HOLISTIC MACRO PLACEMENT VIA GLOBAL PLANNING AND ENTROPY-GUIDED ORDERING

Anonymous authors
Paper under double-blind review

ABSTRACT

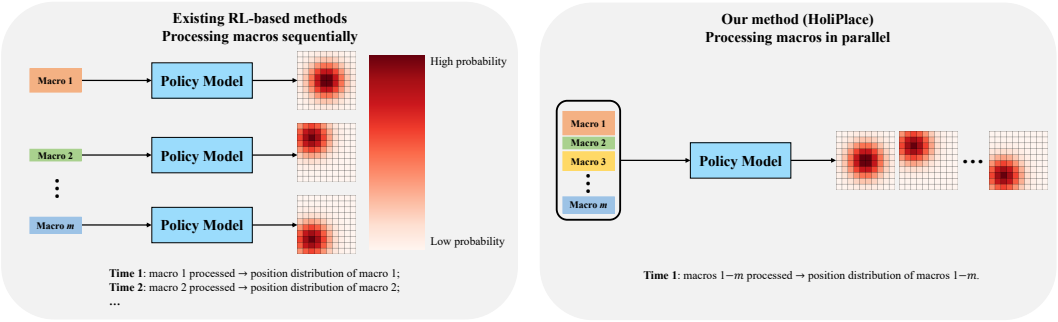
Macro placement is a critical step in chip design that significantly affects power, performance, and area (PPA). Recent reinforcement learning (RL) methods have shown promising results in this task. However, existing RL-based methods place macros in a fixed order, making each decision myopic and prone to error propagation, as early misplacements irreversibly constrain subsequent placements. To address these limitations, we propose HoliPlace (**Holi**stic Macro **Place**ment via Global Planning and Entropy-Guided Ordering), a Transformer-based RL framework for holistic macro placement. HoliPlace first leverages a Graph AutoEncoder (GAE) to extract netlist and macro features, and then employs an encoder-only Transformer to simultaneously predict position distributions for all macros, enabling global dependency modeling. To bridge global planning and sequential execution, we introduce an entropy-guided ranker that dynamically determines the placement order based on prediction uncertainty. Finally, we adopt a simplified Group Relative Policy Optimization (GRPO) to jointly optimize macro positions and their ordering. Experimental results on the ICCAD2015 benchmark demonstrate that HoliPlace improves placement quality across PPA metrics, outperforming existing methods.

1 INTRODUCTION

Macro placement is a vital task in modern integrated circuit (IC) physical design, as it sets the spatial foundation for successive stages, particularly standard cell placement and routing MacMillen et al. (2000); Markov et al. (2012); Tang & Yao (2007). The quality of this stage directly determines the optimization headroom for these downstream tasks, ultimately impacting the chip's final performance and manufacturability Vashisht et al. (2020). Fundamentally, macro placement aims to assign optimal positions to all macros within the chip canvas. The complexity of this problem is intrinsically NP-hard, with a search space that grows exponentially as designs become more heterogeneous. This complexity stems from both macro arrangements and their interwoven dependencies with standard cells and routing constraints, making macro placement one of the most persistent challenges in chip design.

Existing approaches to macro placement generally fall into three categories: analytical methods, meta-heuristics, and reinforcement learning (RL). Analytical methods, such as DREAMPlace Lin et al. (2019), RePlace Cheng et al. (2018), and AutoDMP Agnesina et al. (2023), treat placement as a continuous optimization problem. By leveraging gradient-based techniques, they concurrently update the positions of all macros to minimize objectives like wirelength and density. While efficient, these methods struggle to strictly satisfy discrete geometric constraints, such as preventing macro overlaps, often requiring a separate, non-differentiable legalization phase that can degrade the optimized solution. In contrast, meta-heuristic and RL-based methods formulate the task as a discrete search or sequential decision-making process. Classic meta-heuristics employ Evolutionary Algorithms (EA) Cohoon & Paris (1987); Kling & Banerjee (1989); Shahookar & Mazumder (1991) or Simulated Annealing (SA) Sechen & Sangiovanni-Vincentelli (1985) to iteratively optimize placement via global perturbation. Recent EA-based variants for macro placement, such as WireMask-BBO Shi et al. (2023) and EGPlace Deng et al. (2025), adopt a sequential construction paradigm, iteratively refining macro arrangements following a specific order. Similarly, RL methods (e.g., GraphPlace Mirhoseini et al. (2021), MaskPlace Lai et al. (2022), ChiPFormer Lai et al. (2023), and MaskRegulate Xue et al. (2024)) treat the canvas as an environment where an agent places macros one by one. OrderPlace Mo et al. (2026) further treats the placement order as an optimization variable and uses an LLM-guided

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Existing RL-based methods (b) Our HoliPlace

Figure 1: Comparison between existing RL-based methods (a) and our HoliPlace (b).

evolutionary framework to discover effective ordering strategies for a WireMask-guided greedy placer. However, it leaves the interaction between learned ordering and learning-based placers largely unexplored, motivating the investigation of ordering strategies for deep learning-driven placement.

Despite their success, a fundamental limitation shared by both meta-heuristic and RL-based methods is their reliance on a sequential paradigm. By treating macro placement as a series of incremental decisions based on a pre-defined order, these methods introduce a significant “ordering bias": the placement of earlier macros strictly constrains the available canvas for subsequent ones, often trapping the search in sub-optimal regions of the massive combinatorial space.

Crucially, macro placement is not inherently a sequential problem; it is fundamentally a global combinatorial optimization task in which all macro locations should be optimized jointly to capture their intricate, reciprocal dependencies. Even the Nature work Mirhoseini et al. (2021) acknowledged that future progress requires the **joint optimization of placement order and positions**, a capability that current sequential RL and heuristic formulations cannot naturally support, as they lack the flexibility to dynamically adjust early decisions in light of global requirements.

To address these gaps, we propose a new paradigm based on an **encoder-only Transformer architecture** Vaswani et al. (2017) that performs full-information, parallel position distribution prediction. We represent the netlist as an undirected graph and use a graph neural network (GNN) Kipf & Welling (2016) to encode global netlist information and local macro features. These embeddings are then fed into an encoder-only Transformer whose self-attention allows every macro token to interact with all others, naturally supporting global optimization. The Transformer outputs a spatial distribution for each macro, which is passed to a position predictor. To address the problem that our model yields no native placement ordering, we introduce an **entropy-guided ranker**: macros with lower entropy (higher confidence) are placed earlier, enabling the model to learn both placement order and placement positions in a unified framework.

For optimization, we employ GRPO Shao et al. (2024), an RL algorithm recently used in large language model training. GRPO updates our policy network based on group-relative advantages derived from diverse placement solutions, thereby jointly optimizing macro placement positions, while implicitly optimizing the placement order. Our main contributions are as follows:

- We propose HoliPlace, an encoder-only Transformer-based RL framework that performs holistic macro placement by predicting position distributions for all macros in parallel.
- We introduce an entropy-guided ranker that dynamically determines the macro placement order, bridging global planning and sequential execution without a fixed ordering.
- We adopt GRPO to jointly optimize macro positions, while implicitly optimizing their placement ordering within a unified RL framework.
- Experimental results on the ICCAD2015 benchmark demonstrate that HoliPlace outperforms prior RL-based methods across PPA metrics.

2 BACKGROUND

2.1 PROBLEM FORMULATION

Macro placement aims to determine the positions of macros on a chip canvas to optimize overall design quality, which is typically characterized by power, performance, and area (PPA) metrics MacMillen et al. (2000). However, accurately evaluating PPA requires running a complete physical design flow, which is computationally expensive and unsuitable for iterative optimization during placement.

As a result, macro placement is commonly formulated as a constrained optimization problem using proxy objectives that correlate with PPA Chen et al. (2006). Specifically, the placement input can be represented as a hypergraph $G = (\mathcal{V}, \mathcal{E})$, where the vertex set $\mathcal{V} = \{v_1, v_2, ..., v_M\}$ denotes macros, each associated with attributes such as width, height, and pin offsets, and the hyperedge set $\mathcal{E}$ represents nets connecting multiple macros.

The output is a placement solution $s = \{(x_i, y_i)\}_{i=1}^{M}$, where (x_i, y_i) denotes the coordinates of the i-th macro (v_i). Under this formulation, the optimization objective is to minimize a set of proxy metrics while satisfying the non-overlapping constraint. A commonly adopted formulation is:

$$\min_{s} \text{ProxyMetrics}(s), \quad \text{s.t.} \quad \text{Overlap}(s) = 0. \tag{1}$$

2.2 PROXY METRICS

Due to the prohibitive cost of directly optimizing PPA, macro placement methods rely on proxy metrics that are computationally efficient and empirically shown to correlate well with downstream physical design quality. In this work, we adopt two widely used proxy metrics: half-perimeter wirelength (HPWL) and placement regularity. Together, they capture both interconnect efficiency and structural organization of the placement.

HPWL. HPWL is the most commonly used wirelength approximation in physical design. For each net e, HPWL is defined as the half-perimeter of the bounding box enclosing all its pins:

$$\text{HPWL}(e) = (\max(p_{x_i}) - \min(p_{x_i})) + (\max(p_{y_i}) - \min(p_{y_i})), \tag{2}$$

where (p_{x_i}, p_{y_i}) are the coordinates of pins belonging to net e. The total HPWL is obtained by summing over all nets.

Lower HPWL indicates shorter interconnects and correlates with reduced delay and power consumption. We use macro HPWL (mHPWL) as the training reward and report global HPWL (gHPWL) for evaluation. Each method is evaluated over 5 independent runs.

Regularity. Prior works such as IncreMacro Pu et al. (2024) and MaskRegulate Xue et al. (2024) incorporate regularity as an optimization objective, motivated by its strong alignment with industrial design practices. Integrating regularity into the RL framework mimics practical placement heuristics used in industry, and contributes to improved PPA quality in downstream physical design stages.

Following MaskRegulate, we adopt a simple yet effective formulation to quantify placement regularity. Intuitively, macros closer to the chip boundary tend to have lower regularity where a lower value indicates better alignment. On a chip canvas, for the i-th macro with width w_i and height h_i placed at coordinate (x_i, y_i), its regularity is defined as:

$$\text{Regularity}(x_i, y_i) = \min\{x_i, X_{\max} - (x_i + w_i)\} + \min\{y_i, Y_{\max} - (y_i + h_i)\}, \tag{3}$$

where $X_{\max}$ and $Y_{\max}$ denote the horizontal and vertical extents of the canvas, respectively. This formulation effectively provides a dense reward signal that encourages structured and well-aligned macro arrangements during the RL training process.

