2025 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE
MUNICH, GERMANY
OCTOBER 14-15, 2025

GAIL-V: Generative AI Leveraged -Verification

A Systematic Framework for Leveraging Generative AI in Verification

Sriram Madavswamy, Staff Verification, ARM, Manchester, UK (sriram.madavswamy@arm.com)

Daniel Larkin, Director of ISP Engineering, ARM, Manchester, UK (daniel.larkin@arm.com)

***Abstract*— As the complexity of semiconductor designs continues to grow, the verification effort has expanded significantly in both time and resources. Traditional automation techniques—such as constrained random testing, assertion-based verification, and metric-driven verification—have helped streamline the process but remain heavily manual in nature when it comes to planning, debug, and stimulus generation. The emergence of Generative AI (Gen-AI) has shown promise in domains such as software engineering and natural language processing; however, its application in hardware verification remains largely exploratory. Most attempts are ad hoc, lacking a structured way to assess where and how Gen-AI can be embedded into the verification lifecycle. This paper builds on these explorations and proposes a systematic framework that identifies, categorizes, and evaluates Gen-AI opportunities across the verification flow.**

Keywords—generative ai, verification framework, AI in verification

I. INTRODUCTION

Generative AI (Gen-AI) shows potential to improve productivity, as demonstrated by a few early, though limited, use cases. This gives rise to the question “What can we use Gen-AI for in verification specifically?”. Most use cases from our observations and peer groups have been ad hoc. For instance, individual engineers use Gen-AI models to help with a part of script or entire scripts or uploading documents and asking queries related to it etc. These approaches are not scalable and not reusable. We came up with a framework to systematically assess where Gen-AI can be used to support verification processes. To support the adoption of this framework, we are building a scorecard/dashboard that can quantify Gen-AI performance. This evolving system will be refined using feedback from the broader verification community subject to conditions (refer section II.C) and contribute to a growing knowledge base on where Gen-AI works, how well it performs, and what input configurations matter. To systematically evaluate where Gen-AI can be applied, we introduce the GAIL-V framework, which decomposes verification tasks into sub-goals and benchmarks Gen-AI performance on each

II. METHODOLOGY

A. Steps in the framework

Consider a design under test (DUT) which is reasonably complex; With the current state of Gen-AI, can we just provide the design specification and RTL code base path as inputs and expect Gen-AI to provide a complete setup including verification plan, test bench plan and its implementation, running tests, finding bugs, closing coverage and sign-off the verification? The answer is ‘no’, but Gen-AI does show promise in certain isolated tasks. So, the hypothesis is that if we break down verification processes into modularized goals and then use Gen-AI for the goals that can leverage it well (refer from now on as Gen-AI-fit goal) and continue to use existing human work in other goals (refer from now on as non-Gen-AI-fit areas).The steps in the methodology are as follows:

1. Identify and hierarchically decompose verification flow into goals and sub-goals
    a. Sub-goals decomposed into further sub-goals till we reach a leaf-level sub-goal (we will refer to 'leaf-level' goal from now interchangeably as sub-goal)
2. Gen-ai exploration for a sub-goal
        i. Identify the workflow and assess if it could be a Gen-AI-fit goal
        ii. Identify the ideal Gen-AI based workflow
        iii. Come up with a benchmark
        iv. Identify the ideal gen-ai based workflow
        v. Come up with a benchmark
            1. Come up with data set model

2. Come up with evaluation criteria for benchmarking
3. Regress over different models and prompting techniques

vi. Apply on real world situations

B. Hierarchical decomposition

Figure 1, 2 below visualizes the hierarchical decomposition of the verification workflow, labeling each component with an identifier such as DV_L1_G1[Figure 1] or DV_L1G2_L2G2[Figure 3]. This structure supports a common vocabulary among verification teams to refer to specific blocks, enabling traceability, automation mapping, and a shared understanding of where Gen-AI can be most effectively applied. Figure 1 shows the high-level decomposition of the verification process into level 1 goals, namely verification plan, implementation, and analysis. Within each main goal, there are subgoals. For example, Figure 2 shows the decomposition of level 1 goal into sub-goals. DV_L1G1_L2G2 is a level 2 subgoal within a level 1 goal, and it is to extract verification features from various inputs such as Unit Design Specification (UDS), Top Design Specification (TDS), Algorithm document (ALG) etc. Figure 3 shows sub-goals within the main goal 'implementation'. We understand well that a single framework may not be the best solution for every verification team, but this acts as a good starting point.

