LLM Challenges Fixing Verilog Testbenches

Mark Zakharov, Farzaneh Rabiei Kashanaki, Alex Lee, Milind Varma Penumathsa, Jose Renau
Computer Science & Engineering Department
University of California, Santa Cruz
Santa Cruz, USA
{mzakharo, frabieik, alee156, mpenumat, renau}@ucsc.edu

***Abstract*—Traditional chip design heavily invests in Verilog verification, with more effort often spent on verification than coding. LLMs have shown promise in translating error feedback into source code corrections and can potentially correct Verilog programs using testbench feedback. This study explores the challenges of automating Verilog fixes and presents several potential solutions.**

I. Introduction

This study explores integrating Large Language Models (LLMs) into the chip design verification process to automate Verilog code repair with testbench feedback. Despite challenges like limited Verilog resources and a weaker compiler ecosystem, leveraging LLMs has shown potential. The research focuses on comparing bug-free Verilog (Gold) with buggy versions (DUT) using Logical Equivalence Checks (LEC) for discrepancy identification. Preliminary findings suggest that structured feedback is crucial and may be more impactful than extensive fine-tuning, offering new strategies for employing LLMs in specialized fields.

II. Setup

The study aimed to improve LLMs' Verilog fixing ability by providing structured feedback on semantic errors using an automated flow with Yosys for formal verification of combinational logic. An Agent was developed to interact LLMs with Yosys, iterating over compiler errors and using LEC against a reference model to guide corrections. The LLM was fine-tuned on a dataset of conversations focused on correcting existing Verilog programs, with various configurations tested to evaluate overfitting and underfitting effects. Higher-performing configurations passed up to 6 of 17 tests, showing that feedback significantly improved success rates for certain problems.

III. Evaluation

The evaluation revealed two main issues: result regression and unnecessary modifications. To address these, a regression trapping filter and an "update" parameter were introduced to monitor and correct failures, and the feedback format was refined to exclude correct outputs. Eight trials indicated that feedback and the "update" feature slightly impeded performance due to bias from previous code versions. Future improvements could involve more varied corrections and longer conversations to avoid the model becoming "stuck" on similar, incorrect implementations. These changes aim to enhance the LLM's ability to consistently correct Verilog programs effectively. Overall, refining the feedback mechanism and dataset can significantly improve the LLM's performance in correcting complex logic errors.

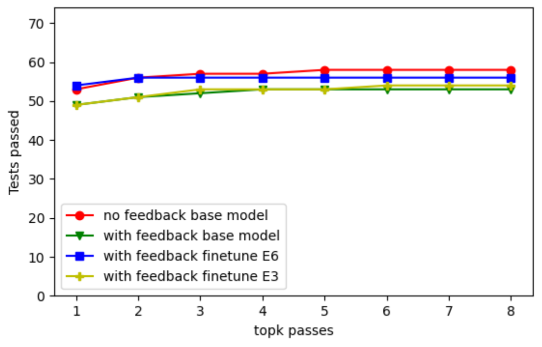

Fig. 1. Final Results with Update

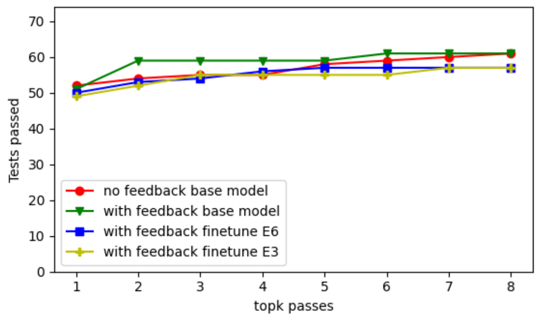

Fig. 2. Final Results without Update

IV. Conclusions

This work explored improving GPT-3.5-turbo's ability to correct Verilog programs using testbench feedback. An agent flow and fine-tuning dataset were developed, which improved code generation but are not ideal. The methods presented may be useful for future work, potentially leading to a hardware design paradigm where designers focus solely on verification.

References

[1] HDLBits - Verilog Practice. website, November 2017.
[2] The 2020 Wilson Research Group Functional Verification Study, 2020.