MACROSPA: SPATIAL PRIOR ADAPTATION FOR DIFFUSION-BASED MACRO PLACEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Macro placement determines the spatial organization of large circuit blocks and affects subsequent standard-cell placement and routing. However, the spatial preferences inherited by pretrained placers may not align with target design objectives. We propose **MacroSPA**, a post-training framework for **S**patial **P**rior **A**daptation (SPA) that enables the same pretrained diffusion-based macro placer to acquire distinct spatial priors directly from target context–placement samples. These samples can encode prescribed preferences through synthetic generation or capture layout patterns produced by an existing placement optimizer. SPA adjusts how features of the current layout enter graph message passing and feature fusion while updating only 3.03% of the adapted model's parameters and keeping the pretrained backbone frozen. The framework complements SPA with Minkowski Guided Refinement to reduce residual geometric conflicts and with size-adaptive constraint evaluation for efficient inference on large circuits. Experiments show that MacroSPA acquires distinct spatial priors and can improve downstream outcomes relative to the matched pipeline without SPA. It also achieves competitive routing and post-route performance among the evaluated methods. The relative benefits of different spatial priors vary across circuits and objectives, highlighting the practical value of adapting the inherited spatial prior to the target design.

1 INTRODUCTION

Macro placement determines the locations of large circuit blocks such as memories on a chip canvas. Their spatial organization affects macro interconnects and shapes the space available for subsequent standard-cell placement and routing, ultimately influencing the chip's power, performance, and area (PPA). Macro arrangement is therefore a key design decision, with the preferred spatial organization depending on circuit connectivity and downstream physical design requirements.

Analytical and search-based placers typically optimize layouts separately for each circuit (Cheng et al., 2019; Lin et al., 2019; Shi et al., 2023), while transferable learning-based placers amortize part of this optimization across circuits (Mirhoseini et al., 2021; Lai et al., 2023). Among learning-based approaches, generative placers learn circuit-conditioned distributions over macro layouts (Lee et al., 2025; Xie et al., 2026). Like other learned placers, pretrained generative models encode both reusable placement knowledge and spatial preferences shaped by their training data and objectives. We use the term **spatial prior** to denote these systematic preferences, as reflected in the conditional distribution of layouts given circuit and geometric context, rather than the exact location of any individual macro.

Spatial priors inherited from pretraining, however, may not suit every target setting, as the appropriate spatial organization depends on physical design conditions. For example, placing large macros near the die periphery can preserve central space for standard-cell placement and routing by reducing interior blockages (Pu et al., 2024; Xue et al., 2024). Yet in designs with many macros, allowing macros in the interior through hierarchical organization can reduce peripheral stacking and better preserve dataflow (Kahng et al., 2024). Moreover, when connection length scales differ from those seen during training, inductive biases may lead to excessive clustering or unnecessarily long connections (Lee et al., 2025). This motivates a reusable post-training mechanism for adapting inherited spatial priors to target design conditions while retaining the same pretrained base.

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.

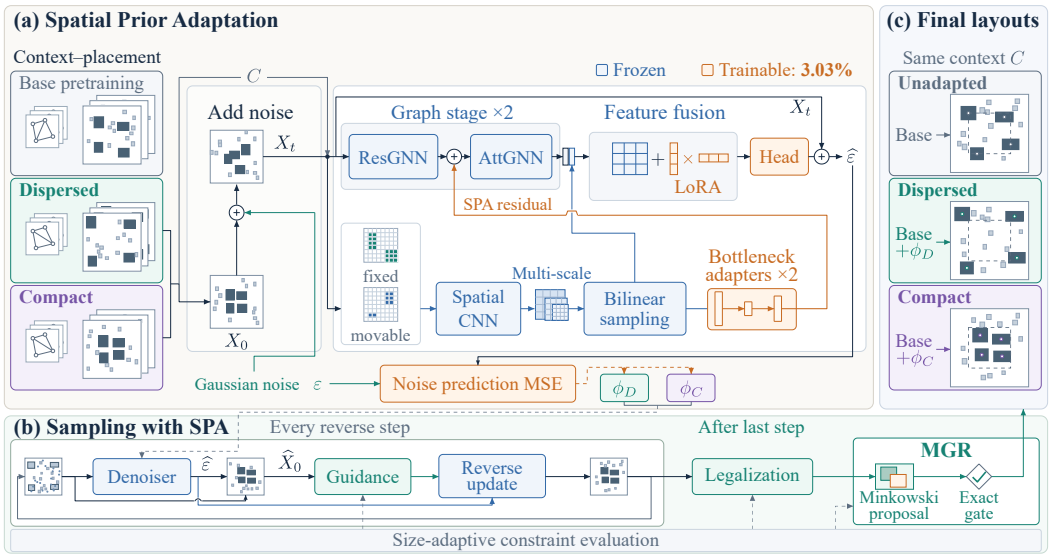


Figure 1: **Overview of MacroSPA. (a)** SPA uses target context–placement samples to adapt the same pretrained base, adjusting how spatial features enter graph message passing and feature fusion while keeping the backbone frozen. **(b)** Diffusion sampling with the adapted model is followed by MGR for residual geometric refinement, while size-adaptive constraint evaluation supports scalable inference. **(c)** MacroSPA enables the same pretrained base to acquire distinct spatial priors specified by target samples, yielding corresponding spatial organizations in the final refined layouts.

Existing approaches use learned metric guidance or preference optimization to align placement behavior with design objectives (Geng et al., 2025; Gao et al., 2026). However, when target context–placement samples from synthetic generators or existing placement optimizers already specify the desired spatial organization, using these approaches requires additional quality scores or rankings beyond the samples themselves. For pretrained generative placers, these samples can directly supervise adaptation of the layout distribution, as demonstrated by fine-tuning for transfer to larger circuits (Lee et al., 2025). But full fine-tuning requires updating the entire model for each target distribution. This raises two questions: what correction to the pretrained placer's behavior is needed to reflect the spatial prior specified by target samples, and how can that correction be learned through a compact trainable path?

We introduce **MacroSPA**, a post-training framework for **S**patial **P**rior **A**daptation (SPA) that enables the same pretrained diffusion-based macro placer to acquire distinct spatial priors directly from target context–placement samples. The framework separates distribution-level prior adaptation from instance-level geometric refinement, learning target placement behavior through a compact adaptation path and reducing residual overlaps and boundary violations in generated layouts. Figure 1 summarizes this two-level framework.

Our contributions are threefold:

Sample Specified Prior Acquisition. We formulate post-training acquisition of distinct spatial priors in the same pretrained diffusion-based macro placer, with target conditional layout distributions specified by context–placement samples. Under the standard noise prediction objective, we characterize the learning target for adaptation as a denoising correction relative to the pretrained base.

Spatial Prior Adaptation. We propose SPA to approximate this correction through a compact trainable path, using context–placement samples from synthetic generators or optimization pipelines. SPA adjusts how pretrained spatial features enter graph message passing and graph–spatial fusion while keeping the backbone frozen, enabling the same model to adapt to different spatial priors.

Scalable Constraint-Aware Inference. We develop Minkowski Guided Refinement (MGR) to reduce residual overlap and boundary violations while limiting HPWL degradation. To address the quadratic cost of dense pairwise constraint evaluation, we implement size-adaptive evaluation routes, thereby improving inference efficiency and scalability for large circuits.

Empirically, SPA enables the same pretrained base to acquire distinct spatial priors whose effects persist after geometric refinement. Compared with full fine-tuning, SPA achieves better macro-only placement quality at lower adaptation cost while updating approximately 3% as many parameters. On IBM (Adya et al., 2004) and ISPD (Nam et al., 2005), MacroSPA with synthetic targets achieves the best average macro-only HPWL and RUDY (standard proxies for wirelength and routing demand) among the compared methods, and reduces average downstream weighted HPWL relative to the matched pipeline without SPA. On ChiPBench (Wang et al., 2025), it achieves the best overall average rank across routing and post-route PPA metrics among the evaluated methods. Beyond placement quality, MacroSPA uses sparse constraint evaluation to support scalable inference, achieving a 32.82× speedup and 89.07% lower peak GPU memory usage than the matched dense route on the largest evaluated instance.

2 Related Work

Macro placement paradigms. Methods such as RePlAce, DREAMPlace, and WireMask-BBO optimize layouts for individual designs using analytical or search-based procedures (Cheng et al., 2019; Lin et al., 2019; Shi et al., 2023). Learning-based approaches include sequential placement policies (Mirhoseini et al., 2021; Lai et al., 2022; 2023) and joint generative placers that learn circuit-conditioned distributions over macro layouts (Lee et al., 2025; Yoon et al., 2026; Xie et al., 2026).

Spatial priors and post-training adaptation. Spatial preferences enter macro placement through representations, objectives, and training data. Hier-RTLMP exploits register-transfer-level hierarchy and dataflow (Kahng et al., 2024), while LaMPlace guides placement using learned predictions of cross-stage metrics (Geng et al., 2025). At the data level, FlowPlace incorporates spatial preferences through synthetic layouts sampled with legal position masks and configurable spatial scores (Xie et al., 2026). Post-training adaptation has also been explored for transfer and preference optimization. ChipDiffusion fine-tunes on larger synthetic circuits to improve transfer to realistic designs (Lee et al., 2025). EXPlace encodes expert knowledge in states and rewards, then applies direct preference optimization using timing-ranked trajectories (Gao et al., 2026). MacroSPA adapts a pretrained diffusion-based macro placer to spatial preferences specified by target context–placement samples. Appendix E further compares how related methods specify and incorporate spatial priors, including their supervision requirements and parameter updates.

Constraint handling. Generative placers address physical constraints through sampling-time guidance and projection. ChipDiffusion and MacroDiff+ use gradient guidance (Lee et al., 2025; Yoon et al., 2026), DiffPlace combines energy guidance with constrained manifold diffusion (Trung & Hy, 2025), and FlowPlace applies hard constraint projection during sampling (Xie et al., 2026). For methods using dense pairwise overlap penalties, each evaluation requires $O(N^2)$ work for N placement objects, and this cost accumulates across repeated guidance and legalization steps.

3 Methodology

MacroSPA learns target spatial preferences from samples and reuses the resulting adaptation across circuits (Figure 1). At the **distribution level**, Spatial Prior Adaptation (SPA) adjusts how a pretrained placer uses spatial features in graph message passing and feature fusion (Section 3.1). At the **instance level**, the adapted model's parameters remain fixed while guided sampling, legalization, and residual refinement update the coordinates of movable objects (Section 3.2).

3.1 Spatial Prior Adaptation

Sample specified spatial priors. Let C denote the complete circuit and geometric context supplied to the placer, and let X_0 denote the coordinates of the movable objects represented by the model, including macros and optional standard-cell cluster proxies. A spatial prior describes systematic preferences in the conditional distribution of layouts given C, rather than prescribing fixed coordinates for individual macros. For example, target layouts may favor larger macros near the die periphery or a more compact macro arrangement. SPA acquires such preferences from a target corpus $\mathcal{D}_T = \{(C_n, X_{0,n})\}_{n=1}^{n_T}$ containing n_T context–placement pairs.

Target pairs can come from synthetic generators or placement optimizers. Following the generation order used in ChipDiffusion (Lee et al., 2025), we generate synthetic pairs by first sampling legal layouts under a prescribed spatial profile and then constructing connectivity and pins. This construction uses target-derived macro sizes and degree statistics while favoring short connections (Appendix B.3). Similarly, layouts generated by an optimizer can be projected into the model's object representation to form target pairs in the same format (Appendix D.4.3). Appendix A.1 describes the joint context–placement distribution induced by either source and the conditional layout distribution that encodes the target spatial prior. In both cases, SPA learns directly from these pairs without requiring additional quality scores or rankings.

Spatially conditioned adaptation. Our pretrained Spatial Base (hereafter Base) combines a GNN encoding circuit structure with a CNN encoding the current layout. At each denoiser evaluation, the CNN encodes separate occupancy maps of movable and fixed objects into feature maps at three resolutions. We bilinearly sample these feature maps at each object's current position and concatenate the resulting vectors to obtain $z_{i,t}$ (Appendix B.1). These features change as the layout evolves during denoising. SPA adapts how the model uses these spatial features in graph message passing and fuses them with the graph representation.

To let current geometry influence the information propagated through circuit connections, we insert an independent bottleneck residual adapter immediately before each of the two graph attention blocks. For block k, let $h_{i,t}^{(k)}$ and $\widetilde{h}_{i,t}^{(k)}$ denote node i's states before and after the adapter. With diffusion time embedding e_t and trainable adapter parameters ϕ_k, the update is

$$\widetilde{h}_{i,t}^{(k)} = h_{i,t}^{(k)} + a_{\phi_k}([z_{i,t}; e_t]). \tag{3.1}$$

The subsequent graph layers use these updated node representations for message passing over the circuit graph. At the decoder, we apply LoRA (Hu et al., 2022) to the first linear layer to adapt the fusion of the final GNN state with $z_{i,t}$ while keeping its pretrained weights frozen. The existing output head is also fine-tuned for coordinate-noise prediction. Together, these components parameterize a spatially conditioned correction to Base's denoising predictions.

We collect the trainable parameters in ϕ and keep all other pretrained parameters frozen. We denote the adapted denoiser by $\widehat{\varepsilon}_{B,\phi}$ and the pretrained denoiser by $\widehat{\varepsilon}_B$. The adapters and LoRA initially contribute zero, and the output head is initialized from Base, so $\widehat{\varepsilon}_{B,\phi_0} = \widehat{\varepsilon}_B$. In our design, SPA updates approximately 3% of the adapted model's parameters (Appendix B.2).

SPA training and reuse. For each target pair, we independently sample diffusion time $t \sim \mathcal{U}(0,1)$ and standard Gaussian noise ε, and form $X_t = \alpha_t X_0 + \sigma_t \varepsilon$ using the signal and noise coefficients inherited from Base (Appendix A.2). With $Y = (X_t, C, t)$, SPA minimizes the noise prediction loss (Ho et al., 2020)

$$\mathcal{L}_T(\phi) := \mathbb{E}_T\left[\|\varepsilon - \widehat{\varepsilon}_{B,\phi}(Y)\|_C^2\right], \tag{3.2}$$

where $\|\cdot\|_C^2$ averages over movable coordinate entries and the expectation $\mathbb{E}_T$ is taken over target pairs, diffusion times, and Gaussian noise. We estimate this expectation using target pairs from $\mathcal{D}_T$ and update only ϕ. Minimizing Eq. (3.2) is equivalent to approximating the target residual $\mathbb{E}_T[\varepsilon \mid Y] - \widehat{\varepsilon}_B(Y)$ with the induced correction $\widehat{\varepsilon}_{B,\phi}(Y) - \widehat{\varepsilon}_B(Y)$. Here, the error is measured in mean square under the target distribution, and the correction is restricted to SPA's correction class (Appendices A.2–A.3).