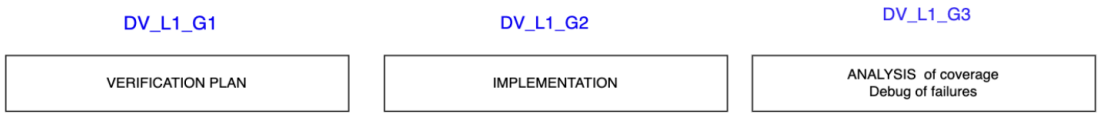

Figure 1. Level 1 goals for Verification

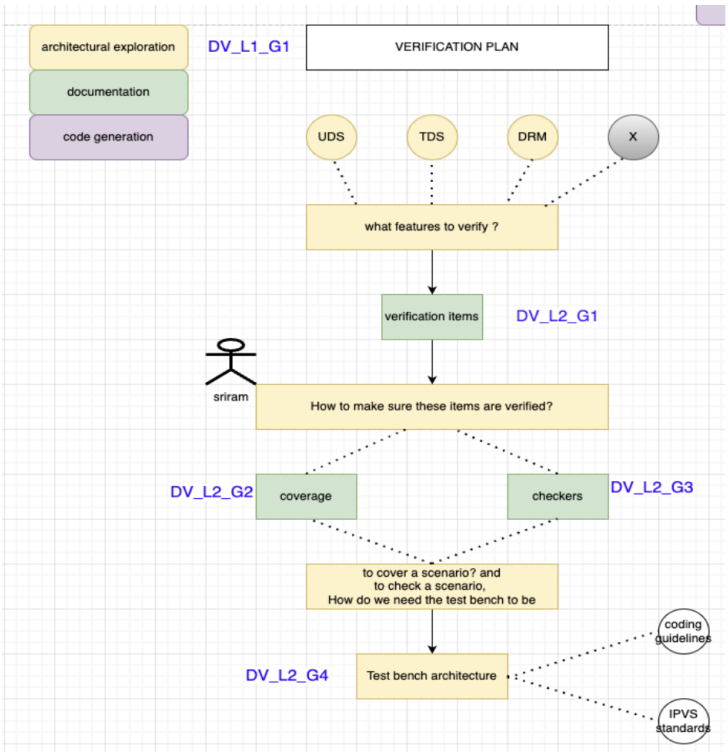

Figure 2. Sub goals in Verification Plan

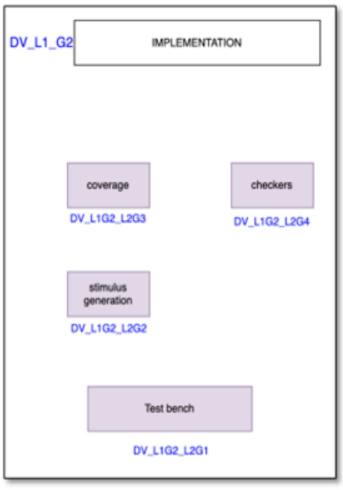

Figure 3. Sub goals within Implementation

C. Gen-ai exploration for a sub-goal

Note 1: For the scope of this paper, out of various possible sub-goals, a few are selected, and the exploration related to those goals is presented. The sub-goals are selected across level 1 goals, such that they emphasize the challenges unique to verification when it comes to using Gen-AI to a variety of sub-goals.

Note 2: While the long-term vision of the GAIL-V framework is to encourage collaboration and contributions from the verification community, the current work focuses on presenting concepts, methodologies, and evaluation strategies. Any future decision to release code or resources as open source will depend on organizational approvals. For this paper, we limit the scope to discussing ideas, challenges, and mock-ups rather than publishing full implementations.

