# BRIDGING THE LAST MILE OF CIRCUIT DESIGN: POSTEDA-BENCH, A HIERARCHICAL BENCHMARK FOR PPA CONVERGENCE AND DRC FIXING

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

## ABSTRACT

LLM-based agents increasingly address the "last mile" of Electronic Design Automation (EDA): repairing sign-off Design Rule Check (DRC) violations and converging Power–Performance–Area (PPA) targets. Existing EDA-LLM benchmarks omit DRC fixing and use flat hierarchies tied to single toolchains. We introduce POSTEDA-BENCH, a hierarchical, machine-checkable benchmark of 145 tasks spanning **DRC-Bench** (**DRC-Essential** and **DRC-Reasoning**) and **PPA-Bench** (**PPA-Mono** and **PPA-Multi**). Across nine commercial and open-source LLMs under multiple agent scaffolds, agents handle synthetic DRC-Essential and single-objective PPA-Mono reasonably well but degrade sharply on practical DRC-Reasoning (best SR 54.67%) and PPA-Multi (best SR 20.00%). Vision improves pooled DRC success rates, while trade-off reasoning—rather than knob knowledge—remains the main PPA-Multi bottleneck. The anonymous repository is available at `https://anonymous.4open.science/r/posteda-bench-anon-12F4/README.md`.

## 1 INTRODUCTION

Large language model (LLM)–based agents have shown strong performance in code generation (Hong et al., 2024; Zhang et al., 2024; Chen et al., 2024), robotics and embodied AI (Yao et al., 2023b; Driess et al., 2023), and mathematical reasoning (Schick et al., 2023; Gao et al., 2023). These advances motivate LLM-driven power–performance–area (PPA) convergence in electronic design automation (EDA) flows (Wu et al., 2024; 2025; Wang et al., 2025; Lu et al., 2025; Jiang et al., 2024; Fu et al., 2023; Liu et al., 2024; Ghose et al., 2025; Lu et al., 2025), where engineers spend substantial time iterating after tool runs to meet PPA targets.

Despite this progress, existing EDA-agent benchmarks still fall short of what is needed to evaluate practical post-EDA closure. As summarized in Table 1, most benchmarks either focus on NLP-to-script generation rather than target-driven PPA optimization, contain only a small number of optimization tasks, use flat or coarse task hierarchies, or are tied to a single toolchain, limiting both diagnostic resolution and generality. More importantly, none of them evaluates design rule check (DRC) fixing, although residual sign-off DRC violations remain a time-critical "last-mile" workload even after modern routing. These gaps make a unified and realistic benchmark for post-EDA PPA optimization and DRC repair both urgent and necessary.

To fill this gap, we introduce **POSTEDA-BENCH**, a hierarchical benchmark evaluating LLM-based agents on post-EDA closure: 145 tasks across two dimensions, **DRC-Bench** and **PPA-Bench**, evaluating whether an agent can (i) repair sign-off DRC violations and (ii) drive designs to specified PPA targets, supported by open-source (OPENROAD) and commercial (DC+INNOVUS) toolchains with machine-checkable evaluation. **DRC-Bench:** splits into DRC-Essential, three levels probing rule knowledge, contextual robustness, and cascaded multi-violation fixing on synthetic cases, and DRC-Reasoning, three levels of practical residual post-flow violations that demand geometric reasoning, stratified by minimum edit steps. **PPA-Bench:** splits into PPA-Mono (Performance, Power, Area), with single- and multi-knob perturbation levels plus a Performance re-architecture level, and PPA-Multi, with 2D and 3D Pareto-targeted tasks testing trade-off reasoning under constraint floors.



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.

Table 1: **Comparison with related benchmarks.**

| Benchmark | #Tasks | Open -sourced | Objective | Hierarchy | EDA Toolchains | DRC Fix |
|---|---|---|---|---|---|---|
| ChatEDA-bench (Wu et al., 2024) | 50 | ✓ | NLP→scripts | Simple–Complex | OpenROAD | ✗ |
| GPT4AIGChip (Fu et al., 2023) | 6 | ✗ | Codegen&PPA Opt. | Flat | Vivado HLS tools | ✗ |
| Chipnemo (Liu et al., 2024) | 959 | ✗ | NLP→scripts | Easy–Medium–Hard | n/a | ✗ |
| IICPilot (Jiang et al., 2024) | 4 | ✗ | PPA Opt. | Flat | OpenROAD | ✗ |
| iEDA-bench (Wu et al., 2025) | 50 | ✗ | NLP→scripts | Simple–Complex | OpenROAD | ✗ |
| MCP4EDA (Wang et al., 2025) | 10 | ✓ | NLP→scripts | Flat | OpenROAD | ✗ |
| ORFS-agent (Ghose et al., 2025) | 6 | ✓ | PPA Opt. | Flat | OpenROAD | ✗ |
| AutoEDA (Lu et al., 2025) | 100 | ✗ | NLP→scripts | Different–stage in EDA | DC&Innovus | ✗ |
| **POSTEDA-BENCH** | 145 | ✓ | PPA Opt. | Multi-dimension& Multi-level | OpenROAD& DC&Innovus | ✓ |

†: NLP→scripts: generates EDA scripts from natural language; it runs tools but doesn't optimize PPA.
§: PPA Opt. denotes PPA Optimization. ⋆: OpenROAD is open-source, whereas DC and Innovus are commercial tools.
‡: DC and Innovus are only used to generate part of DRC-Bench tasks in POSTEDA-BENCH.

Using POSTEDA-BENCH, we evaluate commercial and open-source models, frontier and small backbones, and text-only versus vision–language variants under several agent scaffolds (ReAct, Proposer–Critic, ORFS-Agent, Reflexion and ToT). **Our key findings are as follows:**

- ▷ **Synthetic vs. practical asymmetry.** Agents perform reasonably well on synthetic DRC-Essential (best SR $95.00\%$) and single-objective PPA-Mono (best SR $64.56\%$), but degrade sharply on the more practical DRC-Reasoning (best SR $54.67\%$) and PPA-Multi (best SR $20.00\%$), where post-flow geometric reasoning and multi-objective trade-offs dominate.
- ▷ **Vision augmentation enhances DRC-Bench.** Adding a layout-image channel maintains or improves SR/VRR, with positive pooled SR effects on DRC-Essential and DRC-Reasoning. Individual gains are not uniformly significant; paired bootstrap CIs appear in Section H.
- ▷ **Trade-off reasoning is the PPA-Multi bottleneck.** Several backbones produce *negative* NIS on PPA-Multi by regressing constraint metrics while chasing the violated objective, indicating they greedily optimize one PPA dimension rather than balancing competing targets.

## 2 RELATED WORK

**DRC fixing.** Design rule checking verifies layout manufacturability, yet residual sign-off errors and cascaded fixes often remain after automated repair. This motivates evaluating agents on rule understanding, layout inspection, geometric editing, and iterative DRC checking; additional background is in Section A. **PPA optimization.** Power, performance, and area are coupled objectives, and automated flows often miss tight targets despite many configuration, constraint, and design-level knobs. PPA-Bench targets this last-mile loop by testing whether agents can use tool feedback for single-objective improvement and multi-objective trade-offs; extended context is in Section A.

## 3 POSTEDA-BENCH

**Preliminary Source Design Collection.** As shown on the left of Figure 2, POSTEDA-BENCH starts from three public RTL sources: RTLLM v2 (Lu et al., 2024) (50 designs), VERILOGEVAL-Human (Liu et al., 2023) (156 problems), and OPENCORES (OpenCores.org, 2026) (1,283 listed IP projects), covering arithmetic circuits, controllers, and community IP blocks. We map retained designs to ASAP7 (Clark et al., 2016) and keep only candidates satisfying four auditable criteria: (i) **parse and synthesis validity**, i.e., a well-defined top module, valid syntax, and successful gate-level synthesis; (ii) **non-trivial scale**, with more than 300 instantiated standard cells; (iii) **cross-source deduplication** by top-module names, port signatures, and post-synthesis structural similarity, merging designs with more than 90% gate-level overlap; and (iv) **release and audit eligibility**, requiring documented provenance, license, selected top module, and derived artifacts. Aggregate statistics are reported in Table 2; Section B details deduplication, and the release package.

**Contamination control.** We mitigate contamination through task-specific geometric edits and PPA targets verified by fresh tool runs; task design and verification are detailed in Section B. **Reproducibility.** Most tasks are constructed manually by the authors, with scripts assisting artifact generation and parameter sweeps. Both suites provide fixed initial artifacts, prompts, metadata, pinned tool setup, and evaluation drivers so that evaluation can be repeated without recreating the

Table 2: **Statistics of POSTEDA-BENCH**

| Statistic | Number |
|---|---|
| Tasks | 145 |
| Std. cells per source design | min 301; avg 16,505; max 95,061 |
| Covered layers in DRC-Bench | N(P)Well – M9 |
| DRC-Essential rules (before pruning) | 168 |
| Human fix steps (DRC-Reasoning) | min 1; max 8 |
| DRC runtime† | min 10 s; max 3 h |
| Needed Tunable parameters§ | min 1; max 7 |
| Mono-Perf tasks requiring re-architecture⋆ | 4 |

†: DRC runtime is the wall-clock time for one signoff DRC check; it reflects design-scale diversity and is not agent or human repair time.
§: Needed tunable parameters are minimum critical knobs; agents may edit all OpenROAD parameters.
⋆: Mono-Perf re-architecture tasks require RTL/micro-architectural changes beyond parameter tuning.

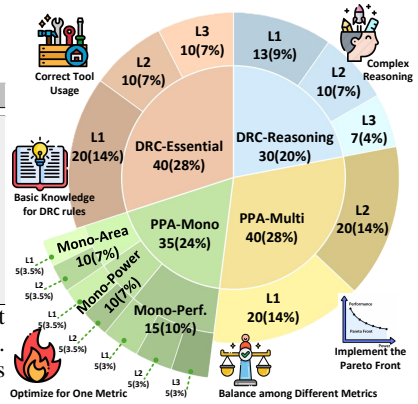



Figure 1: **Overview of POSTEDA-BENCH composition.**

authors' manual decisions. All released tasks can be scored with open-source tools. Construction provenance, release contents, and versioning are documented in Sections B to D.

### 3.1 DRC-BENCH

DRC-Bench evaluates last-mile sign-off DRC repair, a workload absent from existing EDA-LLM benchmarks (Table 1), through two branches: **DRC-Essential**, which isolates rule knowledge, layout-tool use, context robustness, and cascaded fixing on synthetic cases, and **DRC-Reasoning**, which targets residual post-flow violations requiring geometric reasoning.

#### 3.1.1 TASK SPECIFICATIONS

**Input.** Each task provides two artifacts: the target `.gds` layout and a natural-language prompt (Figure 3a). The prompt includes the initial sign-off DRC violations, with each entry specifying the rule identifier (e.g., `V0.M1.AUX.3`), a short rule description, the geometry type (`edge-pair`, `polygon`, or `edge`), and layout coordinates. **Metrics.** A run is successful when the final layout is DRC-clean under sign-off DRC, and we report the **Success Rate (SR)**, computed as the mean of the success indicator over 5 independent runs per task. To capture partial progress, we additionally report the **Violation Reduction Rate (VRR)**:

$$\mathrm{VRR} = \frac{\max(0,\, E_{\text{initial}} - E_{\text{final}})}{E_{\text{initial}}} \times 100\%, \tag{1}$$

where $E_{\text{initial}}$ and $E_{\text{final}}$ denote the total violation count before and after the agent's edits. The choice of sign-off tool and DRC deck is documented in Section C. A post-hoc LVS audit is in Section C.7.
**Tools.** Agents can query layout context textually or visually, edit GDS geometry and placement, and rerun sign-off DRC through a compact layout-viewer-style tool set; full schemas are in Section C.3.

