Plateforme CAO pour le test de circuits mixtes

Ahcène Bounceur

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

THÈSE

pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité : Micro et Nano Électronique

préparée au laboratoire **TIMA** dans le cadre de
l'École Doctorale d'Électronique, d'Électrotechnique, d'Automatique et de Traitement du Signal

présentée et soutenue publiquement par

Ahcène BOUNCEUR

le 13 Avril 2007

Titre :

Plateforme CAO pour le test de circuits mixtes

Directeur de thèse : M^{r}. Salvador MIR
Co-directeur : M^{r}. Emmanuel SIMEU

Jury

M^{r}.	Pierre GENTIL,	Président
M^{r}.	Yves BERTRAND,	Rapporteur
M^{me}.	Bozena KAMINSKA,	Rapporteur
M^{r}.	Salvador MIR,	Directeur de Thèse
M^{r}.	Emmanuel SIMEU,	Co-directeur
M^{r}.	Bernard COURTOIS,	Examinateur
M^{r}.	Jean-Louis CARBONÉRO,	Examinateur

A mon épouse Kahina et à notre fille Inès,
A mes parents El-hadi et Nadia,
A mes grand-parents Baba-Sadek et Ima-baya et à tous mes oncles,
A mes beaux parents Da Mouhand-Akli et Khalti Baya.

Avant-Propos

Il vaut mieux honorer sa profession
que d'être honoré par elle
(Proverbe).

Ce travail de thèse a été réalisé au laboratoire TIMA (Techniques de l'Informatique et de la Microélectronique pour l'Architecture des ordinateurs), sous la direction de Monsieur Salvador Mir, chef du groupe RMS et sous la co-direction de Monsieur Emmanuel Simeu, chercheur du groupe RMS. Ce groupe est spécialisé particulièrement dans le domaine du test des circuits analogiques, mixtes, RF et microsystèmes. En effet, la miniaturisation continue des dispositifs intégrés rend leur test de plus en plus compliqué surtout avec la possibilité, aujourd'hui, d'intégrer dans une seule puce des parties à la fois numériques et analogiques.

Durant nos travaux, nous avons développé une plateforme CAT (Computer Aided Test) pour le test des circuits analogiques et mixtes. Cette plateforme est actuellement intégrée dans l'environnement de conception de circuits microélectroniques Cadence. Son développement a été fait en partie dans le cadre du projet européen NanoTEST 2A-702 du programme MEDEA+, en particulier, en ce qui concerne la génération de vecteurs de test et l'extension des outils de simulation de fautes aux circuits RF. Ce dernier aspect a été réalisé avec la collaboration de ST Microelectronics dans le cadre de NanoTEST.

L'état de l'art présenté est développé avec assez de détails pour la partie de génération de vecteurs de test. La compréhension détaillée de cette partie n'est peut être pas nécessaire en première lecture, mais elle permet de comprendre les choix qui ont été faits pour l'intégration des outils de génération de vecteurs de test dans la plateforme. En outre, les algorithmes des autres techniques de génération présentés dans l'état de l'art pourront être considérés pour le développement futur de la plateforme.

On commence par vouloir tout le bien,
et on finit par espérer le moindre mal
(Proverbe).

Remerciements

Je tiens tout d'abord à remercier Monsieur Bernard Courtois, *Directeur du laboratoire TIMA-CMP*, de m'avoir accueilli au sein de TIMA et d'avoir accepté d'examiner cette thèse.

Je tiens à exprimer ma reconnaissance à mon directeur de thèse, Monsieur Salvador Mir, *Chargé de recherche au CNRS et chef du groupe RMS*, pour m'avoir accueilli au sein de son équipe, pour tous les précieux conseils qu'il m'a donnés, pour le temps qu'il a consacré pour diriger cette thèse et surtout pour la confiance qu'il m'a témoignée. Ceci m'a vraiment aidé à promouvoir mes recherches dans le domaine scientifique.

Je remercie chaleureusement mon co-directeur de thèse, Monsieur Emmanuel Simeu, *Maître de conférence UJF*, pour ses conseils et discussions très constructifs, ayant permis une très bonne orientation des travaux de cette thèse et pour le temps consacré à la correction de ce rapport.

Mes remerciements vont tout particulièrement à Monsieur Jean-Louis Carbonéro, *manager du groupe Analog, Mixed-Signal & RF Design for Test & Test (ST Microelectronics, Crolles)*, d'avoir participé aux réunions à TIMA grâce auxquelles le travail de cette thèse a pu avoir une valeur à la fois théorique et pratique. Je le remercie aussi d'avoir accepté d'examiner cette thèse.

Je voudrais exprimer mes remerciements les plus sincères à Monsieur Pierre Gentil, *Directeur de l'École EEATS*, d'avoir accepté de présider le jury de cette thèse.

Enfin, je remercie tout particulièrement et sincèrement, Madame Bozena Kaminska, *Professeur à l'Université Simon Fraser University de Canada*, et Monsieur Yves Bertrand, *Professeur de l'Université de Montpellier*, d'avoir accepté de lire ce manuscrit en tant que rapporteurs.

Cette thèse n'aurait pas été aussi enrichissante sans la présence et l'expérience de certaines personnes. Je pense en particulier à Monsieur Alexandre Chagoya, *Ingénieur de support des logiciels de simulation au CIME*, qui a fortement contribué aux différentes installations et à l'entretien de toute la suite Cadence qui était le noyau principal de la plateforme CAT développée dans le cadre de cette thèse, et à Monsieur Cosmin Roman, *Post-Doctorant*, pour ses premiers cours bien structurés du langage SKILL. De même, je souhaiterai remercier, Monsieur Ikhlef Bechar, *Doctorant à l'INRA*, pour ses précieuses discussions sur les statistiques appliquées au traitement d'image. Je remercie aussi mes collègues de bureau, Monsieur Achraf Dhayni et Monsieur Luis Rolíndez, grâce à qui j'ai pu me familiariser avec le monde de la micro-électronique. Je les remercie aussi d'avoir accepté mes collaborations dans certains de leurs travaux de thèse dans une ambiance fraternelle.

Enfin, je tiens à exprimer ma sympathie à tout ceux qui ont contribué de près ou de loin au bon déroulement de ces trois ans, et tout particulièrement, toute l'équipe du CMP (Hubert, Jean-François et Kholdoun), les secrétaires (Joelle, Lucie, Anne-Laure, Isabelle, Sophie, Patricia, Chantale, Corinne et Marie-Christine), l'équipe informatique (Nico et Fred), Gérard, les membres de tous les groupes de TIMA (Paco, Yasser, Marius, Estelle, Kati, Uliana, Amel, Mohamed, Youssef, Abdelaziz, Eric, Gilles, Saeed, Hela, Louis, Nacer, Patrice, Lobna, Aimen, Wassim, etc.) et en particulier ceux du groupe RMS (Libor, Haralampos, Rafik, Nourredine, Matthieu, Nam, Livier, Jeanne, Anna, Jonathan, Lucas, Farès et Imen) et un merci bien particulier pour mon amie précieuse Françoise. Merci aussi à Yassine et Aziz pour tout le bien qu'ils m'ont fait.

Bien sûr, je ne saurai oublier mes parents pour leurs encouragements permanents et enfin ma tendre épouse Kahina pour ses soutiens et ses encouragements dans les instants les plus difficiles. Merci pour son aide et pour avoir supporté mes humeurs durant cette période très dense en activités. Un grand merci.

Table des matières

Liste des figures

Liste des tableaux

Liste des Algorithmes

Chapitre 1

Introduction

1.1 Introduction

Plusieurs nouveaux domaines sont devenus très importants dans la vie quotidienne ces derniers temps, tels que la télécommunication, le multimédia, les applications biomédicales, etc. Ces domaines nécessitent une utilisation importante des circuits intégrés analogiques et mixtes. Tester ces derniers cause un impact considérable sur le coût de production. En outre, la tendance d'intégrer des systèmes Analogiques/Numériques complets dans une seule puce pose de très sérieux problèmes d'accès pour le test.

Le test des circuits intégrés (ICs[1]) est devenu une tâche essentielle dans l'industrie des semi-conducteurs. Ces circuits peuvent contenir dans une seule puce des millions de transistors (en plus à d'autres composants tels que les résistances, les capacités ou les diodes). Le développement d'approches et d'outils pour la génération et l'application des stimuli de test (ou de vecteurs de test) pour détecter le dysfonctionnement des dispositifs (ou des circuits), et localiser les fautes ou les erreurs du design, nécessite un grand effort dans la recherche en raison de l'évolution rapide des technologies des semi-conducteurs.