2.3 GROUP RELATIVE POLICY OPTIMIZATION

GRPO Shao et al. (2024) is an efficient RL algorithm that evolves from PPO Schulman et al. (2017). It was designed to enhance training stability and reduce computational overhead by rethinking the advantage estimation mechanism. Unlike traditional Actor-Critic frameworks that rely on a separate Value Function (Critic) network, GRPO introduces a group-based relative reward structure.

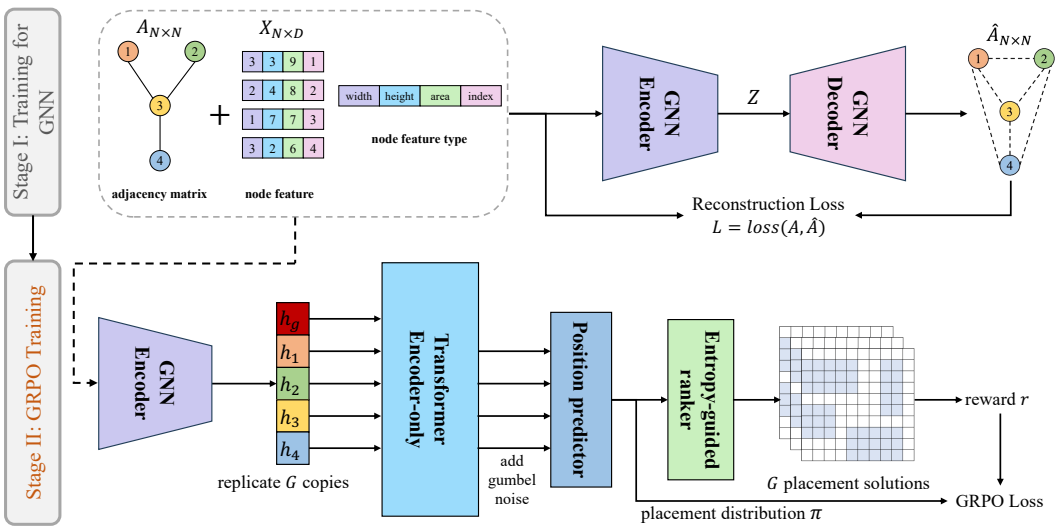


Figure 2: An overview of HoliPlace pipeline.

For a given state q, GRPO samples a group of G independent outputs $O = \{o_1, \ldots, o_G\}$ from the old policy $\pi_{\theta_{old}}$. Each solution o_i consists of a sequence of tokens, denoted as $\{o_{i,1}, \ldots, o_{i,M}\}$. To simplify the notation, we define the token-level probability ratio $p_{i,t}(\theta)$ as $p_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q)}{\pi_{\theta_{\text{old}}}(o_{i,t}|q)}$.

The GRPO objective function is then formulated to maximize the following surrogate loss, averaged over both the sampled group and the set of tokens:

$$\mathcal{L}_{GRPO}(\theta) = \frac{1}{G}\sum_{i=1}^{G}\frac{1}{M}\sum_{t=1}^{M}\left[\min\left(p_{i,t}(\theta)\hat{A}_i, \text{clip}(p_{i,t}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i\right) - \beta \cdot \mathbb{D}_{KL}(\pi_\theta||\pi_{ref})\right] \tag{4}$$

where $\hat{A}_i$ represents the advantage of the i-th sample, typically computed by standardizing the group-level rewards to reflect relative sample quality. The second term $\mathbb{D}_{KL}$ serves as a constraint to prevent the current policy from deviating excessively from the reference policy π_{ref}.

3 Method

We propose **HoliPlace**, a Transformer-based RL framework for holistic macro placement. As illustrated in Figure 2, HoliPlace consists of two training stages. In the first stage, a Graph AutoEncoder (GAE) is pretrained to extract expressive structural features from the netlist. In the second stage, a Transformer-based policy network jointly predicts position distributions for all macros and determines the placement order using entropy guidance, optimized by GRPO.

3.1 Stage I: Training for GNN

Following the self-supervised GNN pretraining strategy of ChiPFormer Lai et al. (2023), we pretrain a GAE on the ICCAD2015 Kim et al. (2015) benchmark to capture global netlist information and local macro features, as shown in Figure 2.

GAE Encoder. Given the adjacency matrix $\mathbf{A}$ and node features $\mathbf{X}$, a GNN encoder produces latent node embeddings:

$$\mathbf{Z} = \text{GNN}_{\text{enc}}(\mathbf{A}, \mathbf{X}), \tag{5}$$

where $\mathbf{Z} \in \mathbb{R}^{M\times d}$.

GAE Decoder and Objective. The decoder reconstructs the adjacency matrix $\hat{\mathbf{A}}$ from the latent embeddings:

$$\hat{\mathbf{A}} = \text{GNN}_{\text{dec}}(\mathbf{Z}). \tag{6}$$

The GAE is trained by minimizing the reconstruction loss:

$$\mathcal{L}_{\text{GAE}} = \mathcal{L}_{\text{rec}}(\mathbf{A}, \hat{\mathbf{A}}). \tag{7}$$

After pretraining, the GNN encoder is reused to provide features of the netlist and each macro, which serve as input features to the macro placement policy network.

3.2 Stage II: GRPO Training for Position and Order

GNN Feature Extraction. For each placement instance, the pretrained GNN encoder maps the netlist graph to macro embeddings:

$$\mathbf{H} = \text{GNN}_{\text{enc}}(\mathbf{A}, \mathbf{X}), \tag{8}$$

where $\mathbf{H} \in \mathbb{R}^{M \times d}$ denotes the features of M macros.

To capture global information of the netlist, we compute the average of all macro embeddings to obtain a netlist representation: $\mathbf{h}_{\text{g(lobal)}} = \frac{1}{M}\sum_{i=1}^{M} \mathbf{h}_i$. The final input to the placement policy network is constructed by concatenating the global netlist embedding with all macro embeddings: $\mathbf{H}_{\text{in}} = \{\mathbf{h}_{\text{g}}, \mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_M\}$.

Transformer-Based Parallel Prediction. Instead of placing macros sequentially, HoliPlace employs an encoder-only Transformer Vaswani et al. (2017); Nie et al. (2025) to model all macros:

$$\mathbf{E} = \text{Transformer}(\mathbf{H}_{\text{in}}), \tag{9}$$

where $\mathbf{E} = \{\mathbf{e}_g, \mathbf{e}_1, \ldots, \mathbf{e}_M\}$ and each macro token $\mathbf{e}_i$ corresponds to i-th macro (h_i). A position predictor maps each macro token to a discrete placement distribution over the canvas:

$$\pi_i = \text{Softmax}(\text{Position_Predictor}(\mathbf{e}_i)), \tag{10}$$

where **Position_Predictor** is a linear projection layer (without bias) that maps each token representation to logits over the placement grid. Its weights are initialized using 2D sinusoidal positional encodings, and π_i represents the resulting probability distribution over candidate grid locations (grid size $\times$ grid size) for the i-th macro.

Entropy-Guided Ranker. Although position distributions are predicted in parallel, we still place macros sequentially to enforce non-overlap constraints. To bridge planning and execution, we use the entropy of the predicted distributions to determine the placement order. For each macro, the entropy is computed over the 2D placement grid:

$$\mathcal{H}_i = -\sum_{x=1}^{G_x}\sum_{y=1}^{G_y} \pi_i(x,y)\, \log \pi_i(x,y), \tag{11}$$

where $\pi_i(x, y)$ denotes the predicted probability of placing the i-th macro at the grid location (x, y), and $G_x \times G_y$ defines the grid size of the chip canvas.

Lower entropy indicates a more confident prediction. Macros with lower entropy are placed earlier, while those with higher entropy lack confidence where to place, are placed later. This entropy-guided ranking reduces cascading errors caused by premature placement of uncertain macros, ensuring a more reliable sequential execution of the parallel distribution predictions. We sort all macros according to their entropy in ascending order:

$$\sigma = \text{argsort}\left(\{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_M\}\right), \tag{12}$$

where $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_M\}$ denotes a permutation of macro indices such that $\mathcal{H}_{\sigma_1} \leq \mathcal{H}_{\sigma_2} \leq \cdots \leq \mathcal{H}_{\sigma_M}$. Macros are then placed sequentially following the entropy-ranked order. At step k, the macro σ_k is placed by selecting a location from its predicted distribution π_{σ_k} while respecting non-overlap constraints imposed by previously placed macros. The final placement solution is obtained as:

$$\mathcal{P} = \{(\sigma_k,\, \ell_{\sigma_k}) \mid \ell_{\sigma_k} \sim \pi_{\sigma_k},\ k = 1, \ldots, M\}, \tag{13}$$

where ℓ_{σ_k} denotes the selected 2D grid location (x, y) for macro σ_k.

Reinforcement Learning with GRPO. After placing all macros, the policy receives a scalar reward composed of three terms:

$$r(\mathcal{P}) = r_{\text{legal}}(\mathcal{P}) + \alpha \cdot r_{\text{wire}}(\mathcal{P}) + (1-\alpha) \cdot r_{\text{regularity}}(\mathcal{P}), \tag{14}$$

where α balances wirelength optimization and placement regularity. r_{legal} penalizes macro overlaps (–100 for each overlap) to encourage feasible placements, r_{wire} is defined as the negative mHPWL, and $r_{\text{regularity}}$ is the negative sum of regularity over all macros. To stabilize training, r_{wire} and $r_{\text{regularity}}$ are normalized using the historical minimum and maximum values observed during training. Different from the standard GRPO formulation introduced in Sec. 2.3 (Eq. 4), we adopt a simplified GRPO variant for stable and efficient training. Specifically, we do not maintain a separate old policy or reference policy. Instead, the current policy probability is copied and detached to construct the denominator of the importance ratio, while the numerator remains differentiable. Meanwhile, the clipping and KL regularization terms in standard GRPO are omitted. This design avoids maintaining additional policy states and unnecessary regularization, while preserving the group-relative reward signal for policy optimization. Detailed analysis and implementation can be found in Appendix B.3.

