Computer Science and Application 计算机科学与应用, 2026, 16(8), 100-113
Published Online August 2026 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2026.168266

Hans汉斯

面向模拟电路设计的大模型检索增强与仿真反馈闭环框架

王俊垒[1]，江赛标[2*]，陈　翀[1]，程良伦[1]

[1]广东工业大学，广东省信息物理融合重点实验室，广东 广州
[2]珠海科技学院，电子信息工程学院，广东 珠海

收稿日期：2026年7月5日；录用日期：2026年8月6日；发布日期：2026年8月12日

摘　要

大语言模型为自然语言驱动的电路设计提供了新的交互方式，但在模拟电路场景中，模型输出仍容易受到器件库缺失、参数不匹配和仿真不可通过等问题影响。针对自然语言需求难以直接转化为可执行网表的问题，本文设计了一种检索增强与仿真反馈相结合的模拟电路生成流程。该流程首先将用户需求整理为结构化元器件清单，再通过精确别名匹配和向量语义检索将抽象器件映射到KiCad符号库中的可用元件，随后依据匹配结果生成SPICE网表，并由设计者结合Ngspice仿真结果进行反馈修正。该方法强调工具链可用性与人工可控性，在避免复杂多智能体编排的同时保留了从需求理解、器件落地到仿真验证的闭环。基于5类典型模拟电路的实验表明，DeepSeek-V3.2后端下该流程取得86.7%的仿真通过率和3.1%的平均目标偏差，说明该方案能够提升LLM生成模拟电路的可执行性和指标贴合度。

关键词

大语言模型，模拟电路设计，向量检索，SPICE仿真，元器件库匹配

A Closed-Loop LLM Framework with Retrieval Augmentation and Simulation Feedback for Analog Circuit Design

Junlei Wang[1], Saibiao Jiang[2*], Chong Chen[1], Lianglun Cheng[1]

[1]Guangdong Provincial Key Laboratory of Cyber-Physical Systems, Guangdong University of Technology, Guangzhou Guangdong
[2]School of Electronics and Information Engineering, Zhuhai College of Science and Technology, Zhuhai Guangdong

Received: July 5, 2026; accepted: August 6, 2026; published: August 12, 2026

[*]通讯作者。

文章引用: 王俊垒, 江赛标, 陈翀, 程良伦. 面向模拟电路设计的大模型检索增强与仿真反馈闭环框架[J]. 计算机科学与应用, 2026, 16(8): 100-113. DOI: 10.12677/csa.2026.168266

Abstract

Large language models (LLMs) often struggle to generate executable analog circuit netlists due to component mismatches and simulation failures. To address this, we propose a closed-loop workflow integrating retrieval augmentation and simulation feedback. The system parses user requirements into structured lists, maps abstract devices to KiCad libraries via exact alias and vector semantic retrieval, and generates SPICE netlists. Designers then iteratively refine these netlists using Ngspice simulation feedback. This highly controllable, human-in-the-loop approach avoids complex multi-agent orchestration while ensuring practical usability. Evaluated on five typical analog circuits using DeepSeek-V3.2, our method achieved an 86.7% simulation pass rate and a 3.1% average target deviation, demonstrating significantly improved executability and metric alignment for LLM-generated circuits.

Keywords

Large Language Model, Analog Circuit Design, Vector Retrieval, SPICE Simulation, Component Library Matching

1. 引言

1.1. 研究背景与意义

近年来，以 DeepSeek-V3、GPT-4o 和 Qwen3 等为代表的大语言模型(Large Language Model, LLM)在代码生成、文本理解和结构化输出任务中表现突出，使自然语言逐渐成为复杂工程工具的新型交互入口。在电子设计自动化(Electronic Design Automation, EDA)领域，利用 LLM 辅助电路设计已成为学术界和工业界共同关注的方向[1] [2]，相关研究覆盖数字 IC 设计中的 RTL 生成、EDA 脚本生成以及模拟电路拓扑综合等任务[3]。与一般软件代码生成相比，电路设计尤其是模拟电路设计对正确性的要求更高，不仅需要网表语法合法，还要满足器件模型兼容、电气连接合理、偏置条件成立和整体拓扑可实现等物理约束。

模拟电路设计是 EDA 流程中经验依赖较强的环节。设计者不仅要确定拓扑结构，还需要完成元器件选型、参数估算、偏置设定和信号特性分析等相互耦合的工作。传统流程通常从指标分析开始，随后选择拓扑、绘制原理图、导出网表并进行 SPICE 仿真；若仿真结果不满足要求，还需要回到参数或拓扑层面反复调整。尽管 LLM 擅长理解自然语言并生成代码，直接把自然语言需求转换为可执行的模拟电路网表仍面临两类困难：一是模型缺少对具体 EDA 元器件库的可靠感知，可能生成实际库中不存在或无法实例化的器件[4]；二是网表的文本语法正确并不意味着电路能够工作，参数偏差、拓扑连接错误和偏置条件不合理等问题往往只有经过 SPICE 仿真[5]后才会暴露。

围绕上述问题，已有研究分别从检索增强生成(RAG)、多智能体协作和仿真引导优化等角度展开探索[6]。RAG 方法试图将外部元器件或工具文档注入模型上下文，多智能体方法通过任务分工降低单次生成难度，仿真引导方法则利用 SPICE 结果驱动后续修正。这些工作为 LLM 进入 EDA 流程提供了重要基础，但不少方法仍偏重单一环节，例如检索质量、代码生成或自动修复；同时，部分系统依赖复杂的 Agent 编排和自动评估器，增加了复现与工程维护成本。针对上述不足，本文设计了一种轻量级的 LLM 辅助模

拟电路生成方法，将需求细化、元器件库检索、SPICE 网表生成和 Ngspice 仿真反馈串联为可复现的工作流，使 LLM 的语义生成能力受到真实元器件库和物理仿真的双重约束。

1.2. 国内外研究现状

大语言模型在 EDA 中的应用已经形成若干具有代表性的技术路线。本文围绕与模拟电路生成关系最密切的四类工作进行讨论。

LLM 直接生成电路代码。在数字电路层面，开源 7B 参数模型在 Verilog RTL 生成任务上取得了接近 GPT-4 的表现，说明小模型在特定 EDA 任务中具有可训练空间。在原理图生成方面，黄俊等人探索了从电路描述到原理图表示的自动生成[7]；在模拟电路层面，SPICEPilot [5]将 Ngspice 结果引入 SPICE 代码迭代修复，为仿真反馈提供了有益示例，但其重点仍在网表局部修复，对元器件规划和库匹配阶段的错误覆盖不足。

