



UVM Methodology for Industry-Specific Applications Using AI in Hardware Verification

Kaushik Velapa Reddy

Microsoft Corporation, United States of America

ABSTRACT: These increasing sophistications of modern System-on-Chip (SoC), semiconductor, automotive, telecommunications and embedded systems have resulted in severe hardware verification issues, especially in achieving full coverage, reducing the connotation timeline, and uncovering difficult-to-find system defects. Universal Verification Methodology (UVM) provides a setting where standardized and re-usable functional verification takes place, however standard UVM-based verification often heavily depends on manually crafted test-cases and minimal helpers in order to set-up constrained-random stimulus. This work proposes an AI-powered system of UVM method to confirm the hardware specific to industries, which includes the implementation of artificial intelligence and machine learning technologies into the key phases of the UVM verification life cycle. The proposed framework also consists of AI-assisted code coverage, AI assisted test generation, failure classification, verification prioritization and adaptive optimization of stimulus in a modular UVM. Configurable verification components are used to represent industry-specific requirements, and allow the framework to be used in automotive, communication, consumer electronics, industrial, and safety-critical hardware applications. The framework is a continual process of analyzing results of simulations and coverage data to detect verification gaps and dynamically tune the further test cases. This strategy sets up a dynamic verification framework that integrates the reusability and standardization of UVM with the analytical powers of AI. The suggested methodology provides a systematic foundation of improving the performance of the verification, convergence of coverage and fault detection and scaling by heterogeneous hardware verification environments.

KEYWORDS: UVM, Artificial Intelligence, Hardware Verification, Machine Learning, SoC Verification, Adaptive Test Generation

I. INTRODUCTION

The recent history of the semiconductor technology has led to the development of very complex digital systems with millions of transistors and at the advanced systems, billions of transistors. System-on-Chip (SoC) architecture integrates one chip with processors, memory controllers, communication interfaces, accelerators, security modules, sensors and application-specific intellectual property (IP) blocks. This growing deepening has greatly widened the functional area that has to be confirmed prior to hardware implementation. Hardware verification has therefore been deemed one of the most expensive in terms of time and resources during the lifecycle of the digital design. Poor verification can lead system-level, interface, timing, power, or configuration bugs to be pushed forward to later development process stages, where they can be much more expensive to fix.

Traditional hardware verification techniques are inclined to hand-written test cases, guided testing, constrained-random stimulus testing, testing, assertions, functional scope, and waveform based testing. Although these methods remain important, an increasing complexity of hardware designs effectively renders the exhaustive verification of a hardware design to be virtually difficult. A verification space can be large and have lots of cornercases which would not be easily detected using manually done scenarios. This has led to a shift towards more automation, reusability, coverage based testing, and managed verification environment management in verification methodologies. Mechanized test-check techniques have demonstrated to reduce the effort invested in developing tests manually and to boost the productivity of folding the test-coverage [1].

Now, Universal Verification Methodology (UVM) has become an influential published standardized methodology of the design of reusable scalable verification environments. UVM is a formal architecture that contains such elements as sequences, sequencers, drivers, monitors, agents, scoreboards, environment and test classes. These components can enable verification engineers to separate the generation of stimuli, communication at the transaction level, coverage checking and collection. The normalized architecture, design, and verification project are easily reused between different designs. With a UVM-based smart functional verification platform demonstrated how UVM might be



leveraged to offer systematic and reusable functional verification, in addition to indicating future automation opportunities [2].

The traditional UVM environments still involve a significant amount of work on its part with such advantages. Mostly, the verification engineers impose constraints, compile the sequences, select the test scenarios, make interpretations of the coverage reports, and classify failures and determine the next test to be executed. The size of the potentially stimulus combinations can increase exponentially by the complexity of the design. It is not enough to run randomly generated tests to make sure that significant functional states are searched well. Similarly, it is not necessarily true that when a numerical coverage target has been reached then all critical behaviors have been satisfactorily exercised. Hence, this necessitates intelligent methods to process verification information and steer the verification process either in areas that have not been explored or are risky.

UVM automation has already been tapped in as a means of addressing some of these challenges. Register-access and automated coverage tools have demonstrated that one can extend UVM environments to environments not only involving manually-controlled verification processes, but also involving traditional ones [3]. These methods can contribute to better access to verification data and lessen the recurrent processes. The automatic generation of test-intents has also shown that they can be used to generate verification scenarios based on design information and assist the creation of verification much more effectively [1]. This shows that UVM has the potential to offer a beneficial infrastructural framework in achieving greater automation.

The other prominent problem is that hardware products are growing to be industry specific. Automobile control unit languages may vary greatly in verification levels compared to a telecommunications processor, industrial controller, consumer-electronics device, medical system or security-oriented SoC. Some examples are that automotive hardware may require a significant amount of validation of safety related functionality and faults and that communication hardware may be concerned with protocol compliance, throughput, latency and interoperability. Consumer devices could be needs-optimized to focus on functional diversity and power efficiency whereas industrial systems could be needs-optimized to dependable operation under some operating conditions. In turn, before it is possible to make the generic verification methodology applicable to the needs of a specific application domain, it might need a substantial amount of customization.

The idea of power-aware verification also underscores the importance of application-specific depiction of verification strategies. Contemporary SoCs often have several power domains, power-management states and dynamic operating conditions. An early check on power-management specifications also would help in detecting violations ahead of time before propagating to the subsequent stages in design [4]. This explains that verification must continuously increase in the degree of multi-dimensionality of its character both as regards the behaviour of the system and not just in the regard of the correctness of the functional output. This should include a scalable methodology so that the domain-specific verification objectives and constraints can not have to be reconfigured to use the entire verification infrastructure.

Machine learning (ML) and, more generally, artificial intelligence (AI) can provide an opportunity to address a number of the shortcomings of the traditional verification. Machine-learning can be used to analyze simulations performed in the past to infer patterns which may or may not be necessarily observable under manual analysis using such variables as historical simulation, functional coverage, test results, failure information, and design characteristics. Potential application of AI is intelligent test generation, prioritizing stimuli, coverage-gap detection, classifying failures, choosing regression-tests and planning adaptive verification. Instead of viewing every test generated as equal a smart verification system can be made aware of the outcomes of other verification processes and focus future verification processes on more informative or unexplored cases.

