Flip Flop Weighting
A technique for estimation of safety metrics in Automotive Designs

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

DOI
10.1109/IOLTS52814.2021.9486697

Publication date
2021

Document Version
Final published version

Published in
2021 IEEE 27th International Symposium on On-Line Testing and Robust System Design (IOLTS)

Flip Flop Weighting: A technique for estimation of safety metrics in Automotive Designs

Felipe Augusto da Silva*†, Ahmet Cagri Bagbaba*, Said Hamdioui† and Christian Sauer*

*Cadence Design Systems
Munich, Germany

†Delft University of Technology
Delft, The Netherlands

***Abstract*—The requirements of ISO26262 for the development of safety-critical Integrated Circuits (IC) demand substantial efforts on fault analysis for safety metrics evaluation. Failing to achieve the required conditions entails modifications to the circuit, additional iterations through critical design phases, and consequently extra costs and delays. For that reason, providing accurate methods to estimate safety metrics is of great importance. This paper proposes a methodology that can efficiently and precisely estimate the safety metrics of Automotive designs. The technique is based on the characterization of a netlist to determine how hardware components contribute to fault propagation. Also, by examining the test stimuli applied during simulation, we can rank Workloads/Testbenches according to their fault detection coverage. The approach was verified running fault injection campaigns on distinct gate-level hardware designs, including an Automotive CPU. Our results show that the fault detection coverage can be estimated with an average error rate of 3% at up to 20X faster execution times when compared to the traditional campaigns. Hence the methodology provides an efficient and cost-effective mechanism to support engineers in a confident design space exploration.**

Keywords - ISO26262; Design Space Exploration; Fault Injection; Formal Methods; Simulation; Functional Safety; Verification.

I. INTRODUCTION

The development of safety-critical applications, such as autonomous driving, requires additional efforts to reduce the risk of failures that can cause life-threatening situations. For such applications, the system must include Safety Mechanisms being able to detect up to 99% of the circuit random faults. Functional Safety Verification, as defined by ISO26262, is usually performed at later stages of the development cycle, and failing to achieve the required safety-metrics demands additional iterations through critical development and verification phases. In other words, it has a high impact on costs and development time. In a typical lifecycle, the safety concept and architecture are established at the early development stages, requiring engineers to estimate fault detection coverage without a proper evaluation methodology. A misleading architecture decision before implementation will be exposed only at the final stages of the development when modifications are expensive. For that reason, there is a high demand for techniques that can support safety-engineers with design space exploration of safety features, increasing the confidence in conceptual decisions, and avoiding rework.

Numerous works address the fault classification demands of ISO26262. Fault Injection (FI) Simulation, as the method recommended by ISO26262, is explored by several researchers [1][2][3][4]. Also, alternative technologies, as formal, are explored for the classification of faults. The ability of formal techniques in analyzing the design behavior for all possible combinations of inputs it's a powerful tool for the identification of Safe Faults [5][6][7]. The combination of FI Simulation and formal methods is also examined [8][9][10]. The recent advances in fault classification propose optimizations in fault analysis and highlight the strengths of the different technologies. However, aspects of early design exploration are not addressed, causing a gap in the development lifecycle. Design space exploration refers to the activity of exploring design alternatives before implementation. The concept is established in several domains of IC development, like area, performance, power consumption [11][12], high-level synthesis [13], deep learning [14], among others. Nevertheless, to the best of our knowledge, there are no methodologies for early design space exploration targeting safety metrics for compliance with ISO26262.