In the following sections, we go through a few sub-goals where we explore the usage of Gen-AI. Please note that each of the sub-goals themselves demands a separate paper to explain in detail. We did our best to present the idea within the limits of this paper. The first sub-goal will be explored in detail explaining the trade-offs and challenges, walk through the methodology in detail and the rest at a high-level sufficient to bring out the idea.

A. Sub-goal overview

Reviews are the final stage of the verification process and are typically conducted at the milestone level. In the hierarchy, this sub-goal falls under DV_L1G4_L2G1 i.e. L1G4 is the main goal in level 1 and L1G4_L2G1 states it as the goal-1 within this main goal of Reviews. Figure 4 captures the essence of the process where there would be a set of questions (this differs by names as checklist etc.) to be answered by the DV owner and reviewed by an auditor (peer review, or external review). Figure 5 shows the workflow of an idea Gen-AI based auditor where Gen-AI would replace the manual auditor and perform the reviews marking if compliance is met.

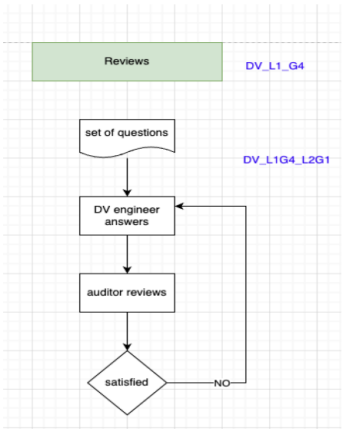

Figure 4: review workflow

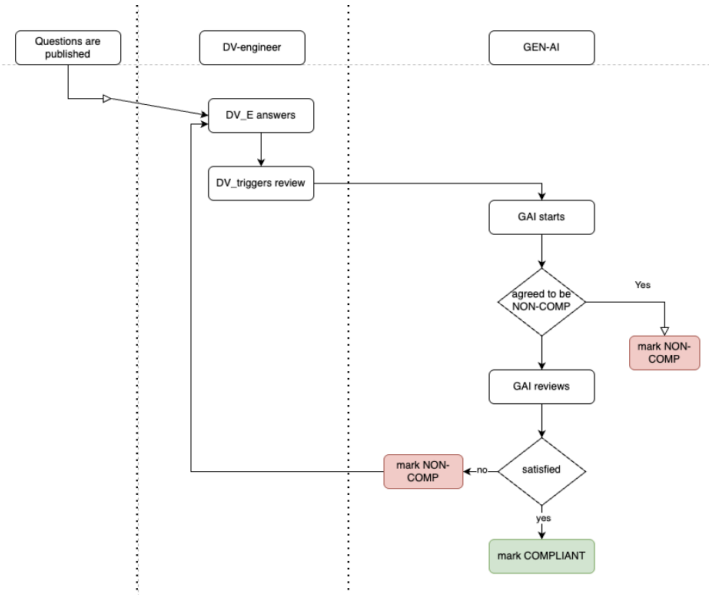

Figure 5: review ideal workflow using gen-ai

B. Assessment to leverage gen-ai

We begin by asking: Would Gen-AI be a suitable solution for this sub-goal? A rough estimate shows that it could be a good solution because the review process by a human auditor involves

1. Understanding the large number of questions that needs general DV domain knowledge
2. Diligence in going through the answers and validating the answers
    a. Some could be trivial such as ‘Is the naming convention correct’ and some could be complex such as ‘is coverage sampled after a transaction is deemed correct by scoreboard. There is a wide range and variety of questions.

III. Sub-goal : Verification sign-off review for a milestone: DV_L1G4_L2G1

b. Checking various sources of information for correctness of answers such as verification plans, design specifications, bug trackers, code database, coverage reports

In our experience, this is not the most favorite of tasks for verification engineers and the process is time consuming with multiple iterations of the reviews going back and forth between engineer and auditor. Gen-ai seems like a good idea for this sub-goal because LLMs (large language models) which are the core of gen-ai are good at consuming large amounts of data and reasoning from it.

