Multimodal Chip Physical Design Engineer Assistant

Anonymous Author(s)
Affiliation
Address
email

Abstract

Modern chip physical design relies heavily on Electronic Design Automation (EDA) tools, which often struggle to provide interpretable feedback or actionable guidance for improving routing congestion. In this work, we introduce a Multimodal Large Language Model Assistant (MLLMA) that bridges this gap by not only predicting congestion but also delivering human-interpretable design suggestions. Our method combines automated feature generation through MLLM-guided genetic prompting with an interpretable preference learning framework that models congestion-relevant tradeoffs across visual, tabular, and textual inputs. We compile these insights into a "Design Suggestion Deck" that surfaces the most influential layout features and proposes targeted optimizations. Experiments on the CircuitNet benchmark demonstrate that our approach outperforms existing models on both accuracy and explainability. Additionally, our design suggestion guidance case study and qualitative analyses confirm that the learned preferences align with real-world design principles and are actionable for engineers. This work highlights the potential of MLLMs as interactive assistants for interpretable and context-aware physical design optimization.

1 Introduction

The field of chip physical design encompasses critical prediction tasks such as congestion, timing, Design Rule Check (DRC), and IR Drop. These tasks, traditionally tackled using Electronic Design Automation (EDA) tools, often suffer from slow processing speeds and limited ability to provide actionable guidance for design improvement. Existing approaches using image-to-image translation models focus primarily on prediction accuracy but lack interpretability and the ability to suggest corrective measures. This paper explores the potential of Multimodal Large Language Models (MLLMs) in providing interpretable predictions along with actionable suggestions to assist engineers in optimizing their designs. Recent advances in foundation models and instruction-tuned Large Language Models (LLMs) have led to impressive progress in tasks that require reasoning, code generation, and interactive assistance. However, their application to chip physical design remains largely underexplored. In particular, current EDA solutions lack the capability to provide real-time, explainable, and multimodal assistance that integrates layout visuals, tabular metrics, and design constraints into a unified decision-making process. We argue that a Multimodal Language Model Assistant (MLLMA) tailored to chip physical design can fill this gap.

The motivation for this research is to address the gap between prediction accuracy and interpretability in physical design tasks. While current models are adept at predicting various metrics such as congestion and DRC violations, they fail to explain the underlying causes or offer corrective measures. By leveraging multimodal LLMs, we aim to move beyond mere prediction and offer a framework that provides both predictions and accompanying explanations. These explanations can be compiled into a "Design Suggestion Deck" which assists engineers by presenting comprehensible, human-readable reasoning to guide their design decisions. The primary research challenge is developing an MLLM-

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

based agent framework capable of processing multimodal data inputs, including geometric images, tabular features, and circuit graphs, and providing interpretable predictions with actionable design suggestions. Furthermore, we aim to compare this approach against baseline models to demonstrate its effectiveness in improving design quality and interpretability.

Our method consists of two stages: (1) Automate Feature Engineering and Generation, and (2) Interpretable Preferences Learning to Design Suggestion Deck. In the first stage, a Multimodal Large Language Model (MLLM) agent generates and extracts numeric features from Macro Region, RUDY, and RUDY pin images and text-based features from configuration and logs. Inspired by the Genetic Instruct (Majumdar et al., 2024), a genetic algorithm incorporating mutation and crossover is applied to expand the multimodal feature pool, with a deduplication process ensuring uniqueness. Feature importance and cross validation scores are evaluated using a Random Forest model to guide feature selection and refinement. The second stage uses these features and their importance scores to create a Design Suggestion Deck. The MLLM agent translates important features into interpretable rules, providing actionable guidance for chip design optimization.

Our contributions are summarized as follows:

- Development of a multimodal LLM framework capable of handling geometric image features, tabular data, and circuit graph inputs for prediction tasks in physical design.
- Design of an interpretable preference learning approach that generates interpretable predictions and actionable suggestions compiled into a "Design Suggestion Deck".
- Evaluation of the proposed framework against the baseline models that demonstrate strong improvements in design quality and efficiency.

2 Related Work

2.1 Multimodal Language Model Assistant

Recent works such as Mipha (Zhu et al., 2024), MM-ReAct (Yang et al., 2023), and MultiModal-GPT (Gong et al., 2023) demonstrate that aligning visual inputs with language reasoning modules allows models to solve complex real-world tasks. Mipha, for example, focuses on efficiency by enabling multi-turn multimodal dialogue with domain grounding, while MM-ReAct leverages reasoning traces and visual memory for tool-augmented perception.

In the context of chip design, prior efforts such as Chip-Chat (Blocklove et al., 2023) investigate interactive co-design with conversational LLMs for hardware description generation, while ChipNeMo (Liu et al., 2023) introduces domain-adapted LLMs for industrial chip tasks including EDA script generation and bug analysis. Unlike these works, which primarily focus on language-to-code translation or domain adaptation, our approach leverages MLLMs for spatially grounded reasoning over fine-grained layout features such as routing demand and macro placement. Additionally, we explicitly model design tradeoffs through learned interpretable preferences, extending the role of LLMs from passive code generators to interactive design assistants capable of justification and feedback.

2.2 Congestion Map Prediction

ClusterNet (Min et al., 2023) focuses on predicting routing congestion caused by netlist topology using netlist clustering and Graph Neural Networks (GNNs). By employing the Leiden algorithm to generate cohesive clusters and developing a GNN-based model to generate cluster embeddings, ClusterNet achieves improved prediction performance and congestion optimization without requiring placement and routing (P&R) tools. CircuitNet (Chai et al., 2023; Jiang et al., 2023) provides an open-source dataset for machine learning applications in VLSI CAD, facilitating the development and benchmarking of ML models for tasks like congestion prediction and design rule check (DRC) violation prediction. The dataset includes diverse samples collected from various chip designs and supports comprehensive data, including routability, timing, and power metrics. CircuitFormer (Zou et al., 2023) introduces a novel approach by treating circuit components as point clouds and utilizing Transformer-based point cloud perception methods for feature extraction. This method enables direct feature extraction from raw data without any preprocessing, allows for end-to-end training, and achieves state-of-the-art performance in congestion prediction tasks on both the CircuitNet

