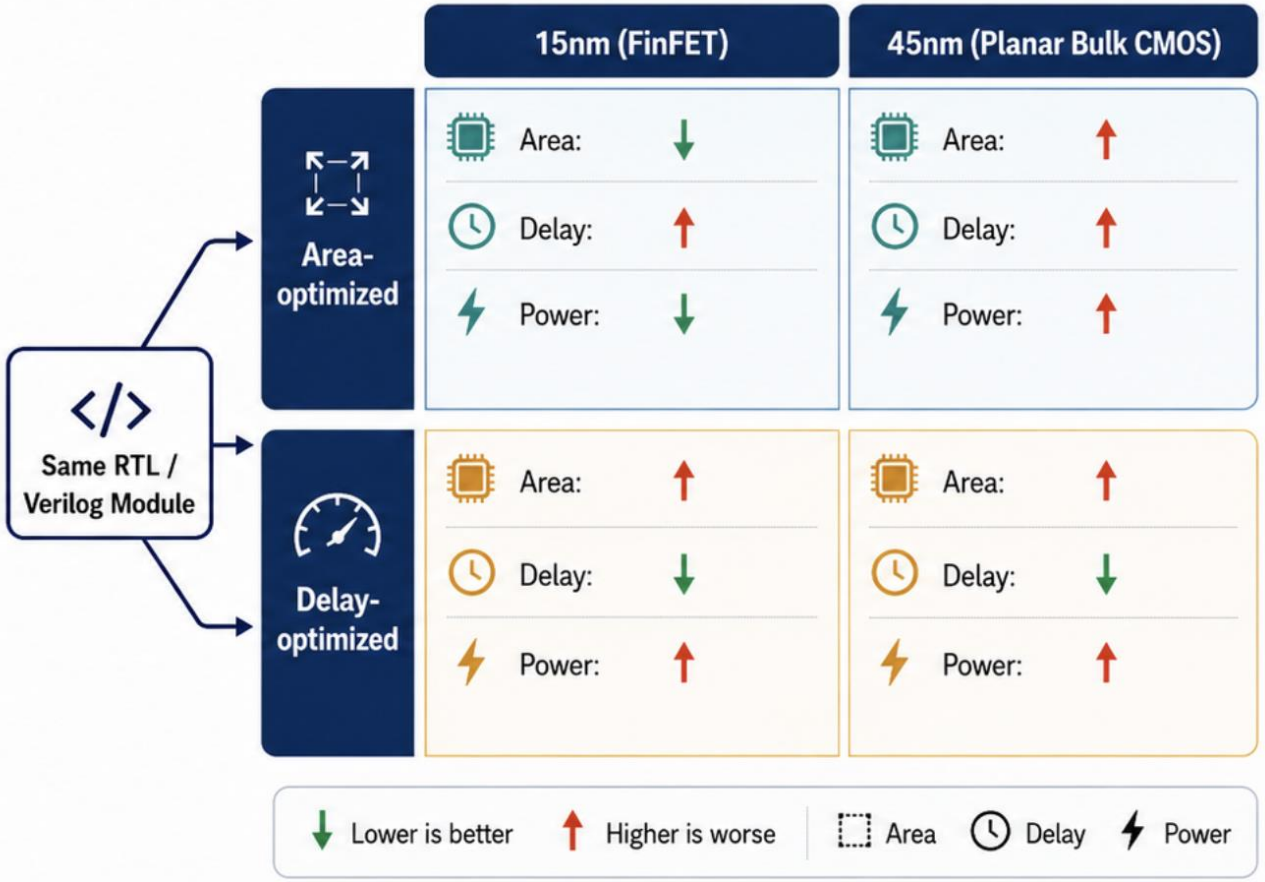
Same Verilog module → different PPA under different conditions

Two axes of variation

- **Technology mode:** 15nm (FinFET) vs. 45nm (planar bulk CMOS) — different leakage, drive strength, parasitics
- **Optimization objective:** area-minimization vs. delay (timing closure)

This creates 4 distinct synthesis regimes for every design

Prior work: most estimators target a single fixed regime; none predict all three metrics simultaneously across multiple nodes



