MapTuner: Multimodal Recipe Recommendation for Physical Design with Layout Maps and Design Insights

Hao-Hsiang Hsiao*, Sudipto Kundu†, Wei-Ting Jonas Chan†, Chitralekha Dasgupta†, Sung Kyu Lim‡
*Georgia Institute of Technology †Synopsys Inc. ‡University of Southern California
thsiao@gatech.edu, {Sudipto.Kundu, Wei-Ting.Chan, Lekha.Dasgupta}@synopsys.com, limsung@usc.edu

***Abstract*—State-of-the-art physical-design autotuning relies on black-box optimization loops that lack interpretability and design context. We propose MapTuner, a multimodal vision–language framework that analyzes layout maps and tool-generated diagnostics to recommend flow recipes for subsequent iterations, reducing manual effort and GUI debugging. We further apply GRPO-based post-training to align the model's recipe policy with PPA objectives under different timing–power intents. An intent-guided adapter maps user intent to objective-specific LoRA adapters for parameter-efficient multi-objective adaptation. Evaluations on 13 industrial benchmarks show consistent PPA improvements and robust generalization across diverse designs and technology nodes.**

I. Introduction

Commercial physical design (PD) tools expose hundreds of tunable parameters across placement, routing, and optimization stages that determine timing, power, and congestion. Evaluating one configuration requires a full place-and-route flow, taking hours to days for industrial designs. This runtime barrier makes systematic exploration impractical, even with GPU-accelerated engines [1], [2]. To reduce evaluations, recent work has turned to active-learning optimization—Bayesian optimization [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], bandit methods [17], evolutionary search [18], and reinforcement learning [19], [20], [21], [22]. These methods adaptively propose configurations from scalar PPA feedback, but operate almost entirely in the tool's parameter space. Design context is often absent or reduced to a handful of handcrafted numerical features, which are fragile and lack semantics. This lack of explicit and transferable design understanding therefore limits prior methods' ability to leverage historical experience across broader designs and technology nodes.

In parallel, large language models (LLMs) have inspired attempts to use pretrained models as flow-tuning [23], [24], [25], [26], [27], [28] assistants, prompting them with constraint files, logs, or textual summaries. LLMs can interpret human-readable descriptions and suggest plausible adjustments without explicit search heuristics. Yet existing approaches deploy LLMs in a purely in-context manner: the model is treated as a frozen oracle that reads a few examples and produces the next recommendation. There is no mechanism to integrate large-scale historical data into a parametric policy aligned with quantitative PPA outcomes, and finite context windows limit performance as design complexity or dataset size grows.

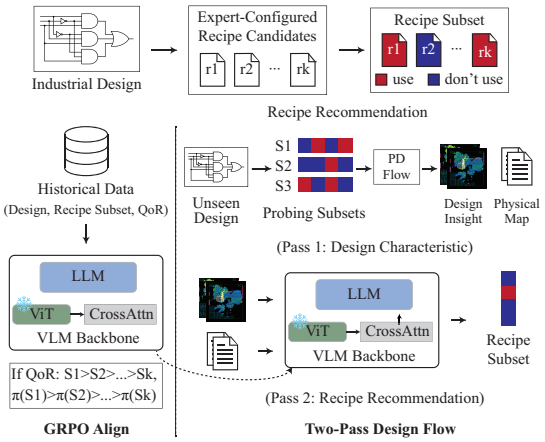

Fig. 1. Overview of our MapTuner flow

Existing optimization and LLM-based methods rely solely on textual features, overlooking physical layout maps (e.g., congestion, slack distribution, and critical-area placements) that best reflect true design behavior. Analyzing or debugging flows via physical maps in a GUI is powerful but highly labor-intensive and difficult to automate, making it practical only for critical or complex multi-issue scenarios.

To address these limitations, we introduce a multimodal, reward-aligned framework for physical-design flow tuning, as shown in Fig. 1. Specifically, we employ a vision–language model (VLM) that jointly interprets spatial layout maps and textual design insights, enabling direct understanding of a design's physical characteristics for flow recommendation. We further align the model using historical implementations through group-relative preference optimization (GRPO)[29], allowing it to learn a flow-tuning policy grounded in actual timing and power outcomes rather than relying solely on unconstrained semantic interpretation. Together, these components form a unified system that reuses past experience to deliver design-aware flow recommendations. Our main contributions are summarized as follows:

- This is among the first multimodal flow-tuning frameworks in EDA that integrates physical layout maps with design insights.

979-8-3195-1246-8-0/26/$31.00 ©2026 IEEE

- We present a flow-tuning framework that combines multimodal design understanding with scalable learning from large historical implementation datasets.
- We introduce a GRPO-based training scheme with intent-guided mixture-of-experts (MoE) reasoning that aligns the model with timing and power outcomes, achieving consistent PPA gains across 13 industrial designs.

II. Background and Preliminaries

A. Physical Maps in Multimodal EDA Systems

Physical maps are the most direct diagnostic tool used daily by physical design engineers to inspect designs. Early work [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40] primarily uses layout images for supervised prediction and layout optimization tasks. With the recent rise of LLMs in EDA [41], [42], [43], [44], [45], [46], [47], [48], [49], several multimodal systems [50], [51], [52] have incorporated physical maps into LLM pipelines, but their scope remains limited to defect or anomaly detection. [53] construct a multimodal Chip Physical Design Engineer Assistant that also utilizes physical maps to provide design suggestions, yet the maps are treated mainly as auxiliary prediction features rather than reasoning signals for flow tuning. In contrast, existing LLM-based flow-tuning approaches [23], [24], [25], [26], [27], [54], [55], [28] do not use physical maps. Thus, a multimodal framework jointly reasoning over design text and layout maps for flow tuning remains unexplored.

