Formal Assisted Fault Campaign for ISO26262 Certification

Nitin Ahuja, Synopsys, Noida, India (nitin.ahuja@synopsys.com)

Mayank Agarwal, NXP Semiconductors, Noida, India (mayank.agarwal@nxp.com)

Sandeep Jana, Synopsys, Bangalore, India (sjana@synopsys.com)

***Abstract*— For ASIL C &D devices ISO26262 recommends Fault Injection (FI) testing as a methodology to measure the effectiveness and robustness of the safety mechanism. Diagnostic coverage is the quantitative measure of the effectiveness of the safety mechanism, that is defined as the number of faults observed or diagnosed by the safety mechanism. Conventionally fault campaign or fault injection flows are simulation based and they suffer with exhaustivity issues and hence the faults that are not observed needs to be manually analyzed. As a methodology we propose using formal for such faults to find what amount of not observed faults are untestable by design.**

Keywords—Functional safety, FuSa, diagnostic coverage, formal FuSa, ISO26262

I. INTRODUCTION

Today's vehicles are no longer only mechanical, major part of it is electronics. With the demand for the autonomous vehicle and in a need to make roads safer, the amount of electronics that are going into the automotive is immense!

Electronics in modern day cars is not limited to infotainment systems or information console, a lot of crucial operations are controlled and assisted by electronics. There is a growing concept of by-wire in the automotive industry for example, break-by-wire [1]. Brake by wire technology has a small electric motor near the wheels that generate the braking pressure. They are governed by electronic control units connected to the brake pedal that receive input from the driver's actions and interpret that information into a series of electronic instructions. These instructions are then communicated through a miniature, real time network connecting the entire braking system. Already more than 30% of car's manufacturing cost is that of electronic components and is expected to go to 50% in a decade's time [2]. This is huge amount of electronics in the cars! This is the same reason why there is a need for functional safety verification because electronics tend to fail no matter how much we verify them before production.

ISO 26262 is the standard for automotive functional safety that provides an automotive safety lifecycle, i.e. management, development production, operation, service, decommissioning, and supports tailoring the necessary activities during the lifecycle phases [3].

II. PROBLEM STATEMENT

The design under consideration for the safety analysis is an ECC block which has SECDED (Single error correction and Double error detection) scheme. The block has 8K memory attached to it.

The intention of the safety mechanism is to correct any single bit error in the data and raise the corresponding error signal on the ECC boundary to alert the system of occurrence of fault in the system. While in case of multi bit fault, the corrupted data is not corrected but a corresponding error signal is raised to alert the system of occurrence of fault in the system.

For fault classification Stuck at 0 & Stuck at 1 fault at all the input and output ports, flops and internal wires of the design are inserted and then fault simulation is run to see how many of the faults are observed and detected. Hence, gives evidence of the robustness of the Safety Mechanism.

Faults need to be classified as safe or dangerous base on which quadrant the fall in from the classification shown below.

Table 1 Four Quadrant for fault classification

	Faults not detected at the error signal	**Faults detected at the error signal**
Faults not detected at output data port	Fault is not observed at the output and is also not observed at the error signal. This means this fault cannot corrupt an otherwise normal transaction. => **SAFE** (Potential Latent Fault that needs alternative diagnostic mechanism such as EIM or LBIST)	Fault is not observed at the output but is observed at error pins. So, this is not a dangerous fault but triggered diagnosis which is OK. => **SAFE** (Potential Latent Fault that is diagnosed)
Faults detected at output data port	Fault is observed at the output but is not observed at the error signal. This means this fault can corrupt an otherwise normal transaction without any indication. => **DANGEROUS** (Single Point Fault)	Fault is not observed at the output and is also not observed at the error signal. This means this fault cannot corrupt an otherwise normal transaction. => **SAFE** (Potential Latent Fault that needs alternative diagnostic mechanism such as EIM or LBIST)

The block in consideration qualifies to be formal friendly except the 8K memory plugged at its periphery, hence formal only approach was used with correct methodology in place to deal with memory complexity.
Block diagram of the design in consideration looks like as follows:

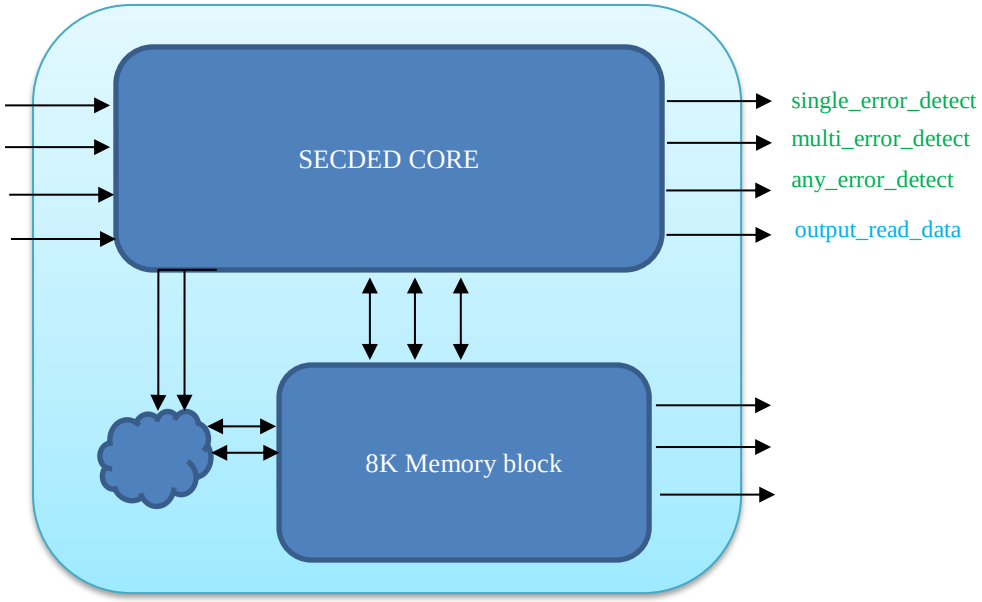

Figure 1 SECDED Block Diagram

ECC error signals, multi_error_detect, single_error_detect, , any_error_detect, marked in green were considered as detection or diagnostic points while output_read_data marked in blue was taken as observation point for the fault analysis.

III. BACKGROUND

ISO 26262 makes use of HARA (Hazard Analysis and Risk Assessment) to identify the hazards and define safety goals to mitigate the unreasonable risk. ISO 26262 defines functional safety as absence of *unreasonable risk* (3.176) due to *hazards* caused by *malfunctioning behaviour* (3.88) of *E/E systems* (3.40). In simple words, in case of any potentially dangerous condition, there should be a safety mechanism in the system that can identify the condition and take preventive or corrective mechanism to any risk to the human life.

ISO classify the items based on the risk assessment of the potential hazard by looking at factors such as severity, exposure and controllability in 4 different levels of severity known as ASIL [4] (Automotive Safety Integrity Level), ranging from ASIL A to ASIL D, ASIL D being most critical and A being most relaxed.
Part 5 of ISO26262 defines "Product Development at Hardware phase" and associated processes. It talks about the need to perform random fault analysis and provide evidence of safety mechanism being robust enough to detect enough random faults and take corrective measures. Based on the class of ASIL ISO recommends the fault injection testing to verify the effectiveness of the safety mechanism [5].

Table 2 ISO 26262 Recommendation For Fault Injection Testing

ISO Recommendation	**ASIL A**	**ASIL B**	**ASIL C**	**ASIL D**
Fault Injection Testing	+	+	++	++

+ : method is recommended for specified ASIL
++ : method is highly recommended for specified ASIL

Devices needs to provide evidence of the effectiveness of implemented safety mechanism to prevent faults from leading to single-point failures or from being latent. Various metric that are needed as evidence for ISO certification are:

1) Diagnostic Coverage
2) Single point fault metric (SPFM)
3) Latent fault metric (LFM)
4) Probabilistic Metric for Random Hardware Failures (PMHF)