Where prior art falls short

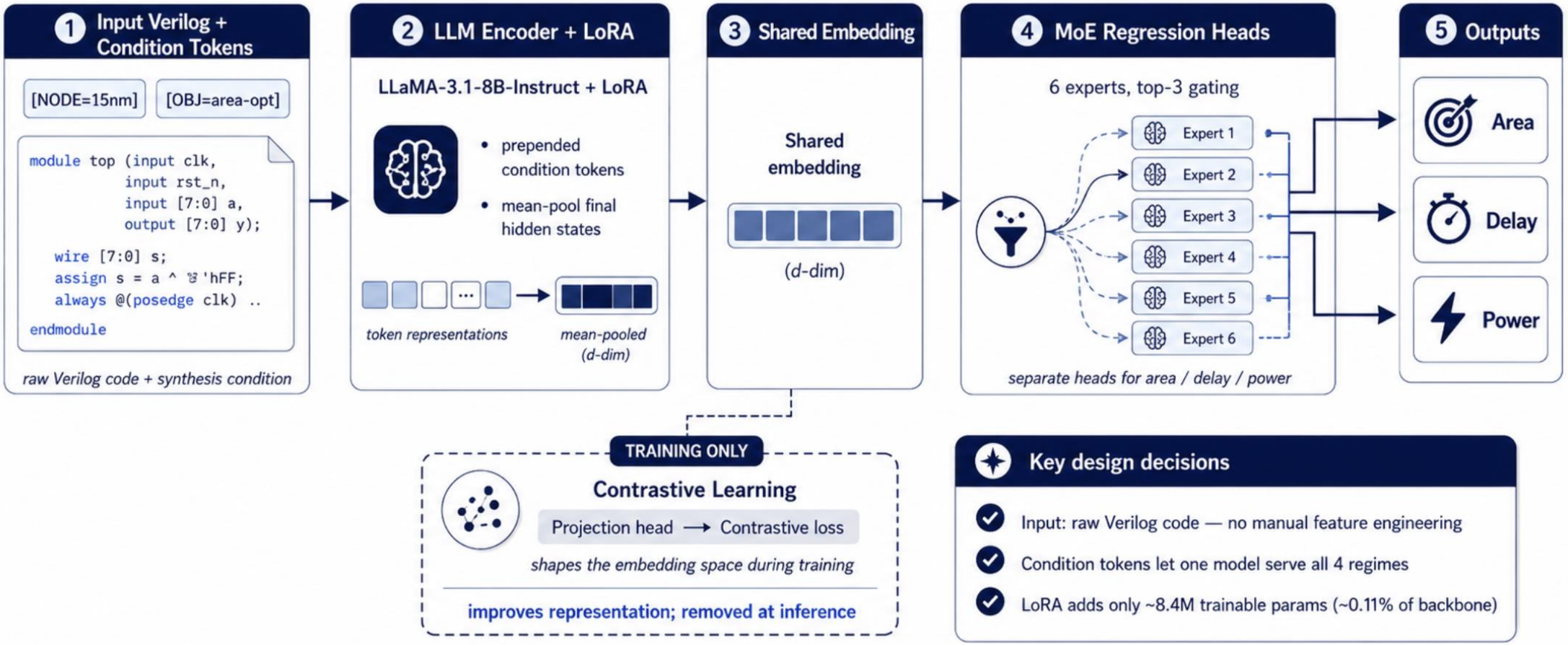
Method	Approach	What it covers	Limitation for our goal
MetRex	LLM-based Verilog metric reasoning	Area, delay, static power; large Verilog benchmark	Not condition-aware across our target 15nm/45nm regimes and objectives; focuses on numeric estimation accuracy
MasterRTL	RTL → SOG representation + separate ML models	Timing, power, area at RTL stage	Customized preprocessing and per-metric models; not a unified multi-node, multi-objective predictor
CircuitFusion	Code + graph + retrieval representation	Multi-modal RTL/PPA modeling	Rich pipeline but slower/heavier; not designed for sub-second full-suite inference across regimes

