# FIXME: Towards End-to-End Benchmarking of LLM-Aided Design Verification

**Gwok-Waa Wan[1,2], SamZaak Wong[1,2], Shengchu Su[1], Chenxu Niu[3], Ning Wang[4], Xinlai Wan[1], Qixiang Chen[2], Mengnv Xing[2], Jingyi Zhang[1], Jianmin Ye[1], Yubo Wang[2], Rongchang Song[2], Tao Ni[2], Qiang Xu[2,5], Nan Guan[2,4], Zhe Jiang[1,2*], Xi Wang[1,2*], Yong Chen[3], Jun Yang[1]**

[1]Southeast University, Nanjing, Jiangsu, China
[2]National Center of Technology Innovation for EDA, Nanjing, Jiangsu, China
[3]Texas Tech University, Lubbock, TX, USA
[4]City University of Hong Kong, Kowloon, Hong Kong SAR, China
[5]The Chinese University of Hong Kong, Shatin, Hong Kong SAR, China

## Abstract

We introduce **FIXME**, the first end-to-end and large-scale benchmark for evaluating Large Language Models (LLMs) in hardware design functional verification (FV). Comprising 747 tasks derived from real-world hardware designs, FIXME spans five core FV sub-sets: specification comprehension, reference model generation, testbench generation, assertion design, and RTL debugging. To ensure high data quality, we developed an AI-human collaborative framework for agile data curation and annotation. This process resulted in 25,000 lines of verified RTL, 35,000 lines of enhanced testbenches, and over 1,200 SystemVerilog Assertions. Furthermore, through expert-guided optimization within the multi-agent aided flow, we achieved a remarkable 45.57% improvement in average functional coverage, underscoring the benchmark's robustness. Through evaluation of state-of-the-art LLMs like GPT-4.1, FIXME identifies key limitations and provides actionable insights, advancing the potential of LLM-driven automation in hardware design functional verification.

## Introduction

The integration of Large Language Models (LLMs) (Naveed et al. 2023) into hardware design workflows, known as LLM-Aided Design (LAD) (IEEE 2025), is emerging as a transformative paradigm in Very Large Scale Integration Circuit (VLSI) (Barbe 2013) design methodologies (Thakur et al. 2023; Fu et al. 2023). This integration promises to revolutionize hardware design by enabling more agile design automation and intelligent iteration (Fang et al. 2025).

Traditionally, hardware design encompasses register-transfer level (RTL) implementation using hardware description languages (HDLs) and functional verification (FV) (Mano and Ciletti 2012). Existing research on LAD focuses on HDL generation, such as RTLCoder (Liu et al. 2024b), BetterV (Pei et al. 2024), ChatCPU (Wang et al. 2024), VGV (Wong et al. 2024), GPT4AIChip (Fu et al. 2023), and ChatChisel (Liu et al. 2024c).

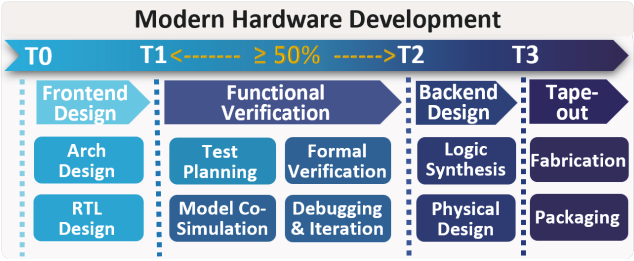


Figure 1: Typical IC development workflow. Verification typically accounts for over 50% of the time consumption (Wilson Research Group and Siemens EDA 2022).

While these studies demonstrate the ability of LLMs to generate functionally simple designs, their capabilities are limited by a critical constraint: the lack of systematic and sufficient data alignment to develop a deep understanding of hardware functionality (Qayyum et al. 2024). This limitation is largely due to the relative scarcity of research focused on applying LLMs to the FV stage, which in turn impedes their ability to design and verify complex circuits.

In this context, a fine-grained evaluation of the capabilities of LLMs in FV is crucial for guiding future research. Unfortunately, existing LAD benchmarks are inadequate for this task: 1. Existing works, such as RTLLM (Lu et al. 2024) and VerilogEval (Liu et al. 2023), mainly evaluate HDL generation and do not involve verification. The hardware designs provided in them are mostly sampled from beginner websites like HDLBits (hdlbits.01xz.net 2024), which cannot reflect the complexity of FV challenges in the real world, so these designs remain insufficient even after modifications. 2. Recently emerging FVEval (Kang et al. 2024) and AssertionBench (Pulavarthi et al. 2025) focus solely on specific scenarios in FV. This scope fails to capture the full breadth and complexity of real-world FV, which includes critical tasks such as specification comprehension (Li et al. 2024), testbench (Yang, Wille, and Drechsler 2014) generation, reference model (Swan 2006), assertion (Lu et al. 2025) design, and debugging (Wang et al. 2025). The time allocated to these tasks in FV is significant, as shown in Figure 1.

*Corresponding author: xi.wang@seu.edu.cn, zhejiang.arch@gmail.com

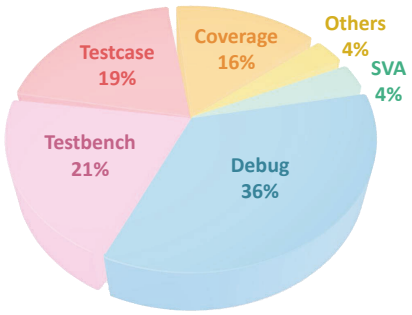


Figure 2: Time Cost Distribution of Traditional Hardware Design Functional Verification (Synopsys 2015)

To address these critical limitations, we propose FIXME, the first comprehensive evaluation framework designed for LLM-aided functional verification. FIXME includes 747 tasks sampled from real hardware designs across the network, categorized into three difficulty levels (L1-L3), divided into 5 subsets: SpeCom, MGen, TBGen, SVAGen, and RTLFix. It matches the FV process end-to-end, finely examines the capabilities of LLMs in design understanding, reference model design, testbench planning, SVA generation, and debugging. The main contributions are as follows:

