2026
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SANTA CLARA, CA, USA
MARCH 2 - 5, 2026

Unified AI-Driven Verification: Combining Spec-RAG, Memory Networks, and Generative AI

Seonghee Yim, Insoo Jang, Hanna Jang, Youngsik Kim
Samsung Electronics Co., LTD.

SAMSUNG

Contents

- Background and Motivation
- Problem Statement
- Related works
  - Design-Verification Lifecycle Insight
  - STM vs LTM Definition
- Main Idea
  - Dynamic STM/LTM Decision Logic
  - Integrated Pipeline
- Experimental results
- Conclusion and Future works

Background & Motivation

- Why is SoC Verification Getting Harder?
  - Heterogeneous IP blocks and deep hierarchy
  - Rapidly evolving specifications
  - Massive verification environments and simulation logs
  - Debugging depends on historical knowledge + cross-domain reasoning

- Key Pain Point
  - Engineers manually search fragmented data across multiple systems

Problem Statement

- Limitations of Existing Approaches
  - Data scattered across different repositories (Oracle RDB, MongoDB, Jira, file servers)
  - Manual spreadsheets and static trace matrices fail to track temporal & hierarchical changes
  - Need for automated, explainable, version-aware traceability
- Existing AI approaches:
  - Document-centric RAG → lacks structural reasoning
  - Statistical bug prediction → ignores relational context

Deep Dive: Design–Verification Lifecycle Insight

- **High-Change Phase (ML1–ML2)**
  - Frequent updates
  - Sparse relations → semantic expansion dominates

- **Mature Phase (ML3–ML4)**
  - Stable artifacts
  - Dense relations → structured reasoning dominates

✓ *A single retrieval strategy cannot handle both effectively.*

*ML : Design Maturity Level

Deep Dive: Dual-Memory RAG - Adaptive Context

Short-Term Memory

STM

Immediate Conversational Context

- Recent Test Logs (MongoDB)
- Real-time Sanity Data
- Temporary Working Scratchpad

Mechanism

Dynamic Weighting

High-Change Phase: STM ↑

Mature Phase: LTM ↑

Long-Term Memory

LTM

Persistent Historical Archive

- Design Specs (Oracle RDB)
- Knowledge Graph (HippoRAG)
- Cross-Version revision history

Deep Dive: LLM-Database Reasoning - NL to Data

Oracle RDB (Structured)

- Design Integration Metadata
- Verification Plans & Maturity
- Release Tags & Versioning

MongoDB (Unstructured)

- Test Logs & Error Traces
- Regression Outcomes
- Coverage Metrics

Main Idea

- Core Insight

  Traceability quality depends on **when and how STM and LTM are weighted**, not merely on what is retrieved.

Data Traceability $\propto$ Debugging Efficiency

Framework Orchestration:
The Integrated Pipeline

Step 01: Decision

Decision Model

Detects ML stage, computes dynamic STM/LTM weights, and analyzes issue type.

Step 02: Retrieval

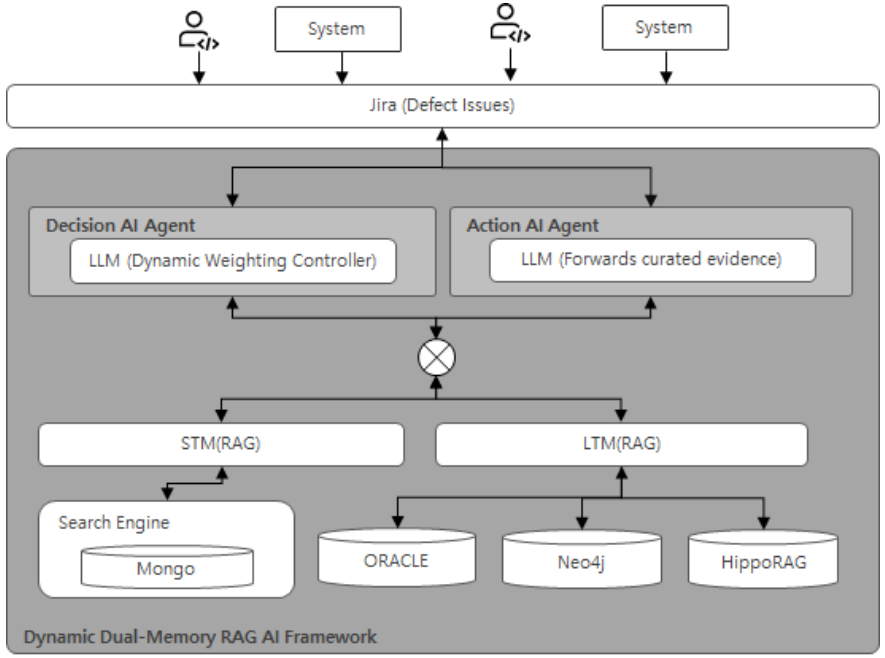
Evidence Gathering

Collects data from STM (logs, diffs) and LTM (specs, design history, HippoRAG).

Step 03: Analysis

Action Model

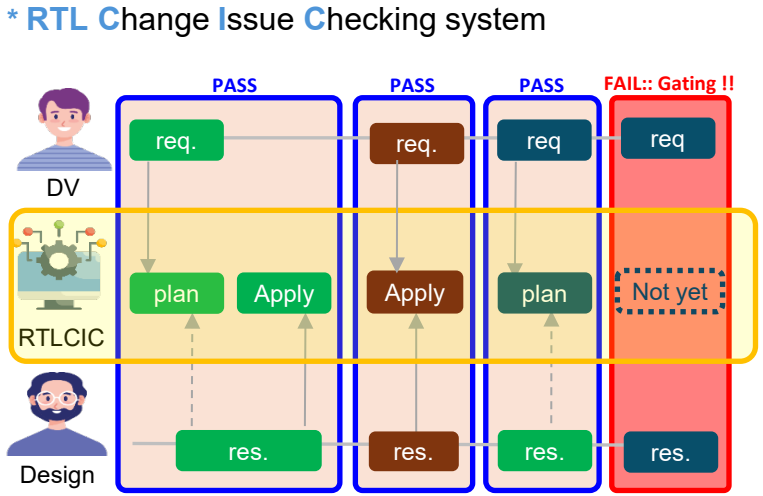
Performs deep analysis of curated evidence to generate structured root cause descriptions.



