# PLACEREASONER-PRO: AN 8B OPEN-WEIGHT REASONING MODEL FOR FRONTIER-LEVEL VLSI MACRO PLACEMENT

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Reasoning-driven agents have substantially advanced VLSI (i.e., chip) macro placement, but their reliance on proprietary frontier models raises severe concerns on confidentiality, specialization, and inference cost. We present **PlaceReasoner-Pro**, an 8B open-weight vision-language model for macro placement, trained with supervision that moves step by step from cheap geometric checks to the true objective: post-route power, performance, and area (PPA). Because real designs are proprietary and each place-and-route (P&R) run is expensive, we first build a circuit-agnostic synthetic environment that generates realistic floorplans (core outlines, macro sizes, and pin sides) and automatically verifies that a placement is complete, inside the core, and overlap-free; it yields 10K verified, reasoning-annotated placements for supervised fine-tuning. Two-stage reinforcement learning then refines the model: 1) Stage-I uses this verifier as the reward to make placements geometrically legal, raising overlap-free placement from 78.1% to 90.9%; and 2) Stage-II runs every candidate through a full OpenROAD P&R flow and uses the measured PPA as the reward, improving total negative slack (TNS) over Stage-I on 13 of 16 benchmark tasks. **On these 16 tasks, PlaceReasoner-Pro achieves timing comparable to Claude Opus 4.8 in the same harness (7 vs. 7 wins on worst negative slack, 7 vs. 9 on TNS) while running entirely on premises.** Our results show that compact, domain-specialized small models can approach frontier-level macro placement without relying on proprietary inference.

## 1 INTRODUCTION

Macro placement, which positions and orients dozens to hundreds of macros (i.e., large blocks) in a chip canvas, is the most critical stage of physical design for very-large-scale integrated circuits (VLSI), as it largely determines much of the chip's achievable power, performance, and area (PPA) (Sait & Youssef, 1999). The task is an especially challenging constrained combinatorial optimization problem that must simultaneously account for dataflow locality, pin accessibility, non-overlap and boundary constraints, whitespace utilization, and routability. Classical analytical, black-box, and reinforcement learning (RL) placers often optimize tractable proxies such as half-perimeter wirelength (HPWL), but these objectives only indirectly capture downstream physical-design quality (Liao et al., 2022; Mirhoseini et al., 2021; Shi et al., 2023; Kahng et al., 2023). As such, production flows still rely heavily on expert designers to inspect placements, interpret place-and-route (P&R) feedback, and iteratively refine floorplans. Since each iteration can take hours to days of P&R evaluation, this expert-in-the-loop process scales poorly as design complexity increases and time-to-market windows shrink (Tseng, 2024).

Recent reasoning-driven vision-language models (VLMs) offer a new paradigm to address this problem: rather than pure optimization, they reason jointly over floorplan geometry, connectivity, design constraints, and downstream physical-design feedback. Such approaches have shown substantial gains over conventional placers (Kahng et al., 2026; Uchendu et al., 2026; Chen et al., 2026; Li et al., 2026b;a). Yet, their performance relies on proprietary frontier models, creating two primary barriers for use. **First, confidentiality.** Process design kits (PDKs), netlists, layouts, and electronic design automation (EDA) scripts are highly sensitive design assets, making external hosted inference undesirable or unacceptable in many production settings. **Second, specialization and cost.** General-



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.

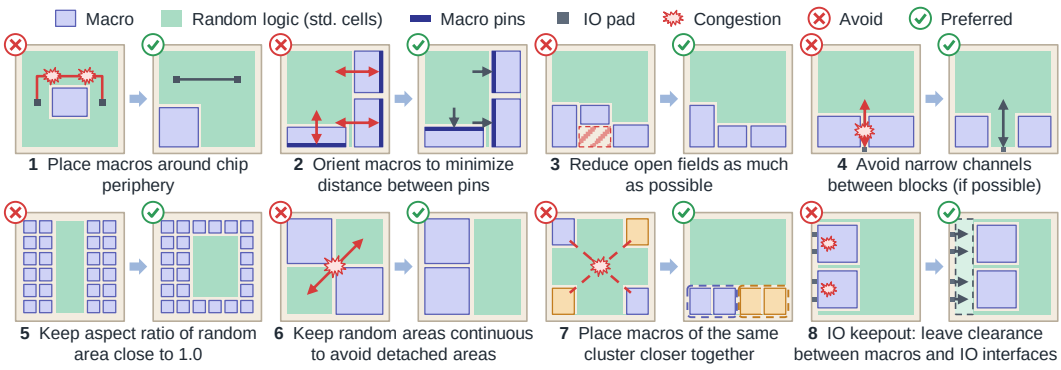


Figure 1: Macro placement principles; each panel contrasts a placement to avoid (left) with a preferred one (right).

purpose frontier models are not explicitly trained for VLSI layout geometry and can misjudge spatial relationships or violate placement instructions, requiring extensive verifier gating, repeated sampling, and expensive token costs, averaging $310 per design Kahng et al. (2026). These limitations motivate a natural alternative: a compact, open-weight model specialized for macro placement and deployable entirely on premises.

This leads to the central question of this work: **Can a small open-weight reasoning model, with proper domain-specific fine-tuning, achieve frontier-level and even beyond macro-placement quality?** The primary obstacle is not model capacity, but *supervision*. High-quality placement data are scarce and predominantly proprietary, while the ultimate objective, i.e., post-route PPA, requires expensive P&R evaluation. Our key insight is that these two difficulties need not be solved simultaneously. Much of the competence required to construct a good macro placement is *geometric and transferable*: designers repeatedly apply principles, such as placing macros near the periphery, orienting pins toward connected logic and fragmented whitespace, maintaining a compact standard-cell region, clustering strongly connected macros, and preserving clearance around I/O interfaces (Figure 1). These principles rely primarily on geometry and connectivity rather than on a particular PDK or EDA environment. Thus, we can **decouple *learning how to place* from *learning which placements optimize downstream PPA***: the former can be learned cheaply from synthetic, programmatically verifiable supervision, while the latter can be refined using expensive EDA feedback only after the model has acquired strong placement competence.

Based on this insight, we develop a synthetic-physical-collaboration training framework, where each part is defined by the environment that supplies its reward. **Synthetic training: learn placement reasoning in a synthetic environment.** We construct a circuit-agnostic environment with realistic floorplan dimensions, macro geometries, and pin configurations, and generate 10K supervised fine-tuning (SFT) trajectories and 2K RL tasks. Synthetic training first trains the model by SFT and then by RL against programmatically verifiable rewards, teaching spatial reasoning, constraint satisfaction, and adherence to placement principles without requiring proprietary circuit data or costly P&R runs. **Physical training: optimize physical design quality in a real EDA environment.** Starting from the geometrically competent model produced by the synthetic stage, we perform agentic RL within a complete open-source P&R flow (Ajayi & Blaauw, 2019; NC State EDA, 2011). Crucially, the reward is *measured post-route PPA*, rather than a placement proxy such as HPWL. The model therefore learns which placement decisions actually improve downstream implementation quality.

The resulting model, **PlaceReasoner-Pro**, fine-tunes an 8B open-weight VLM (Bai et al., 2025) into a specialized macro-placement reasoner. We evaluate PlaceReasoner-Pro and proprietary frontier models using an identical placement harness, controlling the surrounding agentic workflow. Thus, differences only reflect the underlying model rather than tooling. Synthetic training substantially improves placement validity, instruction following, and placement quality over the untuned base model; Physical training further improves downstream physical-design quality. **Together, the synthetic-physical training enables PlaceReasoner-Pro to achieve macro-placement quality comparable to Claude Opus 4.8 (Anthropic, 2026), while supporting fully on-premises deployment**. **Our contributions are fourfold.** (i) We show that frontier-level macro-placement performance can be

achieved by specializing an 8B open-weight VLM, providing an alternative to reliance on proprietary frontier models. (ii) We introduce a circuit-agnostic synthetic placement environment and associated 10K-SFT/2K-RL training corpus that teaches transferable geometric placement reasoning without proprietary design data. (iii) We develop a synthetic-physical-collaboration training framework that separates inexpensive acquisition of placement competence from expensive downstream optimization, with physical training directly optimizing *measured post-route PPA* through RL. (iv) We provide a controlled evaluation against frontier models under an identical placement harness, together with ablations that isolate the effects of synthetic supervision and EDA-grounded RL.

## 2 Background and Related Work

**VLSI Physical Design Flow.** VLSI physical design (PD) translates a synthesized netlist into a manufacturable layout through floorplanning and macro placement, standard-cell placement, clock-tree synthesis, and routing (Kahng et al., 2011; Sait & Youssef, 1999). Macro placement is the most consequential stage. It determines the positions and orientations of large circuit blocks and strongly constrains downstream routability, timing, and ultimately power, performance, and area (PPA) (Kahng et al., 2023). Given a floorplan, including die/core geometry, I/O locations, macro dimensions and pins, and connectivity, expert designers reason jointly about geometric constraints, dataflow, and whitespace, applying principles such as peripheral placement, connectivity-aware clustering, and pin-facing orientation (Pujari, 2020). Candidate placements are checked for geometric validity and iteratively refined using downstream P&R feedback. This workflow exposes two defining characteristics of macro placement: its inputs are inherently *multimodal*, combining spatial geometry with structured connectivity, while its most faithful objective, i.e., post-route PPA, is *expensive* to obtain because it requires downstream EDA evaluation costing hours to days. Classical and learning-based macro placers are reviewed in Appendix A.2.

**Agentic and Reasoning-Driven Macro Placement.** Recent work instead casts macro placement as a *multimodal reasoning* problem, building on VLMs' spatial capabilities for localization, relative-position estimation, and geometric understanding (Radford et al., 2021; Wang et al., 2024; 2025; Chen et al., 2024; Ran et al., 2026). VeoPlace (Uchendu et al., 2026) uses a VLM to guide the search space of an analytical placer, while MAGE (Kahng et al., 2026) and CoEvoP&R (Chen et al., 2026) employ agentic workflows in which VLMs propose, evaluate, and iteratively refine physical designs; PDAgent-Bench (Li et al., 2026a) further benchmarks agent capabilities across PD tasks. Collectively, these studies show that VLMs can exploit layout imagery, connectivity, design rules, and iterative feedback in ways that are difficult to encode in a single handcrafted optimization objective. Yet, existing reasoning-driven approaches largely treat the VLM as a fixed, general-purpose reasoner and rely on proprietary frontier models for strong performance. Whether macro-placement competence can instead be *learned through domain-specific post-training* in a compact open-weight model remains largely unexplored.

**Post-Training for Reasoning.** Modern reasoning models are commonly specialized through post-training, combining SFT on reasoning demonstrations with RL (Ouyang et al., 2022; Lambert et al., 2024). In particular, RL with verifiable rewards (RLVR) replaces learned preference rewards with programmatically checkable signals and has enabled substantial reasoning improvements in domains where solution quality can be objectively evaluated (Guo et al., 2025; Shao et al., 2024). Parameter-efficient adaptation further makes such specialization practical for small open-weight models (Hu et al., 2022). Macro placement provides a particularly suitable setting for this paradigm: geometric constraints and placement principles can be evaluated cheaply and automatically, whereas the ultimate downstream objective, i.e., post-route PPA, can be measured directly via an EDA environment.

## 3 PlaceReasoner-Pro

PlaceReasoner-Pro specializes a compact open-weight VLM for macro placement through progressively more faithful supervision. The central challenge is that post-route PPA provides the most meaningful placement signal but requires an expensive P&R run, making it impractical for teaching a model basic placement competence. We therefore organize training into two parts, each defined by the environment that supplies its reward (Figure 2). **Synthetic training** operates in a synthetic environment: SFT first teaches the model to reason about and execute macro placement, after which

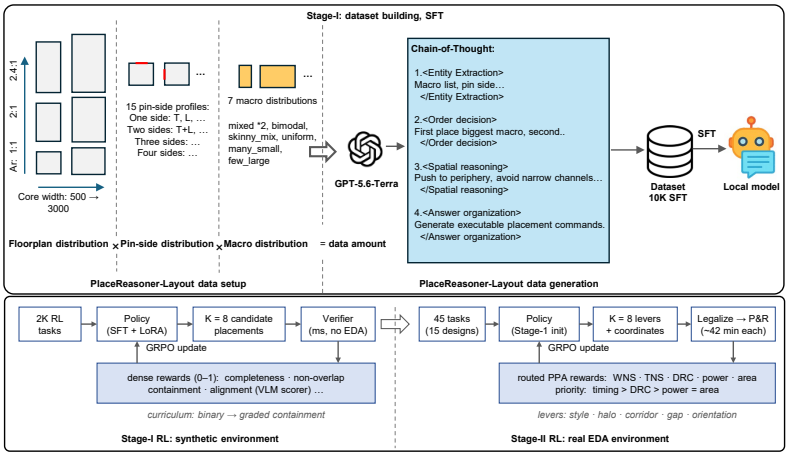


Figure 2: **Training pipeline.** *Top:* synthetic macro-placement dataset building and SFT. *Bottom:* Stage-I RL in the synthetic environment, Stage-II RL in the real EDA environment.