#### 3.1.2 DRC-ESSENTIAL

DRC-Essential has three levels (light-apricot block in Figure 2), each isolating a primitive competency. **L1 (Atomic rule understanding).** L1 uses atomic single-rule violations with only the required shapes. Cases are hand-drawn from the ASAP7 design-rule manual or generated by per-task scripts (`create_errors.py`; Section C.4). We group *topologically equivalent* rules—those differing only in layer assignment—into one representative per class, balanced to one case per layer. **L2 (Contextual comprehensiveness).** L2 embeds one L1-style violation in realistic layout context, forcing agents to filter relevant geometry from clutter. Construction: *(i)* run a full EDA flow—OPENROAD-FLOW-SCRIPTS or SYNOPSYS DC +CADENCE INNOVUS—on a source-pool design; *(ii)* have an engineer repair residuals to a DRC-clean reference; *(iii)* inject one L1-style violation whose fix touches only the offending shapes. L2 thus isolates context filtering without adding geometric reasoning. **L3 (Sequential fixing).** L3 tests whether agents can resolve interacting violation chains. For each DRC-clean design we inject 5–15 L1-style violations across rule families,

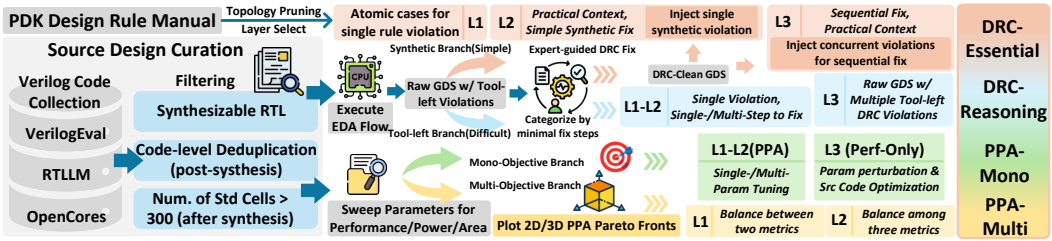



Figure 2: **Overview of POSTEDA-BENCH construction process.**

mainly width/via (e.g., `M2.W.1`, `M3.W.1`, etc.). Sites are co-located so edits interact; each violation is single-step in isolation, but agents must order edits and re-query DRC since fixes can trigger or eliminate nearby ones.

### 3.1.3 DRC-REASONING

**DRC-Reasoning targets practical residual violations that remain after full EDA flows and built-in repair** (blue block in Figure 2; Section C.5 quantifies the residual gap for Innovus). These cases are harder than DRC-Essential because they require non-trivial geometric reasoning. We partition them into three levels by the *minimum number of agent editing steps* required (one step = one `add_shape`/`change_shape`/`move_cell` call; inspection not counted). Labels come from an experienced physical-design engineer following the protocol in Section C.6, with a delayed self-consistency pass. Two physical-design engineers completed cross-verification of the labels under this protocol. **L1–L2 (Single- vs. multi-step fixes).** We sweep OPENROAD and DC+INNOVUS for post-flow residual violations across open-source and industrial settings. A violation is **L1** if its repair takes one editing step, **L2** if multiple (no upper cap). When unrelated violations co-exist, we isolate the target by cropping or temporarily fixing surroundings, so L1/L2 cleanly probe single-violation reasoning. **L3 (Multi-violation debugging).** L3 reproduces the practical multi-violation scenario: each task is a raw post-flow GDS with several real violations that the agent must iteratively debug and resolve. L3 designs may overlap with those seeding L1/L2 (different post-flow configs of the same source), but the per-task violation set is disjoint by construction.

## 3.2 PPA-BENCH

Beyond DRC, post-tool closure often fails because designs miss PPA (power, performance, and area) constraints. PPA-Bench contains two subsets: **PPA-Mono** optimizes one metric (performance, power, or area), while **PPA-Multi** evaluates trade-offs among multiple metrics.

### 3.2.1 TASK SPECIFICATIONS

**Inputs.** Each task is a self-contained OpenROAD project with Verilog sources (`src/`), configs (`config.mk`, `constraints.sdc`), and previous-run artifacts (`logs/`, `reports/`, `objects/`, `results/`). The prompt summarizes current PPA, target thresholds, constraint floors, and the `FlowVariables.md` knob list (Figure 3b); ground-truth metadata is access-controlled in Section D.2. **Metrics.** A run is successful when every target metric is met and the modified RTL passes the task's applicable testbench-based functional checks (Section D.5); **Success Rate (SR)** is the mean success fraction over 5 temperature-0 runs per task. We also report the **Nor-**

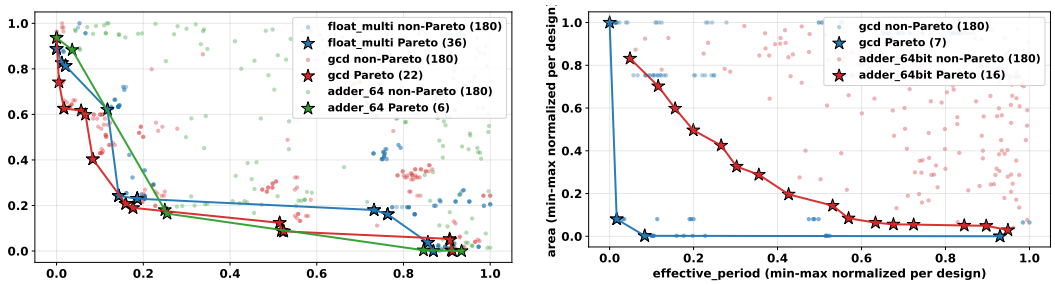
```
the input gds file is ./top_cell.gds
Top Cell: aes_cipher_top
=========================================
--- V0.M1.AUX.3  |  cell: aes_cipher_top
[0] {'category': 'V0.M1.AUX.3',
'description': V0 must exactly be the
same width as M1 along the Direction ...
```

(a) Sample prompt in DRC-Bench.

```
The design has completed the OpenROAD
flow, but effective period is not met.
Current effective_period: 212.2 ps
Target effective_period: <= 210.0 ps
Analyze any available files and optimize
design configuration or timing
constraints in `conf/.mk`
```

(b) Sample prompt in PPA-Bench.

Figure 3: **Example prompts used to drive the agent in DRC-Bench and PPA-Bench.**




(a) Effective period vs. total power.

(b) Effective period vs. die area.

Figure 4: **Two-objective Pareto fronts for representative PPA-Multi designs.** Gray dots denote non-Pareto solutions; blue stars indicate Pareto-optimal points. The combined panels summarize representative period–power and period–area trade-off fronts.

**malized Improvement Score (NIS)** for partial progress. For each constrained metric $i \in \mathcal{M}$, let $\mathcal{A} = \{i : M_i^{\text{init}} > M_i^{\text{tgt}}\}$ be the violated metrics to improve and $\mathcal{B} = \{i : M_i^{\text{init}} \le M_i^{\text{tgt}}\}$ be constraints that must not regress; the per-metric score is

$$s_i = \begin{cases} \min\big(1, r_i\big), & i \in \mathcal{A}, \\ 1, & i \in \mathcal{B} \text{ and } M_i^{\text{fin}} \le M_i^{\text{tgt}}, \\ 1 - r_i, & i \in \mathcal{B} \text{ and } M_i^{\text{fin}} > M_i^{\text{tgt}}, \end{cases} \quad \text{where } r_i = \frac{M_i^{\text{fin}} - M_i^{\text{init}}}{M_i^{\text{tgt}} - M_i^{\text{init}}}. \tag{2}$$

The aggregate score is then

$$\text{NIS} = \begin{cases} 0, & \mathcal{A} \neq \emptyset \text{ and } M_i^{\text{fin}} \ge M_i^{\text{init}}\ \forall\, i \in \mathcal{A}, \\ \dfrac{1}{|\mathcal{M}|} \displaystyle\sum_{i \in \mathcal{M}} s_i, & \text{otherwise.} \end{cases} \tag{3}$$

All metrics are lower-is-better: $r_i = 1$ means the target is reached, $r_i \in (0, 1)$ means partial progress, and $r_i < 0$ means regression. For single-objective tasks ($|\mathcal{M}| = 1$) we clip $s_1$ to $[0, 1]$; for multi-objective tasks, negative scores penalize regression, and Eq. 3's zero-floor rule prevents trade-off exploits that improve only constraint metrics. Per-task scores are averaged within each class. **Run protocol.** Each run is bounded by a tool-call iteration cap, treated as an evaluation hyperparameter and varied in the iteration ablation. **Tools.** Agents inspect files, edit configuration/constraints/RTL, run OpenROAD in an isolated container, and parse PPA reports through four sandboxed tools; full schemas and locked variables are in Section D.4.

### 3.2.2 PPA-Mono

PPA-Mono evaluates single-objective optimization across **Mono-Performance**, **Mono-Power**, and **Mono-Area** (green block in Figure 2), with no constraints on the other two metrics so each task isolates one objective. **Construction.** For each source design, we grid-sweep a curated subset of OpenROAD-flow-scripts variables (e.g., `ABC_AREA`, `ASAP7_USE_VT`, `CORE_UTILIZATION`, `PLACE_DENSITY`, `SYNTH_HIERARCHICAL`; full list in Section D.3) and identify the configuration with the best target-metric value. This configuration defines the reference fix and task target; task instances are then created by controlled perturbations that move the design *away* from the reference. **L1–L2 (single- vs. multi-knob perturbation). L1** perturbs one uniformly sampled knob to another reasonable value that worsens the target metric. **L2** jointly perturbs several knobs, typically 2–3, under the same degradation constraint. The agent is *not* told which knobs were changed; it observes only the degraded post-flow state and must localize responsible parameters through tool calls. **L3 (Performance-only re-architecture).** Some performance gaps cannot be closed by knob retuning and require source-level restructuring, such as pipelining, retiming, or SDC changes. We restrict L3 to Performance because Power and Area gaps in our pool are generally configuration-closable.

