SCALE: Self-Supervised Constraint-Aware Layout GEneration for Local P&R DRV Fixing at Advanced Nodes

Chia-Tung Ho[1,*], Haoyu Yang[1,*], Guanglei Zhou[1,2,*], Yoshi Nishi[1], Yaguang Li[1], Walker Turner[1], Cunxi Yu[1], Yiran Chen[2] and Brucek Khailany[1]
[1]NVIDIA [2]Duke University

arXiv:2607.21850v1 [cs.CV] 23 Jul 2026

Abstract

As semiconductor manufacturing advances toward sub-2nm nodes, local place-and-route (P&R) design-rule violation (DRV) fixing is increasingly limited by complex rule interactions, dense multi-layer routing geometries, and foundry-specific constraints. While Large Language Models (LLMs) have recently demonstrated strong capabilities in EDA scripting and documentation, their application to *visual* layout understanding remains largely unexplored: diagnosing DRC violations from layout imagery demands precise geometric reasoning and foundry-specific rule knowledge absent from general-purpose VLM training.

We propose *SCALE*, a framework with a self-supervised layout-generation stage for local DRV fixing at advanced nodes. Multi-layer layout geometry is serialized into structured text, and a fine-tuned language model learns to reconstruct randomly masked polygons from surrounding BEOL context alone without violation labels. At inference, natural-language rule constraints and high-temperature sampling steer generation toward diverse, violation-prone layout variants validated by an industrial signoff DRC checker, producing DRC-annotated layout–violation pairs used to fine-tune a domain-adapted DRC-VLM. This VLM provides rule-aware geometric guidance for local DRV repair, boosting state-of-the-art agents' solve rates by +12–25% (up to 97%) on 100 real sub-2 nm cases spanning enclosure, spacing, width, and color-spacing violations.

1 INTRODUCTION

Design Rule Checking (DRC) and Violation (DRV) correction have emerged as critical bottlenecks in achieving design closure at advanced process nodes. At sub-2 nm nodes, EUV lithography and Design-Technology Co-optimization (DTCO) have introduced hundreds of conditional, topology-dependent constraints (e.g., conditional enclosure, color-dependent spacing, discrete width quantization) whose legality depends jointly on neighboring geometries, multi-layer enclosure, and double-patterning color assignment. Commercial signoff checkers [1] only *detect* such violations, leaving engineers to manually interpret and fix each one. In-design DRC and detail-routing engines [2, 3] fix violations automatically but rely on hand-tuned geometric templates [4]; as rules grow increasingly conditional, this rule-by-rule approach cannot keep pace, leaving complex rules uncovered and local repairs prone to cascading new violations. A more flexible, automated approach to local Place-and-Route (P&R) DRV fixing is therefore needed.

Large Language Models (LLMs) offer a promising alternative: since design rules are textual, an LLM agent can interpret a rule constraint and propose a targeted fix without hard-coding it into a routing engine. Prior EDA work applies LLMs to text-centric tasks such as documentation retrieval [5], DRC checker code generation [6, 7], and script automation [8], but DRV fixing additionally requires visually grounding a rule in the local layout, a capability largely unexplored for vision-language models (VLMs). As Fig. 1 shows, off-the-shelf VLMs fail at this grounding: they hallucinate geometric features and lack the domain knowledge to map visual patterns onto rule semantics, causing agents that rely on them to misfix or reintroduce violations. No structured layout–violation dataset exists to teach this grounding, making fine-tuning-based domain adaptation impractical.

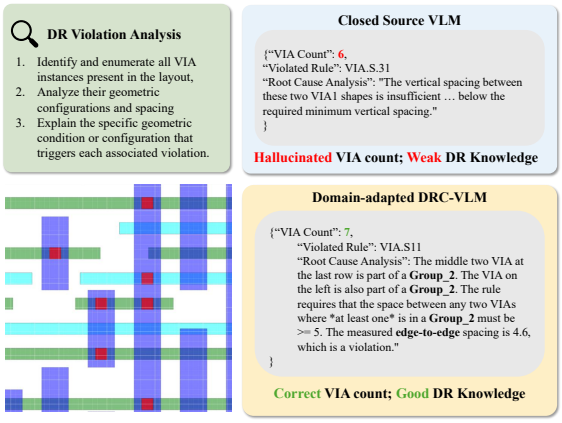

Figure 1: Off-the-shelf closed-source VLMs hallucinate geometric features (e.g., incorrect VIA count) and lack foundry-specific design rule knowledge, producing inaccurate violation analyses. Our domain-adapted VLM correctly perceives layout geometry and applies precise rule semantics, enabling reliable DRV root cause analysis.

Constructing such training data faces two barriers. First, *data scarcity*: production layouts are predominantly DRC-clean, real violations skew toward a narrow rule subset, and agentic synthesis from rule descriptions alone fails to reproduce realistic DRVs. Second, a *domain gap*: conditional design rules are foundry-confidential, so VLMs pretrained on public, natural-world images have never encountered nanometer-scale VLSI geometry and cannot generalize their spatial reasoning to it. Overcoming both barriers is the core challenge this work addresses.

Here, we address both barriers with a pipeline whose self-supervised generation stage produces diverse layout–violation pairs at scale from real designs, without hard-coding target rules into a routing/Engineering Change Order (ECO) engine or creating the DRV layouts manually. We then fine-tune an open-source VLM [9] on the resulting corpus, yielding a rule-aware, domain-adapted DRC-VLM that guides a state-of-the-art (SOTA) coding agent to fix local P&R DRVs more efficiently and effectively than unguided agents alone, addressing violations that commercial signoff checkers can only detect but not repair. The main contributions of this work are as follows:

*Equal contribution and corresponding authors: Chia-Tung Ho (chiatungh@nvidia.com), Haoyu Yang (haoyuy@nvidia.com), and Guanglei Zhou (gabzhou@nvidia.com).