检索增强在 EDA 中的应用。为补足 LLM 对专有工具和元器件知识的缺口，检索增强生成(RAG)被广泛用于 EDA 任务。PE-GPT [8]通过检索注入电力电子领域知识，EDA-Copilot [9]连接 EDA 工具文档以辅助脚本生成，通过综合文档、设计实例和仿真数据进行多模态检索。需要注意的是，通用 RAG 主要优化文本语义相似度，而 EDA 设计还要求工具链层面的可实例化性。语义上相近的器件在 KiCad 库中可能具有不同符号格式、引脚布局或 SPICE 模型，检索结果若缺少库级约束[10]，仍可能无法直接用于后续设计。

多智能体与工作流自动化。多智能体系统通常将复杂设计任务拆分为若干专业子任务。Circuit LM [11]将电路设计划分为多个专业 Agent 协作；曾怡等人探讨机器学习在布局领域的应用，通过分析逻辑与技术特点优化布局算法[12]。这类方法强化了任务分解能力，但也带来了通信协调、错误传播和流程调参等额外复杂度，最终的物理可行性仍需要依赖外部仿真验证。

总体而言，现有研究已经证明 LLM 可参与 EDA 任务，但在模拟电路生成场景中仍存在三个不足：一是从器件库匹配到仿真验证的衔接不够完整；二是部分方案依赖复杂协作架构，复现和维护成本较高；三是仿真环节往往在全自动评估和简单语法检查之间摇摆，缺少适合中小规模实验的人工协作机制。本文围绕这些问题构建线性流水线，采用精确匹配与向量检索结合的库匹配策略，并引入人工在环的仿真反馈，以便在工程可实现性和系统简洁性之间取得平衡。

1.3. 本文工作与贡献

本文面向自然语言驱动的模拟电路生成任务，提出一种便于复现和验证的 LLM 辅助设计方法，主要工作如下：

(1) 实现了面向 KiCad 符号库的元器件信息提取与向量化检索流程，覆盖 S-表达式解析、结构化描述文本构建、ChromaDB 向量存储以及精确匹配与语义搜索结合的两级检索。

(2) 设计了基于 Prompt 工程的元器件规划方法，通过需求细化和格式约束使 LLM 输出可解析、可检索的 BOM 清单，为后续库匹配和网表生成提供稳定输入。

(3) 构建了人机协作的仿真反馈闭环：LLM 生成 SPICE 网表，设计者使用 Ngspice 检查关键指标，并将偏差以自然语言反馈给模型进行针对性修正。

(4) 在 5 类典型模拟电路上进行了实验验证，结果表明该方法在仿真通过率、目标偏差和元器件匹配等指标上具有较好的可用性，并优于去除关键环节后的基线配置。

(5) 构建了元件符号－SPICE 模型－数据手册参数的显式链接知识库。在元件检索阶段不仅考虑文本语义相似度，同时验证候选元件对应 SPICE 模型的可用性和质量等级，将模型可用性作为检索排序的重要依据，从源头上确保生成网表的可仿真性，避免了“符号匹配但模型缺失”导致的仿真失败。

2. 相关技术基础

2.1. 大语言模型与 Prompt 工程

大语言模型是基于 Transformer 架构的大规模预训练语言模型，其核心机制是自注意力(Self-Attention)。该机制使模型在生成每个 token 时能够关注输入序列中的不同位置，从而建模长距离语义依赖[13]。当前主流 LLM 多采用仅解码器(Decoder-Only)架构，通过自回归语言建模(Autoregressive Language Modeling)学习条件概率分布 $P(x_t|x<t)$。在预训练之后，模型通常还会经过指令微调(Instruction Tuning)和人类反馈强化学习(RLHF)，以增强指令遵循和结构化输出能力。

在实际应用中，Prompt 工程直接影响 LLM 的输出稳定性。Prompt 通常由系统消息(System Message)和用户消息(User Message)组成，前者规定角色、输出格式和领域规则，后者提供具体任务输入。针对电路设计任务，Prompt 需要同时约束输出格式、术语单位和领域规则：元器件规划需要结构化 JSON 而不是自由文本，元器件型号和电气量纲应保持标准化，不同电路类型的拓扑规则也需要显式写入 System Prompt。本文还引入需求细化(Query Expansion)作为辅助步骤，先由 LLM 将简短需求扩展为包含电路功能、器件类型、关键参数范围和技术约束的设计说明，再将其作为元器件规划 Prompt 的输入，以提高后续规划的稳定性。

2.2. 文本嵌入与向量检索

文本嵌入(Text Embedding)将自然语言文本映射为固定维度的稠密向量，使语义相近的文本在向量空间中距离更近。本文采用 Sentence-Transformer 框架下的 paraphrase-multilingual-MiniLM-L12-v2 模型进行文本向量化。该模型基于 Microsoft MiniLM 架构，包含 12 层 Transformer 编码器，输出 384 维嵌入向量。相比早期句向量方法，Sentence-Transformer 通过孪生网络(Siamese Network)和对比学习目标(Contrastive Objective)提升语义相似度建模能力，并能较好处理中文需求与英文元器件术语混合出现的输入。

向量检索基于余弦相似度度量查询向量 q 与候选向量 d_i 之间的距离，其定义为：

$$\cos(q,d_i)=\frac{q\cdot d_i}{\|q\|\|d_i\|} \tag{1}$$

余弦相似度的取值范围为[−1, 1]，值越接近 1 表示两个向量在方向上的对齐程度越高，即对应的文本在语义上越相似。在实际检索中，ChromaDB 内部使用 $1-\cos(q,d_i)$ 作为距离度量(即余弦距离)进行最近邻搜索，返回距离最小的 Top-K 个候选结果。

ChromaDB 是轻量级开源向量数据库，适合中小规模语义检索。它支持预计算向量写入和内置嵌入函数编码，提供基于 HNSW (Hierarchical Navigable Small World)图索引的近似最近邻搜索，并允许为向量条目附加结构化 Metadata。本文利用 ChromaDB 管理从 KiCad 符号库提取的元器件条目，通过批量向量化存储和增量更新实现本地检索。与 Milvus 或 Pinecone 等更重的方案相比，ChromaDB 的 Python 接口、本地持久化和低服务依赖更适合千级元器件库的实验复现。

2.3. EDA 工具链

KiCad 是常用的开源 EDA 设计套件，提供原理图绘制(Eeschema)、PCB 布局(Pcbnew)和元器件库管理等功能。KiCad 符号库文件(.kicad_sym)采用 S-表达式(S-expression)保存符号定义。每个元器件符号由属性(Property)、图形元素(Polyline, Rectangle)和引脚(Pin)节点组成，在 S-表达式中以(symbol lib_id)为根节点。本文关注的关键字段包括唯一库标识符 lib_id、功能描述 Description、搜索关键词 ki_keywords、位号前缀 Reference、默认 PCB 封装 Footprint 以及引脚列表。SPICE (Simulation Program with Integrated

Circuit Emphasis)起源于20世纪70年代的加州大学伯克利分校，经过长期发展已成为常用电路仿真语言。Ngspice是SPICE-3f5的开源后继版本，支持直流工作点分析、直流扫描、交流小信号频率响应分析、瞬态时域分析等模式，并提供交互式命令行与批处理两种运行方式。