### 3.2.3 PPA-Multi

PPA-Multi evaluates multi-objective tuning under explicit metric trade-offs (light-yellow block in Figure 2). For each design we conduct the same grid sweep used in PPA-Mono, and for each task we

Table 3: **Overall performance on POSTEDA-BENCH.** For both SR and VRR/NIS, all values are in % and *larger is better*. Within each performance column, the largest observed mean is **bold** and the second-largest is underlined. "–" means not applicable (ORFS-Agent targets PPA only).

| | DRC-Essential | | DRC-Reasoning | | PPA-Mono | | PPA-Multi | |
|---|---|---|---|---|---|---|---|---|
| | *SR(%)* | *VRR(%)* | *SR(%)* | *VRR(%)* | *SR(%)* | *NIS(%)* | *SR(%)* | *NIS(%)* |
| *React* | | | | | | | | |
| GPT-5 | 84.50 | 91.94 | 31.33 | 33.80 | 39.42 | 51.59 | 11.50 | 23.69 |
| GPT-5-mini | 65.00 | 75.21 | 20.66 | 21.88 | 42.28 | 47.74 | 10.00 | 17.36 |
| Gemini-3-Flash-preview | 85.50 | 87.78 | 36.66 | 36.75 | 48.56 | 54.16 | 18.50 | 51.62 |
| Claude Opus 5 | **95.00** | **95.17** | **54.67** | **54.67** | 40.57 | 47.69 | 2.00 | -1.39 |
| DeepSeek-V3.2 | 71.00 | 79.78 | 16.66 | 18.64 | 1.14 | 7.84 | 0.00 | 1.33 |
| Qwen3.5-122B-A10B | 51.50 | 58.97 | 13.33 | 14.09 | 39.42 | 47.84 | 4.00 | 35.46 |
| Gemma-4-31B-it | 82.00 | 92.02 | 27.99 | 32.05 | **64.56** | **69.34** | 11.00 | 16.34 |
| Qwen3.5-27B | 68.50 | 77.66 | 27.32 | 30.07 | 30.85 | 42.58 | 0.0 | -39.34 |
| Qwen3.5-9B | 56.00 | 64.85 | 5.99 | 5.99 | 25.71 | 36.56 | 1.00 | 10.88 |
| *Proposer-Critic* | | | | | | | | |
| GPT-5 | 60.50 | 71.57 | 15.99 | 17.00 | 42.85 | 50.36 | 0.50 | 11.79 |
| Qwen3.5-122B-A10B | 66.50 | 75.27 | 17.33 | 18.89 | 39.99 | 54.16 | 2.50 | 25.03 |
| Gemma-4-31B-it | 78.00 | 87.00 | 31.33 | 33.51 | 42.28 | 49.95 | 2.00 | -1.33 |
| Qwen3.5-9B | 54.50 | 62.58 | 8.66 | 9.18 | 3.99 | 13.61 | 0.00 | -16.54 |
| *Domain-specific Agent (ORFS-agent‡)* | | | | | | | | |
| GPT-5 | – | – | – | – | 25.71 | 33.75 | 5.00 | 50.28 |
| Qwen3.5-122B-A10B | – | – | – | – | 40.00 | 48.91 | **20.00** | **67.80** |
| Gemma-4-31B-it | – | – | – | – | 47.14 | 47.31 | 1.50 | -15.90 |

§: Excludes script-generating frameworks ChatEDA (Wu et al., 2024) and MCP4EDA (Wang et al., 2025).
‡: ORFS-Agent with automated search-space inference (no manual parameter/range definitions).

set targets at points selected from the Pareto frontier so that they are jointly achievable but inaccessible by greedy single-metric optimization. **L1 (two-objective trade-off).** Each task targets a point on a 2D Pareto front. We cover all three metric pairs—performance–power, performance–area, and power–area—so the corpus jointly stresses every pairwise trade-off (Figure 4 illustrates representative combined 2D fronts). **L2 (three-objective trade-off).** Each task targets a point on the full 3D Pareto front spanning performance, power, and area; targets are chosen from the frontier to ensure achievability while precluding shortcut solutions. In both levels the initial configuration sits in the off-frontier interior, so the agent must *reach the front from the interior* while satisfying floor constraints on all non-target metrics: the score in Eq. 3 explicitly penalizes regressions on those constraints. PPA-Multi therefore tests whether agents can reason about coupled parameters and balance non-trivial trade-offs, rather than greedily optimize a single dimension.

# 4 EXPERIMENTS

## 4.1 EXPERIMENTAL SETUP

**Models.** We evaluate four commercial models (GPT-5, GPT-5-mini, Gemini-3-Flash-preview, Claude Opus 5) and five open-source models: MoE (DeepSeek-V3.2, Qwen3.5-122B-A10B), mid-size dense (Qwen3.5-27B, Gemma-4-31B-it), and small dense (Qwen3.5-9B). **Agent frameworks.** We compare **ReAct** (Yao et al., 2023b), **Proposer–Critic** (Madaan et al., 2023), and an adapted **ORFS-Agent** (Ghose et al., 2025) (Bayesian optimization, PPA only; Section E). **Run protocol.** We report mean SR/VRR/NIS over 5 runs per (model, scaffold, task). Paired task-level bootstrap CIs appear in Section H. ReAct and Proposer–Critic use adaptive caps of 8 DRC / 16 PPA agent actions; ORFS-Agent uses 18 OpenROAD runs per PPA task (Sections E.2 and E.3).

## 4.2 MAIN RESULTS

Table 3 shows that no evaluated configuration leads all four task families. Claude Opus 5 has the largest observed DRC-Essential SR/VRR (95.00%/95.17%) and DRC-Reasoning SR/VRR (54.67%/54.67%); Gemma-4-31B-it leads PPA-Mono SR/NIS (64.56%/69.34%), and ORFS-wrapped Qwen3.5-122B leads PPA-Multi (20.00%/67.80%). DeepSeek-V3.2 drops from 71.00%

Table 4: **DRC-Bench per-level performance.** SR/VRR for representative ReAct backbones on L1–L3; parenthesized Δ values report the difference from the corresponding overall DRC score.

| | L1 | | L2 | | L3 | |
|---|---|---|---|---|---|---|
| | *SR(%)* | *VRR(%)* | *SR(%)* | *VRR(%)* | *SR(%)* | *VRR(%)* |
| *DRC-Essential* | | | | | | |
| GPT-5 | **94.00** (+9.50) | **94.00** (+2.06) | **96.00** (+11.50) | **97.00** (+5.06) | 54.00 (−30.50) | 82.78 (−9.16) |
| Gemini-3-Flash | 92.00 (+6.50) | 92.00 (+4.22) | **96.00** (+10.50) | 96.00 (+8.22) | **62.00** (−23.50) | 71.13 (−16.65) |
| Gemma-4-31B-it | **94.00** (+12.00) | **94.00** (+1.98) | **96.00** (+14.00) | 96.00 (+3.98) | 44.00 (−38.00) | **84.09** (−7.93) |
| *DRC-Reasoning* | | | | | | |
| GPT-5 | **63.07** (+31.74) | **63.07** (+29.27) | 6.00 (−25.33) | 7.00 (−26.80) | 8.56 (−22.77) | 17.74 (−16.06) |
| Gemini-3-Flash | 56.91 (+20.25) | 56.91 (+20.16) | **18.00** (−18.66) | **18.00** (−18.75) | **25.71** (−10.95) | **26.12** (−10.63) |
| Gemma-4-31B-it | 58.45 (+30.46) | 58.45 (+26.40) | 2.00 (−25.99) | 2.00 (−30.05) | 8.56 (−19.43) | 25.95 (−6.10) |

Δ: Difference from the corresponding overall DRC score in Table 3. †: All rows use the ReAct workflow.

DRC-Essential SR to near-zero PPA. **The synthetic-to-practical gap persists with Claude:** the best DRC-Reasoning SR is only $\sim 58\%$ of the best DRC-Essential SR, while PPA-Multi reaches only $\sim 31\%$ of PPA-Mono SR. **Geometric reasoning is decoupled from rule recall:** GPT-5, Gemini-3-Flash, Gemma-4, and Claude retain approximately 34–58% of their DRC-Essential SR on DRC-Reasoning, so the post-flow repair gap remains despite improved backbone capability. **PPA-Mono is target-shortfall:** agents move the right metrics (NIS $>$ SR by 5–12 points) but stop short of the target. **PPA-Multi is trade-off failure:** negative NIS (e.g., Qwen3.5-27B $-39.34$; Gemma-4/ORFS $-15.90$; Claude $-1.39$) reflects constraint regression while chasing the violated objective rather than balancing competing targets. **Scaffold effects are dimension-conditioned.** Proposer–Critic improves some DRC cells by validating edits before dispatch but hurts PPA by over-constraining exploration. ORFS-Agent is complementary: Bayesian optimization lifts Qwen3.5-122B on PPA-Multi ($4.00/35.46 \rightarrow 20.00/67.80$ SR/NIS) yet amplifies weak inferred search spaces for Gemma-4 ($16.34 \rightarrow -15.90$ NIS). Framework choice is thus not a universal upgrade; it trades off edit validation, search-policy flexibility, and search-space coverage (Section F).

#### 4.2.1 DRC-BENCH: PER-LEVEL BREAKDOWN

Table 4 localizes DRC sub-skill failures using per-level SR/VRR for three representative ReAct backbones; parenthesized Δ denotes differences from each model's overall score (Table 3). **Essential L1/L2 are saturated; L3 is the discriminator.** All backbones reach 92–96% SR on Essential L1/L2, so atomic rule recall and contextual clutter filtering are largely solved. The L3 cascade drops to 44–62% SR, while VRR remains much higher for GPT-5 and Gemma-4 ($54.00/82.78$, $44.00/84.09$ SR/VRR): agents often fix most violations but leave one residual after interacting edits. The bottleneck is edit-interaction modeling rather than local rule knowledge. **Reasoning collapses after the first step.** On DRC-Reasoning, L1 remains partially solvable (56.91–63.07% SR), but L2/L3 fall to 2.00–25.71% SR. Gemini-3's advantage lies in multi-step regimes (L2/L3: $18.00/25.71$ vs GPT-5 $6.00/8.56$ and Gemma-4 $2.00/8.56$), not on L1. Practical post-flow violations primarily stress trajectory coherence and coupled geometric planning, not single-edit accuracy.

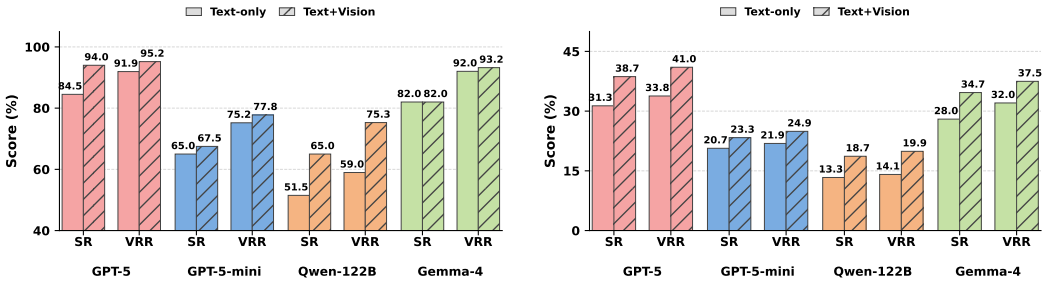
#### 4.2.2 PPA-BENCH: PER-LEVEL BREAKDOWN

We focus on PPA-Performance and PPA-Multi, which expose re-architecture and multi-objective trade-offs. Table 5 reports SR/NIS for Gemini-3-Flash and Gemma-4-31B-it under ReAct, plus ORFS+Qwen-122B under ORFS-Agent; PPA-Power/Area and full bar plots appear in Section F.4. **PPA-Performance L3 is the re-architecture wall.** L3 drops to only 8–20% SR and 20–27% NIS across all three backbones, with negative deltas for every L3 cell. This isolates RTL micro-architectural reasoning as the bottleneck: agents improve shallow timing knobs on L1/L2, but fail when retiming or pipelining is required. **PPA-Multi exposes constraint-tracking failure, while ORFS improves coverage.** Moving from L1 (two-objective) to L2 (three-objective) lowers SR/NIS for every backbone; Gemma's L2 NIS even turns negative ($-0.10\%$), indicating that as competing objectives increase, agents increasingly regress constraint metrics rather than balance the trade-off. ORFS+Qwen-122B still leads PPA-Multi overall ($20.00/67.80$ SR/NIS), so structured exploration helps cover the Pareto frontier, but does not remove the L2 trade-off gap.

