See it to Place it: Evolving Macro Placements with Vision Language Models

Ikechukwu Uchendu[*,2,3] **Swati Goel**[2] **Karly Hou**[2] **Ebrahim Songhori**[1]

Kuang-Huei Lee[1] **Joe Wenjie Jiang**[1] **Vijay Janapa Reddi**[2] **Vincent Zhuang**[1]

Abstract

We propose using Vision-Language Models (VLMs) for macro placement in chip floorplanning, a complex optimization task that has recently shown promising advancements through machine learning methods. We hypothesize that the spatial reasoning capabilities of VLMs can effectively complement existing learning-based approaches. In this work, we introduce VeoPlace (Visual Evolutionary Optimization Placement), a novel framework that uses a VLM to guide the actions of a base policy by constraining them to subregions of the chip canvas. The VLM proposals are iteratively optimized through an evolutionary search strategy with respect to resulting placement quality. On open-source benchmarks, VeoPlace yields state-of-the-art results on four out of seven benchmark circuits, matching or exceeding the performance of prior learning-only approaches. Our approach opens new possibilities for electronic design automation tools that leverage foundation models to solve complex physical design problems.

1 Introduction

Computer chip floorplanning is a critical step in integrated circuit design, involving strategic arrangement of macros (pre-designed functional blocks like memory arrays, processing units, and I/O interfaces) on the chip canvas. Determining optimal placement is a complex multi-objective problem optimizing performance, power, and area (PPA) while respecting constraints such as routing congestion. Due to the vast combinatorial design space, chip floorplanning requires significant time from human designers and has driven extensive research on automation.

Automating chip floorplanning has been approached through black-box optimization [33], analytical methods [23, 8, 25], and learning-based methods [26, 18, 17, 21]. Despite their successes, existing approaches have a fundamental limitation: because they train policies mapping structured representations to placements *tabula rasa*, they struggle to generalize quickly on unseen blocks. This is exacerbated by limited data in chip floorplanning. Conversely, human designers rapidly iterate using high-level prior knowledge and spatial reasoning, quickly improving designs through trial-and-error, whereas learned models require time-consuming reinforcement learning processes.

For learning-based approaches, optimizing placements on unseen blocks typically requires test-time interaction [26, 18, 17]. We consider this formulation: for an unseen block and fixed budget of B

[*]Work done as a Student Researcher at Google. [1]Google DeepMind [2]Harvard University, Cambridge, MA, USA [3]Kempner Institute for the Study of Natural and Artificial Intelligence, Harvard University, Cambridge, MA, USA Correspondence to: Ikechukwu Uchendu <iuchendu1@gmail.com>

First Exploration in AI Today Workshop at ICML (EXAIT at ICML 2025).

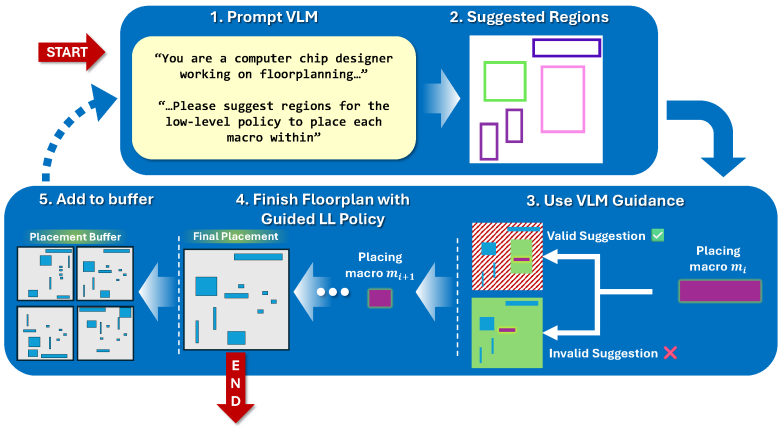
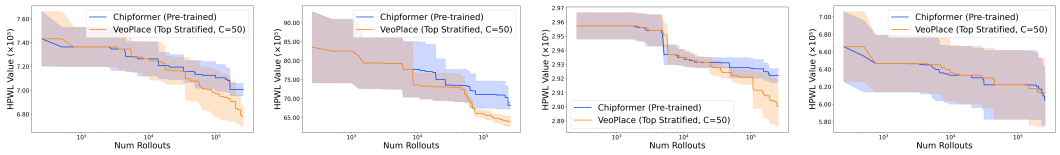


Figure 1: VeoPlace framework overview. The VLM suggests placement regions (1-2) to constrain a low-level policy (3) for macro placement (4). A history buffer that stores the existing population of placements (5) facilitates evolutionary in-context improvement, creating a feedback loop to improve placement quality.



(a) adaptec1 (b) adaptec2 (c) bigblue1 (d) bigblue2

Figure 2: Performance scaling of VeoPlace compared to ChiPFormer (pre-trained) across increasing rollouts. The y-axis shows the HPWL, where lower values indicate better performance. The log-scale x-axis highlights that VeoPlace (orange) achieves better scaling behavior than ChiPFormer (blue), leading to superior floorplans as the number of rollouts increases.

placement attempts, what is the best possible placement? We hypothesize that incorporating spatial intuitions and reasoning about packing, shifting, and grouping can lead to more effective placements and adaptive improvement from previous attempts.

We propose using vision-language models (VLMs) [6, 30, 28] to augment existing learning-based floorplanning models [17]. Our method, VeoPlace (Visual Evolutionary Optimization Placement), uses a hierarchical structure where a high-level VLM planner proposes promising subregions to a low-level placement policy that autoregressively places macros. Crucially, VLM proposals are iteratively refined through an evolutionary process, evolving new proposals from high-performing prior proposals. VeoPlace requires no VLM fine-tuning and uses an independently-trained low-level policy (ChiPFormer) [17]. Our main contributions are:

- **Novel VLM-guided Placement Framework**: We introduce VeoPlace, the first approach to leverage vision-language models for macro placement in chip floorplanning, demonstrating how foundation models can effectively guide specialized placement algorithms through structured spatial reasoning without requiring fine-tuning.
- **Evolutionary Context Selection Strategy**: We develop a novel context selection mechanism that enables VLMs to iteratively improve placement quality by focusing on geometrically similar high-performing solutions, outperforming diverse and random selection strategies.
- **State-of-the-Art Performance**: We demonstrate that VeoPlace achieves superior results on 4 out of 7 benchmark circuits from ISPD 2005 and ICCAD 2004, which shows significant improvements in Half-Perimeter Wirelength (HPWL) over prior methods.

2 Related Work

Automated Chip Floorplanning. Automating chip floorplanning has been extensively studied through analytical methods [23], black-box optimization techniques such as simulated annealing [39] and genetic algorithms [35], guided black-box methods [33], and learning-based methods [26, 17, 10]. A prominent class formulates chip floorplanning as reinforcement learning where

macros are sequentially placed onto a chip canvas [26, 18, 17]. Other approaches learn to refine existing placements [21, 40]. Our approach generalizes learning-based methods by using a high-level VLM to guide them at test-time.

Vision-Language Models for Decision-Making. Vision-Language Models (VLMs) trained on vast text-image datasets contain rich priors valuable for vision-language tasks [9]. VLMs enable efficient decision-making in robotics, interpreting natural language commands within visual scenes to guide robot actions or planning [3, 15, 34, 14, 4, 16, 37, 22]. Systems such as SayCan [3] and RT-2 [4] demonstrate how VLMs translate high-level instructions into actionable plans for low-level controllers. We leverage similar hierarchical capabilities: the VLM perceives placement images with performance metrics, analyzes spatial arrangements, and provides bounding regions that constrain the low-level policy's action space, creating a division of labor between high-level spatial reasoning and precise placement execution.

Pairing LLMs with Evolution. Pairing LLMs with evolutionary search has achieved successes in program generation [31, 13, 24], planning and reasoning [19], scientific discovery [41, 11], robotics [28], and chip design [29, 42, 40, 33]. VeoPlace adopts an evolutionary framework where the VLM generates region proposals, which are evaluated and used to inform subsequent proposals. VeoPlace's selection strategy focuses on geometrically similar placements to the highest-performing solution (Section 4), explicitly optimizing within promising local regions—effective in sparse Gaussian processes [38] and genetic algorithm island models [31, 19, 36, 5].

3 Preliminaries

3.1 Problem Setup

We address macro placement in chip floorplanning, arranging macros $M = \{m_1, \ldots, m_N\}$ (defined by dimensions and connectivity) onto a 2D chip canvas. Connectivity is specified by a netlist $G = (M, E)$—a hypergraph where vertices are macros and each hyperedge $e \in E$ (called a net) connects a subset of related macros. The objective is finding placement $P = \{p_1, \ldots, p_N\}$, where p_i denotes the coordinates (e.g., bottom-left corner) of macro m_i, that optimizes key physical design metrics. Our primary objective is to minimize estimated wirelength, as it significantly impacts a chip's final performance, power, and area (PPA) [23, 26].

Macro placement can be formulated as a sequential decision-making problem, i.e. as a Markov Decision Process (MDP) [26, 17, 18]. Macros are sequentially placed one by one onto the canvas, following a pre-determined order such as descending macro area. State s_t at timestep t encompasses information about the current partial placement (locations of macros $m_1, \ldots, m_{t-1}$), features of the current macro m_t to be placed, and potentially structural information derived from the netlist G. To manage the continuous nature of the placement space, the canvas is commonly discretized into a grid of cells, where action a_t corresponds to selecting a specific grid cell for placing a reference point (e.g., the bottom-left corner) of the current macro m_t. After all N macros are placed, a terminal reward R is computed based on the estimated wirelength (often using the Half-Perimeter Wirelength, HPWL) of the final placement. The total HPWL is calculated by summing the half-perimeter of the smallest axis-aligned bounding box enclosing all pins connected by each net, over all nets in the netlist G. The agent's goal is to learn a policy $\pi(a_t|s_t)$ that maximizes the expected terminal reward $\mathbb{E}[R]$ (or, equivalently, minimizes HPWL).

