

Multiple Faults Simulation of Analogue Circuits

Eduard Weber, Klaus Echtle
 University of Duisburg-Essen
 Dependability of Computing Systems
 Essen, Germany
 e-mail: (echtle, weber)@dc.uni-due.de

Abstract—Software-based fault simulation can support all abstraction levels, is flexible and allows reliability assessment at different stages in the design process. Fault diagnosis and reliability analysis are increasingly important in circuit design and determine the product's time-to-market. In this paper, we provide a new efficient method and systematic scheme for reducing the time for simulation of multiple simultaneous faults and/or multiple failure modes per element in an analogue circuit. By arranging similar multiple faults in groups, some so-called failure classes can be interpolated with an adequate precision rather than being evaluated by time-consuming simulation. The technique can be used to perform efficient multiple fault diagnosis based on multiple fault injection. Finally, the implemented procedure is validated with some simulation results.

Keywords—Fault simulation; fault modeling; multiple fault injection; fault diagnosis; reliability prediction

I. INTRODUCTION

Fault diagnosis of circuits is a well-developed research field with a long tradition. The first scientific publications are from early 1960s. Circuit simulation is nowadays an accepted standard in the development of electronic circuits. Small to complex analogue, digital and mixed-signal circuits can be tested and verified with appropriate simulation software. A lot of progress has been made in the development of software tools for the design and verification of analogue and/or mixed-signal circuits, both in the open-source and in the commercial sector. Already two decades ago the method of analogue fault modelling has been suggested to enable both fault diagnosis and reliability evaluation. Different approaches have been developed for fault simulation of analogue and mixed-signal circuits. Previous work on analogue fault modelling focuses on parametric defects (soft faults) and catastrophic defects (hard faults). Parametric faults are typically simulated with parameter modifications, while open and short defects are dealt with via injecting a high or low resistance on transistor level, respectively.

Fault simulation is generally done by injecting a fault on transistor level and analysing the circuit's behaviour by applying single DC, transient or AC simulation for linear or nonlinear circuit models. Also, software tools for automatic fault injection and efficient test generation have been developed. However, mostly single faults have been considered in the past.

Test cases for fault injection have been generated often by hand from an understanding of the design and fault expectations of major circuit elements (components). Most of the fault simulators for analogue circuits presented in the literature cover only parameter or catastrophic faults. Some tools have attempted to automate test generation and the fault simulation process for analogue circuits. The runtime problem of analogue circuit simulation also needs to be addressed, and advanced simulation techniques are required to accelerate the simulation to an acceptable proportion [1].

Most existing fault simulators use the Simulation Program with Integrated Circuits Emphasis (SPICE), and modify SPICE net lists to represent faults [2] - [3]. The fault simulation software [4] used for the work presented in this paper defines circuit faults in Visual Basic (VB-Script) language and allows flexible and very accurate fault modelling. The main goal of this paper is to speed up the simulation for multiple faults.

II. DIAGNOSIS OF ANALOGUE CIRCUITS

Test and fault diagnosis of analogue circuits are necessary despite the ongoing digitalization. Analogue circuits are always required to form the interface to the physical environment. Analogue signals do not consist of just "low" or "high" values like in the digital field. In principle, infinite numbers of signal values are conceivable. The time and frequency characteristics of analogue signals bring another dimension, and are an additional issue within circuit assessment. The propagation of faults is more difficult than in the digital field. Typically, it does not occur in just one direction, but could be from any element in all directions towards neighbour elements within the circuit. A particular fault in an element (like resistor, capacitor, transistor, etc.) does not provide explicit information about the resulting signal values. Therefore, a calculation of signal values (done by circuit simulation) is always necessary.

Nonlinear models, parasitic elements, charges between elements or energy-storing elements make diagnosis and reliability analysis more complex [5]. Because of these reasons, the automation level of fault diagnosis procedures for analogue circuits has not yet achieved the development level realized in the digital field. The reason for the limited automation is simply due to the nature of analogue circuits. The predominant design methodology for analogue circuits is still the individual design based on the designer's experience.

The simulation of multiple simultaneous faults is even more complex. The consideration of multiple faults is important for the following reasons. Different fault modes can be present in the elements of complex circuits. Their occurrence increases even more in rough environments. Also, multiple parametric faults can be present in the field as a result of ageing, environmental stress and design errors. Moreover, multiple fault diagnosis is relevant when a new circuit design is introduced and a high failure density exists. The restriction to single fault simulation only can lead to incorrect evaluation results.

One of the main issues in software-based fault simulation is the relatively long runtime in case of complex analogue circuits. In general, the runtime increases rapidly with the simulated circuit size, the number of faulty elements (fault depth or multiple fault simulation) and the failure modes per element. When performing fault simulation, the runtime is mostly determined by the number of fault injections. Each injection of a multiple fault has to be simulated separately. Usually, the simulation time for single faults (at transistor level) is tractable because of available computer performance. Also, the performance of Electronic Design Automation (EDA) tools has been increased during the last decade. However, multiple fault injection is a challenge with respect to runtime.

The fault simulation framework [4] used for the work presented in this paper can deal with several fault modes injected simultaneously into elements of a circuit. We consider permanent hard (open and short circuit) and soft faults (parametric faults). Please note, that even shorts and opens are dealt with as analogue (not digital) faults, because the simulator generates the analogue signal throughout the complete circuit in the case of these faults.

Figure 1 shows how the total simulation time (here number of simulation runs) is influenced by the number of multiple faults and the failure modes per element. The diagram shows a medium-sized circuit example composed of 20 elements where faults are injected, each of which leads to two different failure modes. The solid line represents the number of simulation runs for all necessary test cases. This quantity increases rapidly with the number of multiple faults.

