arXiv:2603.17204v1 [cs.CL] 17 Mar 2026

CODMAS：用于结构化RTL优化的辩证式多智能体协作框架

Che-Ming Chang*[1], Prashanth Vijayaraghavan*[2], Ashutosh Jadhav[2], Charles Mackin[2], Vandana Mukherjee[2], Hsinyu Tsai[2], Ehsan Degan[2]

[1]台湾大学， [2]IBM研究院 b09901156@ntu.edu.tw、prashanthv@ibm.com、ashutosh@us.ibm.com、charles.mackin@ibm.com、vandana@us.ibm.com、htsai@us.ibm.com、edehgha@us.ibm.com

摘要

优化寄存器传输级（RTL）代码是电子设计自动化（EDA）中提升功耗、性能和面积（PPA）的关键步骤。我们提出了CODMAS（通过辩证式多智能体系统实现协作优化），这是一个结合结构化辩证推理、领域感知代码生成和确定性评估的框架，用于实现自动化RTL优化。CODMAS的核心是两个辩证智能体：阐述者受到橡皮鸭调试的启发，负责阐述分步变换计划并揭示潜在假设；假设伙伴负责预测结果，并协调预期行为与实际行为之间的偏差，以指导有针对性的改进。这些智能体指导领域特定编码智能体（DCA）生成架构感知的Verilog编辑，并指导代码评估代理（CEA）验证语法、功能以及PPA指标。我们推出RTLOPT，这是一个包含120组三元Verilog数据（未优化版本、已优化版本和测试平台）的基准测试，用于评估流水线化和时钟门控变换。在专有和开源大语言模型上，与强提示和智能体基线相比，CODMAS使流水线化的关键路径延迟降低 ∼25%，使时钟门控的功耗降低∼22%，同时减少功能和编译失败。这些结果表明，结构化多智能体推理能够显著增强自动化RTL优化，并扩展至更复杂的设计和更广泛的优化任务。

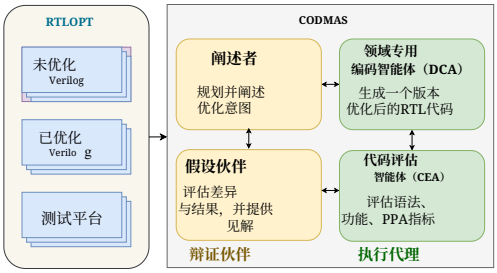


图1：CODMAS框架概览，展示智能体之间的辩证互动以及RTL代码的迭代优化。

1 引言

现代芯片设计日益复杂，加速了人工智能 (AI) 在电子设计自动化 (EDA) 工作流程中的集成，减少了对人工投入的依赖，并使设计周期更短。尽管人工智能已

在逻辑综合（Pei 等人，2023年）和布局（Lai 等人，2023年）等任务中取得显著成效，但硬件描述语言（HDL）的生成与优化仍然面临挑战。寄存器传输级（RTL）代码通常使用 Verilog 或 VHDL 编写，需要经过仔细且针对性调优的变换，才能满足功耗、性能和面积（PPA）约束。流水线化和时钟门控等变换对于提升性能和能效尤为关键，但这些变换既耗时、容易出错，又要求深厚的领域专业知识。商业电子设计自动化工具为某些 RTL 优化提供了自动化支持，但缺乏对设计意图的明确推理，也难以支持早期架构变换。现有的基于学习的方法要么侧重于硬件描述语言语法生成（Thakur 等人，2024a；Akyash 等人，2025年；Yu 等人，2025年；Liu等人，2024b；Chang 等人，2023年；Thakur 等人，2024b；Ho 等人，2024），要么依赖启发式搜索或局部重写（Thorat 等，2024；Yao 等人，2024；DeLorenzo 等人，2024），因此难以在不同设计间泛化，也难以捕获全局架构模式。此外，RTL 优化数据集仍然稀缺，限制了基于学习的方法的可复现性、基准测试和系统评估。

为解决这些不足，我们推出 RTLOPT，这是一个

*两位作者对本研究的贡献相同。本研究部分内容是在IBM研究院实习期间完成的。

专为寄存器传输级优化设计的精选基准测试。它包含 120 组三元组 Verilog 代码，每组包括一个未优化设计、一个优化后的对应版本（通过流水线化或时钟门控实现），以及一个关联的测试平台。这种组织方式既支持功能验证，也便于对功耗、性能和面积指标进行定量评估。该数据集涵盖多种设计模式和复杂度等级，为研究规模的实验和模型评估提供了易于处理且具有代表性的基础。在此基准测试的基础上，我们提出 CODMAS，一个用于自动化 RTL 优化的多智能体框架，将 辩证推理 融入优化循环。 阐述者智能体将优化意图转化为语言表达，而 假设伙伴 则评估预期结果与实际结果之间的差异。这些推理智能体与 领域专用编码智能体（*DCA*）和 代码评估代理（*CEA*）协同工作，迭代优化 RTL 设计，在保持功能正确性的同时改进功耗、性能和面积指标。图1展示了这一闭环交互过程。我们的主要贡献包括：

RTLOPT：基于 Verilog 的流水线化和时钟门控基准优化数据集。
CODMAS：一种多智能体框架，结合辩证推理、领域知识指导的代码生成以及面向 RTL 优化的功耗、性能和面积评估。
实证验证：结果表明，该方法在不同模型和优化场景中均能持续改进，凸显了结构化推理在自动化RTL优化中的有效性。

2 RTLOPT：HDL 优化数据集

为了评估大语言模型在 HDL 代码优化方面的表现，我们从实现优化方法的 GitHub 仓库中收集数据，重点关注 Verilog。由于流水线化和时钟门控等技术并不总是有明确标注，我们使用了“(pipelining OR clock gating) + verilog”等搜索词（例如“pipelining verilog”或“pipelined verilog”）来识别相关仓库。数据收集过程如下：我们筛选了所有公开可用的仓库，选取其中自包含的 Verilog 文件，以减少跨文件依赖并简化分析。随后，我们提取了配套的测试平台，并使用标记级相似度和 AST结构哈希对收集到的模块进行去重，以减少近似重复项，并降低与公开训练语料库的重叠。

RTLOPT 数据集统计	
# 流水线化代码三元组	70
# 时钟门控代码三元组	50
功耗范围（nW）	$\sim 1 - \sim 19,000$
面积范围（μm^2）	$\sim 1 - \sim 18,000$
延迟范围（ns）	$\sim 15 - \sim 1,600$