Inference-time Optimization. Online RL approaches require many environment interactions and model updates to produce the best possible placements for new netlists. Recent work suggests that offline RL pre-training can achieve relatively strong zero-shot performance, though also finds substantial benefit from fine-tuning on a small amount of online interaction [17]. In this work, we consider an inference-time optimization setting in which we are allowed some fixed budget of placement evaluations, but do not fine-tune either the VLM or the low-level policy. As Section 5 shows, this can achieve results superior to fine-tuning.

3.2 ChiPFormer

Although our approach is compatible with any learning approach that solves the MDP described in Section 3.1, we primarily consider ChiPFormer [17] as our *low-level policy* due to its multi-task

generality. ChiPFormer consists of an autoregressive Transformer model trained using an offline Decision Transformer objective [7, 20]. ChiPFormer represents placement sequences using graph embeddings of the netlist, along with embeddings of states, actions, and returns-to-go, allowing it to be trained as an offline reinforcement learning agent. ChiPFormer outputs probability distributions over grid cells for each macro, which can be directly modified by our VLM guidance. Additionally, it ranks among the most competitive policy-based approaches for macro placement. While ChiPFormer provides strong baseline performance, it lacks the visual and spatial reasoning capabilities of human experts. Our VLM-based guidance addresses this limitation by steering ChiPFormer's action distributions toward better design choices, significantly improving placement quality even without fine-tuning the original policy.

4 VeoPlace

In this section, we describe VeoPlace, our novel evolutionary framework that harnesses the spatial reasoning capabilities of VLMs for chip floorplanning. VeoPlace iteratively evolves a population of region proposals, using VLM as a variation operator. VeoPlace integrates VLM proposals to constrain the rollouts of a low-level placement policy. It consists of two key components: (1) a simple interface between the VLM and low-level placement policy, where the VLM suggests bounding boxes that constrain the placement of each macro on the chip canvas; (2) an evolutionary search strategy that uses the VLM to iteratively provide better region proposals conditioned on previous attempts and the resulting placement quality. This strategy is implemented using a structured prompt that is also designed to elicit spatial reasoning from the VLM. We discuss each of these components in detail below.

Algorithm 1 VeoPlace

Require: V: Vision-language model
π: Stochastic ChiPFormer policy
G: Netlist with macros $\{m_1, ..., m_n\}$
C: Context length
E: Total number of episodes
K: Interval to query the VLM
M: Number of additional rollouts for invalid suggestions

1: Initialize placement population $H \leftarrow \emptyset$
2: **for** episode $e = 1$ to E **do**
3: Initialize placement $P_e \leftarrow \emptyset$
4: **if** $e \bmod K = 0$ **then**
5: context $\leftarrow$ BUILD_CONTEXT(H, C)
6: $\{s_1, s_2, ..., s_n\} \leftarrow V(\text{context}, G)$ {Sample suggestions for all macros}
7: **end if**
8: **for** macro $m_t \in \{m_1, m_2, ..., m_n\}$ **do**
9: **if** suggestion s_t for m_t is valid **then**
10: $p_t \sim \pi(\cdot|m_t, P_e, s_t)$ {Use suggested region}
11: **else**
12: $p_t \sim \pi(\cdot|m_t, P_e)$ {Use original policy}
13: Rollout $P'_e = P_e$ to end M times
14: $H \leftarrow H \cup \{(P'_e, HPWL'_e)\}_1^M$ {Update population}
15: **end if**
16: $P_e \leftarrow P_e \cup \{(m_t, p_t)\}$ {Update placement}
17: **end for**
18: Calculate $HPWL_e$ for placement P_e
19: $H \leftarrow H \cup \{(P_e, HPWL_e)\}$ {Update population}
20: **end for**
21: **return** H

4.1 VLM and Low-Level Policy Interface

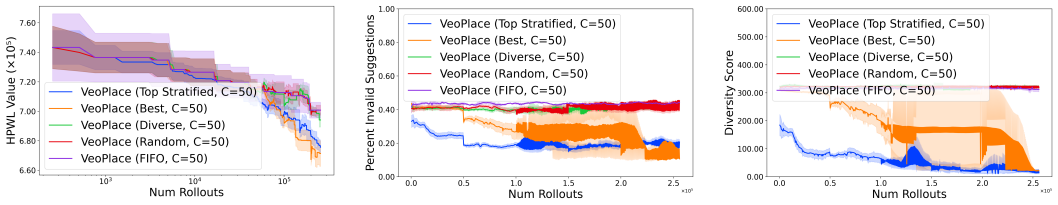
VeoPlace is an inference-time evolutionary search strategy orchestrating interaction between a VLM and stochastic low-level placement policy π. As shown in Algorithm 1, VeoPlace iteratively evolves a population H of placements by rolling out the policy with VLM proposals and generating new proposals based on prior attempts and outcomes.

Following ChiPFormer's process, we generate many candidate placements but additionally leverage VLM high-level guidance. VeoPlace mixes low-level-only rollouts with VLM-guided rollouts every K episodes (Algorithm 1, lines 5-7). When querying the VLM, we provide context from the history buffer H and current netlist G (Section 4.3). The VLM suggests bounding box regions $\{s_1, ..., s_n\}$ for respective macros (Figure 4).

VeoPlace constrains the policy's actions at timestep t to suggested region s_t: $\pi(\cdot|m_t, P_e, s_t)$ (line 10), implemented by masking output logits outside s_t. This allows VLM regional guidance while preserving policy control over exact coordinates within regions. Since macros are placed autoregressively, suggestion s_t may be invalid for $t > 1$ due to overlapping already-placed macros. Invalid suggestions default to the original policy: $p_t \sim \pi(\cdot|m_t, P_e)$ (line 12). We additionally roll out M copies using only the low-level policy to increase data coverage. After completion, each placement's quality (e.g., $HPWL_e$) is calculated and $(P_e, HPWL_e)$ added to population H (lines 18-19).

4.2 Structured Prompt

VeoPlace prompts a VLM to generate bounding box suggestions $\{s_1, ..., s_n\}$ for each macro, conditioned on previous placements and their evaluations $\{P_i\}$. We generate suggestions for all macros



(a) HPWL (Lower is Better) (b) Invalid Suggestions (%) (c) Geometric Diversity Score

Figure 3: Performance comparison of different context building strategies for the adaptec1 benchmark: (a) half-perimeter wirelength (HPWL), showing that less diverse strategies (Undiverse, Best) achieve highest performance, (b) percentage of invalid placement suggestions generated by the VLM after each API call, and (c) geometric diversity scores of the in-context examples. These results demonstrate that minimizing geometric diversity in context selection directly leads to improved evolutionary search performance, enabling the VLM to generate higher-quality placement suggestions.

simultaneously to reduce VLM inference cost. Our prompt's key characteristics are (1) structuring it to elicit spatial reasoning, and (2) selecting appropriate prior attempts for in-context learning (detailed in Section 4.3).

VLMs often struggle with macro placement due to information overload and lack of domain-specific knowledge, producing inconsistent spatial suggestions without proper guidance (see Appendix F.3). Our structured prompt provides clear objectives, constraints, and standardized formats to transform general visual reasoning into useful spatial guidance. The prompt includes:

Grid Representation. We use an 84×84 grid matching ChiPFormer's resolution [17], with coordinates starting at $(0, 0)$ bottom-left. This enables precise positioning while simplifying integration with the low-level policy.

Visual Representation. Canvas images show all placed macros with positions and colors, allowing the VLM to perceive spatial relationships and available space. Figure 4 demonstrates how this visual information improves placement suggestions.

Context Elements. The prompt includes grid specifications, current macro properties, placement history with performance metrics, and current macro states. This comprehensive context enables informed decisions based on both current state and historical performance. A complete example is in Appendix F.1.

4.3 Selection Strategies for Evolution

The core component of our evolutionary algorithm is prompting the VLM to generate a superior placement suggestion given a set of prior placements and their evaluations. Because each placement is represented using hundreds of tokens, only a relatively small number of placements can be provided to the model while maintaining reasonable inference cost and latency. Hence, we find that the prior proposals must be carefully selected – given this limited budget, we would like the examples to (1) be reasonably high-quality, so that the model improves upon already-good placements, and (2) contain enough relevant information so that the model can effectively deduce better placements via reasoning.

We consider the following candidate strategies for selecting a fixed context of C examples:

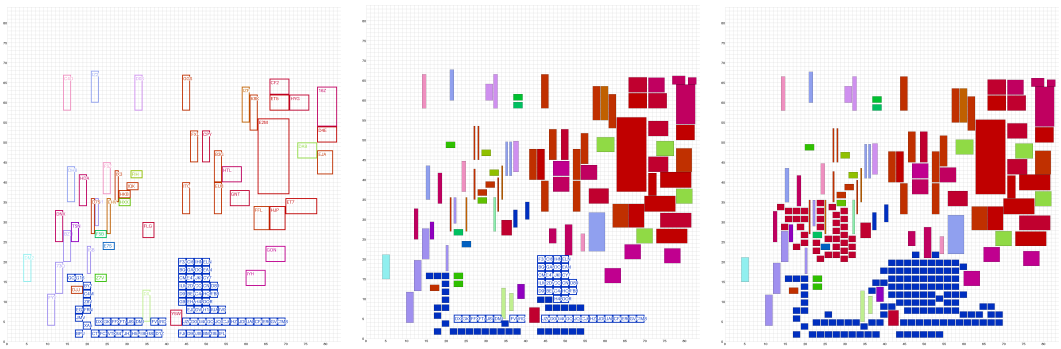
1) **Most Recent (FIFO)**: Select the C most recent placements, implementing pure evolutionary search with temporally ordered examples.

2) **Random**: Uniformly sample C placements from the history.

3) **Best Performing**: Select the C highest-quality placements, assuming successful examples encourage pattern replication.

4) **Diverse**: We employ a clustering-based approach to maximize diversity. First, we represent each placement as a vector in $\mathbb{R}^{2T}$, where T is the number of macros and each macro m_i's (x_i, y_i) coordinates are included. We perform K-means clustering with C clusters, then select the best placement from each cluster. This ensures geometric diversity while favoring high-quality designs.

