Efficient Methodology for ISO26262 Functional Safety Verification

Silva, Felipe Augusto Da; Bagbaba, Ahmet Cagri; Hamdioui, Said; Sauer, Christian

DOI
10.1109/IOLTS.2019.8854449

Publication date
2019

Document Version
Accepted author manuscript

Published in
2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design, IOLTS 2019

Efficient Methodology for ISO26262 Functional Safety Verification

Felipe Augusto da Silva[1,2], Ahmet Cagri Bagbaba[1], Said Hamdioui[2] and Christian Sauer[1]
[1]Cadence Design Systems, Feldkirchen, Germany
[2]Delft University of Technology, Delft, The Netherlands

***Abstract*— Tolerance to random hardware failures, required by ISO26262, entails accurate design behavior analysis, complex Verification Environments and expensive Fault Injection campaigns. This paper proposes a methodology combining the strengths of Automatic Test Pattern Generators (ATPG), Formal Methods and Fault Injection Simulation to decrease the efforts of Functional Safety Verification. Our methodology results in a fast-deployed Fault Injection environment achieving Fault detection rates higher than 99% on the tested designs. In addition, ISO26262 Tool Confidence level is improved by a fault analysis report that allows verification of malfunctions in the outputs of the tools.**

Keywords - ISO26262; Fault Injection Simulation; Formal Methods; ATPG; Functional Safety.

I. INTRODUCTION

Functional Safety Verification is one of the most challenging steps for Integrated Circuit (IC) compliance with ISO26262. In safety-critical applications the system must include Safety Mechanisms being able to detect up to 99% of the random faults susceptive of the design. In addition, ISO26262 requires that all possible malfunctions of tools (used during fault analysis) have to be considered as per Tool Qualification requirements [1]. Therefore, there is a high demand for effective Functional Safety Verification methodologies allowing the reduction of costs while maintaining the same levels of safety.

The commonly used method for Functional Safety Verification is Fault Injection (FI) Simulation [2][3]. The purpose is to show that fault effects can propagate to outputs and that Safety Mechanisms can detect them. In order to cause propagation of all faults, complex verification environments with numerous test inputs are required, resulting in long FI Campaigns. To address this challenge, we can deploy different verification technologies in a single methodology. Methodologies applying Formal Methods to identify faults that cannot propagate to outputs of the design (Safe faults) [4][5][6], and ATPG techniques to generate test patterns that potentialize fault propagation [7][8] have been proposed. Even though Simulation, Formal Methods and ATPG have complementary strengths, to the best of our knowledge, they were not previously combined in a single fault analysis flow that aims at fault propagation for compliance to ISO26262 requirements.

Our work takes advantage of three different technologies aiming to achieve high fault detection rates while decreasing the efforts of traditional FI Campaigns. ATPG is used for fast deployment of a verification environment that provides high fault propagation rate. The outputs from ATPG are used by the FI Simulator, for verifying the functional behavior of the design under each fault. In parallel, Formal Methods are applied to the design to identify Safe faults. In addition, the outputs of each tool are verified against each other to identify malfunctions, increasing the confidence in the tool's outputs, as required by ISO26262 [1]. The main contributions of our methodology are:

- Reduction of the efforts with Test Environment developments and in the number of required Simulations by deploying automatic generated ATPG Test Environments to FI Simulation campaigns.
- Increasing compliance to ISO26262 fault metrics by identification of Safe faults with formal methods.
- Generation of report containing detailed information of tool outputs to detect malfunctions.

II. PROPOSED METHODOLOGY

Our methodology aims to automate the execution of Simulation, ATPG and Formal analysis for a specific design. At the end of the execution, the outputs of the tools are compared to find discrepancies. An application was develop in order to control the execution flow and generate final reports. The Fault Checker application can be configured to use any ATPG, Simulation, and Formal tools. At the beginning of the execution, the user should configure the scripts to control the execution of each tool and provide the rules for parsing the tool reports.

The application starts with the execution of the ATPG and Formal flows. As these two flows are independent, they can be executed in parallel using different CPUs. Simulation flow requires the ATPG Testbench and test vectors to start. So, after the ATPG flow is finished, the Fault Checker will extract the generated Test Environment and will use it for the FI Simulation. At the end of each flow, the reports generated by the tools are parsed to a common format, allowing verification of the results to identify discrepancies between the tools. Finally, at the end of all flows, the relevant parsed data is retrieved and compared. The comparison is based on rules that associate the annotations used by each tool. For example, faults classified as Untestable by the Simulator are equivalent to faults classified as Safe by Formal and Ignored by ATPG. In case a rule is not obeyed, the Fault Checker will include a Warning tag to the report, informing that this fault requires attention from the designer.

This project has received funding from the European Unions Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 722325.

Results can be analyzed in a CSV report that details the annotation of each fault by each tool. An error caused by a malfunction in one of the tools, will be indicated by a Warning in the CSV report. For example, if simulation annotates a fault as Detected and Formal annotates the same fault as Safe, it indicate a malfunction in one of the tools. The report provides supplementary information with further possibilities for fault analysis. For example, if a specific fault is considered Undetected by Simulator and Dangerous by Formal, it means that formal analysis identified at least one test stimulus that can propagate the fault. This information can be used on a new FI Simulation to achieve detection of this fault. Any other discrepancy between the faults is indicated in the report with a Warning.

