International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

BRIDGING COVERAGE CLOSURE AND FORMAL VERIFICATION: SCALABLE STRATEGIES FOR AI/ML-DRIVEN HARDWARE DESIGNS

Aparna Mohan

North Carolina State University, Raleigh, North Carolina

aparna.m1988@gmail.com

Abstract

As AI/ML-driven hardware architectures such as neural processing units (NPUs), tensor accelerators, and heterogeneous systems-on-chip (SoCs) continue to increase in complexity, traditional hardware verification methods face growing challenges in achieving comprehensive coverage closure and ensuring correctness. While simulation-based verification scales well and measures coverage effectively, it often misses elusive corner-case bugs. Conversely, formal verification provides strong mathematical guarantees but is hindered by state-space explosion and poor scalability. This paper introduces a hybrid verification framework designed to bridge the gap between simulation and formal methods, offering scalable, automated assurance for AI/ML hardware systems. The framework integrates coverage-guided simulation, automated synthesis of formal properties, and AI/ML-driven test generation to systematically identify and verify uncovered logic and high-risk areas in the design. Through the application of machine learning to forecast coverage gaps, synthesize assertions, and refine verification based on counterexamples, the approach supports adaptive workflows that significantly improve bug detection and formal proof convergence. Ultimately, this work lays a scalable and intelligent foundation for next-generation verification strategies, enabling more autonomous, reliable, and efficient validation pipelines for complex AI/ML hardware.

Keywords: Coverage Closure, Formal Verification, AI/ML Hardware, Hardware Verification, Assertion-Based Verification, Coverage-Driven Testing, Property Checking, Scalable Verification

1. Introduction

The most recent generation of special-purpose hardware, including Neural Processing Units (NPUs), tensor processors, Graphics Processing Unit (GPU) and heterogeneous System-on-Chip (SoC) systems were developed as a result of the accelerated development of artificial intelligence and machine learning (AI/ML) systems, enabling them to execute data-intensive workloads more efficiently and in a shorter time [1][2]. These architectures are designed with a specific focus on implementing deep learning operations with the use of massive parallelism, pipelined execution, approximate arithmetic schemes and hierarchical memory to reduce latency and maximize computational throughput. At training and inference, these platforms are far faster than traditional general-purpose processors by way of assigning workloads to thousands of processing cores and optimizing memory access patterns. The complexity currently growing in such hardware designs is, however, a new and serious issue with respect to verification, particularly in guaranteeing functional correctness, operational safety,

Received: Aug 06, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

reliability and resilience, on very large and dynamic state spaces [3]. The verification with the conventional hardware verification tools have primarily involved the use of simulation-based testing, functional coverage analysis and assertion-based verification as the primary method of testing the correctness of the design and additional potential defects. The techniques may be used to test the functionality of the system through the execution of test cases and quantifying the progress through coverage closure that involves attaining the predetermined targets of the code coverage, branch coverage and functional coverage variables. Although coverage closure is currently the most commonly used completeness measure of verification, it is still unable to be able to detect the presence of corner-case defects, race conditions, or other atypical behavior in extremes and surprising situations of operation [4]. To address these shortcomings, formal verification applies mathematically rigorous techniques such as model checking, equivalence checking, and theorem proving to systematically prove design properties against precise formal specifications, thereby providing stronger theoretical guarantees of correctness [5]. Formal methods allow exploration of states exhaustively and allow building of logical proofs, which are particularly important in detecting errors that cannot be detected by simulation-based testing. Nonetheless, the practical scalability is formally verified through its practical implementation, which is not always feasible due to state-space explosion, complexity of abstraction, requirement of computational resources and the complexity of modeling and verifying large-scale AI/ML accelerator architectures [6]. As the AI/ML hardware becomes more frequently used as an autonomous system, safety-critical platform, health care technology, robotics, intelligent transportation, and high-performance computing infrastructure, the disadvantages of relying solely on isolated verification paradigms become clearer [7]. These systems can malfunction with critical operational, financial, or human safety consequences, which makes the significance of effective and thorough verification plans more significant. This has generated more research and industry interest in hybrid verification techniques that integrate scalability, flexibility and practical efficiency of the simulation-based coverage-driven verification with the mathematical rigor, completeness, and demonstrable correctness of formal reasoning techniques [8]. These integrated frameworks will be capable of offering improved closure of coverage and formal analysis, therefore, enhancing the likelihood of fault detection, raising confidence in system correctness, reducing the resources expended on verification, and raising reliability in successive AI/ML hardware. Lastly, there is a need to develop scalable and end-to-end verification procedures, which would help in the safe, reliable, and trustworthy deployment of advanced AI/ML accelerators into a reality and mission-critical environment.

Though existing electronic design automation (EDA) systems and verification models are more sophisticated, the fact that coverage-based simulation and formal verification models began and are now highly fragmented, de-scoped process, in the industry, further diffuses them [9]. The limitations of simulation-based workflows arise from randomized test generation, coverage-driven metrics, and ad hoc stimulus refinement, which often fail to target rare corner-case failures even when overall coverage appears high [4][10].Meanwhile, formal verification places significant demands on human effort, as it requires substantial expertise to formalize properties, construct suitable abstractions, and guide proof convergence challenges that

Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

become especially pronounced for large-scale AI/ML accelerators with complex control and data flows [5][6]. Recent hybrid verification approaches increasingly rely on heuristic integrations such as leveraging simulation traces to guide formal verification but they still lack a principled framework for translating coverage gaps into formally testable properties, limiting their systematic effectiveness [7][11]. In addition to this, the probabilistic and changing nature of AI/ML workloads containing approximate computation cannot be easily specified with a deterministic measure of correctness [2][3]. The other critical missing element is the lack of application to AI so that verification choices can be made autonomously, despite the fact that AI has proven useful in creating tests in deriving property inferences and state space reduction [8][12]. As a result, verification teams continue to be subjected to the inflated price, time-to-close and chronic probability of uncovered design flaws. The absence of such a scalable and intelligent, structure that brings coverage analytics into close contact with formal reasoning on autopilot, and converts coverage data to formal verification artefacts that can be acted upon, demonstrates the absence of the latter.

In order to address these limitations, the present paper will propose a scaled-up hybrid verification framework that can be developed to overcome the coverage closure and formal verification of AI-enabled hardware designs. It has a straightforward concept: transforming simulation coverage data into input for the automated generation of formal properties, thereby enabling focused, proof-based verification of previously identified or high-risk design regions. It is a framework comprising a combination of coverage-guided simulation, machine-learned property generation and counterexample-based refinement to create a loop-refinement verification pipeline, which is more apt to improve the quality and efficacy of the tests and the proofs as the model advances [1][5][8]. The method significantly reduces human effort. As the goal approaches exhaustive (or binary-like) verification coverage, AI/ML model generation and deployment enable forecasting of coverage gaps, targeting of key verification objectives, and faster convergence to complete verification closure [3][6][11].

The key findings of the article are:

- **Hybrid Verification Framework:** The paper proposes a unified verification approach that combines coverage-guided simulation, automated formal property generation, and AI/ML-based test generation to improve verification accuracy and scalability for AI/ML hardware.
- **AI/ML-Driven Automation:** Machine learning is used to identify high-risk logic regions, generate formal assertions, optimize test stimuli, and accelerate coverage closure, significantly reducing manual verification effort.
- **Scalable Assurance for Complex Designs:** The framework addresses the scalability challenge of verifying AI accelerators and heterogeneous SoCs by integrating formal proofs with simulation and AI prioritization, enabling efficient bug detection and correctness validation at scale.

