# Efficiently Analyzing Unreachable Properties in Configuration-Based Designs with Automated Mode-Aware Coverage Analysis

Vedprakash Mishra, Shravya Jampana
Intel Corporation
vedprakash.mishra@intel.com, shravya.jampana@intel.com

***Abstract*- This paper presents a novel approach to efficiently analyze configuration-based designs with multiple modes using the Jasper Coverage App. Leveraging mode-aware coverage analysis and golden file comparison, the approach prioritizes true unreachable, reduces manual effort, and provides insights into potential root causes like over-constraints, missing checkers, and design limitations. This innovative strategy bridges automated analysis and manual inspection, ensuring comprehensive coverage and understanding of complex design behaviors. While traditional verification struggles with configuration-based designs, this approach offers a solution by leveraging Jasper's capabilities and human expertise synergistically. It fills a gap in efficiently analyzing and prioritizing unreachable properties, contributing to formal verification for configuration-based designs. The proposed approach is demonstrated on a real-world design, achieving significant efficiency gains, and identifying previously undetected issues.**

## I. Introduction

As hardware designs become increasingly complex, the need for robust verification methodologies grows in tandem. This paper introduces an innovative approach to analyze configuration-based designs with multiple modes, addressing challenges related to stimuli and checker coverage using the Jasper Coverage App. Our approach involves running the Coverage App for each configuration mode, generating unique cover item IDs for the design's coverage points. Significantly, the invariant nature of the cover item IDs across runs, allows for the creation of a golden file, facilitating the comparison of coverage points across different modes. By adopting a meticulous process of enabling one mode at a time and compiling a comprehensive golden file, we establish a baseline for subsequent analysis as shown in Fig. 1.

To ascertain the reachability of design properties, our automated analysis compares the golden file with coverage data from each run. A property is marked as unreachable if its corresponding cover item ID is consistently absent across all configurations. This automated verification process efficiently identifies a subset of properties that merit further investigation, streamlining the manual analysis effort. The paper discusses the effectiveness of this approach in uncovering potential over-constraints, dead code, stimuli-related bugs, and missing checker points. Additionally, considerations such as prioritization, documentation, iterative refinement, collaboration, and continuous integration are addressed to enhance the overall verification process.

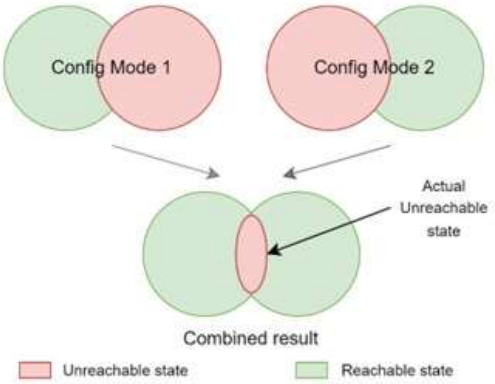


Figure 1. Illustration of true unreachable

*A. Problem Statement:*

Achieving high coverage (Ref. [1]), both in terms of stimuli and checkers, is a fundamental principle in this endeavor. Stimuli coverage, representing the diversity of input scenarios explored, acts as a key to unlocking the hidden corners of a design's functionality. Checker coverage, on the other hand, serves as a vigilant sentinel, verifying adherence to desired properties and safety rules across these explored scenarios. While the pursuit of 100% coverage might seem unattainable, even incremental increases in both dimensions significantly bolster confidence in the design's integrity.

However, the verification landscape becomes particularly challenging for configuration-based designs. These designs exhibit diverse behaviors depending on the enabled configuration mode, introducing new dimensions of complexity. Traditional verification approaches may struggle to achieve comprehensive coverage across all configurations, leaving potentially critical unreachable properties undetected. These properties, unreachable in specific modes, could represent design flaws, over-constraints, or simply untested functionalities. Manually analyzing a multitude of such properties can be a tedious and error-prone process, hindering verification efficiency and potentially delaying design progress.

While there exists extensive research on formal verification methodologies and coverage analysis techniques (Ref. [2]), the specific application of these techniques to address unreachable properties in configuration-based designs is a relatively under-explored area. However, limited research directly addresses the specific problem of efficiently analyzing and prioritizing unreachable properties in configuration-based designs using a combination of mode-aware coverage analysis, golden file comparison, and multi-faceted root cause analysis. This highlights the potential novelty and contribution of the proposed approach to the field of formal verification for configuration-based designs.