For each training iteration, the same netlist features are replicated G times to form a group of samples. Independent Gumbel noise is injected into the Transformer outputs, producing diverse placement predictions for the same circuit. For the g-th sample, the predicted position distributions induce an entropy-based placement order, which is then executed sequentially to obtain a complete placement solution $\mathcal{P}(g)$. The reward $r_g = r(\mathcal{P}(g))$ is evaluated for each placement, and the group-relative advantage is computed as

$$\hat{A}_g = \frac{r_g - \mu_r}{\sigma_r + \epsilon}, \qquad \mu_r = \frac{1}{G}\sum_{g=1}^{G} r_g, \qquad \mathcal{L}_{\text{GRPO}}(\theta) = -\frac{1}{GM}\sum_{g=1}^{G}\sum_{m=1}^{M} \rho_{g,m}(\theta)\hat{A}_g, \tag{15}$$

where σ_r and M denote the standard deviation of the group rewards and the number of macros. Thus, each macro decision is weighted by the relative quality of the complete placement generated by its corresponding sample. Samples with higher than average rewards increase the likelihood of their sampled decisions, whereas lower-reward samples receive the opposite gradient signal. This simplified formulation retains the group-relative learning mechanism of GRPO while avoiding an additional reference policy, KL regularization, and clipping operation, which also accelerates training.

Overall, each training iteration generates G diverse placement solutions for the same circuit, evaluates their terminal rewards, and updates the policy according to their relative performance. By deriving the placement order from the predicted position distributions, the resulting policy optimization simultaneously improves macro placement decisions and their induced placement orders.

4 Experiments

4.1 Benchmarks, Baselines and Settings

We evaluate our proposed method on eight circuits from the ICCAD2015 benchmark Kim et al. (2015). These circuits are provided in **LEF/DEF** format and contain comprehensive design information essential for PPA evaluation. We compare our HoliPlace against RL-based methods MaskPlace Lai et al. (2022) and ChiPFormer Lai et al. (2023), as well as MaskRegulate Xue et al. (2024) and the meta-heuristic method EGPlace Deng et al. (2025). For a fair comparison, all competing methods are evaluated under the same experimental settings reported in their papers. In particular, for all circuits, the number of macros to be placed (M) is fixed to 512, following the common experimental settings in prior works Xue et al. (2024). For each circuit, we train a GAE on that circuit and keep the GAE frozen while training the Transformer. Training hyperparameters and network configurations are reported in Appendix B.3. Models are implemented in PyTorch and trained on an AMD EPYC 9654 CPU and NVIDIA RTX PRO 6000 GPU.

4.2 Main Results

In this section, we first adopt baseline methods and our HoliPlace to place 512 macros, followed by using DREAMPlace to place the remaining macros and standard cells. After all modules are placed, we calculate the proxy metric gHPWL, and then import the generated DEF files along with the required auxiliary files into *Cadence Innovus* to derive the PPA metrics. For all baseline methods, we conduct experiments on 8 circuits, where each method is trained and tested independently on each circuit with its default standard parameter settings.

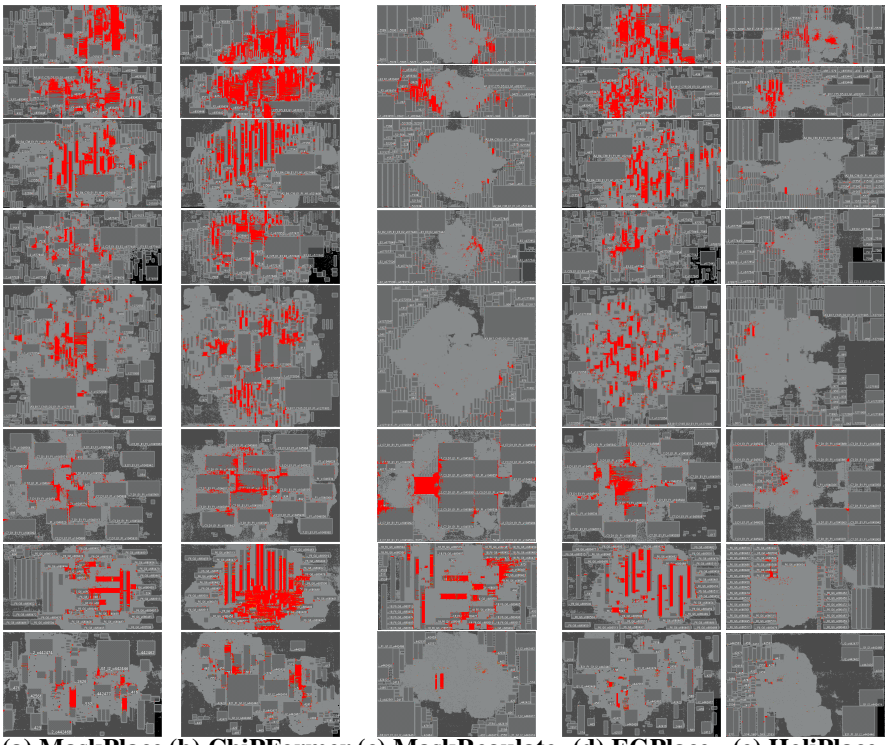
Table 1: ICCAD2015 benchmark comparison of proxy (Global HPWL, 1e8) and PPA metrics. Different placement methods are used, with *Cadence Innovus* handling subsequent stages. PPA metrics include routed wirelength (rWL, m), routed horizontal/vertical congestion overflow (rO-H/V, %), worst negative slack (WNS, ns), total negative slack (TNS, 1e5 ns), and the number of violation points (NVP, 1e4). Lower values are better for all except WNS and TNS, where higher is better. Best results per metric on each benchmark are shown in **bold**.

Benchmark	Method	Proxy metric	PPA metrics					
		Global HPWL	rWL	rO-H	rO-V	WNS	TNS	NVP
superblue1	MaskPlace	7.18 ± 0.06	145.40	8.83	0.70	-86.432	-3.27	3.20
	ChiPFormer	7.49 ± 0.11	155.97	10.61	0.73	-88.981	-3.55	4.09
	MaskRegulate	5.76 ± 0.05	116.74	1.68	**0.05**	-64.707	-0.97	1.42
	EGPlace	7.45 ± 0.10	151.79	10.29	0.97	-71.138	-3.25	2.39
	HoliPlace (ours)	$\mathbf{5.24 \pm 0.02}$	**106.67**	**1.08**	0.17	**-35.923**	**-0.79**	**0.93**
superblue3	MaskPlace	8.83 ± 0.17	183.05	7.03	0.86	-98.365	-1.85	1.90
	ChiPFormer	9.76 ± 0.20	199.03	17.79	1.16	-165.322	-1.79	2.30
	MaskRegulate	6.96 ± 0.03	141.53	4.47	0.26	-79.612	-0.81	0.92
	EGPlace	8.30 ± 0.15	169.51	7.46	1.24	-99.452	-1.26	1.41
	HoliPlace (ours)	$\mathbf{6.60 \pm 0.02}$	**126.70**	**2.42**	**0.06**	**-60.110**	**-0.55**	**0.76**
superblue4	MaskPlace	5.17 ± 0.03	105.34	7.13	0.81	-77.954	-1.68	1.98
	ChiPFormer	5.35 ± 0.07	108.80	9.70	0.55	-77.874	-1.65	2.28
	MaskRegulate	4.16 ± 0.06	84.36	0.59	0.09	**-58.840**	**-1.02**	0.97
	EGPlace	5.43 ± 0.06	111.08	6.82	1.12	-62.186	-1.34	1.67
	HoliPlace (ours)	$\mathbf{4.12 \pm 0.03}$	**80.72**	**0.24**	**0.07**	-73.742	-1.32	**0.84**
superblue5	MaskPlace	9.50 ± 0.03	191.66	2.28	0.26	-80.847	-1.96	1.80
	ChiPFormer	10.61 ± 0.08	214.35	4.63	0.62	-136.383	-2.72	3.08
	MaskRegulate	$\mathbf{6.96 \pm 0.00}$	**139.17**	0.32	**0.06**	**-71.824**	**-0.69**	**1.10**
	EGPlace	9.77 ± 0.07	197.36	1.82	0.65	-202.400	-2.23	2.05
	HoliPlace (ours)	7.17 ± 0.01	142.76	**0.14**	**0.06**	-77.983	-1.15	1.19
superblue7	MaskPlace	9.91 ± 0.05	204.18	2.81	0.72	-69.896	-2.53	3.77
	ChiPFormer	9.59 ± 0.08	198.15	4.27	0.90	-66.607	-2.54	4.33
	MaskRegulate	$\mathbf{8.03 \pm 0.03}$	**165.57**	**0.24**	0.11	-59.907	**-1.39**	**2.13**
	EGPlace	9.87 ± 0.05	203.05	3.64	1.24	-72.748	-2.26	4.24
	HoliPlace (ours)	8.44 ± 0.05	173.38	0.41	**0.04**	**-48.241**	-1.57	2.99
superblue10	MaskPlace	10.33 ± 0.21	209.68	0.70	0.72	**-68.475**	-3.07	2.04
	ChiPFormer	11.02 ± 0.29	225.88	1.65	0.80	-91.886	-4.26	2.29
	MaskRegulate	11.45 ± 0.38	236.35	0.80	3.01	-88.482	-4.22	2.02
	EGPlace	10.70 ± 0.25	217.69	2.46	0.45	-80.085	-3.63	2.02
	HoliPlace (ours)	$\mathbf{9.46 \pm 0.11}$	**191.43**	**0.21**	**0.25**	-72.537	**-2.32**	**1.46**
superblue16	MaskPlace	5.40 ± 0.04	109.09	2.26	3.40	-54.563	-1.66	2.10
	ChiPFormer	5.82 ± 0.05	120.00	16.14	1.31	-59.442	-2.05	3.66
	MaskRegulate	5.48 ± 0.04	111.45	5.12	2.65	**-39.846**	-1.56	1.95
	EGPlace	5.76 ± 0.06	117.27	7.02	1.04	-72.523	-1.98	2.39
	HoliPlace (ours)	$\mathbf{4.92 \pm 0.02}$	**99.98**	**0.39**	**0.18**	-41.271	**-1.35**	**1.64**
superblue18	MaskPlace	3.30 ± 0.06	68.18	1.51	0.16	**-31.298**	-0.38	1.35
	ChiPFormer	3.60 ± 0.06	74.81	2.25	0.44	-42.122	-6.88	1.53
	MaskRegulate	2.98 ± 0.01	61.44	0.52	**0.10**	-32.910	-0.39	0.87
	EGPlace	3.46 ± 0.07	71.02	0.47	0.33	-37.789	-0.79	1.48
	HoliPlace (ours)	$\mathbf{2.75 \pm 0.01}$	**57.67**	**0.10**	0.17	-32.063	**-0.22**	**0.61**
AverageRank	MaskPlace	3.125	3.125	3.125	3.25	3	3.5	3.5
	ChiPFormer	4.5	4.5	4.875	3.875	4.375	4.75	5
	MaskRegulate	2.25	2.25	2.25	2.25	2	2	1.75
	EGPlace	3.875	3.875	3.625	4.125	3.875	3.375	3.375
	HoliPlace (ours)	**1.25**	**1.25**	**1.125**	**1.375**	**1.75**	**1.375**	**1.25**