Thus, this paper presents a scalable hybrid verification framework that combines simulation-based coverage, formal verification, and AI/ML-driven automation to enhance the correctness, efficiency, and reliability of AI/ML hardware design validation.

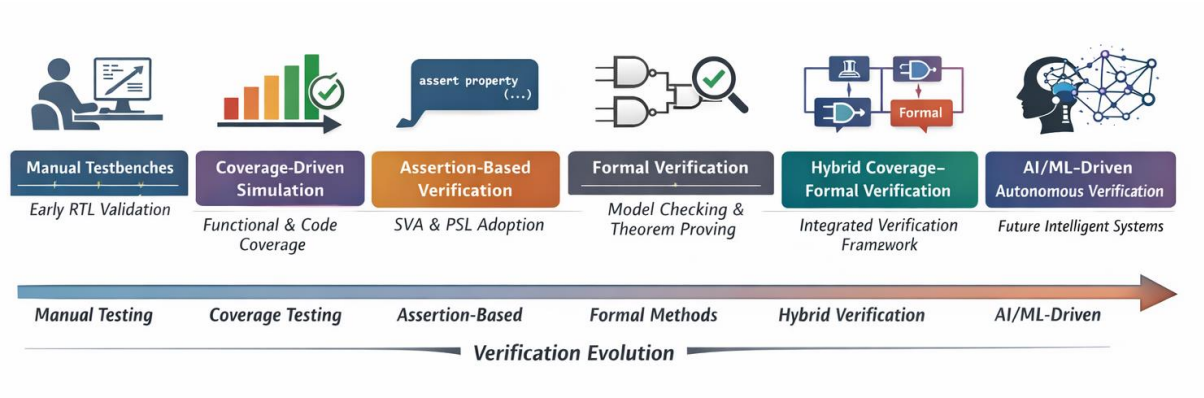
2. Background & Technical Foundations

Received: Aug 06, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

As part of its quest to handle the growing design complexities, coverage-based and formalized correctness assurance systems have emerged as a result of manual test-based validation in the move to handle these growing complexities [13]. As AI/ML accelerators and numerous parallel data aisles increasingly become integrated into more modern systems, the verification of these new architectures needs to take into account massive state space, timing-sensitive behaviour, and workload-sensitive execution patterns. The current paradigms of the current verification are two-fold: coverage-based simulation that is interested in statistical confidence on exhaustive test stimulus, and formal verification that makes sure that something is correct according to the methods of mathematical proof [14]. The coverage metrics, such as code, branch, toggle and functional coverage, are quantitative proxies of verification completeness, and are employed to regulate the test generating and refining processes. Meanwhile, formal verification is going to make use of property checking, equivalence checking, bounded model checking and theorem proving to thoroughly search design behaviors taking into account the possible input [15]. These paradigms are complementary, but in large-scale AI/ML hardware systems are largely very siloed and do not cooperate effectively with each other. Moreover, probabilistic and approximate computation of AI also complicates the specification of correctness constraints in deterministic determination, thus complicating verification problems. There is also new work on AI-aided verification, where machine learning models are applied to predict high-risk logic, optimize test stimulus and reason over formal assertions based on simulation traces [16]. However, no single technical foundation that unites coverage analytics, formal reasoning, and AI-based automation in a methodical manner exists. This would prove valuable in developing scalable, adaptive and correctness-assured verification pipelines over the next generation intelligent architecture hardware platforms [17].

Figure 1: Verification Paradigms for AI/ML Hardware



2.1 Coverage Closure in Hardware Verification

The increase in the coverage is what is known as the coverage closure; predetermined levels of coverage are attained to prove adequate coverage of the design of the test [13]. The metrics of coverage give quantitatively based feedback on coverage in the simulation process in the portions of the RTL, control logic and functional intent. Some of the more popular are code coverage (statement, branch, toggle), functional coverage (scenario-based conditions) and assertion coverage (verification of temporal properties). Coverage closure is used to guide

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

stimulus generation, prioritize tests, and refine regression suites in industrial workflows, forming the foundation for simulation-based validation methods [14]. Comparability, however, need not be accompanied by coverage; despite coverage that seems all encompassing, low-probability fault tracks can be missed. This weakness is especially critical in hardware that implements AI/ML, which is implemented as parallel execution, pipelined arithmetic and approximate computation producing huge non-linear state spaces. In addition, the coverage measures also fall into the false confidence since the achievement of an area of coverage is not equal to semantic accuracy. This has necessitated a reaction to such coverage closure to offer higher precision of correctness coverage that overrules the statistical confidence and enables confirmation of the presence of logic paths discovered to exist.

2.2 Formal Verification Foundations

Formal verification is mathematically legitimate verification, where a hardware implementation is proved to meet a set of properties on all conceivable input combinations [15]. In order to reach exhaustive system behaviour exploration, systems Model checking, bounded model checking, equivalence checking and theorem proving are used to support exploration of system behaviour without randomised simulation. The formal methods have the advantage of uncovering corner cases, deadlocks, inaccessible states and faults of protocols that are easy and difficult to identify through simulation. Instead, they are frequently limited by state-space explosion and the inability to make abstractions and write down correctness properties. Formal verification is particularly difficult to apply at a large scale to AI/ML accelerators, to large datapaths and multidimensional tensor operations, and parallel control structures. Its demands of realistic implementation that are characteristic of modular decomposition, abstraction refinement and reduction of cones of influence are required to ensure that it is tractable [16]. The formal verification, regardless of the superior correctness guarantees, has not been popular due to the complexity of the tools used, the run time cost and expertise needed to develop useful and complete properties. These limitations spur the investigation of automatic inference of properties, mixed methods scalability, and high formal rigor of simulation.

2.3 AI/ML-Driven Hardware Architectures - Characteristics & Verification Implications

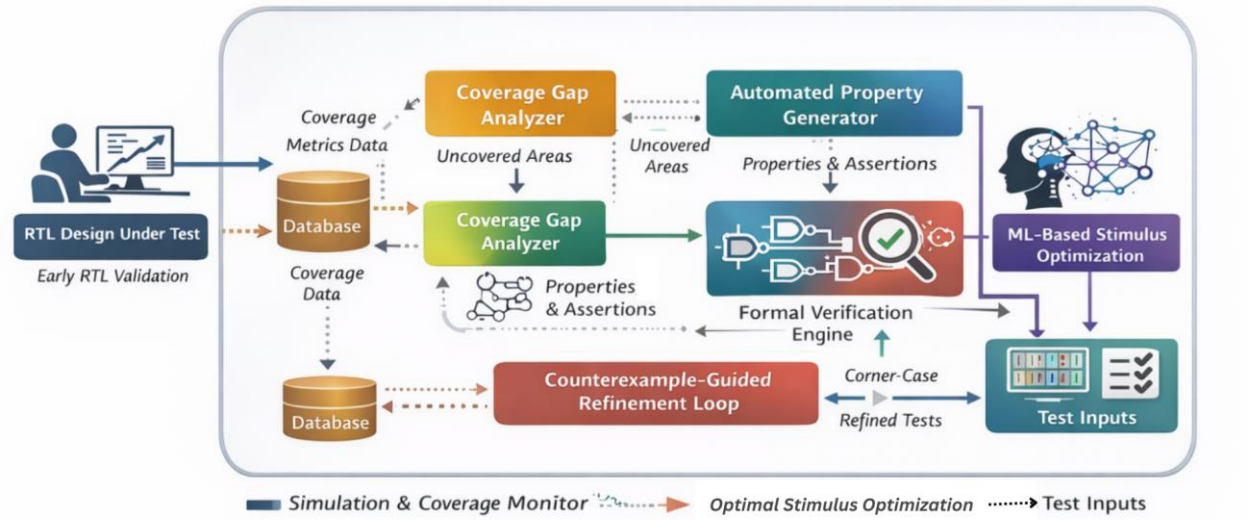
The produced hardware architecture of AI/ML poses novel verification issues that depend on the intricacy of the architecture, workload association, and rough models of computing [17]. Heterogeneous accelerator fabrics are designed based on massive parallel in order to demand logical correctness, a verification landscape must demand correctness, and also statistical consistency, numerical tolerance and workload consistent behavior. Besides this, AI/ML devices are commonly deployed in safety-critical systems (self-driving cars), on medical scans, or on financial predictions, which further increases the cost of the oversights of failure. Such workload-sensitive and probabilistic phenomena are hard to check through the existing systems of verification, and new technical principles are needed that combine coverage analytics, formal proof techniques and AI-assisted automation.

