



Comparative Study of UVM and ML-Infused Methodologies for Next-Gen SoC Storage Verification

Amit Kumar*

Independent Researcher, USA

* Corresponding Author Email: kalayat.amit.rana@gmail.com - ORCID: 0000-0002-5247-7487

Article Info:

DOI: 10.22399/ijcesn.5420

Received : 11 May 2026

Revised : 05 July 2026

Accepted : 10 July 2026

Keywords

Machine Learning Verification,
Universal Verification
Methodology,
Portable Stimulus Standard,
SoC Storage Controllers,
Coverage-Driven Verification

Abstract:

It is a challenge to verify the high complexity of Storage controllers in System-on-Chip (SoC) designs. Universal Verification Methodology (UVM) is the de facto standard for functionally verifying these chips, but UVM suffers from fundamental limitations on modern Storage controllers that include complex NVMe protocols, multi-queue management, error correction, and power management states. ML-based verification methods, especially those utilizing extensions to the Portable Stimulus Standard, mark a new shift toward clever and data-driven methods for generating stimulus. These leverage Reinforcement Learning agents to solve for the best exploration policies, use Supervised Learning to predict the coverage impact of a simulation, and exploit Generative models to sample tests based on learned representations. Evaluations of real-world NVMe storage controller designs show that ML-improved methods can considerably reduce simulation time for high coverage. Furthermore, during late stages of verification, when naive random-based techniques show diminishing returns for convergence, the ML-based approach achieves statistically meaningful coverage improvement through learned coverage strategies without redundant tests. The method also discovers additional critical bugs in corner cases that are commonly missed by naive random-based verification techniques. Resource utilization analysis showed that although feature extraction and inference have some per-simulation overheads, due to a meaningful reduction in the number of simulations, the total cost in terms of computation and verification time is substantially reduced. The trained models are also transferable to other design variants with almost no retraining. The scalability studies show good scalability properties as design complexity increases, and the ML techniques were found to match the original performance benefits as the complexity increased. Overall, the results show that ML-assisted verification can complement conventional techniques and that hybrid flows utilizing existing infrastructure with tailored AI optimizations are practical for semiconductor enterprises to adopt.

1. Introduction

Complex SoC storage controller verification is increasingly difficult, with sizes exceeding billions of transistors. The 2024 Siemens EDA and Wilson Research Group IC/ASIC Functional Verification Trend Report shows that functional verification accounts for 65–70% of the total SoC design cycle. This increasing percentage is the result of the increasing complexity of designs, which are growing faster than the predictions of Moore's Law [1]. As the average size of high-performance SoC implementations is now more than 50 million gates, the verification groups are unable to scale

productivity and meet their coverage goals within the project's available time.

For many years, standard verification methodologies, like the Universal Verification Methodology (UVM), have been used in the functional verification of the new storage controllers, as they provide predictable frameworks and reusable building blocks to perform structured verification. Storage controllers now implement many features, which include NVMe protocol with advanced command queuing (multi-queuing), complex error correction codes, PCIe Gen5, multiple power states, and security engines. Recent work on semiconductor design verification for AI and machine learning systems has found that

functional coverage closure must be performed over an increasing number of states while preventing false therefore requires increasingly complex coverage metrics to deal with corner cases and interaction effects [10].

In this article, a comparison of UVM-based verification methodologies vs. next-generation Machine Learning (ML)-based technologies, such as PSS with ML capabilities in the areas of coverage convergence, bug detectivity, resource and time efficiency, and scalability, is presented when applied to next-generation SoC storage controller verification environments where intelligence and high degrees of automation are critical for success.

2. Universal Verification Methodology Architecture and Limitations

Perhaps the most widely used functional verification methodology in the semiconductor industry today is the Universal Verification Methodology (UVM). The 2024 Siemens EDA and Wilson Research Group (WRG) verification trend report suggests the UVM is near saturation in companies developing advanced semiconductor products and is used in a variety of applications, including automotive electronics, mobile systems, and high-performance computing [1]. The report notes the large ecosystem that now exists for organizations using UVM, including commercially developed and open-source verification IP libraries as well as training materials and best practices that have evolved over years of industry adoption.

