Pre-Silicon Safety-Related Functional Verification of Automotive Smart Power ICs Using the Fault Injection Technique

Özlem Karaca

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Universität der Bundeswehr München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation.

Gutachter:

1. Prof. Dr. techn. Linus Maurer
2. Prof. Dr. rer. nat. Georg Pelz

Die Dissertation wurde am 12. 04. 2017 bei der Universität der Bundeswehr München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 05. 09. 2017 angenommen. Die mündliche Prüfung fand am 23. 10. 2017 statt.

Acknowledgement

First and foremost I would like to thank my doctoral advisors Prof. Dr. Linus Maurer and Prof. Dr. Georg Pelz for supervising my research work. Thank you for your support and guidance from an academic as well as industrial perspective.

I'd like to thank the members of the Automotive Design Methodology group at IFX Munich and Bucharest Jerome Kirscher, Matthias Kunze, Dr. Monica Rafaila, Dr. Manuel Harrant, Dr. Andi Buzo, Ciprian Ivu Petru and Dr. Thomas Nirmaier for supporting my work and being great colleagues. I appreciate your accomplishments on which I was able to build on. Moreover, I'd like to thank Dr. Thomas Rickes and my fellow doctoral candidates Dr. Klaus Hoermaier and Sebastian Simon for your contributions to my work.

From the IFX sites in Villach and Padova, I'd like to thank Arnaud Laroche, Andreas Tributsch, Roland Lengfeldner, Stefano Orlandi, Christian Garbossa, Enrico Orietti and Filippo Mele for supporting my work and giving me the opportunity to establish a sound basis for my work in real projects.

Finally, I would like to express my special thanks and appreciation to my husband Lukas, my family and friends for their support during my education.

Abstract

The increasing demand for electrification of automotive systems with electronic control units (ECUs) is driven by automotive trends such as electromobility and autonomous driving. However, the functional complexity of modern analog/-mixed-signal integrated circuits (ICs) causes an increasing risk from random hardware failures in the electronics. Even more for safety-related applications, the presence of a hardware failure in the IC may have severe consequences for humans and environment. Safety-related ICs such as Smart Power ICs must ensure a safe operating state even in the presence of a random hardware failure. For this purpose, the ICs implement diagnostic capability by safety mechanisms which prevent random hardware failures from causing hazard. Recently, this topic has come to the focus of interest in semiconductor companies due to the advent of the functional safety standard for road vehicles, ISO 26262. The standard provides a framework of requirements which affects the whole development process of safety-related ICs due to which semiconductor companies face new challenges in terms of compliance.

The pre-silicon functional verification is generally a crucial stage within the IC development. It allows to detect functional misalignments between the circuit's required and actual behaviour before manufacturing. In this context, the standard explicitly requires a simulation-based method utilizing the fault injection technique. The purpose of this method is to evaluate the circuit in the presence of hardware failures in terms of diagnostic capability and compliance with functional requirements.

Fault injection and simulation of analog/mixed-signal circuits has been a challenging task since the early 90s. This is mainly due to the lack of a generally accepted definition of fault coverage and corresponding fault models. Additionally, thorough fault simulation with the transistor-level netlist becomes infeasible for the top-level circuit. Moreover, commercially available computer-aided design tools do not yet offer a feature for automatizing the fault injection technique for analog/mixed-signal circuits. Thus, this task requires so far labour-intensive manual effort. In order to utilize the fault injection technique for pre-silicon safety-related functional verification of analog/mixed-signal circuits, the above mentioned challenges must be addressed by an adequate methodology.

The work presented in this thesis facilitates integration and automation of the fault injection technique in the electronic design automation tool Cadence® Virtuoso®. A fault model library is developed comprising diverse fault models which facilitate fault injection at different levels of abstraction of the circuit design. As an initial approach to safety-related verification, the

informal safety analysis method Failure Mode, Effects and Diagnosis Analysis (FMEDA) is utilized to structure the safety-related verification plan. Subsequently, a hierarchical fault injection approach is presented for efficient top-level verification by skipping redundant fault simulation runs. This approach is based on fast component-level fault simulations and the determination of functional equivalent faults. Functional fault equivalence is determined by a clustering algorithm which processes simulated component-level circuit responses in order to find similarities among the waveforms. Eventually, functional equivalent faults must not be simulated repeatedly for top-level verification. Finally, soft faults by means of parametric and soft-structural fault models are considered for fault injection with variable parametrization. Their effects in the circuit are evaluated by means of global sensitivity analysis. Sensitivity indices are calculated which quantify the contribution of each soft fault to the variability of the circuit response. The sensitivity indices are used to rank faults and identify non-influential soft faults. A statistical significance test is exercised in order to eliminate non-significant faults from the soft fault list. Moreover, the cumulative contribution of each soft fault to the variability of the circuit response is used to further eliminate soft faults by keeping only those in the soft fault list which account for most of the variability. Although this approach reduces the soft fault coverage, most of the output variability due to soft faults in maintained for the verification.

Experimental results are presented for safety-related functional verification of an automotive high-voltage Lithium-ion cell balancing and monitoring module and a general purpose gate driver circuit which is a safety-related module of an automotive System-on-Chip (SoC). Finally, the work is concluded and an outlook is given.

Abstrakt

Der steigende Bedarf an elektrifizierten Systemen im Automobil durch elektronische Kontrolleinheiten, engl. Electronic Control Units (ECUs), wird angetrieben durch Trends wie Elektromobilität und autonomes Fahren. Gleichzeitig steigt die funktionale Komplexität moderner integrierter Analog/Mixed-Signal-Schaltungen (AMS ICs), wodurch sich auch das Risiko für das Auftreten eines zufälligen Hardwarefehlers, engl. Random Hardware Failure, in der Elektronik erhöht. Dies gilt insbesondere für sicherheitsrelevante Applikationen, in denen ein Hardwarefehler schwerwiegende Konsequenzen für Mensch und Umwelt haben kann. Aus diesem Grund müssen sicherheitsrelevante ICs, wie Smart Power ICs, in der Gegenwart von zufälligen Hardwarefehlern einen sicheren Betriebszustand gewährleisten. Hierzu wird die Funktionalität des ICs durch diagnostische Funktionen und Sicherheitsmechanismen erweitert.

Halbleiterunternehmen haben dieses Thema unlängst mit der Einführung der ISO 26262 erschlossen. Der Standard adressiert die funktionale Sicherheit von Straßenfahrzeugen und widmet sich dabei insbesondere den elektrischen, elektronischen und programmierbar elektronischen Komponenten im System. Im Allgemeinen bietet der Standard ein Rahmenwerk an Anforderungen, das den gesamten Entwicklungsprozess sicherheitsrelevanter ICs betrifft. Halbleiterunternehmen stehen in diesem Kontext neuen Herausforderungen gegenüber.

Die funktionale Verifikation mithilfe computergestützter Entwurfsmethoden, engl. computer aided design (CAD), und Simulation, d.h. Pre-Silicon Verifikation, stellt eine entscheidende Phase innerhalb der IC-Entwicklung noch vor der Produktion von Prototypen dar. Dabei wird der Schaltungsentwurf auf Abweichungen von seinem vorgesehenen funktionalen Verhalten überprüft. In diesem Kontext fordert die ISO 26262 explizit die simulationsbasierte Injektion von Hardwarefehlern in den Schaltungsentwurf. Dies gilt der Evaluierung des Schaltungsentwurfs hinsichtlich seiner diagnostischen Kapabilität sowie Erfüllung von sicherheitsbezogenen funktionalen Anforderungen in der Gegenwart von Hardwarefehlern.

Die simulationsbasierte Fehlerinjektion für AMS ICs gilt bereits seit den frühen 90ern als eine herausfordernde Aufgabe. Dies ist hauptsächlich auf das Fehlen eines generell akzeptierten Maßes für die Fehlerabdeckung sowie die damit verbundenen Fehlermodelle zurückzuführen. Außerdem ist eine eingehende Fehlersimulation auf Transistor-Ebene (Netzliste) für große Schaltungsentwürfe aufgrund der Simulationsdauer nicht praktikabel. Zudem bietet derzeit kein kommerzielles CAD-Tool die automatisierte Fehlerinjektion für AMS ICs. Dadurch ist die Fehlerinjektion mit erheblichem manuellem Arbeitsaufwand verbunden. Es wird eine adequate Methodik benötigt, die

die genannten Herausforderungen berücksichtigt und entsprechende Lösungsansätze liefert, um die Pre-Silicon Verifikation im Kontext der funktionalen Sicherheit zu realisieren. Die vorliegende Arbeit widmet sich diesem Thema.

Zunächst wird ein Ansatz für die Integration und Automatisierung der Fehlerinjektion in dem kommerziellen CAD-Tool Cadence® Virtuoso® vorgeschlagen und implementiert. Hierzu wird eine Fehlermodellbibliothek entwickelt, mit der die Fehlerinjektion auf unterschiedlichen Abstraktionsebenen des Schaltungsentwurfs ermöglicht wird. Für die Strukturierung und Planung der Pre-Silicon Verifikation im Kontext der funktionalen Sicherheit wird initial eine Failure Modes, Effects and Diagnostic Analysis (FMEDA) verwendet. Danach wird ein Ansatz für hierarchische Fehlerinjektion vorgeschlagen, mit dem Zweck redundante Fehlersimulationen zu vermeiden und somit die Effizienz der Verifikation bei großen Schaltungsentwürfen zu erhöhen. Schließlich werden parametrische Soft-Fehler bei der Fehlerinjektion betrachtet. Die Effekte der Soft-Fehler werden auf der Basis der Fehlerinjektion mit variabler Parametrierung durch eine qualitative und quantitative globale Sensibilitätsanalyse untersucht. Zu diesem Zweck werden aus den Simulationsergebnissen Sensibilitätsindizes berechnet, die jeweils den Beitrag eines Soft-Fehlers zur Varianz des Schaltungsausgangssignals quantifizieren. Die Sensibilitätsindizes werden verwendet um Soft-Fehler nach Wichtigkeit einzuorden sowie vernachlässigbare Soft-Fehler zu identifizieren.

Die experiementellen Ergebnisse werden anhand von zwei Modulen in sicherheitsrelevanten Applikationen aus dem Automobilbereich generiert. Dabei handelt es sich um ein Modul zum Ausgleich und zur Überwachung von Lithium-Ionen Zellen sowie einem Gate-Treibermodul, welches eine sicherheitsrelevante Funktion in einem System-on-Chip (SOC) durchführt. Abschließend wird die Arbeit zusammengefasst und ein Ausblick auf die zukünftige Entwicklung gegeben.

Contents

List of Abbreviations

ASIL	Automotive Safety Integrity Level
CAD	Computer-Aided Design
DC	Diagnostic Coverage
DPF	Dual-Point Fault
E/E/PE	Electrical/Electronic/Programmable Electronic
ECU	Electronic Control Unit
FET	Field Effect Transistor
FMEA	Failure Mode and Effect Analysis
FMEDA	Failure Mode, Effect and Diagnostics Analysis
FPGA	Field Programmable Gate Array
FSR	Functional Safety Requirements
HDL	Hardware Description Language
HSR	Hardware Safety Requirements
MEMS	Micro-Electromechanical System
MPF,D	Multiple-Point Fault, Detected
MPF,L	Multiple-Point Fault, Latent
MPF,P	Multiple-Point Fault, Perceived
PDF	Probability Density Function
RF	Residual Fault
RHF	Random Hardware Failure
RTL	Register Transfer Level
SF	Safe Fault
SG	Safety Goal
SIL	Safety Integrity Level
SM	Safety Mechanism
SoC	System-on-Chip
SPF	Single Point Fault
SPICE	Simulation Program with Integrated Circuit Emphasis
TSR	Technical Safety Requirements
UML	Unified Modelling Language

List of Figures

List of Tables

Chapter 1

Introduction

Safety-related functions in road vehicles are more and more implemented by electronic control units (ECUs). Modern automotive ECUs are highly-integrated, multi-functional integrated circuits (ICs), processing digital as well as analog (mixed) signals. With increasing automotive electrification and functional complexity, the risk from random hardware failures increases. The evaluation of the circuit's functional performance in the presence of a random hardware failures thereby plays a crucial role. Recently, this topic came to the spotlight in the semiconductor industry due to the functional safety standard for road vehicles, ISO 26262.

This thesis addresses the topic safety-related pre-silicon functional verification of automotive ECUs with emphasis on the analog/mixed-signal domain. This chapter provides a background on fundamental concepts related to this topic and the motivation for this work. Subsequently, the challenges regarding the addressed topic are explained and the scope of this thesis is defined. Finally, an outline for the thesis is provided.

1.1 Background

In this section, the background on automotive ECUs, failures in semiconductor devices, pre-silicon functional verification as well as concepts and definitions in the ISO 26262 is provided.

1.1.1 Automotive electronic control units

Automotive ECUs comprise smart power devices [10]. Smart power devices are semiconductor devices which integrate power semiconductors [11] and peripheral circuitry to interface the digital control logic, loads and sensors

into a single unit. Additionally, they integrate self-diagnosis and peripheral protection circuits against for example current, voltage, temperature and electrostatic discharge surges [10]. Due to its advanced capability for large-scale integration and low energy consumption, the metal-oxide semiconductor (MOS) technology is predominantly used in smart power devices [10]. Fig. 1.1 shows a simplified example of an automotive ECU. Smart power devices can be distinguished with respect to their functionality in the application into [10, 1]:

- **Driver** The driver functionality is implemented by analog/mixed-signal circuits to drive several types of power actuators, for example LED modules, various types of electric motors or squibs.
- **Supply** The supply functionality maintains the voltage and current supply to other devices. Depending on the application, examples of supply circuits are bias circuits, linear voltage regulators and DC/DC converters.
- **Transceiver** this device facilitates the communication between different ECUs via various types of bus systems, for example CAN, LIN and FlexRay. Transceiver circuits convert the transmitted (received/sent) information for/from the ECU's embedded microcontroller.
- **Power management** This device is implemented by analog/mixed-signal circuits. Examples are battery management ICs, dedicated to monitor and balance Lithium-ion battery cells in electric vehicles and alternator control ICs, dedicated to stabilizing the vehicle's supply voltage.
- **Sensor interface** This device facilitates information transfer from the sensors to the ECU's embedded microcontroller.

1.1.2 Pre-silicon functional verification

The purpose of functional verification of an IC design is "*to ensure that the design meets the functional requirements as defined in the functional specification*" [2]. With increasing complexity and functional diversification of ICs ("*More than Moore*"), functional verification is getting even more challenging [12, 13]. This becomes evident when up to 70% of project resources are spent on functional verification during IC development [14]. Functional verification is generally distinguished into pre-silicon and post-silicon verification [15]. For post-silicon verification, a prototype of the actual hardware

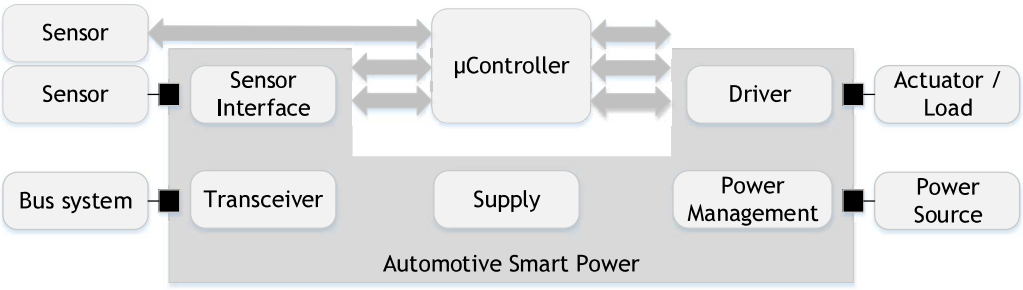


Figure 1.1: Example of an ECU with smart power devices and peripheral devices [1].

is verified in a test environment. Pre-silicon verification on the other hand is undertaken before any silicon prototype is available. For this purpose, a model of the IC is verified at various abstraction levels using formal-based [16] and/or simulation-based methods [2]. Simulation-based verification is predominantly used in the industry due to its advantages in scalability compared to formal verification [17, 18].

Model abstraction levels

For pre-silicon verification, the IC is typically modelled at different abstraction levels: architectural, behavioural, register-transfer level (RTL), gate level and circuit level (also *transistor level* or *primitive level*) [19]. The top-down design methodology starts with an abstract description of the circuit design and iteratively adds detail to the model [2], see fig. 1.2. The final level of lowest abstraction is the physical design (layout). Additionally, the gate-level/primitive-level netlist can be distinguished into pre-layout and post-layout netlist. Latter incorporates layout-specific parasitic resistive and capacitive elements.

Due to digital synthesis [20] (automatic translation of an abstract model, typically RTL, into the logic-gate level design), the digital design implementation methodology diverges substantially from that of the analog design implementation [19]. Contrary, analog/mixed-signal design implementation at transistor level is done manually by means of electronic computer-aided design (CAD) software tools [21].

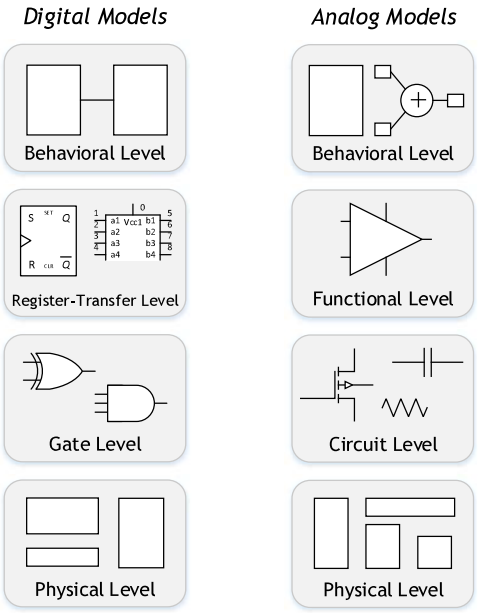


Figure 1.2: Top-down abstraction models [2].

Mixed-signal verification methodology

Simulation-based verification methodologies diverge strongly for the digital and analog parts in mixed-signal circuits. For digital circuits, typically the RTL model is used to represent the design under verification (DUV) in a simulation test bench. The verification thoroughness is expressed in terms of coverage metrics: code coverage [22], assertion coverage [23] and functional coverage [24]. They quantify how well the circuit's functionalities are exercised for a given set of input signal pattern (stimuli). Stimuli are developed in a directed (user-defined) fashion or in a randomized fashion by including constraints to the multi-dimensional input space ("*Constrained-Random Stimuli*)" [25].

Analog verification methodology

For analog/mixed-signal circuits, the constrained-random and coverage-driven verification methodology is a topic of recent research and still under development [26]. This is mainly due to two reasons: the ambiguity in defining analog coverage metrics [27, 28, 29] and the lack of a correct-by-construction abstract model of the analog circuit, analogous to the RTL model in the dig-

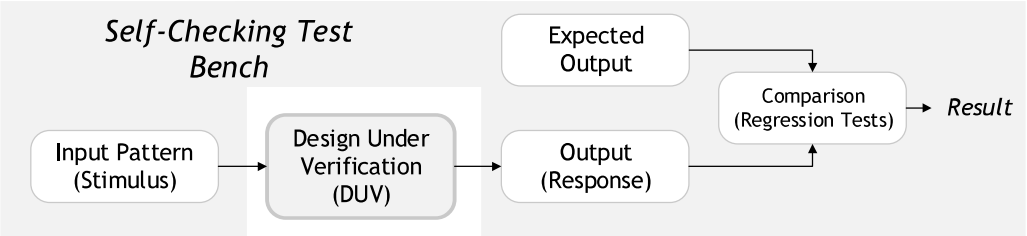


Figure 1.3: Self-checking test bench [2].

ital domain [30]. Additionally,transient simulation time is a major concern for complex circuits at the transistor level.

The computational burden of circuit-level simulation is addressed by analog or real-number behavioural modelling using hardware description languages (HDL) [31], like for example VHDL/-AMS [32, 33] and Verilog/-AMS [34, 35]. However, the models are not correct-by-construction, i.e. subsequent equivalence checks between the abstract model and the transistor-level model are necessary [18]. The accuracy of the behavioural model depends on its level of detail. For example, a highly detailed behavioural model can take even more simulation time as its circuit-level equivalent [36].

For complex circuits this is addressed by hierarchical verification [30] and mixed-mode/mixed-level simulations [37, 38]. To attain consistency throughout the design hierarchy, the verification typically starts by component-level verification with circuit-level DUV verified against the component-level design specification. Subsequently, the verified components are used in the top-level (or intermittent design level) test bench. For top-level verification, mixed-mode (combination of circuit-level and RTL) and mixed-level (combination of circuit-level and behavioural model) simulations are considered [37, 38]. For example, critical circuits which are in focus of verification are kept at circuit level and non-critical circuits, like bias generators and digital parts are modelled in HDL, respectively [30].

The test bench for the analog/mixed-signal DUV is typically constructed as code or as a schematic in a CAD software tool, see fig. 1.3 [2]. The test bench comprises stimuli defining input pattern for which the DUV is exercised for different operating modes. In a self-checking test bench the DUV output is compared with the expected output by regression testing [2].

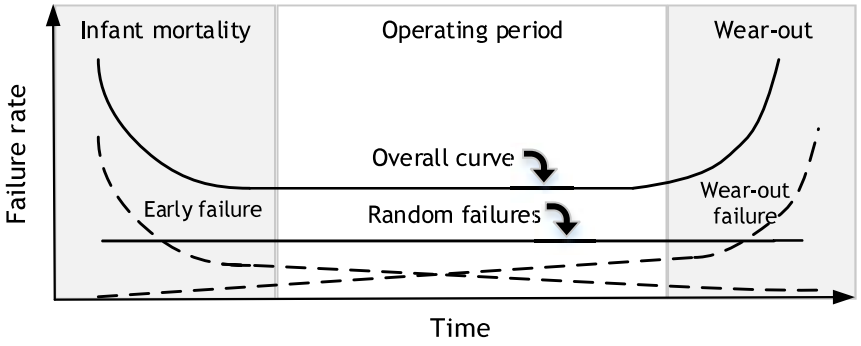


Figure 1.4: Bathtub curve function for semiconductor failure rate calculation [3].

Analog verification flow

The analog verification flow starts with the design documentation from which design specification is extracted and design risks are identified. A verification plan is developed to cover the design requirements with the purpose of risk mitigation. The verification plan is implemented by the modelling plan and simulation plan for the component-level and top-level models. At the top level, a self-checking test bench is developed based on the verification plan, which can be configured regarding the component models to use for each test case (circuit level or behavioural/RTL level). The final design and layout is obtained in an iterative process when all regression tests pass. This means that the design implementation is conform with the design specification.

1.1.3 Failures in semiconductor devices

For semiconductor devices, three failure regions are distinguished: early failure, random failure and wear-out failure [39]. The failure rate calculation is based on the bathtub curve function, see fig. 1.4 [40]. In this model, the early failure rate decreases steadily over time, the random failure rate is a constant value and the wear-out failure rate decreases steadily. At each point in time, the bathtub-shaped curve is the sum of the three failure rates.

Random failures

Between the early failure period and wear-out, the device may be subject to random failures during field operation. The failure rate λ of random failures is a constant and much lower than the early or wear-out failure rates. Random failures and failure rates are determined by operating life tests in dynamic electric operation. The failure rate is generally calculated by [3]

$$\lambda = \frac{\text{Sum of failures}}{\sum(\text{Quantity} \times \text{Time to failures})}. \tag{1.1}$$

The failure in time (FIT) rate is another common notation, with one FIT being equal to $\lambda = 10^9 \frac{1}{\text{h}}$ failures per device hours. Besides operating life tests, reliability handbooks are used to obtain the information on failure rates and generic random failures for different devices, that is failure modes [39]. Additionally, frequent causes for transient random failures are α-particles and cosmic radiation [41], electromagnetic interference (EMI) [42] and crosstalk [43], as well as electrostatic discharge [10].

Safety analysis

Safety analysis is performed with the purpose to evaluate and mitigate the risk of hazardous events due to random failures. The Failure Mode and Effect Analysis (FMEA) [44] and Fault Tree Analysis (FTA) [45] are commonly used safety analysis methods in the automotive and electronics industry. Both, FMEA and FTA consider the possible failures which may occur in the system and their effects. The FMEA is based on an inductive approach, inferring consequences (effects) from causes (failure). The FTA on the other hand, deductively infers from consequences to possible causes.

The FMEA was developed [44] and standardized [46] by the U.S. military to study the problems arising from malfunctioning military systems. However the tool was further developed by aerospace and automotive industries to fit in to the respective field. The Failure Modes, Effects and Diagnostic Analysis (FMEDA) is an extension of FMEA for safety-related systems and additionally includes information on quantitative failure rates, failure mode distributions and diagnostic capability of the respective failure detection and prevention measures in a system.

1.1.4 Fault injection technique

From a general point of view, the fault injection technique [47] is used to evaluate the performance of a system in the presence of a fault (e.g. in terms

Fault injection into	Circuit representation	Fault coverage	Effort
Hardware	+	-	-
Emulation	+	0	-
Simulation	0	+	+
Formal model	-	0	-

Table 1.1: Qualitative comparison of advantages (+) and disadvantages (-) of different methods to conduct the fault injection.

of diagnostic capability) [48] and cause-consequence relationships between faults and their effects [49]. Fault injection technique is reportedly conducted hardware-based [50, 51], emulation-based [52, 53, 53], simulation-based [54, 55, 56] and formal-model-based [57, 58], see tab. 1.1. In the following, they are evaluated respective to their capabilities in circuit representation, fault coverage and effort for fault modelling and injection. Fault coverage refers to the capability to address any desired device or node in the circuit for fault injection.

While by hardware-based fault injection obviously a very high circuit representation is obtained, a limited number of nodes can be addressed for fault injection, thus a low fault coverage. Additionally, high effort is required to conduct the fault injection campaigns (e.g. Design-for-Test [59]). Only externally caused faults at primary input and output ports can be injected as well as at test ports, designed for this purpose.

In the emulation-based fault injection, the circuit RTL code including the fault models are first synthesized on a Field Programmable Gate Array (FPGA). With this technique, the performance of the circuit can be evaluated in real-time [60]. However, it is associated with high effort to conduct a fault injection campaign and is limited to digital circuits only. Additionally, the correlation between gate-level and RTL fault models is limited to stuck-at faults but not for example for bridging faults [61].

Simulation-based fault injection incorporates a circuit netlist as a model and a Simulation Program with Integrated Circuit Emphasis (SPICE) simulator [62]. This technique offers a variety of abstraction capabilities of the circuit for the benefit of lower simulation time. The fault coverage is very high as any node and device in the netlist can be addressed for fault injection. Compared to the other approaches, low effort for fault injection is required. However, the main disadvantage of simulation-based fault injection is that it is computation intensive.

For the formal-model-based fault injection, first a model of the circuit and

fault models are developed which are described in a formal language. This is generally associated with modelling effort. Moreover, large-scale complex circuits and analog circuits boost up the formal model and make it intractable and impractical for use for fault injection [58].

1.1.5 Concepts and definitions of the ISO 26262

The compliance of a product with the functional safety standard for road vehicles, ISO 26262, is determined for items of the vehicle. Items are defined to implement safety-related functions at the vehicle level and can be regarded as isolated from the rest of the vehicle. The standard [4] provides further concepts and definitions which are described in the remainder.

Hardware design

The item is divided into different element types based on the level of detail: (sub-)systems, components, hardware parts and software units. Moreover, it is generally comprised of at least one system which is composed of at least one sensor, one controller and one actuator. Each of them is called a component. Components are composed of at least one hardware part (HW-part) or software unit (SW-Unit). In addition, elements comprised of lower-level HW-parts from electrical/electronic technology are also called components.

Fault, error, failure relationship

Faults are physically present in the hardware and cause an error which may propagate to a failure at the item level. A failure at the level of a hardware element is a hardware element fault at the item level, see fig. 1.5. An error is the deviation of a function implemented by a hardware element from its nominal function. A failure is the malfunctioning behaviour of a hardware element and a failure mode defines the manner in which a hardware element fails.

Based on random causes or systematic causes, failures in the hardware are distinguished into random hardware failures (RHFs) and systematic hardware failures, see fig. 1.6. Latter are due to deterministic issues in the hardware and can only be eliminated by re-design or change of the respective issues. RHFs on the other hand are due to physical processes including wear-out, environmental stress and physical degradation. Moreover, the standard addresses independent random hardware failures. That is, the probability of occurrence of two independent failures P_{AB} is the product of each single

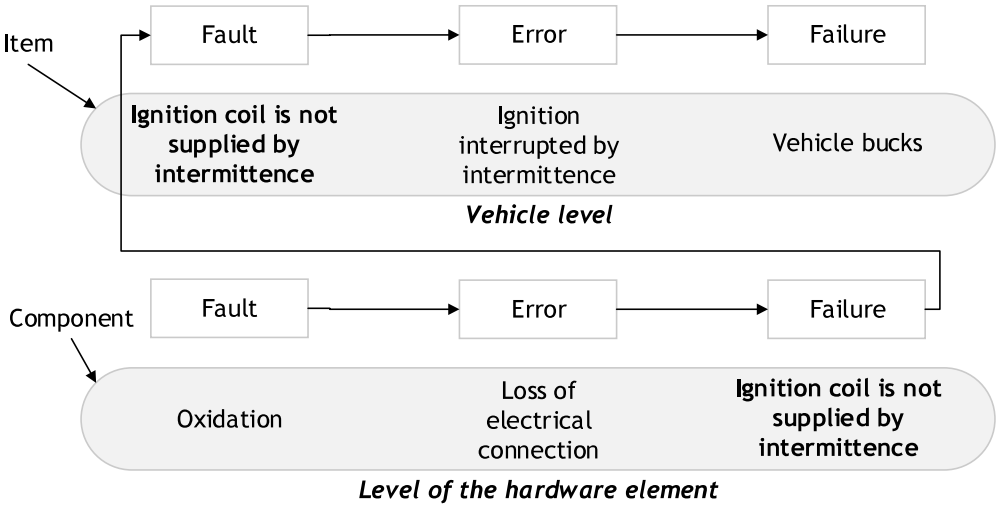


Figure 1.5: Faults, errors and failures in the hardware design [4].

failure probability P_A and P_B. Random hardware failures may occur unpredictably during application with a non-zero failure rate and are assumed to follow the exponential distribution

$$F(x) = \int_{-\infty}^{x} f_\lambda(t)\mathrm{d}t = \begin{cases} 1 - \mathrm{e}^{-\lambda x}, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases}. \tag{1.2}$$

Types of safety requirements

In the concept phase of the safety life-cycle, possible hazardous events due to a RHF at the vehicle level are identified by a Hazard and Operability Analysis (HAZOP) and respective safety goals (SGs) are formulated. Thereby, hazardous events are assessed based on their severity, probability of exposure and controllability. Finally, an Automotive Safety Integrity Level (ASIL) for the avoidance of the hazardous event is derived which is based on this assessment and assigned to the respective SG. An example of a SG formulation is "*Avoid deep discharge of any battery cell, ASIL D*". In this example, the violation of the SG may cause the corresponding hazardous event "*deep discharge of any battery cell*" to occur. This occurrence must be prevented by utilizing prevention measures required to achieve ASIL D. Thus, SGs are top-level requirements which are assigned to the item.

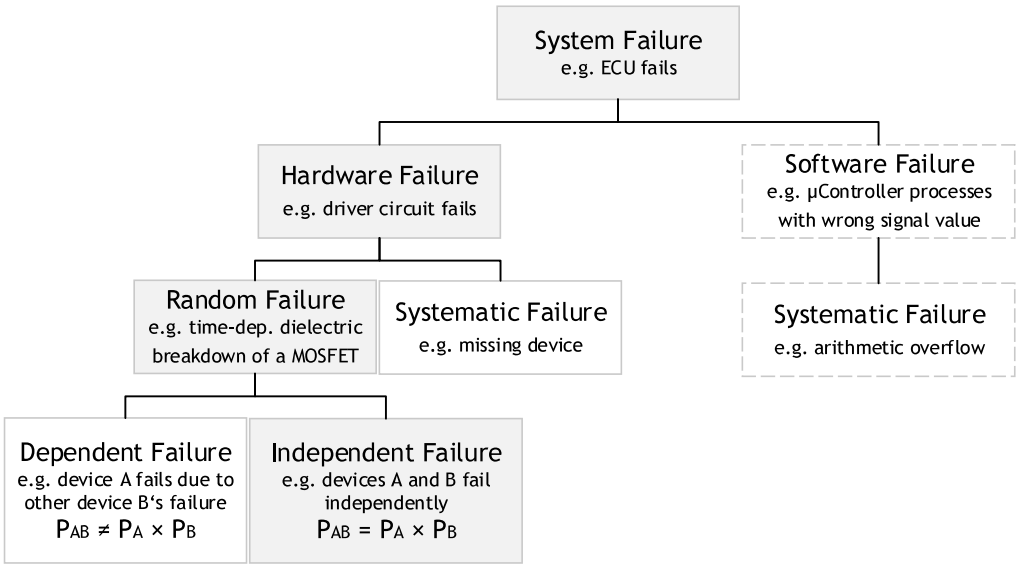


Figure 1.6: Classification of system failures.

At a lower level of the hardware design, functional safety requirements (FSRs) are assigned to the elements in the hardware architecture. Compliance with FSRs is necessary to achieve the SGs. FSRs are independent from the hardware detailed design and implementation and are thus formulated at an early stage of the safety life-cycle.

At the level of the hardware detailed design, technical safety requirements (TSRs) are derived to implement the FSRs. TSRs comprise requirements for hardware as well as software. The hardware-related requirements of the TSRs are isolated from the software-related requirements in the hardware safety requirements (HSR). HSR are formulated with respect to the hardware implementation.

Failure mode classification

Failure modes are classified based on their potential to violate safety goals at the item level and their detectability by safety mechanisms. The classes are:

- **Single-point fault (SPF)** Causes a safety goal violation if no safety mechanism is implemented in the item.

- **Residual fault (RF)** Causes a safety goal violation in the presence of a safety mechanism which does not detect or control it.
- **Multiple-point fault, latent (MPF,L)** Singly not detected or perceived but violates a safety goal in combination with another fault.
- **Multiple-point fault, detected (MPF,D)** Singly detected but violates a safety goal in combination with another fault.
- **Multiple-point fault, perceived (MPF,P)** Singly perceived but violates a safety goal in combination with another fault.
- **Safe fault (SF)** Cannot cause singly or in combination with another fault a safety goal violation; or does not occur in a safety-related hardware element.

The total failure rate λ for the hardware element is the sum of all faults

$$\lambda = \lambda_{SPF} + \lambda_{RF} + \lambda_{MPF} + \lambda_{SF}. \tag{1.3}$$

Safety mechanisms and diagnostic coverage

Safety mechanisms are active or passive components in the item which detect and prevent RHFs from violating a safety goal. The diagnostic coverage (DC) quantifies the proportion of the hardware element failure rate of RFs and latent MPFs faults in percentage which are detected or controlled by the safety mechanisms, see equations 1.4 and 1.5, respectively.

$$K_{\mathrm{DC,RF}} = (1 - \frac{\lambda_{\mathrm{RF}}}{\lambda}) \cdot 100 \tag{1.4}$$

$$K_{\mathrm{DC,MPF,L}} = (1 - \frac{\lambda_{\mathrm{MPF,L}}}{\lambda}) \cdot 100 \tag{1.5}$$

For the purpose of subsequent hardware architectural metrics calculation, the standard proposes two approaches for DC quantification. One approach is to use a conservative DC value proposed by the standard itself with respect to the type of safety mechanism implemented in the design. Thereby, the DC estimates $K_{\mathrm{DC,RF}}$ and $K_{\mathrm{DC,MPF,L}}$ are used to calculate the failure rates λ_{RF} and $\lambda_{\mathrm{MPF,L}}$ which are required for hardware architectural metrics calculation. However, generally the conservative DC values are considered too pessimistic and may degrade the hardware design, unnecessarily. To avoid this, a second approach is proposed for DC quantification which is based on the failure mode classification of each individual failure mode of the hardware element in order to obtain the failure rates λ_{RF} and $\lambda_{\mathrm{MPF,L}}$ for diagnostic coverage calculation.

ASIL	Random hardware failure rate target values
D	$< 10^{-8}\mathrm{h}^{-1}$
C	$< 10^{-7}\mathrm{h}^{-1}$
B	$< 10^{-7}\mathrm{h}^{-1}$

Table 1.2: Possible target random hardware failure rates [4].

Hardware architectural metrics

Hardware architectural metrics quantify the overall effectiveness of the item against random hardware failures. It accounts for the diagnostic coverage of safety mechanisms as well as for the hardware designs inherent reliability. The hardware architectural metrics respective to single and multi-point faults are calculated using the equations

$$H_{\mathrm{SPFM}} = 1 - \frac{\sum_{\mathrm{SR,HW}}(\lambda_{\mathrm{SPF}} + \lambda_{\mathrm{RF}})}{\sum_{\mathrm{SR,HW}} \lambda} = \frac{\sum_{\mathrm{SR,HW}}(\lambda_{\mathrm{MPF}} + \lambda_{\mathrm{S}})}{\sum_{\mathrm{SR,HW}} \lambda}, \tag{1.6}$$

$$H_{\mathrm{LFM}} = 1 - \frac{\sum_{\mathrm{SR,HW}}(\lambda_{\mathrm{MPF,L}})}{\sum_{\mathrm{SR,HW}}(\lambda - \lambda_{\mathrm{SPF}} - \lambda_{\mathrm{RF}})} = \frac{\sum_{\mathrm{SR,HW}}(\lambda_{\mathrm{MPF,P}} + \lambda_{\mathrm{MPF,D}} + \lambda_{\mathrm{S}})}{\sum_{\mathrm{SR,HW}}(\lambda - \lambda_{\mathrm{SPF}} - \lambda_{\mathrm{RF}})}, \tag{1.7}$$

where sum of all safety-related random hardware (SR,HW) failure rates are considered. A high hardware architectural metric indicates high effectiveness of the item against random hardware failures.

ASIL achievement

The achievement of the required ASIL is assessed for the item based on the

1. sum of all safety-related random hardware failure rates $\sum_{\mathrm{SR,HW}} \lambda$ and
2. hardware architectural metrics H_{SPFM} and H_{LFM}.

Possible target values for the failure rate and hardware architectural metrics are proposed by the standard and listed in tables 1.2 and 1.3, respectively.

Fault modelling and fault injection

In terms of fault modelling, the standard addresses only independent random hardware failures, see fig. 1.6 [4]. That is common-cause and cascaded failures are excluded from design verification and safety analysis. Moreover, the generic fault models to be covered are determined by

Hardware architectural metrics	ASIL B	ASIL C	ASIL D
Single-point fault metric	$\geq 90\%$	$\geq 97\%$	$\geq 99\%$
Latent-fault metric	$\geq 60\%$	$\geq 80\%$	$\geq 90\%$

Table 1.3: Possible target hardware architectural metrics [4].

- device failure modes,
- layout-dependent shorts and opens (also "*direct-current* (d.c.)" fault model), and
- soft transient errors.

In the standard, the simulation-based verification using the fault injection technique is explicitly required for the design verification (hardware or system) for completeness and compliance with safety requirements.

System-level and hardware-level design verification The simulation-based verification is required in the product development phase (part 4) for system design verification (part 4, 7.4.8) and hardware design verification (part 5, clause 7.4.4). At system level, it is an obligatory (*shall ++*) requirement to achieve ASIL D or C. The target is to verify the system design for completeness with the technical safety concept and for compliance with technical safety requirements. At the hardware level, it is an optional (*should +*) requirement to achieve ASIL D or C. The target is to verify the hardware detailed design including safety mechanisms for compliance and completeness with hardware safety requirements.

Non-conservative determination of diagnostic coverage In part 5, the standard proposes the generic fault models to be considered for the non-conservative determination of diagnostic coverage. The proposed fault models are distinguished depending on the type of hardware element type and the target diagnostic coverage which must be achieved. With respect to the target diagnostic coverage, different fault models must be considered.

Complement to fault injection hardware tests Generally, the fault injection technique is proposed as a hardware test to verify the completeness and correct implementation of safety mechanisms with respect to the hardware safety requirements (part 5, clause 10.4.5). However, as fault coverage of hardware-based fault injection is very low, this is only feasible for

a limited number of faults which can be introduced in a hardware test. To address this, the fault injection is alternatively proposed to be applied on a model of the hardware, e.g. gate-level or transistor-level netlist for which the hardware response can be simulated.