Christianity puts AI and UVM together is therefore a possibility. UVM can provide the normalized verification infrastructure and AI can be a smart decision-layering. This type of architecture is able to sustain the reusability and modularity of UVM and offer flexible alternatives to test creation and verification verification. Smart UVM is already setting the foundations of the role of extrapolating the traditional UVM to more automated verification circumstances [2]. Nevertheless, an all-encompassing approach to the combination of AI-based analysis and reusable UVM blocks and a clear positioning of the approach as needed in multiple industry-related use cases is a valuable research potential.

The present research proposes a UVM framework based on AI of verifying hardware specifically in the industry. The proposed architecture incorporates the modular design of UVM, and AI test generation, test coverage, test prioritization, failure mode and adaptive test optimization. Certain industry configuration is given in a bid to ensure the



verification goals, constraints, protocols, and critical situations, to be geared towards the desired application domain. The structure will be structured to offer feedback-oriented feedback mechanism through which outcomes of the simulation and data on coverage will provide continuous impacts thereon on subsequent verification processes.

The important focus of the proposed solution to the subject is to provide the systematic method of optimizing the performance of verification without jeopardizing the aspect of reusability as well as scalability, which accompanies UVM. The research seeks to limit unneeded test execution, detect uncertainty of verification gaps in a better fashion, enhance convergence of coverage and aid systematic research of verification failures by combining AI with automated verification systems. The process thus encompasses the growing needs of intelligent, versatile and application-aware application level hardware verification platforms. It provides a basis to say goodbye to traditional test-based UVM verification and pass to a more data-based and flexible verification paradigm that can fit an ever more complex industry-specific hardware system.

II. RELATED WORK

The hardware verification is becoming more important in the digital-system design cycle as the System-on-Chip (SoC) itself, intellectual property (IP) and embedded processor and application-specific hardware grows in complexity. Old forms of verification are very much based on test cases that are developed manually, constrained-random stimulus, simulation, and coverage analysis. The recent activity has particularly been in the area of automation, reusable verification environments, UVM-based methodologies, and, most recently, machine-learning-assisted verification. The proposed UVM methodology of industry-specific applications with AI in hardware verification is based on these developments.

Sharma, Bhargava and Kumar studied automated bug resistant test intent in their research and observed automated test generation where a verification strategy based on register-headers is applied [1]. Their methodology deals with the challenge of creating a large enough number of test cases which result in validity of a complex SoC design. The method uses register intent as a source of test intent and has a UVM register-abstraction environment to automate test generation. The authors reported that simulation time, CPU utilization and functional-coverage closure showed positive gains whereas the amount of effort in the development of manual tests was reduced. It is especially helpful in the context of the proposed research as it shows that UVM may be used as an automation-focused platform, as opposed to a traditional testbench platform. However, the approach mostly turns to rule-based and database-based automation as opposed to experience of what is already known about past verification outcomes. This constraint opens the possibility of implementing AI-driven dynamic decision-making on UVM environments.

Also noted was power-aware verification, since modern SoCs are faced with fewer functional requirements, and stricter power constraints are imposed on them. Macko, Jelemenskaya and Cichak examined how specifications of power-management can be checked at a very low level in the power-constrained system design [2]. It goes without saying that work offers an excellent mention of the importance of testing the behavior of power-management before other subsequent stages of hardware development. This verification can save the expense of discovering design problems, and enhance the trust of the power-aware systems. Although the study does not address AI or UVM-based adaptive verification, it demonstrates that it is necessary to go beyond the notion of functional correctness in verification and implement it to system-specific requirements. The principle applies most to an industry-based construct where the intentions of verification may differ across the automotive industry, the telecommunications industry, consumer electronics and other hardware market divisions.

Access to coverage registers with a UVM framework were also studied by Sharma, Bhargava and Kumar [3]. Their work shows the protrusion of UVM to automation mechanisms to make verification information more available and analyzed. Particular attention should be paid to the coverage because it does not necessarily imply that in the case a large number of tests is carried out, the verification is sufficient. Understanding Case coverage-driven verification on the other attempts to indicate untested or under-tested functional cases. This study, therefore, proposes the use of coverage data in an intelligent verification cycle. An AI component of the proposed methodology can also be fed with the information about coverage to identify the gap of verification that would further guide the additional stimulus generation. In this sense, the previous UVM automation study make up the foundation of the architectural structural foundation, whereas AI may have a potential to provide a more responsive procedure in inferring which situations of verification are to be probed next.



The authors Trummer and colleagues discussed the automated check of power requirements of SoCs over simulation and checking power requirements [4]. This methodology entails semi-formal use-case representation in the specification of power and system requirements, and automatic derivation of test cases, generation of a verification environment and utility of simulation and power estimation when performing verification. The example reveals that there is a big change in the manually developed verification processes to the automated requirement-to-test flows. The concept is especially important to industry-specific verification as various products may sprinkle varying requirements. Safety and power-management behavior verification may be needed at a car controller, e.g. a car brake one, as well as protocol correctness and performance at a communication device. A flexible UVM architecture may provide common verification infrastructure whereby the specifications of an application can specify the conditions generated and the coverage goal.

Salah proposed intelligent functional verification platform founded on UVM which involves the analysis of the concepts, benefits, constraints and opportunities of UVM [5]. On the methodological level, the work is important, in that UVM has reusable parts and standardized procedures to create advanced functional verification environments. The concept of a smart verification platform is mustered with the orientation of AI-assisted verification, as the former is deprived of the machine-learning capabilities that the study under analysis. The proposed framework adds to this direction with AI acting as an analytical layer with the capability of learning through verify feedback and enable adaptive test selection and test generation.

Swain and Mahapatra conducted a study on the design and validation of a bus interface WISHBONE at the level of SoC interface [6]. Their efforts indicate that systematic verification of communication interfaces is an important concept since it proves that interface-level errors may spread to bigger system-level errors. The verification of interfaces is also an adequate field to reuse UVM components as the UVM would be anticipated to perform the same verification framework with the difference of IP configurations. This justifies this industry-specific approach whereby reusable UVM agents, drivers, monitors, scorecards and coverage models can be set-up as per the needs of a specific hardware implementation.