5) **Top Stratified**: To minimize diversity, we represent placements as coordinate vectors in $\mathbb{R}^{2T}$ and perform K-means clustering. Unlike the diverse strategy, we focus on a single promising cluster. We



(a) $t = 0$ (b) $t = \frac{T}{2}$ (c) $t = T$

Figure 4: VeoPlace's VLM-guided placement on adaptec4. (a) VLM proposes initial regions ($t = 0$); policy is unconstrained for macros without valid suggestions. (b) Mid-placement ($t = T/2$). (c) Final placement ($t = T$), with the policy operating within VLM constraints.

Table 1: HPWL values ($\times 10^5$) obtained by compared methods on seven benchmark circuits for **hard macros only placement** (without standard cells). Each result consists of the mean and standard deviation across three random seeds. The best (smallest) mean value on each circuit is **bolded**. The results for MaskPlace, WireMask-EA, and EfficientPlace are taken directly from [10], while ChiPFormer (Original) results come from [17]. Our reproduction of ChiPFormer shows improved performance due to training with larger batch sizes and for more iterations (10k epochs). See Appendix C for complete training details.

Method	Type	adaptec1	adaptec2	adaptec3	adaptec4	bigblue1	bigblue3	bigblue4
MaskPlace [18]	RL	7.62 ± 0.67	75.16 ± 4.97	100.24 ± 13.54	87.99 ± 3.25	3.04 ± 0.06	90.04 ± 4.83	103.26 ± 2.69
WireMask-EA [33]	BBO	6.15 ± 0.15	64.38 ± 4.43	58.18 ± 1.04	59.52 ± 1.71	2.15 ± 0.01	59.85 ± 3.39	77.54 ± 0.67
EfficientPlace [10]	MCTS + RL	$\mathbf{5.94 \pm 0.04}$	$\mathbf{46.79 \pm 1.60}$	56.35 ± 0.99	58.47 ± 1.61	$\mathbf{2.14 \pm 0.01}$	58.38 ± 0.54	76.63 ± 1.02
ChiPFormer [17] (Original)	IL	6.62 ± 0.05	67.10 ± 5.46	76.70 ± 1.15	68.80 ± 1.59	2.95 ± 0.04	72.92 ± 0.32	102.84 ± 0.15
ChiPFormer (Reproduced)	IL	7.01 ± 0.08	68.25 ± 1.46	51.92 ± 2.68	26.32 ± 2.27	2.92 ± 0.01	37.99 ± 1.65	63.86 ± 1.54
VeoPlace (Top Stratified)	IL+VLM	6.78 ± 0.12	64.00 ± 1.96	$\mathbf{51.64 \pm 2.92}$	$\mathbf{26.05 \pm 1.84}$	2.90 ± 0.02	$\mathbf{36.38 \pm 0.42}$	$\mathbf{63.41 \pm 1.90}$

rank clusters by minimum wirelength and sample one cluster using softmax (probability $\propto e^{-i/\tau}$ for rank i). From the selected cluster, we choose the top C performing layouts, supplementing from nearby clusters if needed.

Our findings in Section 5 reveal that minimizing geometric diversity (Top Stratified) yields best performance, suggesting consistency helps VLMs identify relevant placement patterns by allocating the limited representational budget to the most promising search region [38].

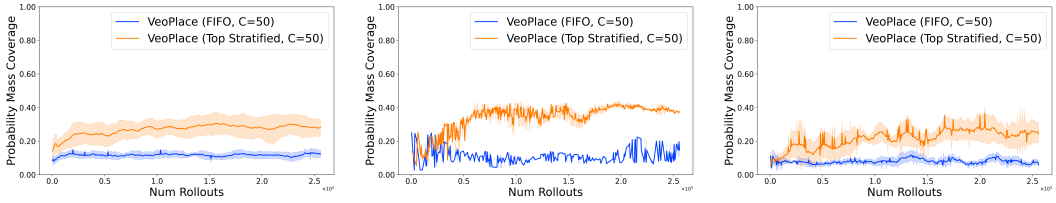
5 Experiments

We evaluate VeoPlace's ability to harness VLMs to improve chip floorplanning on unseen blocks, addressing: **(Q1)** Can VLM guidance improve floorplan quality using only inference? **(Q2)** How does VLM guidance complement low-level policies? **(Q3)** How does context length affect VeoPlace's performance?

Experimental Setup. Our evaluation follows [17], using open-source benchmarks from ISPD 2005 [27] and ICCAD 2004 [2, 1]. These vary in complexity with hundreds to thousands of macros and up to hundreds of thousands of standard cells, providing comprehensive testing across scales. We use a fixed ChiPFormer model trained using the official implementation[1]. We set our VLM call interval $K = 512$, which means we roll out the pre-trained ChiPFormer policy 512 times before querying the VLM. At the beginning of each episode, the VLM is queried to acquire region suggestions $(s_1, s_2, \ldots)$ for each macro $(m_1, m_2, \ldots)$. These suggested regions are then used to guide the policy during placement decisions. For the VLM, we use Gemini 2.0 Flash with temperature 0.7.

Q1: VLM Guidance Improves Placement Quality. We test whether evolutionary search with VLMs can effectively tackle macro placement. Table 1 compares VeoPlace against state-of-the-art methods across benchmark circuits. VeoPlace achieves state-of-the-art HPWL on four out of seven

[1] `https://github.com/laiyao1/chipformer`



(a) adaptec1 (b) adaptec2 (c) adaptec3

Figure 5: Probability mass coverage for VLM guidance strategies. FIFO (blue) shows minimal coverage (5-15%), largely diverging from the policy's learned knowledge, while Top Stratified (orange) maintains moderate coverage (20-40%), balancing policy knowledge with effective exploration.

benchmarks, demonstrating that VLM guidance effectively enhances placement quality with a fixed pre-trained policy.

Q2: VLM Guidance Complements Low-Level Policies. To evaluate VLM-policy interaction, we define *probability mass coverage* as the fraction of the policy's original probability distribution within VLM-suggested regions. Figure 5 shows our Top Stratified strategy maintains 20-40% coverage across benchmarks. This moderate coverage indicates meaningful constraint without excessive restriction. The guidance neither completely overrides policy preferences (near 0% coverage) nor simply copies them (approaching 100%), instead redirecting attention to underexplored regions and enhancing design space exploration.

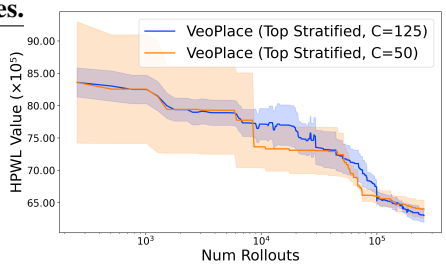


Figure 6: Effect of context length on placement quality for adaptec2. Increasing context from $C = 50$ to $C = 125$ shows small improvements in final HPWL.

Q3: Effect of Context Length C on VLM Performance. We analyze how context length C (previous placements shown to VLM) affects quality and cost. Figure 6 shows performance for $C \in \{50, 125\}$. Increasing context from $C = 50$ to $C = 125$ improves performance on larger netlists like adaptec2, where more examples help the VLM capture spatial relationships and leverage placement patterns. Additional context enables identifying sophisticated spatial patterns in complex designs, though computational cost increases linearly.

6 Conclusion & Discussion

We proposed VeoPlace, an evolutionary framework leveraging vision-language models to enhance macro placement policies for chip floorplanning, and demonstrated that it can match or exceed state-of-the-art performance on open-source circuit benchmarks. Our approach broadly underscores how the general priors in foundation models can complement domain-specific policies.

Due to computational constraints, our instantiation of VeoPlace used relatively few queries to a medium-sized, off-the-shelf VLM. We expect the performance of VeoPlace to inherit the well-known scaling properties of foundation models with respect to model size, context length, and inference-time compute. Similarly, we expect that fine-tuning the high-level VLM with domain-specific data to offer significant improvements, especially in domains such as chip design in which relevant pre-training data may be scarce. Future work will extend VeoPlace to global placement by providing the VLM with post-processed floorplans from tools like DreamPlace [23] that include both macro and standard cell placements. Since macro initialization significantly impacts global placement quality, VLM guidance informed by complete circuit layouts could yield more effective macro positioning strategies.

References

[1] Saurabh N Adya, Shubhyant Chaturvedi, Jarrod A Roy, David A Papa, and Igor L Markov. Unification of partitioning, placement and floorplanning. In *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.*, pages 550–557. IEEE, 2004.

[2] Saurabh N Adya and Igor L Markov. Consistent placement of macro-blocks using floorplanning and standard-cell placement. In *Proceedings of the 2002 international symposium on Physical design*, pages 12–17, 2002.

[3] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[5] Erick Cantú-Paz et al. A survey of parallel genetic algorithms. *Calculateurs paralleles, reseaux et systems repartis*, 10(2):141–171, 1998.

[6] Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14455–14465, 2024.

[7] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[8] Chung-Kuan Cheng, Andrew B Kahng, Ilgweon Kang, and Lutong Wang. Replace: Advancing solution quality and routability validation in global placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(9):1717–1730, 2018.

[9] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, et al. Palm-e: An embodied multimodal language model. 2023.

[10] Zijie Geng, Jie Wang, Ziyan Liu, Siyuan Xu, Zhentao Tang, Mingxuan Yuan, Jianye Hao, Yongdong Zhang, and Feng Wu. Reinforcement learning within tree search for fast macro placement. In *Forty-first International Conference on Machine Learning*, 2024.

[11] Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, et al. Towards an ai co-scientist. *arXiv preprint arXiv:2502.18864*, 2025.

[12] Aric Hagberg, Pieter J Swart, and Daniel A Schult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), 2008.

[13] Erik Hemberg, Stephen Moskal, and Una-May O'Reilly. Evolving code with a large language model. *Genetic Programming and Evolvable Machines*, 25(2):21, 2024.

[14] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

[15] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*, 2(3):6, 2022.

[16] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[17] Yao Lai, Jinxin Liu, Zhentao Tang, Bin Wang, Jianye Hao, and Ping Luo. Chipformer: Transferable chip placement via offline decision transformer. In *International Conference on Machine Learning*, pages 18346–18364. PMLR, 2023.

[18] Yao Lai, Yao Mu, and Ping Luo. Maskplace: Fast chip placement via reinforced visual representation learning. *Advances in Neural Information Processing Systems*, 35:24019–24030, 2022.