Stage-I RL improves geometric validity and adherence to placement principles using inexpensive verifiable rewards. **Physical training** operates in a real EDA environment, where Stage-II RL directly optimizes downstream physical-design quality using measured post-route PPA.

**Problem Formulation.** A macro placement task $\tau = (\mathcal{I}, \mathcal{M}, \mathcal{C}, \mathcal{P}, \mathcal{K})$ consists of a rendered floorplan image $\mathcal{I}$, macros $\mathcal{M} = \{(w_i, h_i)\}_{i=1}^{n}$, core bounds $\mathcal{C}$, I/O pin information $\mathcal{P}$, and cluster assignments $\mathcal{K}$ grouping macros with high connectivity affinity (Kahng et al., 2023). Given $\tau$, a policy $\pi_\theta$ generates $a = \{(x_i, y_i, o_i)\}_{i=1}^{n}$, specifying the position and orientation of every macro. Downstream standard-cell placement, CTS, and routing are held fixed when evaluating candidates, isolating the effect of macro placement. Our goal is to learn $\pi_\theta$ such that its placements are geometrically valid, follow transferable design principles, and ultimately maximize post-route PPA.

### 3.1 Synthetic Placement Environment

Practical high-quality macro-placement data are scarce and often proprietary, while evaluating placements through complete P&R is expensive. Our key observation is that much of the prerequisite competence, i.e., interpreting floorplan geometry, organizing macros spatially, satisfying constraints, and producing precise coordinates, can be learned independently of a specific netlist, PDK, or EDA tool. We therefore construct a synthetic environment that provides scalable supervision for these transferable skills.

**Environment Construction.** We build a lightweight environment that mirrors OpenROAD's (Ajayi & Blaauw, 2019) macro-placement interface, see Appendix A.3 for details on OpenROAD and the PDK. Each task instantiates a core, macros, I/O pins, and connectivity-derived macro clusters; renders the corresponding floorplan; and accepts placements in an OpenROAD-compatible format, without requiring a complete circuit netlist. We independently sample core dimensions and aspect ratios, I/O configurations, and macro profiles, drawing macro geometries from Fakeram 2.0 types (Cheng et al., 2023) and filtering infeasible configurations. This decoupling enables large-scale generation of diverse placement tasks without proprietary circuit data or P&R execution. Table 1 summarizes the configuration space.

Table 1: Configuration coverage of the synthetic placement environment.

| Dimension | Coverage |
|---|---|
| Core width | 500–3000 |
| Aspect ratio | 1–2.4 |
| I/O pin patterns | 15 |
| Macro profiles | uniform, many_small, few_large, mixed, bimodal, skinny_mix |
| Macro types | Fakeram 2.0 |
| Macros per task | 2–140 |
| Feasible tasks | 10K |

**Programmatic Verification.** As complete geometries are known, candidate placements can be evaluated in milliseconds. The verifier checks *hard constraints*, including placement completeness, core containment, non-overlap, and valid orientation, and computes graded scores for established place-

ment principles (Pujari, 2020), including peripheral placement, routing-channel quality, dead-space avoidance, standard-cell-region compactness, and cluster coherence. The same verifier serves two purposes: it filters generated supervision and provides dense rewards for Stage-I RL. Thus, it replaces expensive EDA evaluation during geometric competence learning. Importantly, the synthetic environment is intentionally *PPA-agnostic*: it teaches where macros can be placed legally and sensibly, while physical training subsequently learns which such placements perform best after routing.

### 3.2 Building Reasoning Supervision

We generate two disjoint training sets from the synthetic environment: 1) 10K tasks with validated reasoning trajectories for SFT; 2) 2K tasks without reference solutions for Stage-I RL. We term the overall dataset as **PlaceReasoner-Layout**. We reserve an additional disjoint set for evaluation.

**Reasoning-Annotated Trajectories.** Coordinates alone provide only outcome supervision and reveal little about how to construct a placement. We therefore annotate each SFT example with a structured reasoning trajectory that decomposes placement into four steps: (1) interpret floorplan geometry and macro clusters; (2) determine placement order, prioritizing large and structurally important macros; (3) select positions and orientations according to placement principles; and (4) emit executable coordinates. This decomposition provides process supervision for spatial planning in addition to the final placement target, illustrated in PlaceReasoner-Layout data generation in Fig. 2.

**Annotation and Verification.** A frontier model (Singh et al., 2025), prompted with the same placement principles available to our planner, generates candidate placements. Each candidate is checked by the programmatic verifier; hard-constraint violations trigger regeneration, while geometrically valid candidates undergo human review for higher-level placement quality. Accepted placements are then paired with four-step reasoning traces to form the SFT corpus. **It is key to note that although a frontier model assists annotation, it does not define the training objective:** examples are admitted according to programmatic verification and expert review rather than teacher-model likelihood or direct imitation of its outputs. More importantly, subsequent RL no longer optimizes against teacher demonstrations: Stage-I RL learns from environment rewards, while Stage-II RL directly optimizes measured PPA. The resulting pipeline therefore uses frontier-model annotation to bootstrap supervision rather than treating the frontier model as the target policy.

**Domain-Specific Supervised Fine-Tuning.** We fine-tune Qwen3-VL-8B (Bai et al., 2025) on the 10K annotated tasks with LoRA adapters (Hu et al., 2022), training on the standard next-token objective with the loss masked to the completion so that gradients flow only through the reasoning trace and the emitted coordinates:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(\tau,\, r,\, a)\sim\mathcal{D}}\left[\log \pi_\theta(r, a \mid \tau)\right], \tag{1}$$

where $r$ is the four-step reasoning trace and $a$ the placement. SFT establishes the basic capabilities required for placement: interpreting the multimodal task, following the prescribed reasoning procedure, producing complete and parseable outputs, and satisfying geometric constraints. In contrast, the untuned backbone frequently fails to complete large placements within the output budget or produces invalid coordinates. **SFT alone, however, optimizes imitation rather than placement quality**. It neither directly rewards solutions that improve upon their demonstrations nor exposes the model to the distribution of its own generated placements. We therefore continue post-training with RL, replacing reference imitation with environment feedback.

### 3.3 Reinforcement Learning

We progressively increase reward fidelity across the two stages: **Stage-I RL** improves geometric placement using inexpensive synthetic rewards, while **Stage-II RL** directly optimizes downstream quality using measured post-route PPA. RL in both stages optimizes $\pi_\theta$ with GRPO (Shao et al., 2024). For each task, the policy samples $K$ candidates, and each candidate $j$ is scored by several reward components $r_{j,k}$, combined into a scalar reward $R^j = \sum_k w_k \cdot r_{j,k}$ with priority weights $w_k$. This lets us compare candidates for the same task w/o a learned critic. We optimize LoRA adapters initialized from the SFT using the token-level DAPO objective (Yu et al., 2026), w/o a KL penalty.

**Stage-I RL: (i) Geometric Placement.** Stage-I RL trains on the 2K synthetic RL tasks, with $K = 8$ candidates per task, illustrated in Stage-I RL in Fig. 2. The reward combines four dense signals in

$[0, 1]$ rather than pass/fail checks. Three are computed programmatically by the verifier: *completeness*, the fraction of macros placed exactly once; *non-overlap*, one minus the fraction of placed macros involved in overlaps; and *containment*, measuring whether macros remain within the core. The fourth, *alignment*, is produced by a VLM scorer use base model and evaluates the rendered placement against nine expert design criteria. We weight alignment by 2 and each geometric reward by 1. We further multiply alignment by completeness to prevent the policy from improving apparent layout quality by omitting difficult macros; unparseable outputs receive zero reward.

**(ii) Graded Containment Reward.** We train Stage-I RL in two rounds. The first round uses binary per-macro containment, which exposes a systematic reward exploit: the policy learns to construct overlap-free rows by packing macros sequentially, but occasionally allows the final macro of an overfull row to extend beyond the core boundary. Binary containment provides insufficient resolution to correct this behavior because a 1 $\mu$m overhang receives the same penalty as a severe violation. The second round therefore replaces it with a graded containment reward that decreases continuously with overhang magnitude, providing stronger relative feedback for near-boundary violations:

$$r_{\text{core}} = 1/n \sum\nolimits_{i \in \text{placed}} \max\Big(0,\ 1 - \max\left(o_i^x/w_i,\ o_i^y/h_i\right)\Big), \tag{2}$$

where $o_i^x$ and $o_i^y$ are the distances by which macro $i$ extends beyond the core $(w_i, h_i)$ along each axis. Thus, credit falls linearly from 1 inside the core to 0 at one macro dimension outside.

**Stage-II RL: (i) Agentic RL with EDA Feedback.** Stage-I RL teaches the model to produce geometrically sound placements, but geometric quality does not uniquely determine post-route PPA. The optimal floorplanning strategy depends on circuit structure and downstream interactions among placement, congestion, timing, and routing, illustrated in Stage-II RL in Fig. 2. To enhance placement performance, Stage-II RL learns directly from measured P&R outcomes. Initialized from Stage-I RL, the policy serves as the planner agent in the PlaceReasoner-Beta framework and interacts with an OpenROAD flow using Nangate45 (Ajayi & Blaauw, 2019; NC State EDA, 2011). In a single response, it jointly specifies macro coordinates and orientations together with higher-level floorplanning decisions: a placement style (16 predefined topologies, such as ring and top band, or a custom strategy), macro halo (1–15 $\mu$m), routing corridor along pin-bearing die sides (0–150 $\mu$m), minimum macro spacing (0–20 $\mu$m), and orientation policy. Thus, Stage-II RL optimizes not only individual macro coordinates but an *agentic floorplanning strategy* for downstream implementation. The PlaceReasoner-Beta legalizer removes only residual overlaps; it rejects incomplete placements before P&R. All remaining candidates pass through an identical downstream flow comprising standard-cell placement, CTS, global and detailed routing, and timing repair, ensuring that differences in measured outcomes originate from the policy's decisions. We train on 45 tasks derived from 15 open-source designs, each instantiated at three core aspect ratios. As each rollout requires a full P&R run (42 minutes on average), each optimization step samples 8 candidates per task and evaluates them in parallel.

Table 2: **PlaceReasoner-Layout statistics** (30–70% macro area ratio). *Macros*: mean macros per task. *Complete*, *In-core*, and *Overlap-free* are averaged over the reference placements at each filtering stage; *Align* is the rubric alignment score in $[0, 1]$. *Annotated*: all annotator placements. *Accepted*: placements that are complete, in-core, and overlap-free and also pass the principle and reasoning review; they form the SFT corpus. Stage-I RL tasks have no reference placements, so their placement metrics are shown as —.

| Stage | Tasks | Macros | Complete | In-core | Overlap-free | Align |
|---|---|---|---|---|---|---|
| annotated | 13,483 | 25.1 | 0.990 | 0.988 | 0.965 | 0.551 |
| accepted (SFT) | 10,620 | 24.0 | 1.000 | 1.000 | 1.000 | 0.595 |
| Stage-I RL tasks | 2,000 | 27.2 | — | — | — | — |

**(ii) Post-Route Reward.** We compute the Stage-II reward directly from the routed design, prioritizing timing and routability while also accounting for power, area, and legalization cost. Timing is captured by worst negative slack (WNS), normalized by clock period $T$, and a log-scaled total negative slack (TNS) term, each weighted by 1.5. Routability is measured as $-\log(1 + N_{\text{DRC}})$ with weight 2.0, while power and area each receive weight 0.5. An additional overlap term with weight 0.5 penalizes displacement introduced by legalization. Log scaling compresses the heavy-tailed variation in TNS and DRC counts. Incomplete or unroutable candidates are assigned rewards below all successfully routed candidates in the same group. Together, the two stages create a progression from *cheap, dense geometric supervision* to *expensive, faithful implementation feedback*:

Stage-I RL learns how to construct geometrically strong placements, while Stage-II RL learns which placement strategies actually translate into improved post-route PPA.

# 4 EXPERIMENTS

## 4.1 EXPERIMENTAL SETUP

**(1) Models.** We evaluate Qwen3-VL-8B (Bai et al., 2025) throughout our training pipeline: QWEN-BASE (untuned), PLACEREASONER-PRO-SFT (after SFT on PlaceReasoner-Layout), PLACEREASONER-PRO-I (after Stage-I RL), and PLACEREASONER-PRO-II (after Stage-II RL). All variants run in the same PlaceReasoner-Beta Li et al. (2026b) harness and use identical netlists, floorplans, macro libraries, and P&R flows, isolating the effect of post-training. As a frontier-model reference, we run the same harness with Claude Opus 4.8 (Anthropic, 2026), denoted PLACEREASONER. Appendix A.4 provides training details. **(2) Benchmarks and Baselines.** We evaluate complementary aspects of placement using two benchmarks. PLACEREASONER-BENCH Li et al. (2026b) measures end-to-end physical-design quality after complete P&R, while PLACEREASONER-LAYOUT-BENCH isolates spatial-planning ability on held-out synthetic tasks. Designs are synthesized with Yosys (Wolf et al., 2013) and implemented with OpenROAD (Ajayi & Blaauw, 2019) using the Nangate45 technology (NC State EDA, 2011) and memory macros generated by Fakeram 2.0 (Cheng et al., 2023). On PLACEREASONER-BENCH, **we additionally compare against representative optimization- and learning-based placers**: DREAMPlace 4.0 (Liao et al., 2022), RTL-MP (Kahng et al., 2023), ChiPFormer (Lai et al., 2023), and simulated annealing (SA) (Cheng et al., 2023). **(3) Evaluation Protocol and Metrics.** We follow the evaluation settings of PlaceReasoner-Beta Li et al. (2026b). End-to-end metrics are computed only for successfully detailed-routed designs; timeouts and tool failures are recorded as failures rather than substituted with pre-route estimates. On PLACEREASONER-BENCH, we report routed worst negative slack (WNS), total negative slack (TNS), design-rule-check (DRC) violations, wirelength, total power, and chip area. On PLACEREASONER-LAYOUT-BENCH, we measure macro overlap rate (MOR, $\downarrow$), completeness ($\uparrow$), core containment ($\uparrow$), row overflow ($\downarrow$), and rubric alignment ($\uparrow$). GPT-5.6-luna independently scores alignment using a fixed rubric, with the placement model's identity hidden from the judge.