C. Ideal workflow using gen-ai

Figure 5, shows what would be the ideal set up for this sub-goal. Once the questions are published, the DV engineer answers them which triggers a review by Gen-AI. Based on the answers and input sources, it acts as an auditor and marks which questions and complaints and not. One human (the auditor) is out of the loop in this case. How the questions are published and how the reviews are triggered are not of significance here as the paper is to generalize the idea and not the details. For instance, the questions could be in a simple excel sheet and a python script to read and trigger review, or it could be a complex system fully automated using a web interface.

D. Benchmarking : method

Can we go ahead and feed the questions and answers to gen-ai and be confident about the audit process? How can we be sure that it did the job correctly? One extreme is to accept that gen-ai did a robust audit and live with it, or another extreme is to again check all its output using a human auditor which defeats the purpose of gen-ai altogether. This is why we need benchmarking. For example, consider a simple task where Gen-AI is asked to solve a contrived puzzle of finding the shortest path from point A to point Z, and it provides the solution A → C → S → Z. To assess correctness, we must already know the answer, test Gen-AI across multiple such puzzles, and compare its output against the known solutions (labels).

How do we create labels for this sub-goal of verification reviews? The questions vary from team to team, domain to domain, and there is no definitive answer. We came up with a solution to create synthetic verification plans with planted mistakes and testing Gen-AI response to them.

E. Benchmarking : dataset

Figure 6 shows the benchmark creation process for this sub-goal. Let's walk through a contrived example of a review question, call this main question M1- “Cover points should have a naming convention to indicate their priority, and they should have a mapping to a design feature. Cover point bins should have a min and max bin to hit corners, and they should be in a hierarchical coverage model”. We write a prompt to first split M1 into sub questions which are pointed such as S1-do the cover points follow naming convention, S2: is it mapped to design, S3: is there a max bin, S4: is there a min bin, S5: is the cover group in a hierarchy of coverage model.

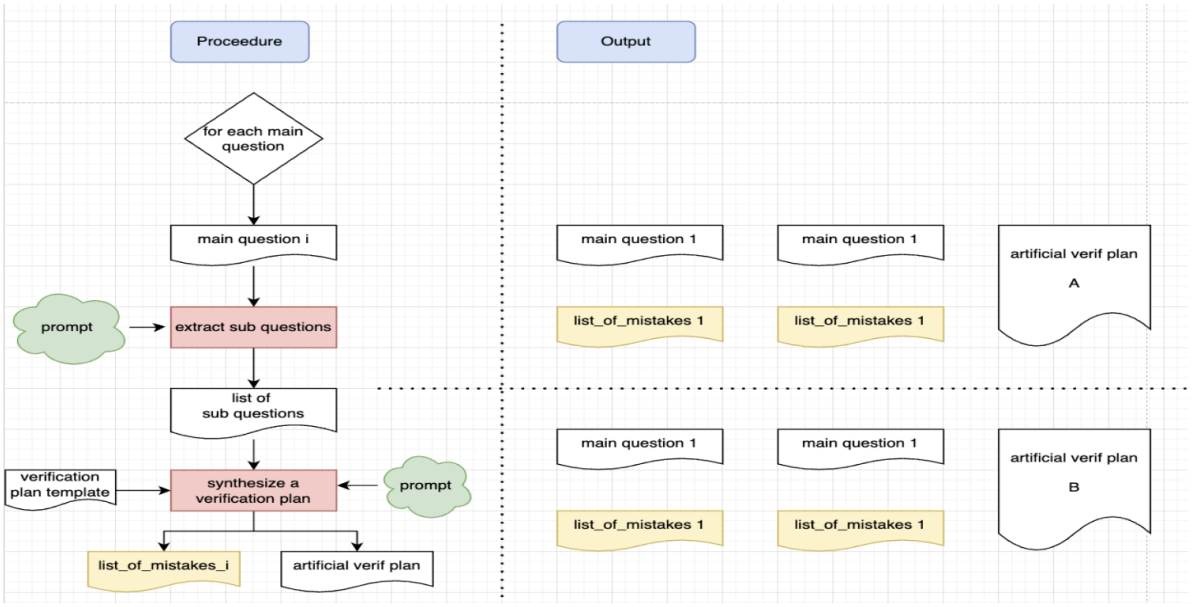

Figure 6: Benchmarking method and dataset generation for the sub-goal reviews

Figure 6 shows how we feed a main question M1 along with sub questions S1 to S5 and a verification plan template and write a prompt to synthesize a 'artificial verification plan' which adheres to the template and an artificial cover point code and adds some synthesized content. Importantly the prompt asks to plant 'mistakes' which will act as our labels. The output of this exercise is an 'artificial verification plan' along with the list of questions and labels (known mistakes). We can generate many such examples, though the exact scale is subjective.