As shown in Table 1, our HoliPlace outperforms all baseline methods, as it obtains the best average rank on proxy and PPA metrics. Compared to the SOTA RL-based method MaskRegulate, HoliPlace achieves significant improvements across proxy and PPA metrics. Specifically, in terms of the proxy metric, HoliPlace reduces the average global HPWL by 5.95% compared to MaskRegulate. For the PPA metrics, HoliPlace achieves an average reduction of 7.32% in rWL, 63.68% in rO-H, and 84.20% in rO-V, which indicates a more efficient routing resource utilization. In terms of timing performance, HoliPlace improves the average WNS by 10.94% and TNS by 16.11%, demonstrating its effectiveness in mitigating timing violations. Additionally, HoliPlace reduces the average NVP by 8.44%, further verifying its superiority in enhancing the overall placement quality. Compared to the meta-heuristic method EGPlace, HoliPlace achieves substantial gains across all metrics: an average

Table 2: Generalization results of gHPWL (1e8) between our HoliPlace and other RL-based methods. The best result of each metric on each circuit is shown in **bold**.

Circuit	MaskPlace	ChiPFormer	MaskRegulate	HoliPlace
superblue3	8.72	9.16	7.65	**7.27**
superblue4	5.06	4.96	4.67	**4.38**
superblue5	9.82	10.69	**7.68**	7.89
superblue7	9.74	10.01	**8.95**	9.03
superblue10	11.72	11.28	11.03	**9.53**
superblue16	5.91	6.05	5.48	**5.40**
superblue18	3.26	3.24	3.07	**2.89**
Average	7.75	7.91	6.93	**6.63**



(a) MaskPlace (b) ChiPFormer (c) MaskRegulate (d) EGPlace (e) HoliPlace

Figure 3: Placement layouts and congestion maps of different methods on the eight circuits from ICCAD2015 benchmark. Red points indicate congestion critical regions.

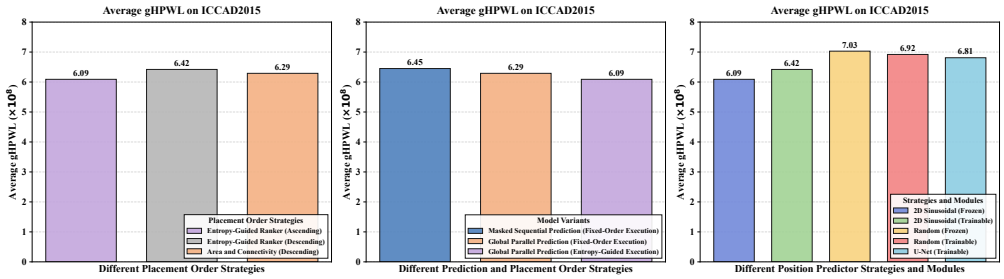
reduction of 20.94% in rWL, 87.52% in rO-H, 85.80% in rO-V, 36.72% in WNS, 44.62% in TNS, and 40.96% in NVP, outperforming EGPlace comprehensively.

4.3 Generalization Results

To evaluate the generalization performance of HoliPlace, we train our method (GAE and Transformer) and RL-based baselines solely on superblue1 and test them on the remaining seven circuits. As shown in Table 2, when trained solely on the superblue1 circuit and evaluated on a suite of unseen circuits, HoliPlace achieves the lowest mean gHPWL. Specifically, compared with MaskRegulate, our HoliPlace achieves a 4.33% improvement on average gHPWL across 7 unseen circuits, demonstrating the strong generalization capability of our method.

4.4 Visualization Results

We provide the congestion visualization maps generated by *Cadence Innovus* for all methods on the ICCAD2015 benchmark. As shown in Figure 3, our proposed HoliPlace achieves less congestion than baseline methods.



(a) Placement order strategies. (b) Prediction and order strategies. (c) Position predictor modules.

Figure 4: Overall ablation study results on ICCAD2015 benchmark.

4.5 Ablation Studies

Placement Order. We study the impact of placement order via an ablation comparing three strategies (Figure 4a): (1) entropy-guided ranker (ascending), (2) entropy-guided ranker (descending), and (3) macro area and connectivity (descending), which is a widely used heuristic order Lai et al. (2022); Xue et al. (2024). The results in Figure 4a show that entropy-guided ranking in ascending order outperforms both its descending variant and the heuristic baseline, indicating that placement order should be treated as an optimizable component rather than a predefined heuristic.

Global Prediction and Entropy-Guided Ordering. We quantify the individual contributions of global parallel prediction and entropy-guided ordering via an ablation comparing three model variants (Figure 4b): (1) fixed-order with sequential prediction, which adopts causal masking and a heuristic order, (2) global parallel prediction with fixed-order execution, which enables full-token parallel prediction while retaining the same fixed placement order, and (3) global parallel prediction with entropy-guided execution, our full model that dynamically determines placement order via the entropy-guided ranker. The results show that global parallel prediction reduces the average gHPWL from 6.45 to 6.29 under fixed execution orders, validating the superiority of full global information interaction. Further integrating entropy-guided ordering lowers the average gHPWL to 6.09, demonstrating that both global parallel prediction and adaptive ordering optimization effectively improve placement performance.

Position Predictor. We further conduct an ablation study on the configuration of the position predictor, as illustrated in Figure 4c. Specifically, we compare five action mapping modules, where modules (1)–(4) correspond to different embedding layer strategies and module (5) is a U-Net architecture commonly used in prior methods Lai et al. (2022); Xue et al. (2024). The evaluated settings include 2D sinusoidal initialization (frozen or trainable, with the frozen variant as our proposed method), random initialization (frozen or trainable), and a trainable U-Net architecture. The results show that frozen 2D sinusoidal encoding achieves the lowest gHPWL, about 5.43% lower than its trainable variant and 11.86% lower than the U-Net architecture. In contrast, random initialization performs worst, with gHPWL up to 15.39% higher. This indicates that a fixed spatial prior is more effective for position prediction than trainable or unstructured embeddings.

5 Final Remarks

Conclusion. In this paper, we propose HoliPlace, a Transformer-based RL framework for holistic macro placement. Unlike prior RL methods with fixed ordering, HoliPlace predicts position distributions for all macros using an encoder-only Transformer, capturing global dependencies. We further introduce an entropy-guided ranker to determine the placement order and adopt GRPO for unified optimization of positions and ordering. Experiments on the ICCAD2015 benchmark show that HoliPlace consistently outperforms existing methods, demonstrating the effectiveness of holistic modeling and joint optimization.

Limitations and future work. Despite the effectiveness of HoliPlace, several limitations remain. First, the placement order is optimized implicitly through entropy-guided ranking, rather than being explicitly learned, which may limit optimality in certain cases. Second, the current framework operates at a relatively coarse granularity, which may overlook fine-grained placement refinements. Future work will explore explicit order modeling, multi-scale or fine-grained optimization to better handle macro placement scenarios.

AI USE STATEMENT

Generative AI tools were used only for language editing, grammatical refinement, and auxiliary numerical-computation verification of this manuscript. All research content, including design, implementation, experiments, and analysis, was conducted entirely by the authors, who have reviewed and approved the final text and take full responsibility for this submission.

REFERENCES

Anthony Agnesina, Puranjay Rajvanshi, Tian Yang, Geraldo Pradipta, Austin Jiao, Ben Keller, Brucek Khailany, and Haoxing Ren. Autodmp: Automated dreamplace-based macro placement. In *Proceedings of the International Symposium on Physical Design*, pp. 149–157, 2023.

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.

Tung-Chieh Chen, Zhe-Wei Jiang, Tien-Chang Hsu, Hsin-Chen Chen, and Yao-Wen Chang. A high-quality mixed-size analytical placer considering preplaced blocks and density constraints. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 187–192, 2006.

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.

Ruoyu Cheng and Junchi Yan. On joint learning for solving placement and routing in chip design. *Advances in Neural Information Processing Systems*, 34:16508–16519, 2021.

J.P. Cohoon and W.D. Paris. Genetic placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 6(6):956–964, 1987.

Ji Deng, Zhao Li, Ji Zhang, and Jun Gao. EGPlace: An efficient macro placement method via evolutionary search with greedy repositioning guided mutation. In *Proceedings of the International Conference on Machine Learning*, volume 267, pp. 13237–13255, 2025.

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 *Proceedings of the International Conference on Machine Learning*, volume 235, pp. 15402–15417, 2024.

Myung Chul Kim, Jin Hu, Jiajia Li, and Natarajan Viswanathan. Iccad - 2015 cad contest in incremental timing - driven placement and benchmark suite. In *2015 IEEE/ACM International Conference on Computer - Aided Design (ICCAD)*, pp. 921–926, 2015.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

R.M. Kling and P. Banerjee. Esp: Placement by simulated evolution. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 8(3):245–256, 1989.

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

Y. Lai, J. Liu, Z. Tang, B. Wang, J. Hao, and P. Luo. Chipformer: Transferable chip placement via offline decision transformer. In *Proceedings of the International Conference on Machine Learning*, pp. 18346–18364, 2023.

Vint Lee, Minh Nguyen, Leena Elzeiny, Chun Deng, Pieter Abbeel, and John Wawrzynek. Chip placement with diffusion models. In *Proceedings of the International Conference on Machine Learning*, volume 267, pp. 33499–33514, 2025.

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*, pp. 1–6, 2019.

Don MacMillen, Raul Camposano, Dwight Hill, and Thomas W Williams. An industrial view of electronic design automation. *IEEE transactions on computer-aided design of integrated circuits and systems*, 19(12):1428–1448, 2000.

Igor L Markov, Jin Hu, and Myung-Chul Kim. Progress and challenges in vlsi placement research. In *Proceedings of the International Conference on Computer-Aided Design*, pp. 275–282, 2012.