The learned SPA parameters $\widehat{\phi}_T$ are reused across circuits and placement runs within each target setting. During inference, all model parameters remain fixed while the adapted denoiser drives the reverse diffusion sampler. Appendix D.1.2 specifies the target corpora and training groups.

3.2 Constraint-Aware Inference

Minkowski guided proposals. Residual overlaps may remain after guided sampling. Further repair requires modifying the existing placement while controlling wirelength degradation. We develop Minkowski Guided Refinement (MGR), which generates local separation proposals and retains only

updates that satisfy exact geometric checks and a source-relative HPWL bound. When one rectangle's projection onto an axis lies strictly inside the other's, small relative displacements along that axis leave the exact intersection area unchanged. For axis-aligned rectangles, the interior of the Minkowski exclusion box contains exactly the relative center displacements that cause positive-area overlap (Lozano-Perez, 1983). MGR guides separation in the current layout X using a bilinear potential on the axial slacks of the exclusion box (Anjos & Vannelli, 2006):

$$\phi_{ij}(X) = \prod_{a\in\{x,y\}} \left[\frac{s_i^a + s_j^a}{2} - |x_i^a - x_j^a|\right]_+, \quad [u]_+ = \max\{u, 0\}, \tag{3.3}$$

where x_i^a and s_i^a are rectangle i's center coordinate and side length on axis a. In Figure 2, strict containment makes the intersection area A_{01} of m_0 and m_1 locally constant with respect to m_1's center x_1. Thus, $\nabla_{x_1} A_{01} = 0$, while $\nabla_{x_1}\phi_{01} \neq 0$ provides a separation signal. At each round, MGR optimizes movable coordinates to form a proposal from X, treating frozen active rectangles as obstacles. The proposal objective combines an area-normalized sum of pair potentials, a penalty for movable area outside the die, and an HPWL penalty anchored to the layout X_{src} entering MGR (Appendix C.1).

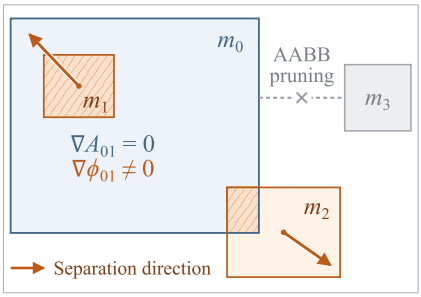


Figure 2: MGR separation and AABB pruning with physical query boxes.

Exact acceptance checks. Given a proposal $\widetilde{X}$, MGR tests $X_\alpha = X + \alpha(\widetilde{X} - X)$ for decreasing fractions $\alpha \in (0, 1]$. Candidates are evaluated on all active non-port rectangles, including frozen obstacles, independently of the proposal surrogate and its pair set. Let D denote the duplicate area ratio within the die and O the outside area ratio, both normalized by the total active footprint area. Let H denote HPWL in normalized chip coordinates. With outside area floor ϵ_O and current HPWL allowance η_k, MGR accepts the candidate X_α only if

$$D(X_\alpha) < D(X), \quad O(X_\alpha) \leq \max\{O(X), \epsilon_O\}, \quad H(X_\alpha) \leq (1 + \eta_k)H(X_{\text{src}}). \tag{3.4}$$

The first candidate satisfying these conditions is accepted, and the current layout is retained if none qualifies. Accepted updates reduce duplicate area subject to the outside area and HPWL limits. Keeping the HPWL reference fixed at X_{src} prevents the wirelength allowance from compounding across rounds. MGR may terminate with residual overlap or boundary violations. Appendices C.1 and C.3 specify the full procedure, numerical tolerances, and allowance schedule.

Size-adaptive constraint evaluation. All-pairs constraint evaluation during inference (including guidance, legalization, and MGR) incurs quadratic cost in the number of objects. For interactions among model objects, we use dense (all-pairs) evaluation on small inputs and sparse evaluation on large inputs via a GPU implementation of sweep and prune (Cohen et al., 1995). Query boxes are axis-aligned bounding boxes (AABBs) chosen to retain every pair contributing to the value or backward update. For smooth guidance and legalization, each physical box is expanded uniformly by a margin based on the smoothing parameter to retain pairs with nonzero penalties even without physical overlap. MGR uses physical boxes directly, since strictly separated pairs have zero potential and gradient, as illustrated by (m_0, m_3) in Figure 2. The candidate pair set is recomputed whenever the coordinates or the operator's support changes. This preserves pointwise values and backward updates in exact arithmetic, although candidate counts remain quadratic in the worst case. Appendices C.2–C.3 provide pseudocode, complexity analysis, and implementation settings.

4 Experiments

4.1 Experimental Setup

Training. We pretrain Spatial Base from random initialization on 40,000 synthetic graph–layout pairs following ChipDiffusion's data-generation procedure (Lee et al., 2025). Pretraining uses randomly sampled legal layouts without explicitly imposing a target spatial profile. All MacroSPA

variants start directly from the same pretrained checkpoint, without ChipDiffusion's additional full model fine-tuning on larger synthetic circuits. Motivated by placement practice (Pu et al., 2024), we construct synthetic targets for the primary comparisons on all benchmarks to favor the die periphery for the largest 25% of movable macros by area and mildly favor the interior for the rest. Unless otherwise specified, each synthetic SPA state is trained for 2,000 updates on 20 target pairs from a circuit group and remains fixed during inference. Appendix B.3 details synthetic target construction. Appendices D.1.1– D.1.2 describe Spatial Base pretraining and SPA training.

Benchmarks. Macro-only and downstream mixed-size evaluations cover the same 17 IBM circuits (Adya et al., 2004) and eight ISPD 2005 circuits (Nam et al., 2005), excluding ibm05 because it has no movable macros. Complete physical design evaluation uses the eight ChiPBench (Wang et al., 2025) circuits with the largest macro counts.

Comparisons. We evaluate DREAMPlace (Chen et al., 2023), WireMask-EA (Shi et al., 2023) (the best-performing variant reported in the original WireMask-BBO study), ChiPFormer (Lai et al., 2023), and ChipDiffusion (Lee et al., 2025) locally using the authors' released code and available checkpoints. Additional comparisons with FlowPlace (Xie et al., 2026) and MacroDiff+ (Yoon et al., 2026) are reported in Appendix E. All MacroSPA variants share inference and downstream settings. MacroSPA (w/o SPA) uses Base without SPA. MacroSPA (synthetic) uses SPA trained on synthetic targets, and MacroSPA (DP-teacher) uses SPA trained on DREAMPlace-generated targets.

Evaluation. We report HPWL and RUDY for macro-only placement as proxies for wirelength and routing demand, and weighted HPWL (wHPWL) after mixed-size placement with DREAMPlace. For methods supplying macro placements in IBM/ISPD mixed-size evaluation, DREAMPlace 4.1 places standard cells with macros fixed. The DREAMPlace baseline uses its native mixed-size flow. For ChiPBench, macro layouts from all methods undergo common legalization before entering the official evaluation flow, where OpenROAD completes standard-cell placement and subsequent physical implementation. We report routing and post-route PPA metrics. Appendix D.1.3 details benchmark conventions, metrics, and method-specific initialization.

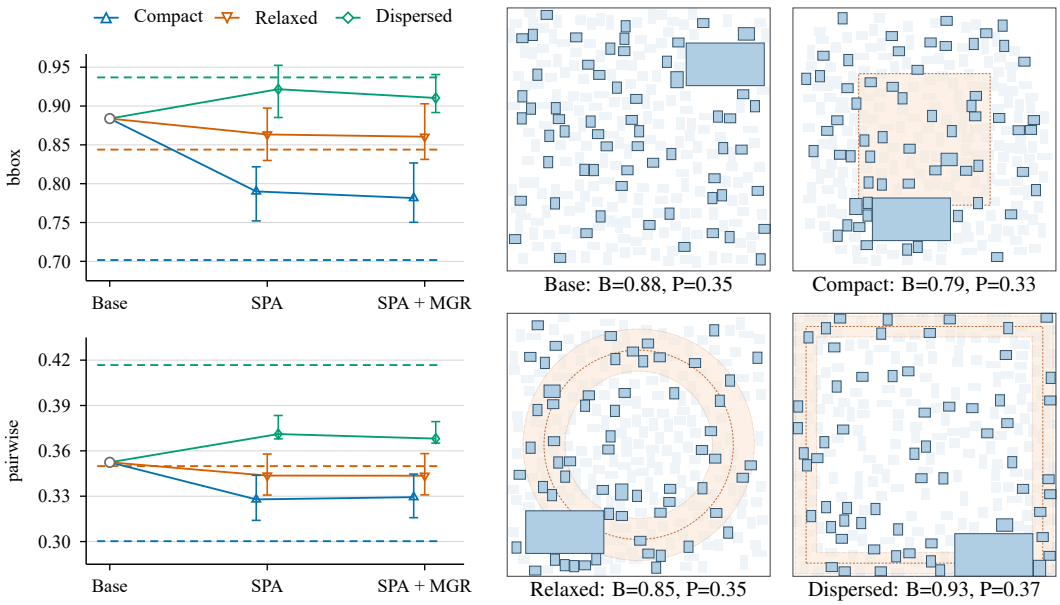
4.2 Effectiveness of Spatial Prior Adaptation

Target denoising risk reduction. We first test whether SPA improves denoising relative to the frozen Spatial Base, using held-out target layouts generated under the peripheral profile described in Section 4.1. Across five adaptation groups at 1K and 2K updates, SPA reduces denoising risk by 11.98%–19.40%. Under the decomposition in Appendix A.2, these reductions provide evidence that the learned corrections approximate the base-relative target residual more accurately than the zero correction does, with accuracy measured in mean square under the target distribution. Appendix D.2.1 provides the protocol and full results.

Spatial prior acquisition under matched contexts. To test whether generated layouts reflect the target spatial preferences, we adapt the same Base to Compact, Relaxed (annular), and Dispersed target corpora that share the same circuit and geometric contexts but differ only in movable macro coordinates. We evaluate five IBM circuits with guidance and legalization disabled. For the largest 25% of movable macros by area, we measure the bounding box area of their centers (bbox) and their average pairwise distance (pairwise). SPA outputs follow the target ordering Compact $<$ Relaxed $<$ Dispersed in both metrics before and after MGR (Figure 3a), with representative refined layouts shown in Figure 3b. Specifically, Compact decreases and Dispersed increases both metrics relative to the matched Base, while Relaxed improves annular alignment and angular uniformity. These results show that SPA enables the same pretrained model to adapt to different target priors. Appendix D.2.2 provides the evaluation protocol, metric definitions, and statistical results.

Component controls. We next examine the contributions of SPA's trainable components through ablations on IBM G1, Adaptec, and Bigblue. The ablations show that both the residual adapters and fusion LoRA contribute to improved placement quality. Full SPA achieves the lowest average HPWL and RUDY and the highest area-based geometry scores among the evaluated configurations, supporting the use of both pathways in the complete adaptation scheme (Appendix D.2.3). Rank and training-budget sensitivity results are provided in Appendix D.2.5.

Quality and adaptation cost. We then compare SPA with full fine-tuning (Full-FT) from the same Spatial Base using matched target data. At a matched budget of 1K updates, SPA reduces average



(a) Spatial responses. (b) Layouts after MGR.

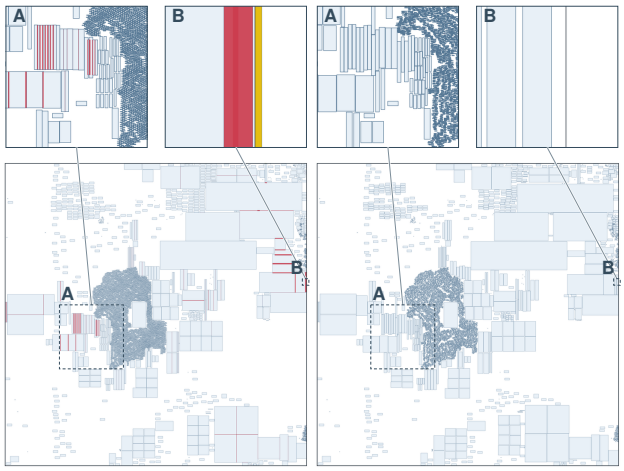
Figure 3: Spatial prior acquisition under matched contexts. Target corpora differ only in movable macro coordinates. (a) Average bounding box area (B) and pairwise distance (P) of macro centers. Error bars indicate 95% confidence intervals, and dashed lines denote target averages. (b) Representative layouts after MGR. Both metrics use the largest 25% of movable macros by area, shown with dark outlines. Orange shading illustrates their target spatial preferences.

macro HPWL by 4.40% and RUDY by 1.69%, while requiring less training time and peak GPU memory. Extending Full-FT to 20K updates does not close the quality gap under the evaluated training configuration. These results show that useful adaptation of pretrained spatial behavior can be achieved without full model updates in this setting. SPA achieves better placement quality at lower adaptation cost while updating approximately 3% as many parameters as Full-FT. Appendix D.2.4 provides the evaluation protocol, quality curves, and resource measurements.

4.3 Scalable Constraint-Aware Inference

Residual geometric refinement. We compare MGR with continued execution of ChipDiffusion's legalization procedure on eight ISPD circuits, using layouts generated with and without SPA. In each paired comparison, both methods start from the same post-legalization layout and use 400 optimization updates, with MGR performing 20 rounds of 20 proposal updates. Relative to the starting layouts, MGR reduces the average duplicate area ratio by 74.60% without SPA and 57.54% with SPA, versus less than 0.3% for continued legalization. MGR also achieves lower final average outside area ratios and HPWL than continued legalization, supporting its use as a dedicated stage for residual geometric repair. Figure 4 illustrates representative repairs, and Appendix D.3.1 provides the paired protocol and complete results.

Size-adaptive constraint evaluation. To assess inference efficiency across circuit sizes, we compare the dense and sparse MacroSPA routes under matched settings on four circuits, using GPU sweep and prune for sparse evaluation. Other placement methods serve as external baselines. All reported results are averaged over three runs per circuit and configuration. WireMask-EA is evaluated on the CPU, so peak GPU memory is not applicable. As shown in Figure 5, dense evaluation is faster on smaller circuits, whereas sparse evaluation is substantially faster on larger ones. For the full inference pipeline on bigblue2 (23,084 objects), MacroSPA with sparse evaluation achieves speedups of $32.82\times$ over dense evaluation and $43.35\times$ over ChipDiffusion. Peak GPU memory usage drops by 89.07% versus dense and 89.11% versus ChipDiffusion. Final HPWL is lower than ChipDiffusion's on three circuits and 2.41% higher on bigblue2. Appendix D.3.2 breaks down the speedups over the dense MacroSPA route and ChipDiffusion by inference stage.



(a) Before MGR (b) After MGR

Figure 4: Post-legalization placements on bigblue4. Red denotes overlap and yellow denotes regions outside the die. MGR resolves the highlighted violations while largely preserving the overall placement structure.

