arXiv:2603.02895v1 [cs.AR] 3 Mar 2026

SpecLoop：具有形式化验证反馈循环的智能体 RTL 到规格说明框架

Fu-Chieh Chang[1]，Yu-Hsin Yang[1],Hung-Ming Huang[1],Yun-Chia Hsu[2],Yin-Yu Lin[2],
Ming-Fang Tsai[2],Chun-Chih Yang[2],Pei-Yuan Wu[1]1国立台湾大学通信工程研究所，台北，台湾2
联发科技，新竹，台湾

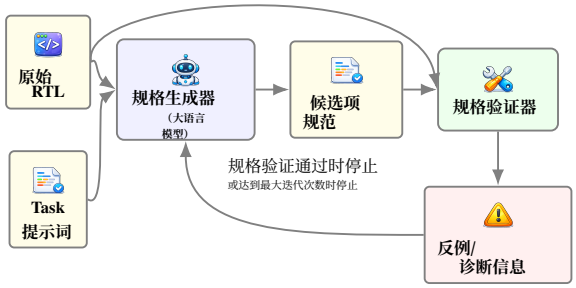
摘要—**寄存器传输级实现往往缺少最新或一致的规格说明，使得理解、维护和验证既成本高昂又容易出错。尽管先前工作已探索使用大语言模型（大语言模型）从寄存器传输级生成规格说明，但确保所生成的文档忠实捕捉设计意图仍是一项重大挑战。我们提出 SpecLoop，这是一个用于寄存器传输级到规格生成、并带有形式化验证驱动的迭代反馈循环的智能体框架。SpecLoop 首先生成候选规格，然后根据这些规格说明重构寄存器传输级；它使用形式等价性检查工具，在重构后的寄存器传输级与原始设计之间验证功能一致性。当检测到不匹配时，反例会被反馈回去，以迭代改进规格说明，直到证明等价性或无法再取得进一步进展。在多种大语言模型和 RTL 基准测试上的实验表明，纳入形式化验证反馈可显著提高规范的正确性与稳健性，优于仅使用大语言模型的基线方法，证明了以验证为导向的规范生成的有效性。**

索引词——**Verilog、大语言模型、形式化验证、设计自动化**

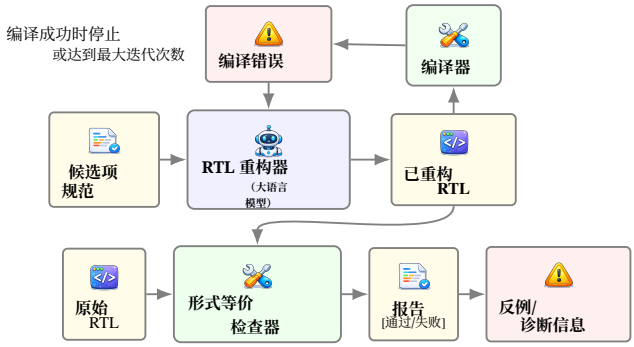
I. 引言

大语言模型最近在硬件设计自动化方面展现出强大能力，涵盖基于规格说明的 RTL 生成以及 RTL 调试与重写。先前工作 [1]–[5] 通过智能体式的工具增强框架提升 RTL 生成质量，提供用于评估和修正所生成设计的语义级信号。在相反方向上，诸如 RTL 理解等 RTL 到文本任务也已得到研究 [6]–[8]，并且早期工作已开始探索 RTL 到规范的生成 [9]–[11]。

然而，从寄存器传输级生成高质量规范仍然颇具挑战：规范需要对设计意图和约束进行高层次抽象，但其质量难以自动衡量和改进。受 Huang 等人的基于重构的评估协议启发， [11],我们提出了一种将大语言模型与形式等价检查工具相结合的智能体框架，通过验证引导的寄存器传输级到规范生成，利用寄存器传输级重构的反馈以及与原始设计进行形式化等价性检查的结果，迭代优化候选规范。图 1 展示了该架构的概览。我们的主要贡献如下：



(a) SpecLoop：使用验证器反馈迭代生成并完善候选规格。



(b) 规格验证器：根据规范重建寄存器传输级，并与原始设计进行等价性检查，返回用于细化的诊断信息。

图 1：SpecLoop 和规格验证器的架构。

- 我们提出了一种由验证引导循环驱动的智能体式寄存器传输级到规范生成框架，并利用形式化等价性检查提供语义层面的反馈。
- 我们提出一种策略，将编译错误与功能不匹配区分开来，然后利用可操作的诊断信息改进规范。
- 我们的实验表明，该框架适用于多种大语言模型和 RTL 基准测试，性能优于先前工作 [11]。该方法实现了最先进的 寄存器传输级到规格性能。