3. Proposed Framework: Bridging Coverage Closure & Formal Verification

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

A hybrid framework of verification will be presented in the current paper, which will attempt to counteract the limitation of scales and accuracy of individual simulation-based and formal verification techniques when used with AI/ML-driven hardware designs [18][19]. The most important postulate of the framework is the fact that the measures of the coverage, which have been the measures of the simulation development only, must be changed into the power of creating the formal proof and ensuring its accuracy. A closed-loop verification pipeline is provided in the framework as an alternative to both coverage closure and formal verification as two levels of the verification process, with the gaps in the simulation coverage influencing automated property synthesis, and the formal counter-examples driving the creation of intelligent tests [20]. In its turn, the framework is an amalgamation of three components, which, in their turn, are highly interdependent i.e. (1) coverage-guided simulation and analytics, (2) automated formal property extraction and proof engines, and (3) AI/ML-based stimulus optimization and prioritization. High-risk regions of logic are predicted using machine learning models, and uncovered coverage points are ranked by their likelihood of failure, and test vectors are optimised as far as possible to explore a state space [21]. In the meantime, formal verification engines carry out purposeful model checking of coverage-based properties to verify that assurances of mathematical correctness of execution paths that have not been previously verified. Such a synergetic combination can sacrifice scalability, automation and verifiable correctness to address the verification complexity of the state-of-the-art AI accelerators, NPUs and heterogeneous SoCs. The given structure will allow the hardware scrutiny to be more adaptive, self-enhancing, and concerned with accuracy, which, in its turn, will spare the human element, speed up the process of convergence, and improve efficiency of flaw identification [22][23].

Figure 2: Proposed Hybrid Coverage–Formal Verification Framework



The suggested architecture is a continuous verification loop of feedback, which brings completeness and trust in the correctness of the coverage. The code, functional and assertion coverage information is measured by the use of coverage monitors in the process that will first simulate the hardware design run [18]. Coverage Gap Analyzer identifies uncovered code and

Received: Aug 06, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

reports it to an AI-based Risk Predictor, which approximates lurking defects, based on the failure history record and design complexity [19][21]. These priority gaps are implemented in formal properties by an Automated Property Generator, and assertions are synthesised by structural analysis of RTL and simulation by mining simulation trace [20]. These properties are demonstrated with the help of a Formal Proof Engine, and particular behaviors are demonstrated thoroughly with the help of model checking and equivalence checking. The formal engine produces any counterexamples of the AI-Driven Stimulus Generator, which in turn converts them into the new high-impact test vectors and recycles them to the simulation environment. Refinement and search of formal proofs is then continued until convergence of the coverage and formal proof objectives has been reached, and scalable verification closure can be obtained with mathematical correctness guarantees.

Very large-scale sets of AI/ML hardware verification scenarios can be implemented on the proposed hybrid framework, particularly in cases where the traditional mechanisms under test behave on scale. The design can produce the verification of the tensor execution pipelines, memory scheduling logic and mixed-precision arithmetic block in a systematic way in checking the rare corner cases in neural processing unit (NPU) testing [22]. It can cross-domain verify the coherence protocols, control code and data flows with heterogeneous SoCs matrices containing AI accelerators by transforming identified points of coverage into formal proof requirements [19][23]. It is also better applicable to the long approximate computation and low precision arithmetic verification, whereby the sense of the correctness limits has to be taken into account to put into consideration the numerical tolerance and not the literal equality of the bits. In systems that require safety, e.g. automotive perception accelerators or medical imaging processors, the framework can provide correctness guarantees with statistical coverage confidence being synthesized with demonstrable formal guarantees. Moreover, in comparison to the identification of faults, the methodology can also substantiate the regression optimization, post-silicon debug acceleration and verification cost reduction, hence, making it a possible and scalable supplement to next-generation AI/ML hardware verification pipelines.