1.2 Motivation

Simulation-based functional verification of analog/mixed-signal designs is an important task within IC development. However, the state-of-the-art does not elaborately incorporate the safety-related functional verification in the presence of possible random hardware failures in the circuit. This gives rise to utilize the fault modelling and fault injection technique in this context. Moreover, compliance with ISO 26262 explicitly requires the fault injection technique for system and hardware design verification if informal safety analysis methods (e.g. FMEA, FTA) are not considered sufficient, also see Annex A A.3.8.2 Verification using fault injection simulation [4]. Additionally, the diagnostic coverage of safety mechanisms is an important factor in hardware architectural metrics calculation and consequently determines the compliance with the target ASIL. Therefore, it must be quantified accurately. For a generic set of safety mechanisms, the standard proposes diagnostic coverage values which are conservative estimates. This means for example that a safety mechanism in an actual design may be much more effective, but by using the conservative value it is degraded for no rational reason. In this context, fault simulation can be utilized to provide a rationale for evidently increasing the diagnostic coverage.

1.3 Challenges

The implementation of the pre-silicon safety-related functional verification into the IC development process is associated with a number of challenges in terms of feasibility. The main challenges are explained in this section.

Contrary to the digital domain, for analog fault injection there is generally no well-established methodology. This is mainly due to the difficulty regarding the definition of analog fault models/coverage. Consequently, it is difficult to quantify simulation fault coverage in order to express how well the circuit is exercised by simulation of all possible analog faults.

The analog fault injection technique is not a state-of-the-art automated feature of commercially available CAD tools. However, in order to achieve a high verification confidence, extensive fault injection is necessary which is

manually not feasible.

Although the transistor-level/gate-level circuit is suited to achieve high verification confidence, its simulation is computationally demanding if the circuit is complex (e.g. at system level). In order to tackle the simulation time, an adequate fault simulation methodology is required.

1.4 Scope of this thesis

The scope of this thesis is to investigate the fault injection technique for safety-related functional verification of automotive Smart Power ICs and therein particularly analog/mixed-signal devices. Based on the state of the art in the analog verification methodology and the fault injection technique, concepts and definitions from the ISO 26262 [4] are used to develop a methodology.

The objectives of this thesis are to achieve high confidence of verification and high quality of safety analysis. Therefore, fault injection and simulation at the gate-level and transistor-level is addressed. However, due to long simulation times and unacceptable manual effort, this is particularly challenging for complex circuit designs and top verification. The work presented in this thesis addresses to tackle these challenges.

1.5 Outline

The thesis is divided into seven main chapters.

Chapter 2: Related work

Based on a literature research, the related work in the fields of computer-aided safety analysis and simulation-based fault injection is presented. Subsequently, a summary is provided.

Chapter 3: Development of a fault simulation environment

A concept and implementation of an automatic fault injection technique in the ® Virtuoso® CAD tool is described. An object-oriented fault model library and a fault injection algorithm is described. The approach is capable of injection of device-dependent, layout-dependent and functional fault models.

Chapter 4: Safety-related functional verification

The scope of the safety-related functional verification within the ISO 26262 is described. Several concepts are described in order to facilitate this in a simulation-based approach. A verification plan is derived from the functional verification plan and the FMEDA. The extraction of hardware architectural level fault models from the FMEDA using the functional fault modelling technique is described. This approach is extended from analog electrical to analog heterogeneous systems.

Chapter 5: Fault grouping approach for hierarchical fault injection

A fault grouping approach is proposed for hierarchical fault injection in modular and hierarchical designs. The technique exploits functional fault equivalence at component level and reduces the fault list for top verification. The chapter is divided into two parts: component-level fault injection and fault grouping algorithm. The fault grouping algorithm allows to reduce the fault list based on a hierarchical clustering algorithm and quantitative criteria for the adequacy of the set of fault groups.

Chapter 6: Evaluation of soft faults by global sensitivity analysis

For the evaluation of the effects of soft (parametric) faults, an approach based on global sensitivity analysis is proposed. Randomized fault injection is exercised at component level to identify statistically significant soft faults based on their contribution to the output response's variability. Eventually, the fault list for soft faults is reduced by eliminating statistically insignificant faults. Based on the global sensitivity analysis approach, a metric for the simulation fault coverage is proposed.

Chapter 7: Experimental results

Experimental results are presented for two safety-related automotive circuits. First, a Lithium-ion cell monitoring and balancing module is considered which is part of a high-voltage battery management system. In the simulation test bench, the module including Lithium-ion cells are represented by behavioural models. Using the simulation-based fault injection, failure modes extracted from the corresponding FMEDA are classified in accordance to ISO 26262 with respect to variations in the application parameters.

Second, a low-side gate-driver circuit module is considered which is part of a safety-related automotive System-on-a-Chip. In the simulation test bench,

the module includes RTL logic and analog/mixed-signal components at circuit level. Initially, the safety-related functional verification in terms of compliance with safety requirements and effectiveness of safety mechanisms is demonstrated. Subsequently, the fault grouping algorithm is executed for analog/mixed-signal components of the module. Finally, soft fault models of MOSFET stuck-open and stuck-shorts are investigated in terms of global sensitivity analysis and statistical significance testing.

Chapter 8: Conclusion and outlook

In the final chapter, the presented work is concluded and future work is proposed.

Chapter 2

Related work

In this chapter, the related work on computer-aided safety analysis and simulation-based fault injection is presented. Subsequently, a summary is provided.

2.1 Computer-aided safety analysis

With increasing system complexity, informal safety analysis methods, like FMEA and FTA, become more complex. As a result, the safety analysis becomes error-prone [63]. To address this, computer-aided safety analysis methods are proposed. Research in this field addresses high-level and low-level model-based approaches. From a general point of view, their common purpose is to support safety analysis in terms of evaluation of cause-consequence relationships between faults and effects in a system.

2.1.1 High-level model-based approaches

High-level model-based safety analysis methods use a system model in a high-level description language, like Unified Modelling Language (UML) [64, 65]. Based on this model, safety analysis is automatically executed in a formal manner using software tools [66]. By using dedicated model-checking algorithms, the model can be directly translated into into an FMEA or FTA [67]. Initially, the system model is not necessarily available and must be created manually. This issue is addressed in recent work by automatic safety-related system modelling using a model-to-model transformation algorithm [68].

Nevertheless, high-level model-based approaches scale not very well and thus are not suited to accurately represent more complex systems, like SoCs

[69]. This is addressed by low-level model-based approaches.

2.1.2 Low-level model-based approaches

Low-level model-based approaches are based on a physical model of the electrical, electronic and programmable electronic system. For this purpose, the gate-level and transistor-level netlists can be used. The evaluation of cause consequence-relationships between faults and effects is based on fault simulation results using for example a SPICE simulator.

In [49], a hierarchical fault simulation approach for large and complex systems is proposed in order to identify all failure modes of the components in the system. In this context, the authors emphasize the importance of low-level modelling and fault simulation for safety analysis in order to obtain accurate results of the effects. The simulation results are subsequently used to build an FMEA table.

For digital systems, simulation-assisted FMEA approaches are proposed [70, 63]. The system modelling and fault simulation is executed using high-level behavioural modelling languages particularly for digital systems, like for example SystemC [56, 71].

In multi-disciplinary (heterogeneous) systems like Micro-Electromechanical Systems (MEMS), low-level fault simulation is proposed in [72]. Their approach incorporates the gate-level and circuit-level netlist together with behavioural modelling of non-electrical parts in the system with the hardware description language VHDL-AMS [33].

2.2 Digital fault simulation

In the digital domain, the fault simulation is a well-established method. It plays an important role in the Automatic Test Pattern Generation (ATPG) for structural testing of digital circuits [73]. The availability of discrete stuck-at fault models, that is stuck-at-1 and stuck-at-0, allows in this context for quantification of the respective structural fault coverage.

2.2.1 Fault collapsing

At the logic gate level, all gate input and output nodes are considered fault sites for fault occurrence. The structural fault list comprises two structural faults (sa1, sa0) for each fault site. Additionally, potential bridging faults among the gate nodes can be considered which are not covered by the structural fault list. The fault collapsing technique is used to reduce the fault

list to a reduced fault list. For structural faults, the structural fault collapsing technique is applied [74, 75] and for bridging faults or circuits including fanout branches, the functional fault collapsing technique is applied [76, 77]. Structural fault collapsing is based on the logic gate-level circuit topology. However, functional fault collapsing involves fault simulation to determine functional equivalence among the faults in the fault list. Two faults are functionally equivalent, if they cause the same malfunctioning behaviour observable at the primary outputs of the circuit. Typically, first structural fault collapsing is applied to the parts in the circuit topology without fanout branches. Subsequently, functional fault collapsing is applied in order to further reduce the equivalence and dominance collapsed fault list [78].

2.2.2 Digital fault modelling and injection

Fault modelling and injection in the digital domain can be distinguished into three techniques based on simulator-commands [71], saboteurs [79, 61] and mutations [80, 79]. The simulator command technique, injects a fault during simulation time to the circuit by forcing an arbitrary logic signal or a model parameter to the desired faulty value. The saboteur technique refers to manipulating the circuit for malfunctioning behaviour by adding a fault model instance, the saboteur, which acts at the interface signals of existing instances in the circuit. Finally, mutation refers to the manual manipulation of an instance for malfunctioning behaviour. For fault simulation, the original component is replaced in the circuit by its mutant counterpart. Eventually, the techniques are complementary and usually combined [81].

2.2.3 Simulation speed-up

The long simulation time of complex digital circuits at the gate-level [82] makes it out of the question for extensive fault simulation. Before synthesis, model abstraction is considered for early fault injection campaigns [82]. Common abstraction methods reported in this context are the RTL [82, 83, 84, 85], behavioural level [86, 87, 88, 89, 90] and transaction level [54, 48, 54].

The concurrent simulation method is a prominent approach to the reduction of simulation time in digital circuits by changing the simulator's source code [91]. It is extended in the context of fault simulation in order avoid re-simulation of some parts of the circuit model, which are not affected by the injected fault [92, 93].

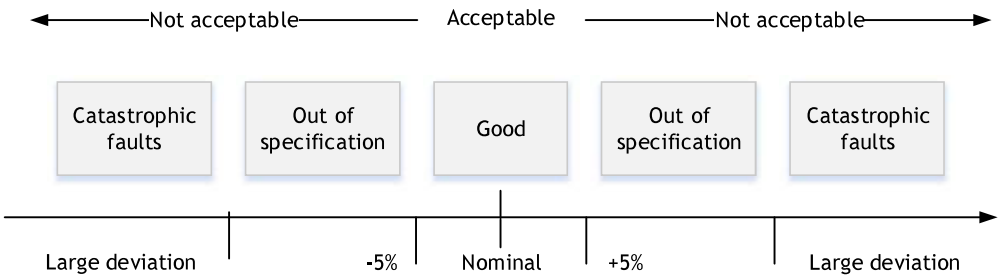


Figure 2.1: Classification of analog circuit faults with ±5% tolerance for deviation from nominal [5].

2.3 Analog fault simulation

The analog fault simulation differs from the digital counterpart due to the absence of a discrete set of universally valid fault models. For example, a short-circuit fault may require a number of simulations with probabilistically varying short-circuit resistance values [94, 95]. This complicates the

- definition of analog fault coverage [96, 97, 98] and
- execution of structural fault collapsing and structural testing [99, 100].

Another inherent ambiguity is related to detecting and classifying a fault based on the resulting circuit output signal failure. With time-/value-continuous signals and voltage-current duality it may be not straightforward to define a discrete cut-off condition for the faulty and the fault-free output response. Thus, the severity of the failure is commonly expressed by the percentage of deviation of a certain attribute (e.g. amplitude) from the nominal output response, see fig. 2.1 [5].

2.3.1 Analog fault classification

Analog faults are classified based on their effect on the primary circuit output signals into two categories [101], see fig. 2.1) [102]:

- Hard faults, also "catastrophic faults", cause an inoperable circuit state, e.g. breakdown of a function like analog-digital conversion.
- Soft faults, also "out-of-specification faults", cause a circuit operation outside acceptable specification limits, e.g. introduction of delay times.

In addition, fault occurrence is distinguished by means of fault classification based on the fault timing into two classes [47]:

- Permanent faults occur at some point in time and are persistent, e.g. short-circuit of signal paths.
- Transient faults occur at some point in time as a single event and disappears after some time or occurs periodically, e.g. power supply disturbance.

Analog fault modelling

The inductive fault analysis (IFA) is a well-established technique to derive potential opens and shorts due to variations in the fabrication process [103, 104, 105]. It is based on defect statistics and a layout-level analysis of adjacent signal paths [106]. Due to its computation-intensive algorithm, IFA is difficult to directly apply for complex circuits [107, 108].

More recently, an alternative layout-based analog fault modelling is reported which uses the parasitic extraction method [109] to derive potential opens and shorts [110]. This approach is motivated by the fact that the increasing trend in technology down-scaling cause parasitic effects which may cause faulty behaviour. In essence, a high parasitic resistance of a long signal path indicates a high likelihood for occurrence of an open and a high parasitic capacitance between two adjacent signal paths indicates a high likelihood of occurrence of a short. Based on the parasitic values, relative likelihoods for opens and shorts fault occurrence is calculated [111].

Circuit-level fault modelling Circuit-level fault models cover signal path defects [107, 94, 95], parametric faults of device parameters [112] and compact fault models for physical fault modelling of a device [113, 39].

Device-specific fault models Fault models for semiconductor devices are proposed with respect to the technology. For bipolar junction transistors (BJTs), commonly six resistive fault models are considered, three high-ohmic in series (open-circuit fault model) and three low-ohmic in parallel (bridging fault model) at the respective emitter, collector and base nodes [102]. For MOSFETs, the common fault models at the switch level are the resistive stuck-open and stuck-short fault models [98]. At a lower level, six resistive/-capacitive fault models (series and parallel) respective to the drain, gate and source nodes are considered [101, 114]. The authors in [102] propose a

standard set of technology-dependent fault models, denoted

$$N_{\mathrm{HF}} = 2(R + C + M) + 6B, \tag{2.1}$$

$$N_{\mathrm{SF}} = 2(R + C) + 2B, \tag{2.2}$$

where N_{HF} is the number of structural faults, N_{SF} is the number of parametric faults. R, C, M and B are the is the number resistors, capacitors, MOS transistors and bipolar transistors in the circuit. Therein, only two fault models for MOS transistors are considered, i.e. stuck-open and stuck-short.

To cover wear-out failure mechanisms, like time-dependent dielectric breakdown of a MOSFET, the compact fault modelling technique is proposed [113, 39]. Thereby, the original device is replaced in the netlist by the corresponding compact fault model. The compact fault model is composed of other compact device models, e.g. MOSFETs, capacitances, resistance, voltage and current sources, etc.

Opens and shorts Opens and shorts may be tractable at component level, however at system level, faults can be totally masked in the error propagation path in analog/mixed-signal circuits [107]. Contrary, weak (also "*soft*") opens and shorts can cause the circuit to function poorly and must be handled respectively, e.g. by a built-in fault diagnosis [107, 115, 116]. The soft open/short faults are simulated in a randomized manner, based on probability density functions (PDF) of potential open and short resistance values [94, 95].

Parametric degradation faults Due to production process fluctuations or, at a later stage of product-life, due to ageing and wear-out, some devices can be subject to parametric degradation. Parametric device faults are based on the respective PDFs of parameter values. Commonly, the parameters are assumed to be normally distributed with mean μ being the nominal parameter value which is subject to some variance σ [39]. Typically, the $\mu \pm 1\sigma$ up to $\mu \pm 3\sigma$ values are considered within acceptable limits and the faulty variation to be the $\pm 6\sigma$ values [102].

Functional fault modelling For functional fault modelling, the circuit is divided into components and each component is considered as a black-box [107]. Based on the assumption that any fault which occurs inside the component propagates to its inputs or outputs, the fault models are injected

directly at the input and output nodes of the components. The fault models are typically modelled in order to violate the component specification limits [117, 118]. Although, functional fault modelling has been validated [118], the overall performance is questioned regarding insufficient correlation of functional fault models to more realistic transistor-level fault modelling [107].

2.3.2 Model abstraction techniques

Abstraction techniques are utilized to reduce the fault simulation time. Additionally, at the beginning product development (e.g. concept phase), the circuit-level design may not be available to be used for fault injection. Therefore, model abstraction techniques are proposed which allow for fault simulation at an early design phase.

Macro modelling

In [119, 120, 121, 122] macro modelling is proposed to replace the original circuit by an equivalent circuit. The equivalent circuit is typically composed of a smaller number of devices and simplified voltage and current source primitives. Fault simulation time can be reduced at the cost of an "acceptable loss" of accuracy [119]. However, this approach requires parameter estimation for the macro-models and the macro models must be created manually.

Behavioural-level modelling

Behavioural-level models (BLM) of analog circuits using analog hardware description languages (AHDL) is a widely accepted technique to support verification activities of analog/mixed-signal systems [123]. Due to significant savings in simulation time and adequate correlations between the transistor-level circuit and BLM, it has become a common method to support fault modelling and simulation activities [124, 125, 126, 127, 128]. Different modelling languages are reported for this purpose, for example Cadence-Verilog-A (previously Cadence-AHDL [118]), SystemC-AMS [71], Verilog-A [129], Verilog-AMS [130] and VHDL-AMS [126].

Hierarchical analog fault simulation using BLMs simultaneously with the circuit-level netlist is multiply reported [131, 55, 132]. In these approaches, BLMs are typically used to propagate the failure of a faulty circuit-level component throughout the design hierarchy to the primary outputs of the system-level circuit. Generally, BLM approach is considered to be less time consuming regarding the modelling aspect compared to the macro-modelling approach [133]. In a more complex approach reported in [134], both BLM

and macro-models are fault simulated together which enables the reduction of fault simulation time compared to transistor level with same accuracy regarding simulation results.

As the manual (static) fault modelling in AHDL means additional modelling effort, statistical fault modelling is proposed to automatically generate a faulty component in AHDL based on statistical fault simulations at circuit level [114, 135, 136, 137, 138]. The general approach is to exercise exhaustive Monte-Carlo fault simulations at circuit level and to train a regression function implemented in AHDL with the simulated data.

2.3.3 Analog functional fault equivalence

The analog functional fault equivalence technique is similar to its digital counterpart. For fault simulation in complex systems, fault groups with functional equivalent faults are identified at the component level and exploited in order to reduce the number of faults to simulate at the system level. It is determined by the similarity of component output responses, subject to different faults.

Fault grouping algorithms

This approach was firstly proposed in [139] for efficient fault simulation and diagnosis of analog/mixed-signal circuits. Based on the component simulation data, two different numeric techniques are proposed to determine the fault groups, namely a

- k-means clustering algorithm [139, 140] and
- fault stratification algorithm [141, 142, 143].

The main disadvantage of the fault stratification algorithm is that the criteria for fault equivalence is the similarity of distance of the faulty to the nominal circuit response. Two faults which have similar distance to the nominal response are not necessarily similar to each other. This is a misconception which is generally avoided in distance-based clustering algorithms [144] and must be considered for the fault grouping technique [116].

Validation of fault groups

The mentioned publications do however not report any objective validation method for choosing an adequate number of fault groups. That is, the validity of the collapsed fault list is based on engineering judgement. In [114] the

optimal number of fault groups is addressed by means of external cluster validation [145]. Therein, the optimal number of fault groups is obtained by comparing the fault groups with a user-specified ground-truth. A validation index is used which has been introduced previously in [146] based on the Mutual Information criterion [114]. The disadvantage of this approach is however, that external cluster validation is based on a ground-truth, that is prior knowledge on how the data can be divided into an adequate set of fault groups. Furthermore, cluster validation based on the Mutual Information criterion states a computation-intensive statistical method, for which a large sample size (i.e. high number of fault simulation runs) is required. However, fault simulations are generally subject to long simulation times and hence may not be suited for this cluster validation method in a straightforward way. Therefore, the authors propose a re-sampling technique called Bootstrap [144] to avoid exhaustive Monte Carlo simulations. The use of the Bootstrap technique in the context of fault simulation is proposed in [147].

2.3.4 Simulator-dependent techniques

The concurrent analog fault simulation technique [148] is based on the assumption, that the iterative calculation of the currents at certain circuit nodes using the Newton-Raphson loop [149] need not to be calculated repeatedly for each fault simulation if the respective node is not affected by the fault model. Several other approaches are reported which are based on concurrent fault simulation in [150, 151]

Another technique to speed-up fault simulations in the context of defect-oriented testing is the Fault Sensitivity Analysis method [152, 153, 154]. In this approach, the main objective is to determine the detectability of faults and not necessarily the continuous waveform of the output failure. According to the authors in [154] this achieves over 1000 times simulation speed-up compared to the conventional transient analysis.

2.4 Heterogeneous fault simulation

Heterogeneous systems are composed of components from different physical domains, e.g. electrical, mechanical, thermal, etc. To simulate such systems, the simulator must be capable of solving equations from these domains. Some commercial tools which allow simulation of heterogeneous systems are Cadence's AMS-Designer [155] and Mentor Graphics' ELDO [156] in combination with the hardware description language VHDL-AMS [32], Analogy Inc.'s SABER [157] with the behavioural language MAST [158] and SMASH

with the behavioural language ABCD [159].

In [49, 160, 161, 162] a fault simulation methodology for MEMS is proposed. The fault injection in [49, 160] includes fault models obtained from the device failure modes of an FMEA.

2.5 Summary

The computer-aided safety analysis allows to automatically evaluate the cause-consequence relationships between faults and effects by using a model-based approach. However, the accuracy and scalability of this approach depends on the model complexity. High-level modelling and evaluation in a formal manner lacks accuracy and scales poorly for complex analog/mixed-signal circuits. For this purpose, low-level modelling and evaluation in a simulation-based manner is suited. However, the simulation time of the circuit netlist must be addressed by an adequate methodology.

Several techniques are reported in order to tackle the long simulation time. These are the model abstraction techniques, the analog functional fault equivalence techniques and simulator-dependent techniques. However, the simulator-dependent techniques apply only for proprietary simulators for which the simulator algorithm can be changed.

Contrary to the digital domain, there is no discrete set of universally valid fault models in the analog domain. This makes it difficult to estimate for example the fault coverage of fault simulation. Nevertheless, in the literature, fault models are proposed with respect to the problem they address, like signal path defects, device degradation, device wear-out, etc.

In a safety-related context, an automotive ECU at the system level may generally comprise components from other domains than analog electrical, for example mechanical, thermal, etc. For the evaluation of the cause-consequence relationships between faults and effects, it is important to also model such components in the simulation test bench. Moreover, in the presence of a malfunctioning behaviour of for example a mechanical component, it is required to evaluate the integrity of the safety-related ECU. Therefore, fault modelling in the heterogeneous domain can be considered.

Chapter 3

Development of a fault simulation environment

In order to facilitate the safety-related functional verification, an adequate fault injection and simulation environment is required. The presented approach is developed within the CAD tool Cadence® Virtuoso®. It includes a fault model library which is in accordance with the ISO 26262 requirements. Additionally, different approaches to automatic fault injection are discussed of which one is implemented and used throughout this thesis.

3.1 Elements of the fault simulation environment

Commercially available CAD tools cover many of the functionalities and automated steps, required to execute simulation-based functional verification. Such functionalities are also useful for safety-related functional verification. Moreover, the fault injection technique must be integrated into the CAD tool. For this purpose, the simulation environment must be extended by additional new elements. In fig. 3.1 the minimum required new elements are coloured in grey. They interact with the regular elements using respective interfaces. The new elements can be described as:

- **Fault list generation** The fault list specifies the faults to be considered for fault injection. The fault list comprises fault models from relevant sources like failure modes of devices, technology data, layout-level analysis or other relevant sources.
- **Fault model library** The fault model library covers the faults specified in the fault list. It can be integrated into the CAD tool in the

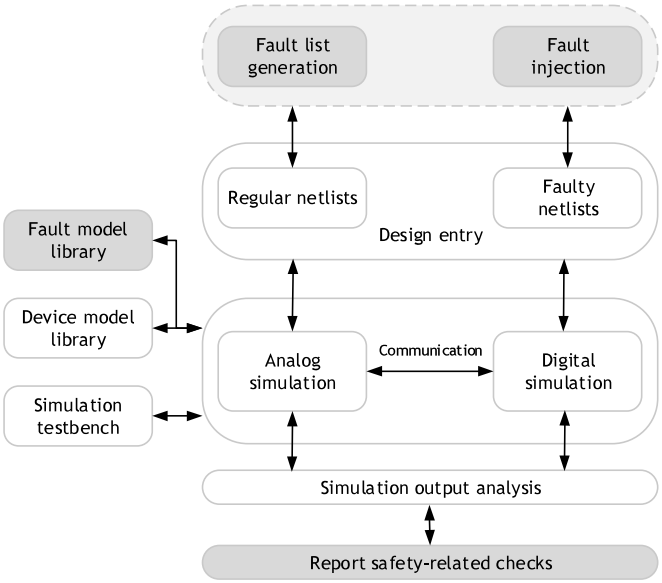


Figure 3.1: Elements of the mixed-signal simulation environment with integrated facility for fault injection.

same manner as a regular design library.

- **Faulty netlist** The faulty netlist is based on the regular circuit netlist but includes a fault model.
- **Fault injection** The fault injection is the procedure which generates the faulty netlist from the regular netlist by using the fault list and the fault model library.
- **Report and safety-related checks** A report which documents safety-related simulation results by using for example checkers.

3.2 Circuit-level fault modelling

Circuit-level fault models cover

- signal path defects and
- device (transistors, resistors, etc.) failure modes.

Signal path defects are modelled using resistive and capacitive fault models. Device failure modes are modelled either by changing the connectivity of the device pins or by changing the device model parameters. Alternatively, more elaborate fault models can be used, like for example the compact fault modelling technique [113, 39].

In terms of the ISO 26262, circuit-level fault models are used for fault injection at the hardware-detailed level.

3.3 Functional fault modelling

For functional fault modelling, the hardware element considered to experience a failure is handled as a black box. Functional fault models are injected at the interface signals of the component in order to initiate a faulty behaviour in the circuit. Fig. 3.2 demonstrates the circuit-level and functional fault modelling techniques (b and c). Therein, a stuck-at high failure mode of a low-side gate driver circuit is realized using both techniques. The functional fault model is a circuit, composed of two ideal switches *S1* and *S2* and a DC voltage source. The behaviour of the functional fault model can be controlled by the switches:

- *S1* open, *S2* closed: stuck-at high
- *S1* closed, *S2* open: nominal circuit behaviour
- *S1* open, *S2* open: high-impedance output
- *S1* closed, *S2* closed: short-circuit (interconnection)

Functional fault models can be added to the fault model library by designing them to be generic and suited for re-use in other circuits and hardware elements. In the example of the functional fault model this can be achieved by a soft-coded high-voltage potential *VHigh* and to allow to set *VHigh*, *S1* and *S2* externally without having to change the implementation of the functional fault model. Additionally, the functional fault model covers four different failure modes for which only a single instantiation of the fault model in the circuit is required.

In terms of the ISO 26262, functional fault models are used for fault injection at the hardware-architectural level.

Re-usability and representativeness

For analog circuits, the analog hardware description languages (AHDLs) are suited for designing functional fault models. Parameters can be passed to the

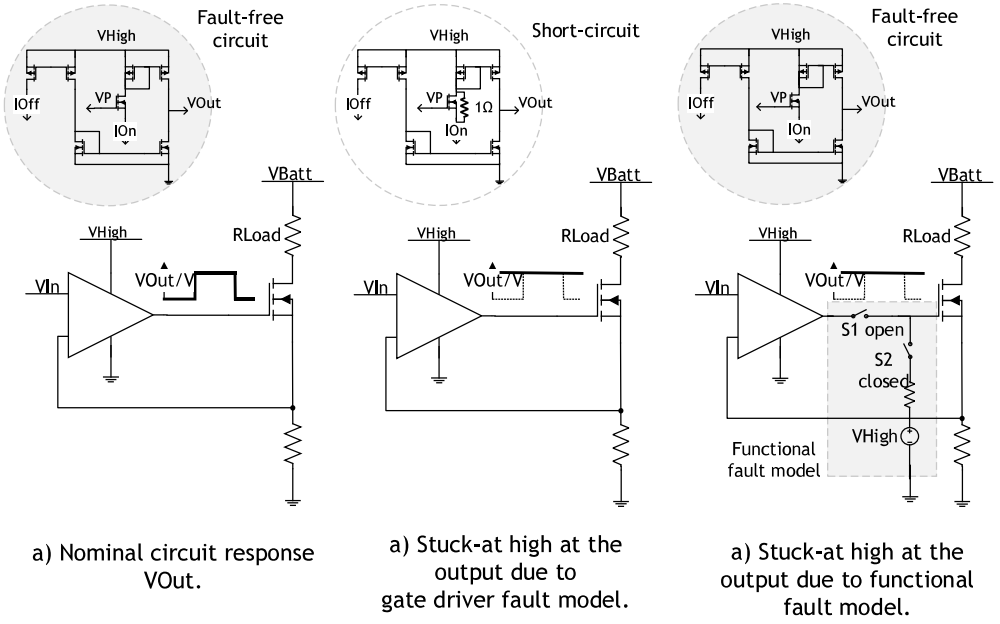


Figure 3.2: Circuit-level and functional fault modelling on a low-side gate-driver example.

model in a generic way and an arbitrary level of abstraction of the fault model can be achieved. A high-level abstraction improves re-usability and allows to keep the circuit complexity from blowing up and thus reduces simulation time. However, these benefits come at the cost of increasing convergence issues during simulation (e.g. high $\frac{du}{dt}$ of a digitally controlled analog variable) [36]. Thus, adequate measures to prevent such issues must be implemented in the functional fault model or the simulator. Additionally, the representativeness of the fault model in terms of its ability to behave like the real failure mode decreases with increasing level of abstraction. Therefore, a compromise regarding re-usability and representativeness must be achieved.

Implementation and performance

The functional fault modelling technique is advantageous when behavioural level or macro models of components are used. Such models describe the component in a more functional and less detailed way by collapsing a number of devices to one or neglecting some devices completely. Thus for the fault injection, some fault sites are missing at the circuit level. In these cases, the functional fault model can be used, as it requires no knowledge on the

implementation of the component.

However, the main disadvantage of the functional fault model is that by acting only at the interface signals of instances, usually at one output signal, it is difficult to be used when a failure mode depends on input signals. Additionally, hardware elements with multiple outputs may experience a failure of more than one output signal. Thus, two simultaneous functional fault models are required for each output signal and each must be set to simulate a different failure mode.

Transient fault models

Transient faults in the digital domain are typically modelled by bit-flips, transients or single-event upsets. Therein, transient faults caused by alpha particles are typically considered which are formally modelled by a double exponential function. In most analog and mixed-signal circuits and therein particularly power electronics, the alpha particle may have small to no effect on the circuit performance (max. amplitude induced current in the range of $2 - 10$mA for the duration of $100 - 600$ps [163]). In analog/mixed-signal circuits, potential transient failures are power supply disturbances or oscillations e.g. due to a permanent fault in a linear voltage regulator. Here, the functional fault modelling technique can be applied to inject the transient failure mode to the component's output signal. The transient failure pattern must be known for the parametrization of the functional fault model, e.g. amplitude, duration, and oscillation frequency, damping factor, etc.

3.4 Implementation of a fault model library

The fault model library comprises circuit-level and functional fault models. A functional fault model is proposed which is generic regarding the fault types which can be injected and includes transient failure modes. The concept of the functional fault model design can also be used in other energy domains. Thus, it can be used for fault injection in heterogeneous systems. Additionally, the library includes digital fault models.

3.4.1 Electrical domain

For fault injection in the electrical domain, circuit-level fault models and functional fault models are used.

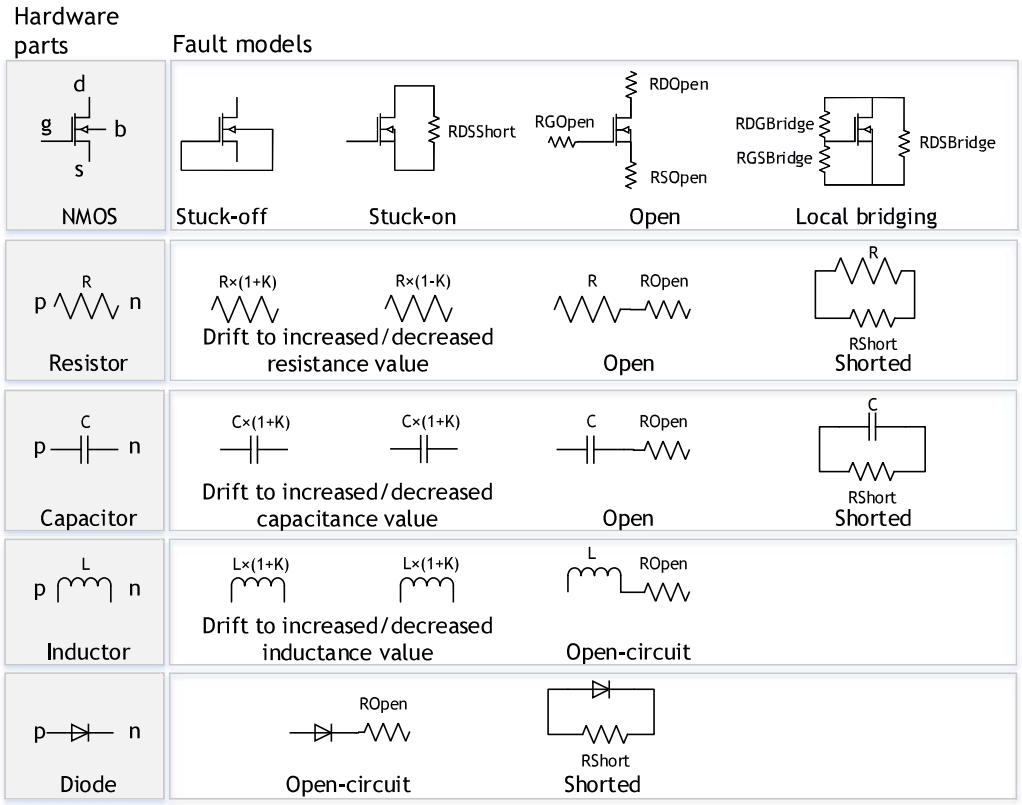


Figure 3.3: Some types of hardware parts and fault models.

Circuit-level fault models

Fault modelling at the circuit-level includes resistive and capacitive fault models as well as parametric fault models. Fig. 3.3 illustrates some circuit-level fault models for common devices in analog/mixed-signal circuits.

Functional fault models

In 3.4, a schematic of the functional fault model is shown. It is implemented in VHDL-AMS. The fault model can be configured in a generic fashion by a set of parameters, see tab. 3.1. The functional fault model can be inserted to the interface signals of instances (primitives or components). For this purpose, the regular connection of an instance terminal to a net, e.g. *net0*, is removed. Subsequently, the fault model is inserted in the schematic by connecting point B to the instance terminal and connecting point A to *net0*. An

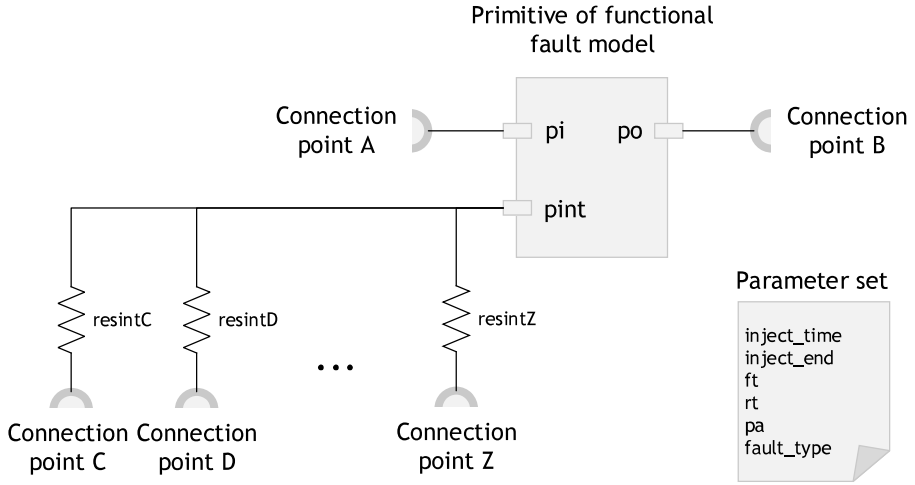


Figure 3.4: Basic schematic of the functional fault model.

Parameter	Description	Unit
$inject\_time$	Fault activation time	Time in s
$inject\_end$	Fault de-activation time	Time in s
ft	$fault\_type = 2$: Falling edge duration	Time in s
	$fault\_type = 4$: Damping factor of oscillation	Frequency in Hz
rt	$fault\_type = 2$: Rising edge duration	Time in s
	$fault\_type = 4$: Period of oscillation	Frequency in Hz
pa	$fault\_type = 2$: Pulse amplitude	Voltage in V
	$fault\_type = 4$: Amplitude of oscillation	Voltage in V
$fault\_type$	=0: no fault,=1: open-circuit,=2: pulse (offset),	-
	=3: short-circuit, =4: damped oscillation	-

Table 3.1: Parameters of the functional fault model.

arbitrary number of short-circuit faults can be realized by using the terminal *pint* and connecting parallel resistances $resintC, resintD, ..., resintZ$ to the respective target short-circuit nets. By setting one or more resistances to short-circuit values, interconnections to the respective nets can be injected.

Architecture In the basic architecture of the functional fault model, two voltages and currents are defined:

- *vnet* and *inet*: Voltage across and current through ports *pi* to *po*
- *vint* and *iint*: Voltage across and current through ports *pi* to *pint*

The analog signals are digitally controlled by the parameters *fault_type*, *inject_time* and *inject_end* in if-then-else statements. One fault model respective to the corresponding fault type is defined in one state. The duration of the fault occurrence is determined by timing parameters *inject_time* and *inject_end*. For a permanent fault, *inject_end* must be set beyond end of simulation time.

Fault types and implementation The functional fault model comprises two structural fault types, the open and short-circuit, as well as two transient fault types, the pulse-shaped and sine-shaped transients. Additionally, by de-activating it, no fault is injected, thus the circuit exercises its nominal behaviour. Generally, the presented functional fault modelling concept is modular and extendible to cover more than the described fault types.

The fault models respective to the fault type are described in the remainder using VHDL-AMS concurrent statements notation [33].

Nominal (fault-free) By setting *fault_type* to 0, the fault model is de-activated. For example, this state is the current state until another fault type is activated after *inject_time* or de-activated after *inject_end*. In this state, the functional fault model shorts *pi* to *po* and disconnects *pi* from *pint* by:

```
1 vnet == 0.0; -- ideal short pi to po
2 iint == 0.0; -- ideal open pi to pint
```

Here and in the remainder, ideal opens (i==0.0) and ideal shorts (v==0.0) are considered. Alternatively, open and short resistances can be defined for the respective fault type. This state can then be defined by:

```
1 vnet == rshort*inet; -- resistive short pi to po
2 vint == ropen*iint; -- resistive open pi to pint
```

Open-circuit The open-circuit fault is activated by setting *fault_type* to 1. In this state, the functional fault model disconnects *pi* from *po* and *pi* from *pint* by:

```
1 inet == 0.0; -- ideal open pi to po
2 iint == 0.0; -- ideal open pi to pint
```

Pulse-shaped transient This fault model is based on the work presented in [128]. A pulse-shaped transient disturbance of a voltage signal is activated by setting $fault\_type$ to 2 and is designed using the ramp() function implemented in VHDL-AMS [33]. In this state, the functional fault model inserts a non-linear voltage source between the ports *pi* and *po* and disconnects *pi* from *pint* by

```
vnet == -intV'ramp(rt,ft) -- ramp voltage between pi and po
iint == 0.0 -- ideal open pi to pint
...
Vp : PROCESS(inject) -- digital driver of pulse-shaped transient
BEGIN
        intV <= 0.0;
        IF inject = '1' THEN
                IF fault_type = 2.0 THEN
                        intV <= pa;
                ELSE
                END IF;
        ELSE
        END IF;
END PROCESS;
```

The pulse amplitude *pa* is passed to the variable $intV$ for the duration of the active fault, indicated by the variable $inject$ set to 1. The ramp function follows the value of $intV$ with the specified rising and falling edge durations rt and ft, respectively.