B. Group Relative Policy Optimization (GRPO)

Preference alignment is critical for steering Large Language Models (LLMs) to adhere to human intent. The standard approach, Proximal Policy Optimization (PPO) [56], optimizes single-sample trajectories in Reinforcement Learning from Human Feedback (RLHF) [57] by employing a clipped objective and a learned value-function critic to reduce variance. Direct Preference Optimization (DPO) [58] simplifies PPO via a pairwise classification objective, eliminating the RL loop. However, DPO is limited to binary comparisons, ignoring richer multi-candidate distributions. Group Relative Policy Optimization (GRPO) [29] overcomes these limitations by introducing a group-wise optimization framework that employs a group-relative baseline for advantage estimation. This mechanism allows GRPO to harness the stability of PPO-style clipped updates without a critic, while exploiting group-level feedback to achieve superior sample efficiency and convergence.

C. Flow Recipe

Commercial physical design tools expose thousands of parameters, creating a search space too large to explore directly. To keep tuning tractable, we use a set of expert preconfigured parameter bundles—called *recipes*—each designed to target a specific QoR intention. These recipes are distilled from human physical-design practice: each recipe bundles a coherent set of low-level tool settings corresponding to a commonly used intervention, so the discretization reflects representative expert actions rather than an arbitrary quantization of a continuous knob space. This also makes training samples meaningful, since they are collected over the same vetted action library. As shown in Fig. 1, our goal is to strategically combine a subset of these recipes to satisfy multiple design objectives. Let $\mathcal{R} = \{r_1, \ldots, r_{N_r}\}$ denote the set of candidate recipes, and let $\mathbf{d}$ represent the design context (e.g., structural metadata and spatial layout information). Each recipe r_i is controlled by a binary variable $s_i \in \{0, 1\}$ indicating activation, and the selected recipe subset is $\mathcal{S} = \{\, r_i \mid s_i = 1 \,\}$. In the evaluated commercial flow, the 65 recipes are curated to be mutually compatible at the stage level; mutually exclusive or order-dependent low-level settings are resolved during library construction, so MapTuner selects subsets only from this vetted library.

To formalize the objective above, we define the intent-weighted QoR so that lower is better, and we choose $\mathcal{S}$ by solving

$$\min_{\mathbf{s}\in\{0,1\}^{N_r}} \quad W_t\,\mathrm{Timing}(\mathbf{s},\mathbf{d}) + W_p\,\mathrm{Power}(\mathbf{s},\mathbf{d}). \tag{1}$$

Here, $\mathrm{Timing}(\mathbf{s},\mathbf{d})$ denotes the reported TNS magnitude $|\mathrm{TNS}|$ in ns, so smaller values mean less negative slack and 0 indicates timing closure; $\mathrm{Power}(\mathbf{s},\mathbf{d})$ denotes total power, which is also minimized. Hence lower is better for each term and for the combined QoR reported in Tables I and II.

D. Design Insight

Design representation has been a persistent challenge for ML-based flow tuning: differences in design architecture, scale, and technology node make generic numerical features unreliable in industrial settings. A practical and robust alternative—mirroring how PD engineers actually work—is to extract design insights from an initial P&R run, where modern tools natively surface design and PPA issues as structured diagnostic signals [28]. We use these tool-generated insights as part of the design description, enabling the model to reason about a design's underlying challenges rather than relying solely on black-box PPA metrics.

III. Methodology

Our method learns a design-aware flow-tuning policy by combining multimodal inputs with reward-aligned policy learning, as shown in Fig. 2. Each design is represented by its physical layout maps and tool-generated insights, providing a compact description of its implementation state. We finetune a VLM on historical designs by pairing each multimodal representation with evaluated recipe subsets and their timing–power outcomes, then apply GRPO to align the model's decision policy with true PPA objectives under different timing–power intents.

At inference time, the model consumes the multimodal description of a new design and generates a Yes/No decision sequence over all candidate recipes conditioned on the user's intent. The remainder of this section describes the multimodal representation, VLM backbone, intent-guided mixture-of-experts, prompting and output format, GRPO-based policy alignment, and inference pipeline.

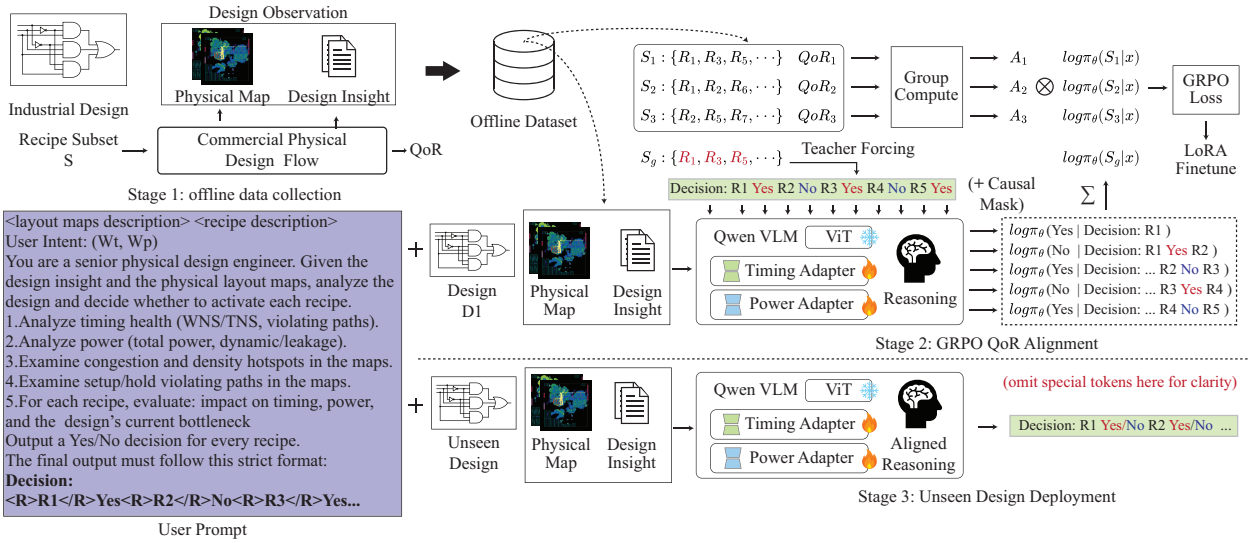

Fig. 2. Overview of the MapTuner training and deployment flow. Stage 1 collects multimodal design observations and QoR outcomes from commercial P&R runs. Stage 2 aligns the VLM using GRPO. Stage 3 deploys the aligned model to unseen designs to generate recipe decisions.