L'approche la plus utilisée pour tester un circuit analogique, c'est-à-dire, savoir si le circuit fonctionne bien ou non, est de mesurer ses performances et voir si celles-ci correspondent aux spécifications attendues (celles d'un bon circuit), ce qui est appelé test fonctionnel. Mais en réalité cette méthode s'avère très coûteuse. D'autres types de test utilisés incluent le test structurel et le test alternatif. Les tests fonctionnel et structurel dépendent de la manière dont le vecteur de test est généré, alors que le test alternatif dépend des mesures effectuées sur le circuit sous test.

Les travaux de recherche concernant le test des circuits analogiques ont été basés traditionnellement sur le diagnostic des fautes. Mais avec l'accroissement rapide du niveau d'intégration des puces de nouvelle génération SoC[2] (Système-sur-Puce), non seulement diagnostiquer les fautes est considéré comme un problème difficile mais aussi distinguer les bons circuits (*circuits fonctionnels*) des mauvais (*circuits défaillants*) est devenu un véritable problème. Des blocs analogiques intégrés dans des systèmes numériques ne peuvent pas être facilement testés [78]. Par conséquent, de nombreuses techniques ont été proposées afin de faciliter la tâche de test des circuits analogiques et mixtes.

Ces techniques visent l'optimisation du test lors de la fabrication (test de production) ou lors de l'application (test en-ligne/test hors-ligne). Certaines techniques permettent d'ajouter des circuits dans la puce pour faciliter le test (DFT[3]) ou même réaliser un auto-test (BIST[4]). Dans

[1] Integrated Circuits.
[2] System-on-Chip.
[3] Design For Test.
[4] Built In Self Test.

tous les cas, il devient essentiel d'évaluer la qualité des techniques de test lors de la conception, en utilisant des outils de la CAO orientés au test (CAT[5]).

1.2 Motivations

Le test analogique fait partie actuellement de l'un des domaines de recherche en micro-électronique en fort développement. Par conséquent, l'ensemble des outils de test, à savoir, les équipements de test ainsi que les logiciels et les outils CAO pour le test n'ont pas encore atteint la maturité nécessaire pour faciliter le test analogique et assurer des résultats de test significatifs. Il reste à mentionner, cependant, que la partie équipement de test est plus développée que la partie logiciel pour le test.

La miniaturisation incessante des circuits microélectroniques a donné naissance à de nouvelles techniques de test, telles qu'intégrer directement dans la puce les circuits de test remplaçant totalement les équipements de test (BIST). Celles-ci nécessitent un ensemble de logiciels de CAT afin d'optimiser le BIST et d'évaluer la qualité des tests effectués. Cependant, ces outils sont rares et il n'existe aucun logiciel complet commercialisé pour ce genre de tâches.

Dans le domaine du test numérique, il existe un ensemble important d'outils de simulation et d'injection de fautes commercialisés, ce qui n'est pas le cas pour le domaine du test analogique. La majorité des outils développés pour ce dernier domaine sont destinés généralement aux groupes de recherche et les laboratoires concernés. Ces outils ne sont pas commercialisés en raison de leur nature académique bien qu'ils soient devenus, actuellement, très intéressants pour les raisons suivantes :

- Le test fonctionnel (à base de performances), est souvent impossible en raison du coût et du temps de test inacceptables et excessifs. D'où la nécessité, soit de le remplacer par un autre test (par exemple, structurel ou alternatif), soit, de réduire l'ensemble des performances du circuit sous test.
- La génération de vecteurs de test analogiques nécessite la simulation de fautes analogiques.
- Le calcul de la Couverture de fautes ne suffit pas pour évaluer un test analogique, car, une bonne Couverture de fautes n'assure pas forcément des bonnes valeurs des métriques de test analogique (Rendement de test, Couverture de rendement, Perte de rendement). D'où la nécessité de développer de nouvelles techniques pour l'évaluation du test analogique.

Cette thèse vise le développement d'une plateforme intégrant les outils et les techniques permettant de remédier à ces problèmes.

1.3 Objectifs

L'objectif principal de ce travail est de mettre en place une plateforme CAO pour le test de circuits analogiques, intégrée dans l'environnement de conception microélectronique Cadence. Cette plateforme intègre des outils de modélisation, de simulation et d'injection de fautes, des outils de génération et d'optimisation de vecteurs de test et des outils d'évaluation de la qualité de test. Aussi, cette thèse vise une technique statistique permettant de calculer les différentes métriques de test analogique.

Contrairement au test *structurel* (test basé sur les fautes) où la Couverture de fautes est la métrique la plus importante utilisée comme élément permettant d'évaluer le test, le test *fonctionnel* (test basé sur les spécifications) nécessite de connaître à la fois toutes les *performances* du circuit ainsi qu'un ensemble de mesures appelées *critères de test* pour pouvoir calculer les métriques permettant d'évaluer le test. Les ***spécifications*** représentent un ensemble d'intervalles des bonnes

[5]Computer Aided Test.

valeurs des performances du circuit. Elles sont données par le constructeur et permettent de savoir si le CUT[6] est **fonctionnel**[7], c'est-à-dire que toutes ses spécifications sont vérifiées, ou bien s'il est **défaillant**[8], c'est-à-dire qu'au moins une de ses spécifications est violée (ou n'est pas vérifiée). Les critères de test sont des mesures qui peuvent être prises sur le CUT, qui sont désignées par l'ingénieur de test. Elles peuvent même être un sous-ensemble des performances. Pendant la phase de test, l'ensemble des critères de test est supposé être une copie, au sens figuré, des performances du CUT. Cependant, si tous les critères de test sont vérifiés, le circuit n'est pas déclaré comme fonctionnel, mais il est déclaré comme *circuit qui passe le test*. Dans le cas où au moins un des critères de test n'est pas vérifié, le circuit est donc déclaré comme *circuit qui échoue le test*. L'évaluation du test consiste à mesurer ou à calculer à quelle point ces nouvelles mesures sont une copie des performances. Ceci se fait en calculant l'ordre de l'erreur commise sur le nombre de circuits fonctionnels qui échouent le test et le nombre de circuits défaillants qui passent le test. L'ordre de l'erreur est aussi évalué en calculant les métriques de test précités. Ce calcul n'est possible qu'après avoir calculé précisément les intervalles des bonnes valeurs des critères de test, appelées ***limites de test*** en utilisant un technique statistique présentée dans le cadre de cette thèse. Ces limites seront ensuite utilisées pour calculer la Couverture de fautes et pour réduire l'ensemble des performances, souvent très grand, d'un CUT analogique ou mixte afin de vérifier sa fonctionnalité.

Pour ce faire, un ensemble d'outils est nécessaire pour automatiser certaines tâches très lourdes en temps d'exécution ainsi qu'en précision de calcul, tel que la simulation de fautes. Ces outils ont aussi été développés dans le cadre de cette thèse.

1.4 Contributions

Le développement des outils de CAO pour le test dans le cadre de cette thèse nécessite un bagage pluridisciplinaire. Le but étant que cette plateforme soit utilisable pour n'importe quel circuit analogique, mixte et RF. Et que n'importe quelle amélioration des algorithmes d'optimisation implémentés soit possible sans modifier le code source du programme principal. Les contributions principales de ce travail sont :

- Mise-en-place d'une plateforme CAO pour le test de circuits analogiques et mixtes. Celle-ci est implémentée dans l'environnement de conception microélectronique Cadence. Elle nécessite l'utilisation de plusieurs langages de programmation tels que le SKILL, C++, Java ainsi que toutes les fonctions de la suite Cadence (Virtuoso [3], Ocean [1], Spectre [2], etc.) [19, 21, 20].
- Développement d'une technique statistique permettant d'estimer les différentes métriques de test analogique et fixer les limites de test [24, 25].
- Intégration dans la plateforme d'un ensemble d'outils pour la modélisation, l'injection et la simulation de fautes. Le développement de ces outils nécessite l'utilisation des techniques de la Recherche Opérationnelle (Théorie des graphes, l'optimisation combinatoire et la programmation linéaire) pour modéliser certains problèmes d'optimisation ayant une grande complexité (problèmes NP-Difficiles), tels que le problème de recouvrement souvent rencontré.
- Une étude approfondie a été faite sur les techniques et les outils pour la génération et l'optimisation de vecteurs de test. Ce qui a permis de choisir les techniques à intégrer dans la plateforme. Un autre outil a été ajouté comme extension de la plateforme pour traiter

[6] Circuit Under Test ou Circuit sous test.
[7] Terme utilisé pour un bon circuit ou qui marche bien.
[8] Terme utilisé pour un circuit mauvais ou qui ne marche pas bien.

le cas des dispositifs non linéaires [37, 38, 39, 40]. Les outils de génération de vecteurs de test ont nécessité une implémentation d'algorithmes génétiques (WARGA et NSGA) dans le but d'optimiser la qualité des vecteurs de test analogiques [18, 23, 22].