- **We propose the first end-to-end benchmark for LLM-aided design functional verification.** To the best of our knowledge, **FIXME** is the first comprehensive, multi-modal, and quantitative evaluation framework specifically designed for LLMs in FV. It encompasses the full FV pipeline, including specification, reference model, testbench, assertion, and debugging, thereby enabling a reliable and realistic assessment of LLM capabilities in practical verification scenarios.
- **We construct a high-quality, functional-verification-specific benchmark dataset with expert-level accuracy.** FIXME features a structured task taxonomy with three difficulty levels, five sub-sets, and 747 diverse tasks—from multiple-choice questions to code generation. Created through an AI-human collaborative workflow powered by our multi-agent system, the benchmark integrates expert-curated specifications, 25K lines of audited design code, 35K lines of enhanced testbench, and over 1,200 SystemVerilog assertions, achieving an average functional coverage improvement of 45.57% over the initial data, thereby enhancing evaluation accuracy.
- **We conduct systematic evaluation and provide actionable insights for LLM-aided design research.** Through comprehensive benchmarking of state-of-the-art LLMs, we identify current limitations and promising directions for future improvement in LAD-based verification. Our findings offer a foundational reference for advancing LLM capabilities in complex, structured LAD FV tasks.

## Related Work

**Functional Verification (FV)** FV is a critical phase in the modern hardware design lifecycle, aimed at ensuring that a design conforms to its specification and behaves as intended under all possible input conditions (Bergeron and Autodesk 2003; Spear and Tumbush 2012; Wang, Chang, and Cheng 2009). A typical FV process is as follows: given a specification, engineers construct reference models, testbenches, and assertions to verify the RTL. If any failures or output mismatches occur, the design undergoes debugging and refinement until correctness is achieved. The time allocation for each step is illustrated in Figure 2, and the entire workflow is depicted in Figure 3. This process is both time-consuming and complex, motivating the exploration of automated solutions (Wu et al. 2024; Abdollahi et al. 2025).

**LLM-Aided Design and Benchmarks** Recent advancements in LLMs have enabled their application in hardware design. Various initiatives focus on leveraging LLMs for different aspects of the design process, including HDL code generation (e.g., VeriGen (Thakur et al. 2024), BetterV (Pei et al. 2024), RTLCoder (Liu et al. 2024b)), RTL debugging (e.g., RTLFixer (Tsai, Liu, and Ren 2024), MEIC (Xu et al. 2024)), and overall chip acceleration (e.g., ChipChat (Blocklove et al. 2023), ChipGPT (Chang et al. 2023), GPT4AIGChip (Fu et al. 2023), ChatCPU (Wang et al. 2024)). Several studies also explore the use of LLMs in FV(e.g., LLM4DV (Zhang et al. 2023), VerilogReader (Ma et al. 2024), AssertLLM (Fang et al. 2024), AutoBench (Qiu et al. 2024), and Assertionforge (Bai et al. 2025)).

Benchmarking has become a key tool for assessing the capabilities of LLMs in hardware design. Most existing benchmarks focus on HDL code generation (e.g., RTLLM (Lu et al. 2024), VerilogEval (Liu et al. 2023), VeriGen (Thakur et al. 2024), and GenBen (Wan et al. 2025)). However, benchmarks specifically targeting the functional verification process remain limited. Although some efforts address formal verification (e.g., FVEval (Kang et al. 2024)), there is a significant gap in comprehensive evaluation on practical, end-to-end simulation-based verification tasks.

**Challenges in LLM-Aided FV Benchmark** Constructing effective benchmarks for LLM-aided Functional Verification (FV) presents several key challenges: **1) Data Scarcity:** The availability of high-quality FV data for LLM training and evaluation remains significantly limited (Liu et al. 2024a). Our comprehensive sampling of open-source Verilog repositories revealed that fewer than 10% provided a complete set of specification documents, RTL code, and corresponding testbenches, underscoring a critical shortage of adequate FV data. **2) Quality Assurance:** Even in cases where complete datasets are available, validating their accuracy requires extensive simulation-based verification, which is inherently time-consuming and resource-intensive. This quality assurance step is essential to ensure data reliability (Chang et al. 2024b), but significantly complicates benchmark preparation. **3) Task Complexity:** FV encompasses multiple intricate stages, including specification interpretation, model design, stimulus iteration, formal verification, and debugging. Each stage presents distinct requirements for benchmark construction, requiring careful curation to achieve an optimal balance between complexity and diversity. This multi-stage process further increases the challenge of developing

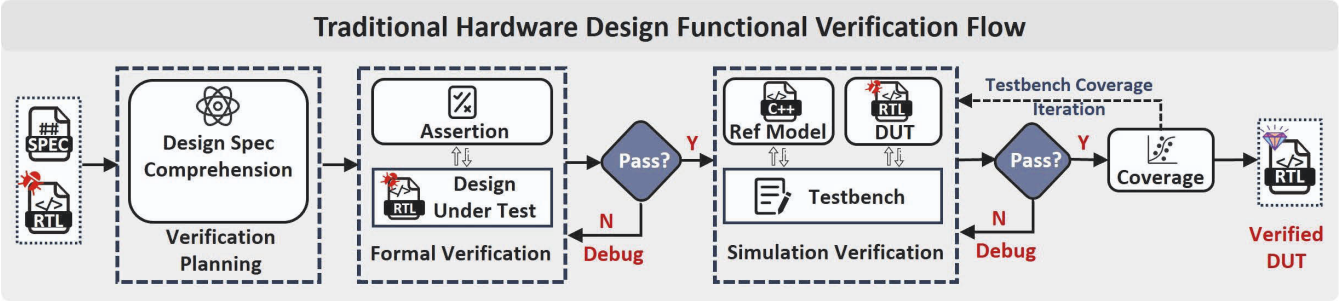


Figure 3: Traditional Design Functional Verification (Synopsys 2015)

