Improving the Confidence Level in Functional Safety Simulation Tools for ISO 26262

Ahmet Cagri Bagbaba, Cadence Design Systems GmbH, Feldkirchen, Germany (*abagbaba@cadence.com*)

Felipe Augusto Da Silva, Cadence Design Systems GmbH, Feldkirchen, Germany (*dasilva@cadence.com*)

Christian Sauer, Cadence Design Systems GmbH, Feldkirchen, Germany (*sauerc@cadence.com*)

***Abstract*— Higher Tool Confidence Level (TCL) is needed for tools used on the verification of safety-critical SoCs, aiming to achieve the required Automotive Safety Integrity Level in ISO 26262. This paper presents a methodology to improve the confidence level of functional safety verification flow. To do this, we compare the fault-list generated by the fault injection (FI) simulator with the Automatic Test Pattern Generation (ATPG) flow for stuck-at (SA) fault types. Moreover, we compare fault coverage results by using test vectors generated by the ATPG tool so the result of the FI simulation is compared to the results gained from the ATPG. This is a way to improve simulator's confidence level by taking advantage of strength of the ATPG.**

Keywords—iso26262; functional safety; tool confidence; atpg; dft

I. INTRODUCTION

Functional safety (FS) refers to the absence of unreasonable risk caused by systematic failures and random hardware failures [1]. FS and especially the analysis of random failures are becoming part of the requirements for the design of complex systems. Therefore, a tighter integration between FS analysis and the standard platform design and verification is required. To achieve the FS of SoCs it is important analyze the use cases for all the flow tools according to their probability of introducing errors. This analysis shall evaluate if the malfunctioning tool or its erroneous output can lead to the violation of a safety requirement [2]. Based on this analysis, ISO 26262 Part 8 covers all aspects of TCL and defines key concepts of confidence and qualification [1]. The TCL assesses the error injection risk of each tool in the flow to document the confidence level for the data processing of each tool.

In this work, we increase the TCL of FI simulator tool. One way for accomplishing this is to compare results of FI simulator with the well-known and trusted results from ATPG. The purpose is to make FI simulator as optimal as the ATPG tool. We focus on two main aspects: The first is to assure that the instrumentation of the ATPG tool and the FI simulator is equivalent i.e. all optimizations are used. The second is to compare fault coverage results reported by the ATPG tool and the FI simulator. Developed methodologies are demonstrated as proof-of-concepts within the Cadence Design & Functional Safety flow [3] and the context of the ISO 26262.

The rest of this paper is organized as follows: in Chapter II, we mention about related works. Chapter III includes overview information about functional safety, fault injection simulation and the ATPG. We explain our approach and show the results in Chapter IV and V. In the final chapter, we conclude this paper by summarizing the work.

II. RELATED WORK

According to the ISO 26262 [1], verification tools employed on safety-critical automotive embedded systems must undergo qualification. In [4] authors present a semi-automatic qualification method involving a monitor and fault injection that reduce cost in the qualification process. In [5], authors define equivalence and dominance relations between fault pairs to optimize fault lists. In [6], authors tie functional safety to the traditional EDA domain.

Our approach, beside literature, uses results of the ATPG to prove the confidence of FI simulator and evaluate the tool potential impact on safety applications and the tool error detection capabilities. This means that same fault list is generated and most of the results are same with the ATPG tool.

III. Overview

A. *Functional Safety and ISO 26262*

Functional safety is the automotive industry standard, designed for safety-related systems for series production passenger vehicles with a maximum gross vehicle mass up to 3500 kg and that are equipped with one or more electrical/electronic subsystems [7]. Malfunction of the electrical/electronic component is classified into two types as systematic failures and random failures. Systematic failures represent the failures in an item or function that are induced a deterministic way during development, manufacturing or maintenance. Random failures represent failures during the lifetime of a hardware element. They can be classified as permanent faults (stuck-at faults) and transient faults (single-event-upsets or soft errors). Our focus in this work is permanent faults.

The design of safety systems involves error correction using checkers and error correction using redundancy. The former defines checkers which monitor the systems and trigger error response and recovery features when necessary. The latter defines redundancy which involves duplication of the entire system or a portion of the system. All requirements must be implemented by tracing from the system to components and ensured their development flow aligns with a tool confidence level. Also, recording and reporting functional safety measures to have a verified system are important.

