Methodology for Efficient Fault Injection using Random Sampling

Vedant Garg, Synopsys USA (gargv@synopsys.com)
Prashantkumar S. Sonavane, Associate Technical Director, Samsung India
(prashant.s2@samsung.com)

A. Abstract

Fault injection is a critical technique for assessing system reliability and resilience under adverse conditions and to test Random Faults. One practice to achieve or maintain a safe state is running an exhaustive fault injection to test the effectiveness of a safety architecture's ability to detect faults or control failures. However, exhaustive fault injection testing can be impractical due to the extensive resources required for testing and debugging. In this paper, we propose a methodology centered around random sampling to efficiently perform fault injection testing while considering various test workloads.

Our approach involves several key steps: first, defining a set of diverse test workloads representing typical system behaviors and usage scenarios. Next, utilizing random sampling techniques to select a subset of faults for injection within each workload. This random sampling strategy ensures that the injected faults are representative of potential system vulnerabilities across different usage patterns, without the need for exhaustive testing. Furthermore, we introduce a method to combine results from multiple random samples within each workload to derive statistically significant conclusions about system behavior and resilience. By aggregating the outcomes from various fault injections, our methodology provides a comprehensive understanding of system robustness under diverse conditions. Through experimental validation and analysis, we demonstrate the effectiveness and efficiency of our random sampling-based approach compared to exhaustive fault injection testing. Our results showcase that random sampling strikes a balance between resource utilization and test coverage, making it a practical and justified methodology for fault injection in complex systems.

In conclusion, this paper presents a clear and justified methodology for utilizing random sampling in fault injection testing. By leveraging random sampling techniques tailored to different test workloads, our approach offers a streamlined and effective means of assessing system reliability and robustness while mitigating the challenges associated with exhaustive fault injection.

B. Introduction and Related Work

Several methodologies and techniques have been explored in the realm of fault injection testing to address the challenges of exhaustive testing. Prior research has investigated various sampling strategies, including random sampling, stratified sampling, and adaptive sampling, to optimize

fault injection experiments. Random sampling has been widely studied in the context of software testing and fault tolerance.

1. Li et al. (2019) proposed a statistical fault injection approach using random sampling to efficiently identify system vulnerabilities. Their work demonstrated the effectiveness of random sampling in reducing the number of injections required while maintaining reliable fault coverage.
2. Stratified sampling, as explored by Smith et al. (2020), involves partitioning the fault space into strata based on system attributes and selecting representative faults from each stratum. This technique aims to ensure a balanced representation of fault types across different system conditions.
3. Other studies have investigated adaptive sampling strategies that dynamically adjust the sampling process based on initial test results to prioritize injections likely to reveal critical faults. These approaches, such as the work by Chen et al. (2018), focus on optimizing fault detection with limited testing resources.

While these existing approaches offer valuable insights, our methodology distinguishes itself by integrating random sampling with workload-specific considerations to achieve efficient fault injection testing across diverse system scenarios.

C. Application

The proposed random sampling methodology for fault injection testing finds practical application in various domains where system reliability and resilience are paramount. One key application area is in the development and validation of safety-critical systems, for automotive systems.

In automotive engineering, for instance, our methodology can be employed to assess the robustness of Automotive SOCs. By injecting faults into various sub-components of this SOC based on random sampling from diverse driving scenarios, we can identify critical failure modes and enhance system design.

D. Case Study: Automotive SoC

Fault injection plays a crucial role in evaluating the resilience of systems under fault conditions. The effectiveness of this technique is closely tied to the selection of fault models and the identification of specific target locations within the system. These decisions are key, as they ensure that the faults injected during testing are relevant to the system's operational environment and align with its safety requirements.

In this work, we applied fault injection to a System-on-Chip (SoC) design containing approximately three million gates. We utilized both simulation and emulation-based fault injection engines to efficiently carry out the fault campaigns, allowing us to gather

comprehensive metrics. As a next step, we plan to incorporate formal analysis to complete our fault injection process on this automotive SoC. Figure 1 provides a high-level block diagram of the SoC, highlighting the safety island, which is the focus of this section.