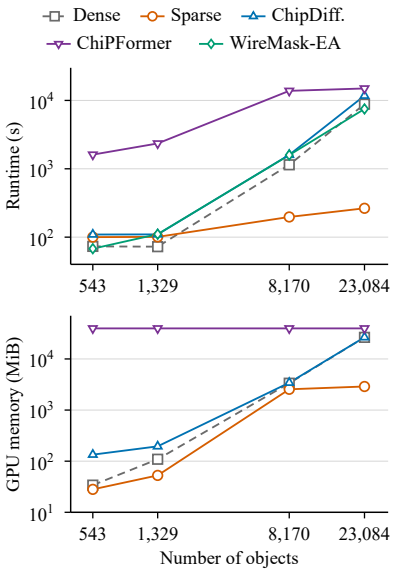


Figure 5: Average runtime and peak GPU memory across four circuit sizes.

Table 1: Comparison of average macro-only HPWL (10^5), RUDY (10^2), and downstream wHPWL (10^6) on IBM and ISPD. Lower is better. Best results are in bold, and the second best are underlined.

Method	IBM			ISPD		
	HPWL	RUDY	wHPWL	HPWL	RUDY	wHPWL
DREAMPlace	2.79	1.88	**22.88**	47.87	6.99	**196.10**
WireMask-EA	5.15†	2.50†	28.69†	157.49	14.66	243.40
ChiPFormer	4.22	2.36	29.95	137.05	15.26	302.62
ChipDiffusion	2.60	1.77	26.47	<u>45.47</u>	5.74	243.49
MacroSPA (w/o SPA)	<u>2.58</u>	<u>1.70</u>	26.50	46.70	<u>5.62</u>	237.42
MacroSPA (synthetic)	**2.49**	**1.62**	<u>25.48</u>	**42.33**	**5.29**	<u>232.10</u>

† Averages include imputed values. See Appendix D.1.3 for reporting conventions.

4.4 Physical Design Utility

We finally assess how target samples affect placement quality and downstream physical design outcomes across settings and objectives. Comparisons with existing placers provide a performance reference, whereas comparisons with the no-SPA pipeline and between SPA target sources evaluate the practical value of revising spatial priors in the same pretrained Base.

Macro-only and mixed-size placement. On IBM and ISPD, we evaluate SPA trained on the primary synthetic peripheral targets, using macro-only HPWL and RUDY together with downstream mixed-size wHPWL. Results in Table 1 are averaged over three independent runs per circuit and method. Relative to the matched no-SPA pipeline, SPA lowers macro-only HPWL on 24 of 25 circuits and achieves the lowest average HPWL and RUDY among the compared methods on both benchmarks. It also reduces average downstream wHPWL by 3.83% on IBM and 2.24% on ISPD, with improvements on 22 of 25 circuits. By average downstream wHPWL, it ranks second to the native DREAMPlace flow on both benchmarks. Appendix D.4.1 provides circuit-level results and geometry scores, and Appendix E extends the comparison on ISPD to FlowPlace and MacroDiff+.

Target prior choice. To assess how target sample profiles affect placement utility, we compare SPA trained on Peripheral, Random, and Clustered samples under shared inference and downstream settings. Random sequentially samples each macro's origin uniformly from currently legal positions, while Clustered favors the interior. On both benchmarks, Random and Clustered yield lower macro-

Table 2: Comparison of average routing and post-route PPA ranks on ChiPBench. Ranks are computed per circuit among the methods shown and then averaged across eight circuits. The last column averages the seven metric ranks. Lower is better. Best results are in bold, and the second best are underlined, including ties.

Method	Wirelength	Cong.	WNS	TNS	NVP	Power	Area	Average
DREAMPlace	**2.00**	**2.13**	3.25	3.63	<u>3.25</u>	4.69	<u>3.38</u>	<u>3.19</u>
WireMask-EA	5.38	4.88	6.25	4.75	4.31	4.13	5.25	4.99
ChiPFormer	3.88	4.38	5.00	5.00	5.50	4.69	5.38	4.83
ChipDiffusion	5.00	4.75	5.00	4.00	<u>3.25</u>	<u>3.63</u>	3.63	4.18
MacroSPA (w/o SPA)	4.13	<u>3.88</u>	3.75	5.00	5.25	4.38	3.88	4.32
MacroSPA (synthetic)	<u>3.75</u>	4.00	<u>2.50</u>	**2.63**	**2.31**	**2.50**	**2.63**	**2.90**
MacroSPA (DP-teacher)	3.88	4.00	**2.25**	<u>3.00</u>	4.13	4.00	3.88	3.59

only HPWL but higher downstream wHPWL than Peripheral. A plausible reason is that placing large macros near the periphery helps preserve more contiguous space for standard-cell placement and routing (Pu et al., 2024), giving the downstream placer more freedom to keep connected cells close. This reversal shows that the relative utility of the evaluated priors depends on the placement setting, motivating adaptation of spatial preferences to the intended downstream use (Appendix D.4.2).

Optimizer-generated targets. We next test whether optimizer-generated placement samples can specify useful target priors, extending acquisition beyond explicitly prescribed spatial profiles. We apply the same SPA training procedure to DREAMPlace placements projected into our clustered representation. The projected targets include coordinates for both movable macros and standard-cell clusters. Like synthetic targets, these projected samples specify a conditional layout distribution for SPA training (Appendix A.1). We train a single shared SPA state on targets from ibm01–09 and evaluate it on all 17 circuits. Compared with SPA trained on the primary synthetic targets, DP-teacher SPA further reduces average downstream wHPWL by 3.71% on ibm01–09 and 2.87% on ibm10–18. These results show that the same SPA framework can also learn useful placement behavior directly from optimizer-generated samples, with downstream improvements extending to circuits excluded from adapter training. Appendix D.4.3 provides the protocol and complete results.

Routing and post-route PPA. We evaluate whether SPA's benefits extend beyond placement-level proxies to routing and post-route PPA on ChiPBench. DP-teacher is trained on targets from six of the eight evaluated circuits. Both SPA variants improve the overall average rank over the matched no-SPA pipeline, with synthetic SPA ranking best among the evaluated methods (Table 2). Between the two variants, DP-teacher ranks better on WNS, whereas synthetic SPA ranks better on TNS and standard-cell area. Appendix D.4.4 provides details and circuit-level results, and Appendix E presents additional comparisons on ChiPBench with FlowPlace and MacroDiff+.

The different post-route strengths of the two SPA variants, together with the ranking reversal between macro-only and mixed-size placement, suggest that a fixed spatial prior may not be uniformly preferable across downstream objectives. These findings and the gains over the matched no-SPA pipeline support adapting inherited spatial preferences through target samples while retaining the same pretrained base model.

5 Conclusion

In this paper, we presented MacroSPA, a post-training framework that enables the same pretrained diffusion-based macro placer to acquire distinct spatial priors directly from target context–placement samples. To this end, SPA approximates a denoising correction relative to the pretrained Spatial Base while updating 3.03% of the adapted model's parameters and keeping the backbone frozen. The learned SPA parameters are reused across circuits within each target setting. During inference, MGR and size-adaptive constraint evaluation complement this adaptation by reducing residual geometric conflicts and improving efficiency on large instances. Experiments show that the acquired priors can improve downstream physical design outcomes over the matched no-SPA pipeline. Target priors have contrasting effects on macro wirelength and downstream physical design, highlighting the importance of choosing target samples for the intended design objectives.

AI USE STATEMENT

In this work, we used generative AI tools to discuss alternative model architectures and assist with selected aspects of experimental setup and implementation. AI agents were also used to monitor experiment runs and report their progress and execution status. We used these tools to assist with code review and debugging. For manuscript preparation, generative AI tools assisted with drafting and language polishing. All AI-assisted content and research outputs were independently examined and validated by the authors. Code reviewed or modified with AI assistance was checked by the authors and tested through experiments. We take responsibility for the final content of this work, including all text, claims, results, and artifacts produced with the aid of generative AI.

REFERENCES

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

Miguel F Anjos and Anthony Vannelli. A new mathematical-programming framework for facility-layout design. *INFORMS Journal on Computing*, 18(1):111–118, 2006.

Yifan Chen, Zaiwen Wen, Yun Liang, and Yibo Lin. Stronger mixed-size placement backbone considering second-order information. In *Proceedings of the 2023 IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–9. IEEE, 2023.

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, 2019.

Jonathan D Cohen, Ming C Lin, Dinesh Manocha, and Madhav Ponamgi. I-COLLIDE: An interactive and exact collision detection system for large-scale environments. In *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, pp. 189–196, 218, 1995.

Chengrui Gao, Yunqi Shi, Ke Xue, Ruo-Tong Chen, Siyuan Xu, Mingxuan Yuan, Chao Qian, and Zhi-Hua Zhou. Expertise can be helpful for reinforcement learning-based macro placement. In *International Conference on Learning Representations*, 2026.

Zijie Geng, Jie Wang, Ziyan Liu, Siyuan Xu, Zhentao Tang, Shixiong Kai, Mingxuan Yuan, Jianye Hao, and Feng Wu. LaMPlace: Learning to optimize cross-stage metrics in macro placement. In *International Conference on Learning Representations*, 2025.

Trevor Hastie, Robert Tibshirani, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, second edition, 2009.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Andrew B. Kahng and Tom Spyrou. The OpenROAD project: Unleashing hardware innovation. In *Government Microcircuit Applications and Critical Technology Conference*, 2021.

Andrew B Kahng, Ravi Varadarajan, and Zhiang Wang. Hier-RTLMP: A hierarchical automatic macro placer for large-scale complex IP blocks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 43(5):1552–1565, 2024.

George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. Multilevel hypergraph partitioning: Application in VLSI domain. In *Proceedings of the 34th Design Automation Conference*, pp. 526–529, 1997.

Yao Lai, Yao Mu, and Ping Luo. MaskPlace: Fast chip placement via reinforced visual representation learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24019–24030, 2022.

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*, volume 202 of *Proceedings of Machine Learning Research*, pp. 18346–18364. PMLR, 2023.

Vint Lee, Minh Nguyen, Leena Elzeiny, Chun Deng, Pieter Abbeel, and John Wawrzynek. Chip placement with diffusion models. In *International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 33499–33514. PMLR, 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 Design Automation Conference*, pp. 117:1–117:6. Association for Computing Machinery, 2019.

Tomas Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32(2):108–120, 1983.

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.

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*, pp. 216–220, 2005.

Ling Pan, Qingpeng Cai, Qi Meng, Wei Chen, and Longbo Huang. Reinforcement learning with dynamic Boltzmann softmax updates. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pp. 1992–1998. International Joint Conferences on Artificial Intelligence Organization, 2020.

Yuan Pu, Tinghuan Chen, Zhuolun He, Chen Bai, Haisheng Zheng, Yibo Lin, and Bei Yu. IncreMacro: Incremental macro placement refinement. In *Proceedings of the 2024 International Symposium on Physical Design*, pp. 169–176, 2024.

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

Peter Spindler and Frank M. Johannes. Fast and accurate routing demand estimation for efficient routability-driven placement. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, pp. 1226–1231. IEEE, 2007.

Kien Le Trung and Truong-Son Hy. DiffPlace: A conditional diffusion framework for simultaneous VLSI placement beyond sequential paradigms. *arXiv preprint arXiv:2510.15897*, 2025.

Zhihai Wang, Zijie Geng, Zhaojie Tu, Jie Wang, Yuxi Qian, Zhexuan Xu, Ziyan Liu, Siyuan Xu, Zhentao Tang, Shixiong Kai, et al. Benchmarking end-to-end performance of AI-based chip placement algorithms. In *Advances in Neural Information Processing Systems*, volume 38, pp. 22670–22696, 2025.

Peng Xie, Ke Xue, Yunqi Shi, Ruo-Tong Chen, Chengrui Gao, Siyuan Xu, Chenjian Ding, Mingxuan Yuan, and Chao Qian. FlowPlace: Flow matching for chip placement. In *Proceedings of the 63rd Design Automation Conference*, 2026.

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. In *Advances in Neural Information Processing Systems*, volume 37, pp. 140565–140588, 2024.

Jongho Yoon, Jinsung Jeon, and Seokhyeong Kang. Physics-guided geometric diffusion for macro placement generation. In *Proceedings of the Thirty-Fifth International Joint Conference on Artificial Intelligence*, pp. 6609–6616, 2026.

A Spatial Prior Acquisition

This appendix formalizes how a pretrained macro placer acquires spatial priors from target context–placement samples.

A.1 Sample Specified Priors

Let the target samples follow

$$(C, X_0) \sim \mathbb{P}_T, \quad \mathbb{P}_T(\mathrm{d}c, \mathrm{d}x_0) = \mu_T(\mathrm{d}c) Q_T(\mathrm{d}x_0 \mid c), \tag{A.1}$$

where C is the complete circuit and geometric context supplied to the placer and X_0 contains the coordinates of represented movable objects, including standard-cell cluster proxies when used. Here μ_T is the distribution of target contexts. The notation $\mathrm{d}c$ and $\mathrm{d}x_0$ indicates the integration variables for the corresponding probability measures, which need not admit densities. We use spatial prior to denote systematic placement preferences encoded by the conditional layout distribution $Q_T(\cdot \mid c)$.

Target spatial organization may be specified by a synthetic generator or reflected in optimizer-generated layouts. Each source induces a target joint distribution $\mathbb{P}_T$ and supplies completed context–placement pairs to the same SPA training procedure.

To relate sample construction to the conditional layout distribution used for adaptation, let $C = (S, E)$, where S denotes the instantiated scaffold, including object sizes and component assignments, and E contains the remaining connectivity and pin context. The source-induced joint distribution admits two factorizations:

$$\mathbb{P}_T(\mathrm{d}s, \mathrm{d}e, \mathrm{d}x_0) = \mu_T(\mathrm{d}s, \mathrm{d}e) Q_T(\mathrm{d}x_0 \mid s, e) = \lambda_T(\mathrm{d}s) \Pi_T(\mathrm{d}x_0 \mid s) K_T(\mathrm{d}e \mid x_0, s).$$

Here λ_T is the scaffold distribution, $\Pi_T(\cdot \mid s)$ is the layout distribution conditioned only on $S = s$, and K_T describes connectivity and pins given placement and scaffold. The first factorization describes the target layout distribution conditioned on the complete context. The second expresses the same joint distribution in terms of sample construction, connecting synthetic context–placement samples to the target distribution used for SPA training. For synthetic samples, Q_T is obtained by conditioning Π_T on the connectivity and pins generated from the layout. Appendix B.3 details the sample construction procedure.