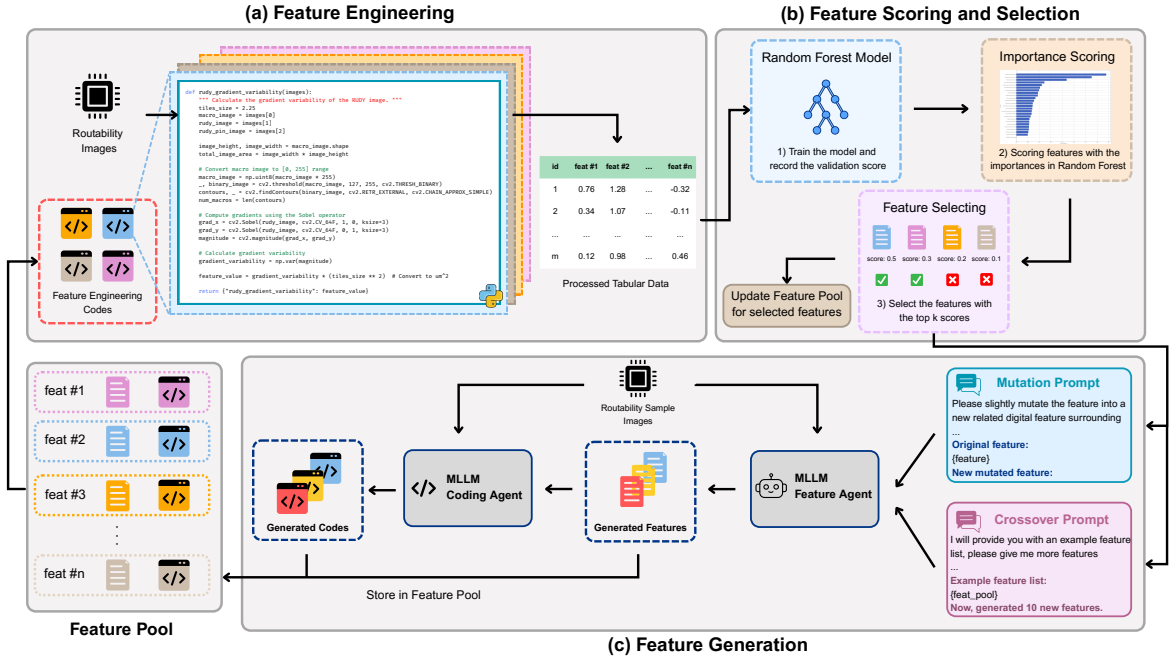


Figure 1: Overview of the Genetic Instruct framework. (a) Spatial metrics are extracted from routability images using domain-specific rules. (b) A Random Forest ranks and selects key features. (c) MLLM agents iteratively generate and mutate features via code, enriching the feature pool to improve downstream performance.

and ISPD2015 datasets, as well as in DRC violation prediction tasks on the CircuitNet dataset. MPGD (Yang et al., 2024) proposes Mini-Pixel Batch Gradient Descent, a plug-and-play optimization algorithm that focuses on the most informative entries in the congestion map. By integrating this method into predictive AI models for physical design flows, MPGD enhances congestion prediction accuracy and efficiency, demonstrating its effectiveness in optimizing chip design processes.

3 Methodology

Our proposed MLLM assistant consists of two core components: (1) automated feature generation and engineering, and (2) interpretable preference modeling for generating actionable design suggestions.

3.1 Automated Feature Generation and Engineering

Figure 1 illustrates our pipeline for automated feature generation and engineering. The approach integrates domain-driven initial features, MLLM-guided mutation, and a feedback loop to iteratively expand and refine a pool of predictive and interpretable features.

3.1.1 Initial Feature Types

We begin with two categories of raw features for routing congestion prediction:

Image-based layout features: *Macro Region* encodes macro and cell placement for global layout context. *RUDY (Routing Demand)* highlights routing density hotspots. *RUDY Pin Images* capture local pin density patterns affecting routing complexity.

Text-based configuration features: Logs from EDA tools include placement constraints, blockages, net priorities, and timing directives. These symbolic features complement spatial representations with interpretable design signals.

3.1.2 Genetic Instruct Feature Generation

We employ a Genetic-Instruct framework (Majumdar et al., 2024) to iteratively evolve and expand feature sets using MLLMs. The process consists of initialization, evaluation, variation, and deduplication steps:

Initialization: The feature pool is seeded with handcrafted features that map visually and semantically to congestion causes. Each feature includes a human-readable name and description. MLLMs are prompted to generate extraction code, producing tabular data for training. Congestion severity is labeled using the average congestion value in the 20 most-congested grid cells of each sample.

Evaluation and Selection: A Random Forest model is trained on the extracted features and labels. The model's feature importance scores act as the fitness function, guiding the pruning of low-impact features. We retain the top k=20 candidates for the next generation.

Crossover and Mutation: New features are created by prompting MLLMs with few-shot examples drawn from high-ranking features (crossover), or by mutating individual features with probabilities based on a rank-sensitive version of the Ebbinghaus forgetting curve. Specifically, mutation probability is defined as: $M = e^{-\frac{r}{N}}$, where r is the feature's rank and N is the total number of features. Higher-ranked features have greater mutation probability, encouraging diversity while maintaining quality.

Deduplication: To avoid redundancy, newly proposed features are compared against the existing pool using a Deduplicator-MLLM. It evaluates semantic similarity in descriptions and removes duplicates with reasoned justifications.

This process iteratively refines the feature space, producing a diverse, interpretable, and model-relevant set of features for downstream congestion prediction. Some generated feature example is shown in Figure 2 and the complete list of generated feature pool is in Appendix F.

Feature Pool Examples

{`macro_compactness_index`: "a measure of how closely packed the macros are, potentially affecting routing paths and congestion",
`clustered_macro_distance_std`: "the standard deviation of distances between clustered groups of macros",
`rudy_pin_clustering_coefficient`: "a measure of how many rudy pins cluster together relative to the total number of rudy pins",
`macro_density_gradient`: "the change in macro density across the layout, impacting local congestion", ...}

Figure 2: Feature Pool Examples

3.1.3 Automated Feature Engineering

We employ Multimodal Large Language Models (MLLMs) to automate feature transformation by generating code snippets tailored to the characteristics of input features and target objectives. The generated code is then executed in a controlled environment, where inputs, outputs, and errors are systematically handled to ensure robustness. This automation streamlines the feature engineering process, improving both efficiency and scalability in modeling routing congestion.

3.2 Interpretable Preferences to Design Suggestions

To bridge the gap between predictive modeling and actionable design insights, we develop a methodology that directly uses the features discovered in Phase 1 as interpretable preference indicators, enabling their translation into concrete and actionable design suggestions.

3.2.1 Interpretable Preferences and Model Architecture

Inspired by the ArmoRM framework (Wang et al., 2024), we formulate design evaluation as a multi-objective task, addressing both scalar congestion prediction and spatial congestion map generation. Central to our approach is the reuse of the feature pool generated in Phase 1 via genetic instruction prompting. These features serve as interpretable preference indicators—each representing a physically

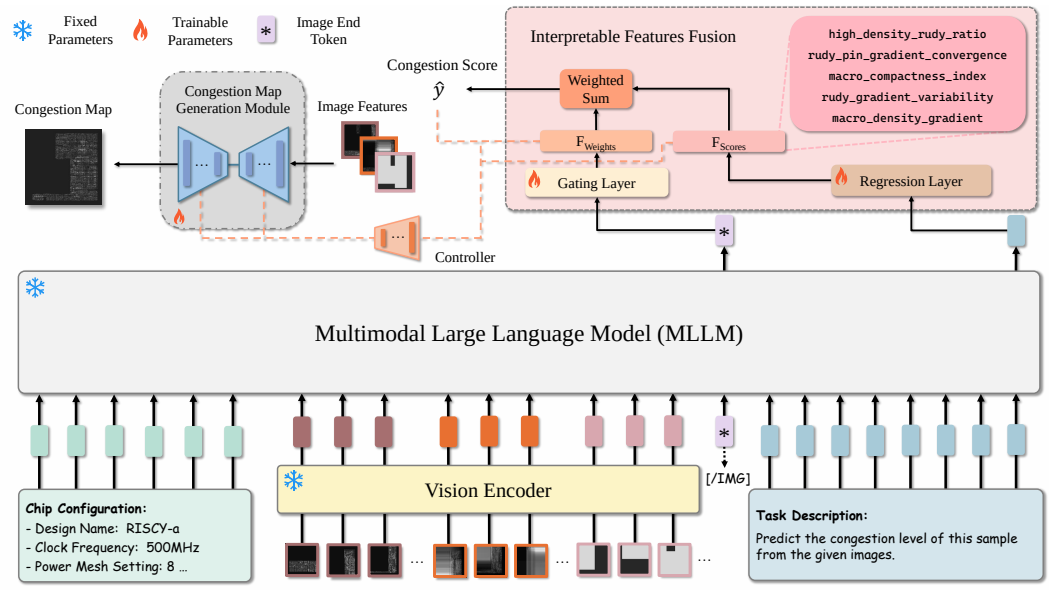


Figure 3: Overview of our interpretable preference modeling framework. The model leverages interpretable features identified in Phase 1 as preference signals to guide congestion prediction and design suggestion generation. A multimodal architecture integrates layout images and task prompts, predicts feature values, learns objective weights via a gating mechanism, and generates both scalar congestion scores and spatial congestion maps. This enables transparent and actionable feedback for physical design refinement.

meaningful and actionable design attribute such as pin clustering, macro placement density, or region-based congestion asymmetry.

