

CoverAssert: Iterative LLM Assertion Generation Driven by Functional Coverage via Syntax-Semantic Representations

Yonghao Wang¹, Yang Yin¹, Hongqin Lyu^{1,2}, Jiaxin Zhou³, Zhiteng Chao¹, Mingyu Shi⁴, Wenchao Ding², Yunlin Du⁵, Jing Ye^{1,2}, Tiancheng Wang¹ and Huawei Li^{1,2}

¹State Key Lab of Processors, Institute of Computing Technology, CAS, Beijing, China

²University of Chinese Academy of Sciences, Beijing, China

³Beijing Normal University, China; ⁴Nanjing University, China; ⁵University of Newcastle, Australia

{wangyonghao22, yinyang22, lvhongqin24}@mails.ucas.ac.cn, zhoujiaxin@mail.bnu.edu.cn,

chaozhiteng@ict.ac.cn, mingyu.shi@smail.nju.edu.cn, dingwenchaohd@gmail.com,

Yunlin.Du@uon.edu.au, {yejing, wangtiancheng, lihuawei}@ict.ac.cn

Abstract—LLMs can generate SystemVerilog assertions (SVAs) from natural language specs, but single-pass outputs often lack functional coverage due to limited IC design understanding. We propose CoverAssert, an iterative framework that clusters semantic and AST-based structural features of assertions, maps them to specifications, and uses functional coverage feedback to guide LLMs in prioritizing uncovered points. Experiments on four open-source designs show that integrating CoverAssert with AssertLLM and Spec2Assertion improves average improvements of 9.57% in branch coverage, 9.64% in statement coverage, and 15.69% in toggle coverage.

Index Terms—Functional Verification, Assertion Generation, Large Language Model, Feature Fusion, Coverage Feedback

I. INTRODUCTION

Functional verification is essential in IC design, ensuring RTL code meets architectural specifications. Assertion-Based Verification (ABV) improves visibility and cuts debugging time by up to 50% [3], [7], but translating ambiguous specs into accurate SVAs is slow and difficult [5], [8]. LLM-based methods can generate assertions from specs or RTL [4]–[6], [9]–[12], yet comprehensive functional coverage remains challenging.

A key challenge is accurately linking functional intent in specifications to assertions. High similarity among assertion codes and overlapping signal names across modules often lead LLMs to mistake lexical resemblance for true semantics, making it difficult to identify uncovered functional points and provide refinement feedback.

To address this, we propose CoverAssert, a lightweight feedback framework for LLM-based assertion generation. It iteratively identifies uncovered functions, enabling precise coverage analysis and guiding assertion refinement.

The contributions of this paper are summarized as follows:

- 1) We propose CoverAssert, a lightweight LLM-based iterative SVA framework that evaluates functional coverage via syntax–semantic assertion representations—the first to introduce coverage feedback into LLM assertion generation, compatible with existing methods.

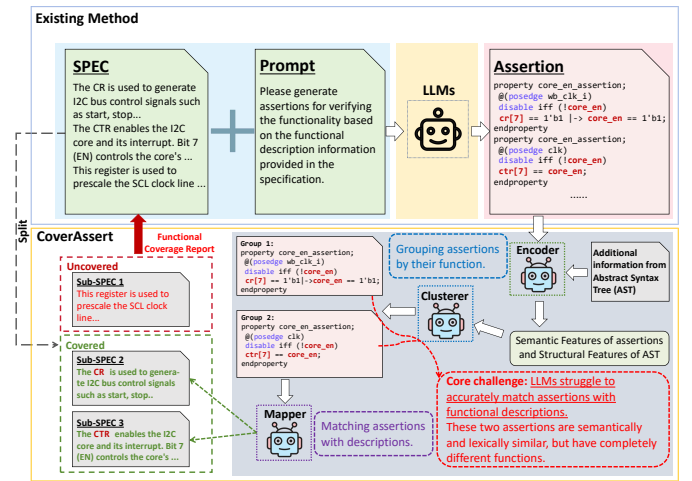


Fig. 1: Our proposed framework, CoverAssert, directly enhances assertion generation by accurately identifying uncovered functional descriptions in the specification and providing feedback to prioritize the completion of uncovered functional points, thereby improving overall verification coverage.

- 2) CoverAssert fuses semantic embeddings with variable positions in syntax trees, enabling precise matching between assertions and functional descriptions.
- 3) Integrated with two SOTA methods, CoverAssert improved branch, statement, and toggle coverage by 9.5%, 9.64%, and 15.69% on four open-source circuits.

II. BACKGROUND

The early work of automated hardware assertion generation proposed by Rahul Kande et al. [11], who pioneered the application of LLMs to generate assertions. AssertLLM by Fang et al. [6] was a milestone, handling full specification files to produce detailed SVAs for all architectural signals. More recently, Wu et al. [12] introduced Spec2Assertion, using

progressive regularization and Chain-of-Thought prompting to generate high-quality assertions directly from specification.