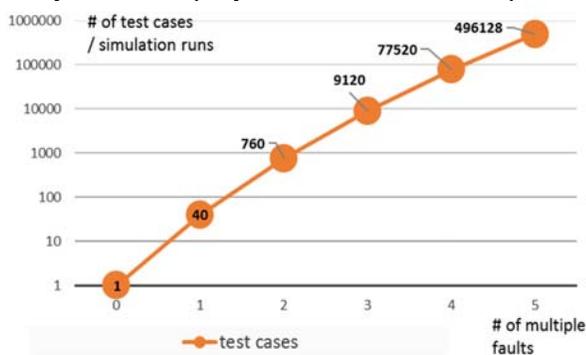


Figure 1. Complexity of fault simulation for an example medium-sized circuit (20 elements with two fault modes per element).

III. FAULT SIMULATION FRAMEWORK

The starting point of EDA-based fault simulation is the circuit's schematic model. The designer can construct a circuit with all available elements by using some circuit design tool. A wide variety of measurements and graphical data representation (also denoted graphs) can be utilized. NI AWR Microwave Office® (National Instruments) [6] features broad post-simulation capabilities, allowing displaying of computed data (measurements, such as gain, noise, power, or voltage) on rectangular graphs, polar grids, Smith Charts, histograms, tabular graphs, and 3D graphs, etc. Every defined measurement point is associated with a particular graph.

The fault simulator [4] uses the graphs to check the circuit's behaviour after fault injection by defining tolerance bands and success areas. The defined tolerance bands and success areas are stored as parts of so-called goals. The circuit under diagnosis (CUD) and its success can be measured in detail by inspection of multiple graphs. After each simulation run the deviation between the fault-free and faulty response is computed for preselected measurements. If the difference exceeds the tolerance band, the injected fault is declared as not being tolerated by the circuit.

The general process of fault simulation is depicted in Figure 2. In the first step the fault-free circuit is simulated. Fault modes for circuit elements are defined within the GUI of the implemented fault simulator (implemented on top of the NI AWR Microwave Office simulator [6]). Usually, several fault modes are possible for each element. Faults are injected into the user-defined circuit via predefined fault modes. The fault injection is done automatically and is undone after every fault simulation run. This means a direct fault modification inside the original circuit within the EDA environment. In addition to the hard faults (open or short circuit) also soft faults (mostly parameter faults that provide a flexible parameter variation of the models of circuit elements) are possible. Faults may change the electrical values (increase or decrease) permanently or for a short time (e.g., temperature), and modify the behaviour of the individual elements which can also lead to a global malfunction of the circuit. After each simulation, measurement data are compared with user-defined goals specified by the tolerance bands. Multiple faults are considered to increase testing quality and enable better reliability analysis. Obviously, the quality of fault simulation highly depends on a realistic set of faults. The fault simulator can automatically generate hard faults (open-circuit, short-circuit) depending on the elements utilized in the circuit. Additionally, parameter faults can be generated automatically or specified by the user.

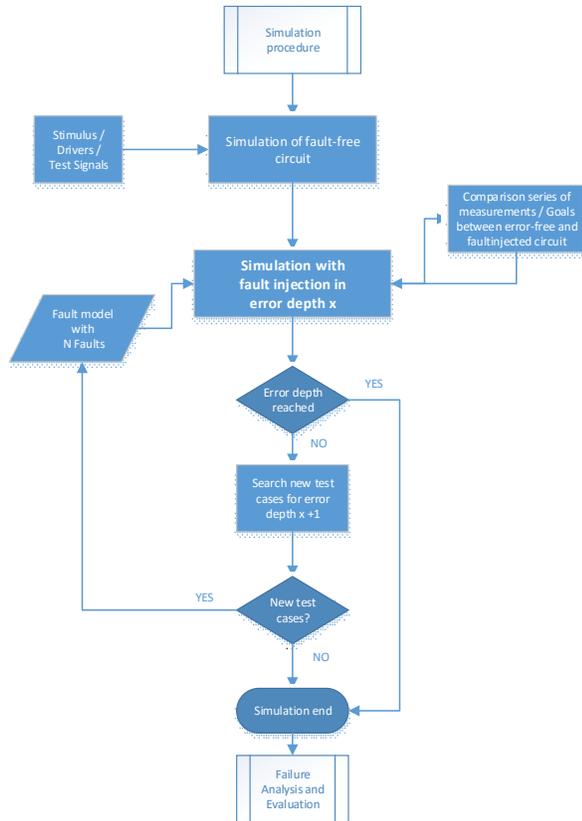


Figure 2. General process of fault simulation.

IV. DEFECTS AND ANALOGUE FAULT MODELING

Fault simulation can only work effectively when the fault model corresponds as closely as possible to the real physical defects. One common approach is the usage of inductive fault analysis (IFA) [7]. The circuit layout and statistical selection among production errors form the basis for IFA. Physical defects of the circuit can be modelled by fault models at the transistor level or at higher abstraction levels. Finally, the fault list will be developed and adapted by the probability of occurrence. Due to the characteristics of analogue circuits, infinitely interim parameter values are possible, so there are an infinite number of analogue errors and, therefore, indeterminate fault models. Therefore, an optimal subset of faults must be selected to do fault simulation with realistic simulation duration and sufficient accuracy. The defined fault list at the transistor level serves as input for the sequential or concurrent fault simulation. The generation of the fault list is therefore a very important step of fault simulation, since it directly determines the quality of the simulation results and time of the analysis. The modelled defects or error types are simulated with test stimuli according to the profile of circuit’s application. The results of fault simulation are used to validate its fault tolerance, fault detection and the circuit’s quality in general. In other words, Fault simulation reveals the redundancy of a circuit (whether implemented in the circuit intentionally or just by chance).

The physical types of defects can be distinguished into two basic classes. Either a fault has arisen from neighbourhood according to the layout, or from an element itself.

Layout-based faults include defects that are only possible in special layout configurations (e.g., placement of elements). In particular, shorts are only possible between neighbouring elements (with very few exceptions). Shorts between distant elements need not be considered in the fault model. Accordingly, parasitic capacitances arise when two electrical connection lines are close enough to each other and the frequency is in the respective range. Trivially, open faults can only occur where a connecting line exists in the layout.