A.2 Base Relative Correction

For a context $C = c$, the target layout is represented by coordinates $X_0 \in \mathbb{R}^{d(c)}$, with

$$X_0 \sim Q_T(\cdot \mid c), \quad d(c) = 2|V_{\mathrm{mov}}(c)|. \tag{A.2}$$

Here $V_{\mathrm{mov}}(c)$ is the set of represented movable objects. Assume $0 < d(C) < \infty$ almost surely and order their coordinates by a deterministic, context-measurable rule. For each target pair $(C = c, X_0)$, we independently draw $t \sim \mathcal{U}(0, 1)$ and $\varepsilon \sim \mathcal{N}(0, I_{d(c)})$, and apply Gaussian forward corruption (Ho et al., 2020) with the schedule inherited from the pretrained base:

$$X_t = \alpha_t X_0 + \sigma_t \varepsilon, \quad \alpha_t = \cos\left(\frac{\pi t}{2}\right), \quad \sigma_t = \sin\left(\frac{\pi t}{2}\right). \tag{A.3}$$

For $u, v \in \mathbb{R}^{d(c)}$, define the following inner product and its induced norm

$$\langle u, v \rangle_c := \frac{1}{d(c)} u^\top v, \quad \|u\|_c^2 := \langle u, u \rangle_c. \tag{A.4}$$

The normalization removes loss weighting due solely to layout dimension. Assume the noise, base prediction, and corrections below are square-integrable under this norm. Let $Y = (X_t, C, t)$, and let $\mathbb{E}_T$ denote expectation over target pairs, diffusion time, and Gaussian noise.

For a fixed pretrained base predictor $\widehat{\varepsilon}_B$ and any square-integrable additive correction Δ, define

$$\mathcal{R}_T[\Delta] := \mathbb{E}_T\left[\|\varepsilon - \widehat{\varepsilon}_B(Y) - \Delta(Y)\|_C^2\right]. \tag{A.5}$$

To characterize the correction associated with Q_T, define

$$m_T(Y) := \mathbb{E}_T[\varepsilon \mid Y], \quad r_T(Y) := m_T(Y) - \widehat{\varepsilon}_B(Y). \tag{A.6}$$

Under squared loss, m_T is the optimal target denoiser (Hastie et al., 2009), and r_T is the optimal additive correction relative to the fixed base. For fixed c and $t \in (0,1)$, Bayes' rule applied to Equation (A.3) gives the posterior distribution of clean layouts:

$$\mathbb{P}_T(\mathrm{d}x_0 \mid X_t = x_t, C = c, t) = \frac{\exp\left[-\frac{\|x_t - \alpha_t x_0\|_2^2}{2\sigma_t^2}\right] Q_T(\mathrm{d}x_0 \mid c)}{\displaystyle\int_{\mathbb{R}^{d(c)}} \exp\left[-\frac{\|x_t - \alpha_t u\|_2^2}{2\sigma_t^2}\right] Q_T(\mathrm{d}u \mid c)}.$$

Here $\|\cdot\|_2$ is the Euclidean norm. Taking conditional expectations of $\varepsilon = (X_t - \alpha_t X_0)/\sigma_t$ yields

$$m_T(x_t, c, t) = \frac{x_t - \alpha_t \mathbb{E}_T[X_0 \mid X_t = x_t, C = c, t]}{\sigma_t}.$$

For every square-integrable correction Δ, the denoising risk decomposes as

$$\mathcal{R}_T[\Delta] = \mathcal{R}_T^\star + \mathbb{E}_T\left[\|r_T(Y) - \Delta(Y)\|_C^2\right], \quad \mathcal{R}_T^\star := \mathbb{E}_T\left[\|\varepsilon - m_T(Y)\|_C^2\right]. \tag{A.7}$$

Here $\mathcal{R}_T^\star$ is the minimum target denoising risk over all square-integrable predictors, attained by m_T. To obtain Equation (A.7), write the error as $(\varepsilon - m_T(Y)) + (r_T(Y) - \Delta(Y))$. The expected cross term vanishes because $\varepsilon - m_T(Y)$ has conditional mean zero given Y, while $r_T(Y) - \Delta(Y)$ and $d(C)^{-1}$ are Y-measurable. The decomposition also holds for other fixed time-sampling distributions under the same integrability assumptions, with expectations taken accordingly.

A.3 Sample-Based Training

Let Φ_{SPA} denote the parameter space of SPA's trainable components specified in Appendix B.2, and let $\widehat{\varepsilon}_{B,\phi}$ denote the adapted denoiser for $\phi \in \Phi_{\mathrm{SPA}}$. Relative to the fixed pretrained predictor $\widehat{\varepsilon}_B$, define

$$\begin{aligned} \Delta_\phi(Y) &:= \widehat{\varepsilon}_{B,\phi}(Y) - \widehat{\varepsilon}_B(Y), \\ \mathcal{F}_{\mathrm{SPA}} &:= \{\widehat{\varepsilon}_{B,\phi} - \widehat{\varepsilon}_B : \phi \in \Phi_{\mathrm{SPA}}\}. \end{aligned} \tag{A.8}$$

By Equation (A.7), the expected target denoising loss satisfies

$$\begin{aligned} \mathcal{L}_T(\phi) &:= \mathbb{E}_T\left[\|\varepsilon - \widehat{\varepsilon}_{B,\phi}(Y)\|_C^2\right] \\ &= \mathcal{R}_T[\Delta_\phi] \\ &= \mathcal{R}_T^\star + \mathbb{E}_T\left[\|r_T(Y) - \Delta_\phi(Y)\|_C^2\right]. \end{aligned} \tag{A.9}$$

Since $\mathcal{R}_T^\star$ is independent of ϕ, minimizing this loss over ϕ is equivalent to approximating r_T within $\mathcal{F}_{\mathrm{SPA}}$ in mean square under the target distribution. For target pairs $\mathcal{D}_T = \{(C_i, X_{0,i})\}_{i=1}^{n_T}$, the empirical objective is

$$\begin{aligned} \widehat{\mathcal{L}}_T(\phi) &= \frac{1}{n_T}\sum_{i=1}^{n_T} \mathbb{E}_{t_i,\varepsilon_i}\left[\|\varepsilon_i - \widehat{\varepsilon}_{B,\phi}(Y_i)\|_{C_i}^2\right], \\ Y_i &= (\alpha_{t_i} X_{0,i} + \sigma_{t_i}\varepsilon_i, C_i, t_i). \end{aligned} \tag{A.10}$$

Here t_i and ε_i are sampled as in Appendix A.2. For a fixed circuit context, $Q_T(\cdot \mid c)$ specifies preferences over layouts under the given connectivity and geometry. Through the conditional noise mean defined in Equation (A.6), these preferences determine the correction r_T relative to the fixed pretrained placer. To approximate this correction within $\mathcal{F}_{\mathrm{SPA}}$, SPA trains the adapted denoiser to predict the sampled noise ε_i.

B SPA Implementation

B.1 Spatial Base Architecture

Spatial Base combines a GNN with a CNN that encodes the current layout. The GNN contains two graph attention blocks and maps node states, object dimensions, fixed indicators, connectivity, and pin context to 96-dimensional node features. A parameter-free sinusoidal encoder maps diffusion time to $e_t \in \mathbb{R}^{32}$, which conditions the graph blocks through feature-wise affine modulation.

At each denoiser evaluation, the current rectangles are rasterized onto a 32×32 grid over the normalized die. Movable positions come from X_t, and fixed positions from C. For grid cell $\mathcal{G}_u$, object rectangle R_i^t, and fixed indicator f_i, the two occupancy channels are

$$O_t^{\mathrm{mov}}(u) = \sum_i (1 - f_i) \frac{|R_i^t \cap \mathcal{G}_u|}{|\mathcal{G}_u|}, \quad O_t^{\mathrm{fix}}(u) = \sum_i f_i \frac{|R_i^t \cap \mathcal{G}_u|}{|\mathcal{G}_u|}. \tag{B.1}$$

Occupancy fractions are summed without clipping. The CNN processes these maps through three convolutional blocks with SiLU activations and average pooling between blocks, producing 32×32, 16×16, and 8×8 feature maps with 16, 32, and 64 channels. Bilinear sampling at the current object position p_i^t gives

$$z_{i,t} = \mathop{\Big\|}_{s \in \{32,16,8\}} \mathrm{Sample}\Big(F_t^{(s)}, p_i^t\Big) \in \mathbb{R}^{112}, \tag{B.2}$$

where $F_t^{(s)}$ has resolution $s \times s$ and $\|$ denotes concatenation. Sampling uses zero padding and maps the normalized die boundaries to the outer edges of the feature-map grid.

The decoder fuses the final GNN state $h_{i,t}^B \in \mathbb{R}^{96}$ with $z_{i,t}$ through a $208 \to 128 \to 2$ MLP. Let Head_B denote its output layer. Including the inherited input–output residual, the noise prediction for represented movable object i is

$$\big[\widehat{\varepsilon}_B(Y)\big]_i = (X_t)_i + \mathrm{Head}_B\big(\mathrm{SiLU}\big(W_{\mathrm{fus}}[h_{i,t}^B; z_{i,t}] + b_{\mathrm{fus}}\big)\big), \quad Y = (X_t, C, t).$$

Here W_{fus} and b_{fus} are the weights and bias of the decoder's first linear layer. Fixed positions remain constant during diffusion, and the training loss is averaged over movable coordinates. Appendix D.1.1 specifies pretraining.

B.2 SPA Modules and Training

SPA adapts this predictor through two residual adapters, fusion LoRA factors, and the existing output head. Their trainable parameters are collected in ϕ, while all other pretrained parameters remain frozen. Both adapters use the same spatial features as the decoder and take $u_{i,t} = [z_{i,t}; e_t] \in \mathbb{R}^{144}$ as input. An independent residual adapter is inserted immediately before each of the two graph attention blocks. Each adapter applies

$$\begin{aligned} \Delta h_{i,t} &= W_2\, \mathrm{SiLU}(W_1 u_{i,t} + b_1) + b_2, \\ \widetilde{h}_{i,t} &= h_{i,t} + \Delta h_{i,t}, \end{aligned} \tag{B.3}$$

where $W_1 \in \mathbb{R}^{32 \times 144}$, $W_2 \in \mathbb{R}^{96 \times 32}$, and b_1, b_2 are bias vectors. In the decoder, we apply LoRA (Hu et al., 2022) with rank $r = 8$ and scaling parameter $\alpha = 16$ to the first linear layer:

$$W_{\text{fus}}^{\text{SPA}} = W_{\text{fus}} + \frac{\alpha}{r} U_\phi V_\phi, \quad V_\phi \in \mathbb{R}^{r\times 208}, \quad U_\phi \in \mathbb{R}^{128\times r}. \tag{B.4}$$

The output head Head_ϕ is initialized from Head_B, with both weights and bias trainable. We use PyTorch's default linear-layer initialization for V_ϕ and for W_1 and b_1 in both adapters. We initialize W_2, b_2, and U_ϕ to zero, so the adapted predictor initially matches the base. Its prediction $\widehat{\varepsilon}_{B,\phi}$ follows the expression in Appendix B.1 with the final state of the adapted GNN, $W_{\text{fus}}^{\text{SPA}}$, and Head_ϕ replacing their base counterparts. Equation (A.8) defines the resulting base relative correction. In our configuration, SPA adds 18,304 parameters and fine-tunes the existing output head. Together, these trainable parameters account for 3.03% of the adapted model's parameters.

To acquire the target spatial prior, SPA trains the adapted denoiser on completed target pairs (C, X_0) from synthetic or optimizer-generated sources. Training updates only $\phi \in \Phi_{\text{SPA}}$ using the noise-prediction objective in Equation (A.10). The trained parameters $\widehat{\phi}_T$ are reused across circuits and placement runs. During inference, the adapted predictor drives the inherited diffusion sampler with these parameters fixed. Appendix D.1.2 specifies the training protocol, and Appendix D.4.3 describes the construction and data split of optimizer-generated targets.

B.3 Synthetic Target Generation

Constructing a legal layout with low wirelength for a fixed topology requires costly optimization. Synthetic pairs can be generated more directly by sampling a legal layout first and then constructing a topology that favors short connections. ChipDiffusion (Lee et al., 2025) uses random legal layouts, and FlowPlace (Xie et al., 2026) samples legal positions according to spatial scores.

For MacroSPA, Algorithm B.1 generates synthetic targets from a macro graph G after self-loop removal. To vary graph size across training samples, we optionally prune a small subset of macro nodes, preferentially selecting small macros. Synthesis preserves the resulting graph's macro-size multiset and each weak component's multiset of out-/in-degree pairs. The N retained macros are placed in decreasing area order on a 512×512 grid over $[-1, 1]^2$, with each rectangle centered in its rounded-up grid footprint. At area rank ℓ, $\mathcal{L}_\ell$ contains in-die origins avoiding placed footprints. The spatial profile $\tau_{\text{spa}} = (\rho, b, p)$ specifies the cutoff fraction, boundary bias, and interior bias:

$$q_\ell(o) \propto \begin{cases} \big(1 + \text{clearance}_\ell(o)\big)^{-b}, & \ell \le \lceil \rho N \rceil, \\ \big(1 + \text{clearance}_\ell(o)\big)^{p}, & \ell > \lceil \rho N \rceil, \end{cases} \quad o \in \mathcal{L}_\ell, \tag{B.5}$$

where $\text{clearance}_\ell(o)$ is the footprint's minimum gap to the die boundary in grid cells.

Algorithm B.1 Target macro synthesis

Input: Target macro graph G, spatial profile $\tau_{\text{spa}} = (\rho, b, p)$
Output: Circuit context C and legal placement X_0

Remove self-loops from G, optionally prune nodes, and extract sizes and component-wise degree multisets.
Randomly assign sizes, component memberships, and degree roles to N macros.
repeat
  Clear occupancy and initialize X_0.
  for $\ell = 1, \ldots, N$ in decreasing macro area **do**
    Construct the legal origin set $\mathcal{L}_\ell$ from the current occupancy.
    if $\mathcal{L}_\ell = \varnothing$ **then**
      break ▷ Restart placement
    end if
    Sample $o_\ell \sim q_\ell$ on $\mathcal{L}_\ell$ using Eq. (B.5).
    Record the footprint center in X_0 and update occupancy.
  end for
until all N macros have been placed
Sample loopless edges within each assigned component, preserving its degree roles.
Rewire edges to favor shorter connections under X_0, preserving degrees and excluding self-loops.
Make each assigned component weakly connected using loopless, degree-preserving swaps.
Sample owner-relative pins and assemble C.
return (C, X_0)

Successful pairs (C, X_0), mixed according to the target corpus's circuit proportions, define $\mathbb{P}_T$ (Appendix A.1) and provide SPA training data.

C Constraint-Aware Inference

This appendix details the implementation of constraint-aware inference, covering residual refinement, size-adaptive constraint evaluation, and their experimental settings.

C.1 Residual Refinement

For a fixed context C, let X denote the movable object coordinates. Geometric quantities are evaluated on the full layout, including fixed object positions supplied by C.

Minkowski Guided Refinement (MGR) refines the output of the guided sampling and legalization procedures inherited from ChipDiffusion (Lee et al., 2025). It combines a geometric separation penalty with acceptance checks on overlap, outside area, and wirelength.