Short-circuit The short-circuit fault is activated by setting $fault\_type$ to 3. In this state, the functional fault model shorts *pi* to *po* and *pi* to *pint* by:

```
vnet == 0.0; -- ideal short pi to po
vint == 0.0; -- ideal short pi to pint
```

Damped sine-shaped transient A sine-shaped transient disturbance of a voltage signal is activated by setting $fault\_type$ to 4 and is designed using the exp and sine functions implemented in VHDL-AMS. In this state, the functional fault model inserts a non-linear voltage source between the ports *pi* and *po* and disconnects *pi* from *pint*: NOW is a pre-defined function provided by VHDL-AMS and returns the current simulation time when it is called [33]. The period of oscillation is set by rt. Additionally, a damped oscillation can be realized by the setting a damping ratio ft.

```
1 vnet == -pa*exp(-(NOW-inject_time)*ft)*...
2 sin(2.0*MATH_PI*rt*(NOW-inject_time));
3 -- damped sine voltage between pi and po
4
5 iint == 0.0; -- ideal open pi to pint
```

3.4.2 Fault injection in heterogeneous systems

The functional fault modelling approach can be used for fault injection in other conservative energy domains (natures) in the same fashion as in the electrical. For this purpose, the analog modelling constructs for diverse natures provided by VHDL-AMS are used in analogy to the electrical nature, see tab. 4.1. Thus, other energy domain systems are assumed to be modelled in accordance to generalized Kirchhoff's circuit laws. The architecture of the functional fault model is illustrated with respect to the effort and flow aspects in fig. 3.5.

3.4.3 Digital fault models

For completeness, the fault model library comprises digital functional fault models designed in VHDL. The digital functional fault model are also generic with respect to the fault type and comprises the following fault types for a single-bit signal:

- Stuck-at
- Inverted
- Single-event upset (SEU) and single-event transient (SET)

In this work, fault injection in digital hardware elements is only considered for the top-level design and only functional fault models are considered. That is, the digital fault models are injected at the primary inputs and outputs of the digital hardware elements.

3.5 Automation concepts

In this section, the sequential arrangement of procedures required for fault injection and simulation automation is described. The automation is implemented for a default set of device-specific fault list, for a layout-based fault list and a custom fault list.

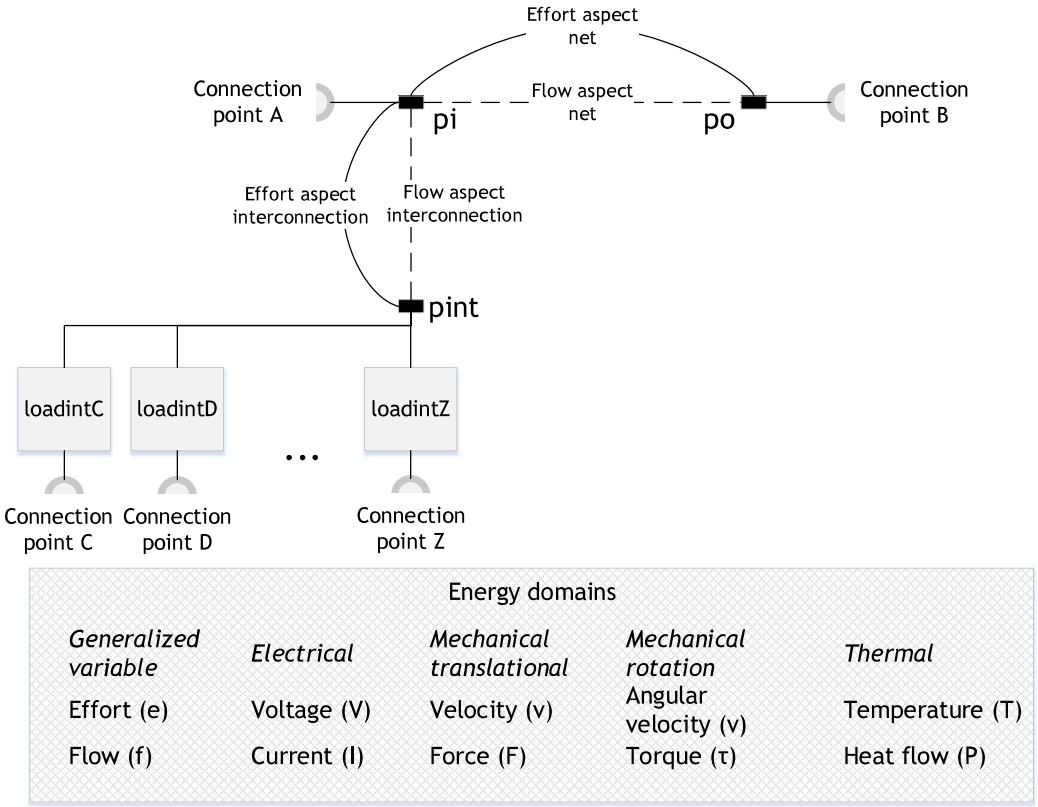


Figure 3.5: Domain-independent architecture of the functional fault model.

3.5.1 Dynamic fault injection

The flow diagram for the automation concept for a dynamic fault injection approach is shown in fig. 3.6. In the dynamic fault injection approach, all fault models corresponding to the faults in the fault list are at once inserted into a regular netlist (copy). The faulty netlist comprises all fault models which will be simulated. The inserted fault models are not active until the scripts for simulation with the test bench are created. Instead, the fault models are initialized by dynamic parameter assignment. The netlist which comprises the fault models with the dynamic parameters is called *generic faulty netlist* in this work. Dynamic parameters are overwritten during simulation by faulty values in order to activate them.

The advantage of this approach opposed to static fault injection, i.e. one netlist for each fault, is that during pre-processing, netlisting and compilation time can be saved. Moreover, the database of faulty netlists is reduced to a

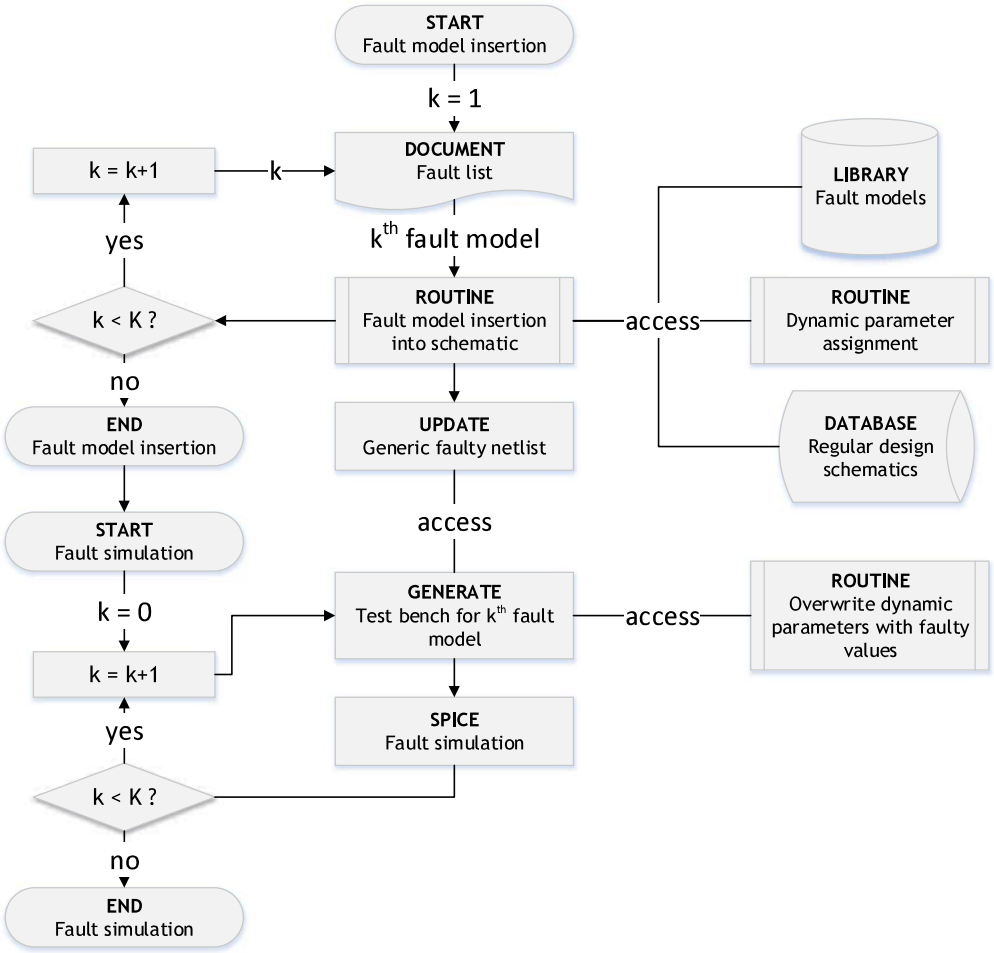


Figure 3.6: Flow diagram of the automation concept for dynamic fault injection.

single faulty netlist which makes it easier to handle in terms of automation.

The approach starts with the selection of the first fault from a fault list in which each fault is assigned to a number in increasing order, i.e. $k = 1, ..., K$ and K is the last element of the fault list. In the fault model insertion routine, a fault model corresponding to the k^{th} fault is selected from the fault model library and instantiated in the target regular netlist from which a generic faulty netlist is generated. This procedure is repeated until $k = K$. Subsequently, the fault simulation starts and is shown in fig. 3.6 in an iterative manner. However, parallel computing is an alternative in order to

to speed-up the total fault simulation time.

3.5.2 Implementation in a CAD tool

The automation of dynamic fault injection is implemented in Cadence® Virtuoso®. The design automation language SKILL/SKILL++ [164] is used for fault injection at schematic-level in association with the OpenAccess (OA) database framework [165]. This implementation has several advantages:

- Fault model insertion algorithm is independent from the netlist format.
- Schematics can be transformed after fault model insertion into any netlist format using the dedicated netlister.
- Direct accessibility of OA database and relevant design information instead of text-based parsing of a netlist.

The test benches for fault simulation are automatically created and simulated with the Executable Verification Plan (XVP) regression tool [166]. As simulator, for example the Cadence's AMS-Designer [155] can be used.

The fault model insertion algorithms are distinguished with respect to the fault modelling techniques: Device-dependent, layout-specific and functional.

Device-dependent fault injection

Device-dependent fault models at the primitive level can be derived from reliability handbooks [167] and technology data. Latter is typically intellectual property of a semiconductor manufacturer. If such information is not available, a default set of device-dependent structural and parametric fault models are provided, see list. 3.2.

The algorithm scans the schematic for the available device instances. Based on the device type, the respective fault models are inserted. Thus, the number of fault models comprised in the generic faulty netlist is the sum of all structural and parametric faults, $N_{\text{structural}}$ and $N_{\text{parametric}}$, see eq. 3.1 and 3.2, respectively. R, C, D, B and M are the number of resistances, capacitances, diodes, BJTs and MOSFETs, respectively. However, the presented approach is generally extensible to cover more devices and more failure modes.

$$N_{\text{structural}} = 2(R + C + D) + 6(B + M) \tag{3.1}$$

$$N_{\text{parametric}} = 2(R + C + B) \tag{3.2}$$

The fault models are defined in an object-oriented fault model library for which the UML diagram is illustrated in fig. A.1. A fault model class

Hardware part	Local structural faults	Parametric faults
MOSFET M	R_{open},R_{short} (including stuck-short, stuck-open)	-
BJT B	R_{open},R_{short}	Gain $\beta_{fault} = \beta_{nom}(1 \pm K)$
Resistance R	R_{open},R_{short}	Value $R_{fault} = R_{nom}(1 \pm K)$
Capacitance C	R_{open},R_{short}	Value $C_{fault} = C_{nom}(1 \pm K)$
Diode D	R_{open},R_{short}	-

Table 3.2: Default set of faults covered by the fault insertion algorithm at the hardware detailed design level.

is defined with respect to the device type. However, all fault model classes are derived from a master fault model class which incorporates elementary attributes and methods, like a method for fault injection, see A.2. Due to the object-oriented design, the method for fault injection runs different routines depending on the class of the fault model object, see list. A.1. For example, the class definition of a functional FET fault model (resistive stuck-open/-short) is shown in list. A.3 and with a complementary routine in list. A.4 for short-circuit fault insertion. The fault model is instantiated in the schematic, however not visibly connected by nets but inside the design database. The fault models are instantiated at proximate distance to the regular design by using a xy-positioning routine, see list. A.5.

Layout-dependent fault injection

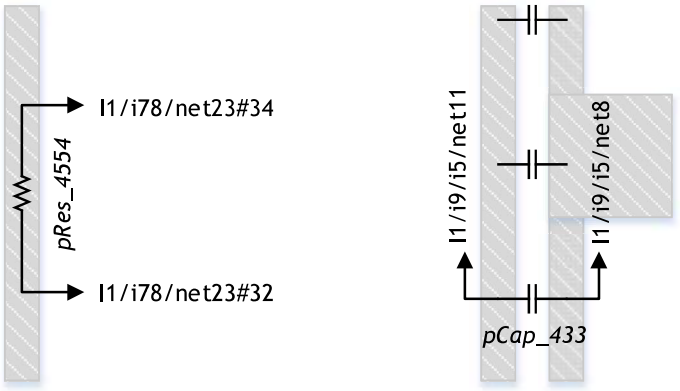
Layout-dependent fault model insertion algorithm uses the parasitic capacitance and resistance values in a layout-extracted cell view in order to calculate the relative likelihood of potential short and open faults, respectively. Fig. 3.7 shows an example set of parasitics extracted from the design layout.

The algorithm proceeds in two stages:

1. Determine the parasitic threshold values for which fault injection is considered necessary based on a cumulative relative likelihood calculation.

2. Fault injection of opens and shorts only for parasitics whose values are greater-than or equal the respective threshold values.

Fig. 3.8 illustrates the threshold calculation algorithm for a given cumulative relative likelihood target value which should be covered by fault



a) Parasitic resistor a) Parasitic capacitor

Figure 3.7: Example of layout extracted parasitic resistance and capacitance.

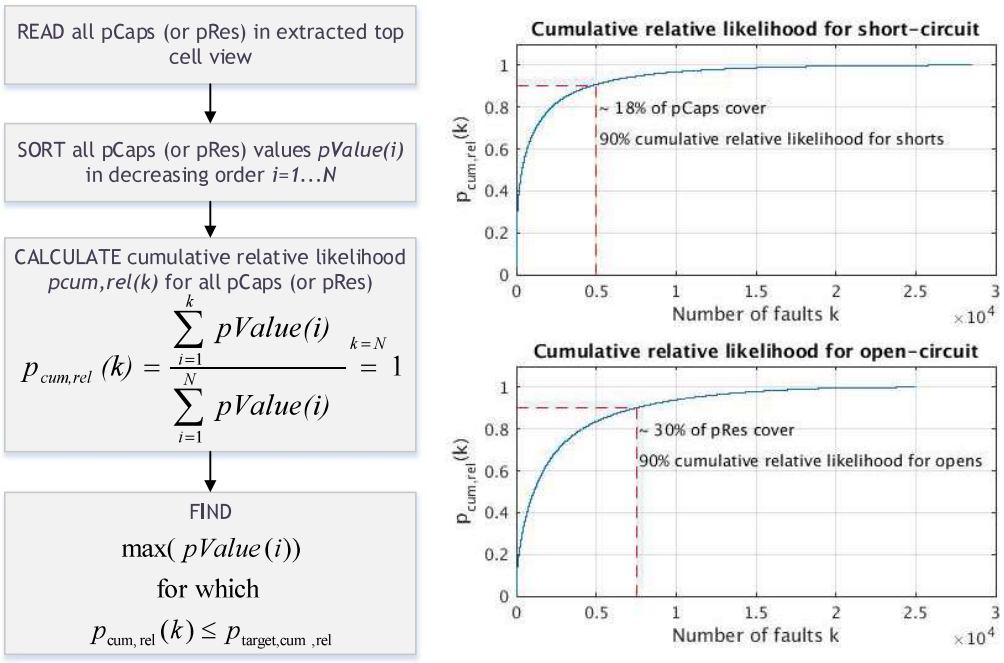


Figure 3.8: Algorithm for determining the parasitic threshold values.

injection. The algorithm for threshold calculation is shown in list. A.6. After sorting the parasitic capacitors (or resistors) in decreasing order from $i = 1, ..., k, ..., N$, the relative cumulative likelihood of opens (or shorts) is calculated by

$$p_{cum,rel}(k) = \frac{\sum_{i=1}^{k} pValue(i)}{\sum_{i=1}^{N} pValue(i)}, \tag{3.3}$$

where $p_{cum,rel}(k) = 1$ for $k = N$ (maximum coverage of all layout-dependent opens and shorts). If $p_{cum,rel}(k) = 1$ is chosen, all parasitics are considered for fault injection. However, based on a plot of $p_{cum,rel}(k)$ over the number of parasitics (i.e. number of faults) k, it can be seen in fig. 3.8 that a minor set of parasitics covers over 90% of the cumulative relative likelihood. For this purpose, a target cumulative relative likelihood $p_{target,cum,rel}$ is passed to the algorithm. It subsequently quantifies the maximum values of parasitic capacitors and resistors for which

$$p_{cum,rel}(k) \leq p_{target,cum,rel}. \tag{3.4}$$

These values are used as threshold values in the fault injection algorithm. Parasitic capacitors and resistors which are greater or equal to the threshold are considered as relevant for fault injection.

Custom fault list

The user-specific fault list is suited for fault injection using the functional fault model. The user-specific fault list comprises the fault sites at which the functional fault models are inserted. Additionally, potential short-circuit sites and parameters of existing instances in the schematic can be addressed (see fig. 3.4). The user-specific fault list is written in a certain syntax which can be read by the fault insertion algorithm. Listing 3.1 shows the syntax in Backus Naur notation. Per default, the short-to-ground fault is considered in the functional fault model.

Listing 3.1: Fault list in Backus Naur notation for the fault insertion algorithm at the hardware architectural design level.

```
1 (*1: Fault list with multiple lines*)
2 <faultlist> ::= {<aLine> <CR> <LF>}
3
4 (*2: Definition of one line*)
5 <aLine> ::= <libName> <cellName> <viewName> ';' {<ffunc> ';' | <
      fpar> ';'}
6
7 (*3: Definition of functional fault model insertion*)
8 <ffunc> ::= <fsinglesig> | <fmultisig>
```

```
9
10 (*4: Architecture-dependent fault insertion*)
11 <fsinglesig> ::= <instName> <pinName> {<shortNetName>}
12 <fmultisig> ::= <instName> <pinName>'('<msBit>:<lsBit>')' <
       bitNum> ...
13 {<shortNetName>}
14
15 (*5: Definition of parametric fault model insertion*)
16 <fpar> ::= <instName> 'CDF!' <parName>
```

3.6 Summary

The fault injection technique is integrated into the CAD tool Cadence® Virtuoso® including a fault model library and procedures for fault injection. A fault model library is included into *Cadence*™in the same fashion as regular device model libraries. For circuit-level fault injection, the fault model library comprises fault models which correspond to the state-of-the-art in the analog and digital domain. For component-level fault injection, the fault model library comprises a functional fault model by which circuit-level fault models can be injected, as well as arbitrary bridging faults and transient faults. It is configurable regarding the fault type it implements during simulation, the fault timing as well as the parametrization of the fault types (e.g. frequency of oscillation failure mode).

A dynamic fault injection and simulation approach is implemented which allows to automatically execute fault injections, generate faulty test cases and run fault simulations in *Cadence*™. By using this automation, very high fault coverage can be achieved at the cost of almost no manual effort. However, this is also limited by the typically very long simulation times for analog/mixed-signal circuits. Running as many fault simulations as possible in a brute-force manner is not an option for realizing a high fault coverage in complex circuits. Therefore, adequate techniques must be applied in order to reduce the simulation time in general and in particular to reduce the number of faults to simulate.

Chapter 4

Safety-related functional verification

This chapter addresses the safety-related functional verification in accordance to ISO 26262 [4]. Generally, it is considered as an extension of the state-of-the-art functional verification methodology and covers safety requirements and the occurrence of random hardware failures within verification.

4.1 Scope within ISO 26262

The safety-related functional verification addressed in this work covers the

- evaluation of safety goal violations due to random hardware faults (A),
- verification of the design implementation with safety requirements (B), and
- verification of the effectivity of safety mechanisms (C).

All three state interrelated aspects of the safety-related functional verification. However, (A) is required at the item level. That is, the simulation test bench must represent the circuit within the safety-critical application. The violation of safety goals can only be determined if the safety-critical equipment is also represented in the test bench. (B) is required at hardware level or system level. The corresponding safety requirements are the hardware safety requirements and technical safety requirements. (C) is dedicated to the safety goals and is therefore required at the item level.

For the simulation-based verification approach, the test bench and the design-under-verification must allow to accurately evaluate the effects of random hardware failures.

4.2 Evaluation of effects of random hardware failures

The evaluation of effects of random hardware failures plays a central role in the safety-related functional verification. It is the basis for the

- failure mode classification into safe faults, single-point/residual faults and multi-point faults (latent, detected and perceived),
- determination of compliance with safety requirements, and
- determination of diagnostic coverage of safety mechanisms.

The confidence of the safety-related functional verification and safety analysis generally depends on the accurate evaluation of effects of random hardware failures [72].

4.2.1 Generic architecture of a safety-related design

In order to realize safety-related functional verification in a simulation test bench, several considerations are made on a generic architecture of safety-related designs, illustrated in fig. 4.1. The architecture of complex ICs, like system-on-chips (SoCs), is typically designed in a hierarchical and modular manner. The design comprises a number of hierarchy levels, starting by the first level which is typically a component comprised of primitives like transistors, resistors, etc. The second level is comprised of a number of components which communicate via interface signals. After an arbitrary number of intermediate levels, the top level comprises all components of the design. In a safety-related circuit this includes the safety-critical equipments (systems) as well the safety mechanisms.

4.2.2 Error propagation

Fig. 4.1 illustrates the propagation of an error from the first level to the top level of the hierarchical architecture. An error is caused by for example a fault of a transistor device. At the top level, the safety-critical system comprises a signal sg which is controlled by the safety-related circuit. Safety goals are assigned to the sg signal. The safety mechanism is implemented in order to prevent the state variable sg from violating a safety goal. The safety mechanism may act in different ways, for example:

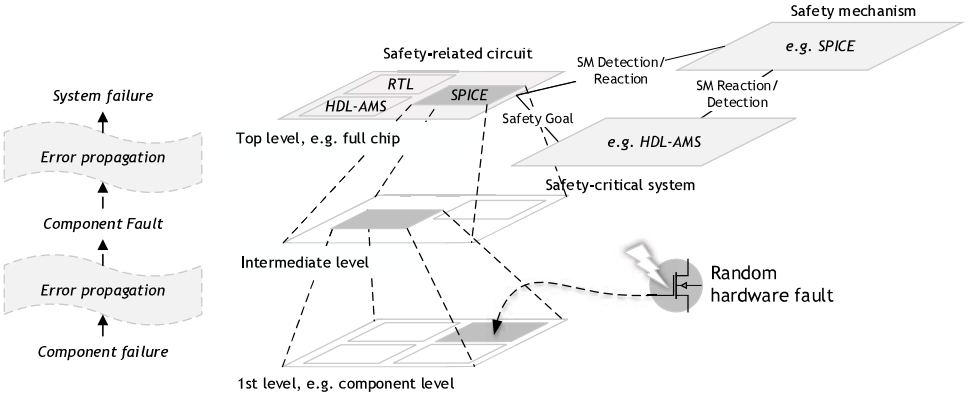


Figure 4.1: Error propagation of a random hardware fault to a top level failure in a safety-related circuit with hierarchical architecture.

- Detects the failure of the safety-related circuit via *sm_detect* and controls its effect on the safety-critical system by interfering to the safety-critical system via signal *sm_react*.
  Example Critical threshold for over-voltage at the electrodes of a battery cell is exceeded and a safety switch is activated to decouple the cell from the safety-relevant circuit.

- Detects a critical deviation in the safety-critical system via *sm_detect* and initiates the safety-related circuit via signal *sm_react* to react respectively.
  Example Critical threshold for over-temperature of a battery cell is exceeded during charging and signal to safety-relevant circuit is sent to stop the charging process.

Eventually, random hardware faults occur at a very low architectural level. Their effects on the other hand are be evaluated at a higher level, e.g. item level.

4.2.3 Realization of item-level fault simulations

Item-level fault simulation may necessitate non-E/E/PE components to be part of the simulation test bench. In order to include such components, heterogeneous behavioural modelling languages like VHDL-AMS can be used.

Failures in heterogeneous systems

In order to initiate hazardous events in a simulation test bench or to evaluate the integrity of the ECU in the presence of non-E/E/PE failures, heterogeneous fault injection is conducted.

The functional fault model proposed in this work can be used for this purpose. In tab. 4.1 examples for the analogy of errors, faults and failure modes in the electrical, mechanical and thermal energy domains are listed.

Acceleration of fault simulation

Abstraction techniques using behavioural level or macro modelling are the state-of-the-art in the functional verification of analog/mixed-signal circuits [37, 38]. The simulator-dependent techniques on the other hand are yet only available for proprietary simulators. Moreover, their implementation into commercial simulators is not possible because access to the source code is restricted.

Moreover, two options depending on the fault modelling technique are used in this work to accelerate fault simulations:

- **Mixed-mode/mixed-level simulation with circuit-level fault modelling** All components are replaced by abstract models, except the faulty component which is kept as the transistor-level model. The abstract models propagate the error.

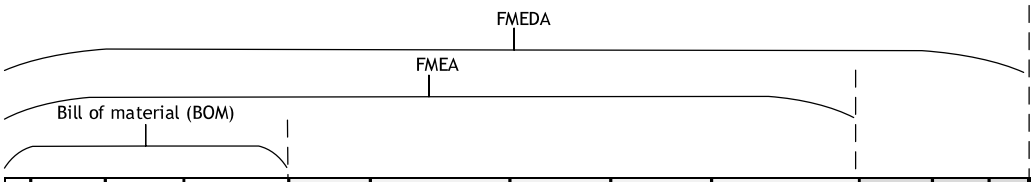
- **Behavioural-level simulation with functional fault modelling** All components including the faulty component are replaced by abstract models. The faults are injected to the interface signals of the behavioural-level components.

Reduction of the fault list

Faults and failure modes with low failure rates/likelihood can be eliminated from the fault list. This approach is based on the assumption that a risk of a hazard is inherently present in any system but by preventing faults which are more likely to occur, the absence of unreasonable risk can be achieved. In this context, fault simulation is only exercised for the prioritized set of most likely faults.

Table 4.1: Examples of fault, error and failure relation in electrical, mechanical and thermal domain which can be represented with a functional fault model.

Energy domains	**Electrical**	**Trans. mechanical**	**Rot. mechanical**	**Thermal**
Effort aspects	Voltage u	Displacement s, Velocity v	Angle φ, Angular velocity α	Temperature T
Flow aspects	Current i	Force F	Torque M	Heat flow P
Fault types	**Examples of hardware fault, error and failure mode relation in different energy domains**			
Open-circuit	*Fault:* Galvanic isolation at ADC's input pin *Error:* Only (non-) inv. diff. voltage at ADC *Failure:* ADC is stuck-at H/L	*Fault:* Motor torque shaft break *Error:* No load transfer *Failure:* Idle running motor	*Fault:* No coupling to dissipator *Error:* No heat transfer *Failure:* Rise of circuit temperature	
	Explanation: Failure during load transfer from point A to B in the presence of effort aspects $\{u, s/v, \varphi/\alpha, T\}_{A-B} \neq 0$ *resulting in flow aspects* $\{i, F, M, P\}_{A-B}$ *very low to zero*			
Short-circuit	*Fault:* Interconn. of ADCs adjacent input pins *Error:* ADC writes wrong output *Failure:* ADC reads const. input voltage 0V	*Fault:* Radial load applied to torque shaft *Error:* Load distribution *Failure:* Overloading of motor	*Fault:* Ambient temp. out of spec. *Error:* Ref. temperature too high *Failure:* Rise of circuit temperature	
	Explanation: Failure during load transfer from point A to B with additional undesired coupling to point C in the presence of effort aspects $\{u, s/v, \varphi/\alpha, T\}_{A/B-C}$ *resulting in flow aspects* $\{i, F, M, P\}_{A/B-C} \neq 0$			
Single event transient (SET), offset	*Fault:* EMI to ADC pin(s) *Error:* ADC writes wrong output *failure:* differential voltage oscillates	*Fault:* Impact applied to torque shaft *Error:* Load distribution *failure:* load motion disturbances	*Fault:* Loose coupling to dissipator *Error:* Discontinuous heat transfer *failure:* rise of circuit temperature	
	Explanation: Failure during load transfer from point A to B where effort aspects $\{u, s/v, \varphi/\alpha, T\}_{A-B}$ or flow aspects $\{i, F, M, P\}_{A/B-C}$ *experience a transient disturbance*			



#	Material	Function	Type	Hard/Soft error	Block failure mode	Distribution	Origin	Effect description	Latent analysis	FFM	SM
...	...	...	...	...	...	...	...	...	...	...	...
11	V_reg	Voltage regulator	Main supply	HE	I/O short	30%	LDO fault catalog	Over-voltage (chip destruction)	No	FFM01	SM1
12				HE	I/O open or short to GND	30%	LDO fault catalog	No supply for chip	No	FFM08	SM1
13				HE	Output at higher value (up to 10%)	7.5%	Max. ratings in spec	Over-voltage	Yes	FFM01	SM2
14				HE	Output at lower value (up to 10%)	7.5%	Max. ratings in spec	Under-voltage	Yes	FFM02	SM2
15				HE	Output oscillates	10%	LDO fault catalog	Modulated output no effect: +/- 0.5V	Yes	FFM01 FFM02	SM2
...	...	...	...	...	...	...	...	...	...	...	...
56	T1_1	Power MOSFET	High-voltage switch	HE	Drain-source short	40%	Reliability handbook	Switch cannot be opened	No	FFM03	SM4
57				HE	Drain-source open	20%	Reliability handbook	Parallel T2_1: Temp. Increase	Yes	X	X

Figure 4.2: Simplified extract of an FMEDA.

4.3 Verification plan

The verification plan for safety-related functional verification is based on the general functional verification plan and a safety analysis method like for example an FMEA or FTA. In the industry, the development of safety-related products is typically accompanied by safety analysis tools. Therefore, they are usually available and can be exploited in this context.

The functional specification-driven verification plan comprises for example relevant test cases, test conditions, hardware specification limits, etc. The verification plan is extended by by safety-related content. Additionally, one test case is created for each fault injection campaign. Hardware safety requirements or technical safety requirements are derived from the specification limits of the product [4].

4.3.1 FMEDA-oriented approach

FMEAs are generally composed of a bill of material (BOM), that is a listing of devices and components (materials) of the design, see fig. 4.2 [167]. For each material, the corresponding failure modes, failure rate distributions and

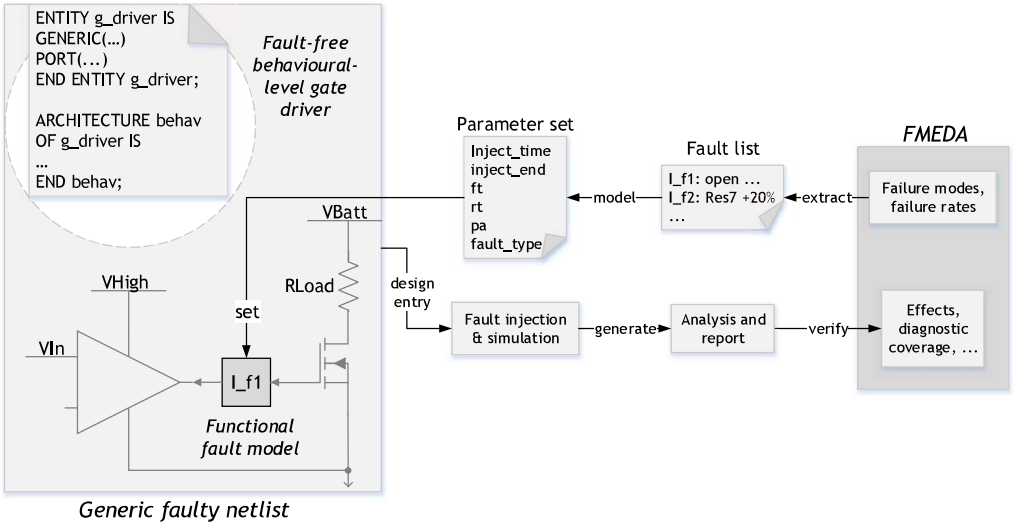


Figure 4.3: Implementation of the FMEDA-oriented simulation-based safety analysis.

expected effect descriptions are specified. The FMEA can be extended to an FMEDA by providing additional information on fault classification (e.g. latent analysis), the mapping of material failure modes to application-specific failure modes (functional failure modes (FFMs)) and the diagnostic coverage of safety mechanisms (SMs).

In this work, the FMEDA is utilized to plan the safety-related functional verification. The simulation-based approach can be used to verify the FMEDA in terms of component failure modes, effects, functional failure modes, etc.

4.3.2 Implementation of functional fault injection

Functional fault models are used to inject faults into the interface signals of components in order to initiate the respective component failure mode during simulation, see fig. 4.3. For this purpose, first a fault list is extracted from the FMEDA. The respective failure rates (and distributions) can be used to rank and eliminate faults in the fault list by their likelihood of occurrence. The parameter set of the functional fault models are configured in accordance to the fault list. Fault simulations are run and a safety-related report is generated. Based on the reported results, the properties of the FMEDA can be verified.

4.4 Summary

In this chapter, the scope of the safety-related functional verification in accordance to the ISO 26262 is explained. Methods to realize this in a simulation-based manner are proposed with respect to the architectural property of the circuit design. The conventional functional verification plan is extended by safety-related content derived from an FMEDA. Functional fault models are used to inject component failure modes. The concept of functional fault modelling is extended for fault injection in heterogeneous systems.

Chapter 5

Fault grouping approach for hierarchical fault injection

The confidence of the safety-related functional verification generally depends on the number of faults injected and the accurate circuit representation by a model [4]. Therefore, the gate-level and transistor-level netlists are suited for this purpose. Long simulation times of such netlists can be addressed by mixed-mode/mixed-level simulation [37, 38] with circuit-level fault modelling. However, due to functional fault equivalence in analog/mixed-signal circuits, many circuit-level faults may cause similar failure modes at the component level and propagate to a similar effect at top level. This introduces redundant fault injection campaigns due to the re-occurrence of effects which have already been simulated and analysed.

Functional fault equivalence can be exploited by the fault grouping technique in which faults are collected in one group if the corresponding component-level circuit responses are similar. This section contributes to the fault grouping technique by proposing an algorithm which is capable of processing similarities between time-continuous waveforms as circuit responses. This is important in the safety-related context because of the qualitative distinction of permanent and transient failure modes. Moreover, in the proposed approach, functional fault equivalence can be quantified for arbitrary number of stimuli pattern by which the component is exercised and for arbitrary number of output ports. For fault grouping, a hierarchical clustering algorithm and numeric cluster validation is used which does not necessitate prior knowledge on the division of faults into an optimal number of fault groups.

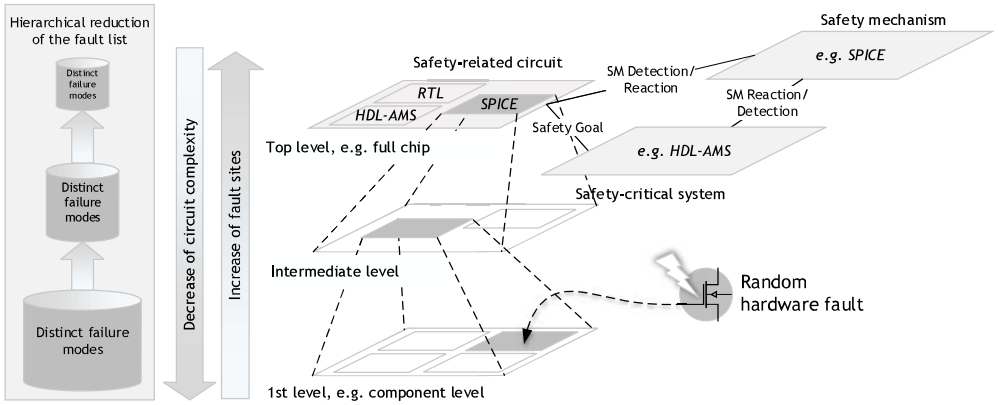


Figure 5.1: The number of distinguishable failure modes decreases with increasing level of design hierarchy due to functional fault equivalence.

5.1 Functional fault equivalence: a review

In a hierarchical design architecture, the circuit complexity and the number of fault sites increases with increasing hierarchy level, see fig. 5.1. That is, the number of faults to be simulated at the top level is greater than at any other lower level. However, the number of distinguishable failure modes decreases with increasing hierarchy level. This is for example due to the fact that some device failures cause a similar component failure. This becomes obvious for example for components with high-impedance inputs like for example an inverter circuit. Whatever the input signal is, the output is either set to the high or low level. That is, it propagates the input error either not or by inverting the expected output level. Moreover, soft failures may not propagate upwards to the top level. Consequently, no failure but instead the nominal circuit response may be perceived at the top level. Additionally, two or more in series or in parallel connected components may cause similar and non-distinguishable failure modes at the top level. This circumstance is referred to in the literature by *functional fault equivalence* [141, 75].

5.1.1 Hierarchical fault grouping

In fig. 5.2, the schematic of a gate-driver circuit is shown. The components *C1* and *C2* a level shifter and an output stage connected in series. At a higher level, the component *C3* is comprised of *C1* and *C2* and is instantiated inside

the component *HT* together with a safety mechanism for the gate-driver output and a level shifter in a feedback loop. The digital logic processes the output of the safety mechanism and controls the gate driver circuit, respectively.

One or more faults inside *C1* may cause a stuck-at high failure mode of *C1*. These faults are called functional equivalent and can be collected in one *fault group* $g_{C_1,2}$ due to the similarity of the output response (failure mode) of C1 in the presence of these faults. In the same manner, one or more faults inside *C2* may cause a stuck-at high failure mode of *C2* and can be collected in another fault group $g_{C_2,1}$. Eventually, all faults inside *C1* and *C2* which cause a stuck-at high, respectively, will propagate to the same failure mode of *C3*. Thus, at the level of *C3*, both fault groups $g_{C_1,2}$ and $g_{C_2,1}$ can be merged into one group. Additionally, a group of faults $g_{C_1,3}$ is dissimilar from the faults in $g_{C_1,2}$ at the level of *C1* but cause the same failure at the level of *C3*. Hence, $g_{C_1,2}$ can be merged together with $g_{C_1,2}$ and $g_{C_2,1}$.

In fig. 5.2, five fault groups are available at the level of *C3*. However, four possible failure modes including the nominal circuit response are listed in the table for *HT*. A reason for this can be, that the safety mechanism reads a critical value for signal $V_{C_3}(t)$ and forces the system to interrupt the gate-driver process.

5.1.2 Representative faults

Due to functional fault equivalence, the total population of faults F in a component accounts for a limited number of distinguishable failure modes $K$for which generally $F \geq K$. Equivalent faults can be divided into fault groups (sub-populations), which must be identified. Eventually, one of the faults within a fault group can be picked to represent the other equivalent faults by a certain accuracy regarding its representativeness. This fault is called the representative fault. Thus, for the total population of faults the number of fault groups and representative faults is equal to $K-1$ if at least one fault within the population causes a negligible failure. However, this is generally the case if the fault-free circuit-response is part of the population.

5.1.3 Functional fault collapsing rate

The total amount of faults F in the component can be reduced to $K-1$ representative faults, with the collapsing rate [142]

$$CR = 1 - \frac{K-1}{F}. \tag{5.1}$$

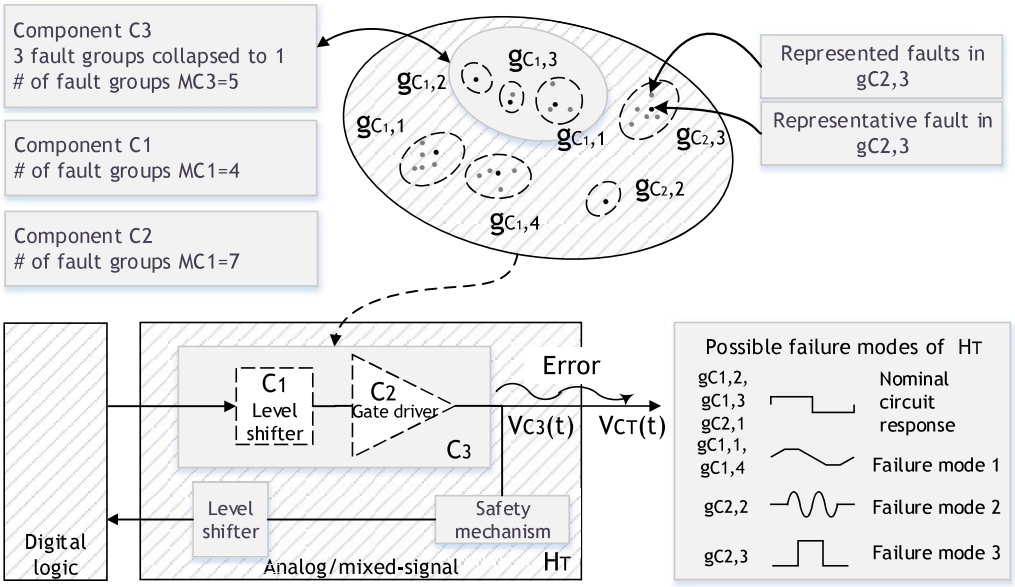


Figure 5.2: Gate-driver example to illustrate hierarchical fault grouping in analog and mixed-signal circuits.