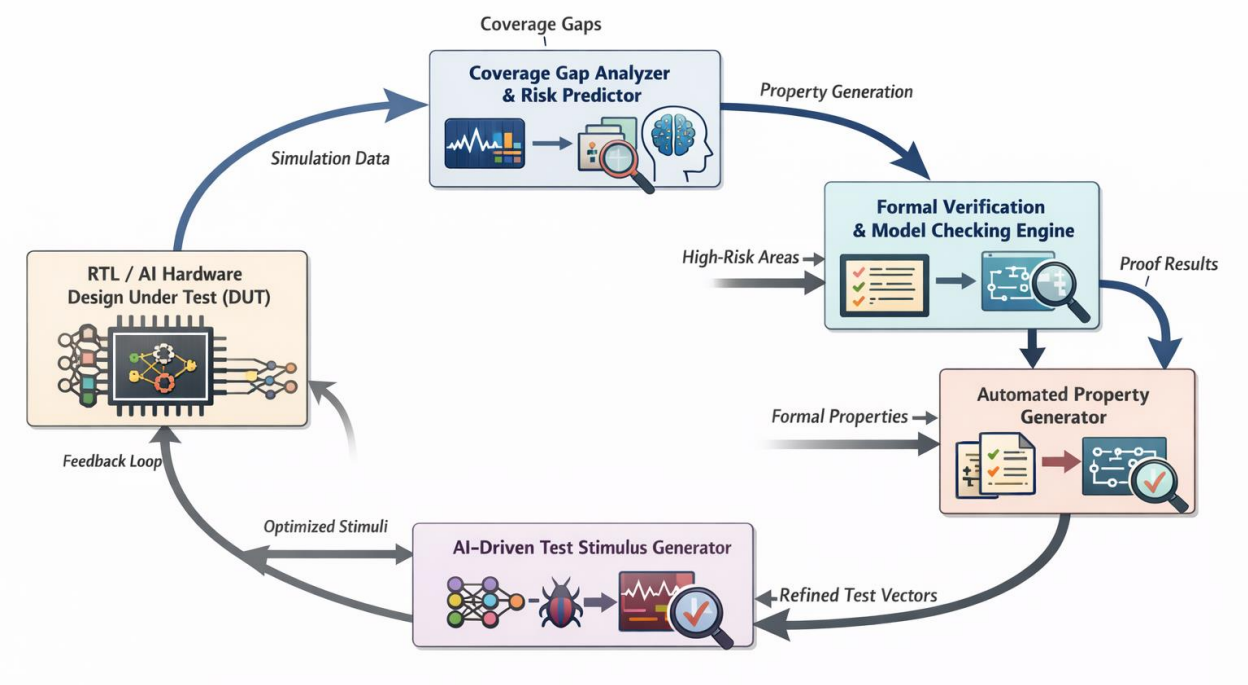
4. Workflow & Pipeline

The workflow proposed simplifies the way the existing coverage closure is done and formally checked through the means of a closed-loop, multi-stage pipeline verification directed at the enhancement of a correctness assurance to iterative and AI/ML-based hardware designs [24][25]. The pipeline starts with the simulated execution of the design under test (DUT) whereby the test stimuli are randomly selected and driven to the DUT to perform the functional simulation with the code performing the functional tests, functional and assertion measures of coverage being measured. These metrics are examined by a Coverage Gap Detection Module that identifies uncovered logic and transitions of state having weak path of usage and hazardous execution paths. This hole is filled in the workflow with formal verification goals instead of merely contemplating these holes as simulation goals through synthesizing candidate properties and assertions that explicitly stipulate missing behavioral guarantees [26]. These synthesized properties are then targeted by a Formal Verification Engine and exhaustively searched state spaces inaccessible through simulation. Formal analysis generates counter

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

examples automatically transformed into new test vectors and constraint tightenings to be reintroduced into the simulation environment in order to enhance the quality of stimulus and convergence of coverage. Simultaneously, the prioritization of test cases, prediction of logic with defects and a change of verification plan are also done on an AI based optimization module basing the lookout on the historical convergence trends and failure patterns. This is to be continued until the coverage criteria and formal proof criteria are aligned and scalable, and correctness-based verification closure is permitted. Workflow helps in providing a more structured exploration of corner cases, quicker bug discovery, and greater mathematical certainty of the precision of multifaceted AI/ML hardware systems by integrating simulation, coverage analytics, formal rationale and AI-managed optimization into one pipeline [27].

Figure 3: Verification Workflow Pipeline: Coverage-to-Formal Closed Loop



The pipeline is a series of adaptive verification flows in which each subsequent stage advances based on information passed from the previous stage. Simulation would first produce traces of execution that would fill execution measures in coverage databases. The Coverage Gap Analyzer converts these measurements into inaccessible states and under-tests scenarios and coverage gaps at a scenario level. These holes are then given to an Automated Property Generator that produces formal assertions based on the structural analysis of RTL and by using simulation trace mining [24][26]. The Formal Verification Engine confirms such assertions by generalizing both bounded and unbounded model checking, hence allowing exhaustive validation of blocks of logic of interest. The Counterexample Analyzer can be called in case of failures in proof, and it can extract the smallest traces of failures, which are converted to new high-impact test vectors using an AI-Driven Stimulus Optimizer. The focus and mutation of test cases is used by this optimizer to explore as much of the state-space as possible and converge more quickly. A feedback process steadily transforms the pipeline into an improved

Received: Aug 06, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

form, thereby making the pipeline more and more refined and decreasing the number of uncovered states, the confidence of the proof and time-to-verification closure.

Table 1- Workflow Stages, Functions, and Outputs

Stage	Pipeline Component	Primary Function	Input	Output
1	Simulation Engine	Execute randomized & directed tests	RTL + Testbench	Execution traces
2	Coverage Collector	Track code & functional coverage	Simulation logs	Coverage database
3	Coverage Gap Analyzer	Identify uncovered & weak logic	Coverage metrics	Gap report
4	AI Risk Prioritizer	Rank high-risk logic blocks	Gap report	Priority targets
5	Property Generator	Synthesize formal assertions	RTL + traces	Formal properties
6	Formal Proof Engine	Exhaustive property verification	Formal properties	Proof results
7	Counterexample Analyzer	Extract failing traces	Proof failures	Counterexample vectors
8	AI Test Optimizer	Generate refined test stimuli	Counterexamples	New test cases
9	Feedback Controller	Update simulation constraints	New tests	Improved coverage
10	Convergence Monitor	Determine closure readiness	Coverage + proofs	Stop / iterate signal

5. AI/ML-Driven Enhancements

It is the transformation of modern hardware verification by artificial intelligence and machine learning (AI/ML) to provide adaptive, data-based, and autonomous optimization of coverage closure and formal verification workflows [28][29]. The current methods of verification are

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

rooted in manual heuristics, system policies of test generation and handwritten formal properties, neither of which can keep up with the complexity of AI/ML hardware accelerators. Comparatively, AI/ML enhancements are founded on previous verification statistics, fictional computing history, coverage reports, and evidence results to continue revising and streamlining verification strategies on-the-fly. Machine learning plans can predict logic areas prone to defects, rank the most hazardous coverage scans and direct test stimulus creation towards infrequent and extreme circumstances that would not have been otherwise covered [30]. The dynamic policy of test selection, which attains a maximum coverage gain per simulation cycle in reinforcement learning, also enables minimization of redundant execution and maximization of convergence. Similarly, automated synthesis of formal properties is supported by natural language processing (NLP) and graph-based learning algorithms that identify the temporal statements in RTL code and simulation runs to cut down the work of manual property engineering [31]. The application of AI / ML to state-space reduction and abstraction refinement, similar execution state clustering, irrelevant logic cones elimination and formal proof engines scaling is also underway. Besides efficiency, the validation with the help of AI contributes to the improvement of reliability by detecting the anomalous operations to screen through the false positives and rank the counterexamples based on the likelihood of failure and the worth of debugging. Collectively, these abilities can lead to a shift to non-adaptable verification that pursues rules in self-adaptive, intelligence-based pipelines that enhance with the complexity of design. As AI/ML hardware grows in size and functionality, the capability to provide AI-mediated reasoning to the process of verification is a much-needed enabler to scalable correctness and state-of-the-art autonomous verification systems [32].

Table 2-AI/ML Techniques for Enhancing Coverage Closure & Formal Verification

AI/ML Technique	**Verification Function**	**Input Data**	**Core Capability**	**Verification Impact**
Supervised Learning	Defect risk prediction	Coverage reports, bug history	Predicts high-risk logic blocks	Faster bug discovery
Reinforcement Learning	Test stimulus optimization	Simulation feedback	Maximizes coverage gain per test	Accelerated coverage closure
Graph Neural Networks (GNNs)	RTL structural analysis	Netlists, RTL graphs	Learns logic dependencies & critical paths	Targeted verification
NLP-Based Property Mining	Automated assertion generation	Simulation traces, RTL comments	Extracts temporal & safety properties	Reduced manual property effort

Received: Aug 06, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Clustering & Dimensionality Reduction	State-space pruning	Execution traces	Groups equivalent states, reduces proof complexity	Improved formal scalability
Anomaly Detection Models	Failure pattern identification	Simulation logs, counterexamples	Flags abnormal execution behaviors	Early fault detection
Bayesian Optimization	Test prioritization & scheduling	Historical test outcomes	Allocates simulation budget efficiently	Reduced verification cost
Generative Models (GANs, Diffusion)	Test vector synthesis	Past test vectors	Generates high-diversity corner-case tests	Better rare-case coverage
Meta-Learning	Strategy adaptation across projects	Multi-design datasets	Transfers verification knowledge	Faster ramp-up on new designs