Proposal objective. For two axis-aligned rectangles, the Minkowski sum of their footprints centered at the origin defines an exclusion box whose interior contains exactly the relative center displacements that produce an overlap of positive area (Lozano-Perez, 1983). MGR uses the following bilinear penalty on the axial exclusion slacks to guide local separation (Anjos & Vannelli, 2006):

$$\phi_{ij}(X) = \prod_{a\in\{x,y\}} \left[\frac{s_i^a + s_j^a}{2} - |x_i^a - x_j^a|\right]_+, \quad [u]_+ = \max\{u, 0\}, \tag{C.1}$$

where $x_i = (x_i^x, x_i^y)$ is the center of axis-aligned rectangle i, and s_i^a is its side length along axis a, in normalized chip coordinates. Under containment along an axis, this penalty can still provide a separating signal even when exact intersection area is locally constant along that axis.

Let $\mathcal{V}_{\text{act}}$ contain the non-port rectangles included in geometric evaluation, and let $\mathcal{F}$ contain ports, fixed objects, blockages, and explicitly frozen objects. The set of movable rectangles is then $\mathcal{V}_{\text{mov}} = \mathcal{V}_{\text{act}} \setminus \mathcal{F}$. The eligible set $\mathcal{P}_{\text{MGR}}$ contains unordered pairs of active rectangles with at least one movable endpoint, so frozen active rectangles serve as obstacles. Write $B_i(X)$ for rectangle i's footprint, $A_i = s_i^x s_i^y$ for its area, and $A_{\text{act}} = \sum_{i\in\mathcal{V}_{\text{act}}} A_i$.

For die Ω and source placement X_{src}, the proposal objective is

$$\begin{aligned}\mathcal{L}_{\text{prop}}(X) = \frac{\lambda_{\text{pen}}}{A_{\text{act}}} &\sum_{(i,j)\in\mathcal{P}_{\text{MGR}}} \phi_{ij}(X) \\ &+ \frac{\lambda_{\text{out}}}{A_{\text{act}}} \sum_{i\in\mathcal{V}_{\text{mov}}} (A_i - |B_i(X)\cap\Omega|) \\ &+ \lambda_{\text{wl}} \left[\frac{H(X)}{\max\{H(X_{\text{src}}), \epsilon_H\}} - (1+\xi)\right]_+^2.\end{aligned} \tag{C.2}$$

Here H denotes HPWL in normalized chip coordinates, with each pin position computed as the center of its owning object plus a fixed offset. We optimize the proposal objective over movable coordinates, using nonnegative weights, a proposal HPWL margin $\xi \geq 0$, and a denominator floor $\epsilon_H > 0$ that prevents division by zero.

Acceptance checks. Candidates are evaluated using the duplicate and outside area ratios over all active footprints:

$$\begin{aligned}D(X) &= \frac{\sum_{i\in\mathcal{V}_{\text{act}}} |B_i(X)\cap\Omega| - \left|\bigcup_{i\in\mathcal{V}_{\text{act}}} (B_i(X)\cap\Omega)\right|}{A_{\text{act}}}, \\ O(X) &= \frac{\sum_{i\in\mathcal{V}_{\text{act}}} (A_i - |B_i(X)\cap\Omega|)}{A_{\text{act}}}.\end{aligned} \tag{C.3}$$

These metrics use the full, unsmoothed geometry independently of the proposal's pair set. From the current placement X and proposal $\widetilde{X}$, MGR tests $X_\alpha = X + \alpha(\widetilde{X} - X)$ for decreasing fractions $\alpha \in \mathcal{A} \subset (0, 1]$. Acceptance requires

$$\begin{aligned} D(X_\alpha) &< D(X) - \tau, \\ O(X_\alpha) &\le \max\{O(X), \epsilon_O\} + \tau, \\ H(X_\alpha) &\le (1 + \eta_k) H(X_{\mathrm{src}}) + \tau, \end{aligned} \tag{C.4}$$

where $\tau \ge 0$ is the numerical comparison tolerance used in the implementation, $\epsilon_O \ge 0$ is the outside area floor, and $\eta_k \ge 0$ is the current HPWL allowance. Each accepted update must reduce duplicate area while satisfying the outside area and HPWL limits, so improving outside area alone is insufficient for acceptance. The HPWL limit remains anchored to the source placement to prevent allowances from compounding across rounds. Algorithm C.1 summarizes the refinement procedure.

Algorithm C.1 MGR residual refinement

Input: Source placement X_{src}, circuit context C, frozen objects $\mathcal{F}$
Parameters: Rounds R, proposal steps T, descending fractions $\mathcal{A}$, HPWL allowances $\{\eta_k\}$
Output: Retained placement X

$X \leftarrow X_{\mathrm{src}}$; allowance tier $k \leftarrow 1$; rejection counter $c \leftarrow 0$
for $r = 1, \ldots, R$ **do**
  $D_{\mathrm{start}} \leftarrow D(X)$
  $\widetilde{X} \leftarrow \textsc{Propose}(X, C, \mathcal{F}, T)$ ▷ Minimize Eq. (C.2)
  for $\alpha \in \mathcal{A}$ from largest to smallest **do**
    $X_\alpha \leftarrow X + \alpha(\widetilde{X} - X)$
    Evaluate the three checks in Eq. (C.4) on the full layout X_α.
    if all three checks pass **then**
      $X \leftarrow X_\alpha$; **break**
    end if
  end for
  Update (k, c) from D_{start} and this round's checks (Appendix C.3).
end for
return X

Starting from X, PROPOSE takes T steps with a fresh Adam optimizer to minimize the objective in Equation (C.2). It projects each movable center coordinate x_i^a onto $[-1 + s_i^a/2, 1 - s_i^a/2]$ after every step and keeps frozen coordinates unchanged. If no candidate passes, the retained placement remains unchanged. Appendix C.3 specifies the numerical settings and allowance update.

C.2 Sparse Constraint Evaluation

We use sweep and prune (Cohen et al., 1995) to enumerate pairs of intersecting axis-aligned bounding boxes (AABBs), reducing pairwise computation on large layouts. The query boxes depend on the constraint operator so that every pair contributing to its value or backward update is retained.

Operator specific boxes. For guidance and legalization, define the signed axial gap δ_{ij}^a, softmax weights p_{ij}^a, and pair penalty ψ_{ij} by

$$\delta_{ij}^a = |x_i^a - x_j^a| - \frac{s_i^a + s_j^a}{2}, \quad p_{ij}^a = \frac{\exp(\beta \delta_{ij}^a)}{\sum_{b \in \{x,y\}} \exp(\beta \delta_{ij}^b)},$$
$$\ell_{ij} = \sum_{a \in \{x,y\}} p_{ij}^a \delta_{ij}^a, \qquad \psi_{ij} = \tfrac{1}{4}[-\ell_{ij}]_+^2,$$

with inverse temperature $\beta > 0$. The standard entropy bound gives $\ell_{ij} \ge \max_a \delta_{ij}^a - \log 2/\beta$ (Pan et al., 2020). If either axial gap is at least $\log 2/\beta$, the pair penalty and its coordinate derivatives are zero. We therefore expand each physical box by $e = \log 2/(2\beta)$ on each side and enumerate intersecting box pairs to retain every contributing pair. For MGR, physical boxes suffice ($e = 0$) because Equation (C.1) vanishes for strictly separated rectangles.

The candidate set is recomputed whenever coordinates or β change and remains fixed during each value/backward evaluation. Each operator uses the same pair eligibility rules across evaluation routes. The smooth term also retains the endpoint backward weights specified in Appendix C.3. Unary boundary terms are evaluated directly. Algorithm C.2 gives the candidate enumeration step.

Algorithm C.2 Sweep and prune candidate enumeration

Input: Object centers and sizes $\{(x_i, s_i)\}_{i=1}^{N}$, per-side expansion $e \geq 0$
Output: Candidate set $\mathcal{Q}$ of unordered intersecting box pairs

$I_i^a \leftarrow [x_i^a - s_i^a/2 - e,\ x_i^a + s_i^a/2 + e]$ for all i and $a \in \{x, y\}$
for $a \in \{x, y\}$ **do**
    Sort intervals I_i^a by lower endpoint and count intersecting pairs M_a.
end for
$a^* \leftarrow \arg\min_{a \in \{x,y\}} M_a$; let b be the other axis.
$\mathcal{Q} \leftarrow \varnothing$
for each intersecting interval pair $i < j$ enumerated by the sweep on axis a^* **do**
    if $I_i^b \cap I_j^b \neq \varnothing$ **then**
        $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(i, j)\}$
    end if
end for
return $\mathcal{Q}$

The intervals are closed, so edge and corner contacts are retained. Every intersecting query box pair passes both axis tests and is returned once. The implementation sorts the returned pairs lexicographically. Let N be the number of model objects, M the number of intersecting interval pairs on the selected axis, and $K \leq M$ the number of returned pairs. For this implementation, candidate construction requires time T_{sweep} and workspace W_{sweep}, where

$$T_{\text{sweep}} = O(N \log N + M + K \log(1 + K)), \quad W_{\text{sweep}} = O(N + M), \tag{C.5}$$

with $O(K)$ additional work to evaluate the pairs. Candidate counts can remain quadratic in crowded layouts. The routes agree pointwise in exact arithmetic, although floating-point reduction order can affect subsequent optimization trajectories.

C.3 Implementation Settings

MGR. Table 3 summarizes the refinement settings. Adam is reset each round, with $\xi = 0$ and $\epsilon_H = \tau = 10^{-12}$. The adaptive policy initializes $k = 1$ and $c = 0$. The counter c increments only when the round starts with a duplicate area ratio above 0.02 and every candidate passes both geometric checks but fails the HPWL check. Otherwise, c resets to zero. After two consecutive qualifying rounds, the policy advances one tier and resets c, provided a higher tier and another round remain. Runs with a fixed allowance keep $k = 1$, including the matched source comparison in Appendix D.3.1, which uses $\eta = 0.005$.

When two centers coincide, the proposal backward rule selects the axis requiring the shorter translation to separate the rectangles. If the required distances are equal, it selects x when $i + j$ is even and y otherwise. It assigns a negative descent direction to the endpoint with the smaller ID and a positive one to the other endpoint, while preserving the forward penalty value.

Table 3: MGR configuration.

Parameter	Value
Rounds R / proposal steps T	$20/20$
Optimizer / learning rate	Adam / 10^{-3}
Gradient clip	0.1
Weights $(\lambda_{\text{pen}}, \lambda_{\text{out}}, \lambda_{\text{wl}})$	$(1, 1, 50)$
Step fractions $\mathcal{A}$	$(1, 1/2, \ldots, 1/128)$
HPWL allowances η_k	$(0.005, 0.01, 0.02)$
Outside floor ϵ_O	10^{-6}

Constraint evaluation. The smooth penalty between model objects excludes self-pairs and any pair with an endpoint masked out of constraint evaluation. The movement mask determines which coordinates may be updated. For each eligible pair, the backward rule scales the ordinary derivatives at endpoints i and j by $w_{ij} = m_j/(m_i + m_j)$ and w_{ji}, respectively, where $m_i = \sqrt{s_i^x s_i^y}$. All evaluation routes use the same masks and backward weights. Object sizes, masks, pin data, and HPWL indexing are cached. During guidance, β increases linearly from 10 to 20 as diffusion time decreases from $t = 1$ to $t = 0.1$, then remains at 20. During legalization, β increases linearly from 5 to 50 over the first 10% of iterations and remains at 50 thereafter. For interactions between model objects, the dense route uses dense computation for guidance, tiled computation for legalization, and dense or chunked computation for MGR proposals.

Table 4: Size-adaptive pairwise evaluation routes for guidance and legalization. Model–model routes also apply to MGR proposals. N_f is the number of separately represented fixed objects.

Interaction	Dense route	Sparse route
Model–model	$N < 2{,}048$ All-pairs evaluation	$N \geq 2{,}048$ GPU sweep and prune
Model–fixed	$NN_f \leq 10^6$ All-pairs evaluation	$NN_f > 10^6$ GPU uniform grid or CPU STRtree

D Supporting Experimental Results

D.1 Training and Evaluation Protocol

D.1.1 Spatial Base Pretraining

Spatial Base is pretrained on 40,000 synthetic graph–layout pairs, with 2,000 additional pairs reserved for validation. Following ChipDiffusion's data-generation procedure, we first sample legal layouts at random within a normalized canvas without explicitly imposing a target spatial profile, then sample connectivity based on pin distances. We use a cosine diffusion noise schedule and minimize the Gaussian noise-prediction MSE with Adam for 3,000,000 iterations at a constant learning rate of 3×10^{-4}. The resulting checkpoint is used as Base in all subsequent experiments, without ChipDiffusion's additional full model fine-tuning on larger synthetic circuits.

D.1.2 SPA Training

For the primary synthetic targets, we train SPA separately for each circuit group within each placement setting: IBM G1 (ibm01–04, ibm06), IBM G2 (ibm07–12), IBM G3 (ibm13–18), Adaptec (adaptec1–4), and Bigblue (bigblue1–4). We exclude ibm05 because it contains no movable macros. For each group, Algorithm B.1 in Appendix B.3 generates a fixed corpus of 20 training pairs based on statistics from the circuits in that group, without using reference or optimized benchmark placements. For mixed-size placement, synthetic targets include coordinates for both movable macros and standard-cell cluster proxies (treated as macros during synthesis). The primary targets favor the die periphery for the largest 25% of movable macros by area and mildly favor the interior for the rest. Unless specified otherwise, SPA uses Adam with learning rate 5×10^{-5} for 2,000 updates. Each update uses one context–placement pair and eight independently sampled timestep–noise pairs. The learned SPA parameters are reused across circuits within the corresponding target setting and remain fixed during inference.

D.1.3 Evaluation Protocol and Metrics

Benchmarks. The macro-only setting follows the I/O-freed convention and ChipDiffusion's macro-selection rule. The mixed-size setting uses the ISPD subset of MMS and selects explicit macros as in MacroDiff+. In our mixed-size representation, we use hMETIS (Karypis et al., 1997) to partition movable standard cells into 128 clusters, each represented by a proxy whose area equals the total area of its constituent cells, while macros remain individually represented. In both settings, the 2,480 originally movable double row height objects in bigblue3 are excluded from explicit macro

prediction. In the mixed-size representation, they are included in standard-cell clusters. Following DREAMPlace 4.1's dedicated macro-mask height criterion, bigblue3 has 1,298 explicit macros in macro-only evaluation and 69 in mixed-size evaluation. Complete physical design evaluation uses the eight ChiPBench circuits with the largest macro counts. All methods use the same ORFS configuration, keeping supplied macros fixed during standard-cell placement and legalization, clock tree synthesis, and global and detailed routing. Further details are provided in the ChiPBench and OpenROAD papers (Wang et al., 2025; Kahng & Spyrou, 2021).