Table 5: **PPA-Bench per-level performance.** SR/NIS for representative backbones on PPA-Performance and PPA-Multi; parenthesized Δ values report the difference from the corresponding dimension-level overall score. The best score in each column is bolded, with ties sharing bold.

| *PPA-Performance (Mono)* | | | | | | |
|---|---|---|---|---|---|---|
| | **L1** | | **L2** | | **L3** | |
| | *SR(%)* | *NIS(%)* | *SR(%)* | *NIS(%)* | *SR(%)* | *NIS(%)* |
| Gemini-3-Flash | 56.00 (+21.34) | 59.99 (+17.70) | **40.00** (+5.34) | 40.28 (−2.01) | 8.00 (−26.66) | **26.59** (−15.70) |
| Gemma-4-31B-it | **84.00** (+37.34) | **84.00** (+34.58) | 36.00 (−10.66) | 37.78 (−11.64) | **20.00** (−26.66) | 26.50 (−22.92) |
| ORFS+Qwen-122B | 20.00 (−6.66) | 38.61 (+3.27) | **40.00** (+13.34) | **47.40** (+12.06) | **20.00** (−6.66) | 20.00 (−15.34) |
| *PPA-Multi* | | | | | | |
| | **L1** | | **L2** | | **Overall** | |
| | *SR(%)* | *NIS(%)* | *SR(%)* | *NIS(%)* | *SR(%)* | *NIS(%)* |
| Gemini-3-Flash | **25.00** (+6.50) | 54.93 (+3.31) | 12.00 (−6.50) | 48.31 (−3.31) | 18.50 (+0.00) | 51.62 (+0.00) |
| Gemma-4-31B-it | 17.00 (+6.00) | 32.77 (+16.43) | 5.00 (−6.00) | -0.10 (−16.44) | 11.00 (+0.00) | 16.34 (+0.00) |
| ORFS+Qwen-122B | **25.00** (+5.00) | **73.85** (+6.05) | **15.00** (−5.00) | **61.75** (−6.05) | **20.00** (+0.00) | **67.80** (+0.00) |

Δ: Difference from the corresponding dimension-level overall score; negative deltas are red.
†: Gemini-3-Flash and Gemma-4-31B-it use ReAct; ORFS+Qwen-122B uses ORFS-Agent.
‡: PPA-Power/Area figures and analysis are provided in Section F.4.




(a) DRC-Essential. (b) DRC-Reasoning.

Figure 5: **Effect of vision modality on DRC-Bench.** Each subfigure combines SR and VRR for four backbones under text-only and text+vision settings.

### 4.3 Effect of Vision Modality on DRC-Bench

We pair each text-only backbone (GPT-5, GPT-5-mini, Qwen3.5-122B-A10B, Gemma-4-31B-it) with its vision-augmented variant under ReAct; vision runs add `vision_query_with_pts`, which renders a KLayout image of the violation region (Figure 5). **Vision augmentation enhances DRC-Bench:** across all 16 observed (model, dimension, metric) cells the layout-image channel does not reduce SR or VRR (the only zero-delta is Gemma's DRC-Essential SR, 82.00 → 82.00) and yields consistent gains on both DRC-Essential and DRC-Reasoning, supplying geometric evidence that text-only rule identifiers and coordinates do not carry. **DRC-Essential gains vary across backbones.** Qwen3.5-122B has the largest observed gains (+13.5 SR and +16.3 VRR) from a low baseline (51.50/58.97). GPT-5 also gains +9.5 SR (84.50 → 94.00), whereas Gemma-4's SR is unchanged and its VRR increases by +1.20. **DRC-Reasoning gains are positive across backbones.** SR gains are positive for every backbone (+2.67 to +7.33), with similar VRR gains (+3.04 to +7.22); GPT-5-mini gains +2.67 SR and +3.04 VRR, compared with GPT-5's +7.33 SR and +7.22 VRR. These cell-wise gains are descriptive: only two of eight individual SR confidence intervals exclude zero, while the pooled SR gains are positive for both task families (Section H).

### 4.4 Effect of Iteration Budget

We use Gemma-4-31B-it as a fixed backbone to sweep the ReAct iteration cap and evaluate **Reflexion** (Shinn et al., 2023) (Figure 6). **Reflexion protocol.** Reflexion runs two attempts of the per-attempt cap (8 for DRC, 16 for PPA), allocating 16 DRC or 32 PPA actions in total. Each attempt starts from the original task state; a verbal reflection summarized from attempt 1 is supplied to attempt 2 (Section E.6). We compare this two-trial configuration with ReAct at the same per-attempt

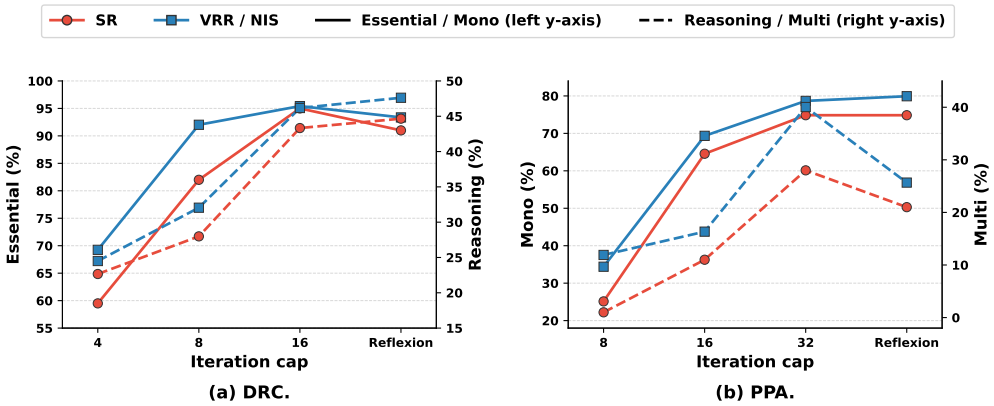



Figure 6: **Effect of iteration cap and Reflexion on Gemma-4-31B-it.** Left: DRC combines DRC-Essential and DRC-Reasoning. Right: PPA combines PPA-Mono and PPA-Multi. Colors encode metrics; solid lines use the left y-axis and dashed lines use the right y-axis.

Table 6: **Thinking-mode ablation** for Gemma-4-31B-it under ReAct.

| | Thinking | DRC-Essential | | DRC-Reasoning | | PPA-Mono | | PPA-Multi | |
|---|---|---|---|---|---|---|---|---|---|
| | | *SR(%)* | *VRR(%)* | *SR(%)* | *VRR(%)* | *SR(%)* | *NIS(%)* | *SR(%)* | *NIS(%)* |
| *React* | | | | | | | | | |
| Gemma-4-31B-it | On | 82.00 | **92.02** | 27.99 | **32.05** | **64.56** | **69.34** | 11.00 | 16.34 |
| | Off | 51.00 | 63.75 | 8.66 | 8.66 | 54.28 | 65.20 | **18.50** | **38.40** |
| | ToT | **85.50** | 90.35 | **29.99** | 31.55 | 31.42 | 40.53 | 0.00 | 1.35 |

cap. **Iteration scaling is dimension-specific.** DRC-Essential and PPA-Mono show diminishing returns, while DRC-Reasoning and PPA-Multi benefit from longer repair and search trajectories across the measured caps. **Two-trial Reflexion improves on the per-attempt baseline.** At the per-attempt cap (iter=8/16), Reflexion lifts DRC-Essential SR from 82.00 to 91.00, DRC-Reasoning from 27.99 to 44.66, PPA-Mono from 64.56 to 74.85, and PPA-Multi from 11.00 to 21.00.

### 4.5 Effect of Thinking Mode

We compare standard chain-of-thought (On), thinking suppressed (Off), and Tree-of-Thought (ToT) (Yao et al., 2023a) (Section E.6) on Gemma-4-31B-it under ReAct (Table 6). **Thinking is dimension-conditioned.** Turning thinking on improves DRC-Essential SR by 31 points, DRC-Reasoning by 19.3, and PPA-Mono by 10.3, but hurts PPA-Multi (SR/NIS drop from $18.50/38.40$ to $11.00/16.34$): DRC and PPA-Mono benefit from reasoning before acting, while PPA-Multi benefits more from faster exploration and tool dispatch under a fixed iteration cap. **ToT helps DRC slightly but fails on PPA.** ToT raises DRC SR modestly ($+3.5$ Essential, $+2.0$ Reasoning) without improving VRR, suggesting branching closes a few near-complete fixes rather than opening new paths; on PPA it sharply reduces Mono ($31.42/40.53$ vs. $64.56/69.34$ SR/NIS) and nearly fails Multi ($0.00/1.35$), so extra deliberation is not a substitute for search coverage.

## 5 Conclusion

We presented PostEDA-Bench, a 145-task hierarchical benchmark that jointly evaluates sign-off DRC fixing and PPA convergence. Experiments across nine LLMs under multiple agent scaffolds expose a synthetic-to-practical asymmetry—agents do well on DRC-Essential and PPA-Mono but degrade sharply on DRC-Reasoning and PPA-Multi—and identify trade-off reasoning, rather than knob knowledge, as the PPA-Multi bottleneck; vision augmentation improves pooled DRC success rates. PostEDA-Bench provides a grounded testbed to drive future research on geometry-aware DRC repair and trade-off-faithful PPA optimization. In future work, we will expand PPA-Bench with additional tasks that more closely reflect industrial design scenarios, aiming to narrow the gap between academic research and industrial practice.

## IMPACT STATEMENTS

This paper presents work whose goal is to advance the field of machine learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

## LLM USAGE STATEMENT

The authors wrote the initial draft of the manuscript. Large language models (LLMs) were subsequently used to polish the language and provide suggestions for improving the clarity, organization, and presentation of the manuscript.

### REPRODUCIBILITY STATEMENT

Most tasks in POSTEDA-BENCH are manually constructed by the authors. Reproducing evaluation therefore starts from the released task artifacts, prompts, and scoring specifications, with construction scripts and provenance supplied where applicable (Section B). The pinned EDA tools, per-task manifests, agent interfaces, and inference protocols are documented in Sections C to E. All released tasks can be scored with open-source tools. Five-run means and task-level uncertainty summarize evaluation outcomes (Section H). Runtime, API-cost accounting, and benchmark versioning are documented in Sections B and E.1.

## REFERENCES

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=KuPixIqPiq.

L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric. Asap7: A 7-nm finfet predictive process design kit. *Microelectronics Journal*, 53:105–115, 2016. doi: 10.1016/j.mejo.2016.04.006.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: an embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Yonggan Fu, Yongan Zhang, Zhongzhi Yu, Sixu Li, Zhifan Ye, Chaojian Li, Cheng Wan, and Yingyan Celine Lin. Gpt4aigchip: Towards next-generation AI accelerator design automation via large language models. In *IEEE/ACM International Conference on Computer Aided Design, ICCAD 2023, San Francisco, CA, USA, October 28 - Nov. 2, 2023*, pp. 1–9. IEEE, 2023. doi: 10.1109/ICCAD57390.2023.10323953. URL https://doi.org/10.1109/ICCAD57390.2023.10323953.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Amur Ghose, Andrew B. Kahng, Sayak Kundu, and Zhiang Wang. Orfs-agent: Tool-using agents for chip design optimization, 2025. URL https://arxiv.org/abs/2506.08332.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VtmBAGCN7o.

Zesong Jiang, Qing Zhang, Cheng Liu, Long Cheng, Huawei Li, and Xiaowei Li. Iicpilot: An intelligent integrated circuit backend design framework using open eda, 2024. URL https://arxiv.org/abs/2407.12576.

Mingjie Liu, Nathaniel Pinckney, Brucek Khailany, and Haoxing Ren. VerilogEval: evaluating large language models for verilog code generation. In *2023 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2023.

Mingjie Liu, Teodor-Dumitru Ene, Robert Kirby, Chris Cheng, Nathaniel Pinckney, Rongjian Liang, Jonah Alben, Himyanshu Anand, Sanmitra Banerjee, Ismet Bayraktaroglu, Bonita Bhaskaran, Bryan Catanzaro, Arjun Chaudhuri, Sharon Clay, Bill Dally, Laura Dang, Parikshit Deshpande, Siddhanth Dhodhi, Sameer Halepete, Eric Hill, Jiashang Hu, Sumit Jain, Ankit Jindal, Brucek Khailany, George Kokai, Kishor Kunal, Xiaowei Li, Charley Lind, Hao Liu, Stuart Oberman, Sujeet Omar, Ghasem Pasandi, Sreedhar Pratty, Jonathan Raiman, Ambar Sarkar, Zhengjiang Shao, Hanfei Sun, Pratik P Suthar, Varun Tej, Walker Turner, Kaizhe Xu, and Haoxing Ren. Chipnemo: Domain-adapted llms for chip design, 2024. URL https://arxiv.org/abs/2311.00176.