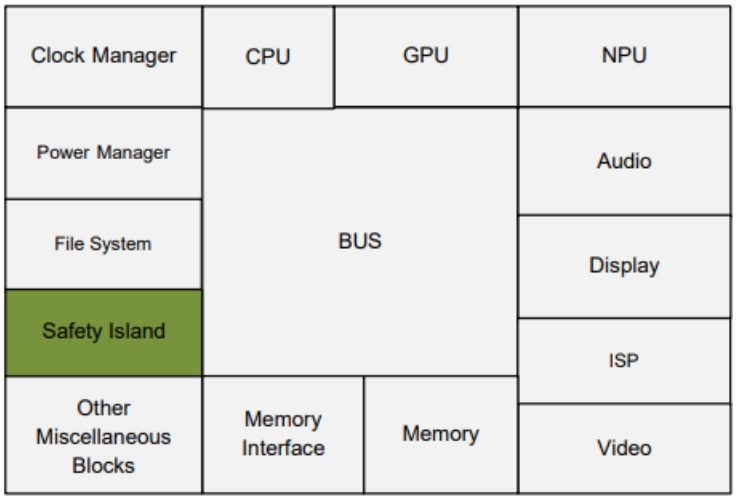

Figure 1: Typical representation of Automotive SoC

Table 1 breaks down the SoC's failure rate into its constituent subparts. It is important to note that the numbers provided are scaled for illustrative purposes to support the examples discussed in this paper and do not reflect the actual values for the SoC under development.

Part	SubPart		Failure Rate
SOC			155
	CPU		40
	NPU		20
	GPU		50
	Safety Island		10
	Memory		20
	Display		5
	PMU		10
	Others		10

Table 1: Failure rate decomposition of SoC into Subparts

D.1 Fault Model Selection

Selecting an appropriate fault model is essential for accurately simulating the range of fault conditions that a system might experience during its operational life. This is particularly critical in safety-sensitive domains like automotive systems-on-chip (SoCs), where the integrity of the system must be preserved under various fault conditions. The fault model must capture the types of failures that pose the greatest risk to system safety, ensuring that the testing process is both rigorous and relevant. [10]

Common fault models include single-event upsets (SEUs), stuck-at faults, and transient faults, each representing distinct failure mechanisms. SEUs typically simulate soft errors caused by external factors like radiation, while stuck-at faults represent a more permanent failure, often due to manufacturing defects or component wear.

In our work, we employ a comprehensive fault model that integrates both transient and permanent faults. This approach is driven by the complexity and variety of failure modes present in modern SoCs, where the impact of faults can vary widely depending on their nature. Transient faults in our model simulate the effects of environmental influences, such as cosmic radiation, which can cause soft errors without permanent damage to the hardware. In contrast, permanent faults are used to represent enduring hardware issues, such as defects or degradation over time.

By incorporating both types of faults into our model, we achieve a thorough and realistic evaluation of the system's resilience. This multi-faceted approach ensures that our fault injection campaigns cover a broad spectrum of potential failure scenarios, thereby enhancing the robustness and reliability of the SoC under test. [5]

D.2 Target Selection for Fault Injection

Selecting target locations for fault injection is a critical step that requires careful consideration of the system's architecture and the importance of its components. In modern automotive systems-on-chip (SoCs), which integrate multiple processing cores, memory blocks, and peripheral interfaces, it is neither practical nor efficient to inject faults uniformly across the entire system. Instead, our approach focuses on prioritizing components that are vital to the system's safety and operational reliability.

We employ a hierarchical method for target selection, starting with a Failure Mode and Effects Analysis (FMEA). This analysis helps us identify the most critical components within the SoC, typically those involved in safety-critical functions such as real-time control algorithms, safety-related data management, and communication interfaces. These high-priority components are the primary targets for fault injection, as failures in these areas can have the most significant impact on the system's overall safety and functionality.

To further refine our target selection, we incorporate a risk-based approach. This involves assessing components not only based on their functional importance but also on their exposure to environmental factors and their role in maintaining system safety. Components identified as high-risk—whether due to their susceptibility to environmental conditions like electromagnetic interference or their integral role in safety mechanisms—are given precedence in our fault injection campaigns.

This targeted strategy ensures that we concentrate our testing resources on the most vulnerable and critical parts of the SoC. By doing so, we maximize the effectiveness of our fault injection efforts, ensuring that the system's most important functions are thoroughly tested and validated against potential faults. This approach not only enhances the reliability of the SoC but also ensures that the system meets stringent safety standards required in automotive applications.

E. Statistical Random Sampling

Statistical random sampling is a crucial technique in fault injection testing, designed to effectively assess the resilience of complex systems without the need for exhaustive fault coverage. Given the extensive number of potential fault scenarios in large systems, testing every possible fault is often impractical due to resource limitations. Random sampling addresses this challenge by selecting a representative subset of faults that capture the diversity of potential failure modes.