总而言之，我们的智能体验证引导方法将规格生成转变为迭代且可检查的过程，从而能够系统性地提升规格质量，超越单轮提示。

表 I：给定 RTL 代码片段的 RTL 理解与规范生成输出对比示例。

Type	示例
RTL 代码片段	模块 半加器（输入 a，输入 b，输出 和，输出 进位输出）；赋值 {进位输出，和} = a+b；模块结束
RTL 理解（叙述）	这段 Verilog 代码定义了一个名为 HalfAdder 的模块，该模块实现了一个基本的半加器电路。该模块接收两个单比特输入（a 和 b），并产生两个输出：sum（加法结果）和进位输出（进位输出）。assign 该语句使用拼接来计算和位以及输入 a 和 b 二进制相加的进位位。
规范生成（结构化）	**顶层模块：** 半加器。 **输入：** a （1 位），b （1 位）。**输出-输出：** 和 （1 位），进位输出 （1 位）。**功能描述：** 该模块实现了一个半加器电路，用于执行二进制加法，计算两个 1 位操作数。该模块接受两个单位输入，并输出 1 位的和以及 1 位的进位输出。

II. 相关工作

A. 大语言模型中的智能体工具使用与迭代优化

近期研究表明，借助外部工具和反馈，大语言模型的推理效果会有所提升。智能体工具增强方法会调用编译器和分析器 [12]–[14], ，这对于编程等可验证领域尤其有效。另一方面，大语言模型还可以通过迭代优化——反思或自我纠正 [15]–[17]——来增强响应。无论反馈来自模型自身还是工具，这两种范式都以多步错误减少取代单轮生成。这些发现促使我们采用当前方法：工具生成的信号（编译、等价性检查）能够提供客观反馈，而迭代优化则能在多轮过程中提升结果。

B. 面向大语言模型 *RTL* 生成的智能体工具使用

近期研究推动了智能体大语言模型框架的发展，这些框架利用外部工具进行 RTL 生成、调试和重写。编译和仿真工具用于改进 RTL 生成 [3]–[5] 以及语法修复 [1]。等价性检查工具也被集成进来，以便在保持行为的同时重写 RTL，从而改善功耗、性能和面积 [2], ，并支持规格到寄存器传输级工作流中的数据集构建 [18] 和评估 [19] 。然而，这些方法无法直接应用于反向任务——寄存器传输级到规格——因为规格质量难以评估，限制了它们的直接适用性。

C. RTL 理解与规范生成

以往的寄存器传输级到文本研究大致分为寄存器传输级理解和规范生成。寄存器传输级理解会生成叙述性内容，通常是逐行解释；而规范生成则需要更高层次的结构化语义（例如明确的端口名称）。表 I 给出了具体对比。下面总结相关工作。**(i) 寄存器传输级理解**。Pinckney 等人 [6]提出了 *CVDP*。Liu 等人 [7], [8]提出了 *DeepRTL* 和 *DeepRTL2*。两者主要面向代码级理解，而非规格推导；Pinckney 等人 [6]还指出，规范生成比理解更具挑战性。**(ii)规范生成**。He 等人 [9] 使用合成规格来改进寄存器传输级生成，但他们关注的是寄存器传输级的质量。SpecLLM [10] 对规格模态进行了分类，但仍主要停留在定性层面。Huang 等人 [11]全面评估了寄存器传输级到规范生成，但只关注基准测试。不同于先前工作，我们首次明确研究如何通过形式验证指导的迭代优化来提升规格质量。

III. 系统架构

受 Huang 等人 [11],的启发，我们采用迭代重构与检查方法。图 1 概述了我们的智能体验证引导的寄存器传输级到规格框架。图 1a 展示了 SpecLoop 的架构，其中**规格生成器**根据**输入寄存器传输级代码**和**任务提示**生成**候选规格**。随后，将该候选规格交给**规格验证器**检查其正确性。如果验证失败，则将产生的**反例和诊断信息**反馈给规格生成器，以生成修订后的规格。此过程不断迭代，直至证明等价或达到最大迭代预算。图 1b 展示了**规格验证器**的架构。它通过**RTL 重构器**将**候选规格**转换为**重建后的 RTL**，然后进行编译。如果编译失败，则将**编译错误**反馈给重构器，以重新生成可编译设计。随后，使用**形式等价性检查器**将可编译的 RTL 与原始 RTL 进行比较，并返回**报告**（通过/失败）。如果失败，则将**反例和诊断信息**反馈给规格生成器，以细化规格。

A. 规格生成器

Huang 等人 [11] 表明，提示设计会显著影响 RTL 到规格生成。我们对其多步推理提示进行调整和扩展，使模型能够在最终确定规范前进行推理和自我反思，并强制采用结构化输出格式，如图 2 所示。**(1) 左侧模块**展示了我们的首轮提示。它包含三个步骤：(a) 分析寄存器传输级（信号、数据流/控制流、时钟/复位）；(b) 使用固定字段编写结构化规格；(c) 自检其完整性和 RTL 一致性。**(2) 右侧模块** 展示了我们的细化轮提示。它要求模型：(a) 总结验证报告；(b) 找出可能的根本原因（见第 III-B 节和第 III-C 节）；(c) 判断是否需要修改（以及如何修改）受影响的规格字段。这使反馈更具可操作性，并支持渐进式改进。在这些提示的底部，输出规则使我们的重建与检查流程更加稳健。首先，明确的推理步骤能够提高正确性和可解释性。其次， [SPEC_START]和 [SPEC_END] 等标记符使规格易于解析，并能容忍额外文本。第三，禁止在规格中使用 Verilog 代码，可促使模型生成抽象且简洁的规格。