Table shows the required %of various metrics discussed above for various ASIL

Table 3 SPFM, LFM, PMHF requirement

	ASIL B	**ASIL C**	**ASIL D**
Single Point Fault Metric (SPFM)	>= 90% +	>= 97% ++	>= 99% ++
Latent Fault Metric (LFM)	>= 60% +	>= 80% +	>= 90% ++
Prob. Metric HW Failure (PMHF)	<100 (FIT)	<100 (FIT)	<10 (FIT)

Conventionally way of fault injection testing is through simulation. Where fault simulator inserts the faults to create the faulty model or bad machine. Test pattern developed by the verification engineer in consultation with functional safety engineer is applied and the output at the observation points (defined by user) from faulty machine is compared with that of the good machine and in case there is divergence from the actual behavior the fault is termed to be observed. Once observed, the fault is expected to hit the diagnostic point or the detection point which is the safety mechanism for it to take appropriate action.
Let's consider the figure below where 4 faults are inserted in the design, faults F1 through F4.

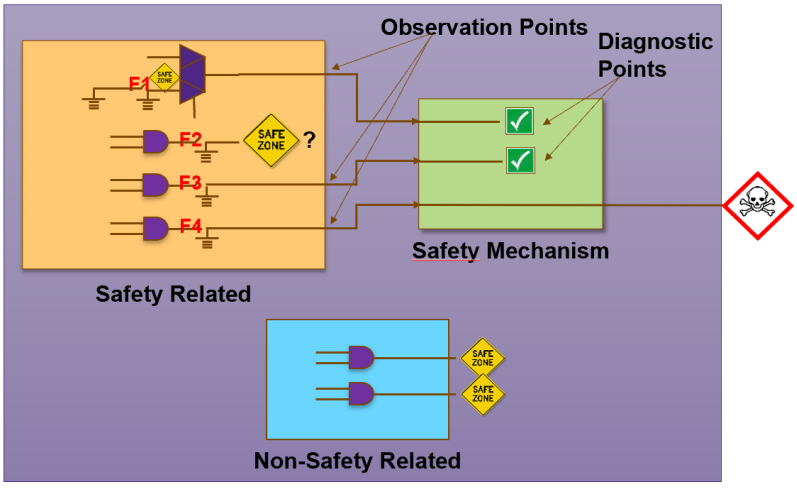

Figure 2 Fault Classification

On application of the test pattern fault F1, F3 and F4 are propagated to the observation points and hence known as the observed faults. Once they are observed, Fault F1 and F3 are detected by the safety mechanism. Depending on what corrective active safety mechanism take, faults can either be safe detected or dangerous detected. But the faults of the nature F4 which are observed at the observation point, i.e. they hamper the default behavior of the device but are not caught by the safety mechanism, are the dangerous undetected faults that needs to be accounted for. All the Faults F1, F3 and F4 have deterministic behavior, i.e. they are either safe or dangerous because they are activated and propagated by the applied test pattern. Fault F2 is neither observed at the observation points nor detected by the safety mechanism, such faults are of indeterministic nature and needs manual analysis.

This approach suffers from the issue of identification or generation of test patterns to excite the injected faults and propagate them to the diagnostic points, leading to lower diagnostic coverage. During the start of the simulation cycle the progress is generally linear, but soon, simulation tests hit a limit where we start seeing diminishing results until we reach the plateau where the gains are negligible with passage of time.

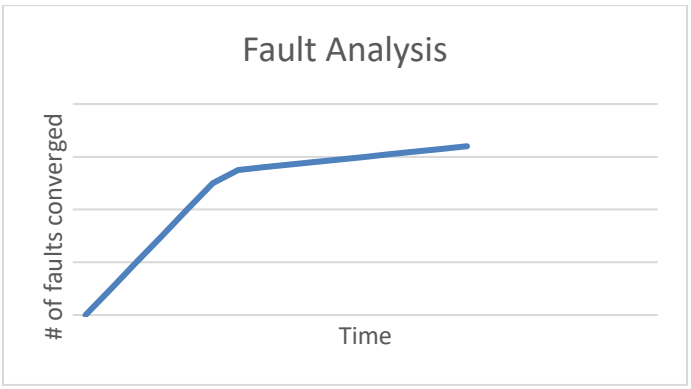

Figure 3 Simulation Fault Analysis Trend

Verification engineer needs to go and do manual analysis of each and every fault that is not observed by the fault simulation to make sure the fault was not observed because of the nature of the design and not due to non-exhaustivity of the simulation environment.