Prior art emphasizes value-regression error, not decision-oriented pass rate

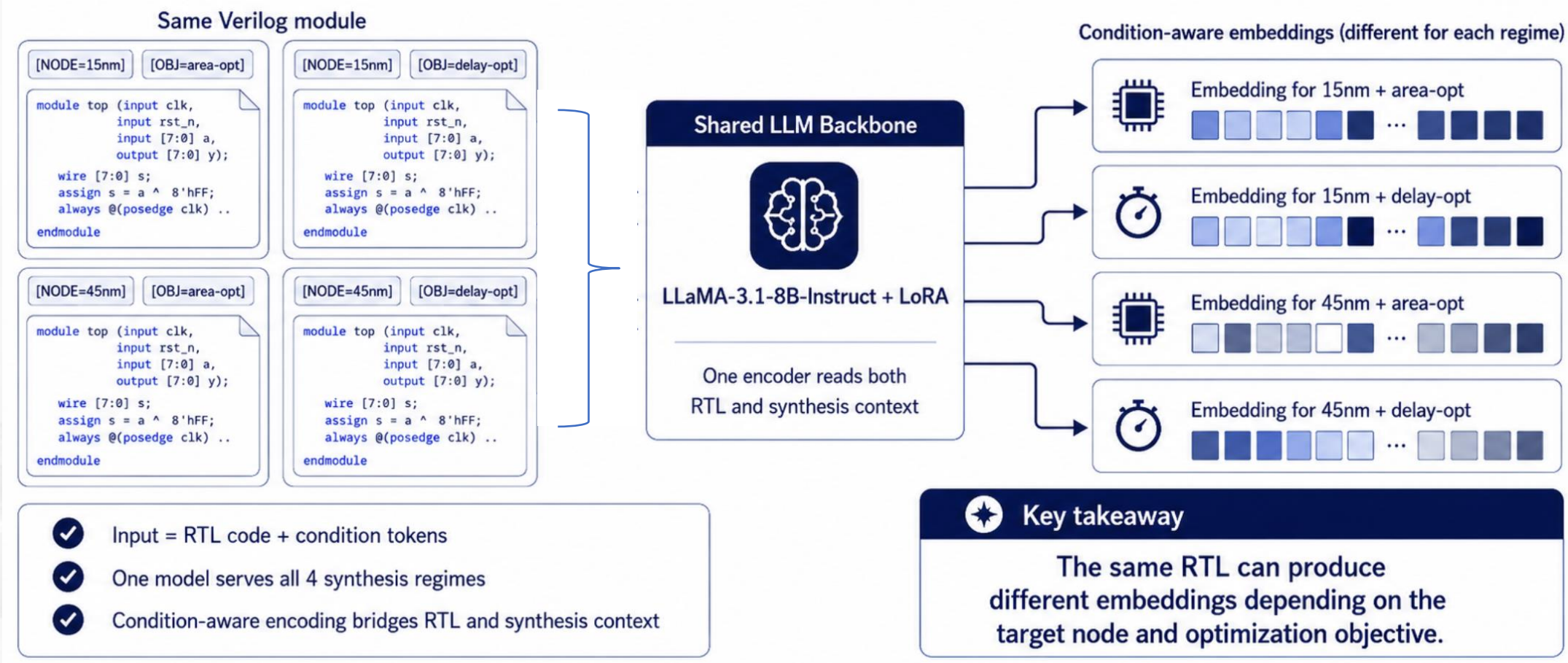
Existing flows are not optimized for sub-second inference at full test-suite scale

