INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UNIVERSITÉ DE MONTRÉAL

OUTIL AUTOMATIQUE DE TEST DE CIRCUITS ANALOGIQUES

KHALED SAAB
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
ET DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIAE DOCTOR
(GÉNIE ÉLECTRIQUE)
DÉCEMBRE 1999

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée:

OUTIL AUTOMATIQUE DE TEST DE CIRCUITS ANALOGIQUES

présentée par: SAAB Khaled

en vue de l'obtention du diplôme de:Philosophiae doctor

a été dûment acceptée par le jury d'examen constitué de:

M. SAVARIA Yvon, Ph.D., président

Mme KAMINSKA Bozena, Ph.D. membre et directeur de recherche

M. HOULE Jean-Louis, Ph.D. membre

M. THIBEAULT Claude, Ph.D membre

A mes parents

REMERCIEMENTS

Je tiens tout d'abord à remercier Madame Bozena Kaminska pour ses conseils, ses critiques constructives et surtout pour la confiance qu'elle a mise en moi, ce qui m'a permis de réaliser ce travail.

Je souhaiterais également adresser mes remerciements aux membres du jury de mon examen pré-doctoral, en particulier Messieurs Jean-Jules Brault, Guy Bois et Claude Thibeault, pour leurs conseils et critiques qui ont fait avancer mon travail en dirigeant mes recherches de manière méthodique.

Je remercie aussi tous les étudiants du laboratoire VLSI et toute l'équipe d'Opmaxx Inc. qui ont contribué de près ou de loin à ce projet et en particulier: Dr Naim Ben Hamida pour son aide et son soutien tout au long de cette thèse, Jim Abbate, Patryk Lech, John Peyton, Anita Govindarajan et Thomas Foy pour tout l'effort et le temps qu'ils ont consacrés à l'implémentation des algorithmes, Olivier et Sandrine Ganry et Laurent et Maryline Bordes pour leur aide à la rédaction et à la correction de cette thèse.

Finalement, j'adresse un remerciement spécial à ma femme Sanaa pour son support et sa patience pendant ces longues et nombreuses fins de semaines que j'ai passé à étudier et à rédiger cette thèse.

RÉSUMÉ

La déviation des caractéristiques d'une composante dans un circuit analogique se reflète par la déviation d'un ou plusieurs paramètres de sortie. La relation qui existe entre les deux est définie par la sensibilité.

L'étude de la sensibilité réalisée par Mustapha Slamani et Naim Ben Hamida [1]-[9] a été adaptée pour l'analyse de la testabilité ainsi que pour la génération des vecteurs de tests de circuits analogiques.

Pour de petits circuits, cette analyse pourrait se faire manuellement, mais pour un circuit de taille moyenne, qui peut comprendre plusieurs milliers de composants, une certaine automatisation est nécessaire. La technique de calcul de sensibilité par la méthode des circuits adjoints s'avère appropriée.

La sensibilité est par la suite utilisée comme base pour la génération de vecteurs de test pour les fautes paramétriques. En effet, une nouvelle méthode de test basée sur une approche probabiliste est utilisée. Cette méthode prend en considération la variation des composantes de leur valeur nominale, ainsi que la corrélation entre ces composantes pour générer une distribution nominale et une distribution fautive du paramètre de sortie. Le vecteur de test, pour lequel la probabilité de prendre une fausse décision sur l'état du circuit est minimale, est alors considéré comme le vecteur de test nominal pour le composant en question.

D'une manière similaire, la sensibilité a été utilisée pour la génération de vecteurs de test pour les pannes catastrophiques (i.e. des pannes dues à un ajout ou à un manque de métal dans le circuit intégré). L'approche proposée combine l'efficacité de la méthode des circuits adjoints (nombre réduit de simulations) et une analyse de premier ordre des fautes catastrophiques.

Or, le temps de test se traduit directement par un coût monétaire qui devient de plus en plus élevé avec l'augmentation de complexité des circuits VLSI. Il devient donc nécessaire de minimiser le nombre de vecteurs de test appliqués au circuit, tout en gardant une couverture de fautes aussi proche que possible de la couverture nominale. Ce problème de compaction de vecteurs de test a été formulé en un problème d'optimisation combinatoire.

En général, la couverture obtenue est relativement faible, surtout lorsqu'un nombre minimal de ports d'accès est disponible. En effet, l'augmentation de la densité des circuits VLSI actuels et la limitation des ports rendent l'accès à des nœuds internes très difficile et diminuent donc la possibilité d'isoler les composants dans le but de test, ce qui se traduit par une faible couverture de fautes. Pour répondre à ces besoins, un algorithme automatique d'insertion de points de test a été développé dans le but d'augmenter l'observabilité et donc la couverture des fautes dans les circuits analogiques.

Les algorithmes développés lors de cette thèse offrent les possibilités suivantes:

- *Analyse automatique de la sensibilité* d'un circuit analogique;

- *Génération des vecteurs de test pour des fautes paramétriques* ("soft faults");

- *Génération des vecteurs de* test *pour des fautes catastrophiques* ("hard faults");

- *Compaction de vecteurs de test;*

- *Insertion de points de test.*

ABSTRACT

In analog circuits, deviation in component values causes deviation in the measured output parameters. The relation between the two deviations is the sensitivity function.

Mustapha Slamani and Nain B. Hamida [1]-[9] proposed and developed the sensitivity analysis approach for testability analysis and test vector generation in analog circuits. This theory had been manually validated and applied for small circuit testing. For large circuits (several thousands of components), the manual approach is not valid, and some automation is needed. The adjoint network method for automatic sensitivity analysis seems the most appropriate. Indeed, the efficiency of the adjoint network method is due to its ability to measure the sensitivity of any network parameter with respect to all possible component variations with only two simulations: one for the initial network and one for the corresponding adjoint network.

Once the sensitivity computation algorithm was in place, soft fault modeling, simulation, and test vector generation became a reality: Circuit sensitivity could then be used to generate a statistical model of the circuit. In fact, two statistical models are generated: one for the fault-free circuit and one for the faulty circuit. The statistical models of the fault free circuit and the faulty circuit are then used to estimate the probability of accepting a faulty circuit, and the probability of rejecting a good circuit. The obtained information is then used to measure and to quantify the circuit testability.

Similarly to soft faults, hard fault modeling, simulation, and test vector generation where addressed. The proposed approach allows parallel fault simulation and test vector specification based on effect-cause analysis. From the distance between the fault-free circuit statistical model and the faulty circuit statistical model (effect), we approximate the fault (cause) value for all modeled faults simultaneously by linear estimation. Here again, the statistical models of the fault-free circuit and the faulty circuit are used as a measurement to quantify the circuit testability.

However since test time is directly absorbed into the cost of the circuit, the reduction of the test vectors while maintaining high fault coverage become a must. This problem of test vector compaction was formulated as a combinatorial optimization problem.

To increase the hard-to-detect-faults fault coverage and to minimize the number of test vectors, additional test points could be added to the circuit. However, adding test points could have negative effects on circuit performances, and should be taken into consideration. In order to consider all those constraints, a test points insertion algorithm that assess the impact of test points on circuit performances and on the overall circuit testability was developed.

The proposed algorithms, which were developed during this thesis, offer the following possibilities:

- *Automatic sensitivity analysis of any time continuous analog circuit;*
- *Soft fault test vector generation;*
- *Hard fault test vector generation;*
- *Test vector compaction;*
- *Test point insertion.*

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

CHAPITRE 1

MÉTHODOLOGIE ET ALGORITHMES POUR LE TEST DE CIRCUITS ANALOGIQUES.

1.1 Introduction aux circuits analogiques et mixtes

Actuellement, les circuits analogiques sont très utilisés, mais la complexité et la densité de ces circuits en rendent la conception et le test très difficiles. Les algorithmes et les outils CAO étant pratiquement inexistants, le coût du développement de circuits analogiques devient de plus en plus prohibitif. Jusqu'à présent, les efforts ont principalement portés dans le domaine numérique. Actuellement, l'industrie utilise des outils qui permettent le design des circuits à partir d'un langage de haut niveau (VHDL) ainsi que la synthèse et l'implantation du test au niveau des circuits, cartes et systèmes numériques.

De plus, le développement de la technologie à un haut niveau d'intégration (VLSI: "Very Large Scale Integration") offre la possibilité d'intégrer un système complet sur la même puce, où le nombre de composants peut atteindre plusieurs millions de transistors. Pour plusieurs applications telles que les télécommunications, l'instrumentation et le traitement d'images, les systèmes VLSI nécessitent l'implantation d'une partie analogique et d'une partie numérique dans le même circuit intégré et ceci en utilisant la même technologie.

En pratique, la partie analogique de ces systèmes représente 20 à 30% de la surface totale du circuit. Par contre, la complexité relative des circuits analogiques, associée au manque d'outils performants, rend la conception et le test des circuits analogiques itératifs, lents et, par conséquent, coûteux.

Les recherches actuelles, incluant le travail présenté dans cette thèse, visent à développer une nouvelle classe d'outils et d'approches qui couvrent la testabilité des circuits analogiques, en y incluant la conception axée sur la vérification systématique ("DFT: Design for Testability"). Ces développements ont pour objectif de faciliter la tâche des ingénieurs de conception et de test, en leur permettant de concentrer leurs efforts sur l'innovation. De plus, ces techniques offrent une amélioration au niveau coût, qualité et flexibilité dans la fabrication des produits à hautes performances.

1.2 Problématique du test des circuits analogiques

Les circuits VLSI présentent plusieurs avantages. Par contre, la complexité et la densité des circuits VLSI atteintes aujourd'hui exigent des méthodes de test sophistiquées, dont le coût est de plus en plus élevé. Plusieurs raisons expliquent la complexité de l'analyse des défauts et la génération des vecteurs de test [12], [13]:

1. La nature continue des fautes analogiques, qui engendrent une déviation de la valeur nominale et le manque de modèles de fautes robustes et pratique comme le collage, qui est très répandu dans le domaine du test numérique.

2. La nature continue des paramètres de sortie des circuits analogiques, tels que le gain, la phase, la fréquence de coupure, le taux de rejection du mode commun, etc.

   L'utilisation des méthodes déterministes de calcul de couverture de fautes n'est pas toujours efficace. Pour un circuit numérique, la présence d'une anomalie modifiera généralement la signature binaire du circuit. La réponse 1/0 obtenue correspond à une probabilité de 100% ou à une probabilité nulle de détection de faute. Cependant, la détection d'erreur pour les circuits analogiques n'est pas aussi simple à réaliser que pour les circuits numériques. En effet, en raison de la nature continue des signaux analogiques, il est impossible d'obtenir, comme pour les circuits numériques, une signature aussi radicale. Ceci peut donc entraîner une mauvaise interprétation de la couverture de fautes.

3. La non-linéarité des caractéristiques des circuits analogiques. En d'autres termes, si la valeur d'un composant change d'un facteur quelconque k, la réponse ne changera pas nécessairement avec le même facteur.

4. La haute sensibilité des performances du circuit aux capacités parasites sur les nœuds internes. Ce phénomène diminue de manière significative l'observabilité et la contrôlabilité du circuit.

1.3 Types de fautes dans les circuits analogiques

Pour les circuits analogiques, principalement deux types de fautes existent: en général, une panne peut se présenter sous forme d'une panne catastrophique ou sous forme d'une panne paramétrique [1].

Les pannes paramétriques peuvent provenir:

1. De la déviation de la composante (résistance, capacité, bobine, source contrôlée, etc.) de sa valeur nominale.

   Ce type de variation est dû à une tolérance de fabrication, au vieillissement, à la variation de la température, au changement environnemental ou à un changement des conditions d'opération.

2. De la différence entre le comportement du modèle mathématique (utilisé dans la phase de conception du circuit) et le comportement du composante elle-même. En effet, cette différence peut provenir de l'inexactitude de mesure, d'une mauvaise calibration des sondes, de la présence de bruit ou du changement des conditions initiales (les conditions initiales supposées au moment de la mesure n'ont pas été conservées lors de la simulation).

3. De l'inexactitude et des simplifications apportées au modèle (inductance et capacité parasite).

   Celles-ci sont utilisées pour diminuer la complexité du circuit de façon à en simplifier l'analyse et à accélérer le temps de simulation. Ce type de simplification est couramment utilisé. Pourtant, cette procédure est accompagnée d'un certain risque car les performances réelles du circuit peuvent différer des performances prédites à partir des modèles mathématique. En effet, il est connu que la simplification apportée peut engendrer une erreur et même une instabilité dépendant de la sensibilité du système par rapport aux paramètres négligés [10].

En revanche, les pannes catastrophiques sont généralement causées par un court-circuit ou un circuit ouvert.

1.4 Approche existante

En général, le test d'un circuit intégré comprend deux étapes principales: la première consiste à générer des vecteurs de test, et la seconde, à analyser la réponse à la sortie de celui-ci. La vérification fonctionnelle des paramètres et des signaux de sortie permet la détection des différents défauts dans le circuit. Cependant, un circuit ne peut pas être testé uniquement en vérifiant sa fonctionnalité [2]-[8]. En effet, le test fonctionnel ne permet pas de:

1. Calculer une couverture de fautes.

2. Quantifier l'efficacité des vecteurs de test pour déterminer s'ils sous-testent ou sur-testent le circuit.

3. Éliminer les vecteurs de test redondants.

1.5 Approche proposée

Une des alternatives au test fonctionnel sera le test du circuit au complet, composante par composante. En effet, si l'on est certain que les valeurs de toutes les composantes du circuit se situent à l'intérieur de leur tolérance, on peut garantir que la fonctionnalité du circuit sera respectée [3] et [6]. Cette méthode vise à augmenter l'observabilité de la composante sous-test et ce, en se basant sur une analyse de la sensibilité. En effet, la sensibilité permet d'estimer la variation des paramètres d'un circuit ou d'un système par rapport à la variation des composantes.

Pourquoi la sensibilité?

1. La sensibilité permet d'isoler l'effet de chaque composante sur la sortie. A partir de la distribution des composantes, on peut alors calculer la distribution des paramètres de sortie nominaux (sans panne) et fautifs (avec panne). Une fois les distributions obtenues, on procède à la génération de vecteurs de test, ce qui implique de trouver le stimulus qui maximisera l'observabilité de la panne et minimisera la probabilité de prendre une mauvaise décision sur l'état du circuit.

2. La sensibilité permet également d'effectuer une analyse de l'effet de charge de chaque nœud dans le circuit sur les paramètres de celui-ci. Cette analyse est très utile pour la conception pour la testabilité (CPT) et notamment pour l'insertion de points de test.

Gardons à l'esprit, que l'approche proposée dans la thèse est valide en théorie pour les circuits linéaires seulement. Bien que non prouvée mathématiquement, la méthode a été généralisée pour les circuits non linéaires. En effet, cette approche s'est révélée correcte dans plusieurs cas de circuits non linéaires tels que ceux démontrés au cours de cette thèse.

1.6 Objectif

Notre premier objectif est de développer une nouvelle approche pour le test des circuits analogiques. La méthode proposée est basée sur le calcul des sensibilités des composantes. Cette approche doit prendre en considération les deux types de fautes possibles:

1. Les fautes paramétriques à petite déviation.

2. Les fautes catastrophiques causées par un court-circuit ou un circuit ouvert.

Une fois la couverture de fautes obtenue, notre deuxième objectif repose sur la réduction du nombre des vecteurs de test (compaction des vecteurs de test) et sur l'insertion de points de test (Figure 1.1).

Dans les chapitres suivants, nous présenterons une méthode efficace de calcul de sensibilité ainsi que deux utilisations de cette méthode. La première utilisation consiste à générer des vecteurs de test pour les fautes paramétriques, alors que la deuxième consiste à générer des vecteurs de test pour les fautes catastrophiques. Par la suite, nous présenterons une méthode de compaction des vecteurs de test utilisant un algorithme d'optimisation tel que le recuit-simulé. Ce même algorithme sera également utilisé comme base d'un module de sélection de points de test. Dans le dernier chapitre, une conclusion résumera l'ensemble du travail présenté tout au long de cette thèse.

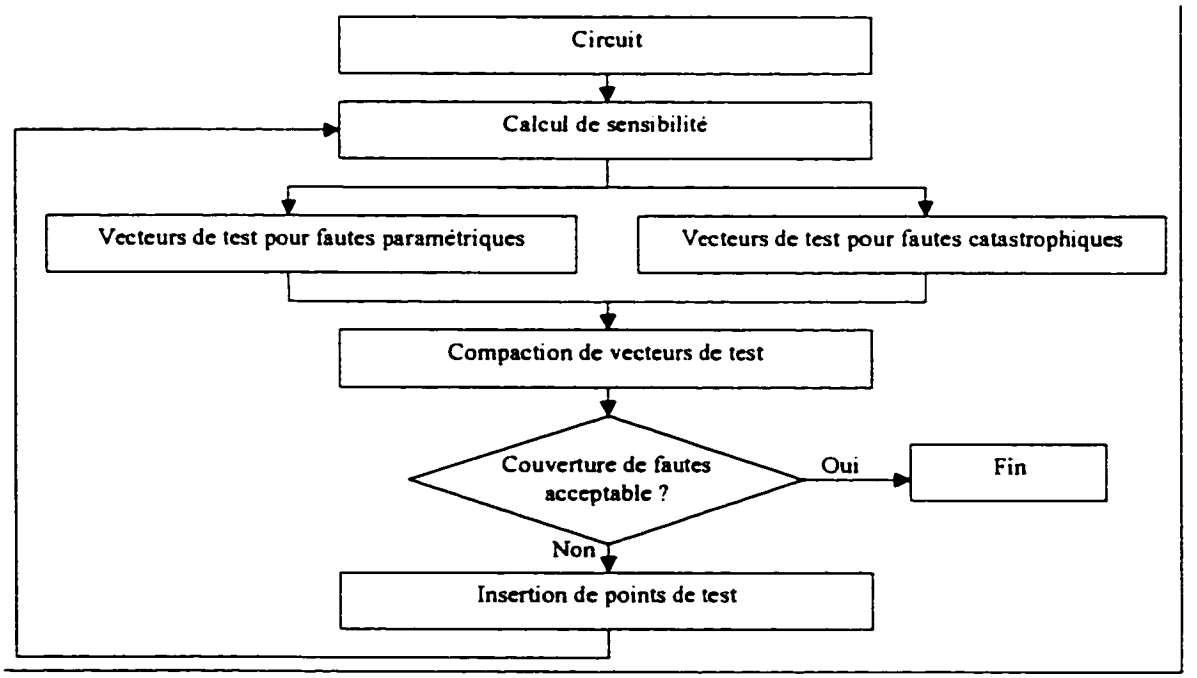

Figure 1.1: Méthodologie

CHAPITRE 2

MÉTHODE AUTOMATIQUE DE CALCUL DE SENSIBILITÉ

Dans les circuits analogiques, la déviation paramétrique des composants de leur valeur nominale cause une déviation des paramètres de sortie. La relation entre les deux déviations peut être modélisée par la fonction de sensibilité. En effet, la sensibilité peut être considérée comme une relation de premier ordre (linéarisation) qui permettra de calculer l'effet de la variation des composantes x_i sur les performances W_k (gain, phase, etc.) du circuit (Figure 2.1)

où:

W_k est le k-ième paramètre de sortie $k = 1 \cdots K$

x_i le i-ième composant dans le circuit $i = 1 \cdots M$.

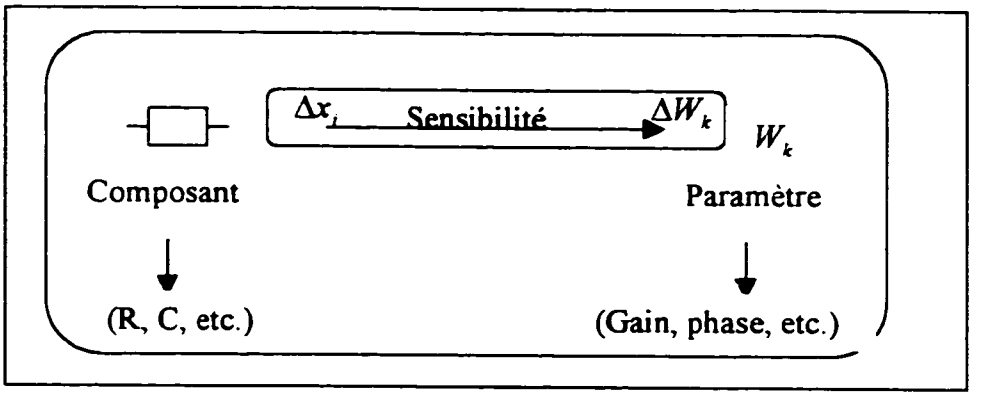

Figure 2.1: Relation composant / performance

De nombreuses méthodes pour le calcul numérique de la sensibilité ont été proposées dans les années 60 [18]. Pour nommer quelques-unes des plus importantes, on peut mentionner:

- la méthode directe,
- la méthode symbolique,
- la méthode incrémentale,
- la méthode adjointe.

La méthode des circuits adjoints sera utilisée comme méthode de base pour le calcul de la sensibilité. Ce choix de méthode est justifié par la puissance de celle-ci et par la possibilité qu'elle offre de calculer la sensibilité de(s) paramètre(s) de sortie du circuit par rapport à la variation de tous les paramètres composants de ce circuit en deux simulations seulement: une pour le circuit initial et l'autre pour le circuit adjoint. De plus, la méthode adjointe a été adaptée pour l'analyse de sensibilité dans le domaine AC, DC et transitoire [19]-[22]. D'autres publications ont montré l'adaptation de la méthode

des circuits adjoints pour l'optimisation automatique des circuits et pour le calcul de bruit [23] et [24].

L'article qui suit, présente une description détaillée de la méthode des circuits adjoints. Il démontre comment on peut utiliser les sensibilités pour la génération des vecteurs de test pour les fautes paramétriques, ainsi qu'un échantillon représentatif des résultats obtenus après implémentation. Deux types de résultats sont présentés:

1. Les courbes de sensibilité obtenues dans les trois domaines (AC, DC et transitoire) et ceci pour différents types de composants linéaires tels que les résistances et les capacités et pour des composants non-linéaires tels que les transistors MOS (longueur et largeur).

2. Les courbes de déviation minimale détectable (définies par la plus petite déviation paramétrique du composante sous test qui causera des distributions distinctes entre le circuit nominal et le circuit fautif). Ces courbes sont utilisées pour la génération des vecteurs de test qui ciblent trois états possibles du circuit, soient:

    - Le cas nominal: l'effet de la déviation des composantes de leur valeur nominale (due à une fluctuation du processus de fabrication) sur le paramètre de sortie s'annule.

- Le pire des cas: tous les composants masquent l'effet du composant fautif.

- Le meilleur cas: toutes les composantes agissent pour rendre visible l'erreur d'un composant fautif.

Note: Dans la figure 2.14, la déviation détectable de la capacité C2 semble être erronée. En effet, on ne doit pas avoir une déviation détectable minimale (équation 2.7, 2.8 et 2.9) égale a zéro, car ceci implique qu'une déviation nulle (pas de déviation) sur le composant forcera les deux distributions à être disjointes, ce qui est impossible.

L'article a été accepté et publié à "IEE Proceedings Circuit Devices and Systems" Vol. 143, No. 6, en décembre 1996.

LIMSoft: Automated tool for sensitivity analysis and test vector generation

Khaled Saab, David Marche, Naim B. Hamida* and Bozena Kaminska

Ecole Polytechnique of the University of Montreal

P.O. Box 6079, Station "Centre-ville", Montreal, PQ, Canada, H3C 3A7

* OPMAX Inc, Montreal, PQ, Canada

Abstract

In analog circuits, deviation in component values cause deviation in the measured output parameters. The relation between the two deviations is the sensitivity function. This paper presents an automated sensitivity analysis tool called LIMSoft, based on the adjoint network method which offers the possibility of DC, AC and transient sensitivity computation. The sensitivity value is then used for soft fault test vector generation.

2.1 Introduction

The revival of industrial interest in analog integrated circuits has resulted from a new series of circuit functions and higher levels of performance accuracy. Unlike digital circuit designs, analog circuit designs are still hand-crafted by expert circuit designers, who have to alternate between simulation and manual adjustment of the circuit variables (e.g. resistor value, capacitor value, transistor size and area, etc.) in order to improve the performance of the circuit. This circuit adjustment process is time-consuming, since it is

based on circuit behavior and the relationship between circuit performance and the circuit variables.

The testing of analog circuits presents an even bigger challenge. The difficulty in analog-circuit testing results from their time-continuous nature and the lack of a robust analog fault model similar to the digital one (stuck-at-0 and stuck-at-1). However, two types of analog faults can be clearly distinguished: the soft or parametric fault [1] which is caused by component variation around the nominal value, fabrication tolerance, temperature drift, aging, etc., and the hard, or catastrophic, fault, which has to do with added short and open circuits.

It has been shown that the sensitivity concept can help in both analog circuit design [19] and analog circuit testing [1-7]. In fact, the sensitivity function is an appropriate relationship between circuit performance and circuit variables since it expresses the ratio of the relative deviation of the output parameter to the relative deviation of the circuit's elements.

A number of numerical methods for numerical sensitivity computation have been proposed: the brute force method, the symbolic method, the incremental method and the adjoint network method, to name a few. However, these methods have all been used mainly for circuit optimization during the design process. An automated tool for efficient design and test is still not available. SPICE [30], for example, offers the limited possibility of a single DC operation point sensitivity computation, but this is clearly not enough.

To accelerate circuit development, from the design and test point of view, a complete sensitivity analysis is indispensable. By complete analysis, we mean DC, AC and transient analysis. For this reason, we have conducted some theoretical and feasibility studies in our previous work [1-7], and the objective of this paper is to introduce our new, automated tool, called LIMSoft.

LIMSoft offers the possibility of DC, AC and transient sensitivity computation, fault simulation, test vector generation and test vector compaction. The sensitivity computation is based on the adjoint network method [18, 40]. This method was chosen because of its effectivenes and its ability to measure the sensitivity of network performance(s) efficiently with respect to all possible component variations (including topology) with only two simulations: one for the initial network and one for the corresponding adjoint network. In this paper, we show the tool for the sensitivity computation which is used for automatic generation of an optimal set of test vectors.

The structure of LIMSoft is given in Figure 2.2.

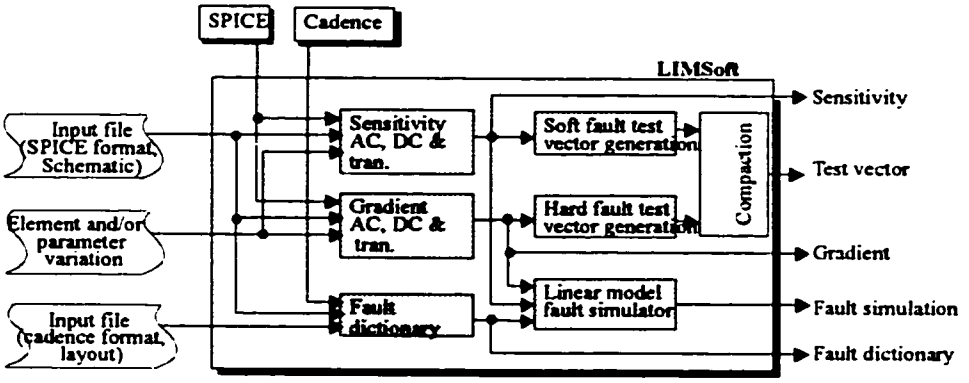

Figure 2.2: LIMSoft structure

The paper is organized as follows. The second section describes previous work and the adjoint network method for numerical sensitivity computation. The third section presents the sensitivity analysis tool. In the fourth section a method for soft fault test vector generation is described. Practical examples and results are also included in the sections 2.3 and 2.4.

2.2 Sensitivity computation and the adjoint network method

The sensitivity function (2.1) expresses the effect on circuit performance (out) due to the variation of some circuit elements x_i.

$$S_x^{out} = \frac{x}{out}\frac{\partial out}{\partial x} = \left.\frac{\frac{\Delta out}{out}}{\frac{\Delta x}{x}}\right|_{\Delta x \to 0} \tag{2.1}$$

LIMSoft uses the adjoint network method to compute the sensitivity of the output parameter (out) with respect to all x_i in the circuit simultaneously .

The branch voltage and current of the original network are denoted by V and I, and the branch voltage and current in the adjoint network are denoted by $\hat{V}$ and $\hat{I}$. A complete description of the adjoint network method is given in [19-22].

Since the adjoint circuits are defined for linear time invariant components (Table 2.1), any complex model (transistor, diode, nonlinear elements, op-amp., etc.) should be decomposed into basic components manageable by the adjoint network method. Thus, for nonlinear elements, such as a current-controlled or a voltage-controlled element, a piece-wise linear approach is used. At each iteration (time step for time analysis or voltage step for DC analysis), the operating point of the circuit is evaluated and the corresponding linearized model of each nonlinear component is updated according to (2.2) for a voltage-controlled nonlinear current source, and (2.3) for a current-controlled nonlinear voltage source [18].

$$\hat{I} = (\partial I/\partial V)\hat{V} \tag{2.2}$$

$$\hat{V} = (\partial V/\partial I)\hat{I} \tag{2.3}$$

Often the sensitivities of the various components of the transistor model (transistor gain, input/output impedance, parasitic capacitors, etc., Figure 2.3) may not all be of interest to the designer. But, by judiciously using the sensitivity values and the relationships between the model components and the related circuit, we are able to compute the sensitivity of the output parameter(s) not only with respect to the transistor component model, but also with respect to process parameter (P_P) variation and to

transistor size and area (W and L for MOS transistors and area size for BJT). Indeed, the sensitivity of the output parameter to a voltage-controlled nonlinear element is obtained by equation (2.4) and the sensitivity of the output parameter to a current-controlled nonlinear element is obtained by equation (2.5) [18].

$$S_{P_P}^{out} = -\hat{V}\left(\frac{\partial I}{\partial P_P}\right)\frac{P_P}{out} \tag{2.4}$$

$$S_{P_P}^{out} = -\hat{I}\left(\frac{\partial V}{\partial P_P}\right)\frac{P_P}{out} \tag{2.5}$$

For a MOSFET transistor, the drain current is described in terms of the terminal-applied voltages and process parameters as $I_{DS} = f_{DS}(V_{DS}, V_{GS}, V_{BS}, P_P)$, where P_P contains the process parameters, such as the channel length and channel width, and V_{DS}, V_{GS} and V_{BS} are the MOSFET drain-source, gate-source and substrate-source terminal-applied voltages respectively. The sensitivity of a circuit output with respect to MOSFET process parameter P_P is

$$\frac{dout}{dP_P} = \hat{V}_{DS}\left(\frac{\partial I_{DS}}{\partial P_P}\right) + \hat{V}_{GS}\left(\frac{\partial I_{GS}}{\partial P_P}\right) + \hat{V}_{BS}\left(\frac{\partial I_{BS}}{\partial P_P}\right) \tag{2.6}$$

where out can be the output voltage or output current; $\hat{V}_{DS}$, $\hat{V}_{GS}$ and $\hat{V}_{BS}$ are the terminal-applied voltages in the adjoint circuit; and I_{DS}, I_{GS} and I_{BS} are the drain-source, gate-source and substrate currents respectively [40].

