Manickam Muthiah. International Journal of Engineering Research and Applications
www.ijera.com
ISSN: 2248-9622, Vol. 15, Issue 10, October 2025, pp 01-05

RESEARCH ARTICLE OPEN ACCESS

Failure Context Presenter for a Test Environment

Manickam Muthiah*
**(Principal Engineer, ARM Inc., Chandler, Arizona, USA.)*

ABSTRACT
In pre-silicon verification, regression failures are often difficult to analyze due to the scattered nature of simulation logs. Identifying tests with similar failures or contextual patterns requires significant manual effort, resulting in inefficiency and delay. This paper presents a novel component, the Failure Context Presenter (FCP), integrated into the test/verification environment to automatically generate and present the failure context across regression suites. The FCP collects data from multiple tests/simulations, organizes checks and errors information into structured status files, and maintains a dedicated Failure Context Presenter Database. This novel approach introduces a Failure Context Presenter module embedded within the test environment, using a unique CES packet mechanism and a dedicated Failure Context Database. These features enable real-time correlation of failures, early termination of redundant simulations, and context-driven debug visibility—capabilities not offered by current coverage or debug automation frameworks. The solution is independent of verification framework and can be applied across industry-standard methodologies such as OVM, VMM, and UVM.
Keywords **–** Failure Context Generation, Checking Context Metadata, Failure Context Presenter, Failure Context Database, Checks & Errors Status Packets, Partial Simulation Termination Threshold, Verification Checks Aggregation.

Date of Submission: 28-09-2025 Date of acceptance: 08-10-2025

I. INTRODUCTION

In functional verification, especially in simulation environments, failures from regression suites are challenging to debug. Traditionally, engineers must parse through large volumes of simulation logs to identify whether a failure is isolated or repeated across multiple tests. This process is time-consuming, resource-intensive, and error-prone. Studies have shown that debug/triage can consume up to 50% of overall verification cycle time [10].

Modern verification flows increasingly demand automation and correlation across regression runs [2][6][11]. While existing methods such as assertion-based verification [10], coverage closure [8], and regression management [11] address portions of the debug challenge, none provide an integrated, methodology-independent mechanism to present a unified failure context across regression runs. This gap motivates the introduction of the Failure Context Presenter (FCP), a novel component in the test environment.

II. PROBLEM STATEMENT

When a regression test fails, engineers often cannot immediately identify related failing tests, the specific checks involved, or the test's configuration/intent. Without this contextual data, debugging requires examining logs from multiple runs, which is highly inefficient. The existing debug paradigm typically operates in isolation per failing simulation, with no unified mechanism to cross-reference failures across runs [10].

III. PROPOSED SOLUTION

The Failure Context Presenter (FCP) introduces a new module within a test environment that automatically gathers checks and errors data from different components, correlates it across tests, and generates a consolidated failure context.

A. Module Integration in Verification Environment

- The FCP module resides in the testbench alongside standard verification components.
- It receives errors/checks information from verification components such as monitors, drivers, scoreboards and sequence/test.
- Supports OOP-based methodologies such as SystemVerilog/OVM/UVM/VMM [1], [2], [3].

Figure 1 shows the block diagram of a test environment with the Failure Context Presenter module.

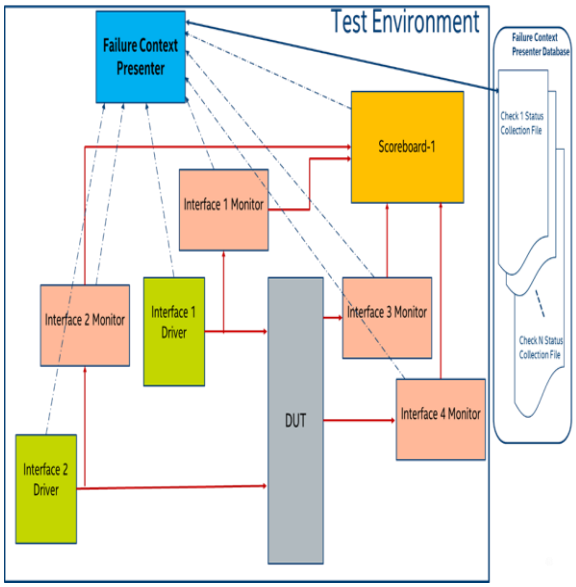


Figure 1: Block Diagram of a Test Environment with the Failure Context Presenter Module

B. CES Packet Mechanism

Communication between testbench components and the FCP uses Checks & Errors Status (CES) Packets. A Check CES Packet is sent at the end of a test or simulation to report the number of checks performed, while an Error CES Packet is sent immediately when an error is detected to report the ordinal position of the error.