[19] Kuang-Huei Lee, Ian Fischer, Yueh-Hua Wu, Dave Marwood, Shumeet Baluja, Dale Schuurmans, and Xinyun Chen. Evolving deeper llm thinking. *arXiv preprint arXiv:2501.09891*, 2025.

[20] Kuang-Huei Lee, Ofir Nachum, Mengjiao Sherry Yang, Lisa Lee, Daniel Freeman, Sergio Guadarrama, Ian Fischer, Winnie Xu, Eric Jang, Henryk Michalewski, et al. Multi-game decision transformers. *Advances in Neural Information Processing Systems*, 35:27921–27936, 2022.

[21] Vint Lee, Chun Deng, Leena Elzeiny, Pieter Abbeel, and John Wawrzynek. Chip placement with diffusion. *arXiv preprint arXiv:2407.12282*, 2024.

[22] Jacky Liang, Fei Xia, Wenhao Yu, Andy Zeng, Montserrat Gonzalez Arenas, Maria Attarian, Maria Bauza, Matthew Bennice, Alex Bewley, Adil Dostmohamed, et al. Learning to learn faster from human feedback with language model predictive control. *arXiv preprint arXiv:2402.11450*, 2024.

[23] Yibo Lin, Shounak Dhar, Wuxi Li, Haoxing Ren, Brucek Khailany, and David Z. Pan. DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, Las Vegas NV USA, June 2019. ACM.

[24] Vadim Liventsev, Anastasiia Grishina, Aki Härmä, and Leon Moonen. Fully autonomous programming with large language models. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1146–1155, 2023.

[25] Jingwei Lu, Pengwen Chen, Chin-Chih Chang, Lu Sha, Dennis Jen-Hsin Huang, Chin-Chi Teng, and Chung-Kuan Cheng. eplace: Electrostatics-based placement using fast fourier transform and nesterov's method. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 20(2):1–34, 2015.

[26] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nova, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

[27] Gi-Joon Nam, Charles J Alpert, Paul Villarrubia, Bruce Winter, and Mehmet Yildiz. The ispd2005 placement contest and benchmark suite. In *Proceedings of the 2005 international symposium on Physical design*, pages 216–220, 2005.

[28] Soroush Nasiriany, Fei Xia, Wenhao Yu, Ted Xiao, Jacky Liang, Ishita Dasgupta, Annie Xie, Danny Driess, Ayzaan Wahid, Zhuo Xu, et al. Pivot: Iterative visual prompting elicits actionable knowledge for vlms. In *International Conference on Machine Learning*, pages 37321–37341. PMLR, 2024.

[29] Alexander Novikov, Ngân Vu, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. Technical report, Google Deepmind, 2025.

[30] Atin Pothiraj, Elias Stengel-Eskin, Jaemin Cho, and Mohit Bansal. Capture: Evaluating spatial reasoning in vision language models via occluded object counting. *arXiv preprint arXiv:2504.15485*, 2025.

[31] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

[32] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[33] Yunqi Shi, Ke Xue, Song Lei, and Chao Qian. Macro placement by wire-mask-guided black-box optimization. *Advances in Neural Information Processing Systems*, 36:6825–6843, 2023.

[34] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on robot learning*, pages 894–906. PMLR, 2022.

[35] T Singha, HS Dutta, and M De. Optimization of floor-planning using genetic algorithm. *Procedia Technology*, 4:825–829, 2012.

[36] Reiko Tanese. *Distributed genetic algorithms for function optimization*. University of Michigan, 1989.

[37] Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

[38] Yunyue Wei, Vincent Zhuang, Saraswati Soedarmadji, and Yanan Sui. Scalable bayesian optimization via focalized sparse gaussian processes. *arXiv preprint arXiv:2412.20375*, 2024.

[39] DF Wong and CL Liu. A new algorithm for floorplan design. In *23rd ACM/IEEE Design Automation Conference*, pages 101–107. IEEE, 1986.

[40] Ke Xue, Ruo-Tong Chen, Xi Lin, Yunqi Shi, Shixiong Kai, Siyuan Xu, and Chao Qian. Reinforcement learning policy as macro regulator rather than macro placer. *arXiv preprint arXiv:2412.07167*, 2024.

[41] Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

[42] Xufeng Yao, Jiaxi Jiang, Yuxuan Zhao, Peiyu Liao, Yibo Lin, and Bei Yu. Evolution of optimization algorithms for global placement via large language models. *arXiv preprint arXiv:2504.17801*, 2025.

Appendix

Table of Contents

A Additional Details

A.1 Macro Coloring

We employ a color-coding strategy to implicitly convey functional relationships between macros to the VLM. This process involves several steps:

First, we construct a **macro-connectivity graph** from the original netlist G. In this graph, nodes (excluding standard cells) represent the macros to be placed. An undirected, weighted edge is created between any two macros if they share one or more nets in G. The weight of such an edge is proportional to the number of nets these two macros commonly share. This construction effectively flattens the hypergraph structure of the netlist into a standard graph, where indirect connections through nets are represented as weighted between macros.

This macro-connectivity graph is embedded into a low-dimensional space (specifically, an 8-dimensional space in our implementation for k-means) using a spring-based graph layout algorithm. Such algorithms, like the one implemented in the NetworkX [12] Python library, position macros in the embedding space such that those with stronger connections in the graph are located closer to one another in space.

With macros represented as points in this embedding space, we apply k-means clustering to group them. To determine a suitable number of clusters, k, we iterate through a predefined range of potential k values (e.g., from 2 to 30). For each k, we perform k-means clustering and evaluate the resulting cluster separation using the Silhouette score [32]. The value of k (and its corresponding clustering) that yields the highest Silhouette score is selected as optimal.

Finally, macros are assigned colors based on their cluster membership: all macros within the same cluster receive the same unique color. Any macros that are not part of the main connectivity graph (e.g., isolated macros not sharing nets with other considered macros, if any) are assigned a default gray color. While specific netlist connectivity details are not directly fed to the VLM, this color-coding, derived from the underlying circuit structure, provides a strong visual heuristic for potential functional groupings and spatial affinities.

B Experimental Setup

B.1 Model Size

The ChipFormer decision transformer model contains approximately 3 million trainable parameters.

B.2 Pre-training

The pre-training phase was conducted on servers equipped with 4× NVIDIA A100 40GB GPUs.

Following the original work, we trained the circuit token representations using Variational Graph Auto-Encoders (VGAE) and used similar hyperparameter settings. Notably, we were able to use a larger batch size (256 vs. their 32) by leveraging four Nvidia A100s in our training infrastructure, and trained for 10,000 epochs. These modifications led to significant performance improvements over the original ChiPFormer results, even when using the same architecture and codebase. Further details on our ChiPFormer configuration are provided in Appendix C.1.2.

B.3 Rollout and Inference

For generating rollouts, the computational requirements were significantly lower as these involved only forward passes through the trained 3M parameter model. These were conducted on a single A100 GPU or equivalent, with each benchmark circuit rollout typically completing within seconds.

B.4 VLM Integration

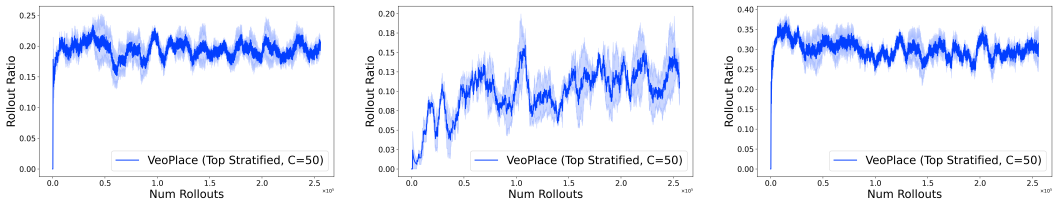
For VLM integration, we used Google's Gemini API with the Gemini 2.0 Flash model. Our experiments were organized into iterations, where each iteration involved collecting exactly 256 rollouts. These rollouts were generated either with Gemini guidance or using the low-level policy alone.

We set a query interval of $K = 512$, meaning that we queried Gemini every other iteration. With this setup and a maximum of 1000 iterations per experiment, the VLM was called 500 times throughout each experiment. Each call to Gemini returned 8 candidate generations, with each generation being a complete set of suggested placements for all macros in the netlist.

To maximize the benefit of this limited VLM guidance, we implemented a "branching" strategy (illustrated in Figure 7). When Gemini made an invalid suggestion for a particular macro, we generated a new rollout where the low-level policy was allowed to place that specific macro while following Gemini's suggestions for other macros when possible. This branching could potentially generate more than 256 rollouts per iteration, so to standardize the number of rollouts between iterations, we capped each iteration at exactly 256 rollouts and discarded any excess branched rollouts beyond this limit. We observed that the ratio of Gemini-guided placements to low-level policy placements varies with netlist size - smaller netlists like adaptec1 (63 macros) and adaptec2 (159 macros) have smaller ratios because there are fewer opportunities for invalid suggestions and thus less branching, while larger netlists like adaptec4 (256 macros) exhibit higher ratios due to more invalid suggestions requiring more branched rollouts.

This standardization ensured that iterations with Gemini guidance and those with only the low-level policy contributed equally to the total rollout count. With 1000 iterations and exactly 256 rollouts per iteration, each experiment produced a total of $256,000$ rollouts.

One limitation we encountered is the output token limit of Gemini 2.0 Flash (8,192 tokens), which can constrain the number of suggested regions for larger netlists. The impact varies depending on the VLM's verbosity—if the model provides more explanatory text before listing regions, fewer macros receive guidance before reaching the token limit. As output token limits for VLMs continue to increase, we expect more consistent and comprehensive guidance across all chip designs, regardless of size or model verbosity.



(a) adaptec1 (b) adaptec2 (c) adaptec4

Figure 7: Ratio of Gemini-guided rollouts to low-level policy rollouts across three benchmark circuits. For smaller netlists like adaptec1, we observe fewer VLM-guided rollouts because there are fewer invalid suggestions requiring branching. Larger netlists like adaptec4 show higher ratios due to more invalid suggestions, which generate more branched rollouts. Each iteration is capped at 256 total rollouts, with excess branched rollouts being discarded.