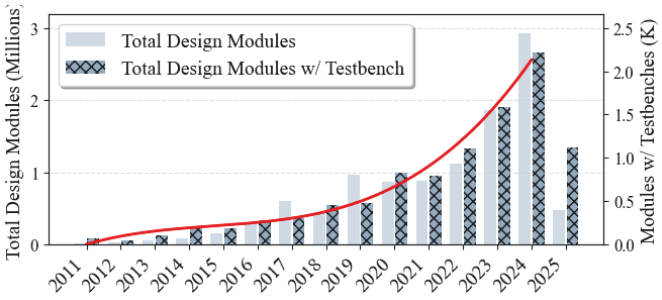


Figure 4: Total Hardware Design Statistics on Public Data from 2011.01-2025.06. Out of over one million design modules, only over 7000 have testbenches.

representative benchmarks.

## FIXME: The Proposed Benchmark

### Overview

To address the critical need for a comprehensive evaluation of LLMs in hardware FV, this paper proposes FIXME as a large benchmark with 747 tasks, which were derived from real-world hardware designs, representing a substantial collection of approximately 10% of all publicly available design modules that include an executable testbench (see Figure 4). Designed to facilitate a holistic, end-to-end assessment that mirrors a typical FV workflow, FIXME is methodically organized into five distinct sub-sets. To enable a more granular analysis of model capabilities, tasks within each sub-set are further classified into three levels of difficulty. The sub-sets are described as below:

- **SpeCom (Specification Comprehension Evaluation)**: contains 149 tasks for evaluating the specifications comprehension of LLM, including 51 multi-modal tasks.
- **MGen (Reference Model Generation Evaluation)**: comprises 111 tasks, with 31 multi-modal tasks.
- **TBGen (TestBench Generation Evaluation)**: With 158 tasks, this sub-set targets testbench generation, featuring 51 multi-modal tasks.
- **SVAGen (SystemVerilog Assertion Generation Evaluation)**: This sub-set includes 61 tasks for SVA design, of which 21 are multi-modal.
- **RTLFix (RTL Fix Evaluation)**: This is the largest subset with 268 tasks, including 166 multi-modal tasks.

Specifically, the proportion of tasks is partially based on the time cost distribution of corresponding tasks in real-world hardware functional verification (FV), as Figure 2, aiming to strike a balance between the importance of each component and feasibility for independent evaluation.

Moreover, considering that practical FV processes often involve visual content such as circuit diagrams, FIXME introduces multi-modal task types for better simulating real-world scenarios. As demonstrated above, the construction of FIXME is complex and is designed to provide a more realistic and challenging platform for evaluating the potential of LLMs in the context of hardware FV.

### Design Philosophy

**End-to-End Evaluation Paradigm** FV is a complex, multi-stage process in which domain-specific knowledge and underlying capabilities are tightly coupled. Failures at any stage can significantly delay design convergence. Therefore, an effective evaluation of FV capabilities should be conducted in an end-to-end manner. This principle motivates the design of FIXME: a comprehensive assessment framework that balances evaluation fidelity, task objectivity, and diversity across a wide range of design types and verification requirements. To achieve this, we identify five core subsets and design corresponding task classifications that follow a one-to-one mapping with real-world verification steps, as summarized in Section 2.1.

**Modular Sub-Set Organization** To ensure extensibility and flexibility in accommodating diverse verification needs, FIXME organizes its tasks into five mutually decoupled subsets. These sub-sets are designed to be independent of each other and the overall evaluation workflow. This modular architecture enables future benchmark iterations to focus on specific sub-sets or introduce new tasks without compromising the integrity of the entire framework.

**AI-Augmented Benchmark Construction** To address the challenge of data scarcity and labor-intensive checks in real-world FV scenarios, we propose an AI-human collaborative framework tailored for benchmark construction. We design a multi-agent system that automates data filtering,