Figure 2 illustrates the CES Packet format and sample CES packets, while Figure 3 provides additional CES packet examples.

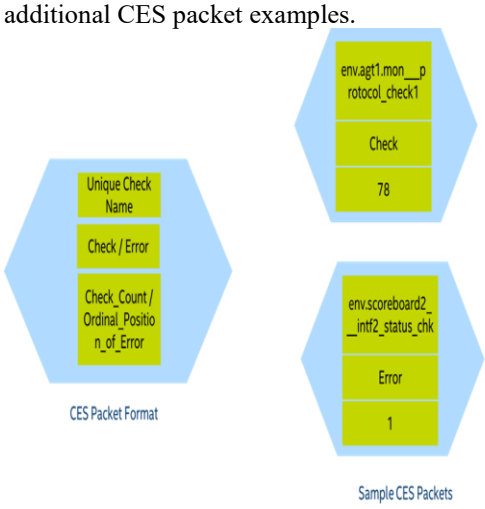
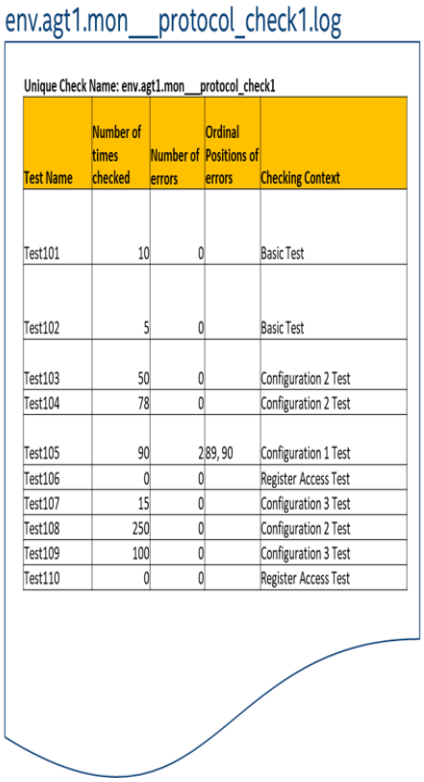


Figure 2: CES Packet Format and Sample CES Packets

Figure 3: Additional Sample CES Packets

C. Failure Context Presenter Database

Each type of check corresponds to a Status Collection File in the FCP Database. These files store error counts, check counts, and contextual information across multiple test runs. Examples of Status Collection Files are shown in Figures 4–6.



Test Name	Number of times checked	Number of errors	Ordinal Positions of errors	Checking Context
Test101	10	0		Basic Test
Test102	5	0		Basic Test
Test103	50	0		Configuration 2 Test
Test104	78	0		Configuration 2 Test
Test105	90	2	89, 90	Configuration 1 Test
Test106	0	0		Register Access Test
Test107	15	0		Configuration 3 Test
Test108	250	0		Configuration 2 Test
Test109	100	0		Configuration 3 Test
Test110	0	0		Register Access Test

Figure 4: Sample Status Collection File for env.agt1.mon___protocol_check1

env.scoreboard1___intf1_outdata_chk.log

Unique Check Name: env.scoreboard1___intf1_outdata_chk

Test Name	Number of times checked	Number of errors	Ordinal Positions of errors	Checking Context
Test201	500	5	10, 45, 108, 324, 446	Interface 1 Random Mode Stress Test
Test202	100	0		Interface 1 Normal Mode Test
Test203	30	1	27	Interface 1 Random Mode Test
Test204	0	0		Register Access Test
Test205	69	0		Interface 1 Normal Mode Test