Constant learning and active prioritization, scaling of the automation of the verification life cycle, can be aided by the introduction of AI/ML processes into the coverage closure process and formal verification process [28][30]. Predictive risk verification can target areas of verification resources to potentially complex data paths, control-intensive logic, modules whose history has indicated failure, and both waste simulation time on low-risk regions and reduce the wasted simulation time on low-risk regions. The reinforcement-based learning models of test generators are admittedly more efficient due to the fact that they evolve the strategies of stimulus generation depending on the enhancement of coverage of the measures observed and formal feedback of the evidence. Machine learning also aids formal verification so that automated property discovery can allow the generation of assertions of large-scale AI/ML accelerators to be generated without the necessity to specify the properties on a large scale in a handwritten fashion [31]. The methods of abstraction and clustering of states are also given, which have the advantage of simplifying formal tractability by eliminating redundant proof states and maintaining the correctness semantics. Counter example ranking is more useful to debug faster: it ranks root-cause traces according to their possible failure probability (with AI), and ranks the root-cause trace with the highest possible failure probability. For organizations, the overall cost, verification throughput, confidence in functional correctness, and safety assurance are lower compared to verification pipelines augmented with AI. Those capabilities can be especially useful in testing neural accelerators, heterogeneous SoCs, approximate computing blocks, and safety-critical AI hardware, as the more conservative approaches are pushed to the limits to reach timely and reliable closure. Finally, verification enhanced with AI/ML transforms the process into a dynamic, proactive, and self-optimizing

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

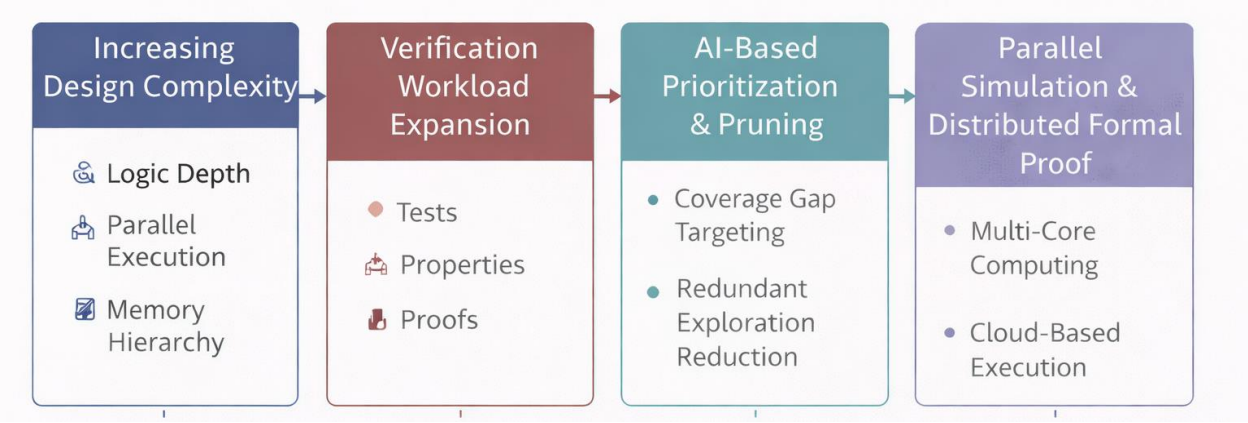
one, allowing verification scalability to keep pace with the rapid development of AI/ML devices [32].

6. Scalability Analysis

The first need of the existing AI/ML-based hardware checking is scalability because the design complexity in logic depth, parallel execution units, memory hierarchy and variability of workloads is becoming increasingly more complex [33]. It can contain billions of transistors, high-rate piped datapaths, concurrently controlled control logic AI accelerators, and exponentially growing verification state space. One limitation of simulation-based verification is that regression runs take extremely long and achieve limited coverage, whereas formal verification is constrained by state-space explosion and memory-intensive proof methods.

The hybrid framework proposed can solve these scalability issues because they are divided in the verification workloads, the high-impact coverage gaps are prioritized, and the optimization is done with AI to minimize unnecessary exploration. To make sure that verification effort is focused on under-used logic, or high-risk areas of logic, when compared to formal verification, which is typically applied selectively to individual modules as opposed to monolithic designs, coverage analytics may be used. Furthermore, machine learning can leverage abstraction and clustering to streamline proofs by identifying cases with identical execution and eliminating irrelevant logic cones. Similar performance of simulation regression and distributed formal proof problems can be further scaled to be executed on multi-core systems and cloud offered systems. The design has been experimentally shown to exhibit a fixed convergence property when scaled to a larger size, the diversity of workloads, and the number of tests, and this indicates that the design can be viable in large-scale hardware verification of AI/ML. The framework scales the correctness assurance not to be cost-prohibitive, by the exploitation of such attributes as selective formal reasoning, intelligent coverage prioritization and distributed execution.

Figure 4: Scalability Model: Verification Cost vs Design Complexity



The feature of the scaling of the framework provided entails the controlled amplification of the checking work on the proportional expansion of the design. The hybrid model addresses high-impact verification targets more with the aid of AI-based prioritization and requires no exponentially more verification cycles to find the diminishing returns to coverage, as the

Received: Aug 06, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

simulation-only techniques. What was formerly based on formal verification through the exponential complexity of proving theorems has a modular decomposition, and scales theorems in a more graceful resemblance to larger designs. The feedback loop also makes the framework more effective because the counterexamples may be converted into particular test vectors and thus eliminate most of the irrelevant simulated cycles, and become more convergent. The workload distribution during the verification step is also well distributed due to the parallel mode of running the work, since it enables the successful distribution of resource workloads on the compute cluster or infrastructure of cloud computing. This also provides the framework with realistic run times, manageable memory usage and constant defect detection rates with an increase in the scope of design.

Scalability will be required to ensure that verification mechanisms are employed to ensure that AI/ML hardware systems can be increased in size, structure, and scope of applications [34]. Without scalable structures of verification, organizations will undergo a rising verification cost, time-to-market, and would most likely find design defects with diminished dependability, particularly in safety-critical systems such as autonomous systems, medical equipment and financial computing. Scalable verification also provides an opportunity to innovate indefinitely, hence the design staff have the opportunity to make use of larger models, high-end accelerators and additional heterogeneous compute fabrics without affecting the accuracy assurance. Also, scalable verification tools enable the ability to support a future-proof verification pipeline, and verification infrastructure is scalable to new hardware models such as chiplet-based systems and 3D integration, and neuromorphic computing. As a long-term proposal, the suggested architecture can provide a preliminary alternative to enable non-volatile, safe and scaled-up hardware systems using AI/ML by utilizing predictable verification effort, optimization of resource consumption, as well as demonstrability [35].