Element-based fault types are defects, which arise within an element. Typically, the model of these faults expresses the element’s behaviour between its terminals (e.g., emitter, base and collector of a bipolar transistor, connection pins of resistors, capacitors, diodes, etc.). Six primary error types (three shorts and three opens) can be defined for elements having three terminals and at least two types for passive elements with two pins. Traditionally, opens and shorts are modelled in form of resistors with high or low resistance, respectively. Open defects have a value greater than 1 GΩ and shorts between 0 and a few ohms, within chips up to 500 ohms [8]. If the resistance of a short is relative small, we speak of a strong short, otherwise of a weak short. Analogously, we define a strong open by an almost infinite resistance, and a weak open by a resistance of some 100 MΩ or GΩ. Real analogue circuit’s faults have ideal shorts and opens only in rare cases. Therefore, a set of appropriate parameter values must be chosen.

TABLE I. PHYSICAL DEFECTS AND ELECTRONIC FAULT MODELS FOR OPEN ERRORS.

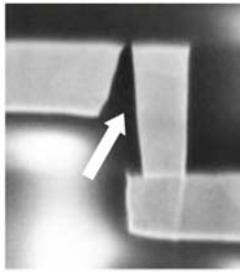
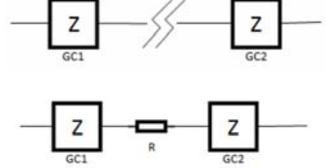
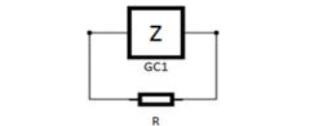
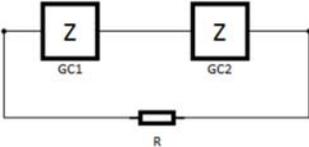
Physical defect	Electronic fault model
 <p>Open circuit [9]</p>	 <p>The resistance value (R) has a high value (> 100MΩ).</p> <p>An open defect can have multiple subtypes:</p> <ul style="list-style-type: none"> • Complete (strong) open, without any electrical connection • Partial (weak) open, variates the line resistance

TABLE II. PHYSICAL DEFECTS AND ELECTRONIC FAULT MODELS FOR SHORT ERRORS.

Physical defect	Electronic fault model
bridging defect / short	
	
Short circuit with low resistance [9]	The resistance value (R) has a low value ($< 500\Omega$).
	
Short circuit with high resistance [9]	A short circuit can at least be assumed between the pins of elements or generally between each connection line. Real shorts have rarely a resistance value of exactly 0 ohms, most shorts are in the range between 0 and 500 ohms.

To describe fault models precisely, the cause and the appearance of errors must be understood. Especially for analogue circuits, the circuit layout and the element's parameters influence the possible errors and the effect on the circuit's performance. Defects or undesirable characteristics can sneak up not only during manufacturing, but can also arise at the utilization phase. In Table I and Table II the reader will find general physical defects and the equivalent electronic fault models.

The following Tables III, IV, and V provide an overview of the basic types of analogue elements and error models. We show the simplest errors at first, i.e., shorts by lines and, therefore, without a resistance value or a capacitance. Complete interruptions (i.e., opens) of elements are modelled by disabling the elements (i.e., switching the respective element OFF). Parameter errors, interruptions and weak shorts can be expressed by variation of the respective parameter values. Basically, the implemented fault simulator can be applied to all assigned elements and circuit models in the EDA environment. All types of faults are described by a script language (VBScript) and applied directly to the circuit schematics before each fault simulation.

TABLE III. SELECTION OF FAULT MODELS FOR A RESISTOR

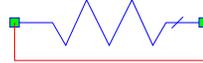
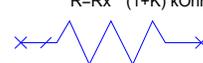
Element with fault injection	
Resistor	
	
Strong short	Strong short
	
Parameter fault (Depending on model)	Strong open

TABLE IV. SELECTION OF FAULT MODELS FOR A TRANSISTOR.

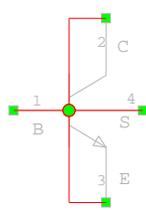
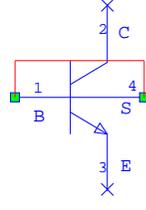
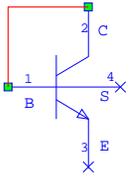
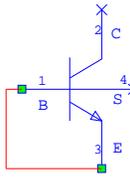
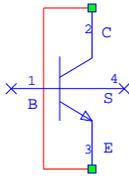
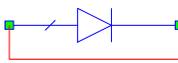
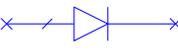
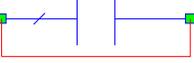
Element with fault injection	
Transistor	
	
Strong short	BS short (S = substrate)
	
BC short	BE short
	
CE short	
Faults of any combination of the ports are possible.	

TABLE V. SELECTION OF FAULT MODELS FOR A DIODE AND CAPACITOR.

Element with fault injection	
Diode	
 <p>Strong short</p>	 <p>Strong short</p> <p>DIODE1 ID=D2 Nu=1.2 $T=21.85 * (1 + K) \text{ DegC}$ $I_o=1e-6 * (1 + K) \text{ mA}$</p>
 <p>Strong open</p>	 <p>Parameter fault (Depending on the model)</p>
Capacitor	
 <p>Strong short</p> <p>CAP ID=CAP1 C=Cx pF</p>	 <p>Strong short</p> <p>CAP ID=CAP1 C=Cx pF</p>
 <p>Parameter fault (Depending on model)</p> <p>CAP ID=CAP1 C=Cx * (1 + K) pF</p>	

V. STRATEGIES FOR REDUCING SIMULATION TIME

One of the main problems in analogue fault simulation is the relatively long simulation time. Acceleration of fault simulation / reducing the simulation duration is an important goal. To reduce the runtime of simulation with fault injection the following two general approaches are feasible: reduce the amount of fault injections (simulation runs) or speed up the simulation procedure for every fault injection. Several approaches are described in the literature to speed-up the simulation process, including fault or test case ordering [10–13] and distributed fault simulation [14][15]. Several approaches for multiple fault generation [16][17] and simulation [18][19] for reliability analysis are described in the literature as well. In the following, the reader can find an overview of several techniques on how the simulation duration can be reduced.

