Qeios

Peer Review

Review of: "DRC-Coder: Automated DRC Checker Code Generation Using LLM Autonomous Agent"

Sooyong Lee[1]

1. Samsung, Seoul, South Korea

Review of "DRC-Coder: Automated DRC Checker Code Generation Using LLM"

This paper introduces DRC-Coder, a novel framework that leverages Large Language Models (LLMs) and Vision-Language Models (VLMs) to automate the generation of Design Rule Checking (DRC) code. By adopting a multi-agent structure with distinct Planner and Programmer roles, the authors propose a scalable and efficient solution to address the complexities of DRC workflows. This framework demonstrates strong performance and aligns well with recent AI trends, but this review highlights its strengths and suggests improvements to further enhance its practical utility and impact.

Strengths

1. Innovative Multi-Agent System: The separation of tasks into Planner and Programmer agents reflects a well-thought-out design that intelligently mimics human workflows. This division enhances specialization, reduces error rates, and improves overall efficiency.

2. High Performance with Multi-Modal Inputs: The ability to process textual descriptions, visual data, and pixel grid-based layout representations is a significant strength. This integrated approach enables the framework to interpret and process complex design rules effectively, achieving an F1 Score of 1.0 across all test cases.

3. Efficiency in DRC Code Generation: The capability to generate DRC code in an average of 4 minutes per rule is impressive, significantly reducing the time compared to traditional manual workflows,

 doi.org/10.32388/Z3CVE7

which can take days.

4. Alignment with Current AI Trends: The use of autonomous AI agents aligns with recent advancements in AI research, making this framework highly relevant and contemporary.

Suggestions for Improvement

1. Expanding Standard Cell Evaluations

The paper claims that the 207 standard cells used represent sub-3nm nodes. However, in practice, larger chip designs often involve significantly more standard cells. Expanding the dataset size and diversity to better reflect the real-world complexities of DRC workflows is crucial. Providing a report on the trade-offs between computational cost and accuracy when increasing or decreasing the number of standard cells would help potential users better gauge its applicability in practical settings. Additionally, analyzing how the F1 Score or computational costs vary with increasing standard cell count and design rule complexity could provide deeper insights into the framework's scalability and robustness.

2. Improving Layout Representation

While the current research employs pixel grid-based representations, industry practices often rely on vector-based layouts. Providing support for vector-based inputs, or at least offering both grid and vector options, would align the framework more closely with industry standards. If the pixel grid size and layout pitch are not commensurate, there is concern that the VLM might not accurately interpret distances or dimensions at a resolution matching design rules, such as the database unit (DBU). Instead, vector-based processing could achieve DBU precision (typically 0.1nm), which is critical for advanced semiconductor nodes.

3. Enhancing Explainability

Incorporating mechanisms to explain decisions made by Planner and Programmer agents would improve transparency and user trust: Providing detailed commentary on the Planner's rule interpretation process. Explaining intermediate results and the Programmer's code generation decisions to allow users to understand and refine outputs. These explainability features are

particularly useful for addressing edge cases and novel scenarios, helping bridge the gap between automation and human understanding.

4. Strengthening Human-AI Collaboration

As mentioned in the limitations section, Human-in-the-loop (HITL) processes can significantly enhance the reliability and adaptability of DRC-Coder: Allowing engineers to interactively refine Planner-generated rules or Programmer-generated code to address edge cases and domain-specific nuances. Integrating feedback loops where human corrections directly inform agent learning enables the system to adapt more effectively over time.

As an additional direction, employing differentiable human feedback methods such as Direct Preference Optimization (DPO) would further refine the system. For example, offering designers multiple options (e.g., A vs. B) and incorporating their preferences into the optimization process could help align outputs more closely with human expectations.

Conclusion

DRC-Coder represents a significant advancement in automating DRC workflows, demonstrating exceptional efficiency and accuracy. Its multi-agent structure, alignment with current AI trends, and adaptability to future extensions make it highly valuable for publication. By addressing the suggested improvements—such as expanding standard cell evaluations and refining layout vector representations—the framework could become even more applicable to real-world scenarios. Additionally, emphasizing explainability and fostering human-AI collaboration would allow DRC-Coder to overcome the limitations of fully automated systems, improving trust and reliability in industrial workflows. This research showcases the potential of AI-driven solutions for complex engineering tasks, providing a robust foundation for future studies and applications. I look forward to seeing the evolution of this promising framework.

Declarations

Potential competing interests: No potential competing interests to declare.