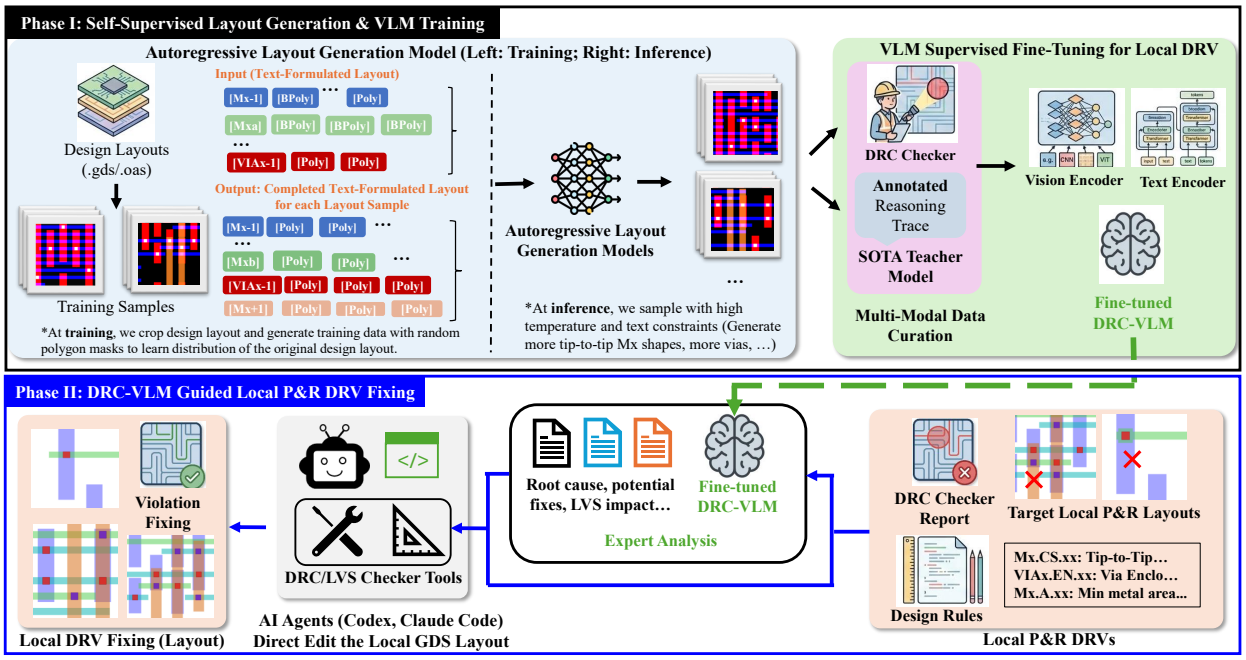

Figure 2: Proposed Framework. Phase I: Layouts are serialized into text to train an autoregressive model; text-conditioned sampling generates diverse layouts, annotated by industrial DRC and a teacher model for fine-tuning the DRC-VLM. Phase II: The DRC-VLM guides AI agents to edit the local GDS (boundary polygons may only be extended, not removed) with DRC/LVS checker-in-the-loop feedback.

- **Self-Supervised, Rule-Conditioned DRV Data Generation from Real Layouts.** We introduce *SCALE* and develop an LLM-based generation pipeline that reconstructs masked polygons from BEOL context alone without violation labels. At inference, natural-language constraints steer sampling toward violation-prone configurations, and an industrial signoff checker verifies the outputs, yielding diverse layout–violation pairs at scale from real designs.
- **Constraint-Aware Data Curation and Domain-Adapted DRV Fixing.** On top of the generated corpus, we curate multi-category supervision (layout descriptions, violation labels, and step-by-step reasoning traces) to fine-tune a domain-adapted DRC-VLM. This model provides dynamic, rule-conditioned guidance to SOTA coding agents for local P&R DRV repair.
- **Empirical Validation and SOTA Local P&R DRV Fixing at Sub-2 nm.** We conduct holistic studies on our data generation against discrete-diffusion and agentic baselines, demonstrating superior topology diversity and rule coverage. On 100 real sub-2 nm local P&R fixing scenarios, DRC-VLM guidance boosts SOTA agents' solve rates by +12–25% (up to 97%) across enclosure, width, spacing, color-spacing, and area violations.

2 Related Work

LLMs and VLMs for EDA. LLMs have been widely adopted for text- and code-centric EDA tasks, including document and log QA [5, 8, 10], chip design assistance [11], RTL generation [12–14], formal verification [15], and multi-task reasoning [16]. For design rule understanding, DRC-Coder [6] and Rule2DRC [7] combine VLMs and LLMs to generate DRC checker code from rule descriptions before any violation occurs, whereas to our knowledge no prior work uses a VLM to visually ground a flagged violation, diagnose its root cause, and guide an agent to repair it.

Spatial Reasoning in VLMs. General-purpose VLMs struggle with spatial reasoning [17, 18], especially for VLSI layouts governed by proprietary nanometer-precision rules, motivating domain-adapted training on DRC-annotated layout–violation data.

Layout Pattern Generation. Prior CNN [19], GAN [20, 21], token [22], and diffusion [23–27] methods advance DFM pattern diversity [28–31] but are limited to single-layer rasterized images. Our stage serializes multi-layer geometry into text, trains via self-supervised masked prediction from BEOL context, and steers generation with natural-language constraints at inference, capturing cross-layer diversity.

In summary, these gaps motivate our framework: a self-supervised layout-generation stage producing DRC-annotated multi-layer layout–violation pairs, coupled with VLM fine-tuning that provides actionable guidance for local P&R DRV fixing.

3 Methodology

We propose a two-phase framework (Fig. 2) that aligns VLMs with VLSI geometry to enable automated local P&R DRV fixing. Section 3.1 describes our self-supervised data generation pipeline and VLM supervised fine-tuning for a domain-adapted DRC model (DRC-VLM); Section 3.2 describes the DRC-VLM-guided agentic framework for local P&R DRV fixing; and Section 3.3 discusses why the two phases use different model modalities.

3.1 Phase I: Self-Supervised Layout Generation & VLM Supervised Fine-Tuning

Phase I is designed to close the data-scarcity and domain-gap barriers in two steps. Section 3.1.1 trains an autoregressive model to generate diverse, DRC-annotated layout–violation pairs from real designs. Section 3.1.2 curates multi-modal supervision from this corpus and uses it to fine-tune a domain-adapted DRC-VLM.