B. RTL 重构器

我们使用基于大语言模型的 RTL 重建器，将每份自然语言规格转换为可编译的 RTL。尽管这一步使语义级验证成为可能，但重建

规格生成器提示（第一轮）

你是一名资深硬件设计工程师，专长于 RTL 设计分析与文档编写。你的任务是解读 **[RTL START]** 与 **[RTL END]** 之间的 Verilog RTL 模块，并编写一份准确、易读的**功能规格说明**。

阶段 1 — 内部分析（推理）： 在内部逐步分析寄存器传输级。你可以从以下方面进行推理：(1) 信号定义和位宽 (2) 输入、输出与内部逻辑之间的数据依赖关系 (3) 时序行为与组合行为 (4) 控制流、计数器、有限状态机转换和使能信号 (5) 时钟控制、复位条件和时序依赖关系 (6) 特殊或异常信号行为 使用这些推理步骤构建正确且完整的规范。

阶段 2 — 结构化规格输出： 推理完成后，撰写最终规范时请使用以下格式：

[SPEC START]

摘要：

- 简要概述该设计的主要目标和整体功能。
- 如果包含多个模块，请先确定顶层模块，并在本节中明确写出其名称。

模块名称： <寄存器传输级中的确切名称>

输入：

- <信号_名称>（<位宽>）： <简要用途>**输出：**
- <信号_名称>（<位宽>）： <简要用途>

详细功能描述：

- 描述模块的用途，以及输出如何由输入得出。
- 连贯地说明内部逻辑、时序行为和控制机制。
- 重点说明任何计数器、状态机、算术逻辑或握手信号。

时钟与复位行为：

- 指定有效时钟边沿、复位极性和初始化序列。

备注（如适用）：

- 提及参数、默认值、流水线延迟或异常情况。

[规范结束]

If multiple modules 存在，请使用上述结构分别描述每个模块。

阶段 3 — 自我反思与一致性检查： 在最终确定你的答案之前：（1）在内部验证所提及的所有输入/输出信号均存在于寄存器传输级中。（2）确保在功能描述中解释每个输出。（3）验证时钟和复位行为均已正确识别。（4）如果仍存在不确定性，请作出合理推断，并在备注部分**明确说明**。

输出规则： （1）在提供最终规范之前，必须清楚地呈现逐步推理过程。（2）使用 **[SPEC START]** 括起你的规范，并以 **[SPEC END]** 结束。（3）最终规范中**不得**输出 Verilog 代码。（4）主要-保持清晰、正式且符合规格说明的技术语气。

[寄存器传输级 START] ...(omitted)... **[寄存器传输级 END]**

规格生成器提示（结合验证反馈的细化轮次）

当规格验证失败时，请仔细遵循以下步骤。

回顾验证的工作原理：

1) 大语言模型仅根据你的规格说明重建寄存器传输级代码。2) 形式等价性检查器将重建的寄存器传输级代码与原始 RTL 进行比较。

失败意味着重建的寄存器传输级代码与原始代码**不** 等价——这可能是由规格说明含糊或错误导致的，也可能是重建大语言模型误解了规格说明。

步骤 1 — 总结验证报告： 简要总结验证报告，范围为 **[报告开始]**至 **[报告结束]**，内容包括：(1) 哪些输出信号或属性已证明等效，哪些未证明。(2) 报告中指出的具体反例、不匹配信号或逻辑差异。(3) 发现的其他问题（等效性检查之外），例如 RTL 语法错误或多个边沿敏感事件。

步骤 2 — 根因分析： 对于每项等价性失败，确定最可能的原因：(1) **规范不准确** ——规范本身有误或不完整（例如位宽错误、缺少信号、逻辑描述不正确、时钟/复位极性错误），因此任何忠实的重建结果都会与原始 RTL 不一致。(2) **规范歧义** ——规范在技术上与原始 RTL 一致，但措辞含糊，导致重建大语言模型给出了有效但不同的解读。(3)**重建局限** ——规范清晰且正确，但重建大语言模型未能准确地将其转换为 RTL。请针对每项失败给出结论，并简要说明理由。

步骤 3 — 决策 (1) 如果任何失败归因于（**a**）**规范不准确**，则仅修订规范，以纠正所有已发现的错误。(2) 如果任何失败归因于（**b**）**规范歧义**，则仅修订规范，使相关描述更加精确、明确，同时不改变原意。(3) 如果**所有** 失败都归因于（**c**）**重建局限**，则原样输出上一轮的规范，并停止。

输出规则： (1) 在提供最终规范之前，必须清楚地展示逐步推理过程。(2) 请将最终规范括在 **[SPEC_START]**和 **[SPEC_END]**之间。(3) 请**不要**在最终规范中输出 Verilog 代码。(4) 保持清晰、正式且符合规范要求的技术语气。

[报告开始]

规格验证 [通过/失败]。

重构 RTL:

...（省略）...