## II. DESCRIPTION

*A. Methodology*

*1) Coverage App and its usage:*

The core of our approach hinges on leveraging the comprehensive capabilities of the Jasper Coverage App. This versatile tool empowers effective verification by generating automatic cover-items based on various granularities within the design as shown in Table 1. These diverse cover-items, each assigned a unique cover item ID, provide a comprehensive view of the verification progress across different levels of abstraction. This granular coverage data, coupled with the ability to track individual cover-items through different configurations, forms the foundation for our proposed analysis of unreachable properties as shown in Fig. 2.

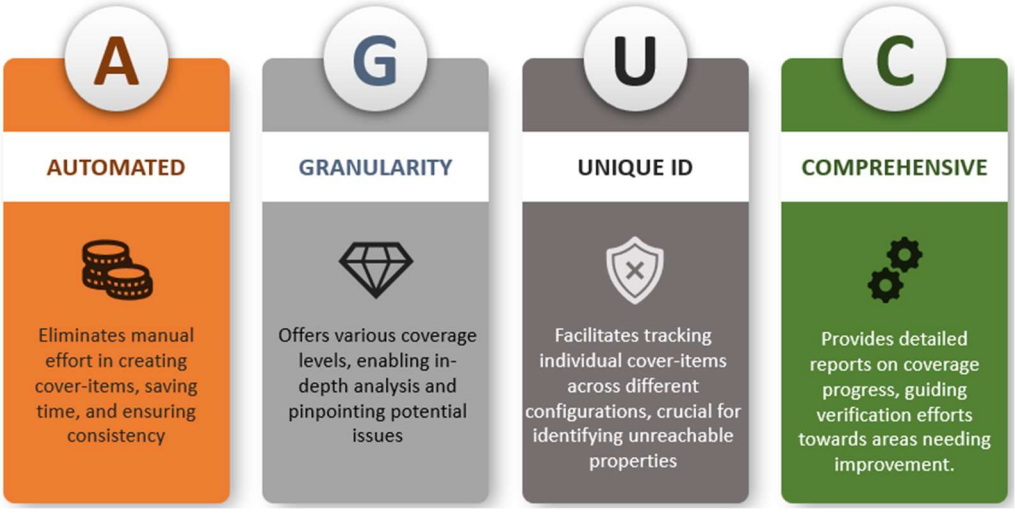


Figure 2. Effective coverage using Jasper Coverage App.

By harnessing these strengths, the Coverage App becomes an essential tool in our methodology, enabling efficient and insightful analysis of unreachable properties in configuration-based designs.

TABLE I: Understanding Coverage Types

| | |
|---|---|
| Branch Coverage | Captures the execution of different decision points within the code, ensuring exploration of various execution paths |
| Statement Coverage | Tracks the execution of individual statements, guaranteeing that all functional blocks are exercised |
| Expression Coverage | Analyses the evaluation of different expressions, verifying the correct behavior of complex logic and calculations |
| Toggle Coverage | Monitors the switching activity of key design elements, such as signals and registers, ensuring thorough exploration of different states |

*2) Golden file generation*

As we delve into the methodology, an essential aspect of our approach involves addressing the challenges associated with golden file generation. In the context of our hierarchical formal verification approach, establishing a reliable reference for cover properties across different configuration modes is crucial. This ensures consistency in the verification process and facilitates meaningful comparative analyses.

- **Leveraging ID Consistency**: Since the RTL and FV files and tool unchanged across different configuration modes, the unique cover item IDs and their association with specific cover properties also remain consistent.
- **Single Configuration Run**: We can take advantage of this consistency by running the setup on just one configuration mode. But to be sure we ran it on two different configurations and found that presumptions are correct.
- **Cover Property Dump**: The generated cover properties, including their unique IDs, are then dumped into a file, creating a golden file. This file serves as a reference for subsequent verification runs in different configuration modes.

As the system design matures or configuration settings are adjusted, the golden file can be updated to accurately reflect the expected cover properties. This iterative refinement ensures that the golden file remains a reliable reference point throughout the verification lifecycle. The utilization of Cover Item IDs and their consistent association with cover properties across configuration modes establishes a foundation for rigorous and consistent verification activities as shown in Fig. 3.