表 1：用于评估流水线化和时钟门控优化的 RTLOPT 数据集概述。

为了专门针对流水线化和时钟门控优化，我们人工审查了筛选出的 Verilog 文件，检查其中是否有代码或描述表明采用了这些优化模式。当缺少优化细节时，我们会生成未优化版本（即未采用流水线化或时钟门控）及相应的测试平台，方法包括修改现有描述或编辑大语言模型生成的代码（例如 DeepSeek 模型（Liu等人，2024年a））。所有生成或编辑的示例随后均经过综合和仿真验证，以确保其功能等价性并带来可测量的指标提升；同时，我们对其进行了规范化处理，以尽量消除未优化版本与已优化版本之间的风格线索。重要的是，未优化版本是独立获得的，而非通过机械地劣化已优化代码生成，从而避免了逆向工程造成的伪影。

这一过程形成了 RTLOPT 数据集，其中包含用于评估的三元组（未优化代码、已优化代码、测试平台），涵盖算术、控制和混合模块等设计，确保多样性。尽管规模较小（120 个三元组），RTLOPT 的规模与现有代码评估基准相当或更大，并且旨在优先考虑真实的变换和可复现的评估，而非单纯追求规模。当前版本的数据集侧重于流水线化和时钟门控，但其结构和工具旨在支持未来扩展至重定时、资源共享和有限状态机重构等变换。
RTLOPT 是一个面向社区扩展的种子基准。数据集统计信息见表1 。

3 方法

我们提出了 CODMAS（面向结构化寄存器传输级的辩证多智能体系统协同优化），一种用于自动化 RTL 优化的多智能体框架。
CODMAS 基于两点洞见：（1）互补智能体之间迭代的结构化推理能够提升优化质

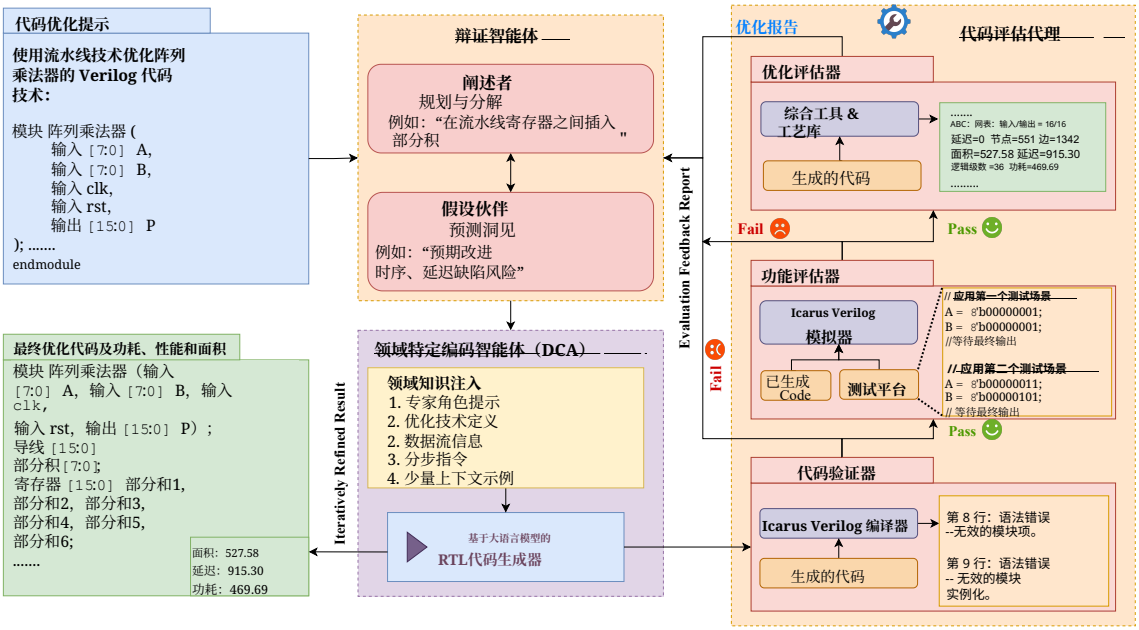


图 2：CODMAS 完整架构示意图，展示了辩证智能体（阐述者与假设伙伴）、执行代理（领域专用编码智能体与代码评估代理），以及迭代式寄存器传输级优化循环（例如流水线阵列乘法器）。该示例重点展示了流水线阵列乘法器的优化过程，不同组件提供的反馈推动设计逐步改进，从而提升功耗、性能和面积指标。

量；（2）领域专用的架构知识必须指导基于大语言模型的 RTL 变换。该系统由四个专用智能体组成：两个辩证推理智能体（阐述者和假设伙伴）以及两个执行代理（领域专用编码智能体（DCA）和代码评估代理（CEA））。与传统的基于大语言模型的流程不同，CODMAS 将规划、假设形成、代码生成和确定性评估整合到一个闭环反馈中。这种结构支持可解释且注重性能的优化，并可推广至 RTLOPT 中最初的 120 个 Verilog 三元组之外，从而应用于更大型设计、跨文件模块和多种 RTL 变换。

3.1 辩证推理智能体

辩证推理智能体在CODMAS 中作为互补协作者，参与一种结构化、受结对编程启发的优化工作流。不同于典型的单智能体或单体式推理流程，RTL 优化受益于明确区分设计阐述与假设生成。因此，两个智能体都会进行反思式设计思考，但承担不同的认知角色：**阐述者**专注于分解和规划，而**假设伙伴**擅长预测性推理和诊断性推断。它们的迭代交互形成了一个闭环辩证过程，其中阐述、批判和假设细化共同引导执行代理，生成在功能上正确且性能已优化的 RTL。

3.1.1 阐述者

Articulator 是系统的规划与语言表达模块。受橡皮鸭调试法和结构化推理的启发，它会分析未优化的寄存器传输级设计，然后生成逐步的变换计划，使之与目标优化目标相一致，例如流水线化或时钟门控。每一步都清晰易懂，并旨在保持功能正确性，从而实现可追溯的优化流程。程序化推理使 Articulator 能够将复杂变换分解为有序操作，同时揭示潜在的设计假设。这些假设以结构化形式表达，供人类设计人员和执行代理使用。例如，它可能建议在关键计算阶段之间插入流水线寄存器，或调整延迟对齐逻辑以保持正确性。该阐述计划为假设形成、代码生成和迭代改进提供了共享的语义支架；在不同假设提案之间，它基本保持稳定，从而能够在统一的电路模型下持续评估各种变换方案。

3.1.2 假设伙伴

假设伙伴与 Articulator 协同工作，利用溯因和基于模型的推理，预测计划中的变换对功能正确性和 PPA指标的影响。与传统代码生成方法不同，它会在生成任何优化代码之前开始推理，并基于 Articulator 所提供的、带有性能标注的电路表示形成假设。在固定的结构解释下，假设伙伴提出面向指标的变换，预测性能提升，识别功能或结构方面的隐患，并指导纠正策略。当综合或仿真反馈显示结果存在偏差时，该智能体会修订先前的假设，归因失败原因，并提出针对性改进。这一迭代推理循环将验证感知融入优化流程，确保代码生成以可操作且符合性能目标的洞见为指导。这种职责分离使系统能够在共享设计模型下系统地探索多个候选变换，避免过早选定次优方向。如第5.2节所示，将这些角色合并为一个智能体会降低收敛稳定性和最终指标提升，说明这种辩证结构对于寄存器传输级优化不可或缺。

