RECHISEL: Effective Automatic Chisel Code Generation by LLM with Reflection

Juxin Niu [1], Xiangfeng Liu [1,2], Dan Niu [3], Xi Wang [3,4*], Zhe Jiang [3,4], Nan Guan [1*]

[1] City University of Hong Kong, Hong Kong SAR [2] Northeastern University, China
[3] Southeast University, China [4] National Center of Technology Innovation for EDA, China

1. Background & Motivation

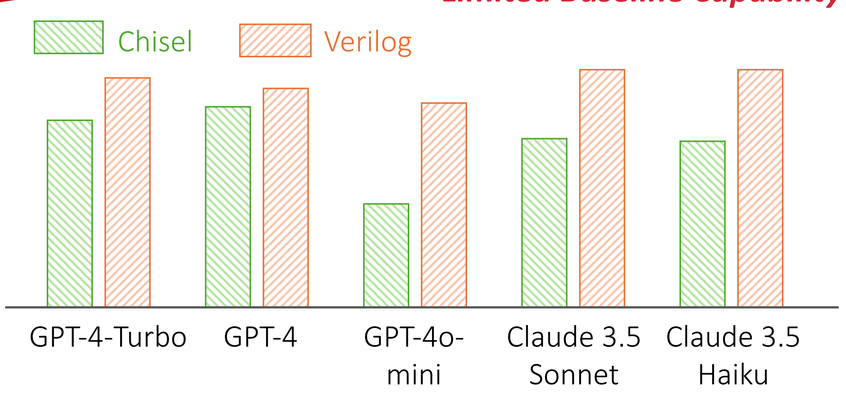
High-Level Abstraction
Chisel uses advanced language structures to simplify complex hardware modeling.

Concise & Maintainable
Chisel offers rich sets of APIs to minimize boilerplate for clarity and readability.

Scalable & Reusable
Chisel supports object-oriented features to efficiently scale for larger projects.

Chisel's Promise
Future of Agile Hardware Design

LLM Hurdle
Limited Baseline Capability

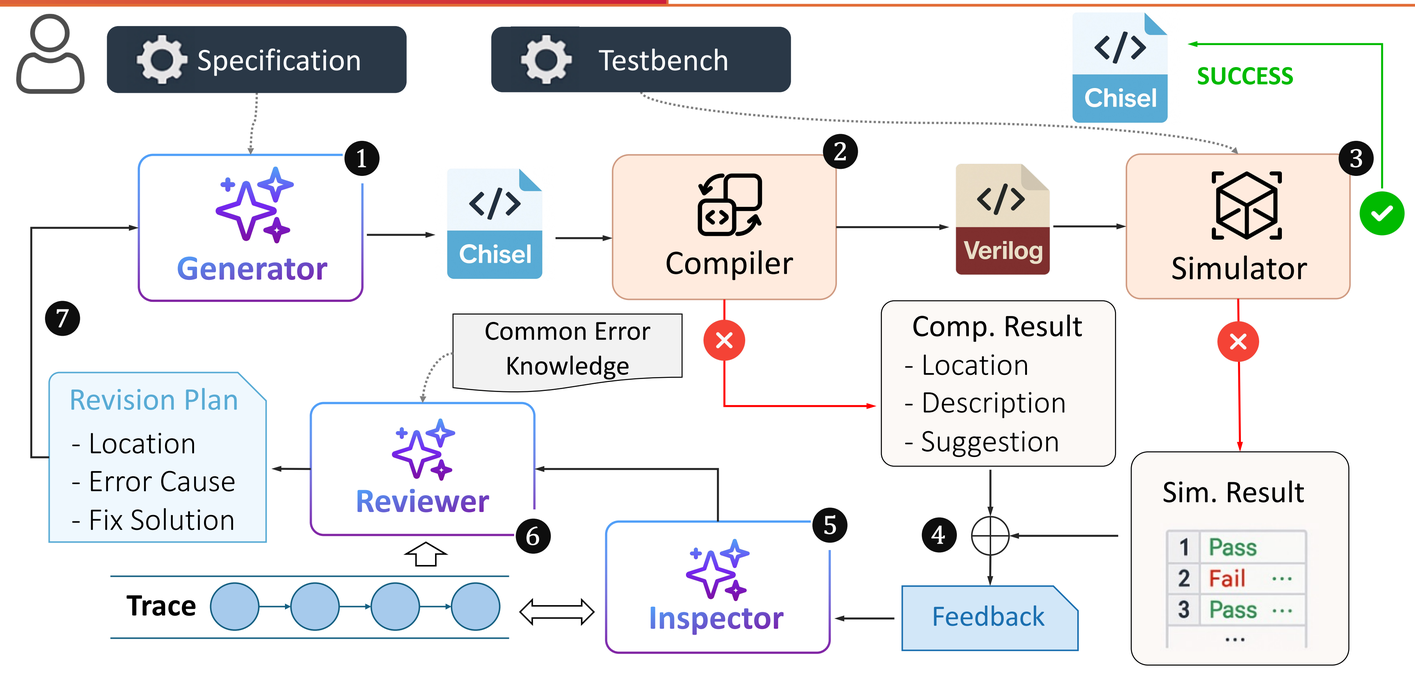


Why? — Scarce Training Data

High-Level Thinking

Use test-time strategies to compensate for the deficiencies in its baseline capabilities.