3.1.1 Autoregressive Layout Generation Model. We design the pipeline to operate on production GDS files through four stages: (1) tile the design into crops and rasterize each layer; (2) extract connected components and serialize into a text representation; (3) generate target-layer `[Poly]` sequences conditioned on context and optional text constraints; (4) decode generated tokens into physical polygons, insert into the original layout, and run industrial DRC for rule annotation. This produces a DRC-annotated, rule-labeled corpus anchored in real routed-layout topology.

Text-Formulated Layout Representation. We propose a text-formulated, multi-layer layout representation (Fig. 2) as the key enabler. Inspired by LayouTransformer [22], we serialize each layout crop into structured tokens preserving layer identity, absolute position, and polygon extent. Each rectangle is encoded as a directional edge walk from its top-left vertex (x, y) via cardinal deltas ($\rightarrow$ width, $\downarrow$ height, $\leftarrow$ width, $\uparrow$ height). The representation uses:

- **Layer headers** `[M1]`, `[M2]`, `[V1]` etc.: delimit each metal or via layer section.
- **`[Poly] ... [\Poly]`**: a complete polygon fully contained within the crop window.
- **`[BPoly] ... [\BPoly]`** (Boundary Polygon): a shape clipped by the crop boundary, used only as context, never as a generation target.
- **Coordinate tokens** (x, y): absolute top-left vertex position in layout units.
- **Direction deltas** $\rightarrow w, \downarrow h, \leftarrow w, \uparrow h$: cardinal steps tracing the rectangle boundary.

This encoding is compatible with standard language-model training while preserving geometry needed for signoff-quality DRC annotation. We introduce the boundary-aware `[BPoly]` token to separate complete targets from context-only boundary shapes, ensuring the model does not treat clipped boundary shapes as valid generation targets or hallucinate geometry at crop edges.

Self-Supervised Training & Controllable Sampling. Let $X = \mathrm{Tok}(\mathcal{L}_{\mathrm{ctx}})$ denote the serialized context (both `[Poly]` and `[BPoly]`) and $Y = (y_1, \ldots, y_T) = \mathrm{Tok}(\mathcal{L}_{\mathrm{gen}})$ the generated layer sequence (only `[Poly]`). During training, the model learns to reconstruct masked polygons from context alone:

$$p_\theta(Y \mid X) = \prod_{t=1}^{T} p_\theta(y_t \mid y_{<t}, X).$$

Training minimizes teacher-forced negative log-likelihood:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\sum_{(X,Y)\in\mathcal{D}} \sum_{t=1}^{T} \log p_\theta(y_t \mid y_{<t}, X).$$

At inference, we augment the context with an optional natural-language constraint C describing the target rule family (e.g., "generate more tip-to-tip Mx shapes" or "target VIA spacing violations") and sample tokens with temperature τ:

$$y_t \sim \mathrm{softmax}\left(\frac{\mathbf{z}_t}{\tau}\right), \quad \mathbf{z}_t = f_\theta(y_{<t}, X, C),$$

where larger τ increases diversity and the likelihood of rare rule-triggering configurations. Constraint following relies on the instruction-following capability of the pretrained language model; layout fine-tuning teaches only the structured polygon representation, leaving the model's ability to condition on natural-language prompts intact. We decode sampled sequences back into polygons and insert them into the original layout; signoff DRC checker then annotates each candidate with the violated rule and violation location.

Table 1: Training data composition by generation method and layer type in Section 4.2. The DRC analysis task is augmented with reasoning source for the proposed in Table 5. The description task includes via count, via location, and metal/via bounding box extraction tasks.

Method	Layer Type	DRC Analysis (w/wo Reasoning)	Description Task	Total
Proposed	VIA	7,606	9,255	16,861
	Metal	8,946	8,308	17,254
Script	VIA	6,244	9,255	15,499
	Metal	5,583	11,166	16,749

3.1.2 VLM Supervised Fine-Tuning for Local DRV. We run a DRC checker [1] on the generated layout crops from Section 3.1.1 to obtain ground-truth violation labels and locations, which we then curate into two complementary multi-modality supervision categories (Table 1):

- **DRC Analysis Task:** A teacher model (Gemini-3-Pro) receives each layout crop image together with its DRC violation report and generates a reasoning trace that identifies relevant layout features, measures dimensions, compares against rule thresholds, and concludes with the violated rule. We apply multi-stage filtering on rule coverage, violation polygon location, and direction/spacing consistency, removing 13.40% of traces with image input (Table 2).
- **Description Task:** We pair each crop with programmatically extracted metal shape and VIA center coordinates in a structured JSON response, teaching the model to perceive, count, and localize geometric features.

We combine both data categories into a unified, shuffled training mixture for supervised fine-tuning of Qwen3-VL-8B [9]. Each sample pairs a layout crop and rule description with either a structured JSON prediction or a reasoning trace followed by the prediction. We train the model with autoregressive cross-entropy loss over target tokens (prompt and image tokens masked), jointly optimizing layout perception, violation classification, and diagnostic reasoning.

3.2 Phase II: VLM-Guided Local P&R DRV Fixing

We employ the domain-adapted DRC-VLM as an expert diagnostic front-end to a SOTA AI agent for local P&R DRV repair (Fig. 2). We input the DRC-VLM with a rendered image of the local layout region, the DRC error report, and the relevant design-rule description; it produces a structured expert analysis: root-cause identification, the specific polygons involved with their coordinates, and a ranked list of top-k potential fix actions with predicted connectivity impact.

This analysis is then provided to a SOTA AI coding agent equipped with DRC and LVS checker tools. The agent directly edits the local GDS without a predefined action space: it may move, resize, add, or remove interior polygons as needed, while boundary polygons may only be extended or supplemented, not removed, to preserve connectivity with the surrounding design. DRC and LVS checkers are invoked after every edit to verify rule legality and preserve circuit connectivity, forming a checker-in-the-loop repair cycle. Unlike template-based engines that narrow the search space through static, pre-encoded correction patterns, the DRC-VLM provides dynamic, rule-conditioned guidance grounded in domain knowledge of complex conditional design rules, enabling the agent to adapt its fixes to each violation's geometric and topological context.

