Efficient Simulation of Multiple Faults for Reliability Analysis 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 for multiple simultaneous faults and/or multiple failure modes per element in an analogue circuit. By arranging similar multiple faults in groups, some 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 experimentally.**

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. 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. Most existing fault simulators use the Simulation Program with Integrated Circuits Emphasis (SPICE) and modify SPICE net lists to represent faults [1] - [7]. The fault simulation software [8] 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 [9]. 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 individual optimization of reusable topologies.

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,

 ISBN: 978-1-61208-429-9

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 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 circuit size and the number of faulty elements (fault depth) 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 [8] 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. 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 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, because experience has turned out that in practice there are only few exceptions. This monotonicity does not always exist. Instead we have observed that it exists in overwhelming majority of cases with only very few exceptions.

III. STRATEGIES FOR REDUCING SIMULATION TIME

To reduce the runtime for simulation with fault injection the following two general approaches are possible: reduce the number of test cases (simulation runs) or speed up the simulation procedure for each test case. 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. A general rule (if applicable) is the assumption of monotonic behaviour (see previous section). 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 ideal output. The monotony 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 fault depths (see Figure 1). In general, the monotony assumption reduces the number of both considered elements and failure modes per element.

In the remainder of the paper, we present a further method how the number of simulation runs can be reduced, see Sections 4 and 5. 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 the new method is making use of.

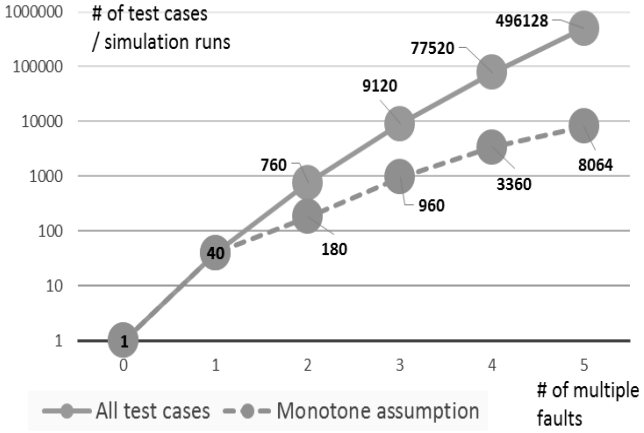

Figure 1. Complexity of fault simulation for an example medium sized circuit (20 elements with two fault modes per element).

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 = {transistor1, transisitor2, ..., resistor1, ... , … } is the set of elements of the circuit c_0.
3) F = {short_circuit, open_circuit, parameter_modification, ... } is the set of considered fault modes of the circuit c_0.
4) I = { (f, e) ∈ F × E : probability of fault f 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 ≠ 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:
$\Sigma_{f \in F}: Q(f, e) = 1$.

Example: If we assume only two fault modes F = {open, short} and only two elements $E = \{R_1, R_2\}$, there may be four injections $I = \{(open, R_1), (open, R_2), (short, R_1), (short, R_2)\}$ and four double injections. In all we obtain:
$I^* = \{ \{(open, R_1)\}, \{(open, R_2)\}, \{(short, R_1)\}, \{(short, R_2)\}, \{(open, R_1), (open, R_2)\}, \{short, R_1), (short, R_2)\}, \{(open, R_1), (short, R_2)\}, \{short, R_1), (open, R_2)\} \}$.
If shorts are more likely for R_1 and opens are more likely for R_2 we may get, say,
$Q(open, R_1) = 0.2$, $Q(short, R_1) = 0.8$ *(0.2 + 0.8 = 1).*
$Q(open, R_2) = 0.4$, $Q(short, R_2) = 0.6$ *(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 monotony.
$T_2 = \{i^* \in I_2 : R(i^*) = 1\}$ 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 monotony.
$T_3 = \{i^* \in I_3 : R(i^*) = 1\}$ is the set of triple injections that have been tolerated.
$P_3 = \sum_{i^* \in I_3} R(i^*) \cdot \prod_{x \in i^*} \big(1 - P(x|E)\big) \cdot Q(x|F) \cdot \prod_{y \in (I_3 \setminus i^*)} P(y|E)$
expresses the probability of tolerated triple injections.