Table 2.1: Conversion table and sensitivity computation in the Frequency Domain

Element Type	Description in the initial circuit	Description in the adjoint circuit	Sensitivity
R	$V = RI$	$\hat{V} = R\hat{I}$	$\frac{dout}{dR} = I\hat{I}$
G	$I = GV$	$\hat{I} = G\hat{V}$	$\frac{dout}{dG} = -V\hat{V}$
Z	$V = ZI$	$\hat{V} = Z\hat{I}$	$\frac{dout}{dZ} = I\hat{I}$
Y	$I = YV$	$\hat{I} = Y\hat{V}$	$\frac{dout}{dY} = -V\hat{V}$
C	$I = jwCV$	$\hat{I} = jwC\hat{V}$	$\frac{dout}{dC} = -jwV\hat{V}$
L	$V = jwLI$	$\hat{V} = jwL\hat{I}$	$\frac{dout}{dL} = jwI\hat{I}$
μ	$V_2 = \mu V_1$ $I_1 = 0$	$\hat{I}_1 = -\mu\hat{I}_2$ $\hat{V}_2 = 0$	$\frac{dout}{d\mu} = V_1\hat{I}_2$
g_m	$I_2 = g_m V_1$ $I_1 = 0$	$\hat{I}_1 = g_m\hat{V}_2$ $\hat{I}_2 = 0$	$\frac{dout}{dg_m} = -V_1\hat{V}_2$
β	$I_2 = \beta I_1$ $V_1 = 0$	$\hat{V}_1 = -\beta\hat{V}_2$ $\hat{I}_2 = 0$	$\frac{dout}{d\beta} = -I_1\hat{V}_2$
r_m	$V_2 = r_m I_1$ $V_1 = 0$	$\hat{V}_1 = r_m\hat{I}_2$ $\hat{V}_2 = 0$	$\frac{dout}{dr_m} = I_1\hat{I}_2$
T	$V_2 = nV_1$ $I_1 = -nI_2$	$\hat{V}_2 = n\hat{V}_1$ $\hat{I}_1 = -n\hat{I}_2$	$\frac{dout}{dn} = \hat{I}_2V_1 + \hat{V}_1I_2$

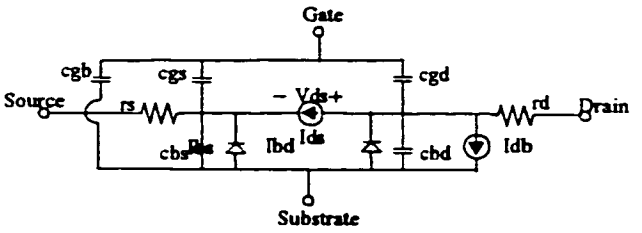

Figure 2.3:Equivalent transient MOSFET transistor model

2.3 Automatic sensitivity computation

LIMSoft is not a circuit simulator; it basically uses the existing simulator, e.g. SPICE, APLAC, etc., and manipulates the input/output files. The present version of LIMSoft is programmed to handle the SPICE format. Extension to include other formats is possible.

The procedure for calculating the sensitivity of an output (voltage and/or current) may be summarized as follows (see Figure 2.4):

- **Redraw the original network** by replacing all transistors and diodes by their models (SPICE nonlinear model). This step is imperative for the construction of the adjoint network.

- **Construct the adjoint network** by replacing all the network components by their adjoint components. Use Table 2.1 for linear elements and equations 2.2 and 2.3 for nonlinear elements.

- Run two **SPICE simulations**, one for the original circuit and one for the adjoint circuit, and obtain all node and branch voltages and currents respectively.

- **Evaluate the sensitivity** of the output parameter to all element variations according to Table 2.1 for linear elements, and to equations 2.4 and 2.5 for nonlinear elements [18, 19, 20, 21, 22, and 40].

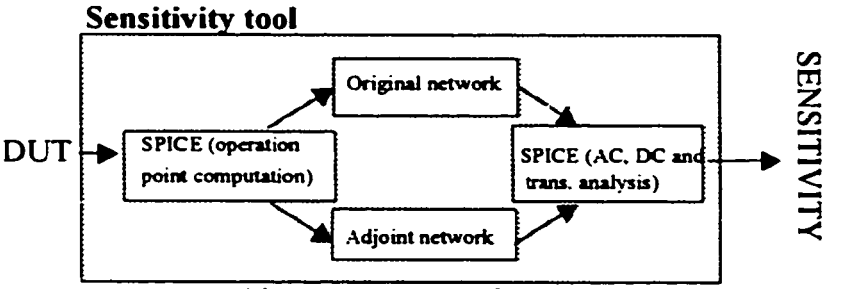

Figure 2.4:LIMSoft structure

2.3.1 Experimental results

The adjoint network method for sensitivity computation has been implemented in C programming language on a SPARC10 SUN workstation. The procedures outlined in this paper were applied to the second-order band-pass filter shown in Figures 2.5 and 2.6. The sensitivity results for the output gain in the DC, AC and transient domains are shown in Figures 2.7, 2.8 and 2.9 respectively.

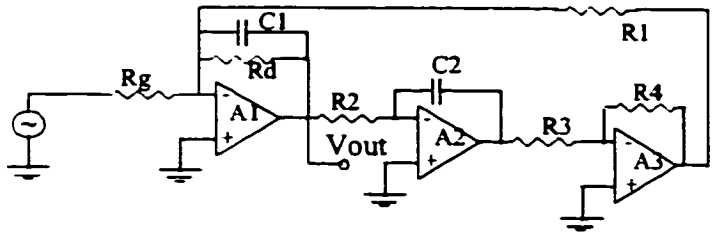

Figure 2.5: Second-order band-pass filter

Rg=200k, R1=10, R2=10k, R3=10k, R4=10k, Rd=200k, C1=1.59nf, C2=1.59nf

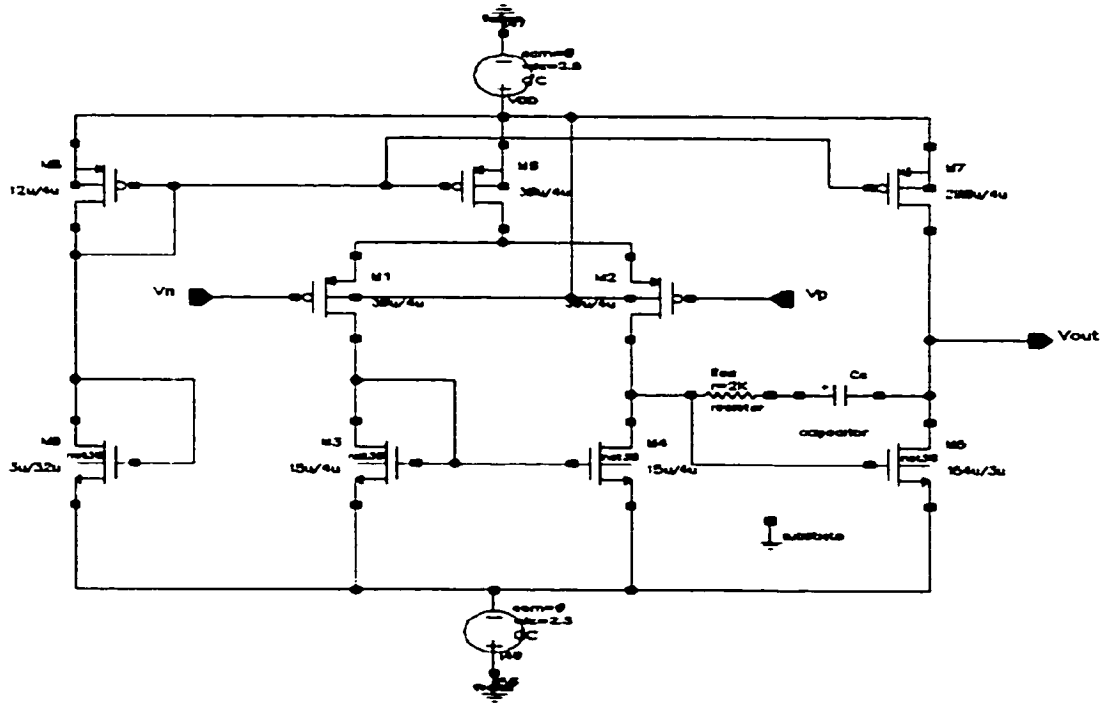

Figure 2.6: 3μμ m CMOS op-amp

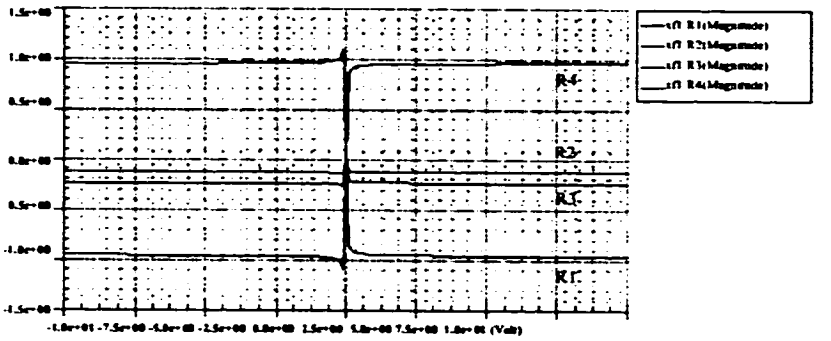

Figure 2.7: DC sensitivity of the output gain with respect to variations in R_1, R_2, R_3 and R_4

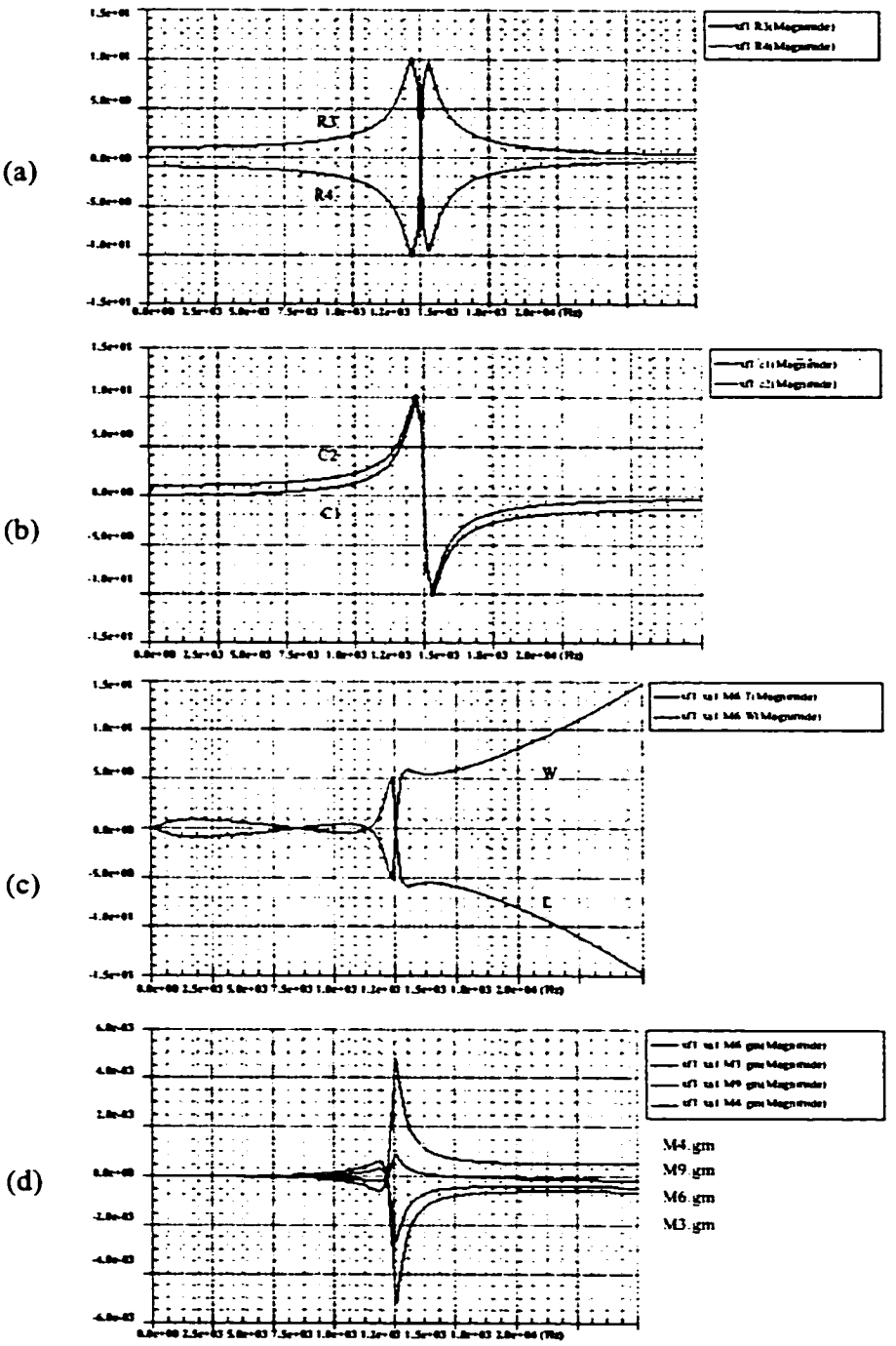

Figure 2.8:AC sensitivity of the output gain with respect to: a) R_3 and R_4; b) C_1 and C_2; c) W and L of transistor $M6$ of the op-amp $A1$; d) gain of transistors $M3$, $M4$, $M6$ and $M9$ of op-amp $A1$.

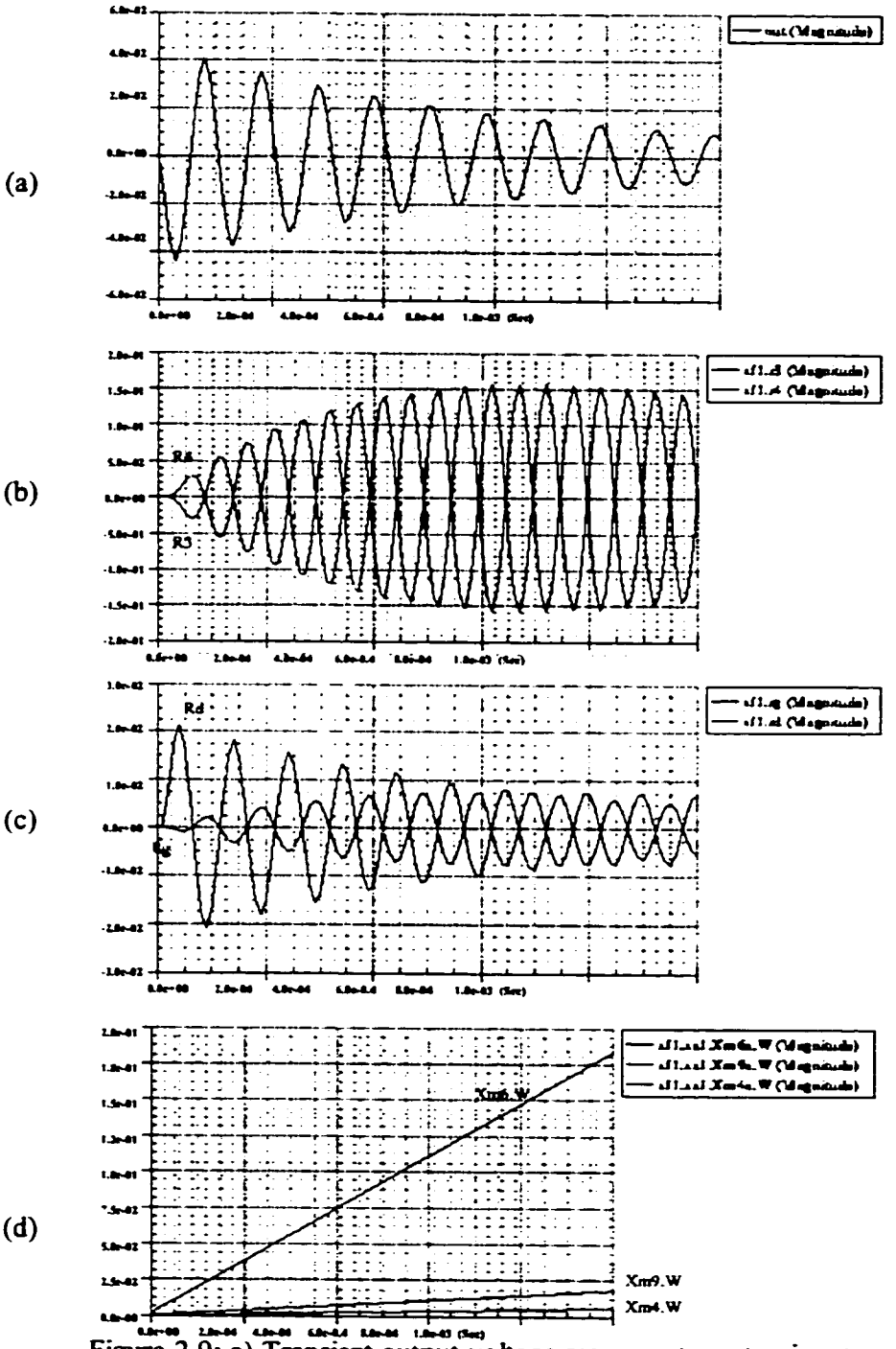

Figure 2.9: a) Transient output voltage response to a step input. Transient sensitivity curve with respect to: b) R_3 and R_4; c) R_d and R_g; d) W of transistors $M4$, $M6$ and $M9$ of op-amp $A1$.

These computed sensitivities exactly match the theoretical values obtained by Matlab [42] using the transfer function. Figure (2.7) shows the DC sensitivity of the output gain to the variations in resistances R_1, R_2, R_3 and R_4.

Figure (2.8a) and (2.8b) shows the AC sensitivity of the output gain to R_3 and R_4, and to C_1 and C_2 variation respectively. Figure (2.8c) shows the AC sensitivity of the output gain with respect to the width and length of the transistor $A1.M6$. The AC sensitivity of the output gain to the transistor transconductance variation (gm) are included in Figure (2.8d).

Computation results for transient sensitivity computation of different elements to an input step signal are presented in Figure (2.9): the output voltage response to a step input is given in Figure (2.9a). Figures (2.9b), (2.9c) and (2.9d) . present the transient sensitivity curves with respect to R_3 and R_4, R_d and R_g, and W of transistors $M4$, $M6$ and $M9$ of op-amp $A1$ respectively.

Table (2.2) gives the CPU time for each type of simulation on a SPARC10 workstation. The results are obtained for the DC, AC and transient sensitivity computation of the output voltage of the second-order band-pass filter of Figure 2.5 and 2.6 to all element variations.

Table 2.2: CPU time for band-pass filter

Type of simulation	CPU time (sec)	Number of SPICE steps
DC sensitivity	17.0	200 DC steps
AC sensitivity	18.0	100 frequencies
Transient sensitivity	10.4	100 time steps

2.4 Analog testing

In analog circuits, the concern is *which components to select for testing, the accuracy at which they should be tested and the test vectors that will test these components at the desired accuracy.*

For analog circuit testing, we propose hierarchical testing of all the elements that make up the circuit. For example, if we want to test the simple amplification circuit in Figure (2.10), we start by testing the op-amp. The op-amp gain is a function of the set of transistor characteristics (width, length), resistors, capacitances, etc. If we are sure that all the elements on which the op-amp gain depends are fault-free, then we are sure that the amplifier gain is also fault-free. Furthermore, if we are sure the op-amp, as well as resistances R_1 and R_2, are fault-free, then the amplification circuit is also fault-free. So, for analog circuit testing, we should test all the elements that make up the circuit hierarchically and make sure that all elements are within their tolerance values. With LIMSoft, the single-fault model case is considered. We suppose that only one element is

faulty and that the others are fault-free. An element is considered faulty if its deviation is out of the tolerance box (computed by LIMSoft) and fault-free if its deviation is within its specifications, which are taken from the circuit data sheets.

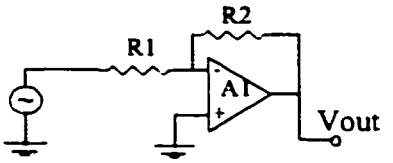

Figure 2.10: Amplification circuit

In the first step, LIMSoft automatically computes the circuit sensitivities as presented earlier, and then, from the output parameter specifications and the fault-free element's specifications, the faulty element's relative deviation is computed using equations (2.7), (2.8) and (2.9).

$$\left(\frac{\Delta x_t}{x_t}\right)_{out} = \frac{\left|\frac{\Delta out}{out}\right| + \sum_{j=1}^{t-1}\left|S_{x_j}^{out}\right|\left|\frac{\Delta x_j}{x_j}\right| + \sum_{j=t+1}^{M}\left|S_{x_j}^{out}\right|\left|\frac{\Delta x_j}{x_j}\right|}{S_{x_t}^{out}} \tag{2.7}$$

$$\left(\frac{\Delta x_t}{x_t}\right)_{out} = \frac{\left|\frac{\Delta out}{out}\right| - \sum_{j=1}^{t-1}\left|S_{x_j}^{out}\right|\left|\frac{\Delta x_j}{x_j}\right| - \sum_{j=t+1}^{M}\left|S_{x_j}^{out}\right|\left|\frac{\Delta x_j}{x_j}\right|}{S_{x_t}^{out}} \tag{2.8}$$

$$\left(\frac{\Delta x_t}{x_t}\right)_{out} = \frac{\frac{\Delta out}{out} + \sum_{j=1}^{t-1} S_{x_j}^{out}\frac{\Delta x_j}{x_j} + \sum_{j=t+1}^{M} S_{x_j}^{out}\frac{\Delta x_j}{x_j}}{S_{x_t}^{out}} \tag{2.9}$$

$\frac{\Delta out}{out}$ is the maximum tolerance of output parameter, $\frac{\Delta x_j}{x_j}$ the component tolerance (data sheet) and $S_{x_j}^{out}$ the adjoint network's computed sensitivity.

Equation (2.7) computes the element's relative deviations in the worst case, where the faulty element deviation is masked by the effect of the fault-free elements. In equation (2.8), the best case, the fault-free element's effect increases the detectability of the faulty elements. In the general case, equation (2.9), the sensitivity sign is conserved, thus the masking effect of the elements is considered. Figure (2.8a) shows two perfectly matched resistances, and, as long as they are correlated (deviate in the same direction), their deviations will have no effect on the output parameter. By contrast, in Figure (2.8b) there is no similar compensation effect for the capacitances c_1 and c_2.

In the second step, a set of test vectors is generated. A test vector is an input stimulus which maximizes the observability of the faulty component on the output parameter. Finding a test set for the elements of the circuit consists in finding a set of vectors which guarantees the observability of the faulty elements on at least one of the output parameters. The best test vector is the one for which the minimum element deviation causes the maximum output variation (equation 2.10). More details on this technique are presented in [6].

$$min\left(\left(\frac{\Delta x_i}{x_i}\right)_{out}\right) \tag{2.10}$$

The structure of the sensitivity computation and of the soft fault generation tools are given in Figure (2.11). The input to the program is a SPICE description file of a circuit and the output is a set of vectors which tests all circuit components. Depending on the level of the circuit description, the components can be described either at the behavioral level or at the transistor level.

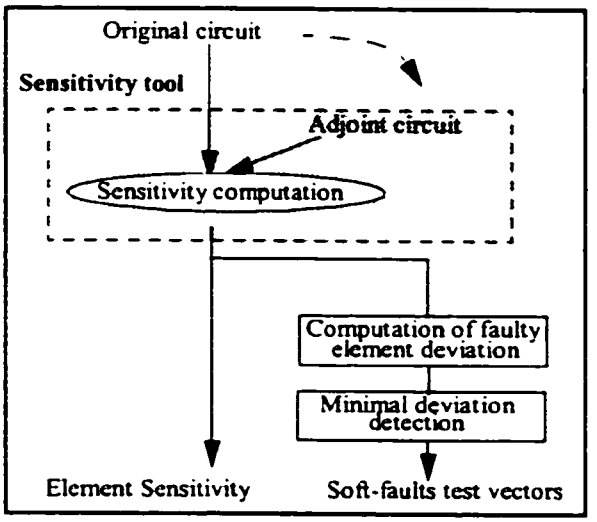

Figure 2.11: Sensitivity computation and soft-fault test vector generation

2.4.1 Experimental results

LIMSoft was used to compute the output voltage gain and voltage phase AC sensitivity and for test vector generation in the best-case, worst-case, and general-case deviations for the state-variable filter in Figure (2.12).

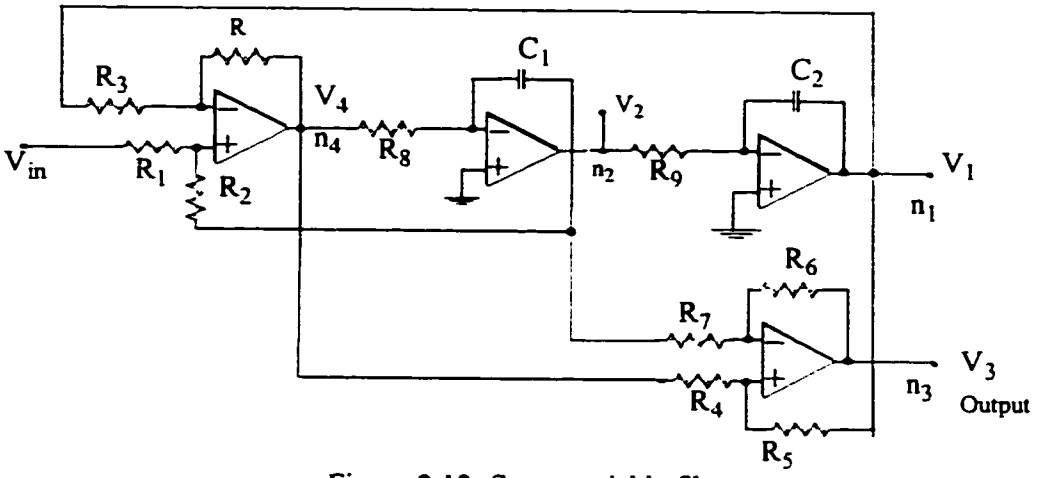

Figure 2.12: State-variable filter

R_1=10k, R_2=20k, R_3=10k, R_4=10k, R_5=100k, R_6=10k, R_7=30k, R_8=45.5k, R_9=2.2k, R=10k, C_1=1.6nf, C_2=1.6nf

First, the sensitivity of the output parameter's (V3) gain and phase with respect to all elements of the circuit are computed according to the procedure described in section 2.3. A sample of these results is presented in Figure (2.13). The sensitivity of |V3| with respect to R, R_3, R_4, R_5, C_1 and C_2 variation is presented.

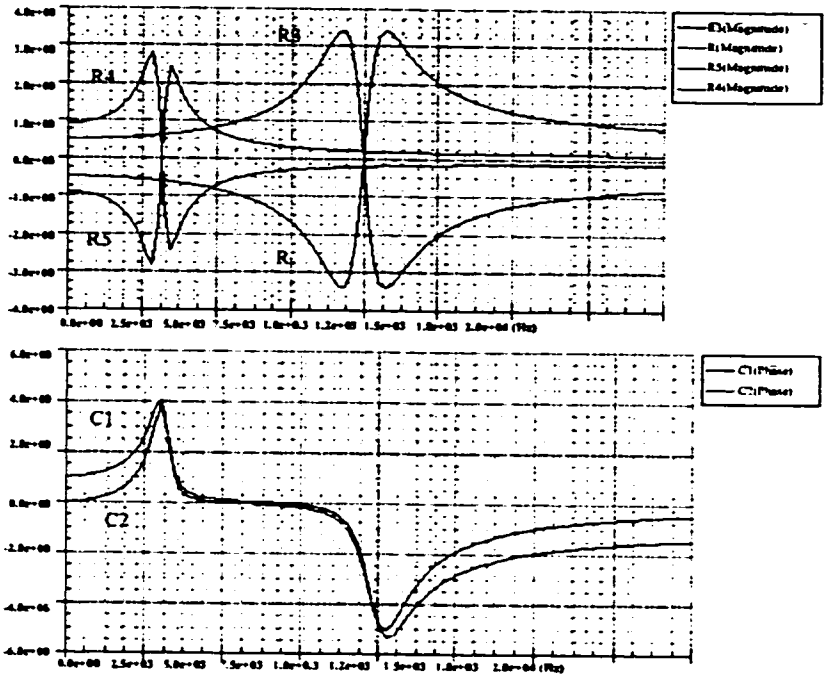

Figure 2.13: State-variable filter: AC sensitivity

For test vector generation, the minimum detectable deviation of all elements was computed for the entire frequency range. The minimum detectable deviation will never be greater than the worst-case deviation curves and no less than the best-case deviation curves. An interesting compromise which gives a realistic value is given by the general-case curves. Some results are presented in Figure (2.14).

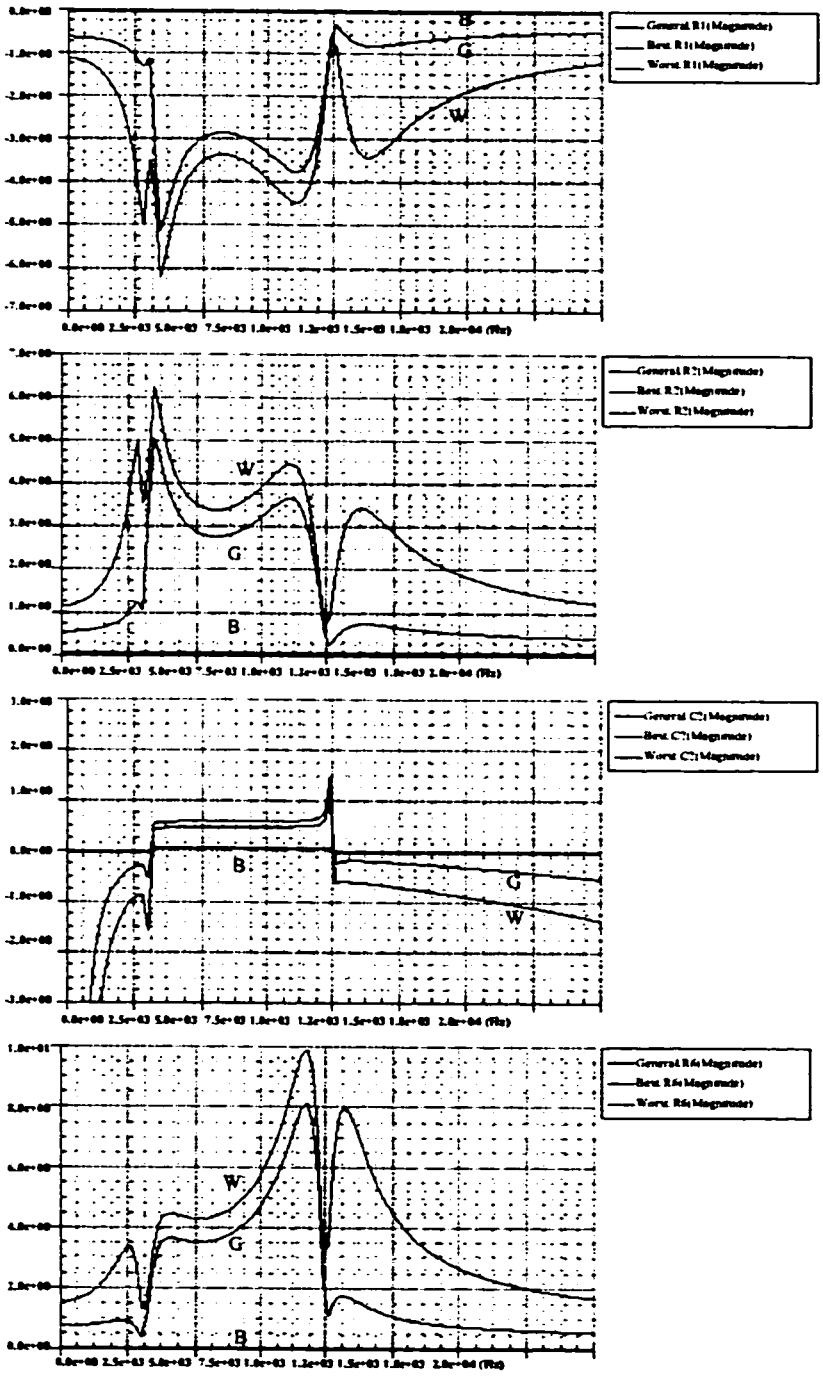

Figure 2.14: Minimum detectable deviation curves for the state-variable filter when the gain is measured in the worst-case (W), general-case (G) and best-case (B)

The best test frequencies are extracted by equation (2.10), and they represent the frequency for which the curves are at their minimum values. A summary of these minima and their corresponding frequencies is presented in Tables 2.3 and 2.4. It is also possible to minimize the frequency test set by choosing some test frequencies corresponding to the minima of several elements.