Oudjida and Berrandjia investigated the architecture and testing of general-purpose forms of SPI master/slave IPs into the OPB bus [7]. Their research provides an application-based example of the IP-level testing and applicability of the testing of communication protocols in the integrated hardware configuration. On a related note, Zhou, Xie and Wang were able to come up with a SystemVerilog-based verification signatory on an SPI master interface [8]. These pieces of work show how the protocol-specific verification environments are getting elaborated by means of more systematic hardware verification approaches. However, these are typically limited to the specifications of the prescribed test conditions and protocols. The suggested framework aims at generalizing this process with a combination of reusable UVM infrastructure, along with AI-aided choice and prioritization of verification situations.

Moon used the example of an interface to multimedia cards in ARM embedded systems [9], whereby, interface and embedded-system requirements are intricately related. UVM was then implemented by Roopesh and Siddesha on the RTL design and verification of an SPI master-slave architecture [10]. Reddy, Sreekanth, and Kumar [11] also studied a universal verification component of an SPI master that is based on UVM. Combinations of these studies substantiate UVM as a suitable asset to reusable protocol, and IP verification. But media In their application, however, they employ a more or less standard interface, and a uniform verification systems. They could be the reusable components in case of industry-specific AI-assisted approach so that they are the elements that form a verification infrastructure with an AI layer that examines the coverage, test failures, and the previous test run in order to inform the subsequent verification work.

Zhang, Li and Zeng [12] discussed the problem of the design of a high-speed and high-reliability SPI interface. Their study has the quality of reliability and performance of the interface oriented hardware systems as its center. Functional correctness might not be enough in an actual industrial verification, timing and reliability, power, performance and protocol-specific constraints may all be required. This means that an industry-specific verification framework should be able to support a variety of verification objectives unlike adherence to the same test strategy, regardless of the hardware design.

The birth of machine learning offers another twist to the scheme of enhancing the productivity of verification. A design verification with machine-learning under open-source verification environment was proposed by Varambally and Sehgal [13]. Their work also recognizes the fact that constrained-random verification can be very assisted by human know-how to make any serious progress to key corner-cases and propose machine learning to create input stimuli. The



authors take a particular interest in discussing the applicability of ML to minimize human interference and enhance all-encompassing verification. The research is quite relevant to the present study as it has demonstrated feasibility of generating stimuli with the help of learning strategies. It is however verified using Cocotb instead of UVM. There is, therefore, an opportunity to research and combine the stimulus generation with the standardized and reusable architecture of UVM, which is ML-driven.

Overall, three major research avenues identified in the literature considered are the UVM-based reusable verification, the automation, and coverage-driven verification, and the machine-learning-assisted verification. Recent research in UVM is reporting good performance in capabilities in reusable IP and protocol verification [3], [5], [10], [11], and automation research is reporting better performance in one-on-one test development and coverage-driven process [1], [4]. The research in machine-learning suggests the idea of adaptive stimulus generation and reduced human-intervention [13]. These research avenues are however by and large researched individually. There is one AI-based UVM ecosystem, which involves reusable UVM components to provide the verification infrastructure and AI to analyze past test result and coverage data, and custom industry-specific configuration determines on verification goals and constraints. The proposed research addresses this gap by integrating these capabilities and including the structured approach to facilitate adaptive, scalable and application-aware verification of the hardware.

III. AI-ORCHESTRATED UVM VERIFICATION FRAMEWORK FOR INDUSTRY-SPECIFIC HARDWARE

The given scheme adds Universal Verification Methodology (UVM) and artificial intelligence (AI) and machine learning (ML), coverage analytics, adaptive stimulus generation and industry-specific hardware verification requirements to a unified hardware verification architecture. The framework will overcome the constraints of traditional UVM environments in which one frequently needs to do much manual work to generate tests, make decisions about the priorities of regression, conduct coverage analysis, and debug failures. The main concept behind the proposed framework is that modularity and reusability of UVM can be maintained but a smart feedback loop that will continually learn out of verification results is implemented.

3.1 Framework Architecture

There are seven inter-aspect layers of the framework: Industry Requirement Layer, UVM Configuration Layer, Stimulus Generation Layer, Adaptive Feedback Layer, Simulation and Execution Layer, Verification Analytics Layer, and AI Intelligence Layer. These layered are not discrete phases.

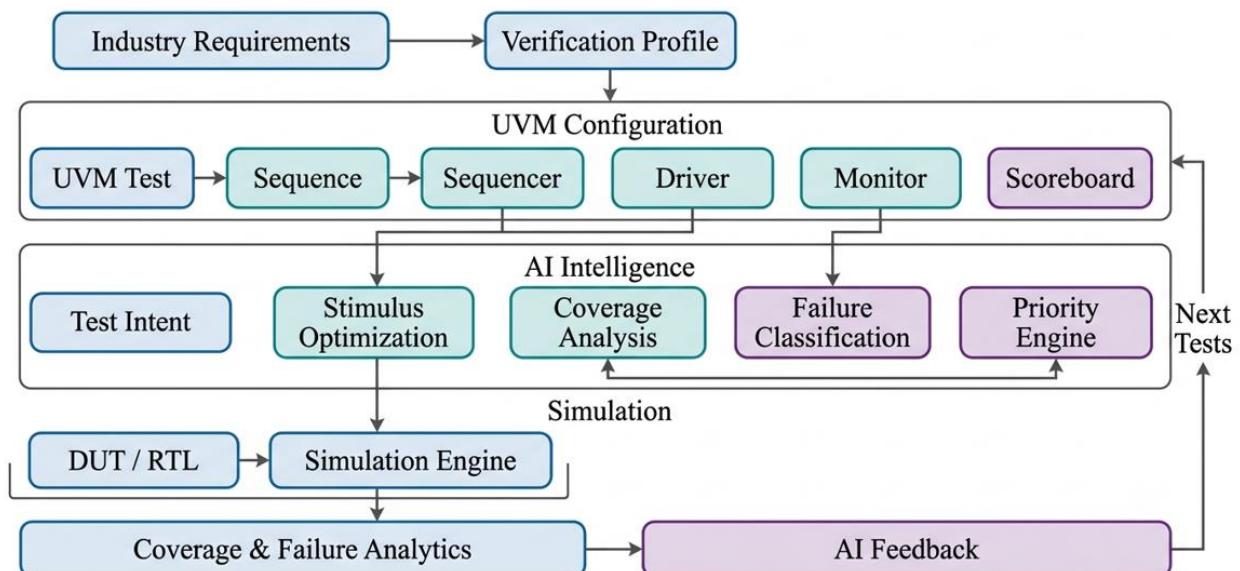


Figure 1- AI-Orchestrated UVM Verification Architecture

In Industry Requirement Layer, application specific verification objectives are determined. It reflects attributes of the target hardware domain, such as functional and communication requirements, operating environments, safety limits,



power limits, performance requirements, and critical failure scenarios. The same framework is reconfigured using configuration parameters to other hardware applications including the automotive electronics, telecommunications, industrial controllers, consumer electronics and embedded systems.