A. Concurrent or parallel fault simulation

Simulation by using multiple physical or logical CPU cores can accelerate the circuit or fault simulation, respectively. Multiple CPU cores can simulate parts of a single circuit, and collect the results at the end. Then the faster simulating cores wait for the final part of the simulation and can then move on to the next fault simulation. In the general case of fault simulation, the different injected faults can be

simulated independently. For this reason, the fault simulation can be performed very efficiently because each CPU core simulates a different fault. If one CPU core finishes earlier, it starts immediately with the next fault simulation. Parallel fault simulation can also be processed on independent workstations that are connected over a network, see [20].

B. Random fault simulation

Simple random sampling is a widely known test method for selecting errors in digital systems. It is a classic approach, where a small subset of errors out of the large set of errors is selected to determine certain characteristics of the system. The achieved accuracy depends on the size of the selected subset. For example, if the number of potential or possible errors is 5,000 and 500 errors have been selected and simulated, then the reduction in simulation time is a factor of 10.

C. Fault models on higher level of abstraction (e.g., for elements not being subject to fault injection)

A widely known method to speed up the fault simulation is partial simulation on a higher level of abstraction. Parts of the circuit can be simulated, for example, in the form of an analogue description language (Verilog-AMS or VHDL-AMS) or by simplified SPICE models. The fault simulator can thereby accelerate simulation runs by replacing all elements without fault injection through behavioural models which are on a higher level and more efficient to simulate. In most cases, this reduces the simulation time significantly.

TABLE VI. IMPORTANT PARAMETERS THAT INFLUENCING ELEMENT FAILURE RATE. BASED ON [21].

D is a symbol for dominant parameter and x for important parameter.

Stress parameters	Hybrid circuits	Bipolar transistors	FETs	Diodes	Resistors	Capacitors	Coils, transform.	Relays, switches	Connectors
Ambient temp.	D				D	D	D	D	D
Junction temp.	D	D	D	D					
Power stress	D	D	D	x	D		x		
Voltage stress	D	x	x	x		D	x	x	
Current stress	D			x				x	x
Breakdown voltage	x	x	x	x					
Technology	x	x	x	x	x	x	x	x	x
Complexity	x							x	
Package	x	x	x	x		x		x	x
Application	x	x	x	x		D		x	x
Contact construction	x	x	x	x				D	D
Range	x	x		x	x	x			x
Production maturity	x	x	x	x	x	x	x	x	x
Environment	x	x	x	x	x	x	x	x	x
Quality	x	x	x	x	x	x	x	x	x

D. Probability of occurrence of element defects

Some studies have examined the probability of occurrence of individual element fault types. In Table VI, the reader will find the most relevant stress parameter types that influence the failure rate of analogue electronic elements. Table VII provides an overview of statistical distributions of faults for different electronic elements. It can be noted that 60% to 100% of analogue element's defects are some kind of shorts and opens. The remaining faults are mainly drifts of parameter values. The fault simulation can be performed only on the most likely types of element faults (e.g., 70%). The less likely faults can be omitted.

TABLE VII. STATISTICAL INDICATOR VALUE FOR FAILURE MODES BASED ON [21].

Element	Shorts	Opens	Drift	
Bipolar transistors	80	20		
Field effect transistors (FET)	80	10	10	
Diodes (Si)	general purpose	80	20	
	Zener	70	20	10
Resistor, fixed (film)		60	40	
Capacitors	foil	15	80	5
	ceramic	70	10	20
	Ta (solid)	80	15	5
	Al (wet)	30	30	40
Coils	20	80		
Quartz crystals		80	20	

E. Early abortion of a simulation run

The values of element parameters and the tolerance bands are specified before fault simulation starts. This can be done by simulating the fault-free circuit and determining the allowed deviations of the measures values. During the fault simulation procedure, the measured values can be compared with the predetermined tolerance range. If some value is above or below the specified limits of fault-free circuit, then it is immediately clear that the injected fault has not been tolerated. Consequently, further simulation of this injected fault can be skipped. The reduction in simulation time depends on the amount of faults that allow early abortion of simulation and, moreover, the point in time when the tolerance band is violated. This method has already been used in transient analogue fault simulation, see [10, 22] for example.

F. Leave out elements outside the test object

Circuit may include some "additional elements" that are not of interest because they are outside the test object. They should be excluded from simulation.

G. Leave out unrecognizable defects

In special cases, it may be clear that some faults of an element do not cause an effect that can be recognized. Examples are unconnected elements, special types of elements, or elements that fulfil some protection functionality. Since the simulator would not notice any effect of an injected fault, the respective fault case can be omitted.

H. Monotonicity assumption

A basic rule (if applicable) is the assumption of monotonic behaviour. Two joint faults will not be tolerated, if at least one of them is not tolerated when injected as single fault. By "tolerated" we mean that the circuit under diagnosis (CUD) is still providing its function according to a given maximum deviation from the expected behaviour. The monotonicity assumption has the advantage that many irrelevant multiple fault combinations can be discarded before being simulated. The effect to the number of test cases (= simulation runs) is quite substantial. Discarding dual faults will also result in a smaller number of considered triple faults, and so on. The simulation time is reduced for all multiple fault combinations (see Figure 3). The dashed line shows that the quantity of simulation runs can be reduced significantly by assuming monotonic behaviour as follows: When a set F of multiple simultaneous faults is not tolerated, then also a superset of F will not be tolerated. Consequently, the superset needs not be simulated. The assumption of monotonic behaviour is slightly pessimistic. In practice there are rare exceptions. Think of two resistors in series, each of 1 k Ω . If both of them are parametrically faulty and half their resistance down to 500 ohms, then the voltage at the point between them does not necessarily change. It may still be correct. Monotonicity does not always exist. However, we have observed that it exists in an overwhelming majority of cases with only very few exceptions. In general, the monotonicity assumption reduces the number of both considered circuit elements and failure modes per element.