Table 2.3: Minimum deviations and corresponding frequencies for the state-variable filter in the best case and the worst case.

$\Delta x/x$ in Best Case			$\Delta x/x$ in Worst Case		
	deviation	frequency		deviation	frequency
C1	-4.99e-02	all frequencies	C1	4.53e-01	9.95e+03
C2	-4.99e-02	all frequencies	C2	5.37e-01	3.21e+03
R1	-4.99e-02	all frequencies	R1	-4.69e-01	9.95e+03
R2	4.99e-02	all frequencies	R2	4.69e-01	9.95e+03
R3	4.99e-02	all frequencies	R3	-4.66e-01	2.00e+04
R4	-4.99e-02	all frequencies	R4	3.88e-01	1.00e+00
R5	4.99e-02	all frequencies	R5	-3.88e-01	1.00e+00
R6	4.99e-02	all frequencies	R6	6.35e-01	3.21e+03
R7	-4.99e-02	all frequencies	R7	-6.35e-01	1.31e+03
R8	-4.99e-02	all frequencies	R8	4.53e-01	9.95e+03
R9	-4.99e-02	all frequencies	R9	5.37e-01	3.21e+03
R	-4.99e-02	all frequencies	R	4.66e-01	2.00e+04

Table 2.4: Minimum deviations and corresponding frequencies for the state-variable filter in the general case.

$\Delta x / x$ in General Case		
	deviation	frequency
C1	-2.20e-01	1.10e+04
C2	-1.97e-01	1.10e+04
R1	-3.17e-01	1.10e+04
R2	2.17e-01	1.10e+04
R3	-1.80e-01	1.10e+04
R4	1.01e-01	2.71e+03
R5	-2.01e-01	2.71e+03
R6	3.18e-01	3.11e+03
R7	-4.18e-01	3.11e+03
R8	-2.20e-01	1.10e+04
R9	-1.97e-01	1.10e+04
R	8.05e-02	1.10e+04

By analyzing these results, it is now possible to find the minimum set of test frequencies to be applied on the state-variable filter. In our case, in order to test all elements of the circuit, three frequencies are needed: 2.71KHz, 3.11KHz and 11.0KHz.

In this example, the deviations of the output amplitude were used to determine the test frequencies. It is easy to use the phase deviation, since LIMSoft provides phase sensitivity as well. It is also possible to combine results on the phase and the amplitude analysis to get the best test frequencies. This can also be useful for fault diagnosis in analog circuits.

2.5 Conclusion

An automatic tool for sensitivity computation and for analog circuit testing has been presented. The efficiency of the tool is the result of combining the adjoint network method with SPICE. The tool was tested for a large number of analog circuits and gave conclusive results.

CHAPITRE 3
GÉNÉRATION DE VECTEURS DE TEST POUR LES FAUTES PARAMÉTRIQUES

Dans le chapitre précédent, la sensibilité a été utilisée pour déterminer la déviation détectable sur les paramètres d'un composant en considérant la tolérance sur le paramètre de sortie et celle sur les composantes. Le stimulus, pour lequel la déviation détectable est maximale, était considéré comme le vecteur de test nominal pour le composant en question.

Dans ce chapitre, nous présentons une nouvelle méthode de test basée sur une approche probabiliste. Cette méthode prend en considération les variations paramétriques des composants par rapport à leurs valeurs nominales, ainsi que la corrélation entre ces variations pour générer une distribution nominale et une distribution fautive du paramètre de sortie. Le stimulus, pour lequel la probabilité de prendre une fausse décision sur l'état du circuit est minimale, est alors considéré comme le vecteur de test nominal pour le composant en question.

Le problème dans les circuits analogiques provient de la déviation des paramètres de leur valeur nominale (due à une fluctuation du procédé de fabrication), ce qui cause une

déviation des paramètres de sortie. Il est nécessaire de caractériser ces fluctuations et de générer des vecteurs de test qui ciblent ces fautes.

Différentes approches ont été utilisées pour caractériser ces fluctuations:

1. L'analyse du pire cas qui force les composants à la valeur limite. Le point négatif de cette méthode est qu'elle ne tient pas en considération la corrélation entre les paramètres, ce qui aboutit à un design sur-optimisé.
2. L'analyse appelée "corner analysis" où les six combinaisons possibles des transistors sont analysées. Cette méthode donne une meilleure prédiction du comportement du circuit, mais ne tient pas compte des fluctuations dans les valeurs des résistances et des capacités.
3. L'analyse Monte Carlo qui semble mieux adaptée pour prédire le comportement du circuit. Mais elle est très coûteuse en temps de simulation et devient donc non pratique pour des gros circuits où une simulation Monte Carlo peut prendre des heures, voire des jours.

En prenant l'hypothèse que l'on puisse caractériser les fluctuations des composantes dans un temps réaliste, il reste toujours le problème de génération de vecteurs de test. Pour résoudre ce problème, il faut répondre à quatre critères:

1. Définir la faute analogique,
2. Définir un modèle pour la faute analogique,
3. Générer les vecteurs de test pour ces fautes,
4. Définir la couverture de fautes obtenue.

Pour résoudre le problème de fluctuations des composants et de la génération des vecteurs de test, notre approche consiste à remplacer le simulateur par une analyse statistique du circuit, basée sur le développement de la série de Taylor et sur la sensibilité (présentée dans le chapitre précédent). Cette approche permet de considérer toutes les données statistiques du processus de fabrication pour la génération de la distribution statistique du circuit nominal (caractérisation des fluctuations du circuit). De plus, dans le but de générer des vecteurs de test, il n'est pas nécessaire d'injecter la faute dans le circuit nominal et de simuler à nouveau; il suffit d'injecter la faute dans l'équation et de réévaluer cette dernière.

Une comparaison a été réalisée entre cette méthode et l'outil Hspice, et elle prouve que les résultats corrèlent à une erreur près. Cette perte de précision est plus que compensée par le gain de vitesse obtenue par rapport au simulateur.

Cette méthodologie est présentée en détail dans l'article qui suit. L'article a fait l'objet d'une soumission à "IEEE Transactions on CAD" en novembre 1999.

Parametric Fault Simulation and Test Vector Generation

Khaled Saab, Naim B. Hamida, Bozena Kaminska

Fluence Technology Inc., 8700 SW Creekside Place, Beaverton-OR, 97008 USA.

Abstract

Process variation has forever been the major fail cause of analog circuit where small deviations in component values cause large deviations in the measured output parameters. This paper presents a new approach for parametric fault simulation and test vector generation. The proposed approach utilizes the process information and the sensitivity of the circuit *principal components in order to generate statistical models of the fault-free and the faulty circuit. The obtained information is then used as a measurement to quantify the testability of a circuit.*

This approach extended by hard fault testing has been implemented as automated tool set for IC testing.

3.1 Introduction

Analog MOS-integrated circuit (IC) design has received great attention in the last few years, mainly due to the trend toward mixed analog-digital chips, to low voltage operation and even to artificial neural network. However, due to the technological tolerances model parameters are subject to randomness. As a consequence, the circuit behavior differs for different runs, different slices of the same run, and for different dice of the same silicon slice [45], [47] and [48].

Process variation has forever been the major fail cause of analog circuit. But with the advancement in the fabrication process and the reduction of transistor sizes, process variation is increasing the fail rate of digital circuits as well. These variations are hard to detect since they do not cause failure at all conditions and are mostly affecting the long-term reliability of a circuit.

Detecting these variations and generating test vectors for them, will be the subject of this paper where a general methodology for determining the statistical behavior of a nominal circuit and a faulty circuit is used.

3.2 Overview

In the early stages, simulations of the integrated circuits were accomplished only with nominal parametric conditions. Soon after, variations in the fabrication process were found to be the primary cause of parametric yield loss.

Number of approaches for the characterization of the fluctuation in an IC process were used including the extraction of the worst-worst case conditions of the process. This

method requires that each parameter is set to a maximum or minimum (whichever is more degrading the performance) allowable value without regard to the particular parameter correlation. In addition to being quite tedious and time consuming, the analysis resulted in overly designed products.

Other methods took advantage of the devices correlation where the circuit "slow" and "fast" corners of the MOS transistors are simulated. This method gives a much more realistic description and requires only six corners analysis.

However, analog and mixed circuit analyses require considering parametric variations on resistance, capacitance, and even transmission lines. This new requirement increases the number of corners (since we need to consider all combinations of those elements to find the worst case) and thus more Monte Carlo simulations are needed to capture the circuit behavior.

In order to reduce the required number of Monte Carlo simulations, some suggested evaluating circuit sensitivity and using the obtained information as guide lines to deterministically identifying the circuit corners. Once the circuit corners have been identified, the appropriate reduced number of Monte Carlo simulations for corner analysis can be executed (see [10] and [43]).

But still, what is an analog fault? And what is an analog fault model? And what is the fault coverage?

In an attempt to answer those questions, structural testing has been introduced to the analog domain. In [2] and [7] it has been suggested that the circuit could be tested by

testing all of its device parameters. If all device parameters are within their tolerance ranges the circuit is classified as fault-free. As a consequence, the analog fault was defined as a deviation of the device parameter outside of its predefined range of operation.

In [26], the authors proposed a realistic defect oriented testability methodology that creates the signature of a faulty free circuit in a multi-dimensional space. If the faulty circuit signature exhibits a response outside this space at least for one of the test stimuli, then it is recognized as faulty. However, the concepts of pass and fail in analog circuit is not clear-cut, thus no absolute signature could be generated for comparison.

In [51] a fault simulator for linear analog circuits has been presented where abstracting the analog circuit at the behavioral level, and then transforming it from the continuous Laplace domain into the discrete Z-domain achieves fault simulation.

One of the problems that kept this solution at bay is that for each fault there is a need to perform fault injection followed by a full fault simulation. In the case of M faults and K iterations per fault (K is the number of Monte Carlo simulations), if a standard Monte Carlo approach is used, the total number of Monte Carlo simulations needed is $O(K.M)$. This is not realistic for large circuits with large number of faults.

To correct this problem and make the structural testing affordable, in [7] a first approach for test vectors generation based on structural testing and on the device parameter sensitivities has been proposed. Indeed, sensitivity has been computed for each device in the circuit. The stimulus with highest sensitivity has been taken as the test vector.

In [9] and [10] a similar approach was proposed, but instead of relying only on the individual devices sensitivities information, device variations and output tolerances where taken into consideration as well.

In this paper we present an efficient and realistic approach for parametric faults simulation and test vector generation. Indeed, the process information and the sensitivity to the circuit principal components are used in order to generate statistical models of the circuit: one for the fault-free circuit and one for each fault in the fault list by linear approximation. The statistical models of the fault free circuit and the faulty circuits are used to estimate the probability of accepting a faulty circuit and the probability of rejecting a good circuit. The obtained information is then used as a measurement to quantify the circuit testability.

3.3 Proposed approach

Due to the complexity of analog signal, efficient analog and mixed signal circuit testing could only be approached by statistical analysis of a circuit. From the device parameter distribution data and the sensitivity computation results it is possible to compute not only the fault free output parameter distribution but also the faulty output parameter distribution. The proposed approach is divided into eight steps (Figure 3.1):

1. Compute output parameter sensitivity with respect to all device model parameters in a circuit.
2. Obtain the device statistical data for each device model parameter in a circuit.

3. Compute the circuit fault free output parameter distribution when all device model parameters vary within the tolerance margin defined by the device statistical data. Mapping the device variation to performance variation performs this step.
4. Construct the fault list. Fault list consists of all or a subset of the device components and model parameters.

For each fault in the fault list:

5. Perform fault injection by replacing the fault-free device model parameters by its faulty model.
6. Compute the circuit faulty output parameter distribution when all device parameters (except the faulty device parameters) vary within the tolerance margin defined by the device statistical data. This step performs the mapping of the device variation and of the fault effect to performance variation.
7. From the fault free and faulty circuit distributions quantify the degree of testability of the fault. In this step false accept and false reject concept is used as a measure of the circuit testability.
8. Based on the above information, for each fault, the test vector is obtained by choosing the stimulus that minimizes the false accept and/or the false reject.

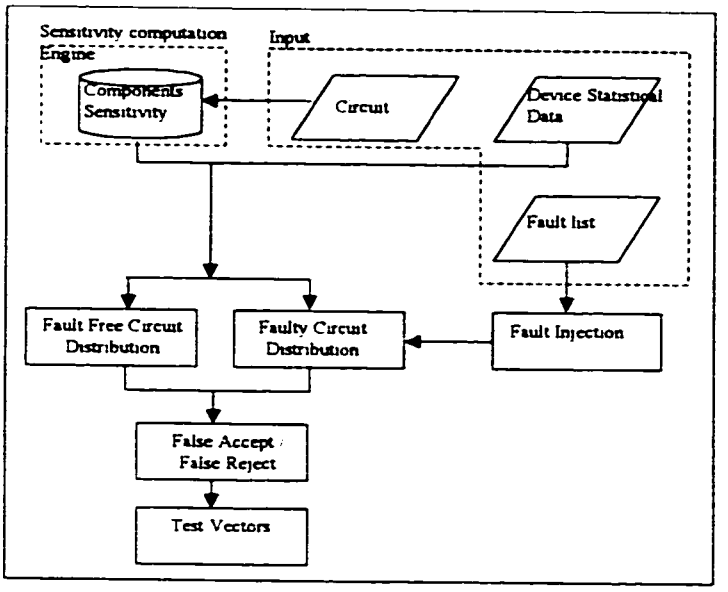

Figure 3.1: Soft Fault Testing Flow Chart

This paper is organized as follow: first, the device statistical data will be described. Then in paragraphs 3.4, 3.5 and 3.6, we will present how a device statistical data is used to predict the output parameter variation. Indeed, the devices statistical data and the fault list are used to predict the fault-free and the faulty circuit distribution respectively. Finally, from the fault-free and the faulty output distributions, the false accept and false reject are derived as a measuring criteria in order to obtain the circuit test vectors. Practical examples and results are included in the last two sections.

3.4 *Fault free circuit*

Because of the uncontrollable variations in device model parameter values inherent to the manufacturing process, the circuit output usually deviates from the nominal

response. As long as these deviations are within their specifications, the circuit is declared as fault-free for all functional testing purposes. For structural testing, this definition is extended to the circuit devices where a fault-free circuit is not only defined by its output specifications but by its device specifications as well. The device fault-free specifications are defined to guaranty a fault-free behavior of the circuit under all operation conditions (process variation, temperature variation, and radiation...).

Starting from the nominal circuit and using the device statistical data and a standard Monte Carlo approach, circuit output mean and standard deviations can be simulated. However, computational requirements could easily become prohibitively expensive as the circuit size grows and the number of the variable parameters increases.

Our objective is to find a realistic, accurate, and efficient solution that allows obtaining the statistical information on the fault-free circuit outputs.

3.4.1 Fault free circuit distribution

We are interested in the general case where we want to describe the effect of a large number of random devices on a given output (Figure 3.2) using a piece wise linear estimation of the fault-free circuit output. Let *out* be the output parameter of interest and $x_i\ (i = 1...N)$ N devices in the circuit such that *out* is a function of x_i. The Taylor series generated by $f(x_0, x_1 \cdots, x_N)$ up to the first order is computed:

Equation 3.1

$$\begin{aligned} out &= f(x_0, x_1 \cdots, x_N) \\ &= out_0 + \frac{df(x_0, x_1 \cdots, x_N)}{dx_0}\Delta x_0 + \frac{df(x_0, x_1 \cdots, x_N)}{dx_1}\Delta x_1 + \cdots + \frac{df(x_0, x_1 \cdots, x_N)}{dx_N}\Delta x_N \\ &= out_0 + S_{x_0}^{out}\Delta x_0 + S_{x_1}^{out}\Delta x_1 + \cdots + S_{x_N}^{out}\Delta x_N \end{aligned}$$

Or in a more compact form we write:

Equation 3.2 $$out = out_0 + \sum_{i=1}^{N} S_{x_i}^{out}\Delta x_i$$

Where out is the estimated value of the output parameter due to device variation, out_0 is the nominal output when all device parameters are at their nominal value, Δx_i the variation of the circuit device parameter x_i, and N is the number of variable parameters in a circuit.

From Equation 3.2, the desired output distribution function of a fault-free circuit is obtained by

Equation 3.3
$$\begin{aligned} \mu_{out} &= \mu_{out_0} + \sum_{i=1}^{N} S_{x_i}^{out}\mu_{\Delta x_i} \\ &= \mu_{out_0} \end{aligned}$$

Equation 3.4 $$\sigma_{out} = \sum_{i=1}^{N}(S_{x_i}^{out})^2\sigma^2_{x_i} + \sum_{i=1}^{N}\sum_{j=1, i\neq j}^{N} S_{x_i}^{out}S_{x_j}^{out}\sigma_{x_i x_j}$$

Where σ_{x_i} is the standard deviation of the device parameter x_i, and $\sigma_{x_i x_j}$ is the covariance term of $[x_i, x_j]$.

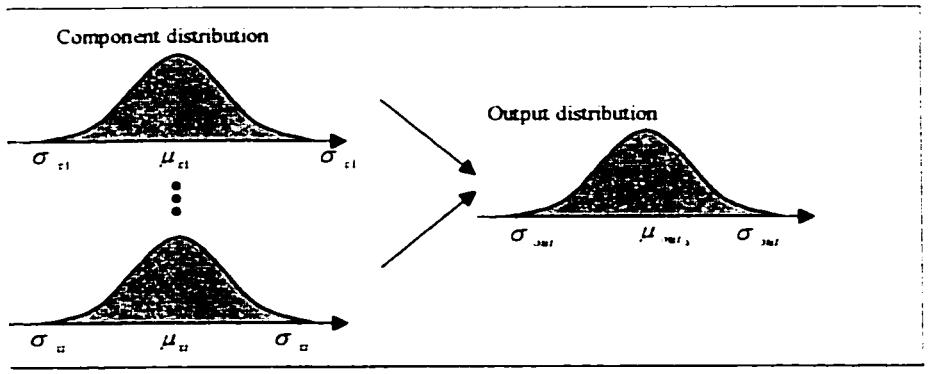

Figure 3.2: Fault-free circuit distribution

3.5 Fault list

The fault list is composed of all or a subset of the circuit components and model parameters. Typically, resistance values, capacitance values, the MOS geometric parameters (width, length, and oxide thickness), and the MOS electrical parameter (flat band voltage) are considered. Additional parameters that could affect the circuit behavior (temperature for example) could be considered as well.

3.6 Faulty circuit

Two typical steps are used in order to characterize a faulty circuit. For each fault in the fault list perform:

- Fault injection
- Fast fault simulation

The difference with respect to a standard fault simulation approach is that in our case fault injection consists on replacing the fault-free device parameter statistical data by its

faulty model. Fault simulation consists on computing the fault output PDF function using a statistical approach and not the simulator.

In the next section, fault injection process will be described followed by the computation of the faulty circuit output distribution function.

3.6.1 Fault injection

First we need to define the fault model. In our case, two fault models could be used:

- The standard single fault model
- The group fault model.

For the single fault model case only one faulty component is considered during fault simulation. Let $X_p \sim N(\mu_{X_p}, \sigma^2 x_p)$ be the target fault. As long as X_p value is inside the tolerance range $\mu_{X_p} - 3\sigma_{X_p} < X_p < \mu_{X_p} + 3\sigma_{X_p}$ the device parameter is considered as fault-free and thus the circuit is considered as fault free. Setting X_p value to be outside the tolerance range performs fault injection. In other term, changing the X_p distribution from $X_p \sim N(\mu_{X_p}, \sigma^2 x_p)$ to $X_p \sim N(\mu_{X_p} \pm 3\sigma_{X_p}, 0)$ performs a single fault injection (see Figure 3.3).

However, in most cases the single fault model is inadequate to represent the silicon faults. Indeed, some of the process parameter variations (e.g. threshold voltage of a transistor caused by doping variation) are not local to a single transistor but are global, affecting multiple transistors. In this case, the "faulty group" could be obtained by grouping the affected components by their position on the die or by their component

type (e.g. NMOS vs. PMOS). In a similar manner as in the single fault case, in the group fault model case simultaneously shifting all affected components out of their tolerance range performs fault injection.

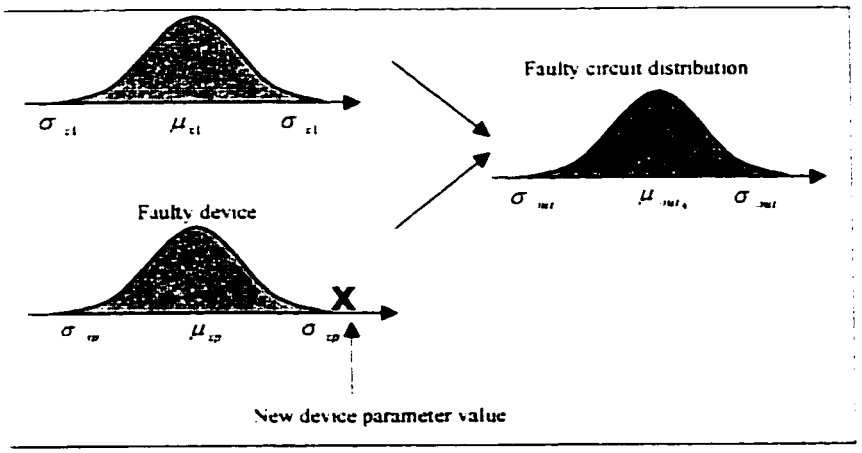

Figure 3.3: Faulty circuit distribution

3.6.2 Fault simulation and faulty circuit distribution

Similarly to the fault-free circuit case, instead of using one Monte Carlo simulation, we use a statistical approach and linear estimation of the fault effect to obtain the faulty circuit PDF.

For each fault in the fault list, replacing the fault-free device statistical data by its faulty value (as discussed in the previous paragraph) performs fault injection.

Under this assumption Equation 3.3 and Equation 3.4 become

$$\mu_{out}\big|_{X_p \text{ is faulty}} = \mu_{out_0} + \sum_{i=1}^{N} S_{x_i}^{out} \mu_{\Delta x_i}$$

Equation 3.5

$$= \mu_{out_0} + \sum_{i=1, i \in p}^{N} S_{x_i}^{out} \mu_{\Delta x_i} + \sum_{i=1, i \in p}^{N} S_{x_i}^{out} (\mu_{\Delta x_i} + 3\sigma_{x_i})$$

$$= \mu_{out_0} + \sum_{i=1, i \in p}^{N} S_{x_i}^{out} (3\sigma_{x_i})$$

And

Equation 3.6

$$\sigma^2_{out}\big|_{X_p \text{ is faulty}} = \sum_{i=1, i \in p}^{N} (S_{x_i}^{out})^2 \sigma^2_{x_i} + \sum_{i=1, i \in p}^{N} \sum_{j=1, j \neq i \in p}^{N} S_{x_i}^{out} S_{x_j}^{out} \sigma_{x_i x_j}$$

Where $\mu_{out}\big|_{X_p \text{ is faulty}}$ and $\sigma^2_{out}\big|_{X_p \text{ is faulty}}$ represent the mean and the standard deviation of the faulty circuit respectively. "p" is the set of faulty elements under test: this set contains either one component in the single fault model case or all components of the "faulty group" in the group fault model.

Note that in this paper only the single fault model is implemented. The extension to "group fault" model is strait forward.

3.7 Measurement of circuit testability

Due to the continuous nature of analog circuit, the distinction between the fault-free circuit and the faulty circuit is not as clear as in the digital case where stuck-at fault model effect is seen as a 0/1 effect on the output. Figure 3.4 shows a typical digital case where there could be no error in distinguishing between the fault-free and the faulty circuit. On the other hand, Figure 3.5 shows a typical analog case where there is no clear distinction between the fault-free and the faulty circuit. From Figure 3.5 it is clear that if

any decision is made regarding the circuit, this decision is subject to error. There is a need for a way to be able to measure the error that is committed and thus to measure the degree of testability of a circuit. Once the "degree of testability" is obtained, circuit classification could be done.

Let set up the two hypothesis:

$\mathcal{H}_0$: the circuit is faulty

$\mathcal{H}_1$ the circuit is fault-free

The decision to accept or reject any hypothesis is subject to error. Two kinds of errors are possible. If the circuit is accepted as fault-free circuit when it is faulty then a *type I error* occurred. If the circuit is rejected as faulty circuit when it is fault-free a *type II error* is present [49]. The situation is described in Table 3.1.

Table 3.1: Decision in Hypothesis Testing

Decision \ state	$\mathcal{H}_0$	$\mathcal{H}_1$
$\mathcal{H}_0$	No error	Type II error
$\mathcal{H}_1$	Type I error	No error

The probability of occurrence of type I and type II errors are thus defined as:

$$P(\text{type I error}) = P(\mathcal{H}_1|\mathcal{H}_0)$$

$$P(\text{type II error}) = P(\mathcal{H}_0|\mathcal{H}_1)$$

The notation $P(\mathcal{H}_i|\mathcal{H}_j)$ indicates the probability of deciding $\mathcal{H}_i$ when $\mathcal{H}_j$ is true. In the literature $P(\mathcal{H}_1|\mathcal{H}_0)$ is commonly referred to as the probability of false accept (PFA) and $P(\mathcal{H}_0|\mathcal{H}_1)$ is referred to as the probability of false reject (PFR). These errors are

illustrated in Figure 3.5. These two errors are unavoidable to some extent and *it is not possible to* reduce *both error probabilities simultaneously.*

However, those errors may be traded off against each other. To do so we need only to change the threshold as shown in Figure 3.6. Clearly, the type I error probability is decreased at the expense of increasing the type II error probability. Trading one probability against the other, change the circuit's quantification mechanism and could have serious repercussion: if we decide the device parameter is fault-free but it proves to be defective, the entire circuit could be defective and we incur a large cost (packaging, additional testing...). If however, we decide the device parameter is defective when it is not, we incur the smaller cost of the circuit only. Thus, care should be considered when selecting a new threshold depending on the circuit at hand and the type of test that the user is interest in.

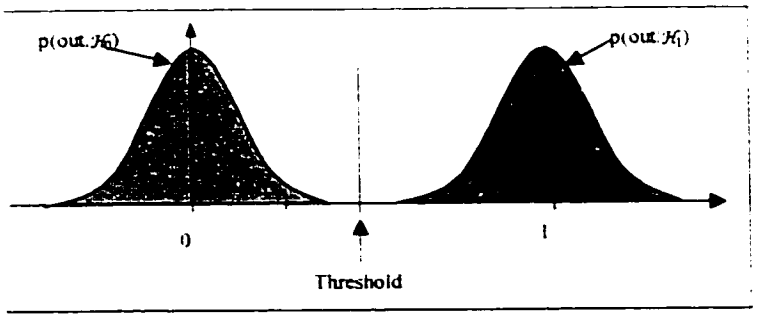

Figure 3.4: Digital circuit fault-free and faulty probability distribution function

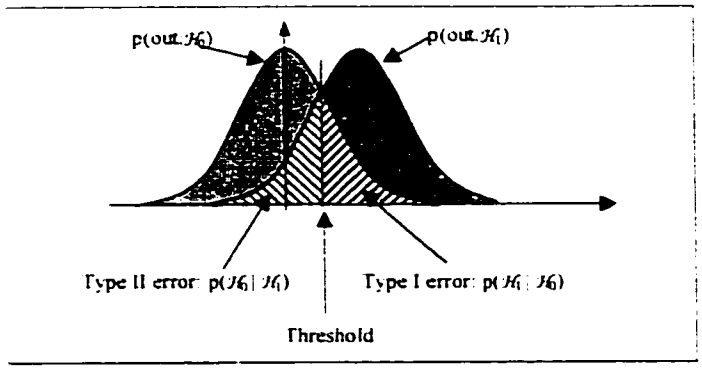

Figure 3.5: Possible hypothesis testing errors and their probabilities.

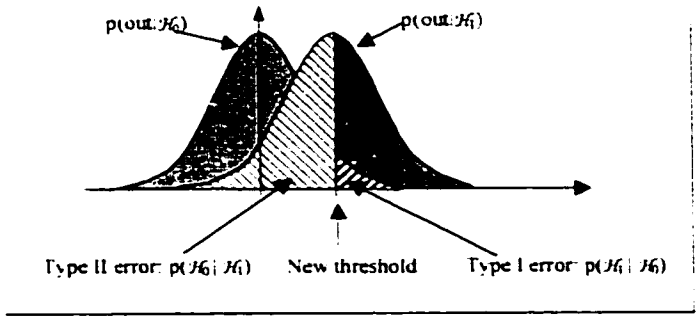

Figure 3.6: Trading off errors by adjusting threshold

Probability of false accept and probability of false reject are the natural way to represent the error in each one of the two hypothesis. False accept and false reject will be used as criteria for analog circuit test vector generation.