La Figure 1.1 montre l'architecture de la plateforme CAO pour le test proposée. Elle est composée d'un ensemble de trois boites d'outils séparées. La modélisation de fautes, l'injection et la simulation de fautes sont effectuées en utilisant la boite d'outils *FIDESIM*. Les résultats sont sauvegardés dans une base de données qui pourra être consultée par les autres outils, en particulier, la boite d'outils *OPTEVAL* pour l'évaluation de test et la boite d'outils *OPTEGEN* pour la génération et l'optimisation des vecteurs de test.

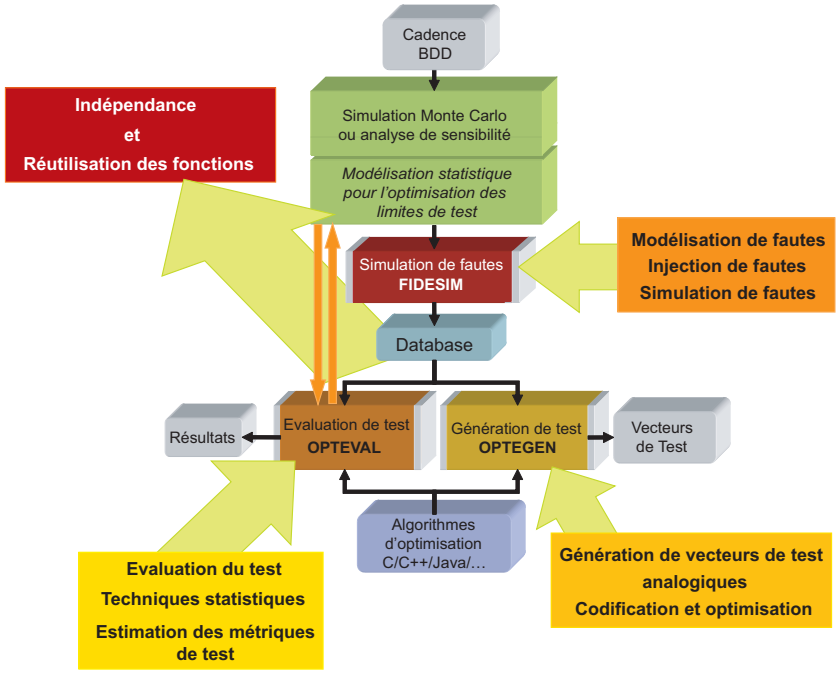


FIG. 1.1 – L'architecture simplifiée de la plateforme CAO pour le test.

1.5 Présentation de la thèse

Dans le Chapitre 2 les concepts de base du test analogique sont rappelés. A cause de la non uniformité de la terminologie dans le domaine du test analogique, dans ce chapitre, une terminologie concernant le test analogique sera présentée. Ensuite, les différentes techniques de test ainsi que les métriques permettant d'évaluer ces tests seront introduites.

Dans l'état de l'art (Chapitre 3), nous présenterons des travaux récents sur la simulation de fautes analogiques ainsi que sur la génération et l'optimisation de vecteurs de test. Les techniques de génération des vecteurs de test sont classées en trois familles. La première représente les vecteurs de test fonctionnels. La deuxième représente les vecteurs de test structurels (vecteurs de test à base de fautes). La dernière famille, représente les vecteurs de test alternatifs, qui ressemble à la première famille, sauf que pour celle-ci, les performances sont déduites directement à partir de nouvelles mesures de test considérées (par régression par exemple).

Il sera présenté dans le Chapitre 4 trois techniques permettant d'évaluer un test analogique en modélisant statistiquement les circuits sous test. La première technique est basée sur la régression mais elle ne permet pas d'obtenir des résultats précis ; la deuxième technique considère les circuits dont la distribution de leurs performances et critères de test est multinormale. Basée sur cette

dernière, la troisième technique permet d'améliorer la précision sur les calculs des métriques de test, et dans certains cas, il n'est pas nécessaire de connaître la distribution de probabilité des performances et des critères de test. Ces techniques seront ensuite validées sur un amplificateur opérationnel différentiel conçu sous la technologie $0,18\mu m$ de ST Microelectronics.

Le Chapitre 5 sera consacré à la présentation des outils de modélisation, d'injection et de simulation de fautes de la plateforme CAT développée. Ces outils permettent d'injecter les fautes directement au niveau schématique des circuits sous test, avant la génération de la netlist. Ceci, rend ces outils complètement indépendants du langage des simulateurs utilisés tels que spectre par exemple. Car, la netlist est générée une fois la forme schématique du circuit sous test est modifiée. Les modèles de fautes sont créés comme des circuits Cadence. Ces circuits doivent respecter certaines règles sur la nomination des différentes connections et composants. Ces règles sont définies par un langage dit FMDL (Fault Model Description Language) afin d'automatiser l'injection de fautes. La procédure de description des différentes injections de fautes de l'utilisateur est décrite par un pseudo-code écrit dans des fichiers FID (Fault Injection Description). Huit types de fichiers FID on été définis. Ceux-ci permettent de décrire comment une certaine faute doit être injectée dans le circuit sous test. Par exemple, le fichier FID de type 4 permet d'injecter les court-circuits. L'ensemble des performances avec leurs spécifications et des critères de test avec leurs limites sont décrits dans un fichier appelé Programme de Configuration de Test. Ces outils peuvent utiliser plusieurs programmes de configuration de test pour considérer si une faute est détectée ou non, dans le cas où le calcul des performances et critères de test se fait à base de plusieurs configurations des bancs de test du CUT.

Dans le Chapitre 6, nous présenterons des outils d'optimisation et de génération de vecteurs de test. Ce chapitre est composé de quatre parties principales. Dans la première partie, nous avons présenté un nouvel algorithme permettant de réduire l'ensemble des performances d'un circuit. Cette réduction est basée sur le calcul du Taux de défauts engendré par un certain sous-ensemble des performances de test. Le sous-ensemble pour lequel le Taux de défauts est minimal sera donc choisi comme étant l'ensemble des performances réduit à considérer. La deuxième partie de ce chapitre présente une application de l'algorithme de génération de vecteurs de test multi-fréquences présenté dans [81]. Cet algorithme a été présenté pour une simulation de fautes à base de la fonction de transfert d'un filtre biquadratique. Dans ce chapitre cet algorithme a été présenté pour une simulation de fautes au niveau transistor de ce même filtre. Les résultats obtenus étaient plus précis que ceux du premier cas. Cet algorithme a été aussi utilisé pour le cas d'un amplificateur de ST Microelectronics. Le vecteur de test obtenu a permis d'atteindre une couverture de fautes maximale de 100%. Ensuite, nous avons présenté dans la troisième partie une technique permettant la génération de vecteurs de test numériques optimaux pour tester des circuits analogiques. La génération de ces vecteurs est un problème non linéaire et multi-critère. C'est-à-dire qu'il faut plusieurs critères pour pouvoir décider si un vecteur est d'une bonne qualité ou non, tels que le THD, $SFDR$, etc. Et aussi, ces critères sont de nature non linéaires. L'optimisation de ce genre de problèmes est très complexe, d'où donc le développement d'algorithmes génétiques multi-objectifs tels que WARGA ou NSGA pour les optimiser. La dernière partie de ce chapitre concerne les circuits non-linéaires. Nous avons donc montré comment générer un vecteur de test pour ce genre de circuits. Ce vecteur de test est de nature pseudo-aléatoire et multi-niveaux. Il permet à la fois de tester ces circuits et aussi de les caractériser en calculant les noyaux de Volterra. Un outil a été développé et intégré dans la plateforme pour effectuer ces opérations.

Enfin, dans le dernier chapitre (Chapitre 7) présentant la conclusion, quelques recommandations et perspectives seront énumérées.

Chapitre 2

Concepts de base du test analogique

2.1 Introduction

Le test peut être définit comme "*Le processus de vérifier qu'un circuit intégré réponde aux spécifications pour lesquelles il a été conçu*" [61]. D'une autre manière, le processus de test tente de répondre aux trois questions suivantes :

- Est ce que le circuit fonctionne ?
- Est ce que le circuit cache un problème potentiel ?
- Le circuit a-t-il sa capacité complète ?

En se basant sur ces trois questions formulées ci-dessus, trois types majeurs de test ont été historiquement considérés [26]. Le premier type représente le *test fonctionnel*, puisqu'il est destiné à vérifier que le circuit sous test possède les caractéristiques fonctionnelles attendues. Le deuxième type représente le *test structurel*, qui est destiné à vérifier s'il y a présence d'une faute dans le circuit sous test. Ce type de test peut être facile comme il peut être difficile, car il dépend du but, si la faute doit être détectée ou bien localisée. Enfin, le dernier et le troisième type de test est le *test paramétrique*. Celui-ci est utilisé pour vérifier si certains paramètres du circuit sont dans la plage des valeurs requises.