III. VALIDATION

This section describes the validation process of the proposed methodology. First the Fault Checker was configured with execution scripts to deploy Cadence®Xcelium™Fault Simulator (XFS), Cadence®JasperGold (JG) Formal Verification Platform and Cadence®Modus DFT Software Solution ATPG component as the representatives of each technology. Second, selected designs were verified by the Fault Checker. Table I details, for each design, the total number of faults, the fault detection rate, and the indication of Pass or Warning resulting from the verification of the tools by the Fault Checker.

TABLE I
FAULT CHECKER RESULTS.

Design	Faults (SA0/SA1)	Detection Rate	PASS	WARNING
Up Down Counter	162	100%	162	0
Memories	2782	99.78%	2776	6
AC97	57226	99.77%	57108	118
Conmax	153454	99.80%	153191	263

During the Up Down Counter and Memories designs verification, the Fault Checker confirmed that all faults have equivalent annotations. As the examples are relatively simple, the different tools can determine that all faults can propagate to outputs. For the Memories design, the application detected 6 faults that were annotated as Safe by the Formal analysis, and can be disregarded.

On the AC97 design, the Fault Checker was able to detect 118 faults with distinctive annotations. From these, 49 faults were annotated as Safe by Formal and can be disregarded; 23 were annotated as Dangerous by Formal meaning that Formal can extract test inputs to cause propagation of the faults; and 46 faults were not classified, meaning that they require manual analysis.

During the analysis of the Conmax design, the methodology detected 263 discrepancies between the tools. From these, 7 faults were annotated as Dangerous by Formal. Meaning that results from Formal can be applied for detecting these faults during simulation. The other 256 faults have non conclusive annotations and should be manually analyzed.

The results demonstrated above corroborate with the listed contributions. First, the comparison of the fault classifications from each tool enables identification of tool malfunction. The report generated by the Fault Checker allows detailed analysis of faults and can be used to support ISO26262 Tool Qualification. Second, Safe faults classification by Formal Methods permits improvement of fault tolerance, by decreasing the total number of faults and improving ISO26262 metrics. Third, the proposed methodology shows considerable fault detection rates for all tested designs.

IV. CONCLUSIONS

Due to the harsh requirements for random hardware failures tolerance, Functional Safety verification is a challenging step for ISO26262 compliance. FI simulation, as part of this process, becomes a long and expensive procedure, that is usually repeated numerous times until the metrics for fault detection are achieved. We propose a methodology that deploys ATPG and Formal to support Simulation results and to decrease the overall effort of FI Simulations. Our methodology enables the use of test environments created with ATPG for the simulation of faults, and the use of Formal for identification of Safe faults. Formal results allow the optimization of the Fault List, reducing the number of faults to be simulated. In addition, the results of the tools are compared to identify discrepancies and potential defects. The inclusion of redundancy as a method to detect malfunctions in tools is suggested for achieving ISO26262 Tool Confidence [1]. Our results have shown high fault detection rates, achieving more than 99% of detected faults. In addition, the proposed methodology can identify Safe faults, contributing to reaching ISO26262 metrics.

REFERENCES

[1] ISO, *ISO 26262 - Road Veichles - Functional Safety - Part 8: Supporting processes*, International Standardization Organization Std., Nov. 2011.

[2] A. Nardi and A. Armato, "Functional safety methodologies for automotive applications," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, nov 2017.

[3] Y.-C. Chang, L.-R. Huang, H.-C. Liu, C.-J. Yang, and C.-T. Chiu, "Assessing automotive functional safety microprocessor with ISO 26262 hardware requirements," in *Technical Papers of 2014 International Symposium on VLSI Design, Automation and Test*. IEEE, 2014.

[4] K. Devarajegowda and J. Vliegen, "Deploying formal and simulation in mutual-exclusive manner using jaspergolds proofcore technology," in *Cadence User Conference CDNLive EMEA*, 2017.

[5] S. Marchese and J. Grosse, "Formal fault propagation analysis that scales to modern automotive SoCs," in *2017 Design and Verification Conference and Exhibition DVCON Europe*, 2017.

[6] A. Traskov, T. Ehrenberg, and S. Loitz, "Fault proof: Using formal techniques for safety verification and fault analysis," in *2016 Design and Verification Conference and Exhibition DVCON Europe*. DVCON, 2016, pp. 27–32.

[7] S. Praveen, S. Yellampalli, and A. Kothari, "Optimization of test time and fault grading of functional test vectors using fault simulation flow," in *2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE)*. IEEE, nov 2014.

[8] S. Arekapudi, F. Xin, J. Peng, and I. G. Harris, "ATPG for timing-induced functional errors on trigger events in hardware-software systems," in *Proceedings The Seventh IEEE European Test Workshop*. IEEE Comput. Soc, 2002.