AgentDRC：面向空间感知的自动DRC校正多智能体算法

Pengju Liu、Nuo Xu、Winson Chen、Peijun Li、Yang (Katie) Zhao、Yu Cao、Caiwen Ding 明尼苏达大学双城分校，美国明尼苏达州明尼阿波利斯

arXiv:2611.31876v2 [cs.AR] 25 Nov 2026

摘要

设计规则检查（DRC）错误修复仍是物理设计中耗时的瓶颈，尽管电子设计自动化（EDA）优化技术不断进步，通常仍需人工干预。现有基于大语言模型的智能体方法表现不佳，原因包括上下文中存在冗余图形、对原始坐标的空间推理能力有限，以及采用低效的扁平布局表示。我们提出 AgentDRC，将迭代式 DRC 校正建模为多智能体算法，并引入一种空间感知验证与纠正算法，重建关键形状之间的结构化空间关系，并通过几何与语义约束检查验证修复结果。该系统协调四个专用智能体，以感知多层级、多模态几何特征，过滤无关图形，并迭代纠正违规项。实验表明，在具有挑战性的违规项上，AgentDRC 的 pass@1 最高达到 30.4%，pass@5 最高达到 55.7%，pass@5 比最佳基线高出 13.8%。论文录用后，我们将开源所有代码和基准测试任务。

关键词

设计规则检查修复，多智能体系统

1 引言

基于大语言模型（LLM）的智能体系统近期在代码生成 [5, 11, 23], 机器人学 [7, 22], 和数学推理 [8, 16], 方面取得了令人瞩目的成果，这些系统能够自主分解复杂任务、利用外部工具 [15], 并执行逐步推理。这些能力——工具使用、多步计划以及上下文理解——与物理设计任务的需求天然契合，因为此类任务要求与版图环境进行迭代交互。

相比之下，基于大语言模型的智能体在电子设计自动化（EDA）中的应用仍基本处于探索阶段，尤其是在设计规则检查（DRC）错误修复方面。这项任务需要修改布局几何图形，以满足工艺设计套件（PDK）约束 [6, 18]。现代电子设计自动化工具（例如 Cadence Innovus [3] 和西门子 Cali- bre [19], ）可以自动解决部分违规项。然而，许多 DRC错误仍需要人工干预，对于大规模设计而言，这一过程可能需要数周时间 [10, 17]。这些局限性凸显了对更有效的自动化设计规则检查修复方案的需求 [9]。

受此启发，我们探索采用基于 LLM 的智能体自动修复 DRC错误的潜力。然而，这也带来了若干挑战。**冗余且嘈杂的上下文信息（挑战 1）**。DRC错误周围的区域通常包含跨越多个图层的大量邻近图形，但其中大多数与该违规不存在任何约束关系。如果没有合理的过滤机制，这些无关图形会引入大量噪声并浪费智能体的推理能力，从而导致错误或次优的修复结果（见图 1）。**空间推理能力有限**

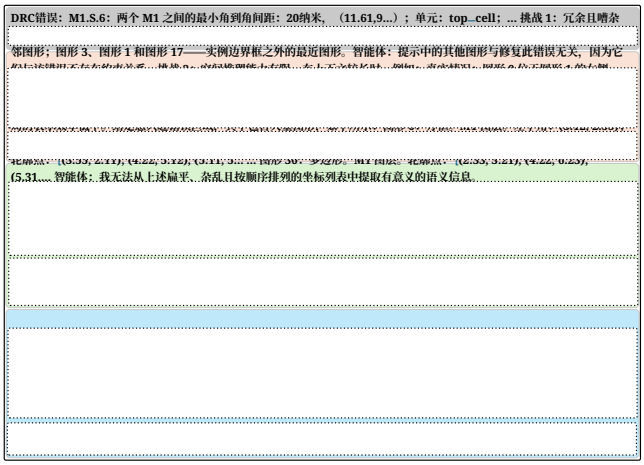

图1：基于智能体的设计规则检查错误理解与解决所面临挑战的示意图：（1）冗余的周围图形会分散智能体对约束相关图形的注意力；（2）从坐标数据推断空间关系和纠正操作的能力有限；（3）难以从扁平的顺序表示中提取有意义的几何或语义信息。

（挑战 2）。即使已经识别出相关图形，智能体仍难以从原始坐标数据中理解空间关系——相对方向、距离和包含关系——但这些信息对于确定纠正操作以及检测与邻近图形之间的冲突至关重要（见图 1）。**复杂几何布局的表示效率低下（挑战 3）**。用于描述布局图形的扁平、顺序化坐标数据无法捕捉结构模式或层次化上下文，从而为 DRC修正所需的高层次空间推理造成严重的信息瓶颈（见图 1）。

为应对这些挑战，我们提出了**AgentDRC**，该方法引入了一组用于空间感知 DRC 修正的算法，通过工具调用以及与布局的多轮交互，协调多个大语言模型智能体迭代诊断并修复违规项。据我们所知，AgentDRC 是首个利用大语言模型智能体自动修复 DRC错误的工作。我们的贡献如下：

▹ 我们将设计规则检查（DRC）修正形式化为一种迭代式多智能体算法（算法 1）：该算法将动作空间划分为几何子集和非几何子集，并根据情况将每个拟议动作路由至具备空间感知能力的几何验证路径或仅语义验证路径，从而为多轮修正过程提供有原则的协调机制。

▹ 为应对**挑战 1**，我们设计了一个上下文选择智能体，对多层级形状描述执行两步筛选：首先识别与违规直接相关的目标形状，然后根据层邻接关系、空间邻近性和约束相关性，选择最多 k 个约束相关邻近形状。这样可以将几何上下文有效地从数十个形状缩减为紧凑的关键子集，从而显著提升下游空间推理的准确性（§4.2）。

▹ 为应对**挑战 2**，我们设计了一种包含两个阶段的空间感知验证与纠正算法（算法 2）：空间感知智能体重构关键形状之间的结构化空间关系（距离、方向、包含关系），将原始坐标转换为可解释的表示；随后，检查器依据几何标准验证动作——方向有效性、幅度充分性和冲突检测——并自动纠正缺陷。对于非几何动作，该算法还通过额外的语义分支（§4.3）进一步验证图层完整性和工具一致性。

▹ 为应对**挑战 3**，我们设计了一种模仿人工工程师检查布局方式的多层级、多模态观察机制。几何上下文被分解为三个层级——直接相关、紧邻上下文和相邻形状——并在支持时同时以文本和视觉模态呈现，从而实现多个粒度层面的更丰富空间感知（§4.4）。

综合实验表明，我们提出的算法具有有效性和泛化能力。在一项具有挑战性的 DRC 基准测试中，与分别基于 GPT5-mini、Gemini-2.0 和 DeepSeek-V3.1 的最佳基线相比，其 pass@5 分别提升了 13.8%、9.3% 和 8.6%。