Our work introduces a methodology for the estimation of safety metrics in Automotive designs. By allowing engineers to estimate fault detection rates before the final development stages, we provide a tool for the design space exploration of safety architectures, improving the confidence in conceptual decisions and decreasing the chances of rework. The methodology is based on the characterization of the netlist and of the workload concerning fault propagation. First, we identify the prime propagation nodes of the hardware design and designate a weight that represents the faults in their Cone of Influence (CoI). Next, we consider the impact of fault activation by analyzing nodes that are constant during the simulation of the workload. At this stage, we can rank the Workload/Testbench by their fault detection rate potential. Finally, we perform Fault Injection simulation on the prime propagation nodes to estimate the fault propagation behavior of all faults in the design. The compiled information allows an early assessment of the Fault Injection campaign results. The main contributions of this work are:

- A systematic approach for the estimation of safety metrics of Automotive designs;
- An effective method to rank Workloads/Testbenches according to their impact in the fault detection coverage;
- Validation of the methodology in real IPs, including an Automotive CPU;
- The results show an estimation of the fault detection

coverage with an average error of 3% at up to 20X faster execution times.

The rest of the paper is organized as follows: Section II introduces fault analysis techniques and illustrate how they are deployed in this work. Section III describes the proposed methodology. Section IV explains the validation process and discusses our results. Section V concludes.

II. FAULT ANALYSIS TECHNOLOGIES

This section outlines the processes employed by Formal Methods and Fault Injection Simulation targeting fault classification. Our work applies the strengths of both technologies to examine the propagation of faults in a given design and estimate the safety metrics.

A. Formal Methods

The formal analysis is one of the leading technologies for the identification of Safe faults. Employing structural and formal analysis, formal tools can verify a circuit in the global scope, considering every evaluation context and test stimuli [15]. Consequently, these tools can exhaustively prove that a fault can never produce any failure.

1) Structural Analysis: The Structural Analysis aims to determine the testability of faults. The testability of a fault is determined by verifying the physical characteristics of the design. Figure 1 illustrates the examination applied by the Structural Analysis.

Figure 1 represents a circuit with combinational logic (g), inputs (in), outputs (out) and fault targets (f). Considering this circuit, it is possible to define the following fault behaviors by applying Structural Analysis:

1) As the only Observation Point (strobe) configured for the fault analysis is ’out0’, any fault that is outside of its Cone of Influence is considered Untestable. For that reason, any fault in ’f1’ is Structural-Safe as there is no physical connection between the fault location and the strobe.
2) Depending on the characteristics of ’g1’ drivers, it is possible to define the activation of ’f2’. For example, if ’g1’ always output the logic value one, ’f2’ would not be activated for Stuck-at-1 faults. Consequently, a Stuck-at-1 fault in ’f2’ would be Structural-Safe.
3) Characteristics of the combinational logic ’g2’ could block propagation of a fault in ’f3’. If, for example, ’g2’ is an AND gate, with one of the inputs always set with the logic value zero, the effect of a fault in ’f3’ would never propagate to ’out0’. Therefore, ’f3’ would be Structural-Safe for Stuck-at-1 and Stuck-at-0 faults.

2) Formal Analysis: The formal analysis deploys formal techniques to investigate the behavior of a design under fault. The fundamental theory consists in creating a representation of the boolean function implemented by the design under test, where formal proves can be deployed. Modern formal tools employ different formal techniques to achieve better

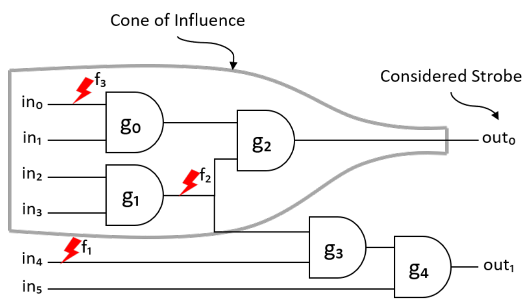

Fig. 1. Structural Analysis Example.

performance. Although details of implementation are not disclosed, common forms of design representation are Binary Decision Diagrams (BDDs) [16] and Multiway Decision Graphs (MDGs) [17].

Two copies of the design model are built for formal analysis: the Good Machine and the Bad Machine. The same inputs and constraints are deployed on both models. Fault effects are applied in the Bad Machine only and the Strobe point of both copies are monitored. A difference in the Strobe Points indicate the propagation of the fault.