## 4.2 SYNTHETIC TRAINING RESULTS

**PlaceReasoner-Layout.** Table 2 summarizes the SFT and RL tasks in PlaceReasoner-Layout. Raw annotations are nearly legal but still contain 1.1% missing, 1.4% out-of-core, and 3.7% overlapping macros. Programmatic verification and expert review eliminate these violations, yielding a fully legal SFT corpus while increasing mean rubric alignment from 0.55 to 0.61, indicating that filtering improves design quality beyond legality. The 2K RL tasks contain no reference placements and are supervised solely by verifier rewards. For evaluation, PlaceReasoner-Layout-Bench contains 288 tasks disjoint from PlaceReasoner-Layout in core dimensions, floorplans, and macro sets (Appendix A.4), testing generalization beyond memorized training layouts.

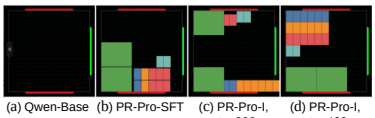


Figure 3: **One synthetic task, placed by each policy** (benchmark_0029_bimodal; 22 macros, five clusters, 2210 $\times$ 2210 core, 48% macro area ratio). Fill color denotes the cluster; bars mark pin-bearing die edges.

**From SFT to Stage-I RL.** Figure 4 isolates the contribution of each training stage. **The untuned backbone** largely fails as a placer, positioning only 24.9% of macros, with 22.9% containment and 12.5% overlap-free placement, primarily because it exhausts its output budget before completing the coordinate list. **SFT** establishes basic placement competence: completeness reaches 97.8%, while containment and overlap-free placement rise to 84.3% and 78.1%, respectively. Yet, geometric violations remain, with row overflow averaging 0.86 per task. **Stage-I RL** closes much of this gap. Containment increases to 92.9% and overlap-free placement to 90.9%, i.e., a 12.8-point gain, while row overflow drops by 74%, from 0.86 to 0.22, without sacrificing completeness. Alignment also improves from 0.552 to 0.673, indicating that RL strengthens adherence to design principles beyond constraint satisfaction. The training dynamics reveal how reward design drives these gains. Overlap-free placement

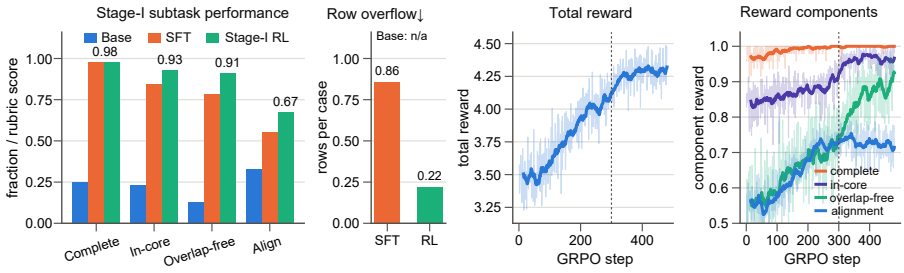


Figure 4: **Stage-I results.** *Left:* subtask performance on PlaceReasoner-Layout-Bench benchmark for QWEN-BASE (Base), PLACEREASONER-PRO-SFT (SFT), and PLACEREASONER-PRO-I (Stage-I RL). *Centre left:* overflowing rows per case, on its own axis because it is a count rather than a fraction; the base model places too few macros for the metric to be meaningful. *Right:* Stage-I RL training, showing the weighted total reward and its four components. Containment switches from a binary to a distance-graded form at step 300.

improves most strongly, while containment rises sharply after step 300, when binary containment is replaced by distance-graded feedback. Binary scoring treats a 1 $\mu$m overhang like a severe violation, providing little distinction among near-boundary errors. As Figure 3 illustrates, optimizing non-overlap initially encourages long rows of side-by-side macros, occasionally extending beyond the core. After introducing graded containment, larger overhangs incur progressively stronger penalties, prompting the policy to reorganize rather than extend overflowing rows. By step 480, the same macro clusters form compact, non-overlapping rows contained within the core.

### 4.3 PHYSICAL TRAINING RESULTS

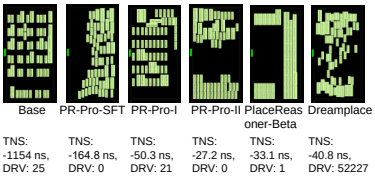


Figure 5: Macro placements on vga-lcd at aspect ratio 2:1.

We evaluate all five policies on PlaceReasoner-Bench using post-route PPA and DRV after complete P&R (Table 3). Stage-II RL consistently improves timing over earlier stages: compared with Stage-I, it achieves better WNS on 13 of 16 routed tasks and better TNS on 13 of 16, and improves TNS over SFT on all 15 tasks that SFT routes. Gains are particularly pronounced on elongated floorplans where Stage-I struggles: TNS improves from $-83.2$ to $-0.4$ ns on ariane81, $-223.9$ to $-13.9$ ns on VeriGPU, and $-50.3$ to $-27.2$ ns on vga_lcd. Routed wirelength also decreases on 11 of 16 tasks relative to Stage-I, indicating that the timing gains do not require longer interconnects. Against PlaceReasoner-Beta, which uses a frontier model within the same harness, Stage-II achieves comparable timing, with 7 versus 9 wins on TNS and 7 versus 7 on WNS (two ties). These timing gains expose a routability trade-off. Stage-II produces DRV-clean layouts on 8 of 16 tasks, compared with 11 of 16 for Stage-I, with most violations concentrated in three cases (swerv_wrapper at 2:1, bp_be at 1:1, and ethernet), which together account for 21 of the 29 violations. This behavior is consistent with the reward design: timing receives a combined weight of 3.0 versus 2.0 for DRV, allowing improvements in slack to outweigh moderate routability degradation. These results highlight reward balancing between timing and routability as an important direction for further refinement.

### 4.4 ABLATION STUDY

We ablate three design choices in Stage-I RL while holding the reward set, task pool, and training budget fixed, varying only the component under study, the results see Appendix. A.6. **(1) Reward aggregation.** GRPO normalizes the weighted aggregate reward within each rollout group, whereas GDPO (Liu et al., 2026) first normalizes each reward component and then applies its weight, decoupling an objective's influence from its raw variance. We compare the two under otherwise identical settings. **(2) Reward curriculum.** Our curriculum prioritizes geometric validity before emphasizing alignment with design principles. To isolate the effect of scheduling, we compare it against a fixed scheme in which all rewards are active from the first step at their final weights. **(3) Alignment**

Table 3: Post-route PPA and DRV on the eight PlaceReasoner-Bench designs, under square (1:1) and elongated (2:1) aspect ratios. PR-BETA is the PlaceReasoner-Beta harness driven by Claude Opus 4.8; the four PR-PRO rows are the same harness driven by our 8B policy at successive stages of training. Every row shares the same netlists, floorplans, and P&R scripts, so differences come from the macro placement alone. Best WNS and TNS per design and aspect ratio are in bold; lower DRV is better. *Fail*: routing did not complete (cause given).

| | | Square (1:1) | | | | | | Elongated (2:1) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Design | Policy | WNS (ns) | TNS (ns) | DRV | WL ($\mu$m) | Power (W) | Area ($\mu m^2$) | WNS (ns) | TNS (ns) | DRV | WL ($\mu$m) | Power (W) | Area ($\mu m^2$) |
| ariane133 | PR-Beta (Opus) | -0.02 | -0.35 | 0 | 5622394 | 0.258 | 728851 | **-0.02** | **-0.08** | 0 | 5690597 | 0.285 | 730228 |
| | Qwen-Base | *Fail: DRT-0073* | | | | | | *Fail: GRT-0116* | | | | | |
| | PR-Pro-SFT | -0.06 | -3.58 | 0 | 6102605 | 0.266 | 733242 | *Fail: route* | | | | | |
| | PR-Pro-I | -0.06 | -0.84 | 0 | 6134664 | 0.294 | 729788 | -0.12 | -11.33 | 1 | 6131396 | 0.293 | 729938 |
| | PR-Pro-II | **-0.01** | **-0.21** | 0 | 6100372 | 0.284 | 728322 | -0.06 | -2.52 | 2 | 6208345 | 0.292 | 729633 |
| ariane81 | PR-Beta (Opus) | -0.04 | -1.20 | 0 | 5203922 | 0.171 | 692142 | **-0.02** | **-0.15** | 0 | 5798709 | 0.194 | 698754 |
| | Qwen-Base | -0.09 | -6.24 | 0 | 6631628 | 0.198 | 693184 | -0.40 | -310.10 | 28 | 7017977 | 0.202 | 694691 |
| | PR-Pro-SFT | -0.07 | -13.59 | 0 | 6946901 | 0.201 | 694171 | -0.50 | -356.16 | 0 | 6554251 | 0.199 | 694266 |
| | PR-Pro-I | -0.03 | -0.22 | 0 | 6159825 | 0.197 | 692675 | -0.16 | -83.17 | 5 | 6970316 | 0.198 | 695819 |
| | PR-Pro-II | **-0.01** | **-0.05** | 0 | 6441924 | 0.197 | 693138 | **-0.02** | -0.42 | 0 | 6655187 | 0.197 | 692407 |
| bp_be | PR-Beta (Opus) | **-0.13** | **-9.42** | 0 | 2287187 | 0.118 | 238089 | **-0.22** | **-18.06** | 1 | 2873409 | 0.109 | 240882 |
| | Qwen-Base | -0.29 | -28.59 | 0 | 2435295 | 0.107 | 238967 | -0.39 | -31.03 | 0 | 2647452 | 0.107 | 239686 |
| | PR-Pro-SFT | -0.23 | -20.07 | 0 | 2606992 | 0.105 | 239488 | -0.33 | -31.56 | 0 | 2810438 | 0.108 | 239999 |
| | PR-Pro-I | -0.19 | -18.49 | 0 | 2454375 | 0.106 | 239509 | -0.33 | -32.06 | 0 | 2700561 | 0.106 | 239595 |
| | PR-Pro-II | -0.16 | -14.38 | 7 | 2526559 | 0.104 | 239494 | -0.23 | -21.84 | 0 | 2540448 | 0.105 | 239185 |
| bp_fe | PR-Beta (Opus) | **-0.03** | **-0.07** | 0 | 1553678 | 0.141 | 213874 | -0.11 | **-0.23** | 0 | 1638383 | 0.131 | 217638 |
| | Qwen-Base | -0.13 | -4.75 | 0 | 1550338 | 0.126 | 215049 | -0.09 | -2.43 | 0 | 1719403 | 0.135 | 219808 |
| | PR-Pro-SFT | -0.04 | -0.19 | 0 | 1557515 | 0.128 | 214973 | **-0.06** | -0.57 | 0 | 1700999 | 0.131 | 218035 |
| | PR-Pro-I | -0.04 | -0.09 | 0 | 1565052 | 0.129 | 215248 | -0.17 | -7.70 | 0 | 1653510 | 0.131 | 216427 |
| | PR-Pro-II | -0.04 | -0.14 | 0 | 1532330 | 0.127 | 214990 | -0.07 | -0.28 | 0 | 1610257 | 0.128 | 214585 |
| ethernet | PR-Beta (Opus) | -0.30 | -1.44 | 0 | 760262 | 0.0142 | 110597 | -0.28 | -1.34 | 0 | 825144 | 0.0139 | 110333 |
| | Qwen-Base | -0.30 | -1.45 | 0 | 735008 | 0.0171 | 110435 | -0.29 | -1.43 | 0 | 761143 | 0.017 | 110252 |
| | PR-Pro-SFT | -0.30 | -1.34 | 0 | 692125 | 0.0169 | 110196 | -0.30 | -1.47 | 0 | 845543 | 0.0172 | 110111 |
| | PR-Pro-I | -0.30 | -1.38 | 0 | 750573 | 0.0172 | 110454 | -0.29 | -1.40 | 393 | 773340 | 0.0171 | 110260 |
| | PR-Pro-II | **-0.28** | **-1.30** | 4 | 723829 | 0.0171 | 110434 | **-0.26** | **-1.28** | 2 | 781740 | 0.0169 | 110202 |
| swerv_wrapper | PR-Beta (Opus) | **-0.42** | **-409.13** | 0 | 3447721 | 0.243 | 647807 | -0.58 | -476.28 | 2 | 3925621 | 0.242 | 648145 |
| | Qwen-Base | -0.60 | -617.00 | 0 | 4067535 | 0.249 | 646917 | -0.67 | -755.17 | 0 | 4471772 | 0.245 | 646254 |
| | PR-Pro-SFT | -0.70 | -729.67 | 1 | 4060092 | 0.253 | 645750 | -0.61 | -529.81 | 1 | 4485502 | 0.245 | 647056 |
| | PR-Pro-I | -0.58 | -484.92 | 0 | 4343919 | 0.250 | 647000 | -0.62 | -666.94 | 26 | 3988478 | 0.241 | 646140 |
| | PR-Pro-II | -0.56 | -488.89 | 1 | 3836204 | 0.242 | 646136 | **-0.51** | **-466.50** | 8 | 3856127 | 0.241 | 645894 |
| vga_lcd | PR-Beta (Opus) | **-0.74** | -37.73 | 0 | 2342025 | 0.180 | 651477 | -0.76 | -33.13 | 1 | 2547982 | 0.183 | 649875 |
| | Qwen-Base | -0.89 | -41.92 | 0 | 2449050 | 0.190 | 648133 | -0.76 | -1154.16 | 25 | 2605124 | 0.187 | 648128 |
| | PR-Pro-SFT | -0.78 | -33.09 | 0 | 2513284 | 0.176 | 648240 | -0.85 | -164.84 | 0 | 2506564 | 0.181 | 647357 |
| | PR-Pro-I | -0.75 | **-30.81** | 0 | 2672866 | 0.191 | 648505 | -0.91 | -50.25 | 21 | 2875520 | 0.193 | 648135 |
| | PR-Pro-II | -0.75 | -31.88 | 0 | 2232123 | 0.187 | 647353 | **-0.68** | **-27.23** | 0 | 2464813 | 0.181 | 646603 |
| VeriGPU | PR-Beta (Opus) | **-0.25** | **-12.31** | 0 | 1697586 | 0.0911 | 274109 | **-0.28** | **-13.08** | 0 | 1896523 | 0.101 | 274461 |
| | Qwen-Base | **-0.25** | -17.43 | 0 | 1705359 | 0.128 | 274236 | -0.45 | -54.06 | 0 | 1906068 | 0.129 | 274392 |
| | PR-Pro-SFT | -0.30 | -14.68 | 0 | 1710691 | 0.129 | 274565 | **-0.28** | -15.40 | 0 | 1994577 | 0.128 | 274655 |
| | PR-Pro-I | **-0.25** | -13.34 | 0 | 1691729 | 0.128 | 274485 | -0.42 | -223.85 | 0 | 2031801 | 0.128 | 274802 |
| | PR-Pro-II | -0.26 | -12.48 | 4 | 1701958 | 0.133 | 274213 | **-0.28** | -13.92 | 1 | 1903553 | 0.128 | 274565 |