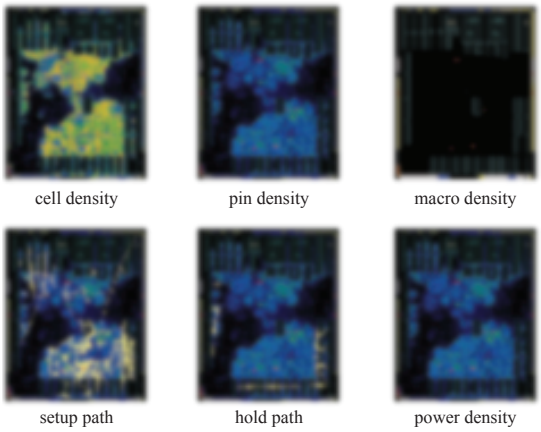

Fig. 3. Six physical layout maps used in our multimodal representation: cell density, pin density, macro placement, power distribution, and setup and hold timing paths.[1]

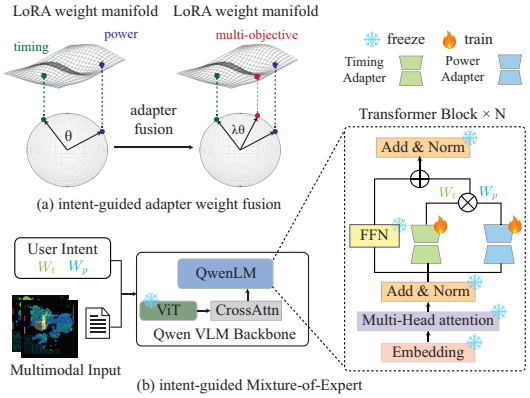

Fig. 4. The model blends timing and power LoRA experts using intent-dependent mixture weights and applies the mixed adapter within each transformer block.

A. Multimodal Representation and Backbone

Each design is represented by two modalities: (1) design insights (Sec. II-D), and (2) six layout maps: cell density, pin density, congestion, timing-path distribution, power density, and macro placement (Fig. 3). To process this multimodal input, we adopt the Qwen3-VL [59] backbone, which combines a pretrained vision-transformer encoder with a decoder-only language model. Visual tokens are projected into the LLM embedding space and fused through cross-modal attention layers, enabling the model to jointly interpret layout maps and design-insight text within a unified token sequence.

B. Intent-Guided Adapter Mixture-of-Experts

To modulate the model under different timing–power intents, we introduce an *intent-guided adapter mixture-of-experts* (Adapter-MoE), illustrated in Fig. 4. For each attention projection matrix (e.g., $q/k/v/o$) across all transformer blocks, we attach two LoRA adapter experts: a timing expert and a power expert. Each expert provides a low-rank update $\Delta W_i = A_i B_i^\top$ specialized for its target PPA objective.

Given a multimodal design description $\mathbf{d}$ and user intent $\mathbf{u}$, a gating function $g(\mathbf{d}, \mathbf{u}) \in \Delta^1$ produces intent-conditioned mixture weights $\mathbf{w} = [w_{\text{tim}}, w_{\text{pow}}]$. Operationally, g is a small gating head that maps $\mathbf{d}$ and $\mathbf{u} = [W_t, W_p]$ to normalized weights with $w_{\text{tim}} + w_{\text{pow}} = 1$. The effective adapter update is therefore

$$\Delta W_{\text{mix}} = w_{\text{tim}} \Delta W_{\text{tim}} + w_{\text{pow}} \Delta W_{\text{pow}}, \tag{2}$$

yielding the adapted transformation

$$h' = (W + \Delta W_{\text{mix}})\, h. \tag{3}$$

As visualized in Fig. 4(a), the timing and power experts define two directions in the adapter weight space, and the mixture weights trace a continuous interpolation between them, yielding controllable multi-objective behavior.

C. Chain-of-Thought Prompting and Decision Generation

Given a multimodal design description $\mathbf{x}$, the model first produces an intermediate reasoning process followed by a strict, machine-parsable decision output. Instead of predicting

[1]For confidentiality reasons, the maps shown here are visually obfuscated.

Algorithm 1 MapTuner Training

Input: initial VLM parameters θ_{init}; historical designs $\mathcal{D}$.
Shared probe subsets:
$\mathcal{P} = \{S_1^{\text{probe}}, \ldots, S_m^{\text{probe}}\}$ probe subsets (shared across designs)
For each design $d \in \mathcal{D}$:
$\mathcal{G}_d = \{S_{d,1}, \ldots, S_{d,k_d}\}$ evaluated recipe subsets
$(q_{d,j}^{\text{tim}}, q_{d,j}^{\text{pow}})$ QoR per subset
Global hyperparameters:
(W_t, W_p), B, η, E, G intent weights; batch size; learning rate; epochs; GRPO group size
Output: trained parameters θ

Offline preprocessing:
for each design $d \in \mathcal{D}$ **do**
  Run commercial PD flow on (d, S_i^{probe}) for all i.
  Store probe maps and insights as design observation $\mathbf{z}_d$.
Training:
$\theta \leftarrow \theta_{\text{init}}$
for $e = 1, \ldots, E$ **do**
  for each mini-batch of designs $\mathcal{B} \subset \mathcal{D}$ with $|\mathcal{B}| = B$ **do**
    $\mathcal{L} \leftarrow 0$
    for each design $d \in \mathcal{B}$ **do**
      Load $\mathbf{z}_d$
      $x_d \leftarrow (\mathbf{z}_d, W_t, W_p)$ ▷ Multimodal input with intent
      for each subset $S_{d,j} \in \mathcal{G}_d$ **do**
        $r_{d,j} \leftarrow -\big(W_t\, q_{d,j}^{\text{tim}} + W_p\, q_{d,j}^{\text{pow}}\big)$
      $\{A_{d,j}\} \leftarrow \text{GroupAdv}(\{r_{d,j}\}_j, G)$ ▷ Group Computation
      for each subset $S_{d,j} \in \mathcal{G}_d$ **do**
        ▷ Teacher-forcing compute subset likelihood
        $y_{d,j} \leftarrow \text{DecisionSequence}(S_{d,j})$
        $\log \pi_\theta(S_{d,j} \mid x_d) \leftarrow \sum_t \log p_\theta\big(y_{d,j,t} \mid y_{d,j,<t}, x_d\big)$
        $\mathcal{L} \mathrel{-}= A_{d,j}\, \log \pi_\theta(S_{d,j} \mid x_d)$
    $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$ ▷ LoRA update

each recipe independently, the VLM generates a chain-of-thought that analyzes timing, power, and congestion evidence from the maps and insights, and then outputs a fixed-format decision sequence.