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*, pp. 207–212, 2021.

Shibing Mo, Jing Liu, Jianchu Xu, and Ruilin Wu. Order matters: Unveiling the hidden impact of macro placement sequences via proxy-guided llm evolution. In *Proceedings of the International Conference on Machine Learning*, 2026.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

Y. Pu, T. Chen, Z. He, C. Bai, H. Zheng, Y. Lin, and B. Yu. Incremacro: Incremental macro placement refinement. In *Proceedings of the International Symposium on Physical Design*, pp. 169–176, 2024.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

C. Sechen and A. Sangiovanni-Vincentelli. The timberwolf placement and routing package. *IEEE Journal of Solid-State Circuits*, 20(2):510–522, 1985.

K. Shahookar and P. Mazumder. Vlsi cell placement techniques. *ACM Computing Surveys*, 23(2): 143–220, 1991.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Y. Shi, K. Xue, L. Song, and C. Qian. Macro placement by wire-mask-guided black-box optimization. *Advances in Neural Information Processing Systems*, 36:6825–6843, 2023.

Maolin Tang and Xin Yao. A memetic algorithm for vlsi floorplanning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(1):62–69, 2007.

Dhruv Vashisht, Harshit Rampal, Haiguang Liao, Yang Lu, Devika Shanbhag, Elias Fallon, and Levent Burak Kara. Placement in integrated circuits using cyclic reinforcement learning and simulated annealing. *arXiv preprint arXiv:2011.07577*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

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. *Advances in Neural Information Processing Systems*, pp. 140565–140588, 2024.

APPENDIX

The Appendix presents detailed supplementary materials for HoliPlace, including extended related work discussions, in-depth descriptions of the benchmark suites, comprehensive configurations of the HoliPlace framework, and a series of supplementary experimental validations to corroborate the core findings.

A RELATED WORK

GraphPlace Mirhoseini et al. (2021) is the first work to introduce RL into macro placement via modeling placement as a sequential decision problem. It divides the chip canvas into discrete grids, and for each step, the policy model will output the position of the current macro. DeepPR Cheng & Yan (2021) inherits the reinforcement learning paradigm proposed in GraphPlace, incorporating multi-modal inputs of graph and image data. Furthermore, it optimizes the two stages of macro placement and routing in a joint manner. MaskPlace Lai et al. (2022) represents input information exclusively through images, employing high-dimensional images with a resolution of 224×224. Meanwhile, it introduces a wire mask to characterize the impact of placing each macro at a specific grid position on the half-perimeter wire length (HPWL), as well as a position mask to filter out invalid locations (e.g., out-of-bounds positions and overlapping placements). ChiPFormer Lai et al. (2023) is the first method to formulate macro placement as an offline RL problem, leveraging the Decision Transformer Vaswani et al. (2017); Chen et al. (2021) to learn transferable placement strategies from pre-collected multi-task expert data. WireMask-BBO Shi et al. (2023) leverages wire mask to devise a greedy search strategy and it iteratively adjusts the position of one macro at a time, thereby guiding the evaluation and optimization of the objective function in an efficient and structured manner. EfficnetPlace Geng et al. (2024) combines Monte Carlo Tree Search (MCTS) and RL that MCTS provides high-value frontier nodes for the RL agent, which in turn updates tree node statistics through effective exploration to inversely facilitate tree pruning and expansion. MaskRegulate Xue et al. (2024) adopts the paradigm of adjusting the positions of macros and incorporates a regularity Pu et al. (2024) as a reward signal to train the policy model. Diffusion Lee et al. (2025) introduces a diffusion model to generate placements and constructs a synthetic dataset for training diffusion models. EGPlace Deng et al. (2025) introduces a greedy repositioning-guided mutation operator for systematic critical region optimization and an efficient mask computation algorithm accelerating layout evaluation.

B IMPLEMENTATION DETAILS

B.1 BENCHMARKS

Table 3 summarizes the detailed statistical characteristics of the eight circuits from the ICCAD2015 benchmark Kim et al. (2015), which are widely used in the evaluation of placement algorithms.

Table 3: Detailed Statistics of the ICCAD2015 Benchmark.

Circuit	#Macros	#Standard Cells	#Nets	#Pins	#Width	#Height	#Grid_Size
superblue1	424	1215820	1215710	3767494	8544	3132	224
superblue3	565	1219170	1224979	3905321	9710	3145	224
superblue4	300	801968	802513	2497940	5656	3150	224
superblue5	770	1090247	1100825	3246878	9399	4325	224
superblue7	441	1937699	1933945	6372094	6047	5413	224
superblue10	1629	984379	1898119	5560506	8264	5850	224
superblue16	99	985909	999902	3013268	5638	3061	224
superblue18	201	771845	771542	2559143	4614	3051	224

Table 4 presents the statistical information of 512-macro placement tasks based on the eight circuits from the ICCAD2015 benchmark, focusing on net-related characteristics. For each circuit, the table specifies the fixed number of macros to be placed (512 for all tasks), the total number of nets (**Total Nets**), as well as the minimum (**Macros with min net**) and maximum (**Macros with max net**)

number of macros connected by a single net. To ensure a consistent problem size across circuits, we construct a fixed set of 512 macros for each design. Specifically, when the number of macros in a circuit is fewer than 512, we follow prior work Xue et al. (2024) and supplement the set by selecting the largest standard cells (by area) as pseudo-macros. Conversely, when the number of macros exceeds 512, we select the 512 largest macros based on area. It can be seen that the total nets of different circuits vary (ranging from 313 to 930), and the connectivity complexity of nets differs significantly: most circuits have a minimum of 2 macros per net, while the maximum number of macros per net spans from 26 to 68. These diverse net connectivity features provide a representative test set for evaluating the performance of macro placement algorithms in handling different levels of wiring constraints.

Table 4: **Benchmark Circuit Statistics:** Total Nets and the Minimum/Maximum Number of Macros per Net for 512-macro Placement Tasks.

Circuit	Macros to place	Total Nets	Macros with min net	Macros with max net
superblue1	512	325	2	29
superblue3	512	704	2	61
superblue4	512	576	2	56
superblue5	512	930	2	68
superblue7	512	580	2	60
superblue10	512	313	6	26
superblue16	512	726	2	58
superblue18	512	322	2	33

B.2 PPA Evaluation Flow

For the main results reported in the paper, We evaluate PPA using a complete physical design evaluation flow based on commercial EDA tool *Cadence Innovus*. Specifically, baseline methods and our HoliPlace are first applied to place 512 macros. After macro placement, DREAMPlace Lin et al. (2019) is used to place the remaining standard cell modules. The complete placement result is then exported as a `.def` file.

Subsequently, we import all required design files into *Cadence Innovus* and load the generated `.def`. Early global routing is performed to obtain routing-related metrics, including rO-H, rO-V, and rWL. Finally, we use `report_timing` with pre-CTS timing settings to obtain WNS, TNS, and NVP.

B.3 Detailed settings of HoliPlace

Table 5 details the key training hyperparameters tailored for our HoliPlace in the experiments. For the evaluation phase, we first generate macro placement solutions via our proposed method and the baselines, then rerun DREAMPlace Lin et al. (2019) three times to place all the remaining macros and standard cells based on each macro placement. This repetition helps mitigate randomness in the standard cell placement results.

Table 5: Training and evaluation hyperparameters.

Configuration	Value	Configuration	Value
Optimizer	AdamW	Learning rate	1.0×10^{-4}
Total training steps	5000	Evaluation steps	50
Group size	16	Number of macros to place	512
Clip gradient norm	0.5	DREAMPlace evaluation number	3
Trade-off coefficient α	0.7		

Figure 5 illustrates the architecture of our encoder-only Transformer Vaswani et al. (2017), which adopts the same backbone as LlaDA Nie et al. (2025). The model consists of a stack of N identical encoder-only Transformer blocks. Each block follows a pre-normalization design: the input is first

Table 6: Configuration parameters of our Transformer model.

Category	Parameter	Our Setting
Model Dimensions	d_{model}	128
Architecture	n_{layers}	2
	n_{heads}	4
	$n_{\text{kv_heads}}$	4
	mlp_hidden_size	512
Normalization & Regularization	rms_norm_eps	1e-5

normalized using RMSNorm and then fed into a bidirectional self-attention module. The attention output is added to the residual connection, followed by another RMSNorm layer before being passed to a feed-forward network (FFN). A final residual connection completes the Transformer block. After stacking N such blocks, a top-level RMSNorm layer is applied to produce the final contextualized representations. The configuration of the Transformer model is summarized in Table 6.

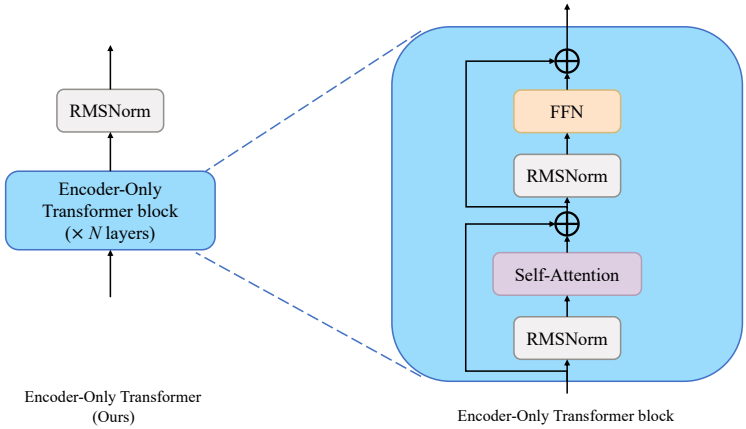


Figure 5: Overview of our encoder-only Transformer, which shares the same backbone architecture as LlaDA Nie et al. (2025).