The verifiers are the reusable ones located in the UVM Config Layer. It consists of the UVM testbench, test classes, sequences, sequences of sequencers, drivers, monitors, agents, score boards, reference models, assertions, and coverage collectors. UVM agents, being protocol protocol-oriented, can be turned on or off depending on hardware desires. The layer divides AI-based decision-making mechanism and design-specific verification logic of the verification mechanism in order to maintain the modularity of the verification environment.

The main innovation of the framework is the AI Intelligence Layer. It obtains input using the history of verification cycles done in the past and looks into the history of the test, coverage levels and the simulation results, failures, transactional trends, and gaps in verification. Based on the use case, it can include ML classification, clustering, prediction, optimization, or reinforcement-learning algorithms. The layer develops suggestions on the selection of test, nature of stimulus, priorities on verification and coverage goals.

The Stimulus Generation Layer converts AI recommendations to UVM sequences, which can be instantiated. The layer conveys the synthesis of certain stimuli in line with the revealed functional condition, the experienced failures previously, protocol constraints, and industry-specific requirements as compared to entirely basing on the transactions which are manually-specified or randomly-uniform. To UVM sequencer are sent generations, and to the respective driver.

3.2 Industry-Specific Requirement Modeling

One of the key elements of this framework is the Industry-Specific Verification Profile (ISVP). The profile attempts to provide a model that is customizable on the validation objectives of a single application domain. It comprises of five key areas, which include functional requirements, interface requirements, performance requirements, reliability requirements, and application specific constraints.

An automotive profile of verification may bring into focus examples of safety-related situations, faults, and reliability of the communication and power conditions as an example. A telecommunications profile may be biased towards protocol series, conditions of high throughput, transactions that are sensitive to latency, and interoperability. The deterministic operation, fault recovery, and long operating conditions can be targeted by an industrial-control profile.

Industry-Specific Verification Profile Model

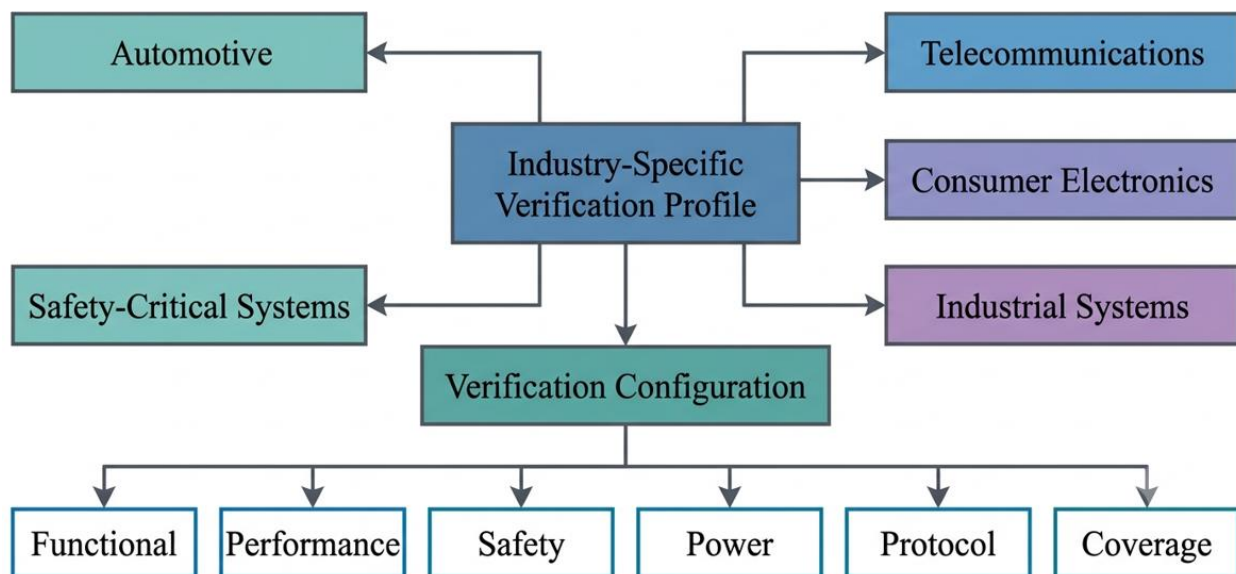


Figure 2- Industry-Specific Verification Profile Model



The AI model cannot solve various problems of verification in the same way when it is implemented by the ISVP. Instead, the AI layer receives domain-specific goals, deciding which test is of high priority and the stimuli generated. That enables the framework to be more flexible with no requirement to possess an entirely new UVM architecture in every industry.

3.3 Reusable UVM Verification Infrastructure

The UVM layer is hierarchical in nature and modular. At the base, UVM drivers convert the transactions into interface signals, and UVM monitors gather design outputs and record information on transactions. Sequencer; This is the controller of the transactions generated and sequences determine the states of functionality where one can be simulated.

The scoreboard also compares the expected and actual behaviour and the reference models that provide expected results at the level of transactions. Propositions can keep track of time-dependent and protocol-dependent properties. Functional and structural information which is needed to determine verification is captured by coverage collectors.

When verification requirement is the case, reusable UVM agent can have a driver, a sequencer and monitor and be located as an active agent or a passive agent. There are many different agents that can be attached to an environment which has a number of interfaces or IP blocks. This type of organization enables the proposed AI layer to not depend on the implementation of specific protocols.