If K is chosen too small, fewer representative faults are available and less top-level fault simulations are required. However, the risk of inaccurate representation of the fault groups by the representative faults rises. This is due to the fact that within-dispersions (errors) are high because groups contain faults which are too dissimilar to be grouped. On the other hand, choosing K too high, will improve the representativeness but at the cost of more top-level fault simulations. Therefore, it is important to decide for an optimal K which balances both, representativeness and the number of faults to simulate at the top level.

5.2 Implementation overview

To guide through the next sections, a short overview regarding the implementation of the hierarchical fault injection is given.

1. **Component-level fault simulation** To identify the failure modes of a component, component-level fault simulations are run. For this purpose, a component-level test bench is required comprising adequate

stimuli and loads.

2. **Hierarchical clustering algorithm** Based on the similarity of the circuit responses (failure modes) at the component level, equivalent faults can be identified. As criterion for the similarity (or dissimilarity), a distance measure between the output responses of the component is used. A distance-based grouping of equivalent faults is implemented by a hierarchical clustering algorithm.

3. **Optimal fault groups** The optimal number of fault groups is determined by using internal cluster validation indices.

4. **Representative faults** From the optimal number of fault groups, one fault from each group is chosen to represent the other equivalent faults based on a criterion.

5.3 Component-level simulation test bench

Circuit-level and gate-level fault models are chosen for fault simulation at the component level. A test-bench is prepared for the multi-input and multi-output component including adequate stimuli and loads, see fig. 5.3. Fault equivalence is generally determined with respect to the input stimulation. The component must be stimulated for the input range for which the component exercises its nominal behaviour in the test cases. Alternatively, a randomized input range to cover possible input stimulation outside the specification of the input range can be considered. However, this may reduce the fault collapsing rate because more failure modes may be triggered depending on the stimuli diversity. However, if the failure of other components are considered, the component must be stimulated with respect to the full range of variation it may experience, if other components in the circuit fail.

From component-level fault simulations, waveform data of the output responses are obtained and stored. The waveform data must be re-sampled in order to obtain equidistant time-value parts of unit size. The post-processed waveforms corresponding to the faults $f = 1, ..., F$, the s^{th} stimulus and the o^{th} output are stored in column vectors $\vec{w}_{o,s,f}$. The complete data set is composed of multi-modal waveform data with respect to stimulus $s = 1, ..., S$ and output $o = 1, ..., O$ and is shown in tab. 5.1.

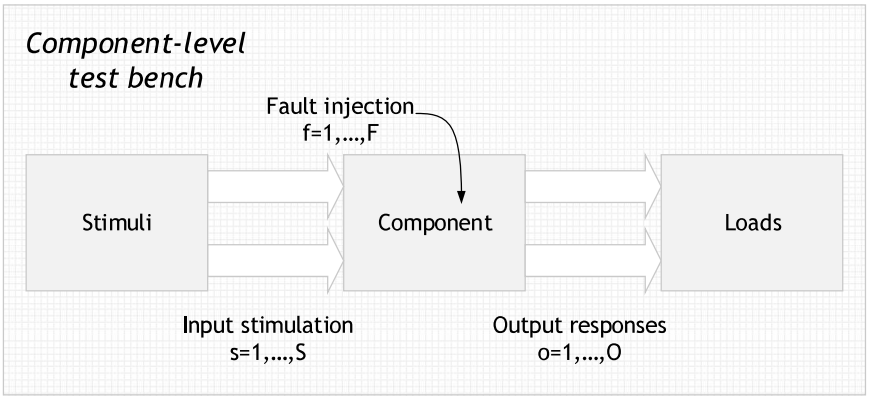


Figure 5.3: Schematic test bench for the multi-input, multi-output component.

	Output 1			... Output o ...			Output O
Fault no.	Stimuli 1	...	Stimuli S	... Stimuli s ...	Stimuli 1	...	Stimuli S
Fault 1	$\vec{w}_{1,1,1}$	...	$\vec{w}_{1,S,1}$	... $\vec{w}_{o,s,1}$ ...	$\vec{w}_{O,1,1}$	...	$\vec{w}_{O,S,1}$
...	...	...	...	...	...	...	...
Fault f	$\vec{w}_{1,1,f}$	...	$\vec{w}_{1,S,f}$	... $\vec{w}_{o,s,f}$ ...	$\vec{w}_{O,1,f}$	...	$\vec{w}_{O,S,f}$
...	...	...	...	...	...	...	...
Fault F	$\vec{w}_{1,1,F}$	...	$\vec{w}_{1,S,F}$	... $\vec{w}_{o,s,F}$ ...	$\vec{w}_{O,1,F}$	...	$\vec{w}_{O,S,F}$

Table 5.1: Data-set of vectors $\vec{w}_{o,s,f}$ comprising component output responses.

5.4 Hierarchical clustering algorithm

Hierarchical agglomerative clustering is a method for dividing data into natural groupings [144]. Generally, the data is passed to the hierarchical clustering (HC) algorithm in form of a two-dimensional coordinate matrix composed of objects×features. The main steps of the HC algorithm are [168]

1. **Distance matrix** Calculation of a square distance matrix comprising the pairwise dissimilarities (or similarities) among all objects

2. **Linkage function** Generation of a hierarchically nested cluster tree based on a linkage function for grouping objects using the distance matrix

3. **Optimal number of clusters** Determine where to cut the cluster tree based on a criterion for the optimal number of clusters

The HC algorithm produces $F-1$ levels of hierarchically nested clusters, where F is the number of objects. The cluster tree can be displayed in a dendrogram which provides a complete graphical description of the clustering [168, 144].

5.4.1 Data set preparation

The decision about fault equivalence depends on the accurate calculation of the dissimilarities among the output responses. To fully cover the variability of time and value-continuous output responses, the vectors $\vec{w}_{o,s,f}$ obtained from waveform data are processed for dissimilarity calculation. To analyse the data set by the HC algorithm, first a square distance matrix must be calculated which comprises the pair-wise dissimilarities among the vectors $\vec{w}_{o,s,f}$. HC algorithms are typically not designed to process multi-modal waveform data. In order to facilitate this, the multi-modal waveform must be prepared for the HC algorithm.

Euclidean distance metric

The dissimilarity among the output responses respective to faults f and f' is quantified by the multivariate euclidean pair-wise distances for the o^{th} output and s^{th} stimuli in vector notation [144]

$$d^{o,s}_{f,f'} = \sqrt{(\vec{w}_{o,s,f} - \vec{w}_{o,s,f'})(\vec{w}_{o,s,f} - \vec{w}_{o,s,f'})^{\mathrm{T}}}. \tag{5.2}$$

It is zero for $f = f'$. That is, faults f and f' produce identical output responses for any stimulation s.

For $f, f' = 1, ..., F$, a square distance matrix $\mathbf{D}_{o,s} = (d^{o,s}_{f,f'})$ is built. This is exercised for all output-stimuli pair data sets, resulting in a set of $O \cdot S$ distance matrices.

Combined distance matrix

Squared euclidean distances are generally additive. Therefore, without loss of information they can be linearly combined to one distance matrix

$$\mathbf{D} = \frac{1}{O \cdot S} \sqrt{\sum_{o=1,s=1}^{O,S} \mathbf{D}^2_{o,s}} = (d_{f,f'}). \tag{5.3}$$

It is composed of euclidean elements $d_{f,f'}$ which can be multiplied by a scaling factor $\frac{1}{O \cdot S}$ to match the distance range of $\mathbf{D}$ to $\mathbf{D}_{o,s}$. In this notation

all distance matrices $\mathbf{D}_{o,s}$ contribute equally to the combined distance matrix. The combined distance matrix can be directly used as input to the HC algorithm, see list. B.1 and B.2.

Principal Coordinates Analysis

Data sets in form of a coordinate matrix are generally more convenient to analyse than multi-modal waveform data. Although the HC algorithm processes the distance matrix, linkage functions and cluster validation methods [169] generally process two-dimensional coordinate matrices instead of distance matrices. Moreover, some clustering algorithms (e.g. K-Means [170]) only process coordinate matrices. In order to use the multi-modal waveform data in other algorithms, it must be transformed into a two-dimensional coordinate matrix.

Given a euclidean distance matrix, a coordinate matrix can be derived by applying Principal Coordinates Analysis (PCoA) to the distance matrix [171]. PCoA is a dimensionality reduction technique which is based on eigenvalue decomposition and orthogonal projection of the original data into a lower-dimensional coordinate matrix, composed of new orthogonal features. By applying PCoA to the combined distance matrix D, the resulting coordinate matrix is equivalent to the multi-modal waveform data set in terms of the pair-wise distances among the output responses. For this purpose, the squared distance matrix is first processed by double centring using the centring matrix $\mathbf{J} = \mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}'$

$$\mathbf{B} = -\frac{1}{2}\mathbf{J}\mathbf{D}^2\mathbf{J}, \tag{5.4}$$

where $n = F$. The double centred distance matrix $\mathbf{B}$ is subsequently processed in terms of eigenvalue decomposition

$$\mathbf{B} = \mathbf{E}\boldsymbol{\Lambda}\mathbf{E}'. \tag{5.5}$$

The eigenvalues in Λ and orthogonal eigenvectors in E are ordered by decreasing eigenvalue. The m largest positive eigenvalues $\lambda_1, ..., \lambda_m$ and corresponding eigenvectors $e_1, ..., e_m$ are subsequently used to calculate the coordinate matrix $\mathbf{X}$

$$\mathbf{X} = \mathbf{E}_m\boldsymbol{\Lambda}_m^{1/2}. \tag{5.6}$$

The coordinate matrix represents the multi-modal waveform data in a two-dimensional space. It can be used for cluster validation and other algorithms which require a coordinate matrix as input.

5.4.2 Optimal set of fault groups

The optimal number of clusters K_{opt}, that is fault groups, can be determined for example by visually inspecting the

- dendrogram of the cluster tree produced by the HC algorithm, or
- scatter plot of the two-dimensional coordinate matrix.

However, to automatize the determination of K_{opt}, various cluster validation methods are proposed in the literature on cluster analysis [169]. They can be roughly distinguished into internal and external methods. Internal cluster validation is reported to outperform external cluster validation in determining adequate K_{opt} [145]. Additionally, opposed to external methods, internal cluster validation is advantageously based on the information inherent to the data set and thus requires no further information as reference.

Besides the cluster validation method, the linkage function of the HC algorithm contributes strongly in reducing the dissimilarities of the objects within each cluster in the hierarchical clustering scheme. Therefore, an adequate linkage function is determined.

Selection of a linkage function

A HC algorithm with the linkage functions listed in tab. 5.2 are used in this work [6]. Generally, different linkage functions may result in different sets of hierarchical clusters. They differ with respect to the similarity of the objects linked to one cluster at a subsequent level. The within-cluster sum-of-squared errors $SSE(K)$ is a measure of within cluster compactness, see list. B.3. It quantifies the dissimilarity of the objects within the same cluster. It is defined as

$$SSE(K) = \sum_{k=1}^{K} \sum_{f,f' \in C_k, f<f'} d_{f,f'}^2, \tag{5.7}$$

where C_k is the k^{th} cluster for $k = 1, ..., K$.

To achieve high representativeness of the representative faults within each fault group, high within-cluster similarity at any level of the hierarchical cluster tree is addressed. Thus, the linkage which produces the smallest SSE at any level from $k = 1$ to $k = K$ is preferred. This is generally given by the Ward linkage, which seeks to minimize the total SSE for any K. However, other linkages can be considered and ranked by equation 5.7 based on their cumulative SSE.

Linkage	Description	Function
Single	Smallest distance between objects in two clusters (nearest neighbour)	$d(r,s) = \min(dist(x_{ri}, x_{sj}))$, $i \in (1,...,n_r)$, $j \in (1,...,n_s)$
Complete	Largest distance (furthest neighbour)	$d(r,s) = \max(dist(x_{ri}, x_{sj}))$, $i \in (1,...,n_r)$, $j \in (1,...,n_s)$
Average	Average distance between all pairs of objects in two cluster	$d(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} dist(x_{ri}, x_{sj})$
Centroid	Euclidean distance between centroids of two clusters	$d(r,s) = \|\overline{x}_r - \overline{x}_s\|_2$, $\overline{x}_r = \frac{1}{n_r} \sum_{i=1}^{n_r} x_{ri}$
Median	Euclidean distance between weighted centroids	$d(r,s) = \|\widetilde{x}_r - \widetilde{x}_s\|_2$, $\widetilde{x}_r = \frac{1}{2}(\widetilde{x}_p + \widetilde{x}_q)$
Ward	Minimum variance when joining objects in two clusters into one cluster	$d(r,s) = \sqrt{\frac{2n_r n_s}{(n_r+n_s)}} \|\overline{x}_r - \overline{x}_s\|_2$,
Weighted	Recursive distance between two clusters	$d(r,s) = \frac{(d(p,s)+d(q,s))}{2}$

Table 5.2: Linkage functions for the hierarchical clustering algorithm [6]. Cluster r is induced from clusters p and q, n_r the number of objects in cluster r and x_{ri} is the i^{th} object in cluster r.

Internal cluster validation methods

For internal cluster validation, three different methods are used, see tab. 5.3. Generally, the functions are evaluated for $K = 1,...,F$. The optimal number of clusters $K = K_{opt}$ is determined for the global maximum or global minimum of the respective function, see list. B.4, B.5 and B.6. All three methods are used simultaneously. If the methods suggest different K_{opt}, then the candidate satisfying the practical meaningfulness of the resulting fault groups is chosen.

Method	Criterion	Function
Davies	Min	$DB(K) = \frac{1}{K}\sum_{k=1}^{K} \max_{k \neq l} \{D_{k,l}\}$,
Bouldin		$D_{k,l} = \frac{(\overline{d}_k + \overline{d}_l)}{d_{k,l}}$
Silhouette	Max	$Sil(K) = \frac{1}{K}\sum_{k=1}^{K} \frac{1}{n_{C_k}} \sum_{f \in C_k} \frac{b_f - a_f}{\max(a_f, b_f)}$
		$a_f = \frac{1}{n_{C_k}-1} \sum_{y \in C_k, y \neq x} d(f, f')$
		$b_f = \min_{l, l \neq k}(\frac{1}{n_{C_k}} \sum_{y \in C_l} d(f, f'))$
Dunn	Max	$D(K) = \min_k(\min_l(\frac{\min_{f \in C_k, f' \in C_l} d(f,f')}{\max_m(max_{f,f' \in C_m} d(f,f'))}))$

Table 5.3: Cluster validation methods [7, 8, 9] used for the determination of final set of fault groups K_{opt}.

Davies Bouldin The Davies Bouldin function is based the within-cluster between-cluster distance term $D_{k,l}$, where $\overline{d}_k$ is the average distance between each point in C_k to the cluster centroid and $\overline{d}_l$, respectively [7]. The Davies Bouldin function is minimum 0, indicating the optimal number of clusters when for within-cluster similarity is high and between-cluster distance similarity is low.

Silhouette The Silhouette function compares the average distance a_i of one point f to the other points f' in its own cluster C_k with its average distance b_i to points f' in other clusters C_l [8]. The Silhouette function is maximum 1 when all points are well-matched in their respective cluster, indicating the optimal number of clusters.

Dunn The Dunn function, aims to balance intra-cluster compactness and intra-cluster separateness by minimizing the within-cluster sum of squares and maximizing the between-cluster sum of squares [9]. The global maximum of the Dunn function indicates the optimal number of clusters.

5.5 Choice of representative faults

Representative faults are determined using two criteria, the medoid and worst-case. In a cluster, the medoid [144] is the object closest to the centre of the cluster (centroid). The worst-case object in a cluster is determined for the object which is most dissimilar to a reference object. In this context, the worst-case fault in a fault group is the one which is most dissimilar to

the nominal fault-free circuit response. The implementation of both criteria is shown in list. B.8.

5.5.1 Medoid criterion

In terms of cluster analysis, the best representative object of a cluster is the medoid [172]. The medoid is closest object to the cluster centroid and its average dissimilarity $\overline{d}_{f \in C_k}$ to all other objects in the same cluster is minimal. The medoid is defined as

$$\min_{f \in C_k}\left(\sum_{f' \in C_k} d_{f,f'}\right), \tag{5.8}$$

where M_k is the number of faults in the k^{th} cluster C_k.

5.5.2 Worst-case criterion

The worst-case object f in cluster C_k is defined as

$$\max_{f \in C_k}(d_{f,f' \equiv \text{nominal}}), \tag{5.9}$$

where $f' \equiv$ nominal is the nominal circuit response and must be an object of the coordinate matrix $\mathbf{X}$.

If the within-cluster sum-of-squared errors is large for C_k (e.g. because of a high collapsing rate CR), the worst-case object can be chosen as a representative fault. This way, the missing out of the most severe fault in a fault group is avoided. Here, the most severe fault in a fault group is considered the fault which is most dissimilar to the nominal circuit response.

5.6 Summary

In order to increase the verification confidence but reduce top-level fault simulation runs, a hierarchical fault injection technique is proposed which is based on functional fault equivalence and fault grouping.

Functional fault equivalence is determined for a component with respect to its output signals and its stimulation by component-level fault injection. Subsequently, the component output responses in form of waveforms are processed by a hierarchical clustering algorithm. The optimal number of fault groups is determined by internal cluster validation. One fault from each fault group is determined based on a criterion to represent all other equivalent faults in the same fault group. This fault is called the representative fault. For top-level verification, fault simulation is only executed with the representative faults.

Chapter 6

Evaluation of soft faults by global sensitivity analysis

Faults can generally be classified into soft and hard faults, based on the severity of their effects in the circuit [102]. The soft faults considered in this chapter are parametric degradation faults of devices [112, 39] and signal path defects [94, 95, 107].

The work presented addresses the reduction of soft faults from the fault list based on the significance of their effects on the circuit response at component level.

6.1 Modelling soft faults

Parametric degradation faults of devices are most frequently modelled with a symmetric distribution of the parameter value for example a Gaussian distribution [112, 39]. The mean of the distribution μ is set to be the nominal parameter value. The out-of-specification limits are determined to be a multiple of the distribution variance σ.

Signal path defects such as opens and bridging faults are frequently modelled by resistive fault models [107, 116]. However, for resistive fault models, symmetric distribution cannot be considered. For example the nominal (i.e. fault-free) resistance value of the open-circuit fault model is zero. In the case of a soft open-circuit, the value can be between zero and a value after which it becomes a hard open-circuit fault. Generally, the resistance value of the open-circuit fault can vary from less than 100kΩ to several GΩ and the resistance value of a short-circuit fault can reach 20kΩ [115, 116, 173]. The resistance range for which the resistive fault model causes a soft failure can for example be determined by simulation with varying resistance values.

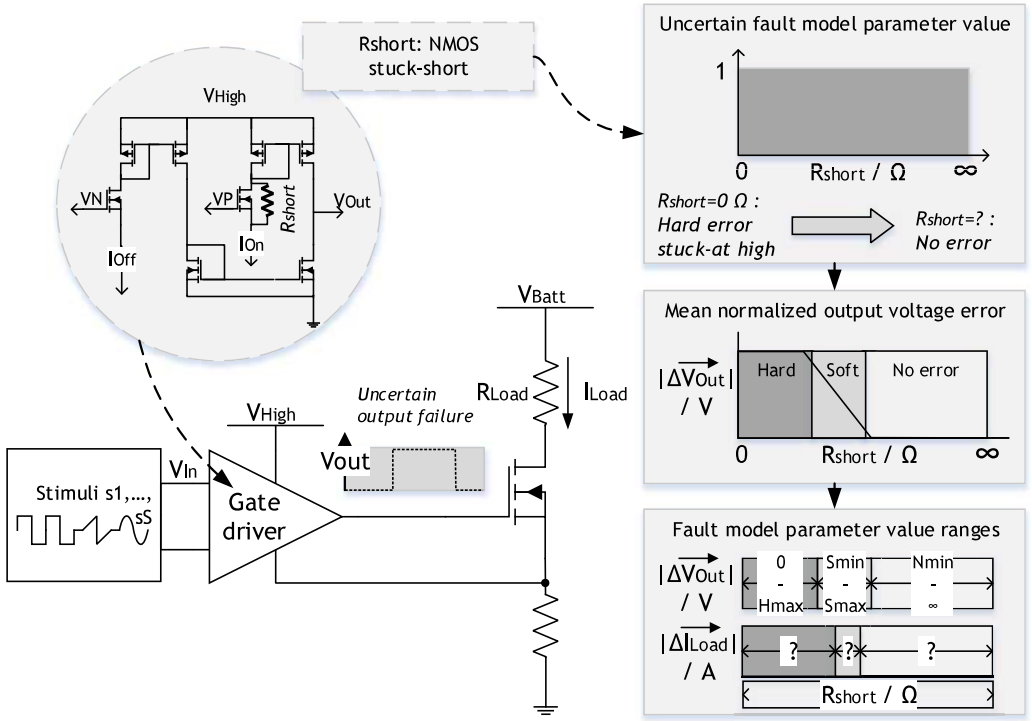


Figure 6.1: Component failure with respect to a resistive short-circuit fault model.

6.2 Component failure and fault model parameter ranges

From a general point of view, parametric degradation faults and signal path defects can cause three types of failures of the circuit output: hard failure, soft failure and no failure. This is illustrated in fig. 6.1 for a low-side gate driver circuit composed of MOSFETs. An NMOS in the gate driver experiences a stuck-short failure. It is modelled with a resistive fault model R_{short}. The fault model is analysed with respect to its effect on the current I_{Load} in the application. The gate driver is a component with primary output signal V_{Out} and input signal V_{In}. In the illustration, it is assumed that an ideal short circuit with $R_{short} = 0\Omega$ will cause a hard failure, e.g. a stuck-at high. This will cause the Power MOSFET (n-type) in the application to be stuck-on, hence $I_{Load} > 0$. However, it is initially uncertain at which R_{short} value, I_{Load}

will only experience a soft failure or no failure, at all.

6.2.1 Determination of soft failure ranges

Generally, the resistance value, at which I_{Load} experiences a soft failure depends on the gate driver circuit and the application. However, if V_{Out} experiences no failure, I_{Load} will not, either. If V_{Out} experiences a hard failure, I_{Load} will either. If V_{Out} experiences a soft failure, I_{Load} may either experience a hard failure, a soft failure or no failure. Thus, to evaluate the soft fault in the application, it is sufficient to initially identify the resistance range for a soft failure at component level, i.e. $S_{min} < R_{short} < S_{max}$.

The failure and parameter ranges for each fault $f = (f_1, f_2, ..., f_{N_F})$ are determined in a test bench exclusively for the component in which fault simulation is exercised with varying parameter values $p = (p_1, p_2, ..., p_{N_p})$. The error of the primary output in the presence of a fault may generally depend on the input stimulation. Thus, the test bench comprises a set of stimuli $s = (s_1, s_2, ..., s_{N_S})$.

6.2.2 Mean normalized output error calculation

With respect to fault f, parameter p and stimulus s, the mean normalized error of the primary output amplitude can be calculated by

$$\left|\Delta\overline{Vout}_p^{(f)}\right| = \frac{1}{N_S}\sum_{s=s_1}^{s_{N_s}}\left|Vout_s^{(nominal)} - Vout_{s,p}^{(f)}\right|. \tag{6.1}$$

The voltage $VOut_s^{(nominal)}$ is the primary output response for the given stimulus s in the presence of no fault (nominal). If the primary output error is processed as a time-dependent waveform, the euclidean distance in vector notation can be used for error calculation by

$$\left|\Delta\overline{Vout}_p^{(f)}\right| = \frac{1}{N_S}\sum_{s=s_1}^{s_{N_s}}\sqrt{(\vec{Vout}_s^{(nominal)} - \vec{Vout}_{s,p}^{(f)})(\vec{Vout}_s^{(nominal)} - \vec{Vout}_{s,p}^{(f)})^{\mathrm{T}}} \tag{6.2}$$

The vector notation indicates column vectors composed of samples of the output response. This is advantageous, if the time at which a fault will cause an error of the output is not known, that is the sampling time not defined.

The mean normalized error is plotted over p from which the hard, soft and no error ranges are extracted, see middle box in fig. 6.1. If the component has more than one output signals, the parameter value ranges are extracted for each of them individually.

6.3 Global sensitivity analysis

Sensitivity analysis is a method to investigate the contribution of the variability of a number of inputs X on the variability of the output Y in a model $Y = f(\mathrm{X})$. The inputs are given in a $n \times k$ input matrix $\mathbf{X} = (X_1, X_2, ..., X_k)$, where k is the number of inputs and n is the sample size.

Sensitivity analyses are roughly divided into local [144] and global methods [174]. In a local sensitivity analysis, the input variability is limited to a close proximity around a specific point $\overline{X}$. Moreover, only one input can be varied at a time. This limitations are not given in a global sensitivity analysis. Here,the full input feasibility space is accounted for. The contribution of the inputs X on the output Y are quantified by sensitivity indices, also called *effects*), which range from 0 to 1. Sensitivity indices close to 0 indicate non-influential inputs and higher values indicate more influential inputs. Effect estimates obtained from global sensitivity analysis (GSA) are typically used to [175]

- rank inputs by their relative contribution on the output (factor prioritization [176]) and
- identify inputs which are non-influential on the output (screening [177, 178] or factor fixing [174]).

6.3.1 Qualitative methods

For screening, a qualitative GSA method is typically used which is only capable of identifying non-influential inputs. The effect estimates are quantitative but they cannot be used to rank or compare the effects of different inputs with each-other. A frequently used qualitative GSA method is the elementary effects method [179].

6.3.2 Quantitative methods

Quantitative GSA provides effect estimates of relative contributions for each input on the output. It can be used for both, screening and factor prioritization. Quantitative GSA is usually more computationally demanding than qualitative GSA. The most prominent quantitative GSA methods are for example based on

- variance decomposition [180, 181],
- density estimation [175], and

- sampling-based method under the linear model assumption [182, 183, 184].

Correlation analysis analysis and regression analysis are frequently used sampling-based methods [184]. Sampling-based methods are based on the linearity assumption made on the model $f(\mathrm{X})$. Thus, the linearity hypothesis must be confirmed in order to prove the validity of the analysis. The variance-based and density-based GSA are alternative methods which are not restricted to any assumptions made on the model $f(\mathrm{X})$. However, these methods are typically more computationally demanding than the sampling-based methods.

6.4 Evaluation of soft faults

The soft faults are evaluated in terms of global sensitivity analysis. The sampling-based methods are applied in order to rank soft faults based on their effects on the component output response. In order to use the sampling-based method, the following topics are discussed in the remainder of this section:

- **Linear model assumption** In order to use a linear model for GSA, the linearity of the function $f(\mathrm{X})$ must be confirmed statistically.
- **Screening** A parameter screening is utilized in order to eliminate non-influential soft faults (i.e. with negligible effects) from the fault list.
- **Effect estimates** Effect estimates which can be used in the sampling-based method.
- **Non-parametric re-sampling** A sampling technique which is used to calculate the confidence intervals and statistical significance of the effect estimates.

Finally, a screening method is discussed which is used to identify non-influential soft faults. This analysis is used to reduce the number of inputs prior to the sampling-based method.

6.4.1 Linear model assumption

For a given fault list comprised of k soft (parametric) fault models and their corresponding parameter ranges, each fault is evaluated in terms of its contribution to the explained variance of the circuit response. The explained variance is the proportion of the total variance of the output variable, to

which a mathematical model of the circuit response accounts for. Complementary with the unexplained variance, it sums up to the total variance $\mathrm{V}(Y)$.

Coefficient of determination

The proportion of total variance to which the mathematical model accounts for is quantified by the coefficient of determination, denoted R^2, and is defined as [144]

$$R^2 = 1 - \frac{\text{Sum of squares of residuals}}{\text{Total sum of squares}} = 1 - \frac{\sum_i^k (Y_i - \widehat{Y}_i)^2}{\sum_i^k (Y_i - \overline{Y})^2}, \tag{6.3}$$

where Y_i is the i^{th} sample of the output variable, $\overline{Y}$ is the sample mean and $\widehat{Y}_i$ is the predicted output variable (circuit response) by the mathematical model.

Multiple linear regression model

A linear multiple regression model is assumed to predict the output variable in the soft failure range. The faults are represented by input variables in a $n \times k$ input matrix $\mathbf{X} = X_1, X_2, ..., X_k$ which are regressed on the output variable Y by [144]

$$Y = a + \sum_{j=1}^{k} b_j X_j + \epsilon, \tag{6.4}$$

where a is the intercept, b_j are the regression coefficients for $j = 1, ..., k$ input variables and ϵ is the residual of Y and estimated $\widehat{Y}$.

Linearity hypothesis

The linear regression model is a valid predictor for Y, if the linearity hypothesis can be confirmed statistically by providing evidence on [182, 183, 184]

- a normal distributed $\epsilon = Y - \widehat{Y}$ with $\overline{\epsilon} = 0$ and
- a significant R^2 close to 1.

In the presence of uncorrelated input variables, the sum of all total effects is equal to R^2 [182]. Thus, it indicates how much variability is omitted in the output variable, if one or more input variables are eliminated from the regression model. Consequently, the number of input variables can be reduced by keeping a scalable amount of variability, e.g. 90%, in the output variable.

Simulation fault coverage

Under the linearity assumption, a measure for the simulation fault coverage is derived. It is based on the cumulative contribution of each fault model parameter X_i to the coefficient of determination R^2. First, the total effect estimates are sorted in decreasing order from $X_i = X_1$ to X_k. The simulation fault coverage FC for keeping the first m faults is calculated by

$$FC(m) = \frac{1}{R^2}\left(1 - \frac{\sum_i^m (Y_i - \widehat{Y}_i)^2}{\sum_i^m (Y_i - \overline{Y})^2}\right). \tag{6.5}$$

It is 1, if $m = k$ and decreases for decreasing m. It is used to determine how much variability is omitted in the output variable, if one or more soft faults are eliminated from the soft fault list.

6.4.2 Screening

Prior to the sampling-based approach it is desirable to eliminate non-influential input variables in a computationally efficient manner. For this purpose, screening using the elementary effects method is applied.

Overview

Using the extended elementary effects method [185], inputs can be ranked and non-influential inputs can be identified. The range of the input parameters are first subdivided into a discrete number of levels p. Only one parameter value is changed per simulation run from one level to another by $\pm\Delta$. This One-At-A-Time (OAT) procedure is defined in sampling matrix. Simulations are exercised in accordance to the sampling matrix for the total amount of inputs k and for a number of samples r. Based on the simulated response of the primary output, three sensitivity measures can be calculated according to [179]:

- μ: estimate of the overall linear and additive effect of a fault model on the output
- σ: estimate of the non-linear effect and interactions

In [177], the elementary effects method is extended by an additional measure $\mu*$ which is used to rank the fault models based on their effect on the output.

Elementary effects

Given the sampling matrix which defined for the k-dimensional input vector $\mathbf{X}$, each element X_i can assume integer values in the set $\left\{0, \frac{1}{p-1}, \frac{2}{p-1}, ..., 1\right\}$. Thus, the experimentation region, Ω, is a k-dimensional, p-level grid. The elementary effect of the i^{th} input for a realization $\mathbf{x}$ of $\mathbf{X}$, is defined as [179]

$$d_i(\mathbf{x}) = \frac{y(x_1, ..., x_{i-1}, x_i + \Delta, x_{i+1}, ..., x_k) - y(\mathbf{x})}{\Delta}, \tag{6.6}$$

where x_i is varied by $\Delta = 1/(p-1)$. To output obtained for an input $\mathbf{x}$ is $y(\mathbf{x})$.

Sensitivity measures

The sensitivity measures are calculated from the mean and variance of the elementary effects [179]

$$\mu = \sum_{i=1}^{r} \frac{d_i}{r} \tag{6.7}$$

and

$$\sigma = \sqrt{\sum_{i=1}^{r} \frac{(d_i - \mu)^2}{r}}, \tag{6.8}$$

respectively. The measure which is used to rank the fault models based on their importance, is calculated by [177]

$$\mu* = \sum_{i=1}^{r} \frac{|d_i|}{r}. \tag{6.9}$$

In the literature, μ is not considered to be suited to rank the inputs based on their importance, as elementary effects with opposed signs may cancel each other out. However, with $\mu*$ this is avoided by calculating the mean of the absolute values of elementary effects.

Sampling matrices

The experimental plan is based on r sampling matrices which must be designed in order to calculate the elementary effects. A sampling matrix $\mathbf{B}*$ with dimension $(k+1) \times k$ is composed of randomized elements. One sampling matrix $\mathbf{B}*$ provides one elementary effect per input and is defined as [174]

$$\mathbf{B}^* = (\mathbf{J}_{k+1,1}\mathbf{x}^* + (\Delta/2)[(2\mathbf{B} - \mathbf{J}_{k+1,k})\mathbf{D}^* + \mathbf{J}_{k+1,k}])\mathbf{P}^*, \tag{6.10}$$

where $\mathbf{J}_{k+1,k}$ is a $(k+1) \times k$ matrix of ones, $\mathbf{x}^*$ is a randomized base value of $\mathbf{X}$ to start the OAT procedure for the r^{th} sampling matrix. The diagonal matrix $\mathbf{D}^*$ is composed of elements which are either $+1$ or -1 with equal probability and $\mathbf{B}$ is a $(k+1) \times k$ matrix which is a lower triangular matrix of 1s. $\mathbf{P}^*$ is a $k \times k$ matrix in which each column is composed of 0s and a single 1 where no to columns have a 1 in the same line (here realized by a diagonal matrix).

The lines of $\mathbf{B}^*$ are composed of surrogate values $\left\{0, \frac{1}{p-1}, \frac{2}{p-1}, ..., 1\right\}$. The surrogate values must be mapped to the corresponding values at the respective levels defined for the fault model parameters $p_1, p_2, ..., p_P$, where $P = k$. This mapping transforms $\mathbf{B}^*$ into **B'** which can be passed to the fault simulation via fault injection scripts to assign the values in **B'** to the respective fault model parameters. Fault simulations are exercised accordingly.

6.4.3 Sampling-based effect estimates

The effect estimates are presented for the direct and total effects. Direct effects quantify the contribution of each input to the output variable and do not account for the variability of the other effects. Total effects take account of the variability of other effects.

Generally, total effects are used for determining non-influential inputs and ranking. They comprise among the direct effects of inputs, any effects due to the shared variances of the inputs [174]. However, the direct effect can generally be used complementary with the total effect in order to identify suppressor variables among the inputs [182]. A suppressor is a variable which shares small variance with the output but contributes to the shared variance by removing variance from other inputs.

Zero-order correlation

The zero-order correlation, that is Pearson correlation coefficient, quantifies the shared variance between a single input X_i and the output Y [186]. It is defined as

$$\rho(X_i, Y) = \frac{\text{Cov}(X_i, Y)}{\sqrt{\text{Var}(X_i)\text{Var}(Y)}} \tag{6.11}$$

where $\text{Cov}(X_i, Y)$ is the covariance of X_i and Y. $\text{Var}(X_i)$ and $\text{Var}(Y)$ are the variances of the input and output, respectively. The implementation is shown in list. C.2.

Beta weights

The beta weights are a measure of total effect of a variable X_i on the output Y. Beta weights are based on linear multiple regression (see equation 6.4) and are calculated by

$$\beta_i = b_i \sqrt{\frac{\mathrm{Var}(X_i)}{\mathrm{Var}(Y)}} \tag{6.12}$$

Therein, b_i is the regression coefficients for $i = 1, ..., k$ input variables. For uncorrelated inputs, the beta weights range between -1 and $+1$. Total effect estimates are obtained by squaring the beta weights [182]. The implementation is shown in list. C.3.

Relative weights

Opposed to beta weights, the relative weights [187] are more accurate in partitioning the shared variance among inputs and are particularly suited in the presence of multi-collinearity and a high number of inputs (more than 10) [182]. It is based on calculation of an orthogonal set of new variables $\mathbf{Z}$ which are maximally correlated to $\mathbf{X}$ by its singular value decomposition into singular values $\sigma_1, ..., \sigma_m = \mathrm{diag}(\Sigma)$ as well as left and right eigenvectors $\mathbf{U}$ and $\mathbf{V}$

$$\mathbf{X} = \mathbf{U\Sigma V'}. \tag{6.13}$$

Subsequently, the output vector $\mathbf{y}$ and input matrix $\mathbf{X}$ are regressed to the orthogonal variables $\mathbf{Z}$ by

$$\mathbf{B} = (\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{y} \tag{6.14}$$

and

$$\mathbf{\Lambda} = \mathbf{(Z'Z)^{-1}Z'X} \tag{6.15}$$

respectively. Finally, the relative weights ϵ are obtained by

$$\boldsymbol{\epsilon} = \mathbf{\Lambda^2 B^2}. \tag{6.16}$$

The implementation is shown in list C.1.

6.4.4 Sampling approach

Randomized fault simulations are run to calculate the effect estimates based on the circuit responses (mean normalized error, see equation 6.2). Thereby, all fault model parameters are varied simultaneously within their feasibility space. Afterwards, non-parametric re-sampling is applied to calculate confidence intervals for the effect estimates. Subsequently, the statistical significance of the effect estimates are obtained with a hypothesis test.

Non-parametric resampling

For computation-intensive models, exhaustive simulation (sampling) should generally be avoided. However, effect estimates obtained from a small sample size are prone to error. The uncertainty of effect estimates can be quantified by bootstrap confidence intervals utilizing the non-parametric resampling technique [188]. Non-parametric resampling makes no prior assumptions on the distribution of the data and aims to reduce the error due to a low number of samples.

Given a data set with sample size n, number of inputs d and a single output, the inputs and output can be merged into one $n \times (d+1)$-dimensional data set $\mathbf{D} = (\mathbf{X}, \mathbf{Y})$. Non-parametric resampling is applied on the rows of the data set $\mathbf{D}$.

Confidence interval of effects

The non-parametric resampling technique is used to calculate confidence intervals for the estimates of total and direct effects. For this purpose, the following steps are followed [189]:

1. Apply non-parametric resampling with replacement on $\mathbf{D}$ to create a large number (e.g. 1000) of bootstrapped data sets of the original data set

2. Calculate estimates of direct and total effects individually for each bootstrapped data set

3. Calculate the confidence intervals around the estimates

Statistical significance test

In order to determine non-influential parameters, a significance test can be exercised by employing the non-parametric resampling technique. Thereby, statistical significance of the estimates are obtained with the null hypothesis that the difference between the effect estimates associated with the inputs and a randomly generated dummy variable are zero. The steps required for this purpose are [190]:

1. Add a randomly generated dummy variable of sample size n to the input matrix $\mathbf{X}$ and construct $\mathbf{D}_{dummy}$ comprising the dummy variable as input

2. Apply non-parametric resampling with replacement on $\mathbf{D}_{dummy}$ to create a large number (e.g. 1000) of bootstrapped data sets

3. Calculate estimates of direct and total effects individually for each bootstrapped data set

4. Calculate the differences between the input effects and the dummy variable effect

5. Calculate the confidence intervals around the differences

Confidence intervals of effect estimates which include zero are considered not significant at the desired confidence level.

6.5 Summary

In this chapter, the evaluation of soft (parametric) faults in terms of global sensitivity analysis is proposed. The approach facilitates to prioritize soft faults based on the severity of their effects on the circuit response. Initially, the fault model parameter value range for which it causes a soft failure of the circuit response is determined. Subsequently, a screening is conducted in order to identify non-influential faults. Afterwards, effect estimates with confidence intervals are calculated. Based on their statistical significance soft faults are kept or eliminated from the fault list. A metric for the simulation fault coverage is proposed by which the fault list can be further reduced by keeping most of the variability in the circuit response.

The proposed approach for effect estimation is strictly based on the linearity hypothesis between the circuit response and the fault model parameters. If the linearity hypothesis is rejected, the proposed approach can still be used by prior utilizing a rank transformation of the data set [174]. Alternatively, approaches which are not based on the linearity hypothesis can be used, like the variance-based [176] or density-based [175] global sensitivity analysis.

Chapter 7

Experimental results

The experimental results are presented for modules from two different automotive safety-related electronic control units (ECUs):

1. High-voltage Lithium-ion battery management system.
2. Low-side gate driver circuit module with safety-critical load.

In this chapter, first the ECUs are introduced, their elements and functionality is described. They are used in the case studies on evaluation of safety goal violation, safety-related functional verification, fault grouping and evaluation of soft faults.