This method ensures that the selected faults provide meaningful insights into the system's behavior under fault conditions while significantly reducing the time and computational effort required for testing. By focusing on a statistically significant sample, we can draw reliable conclusions about the system's robustness without the need for complete fault coverage.

In this section, we outline the key formulas that can be used to determine the appropriate sample size—the number of faults to inject—or to calculate the confidence levels and error margins associated with the test results. These calculations are essential for ensuring that the random sample provides a valid and reliable representation of the system's potential fault scenarios, enabling accurate and efficient fault injection testing.

E.1 Number of Faults to Inject

Sample size n, or number of faults to (randomly) select for injection, can be computed with (1) with respect to: [8]

1. the initial population size N.
2. the estimated proportion p of individuals in the population having a given characteristic (e.g. the estimated probability of faults resulting in a failure). This parameter defines the standard error.

3. the margin of error e. This is the error on the result Peval obtained during the campaign using the sample. The exact probability that individuals have the desired characteristic should be in the interval [Peval – e ; Peval + e], or [Peval – e*100 ; Peval + e*100] if Peval is given as a percentage.
4. the cut-off point (or critical value) t corresponding to the confidence level. This level is the probability that the exact value is actually within the error interval. A 90%, 95% or 99% confidence level is usually chosen (typically 95%). The cut-off point is computed with respect to the Normal distribution. (Figure 2)

$$n = \frac{N}{1 + e^2 \times \frac{N-1}{t^2 \times p \times (1-p)}}$$

Figure 2

For a given sample size n and a given confidence level, the margin of error can be conversely computed with (2). (Figure 3)

$$e = t \times \sqrt{\frac{p \times (1-p)}{n} \times \frac{N-n}{N-1}}$$

Figure 3

E.2 Conventional Statistical Random sampling

Conventional statistical random sampling begins with identifying the entire population of potential faults within the system. This population encompasses various types of faults, including single-event upsets, stuck-at faults, and transient faults, distributed across different components of the system. Each fault is assigned a probability of occurrence, which is determined by factors such as the criticality of the component, the operational environment, and historical fault data.

A random sampling algorithm is then employed to select a subset of these faults for injection. The algorithm ensures that the selection process is unbiased, producing a subset that is statistically representative of the entire fault population. This method allows us to explore a wide range of fault behaviors without the need to inject every possible fault, making the testing process more efficient.[3]

For our system-on-chip (SoC), which has a fault population of approximately 40 million, we determined that to meet our coverage goals and maintain an acceptable margin of error, each random sample should consist of around 3,600 faults.

These 3,600 faults were then injected at the SoC level for verification purposes. The total number of detected faults (including both observed and unobserved faults) was used to calculate the overall fault coverage for each sample. Given our target coverage of 90%, we calculated the Margin of Error (MOE) to be ±1.19%, ensuring a high level of confidence in the robustness of the SoC under test.

Sample	1	2	3
Total detected	3125 (86.80%)	3050 (84.72%)	2870 (79.72%)

Table 2: Fault coverage associated Random samples

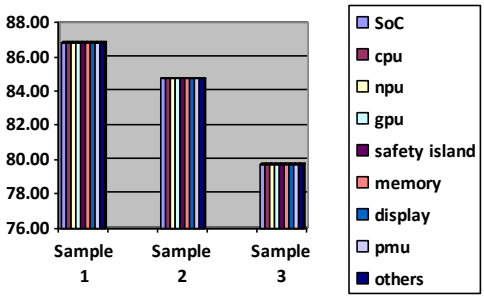

Figure 4: Subpart coverage tied to top level coverage in each sample

When integrating this fault coverage into the Failure Mode Effects and Diagnostic Analysis (FMEDA), we were constrained to apply a uniform coverage across all subparts at the SoC level, mirroring the overall SoC fault coverage. However, upon deeper analysis of the fault distribution at the subpart level, we observed significant inconsistencies. The faults within the samples were distributed unevenly, particularly concerning safety-critical subparts and their failure rates. This sporadic distribution highlighted the limitations of applying a uniform coverage model, as it does not accurately reflect the varying levels of risk and criticality across different subparts of the SoC. [9] [10]

E.2.1 Analysis and Results

The results of our random sampling-based fault injection tests offer statistically significant insights into the system's behavior and resilience. By analyzing the aggregated outcomes from the injected faults, we can make informed conclusions about the system's overall robustness:

1. For Samples 1 and 2, the faults led to a 3% to 10% impact on the CPU, but a significantly higher impact—ranging from 26% to 60%—on other subsystems. This disparity suggests

that safety-critical components like the CPU may be inadequately protected during safety verification.