Key gap: no unified, condition-aware, fast model across nodes and objectives

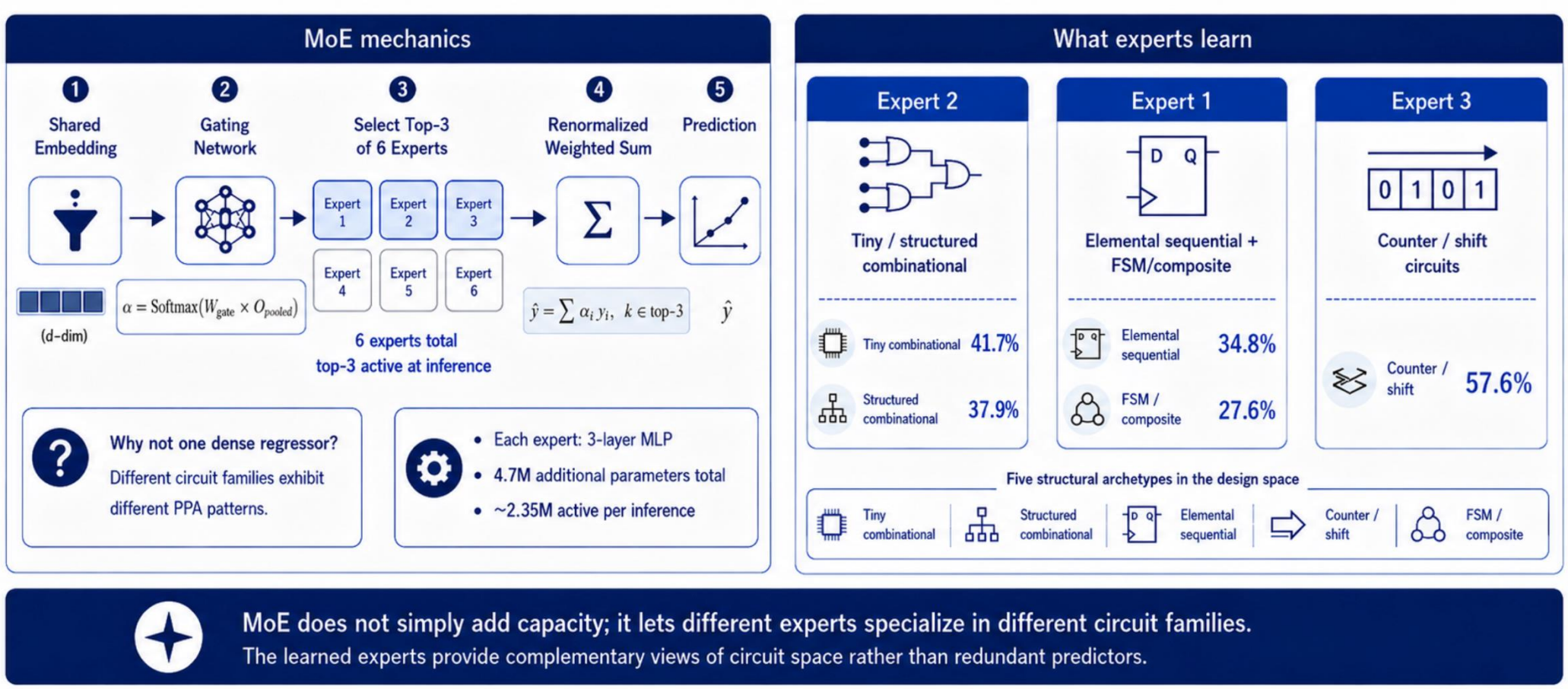
RocketPPA: One model, four conditions