3.4 AI-Driven Test Intent Generation

The system proposes AI-aided intent to test generation process. Test intent is a behavioral objective, which a particular verification scenario is supposed to perform. The AI engine analyzes the state of the verification, and decide to further criticize the area that requires exploration rather than giving incorrect random transactions.

Input features may include:

- Current functional coverage
- Code coverage
- Assertion coverage
- Previously executed tests
- Failure frequency
- Transaction patterns
- Uncovered bins
- Interface states
- Protocol constraints
- Industry-specific priorities
- Historical test effectiveness

The AI engine assigns priorities to potential verification situations relying on these attributes. The scenarios with high priorities are translated into UVM sequences and sent to the simulation environment. This brings about a shift between simply random stimulus generation and coverage-conscious and learning-directed stimulus generation.

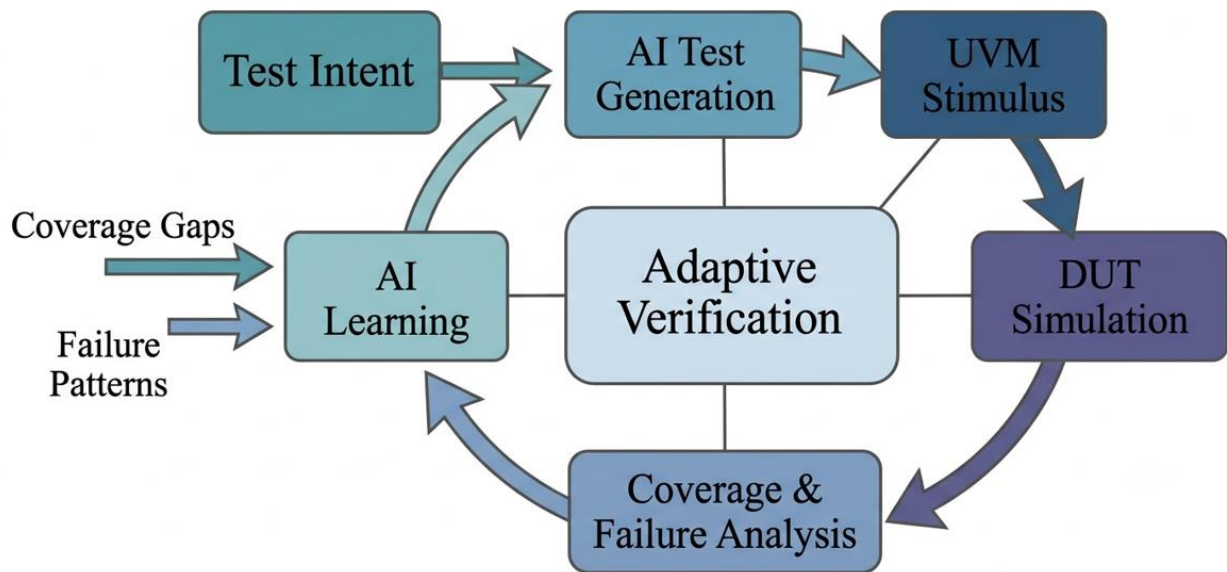


Figure 3- AI-Driven Adaptive Test Generation and Verification Loop

3.5 Adaptive Stimulus Optimization

Once every verbal cycle is completed, the results of the simulation are sent back to the AI engine. The framework identifies the added significance to the completed test, whether through the failure in a test, use of unexplored state, or unnecessary coverage.

The additional information in tests which are not important by themselves may be assigned low priority in the subsequent regressions. Conversely, any circumstance that illustrates any considerable failures or a considerable coverage can be a root out of which corresponding test cases can be created.

The system of feedback style architecture enables the verification environment to evolve according to the observed results, rather than follow an established order of specially writing down evaluation-of tests.

3.6 Intelligent Coverage and Verification-Gap Analysis

The coverage data are not an endpoint verification measure but rather, are modeled as an input into the AI layer. The framework collects code, protocol, assertion, scenario and functional coverage and constructs a profile of verification-gap.

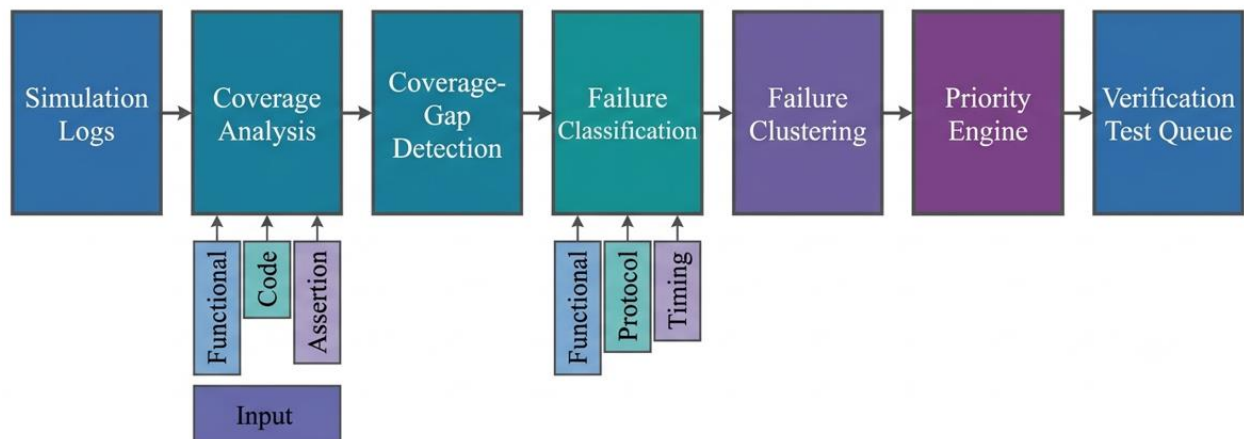


Figure 4- Intelligent Coverage and Failure Analytics Pipeline



The AI engine reviews and analyses the open spaces or poorly covered spaces and values the space according to industry-related priorities set. To provide a case, a critical situation of safety may be assigned much higher priority as compared to the low-risk functional situation.

The regression cycles are carried out to check convergence of the coverage. The framework can like identify the unnecessary tests, slow regions, persistent regions of coverage, and the circumstances pertaining to recurrent failures. As a result, the verification will be more resource-conscious and focused.