3.8 Test vector generation

Now all the building blocks are introduced to allow us to generate analog test vectors. First the fault-free and the faulty circuit PDF's functions are obtained by using the mapping of the device statistical data and the fault effect to the output parameter. Then.

the obtained probability of false rejects and false accept are used as a method to quantify the classification error. Those building blocks are used to identify the best stimulus that minimizes the probability of taking the wrong decision on the circuit. In other words we want to identify the stimuli that minimizes the "*Bayes risk*" defined as

Equation 3.7 $$R = \sum_{i=0}^{1}\sum_{j=0}^{1} C_{ij} P(H_i / H_j) P(H_j)$$

Where C_{ij} is the cost if we decide $\mathcal{H}_i$ but $\mathcal{H}_j$ is true. $P(\mathcal{H}_i | \mathcal{H}_j)$ the probability of deciding $\mathcal{H}_i$ when $\mathcal{H}_j$ is true and $P(\mathcal{H}_j)$ the probability of occurrence of the fault. Usually if no error is made, we do not assign a cost so that $C_{00} = C_{11} = 0$. In addition since each fault is tested independently of the other faults and the probability of the fault being present in the circuit is independent of the applied stimulus, then, the probability of occurrence of the fault $P(\mathcal{H}_0)$ could be set to a constant, namely 1. Under those assumptions Equation 3.7 is reduced to

Equation 3.8 $$R = C_{10} P(H_1 / H_0) P(H_0) + C_{01} P(H_0 / H_1) P(H_1)$$

The Bayes risk R is computed for all stimuli and for each fault in the fault list. The stimulus for which R is minimal, is taken as the test vector for the fault under consideration

Equation 3.9 $$\text{Test Vector} = \text{Stimuli} \mid R(\text{stimuli}) \text{ is minimal}$$

For example, let's consider a simple case where the possible stimulus is a predefined frequency range [f1, f2]. For each frequency in [f1, f2] the fault free circuit and the

faulty circuit PDF functions are computed for each device parameters in the fault list. The test vector is the stimuli for which the two distributions are further apart (Figure 3.7).

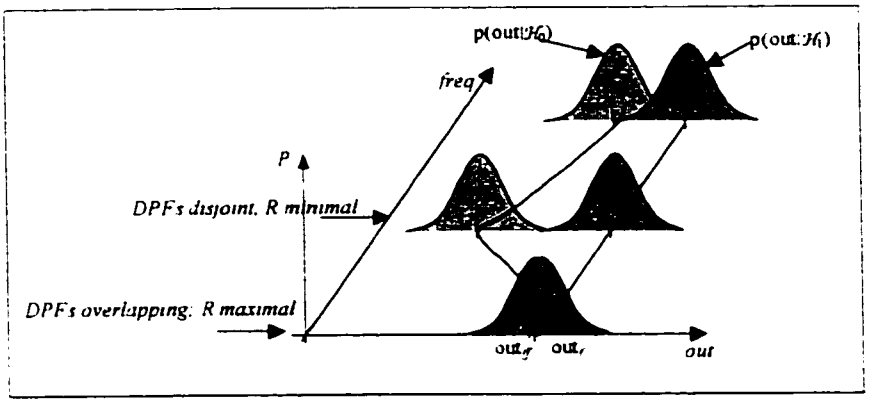

Figure 3.7: Fault-free circuit PDF and faulty circuit PDF as function of the frequency

3.9 Implementation

The algorithm presented in paragraph 3 and the theory behind it has been implemented on Unix workstation in "C" language. Experimental results and some examples are presented next.

3.10 Experimental results

3.10.1 Detailed Experimental results

The proposed method has been applied on a closed loop unity gain operational amplifier of Figure 3.8. In this test case, the stimulus space is a DC sweep [-10v 10v] of the input voltage with a step of 0.1v.

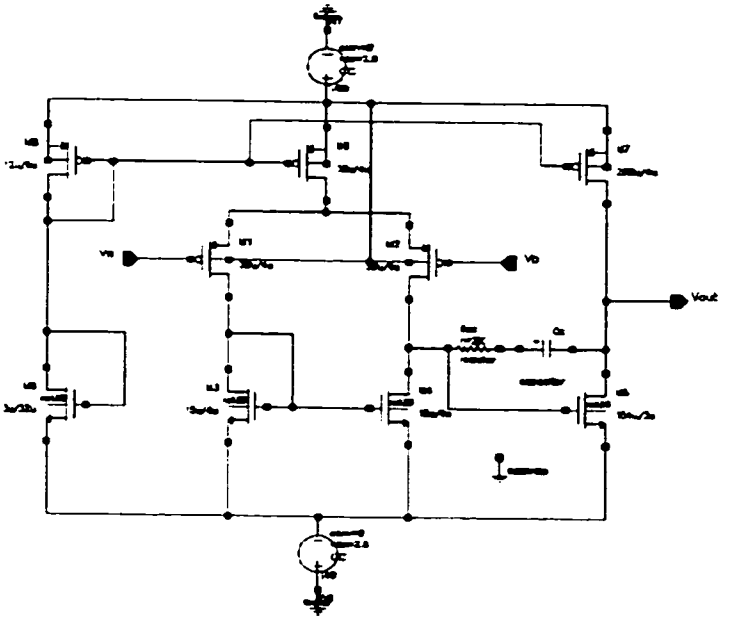

Figure 3.8: Open loop operational amplifier

3.10.1.1 Device Statistical Data

The statistical data variation file has been defined by:

- For all resistances
  - Deviation(resistance)=Gaussian(die-to-die variation=20%, mismatch variation=1%)
- For all MOS Transistors
  - Deviation(MOS width)=Gaussian(die-to-die variation=0.1u, mismatch variation=0.01u)
  - Deviation(MOS length)=Gaussian(die-to-die variation=0.1u, mismatch variation=0.01u)

3.10.1.2 Fault-free distribution

From the device statistical data and by using Equation 3.3 and Equation 3.4 the fault-free output mean and variance for each stimuli are computed. The same device statistical data where submitted to Hspice and the fault-free circuit mean and variance where computed using 33 Hspice Monte Carlo runs. The obtained Hspice results agreed with Equation 3.3 and Equation 3.4 results with less than 5% error on the fault-free output mean (Figure 3.9) and less that 0.5% error on the fault-free output standard deviation (Figure 3.10).

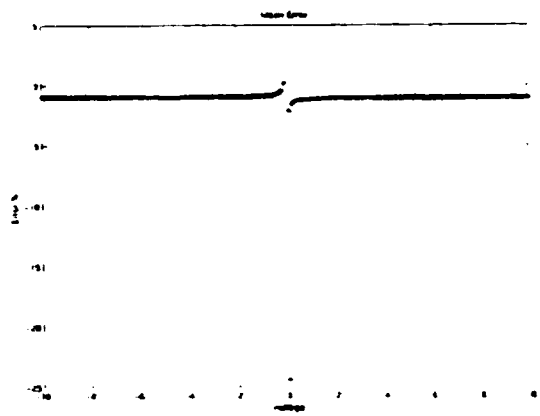

Figure 3.9: Error (%) on The Fault-Free Circuit Mean

plot(xaxis,(mean_FTmaxx-mean_Hspice')./mean_Hspice','*')

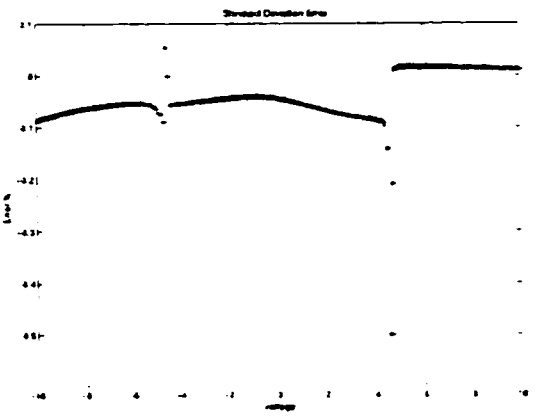

Figure 3.10: Error (%) on The Fault-Free Circuit Standard Deviation

plot(xaxis,(std_FTmaxx-std_Hspice')./std_Hspice','*')

3.10.1.3 Faulty circuit distribution

In this step, using Equation 3.5 and Equation 3.6, for each stimuli and each fault in the fault-list (Table 3.2), fault injection and fault simulation where performed in order to obtain the mean and the variance value of the faulty circuits. For comparison, faults were manually injected into the circuit and simulated with Hspice. Then, for each fault in the circuit, the error between the Hspice results and our approach has been computed.

show the error on the output mean value ($\mu_{out}|_{X_p \text{ is faulty}}$).

Figure 3.12 show the error on the output standard deviation ($\sigma_{out}|_{X_p \text{ is faulty}}$).

Figure 3.13 show the error on the output envelope ($\mu_{out}|_{X_p \text{ is faulty}} + 3\sigma_{out}|_{X_p \text{ is faulty}}$).

Table 3.2: soft fault dictionary

Fault name	Nominal Value	Inter-die Fault value	Intra-die Fault value
r1.value	10k	20%	1%
r2.value	10k	20%	1%
xop1.c76.value	1pF	0%	0%
xop1.m1.l	4u	0.1u	0.01u
xop1.m1.w	12u	0.1u	0.01u
xop1.m2.l	32u	0.1u	0.01u
xop1.m2.w	3u	0.1u	0.01u
xop1.m3.l	4u	0.1u	0.01u
xop1.m3.w	30u	0.1u	0.01u
xop1.m4.l	4u	0.1u	0.01u
xop1.m4.w	15u	0.1u	0.01u
xop1.m5.l	4u	0.1u	0.01u
xop1.m5.w	30u	0.1u	0.01u
xop1.m6.l	4u	0.1u	0.01u
xop1.m6.w	30u	0.1u	0.01u
xop1.m7.l	4u	0.1u	0.01u
xop1.m7.w	15u	0.1u	0.01u
xop1.m8.l	4u	0.1u	0.01u
xop1.m8.w	200u	0.1u	0.01u
xop1.m9.l	3u	0.1u	0.01u
Xop1.m9.w	154.2u	0.1u	0.01u
Xop1.r77.value	2k	0.1u	0.01u

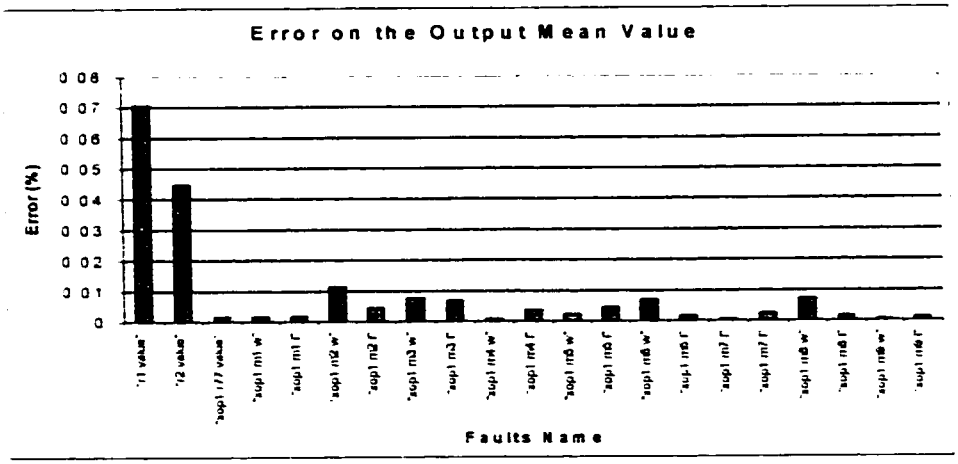

Figure 3.11: Error on the Faulty Output Mean Value.

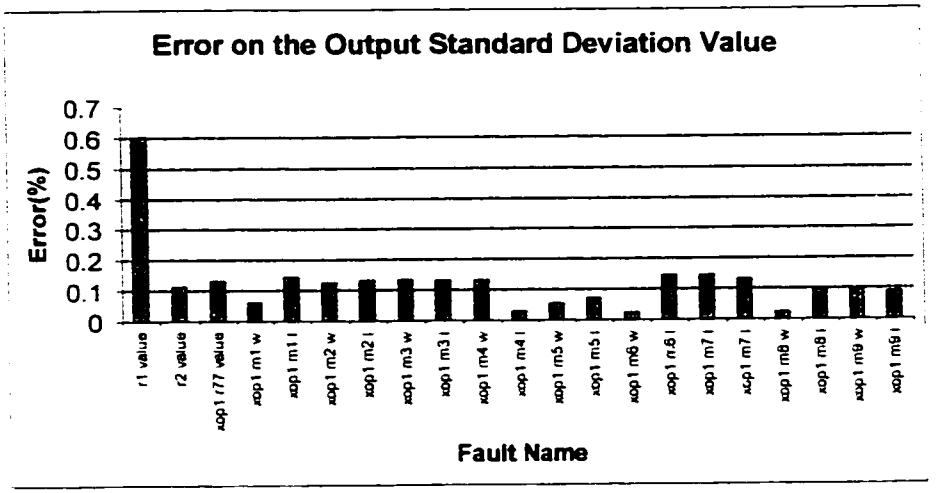

Figure 3.12: Error on the Faulty Output Standard Deviation Value

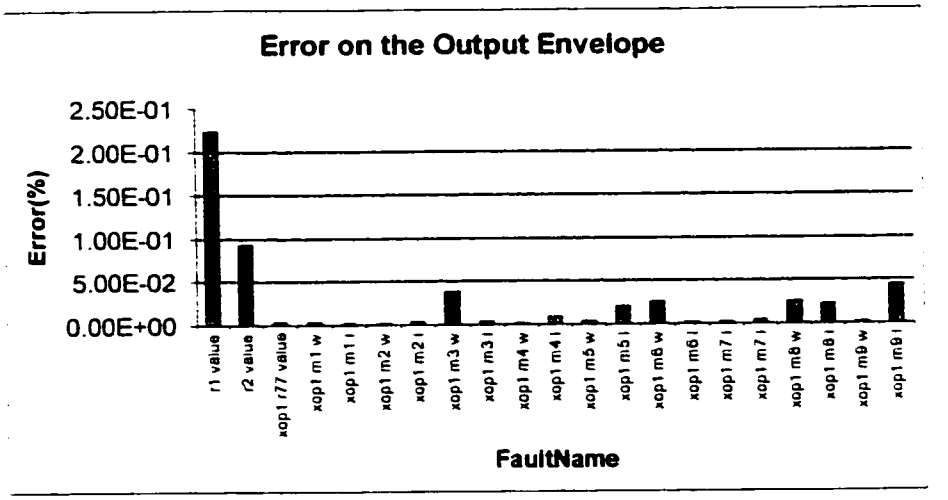

Figure 3.13: Error on the Faulty Output Envelope Value

From Figure 3.11, the average error on the faulty output mean value is 2%. From Figure 3.12, the average error on the faulty output standard deviation value is 14%. And from Figure 3.13, the average error on the faulty output envelope is 1.3%.

In general the Hspice results and the computed results agree closely. Note, that even some anomaly could be seen (fault number 1 and 2) which are due to the first order approximations, the total computation time (1.3 second) was a small fraction of the Hspice simulation time (26.4 second).

3.10.1.4 False Accept and False Reject Computation

In this step, for each stimuli and each fault, the probability of false accept and the probability of false reject where obtained. Figure 3.14 shows the false accept results for some of the circuit components as function of the input stimuli. Similar graphs where obtained for false reject.

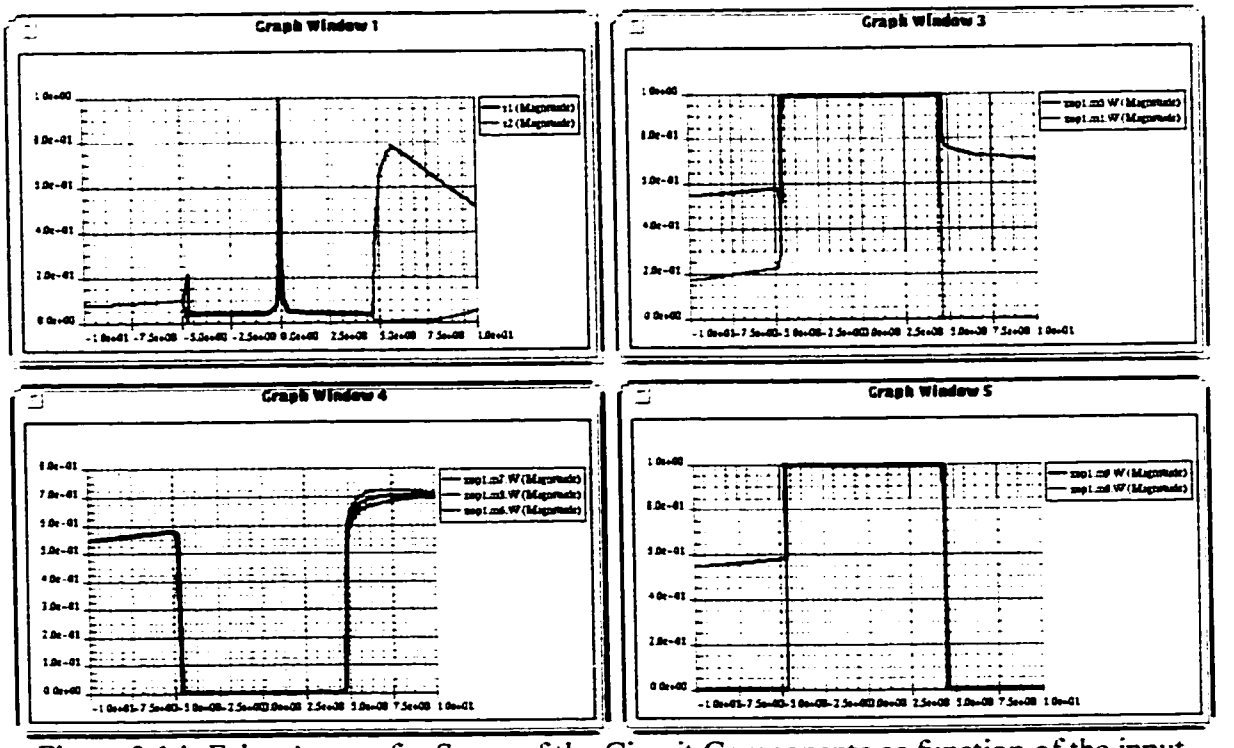

Figure 3.14: False Accept for Some of the Circuit Components as function of the input stimuli

Note that a false accepts close to 1.0 indicates that the component under test has a high probability of being classified as fault-free when it is faulty. Vice versa, a false accepts close to 0.0 indicates that the component under test has a low probability of being classified as fault-free when it is faulty.

3.10.1.5 Test Vectors

Finally, Equation 3.8 was used to compute the Bayes risk and the stimuli that minimize, this Bayes risk was selected as test vector (Equation 3.9).

In our case, for the amplifier shown in Figure 3.8, the following test vectors were obtained:

Table 3.3: Obtained test vector for each fault in the circuit

Fault name	False Accept	False Reject	Test Vector (volt)
Xop1.m1.l	7.83E-01	1.349809e-03	4.70E+00
Xop1.m1.w	8.24E-01	1.349809e-03	4.70E+00
Xop1.m2.l	2.22E-02	1.349809e-03	2.00E-01
Xop1.m2.w	1.84E-02	1.349809e-03	1.00E+00
Xop1.m3.l	2.33E-02	1.349809e-03	1.30E+00
Xop1.m3.w	3.82E-02	1.349809e-03	5.00E-01
Xop1.m4.l	3.17E-01	1.349809e-03	4.70E+00
Xop1.m4.w	3.75E-01	1.349809e-03	4.70E+00
Xop1.m5.l	2.00E-01	1.349809e-03	4.70E+00
Xop1.m5.w	2.64E-01	1.349809e-03	4.70E+00
Xop1.m6.l	3.73E-01	1.349809e-03	4.70E+00
Xop1.m6.w	4.00E-01	1.349809e-03	4.70E+00
Xop1.m7.l	9.98E-01	1.349809e-03	-4.70E+00
Xop1.m7.w	9.98E-01	1.349809e-03	-4.70E+00
Xop1.m8.l	2.26E-02	1.349809e-03	1.00E-01
Xop1.m8.w	2.08E-02	1.349809e-03	9.00E-01
Xop1.m9.l	2.32E-02	1.349809e-03	1.10E+00
Xop1.m9.w	4.15E-02	1.349809e-03	5.00E-01
R1.value	3.97E-03	1.349809e-03	-4.60E+00
R2.value	3.94E-03	1.349809e-03	-4.70E+00
Xop1.R77.value	1.00E+00	1.349809e-03	-1.00E+01

In summary, 11 test vectors where needed to test the 21 faults in the circuit (-10v, -4.7v, -4.6v, 0.1v, 0.2v, 0.5v, 0.9v, 1.0v, 1.1v, 1.3v, and 4.7v) that could be compacted and reduced to 4 test vectors (-10v, -4.6v, 0.1v and 4.7v). As a result, the obtained probability of the circuit false accepts was 0.31 and the obtained probability of the circuit false reject was 0.00128.

3.10.2 Summary of Experimental Results

A set of 7 benchmark circuits has been simulated. The benchmark circuits range from an operational amplifier to a 4 bit current DAC. Level 3, Level 28 and level 49 transistor MOSFET models are used.

All the principal components (resistances, capacitances, and transistors W and L) were considered in the fault dictionary with 5% statistical variation. The output voltage and the output current (including IDDQ) were considered for testing. The output detection threshold for false accept and false reject computation was set to $3\sigma_{out}$ (the output standard deviation of the fault-free circuit) such that $PFR \approx 0$ and $fault\,coverage = 1 - R \approx 1 - PFA$.

Circuit statistics and simulation results are summarized in Table 3.4. This table gives: the circuit name, the number of faults injected in the circuit, the obtained fault coverage, the number of test vectors used to obtain the indicated fault coverage, and their corresponding test domain.

Note that stimulus in the AC, DC and transient domains have been used for testing. The combination of these domains is also possible.

Table 3.4: Fault Simulation and Test Vector Result Summary

Circuit	Number of faults	Fault Coverage	Number Test Vectors	Test Vectors	Remarks
Cmosoa (Unity gain OpAmp) *	44	73%	3	DC (v): 9.9, -4.9, 4.8	IDD testing increased fault coverage from 63% to 83%
Comparator (open loop OpAmp)	42	91%	5	DC (v): 9.9, -0.2, 0.0, 0.1, 0.3	This test result could be misleading since for an open loop amplifier the first order sensitivities are not sufficient to adequately represent the circuit behavior.
Kfiltre	58	63%	6	AC (Hz): 165k, 229k, 239k, 138k, 0.186meg, DC (v): -10.0 v	IDD testing increased fault coverage from 48% to 63%
I_DAC 4bits	50	44%	5	Transient (input code): 1110, 1101, 1011, 0111, 0011	IDD testing did not increase fault coverage
Low pass filter	10	80%	3	Transient (s): 70u, 0.138m, 0.141m	Ideal OpAmp
Fifth order Chebychev filter	150	35%	1	AC (Hz): 15.1k	Low fault coverage was caused by low observability of the transistors width and length.
State variable filter	26	100%	5	AC (Hz): 1.34, 5.75k , 6.16k, 6.45k, 97.7k	Ideal OpAmp

*See Annex A for detailed results

It is known that soft faults are very difficult to detect mainly due to the feedback loops and to the nature of the analog signals. From the test results, soft faults could be classified into two different types: the internal faults (inside closed loop Op Amps) and external faults.

External faults are relatively easy to test since they have direct impact on the circuit outputs. The state variable circuit and the low pass circuit are perfect examples of external faults where fault coverage is 100% and 80% respectively.

Internal faults are much more difficult to detect. The Chebychev filter is a good example of low fault coverage (35%). This low coverage is mainly due to the low sensitivity of the output gain with respect to the transistors width and length.

In an attempt to increase fault coverage, the closed loop configuration has been changed to the open loop. This change increased fault coverage as we can see from the cmosoa (closed loop Op Amp) and the comparator (open loop Op Amp) cases, where fault coverage increased from 63% to 91%. However, we should be very careful while reading those results, since for circuits with high nonlinearity and with high gain, a first order sensitivity equations may not be sufficient to adequately describe the circuit behavior.

In another attempt to increase fault coverage, the stimulus space has been increased by adding new test vectors. IDD testing proved to be a major supplement to the voltage testing: In the cmosoa case fault coverage increased from 63% to 83% and in the kfiltre case fault coverage increased from 48% to 63%.

Despite the difficulties in defining and testing soft faults, our methodology efficiently simulated faults and generated test stimulus for the deterministic fault coverage.

3.11 *Conclusion*

In this paper we have presented a method for fast and accurate simulation of the fault-free and the faulty circuits. Indeed, for each fault in the fault dictionary and for all possible stimulus of a circuit, a faulty circuit PDF function is generated. The generation

of the PDF functions relies on the process information and on the circuit sensitivities to the circuit principal components in order to generate statistical models of the fault-free and the faulty circuits. Note that since no physical Monte Carlo simulation where performed, the fault simulation time was only a fraction of a full Hspice Monte Carlo simulation without a significant degradation of the accuracy of the results.

False accept and false reject values derived from the circuit statistical models are then used as classification criteria for test vector generation.

Acknowledgements

We would like to acknowledge Jim Abbatte and Patryk Lech for their help in developing many of the algorithm building blocks and the efforts and the time of Andrew Levy in data collection and presentation.

CHAPITRE 4

GÉNÉRATION DE VECTEURS DE TEST POUR LES FAUTES CATASTROPHIQUES.

Une défectuosité de fabrication, due à un ajout ou à un manque de métal (ou de silicum polycristallin, etc.), peut causer une grande déviation du paramètre de sortie de sa valeur nominale. Ce type de panne est généralement considéré comme une panne catastrophique [10], [11], [12] et [29].

Les efforts pour produire des algorithmes et des outils de CAO ont été principalement orientés vers les circuits numériques. En effet, pour les circuits numériques, un modèle de panne simple et robuste (collages) a été développé [37]. Ce modèle a permis le développement de puissants algorithmes de simulation sérielle et parallèle de fautes. Les algorithmes de simulation sérielle sont les plus simples: les fautes sont injectées une par une dans le circuit nominal et simulées. Cependant, avec l'augmentation de la taille des circuits, la méthode de test par simulation sérielle est devenue lente et inefficace. D'autres algorithmes de simulation parallèle furent développés. Ces algorithmes diffèrent de la méthode sérielle en deux aspects:

1. ils déterminent la réponse du circuit fautif sans explicitement changer le circuit et

2. ils sont capables de simuler l'effet de plusieurs fautes simultanément.

Contrairement aux circuits numériques, les recherches sur la simulation et le test des pannes catastrophiques pour les circuits analogiques n'ont pas abouti au même degré de succès. Ceci est principalement dû au manque de modèle de fautes robuste semblable à celui des circuits numériques. En prenant pour acquis le fait qu'il n'y ait pas de modèle de faute semblable à celui au collage la seule approche possible est une approche sérielle où pour chaque faute il faut explicitement:

1. changer le circuit en injectant la faute et

2. simuler ce circuit fautif.

Plusieurs problèmes sont liés à l'utilisation de cette approche; en particulier, le problème du temps de simulation, qui augmente linéairement avec le nombre de fautes à simuler.

On propose une nouvelle approche pour la génération de vecteurs de test. Celle-ci est basée sur le calcul du gradient et sur la méthode adjointe. Il faut noter que l'approche basée sur le test par la méthode adjointe est originale et présente une nouvelle philosophie de test pour les fautes catastrophiques [25]-[28].

La méthode consiste à utiliser la sensibilité par rapport à des composants inexistants dans le circuit. En effet, comme on est capable de calculer la tension de n'importe quelle paire de nœuds et le courant dans n'importe quelle branche du circuit, on est capable de calculer la sensibilité des éléments qui n'existent pas dans le circuit et, en particulier, la sensibilité des résistances qui simulent l'effet d'un court-circuit ou d'un circuit ouvert.

La puissance de cette technique (calculer la sensibilité des fautes catastrophiques et l'adapter à la méthode des circuits adjoints) vient du fait qu'elle permet d'évaluer simultanément l'effet de toutes les fautes catastrophiques sur le paramètre de sortie en seulement deux simulations: une pour le circuit initial et une autre pour le circuit adjoint correspondant.

En considérant le calcul de la sensibilité des résistances simulant les différents défauts (obtenue par la méthode des circuits adjointes décrite ci-dessus) et en connaissant la valeur de la déviation maximale tolérable sur le paramètre de sortie (par exemple 5%), l'équation (4.10) nous permet d'estimer la valeur de ces résistances qui forceront le paramètre de sortie à dépasser la marge de tolérance permise. Ces résistances sont définies comme les valeurs limites au-delà desquelles la signature du circuit sera suffisamment modifiée de façon à ce que la faute soit détectée.

Par exemple, pour simuler un défaut causé par manque de métal (circuit ouvert), on utilise la sensibilité pour estimer la valeur de la résistance qui fera sortir le paramètre de sa marge de tolérance. Si cette résistance calculée est inférieure à la résistance typique d'un circuit ouvert alors, la résistance réelle de cette faute modifiera suffisamment le paramètre de sortie pour que la faute soit détectée. Le vecteur de test est alors défini comme le stimulus pour lequel la valeur de la résistance calculée est la plus petite.

Réciproquement, pour les court-circuits, il faut que la valeur de la résistance calculée soit supérieure à la valeur de la résistance typique.

L'approche a été validée par un test sur un circuit filtre passe-bande du second ordre. Pour celui-ci, la méthode a été utilisée pour calculer la valeur des résistances de chacune

des fautes. Ces résistances ont été ré-injectées dans le circuit et simulées. On a alors observé une modification significative de la signature du circuit.

Cette méthodologie est présentée en détail dans l'article qui suit. L'article a fait l'objet d'une soumission à "IEEE Transactions on CAD" en novembre 1999.

Closing the Gap Between Analog and Digital Testing

Khaled Saab, Naim B. Hamida, Bozena Kaminska

Fluence Technology Inc., 8700 SW Creekside Place, Beaverton OR 97008 USA.

Abstract

This paper presents a highly effective method for parallel hard fault simulation and test specification development. The proposed method formulates the fault simulation problem as a problem of estimating the fault value based on the distance between the output parameter distribution of the fault-free and the faulty circuit. We demonstrate the effectiveness and practicality of our proposed method by showing results on different designs. This approach extended by parametric fault testing has been implemented as automated tools set for IC testing.

4.1. Introduction

Research in the area of analog circuit fault simulation and test vector generation has not achieved the same degree of success as its digital counterpart owing to the difficulty in modeling analog behavior, the continuous nature of their input and output signals, the non-linearity of circuit elements, and the complicated relations between input and output signal called transfer function. Thus, a direct application of digital *stuck-at* fault model proves to be inadequate for the analog circuits fault simulation and test vector generation.

Despite those difficulties, there was a high pressure from the testing community and the industry to come up with a fault model and test methodology that serve the same purpose on the analog side. One major step in this direction was the migration from functional fault model to structural fault model. This move allowed fault grading and acted as a measure to quantify the quality of the test plan, permitting test requirements and benchmarking of design-for-test strategies. However, analog testing was still done serially where faults are inserted in the circuit and simulated one at a time in a serial manner.

In this paper, we present a parallel and accelerated fault simulation method that does not rely on the simulator in the loop. The proposed method builds a statistical model of the fault-free circuit and uses the circuit sensitivity and linear approximations in order to generate a fault model for each defect in the circuit. The obtained information is used for test vector specification.

4.2. Overview

Previous work in fault simulation and test generation focuses on digital circuits using the classical *stuck-at* fault model, where serial fault simulation techniques and parallel fault simulation techniques have been developed. Serial fault simulation is the simplest method of simulating faults. It consists on transforming the model of the fault-free circuit N so that it models the circuit Nf created by the fault f. Then Nf is simulated. The entire process is repeated for each fault of interest [37]. Other fault simulation techniques: parallel, deductive, and concurrent, have been developed and differ from the serial

method in two fundamental aspects: a) they determine the behavior of the circuit N in the presence of fault without explicitly changing the model of N and b) they are capable of simultaneously simulating a set of faults.