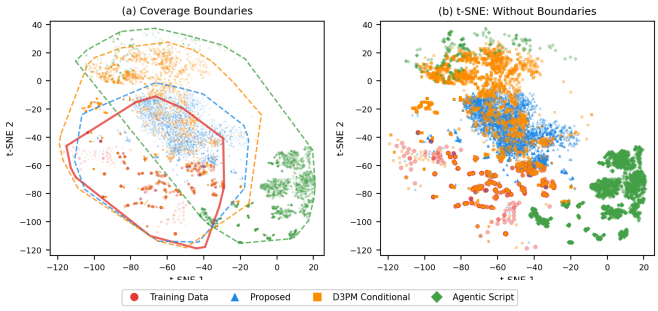

Figure 3: t-SNE Diversity Analysis. t-SNE projection comparing synthetic datasets against the ground truth (red). Both the trained proposed (autoregressive) and D3PM models achieve high-fidelity, in distribution coverage. In contrast, the heuristic script-based baseline fails to capture the realistic geometric distribution.

3.3 Discussion: Choosing the Model Modality

We use a text LLM for layout augmentation because our structured representation enables diverse outputs that decode directly into editable polygons, whereas image-based generation would require vectorization, grid snapping, layer assignment, and filtering before signoff. DRC reasoning, however, demands understanding of spatial relations such as spacing, enclosure, and multi-layer interactions that layout images expose directly but polygon sequences flatten into long token streams. To validate this, we compare the filtering rates of reasoning traces generated from text-represented layouts versus image inputs on the DRC Analysis Task (Table 2). Text input leads to a 30.66% overall filtering rate versus only 13.40% for image input, with the gap concentrated at the location stage (21.60% vs. 3.50%): the text representation enables correct rule identification and logically valid reasoning, yet lacks the spatial context to distinguish the exact violating polygon from geometrically identical neighbors. These results confirm complementary roles for the two modalities, text for controllable layout generation and vision for spatial DRC analysis, and demonstrate that visual input is irreplaceable for polygon-level localization in dense layouts.

Table 2: Filtering rate of reasoning traces at each stage for text-represented layout versus image input on the DRC Analysis Task.

	Filter Stage			
	DRV Rule	Location	Dir./Measure	Total
Text Rep.	4.03%	21.60%	5.03%	30.66%
Image	9.60%	3.50%	0.30%	13.40%

4 Experimental Results

Our evaluation covers layout data generation quality, DRV detection and understanding, and local DRV fixing at an advanced technology node. Both the auto-regressive layout generation model (Qwen2.5-3B-Instruct) and DRC-VLM (Qwen3-VL-8B) are fine-tuned using AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$) with a cosine learning-rate schedule on 4× A100-80GB GPUs. For the DRV rule set, we select a subset of local P&R relevant DRC rules from the thousands available at the sub-2 nm technology node, covering spacing, enclosure, color spacing, group via spacing, and width constraints.

4.1 DRV Layout Data Generation Evaluation

We evaluate the quality of the generated DRC violation layout dataset. The auto-regressive layout generation model is trained on 1,000 real design layout crops using random polygon masking and reconstruction from BEOL context alone, without constraints or violation labels. During inference, we supply rule-specific or geometric constraints (e.g., generate more VIA groups, more tip-to-tip Mx shapes) and sample under high temperature to steer generation toward violation-prone configurations. Each method generates ∼15K samples; after filtering out-of-scope and near-duplicate DRVs, these are used for the downstream VLM training comparison in Section 4.2.

Table 3: Pattern diversity comparison across generation methods. H is the Shannon entropy of the joint complexity distribution $P(c_x, c_y)$. Higher H indicates more diverse layout patterns. DRC columns report design rule violation metrics for VIA and Metal layers separately. DRC columns report unique validated triggered rules by category (VIA / Metal). Abbreviations: W = Width, W+A = Width + Area, Spc = Spacing, G-Spc = Group-conditioned Spacing, EN = Enclosure, Grp = Group, R = Routing, CS = Colored Spacing.

Method	H ↑	Unique Top.	Validated VIA / Metal DRV Type					
			W / W+A	Spc	EN	G-Spc / R	Grp / CS	Total
Training Data	2.93	93	–	–	–	–	–	–
D3PM	4.61	236	1 / 5	5 / 4	7 / 2	3 / 1	2 / 1	18 / 13
Script	3.94	133	2 / 7	3 / 2	1 / 2	2 / 1	2 / 0	10 / 12
Proposed	5.07	289	1 / 13	13 / 4	6 / 4	7 / 10	3 / 3	29 / 34

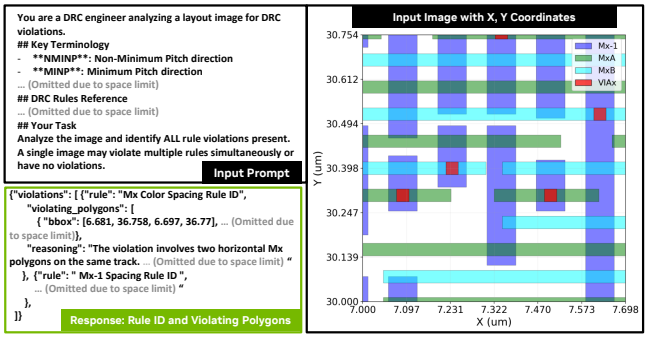

Figure 4: An example of the VLM detection task used for SFT. Each SFT training pair consists of a layout image, input prompt, and response. Axis coordinates are scaled for confidentiality.

Baselines. We compare against two baselines: a discrete diffusion model (D3PM) [25] trained on the same data, and an agentic script-based method in which a SOTA coding agent [32] is provided with DRC rule descriptions and layer parameters (routing directions, pitch, and width) to generate Python scripts. These scripts are then executed to produce synthetic DRV layout–violation pairs.