B. *Fault Injection Simulation*

Fault injection enables to verify the capability of a safety mechanism to recognize failures in a design's functionality, by injecting faults into the design. One way of doing this is fault injection simulation. In fault injection simulation, target system and the possible hardware faults are modeled and simulated by the simulator. In this process, the system behaves as if there is a hardware fault. The advantage of the fault injection simulation is that there is no risk to damage the system in use. Moreover, it is cheap in terms of time and effort. Additionally, fault injection simulation has a high observability and controllability in presence of faults.

To inject faults into a design, the FI simulator needs to know the fault target at which to inject fault. In this work, we enable fault instrumentation on the ports of cell instances. Therefore, all library cell ports are identified as faultable within the specified instance or module. This is equivalent to using the ATPG semantics.

There are different types of fault model to inject on specified fault nodes. The FI simulator supports single event upsets, stuck-at-0/stuck-at-1 and single event transient. We use stuck-at fault models in this work. The stuck-at model forces a signal to either 0 or 1 from the start of fault injection through to the end of simulation. This model can apply to nets or registers. The fault injection simulation flow starts with invoking of elaborator and instrumenting of faults according to a fault target. Then, a good simulation is run to generate reference values for fault simulation. Additionally, strobe points are specified in this point to monitor signals. Finally, one or more fault simulations with faults injected are run and generated reports are analyzed by the help of functional outputs (F-O) and checker outputs (C-O). Single run is illustrated in Figure 1.

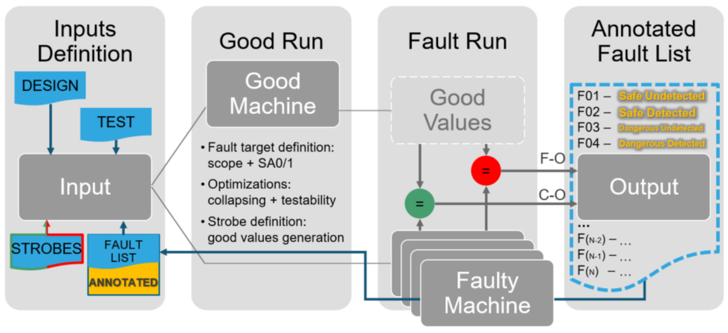

Figure 1-Fault Injection Simulation

C. *Automatic Test Pattern Generation*

Testability has become a critical concern for ASIC designers. Design-For-Test (DFT) techniques provide measures to test the manufactured device for coverage and quality. We use here Scan-based test which is one of the DFT techniques. This method basically replaces D flip flops with their scan-equivalent flops and serially connects the scan flops into scan chains. By replacing the flip-flops with their scan-equivalent flip-flops, the ATPG tool can achieve higher fault coverage and generate a more compact test pattern set for the design.

Defining a testmode for ATPG is an important step. A testmode is a specific test configuration of the design. The configuration defines how the test structures are accessed and how clocking is controlled. Each configuration is defined by the clocking and other test function pins, and the current test methodology. In this work, we use FULLSCAN testmode. The purpose of this testmode is to get static test using no compression.

Another step of the ATPG is generation of test vectors. ATPG tool can write test vectors to meet the manufacturing interface needs of IC manufacturers in different formats such as Standard Test Interface Language, Waveform Generation Language, and Verilog. In this work, we use Verilog test vectors.

IV. APPROACH

ISO 26262 tool confidence level can be improved by use of redundant methodologies to detect errors in the tool outputs. Cadence offers different technologies capable of generating fault list and ATPG from the Cadence® Modus™ DFT Software Solution can work together with the Cadence® Xcelium™ Fault Simulator (XFS) to create robust and optimized fault injection campaign. In this work, XFS for FI simulation, Cadence® Genus™ Synthesis Solution for synthesis and Modus for ATPG are used. Our purpose is to compare two different approaches regarding the results and improve the FI simulator's confidence. Comparison of the results to a different approach increases TCL and provide more optimized fault list. In other saying, we show that different tools capable of generating fault list bring about same fault list. Scan-based Design-for-Test structure is inserted during the synthesis. We have two methodologies explained below.

A. *Instrumentation comparison between the ATPG tool and FI simulator*

Fault-list can be generated both by the ATPG tool and FI simulator. ATPG tool is accepted as reference because of its usage for long years in industry. Hence, it is expected that FI simulator's fault optimization achieves the same capabilities as the ATPG tool. The purpose is to prove that FI simulator contains all instrumentation and optimization potential. In other saying, the aim is to show that instrumentation of FI simulator and ATPG tool is equivalent.