The introduction of the *stuck-at* fault model for digital circuits enabled digital testing to cope with the exponential growth in the digital circuit size, and complexity. Indeed, the *stuck-at* fault model enabled the functional testing to be replaced by structural testing, and acted as a measure to quantify the quality of the test plan, permitting test requirements and benchmarking of design-for-test strategies. For those reasons, there is a high pressure from the testing community and from industry to come up with a fault model and a test methodology that serve the same purpose on the analog side.

Hard fault modeling and simulation which addresses analog and mixed-signal circuits has been the subject of many publications [12], [25], [26], [51] and [52]. In [12] it has been suggested that the faulty analog behavior should be modeled as a modification to the nominal macromodel. For example, the fault model for a transistor has been implemented by replacing each transistor by a transistor surrounded by switches as shown in Figure 4.1. A faulty circuit can be obtained from the good one by opening or closing the appropriate switch.

In [25] to enable the circuit fault-effects to be simulated with a reasonable simulation time, behavioral models of each circuit block where developed. Hybrid fault simulations were performed by replacing each circuit block with its behavioral model equivalent except the block that is to have a target fault inserted. Each fault is manually inserted in

the netlist for simulation. The behavioral models reduced the simulation time by a factor of 10 to 36.

However, all presented approaches for analog and mixed-signal circuits are all based on *cause-effect* analysis and do not allow parallel fault simulation. Indeed, *cause-effect* analysis enumerates all the possible faults (*causes*) existing in a fault model and determines all their corresponding responses (*effects*) to a given applied test in a serial manner. The required simulation time can become impractically long, especially for large analog designs.

The novelty of our approach consists of parallel hard fault simulation and test vector specification based on *effect-cause* analysis. Indeed, from the distance between the fault-free circuit distribution and the faulty circuit distribution (*effect*), the fault value (*cause*) for all defects could be approximated simultaneously by linear estimation. Then using fault dominance theorem and the fault value data, test vectors are derived.

The objective of this paper is to present the methodology and a practical implementation that address two fundamental problems that are limiting the growth of the testing capabilities of analog and mixed signal circuits, structural fault model and parallel simulation of faults. The paper is organized as follows. In the second and third sections, hard fault dictionary generation, fault simulation, fault coverage computation and test vector specification are presented. Implementation details, practical examples and results that demonstrate the effectiveness of the proposed methodology are included in section four. Section five concludes the paper.

4.3. Fault Simulation

Fault simulation is used to construct the fault dictionary. Conceptually, a fault dictionary stores the signatures of the faults for a specific stimulus T. This approach requires computing the response of every possible faults before testing, which is impractical.

We propose a new type of fault dictionary that does not store the output signature of the fault f(*effect*), but computes and stores the fault value R_{fault} (*cause*) that if added to the circuit, will drive the output parameter out of its tolerance box. Using this new definition of fault dictionary two steps are needed to construct the fault dictionary:

(i) First, the fault list is generated, and

(ii) Second, from the fault-free and the faulty circuit distributions, we simultaneously compute and store the fault value R_{fault} for all defects in the fault list.

4.3.1 Fault Listing

The fault list contains the list of all possible shorts and opens in a circuit. Two fault list extractors could be used: a layout-based fault list (standard inductive fault analysis) and/or a schematic-based fault list extractor. In this case all branches are considered as potential opens and all nodes of the same element are considered as potential shorts. Generations of all combinations of two, three or more shorted nodes is possible but it may lead to a large fault list containing a significant amount of nonrealistic faults.

For MOSFET transistor, the layout extracted fault list is constructed of three shorts and two opens: short gate-source, short gate-drain, short source-drain, open drain, and open source (Figure 4.1).

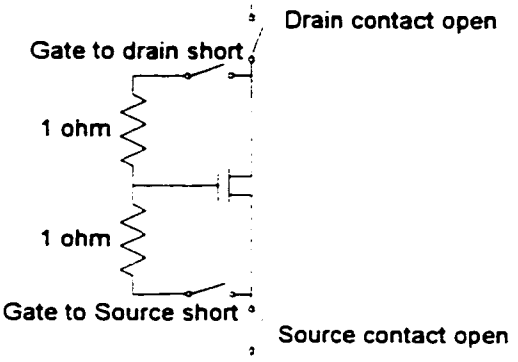

Figure 4.1 Transistor fault model

4.3.2 Fault Model (The Minimal Detectable Fault Value)

As mentioned previously, we propose to construct a fault dictionary by storing the fault value R_{fault} (*cause*) instead of storing the output parameter (*effect*). The *effect* is a constant value which represents the detectability threshold. The *cause* is the minimal defect value that, if added to the circuit, will cause the output parameter to go out of tolerance range and make the fault detectable.

Several methods could be used to define the detectability threshold at the output parameter:

(i) A constant absolute value deviation of the output parameter from its nominal value i.e. $D_{threshold} = 10mv$, or

(ii) A constant percentage of the output parameter value i.e. $D_{threshold} = 5\%$, or

(iii) A constant factor of the fault-free output distribution $3\sigma_{ff}$.

The first two methods to define the detectability threshold are straightforward. For the third method (constant factor of the output distribution), we use a piecewise linear estimation to compute the fault-free circuit output due to process variations using Equation (4.1) [9], [39]. From Equation (4.1), the fault-free output mean μ_{ff} and standard deviation σ_{ff} can be obtained by Equation (4.2) and (4.3) respectively.

$$out = out_0 + \sum_{i=1}^{N} S_{x_i}^{out} \Delta x_i \tag{4.1}$$

$$\mu_{ff} = \mu_{out_0} + \sum_{i=1}^{N} S_{x_i}^{out} \mu_{\Delta x_i} \tag{4.2}$$

$$\sigma_{ff}^2 = \sum_{i=1}^{N} \left(S_{x_i}^{out}\right)^2 \sigma_{x_i}^2 + \sum_{i \neq j; i=1; j=1}^{N} \sum_{x_i}^{N} S_{x_i}^{out} S_{x_j}^{out} \sigma_{x_{ij}} \tag{4.3}$$

where out_0 is the nominal output value and out is the estimated output value due to the components variation. Δx_i is the variation of the circuit component x_i due to process variation and $S_{x_i}^{out}$ is the sensitivity of out to x_i. σ_{x_i} is the standard deviation of the component x_i, and $\sigma_{x_{ij}}$ are the covariance terms.

Now, we compute the faulty circuit nominal value μ_f and its standard deviation σ_f due to added resistance. Equation (4.1), (4.2), and (4.3) respectively become

$$out = out_0 + \sum_{i=1}^{N} S_{x_i}^{out} \Delta x_i + G_{f_i}^{out} R_{f_i} \tag{4.4}$$

$$\mu_f = \mu_{out_0} + \sum_{i=1}^{N} S_{x_i}^{out} \mu_{\Delta x_i} + G_{f_i}^{out} \mu_{R_{f_i}} \tag{4.5}$$

$$\sigma_f^2 = \sum_{i=1}^{N} \left(S_{x_i}^{out}\right)^2 \sigma_{x_i}^2 + \sum_{i \neq j=1}^{N} \sum_{j=1}^{N} S_{x_i}^{out} S_{x_j}^{out} \sigma_{x_i x_j} + \left(G_{f_i}^{out}\right)^2 \sigma_{R_{f_i}}^2 + \sum_{i=1}^{N} \sum_{j=cte} G_{f_j}^{out} S_{x_i}^{out} \sigma_{x_i R_{f_j}} \tag{4.6}$$

where R_{f_i} is the newly added component due to a short or open and $G_{f_i}^{out}$ is the gradient of out with respect to the fault f_i. $\sigma_{R_{f_i}}$ is the standard deviation of the resistance, and $\sigma_{x_i R_{f_j}}$ the covariance term between the newly added component and the components in the original circuit.

Since $\sigma_{R_{f_i}}$ is always zero, and if we consider the variable as independent, the covariance terms $\sigma_{x_i R_{f_j}}$ are zero, the expression for the faulty output variance is greatly simplified, and Equation (4.6) reduces to

$$\sigma_f^2 = \sum_{i=1}^{N} \left(S_{x_i}^{out}\right)^2 \sigma_{x_i}^2 + \sum_{i \neq j=1}^{N} \sum_{j=1}^{N} S_{x_i}^{out} S_{x_j}^{out} \sigma_{x_i x_j} \tag{4.7}$$

Thus, under the above assumptions, hard defects do not modify the fault-free circuit standard deviation but affect only the mean value and $3\sigma_f = 3\sigma_{ff}$.

Now that the fault-free and faulty output distributions are obtained, the detectability threshold is set to be the minimal distance between the fault-free and faulty circuit that guarantee detectability of the fault (Figure 4.2)

$$D_{threshold} = \mu_f - \mu_{ff} \approx 3\sigma_f + 3\sigma_{ff} \tag{4.8}$$

where μ_{ff} and μ_f are the mean values for the fault-free and faulty circuit and σ_{ff} and σ_f are the estimated fault-free and faulty output standard deviations.

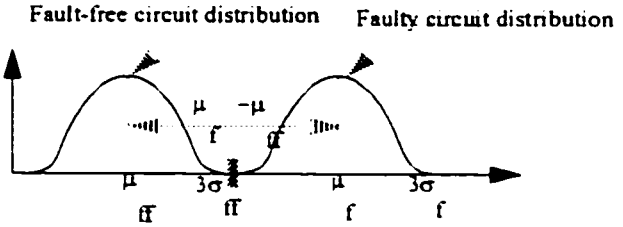

Figure 4.2 Fault-free and faulty circuit output voltage

The fault value R_{f_i} could now be obtained from Equation (4.2) and (4.5):

$$R_{f_i} \approx \frac{|\mu_f - \mu_{ff}|}{G^{out}_{f_i}} \approx \frac{|3\sigma_f + 3\sigma_{ff}|}{G^{out}_{f_i}} \tag{4.9}$$

$$R_{f_i} \approx \frac{6\sigma_{ff}}{G^{out}_{f_i}} \tag{4.10}$$

combining (4.3) and (4.10), we obtain

$$R_{f_i} \approx \frac{6\sqrt{\sum_{i=1}^{N}\left(S_{x_i}^{out}\right)^2\sigma_{x_i}^2 + \sum_{i\neq j=1}^{N}\sum_{j=1}^{N} S_{x_i}^{out} S_{x_j}^{out}\sigma_{x_{ij}}}}{G_{f_i}^{out}} \tag{4.11}$$

S and G are obtained using the well known adjoint network method [18]-[22].The adjoint network method allows us to compute the sensitivity of one output parameter with respect to all component variations (existing and non-existing components) in only two simulations, one for the original network and one for the corresponding adjoint network. The adjoint network method for sensitivity computation in AC, DC and transient domain has been implemented using Hspice [30] as a basic simulator [9], [39] and [43].

In summary, the algorithm consists of two steps. First, from fault free circuit simulation, we compute the fault-free mean and standard deviation of the output parameters due to process variations [9]. Then, in the second step, from the output parameter distributions and gradient values, we compute and store the resistance values that will drive the output parameter(s) out of their tolerances (Figure 4.3). Note that the obtained resistance value indicates the value of the resistance that, if added to the branch (open circuit case) or between two nodes (short circuit case), will cause the output parameter to go out of its tolerance range. This resistance value will be used for test vector specification.

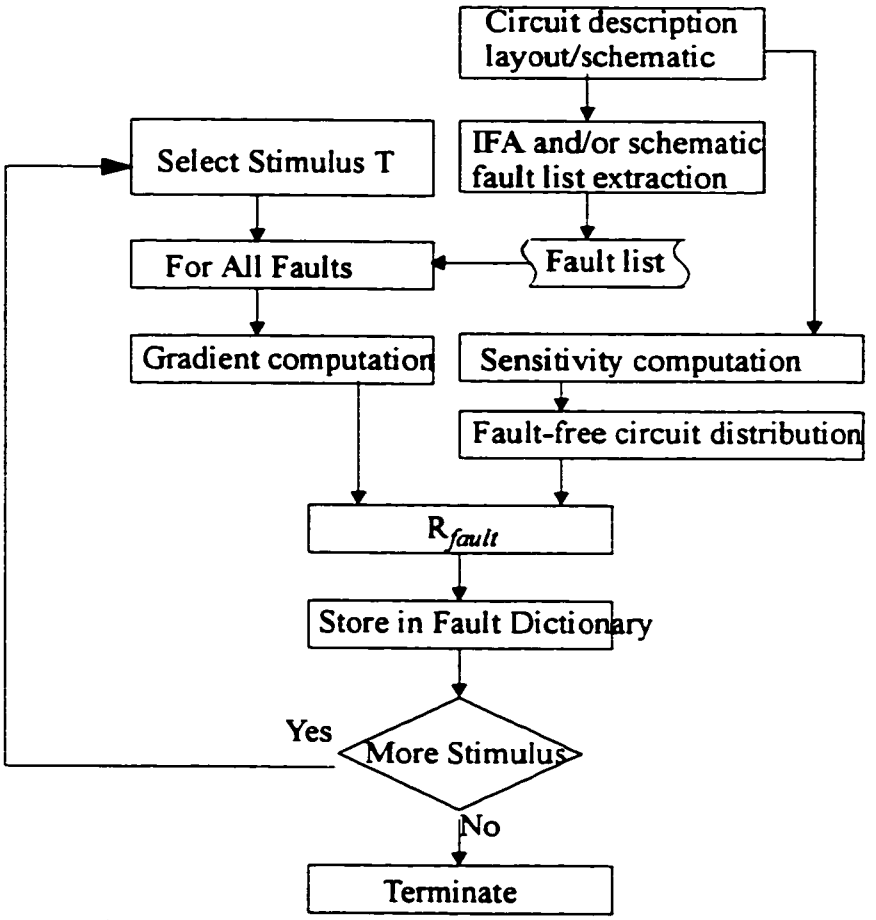

Figure 4.3 Fault Dictionary Construction Flow Chart.

4.4. Test Vector Generation

In this section we describe an algorithm which uses the fault dictionary generation approach proposed earlier and the fault dominance concept to derive the fault coverage and the set of test specifications that detect the largest set of faults without targeting individual faults.

4.4.1 Fault Dominance

In digital circuit, fault dominance is used to reduce the number of faults that need to be considered.

Definition [37]: Let T_g be the set of all tests that detect a fault g. We say that a fault f dominates the fault g if f and g are equivalent under T_g.

In other words, if f dominates g, then any test that detects g, will also detect f (on the same primary output). Therefore, for fault detection it is unnecessary to consider the dominating fault f, since by deriving a test to detect g we automatically obtain a test that detects f as well.

In analog circuit, the input output relationship is more complex, but to the first order approximation, the fault dominance theorem could be used as well.

Indeed, instead of testing for the upper and lower resistance values for faults as in [25] (Table 4.1), in our case, fault dominance will be used where only the least dominating resistance (the largest) will be tested for shorts, and the least dominating resistance (the smallest) will be tested for open circuit.

TABLE 4.1 Upper and lower resistances used for hard fault modeling.

Defect Type	Lower Resistance (ohms)	Upper Resistance (ohms)
Added metal 1	0.2	1000
Added metal 2	0.2	1000
Via short	5	5
Junction leakage	100	10 000
Poly-metal 1 short	0.2	1000
Poly-metal 2 short	0.2	1000
Poly-poly short	20	1000
Open	1Meg	∞

4.4.2 Test Vector Specification

The algorithm (see Figure 4.4) consists of four main steps: a) select a set of stimuli. A default stimulus could be used or it can be made by the test engineers in an interactive mode in order to consider any special characteristic of the circuit. Stimuli are divided into DC, AC and transient stimulus [26]: sine wave, pulse, ramp, any PWL function, etc. b) construct the fault dictionary and obtain R_{fault}, then c) loop through all stimuli and all faults in the fault dictionary and compare the fault value R_{fault} with the typical values for short and opens (Table 4.1), and d) evaluate the proposed test T, then go to the next candidate T. This iteration loop allows one to select the test vector that maximizes the observability of a fault.

The technique for open circuit faults will be described in the following section. The exact same reasoning is used for short circuit faults.

Knowing the component tolerance Δx_i, the output parameter distribution of the fault-free and faulty circuit $3\sigma_{ff}$ and $3\sigma_f$ can be estimated. Then from equation (4.11) we estimate the resistor value R_{fault} that will cause the output parameter to go out of the tolerance range. R_{fault} are stored in fault dictionary.

Table 4.1 presents a list of circuit defects and the corresponding typical resistance range obtained by statistical analysis of circuit defects [12]. R_{fault} is compared to $R_{typical}$. If R_{fault} is less than $R_{typical}$, we conclude that R_{fault} (or a higher resistance value) *will cause the output parameter to deviate out of tolerance, and the stimulus T is* ***accepted*** *as a valid test vector*. On the other hand, if R_{fault} is higher than $R_{typical}$, no deviation of the output parameter could be detected and the stimulus T is **rejected**.

Since shorts are the dual of opens, we do not need to repeat the steps above where the same technique is applied, except that R_{fault} needs to be higher than $R_{typical}$ in order to cause a detectable deviation in the output parameter signature.

The structure of the hard fault test vector specification tools is given in Figure 4.4.

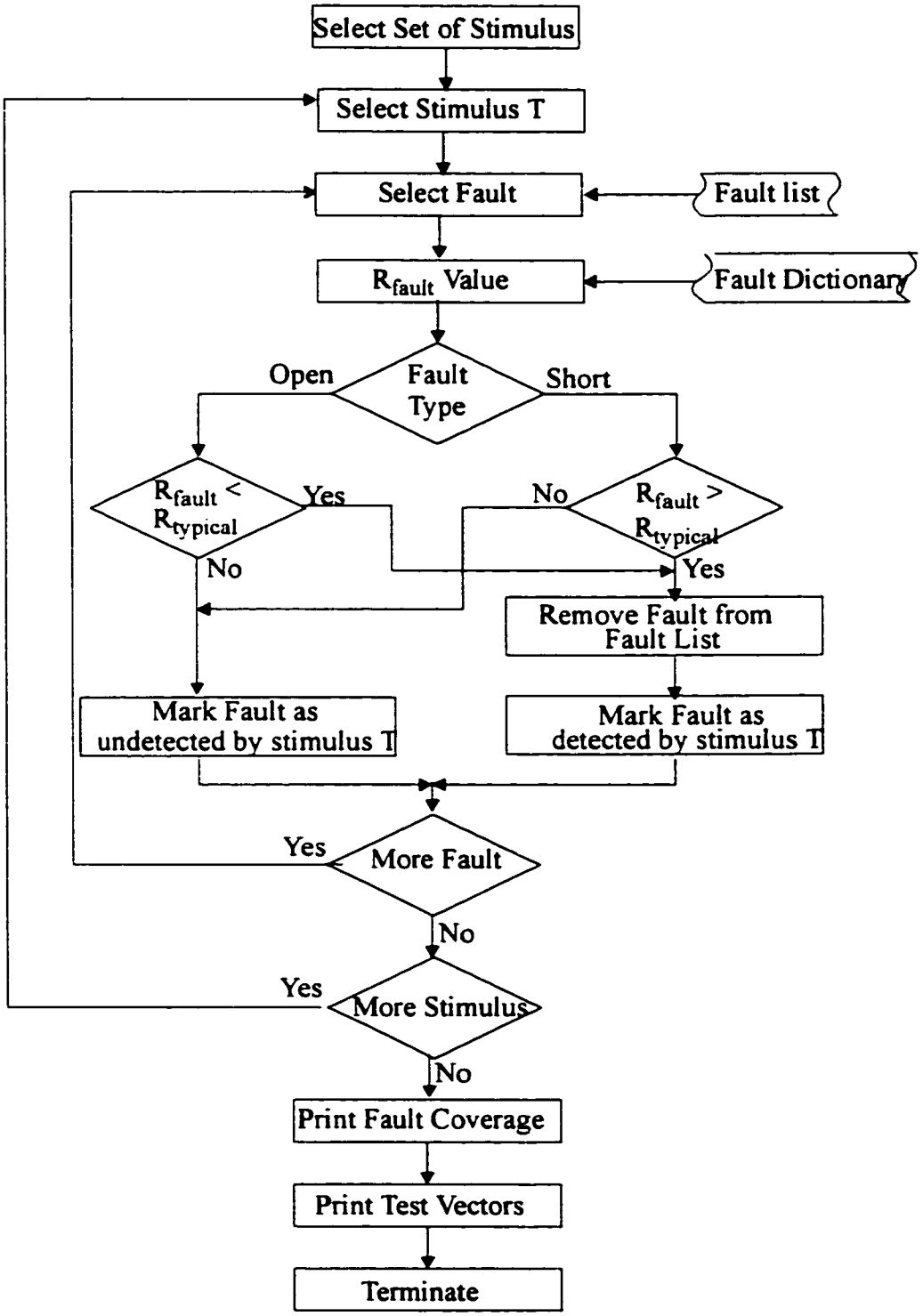

Figure 4.4 Test Vector Generation Flow Chart

4.5. Experimental Results

The proposed algorithm has been implemented as an automated tool set for IC testing, fault simulation and test vector generation. The implementation was done in the C programming language on a SUN workstation.

The procedures outlined in this paper were applied to the second order band-pass filter shown in Figure 4.5, to the fifth order chebychev band-pass filter (Figure 4.8), and to several benchmark circuits as shown in Table 4.4.

Example 1:

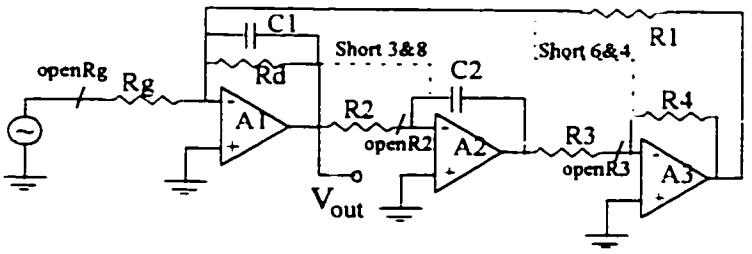

Figure 4.5 Second-Order Band-Pass Filter

Rg=200k, R1=10, R2=10k, R3=10k, R4=10k, Rd=200k, C1=1.59nf, C2=1.59nf

The gradient method has been used for the band-pass filter testing at the circuit's nominal frequency response, 10 kHz. For simplicity, three possible opens and two possible shorts circuits were considered, Figure 4.5. The output voltage was used as a detection mechanism and the detection threshold was set to 5% of the nominal voltage

value. In other words, the algorithm was used to estimate the resistance value R_{fault} beyond which the signature of the output parameter will be modified by more than 5% (Figure 4.6).

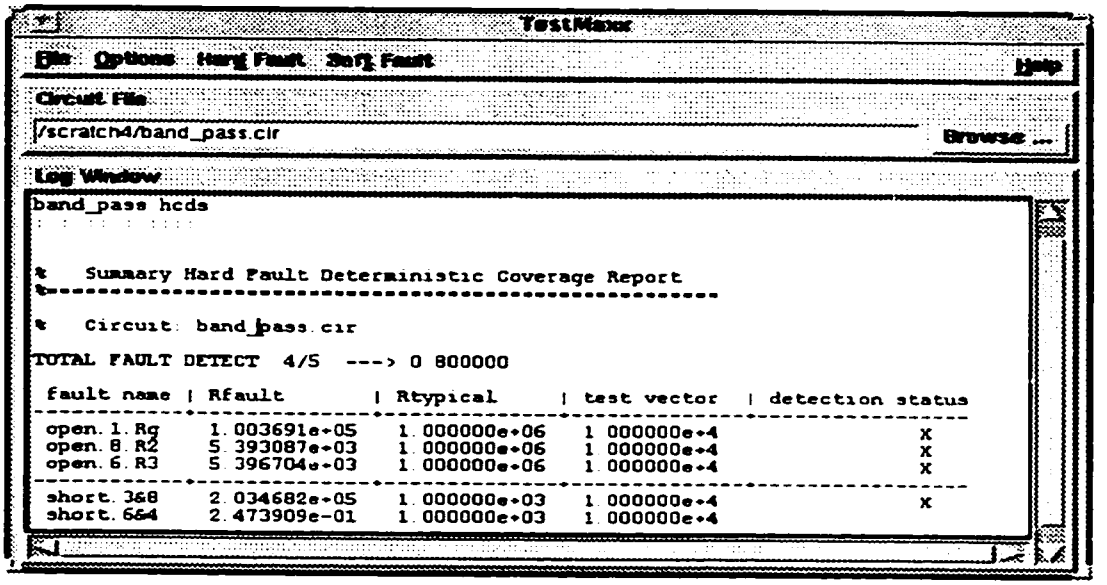

Figure 4.6 :Computed Fault Resistance Values R*fault*

In this example, four out of the five faults has been detected; R_{fault} for OpenRg, Open.R2, and Open.R3 was less than $R_{typical}$ and the test vector (10KHz sine wave) is marked as a valid test vector. For short3&8 R_{fault} was higher than $R_{typical}$ and the test vector was also marked as a valid test vector for this fault. However, for short6&4, R_{fault} was less than $R_{typical}$ and the test vector was reject as a non-valid.

To validate the effectiveness of this method, each fault was reinjected manually into the circuit and simulated. Figure 4.7 shows the results of fault simulation over the considered catastrophic defects. In this figure, each fault is analyzed to confirm whether or not it was detected at the output node at the 10KHz test frequency. Indeed, all the accepted resistances modified the signature of the fault-free output voltage by more than 5% which has been determined as a detection criteria.

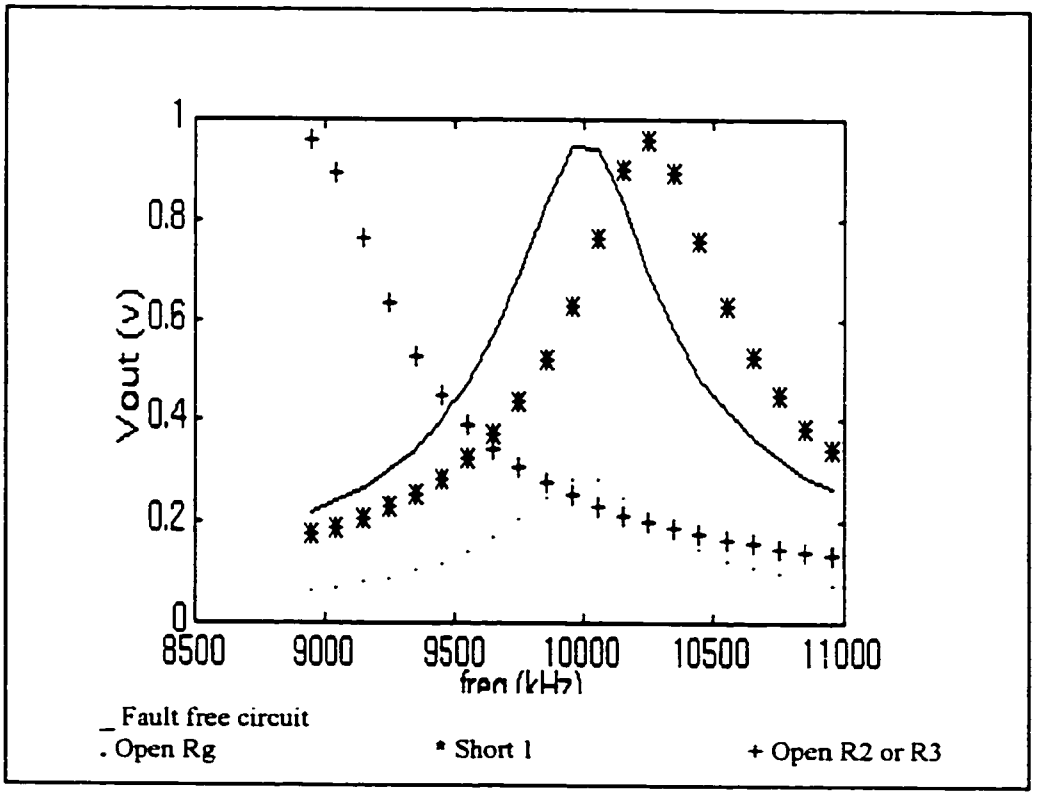

Figure 4.7 SPICE Output voltage For The Fault-Free Circuit and The Faulty Circuit.

Example 2:

The fifth order Chebychev filter (Figure 4.8), is tested for 25 possible hard faults. The stimulus set T was selected as the frequency range [0,20KHz]. The detection threshold was set to 5% of the nominal output voltage, $R_{typical}$ for open set to 1Meg ohm, and $R_{typical}$ for shorts set to 1Kohm.

It is found that all the relevant possible shorts and open are easily detected. Table 4.2 and Table 4.3 give some possible hard faults, their computed nominal values and the frequency at which they are detected.

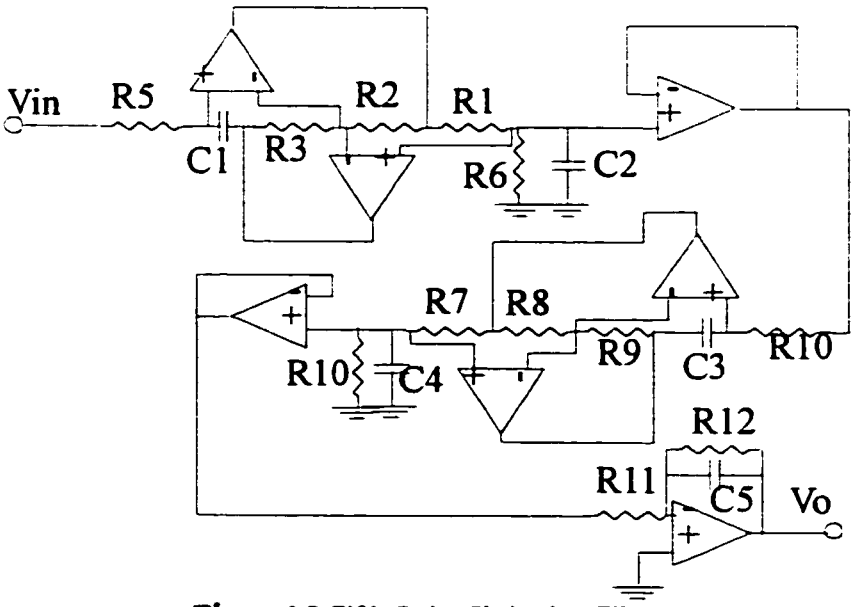

Figure 4.8 Fifth Order Chebychev Filter

TABLE 4.2 Fifth Order Chebychev Filter, Open Circuit Fault Results

OPEN	R_{fault} (ohm)	$R_{typical}$ (ohm)	Frequency (Hz)	Decision
R1	1.1k	1Meg	300	Accepted
R2	1.93k	1Meg	300	Accepted
R3	400	1Meg	0	Accepted
R4	662.5	1Meg	0	Accepted
R5	4.83k	1Meg	100	Accepted
R6	2.78k	1Meg	200	Accepted
R7	2.78k	1Meg	200	Accepted
R8	431	1Meg	0	Accepted
R9	555	1Meg	0	Accepted
C1	5.12K	1Meg	100	Accepted
C2	4.43k	1Meg	100	Accepted
C3	2.89k	1Meg	100	Accepted
gm1	872	1Meg	200	Accepted
gm2	243	1Meg	0	Accepted
gm3	1k	1Meg	1k	Accepted

TABLE 4.3 Fifth Order Chebychev Filter, Short Circuit Fault Results

SHORT	R_{fault} (ohm)	$R_{typical}$ (ohm)	Frequency (Hz)	Decision
Vin & 0	INF	1k	all frequencies	Accepted
VDD & 0	INF	1k	all frequencies	Accepted
VSS & 0	INF	1k	all frequencies	Accepted
R1 & 0	20Meg	1k	0	Accepted
R1 & R2	8Meg	1k	0	Accepted
R5 & R6	85k	1k	0	Accepted
R9& 0	50M	1k	1k	Accepted
in-&0	25Meg	1k	0	Accepted
in+ & 0	2e-5	1k	0	Rejected

The obtained fault coverage is 96% with only 5 test vectors (0Hz, 100Hz, 200Hz, 300Hz, 1KHz).