Built on SystemVerilog, UVM promotes a layered architecture for creating reusable and modular testbenches. Five concepts form the foundation of the UVM philosophy: driver, monitor, scoreboard, sequencer, and agent. A UVM testbench driver takes high-level transactions and generates pin-level activity. A monitor is an interface observer, a scoreboard verifies the correctness of the interface responses, a sequencer generates stimulus, and an agent is a testbench-specific wrapper. UVM testbenches for storage controller verification often implement protocol-specific VIP for PCIe, NVMe, and SATA protocols. As per the NVMe Base Specification, one of the requirements of a storage controller is that it must support multiple command types for controller configuration and data transfer using I/O commands. Each command type has its own verification challenges [7].

The central operation of a UVM verification environment is constrained-random stimulus generation, which is the generation of valid transaction sequences that satisfy constraints. The algorithms used in constraint solvers are generally

considered to be state of the art but can have performance issues when faced with highly correlated constraints, as is typical for modern storage controller designs [8]. Coverage-driven verification techniques use functional coverage groups to constrain random exploration by specifying goals and closure across multiple dimensions.

UVM has several fundamental limitations when applied to modern SoC storage controllers. In constrained random mode, UVM is inefficient in exploring the design space since its tests often lack diversity and fail to explore corner cases. The coverage convergence is logarithmic. While we quickly achieve the first coverage points, achieving the last percentage requires a large number of simulation runs. A major bottleneck is the large amount of manual effort required to close coverage gaps. Trends in verification suggest that coverage-driven verification requires verification engineers to spend substantial effort analyzing coverage gaps, speculating about missing coverage scenarios, and making tests more directed or distributions more random [10]. This process is not well-defined but relies on the engineer's skill and intuition. Furthermore, UVM's static constraint solving is not well-suited to certain types of complex interdependencies, such as command queue ordering constraints, cache coherency protocols, and power state transitions.

3. ML-Enhanced Verification Methodologies and PSS Framework

An advanced verification approach, leveraging Machine Learning, focuses on cleverly generating stimuli and verification scenarios based on knowledge acquired throughout the verification process. The Portable Stimulus Standard establishes an abstract specification layer, devoid of implementation details, that outlines the purpose of verification. It was developed and ratified by Accellera. According to the Verification Academy Portable Stimulus resources, portable stimulus (PSS) lets verification teams express their verification intent at a higher level of abstraction so that what needs to be verified can be specified in a portable manner in simulation, emulation, FPGA (field-programmable gate array) prototyping, and post-silicon validation environments rather than how to verify. PSS provides constructs for modeling actions, data flows, resources, and coverage goals for the verification environment.

The PSS solution with ML includes data-driven optimization of the verification procedure by learning from historical simulation data. The verification framework with ML consists of the

following components. As a foundation, PSS models a verification problem space in terms of declarative constructs, independent from its implementation details. Using constraint-solving algorithms, the PSS solver generates concrete test cases to explore the solution space defined by the PSS model.

Machine Learning algorithms can enhance this field on multiple levels. Machine Learning has been applied efficiently for SoC verification. For example, these Reinforcement Learning agents learned exploration policies for verification frameworks (such as for hardware model checking) by solving sequential decision problems in which rewards are maximized [9]. The agent gets input state representations with active verification status and outputs stimulus parameters and a sequence of tests to maximize, e.g., coverage improvement rate and bug discovery rate. The RL agent learns good policies over time by exploring the simulation environment and learning from it.