Evaluation Metrics and Results. We measure generation quality using layout entropy (H) [19, 22], unique topologies, and validated triggered design rules, as shown in Table 3. Following [22], H is the Shannon entropy over scan-line counts (c_x, c_y):

$$H = -\sum_i \sum_j P(c_{x_i}, c_{y_j}) \log P(c_{x_i}, c_{y_j}).$$

Unique topologies are distinct (c_x, c_y) pairs; rule coverage counts validated triggered rules after filtering out-of-scope and near-duplicate DRVs. Our method achieves the highest entropy ($H = 5.07$), the most unique layouts (289), and the broadest validated rule coverage, triggering 29 VIA and 34 metal rule types. The t-SNE visualization in Fig. 3, further confirms that the auto-regressive generator and D3PM match the geometric distribution of real routed layouts, while the agentic script baseline produces simpler patterns that deviate from real designs.

Table 4: DRV detection testing dataset rule frequency by category.

VIA / Metal Category	VIA (100 samples)	Metal (100 samples)
W / W+A	5	180
Spc	177	108
EN	3	81
G-Spc / R	308	25
Grp / CS	30	15
G-Spc / EN	308	81
Total triggers	523	409
Avg./sample	5.23	4.09

Table 5: DRV detection results averaged across Metals and VIAs. Rule F1 and Loc F1 measure rule identification and rule-gated localization accuracy; Comb F1 is their harmonic mean. All values are pass@k over 10 trials. † Ablations isolating data source and reasoning contributions.

	Rule F1 (%)			Loc F1 (%)			Comb F1 (%)		
Model	@1	@3	@5	@1	@3	@5	@1	@3	@5
Commercial & Open-Source Baselines									
Gemini-3-Pro	34.9	46.2	49.6	14.4	21.5	24.6	20.4	29.3	32.8
GPT-5.2	24.0	35.3	40.2	7.4	14.0	17.5	10.7	19.3	23.6
Qwen3-Base	19.4	38.6	47.4	3.7	8.8	12.4	6.2	14.3	19.4
Qwen3-VL-8B-SFT									
Mixed-NoR-Script†	37.0	48.5	54.0	4.8	10.5	14.0	7.6	15.9	20.7
Mixed-NoR-AutoReg†	17.3	49.5	55.7	15.1	25.9	31.2	20.3	33.5	39.4
DRC-VLM (Proposed)	**43.8**	**57.4**	**61.6**	**23.5**	**31.9**	**35.6**	**29.5**	**39.9**	**44.2**

4.2 VLM Supervised Fine-Tuning Result

The DRV detection task is designed to strengthen the model's understanding of layout geometry and design-rule constraints as shown in Fig. 4. Unlike simple object localization, this task requires the VLM to reason about spatial relationships between polygons, measure rule-relevant distances, and identify violation-causing regions. This provides high-value supervision for downstream layout tasks, which are diverse, scenario-dependent, and difficult to exhaustively construct as training data.

Training Data Composition. Table 1 summarizes the training data. After filtering DRC violations and removing near-duplicate tasks, our proposed pipeline yields 16,861 VIA and 17,254 metal samples spanning DRC analysis (with both reasoning and non-reasoning supervision) and layout description tasks (via counting, localization, and bounding-box extraction). As a baseline, we collect 15,499 VIA and 16,749 metal samples using script-based generation with the same filtering procedures.

DRV Detection Testing Dataset. We construct a 200-sample test set (100 VIA, 100 metal) from real layout crops of a *different IP* than the training data; both share the same sub-2 nm rule deck but have disjoint routing patterns. Each sample may contain multiple co-occurring violations; as shown in Table 4, the set contains 523 VIA-rule and 409 metal-rule triggers (avg. 5.23 and 4.09 per sample) spanning categories mentioned in Table 3.

Evaluation Metrics. Each response is scored at two levels: *Rule identification* via set-level F1 between predicted and ground-truth violated-rule sets, and *Localization*, gated by rule correctness—a prediction $(\hat{r}, \hat{p})$ is a TP only if $\hat{r}$ matches a ground-truth rule and $\hat{p}$ matches its ground-truth location, else FP:

$$\mathrm{Loc}(\hat{r}, \hat{p}) = \begin{cases} \mathrm{TP}, & \hat{r} \in \mathcal{R}_{\mathrm{GT}} \text{ and } \hat{p} \leftrightarrow \mathcal{P}_{\mathrm{GT}}(\hat{r}), \\ \mathrm{FP}, & \text{otherwise}, \end{cases}$$

Unmatched ground-truth instances count as FN. Each sample is evaluated over $n = 10$ trials and aggregated via the pass@k estimator [33] for Rule F1 and Loc F1; Comb F1 is their harmonic mean.

Baselines. We compare against strong SOTA VLM baselines, including Gemini-3-Pro, GPT-5.2, and Qwen3-VL-8B-Base, using identical rule descriptions and layout images as input.

Evaluation Result. Table 5 shows that higher-quality generated training data directly improves VLM performance on DRV detection: our method achieves the best overall pass@5 Comb F1 of 44.2, a 24.8-point gain over the base model, versus 32.8 for Gemini-3-Pro. The main bottleneck for general VLMs is localization rather than rule recall: Gemini-3-Pro reaches only 24.6 Loc F1 at pass@5, GPT-5.2 reaches 17.5, and Qwen3-VL-8B-Base reaches 12.4, despite receiving the same rule descriptions and layout images. The gain from our data is also largest in localization—the script-based baseline (Mixed-NoR-Script) attains high Rule F1 but poor Loc F1, showing it exposes rule labels without the geometric fidelity needed for spatial grounding—confirming that realistic generated layouts are critical for accurate violation localization. This aligns with the filtering analysis in Section 3.3: even Gemini-3-Pro, our teacher model, achieves only 24.6 Loc F1—it can generate reasoning traces but cannot reliably localize violating polygons, making fine-tuning on our DRC-annotated layout data essential for Phase II guidance. As a result, our domain-adapted DRC-VLM is used next to guide local P&R DRV fixing.