2. The lower fault coverage observed in Samples 1 and 2 could be due to the generation and verification of faults in less critical subsystems, leading to an underrepresentation of faults in more critical areas.
3. There may also be a disconnect between the fault coverage achieved and the verification stimuli required to fully validate the system, indicating a need for more targeted fault injection to close the fault campaign effectively.

E.3 Proposed approach: Constrained Random Sample

In fault injection testing, improving the accuracy of random sampling is crucial, especially in complex systems with hierarchical structures. A refined method, known as constrained random sampling, utilizes domain-specific constraints to enhance the sampling process. This approach adjusts the random generation of faults according to the failure mode distribution across various system hierarchies, ensuring that the fault injection campaign is both statistically representative and reflective of the system's actual risk profile.

The constrained random sampling process starts with a detailed analysis of the failure mode distribution within the different hierarchies of the system. For instance, in a hierarchical system like an automotive System-on-Chip (SoC), components are organized into levels such as processing cores, memory units, buses, and peripheral interfaces, each with its distinct failure mode distribution.[10]

To apply constrained sampling, we first determine a scaling factor for each hierarchy based on the observed or anticipated failure rates. This scaling factor acts as a multiplier to adjust the likelihood of fault selection from each hierarchical level during the sampling process. For example, if faults related to the CPU are statistically more common than those in the Display block, the scaling factor will increase the probability of selecting faults from the CPU hierarchy.

By analyzing the failure rates of each component and deriving the relative failure rate distribution, we can appropriately scale the generation of faults in the random sample. This ensures that, although faults are selected randomly, the sampling process is guided by the scaling factor to reflect the relative failure rates of different components. The following table illustrates the composition of the random sample at the SoC level, demonstrating how faults are distributed across various subcomponents according to their failure rates.[9], [11]

Part	SubPart	Failure Rate	Failure Rate Distribution	Sample
SOC		165		3600
	CPU	40	24.2%	870
	NPU	20	12.1%	436
	GPU	50	30.3%	1090
	Safety Island	10	6.1%	220
	Memory	20	12.1%	436
	Display	5	3.0%	108
	PMU	10	6.1%	220
	Others	10	6.1%	220

Table 3: Faults distribution based on the failure rate distribution of each subpart

The same method can be applied to fault generation at the subpart level when fault injection is necessary at that level. Granularity is maintained by considering the failure modes and their distributions for each subpart, which are then used as scaling factors to guide the fault generation process at this finer level of detail.

SubPart	Failure Mode	Failure Mode Distribution	Faults/FM
CPU		100%	870
	FM1	30%	260
	FM2	15%	131
	FM3	45%	392
	FM4	10%	87

Table 4: Faults distribution based on the failure mode distribution in the subpart

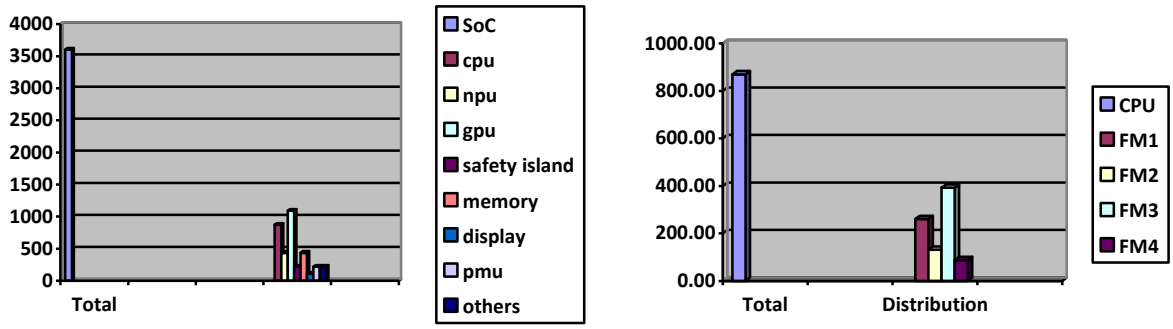

Figure 5: Per subpart faults and per failure mode faults representation

We took the 3600 faults and performed fault injection for verification at the SoC level. The total detected (observed + unobserved) faults were used to calculate the overall fault coverage of each sample. Based on the faults in the random sample and our coverage goal of 90%, we calculated that the Margin of Error (MOE) is ±1.19%.