2 相关工作

设计规则检查（DRC）。流片前，电路布局必须符合工艺设计套件（PDK）中定义的物理设计规则。违反这些规则可能导致制造失败或功能错误。例如，在 ASAP7 PDK [6], 中，金属层 M1 的最小宽度为 7nm；任何窄于此宽度的 M1导线都违反可制造性约束。因此，实现无设计规则检查（DRC）违规的布局对于成功制造至关重要。然而，即使使用 Calibre [13], Innovus [3], 或开源 OpenROAD [1], 等先进电子设计自动化工具，仍会残留 DRC 违规，通常需要耗时的人工纠正。

单智能体系统。基于智能体的系统已成为实现通用智能行为的关键范式 [7, 8, 15, 22, 23]。现代人工智能模型——例如 OpenAI 的 GPT、Google 的 Gemini 和 Anthropic 的 Claude——采用基于智能体的架构，以增强推理和工具使用能力。在单智能体系统中，一个智能体在统一的循环中同时执行推理和动作，ReAct 框架 [22]便是一个例子。

多智能体系统。对于复杂任务，多智能体系统通过为多个智能体分配专业化角色，使其朝着共同目标协作，从而提供更强的可扩展性 [2, 14, 24]。典型架构采用规划器–执行器方案，由规划器分解任务，执行器则执行具体动作 [20, 21]。这种结构能够提升不同领域中的协调能力、推理深度和任务效率。

3 问题表述

3.1 初始状态 s_0

开始时，用户提供系统提示 s 和用户提示 q，其中包含 GDS布局 L 的路径以及已知 DRC 违规的初始集合 E_0，由此形成初始状态：

$$s_0 = \{s,\, q\}.$$

这对应于工程师使用传统电子设计自动化工具进行布局布线，随后执行数次自动设计规则检查修复迭代的场景。然而，即使经过多次运行，仍有若干设计规则检查违规未能解决，通常需要人工工程师进行人工纠正。

3.2 动作空间 $\mathcal{A}$

智能体通过一组离散的工具类动作与版图环境交互：

▹ query_with_pts：使用 DRC 报告中提供的错误标记坐标，访问相关图形。智能体还可以指定要检查的图层（例如 M1、V1 等）。▹ move_cell：通过设置新的 (x, y) 坐标来移动指定的单元实例。▹ change_shape：修改指定形状的几何形状，可以是特定单元实例内的形状，也可以是所有实例中的形状。▹ add_shape：在指定位置的特定图层上添加新形状。▹ run_drc：运行设计规则检查引擎，以验证当前版图状态是否符合设计规则，或是否引入了新的违规。

3.3 观测O

在每一步 t，智能体都会收到一个观测 $o_t \in \mathrm{O}$，其中 O 是环境返回的所有可观测信号的集合，包括用户提示、工具反馈、更新后的 DRC 报告以及几何上下文：

$$o_t = \begin{cases} s_0, & t = 0 \text{ (initial prompt)}, \\ \mathcal{T}(a_{t-1},\, \mathcal{L}), & t > 0 \text{ (tool feedback)}. \end{cases}$$

此处，$\mathrm{T}(\cdot)$ 是环境转移函数，它对当前布局 L 执行动作，并返回生成的观测（例如，查询到的图形、操作反馈或更新后的 DRC 报告）。智能体会维护完整的交互轨迹 τ，其初始值为 $\tau \leftarrow s_0$，并在每一步更新为 $\tau \leftarrow \tau \cup \{p_t,\ a_t,\ o_t\}$，其中 p_t 是推理计划，a_t 是步骤 t中采取的动作。

3.4 目标

智能体系统 π_θ 在每一步都会根据累积的轨迹确定推理计划p_t 和候选动作 a_t：$(p_t,\ a_t) = \pi_\theta(\tau)$，而环境则通过转移函数演化：$o_t = \mathrm{T}(a_t,\ \mathrm{L})$。目标是在经过 T次迭代后，将剩余的 DRC 违规数量降至最低：$\min_{\pi_\theta} |\mathrm{E}_T|$。其中，$\pi_\theta$ 概括了完整的多智能体架构，而 E_T 是经过 T 次迭代后返回的最终 DRC 违规集合。

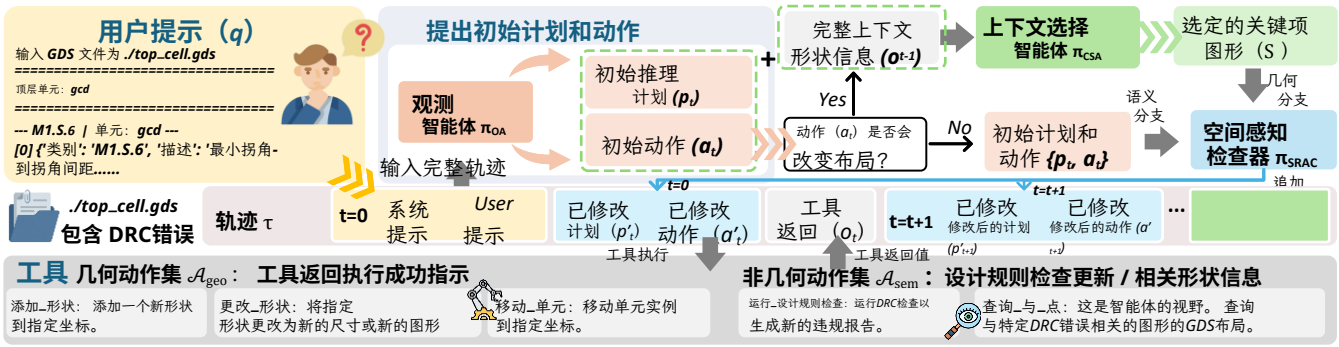

图2：所提出的用于自动修复DRC错误的多智能体算法 AgentDRC 概览。

4 修复DRC错误的多智能体算法

在这项工作中，我们提出了一种多智能体算法，如图 2 所示。系统从用户提示和 GDS布局开始，通过协作智能体进行迭代推理和动作规划。

4.1 概述

我们在算法 1 中形式化了 AgentDRC校正过程。该算法以系统提示 s、用户提示 q（其中包含 GDS 文件路径和已知的 DRC 违规集合 E_0）以及最大迭代预算 T 为输入，并输出违规项最少化后的布局 L^*。首先，我们将动作空间 A 划分为两个互不相交的子集：

$$\mathcal{A} = \underbrace{\mathcal{A}_{\text{geo}}}_{\text{移动_单元，更改_形状，添加_形状}} \cup \underbrace{\mathcal{A}_{\text{sem}}}_{\text{查询_带_点，运行_设计规则检查}}, \quad (1)$$

这驱动了自适应分支机制：几何动作会直接修改布局几何，因此触发上下文选择（§4.2）和空间感知验证（§4.3）的完整流程；非几何动作则用于查询上下文信息或验证当前违规集合，因此会转入纯语义验证。在每次迭代 t中，算法都会按照图 2 和算法 1 所示，依次经过四个阶段：