Experimental results:

A set of 7 benchmark circuits are simulated. The benchmark circuits range from a simple operational amplifier to a complex 8 bit current DAC. Level 3, Level 28 and level 49 transistor MOSFET models are used in the sensitivity computation environment.

In all experiments the output voltage measurements and/or output current measurements were considered for testing. The detection threshold was set to 6σ where σ is the standard deviation of the fault-free circuit. Table 4.4 describes the circuit type, the test domain, the number of transistors in the circuit, the number of faults in the fault list obtained by a schematic fault dictionary extractor, and the total CPU time on a SPARC 10 workstation. The approximated CPU time is given for serial methods using one simulation per fault.The obtained fault coverage is also provided for comparison purposes.

TABLE 4.4 Benchmark Results

Circuit	Test domain	Number of transistors	number of faults	TestMaxx CPU time (s)	Serial method CPU time(s)*	fault coverage
Inverter	DC	9	47	0.28	5.68	91.5%
low pass filter	Transient	9	51	0.43	7.92	94.1%
State variable filter	AC	36	184	3.64	334.88	85.3%
Chebychev filter	AC	27	149	5.32	348.16	87.2%
4 bit ADC	DC	153	737	0.19k	36.68k	85.7%
8 bit ADC flash	DC	63	295	0.02k	1.66k	87.4%
8 bit current DAC	Transient	132	669	1.20k	267.6k	98.6%

*estimated based on one simulation per fault

The average fault coverage was 90% with small CPU cost. There was no way to compare the fault coverage and the simulation time to other publications due to the large variety of analog benchmarks.

Figure 4.9 present the fault coverage as function of the number of test vectors.

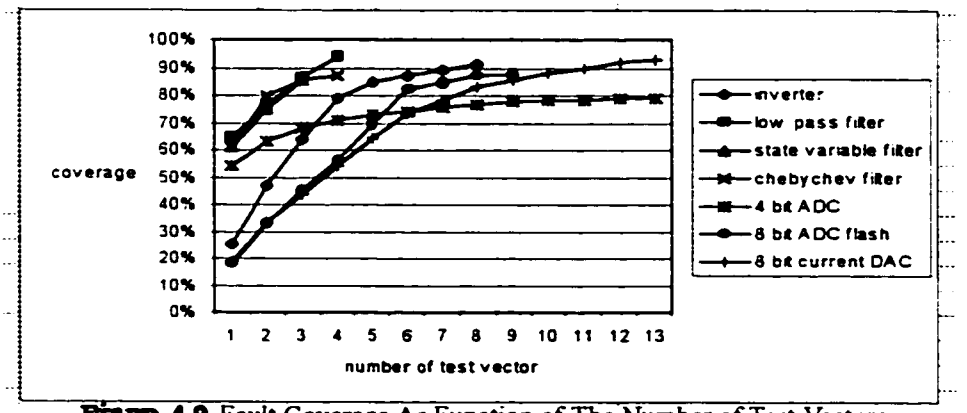

Figure 4.9 Fault Coverage As Function of The Number of Test Vectors

4.6. Conclusion

By using structural fault model and by moving from the standard serial test approaches to a parallel test approach, mixed-signal circuit testing is now one step closer to the digital testing. Indeed, an automatic tool for parallel fault simulation and test vector generation based on cause-effect approach has been presented. The concept is new, and from our simulation results, we can conclude that this method is highly efficient. Several examples and test cases showed that fault coverage was as high as 98.6% with simulation time several order of magnitude less than the serial methods.

Acknowledgments

We wish to thank Jim Abbat for his help in generating the results.

CHAPITRE 5

COMPACTION DE VECTEURS DE TEST

5.1 Introduction

L'analyse de la sensibilité des circuits analogiques a été présentée comme un moyen pour augmenter l'observabilité des effets d'une composante défectueuse à la sortie [2], [3], [4], [5] et [6]. Cette méthode est principalement basée sur la sélection du paramètre de test et du stimulus qui maximise l'observabilité de la variation de la composante sous-test pour le paramètre en question. Comme présenté dans le chapitre précédent, il existe un lien direct entre sensibilité et faux-rejet, fausse-acceptation et observabilité. Il est évident que la fonction de sensibilité (donc faux-rejet, fausse-acceptation et observabilité) dépend directement du stimulus appliqué.

Pour le test d'un circuit, il est nécessaire d'obtenir les vecteurs de test qui minimisent la fonction le faux-rejet et la fausse-acceptation et maximisent l'observabilité. Or, pour chaque vecteur de test, un coût peut lui être associé. Ce coût dépend de plusieurs facteurs, et principalement du temps que la sonde reste sur le circuit. Il est donc nécessaire de sélectionner les vecteurs de test à coût faible, de diminuer le nombre de vecteurs de test à appliquer et ceci en gardant la même couverture originale. D'où la

nécessité de recourir à un algorithme d'optimisation, afin de trouver la liste optimale des vecteurs.

5.2 Objectif

Dans les chapitres précédents (chapitre 3 et chapitre 4), nous avons développé différentes approches de génération des vecteurs de test pour les circuits analogiques. Une certaine couverture de panne est alors garantie par ces vecteurs.

Dans ce chapitre, notre objectif est d'utiliser les algorithmes d'optimisation combinatoire et plus précisément, l'algorithme du recuit-simulé pour la minimisation du nombre de vecteurs de test, tout en gardant une couverture de pannes quasi optimale (aussi proche que possible de la couverture obtenue avant compaction).

5.3 Formulation du problème

Soit C le circuit donné, $V = \{v_1, v_2, \cdots, v_n\}$ l'ensemble des vecteurs de test susceptible d'être appliqué à C, et $F = \{f_1, f_2, \cdots, f_m\}$ l'ensemble des fautes possibles de C. Soit également fc_{ij} la couverture de fautes associée à la faute f_j si le vecteur de test v_i est utilisé comme vecteur de test pour tester la faute en question. Et soit $FC_j = \{fc_{j1}, fc_{j2}, \cdots, fc_{jm}\}$ l'ensemble des couvertures de fautes associé avec chaque faute (Figure 5.1).

Le problème de la compaction des vecteurs de test peut être maintenant formulé comme la façon d'obtenir le nombre minimum et l'arrangement optimal des vecteurs de test donnant la plus grande couverture de fautes. Notons également que la définition de la couverture de fautes change avec le type de données traitées. Ainsi, la couverture de fautes signifie une ou plusieurs choses suivant le problème à résoudre:

1. Soit minimiser la probabilité d'accepter un circuit défectueux, ce qui implique la minimisation de la fonction:

$$fc_{ij} = P(faux\ accepte_{ij}).$$

(Voir le test des fautes douces, chapitre 3).

2. Soit minimiser la probabilité de rejeter un circuit sain, ce qui implique la minimisation de la fonction:

$$fc_{ij} = P(faux\ rejet_{ij}).$$

(Voir le test des fautes paramétriques, chapitre 3).

3. Soit minimiser le risque de Bayes comme présenté au chapitre 3. Ceci a pour but de minimiser le risque de prendre une décision incorrecte sur l'état du circuit. Dans ce cas, la fonction à minimiser devient:

$$\begin{aligned} fc_{ij} = {} & C_{FA_j} P(fausse\ acceptation_{ij}) P(que\ la\ faute\ existe_j) \\ & + C_{FR_j} P(faux\ rejet_{ij})(1 - P(que\ la\ faute\ existe_j)) \end{aligned}$$

(Voir le test des fautes catastrophiques, chapitre 3, équation 3.8).

4. Pour maximiser la détectabilité des fautes catastrophiques, ce qui implique la maximisation de la fonction:

$$fc_{ij} = \begin{cases} 0 \text{ si la faute } j \text{ n'est pas détectée par le vecteur } v_i \\ 1 \text{ si la faute } j \text{ est détectée par le vecteur } v_i \end{cases}$$

(Voir le test des fautes catastrophiques, chapitre 4).

$$F = \begin{Bmatrix} f_1 \\ \\ f_j \\ f_m \end{Bmatrix} \quad FC = \begin{Bmatrix} fc_{11} & fc_{21} & & fc_{n1} \\ \vdots & \ddots & & \\ \vdots & & fc_{ij} & \\ fc_{1m} & \cdots & & fc_{nm} \end{Bmatrix} \quad V = \{ v_1 \quad v_2 \quad \cdots v_i \quad v_n \}$$

Figure 5.1:Matrice de couverture de fautes

5.4 Généralisation de la formulation du problème

La formulation donnée précédemment reste incomplète. En effet, elle ne prend pas en compte certaines considérations pratiques affectant la solution finale, telle que le type de mode de test du circuit en particulier. En effet, un circuit C peut être testé dans le mode AC, dans lequel on considère un modèle petit signal du circuit et où les vecteurs de test sont des sinusoïdes de différentes fréquences, ou bien dans le mode DC, où les vecteurs de test sont simplement des tensions DC. Mais ce circuit C peut être également testé par un dernier mode : le mode transitoire où, pour une excitation donnée du circuit (rampe, impulsion, sinusoïde), le vecteur de test est représenté par le moment où la sortie doit être mesurée.

Apres avoir pris en considération les modes d'opération (AC, DC, Transitoire), nous devons être en mesure, soit de maximiser la couverture de fautes (fautes catastrophiques), soit de minimiser les probabilités de faux-rejet, de fausse acceptation ou du risque de Bayes (fautes paramétriques).

C'est pourquoi l'algorithme du problème généralisé doit considérer l'optimisation globale du problème en trouvant une solution qui minimise le nombre de vecteurs de test tout en ayant une combinaison (dans un ou plusieurs modes) maximisent la couverture de fautes.

Afin de répondre à toutes ces possibilités, la matrice de couverture de fautes (Figure 5.1) doit être agrandie horizontalement pour pouvoir traiter tous les modes possibles, mais aussi verticalement pour les multiples types de couvertures de faute (Figure 5.2).

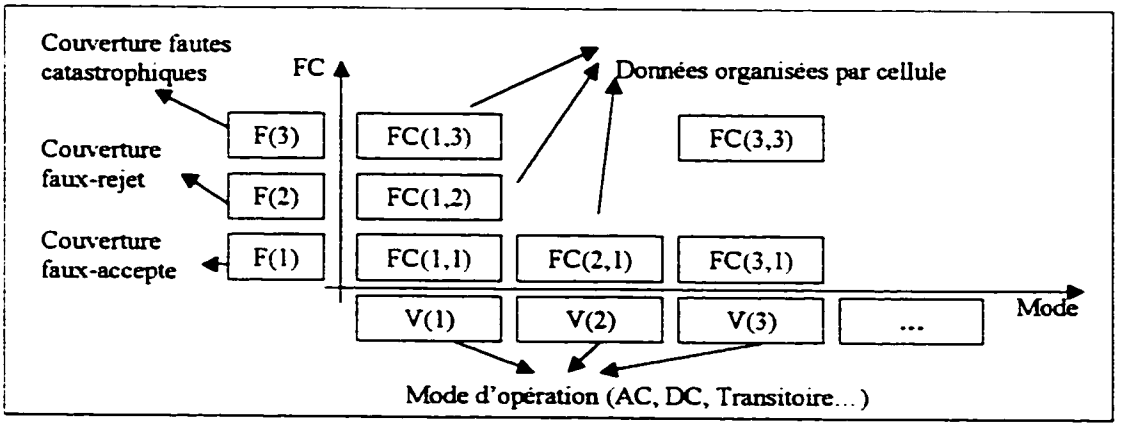

Figure 5.2: Représentation interne des données

5.5 Approche

Avec les définitions énoncées ci-dessus, nous pouvons remarquer que le problème de la compaction des vecteurs de test est un problème du type NP-complet, dans lequel le temps de calcul augmente de façon exponentielle avec n (n étant le nombre de vecteurs de test possibles); ce temps devient rapidement prohibitif en coût avec l'augmentation de n. Comme les problèmes de compaction présentent beaucoup de minima locaux, trouver un minimum (même un minimum sous-optimal) parmi ceux-ci est largement suffisant. La méthode du recuit-simulé permet de résoudre ce type de problème tout en nous cantonnant à des faibles puissances de n.

5.5.1 Les algorithmes d'approximation et l'optimisation combinatoire

Pour un grand nombre de problèmes pratiques et théoriques, l'objectif consiste à choisir la meilleure solution parmi un grand nombre de solutions possibles. Ce problème fait partie des problèmes typiquement connus sous le nom de problèmes d'optimisation combinatoire. Dans la plupart d'entre eux, la solution consiste en un arrangement d'un ensemble d'objets discrets répondant à un ensemble de contraintes données. La solution est également appelée configuration et l'ensemble des solutions constitue l'espace de solution. Une fonction objectif f(X) attribuée à chaque solution X définie le de coût de la solution. Notre but est de trouver des algorithmes efficaces pour déterminer la configuration qui minimise la fonction objectif.

Malheureusement, à cause de beaucoup de problèmes posés par l'optimisation combinatoire, des algorithmes efficaces déterminant les solutions optimales restent inconnus à l'heure actuelle. En fait, un grand nombre de problèmes se rapportent à la classe dite des problèmes NP-complets pour lesquels le temps de calcul, nécessaire pour trouver une solution exacte, augmente de manière exponentielle (exp.(cste*N)) avec la taille du problème (N) [31]; ce temps devient alors rapidement prohibitif. En pratique, des algorithmes efficaces d'approximation, qui ne donnent pas la solution optimale mais des solutions proches de l'optimal, ont été développées.

Une méthode générale de conception de ces algorithmes d'approximation consiste à apporter des améliorations itératives. Cette méthode est fondée sur la technique simple et naturelle de l'essai et de l'erreur. L'utilisation de la méthode d'amélioration itérative nécessite la définition de l'espace de solution, d'une fonction objectif, ainsi que d'un ensemble d'actions qui peuvent être utilisées dans le but de modifier une solution. Une solution Y est dite solution voisine de la solution X si Y peut être obtenu à partir de X à l'aide d'une et une seule de ces actions. Dans la méthode de l'amélioration itérative, on commence avec une solution initiale et on examine toutes les valeurs voisines jusqu'au moment où une solution Y de moindre coût est trouvée. Dans ce cas, la solution Y devient la nouvelle solution et le processus est réitéré: on examine à nouveau les solutions voisines de la nouvelle. L'algorithme s'arrête quand aucune nouvelle solution de moindre coût ne peut être trouvée. La méthode du recuit-simulé est une extension de cette méthode. Elle est fondée sur l'analogie faite entre un problème d'optimisation combinatoire et la détermination de l'état de plus faible énergie d'un système physique.

5.5.2 La méthode du recuit-simulé

Le principe de la méthode du recuit-simulé repose sur l'analogie faite avec la thermodynamique et, plus précisément, sur la façon dont les liquides gèlent et se cristallisent, ou encore sur la façon dont les métaux refroidissent et recuisent [31]. À haute température, les molécules d'un liquide bougent librement. Si le liquide est refroidi doucement, cette mobilité est réduite. Pour des températures assez basses, la mobilité est complètement perdue et, dans ce cas, les atomes s'organisent formant généralement un cristal pur, complètement ordonné sur une distance qui peut atteindre 1 milliard de fois la taille d'un atome individuel et ceci de manière isotrope. Le cristal est l'état d'énergie minimale d'un système. Le fait étonnant est que si l'on refroidit rapidement le métal liquide (trempe), celui-ci n'atteint pas l'état de plus basse énergie mais un état polycristallin ou amorphe ayant un niveau d'énergie plus élevé.

Le principe du processus consiste donc en un refroidissement lent, nécessitant un temps suffisamment long pour une redistribution des atomes dont la mobilité diminue avec le temps. Ceci est la définition même du recuit et c'est essentiel pour s'assurer qu'un état de basse énergie est atteint.

L'analogie n'est pas parfaite, mais nous pouvons tout de même considérer que tous les algorithmes de minimisation correspondent à un refroidissement rapide ou trempage. En effet, dans tous les cas, nous cherchons une solution par le chemin le plus court possible, c'est à dire une solution très proche: au point de départ, nous empruntons le chemin qui semble le plus prometteur (par analyse du gradient par exemple) et nous continuons sur

ce chemin jusqu'au moment où aucune amélioration n'est possible. Vu que cette approche ne considère pas la possibilité de prendre un chemin moins prometteur, elle risque alors de nous donner un minimum local et pas forcément le minimum global.

La nature a son propre algorithme de minimisation fondé sur un processus différent: la probabilité de distribution de Boltzmann.

Equation 5.1 $\text{Prob}(E) = \exp(-E/kT)$

Cette équation exprime l'idée que, pour un système à l'équilibre thermique et à une température T, son énergie se distribue sur différents états d'énergie. Même à une basse température, il existe une chance, même infime, que le système se trouve dans un état de haute énergie; il est donc possible que le système sorte d'un minimum local à la faveur d'un minimum global. Généralement, la température appliquée au système peut être reliée à l'énergie du système par une constante naturelle: k, la constante de Boltzmann. Cette formulation permet au système de diminuer son énergie ou de l'augmenter; mais plus la température sera basse, plus les excursions vers des états d'énergie plus élevée seront improbable.

En 1953, Metropolis et ses collaborateurs ont été les premiers à appliquer ces principes dans des calculs numériques. Avec un grand nombre d'options, un système thermodynamique est supposé changer sa configuration d'un état d'énergie E_1 vers une énergie E_2 avec une probabilité [36]

Equation 5.2 $\text{Prob}(\Delta E) = \exp(-(E_2 - E_1)/kT)$

Nous remarquons que si $E_2<E_1$, cette probabilité est plus grande que l'unité; dans ce cas, nous assignons de façon arbitraire une probabilité de $\text{Prob}(\Delta E) = 1$, c'est à dire que le système passera obligatoirement de l'état d'énergie E_1 à l'état d'énergie E_2.

L'approche qui consiste à toujours prendre une voie descendante et certaines fois une voie montante est connue comme étant l'algorithme de Metropolis [31].

Afin de pouvoir utiliser l'algorithme de Metropolis, les éléments suivants doivent être fournis:

1. Une description des configurations possibles du système. Une configuration est définie comme une solution, obtenue grâce à un arrangement d'objets discrets répondant à un ensemble de contraintes données.

2. Un générateur de changements aléatoires de ces configurations ; ces changements seront les options ou déplacements présentés par le système.

3. Une fonction objectif E (analogue à l'énergie) où la minimisation est le but de la procédure.

4. Un paramètre de contrôle T (analogue à la température) et un calendrier de recuit expliquant comment est fait le refroidissement, c'est-à-dire combien de changements aléatoires sont permis avant de faire baisser la température et de combien on baisse celle-ci (la largeur du pas). La définition de la température initiale (l'état liquide à haute température), du pas et de la température finale (l'état solide) dans ce contexte de calendrier dépend de chaque système et ceci peut demander une connaissance physique du système ou une série d'expérimentations.

5.6 Implémentation

Avant d'atteindre une solution finale, plusieurs étapes sont nécessaires. Il faut:

1. obtenir une solution consistant à obtenir une combinaison de vecteurs X (qui implique la définition de l'espace des solutions et la recherche des solutions voisines),

2. calculer le coût de la solution,

3. calculer également la couverture de fautes correspondantes

4. et enfin, suivant le coût de la solution et suivant la température du système, accepter ou rejeter la solution proposée [14].

Dans les paragraphes suivants, une description de chacune de ces étapes est présentée.

5.6.1 L'espace des solutions

L'espace des solutions est constitué par l'ensemble des combinaisons possibles entre les vecteurs de test. L'algorithme peut démarrer avec n'importe quelle condition initiale (même une solution vide) et itérer jusqu'à ce que la meilleure combinaison soit trouvée.

5.6.2 Les solutions voisines

Nous allons maintenant définir la notion de solution de voisinage dans l'ensemble des solutions. Soit un ensemble de vecteurs de test X, nous définissons 3 types d'actions pouvant modifier X:

M1: L'échange de deux vecteurs, l'un appartenant à X et l'autre n'appartenant pas à X.

M2:La suppression d'un vecteur de l'ensemble X

M3:L'ajout d'un vecteur à l'ensemble X

Deux ensembles valides de vecteurs de test X1 et X2 sont dits voisins si l'un peut être obtenu à partir de l'autre uniquement grâce à l'une de ces trois actions. Notons que, la sélection de l'action suivante est alors réalisée de manière aléatoire: la décision d'appliquer ou non cette action dépendra de l'énergie et de la température du système (Equation 5.1).

Nous allons maintenant considérer les effets de ces actions sur la fonction coût. Il est clair que l'action de type M1 a peu d'effet sur le coût, l'action M3 augmente le coût

alors que l'action M2 le fait baisser. En général, les actions de type M2 encouragent la réduction du nombre de vecteurs de test, alors que des actions de type M3 empêchent l'algorithme de rester bloqué sur une mauvaise solution. D'un autre côté, les actions de type M1 sont aussi importantes dans le sens où elles autorisent un réarrangement parmi les solutions en gardant pratiquement le même coût.

De plus, nous pouvons dire que toutes les actions sont réversibles. En effet, l'inverse d'une action M1 reste une action de type M1, alors que l'inverse d'une action de type M2 est une action de type M3 et vice versa.

5.6.3 La fonction objectif

Soit n le nombre de vecteurs de test possible, et m le nombre d'éléments d'un circuit.

Rappelons que pour la compaction des vecteurs de test, le but premier est d'obtenir une solution qui comporte un nombre minimale de vecteurs mais qui de maximise la couverture de fautes. C'est pourquoi nous utilisons une fonction objectif définie comme:

Equation 5.1 $$Coût = Coût(vecteur\_test) - Coût(couverture)$$

Où $Coût(vecteur\_test)$ est défini comme:

Equation 5.2
$$Coût(vecteur\_test) = \sum_{i=0}^{n} Poids\_x[i] * x[i]$$
$$x[i] = 1 \; si \; x \in X$$
$$x[i] = 0 \; si \; x \notin X$$

Poids _ x[i] est défini comme le coût de chaque vecteur de test et *x[i]* un nombre booléen qui définit si le vecteur est utilisé ou pas dans la solution X. En d'autres termes, si $x[i] = 1$ alors le vecteur est appelé actif et fait partie de la solution. Dans ce cas, le coût du vecteur sera ajouté au coût des vecteurs de test. Par contre, si $x[i] = 0$ alors le vecteur est appelé inactif et ne sera pas pris en compte dans le calcul du coût des vecteurs.

En prenant la définition ci-dessus de *x[i]*, nous pouvons maintenant l'associer à la définition des actions faites dans le paragraphe 5.6.2. En effet, pour faire une action de type M3 (enlever le vecteur v_i de la solution X), nous devons mettre *x[i]* à zéro. De même, pour faire une action de type M2 (ajouter le vecteur v_i à la solution X), nous devons mettre *x[i]* à un. Et finalement dans le cas d'une action M1 sur v_i et v_j nous devons assigner *x[i]* égal à un (ajouter un vecteur v_i) et *x[j]* égal à zéro (enlever le vecteur v_j).

D'une manière similaire *Coût(couverture)* est défini par:

Equation 5.3
$$Coût(couverture) = \sum_{j=0}^{m} Poids\_faute[j] * Couverture\_faute[j]$$
$$0 \le Couverture\_faute[j] \le 1$$

avec

Equation 5.4

$$Couverture\_faute[j] = \max(Couverture\_faute[i][j] * x[i])$$
$$x[i] = 1 \; si \; x \in X$$
$$x[i] = 0 \; si \; x \notin X \quad i = 1, 2, \cdots, n$$

Où $Poids\_faute[j]$ est défini comme le coût de chaque faute, et $Couverture\_faute[j]$ la couverture maximale de fautes sur la faute j pour la solution actuelle de X.

5.6.4 Calendrier du recuit-simulé

L'évolution de la température avec le temps est de la forme $T_k = r * T_{k-1}, \quad k = 1, 2, \cdots$. Une valeur type de r est 0.80 (cette valeur de r a été déterminée après une série d'expérimentation). La température initiale T_0 est déterminée de telle sorte que, dans le pire des cas, nous devrions avoir $\text{Prob}(\Delta E) = \exp(-\Delta(E)_{pire\_cas} / T_0) \approx 1$. Ainsi, il existe une probabilité raisonnable d'acceptation à haute température. T_0 doit donc être choisie de la façon suivante:

Equation 5.1

$$T_0 = -\Delta(E)_{pire\_cas} / \ln(\text{Prob}(\Delta E))$$
$$= -\Delta(E)_{pire\_cas} / \ln(\approx 1)$$

L'algorithme peut démarrer avec n'importe quelle condition initiale. La configuration initiale par défaut est une solution vide ne possédant aucun vecteur actif. Puis, pour chaque température, il faut appliquer un nombre suffisant d'actions, de façon à ce qu'il y ait eu N mouvements vers des états d'énergie plus bas (avec réduction du coût), ou

bien de façon à ce que le nombre total d'actions soit supérieur à $2N$, où $N = \alpha * n$ et α une constante spécifiée par l'utilisateur.

Le processus du recuit-simulé est terminé quand, à une certaine température, plus aucun mouvement n'est possible (Figure 5.3 et Figure 5.4).

5.6.4.1 Le calcul de $\Delta(E)_{pire\_cas}$

$\Delta(E)_{pire\_cas} = E_{la\_plus\_élevée} - E_{la\_plus\_basse}$ où :

- la plus basse énergie correspond au meilleur scénario: un seul vecteur de test est nécessaire pour tester toutes les fautes,

- la plus haute énergie correspond au pire cas: tous les vecteurs de test sont utilisés et la couverture de fautes reste ''insuffisante''.

Ainsi le calcul de $\Delta(E)_{pire\_cas}$ dépend du coût des vecteurs de test et du coût des fautes. En effet, dans le meilleur des cas, un seul vecteur est nécessaire à la solution pour tester toutes les fautes et la couverture de fautes obtenue pour chaque faute j est maximale. Ainsi: $Couverture\_faute[j] = \max(Couverture\_faute[i][j]) = 1$. Dans ce cas, l'énergie $E = E_{la\_plus\_basse}$ et

$$Coût = Coût(Vecteur\_test) - Coût(Couverture\_faute)$$

Equation 5.1

$$= \min(Poids\_x[i]) - \sum_{j=0}^{m} Poids\_faute[j] * Couverture\_faute[j]$$

$$\approx -\sum_{j=0}^{m} Poids\_faute[j]$$

Où $\min(Poids\_x[i])$ définit le plus petit poids de tous les vecteurs de test. Pour des raisons de simplification, $\min(Poids\_x[i])$ peut être approximé par zéro.

De la même manière, dans le pire des cas, la plupart des vecteurs de test sont utilisés pour la solution et la couverture de fautes reste à son minimum. Ainsi: $Couverture\_faute[j] = \min(Couverture\_faute[i][j]) = 0$. Dans ce cas: $E = E_{la\_plus\_élevée}$ et

$$Coût = Coût(Vecteur\_test) - Coût(couverture\_faute)$$

Equation 5.2

$$= \sum_{i=0}^{n} Poids\_x[i] - \sum_{j=0}^{m} Poids\_fault[j] * Couverture\_faute[j]$$

$$\approx \sum_{i=0}^{n} Poids\_x[i]$$

De l'Equation 5.1 et de l'Equation 5.2, on peut déduire que:

$$\Delta(E)_{pire\_cas} = E_{la\_plus\_élevée} - E_{la\_plus\_basse}$$

Equation 5.3

$$= \sum_{i=0}^{n} Poids\_x[i] + \sum_{j=0}^{m} Poids\_fault[j]$$

Calendrier du recuit-simulé
Définir *alpha*

Calculer la *température _initial*
Initialiser *température* à *température _initial;*
Boucle
    Initialiser le *numéro _ succès* à zéro
    Pour numéro _d'essai = 0; numéro _d'essai < *(2*alpha*n)*
        Choisir le vecteur à ajouter
        Choisir le vecteur à enlever
        Choisir l'*action* (ajoute, enlève, remplace)
        Calcule le coût de l'*action*
        Consulter l'oracle (fonction Metropolis)
        Si l'*action* est approuvée par la fonction de Metropolis
            Exécuter l'*action* et modifie la solution
            Incrémenter le *nombre _succès*
        Si le *nombre _succès* est supérieur à (*alpha*N*)
            Sortir de la boucle
    Si le *nombre _succès* est zéro
        Sortir (aucune amélioration n'est possible)
    Réduire la *température*

Figure 5.3: Calendrier du recuit-simulé

Fonction Metropolis
Calculer *delta _coût = dernier _coût – cou _précèdent*
Si *delta _coût* < 0
    Accepter l'*action*

Sinon

    Accepter l'*action* avec une probabilité de exp(- *delta -coût t/ température*)

Figure 5.4: Fonction de Metropolis

5.6.5 Améliorations possibles

Plusieurs améliorations peuvent être apportées à l'algorithme général dans le but d'augmenter la vitesse de convergence. En fait, trois modifications importantes sont implantées:

- des solutions tampons,
- un facteur de compression des vecteurs de test,
- un calcul par incrément de la fonction coût.

5.6.5.1 Les solutions tampons

L'idée consiste à ajouter un tampon à l'algorithme original de recuit-simulé. Ce tampon conservera les meilleures β solutions (β est une constante définie par l'utilisateur). A chaque fois qu'une nouvelle solution est ajoutée au tampon, un algorithme de tri enlèvera en échange la solution la plus coûteuse du tampon.

Pour chaque nouvelle action, on vérifie d'abord si la solution proposée n'est pas dans le tampon; si oui (la solution a déjà été traitée), le processus est stoppé et une nouvelle solution est choisie. Dans le cas contraire (la nouvelle solution n'est pas dans le tampon), le coût de la nouvelle solution est calculé et, si la solution est acceptée par l'algorithme de Metropolis, alors celle-ci est ajoutée au tampon.

L'avantage majeur de l'utilisation des tampons vient du fait que la recherche dans le tampon, avant d'accepter la solution, élimine les problèmes d'oscillation (quand il y a

deux solutions de même coût ou de coût sensiblement égal). En effet, seules les nouvelles solutions qui améliorent les précédentes sont ajoutées au tampon.

De plus, le fait de maintenir les meilleures β solutions constitue un avantage supplémentaire, car l'utilisateur peut alors, à n'importe quel moment, arrêter l'algorithme et prendre les solutions obtenues. Par ailleurs, l'utilisateur aura le choix de sélectionner la solution qui lui convient le mieux (comme par exemple, une solution plus coûteuse mais présentant une meilleure couverture de fautes).