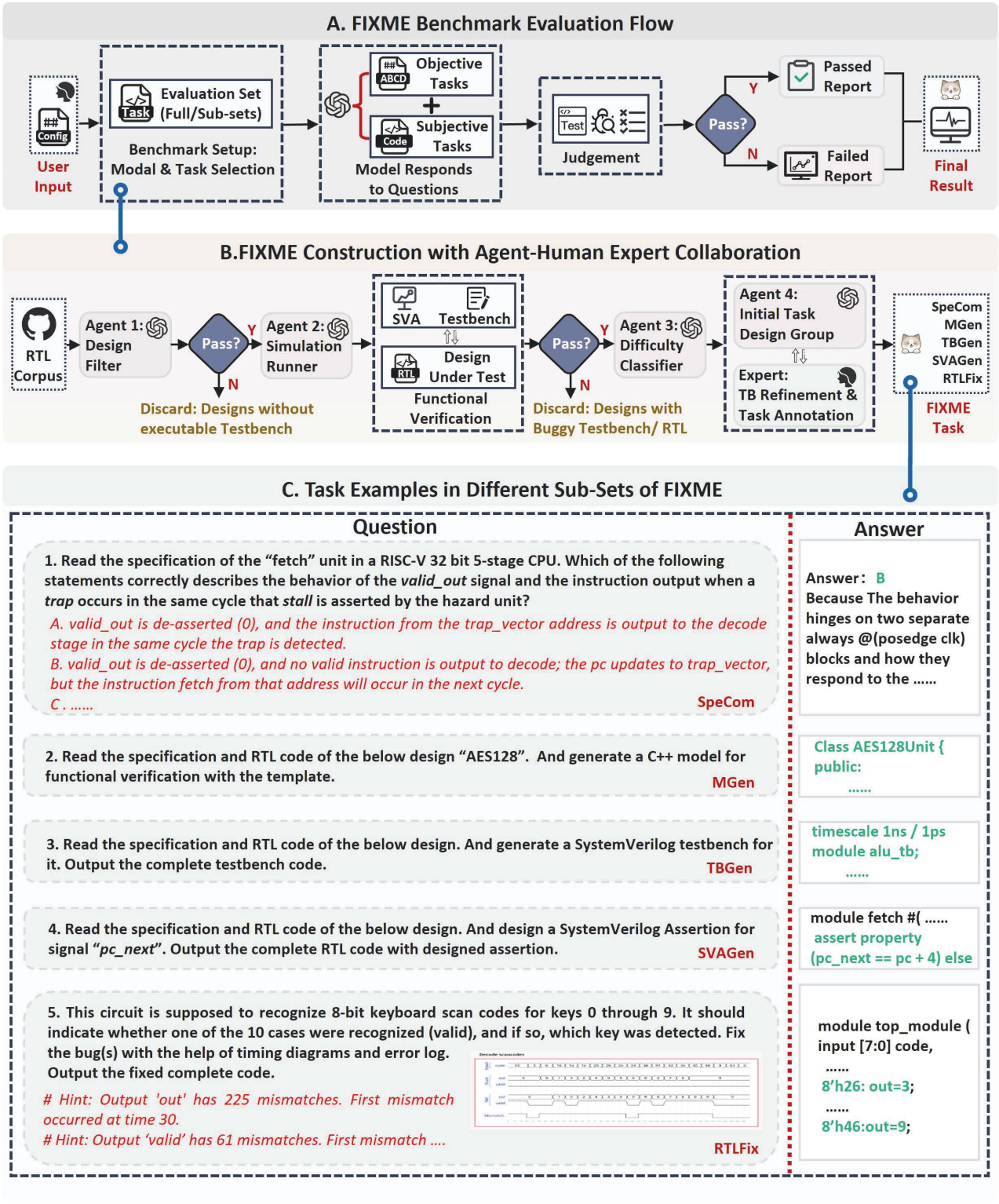


Figure 5: FIXME: Workflow, Construction, and Task Example. Part A illustrates the general workflow with FIXME, where a configuration file specifying the LLM API, modal, and evaluation set, etc. Part B details AI-human collaborative construction of FIXME, as exemplified by its efficient identification of 7,473 hardware designs with testbenches from over ten million collected entries. Finally, part C provides examples for each sub-set: C.1 provides a case of level L2; C.2 provides one case of level L3; C.3-C.5 provide cases of level L1; and C.5 demonstrates the situation of multimodal cases.

simulation tools execution, initial auto-labeling, and preliminary question generation. As illustrated in Figure 5, we implement this methodology through this framework. Moreover, the methodology can be generalized to other domains facing similar challenges in data availability and annotation.

## Construction of FIXME

**Data Collection** We initially crawled hardware design repositories from across the web as our primary data source. The detailed data statistics are presented in Figure 4. There are over 7000 design modules with a testbench in total. To systematically process this data, we introduce an automated pipeline comprising specialized agents designed for various stages of data curation and benchmark construction. The initial phase, automated basic data filtration, begins with **Agent 1**, which scans project file structures to retain only those containing essential testbench files and verification scripts. Next, **Agent 2** employs few-shot learning to analyze the filtered project directories, extracting RTL code and testbench files, and generating verification scripts compatible with Synopsys VCS (Synopsys, Inc. 2023). Afterwards, these scripts are executed through the VCS task API, and the correct ones are selected as primary candidates for the FIXME benchmark.

**Question Annotation** The process begins with a two-phase difficulty classification. Initially, an automated agent (**Agent 3**) performs a preliminary classification of all candidate modules into three tiers based on Lines of Code (LoC): small-scale (0–100 LoC), medium-scale (100–200 LoC), and large-scale ($\geq$ 200 LoC). Subsequently, human experts refine this quantitative sorting by incorporating qualitative metrics of functional complexity, such as control flow depth, state machine intricacy, and interface design, thereby ensuring a robust, hybrid evaluation of task difficulty.

After classification, tasks are generated and refined using a collaborative human-agent workflow. A key preliminary step involves **Agent 4** performing a semantics-preserving rewrite of all design specifications, which is then validated by experts. This is crucial for creating novel task prompts that mitigate the risk of LLMs solving problems via memorization of public data. The generation process is then tailored for each subset:

- **SpeCom**: Multiple-choice and true-false questions are generated by **Agent 4** from the specifications.
- **MGen, TBGen, and SVAGen**: For these tasks, the problems and reference solutions are derived from the original project's specifications and expert-verified code to maintain fidelity to the source design.
- **RTLFix**: **Agent 4** programmatically injects common, realistic bugs into the verified RTL. Human experts then review and refine these fault scenarios to ensure they possess genuine diagnostic value and reflect real-world hardware design challenges.

**Answer Annotation** To enhance the reliability and precision of the FIXME benchmark, human experts meticulously augmented each candidate task with optimized testbenches and SVAs. This process entailed: (1) refining testbenches to achieve over 90% functional coverage, (2) crafting SVAs for formal verification, and (3) developing reference model interfaces. This effort served two purposes: to eliminate modules with corner-case failures, strengthen dataset robustness, and to produce high-fidelity reference solutions, reducing evaluation ambiguity.

Quantitatively, these enhancements optimized 58,000 tokens of technical specifications, audited 50,000 lines of design code (selecting 25,000 lines), added 25,000 lines of verification testbenches, integrated 1,235 critical assertions, and established a unified debugging interface. Resulting coverage improvements include a 45.57% increase in functional coverage, a 7.5% increase in average line coverage, and a 43.14% boost in toggle.

The above efforts provide standard answers or references for MGen, TBGen, and SVAGen. Besides, for SpeCom, correct answers are established by human experts during question generation by **Agent 4**. As for RTLFix, reference solutions are also defined within the fault injection process by **Agent 4**, with expert oversight ensuring accuracy. This method provides a definitive standard for model assessment.