Supervised learning models can be trained to predict coverage impact, using data from previous simulation runs that map test parameters to observed coverage. These models can then be employed to predict the configurations of stimuli that are most likely to cover uncovered states. In semiconductor design verification for power-efficient chips, it has been shown that coverage models and predictive analytics can be combined to reduce verification cost [3]. This allows verification teams to focus limited verification resources on simulations that are likely to progress the verification effort rather than running unnecessary simulations.

Generative models learn latent representations of excellent test patterns by inferring them from a set of successful verifications. In doing so, they embed a high-dimensional stimulus space into low-dimensional manifolds that represent characteristics of the verification process. The generator then produces new test patterns from this distribution that are still successful but in unexplored areas. Research into portable stimulus methodologies has shown that generative techniques can produce diverse, valid test scenarios as an alternative to random generation [6].

Custom engineering is required to integrate ML infrastructure with existing verification flows. Data pipelines are created that extract features from simulation logs, coverage databases, and design metadata. Feature engineering includes transforming source data into representations that are useful to the ML algorithms and knowledge of the storage protocols and SoC architectures. The training loops implement the appropriate learning models and hyperparameter tuning for each ML

component. Inference engines use the models generated during earlier steps to select regression tests.

4. Experimental Methodology and Benchmark Framework

To allow for a comparison to be made, a test bed of benchmark cases will be created based on representative designs of SoC storage controllers. The use of a synthesizable NVMe-compliant storage controller with a PCIe host interface, which implements the advanced features of modern storage subsystems, as the main platform. The NVMe Base Specification defines command processing, queuing, data transfer, error handling, and power management features for storage controllers. Compliance testing for the specification is conducted on a wide variety of administrative and I/O commands, which have their specific verification requirements [7]. These include combinations of command parameters, different types of error conditions, and performance testing requirements.

The controller implemented multiple I/O queue pairs to support parallel commands, end-to-end data protection mechanisms, and power management states to provide power saving based on controller operating modes. Its architecture had hierarchical partitioning into functional subsystems, including protocol interface layers, command processing engines, data path components, and the control plane. This hierarchy creates challenges for verifying functionality, as verification must consider both multiple abstraction levels and the boundaries between subsystem components.

All verification infrastructure, including the UVM and the ML-improved PSS testbenches, was run in parallel against the same set of functional coverage models for a direct comparison. Coverage goals for the testbenches included cross-coverage bins, protocol compliance, corner case scenarios, error injection, and performance stress conditions. Research on machine learning techniques in electronics systems also mentions the need for coverage models to capture functional correctness, performance, and power management capabilities [5]. The authors used similar reference models to verify the design responses and the same simulation environment to avoid tool-dependent results.

The creation of a production-quality testbench for the UVM baseline, as advised by the process for creating UVM testbenches, which includes the following guidance: effective UVM testbenches require good architecture, application of factory patterns, structured sequence libraries, and coverage modeling [8]. The environment is

composed of constrained-random sequence libraries for normal operation, error injection sequences, and corner cases for the protocol. Coverage-driven feedback from coverage analysis was used to adjust the constraint distributions. Iterative refinement was used to modify constraint weights to bias stimulus generation in favor of previously uncovered functionality.

The ML-based approach used PSS to abstract the scenario specification and introduced several ML integration patterns, allowing the RL agent to learn testing strategies through interactions with the simulation environment. Supervised learning identified the test parameters that predicted coverage, filtering the tests to remove those with redundancy. A generative model learned the best patterns of stimuli. According to studies, combining multiple machine learning models generally achieves better results than using any one model by itself in verification tasks [9].

The efficiency is in terms of coverage convergence speed, which is the number of runs of the simulation to reach a certain coverage, and effectiveness, which is measured in terms of the number of unique bugs found per unit of simulation budget. The term "resource utilization" pertains to the costs associated with CPU, Memory, and Storage. The model prediction methods' accuracy and scalability are evaluated. The experiment performed multiple rounds of verification with different random seeds to ensure repeatability. Finally, we calculated the confidence intervals, conducted hypothesis testing, and calculated the effect size. The fine-grained telemetry, including per-simulation coverage deltas, timestamps of bug detections, computational resource consumption, and ablation studies, was conducted.