3.2 执行代理

CODMAS 中的执行代理通过生成候选寄存器传输级代码并提供严格反馈，将辩证智能体的推理付诸实践。辩证智能体负责规划和预测性推断，执行代理则将这些见解转化为可执行代码并进行确定性评估。该层由用于生成已优化寄存器传输级代码的特定领域编码智能体 *(DCA)* 和代码评估代理 *(CEA)*组成，前者负责代码生成，后者负责语法、功能和功耗、性能和面积评估。二者共同构成紧密耦合的循环，迭代优化寄存器传输级设计。

3.2.1 特定领域编码智能体 (DCA)

DCA 将阐述者的变换计划和假设伙伴预测的结果转化为首轮优化后的 Verilog 代码。它通过领域知识注入器（*DKI*）采用多方面的提示策略：(a) 将大语言模型设定为专家级 RTL 设计师，指示其应用流水线化或时钟门控等优化技术（角色提示）；(b) 嵌入 K与阐述计划和预测结果相对应的未优化与已优化 Verilog 代码示例（基于示例的指导）；(c) 纳入使用 Pyverilog 提取的数据流图所提供的结构上下文（工具辅助的上下文）。注入的数据流图并非仅提供原始 RTL，而是以受限模式编码寄存器级、组合逻辑锥、控制依赖和时钟使能，使模型能够明确推理流水线深度、级间不平衡以及安全的门控机会。例如，在流水线化任务中，该图会展示两个寄存器之间较长的组合路径，从而引导模型插入中间寄存器，同时保持控制对齐。

借助这一增强后的提示，DCA 会生成在功能上等效且具备架构感知能力的候选设计。CEA 会评估此代码，并将反馈返回给辩证智能体：阐述者修订变换步骤，假设伙伴更新预测结果，DCA 则将这些调整纳入下一轮迭代。此循环不断重复，直至实现功能正确性并达到目标功耗、性能和面积改进。

3.2.2 代码评估智能体（CEA）

CEA 使用开源电子设计自动化工具（Icarus Verilog 用于编译，GTKWave 用于仿真，Yosys 用于综合和时序分析）提供确定且可验证的评估。它从三个维度评估设计：语法正确性、功能等价性和优化质量。代码验证器 确保 Verilog 代码可编译，并生成结构化错误消息以指导修正；功能评估器 使用针对流水线化等变换进行调整的测试平台对设计进行仿真，以检测行为不匹配或性能回退；优化评估器 则综合设计，提取功耗、性能和面积指标，以支持面向性能的改进。CEA 汇总生成一份统一的反馈报告，供辩证智能体使用：语法错误会促使其修订变换计划，功能不匹配会触发假设更新，而未达最优的功耗、性能和面积指标则会指导其调整优化策略。更新后的计划和假设会发送回 DCA，由其生成改进后的寄存器传输级代码。

3.3 集成优化工作流

算法1概述了 CODMAS 优化流程。给定未优化的寄存器传输级 C_0、目标 G （例如改善功耗、性能和面积），以及迭代次数上限 T，阐述者 会生成结构化的转换计划，而

假设伙伴 则预测预期功能行为和功耗、性能和面积的结果。数据流有向无环图支持结构推理，领域知识注入器 则整合计划、假设和结构上下文，构造提示词，供大语言模型生成候选的已优化寄存器传输级。由代码验证器、功能评估器和优化评估器组成的代码评估代理（*CEA*）负责评估语法、功能等价性和功耗、性能和面积。反馈会被纳入辩证循环：语法错误会促使调整计划，功能不匹配会更新假设，而不理想的功耗、性能和面积指标则会引导策略改进。该循环会持续进行，直至实现优化目标或达到迭代次数上限，从而提供可追溯、兼顾验证且以性能为导向的优化工作流程。

4 实验

我们评估 CODMAS 在 RTL 优化方面的表现，并介绍实验设置、基线和指标。本研究探讨三个问题：**RQ1：C 的有效性ODMAS：**与其他基线相比，该框架如何提升功耗、性能和面积？**RQ2：各组件的贡献：**辩证智能体和执行智能体对优化性能有何影响？**RQ3：迭代优化的优势：**多步推理和迭代优化如何影响不同设计的收敛速度和优化质量？

基线 我们将 CODMAS 与具有代表性的提示方法和基于智能体的寄存器传输级优化方法进行比较。基线包括**零样本** 提示，以及 **CoDes** (Vijayaraghavan 等人，2024年)、**ReAct** (Yao 等人，2023年) 和 **Reflexion** (Shinn 等人，2024年) 等中间推理方法，以及两阶段纠错—优化流程 **LLM-VeriPPA** (Thorat 等人，2024年)。所有系统均在专有模型（GPT-4o、GPT-3.5-turbo）和开源模型（Llama-3、DeepSeek-v2.5、Granite-34B-Code、CodeLLaMA-34B）上，采用相同的仿真和综合流程进行评估。基线配置的详细信息见附录D。尽管 ReAct、Reflexion 和 CoDes 等基线最初是为通用软件推理任务开发的，我们通过为它们提供与 CODMAS 相同的寄存器传输级解析、变换和综合反馈接口，使其适用于寄存器传输级优化。这样既能确保比较公平，又能保留这些模型原有的推理策略。

在硬件优化场景中。

指标 我们使用 Yosys 和基于 Liberty 的标准单元库进行综合与仿真，以原始设计为参照评估优化后的 RTL。面积（A）以 A/A_0表示，其中A_0 为基线；数值低于 1 表示面积有所降低。功耗（P）和时序（T）的改善以百分比增益表示，时序通过关键路径延迟（CPD）衡量，正增益表示 CPD 降低。失效率（FR）表示优化后未通过功能、综合或时序检查的设计所占比例；FR 越低，可靠性越高（见附录B）。所有优化后的设计均使用源自原始代码库或根据行为规范构建的模块级测试平台进行验证。虽然我们尚未纳入形式等价性检查，但我们的方法符合常见的工业级 RTL 优化流程，其中基于仿真的测试仍是关键验证机制之一。