Evaluation. Macro HPWL uses unit net weights, and downstream wHPWL uses DREAMPlace's full netlist weights. We use RUDY (Spindler & Johannes, 2007) as a proxy for routing demand and report its average value over the highest-demand 10% of bins in a 512×512 grid. The area-based geometry score is $100(1-D-O)$, using the exact duplicate and outside area ratios in Equation (C.3). Higher geometry scores and lower wirelength and RUDY values are preferred. For ChiPBench, we report three groups of physical-design metrics. Routed wirelength characterizes interconnect length, and congestion (Cong.) is measured by global routing resource usage. Worst negative slack (WNS), total negative slack (TNS), and the number of violating paths (NVP) characterize the severity and extent of timing violations. We also report power consumption and the total area of standard cells. Higher WNS and TNS and lower values of the other metrics are preferred.

Reporting conventions. *Failed* denotes unsuccessful placement, and *OOM* denotes an out of memory error. To compute averages marked with †, we replace missing values with the best available result for the same circuit and metric across all methods evaluated on the same benchmark. This optimistic imputation is used only for averaging.

Mixed-size handoff. For IBM/ISPD methods supplying a macro handoff, DREAMPlace 4.1 optimizes individual standard-cell positions while keeping macros fixed. ChipDiffusion and our pipeline initialize standard cells by expanding generated cluster layouts into individual positions. All other evaluated methods use random standard-cell initialization.

Hardware. Placement experiments use Intel Xeon Gold 6459C CPUs and modified RTX 4090 GPUs with 48 GB of memory, using CUDA 12.1 and Python 3.10.4. Timing and memory comparisons use matched hardware and software conditions within each experimental cohort.

D.2 Effectiveness of Spatial Prior Adaptation

D.2.1 Target Denoising Risk Reduction

For each of the five adaptation groups, we compare Base with SPA checkpoints saved at 1K and 2K updates from three training runs. All models are evaluated on the same set of 20 target context–placement pairs, none of which were used for SPA training. For each pair, we construct forward-noised inputs at eight equally weighted timestep-bin midpoints, using eight Gaussian noise draws per midpoint. Base and SPA receive identical noisy coordinates, contexts, and timesteps, and their masked noise-prediction losses are computed using Equation (3.2). Configurations and checkpoints were fixed before evaluation, and no model parameters are updated during the diagnostic. We define the gain as the average of the paired loss differences (Base $-$ SPA) over timestep–noise pairs, layouts, and training runs. Pointwise 95% confidence intervals are obtained from 5,000 whole-layout bootstrap resamples. Each resampled layout retains all of its timestep–noise evaluations and training-run results, with the trained checkpoints held fixed. All ten gain intervals lie above zero (Table 5).

Table 5: Target denoising risk reduction. Gains and pointwise 95% CIs are in 10^{-2} units. Relative reductions are percentages of Base risk.

Adaptation group	1K gain [95% CI]	2K gain [95% CI]	Risk reduction (%), 1K → 2K
IBM G1	3.81 [2.74, 5.01]	4.61 [3.30, 6.25]	11.98 → 14.47
IBM G2	6.02 [4.90, 7.15]	6.69 [5.47, 7.94]	17.45 → 19.40
IBM G3	5.39 [4.94, 5.80]	5.95 [5.46, 6.43]	16.66 → 18.40
ISPD Adaptec	6.17 [4.59, 7.74]	6.85 [5.08, 8.56]	17.22 → 19.12
ISPD Bigblue	5.53 [3.95, 7.10]	6.13 [4.43, 7.90]	15.63 → 17.31

D.2.2 Spatial Prior Acquisition under Matched Contexts

Compact, Relaxed, and Dispersed share the same 20 IBM G1 training contexts, balanced across the five circuits. These shared contexts are taken from Dispersed and include connectivity, pin ownership, and pin offsets. After verifying that object order and dimensions match, we pair each context with the corresponding unchanged Compact coordinates. Algorithm B.1 in Appendix B.3 uses $\tau_{\mathrm{C}} = (0, 2, 2)$ and $\tau_{\mathrm{D}} = (0.25, 2, 0.5)$. Relaxed resamples coordinates similarly to Dispersed, replacing the largest-quartile boundary preference with the annular weight $w(r) = \exp[-\frac{1}{2}((r - 0.72)/0.16)^2]$, where $r = \sqrt{x^2 + y^2}$ for $(x, y) \in [-1, 1]^2$.

Each profile has five independent SPA training runs of 2K updates. We evaluate each of the five SPA checkpoints on all five circuits, yielding 25 evaluations per profile. In each paired evaluation, Base and SPA use the same Gaussian sampling noise. Sampling uses 1,000 reverse steps with guidance and legalization disabled. We then apply MGR under shared settings and evaluate spatial responses before and after refinement to assess whether the learned spatial preferences persist.

For the $K = \lceil 0.25N \rceil$ movable macros with the largest areas (ties broken by node ID), let S be their index set, with centers $p_i = (x_i, y_i)$ in $[-1, 1]^2$. We quantify their spatial spread using

$$B = \frac{(\max_{i \in S} x_i - \min_{i \in S} x_i)(\max_{i \in S} y_i - \min_{i \in S} y_i)}{4},$$

$$P = \frac{1}{\binom{K}{2}} \sum_{i<j;\, i,j \in S} \frac{\|p_i - p_j\|_2}{\sqrt{8}}.$$

Thus, B is the normalized area of the bounding box defined by macro centers rather than macro footprints, while P is the normalized average pairwise distance between the centers. Target averages (B, P) are $(0.70, 0.30)$ for Compact, $(0.84, 0.35)$ for Relaxed, and $(0.94, 0.42)$ for Dispersed. Base averages are $(0.88, 0.35)$ before refinement and $(0.87, 0.35)$ afterward. For Relaxed, we additionally measure annular affinity A and normalized angular entropy H:

$$A = \frac{1}{K} \sum_{i \in S} w(r_i), \quad H = -\frac{\sum_{b=1}^{8} p_b \log p_b}{\log 8}.$$

Here, $r_i = \sqrt{x_i^2 + y_i^2}$ and $\theta_i = \mathrm{atan2}(y_i, x_i)$. The full circle is divided into eight equal 45° sectors, and p_b is the fraction of selected centers in sector b. We use the convention $0 \log 0 = 0$. Higher A indicates closer alignment with the prescribed radius of 0.72, while higher H indicates more uniform angular coverage.

We report the average of the paired differences $\Delta = \mathrm{SPA} - \mathrm{Base}$, with pointwise 95% percentile confidence intervals based on 5,000 crossed bootstrap replicates. Circuits and training runs are resampled independently, with pairing preserved across methods and stages. Both before and after MGR, Compact reduces B and P, Dispersed increases B and P, and Relaxed increases A and H. All 12 intervals for these contrasts exclude zero (Table 6).

Table 6: Spatial responses before and after MGR. Average paired differences (SPA − Base) and pointwise 95% CIs are reported in 10^{-2} units.

Metric	Profile	Raw: Δ [95% CI]	MGR: Δ [95% CI]
B	Compact	-9.38 $[-13.18, -6.21]$	-8.39 $[-11.51, -3.87]$
B	Relaxed	-2.05 $[-5.39, +1.35]$	-0.49 $[-3.41, +3.75]$
B	Dispersed	$+3.78$ $[+0.14, +6.85]$	$+4.50$ $[+2.62, +7.51]$
P	Compact	-2.45 $[-3.84, -0.86]$	-2.10 $[-3.46, -0.57]$
P	Relaxed	-0.88 $[-2.17, +0.55]$	-0.68 $[-1.95, +0.78]$
P	Dispersed	$+1.87$ $[+1.55, +3.10]$	$+1.77$ $[+1.48, +2.90]$
A	Relaxed	$+3.14$ $[+1.05, +5.55]$	$+3.21$ $[+0.36, +5.78]$
H	Relaxed	$+2.87$ $[+0.93, +5.24]$	$+3.18$ $[+1.13, +5.75]$

D.2.3 TRAINABLE COMPONENTS

We ablate residual adapters and fusion LoRA individually and jointly, retaining output head fine-tuning in all adapted configurations. For each group, we perform three independent runs, each regenerating a shared adaptation corpus and retraining all adapted configurations. Within each run, configurations differ only in trainable components. Table 7 reports macro-only results without MGR.

Table 7: SPA component ablations. We report average HPWL (10^5), RUDY (10^2), and geometry scores (%). Best unrounded results are in bold.

	Adaptec			Bigblue			IBM G1		
Configuration	**HPWL**	**RUDY**	**Geom.**	**HPWL**	**RUDY**	**Geom.**	**HPWL**	**RUDY**	**Geom.**
w/o SPA	37.58	4.87	98.84	57.89	6.29	99.03	1.49	1.36	99.95
Head only	37.05	4.86	97.95	56.33	5.91	99.59	1.49	1.30	99.93
w/o Fusion LoRA	36.56	4.80	98.01	51.72	5.91	98.75	1.48	1.29	99.95
w/o Residual Adapters	36.44	4.73	96.84	52.71	5.84	99.45	1.45	1.29	99.95
Full SPA	**36.36**	**4.73**	**98.84**	**51.51**	**5.76**	**99.59**	**1.43**	**1.27**	**100.00**

D.2.4 COMPARISON WITH FULL FINE-TUNING

SPA and Full-FT start from the same Spatial Base checkpoint and use matched target data and a learning rate of 5×10^{-5}. Evaluation covers 13 macro-only circuits from IBM G1, Adaptec, and Bigblue, with MGR and downstream processing disabled. We average equally over circuits and three independent training runs per method and group. We compute 95% percentile bootstrap intervals by resampling runs within groups, keeping each run's circuits together. Figure 6(a) reports average HPWL and RUDY for Full-FT at 1K, 5K, 10K, 15K, and 20K updates, alongside the fixed SPA reference trained for 1K updates.

Under the evaluated training configuration, Full-FT performance varies non-monotonically with the training budget. Across all reported checkpoints from 1K to 20K updates, its average HPWL and RUDY remain higher than those of SPA trained for 1K updates. At 20K updates, Full-FT has 4.88% higher HPWL and 2.50% higher RUDY than SPA.

In terms of adaptation cost, SPA updates 18,562 parameters, 3.13% of the 593,906 updated by Full-FT. Average training time for 1K updates is 47.5 s for SPA and 66.5 s for Full-FT, a 28.6% reduction. Training Full-FT for 20K updates takes 1,313.8 s, 27.7 times as long as training SPA for 1K updates. Peak allocated GPU memory is 1,819.4 MiB for SPA versus 3,209.9 MiB for Full-FT.

For reference, we locally reproduced the full model fine-tuning of ChipDiffusion Large for transfer to larger synthetic circuits (Lee et al., 2025). This stage used 5,000 training samples and 250,000 optimizer updates to fine-tune all 6,285,058 parameters, taking approximately 13.4 hours on the same hardware. For placement evaluation, however, we use the authors' released checkpoint.

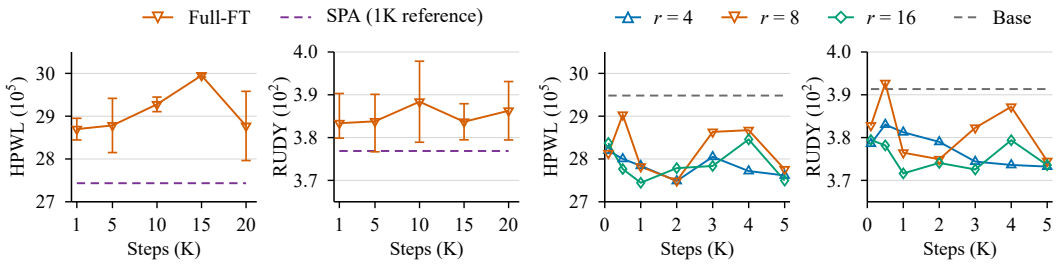
D.2.5 RANK AND TRAINING BUDGET SENSITIVITY

In a separate sensitivity experiment, we vary the fusion rank $r \in \{4, 8, 16\}$ with $\alpha/r = 2$, using three matched training runs per setting. Figure 6(b) reports HPWL and RUDY before MGR over 0.1K–5K updates. Results are averaged over the three runs for each circuit and then equally over the 13 circuits. Performance varies non-monotonically with training budget, and the best-performing rank changes across budgets and metrics.

D.3 SCALABLE CONSTRAINT-AWARE INFERENCE

D.3.1 MGR RESIDUAL REFINEMENT

We test whether MGR corrects residual geometric violations more effectively than extending legalization. The starting layouts come from eight ISPD circuits and five paired sampling streams per model. Starting from the same layouts after 20,000 legalization updates, continued legalization



(a) SPA versus Full-FT. (b) Rank and training budget.

Figure 6: Average HPWL and RUDY. Lower is better. (a) Full-FT with 95% confidence intervals versus SPA (1K updates). (b) Sensitivity to fusion LoRA rank ($r = 4, 8, 16$) across training budgets.

takes 400 more updates without resetting its optimizer or legalization state. MGR takes 20 rounds of 20 proposal updates. The HPWL allowance is fixed at $\eta = 0.005$, with other settings following Appendix C.3.

Relative to the post-legalization source layouts, MGR yields lower/unchanged/higher D values in 40/0/0 cases for MacroSPA (w/o SPA) and 38/2/0 cases for MacroSPA (synthetic). The corresponding counts for O are 36/3/1 and 36/4/0.

Table 8: Average residual repair results. Continued legalization and MGR each use 400 additional updates from matched post-legalization layouts. D and O denote duplicate and outside area ratios.

Model	Stage	D (10^{-3})	O (10^{-6})	HPWL (10^6)
MacroSPA (w/o SPA)	Source	8.69	351.87	4.68
	Continue	8.68	345.66	4.68
	MGR	2.21	0.40	4.61
MacroSPA (synthetic)	Source	10.69	691.45	4.43
	Continue	10.67	687.27	4.42
	MGR	4.54	364.60	4.26

D.3.2 Stage-wise Sources of Inference Speedup

Table 9 compares runtime at each inference stage under matched hardware conditions. Each value is averaged over three runs for the corresponding circuit and configuration. The MacroSPA routes differ only in constraint evaluation. The dense route evaluates all eligible pairs of model objects, whereas the sparse route evaluates candidate pairs constructed by GPU sweep and prune (Appendices C.2–C.3). ChipDiffusion provides an external pipeline reference. Circuit-level placement-quality comparisons under the main evaluation protocol are reported in Appendix D.4.1.

On both circuits, legalization contributes most of the savings over the dense route, leaving guided sampling as the main cost. Sampling and legalization contribute similar savings over ChipDiffusion. MGR accounts for only about 2% of total inference time with the sparse route.

Table 9: Average inference time (s) by stage on the two larger circuits. Object counts are shown in parentheses. ChipDiffusion has no MGR stage.