7.1 Battery management system module

The high-voltage battery management system (BMS) module in this case study comprises six modules, each comprising twelve lithium-ion cells. The modules control the charging and discharging of each cell. Moreover, they monitor the cells to detect and control safety-critical events, for example cell over-/under-voltage and over-temperature.

7.1.1 Passive balancing of Lithium-ion cells

The Lithium-ion cells in the modules are connected in series. The power electronics and control logic for each cell are connected in parallel see fig. 7.1. In the circuit, the Module Management System (MMS) monitors voltages and temperatures for all twelve cells and balances charge across the cells using passive balancing. In the following, the passive balancing procedure for charging the cells is described. In charging mode, all twelve cells are charged in series at constant current. The MMS halts the charging procedure, if it

detects the maximum allowed voltage (determined by the cell specification) at the electrodes of any cell. Subsequently, the MMS turns on the corresponding power switch, by which the particular cell is discharged via the parallel resistance for a specific lower state-of-charge (SOC) threshold value. When the particular cell reaches this SOC threshold, the MMS turns off the corresponding power switch and continues the constant current charging procedure. This procedure is executed iteratively, until all cells are at the same SOC level, that is the cells are balanced (end of charge).

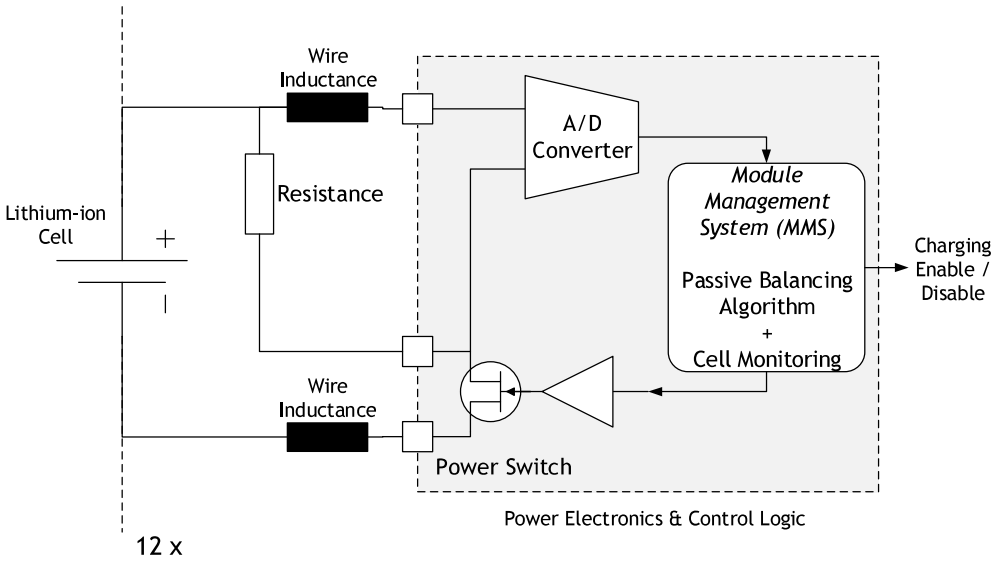


Figure 7.1: Simplified schematic of the battery management module with passive balancing function.

7.1.2 Elements and functionality

The voltages at the cell electrodes are measured by two different A/D converters for different purposes, namely for the cell balancing and voltage monitoring. For cell balancing, 13-bit delta-sigma converters are used for precise voltage reading. For voltage monitoring, 10-bit successive approximation register (SAR) converters are used for fast cell over-voltage and under-voltage detection. In the circuit, the SAR converters are safety mechanisms, dedicated to the prevention of a hazardous event due to over-voltage or under-

voltage at the cell electrodes. In the presence of a failure of the delta-sigma converters, the SAR converters act by the respective prevention measures, like stopping the charging procedure due to over-voltage detection. In particular, the SAR converters implement two functions during the charging procedure:

- To stop the charging process by sending a short-term digital high over-voltage signal to the MMS, if any cell voltage reaches end-of-charge voltage (nominal case or fault is not detected).
- To stop the charging process by sending a permanent digital high over-voltage or under-voltage signal to the MMS, if any cell exceeds end-of-charge voltage or if any cell falls under the end-of-discharge voltage (fault is detected).

7.1.3 Case study: Evaluation of safety goal violations

The high-voltage battery management system (BMS) is a safety-related element of the electric drive train in electric vehicles, see fig. 7.2. The electric drive train is an item of the electric vehicle. That is, safety goals are defined for the high-voltage battery cells. Their potential violation due to random hardware failures must be evaluated at the item level. That is, a test bench covering the BMS and the battery cells must be considered. The presented approach addresses the evaluation of safety goal violations due to random hardware failures in the power electronics and control logic. In particular, the occurrence of random hardware failures during the charging process is considered. The failure modes of several components in the BMS module are extracted from a dedicated FMEDA and are listed in tab. 7.1.

Safety goals for the Lithium-ion cells

Safety goals are determined by a hazard and risk analysis of the BMS module. The safety goals considered in this case study are:

- SG1: Avoid deep-discharge of any battery cell (ASIL D)
- SG2: Avoid over-charge of any battery cell (ASIL D)
- SG3: Avoid external short circuit of any cell (ASIL D)

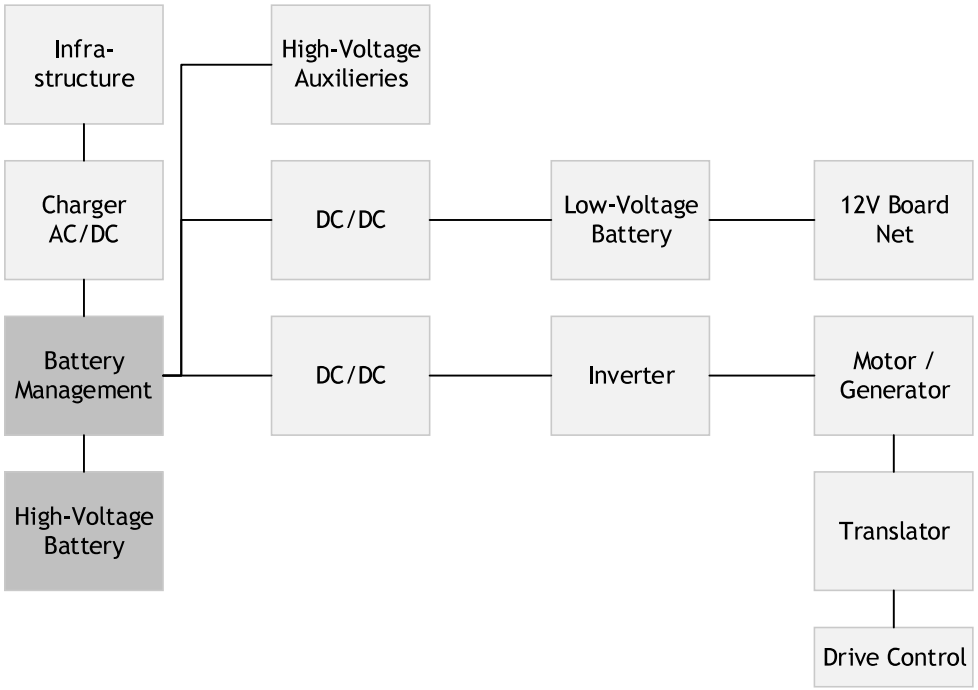


Figure 7.2: Electric drive train as item with corresponding hardware elements.

Failure mode classification

The BMS module comprises a safety mechanism stated by the SAR converter. Therefore, all single-point faults (SPFs) which lead to a safety goal violation can be classified residual faults (RFs). Faults which lead to a failure (i.e. any deviation of the component's performance from nominal) but no safety goal violation, must be considered for further MPF analysis because of a potential perceived multi-point fault (MPF,P) in combination with another independent fault. Faults which lead to no failure and hence to no safety goal violation (not perceived, nor detected) need further MPF analysis because of a potential latent multi-point fault (MPF,L) in combination with another independent fault. Faults which are detected by the SAR converter are considered for further MPF analysis because of a potential detected multi-point fault (MPF,D) in combination with a fault in the SAR converter.

Table 7.1: Test case, components, failure modes and ISO 26262 related failure mode classification based on simulated results for one Lithium-ion cell and corresponding components.

Case study description		
Safety goals	Li-ion battery cells; over-/under-voltage, external short-circuit	
Circuit	Li-ion BMS module; power electronics and control logic	
Test case	10A constant current charging with passive balancing	
Condition	Nominal ambient conditions; all cell SOCs at 60%	
Fault list for fault injection and failure mode classification results		
Component	Component failure mode⟶ Effect description	Classification
delta-sigma converter	stuck-at high⟶ only non inverted differential signal	MPF,P,D
	stuck-at low⟶ only inverted differential signal	MPF,P,D
	drift (high/low)⟶ wrong diff. voltage	MPF,L
	DC-fault⟶ short between HV and LV part	RF(SG1, SG3)
	transient spike p/m⟶ differential voltage oscillates	MPF,L
SAR converter	output stuck-at 1⟶ stuck-at max. output value	MPF,P,D
	output stuck-at 0⟶ stuck-at min. output value	MPF,P,D
	switching point wrong⟶ 10% deviation	MPF,P,D
	stuck-at high⟶ only non inverted differential signal	MPF,P,D
	stuck-at low⟶ only inverted differential signal	MPF,P,D
	drift (high/low)⟶ wrong diff. voltage	MPF,L
	DC-fault⟶ short between HV and LV part	RF(SG1, SG3)
	transient spike p/m⟶ differential voltage oscillates	MPF,P,D
Voltage regulator	input to output short⟶ chip over-voltage/destruction	RF(SG2)
	input to output open⟶ no supply for chip	MPF,P,D
	output too high/low⟶ supply over/under-voltage	MPF,P,D
	output oscillates⟶ modulated supply voltage	MPF,P,D
Reference voltage (to delta-sigma converter)	input to output short⟶ reference at supply level	MPF,P,D
	input to output open⟶ no supply for converter	MPF,P,D
	output too high/low⟶ wrong converter output	MPF,L
	output oscillates⟶ wrong delta-sigma output	MPF,P,D

Schematic simulation test bench

The schematic simulation test bench is designed in Cadence Virtuoso. It is basically comprised of individual behavioural model instances for the Lithium-ion cells, the power switches, passive balancing circuitry, the A/D converters (delta-sigma and SAR), a voltage reference for the delta-sigma converters and a voltage regulator.

Configuration

The charging of high-voltage Lithium-Ion cells takes a few hours. However, in the simulation this procedure can be accelerated by scaling the cell capacity to a lower value and adjusting other test-bench parameters, respectively.

With this adjustment, the simulation time can be set to less than a minute in order to simulate the whole charging process, but it also depends on the initial SOC setting for the cells. In the test bench, the cells are charged with 10A constant current and varying state of charge (SOC) initializations of the cells. The charging stops when any delta-sigma converter reads the end-of-charge voltage at any cell. In parallel, the SAR converter checks the cell voltages and disconnects the cells from the charging source in case of failure at any element by disabling a safety relay which is in series connected to the battery. The test bench switches to idle state after the charging process.

Fault modelling

The presented fault modelling covers failure modes defined for the power electronics and control logic of three adjacent cells (10th, 11th and 12th, latter is directly connected to the charging source). Additionally, the failure modes of a reference voltage generator for the delta-sigma converters and a voltage regulator are considered, see tab. 7.1. In this case study, the analog and digital functional fault modelling technique at the component-level is used, see fig. 3.4.

Simulation results

The presented simulation results cover single-point and dual-point faults for several components of the circuit. In particular the power electronics and control logic corresponding to the three adjacent cells (10th, 11th and 12th) are considered. Single-point faults which do not violate any SG are analysed by dual-point fault injection. For the different SOC initializations, the simulation time is determined to be 10s in order to simulate the whole charging procedure until the end of charge.

Failure mode classification results The failure mode classification results obtained from fault simulation of each individual failure mode corresponding to the components of one Lithium-ion cell are shown in tab. 7.1. Most of the faults are perceived and eventually detected by the SAR converter (MPF, P/D). Safety goal violations SG1, SG2 and SG3 are all caused due to bridging failures of component I/Os. For example, a bridging fault at the inputs of the delta-sigma converter causes the violation of SG1 and SG3 because the corresponding cells are then externally shorted (SG3). This causes an under-voltage (SG1) at the cell, because it cannot be controlled by the SAR converter. Some faults are not perceived or detected, nor do they

cause a safety goal violation. These faults are latent and must be considered for further multi-point fault potential.

Next, the failure mode classification under the following assumptions is presented:

- A failure of a component corresponding to one cell may have an effect on an adjacent cell.
- A failure of a component may have different effects for different cell SOCs.

For this purpose, a total amount of 88 failure modes are considered, respective to the components of three adjacent cells, namely the 10th, 11th and 12th, with latter being directly connected to the charging source. Moreover, different initializations for the cell SOCs are considered:

- All cell SOCs at 50%.
- 11th cell with SOC=60%, all other cells at 50%.
- 11th cell with SOC=40%, all others 50%.
- 11th cell with SOC=80%, all others at 40%.

The failure mode classification results with respect to the SOC initialization are plotted in a histogram in fig. 7.3. It can be seen that safety goal violations occur most frequently, when there is a large deviation of the SOC levels among the cells (11th cell with SOC=80%, all others at 40%). Overall, it can be seen that for different cell SOCs, different effects for the same fault can be expected.

Evaluation of failure modes In the remainder, the component failure modes are evaluated with regard to their potential to directly violate SG1 (avoid under-voltage of any cell voltage) and SG2 (avoid over-voltage of any cell voltage). For this purpose, the simulation results for the cell voltages Vcell_x are inspected. The evaluation is presented for a selected set of enumerated single-point faults:

- Fault 42: SAR converter over-voltage detection stuck-at 0 at 11th cell.
- Fault 61: SAR converter over-voltage detection stuck-at 0 at 12th cell.
- Fault 74: Delta-sigma converter inputs shorted at 12th cell.

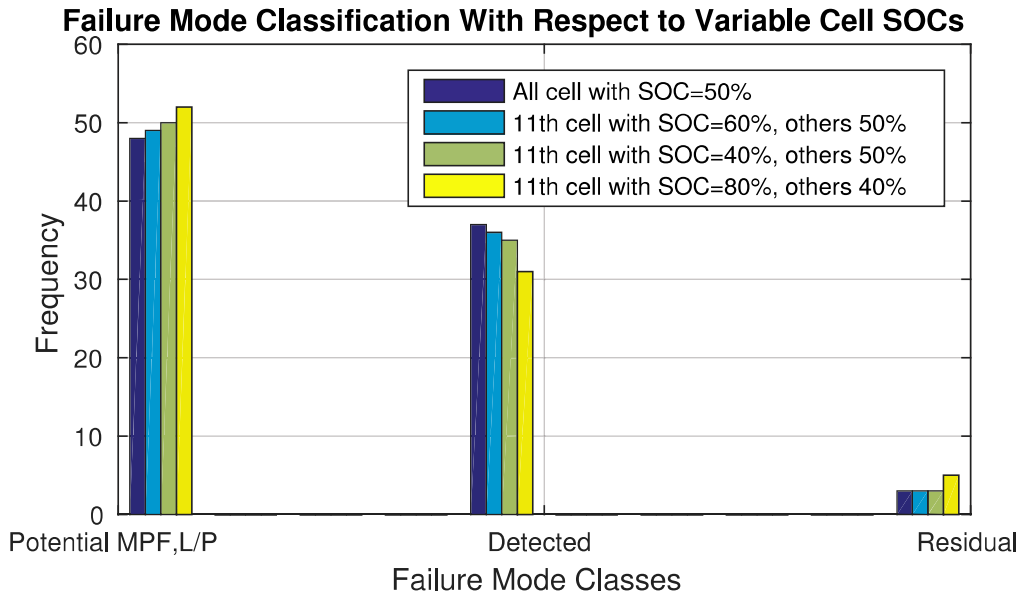


Figure 7.3: Failure mode classification results for single-point fault injection and variable cell SOCs.

- Fault 44: SAR converter under-voltage detection stuck-at 0 at 12th cell.

For evaluation, the waveforms of all cell voltages Vcell_x and the digital outputs of the SAR converter are drawn altogether in one plot for the nominal simulation, see 7.4, as well as for a selected set of fault injection campaigns, see figures 7.5, 7.6 and 7.7. Te plots include the critical over-voltages and under-voltages of the cells are drawn. If any cell voltage is higher than the critical over-voltage or lower than the critical under-voltage, SG2 or SG1 are violated, respectively.

Single-point faults In fig. 7.5, the simulation results corresponding to equal cell SOCs (all cells are initialized at 50%) are shown. Injection of fault 42 is not perceived at the cell voltages and nor is it detected by any SAR converter. After reaching the end-of-charge voltage of all cells, the SAR converter corresponding to the 12th cell stops the charging process by a short-term digital high (pulse) to the MMS. In this context, fault 42 is latent and can be classified as a safe fault (SF), if we can exclude its potential for contribution to a latent multi-point fault (MPF,L) with another independent fault.

Injection of fault 61 (SAR converter over-voltage detection stuck-at 0 at 11th cell) causes the cell voltages to exceed the end-of-charge voltage and is detected by the SAR converters. It writes a permanent digital high to the MMS to stop the charging process permanently. This prevents the fault from violating SG2. In this context, fault 61 is perceived and detected and can be classified a safe fault (SF), if we can exclude its potential for contribution to a detected or perceived multi-point fault (MPF,P or MPF,D) with another independent fault.

Injection of fault 74 (SAR converter over-voltage detection stuck-at 0 at 12th cell) is perceived at the cell voltages and detected by the SAR converter which sends a permanent digital high to the MMS. This fault causes the 12th cell to be externally shorted. The SAR converter cannot prevent the cell voltage to fall below the critical under-voltage. Therefore, fault 74 violates SG1 and SG3 (avoid external short circuit of any cell) and must be classified a residual fault (RF).

In fig. 7.6, simulation results respective to a different cell initialization are shown. Therein, the 11th cell SOC is initialized at 80% and all other cells SOCs are initialized at 40% initialisation are drawn. Same as in the previous results, injection of fault 42 is not detected by any SAR converter. However, contrary to the previous results, fault 42 does violate SG2 and hence must be classified a residual fault (RF). This is due to the fact, that fault 42 is a stuck-at 0 of the digital output of the SAR converter corresponding to the 11th cell. Due to its initialization, the 11th cell reaches as first the end-of-charge voltage. Due to the stuck-at 0 fault of the SAR converter, it cannot send the digital high to the MMS in order to stop the charging process. Thus, the charging continues until another cell reaches it's end-of-charge voltage (in this case the 12th cell). During this, the voltage of the 11th cell exceeds the critical over-voltage.

Injection of fault 61 is perceived and detected by the SAR converters. The charging process stops because 11th cell exceeds the end-of-charge voltage. This fault does not violate any safety goal. However, further analysis to exclude perceived or detected multi-point fault (MPF,P or MPF,D) potential is required.

Injection of fault 74 is perceived but it is not detected by the SAR converter. Also, for the time frame captured in 7.6 it does not violate a safety goal either.

Dual-point fault In fig. 7.7, the simulation results corresponding to a different cell SOC initialization are shown (11th cell with SOC=60%, all other cells at 50%). It can be seen that fault 42 is latent because it is not

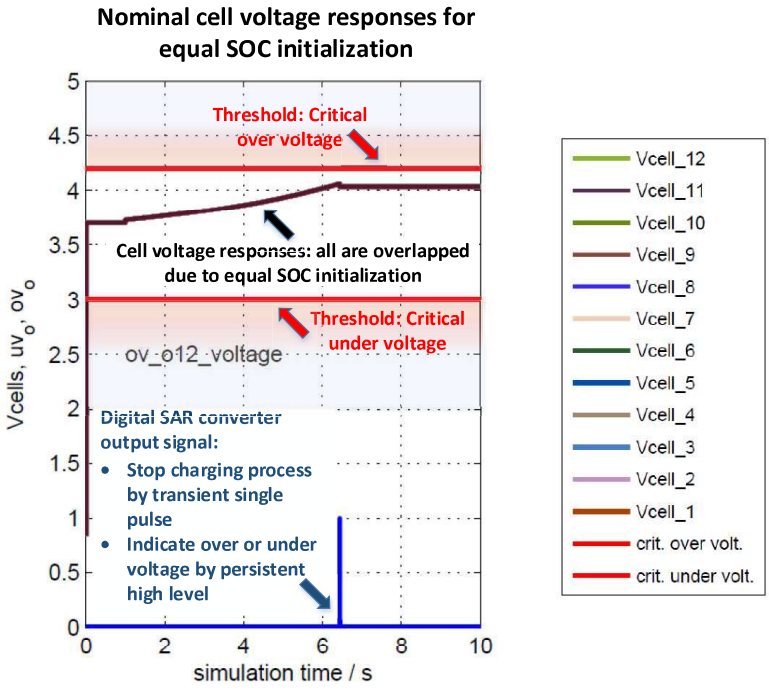


Figure 7.4: Simulation results for the nominal cell voltage response with all cells initialised to equal SOC=50%. Cell voltages, critical over- and under-voltages are drawn together with the digital over-voltage and under-voltage detection.

detected by a SAR converter and also does not violate any safety goal. It is considered for latent multi-point fault potential with another independent fault. In fig. 7.7 it can be seen that this fault causes together with fault 44 (SAR converter under-voltage detection stuck-at 0 at 12th cell) the violation of the over-voltage (SG1) of the 11th cell. Therefore, fault 42 can be classified differently with respect to the cell SOCs:

- SF, if all cell SOCs are at similar level (see fig. 7.5).
- RF, if corresponding cell SOC is at higher level than other cell SOCs (see fig. 7.6).
- MPF,L, in combination with fault 44, if cell SOCs are at similar levels (see fig. 7.7)

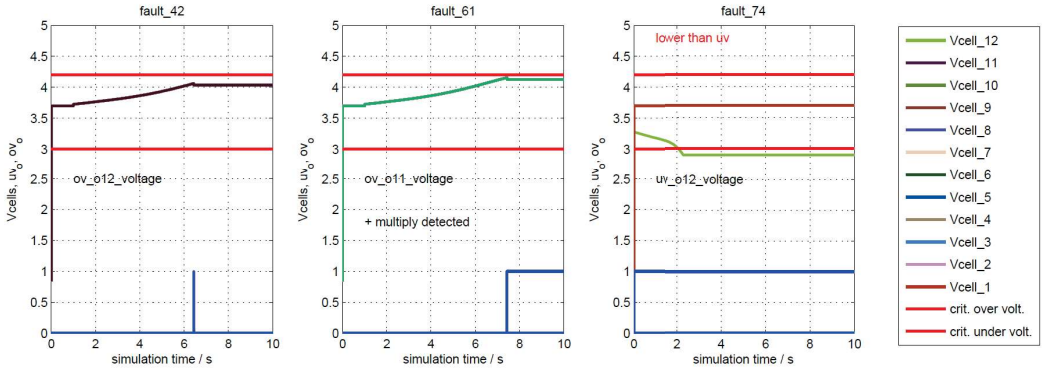


Figure 7.5: Simulation results for single-point fault injection. All cells are initialised to equal SOC=50%. Cell voltages, critical over- and under-voltages are drawn together with the digital over-voltage and under-voltage detection.

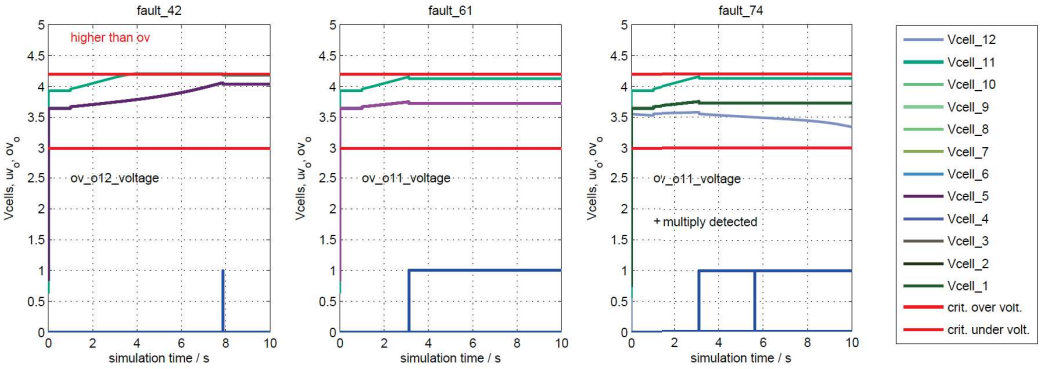


Figure 7.6: Simulation results for single-point fault injection. The 11th cell is initialised to SOC=80% and all other cells to SOC=40%. Cell voltages, critical over- and under-voltages are drawn together with the digital over-voltage and under-voltage detection.

Discussion

The battery management system (BMS) module is a safety-related automotive circuit with stringent safety requirements (ASIL D safety goals). In order to comply with the safety requirements in automotive applications in accordance to the ISO 26262, a method for evident argumentation within the evaluation of safety goal violations due to random hardware failures is needed. To address this, a simulation-based approach is presented, in which the fault injection technique is used. The fault injection covers analog and

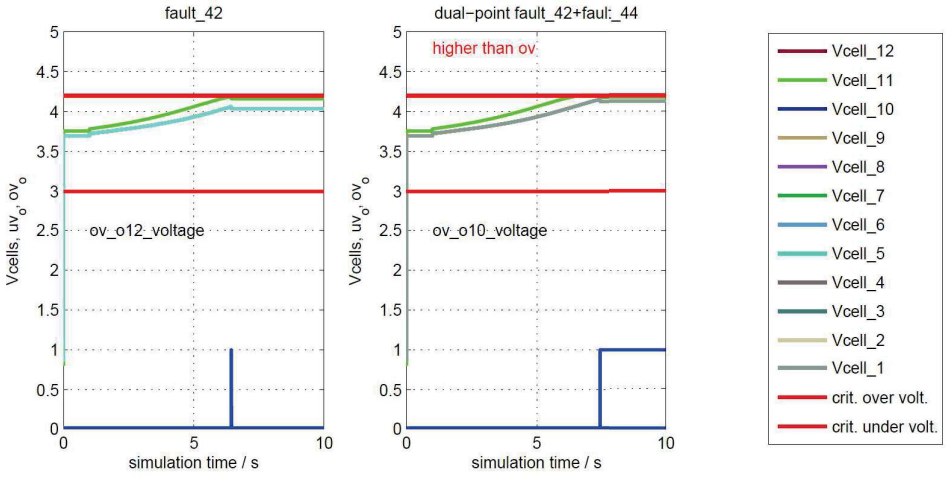


Figure 7.7: Comparison of single-point and dual-point fault injection. The 11th cell is initialised to SOC=60% and all other cells to SOC=50%. Cell voltages, critical over- and under-voltages are drawn together with the digital over-voltage and under-voltage detection.

digital fault models. The faults are injected at the component-level using the functional fault model. The simulation results are used within the failure mode classification procedure in accordance to the ISO 26262. The failure mode classification can subsequently be used for quantification of diagnostic coverage of the safety mechanisms. Therefore, it states a crucial task within the standard and determines the compliance of a product with the target ASIL.

The presented work comprises a constitutive approach and can be used for further investigations on the evaluation of safety goal violations due to random hardware failures. This also covers additional operating conditions as well as test cases in order to expose other hazardous events. Moreover, in the presented simulation results it can be seen that the exposure of a hazardous event also depends on the state of the safety-critical equipment. For Lithium-ion cells, the state-of-charge (SOC) values determine the severity of the effect of the considered failure modes. Thus, depending on the SOCs, failure modes can be classified differently. This consideration adds additional complexity to the overall evaluation procedure and must be covered by the simulation-based approach.

7.2 Low-side gate driver circuit

The low-side gate driver (GD) module is a generic safety-related automotive smart power device in the automotive field. It is designed to turn on and off LEDs or any kind of loads connected to a battery. This functionality is controlled by a digital control unit which communicates with other automotive ECUs via a bus system.

7.2.1 Elements

In the experimental set-up, the GD module drives an LED, see block diagram in fig. 7.8. A typical application of this device is to visually indicate (LED) the failure of other automotive ECUs in the vehicle. The GD module is based on a modular concept in which the digital functionality is divided from the analog functionality, see fig. 7.8. The digital control unit comprises registers, logic and timers to drive and control the analog functionality. The analog part comprises buffers and a low-side driver including a safety mechanism. The buffers interface the digital control unit, the low-side driver and the safety mechanism in a feedback loop.

7.2.2 Functionality

The basic functionality of the GD module is described for two use cases. First, when the GD driver is requested to turn the LED on and second, when a system-level failure occurs during this use case.

Gate driver on

The digital enable signal *en* initiates to turn the LED on. Via the buffer interface, it is an input to the level shifter. The level shifter lifts the voltage level of the input from the digital VDD_{D} to the analog VDD_{A}. This generates the output signal *vs* which drives the gate driver output stage via an intermediate logic circuit to produce the output stage inputs *vn* and *vp*. The circuits of the level shifter and output stage are shown in fig. 7.9 and fig. 7.10, respectively. The output-stage produces an output signal *vg* which drives the Power MOSFET gate voltage in order to turn the LED on or off.

System-level failure

The circuit comprises an over-current limitation (OCL) circuit which is dedicated to over-current failure detection at the system level. For this purpose,

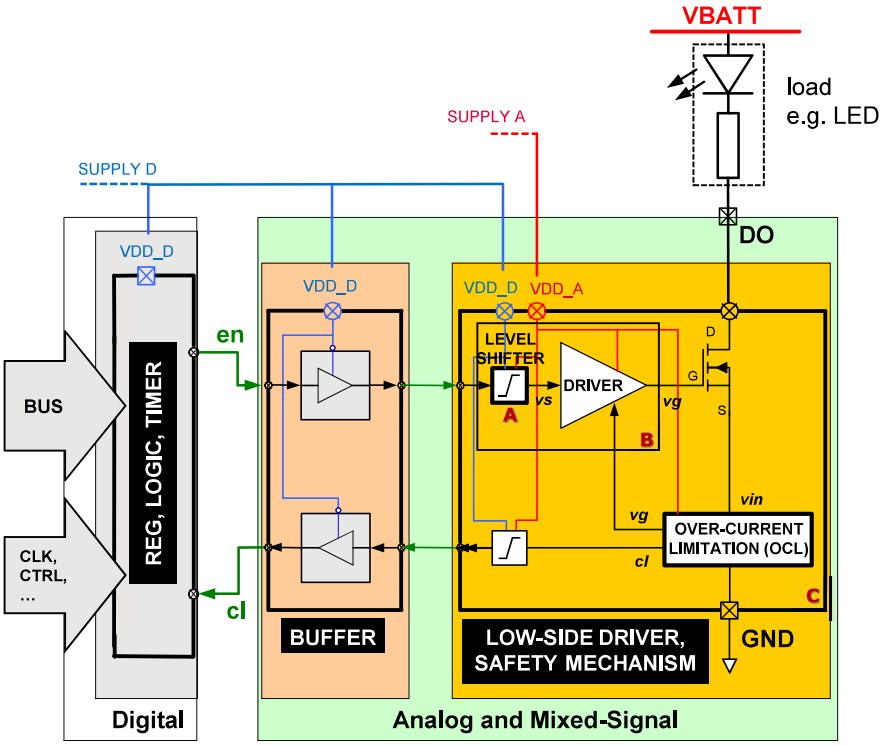


Figure 7.8: Interconnect and stuck-open/short fault grouping results for the level-shifter circuit (LS), gate driver circuit (GD) and over-current limitation circuit (OC).

the current in the Power MOSFET IDS is measured indirectly via a resistance. The voltage on the resistance vin is compared via a reference voltage $vref$ in the OCL circuit. If vin is lower than $vref$, over current is detected. Subsequently, the OCL circuit limits IDS indirectly by controlling the gate voltage vg. This action is communicated with the digital control unit via the active-high signal cl. After a pre-defined shut-down delay time $t_{\mathrm{CL,SD}}$, the digital control unit disables the gate driver and resets cl via the en signal.

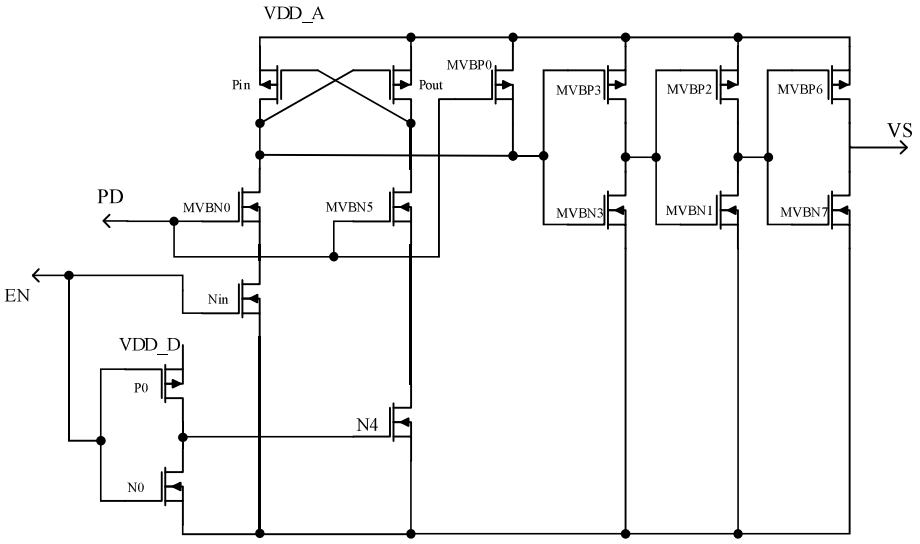


Figure 7.9: Circuit of the active-high level shifter shifter (LS) which lifts the input signal en from the digital level VDD_{D} into the analog level VDD_{A}, resulting in the output signal vs. The second input signal pd indicates a power supply failure in the digital control unit and disables the output signal vs.

7.2.3 Case study: Safety-related functional verification

The low-side gate driver (GD) module functionality is verified at the system level for a number of safety-related test cases. The verification target is compliance with the technical safety requirements. These are derived from the product specification. A self-checking simulation test bench is set-up for the test cases. Simulation results and discussion is presented.

Safety-related test cases

The safety-related test cases are derived from the functional verification plan. In presented experimental results, four different test cases are considered. These are:

- **Driver on** The GD module is requested to turn the LED on for a period of time. The verification target is the compliance with the require-

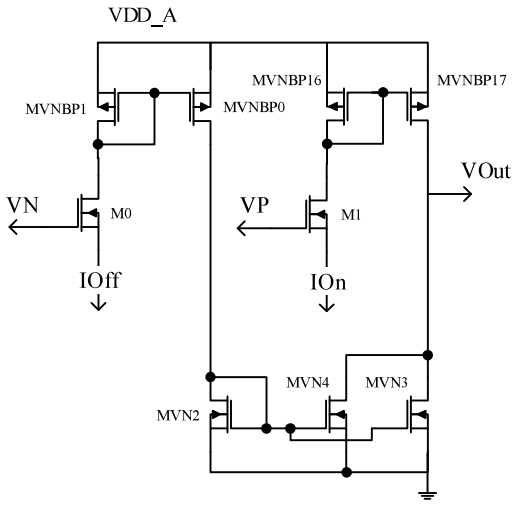


Figure 7.10: Circuit of the low-side gate driver output stage (OS). The gate driver signal v_{Out} is driven by input signals V_N and V_P. The output slew rate is controlled via I_{Off} and I_{On}, provided by bias circuitry (not shown).

ments on the Power MOSFET on-voltage VDS_{on} and its on-resistance RDS_{texton}.

- **Driver off** The GD module is requested to turn the LED off.The verification target is the compliance with the requirements on the Power MOSFET pull-down current IDS_{pd}.

- **Dynamic current limitation** During the Driver on request, the GD module experiences a system-level failure. The failure is due to a short-circuited output load (LED). The verification target is the compliance with the requirements on the limitation of the Power MOSFET current IDS to IDS_{CL}.

- **Shut-down delay time** During current limitation, the digital control unit must disable the GD module after a pre-defined shut-down delay time. The verification target is the compliance with the requirements on the shut-down delay time $t_{\mathrm{CL,SD}}$ between the rising and falling edges of the current limitation signal cl.

Safety-related test cases are created for the analog/mixed-signal parts of the GD module. For this purpose, fault injection is considered in the

level shifter, the output-stage and the over-current limitation circuit. The components are mainly composed of MOSFET devices. The MOSFET stuck-open and stuck-short failure modes cover already 60% of the failure rate of one device. They are modelled by very high-ohmic resistive opens (1GΩ in series to the transistor drain) and very low-ohmic resistive shorts (1mΩ in parallel to the transistor), respectively. All faults in all components are considered with equal probability of occurrence.

Simulation test bench

The simulation test bench is self-checking regarding the verification targets and covers the safety-related test cases. It comprises a mixed-mode model of the GD module, where the digital core is modelled in RTL and the analog part at circuit level. Different loads are configured to cover the nominal case as well as the system-level failure case.

Simulation results

The simulation results are presented in two parts. In the first part, the collective verification results are shown for the component fault injections in different test cases. In the second part, the simulated waveforms are shown for two safety-related test cases.

Verification results Fig. 7.11 gives an overview of the verification results. It shows the percentage of failed tests in the presence of MOSFET stuck-open and stuck-short faults in the level shifter, output stage and the over-current limitation circuit. The results are illustrated for the different test cases driver on, driver off dynamic current limitation and shut-down delay time.

Overall, most tests failed during the dynamic current limitation and shut-down delay time test. These test cases are most sensitive to the accurate function of all components.

No test failed during the driver off test case with fault injection in the over-current limitation circuit. That is, all faults occurring in the over-current limitation circuit are latent until the GD module is turned on. Generally, the driver off test case e states that there is no request from any other automotive ECU to turn on the LED in order to visually indicate a failure of the requester. This test case can be considered to be the most common use case for the GD module. In this context, the verification results show that unless adequate measures are implemented, the failure of the over-current limitation will be undetected until an automotive ECU fails and requests the GD module to indicate this. For an equal probability of occurrence for all

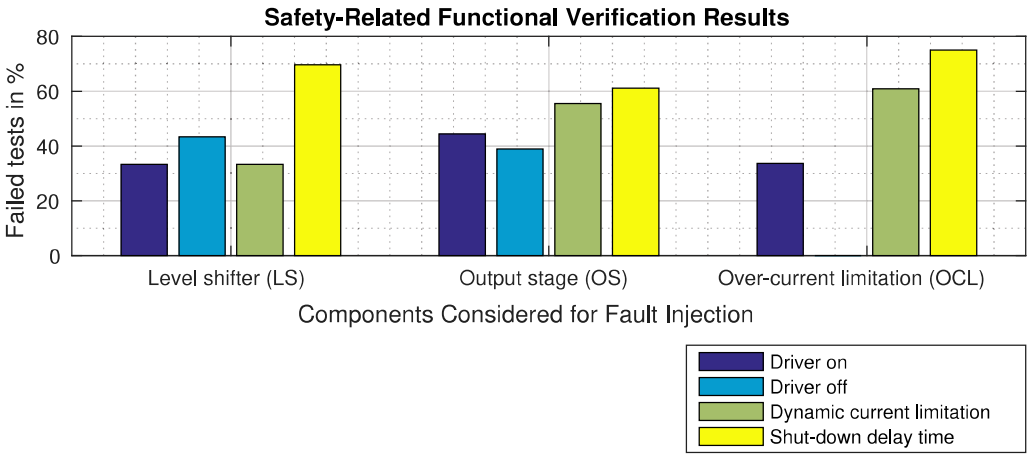


Figure 7.11: Functional verification results for safety-related test cases. The percentage of failed tests in the presence of MOSFET stuck-open and stuck-short faults in the level shifter, output stage and over-current limitation circuit are illustrated for the different test cases.

faults in the over-current limitation circuit, this means that 30% of faults will cause the GD module to not comply with its requirements to drive the LED on. Moreover, in the presence of a simultaneous system-level failure, more than 60% of faults will cause the GD module to not comply with its requirement on current limitation and more than 70% of faults will cause the GD module to not comply with its requirement on the shut-down delay time.

Waveforms The simulated waveforms of the Power MOSFET voltage VDS and current IDS are plotted for the dynamic current limitation test case. The waveforms for nominal (fault-free) simulation are compared with the waveforms in the presence of (see gate driver output stage circuit topology in fig. 7.10)

- PMOS MVBP16 stuck-open,
- NMOS M0 stuck-open, and
- PMOS MVBP1 stuck open.