5 结果

5.1（RQ1）CODMAS 的有效性

表2 总结了六种大语言模型在四种优化策略下的功耗、性能和面积表现。在所有模型中，CODMAS 始终带来最显著的提升：流水线化实现了超过 20% 的时序改善（ GPT-4o 上达到 25.5% ），而时钟门控平均降低功耗超过 19%，相比之下，所有基线的降幅都不到 10%。CODMAS 下的失效率保持在 30% 以下，而提示式和智能体基线通常超过 40% 至 50%，这表明语法、功能和功耗、性能和面积违规更为频繁。各模型的表现趋势进一步印证了这些发现。GPT-4o 在各项策略中均领先，在流水线化中实现了 $\sim 25\%$ 的时序改善，在时钟门控中实现了 $\sim 22\%$ 的功耗降低，且失效率低于 25%。开源模型 Granite-34b 和 CodeLlama-34b 在零样本或朴素提示下表现不佳，功耗、性能和面积增益有限，失效率高于 60%；但在 CODMAS 下，它们实现了最高 13% 的时序和功耗改善，且失效率低于 30%。Llama3 最具挑战性：基线模式使面积最多增加 18%，失效率超过 50%；而 CODMAS 将净面积影响降至 $A/A_0 \approx 1.03$ ，同时实现接近 20% 的时序改善。错误分析表明，基线失效源于语法（~ 40 %）、功能（$\sim 35\%$）和功耗、性能和面积（~ 25 %）问题，

流水线化

模型	零样本			提示/智能体			LLM-VeriPPA			CODMAS		
	A (↓)	T (↑)	FR (↓)	A (↓)	T (↑)	FR (↓)	A (↓)	T (↑)	FR (↓)	A (↓)	T (↑)	FR (↓)
GPT-4o	**0.985**	**11.4**	**54.2**	**0.982**	**15.2**	**49.6**	**0.986**	**15.4**	**39.7**	**0.960**	**25.5**	**19.5**
GPT-3.5-turbo	0.989	10.0	58.0	0.988	14.0	51.2	1.008	13.6	43.3	0.972	21.3	23.4
DeepSeek-v2.5	1.009	9.8	57.8	0.986	12.8	50.0	1.025	13.0	42.8	0.979	21.4	22.8
Llama-3	1.045	8.7	61.4	1.181	11.6	53.8	1.116	11.3	47.1	1.032	19.8	25.5
Granite-34b	**1.026**	3.9	65.2	**1.015**	5.3	**59.1**	**1.022**	6.6	57.7	**0.998**	10.5	**28.3**
CodeLlama-34b	1.035	**4.2**	**64.7**	1.036	**6.8**	60.7	1.039	**7.2**	**56.1**	1.030	**11.2**	29.5
人类	N/A			N/A			N/A			**0.848**	**45.6**	**0.0**

时钟门控

模型	零样本			提示/智能体			LLM-VeriPPA			CODMAS		
	准确率 (↓)	精度 (↑)	失败率 (↓)	准确率 (↓)	精度 (↑)	FR (↓)	A (↓)	P (↑)	FR (↓)	A (↓)	P (↑)	FR (↓)
GPT-4o	**1.023**	**7.8**	**52.5**	**1.010**	**9.1**	**46.8**	**1.009**	**9.3**	**38.9**	**0.999**	**21.7**	**21.8**
GPT-3.5-turbo	1.035	6.3	55.6	1.018	7.5	49.6	1.019	7.9	43.4	1.020	18.8	24.2
DeepSeek-v2.5	1.030	6.2	54.8	1.014	7.8	48.9	1.020	8.2	42.9	1.015	19.0	23.6
Llama-3	1.093	5.5	58.6	1.060	6.7	52.1	1.062	7.0	46.5	1.048	16.5	26.3
Granite-34b	**1.037**	3.0	63.0	**1.048**	**5.8**	59.8	**1.049**	**5.9**	**55.8**	**1.030**	**12.9**	**29.5**
CodeLlama-34b	1.054	**3.4**	**61.6**	1.053	5.5	**58.7**	1.050	5.4	55.3	1.042	10.6	31.3
人类	N/A			N/A			N/A			**0.925**	**30.4**	**0.0**

表2：流水线化和时钟门控（CG）的性能对比。流水线化报告面积（A）、时序（T）和失效率（FR）；时钟门控报告 A、功耗（P）和 FR。粗体表示最佳；↑越高越好，↓越低越好。所有改进均具有统计显著性（$p < 0.01$），依据表6中的标准差进行配对t检验。

这些问题均通过 CODMAS 的显式规划、假设引导推理和确定性评估得到缓解。总体而言，CODMAS 持续改善功耗、性能和面积表现，降低失效率，并使专有和开源大语言模型的优化过程更加稳定。

5.1.1 面积影响

流水线化会增加寄存器和握手逻辑，时钟门控则会增加门控单元和控制逻辑，因此即使时序或功耗有所改善，面积也往往会增加。在我们的实验中，CODMAS 将面积维持在接近基线的水平；GPT-4o 在两种优化下均实现了 $\sim 4\%$ 的面积缩减，同时显著改善了时序或功耗。面积变化通常较小，这归因于综合差异或轻微的结构调整。基线方法通常只有极小的面积变化，PPA 增益也有限；例如，Llama-3 在零样本流水线化中面积增加了$\sim 4.5\%$，时序仅提升了 $\sim 8.7\%$，且失效率较高（61.4%）。经 CEA 验证后，阐述者的变换和假设伙伴的预测侧重于改善 PPA，同时将面积变化置于次要位置。

5.2（RQ2）各组件的贡献 on

表3 展示了对 CODMAS 三个关键组件的消融研究：辩证智能体（DA）、主知识注入（DKI）和代码评估代理（CEA），并报告了 GPT-4o 和 Llama-3 的指标。完整的 CODMAS 流水线始终优于所有消融变体，体现了各模块的重要性：DA 协调结构化推理，DKI 根据设计意图为变换提供依据，CEA 则过滤无效或低质量的修改。

移除辩证智能体（无 *DA*）会导致性能下降幅度最大。GPT-4o 流水线化带来的时序收益从 25.5% 降至 12.9%，失效率则升至 38.5%–44.7%，凸显了辩证代理在结构化优化和基于假设的筛选中的作用。移除 CEA（无 *CEA*）后，失效率会上升（例如，GPT-4o 时钟门控的失效率从 21.8% 升至 32.7%），因为有缺陷的修改会被保留下来。省略 DKI（无 *DKI*）会降低优化质量和鲁棒性，尤其是在较小模型上：Llama- 3 流水线化带来的时序收益从 19.8% 降至 9.3%，失效率上升了 ~ 10 个百分点。这些结果证实，DA、DKI 和 CEA 各组件都能提供互补优势，对于实现稳定的寄存器传输级优化至关重要。

5.2.1 辩证代理消融实验

为了单独评估 CODMAS 中辩证式智能体设计的影响，我们将完整系统与

	FULL	不含 CEA	不含 辩证代理	不含 DKI
流水线化				
T（GPT-4o）	**25.5**	17.2	12.9	11.5
FR（GPT-4o）	**19.5**	31.0	38.5	32.3
T（Llama-3）	**19.8**	12.6	10.4	9.3
FR（Llama-3）	**25.5**	36.9	44.7	35.8
时钟门控				
P（GPT-4o）	**21.7**	14.4	9.5	11.1
FR（GPT-4o）	**21.8**	32.7	37.2	40.5
P（Llama-3）	**16.5**	10.2	6.8	7.2
FR（Llama-3）	**26.3**	35.5	41.3	43.2