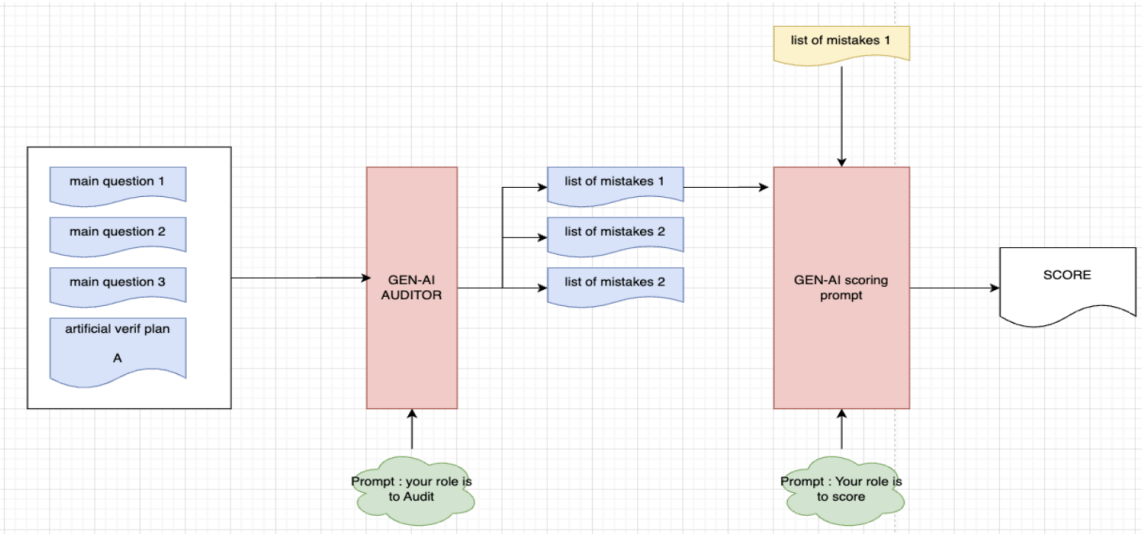

Figure 7: Benchmarking evaluation and scoring

F. Benchmarking : Scoring and Evaluation metrics

Figure 7 shows evaluation of Gen-AI as an auditor. For simplicity consider Gen-AI as an auditor is a simple prompt that checks if all planted mistakes are found. We feed the artificial verification plans and get the review output. We then use a scoring prompt to compare this output with the labels.

Sample prompt for illustration of scoring: "Your goal is to score how generative AI auditor performed against the benchmark. You'll be given two inputs one as a reference or benchmark which has the complaint review against each sub question the second will be the auditors review. You must find out how many auditors' reviews were correct for

example if the benchmark said question is non-compliant and auditor reviews compliant it is a mismatch and vice versa. We want to see how many we caught correctly how many we caught incorrectly and the percentages et cetera data".

Figure 7 and Figure 8 show sample scores for a data set and evaluation metrics.

Performance Summary

Metric	Value
✅ True Positives (TP)	27
✅ True Negatives (TN)	5
❌ False Positives (FP)	6
❌ False Negatives (FN)	9

Figure 7: Benchmarking scores

Evaluation Metrics

- **Accuracy: 68%**
  - Overall proportion of correctly identified sub-questions (compliant or non-compliant).
- **Precision: 82%**
  - Of all items the AI flagged as non-compliant, 82% were correct.
- **Recall: 75%**
  - The AI caught 75% of all true non-compliant items.
- **F1 Score: 78%**
  - Harmonic mean of precision and recall — a good balance measure of overall performance.