**Multi-Modal Features** FIXME naturally supports multimodal, including text, code, circuit diagrams, tables, and waveform files, to simulate real-world FV scenarios (Chang et al. 2024a). For the waveform files required during debugging, we use the VCD format, which can be converted into text-based time-series signal sequences and waveform screenshots for the model to reference.

**Evaluation metrics** Specification comprehension is evaluated based on the pass rate of objective questions. For code generation tasks — including reference model generation, testbench generation, and assertion generation — evaluation metrics include both syntax correctness and functional pass rate. Additionally, for testbench generation, we further report scores from simulation tools, along with line coverage statistics to assess the completeness of test stimuli. Debugging tasks are evaluated based on the pass rate after bug fixes. These metrics collectively provide a comprehensive assessment of both functional correctness and verification coverage, enabling fine-grained analysis of LLMs' capabilities across different stages of the FV pipeline. Finally, we report the fraction of tasks in which the answer produced by the model matches the reference answer under "Pass@5".

# Evaluation

We selected 10 models, including SOTA closed-source, open-source, and domain-specific fine-tuned VLSI models. To accommodate the extended context requirements of more complex tasks, we chose versions of these models that support longer input/output token lengths, namely GPT-4.1-2025-04-14 (MacCallum and Lee 2025), Claude-Sonnet-4-202505 (Anthropic 2025), Gemini-2.5-Flash (Comanici et al. 2025), DeepSeek-R1-20250324 (Guo et al. 2025), Llama-4-Scout-B10 (Meta AI 2025), Qwen-QwQ-32B (Qwen Team 2025), Mistral-Large-2-128k (Mistral AI 2025), GPT-4o-128k (OpenAI 2024), Gemini-1.5-Pro-128k (Google 2024), and Semikong-70B (pentagoniac


Figure 6: Evaluation Results of Pass Rate.

2024). The hyperparameter settings for all models remain consistent.

To collect comprehensive simulation test coverage data for the testbench generation task, we utilized Synopsys VCS for data collection, focusing on line coverage and toggle coverage metrics. FIXME also supports the collection of additional metrics, such as branch coverage. We performed a pass@5 evaluation for all tested models.

## Results and Discussion

### Results

The pass rate results for all evaluated models are depicted in Figure 6, while detailed coverage statistics specific to the TBGen subset are illustrated in Figure 8.

For all subsets except SpeCom, none of the models surpassed a 50% functional PR, with the mean functional PR across all models and subsets standing at 38.15%, and the mean syntax PR at 65.73%. Focusing on the TBGen subset, all the generated testbenches fell below the 50% threshold in both line and toggle coverage. This limitation suggests that LLM-based testbench generation often fails to cover key signals and corner cases, potentially leaving critical design bugs undetected. To underscore the practical importance of coverage, we conducted an ablation within the MGen subset by evaluating LLM-generated reference models using both the original testbenches and those refined by human experts. As shown in Figure 9, roughly 30% of errors went undetected when using low-coverage testbenches, clearly demonstrating that coverage gaps have an impact on the reliability.

### Analysis

**Task Quality** As illustrated in Figure 6, the observed functional PRs of LLMs across our collection of real-world tasks range from 15% to 50%. This distribution reflects the carefully controlled task difficulty and the effective differentiation of model capabilities, in contrast to earlier benchmarks that frequently emphasized surface-level correctness or template-based problem structures, which can inadvertently inflate performance and obscure model limitations.

Moreover, the use of coverage-enhanced testbenches plays a central role in revealing subtle functional errors, as illustrated in Figure 9. For example, the functional PR of the reference model using GPT-4.1 dropped by roughly 27% after increasing testbench coverage. This decline indicates that more fine-grained test cases are effective at uncovering functional issues in model-generated code. The result not only demonstrates the necessity of our approach but also highlights a broader point: to properly assess a model's actual capability limits in tasks such as code generation or task-oriented reasoning, the evaluation must include methods that reveal hidden failure modes and expand test-case coverage, rather than relying solely on surface-level correctness checks. This underscores the need for tasks to be carefully constructed, detailed, and sufficiently challenging to expose the true limitations of model reasoning and generation across different autonomous-system workflows.

**Design Rationality** To better understand how different stages of the end-to-end evaluation pipeline interact, we conduct a Pearson correlation analysis (Sedgwick 2012) over multiple indicators covering specification understanding, code synthesis, local debugging, and final functional correctness. As shown in Figure 7, all pairwise correlations exceed 0.5, and the average surpasses 0.8. These results reveal two important insights.

First, the strong correlations suggest that performance in intermediate reasoning or synthesis stages is highly predictive of final task success, supporting the hypothesis that failures in LLMs often originate from early-stage semantic mis-

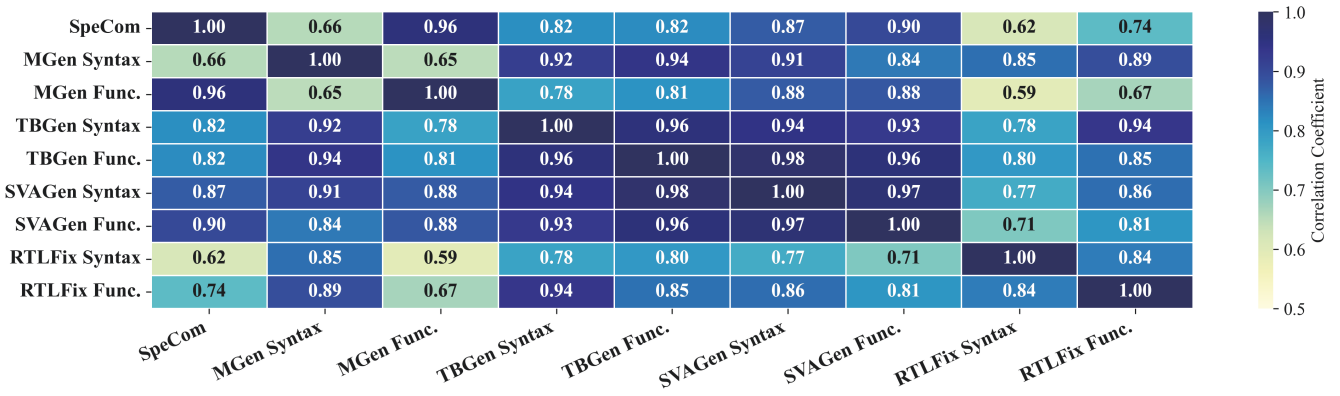


Figure 7: Pearson Correlations Between the Metrics of Different Task Types.