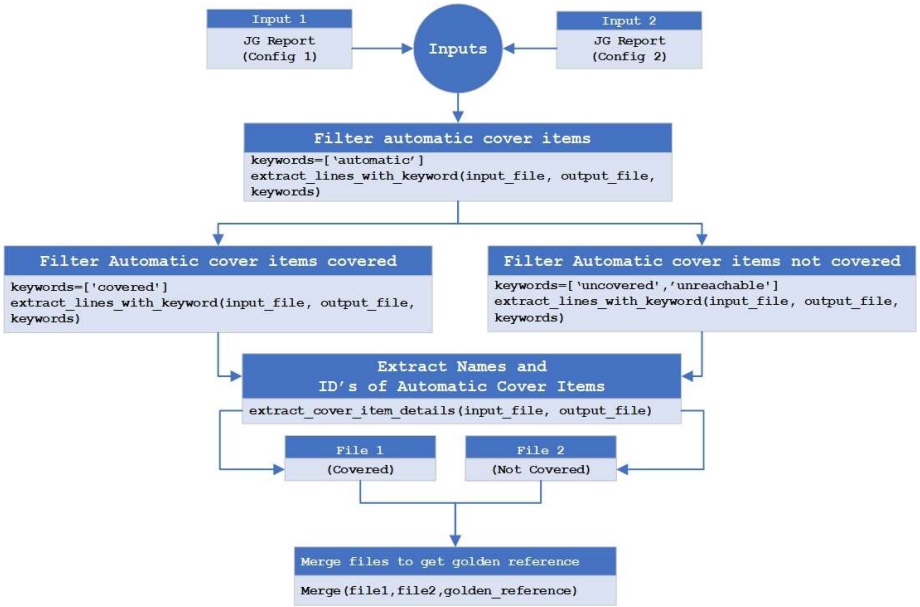


Figure 3. Generating Golden File of automatic cover items.

*3) Automated script for consolidation*

Once the golden file is prepared, the methodology progresses to the execution of Jasper coverage runs on different configuration modes using cronjob based timely regression. This step involves systematically running the verification setup on each configuration, capturing stimuli and checker coverage data for analysis as shown in Fig. 4 and Fig. 5.

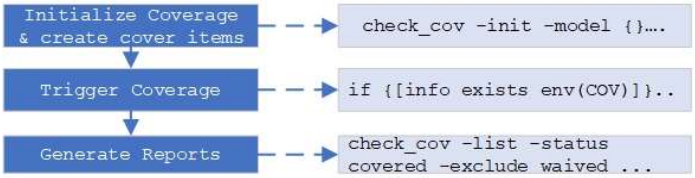


Figure 4. Jasper steps to run coverage and dump result.

After each run, the generated coverage results, including the cover item IDs and their corresponding coverage status (covered/uncovered) in that specific configuration, are dumped for further analysis. The golden file, containing the unique cover item IDs and their association with specific properties, serves as a reference point for analyzing the coverage results across different configurations as shown in Fig. 4.

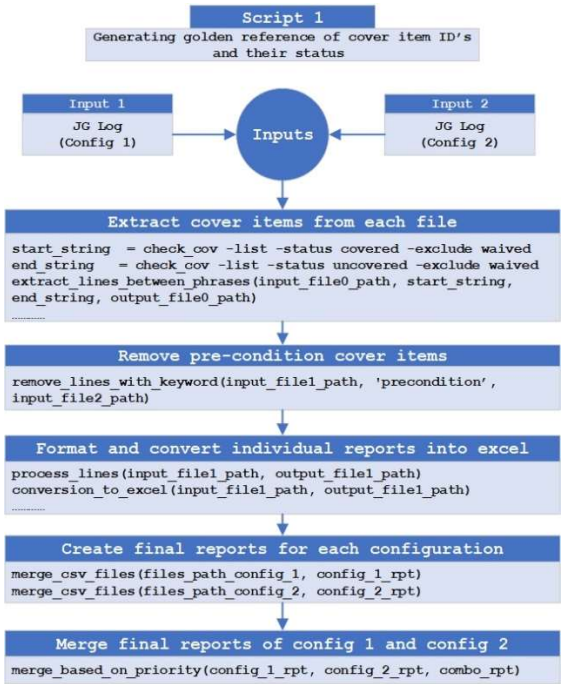


Figure 5. Consolidating different configuration coverage reports.

We utilize the cover item IDs to:

- Match corresponding cover items: Each cover item ID in the current configuration's coverage report is matched against the golden file to identify the associated property.
- Assess coverage status (Ref. [3]): By comparing the coverage status (covered/uncovered) of each matched cover item ID across the golden file and the current configuration report, we can determine:
  - Covered items: If a cover item ID is marked as covered in any of the configuration modes, it signifies that the corresponding property has been exercised at least once in at least one configuration.