The formal analysis deploys formal methods to determine the Activation and Propagation of faults. Activation Analysis indicates whether the fault can be functionally activated by any combination of inputs. Propagation Analysis verifies if there is a combination of inputs that provoke fault propagation. Formal analysis will classify the faults, which were not previously classified by the Structural Analysis, in three groups:

- Safe: Faults that cannot be activated or propagated.
- Dangerous: The tool identified at least one combination of test inputs that results in fault propagation.
- Unknown: All the others.

Formal properties to perform the analysis are automatically generated and verified with respect to all possible input stimuli. The formal analysis relies on formal properties and verification to prove the properties to be true.

The characteristics of Formal Methods analysis distinguish critical behavioral information of a design under the effect of faults. By investigating this data, we can understand the fault propagation effects on the different parts of the hardware design. These aspects are deployed for the netlist characterization, as described in Section III-A.

B. Fault Injection Simulation

Analysis of Fault Injection (FI) by Simulation is widely used and available in a variety of tools. These tools are able to analyze a Register Transfer Level (RTL) or Gate-Level (GTL) descriptions of an IC and, based on given test inputs, simulate their behavior. The effect that a fault produces in the design is determined by comparing the behavior of the design with and without faults. The flow implemented by FI Simulation Tools is described below:

1) Elaboration of RTL/GTL design description.

2) Fault List Generation: candidates for fault injection are defined for each available fault model. The user should define rules (e.g. all signals) to identify fault node candidates and fault models (e.g. Stuck-at-0 (SA0) and Stuck-at-1 (SA1)). Information is stored in a fault database.
3) Fault List Optimization: Faults list is analyzed to identify candidates for optimization. Based on the elaboration results, tools can estimate the behavior of some faults decreasing the number of faults to be simulated. Information is updated on the fault database.
4) Good Simulation: fault-free behavior of design is simulated. The user should define observation points in the design to identify: (1) Fault propagation to a functional output: functional strobes; (2) Activation of the Safety Mechanism: checker strobes. The values of the Strobes during good simulation are stored.
5) Fault Injection Simulation: For each fault in the fault database, the design faulty behavior is simulated, and the observation points compared against the reference values from the Good Simulation. The behavior of the design under each fault is analyzed and stored.

FI Simulation determines the behavior change provoked by a fault when the effect is observable in one of the outputs (strobes). If the fault effect is observed in a Functional Strobe, the fault is considered Dangerous. If the effect is perceived in a Safety Mechanism Strobe, the fault is classified as Detected. Faults that don't produce changes in the strobes are classified as Undetected. This is considered a weak result of the simulation, as a different test may cause fault propagation.

Differently from formal methods that only considers characteristics of the netlist, the simulation also reflects the test stimuli (workload deployed for simulation) effect for fault behavior analysis. This property allows the characterization of the workload concerning fault activation and propagation, as described in Section III-B.

III. FLIP FLOP WEIGHTING METHODOLOGY

Fault analysis targeting ISO26262 compliance aims to identify faults that can propagate to safe-related outputs of the system. As these faults may disturb a Safety Goal, the design should include mechanisms to detect and control them maintaining a safe state. The propagation of the fault effect depends on the hardware characteristics and the stimuli applied to the netlist. These two aspects are fundamental to understand the behavior of a design under the influence of faults.

Aiming to provide an estimation regarding the number of Detected faults, without the need to execute a full Fault Injection campaign, we propose the Flip Flop Weighting methodology. Figure 2, illustrated the proposed flow. Initially, weights for selected hardware nodes are calculated based on the netlist characteristics. After, we cover the workload contribution by analyzing fault activation and the propagation of the selected hardware nodes to the circuit outputs. The combination of netlist and workload characterization allows