3.7 AI-Assisted Failure Classification

Failure analysis is another aspect that is applicable in the framework. Widely used regression or projection environments are capable of producing extensive amounts of failure logs, and it is time-consuming to manually classify all of them. The proposed framework derives the characteristics of relevance in the simulation logs, embrace violations, transaction traces, scoreboard discrepancies and error messages.

The AI engine categorizes failures into groups such as:

1. Functional mismatch
2. Protocol violation
3. Timing-related anomaly
4. Configuration error
5. Assertion failure
6. Environmental/testbench error
7. Repeated or previously identified failure

Failure clustering can avert revisiting the investigation of basically the same errors. The framework is able to map onto an existing failure signature an underlying failure that is known. New failure modes can be of higher priority to investigate.

3.8 Verification Priority Engine

Verification Priority Engine (VPE) consists of coverage status, failure information, test effectiveness and criticality (industry-specific). These factors assign a dynamically calculated priority to a problem of each possible test scenario.

The priority mechanism enables the limited computational resources to be directed towards the most informative checking. The framework is not required to execute all the tests with equal rate; one can emphasize the scenarios of uncovered functionality, critical requirements, recent failures or high-risk operating conditions.

The priority engine can also be used to aid in the optimization of regressions, where a smaller but more informative sub set of tests is selected where the entire regression can not be executed.

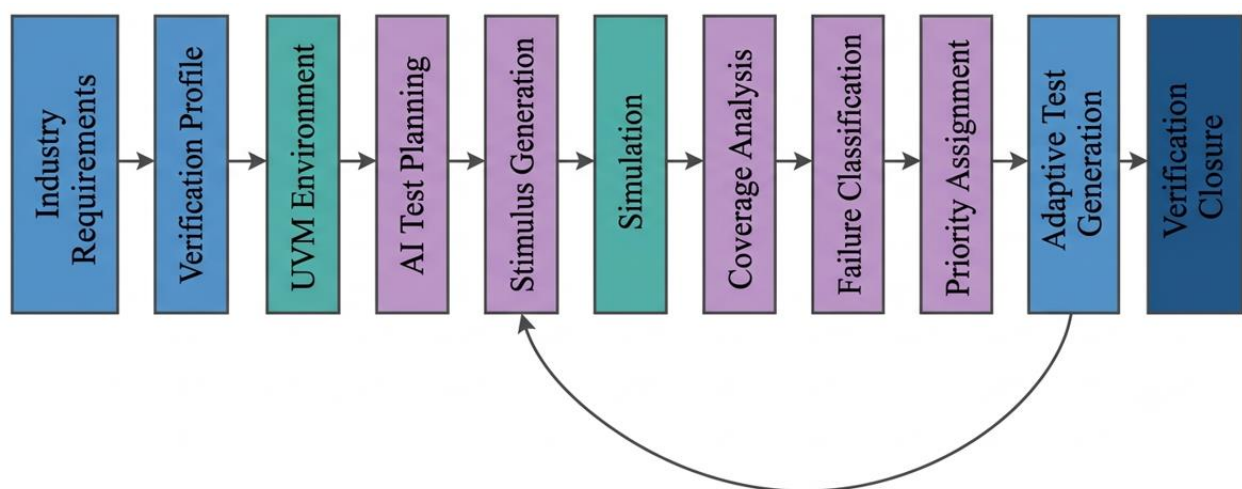


Figure 5- End-to-End AI-UVM Verification Workflow



3.9 Feedback and Continuous Learning

The final layer offers a real-time feedback of the engine and the UVM surrounding to the AI. Any simulation done incorporated into the verification knowledge base. The stored information can include the inputs of the test, coverages-changes, test failure-signatures, the time of test execution, effects of such tests on functions and the effectiveness of the test.

Periodically, the AI model refreshes its knowledge about the verification space relying on this body of collected information. As a result, the framework enhances during the verification campaign. The earlier phases may take advantage of a exploratory testing in order to develop a preliminary level of knowledge, however the phases later may apply more concentrated select of tests which may be influenced by the growing body of knowledge.

This educational procedure keeps the proposed organization of the immutable UVM testbenches. The verification environment does not simply put a series of already established tests into motion; it monitors, evaluates, prioritizes and changes.

3.10 Framework Execution Workflow

The whole working process starts with the target hardware configuration and the selection of the appropriate industry concrete verification profile. Next the UVM environment set up is configured with relevant agents, sequence, monitoring, coverage models and scorecards. Predefined constraints and AI-assisted stimulus mechanisms are used to create initial tests.

The transactions generated are run using the UVM simulation environment. The verification analytics layer collects the simulation results, coverage data, assertion outcome and loss of logs. This results are analyzed by the AI engine which identifies gaps in verification and high-value cases, repeated failures, and unexplored potential states.

Subsequent verification objectives are then prioritized in the priority engine. Based on these priorities, new UVM sequences are created and reentered into the simulation. This procedure will continue until the objectives of the verification, which are set, are achieved or the budget allocated on the verification is exhausted.

IV. EVIDENCE-DRIVEN EVALUATION OF THE AI-UVM VERIFICATION ARCHITECTURE

The proposed AI-powered UVM framework is discussed in terms of efficiency of the verification, convergence of coverage, efficiency of a test-generation, detection of failures, flexibility, and configuration control related to a specific industry. This will be evaluated by the ability of the integration of AI with UVM to deliver a more adaptive and systematic verification process as compared to a traditional UVM environment. Instead of the final percentage of completion, the evaluation penetrates the procedure of fulfilling the framework to the objectives of verification, in the context of response to information obtained in the simulation loops.

4.1 Verification Coverage Assessment

Coverage convergence is one of the key indicators that will have to be used to implement the efficiency of the proposed framework. Assertion coverage and scenario coverage are monitored through successive verification, functional coverage, code coverage. AI-UVM model is a meta-analysis which results in discovering the revealed or unused areas and creating direct stimulus that will address the gaps identified.

The assessment is done in relation to the growth rate of coverage of the regression cycles. Higher order of convergence suggests that the framework is a priori giving very high priority on the informative verification scenarios. Special consideration is given in particular to difficult-to-reach functional bins and to application-specific situations because, in these cases, the classical random stimulus would only explore partially.