- **阶段 1**（第 5–6 行）：观察智能体πOA 使用完整的交互轨迹 τ，生成初始推理计划 p_t 以及候选动作 $a_t \in$ Ageo ∪ Asem。随后，分支指示变量 $\mathbb{I}$geo $= \mathbb{1}[a_t \in$ Ageo$]$ 决定采用哪条验证路径。
- **阶段 2**（第 9–10 行）：如果 $a_t \in$ Ageo，上下文选择智能体 πCSA（§4.2）会从先前工具反馈中筛选几何上下文 o_{t-1}（通常是通过查询_使用_pts返回的多层级形状描述），以生成紧凑的关键形状集 S。
- **阶段 3**（第 11–15 行）：空间感知检查器πSRAC 使用 S（几何分支）或 ∅（语义分支）进行验证和纠正（p_t，a_t），详见算法 2。
- **阶段 4**（第 17–20 行）：通过环境转移函数 T 执行已验证的动作 a_t，该函数返回一个观测o_t（例如工具反馈、更新后的图形或新的 DRC 报告）。轨迹更新为 $\tau \leftarrow \tau \cup \{p_t, a_t, o_t\}$，并且每当 $a_t =$run_drc 时，违规集合 E_t都会刷新。

算法 1 AgentDRC：迭代式多智能体 DRC 校正 要求：系统提示 s、用户提示 q（包含 GDS 路径和已知 DRC 违规 E_0）、最大迭代次数 T 确保：已修改的布局 L^*，且 DRC 违规最少 1：$\tau \leftarrow \{s, q\}$；$t \leftarrow 0$ 2：定义 $A_{geo} \triangleq$ {move_cell、change_shape、add_shape} ▷ ▷几何动作集 3：定义 $A_{sem} \triangleq$ {query_with_pts、run_drc} ▷ ▷非几何动作集 4：重复：5：▷ 阶段 1：观察智能体 6：$(p_t, a_t) \leftarrow \pi_{OA}(\tau)$ ▷ ▷ 生成推理 p_t和候选动作$a_t \in A_{geo} \cup A_{sem}$7：▷ 阶段 2：上下文选择智能体（$\pi_{CSA}$）8：$\mathbb{I}_{geo} \leftarrow \mathbb{1}[a_t \in A_{geo}]$ ▷ ▷ 分支指示变量 9：如果 $\mathbb{I}_{geo} = 1$ 则 10：$S \leftarrow \pi_{CSA}(o_{t-1}, p_t, a_t)$ ▷ ▷ §4.2；从先前工具反馈中选择关键形状 11：▷ 阶段 3：空间感知检查器（π_{SRAC}）— 几何分支 12：$(p'_t, a'_t) \leftarrow \pi_{SRAC}(p_t, a_t, S)$ ▷ ▷ 算法 2；空间智能体 + 检查器 13：否则 14：▷ 阶段 3：空间感知检查器（π_{SRAC}）— 语义分支 15：$(p'_t, a'_t) \leftarrow \pi_{SRAC}(p_t, a_t, \emptyset)$ ▷ ▷ 算法 2；纯语义验证与校正 16：结束如果 17：▷ 阶段 4：工具执行与状态转移 18：$o_t \leftarrow T(a'_t, L)$ ▷ ▷ 执行动作；环境返回观测o_t19：$\tau \leftarrow \tau \cup \{p'_t, a'_t, o_t\}$；$t \leftarrow t + 1$20：$E_t \leftarrow E_t|_{o_t}$ ▷ ▷ 违规集合在 $a'_t =$ run_drc 时更新 21：直到 $|E_t| = 0$ 或 $t \geq T$22：返回 $L^* \leftarrow$取 $\arg\min_{\mathcal{L} \in \{s_0, \ldots, s_t\}} |E(L)|$

当所有违规项均已解决（$|E_t| = 0$）或迭代预算 T 耗尽时，循环终止，并返回轨迹中遇到的剩余违规项最少的布局。

4.2 上下文选择智能体

设计规则检查修复面临的一个关键挑战是，目标 DRC标记周围的几何上下文通常包含跨越多个图层的数十个图形，其中大多数与当前违规无关。将所有这些图形输入下游推理会引入显著噪声，并降低智能体的空间推理准确性。

鉴于来自查询_的工具反馈 o_{t-1}（包含_个点，其中含有多层级形状 C $=\{s_1, \ldots\ldots, s_N\}$），这些形状按照第1—3层级（§4.4）组织；结合当前计划 p_t、拟议动作 a_t和邻居预算k，CSA（上下文选择智能体）生成一个紧凑的关键形状子集 S $\subseteq$C。选择过程分为两个步骤，旨在模拟人工工程师的推理过程：他们首先识别直接导致违规的形状，然后仅检查少量可能施加附加约束的邻近图形。

步骤 1：目标识别。 智能体接收一个结构化提示

$$\mathcal{P}_1 \leftarrow \langle\, C^{(\ell_1)},\ p_t,\ a_t,\ \text{violation description} \,\rangle,$$

其中包含第1层级形状 $C^{(\ell_1)}$（即与 DRC 错误标记发生物理接触的形状）。基于此，π_{CSA} 识别出与所报告违规直接相关的目标形状集合：

$$\mathcal{S}_{\text{target}} \leftarrow \pi_{\text{CSA}}(\mathcal{P}_1).$$

例如，对于两个 M1边缘之间的最小间距错误，智能体会选择其边缘被 DRC标记标出的两个 M1形状。

步骤 2：约束感知邻居选择。 随后，智能体会接收基于第2—3层级形状以及已识别目标构建的第二个提示：

$$\mathcal{P}_2 \leftarrow \langle\, C^{(\ell_2)} \cup C^{(\ell_3)},\ \mathcal{S}_{\text{target}},\ p_t,\ a_t \,\rangle,$$

其中增加了三个选择标准：(i) 层邻接关系——优先考虑相邻层上的形状（例如，当违规发生在 M1 上时，还要考虑 V0、V1 和 M2 上的形状）；(ii) 空间邻近性——优先选择距离目标形状最近的形状；以及 (iii)约束相关性——选择对计划中的修改施加间距、包围或重叠规则的形状。智能体最多选择 k 个最有可能影响所提修复正确性的邻居形状：

$$\mathcal{S}_{\text{nbr}} \leftarrow \pi_{\text{CSA}}(\mathcal{P}_2).$$

最终输出是两个集合的并集，

$$\mathcal{S} = \mathcal{S}_{\text{target}} \cup \mathcal{S}_{\text{nbr}},$$

该并集将传递给空间感知检查器。通过将上下文从 N 个形状缩减为通常为 $k + |\text{S}_{\text{target}}|$个形状的紧凑子集（我们在实验中设为 $k = 4$），CSA 在保留约束感知推理所需关键信息的同时，显著减少了上下文噪声。

4.3 空间感知验证与纠正

即使经过上下文选择，所提出的计划和动作仍可能包含空间推理错误——例如，将形状向错误方向移动；移动距离不足以满足间距规则；无意中与相邻形状引入新的违规项；或查询不完整的图层集合而遗漏关键约束。算法 2 通过双分支机制处理这些问题，对（p_t, a_t）进行验证，并在必要时进行纠正，以生成（p_t', a_t'）。**语义分支**（S $=\emptyset$, 第 2–6 行）。对于非几何动作（例如，使用_查询_个点、运行_设计规则检查），π_{SRAC} 会检查两个标准：