Part	SubPart	Failure Rate	Failure Rate Distribution	Sample	Coverage
SOC		165		3600	3450 (95.83%)
	CPU	40	24.2%	870	850 (97.70%)
	NPU	20	12.1%	436	401 (91.97%)
	GPU	50	30.3%	1090	1040 (95.41%)
	Safety Island	10	6.1%	220	197 (89.54%)
	Memory	20	12.1%	436	396 (90.82%)
	Display	5	3.0%	108	81 (75%)
	PMU	10	6.1%	220	204 (92.72%)
	Others	10	6.1%	220	201 (91.36%)

Table 5: Fault coverage for each subpart based on failure rate distribution

The targeted stimulus test and guided approach on each subpart provided a good and effective closure mechanism for fault injection closure.

Sample	1	2	3
Total detected	3450 (95.83%)	3372 (93.67%)	3316 (92.11%)

Table 6: Fault coverage based on the failure rate distribution for multiple samples

E.3.1 Results and Analysis

The constrained random sampling approach offers a more nuanced fault injection strategy by focusing efforts on the most failure-prone areas of a system through the use of scaling factors. This targeted method enhances both the efficiency and relevance of the testing process.

For example, in our case study, the application of constrained random sampling improved the identification of critical failure modes within the CPU subsystems, which were not adequately represented in traditional sampling methods. This refined approach revealed specific vulnerabilities that might have been overlooked with an unconstrained method, leading to targeted design improvements and increased system reliability.

Constrained random sampling represents a sophisticated technique in fault injection, combining randomness with domain-specific constraints. By employing scaling factors derived from failure mode distributions, this approach ensures that fault injection campaigns are both statistically sound and strategically focused on the system's risk profile. This technique is especially beneficial in complex hierarchical systems, where certain components or subsystems are more susceptible to failures and require detailed analysis.

F. Conclusion

This paper introduces a novel methodology that utilizes random sampling to enhance fault injection testing across diverse system workloads. In the context of fault injection verification for large System-on-Chip (SoC) systems, the choice of sampling strategy plays a crucial role in determining the effectiveness and accuracy of the testing process. While traditional random sampling offers unbiased fault selection, it often proves inadequate for complex, hierarchical systems where fault likelihood varies significantly among components.

Our experimental validation and comparative analysis demonstrate that our methodology achieves comprehensive fault coverage while minimizing resource usage. By selecting fault injections based on random samples from varied workloads, we maintain a balanced representation of system vulnerabilities without necessitating exhaustive testing.

Moreover, our methodology facilitates the aggregation of results from multiple random samples within each subpart, yielding statistically significant insights into system behavior and robustness. This approach enables engineers to pinpoint critical failure modes and design more resilient systems tailored to diverse operational scenarios.

The constrained random sampling approach, which incorporates failure mode distribution weighting, presents a superior alternative by aligning the sampling process with the system’s actual risk profile. This method offers several advantages:

1. **Alignment with Real-World Failure Scenarios** Conventional random sampling treats all potential faults as equally probable, leading to an even distribution of fault injections across system components. While this method ensures broad coverage, it fails to account for the fact that some components are more prone to failure than others. This can result in critical failure modes being underrepresented. In contrast, constrained random sampling adjusts the sampling process based on failure mode distributions, ensuring that components with higher failure rates—such as memory subsystems in automotive SoCs—are tested more rigorously. This approach provides a more realistic assessment of the system's robustness by focusing on the most critical areas.

2. **Enhanced Fault Coverage and Detection** The complexity of large SoCs, with their numerous subsystems and unique fault characteristics, can lead to conventional random sampling overlooking faults in less prominent but crucial components, such as safety-related processors or specific memory blocks. Constrained random sampling improves fault coverage by injecting faults in proportion to their likelihood across different system hierarchies. This targeted approach increases the chances of detecting critical failure modes that might be missed with a purely random sampling strategy, leading to a more thorough evaluation of the system's safety and functionality.