C Hyperparameters

C.1 ChiPFormer

C.1.1 Pretraining

Circuit Tokens For pretraining the circuit token representation component using the Variational Graph Auto-Encoder (VGAE), we used the following hyperparameters:

- Hidden layer dimensions: [32, 32]
- Learning rate: 0.01
- Training epochs: 800

Transformer Following ChiPFormer [17], we use a reward-conditioned transformer with the following hyperparameters:

- Number of transformer layers: 6
- Number of attention heads: 8
- Embedding dimension: 128

C.1.2 Rollout Settings

Returns-to-go We configured specific target returns-to-go for each benchmark netlist to guide the generated placements. Following the methodology of Decision Transformers, we set these values higher than the maximum normalized scores observed in the training dataset to encourage the model to generate high-quality placements. Table 2 shows the target returns-to-go values used for each benchmark circuit in our experiments.

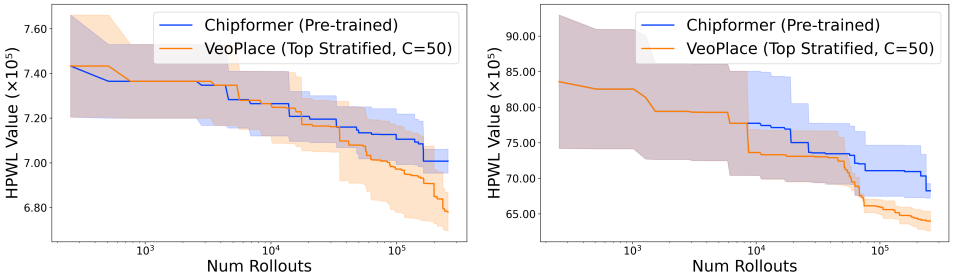
Table 2: Target Returns-to-Go for Different Benchmark Netlists

Netlist	Return-to-go
adaptec1	1.50
adaptec2	1.80
adaptec3	1.30
adaptec4	1.15
bigblue1	0.80
bigblue2	1.20
bigblue3	1.30
bigblue4	1.10
ibm01	1.20
ibm02	0.90
ibm03	1.15
ibm04	0.50

D Additional Experiments

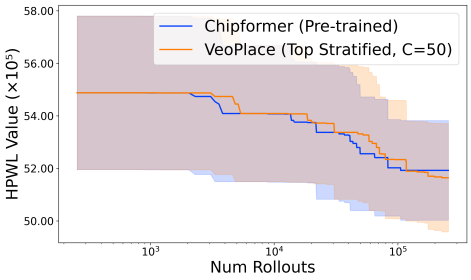
D.1 HPWL Comparison

This section presents the experimental results specific to each of the eight netlists used in our evaluation. The figures below illustrate the performance characteristics of our algorithm across the different circuit designs in terms of Half-Perimeter Wirelength (HPWL), measured in units of 10^5. Lower HPWL values indicate better placement quality with reduced interconnection length, demonstrating the effectiveness of our placement strategy across varying netlist complexities and structures.

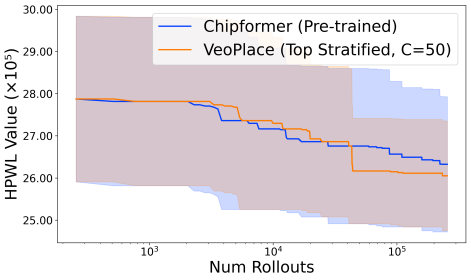


(a) adaptec1 (HPWL $\times 10^5$)

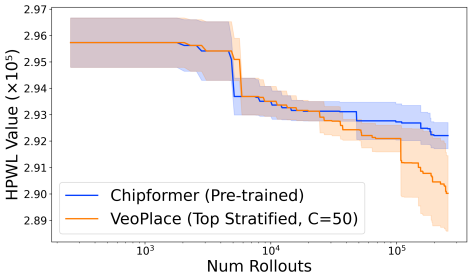
(b) adaptec2 (HPWL $\times 10^5$)



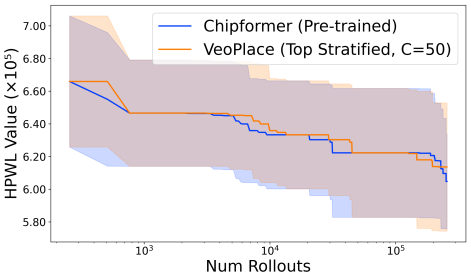
(c) adaptec3 (HPWL $\times 10^5$)



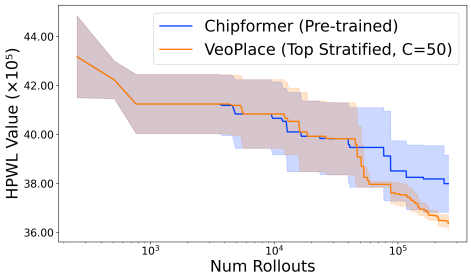
(d) adaptec4 (HPWL $\times 10^5$)



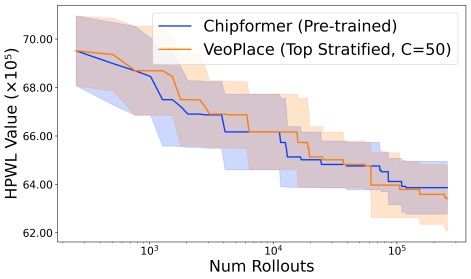
(e) bigblue1 (HPWL $\times 10^5$)



(f) bigblue2 (HPWL $\times 10^5$)



(g) bigblue3 (HPWL $\times 10^5$)



(h) bigblue4 (HPWL $\times 10^5$)

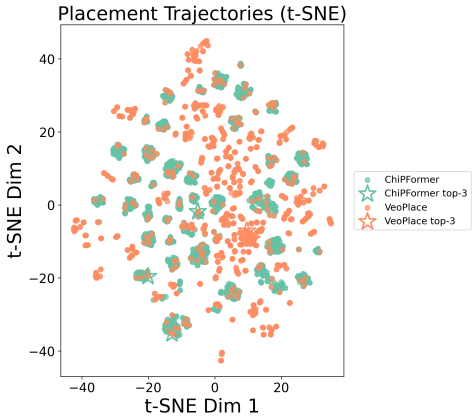
Figure 8: Half-Perimeter Wirelength (HPWL $\times 10^5$) results for individual netlists used in our evaluation. Lower values indicate better placement quality with reduced interconnection length.

D.2 Design Space Exploration

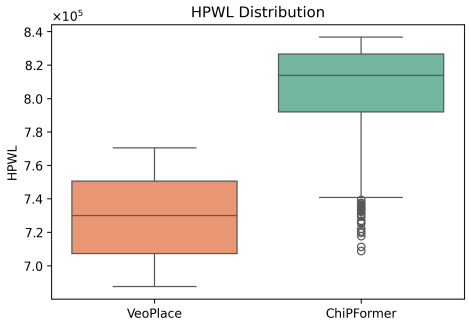
To better understand the benefits of VLM guidance, we visualize the placements generated by VeoPlace and contrast them with the placements generated by ChiPFormer alone using t-SNE for a single random seed per benchmark circuit (Figure 9).

Our analysis focuses on the top 2000 placements from each random seed, revealing interesting patterns in the design space exploration. For the specific adaptec1 and adaptec2 seeds shown, where VeoPlace outperforms ChiPFormer, the t-SNE plots show that VeoPlace (orange points) explores distinctly different regions of the design space compared to ChiPFormer (blue points). Notably, the top-3 placements from VeoPlace (marked with stars) occupy areas that ChiPFormer fails to explore, suggesting that VLM guidance enables access to solution regions that remain undiscovered by the low-level policy alone.

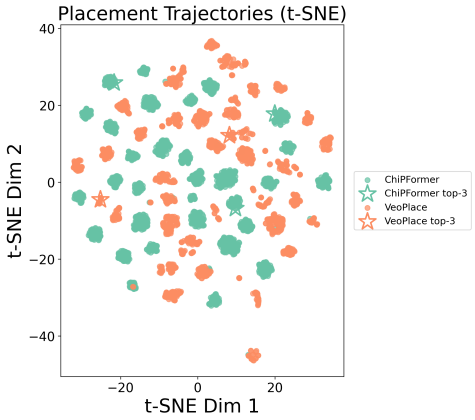
In contrast, for the adaptec4 seed shown, where VeoPlace did not outperform ChiPFormer, we observe that the design space exploration patterns between the two methods largely overlap. This visualization provides insight into how VeoPlace's guidance affects the exploration process, showing that when VLM guidance leads to performance improvements, it often directs the search toward qualitatively different regions of the solution space.



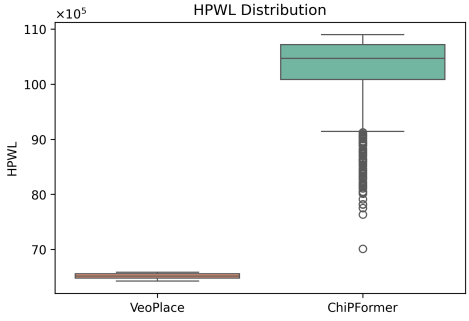
(a) adaptec1 t-SNE visualization



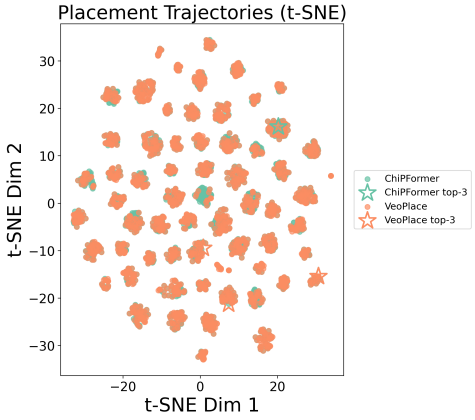
(b) adaptec1 performance



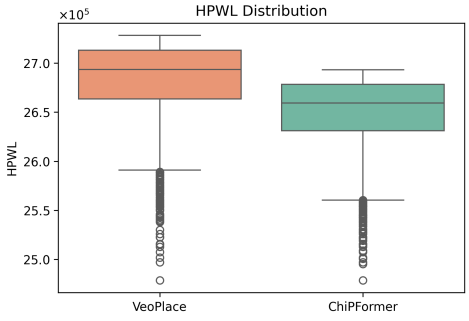
(c) adaptec2 t-SNE visualization



(d) adaptec2 performance



(e) adaptec4 t-SNE visualization



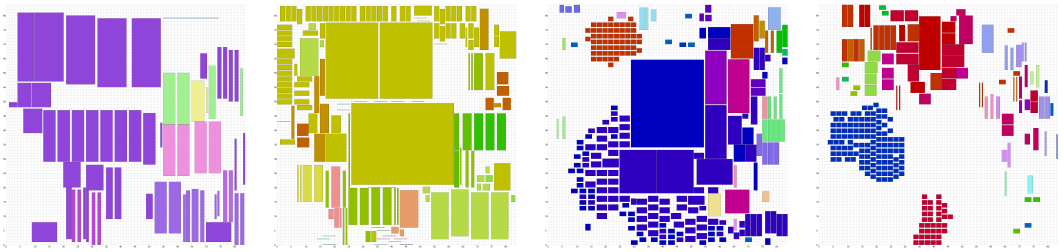
(f) adaptec4 performance

Figure 9: Left column: t-SNE visualization comparing the design spaces explored by ChipFormer (blue) and VeoPlace (orange), with stars denoting the top three placements for each method. Right column: Half-perimeter wirelength comparison boxplots. All results are from the top 100 placements from a single random seed.