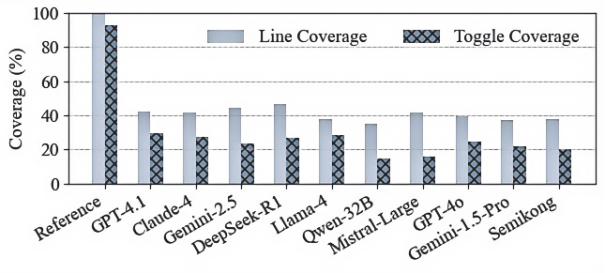


Figure 8: TBGen Coverage Results.

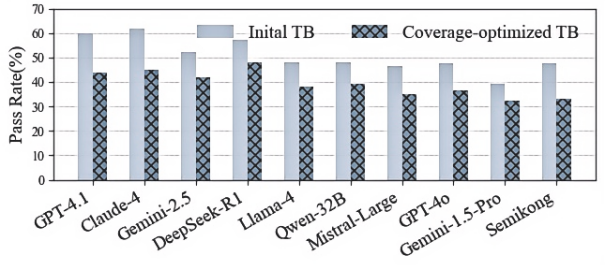


Figure 9: The Impact of TB Coverage on Func.PR of MGen.

interpretation rather than execution-level faults. Second, the correlation pattern provides empirical support for system-level training and evaluation paradigms. If upstream and downstream stages are strongly aligned, then end-to-end optimization strategies, which include multi-stage alignment, holistic supervised fine-tuning, are empirically justified.

**Further Discussion** Our evaluation shows that, despite the promising outlook, integrating LLMs into real hardware design and development workflows remains a substantial challenge. In the context of functional verification, current models still exhibit high error rates, low reliability, and considerable uncertainty. This stems from the probabilistic nature of LLMs: they are prone to hallucination and can produce outputs that are syntactically fluent yet functionally incorrect or semantically incoherent. In integrated-circuit development, this weakness is magnified, as even minor errors in RTL code or assertions can trigger severe downstream failures, costly silicon re-spins, or product delays (He and Yu 2024; Xu et al. 2025).

More broadly, these observations extend beyond hardware design and reflect a fundamental limitation in applying current LLMs to demanding scientific and engineering workflows. Across domains that require precise reasoning, trustworthy decision making, and fidelity to formal or executable semantics, the gap between surface correctness and functional correctness remains a critical barrier to deployment.

Addressing this challenge requires rethinking how we train, align, and evaluate LLMs. It's essential to build end-to-end datasets grounded in industrial practice, strengthen the alignment between upstream and downstream tasks, and adopt fine-grained evaluation methodologies. Achieving these goals will require deeper, more transparent collaboration between industry and academia.

## Conclusion

In this work, we introduced FIXME, a comprehensive benchmark designed to evaluate a broad spectrum of hardware design functional verification skills. Comprising 747 question-answer pairs across five distinct sub-domains, FIXME bridges the gap between academic research and the complexities of real-world design applications. We evaluated over 10 state-of-the-art LLMs on FIXME. Despite recent advancements in LLM-Aided Design, even the most capable models still lag significantly behind human performance on FIXME. This underscores the challenges of functional verification tasks within practical, real-world applications and highlights areas for future LLM development.

## Acknowledgments

We thank all anonymous reviewers for their valuable comments. This work is supported by the National Key Research and Development Program (Grant No.2024YFB4405600) and the National Natural Science Foundation of China under NSFC (Grant No. 92464301). We gratefully acknowledge the support from these programs.

## References


Abdollahi, M.; Yeganli, S. F.; Baharloo, M. A.; and Baniasadi, A. 2025. Hardware design and verification with large language models: A scoping review, challenges, and open issues. *Electronics*, 14(1): 120.

Anthropic. 2025. Introducing Claude 4. Anthropic Blog.

Bai, Y.; Hamad, G. B.; Suhaib, S.; and Ren, H. 2025. Assertionforge: Enhancing formal verification assertion generation with structured representation of specifications and rtl. *arXiv preprint arXiv:2503.19174*.

Barbe, D. F. 2013. *Very large scale integration (VLSI): fundamentals and applications*, volume 5. Springer Science & Business Media.

Bergeron, J.; and Autodesk, I. 2003. *Writing Testbenches: Functional Verification of HDL Models*. Norwell, MA, USA: Springer Science and Business Media LLC, 2nd edition. ISBN 9781402074018.

Blocklove, J.; Garg, S.; Karri, R.; and Pearce, H. 2023. ChipChat: Challenges and Opportunities in Conversational Hardware Design. In *2023 ACM/IEEE 5th Workshop on Machine Learning for CAD (MLCAD)*, 1–6.

Chang, K.; Chen, Z.; Zhou, Y.; Zhu, W.; kun wang; Xu, H.; Li, C.; Wang, M.; Liang, S.; Li, H.; Han, Y.; and Wang, Y. 2024a. Natural language is not enough: Benchmarking multi-modal generative AI for Verilog generation.

Chang, K.; Wang, K.; Yang, N.; Wang, Y.; Jin, D.; Zhu, W.; Chen, Z.; Li, C.; Yan, H.; Zhou, Y.; et al. 2024b. Data is all you need: Finetuning llms for chip design via an automated design-data augmentation framework. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 1–6.