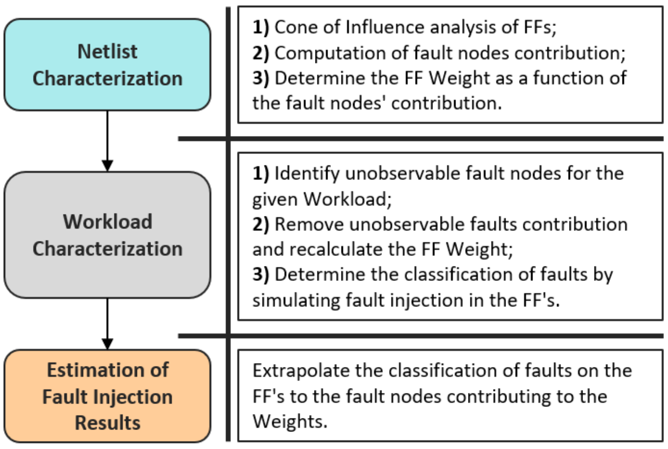
Netlist Characterization
1) Cone of Influence analysis of FFs;
2) Computation of fault nodes contribution;
3) Determine the FF Weight as a function of the fault nodes' contribution.
Workload Characterization
1) Identify unobservable fault nodes for the given Workload;
2) Remove unobservable faults contribution and recalculate the FF Weight;
3) Determine the classification of faults by simulating fault injection in the FF's.
Estimation of Fault Injection Results
Extrapolate the classification of faults on the FF's to the fault nodes contributing to the Weights.

Fig. 2. Flip Flop Weighting Methodology Flow.

the estimation of Detected faults for a given hardware netlist and workload.

A. Netlist Characterization

The netlist characterization examines the number of faults that can reach each sequential element of the design. Thus, we can calculate a weight representing all fault nodes as a function of the sequential elements. The netlist characterization is based on the physical attributes of the hardware design and doesn't consider the influence of the workload. Formal Methods were the selected technology to extract the required information from the netlist. This work deploys the Functional Safety Verification (FSV) application from Cadence®JasperGold (JG) Formal Verification Platform.

As required by ISO26262, we must consider all cell ports on the Gate-level representation of the design as fault nodes. Therefore, the netlist characterization analyzes the results of the elaboration of the Gate-level circuit description. All cell ports are identified and included in the fault list as targets for Stuck-at-0 and Stuck-at-1 fault models. Next, the fault list is optimized with the results of the Structural Analysis, as described in II-A1. Formal Methods can prove that Safe faults cannot propagate to design outputs. Therefore, these faults can be ignored during the netlist characterization reducing the fault propagation estimation error.

The next step is the analysis of the Cone of Influence (CoI) of each prime propagation node. The CoI is a list with all the fault nodes with a physical connection to a given hardware element. In general, the CoI analysis is based on the primary outputs of the design, aiming to identify Safe faults. However, this work deploys the CoI analysis to understand the propagation of faults to particular hardware components. The formal tool enables the identification of all sequential elements in the hardware design. From these, we classify all Flip Flops and primary output ports as prime propagation nodes. In this context, the output ports are included with the Flip Flops to cover any fault nodes with a direct connection to the primary outputs. For each component classified as a prime propagation node, the analysis continues as follows:

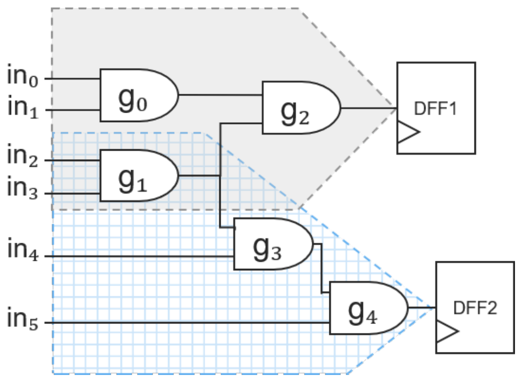

Fig. 3. Flip Flop Cone of Influence example.