Yao Lu, Shang Liu, Qijun Zhang, and Zhiyao Xie. Rtllm: An open-source benchmark for design rtl generation with large language model. In *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 722–727. IEEE, 2024.

Yiyi Lu, Hoi Ian Au, Junyao Zhang, Jingyu Pan, Yiting Wang, Ang Li, Jianyi Zhang, and Yiran Chen. Autoeda: Enabling eda flow automation through microservice-based llm agents, 2025. URL https://arxiv.org/abs/2508.01012.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

OpenCores.org. Opencores: Home, 2026. URL https://opencores.org/. Accessed January 29, 2026.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessí, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: language models can teach themselves to use tools. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Yiting Wang, Wanghao Ye, Yexiao He, Yiran Chen, Gang Qu, and Ang Li. Mcp4eda: Llm-powered model context protocol rtl-to-gdsii automation with backend aware synthesis optimization, 2025. URL https://arxiv.org/abs/2507.19570.

Haoyuan Wu, Zhuolun He, Xinyun Zhang, Xufeng Yao, Su Zheng, Haisheng Zheng, and Bei Yu. ChatEDA: A Large Language Model Powered Autonomous Agent for EDA. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

Haoyuan Wu, Haisheng Zheng, Zhuolun He, and Bei Yu. Divergent Thoughts toward One Goal: LLM-based Multi-Agent Collaboration System for Electronic Design Automation. In *Annual Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics*, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023b.

Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. CodeAgent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13643–13658, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.737. URL https://aclanthology.org/2024.acl-long.737/.

# A Additional Related Work

## A.1 EDA background

**Layout and PDK layers.** A physical layout is a stack of two-dimensional polygon layers defined by a process design kit (PDK). These include transistor-related layers and metal interconnect layers, with vias connecting conductors across layers. In PostEDA-Bench, the layout tools expose transistor-related layers, metals `M1`–`M9`, and vias `V0`–`V9` (Section C.3); the process is ASAP7 (Clark et al., 2016). Each layout is supplied as a GDS file, a binary geometric database containing the polygons and cell hierarchy that the agent inspects and edits.

**DRC decks, sign-off, and residual violations.** A DRC deck is a machine-readable implementation of the foundry or PDK design rules. For a fixed layout, deck, and checker configuration, running DRC determines which geometric constraints are violated and returns their locations. Sign-off DRC applies the designated deck to assess whether the final layout satisfies those rules. Synthesis, placement, and routing produce the layout; violations that remain after the flow and its built-in repair procedures are *residuals*. Engineers inspect and repair these residuals while avoiding new violations. Such post-flow residuals are the source of DRC-Reasoning (Section 3.1.3).

**Kinds of violation.** The rule families considered here constrain geometric relations between polygons on one or two layers: minimum width, spacing, enclosed or overlap area, and layer-pair extension. For example, a spacing rule specifies the minimum permitted separation between shapes, while an extension rule requires one layer to extend beyond another by a prescribed distance. A violation is therefore a spatial property of the actual layout. Its rule identifier and textual description specify the requirement, but determining a valid repair also requires the offending coordinates and surrounding geometry.

**PPA constraints.** Power–performance–area (PPA) targets constrain total power, clock period (represented by effective period in our evaluation), and die area. These quantities are coupled: a change that improves one can worsen another. Multi-objective tasks therefore combine optimization targets with quality floors on metrics that must remain acceptable; for the lower-is-better quantities used here, these floors are expressed as upper bounds. Agents must meet all specified thresholds by adjusting flow parameters, timing constraints, or RTL. PPA-Mono isolates one objective without bounds on the other two, whereas PPA-Multi explicitly evaluates simultaneous target attainment (Equation (3)).

## A.2 Evaluation gaps in prior EDA-agent work

Table 1 highlights several gaps in existing evaluations. **Objective and scale.** ChatEDA-bench (Wu et al., 2024), ChipNeMo (Liu et al., 2024), iEDA-bench (Wu et al., 2025), and AutoEDA (Lu et al., 2025) emphasize script generation or flow execution, providing limited evidence of convergence to explicit PPA targets. Optimization studies also have narrow coverage: GPT4AIGChip (Fu et al., 2023), IICPilot (Jiang et al., 2024), and ORFS-Agent (Ghose et al., 2025) evaluate six accelerator workloads, four designs, and six circuit–technology instances, respectively.

**Hierarchy and toolchain coverage.** Existing suites, including MCP4EDA (Wang et al., 2025), use flat or coarse task groupings that do not isolate configuration recovery, RTL re-architecture, and coupled PPA trade-offs. Evaluations are also commonly tied to a particular flow, such as Vivado HLS or OpenROAD, limiting evidence of generalization across toolchains.

**Missing DRC repair.** None of the related benchmarks in Table 1 evaluates residual sign-off DRC fixing, leaving geometric reasoning, layout-context filtering, and interacting repair sequences

unmeasured. These gaps motivate POSTEDA-BENCH's joint, hierarchical evaluation of DRC repair and target-driven PPA convergence.

# B BENCHMARK CONSTRUCTION DOCUMENTATION

**Filtering and deduplication.** The source pool is filtered in the following order: parsing, top-module resolution, synthesis, standard-cell scale, license/release eligibility, and deduplication. Deduplication is performed across all sources rather than within each source separately. We first flag candidates with identical or near-identical top-module names, then compare port signatures (name, direction, and bit width), and finally compare post-synthesis gate-level structure. Designs with more than 90% gate-level overlap are merged, retaining the instance with more complete source files, clearer provenance, or broader downstream task utility.

**Contamination control.** The prompt is the part most likely to resemble pretraining data—public source RTL, a published ASAP7 rulebook—but it is not what the agent must produce.

**DRC-Bench.** The prompt does not stand alone: the agent is handed a GDS layout, a binary geometric database, and must localize the violation inside it and emit an edit at exact coordinates KLayout then re-checks. The rulebook says M1 corner-to-corner spacing is 20 nm; it does not say which of `gcd`'s polygons violates it, at which coordinate, or which repair clears it without creating a new one.

**PPA-Bench.** The absorbed RTL is the starting design; the scored object is the modification that closes on this task's targets. Memorizing the public source gives an agent the state it was already handed, not the retiming, pipelining or knob setting that reaches an unseen target—verified by an OpenROAD run plus the applicable testbench-based check (Section D.5). The gap between ingested RTL and required fix is the task.

**Reproducibility of manually constructed tasks.** Most tasks are manually constructed, and evaluation is reproduced from the released initial artifacts. Each evaluation unit includes its prompt, target or violation specification, tool environment, and scoring configuration. Users can rerun an agent from that fixed initial state and independently recompute SR, VRR, and NIS from the resulting artifacts and reports. Construction scripts are retained where applicable, alongside the source-design provenance, rule or knob choices, and task metadata documenting the manually selected cases. The DRC step-count protocol is documented separately in Section C.6.

All released tasks can be evaluated and scored with open-source tools: KLayout for DRC and OpenROAD for PPA. Tool versions, per-task manifests, and evaluation drivers are specified in Sections C.1, C.2, D and D.2. Fixed artifacts and scoring rules make outcomes auditable, while five-run evaluation and retained tool traces document agent behavior (Sections E.1 and H).

**Release package and versioning.** The anonymized submission package contains task definitions, prompts, source RTL, GDS files for DRC-Bench, flow configurations, DRC decks, report parsers, scoring scripts, agent tool interfaces, and machine-readable dataset metadata describing provenance, licenses, and intended use. For DRC-Bench, released layouts, prompts, sign-off reports, and KLayout setup are sufficient for agent evaluation without regenerating Innovus-derived tasks; for PPA-Bench, each released OpenROAD project directory includes the files, reports, driver, and pinned flow setup needed for re-evaluation.

# C DRC-BENCH: REPRODUCIBILITY, TOOL CONTRACTS, AND CONSTRUCTION DETAILS

## C.1 SOFTWARE STACK AND TOOL VERSIONS

DRC-Bench is reproduced end-to-end with the following pinned stack:

- **PDK:** ASAP7 (Clark et al., 2016), the predictive 7 nm process design kit; the bundled DRC deck is used as-is for sign-off.

- **Sign-off DRC:** KLAYOUT v0.30.3 driven by the ASAP7 DRC deck. KLayout alone is sufficient to score any submitted agent.
- **LVS availability:** The certified ASAP7 layout-versus-schematic (LVS) deck targets MENTOR CALIBRE v2018, a legacy version no longer broadly accessible to academic groups. The original evaluation therefore uses DRC-clean scoring without a mandatory Calibre LVS gate. We subsequently wrote a KLayout-based checker to audit connectivity and device preservation in archived repairs (Section C.7).
- **Open-source flow (used to seed DRC-Reasoning L1/L2 and the post-flow layout of DRC-Essential L2):** OPENROAD-FLOW-SCRIPTS; the sweep covers the broad set of tunable variables documented in `FlowVariables.md` (clock period, utilization, placement effort, routing effort, and many others) rather than a fixed shortlist, so the resulting violation pool is diverse along multiple axes.
- **Commercial flow:** SYNOPSYS DESIGN COMPILER W-2024.09-SP5-3 followed by CADENCE INNOVUS 21.1, with sweeps over the standard synthesis-, placement-, and routing-effort knobs exposed by both tools.
- **Eval driver:** `eval/eval_drc_sr.sh`. Default configuration: `TEMPERATURE=0`, `NUM_RUNS=5`, `MAX_PARALLEL_AGENTS=5`. The per-task tool-call iteration limit is treated as an evaluation hyperparameter rather than a fixed cap and is varied in our iteration ablation. The driver reports per-task SR (mean over 5 runs) and per-task VRR, aggregated per-class and overall.
- **Agent tools:** released under `agents/drc/tools/` (full schemas in Section C.3).

### C.2 PER-TASK LAYOUT AND RELEASE MANIFEST

Every task is a self-contained directory:

- `top_cell.gds`: the input layout supplied to the agent.
- `prompt.txt`: the natural-language prompt embedding the violation list (rule, geometry type, coordinates).
- `info.json`: structured metadata—rule identifier(s), initial violation count, top-cell name, and sign-off DRC runtime.
- `6_drc_count.rpt`: the raw KLayout DRC report used to construct `prompt.txt`.
- `create_errors.py`, `drc_error_collection.py`: the construction scripts that produced the violation, retained for auditability.

Tasks are organized as `benchmark/drc_bench/drc_essential/{L1,L2,L3}/` and `benchmark/drc_bench/drc_reasoning/{L1,L2,L3}/`. The full benchmark, agent tools, and per-task evaluation logs will be released under a permissive license upon acceptance.

### C.3 AGENT TOOL CONTRACTS

The five tools exposed to the agent have the following contracts (full Python signatures and return schemas are released under `agents/drc/tools/`).