Figure 8: Benchmarking evaluation metric

G. Benchmarking : Regression over different models

Figure 9 below is a mock-up imitation of the website which hosts the regression results. For instance, DV_L1G2_L2G4 shows how various models perform in terms of accuracy for the subgoal of 'checker definition'

GenAI Scorecard for Verification Sub-Goals

Comparing Model Performance to Human Baseline

Sub-goal	Human Baseline	GPT-4o	o3	o4-mini-high	GPT-4.5
DV_L1_G1 – Verification Planning					
DV_L1G1_L2G1: Feature Extraction	100%	95%	92%	78%	90%
DV_L1G1_L2G4: Testbench Architecture	100%	82%	60%	58%	77%
DV_L1_G2 – Implementation					
DV_L1G2_L2G2: Stimulus Generation	100%	76%	70%	54%	68%
DV_L1G2_L2G4: Checker Definition	100%	88%	75%	60%	84%
DV_L1_G3 – Coverage & Debug					
DV_L1G3_L2G1: Root Cause Analysis	100%	91%	73%	65%	85%
DV_L1G3_L2G2: Code Coverage Analysis	100%	80%	75%	62%	78%

Note: Scores are illustrative and represent potential capability gaps/fits in GenAI models for different tasks.

Figure 9: mock-up of scorecard for illustration

IV. Two more subgoals presented at a high-level

In the following sections we cover 2 more sub-goals; the idea is explained at a high-level to limit the scope of the paper. The emphasis is on the wide variety of the type of tasks in verification processes that makes it challenging to leverage Gen-AI and the need to breakdown into sub-goals and to also emphasize out of the box approaches to benchmark, dataset and evaluation.

A. Sub-goal: Stimulus generation to hit a missing coverpoint bin in a bound package

Overview: Most test benches use VIPs (verification IPs) which are imported as packages. As an example, consider AXI protocol, which is a widely used protocol that may be used in verification by importing the package. The VIP contains code and related documentation. Let's say we utilize VIP and enable functional coverage, and we

want to hit a cover point which is to hit a burst transaction of 4 beats. Figure 10 shows the ideal use case using Gen-AI that we would like where Gen-AI provides the necessary sequence which will then be used in Figure 11

Assessment - Human engineers will have to understand the protocol by going through the specification and figuring out what scenario needs to be generated to hit the cover point. The engineer then reads through the code documentation and sequences the scenario to implement it.. Depending on the complexity of the protocol and the quality of documentation, this can be a tedious and time-consuming task. Gen-ai may be a good fit as it can sift through huge documentation and bound code (hence we signify a package) to find out what needs to be done.

Benchmarking: With prototype experiments and feedback, our observation is that for this sub-goal the best way to measure performance is to break down the Gen-AI task into stages as follows: 1. Does Gen-AI understand the cover point 2. Does Gen-AI understand the transaction required in terms of protocol 3. Does Gen-AI generate code that is sensible 4. Is the sequence compile-clean 5. Is the sequence runnable without errors 6. Does the sequence hit the cover point: 6 stages.

Dataset: create a test bench manually which is compile clean and run clean by connecting VIP in back-to-back transmitter and receiver combination. We then generate coverage reports with cover point holes to which we know the exact sequences of code. For each of these holes human engineers write answers to the 6 stages. The need to break down into these stages is that from our experiments stage 1,2,3 show good results with gen-ai and 4,5 seem to be buggy and difficult to debug Also, these stages help in debugging where gen-ai performs poorly and use results from earlier stages that seem to give better output. For example, rather than asking for final sequence that we can copy paste; which doesn't seem to work well; stages 1,2,3 would explain how to hit the cover point and from there if the final sequence doesn't work we can debug if the logic derived in 1,2,3 was the cause or the coding generation part is performing poorly in which case human engineer can take control from stage 4 by writing code aided by explanation from stages 1,2 and 3.
Evaluation: Rather than scoring based on the ability of final sequence to hit the cover point, for this sub-goal we recommend scoring based on the stages 1 point for passing stage 1 and 1 point for stage 2 so on for a total of 6 points.