5. Results Analysis and Performance Comparison

In almost all the metrics, the UVM-only and ML-augmented methods exhibited a large gap in performance. Only in the metric for coverage convergence speed did the ML-augmented method show a large advantage in later stages of the verification process. In contrast, customary UVM simulations required significantly more simulation, and this gap grew as the verification coverage increased. In SoC design verification, machine learning is best used late in the verification process, when random simulation struggles to find uncovered corner cases [9].

Additionally, the coverage convergence curves were different, with UVM showing logarithmic curves typical of random exploration and ML-based methods showing more linear curves due to

learning. Coverage redundancy analysis during testing also showed that for large coverage checkpoints, there were many UVM simulation runs resulting in no new coverage hits, wasting compute effort. The ML approach reduced test redundancy by discarding stimulus configurations expected to have minimal benefit to coverage. Unlikely configurations were rejected at the beginning of the simulation. In the semiconductor industry, one of the most important ways to improve productivity is to eliminate redundant tests, which can be done through smart test selection [10]. The ability to predict the value of a test before running allows for more efficient use of computational resources for tests that will progressively approach the verification goal.

Each identified a similar number of unique bugs per initial verification when moderate coverage wasn't reached, and easy bugs still existed within the design, with later verification revealing complementary strengths of the two methods. With UVM-provided models and constraints, the ML approach also identified additional corner-case critical bugs during verification late in the flow that UVM did not cover in the same number of simulations. Because these bugs required specific stimuli or time relationships to trigger, they would be difficult to detect from random generation but easier to detect from ML-guided generation, which is targeted. This ability to generate targeted stimulus to fill holes in coverage translated to measurable improvements in bug-finding performance, particularly during the most challenging phases of verification.

Analysis of resource consumption indicated a tradeoff between per-simulation and total cost for the verification task. Although ML-based simulations required a greater use of resources for feature By focusing on extraction and inference, the team significantly reduced the number of simulations run, which in turn led to a meaningful decrease in total simulation costs. While the UVM has better per-simulation performance, the total compute hours spent achieving high coverage levels were considerably greater than with the ML approach, showing the importance of considering total campaign efficiency. Memory requirements for ML model training and inference introduce only modest cost overhead for infrastructure. The efficiency increase outweighs this cost.

An initial cost is incurred when training ML models for the simulations to bootstrap the dataset; however, this expense is amortized over the verification time. The models demonstrated transfer learning ability across design variations. Models pretrained on a single controller configuration required only a light fine-tuning process, rather

than complete retraining, showing an efficient knowledge transfer opportunity that allows them to adapt quickly to derivative designs, improving the ROI of the machine learning development. High-value tests with low coverage were false positives, and unpredicted high-value tests were false negatives. The predictions were not entirely accurate, but the collective average was sufficient to help prioritize tests, and cost-benefit analysis suggests that mispredictions were outweighed by the effect on test selection.

In the case of scaling design size, with respect to pure performance, the scaling of the gain observed for the ML-based approaches improved with the design size; as the number of coverage bins and the cov model complexity increased, the relative gain with respect to the UVM increased. In the case of pattern duplication, the customary random

approaches reached a point of diminishing returns, while the ML approaches exhibited an improved scalability, as they leverage the same learned approach regardless of the case study complexity. Human effort was additionally a differentiating factor for engineering productivity. UVM verification required continuous engineering effort to analyze coverage, provide feedback to directed test generation, and tune coverage constraints. By autonomously achieving coverage closure, the ML-improved approach considerably reduces the amount of time engineers need to be involved with the process, because they can focus on erroneous behavior or bugs rather than the details of coverage closure. This productivity increase translates to meaningful cost savings with respect to engineer effort and schedule compression in the verification process.