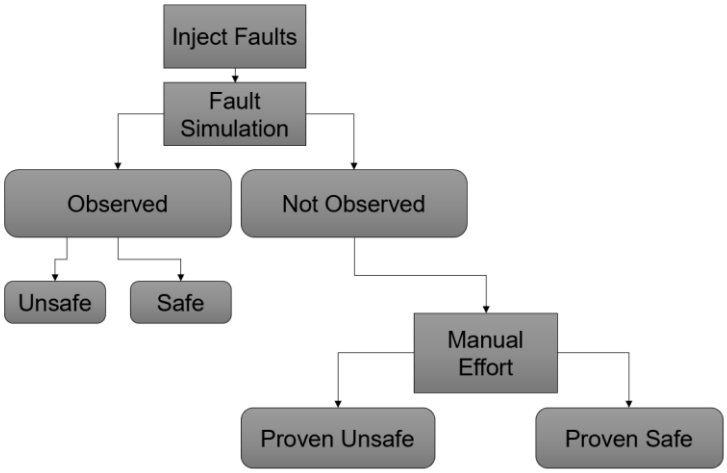

Figure 4 Simulation Fault Injection Testing Pain Point

This manual effort of reviewing all not-observed faults is not trivial and needs several man days to man weeks.

In this paper we propose the use of Formal methods to classify the indeterministic faults left from simulation based fault campaigning as safe or dangerous based on whether the faults are blocked by design or can be formally observable. The paper also proposes to make use of Formal as a standalone in case the block is small and is suitable for formal analysis, we'll be talking in brief about how to classify the block as formal friendly or hard for formal.

For a block to be classified as the formal friendly or good for formal one needs to do some design state space analysis and then take a call. As a general recommendation, a block that satisfies the following criteria is good for formal:

1) Control centric
2) IP Level designs suits the best going to Subsystem and then to SoC being very complex
3) No data transformation
4) No or small sequential depth
5) No or small sequential elements
    a. Memories
    b. Counters
    c. Multipliers
    d. Fifo

For the blocks that qualify the above criteria can be used by formal only approach. For the blocks that are complex hybrid formal approach can be used where some fault filtering can be done at Simulation and formal can be used to identify the safe faults to get the boost in diagnostic coverage.

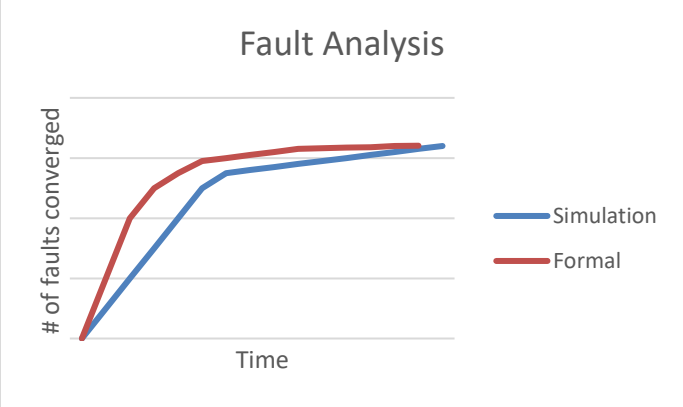

Figure 5 Boost With Formal Fault Classification Flow

The methodologies discussed in paper applies to both the scenarios.

IV. FORMAL FAULT CLASSIFICATION FLOW

Using VC Formal FuSa flow, faults are classified as a three-step flow:

1) Structural Observability
   All the faults that lie outside the logic cone or Cone Of Influence (COI) of the union of observation points, can never influence observation points. Such faults are marked as untestable and hence can be bucketed as safe faults as they can never cause malfunctioning of the device.
   This particular step is not Formal but static in nature where the logic cone is assessed statically and hence it scales really well to all the design abstractions ranging from IP to SoC and can also handle huge fault space.