表3：CODMAS 的组件级消融研究。DA：辩证智能体；DKI：领域知识注入；CEA：代码评估代理。

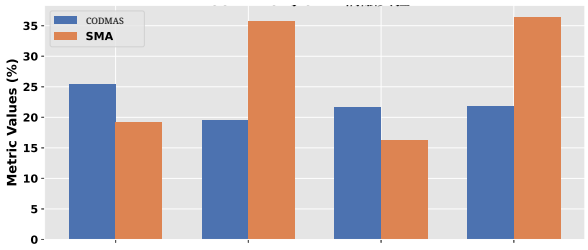


图3：将 CODMAS 与一种共享记忆多智能体（SMA）变体进行消融比较，其中 Articulator 和假设伙伴的角色合并。

一种替代架构进行比较：共享内存多智能体（SMA）变体，其中表达者和假设伙伴的角色合并，两个智能体共同解读并修改寄存器传输级（RTL），而不明确区分规划与预测推理。所有变体的综合调用次数均保持一致，以确保比较公平。

图中的结果3 表明，将这两个角色合并为共享内存多智能体系统，会持续降低性能提升并增加失效率。例如，对于 GPT- 4o，流水线化带来的时序改善从 25.5% 降至 19.2%，失效率则从 19.5% 升至 35.7%。类似地，时钟门控的功耗收益也有所下降，失效率上升。这些发现表明，明确区分 Articulator 与假设伙伴至关重要：阐述式规划提供了稳定的语义支架，而预测性假设则引导针对性变换。这种结构相辅相成，能够带来更高的指标提升，并使 RTL 优化更具可复现性。

5.3（RQ3）迭代优化的益处

为了评估迭代反馈对流水线化的影响，我们对简单和困难 任务进行了

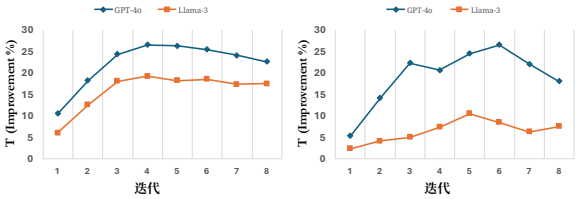


图 4：CODMAS 中迭代优化对简单（左）和困难（右）RTLOPT 问题的流水线化时序改善（%）的影响。早期迭代带来显著提升；后续迭代的收益则逐渐递减或表现不稳定，尤其是在更困难的案例中。

八次优化迭代（图4）。两种情形都显示，前三次迭代带来了显著提升，说明初期反馈捕捉到了最大的优化机会。简单任务在第 4 次迭代时趋于平稳，此后略有下降；困难任务则在第 5 至 6 次迭代左右达到峰值，之后出现波动，这反映了密集流水线以及相互冲突的变换假设。这些趋势凸显了迭代优化对于系统性提升性能的价值，同时也表明，过多迭代可能只带来微小或不稳定的收益。自适应停止或动态迭代策略可以减少不必要的计算，并防止复杂设计出现性能回退。

6 结论

在 RTL 设计中实现高效的功耗、性能和面积（PPA）是一项关键但极具挑战性的现代硬件设计任务。为此，我们推出了 RTLOPT——一个针对流水线化和时钟门控优化的基准测试，以及 CODMAS——一个融合辩证推理、领域特定代码生成和确定性评估的多智能体框架。阐述者和假设伙伴引导执行代理（领域专用编码智能体和代码评估代理），严格地生成并评估 Verilog 设计。实验结果表明，时序改善了 $\sim 25\%$，功耗降低了 $\sim 22\%$，失效率为 $< 30\%$；消融实验显示，所有组件和迭代优化对于实现稳健性能都至关重要。未来的研究方向包括扩充数据集、探索自适应迭代策略、扩展至其他 RTL 优化、利用检索增强提示、验证完整综合流程，以及引入自博弈或强化学习，以进一步提升优化效果。

局限性

尽管我们的框架推动了 RTL 优化的自动化，但仍存在若干局限。首先，CODMAS 目前主要针对流水线化和时钟门控变换进行了评估，其向更广泛 RTL 优化类别（例如重定时、资源共享和有限状态机重构）的泛化能力尚未得到充分验证。其次，尽管评估流程采用了确定性的电子设计自动化工具，但受工具运行时间以及需要反复执行综合查询的影响，其扩展到超大型工业设计的能力有限。第三，辩证推理智能体偶尔会生成过于笼统的变换计划，需要通过迭代优化加以完善，这表明系统仍依赖有原则的提示和针对具体任务的模板。第四，RTLOPT 是一个规模有限的初始基准测试，未来一项重要工作是扩充该基准，以涵盖工业级 RTL 编码风格的多样性及多文件层级结构。最后，由于功能等价性是通过标准测试平台而非穷尽式形式化技术来验证的，因此在少见情况下，细微的边界情形差异可能无法被发现。这些局限凸显了未来改进框架推理鲁棒性、数据集覆盖范围和可扩展性的机会。

伦理考量

尽管 RTLOpt 完全由采用公开许可证的 Verilog 代码构建而成，但将其集成到自动化的寄存器传输级优化工作流中会带来一些安全方面的考量。已有研究表明，大语言模型生成的寄存器传输级代码可能存在通用弱点枚举（CWEs）中所列的漏洞（Gadde等人，2024年）。此外，用于硬件描述语言生成的大语言模型可能容易受到数据投毒或后门攻击的影响；受损的训练数据可能导致生成不安全或恶意的电路组件（Mankali等人，2025年）。为降低这些风险，我们的系统通过生成明确的转换计划和假设，强调人工监督和可解释性，供专家设计人员审阅和批准。我们还进行严格的仿真和综合检查，以确保确定性验证，并检测意外的结构缺陷或安全漏洞。RTLOPT 中的所有模块均附有完整文档，注明来源和许可信息，以支持清晰的来源追踪。最后，我们建议，任何自动化部署……已优化的硬件都应接受额外的安全审计、形式化验证和人工审查，尤其是在安全攸关或安全关键型应用中。

参考文献

AI@Meta. 2024年 Llama 3 模型卡。

Mohammad Akyash、Kimia Azar 和 Hadi Kamali。2025年。Rtl++：用于寄存器传输级代码生成的图增强型大语言模型。*arXiv* 预印本 *arXiv:2505.13479*。

Kaiyan Chang、Ying Wang、Haimeng Ren、Mengdi Wang、Shengwen Liang、Yinhe Han、Huawei Li 和 Xiaowei Li。2023年。Chipgpt：自然语言硬件设计距离我们还有多远。*arXiv* 预印本 *arXiv:2305.14019*。