等价性检查工具的日志:

...（省略， 例如信号不匹配）...

[报告结束]

图 2：规范生成器的多步提示模板。（**左**）第一轮：分析寄存器传输级，编写结构化规格，并进行自检。（**右**） 细化：利用验证器诊断信息（例如编译器错误或反例）仅编辑受影响的字段，其余保持不变。由于版面限制，部分换行已省略。

可能会失败：模型可能误解未充分说明的需求、遗漏边界情况，或生成语法无效的寄存器传输级。因此，等价性检查失败不能仅归因于规格生成器。为减轻这一混杂因素，我们将重建有效性与功能等价性区分开来。在形式化等价性检查（FEC）之前，我们使用轻量级编译器编译重建的寄存器传输级；若编译失败，则根据编译错误重新运行重建。若编译成功但等价性仍然失败，则认为规范不完整或不正确，并反馈反例以进行细化。即便如此，重构器的能力仍可能限制端到端性能，包括规范tion 正确（参见第 III-C 节和第 VI 节）。

C. 形式等价性检查器

我们使用 FEC 工具比较原始 RTL 与重构后的 RTL。与仿真相比，等价性检查提供了一种穷尽式的语义层面判据：若检查通过，则表明规范足够详细，能够使重构后的 RTL 与原始 RTL 在功能上等价。若检查失败，原因可能有多种：**E.1 原始 RTL 无效**。 如果原始 RTL 编译失败，或 FEC 工具无法解析它，检查器可能无法给出有意义的判定。在这种情况下，我们会停止迭代。

表 II：验证器错误处理策略。

发送至重构器的错误		发送至规格生成器
E.1	停止迭代	不可验证；无反馈循环
E.2	重构 with 编译错误	达到重构预算后，发送编译错误
E.3	重构	达到重构预算后，发送反例
E.4	重构	达到重构预算后，发送错误消息

E.2 无法编译的重构 RTL。 如果重构后的 RTL 无法编译，我们会将其视为重构失败，并将编译器错误反馈给重构器后重新运行。由于此类错误也可能源于规范不准确（例如，缺少位宽信息），在达到固定的重试次数后，我们会将持续出现的失败归因于规范质量问题，并将编译错误反馈给规范生成器。

E.3 功能不匹配。 真正的不匹配会产生一个反例，其中输出信号存在差异。由于不匹配也可能源于重构器缺陷，我们可以在固定的预算内重复进行重建；如果不匹配仍然存在，就将反例反馈给规范生成器，以修订规范。

E.4 不确定性故障。 FEC 可能因超时或其他工具问题而返回不确定的结果。我们首先重新运行重构器；如果结果仍不确定，则将其归因于定义不当的规范，并向规范生成器发送错误消息，请求进行细化。

设计选择——信息隐藏： 为了公平评估规范中包含的信息，我们有意不 将 FEC 日志反馈给重构器，因为这些日志可能会泄露原始 RTL（检查器的输入）的细节，从而实际上绕过规范。唯一的例外是编译失败：我们会向重构器提供编译器错误，因为这些错误不会直接泄露原始 RTL 的语义，而且生成可编译设计也需要这些信息。我们在表 II 中总结了对重构器和规范生成器采取的操作。

IV. 实验设置

我们开展实验，从三个方面评估系统：（1）验证器引导的细化是否比单轮基线更能提升 RTL 到规范的生成效果；（2）诊断反馈（编译器错误和反例）是否比二元通过/失败信号更有效；（3）该流水线在不同大语言模型骨干和基准测试上的鲁棒性如何。

A. 大语言模型候选项

我们的流水线中有两种大语言模型角色：规范生成器 和 *RTL* 重构器。原则上，使用更强大的重构器可以减少重构引起的错误并提高端到端成功率；然而，这也会因引入额外的模型容量而干扰比较。为隔离验证引导循环的作用，我们在这两种角色中都使用完全相同的 大语言模型。因此，相较于单轮方案的任何提升都反映了验证器的集成效果。此外，我们的方法与模型无关，无需微调；我们直接接入现有的大语言模型。因此，我们评估了多种现有大语言模型，包括 Llama4-Scout、Llama4-Maverick、Qwen3-Coder-30B、Qwen3-Coder-480B、DeepSeek-V3.1 和 GLM-4.6，并观察到其在不同模型规模和架构下均取得了稳定的提升。