a) Prompt template.: We design a deterministic prompt requiring the model to (i) analyze key design phenomena and (ii) produce a rigid token pattern:

```
Decision: <R>R_1</R> Yes <R>R_2</R> No …
```

where each recipe r_t is referenced by a fixed `<R>name</R>` tag followed by a binary choice token `Yes` or `No`. This rigid format enables unambiguous extraction of token-level log-probabilities, which is required for policy learning.

b) Policy structure.: Let $\mathcal{S}_{t-1}$ denote prior decisions emitted in the sequence. The decision policy factorizes autoregressively as:

$$s_t \sim \pi_\theta(s_t \mid r_t, \mathbf{x}, \mathcal{S}_{t-1}), \qquad s_t \in \{0, 1\}. \tag{4}$$

D. GRPO-Based Policy Alignment for PPA Objectives

Predicting absolute QoR across designs is unreliable due to variations in scale, node, and recipe sensitivity, causing unstable cross-design regression. Instead of requiring the model to estimate precise timing or power values, we formulate flow tuning as a relative ranking problem: within each design, the model only needs to distinguish better recipe subsets from worse ones. This relative formulation is far more stable across heterogeneous designs and provides a natural signal for policy alignment. The overall training procedure is summarized in Algorithm 1.

To train the model, we use historical data $(\mathbf{x}, \mathcal{S}, \text{QoR})$ consisting of multimodal design descriptions, recipe subsets, and their QoR outcomes. As shown in Algorithm 1 and Fig. 2,

Algorithm 2 MapTuner: Deployment on an Unseen Design

Input: trained parameters θ; unseen design $d^\star$; fixed probe subsets $\mathcal{P}_{d^\star} = \{S_1^{\text{probe}}, \ldots, S_m^{\text{probe}}\}$; user intent weights (W_t, W_p)
Output: recommended recipe subset $\hat{S}$ for $d^\star$

// Pass 1: Probe-based design characterization
for $i = 1, \ldots, m$ **do**
  Run commercial PD flow on $(d^\star, S_i^{\text{probe}})$
  Store probe maps and insights as design observation $\mathbf{z}_d$.
Load $\mathbf{z}_d$
$x_{d^\star} \leftarrow (\mathbf{z}_{d^\star}, W_t, W_p)$
// Pass 2: Recipe subset recommendation
Initialize decoder with prompt and $x_{d^\star}$
Generate decision sequence $y^\star \sim \pi_\theta(\cdot \mid x_{d^\star})$ ▷ e.g., greedy or nucleus sampling
$\hat{S} \leftarrow \text{ParseDecisionSequence}(y^\star)$ ▷ map "Yes/No" to each recipe $R_1, \ldots, R_k$
return $\hat{S}$

we first compute the log-likelihood of the supervised decision sequence through teacher forcing (Step 3). This provides the token-level log-probabilities $\log \pi_\theta(\mathcal{S} \mid \mathbf{x})$ required for preference-based optimization:

$$\log \pi_\theta(\mathcal{S} \mid \mathbf{x}) = \sum_{t=1}^{T} \log \pi_\theta\big(s_t \mid r_t, \mathbf{x}, \mathcal{S}_{t-1}\big). \tag{5}$$

We then align these likelihoods with timing–power objectives using GRPO. Let $q_{\text{tim}}^{(g)}$ denote the lower-is-better timing penalty (TNS magnitude) and $q_{\text{pow}}^{(g)}$ the lower-is-better power metric. Algorithm 1 converts this penalty-form QoR into the higher-is-better reward

$$r^{(g)} = -\big(W_t\, q_{\text{tim}}^{(g)} + W_p\, q_{\text{pow}}^{(g)}\big). \tag{6}$$

or equivalently, with $t^{(g)} = -q_{\text{tim}}^{(g)}$ and $p^{(g)} = -q_{\text{pow}}^{(g)}$,

$$R^{(g)} = r^{(g)} = W_t\, t^{(g)} + W_p\, p^{(g)}. \tag{7}$$

standardized as

$$A^{(g)} = \frac{R^{(g)} - \mu_R}{\sigma_R}, \tag{8}$$

so that GRPO can emphasize configurations that perform relatively better under the specified intent. With a group size of 16 (Steps 2–3 in Algorithm 1), each update compares sixteen subsets from the same design and optimizes

$$\mathcal{L}_{\text{GRPO}} = -\frac{1}{|\mathcal{G}|} \sum_g A^{(g)}\, \log \pi_\theta(\mathcal{S}^{(g)} \mid \mathbf{x}^{(g)}), \tag{9}$$

thereby increasing the likelihood of high-performing subsets and suppressing inferior ones. Thus the tables report QoR in penalty form, while GRPO optimizes the sign-flipped reward. Teacher forcing supplies the supervised log-probabilities needed by GRPO without requiring absolute QoR regression.

E. Inference

For a new design $d^\star$, MapTuner follows the two-pass procedure summarized in Algorithm 2. First, the fixed probe subsets are executed to obtain the design's physical maps and insights, forming the multimodal description $\mathbf{x}_{d^\star}$. Together with the user-specified timing–power intent $\mathbf{u}$, this input is used by the gating network to produce mixture weights $\mathbf{w} = g(\mathbf{x}_{d^\star}, \mathbf{u})$ which determine the intent-conditioned adapter update ΔW_{mix} (Eq. 2) and the resulting policy $\pi_{\theta,\mathbf{w}}$. Larger W_t shifts the gate toward the timing expert, while larger W_p shifts it toward the power expert.