4.3 Local DRC Fixing Experiment

We demonstrate that DRC-VLM can significantly improve SOTA AI agents on fixing local P&R DRC violations at sub-2nm technology node. Given that general VLMs fail at the prerequisite localization task (Table 5), we employ our domain-adapted DRC-VLM as the expert front-end. For each DRV, we extract a local GDS region surrounding the violation site as the repair context. Interior polygons may be freely edited, while boundary polygons may only be extended or supplemented, not removed, to preserve surrounding connectivity. Given the target DRV error report, the local layout netlist, and a rendered image of the region, the fine-tuned DRC-VLM analyzes the violation and proposes an expert analysis including top-k potential fixes, where $k = 3$. This expert analysis is then supplied to a SOTA coding agent equipped with DRC and LVS checker tools to fix the local DRVs. We run the agent fix once per problem for both proposed method and baselines for a fair comparison.

Local P&R DRV Dataset. From an unseen block-level IP design that was DRC-clean under a prior developing rule deck but exhibits numerous errors under the latest rule deck at sub-2nm, we curate 100 local P&R DRV cases by selecting representative error regions and filtering duplicates. The dataset spans five categories (EN: 39, Width: 27, Spacing: 21, Color Spacing: 10, Area: 3) covering same-layer and cross-layer interactions, making it representative of local Engineering Change Order (ECO) scenarios where geometric legality, connectivity, and advanced-node context must all be satisfied. Curation prioritizes hard-to-fix conditional rules (enclosure, color spacing); the category distribution matches the actual P&R error distribution. Each instance requires up to 1,800 s of agent compute with iterative checker feedback, making large-scale evaluation costly; the Area category (3 cases) is included for completeness but is too small to draw per-category conclusions.

Baselines. We compare against two SOTA coding agents [32, 34] *without* the proposed expert analysis from DRC-VLM. Claude Code uses Sonnet 4.6 [35] as its core model, and Codex uses GPT-5.4 [36]. Each agent is provided with DRC rule descriptions, metal pitch and

Table 6: Local DRV fixing results and token cost per case. Baseline agents (Claude Code, Codex) are SOTA coding agents with access to DRC rule descriptions, raw DRC reports, rendered layout images, and DRC/LVS checker tools. Proposed methods augment each agent with expert analysis from DRC-VLM, the strongest model selected from Section 4.2. Solved Count (SC) and Solve Rate (SR) are reported per DRV rule category and in total. Absolute SR improvement is shown below proposed entries (green). Token cost in thousands (K): Codex reports tokens_used; Claude Code reports output + cache-creation tokens (excluding cache-read input). Relative reduction below (green = fewer, red = more). Bold = best.

Method	DRV Rule Category SC / SR (%)					Total	Avg. Tokens / Case (K)					
	EN	Width	Spacing	Color Spacing	Area	SC / SR (%)	EN	Width	Spacing	Color Spacing	Area	Overall
Baselines												
Claude Code	11 (28.2%)	26 (96.3%)	20 (95.2%)	5 (50.0%)	1 (33.3%)	63 (63.0%)	300.8	82.0	96.2	44.4	187.7	169.7
Codex	33 (84.6%)	26 (96.3%)	18 (85.7%)	5 (50.0%)	**3 (100.0%)**	85 (85.0%)	96.0	82.3	92.2	**28.4**	**130.3**	85.8
Proposed												
DRC-VLM + Claude Code	32 (82.1%)	**27 (100.0%)**	**21 (100.0%)**	6 (60.0%)	2 (66.7%)	88 (88.0%)	214.3	**74.4**	**83.0**	**21.5**	162.4	128.2
	+53.8%	+3.7%	+4.8%	+10.0%	+33.3%	+25.0%	(28.7%)	(9.3%)	(13.7%)	(51.5%)	(13.5%)	(24.5%)
DRC-VLM + Codex	**39 (100.0%)**	**27 (100.0%)**	**21 (100.0%)**	**7 (70.0%)**	**3 (100.0%)**	**97 (97.0%)**	**91.8**	83.5	88.9	29.8	**130.3**	**83.9**
	+15.4%	+3.7%	+14.3%	+20.0%	+0.0%	+12.0%	(4.6%)	(-1.4%)	(3.6%)	(-4.9%)	(0.0%)	(2.2%)

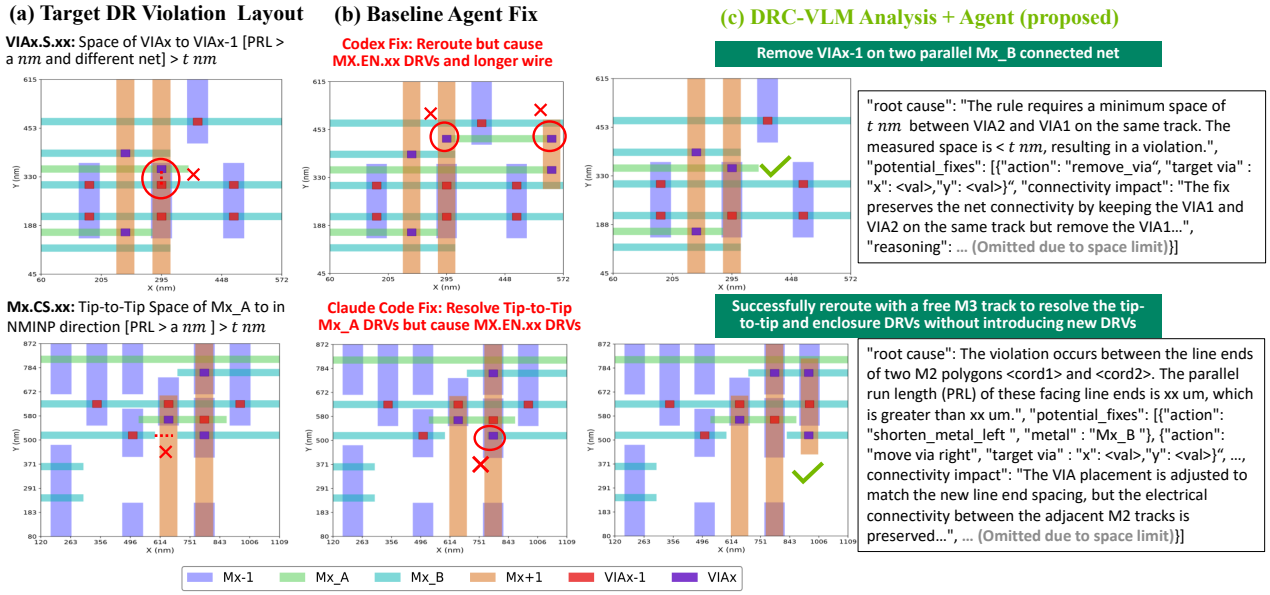

Figure 5: Examples of DRC fix approaches for VIA spacing (top) and metal tip-to-tip (bottom) rules. (a) Target layouts with the original violations. (b) Baseline agent fixes introduce new violations while try to fix the target DRV. (c) Proposed DRC-VLM analysis with structured root-cause and connectivity-aware actions yielding clean fixes. PRL=Parallel Running Length. Axis coordinates are scaled for confidentiality.