一个完整的SPICE网表通常由元器件实例化语句、模型定义语句和分析控制块三部分组成。元器件实例化语句以元件类型首字母(R/C/L/D/Q/M/X 等)开头，后接名称、连接节点和参数值；模型定义语句以.MODEL或.SUBCKT开头，用于描述半导体器件或子电路宏模型；分析控制块以CONTROL ENDC包裹，用于指定仿真类型和输出格式。本文在设计流程中连接KiCad与Ngspice的数据流：从KiCad符号库提取的元器件信息经向量化后用于检索匹配，匹配到的lib_id再与对应SPICE模型关联，并在网表生成阶段注入所需模型语句，以保持原理图符号、器件引脚和仿真模型的一致性。

3. 元器件库构建与向量化检索

元器件库连接LLM的抽象规划结果与EDA工具中的物理可用器件。本章介绍从KiCad符号库提取信息、构建结构化索引、向量化存储到两级检索的技术流程。

3.1. KiCad符号库解析与信息提取

KiCad符号库文件采用S-表达式格式存储。每个.kicad_sym文件以(kicad_symbol_lib)为根节点，内部包含若干(symbol)子节点，每个子节点对应一个元器件符号定义。本文设计递归下降解析器对符号库进行信息提取。解析器读取库文件文本后，调用Python的s-exp data库将S-表达式解析为嵌套列表结构，并遍历其中的所有符号节点。对每个(symbol lib_id)节点，解析器提取lib_id、Description、ki_keywords、Reference、Footprint、Pins和raw_symbol_definition等字段，分别用于标识器件、描述功能、支持搜索、确定位号前缀、记录封装、保存引脚信息和保留原始符号定义。extends父子继承关系需要单独处理。KiCad符号库中，基础符号通常定义通用引脚布局和图形表示，具体型号通过extends继承父类并覆写Description、Keywords、Footprint等属性。从检索角度看，用户查询具体型号时应能命中该型号对应的完整信息，而不是仅返回父类基础定义。因此，解析器在遍历时识别extends标记，将子类名称记录到父类映射表中，并在遍历完成后批量合并到对应父类的keywords列表。所有提取的元器件信息最终汇总为master_library_index.json文件，作为后续向量化处理的数据输入源。

3.2. 元器件描述文本构建

向量检索的效果取决于被编码文本的信息密度和语义完整性。元器件元数据本质上是结构化键值对，若直接拼接或原样编码，容易削弱不同封装、不同额定参数器件之间的区分度。因此，本文将结构化元数据转换为更适合嵌入模型处理的描述文本，并采用加权拼接策略构建每个元器件的搜索文本，字段优先级为Lib ID > Keywords > Description > Footprint > Source Library。该模板将标识信息置于文本最前，保留Description和Keywords的原始语义信息，并引入Footprint和Source Library作为辅助区分信息。引脚编号、电气类型和坐标属于物理属性，不直接写入搜索文本，而是通过ChromaDB的Metadata机制与搜索文本关联，在检索命中后随结果返回。

3.3. ChromaDB向量化存储

搜索文本构建完成后，系统将每个元器件的描述编码为固定维度稠密向量，并连同结构化元数据写入向量数据库。本文使用Sentence-Transformer模型paraphrase-multilingual-MiniLM-L12-v2对搜索文本进行编码，最终输出384维单精度浮点数向量。编码过程支持批量处理，将多个文本组成列表传入model.encode(texts)后可并行计算嵌入向量。对于包含约2000个元器件的库，全量编码在CPU上可在数

秒内完成。向量数据存储在 ChromaDB 中。系统首先创建或获取名为 electronic_components 的 Collection，并在 Collection 元数据中指定 hnsw:space 为 cosine。对于每个元器件条目，ChromaDB 存储 ID、Embedding、Document 和 Metadata 四类数据，其中 Metadata 包含 lib_id、source_library、designator_prefix、raw_symbol_definition、pins_json、default_footprint 等字段。本文将引脚信息和原始符号定义序列化后写入 Metadata，使检索命中后能够一次性取得后续 SPICE 网表连接和 KiCad 原理图输出所需的库标识、引脚和符号数据。为提高写入效率，系统采用 Batch Size = 64 的批量处理策略，通过 collection.upsert()一次性写入一批数据，以减少索引更新和 I/O 开销。

3.4. 两级检索匹配策略

元器件库向量化后，检索模块采用“先精确识别、后语义搜索”的两级匹配策略，流程如图 1 所示。该策略将确定性规则与语义检索互补使用：对基础无源器件和经典芯片型号等高频对象，优先通过规则映射快速定位；对规则无法覆盖的新型号或模糊表述，再使用向量语义检索进行补充匹配。