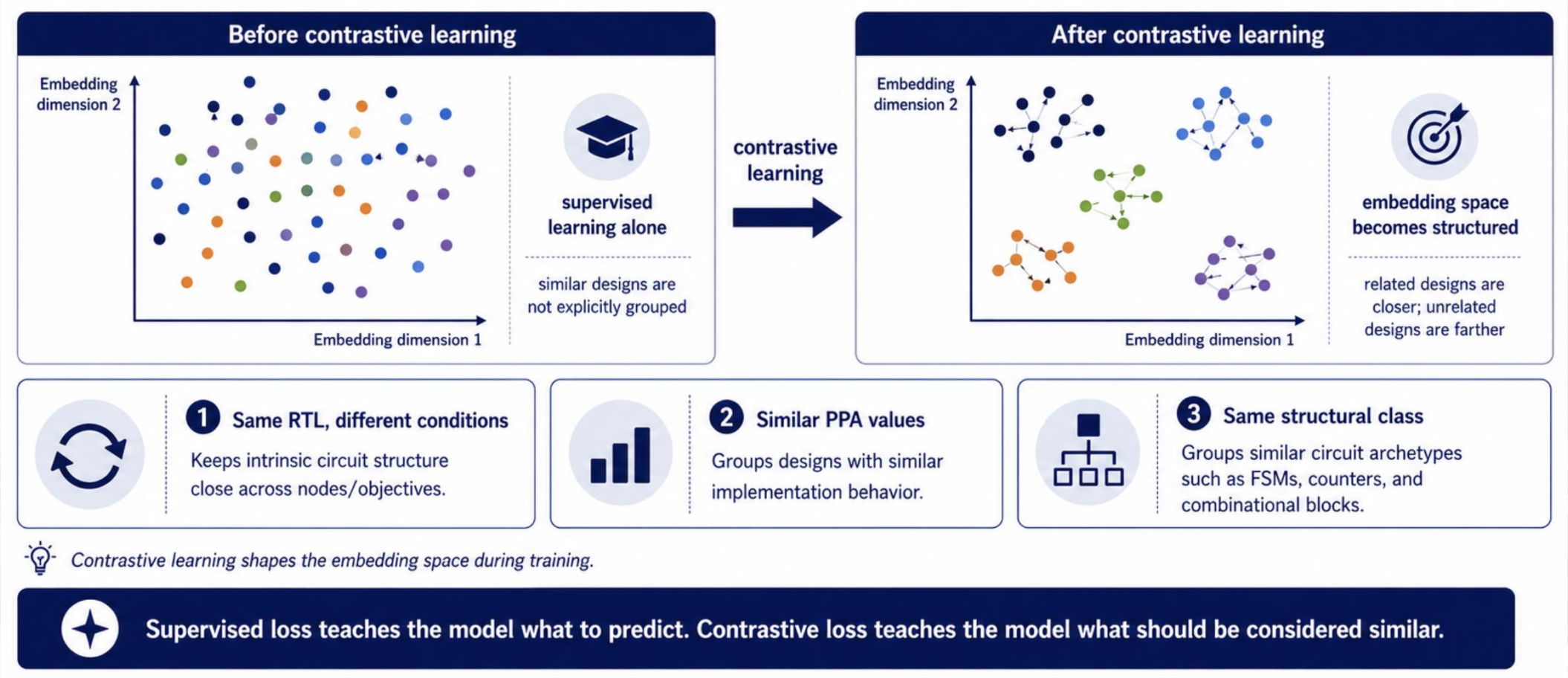
Encoding Both RTL and Synthesis Context



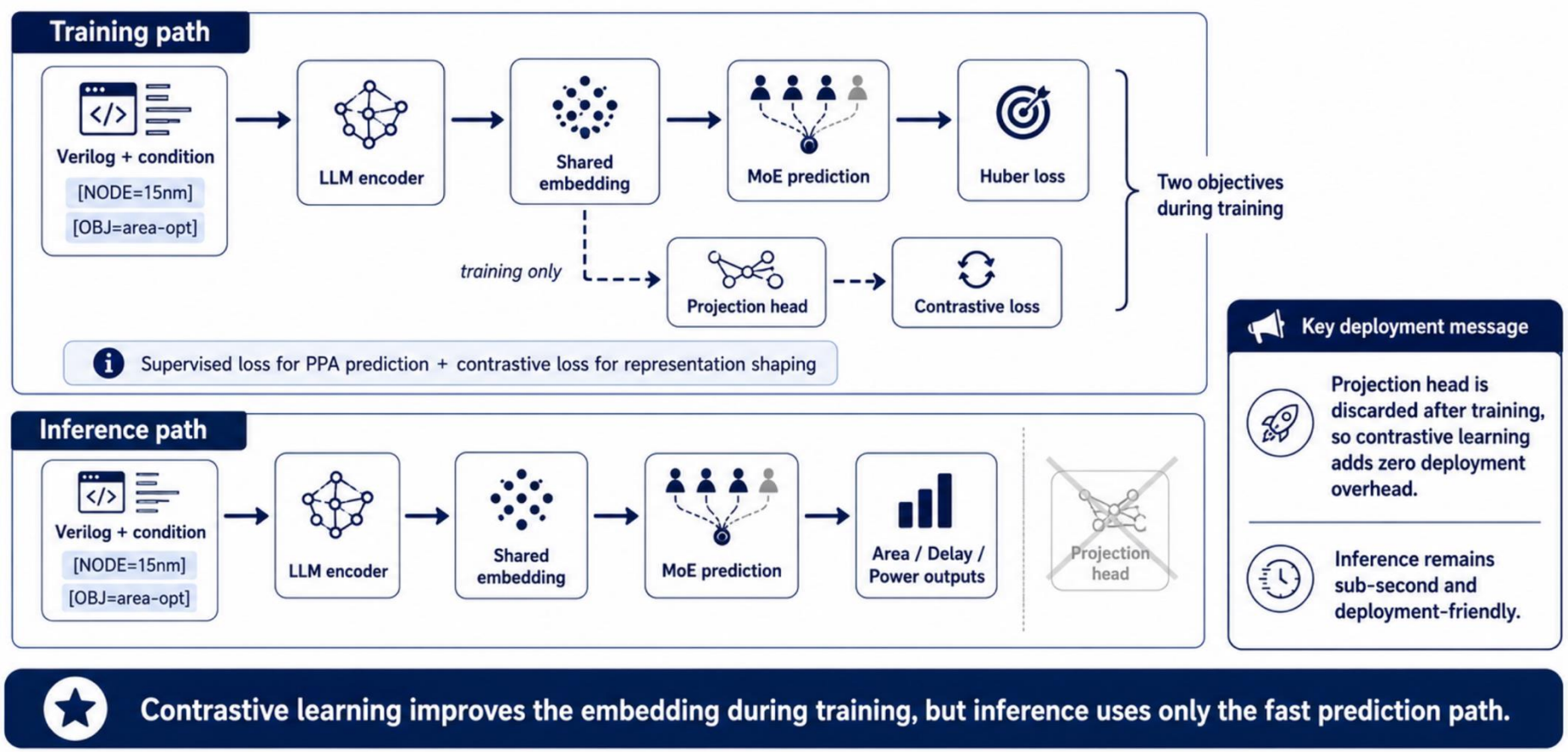
Mixture of Experts: Specialized Regression Heads