Algorithms 1 and 2 provide the detailed pseudocode of the proposed simplified GRPO variant Shao et al. (2024) training framework and the entropy-guided action selection strategy, respectively. We do not directly adopt the original GRPO algorithm in our framework, mainly due to severe numerical instability under our from-scratch policy training setup. The instability stems from numerical issues in early training. Since we train the policy from scratch rather than initializing it from a pretrained model, the gap between `actor_logp` and `old_logp` can become extremely large, causing $\exp(\texttt{log_ratio})$ to overflow (Inf). For numerical stability, we additionally clip the log-ratio before exponentiation. However, this extra clipping inevitably zeroes gradients for samples whose `log_ratio` fall outside the clipping range. Combined with the subsequent ratio clipping in standard GRPO, a substantial portion of training samples contributes little or no effective gradient during the early optimization stage, resulting in slow and unstable convergence. This behavior is specific to our from-scratch training setting and differs from the typical GRPO setting for LLM fine-tuning, where the policy is initialized from a pretrained model and the `log_ratio` remains well-behaved during optimization. Algorithm 1 describes the overall training procedure, where the policy network is optimized using group-relative advantages computed from multi-objective placement rewards. For better training stability, we set the old policy as $\log \pi_{\text{old}} = \text{detach}(\log \pi_\theta)$ in our implementation. As a result of this design, our method operates in a purely on-policy setting, where trajectories are sampled directly from the current policy without reuse, and importance weighting is no longer required. In practice, the original GRPO objective (Eq.4) led to severe training instability, including occasional loss explosions. We attribute this to three factors: (1) our shallow 2-layer Transformer is sensitive to abrupt updates; (2) the reference policy is updated infrequently

Algorithm 1 GRPO Training for Macro Placement

Input: current policy parameters θ, batch data $\mathcal{D}$ (netlist + macro embeddings), clipping coefficient ϵ, wirelength coefficient α_{wire}, iteration steps S, max gradient norm $\gamma_{\text{max_grad}}$
model.train()
for $s = 1$ **to** S **do**
  1. Forward pass
  $\text{logits} = \text{Forward}(\mathcal{D}, \theta)$

  2. Sample actions and calculate multi-objective reward
  $(\text{pos}, r_{\text{legal}}, r_{\text{wire}}, r_{\text{reg}}) = \text{SelectActionWithEntropy}(\text{logits})$
  $r_{\text{total}} = r_{\text{legal}} + \alpha_{\text{wire}} \cdot r_{\text{wire}} + (1 - \alpha_{\text{wire}}) \cdot r_{\text{reg}}$

  3. Group-based advantage normalization
  $\hat{A} = \frac{r_{\text{total}} - \text{mean}(r_{\text{total}})}{\text{std}(r_{\text{total}}) + 10^{-10}}$

  4. Log probabilities for current and old policy
  $\log p_\theta = \text{ComputeLogProb}(\text{logits}, \text{pos})$
  $\log p_{\text{old}} = \text{detach}(\log p_\theta)$

  5. Compute GRPO loss, clip term and KL term are omitted
  $\rho = \exp(\log p_\theta - \log p_{\text{old}})$
  $L_1 = \rho \cdot \hat{A}$
  $\mathcal{L}_{\text{GRPO}} = -\text{mean}(L_1)$

  6. Policy update
  optimizer.zero_grad()
  $\text{Backpropagate}(\mathcal{L}_{\text{GRPO}})$
  $\text{ClipGradNorm}(\theta, \gamma_{\text{max_grad}})$
  optimizer.step()
end for

via EMA, creating a lag behind the current policy; and (3) Gumbel noise injected into the actor output increases the variance of exploration. These factors can produce extreme importance ratios $p = \frac{\pi_\theta}{\pi_{\text{old}}}$. When p is large and the advantage is negative, the clipped objective selects the unclipped term, resulting in excessively large loss values and unstable updates. Algorithm 2 details how macro placement actions are sampled by dynamically determining the placement order based on macro-wise entropy, enabling joint optimization of macro positions and placement order. Together, these algorithms illustrate how entropy-guided sequential execution is integrated into the GRPO training pipeline.

C Additional Experimental Results

C.1 Comparison with Analytical Placement Methods

It should be noted that the analytical methods perform placement of both macros and standard cells, whereas existing RL-based placement methods (including ours) are constrained to a limited subset of macros, e.g., the 512 largest macros. The placement of all remaining macros and standard cells is subsequently addressed by analytical placers (e.g., DREAMPlace Lin et al. (2019)). Thus, RL-based placement methods can be regarded as a complementary enhancement built upon the foundational capabilities of analytical methods.

As shown in Table 7, HoliPlace achieves the best average performance among all compared methods, reducing gHPWL to 6.09 compared to 7.27, 7.18, and 7.13 for RePlace Cheng et al. (2018), DREAMPlace Lin et al. (2019), and AutoDMP Agnesina et al. (2023), respectively. This corresponds to improvements of 16.23%, 15.25%, and 14.58%. HoliPlace outperforms analytical methods on 6 out of 8 circuits, with particularly significant gains on superblue4, superblue7, and superblue16.

Algorithm 2 Entropy-Guided Action Selection for Macro Placement (denoted as SelectActionWithEntropy)

Input: policy logits $\mathbf{Z} \in \mathbb{R}^{G \times M \times P}$, training flag train
Output: sampled positions $\mathbf{pos}$, reward components $(r_{\text{legal}}, r_{\text{wire}}, r_{\text{reg}})$
Initialize position tensor $\mathbf{pos} = \mathbf{0}$
Initialize reward vectors $r_{\text{legal}}, r_{\text{wire}}, r_{\text{reg}} = \mathbf{0}$
if train $=$ true **then**
    Add Gumbel noise to logits: $\eta \sim \text{Uniform}(0,1), \mathcal{G} = -\log(\eta)$, $\tilde{\mathbf{Z}} = \frac{\exp(\mathbf{Z})}{\mathcal{G}}$
else
    $\tilde{\mathbf{Z}} = \mathbf{Z}$
end if
for $g = 1$ to G **do**
    Initialize macro placement map $s = \emptyset$
    Compute macro-wise entropy: $H_m = -\sum_p \pi_{g,m,p} \log \pi_{g,m,p},\ \pi = \text{softmax}(\tilde{\mathbf{Z}})$
    Determine placement order: $\mathcal{O} = \text{argsort}(H_m)$ *(low entropy macros first)*
    for all macro $m \in \mathcal{O}$ **do**
        Generate position mask $\mathbf{mask}_m$ based on $\mathcal{M}_{\text{placed}}$
        Mask invalid positions: $\tilde{\mathbf{z}}_m = \tilde{\mathbf{Z}}_{g,m}$ with $\mathbf{mask}_m$
        Select position: $p_m = \arg\max(\tilde{\mathbf{z}}_m)$
        if p_m is illegal **then**
            $r_{\text{legal}}[g] = r_{\text{legal}}[g] - C_{\text{illegal}}$
        end if
        Record placement $\mathbf{pos}[g, m] = p_m$
        Update $\mathcal{M}_{\text{placed}} = \mathcal{M}_{\text{placed}} \cup \{m\}$
        Update partial solution s
    end for
    Calculate placement rewards: $(r_{\text{wire}}[g], r_{\text{reg}}[g]) = \text{Eval}(s)$
end for
Normalize r_{wire} and r_{reg} across groups
if train $=$ true **then**
    return $\mathbf{pos}, r_{\text{legal}}, r_{\text{wire}}, r_{\text{reg}}$
else
    return $\mathbf{pos}$
end if

Although analytical methods achieve competitive or better results on a few circuits (e.g., superblue5 and superblue18), the overall trend favors HoliPlace.

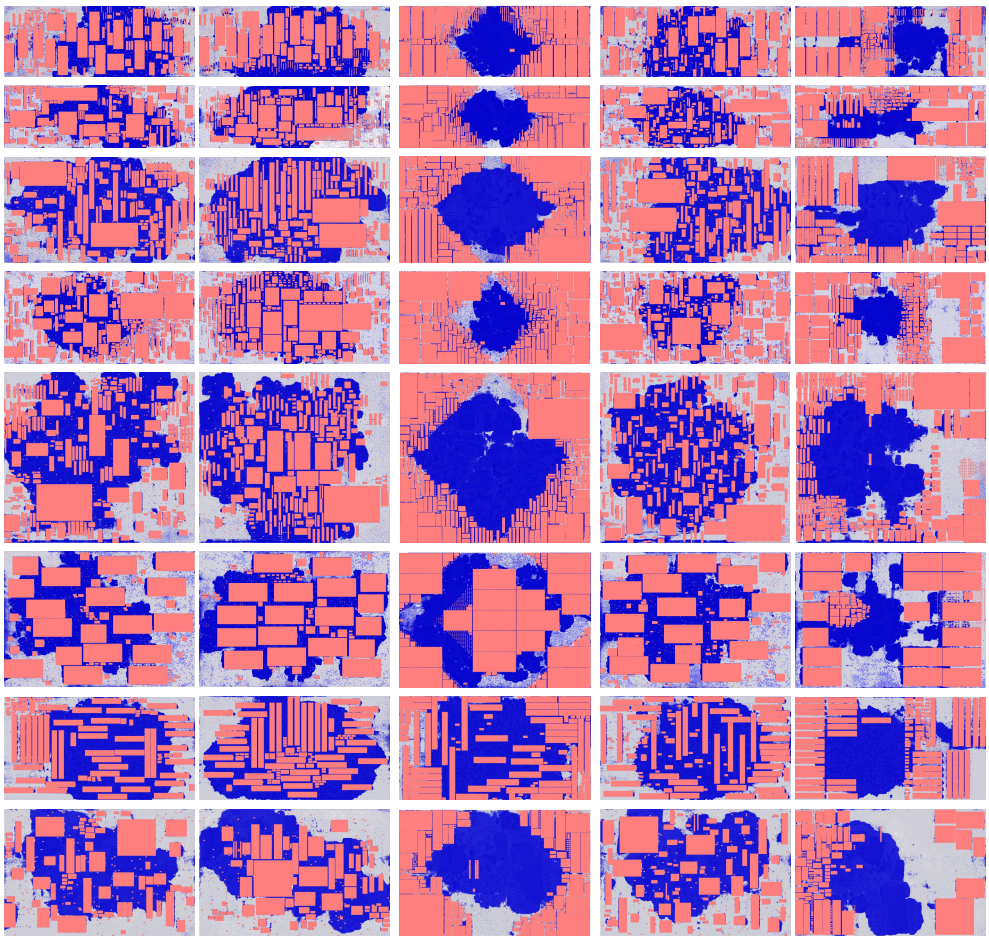
These results demonstrate that HoliPlace effectively improves global macro placement quality, providing a stronger initialization for subsequent analytical placement. This further supports the view that RL-based macro placement serves as a complementary enhancement to analytical methods, leading to improved overall placement solutions.