Ceci ne représente pas l'unique façon de classifier les tests. Un test peut être effectué d'une manière *statique* ou *dynamique*. Dans la première, seulement les caractéristiques de l'état stationnaire, ou DC, du circuit sous test sont intéressantes. D'autre part, dans la dernière, la réponse temporelle est d'une importance capitale pour évaluer une puce. Une sous-classe principale du test dynamique est celle appelée *at-speed testing* où le circuit sous test est testé à vitesse d'exploitation[1] nominale (ou même maximale).

D'autres classifications peuvent être aussi basées sur la manière dont le vecteur de test est appliqué [28] où nous distinguons le test *externe* et *interne (ou autotest)*. Le test externe nécessite d'utiliser le circuit hors son environnement opérationnel et l'utilisation d'équipements de test externes. Contrairement au test externe, les techniques de l'autotest sont directement intégrées dans la puce même dans le but d'éviter l'utilisation des équipements de test externes.

Finalement, une autre subdivision distingue entre les méthodes de test *concurrent* et *non-concurrent* dépendantes sur comment le circuit sous test peut être testé pendant sa phase opérationnelle (ou pendant que le circuit effectue ses fonctions).

[1] Operating speed.

2.2 Concepts de base du test analogique

Il existe quelques concepts qui doivent être définis précisément pour éviter une fausse interpretation pour la théorie et l'application du test analogique. La première définition devant être considérée avec attention est l'idée du test elle même, qui est introduite de façon ambiguë dans la littérature. Une large étude et analyse de ces concepts a été faite dans [58].

Le test des circuits analogiques est caractérisé par trois tâches principales : la détection, la localisation et le diagnostic [46]. Toutes ces trois tâches nécessitent l'utilisation d'un vecteur de test. Cependant, la détection d'une faute, sa localisation et son diagnostic dépendent de la qualité du vecteur du test.

2.2.1 Terminologie et Définitions

Terminologie générale

- *Performances :* elles décrivent le fonctionnement du circuit. Connaître les performances d'un circuit permettra de juger s'il est fonctionnel ou bien défaillant.
- *Spécifications :* c'est l'ensemble des valeurs acceptables de chaque performance. Chaque performance possède deux spécifications (la borne inférieure et la borne supérieure). Si une performance possède uniquement une spécification alors l'autre est fixée à l'infini.
- *Mesures de test ou Observations :* ce sont des mesures qui sont effectuées sur le circuit.
- *Critères de test :* c'est un sous ensemble de mesures de test qui doivent corréler avec les spécifications afin de pouvoir décider si le circuit passe le test ou non. L'un des buts de la simulation de fautes est de valider l'efficacité de tels critères.
- *Défaut :* mal construction au niveau d'une des parties du circuit qui le rend défaillant.
- *Défaillance ou panne :* l'effet d'un défaut.
- *Faute :* circuit conçu pour modéliser un défaut dans le but de simuler ce dernier.
- *Erreur :* fonctionnement anormal produit par une faute.
- *Détection :* procédure permettant de confirmer l'existence d'une faute dans un circuit.
- *Localisation :* s'effectue une fois la faute est détectée. C'est une procédure qui permet de déterminer l'élément défectueux qui a causé la défaillance (ou la panne).
- *Diagnostic :* détermination de la cause produisant la faute (ou le dysfonctionnement du circuit).
- *Stimulus :* signal appliqué à l'entrée d'un circuit.
- *Signature de faute :* caractérisation d'une faute résultant d'une défaillance.
- *Points de test :* les différentes connections ou nœuds d'un circuit dans lesquelles il est possible de brancher un instrument de mesure.
- *Équipement Automatique de Test (ATE) :* ensemble intégré de logiciels et matériels utilisé pour réaliser la procédure de test.
- *CUT (Circuit Under Test) :* Circuit Sous Test, représente le circuit dans la phase de test.
- *DUT (Design Under Test) :* Dispositif Sous Test, c'est une autre nomination du CUT.
- *Paramètres process :* les paramètres liés au procédé de fabrication et la physique des composants.
- *Paramètres du circuit :* les paramètres liés à la géométrie du design (résistance, capacité, largeur d'un transistor, etc.).
- *Paramètres du design :* appelés aussi les performances. Représentent les paramètres permettant de décider si le CUT est fonctionnel ou non.
- *Paramètres de test :* appelés aussi critères de test, ils peuvent être une partie des paramètres du design ou bien d'autres paramètres pouvant aider à décider si le circuit passe le test ou non.

Les fautes

Une description de la terminologie sur les fautes, en particulier, les classes de fautes est donnée dans [8]. Nous ajoutons dans ce qui suit une terminologie qui concerne en particulier les types de fautes.

- *Faute paramétrique :* la variation d'un paramètre physique ou géométrique du circuit sous test, qui cause la violation d'au moins une des spécifications de ce circuit.
- *Faute catastrophique :* les fautes causées par une variation brusque ou bien large d'un paramètre (exemple, court-circuit, circuit ouvert).
- *Faute simple :* les fautes causées uniquement par un seul paramètre ou composant du CUT à la fois.
- *Fautes multiples :* les fautes causées par plusieurs paramètres ou composants du CUT à la fois.
- *Fautes permanentes :* les fautes qui ne peuvent pas être réparées (exemple, court-circuit, circuit ouvert).
- *Fautes intermittentes :* les fautes qui paraissent temporairement (exemple, faux contact sensible aux vibrations).
- *Fautes équivalentes :* deux fautes sont dites équivalentes s'il n'existe pas des vecteurs de test les distinguant.
- *Faute dominante :* une faute F_i domine une autre faute F_j si tous les tests qui détectent la faute F_i détectent aussi la faute F_j.
- *Réduction de fautes :* La méthode qui permet de réduire l'ensemble des fautes soit par équivalence soit par dominance.
- *Couverture de fautes :* représente le rapport du nombre de fautes détectées sur le nombre de fautes global.

Les différents types de tests

- *Test fonctionnel :* vérification des spécifications du circuit.
- *Test structurel :* il permet de détecter les défauts de fabrication qui peuvent affecter le fonctionnement des circuits.
- *Test paramétrique :* il est utilisé pour vérifier si certains paramètres du circuit sous test sont dans la plage des valeurs requises.
- *Test alternatif :* test qui permet de calculer (ou prédire) les performances du circuit à partir de tests plus simples.
- *Test statique :* vérification des états stationnaires du système sous test.
- *Test dynamique :* vérification des caractéristiques dynamiques du système sous test.
- *Test exhaustif :* vérification de tous les modes d'opération pour tous les types de fautes.
- *Test on-line :* (test en-ligne) le test est effectué pendant le fonctionnement normal du circuit.
- *Test off-line :* (test hors-ligne) le test est effectué en mode test.
- *Test concurrent :* le test est effectué pendant l'opération.
- *Test semi-concurrent :* le test est effectué pendant l'opération mais pendant le temps de repos.
- *Test non-concurrent :* le test est effectué hors-ligne.
- *Test wafer :* le test est effectué au niveau plaquettes (Wafer) pendant la fabrication des circuits intégrés.
- *Test de production :* test qui se fait durant la phase de production des circuits intégrés.
- *Test sur-puce (on-chip) :* un test qui s'effectue dans la puce et à l'aide d'un ensemble de

circuits qui se trouvent à l'intérieur de la même puce.
– *Test in-field :* un test qui s'effectue une fois la puce est dans son application finale.

2.2.2 Étapes de fabrication d'un circuit intégré (IC)

La fabrication d'un circuit intégré (IC) passe par deux étapes principales : l'étape de conception ou de design et l'étape de fabrication ou de manufacturing (Voir Figure 2.1) [63].

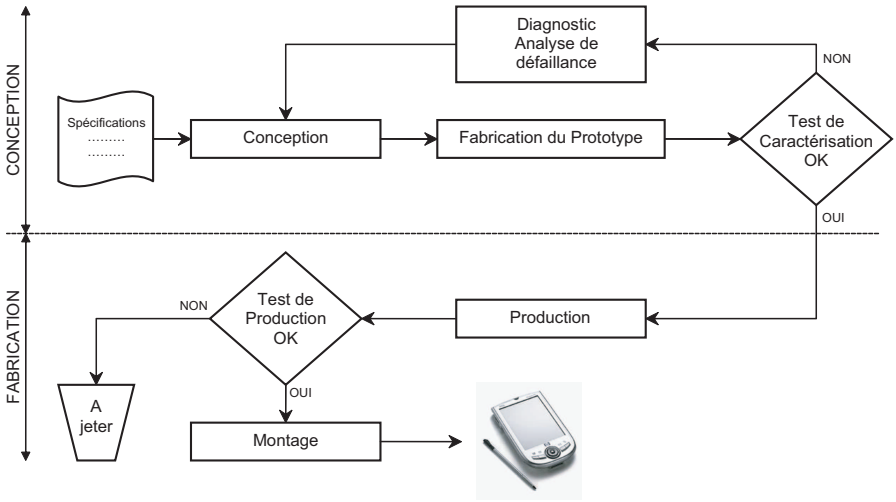


FIG. 2.1 – Les étapes de fabrication d'un circuit intégré (IC) [63].