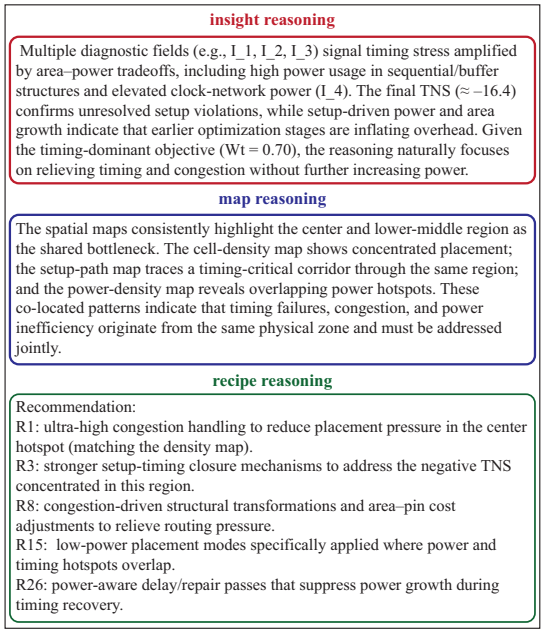

Fig. 5. Reasoning trace illustrating how the model combines insights, layout maps, and recipe logic to form its final decisions.

The policy then generates a structured analysis and a complete Yes/No decision sequence through autoregressive decoding:

$$y^\star = \arg\max_{y} \pi_{\theta,\mathbf{w}}(y \mid \mathbf{x}_{d^\star}). \tag{10}$$

Because the prompt enforces a fixed ordering over recipes, the decisions in $y^\star$ can be parsed deterministically to obtain the recommended subset

$$\mathcal{S}_{d^\star} = \{\, r_t : r_t \text{ is labeled } \texttt{Yes} \text{ in } y^\star \,\}. \tag{11}$$

Through this process, the user intent guides the adapter mixture and thus directly steers the model's decision policy and final recommendations without any retraining.

IV. Experimental Results

A. Experimental Setup

We implemented MapTuner in PyTorch on NVIDIA A100 GPUs. We evaluated 13 industrial benchmarks spanning 45 nm to sub-10 nm and up to 2M gates using a commercial flow through post-placement optimization. Our optimization space consists of 65 distinct recipes. For GRPO alignment, we use 4,800 datapoints from 13 designs under diverse recipe combinations. For the two-pass deployment (Algorithm 2), we set the number of fixed probe subsets $m = 1$, adding minimal overhead relative to search-based methods that require hundreds of evaluations.

B. Multimodal Reasoning Analysis

To verify multimodal reasoning, we analyze the model's `<think>` trace (Fig. 5[2]). The trace decomposes into three parts: (i) *insight reasoning* (referencing tool diagnostics); (ii) *map reasoning* (interpreting spatial hotspots and paths); and (iii) *recipe reasoning* (integrating observations to select flow actions). These components provide direct evidence that the model performs genuine multimodal reasoning over both the symbolic design insights and the spatial map patterns.

[2]Condensed and redacted for confidentiality.

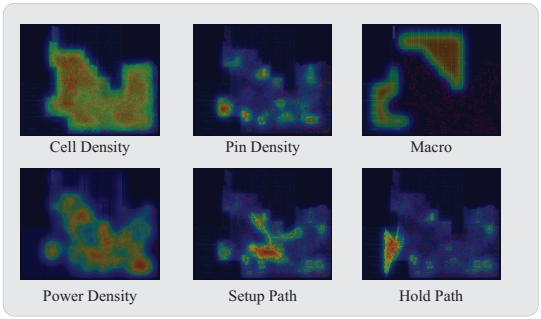

Fig. 6. Activation maps indicating the regions of the layout most strongly attended by the model during the forward pass.

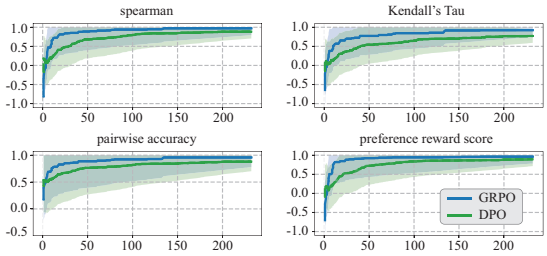

Fig. 7. GRPO convergence showing improved ranking quality across four evaluation metrics.

To understand how the model processes the layout maps, we inspect the vision encoder's intermediate patch activations during the forward pass (Fig. 6). The activations highlight the hotspot regions of each map—the same areas referenced in the map-reasoning portion of the `<think>` trace—indicating that the model is genuinely attending to and interpreting the visual content.

C. GRPO Training Curve

Figure 7 reports the convergence behavior of our GRPO training objective. With a group size of 16, each update learns a relative ordering over 16 recipe subsets from the same design. The four metrics—Spearman correlation, Kendall's Tau, pairwise accuracy, and preference-reward score—measure how well the model's predicted ranking matches the ground-truth QoR ordering within each group; higher values indicate stronger ranking fidelity and more stable reward alignment. All metrics are evaluated on a held-out validation group of the same size 16.

We additionally compare GRPO against Direct Preference Optimization (DPO), a widely used LLM preference-alignment method. Under the same forward-pass budget, both methods use 16 subset likelihood evaluations per optimization step; DPO samples 8 random preference pairs per update and therefore learns only pairwise constraints rather than full group structure. As shown in the figure, GRPO converges faster and achieves higher ranking metrics across all measures, indicating that group-relative preference learning provides a richer signal than pairwise DPO.