B. 基准测试与指标

我们在两个具有代表性的 RTL 基准测试上进行评估：VerilogEval [20]和 RTLLM [21]。VerilogEval 包含玩具级模块，而 RTLLM 则包括规模更大的设计。由于这些基准测试最初是为规范到 RTL 的评估而设计的，我们遵循 Huang 等人 [11]提出的 RTL 到规约协议：对于每个输入 RTL，我们生成自然语言规格，并使用 RTL 重构评分（RR 分数） [11]对其进行评估。RR 会检查根据规格重建的 RTL 是否通过测试平台，重建工作由 GPT-5 Codex [22]完成。我们选择 RR 而非 GPT-Score [7]等语义级替代指标，因为与语义级指标相比，它对规格缺陷更敏感 [11] 。尽管 RR 也包含 RTL 重建步骤，但这并不会天然偏袒我们的方法：（1）RR 使用前沿重构器（例如 GPT-5 系列）来确保重建保真度，而 SpecLoop 使用同一个（通常较弱的）模型同时进行生成和重建，以避免额外能力造成混淆；（2）RR 使用测试平台而非形式等价来评估重建后的 RTL，衡量其在给定环境下的行为，而不是我们的等价性检查目标。

C. 基线方法

我们考虑两种基线方法。**(1) 单轮：** 这遵循先前工作 [11]：大语言模型根据输入的寄存器传输级规范，一次性生成规范，不进行验证器引导的细化。我们通过一项细微的提示词修改（启用中间推理输出）复现其结果，使输出更易于理解。**(2) 仅通过/失败验证器：** 我们运行相同的重建和等价性检查流程，但验证器仅返回二元决策（通过/失败），不提供诊断消息（例如编译器错误或反例）。如果验证失败，我们会提示大语言模型重试，不提供额外信号，重试预算与完整反馈方法相同。

D. 其他实现细节

对于等价性检查，我们使用带有 SAT 后端和深度界限 10 的 Yosys EQY [23] 。我们构建了一个轻量级日志解析器，将 EQY 的结果映射为错误类型（E.1– E.4），并构建了一个 VCD 解析器，用于提取 E.3 的反例。当出现错误 E.1 时，我们会终止反馈循环，因为验证器反馈无法改善这种情况。我们最多允许规格生成器重试两次，因此包括初始轮次在内，总共生成三次规格。每个规格最多允许 RTL 重构器重试两次。我们通过 OpenRouter [24]调用大语言模型 API，并将温度设为 0.4，以平衡随机解码与正确性。为降低统计

表III：VerilogEval 和 RTLLM 上采用 RR-Score 的实验结果（均值 ± 标准差）；色调越深表示排名越高。验证器引导变体（仅通过/失败和完整诊断）的表现优于单轮基线；总体而言，完整诊断排名最高，仅通过/失败排名第二。

模型\方法	单轮 ([11])	仅通过/失败 (我们的方法)	完整诊断 (本文)
数据集：VerilogEval			
Qwen3-Coder-30B	0.722 ± 0.016	0.759 ± 0.017	**0.795 ± 0.021**
Qwen3-Coder-480B	0.885 ± 0.016	0.878 ± 0.009	**0.912 ± 0.012**
DeepSeek-V3.1	0.865 ± 0.019	0.908 ± 0.011	**0.940 ± 0.003**
Llama4-Scout-16E	0.686 ± 0.009	**0.722 ± 0.016**	0.705 ± 0.019
Llama4-Maverick-128E	0.818 ± 0.006	0.812 ± 0.017	**0.842 ± 0.015**
GLM-4.6	0.906 ± 0.021	0.921 ± 0.006	**0.934 ± 0.020**
数据集：RTLLM			
Qwen3-Coder-30B	0.733 ± 0.009	**0.747 ± 0.019**	**0.747 ± 0.019**
Qwen3-Coder-480B	0.767 ± 0.025	**0.787 ± 0.019**	0.767 ± 0.041
DeepSeek-V3.1	0.767 ± 0.025	0.793 ± 0.025	**0.800 ± 0.016**
Llama4-Scout-16E	0.667 ± 0.025	**0.727 ± 0.034**	0.687 ± 0.025
Llama4-Maverick-128E	0.793 ± 0.034	0.767 ± 0.009	**0.813 ± 0.009**
GLM-4.6	0.813 ± 0.034	0.847 ± 0.009	**0.880 ± 0.000**

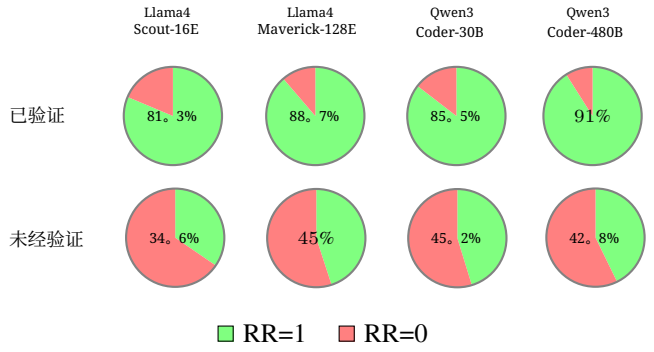


图 3：不同模型的已验证规格与未经验证规格之比，以及在基准测试和验证器变体上取平均的 RR 分数。在各模型中，已验证规格的 RR=1比例均高于未经验证规格，对更强的模型尤为明显。