Chang, K.; Wang, Y.; Ren, H.; Wang, M.; Liang, S.; Han, Y.; Li, H.; and Li, X. 2023. ChipGPT: How far are we from natural language hardware design. *arXiv preprint arXiv:2305.14019*.

Comanici, G.; Bieber, E.; Schaekermann, M.; Pasupat, I.; Sachdeva, N.; Dhillon, I.; Blistein, M.; Ram, O.; and Dan, . 2025. Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities. *arXiv preprint arXiv:2507.06261*.

Fang, W.; Li, M.; Li, M.; Yan, Z.; Liu, S.; Zhang, H.; and Xie, Z. 2024. AssertLLM: Generating Hardware Verification Assertions from Design Specifications via Multi-LLMs. In *2024 IEEE LLM Aided Design Workshop (LAD)*, 1–1.

Fang, W.; Wang, J.; Lu, Y.; Liu, S.; Wu, Y.; Ma, Y.; and Xie, Z. 2025. A Survey of Circuit Foundation Model: Foundation AI Models for VLSI Circuit Design and EDA. arXiv:2504.03711.

Fu, Y.; Zhang, Y.; Yu, Z.; Li, S.; Ye, Z.; Li, C.; Wan, C.; and Lin, Y. C. 2023. Gpt4aigchip: Towards next-generation ai accelerator design automation via large language models. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 1–9. IEEE.

Google. 2024. Gemini 1.5: Our next-generation model, now available for private preview. https://blog.google/technology/ai/google-gemini-next-generation-model-1-5-pro/.

Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; Zhang, X.; Yu, X.; Wu, Y.; Wu, Z.; Gou, Z.; Shao, Z.; Li, Z.; Gao, Z.; and and others. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948*.

hdlbits.01xz.net. 2024. HDLBits. https://hdlbits.01xz.net/wiki/index.php/Main_Page. Accessed: August 2, 2025.

He, Z.; and Yu, B. 2024. Large language models for eda: Future or mirage? In *Proceedings of the 2024 International Symposium on Physical Design*, 65–66.

IEEE. 2025. LLM-Aided Design Conference. https://iclad.ai/. Accessed: Jun. 30, 2025.

Kang, M.; Liu, M.; Hamad, G. B.; Suhaib, S.; and Ren, H. 2024. FVEval: Understanding Language Model Capabilities in Formal Verification of Digital Hardware. *arXiv preprint arXiv:2410.23299*.

Li, M.; Fang, W.; Zhang, Q.; and Xie, Z. 2024. Specllm: Exploring generation and review of vlsi design specification with large language model. *arXiv preprint arXiv:2401.13266*.

Liu, M.; Ene, T.-D.; Kirby, R.; Cheng, C.; Pinckney, N.; Liang, R.; Alben, J.; Anand, H.; Banerjee, S.; Bayraktaroglu, I.; Bhaskaran, B.; Catanzaro, B.; Chaudhuri, A.; Clay, S.; Dally, B.; Dang, L.; Deshpande, P.; Dhodhi, S.; Halepete, S.; Hill, E.; Hu, J.; Jain, S.; Jindal, A.; Khailany, B.; Kokai, G.; Kunal, K.; Li, X.; Lind, C.; Liu, H.; Oberman, S.; Omar, S.; Pasandi, G.; Pratty, S.; Raiman, J.; Sarkar, A.; Shao, Z.; Sun, H.; Suthar, P. P.; Tej, V.; Turner, W.; Xu, K.; and Ren, H. 2024a. ChipNeMo: Domain-Adapted LLMs for Chip Design. arXiv:2311.00176.

Liu, M.; Pinckney, N.; Khailany, B.; and Ren, H. 2023. Verilogeval: Evaluating large language models for verilog code generation. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 1–8. IEEE.

Liu, S.; Fang, W.; Lu, Y.; Zhang, Q.; Zhang, H.; and Xie, Z. 2024b. RTLCoder: Outperforming GPT-3.5 in Design RTL Generation with Our Open-Source Dataset and Lightweight Solution. In *2024 IEEE LLM Aided Design Workshop (LAD)*, 1–5.

Liu, T.; Tian, Q.; Ye, J.; Fu, L.; Su, S.; Li, J.; Wan, G.-W.; Zhang, L.; Wong, S.-Z.; Wang, X.; et al. 2024c. ChatChisel: Enabling Agile Hardware Design with Large Language Models. In *2024 2nd International Symposium of Electronics Design Automation (ISEDA)*, 710–716. IEEE.

Lu, H.; Xing, Y.; Gupta, A.; and Malik, S. 2025. Hierarchical Formal Verification of Hardware. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.

Lu, Y.; Liu, S.; Zhang, Q.; and Xie, Z. 2024. RTLLM: An Open-Source Benchmark for Design RTL Generation with Large Language Model. In *Proceedings of the 29th Asia and South Pacific Design Automation Conference*, ASPDAC '24, 722–727. IEEE Press. ISBN 9798350393545.

Ma, R.; Yang, Y.; Liu, Z.; Zhang, J.; Li, M.; Huang, J.; and Luo, G. 2024. VerilogReader: LLM-Aided Hardware Test Generation. In *2024 IEEE LAD*, 1–5.

MacCallum, N.; and Lee, J. 2025. GPT-4.1 Prompting Guide. OpenAI Cookbook. Accessed on April 14, 2025.

Mano, M. M.; and Ciletti, M. D. 2012. *Digital Design: With an Introduction to the Verilog HDL*. Hoboken, NJ, USA: Prentice Hall, 5th edition. ISBN 9780132774208.

Meta AI. 2025. The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation. Meta AI Blog.

Mistral AI. 2025. Mistral Large 2. Web Page. Information synthesized from secondary sources including technical blogs and research papers citing the model (e.g., Kamath, 2025; Tanner, 2025; Zhang et al., 2025).