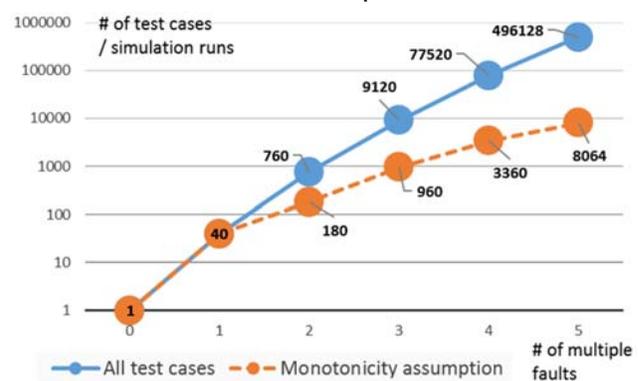


Figure 3. Complexity of fault simulation for an example medium sized circuit (20 elements with two fault modes per element). Only ten elements are considered at monotonicity assumption for multiple faults (≥ 2).

I. Standardized input parameters

During fault simulation, a circuit simulation is made for every fault injection. Repetitive parts of the circuit simulation, such as DC analysis (steady-state) can be conducted only once by simulation of the fault-free circuit. Thus, the fault simulations finish earlier and reduce the total simulation time.

J. Measurement-based simulation

Once the measurement point within the circuit have been defined, one may identify parts of the circuit that do not exercise an influence to these measurement points. Consequently, the non-relevant parts can be excluded from simulation to achieve an overall acceleration. The exclusion does not affect the checks whether or not the tolerance bands are violated. Note that the decision on an exclusion is more complicated in analogue circuits compared to digital circuits, because there may be no clear input and output pins of an element. In a bipolar transistor, for example, the base current determines the collector current. However, depending on the operation conditions, there may also be an influence from the collector to the base.

K. Faults classes (focus of this paper)

In the remainder of the paper, we present a further method how the number of simulation runs can be reduced, see Sections IV and V. Before we describe the method we will formalize the selection of test cases to achieve a better precision in the description of the fault classes (= sets of fault cases) the new method is making use of.

Formally, the relationship between faults, elements of the circuit, injections and simulation runs is defined by the following tuples and functions:

- 1) $C = \{c_0, \dots, c_m\}$ is the set of circuits to be evaluated, $c_0 \in C$ is the fault-free circuit.
- 2) $E = \{\text{transistor1, transistor2, ..., resistor1, ..., ...}\}$ is the set of elements of the circuit c_0 .
- 3) $F = \{\text{short_circuit, open_circuit, parameter_mod., ...}\}$ is the set of considered fault modes of the circuit c_0 .
- 4) $I = \{(f, e) \in F \times E : \text{probability of fault } f \text{ in element } e\}$ is the set of potential injections.
- 5) $I^* = \{i^* \subset I : (x \in i^*, y \in i^*, x \neq y) \Rightarrow x|E \neq y|E\}$ is the set of potential multiple injections. I^* is a subset of the power set of I . By $x|E$ and $y|E$ we denote the element of injection x or injection y , respectively. The inequality $x|E \neq y|E$ excludes joint injection of different faults to the same element of the circuit.
- 6) $Q : F \times E \rightarrow [0, 1]$ is the probability of fault $f \in F$ in a faulty element $e \in E$. If a fault $f \in F$ is not applicable to an element $e \in E$ then $Q(f, e) = 0$. For a given faulty element $e \in E$ the sum of fault probabilities is always 1:
 $\sum_{f \in F} Q(f, e) = 1$.

Example: If we assume only two fault modes $F = \{\text{open, short}\}$ and only two elements $E = \{R_1, R_2\}$, there may be four

injections $I = \{(\text{open}, R_1), (\text{open}, R_2), (\text{short}, R_1), (\text{short}, R_2)\}$ and four double injections. In all we obtain:

$$I^* = \{ \{(\text{open}, R_1)\}, \{(\text{open}, R_2)\}, \{(\text{short}, R_1)\}, \{(\text{short}, R_2)\}, \{(\text{open}, R_1), (\text{open}, R_2)\}, \{(\text{short}, R_1), (\text{short}, R_2)\}, \{(\text{open}, R_1), (\text{short}, R_2)\}, \{(\text{short}, R_1), (\text{open}, R_2)\} \}$$

If shorts are more likely for R_1 and opens are more likely for R_2 we may get, say,

$$Q(\text{open}, R_1) = 0.2, \quad Q(\text{short}, R_1) = 0.8 \quad (0.2 + 0.8 = 1) \\ Q(\text{open}, R_2) = 0.4, \quad Q(\text{short}, R_2) = 0.6 \quad (0.4 + 0.6 = 1)$$

$P : E \rightarrow [0, 1]$ is the function indicating the probability that element $e \in E$ is fault-free.

Function $R : I^* \rightarrow \{0, 1\}$ is a simulation run with joint injection of all faults from $i \in I^*$. The method returns 1 if the injected faults are tolerated according to the tolerance criterion, otherwise 0. In the following, the fault simulation procedure is described for single, double, triple fault injection.

Single faults:

$I_1 = I$ is the set of single fault injections to be evaluated by simulation.

$T_1 = \{i \in I_1 : R(i) = 1\}$ is the set of single injections that have been tolerated. The function

$$P_1 = \sum_{i \in I_1} R(i) \cdot (1 - P(i|E)) \cdot Q(i|F) \cdot \prod_{y \in (I_1 \setminus i)} P(y|E)$$

expresses the probability of tolerated single injections.

Double faults:

$I_2 = \{(f, e), (f', e') : (f, e) \in T_1, (f', e') \in T_1, e \neq e'\}$ is the set of double injections to be evaluated by simulation.

I_2 has been defined on the basis of T_1 , not I_1 , because the non-tolerated injections from the complement $I_1 \setminus T_1$ are excluded due to the assumption of monotonicity.