Dynamic current limitation (nominal) In the dynamic current limitation test case, the driver is off until it is turned on at $t = 50us$, see for example fig. 7.12. While the driver is on, a system-level failure occurs at $t = 100us$ and the load (LED) is shorted. This causes an increasing current

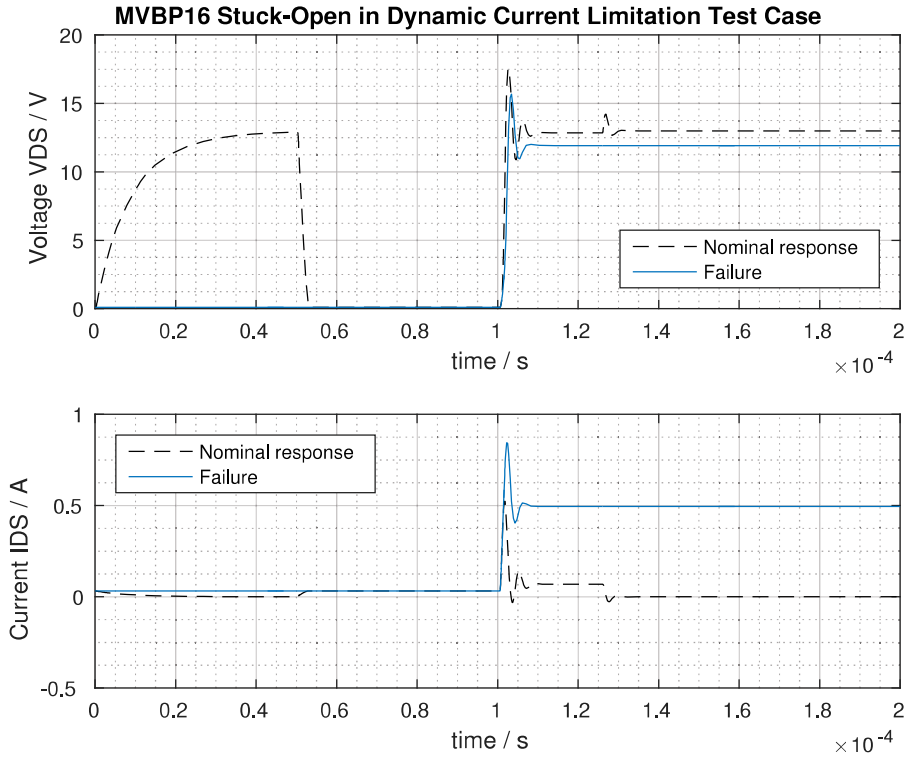


Figure 7.12: Failure caused by fault injection of a stuck-open fault model of PMOS MVBP16 in the gate driver output stage. Simulated waveforms of the Power MOSFET voltage VDS and current IDS in nominal and failure mode during the dynamic current limitation test case.

IDS through the Power MOSFET which is subsequently limited by current limitation circuit. After a delay time the GD module is shut down at $t \approx 130us$.

Stuck-open fault of MVBP16 The stuck-open failure mode in the output stage causes the Power MOSFET to be stuck-on. Starting by $t = 0$, the voltage across the Power MOSFET is set to VDS_{on}. When the load is shorted, current IDS through the Power MOSFET increases and cannot be sufficiently limited by the over-current limitation circuit.

Functional equivalent faults The NMOS M0 stuck-short fault and PMOS MVBP1 stuck-open fault cause very similar circuit responses in the dynamic current limitation test case, see fig. 7.13 and 7.14, respectively. Both cause the Power MOSFET to be stuck-on. When the load is shorted,

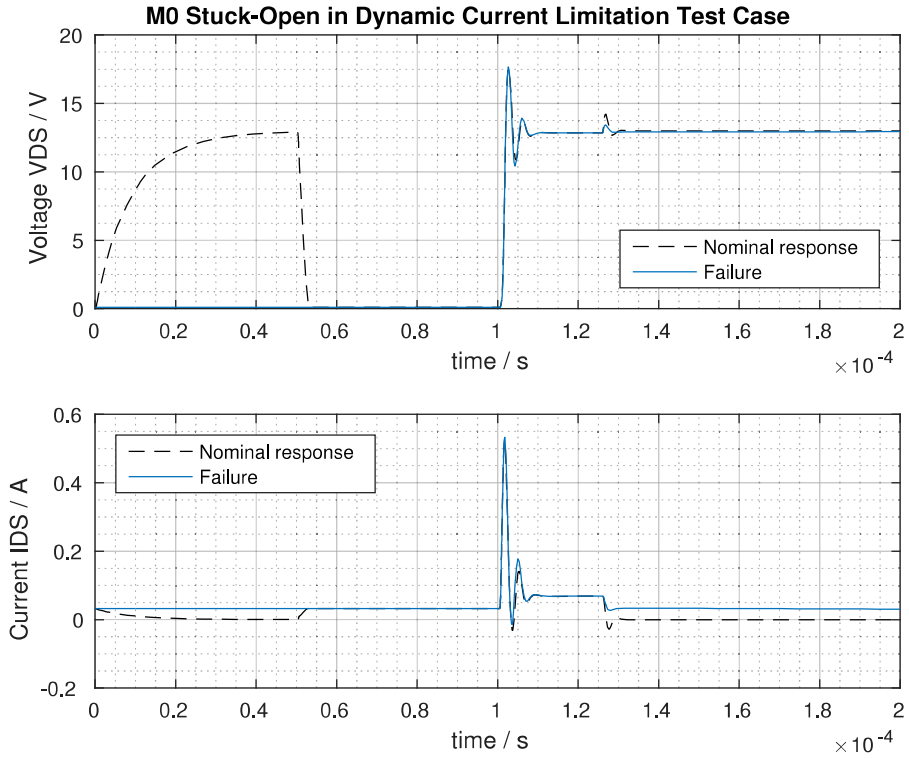


Figure 7.13: Failure caused by fault injection of a stuck-open fault model of NMOS M0 in the gate driver output stage. Simulated waveforms of the Power MOSFET voltage VDS and current IDS in nominal and failure mode during the dynamic current limitation test case.

current IDS through the Power MOSFET increases and can be sufficiently limited by the over-current limitation circuit. However, after the shut-down delay time, the Power MOSFET is still conducting with $IDS = IDS_{\text{on}}$.

Before the load is shorted, all three failures of M0, MVBP1 and MVBP16 are similar. Thus, for the test cases driver on and driver off, they can be considered functional equivalent.

Discussion

Based on functional test cases of the low-side gate driver (GD) module, safety-related test cases are derived. The verification target is compliance with safety requirements, derived from the product specification. The simulation test bench is set-up for system-level verification. Fault injection is done to three components in the analog part of the GD module. These are

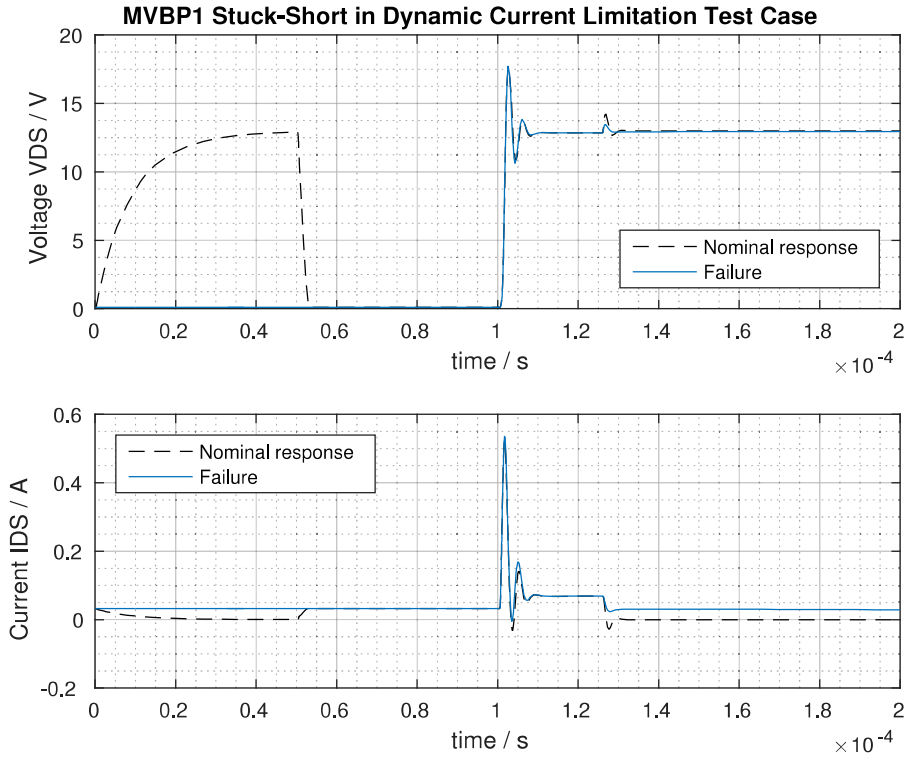


Figure 7.14: Failure caused by fault injection of a stuck-short fault model of PMOS MVBP1 in the gate driver output stage. Simulated waveforms of the Power MOSFET voltage VDS and current IDS in nominal and failure mode during the dynamic current limitation test case.

a level shifter (LS), the gate driver output stage (OS) and an over-current limitation (OCL) circuit .

The simulation results are particularly useful to determine compliance with safety requirements and the latency of faults for different test cases. It can be shown, that depending on the test case, different faults can cause similar effects on the circuit response. These are individually simulated but are evaluated identical in the verification procedure.

7.2.4 Case study: Hierarchical fault injection

The hierarchical fault injection approach is deployed on the low-side gate driver (GD) module. In order to identify functional equivalent fault groups at component-level, component-level faults are run. Subsequently, the simulated waveform data is processed by the fault grouping algorithm. An optimal

set of fault groups is calculated and representative faults are presented.

In the GD module, the level shifter (LS), gate driver output stage (OS) and over-current limitation (OC) circuit are considered in this context.

Simulation test bench

Component-level simulation test benches are created for the LS, OS (including LS and output stage) and OCL components. The stimuli for the components are derived from measurements of the component input signals from the system-level nominal simulations for the driver on and driver off test cases.

For all components, fault injection of MOSFET stuck-open (1GΩ in series to the transistor drain) and stuck-short (1mΩ in parallel to the transistor) faults are considered. Additionally, for the LS component, local bridging faults (1mΩ in parallel to the transistor) on the MOSFET pins is considered.

The schematics of the LS and OS are shown in fig. 7.9 and fig. 7.10, respectively. The netlist of the OC is composed of 46 MOSFET devices.

Simulation results and discussion

An overview of the fault grouping results for each component are shown and subsequently discussed in-detail for the level shifter and output stage. For this purpose, the hierarchical clustering scheme in form of a dendrogram is presented. The cluster validation results are shown and the representative faults are presented.

Overview of fault grouping results The results of fault list reduction by the fault grouping approach are shown for each component and fault type in fig. 7.15 with respect to the cluster validation index which was used to determine the optimal number of fault groups.

The strongest fault list reduction is obtained for the over-current limitation component with the stuck-short/open fault type which is reduced by more than than 82% compared to the initial fault list comprised of 92 faults. All cluster validation indices suggest an optimal number of fault groups to be eleven.

For the output stage component, the level shifter is considered to be its sub-component, hence, the stuck-short/open faults in the level shifter are also comprised in the results for the output stage. It can be seen, that for the output stage even smaller number of fault groups is suggested than for the level shifter. The reason for this is that some faults of the LS propagate either to stuck-at-high or stuck-at-low faults when entering the output stage.

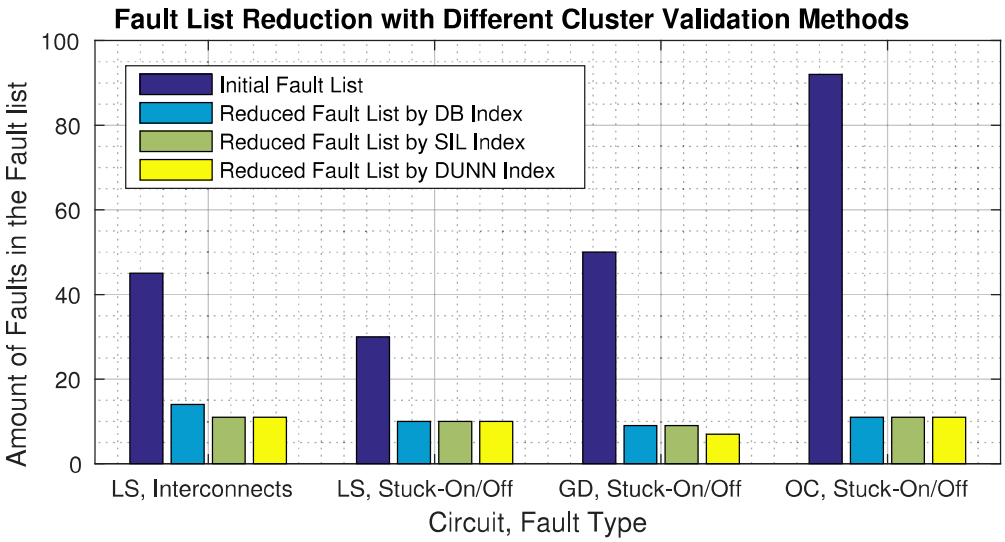


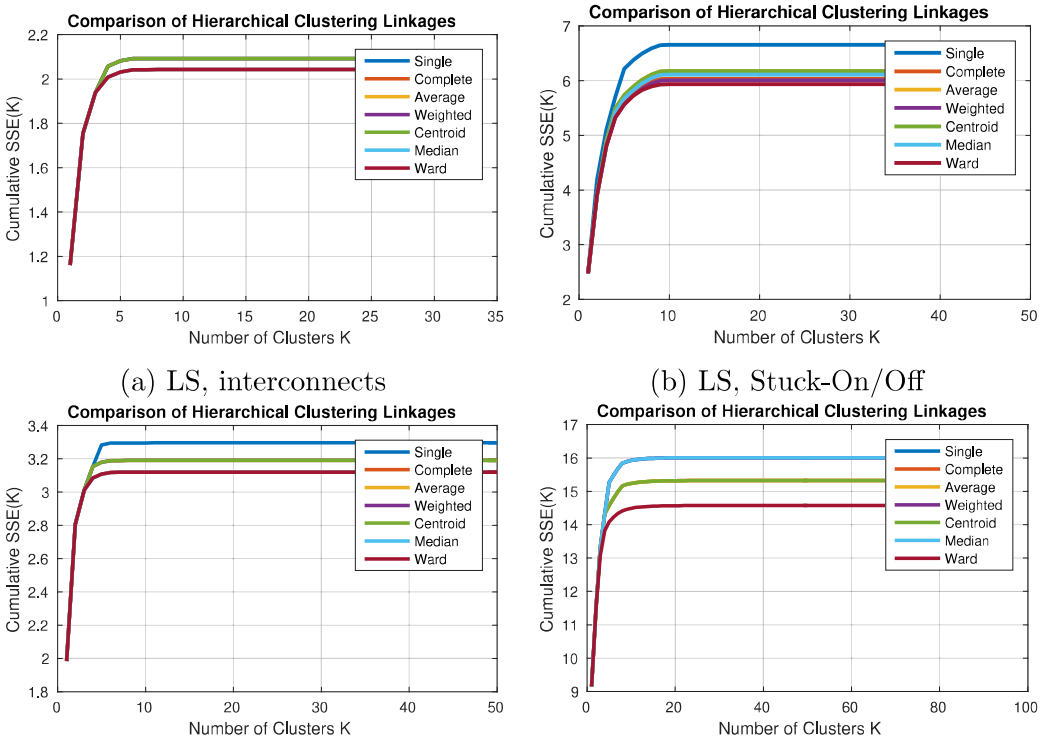
Figure 7.15: Interconnect and stuck-open/short fault grouping results for the level-shifter circuit (LS), gate driver circuit (GD) and over-current limitation circuit (OC).

The level shifter and the output stage are connected with each other via an intermediate CMOS circuit, which transforms the single output of the level shifter into two inputs vp and vn for the output stage. That is, although the number of faults increases from sub-component to component-level, the number of fault groups decreases with increasing hierarchy level.

For the level-shifter, stuck-short/open and bridging faults are separately considered. For the bridging fault type a higher fault list reduction is achieved (76%) than for the stuck-short/open fault type (67%).

It can be concluded that the hierarchical fault injection approach scales well for hierarchical architectures (level shifter as a sub-component of the output stage), large circuits with many devices (over-current limitation circuit) and in the presence of many faults (bridging faults in the level shifter).

Hierarchical clustering scheme The hierarchical clustering algorithm produces different results, depending on the linkage function used, see tab. 5.2. Fig. 7.16 shows the cumulative within-cluster sum-of-squared errors (SSE) over the number of total clusters K. It can be seen that in any case, the Ward Linkage produces clustering results with minimum cumulative SSE. For fault grouping, the goal is to generate fault groups in which all faults are as similar as possible, i.e. fault groups with minimal SSE. Therefore, the Ward linkage is used for fault grouping.



(a) LS, interconnects

(b) LS, Stuck-On/Off

(c) GD, Stuck-On/Off

(d) OC, Stuck-On/Off

Figure 7.16: Comparison of the cumulative sum-of-squared errors using different linkage functions in the hierarchical clustering algorithm.

The hierarchical clustering schemes are shown in form of dendrogram with Ward linkage function for the level shifter component and the output stage component with level shifter as sub-component, see fig. 7.17 and fig. 7.18, respectively. The faults are enumerated including the nominal circuit response (number 1 for the level shifter, 1 and 32 in the output stage). In fig. 7.17 it can be seen that five other faults (number 5, 8, 25, 26, 29, 30) cause a similar circuit response as the nominal circuit response. The distance (vertical axis) of these objects are very small to each-other.

From the structure of the presented dendrogram it can be concluded that the faults are mostly either very similar to each-other or very different from each-other. This conclusion generally applies for the components which are stimulated by a limited number of non-randomly generated input signals. In this case, non-random stimuli are chosen because they correspond to the stimulation of the component during the driver on and off test cases.

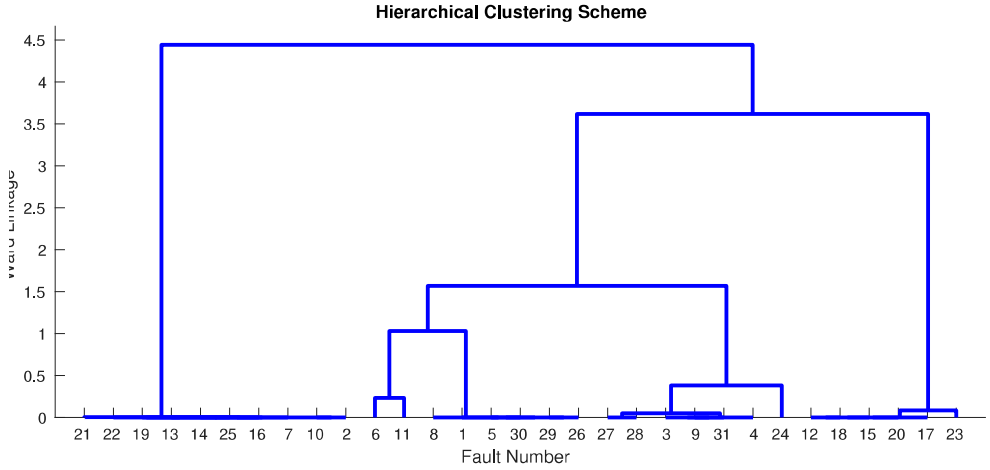


Figure 7.17: Hierarchical clustering scheme (dendrogram) of stuck-open/-short faults for the level shifter with enumerated faults (1 is the nominal circuit response).

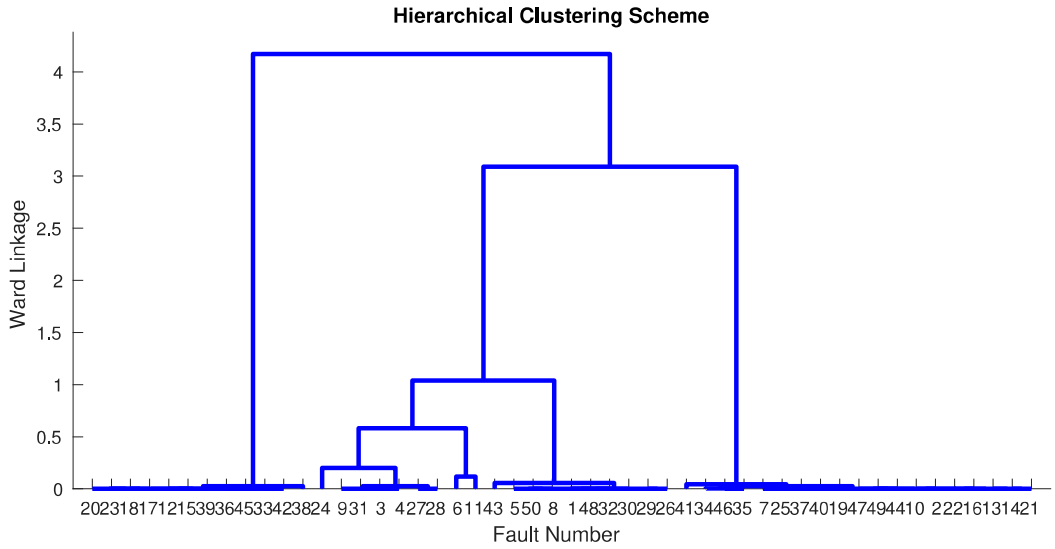
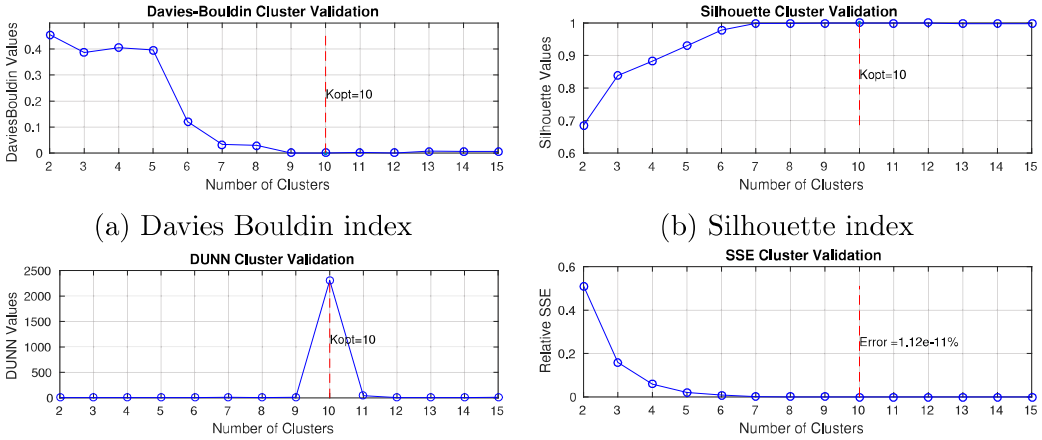


Figure 7.18: Hierarchical clustering scheme (dendrogram) of stuck-open/-short faults in the output stage with the level shifter as sub-component. The level shifter faults are enumerated from 1 to 31 , the output stage faults are enumerated from 32 to 50 (1 and 32 are nominal circuit responses).

Cluster validation The cluster validation results for the level shifter component and output stage with level shifter as sub-component are shown in fig.



(a) Davies Bouldin index

(b) Silhouette index

(c) DUNN index

(d) Normalized sum-of-squared errors

Figure 7.19: Cluster validation results for stuck-open/-short faults in the level shifter.

7.19 and 7.20, respectively. Additionally, the sum-of-squared errors (SSE) is drawn over the number of fault groups. It is generally desirable to achieve an SSE as low as possible in order to guarantee high representativeness of the representative faults.

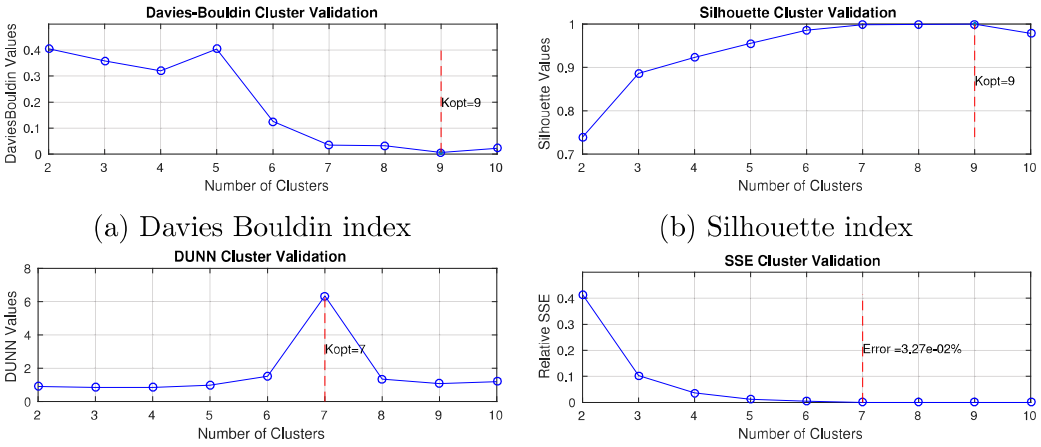
For the level shifter, all cluster validation indices suggest the ten to be the optimal number of fault groups. For the output stage, Davies Bouldin index and Silhouette index suggest nine but the Dunn index suggests seven. The circuit responses (waveforms) are inspected for each suggestions for similarity. Qualitatively, seven is considered to be an adequate number of fault groups. The suggestion of ten fault groups is considered too conservative.

Representative faults and reduced fault lists Tab. 7.3 shows a listing of all devices, enumerated stuck-open/-short faults considered in the level shifter, fault group assignments and the arguments for medoid and worst-case representative fault calculation. A medoid argument which is zero indicates that the fault is single in its fault group. A worst-case argument which is very small or zero indicates that the fault is equal or very similar to the nominal circuit response.

In tab. 7.3, the reduced fault list is shown for the level shifter component. It is based on the representative faults obtained by using the medoid and worst-case criteria. If the faulty circuit responses are distributed homogeneously and the SSE in the fault group is small, the medoid criterion is considered to determine the representative fault. However, if the faulty circuit responses are distributed in-homogeneously and the SSE within the fault

Fault number f	Fault model	Fault group k	Medoid argument $\sum_{f' \in C_k} d_{f,f'}$	Worst-case argument $d_{f,f'=nominal}$
1	Nominal	10	1.4845e-08	0
2	Pin, stuck-short	2	1.0491e-07	0.24749
3	Pin, stuck-open	4	1.3972e-08	0.17048
4	Pout, stuck-short	4	5.5877e-08	0.17048
5	Pout, stuck-open	10	2.9689e-08	2.5244e-13
6	MVBN0, stuck-short	7	0	0.19203
7	MVBN0, stuck-open	2	1.3607e-07	0.24749
8	MVBN5, stuck-short	10	1.0391e-07	1.4844e-08
9	MVBN5, stuck-open	4	6.9848e-08	0.17048
10	MVBP0, stuck-short	2	2.4098e-07	0.24749
11	MVBP0, stuck-open	8	0	0.19039
12	MVBP3, stuck-short	6	6.4956e-07	0.3562
13	MVBP3, stuck-open	2	2.7214e-07	0.24749
14	MVBN3, stuck-short	2	3.033e-07	0.24749
15	MVBN3, stuck-open	6	1.2991e-06	0.3562
16	MVBP2, stuck-short	2	3.3446e-07	0.24749
17	MVBP2, stuck-open	6	3.8345e-06	0.3562
18	MVBN1, stuck-short	6	4.4841e-06	0.3562
19	MVBN1, stuck-open	2	3.6577e-07	0.24749
20	MVBP6, stuck-short	6	5.1671e-06	0.3562
21	MVBP6, stuck-open	1	0	0.24756
22	MVBN7, stuck-short	2	4.0298e-07	0.24749
23	MVBN7, stuck-open	5	0	0.36875
24	Nin, stuck-short	9	0	0.20117
25	Nin, stuck-open	2	4.3414e-07	0.24749
26	P0, stuck-short	10	1.1875e-07	3.0894e-13
27	P0, stuck-open	3	9.249e-15	0.17002
28	N0, stuck-short	3	1.8498e-14	0.17002
29	N0, stuck-open	10	1.3359e-07	2.812e-13
30	Nout, stuck-short	10	1.4844e-07	2.5981e-13
31	Nout, stuck-open	4	8.3819e-08	0.17048

Table 7.2: Results for $K_{\text{opt}} = 10$ optimal fault groups in the level shifter component: devices, enumerated stuck-open/-short faults, fault group assignments and arguments for medoid and worst-case representative fault calculation.



(a) Davies Bouldin index

(b) Silhouette index

(c) DUNN index

(d) Normalized sum-of-squared errors

Figure 7.20: Cluster validation results for stuck-open/-short faults in the output stage (with level shifter as sub-component).

Fault group k	Medoid f	Worst case f
1	21	21
2	16	22
3	27	27
4	9	4
5	23	23
6	18	20
7	6	6
8	11	11
9	24	24
10	30	8

Table 7.3: Reduced stuck-open/short fault list for the level shifter.

group is large, the worst-case criterion is considered to determine the respresentative fault. The reason for this is that in the presence of high SSE within the fault group, the medoid may not represent the faults which may be more dissimilar to the nominal circuit response. However, latter faults may bear more severe consequences than the medoid and hence be more safety-critical. Thus, the worst-case criterion is more useful in this case because the missing of the worst-case within each fault group is avoided.

7.2.5 Case study: Evaluation of soft structural faults

The evaluation of soft structural faults is deployed on the low-side gate driver (GD) module. A total of 18 soft stuck-open/short faults are considered among the MOSFET devices in the output stage, see fig. 7.10. First, the soft failure ranges are determined by simulation. Subsequently, screening is applied in order to identify non-influential faults. Afterwards, the direct and total effect estimates are calculated. The contribution of each fault on the explained variance of the circuit response is calculated. Non-parametric re-sampling is used to calculate confidence intervals for the effect estimates and statistical significance of the effects.

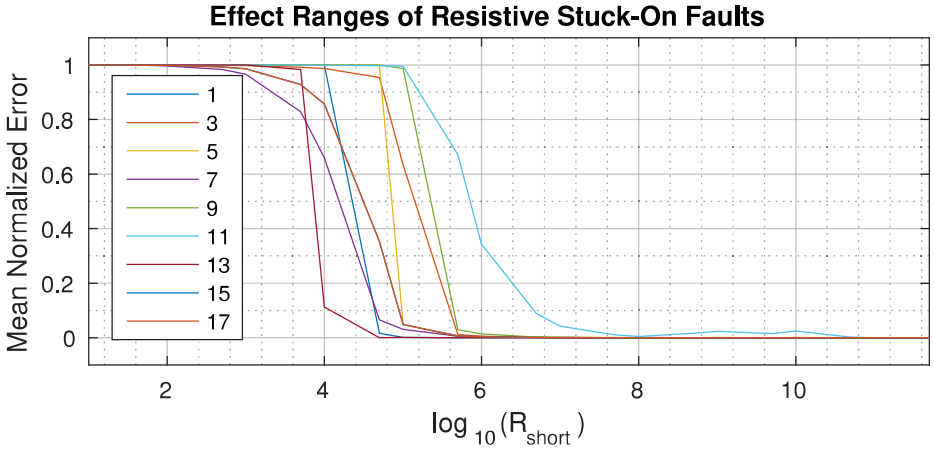
Determining soft failure ranges

Resistive fault models are considered for the injection of soft stuck-open (fault model in series to the transistor) and stuck-short (fault model in parallel to the transistor) faults in the MOSFET devices of the output stage. Each fault model is enumerated and listed in tab. 7.4. They are simulated for uniform distributed resistance values. Stuck-short resistance R_{short} of is varied between 10Ω and $1 \cdot 10^{11}\Omega$. Stuck-open resistance R_{open} is varied between 100Ω and $1 \cdot 10^{12}\Omega$.

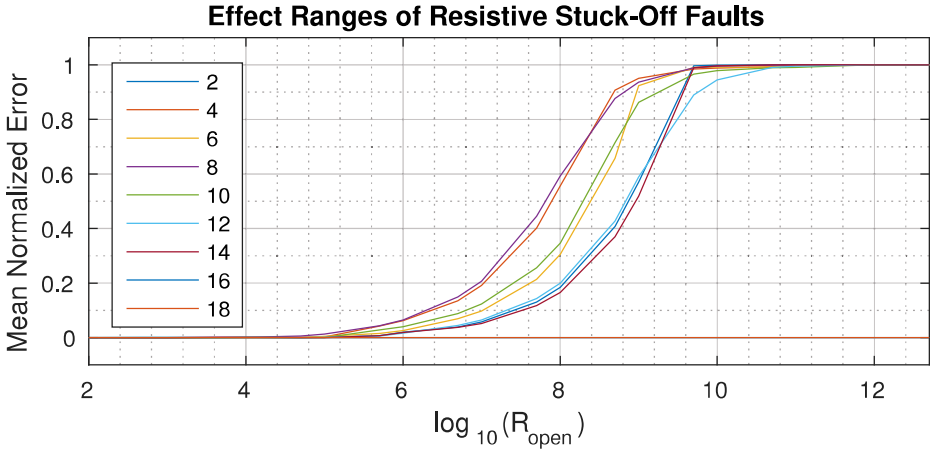
From the simulations, the circuit response is produced, post-processed and the mean normalized output error is calculated for the waveform with respect to the nominal circuit response, see 7.21. A normalized error which is close to zero indicates high similarity of the fault with the nominal circuit response for the corresponding resistance value. For these resistance values, no failure is perceived on the output signal. A normalized error which is close to one indicates maximum dissimilarity of the fault with the nominal circuit response. For these resistance values, maximum failure is perceived on the output signal, thus they are hard faults. The circuit response for each fault and for variable resistance values are plotted in fig. 7.21, including the nominal fault-free circuit response. The circuit responses for all faults

Fault no.	Device name	Fault model	Distribution
1	MVBP1	stuck-short	Uniform($5 \cdot 10^4$,$1 \cdot 10^5$)
2	MVBP1	stuck-open	Uniform($1 \cdot 10^5$,$1 \cdot 10^9$)
3	MVBP0	stuck-short	Uniform($1 \cdot 10^5$,$1 \cdot 10^6$)
4	MVBP0	stuck-open	Uniform($1 \cdot 10^5$,$1 \cdot 10^8$)
5	MVBP16	stuck-short	Uniform($1 \cdot 10^5$,$5 \cdot 10^5$)
6	MVBP16	stuck-open	Uniform($5 \cdot 10^4$,$5 \cdot 10^8$)
7	MVBP17	stuck-short	Uniform($1 \cdot 10^4$,$5 \cdot 10^5$)
8	MVBP17	stuck-open	Uniform($1 \cdot 10^4$,$1 \cdot 10^8$)
9	M0	stuck-short	Uniform($5 \cdot 10^5$,$5 \cdot 10^6$)
10	M0	stuck-open	Uniform($1 \cdot 10^5$,$1 \cdot 10^8$)
11	M1	stuck-short	Uniform($5 \cdot 10^5$,$5 \cdot 10^6$)
12	M1	stuck-open	Uniform($5 \cdot 10^5$,$1 \cdot 10^9$)
13	MVN2	stuck-short	Uniform($1 \cdot 10^4$,$5 \cdot 10^4$)
14	MVN2	stuck-open	Uniform($1 \cdot 10^5$,$1 \cdot 10^9$)
15	MVN4	stuck-short	Uniform($5 \cdot 10^4$,$1 \cdot 10^6$)
16	MVN4	stuck-open	Uniform($5 \cdot 10^3$,$1 \cdot 10^7$)
17	MVN3	stuck-short	Uniform($5 \cdot 10^4$,$1 \cdot 10^6$)
18	MVN3	stuck-open	Uniform($5 \cdot 10^3$,$1 \cdot 10^7$)

Table 7.4: Stuck-open/short fault list for soft fault injection with resistive fault models in the output stage of the low-side gate driver module.



(a) Normalized output error over $\log_{10}(R_{short})$



(b) Normalized output error over $\log_{10}(R_{open})$

Figure 7.21: Resistance ranges for hard, soft and nominal stuck-open/-short faults in the output stage.

with respect to the variable fault model parameter are plotted in 7.23 with a close-up in fig. 7.22.

For the faults with numbers 16 and 18, straight lines are drawn in 7.21b, because they have singly no effect on the output for any resistance value. However, they only have an effect if they occur simultaneously, see right-most plot at the bottom in 7.21b. This is due to the fact that both transistors are connected in parallel. That is, the single stuck-short faults will have the same effect, but the single stuck-open faults may have no effect.

The soft failure range is determined with respect to the resistance values which cause a mean normalized error between the nominal and hard failing circuit response. This is quantitatively determined to be the range between

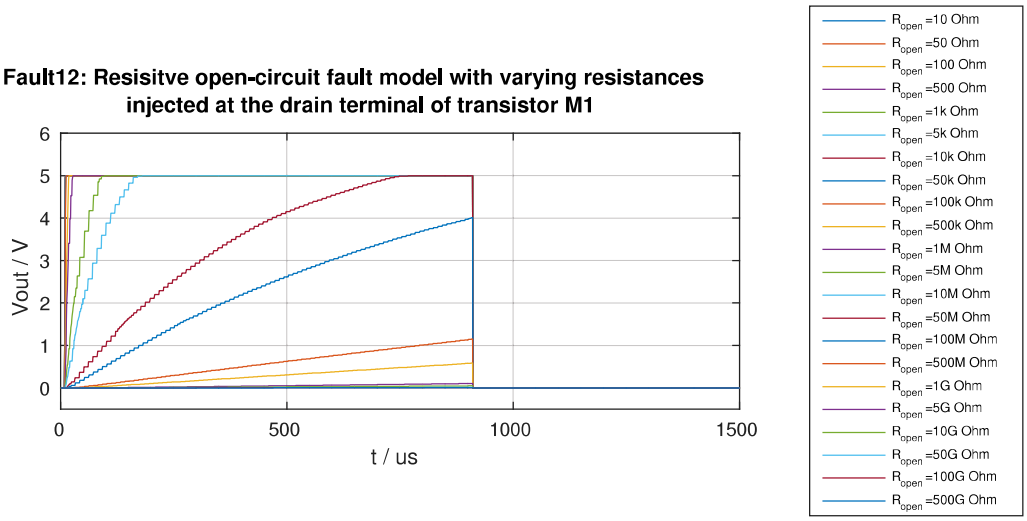


Figure 7.22: Circuit responses of the gate driver output-stage to soft stuck-open fault model of transistor M1.

0.8 and 0.2.

Screening

Screening is applied in order to identify non-influential soft faults. Using the extended elementary effects method, the total effect μ^* is used as indicator of influence. A small total effect indicates small influence on the circuit response and is eliminated from the fault list further analysis. The total effect (red) is plotted together with the linear and non-linear/interaction effects (blue) in fig. 7.24.

Regarding the stuck-short faults, fault number 9 has no effect (μ^* near 0). Fault number 1 has a small effect, compared to the other faults, which are located in the far right of the horizontal axis. Fault number 9 can be eliminated from the further analysis, due to the negligible total effect. Fault number 1 is also eliminated, due to its strictly non-linear output response which switches from nominal to hard failure for variable resistance values, see 7.21. Fault number 3 has a very small direct effect but a very strong non-linear/interaction effect. Keeping fault 3 for further sampling-based GSA analysis may introduce non-linearity to the circuit response but it is not eliminated for further analysis.