Table 1: UVM Architecture Components and Characteristics [3, 4]

Drivers	Transaction-to-signal conversion	Pin-level signal generation	Stimulus application
Monitors	Interface observation	Transaction extraction	Behavior capture
Scoreboards	Correctness checking	Expected vs. actual comparison	Validation
Agents	Interface encapsulation	Protocol-specific functionality	Modular reusability

Table 2: ML Integration Techniques in Verification [5, 6]

ML Technique	Learning Paradigm	Primary Function	Key Benefit
Reinforcement Learning	Sequential decision-making	Optimal exploration policy discovery	Consistent coverage gains
Supervised Learning	Prediction from labeled data	Coverage impact estimation	Proactive test filtering
Generative Models	Latent representation learning	Novel stimulus generation	Diverse scenario production
Transfer Learning	Knowledge adaptation	Cross-design model reuse	Reduced retraining requirements
Feature Engineering	Domain knowledge encoding	Raw data transformation	ML-suitable representations
Hyperparameter Optimization	Performance tuning	Model configuration refinement	Accuracy maximization

Table 3: Performance Comparison Metrics [7, 8]

Metric Category	UVM Characteristics	ML-Enhanced Characteristics	Performance Difference
Coverage Convergence	Logarithmic trajectory	Consistent gain maintenance	Substantial reduction in simulations
Test Redundancy	High redundant simulation rate	Predictive filtering reduces waste	Significant efficiency improvement
Bug Detection	Effective in early phases	Superior in corner-case discovery	Additional critical defects found
Resource Per-Simulation	Lower individual overhead	Modest feature extraction cost	Marginally higher overhead
Total Resource Consumption	Higher aggregate requirements	Lower total due to fewer simulations	Substantial aggregate savings
Transfer Learning	Limited cross-design reuse	Efficient model adaptation	Rapid derivative deployment

Table 4: Scalability and Human Effort Analysis [9, 10]

Dimension	UVM Behavior	ML-Enhanced Behavior	Scaling Advantage
Design Complexity Scaling	Near-linear degradation	Sublinear scaling coefficient	Better efficiency with complexity
Pattern Reuse	Minimal generalization	Substantial knowledge transfer	Enhanced cross-variant effectiveness
Engineer Intervention	Continuous manual involvement	Largely autonomous operation	Significant productivity gains
Coverage Analysis Time	Extensive report examination	Automated gap targeting	Reduced manual analysis
Directed Test Development	Frequent new test creation	Minimal directed test needs	Lower development overhead
Constraint Refinement	Iterative manual tuning	Learned optimization	Eliminated tuning cycles

6. Conclusions

The article shows that ML-based verification methodologies have meaningful advantages over customary UVM-based verification methods, particularly for next-generation SoC storage controller verification environments. Clear quantitative advantages in coverage convergence efficiency were demonstrated, as ML-based methodologies achieve considerably superior results in terms of the number of simulations required to meet high coverage levels. Thus, organizations are realizing that approaches based on guided intelligence might hold the key to alleviating bottlenecks in verification productivity in the semiconductor verification process that could impact time to market. Examples include reducing the cost of verification via reduced simulation, improving productivity via automated coverage closure, and improving schedules to accelerate moving the product into a competitive market. The main perception here is that the verification task can be framed as an optimization problem, from which statistical learning methods can benefit from the high dimensionality, sparse feedback, and regularities in the test structure. Classical constrained random approaches do not perform memory-oriented global searches but rather make independent low-cost guesses. Overall, these solutions approach directed exploration through experience-based learning within a single loop, including direct reinforcement learning to learn the optimal policy, prediction with a supervised model to filter for informative stimuli, and generative models to retain validity while exploring new states. Making these solutions real requires making HR and organizational investment choices, from setting up training infrastructure, data pipelines, and model development workflows to enabling model training, validation, deployment, and monitoring. Organizations that want to realize AI/ML solutions, therefore, need to invest in