2) Formal Observability
   The faults that are structurally observable at the observation points are formally assessed. In formal observability, a fault is assessed based on the fact that if it can cause a deviation in the actual behavior of the design and causes a change in observation point from the non-fault design, then this fault is propagated to the observation point and hence is dangerous. But if it does not create any difference what so ever from the actual design behavior, it cannot propagate to the observation point and hence is untestable. Such fault can safely be bucketed as safe fault.

3) Formal Detectability
   The faults that propagate to the observation point are expected to hit the detection points at the SM (Safety Mechanism) to trigger the corrective or preventive mechanism. The faults that are observed at the observation points but are not detected at the detection point are really dangerous faults because they causes the malfunctioning in the system but are not caught by safety mechanism.

Step 2 and 3 are hard formal problem and needs correct methodology to be successful, otherwise it could run into convergence issues.
This paper will describe the methodology used to tackle the design complexity.

V. METHODLOGY AND RESULTS

Four phase approach was used to do the complete fault classification, with each phase more and more faults are found to be safe.
Before running any fault analysis, fault list is pruned by the fault compiler which does the first level of testability analysis by considering all the outputs of the block as observation point and then find out the faults that are untestable. It also does fault collapsing to collapse the equivalent faults
Faults are classified as either prime or collapsed. A prime fault is a fault which represents one or more faults. A collapsed fault is a fault which produces the same observable behavior as its equivalent prime fault. An untestable fault is a fault which cannot be detected under any condition. Only prime faults are considered for fault classification and collapsed fault will have same behavior as its prime fault.

Table 4 Result of Testability Analysis

Fault types	Total Faults	Collapsed Faults	Untestable Faults	Residual faults
Port, Flop & Wire	3606	973	144	2489

Phase 1 Structural Observability:
Output read data port (output_read_data) is considered as observation point and any fault that lie outside the COI is marked as untestable.

Table 5 Phase1 : Structural Observability Results

Fault types	Total faults for formal analysis	Structurally non-observable Untestable (UU)	Structurally Observable
Port, Flop & Wire	2489	763	1726

Phase 2 Formal Observability
Since this design had 8K memory which is a formal bottleneck, it tend to hit the convergence issues. With original observation point and taking complete design in consideration it could not converge in even 12 hours.
We deployed intelligent blackboxing where we blackboxed the memory and added its input port as new observation points. Memory inputs were added as new observation points to make sure that the faults that were in the memory write path are not falsely marked as safe as they can take effect once the memory read is issued.

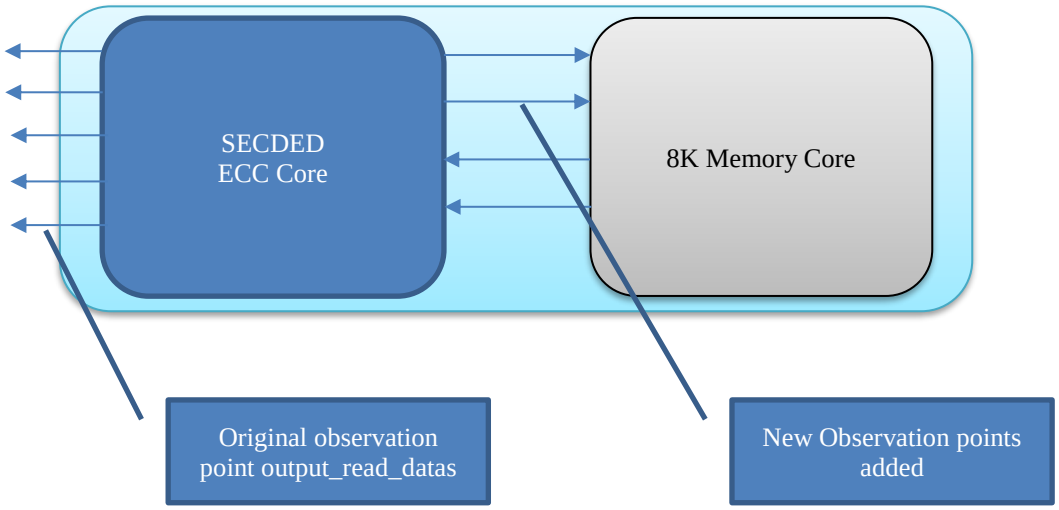

Figure 6 Intelligent Blackboxing Methodology

Adding new observation points can give us false negatives and not false positives in terms of number of faults being safe. Any fault that is marked safe will still be safe as if it is not observable at the memory input it can never be observable at the red data port. With this abstraction we were able to find out many faults that were not