Dans chacune des étapes, un test doit être effectué. Dans la première étape, le test à effectuer est appelé test de validation ou test de caractérisation. Il s'agit d'un test fonctionnel devant vérifier si les performances du circuit correspondent à celles prévues (du cahier de charges). Si ce n'est pas le cas, un diagnostic doit être fait pour localiser l'erreur de conception ou de fabrication. Dans la deuxième étape, le test à effectuer est un test de production. Son but est de séparer les circuits défaillants des circuits fonctionnels. Mais dans cette étape les circuits ne peuvent pas être réparés, ils seront donc jetés. Cependant le diagnostic est utilisé dans le but d'améliorer le rendement de la chaîne de fabrication.

2.3 Les défauts et les fautes analogiques

Les mal-constructions d'un circuit au delà des variations du process sont appelées *Défauts* et l'effet des défauts sur les caractéristiques électriques d'un circuit intégré déviant au delà des valeurs spécifiées est appelé *Faute* [7, 49]. En d'autres termes, une faute est une conséquence d'un défaut, mais il est possible qu'il n'y ait aucune faute dans un circuit avec défaut. Avant toute tentative de classification de fautes, nous passerons en revue les mécanismes provoquant les fautes analogiques dans les circuits intégrés.

2.3.1 Sources des fautes analogiques

A part les fautes provoquées par le design et qui sont sensées être corrigées après la vérification des prototypes, le process technologique est le responsable principal de la présence des fautes, que soit immédiatement après fabrication ou bien après un temps d'attente qui peut être long.

La fabrication des circuits intégrés est destinée à produire un nombre important de puces identiques. Les compositions ainsi que le layout de n'importe quel couple de circuits (supposés

identiques) fabriqués à partir d'une même ligne de production sont supposées être exactement les mêmes, mais les performances de ces deux circuits peuvent être différentes à cause des variations process. Les sources majeures de ces variations sont [74] :

1. Les erreurs humaines et les défaillances des équipements.
2. Instabilité dans les conditions du process, en terme de changement de valeurs de n'importe quelle variable physique supposée constante.
3. Instabilité du materiel, qui se rapporte à de petites variations dans les compositions chimiques utilisée dans la ligne du process. Par exemple, la contamination chimique venant des résidus d'un autre process.
4. L'hétérogénéité du substrat, incluant les points défectueux, l'imperfection des surfaces, en particulier celles venant de l'implantation des ions.
5. Les non-alignements des masques. Les erreurs dans les translations des alignements sont souvent dominantes parce qu'il y a plusieurs masques qui doivent être parfaitement alignés durant des étapes successives du process.
6. Les points lithographiques, causés par des poussières dans les régions transparentes ou les rayures dans les régions opaques.

Probablement, avec uniquement l'exception des erreurs humaines, les effets de toutes ces sources peuvent être modélisés comme étant des phénomènes aléatoires. La majorité de ceux-ci sont globaux, parce qu'ils affectent, approximativement de la même manière, tous les dispositifs sur une puce (et, dans la majorité des cas, toutes les puces dans une plaquette).

D'autre part, les autres sources de défauts sont de nature locales, et affectent les dispositifs individuellement ou bien une très petite région d'une puce. Ces défauts ou *spots* sont souvent causés par des particules dans l'environnement de fabrication et affectent soit les couches individuelles, soit les interconnections entre deux couches. Les sources des spots les plus fréquentes sont :

1. Les vides (et *extras*) dans le polysilicon ou les lignes métalliques (généralement causés par les spots lithographiques).
2. Trou d'oxyde.
3. Contacts absents (souvent causé par une sous-gravure).
4. Contamination ionique mobile, qui peut être concentrée dans une région particulière du circuit quand il est biaisé.
5. Pipes de conduction locales formées par des imperfections dans la structure du cristal.

Les perturbations du process locales et globales peuvent engendrer des fautes catastrophiques et paramétriques. Les perturbations du process globales peuvent causer la violation des spécifications. Ceci dépend de leur importance et de la sensibilité du circuit pour un paramètre particulier. Normalement ce type de perturbations globales ne vient pas des changements de la topologie, ainsi, la majorité engendrent des fautes paramétriques.

2.3.2 Classification des fautes analogiques

La Figure 2.2 montre une classification générale des fautes analogiques faite en se basant sur [7, 49, 58, 78, 116].

Les perturbations locales et globales du process peuvent causer des fautes *Structurelles* (catastrophiques et non-catastrophiques) et *Paramétriques*.

L'ensemble des fautes structurelles inclut les circuit-ouverts, les court-circuits ainsi que d'autres changements topologiques dans un circuit. Les fautes structurelles peuvent être catégorisées en

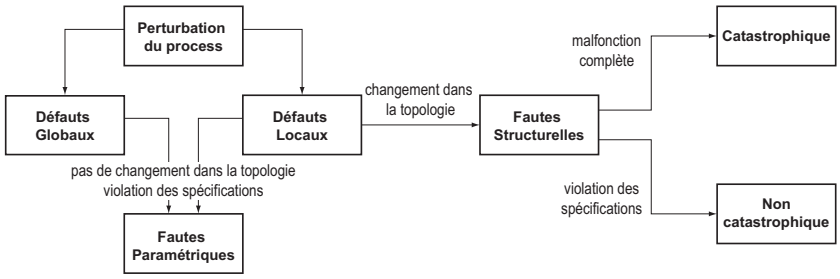


FIG. 2.2 – Classification des fautes analogiques.

fonction des effets des fautes sur les spécifications du circuit. Les fautes qui causent la violation de toutes les spécifications à la fois sont dites *Fautes catastrophiques*. Typiquement, un test DC suffit pour détecter ce genre de fautes. Les fautes qui causent la violation, uniquement, de certaines spécifications sont dites *Fautes non-catastrophiques*.

Les fautes paramétriques représentent les fautes qui ne changent pas la topologie du circuit et elles ont uniquement un impact sur les valeurs des paramètres (par exemple, une déviation de 20% de la valeur typique d'une résistance ou d'une capacité). De telles fautes sont causées par les défauts locaux et globaux. Les fautes paramétriques globales sont dues au contrôle imparfait du process de fabrication des circuits intégrés et peuvent causer la violation des spécifications. Les fautes paramétriques locales, comme les fautes non-catastrophiques, causent une violation de certaines spécifications. Elles sont les conséquence d'un mécanisme local défectueux comme les particules qui augmentent la largeur d'un transistor.

La modélisation et le test des effets de toutes les variations des paramètres process sont essentiels afin d'augmenter la qualité du test. Ces paramètres sont fortement dépendants du type du process, et leur effet sur tout le comportement du circuit dépend des tolérances du design. Malheureusement, le nombre de combinaisons des variations possibles est presque illimité. En revanche, du point de vue de la simulation de fautes, certaines fautes sont plus probables que d'autres [116].

Le Tableau 2.1 regroupe la liste des défauts et des fautes. Les lignes correspondent aux défauts du process de fabrication qui causent les fautes. Les significations des cases du Tableau 2.1 sont données comme suit [116] :

Défaut (cause)	**Faute (effet)**		
	Toutes les spécifications sont vérifiées	Défaillance paramétrique	Défaillance catastrophique
Paramètres process dans leurs intervalles de tolérance	Circuit sans défauts et sans fautes (Circuit fonctionnel)	A2	A3
Paramètres process en dehors de leurs intervalles de tolérance (local)	B1	B2	B3
Court-circuits et circuit-ouverts	C1	C2	C3

TAB. 2.1 – Catégorie des défauts et des fautes analogiques [116].

A2 : Toutes les déviations (ou bien une combinaison de déviations) locales ou globales des paramètres process sont à l'intérieur de leurs intervalles de tolérance mais au moins une des spécifications du circuit n'est pas vérifiée (circuit défaillant). En théorie, ce genre de fautes ne doit pas se produire si le design est robuste. En effet, si tous les paramètres du circuit sont à

l'intérieur de leurs intervalles de tolérance, alors, le circuit fonctionnera correctement. En réalité, le designer ne peut pas simuler toutes les combinaisons des paramètres process pour toutes les conditions possibles.