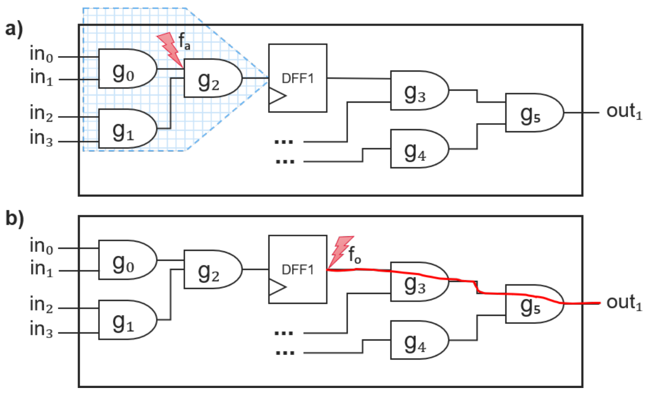

Fig. 4. Fault Activation and Propagation example.

1) Extract the fault nodes inside the CoI
2) Remove sequential elements as they are part of the prime propagation nodes list
3) Compute all remaining fault nodes

After the analysis of all prime propagation nodes, we have collected the required information for calculation of the Flip Flop Weights. The FF Weight represents the number of faults that can propagate to a given Flip Flop. A fault node that propagates to only one FF has a weight contribution of one. Faults nodes with a physical propagation path to multiple FFs, need to have their weight contribution calculated based on the number of FFs they can affect. Figure 3 illustrates the FF Weight contribution on an example circuit.

Figure 3 shows an example design containing Flip Flops (DFF) and gates (g). The figure also highlights the Cone of Influence (CoI) of each FF. The solid pattern represents the CoI of 'DFF1', while the checkered pattern represents the CoI of 'DFF2'. The gate 'g1' is inside both CoI, implying that a fault in its ports can affect both 'DFF1' and 'DFF2'. As previously described, a fault node that propagates to only one FF has a weight contribution of one. Considering that each gate (g) contains three fault nodes, we can calculate that 'DFF1' has a starting weight of six, from the faults in 'g0' and 'g2', while 'DFF2' has a starting Weight of 6, from the faults in 'g3' and 'g4'. To calculate the weight contribution of 'g1', as it is inside of multiple CoI, we need to divide the number of fault nodes by the number of FF they can affect. In the example, as 'g1' is inside two CoI, any fault node in the gate will have a weight contribution of 0,5. Consequently, 'g1' will have an addition of 1,5 to 'DFF1' and 'DFF2'. It is important to highlight that this analysis should be repeated to every analyzed fault model. Finally, considering a single fault model, the netlist characterization of the example design would result in a final FF Weight of 7,5 for both 'DFF1' and 'DFF2'.

The netlist characterization will result in a representation of the fault propagation potential of a given circuit, by only their prime propagation nodes (flip flops and primary outputs). Each prime propagation node will have a Weight that expresses the number of faults factored by the node. Therefore, we can estimate the classification of all fault targets in the design by analyzing fault effects in the prime propagation nodes. After understanding how the physical characteristics of the circuit impact in the fault behavior, it is necessary to compute the contribution of the workload regarding fault activation and propagation.

B. Workload Characterization

The workload has an essential role in fault classification analysis. The stimuli applied by the Testbench for simulation of the circuit behavior determine if a fault node will be activated and also how a fault will propagate. The combination of test inputs will determine the logic value on the gate ports. In case this value is a constant, a stuck-at-fault with the same logic value is never activated. Also, the test inputs can put the design in a state that can mask the propagation of fault effects. Therefore, the workload analysis is essential to understand how a fault can affect the behavior of a given circuit.

The chosen technology for the analysis of the workload is Fault Injection Simulation. By deploying FI Simulation, it is possible to determine the effect that a fault produces by comparing the behavior of the design with and without the fault injection. Our work deploys Cadence®Xcelium™Fault Simulator (XFS) as the representative of this technology.

1) Fault Activation: The test stimuli applied to the circuit will determine if a fault will be activated. Hence, aiming to identify fault nodes that are not stimulated by the current workload, we need to analyze the inputs of the circuit during the simulation.