Après l'ajout de ce mécanisme de tampon, la formulation de l'algorithme devient:

Calendrier du recuit-simulé
Définir *alpha*

Calculer de la *température _initial*
Initialiser *température* à *température _initial;*
Boucle
    Initialise le *numéro _ succès* à zéro
    Pour numéro _d'essai = 0; numéro _d'essai < *(2*alpha*n)*
        Choisir le vecteur à ajouter
        Choisir le vecteur à enlever
        Choisir l'*action* (ajoute, enlève, remplace)
        Si l'*action* proposée existe dans le tampon
            Continue (retourne au début de la boucle)
        Calculer le coût de l'*action*
        Consulter l'oracle (fonction Metropolis)
        Si l'*action* est approuvée par la fonction de Metropolis
            Exécuter l'*action* et modifie la solution
            Incrémenter le *nombre _succès*
            Ajoute la solution au tampon
        Si le *nombre _succès* est supérieur à (*alpha*N*)
            Sortir de la boucle
    Si le *nombre _succès* est zéro
        Sortir (aucune amélioration n'est possible)
    Réduire la *température*

Figure 5.5: Calendrier du recuit-simulé modifié

Ajouter une solution au tampon
    Ajouter la solution à la fin du tampon
    Trier les solutions par coût croissant
    Enlèver la solution la plus coûteuse

Figure 5.6: Ajout d'une solution au "tampon de solutions"

5.6.5.2 Facteur de compression des vecteurs de test

Le facteur de compression des vecteurs de test permet à l'utilisateur d'accroître ou de réduire leur coût, tout en respectant celui des fautes. En effet, le facteur de compression va ajouter un nouveau degré de liberté à l'algorithme et sera considéré comme un facteur de pondération entre le coût de la couverture de fautes et celui des vecteurs de test. Afin de prendre en compte ce facteur de compression des vecteurs de test, l'Equation 5.1 est modifiée comme suit:

Equation 5.1

$$Coût = Coût(vecteur\_test) * facteur\_compression - Coût(couverture\_faute)$$

La compression proposée offre un avantage important dans le cas où un petit nombre de vecteurs de test est demandé pour différencier les circuits sains des circuits fautifs. Dans ce cas, le facteur de compression sera élevé, ce qui induit une augmentation du coût des vecteurs de test tout en respectant le coût associé à la couverture de fautes, ce dernier forçant l'algorithme d'optimisation à choisir moins de vecteurs de test. De même, si on diminue le facteur de compression, le coût des vecteurs de test diminuera en respectant celui associé à la couverture de fautes. Là on utilisera plus de vecteurs de test dans la solution. Une diminution du facteur de compression devient très utile quand une couverture de fautes élevée est requise, notamment dans le cas des circuits VLSI.

5.6.5.3 Calcul par incrément de la fonction objectif

L'idée est de pouvoir mettre à jour le coût des vecteurs de test ainsi que celui de la couverture de fautes. Soit X la solution précédente et Y la nouvelle solution, le coût de Y est d'abord calculé en mettant à jour le coût des vecteurs de test puis en mettant à jour le coût de la couverture de fautes.

Pour une action de type M3, X est mis à jour en ajoutant un nouveau vecteur v_t. En premier lieu, on rafraîchit le coût des vecteurs de test en ajoutant le coût de v_t (Equation 5.2). Puis, pour chaque faute, si la couverture de fautes trouvée avec le nouveau vecteur v_t est supérieure à la couverture de fautes générée par X, alors la couverture de fautes Y est donnée par la couverture de fautes de v_t. Sinon, la couverture de fautes reste inchangée.

Pour une action de type M2, X est mis à jour en enlevant un vecteur v_t. Le coût du vecteur de test est d'abord calculé en soustrayant le coût de v_t. Puis, pour chaque faute, si la couverture de fautes du vecteur à enlever est égale à la couverture de fautes de la solution X, alors la couverture de fautes est recalculée en utilisant l'équation 5. Sinon, la couverture de fautes est laissée inchangée.

Pour une action de type M1, X est mis à jour en échangeant les deux vecteurs et on utilise alors les deux types de mises à jour présentées précédemment.

Cependant, notons que le processus est sujet à une accumulation d'erreurs. Ainsi, pour chaque nouvelle température, le coût précédent est supprimé et recalculé.

5.7 Résultats expérimentaux

Deux circuits de référence sont simulés: le premier est un simple amplificateur de tension à 9 transistors et l'autre est un convertisseur numérique analogique (CNA) 4 bits comportant 153 transistors. Les modèles de transistors MOSFET utilisés sont de niveau 3, 28 et 49.

Dans les deux expériences, nous considérons le facteur de compression comme une variable. Les Figure 5.7 et Figure 5.8 montrent le nombre de vecteurs de test ainsi que le fausse-acceptation en fonction du facteur de compression. On retrouve bien le fait que le nombre de vecteurs de test et la couverture de fautes (dans ce cas (1-fausse-acceptation)) diminuent avec le facteur de compression. En fait, le choix du bon facteur de compression dépend de l'application, du coût du test et de la couverture de fautes désirés.

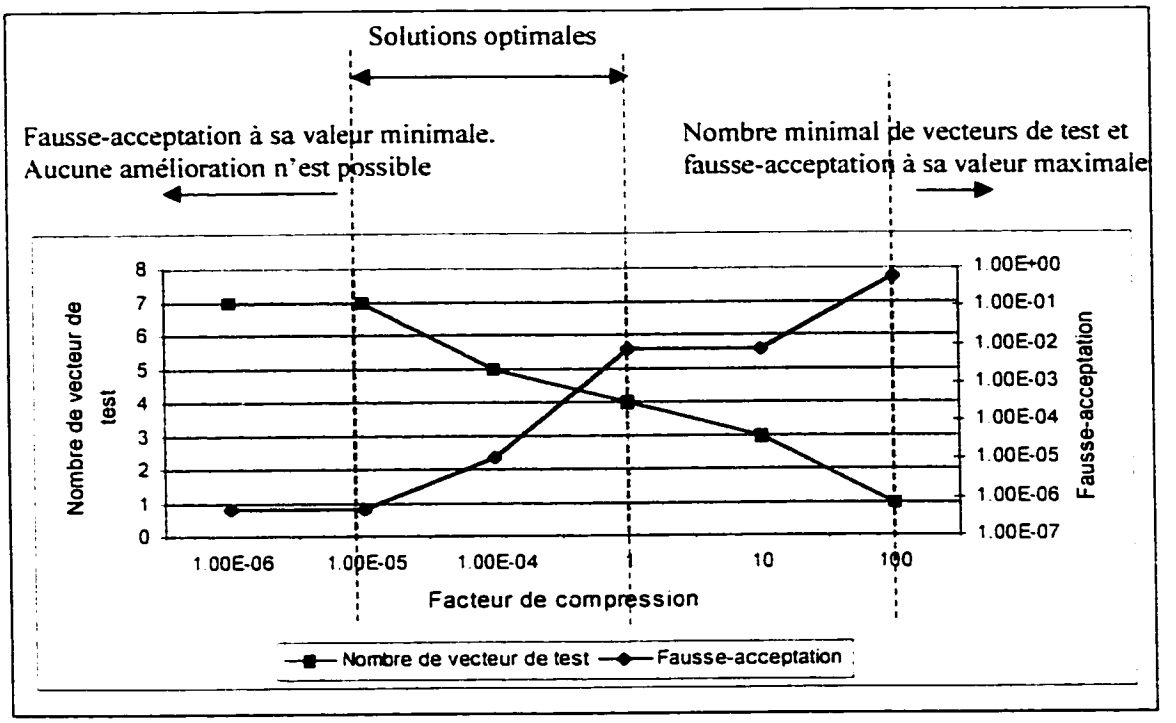

Figure 5.7: Amplificateur de tension(Fausse-acceptation et nombre de vecteurs de test en fonction du facteur de compression)

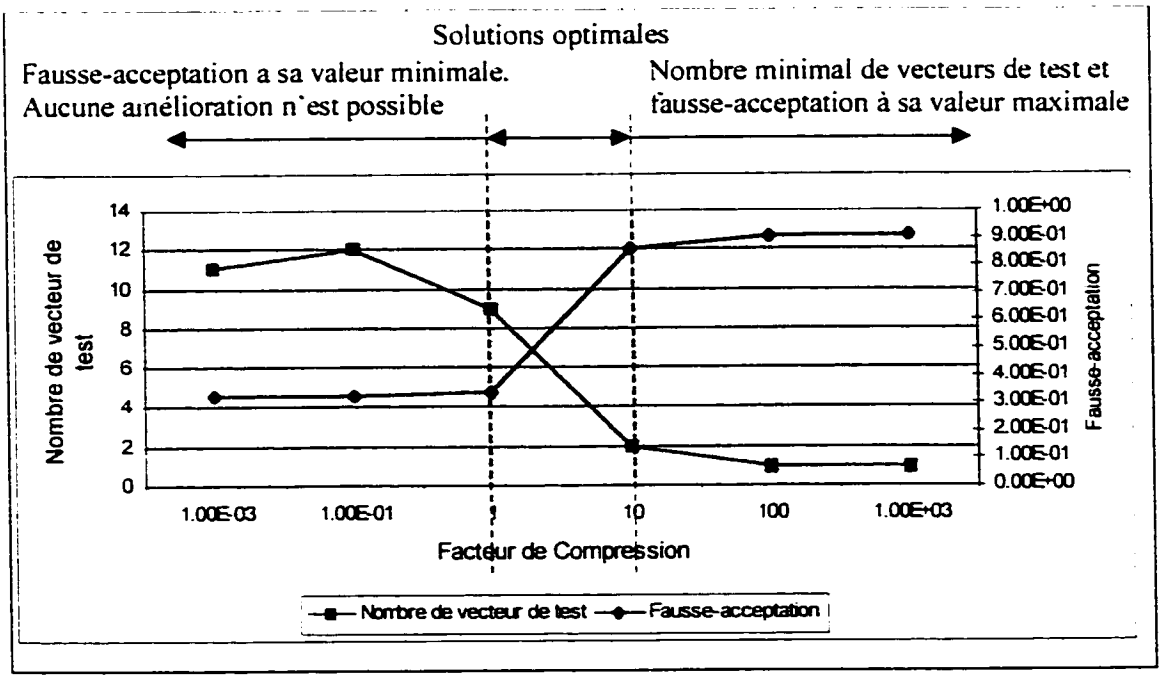

Figure 5.8: Convertisseur numérique analogue à 4 bits (Fausse-acceptation et nombre de vecteurs de test en fonction du facteur de compression)

5.8 Remarques et conclusion

Le problème de compaction des vecteurs de test est connu comme étant un problème du type NP-complet. Dans ce document, nous avons présenté une formulation généralisée du problème qui prend en considération les multiples modes de test ainsi que la définition de la couverture de fautes tous deux étant traités simultanément.

Nous avons également présenté un algorithme résolvant le problème de la formulation généralisée. Cet algorithme est capable de minimiser le coût des vecteurs de test et de trouver la meilleure combinaison des vecteurs de test qui maximise la couverture de fautes. Les résultats expérimentaux confirment que cette méthode donne de bons résultats aussi bien au niveau du coût des vecteurs de test qu'au niveau de la couverture de fautes.

Les solutions de quelques circuits de référence ont montré des améliorations substantielles dans le nombre de vecteurs de test requis alors que la couverture de fautes reste à son niveau optimal. Pour le simple amplificateur de tension présenté précédemment, une amélioration de 50% a pu être apportée par rapport à une solution donnée par une méthode conventionnelle où, pour chaque faute, le vecteur de test est le vecteur qui maximise la couverture de fautes indépendamment des autres fautes.

CHAPITRE 6

INSERTION DE POINTS DE TEST

6.1 Introduction

Dans les chapitres précédents, une technique a été présentée: celle-ci permet la sélection des vecteurs de test et la quantification de la couverture de fautes sur n'importe quel nœud/branche du circuit. Or, l'augmentation de la densité des circuits VLSI actuels et la limitation des ports rendent l'accès à des nœuds internes très difficile et diminuent donc la possibilité d'isoler les composantes dans le but de test, ce qui se traduit par une faible couverture de fautes. L'incorporation d'une philosophie de test au début de la conception ("Design For Testability, DTF") [17], [35], s'avère donc une étape importante durant le développement d'un circuit intégré.

En effet, l'incorporation des techniques "DFT" permet d'augmenter de façon considérable la contrôlabilité, l'observabilité et, par conséquent, la testabilité et la diagnosticabilité du circuit.

Or, augmenter la contrôlabilité implique trois contraintes:

(i) trouver un bon point pour l'insertion d'une sonde qui permet d'augmenter la contrôlabilité.

(ii) Isoler le nœud sélectionné de l'étage précédent afin d'assurer une bonne contrôlabilité du bloc d'intérêt,

(iii) minimiser l'effet de charge dû à l'ajout de la sonde.

Par contre, l'augmentation de l'observabilité n'implique que deux contraintes:

(i) trouver un bon point pour l'insertion d'une sonde de test qui permet d'augmenter l'observabilité,

(ii) minimiser l'effet de charge dû à l'ajout de la sonde.

Dans les deux cas, il faut s'assurer que l'insertion d'une sonde ne dégrade pas les performances du circuit. En effet, il est connu que pour les circuits digitaux, on évite d'ajouter une sonde sur le chemin critique, même si des sondes spéciales avec charge minimale sont utilisées [38]. Les mêmes précautions doivent être considérées pour les circuits analogiques où une charge additionnelle sur les nœuds internes peut dégrader les performances du circuit analogique d'une manière significative.

6.2 Objectif

Notre objectif est de développer un algorithme d'insertion de points de test (PDT) dans le but d'augmenter l'observabilité et donc la couverture des fautes dans les circuits analogiques. Le choix du point de test à ajouter est déterminé par:

(i) la couverture de fautes additionnelle,

(ii) l'analyse de l'effet de charge sur les performances du circuit dû à l'ajout de la sonde,

(iii) le coût additionnel dû à l'insertion du point de test,

(iv) le coût des vecteurs de test supplémentaires.

Une approche qui soulève tous ces problèmes simultanément peut devenir très coûteuse car le nombre de combinaisons possibles des nœuds de test et des vecteurs de test peut devenir très élevé. Une simplification du problème et une utilisation des approches heuristiques sont nécessaires.

6.3 Approche

Vue la complexité du problème à résoudre, il est nécessaire de le diviser en des problèmes plus simples et de les résoudre d'une manière indépendante. En effet, il est possible de diviser le problème en quatre sous-problèmes, soient:

(i) La construction de la liste des fautes,

(ii) La construction de la liste des nœuds de test,

(iii) Le calcul de la couverture de fautes sur chacun des nœuds de test,

(iv) La sélection du point de test en utilisant un algorithme heuristique d'optimisation.

Figure 6.1 résume l'approche.

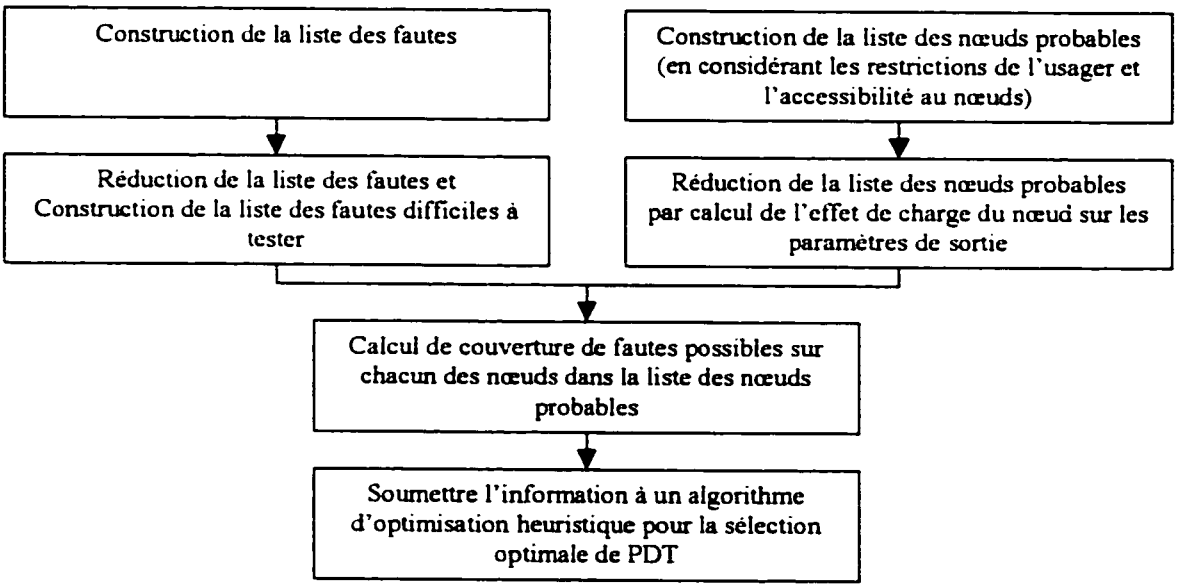

Figure 6.1: Approche générale de sélection de points de test

Dans les paragraphes suivants, nous présenterons chacune de ces quatre étapes. Comme la premiere et la troisième étape découlent directement des études présentées dans les chapitres 3 et 4, seules les approches utilisées pour la construction des nœuds probables et l'algorithme d'optimisation seront présentées en détail.

6.3.1 Construction de la liste des fautes

Une première étape vers l'automatisation de l'insertion de points de test sera d'identifier la liste des fautes difficiles à tester. Or, dans les chapitres 3 et 4, nous avons présenté une méthode de test qui nous a permis d'associer une couverture de panne à chaque faute lorsqu'un certain nombre de vecteurs de test sont appliqués au circuit nominal (sans transformation). Toutes les fautes pour lesquelles la couverture n'est pas suffisamment élevée doivent être inclues dans la liste des fautes difficiles. Cette étape est essentielle, car la majorité des fautes sont faciles à tester et aucun effort supplémentaire n'est nécessaire pour les détecter.

6.3.2 Construction de la liste des nœuds probables.

L'analyse de l'effet de l'ajout de la sonde sur les performances du circuit constitue la deuxième étape. Celle-ci permet de construire la liste des nœuds probables. En effet, initialement, tout nœud dans le circuit peut être considéré comme un nœud probable pour l'insertion d'une sonde. Or, une sonde (modélisée par une charge capacitive), insérée dans le circuit peut détériorer de manière significative la fonctionnalité du circuit. Une technique doit être développée afin d'éliminer à l'avance tous les nœuds "sensibles" de la liste des nœuds probables.

Or, les questions qui se posent maintenant sont: comment identifier ces nœuds critiques? Comment automatiser le processus de réduction de point de test? Finalement, il faut déterminer si un nœud doit être enlevé de la liste des nœuds probables.

Deux solutions se présentent:

- une basée sur une analyse sérielle de l'effet de charge sur les paramètres de sortie,

- une deuxième, basée sur une analyse parallèle de l'effet de charge.

Une approche sérielle nécessite l'insertion d'une charge dans le circuit sur chacun des nœuds suivie d'une simulation et d'une analyse de l'effet de la charge sur les performances du circuit. Le coût associé à cette approche 'sérielle' est exorbitant.

Or, la méthode de calcul de sensibilité par la technique des circuits adjoints permet un calcul parallèle de la sensibilité de n'importe quel paramètre de sortie par rapport à des composantes non-existantes dans le circuit et, en particulier, par rapport à des capacités non-existantes [22]. Si la sensibilité du paramètre de sortie par rapport à l'ajout de la charge est trop élevée, alors le nœud est enlevé de la liste des nœuds de test.

Pour des raisons de vitesse de calcul et d'efficacité, la deuxième approche sera utilisée.

6.3.2.1 Approche

Soit le nœud i, avec une capacité parasite C_i et une valeur nominale de "zéro". La sensibilité du paramètre de sortie W_k par rapport à C_i est définie par

Equation 6.1 $$S_{C_i}^{W_k} = \left.\frac{dW_k}{dC_i}\right|_{C_i=0}$$

et la contribution d'une charge C_i sur la dégradation de performance du paramètre W_k est obtenue par:

Equation 6.2 $$\Delta W_k = S_{C_i}^{W_k} \Delta C_i .$$

Où $S_{C_i}^{W_k}$ est obtenue par la méthode des circuits adjoints présentée dans le chapitre 2, Table 1. Notons que le calcul de $S_{C_i}^{W_k}$ ne requiert que les tensions aux bornes de la capacité et non la valeur de la capacité même.

De l'équation 6.2, on peut conclure que la dégradation des performances est proportionnelle à la sensibilité $S_{C_i}^{W_k}$. La classification des nœuds en fonction de leurs effets sur le/les paramètre(s) de sortie est donc possible.

Alors que la modélisation d'une sonde de test par une charge capacitive n'est pas nouvelle, la combinaison d'un modèle capacitif avec le calcul des sensibilités dans le but d'insertion de PDT est une innovation.

6.3.2.2 Résultats préliminaires

L'approche proposée a été appliquée à l'amplificateur opérationnel montré à la Figure 2.3. Tout d'abord, la sensibilité du gain de l'amplificateur opérationnel par rapport à des

charges capacitives sur tous les nœuds internes a été obtenue. La Figure 6.2.a) montre la sensibilité du gain par rapport à une présence d'une charge capacitive "nulle" sur un nœud interne. Cette sensibilité prédit une baisse de gain pour les fréquences inférieures à 180MHz et une augmentation du gain pour les fréquences supérieures.

Pour valider l'approche, une capacité a été insérée sur ce nœud et le gain du circuit a été comparé au gain du circuit nominal (Figure 6.2.b). Le comportement du gain corrèle bien avec les résultats prédits par la sensibilité.

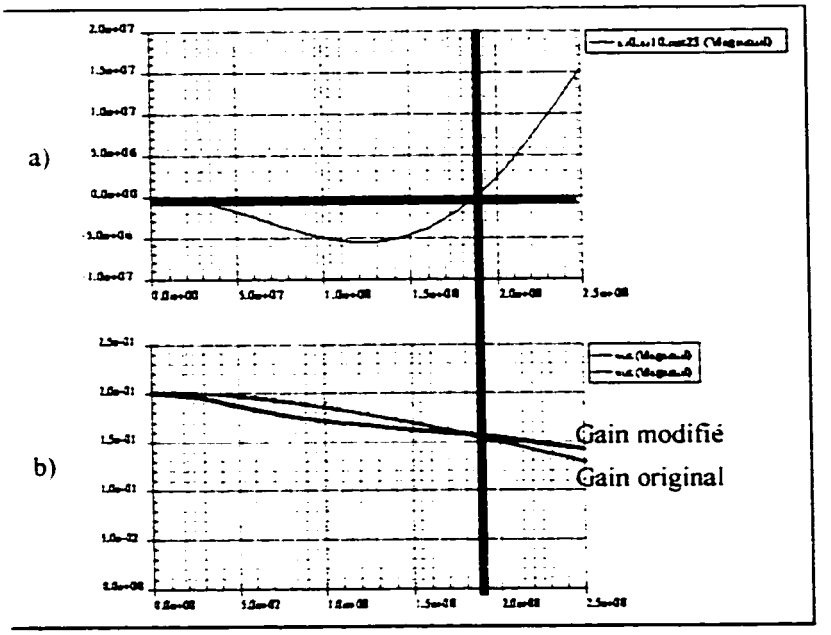

Figure 6.2 : Effet de charge sur le gain du circuit a) La sensibilité du gain par rapport à la présence d'une charge capacitive sur le nœud interne de l'amplificateur et b) l'effet de charge sur le gain de circuit obtenu par simulation.

6.3.3 Calcul de la couverture de fautes

Une fois la liste des fautes difficiles et la liste des nœuds probables obtenues, la probabilité de détection de toutes les fautes difficiles à tester, à tous les nœuds de test probables, est calculée en utilisant l'approche de test présentée dans les chapitres précédents.

6.3.4 Sélection de points de test

Cette étape est relativement coûteuse car une simulation du circuit adjoint est nécessaire pour chacune des PDT dans la liste des nœuds probables. En effet, après que la liste des fautes difficiles à tester soit obtenue, la liste des nœuds probables est générée et la probabilité de détection de chaque composante sur chacun des nœuds probables est calculée. Alors, à ce moment, les données obtenues peuvent être soumises à un algorithme d'optimisation heuristique pour la sélection optimale des PDT. L'algorithme aura comme objectif de minimiser le nombre de points de test à insérer, tout en gardant une couverture de fautes aussi élevée que possible.

Vu la complexité et le nombre de combinaisons possibles, l'emplacement optimal du point de test ne peut être résolu qu'en utilisant des heuristiques. L'algorithme du recuit-simulé, présenté dans le chapitre précédent, a été modifié pour prendre en considération ces contraintes supplémentaires. Les modifications se résument en deux points:

1. Modification de la structure de donnée pour prendre en considération la présence de plus d'un seul paramètre de sortie.

2. Remplacer la boucle d'itération sur les vecteurs de test par deux boucles d'itération : la boucle maître, qui itère sur les PDT dans la liste des nœuds probables pour trouver la meilleure combinaison de PDT, et la boucle interne, qui sélectionne la meilleure solution des vecteurs pour la solution des nœuds choisie par la boucle maître.

6.3.4.1 Implémentation

La sélection d'un point de test implique l'utilisation d'un algorithme d'optimisation qui prend en considération la couverture de fautes, la dégradation des performances, le coût du point de test ajouté et le coût du vecteur de test.

Il est évident que les étapes nécessaires pour atteindre une solution consistent à obtenir une combinaison de nœud Y et une combinaison de vecteurs X, calculer le coût de la solution, calculer la couverture de fautes correspondante, et dépendamment du coût de la solution et de la température du système, accepter ou rejeter la solution proposée.

L'espace de solution, la fonction coût, la structure de donnée, ainsi que l'algorithme d'optimisation sont présentés dans les paragraphes suivants.

6.3.4.1.1 L'espace des solutions

L'espace des solutions est constitué par l'ensemble des combinaisons possibles entre les nœuds de test et les vecteurs de test. L'algorithme peut démarrer avec n'importe quelle condition initiale (même une solution vide) et être itéré jusqu'à ce que la meilleure combinaison soit trouvée.

6.3.4.1.2 La fonction objectif

Rappelons-nous que pour l'insertion de PDT, le but premier est d'obtenir la meilleure solution minimale mais aussi de maximiser la couverture de fautes. C'est pourquoi nous utilisons une fonction objectif définie comme suit:

Equation 6.3

$$\begin{aligned} Coût = {} & Coût(degradation\_performances) \\ & + Coût(point\_de\_test) \\ & + Coût(vecteur\_test) \\ & - Coût(couverture) \end{aligned}$$

6.3.4.1.3 La structure de donnée

La structure de donnée présentée dans le chapitre 5, figure 2 a été modifiée pour considérer la présence de plus d'un seul paramètre de sortie. En effet, la structure de données tridimensionnelles (Modes d'opération, vecteurs de test pour chaque mode d'opération et la couverture de fautes associée) est insuffisante. Une dimension supplémentaire est nécessaire pour considérer les PDT dans la liste des nœuds probables (Figure 6.3).

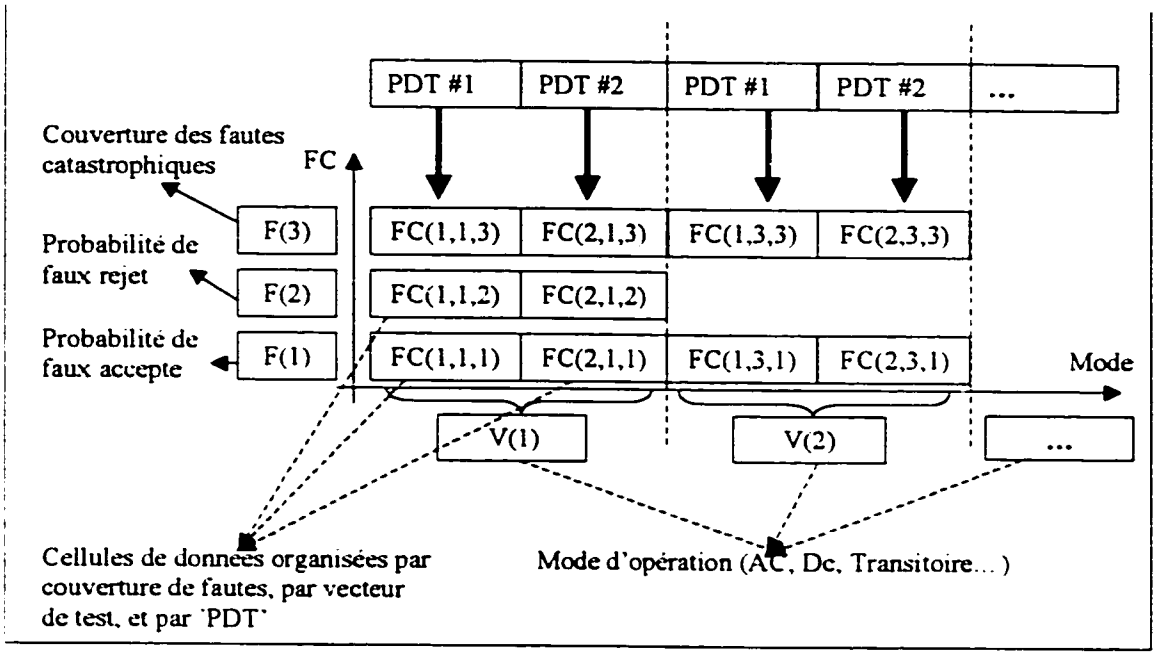

Figure 6.3: Représentation interne des données

6.3.4.1.4 Algorithme

L'algorithme est constitué de deux boucles imbriquées l'une dans l'autre: la boucle maître et la boucle esclave. La boucle maître est responsable de la sélection d'une solution des nœuds de test, alors que la boucle esclave est responsable de la sélection d'une solution des vecteurs de test. La solution des nœuds de test est identifiée par la liste des nœuds actifs Y et la solution des vecteurs de test est identifiée par la liste des vecteurs de test actifs X.

Tout d'abord, la boucle maître modifie la liste des nœuds actifs en ajoutant, en enlevant, ou en remplaçant un nœud dans la liste des nœuds actifs Y. Par la suite, pour chaque solution des nœuds Y, la boucle esclave est appelée pour la sélection des vecteurs de test X qui minimise la fonction coût définie par l'Equation 6.3. L'algorithme de sélection des vecteurs de test est essentiellement le même algorithme que celui décrit dans le chapitre 5. L'exception est que la couverture de test pour la faute j, pour la solution courante des nœuds et des vecteurs de test (équation 5.6), est redéfinie comme:

Equation 6.4

$$Couverture\_faute[j] = \max(Couverture\_faute[i][j][k] * x[i] * y[k])$$
$$x[i] = 1 \; si \; x \in X$$
$$x[i] = 0 \; si \; x \notin X \quad i = 1, 2, \cdots, n$$
$$y[k] = 1 \; si \; y \in Y$$
$$y[k] = 0 \; si \; y \notin Y \quad k = 1, 2, \cdots, p$$

n est le nombre de vecteurs de test possibles, m le nombre d'éléments dans le circuit, et p le nombre de nœuds dans la liste des nœuds probables.