preferred routing direction parameters, the raw DRC report, the input GDS, and the rendered PNG layout image in a shared working directory. Both agents have access to DRC and LVS signoff checker tools, with an 1800 s time limit imposed per DRV fixing instance. These constitute strong baselines: as SOTA coding agents, they can programmatically analyze DRC reports, restructure rule and layout information, inspect rendered images, and iteratively repair the GDS with checker-in-the-loop feedback.

Evaluation Metrics. A case is *solved* if the target DRV is fixed without introducing new DRVs in the region *and* the design passes LVS. We report the solve rate per category and overall.

Experiment Results. Table 6 reports local DRV fixing results (left) and token cost (right) across five rule categories on the 100-case dataset. DRC-VLM-assisted methods consistently outperform the coding-agent baselines: Claude Code's overall solve rate rises from 63.0% to 88.0%, and Codex's from 85.0% to 97.0%. Without knowledge of proprietary conditional rules, baseline agents struggle to resolve DRVs without introducing new violations; the DRC-VLM provides structured root-cause analysis, potential fixes, and connectivity impact, enabling successful repair without cascading errors. As Figure 5 illustrates, baseline agents attempt plausible but rule-unaware edits that trigger new violations, whereas DRC-VLM-guided agents apply rule-compliant corrections, confirming that explicit geometric reasoning is a stronger prior than iterative trial-and-error. The remaining unsolved cases concentrate in the Color Spacing category, where tight spacing interactions risk triggering cascading violations that require net-level rerouting beyond the local repair scope. For token cost, DRC-VLM guidance reduces average tokens by 24.5% for Claude Code and 2.2% for Codex, with the largest savings in Color Spacing and EN categories, reflecting fewer exploratory iterations when the agent receives rule-aware analysis upfront.

5 Conclusion and Future Works

Local P&R DRV fixing at advanced nodes is bottlenecked by scarce layout–violation training data and the domain gap between general-purpose VLMs and proprietary design rules. We presented *SCALE*, a framework whose self-supervised generation stage produces diverse, DRC-annotated layout–violation pairs from real BEOL context without violation labels during training. The domain-adapted DRC-VLM provides rule-aware geometric analysis that boosts SOTA

agents' solve rates by +12–25% (up to 97%) on 100 real sub-2 nm cases, demonstrating that scalable, data-driven domain adaptation can bridge general-purpose VLMs to advanced-node DRC reasoning without rule-by-rule engineering. Future work will extend to cascading multi-step DRV fixing and net-level rerouting beyond the local repair scope, as well as adapting the self-supervised generation–fine-tuning paradigm to other geometry-intensive EDA tasks where proprietary rules and data scarcity pose similar challenges.

Acknowledgment

We gratefully thank Chen Cui, Joosung Yoon, Nithin Kumar Mali, George Kokai, Ting Ku, Yan He, Georgios Kalogerakis, and Tom Gray for their helpful discussions and support.

References

[1] Siemens EDA, "Calibre nmDRC hierarchical design rule checking." https://eda.sw.siemens.com/en-US/ic/calibre-design/drc/, 2024.

[2] Cadence Design Systems, "Innovus implementation system." https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/soc-implementation-and-floorplanning/innovus-implementation-system.html, 2024.

[3] Synopsys, Inc., "IC Compiler II place and route solution." https://www.synopsys.com/implementation-and-signoff/physical-implementation/ic-compiler.html, 2024.

[4] J. Yang, N. Rodriguez, O. Omedes, F. Gennari, Y.-C. Lai, and V. Mankad, "DRCPlus in a router: automatic elimination of lithography hotspots using 2D pattern detection and correction," in *Design for Manufacturability through Design-Process Integration IV* (M. L. Rieger and J. Thiele, eds.), vol. 7641, p. 76410Q, International Society for Optics and Photonics, SPIE, 2010.

[5] A. Kaintura, P. R, S. S. Luar, and I. I. Almeida, "Orassistant: A custom rag-based conversational assistant for openroad," 2024.

[6] C.-C. Chang, C.-T. Ho, Y. Li, Y. Chen, and H. Ren, "Drc-coder: Automated drc checker code generation using llm autonomous agent," in *Proceedings of the 2025 International Symposium on Physical Design*, ISPD '25, (New York, NY, USA), p. 143–151, Association for Computing Machinery, 2025.

[7] J. Kim, J. Byun, D. Hwang, S.-J. Park, and H. O. Song, "Rule2drc: Benchmarking llm agents for drc script synthesis with execution-guided test generation," *arXiv preprint arXiv:2605.15669*, 2026.

[8] Y. Pu, Z. He, T. Qiu, H. Wu, and B. Yu, "Customized retrieval augmented generation and benchmarking for eda tool documentation qa," in *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '24, (New York, NY, USA), Association for Computing Machinery, 2024.

[9] J. Bai *et al.*, "Qwen3-VL: Enhanced vision-language model." https://github.com/QwenLM/Qwen3-VL, 2025.

[10] L. J. Wan, C.-T. Ho, R. Liang, C. Yu, D. Chen, and H. Ren, "Schemacoder: Automatic log schema extraction coder with residual q-tree boosting," *arXiv preprint arXiv:2508.18554*, 2025.