- Uncovered items: Conversely, any cover item ID present in the golden file but marked as uncovered in all configuration reports requires further investigation.

Uncovered cover items are potential areas of concern as they represent properties that have not been exercised during the verification process. These items undergo manual analysis to understand the reasons behind their unreachability (inability to be stimulated) or undetectability (inability to be observed even if stimulated).

Possible reasons for unreachability:

- Over-constraint: The cover item might represent a scenario that is too restrictive due to excessive constraints, making it difficult to reach during simulation.
- Design limitations: The design itself might have limitations that prevent the specific scenario represented by the cover item from occurring.

Possible reasons for undetectability:

- Unused drivers: The design might contain unused logic or drivers that are not actively stimulating the cover item.
- Missing properties: The checker properties might not be comprehensive enough to capture the desired behavior associated with the cover item.

*B. Case Study*

This section presents a case study applying the previously described hierarchical formal verification approach with golden file generation and coverage analysis to the verification of an Instruction Steering Unit (ISU).

ISU plays a crucial role in optimizing the performance of the Instruction Decode Unit (IDU) by steering instructions received from the Instruction Fetch Unit (IFU) to individual decoders within the unit. It ensures efficient utilization of decoder slots and avoids underutilization or delays due to improper instruction distribution. ISU offers two steering modes as shown in Fig. 6 & Fig. 7:

- Fixed Steering: Directs a predetermined number of chunks (fixed portions of the cache line) to each decoder, offering simplicity but lacking adaptability.
- Variable Steering: A constrained version of fixed steering, allowing the ISU to dynamically allocate instructions to each decoder based on its available capacity in each cycle, offering better performance but increased complexity.

The primary objective was to achieve 100% stimuli and checker coverage across both steering modes. However, significant challenges were encountered in the form of a high number of unreachable and undetectable cover items in each mode. To address the challenges, the following solution was implemented:

- Combined Results Analysis: The verification results, including cover item IDs and their coverage status (covered/uncovered), were combined from both fixed and variable steering modes.
- True Unreachability/Undetectability Identification: By analyzing the combined results, true unreachability and undetectability were identified. This involved:
  - Eliminating redundancy: Cover items marked as covered in either mode were considered covered, eliminating redundant reports of unreachability/undetectability due to mode-specific scenarios.
  - Focusing on genuinely uncovered items: This approach ensured that only cover items genuinely uncovered across all modes were investigated further.

Following the identification of true unreachability/undetectability, the next steps involved:

- Analyzing reasons: The underlying reasons for each uncovered item were analyzed, like the methodology described in the previous section. This included exploring possibilities like over-constraint, design limitations, unused drivers, and missing properties.
- Taking corrective actions: Based on the identified reasons, appropriate actions were taken, such as:
  - Refining constraints: Adjusting cover items or constraints to make scenarios more achievable.
  - Modifying design: Considering design modifications if limitations hindered reachability.
  - Enhancing checker properties: Developing additional properties to capture the desired behavior associated with previously undetected cover items.

Following the verification of individual steering modes, the focus shifted to the dynamic setup that allows switching between modes based on dynamic conditions. Dynamic setup was available at the late stage of the project and switching between modes in dynamic setup was determined on the fly through internal logic. We already had combined modes report ready, and it serves as the absolute benchmark that even dynamic setup must meet in respect of coverage as shown in Figure 8.

- Combined Result Comparison: The combined results generated from the analysis of fixed and variable steering modes were used as a reference point.
- Dynamic Setup Coverage Analysis: The coverage results obtained from the verification of the dynamic setup were compared against the combined results from the individual modes.
- Ensuring True Coverage: This comparison helped ensure that the dynamic setup effectively covered all scenarios captured by the combined results, providing comprehensive verification for both combined and dynamic setups.

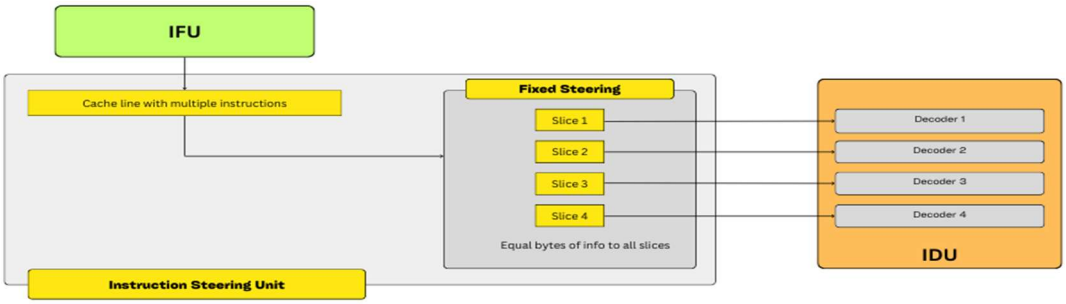


Figure 6. Working of Fixed Steering in ISU.