A3 : Toutes les déviations (ou bien une combinaison de déviations) locales ou globales des paramètres process sont à l'intérieur de leurs intervalles de tolérance mais la fonction du circuit n'est pas bonne (circuit défaillant). En théorie, ce genre de fautes ne doit pas se produire et sont en effet très improbables. Typiquement, si une telle faute se produit, alors elle est due à une erreur dans le design, telle que la *Marge de phase* insuffisante causée par oscillation pour certaines combinaisons imprévues de paramètres process et une température externe ou bien un voltage d'alimentation (*supply voltage*).

B1 : Les paramètres process sortent de leurs intervalles de tolérance mais ils n'influent pas sur les spécifications du circuit. Ce genre de défaut peut éventuellement causer une défaillance dans le circuit et présente un risque probable.

B2 : Les paramètres process sortent de leurs intervalles de tolérance et ils violent les spécifications du circuit. Cette catégorie contient les fautes paramétriques classiques telles qu'une valeur grande d'une résistance qui provoque une insuffisance de la fréquence de coupure.

B3 : Les paramètres process sortent de leurs intervalles de tolérance et ils provoquent un mauvais fonctionnement du circuit.

C1 : N'importe quel court-circuit ou circuit-ouvert (c'est-à-dire qui change la topologie du circuit) qui n'influence pas sur les spécifications du circuit. Ceci peut se produire quand une spécification est redondante ou bien n'est pas spécifiée.

C2 : N'importe quel court-circuit ou circuit-ouvert qui fait que les spécifications du circuit soient légèrement violées.

C3 : N'importe quel court-circuit ou circuit-ouvert ayant effet sur la fonction du circuit. Cette catégorie contient les fautes catastrophiques classiques.

2.3.3 Diagnostic

Le diagnostic est la concatenation de deux processus : la détection d'une faute engendrant la violation des spécifications du circuit et la localisation des dispositifs et des composants à l'origine de la faute. Le premier de ces deux processus est commun aux autres problèmes de test avec des différences liées au niveau de détection de la faute (niveau composant ou structure, niveau fonctionnel, niveau système). Cependant, quand ces deux processus sont combinés dans la procédure de diagnostic, il n'est pas facile de les séparer en procédures clairement distinctes.

Le diagnostic est souvent basé sur les techniques d'analyse de fautes, d'où le besoin d'avoir de bons modèles de fautes. Ces techniques ne sont pas l'objet d'étude de cette thèse, mais elles sont largement expliquées dans [58].

2.3.4 Caractérisation des effets des fautes (Signature de fautes)

Parmi les différents chemins de caractérisation de fautes, c'est-à-dire, les symptômes d'une faute, nous pouvons mentionner comme exemples :

1. Les signaux d'entrée et de sortie du système sous test mesurés à des intervalles de temps ou de fréquences différents (exemple, les voltages dc ou ac sur un ensemble de nœuds ou bien sur des connections externes du système sous test) ;
2. Dans le cas d'une utilisation d'un probe, les différentes mesures d'un courant injecté ;
3. Les paramètres de la fonction de transfert d'un système linéaire. Dans ce cas là, l'ATE doit permettre l'extraction de ces paramètres à partir des échantillons des signaux d'entrée et de sortie ;

4. L'énergie rayonnée mesurée par les détecteurs infrarouge.

Afin de faciliter la détection et la caractérisation de fautes, l'une des idées classiques du test analogique est basée sur la réponse du circuit et l'analyse d'une signature. Une signature représente une forme compacte de l'ensemble des mesures ou combinaison de mesures prises sur le circuit sous test afin de pouvoir le caractériser. La Figure 2.3 résume ce principe. En premier lieu, un signal analogique est appliqué à l'entrée du circuit sous test, puis à partir du signal de la sortie, des paramètres sont calculés (considérés comme une signature du circuit) dans le but de détecter un maximum de fautes.

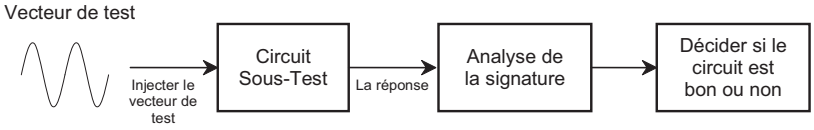


FIG. 2.3 – Analyse de la signature.

Les avantages d'une technique de test basée sur l'analyse d'une signature sont :

1. Plusieurs spécifications du circuit sous test peuvent être calculées en utilisant uniquement la signature.
2. Contrairement aux tests conventionnels où quelques éléments sont ajoutés au circuit pour pouvoir calculer ses spécifications, l'utilisation de la signature nécessite uniquement l'utilisation d'un vecteur de test.
3. Les instruments nécessaires pour générer des vecteurs de test et mesurer la réponse du circuit sont beaucoup plus simples et moins coûteux par rapport à ceux utilisés pour calculer toutes les spécifications.

2.3.5 Simulation de fautes analogiques

La simulation de fautes est utilisée afin de déterminer la qualité des vecteurs de test et d'évaluer une technique de test. La simulation de fautes analogique nécessite de modéliser un circuit avec fautes. Les modèles de fautes représentent les effets des défauts sur le comportement du circuit. La simulation de fautes permet d'évaluer la qualité de l'ensemble des vecteurs de test en calculant les métriques de test analogiques[2]. La simulation de fautes permet aussi de réduire la taille de l'ensemble des vecteurs de test. Elle se caractérise par son efficacité, son temps d'exécution, et par la mémoire utilisée.

La réalisation d'une simulation de fautes analogique est basée sur l'utilisation d'un simulateur de circuits analogiques comme SPECTRE, SPICE et ELDO. Classiquement, les différentes étapes d'une simulation de fautes analogiques sont la simulation du circuit sans fautes (circuit fonctionnel), la sauvegarde des résultats obtenus, l'injection d'une faute dans ce circuit, la simulation du circuit avec la faute (le circuit défaillant) et enfin la comparaison des résultats des deux simulations.

Nous mentionnerons quelques techniques d'injection de fautes analogiques par la suite [8] :

– *Injection de fautes au niveau transistor* : Une présentation d'injection de fautes au niveau transistor et en particulier les fautes de type transitoire a été faite, par exemple, dans [109, 108]. Les fautes de type court-circuit sont injectées en branchant en parallèle une résistance de valeur très petite ou proche de zéro ohm. Ce type d'injections est considéré dans le cadre de cette thèse et il est présenté en détail dans le Chapitre 5.

[2]Rendement, Rendement de test, Couverture de Rendement, Perte de Rendement, Couverture de fautes, Fausse Acceptation et Faux rejet.

– *Injection de fautes au niveau comportemental* : L'injection de fautes dans les blocs de description comportementale analogique a été considérée, par exemple, dans [88] et [130]. Ces fautes sont injectées en modifiant le bloc décrit à base de langages de haut niveau tels que VHDL-AMS [56] ou Verilog-A. La modification de ces blocs est réalisée en changeant les valeurs des paramètres décrivant le comportement, c'est-à-dire en injectant des fautes paramétriques [8].
– *Injection de fautes sous radiations* : Juste à titre d'information, de nombreux travaux [48, 68, 27] présentent des expérimentations d'injection de fautes réalisées sous radiations. Le principe reste rigoureusement le même que pour les circuits numériques [8]. La seule chose qui change est l'analyse des résultats qui est plus complexe à cause de la nature du signal analogique. Toutefois, ce type d'injection n'est réalisable qu'après fabrication du circuit et sort donc du cadre de cette thèse.

2.4 Génération de vecteurs de test

Un vecteur de test est une combinaison des entrées du circuit sous test qui permet de contrôler les fautes à partir des entrées primaires et de les observer sur des sorties primaires. L'opération de la génération des vecteurs de test consiste à définir le programme qui permet de décrire les vecteurs de test spécifiques pour détecter l'ensemble des fautes données dans le but de réduire le coût de test. La Figure 2.4 montre un schéma général résumant la génération et l'application des vecteurs de test [63].

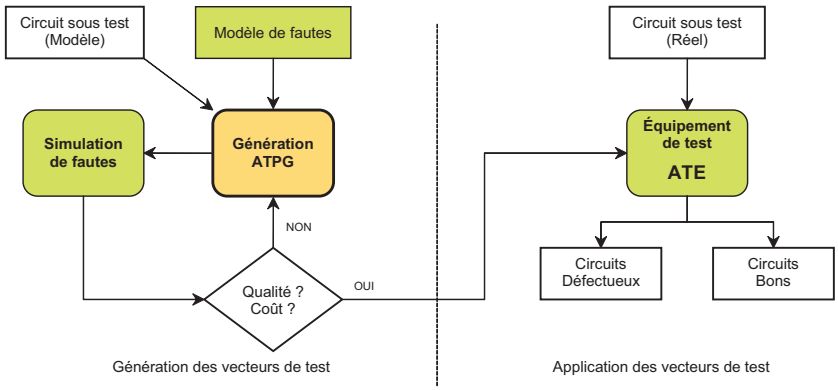


FIG. 2.4 – Génération et application des vecteurs de test [63].