$T_2 = \{i^* \in I_2 : R(i^*) = 1\}$ is the set of double injections that have been tolerated.

$$P_2 = \sum_{i^* \in I_2} R(i^*) \cdot \prod_{x \in i^*} (1 - P(x|E)) \cdot Q(x|F) \cdot \prod_{y \in (I_2 \setminus i^*)} P(y|E)$$

expresses the probability of tolerated double injections.

Triple faults:

$I_3 = \{(f, e), (f', e'), (f'', e'') : (f, e), (f', e') \in T_2, (f'', e'') \in T_1, e \neq e', e \neq e'', e' \neq e''\}$

is the set of triple injections to be evaluated by fault simulation. Again, the non-tolerated previous injections have been excluded due to the assumption of monotonicity.

$T_3 = \{i^* \in I_3 : R(i^*) = 1\}$ is the set of triple injections that have been tolerated.

$$P_3 = \sum_{i^* \in I_3} R(i^*) \cdot \prod_{x \in i^*} (1 - P(x|E)) \cdot Q(x|F) \cdot \prod_{y \in (I_3 \setminus i^*)} P(y|E)$$

expresses the probability of tolerated triple injections.

The injections of higher numbers of joint faults are defined accordingly.

VI. FAULT CLASS ALGORITHM

The algorithm is an heuristic approach that is based on an observation of simulation results [4] of so-called fault classes. A fault class is a set of test cases (series of fault injections)

all of which have the same number of faults and the same types of fault modes, independent of the element where the faults are injected.

Experimental results show that three fault classes FC1, FC2 and FC3 for multiple faults mostly exhibit a monotonically increasing degree of tolerance, when the fault distance between FC1 and FC2 is 1, and also the fault distance between FC2 and FC3 is 1. By a fault distance $d(FC, FC')$ (similar to the Hamming distance), we understand the number of fault modes that differ between FC and FC'. The degree t of tolerance is defined by the number of tolerated test cases divided by the number of all test cases of a fault class.

The case $d(FC1, FC2) = d(FC2, FC3) = 1$ means that each pair of fault classes differs by just one fault mode. For example, consider the following fault classes:

FC1 (open, open, open),
FC2 (open, open, short),
FC3 (open, short, short).

The fault distances are $d(FC1, FC2) = d(FC2, FC3) = 1$ and $d(FC1, FC3) = 2$. Typically this leads to either $t(FC1) \leq t(FC2) \leq t(FC3)$ or $t(FC1) \geq t(FC2) \geq t(FC3)$.

From this observation we developed an algorithm that can be characterized as follows:

- Search for fault classes FC1, FC2, FC3 satisfying the condition above – or search for even longer chains (>3 multiple faults) of fault classes with this property.
- Determine which of the chains will typically lead to an ascending or descending degree of tolerance. To decide that, analysing the fault classes of the previous fault depth is necessary, see Step 2 of this section below.
- Quantify the tolerance of the first and the last fault class of a chain by simulation.
- Quantify the tolerance of the remaining fault classes of a chain by interpolation.

Fault classes are defined by the modes of the injected faults and their number of simultaneously injected faults. $FC_2(x, y)$ denotes a fault class for two joint injections, namely fault modes x and y . Since the fault classes $FC_2(x, y)$ and $FC_2(y, x)$ are identical, we enforce a unique notion by assuming an order among the fault modes. Since fault modes x and y may be identical (injection of two faults of identical mode into different elements), we require $x \leq y$ for $FC_2(x, y)$. For an arbitrary fault class $FC_n(x_1, x_2, \dots, x_n)$ we require $x_1 \leq x_2 \leq \dots \leq x_n$. Then, a fault class for double fault injection is defined as follows:

$$FC_2(x, y) = \{ \{(f, e), (f', e')\} \in I_2 : f = x, f' = y \}$$

A fault class for the injection of n faults is defined accordingly: $FC_n(x_1, \dots, x_n) = \{ \{(f_1, e_1), \dots, (f_n, e_n)\} \in I_n : f_i = x_i \}$.

The subset of test cases in a fault class $FC_n(x_1, \dots, x_n)$ that has been tolerated is called tolerance class $TC_n(x_1, \dots, x_n)$. The following holds: $TC_n(x_1, \dots, x_n) \subset FC_n(x_1, \dots, x_n)$. Moreover, $TC_n(x_1, \dots, x_n) = FC_n(x_1, \dots, x_n) \cap TC_n$. The quotient of the

cardinality of $TC_n(x_1, \dots, x_n)$ and the cardinality of $FC_n(x_1, \dots, x_n)$ is called tolerance degree $t_n(x_1, \dots, x_n)$. Thus

$$t_n(x_1, \dots, x_n) = \frac{|TC_n(x_1, \dots, x_n)|}{|FC_n(x_1, \dots, x_n)|}$$

The heuristic approach is defined in the following steps and the algorithm is shown in Figures 4 and 5. We assume that the tolerance classes $TC_1(\dots)$ and $TC_2(\dots)$ have already been generated by the respective fault simulations. Consequently, the tolerance degrees $t_1(\dots)$ and $t_2(\dots)$ are known. Then the following steps describe how the fault classes $FC_3(\dots)$ for triple fault simulation – or interpolation! – are formed.

A. Step 1 – Generation Of Fault Classes

A fault class $FC_3(x, y, z)$ with 3 faults is generated by combining all test cases of TC_2 with all test cases of TC_1 in the following way: Each union of a test case $tc_2 \in TC_2(x, y)$ and a test case $tc_1 \in TC_1(z)$ form a test case $tc_3 \in FC_3(x, y, z)$ provided x, y and z inject faults into different elements. Since we avoid double injections into a single element, the respective combined injections $\{x, y, z\}$ are filtered out. The corresponding algorithm is shown in Figure 4. In the algorithm we denote the fault mode of injection x by $x|F$.