Circuit	**Method / route**	**Guided sampling**	**Legalization**	**MGR**
bigblue4 (8,170)	ChipDiffusion	808.73	789.85	–
	MacroSPA (Dense)	394.92	749.27	20.15
	MacroSPA (Sparse)	136.82	60.52	4.32
bigblue2 (23,084)	ChipDiffusion	5945.78	5789.94	–
	MacroSPA (Dense)	2999.31	5819.28	66.79
	MacroSPA (Sparse)	171.10	93.20	6.41

D.4 Physical Design Utility

D.4.1 Circuit-Level Placement Results

Tables 10–11 and Figure 7 provide circuit-level placement and geometry results for Section 4.4 and Appendix E, under the protocol in Appendix D.1.3. Each circuit-level result is averaged over three independent runs.

Table 10: IBM placement results on 17 circuits, excluding ibm05. We report macro-only HPWL (10^5), RUDY (10^2), and downstream wHPWL (10^6). MacroSPA (DP-teacher) is evaluated for wHPWL only.

Method	Metric	ibm01	ibm02	ibm03	ibm04	ibm06	ibm07	ibm08	ibm09	ibm10
DREAMPlace	HPWL	1.18	2.43	0.98	2.46	0.52	0.80	10.85	0.66	6.83
	RUDY	1.46	1.65	1.03	2.46	0.47	0.86	9.35	0.61	2.80
	wHPWL	2.40	4.77	6.65	7.52	5.71	9.63	10.88	11.92	30.45
WireMask-EA	HPWL	2.61	4.40	2.08	3.38	0.47	1.84	11.92	1.42	14.70
	RUDY	2.36	2.22	1.51	2.70	0.65	1.34	7.96	1.02	3.97
	wHPWL	3.10	6.50	10.31	10.70	8.65	13.01	16.97	16.77	43.05
ChiPFormer	HPWL	2.53	4.75	1.13	3.36	0.42	1.12	11.27	0.71	9.88
	RUDY	2.42	2.40	1.14	2.77	0.57	1.04	7.61	0.67	3.32
	wHPWL	3.17	6.90	9.38	9.76	8.78	13.56	16.12	15.88	42.85
ChipDiffusion	HPWL	1.19	2.67	1.01	2.43	0.56	0.73	10.69	0.47	5.76
	RUDY	1.54	1.72	1.10	2.48	0.76	0.75	8.14	0.41	2.56
	wHPWL	2.65	5.54	7.70	8.52	8.17	11.25	13.87	14.80	36.04
MacroSPA (w/o SPA)	HPWL	1.19	2.46	0.88	2.35	0.53	0.77	11.32	0.49	5.50
	RUDY	1.41	1.65	0.95	2.27	0.67	0.78	8.48	0.42	2.19
	wHPWL	2.60	6.80	8.64	8.52	6.65	10.35	12.99	13.53	36.64
MacroSPA (synthetic)	HPWL	1.17	2.43	0.83	2.28	0.38	0.72	10.40	0.47	5.40
	RUDY	1.39	1.65	0.87	2.18	0.30	0.73	7.75	0.41	2.19
	wHPWL	2.57	5.18	7.48	8.43	6.61	10.70	13.26	13.97	33.39
MacroSPA (DP-teacher)	wHPWL	2.39	5.26	7.23	8.38	6.22	9.90	13.29	13.00	33.23

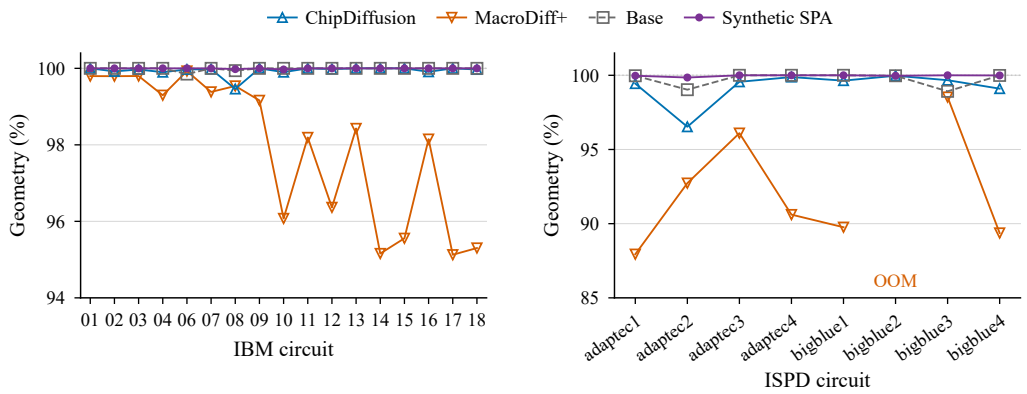
Method	Metric	ibm11	ibm12	ibm13	ibm14	ibm15	ibm16	ibm17	ibm18	Average
DREAMPlace	HPWL	0.87	3.25	1.13	2.51	1.19	7.44	2.89	1.48	2.79
	RUDY	0.69	1.80	0.69	1.85	0.67	2.92	1.59	1.02	1.88
	wHPWL	17.06	31.21	23.05	33.03	43.59	50.83	60.70	39.64	22.88
WireMask-EA	HPWL	2.65	*Failed*	3.81	6.93	*Failed*	13.64	10.20	3.94	5.15†
	RUDY	1.47	*Failed*	1.66	3.18	*Failed*	4.75	3.62	2.07	2.50†
	wHPWL	23.68	*Failed*	30.06	48.15	*Failed*	65.65	74.75	41.50	28.69†
ChiPFormer	HPWL	1.28	5.74	1.96	3.73	1.44	11.71	7.45	3.25	4.22
	RUDY	1.03	2.69	1.23	2.61	0.90	4.31	3.42	1.97	2.36
	wHPWL	23.86	42.63	29.62	46.96	58.03	68.26	71.91	41.50	29.95
ChipDiffusion	HPWL	0.81	2.81	1.05	2.39	0.98	5.97	3.15	1.55	2.60
	RUDY	0.65	1.54	0.79	1.77	0.57	2.47	1.77	1.06	1.77
	wHPWL	19.59	40.62	26.20	36.89	48.77	60.92	68.10	40.38	26.47
MacroSPA (w/o SPA)	HPWL	0.77	2.78	0.95	2.43	0.92	5.94	3.10	1.57	2.58
	RUDY	0.56	1.60	0.67	1.70	0.53	2.51	1.49	1.07	1.70
	wHPWL	19.97	39.46	26.57	36.67	50.22	60.08	70.95	39.81	26.50
MacroSPA (synthetic)	HPWL	0.73	2.75	0.93	2.41	0.87	5.88	3.11	1.52	2.49
	RUDY	0.57	1.60	0.65	1.67	0.44	2.46	1.60	1.02	1.62
	wHPWL	19.38	38.89	24.65	36.27	47.76	58.10	67.42	39.16	25.48
MacroSPA (DP-teacher)	wHPWL	19.15	38.09	24.19	35.32	47.44	55.14	63.42	38.57	24.72

† Averages include imputed values. See Appendix D.1.3 for reporting conventions.

Table 11: ISPD placement results. We report macro-only HPWL (10^6), RUDY (10^2), and downstream wHPWL (10^7).

Method	Metric	Adaptec				Bigblue				Average
		1	2	3	4	1	2	3	4	
DREAMPlace	HPWL	1.02	5.05	6.48	4.29	0.39	3.79	3.61	13.66	4.79
	RUDY	3.18	9.46	7.56	6.25	1.30	7.18	5.02	15.97	6.99
	wHPWL	6.54	7.83	15.51	14.18	8.54	12.53	27.34	64.41	19.61
WireMask-EA	HPWL	0.93	7.08	6.67	6.61	0.33	17.86	6.36	80.14	15.75
	RUDY	1.47	14.88	7.34	8.03	0.29	19.58	10.99	54.70	14.66
	wHPWL	8.42	9.82	18.02	18.14	9.87	14.51	31.85	84.09	24.34
ChiPFormer	HPWL	1.16	6.56	8.34	6.96	0.44	12.19	8.63	65.37	13.71
	RUDY	2.68	11.53	7.26	7.96	0.50	18.23	9.80	64.11	15.26
	wHPWL	8.70	12.45	20.59	19.51	10.34	17.10	45.13	108.28	30.26
ChipDiffusion	HPWL	0.92	3.29	5.38	5.42	0.27	3.98	3.50	13.61	4.55
	RUDY	1.60	6.64	6.23	6.03	0.48	5.99	4.12	14.85	5.74
	wHPWL	7.14	9.04	17.60	15.44	9.84	13.19	41.71	80.84	24.35
FlowPlace	HPWL	1.94	12.68	13.79	13.86	0.74	17.30	10.14	19.67	11.26
	RUDY	3.16	14.08	10.15	10.98	1.13	31.00	20.96	56.78	18.53
	wHPWL	8.88	12.52	22.16	26.62	9.72	16.03	40.52	107.06	30.44
MacroDiff+	HPWL	2.31	11.19	17.06	20.77	0.92	*OOM*	10.46	16.18	10.34†
	RUDY	3.71	14.31	11.48	10.84	1.41	*OOM*	12.00	10.98	8.74†
	wHPWL	7.73	8.90	16.76	16.12	9.95	14.57	39.23	75.95	23.65
MacroSPA (w/o SPA)	HPWL	0.94	3.14	5.51	5.34	0.27	4.01	4.30	13.85	4.67
	RUDY	1.44	6.15	6.12	5.89	0.28	5.22	5.10	14.74	5.62
	wHPWL	7.63	9.11	18.04	15.65	10.10	13.26	38.77	77.37	23.74
MacroSPA (synthetic)	HPWL	0.91	2.93	5.27	5.32	0.27	3.88	3.26	12.04	4.23
	RUDY	1.45	5.74	6.10	5.97	0.29	5.57	3.86	13.37	5.29
	wHPWL	7.41	8.83	17.72	15.40	9.90	13.16	37.47	75.79	23.21

† Averages include imputed values. See Appendix D.1.3 for reporting conventions.



Average (%)	ChipDiffusion	FlowPlace	MacroDiff+	MacroSPA (w/o SPA)	MacroSPA (synthetic)
IBM	99.94	100.00	97.94	99.99	100.00
ISPD	99.22	100.00	93.13†	99.74	99.97

† Averages include imputed values. See Appendix D.1.3 for reporting conventions.

Figure 7: Macro placement geometry on IBM (left) and ISPD (right). Higher is better.

D.4.2 SYNTHETIC TARGET PROFILES

To compare synthetic target profiles, we use the same circuit sampling, macro-size sequences, and degree-sequence conditions for three SPA variants. Peripheral is the primary profile, while Clustered favors the interior. We use Algorithm B.1 with $\tau_{\mathrm{P}} = (0.25, 2, 0.5)$ for Peripheral and $\tau_{\mathrm{C}} = (0, 2, 2)$ for Clustered. Random sequentially samples each macro's origin uniformly from currently legal positions. For all three profiles, we follow the training protocol in Appendix D.1.2 and use the same inference and downstream settings.

Table 12: Macro-only and downstream placement under different synthetic target profiles. Entries are percentage changes relative to Peripheral, averaged across circuits within each benchmark. Negative values indicate improvement.

Benchmark	Target profile	ΔHPWL (%) ↓	ΔRUDY (%) ↓	ΔwHPWL (%) ↓
IBM	Random	−0.40	+0.13	+1.78
IBM	Clustered	−0.16	+0.27	+1.42
ISPD	Random	−2.42	−1.80	+4.14
ISPD	Clustered	−2.36	+0.36	+3.69

On both IBM and ISPD, Random and Clustered lower macro-only HPWL but increase downstream wHPWL relative to Peripheral. This reversal shows that improvements in macro-only wirelength do not necessarily translate into better downstream mixed-size placement, motivating the evaluation of adapted spatial priors against downstream physical design objectives.

D.4.3 OPTIMIZER-GENERATED TARGETS AND TRANSFER

We evaluate SPA with optimizer-generated targets and assess transfer to circuits excluded from adaptation. DREAMPlace generates legalized IBM placements, which we project into the model's clustered representation using a fixed hMETIS map. Macros remain individually represented, and fixed object positions are preserved. For each cluster of movable standard cells, we compute its target position as the area-weighted average of its member cells' positions in the DREAMPlace placement. This projection yields target coordinates for both movable macros and standard-cell clusters.

We train a shared set of SPA parameters on 50 samples from ibm01–09, following the training protocol in Appendix D.1.2. After 2,000 updates, we evaluate the same adapted model on all 17 IBM circuits, including ibm10–18, which are excluded from adapter training. The synthetic SPA results reported in Appendix D.4.1 serve as the reference.

Table 13: Downstream placement with optimizer-generated targets. We report average wHPWL (10^6) across the circuits in each row. The last column reports DP-teacher's wHPWL reductions (%) over w/o SPA / synthetic SPA.

Circuits	w/o SPA	Synthetic SPA	DP-teacher SPA	Reduction (%)
ibm01–09 (target)	8.76	8.53	8.21	6.29% / 3.71%
ibm10–18 (excluded)	42.26	40.56	39.39	6.79% / 2.87%

Relative to synthetic SPA, DP-teacher further narrows the mixed-size wHPWL gap to native DREAMPlace while retaining MacroSPA's clustered representation. It reduces wHPWL relative to MacroSPA (w/o SPA) on 16 of 17 circuits and relative to synthetic SPA on 15 of 17. These gains indicate that SPA can acquire useful and transferable placement behavior directly from optimizer-generated samples.

In the evaluated flow, DREAMPlace jointly optimizes movable macros and individual standard cells under wirelength and density objectives before legalizing and fixing the macros (Chen et al., 2023). In contrast, MacroSPA determines macro positions using standard-cell cluster proxies and keeps the macros fixed during downstream standard-cell placement. These differences in representation and optimization may partly explain the remaining wHPWL gap.

D.4.4 Downstream Physical Design

Tables 14–15 report complete ChiPBench results for all evaluated methods under the protocol in Appendix D.1.3. For each ChiPBench circuit, external baselines are evaluated once, whereas results for each MacroSPA variant are averaged over three evaluation runs. For synthetic SPA on ChiPBench, we train one shared set of SPA parameters for each of two circuit groups: {ariane133, ariane136, dft68, mor1kx} and {ariane81, bp68, vga_lcd, ethernet}. Each circuit contributes five synthetic training layouts, yielding 20 samples per group and 40 in total. DP-teacher SPA uses 30 DREAMPlace training layouts, five from each of six circuits excluding ariane81 and bp68. Projection follows the procedure in Appendix D.4.3, with supervision on movable macros and standard-cell clusters.

We assess physical design quality using routing, timing, power, and area metrics. Wirelength is measured after detailed routing. Congestion is measured using the Usage value in the Total row of the final post-CTS global-routing report. Timing, power, and area are evaluated after post-route RC extraction. Boldface and underlining indicate the best and second best unrounded values per circuit and metric, including ties.

Table 14: Routing quality and worst negative slack on eight ChiPBench circuits.