**evaluator.** As alignment is the only learned reward, we examine both its necessity and evaluator choice. We compare training without alignment, with our distilled 8B scorer, and with the frontier judge directly in the RL loop.

## 5 CONCLUSION

We presented PlaceReasoner-Pro, which specializes an 8B open-weight VLM for macro placement through synthetic supervision, geometric RL, and EDA-grounded RL with post-route PPA rewards. PlaceReasoner-Pro achieves placement quality competitive with frontier models while enabling fully on-premises deployment. More broadly, our results demonstrate that compact models can acquire strong domain-specific reasoning through progressively more faithful supervision.

## REFERENCES


Tutu Ajayi and David Blaauw. Openroad: Toward a self-driving, open-source digital layout implementation tool chain. In *Proceedings of Government Microcircuit Applications and Critical Technology Conference*, 2019.

Anthropic. Introducing claude opus 4.8. https://www.anthropic.com/news/claude-opus-4-8, May 2026. Accessed: 2026-07-29.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Yun-Chih Chang, Yao-Wen Chang, Guang-Ming Wu, and Shu-Wei Wu. B*-trees: A new representation for non-slicing floorplans. In *Proceedings of the 37th annual design automation conference*, pp. 458–463, 2000.

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

Ruogu Chen, Weihua Xiao, Ramesh Karri, and Jie Han. Coevop&r: Co-evolving placement objectives with routing feedback via large language models. *arXiv preprint arXiv:2607.17398*, 2026.

Tung-Chieh Chen, Zhe-Wei Jiang, Tien-Chang Hsu, Hsin-Chen Chen, and Yao-Wen Chang. Ntuplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(7):1228–1240, 2008.

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

Chung-Kuan Cheng, Andrew B Kahng, Sayak Kundu, Yucheng Wang, and Zhiang Wang. Assessment of reinforcement learning for macro placement. In *Proceedings of the 2023 International Symposium on Physical Design*, pp. 158–166, 2023.

Jason Cong and Min Xie. A robust mixed-size legalization and detailed placement algorithm. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(8):1349–1362, 2008.

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

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Andrew B Kahng, Jens Lienig, Igor L Markov, and Jin Hu. *VLSI physical design: from graph partitioning to timing closure*, volume 312. Springer, 2011.

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, 2023.

Andrew B Kahng, Sayak Kundu, and Bodhisatta Pramanik. Mage: Human-like macro placement via agentic multimodal reasoning. *arXiv preprint arXiv:2607.18536*, 2026.

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

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*, pp. 18346–18364. PMLR, 2023.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Qiufeng Li, Rongqian Chen, Quan Cheng, Chengxuan Wang, Sizhe Tang, Chia-Tung Ho, David Z Pan, Tian Lan, and Weidong Cao. Pdagent-bench: Characterizing, grounding, and architecting llm/vlm agents for vlsi physical design. *arXiv preprint arXiv:2606.17253*, 2026a.

Qiufeng Li, Chengxuan Wang, Rongqian Chen, Quan Cheng, Yihui Ren, Chia-Tung Ho, David Z. Pan, Tian Lan, and Weidong Cao. Placereasoner-beta: Reasoning-driven macro placement and benchmarking, 2026b. URL https://arxiv.org/abs/2609.21263.

Peiyu Liao, Siting Liu, Zhitang Chen, Wenlong Lv, Yibo Lin, and Bei Yu. Dreamplace 4.0: Timing-driven global placement with momentum-based net weighting. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 939–944, 2022. doi: 10.23919/DATE54114.2022.9774725.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, et al. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization. *arXiv preprint arXiv:2601.05242*, 2026.

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.

NC State EDA. FreePDK45. https://eda.ncsu.edu/freepdk/freepdk45/, 2011. Version 1.4; accessed July 22, 2026.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Swasti Pujari. Macro placement guidelines. iVLSI Technologies, August 2020. Accessed: July 21, 2026.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Xingjian Ran, Yixuan Li, Linning Xu, Mulin Yu, and Bo Dai. Direct numerical layout generation for 3d indoor scene synthesis via spatial reasoning. *Advances in Neural Information Processing Systems*, 38:125055–125081, 2026.

Sadiq M Sait and Habib Youssef. *VLSI physical design automation: theory and practice*, volume 6. World Scientific Publishing Company, 1999.

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

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

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

I-Lun Tseng. Challenges in floorplanning and macro placement for modern socs. In *Proceedings of the 2024 International Symposium on Physical Design*, pp. 71–72, 2024.

Ikechukwu Uchendu, Swati Goel, Karly Hou, Ebrahim Songhori, Kuang-Huei Lee, Joe Wenjie Jiang, Vijay Janapa Reddi, and Vincent Zhuang. See it to place it: Evolving macro placements with vision-language models. *arXiv preprint arXiv:2603.28733*, 2026.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025.

Clifford Wolf, Johann Glaser, and Johannes Kepler. Yosys-a free verilog synthesis suite. In *Proceedings of the 21st Austrian Workshop on Microelectronics (Austrochip)*, volume 97, pp. 1–6, 2013.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

# A Appendix

## A.1 AI Use Statement

We used large language models (LLMs) in two roles. **(1) Writing assistance.** LLMs were used to polish the wording of the paper and to check consistency between the text and tables. **(2) Components of the method and evaluation.** As described in the main text, GPT-5.6-Terra annotates the reasoning trajectories of the synthetic SFT dataset (Appendix A.4), GPT-5.6-luna serves as the rubric judge for the alignment metric, and Claude Opus 4.8 drives the PlaceReasoner-Beta baseline. The research ideas, method design, experiments, and analysis were developed by the authors, who take full responsibility for all content of this paper.

## A.2 Classical and Learning-Based Macro Placement

Classical macro placers broadly follow packing-based or analytical optimization paradigms (Chen et al., 2008; Cheng et al., 2018; Chang et al., 2000; Kahng et al., 2023). Packing-based methods represent relative macro positions using structures such as sequence pairs or B*-trees and explore placements through stochastic search (Chang et al., 2000). Analytical methods optimize differentiable surrogates combining smoothed wirelength with density and overlap penalties, followed by legalization (Liao et al., 2022; Cong & Xie, 2008). Learning-based approaches formulate placement as sequential decision-making and train reinforcement learning (RL) policies to select macro locations (Mirhoseini et al., 2021; Lai et al., 2022; 2023; Geng et al., 2024; Shi et al., 2023). Despite different optimization mechanisms, these approaches predominantly rely on tractable proxy objectives such as half-perimeter wirelength (HPWL), which do not fully capture downstream physical-design quality, e.g., PPA (Cheng et al., 2023). Moreover, prior placement RL typically optimizes a policy for a particular circuit over a coordinate-level action space. It therefore primarily learns a search policy for a specific instance, rather than acquiring reusable placement principles that can transfer across designs.

## A.3 EDA Tools and PDKs

**OpenROAD.** OpenROAD (Ajayi & Blaauw, 2019) is an open-source physical design toolchain developed by an academic consortium led by UC San Diego under DARPA's IDEA program, and

Table 4: Composition of PlaceReasoner-Layout-Bench: 288 purpose-built evaluation tasks, drawn from configurations disjoint from training.

| Dimension | Coverage |
|---|---|
| Tasks | 288 (180 at 30–40%, 108 at 40–70%) |
| Macro area ratio | 32–70% |
| Core widths | 560–2760 (21 values) |
| Aspect ratios | 1:1, 2:1, 2.4:1 |
| I/O pin patterns | 15 (all one-, two-, three-, four-side combinations) |
| Macro profiles | uniform, many_small, few_large, mixed, bimodal, skinny_mix |
| Macros per task | 2–125 (mean 26.5) |

the most widely used open P&R flow in physical design research. It is the open counterpart of commercial tools such as Cadence Innovus and Synopsys IC Compiler II, and covers every stage our experiments exercise: floorplanning, macro and standard-cell placement, CTS, and global and detailed routing. Two properties make it usable as an RL environment. Every stage is scriptable through its Tcl and Python interfaces, so a candidate placement can be injected and driven to a routed design with no manual intervention. And it reports post-route timing, power, area, and routing violations, which is precisely the signal Stage 2 optimizes.

**Nangate45.** Our experiments run on the Nangate 45 nm open cell library over FreePDK45 (NC State EDA, 2011), the standard open technology for academic P&R studies. It is a predictive rather than manufacturable PDK: its libraries and rule decks support placement, routing, and the timing, power, area, and violation counts we use as rewards, but not silicon sign-off. The violation counts we report come from the detailed router's own rule checks rather than from a foundry DRC deck.

### A.4 Experimental Setup

**Models and training.** Every method runs inside the same PlaceReasoner-Beta harness, so measured differences reflect the underlying model rather than the surrounding agentic workflow; in tables, PlaceReasoner denotes that harness driven by Claude Opus 4.8 (Anthropic, 2026). We evaluate Qwen3-VL-8B (Bai et al., 2025) at four points of our pipeline: Qwen-Base (untuned), PlaceReasoner-Pro-SFT (after SFT on PlaceReasoner-Layout), PlaceReasoner-Pro-I (after Stage-I RL), and PlaceReasoner-Pro-II (after Stage-II RL). SFT applies LoRA (Hu et al., 2022) ($r$=16, $\alpha$=32, dropout 0.05) to the attention and MLP projections of the language model while freezing the vision tower and projector, and trains for three epochs at learning rate $10^{-4}$ with a cosine schedule. Unless otherwise specified, generation temperature is 0.1. GPT-5.6-Terra annotates the SFT reasoning trajectories, and GPT-5.6-luna serves as the rubric judge for the alignment metric.

**EDA flow.** We use Yosys (Wolf et al., 2013) for logic synthesis, OpenROAD (Ajayi & Blaauw, 2019) for P&R, and Fakeram 2.0 (Cheng et al., 2023) to generate memory macros, all under the open-source Nangate45 technology (NC State EDA, 2011). Unless stated otherwise, all methods share the same netlist, floorplan, macro library, and P&R scripts, so only the macro placement differs.