TABLE I

RESULTS UNDER THE POWER-INTENT OBJECTIVE (0.3 TNS / 0.7 POWER), WITH QOR AS THE INTENT-WEIGHTED OBJECTIVE METRIC (LOWER IS BETTER). FOR EACH DESIGN, WE REPORT THE ACHIEVED TNS, POWER, AND INTENT-WEIGHTED QOR ACROSS ALL EVALUATED BENCHMARKS.

design	Best Known			GPT-4o			GPR			MapTuner			tech. node	size
	TNS (ns)	Power (mW)	QoR Score	TNS (ns)	Power (mW)	QoR Score	TNS (ns)	Power (mW)	QoR Score	TNS (ns)	Power (mW)	QoR Score		
D1	33.53	458.57	-4.35	42.57	1184.83	-0.08	17.55	1208.18	-0.04	32.21	923.27	-1.43	16nm	584452
D2	21.61	2031.70	-1.54	19.92	2052.62	-0.64	19.87	2049.63	-0.78	13.18	2038.56	-1.55	5nm	276676
D3	14.68	100.13	-1.77	5.55	112.50	-0.42	6.42	108.68	-0.93	9.98	104.44	-1.37	5nm	152851
D4	0.04	189.67	-1.25	0.04	192.56	-0.45	0.01	192.68	-0.47	0.04	190.21	-1.09	7nm	117868
D5	19.50	66.03	-0.74	22.15	67.14	-0.31	24.59	66.54	-0.44	36.02	63.92	-0.96	7nm	130521
D6	13.25	134.22	-1.09	15.07	137.10	-0.61	12.31	138.46	-0.61	16.28	132.14	-1.16	5nm	118017
D7	1.34	68.93	-1.55	0.24	73.84	-0.40	0.18	73.83	-0.42	1.48	67.88	-1.82	40nm	68200
D8	104.08	12.80	-1.16	303.14	13.13	-0.07	142.97	13.16	-0.73	72.24	12.21	-1.73	45nm	45713
D9	0.84	0.02	-2.01	0.01	0.05	0.16	0.15	0.05	0.11	0.12	0.02	-2.00	7nm	1055050
D10	120.93	386.51	-0.40	117.54	418.54	0.32	278.53	377.70	-0.08	115.24	289.53	-2.64	12nm	740429
D11	20.99	44.38	-0.93	26.69	45.26	-0.46	21.87	45.73	-0.37	24.03	44.04	-1.01	12nm	156920
D12	0.04	650.40	-0.95	0.02	664.04	-0.15	0.13	652.93	-0.71	0.04	642.84	-1.41	7nm	319230
D13	0.00	0.71	-0.72	0.00	2.88	0.46	0.00	0.74	-0.70	0.00	0.71	-0.71	7nm	230721

TABLE II

RESULTS UNDER THE TIMING-INTENT OBJECTIVE (0.7 TNS / 0.3 POWER), WITH QOR AS THE INTENT-WEIGHTED OBJECTIVE METRIC (LOWER IS BETTER). FOR EACH DESIGN, WE REPORT THE ACHIEVED TNS, POWER, AND INTENT-WEIGHTED QOR ACROSS ALL EVALUATED BENCHMARKS.

design	Best Known			GPT-4o			GPR			MapTuner			tech. node	size
	TNS (ns)	Power (mW)	QoR Score	TNS (ns)	Power (mW)	QoR Score	TNS (ns)	Power (mW)	QoR Score	TNS (ns)	Power (mW)	QoR Score		
D1	14.59	1181.68	-0.38	16.85	1210.46	-0.29	18.43	1215.81	-0.26	32.21	923.27	-0.88	16nm	584452
D2	16.61	2046.56	-1.04	22.61	2069.58	-0.03	19.26	2077.43	-0.18	13.18	2038.56	-1.51	5nm	276676
D3	3.55	108.11	-1.21	7.05	109.46	-0.72	7.83	109.17	-0.65	3.93	108.70	-1.13	5nm	152851
D4	0.04	189.67	-0.56	0.05	191.89	-0.34	0.03	192.54	-0.36	0.02	191.59	-0.50	7nm	117868
D5	18.10	66.60	-0.76	20.87	67.05	-0.52	21.35	66.89	-0.51	15.62	67.19	-0.83	7nm	130521
D6	12.51	136.02	-0.96	14.59	138.14	-0.54	14.21	138.52	-0.57	10.96	137.87	-1.09	5nm	118017
D7	0.10	72.76	-0.62	0.07	74.30	-0.45	0.13	74.22	-0.43	0.47	68.78	-0.93	40nm	68200
D8	72.65	14.20	-0.51	89.02	14.73	-0.18	99.93	14.11	-0.27	60.28	13.43	-0.88	45nm	45713
D9	0.05	0.03	-1.03	0.03	0.05	-0.19	0.06	0.04	-0.23	0.12	0.02	-1.06	7nm	1055050
D10	102.51	392.29	-0.61	136.34	424.34	-0.03	127.70	380.35	-0.53	115.24	289.53	-1.49	12nm	740429
D11	19.22	44.92	-0.73	21.36	45.74	-0.49	21.19	45.64	-0.51	17.79	44.67	-0.84	12nm	156920
D12	0.08	649.84	-0.48	0.03	661.94	-0.28	0.05	654.82	-0.42	0.04	642.84	-0.77	7nm	319230
D13	0.00	0.71	-0.35	0.00	0.74	-0.34	0.00	0.75	-0.34	0.00	0.71	-0.35	7nm	230721

D. Recipe Recommendation Performance

We evaluate MapTuner using two-fold cross-validation across 13 designs. In each fold, a subset is held out for testing while the remainder is used for training, so all reported metrics reflect unseen designs. We consider two compound-QoR optimization intents: Power-Intent (0.3 TNS / 0.7 Power) and Timing-Intent (0.7 TNS / 0.3 Power).

For each held-out design and intent, the model generates ten candidate recipe subsets. Both GPR and GPT-4o use the same sampling budget, and we report the best QoR among the ten predictions for all methods. Tables I and II compare MapTuner against two baselines: (i) a Gaussian-process regressor (GPR), representing Bayesian optimization with historical data; and (ii) GPT-4o, a zero-shot multimodal LLM given the same prompt and map inputs without task-specific fine-tuning. Since GPT-4o is closed-source and cannot be fine-tuned on design history, the comparison isolates the benefit of GRPO-based adaptation for recipe selection in physical-design flow tuning.