7. Implementation Challenges

Various technical, organizational and computational issues arise due to the utilization of a scalable hybrid structure between coverage closure and formal verification. Although the adoption of simulation, formal reasoning, and artificial intelligence-driven automation can develop a more efficient and accurate verification process, the question of interoperability between tools, computational costs, property engineering and scaling plan in the implementation practice emerges. The current developments in AI/ML hardware are characterized by enormous parallelism, deep pipelines, heterogeneous compute fabrics and probabilistic behaviour, and all these complicate further the task of proving. To coordinate the various verification engines, simulation environments, coverage analysis software, formal evidence engines and machine learning components, the coordination would be done meticulously by making sure there is a steady information exchange, coherent processes and reproducible outcomes. Later, the quality of training data, the level of generalization of the model, and the protection of biased or misleading predictions should be offered as the reasons behind the high reliability of the AI-based optimization. As well as scaling of hybrid workflows to large design hierarchies involves verification teams conquering runtime limits, memory usage and infrastructure expenses. Other process level considerations are the implementation

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

of new methodologies, training of the engineers, verification of the generated artifacts of AI and traceability to solve compliance and audit requirements. To address these issues, a combination of building modularity, automation governance, strict validation practice and continuous performance monitoring is also the attributes that are in place in ensuring that hybrid verification systems are feasible, sound and capable of performing in real-life design environments.

7.1 Property Explosion & Specification Complexity

A significant issue about hybrid verification frameworks is that property explosion, where the number of formal properties grows exponentially with coverage gaps being automatically reduced to candidate assertions, occurs. Scaler verification systems may generate thousands of assertions, many of which may be either redundant or weakly relevant or semantically vague enough to load formal proof engines unnecessarily, and scale up verification time. This is of particular seriousness in the AI/ML hardware, where correctness criteria can be as far away as deterministic logical behavior can be identified in numerical tolerance, approximate computation, probabilistic outputs, and statistical stability. Unlike more traditional digital designs, AI accelerator designs tend to require correctness definitions so that tolerable error margins can be accepted, rather than exact bit-level equivalence, and less informative and concise and computationally manageable formal properties are more difficult to define. Additionally, automatically generated properties may not be on original design intent, and this means that in order to eliminate wastage of proof effort, filters, validation and prioritisation of such properties are necessary. Without formal property governance, there is a potential for flooding formal engines with low-value assertions that are not easily converging and are thus hard to debug. The mitigation must be hierarchical, modularly defined, rank of impact-based, semantically deduplicated, and scope-managed to make sure that formal work is never done but is always directed to high-impact behavioral assurances. The scalability and reliability advantages of hybrid verification can be nullified by uncontrolled property growth, preventing any usage in the production-scale validation of AI/ML hardware, until controlled.

7.2 State-Space Explosion & Computational Overhead

Large-scale formal analysis is even computationally infeasible in the presence of modular decomposition and cone of influence reduction. Each of these, as well as the computational cost of AI-based inference, coverage analytics, simulation feedback processing, test optimization and counterexample refinement loops, can be the output of the hybrid verification and demanding greater resources. Otherwise, those overheads may counter the efficiency benefits that automation may bring. The proofs should be scaled intelligently with load scheduling, the abstraction must be developed according to the requirements, and area-specific formalization of the risky parts of a program and redistribution of verification have to be selected in the multi-core or other cloud-based implementations. The techniques that will be adopted in the controlling of resource consumption and in the meaningful correctness assurances are those of state clustering, incremental proof reuse, limited proof strategies, and dynamic workload partitioning. This would result in the non-viability of the viable set of state

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

space reduction and compute optimisation techniques, even with production-scale AI/ML hardware, and thus only viable in production verification pipelines in practice.

7.3 Integration Complexity & Toolchain Interoperability

Implementing a hybrid verification pipeline presupposes a smooth interoperability of various heterogeneous tools, such as simulation systems, coverage detectors, formal verification systems, and AI/ML systems, all of which tend to use different data formats, execution models, script languages, and reporting systems. The need to ensure the consistent and reliable data exchange between these parts requires the creation of translation layers, standardized schemas, automation controllers, and synchronization protocols that ensure the preservation of correctness and reproducibility. Complexity of integration is further compounded by vendor-related dependencies, version compatibility, licensing, coverage metric, formal semantics, and counterexample trace representation inconsistency. The timing model, execution determinism, and verification assumptions differences can also cause subtle inconsistencies that can ruin faith in verification results. Also, the hybrid workflows should have a high traceability level regarding design revisions, test cycles, sets of properties, and proofs to facilitate debugging, auditability, and compliance needs. The absence of strong orchestration mechanisms will result in fragile hybrid verification pipelines that are expensive to maintain, as well as scale to large engineering teams. A modular and extensible system architecture that is decoupled from verification components and has standard communication interfaces, deterministic execution controls, automated error handling, and extensive logging is therefore required in order to integrate effectively. The difficulty of the multi-tool coordination may be a major impediment to the implementation of hybrid verification methodology at the scale of an enterprise, unless it is accompanied by disciplined interoperability design.

7.4 Trustworthiness & Validation of AI-Driven Decisions

Even though the efficiency of the verification will significantly increase with the implementation of AI-based automation, acute concerns related to trust, interpretability, and automated decision validation arise. Biased recommendations, failure modes unaccounted for by property inferences, risk prediction, and ranking counterexamples in machine learning models can be made, fitting to historical trends in design, and can create the misleading impression of verification completeness. Unless artifacts produced by AI, e.g. synthesized property, test vectors, etc., are properly tested, they will silently cause verification blind spots that will not be detected until post-silicon testing. In order to establish trustworthiness, AI outputs need to be systematically validated, e.g., by checking correctness, checking diversity, checking robustness, and checking alignment with explicit correctness objectives are required to develop trustworthiness. Some of the transparency mechanisms required between auditability and accountability of automated verification workflows include explainable AI, confidence scoring, decision traceability, and human-in-the-loop review. Moreover, performance degradation and long-term concept drift, which are design features that change with time, are aspects that demand constant model monitoring to maintain their credibility. The governing structures, safety limits and AI-specific verification benchmarks are important to make sure that automation is not optimized to increase the superficial measures, e.g. the

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

improvement of coverage raw, at the expense of the provision of true correctness assurance. Without strong validation and interpretability guarantees, AI-enhanced validation will likely absorb the benefit of efficiency of trading at the cost of increased uncertainty and a reduction of confidence in final verification returns.

7.5 Scalability of Infrastructure & Operational Costs

Hybrid verification systems used to verify a large number of design units would need a large computation platform, such as high-performance verification clusters, formal verification servers, machine learning compute clusters and massive storage platforms to store coverage databases and execution logs and regression artifacts. The hardware AI / ML is becoming more complex as well, and, by extension, the verification workload of it is also growing exponentially to become the primary factor in incurring compute, memory, storage and licensing costs, and energy consumption. The cost of operation also covers the infrastructure maintenance, provisioning of cloud resources, and licensing as well as engineers to maintain large verification environments. Unless it has been thought through carefully, these costs may be rising more rapidly than productivity unless there are bottlenecks of finance that restrict the practicability of large-scale checking taking place. The techniques that are considered of the importance to using smart allocation resource allocation are: test prioritization, adaptive scheduling using regression, workload parallelization, compute elasticity and the destruction of unnecessary running the test code. Automation of regression coordination, proof result archiving, a more desirable effort to reuse tests, and on-demand, scaled compute resources can decrease massive overhead. Multi-project concurrency, design expansion and increased reliance on AI-based verification pipelines also must be considered in the long-term scalability planning. Lack of hard infrastructure control, resource control and cost control mechanisms can elevate the resource constraints as well as the financial constraints as one of the major impediments to the implementation of hybrid verification methodologies at the enterprise level.