4.2 Adaptive Test-Generation Assessment

The adaptive test-generation power is viewed by comparing the working of the conventional constrained-random UVM sequences and AI-informed sequences. The test is determined by the number of different scenarios formed, the percentage of the test that forms new coverage, repeat test executions and simulation run cycles to meet defined coverage objectives.

The AI layer will start altering the choice of stimuli with time, by the results of prior checks. As a result, subsequent test events should show greater targeting of the exposed functionality as opposed to re-performing already fully-



covered scenarios. This will provide an indication on whether there is valuable adaptation added to the proposed feedback mechanism.

4.3 Failure Detection and Classification

As error in simulation, assertion violation, scoreboard mismatch and protocol failure are measures of failure-analysis capability. The framework classifies the failures that are detected based on their behavioral features and recognizes recurring failure modes.

The means of measurement include the quantity of failures found, new failure signatures, classification stability and the time it took to detect failures. The former gives particular attention to the framework capability of classifying real design-related failures, and to the testbench or configuration-related ones. A proper classification can assist in minimizing the extra manual analysis and can assist in a more guided debugging.

4.4 Verification Efficiency

Some examples of tests to test verification are: verification efficiency (evaluated by time to execute simulation, time to generate test), the size of the regression, and the use of computational resources. It is hoped that the suggested framework will minimize unnecessary executions of tests by initially prioritizing the scenarios which will likely give helpful verification information.

A good appraisal will be the correlation between what has been calculated and the enhancement of coverage. Rather than, as ignoring the total amount of the conducted tests, the analysis looks at the coverage contribution per simulating cycle of at least per unit of the computational effort. This is more appropriate in the measurement of verification productivity.

4.5 Industry-Specific Adaptability

The framework is also tested via various profiles of validation in terms of the industry. Depending on the verifying requirements and constraints, the same UVM infrastructure can have different verification priorities. An automated profile may be safety and fault-conscious (e.g. automotive), or it may be communication-conscious (e.g. communication) as typically done to emphasize protocol sequence, latency and throughput.

The question to be answered is whether the framework will be capable of having to facilitate these differences without having to radically restructure the underlying UVM architecture. An efficient configuration depicts efficiency and scalability of the proposed methodology.

4.6 Comparative Evaluation

Lastly, the framework may be equated with a traditional UVM verification environment with similar design configurations and verification goals. Convergence of coverage, effective test generation, redundant test execution and verification effort has been compared.

The evaluation will be aimed at establishing whether AI integration can introduce a quantifiable value without causing loss to the modularity of UVM and its reusability. Overall, the assessment framework provides the multidimensional framework according to which the proposed methodology is to be checked, and determines its relevance to the task of adaptive, industry-specific verification of hardware.

V. FORWARD-LOOKING EVOLUTION OF AI-AUGMENTED UVM VERIFICATION

The proposed AI-based UVM platform can be used as a starting point where verification of adaptive hardware and hardware per industry can occur. This can even be advanced more in future to enhance its intelligence, scale, interoperability and its applicability in more complex SoC and IP verification settings.

5.1 Reinforcement Learning for Autonomous Test Planning

In the future reinforcement learning (RL) may be introduced to enable the verification engine to be trained to the best test-selection strategies by being repeatedly interacted with the simulation environment. Learning can deal with improvement in coverage, new failure detection and investigating untested states. This can allow more and more independent verification planning without the need to specify manually the priorities of each verification scenario.



5.2 Generative AI for Test and Sequence Synthesis

Generative AI is one of the resulting avenues of exploration, with an automatic generation of the UVM sequences, constraints, assertions and test scenarios, all possible to generate given the design specifications and verification requirements. Natural-language requirements might be converted to structured verification intents and then converted to executable components of UVM. This would massively relax effort that would otherwise be geared towards establishing verification environments of new IP blocks.

5.3 Digital-Twin and Virtual Verification Integration

Connecting the framework with digital-twin and virtual-prototype environments would potentially enable the framework to be used to assess the behavior of hardware not yet physically prototyped. The AI models might learn the virtual simulation results and be used in the corresponding knowledge on verifying this in a RTL-level UVM environment to create a pipeline of continuous verification initiated with early exploration and carried on with the architecture implementation.

5.4 Explainable AI for Verification Decisions

The further procedures are adding explainable AI logic to justify why a particular test should receive priority, to justify why a coverage area is defined to be important, or to justify why a failure must be classified into a particular class. This transparency would enhance the verification-engineer trust and make it easier to debug AI-aided decisions.

5.5 Continuous Learning and Cross-Domain Reuse

A model can be reused in the future with verification expertise shared across projects and domain settings can be maintained. Transfer learning is potentially useful to enable models developed on one hardware family to fast-track validation of a new design in the family. This may help minimize the training need of each new verification project.

5.6 Cloud-Scale Intelligent Regression

AI-based regression scheduling, which is associated with distributed/cloud-based simulation services (p.t.), could be supported by enormous computational resources. The framework had the potential to dynamically redistribute high-priority test and reduce the number of redundant simulation, facilitate large-scale verification of next generation SoCs.

VI. DISCUSSION: FROM SCRIPTED UVM TO ADAPTIVE VERIFICATION INTELLIGENCE

The proposed AI-based UVM architecture will be a paradigm shift of moving away to more manual and scripted verification rather than an information-driven and adaptive one. Traditional UVM encourages the modularity, transactional abstraction, reuse and creation of testbench structures; however verification engineers are left to determine types of tests to generate, execute, or re-execute tests, and the priority of tests. The introduction of AI to this model will add an additional element of decision making that will have the capacity to process information related to coverage, previous results, trends in failures and application specific requirements.

A major advantage of the framework is that the verifying closed circle is taken into consideration. The simulations are not considered as an isolated incident but the output of simulations is an independent to further test planning. This will enable the framework to incrementally target the verification resources to the uncovered functionality, difficult scenarios, and noteworthy failure cases. This greedy behavior has a potential to improve convergence of coverage as well as reduce redundant implementation of regression.

The other factor is the industry specific structure. The needs of hardware verification differ significantly when comparing automotive, telecommunications, industrial, consumer-electronics, and embedded. The framework can be architecturally sound by isolating reusable UVM infrastructure and reuse-specific verification profiles as well as allowing verification priorities and constraints to adapt depending on the target domain.