observable and hence safe. The faults that are marked observable still needs assessment with original observation points.

Table 6 Phase2 : Formal Observability Results

Fault types	Structurally observable faults	Formally not observable	Formally Observable
Port, Flop & Wire	1726	232	1494

Since the observation points used are not the original observation points but intermediate nodes, observable faults are not the actual observable faults and need further analysis.

Phase 3:

Taking the original observation point and abreacting the memory with back to back flops.

In this case back to back flops were used to replace the memory and to short the read and write path. Additional constraint was added to make sure there is always a read followed by write. Thus reducing the formal complexity by many folds.

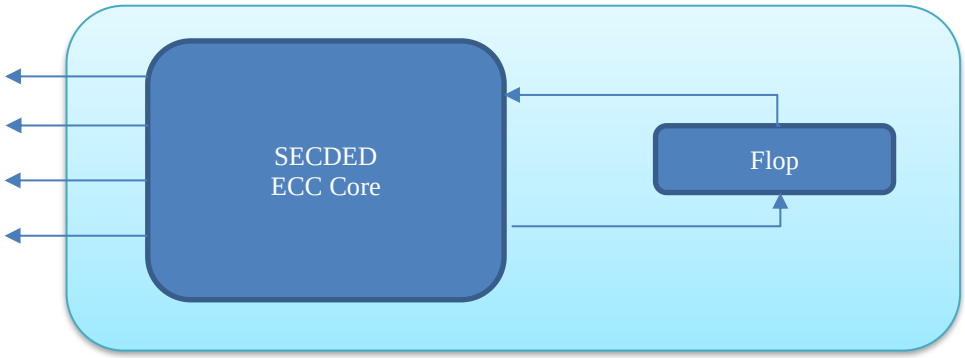

Figure 7 Abstraction Methodology to deal with Complexity

With these changes, more safe faults were identified and the faults that effect the observation point, output_read_data, were marked as observed

Table 7 Phase3 : Formal Observability Results

Fault types	Faults in consideration	Formally not observable	Formally Observable
Port, Flop & Wire	1494	16	1478

Phase 4: Formal detection analysis

The faults that were formally observed and not observed at the observation points were used for the detection analysis where ECC error ports were used as diagnostic or detection points.

At the end of the fault classification we could find 512 such faults that causes the corruption of data but does not raise the ECC error signals and hence are dangerous.

Table 8 Phase4 : Formal Detection Analysis Results

Fault types	Formally Observable and not observable	Not Observable and Not detected (NN)	Not Observed but detected (ND)	Observed but not detected (ON)	Observed and detected (OD)
Port, Flop & Wire	1726	225	23	512	966

VI. CONCLUSION

Formal analysis with correct methodology can be leveraged for fault classification to get the desired diagnostic coverage and in turn reducing numerous man hours and tedious manual analysis of the not observed faults.

REFERENCES

[1] https://www.brakebywire.com/how-brake-by-wire-works.html
[2] https://www.statista.com/statistics/277931/automotive-electronics-cost-as-a-share-of-total-car-cost-worldwide/
[3] https://www.tuvsud.com/-/media/global/pdf-files/whitepaper-report-e-books/tuvsud-iso-26262-compliance.pdf?la=en&hash=6D85CEDDF5BF0ADD8641CBBCECD273ECE5080061
[4] https://en.wikipedia.org/wiki/Automotive_Safety_Integrity_Level#cite_note-ISO26262Part3-2
[5] ISO 26262-4:2018 Road vehicles -- Functional safety -- Part 4: Product development at the system level