C.2 Visualization Results

Figure 6 illustrates the macro placement results generated by different baseline methods and our proposed HoliPlace on eight circuits from the ICCAD2015 benchmark Kim et al. (2015). From top to bottom, the circuits correspond to **superblue1**, **superblue3**, **superblue4**, **superblue5**, **superblue7**, **superblue10**, **superblue16**, and **superblue18**. Each column represents a different placement method: (a) MaskPlace Lai et al. (2022), (b) ChiPFormer Lai et al. (2023), (c) MaskRegulate Xue et al. (2024), (d) EGPlace Deng et al. (2025), and (e) HoliPlace (ours). Red regions denote macros placed by each method, while blue regions correspond to the remaining macros and standard cells placed by DREAMPlace Lin et al. (2019).

Table 7: Results of gHPWL (1e8) between our HoliPlace and analytical methods on ICCAD2015 benchmark.

Circuit	RePlace	DREAMPlace	AutoDMP	HoliPlace
superblue1	6.76	7.06	7.03	**5.24**
superblue3	7.52	7.44	7.23	**6.60**
superblue4	6.69	6.66	6.44	**4.12**
superblue5	6.78	6.95	**6.50**	7.17
superblue7	10.97	10.24	10.38	**8.44**
superblue10	9.94	10.23	10.77	**9.46**
superblue16	6.72	6.34	6.10	**4.92**
superblue18	2.77	**2.54**	2.56	2.75
Average	7.27	7.18	7.13	**6.09**



(a) MaskPlace **(b) ChiPFormer** **(c) MaskRegulate** **(d) EGPlace** **(e) HoliPlace**

Figure 6: Placement Visualization via DREAMPlace on Eight Circuits from the ICCAD2015 Benchmark. Red regions denote the macros placed by baselines and our method, while blue regions represent the remaining macros and standard cells placed by DREAMPlace.

C.3 Detailed Results about Ablation Studies

In this section, we provide the per-circuit performance details for the ablation studies conducted on the ICCAD2015 benchmark. While the main text reports the average results to demonstrate overall

Table 8: Comparison of gHPWL (1e8) across different placement order strategies on each circuit from ICCAD2015 benchmark. The best result for each circuit is shown in **bold**.

Circuit	Entropy-guided ranker (ascending)	Entropy-guided ranker (descending)	Area and connectivity (descending)
superblue1	**5.24**	5.86	5.50
superblue3	**6.60**	6.80	6.88
superblue4	**4.12**	4.69	4.22
superblue5	**7.17**	7.61	7.29
superblue7	**8.44**	9.17	8.70
superblue10	9.46	**9.32**	9.71
superblue16	**4.92**	4.97	5.22
superblue18	**2.75**	2.90	2.80
Average	**6.09**	6.42	6.29

Table 9: Comparison of gHPWL (1e8) across different position predictor strategies and modules on each circuit from ICCAD2015 benchmark. The best result for each circuit is shown in **bold**.

Circuit	2D Sinusoidal (frozen)	2D Sinusoidal (trainable)	Random (frozen)	Random (trainable)	U-Net (trainable)
superblue1	**5.24**	5.73	6.14	6.05	6.06
superblue3	**6.60**	6.95	8.08	7.72	7.61
superblue4	**4.12**	4.28	4.80	4.79	4.75
superblue5	**7.17**	7.76	8.99	8.78	8.23
superblue7	**8.44**	8.76	9.82	9.62	9.54
superblue10	**9.46**	9.96	10.38	10.03	10.08
superblue16	**4.92**	5.08	5.00	5.29	5.23
superblue18	**2.75**	2.80	3.02	3.07	3.01
Average	**6.09**	6.42	7.03	6.92	6.81

trends, these detailed tables offer a more granular view of the consistency of our proposed methods across various circuit designs.

Table 8 presents a comparison of different placement order strategies. The results show that the **Entropy-guided ranker (ascending)** achieves the lowest gHPWL on nearly all tested circuits. It outperforms the traditional **Area and connectivity (descending)** heuristic on every single test case, and surpasses the descending order variant across all circuits except superblue10. This overall superiority validates that placing macros with lower entropy earlier in the sequence effectively reduces overall wirelength.

Table 9 evaluates various position predictor strategies and architectural modules. The **2D Sinusoidal (frozen)** encoding strategy emerges as the top performer for every circuit in the benchmark. Notably, keeping the sinusoidal embeddings fixed (frozen) yields better results than the trainable version or more complex alternatives such as U-Net. This suggests that the fixed geometric prior provided by the sinusoidal encoding serves as a robust spatial representation that simplifies the optimization landscape for macro placement, leading to superior and more stable convergence compared to learned or random embeddings.

Table 10 evaluates sequential prediction, global parallel prediction with fixed-order execution, and global parallel prediction with entropy-guided execution on ICCAD2015. Even with the same predefined placement order, global parallel prediction outperforms sequential prediction by exploiting full cross-macro information. Combining parallel prediction with entropy-guided execution achieves further improvements across most circuits, lowering the average gHPWL from 6.45 to 6.09. This confirms that both global-parallel prediction and adaptive entropy-guided ordering contribute to better macro-placement solutions.

C.4 Ablation Study on Group Size

We evaluate the impact of group size on macro placement quality using our GRPO algorithm. Table 11 reports the gHPWL results ($\times 10^8$) on the ICCAD2015 benchmark circuits across different group sizes. For each circuit, the best-performing group size is highlighted in **bold**. We observe that moderate group sizes, such as 16 for most circuits, often yield the lowest gHPWL, indicating a more balanced exploration of placement orders and positions. Extremely small or large group sizes

Table 10: Ablation study comparing sequential prediction with fixed-order execution, global parallel prediction with fixed-order execution, and global parallel prediction with entropy-guided execution on the ICCAD2015 benchmark (gHPWL, 10^8).

Circuit	Placement Strategy		
	Sequential prediction with Fixed-order	Global parallel prediction with fixed-order execution	Global parallel prediction with entropy-guided execution
superblue1	5.73	5.50	**5.24**
superblue3	6.93	6.88	**6.60**
superblue4	4.34	4.22	**4.12**
superblue5	7.79	7.29	**7.17**
superblue7	8.65	8.70	**8.44**
superblue10	9.96	9.71	**9.46**
superblue16	5.36	5.22	**4.92**
superblue18	2.87	2.80	**2.75**
Average	6.45	6.29	**6.09**

Table 11: Comparison of gHPWL (1e8) across different group sizes on the ICCAD2015 benchmark circuits. The best result for each circuit is shown in **bold**.

Circuit	Group Size					
	2	4	8	16	32	64
superblue1	5.95	5.46	5.89	**5.24**	5.24	5.55
superblue3	6.89	6.99	6.66	**6.60**	6.63	6.82
superblue4	4.25	4.10	**4.08**	4.12	4.23	**4.08**
superblue5	7.28	7.22	7.80	**7.17**	7.29	7.21
superblue7	8.95	**8.40**	8.77	8.44	8.70	8.42
superblue10	9.43	**9.04**	9.56	9.46	9.46	9.30
superblue16	5.15	5.23	5.20	**4.92**	4.97	5.10
superblue18	**2.73**	2.92	2.84	2.75	2.78	2.85

tend to produce slightly worse results, suggesting that both under- and over-grouping can limit the effectiveness of the policy.

Figure 7 visualizes the average gHPWL across all ICCAD2015 circuits for different group sizes. The trend confirms that a group size of 16 achieves the lowest average gHPWL, consistent with the per-circuit results in Table 11. This ablation study demonstrates that the choice of group size is critical for GRPO, as it directly affects the policy's ability to optimize macro placements effectively.

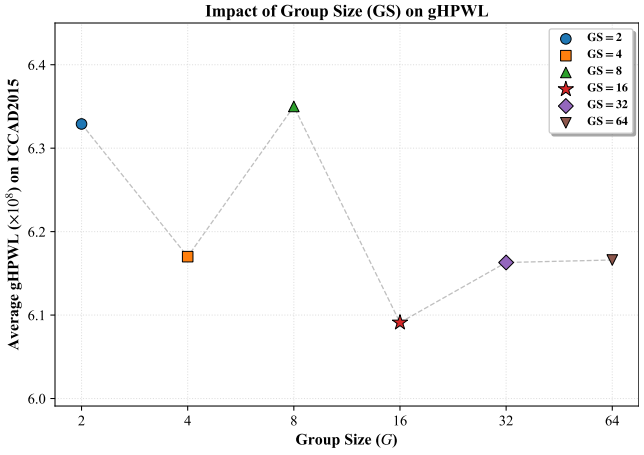


Figure 7: Comparison of average gHPWL on ICCAD2015 between different group sizes.

Table 12: Results of gHPWL ($1e8$) for different RL algorithms on the ICCAD2015 benchmark.

Circuit	HoliPlace w/ PPO	HoliPlace w/ Standard GRPO	HoliPlace w/ Simplified GRPO
superblue1	6.75	6.40	**5.24**
superblue3	7.41	7.38	**6.60**
superblue4	4.74	4.51	**4.12**
superblue5	8.81	7.87	**7.17**
superblue7	9.40	8.86	**8.44**
superblue10	10.08	9.49	**9.46**
superblue16	5.59	5.19	**4.92**
superblue18	2.97	2.84	**2.75**
Average	6.97	6.57	**6.09**

C.5 Ablation Study on RL Algorithm

We compare three RL training variants of HoliPlace to examine the effect of the proposed GRPO design. As shown in Table 12, our simplified GRPO variant achieves lower gHPWL than standard GRPO across all evaluated circuits. On average, it reduces gHPWL from 6.57 to 6.09, corresponding to a reduction of 7.31%. We further compare HoliPlace with a PPO-style variant. Since PPO is inherently a step-wise optimization algorithm that relies on a critic and intermediate feedback for advantage estimation, we adopt the critic design from prior work, including MaskPlace and MaskRegulate. The PPO-style variant achieves an average gHPWL of 6.97, which is 12.63% higher than HoliPlace.

These results highlight the advantage of terminal-reward optimization for our constructive placement setting. PPO relies on a critic and intermediate reward signals to estimate step-wise advantages, introducing additional estimation error and optimization complexity. In contrast, our method directly compares complete placement outcomes within each group, providing a learning signal aligned with the final placement objective.

C.6 Ablation Study on GAE Pretraining

As shown in Table 13, HoliPlace with GAE pretraining outperforms the variant without pretraining across all evaluated circuits. On average, GAE pretraining reduces gHPWL from 6.62 to 6.09, yielding an improvement of 8.01%.