Matthew DeLorenzo、Animesh Basak Chowdhury、Va- sudev Gohil、Shailja Thakur、Ramesh Karri、Sid- dharth Garg 和 Jeyavijayan Rajendran。2024年。让每一步都发挥作用：使用蒙特卡洛树搜索的大语言模型高质量寄存器传输级代码生成。*CoRR*。Deepak Narayan Gadde、Aman Kumar、Thomas Nala- pat、Evgenii Rezunov 和 Fabio Cappellini。2024年。全人工，少智能：从形式化验证的视角看生成式人工智能。*arXiv*预印本 *arXiv:2403.16750*。Dario Garcia-Gasulla、Gokcen Kestor、Emanuele Parisi、Miquel Albert' i-Binimelis、Cristian Gutier- rez、Razine Moundir Ghorab、Orlando Montenegro、Bernat Homs 和 Miquel Moretó。2025年。TuRTLe：对用于寄存器传输级生成的大语言模型进行统一评估。*CoRR*。Ce Guo 和 Tong Zhao。2025年。ResBench：面向大语言模型生成的FPGA设计的资源感知型基准测试。载于高效加速器与可重构技术国际研讨会第*15*届会议论文集，第25–34页。Chia-Tung Ho、Haoxing Ren 和 Brucek Khailany。2024年。VerilogCoder：基于图规划和基于抽象语法树（AST）的波形追踪工具的自主Verilog编码智能体。预印本，arXiv:2408.08927。Yao Lai、Jinxin Liu、Zhentao Tang、Bin Wang、Jianye Hao 和 Ping Luo。2023年。ChiPFormer：通过离线决策Transformer实现可迁移的芯片布局。载于第*40*届国际机器学习会议（*ICML*）论文集，第202卷，第18346–18364页。PMLR。Aixin Liu、Bei Feng、Bing Xue、Bingxuan Wang、Bochao Wu、Chengda Lu、Chenggang Zhao、Chengqi Deng、Chenyu Zhang、Chong Ruan 等。2024a。DeepSeek-V3技术报告。*arXiv*预印本 *arXiv:2412.19437*。Mingjie Liu、Nathaniel Pinckney、Brucek Khailany 和 Haoxing Ren。2023年。VerilogEval：评估用于Verilog代码生成的大语言模型。载于*2023*年 *IEEE/ACM*计算机辅助设计国际会议（*ICCAD*）。IEEE。

Shang Liu、Wenji Fang、Yao Lu、Qijun Zhang、Hongce Zhang 和 Zhiyao Xie。2024b。RTLCoder：借助我们的开源数据集和轻量级解决方案，在寄存器传输级设计生成方面超越 GPT-3.5。预印本，arXiv:2312.08617。

Yao Lu、Shang Liu、Qijun Zhang 和 Zhiyao Xie。2024年。RTLLM：基于大语言模型的设计寄存器传输级生成开源基准。在第*29*届亚洲及南太平洋设计自动化会议（*ASP-DAC*）论文集，第722–727页。IEEE。

Lakshmi Likhitha Mankali、Jitendra Bhandari、Manaar Alam、Ramesh Karri、Michail Maniatakos、Ozgur Sinanoglu 和 Johann Knechtel。2025年。Rtl-breaker：评估大语言模型抵御HDL代码生成后门攻击的安全性。在*2025*年欧洲设计、自动化与测试会议（*DATE*），第1–7页。IEEE。

Mayank Mishra、MattStallone、Gaoyuan Zhang、Yikang Shen、Aditya Prasad、Adriana Meza Soria、Michele Merler、Parameswaran Selvam、Saptha Surendran、Shivdeep Singh、Manish Sethi、Xuan-Hong Dang、Pengyuan Li、Kun-Lung Wu、Syed Zawad、Andrew Coleman、Matthew White、Mark Lewis、Raju Pavuluri 和其他27位作者。2024年。Granite代码模型：面向代码智能的一系列开放基础模型。预印本，arXiv:2405.04324。

OpenAI。2024年。ChatGPT。https://chatgpt.com/。

Zehua Pei、Fangzhou Liu、Zhuolun He、Guojin Chen、Haisheng Zheng、Keren Zhu 和 Bei Yu。2023年。Al-phaSyn：利用高效蒙特卡洛树搜索进行逻辑综合优化。载于*2023*年 *IEEE/ACM* 国际计算机辅助设计会议（*ICCAD*）。IEEE。

Nathaniel Pinckney、Chenhui Deng、Chia-Tung Ho、Yun-Da Tsai、Mingjie Liu、Wenfei Zhou、Brucek Khailany 和 Haoxing Ren。2025年。全面的 Verilog 设计问题：用于评估大语言模型和智能体在 RTL 设计与验证方面表现的新一代基准数据集。*arXiv* 预印本 *arXiv:2506.14074*。

Baptiste Rozière、Jonas Gehring、Fabian Gloeckle、Sten Sootla、Itai Gat、Xiaoqing Ellen Tan、Yossi Adi、Jingyu Liu、Romain Sauvestre、Tal Remez、Jérémy Rapin、Artyom Kozhevnikov、Ivan Evtimov、Joanna Bitton、Manish Bhatt、Cristian Canton Ferrer、Aaron Grattafiori、WenhanXiong、Alexandre Défossez 等另外 7 位作者。2024年。Code Llama：面向代码的开放式基础模型。预印本，arXiv:2308.12950。

Noah Shinn、Federico Cassano、Ashwin Gopinath、Karthik Narasimhan 和 Shunyu Yao。2024年。《Re-flexion：通过语言强化学习实现的语言智能体》。载于《第*37*届国际神经信息处理系统大会论文集（*NeurIPS*）》。Curran Associates Inc.

Shailja Thakur、Baleegh Ahmad、Hammond Pearce、BenjaminTan、Brendan Dolan-Gavitt、RameshKarri，以及 Siddharth Garg。2024年a。《VeriGen：用于 Verilog 代码生成的大型语言模型》。*ACM* 电子系统设计自动化汇刊（*TODAES*），29(3)：1–31。

Shailja Thakur、Jason Blocklove、Hammond Pearce、Benjamin Tan、Siddharth Garg 和 Ramesh Karri。2024b。AutoChip：利用大语言模型反馈自动生成硬件描述语言。预印本，arXiv:2311.04887。

Kiran Thorat、Jiahui Zhao、Yaotian Liu、Hongwu Peng、Xi Xie、Bin Lei、Jeff Zhang 和 Caiwen Ding。2024年。高级大语言模型（LLM）驱动的 Verilog 开发：提升代码综合中的功耗、性能和面积优化。预印本，arXiv:2312.01022。