2.4.1 Types de génération de vecteurs de test

Nous distinguons quatre types de génération de vecteurs de test :

1. *Génération manuelle des vecteurs de test* : C'est l'ingénieur de test qui décrit l'ensemble des vecteurs de test à utiliser. L'ensemble de vecteurs de test est défini en fonction de la Couverture de fautes. Généralement, une bonne Couverture de fautes est difficilement atteinte de cette façon.
2. *Génération pseudo-aléatoire des vecteurs de test* : Les vecteurs de test peuvent être générés aléatoirement. Pour le cas des circuits numériques, il suffit de générer des vecteurs binaires aléatoires et obtenir ainsi un ensemble de vecteurs de test aléatoire. Pour ce qui est des circuits analogiques et mixtes, il est possible d'utiliser comme stimuli de test des signaux binaires pseudo-aléatoires pour le test des circuits linéaires mais ces techniques ne trouvent pas encore un intérêt industriel important.

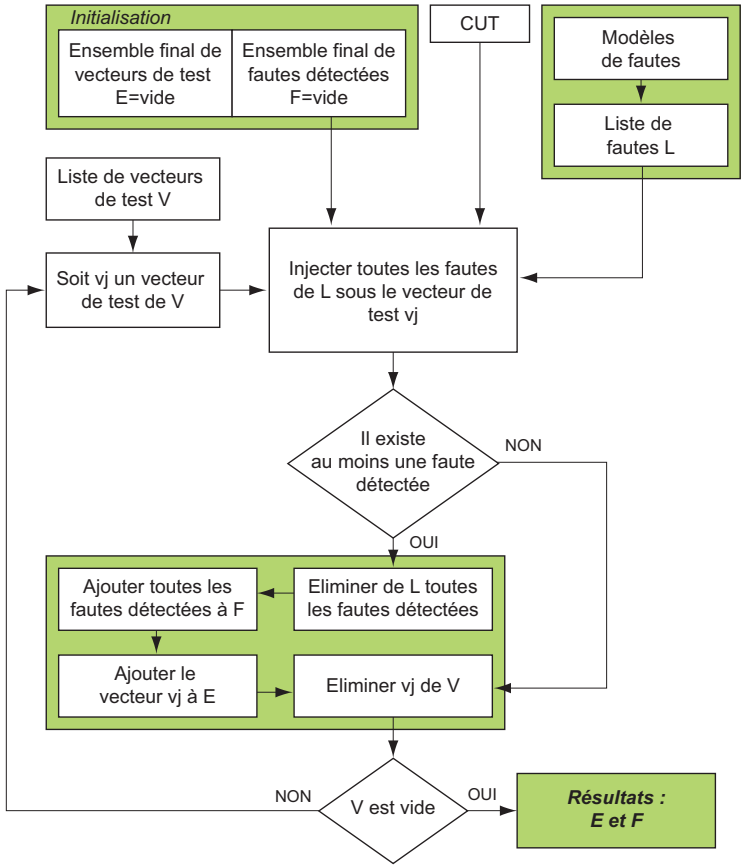


FIG. 2.5 – Principe de base de la génération automatique de vecteurs de test (ATPG).

3. *Génération exhaustive des vecteurs de test* : Dans ce type de test, tous les vecteurs de test possibles sont générés et ne nécessite pas de simulation de fautes. Ce type de génération est impossible pour les circuits numériques ayant un grand nombre d'entrées ainsi que pour les circuits analogiques étant donné le domaine infini des signaux d'entrée.
4. *Génération automatique des vecteurs de test (ATPG)* : Dans ce cas là, la recherche des vecteurs de test se fait de manière à trouver pour chaque faute le vecteur de test qui la détecte. La Figure 2.5 montre le principe de base de la génération automatique des vecteurs de test.

2.4.2 Méthodes pour la génération automatique de vecteurs de test analogiques

Les différentes catégories des méthodes utilisées pour générer automatiquement un vecteur de test analogique seront présentées dans ce qui suit. Plusieurs critères peuvent être utilisés pour catégoriser ces méthodes, par exemple :

- *But du test* : la détection, la localisation, ou le diagnostic.
- *Condition du test* : en-ligne (on-line) ou hors-ligne (off-line).
- *Mode de présentation du comportement du système sous test* : description structurelle ou analytique.
- *Techniques mathématiques utilisées* : on distingue les techniques déterministes, et les statistiques (ou probabilistes).
- *Type du signal d'entrée (vecteur de test)* : signal utilisé pour tester le système sous test.

– *Type des réponses mesurées* : DC, AC, transitoire, impédance, voltage, température, etc.

En pratique, il est difficile d'adopter une telle catégorisation en raison de la grande variété des méthodes disponibles. Cependant, dans ce qui suit, nous adoptons une catégorisation basée sur les modes de représentation du système sous test ; une telle catégorisation est utile pour décrire les domaines d'application des différentes méthodes que nous présenterons dans le chapitre suivant. Nous considérons quatre catégories de méthodes (voir Figure 2.6) [46].

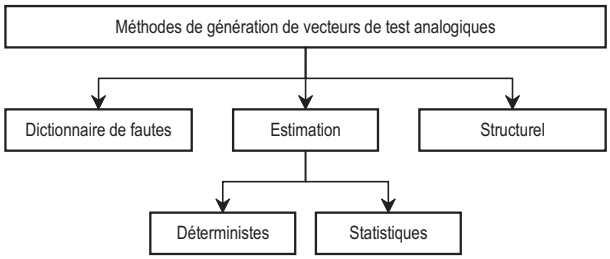


FIG. 2.6 – Catégorisation des méthodes de génération de vecteurs de test analogiques [46].

La première catégorie contient les méthodes basées sur la technique de l'estimation et qui se compose à son tour de deux sous catégories : la catégorie des méthodes déterministes et celle des méthodes probabilistes (ou statistiques). L'utilisation des méthodes déterministes consiste en la description du système sous test par des équations analytiques, la détection de fautes se fait en optimisant un facteur appelé *Facteur de Mérite*, qui caractérise la différence entre la valeur nominale et réelle des caractéristiques ou des mesures du système. Sept méthodes sont détaillées dans [46]. Nous distinguons la méthode du moindre carré, la méthode du vecteur à déviation minimale, la méthode de la programmation quadratique, la méthode de vote, la méthode de la distribution minimale, la méthode des fonctions potentielles. Pour ce qui est des méthodes probabilistes, la méthode de la Probabilité Inverse a été présentée dans [46]. Son principe est le calcul de la probabilité d'occurrence d'une faute pour chaque paramètre du système. La deuxième catégorie englobe les méthodes structurelles qui nécessitent la connaissance de la structure du système. La troisième catégorie réunit les méthodes basées sur le dictionnaire de fautes. Le principe de cette méthode est de comparer les caractéristiques mesurées avec une signature de fautes prédéterminée.

2.5 Conception en vue du test (DFT)

La réalisation la plus classique d'un test consiste à connecter le circuit à une machine chargée d'appliquer des valeurs sur les broches d'entrées, puis de comparer les valeurs apparaissant sur les broches de sortie à des valeurs prédéfinies. Cet équipement de *test externe* peut avoir différentes caractéristiques, selon qu'il s'agisse de tester le circuit sous pointes, de vérifier la fonction d'un circuit après encapsulation, ou encore de réaliser un test paramétrique. De façon générale, cet équipement de test automatique (ATE) est coûteux et ne permet souvent pas de tester le circuit à sa fréquence nominale de fonctionnement.

La conception en vue de test DFT[3] permet de concevoir des puces intégrant en plus à leur circuit principal un ensemble de circuits de test afin de réduire le coût de test et d'améliorer la fiabilité des circuits. Le DFT prend en compte les problèmes de test très tôt dans l'étape de conception des circuits. Actuellement, ces techniques sont très utilisées pour les circuits numériques. Ainsi, plusieurs techniques d'aide à la conception en vue du test sont disponibles tels

[3]Design For Test.

que les techniques de scan (SCAN-PATH), l'auto-test (BIST : Built In Self Test) et le test des frontières (IEEE 1149.1 Boundary Scan Standard). Cette dernière représente la technique DFT la plus largement adoptée [73]. Cette architecture facilite l'application des vecteurs de test et la lecture des réponses de test pour les puces numériques. L'avantage de cette technique est qu'elle possède une interface simplifiée pour les équipements de test facilitant ainsi le test des interconnections entre la puce et la plaquette du testeur.