The fault activation analysis starts by identifying test inputs that are constant during the simulation of the design. Then, this information is applied to perform the testability analysis. The testability analysis identifies faults that are unobservable for a given workload. Figure 4-a, illustrates an example of fault activation analysis. In the example, a fault node 'fa' depends on the values 'in0' and 'in1' to be activated. Any combination of the inputs that produce a logic value zero, in the output of the gate 'g0', results in an unobservable Stuck-at-0 fault in 'fa'. The same would be true for a Stuck-at-1 fault if 'g0' outputs a constant logic value one.

By examining all test inputs and identifying signals that are constants throughout the circuit, it is possible to classify several fault nodes that are unobservable for a specific fault model. As the workload never activates these fault nodes, we can conclude that they will never propagate to circuit outputs during Fault Injection Simulation. Therefore we can determine that these faults will be Undetected when simulating the current workload.

After identifying all unobservable fault nodes for a given workload, we need to consider them for the Flip Flop Weight calculation. All faults identified as unobservable will have a Weight contribution of zero. Consequently, we can recalculate the Flip Flop Weight from the netlist characterization step, with the inputs from the activation analysis. The result Flip Flop Weight is a representation of the circuit comprising the workload stimuli information.

2) Fault Propagation: After calculating the fault weight of each prime propagation node and optimizing the results by computing the contribution of not activated faults, we need to investigate the effects of fault propagation. The fault propagation analysis starts by identifying a fault target for each prime propagation node. In general, the fault targets will be the output port of Flip Flops and the primary output ports of the circuit. Next, the propagation of each prime propagation node is determined by Fault Injection Simulation.

Figure 4-b illustrates the propagation of a fault 'fo' injected in the output port of 'DFF1' to the circuit output 'out1'. The observability of the effects of 'fo' in 'out1' depends on the logic level of the gates 'g3' and 'g5'. As the workload is responsible for setting the logic level of these gates, the simulation can confirm the propagation of the fault to the output.

C. Estimation of Fault Injection Results

With the determination of the fault effects on the prime propagation nodes, we can estimate the fault classification for all faults in the circuit. For example, 'DFF1' in Figure 4-b has a final Flip Flop Weight of 9, and the fault classification of a fault injected in 'DFF1' output is *Detected*; in such a case, we can conclude that the annotation *Detected*, is valid for the all faults represented by 'DFF1'. By repeating this analysis to all prime propagation nodes, we can estimate the classification of all faults in the circuit.

IV. RESULTS

The validation of the proposed methodology consists of a comparison between estimated fault classification results and actual fault injection results on target designs. By incorporating the actual results, we can analyze the estimation error rate and also the performance improvements achieved by the Flip Flop Weighting. The selection of the target designs objects to include circuits with different physical characteristics and multiple simulation workloads. The ac97 is an Audio Codec Controller IP compatible with a wishbone bus. It includes a functional testbench that verifies all circuit functionalities and an ATPG simulation environment [18]. The conmax is an interconnect matrix IP core featuring a parameterized priority-based arbiter, including a functional testbench [18]. Finally, the AutoSoC is an open-source initiative for an automotive SoC benchmark suite. The AutoSoC CPU is based on an OpenRISC implementation, it includes a full automotive SoC with several simulation features and workloads [19].

TABLE I
DETAILED AUTOSOC WEIGHTING ANALYSIS

AutoSoC Analysis (SA0/1)	Prime Nodes	Total Weight	Max Weight	Average Weight
Netlist Characterization	4648	83427	1806,4	33,8
NL+WL helloWorld	4648	66240	1232,2	26,8
NL+WL calcPrime	4648	67314	1301,9	27,3
NL+WL Test Library	4648	76110	1806,4	30,8

Table I details the results of the netlist characterization and the combination between netlist (NL) and workload (WL) characterizations for the AutoSoC. The results from the netlist characterization highlight the fault propagation potential of the circuit. If a workload could activate and propagate all faults in the design, the Total Weight would describe the number of detected faults as it represents the total number of fault nodes after the Structural Analysis optimizations. The column Max Weight promotes the identification of critical nodes in the design, as it illustrates the maximum weight in a single node. The workload characterization step recalculates the parameters considering the activation of fault nodes. Therefore, the NL+WL rows represent the maximum fault detection coverage when simulating the given workload. These parameters can be applied to rank the workloads by their fault detection rate potential. The workload with resulting parameters closest to the netlist characterization parameters will be most likely to have higher fault detection rates.