Our model adopts a multimodal LLM backbone based on MiniCPM (Yao et al., 2024). Design configuration images are tokenized as vision inputs, while the task descriptions are encoded as text tokens. We then apply a regression layer trained on the last tokens to predict interpretable feature values, and a gating layer trained on vision tokens to learn objective weights based on layout context. The gating layer dynamically assigns importance to each interpretable feature dimension, effectively scalarizing the multi-objective signal into a preference-aware score. This architecture enables both granular score decomposition and end-to-end learning across objectives. The model not only predicts congestion values but also generates pixel-level congestion maps, providing actionable, interpretable guidance to support iterative physical design refinement.

In the congestion map generation stage, a conditional U-Net architecture is employed for the image-to-image translation task, where the input consists of image-based layout features and the output is the predicted congestion map. To incorporate the interpretable features as conditional information, we explore two representation methods: numerical vectors and text-based descriptions. Through the ablation studies in Appendix D, we find that using interpretable features with gating weights in a text-based representation achieves better performance, demonstrating their effectiveness in providing semantic information to guide the generation of congestion maps.

3.2.2 Design Suggestions

Using the interpretable preference signals, we generate targeted design suggestions focused on improving the most critical objectives. Since the model reasons over features that are both informative and physically grounded (e.g., pin density imbalance, macro boundary interactions), its suggestions are inherently actionable. For example, in cases where congestion is identified as a dominant issue, the model might recommend reducing cluster density in specific regions or modifying routing constraints. Each suggestion is directly traceable to feature-level insights, ensuring that design decisions are not only effective but also well-justified.

4 Experiment

Details of the experimental setup and hyperparameters are provided in Appendix E.

4.1 Dataset

The CircuitNet dataset (Jiang et al., 2023; Chai et al., 2023) serves as the foundation for this study. It is a large-scale, open-source benchmark comprising over 10,000 synthesized chip designs—including CPUs, GPUs, and AI accelerators—fabricated using the 14nm FinFET process. It provides two types of features: image-like layout representations and text-based features that extracted from configuration log. Details on preprocessing and data augmentation are included in Appendix E.

4.2 Baselines

We compare our method against three state-of-the-art congestion prediction models. **GPDL** (Chai et al., 2023) is a CNN-based architecture that predicts congestion maps from layout images using spatial convolutions to model local routing patterns. **CircuitFormer** (Zou et al., 2023) is a Transformer-based model operating on point cloud data, learning spatial relationships from raw layout coordinates. **MPGD** (Yang et al., 2024) introduces Mini-Pixel Batch Gradient Descent that targets the most informative regions of the congestion map.

4.3 Evaluation Metrics

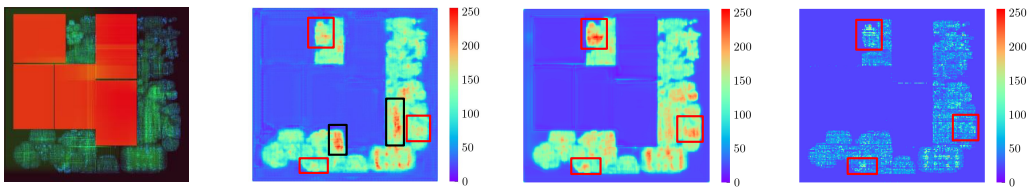
Each model is trained to output a pixel-level prediction of congestion, which is then compared against the ground-truth congestion heatmap. We adopt standard metrics used in prior work to evaluate congestion prediction quality. These include SSIM for perceptual similarity, NRMSE and PeakNRMSE for error measurement, and PLCC, SRCC, and KRCC for correlation analysis. A detailed explanation and formal definitions of each metric are provided in Appendix B.

4.4 Experiment Results

Table 1 shows that our model consistently outperforms baselines in SSIM, NRMSE, and peak error metrics, validating the benefit of interpretable preference modeling and multi-objective supervision. Our qualitative results in Figure 4 further show that the predictions from our model more accurately capture fine-grained congestion boundaries compared to the baselines.

Table 1: Congestion prediction performance on CircuitNet across pixel-based and correlation-based metrics.

Method	**SSIM** ↑	**NRMSE** ↓	**Peak NRMSE** ↓					**Correlation Metrics**		
			0.5%	1%	2%	5%	avg	PLCC ↑	SRCC ↑	KRCC ↑
GPDL (Chai et al., 2023)	0.773	0.047	0.441	0.323	0.236	0.155	0.289	0.5032	0.5143	0.3787
MPGD (Yang et al., 2024)	0.787	0.046	0.271	0.217	0.171	0.121	0.195	–	–	–
CircuitGNN	–	–	–	–	–	–	–	0.3287	0.4483	0.3688
CircuitFormer (Zou et al., 2023)	–	–	–	–	–	–	–	0.6374	**0.5282**	0.3935
Ours	**0.807**	**0.045**	**0.263**	**0.205**	**0.169**	**0.118**	**0.189**	**0.6452**	0.5233	**0.3941**



(a) Input Image Features (b) GPDL (c) MLLMA (Ours) (d) Ground-truth

Figure 4: Qualitative comparison of congestion prediction heatmaps between the GPDL baseline and our proposed multimodal model, MLLMA. Red bounding boxes highlight congestion hotspots where MLLMA more accurately captures the spatial patterns and severity of congestion compared to the ground truth. Black bounding boxes indicate regions where GPDL fails to localize congestion accurately.

Table 2: Ablation Studies.

(a) Ablation on feature generation pipeline.

Pipeline Variant	zero-riscy-a			zero-riscy-b		
	PLCC ↑	SRCC ↑	KRCC ↑	PLCC ↑	SRCC ↑	KRCC ↑
Crossover-Only	0.630	0.674	0.492	0.589	**0.686**	**0.493**
Mutation-Only	0.353	0.485	0.348	0.469	0.462	0.314
Genetic Instruct	**0.705**	**0.786**	**0.595**	**0.599**	0.591	0.412

(b) Ablation on interpretable features.

Feature Source Modality	SSIM ↑	NRMSE ↓
Image-Based	**0.791**	0.047
Log-Based	0.764	0.052
Mixed (Image + Log)	0.789	**0.046**

4.5 Ablation Study

To understand which components contribute most to our model's effectiveness, we conduct a two-phase ablation study.

Phase 1: Ablation Study on Feature Generation Pipeline This phase evaluates the necessity and effectiveness of our proposed **Genetic Instruct prompting strategy** for MLLMs. We compare it against two variants: a crossover-only version that recombines prompt elements and a mutation-only version that introduces random prompt variations. Our Genetic Instruct method integrates both crossover and mutation with evolutionary selection, and achieves significantly better feature quality for downstream congestion prediction. As shown in Table 5a, the Genetic Instruct strategy leads to substantial performance gains, demonstrating that careful prompt optimization is crucial when leveraging MLLMs for physical design tasks.

Phase 2: Ablation Study on Interpretable Preference Pipeline This phase investigates which components of our interpretable preference modeling pipeline are most critical to performance. We ablate four key aspects: the input modalities, the loss functions, the congestion map generation modules, and the interpretable preference features used for gating mechanism. Each removal leads to varying degrees of performance degradation, highlighting the importance of both structural and semantic components in learning meaningful preferences. A subset of the ablation results is summarized in Table 2, with additional details provided in Appendix D. These findings confirm that the learned preference signals are not only interpretable but also integral to model performance. Notably, removing structured inputs or disabling the gating mechanism leads to substantial degradation, underscoring the critical role of each component in modeling contextual and actionable design preferences.