Regarding the stuck-open faults, none has a negligible total effect. However, faults with numbers 4, 10 and 16 have the smallest total effects, compared to the other faults. However, all faults are kept for the subsequent

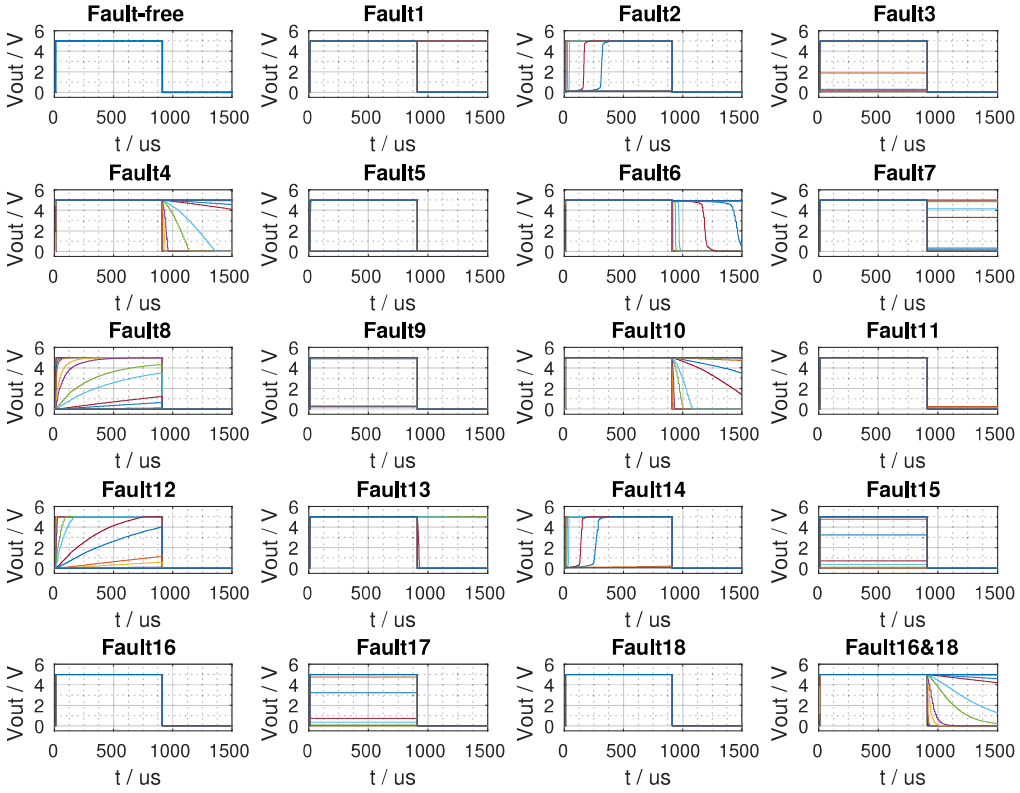
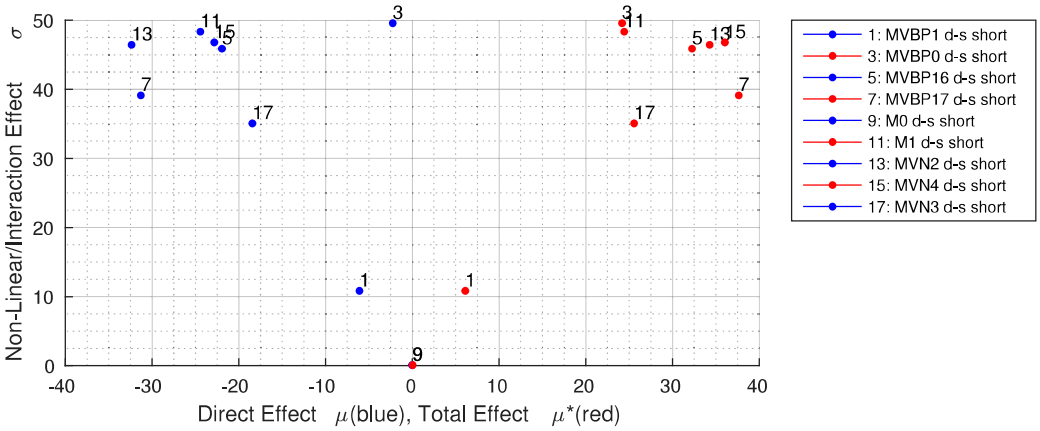


Figure 7.23: Circuit responses of the gate driver output-stage to soft stuck-short (uneven numbers) and stuck-open (even numbers) fault injection. Faults 16 and 18 have only an effect when occurring simultaneously.

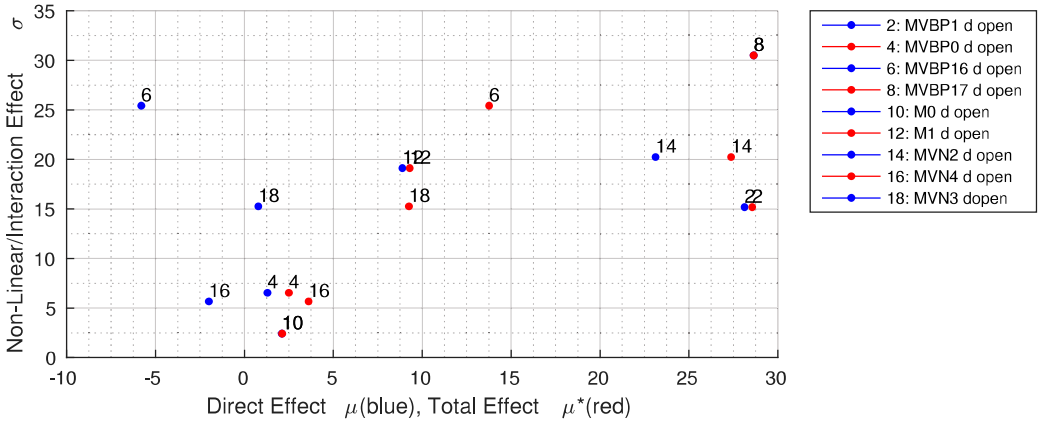
effect estimation.

Sampling-based global sensitivity analysis

The influential soft faults are sampled with uniform distribution within their soft failure resistance ranges. The circuit output responses are obtained and post-processed for mean normalized error calculation and standardization. The effect estimates and confidence intervals for soft stuck-open and stuck-short faults are plotted in 7.25 and 7.26, respectively. A dummy variable (fault 19) is added to the data set which is used as a reference to identify statistically significant effects. Any effect is statistically significant if its confidence boundary does not include the zero-reference line.



(a) Screening results for soft stuck-short faults.

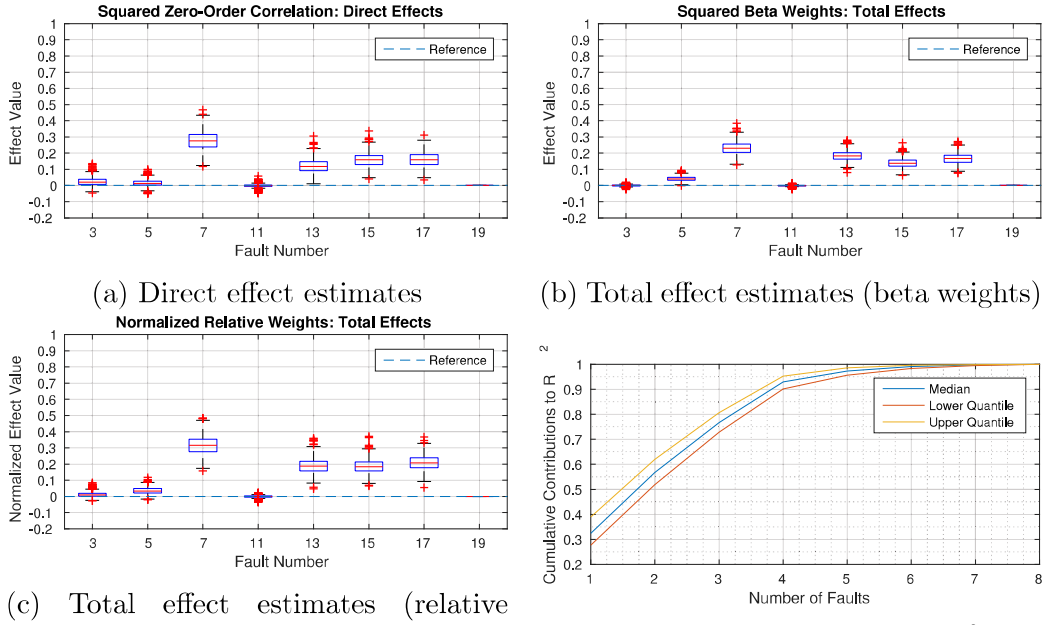


(b) Screening results for soft stuck-open faults.

Figure 7.24: Screening results for soft stuck-short and stuck-short faults. The total effects are read on the horizontal axis in increasing order.

Linear model assumption For the validation of the linear model assumption, the predicted output errors are drawn over the standardized output errors in scatter plots and the histograms of regression errors are shown, see fig. 7.27. For soft stuck-short and stuck-open faults, the linearity hypothesis can be confirmed due to a high coefficient of determination (more than 0.7) and normally distributed, mean 0, regression errors.

Effect estimates For soft stuck-short faults, the zero-order correlation in fig. 7.25 indicates the strongest direct effect for fault 7 and the weakest for fault 11. Faults 15 and 17 have very similar direct effects. This is due to the fact that both stuck-short faults belong to two in parallel connected transistors. All faults except faults 5 and 11 are statistically significant.



(a) Direct effect estimates

(b) Total effect estimates (beta weights)

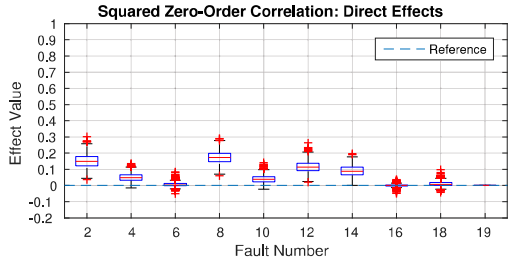
(c) Total effect estimates (relative weights)

(d) Contribution to model R^2

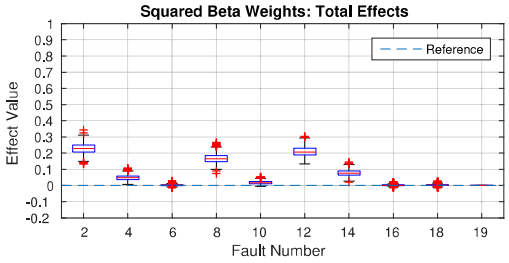
Figure 7.25: Direct and total effects and contributions of soft stuck-short faults to the model R^2.

Fault 3 is the weakest, yet statistically significant fault. Both, the squared beta weight in fig. 7.25b and the relative weight in fig. 7.25c indicate the strongest total effect for fault 7 and the weakest for fault 11. However, fault 3 is in contrast to the beta weights, statistically significant according to the relative weights. The relative weights of each soft stuck-short fault are used to calculate the cumulative contribution to the model R^2 within its confidence bounds. The cumulative contribution is plotted in fig. 7.25d over the number of soft stuck-short faults kept in the model with 8th being the dummy variable. It shows that four soft stuck-short faults account for more than 90% of the explained variance in the model R^2. By eliminating the three (out of seven) least influential soft stuck-short faults, the fault list for soft stuck-short faults can be reduced by still maintaining more than 90% of the total variability of the circuit output response.

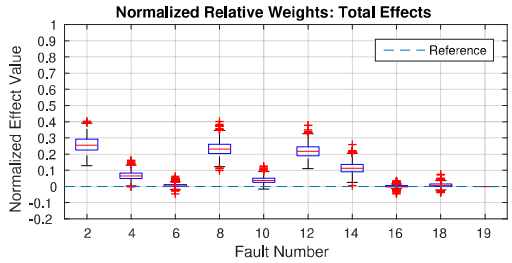
For soft stuck-open faults, the zero-order correlation in fig. 7.26 indicates the strongest direct effects for fault 8 and the weakest for fault 6 as well as for the soft stuck-open faults 16 and 18 of two parallel connected transistors. All faults, except the three weakest are statistically significant based on the zero-order correlation. Both, the squared beta weight in fig. 7.26b and the relative weight in fig. 7.26c indicate the strongest total effect for fault 2 and



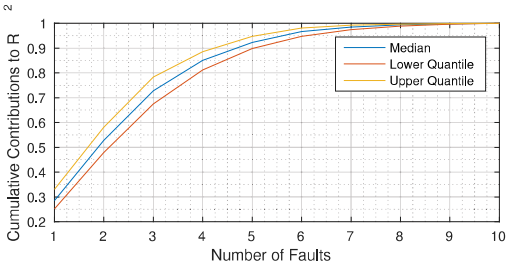
(a) Direct effect estimates



(b) Total effect estimates (beta weights)



(c) Total effect estimates (relative weights)



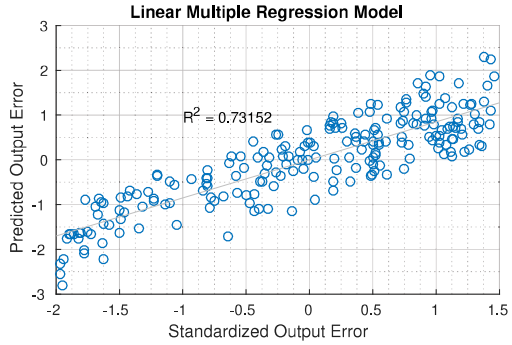
(d) Contribution to model R^2

Figure 7.26: Direct and total effects and contributions of soft stuck-open faults to the model R^2.

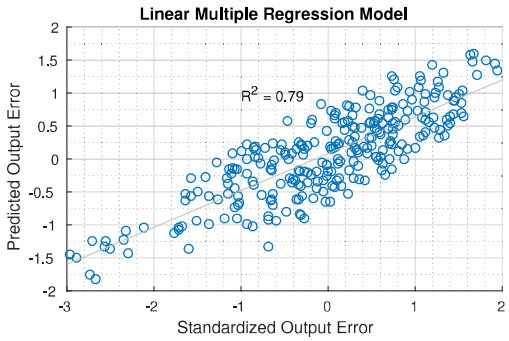
the weakest for faults 6, 16 and 18. Latter have a statistically insignificant total effect based on their squared beta weights and relative weights. The relative weights of each soft stuck-open fault are used to calculate the cumulative contribution to the model R^2. It shows that five soft stuck-open faults account for more than 90% of the explained variance in the model R^2. By eliminating the five (out of nine) least influential soft stuck-open faults, the fault list for soft stuck-open faults can be reduced by still maintaining more than 90% of the total variability of the circuit output response.

Discussion

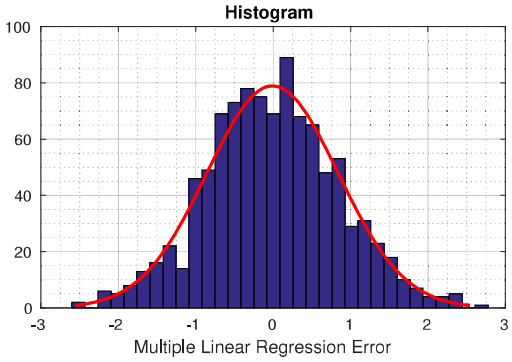
Generally, the determination of the soft failure ranges has a crucial impact on the validity of the linear model-based approach. It is not straightforward to determine the exact resistance values when a soft fault starts and ends if a limited number of samples used. If the resistance range is not determined accurately, the sensitivity analysis includes samples for hard error and nominal circuit responses among the soft errors. For screening, hard errors cause to cancel out the effects of other faults. Thus, they can be seen as non-influential faults. In the linear model-based approach, the hard and no error samples reduce the coefficient of determination R^2 of the model by adding



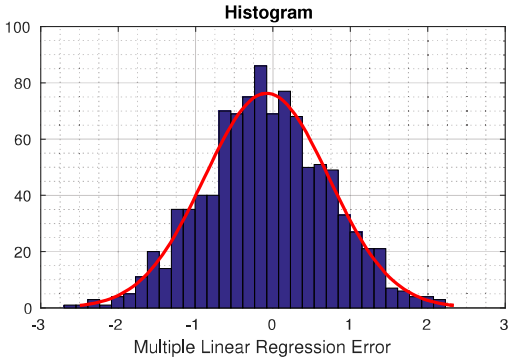
(a) Scatter plot and coefficient of determination for soft stuck-short faults



(b) Scatter plot and coefficient of determination for soft stuck-open faults



(c) Distribution of regression errors for soft stuck-short faults



(d) Distribution of regression errors for soft stuck-open faults

Figure 7.27: Validation of the linear regression model: coefficient of determination and regression errors.

non-linear samples to the scatter plot. Thus, this approach strongly relies on the correct determination of the resistance values which cause soft errors. In order to increase the accuracy in determining the soft failure ranges, more samples must be considered for each fault.

Chapter 8

Conclusion and outlook

In this work, the fault injection technique is investigated for pre-silicon safety-related functional verification of analog/mixed-signal circuits. The emphasis is on automotive systems and the functional safety standard for road vehicles, ISO 26262. The presented work goes beyond engineering judgement as in the classical product verification and safety analysis. Furthermore, it essentially facilitates to make safety for analog/mixed-signal circuits simulatable and thus, states a crucial feature of the pre-silicon verification activity for safety-related circuits.

The particular goal of this work in this context is to analyse and verify hardware designs for compliance with safety requirements in the presence of random hardware failures by simulation-based fault injection. In the remainder of this chapter, the work is concluded and future work is proposed.

8.1 Conclusion

In the presented work, the fault modelling and injection technique is integrated and automatized in the Cadence® Virtuoso® CAD tool. An object-oriented fault model library and fault injection algorithms are implemented using the electronic design automation language SKILL/SKILL++. The implementation facilitates device-dependent, layout-dependent and functional fault injection. A functional fault model is proposed which is capable of imitating an arbitrary number of failure modes at the interface signals of components and primitives. A dynamic fault injection approach is proposed which reduces simulation overhead by avoiding repeated netlisting and compilation for each fault because all faults in a circuit can be simulated by using a single netlist. The experimental results presented in this work are produced using the dynamic fault injection approach.

For functional safety verification and analysis, an FMEDA-oriented simulation-based approach for the evaluation of effects of random hardware faults is presented. The FMEDA is used to derive a verification plan including a fault list and safety-related verification criteria. For fault injection, functional fault models are used which model component failure modes in the circuit. This approach is particularly suited for fault injection in an early design phase, where the design implementation is yet not available but abstract models in a hardware description language. Identification of design risks by early fault injection can eventually have an impact on the final design implementation. This approach relies on the correctness of the FMEDA. However, FMEAs and FMEDAs are generally considered error-prone and ambiguous. Moreover, for simulating component failure modes by functional fault models it is not straightforward to determine the simulation fault coverage or other faults occurring inside the component which are not captured by the FMEDA. To address this, fault injection is also required for the implemented design at circuit level.

For efficient top-level verification, a hierarchy-oriented fault injection approach is presented. The goal of this approach is to increase the top-level verification confidence but avoid exhaustive top-level fault injection campaigns. The approach is based on extensive fault injection at component-level and a fault list reduction technique by utilizing a fault grouping algorithm. The fault grouping can process the component responses as waveforms for an arbitrary number of fault injection campaigns, component stimuli pattern and component output ports. It is based on a hierarchical clustering algorithm and cluster validation in order to identify the optimal reduced fault list. Experimental results show that depending on the size of the circuit, the fault list can be reduced by more than 82%. Using this approach, the fault simulation runs for top-level verification can be effectively reduced with no risk of omitting a safety-critical fault.

Besides catastrophic faults, soft (parametric) faults are considered by fault injection and a global sensitivity analysis. The goal of this approach is to identify the most influential soft faults in a component and eliminate the non-influential soft faults from further analysis. For this purpose, first a parameter screening is conducted in order to filter-out the non-influential faults. Subsequently, sampling-based global sensitivity analysis is conducted in order to rank the faults by influence based on their effect on the circuit response. Based on a significance test and confidence interval calculation, statistically significant faults are identified. Additionally, a metric for the simulation fault coverage for soft parametric faults is proposed which is based on the percentage of output's variability explained by the fault model parameters. Experimental results for a low-side gate driver output-stage circuit show that

from a total set of 18 soft stuck-open/-short fault models, the three most influential soft faults (from each fault type) account for more than 70% of the output variability. The most influential soft faults can potentially cause the most severe component failure. Therefore, for further analysis, all faults can be eliminated except the most influential ones.

8.2 Future work

The dynamic fault injection technique can be further exploited in order to reduce simulation time by the save/restore functionality of some SPICE-like simulators (e.g. Cadence® Spectre® Circuit Simulator [155]). The functionality avoids re-simulation of the typically computationally demanding transient analysis of the circuits initial start-up phase. However, it depends on keeping the netlist unchanged. Therefore, this functionality can be combined with the dynamic fault injection technique using the functional fault model in order to speed-up the fault simulation.

The FMEDA-oriented functional safety verification can be further improved by using simulation results from component-level fault injection campaigns utilizing the circuit-level netlist in order to build an FMEDA. This way, errors and ambiguities in the FMEDA can be mitigated and the verification confidence can be improved.

The hierarchical fault injection can be further improved by automatizing the steps required for component-level simulation test bench generation. Moreover, a classification algorithm [144] can be used in order to automatically identify the failure modes represented by the component-level circuit responses. This approach can subsequently be used to build an FMEDA.

The approach on the evaluation of soft (parametric) faults can be used in order to provide a quantitative measure for the simulation fault coverage of the safety-related verification. Additionally, parametric failure modes of devices (e.g. capacitor value changed by a certain ± percentage) can be included or eliminated from further analysis due to their significant or insignificant effect on the circuit response. This can also be used to build the FMEDA which also comprises parametric failure modes of devices.

List of Publications

Conference Papers

Karaca, O., Kirscher, J., Laroche, A., Tributsch, A., Maurer, L. and Pelz, G. (2016). Fault grouping for fault injection based simulation of AMS circuits in the context of functional safety. In *13th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, pages 1-4. IEEE Conference Publications.

Simon, S., Karaca, O., Kirscher, J., Rath, A., Pelz, G. and Maurer, L. (2016). Safety-oriented mixed-signal verification of automotive power devices in a UVM environment. In *13th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, pages 1-4. IEEE Conference Publications. (**Runner-Up of the EDA Competition Award**)

Karaca, O., Kirscher, J., Maurer, L. and Pelz, G. (2014). Towards simulation based evaluation of safety goal violations in automotive systems. In *Forum on Specification and Design Languages (FDL)*, volume 978-2-9530504-9-3, pages 1-8. IEEE Conference Publications. (**Winner of the Best Paper Award**)

Books and Journals

Karaca, O., Kirscher, J., Maurer, L. and Pelz, G. (2016). Towards simulation-based evaluation of safety goal violations in automotive systems. In *Languages, Design Methods, and Tools for Electronic System Design*, pages 23-40. Springer.

Karaca, O., Kappeler, F., Waldau, D., Rackles, J. and Kennel, R. (2014). Eigenmode analysis of a multiresonant wireless energy transfer system. In

Transactions on Industrial Electronics, pages 4134-4141. IEEE Journals & Magazines.

Patents

Karaca, O., Kirscher, J. and Pelz, G. (2015). Failure Sensitivity Analysis. Filed patent application at *US Patent Office*.

Colloquium

Karaca, O. (2015). Stochastic fault simualtions for verification of automotive functional safety. In *PhD Seminar at the Institute Electronic Components and Integrated Circuits at the Bundeswehr University Munich*.

Karaca, O. (2014). Simulation based evaluation of safety goal violations in analogue and mixed-signal automotive systems. In *PhD Seminar at the Institute Electronic Components and Integrated Circuits at the Bundeswehr University Munich*.

Appendix A

Algorithms for fault injection automation using the Cadence® OpenAccess database

Listing A.1: Algorithm of the main routine for fault injection into instances (primitives) in a schematic in Cadence SKILL/SKILL++.

```
1  ;----FAULT INJECTION routine
2  ;1. identify the instance by common cellView
3  ;2. create an object corresponding to the instance
4  ;3. run method faultInjection of the object
5  (foreach cell uniqueCellViews
6          ;----Find all the instances which origin from the same
               cellView
7          insts = (setof instance cV~>instances instance~>master~>
               cellView==cell);
8          instNames = insts~>name;
9          inst = car(insts);
10         ;----Identify the device for one of the instances and
               pass the
11         ;corresponding object to a variable
12         obj = deviceIdentification(inst cV);
13         set_goldenCellViewID(obj cV);
14         set_switchCellViewMode(obj switchCellViewMode);
15         ;----Instance-wise fault injection routine
16         (foreach instName instNames
17                 set_instName(obj instName);
18                 faultInjection(obj); is a method of object obj
19         );foreach
20 );foreach
```

Listing A.2: Master class of the fault model library and generic fault injection method in Cadence SKILL/SKILL++.

```
;-- classMasterFault: a master fault model from which all other faults are derived
defclass( classMasterFault
        ( standardObject )
        (
                ( instID
                        @initarg instID
                        @reader get_instID
                        @writer set_instID
                )
                ( instName
                        @initarg instName
                        @reader get_instName
                        @writer set_instName
                )
                ( cellViewID
                        @initarg cellViewID
                        @reader get_cellViewID
                        @writer set_cellViewID
                )
                ( goldenCellViewID
                        @initarg goldenCellViewID
                        @reader get_goldenCellViewID
                        @writer set_goldenCellViewID
                )
                ( switchCellViewMode
                        @initarg switchCellViewMode
                        @reader get_switchCellViewMode
                        @writer set_switchCellViewMode
                )
        );
);
;-- Generic method for fault injection
defgeneric( faultInjection ( obj )
        printf("faultInjection generic function for %s\n" obj~>instID~>name);
);
```

Listing A.3: Child class of the fault model library for functional fault injection into FETs in Cadence SKILL/SKILL++.

```
;-- classFETFunctional: stuck-short and stuck-open of FETs
defclass( classFETFunctional
        ( classMasterFault )
        (
                ( openRes
                        @initarg openRes
                        @reader get_openRes
                        @writer set_openRes
                )
                ( shortRes
                        @initarg shortRes
```

```
12                          @reader get_shortRes
13                          @writer set_shortRes
14                  )
15          );
16  );
17  ;----Method of object to inject a functional fault to a FET
18  defmethod( faultInjection ((obj classFETFunctional))
19          printf("A method of object : classFETFunctional for
                instance %s\n" get_instID(obj)~>name);
20          ;----OPEN drain terminal
21          set_cellViewID( obj switchCellViews(
                get_switchCellViewMode(obj), get_goldenCellViewID(obj
                )));
22          set_instID( obj dbFindAnyInstByName(get_cellViewID(obj)
                get_instName(obj)));
23          instTerm = (car (setof x get_instID(obj)~>instTerms x~>
                name=="d"));
24          instTermOpen(instTerm, get_cellViewID(obj));
25          netlister(get_switchCellViewMode(obj));
26          ;----SHORT drain terminal to source terminal
27          set_cellViewID( obj switchCellViews(
                get_switchCellViewMode(obj), get_goldenCellViewID(obj
                )));
28          set_instID( obj dbFindAnyInstByName(get_cellViewID(obj)
                get_instName(obj)));
29          instTermA = (car (setof x get_instID(obj)~>instTerms x~>
                name=="d"));
30          instTermB = (car (setof x get_instID(obj)~>instTerms x~>
                name=="s"));
31          instTermShort(instTermA, instTermB, get_cellViewID(obj))
                ;
32          netlister(get_switchCellViewMode(obj));
33  );
```

Listing A.4: Procedure to insert a resistive short-circuit fault model between two terminals of an instance in Cadence SKILL/SKILL++.

```
1   ;----Resistive short between two instTerm of same instance:
        instTermA and instTermB
2   procedure(instTermShort(instTermA, instTermB, cV)
3           ;--- Short-circuit resistance
4           fmInstName = "res"; fault model name
5           fmLibName = "analogLib"; fault model lib name
6           fmParamName = "r"; fault model parameter name
7           fmParamValue = "220";
8           ;---- POSITION and instantiate and connect fault model
9           initFaultInstPos(cV) setFaultInstPos()
10          instShortID = dbOpenCellViewByType( fmLibName fmInstName
                "symbol" "" 'w )
11          sprintf(fmInstName "fm_short__%s__%s__%s" instTermA~>
                inst~>name instTermA~>name instTermB~>name)
12          instShort = schCreateInst( cV instShortID fmInstName
                feverit.okFaultInstPos->x:feverit.okFaultInstPos->y "
                R0" );
```

```
13          feverit.insts_fm = cons(instShort feverit.insts_fm)
14          dbCreateConnByName(instTermA~>net instShort "PLUS")
15          dbCreateConnByName(instTermB~>net instShort "MINUS")
16          dbSetConnCurrent(cV)
17          ;--- Declare design variables for short-circuit
18          sprintf(cdfName "%s__res" fmInstName)
19          cdfgData = cdfGetInstCDF(instShort) get(cdfgData
                fmParamName)~>value = fmParamValue
20          dbSetConnCurrent(cV)
21          dbSave(cV);
22  );
```

Listing A.5: Procedures to calculate the positioning of fault models in the schematic in Cadence SKILL/SKILL++.

```
1   ;-- Calculate next position
2   procedure(setFaultInstPos()
3           feverit.okFaultInstPos->count = feverit.okFaultInstPos->
                count +1
4           feverit.okFaultInstPos->x = feverit.okFaultInstPos->x -
                1
5           (if (equal (modulo feverit.okFaultInstPos->count 10) 0)
                then
6                   feverit.okFaultInstPos->y = feverit.
                        okFaultInstPos->y - 1.5
7                   feverit.okFaultInstPos->x = feverit.
                        okFaultInstPos->x + 10
8           );if
9   );procedure
10  ;-- Initialize xy positioning values of fault instances
11  procedure(initFaultInstPos(cellview)
12          feverit.okFaultInstPos->bBox_design = cellview~>bBox
13          xspan_bBox_design = nth(0 feverit.okFaultInstPos->
                bBox_design)
14          yspan_bBox_design = nth(1 feverit.okFaultInstPos->
                bBox_design)
15          xstart_bBox_design = nth(0 xspan_bBox_design)
16          xstart_bBox_design = xstart_bBox_design - 1
17          ystart_bBox_design = nth(0 yspan_bBox_design)
18          feverit.okFaultInstPos->x = xstart_bBox_design
19          feverit.okFaultInstPos->y = ystart_bBox_design
20  );procedure
```

Listing A.6: Procedure to calculate the parasitic threshold values for fault injection.

```
1           ;-- pCaps THRESHOLD calculation routine
2           ;-------------------------------------
3           pCapInsts = setof(inst cV~>instances inst~>cellName=="
                pcapacitor");
4           pCapValues = CCFflattenList(pCapInsts~>prop~>value);
5           ;pCapSum = (apply 'plus pCapValues);
6           pCapSum = 0;
7           parasiticThreshold.help = pCapValues;
```

```
8         unt = length(pCapValues)-1;
9         ia = 0;
10        (while ( ia <= unt )
11                pCapSum = pCapSum + (nth ia++ pCapValues);
12        );
13        pCapRel='(nil);
14        pCapRelSorted='(nil);
15        pCapRelCumsum = '(nil);
16        (foreach pCap pCapValues
17                pCapRel = cons(pCap/pCapSum pCapRel);
18        );
19        pCapRel = remd(nil pCapRel);
20        ;-- SORT relative likelihoods & determine threshold
21        pCapRelSorted = sort(pCapRel 'greaterp);
22        pCapRelCumsum = cons((nth 0 pCapRelSorted) pCapRelCumsum
              );
23        pCapRelCumsum = remd(nil pCapRelCumsum);
24        ib = 0;
25        icum = (nth 0 pCapRelSorted);
26        unt = length(pCapRelSorted)-1;
27        (while ( ib <= unt )
28                icum = icum + (nth ib++ pCapRelSorted);
29                pCapRelCumsum = cons(icum pCapRelCumsum);
30        );
31        Nsubset = (length (exists x pCapRelCumsum x<
              shortLikelihood));
32        pCapValuesSorted = sort(pCapValues 'greaterp);
33        pCapthresholdLargerThanOrEqual = (nth Nsubset
              pCapValuesSorted);
34        ;--------------------------------------
35
36        ;---- pRes THRESHOLD calculation routine
37        ;--------------------------------------
38        ;.... same routine for presistor
39        pResthresholdLargerThanOrEqual = (nth Nsubset
              pResaluesSorted);
40        ;--------------------------------------
```

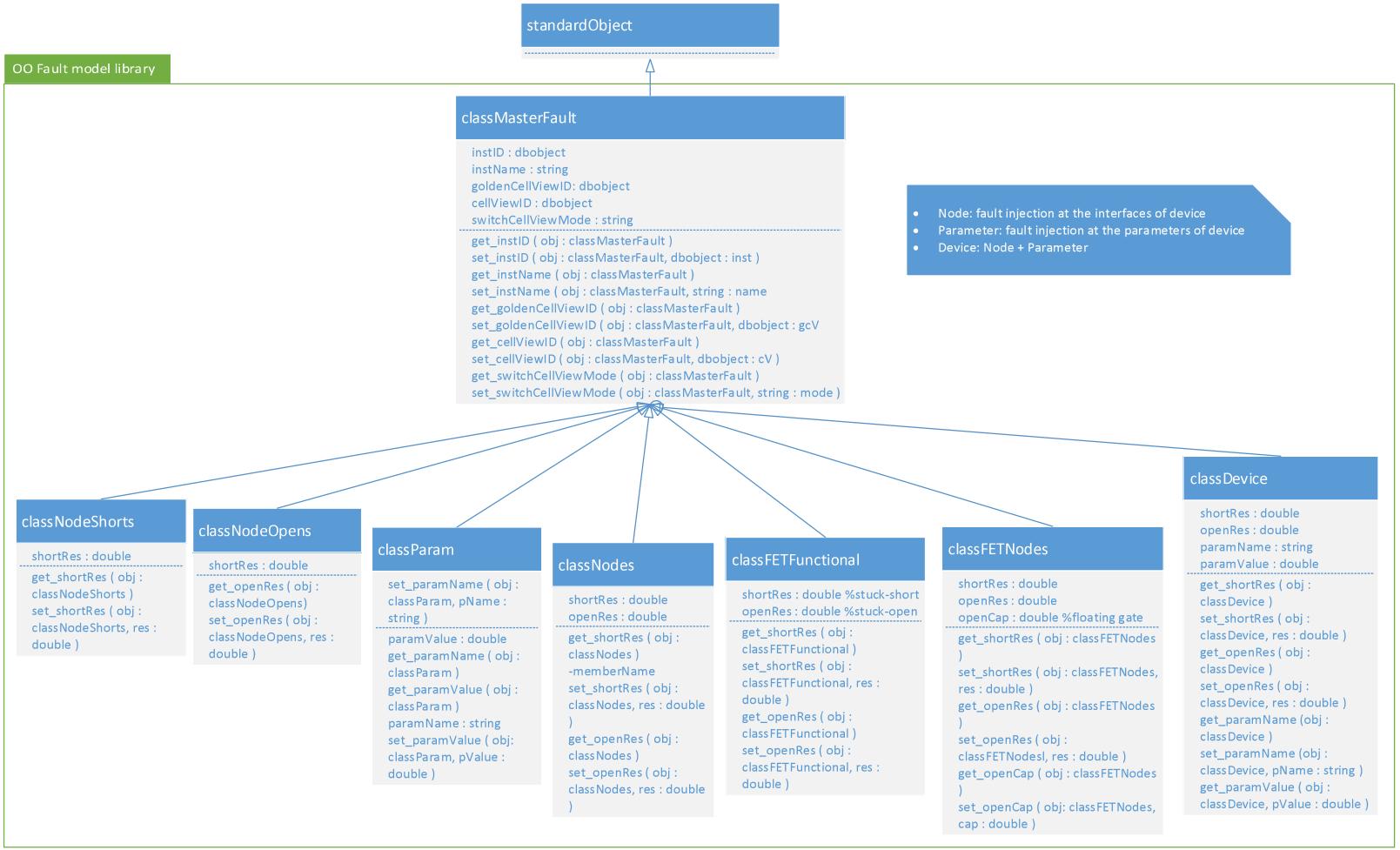


Figure A.1: UML class diagram of the object-oriented fault model library.

Appendix B

Algorithms for hierarchical fault injection

Listing B.1: Algorithm to generate a combined distance matrix with respect to component stimuli and outputs in Matlab.

```
%% Calculate combined euclidean distance matrix
% Format of input cell array X = {SxF cell} (length = O)
% S: stimuli, F: faults, O: outputs
DX=[]; d=0;
for j=1:size(X,2)
    for s=1:size(X{j},1)
        DX_col=[];
        for fdim=1:fdim_max
            X_fdim=(X{j}{s,fdim})'; DX_col=[DX_col,X_fdim];
            for locdim=1:fdim_max
                    X_floc=(X{j}{s,locdim})';
                    dx(fdim,locdim)=sqrt((X_fdim-X_floc)'*(
                        X_fdim-X_floc))^2;
            end;
        end;
        DX{j,s}=dx;
    end;
end;
for j=1:size(DX,1)
    for s=1:size(DX,2)
        d=d+DX{j,s}; % Additive squared euclidean distance
    end;
end;
d=sqrt(d);% Calculate euclidean distance
D=d./(size(DX,1)*size(DX,2)); %Adjust range
```

Listing B.2: Algorithm for hierarchical clustering using Ward's linkage function and dendrogram plot in Matlab.

```
%% Hierarchical clustering algorithm using Ward's linkage
%% function and dendrogram plot.
% Input: combined euclidean distance matrix D
```

```
link='ward';
tree=linkage(D,link);
leafOrder=optimalleaforder(tree,D);
hfig=figure(2)
H=dendrogram(tree,0,'Reorder',leafOrder,'orientation','top');
xlabel('Fault Number');
ylabel('Ward Linkage')
title('Hierarchical Clustering Scheme')
set(H,'LineWidth',2.3)
xlhand=get(gca,'xaxis');
set(xlhand,'fontsize',6.0)
set(hfig,'Position',[500 500 900 400])
```

Listing B.3: Algorithm for within sum of squared errors calculation in Matlab.

```
%% Calculation of within sum of squared errors
% Input: combined euclidean distance matrix D
[X,eigvals]=cmdscale(D);
D=dist(X');
SSE_comp=[]; CSSE_comp=[];
Kmax = 20;
ik=0;
for k=1:Kmax
    ik=ik+1;
    T=clusterdata(X,'linkage',link, 'linkage','ward', 'maxclust
        ',k);
    w(ik) = sum((grpstats(T, T, 'numel')-1).* sum(grpstats(X, T,
        'var'), 2));
end;
```

Listing B.4: Algorithm for cluster validation using Silhouette criterion and plotting the optimal number of fault groups in Matlab.

```
%% Cluster validation using Silhouette criterion
% Input: coordinate matrix X (=cmdscale(D))
myfunc=@(X,K) clusterdata(X,'linkage',link,'linkage','ward','
    maxclust',K);
eva=evalclusters(X,myfunc,'Silhouette','Klist',[1:Kmax]);
hfig=figure
plot(eva); set(hfig,'Position',[500 500 width height]); hold on;
plot([eva.OptimalK eva.OptimalK],[min(eva.CriterionValues) 1], '
    r--')
grid on;
text(eva.OptimalK,max(eva.CriterionValues)-(max(eva.
    CriterionValues)-min(eva.CriterionValues))/2,strcat('Kopt=',
    num2str(eva.OptimalK)))
ax=gca;
ax.XTick=[1:Kmax]; siopt=eva.OptimalK; xlim([2 Kmax])
title('Silhouette Cluster Validation')
```

Listing B.5: Algorithm for cluster validation using Davies Bouldin criterion and plotting the optimal number of fault groups in Matlab.

```
%% Cluster validation using Davies Bouldin criterion
% Input: coordinate matrix X (=cmdscale(D))
myfunc=@(X,K) clusterdata(X,'linkage',link,'linkage','ward','
    maxclust',K);
eva=evalclusters(X,myfunc,'DaviesBouldin','Klist',[1:Kmax]);
hfig=figure; plot(eva); set(hfig,'Position',[500 500 width
    height])
hold on
plot([eva.OptimalK eva.OptimalK],[min(eva.CriterionValues) max(
    eva.CriterionValues)], 'r--')
grid on;
text(eva.OptimalK,(max(eva.CriterionValues)-min(eva.
    CriterionValues))/2,strcat('Kopt=',num2str(eva.OptimalK)))
ax=gca; ax.XTick=[1:Kmax]; dbopt=eva.OptimalK; xlim([2 Kmax])
title('Davies-Bouldin Cluster Validation')
```

Listing B.6: Algorithm for cluster validation using DUNN criterion and plotting the optimal number of fault groups in Matlab.

```
%% Cluster validation using DUNN criterion
% Input: coordinate matrix X (=cmdscale(D))
DUNN=[];
for i=2:Kmax
    H=clusterdata(X,'linkage',link,'linkage','ward','maxclust',i
        );
    DUNN(i)=dunns(i,D,H)
end;
[~,dunnopt]=max(DUNN);
hfig=figure; plot(DUNN,'bo-'); [~,ind]=max(DUNN);
set(hfig,'Position',[500 500 width height]); hold on;
plot([ind ind],[min(DUNN) max(DUNN)], 'r--'); grid on; hold off;
text(ind,(max(DUNN)+min(DUNN))/2,strcat('Kopt=',num2str(ind)))
xlabel('Number of Clusters')
ylabel('DUNN Values')
ax=gca; ax.XTick=[1:Kmax];xlim([2 Kmax])
title('DUNN Cluster Validation')
```

Listing B.7: Algorithm for cluster validation using elbow criterion and plotting the optimal number of fault groups in Matlab.

```
%% Cluster validation with elbow method (within sum of squared
    errors)
Ksets=[siopt,dbopt,dunnopt]
optK=mode(Ksets); % most frequent Kopt as reference
for i=1:Kmax
    w_rel(i)=w(i)/w(1);
end;
hfig=figure; set(hfig,'Position',[500 500 width height])
plot(w_rel,'bo-'); xlabel('Number of Clusters'); ylabel('
    Relative SSE')
grid on; xlim([2 Kmax]); hold on;
plot([optK optK],[min(w_rel) max(w_rel(2:Kmax))], 'r--'); grid
    on;
```

```
hold off;
text(ind,(max(w_rel(2:Kmax))+min(w_rel))/2,strcat('Error = ',
    num2str(w_rel(ind)*100,'%8.2e'),'%'))
ax=gca; ax.XTick=[1:Kmax]
title('SSE Cluster Validation')
```

Listing B.8: Algorithm for calculating the representative faults in Matlab.

```
%% Calculation of representative faults for medoid and
%% worst-case criterion
% Input T: vector of size 1xF comprising respective cluster
   number
T=cluster(tree,'maxclust',optK);
d_meds=[];
d_worst=[];
for ck=1:optK
    obj=find(C==ck); DCk=[]; WCk=[];
    for x=1:size(obj)
        for y=1:size(obj)
            DCk(x,y)=D(obj(x),obj(y));
        end;
        WCk(x)=D(obj(x),1);
        d_meds(obj(x))=sum(sum(DCk));
        d_worst(obj(x))=D(obj(x),1);
    end;
    % Medoid criterion
    totalDist=mean(DCk,1);
    [MCk,i]=min(totalDist);
    mmedoid(ck)=obj(i);
        % Worst-case criterion
    [~,i]=max(WCk);
    mworst(ck)=obj(i);
end;
reps=[[1:optK]', mmedoid', mworst']; % list of representatives
```

Appendix C

Algorithms for the evaluation of soft faults

Listing C.1: Algorithm for calculating relative weights with confidence intervals on bootstrapped data in Matlab.

```
%% Calculation of relative weights with confidence intervals on
%% bootstrapped data in Matlab
BSTRP=1000; % Bootstrap size
data=[X,Y]; % data set
K=size(X,2); W=K+1; Rrw=[]; YDATA=[]; DATA=[];
for bstrp=1:BSTRP
    DATA{bstrp}=datasample(data,size(data,1));
end;
EPSILON=[];
for bstrp=1:BSTRP
    D=DATA{bstrp};
    X=D(:,1:W); Y=D(:,W+1);
    [U,S,V]=svd(X,'econ');
    Z=U*V'; beta=inv(Z'*Z)*Z'*Y; L=inv(Z'*Z)*Z'*(X);
    epsilon=(L.^2)*beta.^2;
    epsilon_norm=(1/sum(epsilon))*epsilon(1:end);
    EPSILON(:,bstrp)=epsilon_norm;
    [B0,I0]=sort(epsilon_norm,'descend');
    varexplainedrl=cumsum(B0)./sum(B0);
    VAREXPLRL(:,bstrp)=varexplainedrl;
end;
```

Listing C.2: Algorithm for calculating correlation with confidence intervals on bootstrapped data in Matlab.

```
%% Calculation of correlation with confidence intervals on
%% bootstrapped data in Matlab
BSTRP=1000; % Bootstrap size
data=[X,Y]; % data set
K=size(X,2); W=K+1; Rrw=[]; YDATA=[]; DATA=[];
for bstrp=1:BSTRP
    DATA{bstrp}=datasample(data,size(data,1));
```

```
end;
EPSILON=[];
for bstrp=1:BSTRP
    b = corr(X,Y); b_norm=abs(b).^2;
    [b0,I0]=sort(b_norm,'descend');
    varexplainedco=cumsum(b0)./sum(b0);
    CO(:,bstrp)=b_norm;
    VAREXPLCO(:,bstrp)=varexplainedco;
end;
```

Listing C.3: Algorithm for calculating beta weights with confidence intervals on bootstrapped data in Matlab.

```
%% Calculation of beta weights with confidence intervals on
%% bootstrapped data in Matlab
BSTRP=1000; % Bootstrap size
data=[X,Y]; % data set
K=size(X,2); W=K+1; Rrw=[]; YDATA=[]; DATA=[];
for bstrp=1:BSTRP
    DATA{bstrp}=datasample(data,size(data,1));
end;
EPSILON=[];
for bstrp=1:BSTRP
    X=[ones(size(X,1), 1) X];
    b = regress(Y,X); b=b(2:end); b_norm=abs(b).^2;
    [b0,I0]=sort(b_norm,'descend');
    varexplainedb=cumsum(b0)./sum(b0);
    BETA(:,bstrp)=b_norm;
    [B0,I0]=sort(b_norm,'descend');
    varexplainedb=cumsum(B0)./sum(B0);
    VAREXPLB(:,bstrp)=varexplainedb;
end;
```

Bibliography

[1] M. Harrant. *Application-driven Post-Silicon Verification of Automotive Smart Power ICs*. Shaker, 2015.