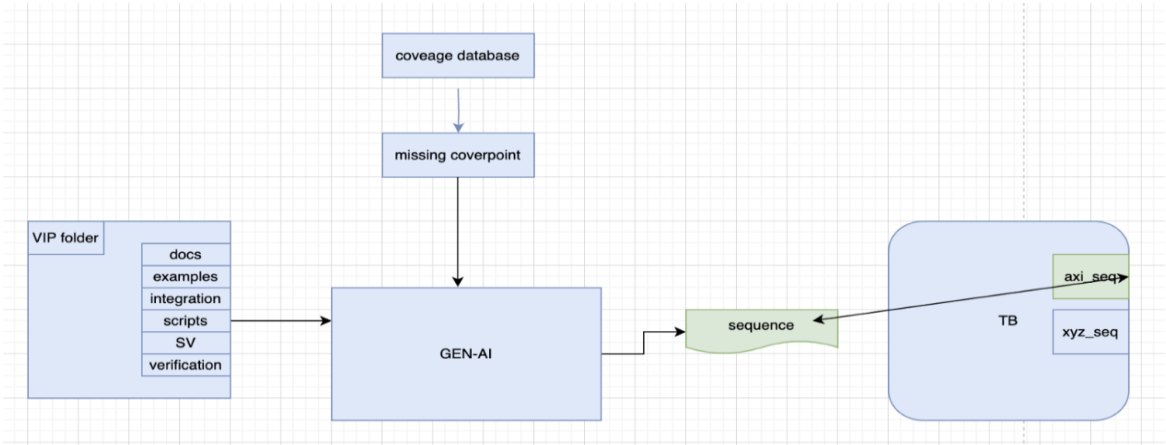

Figure 10: overview of sub-goal: hitting cover point of a bound VIP package

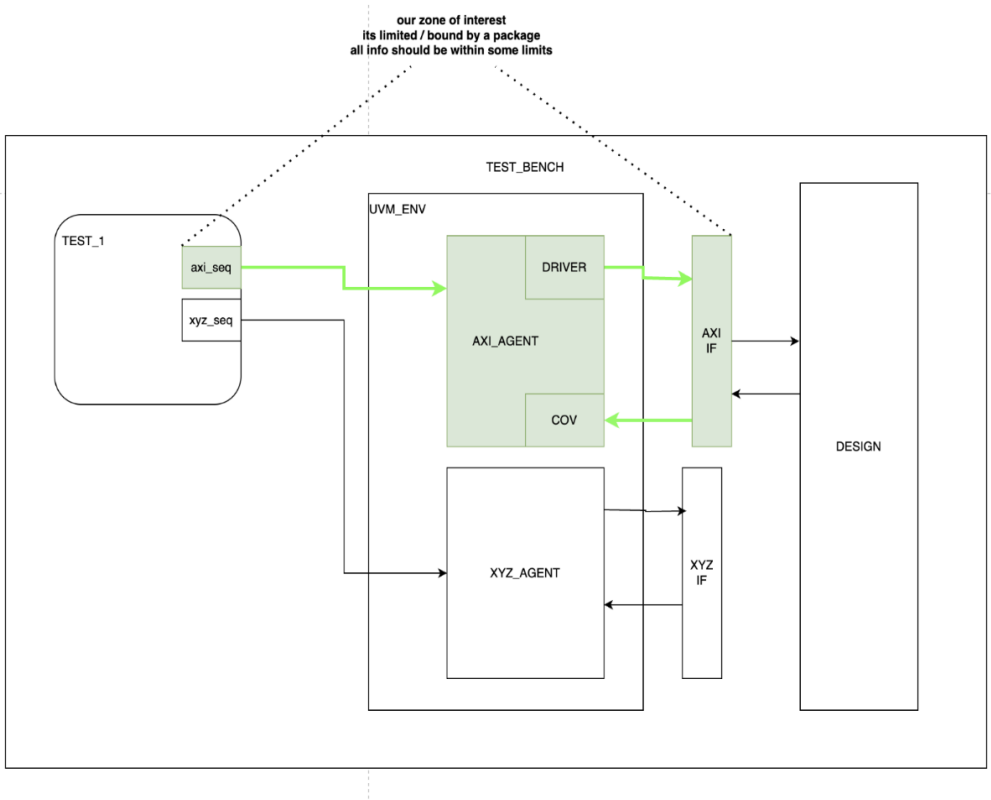

Figure 11: overview of sub-goal: hitting cover point of a bound VIP package

B. Sub-goal: Verification requirements extractor

Overview: A verification plan is the most important document that acts as a strong foundation for robust verification. A human engineer will have to understand the design under test in context and depending on the scope of verification, come up with a robust yet achievable plan. Core to the plan is the verification requirements, which is a list of items or features to verify. This often requires broad knowledge of protocols and design and sometimes deep design details demanding the need to go through huge amounts of design specification, architecture, micro architecture, protocol specifications etc.