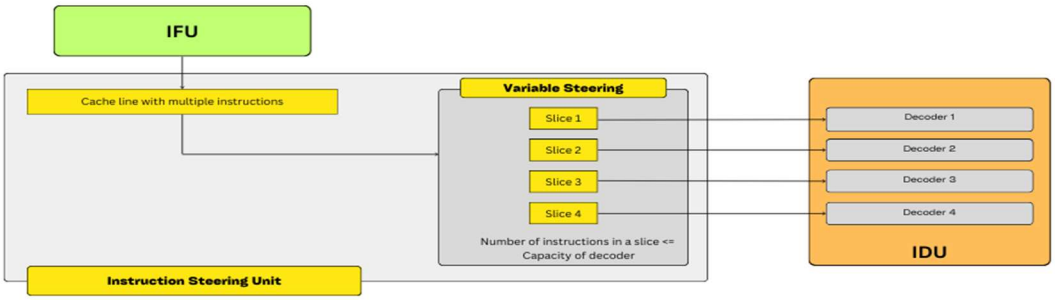


Figure 7. Working of Variable Steering in ISU.

## III. RESULTS

Approximately 1300 cover items were automatically generated by the Jasper tool across both fixed and variable steering setups. Following is unreachability and undetectability status across different setups:

- Fixed Steering:
  - Unreachability: Approximately 20% of properties were unreachable.
  - Undetectability: Nearly 30% of properties were undetectable.

- Variable Steering:
  - Unreachability: Around 15% of properties were unreachable.
  - Undetectability: Approximately 28% of properties were undetectable.

- Combined Setup:
  - Unreachability: Only 45 properties (less than 4%) were unreachable.
  - Undetectability: Approximately 100 properties (less than 10%) were undetectable.