方差，我们将每项实验运行三次，并报告均值和标准差。

V. 实验结果

A. 总体定量结果

表 III 报告了三种设置下的 RTL 重构评分（均值 ±标准差）：单轮、仅通过/失败验证器和 完整诊断。我们重点指出以下观察结果。(i) **迭代循环始终优于单轮生成**。综合各项结果，加入验证循环（仅通过/失败或完整诊断）均优于单轮生成，这表明验证器反馈——无论是通过/失败结果，还是完整诊断信息（在 12 种模型/数据集设置中分别有 4 种和 9 种设置排名第一）——都是有价值的推理时信号。(ii) **即使是仅通过/失败验证器，也能优于单轮生成**。尽管缺少详细诊断信息，仅通过/失败设置的排名第一次数仍多于单轮设置（在 12 种模型/数据集设置中为 4 次对 0 次）。这反映了先前工作中的观察结果 [16], [17]：在没有外部指导的情况下，反复自我纠正并反思推理步骤或结果仍然有益。

（iii）**更丰富的诊断信息最有帮助**。完整诊断取得了最强的性能，这表明详细的反馈（例如反例）能够帮助模型进行更精确的细化。总体而言，这些结果表明，使 RTL 到规格生成过程具备迭代性和可检查性能够提高端到端正确性，并在性能上达到最先进水平，优于先前的单轮基线。

B. 已验证与未验证规格

我们进一步分析规格验证器在反馈循环中的影响。图 3 报告了每个模型在两组中达到 $RR{=}1$ 和 $RR{=}0$ 的样本比例：在预算内通过规格验证器 FEC 的已验证规格说明，以及预算耗尽后仍未经验证的规格说明。每个饼图汇总了已验证与未经验证的划分，该结果在基准测试和反馈变体（Pass/Fail-Only 与 FullDiagnosis）上取平均。由此可得两点观察。(i) 已验证规格更有可能达到 $RR{=}1$。(ii) 更强的 RTL 重构器在已验证规格中产生更高的 $RR{=}1$ 比例，如 Qwen3-Coder-480B 和 Llama4-Maverick 相较于 Qwen3-Coder-30B 和 Llama4-Scout 所示。这些观察表明，已验证规格的质量更高（以 RR 分数衡量），且更强模型的收益更大。

C. 定性研究

图 4 给出了一个定性示例。(i) 原始 RTL 设计是一个从 0 递增到 999 并带有同步复位的计数器。(ii) 然而，在大语言模型的首次响应中，模型错误地推断复位是异步的。(iii) 我们的规格验证过程随后根据这一错误规范重构 RTL；由于重构后的设计包含异步复位，等价性检查器标记了不匹配。(iv) 我们将所得诊断反馈给大语言模型，在第二轮中，它正确识别出错误源于对复位行为的错误刻画，并将规范修订为使用同步复位。(v) 在下一轮验证中，重构器生成了带有同步复位的计数器，且该设计通过了等价性检查。该示例说明规格验证循环如何检测生成规格说明中的错误，并引导大语言模型加以修正。

VI. 局限性与未来工作

我们的工作存在若干局限，可分为两类。**基于重建范式和度量指标的根本局限：**端到端结果取决于 RTL 重构器和等价性检查配置，因此，重建失败或资源受限检查（例如固定深度）可能会阻碍针对更复杂设计提供有用反馈。此外，使用 RR 分数进行评估也取决于前沿模型的重建能力。**可改进的工程方面不足：**反馈解析和提示仍依赖启发式方法，而要扩展到大型工业级 RTL 代码库，则需要多智能体系统 [3]–[5]（例如，规划、工具

第一轮提示

你是一名专精于 RTL 设计分析的资深硬件设计工程师
与文档编写。你的任务是解读介于以下内容之间的 Verilog RTL 模块
[寄存器传输级开始] 与 **[寄存器传输级结束]**，并生成一份精确、人类可读的**功能性规范**. . ..（省略）. . .

[寄存器传输级开始]

模块 TopModule（输入时钟,
输入复位,　　寄存器输出 [9:0] q）;
　始终@(时钟上升沿)
　　如果（复位 || q == 999） q <= 0;
　　否则 q <= q+1;
模块结束
[寄存器传输级结束]

大语言模型在第一轮的响应

阶段 1 — 内部分析（推理）. . .（省略）.. .**控制逻辑：**(i) 简单计数器，终值为 999。(ii) 复位条件在行为上是同步的，但使用了包含输入复位的 OR 条件。由于复位未在敏感列表中与时钟同步，因此实际上表现为异步复位，因为 OR 逻辑在时钟块内求值。**时钟与复位：**(i) 时钟：上升沿触发。(ii) 复位：由于 always 块内部的异步，if (reset — ...)—，看起来是异步的。. . .（省略）. .