5 Analysis

5.1 Feature Attribution Analysis

Each design feature is associated with a scalar gating value representing its relative importance or preference weight. To interpret the sign of these values: These interpretations allow us to understand how each feature drives model predictions in different design contexts. We observe that low-congestion samples tend to exhibit high values on the `macro_rudy_boundary_interaction_index` feature, as illustrated in Table 3. This trend is consistent across multiple instances, with a negative Pearson correlation of -0.72 between this feature and the congestion score.

Similarly, features such as `rudy_pin_clustering_coefficient` and `rudy_pin_compaction_ratio` tend to have high values in high-congestion samples, showing a positive correlation. This suggests that pin-related density features are dominant contributors to congestion under high-load scenarios.

To further support interpretability, we conduct a detailed breakdown analysis covering the value distribution, gating value, and associated congestion level for every major design feature. This comprehensive analysis is included in Appendix C and Figure 7, where we also include additional observations and insights derived from these patterns.

5.2 Design Suggestion Guidance Leads To Lower Congestion

To evaluate the practical utility of the model's interpretable outputs, we conduct a case study that demonstrates how applying model-guided feature adjustments can lead to a substantial reduction

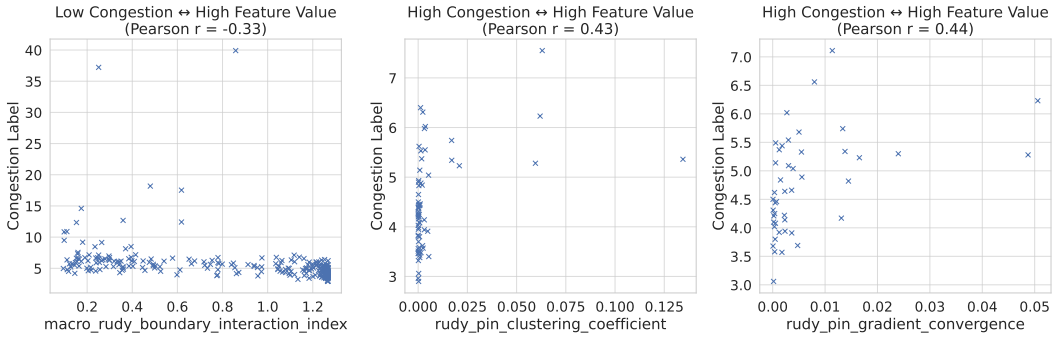
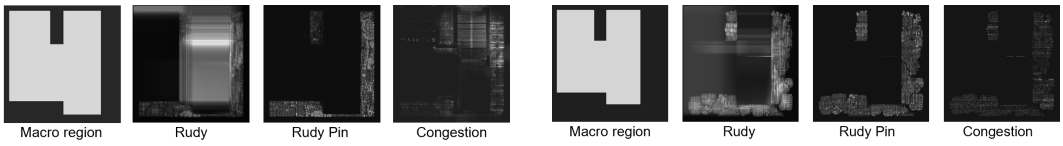


Table 3: Scatter plots showing the relationship between key layout features and congestion labels. (Left) `macro_rudy_boundary_interaction_index` is negatively correlated with congestion ($r = -0.33$), indicating that low-congestion samples tend to exhibit higher values in this macro-level feature. (Middle) `rudy_pin_clustering_coefficient` shows a moderate positive correlation with congestion ($r = 0.43$), suggesting that dense pin clusters contribute to congestion under high-load conditions. (Right) `rudy_pin_gradient_convergence` similarly displays a positive trend ($r = 0.44$), further highlighting the influence of pin-level spatial patterns on congestion outcomes.

in routing congestion. Specifically, we use the model's top-attributed features to inform actionable design changes and compare outcomes before and after applying these suggestions. Figure 5 presents one representative example. The left side shows a high-congestion design sample, along with the top-5 features contributing most to the model's prediction. Based on these attributions, we systematically adjust feature values—particularly those with strong positive influence on congestion—toward more favorable levels. The adjusted design, shown on the right, exhibits a notable improvement.

This case highlights the effectiveness of the model's interpretable reasoning. The ability to trace predictions back to key design features enables not just post hoc understanding, but also actionable guidance. By following these learned preferences, designers can reduce congestion and iteratively improve design quality in practice.



(a) Before adjustment.

(b) After adjustment.

Figure 5: By following the top model-attributed features and adjusting actionable design parameters, the resulting design exhibits significantly lower predicted and actual congestion.

Feature Attribution and Congestion Change Summary:

Table 4: Top features before/after adjustment and their impact on congestion.

Feature	Before Value	After Value	Attribution (Before)	Congestion Impact
rudy_pin_gradient_convergence	0.3351	0.1202	0.1551 → 0.0001	High
rudy_pin_clustering_coefficient	0.5913	0.3606	0.1409 → 0.0002	↓
rudy_pin_compaction_ratio	0.4273	0.3035	0.1216 → 0.0001	↓
high_density_rudy_pin_ratio	0.3138	0.4867	0.0638 → 0.0003	Mixed
demarcated_macro_proximity_index	0.3765	-0.0030	0.0539 → 0.0000	↓
macro_rudy_boundary_interaction_index	0.0849	0.0721	0.0321 → 1.2655	High post-adjustment role

This case illustrates how interpreting the gating values and using them to guide feature adjustment can lead to measurable congestion reduction, demonstrating the interpretability and actionability of our model's learned preferences.

We compute the consistency rate across a large number of matched sample pairs. A high agreement between the sign of the gating value and the direction of the label difference indicates that the learned

preferences are interpretable and aligned with the optimization objective. Additionally, we collaborate with experienced chip designers and researchers to manually review selected cases, further validating that the gating values reflect meaningful and actionable design insights.

5.3 Design Suggestion Qualitative Examples

We complement our quantitative and case-based evaluation with a set of qualitative examples that further illustrate how the model adapts its reasoning to context-specific designs. In Figure 6, we compare two designs—Design 8277 and Design 8243—exhibiting congestion scores of 40.21 and 20.04, respectively. Despite having similar raw feature values in some cases, the relative importance assigned to each feature varies dramatically between the two layouts. Design 8277 shows elevated importance on multiple pin-related density features, suggesting that the layout suffers from excessive pin compaction and clustering. In contrast, Design 8243 is dominated by a single macro-level feature—`macro_rudy_boundary_interaction_index`—highlighting the positive influence of well-structured macro placement. These examples demonstrate that our model not only identifies which features matter most, but also how their importance is modulated by spatial context. This supports our broader goal of providing interpretable, design-aware guidance that generalizes across different physical design scenarios.

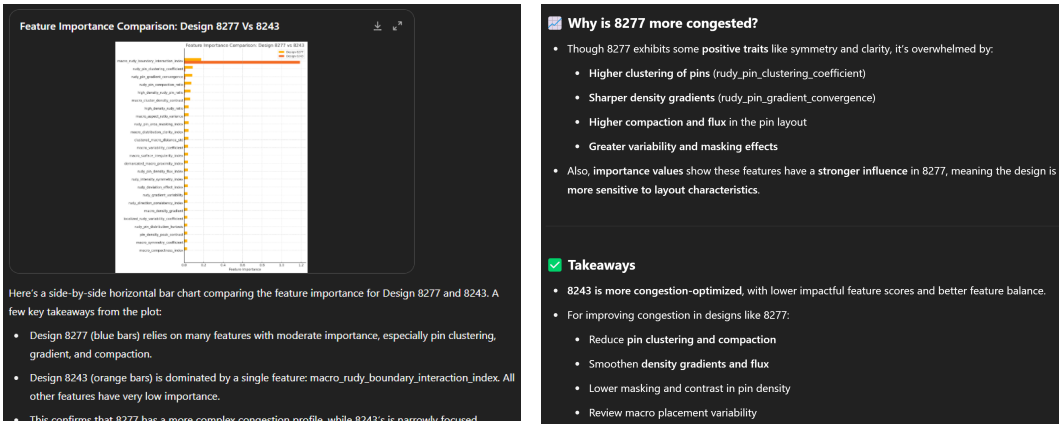


Figure 6: Qualitative interpretation comparing Design 8277 and 8243. Left: Feature importance bar chart. Right: Root causes and actionable takeaways.