**Benchmarks and baselines.** PlaceReasoner-Bench measures end-to-end quality after a complete P&R run on eight open-source designs (Table 5), each instantiated at a square (1:1) and an elongated (2:1) core aspect ratio. PlaceReasoner-Layout-Bench instead isolates geometric and spatial placement reasoning and alignment to design principles, without PPA. It is purpose-built rather than a random held-out split: its core widths are offset midway between the training grid (560–2760 in steps of 110, against 500–2760 in training), no floorplan is shared with training, and every task draws a fresh macro set, so a policy cannot succeed by recalling a training layout (Table 4). On PlaceReasoner-Bench we compare against four representative placers: DREAMPlace 4.0 (Liao et al., 2022), a GPU-accelerated analytical placer; RTL-MP (Kahng et al., 2023), a hierarchy- and dataflow-aware placer; ChiPFormer (Lai et al., 2023), an offline RL method; and a conventional simulated-annealing (SA) packing baseline (Cheng et al., 2023).

Table 5: Composition of PlaceReasoner-Bench. #Types counts distinct macro types, Clk is the target clock period, and M/Core is macro area as a fraction of core area. "–" denotes a quantity without aggregation.

| Design | #Macros | #Clusters | #Types | Clk (ns) | M/Core (%) |
|---|---|---|---|---|---|
| ariane133 | 133 | 14 | 1 | 4.0 | 34.4 |
| ariane81 | 81 | 8 | 6 | 4.0 | 36.0 |
| bp_be | 10 | 3 | 3 | 2.6 | 36.7 |
| bp_fe | 11 | 4 | 3 | 1.8 | 45.6 |
| swerv_wrapper | 28 | 3 | 3 | 2.0 | 44.1 |
| vga_lcd | 62 | 12 | 3 | 2.0 | 40.5 |
| ethernet | 64 | 2 | 1 | 8.0 | 14.2 |
| VeriGPU | 12 | 1 | 1 | 1.2 | 30.5 |
| Total | 401 | 47 | – | – | – |

**Protocol and metrics.** On PlaceReasoner-Bench we report routed worst negative slack (WNS), total negative slack (TNS), design-rule-check (DRC) violations, routed wirelength, total power, and chip area. Metrics are reported only after successful detailed routing: timeouts and tool failures count as failed runs rather than being replaced with pre-route estimates. On PlaceReasoner-Layout-Bench we report macro overlap rate (MOR; lower is better) together with completeness, core containment, row overflow, and the rubric alignment score (higher is better). The rubric judge scores every layout blind: it is never told which model produced a placement.

### A.5 Stage-II RL Setting

**Circuit set.** Stage-II trains on 15 open-source designs taken from OpenROAD-flow-scripts and ChiPBench, each built on Nangate45 at 60% core utilization and instantiated at three core aspect ratios (1:1, 2:1, 2.4:1), giving 45 placement tasks (Table 6). The set is deliberately heterogeneous: instance counts span 10K to 321K, macro counts 2 to 136 drawn from 1 to 18 distinct types, cluster counts 1 to 64, and clock targets 1.0 to 10.6 ns, so a strategy fitted to one design size or memory mix does not transfer for free. PDK, utilization, and the P&R template are identical across all 45 tasks, and the reward normalizes timing by each design's clock period, so a single policy is trained against designs of very different scale. No task is shared with the PlaceReasoner-Bench designs used for evaluation, although three families (Ariane, and the BlackParrot front- and back-end) appear in both under different macro configurations.

Beyond these shared families, the two sets(Circuit set and PlaceReasoner-Bench) are close in design class. 14 of the 15 training designs are processor cores or multicore clusters, and five of the eight PlaceReasoner-Bench designs (both Ariane variants, bp_be, bp_fe, and swerv_wrapper) are RISC-V cores, so the policy meets the same kind of floorplan at evaluation as in training: SRAM macros for caches and register files arranged around the pipeline logic that accesses them. Placement experience gathered on these designs can therefore be reused directly on the benchmark. The remaining benchmark designs (vga_lcd, ethernet, and VeriGPU) have no close counterpart in training, and for them we rely on the general macro-placement knowledge that Stage-II RL learns across all 45 tasks, such as keeping macros at the core boundary, leaving pin-side corridors clear, and choosing halos and spacing that leave routing resources for standard cells.

**Why small circuit set is sufficient.** Stage-II does not teach the model to place macros from scratch. General placement ability is acquired earlier, on synthetic data: SFT on 10K verified trajectories and Stage-I RL on 2K tasks already produce complete, in-core, and largely overlap-free placements (90.9% overlap-free on Layout-Bench, Figure 4). Stage-II therefore has a narrower goal: to learn which floorplanning style pays off in post-route PPA for a given class of designs, and how much halo and corridor space to leave for routing. This style is shared across designs of the same class, so a small set of representative designs is enough to learn it, and each task already yields a dense signal, as every step returns eight fully routed PPA outcomes. This small-set setting also matches how we expect the model to be deployed. With Stage-II, users can start from our Stage-I model and specialize it on a modest number of its own designs, entirely on premises. Scaling Stage-II to larger and more diverse circuit sets is a natural extension that we leave to future work.

**Rollout loop.** The policy is initialized from the Stage-I LoRA and acts as the planner agent of PlaceReasoner-Beta. Each step samples $K{=}8$ candidates for a single task and cycles through the

Table 6: Circuit set for Stage-II RL training. Every design is built on Nangate45 at 60% core utilization under three core aspect ratios (1:1, 2:1, 2.4:1), giving 45 placement tasks. Instances, macros and IO pins are counted in the shipped floorplan DEF; *Die* is the 1:1 die edge; all three aspect ratios of a design share the same core area. *Clusters* is the macro cluster count produced by RTL-MP autoclustering.

| Design | Description | Source | #Insts | #Macros | #Types | #Clust. | Clk (ns) | Die ($\mu$m) | IO |
|---|---|---|---|---|---|---|---|---|---|
| ariane | RISC-V application core (PULP Ariane) | ORFS | 161,584 | 37 | 1 | 2 | 5.7 | 1061 | 495 |
| ariane136 | Ariane core, 136-cut SRAM configuration | ChiPBench | 161,661 | 136 | 1 | 14 | 4.0 | 1143 | 495 |
| black_parrot | BlackParrot multicore RISC-V | ChiPBench | 207,046 | 24 | 5 | 9 | 7.9 | 1065 | 1,198 |
| bp_be12 | BlackParrot back-end (execution) | ChiPBench | 38,115 | 12 | 5 | 5 | 5.4 | 704 | 3,029 |
| bp_fe38 | BlackParrot front-end (fetch) | ChiPBench | 25,727 | 38 | 7 | 7 | 5.4 | 720 | 2,511 |
| bp_multi57 | BlackParrot multicore top, 57-macro build | ChiPBench | 95,452 | 57 | 18 | 22 | 10.6 | 1016 | 1,453 |
| bp_multi_top | BlackParrot multicore top | ChiPBench | 120,741 | 26 | 6 | 10 | 7.5 | 960 | 1,453 |
| coyote | BSG Celerity Rocket node client | ORFS | 321,175 | 15 | 4 | 4 | 5.6 | 1068 | 784 |
| cva6 | OpenHW CVA6 RISC-V core | ORFS | 106,865 | 8 | 4 | 4 | 2.7 | 699 | 962 |
| dft68 | DFT/permutation datapath, distributed memories | ChiPBench | 59,002 | 112 | 4 | 64 | 1.0 | 1159 | 132 |
| mempool_group | PULP MemPool group (Snitch cores + TCDM) | ORFS | 143,052 | 20 | 2 | 2 | 6.0 | 820 | 1,273 |
| mor1kx | OpenRISC mor1kx core | ChiPBench | 68,254 | 78 | 7 | 20 | 9.0 | 594 | 576 |
| or1200 | OpenRISC OR1200 core | ChiPBench | 26,380 | 36 | 6 | 8 | 8.0 | 941 | 383 |
| riscv32i | RISC-V RV32I core (Oklahoma State) | ORFS | 10,251 | 4 | 1 | 1 | 2.0 | 285 | 135 |
| tinyRocket | Rocket tile, TinyConfig | ORFS | 27,315 | 2 | 1 | 2 | 1.2 | 338 | 269 |

42 tasks, so one gradient step consumes one task group. A candidate is one JSON response giving coordinates for every macro together with five flow-level levers: placement style, macro halo (1–15 $\mu$m), a corridor kept clear along pin-bearing die sides (0–150 $\mu$m), minimum macro spacing (0–20 $\mu$m), and an orientation policy. Each candidate is then executed in its own OpenROAD process, with all eight run in parallel: the legalizer resolves residual overlaps, and the fixed template runs standard-cell placement, CTS, and global and detailed routing with timing repair.

**Reward and cost.** The six reward components are WNS and log-scaled TNS, both normalized by the clock period and weighted 1.5; DRC violations as $-\log(1 + N_{\mathrm{DRC}})$ weighted 2.0; negated power and area weighted 0.5 each; and a legality term weighted 0.5 that combines the placed fraction with penalties for legalizer displacement and out-of-range levers. A candidate that fails to route is assigned, on every PPA component, a value one group range below its worst routed sibling, so it ranks last without being discarded. Cost is dominated by routing. Across 688 logged candidates over 86 steps, a candidate takes 42 minutes on average and 30 at the median, of which CTS and routing account for 31 minutes and detailed placement 4. Of those candidates 87% route; the remainder are incomplete placements (59), timeouts (19), routing failures (6), and parse or tool errors (8). We cap a candidate at three hours, having found that a six-hour cap let four stragglers consume roughly a day of wall time, and we checkpoint the adapter every five steps so a run can be resumed after a cap change. The adapter trains at learning rate $10^{-5}$ with no KL penalty, matching Stage-I.

### A.6 Ablation study results

**GDPO vs. GRPO.** Because macro placement is a multi-reward problem, we also implement GDPO (Liu et al., 2026), which normalizes each reward component within the rollout group before applying the weights instead of normalizing the weighted sum. Its premise is that per-component normalization preserves advantage resolution that sum-then-normalize discards. We train a matched pair over 500 steps, identical in reward set, weights, curriculum, task order, and seed, and differing only in the aggregation (Figure 6). The two runs are indistinguishable: total reward and every component track each other throughout, and the small end-of-run gaps point in opposite directions (overlap-free slightly favors GRPO, graded containment slightly favors GDPO), well within the fluctuation band of the curves themselves. We attribute this to reward density. GDPO helps most when the summed reward collapses to a few distinct values within a group, which is the sparse pass/fail regime; with four dense rewards in $[0, 1]$, two of eight candidates rarely score identically, so the weighted sum already orders the group and per-component normalization has nothing left to recover. We therefore keep vanilla GRPO, the simpler of the two, in the main results.

**Curriculum learning.** Without the schedule the policy settles into a worse optimum and buys non-overlap with row overflow: total reward plateaus near 4.0 and rows overrun the core 0.88 times per case, against 4.3 and 0.22 under the curriculum (Figure 7). Neither rollout budget nor aggregation rescues it: raising the group size to $K{=}16$, switching to GDPO, and adding dynamic weights with a diversity term all plateau at the same level. What the curriculum changes is the order in which the

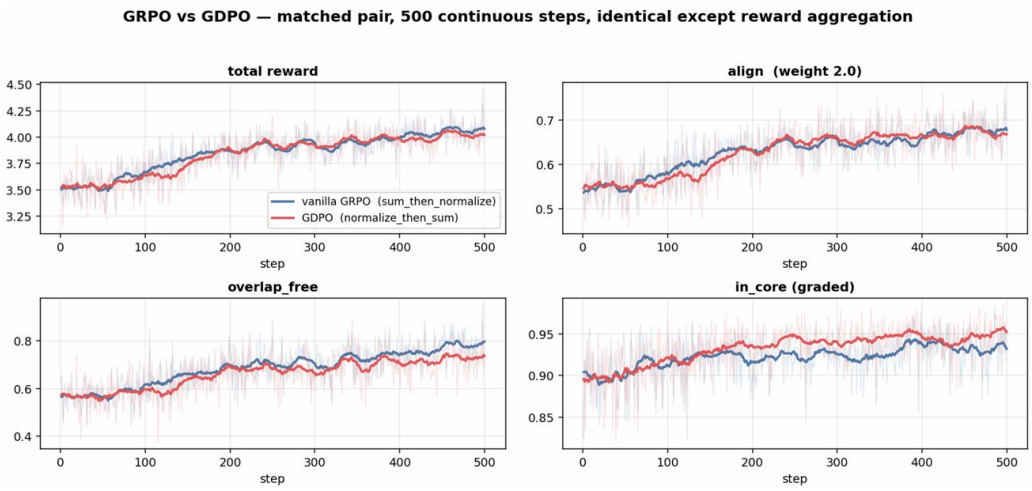


Figure 6: **Reward aggregation is not what matters here.** A matched pair of Stage-I RL runs over 500 steps, identical except that vanilla GRPO normalizes the weighted reward sum within each rollout group while GDPO normalizes each component first. Faint lines are per-step group means, bold lines a moving average.