Contrastive Learning: Organizing the Design Space



Contrastive Learning Helps Training, Not Inference Cost



20K+ Synthesizable Modules via LLM-Driven Repair

1) Dataset Composition (Table 1)

Source	Number of Modules
MG-Verilog	9,239
verilog_github	7,791
VeriGen	3,923
Total	**20,953**

Total: 20,953 clean, synthesizable modules covering diverse circuit families and scales.

2) The Challenge & Our Solution

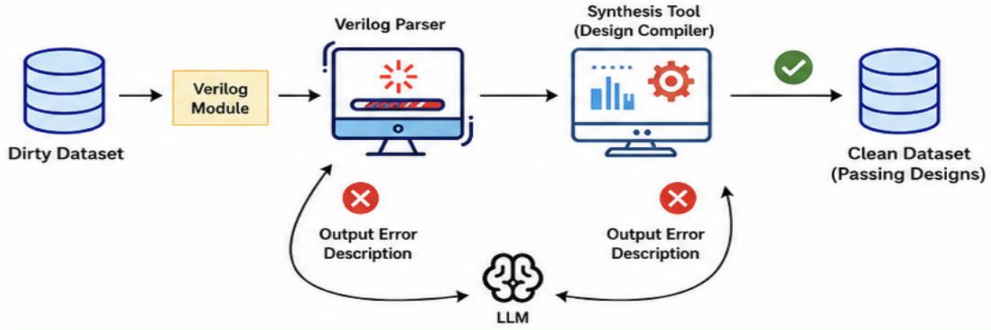
The Challenge:

Public Verilog corpora are not synthesis-ready:

- Missing includes, vendor pragmas
- Simulator-only constructs

Our Solution:

Automated LLM-driven repair pipeline:
iterative syntax + synthesis validation →
LLM generates targeted fixes → only passing designs retained



Ground-Truth Labels:

Synopsys Design Compiler across all 4 conditions
→ up to 4 labeled instances per module

15nm / area-opt | 15nm / delay-opt | 45nm / area-opt | 45nm / delay-opt

80/20 Train/Val Split at Design Identity Level — No Cross-Split Leakage

All instances (up to 4 conditions) of the same module stay in the same split. | Ensures fair generalization to unseen designs.

Experimental Setup

Metric

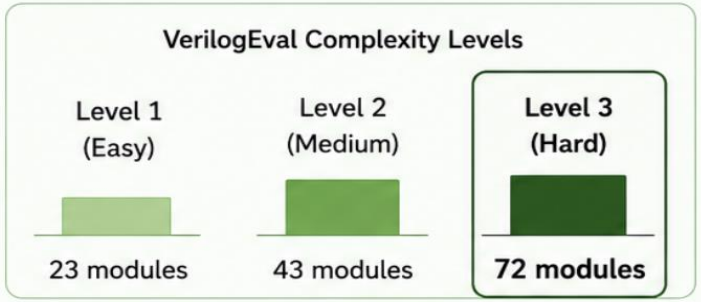
Pass Rate @ θ% threshold

- A prediction is "correct" if relative error ≤ θ
- Reported at $\theta \in \{5\%, 10\%, 20\%\}$ for all three metrics
- Decision-oriented — directly reflects design-time utility

Test Benchmark

VerilogEval (138 HDL codes)