6 Limitation

While our framework shows strong performance, several limitations remain. First, the current evaluation is limited to a RISC-V chip architectures and the CircuitNet dataset. Broader validation across different architectures and more diverse commercial designs is necessary for practical deployment. Second, our focus is confined to congestion prediction; extending the approach to other critical physical design objectives such as power, performance, area (PPA), IR drop, and timing closure remains an open direction.

7 Conclusion

We propose a multimodal LLM assistant for chip physical design that predicts congestion and provides interpretable guidance. By combining genetic prompting and preference-based explanation, our method delivers actionable design suggestions aligned with expert intuition. Experiments show state-of-the-art results on CircuitNet, and case studies confirm practical utility. Future work will expand to broader tasks and interactive, real-time, instruction-like design support.

References

J. Blocklove, S. Garg, R. Karri, and H. Pearce. Chip-chat: Challenges and opportunities in conversational hardware design. In *2023 ACM/IEEE 5th Workshop on Machine Learning for CAD (MLCAD)*, pages 1–6. IEEE, 2023.

Z. Chai, Y. Zhao, W. Liu, Y. Lin, R. Wang, and R. Huang. Circuitnet: An open-source dataset for machine learning in vlsi cad applications with improved domain-specific evaluation metric and learning strategies. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(12):5034–5047, 2023.

T. Gong, C. Lyu, S. Zhang, Y. Wang, M. Zheng, Q. Zhao, K. Liu, W. Zhang, P. Luo, and K. Chen. Multimodal-gpt: A vision and language model for dialogue with humans. *arXiv preprint arXiv:2305.04790*, 2023.

X. Jiang, Y. Zhao, Y. Lin, R. Wang, R. Huang, et al. Circuitnet 2.0: An advanced dataset for promoting machine learning innovations in realistic chip design environment. In *The Twelfth International Conference on Learning Representations*, 2023.

M. Liu, T.-D. Ene, R. Kirby, C. Cheng, N. Pinckney, R. Liang, J. Alben, H. Anand, S. Banerjee, I. Bayraktaroglu, et al. Chipnemo: Domain-adapted llms for chip design. *arXiv preprint arXiv:2311.00176*, 2023.

S. Majumdar, V. Noroozi, S. Narenthiran, A. Ficek, J. Balam, and B. Ginsburg. Genetic instruct: Scaling up synthetic generation of coding instructions for large language models. *arXiv preprint arXiv:2407.21077*, 2024.

K. Min, S. Kwon, S.-Y. Lee, D. Kim, S. Park, and S. Kang. Clusternet: Routing congestion prediction and optimization using netlist clustering and graph neural networks. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2023.

H. Wang, W. Xiong, T. Xie, H. Zhao, and T. Zhang. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10582–10592, 2024.

H. Yang, A. Agnesina, and H. Ren. Optimizing predictive ai in physical design flows with mini pixel batch gradient descent. In *Proceedings of the 2024 ACM/IEEE International Symposium on Machine Learning for CAD*, pages 1–7, 2024.

Z. Yang, L. Li, J. Wang, K. Lin, E. Azarnasab, F. Ahmed, Z. Liu, C. Liu, M. Zeng, and L. Wang. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*, 2023.

Y. Yao, T. Yu, A. Zhang, C. Wang, J. Cui, H. Zhu, T. Cai, H. Li, W. Zhao, Z. He, et al. Minicpm-v: A gpt-4v level mllm on your phone. *arXiv preprint arXiv:2408.01800*, 2024.

M. Zhu, Y. Zhu, X. Liu, N. Liu, Z. Xu, C. Shen, Y. Peng, Z. Ou, F. Feng, and J. Tang. Mipha: A comprehensive overhaul of multimodal assistant with small language models. *arXiv preprint arXiv:2403.06199*, 2024.

J. Zou, X. Wang, J. Guo, W. Liu, Q. Zhang, and C. Huang. Circuit as set of points. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

A Automated Feature Generation and Feature Engineering Algorithm

Algorithm 1 Automated Feature Generation and Feature Engineering

Input: N: Number of automated iterations
k: Maximum size of feature pool
$\mathcal{F}_{init}$: Initial set of hand-crafted features
$\mathcal{D}$: Raw images of training data
$\mathcal{L}$: Congestion levels of training data
I_s: Sample images for the inputs of MLLMs
P_m: Probability of selecting mutation operation
P_{op}: Probability distribution over the operations {Mutation: P_m, Cross-Over: $1 - P_m$}
Output: $\mathcal{F}_{total}$: Final feature pool, $\mathcal{E}_{total}$: Final feature extraction codes

Initialize $\mathcal{F}_{pool} \leftarrow \mathcal{F}_{init}$;
Initialize $\mathcal{E}_{code} \leftarrow CoderMLLM(I_s, \mathcal{F}_{pool})$;
for $r \leftarrow 1$ ***to*** N **do**
    Initialize $T \leftarrow \{\mathcal{L}\}$;
    for each feature extractor ϵ_i in $\mathcal{E}_{code}$ **do**
        $T \leftarrow T \cup \epsilon_i(\mathcal{D})$;
    end for
    $R \leftarrow$ Train a Random Forest Regressor with tabular features T;
    $f \leftarrow$ Return the feature importance function in trained Random Forest Regressor R;
    $\mathcal{F}_{pool}, \mathcal{E}_{code} \leftarrow FeatureSelector(\mathcal{F}_{pool}, \mathcal{E}_{code}, f, \min(k, len(\mathcal{F}_{pool})))$;
    $S_{op} \leftarrow$ Choose an operation from P_{op};
    if $S_{op} =$ Mutation **then**
        $M \leftarrow RankSensitveFormula(\mathcal{F}_{pool}, f)$;
        $\mathcal{F}_{parents} \leftarrow$ Select parent features $\subset \mathcal{F}_{pool}$ from mutation probabilities M;
        $\mathcal{F}_{new} \leftarrow MutatorMLLM(I_s, \mathcal{F}_{parents})$;
    else
        $\mathcal{F}_{new} \leftarrow CrossoverMLLM(I_s, \mathcal{F}_{pool})$;
    end if
    $\mathcal{F}_{new} \leftarrow DeduplicatorMLLM(I_s, \mathcal{F}_{pool}, \mathcal{F}_{new})$;
    $\mathcal{E}_{new} \leftarrow CoderMLLM(I_s, \mathcal{F}_{new})$;
    $\mathcal{F}_{pool} \leftarrow \mathcal{F}_{pool} \cup \mathcal{F}_{new}$;
    $\mathcal{E}_{code} \leftarrow \mathcal{E}_{code} \cup \mathcal{E}_{new}$;
end for
$\mathcal{F}_{total}, \mathcal{E}_{total} \leftarrow \mathcal{F}_{pool}, \mathcal{E}_{code}$;

B Evaluation Metric Definition

Definition 1 (SSIM)**.** *The **Structural Similarity Index Measure (SSIM)** evaluates the perceptual similarity between two images based on luminance, contrast, and structural information. It is defined as:*

$$\mathit{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)},$$

where μ denotes the mean, σ^2 the variance, and σ_{xy} the covariance between prediction x and ground truth y. Constants c_1 and c_2 stabilize the denominator.