▷ v_{layer}：查询层涵盖所有相关邻居（例如，M1 违规 $\Rightarrow$ 还应检查 V0、V1、M2）；

算法 2空间感知检查器（π_{SRAC}）：验证与纠正

要求： 关键形状 S（或 $\emptyset$）、计划 p_t、动作 a_t
确保： 已验证（并可能已纠正）的计划和动作（p_t', a_t'）

1: **如果** $\mathcal{S} = \emptyset$ **则** ▷▷ 语义分支（非几何动作）
2: $v_{\text{layer}} \leftarrow$ 检查层完整性(a_t) ▷▷ 是否查询了所有相邻图层？
3: $v_{\text{tool}} \leftarrow$ 检查工具一致性(a_t) ▷▷ 更正 *GDS* 文件和参数？
4: **如果** $v_{\text{layer}} \wedge v_{\text{tool}}$ **则返回** (p_t, a_t) ▷▷ 原样传递
5: **否则返回**修正(p_t, a_t, 诊断信息) ▷▷ 自动修正并返回
6: **结束如果**
7: **否则** ▷▷ 几何分支（布局修改操作）
8: ▷ 第一阶段：空间关系构建（空间感知智能体）
9: R $\leftarrow \emptyset$
10: **对** 每一对 (s_i, s_j) $\in \mathcal{S} \times \mathcal{S}$ $i < j$**执行**
11: $d_{ij} \leftarrow$最小距离（边界框（s_i），边界框（s_j））
12: 方向$_{ij} \leftarrow$相对方向（边界框（s_i），边界框（s_j））
13: $cont_{ij} \leftarrow$ 包含(方框(s_i)，方框(s_j))
14: R $\leftarrow$R $\cup\{(s_i, s_j, (d_{ij}, \text{方向}_{ij}, cont_{ij}))\}$
15: **循环结束**
16: ▷ 第二阶段：对拟议动作进行约束验证（检查器）
17: $v_{\text{dir}} \leftarrow$方向验证($a_t$, $\mathcal{R}$) ▷▷ 移动方向能否解决违规？
18: $v_{\text{mag}} \leftarrow$值幅（$a_t$, R） ▷▷ 位移是否充分？
19: $v_{\text{cfl}} \leftarrow$检测冲突（$a_t$, R） ▷▷ 是否引入了新违规？
20: **如果** $v_{\text{dir}} \wedge v_{\text{mag}} \wedge \neg\, v_{\text{cfl}}$ **则返回** （p_t, a_t）
21: **否则返回**修正（p_t, a_t, $\mathcal{R}$, 诊断信息）
22: **结束 if**
23: **结束 if**

▷ v_{tool}：工具调用引用最近一次已修改的 GDS 文件，而不是原始文件，并且所有参数（坐标、实例索引、图层规格）格式正确且处于有效范围内。

如果 $v_{\text{layer}} \wedge v_{\text{tool}}$，则该对保持不变；否则将自动纠正。

几何分支（S $\neq \emptyset$, 第 7–16 行）。当动作修改布局几何时，检查器会分两个阶段运行。

第一阶段——空间关系构建。 空间感知智能体处理 S 并构建结构化关系集

$$\mathcal{R} = \left\{ (s_i, s_j, (d_{ij}, dir_{ij}, cont_{ij})) \;\middle|\; s_i, s_j \in \mathcal{S},\ i < j \right\},$$

其中，d_{ij} = MinDist(BBox(s_i), BBox(s_j)) 是最小包围盒距离，dir_{ij}是 相对方向（例如“形状 1 位于形状 0 的右上方”），而 $cont_{ij}$表示 包含关系（完全、部分或无）。这会将原始坐标转换为空间语义线索，如图 3 所示。

第二阶段 — 约束验证。给定 R，检查器会针对 a_t 评估三项几何标准：

▷ v_{dir}：移动方向能够消除违规（而非朝向冲突边缘）；▷ v_{mag}：位移满足最小间距规则；▷ v_{cfl}：不会与 S 中相邻的图形产生新的违规项。

如果是 $v_{\text{dir}} \wedge v_{\text{mag}} \wedge \neg\, v_{\text{cfl}}$，则返回原始配对；否则 π_{SRAC} 会使用 R 和诊断信息发出修正后的 (p_t', a_t')。

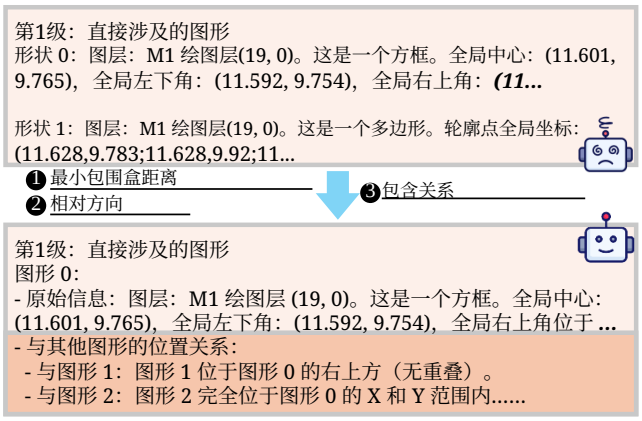

图 3：结构化空间关系。空间感知智能体将 DRC错误周围的原始几何坐标转换为结构化空间关系。

通过在语义分支和几何分支之间动态切换，该算法确保每项决策——无论是上下文查询还是物理布局修改——在逻辑上都保持一致，并充分考虑空间因素，从而在整个多轮修正过程中维持正确性。

4.4 智能体实现

本节介绍算法 1–2中所用智能体的实现细节。

观察智能体（π_{OA}）。OA 是每次迭代的入口，负责感知当前状态并提出初始计划 p_t 和动作 a_t。其输入因迭代而异：

- $t = 0$：系统提示和用户提示，其中包含 GDS 路径以及已知的 DRC 违规项 E_0。
- $t > 0$：完整的累积轨迹 τ，其中包含此前的所有计划、动作和工具观测 $o_{t'}$——包括由 query_with_pts 返回的几何上下文描述、操作成功指示（例如是否成功移动了单元实例），以及由 run_drc 生成的更新 DRC 报告。

为了感知每个 DRC错误周围复杂的空间上下文，我们设计了一种多层级、多模态表示，模拟人工工程师目视检查布局的方式。如图 4 所示，来自查询_和_pts 的几何上下文被分解为三个层级：

- 第1级：与 DRC 错误标记物理接触的图形——对应于工程师双击错误时的放大视图。
- 第2级：位于包含第1级形状的单元实例的组合边界框内的图形——模拟针对邻近约束的稍微缩小视图。
- 第3级：位于组合边界框外侧的图形——对于评估单元移动过程中潜在的实例间冲突至关重要。

智能体可以动态调整观测范围（例如，从最近的形状调整到第 m近的形状），从而自适应地压缩几何上下文。若基础模型支持多模态输入（例如 GPT-5-mini 和 Gemini-2.0），则会同时提供文本描述和视觉