[规格说明开始]

摘要：
Thi 该模块实现一个 10 位递增计数器，从 0 计数到 999 后回绕。它支持异步复位，复位信号有效时会立即清零计数器。

模块名称： TopModule

输入：
- 时钟（1 位）：时钟信号，由上升沿触发。• 复位（1 位）：异步复位信号；信号为高电平时，立即复位计数器。

输出：
- q（10 位）：计数器的输出；除非复位或达到上限，否则每个时钟周期递增。

详细功能描述：
该模块实现了一个同步递增计数器。在时钟的每个上升沿：

- 如果复位信号有效（高电平），或 q 的当前值等于 999，则输出 q 复位为 0。• 否则，q 加 1。

这样可确保计数器从 0 计数到 999，然后回绕至 0。

时钟与复位行为：
- 时钟：时钟上升沿。• 复位：异步--- 当复位为高电平时，无论时钟状态如何，q 都会立即置为 0。• 复位后的初始状态：q = 0。

备注：
- 计数器使用 10 位寄存器，可表示 0 到 1023 的数值，但在功能上限制为 0–999。
- 复位的优先级高于递增操作。

[规范_结束]

第一轮规格验证器报告

规格验证失败。

重构 RTL:

模块 TopModule（输入　　线网 时钟,　输入　　线网 复位,
寄存器输出　　[9:0] q）;
　总是 @(时钟上升沿或复位上升沿) begin
　　如果 (复位) begin
　　q <= 10'd0;
　　end 否则如果 (q == 10'd999) begin
　　q <= 10'd0;
　　结束 否则 开始
　　q <=q + 1'b1;
　　end
　end
模块结束

验证工具日志:

不匹配信号:
t=11 sig=q
寄存器传输级_重构: 0000000000
寄存器传输级_原始: 0000001010
...(omitted)...

大语言模型第二轮的响应

. .. （已省略）.. . **阶段 2 — 根因分析** . ..（已省略）. .. **验证报告摘要：** (i) 验证工具发现信号 q 在时间 t=11 处不匹配。(ii) 在该时刻：原始 RTL：q = 10（二进制：0000001010）。重构寄存器传输级：q = 0（二进制：0000000000）。**根因分析：**问题在于复位行为的解释。. ..（已省略）. ..

[规格开始]
...(omitted)...
时钟与复位行为：
- 时钟：时钟的上升沿。• 复位：同步 --- 当复位信号为高电平时，复位将在时钟的下一个上升沿生效。• 复位后的初始状态：q = 0。

. . .（省略） . . .
[规格结束]

第二轮规格验证器报告

规格验证通过。

重构 RTL:

模块 TopModule (输入　　线网 时钟, 输入　　线网 复位,
寄存器输出　　[9:0] q);
　始终@(时钟上升沿) 开始
　　如果（复位 ||(q == 10'd999)）开始
　　q <= 10'd0;
　　结束，否则开始
　　q <=q + 1'b1;
　　end
　end
模块结束

图4：定性分析。为清晰起见，选定的文本片段以红色突出显示。此示例展示了 SpecLoop 如何修正规格错误：第一轮规范错误地描述了异步复位，随后重建未能通过等价性检查，诊断结果则指导下一轮修改规范，将其改为同步复位，此后验证器通过了检查。

选择以及处理 IP 依赖关系）。未来的工作包括加强重建和验证流水线，并开发多智能体系统来处理更复杂的设计。

第七节：结论

我们提出了一种智能体验证引导框架，用于 RTL 到规范的生成，并将该问题转化为迭代式重建与检查循环。通过重建从候选规格说明生成寄存器传输级代码，并利用形式等价性检查产生语义级反馈，我们的方法系统地改进规格说明，超越单轮提示。实验表明，验证器引导能够提升规格质量，这一点从前沿模型上的重建成功率可见一斑；更丰富的诊断信息也大有裨益。我们相信，这一方向能够使 RTL 到规格生成成为实际硬件文档工作流程中的实用环节。