Prashanth Vijayaraghavan、Apoorva Nitsure、Charles Mackin、LuyaoShi、StefanoAmbrogio、ArvindHa- ran、Viresh Paruthi、Ali Elzein、Dan Coops、David Beymer，以及另外1位作者。2024年。《描述链：改进用于VHDL代码生成和摘要的代码大语言模型》。载于*2024*年*ACM/IEEE*计算机辅助设计机器学习国际研讨会论文集，第1–10页。

Shunyu Yao、Jeffrey Zhao、Dian Yu、Nan Du、Izhak Shafran、Karthik Narasimhan和Yuan Cao。2023年。《ReAct：促进语言模型中的推理与行动协同》。载于国际学习表征会议（*ICLR*）。

Xufeng Yao、Yiwen Wang、Xing Li、Yingzhao Lian、Ran Chen、Lei Chen、Mingxuan Yuan、Hong Xu和 Bei Yu。2024年。《RTLRewriter：大模型辅助寄存器传输级代码优化的方法》。载于*2024 IEEE/ACM*计算机辅助设计国际会议（*ICCAD*）。IEEE。

Zhongzhi Yu、Mingjie Liu、Michael Zimmer、Yingyan Celine、Yong Liu 和 Haoxing Ren。2025年。Spec2rtl- agent：使用大语言模型智能体系统从复杂规范中自动生成硬件代码。载于*2025*年*IEEE*大语言模型辅助设计国际会议（*ICLAD*），第37–43页。IEEE。

Qihao Zhu、Daya Guo、Zhihong Shao、Dejian Yang、Peiyi Wang、Runxin Xu、Y Wu、Yukun Li、Huazuo Gao、Shirong Ma 等。2024年。DeepSeek- Coder-V2：打破闭源模型在代码智能领域的壁垒。

算法：CODMAS

算法1 CODMAS 优化循环

要求： 输入寄存器传输级 C_0，优化目标 G，迭代上限 T

$P \leftarrow$ ArticulatorInit(C_0, G)
$H \leftarrow$ HypoPartnerInit(C_0, G)
$D \leftarrow$数据流图(C_0)
$\Pi \leftarrow$ DKI(P, H, D)
$C \leftarrow$ LLMGenerate(C_0, Π)
($E_{\text{syn}}, E_{\text{func}}, M_{\text{ppa}}$) $\leftarrow$ CEA(C, C_0)
$t \leftarrow 0$
当 ($E_{\text{syn}} \neq \emptyset$ **或** $E_{\text{func}} \neq \emptyset$**或** $M_{\text{ppa}} \not\succeq G$) **并且** $t < T$ **执行**
  如果 $E_{\text{syn}} \neq \emptyset$**那么**
    $E_{\text{func}} \leftarrow \emptyset$, $M_{\text{ppa}} \leftarrow \emptyset$
    $P \leftarrow$ 发音器更新(P, E_{syn})
  else if $E_{\text{func}} \neq \emptyset$**then**
    $M_{\text{ppa}} \leftarrow \emptyset$
    $H \leftarrow$ 假设伙伴更新(H, E_{func})
    $P \leftarrow$ 发音器辅助(P, E_{func})
  else if $M_{\text{ppa}} \not\succeq G$**then**
    $P \leftarrow$ 发音器更新(P, M_{ppa})
    $H \leftarrow$ 假设伙伴更新(H, M_{ppa})
  结束 if
  $\Pi \leftarrow$ DKI(P, H, D)
  $C \leftarrow$ LLMGenerate(C_0, Π)
  (E_{syn}, E_{func}, M_{ppa}) $\leftarrow$ CEA(C, C_0)
  $t \leftarrow t + 1$
结束 while
返回 最终优化后的 RTL C

B 指标计算细节

优化后的 RTL设计从四个关键维度进行评估：面积 (A)、功耗 (P)、时序 (T) 和失效率 (FR)。所有指标均使用 Yosys 和基于 Liberty 的标准单元库计算。

面积（A）： 计算为综合后的单元总面积。相对于基线报告为 A/A_0，其中A_0 为未优化设计。数值 < 1 表示面积减少，而 > 1表示面积增加。

功耗（P）： 使用 Yosys 综合报告估算静态和动态功耗。百分比改进计算为

$$P_{\%} = \frac{P_0 - P}{P_0} \times 100$$

其中 P_0 为原始寄存器传输级设计的功耗。

时序（T）： 以关键路径延迟（CPD）衡量，其定义为经过库单元的最长组合路径延迟。提升百分比为

$$T_{\%} = \frac{CPD_0 - CPD}{CPD_0} \times 100$$

其中 CPD_0 为基线。

失效率（FR）： 优化后未能通过验证或综合检查的 RTL设计所占比例。若设计违反功能正确性、综合失败或超出时序约束，则视为失败。较低的 FR 表示该优化框架具有更高的可靠性和鲁棒性。所有指标均按设计报告，并在整个数据集上提供平均值和标准差，以进行汇总评估。针对流水线化等结构变换（考虑延迟变化）对 CPD 进行调整，以确保公平比较。功耗和面积均按基线 RTL 归一化，以便进行跨设计比较。

C 数据集

为评估自动化RTL优化，我们推出 RTLOpt，这是一个旨在支持可复现、指标驱动评估的数据集。表4 将 RTLOpt 与先前的 Verilog 数据集进行了比较。现有基准测试，如 VerilogEval（Liu等人，2023年）和 TuRTLe（Garcia-Gasulla 等人，2025年），侧重于功能正确性，却缺少可综合性或特定指标评估，因此在评估面向PPA的变换时用途有限。其他数据集，包括 RTLRewriter（Yao 等人，2024年）和 ResBench（Guo 和 Zhao，2025年），提供可综合的RTL，但未包含功能测试或面向指标的目标，因而限制了对优化性能的系统评估。

RTLOPT 包含 120 个 Verilog 三元组（未优化、已优化和测试平台），均可综合且经过功能验证，并明确定义了流水线化和时钟门控等 PPA目标。这使得研究人员能够严格评估正确性和优化效果，并对大语言模型驱动的框架进行定量比较。通过明确纳入特定指标的目标，RTLOPT 填补了关键空白，为以可复现的方式评估端到端寄存器传输级优化流程提供了标准化基准测试。

数据集	Size	功能	可综合性	特定指标
VerilogEval（Liu等人，2023年）	156	✓	✗	✗
RTLLM（Lu等人，2024年）	30	✓	✓	✗
RTLRewriter（Yao 等人，2024年）	95	✗	✓	✗
TuRTLe（Garcia-Gasulla 等人，2025年）	223	✓	✓	✗
CVDP（Pinckney 等人，2025年）	783	✓	✓	✗
ResBench（Guo 和 Zhao，2025年）	56	✓	✓	✗
RTLOPT（我们的方法）	120	✓	✓	✓