Wirelength (10^5 μ**m**) ↓

Method	ariane81	bp68	vga_lcd	ethernet	ariane133	ariane136	dft68	mor1kx
DREAMPlace	**59.91**	161.62	**25.12**	11.22	**69.50**	**69.42**	12.65	**24.38**
WireMask-EA	71.63	207.95	28.83	11.02	75.63	79.26	13.95	24.42
ChiPFormer	69.57	200.53	25.32	10.94	74.87	78.58	15.06	24.40
ChipDiffusion	70.87	146.15	32.24	10.94	75.74	79.02	14.59	24.98
MacroDiff+	69.06	**135.89**	25.69	11.00	77.47	74.95	16.58	25.96
FlowPlace	72.56	226.70	27.03	**10.85**	77.32	79.04	17.27	25.26
MacroSPA (w/o SPA)	67.27	184.95	27.11	11.31	75.66	77.37	**12.20**	24.65
MacroSPA (synthetic)	64.61	154.26	26.35	10.98	72.65	79.43	13.29	25.20
MacroSPA (DP-teacher)	62.91	180.19	27.73	11.00	73.59	78.17	14.42	24.87

Cong. (%) ↓

Method	ariane81	bp68	vga_lcd	ethernet	ariane133	ariane136	dft68	mor1kx
DREAMPlace	28.68	51.46	**12.23**	29.47	**23.30**	**23.47**	9.46	**38.29**
WireMask-EA	32.20	64.58	12.69	28.62	26.10	34.40	9.96	38.46
ChiPFormer	31.29	62.14	12.44	28.66	25.82	26.53	10.77	38.44
ChipDiffusion	31.86	45.61	15.88	28.65	25.50	26.71	10.43	39.35
MacroDiff+	31.03	**43.14**	12.62	28.84	26.08	25.30	11.86	41.22
FlowPlace	31.92	65.02	13.23	**28.39**	26.02	26.68	12.34	39.81
MacroSPA (w/o SPA)	30.23	57.25	12.93	29.64	25.45	26.11	**8.73**	38.84
MacroSPA (synthetic)	29.46	47.87	13.32	28.79	24.43	26.80	9.50	39.71
MacroSPA (DP-teacher)	**28.32**	55.93	13.62	28.83	24.77	26.40	10.31	39.24

WNS (ns) ↑

Method	ariane81	bp68	vga_lcd	ethernet	ariane133	ariane136	dft68	mor1kx
DREAMPlace	-0.33	-3.03	-1.64	**-0.10**	-0.20	-0.22	-0.32	-1.31
WireMask-EA	-0.35	-4.39	-5.06	-0.23	-0.30	-0.31	-0.42	-1.47
ChiPFormer	-0.71	-2.93	-3.54	-0.23	-0.20	-0.52	-0.33	-1.29
ChipDiffusion	-0.80	-3.03	-6.64	-0.13	-0.15	-0.34	-0.33	-1.37
MacroDiff+	-0.35	**-2.35**	**-1.04**	-0.31	**-0.07**	-0.25	-0.34	-1.32
FlowPlace	-0.15	-2.51	-3.21	-0.18	-0.13	-0.24	-0.30	-1.42
MacroSPA (w/o SPA)	-0.32	-4.26	-1.71	-0.23	-0.16	-0.26	-0.31	**-1.28**
MacroSPA (synthetic)	-0.17	-2.38	-1.53	-0.18	-0.13	**-0.19**	-0.29	-1.38
MacroSPA (DP-teacher)	**-0.15**	-3.06	-1.39	-0.14	-0.10	-0.23	**-0.29**	-1.30

Table 15: Post-route timing, power, and standard-cell area on the same ChiPBench circuits.

TNS (ns) ↑

Method	ariane81	bp68	vga_lcd	ethernet	ariane133	ariane136	dft68	mor1kx
DREAMPlace	-64.25	-883.04	-6766.29	-4.24	-113.15	**-87.98**	-63.97	-11109.40
WireMask-EA	-425.40	**-515.67**	-2298.50	-5.23	-273.09	-335.95	-71.34	-11505.50
ChiPFormer	-1483.31	-1506.32	-5901.87	-7.76	-113.42	-975.89	-61.56	**-10133.20**
ChipDiffusion	-1424.05	-1215.09	-5810.44	**-2.08**	-68.66	-360.45	-62.83	-11206.90
MacroDiff+	-302.57	-958.60	**-2149.38**	-6.80	**-18.62**	-170.53	-61.12	-10432.40
FlowPlace	-94.95	-870.76	-5292.50	-5.22	-62.79	-206.77	-60.10	-11491.50
MacroSPA (w/o SPA)	-476.18	-2052.47	-11589.90	-5.16	-95.72	-304.13	-63.07	-10224.80
MacroSPA (synthetic)	**-51.49**	-847.32	-9188.50	-3.72	-55.48	-156.25	-60.74	-10509.60
MacroSPA (DP-teacher)	-111.21	-1295.69	-9143.57	-5.13	-38.58	-292.30	**-58.22**	-10215.60

NVP (count) ↓

Method	ariane81	bp68	vga_lcd	ethernet	ariane133	ariane136	dft68	mor1kx
DREAMPlace	1103	574	**4837**	99	1619	**1305**	278	10197
WireMask-EA	2143	598	7344	99	2469	2729	277	10218
ChiPFormer	3243	1128	12882	99	1406	3176	280	10220
ChipDiffusion	3152	574	12743	**73**	825	2270	**276**	10338
MacroDiff+	2146	570	12234	84	**795**	1832	279	10219
FlowPlace	1319	574	11085	84	1136	1877	278	10186
MacroSPA (w/o SPA)	2435	1679	16329	105	1418	2051	277	10257
MacroSPA (synthetic)	**785**	**569**	19953	86	1204	1757	**276**	10179
MacroSPA (DP-teacher)	1723	1128	21612	99	1144	2951	277	**10062**

Power (mW) ↓

Method	ariane81	bp68	vga_lcd	ethernet	ariane133	ariane136	dft68	mor1kx
DREAMPlace	189.80	154.70	194.00	118.00	345.20	376.30	226.40	256.70
WireMask-EA	192.60	152.00	185.80	**117.00**	352.86	409.62	232.56	254.00
ChiPFormer	188.70	154.90	188.80	118.00	348.60	387.20	228.20	255.20
ChipDiffusion	193.08	**145.13**	187.63	117.19	344.61	386.83	229.36	254.67
MacroDiff+	191.41	154.59	191.04	117.53	345.13	379.96	231.28	254.96
FlowPlace	192.09	159.01	**181.87**	117.13	**341.40**	**376.23**	222.04	**253.05**
MacroSPA (w/o SPA)	189.71	159.02	191.19	117.82	350.21	379.88	**220.10**	255.42
MacroSPA (synthetic)	**187.02**	151.59	192.04	117.35	343.24	386.65	225.53	254.07
MacroSPA (DP-teacher)	187.65	156.44	194.41	117.40	344.44	388.95	222.80	254.34

Standard-cell area (10^3 μm^2) ↓

Method	ariane81	bp68	vga_lcd	ethernet	ariane133	ariane136	dft68	mor1kx
DREAMPlace	337.87	283.27	**253.31**	96.42	**382.40**	**391.25**	86.24	155.71
WireMask-EA	341.30	297.48	259.42	96.32	387.47	396.21	86.85	154.37
ChiPFormer	340.87	303.27	261.89	**96.03**	386.68	397.94	87.70	154.90
ChipDiffusion	341.31	**270.98**	279.49	96.29	385.18	394.46	**85.16**	**153.74**
MacroDiff+	338.89	282.29	261.53	96.37	386.96	392.95	88.88	156.28
FlowPlace	342.66	326.93	268.63	96.12	385.13	394.48	87.83	155.27
MacroSPA (w/o SPA)	338.58	283.82	256.96	96.33	385.33	394.26	85.17	155.08
MacroSPA (synthetic)	**336.88**	279.25	256.01	96.27	383.29	395.40	85.55	154.55
MacroSPA (DP-teacher)	338.65	284.07	262.66	96.26	384.78	394.40	85.56	154.62

E Comparison with Related Spatial Prior Methods

Spatial knowledge and evaluation results can guide placement through complementary mechanisms. IncreMacro uses an optimization cost to encourage peripheral placement, while MaskRegulate encodes regularity in its policy input and reward (Pu et al., 2024; Xue et al., 2024). LaMPlace learns cross-stage metric predictors from offline layout–metric pairs and converts their predictions into placement masks (Geng et al., 2025). ChiPFormer uses evaluated returns to weight online policy updates on an unseen circuit (Lai et al., 2023). EXPlace encodes expert spatial knowledge in policy inputs and rewards, then fine-tunes the policy using preference pairs constructed from timing evaluations of sampled trajectories (Gao et al., 2026). These methods illustrate how rules, metric labels, and relative evaluations can inform placement.

Learned metric guidance and preference optimization rely on metric labels or evaluated comparisons to guide placement behavior (Geng et al., 2025; Gao et al., 2026). When target context–placement pairs are already available, they express the desired spatial organization but do not by themselves specify how alternative layouts should be scored or ranked. Using these mechanisms would therefore require an additional step of defining or learning scores or constructing preference labels that reflect the spatial organization expressed by the samples.

Generative macro placers can instead learn spatial preferences directly from layout samples generated by rules or optimization. FlowPlace generates 40,000 synthetic layouts using legal position masks and configurable spatial scores that favor peripheral macro placement. Its legalization procedure during sampling further encourages positions near the die corners (Xie et al., 2026). Similarly, MacroDiff+ uses 8,000 DREAMPlace reference layouts on augmented netlists for initial placer training (Yoon et al., 2026). These methods demonstrate how prescribed or optimizer-derived spatial organization can be incorporated through layout supervision.

ChipDiffusion provides a concrete example of the limits of inherited spatial preferences. Its study of training data design reports excessive clustering when connection length scales are larger than those represented during training, and unnecessarily long connections when they are smaller (Lee et al., 2025). To improve transfer, it trains on a broad distribution of connection scales and fine-tunes the full model for 250,000 optimization steps on 5,000 larger synthetic circuits. These observations suggest that learned relationships between connectivity and spatial proximity may require revision when target conditions change.

Compared with the methods above, MacroSPA acquires target spatial priors through compact post-training, without training a separate placer for each target prior or updating the full pretrained model. Synthetic or optimizer-generated samples specify the target spatial behavior, while separate SPA states capture the corresponding adaptations on the same frozen backbone. Each state is reused across circuits within its target setting, enabling different spatial priors to be acquired. Table 16 summarizes the comparison of related generative placers.

Table 16: Comparison of generative macro placers by layout source, training stage, and parameter updates. Training stages cover initial training and post-training adaptation. MacroSPA parameter counts are shown as updated / total.

Method	Layout source	Training stage / purpose	Parameter updates
FlowPlace	Mask-guided synthetic layouts	Initial training	Full model (6.29 M)
MacroDiff+	DREAMPlace reference layouts	Initial training	Full model (5.02 M)
ChipDiffusion	Larger-scale synthetic layouts	Post-training / Circuit-size transfer	Full model (6.29 M)
MacroSPA	Synthetic or optimizer-generated	Post-training / Target prior adaptation	SPA only (18.6 K / 0.61 M)

We compare the placement quality of these generative methods on ISPD, considering both macro-only and downstream mixed-size placement (Table 17). Relative to the matched no-SPA pipeline, MacroSPA (synthetic) reduces average macro HPWL, RUDY, and downstream wHPWL, achieving the lowest values among the compared methods. Its average geometry score also improves from

99.74% to 99.97%. FlowPlace attains a geometry score of 100%, but has the highest HPWL, RUDY, and downstream wHPWL. Its broad preference for peripheral placement may separate strongly connected macros when target connectivity favors a more compact arrangement. This can increase connection spans even when overlap is eliminated. Despite weaker macro-only performance, MacroDiff+ ranks second in downstream wHPWL among the compared generative placers. This result is consistent with its DREAMPlace-derived supervision and emphasis on reserving space for standard-cell placement (Yoon et al., 2026). These contrasting results suggest that the utility of a spatial prior depends on the placement objective, motivating adaptation to the target setting. Evaluation protocols and circuit-level results are provided in Appendices D.1.3 and D.4.1.

Table 17: Comparison of generative macro placers on ISPD. Each metric is averaged over eight circuits. Best results are in bold, and the second best are underlined.

Method	HPWL (10^5) ↓	RUDY (10^2) ↓	Geometry (%) ↑	wHPWL (10^6) ↓
ChipDiffusion	<u>45.47</u>	5.74	99.22	243.49
MacroDiff+	103.35†	8.74†	93.13†	<u>236.53</u>
FlowPlace	112.63	18.53	**100**	304.37
MacroSPA (w/o SPA)	46.70	<u>5.62</u>	99.74	237.42
MacroSPA (synthetic)	**42.33**	**5.29**	<u>99.97</u>	**232.10**

† Averages include imputed values. See Appendix D.1.3 for reporting conventions.

We extend the comparison to routing and post-route PPA on ChiPBench (Table 18). Circuit-level results are provided in Appendix D.4.4. Synthetic and DP-teacher SPA achieve the best and second best overall average ranks among the evaluated generative placers, whereas the matched pipeline without SPA ranks last. FlowPlace ranks best on power but poorly on wirelength and congestion, consistent with the trends indicated by HPWL and RUDY on ISPD. Between the two SPA variants, DP-teacher ranks better on congestion and WNS, while synthetic SPA ranks better on wirelength, TNS, and standard-cell area. These results show that adapting the same pretrained base model can improve aggregate downstream performance, with the benefits of each target prior varying across metrics.

Table 18: Comparison of average routing and post-route PPA ranks on ChiPBench. Ranks are computed per circuit among the methods shown and then averaged across eight circuits. The last column averages the seven metric ranks. Lower is better. Best results are in bold, and the second best are underlined, including ties.

Method	**Wirelength**	**Cong.**	**WNS**	**TNS**	**NVP**	**Power**	**Area**	**Average**
ChipDiffusion	3.75	3.88	4.63	4.38	3.50	3.63	<u>3.25</u>	3.86
MacroDiff+	3.50	3.63	3.38	3.00	2.94	4.25	4.38	3.58
FlowPlace	4.63	4.50	3.75	3.25	<u>2.88</u>	**2.13**	4.50	3.66
MacroSPA (w/o SPA)	<u>3.13</u>	**2.88**	4.38	5.00	4.94	4.25	<u>3.25</u>	3.97
MacroSPA (DP-teacher)	<u>3.13</u>	<u>3.00</u>	**2.13**	<u>2.88</u>	4.19	3.88	<u>3.25</u>	<u>3.21</u>
MacroSPA (synthetic)	**2.88**	3.13	<u>2.75</u>	**2.50**	**2.56**	<u>2.88</u>	**2.38**	**2.72**

As summarized in Table 16, FlowPlace and MacroDiff+ learn spatial preferences from synthetic or optimizer-generated layouts during initial training. By contrast, MacroSPA acquires target spatial priors through compact post-training. These downstream results complement the matched prior acquisition experiments and full fine-tuning comparisons in Appendix D.2. Together, they support the practical value of adapting inherited spatial priors to target design objectives.