[11] M. Liu, T.-D. Ene, R. Kirby, C. Cheng, N. Pinckney, R. Liang, J. Alben, H. Anand, S. Banerjee, I. Bayraktaroglu, *et al.*, "Chipnemo: Domain-adapted llms for chip design," *arXiv preprint arXiv:2311.00176*, 2023.

[12] C.-T. Ho, H. Ren, and B. Khailany, "Verilogcoder: Autonomous verilog coding agents with graph-based planning and abstract syntax tree (ast)-based waveform tracing tool," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, pp. 300–307, 2025.

[13] Z. Dong *et al.*, "ScaleRTL: Scaling LLMs with reasoning data and test-time compute for accurate RTL code generation," in *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, 2025. arXiv:2506.05566.

[14] M. Liu, W. Tsai, and H. Ren, "CraftRTL: High-quality synthetic data generation for Verilog code models with correct-by-construction non-textual representations and targeted code repair," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

[15] L. J. Wan, C.-T. Ho, Y. Bai, C. Yu, D. Chen, and H. Ren, "Fvrulelearner: Operator-level reasoning tree (op-tree)-based rules learning for formal verification," *arXiv preprint arXiv:2604.03245*, 2026.

[16] C.-T. Ho, J. Gong, H. Yang, A. Akkur, and H. Ren, "Polymath: Self-improving hierarchical workflow for multi-domain problem solving," in *Proceedings of the 2026 International Symposium on Physical Design*, pp. 121–130, 2026.

[17] B. Chen, Z. Xu, S. Kirmani, B. Ichter, D. Driess, P. Florence, D. Sadigh, L. Guibas, and F. Xia, "Spatialvlm: Endowing vision-language models with spatial reasoning capabilities," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14455–14465, 2024.

[18] J. Mayer, M. Ballout, S. Jassim, F. Nosrat Nezami, and E. Bruni, "iVISPAR – an interactive visual-spatial reasoning benchmark for VLMs," in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Suzhou, China), pp. 26757–26781, 2025.

[19] H. Yang, S. Li, W. Chen, P. Pathak, F. Gennari, Y.-C. Lai, and B. Yu, "Deepattern: Layout pattern generation with transforming convolutional auto-encoder," *IEEE Transactions on Semiconductor Manufacturing*, vol. 35, no. 1, pp. 67–77, 2022.

[20] X. Zhang, J. Shiely, and E. F. Young, "Layout pattern generation and legalization with generative learning models," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, 2020.

[21] X. Zhang, H. Yang, and E. F. Young, "Attentional transfer is all you need: Technology-aware layout pattern generation," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 169–174, 2021.

[22] L. Wen, Y. Zhu, L. Ye, G. Chen, B. Yu, J. Liu, and C. Xu, "Layoutransformer: Generating layout patterns with transformer via sequential pattern modeling," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '22, (New York, NY, USA), Association for Computing Machinery, 2022.

[23] Z. Wang, Y. Shen, W. Zhao, Y. Bai, G. Chen, F. Farnia, and B. Yu, "Diffpattern: Layout pattern generation via discrete diffusion," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2023.

[24] Q. Wang, X. Zhang, M. D. Wong, and E. F. Young, "Controlayout: Conditional diffusion for style-controllable and violation-fixable layout pattern generation," in *Proceedings of the Great Lakes Symposium on VLSI 2024*, GLSVLSI '24, (New York, NY, USA), p. 511–515, Association for Computing Machinery, 2024.

[25] G. Zhou, B. Korrapati, G. R. Reddy, C.-C. Chang, J. Pan, J. Hu, Y. Chen, and D. G. Thakurta, "Patternpaint: Practical layout pattern generation using diffusion-based inpainting," in *Proceedings of the 62nd Annual ACM/IEEE Design Automation Conference*, IEEE Press, 2025.

[26] G. Zhou, B. Korrapati, G. R. Reddy, J. Zhang, Y. Chen, and D. G. Thakurta, "Vario: Enhance pattern diversity with diffusion model," in *Proceedings of SPIE Advanced Lithography + Patterning*, vol. 13425, 2025.

[27] G. Zhou, B. Korrapati, G. R. Reddy, C.-C. Chang, J. Hu, Y. Chen, and D. G. Thakurta, "Diffusion-model-enhanced layout pattern generation for sub-3nm DFM," in *Proceedings of the 44th IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2025. Invited paper.

[28] G. R. Reddy, C. Xanthopoulos, and Y. Makris, "Enhanced hotspot detection through synthetic pattern generation and design of experiments," in *2018 IEEE 36th VLSI Test Symposium (VTS)*, pp. 1–6, 2018.

[29] G. R. Reddy, M.-M. Bidmeshki, and Y. Makris, "Viper: A versatile and intuitive pattern generator for early design space exploration," in *2019 IEEE International Test Conference (ITC)*, pp. 1–7, 2019.

[30] A. Kabeel, S. Kim, Y. G. Park, D. Kim, J. Kwan, S. Rizk, K. Madkour, M. Shafee, and J. Kim, "Design rule manual and drc code qualification flows empowered by high coverage synthetic layouts generation," in *DTCO and Computational Patterning II*, vol. 12495, pp. 415–428, SPIE, 2023.

[31] A. Hamouda, M. Bahnas, D. Schumacher, I. Graur, A. Chen, K. Madkour, H. Ali, J. Meiring, N. Lafferty, and C. McGinty, "Enhanced OPC recipe coverage and early hotspot detection through automated layout generation and analysis," in *Optical Microlithography XXX* (A. Erdmann and J. Kye, eds.), vol. 10147, p. 101470R, International Society for Optics and Photonics, SPIE, 2017.

[32] Anthropic, "Claude (computer software)." https://anthropic.com, 2026. Version Claude 3.5 Sonnet. Code generated on July 5, 2026.

[33] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[34] OpenAI, "Codex." Computer software, 2026.

[35] Anthropic, "Claude sonnet 4.6 interaction." Generative AI chat, 2026. Chat retrieved on July 9, 2026.

[36] OpenAI, "ChatGPT." https://chat.openai.com, 2026. [Large language model].