Procedure 1 Generate Fault Classes

```

for all test cases  $tc_2 \in TC_2$  do
  for all test cases  $tc_1 \in TC_1$  do
    { test case  $\{x, y, z\} = i \cup j$ ;
      if  $x|E \neq y|E$  and  $x|E \neq z|E$  and  $y|E \neq z|E$  then
         $FC_3(x|F, y|F, z|F) = FC_3(x|F, y|F, z|F) \cup \{x, y, z\}$ 
    }

```

Figure 4. Generate Fault Classes.

B. Step 2 – Search Fault Class Chains

The search of fault class chains starts with a search in TC_2 . We inspect all pairs of tolerance classes $TC_2(x, y)$ and $TC_2(x', y')$ and filter out those with a fault distance of 1 and, moreover, with “significantly unequal” tolerance degrees (the difference should be at least Δ). Formally: $d(TC_2(x, y), TC_2(x', y')) = 1$ and $|t_2(x, y) - t_2(x', y')| \geq \Delta$ where Δ may be in the range of 5% of the absolute values. From the fault distance 1 we can conclude that either $x = x'$ or $y = y'$. In the following, we assume $x = x'$ and $y \neq y'$ without loss of generality.

From the two tolerance classes $TC_2(x, y)$ and $TC_2(x, y')$ we derive the following chain of three fault classes:

$$\langle FC_3(x, y, y'), FC_3(x, y, y'), FC_3(x, y', y') \rangle$$

According to the observation of likely monotonicity (see Section II and Section IV) we only simulate the test cases of the first and the last fault class in the chain to obtain the tolerance degrees $t_3(x, y, y')$ and $t_3(x, y', y')$, respectively. The tolerance degree $t_3(x, y, y')$ of the inner fault class in the chain is obtained by interpolation:

$$t_3(x, y, y') = (t_3(x, y, y) + t_3(x, y', y')) / 2.$$

The algorithm can be seen in Figure 5.

TABLE VIII. COMPARISON OF SOME FAULT SIMULATION RESULTS

Circuit name	No. of simulation runs			Speed-up factor	Error
	Number of simulation runs for all possible fault combinations	Number of simulation runs with monotonicity assumption	Number of simulation runs for the new approach with fault classes	Our approach over simulation with monotonicity assumption	Our approach over fault simulation with monotonicity assumption
Two stage BJT amplifier with feedback (Fault depth 1-4)	22422	356	284	1.25	5.4 %
LM741 AMP [23] (Fault depth 1-4)	3923175	2090	1718	1.22	0.5 %
Broadband VHF/UHF amplifier [24] (Fault depth 1-3)	695525	18187	10928	1.66	1.8 %
Limiter BSP [25] (Fault depth 1-4)	1045256	1208	858	1.40	0.2 %
Voltage stabilizer circuit I (Fault depth 1-3)	8358	4088	2688	1.53	0.15 %
Voltage stabilizer circuit II (Fault depth 1-4)	317248	11173	5209	2.14	0.10 %
				Average: 1.53	Average.: 1.28 %

Procedure 2 Search Fault Class Chains

for all pairs (TC_2, TC_2') of tolerance classes with two injections do
 if $d(TC_2(x, y), TC_2(x', y')) = 1$ and $|t_2(x, y) - t_2(x', y')| \geq \Delta$ then
 { fault class $FC = FC_3(x, y, y)$,
 fault class $FC' = FC_3(x, y, y')$,
 fault class $FC'' = FC_3(x, y', y)$;
 $t_3(x, y, y) = \text{simulation of } FC_3(x, y, y)$;
 $t_3(x, y', y) = \text{simulation of } FC_3(x, y', y)$;
 $t_3(x, y, y') = (t_3(x, y, y) + t_3(x, y', y)) / 2$;
 }

Figure 5. Search Fault Class Chains.

C. Step 3 – Calculation of Probabilities

The simulations of $FC_3(x, y, y)$ and $FC_3(x, y', y')$ deliver the set of all tolerated test cases, this means the two tolerance classes $TC_3(x, y, y)$ and $TC_3(x, y', y')$. The probability of tolerating the respective triple faults can be calculated by the formula presented in Section III. When this formula is applied to tolerance class $TC_3(x, y, y)$ we obtain

$$\sum_{i^* \in TC_3(x, y, y)} \prod_{x \in i^*} (1 - P(x|E)) \cdot Q(x|F) \cdot \prod_{y \in (TC_3(x, y, y)) \setminus i^*} P(y|E)$$

For tolerance class $TC_3(x, y', y')$ we obtain:

$$\sum_{i^* \in TC_3(x, y', y')} \prod_{x \in i^*} (1 - P(x|E)) \cdot Q(x|F) \cdot \prod_{y \in (TC_3(x, y', y')) \setminus i^*} P(y|E)$$

The probability of tolerating the triple faults of the interpolated fault class cannot be obtained directly, because the test cases of this class have not been simulated. For this reason, we approximate the probability by multiplying the respective formula with the tolerance degree:

$$t_3(x, y, y') \cdot \sum_{i^* \in TC_3(x, y, y')} \prod_{x \in i^*} (1 - P(x|E)) \cdot Q(x|F) \cdot \prod_{y \in (TC_3(x, y, y')) \setminus i^*} P(y|E)$$

The tolerance class of the non-simulated fault class is generated by selecting a portion of $t_3(x, y, y')$ test cases at random. For the injection of more than three joint faults, steps 1 to 3 can be applied accordingly.

VII. EXPERIMENTAL RESULTS

In this section, the efficiency of the proposed solution to reduce the simulation time is evaluated. The fault simulation framework [4] is used to evaluate the dependability of five example electronic circuits. It should be noted that for the used circuits only permanent faults (e.g., short, open or parameter deviations) have been considered. The simulation time (fault injection and simulation) depends on the number of elements, the number of injected faults per element and the fault depth. Appropriate fault tolerance criteria have been defined on circuit outputs.

All of the circuits have been evaluated in two ways. The first evaluation was without generation of fault classes (all multiple fault combinations have been simulated with the monotonicity assumption). The second evaluation applied the new method with fault classes (therefore, only a portion of the test cases needed to be simulated). The remaining fault classes (which have not been simulated) have been evaluated by interpolation according to the algorithm in steps 1 to 3. This way the new method can be compared directly to the solution without using fault classes.