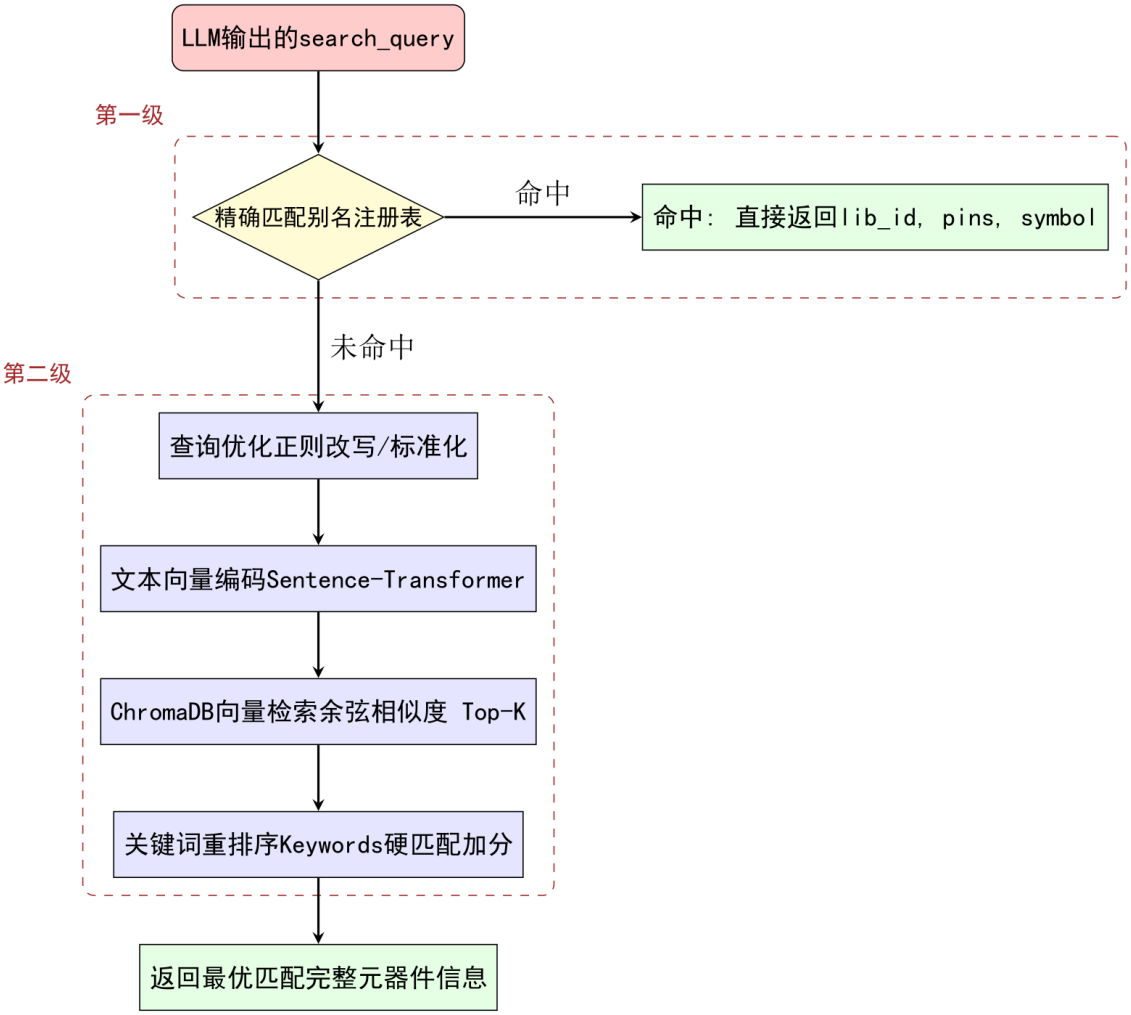

Figure 1. Two-stage component retrieval pipeline
图 1. 元器件两级检索流程图

第一级：精确匹配(Exact Match)。精确匹配模块维护一个别名注册表(Exact Match Registry)，以 Python

字典的形式将常见术语映射到 KiCad 库中的特定 lib_id。当 LLM 输出的 search_query 命中注册表中的任意触发词时(匹配方式为不区分大小写的子串匹配)，直接按 lib_id 精确查询并返回该器件的完整 Metadata 信息，跳过向量检索环节。在实验统计中，精确匹配的命中率约占总体检索请求的 38.1%，主要覆盖基础无源器件和经典芯片型号，这些器件类型在大部分电路设计中反复出现，精确匹配有效保障了高频查询的响应速度和准确性。

第二级：向量语义检索(Vector Semantic Retrieval)。对于未命中精确匹配的查询，检索流程进入第二级。首先执行查询优化(Query Optimization)：通过一系列正则表达式规则对查询文本进行标准化改写，以提升后续向量检索的召回率。

查询优化完成后，系统将改写后的查询文本编码为 384 维向量，并基于余弦相似度返回距离最近的 15 个候选元器件。随后进入重排序(Reranking)阶段，以纠正纯语义相似度可能产生的功能偏差。重排序后，系统选择得分最高的候选作为最终匹配结果，并返回 lib_id、描述文本、引脚列表、原始符号定义和相似度得分。单个查询的平均耗时在 100 毫秒以内(含嵌入编码时间)，能够满足交互式电路生成流程对响应速度的要求。

3.5. SPICE 模型可用性验证与排序增强

上述两阶段检索策略主要基于文本语义和关键词匹配对候选元件进行排序，但未显式验证候选元件的 SPICE 模型是否可用。在实际设计中，即使 KiCad 符号库中存在某元件的符号定义，若其缺少对应的 SPICE 模型，该元件在后续仿真阶段仍将导致失败。为解决这一问题，本文在检索流程中引入了 SPICE 模型可用性验证环节，构建了统一的“符号–模型”知识库。模型根据来源和可靠性分为 Ngspice 内建模型(如 R、C、L、V 等基础元件)、Tier-2 为框架内嵌的 SPICE.model 参数模型、Tier-3 为外部子电路库文件(如 LM2904.lib、NE555.lib)，以.subckt 形式提供完整的器件行为模型。对于无可用模型的符号，系统标记为“仅原理图”等级，在检索排序中被降权处理。

3.6. 检索性能分析

为评估两级检索策略，本文在 30 个电路设计任务构成的规划级测试集上统计命中率分布。实验以 DeepSeek-V3.2 作为规划模型，共生成 134 个独立 search_query。结果显示，精确匹配命中 51 个查询(占比 38.1%)，向量语义检索命中 81 个查询(占比 60.4%)，完全未命中 2 个查询(占比 1.5%)，整体检索命中率(Retrieval Hit Rate, RHR)达到 98.5%。

精确匹配 38.1%的命中率说明别名注册表对高频器件有效，命中对象主要包括电阻、电容、二极管、运放、555 定时器和连接器等基础器件；向量检索 60.4%的命中率则表明，对于名称变体、描述性修饰或非标准表述，语义检索能够提供较好的兜底能力。进一步分析 2 个未命中查询发现，失败主要来自非标准缩写或库中不存在的型号命名，例如 LLM 可能将某类 ADC 芯片写成库中无法识别的简写形式。因此，后续可在查询优化阶段加入 LLM 辅助改写，将未命中查询转换为标准型号或功能等价器件，同时持续扩充元器件库覆盖范围，以进一步降低完全未命中率。

4. LLM 驱动的电路生成与人工验证

本章描述从自然语言需求到仿真验证的设计流程，整体结构如图 2 所示。该流程将 LLM 的语义生成能力与 EDA 工具链中的物理约束结合起来，并通过人机协作完成网表修正。

4.1. 总体流程

如图 2 所示，本文方法包含五个主要阶段和一条反馈闭环，形成“生成–验证–修正”的迭代过程。

阶段一为需求细化(Query Expansion)，阶段二为 LLM 元器件规划。用户输入的自然语言需求通常较短，可能缺少元器件类型、参数范围和接口约束等信息，因此系统首先调用 LLM 将原始需求扩展为较完整的设计说明，内容包括电路功能、可能拓扑、主要元器件及其角色、关键指标和输入输出要求。随后，LLM 在 System Prompt 约束下生成结构化 BOM 清单，输出采用 JSON 格式，包含 circuit_name (电路名称)和 components (元器件列表)两个顶层字段；每个元器件条目包含 uid (位号标识符)、search_query (用于后续库检索的英文关键词或经典型号)和 parameters (推荐参数值)。