III. FRAMEWORK OF COVERASSERT

We propose CoverAssert, a functional coverage-guided assertion generation framework. Its six-module workflow is shown in Fig. 2.

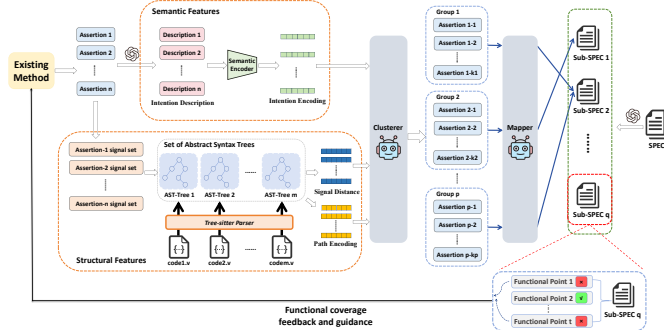


Fig. 2: The CoverAssert framework integrates with existing assertion generation methods.

1) **Semantic Feature Extraction:** High syntactic similarity limits direct embeddings, so we first use an LLM (e.g., ChatGPT-4o) to extract intent, and then encode them using Qwen3-Embedding into vectors T for semantic discrimination.

2) **Signal Structural Features Extraction:** For each assertion pair, signals are extracted and their pairwise longest common ancestor (LCA) distances are averaged; each signal’s path in the AST is concatenated into a padded vector to capture hierarchical and positional structure.

3) **Clustering:** Assertions are filtered by structural distance, clustered by semantic features, and then cluster labels are one-hot encoded and fused with PCA-reduced structural vectors for final semantically and structurally consistent grouping.

4) **Specification Split and Functional Points Extraction:** The specification is split into Sub-SPECs by functional module using an LLM, and fine-grained validation functional points—atomic statements derived directly from the spec—extracted by an LLM for precise coverage feedback.

5) **Assertion-to-Specification Mapping:** Each assertion group is matched to the most relevant Sub-SPEC, and individual assertions are aligned to specific functional points via signals and descriptions to identify coverage gaps.

6) **Coverage-Driven Feedback Loop:** Uncovered Sub-SPECs and functional points are fed back to guide iterative assertion generation, focusing on under-verified regions until all Sub-SPECs reach the coverage threshold $\theta = 0.85$.

IV. EXPERIMENTS

A. Experimental Setup

Benchmark data is from an open-source dataset [1], with correctness checked via Cadence JasperGold (v21.12.002). CoverAssert was compared with AssertLLM [6] and Spec2Assertion [12], all using GPT-4o under identical conditions. We recorded the total generated N , syntax-correct S ,

and FPV-passed SVAs P , along with branch BFC , statement SFC , and toggle TFC coverage within each assertion’s Cone of Influence [2]. Four designs of varying scales were selected for evaluation, as summarized in Table I.

TABLE I: Summary of designs.

Design Name	Func. Description	LoC	Num. of Cells
I ² C	Serial communication protocol.	5369	756
SHA3	Hash function computation.	141185	22228
ECG	Biological signal acquisition.	398686	59084
Pairing	Cryptographic key exchange.	1561498	228287

B. Experimental Results

As shown in Table II, we evaluated CoverAssert with AssertLLM and Spec2Assertion on four circuits. *CoverAssert-n* denotes results after n feedback iterations. The first iteration significantly boosts coverage, and the second further improves most metrics, demonstrating the accuracy of our feedback-guided approach.

TABLE II: Performance comparison of integrating CoverAssert with AssertLLM and Spec2Assertion.