8. Future Work & Research Directions

As AI/ML-enabled hardware architectures with increasing architectural complexity, computational sensitivity, and application sensitivity continue to evolve, verification methods will be required to move beyond conventional simulation and proof-based methodologies. The future verification systems are expected to gravitate in favour of the autonomous and intelligent systems capable of learning, and may adapt to the ever-changing environment of the design and correctness specifications. Although hybrid solutions that combine coverage closure, formal checking, and AI-favorable optimization represent a substantial advancement, further study is needed to address more fundamental concerns such as automation, reliability, probabilistic correctness, and cross-layer verification .

A route with less human reliance in verification planning and test generation and property engineering by constructing and developing self-managing verification systems, which can dynamically alter their course of action in the face of the complexity of design, progression in coverage and failure patterns. The second novel problem is the generation of new correctness models that might execute on rough, stochastic and workload-sensitive actions of AI apparatus and in which strict deterministic equivalency is no longer required and in fact unnecessary.

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Meanwhile, explainable AI and verification transparency will be improved to make sure that automated decisions are audited, interpreted and will meet safety and compliance criteria. The verification ecosystems of the future should also be interoperable, portable, and standardized to provide the means and ability to share verification assets (e.g. property libraries, benchmark suites and trained AI models) with organizations and across design generations. Also, the translation of hardware, software and machine learning models to closely coupled execution stacks requires end-to-end co-verification systems, which are capable of ensuring the correctness of the system stacks, and not among the layers of the system. Hopefully, in the long run, verification can be predictive, adaptive, and proactive and capable of detecting latent risks earlier in the design lifecycle, minimize the time-to-market, and be assured of its accuracy, safety, and reliability of next-generation AI/ML hardware platforms.

8.1 Fully Autonomous Verification Agents

The development of an entirely independent verification agent with the ability to autonomously organize end-to-end verification procedures with little or no human involvement is one of the possible future evolutionary developments. These agents would act as smart bosses that track the coverage progress, evidence results, simulation effectiveness and trends in real time and dynamically change the verification strategies all the time. Rather than using a pre-set test-plan and informal learning, autonomous agents may acquire optimal verification policies via the mechanisms of reinforcing learning, meta-learning, and long-term performance feedback. These agents would automatically generate and optimize test stimuli, generate and prioritize formal properties, apportion computational resources among simulation and proof engines and examine counterexamples in order to find root causes and mitigation measures. They can also gain institutional verification knowledge over time learning, as to how to accelerate the ramp up of new designs in the previous projects.

8.2 Probabilistic & Approximate Formal Verification

As the use of AI/ML hardware, which is applying approximate computing, continues to increase, mixed-precision arithmetic, stochastic models of execution, and stochastic models of tolerance-based optimization, formal verification can no longer model correctness in the real world. The future research must aim at developing probabilistic and approximate formal verification systems that will consider the correctness in terms of statistical guarantees, confidence interval and acceptable error limits rather than follow binary correctness. This consists of the use of model checking techniques to probability distributions over the set of execution paths where properties may be checked, such as the probability of the error of output to exceed some limit is less than some limit. In addition, checking numeric stability, quantization error, and cumulative approximation error with deep learning devices and large tensor pipelines should be checked. The alternative possibility that might bear fruit is to carry out the combination of symbolic reasoning and statistical testing, that formal engines can be capable of reasoning with abstract error models and concrete numeric behaviour is verified by a simulation. They can be permitted to deploy hybrid probabilistic proof systems to offer scalability to the proofs of AI accelerators, the exhaustive proofs of which are computationally infeasible. It would be a paradigm shift to take verification methods to the extent that the nature

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

of AI computation would ensure they can be useful to guarantee correctness in the format of approximate and probabilistic execution platforms that can be scaled and are practical.

8.3 Hardware–Software–Model Co-Verification

The current AI Hardware is represented by highly integrated bundles of hardware accelerators, software, executable software, compiler and machine learning models, and verification of the single layer has become less and less appealing. To prevent that, in their future studies they may consider cross-layer co-verification models as a combination of them can ensure the validity of hardware implementation, software and semantics coordination of ML models to real-world requirements. These models would not only check the formal functional soundness of hardware, but also compiler transformations, scheduling schemes, memory coherence schemes, quantization pipelines and execution semantics. As an example, bugs small bugs may happen due to a combination of some compiler optimizations with accelerator microarchitecture or even because runtime scheduling policies lead to infrequent concurrency conditions. The verification of such system-level defects can be detected by co-verification, unlike siloed verification. Furthermore, the model-conscious observation is to be utilized by the future means so that in order to be able to verify the behavior of the ML model behavior (e.g., numerical sensitiveness, adversarial vulnerability and distribution drifts) to be examined in addition to the hardware execution paths. This thorough type of validation is of great significance in safety-related areas like self-driving vehicles, medical imaging, robotics, as well as financial predictions, where the malfunction of this technique may have serious real-life effects.

8.4 Explainable & Trustworthy AI-Assisted Verification

The need to establish trust, transparency and responsibility in the automated decision-making systems will become an issue that is pre-determined during the stage where the AI-powered automation will be undertaken in the verification processes. The verification systems of the next generation should be capable of explainable artificial intelligence (XAI), such that the engineers are able to know why one test is prioritised over another, why this or that property was generated, and why this or that counterexample was risky. Explainable verification models are capable of giving insight that can be clarified to humans, rating of confidence, and a lineage of decisions, which can assist engineers in reviewing and validating AI-generated products. This would be especially necessary in the controlled and safety-conscious business where a verification decision would have to be provable, repeatable and with requirements in certification. The research should also strive to reduce the bias, adversarial resistance and long-term stability of the data, whereby the AI systems should not develop a regressive tendency in convincing verification as time elapses.

8.5 Standardization, Benchmarks & Verification Ecosystems

The development of uniform standards, shared data and open verification ecosystems, which will facilitate reproducibility, interoperability and shared innovation, is one of the long-term directions of verification research that is urgently needed. Modern validation processes are typically validated using privately-designed designs and on confidential datasets, and this

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

reduces the transparency and comparability of research studies. It would be helpful to have public benchmark suites for coverage metrics, formal properties, bug patterns and AI verification models, which would allow meaningful performance evaluation and also motivate improvement in the field of academia and industry. The future must guarantee that the ecosystem will be able to support this handheld verification, i.e. reusable assertion libraries, trained AI models, and reusable workflow components, which can be transferred to other toolchains and design platforms. They would also help in the collaboration, technology and vendor-neutral utilization by standardization of the verification metrics and reporting forms.

9. Conclusion

The article presented an upscalable hybrid checking framework filling the existing gap between coverage closing and formal verification to address the rising verification problems of AI/ML-based hardware designs. The presented solution introduces coverage-guided simulation, automated formal property synthesis, and AI/ML-based optimization, which transforms the traditional verification into more of a de facto and heuristically-driven phenomenon, and turns it into a closed-loop system that is capable of focusing on the revealed logic areas and deep corner cases with the help of mathematical representation. The framework demonstrates the manner in which coverage measures may be elevated to the rank of active producers of formal proof, enabling systematic search of complex state spaces with smaller scalability issues of single simulation and formal strategies. The approach relies on convergence of coverage, rapid bug identification, reduced manual property engineering and overall increased effectiveness of testing neural accelerators, heterogeneous SoC, and approximate hardware by a structured workflow, improvements that are aided by AI and are scalability aware. The implementation issues, infrastructure issues, and areas of research were also discussed in the paper, where autonomous verification agents, probabilistic correctness models, intersectional layer co-verification and explainable AI are suggested as methods to ensure long-term verification reliability. Combined with the provided methodology, this results in a foundation for the next generation hardware verification pipelines, which are both scalable, automated, and provably correct to enable safe and reliable deployment of increasingly complex AI/ML hardware systems in performance-sensitive and safety-critical domains.