[2] P. Rashinkar, P. Paterson, and L. Singh. *System-on-a-chip verification: methodology and techniques*. Springer Science & Business Media, 2007.

[3] G. Di Giacomo and G. Di Giacomo. *Reliability of electronic packages and semiconductor devices*. McGraw-Hill New York, 1997.

[4] I. 26262. Iso 26262– road vehicles- functional safety, 2011.

[5] S. Anis, M. H. El-mahlawy, M. E. Gadallah, and E. A. El-samahy. Parametric fault detection of analogue circuits. *International Journal of Computer Applications*, 96(9):14–23, 2014.

[6] W. L. Martinez and A. R. Martinez. *Computational statistics handbook with MATLAB*, volume 22. CRC press, 2007.

[7] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.

[8] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[9] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. 1973.

[10] B. J. Baliga. An overview of smart power technology. *IEEE Transactions on Electron devices*, 38(7):1568–1575, 1991.

[11] B. J. Baliga. *Fundamentals of power semiconductor devices*. Springer Science & Business Media, 2010.

[12] G. Moore. Moores law. *Electronics Magazine*, 38(8), 1965.

[13] G.-Q. Zhang and A. V. Roosmalen. *More than moore*. Springer, 2009.

[14] A. Molina and O. Cadenas. Functional verification: approaches and challenges. *Latin American applied research*, 37(1):65–69, 2007.

[15] A. Adir, S. Copty, S. Landa, A. Nahir, G. Shurek, A. Ziv, C. Meissner, and J. Schumann. A unified methodology for pre-silicon verification and post-silicon validation. In *2011 Design, Automation & Test in Europe*, pages 1–6. IEEE, 2011.

[16] E. Barke, D. Grabowski, H. Graeb, L. Hedrich, S. Heinen, R. Popp, S. Steinhorst, and Y. Wang. Formal approaches to analog circuit verification. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 724–729. European Design and Automation Association, 2009.

[17] I. Wagner and V. Bertacco. The verification universe. In *Post-Silicon and Runtime Verification for Modern Processors*, pages 13–42. Springer, 2011.

[18] M. H. Zaki, S. Tahar, and G. Bois. Formal verification of analog and mixed signal designs: A survey. *Microelectronics Journal*, 39(12):1395–1404, 2008.

[19] G. D. Micheli. *Synthesis and optimization of digital circuits*. McGraw-Hill Higher Education, 1994.

[20] P. Coussy and A. Morawiec. *High-level synthesis*, volume 1. Springer, 2010.

[21] G. G. Gielen and R. A. Rutenbar. Computer-aided design of analog and mixed-signal integrated circuits. *Proceedings of the IEEE*, 88(12):1825–1854, 2000.

[22] S. Devadas, A. Ghosh, and K. Keutzer. An observability-based code coverage metric for functional simulation. In *Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design*, pages 418–425. IEEE Computer Society, 1997.

[23] Y. Li, W. Wu, L. Hou, and H. Cheng. A study on the assertion-based verification of digital ic. In *2009 Second International Conference on Information and Computing Science*, volume 2, pages 25–28. IEEE, 2009.

[24] A. Piziali. *Functional verification coverage measurement and analysis.* Springer Science & Business Media, 2007.

[25] Y. Naveh, M. Rimon, I. Jaeger, Y. Katz, M. Vinov, E. s Marcu, and G. Shurek. Constraint-based random stimuli generation for hardware verification. *AI magazine*, 28(3):13, 2007.

[26] C. Liang, G. Zhong, S. Huang, and B. Xia. Uvm-ams based subsystem verification of wireless power receiver soc. In *Solid-State and Integrated Circuit Technology (ICSICT), 2014 12th IEEE International Conference on*, pages 1–3. IEEE, 2014.

[27] S. R. Little. *Efficient modeling and verification of analog/mixed-signal circuits using labeled hybrid petri nets.* ProQuest, 2008.

[28] Y.-B. Sha, M.-S. Lee, and C.-N. J. Liu. On code coverage measurement for verilog-a. In *High-Level Design Validation and Test Workshop, 2004. Ninth IEEE International*, pages 115–120. IEEE, 2004.

[29] T. Nahhal and T. Dang. Test coverage for continuous and hybrid systems. In *International Conference on Computer Aided Verification*, pages 449–462. Springer, 2007.

[30] H. Chang and K. Kundert. Verification of complex analog and rf ic designs. *Proceedings of the IEEE*, 95(3):622–639, 2007.

[31] B. A. Antao and A. J. Brodersen. Behavioral simulation for analog system design verification. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 3(3):417–429, 1995.

[32] E. Christen and K. Bakalar. Vhdl-ams-a hardware description language for analog and mixed-signal applications. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 46(10):1263–1272, 1999.

[33] P. J. Ashenden. *The designer's guide to VHDL*, volume 3. Morgan Kaufmann, 2010.

[34] D. Thomas and P. Moorby. *The Verilog® Hardware Description Language.* Springer Science & Business Media, 2008.

[35] K. Kundert and O. Zinke. *The designers guide to Verilog-AMS.* Springer Science & Business Media, 2006.

[36] F. Pêcheux, C. Lallement, and A. Vachoux. Vhdl-ams and verilog-ams as alternative hardware description languages for efficient modeling of multidiscipline systems. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(2):204–225, 2005.

[37] R. A. Saleh, S.-J. Jou, and A. R. Newton. *Mixed-mode simulation and analog multilevel simulation*, volume 279. Springer Science & Business Media, 2013.

[38] R. A. Saleh, B. A. Antao, and J. Singh. Multilevel and mixed-domain simulation of analog circuits and systems. *IEEE transactions on computer-aided design of integrated circuits and systems*, 15(1):68–82, 1996.

[39] M. Stanisavljević, A. Schmid, and Y. Leblebici. Reliability, faults, and fault tolerance. In *Reliability of Nanoscale Circuits and Systems*, pages 7–18. Springer, 2011.

[40] M. Xie and C. D. Lai. Reliability analysis using an additive weibull model with bathtub-shaped failure rate function. *Reliability Engineering & System Safety*, 52(1):87–93, 1996.

[41] D. Siewiorek and R. Swarz. *Reliable Computer Systems: Design and Evaluatuion*. Digital Press, 2014.

[42] M. Blank, T. Glück, A. Kugi, and H.-P. Kreuter. Digital slew rate and s-shape control for smart power switches to reduce emi generation. *IEEE Transactions on Power Electronics*, 30(9):5170–5180, 2015.

[43] M. Cuviello, S. Dey, X. Bai, and Y. Zhao. Fault modeling and simulation for crosstalk in system-on-chip interconnects. In *Proceedings of the 1999 IEEE/ACM international conference on Computer-aided design*, pages 297–303. IEEE Press, 1999.

[44] M. Handbook. Electronic reliability design handbook. Technical report, MIL-HDBK-338, DoD, 1988.

[45] W.-S. Lee, D. L. Grosh, F. A. Tillman, and C. H. Lie. Fault tree analysis, methods, and applications - a review. *Reliability, IEEE Transactions on*, 34(3):194–203, 1985.

[46] M. Standard. Procedures for performing a failure mode, effects and criticality analysis. *MIL-STD-1629, November, AMSC Number N3074*, 1980.

[47] M.-C. Hsueh, T. K. Tsai, and R. K. Iyer. Fault injection techniques and tools. *Computer*, 30(4):75–82, Apr 1997.

[48] D. Lee and J. Na. A novel simulation fault injection method for dependability analysis. *IEEE Design Test of Computers*, 26(6):50–61, Nov 2009.

[49] R. Rosing, A. M. Richardson, and A. P. Dorey. A fault simulation methodology for mems. In *Design, Automation and Test in Europe Conference and Exhibition 2000. Proceedings*, pages 476–483, 2000.

[50] Z. Segall, D. Vrsalovic, D. Siewiorek, D. Ysskin, J. Kownacki, J. Barton, R. Dancey, A. Robinson, and T. Lin. Fiat-fault injection based automated testing environment. In *Fault-Tolerant Computing, 1995, Highlights from Twenty-Five Years., Twenty-Fifth International Symposium on*, page 394. IEEE, 1995.

[51] B. Rahbaran, A. Steininger, and T. Handl. Built-in fault injection in hardware-the fidyco example. In *Electronic Design, Test and Applications, Proceedings. DELTA 2004. Second IEEE International Workshop on*, pages 327–332. IEEE, 2004.

[52] P. Civera, L. Macchiarulo, M. Rebaudengo, M. S. Reorda, and M. Violante. An fpga-based approach for speeding-up fault injection campaigns on safety-critical circuits. *Journal of Electronic Testing*, 18(3):261–271, 2002.

[53] P. Civera, L. Macchiarulo, M. Rebaudengo, M. S. Reorda, and A. Violante. Exploiting fpga for accelerating fault injection experiments. In *On-Line Testing Workshop, 2001. Proceedings. Seventh International*, pages 9–13. IEEE, 2001.

[54] N. Bombieri, F. Fummi, and V. Guarnieri. Accelerating rtl fault simulation through rtl-to-tlm abstraction. In *Test Symposium (ETS), 2011 16th IEEE European*, pages 117–122, May 2011.

[55] S.-N. Ahmadian and S.-G. Miremadi. Fault injection in mixed-signal environment using behavioral fault modeling in verilog-a. In *Behavioral Modeling and Simulation Conference (BMAS), 2010 IEEE International*, pages 69–74. IEEE, 2010.

[56] C. Bolchini, A. Miele, and D. Sciuto. Fault models and injection strategies in systemc specifications. In *Digital System Design Architectures,*

Methods and Tools, 2008. DSD '08. 11th EUROMICRO Conference on, pages 88–95, Sept 2008.

[57] L. Grunske, P. Lindsay, N. Yatapanage, and K. Winter. An automated failure mode and effect analysis based on high-level design specification with behavior trees. In *Integrated Formal Methods*, pages 129–149. Springer, 2005.

[58] F. Ortmeier and G. Schellhorn. Formal fault tree analysis-practical experiences. *Electronic Notes in Theoretical Computer Science*, 185:139–151, 2007.

[59] A. L. Crouch. *Design-for-test for Digital IC's and Embedded Core Systems*, volume 1. The Rosen Publishing Group, 1999.

[60] A. Myaing and V. Dinavahi. Fpga-based real-time emulation of power electronic systems with detailed representation of device characteristics. In *Power and Energy Society General Meeting, 2011 IEEE*, pages 1–11. IEEE, 2011.

[61] H. Ziade, R. A. Ayoubi, R. Velazco, et al. A survey on fault injection techniques. *Int. Arab J. Inf. Technol.*, 1(2):171–186, 2004.

[62] P. Antognetti, G. Massobrio, and G. Massobrio. *Semiconductor device modeling with SPICE*. McGraw-Hill, Inc., 1993.

[63] M. Chaari, W. Ecker, T. Kruse, and B.-A. Tabacaru. Automation of failure propagation analysis through metamodeling and code generation. ITG/GI/GMM-Workshop Testmethoden und Zuverlässigkeit von Schaltungen und Systemen (TuZ), 2015.

[64] M. Hause et al. The sysml modelling language. In *Fifteenth European Systems Engineering Conference*, volume 9. Citeseer, 2006.

[65] J. A. Estefan et al. Survey of model-based systems engineering (mbse) methodologies. *Incose MBSE Focus Group*, 25(8), 2007.

[66] A. M. Richter. Modellgestützte safety–und reliability–analysen auf funk! onaler ebene für hardware. 2014.

[67] Y. Papadopoulos and J. A. McDermid. Hierarchically performed hazard origin and propagation studies. In *Computer Safety, Reliability and Security*, pages 139–152. Springer, 1999.

[68] P. Helle. Automatic sysml-based safety analysis. In *Proceedings of the 5th International Workshop on Model Based Architecting and Construction of Embedded Systems*, pages 19–24. ACM, 2012.

[69] H. Aljazzar, M. Fischer, L. Grunske, M. Kuntz, F. Leitner-Fischer, and S. Leue. Safety analysis of an airbag system using probabilistic fmea and probabilistic counterexamples. In *Quantitative Evaluation of Systems, 2009. QEST'09. Sixth International Conference on the*, pages 299–308. IEEE, 2009.

[70] M. Chaari, W. Ecker, C. Novello, B. A. Tabacaru, and T. Kruse. A model-based and simulation-assisted fmeda approach for safety-relevant e/e systems. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2015.

[71] S. Misera, H. T. Vierhaus, and A. Sieber. Fault injection techniques and their accelerated simulation in systemc. In *Digital System Design Architectures, Methods and Tools, 2007. DSD 2007. 10th Euromicro Conference on*, pages 587–595. IEEE, 2007.

[72] B. Charlot, S. Mir, E. F. Cota, M. Lubaszewski, and B. Courtois. Fault simulation of mems using hdls. In *Design, Test, and Microfabrication of MEMS/MOEMS*, pages 70–77. International Society for Optics and Photonics, 1999.

[73] T. Kirkland and M. MERCER. Automatic test pattern generation. 1988.

[74] R. Ubar, L. Juerimaegi, E. Orasson, and J. Raik. Scalable algorithm for structural fault collapsing in digital circuits. In *Very Large Scale Integration (VLSI-SoC), 2015 IFIP/IEEE International Conference on*, pages 171–176, Oct 2015.

[75] A. Prasad, V. D. Agrawal, and M. V. Atre. A new algorithm for global fault collapsing into equivalence and dominance sets. In *Test Conference, 2002. Proceedings. International*, pages 391–397. IEEE, 2002.

[76] L. Zhang, I. Ghosh, and M. Hsiao. Efficient sequential atpg for functional rtl circuits. In *null*, page 290. IEEE, 2003.

[77] V. D. Agrawal, A. Prasad, and M. V. Atre. Fault collapsing via functional dominance. In *International Test Conference*, pages 274–280, 2003.

[78] R.-K.-K. Sandireddy. *Hierarchical fault collapsing for logic circuits.* PhD thesis, 2005.

[79] J. C. Baraza, J. Gracia, D. Gil, and P. J. Gil. Improvement of fault injection techniques based on vhdl code modification. In *Tenth IEEE International High-Level Design Validation and Test Workshop, 2005.*, pages 19–26. IEEE, 2005.

[80] H. R. Zarandi, S. G. Miremadi, and A. Ejlali. Dependability analysis using a fault injection tool based on synthesizability of hdl models. In *Defect and Fault Tolerance in VLSI Systems, 2003. Proceedings. 18th IEEE International Symposium on*, pages 485–492. IEEE, 2003.

[81] A. Rohani and H. G. Kerkhoff. A technique for accelerating injection of transient faults in complex socs. In *Digital System Design (DSD), 2011 14th Euromicro Conference on*, pages 213–220, Aug 2011.

[82] K. S. G. M. S. Suma. Fault simulation of digital circuits at register transfer level. *International Journal of Computer Applications*, 30(7):1–5, September 2011.

[83] P. A. Thaker. *Register-Transfer Level Fault Modeling and Test Evaluation Technique for VLSI Circuits.* PhD thesis, The Department of Electrical and Computer Engineering of The George Washington University, May 2000.

[84] M. Karunaratne, A. Sagahayroon, and S. Prodhuturi. Rtl fault modeling. In *48th Midwest Symposium on Circuits and Systems, 2005.*, volume 2, pages 1717–1720, Aug 2005.

[85] U. Reinsalu, J. Raik, R. Ubar, and P. Ellervee. Fast rtl fault simulation using decision diagrams and bitwise set operations. In *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2011 IEEE International Symposium on*, pages 164–170, Oct 2011.

[86] J. R. Armstrong, F.-S. Lam, and P. C. Ward. Test generation and fault simulation for behavioral models. *Performance and Fault Modeling with VHDL*, pages 240–303, 1992.

[87] G. Buonanno, F. Ferrandi, L. Ferrandi, F. Fummi, and D. Sciuto. How an evolving fault model improves the behavioral test generation. In *VLSI, 1997. Proceedings. Seventh Great Lakes Symposium on*, pages 124–129. IEEE, 1997.

[88] E. Ferrandi, G. Ferrara, D. Sciuto, A. Fin, and F. Fummi. Functional test generation for behaviorally sequential models. In *Design, Automation and Test in Europe, 2001. Conference and Exhibition 2001. Proceedings*, pages 403–410. IEEE, 2001.

[89] E. Corno, G. Cumani, M. S. Reorda, and G. Squillero. An rt-level fault model with high gate level correlation. In *High-Level Design Validation and Test Workshop, 2000. Proceedings. IEEE International*, pages 3–8. IEEE, 2000.

[90] L. Cai and D. Gajski. Transaction level modeling: an overview. In *Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 19–24. ACM, 2003.

[91] E. G. Ulrich and T. Baker. The concurrent simulation of nearly identical digital networks. In *Papers on Twenty-five Years of Electronic Design Automation*, 25 years of DAC, pages 318–323, New York, NY, USA, 1988. ACM.

[92] A. Miczo. *Digital logic testing and simulation.* John Wiley & Sons, 2003.

[93] P. Agrawal and S. Bose. Concurrent fault simulation of circuits with both logic elements and functional circuits, April 30 1996. US Patent 5,513,339.

[94] M. Renovell, P. Huc, and Y. Bertrand. The concept of resistance interval: a new parametric model for realistic resistive bridging fault. In *VLSI Test Symposium, 1995. Proceedings., 13th IEEE*, pages 184–189, Apr 1995.

[95] M. Dalpasso, M. Favalli, P. Olivo, and B. Ricco. Parametric bridging fault characterization for the fault simulation of library-based ics. In *Test Conference, 1992. Proceedings., International*, pages 486–, Sep 1992.

[96] M. Dalpasso, M. Favalli, P. Olivo, and B. Ricco. Fault simulation of parametric bridging faults in cmos ic's. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(9):1403–1410, Sep 1993.

[97] D. Bhatta, I. Mukhopadhyay, S. Natarajan, P. Goteti, and B. Xue. Framework for analog test coverage. In *Quality Electronic Design*

(ISQED), 2013 14th International Symposium on, pages 468–475, March 2013.

[98] J. Parky, S. Madhavapeddiz, A. Paglieri, C. Barrz, and J. A. Abraham. Defect-based analog fault coverage analysis using mixed-mode fault simulation. In *Mixed-Signals, Sensors, and Systems Test Workshop, 2009. IMS3TW'09. IEEE 15th International*, pages 1–6. IEEE, 2009.

[99] K. Arabi and B. Kaminska. Parametric and catastrophic fault coverage of analog circuits in oscillation-test methodology. In *VLSI Test Symposium, 1997., 15th IEEE*, pages 166–171. IEEE, 1997.

[100] K. Arabi and B. Kaminska. Oscillation built-in self test (obist) scheme for functional and structural testing of analog and mixed-signal integrated circuits. In *Test Conference, 1997. Proceedings., International*, pages 786–795. IEEE, 1997.

[101] C. Sebeke, J. Teixeira, and M. Ohletz. Automatic fault extraction and simulation of layout realistic faults for integrated analogue circuits. In *Proceedings of the 1995 European conference on Design and Test*, page 464. IEEE Computer Society, 1995.

[102] R. Kondagunturi, E. Bradley, K. Maggard, and C. Stroud. Benchmark circuits for analog and mixed-signal testing. In *Southeastcon '99. Proceedings. IEEE*, pages 217–220, 1999.

[103] W. Maly. Realistic fault modeling for vlsi testing. In *Proceedings of the 24th ACM/IEEE Design Automation Conference*, pages 173–180. ACM, 1987.

[104] M. Soma. An experimental approach to analog fault models. In *Custom Integrated Circuits Conference, 1991., Proceedings of the IEEE 1991*, pages 13.6/1–13.6/4, May 1991.

[105] A. Meixner and W. Maly. Fault modeling for the testing of mixed integrated circuits. In *Test Conference, 1991, Proceedings., International*, page 564. IEEE, 1991.

[106] R. J. A. Harvey, A. M. D. Richardson, E. M. J. G. Bruls, and K. Baker. Analogue fault simulation based on layout dependent fault models. In *Test Conference, 1994. Proceedings., International*, pages 641–649, Oct 1994.

[107] M. Soma. Challenges in analog and mixed-signal fault models. *Circuits and Devices Magazine, IEEE*, 12(1):16–19, 1996.

[108] M. Sachdev and B. Atzema. Industrial relevance of analog ifa: a fact or a fiction. In *Test Conference, 1995. Proceedings., International*, pages 61–70, Oct 1995.

[109] W. H. Kao, C.-Y. Lo, M. Basel, and R. Singh. Parasitic extraction: current state of the art and future trends. *Proceedings of the IEEE*, 89(5):729–739, 2001.

[110] M. Parvathi, N. Vasantha, and K. S. Prasad. Fault model analysis by parasitic extraction method for embedded sram. *International Journal of Research in Engineering and Technology, IJRET*, 2013.

[111] G. Borgmann, C. Burmer, and S. Mézière. Improved parasitic fault modeling for automatic analog fault simulation. In *ISTFA 2012: Conference Proceedings from the 38th International Symposium for Testing and Failure Analysis: November 11-15, 2012, Phoenix Convention Center, Phoenix, Arizona, USA*, volume 1, page 281. ASM International, 2012.

[112] K. Saab, N. Ben-Hamida, and B. Kaminska. Parametric fault simulation and test vector generation. In *Proceedings of the conference on Design, automation and test in Europe*, pages 650–657. ACM, 2000.

[113] X. Li, J. Qin, and J. B. Bernstein. Compact modeling of mosfet wearout mechanisms for circuit-reliability simulation. *IEEE Transactions on Device and Materials Reliability*, 8(1):98–121, 2008.

[114] Z. R. Yang and M. Zwolinski. A methodology for statistical behavioral fault modeling. 1998.

[115] K. Huang, H. G. Stratigopoulos, and S. Mir. Bayesian fault diagnosis of rf circuits using nonparametric density estimation. In *2010 19th IEEE Asian Test Symposium*, pages 295–298, Dec 2010.

[116] K. Huang, H.-G. Stratigopoulos, S. Mir, C. Hora, Y. Xing, and B. Kruseman. Diagnosis of local spot defects in analog circuits. *Instrumentation and Measurement, IEEE Transactions on*, 61(10):2701–2712, 2012.

[117] E. W. Liu, H. C. Chang, and A. L. Sangiovanni-Vincentelli. Analog system verification in the presence of parasitics using behavioral simulation. In *Proceedings of the 30th international Design Automation Conference*, pages 159–163. ACM, 1993.

[118] E. Liu, W. Kao, E. Felt, and A. Sangiovanni-Vincentelli. Analog testability analysis and fault diagnosis using behavioral modeling. In *Proc. CICC*, pages 413–416, 1994.

[119] K. Garje, A. Kumar, S. Biswas, A. Banerjee, P. Srikanth, and S. Mukhopadhyay. Macromodel based fault simulation of linear circuits using parameter estimation. In *Industrial and Information Systems, 2008. ICIIS 2008. IEEE Region 10 and the Third international Conference on*, pages 1–6. IEEE, 2008.

[120] C.-Y. Pan and K.-T. Cheng. Fault macromodeling for analog/mixed-signal circuits. In *Test Conference, 1997. Proceedings., International*, pages 913–922, Nov 1997.

[121] M. Zwolinski, C. Chalk, and B. R. Wilkins. Analogue fault modelling and simulation for supply current monitoring. In *Proceedings of the 1996 European Conference on Design and Test*, EDTC '96, pages 547–, Washington, DC, USA, 1996. IEEE Computer Society.

[122] J. V. Calvano, V. C. Alves, M. S. Lubaszewski, and A. C. Mesquita. Fault models and compact test vectors for mos opamp circuits. In *Integrated Circuits and Systems Design, 2000. Proceedings. 13th Symposium on*, pages 289–294. IEEE, 2000.

[123] K. Kundert, H. Chang, D. Jefferies, G. Lamant, E. Malavasi, and F. Sendig. Design of mixed-signal systems-on-a-chip. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 19(12):1561–1571, 2000.

[124] M. Zwolinski, C. Chalk, and A. Perkins. Multi-level fault modeling of analog circuits. In *IEEE International Workshop on Behavioral Modeling and Simulation (BMAS), Washington DC, USA*, pages 107–113, 1997.

[125] P. R. Wilson, Y. Kiliç, J. N. Ross, M. Zwolinski, and A. D. Brown. Behavioural modelling of operational amplifier faults using vhdl-ams. In *Design, Automation and Test in Europe Conference and Exhibition, 2002. Proceedings*, page 1133. IEEE, 2002.

[126] P. R. Wilson, Y. Kilic, J. N. Ross, M. Zwolinski, and A. D. Brown. Behavioural modelling of operational amplifier faults using analogue hardware description languages. In *Behavioral Modeling and Simulation, 2001. BMAS 2001. Proceedings of the Fifth IEEE International Workshop on*, pages 106–112, 2001.

[127] Y. Kilic and M. Zwolinski. Behavioural fault modelling using vhdl-ams and slow transient analysis with hamster simulator to speed-up analogue fault simulation. 2002.

[128] R. Leveugle and A. Ammari. Early seu fault injection in digital, analog and mixed signal circuits: A global flow. In *Proceedings of the conference on Design, automation and test in Europe-Volume 1*, page 10590. IEEE Computer Society, 2004.

[129] Z. Xueqian, Z. Zhenyu, Z. Minxuan, and L. Shaoqing. Verilog-a based implementation for coupled model of single event transients in look-up table technique. In *ASIC, 2009. ASICON'09. IEEE 8th International Conference on*, pages 666–669. IEEE, 2009.

[130] V. Litovski, M. Andrejević, and M. Zwolinski. Behavioural modelling, simulation, test and diagnosis of mems using anns. In *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, pages 5182–5185. IEEE, 2005.

[131] B. Straube, W. Vermeiren, and V. Spenke. Multi-level hierarchical analogue fault simulation. *Microelectronics journal*, 33(10):815–821, 2002.

[132] S. Mirkhani, M. Lavasani, and Z. Navabi. Hierarchical fault simulation using behavioral and gate level hardware models. In *Test Symposium, 2002.(ATS'02). Proceedings of the 11th Asian*, pages 374–379. IEEE, 2002.

[133] Y. Kiliç and M. Zwoliński. Behavioral fault modeling and simulation using vhdl-ams to speed-up analog fault simulation. *Analog Integrated circuits and signal processing*, 39(2):177–190, 2004.

[134] L. Fang, G. Gronthoud, and H. G. Kerkhoff. Reducing analogue fault-simulation time by using ifigh-level modelling in dotss for an industrial design. In *Test Workshop, 2001. IEEE European*, pages 61–67. IEEE, 2001.

[135] L. Xia, I. M. Bell, and A. J. Wilkinson. A novel approach for automated model generation. In *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pages 504–507. IEEE, 2008.

[136] L. Xia, I. M. Bell, and A. J. Wilkinson. An automated model generation approach for high level modelling. In *Proceedings of the World Congress on Engineering*, volume 1. Citeseer, 2008.

[137] L. Xia, I. M. Bell, and A. J. Wilkinson. Automated model generation algorithm for high-level fault modeling. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 29(7):1140–1145, 2010.

[138] M. U. Farooq, L. Xia, F. A. Hussin, and A. S. Malik. High level fault modeling and fault propagation in analog circuits using nlarx automated model generation technique. In *Intelligent and Advanced Systems (ICIAS), 2012 4th International Conference on*, volume 2, pages 846–850. IEEE, 2012.

[139] R. Voorakaranam, S. Chakrabarti, J. Hou, A. Gomes, S. Cherubal, A. Chatterjee, and W. Kao. Hierarchical specification-driven analog fault modeling for efficient fault simulation and diagnosis. In *Test Conference, 1997. Proceedings., International*, pages 903–912. IEEE, 1997.

[140] S. Chakrabarti and A. Chatterjee. Compact fault dictionary construction for efficient isolation of faults in analog and mixed-signal circuits. In *Advanced Research in VLSI, 1999. Proceedings. 20th Anniversary Conference on*, pages 327–341. IEEE, 1999.

[141] N. Guerreiro and M. Santos. Mixed-signal fault equivalence: Search and evaluation. In *Test Symposium (ATS), 2011 20th Asian*, pages 377–382. IEEE, 2011.

[142] N. Guerreiro, M. Santos, and P. Teixeira. Fault list compression for cost-effective analogue and mixed-signal fault simulation. *27thConference on Design of Circuits and Integrated Systems, Avignon*, pages 261–266, 2012.

[143] N. Guerreiro, M. Santos, and P. Teixeira. Fault list compression for efficient analogue and mixed-signal production test preparation. In *Design of Circuits and Integrated Systems*, pages 1–6, Nov 2014.

[144] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[145] E. Rendón, I. Abundez, A. Arizmendi, and E. Quiroz. Internal versus external cluster validation indexes. *International Journal of computers and communications*, 5(1):27–34, 2011.

[146] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *Neural Networks, IEEE Transactions on*, 5(4):537–550, 1994.

[147] Z. Yang, M. Zwolinski, and C. Chalk. Bootstrap, an alternative to monte carlo simulation. *Electronics Letters*, 34(12), 1998.

[148] J. Hou and A. Chatterjee. Concurrent transient fault simulation for analog circuits. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 22(10):1385–1398, 2003.

[149] A. Newton. Techniques for the simulation of large-scale integrated circuits. *IEEE Transactions on Circuits and Systems*, 26(9):741–749, 1979.

[150] Z. R. Yang and M. Zwolinski. Fast, robust dc and transient fault simulation for nonlinear analogue circuits. In *Design, Automation and Test in Europe Conference and Exhibition 1999. Proceedings*, pages 244–248. IEEE, 1999.

[151] Y. Kilic and M. Zwolinski. Concurrent transient fault simulation of nonlinear analogue circuits. 2000.

[152] H. Hashempour, J. Dohmen, B. Tasić, B. Kruseman, C. Hora, M. Van Beurden, and Y. Xing. Test time reduction in analogue/mixed-signal devices by defect oriented testing: An industrial example. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*, pages 1–6. IEEE, 2011.

[153] B. Kruseman, B. Tasić, C. Hora, J. Dohmen, H. Hashempour, M. Van Beurden, and Y. Xing. Defect oriented testing for analog/mixed-signal devices. In *Test Conference (ITC), 2011 IEEE International*, pages 1–10. IEEE, 2011.

[154] M. Santos, F. Goncalves, M. Ohletz, and J. Teixeira. Defect-oriented testing of analogue and mixed signal ics. In *Electronics, Circuits and Systems, 1998 IEEE International Conference on*, volume 2, pages 419–424 vol.2, 1998.

[155] A. Designer. Cadence design systems, 2016.

[156] M. Graphics. Eldo, 2004.

[157] M. Vlach. Modeling and simulation with saber. In *ASIC Seminar and Exhibit, 1990. Proceedings., Third Annual IEEE*, pages T–11. IEEE, 1990.

[158] H. A. Mantooth and M. Vlach. Beyond spice with saber and mast. In *Circuits and Systems, 1992. ISCAS'92. Proceedings., 1992 IEEE International Symposium on*, volume 1, pages 77–80. IEEE, 1992.

[159] A. Ammous, S. Ghedira, B. Allard, H. Morel, and D. Renault. Choosing a thermal model for electrothermal simulation of power semiconductor devices. *Power Electronics, IEEE Transactions on*, 14(2):300–307, 1999.

[160] R. Rosing, A. Lechner, A. Richardson, and A. Dorey. Fault simulation and modelling of microelectromechanical systems. *Computing Control Engineering Journal*, 11(5):242–250, Oct 2000.

[161] V. Litovski, M. Andrejević, and M. Zwolinski. Acceleration of mems fault simulation using anns. *ELEKTRONIKA ELECTRONICS*, page 49, 2004.

[162] K. C. Lee, W. S. H. Wong, and H. T. Su. Fault analysis of a mems tuneable bandpass filter. In *Research and Development, 2007. SCOReD 2007. 5th Student Conference on*, pages 1–5, Dec 2007.

[163] K. Lilja, M. Bounasser, S.-J. Wen, R. Wong, J. Holst, N. Gaspard, S. Jagannathan, D. Loveless, and B. Bhuva. Single-event performance and layout optimization of flip-flops in a 28-nm bulk technology. *IEEE Transactions on Nuclear Science*, 60(4):2782–2788, 2013.

[164] T. J. Barnes. Skill: A cad system extension language. In *Proceedings of the 27th ACM/IEEE Design Automation Conference*, pages 266–271. ACM, 1991.

[165] M. Guiney and E. Leavitt. An introduction to openaccess: an open source data model and api for ic design. In *Proceedings of the 2006 Asia and South Pacific Design Automation Conference*, pages 434–436. IEEE Press, 2006.

[166] A. Pirker-Fruhauf and M. Kunze. A novel methodology to combine and speed-up the verification process of simulation and measurement of integrated circuits. In *2008 IEEE AUTOTESTCON*. 2008.

[167] R. Handbook. exida. com llc, safety equipment reliability handbook, 2003. Technical report, ISBN 0-9727234-0-4, 2003.

[168] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[169] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Journal of intelligent information systems*, 17(2-3):107–145, 2001.

[170] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[171] G. H. Dunteman. *Principal components analysis*. Number 69. Sage, 1989.

[172] T. Velmurugan and T. Santhanam. Computational complexity between k-means and k-medoids clustering algorithms for normal and uniform distributions of data points. *Journal of computer science*, 6(3):363, 2010.

[173] H. G. Stratigopoulos and S. Sunter. Efficient monte carlo-based analog parametric fault modelling. In *VLSI Test Symposium (VTS), 2014 IEEE 32nd*, pages 1–6, April 2014.

[174] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola. *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[175] F. Pianosi and T. Wagener. A simple and efficient method for global sensitivity analysis based on cumulative distribution functions. *Environmental Modelling & Software*, 67:1–11, 2015.

[176] D. L. Allaire and K. E. Willcox. A variance-based sensitivity index function for factor prioritization. *Reliability Engineering & System Safety*, 107:107–114, 2012.

[177] F. Campolongo, J. Cariboni, A. Saltelli, and W. Schoutens. Enhancing the morris method. In *Sensitivity Analysis of Model Output. Proceedings of the 4th International Conference on Sensitivity Analysis of Model Output (SAMO 2004)*, pages 369–379, 2005.

[178] G. Sin and K. V. Gernaey. Improving the morris method for sensitivity analysis by scaling the elementary effects. *Computer Aided Chemical Engineering*, 26:925–930, 2009.

[179] M. D. Morris. Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2):161–174, 1991.

[180] I. M. Sobol'. On sensitivity estimation for nonlinear mathematical models. *Matematicheskoe Modelirovanie*, 2(1):112–118, 1990.

[181] K. Chan, A. Saltelli, and S. Tarantola. Winding stairs: a sampling tool to compute sensitivity indices. *Statistics and Computing*, 10(3):187–196, 2000.

[182] L. L. Nathans, F. L. Oswald, and K. Nimon. Interpreting multiple linear regression: A guidebook of variable importance. *Practical Assessment, Research & Evaluation*, 17(9), 2012.

[183] A. Saltelli, S. Tarantola, and K.-S. Chan. A quantitative model-independent method for global sensitivity analysis of model output. *Technometrics*, 41(1):39–56, 1999.

[184] J. C. Helton. Uncertainty and sensitivity analysis techniques for use in performance assessment for radioactive waste disposal. *Reliability Engineering & System Safety*, 42(2-3):327–367, 1993.

[185] F. Campolongo, J. Cariboni, and A. Saltelli. An effective screening design for sensitivity analysis of large models. *Environmental modelling & software*, 22(10):1509–1518, 2007.

[186] K. Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[187] J. W. Johnson. Determining the relative importance of predictors in multiple regression: Practical applications of relative weights. *Child Development*, 59:969–992, 2001.

[188] B. Efron and B. Efron. *The jackknife, the bootstrap and other resampling plans*, volume 38. SIAM, 1982.

[189] B. Efron and R. Tibshirani. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical science*, pages 54–75, 1986.

[190] S. Tonidandel, J. M. LeBreton, and J. W. Johnson. Determining the statistical significance of relative weights. *Psychological methods*, 14(4):387, 2009.

Declaration of Authorship

I hereby confirm that I have authored this thesis independently and without use of others than the indicated sources. All passages which are literally or in general matter taken out of publications or other sources are marked as such.

Munich, March 31st, 2017

Özlem Karaca