This approach is shown in Figure 2 as Compare Optimized Fault Lists. Here, the fault lists generated by both tools compared to verify: whether the same faults are instrumented, if they have the same number of prime faults (collapsing), and if they have the same number of untestable faults. Our methodology finds fault collapsed groups and differences between lists if there exists. For this comparison, the designs are synthesized with DFT insertion. Elaboration of gate level (GL) Design Under Test (DUT) is enough in the simulator side. For the ATPG side, test patterns are generated by the ATPG tool after scan-chain insertion. The ATPG tool creates logic/scan test and produce results. All comparisons are done for stuck-at faults.

B. *Compare fault coverage results reported by the ATPG tool and FI simulator*

ATPG is a process used in SoCs testing wherein the test vectors required to check a device for faults. Test vectors are automatically generated by the tool. However, testbench is required for FI simulator. Therefore, it is a good idea to use test vectors generated by the ATPG tool as a test instead of writing a testbench.

The FI simulator can be used in conjunction with ATPG. In this way, each fault detected by the ATPG tool should be detected by FI simulator. The aim is to prove that FI simulator and the ATPG tool have the same coverage.

In this step, DFT is inserted by the synthesize tool and the ATPG tool is used to collect all SA faults at GL netlist and generate test patterns as shown in Figure 2 as Compare Annotated Fault Lists Level. All SA faults instrumented by the ATPG tool are injected by FI simulator. By the help of this, ATPG tool annotated fault list and FI simulator annotated fault list can be easily compared and fault coverage differences can be observed.

ATPG results include different annotation types such as tested, untested, ignored and collapsed. Beside this fault injection simulation results have annotation results such as detected, untestable, and undetected. To make a comparison between results, we define a comparison method shown in Table-I. Second row of the table explains that if the fault is tested in Modus, it needs to be detected in Xcelium. In this case, check annotation result is PASS. Last column shows the annotation result whether it is pass or warning. WARNING means that there is a difference between results. This method helps for debugging in case of difference between results and tool verification for ISO 26262.

Table I-Annotation Results Comparison Method

Xcelium Annotation Types	Modus Annotation Types	Check Annotation
Detected	Tested by simulation	PASS
Untestable	Ignored (unclassified)	PASS
Potentially Detected	Tested by implication	PASS
Undetected	Collapsed Tested by simulation	WARNING
Untestable	Collapsed Tested by simulation	WARNING

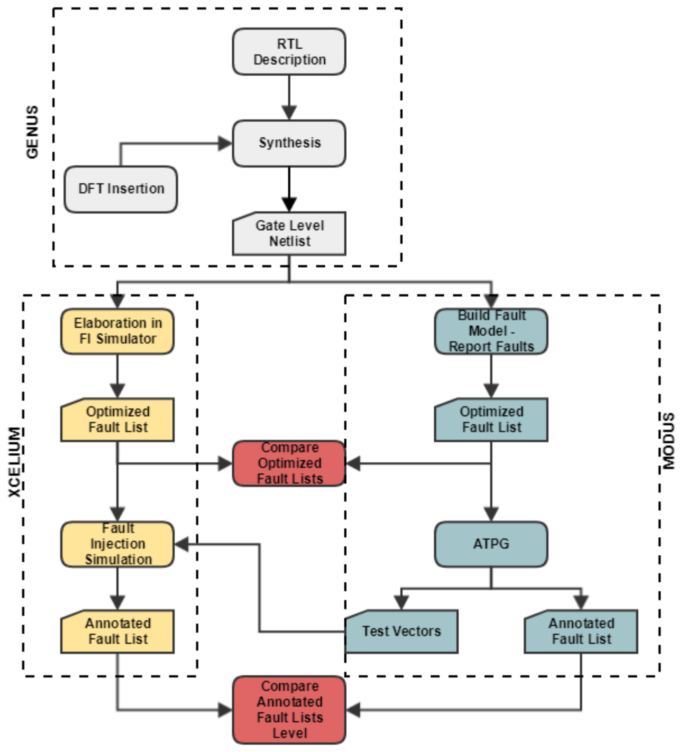

Figure 2-Flow

V. Results

First, we show that ATPG and fault injection simulator generate the same fault lists. This provides us to compare results easily. All circuits are analyzed for SA0 and SA1 fault models. Collapsed and untestable faults are included for analyses.