Definition 2 (MSE, NRMSE, PeakNRMSE)**.** *The **Mean Squared Error (MSE)** is defined as:*

$$\mathit{MSE}(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2.$$

*The **Normalized Root Mean Square Error (NRMSE)** scales the RMSE by the value range of the ground truth:*

$$\mathit{NRMSE}(x, y) = \frac{1}{\max(y) - \min(y)}\sqrt{\mathit{MSE}(x, y)}.$$

*The **PeakNRMSE** evaluates NRMSE restricted to the top-k largest entries of both x and y, emphasizing high-congestion regions that are most critical in physical design.*

Definition 3 (Correlation-Based Metrics: PLCC, SRCC, KRCC)**.** *These metrics evaluate statistical and rank-order correlation between prediction x and ground truth y:*

- ***PLCC (Pearson Linear Correlation Coefficient)** measures linear correlation:*

$$\mathit{PLCC}(x, y) = \frac{\mathit{Cov}(x, y)}{\sigma_x\sigma_y},$$

  where $\mathit{Cov}(x, y)$ is the covariance and σ is the standard deviation.

- ***SRCC (Spearman Rank-Order Correlation Coefficient)** computes PLCC on the rank-transformed values of x and y, capturing monotonic relationships.*

- ***KRCC (Kendall Rank Correlation Coefficient)** compares the number of concordant and discordant pairs:*

$$\mathit{KRCC}(x, y) = \frac{n_c - n_d}{\binom{n}{2}},$$

  where n_c and n_d are the counts of concordant and discordant pairs among all $\binom{n}{2}$ possible pairs.

C Feature Attribution Breakdown Across All Features

To further examine the interpretability of the learned gating values, we analyze all 25 features used by the model. Figure 8 presents a comprehensive visualization, where each subplot corresponds to one feature.

For each subplot:

- The x-axis denotes the raw feature value.
- The y-axis shows the corresponding gating weight assigned by the model.
- Each point is colored by its associated congestion score, with darker colors indicating higher congestion.
- The left panel shows results from model *predictions*, and the right panel shows alignment with *ground-truth labels*.

Observation 1: Prediction-Label Alignment. Across most features, the gating weights show strong alignment between predicted and true congestion labels. This validates that the model is not only fitting the training objective, but that the learned preferences generalize to meaningful physical properties in actual designs.

Observation 2: Unique Negative Attribution. Among all 25 features, only one — `macro_rudy_boundary_interaction_index` — consistently receives a negative gating weight. This aligns with earlier findings in Section 5.1, where this feature ranked highly (inversely) correlated with congestion. The model correctly assigns it a down-weighted preference, consistent with human-understandable design heuristics.

Observation 3: Gating Weights Capture Context Beyond Feature Values. Notably, the spread of gating weights is often broader than that of feature values. In some features, the same value results in different gating weights across samples. This suggests that the model modulates importance dynamically based on context — likely derived from the input image, configuration tokens, or pretraining of the MLLM backbone. This further supports the idea that gating weights encode richer, more contextualized preference signals than raw features alone.

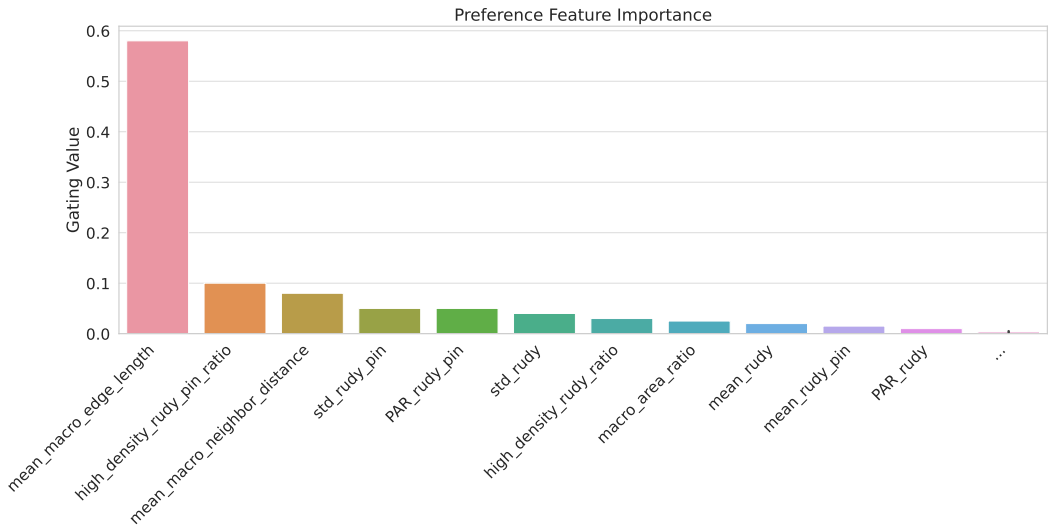
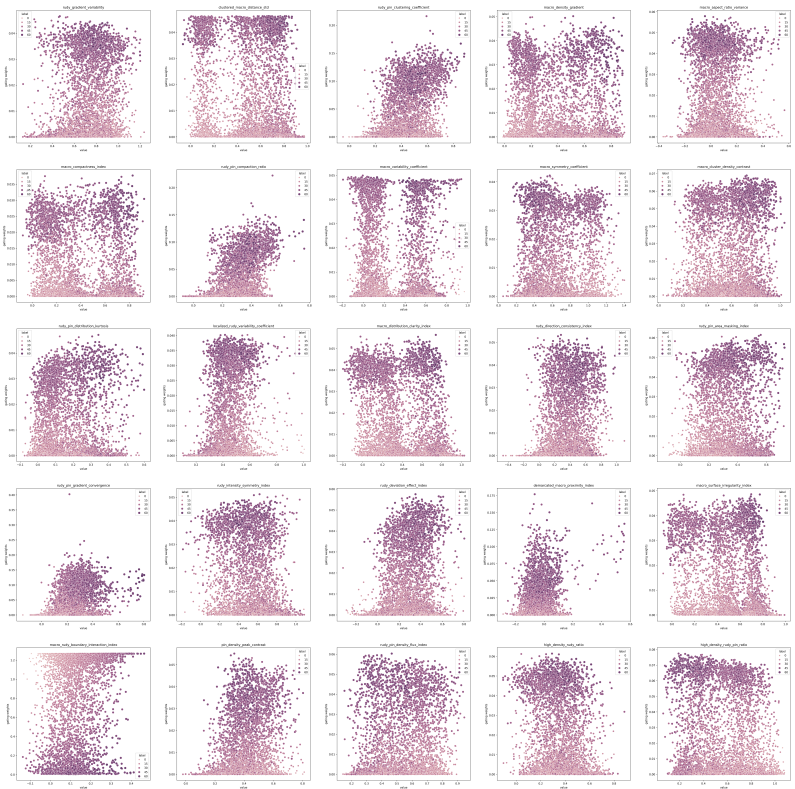
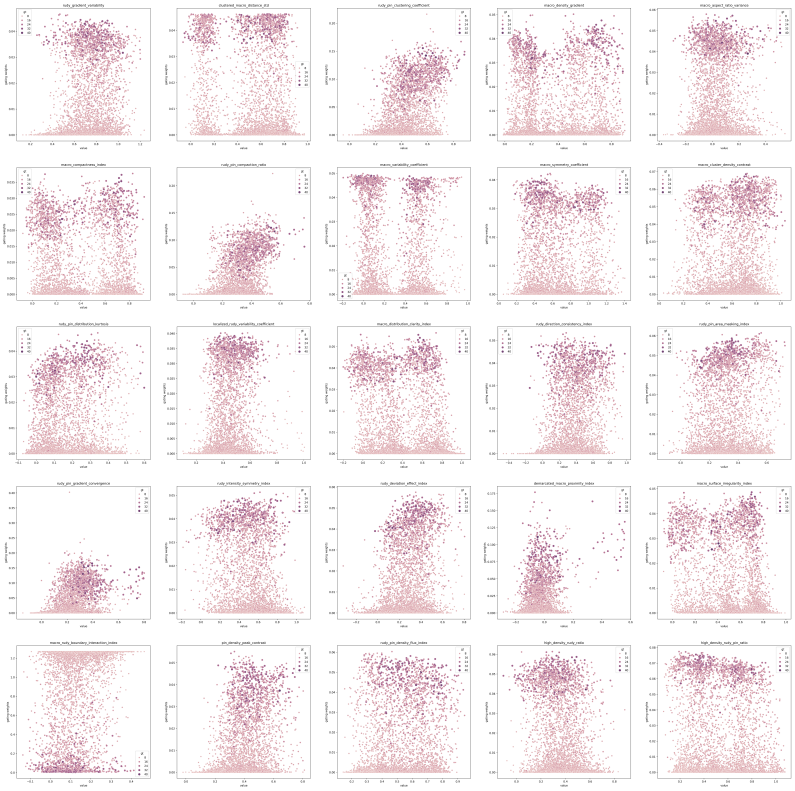