2. RECHISEL Design & Workflow

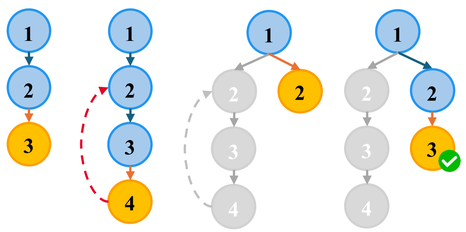


RECHISEL is provided with a natural language specification and a testbench, aiming to produce functionally correct Chisel code that adheres to the specification.

- ✓ ❶ ***Generator*** creates initial Chisel implementations based on the specification.
- ✓ ❷❸ The generated Chisel code is compiled into Verilog. The Verilog output is then verified for its functional correctness against the testbench.
- ✓ ❹ When a failure occurs, the system captures error feedback:
  - Syntax Errors: Compiler output, including the error's location, description, and suggested fixes.
  - Functional Errors: Input stimuli, expected outputs, and actual outputs.
- ✓ ❺❻ ***Inspector*** receives error feedback and analyzes it in the context of *Trace*, which contains the history of all prior attempts. Based on its diagnosis, ***Reviewer*** formulates a *Revision Plan* with a specific *Fix Solution*.
- ✓ ❼ The ***Generator*** receives the *Revision Plan* and generates a new, improved version of the Chisel code, which then re-enters the workflow at ❷.

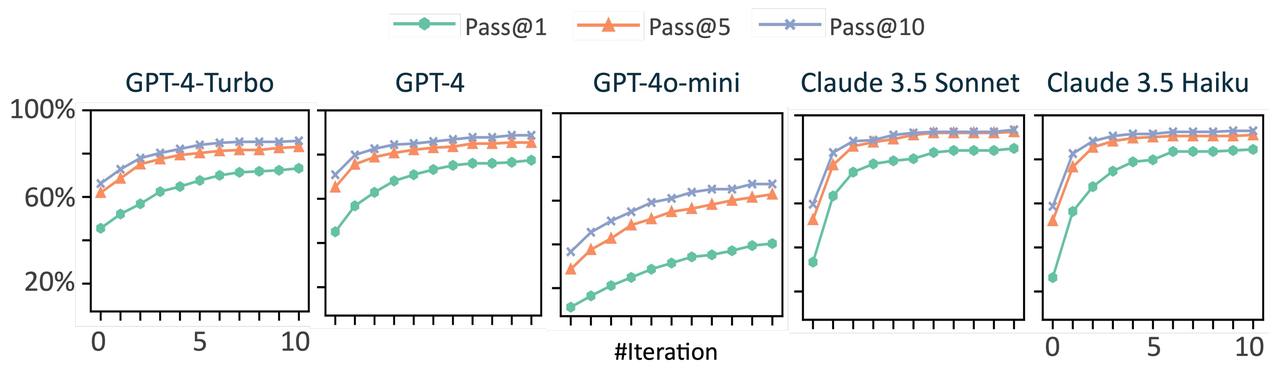
RECHISEL includes special designs for further performance improvements.

- Escape Mechanism
  It prevents the LLM from getting trapped in "*non-progress loops*" where it repeatedly fails to fix the same error. By detecting this repetitive cycle, it discards the unproductive attempts and forces the model to generate a novel solution from a pre-loop state.



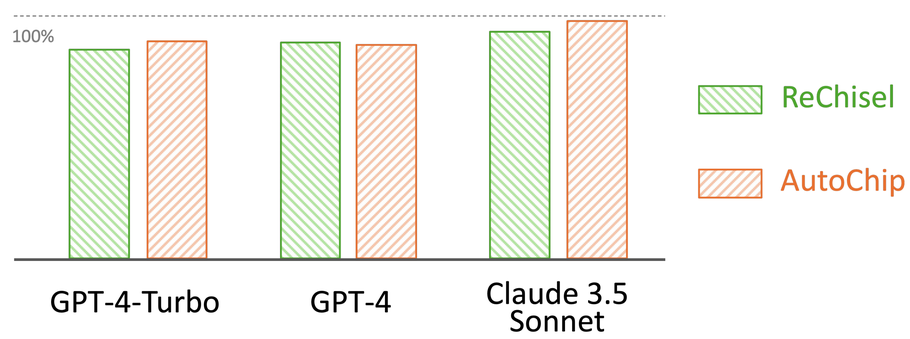
- In-Context Knowledge
  It involves embedding a curated database of common Chisel errors and their solutions directly into the model's prompt. This gives the model immediate "expert" guidance, enabling it to resolve frequent issues much faster and more accurately without reasoning from scratch.

3. Evaluation



The performance improvement of different LLMs on Chisel code generation as they are allowed more iterations for refinement.

The success rates generally increase with each iteration before reaching a plateau.



Comparison of the final code generation success rates between RECHISEL (Chisel) and AutoChip (Verilog).

RECHISEL's performance in generating Chisel code is highly competitive and comparable to AutoChip's performance for Verilog generation.

https://github.com/niujuxin/ReChisel
Niu Juxin juxin.niu@my.cityu.edu.hk

This work is supported by the National Key Research and Development Program (Grant No. 2024YFB4405600), the National Natural Science Foundation of China (Grant No. 62472086), the Basic Research Program of Jiangsu (Grants No. BK20243042, BG2024010), and the Start-up Research Fund of Southeast University (Grant No. RF1028624005).