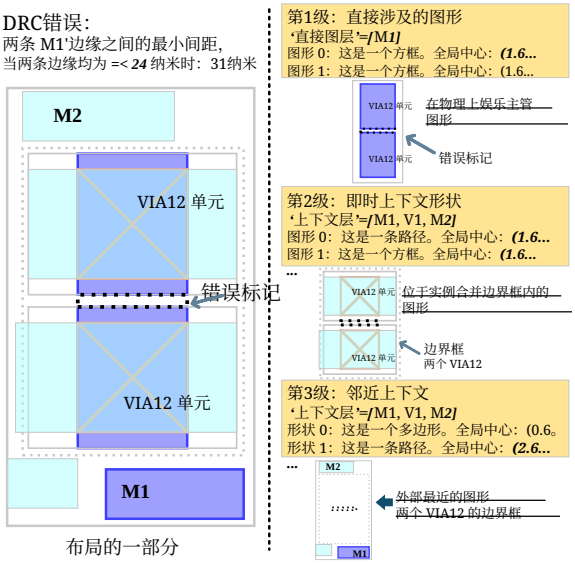

图 4：上下文图形的多层级、多模态表示。（左）DRC错误及其受影响布局区域示例。（右）围绕错误标记构建的三个上下文层级。

渲染，以实现更丰富的空间理解。

基于这种多层级、多模态上下文和完整轨迹 τ，OA 输出初始推理计划 p_t（例如："为修复间距违规，我将使用移动_单元，将该实例移离冲突边缘"）以及相应的候选动作 a_t（例如，移动_单元（新_x：6.09，新_y：7.1））。

上下文选择智能体（π_{CSA}）。如 §4.2 所述，CSA 被实例化为一个大语言模型智能体，接收包含多层级文本上下文和选择标准的结构化提示。在我们的实验中，我们将邻居预算设置为 $k = 4$。

空间感知智能体和检查器。二者均作为空间感知检查器的组件实现（算法 2）。空间感知智能体接收 CSA 选出的原始关键形状，并以自然语言输出结构化的空间关系描述。随后，检查器智能体将这些空间关系与提出的计划和动作相结合，生成最终的验证与纠正输出。

5 评估

我们使用开源的 OpenROAD [1] 流程根据 Verilog 设计生成布局，并使用 KLayout [12] 进行布局操作。所有实验均在 ASAP7 [6]下进行。

5.1 评估设置

基准测试。由于不存在公开的设计规则检查修复基准测试，我们构建了一个多层级基准测试，用于在不同复杂度下评估智能体。该基准测试包含两个级别：**L1级别** 用于基本操作，**L2级别** 用于复杂推理。值得注意的是，*L2*级别任务采集自设计规则检查

表 1：L1 上的总体比较。L1 用于评估基础模型的基本能力。

方法	开源	模型规模	L1 第1类(%)		L1 第2类(%)		L1 第3类(%)		总体(%)	
			pass@1	pass@5	pass@1	pass@5	pass@1	pass@5	pass@1	pass@5
不同基础模型										
ReAct(GPT5-mini)	✗	N/A†	94.8	98.9	91.7	99.9	96.7	100.0	94.3	99.2
ReAct(Gemini-2.0)	✗	N/A†	72.4	94.2	53.3	95.8	90.0	100.0	70.3	95.1
ReAct(DeepSeek-V3.1)	✓	671B	85.2	94.2	66.7	91.3	90.0	100.0	82.0	94.2
ReAct(Qwen-2.5-7B)	✓	7B	9.52	26.7	6.67	23.6	10.0	42.6	9.0	27.7
ReAct（Llama-3.1-8B）	✓	8B	2.38	11.9	0.00	0.00	13.3	47.2	3.00	13.1

⭐ 橙色行表示足以胜任设计规则检查修复智能体的基础模型。 ⊘ 粉色行表示设计规则检查修复能力有限的基础模型。 †: GPT5-mini 和 Gemini-2.0 的模型规模未公开。

违规项，而OpenROAD无法自动解决这些问题，因此没有必要与OpenROAD内置的设计规则检查修复器进行比较。

L1 评估基础模型的基本能力：工具调用和基础设计规则检查知识。所有 L1 错误均由人工构造，包含三类：

⋄ **第1类**：仅涉及一个形状的原子案例。例如，宽度为 14 纳米的单个 M1矩形违反了 18 纳米的最小宽度规则——智能体必须将其加宽至至少 18 纳米。⋄ **第2类**：包含 2–3 个分布在不同图层上的形状的布局，是第1类的扩展，无需推理——与第1类相比，智能体需要处理多个形状。例如，部分重叠于下方V1图形的 M1矩形，必须扩大或重新定位，才能实现完全包围。

⋄ **第3类**：周围存在密集图形的实际布局，其中人工工程师在 DRC 合规布局中注入了一个简单错误。修复仍然直接了当，但智能体必须应对复杂的上下文。例如，在 gcd布局中，一条 M1导线的宽度被缩窄至低于最小宽度规则的要求——只需恢复其宽度即可修复，但在密集环境中，这会占用大量图形空间。

L2级别 用于评估智能体在复杂情境下的推理能力，尤其针对 OpenROAD 未能解决、通常需要人工介入的错误。我们定义了三个推理难度逐渐递增的类别：

⋄ **第1类**：需要恰好一个修复步骤的任务，但该步骤要求对周围图形进行复杂的几何推理。例如，M1.S.6 角到角间距违规（最小值为 20 纳米）无法仅通过移动 M1 形状来解决，因为这样会违反 V1 包围规则，因此智能体必须改为移动 VIA12 单元。

⋄ **第2类**：至少需要多个修复步骤的任务，其中解决主要违规项也会影响受间距或层间规则约束的相邻形状。例如，修复 M1.S.2 边缘间距违规（最小值为 25 纳米，当某条边超过 36 纳米时触发）需要移动 VIA12 及其关联的 M1 和 V1 图形，并添加一个新的 M1 形状，以防止引入新的违规项。

⋄ **第3类**：报告的违规并非根本原因，因此智能体无法仅根据 DRC 报告推导出最少步骤的解决方案。例如，V1.S.4 报告两个 V1 实例之间的最小角到角间距为 27 纳米，其中一个带有 5 纳米的 M2 端盖，而另一个没有。直接增大 V1 间距会在其他位置引入新的错误，需要更多步骤。最少步骤的解决方案应改为延长 M2 端盖，从而消除触发该违规的非对称间距规则。

指标。 每项任务以系统提示 s、用户提示 q（其中包含 GDS 布局路径和初始违规集合 E_0, ）以及最大迭代预算 T 作为输入，并输出修改后的布局 L^*。若 $|E(L^*)| = 0$ （即完全符合设计规则检查要求），则视为成功；否则视为失败，包括仍存在违规项或引入新违规项的情况。我们使用*pass@k* 指标 [4], 报告结果，该指标衡量 k 次独立运行中至少有一次生成符合设计规则检查要求的布局的概率（$k \in \{1, 5\}$）。每项任务在温度 0.7 下采样 $n = 10$ 次。

基线。 我们将方法与以下框架进行比较：*ReAct*（单智能体）：在统一循环中整合推理与工具执行。规划器与执行者（多智能体）： 规划器生成推理计划，执行者则据此调用工具。我们使用三种基础模型评估每种架构：GPT5-mini、Gemini-2.0 和 DeepSeek-V3.1（仅文本模态）。