Figure 7: Bar plot showing the relative importance based on gating values from the preference features used for congestion prediction. The most influential feature is `mean_macro_edge_length`, significantly outweighing others. This suggests that macro-level spatial layout characteristics play a dominant role in determining congestion levels, followed by pin-density features such as `high_density_rudy_pin_ratio` and `mean_macro_neighbor_distance`.



(a) Gating values vs. feature values, colored by predicted congestion.



(b) Gating values vs. feature values, colored by ground-truth congestion.

Figure 8: Feature attribution breakdown across all 25 features. Each point represents a sample, colored by congestion score.

D Detailed Ablation Study

Table 5: Ablation Studies.

(a) Ablation on feature generation pipeline methods.

Pipeline Variant	zero-riscy-a			zero-riscy-b		
	PLCC ↑	SRCC ↑	KRCC ↑	PLCC ↑	SRCC ↑	KRCC ↑
Crossover-Only	0.630	0.674	0.492	0.589	**0.686**	**0.493**
Mutation-Only	0.353	0.485	0.348	0.469	0.462	0.314
Genetic Instruct	**0.705**	**0.786**	**0.595**	**0.599**	0.591	0.412

(b) Ablation on interpretable features.

Feature Source Modality	SSIM ↑	NRMSE ↓
Image-Based	**0.791**	0.047
Log-Based	0.764	0.052
Mixed (Image + Log)	0.789	**0.046**

(c) Ablation on conditional states.

Conditional Modality	SSIM ↑	NRMSE ↓
Features and Gating weights		
- numerical format (MLP)	0.787	0.047
- text format (T5 Encoder)	**0.791**	0.047

(d) Ablation on loss functions.

Loss Variant	SSIM ↑	NRMSE ↓
U-Net with text conditioning		
- MSE	**0.791**	**0.047**
- MPGD	0.789	**0.047**
- MS-SSIM + L1	0.762	0.079

(e) Ablation on congestion map generation modules.

Input Variant	SSIM ↑	NRMSE ↓
U-Net	0.791	0.047
U-Net + CNN Highway	0.792	0.048
U-Net + MLLM (trainable)	**0.807**	**0.045**

E Experiment Setup

E.1 Training Setup

All models are trained using the AdamW optimizer with mixed-precision (FP16) training enabled. Most experiments are conducted on 32 NVIDIA A100 GPUs, while some smaller-scale experiments and ablations are run on NVIDIA RTX A6000 GPUs. We train for 50 epochs with early stopping based on validation loss. The learning rate follows a linear warm-up over the first 500 steps and a cosine decay schedule thereafter. The dataset is split into 80% training, 10% validation, and 10% test. All reported results are based on the held-out test set. To ensure robustness, each experiment is repeated with three random seeds.

E.2 Hyperparameters

We set the initial learning rate to 1e-4 with weight decay of 0.01. The optimizer uses $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The batch size is set to 4. We adopt MiniCPM-V-2_6 (Yao et al., 2024) as the backbone of our MLLM, where the visual encoder follows a SigLIP-like architecture. For the regression layer encoder, we use a linear layer appended to the backbone; for the gating layer, we use a ReLU MLP of 3 hidden layers with 1024 hidden units. The conditioning text descriptions for U-Net are encoded with a frozen T5 encoder model unless otherwise stated. Dropout of 0.1 is applied to all modality-specific encoders.

E.3 Dataset Preprocessing

We preprocess the CircuitNet dataset to construct aligned multimodal representations, including: (1) layout-based raster images (e.g., Macro Region, RUDY, and RUDY pin maps), (2) normalized tabular features extracted through code-driven analysis, and (3) optional configuration text describing design modules and target specifications. We partition the dataset into 80% training, 10% validation, and 10% test. All results reported are on held-out test samples. Additional augmentations include rotational invariance for layout images and controlled masking of tabular features to assess robustness.

F Complete List of Generated Feature Pool and Feature Engineering Details

Image-based Feature Pool (Appendix)

Feature Name	Description
rudy_gradient_variability	the variation in gradient changes across the rudy map indicating potential areas of abrupt routing demand shifts
clustered_macro_distance_std	the standard deviation of distances between clustered groups of macros
rudy_pin_clustering_coefficient	a measure of how many rudy pins cluster together relative to the total number of rudy pins
macro_density_gradient	the change in macro density across the layout, impacting local congestion
macro_aspect_ratio_variance	the variance in aspect ratios of macros, indicating potential alignment and spacing issues that may impact congestion
macro_compactness_index	a measure of how closely packed the macros are, potentially affecting routing paths and congestion
rudy_pin_compaction_ratio	the ratio of compacted rudy pin clusters to the total number of rudy pins, indicating areas with high potential routing conflicts
macro_variability_coefficient	a measure of the consistency in macro sizes and shapes relative to each other, potentially affecting congestion balance
macro_cluster_density_contrast	the contrast in density between clustered groups of macros and their surrounding layout areas, indicating potential localized congestion pressure
rudy_pin_distribution_kurtosis	a measure of the peakedness or flatness in the distribution of rudy pins across the layout
localized_rudy_variability_coefficient	variation in RUDY intensity within localized regions, indicating potential micro-level congestion fluctuations
macro_distribution_clarity_index	a measure of how distinct macro distributions are across the layout, indicating clarity in separation
rudy_direction_consistency_index	a measure of the uniformity in the directional flow of RUDY intensity
rudy_pin_area_masking_index	the ratio of the area masked by rudy pin regions relative to the total layout
rudy_pin_gradient_convergence	a measure of how gradients in the rudy pin map converge into specific regions
rudy_deviation_effect_index	the deviation of RUDY intensities from the mean, indicating areas of abnormal routing demand
demarcated_macro_proximity_index	proximity of macros to predefined boundary regions, potentially affecting congestion near edges
macro_surface_irregularity_index	the irregularity in macro surface shapes, which can impact routing paths
macro_rudy_boundary_interaction_index	interaction between macros and high RUDY regions, indicating potential congestion hotspots
pin_density_peak_contrast	the contrast between peak pin density regions and their surroundings
rudy_pin_density_flux_index	the rate of change in rudy pin density across the layout
high_density_rudy_ratio	the ratio of areas with high RUDY intensity to the total layout area
high_density_rudy_pin_ratio	the ratio of areas with high RUDY pin intensity to the total layout area

Figure 9: Complete list of generated features used in our modeling pipeline. These capture spatial, statistical, and gradient-based properties derived from chip layout images.