The result of the comparison of some simulations results is shown in Table VIII. The second to last column shows that the speedup achieved by the new approach is 50% in the average (see bottom line of Table VIII: "Average 1.53"). It has to be paid by an error in the results (see last column). The error refers to the absolute value of the fraction "result with new method" / "result without new method". A deviation around 1.3% is noticed in the average (see bottom line of Table 8: "Average 1.28 %").

VIII. CONCLUSION

Fault simulation of analogue circuits with multiple faults is an important problem to deal with, since their appearance

is unavoidable in real systems. In this paper, we have introduced the fault class concept for our approach to reduce the simulation time for multiple fault analysis. We discussed the idea of fault classes, providing conditions that ensure chains of fault classes with ascending or descending degree of tolerance. We implemented the procedure and evaluated it experimentally. In this paper, we have successfully reduced the duration of software-based fault simulation for multiple faults and different fault modes. In the evaluated example circuits, our methodology shows that the number of simulation runs is significantly lower while preserving the precision quite well.

REFERENCES

- [1] E. Weber and K. Echtele, "Efficient Simulation of Multiple Faults for Reliability Analysis of Analogue Circuits," in *DEPEND 2015. The Eighth International Conference on Dependability*, 2015, pp. 23–28.
- [2] 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*, 1999, pp. 244–248.
- [3] H. Spence, "Automatic analog fault simulation," in *Conference Record. AUTOTESTCON '96*, 1996, pp. 17–22.
- [4] E. Weber and K. Echtele, "Simulation-Based Reliability Evaluation for Analog Applications," in *2014 IEEE International Reliability Physics Symposium (IRPS)*, The Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08855, USA, 2014, 4B.2.1-4B.2.6.
- [5] P. Kabisatpathy, A. Barua, and S. Sinha, *Fault Diagnosis of Analog Integrated Circuits*. Boston, MA: Springer, 2005.
- [6] *Microwave Office | NI AWR Design Environment*. Available: <http://www.awrcorp.com/de/products/microwave-office> (2016, Feb. 01).
- [7] J. Shen, W. Maly, and F. Ferguson, "Inductive Fault Analysis of MOS Integrated Circuits," *IEEE Des. Test. Comput.*, vol. 2, no. 6, 1985, pp. 13–26.
- [8] R. Rodriguez-Montanes, Bruls, E. M. J. G, and J. Figueras, "Bridging defects resistance in the metal layer of a CMOS process," *J Electron Test*, vol. 8, no. 1, 1996, pp. 35–46.
- [9] H.-J. Wunderlich, Ed, *Models in Hardware Testing: Lecture Notes of the Forum in Honor of Christian Landrault*. Dordrecht: Springer Science+Business Media B.V, 2010.
- [10] J. Hou and A. Chatterjee, "Concurrent transient fault simulation for analog circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 22, no. 10, 2003.
- [11] P. N. Variyam and A. Chatterjee, "FLYER: fast fault simulation of linear analog circuits using polynomial waveform and perturbed state representation," in *Tenth International Conference on VLSI Design*, 1997, pp. 408–412.
- [12] A. V. Gomes, R. Voorakaranam, and A. Chatterjee, "Modular fault simulation of mixed signal circuits with fault ranking by severity," *IEEE International Symposium on Defects and Fault Tolerance in VLSI Systems*, 1998, pp. 341–348.
- [13] H. Hashempour *et al.*, "Test time reduction in analogue/mixed-signal devices by defect oriented testing: An industrial example," *Design, Automation & Test in Europe*, 2011.
- [14] T. Markas, M. Royals, and N. Kanopoulos, "On distributed fault simulation," *Computer*, vol. 23, no. 1, 1990.
- [15] C. P. Ravikumar, V. Jain, and A. Dod, "Faster fault simulation through distributed computing," *Tenth International Conference on VLSI*, pp. 482–487, 1997.
- [16] S. Kajihara, T. Sumioka, and K. Kinoshita, "Test generation for multiple faults based on parallel vector pair analysis," *International Conference on Computer Aided Design (ICCAD)*, 1993, pp. 436–439.
- [17] H. H. Zheng, A. Balivada, and J. A. Abraham, *A Novel Test Generation Approach for Parametric Faults in Linear Analog Circuits: Proceedings / 14th IEEE VLSI Test Symposium, Princeton, New Jersey*. Los Alamitos, Calif: IEEE Computer Society Press, 1996.
- [18] K. Saab, N. Ben-Hamida, and B. Kaminska, "Parametric fault simulation and test vector generation," *Meeting on Design Automation*, 2000, pp. 650–656.
- [19] Y. C. Kim, V. D. Agrawal, and K. K. Saluja, "Multiple faults: modeling, simulation and test," *7th Asia and South Pacific Design Automation Conference*, pp. 592–597, 2002.
- [20] S. Spinks, "ANTICS analogue fault simulation software," in *IEE Colloquium on Testing Mixed Signal Circuits and Systems*, 1997, p. 13.
- [21] A. Birolini, *Reliability engineering: Theory and practice*, 7th ed. Heidelberg, New York: Springer, 2014.
- [22] Junwei Hou, *CONCERT: a concurrent transient fault simulator for nonlinear analog circuits*. New York, NY: Association for Computing Machinery, 1998.
- [23] National Semiconductor, *LM741 Operational Amplifier*. Available: <http://web.mit.edu/6.301/www/LM741.pdf> (2015, Mar. 05).
- [24] C. G. Gentzler and S. K. Leong, "Broadband VHF/UHF amplifier design using coaxial transformers," *High Frequency Electronics*, pp. 42–51, http://www.polyfet.com/HFE0503_Leong.pdf, 2003.
- [25] AWR Corporation, *Bipolar Limiting Amplifier Circuit*. Available: https://awrcorp.com/download/faq/english/docs/Getting_Started/Tonal_Analysis.html (2015, Mar. 05).