Assessment: With the timeline of tape outs shrinking and project timelines shrinking respectively, design specifications often go through many changes and verification teams must keep up with updating verification requirements. With the complexity of modern designs, the amount of documentation and changes to keep track of is overwhelming. Gen-ai seems a very good fit for this sub-goal for its ability to parse many contents to find information quickly and its ability to reason.

Benchmarking: As DUTs can be at an IP level, sub-system level, or soc-level, there is a trade-off of depth in scope of verification at each level. For this sub-goal, our recommendation is to have a different set of sample designs and related documentation for which a human engineer extracts a list of verification requirements. This would be compared against the verification requirements extracted by gen-ai Evaluation : Compared to other sub-goals, the evaluation for this goal is simpler where we check if 1. Are all goals extracted by human engineers extracted by Gen-AI 2. Additional requirements are validated to check their relevance and given extra points 3. missing requirements are penalized.

Dataset: the data set generation for this goal is a huge challenge to synthesize but there is a huge amount of data from designs and verifications from past or current projects that can be leveraged though. The size of the designs can also vary. For example, a simple FIFO could have verification requirements such as verify fifo-full, fifo-empty conditions, verify back pressure etc. whereas at a memory sub-system level there could be requirement such as verify that no data is lost if power domain is changed.

V. Conclusion

This paper presents a structured methodology to evaluate and adopt Generative AI (Gen-AI) within the hardware verification process. As emphasized with the use of 3 sub-goals in above sections, there are challenges unique to the domain of design verification when it comes to leveraging Gen-AI. The wide variety of tasks that range across planning, architecting, debugging etc., make it challenging to understand and measure the performance and consistency of Gen-AI in aiding verification. Rather than ad hoc prompting and expecting one shot complete solution to the process of verification, the use of hierarchical decomposition to break down the verification process into modularized sub-goals and then tackling the sub-goals within a limited scope helps in identifying where we can leverage Gen-AI and where we currently cannot. Starting with the ideal use-case in mind, we assess from our experiments and the sub-goal in context, the applicability of Gen-AI as a useful tool. Benchmarking provides a reliable way to quantify the reach and performance of Gen-AI and as shown we need to come up with clever ways to create datasets to score and evaluate the benchmarks which are unique to each sub-goal. Regressing over various models and prompts helps us identify the best models for the specific goals as there seems to be no one model–and-one-prompt that fits all problems.

The biggest challenge we found is coming up with datasets for benchmarking. Though benchmarking itself needs creative solutions according to the sub-goal, data sets are the key to ensure that we can reliably use Gen-AI for the sub-goals in production. Unlike traditional machine learning data, the dataset for verification varies across sub-goals and the lack of open datasets due to almost all designs and plans being proprietary makes it very difficult. Synthetic data acts as a minor remedy, but a large body of open-source design could help with the data set generation. Also, an open framework would help in the dataset creation and evaluation process.

Future work includes expanding the labeling system, quantifying productivity gains across additional IPs, and building a shared repository of case studies to benchmark Gen-AI effectiveness. This framework thus aims to democratize the use of Gen-AI in verification, making it repeatable, measurable, and explainable. The proposed deliverables of this endeavor are a codified framework for a common language among verification engineers to leverage Gen-AI. For sub-goals, documented solutions, implementation, results, and evaluation criteria. All of this (subject to internal approval) may be hosted on open source sharing platform as mentioned in section II.C note 2. While the results presented here are preliminary, they establish a foundation for community-driven benchmarking and shared best practices in applying Gen-AI to verification.