Experimental Setup

Static Analysis using small SoC project

- 34 weeks of real SoC data
- 398 Jira defect issues
- RTLCIC* provide defect resolving label
- Retrospective evaluation



Metric

- Estimated debugging efficiency gain (System vs Jira resolving time)

Experimental Results:
Best Practice Performance &Impact

Near-Human

Traceability

High accuracy in data-rich industrial SoC contexts.

-27%

Debug Time

Reduction in estimated root-cause analysis time

.

75%

Efficiency Gain

Average gain for information mismatch/missing errors.

Variant	Debug Time Reduction	Notes
Baseline	0%	keyword/manual search
Static Dual Memory	11%	no dynamic adaptation
Dynamic Weight (HippoRAG + STM/LTM)	27%	no auto issue triage with graph reasoning

Ablation Study Summary

✓ Dual-Memory Separation ✓ Dynamic Weighting Logic ✓ LLM-Driven Decision Logic ✓ HippoRAG Graph Reasoning

Type 1 Errors:
Rapid Resolution of Information Mismatches

Problem & Challenge

- Definition: False alarms, specification discrepancies (e.g., memory maps, SFR settings).
- Challenge: Often lead to wasted debugging cycles due to misunderstanding or outdated information.

Framework Impact & Result

- Impact: Immediate resolution using grounded, up-to-date data from STM.
- Result (Best Case): **75%**

Average Efficiency Gain

Type 2 Errors: Addressing Functional Failures Requiring RTL Modifications

Problem & Challenge

- Definition: Functional failures, incorrect port declarations, redundant errors in older versions.
- Challenge: Complex and chronic bugs, often dependent on development maturity and specific design versions.

Framework Impact & Result

- Impact: Reduces debugging time by leveraging LTM for historical patterns and design changes.
- Result (Best Case): **30%**

Average Efficiency Gain

Case Study: Eliminating Redundant Debugging (Type 2 Example)

Scenario

- SYSREG connection error in "Block A" during EVT0 phase.
- Detected in EVT0_ML2_DEV06, resolved in EVT0_ML2_DEV09.

Traditional Approach

- Repetitive manual debugging across multiple RTL versions.
- Time-consuming and prone to human error.

Framework Action

- RTLCIC system tracks target fix versions.
- Leverages LTM for historical context and resolution patterns.

Impact (Best Case)

- Eliminates inefficient repetitive runs.
- Significantly enhances verification quality.
- Reduces debugging time to near zero for such cases.

Emphasizing Best Case Scenarios & Future Potential

Best Case Scenarios

- Presented results reflect optimal conditions and demonstrate the framework's maximum potential.
- Real-world context: Acknowledge variability due to data sparsity or unique project complexities.

Value & Future Potential

- Value Proposition: Even with variability, the framework consistently provides significant gains over manual methods.
- Future Potential: Continuous improvement through adaptive learning and enhanced graph reasoning.

Conclusion

Core Framework

- Dynamic Dual-Memory RAG Framework for SoC verification.
- Integration of STM (volatile data) and LTM (stable relational info).
- Adaptive memory utilization based on development stages.

Key Impact

- Significant acceleration of debugging workflows.
- Effective capture of distinct information needs across ML1-ML4.
- Practical AI-assisted approach for industrial-scale SoC design.

"Enabling explainable, data-driven traceability for modern SoC development lifecycles."

Future Work

Real-Time Verification Pipelines

Integration for **proactive anomaly detection** and automated debugging suggestions before manual analysis.

Enhanced Graph-Based Reasoning

Exploring **Graph Neural Networks (GNNs)** to infer latent architectural relationships and failure propagation paths.

Adaptive Learning Loops

Refining retrieval ranking and STM/LTM weighting through **Human–AI feedback signals** from verification engineers.

Automated Issue Classification

AI-driven module to **replace noisy human-labeled data**, improving decision model reliability and robustness.

Questions & Discussion

Thank you for your attention.

REFERENCES

[1] Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, Yu Su, "HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models" NeurIPS 2024

[2] Zihong He, Weizhe Lin, Hao Zheng, Fan Zhang, Matt W. Jones, Laurence Aitchison, Xuhai Xu, Miao Liu, Per Ola Kristensson, Junxiao Shen1, X-Intelligence Labs, 2 University of Cambridge, 3 University of Bristol, 4 Columbia University, 5 Meta, "Human-inspired Perspectives: A Survey on AI Long-term Memory" arXiv 2411.00489

[3] Seonghee Yim, Hanna, Sunchang Choi and Seonil Brian Choi, "Accelerating SOC Verification using Process Automation and Integration" Design Verification Conference (DVCon), 2020

Appendix I : Results & Ablation

- Strongest gains in false-spec, connection, test bench issues

Temporal Analysis: Issue Trend vs. Memory Utilization

Stage	Issue count*(Avg/Week)	Volume Memory Dominance	Rationale
ML1	Highest(16)	STM ↑	frequent diffs, unstable logs
ML2	High(10)	STM > LTM	active feature rollout
ML3	Moderate(6)	LTM ↑	structural & relational issues
ML4	Low(2)	LTM > STM	convergence & verification closure