We attribute this gain to the decoupled two-stage training strategy. In the pretraining stage, the GAE learns a stable and informative latent representation of the placement space. By fixing the GAE in the second stage, the policy network can focus on optimizing placement decisions without being affected by a shifting representation space. In contrast, jointly training the GAE and the policy network introduces additional non-stationarity, as both the representation and the policy are updated simultaneously, making optimization more challenging.

As further illustrated in Figure 8 and Figure 9, GAE pretraining leads to significantly more stable training dynamics. With pretraining, both HPWL and regularity curves exhibit smoother convergence and reduced variance across training steps for most circuits. In contrast, without pretraining, the training curves are notably noisier, with frequent oscillations and large fluctuations, especially in the early and middle stages of training.

These results demonstrate that GAE pretraining not only improves final placement quality, but also stabilizes the training process by providing a more structured and stationary representation space for downstream policy learning.

C.7 Additional Generalization Results

To further evaluate the generalization capability of our method, we conduct additional experiments by training the model separately on each of the following circuits: superblue3, superblue4, superblue5, superblue7, superblue10, superblue16, and superblue18 circuit, and testing it on the remaining seven circuits. The corresponding cross-circuit generalization results are reported in Tables 14–

Table 13: Results of gHPWL (1e8) for HoliPlace with/without GAE pretraining on ICCAD2015 benchmark.

Circuit	HoliPlace w/ GAE pretraining	HoliPlace w/o GAE pretraining
superblue1	**5.24**	5.81
superblue3	**6.60**	7.32
superblue4	**4.12**	4.45
superblue5	**7.17**	7.94
superblue7	**8.44**	9.42
superblue10	**9.46**	9.88
superblue16	**4.92**	5.32
superblue18	**2.75**	2.82
Average	**6.09**	**6.62**

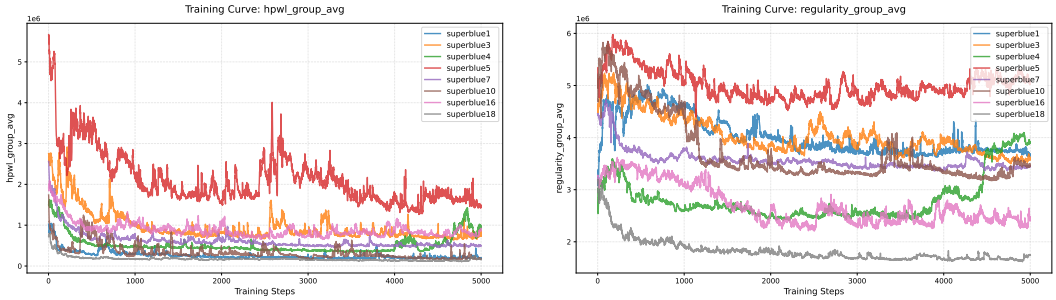


Figure 8: Training curves of average group (group size=16) macro HPWL and regularity for HoliPlace w/ GAE pretraining on eight circuits from ICCAD2015 benchmark.

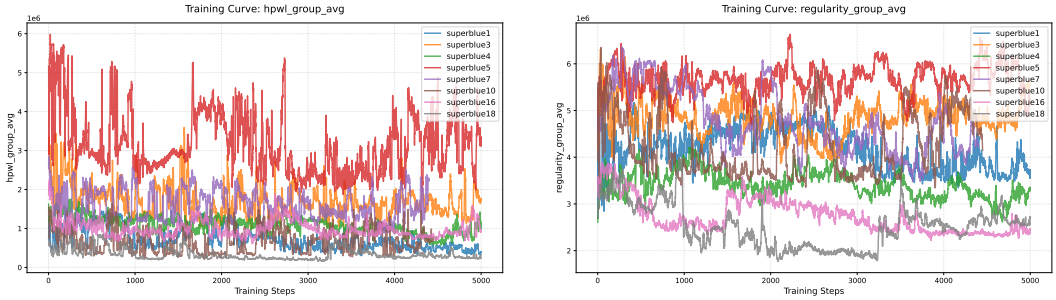


Figure 9: Training curves of average group (group size=16) macro HPWL and regularity for HoliPlace w/o GAE pretraining on eight circuits from ICCAD2015 benchmark.

20, respectively, which further demonstrate the robustness and strong generalization ability of our approach.

C.8 Training and Inference Time for HoliPlace

As the group size is a hyperparameter, the training time of HoliPlace is 0.9, 1.7, 3.4, and 6.5 hours for group sizes of 2, 4, 8, and 16, respectively. In contrast, MaskPlace Lai et al. (2022) requires 3.8 hours of training. It should be noted that, as illustrated in Figure 7, HoliPlace achieves similar placement quality across different group sizes (e.g., group size = 4 or 16). The hardware platform is equipped with an AMD EPYC 9654 CPU and an NVIDIA RTX Pro 6000 GPU.

Thanks to the proposed global planning mechanism, our method can derive the final placement through a single forward inference combined with entropy-guided sequential execution. In contrast, traditional RL methods require each macro to pass through the model sequentially and update the

Table 14: Generalization results of gHPWL (1e8) between our HoliPlace and other RL-based methods. The model is trained on **superblue3** and evaluated on the remaining seven circuits. The best result of each metric on each circuit is shown in **bold**.

Circuit	MaskPlace	ChiPFormer	MaskRegulate	HoliPlace
superblue1	7.33	7.50	6.25	**6.11**
superblue4	5.14	5.31	4.70	**4.53**
superblue5	9.90	10.26	**7.59**	8.24
superblue7	9.78	10.02	**8.95**	9.05
superblue10	10.10	11.33	11.03	**9.62**
superblue16	5.96	5.75	5.44	**5.38**
superblue18	3.38	3.15	**2.87**	**2.87**
Average	7.37	7.62	6.69	**6.54**

Table 15: Generalization results of gHPWL (1e8) between our HoliPlace and other RL-based methods. The model is trained on **superblue4** and evaluated on the remaining seven circuits. The best result of each metric on each circuit is shown in **bold**.

Circuit	MaskPlace	ChiPFormer	MaskRegulate	HoliPlace
superblue1	7.17	7.88	**6.32**	6.56
superblue3	8.83	10.11	7.70	**7.30**
superblue5	9.85	10.13	**7.90**	7.91
superblue7	9.33	9.86	**8.89**	9.09
superblue10	11.31	11.30	11.68	**9.63**
superblue16	5.83	5.62	**5.49**	5.55
superblue18	3.26	3.62	2.87	**2.85**
Average	7.94	8.36	7.26	**6.98**

Table 16: Generalization results of gHPWL (1e8) between our HoliPlace and other RL-based methods. The model is trained on **superblue5** and evaluated on the remaining seven circuits. The best result of each metric on each circuit is shown in **bold**.

Circuit	MaskPlace	ChiPFormer	MaskRegulate	HoliPlace
superblue1	7.25	7.55	**6.27**	6.43
superblue3	8.47	8.77	7.71	**7.32**
superblue4	4.86	5.24	4.69	**4.31**
superblue7	9.58	9.65	**9.03**	9.09
superblue10	11.00	11.38	11.42	**9.83**
superblue16	5.92	5.84	5.51	**5.47**
superblue18	3.41	3.38	2.88	**2.84**
Average	7.21	7.40	6.79	**6.47**

state (e.g., wire masks) to prepare for the next macro's inference (shown in Figure 1). Experimental results show that our method generates a placement solution in just 0.7 seconds, compared to 3.5 seconds for MaskPlace.

Table 17: Generalization results of gHPWL (1e8) between our HoliPlace and other RL-based methods. The model is trained on **superblue7** and evaluated on the remaining seven circuits. The best result of each metric on each circuit is shown in **bold**.

Circuit	MaskPlace	ChiPFormer	MaskRegulate	HoliPlace
superblue1	6.93	7.76	6.29	**6.24**
superblue3	8.74	14.21	7.78	**7.34**
superblue4	5.33	5.44	4.72	**4.44**
superblue5	10.24	10.87	**7.78**	8.18
superblue10	10.37	11.25	10.66	**9.64**
superblue16	5.98	5.78	5.48	**5.31**
superblue18	3.40	3.23	3.06	**3.00**
Average	7.28	8.36	6.54	**6.31**

Table 18: Generalization results of gHPWL (1e8) between our HoliPlace and other RL-based methods. The model is trained on **superblue10** and evaluated on the remaining seven circuits. The best result of each metric on each circuit is shown in **bold**.

Circuit	MaskPlace	ChiPFormer	MaskRegulate	HoliPlace
superblue1	7.25	7.68	**6.31**	6.40
superblue3	8.77	15.06	7.42	**7.04**
superblue4	5.20	5.38	4.71	**4.39**
superblue5	10.12	10.36	**7.67**	8.02
superblue7	9.86	9.70	8.99	**8.96**
superblue16	5.95	5.66	5.52	**5.39**
superblue18	3.25	3.23	3.06	**2.97**
Average	7.20	8.15	6.24	**6.17**

Table 19: Generalization results of gHPWL (1e8) between our HoliPlace and other RL-based methods. The model is trained on **superblue16** and evaluated on the remaining seven circuits. The best result of each metric on each circuit is shown in **bold**.

Circuit	MaskPlace	ChiPFormer	MaskRegulate	HoliPlace
superblue1	7.34	7.25	**6.22**	6.40
superblue3	8.74	9.31	7.93	**7.02**
superblue4	5.08	5.42	4.75	**4.38**
superblue5	9.54	10.33	**7.81**	8.02
superblue7	9.60	10.12	**8.75**	8.96
superblue10	11.80	10.58	11.94	**10.26**
superblue18	3.37	3.23	3.03	**2.97**
Average	7.92	8.03	7.20	**6.86**

Table 20: Generalization results of gHPWL (1e8) between our HoliPlace and other RL-based methods. The model is trained on **superblue18** and evaluated on the remaining seven circuits. The best result of each metric on each circuit is shown in **bold**.

Circuit	MaskPlace	ChiPFormer	MaskRegulate	HoliPlace
superblue1	7.14	7.75	6.20	**6.11**
superblue3	8.35	9.71	7.50	**7.38**
superblue4	5.05	5.23	4.70	**4.31**
superblue5	10.27	9.84	**7.79**	8.41
superblue7	9.74	10.05	**8.89**	9.41
superblue10	11.18	11.36	11.77	**9.98**
superblue16	5.95	5.84	5.51	**5.49**
Average	8.24	8.54	7.48	**7.30**