The framework also makes the opportunities of systematizing the analysis of failures open. The classification of failures in simulations based on AI can put failures into their respective groups based on the nature of their occurrences and recognize trends. This can help verification engineers prioritize their debugging efforts on new or high-priority failures instead of continually investigating similar situations.

However, there are also challenges associated with AI implementation. The quality and diversity of verification data determines the effectiveness of AI models. Smaller training data sets would result in inappropriate prioritization of the tests or failure to explore the unusual cases. Other factors such as the computational cost, model explainability,



interaction of the model to current EDA processes, and validation of AI-generated verification decisions go into the consideration.

In general, the proposed methodology provides a conceptual bridging of the reusable UVM infrastructure and intelligent adaptive verification and opens the prospect of more scalable and application-conscious hardware verification.

VII. CONCLUSION

The suggested AI-based UVM system of verifying hardware-specific industries creates an adaptive verification framework with a fusion of the reusability and modularity of the Universal Verification Methodology and analytical nature of artificial intelligence and machine learning. The increasing simplicity of hardware, a shift in application requirements and a test workload that is ever increasing have created the drive to create verification techniques that can offer an intelligent assistant to test planning, stimulus generation, coverage analysis and failure analysis. This framework addresses these requirements, through an integrated AI-UVM design.

The methodology suggests an Industry-Specific Verification Profile (ISVP) that enables establishing verification objectives, verification constraints, coverage priorities as well as functional requirements as application domain such as automotive electronics, telecommunications, consumer electronics, industrial systems and safety-critical hardware. Relying on the reusable UVM infrastructure, standardized verification components are offered; on top of the infrastructure is the AI intelligence layer, which generates test-intent, does adaptive stimulus optimization, detects coverage-gap, categorizes the failures, and prioritizes verification.

A significant contribution of the framework is the fact that the framework has closed loop verification mechanism. Coverage information, simulation outcomes and failure property is constantly being analyzed and fed-in to further test-generation decisions. In this way, verification activities can be allowed to develop based on the results that are observed instead of only using predefined test scenarios. Therefore, the methodology offers the framework of enhancing convergence of coverage, minimising overlapping regression tasks and allocating verification resources to high priority situations.

The dimensions of the framework evaluation also establish a framework grounded on which the coverage enhancement, adaptive test development, failure examination and efficiency of verification and industry-concrete adaptability can be evaluated. In the meantime, the problems associated with the AI-data quality, model interpretability, computational requirements, complexity of the integration, and AI-prepared decisions, which should be addressed before large-scale implementation, should be tackled.

In general, the suggested methodology illustrates the way in which AI-enhanced UVM could transform reusable automation to intelligent and adaptive and application-aware hardware verification. This methodology can be extended in a variety of ways to support more complex needs of semiconductor and SoC verification, including generative AI, reinforcement learning, explainable AI, digital twins, continuous learning, and cloud-scale regression.

REFERENCES

- [1] Sharma, G., Bhargava, L., & Kumar, V. (2020). Automated bug resistant test intent with register header database for optimized verification. *Journal of Electronic Testing*, 36(2), 219–237. <https://doi.org/10.1007/s10836-020-05866-5>
- [2] Macko, D., Jelemenská, K., & Čičák, P. (2017). Verification of power-management specification at early stages of power-constrained systems design. *Journal of Circuits, Systems and Computers*, 26(8), Article 1740002. <https://doi.org/10.1142/S0218126617400023>
- [3] Sharma, G., Bhargava, L., & Kumar, V. (2018). Automated coverage register access technology on UVM framework for advanced verification. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)* (pp. 1–4). IEEE. <https://doi.org/10.1109/ISCAS.2018.8351413>
- [4] Trummer, C., Holzinger, A., et al. (n.d.). Automated simulation-based verification of power requirements for systems-on-chips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.
- [5] Salah, K. (2014). A UVM-based smart functional verification platform: Concepts, pros, cons, and opportunities. In *Proceedings of the 9th International Design and Test Symposium (IDT)* (pp. 94–99). IEEE. <https://doi.org/10.1109/IDT.2014.7038594>



- [6] Swain, A. K., & Mahapatra, K. K. (2010). Design and verification of WISHBONE bus interface for System-on-Chip integration. In *Proceedings of the IEEE India Conference (INDICON)* (pp. 1–4). IEEE. <https://doi.org/10.1109/INDICON.2010.5712616>
- [7] Oudjida, A. K., & Berrandjia, M. L. (2010). Design and test of general purpose SPI master/slave IPs on OPB bus. In *Proceedings of the 7th International Multi-Conference on Systems, Signals and Devices* (pp. 1–6). IEEE. <https://doi.org/10.1109/SSD.2010.5585592>
- [8] Zhou, Z., Xie, Z., & Wang, X. (2012). Development of verification environment for SPI master interface using SystemVerilog. In *Proceedings of the International Conference on Signal Processing* (pp. 2188–2192). IEEE. <https://doi.org/10.1109/ICoSP.2012.6492015>
- [9] Moon, S. (2012). Design of an SPI interface for multimedia cards in ARM embedded systems. *Journal of the Korean Institute of Information and Communication Engineering*, 16(2), 273–278. <https://doi.org/10.6109/jkiice.2012.16.2.273>
- [10] Roopesh, D., & Siddesha, K. (2015). RTL design and verification of SPI master-slave using UVM. *International Journal of Advanced Research in Electronics and Communication Engineering*, 4, 2159–2162.
- [11] Reddy, P. R., Sreekanth, P., & Kumar, K. A. (2017). Serial peripheral interface-master universal verification component using UVM. *International Journal of Advanced Technologies in Engineering and Management Sciences*, 3, 27–30. <https://doi.org/10.22413/ijastems/2017/v3/i6/49102>
- [12] Zhang, C., Li, B., & Zeng, X. (2019). A high-speed, high-reliability SPI interface design. *Information & Communication*, 5, 61–63.
- [13] Varambally, B. S., & Sehgal, N. (2020). Optimising design verification using machine learning: An open source solution. *arXiv*. <https://arxiv.org/abs/2012.02453>