The FI Simulation employs SA0 and SA1 faults on every cell port of the Gate-level representation of the hardware designs. At the end of each campaign, we collect the actual fault classification results for comparison with the estimated results from the Flip Flop Weighting analysis. Table II summarizes the results, the columns labeled as "Actual" represents the results from the FI Simulation, while the columns labeled as "Estimated" outlines the results from the Flip Flop Weighting. As illustrated in II-B, the classification of a fault depends on the configuration of the strobes. To facilitate the evaluation of the methodology, a single strobe type is declared in the FI Simulation. Therefore, all fault effects observable in the strobes are classified as Detected.

The execution time of the FI Simulation campaigns depends on several factors. However, the most critical parameter is the availability of resources for executing parallel simulations. For the designs with a faster simulation time, the ac97 and the conmax, the FI Simulation campaign is configured sequentially, resulting in execution time in the granularity of days to a week. For the AutoSoC, which contemplates the simulation of a full SoC, the FI simulation is configured in concurrent mode, executing up to 100 faults in parallel, resulting in execution

TABLE II
RESULTS SUMMARY

Design	Workload	Total Faults SA0/SA1	Actual Detected	Actual Detected (%)	Estimated Detected	Estimated Detected (%)	Estimation Error	Exec. Time Improvement
ac97	Funct TB	57220	39863	69,67%	42166	73,69%	4,02%	8,5X
	ATPG TB	57220	57091	99,77%	56923	99,48%	-0,29%	5,8X
conmax	Funct TB	153454	123796	80,67%	133762	87,17%	6,49%	12,8X
autoSoC	helloWorld	96354	38964	40,44%	41109	42,66%	2,23%	17,8X
	calcPrime	96354	45362	47,08%	46198	47,95%	0,87%	19,1X
	Test Library	96354	61230	63,55%	67083	69,62%	6,07%	20,4X

times in the granularity of weeks to a month. In all cases, the Flip Flop Weighting technique execution time is faster than the FI Simulation campaigns. Table II highlights the main achievements from the validation of the Flip Flop Weighting:

- Accuracy: The column ”Estimation Error” highlights the difference between the Estimated fault detection rates and the Actual fault detection;
- Efficiency: The column ”Exec. Time Improvement” notes the performance gain when deploying the proposed methodology.

As illustrated in Table II, the results achieved by the proposed methodology are encouraging. The Flip Flop Weighting estimates fault detection rates with an accuracy between 0,2% and 6,4%, with an up to 20X faster execution times. These figures allow for a confident design space exploration of a hardware design concerning the safety metrics. By deploying such a technique, safety-engineers can explore diverse architecture possibilities with a higher degree of certainty. Even though the hardware design must undergo Functional Safety verification at the final stages of development, the feasibility of an early estimation of the safety-metrics sustains safety-related architectural decisions. The next step of our work is to explore alternatives for fault propagation analysis. Even though FI Simulation is a powerful option, it demands long execution times for gathering the classification of the prime propagation nodes. An alternative technique could improve our efficiency, enhancing the potential of the proposed methodology.

V. CONCLUSIONS

Functional safety verification is a critical step for ISO26262 compliance. At later stages of safety-critical systems development, designers must analyze the behavior of the design under the effect of faults to show conformity with the expected safety metrics. Failing to achieve these conditions entails additional iterations through critical development and verification phases. Our work proposes a methodology for the design space exploration of safety architectures. By allowing engineers to estimate safety metrics before the final development stages, we provide a tool for the investigation of safety architectures, improving the confidence in conceptual decisions and decreasing the chances of rework. Even though our work targets Automotive applications, it also applies to other safety-critical domains with similar requirements as Aviation, Medicine, Space, among others. Our results demonstrate the accuracy of the technique by providing an estimation of the fault detection rate with an average error of 3%. Moreover, the methodology results in an execution time up to 20X faster when compared with the traditional Fault Injection campaigns.