- `query_with_pts(input_gds, level, cell_name, type, direct_layers, context_layers, query_pts, second_level_index_range, third_level_index_range)`: hierarchical textual query at three zoom levels—*first* (shapes directly causing the violation), *second* (shapes inside the bounding box of those shapes), *third* (nearest neighbours just outside the bounding box). `type` is one of `edge-pair`, `polygon`, `edge`; `query_pts` carries coordinates in the layout's native units. The 38 valid layer names span `Active`, `Gate`, `LIG`, `LISD`, `NSELECT`, `PSELECT`, `SDT`, `M1–M9`, `V0–V9`, and several PDK-specific markers. Returned shapes are sorted by proximity to the violation; pagination over context shapes is exposed via `*_index_range`.
- `vision_query_with_pts(...)`: same query semantics as above but renders a KLayout image of the queried region. The image is converted to RGB JPEG, thumbnail-resized to fit within $1024\times1024$ at quality 85, Base64-encoded, and returned as a data URI for direct ingestion by vision-language models.

- `add_shape(...)`, `change_shape(...)`, `move_cell(...)`: editing primitives that operate on the working GDS in place; they validate layer membership and shape primitives and reject illegal edits with structured error messages.
- `query_drc_report(...)`: re-runs the KLayout sign-off DRC deck on the current GDS and returns the updated violation list. Each call counts as one iteration against the per-task iteration limit (varied across experiments).

### C.4 DRC-Essential rule and layer coverage

DRC-Essential L1 and L2 jointly cover several rule families and 14 layers from the ASAP7 DRC deck, balanced to one representative case per layer per family where the rule admits the layer. Rule families include minimum horizontal/vertical width, minimum horizontal/vertical spacing, minimum enclosed/overlap area, layer-pair extension, and ASAP7-specific quantization rules (e.g., `ACTIVE/SDT vertical width must be an integer multiple of 27 nm`). Two rules from the deck are excluded after pruning topologically equivalent variants; the released `statistic.json` enumerates the per-rule and per-layer task assignments. For L1 construction, examples of topology-specific rules that are hand-drawn by physical-design engineers include `NSELECT.GATE.EX.1-2` and `PSELECT.GATE.EX.1-2`, which require arranging selector and gate shapes to expose the intended extension violation. Examples of rules that admit parametric templates include `WELL.S.2` and `WELL.W.2`, where `create_errors.py` instantiates simple rectangles with controlled spacing or width below the ASAP7 threshold.

### C.5 Built-in Innovus DRC repair study

To verify that DRC-Reasoning does not simply stop before commercial tools finish, we applied Innovus built-in DRC/ECO repair to five representative post-route designs and then re-ran sign-off DRC. As shown in Table 7, built-in repair substantially reduces but does not eliminate violations; DRC-Reasoning is constructed from this residual class of violations that still require manual or agent-driven geometric repair.

Table 7: **Residual DRC violations after Innovus built-in repair.**

| Design | gcd | aes | ibex | jpeg | riscv |
|---|---|---|---|---|---|
| Original | 34 | 972 | 1,551 | 4,481 | 9,395 |
| After Innovus repair | 22 | 760 | 796 | 1,815 | 2,750 |

### C.6 DRC-Reasoning step-count labelling protocol

The initial step-count labels that partition DRC-Reasoning into L1 and L2 were produced by a physical-design engineer with 5+ years of industrial sign-off experience and then cross-verified by two physical-design engineers. The reference-labelling procedure is:

1. The engineer loads the violation in KLayout and inspects the offending geometry alongside its surrounding layers.
2. The engineer manually constructs the shortest fix sequence using only the editing operations exposed to the agent (`add_shape`, `change_shape`, `move_cell`); inspection-only tool calls are not counted.
3. The integer count of editing tool calls is recorded as the case's step count.
4. After a one-week delay the engineer re-labels the same case without access to the prior label; cases with disagreeing labels are re-evaluated under the protocol until convergence, and any case that still admits competing minimal fixes is retained at the smaller step count.

**Two-engineer cross-verification protocol.** Two physical-design engineers completed cross-verification of the L1/L2 labels. Each independently inspected the cases and constructed a shortest valid edit sequence using the same agent operations, without access to the other engineer's step count. They compared counts and replayed their proposed repairs under the same KLayout sign-off deck.

Differences were resolved by checking sequence validity and testing whether a shorter valid sequence existed; the smallest verified edit count was retained. Both initial counts, proposed edit sequences, and the agreed count were recorded with the task metadata.

### C.7 LVS-BASED AUDIT OF FUNCTIONAL PRESERVATION

**Extraction and comparison.** To check whether successful DRC repairs preserve circuit connectivity and devices, we wrote a new verification script using KLAYOUT v0.30.3. For each audited task-run, the script uses the two GDS files already retained by the evaluation harness: the golden input `top_cell.gds` supplied to the agent and its output `modified.gds`. It extracts a netlist from each layout with `pya.LayoutToNetlist` over the ASAP7 conductor/via stack, then performs the LVS netlist-comparison step using the golden layout's extracted netlist as the reference schematic. Each net is keyed by the set of `(instance, pin)` terminals it connects. The comparison therefore detects opens, shorts, and via deletions that alter connectivity, while tolerating coordinate changes such as wire widening that preserve those terminal connections. The script also checks the device census.

**Decision policies.** We evaluate two policies. The *connectivity/device-evidence policy* declares a break when the extracted netlist or device census changes. The *strictest policy* additionally rejects any movement of an instance classified as a signal cell, regardless of whether its connectivity changes. This extra displacement rule provides a conservative check on geometry edits.

**Results and effect on reported scores.** We audited every archived DRC-successful task-run for Gemini-3-Flash, GPT-5, and Gemma-4-31B-it, totaling $648$ task-runs. The connectivity/device-evidence policy detected $0/648$ breaks ($0.00\%$); the strictest policy flagged $2/648$ runs ($0.31\%$). Both flags correspond to the same edit: a `VIA_VIA12` instance moved by $3$ nm. In both runs, extraction yields $461$ nets before and after the edit, with zero lost nets, zero new nets, and an equal result from `NetlistComparer`; the flags are caused solely by the additional movement rule. Elsewhere in the corpus, the strictest policy flags $116$ circuit-breaking edits, all in runs that DRC scoring had already rejected.

Thus, no connectivity or device changes were detected among the $648$ audited DRC successes. Recomputing SR with the additional connectivity/device-evidence requirement reproduces the archived success rates to four decimal places, leaving the reported results unchanged for these audited runs.

## D PPA-BENCH: REPRODUCIBILITY, TOOL CONTRACTS, AND CONSTRUCTION DETAILS

### D.1 SOFTWARE STACK AND TOOL VERSIONS

PPA-Bench is reproduced end-to-end with the following pinned stack:

- **PDK:** ASAP7 (Clark et al., 2016), identical to the DRC-Bench setup.
- **Flow:** OPENROAD-FLOW-SCRIPTS, pinned to commit `a2bb042b6` (release tag `26Q1-127-ga2bb042b6`). The flow is invoked inside a Docker container so that local installations cannot perturb scoring.
- **Eval driver:** `eval/eval_ppa_sr.sh`. Default configuration: `TEMPERATURE=0`, `NUM_RUNS=5`, `TOOL_TIMEOUT=1800` s (30-minute wall-clock cap per `run_openroad` call), `MAX_PARALLEL_AGENTS=5`. The per-task tool-call iteration limit is varied as an evaluation hyperparameter rather than fixed.
- **Scorer:** `benchmark/utils/ppa_score.py`, which implements the SR / NIS computation described in Eq. 2–3.
- **Agent tools:** released under `agents/ppa/tools/` (full schemas in Section D.4).

### D.2 PER-TASK LAYOUT

Each PPA-Bench task ships as a self-contained project directory:

- `src/`: the design's Verilog sources.
- `config/` (a.k.a. `config.mk`, `constraints.sdc`): OpenROAD configuration and SDC constraints, the primary surface the agent edits.
- `logs/`, `reports/`, `objects/`, `results/`: the previous-iteration intermediate artifacts produced by OpenROAD.
- `prompt.txt`: the natural-language prompt summarizing current PPA values, target thresholds, any constraint floors, and a pointer to `FlowVariables.md`.
- `info.json`: ground-truth metadata (initial values, targets, reference knob/value records). Read access by the agent is blocked at the `read_file` layer so that targets cannot be trivially reached by reading the reference fix; the file is retained on disk for auditability and scoring.

Tasks are organized as `benchmark/L3/asap7/single_dimension/openroad/{area, performance, power}/{class1,2,3}` for PPA-Mono and `benchmark/L3/asap7/multi_dimension/openroad/{class1,2}` for PPA-Multi.

### D.3 Curated knob list for the parameter sweep

The grid sweep used for PPA-Mono and PPA-Multi construction draws from a curated subset of `eval/FlowVariables.md`. Representative knobs include `ABC_AREA`, `ASAP7_USE_VT`, `CORE_UTILIZATION`, `PLACE_DENSITY`, `SYNTH_HIERARCHICAL`, `TNS_END_PERCENT`, `ROUTE_EFFORT`, and `REPAIR_PDN_VIA_LAYER`. The complete list and the per-knob value ranges used in our sweep are released alongside the benchmark.

### D.4 Agent tool contracts

The four PPA-Bench tools have the following contracts (full Python signatures and return schemas are released under `agents/ppa/tools/`).

- `read_file(command)`: executes a shell-style read command inside a per-task sandbox. A whitelist of safe commands is permitted (`cat`, `head`, `tail`, `grep`, `awk`, `sed`, `wc`, `sort`, `uniq`, `cut`, `tr`, `ls`, `find`, `tree`, `jq`, `less`, `more`, `file`, `stat`, `diff`, `strings`, `xxd`, `od`); modifying commands and shell control operators (`>`, `»`, `&&`, command substitution, etc.) are blocked. Pipes are allowed. Reads of `info.json` or paths outside the sandbox are rejected.
- `edit_file(path, old_content, new_content)`: in-place replacement edit on files inside the sandbox. A small set of structural variables in `.mk` files (`PLATFORM`, `DESIGN_NAME`, `DESIGN_NICKNAME`, `VERILOG_FILES`, `SDC_FILE`) is locked to prevent flow-breaking modifications; all other configuration entries, SDC constraints, and Verilog sources are editable.
- `run_openroad(...)`: launches an OpenROAD flow run on the current sandbox, returning stdout/stderr and exit status. Each call is bounded by a `TOOL_TIMEOUT` environment variable. A flow crash returns the error to the agent and counts as one iteration.
- `report_ppa()`: parses the latest reports and returns a structured summary of effective period, total power, and die area together with the gaps to the per-task targets.

### D.5 Testbench-based functional checks

PPA-Bench evaluates target attainment together with each task's applicable testbench-based functional checks. The check in `eval/equiv/check_equiv.sh` compares the original and modified RTL outputs while accounting for cycle-latency and handshake-timing changes introduced by pipelining, retiming, and datapath restructuring.

For designs with a supplied, simulator-compatible testbench, we compile and simulate the original and modified sources with Icarus Verilog (`IVERILOG_LANG=2012`). The testbench at `eval/equiv/tb/<design>_tb.v` drives a representative input sequence and emits cycle-stamped output traces for both versions. Traces are compared in one of two modes, selected per design:

- **Event-matched mode.** For combinational and untimed designs, traces are diffed line-by-line; any divergence fails the check.

- **Latency-sweep mode.** For pipelined or handshake-timed designs, a separate comparator (`eval/equiv/compare.py`) sweeps a latency offset $L \in [0, L_{\max}]$ (default $L_{\max} = 16$ cycles, with an 8-cycle reset warmup) and checks whether $\text{orig}[t] = \text{mod}[t - L]$ over the overlap window for some $L$. The check passes if any offset yields zero mismatches, so legitimate pipeline-depth or handshake-timing changes do not count as functional regressions.

Task-level records identify testbench applicability and retain the trace-comparison outcome or build/error status.

# E Agent Frameworks and Experimental Protocol

## E.1 Models and inference setup

The nine models reported in Table 3 are evaluated under a shared agent codepath, with the model identifiers, inference settings, and tool-call accounting documented below.