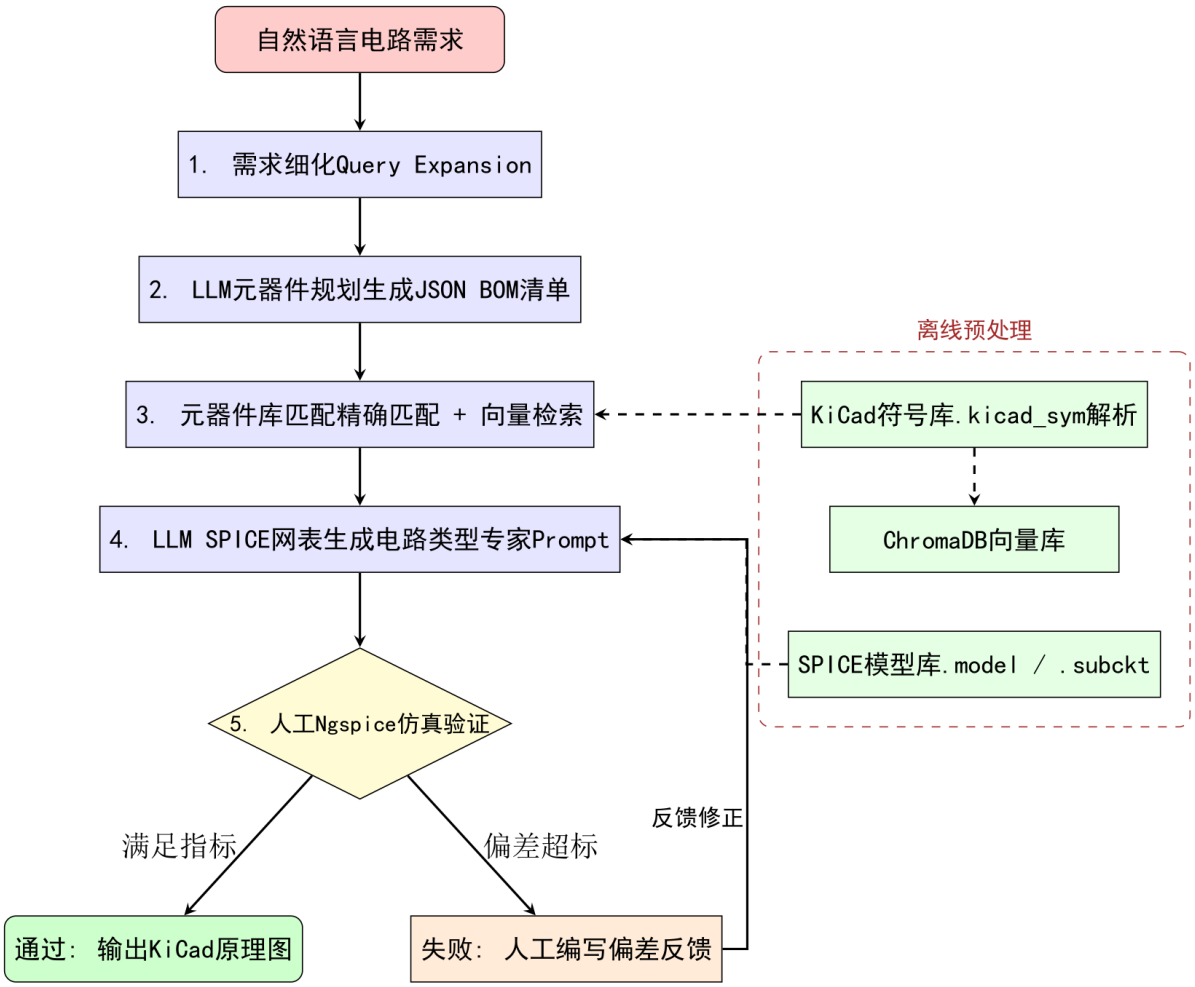

Figure 2. Overall framework of the proposed method
图 2. 本文方法总体流程图

阶段三至阶段五分别完成库检索匹配、SPICE 网表生成和人工仿真反馈。对 BOM 清单中的每个元器件，系统以 search_query 字段为输入执行第 3.4 节所述的两级检索，成功后获得该器件在 KiCad 库中的 lib_id、引脚列表和符号 S-表达式源代码；随后将匹配后的元器件清单、细化需求和自动识别的电路类型一并提交给 LLM，并根据电路类型加载专家 Prompt 模板，约束模型遵循节点命名、模型引用和常见拓扑规则。设计者在 Ngspice 中仿真生成网表，并对照设计指标判断是否满足要求；若结果存在偏差，则以自然语言描述偏差项、实测值和修正方向，再提交给 LLM 进行下一轮修改。上述阶段的数据通过 JSON 中间文件和 Ngspice 输出文件传递，右侧虚线框中的离线预处理模块(KiCad 符号库解析、ChromaDB 向

量库构建、SPICE 模型库整理)只需在初次配置时运行，后续任务可复用已构建资源。

4.2. 基于 Prompt 的元器件规划

元器件规划连接用户需求与后续电路实现，其输出质量直接影响库检索和网表生成。本文首先执行需求细化(Query Expansion)，将用户输入的简短需求扩展为较完整的电路设计说明，并将目标平台设为 KiCad 8，优先选择通用、基础和经典元器件。细化后的需求既用于引导元器件规划，也作为 SPICE 网表生成和仿真评估的目标参考。

在规划阶段，System Prompt 主要从角色、格式、语言和拓扑四个方面约束 LLM 输出。首先，将 LLM 设定为电子硬件设计助手，任务是将自然语言需求转化为结构化元器件物料清单(BOM)，供后续 KiCad 8 原理图设计使用；其次，要求输出为 JSON 格式，包含 circuit_name (电路名称)和 components (元器件列表)两个顶层字段，且该阶段只规划器件，不输出 connections 连线信息，连线关系留待 SPICE 网表生成阶段处理；再次，要求 JSON 中的 search_query 字段使用通用英文专业术语或经典芯片型号，以便与 KiCad 库中的英文标识和 ChromaDB 检索文本保持一致。对于特定功能类别的元器件，Prompt 还设置器件选择优先级；针对不同电路类型，则嵌入明确的拓扑约束。以 Sallen-Key 低通滤波器为例，Prompt 规定仅规划 R1、R2 (滤波电阻)和 C1、C2 (滤波电容)，运放采用单位增益配置，避免额外反馈网络电阻，从而降低 LLM 生成不合理拓扑的概率。

此外，本文基于关键词字典实现了电路类型自动识别，覆盖滤波器、音频放大器、BJT 放大器、运放放大器、稳压器、LED 恒流驱动、整流器、振荡器、电源等 10 种常见电路类型。识别算法采用两级优先级机制：首先检查高优先级关键词；若未命中则通过普通关键词评分进行匹配。识别的电路类型决定了后续 SPICE 网表生成阶段加载的专家 Prompt 模板，以及仿真类型的选择。

4.3. 元器件库匹配

规划阶段输出的每个元器件包含一个 search_query 字段，该字段作为检索模块的输入。检索按照第 3.4 节所述的两级策略执行：首先查询别名注册表进行精确匹配，若命中则通过 ChromaDB 直接返回库中对应器件的完整信息(lib_id、引脚列表、符号 S-表达式源代码)；若未命中，则将查询文本通过嵌入模型编码为 384 维向量，在 ChromaDB 中检索 Top-15 个候选，经 Keywords 重排序后返回最优匹配。