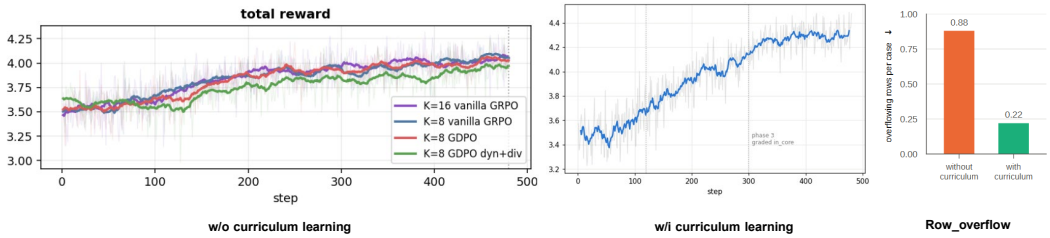


Figure 7: **The reward schedule, not the rollout budget or the aggregation, is what fixes row overflow.** *Left:* total reward for four flat-weight runs ($K$=8 or 16, vanilla GRPO or GDPO, with and without dynamic weights), all plateauing near 4.0. *Middle:* the staged curriculum instead reaches 4.3. *Right:* overflowing rows per case, lower is better.

objectives are optimized, legality first and alignment once constraints largely hold, which suppresses the row-overflow exploit without regressing the other subtask metrics.

**K candidates.** The group size $K$ sets how many candidates share a baseline. With the global batch held at 48 completions, it trades rollouts per prompt against prompts per step, so we compare $K$=8 (6 prompts $\times$ 8) against $K$=16 (3 prompts $\times$ 16) over 480 steps (Figure 8). The two stay within each other's noise band on total reward and on all three components, so the larger group buys nothing measurable while halving the prompts seen per step. We therefore use $K$=8 throughout, which covers twice as many prompts at equal cost.

**Trained scorer.**

Alignment is the only learned component of the reward. We score the training corpus with the frontier judge, transfer those scores into an 8B scorer, and serve that scorer in the RL loop; the ablation replaces it with the same backbone scoring zero-shot (Figure 9). The two arms' training curves are not comparable, because each is scored by its own judge, so we compare the resulting policies on PlaceReasoner-Layout-Bench under one common judge (Table 7). The learned scorer is the better reward signal without being the better teacher: it raises its own alignment reward faster and leaves the geometric metrics unchanged to within a point, yet ends six points lower on rubric alignment (0.61 against 0.67). We read this as over-optimization of the proxy, with the policy exploiting exactly where the learned scorer disagrees with the judge it was learned from, and it is a reminder that a cheap in-loop judge is worth only as much as its agreement with the standard it approximates.

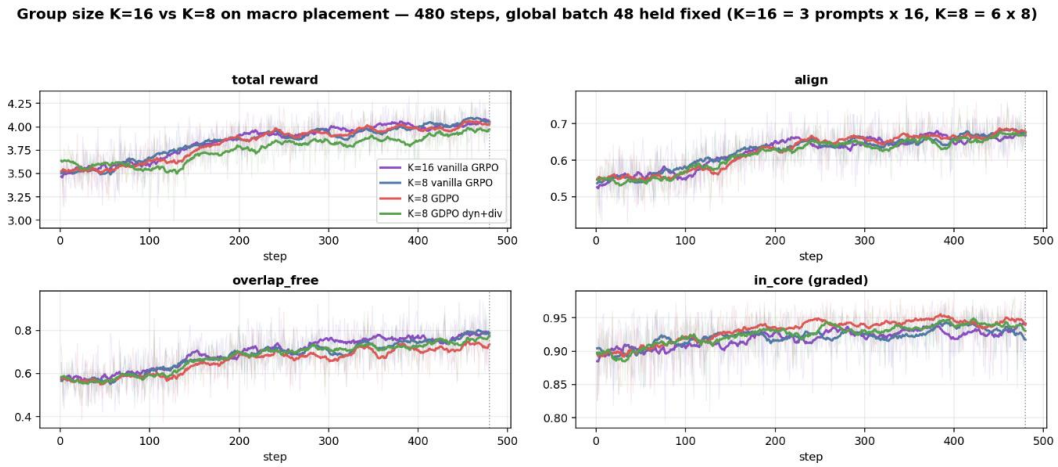


Figure 8: **A larger rollout group does not help.** Group size $K{=}16$ against $K{=}8$ at a fixed global batch of 48 completions, with $K{=}8$ also shown under GDPO and under GDPO with dynamic weights. Faint lines are per-step group means, bold lines a moving average.

Table 7: Alignment reward source, evaluated on PlaceReasoner-Layout-Bench under the frontier judge. Geometric metrics are unchanged; the learned scorer costs six points of rubric alignment.

| Alignment reward | Completeness | Overlap-free | In-core | Alignment |
|---|---|---|---|---|
| Base scorer | 0.98 | 0.91 | 0.93 | **0.67** |
| Learned scorer | 0.98 | **0.92** | **0.94** | 0.61 |

### A.7 Full Results on PlaceReasoner-Bench

Tables 8 and 9 give routed PPA, design-rule violations, and token usage for every method on the square and elongated tasks: four classical placers, PlaceReasoner-Beta, the untuned Qwen3-VL-8B backbone, and our policy after SFT, Stage-I RL, and Stage-II RL. Baseline rows are taken from the PlaceReasoner-Beta Li et al. (2026b) evaluation, which runs the same harness, netlists, floorplans, and P&R scripts that we use, so every row in these tables is directly comparable.

### A.8 Case study

We first follow two PlaceReasoner-Layout-Bench tasks through the training stages (Figure 10), then show every PlaceReasoner-Bench task as placed and routed by the Stage-II policy (Figures 11 and 12).

**PlaceReasoner-Layout-Bench across training stages.** The untuned backbone places no macros on either task. The SFT policy places macros in compact blocks, but covers only part of the macro set on the square task and does not reach a valid layout on either. Stage-I RL changes the arrangement: by step 300 the policy lays macros out in rows along the core edges, and on the elongated task it splits them between the two short edges, leaving the middle of the core free. These layouts still break a constraint (row overflow and macro overlap). By step 480 the square task is valid, with all macros packed into rows in one corner. The elongated task keeps the short-edge split, now without overlap, but its longest row runs past the core boundary, so the case remains a failure. The short-edge split is the same arrangement the Stage-II policy uses on the 2:1 PlaceReasoner-Bench tasks below.

**PlaceReasoner-Bench after Stage-II RL.** Figures 11 and 12 show all 16 PlaceReasoner-Bench tasks as laid out by the Stage-II policy: first the legalized macro placement, then the same placement after CTS and global and detailed routing.

**Macros go to the edges.** On most tasks the policy packs macros against one or two core edges and leaves a single contiguous region free. The macros form a band along one edge (ariane133 and

Table 8: Full post-route results on PlaceReasoner-Bench, square (1:1) tasks. Baseline rows are reproduced from the PlaceReasoner-Beta evaluation under the identical harness and P&R flow; PR-BETA is that harness driven by Claude Opus 4.8 and PR-PRO rows are the same harness driven by our 8B policy at successive training stages. "–" denotes an unavailable metric, and failure states indicate that detailed routing did not complete.

| Design | Method | Route | WNS (ns) | TNS (ns) | DRV | WL ($\mu$m) | Power (W) | Area ($\mu m^2$) |
|---|---|---|---|---|---|---|---|---|
| ariane133 | RTL-MP | Yes | -0.10 | -4.60 | 0 | 5233982 | 0.265 | 729834 |
| ariane133 | DREAMPlace | Fail: timeout | – | – | – | – | – | – |
| ariane133 | ChiPFormer | Yes | -0.04 | -4.61 | 0 | 6211024 | 0.304 | 737387 |
| ariane133 | SA | Yes | -76.52 | -579878 | 80223 | 7135797 | 0.286 | 718246 |
| ariane133 | PR-Beta | Yes | -0.02 | -0.35 | 0 | 5622394 | 0.258 | 728851 |
| ariane133 | Qwen3-VL-8B | Fail: DRT-0073 | – | – | – | – | – | – |
| ariane133 | PR-Pro-SFT | Yes | -0.06 | -3.58 | 0 | 6102605 | 0.266 | 733242 |
| ariane133 | PR-Pro-I | Yes | -0.06 | -0.84 | 0 | 6134664 | 0.294 | 729788 |
| ariane133 | PR-Pro-II | Yes | -0.01 | -0.21 | 0 | 6100372 | 0.284 | 728322 |
| ariane81 | RTL-MP | Yes | -0.14 | -8.43 | 0 | 5577716 | 0.184 | 694809 |
| ariane81 | DREAMPlace | Fail: timeout | – | – | – | – | – | – |
| ariane81 | ChiPFormer | Yes | -1.02 | -2103 | 0 | 6948313 | 0.177 | 700271 |
| ariane81 | SA | Fail: placed-only | – | – | – | – | – | – |
| ariane81 | PR-Beta | Yes | -0.04 | -1.20 | 0 | 5203922 | 0.171 | 692142 |
| ariane81 | Qwen3-VL-8B | Yes | -0.09 | -6.24 | 0 | 6631628 | 0.198 | 693184 |
| ariane81 | PR-Pro-SFT | Yes | -0.07 | -13.59 | 0 | 6946901 | 0.201 | 694171 |
| ariane81 | PR-Pro-I | Yes | -0.03 | -0.22 | 0 | 6159825 | 0.197 | 692675 |
| ariane81 | PR-Pro-II | Yes | -0.01 | -0.05 | 0 | 6441924 | 0.197 | 693138 |
| bp_be | RTL-MP | Yes | -0.28 | -23.54 | 0 | 2354232 | 0.123 | 240519 |
| bp_be | DREAMPlace | Yes | -0.47 | -29.81 | 63393 | 3814816 | – | 243381 |
| bp_be | ChiPFormer | Yes | -0.58 | -54 | 0 | 2929620 | 0.131 | 243411 |
| bp_be | SA | Yes | -0.24 | -21.26 | 37695 | 2411088 | 0.123 | 239997 |
| bp_be | PR-Beta | Yes | -0.13 | -9.42 | 0 | 2287187 | 0.118 | 238089 |
| bp_be | Qwen3-VL-8B | Yes | -0.29 | -28.59 | 0 | 2435295 | 0.107 | 238967 |
| bp_be | PR-Pro-SFT | Yes | -0.23 | -20.07 | 0 | 2606992 | 0.105 | 239488 |
| bp_be | PR-Pro-I | Yes | -0.19 | -18.49 | 0 | 2454375 | 0.106 | 239509 |
| bp_be | PR-Pro-II | Yes | -0.16 | -14.38 | 7 | 2526559 | 0.104 | 239494 |
| bp_fe | RTL-MP | Yes | -0.04 | -0.16 | 0 | 1607291 | 0.143 | 215331 |
| bp_fe | DREAMPlace | Yes | -0.37 | -41.92 | 65340 | 2743022 | – | 223308 |
| bp_fe | ChiPFormer | Yes | -0.46 | -35 | 0 | 1795024 | 0.158 | 223786 |
| bp_fe | SA | Yes | -0.19 | -1.06 | 0 | 1671812 | 0.145 | 217188 |
| bp_fe | PR-Beta | Yes | -0.03 | -0.07 | 0 | 1553678 | 0.141 | 213874 |
| bp_fe | Qwen3-VL-8B | Yes | -0.13 | -4.75 | 0 | 1550338 | 0.126 | 215049 |
| bp_fe | PR-Pro-SFT | Yes | -0.04 | -0.19 | 0 | 1557515 | 0.128 | 214973 |
| bp_fe | PR-Pro-I | Yes | -0.04 | -0.09 | 0 | 1565052 | 0.129 | 215248 |
| bp_fe | PR-Pro-II | Yes | -0.04 | -0.14 | 0 | 1532330 | 0.127 | 214990 |
| ethernet | RTL-MP | Yes | -0.31 | -1.46 | 14 | 730251 | 0.0142 | 110629 |
| ethernet | DREAMPlace | Yes | -0.30 | -1.38 | 30699 | 1496848 | – | 112452 |
| ethernet | ChiPFormer | Yes | -0.30 | -1.50 | 0 | 719208 | 0.014 | 110765 |
| ethernet | SA | Yes | -0.30 | -1.35 | 27609 | 761707 | 0.0142 | 110754 |
| ethernet | PR-Beta | Yes | -0.30 | -1.44 | 0 | 760262 | 0.0142 | 110597 |
| ethernet | Qwen3-VL-8B | Yes | -0.30 | -1.45 | 0 | 735008 | 0.0171 | 110435 |
| ethernet | PR-Pro-SFT | Yes | -0.30 | -1.34 | 0 | 692125 | 0.0169 | 110196 |
| ethernet | PR-Pro-I | Yes | -0.30 | -1.38 | 0 | 750573 | 0.0172 | 110454 |
| ethernet | PR-Pro-II | Yes | -0.28 | -1.30 | 4 | 723829 | 0.0171 | 110434 |
| swerv_wrapper | RTL-MP | Yes | -0.52 | -424.43 | 0 | 3757454 | 0.239 | 648507 |
| swerv_wrapper | DREAMPlace | Fail: timeout | – | – | – | – | – | – |
| swerv_wrapper | ChiPFormer | Yes | -1.55 | -1384 | 0 | 4593545 | 0.246 | 651840 |
| swerv_wrapper | SA | Yes | -0.51 | -421.84 | 76026 | 4238062 | 0.240 | 649332 |
| swerv_wrapper | PR-Beta | Yes | -0.42 | -409.13 | 0 | 3447721 | 0.243 | 647807 |
| swerv_wrapper | Qwen3-VL-8B | Yes | -0.60 | -617.00 | 0 | 4067535 | 0.249 | 646917 |
| swerv_wrapper | PR-Pro-SFT | Yes | -0.70 | -729.67 | 1 | 4060092 | 0.253 | 645750 |
| swerv_wrapper | PR-Pro-I | Yes | -0.58 | -484.92 | 0 | 4343919 | 0.250 | 647000 |
| swerv_wrapper | PR-Pro-II | Yes | -0.56 | -488.89 | 1 | 3836204 | 0.242 | 646136 |
| vga_lcd | RTL-MP | Yes | -0.78 | -498.02 | 45 | 2471807 | 0.182 | 650863 |
| vga_lcd | DREAMPlace | Yes | -0.85 | -146.70 | 62598 | 3804114 | – | 650838 |
| vga_lcd | ChiPFormer | Yes | -0.82 | -1180 | 0 | 2880834 | 0.198 | 655252 |
| vga_lcd | SA | Yes | -0.81 | -36.16 | 67293 | 2391472 | 0.184 | 649634 |
| vga_lcd | PR-Beta | Yes | -0.74 | -37.73 | 0 | 2342025 | 0.180 | 651477 |
| vga_lcd | Qwen3-VL-8B | Yes | -0.89 | -41.92 | 0 | 2449050 | 0.190 | 648133 |
| vga_lcd | PR-Pro-SFT | Yes | -0.78 | -33.09 | 0 | 2513284 | 0.176 | 648240 |
| vga_lcd | PR-Pro-I | Yes | -0.75 | -30.81 | 0 | 2672866 | 0.191 | 648505 |
| vga_lcd | PR-Pro-II | Yes | -0.75 | -31.88 | 0 | 2232123 | 0.187 | 647353 |
| VeriGPU | RTL-MP | Yes | -0.30 | -12.76 | 0 | 1753666 | 0.0931 | 274654 |
| VeriGPU | DREAMPlace | Yes | -0.26 | -13.21 | 51828 | 2322311 | – | 274662 |
| VeriGPU | ChiPFormer | Yes | -0.26 | -13 | 0 | 1807519 | 0.095 | 275147 |
| VeriGPU | SA | Yes | -0.32 | -13.55 | 61089 | 1751564 | 0.0943 | 275045 |
| VeriGPU | PR-Beta | Yes | -0.25 | -12.31 | 0 | 1697586 | 0.0911 | 274109 |
| VeriGPU | Qwen3-VL-8B | Yes | -0.25 | -17.43 | 0 | 1705359 | 0.128 | 274236 |
| VeriGPU | PR-Pro-SFT | Yes | -0.30 | -14.68 | 0 | 1710691 | 0.129 | 274565 |
| VeriGPU | PR-Pro-I | Yes | -0.25 | -13.34 | 0 | 1691729 | 0.128 | 274485 |
| VeriGPU | PR-Pro-II | Yes | -0.26 | -12.48 | 4 | 1701958 | 0.133 | 274213 |