Method	Metrics	I ² C	SHA3	ECG	Pairing
AssertLLM	<i>N/S/P</i>	127/112/58	31/31/26	44/44/19	32/32/12
	<i>BFC</i> (%)	80.23	92	82.22	76.12
	<i>SFC</i> (%)	82.26	90.24	80.74	83.63
	<i>TFC</i> (%)	62	78.41	57.89	67.12
AssertLLM + CoverAssert-1	<i>N/S/P</i>	149/133/76	69/67/47	62/60/31	51/49/26
	<i>BFC</i> (%)	85.33	98.82	98.89	90.96
	<i>SFC</i> (%)	87.27	94.63	97.78	93.18
	<i>TFC</i> (%)	66.98	86.57	81.86	70.33
AssertLLM + CoverAssert-2	<i>N/S/P</i>	173/151/85	82/80/56	78/76/38	64/62/32
	<i>BFC</i> (%)	86.79	100	98.89	93.85
	<i>SFC</i> (%)	88.87	95.06	97.78	94.67
	<i>TFC</i> (%)	80.52	90.71	83.27	81.46
Spec2Assertion	<i>N/S/P</i>	90/89/49	42/42/28	35/29/19	45/41/15
	<i>BFC</i> (%)	87.87	90.89	79.51	83.58
	<i>SFC</i> (%)	89.44	87.78	81.05	66.67
	<i>TFC</i> (%)	64.13	65.43	73.92	46.53
Spec2Assertion + CoverAssert-1	<i>N/S/P</i>	105/104/61	65/65/46	51/42/29	76/68/34
	<i>BFC</i> (%)	89.34	94.36	97.33	86.91
	<i>SFC</i> (%)	90.32	90.84	96.48	81.06
	<i>TFC</i> (%)	75.06	74.03	75.26	58.35
Spec2Assertion + CoverAssert-2	<i>N/S/P</i>	117/116/67	77/77/55	68/58/37	101/90/51
	<i>BFC</i> (%)	89.34	95.07	98.02	86.91
	<i>SFC</i> (%)	90.32	92.84	96.48	82.94
	<i>TFC</i> (%)	82.42	77.75	82.85	61.96

V. CONCLUSION

This paper presents CoverAssert, a framework that improves SVA generation via functional coverage feedback. Combining semantic and structural features, it groups and matches assertions, guiding generators to prioritize uncovered points and enhance coverage, while integrating with existing methods.

ACKNOWLEDGMENT

This paper is supported in part by the Chinese Academy of Sciences under grant No. XDB0660103, and in part by the National Natural Science Foundation of China (NSFC) under grant No.(62090024). The corresponding authors are Tiancheng Wang and Huawei Li.

REFERENCES

- [1] Iwls 2005 benchmaeks. <https://iwls.org/iwls2005/benchmarks.html>.
- [2] Jaspergold: the Next Generation. https://community.cadence.com/cadence_blogs_8/b/breakfast-bytes/posts/jasper-ml.
- [3] Yael Abarbanel, Ilan Beer, Leonid Gluhovsky, Sharon Keidar, and Yaron Wolfsthal. Focs—automatic generation of simulation checkers from formal specifications. In *Computer Aided Verification: 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000. Proceedings 12*, pages 538–542. Springer, 2000.
- [4] Fnu Aditi and Michael S Hsiao. Hybrid rule-based and machine learning system for assertion generation from natural language specifications. In *2022 IEEE 31st Asian Test Symposium (ATS)*, pages 126–131. IEEE, 2022.
- [5] Yunsheng Bai, Ghaith Bany Hamad, Syed Suhaib, and Haoxing Ren. Assertionforge: Enhancing formal verification assertion generation with structured representation of specifications and rtl. *arXiv preprint arXiv:2503.19174*, 2025.
- [6] Wenji Fang, Mengming Li, Min Li, Zhiyuan Yan, Shang Liu, Zhiyao Xie, and Hongce Zhang. Assertllm: Generating and evaluating hardware verification assertions from design specifications via multi-llms. *arXiv preprint arXiv:2402.00386*, 2024.
- [7] Harry Foster. 2024 wilson research group ic/asic functional verification trend report. <https://resources.sw.siemens.com/zh-TW/white-paper-2024-wilson-research-group-ic-asic-functional-verification-trend-report/>.
- [8] Harry Foster. Assertion-based verification: Industry myths to realities (invited tutorial). In *Computer Aided Verification: 20th International Conference, CAV 2008 Princeton, NJ, USA, July 7-14, 2008 Proceedings 20*, pages 5–10. Springer, 2008.
- [9] Hongqin Lyu, Yonghao Wang, Yunlin Du, Mingyu Shi, Zhiteng Chao, Wenxing Li, Tiancheng Wang, and Huawei Li. AssertGen: Enhancement of LLM-aided Assertion Generation through Cross-layer Signal Bridging. *arXiv preprint arXiv:2509.23674*, 2025.
- [10] Marcelo Orenes-Vera, Margaret Martonosi, and David Wentzlaff. Using llms to facilitate formal verification of rtl. *arXiv preprint arXiv:2309.09437*, 2023.
- [11] Yonghao Wang, Jiaxin Zhou, Hongqin Lyu, Zhiteng Chao, Tiancheng Wang, and Huawei Li. DeepAssert: An LLM-Aided Verification Framework with Fine-Grained Assertion Generation for Modules with Extracted Module Specifications. *arXiv preprint arXiv:2509.14668*, 2025.
- [12] Fenghua Wu, Evan Pan, Rahul Kande, Michael Quinn, Aakash Tyagi, David Kebo, Jeyavijayan Rajendran, and Jiang Hu. Spec2Assertion: Automatic pre-rtl assertion generation using large language models with progressive regularization. *arXiv preprint arXiv:2505.07995*, 2025.