growing their AI/ML capabilities either through hiring, training, or using third-party consulting, as well as creating tooling to connect the AI frameworks with the verification environment and workflows. Because this training phase is computationally expensive and time-consuming, projects with tight schedules, limited budgets, or insufficient resources to perform iterations may be less likely to adopt machine learning. Overall, this work indicates ML methods are complementary to customary techniques, and hybrid flows are the best path because UVM's mature infrastructure and readily available assets, such as proven testbench architectures, verification IP, debug/process automation, and best-practice reporting, make incremental adoption efficient and effective. For teams building complex storage controllers or subsystems, or in cases where verification requires resource expenditures that are spread across longer periods, increased productivity may be realized by reusing learned models across product generations. Teams should invest early in a verification data infrastructure, allowing rich telemetry data to be collected for the future application of AI, including post-silicon debug correlation, power analysis, and performance modeling. In the future, a marriage between mathematical proof techniques from formal methods, data-driven optimization techniques from machine learning, and broad functional exploration from customary dynamic verification will drive verification flows as semiconductor designs continue to grow in scale and designers demand shorter time to market.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could

have appeared to influence the work reported in this paper

- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.
- **Use of AI Tools:** The author(s) declare that no generative AI or AI-assisted technologies were used in the writing process of this manuscript.

- [8] Chris Spear, "Improving Your SystemVerilog Language and UVM Methodology Skills," Siemens Verification Academy, 2021. Available: <https://verificationacademy.com/topics/systemverilog/improving-your-systemverilog-language-and-uvm-methodology-skills/>
- [9] Ali Hmedat, "Machine Learning Application in the Verification of SoC Designs," LinkedIn, 2024. Available: <https://www.linkedin.com/pulse/machine-learning-application-verification-soc-designs-ali-hmedat-uormf/>
- [10] Niranjana Gurushankar, "Trends in Semiconductor Design Verification for AI and Machine Learning Applications," International Journal For Multidisciplinary Research, 2022. Available: https://www.researchgate.net/publication/388769013_Trends_in_Semiconductor_Design_Verification_for_AI_and_Machine_Learning_Applications

References

- [1] Harry Foster, "IC/ASIC Functional Verification Trend Report—2024," Verification Academy, 2025. Available: <https://verificationacademy.com/topics/planning-measurement-and-analysis/wrg-industry-data-and-trends/2024-siemens-eda-and-wilson-research-group-ic-asic-functional-verification-trend-report/>
- [2] Vikas Tomar, "Using Functional Verification in Testing NVMe SSD Controller Designs," Flash Memory Summit, 2019. Available: https://files.futurememorystorage.com/proceedings/2019/08-07-Wednesday/20190807_TEST-202A-1_Tomar.pdf
- [3] Michael William and Shoaib Aslam, "Power-Efficient Semiconductor Design: A UVM and Machine Learning Approach," ResearchGate, 2024. Available: https://www.researchgate.net/publication/384486633_Power-Efficient_Semiconductor_Design_A_UVM_and_Machine_Learning_Approach
- [4] Andrew Piziali, "Functional Verification Coverage Measurement and Analysis," Kluwer Academic Publishers, 2023. Available: <https://picture.iczhiku.com/resource/eetop/whKdrFJjiYKEDvCm.pdf>
- [5] Khaled A. Ismail and Mohamed A. Abd El Ghany, "Survey on Machine Learning Algorithms Enhancing the Functional Verification Process," Electronics, 2021. Available: <https://www.mdpi.com/2079-9292/10/21/2688>
- [6] Tom Fitzpatrick, "Portable Stimulus," SIEMENS Verification Academy. Available: <https://verificationacademy.com/topics/portable-stimulus/>
- [7] "NVM Express Base Specification," NVM Express, 2022. Available: <https://nvmexpress.org/wp-content/uploads/NVM-Express-Base-Specification-2.0c-2022.10.04-Ratified.pdf>