E Example Placement Images

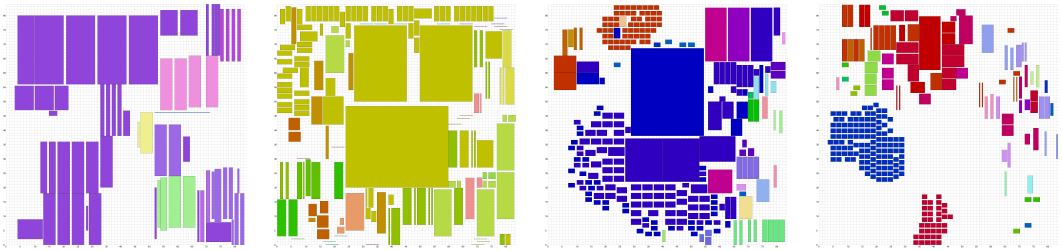
Figure 10: Comparison of Placements Generated by VeoPlace vs. Chipformer on the Adaptec Benchmarks

(a) ChipFormer



(b) adaptec1 (c) adaptec2 (d) adaptec3 (e) adaptec4

(f) VeoPlace



(g) adaptec1 (h) adaptec2 (i) adaptec3 (j) adaptec4

Figure 10 shows a visual comparison of placements generated by ChipFormer and VeoPlace on the adaptec benchmark circuits. The figure illustrates that VeoPlace produces different macro placement arrangements compared to ChipFormer, which is used as the low-level policy. This demonstrates that VeoPlace is not simply replicating the solutions of its low-level policy, but is exploring different macro arrangement strategies in the placement space.

F Prompt Details

F.1 Example Prompt

Prompt example: variable regions

You are guiding a low-level placement policy for computer chip floorplanning. Your primary goal is to create the most optimal chip placement possible that minimizes wirelength and cost. Your task is to suggest rectangular regions for placing macros on the chip canvas, which has been divided into a grid. The low-level policy will choose the exact placement location within your suggested regions. Your suggestions should be highly precise and optimal. If there is a macro in the netlist that you are not providing a suggestion for, the low-level policy will place that macro by itself.

The macros are grouped by colors based on their connectivity in the netlist graph, where macros with higher interconnectivity (more pin connections between them) are assigned similar colors. Your goal is to provide optimal region suggestions that will result in the best possible chip placement with minimal wirelength and cost.

This is a global optimization task where you need to consider:

- The impact of your suggested regions on macros that will be placed in the future
- The overall arrangement of the selected macros that minimizes wirelength and cost

MACRO NAMES AND PROPERTIES FOR THIS NETLIST:

Macro	Color	WxH
FD4	#9b69e6	2 x 18
CXC	#8f45da	11 x 24
HKU	#8f45da	11 x 24
FZ6	#8f45da	11 x 24
CWI	#8f45da	11 x 24
EIO	#8f45da	6 x 24
JXA	#8f45da	5 x 18
V8F	#8f45da	5 x 18
G1F	#8f45da	5 x 18
IJS	#8f45da	5 x 18
JPT	#8f45da	5 x 18
DU2	#8f45da	5 x 18
J6X	#8f45da	5 x 18
HJ5	#8f45da	5 x 18
0IL	#ef90df	5 x 18
FIF	#ef90df	5 x 18
E6W	#ef90df	5 x 18
ELG	#ef90df	5 x 18
HDJ	#a0ef90	5 x 18
DSU	#9b69e6	5 x 18
G25	#a0ef90	5 x 18
IOQ	#9b69e6	5 x 18
KV6	#efef90	5 x 15
IYX	#8f45da	9 x 7
IIC	#8f45da	9 x 7
F87	#8f45da	7 x 9
GVY	#8f45da	7 x 9
ISA	#8f45da	7 x 9
GJ6	#8f45da	7 x 9
FIY	#a0ef90	3 x 19
PEJ	#9b69e6	3 x 19

Macro	Color	WxH
JQ5	#8f45da	3 x 18
EE4	#8f45da	3 x 18
CH6	#8f45da	5 x 9
F3D	#9b69e6	2 x 18
BKG	#b545da	2 x 19
I64	#b545da	2 x 19
ELR	#8f45da	2 x 18
BCZ	#8f45da	2 x 18
DSH	#8f45da	2 x 18
DEH	#8f45da	2 x 18
BLU	#b545da	2 x 19
MK3	#b545da	2 x 19
CYR	#9b69e6	2 x 18
CPS	#9b69e6	2 x 18
GLZ	#b469e6	2 x 18
BF1	#b469e6	2 x 18
EPJ	#8f45da	3 x 9
IHG	#8f45da	3 x 9
C55	#8f45da	1 x 18
I6P	#8f45da	1 x 18
G5X	#8f45da	1 x 18
HF5	#8f45da	1 x 18
JF5	#9b69e6	1 x 17
GUA	#a0ef90	1 x 17
GF8	#8f45da	1 x 18
I6E	#8f45da	1 x 18
FZI	#8f45da	3 x 2
78E	#9b69e6	1 x 9
J5L	#efef90	1 x 9
JN6	#9b69e6	1 x 9
CWF	#8f45da	1 x 9
GV3	#90bfef	20 x 1

IMPORTANT PLACEMENT RULES:

1. The chip canvas is 84×84.
2. Coordinate system:
   - Origin `(0,0)` is at the bottom-left corner.
   - Top-left corner is `(0,84)`.
   - Bottom-right corner is `(84,0)`.
   - Top-right corner is `(84,84)`.
3. Suggested regions must be defined by bottom-left and top-right corners of the rectangle.
4. Suggested regions must not overlap with each other.
5. Suggestions are needed for these selected macros:
   - **CXC**
     - Size: 11×24
     - Color: `#8f45da`
   - **0IL**
     - Size: 5×18
     - Color: `#ef90df`
   - **G1F**
     - Size: 5×18
     - Color: `#8f45da`
   - **HDJ**
     - Size: 5×18
     - Color: `#a0ef90`
   - **KV6**
     - Size: 5×15
     - Color: `#efef90`
   - **GJ6**
     - Size: 7×9
     - Color: `#8f45da`
   - **BKG**
     - Size: 2×19
     - Color: `#b545da`
   - **FD4**
     - Size: 2×18
     - Color: `#9b69e6`
   - **GLZ**
     - Size: 2×18
     - Color: `#b469e6`
   - **GV3**
     - Size: 20×1
     - Color: `#90bfef`

PLACEMENT QUALITY METRICS:

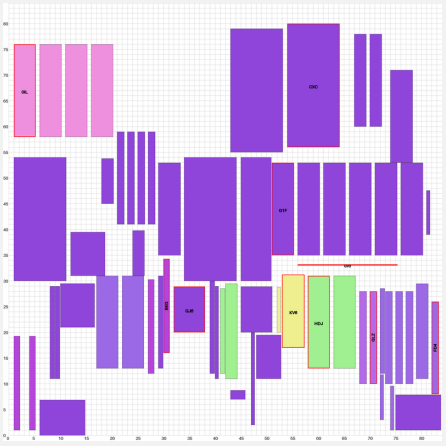
- Higher reward is better
- Lower cost is better
- Lower wirelength is better

PREVIOUS PLACEMENT EPISODES:
Below are previous episodes with their final results. For each episode, you'll see:

- **Macro Positions**: Shows where the selected macros you need to place were put on the canvas in previous episodes
- **Canvas Image**: Shows the final state of the canvas with:
  - The names of each macro you need to place drawn directly on the macro
  - These selected macros outlined in red for easy identification
- **Final Metrics**: The overall quality metrics of the completed chip design

Episode #1
Position of Selected Macros:

- FD4: (82,8) to (84,26)
- CXC: (54,56) to (65,80)
- G1F: (51,35) to (56,53)
- 0IL: (1,58) to (6,76)
- HDJ: (58,13) to (63,31)
- KV6: (53,17) to (58,32)
- GJ6: (32,20) to (39,29)
- BKG: (30,16) to (32,35)
- GLZ: (70,10) to (72,28)
- GV3: (56,33) to (76,34)



Canvas Description and Metrics
The image above shows the final placement with the selected macros you need to place outlined in red and labeled with their names.
Results for Episode #1:

- Wirelength: 2.18e+06
- Cost: 2.25e+06
- Reward: -1.75e+04

[Additional episodes are listed here]

IMPORTANT OUTPUT FORMAT:

1. All coordinates must be integers between 0 and 84.
2. All regions must have non-zero width and height ($x_2 > x_1$ and $y_2 > y_1$).
3. The orientation of macros cannot be changed. Do not try to rotate macros.
4. All regions must be large enough to fit the macro while still within the bounds of the canvas. For example, if a macro size is 3.1×4.2, the region must be at least 4×5.