Naveed, H.; Khan, A. U.; Qiu, S.; Saqib, M.; Anwar, S.; Usman, M.; Akhtar, N.; Barnes, N.; and Mian, A. 2023. A comprehensive overview of large language models. *ACM Transactions on Intelligent Systems and Technology*.

OpenAI. 2024. Hello GPT-4o. https://openai.com/index/hello-gpt-4o/.

Pei, Z.; Zhen, H.; Yuan, M.; Huang, Y.; and Yu, B. 2024. BetterV: Controlled Verilog Generation with Discriminative Guidance. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 40145–40153. PMLR.

pentagoniac. 2024. SEMIKONG-70B. https://huggingface.co/pentagoniac/SEMIKONG-70B.

Pulavarthi, V.; Nandal, D.; Dan, S.; and Pal, D. 2025. AssertionBench: A Benchmark to Evaluate Large-Language Models for Assertion Generation. arXiv:2406.18627.

Qayyum, K.; Ahmadi-Pour, S.; Jha, C. K.; Hassan, M.; and Drechsler, R. 2024. Llms for hardware verification: Frameworks, techniques, and future directions. In *2024 IEEE 33rd Asian Test Symposium (ATS)*, 1–6. IEEE.

Qiu, R.; Zhang, G. L.; Drechsler, R.; Schlichtmann, U.; and Li, B. 2024. AutoBench: Automatic Testbench Generation and Evaluation Using LLMs for HDL Design. In *Proceedings of the 2024 ACM/IEEE International Symposium on Machine Learning for CAD*. New York, NY, USA.

Qwen Team. 2025. QwQ-32B: Embracing the Power of Reinforcement Learning. Qwen Blog.

Sedgwick, P. 2012. Pearson's correlation coefficient. *Bmj*, 345.

Spear, C.; and Tumbush, G. 2012. *SystemVerilog for Verification: A Guide to Learning the Testbench Language Features*. New York, NY: Springer, 3rd edition. ISBN 9781461407140.

Swan, S. 2006. SystemC transaction level models and RTL verification. In *Proceedings of the 43rd annual Design Automation Conference*, 90–92.

Synopsys. 2015. Delivering Functional Verification Engagements. *Synopsys White Paper*.

Synopsys, Inc. 2023. *VCS User Guide*. Synopsys, Inc., Mountain View, CA. Version V-2023.12.

Thakur, S.; Ahmad, B.; Pearce, H.; Tan, B.; Dolan-Gavitt, B.; Karri, R.; and Garg, S. 2024. VeriGen: A Large Language Model for Verilog Code Generation. *ACM Trans. Des. Autom. Electron. Syst.*, 29(3).

Thakur, S.; Blocklove, J.; Pearce, H.; Tan, B.; Garg, S.; and Karri, R. 2023. Autochip: Automating hdl generation using llm feedback. *arXiv preprint arXiv:2311.04887*.

Tsai, Y.; Liu, M.; and Ren, H. 2024. RTLFixer: Automatically Fixing RTL Syntax Errors with Large Language Model. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, DAC '24. New York, NY, USA. ISBN 9798400706011.

Wan, G.-W.; Wang, Y.; Wong, S.-Z.; Xiong, J.; Chen, Q.; Zhang, J.; Zhang, M.; Ni, T.; Xing, M.; Hua, Y.; et al. 2025. GenBen: A Generative Benchmark for LLM-Aided Design. In *Arxiv*.

Wang, L.-T.; Chang, Y.-W.; and Cheng, K.-T. T. 2009. *Electronic Design Automation: Synthesis, Verification, and Test*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN 9780080922003.

Wang, N.; Yao, B.; Zhou, J.; Hu, Y.; Wang, X.; Jiang, Z.; and Guan, N. 2025. Veridebug: A unified llm for verilog debugging via contrastive embedding and guided correction. In *2025 IEEE International Conference on LLM-Aided Design (ICLAD)*, 61–67. IEEE.

Wang, X.; Wan, G.-W.; Wong, S.-Z.; Zhang, L.; Liu, T.; Tian, Q.; and Ye, J. 2024. ChatCPU: An Agile CPU Design & Verification Platform with LLM. In *61st ACM/IEEE Design Automation Conference (DAC'24)*, 6.

Wilson Research Group; and Siemens EDA. 2022. 2022 Functional Verification Study.

Wong, S.-Z.; Wan, G.-W.; Liu, D.; and Wang, X. 2024. VGV: Verilog generation using visual capabilities of multimodal large language models. In *2024 IEEE LLM Aided Design Workshop (LAD)*, 1–5. IEEE.

Wu, N.; Li, Y.; Yang, H.; Chen, H.; Dai, S.; Hao, C.; Yu, C.; and Xie, Y. 2024. Survey of machine learning for software-assisted hardware design verification: Past, present, and prospect. *ACM Transactions on Design Automation of Electronic Systems*, 29(4): 1–42.

Xu, K.; Sun, J.; Hu, Y.; Fang, X.; Shan, W.; Wang, X.; and Jiang, Z. 2024. MEIC: Re-thinking RTL Debug Automation using LLMs. In *2024 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*.

Xu, Q.; Stok, L.; Drechsler, R.; Wang, X.; Zhang, G. L.; and Markov, I. L. 2025. Revolution or Hype? Seeking the Limits of Large Models in Hardware Design. *arXiv preprint arXiv:2509.04905*.

Yang, S.; Wille, R.; and Drechsler, R. 2014. Determining Cases of Scenarios to Improve Coverage in Simulation-based Verification. In *Proceedings of the 27th Symposium on Integrated Circuits and Systems Design*, SBCCI '14. New York, NY, USA: Association for Computing Machinery. ISBN 9781450331562.

Zhang, Z.; Chadwick, G.; McNally, H.; Zhao, Y.; and Mullins, R. 2023. LLM4DV: Using Large Language Models for Hardware Test Stimuli Generation. In *Machine Learning for Systems 2023*.