匹配成功后，每个元器件获得其在 KiCad 库中的具体标识和引脚配置。这些信息以 retrieved.json 文件保存和传递，每个条目包含 uid (元器件位号)、lib_id (库标识符)、planned_value (LLM 规划时推荐的参数值)、pins (引脚列表)、raw_symbol_definition (符号 S-表达式)以及 source_library (来源库文件名)。该 JSON 文件连接元器件规划与网表生成两个阶段。

4.4. SPICE 网表生成

SPICE 网表生成是 LLM 在流程中的第二次主要调用。此时输入已包含细化后的设计需求、匹配完成的元器件清单和自动识别的电路类型，比初始需求具备更明确的器件与约束信息。系统根据电路类型加载对应的专家 Prompt 模板，每类模板包含 SPICE 网表结构描述、输出节点命名规范和常见错误规避规则。例如，模板统一要求输出节点命名为“OUT”，便于仿真控制块捕获输出信号；同时，System Prompt 还约束元器件实例化语句格式、模型引用方式以及不生成分析控制块等 SPICE 语法细节。生成时 temperature 设为 0.1，max_tokens 设为 2048，以提高输出稳定性并容纳较长网表。

系统接收到 LLM 输出的 SPICE 代码后执行两级后处理：首先通过正则表达式剥离可能残留的 Markdown 代码块标记和多余注释行，其次检查网表中是否存在重复元器件名称，若存在则自动添加后缀

重命名。清理后的 SPICE 代码与模型注入语句拼接为 Ngspice 可执行文件(.cir 格式)，供后续仿真验证使用。

4.5. 人工仿真验证与反馈修正

本文采用人在回路(Human-in-the-Loop)的协作验证方式。Ngspice 输出通常是数值化曲线和表格数据，判断电路是否满足设计需求需要结合电路指标和领域经验。因此，本文由设计者负责仿真结果解读与偏差诊断，LLM 则负责根据反馈修改网表。具体而言，系统生成 Ngspice 可执行网表文件(.cir)后，设计者使用批处理模式运行仿真；仿真类型根据电路类型配置，滤波器使用 AC 分析(decade 对数扫描，1 Hz 至 1 MHz)，放大器和稳压器使用 AC 分析配合瞬态分析，LED 驱动器使用直流工作点分析配合瞬态分析。仿真结果以文本文件形式输出，包含频率/幅度或时间/电压等数据。

仿真完成后，设计者使用 Ngspice 数据查看工具或绘图软件检查关键性能指标，包括滤波器的-3 dB 截止频率和通带增益，BJT 放大器的中频增益及频率拐点，Zener 稳压器的输出电压和纹波，运放放大器的增益与带宽，以及 LED 恒流驱动器的输出电流和电流纹波。若仿真结果超出可接受偏差范围，设计者以自然语言形式编写反馈文本，明确指出偏差项、当前实测值、目标期望值和可能的修正方向。随后，系统将反馈文本、上一轮 SPICE 网表和原始设计需求一并提交给 LLM，要求模型针对偏差描述修改网表，并尽量保持已经满足要求的部分不变。

上述过程重复执行，直到仿真结果满足设计指标或达到预设最大迭代次数(8 轮)。若达到最大轮次仍未收敛，则选取迭代过程中偏差最小的网表作为最终输出。实验统计显示，平均收敛轮次为 1.5 轮，说明多数测试任务可在少量反馈后完成修正。人工在环模式的优势在于，设计者可以利用专业知识判断模糊或综合性指标，减少纯自动评估可能带来的误判；同时，每轮修改都有明确的偏差来源和反馈记录，便于后续审查与复现。

5. 实验与结果分析

5.1. 实验环境与设置

实验平台与模型。实验选用 DeepSeek-V3.2 作为主要 LLM 后端，该模型具有 671 B 参数总量，在代码生成和结构化输出任务上表现较好；同时选用 GPT-4o 和 Qwen3.5-27B 作为对比模型，以验证方法的模型无关性。模型通过兼容 OpenAI 协议的 API 调用。软件环境包括 Python 3.10、ChromaDB 0.4.x、Sentence-Transformer 嵌入模型 paraphrase-multilingual-MiniLM-L12-v2、Ngspice 40+和 KiCad 8，其中 Ngspice 用于 SPICE 仿真，KiCad 用于原理图的最终渲染与查看。

测试集与评价指标。实验覆盖 5 类典型模拟电路，共 10 个测试任务：(1) Sallen-Key 低通滤波器(截止频率 600 Hz 和 1000 Hz 各一例)；(2) BJT 共射放大器(增益 40 dB 和 30 dB 各一例)；(3) Zener 稳压器(BJT 扩流型和基础型各一例)；(4) 运放放大器(反相放大 40 dB 和同相放大 20 dB 各一例)；(5) LED 恒流驱动器(20 mA 和 10 mA 各一例)。每个任务独立运行 3 次并取平均结果。评价指标包括仿真通过率(Simulation Pass Rate, SPR)、目标偏差(Target Deviation, TD)、迭代次数(Iter)和模型匹配率(Model Match, MM)，分别衡量网表是否满足任务约束、仿真实测值与设计目标的相对偏差、达到收敛所需的平均仿真轮数，以及检索到的元器件拥有可用 SPICE 模型的比例。

5.2. 元器件规划与检索评估

首先评估元器件规划与检索阶段的效果。在包含 30 个电路设计任务的规划级测试集上，本文方法取得了以下结果：

Table 1. Planning and retrieval evaluation results across three LLMs
表 1. 不同 LLM 的元器件规划与检索评估结果

模型	元器件召回率(%)	检索命中率 RHR (%)	格式通过率 FPR (%)
DeepSeek-V3.2	65.3	98.5	100.0
Qwen3.5-27B	66.7	97.7	100.0
GPT-4o	63.4	97.4	100.0
三模型平均	65.1	97.9	100.0

由表 1 可见，三个模型的格式通过率均为 100%，说明 Prompt 中的 JSON 格式约束能够稳定约束元器件清单输出。检索命中率(RHR)在三个 LLM 上分别为 98.5%、97.7%和 97.4%，平均达到 97.9%，表明两级检索策略能够较稳定地将 LLM 生成的抽象器件描述映射到 KiCad 库中的可用器件，且检索表现主要受向量库覆盖范围和匹配策略影响，而不是完全依赖某一特定 LLM 后端。元器件召回率在三个模型上分别为 65.3%、66.7%和 63.4%，平均为 65.1%；该指标表示 LLM 生成的 BOM 中与参考设计一致的元器件占比，剩余约 35%的差异并不必然代表功能缺失，人工审查发现不少差异来自同功能器件的不同命名或合理替代。因此，后续改进重点之一是进一步规范 search_query 的命名方式。