5.2 研究问题

我们探讨以下**研究问题（RQs）**：（RQ1）AgentDRC 与其他智能体框架相比表现如何？（RQ2）三个特色智能体的效果如何？（RQ3）多层级、多模态表示与扁平化的纯文本输入相比表现如何？（RQ4）top-k 因子如何影响上下文选择？（RQ5）迭代次数限制和成本在不同任务之间有何差异？

5.3 总体性能比较（RQ1）

我们在两个基准测试级别上评估系统：L1（表 1）和 L2（表 2）。L1 侧重于基础的智能体能力，例如理解设计规则、解读几何上下文以及生成有效的工具调用。大规模模型（GPT5-mini、Gemini-2.0、DeepSeek-V3.1）表现可靠，总体 pass@5 分数分别达到 **99.2%** 、 **95.1%** 和 **94.2%**；而较小的模型（Qwen-2.5-7B、Llama-3.1-8B）则经常因工具调用格式错误或语义不正确而失败。因此，L2 评估仅使用大规模模型。

表2：AgentDRC 与 L2级别上各种基线模型的比较。

方法	开源	模态	第1类(%)		L2 第2类(%)		L2 第3类(%)		总体(%)	
			pass@1	pass@5	pass@1	pass@5	pass@1	pass@5	pass@1	pass@5
单智能体										
ReAct(GPT5-mini)	✗	T	24.0	51.7	16.0	34.8	6.7	33.3	18.3	41.9
ReAct(Gemini-2.0)	✗	T	10.0	29.1	8.00	27.5	16.7	56.5	10.0	32.0
ReAct(DeepSeek-V3.1)	✓	T	10.0	26.9	14.0	37.5	10.0	50.0	11.7	34.5
多智能体系统										
MAS(GPT5-mini)	✗	T	8.00	27.5	6.00	25.6	3.33	16.7	6.52	25.3
MAS(Gemini-2.0)	✗	T	13.0	25.0	8.00	17.8	3.33	16.7	9.56	20.8
多智能体系统(DeepSeek-V3.1)	✓	T	3.00	15.0	8.00	37.8	6.67	25.9	5.65	26.3
AgentDRC (GPT)	✓	T+V	39.0	68.9	22.0	42.7	6.67	33.3	30.4	55.7
AgentDRC (Gemini)	✓	T+V	17.0	47.7	7.00	22.5	33.3	82.4	14.8	41.3
AgentDRC (DeepSeek)	✓	T	21.0	44.5	16.0	41.9	10.0	42.6	17.4	43.1

†：性能排名，分别为🏆第 1 名、🏆第 2 名、🏆第 3 名；注：MAS 指规划器与执行器架构。§：“T”表示多层级纯文本模态，而“T+V”表示同时使用多层级文本和视觉模态。⋆：GPT：GPT5-mini；Gemini：Gemini-2.0；DeepSeek：DeepSeek-V3.1。我们将 CSA 的 top-k 参数设置为 $k = 4$。

L2 面向更具挑战性的多步 DRC修复任务。**L2 级别的错误本质上是 OpenROAD 无法自动修复的剩余违规项**。我们采用 GPT5-mini 的多智能体系统表现最佳，pass@1 达到 **30.4%**，pass@5 达到 **55.7%**。对于第1类任务，ReAct（GPT5-mini）表现出色（pass@1 为 24.0%，pass@5 为 51.7%），这得益于 GPT5-mini 强大的内在推理能力。随着第2类任务复杂度的提升，我们的算法在所有基础模型上始终优于单智能体方法——AgentDRC（GPT）取得最佳结果，AgentDRC（DeepSeek）位居第二——这体现了协调式多智能体推理的优势。对于第3类任务，基于 Gemini 的变体表现更佳（pass@1 为 33.3%，pass@5 为 82.4%），表明当智能体必须超越给定的错误描述进行推理时，Gemini 具有更强的探索行为。然而，对于规划器–执行器架构，三个强大的基础模型——GPT5-mini、Gemini-2.0 和 DeepSeek-V3.1——表现均有所下降。这些模型已能有效整合推理与工具调用，因此将规划与执行分开会引入不匹配并导致额外错误。总体而言，结果表明，AgentDRC 的多智能体协作显著提升了复杂场景下设计规则检查修复的可靠性。

5.4 消融实验（RQ2）

我们以 Gemini-2.0 作为基础模型开展消融实验，分别移除观察代理、上下文选择智能体和空间感知智能体。实验变体如下所示，结果汇总于表 3。

• **基线（ReAct）**。 一种单智能体 ReAct 系统，在没有任何专门的智能体分解的情况下执行推理、工具调用和反馈解读。

• **AgentDRC无观察智能体**。 该系统移除了负责提出初始计划和候选工具操作的模块。工具输出会直接转发给上下文选择智能体，以进行形状过滤。

表 3：L2级别上的消融实验。Pass@5 Δ 表示相对于完整 AgentDRC 模型的下降幅度。

变体	L2 pass@1	L2 pass@5	Pass@5 Δ
基线（ReAct）	10.0	32.0	↓ 9.3
无观察智能体	18.3	38.7	↓ 2.6
无上下文选择智能体	11.3	30.9	↓ 10.4
不含空间感知智能体	11.7	33.5	↓ 7.8
AgentDRC	14.8	41.3	=

• **AgentDRC无上下文选择智能体**。 系统不再选择相关形状，而是将所有几何实体传递给空间感知智能体，从而增加上下文冗余，并迫使下游智能体处理不必要的信息。

• **AgentDRC不含空间感知智能体**。原始几何数据会直接传递给检查器智能体，所选图形之间没有结构化空间关系。

• **AgentDRC**。 包含全部三个特色智能体的完整多智能体系统。

如表 3 所示，移除 OA 仅会导致性能适度下降（pass@1 为 18.3%，pass@5 为 38.7%）。这表明，即使没有明确的初始计划，只要保留了足够的空间线索，下游智能体仍能推断出合理的动作。相比之下，省去上下文选择智能体会造成最显著的性能下降（pass@5 降至 10.4%），这说明过滤无关图形对于减少上下文噪声、实现精准推理至关重要。移除空间感知智能体也会导致明显的性能下降（pass@5 降至 7.8%），进一步证实结构化空间关系对于引导检查器提出准确的动作方案十分重要。