- Focus on Level 3 (72 most complex modules): FSMs, complex combinational, advanced sequential
- Up to 607 gates



Baselines

Baselines retrained end-to-end on same data splits:

- MetRex
- MasterRTL
- CircuitFusion

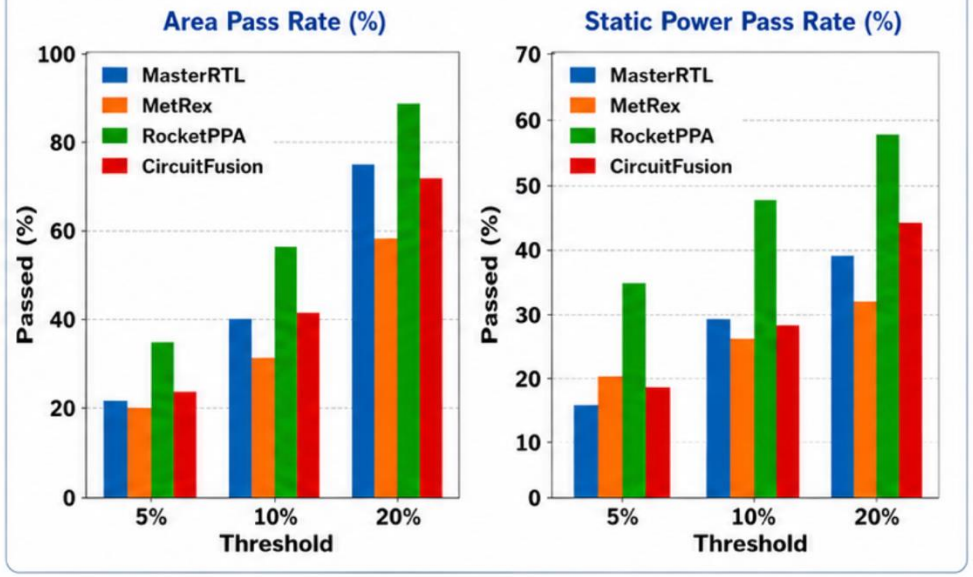
Results: RocketPPA outperforms prior methods

Average Pass Rate on VerilogEval

Method	10% Threshold				20% Threshold			
	Area	Delay	Total Pwr	Static Pwr	Area	Delay	Total Pwr	Static Pwr
MetRex	58.0%	47.8%	N/A	42.0%	75.0%	64.1%	N/A	54.3%
RocketPPA	**71.6%**	**57.2%**	**55.0%**	**56.7%**	**84.6%**	**74.9%**	**70.8%**	**72.8%**
MasterRTL	51.2%	50.7%	45.1%	47.5%	78.1%	67.9%	63.7%	65.2%
CircuitFusion	53.5%	50.5%	46.2%	48.1%	77.2%	67.5%	63.8%	66.1%

N/A: Total Power metric is not available for MetRex.

Level-3 VerilogEval (72 hardest modules)



Best average pass rate across Area, Delay, Total Power, and Static Power.

At 10% threshold, RocketPPA improves over MetRex by **+13.6 pts** (Area), **+9.4 pts** (Delay), and **+14.7 pts** (Static Power).

Advantages persist on Level-3, the most challenging VerilogEval subset.