从模型差异看，Qwen3.5-27B 取得了 66.7%的最高召回率，略高于 DeepSeek-V3.2 (65.3%)和 GPT-4o (63.4%)。这一结果提示，在结构化 Prompt 引导下，元器件规划任务并不单纯由模型规模决定，指令遵循能力和中文需求理解能力也会影响最终规划质量。

5.3. 检索方式对比实验

为验证两级检索策略中精确匹配的关键作用，本文对比了完整检索(精确匹配 + 向量检索)与仅使用向量检索(去除精确匹配环节)的效果，如表 2 所示。

Table 2. Impact of exact match on retrieval performance
表 2. 精确匹配对检索性能的影响

指标	完整检索	仅向量检索	降幅
元器件召回率(%)	65.3	53.5	−18.1%
检索命中率 RHR (%)	98.5	96.7	−1.8%

去除精确匹配后，元器件召回率从 65.3%降至 53.5%，降幅为 18.1%，说明精确匹配对基础无源器件(R、C、L 等)和经典芯片型号(LM2904、NE555 等)的定位具有重要作用。RHR 仅下降 1.8%，表明向量检索仍能找到语义相关候选；但召回率下降说明这些候选与参考设计期望器件在型号层面可能存在偏差。

5.4. 迭代修正消融实验

为评估人工反馈迭代对最终设计质量的贡献，本文设计了三组消融实验，结果如表 3 所示。

Table 3. Iterative refinement ablation results (DeepSeek-V3.2, 10 tasks × 3 runs)
表 3. 迭代修正消融实验结果(DeepSeek-V3.2, 10 tasks × 3 runs)

实验配置	仿真通过率 SPR (%)	目标偏差 TD (%)	平均迭代次数
G0：完整流程(专家引导 + 迭代修正)	86.7	3.1	1.5 ± 1.2
G1：去除专家引导(通用生成器)	3.3	--	--
G2：去除迭代修正(单次生成)	46.7	2.8	1.0 ± 0.0

消融实验结果揭示了两个关键发现。其一，专家引导是提高网表可仿真性的关键条件。G1 组将领域特定专家 Prompt 替换为通用 SPICE 代码生成器后，仿真通过率从 86.7%降至 3.3%，10 个任务中仅 LED 10 mA 恒流驱动任务(test_10)在 3 轮运行中有 1 轮通过。失败案例主要包括三类：节点命名不规范，LLM 使用数字编号而非功能名称作为网表节点，导致仿真控制块无法正确捕获输出信号；拓扑连接错误，例如运放电源引脚遗漏或 BJT 偏置网络连接不当；仿真控制块缺失或包含与 Ngspice 批处理模式不兼容的命令。

其二，反馈迭代能够显著提升参数敏感任务的通过率。G2 组禁用反馈迭代后，仿真通过率从 86.7%降至 46.7%。单次生成通过的任务主要集中在运放放大器类，这类电路拓扑确定、阻值比值与增益关系明确，LLM 较容易一次生成合适参数；相比之下，滤波器和 LED 恒流驱动器对参数更敏感，单次生成更容易出现截止频率或输出电流偏差。引入人工反馈后，设计者通常通过 1 至 2 轮修正即可调整关键电阻取值，使参数敏感任务的通过率明显提高。迭代开销统计显示，成功收敛的平均轮次为 1.5 轮；完整流程的平均 Token 消耗为每轮约 8276 个 Token、每个成功电路约 9549 个 Token，整体计算成本仍处于可接受范围。

5.5. 典型案例分析

案例一：运放反相放大器(约束饱和型)。设计需求为“使用 LM2904 设计的反相放大器，增益 40 dB，输入电阻 1 kΩ”。该电路拓扑较为确定，闭环增益满足 $A_v = -R_f / R_{in}$，输入电阻对应 R_{in}，因此可由指标直接得到 $R_{in} = 1\text{K}\Omega$ 和 $R_f = 100\text{K}\Omega$。完整流程在第 1 轮生成反相放大拓扑：LM2904 的 3 脚(+IN)接地，2 脚(-IN)通过 1 k 电阻接输入信号并通过 100 k 电阻接 1 脚(OUT)，8 脚接+12 V 电源，4 脚接地。Ngspice AC 仿真测得中频带增益为 39.99 dB，与 40.00 dB 目标的偏差为 0.024%，3 轮测试全部通过。该案例说明，对于解析关系明确的电路，LLM 在给定拓扑规则后更容易一次生成满足指标的参数。

案例二：BJT 共射放大器(偏置敏感型)。设计需求为“使用 2N3904 设计的共射放大器，增益 30 dB”。共射放大器的增益受集电极电阻、发射极退化电阻和晶体管跨导共同影响，同时对偏置网络设定的静态工作点较为敏感。完整流程在 3 轮测试中有 1 轮通过(增益 28.9 dB，偏差 3.6%)，其余两轮因偏置不当导致输出削波或增益不足。该案例反映了本文方法的边界：对于涉及半导体非线性工作点的问题，仅用自然语言反馈(如“增益偏低，请增大集电极电阻”)难以准确表达多个偏置电阻的联动调整。后续可引入自动化工作点校验脚本，在每次迭代后提取 BJT 的 V_{ce} 和 I_c 并反馈给 LLM。

案例三：LED 恒流驱动器(反馈敏感型)。设计需求为“基于 2N3904 的 LED 恒流驱动电路，输出电流 20 mA”。该电路的难点在于电流检测电阻 R_{sense} 需要与晶体管 V_{be} (约 0.7 V)和 LED 正向压降(约 2 V)匹配。完整流程首轮将 R_{sense} 设为 65 Ω，仿真结果显示 LED 电流为 12.3 mA，偏差 38.5%。设计者反馈：“当前 LED 电流仅为 12.3 mA，需调整至 20 mA。建议将发射极检测电阻从 65 欧减小至约 33~35 欧。”LLM 据此将 V_{be} 修改为 33 Ω 后，第 2 轮仿真测得 LED 电流为 19.8 mA，偏差 0.88%，达到收敛条件。相比之下，单次生成(G2 组)缺少反馈修正，在 test_9 上全部失败。该案例说明，定量反馈有助于 LLM 定位需要调整的元件和取值方向。

5.6. 与纯 LLM 直接生成的对比

Table 4. Full-pipeline comparison of different planning strategies
表 4. 不同规划策略的全流水线对比

方法	SPR (%)	TD (%)	模型匹配率 MM (%)
本文方法	86.7	7.3	98.3
CoT 思维链提示	86.7	19.2	98.8
自主反思提示	90.0	26.7	92.6

为评估本文方法整体的有效性，表 4 将完整流程与几种通用 Prompt 策略进行了对比。

表 4 显示，CoT [14]和 Self-Reflection [15]等通用推理策略在 DeepSeek-V3.2 上取得了与本文方法相近(CoT 为 86.7%)或略高(Self-Reflection 为 90.0%)的仿真通过率，但其目标偏差(TD)分别为本文方法的 2.6 倍和 3.7 倍。这说明仿真通过率(SPR)不能单独反映电路设计质量。通用推理方法可能提高生成网表可仿真的概率，但由于缺少需求细化中的参数保护和元器件库约束，生成电路与原始指标之间仍可能存在较大偏差。

