EDA Support for Functional Safety – How Static and Dynamic Failure Analysis Can Improve Productivity in the Assessment of Functional Safety

by Dan Alexandrescu, Adrian Evans and Maximilien Glorieux—IROC Technologies

Integrated circuits used in high-reliability applications must demonstrate low failure rates and high levels of fault detection coverage. Safety Integrity Level (SIL) metrics indicated by the general IEC 61508 standard and the derived Automotive Safety Integrity Level (ASIL) specified by the ISO 26262 standard specify specific failure (FIT) rates and fault coverage metrics (e.g., SPFM and LFM) that must be met. To demonstrate that an integrated circuit meets these expectations requires a combination of expert design analysis combined with fault injection (FI) simulations. During FI simulations, specific hardware faults (e.g., transients, stuck-at) are injected in specific nodes of the circuits (e.g., flip flops or logic gates).

Designing an effective fault-injection platform is challenging, especially designing a platform that can be reused effectively across designs. We propose an architecture for a complete FI platform, easily integrated into a general-purpose design verification environment (DVE) that is implemented using UVM. The proposed fault simulation methodology is augmented using static analysis techniques based on fault propagation probability assessment and clustering approaches accelerating the fault simulation campaigns. The overall framework aims to: identify safety-threatening device features, provide objective failure metric, and support design improvement efforts.

We present a working example, implemented using Mentor’s Questa® Verification Platform where a 32-bit RISC V CPU has been subjected to an extensive static and dynamic failure analysis process, as a part of a standard-mandated functional safety assessment.

INTRODUCTION

Functional safety is a major enabler of many of the current and upcoming automotive products and has been formalized by the ISO 26262 standard. This body of work recommends methods and techniques to demonstrate that specific targets for failure (FIT) rates and fault coverage metrics (e.g., SPFM and LFM) are successfully met. In addition to expert design analysis, Fault Injection (FI) techniques can support design validation and verification efforts by providing trustworthy and actionable data while spending the least amount of computational resources and engineering effort.

On the hardware and system side, the ISO 26262 standard mentions or recommends fault injection in several locations in Chapter 4 – System Level, tables 3, 5, 8, 10, 13, 15, and 18 as a way to verify the correct implementation of the functional safety requirements and the effectiveness of a safety mechanism’s failure coverage at the system (i.e., vehicle) level. Chapter 3 – Hardware Level, table 11 also presents fault injection as a method to verify the completeness and correctness of the safety mechanisms implementation with respect to the hardware safety requirements. Chapter 3 further motivates the need for fault injection as a way to evaluate the efficiency of safety mechanisms for both checking the results against the SPF and the LF metrics and providing actual data for the computation of the residual FIT. Lastly, the Chapter 10 - Guideline on ISO 26262 presents fault injection (in addition to other methods) as an approach for the verification of the amount of safe faults and especially of the failure mode coverage.

Accordingly, the standard provides relevant guidance for a successful implementation and execution of fault injection campaigns. Firstly, the fault injection shall aim to prove that the safety mechanisms are working in the system (and not only in isolation). This requirement will obviously drive the need for a fault injection campaign performed on a representative
Secondly, the fault injection should exercise the safety mechanisms and extract data about their coverage. Lastly, the fault injection should highlight the behavior of the system in case of failures in order to provide the data required for the failure rate analysis and classification.

On a more practical note, fault injection campaigns are a significant undertaking, requiring engineering effort and computational resources. This process is costly and requires specific expertise, knowledge, and EDA support. This article presents a series of practical techniques that can be implemented in existing design and verification environments (DVE) using Mentor’s Questa® platform tools and aim at improving the confidence in the fault injection campaign results, minimizing the resources (CPU time, engineering effort) spent during the evaluation and maximizing the usefulness of the simulation results. The presented techniques are a part of the feature set of IROC’s comprehensive EDA platform for failure analysis of SoCs (SoCFIT).

**FAULT UNIVERSE BUILD-UP**

Fault simulation techniques are widely researched and used in electronic design practice. One of the most important challenges consists in dealing with the complexity of the system, the application, and the consequences of the fault.

Complex automotive systems have related complex fault universes. The ISO 26262 standard mentions random hardware faults; such as, caused by internal (technological manufacturing process, aging, alpha contamination) and external causes (electronic interference, single events, extreme operating processes). The corresponding fault models can be more or less precise and can be used only at specific design abstraction levels. Thus, a first challenge resides in the elaboration of a complete ISO 26262 fault universe for the considered circuit. While modern EDA could support a very low fault granularity (even at the transistor level) the cost of such an approach would be prohibitive. High-Level Synthesis (HLS) or RTL design representations are more appropriate and also correspond to the design level where most of the engineering effort is spent and where the required tools and environments – such as the DVE – are available. Therefore, the works presented in this article use the RTL description of the device under test (DUT) and the appropriate Questa® simulation tool.