Table 9: Full post-route results on PlaceReasoner-Bench, elongated (2:1) tasks. Baseline rows are reproduced from the PlaceReasoner-Beta evaluation under the identical harness and P&R flow; PR-BETA is that harness driven by Claude Opus 4.8 and PR-PRO rows are the same harness driven by our 8B policy at successive training stages. "–" denotes an unavailable metric, and failure states indicate that detailed routing did not complete.

| Design | Method | Route | WNS (ns) | TNS (ns) | DRV | WL ($\mu$m) | Power (W) | Area ($\mu m^2$) |
|---|---|---|---|---|---|---|---|---|
| ariane133 | RTL-MP | Yes | -0.12 | -25.29 | 0 | 6033112 | 0.276 | 731337 |
| ariane133 | DREAMPlace | Fail: timeout | – | – | – | – | – | – |
| ariane133 | ChiPFormer | Yes | -0.39 | -620 | 0 | 7079270 | 0.278 | 738872 |
| ariane133 | SA | Fail: GP1 | – | – | – | – | – | – |
| ariane133 | PR-Beta | Yes | -0.02 | -0.08 | 0 | 5690597 | 0.285 | 730228 |
| ariane133 | Qwen3-VL-8B | Fail: GRT-0116 | – | – | – | – | – | – |
| ariane133 | PR-Pro-SFT | Fail: route | – | – | – | – | – | – |
| ariane133 | PR-Pro-I | Yes | -0.12 | -11.33 | 1 | 6131396 | 0.293 | 729938 |
| ariane133 | PR-Pro-II | Yes | -0.06 | -2.52 | 2 | 6208345 | 0.292 | 729633 |
| ariane81 | RTL-MP | Yes | -0.10 | -35.10 | 0 | 6180469 | 0.179 | 696239 |
| ariane81 | DREAMPlace | Fail: timeout | – | – | – | – | – | – |
| ariane81 | ChiPFormer | Yes | -1.82 | -5617 | 0 | 9503525 | 0.181 | 727526 |
| ariane81 | SA | Fail: capped | – | – | – | – | – | – |
| ariane81 | PR-Beta | Yes | -0.02 | -0.15 | 0 | 5798709 | 0.194 | 698754 |
| ariane81 | Qwen3-VL-8B | Yes | -0.40 | -310.10 | 28 | 7017977 | 0.202 | 694691 |
| ariane81 | PR-Pro-SFT | Yes | -0.50 | -356.16 | 0 | 6554251 | 0.199 | 694266 |
| ariane81 | PR-Pro-I | Yes | -0.16 | -83.17 | 5 | 6970316 | 0.198 | 695819 |
| ariane81 | PR-Pro-II | Yes | -0.02 | -0.42 | 0 | 6655187 | 0.197 | 692407 |
| bp_be | RTL-MP | Yes | -0.37 | -38.96 | 0 | 2652279 | 0.123 | 240762 |
| bp_be | DREAMPlace | Fail: timeout | – | – | – | – | – | – |
| bp_be | ChiPFormer | Fail: segfault | – | – | – | – | – | – |
| bp_be | SA | Yes | -0.36 | -36.62 | 34190 | 2749978 | 0.124 | 241710 |
| bp_be | PR-Beta | Yes | -0.22 | -18.06 | 1 | 2873409 | 0.109 | 240882 |
| bp_be | Qwen3-VL-8B | Yes | -0.39 | -31.03 | 0 | 2647452 | 0.107 | 239686 |
| bp_be | PR-Pro-SFT | Yes | -0.33 | -31.56 | 0 | 2810438 | 0.108 | 239999 |
| bp_be | PR-Pro-I | Yes | -0.33 | -32.06 | 0 | 2700561 | 0.106 | 239595 |
| bp_be | PR-Pro-II | Yes | -0.23 | -21.84 | 0 | 2540448 | 0.105 | 239185 |
| bp_fe | RTL-MP | Yes | -0.18 | -6.12 | 0 | 1609885 | 0.143 | 215198 |
| bp_fe | DREAMPlace | Fail: timeout | – | – | – | – | – | – |
| bp_fe | ChiPFormer | Yes | -0.05 | -0.21 | 5 | 2286401 | 0.149 | 216837 |
| bp_fe | SA | Yes | -0.17 | -6.55 | 20479 | 1662195 | – | 218725 |
| bp_fe | PR-Beta | Yes | -0.11 | -0.23 | 0 | 1638383 | 0.131 | 217638 |
| bp_fe | Qwen3-VL-8B | Yes | -0.09 | -2.43 | 0 | 1719403 | 0.135 | 219808 |
| bp_fe | PR-Pro-SFT | Yes | -0.06 | -0.57 | 0 | 1700999 | 0.131 | 218035 |
| bp_fe | PR-Pro-I | Yes | -0.17 | -7.70 | 0 | 1653510 | 0.131 | 216427 |
| bp_fe | PR-Pro-II | Yes | -0.07 | -0.28 | 0 | 1610257 | 0.128 | 214585 |
| ethernet | RTL-MP | Yes | -0.33 | -1.62 | 14 | 812042 | 0.014 | 110330 |
| ethernet | DREAMPlace | Yes | -0.30 | -1.38 | 34020 | 1383506 | – | 112103 |
| ethernet | ChiPFormer | Yes | -0.33 | -1.60 | 0 | 797665 | 0.014 | 110403 |
| ethernet | SA | Yes | -0.27 | -1.31 | 10394 | 867444 | – | 110537 |
| ethernet | PR-Beta | Yes | -0.28 | -1.34 | 0 | 825144 | 0.0139 | 110333 |
| ethernet | Qwen3-VL-8B | Yes | -0.29 | -1.43 | 0 | 761143 | 0.017 | 110252 |
| ethernet | PR-Pro-SFT | Yes | -0.30 | -1.47 | 0 | 845543 | 0.0172 | 110111 |
| ethernet | PR-Pro-I | Yes | -0.29 | -1.40 | 393 | 773340 | 0.0171 | 110260 |
| ethernet | PR-Pro-II | Yes | -0.26 | -1.28 | 2 | 781740 | 0.0169 | 110202 |
| swerv_wrapper | RTL-MP | Yes | -0.64 | -528.28 | 1421 | 3779122 | 0.235 | 647771 |
| swerv_wrapper | DREAMPlace | Fail: timeout | – | – | – | – | – | – |
| swerv_wrapper | ChiPFormer | Yes | -1.40 | -1616 | 1456 | 6958764 | 0.260 | 686355 |
| swerv_wrapper | SA | Fail: placed-only | – | – | – | – | – | – |
| swerv_wrapper | PR-Beta | Yes | -0.58 | -476.28 | 2 | 3925621 | 0.242 | 648145 |
| swerv_wrapper | Qwen3-VL-8B | Yes | -0.67 | -755.17 | 0 | 4471772 | 0.245 | 646254 |
| swerv_wrapper | PR-Pro-SFT | Yes | -0.61 | -529.81 | 1 | 4485502 | 0.245 | 647056 |
| swerv_wrapper | PR-Pro-I | Yes | -0.62 | -666.94 | 26 | 3988478 | 0.241 | 646140 |
| swerv_wrapper | PR-Pro-II | Yes | -0.51 | -466.50 | 8 | 3856127 | 0.241 | 645894 |
| vga_lcd | RTL-MP | Yes | -0.72 | -29.70 | 171 | 2354087 | 0.180 | 647137 |
| vga_lcd | DREAMPlace | Yes | -0.77 | -40.80 | 52227 | 3089394 | – | 649017 |
| vga_lcd | ChiPFormer | Yes | -0.88 | -2317 | 33 | 3688031 | 0.198 | 654877 |
| vga_lcd | SA | Fail: GP1 | – | – | – | – | – | – |
| vga_lcd | PR-Beta | Yes | -0.76 | -33.13 | 1 | 2547982 | 0.183 | 649875 |
| vga_lcd | Qwen3-VL-8B | Yes | -0.76 | -1154.16 | 25 | 2605124 | 0.187 | 648128 |
| vga_lcd | PR-Pro-SFT | Yes | -0.85 | -164.84 | 0 | 2506564 | 0.181 | 647357 |
| vga_lcd | PR-Pro-I | Yes | -0.91 | -50.25 | 21 | 2875520 | 0.193 | 648135 |
| vga_lcd | PR-Pro-II | Yes | -0.68 | -27.23 | 0 | 2464813 | 0.181 | 646603 |
| VeriGPU | RTL-MP | Yes | -0.30 | -16.98 | 22 | 2009432 | 0.091 | 274307 |
| VeriGPU | DREAMPlace | Yes | -0.49 | -19.39 | 62832 | 2727467 | – | 274799 |
| VeriGPU | ChiPFormer | Yes | -0.28 | -17 | 0 | 2108273 | 0.093 | 275311 |
| VeriGPU | SA | Yes | -0.28 | -17.51 | 45375 | 1996265 | – | 275007 |
| VeriGPU | PR-Beta | Yes | -0.28 | -13.08 | 0 | 1896523 | 0.101 | 274461 |
| VeriGPU | Qwen3-VL-8B | Yes | -0.45 | -54.06 | 0 | 1906068 | 0.129 | 274392 |
| VeriGPU | PR-Pro-SFT | Yes | -0.28 | -15.40 | 0 | 1994577 | 0.128 | 274655 |
| VeriGPU | PR-Pro-I | Yes | -0.42 | -223.85 | 0 | 2031801 | 0.128 | 274802 |
| VeriGPU | PR-Pro-II | Yes | -0.28 | -13.92 | 1 | 1903553 | 0.128 | 274565 |

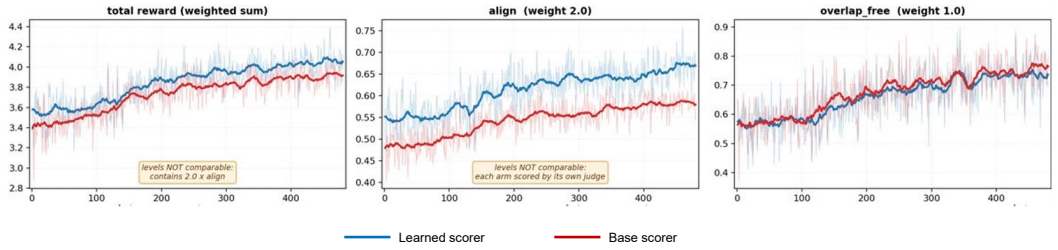


Figure 9: **Training against the learned scorer.** Stage-I RL with the learned scorer against the same backbone scoring zero-shot. Reward levels are not comparable across arms: the total reward contains $2.0\times$ the alignment term, and each arm is scored by its own judge, which is why the two policies are compared on PlaceReasoner-Layout-Bench in Table 7.

ethernet at 1:1), a top and a bottom band (bp_be at 1:1, VeriGPU), or two side columns (bp_fe at 1:1). The routed views show the standard-cell logic and its wiring filling that free region, so the logic that talks to the macros stays in one block instead of being split by narrow channels between macros.