总体而言，完整的 AgentDRC 配置取得了最佳结果，验证了我们多智能体架构中这三个核心智能体的必要性与互补性。

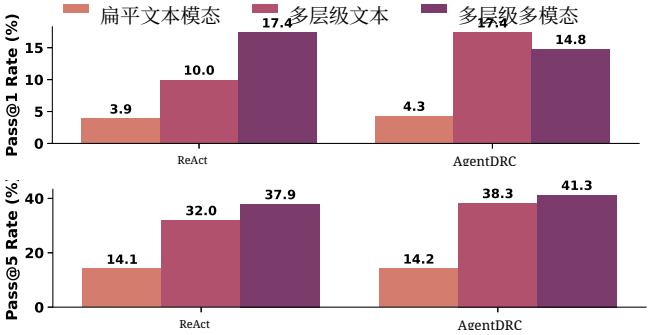

图 5：多层级和多模态几何信息的有效性

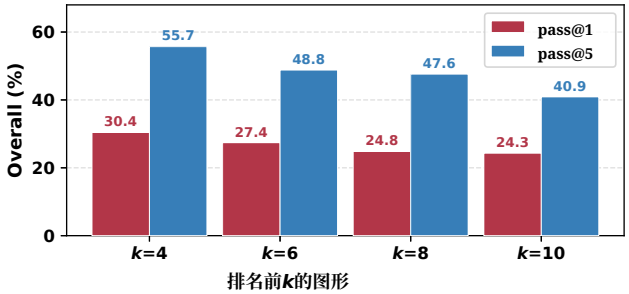

图 6：前 k 个选定图形对 L2级别总体性能的影响。选定的图形越少，结果越好，因为过多的上下文会引入噪声。

5.5 多模态的有效性（RQ3）

为了评估几何信息结构的影响，我们使用 Gemini-2.0，在 ReAct 和 AgentDRC 下比较扁平文本模态、多层级文本和多层级多模态（文本+视觉）输入（图 5）。采用扁平文本输入时，两个系统的表现都很差（ReAct：pass@1/pass@5 分别为 3.9%/14.1%，AgentDRC：4.3%/14.2%），这表明智能体无法根据扁平的原始坐标可靠地定位关键形状。引入多层级层次化文本后，性能显著提升：ReAct 升至 10.0%/32.0%，AgentDRC 升至 17.4%/38.3%。这是因为结构化的多层级几何信息有助于智能体聚焦于相关几何形状。

加入多模态视觉可带来进一步提升（ReAct：17.4%/37.9%，AgentDRC：14.8%/41.3%），尽管 ReAct 的提升更为明显，因为 AgentDRC 已经受益于其上下文选择和空间感知智能体。总体而言，多层级多模态带来了最强的结果，证实层级化线索与视觉线索能够共同增强空间推理。

5.6 Top-k 选择的影响（RQ4）

为研究 CSA 的 top-k 参数敏感性，我们在保持其他设置不变的情况下，将选取的邻近上下文形状数量从 k=4 调整为 k=10（使用 GPT5-mini 的 AgentDRC，针对 L2级别）。结果如图 6 所示。

使用 k=4 时，系统的整体性能最佳（pass@1 为 30.4%，pass@5 为 55.7%）。将 k 增加到 6 和 8 会导致

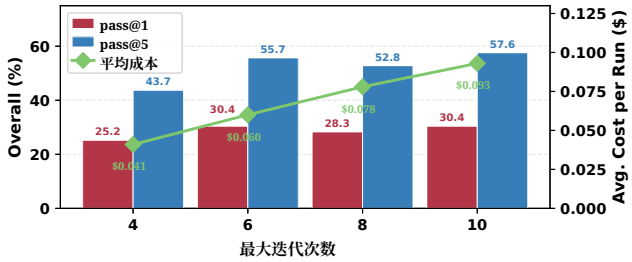

图 7：最大迭代次数对 L2级别性能和成本的影响。（a）总体 pass@1 和 pass@5 在超过 6 次迭代后基本趋于饱和。（b）每次运行的 API 成本随着迭代预算的增加而稳步上升。

性能逐渐下降，当 k=10 时，性能明显下滑（pass@1 为 24.3%，pass@5 为 40.9%）。这一趋势证实，纳入更多图形会引入上下文噪声：少量经过仔细筛选的图形能够为检查器提供聚焦且相关的空间信息，而更大的上下文则会稀释关键线索，降低推理准确性。因此，我们在所有实验中均采用 k=4 作为默认值，在覆盖度与精度之间实现最佳权衡。

5.7 最大迭代次数的影响（RQ5）

我们在使用 GPT5-mini 的 AgentDRC 下，将迭代上限从 4 调整至 10（图 7）；其中，一次迭代是完整执行一遍智能体流水线——观测、上下文选择、空间感知验证和工具执行。

仅进行 4 次迭代时，性能最低（pass@1 为 25.2%，pass@5 为 43.7%），因为智能体通常会在完成多步修复之前耗尽其预算。将上限提高到 6 次后，性能大幅提升（pass@1 为 30.4%，pass@5 为 55.7%）。超过 6 次迭代后，收益逐渐减少：我们观察到，初始推理方向正确的任务往往能在 6 次迭代内成功，而一开始就走错路径的任务即使增加迭代次数也很少能够恢复。具体而言，8 次迭代的表现略低于 6 次（pass@1 为 28.3%，pass@5 为 52.8%），原因在于过度探索；而 10 次迭代时，性能恢复至 pass@1 为 30.4%、pass@5 为 57.6%。

API 成本大致随迭代上限线性增长，从 4 次迭代时的 $0.041 增至 10 次迭代时的 $0.093，在所有设置下仍然经济实惠。我们采用 6 次作为默认值，以实现最佳的性能与成本权衡。

6 结论

我们提出了 AgentDRC，这是一组用于空间感知 DRC 修正的算法，通过协调多个大语言模型智能体，修复传统电子设计自动化工具无法解决的违规项。该系统能够感知多层级、多模态的几何上下文，通过两步上下文选择过滤无关图形，重建关键图形之间的结构化空间关系，并通过几何与语义约束检查验证拟议动作。实验表明，AgentDRC 在具有挑战性的违规项上最高可达到 30.4% 的 pass@1 和 55.7% 的 pass@5，证实了所提出算法及多智能体协同的有效性。