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

IV. FAULT CLASS ALGORITHM

Our new algorithm is an heuristic approach that is based on an observation of simulation results [8] 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 fault modes, independent of the elements 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 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 $FC_n(x_1, x_2, \ldots, x_n)$ we require $x_1 \leq x_2 \leq \ldots \leq x_n$. 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: $FC_n(x_1,\ldots, x_n) = \{\{(f_1, e_1),\ldots, (f_1, e_1)\} \in I_n : f_i = x_i\}$.

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

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

The heuristic approach is defined in the following steps and the algorithm is shown in Figures 2 and 3. We assume that the tolerance classes $TC_1(\ldots)$ and $TC_2(\ldots)$ have already been generated by the respective fault simulations. Consequently, the tolerance degrees $t_1(\ldots)$ and $t_2(\ldots)$ are known. Then the following steps describe how the fault classes $FC_3(\ldots)$ 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 T_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 2. In the algorithm we denote the fault mode of injection x by x|F.

Procedure 1 Generate Fault Classes

```
for all test cases tc2 ∈ TC2 do
  for all test cases tc1 e TC1 do
  {  test case {x, y, z} = i ∪ j;
     if x|E ≠ y|E and x|E ≠ z|E and y|E ≠ z|E then
        FC3(x|F, y|F, z|F) = FC3(x|F, y|F, z|F) ∪ {x, y, z}
  }
```

Figure 2. 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 ≠ 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:
$< FC_3(x, y, y),\ FC_3(x, y, y'),\ FC_3(x, y', y') >$

According to the observation of likely monotonicity (see beginning of 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 from Figure 3.

Procedure 2 Search Fault Class Chains

```
for all pairs (TC2, TC2') of tolerance classes with two injections do
  if d(TC2(x, y), TC2(x', y')) = 1 and |t2(x, y) – t2(x', y')| ≥ Δ then
  { fault class FC = FC3(x, y, y),
    fault class FC' = FC3(x, y, y'),
    fault class FC'' = FC3(x, y', y');
    t3(x, y, y) = simulation of FC3(x, y, y);
    t3(x, y', y') = simulation of FC3(x, y', y');
    t3(x, y, y') = (t3(x, y, y) + t3(x, y', y')) / 2;
  }
```

Figure 3. 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:

$$t3(x, y, y') \cdot \sum_{i^* \in TC_3(x,y,y')} \prod_{x \in i^*} (1 - P(x|E)) \cdot Q(x|F) \cdot \prod_{y \in (TC_3(x,y,y') \setminus i^*)} P(y|E)$$

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.

V. Experimental Results

In this section, the efficiency of the proposed solution to reduce the simulation time is evaluated. The fault simulation framework [8] is used to evaluate the dependability of four example electronic circuits. It should be noted, that for 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

TABLE I. COMPARISON OF EXPERIMENTAL 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 [20] (Fault depth 1-4)	3923175	2090	1612	1.30	0.6 %
Broadband VHF/UHF amplifier [21] (Fault depth 1-3)	695525	18187	10928	1.66	1.8 %
Limiter BSP [22] (Fault depth 1-4)	1045256	1208	758	1.59	2.7 %
				Average: 1.45	Average.: 2.62 %

faults per element and the fault depth. Appropriate fault tolerance criteria have been defined on circuit outputs.

All of the circuits have been evaluated twice: The first evaluation was without generation of fault classes (chains have not been formed and all test cases have been simulated with the monotonicity assumption). The second evaluation applied the new method with fault classes (only a portion of the test cases has been simulated). The remaining ones 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 fault classes.