表4：RTLOPT 与先前 Verilog 数据集的比较。“特定指标”表示评估中是否存在特定指标优化。

模块	原始寄存器传输级	CODMAS 优化后的寄存器传输级设计
乘法器	无流水线的时序乘法	采用级间寄存器流水线处理的部分积
加法器	32 位行波进位加法器	缩短关键路径的流水线加法器
控制逻辑	未加条件限制的使能信号	带有适当门控的握手感知控制信号
基线中观察到的错误	语法错误、功能不匹配、未达到 PPA 目标	
由 CODMAS 修正	所有语法错误均已修复，功能仿真通过，关键路径缩短了 ∼ 25%	

表 5：CODMAS 下流水线化转换与错误缓解示例。辩证智能体引导结构化编辑，CEA 验证正确性并确认功耗、性能和面积方面的改进。

CODMAS 性能：5 次运行的均值 ±标准差						
模型	流水线化			时钟门控		
	A（↓）	T（↑）	FR（↓）	A (↓)	P (↑)	FR (↓)
GPT-4o	0.960 ± 0.007	25.5 ± 1.2	19.5 ± 2.1	0.999 ± 0.006	21.7 ± 1.5	21.8 ± 2.3
GPT-3.5-turbo	0.972 ± 0.009	21.3 ± 1.1	23.4 ± 2.4	1.020 ± 0.007	18.8 ± 1.2	24.2 ± 2.0
DeepSeek-v2.5	0.979 ± 0.010	21.4 ± 1.0	22.8 ± 2.3	1.015 ± 0.008	19.0 ± 1.3	23.6 ± 2.1
Llama-3	1.032 ± 0.012	19.8 ± 1.3	25.5 ± 2.5	1.048 ± 0.010	16.5 ± 1.1	26.3 ± 2.4
Granite-34b	0.998 ± 0.011	10.5 ± 0.9	28.3 ± 2.8	1.030 ± 0.009	12.9 ± 1.0	29.5 ± 2.6
CodeLlama-34b	1.030 ± 0.012	11.2 ± 1.0	29.5 ± 2.7	1.042 ± 0.010	10.6 ± 0.9	31.3 ± 2.8

表6：提　取的 CODMAS 结果，展示五次运行的均值和标准差。指标：面积（A）、时序（T　　），功耗（P）以及失效率（FR）。FR 表示运行中未能通过语法、功能或功耗、性能和面积检查的比例。

D 基线详情

D.1 基线方法

零样本提示。模型会接收一条描述优化目标（面积、时序、功耗）的指令，并被要求一次性生成已优化的 Verilog 实现。整个过程中不提供迭代推理、反馈或纠正。这一基线反映了仅依靠大语言模型进行优化时的性能下限。

CoDes（描述链）。遵循（Vijayaraghavan 等人，2024年），模型会先生成一系列描述性的中间变换——结构变化、对功耗、性能和面积的预期影响，以及计划进行的优化——然后再输出代码。我们对 CoDes 进行调整，使其明确引用寄存器传输级结构、组合路径和流水线边界。

ReAct。ReAct（Yao 等人，2023年）将推理过程与“行动”交替进行。对于寄存器传输级设计，行动包括生成部分代码、检查语法或查询仿真输出。模型会针对已识别的问题进行推理并尝试修正，但缺少 CODMAS 所采用的更深入的结构规划。

Reflexion。Reflexion（Shinn 等人，2024年）使模型能够存储简短的文字“反思”，描述失败的原因（例如功能不匹配或时序退化）。这些反思构成跨多次尝试的情景记忆。对于寄存器传输级任务，反思内容包括流水线级结构不当、敏感度列表错误或推断出锁存器。

LLM-VeriPPA。LLM-VeriPPA（Thorat 等人，2024年）采用两阶段流程：(1) 修正语法和功能行为；(2) 重新提示模型，在保持等价性的同时提升功耗、性能和面积。我们

复现这套流程，并确保验证检查与 CODMAS 使用的检查相匹配，包括测试平台仿真和基于 Yosys 的功耗、性能和面积提取。

D.2 模型后端

所有基线均采用相同的编译、仿真和基于 Yosys Liberty 的综合流程进行评估。我们测试了专有模型（GPT-4o、GPT-3.5-turbo）（OpenAI，2024年）和开源模型（Llama-3（AI@Meta，2024年）、DeepSeek-v2.5（Zhu 等，2024年）、Granite-34B-Code（Mishra 等，2024年）、CodeLLaMA-34B（Rozière 等，2024年））。温度、采样参数和代码长度限制均遵循标准做法，并予以记录以确保可复现性。每个基线最初都针对通用代码生成；我们通过以下方式对其进行调整，使其适用于寄存器传输级专项优化：（1）通过测试平台仿真确保功能等价性；（2）在适用情况下整合功耗、性能和面积反馈循环；（3）将所有方法限制在相同的最大尝试次数和评估预算内。这些细节确保比较能够隔离算法差异，而非评估基础设施造成的差异。

D.3 实现细节与定性分析

所有实验均在一台配备 64 核 CPU 和 NVIDIA A100 GPU 的服务器上进行。我们使用 RTLOPT 基准测试对六种大语言模型进行评估。每项实验均使用不同的随机种子重复五次，以考虑随机波动。指标包括面积（A）、流水线化的时序（T）、时钟门控的功耗（P）以及失效率（FR）。报告的结果为多次运行的均值 $\pm$标准差。配对双样本 t 检验证实，相较于基线的改进具有统计显著性（$p < 0.01$）。

表 5 展示了一种具有代表性的流水线化转换。基线 RTL 往往存在语法错误、功能不匹配以及未达到 PPA 目标等问题，典型的失效分布为：$\sim 40\%$ 语法违规、 $\sim 35\%$ 功能问题和 $\sim 25\%$ PPA 违规。在此示例中，时序乘法器和行波进位加法器造成了较长的关键路径，而控制逻辑缺少适当的门控。CODMAS 在辩证智能体的指导下进行结构化修改：通过级间寄存器对算术单元进行流水线化、优化关键路径，并确保控制信号具备握手感知能力。代码评估智能体（CEA）会验证语法、功能正确性以及 PPA 改善情况。在五次运行中，优化后的 RTL 实现了关键路径延迟降低 $\sim 25.5\% \pm 0.6$、时钟门控下功耗降低 $\sim 21.7\% \pm 0.5$，且失效率始终低于 30%，展现出良好的鲁棒性和可复现性。此示例凸显了推理—评估循环如何系统地纠正错误并提升性能，尤其适用于规模较小或能力相对较弱的模型。

错误分析表明，基线失效主要源于语法错误（$\sim 40\%$）、功能不匹配（$\sim 35\%$）以及未达到功耗、性能和面积目标（$\sim 25\%$）。CODMAS 通过迭代式、推理引导的改进缓解这三类问题，综合阐述者的计划、假设伙伴的预测，以及 CEA 的确定性评估。