Table-II shows the results of instrumentation comparison between Modus and XFS. This comparison is shown in Figure 2 as compare optimized fault lists. IWLS 2005 benchmark circuits [7] are used to collect results. Optimized fault list includes faults that do not need to be simulated since behavior of the design in presence of these faults can be predicted. First row of Table-II gives the number of instrumented faults for each design. Last row of this table shows the number of optimization differences. The reason of this is different collapsing approach of tools. In other saying, collapsing methods of tools have effect on the results. For example, Modus does not collapse primary inputs; however, primary inputs are collapsed by XFS. This does not change functionality of circuits.

Table II-Comparison of optimized fault lists

	ac97	**aes**	**dma**	**total**
Nr of instrumented faults	57226	91916	66708	212850
Nr of optimization differences	12	4	76	92

Table-III shows the results of the second part of this work which is comparison of fault coverage results and annotated fault list level reported by tools. Here, we use ITC'99 benchmark circuits [8] to generate results. For each fault listed in the ATPG annotated fault list, faulty behavior is simulated, and the observation points are compared against the reference values generated during the good run as shown in Figure 1. This flow is shown in Figure 2 as compare annotated fault lists level. For each tested design, number of total faults in Xcelium and Modus can be seen in Table-III. Xcelium columns shows the number of faults after fault simulation with ATPG test vectors. Last column shows the comparison results that is explained in Section IV. Overall results point that although test vectors generated by Modus is for external tester devices, we obtain same fault coverage values when we applied test vectors into the fault injection simulation tool. The length of test vectors is important here. It must be set to maximum length in order to obtain correct results.

Table III- Comparison of annotated fault list level

Design	**Xcelium**	**Modus**	**Check Results**
b01	230	230	PASS
b02	174	174	PASS
b03	882	882	PASS
b04	2268	2268	PASS
b05	2512	2512	PASS
b06	274	274	PASS
b07	1882	1882	PASS
b08	766	766	PASS

VI. Conclusions

FS becomes a crucial requirement in the safety critical automotive systems. This is causing a shift in the traditional design flow and pushing ISO26262 compliance into the traditional EDA tools. Therefore, it is vital to improve TCL. For this purpose, we propose methodologies to improve TCL in FI simulator by using strength of the ATPG. By considering guidance of ATPG results, we show that FI simulator contains all instrumentation/optimization potential and they have same coverage results. In the first method, optimized fault lists are compared. Results show that there are differences between the ATPG and FI simulator due to collapsing approach of tools, but this does not affect functionality of the circuits. In the second part of this paper, we compare annotated fault list level of the tools by using test vectors generated by the ATPG tool as a test in FI simulation. This method shows that both tools have same annotated fault list level and coverage values.

Acknowledgement

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 722325.

References

[1] ISO, “ISO26262-road vehicles functional safety,” International Organization for Standarization in ISO26262, 2011.

[2] M. Conrad, P. Munier, and F. Rauch, "Qualifying Software Tools According to ISO 26262," (white paper) http://www.mathworks.se/automotive/ standards/iso-26262.html, 2010, The Mathworks, Inc.

[3] Meeting Functional Safety Requirements Efficiently Via Electronic Design Tools and Techniques, [online] Available: https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/solutions/automotive-functional-safety-wp.pdf

[4] Q. Wang, A. Wallin, V. Izosimov, U. Ingelsson and Z. Peng, "Test tool qualification through fault injection," 2012 17th IEEE European Test Symposium (ETS), Annecy, 2012, pp. 1-1.

[5] I. Pomeranz and S. M. Reddy, "Equivalence, Dominance, and Similarity Relations between Fault Pairs and a Fault Pair Collapsing Process for Fault Diagnosis," in IEEE Transactions on Computers, vol. 59, no. 2, pp. 150-158, Feb. 2010.

[6] A. Nardi and A. Armato, "Functional safety methodologies for automotive applications," 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Irvine, CA, 2017, pp. 970-975.

Beckers, Kristian, et. Al. “Systematic derivation of functional safety requirements for automative systems.” *International Conference on Computer Safety, Reliability, and Security*. Springer, Cham, 2014.

[7] Cadence Research Berkeley, “International Workshop on Logic and Synthesis (IWLS) 2005 Benchmarks”.

[8] F. Corno, M. S. Reorda and G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results," in IEEE Design & Test of Computers, vol. 17, no. 3, pp. 44-53, July-Sept. 2000.