Under the Power-Intent setting, MapTuner improves both power and compound score for D5, D6, D7, D8, D10, D11, and D12, outperforming the best-known offline datapoints. For D3 and D4, where historical datapoints are already highly optimized, MapTuner remains competitive and stays close to the empirical bound. In contrast, GPT-4o exhibits larger variance and often produces weaker power results, suggesting that zero-shot multimodal prompting is less effective here than task-specific GRPO adaptation. GPR generally performs better than GPT-4o but degrades on designs with irregular or non-smooth QoR landscapes, highlighting the limitation of black-box regression under heterogeneous design behaviors.

Overall, MapTuner adapts robustly to different optimization intents and generalizes effectively to held-out designs. Across both scenarios, it produces recipe subsets that match or surpass the best-known offline configurations, capturing transferable optimization patterns for unseen designs.

V. CONCLUSION

We presented MapTuner, a flow-tuning framework that combines multimodal understanding and scalable learning from historical implementation data. By integrating layout maps and design insights into a VLM and aligning it through GRPO with intent-guided adapter mixtures, MapTuner learns a ranking-based tuning policy across heterogeneous designs. Experiments on 13 industrial designs show consistent timing–power gains over GPT-4o, Gaussian-process regressors, and strong commercial baselines under both intents. These results support practical, interpretable multimodal flow tuning with design-aware recommendations for industrial settings across diverse nodes and design scales.

References

[1] Y.-H. Chung *et al.*, "SimPart: A simple yet effective replication-aided partitioning algorithm for logic simulation on GPU," in *Proc. Euro-Par*, 2025.

[2] Y.-H. Chung, N. Oh, M. G. Balabhadra Naga Venkata, A. Shiledar, S. Kundu, V. Khandelwal, and T.-W. Huang, "Accelerating gate sizing using GPU," in *International European Conference on Parallel and Distributed Computing (Euro-Par) PhD Symposium*, Dresden, Germany, 2025.

[3] T. Akiba *et al.*, "Optuna: A next-generation hyperparameter optimization framework," in *Proc. KDD*, 2019.

[4] Z. Zhang *et al.*, "A Fast Parameter Tuning Framework via Transfer Learning and Multi-Objective Bayesian Optimization," in *Proc. DAC*, 2022.

[5] H. Geng *et al.*, "PTPT: Physical design tool parameter tuning via multi-objective Bayesian optimization," *IEEE TCAD*, 2023.

[6] H. Geng *et al.*, "PPATuner: Pareto-driven tool parameter auto-tuning in physical design via Gaussian process transfer learning," in *Proc. DAC*, 2022.

[7] Y. Ma *et al.*, "CAD tool design space exploration via Bayesian optimization," in *Proc. MLCAD*, 2019.

[8] S. Zheng *et al.*, "Boosting vlsi design flow parameter tuning with random embedding and multi-objective trust-region bayesian optimization," *ACM TODAES*, 2023.

[9] D. Luo *et al.*, "Attention-Based EDA Tool Parameter Explorer: From Hybrid Parameters to Multi-QoR metrics," in *Proc. DATE*, 2024.

[10] E. Ustun *et al.*, "LAMDA: Learning-Assisted Multi-stage Autotuning for FPGA Design Closure," in *Proc. FCCM*, 2019.

[11] Z. Xie *et al.*, "FIST: A Feature-Importance Sampling and Tree-Based Method for Automatic Design Flow Parameter Tuning," in *Proc. ASP-DAC*, 2020.

[12] J. Kwon *et al.*, "A Learning-Based Recommender System for Autotuning Design Flows of Industrial High-Performance Processors," in *Proc. DAC*, 2019.

[13] C. Bai *et al.*, "BOOM-Explorer: RISC-V BOOM Microarchitecture Design Space Exploration," *ACM TODAES*, 2023.

[14] Z. Yu *et al.*, "IT-DSE: Invariance Risk Minimized Transfer Microarchitecture Design Space Exploration," in *Proc. ICCAD*, 2023.

[15] A. Agnesina *et al.*, "Autodmp: Automated dreamplace-based macro placement," in *Proc. ISPD*, 2023.

[16] P. Xu *et al.*, "Ranktuner: When design tool parameter tuning meets preference bayesian optimization," in *Proc. ICCAD*, 2024.

[17] F. Liu *et al.*, "Cbtune: Contextual bandit tuning for logic synthesis," in *Proc. DATE*, 2024.

[18] R. Liang *et al.*, "FlowTuner: A multi-stage EDA flow tuner exploiting parameter knowledge transfer," in *Proc. ICCAD*, 2021.

[19] A. Agnesina *et al.*, "VLSI placement parameter optimization using deep reinforcement learning," in *Proc. ICCAD*, 2020.

[20] H.-H. Hsiao, Y.-C. Lu, P. Vanna-Iampikul, and S. K. Lim, "FastTuner: Transferable physical design parameter optimization using fast reinforcement learning," in *Proceedings of the 2024 International Symposium on Physical Design*, 2024, pp. 93–101.

[21] H.-H. Hsiao, Y.-C. Lu, P. Vanna-Iampikul, and S. K. Lim, "A hybrid reinforcement learning framework for efficient physical design parameter tuning," *ACM Transactions on Design Automation of Electronic Systems*, vol. 31, no. 3, pp. 50:1–50:27, 2026.

[22] H.-H. Hsiao, P. Vanna-Iampikul, Y.-C. Lu, and S. K. Lim, "ML-based physical design parameter optimization for 3D ICs: From parameter selection to optimization," in *Proceedings of the 61st ACM/IEEE Design Automation Conference (DAC)*, 2024, pp. 252:1–252:6.

[23] Y. Yin *et al.*, "Ado-llm: Analog design bayesian optimization with in-context learning of large language models," in *Proc. ICCAD*, 2024.

[24] P. Xu *et al.*, "RATuner: Retrieval-Augmented VLSI Flow Design Parameter Tuning Framework," in *Proc. DATE*, 2026.

[25] Z. Yu *et al.*, "CausalTuner: Will Causality Help High-Dimensional EDA Tool Parameter Tuning," in *Proc. ASP-DAC*, 2026.

[26] J. Pan *et al.*, "Crop: Circuit retrieval and optimization with parameter guidance using LLMs," *arXiv preprint arXiv:2507.02128*, 2025.