Configuration and Log-derived Feature Pool

Feature Name	Description
design_name	Design identifier (e.g., RISCY-a), useful for tracking design variants.
number_of_macros	Total number of macros in the design, affecting placement complexity.
clock_frequency	Operating frequency (e.g., 200MHz), indicating timing pressure that may affect placement/routing.
utilization	Target utilization (e.g., 70%), influencing available routing resources.
macro_placement	Macro placement strategy or group count, impacting layout structure.
power_mesh_setting	Power mesh granularity (e.g., 8), which may influence routing congestion and IR-drop safety margins.
filler_insertion	Stage of filler insertion (e.g., after placement), potentially affecting congestion distribution.
initial_placement_efficiency	Efficiency of macro and cell placement before routing begins.
instance_blockages_count	Number of instance blockages, reflecting routing resource obstruction from macro/cell placement.
hard_to_access_pins_ratio	Proportion of pins difficult to route due to placement or geometry constraints.
pin_density_variance_map	Variance in pin density across layout regions, highlighting routing hotspots.
multi_layer_pin_access_variability	Variation in pin accessibility across metal layers, indicating routing complexity.
non_default_routing_rule_usage	Frequency of non-default routing rules (if known from constraints or synthesis config).
crosstalk_sensitive_zones	Areas near critical nets where routing rules must prevent signal interference.

Figure 10: Complete list of configuration- and log-derived features used in our modeling pipeline. These include routing and congestion-related metrics obtained from the global placement and routing stages, as well as features generated by our own pipeline. They collectively capture layout-level difficulty indicators relevant for congestion prediction.

Feature Extraction Code Example: Macro Density Gradient

```
def macro_density_gradient(images):
    tiles_size = 2.25
    macro_image = images[0]
    rudy_image = images[1]
    rudy_pin_image = images[2]

    image_height, image_width = macro_image.shape
    total_image_area = image_width * image_height

    # Convert macro image to binary [0, 255]
    macro_image = np.uint8(macro_image * 255)
    _, binary_image = cv2.threshold(macro_image, 127, 255, cv2.THRESH_BINARY)

    # Find contours to get macro regions
    contours, _ = cv2.findContours(binary_image, cv2.RETR_EXTERNAL, cv2.
        CHAIN_APPROX_SIMPLE)

    # Calculate macro density per region
    macro_density = np.zeros((image_height, image_width))
    for contour in contours:
        mask = np.zeros_like(binary_image)
        cv2.drawContours(mask, [contour], -1, 255, thickness=cv2.FILLED)
        macro_density += mask

    # Gradient of the macro density
    gradient_x = cv2.Sobel(macro_density, cv2.CV_64F, 1, 0, ksize=5)
    gradient_y = cv2.Sobel(macro_density, cv2.CV_64F, 0, 1, ksize=5)

    # Calculate the magnitude of gradients
    gradient_magnitude = cv2.magnitude(gradient_x, gradient_y)

    # Calculate the average gradient magnitude in micrometers
    macro_density_gradient_um = np.sum(gradient_magnitude) / (image_height *
        image_width)
    macro_density_gradient_um *= tiles_size  # Convert to micrometers

    return {"macro_density_gradient": macro_density_gradient_um}
```

Figure 11: Python implementation of the `macro_density_gradient` feature extractor. This function computes the average gradient magnitude of macro cell density across a chip layout image, offering a scalar representation of spatial density variation in micrometers.

Feature Extraction Code Example: High Density Rudy Ratio

```
def high_density_rudy_ratio(images):
    image = images[1]
    total_area = image.shape[0] * image.shape[1]
    mean_rudy = np.mean(image)
    high_density_rudy_ratio = (image > mean_rudy).sum() /  total_area

    return {
        "high_density_rudy_ratio": high_density_rudy_ratio,
    }
```

Figure 12: Python implementation of the `high_density_rudy_ratio` feature extractor. This function computes the average gradient magnitude of macro cell density across a chip layout image, offering a scalar representation of spatial density variation in micrometers.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract clearly reflects the paper's contributions, including the introduction of a multimodal LLM assistant for congestion prediction and design guidance, the use of genetic prompting, and interpretable preference learning. These claims are supported by experiments and analyses in the main text and do not overstate the scope or results.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitation discussed in Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: No theoretical result.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide comprehensive details of the dataset, model architecture, training setup, and hyperparameters necessary to reproduce the main results. The dataset (CircuitNet) is publicly available, and all preprocessing steps, data splits, and evaluation metrics are clearly described. Appendix E includes further implementation details, including model configurations and ablation setups. While the code is not included, all critical information required to replicate our main findings is disclosed in the paper.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The dataset used (CircuitNet) is publicly available. However, the code and reproduction instructions are not yet released at the time of submission. We plan to make both the codebase and detailed instructions publicly available upon publication to support full reproducibility.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes, the paper specifies all necessary details to understand the results. We describe the dataset splits, the optimizer, learning rate schedule, batch size, number of epochs, and other relevant hyperparameters. Additional implementation details, including architecture configurations and ablation settings, are provided in Appendix E.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

Justification: We do not report error bars or statistical significance metrics across multiple random seeds. However, we provide extensive qualitative analysis, including multiple case studies and human manual reviews, to support the consistency and interpretability of our findings.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We specify the types of compute resources used, including GPU models and training configurations. Detailed information on the training environment, including hardware setup and compute requirements, is provided in Appendix E.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This research fully adheres to the NeurIPS Code of Ethics. It does not involve human subjects, sensitive data, or applications with foreseeable societal or environmental harm. All experiments are conducted on publicly available datasets (CircuitNet), with transparent methodology and reproducibility efforts documented in the appendix. The paper also includes ablation studies, case analyses, and human evaluations conducted with appropriate consent and without harm. Potential misuse of the work is minimal, and no discriminatory or unfair bias is introduced.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: The research focuses on methodological improvements for chip design optimization using publicly available data and does not directly involve societal-facing applications. As such, a discussion of societal impact is not applicable in the context of this work.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The work does not involve high-risk assets such as large pretrained language models, generative models, or scraped datasets. It builds on a publicly available dataset (CircuitNet) and applies domain-specific methods for chip design analysis, which pose minimal risk of misuse. Therefore, safeguards for responsible release are not applicable in this context.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All external assets used in this work are properly credited. The dataset used—CircuitNet—is cited appropriately (Jiang et al., 2023; Chai et al., 2023), and we adhere to its open-source license and terms of use. No proprietary or restricted-access data, code, or models were used without permission.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not introduce new assets. All experiments are based on the existing public dataset CircuitNet, and no new datasets, models, or tools are being released as part of this work.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

Justification: The paper includes human evaluations in the form of manual reviews conducted by the authors or domain experts, but it does not involve formal crowdsourcing or studies with human subjects requiring ethical review or participant compensation. Therefore, this question is not applicable.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing or research with human subjects as defined by IRB standards. All manual evaluations were performed by the authors or internal collaborators and do not constitute formal human subject research. Therefore, IRB approval is not applicable.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: This work leverages a multimodal LLM as a core component to automate feature generation and support interpretable feature learning from chip design layouts and metadata. Additionally, the LLM is used to generate human-readable design suggestions based on model insights, enhancing downstream interpretability and usability. These usages go beyond standard applications and are integral to the proposed methodology; they are therefore clearly described in the paper.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.