L'algorithme peut démarrer avec n'importe quelle condition initiale. La configuration initiale par défaut est une solution vide ne possédant aucun nœud et vecteur actifs. Puis, pour chaque *température_nœuds*, on doit appliquer un nombre suffisant d'actions sur la liste des nœuds actifs de façon à ce qu'il y ait eu *Nnœud* mouvements vers des états d'énergie plus basse (avec réduction du coût) ou bien de façon à ce que le nombre total d'actions soit supérieur à $2Nnœud$, où $Nnœud = \alpha * p$ et α une constante spécifiée par l'utilisateur (Figure 6.4).

Notons que pour chaque solution des nœuds, la solution des vecteurs actifs X est initialisée à une solution vide et la température initiale des vecteurs T_0 est recalculée de telle sorte que, dans le pire des cas, il existe une probabilité raisonnable d'acceptation de n'importe quelle solution des vecteurs X ,même la pire solution (voir paragraphe 5.6.4).

Ceci dit, pour chaque *température_vecteur*, un nombre suffisant d'actions est appliqué sur la liste des vecteurs actifs de façon à ce qu'il y ait eu *Nvect* mouvements vers des états d'énergie plus bas (avec réduction du coût), ou bien de façon à ce que le nombre total d'actions soit supérieur à 2*Nvect*, où $Nvect = \alpha * n$ et α une constante spécifiée par l'utilisateur (Figure 6.5).

Le processus du recuit-simulé est terminé quand, à une certaine température, plus aucun mouvement n'est possible.

Définir *alpha*

Calculer la *température_initial_Nœud*
Initialiser *température_Nœud* à *température_initia_Nœud;*

Boucle
Initialiser le *numéro_succès* à zéro
Pour numéro_d'essai = 0; numéro_d'essai < *(2*alpha*n)*
Choisir le nœud à ajouter
Choisir le nœud à enlever
Choisir l'*action* (ajoute, enlève, remplace le nœud)
Calculer le coût de l'*action*
Calculer de couverture maximale obtenue pour cette combinaison de nœud (faire appel à la boucle esclave pour la sélection des vecteurs de test)
Consulter l'oracle (fonction Metropolis)
Si l'*action* est approuvée par la fonction de Metropolis
Exécuter l'*action* et modifie la solution
Incrémenter le *nombre_succès*
Si le *nombre_succès* est supérieur à (*alpha*N*)
Sortir de la boucle
Si le *nombre_succès* est zéro
Sortir (aucune amélioration n'est possible)
Réduire la *température_Nœud*

Figure 6.4: Calendrier du recuit-simulé de la boucle maîtresse

(sélection du nœuds de test)

Définir *alpha*

Calculer la *température_ initial_ Vecteur*
Initialise *température_ Vecteur* à *température_ initial_ Vecteur;*

Boucle
    Initialiser le *numéro _ succès* à zéro
    Pour numéro _d'essai = 0; numéro _d'essai < *(2*alpha*n)*
        Choisir le vecteur à ajouter
        Choisir le vecteur à enlever
        Choisir l'*action* (ajoute, enlève, remplace le vecteur)
        Calculer le coût de l'*action*
            Calculer de couverture maximale obtenue pour cette combinaison de vecteurs (Max (couverture de chaque nœud actif))
        Consulter l'oracle (fonction Metropolis)
        Si l'*action* est approuvée par la fonction de Metropolis
            Exécuter l'*action* et modifie la solution
            Incrémenter le *nombre _succès*
        Si le *nombre _succès* est supérieur à (*alpha*N*)
            Sortir de la boucle
    Si le *nombre _succès* est zéro
        Sortir (aucune amélioration n'est possible)
    Réduire la *température_ Vecteur*

Figure 6.5: Calendrier du recuit-simulé de la boucle esclave (sélection de vecteurs de test)

6.4 Conclusion

Ce chapitre représente une étape exploratoire vers le développement d'un algorithme d'insertion de points de test. Bien que les recherches et les résultats préliminaires soient encourageants, une investigation plus approfondie et une implémentation complète de l'algorithme proposé sont nécessaires.

CHAPITRE 7

CONCLUSION GÉNÉRALE

Au cours de cette thèse, nous avons présenté une méthodologie complète pour le test des circuits analogiques. Celle-ci, qui est basée sur le calcul de la sensibilité, permet dans une première étape, la génération des vecteurs de test pour les fautes douces et catastrophiques. Dans une deuxième, un algorithme d'optimisation (utilisant la technique du recuit-simulé réalise la compaction de ces vecteurs à un nombre minimal tout en gardant un niveau de testabilité élevé. Enfin, une méthodologie pour l'insertion de points de test est présentée en dernière étape.

L'exploitation d'analyses traitant des fautes sur les circuits analogiques a mis en évidence la difficulté du test. En effet, elle est due à la nature continue des fautes analogiques et à l'inexistence d'un modèle de fautes robuste semblable à celui des circuits numériques ("Stuck-at"). Deux classes de fautes peuvent être distinguées: les fautes douces ou fautes à petite déviation de la composante de sa valeur nominale et les fautes catastrophiques.

Une solution remédiant aux problèmes de test des circuits analogiques est présentée dans ce document. Cette solution de test, proposée pour les 2 types de fautes possibles, est développée respectivement dans les chapitre 3 et 4. Elle consiste en un modèle de fautes, une méthode de simulation ainsi qu'une technique de génération de vecteurs de test.

Le modèle de fautes structurelles est associée à une approche parallèle de simulation de fautes. A la différence des méthodes proposées jusqu'à présent, celle-ci a la particularité de ne pas impliquer le simulateur dans la boucle de simulations. En effet, la variation du procédé et la sensibilité sont utilisées pour la génération d'un modèle statistique du circuit nominal et pour la génération des modèles statistiques des circuits fautifs (un modèle par faute).

Une fois que les modèles statistiques du circuit nominal et des circuits fautifs sont obtenus, il est aisé d'obtenir les probabilités de faux-rejet (rejeter un circuit sain) et de fausse-acceptation (accepter un circuit fautif). Ces données sont utilisés comme critère de décision pour la génération de vecteurs de test.

Il semble important de rappeler ici que la sensibilité, utilisée pour la génération des modèles statistiques, est le maillon central autour duquel s'articule cette thèse. Pourquoi la sensibilité?

La sensibilité peut être considérée comme une relation de premier ordre entre les variations des composantes et les variations des paramètres de sortie du circuit.

Dans le deuxième chapitre, une approche automatique pour le calcul de la sensibilité et la génération des vecteurs de test a été introduite. Cette approche est basée sur la méthode des circuits adjoints. Le choix de la méthode adjointe est due à sa capacité de mesurer la sensibilité du paramètre de sortie par rapport à toutes les variations possibles des composantes en deux simulations seulement: une pour le circuit initial et l'autre pour

le circuit adjoint. De plus, la méthode adjointe permet non seulement le calcul de la sensibilité dans le domaine fréquentiel mais aussi dans le domaine temporel et DC.

Les exemples proposés dans la thèse montrent la validité de la méthode. En effet, on a obtenu une bonne corrélation entre les résultats estimés par cette méthode d'une part, et ceux obtenus par simulation Hspice.

Par contre, il est évident que la génération de vecteurs de test est une méthode insuffisante par elle-même, à cause de la croissance rapide et continuelle du coût du test. Il faut donc établir une solution ayant pour objectif de minimiser le nombre de vecteurs de test tout en conservant une couverture de fautes aussi proche que possible de la couverture nominale. Pour remédier à ce problème, une méthode heuristique de compaction de vecteurs de test a été développée dans le chapitre 5. Les exemples mettent en valeur la validité de la méthode.

Il arrive cependant, que la couverture obtenue n'est pas suffisante. Une augmentation de la contrôlabilité et/ou de l'observabilité peut remédier à ce problème. Dans le chapitre 6, on propose une solution au problème de l'augmentation de l'observabilité par insertion de point de test. Des tests préliminaires ont montré la validité de l'approche, mais une implémentation complète est nécessaire.

Il est évident que les algorithmes sont encore dans leur phase initiale de développement et que plusieurs sous-programmes doivent être greffés au noyau. Pour nommer quelques unes des améliorations possibles, on peut mentionner:

1. L'augmentation de la précision du test en tenant compte de la non-linéarité des circuits. Pour ces circuits, une approche de calcul de sensibilité par la méthode des circuits adjoints conventionnelle, valide uniquement pour les circuits linéaires (ou linéarisés), ne sera plus suffisante et ne pourra fournir qu'une première approximation sur la sortie réelle du circuit.

2. Dans le choix des vecteurs de test, il est parfois nécessaire de préférer un vecteur avec couverture inférieure mais qui présente une meilleure tolérance à l'imprécision du générateur des vecteurs (amplitude du stimulus, fréquence du stimulus, etc.). Prenons par exemple un vecteur ayant une couverture maximal pour une fréquence donnée. Si ce vecteur connaît une dégradation importante de la couverture autour de cette fréquence, il est alors préférable de choisir une fréquence avec une couverture inférieure, mais garantie lorsque la fréquence varie (due à la tolérance sur les générateurs).

   Cette approche peut être facilement implémentée. En considérant la tolérance du générateur de fréquences (par exemple 1%), on obtient la plage de fréquence d'analyse (fréquence nominal $\pm$ 1%). Dans ce cas, la couverture est définie comme :

   Equation 7.1 $\textit{couverture} = \min(\text{couverture sur la plage})$

3. L'extension au test des circuits numériques et mixtes où une majorité des circuits retournés comme défectueux, présente des fautes douces dans la partie

numérique. Donc, ce qui était considéré comme problème typique aux circuits analogiques est devenu une nécessité pour les circuits numériques d'aujourd'hui

Sommairement nos contributions se résument à:

1. La proposition d'un modèle de fautes douces structurelles et d'un modèle de fautes catastrophiques basés respectivement sur les données statistiques de la composante défectueuse et sur la notion de *cause à effet*.

2. La génération parallèle d'un modèle statistique du circuit nominal et des modèles statistiques des circuits fautifs.

3. L'adaptation des notions statistiques (faux-rejet et fausse-acceptation) au problème de test des circuits analogiques.

4. Le développement d'un algorithme de sélection de vecteurs de test. Le choix des vecteurs de test est tel que la probabilité de commettre une fausse décision sur l'état du circuit (risque de Bayes) est minimale. Comme cette approche associe une couverture de fautes aux vecteurs choisis, une classification des vecteurs de test est maintenant possible.

5. Le développement d'un algorithme de compaction de vecteurs de test qui tient compte du coût des vecteurs de test et de la couverture de fautes.

6. La proposition d'une approche et d'un algorithme pour l'insertion de points de test.

RÉFÉRENCES

[1] SLAMANI M., and KAMINSKA B., Soft Large Deviation and Hard Fault Multi-frequency Analysis in Analog Circuits, IEEE Design and Test of Computers, 1995, pp. 70-80.

[2] SLAMANI M., and KAMINSKA B., "Multi-frequency Testability Analysis for Analog Circuit", IEEE Trans. on Circuit and Systems, Part II, February1996, pp. 134-139.

[3] BEN HAMIDA N. and KAMINSKA B., "Multiple Fault Analog Circuit Testing by Sensitivity Analysis", J. of Electronic Testing, JETTTA, Kluwer Academic Publication, No 4, Nov. 1993, pp. 331-343.

[4] SLAMANI M., and KAMINSKA B., "Analog Circuit Fault Diagnosis Based on Sensitivity Computation and Functional Testing", IEEE Design and Test of Computers, March 1992, pp. 30-39.

[5] SLAMANI M., and KAMINSKA B., "An Integrated Approach for Analog Circuit Testing with a Minimum Number of Detected Parameters", IEEE International Test Conference, Washington D.C., Oct. 1994, pp. 631-640

[6] BEN HAMIDA. NAIM, and KAMINSKA BOZENA., "Analog Circuit Testing Based on Sensitivity Computation", IEEE International Test Conference, Baltimore, Oct. 1993, pp. 652-661.

[7] SLAMANI M., and KAMINSKA B., "Testing Analog Circuit by Sensitivity Computation", European Design Automation Conference, EDAC'92, Bruxelles, March 1992, pp. 563-569.

[8] SLAMANI M., and KAMINSKA B., "Analog Circuit Testability Evaluation by Sensitivity Analysis", IEEE Pacific Test Workshop, Vancouver, April 1991.

[9] NAIM B-HAMIDA, KHALED SAAB, DAVID MARCHE, and BOZENA KAMINSKA, FaultMaxx: A Perturbation Based Fault Modeling and Simulation for Mixed-Signal Circuits, Asien Test Conference, October 1997, pp.182-187

[10] N. BEN-HAMIDA, and B. KAMINSKA, "Analog Circuit Testing Based on Sensitivity Computation," IEEE International Test Conference, Baltimore, Oct. 1993, pp. 652-661.

[11] P. DUHAMEL, and J. C. RAULT, "Automatic Test Generation Techniques for Analog Circuit and Systems: A Review", IEEE Transactions on Circuits and Systems, Vol. CAS-26, No 7, July 1979, pp.411-439.

[12] L. MILOR, V. VISVANATHAN, "Detection of Catastrophic Faults in Analog Integrated Circuit", IEEE Transactions on Computer-Aided Design, Feb. 1989, Vol. CAD-8, No 2, pp. 114-130.

[13] J. W. BANDLER, and A. E. SALAMA, "Fault Diagnosis of Analog Circuits", Proceedings of the IEEE, Vol. 73, No 8, August 1985, pp. 1279-1325.

[14] D.F. WONG, H.W. LEONG, and C.L.Liu, Simulated Annealing for VLSI Design, Kluwer Academic Publishers,1988.

[15] R. G. BENNETTS, "Progress in Design for Test: A Personal View", IEEE Design and Test of Computers, March 1994, pp. 89-112.

[16] T. W. WILLIAMS, and K. P. KENNETH, "Design for Testability: A Survey", Proceedings of the IEEE, Vol. 71, No 1, Jan. 1983, pp.89-112.

[17] M. S. ABADIR, A. R. PARIKH, P. A. SANDBORN, K. DRAKE, and L. BAL, "Analyzing Multichip Module Strategies", IEEE Design and Test of Computers, March 1994, pp. 40-52.

[18] LEON O. CHUA, and PEN-MIN LIN, "Computer-Aided Analysis of Electronic Circuits", Prentice-Hall, INC. 'Englewood Cliffs, New Jersey, 1975.

[19] STEPHAN W. DIRECTOR, and RONALD A. ROHRER, "Automated Network Design- The Frequency Domain Case", IEEE Trans. on Circuit Theory, CT-16, no 3, August 1969, pp. 330-337.

[20] J. CHOJEAN, and J.IZYDORCKY, "The Time Domain Sensitivity Computation Using SPICE2. The Linear Network Case", Proceedings Intern. AMSE Conference "Signal & System", Cetinje (Yugoslavia), Sep. 3-5, 1990, Vol 3, pp. 113-123.

[21] ASHOK K. SETH, "Comments on Time Domain Network Sensitivity Using the Adjoint Network Concept", IEEE Trans. on Circuit Theory, July 1972, pp. 367-370.

[22] STEPHAN W. DIRECTOR, and RONALD A. ROHRER, "The Generalized Adjoint Network and Network Sensitivities", IEEE Trans. on Circuit and Theory, Vol. CT-16, no 3, August 1969, pp. 318-323.

[23] STEPHAN W. DIRECTOR, and RONALD A. ROHRER, "Survey of Circuit-Oriented Optimization Techniques", IEEE Trans. on Circuit Theory, Vol. CT-18, no 1. Jan. 1971, pp. 3-10.

[24] KARL HARTMAN, "Noise Characterization of Linear Circuits", IEEE trans. on Circuit and Systems, Vol. CAS-23, no 10, Oct. 1976, pp. 581-590.

[25] R. J. A. HARVEY, A. M. D. RICHARDSON, E, M. J. G. BRRULS, and K. BAKER, "Analog Fault Simulation Based on Layout Dependent Fault Models", IEEE International Test Conference 1995, pp-641-649.

[26] MANOJ SACHDEV, "A Realistic Defect Oriented Testability Methodology for Analog Circuits", IEEE JETTA 1994.

[27] GIRI DEVARAYNADURG, and MANI SOMA, "Analytical Fault Modeling and Static Test Generation for Analog ICs", IEEE International Conference on CAD 1994, pp. 44-47.

[28] ERIC FELT and ALBERTO SANGIOVANNI-VINCENTELLI, Testing of Analog Systems Using Behavioral Models and Optimal Experimental Design Techniques, IEEE International Conference on CAD 94, pp. 672-678.

[29] D. WALKER, "VLASIC System User Manual Release 1.3", Carnegie-Mellon Uni. Pittsburgh USA., June 1990.

[30] META-SOFTWARE, "HSPICE User's Manual", Meta-Software, Inc. 1999.

[31] WILLIAM H. PRESS, SAUL A. TEUKOLSKY, WILLIAM T. VETTERLING, and BRIAN P. FLANNRY, "Numerical Recipes in C, Second Edition", Cambridge University Press, 1992.

[32] LAWRENCE T. PILLAGE, RONALD A. ROHRER, and CHANDRAMOULI VISWESWARIAH, "Electronic Circuit And System Simulation Methods", McGraw-Hill, Inc., 1995.

[33] H. WALKER, and S. W. DIRECTOR, "VLASIC: A Catastrophic Fault Yield Simulator For Integrated Circuits", IEEE Transactions on Computer Aided Design Of Integrated Circuits and Systems", Vol. CAD5 (4), PP541-556, Oct 1986.

[34] TELLEGEN B. D. H., "A General Network Theorem, with Application", Phillips Res. Dept., no. 7, pp. 259-269.

[35] R. G. BENNETTS, "Progress in Design for Test: A Personal View", IEEE Design and Test of Computers, March 1994, pp. 89-112.

[36] METROPOLIS N., ROSENBLUTH A., ROSENBLUTH M., TELLER A., and TELLER E., Journal of Chemical Physics 1953, vol. 21, pp. 1087-1092.

[37] MIRON ABRAMOVICI, MELVIN A. BREUER, and ARTHUR D. FRIEDMAN, Digital System Testing and Testable Design, IEEE Press, New York, 1990.

[38] MOUNIR YOUSSEF, YVON SAVARIA, and BOZENA KAMINSKA, 'A Methodology for Efficiently Inserting and Condensing Test Points, IEEE Proceedings.

[39] N. BEN-HAMIDA, K. SAAB, D. MARCHE, B. KAMINSKA, and G. QUESNEL, "LIMSoft: Automated Tool for Design and Test Integration of Analog Circuits" IEEE International Test Conference, Washington D.C., Oct. 1996, pp.56-71.

[40] W.W. WONG, R.S. WINTON, and J.J.LIOU, "Integrated-circuit sensitivity simulation using SPICE," IEE Proceedings-G, Vol. 138, No. 1, February 1991.

[41] JIRI VLACH, and KISHORE SINGHAL, "Compute Mehtods for Circuit Analysis and Design, second edition", Chapman and Hall, 1994.

[42] Matlab, "Matlab User's Manual," The MathWorks Inc. 1990.

[43] K. SAAB, D. MARCHE, N. BEN-HAMIDA, and B. KAMINSKA, "LIMSoft: Automated Tool For Sensitivity Analysis and Test Vector Generation", IEE Proceeding on Circuit Devices and Systems, Vol. 143, No. 6, Dec. 1996, pp. 386-392.

[44] M.CONTI, S. ORCIONI, C. TURCHETTI, G. SONCINI, and N. ZORZI. "Analytical Device Modeling for MOS Analog IC's Based on Regularization and Baysesian Estimation", IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems. Vol. 15, No. 11, Nov. 1996.

[45] J.P. SPOTO, W.T. COSTON, and C.P. HERNANDEZ. "Statistical Integrated Circuit Design and Characterization". IEEE Transactions on computer-aided Design of Integrated Circuits and Systems. Vol. CAD 5, No. 1, Jan. 1986, p90-p103

[46] P. YANG, D.E. HOCEVAR, P.F. COX, C. MACHALA, and P.K. CHATTERGEE. "An Integrated and Efficient Approach for MOS VLSI Statistical Circuit Design". IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems. Vol. CAD 5, No. 1, Jan. 1986.

[47] M.J.M. PELGROM, A.C.J. DUINMAIGER, and P.P.G. WELBERS. "Matching Properties of MOS", Transistors IEEE J. Solid-state Circuits, Vol. 24, pp. 1433-1439, Oct. 1989.

[48] C.MICHAEL, and M. ISMAIL. "Statistical Modeling of Device Mismatch for analog MOS Integrated Circuits", IEEE Journal of solid state Circuits. Vol. 2. No. 2. Feb. 1992.

[49] WILLIAM W. HINES, and DOUGLAS C. MONTGOMERY, "Probability and Statistics in Engineering and Management Science", WILEY press, 1990 third edition.

[50] KHALED SAAB, DAVID MARCHE, and KAMINSKA B., "Analog Circuit Testability Evaluation by Sensitivity Analysis", IEEE 1st Mixed Signal Testing Workshop, Grenoble, France, June 1995.

[51] NAVEENA NAGI, A. CHATTERJEE, and JACOB A. ABRAHAM, "Fault Simulation of Linear Analog Circuits", Analog Integrated Circuits and Signal Processing 1993.

[52] NAVEENA NAGI, and JACOB A. ABRAHAM, "Hierarchical Fault Modeling For Analog and Mixed-Signal Circuits", IEEE VLSI Test Symposium 1992, pp. 92-101.

ANNEXE A

Annexe A
Operational amplifier

Hspice Netlist

```
* # FILE NAME: ...cmosoa.cir

Vs 0 21  5.0
Vd 20 0  5.0
Vin in 0 AC 1 DC 1 sin(0 .3 1Meg)
R1 in VN 10k
R2 VN VO 10k
VP VP 0 DC 0

Xop1 VO VP VN 20 21 amp
.subckt amp VO VP VN NET67 NET34
R77 NET32 VO ...E3 M=1.0
C76 NET48 NET32 ...E-12 M=1.0
M6 NET48 VP NET44 NET67 PMOS L=4E-6 W=30E-6 M=1.0
M3 NET35 VN NET44 NET67 PMOS L=4E-6 W=30E-6 M=1.0
M9 VO NET48 NET34 NET34 NMOS L=3E-6 W=154.2E-6 M=1.0
M4 NET35 NET35 NET34 NET34 NMOS L=4E-6 W=15E-6 M=1.0
M7 NET48 NET35 NET34 NET34 NMOS L=4E-6 W=15E-6 M=1.0
M2 NET54 NET54 NET34 NET34 NMOS L=32E-6 W=3E-6 M=1.0
M8 VO NET54 NET67 NET67 PMOS L=4E-6 W=200E-6 M=1.0
M1 NET54 NET54 NET67 NET67 PMOS L=4E-6 W=12E-6 M=1.0
M5 NET44 NET54 NET67 NET67 PMOS L=4E-6 W=30E-6 M=1.0
.MODEL NMOS NMOS
.MODEL PMOS PMOS
.ends amp
*.AC DEC       20.0000       1.00000       1.0000000E+09
*.DC Vin -10 10 1
*.OP
.OPTIONS post=1 method=gear reli=1
.end
```

Porcess file

PHENOMENON: Process Variation
r*.value Gaussian 0 0.05 0.05 1 1 Gaussian 0 0.005 0.005 1
.r.value Gaussian 0 0.05 0.05 1 1 Gaussian 0 0.005 0.005 1
c*.value Gaussian 0 0.05 0.05 1 2 Gaussian 0 0.005 0.005 1
.c.value Gaussian 0 0.05 0.05 1 2 Gaussian 0 0.005 0.005 1
*.w Gaussian 0 0.05 0.05 1 3 Gaussian 0 0.005 0.005 1
*.l Gaussian 0 0.05 0.05 1 4 Gaussian 0 0.005 0.005 1

Test Vectors (Stimuli)

```
*  Circuit: /usr2/tmp/Khaled_tmax/cmosoa.cir

*  TestMaxx Test 0
*  .dc_test
*    output
*       I vd  1 main
*    simulation
*       Vin  -10v  10v  0.1v
*    test_vector
*       [-10 10]
*    filename
*       cmosoaIvd
*  .end_of_test

*  TestMaxx Test 1
*  .dc_test
*    output
*       I vs  1 main
*    simulation
*       Vin  -10v  10v  0.1v
*    test_vector
*       [-10 10]
*    filename
*       cmosoaIvs
*  .end_of_test

*  TestMaxx Test 2
*  .dc_test
*    output
*       V vo  0 main
*    simulation
*       Vin  -10v  10v  0.1v
*    test_vector
*       [-10 10]
*    filename
*       cmosoav
*  .end_of_test
```

Fault Coverage

```
%  Summary Soft Fault Probabilistic Coverage Report
%=====================================================

%  Circuit: /usr2/tmp/khaled_tmax/cmosoa.cir

Total false reject: 0.001228
Total false accept: 0.165483
```

fault name	deviation type	false accept	false reject
r1.value	negative deviation	0.000000e+00	1.349809e-03
r1.value	positive deviation	0.000000e+00	1.349808e-03
r2.value	negative deviation	0.000000e+00	1.349808e-03
r2.value	positive deviation	0.000000e+00	1.349808e-03
xop1.c76.value	negative deviation	1.000000e+00	0.000000e+00
xop1.c76.value	positive deviation	1.000000e+00	0.000000e+00
xop1.m1.l	negative deviation	1.239144e-01	1.349808e-03
xop1.m1.l	positive deviation	1.269001e-04	1.353323e-03
xop1.m1.w	negative deviation	1.345781e-01	1.353323e-03
xop1.m1.w	positive deviation	1.583431e-04	1.349808e-03
xop1.m2.l	negative deviation	8.375172e-03	1.353323e-03
xop1.m2.l	positive deviation	0.000000e+00	1.349809e-03
xop1.m2.w	negative deviation	2.393111e-02	1.349808e-03
xop1.m2.w	positive deviation	0.000000e+00	1.349809e-03
xop1.m3.l	negative deviation	8.227952e-07	1.353383e-03
xop1.m3.l	positive deviation	0.000000e+00	1.349808e-03
xop1.m3.w	negative deviation	1.435876e-04	1.349808e-03
xop1.m3.w	positive deviation	7.273043e-16	1.353264e-03
xop1.m4.l	negative deviation	0.000000e+00	1.349808e-03
xop1.m4.l	positive deviation	8.375529e-04	1.353323e-03
xop1.m4.w	negative deviation	5.150280e-05	1.353323e-03
xop1.m4.w	positive deviation	0.000000e+00	1.349809e-03
xop1.m5.l	negative deviation	7.235085e-01	1.349809e-03
xop1.m5.l	positive deviation	1.660425e-02	1.353204e-03
xop1.m5.w	negative deviation	8.126844e-01	1.353204e-03
xop1.m5.w	positive deviation	3.248334e-03	1.349809e-03
xop1.m6.l	negative deviation	0.000000e+00	1.349808e-03
xop1.m6.l	positive deviation	3.472336e-15	1.353264e-03
xop1.m6.w	negative deviation	3.577805e-05	1.353323e-03
xop1.m6.w	positive deviation	0.000000e+00	1.349809e-03
xop1.m7.l	negative deviation	3.185136e-07	1.353383e-03
xop1.m7.l	positive deviation	0.000000e+00	1.349808e-03
xop1.m7.w	negative deviation	1.043081e-04	1.349808e-03
xop1.m7.w	positive deviation	1.297833e-15	1.353264e-03
xop1.m8.l	negative deviation	8.255878e-04	1.353323e-03

«op1.m8.l	positive deviation	0.000000e+00	1.349809e-03
«op1.m8.w	negative deviation	4.100263e-03	1.349809e-03
«op1.m8.w	positive deviation	0.000000e+00	1.349808e-03
«op1.m9.l	negative deviation	5.988055e-01	1.349808e-03
«op1.m9.l	positive deviation	7.540756e-02	1.353323e-03
«op1.m9.w	negative deviation	6.261931e-01	1.353323e-03
«op1.m9.w	positive deviation	7.320392e-02	1.349808e-03
«op1.r77.value	negative deviation	1.000000e+00	0.000000e+00
«op1.r77.value	positive deviation	1.000000e+00	0.000000e+00

```
%  Summary Soft Fault Deterministic Coverage Report
%=======================================================

%  Circuit: /usr2/tmp_khaled_tma<_/cmosoa.cir

TOTAL FAULT DETECT: 32/44  ---> 0.727273

   fault name |   deviation type   | test #
              |                    |  0  1  2
              |                    | DC DC DC
--------------+--------------------+-----------
     r1.value | negative deviation |  <  <  <
     r1.value | positive deviation |  <  <  <
     r2.value | negative deviation |  <  <  <
     r2.value | positive deviation |  <  <  <
<op1.c76.value | negative deviation |
<op1.c76.value | positive deviation |
    <op1.m1.l | negative deviation |        <
    <op1.m1.l | positive deviation |        <
    <op1.m1.w | negative deviation |        <
    <op1.m1.w | positive deviation |        <
--------------+--------------------+-----------
    <op1.m2.l | negative deviation |  <  <  <
    <op1.m2.l | positive deviation |  <  <  <
    <op1.m2.w | negative deviation |  <  <  <
    <op1.m2.w | positive deviation |  <  <  <
    <op1.m3.l | negative deviation |  <  <  <
    <op1.m3.l | positive deviation |  <  <  <
    <op1.m3.w | negative deviation |  <  <  <
    <op1.m3.w | positive deviation |  <  <  <
    <op1.m4.l | negative deviation |  <  <  <
    <op1.m4.l | positive deviation |  <  <  <
--------------+--------------------+-----------
    <op1.m4.w | negative deviation |  <  <  <
    <op1.m4.w | positive deviation |  <  <  <
    <op1.m5.l | negative deviation |
    <op1.m5.l | positive deviation |
    <op1.m5.w | negative deviation |
    <op1.m5.w | positive deviation |
    <op1.m6.l | negative deviation |  <  <  <
    <op1.m6.l | positive deviation |  <  <  <
    <op1.m6.w | negative deviation |  <  <  <
    <op1.m6.w | positive deviation |  <  <  <
--------------+--------------------+-----------
    <op1.m7.l | negative deviation |  <  <  <
    <op1.m7.l | positive deviation |  <  <  <
    <op1.m7.w | negative deviation |  <  <  <
    <op1.m7.w | positive deviation |  <  <  <
    <op1.m8.l | negative deviation |  <     <
    <op1.m8.l | positive deviation |  <     <
```

xop1.m8.w	negative deviation	X X X
xop1.m8.w	positive deviation	X X X
xop1.m9.l	negative deviation	
xop1.m9.l	positive deviation	
xop1.r77.value	negative deviation	
xop1.r77.value	positive deviation	
Detect pct.		63 59 72

Uncovered Faults

```
% Uncovered Faults
%==================

«op1.c76.value positive deviation |
«op1.c76.value negative deviation |
    «op1.m5.l positive deviation |
    «op1.m5.l negative deviation |
    «op1.m5.w positive deviation |
    «op1.m5.w negative deviation |
    «op1.m9.l positive deviation |
    «op1.m9.l negative deviation |
    «op1.m9.w positive deviation |
    «op1.m9.w negative deviation |
«op1.c77.value positive deviation |
«op1.c77.value negative deviation
```