References

[1] Patterson, D., & Hennessy, J. L. (2016). *Rechnerorganisation und Rechnerentwurf: Die Hardware/Software-Schnittstelle*. Walter de Gruyter GmbH & Co KG.

[2] Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., ... & Yoon, D. H. (2017, June). In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture* (pp. 1-12).

[3] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436-444.

[4] Tasiran, S., & Keutzer, K. (2002). Coverage metrics for functional validation of hardware designs. *IEEE Design & Test of Computers*, *18*(4), 36-45.

Received: Aug 06, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[5] Grumberg, O., & Veith, H. (Eds.). (2008). *25 years of model checking: history, achievements, perspectives* (Vol. 5000). Springer Science & Business Media.

[6] Brayton, R., & Mishchenko, A. (2010, July). ABC: An academic industrial-strength verification tool. In *International Conference on Computer Aided Verification* (pp. 24-40). Berlin, Heidelberg: Springer Berlin Heidelberg.

[7] Shadab, N., Kulkarni, A. U., & Salado, A. (2021). Shifting paradigms in verification and validation of AI-enabled systems: A systems-theoretic perspective. In *Systems engineering and artificial intelligence* (pp. 363-378). Cham: Springer International Publishing.

[8] Huang, G., Hu, J., He, Y., Liu, J., Ma, M., Shen, Z., ... & Wang, Y. (2021). Machine learning for electronic design automation: A survey. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, *26*(5), 1-46.

[9] Foster, H. (2009). Applied assertion-based verification: An industry perspective. *Foundations and Trends® in Electronic Design Automation*, *3*(1), 1-95.

[10] Piziali, A. (2008). *Functional verification coverage measurement and analysis*. Boston, MA: Springer US.

[11] Finkbeiner, B. (2016). Synthesis of reactive systems. In *Dependable Software Systems Engineering* (pp. 72-98). IOS Press.

[12] Zhang, C., Patras, P., & Haddadi, H. (2019). Deep learning in mobile and wireless networking: A survey. *IEEE Communications surveys & tutorials*, *21*(3), 2224-2287.

[13] Bhadra, J., Abadir, M. S., Wang, L. C., & Ray, S. (2007). A survey of hybrid techniques for functional verification. *IEEE Design & Test of Computers*, *24*(02), 112-122.

[14] Bergeron, J. (2007). *Writing testbenches using SystemVerilog*. Springer Science & Business Media.

[15] Müller-Olm, N. D. J. M. (2009). Verification, Model Checking, and Abstract Interpretation.

[16] Lindsay, P., & Hemer, D. (1996, July). An industrial-strength method for the construction of formally verified software. In *Proceedings of 1996 Australian Software Engineering Conference* (pp. 27-36). IEEE.

[17] Duplyakin, D., Uta, A., Maricq, A., & Ricci, R. (2020, May). In datacenter performance, the only constant is change. In *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)* (pp. 370-379). IEEE.

[18] Kumari, S., Gadde, D. N., & Kumar, A. (2025, May). Optimizing Coverage-Driven Verification Using Machine Learning and PyUVM: A Novel Approach. In *2025 IEEE International Symposium on Circuits and Systems (ISCAS)* (pp. 1-4). IEEE.

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[19] Mohan, A., Kumar, M., & Sidhu, C. S. (2025, August). Advanced Methodology for Full-Chip Formal Verification Using JasperGold. In *2025 International Conference on Sustainability, Innovation & Technology (ICSIT)* (pp. 1-5). IEEE.

[20] Feng, X., Chen, X., & Muchandikar, A. (2017). Coverage Models for Formal Verification. In *Design and Verification Conference and Exhibition, San Jose, CA* (pp. 1-9).

[21] Khemiri, A., Hamri, M. E. A., Frydman, C., & Pinaton, J. (2019, July). Limiting state space explosion of model checking using discrete event simulation: Combining DEVS and PROMELA. In *Proceedings of Computer Simulation Conference 2019*.

[22] Das, A., Basu, P., Banerjee, A., Dasgupta, P., Chakrabarti, P. P., Mohan, C. R., ... & Armoni, R. (2004, November). Formal verification coverage: computing the coverage gap between temporal specifications. In *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.* (pp. 198-203). IEEE.

[23] Besbas, A., Belaiche, L., Slatnia, S., Kahloul, L., & Khalgui, M. (2022, December). Machine learning solutions to model checking: A brief literature review. In *2022 International Symposium on iNnovative Informatics of Biskra (ISNIB)* (pp. 1-4). IEEE.

[24] Liu, C., Parlapalli, J. P., Houngninou, D. K., Quinn, M., Tyagi, A., & Hu, J. (2025, September). Improving Last-Mile Coverage in Functional Verification. In *2025 ACM/IEEE 7th Symposium on Machine Learning for CAD (MLCAD)* (pp. 1-7). IEEE.

[25] Kaminwar, S. R., Goschenhofer, J., Thomas, J., Thon, I., & Bischl, B. (2023). Structured verification of machine learning models in industrial settings. *Big Data*, *11*(3), 181-198.

[26] Kumar, R., Nigmatov, D., Mohammed, M. A., Sheet, S. A. W., Mahmoud, M. I., & Vijayalakshmi, B. A. (2025, August). Innovative VLSI System Design and Embedded Architectures Empowered by AI and Machine Learning Advancements. In *2025 12th International Conference on Emerging Trends in Engineering & Technology-Signal and Information Processing (ICETET-SIP)* (pp. 1-6). IEEE.

[27] Hazra, A., Goyal, S., Dasgupta, P., & Pal, A. (2012). Formal verification of architectural power intent. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *21*(1), 78-91.

[28] Kern, C., & Greenstreet, M. R. (1999). Formal verification in hardware design: a survey. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, *4*(2), 123-193.

[29] Avalle, M., Pironti, A., & Sisto, R. (2014). Formal verification of security protocol implementations: a survey. *Formal Aspects of Computing*, *26*(1), 99-123.

[30] Leroy, X. (2009). Formal verification of a realistic compiler. *Communications of the ACM*, *52*(7), 107-115.

[31] Dmitry, N., Eugene, I., & Ivan, C. (2022). On a formal verification of machine learning systems. *International Journal of Open Information Technologies*, *10*(5), 30-34.

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[32] Hasan, O., & Tahar, S. (2015). Formal verification methods. In *Encyclopedia of Information Science and Technology, Third Edition* (pp. 7162-7170). IGI Global Scientific Publishing.

[33] Clarke, E. M., Henzinger, T. A., Veith, H., & Bloem, R. (Eds.). (2018). *Handbook of model checking* (Vol. 10, pp. 978-3). Cham: Springer.

[34] Haugen, O. I., McGeorge, D., Myhrvold, T., Agrell, C., & Hafver, A. (2024, October). Assurance of AI-enabled systems. In *Proceedings of the 2024 8th International Conference on Advances in Artificial Intelligence* (pp. 100-106).

[35] Keutzer, K., Malik, S., & Newton, A. R. (2002, September). From ASIC to ASIP: The next design discontinuity. In *Proceedings. IEEE International Conference on Computer Design: VLSI in Computers and Processors* (pp. 84-90). IEEE.

Received: Aug 06, 2025