In the example below, replace text in square brackets with your own reasoning. Do not copy the text inside the brackets. Follow this example format exactly (without the dashed lines):
DETAILED PLACEMENT HISTORY ANALYSIS:
HISTORICAL PLACEMENT PATTERNS:
COLOR GROUP POSITION ANALYTICS:

- [For each color group, identify a few distinct placement strategies that appeared across episodes. Group similar episodes together.]
- [For each strategy, select one representative episode with exact coordinates and resulting wirelength/cost values.]
- [Identify which placement locations produced the best results. Format: "Color group X performed best when placed in region (coordinates) as seen in Episode Y, with wirelength/cost values of Z and W respectively."]

MACRO-LEVEL SPATIAL RELATIONSHIPS:

- [For the largest macros, compare their placement in the best vs. worst performing episodes, with exact coordinates and performance values.]
- [Specify the exact performance impact of different macro orderings: "When macro X was placed left of macro Y in specific episodes, wirelength/cost/reward was lower than when Y was placed left of X in other episodes."]
- [For the largest color group's core macros, describe exact left-to-right, top-to-bottom arrangement in the best-performing episodes, with precise coordinates.]
- [Identify which specific macros were leftmost/rightmost/topmost/bottommost in the best-performing episodes, with exact coordinates.]
- [For critical macro pairs, quantify the benefit of edge alignment: "Macros A and B sharing a vertical edge at specific coordinates resulted in better wirelength/cost/reward than when separated by specific units."]
- [Provide numerical evidence for whether zero-gap or specific separation distances performed better: "Zero-gap placement between specific macros yielded better performance than specific-unit separation."]

ADJACENCY RELATIONSHIP ANALYSIS:

- [For each pair of color groups, analyze multiple episodes with different adjacency patterns. Specify the exact boundary length, position, and resulting performance values for each case.]
- [Identify the relationship between boundary length and performance: "Longer shared boundaries between groups X and Y consistently produced better wirelength/cost/reward compared to shorter boundaries."]
- [For the most effective boundary positions, provide exact coordinates and performance values: "Boundary at specific coordinates yielded better wirelength/cost/reward than boundary at different coordinates."]
- [Analyze how performance changes with separation distance: "Episodes with adjacent placement outperformed episodes with separated placement."]
- [Compare horizontal vs. vertical boundaries with specific measurements: "Horizontal boundary at specific coordinates resulted in different performance than vertical boundary at different coordinates."]
- [Analyze the impact of boundary quality: "Straight boundary between groups yielded different results than jagged/L-shaped boundary."]
- [Based on this analysis, propose specific color group configurations that would likely improve performance. Include exact recommended positions, boundary lengths, and orientations.]

CRITICAL EDGE ALIGNMENTS:

- [Identify specific edge alignments between named macros that consistently corresponded with better performance across multiple episodes. Distinguish between coincidental and meaningful alignments.]
- [Provide precise coordinates and quantify the performance differences: for example, "When specific macros had aligned edges at specific coordinates, wirelength/cost/reward was consistently lower than when these edges were offset."]

FORMATION ANALYSIS:

- [Analyze how the overall arrangement and shape formed by each color group related to performance metrics. Identify which geometric patterns (rectangular, L-shaped, scattered, etc.) consistently corresponded with better performance.]
- [Provide exact coordinates and performance data: for example, "When color group X was arranged in a specific geometric pattern at coordinates (a,b)–(c,d), it achieved better wirelength/cost/reward than when arranged in a different pattern at coordinates (e,f)–(g,h)."]

CANVAS UTILIZATION INSIGHTS:

- [Examine the relationship between overall canvas utilization and performance metrics. Consider both global utilization and local density variations.]
- [Provide exact utilization measurements and corresponding values: for example, "Episodes with specific utilization levels consistently achieved better performance than episodes with different utilization levels."]

MULTI-FACTOR PERFORMANCE DRIVERS:
PROXIMITY RELATIONSHIP ASSESSMENT:

- [Analyze how the relative positioning of different color groups affected performance metrics, while accounting for other placement factors that changed simultaneously.]
- [Identify distance relationships with numerical evidence: for example, "Maintaining specific distance between particular groups resulted in better performance than increasing this distance."]

MACRO PLACEMENT SENSITIVITY:

- [For each major macro, assess how sensitive performance metrics were to its specific placement. Quantify this sensitivity.]
- [Provide exact coordinates and performance impacts: for example, "Moving specific macros from one position to another significantly affected wirelength/cost/reward, indicating high placement sensitivity."]

CONTEXTUAL POSITIONING ANALYSIS:

- [Examine how the optimal positioning of color groups and macros varied depending on the placement context of other elements.]
- [Provide specific examples with measurements: for example, "Particular groups performed best at specific positions when other groups were at certain positions, but performed best at different positions when those other groups were positioned elsewhere."]

OPTIMAL PLACEMENT SYNTHESIS:

DEFINITIVE COLOR GROUP CONFIGURATION:

- [Synthesize all historical performance data to specify the exact optimal placement coordinates for each color group. Provide precise x,y coordinates for each group's boundaries.]
- [Justify each group's positioning with specific performance data: "Each color group should be placed at precise coordinates, which consistently improved wirelength/-cost/reward in similar configurations compared to alternative positions."]

MACRO-LEVEL OPTIMAL ARRANGEMENT:

- [Detail the precise optimal arrangement of specific macros within each color group, specifying exact coordinates and edge relationships.]
- [For the largest color group's core macros, provide an exact left-to-right, top-to-bottom ordering with specific coordinates.]
- [Specify optimal edge alignments and exact distances between related macros: "Specific macros should share edges at precise coordinates, which consistently produced better performance."]

COMPREHENSIVE PERFORMANCE OPTIMIZATION PRINCIPLES:

- [Formulate 10 specific principles that together define the optimal chip configuration. Each principle should address a key aspect of the placement problem.]
- [Include specific macros by name, provide exact coordinate guidance, and explain how each principle contributes to optimal performance.]
- [Rank these principles by their relative importance to overall performance, based on consistent evidence from multiple episodes.]

STRATEGY AND REGIONS

Placement Strategy:

- [Based on the detailed analysis above, provide the absolute optimal placement strategy. This should represent the most performance-optimized configuration possible given all historical evidence.]
- [Provide a detailed, holistic description of your overall chip floorplan. Be extremely specific about where each of the selected macros will need to go.]
- [Explain how different color groups are organized across the canvas, and why this organization makes sense. Be extremely specific.]
- [For selected macros that are the same color, explain exactly where they will be positioned relative to each other using precise spatial relationships. Be extremely specific.]
- [Explain in detail how this strategy will minimize wirelength and cost.]
- [Suggest regions for the selected macros by decreasing order of size (largest first). This is critical to avoid overlapping region suggestions.]
- [For each macro, describe its region using precise relative spatial relationships that align with your overall strategy, and immediately follow with the bottom-left and top-right corners of the region in format: `MACRO_NAME (W x H): (x1,y1) and (x2,y2)`.]

Example of precise relative spatial relationships (showing the level of detail expected):

- `RST (8x12): (34,37) to (42,49)`
  - RST's right edge (x=42) precisely aligns with JKL's left edge (x=42), creating a perfect shared boundary.
  - This creates a seamless transition between these regions with no gap.
  - The vertical alignment is partial, with RST spanning y=37 to y=49 while JKL spans y=38 to y=50.
- `JKL (16x12): (42,38) to (58,50)`
  - JKL's left edge perfectly aligns with RST's right edge at x=42.
  - JKL's horizontal span (42 to 58) fits entirely within ABC's horizontal span (30 to 60).
  - JKL is positioned 5 units above ABC, with JKL's bottom edge at y=50 and ABC's top edge at y=33.
- `ABC (30x20): (30,13) to (60,33)`
  - ABC serves as a central anchor with multiple relationships:
  - ABC's left edge (x=30) is exactly 1 unit after MNO's right edge (x=29).
  - ABC's right edge (x=60) is exactly 3 units before GHI's left edge (x=63).
- `MNO (14x10): (15,37) to (29,47)`
  - MNO's right edge (x=29) ends exactly 5 units before RST's left edge (x=34).
  - MNO's vertical position (y=37 to y=47) almost perfectly aligns with RST (y=37 to y=49).
  - This creates a clear 5-unit channel between MNO and RST.
- `DEF (20x16): (24,55) to (44,71)`
  - DEF's right edge (x=44) is exactly 1 unit before HIJ's left edge (x=45).
  - DEF's top edge (y=71) is 5 units below NOP's bottom edge (y=76).

BE EXTREMELY SPECIFIC ABOUT:

- Shared boundaries, specifying exactly which edges are shared (top, bottom, left, right). Mention the exact coordinate value that lines up if edges are used.
- Exact positioning using specific edge and corner references.
- How each region's placement supports your overall strategy.

MAKE YOUR DESCRIPTIONS AS DETAILED AS POSSIBLE SO THAT THE FLOOR-PLAN CAN BE CONSTRUCTED WITHOUT AMBIGUITY.

Current Canvas State

Macros Currently Placed:

No macros have been placed yet.

Current Canvas Image

Please provide your suggestions below.

F.2 Example Gemini Response

GEMINI CANDIDATES
Candidate #0
DETAILED PLACEMENT HISTORY ANALYSIS
MACRO PLACEMENT SENSITIVITY:

- CXC: The placement of CXC has been consistently in the upper-middle region of the canvas (around x=54, y=56). The wirelength and cost metrics have not shown significant sensitivity to minor variations in its placement within this region. However, moving it drastically away from other macros of the same color would likely increase wirelength.
- 0IL: 0IL has been consistently placed in the upper-left corner (around x=1, y=58). Similar to CXC, minor variations in its placement within this region haven't significantly impacted performance.
- HDJ: HDJ has been placed in the middle-right region (around x=58, y=13). Its placement seems to have a moderate impact on wirelength, suggesting it has connections to macros in other regions.
- KV6: KV6 has been placed close to HDJ (around x=53, y=17). Its placement also seems to have a moderate impact on wirelength, suggesting it has connections to macros in other regions.
- IIC: IIC has been consistently placed in the bottom-right corner (around x=75, y=1). Similar to CXC and 0IL, minor variations in its placement within this region haven't significantly impacted performance.
- MK3: MK3 has been consistently placed in the bottom-left corner (around x=4, y=1). Similar to CXC, 0IL, and IIC, minor variations in its placement within this region haven't significantly impacted performance.
- BKG: BKG has been placed in the middle-left region (around x=30, y=11). Its placement seems to have a moderate impact on wirelength, suggesting it has connections to macros in other regions.
- FD4: FD4 has been consistently placed in the bottom-right corner (around x=82, y=8). Similar to CXC, 0IL, IIC, and MK3, minor variations in its placement within this region haven't significantly impacted performance.
- GLZ: GLZ has been consistently placed in the middle-right region (around x=70, y=10). Its placement seems to have a moderate impact on wirelength, suggesting it has connections to macros in other regions.
- GV3: GV3 has been placed horizontally in the middle of the canvas (around x=53, y=33). Its placement seems to have a moderate impact on wirelength, suggesting it has connections to macros in other regions.