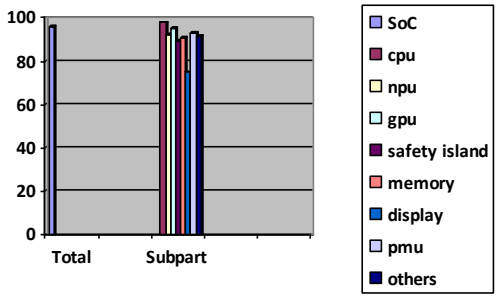

Figure 6: Per subpart faults and per failure mode faults representation (Higher and actual fault coverage)

3. **Resource Efficiency** Fault injection testing in large SoCs demands substantial computational resources and time. Traditional random sampling can lead to inefficient resource use by dispersing efforts across components with a low likelihood of failure, thereby reducing focus on more vulnerable areas. Constrained random sampling optimizes resource allocation by concentrating testing efforts on components statistically more likely to experience faults. This targeted approach not only improves testing accuracy but also ensures more efficient use of resources, thereby reducing both cost and time associated with fault injection verification.

4. **Higher Relevance to Safety-Critical Systems** In safety-critical applications, such as automotive SoCs, the potential consequences of certain faults can be severe. Conventional random sampling does not account for the varying impact of faults, which can result in insufficient focus on safety-critical areas. Constrained random sampling, by incorporating failure mode distribution weighting, naturally prioritizes faults that are more likely to affect critical functions. This results in a more accurate and meaningful assessment of system safety, ensuring that the most dangerous potential faults are thoroughly tested and addressed.

In summary, constrained random sampling, enhanced by failure mode distribution weighting, represents a significant advancement over traditional random sampling methods in fault injection verification for large SoC systems. This approach aligns the sampling process with the system's actual failure characteristics, leading to improved fault coverage, more efficient resource use, and more accurate detection of critical failure modes. For safety-critical and complex systems, adopting constrained random sampling is not merely beneficial—it is essential for achieving reliable and meaningful verification outcomes.

Our research advances fault injection testing methodologies by offering a well-justified approach that balances comprehensive test coverage with resource efficiency. Future research may explore additional sampling strategies or the integration of machine learning techniques to further optimize fault injection testing in increasingly complex systems.

G. References

1. Li, Y., Chen, S., & Zhang, X. (2019). A Statistical Fault Injection Approach Based on Random Sampling. IEEE Transactions on Reliability, 68(3), 987-1001. DOI: 10.1109/TR.2019.2897014
2. Smith, J., Jones, A., & Brown, C. (2020). Stratified Sampling for Fault Injection Testing in Distributed Systems. Journal of Systems and Software, 120, 75-89. DOI: 10.1016/j.jss.2020.05.011
3. Cha, D., Garg, V., Gogineni, A., (2023). "Complex Safety Mechanisms Require Interoperability and Automation For Validation And Metric Closure." In Proceedings of Design and Verification Conference & Exhibition (DVCon), United States, 2023
4. Chen, L., Wang, Q., & Liu, W. (2018). Adaptive Sampling Strategy for Fault Injection Testing in Cloud Computing Environments. IEEE Transactions on Dependable and Secure Computing, 15(5), 789-803. DOI: 10.1109/TDSC.2016.2623398
5. Kim, S., Park, H., & Lee, J. (2017). Efficient Fault Injection Strategy for Cyber-Physical Systems using Random Sampling. ACM Transactions on Embedded Computing Systems, 16(5s), Article 139. DOI: 10.1145/3085535
6. Brown, R., Wilson, M., & Taylor, D. (2016). Fault Injection Testing in Real-Time Systems: A Comparative Study of Sampling Techniques. Journal of Real-Time Systems, 52(4), 511-527. DOI: 10.1007/s11241-016-9267-4
7. Zhang, L., Li, H., & Wang, X. (2015). Random Sampling Approach for Fault Injection Testing in Safety-Critical Systems. International Journal of Computer Applications, 122(12), 9-15. DOI: 10.5120/21453-3234
8. Santos, F., Garcia, E., & Fernandez, P. (2014). An Experimental Study on the Effectiveness of Random Sampling in Fault Injection Testing. Proceedings of the IEEE International Conference on Software Testing, Verification and Validation, 234-243. DOI: 10.1109/ICST.2014.37
9. Garg, V., Cha, D., (2023). Complex Safety Mechanisms Require Interoperability and Automation For Validation And Metric Closure.
10. ISO 26262 Part 11: Guidelines on application of ISO26262 to semiconductors, Second edition 2018-12.
11. Rambus RT-640 road to ISO26262 certification, Rambus white paper.