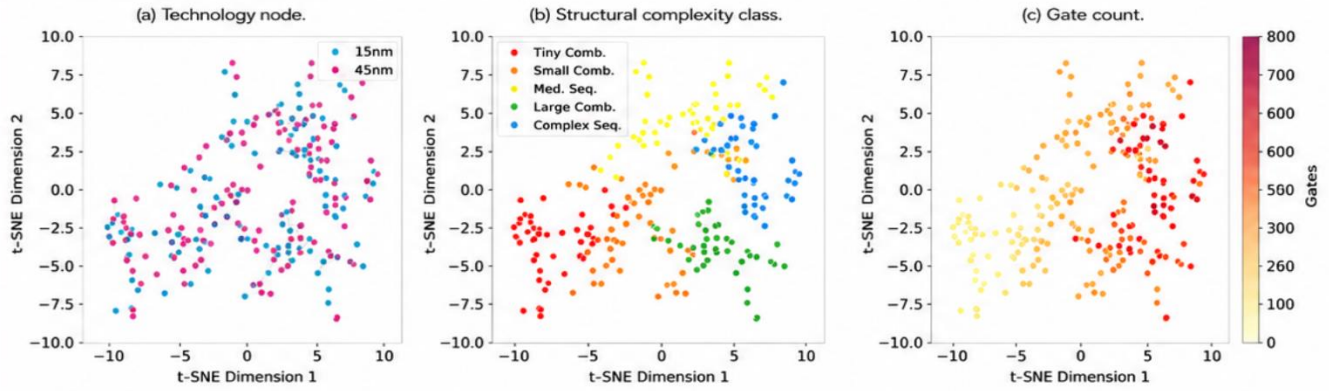
What drives the gains

MoE vs. Dense MLP

- LLaMA-3.1-8B-Instruct + MoE vs. matched dense MLP: **+5.1pp area, +4.1pp delay, +4.3pp power at 10%**
- Consistent across all three backbone variants (CodeLlama-7B, LLaMA3-8B, LLaMA-3.1-8B-Instruct)

Base Model & Method	10% Threshold			20% Threshold		
	Area	Delay	Power	Area	Delay	Power
CodeLlama-7B (MLP)	61.8%	47.7%	45.1%	75.8%	65.6%	57.1%
CodeLlama-7B (MoE)	**65.9%**	**51.7%**	**49.2%**	**79.0%**	**69.2%**	**60.2%**
LLamA3-8B (MLP)	60.2%	48.1%	47.2%	73.2%	70.8%	56.9%
LLamA3-8B (MoE)	**64.3%**	**52.1%**	**49.4%**	**76.7%**	**71.8%**	**58.1%**
LLama-3.1-8B-Instruct (MLP)	66.5%	53.1%	50.7%	80.9%	71.1%	63.0%
LLama-3.1-8B-Instruct (MoE)	**71.6%**	**57.2%**	**55.0%**	**84.6%**	**74.9%**	**70.8%**

Embedding Structure



- Same-design instances cluster across synthesis settings.
- Complexity classes separate naturally.
- **Gate count varies** smoothly in the embedding space.

t-SNE of learned embeddings for 500 validation designs. Colors denote (a) technology node, (b) structural class, and (c) gate count. With contrastive learning, same-design instances cluster across synthesis settings (short connecting lines), classes separate, and area varies smoothly within clusters—evidence of a similarity-aware representation.

Robust generalization to unseen settings

Leave-One-Regime-Out (LORO)

- Train on 3 of 4 synthetis conditions, test on held-out
- Maximum degradation: **~2.5pp** across all metrics and regimes
- Model generalizes well to unseen node-objective combinations

Held-out regime	Area		Delay		Static Pwr	
	Actual	LORO	Actual	LORO	Actual	LORO
15 nm Area-opt	65.8%	64.2%	51.7%	50.3%	54.9%	53.1%
15 nm Delay-opt	66.5%	64.4%	61.9%	59.5%	60.1%	58.0%
45 nm Area-opt	75.8%	74.1%	58.0%	56.6%	53.4%	52.1%
45 nm Delay-opt	78.4%	76.3%	57.1%	55.6%	58.4%	56.3%

Cross-node Few-shot Adaptation

With only 5% 15nm calibration data:
4.2pp degradation

With 10% calibration data:
2.7pp degradation

Practical takeaway:
Organizations can adapt RocketPPA with a small in-house sample.

RocketPPA: fast, accurate, and deployment-ready

What we built:

- Unified condition-aware LLM estimator: raw Verilog → simultaneous power, area, delay predictions across 4 regimes
- Novel contrastive framework with 3 complementary positive-pair strategies
- LLM + LoRA + MoE architecture: parameter-efficient, single-GPU deployable

Key results:

- **+13.6pp** area, **+9.4pp** delay, **+14.7pp** static power at 10% tolerance
- **>20×** throughput over prior LLM-based methods, **>30×** over graph-feature methods
- **0.12 seconds** per design on a single A6000 GPU

Broader impact:

- Enables rapid design-space exploration at the RTL stage — before any synthesis is run
- Hierarchical designs (10–20 submodules) fully profiled in 1–2 seconds
- Few-shot adaptation makes it practical for industrial codebases

Future work teaser: Extending to more technology nodes, dynamic power decomposition, and integration into EDA tool flows

THANK YOU

Questions 

For additional questions, please reach out to

Armin Abdollahi

arminabd@usc.edu