- **Commercial.** GPT-5, GPT-5-mini, Gemini-3-Flash-preview, and Claude Opus 5, accessed via their official APIs at the time of evaluation. The exact API model identifiers are recorded in the per-run cost-tracker logs released with the benchmark. Claude uses the identifier `claude-opus-5`, with the temperature parameter omitted from API requests.
- **Open-source.** DeepSeek-V3.2, Qwen3.5-122B-A10B, Qwen3.5-27B, Gemma-4-31B-it, and Qwen3.5-9B, each served via vLLM on a local GPU cluster behind an OpenAI-compatible `/v1` endpoint. Tensor-parallel degree is selected per checkpoint to fit GPU memory; greedy decoding (`temperature=0`) is enforced server-side.

Token usage and tool-call counts are tracked per run via a shared cost-tracking utility shipped alongside the agent code.

**Repeated runs and inference reproducibility.** Each evaluated configuration is run five times from the same task inputs, using temperature zero where supported and the Claude API settings specified above. We report run-averaged scores and retain per-run outcomes and tool traces; uncertainty is assessed at the task level in Section H.

**Runtime and API-cost accounting.** The runtime and cost panels in Table 8 summarize the archived evaluations. Elapsed times were measured on one 128-core node with five agents running concurrently: a single pass over DRC-Bench takes 1.2–1.6 hours and a pass over PPA-Bench takes 4.4–5.0 hours, with approximately 6.7 and 23 hours respectively for the full five-run protocol. Provider API charges are summed from per-task `cost.json` records for one pass. Charges and totals are reproduced as reported, so rounded components may not sum exactly to the displayed total; the five-run protocol incurs approximately five times the per-pass API charge.

Verification dominates wall-clock time. A `run_openroad_flow` call has a median duration of 271–315 seconds across the measured configurations (90th percentile: 1,807 seconds), compared with 2–8 seconds for an LLM turn, and accounts for 83–93% of PPA episode time. A DRC check has a median duration of 62–69 seconds, reaching 481 seconds on the largest designs, compared with 5–10 seconds for a geometry turn. KLayout accounts for 37–61% of DRC wall-clock time for Gemini-3-Flash and Gemma-4. The corresponding OpenROAD call costs roughly as much time as 35–158 LLM turns. These measurements explain why cost and runtime should be reported alongside success rates, and why the relative value of reasoning and additional tool feedback depends on the task family.

**Claude Opus 5 results and tool-use profile.** Claude has the highest observed DRC scores in Table 3: Essential SR/VRR is $95.00/95.17$ and Reasoning SR/VRR is $54.67/54.67$. The gap remains on individual levels: Essential L3 SR is 98.00%, whereas Reasoning L2 SR is 36.00%. Its PPA-Mono SR/NIS is $40.57/47.69$, and PPA-Multi is $2.00/-1.39$; Performance/Area/Power SR is $0/74/68\%$. These measurements use text-only ReAct with the shared action caps. On PPA, Claude averages 14.0 `read_file` calls and 1.33 flow runs per episode, versus 4.88 and 2.05 for Gemma-4. Each dispatched tool call in a parallel batch counts toward the 16-action cap. These traces characterize

Table 8: **API cost and elapsed runtime on POSTEDA-BENCH.** The panels report provider API charges and elapsed benchmark runtime from the archived evaluations.

**API cost per pass (USD; text-only ReAct)**

| Backbone | DRC (70 tasks) | PPA (75 tasks) | Both (145 tasks) |
|---|---|---|---|
| GPT-5-mini | 1.64 | 5.89 | 7.54 |
| GPT-5 | 13.33 | 37.42 | 50.75 |
| Claude Opus 5 | 50.74 | 184.38 | 235.12 |
| Gemma-4 / Qwen3.5 (self-hosted) | 0 | 0 | 0 |

**Elapsed runtime: one 128-core node, five concurrent agents**

| Suite | One pass (hours) | Five-run protocol (hours) |
|---|---|---|
| DRC-Bench (70 tasks) | 1.2–1.6 | $\sim 6.7$ |
| PPA-Bench (75 tasks) | 4.4–5.0 | $\sim 23$ |

API charges are archived per-pass measurements for the listed ReAct backbones, separate from local CPU/GPU expenses. The five-run protocol costs approximately $5\times$ the listed API charges. Charges and totals are reproduced as reported; rounded components may not sum exactly to the total. Elapsed times correspond to benchmark passes on the hardware specified above. See Sections B and E.1 for accounting, Claude's tool-use profile, and release versioning.

how each model allocates its actions between inspection and flow-based verification. No evaluated configuration leads all four task families.

### E.2 ITERATION SEMANTICS PER FRAMEWORK

We adopt a per-framework definition of one iteration so that the iteration cap (8 for DRC-Bench, 16 for PPA-Bench) is interpreted consistently:

- **ReAct.** One iteration is one tool call dispatched by the agent. Pure reasoning turns that do not emit a tool call are not counted.
- **Proposer–Critic.** The proposer drafts a tool call as a textual description; the critic validates or refines the proposal; only the agreed action is dispatched. The full proposer–critic–dispatch triple counts as one iteration; proposal rounds that do not lead to a dispatch are not counted.
- **ORFS-Agent.** ORFS-Agent uses a fixed Bayesian-optimization budget of 6 outer iterations $\times$ 3 parallel candidates per outer iteration $= 18$ OpenROAD candidate runs per task, hardcoded in the wrapper rather than driven by the global iteration cap. Each candidate run is one OpenROAD flow execution; the single LLM call consumed during the search-space-discovery setup phase is not counted against the budget.

**Adaptive iteration cap.** The headline caps in Section 4.1 (8 DRC, 16 PPA) are stated in units of *agent actions* (tool dispatches), not raw LLM calls. Multi-agent scaffolds that consume additional internal LLM calls per dispatch—e.g., Proposer–Critic's proposer→critic→dispatch triple, or ToT's per-node candidate sampling and judge scoring—therefore receive a proportionally larger total LLM-call budget than ReAct's raw tool-call budget. We scale the underlying LLM-call budget linearly with the number of cooperating agents per dispatch, so that adding more agents to a workflow grants correspondingly more reasoning calls. This adaptive scaling keeps cross-scaffold comparisons about *scaffold value* rather than penalising frameworks for needing more thought per action.

### E.3 RUN TERMINATION AND TIMEOUT HANDLING

A run terminates when (i) the success criterion is met, (ii) the iteration cap is reached, (iii) the underlying tool times out, or (iv) the LangGraph recursion limit is hit; in all non-success cases the partial-progress metric is computed from the last valid state. The default `TOOL_TIMEOUT` is 1800 s for OpenROAD flow runs and the equivalent for KLayout DRC. Tool timeouts and flow crashes return the error to the agent and count as one iteration.

**Excluded baselines.** Most prior EDA-LLM benchmarks (Table 1) target script generation from natural language rather than PPA convergence or DRC fixing, and their agents do not expose a comparable success criterion; we therefore exclude them from Table 3. ORFS-Agent (Ghose et al., 2025) is the closest published baseline for PPA optimization and is included via the wrapper described below.

### E.4 ORFS-AGENT SEARCH-SPACE-DISCOVERY WRAPPER

The published ORFS-Agent (Ghose et al., 2025) requires every (design, PDK) pair to ship a hand-curated `opt_config.json` that lists the tunable parameters and their ranges, plus per-design baseline ECP and wirelength values. Our PPA-Bench tasks are driven by natural-language prompts and do not carry this metadata, so we adapt the published agent with an LLM-based setup phase that automates the configuration step. Concretely:

- A single LLM call reads the task's `prompt.txt`, the contents of `config/` (i.e., `config.mk` and `constraints.sdc`), the list of files under `src/`, and `eval/FlowVariables.md`, and returns a structured search space (tunable parameters, ranges, defaults) via a single tool call. Targets and the optimization objective are not picked by the LLM—they are parsed from `info.json` by the orchestrator outside the agent's view—so the discovery step affects only *which knobs to sweep*, not the success criterion.
- The discovered search space is fed into the original GP-driven outer loop: iteration 1 uses Latin Hypercube initialization around the discovered defaults; iterations 2–$N$ propose new candidates by maximizing a UCB acquisition over a Matérn-kernel GP, with diversity-aware top-$k$ selection. Each candidate runs a real OpenROAD flow inside an isolated per-candidate sandbox.
- The evaluation allocates 6 outer iterations with 3 parallel candidates per iteration, totaling 18 OpenROAD candidate runs per task (Section E.2). Each candidate flow execution counts as one ORFS-Agent iteration.

The wrapper preserves the upstream agent's CLI contract so that `eval_ppa_sr.sh` can swap it in for ReAct or Proposer–Critic without modification.

### E.5 FRAMEWORK-LEVEL SYSTEM PROMPTS

We release the system prompts used by ReAct and Proposer–Critic for both DRC-Bench and PPA-Bench under `agents/{drc,ppa}/{react,proposer_critic}/`. The prompts are held fixed across models within a framework so that benchmark scores reflect model and scaffold capability rather than prompt engineering. Each prompt declares (a) the role and tool repertoire available to the model, (b) the success criterion (DRC-clean for DRC-Bench, target metrics + applicable testbench-based functional checks for PPA-Bench), and (c) framework-specific scaffolding—chain-of-thought interleaving for ReAct, role separation between proposer and critic for Proposer–Critic, and the structured search-space schema for the ORFS-Agent setup phase.

### E.6 REFLEXION AND TREE-OF-THOUGHT WRAPPERS (PPA)

We also release PPA-side Reflexion and Tree-of-Thought wrappers used in the iteration-budget and thinking-mode ablations (Sections 4.4 and 4.5); both are layered on top of the same ReAct PPA primitives and tool surface so that scores reflect the search/reflection policy rather than tool plumbing.

**Reflexion (`agents/ppa/reflexion/`).** The wrapper implements the verbal-reinforcement trial-and-reflect loop of Shinn et al. (2023). The user-visible iteration budget is divided evenly across $N$ trials (we use $N$=2, matching the halve-and-reflect-once protocol of Section 4.4). Each trial is a full ReAct rollout with a freshly initialised conversational memory; between trials the agent makes a tool-less Self-Reflection LLM call that summarises lessons from the previous trajectory, and these lessons are appended to the system prompt of the next trial as long-term memory. The Self-Reflection call does not consume iteration budget. To match canonical Reflexion semantics, the working sandbox is snapshotted before trial 1 and restored before every subsequent trial, so each trial sees the original task content; only the verbal reflection is carried across. Cumulative tool-call counts

and token costs are aggregated across trials so the eval driver (`eval_ppa_sr.sh`) sees a single agent-run summary, matching the ReAct-agent contract.

**Tree-of-Thought (`agents/ppa/tot/`).** The wrapper implements the ToT search of Yao et al. (2023a) over the same tool surface as the ReAct PPA agent. At each tree node the agent samples `PARALLEL_NODE` candidate next actions in parallel, an LLM judge scores them, the highest-scored candidate is executed, and the resulting state becomes a new node. Un-executed candidates from every previously-expanded node remain in a global priority queue, so when the current path stalls the search backtracks to the highest-scored alternative from an ancestor. Iteration budget is the total tool-call budget across the whole tree; `max_depth` caps the deepest single path (default 16, matching ReAct's per-run cap), and per-step LLM calls (candidate sampling and scoring) are billed in tokens but do not consume iteration budget. Each node owns a sandbox snapshot under `$AGENT_RESULT_DIR/_tot_snapshots/`; on backtrack, the sandbox is wiped and restored from the chosen ancestor's snapshot before executing. At end of run the sandbox is restored to the best node, ranked by (i) success, (ii) at least one successful `run_openroad_flow` call, (iii) deeper paths over shallower ones, and (iv) judge score, so the eval driver picks up a node whose OpenROAD reports actually exist.