Figure 5: Sample Status Collection File for env.scoreboard1___intf1_outdata_chk

env.scoreboard3___intf3_outdata_chk.log

Unique Check Name: env.scoreboard3___intf3_outdata_chk

Test Name	Number of times checked	Number of errors	Ordinal Positions of errors	Checking Context	Run Status
Test501	4	2	1, 2	Interface3 Basic Test	Completed
Test502	5	2	1, 2	Interface3 Basic Test	Completed
Test503	12	4	4, 6, 8, 9	Interface3 Normal Mode Test	Completed
Test504	15	3	3, 4, 5	Interface3 Normal Mode Test	Completed
Test505	23	5	2, 6, 7, 9, 20	Interface3 Random Mode Test	Completed
Test506	2	1	2	Interface3 Random Mode Test	Partially Terminated
Test507	4	1	4	Interface3 Random Mode Test	Partially Terminated

Figure 6: Sample Status Collection File for env.scoreboard3___intf3_outdata_chk

D. Checks & Errors Status Table

During a simulation, the FCP maintains a local Checks & Errors Status Table summarizing the number of checks, errors, and their positions. At the end of simulation, this data is written to the Failure Context Presenter database. A sample table is shown in Figure 7.

Unique Check Name	Number of times checked	Number of errors reported	Ordinal Position of Error
env.scoreboard___outdata_chk	25	2	12, 19
env.scoreboard___outcredit_chk	14	0	
env.agt1.drv__credit_chk	50	0	
env.agt2.drv__credit_chk	67	0	
env.agt3.mon__protocol_chk1	80	0	
env.agt3.mon__protocol_chk2	32	0	
env.agt4.mon__protocol_chk	45	0	
test___counters_chk	55	1	1
test___end_of_test_chk	79	0	

Figure 7: Sample Checks & Errors Status Table

E. Threshold-Based Termination

When repeated failures are observed for the same check beyond a configurable Partial Termination Threshold, the FCP can terminate redundant simulations early, conserving time and computational resources.

IV. RESULTS AND DISCUSSION

1. Failure Context Generation

By analyzing CES Packets across simulations, the FCP produces failure contexts that group tests with similar failures. Checking Context metadata provides deeper understanding of failures relative to test intent.

2. Verification Checks Report

Aggregated statistics across regression runs are compiled into a Verification Checks Report. An example report is provided in Figure 8.

Unique Check Name	Number of tests checked	Number of times checked	Number of errors reported
env.scoreboard___outdata_chk1	26	550	0
env.agt3.mon___protocol_chk3	55	1560	0
env.agt4.drv__credit_chk	60	479	0
test___counters_chk	100	100	5
test___end_of_test_chk	100	100	26

Figure 8: Verification Checks Report

3. Debug and Productivity Gains

- Reduces manual log inspection time significantly.
- Enhances debug quality with contextual metadata.
- Saves computational resources by aborting redundant failing simulations.
- Supports historical comparison across DUT (Device Under Test) / TB (Testbench) versions.

4. Industry Alignment

While prior works in coverage closure [8], assertion-based verification [10], and debug automation [11][12] have improved productivity, they do not embed failure correlation within the testbench. The FCP differs fundamentally by integrating into the verification environment itself, generating structured failure context during simulation rather than after the fact. This represents a novel paradigm shift from post-run triage to in-situ debug intelligence. Unlike existing approaches, it introduces the CES packet mechanism, failure context database with Checking Context, and threshold-based termination of redundant simulations, thereby offering a unique, methodology-independent solution.

V. DIFFERENCE FROM EXISTING TECHNOLOGY

Distinct contributions of the FCP include:

- Real-time generation of failure contexts.
- A Failure Context Presenter module operating within the verification environment itself
- A standardized CES Packet protocol for structured reporting across testbench components.
- A Partial Termination Threshold mechanism enabling real-time termination of redundant failing simulations.
- Association of Checking Context metadata with results.

Unlike assertion-based verification [10] or manual coverage-driven approaches [8], the FCP delivers contextualized, cross-run analysis in real-time.