本文方法的 SPR 并非最高，但 TD 为 7.3%，约为通用方法的 1/3 至 1/4。原因在于，需求细化中的参数保护、电路类型专用拓扑规则和两级检索中的精确匹配共同限制了生成空间，使网表更贴近原始设计意图。此外，本文方法保持了 98.3%的模型匹配率，而 Self-Reflection 可能在自主反思过程中引入未经库验证的器件型号，使 MM 降至 92.6%。这一结果进一步说明，元器件库约束有助于提升设计可执行性。

6. 结论与展望

本文提出了一种面向模拟电路生成的检索增强与仿真反馈方法，将 Prompt 驱动的元器件规划、精确匹配与向量检索结合的库匹配、电路类型专家 Prompt 引导的 SPICE 网表生成，以及人工 Ngspice 仿真反馈串联起来，形成从自然语言需求到可执行网表的轻量级流程。在元器件库信息处理方面，本文实现了 KiCad 符号库 S-表达式解析、结构化描述文本构建、ChromaDB 向量存储和两级检索流程，使 LLM 生成的抽象器件描述能够映射到具有库标识、引脚和符号定义的可用器件上。

实验结果表明，精确匹配与向量检索结合的策略在不同 LLM 后端上均达到 97%以上的检索命中率，其中 DeepSeek-V3.2 为 98.5%；去除精确匹配后，元器件召回率下降 18.1%，说明别名注册表对基础器件和经典型号的定位具有明显作用。在电路生成与验证方面，领域专家 Prompt 和人工反馈均对结果有显著影响：去除专家引导后，仿真通过率从 86.7%降至 3.3%；去除迭代修正后，仿真通过率降至 46.7%；完整流程平均需要 1.5 轮迭代即可收敛。与 CoT、Self-Reflection 等通用推理策略相比，本文方法的通过率并非最高，但目标偏差更低，说明库约束、拓扑规则和人工反馈对提升指标贴合度具有实际意义。

本文方法仍存在一定局限：对于偏置敏感的模拟电路，自然语言反馈难以精确表达半导体工作点的定量调整需求；当前流程依赖设计者解读仿真结果并诊断偏差，对非专业用户不够友好；测试电路以中等复杂度为主，尚未覆盖锁相环、ADC/DAC、开关电源等混合信号或强非线性系统；元器件库和 SPICE 模型库也需要持续扩充。后续工作可进一步引入 LLM 辅助的仿真结果自动诊断模块，将 Ngspice 数值输出转换为结构化反馈建议；扩展元器件库和 SPICE 模型库，纳入更多行为级宏模型；探索更轻量的嵌入模型和本地化部署方案；并进一步连接 PCB 布局布线工具，扩展到从自然语言需求到 PCB 设计文件的生成流程。

基金项目

广东省重点建设学科科研能力提升项目(2025ZDJS121、2024ZDJS136)；珠海市基础与应用基础研究项目(2320004002789)；珠海科技学院“三个层次”人才培养项目(江赛标)。

参考文献

[1] 刘睿之, 李捷鹏, 王睿, 等. 人工智能辅助的芯片物理设计方法综述[J]. 计算机研究与发展, 2026, 63(7): 1643-1669.

[2] Fu, Y., Zhang, Y., Yu, Z., Li, S., Ye, Z., Li, C., *et al*. (2023) GPT4AIGChip: Towards Next-Generation AI Accelerator Design Automation via Large Language Models. 2023 *IEEE/ACM International Conference on Computer Aided Design* (*ICCAD*), San Francisco, 29 October-2 November 2023, 1-9. https://doi.org/10.1109/iccad57390.2023.10323953

[3] 顾雨竹, 张海磊, 黄玮. 人工智能在集成电路设计全流程中的应用研究[J]. 信息化研究, 2025, 51(6): 153-157.

[4] 刘泽垣, 王鹏江, 宋晓斌, 张欣, 江奔奔. 大语言模型的幻觉问题研究综述[J]. 软件学报, 2025, 36(3): 1152-1185.

[5] Vungarala, D., Alam, S., Ghosh, A. and Angizi, S. (2024) SPICEPilot: Navigating SPICE Code Generation and Simulation with AI Guidance. 2024 *IEEE International Conference on Rebooting Computing* (*ICRC*), San Diego, 16-17 December 2024, 1-6. https://doi.org/10.1109/icrc64395.2024.10937006

[6] 李子骏, 肖辉, 李雪峰. 面向知识密集型任务的检索增强生成技术综述[J]. 微电子学与计算机, 2025, 42(10): 48-65.

[7] 黄俊, 翁振宇, 黄智聪. 基于大语言模型的 PCB 自动化设计智能体[J]. 电力电子技术, 2026, 60(7): 169-178.

[8] Lin, F., Li, X., Lei, W., Rodriguez-Andina, J.J., Guerrero, J.M., Wen, C., *et al*. (2025) PE-GPT: A New Paradigm for Power Electronics Design. *IEEE Transactions on Industrial Electronics*, **72**, 3778-3791. https://doi.org/10.1109/tie.2024.3454408

[9] 芦存博, 左璇, 金博, 等. 大模型赋能的工业软件在智能制造领域的应用研究与探索[J]. 数字化转型, 2025, 2(11): 41-51.

[10] 张犬俊, 谢杨, 房春荣, 虞圣呈, 赵源, 陈振宇. 检索增强生成在软件工程中的应用综述[J]. 软件学报, 2026, 37(3): 1316-1339.

[11] Hasan, K.S.A., Raiyan, S.R., Alvee, H.M. and Sadik, W. (2026) CircuitLM: A Multi-Agent LLM-Aided Design Framework for Generating Circuit Schematics from Natural Language Prompts. https://arxiv.org/abs/2601.04505

[12] 曾怡, 陈容, 张德, 等. 基于机器学习技术的智能布局方法综述[J/OL]. 微电子学, 1-9. https://link.cnki.net/urlid/50.1090.TN.20260330.1114.003, 2026-08-03.

[13] 王乃钰, 叶育鑫, 刘露, 凤丽洲, 包铁, 彭涛. 基于深度学习的语言模型研究进展[J]. 软件学报, 2021, 32(4): 1082-1115.

[14] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., *et al*. (2022) Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems* 35, New Orleans, 28 November-9 December 2022, 24824-24837. https://doi.org/10.52202/068431-1800

[15] Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K. and Yao, S. (2023) Reflexion: Language Agents with Verbal Reinforcement Learning. *Advances in Neural Information Processing Systems* 36, New Orleans, 10-16 December 2023, 8634-8652. https://doi.org/10.52202/075280-0377