# F Extended Analysis of Main Results

This section expands on the deductions in Section 4.2, focusing on framework-level effects and cross-dimension comparisons that are too detailed to fit inline.

## F.1 Scaffold value is dimension-conditioned and reverses sign on PPA

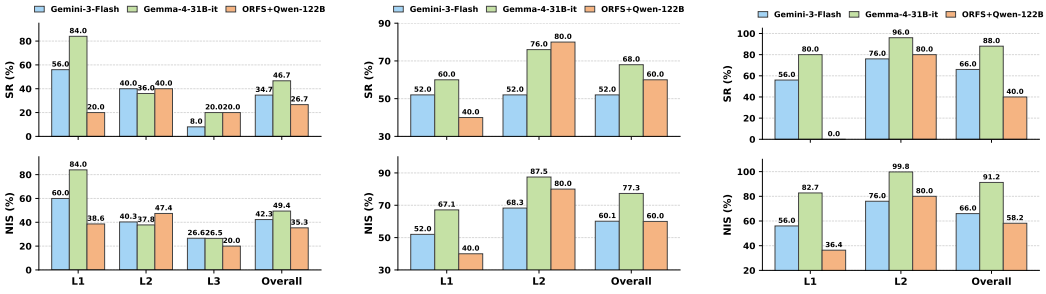
Switching from ReAct to Proposer–Critic raises Qwen3.5-122B's DRC-Essential SR by $+15$ points and Gemma-4-31B-it's DRC-Reasoning SR by $+3.3$, but cuts Gemma-4's PPA-Mono SR by 22 points ($64.56 \rightarrow 42.28$) and Qwen3.5-9B's by 21.7. Since our evaluation uses adaptive, approximately matched budgets across frameworks, this reversal is not explained by fewer allowed optimization steps. Instead, it suggests that proposer–critic validation changes the search policy: on DRC tasks where each call is a local geometric edit, the validator catches malformed edits before dispatch and pays for itself; on PPA tasks, where progress depends on empirical `run_openroad` measurements, critic-side filtering can bias the trajectory toward conservative local changes and reduce broad parameter exploration. Framework choice is therefore not a global hyperparameter, and cross-framework comparisons that aggregate over dimensions will mislead.

## F.2 Bayesian optimization amplifies the LLM's search-space prior

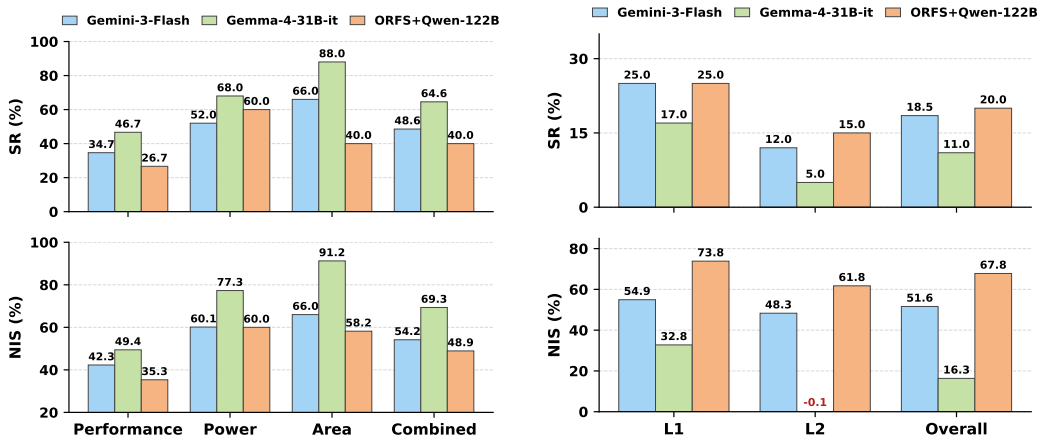
ORFS-Agent lifts Qwen3.5-122B's PPA-Multi NIS from $35.46$ to $67.80$ (the table's highest value, $\sim 1.9\times$) but drives Gemma-4-31B-it's PPA-Multi NIS from $+16.34$ to $-15.90$ and its PPA-Mono NIS from $69.34$ to $47.31$. Both runs share the same outer BO loop and budget; the only difference is the LLM-induced search space inferred from `FlowVariables.md`. The Pareto-frontier failure of pure ReAct is therefore best understood as a poor exploration policy layered on largely correct knob knowledge: when a structured sampler is grafted on top of a competent space, multi-objective SR jumps to first place; when it is grafted on top of a wrong space, it amplifies the error rather than correcting it. The open research problem is LLM-side search-space discovery, not the optimization loop itself.

## F.3 Capability-gap ordering on PostEDA-Bench

Top-cell ratios across dimensions give a quantitative difficulty ordering: PPA-Multi reach is $20.00/64.56 = 31\%$ of PPA-Mono reach, while DRC-Reasoning reach is $54.67/95.00 \approx 58\%$ of DRC-Essential reach. Multi-objective trade-off is therefore the largest absolute capability gap today, ahead of geometric DRC reasoning. This suggests that the next-step research direction is not better DRC-rule prompting but tool-augmented planners with explicit constraint-tracking memory.




(a) PPA-Performance (L1/L2/L3). (b) PPA-Power (L1/L2). (c) PPA-Area (L1/L2).




(d) PPA-Mono (per sub-dimension + Combined). (e) PPA-Multi (L1/L2).

Figure 7: **Full PPA-Bench per-level performance.** SR (top) and NIS (bottom) per construction-time level. Row 1 decomposes PPA-Mono into Performance / Power / Area; row 2 summarizes PPA-Mono per sub-dimension and reports PPA-Multi.

### F.4 Additional PPA-Bench per-level breakdowns

Figure 7 gives the full five-panel PPA decomposition used to support the compact main-text table in Table 5.

**Power/Area show inverted L1→L2.** Both PPA-Power and PPA-Area have L2 ≥ L1 SR for every model (e.g., ORFS+Qwen $40 \to 80\%$ on Power and $0 \to 80\%$ on Area; Gemma $60 \to 76\%$ on Power). This counter-intuitive ordering suggests that joint power/area perturbations create slack: additional misconfigured knobs offer more recovery levers, whereas single-knob cases require identifying the responsible knob exactly.

**The full decomposition clarifies ORFS behavior.** ORFS+Qwen leads PPA-Multi NIS by a wide margin (67.80 vs Gemini 51.62 and Gemma 16.34) and matches or exceeds the others on Power L2 / Area L2, but trails the PPA-Mono Combined score ($40.00/48.91$ vs Gemma $64.56/69.34$) and has $0\%$ SR on Area L1. This supports the main-text conclusion that Bayesian optimization helps through exploration coverage when the inferred search space contains useful knobs, but can amplify a weak inferred space when the task hinges on a specific single knob.

## G Detailed Ablation Readings

We expand on the ablation figures in Sections 4.3 to 4.5. Source tables and per-run logs accompany the released benchmark.

**Detailed iteration-budget reading.** Figure 6 separates two regimes. On DRC-Essential and PPA-Mono, SR follows a concave shape ($59.5 \rightarrow 82.0 \rightarrow 95.0$ and $25.1 \rightarrow 64.6 \rightarrow 74.9$), so extra iterations mostly polish already reachable tasks. DRC-Reasoning and PPA-Multi behave oppositely: Reasoning gains only $+5.3$ SR from $4 \rightarrow 8$ but $+15.3$ from $8 \rightarrow 16$, while Multi gains $+10$ from $8 \rightarrow 16$ and $+17$ from $16 \rightarrow 32$. The PPA-Multi NIS$-$SR gap also narrows then widens ($10.91 \rightarrow 5.34 \rightarrow 12.04$), indicating that additional iterations can improve the violated metric without crossing the success boundary. DRC-Essential shows the opposite pattern: its VRR$-$SR gap collapses from $9.75 \rightarrow 10.0 \rightarrow 0.42$, consistent with all-or-nothing cascaded fixes once enough edit budget is available.

## H Variance and Statistical Uncertainty

**Task-level paired bootstrap.** We quantify uncertainty with a paired cluster bootstrap over tasks using $20{,}000$ resamples. For each task, success is averaged over its five runs. Tasks are resampled as clusters, retaining their repeated-run outcomes and using the same resampled task indices for both configurations in a comparison. This preserves pairing by task rather than treating all task-runs as independent observations. We report differences in SR in percentage points and their $95\%$ confidence intervals. The bootstrap uses the underlying task-level outcomes rather than the rounded means displayed in the main table.

Table 9: **Paired SR differences and uncertainty.** Gemini denotes Gemini-3-Flash-preview, Gemma-4 denotes Gemma-4-31B-it, and Qwen-122B denotes Qwen3.5-122B-A10B. Models use ReAct unless ORFS-Agent is explicitly indicated. Differences are computed as the first configuration minus the second.

| Paired comparison | $\Delta$ SR (pp) | $95\%$ CI (pp) |
|---|---|---|
| DRC-Essential: Gemini $-$ GPT-5 | $+1.00$ | $[-4.00, +6.50]$ |
| DRC-Reasoning: Gemini $-$ GPT-5 | $+5.33$ | $[-5.33, +16.67]$ |
| DRC-Essential: Gemini $-$ Qwen-122B | $+34.00$ | $[+23.00, +45.50]$ |
| PPA-Mono: Gemma-4 $-$ GPT-5 | $+25.14$ | $[+14.29, +37.14]$ |
| PPA-Multi: ORFS+Qwen $-$ Qwen-122B | $+16.00$ | $[+3.50, +29.50]$ |

The near-ties in the first two rows of Table 9 have intervals containing zero and do not support a statistical ordering between the paired models. The larger separations in the remaining rows exclude zero. Thus the bold and underlined entries in Table 3 describe observed means, while the absence of a single leading configuration is a cross-dimension observation. These bootstrap comparisons concern the archived baseline and vision configurations.

**Uncertainty of the vision effect.** Of the eight paired SR improvements from adding vision to four backbones on DRC-Essential and DRC-Reasoning, only two individual intervals exclude zero: GPT-5 on Essential and Qwen3.5-122B-A10B on Essential (Table 10). When pooled over the four backbones, the SR improvement is positive on both task families. The probability that all eight SR differences are non-negative under this analysis is $0.31$. Consequently, the non-decreasing SR/VRR values across all 16 observed metric cells are a description of these runs, not evidence of a universal or statistically significant improvement for every backbone and metric.

Table 10: **Vision-minus-text SR differences.** The first two rows are the individual model–family comparisons whose $95\%$ intervals exclude zero; the remaining rows pool the four evaluated backbones.

| Comparison | $\Delta$ SR (pp) | $95\%$ CI (pp) |
|---|---|---|
| GPT-5, DRC-Essential | $+9.50$ | $[+2.00, +18.50]$ |
| Qwen3.5-122B-A10B, DRC-Essential | $+13.50$ | $[+6.50, +20.50]$ |
| Pooled, DRC-Essential | $+6.37$ | $[+3.38, +9.62]$ |
| Pooled, DRC-Reasoning | $+5.50$ | $[+2.33, +9.00]$ |

**Synthetic-to-practical gap.** For Gemini-3-Flash-preview, the DRC-Essential minus DRC-Reasoning SR gap is $48.83$ percentage points, with a $95\%$ interval of $[29.50, 67.17]$. This supports the cross-family difficulty gap despite uncertainty in nearby within-column model rankings. Per-run, per-task outcomes for all $16$ evaluated DRC configurations and the corresponding PPA records will be released to support independent aggregation and uncertainty analysis.