VI. USES, BENEFITS AND LIMITATIONS

Uses and Benefits:

1. **Methodology Independence:** The FCP is compatible with all OOP-based verification methodologies, including OVM, VMM, and UVM [2][3].
2. **Standardized Reporting:** The CES packet protocol provides a uniform way for all components (monitors, drivers, scoreboards, test/sequence) to report checks and errors, improving data consistency.
3. **Accelerated Debug:** By presenting consolidated failure contexts, the FCP eliminates the need for manual log inspection across multiple regression runs, reducing debug turnaround time.
4. **Resource Optimization:** The Partial Termination Threshold avoids redundant simulations, saving both time and compute resources.
5. **Improved Checking Coverage Closure:** Aggregated statistics in the Verification Checks Report help verification teams track checking completeness and error density, aiding closure strategies [8].
6. **Historical Comparisons:** Lightweight databases allow easy comparison of regressions across DUT/TB versions, enabling regression-to-regression benchmarking.
7. **Contextual Understanding:** Checking Context metadata links failures to test intent/configuration, making failure analysis semantically meaningful.
8. **Scalability:** The architecture supports large regression suites where multiple simulations access the database concurrently without conflicts.
9. **Debug Knowledge Retention:** Databases can be archived, providing a historical debug knowledge base for future analysis or audits.

Limitations

1. **Dependency on OOP Environments:** Implementation is currently limited to SystemVerilog-based flows[1].
2. **Threshold Sensitivity:** Incorrect configuration of the Partial Termination Threshold could lead to premature abortion of useful simulations.
3. **Initial Integration Overhead:** Testbench components must be modified to emit CES packets, requiring some upfront engineering effort.
4. **Database Growth:** While individual files are lightweight, extremely long regressions may lead to large aggregated failure context databases, requiring maintenance.

VII. CONCLUSION

This paper has introduced the Failure Context Presenter (FCP) as a novel contribution to verification methodology. The FCP provides a structured, automated approach to correlate, analyze, and act upon simulation failures across regression runs. By combining CES-based metrics, Checking Context, and configurable threshold-based termination, it streamlines debugging and optimizes simulation resources. These innovations enable faster debug, more efficient use of resources, and deeper contextual understanding of regression failures, marking a significant step forward in pre-silicon verification automation. Future extensions may include machine learning to predict root causes [14][15].

REFERENCES

Standards & Manuals:
[1] IEEE Std 1800-2017, IEEE Standard for SystemVerilog—Unified Hardware Design, Specification, and Verification Language, 2017.
[2] Accellera Systems Initiative, UVM Class Reference Manual, Version 1.2, 2017.
[3] Accellera Systems Initiative, OVM Reference Manual, Version 2.1, 2013.

Books:
[4] J. Bergeron, Writing Testbenches: Functional Verification of HDL Models, Springer, 2003.
[5] G. Tumbush and C. Spear, UVM Cookbook, Doulos, 2010.
[6] J. Fitzpatrick, Advanced Verification Topics, Mentor Graphics, 2010.
[7] M. Keating and P. Bricaud, Reuse Methodology Manual for System-on-a-Chip Designs, Springer, 2002.

Journal Papers:
[8] S. Ramesh and V. Srinivasan, "Efficient Functional Coverage Closure Strategies," Microelectronics Journal, vol. 65, 2017, pp. 38–47.
[9] M. Enamul Amyeen, S. Venkataraman, M. W. Mak, "Microprocessor System Failures Debug and Fault Isolation Methodology," Proc. Int. Test Conf., 2009.
[10] H. D. Foster, "Assertion-Based Verification: Motivation and Methodology," DAC, 2002, pp. 580–585.

Conference Proceedings:
[11] C. Faanes and A. Bhattacharya, "Queue-based Latency Analysis for Simulation Acceleration," IEEE/ACM ICCAD, San Diego, 2020, pp. 25–32.
[12] C-Y. Huang, W. Huang, C. Yang, "Integrated Verification Ecosystem for Regression Management, Coverage Convergence, and Debug Automation," DVCon Taiwan, 2024.
[13] Revolutionary Debug Techniques to Improve Verification Productivity, Cadence/DVCon Tutorial, 2014.
[14] M. Smytzek, M. Eberlein, L. Grunske, A. Zeller, "How Execution Features Relate to Failures: An Empirical Study and Diagnosis Approach," arXiv preprint, 2025.