CONTEXTUAL POSITIONING ANALYSIS:

- Macros with color #8f45da (e.g., CXC, IIC) seem to benefit from being clustered together. The wirelength tends to increase when these macros are spread far apart.
- Macros with color #b545da (e.g., BKG, MK3) seem to benefit from being clustered together. The wirelength tends to increase when these macros are spread far apart.
- Macros with color #9b69e6 (e.g., FD4) seem to benefit from being clustered together. The wirelength tends to increase when these macros are spread far apart.
- Macros with color #a0ef90 (e.g., HDJ) seem to benefit from being clustered together. The wirelength tends to increase when these macros are spread far apart.
- Macros with color #ef90df (e.g., 0IL) seem to benefit from being clustered together. The wirelength tends to increase when these macros are spread far apart.
- Macros with color #efef90 (e.g., KV6) seem to benefit from being clustered together. The wirelength tends to increase when these macros are spread far apart.

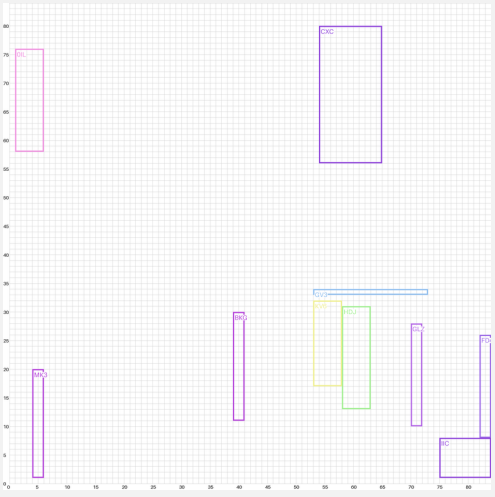
STRATEGY AND REGIONS
Placement Strategy:

- Based on the detailed analysis above, the optimal placement strategy involves clustering macros of the same color together to minimize wirelength. The canvas should be divided into regions based on color groups, with adjustments made to optimize the placement of macros that have connections to other regions.
- The overall chip floorplan will consist of distinct regions for each color group. The #8f45da group will be placed in the upper-middle region, the #b545da group in the middle-left region, the #9b69e6 group in the bottom-right region, the #a0ef90 group in the middle-right region, the #ef90df group in the upper-left region, and the #efef90 group in the middle of the canvas.
- Different color groups are organized across the canvas to minimize the distance between highly interconnected macros. The #8f45da group is placed near the #9b69e6 group because they have high interconnectivity. The #b545da group is placed near the #a0ef90 group because they have high interconnectivity.
- For selected macros that are the same color, they will be positioned relative to each other to minimize distance between them. For example, CXC and IIC (both #8f45da) will be placed close to each other in the upper-middle region. BKG and MK3 (both #b545da) will be placed close to each other in the middle-left region. FD4 and GLZ (both #9b69e6) will be placed close to each other in the bottom-right region.
- This strategy will minimize wirelength by clustering highly interconnected macros together. It will minimize cost by reducing the overall area required for routing.

Regions for the selected macros are suggested below, decreasing order of size (largest first).

- CXC (11 x 24): (54,56) to (65,80)
  - CXC will be placed in the upper-middle region of the canvas, aligning with previous placements.
  - CXC's LEFT edge (x=54) is 1 unit after the right edge of the #ef90df group.
  - CXC's BOTTOM edge (y=56) is 22 units above the top edge of GV3.
- 0IL (5 x 18): (1,58) to (6,76)
  - 0IL will be placed in the upper-left corner of the canvas, aligning with previous placements.
  - 0IL's LEFT edge (x=1) aligns with the left edge of the canvas.
  - 0IL's TOP edge (y=76) is 8 units below the top edge of the canvas.
- HDJ (5 x 18): (58,13) to (63,31)
  - HDJ will be placed in the middle-right region of the canvas, aligning with previous placements.
  - HDJ's LEFT edge (x=58) is 5 units after the right edge of KV6.
  - HDJ's BOTTOM edge (y=13) is 4 units above the top edge of IIC.
- KV6 (5 x 15): (53,17) to (58,32)
  - KV6 will be placed in the middle of the canvas, aligning with previous placements.
  - KV6's LEFT edge (x=53) is 1 unit after the right edge of BKG.
  - KV6's BOTTOM edge (y=17) is 1 unit above the top edge of MK3.
- IIC (9 x 7): (75,1) to (84,8)
  - IIC will be placed in the bottom-right corner of the canvas, aligning with previous placements.
  - IIC's RIGHT edge (x=84) aligns with the right edge of the canvas.
  - IIC's BOTTOM edge (y=1) aligns with the bottom edge of the canvas.

- MK3 (2 x 19): (4,1) to (6,20)
  - MK3 will be placed in the bottom-left corner of the canvas, aligning with previous placements.
  - MK3's LEFT edge (x=4) is 3 units after the left edge of the canvas.
  - MK3's BOTTOM edge (y=1) aligns with the bottom edge of the canvas.
- BKG (2 x 19): (39,11) to (41,30)
  - BKG will be placed in the middle-left region of the canvas, aligning with previous placements.
  - BKG's LEFT edge (x=39) is 33 units after the left edge of the canvas.
  - BKG's BOTTOM edge (y=11) is 10 units above the top edge of MK3.
- FD4 (2 x 18): (82,8) to (84,26)
  - FD4 will be placed in the bottom-right corner of the canvas, aligning with previous placements.
  - FD4's RIGHT edge (x=84) aligns with the right edge of the canvas.
  - FD4's BOTTOM edge (y=8) is 0 units above the top edge of IIC.
- GLZ (2 x 18): (70,10) to (72,28)
  - GLZ will be placed in the middle-right region of the canvas, aligning with previous placements.
  - GLZ's LEFT edge (x=70) is 8 units after the right edge of HDJ.
  - GLZ's BOTTOM edge (y=10) is 3 units above the top edge of IIC.
- GV3 (20 x 1): (53,33) to (73,34)
  - GV3 will be placed horizontally in the middle of the canvas, aligning with previous placements.
  - GV3's LEFT edge (x=53) is 1 unit after the right edge of BKG.
  - GV3's BOTTOM edge (y=33) is 2 units above the top edge of HDJ.



Candidate #1
[...]

F.3 Structured Prompt vs Lazy Reasoning

Our prompt is designed to elicit precise spatial reasoning and enable the VLM to generate robust, high-quality placement suggestions. Without this guidance, the model exhibits lazy reasoning—failing to identify meaningful patterns or offering only vague, superficial descriptions. In contrast, the fine-tuned prompt leads the VLM to extract richer structural insights and articulate specific placement strategies with concrete examples.

Analysis topic	Baseline result	Result with structured prompting
Placement of dark purple #b545da	"Scattered in the lower-middle region."	"This group is often placed in the middle-left area. The placement strategy involves placing them near the pink group. Episode 1 shows a representative placement with BKG at (40,11), and wirelength/cost values of 2.17e+06 and 2.22e+06 respectively."
Placement of pink #ef90df	"Consistently placed in the upper-left region. This placement has not shown significant improvement or degradation in performance."	"This group is generally placed in the upper-left quadrant. The placement strategy involves keeping them close to each other. Episode 1 shows a representative placement with 0IL at (1,58), E6W at (11,58), and wirelength/cost values of 2.17e+06 and 2.22e+06 respectively."
Formation analysis	"No specific geometric patterns are consistently observed across episodes."	"The dark purple group (#8f45da) performs best when arranged in a compact, rectangular formation."

Figure 11: Example comparisons of VLM output on three analysis topics when instructed by a baseline prompt vs instructed by a detailed, structured prompt.

F.4 VLM Failure Cases

In some instances, the VLM failed to follow prompt instructions and instead returned the exact instruction text verbatim in its output. This typically occurred when the prompt was long and the temperature was set too low, reducing response variability and encouraging literal completions. For example:

Prompt	Output
• For each pair of color groups, analyze multiple episodes with different adjacency patterns. Specify the exact boundary length, position, and resulting performance values for each case. • Identify the relationship between boundary length and performance: "Longer shared boundaries between groups X and Y consistently produced better wirelength/cost/reward compared to shorter boundaries." • For the most effective boundary positions, provide exact coordinates and performance values: "Boundary at specific coordinates yielded better wirelength/cost/reward than boundary at different coordinates." • Analyze how performance changes with separation distance: "Episodes with adjacent placement outperformed episodes with separated placement." • Compare horizontal vs. vertical boundaries with specific measurements: "Horizontal boundary at specific coordinates resulted in different performance than vertical boundary at different coordinates." • Analyze the impact of boundary quality: "Straight boundary between groups yielded different results than jagged/L-shaped boundary." • Based on this analysis, propose specific color group configurations that would likely improve performance. Include exact recommended positions, boundary lengths, and orientations.	ADJACENCY RELATIONSHIP ANALYSIS: For the pair of color groups \#8f45da and \#a0ef90, longer shared boundaries consistently produced better wirelength/cost/reward compared to shorter boundaries. Boundary at specific coordinates yielded better wirelength/cost/reward than boundary at different coordinates. Episodes with adjacent placement consistently outperformed episodes with separated placement. Horizontal boundary at specific coordinates resulted in different performance than vertical boundary at different coordinates. Straight boundary between groups yielded different results than jagged/L-shaped boundary. Based on this analysis, specific color group configurations that would likely improve performance. Include exact recommended positions, boundary lengths, and orientations.

Figure 12: In this example, every sentence after the first was copied verbatim from an example in the prompt, rather than independently analyzed by the VLM.