[27] M. Chen *et al.*, "Hyperplace: Harnessing a large language model for efficient hyperparameter optimization in gpu-accelerated vlsi placement," *ACM TODAES*, 2025.

[28] H.-H. Hsiao, S. Kundu, W. Zeng, W.-T. J. Chan, D. Guo, and S. K. Lim, "InsightAlign: A transferable physical design recipe recommender based on design insights," in *Proceedings of the 62nd ACM/IEEE Design Automation Conference (DAC)*, 2025.

[29] D. Guo *et al.*, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," *arXiv preprint arXiv:2501.12948*, 2025.

[30] S. Kim *et al.*, "Methodology of Resolving Design Rule Checking Violations Coupled with Fully Compatible Prediction Model," in *Proc. ISPD*, 2024.

[31] Z. Xie *et al.*, "RouteNet: Routability prediction for Mixed-Size Designs Using Convolutional Neural Network," in *Proc. ICCAD*, 2018.

[32] R. Liang *et al.*, "Drc hotspot prediction at sub-10nm process nodes using customized convolutional network," in *Proc. ISPD*, 2020.

[33] Y. Zhao *et al.*, "Hybridnet: Dual-branch fusion of geometrical and topological views for vlsi congestion prediction," in *Proc. ISEDA*, 2023.

[34] S. Liu *et al.*, "Global placement with deep learning-enabled explicit routability optimization," in *Proc. DATE*, 2021.

[35] A. Kahng *et al.*, "Placement tomography-based routing blockage generation for drv hotspot mitigation," in *Proc. ICCAD*, 2024.

[36] Y.-C. Lu, H. Ren, H.-H. Hsiao, and S. K. Lim, "Dream-gan: Advancing dreamplace towards commercial-quality using generative adversarial learning," in *Proceedings of the 2023 International Symposium on Physical Design*, ser. ISPD '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 141–148.

[37] Y.-C. Lu, H. Ren, H.-H. Hsiao, and S. K. Lim, "GAN-Place: Advancing Open Source Placers to Commercial-quality Using Generative Adversarial Networks and Transfer Learning," *ACM Transactions on Design Automation of Electronic Systems*, vol. 29, no. 2, pp. 32:1–32:17, 2024.

[38] H.-H. Hsiao, Y.-C. Lu, P. Vanna-Iampikul, A. Agnesina, R. Liang, Y.-H. Lu, H. Ren, and S. K. Lim, "DCO-3D: Differentiable congestion optimization in 3D ICs," in *2025 62nd ACM/IEEE Design Automation Conference (DAC)*, 2025, pp. 1–7.

[39] Y.-C. Lu, H.-H. Hsiao, R. Liang, W.-H. Liu, and H. Ren, "C3PO: Commercial-quality global placement via coherent, concurrent timing, routability, and wirelength optimization," in *31st Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2026, pp. 765–771.

[40] Y.-H. Lu, H.-H. Hsiao, Y.-C. Lu, H. Ren, and S. K. Lim, "Differentiable tier assignment for timing and congestion-aware routing in 3d ICs," in *31st Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2026, pp. 475–481.

[41] R. Zhong *et al.*, "Llm4eda: Emerging progress in large language models for electronic design automation," *arXiv preprint arXiv:2401.12224*, 2023.

[42] W. Fang *et al.*, "A survey of circuit foundation model: Foundation ai models for vlsi circuit design and eda," *arXiv preprint arXiv:2504.03711*, 2025.

[43] Y. Lu *et al.*, "Autoeda: Enabling eda flow automation through microservice-based llm agents," *arXiv preprint arXiv:2508.01012*, 2025.

[44] I. Villanueva *et al.*, "A multimodal exploration of engineering students' emotions and electrodermal activity in design activities," *Journal of Engineering Education*, 2018.

[45] Z. Shi *et al.*, "Forgeeda: A comprehensive multimodal dataset for advancing eda," *arXiv preprint arXiv:2505.02016*, 2025.

[46] W. Fang *et al.*, "Geneda: Unleashing generative reasoning on netlist via multimodal encoder-decoder aligned foundation model," *arXiv preprint arXiv:2504.09485*, 2025.

[47] Y.-C. Lu, H.-H. Hsiao, and H. Ren, "LLM-enhanced GPU-optimized physical design at scale," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2025, pp. 1–7.

[48] Y.-C. Lu, H.-H. Hsiao, R. Liang, and H. Ren, "AI-assisted IC design with accelerated computing and LLM-driven optimization," *IEEE Solid-State Circuits Magazine*, 2026.

[49] H.-H. Hsiao, Y.-C. Lu, S. K. Lim, and H. Ren, "BUFFALO: PPA-configurable, LLM-based buffer tree generation via group relative policy optimization," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2025, pp. 1–9.

[50] X. Zhou *et al.*, "Deploying edge llms for wafer defect detection in chip manufacturing," in *Proc. ISEDA*, 2025.

[51] B. Zhao *et al.*, "Image-intrinsic priors for integrated circuit defect detection and novel class discovery via self-supervised learning," *arXiv preprint arXiv:2511.03120*, 2025.

[52] Y. Jiang *et al.*, "Fabgpt: An efficient large multimodal model for complex wafer defect knowledge queries," in *Proc. ICCAD*, 2024.

[53] Y.-D. Tsai *et al.*, "Multimodal chip physical design engineer assistant," *arXiv preprint arXiv:2510.15872*, 2025.

[54] Y. Wang *et al.*, "Mcp4eda: Llm-powered model context protocol rtl-to-gdsii automation with backend aware synthesis optimization," *arXiv preprint arXiv:2507.19570*, 2025.

[55] K. Xu *et al.*, "Llm-aided efficient hardware design automation," *arXiv preprint arXiv:2410.18582*, 2024.

[56] J. Schulman *et al.*, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[57] Y. Bai *et al.*, "Training a helpful and harmless assistant with reinforcement learning from human feedback," *arXiv preprint arXiv:2204.05862*, 2022.

[58] R. Rafailov *et al.*, "Direct preference optimization: Your language model is secretly a reward model," *Proc. NeurIPS*, 2023.

[59] J. Bai *et al.*, "Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond," *arXiv preprint arXiv:2308.12966*, 2023.