**The layout follows the aspect ratio.** The policy does not reuse the square layout on the elongated core. On the 2:1 tasks of ethernet, VeriGPU, vga_lcd, and swerv_wrapper it moves macros to the two short edges and leaves the free region in the middle of the long axis, which keeps the free region compact.

**Harder cases.** On ariane81 and on vga_lcd at 1:1 the macros are many and small relative to the core, and the policy spreads them as an array over much of the core. The free region is then smallest, and the routed views are dense across nearly the whole die, since wires have to run over and between the macros.

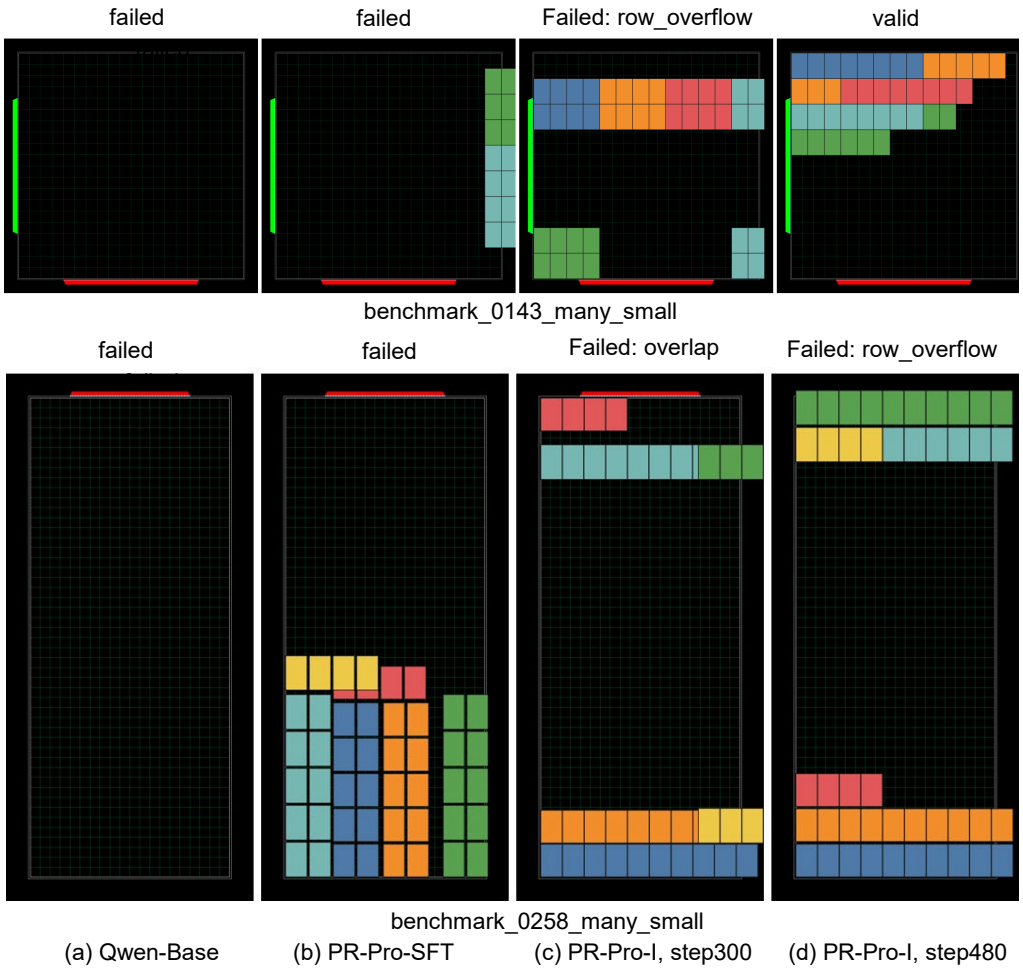


Figure 10: **Placements on two PlaceReasoner-Layout-Bench tasks across training stages.** Top: a square core; bottom: an elongated core; both use the many_small macro profile. Columns show (a) the untuned backbone, (b) the SFT policy, and the Stage-I RL policy at (c) step 300 and (d) step 480. The title of each panel gives the result of the legality check.

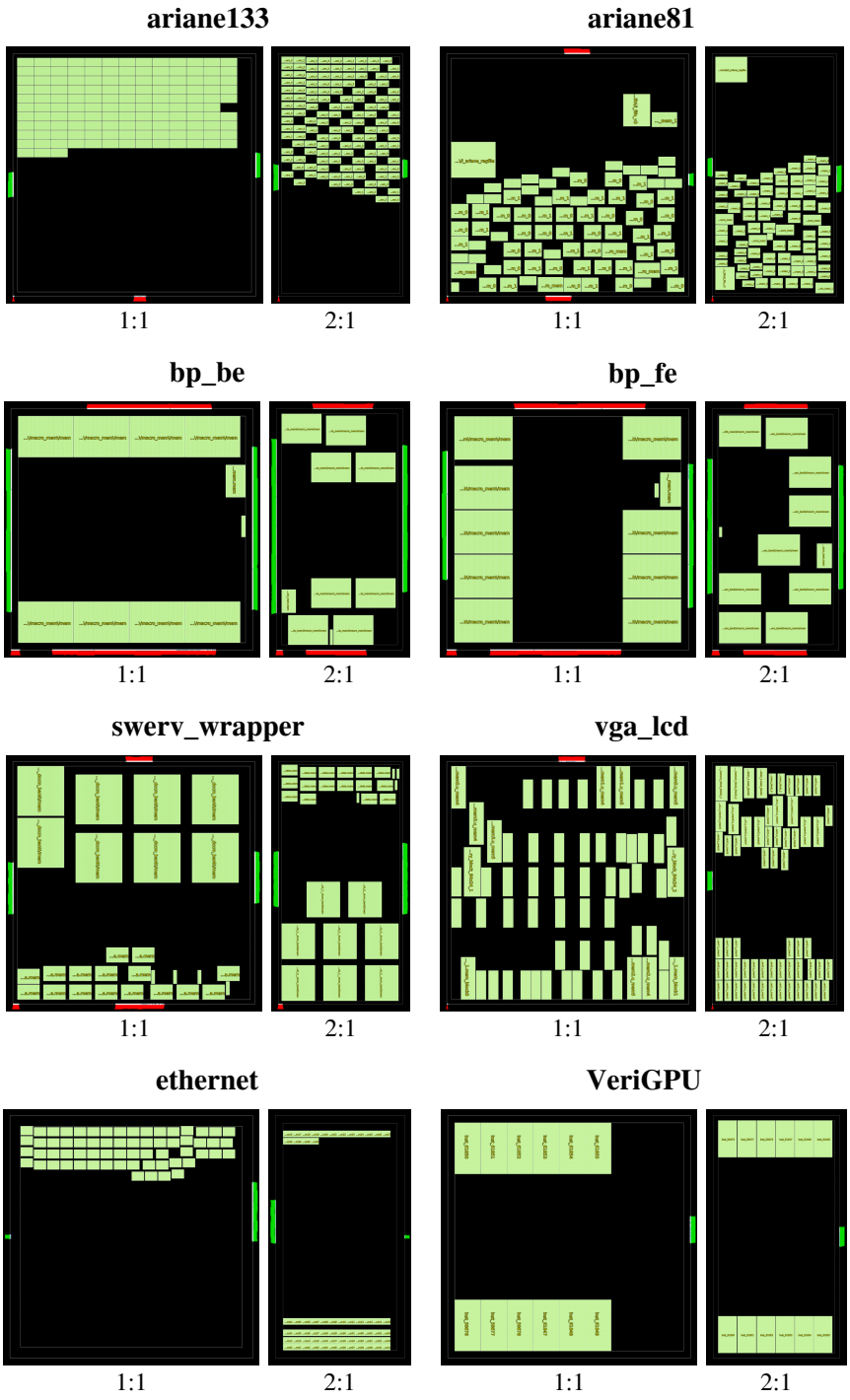


Figure 11: **Macro placements from the Stage-II policy on PlaceReasoner-Bench.** Each design is shown at the 1:1 (left) and 2:1 (right) core aspect ratio. Light-green rectangles are macros; the markers outside the die are I/O pins.

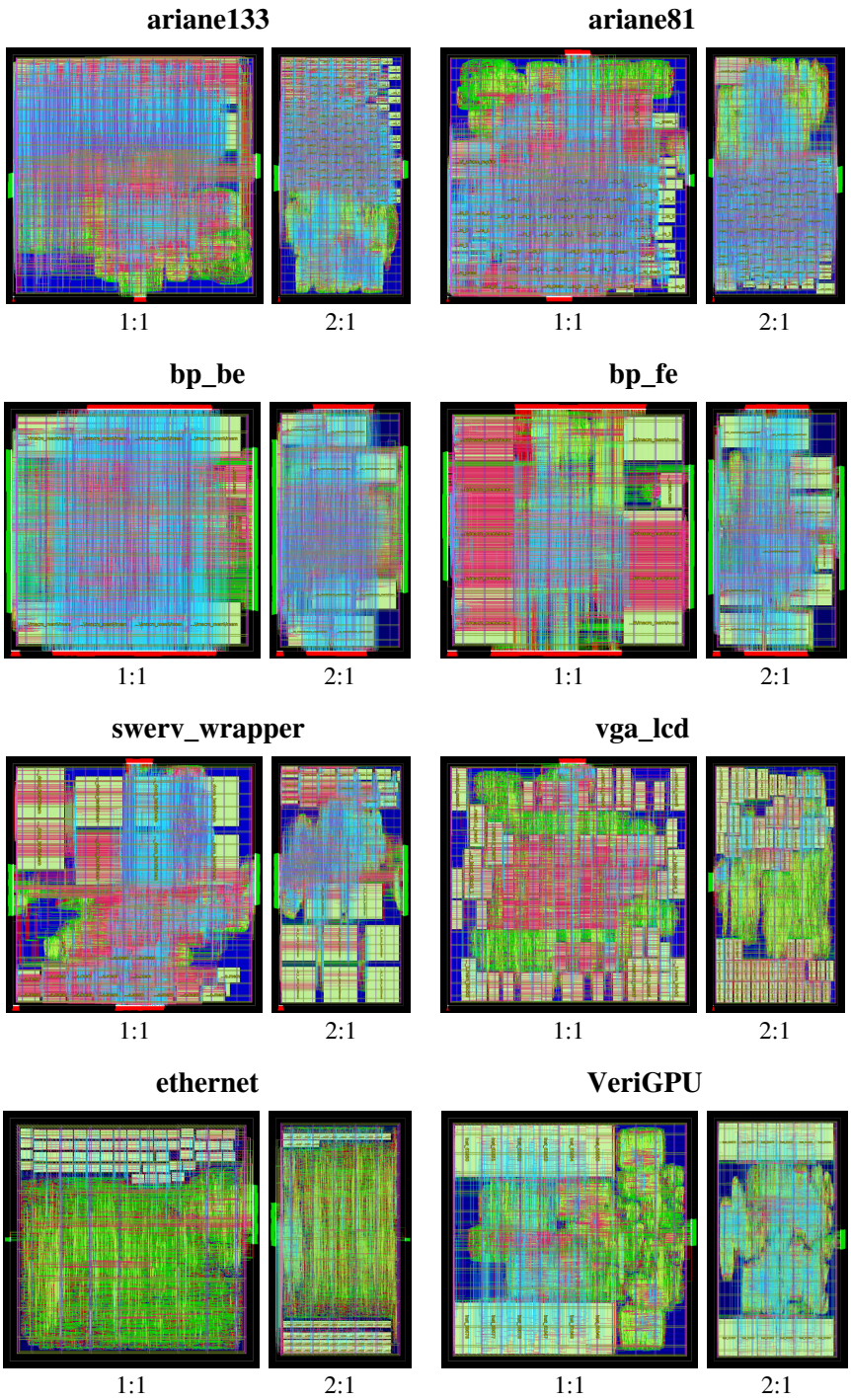


Figure 12: **Routed layouts of the placements in Figure 11.** The same 16 tasks after CTS and global and detailed routing, with the same arrangement as Figure 11; macros lie beneath the routed wires.