The main challenge we will have to address consists in the relevancy of the HLS/RTL description with regard to the faults applicable to physical design features. In some cases, fault models can be attached to RTL elements, while in other cases, the RTL description doesn’t have a representation of the underlying hardware gates. As an example, a single event upset affecting a hardware flip-flop can be modeled in RTL as a fault attached to a Verilog register. However, a list of “single event transient” faults cannot be built as the RTL files have no indication yet of the logic network implementation.

Obviously, a gate-level netlist (GLN) description would have been easier to use, as the GLN instances correspond directly to design features. However, it is not always possible to perform fault simulation in GLN, as the verification designers may not have added gate-level simulation capabilities to the DVE. If such capabilities exist, it is overwhelmingly probable that the simulation speed is greatly reduced when compared to RTL simulations, leading to long (and costly) fault simulation campaigns.

We have built a tool to extract critical design features from RTL and GLN circuits: hierarchical information, list of signals and variables, list of circuit instances, and so on. The RTL and GLN databases are cross-linked. Based on a series of rules, RTL design elements are linked to actual GLN features whenever possible. The tool relies on VPI capabilities in addition to Questa® features such as the “find … -recursive” command. Any compound elements such as buses, arrays, and complex signal types are itemized to single physical signals whenever possible.

Moreover, each type of fault requires the corresponding fault rate that indicates how often the circuit encounters the fault in a given working environment. While the fault rate will be mostly used during the FMDA efforts, it could be useful as a criterion for failure injection prioritization.
FAULT UNIVERSE OPTIMIZATION

We use a series of fault equivalence techniques to compact the fault universe.

State elements (sequential cells, memory blocks) should exist in both RTL and GLN. The tool is able to build an exhaustive fault list associated to state elements. However, combinatorial gates are not available in RTL. In this case, the tool uses a static analysis approach (based on the calculation of a fault propagation probability logic de-rating, or LDR, factor) that links the cell instances from combinatorial networks to the downstream state-element. According to the state of the circuit (the values of the signals and cell outputs), the propagation of the fault is subject to logic blocking. As an example, an AND gate with a “low” input will block any faults on the other inputs. In the context of this document, logic de-rating only covers the initial clock cycle propagation. The SoCFIT tool is able to compute LDR factors for very complex circuits (millions of flip-flops) in a reasonable amount of time (minutes).

LDR is able to link several circuit elements to a representative. Fault simulation can then proceed on the representative element and data for the equivalded elements can be extrapolated back with the use of LDR factors.

Another technique for reducing the fault universe is clustering. RTL/GLN designs are explored in order to find elements that can be grouped together according to their likeness. The criteria can be structural (buses, hierarchical modules, naming conventions, distance) or functional, such as the probability for a fault in an element to propagate into another (LDR). Simulating only one representative of the cluster and generalizing the results to the full cluster will significantly reduce simulation times.

SIMULATION-TIME OPTIMIZATION

A vast majority of faults injected in the circuit will have no impact on the function of the system (safe faults). In this case, Check-point techniques can be very effective to identify simulation cases where the injected fault disappears quickly. We have developed a compact simulation trace approach based on CRC.

First, we calculate the CRC for each flip-flop and memory in the design, each N clock cycle, during a golden, fault-free simulation (reference CRC trace file).

Then, we use the simulator capabilities to check the circuit status during the fault injection, and we stop the simulation as soon as the CRC matches the golden.

Hardware faults that can occur randomly, such as single event effects, can be also masked by temporal de-rating (TDR). TDR represents two different phenomena related to temporal aspects. The first consists in evaluating the effects of the fault with respect to the occurrence time within the clock cycle or application phase. The second aspect concerns only transient faults, such as SETs and copes, with the relevance of the pulse duration versus the clock period. TDR can be evaluated using low-cost probabilistic methods.

Memory blocks can contain data that is changed frequently. Soft errors affecting data stored in a memory block can only propagate if the affected data is actually read. If a write operation stores another data at the affected location, then the fault is overwritten. Accordingly, using the results of a single, golden simulation, we can compute a memory de-rating (MDR) factor which represents the probability for a fault to arrive during a sensitive opportunity window:

\[ MDR = \frac{(t_0 \cdot t_w) + (t_2 \cdot t_c)}{t_c - t_r} \]

Lastly, fault simulation acceleration techniques are readily available and can be used to simulate several errors per run, or stop the simulation, when the injected error has been classified as failure/latent/discard.
FAULT INJECTION AND FAILURE CLASSIFICATION

We have built a manager tool that is able to select faults from the fault universe, prepare a simulation case including a fault injection feature, and run the case using the Questa® simulation tool. The actual fault injection can be performed through simulator capabilities such as force/release or PLI/VPI procedures. The simulation case will be run and the existing DVE features will highlight the outcome of the fault injection, providing the data required for further failure analysis.

The article “Extending UVM Verification Models for the Analysis of Fault Injection Simulations”, published in the Verification Horizons edition from June 2016, shows how the components of a UVM functional verification environment can easily be extended to record additional information about the types of errors that have occurred. This additional information can be used to classify failing tests based on their system level impact.