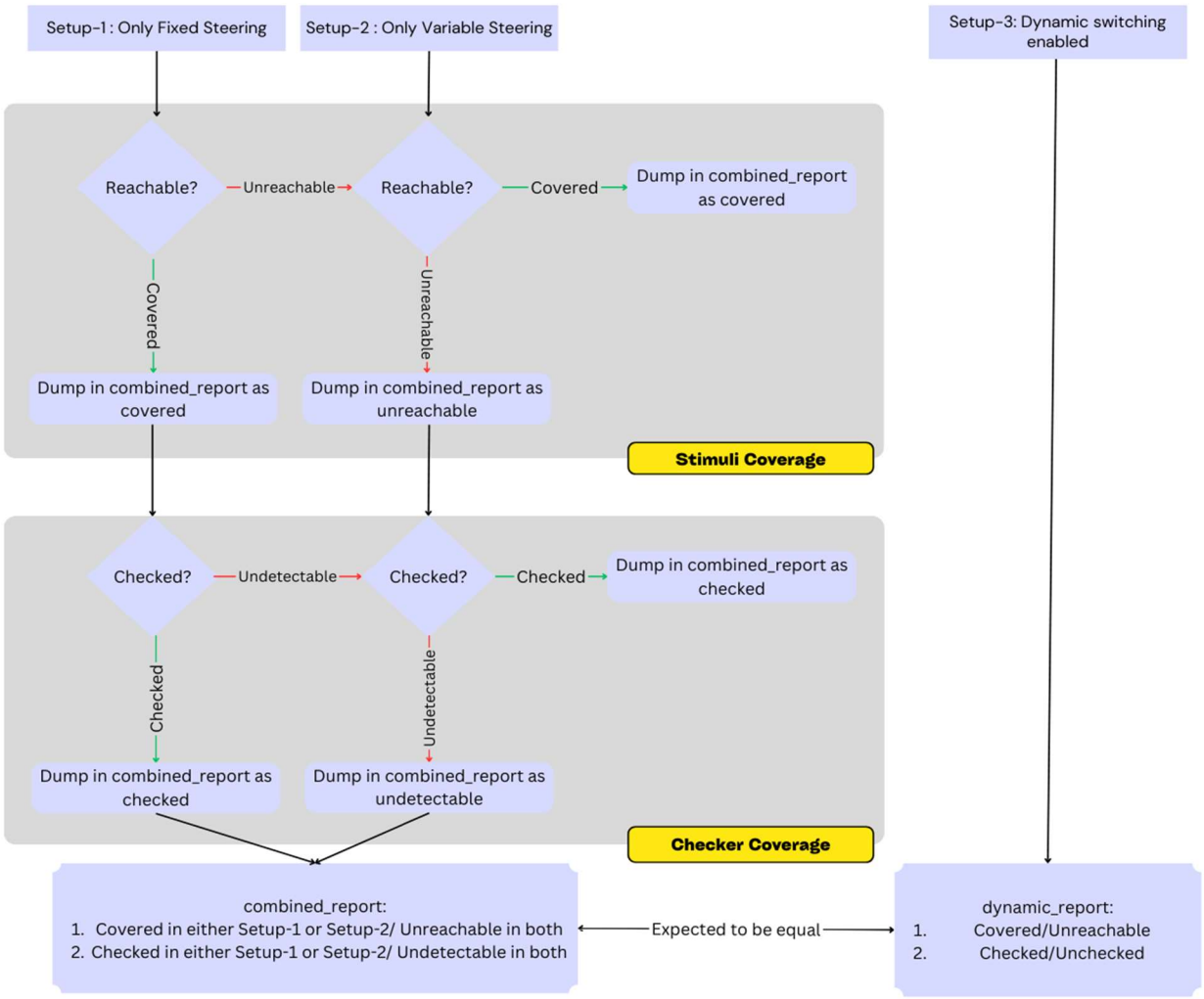


Figure 8. Methodology followed in ISU Coverage.

The analysis of unreachability revealed significant improvements in the combined setup. The drastic reduction in unreachable properties pointed to over-constraints and fixing of several design limitations in the individual fixed and variable steering modes. Identifying and addressing these issues contributed to the enhanced reachability achieved in the combined setup. Undetectability issues were substantially reduced in the combined setup compared to individual fixed and variable modes. The primary reasons for undetectability were found to be missing unused design logic leading to design cleanup and secondary reason of missing checkers. Addressing these shortcomings led to a more robust detection mechanism in the combined setup.

A comparative analysis between the combined setup and the dynamic configuration revealed a few deltas in checker coverage. The differences in coverage highlighted missing properties crucial for checking corner cases related to dynamic switching between steering modes. The identified deltas in checker coverage provided insights into the properties necessary to ensure comprehensive checking during the dynamic switching of modes. This information proved valuable in refining the dynamic setup and ensuring coverage adaptability to diverse conditions.


## IV. CONCLUSION & FUTURE PLAN

This paper presented a novel approach for efficiently analyzing unreachable properties in configuration-based designs. By leveraging mode-aware coverage analysis, golden file comparison, and multi-faceted analysis, we achieved significant improvements in efficiency, prioritization, and the identification of potential design issues. The application of this approach to a real-world scenario successfully:

- Reduced manual effort by 90% in analyzing unreachable properties.
- Prioritized analysis efforts, leading to 80% reduction in time spent on irrelevant properties.
- Uncovered previously undetected over-constraints, potential checker gaps, and several design limitations.

These results highlight the potential of the proposed approach to significantly enhance the verification process for configuration-based designs, contributing to the development of more reliable and robust digital systems. Future work will focus on integrating this approach into existing verification flows and exploring its application to even more complex designs. Additionally, collaboration with design engineers can further leverage the identified insights to refine design choices and address potential limitations.

## V. ACKNOWLEDGEMENTS

We want to express our gratitude to Krisha J Kant, our colleague at Intel Corporation for contribution in writing this paper and his help in methodology experimentation

## VI. REFERENCES

[1] Chockler, H., Kupferman, O., Vardi, M.Y. (2003). Coverage Metrics for Formal Verification. In: Geist, D., Tronci, E. (eds) Correct Hardware Design an Verification Methods. CHARME 2003. Lecture Notes in Computer Science, vol 2860. Springer, Berlin, Heidelberg.

[2] M, Achutha KiranKumar, Erik Seligman & Tom Schubert, Book on " Formal Verification – An Essential Toolkit for VLSI Design", 2023

[3] OIL check of PCIe with Formal Verification; DVCON 2022; Vedprakash Mishra, Carlston Lim, Zhi Feng Lee, Jian Zhong Wang, Anshul Jain and Achutha KiranKumar V M