参考文献

[1] Y. Tsai、M. Liu 和 H. Ren，“Rtlfixer：使用大语言模型自动修复 RTL 语法错误”，载于第 *61* 届 *ACM/IEEE* 设计自动化会议论文集，第 1–6 页，2024 年。[2] X. Yao 、 Y. Wang 、 X. Li 、 Y. Lian 、 R. Chen、L. Chen、M. Yuan、H. Xu 和 B. Yu，“Rtlrewriter：大模型辅助 RTL 代码优化的方法”，载于*2024* 年 *IEEE/ACM* 国际计算机辅助设计会议（*ICCAD*），2024 年。[3] C.-T. Ho、H. Ren 和 B. Khailany，“Verilogcoder：结合基于图的规划与基于抽象语法树（AST）的波形跟踪工具的自主 Verilog 编码智能体”，载于*AAAI* 人工智能会议论文集，第 39 卷，第 300–307 页，2025 年。[4] Z. Yu、M. Liu、M. Zimmer、Y. Celine、Y. Liu 和 H. Ren，“Spec2rtl-agent：利用大语言模型智能体系统从复杂规格说明自动生成硬件代码”，载于*2025* 年 *IEEE* 国际大语言模型辅助设计会议（*ICLAD*），第 37–43 页，IEEE，2025 年。[5] Y. Zhao、H. Zhang、H. Huang、Z. Yu 和 J. Zhao，“Mage：用于自动化 RTL 代码生成的多智能体引擎”，载于第 *62* 届 *ACM/IEEE* 设计自动化会议论文集，DAC ’25，IEEE Press，2025 年。[6] N. Pinckney、C. Deng、C.-T. Ho、Y.-D. Tsai、M. Liu、W. Zhou、B. Khailany 和 H. Ren，“综合 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*，2025 年。[7] Y. Liu、C. XU、Y. Zhou、Z. Li 和 Q. Xu，“DeepRTL：通过统一表示模型衔接 Verilog 理解与生成”，载于第十三届国际学习表征会议，2025 年。[8] Y. Liu、H. Zhang、Y. Zhou、Z. Shi、C. Xu 和 Q. Xu，“DeepRTL2：适用于 RTL 相关任务的通用模型”，载于计算语言学协会论文集：*ACL 2025* 专辑，第 6485–6500 页，2025 年。[9] S. He、C. Chen、X. Zheng、Z. Liu 和 C. Wang，“改进基于大语言模型的 RTL 敏捷设计规格”，2026 年。[10] M. Li、W. Fang、Q. Zhang 和 Z. Xie，“SpecLLM：探索利用大语言模型生成和审查超大规模集成电路设计规格说明”，载于*2025* 年国际电子设计自动化研讨会（*ISEDA*），第 749–755 页，IEEE，2025 年。[11] H.-M. Huang、Y.-H. Yang、F.-C. Chang、Y.-C. Hsu、Y.-Y. Lin、M.-F. Tsai、C.-C. Yang 和 P.-Y. Wu，“评估大语言模型生成 RTL 设计规格说明的能力”，*arXiv* 预印本 *arXiv:2512 00045*，2025 年。.[12] Q. Wu、W. Liu、J. Luan 和 B. Wang，“ToolPlanner：用于多粒度指令、路径规划与反馈的工具增强型大语言模型”，载于*2024* 年自然语言处理实证方法会议论文集，第 18315–18339 页，计算语言学协会，2024 年 11 月。[13] K. Zhang、J. Li、G. Li、X. Shi 和 Z. Jin，“Codeagent：利用工具集成智能体系统增强代码生成，以应对真实世界的代码仓库级编程挑战”，载于第 *62* 届计算语言学协会年会论文集（第一卷：长篇论文），第 13643–13658 页，2024 年。[14] Q. Wu、G. Bansal、J. Zhang、Y. Wu、B. Li、E. Zhu、L. Jiang、X. Zhang、S. Zhang、J. Liu、等，“Autogen：通过多智能体对话实现新一代大语言模型应用”，载于首届语言建模会议，2024 年。[15] D. J. Shah、P. Rushton、S. Singla、M. Parmar、K. Smith、Y. Vanjani、A. Vaswani、A. Chaluvaraju、A. Hojel、A. Ma、等，“重新思考预训练中的反思”，*arXiv* 预印本 *arXiv:2504.04022*，2025 年。[16] Y.-T. Lee、F.-C. Chang、H.-Y. Shih 和 P.-Y. Wu，“大语言模型的内在自我纠正：通过机制可解释性实现可解释提示”，载于第四届可部署人工智能研讨会，2026 年。[17] F.-C. Chang、Y.-T. Lee 和 P.-Y. Wu，“揭示大语言模型中反思的潜在方向”，载于*NeurIPS 2025* 机制可解释性研讨会，2025 年。[18] P. Yubeaton、A. Nakkab、W. Xiao、L. Collini、R. Karri、C. Hegde 和 S. Garg，“Verithoughts：通过推理与形式化验证实现自动化 Verilog 代码生成”，载于神经信息处理系统会议第三十九届年会：数据集与基准赛道，2025 年。[19] P. Jin、D. Huang、C. Li、S. Cheng、Y. Zhao、X. Zheng、J. Zhu、S. Xing、B. Dou、R. Zhang、Z. Du、Q. Guo 和 X. Hu，“Realbench：基于真实世界 IP 设计对 Verilog 生成模型进行基准测试，2025。[20] M. Liu、N. Pinckney、B. Khailany 和 H. Ren，“VerilogEval：评估用于 Verilog 代码生成的大语言模型”，载于*2023* 年 *IEEE/ACM* 国际计算机辅助设计会议（*ICCAD*），第 1–8 页，IEEE，2023。[21] Y. Lu、S. Liu、Q. Zhang 和 Z. Xie，“RTLLM：用于大语言模型设计寄存器传输级生成的开源基准”，载于*2024* 年第 *29* 届亚洲及南太平洋设计自动化会议（*ASP-DAC*），第 722–727 页，IEEE，2024。[22] OpenAI，“GPT-5 Codex”。OpenAI 模型文档。访问日期：2026-02-24。[23] “使用 Yosys 进行等价性检查（EQY）文档。” https://yosyshq.readthedocs.io/projects/eqy/en/latest/。访问日期：2026-02-21。[24] “OpenRouter。” https://openrouter.ai/。访问日期：2026-02-21。