In the context of the ISO 26262 standard, the failures need to be classified in the following categories: “safe fault”, “detected multiple-point fault”, “perceived multiple-point fault”, “latent multiple-point fault”, or “residual fault/single-point fault.” It’s important to understand if a fault simulation campaign can fulfill this requirement.

After analyzing Figure B.2 from the ISO 26262 document, we can conclude that failure classification should be able to bin the observed faulty circuit behaviors to the ISO 26262 classes, but it will have to be complemented by design review and, specifically, by the identification of safety mechanisms.

As an example, “safe faults” correspond to the category of faults that are either a) not affecting a safety-related element or b) affecting a safety element but cannot lead to a safety goal violation in combination with an independent failure.

A fault simulation campaign can help with a), as it will provide the percentage of faults that caused no ill effects. The b) case requires fault simulations with multiple fault injections, which will be more costly.

“Single-point faults” have the potential to cause issues, and they are not covered by safety mechanisms. Fault simulation will be able to help with the evaluation of this class of failure, but it will have to be helped by design review.

In any case, the outcomes of the fault simulation campaign are twofold:

- Quantitative results, such as the percentage of failures in the different classes to be used in the FMEDA efforts
- Qualitative results about the behavior of the circuit in the presence of faults

DEVICE UNDER TEST AND FAULT SIMULATION CAMPAIGN RESULTS

We have prepared a DVE consisting of the PULPino RISC-V core, the related simulation environment, as well as runtime software consisting of a bit counter application. The DVE has been enhanced with modules of IROC’s SoCFIT reliability evaluation platform, running on top of Mentor’s Questa® simulation tool. The platform is able to build a complete fault universe; inject faults in the DUT; schedule, execute, and manage the fault simulation campaign; monitor circuit behavior in the presence of faults; and evaluate and finally classify failures. The failures observed during the simulation campaign are natively reported in terms of “classical” silent data corruption (SDC) and detected unrecoverable error (DUE). SDC failures correspond to cases where the workload executes to completion but the results are false. The DUE category is comprised of cases where the simulation didn’t complete successfully: the CPU hangs, loops, or generates exceptions.

In the default configuration, there are no safety mechanisms (in the ISO 26262 sense) available on PULPino, and we consider that the CPU is a safety-related HW element. Accordingly, the injected faults could be mapped to ISO 26262 single-point faults (when a SDC/DUE has been generated) and safe faults (when the simulation completed successfully) categories. If we assume that the CPU is integrated in an automotive system that features a safety mechanism able to detect major CPU failures (such as a watchdog circuit), then the DUE failures can be managed through a reset/reload. Accordingly, DUE
failures can be categorized as multi-point faults. SDC failures cannot be detected by the watchdog, as the CPU is running and able to service the watchdog requests without any issues. Accordingly, SDC failures can be categorized as single-point faults. In addition to failure classification, the fault injection campaign also allows the calculation of the percentage of safe faults (i.e., faults with no impact on the application).

The simulation campaign consisted in more than 22,000 simulation cases. Transient hardware faults of type single event upset (SEU) have been injected fairly in all the elements of the CPU. The results are presented below (95% confidence intervals are indicated inside the parenthesis):

![Table Image]

The table indicates that for given random, fair, fault injections in the circuit, we need to observe at least 100 failures if ±20% bounds for a 95% confidence interval is required. In our example, the SPF category had 404 failures, which is sufficient for ±10% bounds.

Lastly, it may be of interest to understand what CPU elements are critical for a specific or for multiple applications in the case of a CPU-type device that may be used with different workloads in different products. To address this problem, the CPU can be exercised with several applications and the elements’ susceptibility (ability to cause failures) can be presented using the following format:

The workload-specific failure data can be then exploited to evaluate the list of design elements that are critical for all workloads and that can be prime targets for any hardening or protection efforts.
CONCLUSIONS
We have presented a fault injection environment (FIE) – built on Mentor’s Questa® simulation tools and IROC’s reliability evaluation platform – that is able to evaluate the fault susceptibility of complex microelectronic designs. The presented techniques can be implemented in existing DVEs and deliver results usable in the context of standard-driven reliability assessment and management efforts.
VERIFICATION ACADEMY
The Most Comprehensive Resource for Verification Training

32 Video Courses Available Covering
• UVM Debug
• Portable Stimulus Basics
• SystemVerilog OOP
• Formal Verification
• Intelligent Testbench Automation
• Metrics in SoC Verification
• Verification Planning
• Introductory, Basic, and Advanced UVM
• Assertion-Based Verification
• FPGA Verification
• Testbench Acceleration
• PowerAware Verification
• Analog Mixed-Signal Verification

UVM and Coverage Online Methodology Cookbooks
Discussion Forum with more than 8250 topics
Verification Patterns Library

www.verificationacademy.com