The result is shown in table 1. The last but one column shows the speedup achieved by the new approach: 45% in the average. It has to be paid by an error in the results (see last column). The error refers to the number of tolerated test cases. A deviation of 2.62% has been noticed in the average.

VI. CONCLUSION

Fault simulation of analogue circuits with multiple faults is an important problem to deal with, since multiple faults appearance is unavoidable in real systems. In this paper we have introduced the fault class concept for our approach to reduce the simulation time of multiple fault analysis. We discussed the idea of faults 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] 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.

[2] J. Jagodnik and M. Wolfson, “Systematic fault simulation in an analog circuit simulator,” vol. 26, no. 7, pp. 549–554, 1979.

[3] Y. Cao, Z.-h. Cen, J.-l. Wei, X. Ma, B. Yang, and M. Li, “FDSAC-SPICE: fault diagnosis software for analog circuit based on SPICE simulation,” in *International Conference on Space Information Technology 2009*: SPIE, 2010, pp. 765120–765120-8.

[4] C. Sebeke, J. P. Teixeira, and M. J. Ohletz, “Automatic fault extraction and simulation of layout realistic faults for integrated analogue circuits,” in *the European Design and Test Conference. ED&TC 1995*, pp. 464–468.

[5] S. Spinks, “ANTICS analogue fault simulation software,” in *IEEE Colloquium on Testing Mixed 23 Oct. 1997*, p. 13.

[6] Bernd Straube, Bert Müller, Wolfgang Vermeiren, Christoph Hoffmann, Sebastian Sattler, “Analogue Fault Simulation by aFSIM,” *DATE 2000 - User Forum, Paris*, 2000.

[7] H. Spence, “Automatic analog fault simulation,” in *Conference Record. AUTOTESTCON '96*, pp. 17–22.

[8] Weber E, Echtle K, and 52nd IEEE International Reliability Physics Symposium, IRPS 2014, “Simulation-based reliability evaluation for analog applications,” (English), *IEEE Int. Reliab. Phys. Symp. Proc. IEEE International Reliability Physics Symposium Proceedings*, 2014.

[9] P. Kabisatpathy, A. Barua, and S. Sinha, *Fault Diagnosis of Analog Integrated Circuits*. Boston, MA: Springer, 2005.

[10] Junwei Hou and A. Chatterjee, “Concurrent transient fault simulation for analog circuits,” *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst*, vol. 22, no. 10, pp. 1385–1398, 2003.

[11] P. N. Variyam and A. Chatterjee, “FLYER: fast fault simulation of linear analog circuits using polynomial waveform and perturbed state representation,” *Tenth International Conference on VLSI 4-7*, pp. 408–412, 1997.

[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*, pp. 341–348, 1998.

[13] H. Hashempour, J. Dohmen, B. Tasić, B. Kruseman, C. Hora, M. van Beurden, and Yizi Xing, “Test time reduction in analogue/mixed-signal devices by defect oriented testing: An industrial example,” *Design, Automation & Test in Europe*, 2011.

 ISBN: 978-1-61208-429-9

[14] T. Markas, M. Royals, and N. Kanopoulos, “On distributed fault simulation,” *Computer*, vol. 23, no. 1, pp. 40–52, 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)*, pp. 436–439, 1993.

[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*, pp. 650–656, 2000.

[19] Yong Chang 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] National Semiconductor, *LM741 Operational Amplifier*. Available: http://web.mit.edu/6.301/www/LM741.pdf (2015, Mar. 05).

[21] 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 and https://awrcorp.com/download/faq/english/appnotes/uhf_vhf_amplifier.aspx, 2003.

[22] AWR Corporation, *Bipolar Limiting Amplifier Circuit.* Available: https://awrcorp.com/download/faq/english/docs/Getting_Started/Tonal_Analysis.html (2015, Mar. 05).

 ISBN: 978-1-61208-429-9