ACKNOWLEDGMENT

This research was supported by project RESCUE funded from the European Union’s Horizon 2020 research and innovation programme under the Marie Sklodowaska-Curie grant agreement No 722325.

REFERENCES

[1] 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.

[2] S. Pateras and T.-P. Tai, “Automotive semiconductor test,” in *2017 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*. IEEE, apr 2017.

[3] D. Alexandrescu, A. Evans, M. Glorieux, and I. Nofal, “EDA support for functional safety — How static and dynamic failure analysis can improve productivity in the assessment of functional safety,” in *2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, jul 2017.

[4] 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.

[5] F. Augusto da Silva, A. C. Bagbaba, S. Sartoni, R. Cantoro, M. S. Reorda, S. Hamdioui, and C. Sauer, “Determined-safe faults identification: A step towards ISO26262 hardware compliant designs,” in *2020 IEEE European Test Symposium (ETS)*. IEEE, may 2020.

[6] M. Syal and M. Hsiao, “New techniques for untestable fault identification in sequential circuits,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 6, pp. 1117–1131, jun 2006.

[7] H.-C. Liang, C. L. Lee, and J. Chen, “Identifying untestable faults in sequential circuits,” *IEEE Design & Test of Computers*, vol. 12, no. 3, pp. 14–23, 1995.

[8] F. Augusto da Silva, A. C. Bagbaba, S. Hamdioui, and C. Sauer, “Combining fault analysis technologies for ISO26262 functional safety verification,” in *2019 IEEE 28th Asian Test Symposium (ATS)*. IEEE, dec 2019.

[9] 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.

[10] A. Bernardini, W. Ecker, and U. Schlichtmann, “Where formal verification can help in functional safety analysis,” in *Proceedings of the 35th International Conference on Computer-Aided Design - ICCAD*. ACM Press, 2016.

[11] E. Kang, E. Jackson, and W. Schulte, “An approach for effective design space exploration,” in *Foundations of Computer Software. Modeling, Development, and Verification of Adaptive Systems*. Springer Berlin Heidelberg, 2011, pp. 33–54.

[12] S. Chtourou and O. Hammami, “SystemC space exploration of behavioral synthesis options on area, performance and power consumption,” in *2005 International Conference on Microelectronics*. IEEE.

[13] B. C. Schafer, “Probabilistic multiknob high-level synthesis design space exploration acceleration,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 3, pp. 394–406, mar 2016.

[14] K. Roy, H. T. Mert, and M. Swaminathan, “Preliminary application of deep learning to design space exploration,” in *2018 IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS)*. IEEE, dec 2018.

[15] J. Raik, H. Fujiwara, R. Ubar, and A. Krivenko, “Untestable fault identification in sequential circuits using model-checking,” in *2008 17th Asian Test Symposium*. IEEE, nov 2008.

[16] G. Cabodi and M. Murciano, “BDD-based hardware verification,” in *Formal Methods for Hardware Verification*. Springer Berlin Heidelberg, 2006, pp. 78–107.

[17] F. Corella, Z. Zhou, X. Song, M. Langevin, and E. Cerny, “Multiway decision graphs for automated hardware verification,” *Formal Methods in System Design*, vol. 10, no. 1, pp. 7–46, 1997.

[18] C. R. Berkeley, “International Workshop on Logic and Synthesis (IWLS) 2005 benchmarks,” Tech. Rep., 2005.

[19] F. Augusto da Silva, A. C. Bagbaba, A. Ruospo, R. Mariani, G. Kanawati, E. Sanchez, M. Sonza Reorda, M. Jenihhin, S. Hamdioui, and C. Sauer, “Special session: AutoSoC - a suite of open-source automotive SoC benchmarks,” in *2020 IEEE 38th VLSI Test Symposium (VTS)*. IEEE, apr 2020.