参考文献

[1] T. Ajayi、A. B. Kahng、B. Li、R. Nath、P. Sharma、M. Soroushmehr 和 B. Xu。2020。OpenROAD：迈向自动驾驶式、开源数字版图实现工具链。在*ACM/IEEE* 设计自动化会议（*DAC*）论文集中。1–6。doi:10.1109/DAC18072.2020.9218696[2] Xiaohe Bo、Zeyu Zhang、Quanyu Dai、Xueyang Feng、Lei Wang、Rui Li、Xu Chen 和 Ji-Rong Wen。2024。基于大语言模型的反思式多智能体协作。在神经信息处理系统第 *38* 届会议（*NeurIPS*）论文集中。[3] Cadence 设计系统公司。2023。*Cadence Innovus+* 综合与实现系统——*RTL* 到 *GDS* 芯片设计：加速交付成果。技术报告。Cadence 设计系统公司，https://www.cadence.com/en_US/home/resources/white-papers/innovus-plus-synthesis-implementation-system-wp.html 白皮书。[4] Mark Chen、Jerry Tworek、Heewoo Jun、Qiming Yuan、Henrique Ponde de Oliveira Pinto、Jared Kaplan、Harri Edwards、Yuri Burda、Nicholas Joseph、Greg Brockman、Alex Ray、Raul Puri、Gretchen Krueger、Michael Petrov、Heidy Khlaaf、Girish Sastry、Pamela Mishkin、Brooke Chan、Scott Gray、Nick Ryder、Mikhail Pavlov、Alethea Power、Lukasz Kaiser、Mohammad Bavarian、Clemens Winter、Philippe Tillet、Felipe Petroski Such、Dave Cummings、Matthias Plappert、Fotios Chantzis、Elizabeth Barnes、Ariel Herbert-Voss、William Hebgen Guss、Alex Nichol、Alex Paino、Nikolas Tezak、Jie Tang、Igor Babuschkin、Suchir Balaji、Shantanu Jain、William Saunders、Christopher Hesse、Andrew N. Carr、Jan Leike、Josh Achiam、Vedant Misra、Evan Morikawa、Alec Radford、Matthew Knight、Miles Brundage、Mira Murati、Katie Mayer、Peter Welinder、Bob McGrew、Dario Amodei、Sam McCandlish、Ilya Sutskever 和 Wojciech Zaremba。2021。评估在代码上训练的大语言模型。（ 2021 2107 03374 [ ] ）。arXiv：。[5] 2024 cs.LG Xinyun Chen、Maxwell Lin、Nathanael Schärli 和 Denny Zhou。。教授大语言模型进行自调试。在第十二届国际学习表征会议中。https://openreview.net/forum?id=KuPixIqPiq [6] T. Clark、L.、V. Vashishtha、L. Shifren、A. Gujja、S. Sinha、B. Cline、C. Rama- murthy 和 G. Yeric。2016。ASAP7：7 纳米 FinFET 预测工艺设计套件。*Microelectronics Journal* 53, 2016 105 115 10 1016 2016 04 006 —（），–。doi：。/[7] j.mejo.。。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 和 Pete Florence。2023。PaLM-E：具身多模态语言模型。在第 *40* 届国际机器学习会议（美国夏威夷檀香山）（*ICML'23*）中。JMLR.org，第 340 号文章，共 20 页。[8] Luyu Gao、Aman Madaan、Shuyan Zhou、Uri Alon、Pengfei Liu、Yiming Yang、Jamie Callan 和 Graham Neubig。2023。PAL：程序辅助语言模型。在第 *40* 届国际机器学习会议（美国夏威夷檀香山）（*ICML'23*）中。JMLR.org，第 435 号文章，共 36 页。[9] GlobalFoundries 和西门子业务部门 Mentor。2019。自动化 *DRC* 违规修正加速设计流片。技术报告。https://www.edn.com/automated-drc-violation-correction-speeds-designs-to-tapeout/ 白皮书。[10] GLOBALFOUNDRIES 和西门子 EDA。2020。自对准双重图形化（*SADP*）设计收尾：通过 *DRC* 自动修复改善结果。技术报告。西门子数字化工业软件。https://resources.sw.siemens.com/en-US/technical-paper-self-aligned-double-patterning-design-finishing-improving-results-with-drc 描述 SADP 布局中挑战及自动化设计规则检查修复局限性的技术论文。[11] 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 和 Jürgen Schmidhuber。2024。MetaGPT：面向多智能体协作框架的元编程。在第十二届国际学习表征会议中。https: //openreview.net/forum?id=VtmBAGCN7o [12] Matthias Köfferlein。2024。KLayout——高性能版图查看器和编辑器。https://www.klayout.de/ 开源版图查看器和编辑器。[13] 西门子业务部门 Mentor Graphics。2016。*Calibre* 物理验证系统：设计规则检查与签核验证。技术报告。Mentor Graphics 公司。https://resources.sw.siemens.com/en-US/white-paper-calibre-physical-verification-system 技术白皮书。[14] S. Nayak 等。2024。部分可观测环境中的多智能体机器人长时域规划。在神经信息处理系统第 *38* 届会议（*NeurIPS*）论文集中。[15] Shishir G Patil、Tianjun Zhang、Xin Wang 和 Joseph E. Gonzalez。2024。Gorilla：连接海量 API 的大语言模型。在神经信息处理系统第三十八届年会中。https://openreview.net/forum?id=tBRNC6YemY[16] Timo Schick、Jane Dwivedi-Yu、Roberto Dessí、Roberta Raileanu、Maria Lomeli、Eric Hambro、Luke Zettlemoyer、Nicola Cancedda 和 Thomas Scialom。2023。Toolformer：语言模型能够教会自己使用工具。在第 *37* 届国际神经信息处理系统会议论文集中（美国路易斯安那州新奥尔良）（*NIPS '23*）。Curran Associates Inc.，美国纽约州雷德胡克，文章 2997，13 页。[17] 西门子 EDA。2023。用于 *DRC* 调试的自动化后处理。技术报告。西门子数字化工业软件。https://static.sw.cdn.siemens.com/siemens-disw-assets/public/5R5kzgoX1k13WtoqjEEFqO/en-US/Siemens-SW-Automated-post-processing-DRC-debugging-83714.pdf 白皮书阐述了 DRC/DFM 复杂性，以及仅通过自动化实现完全无错误的设计并不切实际。[18] 西门子数字化工业软件。2018。工艺设计套件：保护设计诀窍。技术报告。西门子 EDA。https://static.sw.cdn.siemens.com/siemens-disw-assets/public/82831/en-US/Siemens-SW-The-process-design-kit-WP-81843-C2.pdf 白皮书。[19] 西门子数字化工业软件。2020。*DRC*错误的自动化后处理可提高调试效率。技术报告。西门子 EDA。https://static.sw.cdn.siemens.com/siemens-disw-assets/public/5R5kzgoX1k13WtoqjEEFqO/en-US/Siemens-SW-Automated-post-processing-DRC-debugging-83714.pdf 白皮书。[20] Chan Hee Song、Jiaman Wu、Clayton Washington、Brian M. Sadler、Wei-Lun Chao 和 Yu Su。2023。LLM-Planner：利用大语言模型为具身智能体进行少样本具身规划。载于*IEEE/CVF* 国际计算机视觉会议（*ICCV*）论文集。[21] Guanzhi Wang、Jiahui Fan、Chen Qian、Weiyu Liu、Jiwoo Park、Yichi Zhou、Yuxuan Liu 等。2024。Voyager：一种基于大语言模型的开放式具身智能体。载于第 *41* 届国际机器学习会议（*ICML*）论文集。https://arxiv.org/abs/2305.16291 采用基于大语言模型的规划器—执行器框架来完成开放世界任务。[22] Shunyu Yao、Jeffrey Zhao、Dian Yu、Nan Du、Izhak Shafran、Karthik Narasimhan 和 Yuan Cao。2023。ReAct：语言模型中的推理与行动协同。载于国际学习表征会议（*ICLR*）。[23] Kechi Zhang、Jia Li、Ge Li、Xianjie Shi 和 Zhi Jin。2024。CodeAgent：通过工具集成智能体系统增强代码生成，以应对真实世界仓库级编程挑战。载于计算语言学协会第 *62* 届年会论文集（第一卷：长篇论文），Lun-Wei Ku、Andre Martins 和 Vivek Srikumar（编）。计算语言学协会，泰国曼谷， 13643 13658 10 18653 1 2024 737 –。doi：。/v/。[24] 2021 acl-long。Kaiqing Zhang、Zhuoran Yang 和 Tamer Basar。。《多智能体强化学习：理论与算法精选综述》。强化学习与控制手册 （2021），321–384。