Les techniques de DFT sont souvent proposées afin de faciliter la contrôlabilité et l'observabilité des nœuds profondément intégrés dans le design. Pour ce qui est des circuits analogiques, les techniques de DFT ont commencé à émerger ces dernières années et notamment pour le BIST où plusieurs techniques ont été développées.

Un BIST est un ensemble de circuits intégrés dans une puce permettant à un circuit de s'auto-tester. Les dispositifs de BIST peuvent permettre de limiter le matériel externe à l'alimentation et à la génération des horloges. Ils peuvent aussi n'être utilisés que pour tester une partie du circuit, dans ce cas, un équipement de test (ATE) reste nécessaire, mais sa complexité peut être réduite, limitant d'autant son coût. Un autre avantage du BIST est la possibilité de réaliser le test à la fréquence nominale de fonctionnement du circuit, puisque les éléments de test sont intégrés dans le circuit et peuvent donc fonctionner à la même vitesse que les circuits réalisant la fonction. Ceci permet de vérifier le bon fonctionnement dynamique du circuit, ce qui prend une importance cruciale avec l'évolution des technologies et l'augmentation des fréquences d'horloge. La Figure 2.7 montre un exemple d'un BIST pour un test pseudo aléatoire.

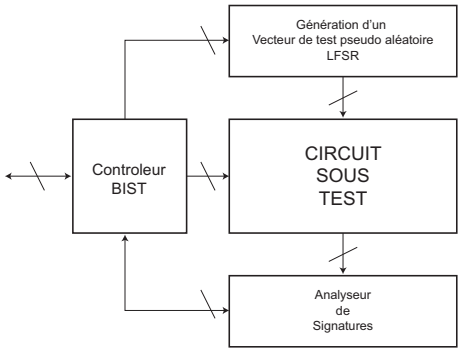


FIG. 2.7 – BIST typique pour un test pseudo aléatoire.

2.6 La qualité d'un test

Les activités de test sont généralement effectuées durant les différentes étapes du cycle de vie des circuits intégrés. La validation des tests est réalisée toujours durant le design du produit, assurant que les spécifications sont satisfaites. Pour le premier échantillon des prototypes des circuits intégrés, la caractérisation des tests est réalisée dans un laboratoire et les prototypes défectueux sont étudiés en détail, par l'analyse des défauts et le diagnostic de fautes menant souvent à la reconstitution des prototypes. Une bonne caractérisation des tests aboutit à un large volume de fabrication et du test de production. Les puces sont testées dans les plaquettes (wafer) avant qu'elles soient coupées et empaquetées. Les puces mises en boîtiers seront testées encore une fois avant qu'elles soient envoyées à la commercialisation. En plus, les puces peuvent être testées sur le champ dans une application donnée.

Trouver les problèmes le plutôt possible dans le cycle de vie d'un circuit intégré est très important. Manifestement, il est plus coûteux de détecter un disfonctionnement sur le champ que durant le test de production, de caractérisation et de validation.

La Figure 2.8 [79] montre comment évaluer la qualité d'un test. Les différentes métriques de test peuvent être directement calculées. En effet, le pourcentage des dispositifs qui passent le test est référencé par le *Rendement du test*[4]. D'une part, parmi les dispositifs qui ne passeront pas le test (ou échoués) certains peuvent être bons (fonctionnels). On parle ici d'une *Perte de rendement*[5]. D'autre part, parmi les dispositifs qui passeront le test certains peuvent être défaillants (ou mauvais). On parle ici d'un *Taux de défauts*[6]. Dans les deux cas, les décisions prises sont fausses dues à l'imprécision des tests. Plus le test est serré plus le nombre des dispositifs qui passeront le test est petit.

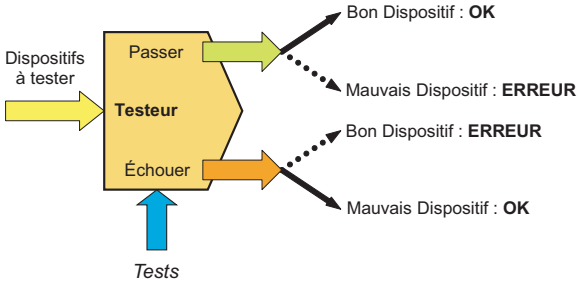


FIG. 2.8 – La qualité d'un test [79].

2.7 Types de test

2.7.1 Test fonctionnel

Le but d'un test fonctionnel est de vérifier le fonctionnement d'un circuit avant de l'envoyer en fabrication. D'une manière classique, le test fonctionnel est effectué pour examiner si le circuit vérifie toutes les spécifications décrites dans le cahier de charges (Figure 2.9) [79]. Il représente une approche de base pour les petites puces analogiques et mixtes. Puisque la fonctionnalité des circuits numériques devient de plus en plus complexe, il apparaît qu'il est pratiquement impossible de vérifier les fonctionnalités d'une puce, en particulier les composants numériques larges tels que les microprocesseurs.

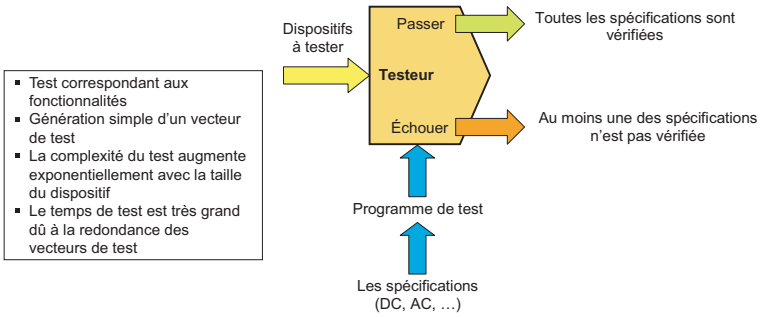


FIG. 2.9 – Principe d'un test fonctionnel [79].

2.7.2 Test structurel

Le test structurel (ou test basé sur les fautes) est largement adopté pour le test des circuits numériques (Figure 2.10) [79]. Dans cette approche, les vecteurs de test sont appliqués pour voir

[4]Test Yield.
[5]Yield Loss.
[6]Defect Level.

s'il existe ou non des fautes dans le circuit, plutôt que de vérifier le fonctionnalité du circuit. Un ensemble de fautes est nécessaire, qui nécessite une connaissance détaillée des défauts potentiels et des mécanismes de panne et de représenter ces derniers par des modèles de fautes décrivant (ou simulant) le mauvais comportement du circuit. Cependant, la simulation de fautes et la génération automatique de vecteurs de test (ATPG) sont utilisées pour décrire l'ensemble de vecteurs de test nécessaires pour un circuit donné afin de détecter les fautes prévues (ou afin d'atteindre une certaine Couverture de fautes). Généralement le test structurel est plus difficile à accomplir que le test fonctionnel, mais il permet d'utiliser un ensemble optimal de vecteurs de test et nécessite un minimum de temps de test, et par conséquent il permet de réduire efficacement le coût de test. Pour les circuits analogiques et mixtes (AMS), le test fonctionnel est le plus favorisé parce qu'il est difficile d'avoir des modèles de fautes adéquats et des outils de simulation de fautes pour pouvoir traiter les fonctions analogiques les plus complexes [79].

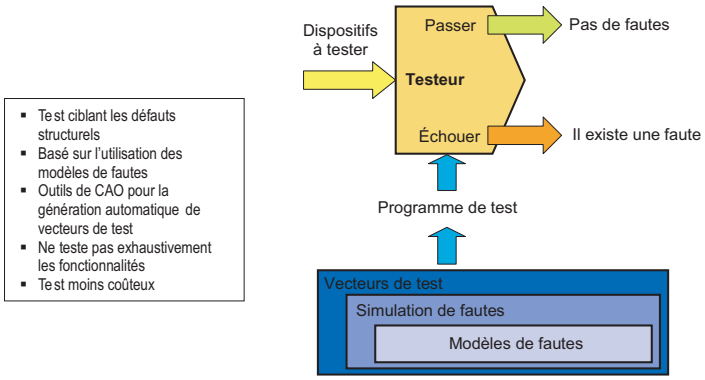


FIG. 2.10 – Principe d'un test structurel [79].

2.7.3 Test paramétrique

Le test paramétrique est une approche qui se situe entre le test fonctionnel et le test structurel. Les tests paramétriques sont souvent appliqués aux circuits pour mesurer un paramètre jugé très important. Par exemple, le cas de la mesure de la consommation d'un circuit (test du courant de consommation) est un test paramétrique bien qu'il suit une approche de test structurel car le but est la détection de défauts.

2.7.4 Test alternatif

Le test alternatif a été récemment proposé pour les systèmes ou les circuits analogiques, mixtes et RF (nécessitant un test fonctionnel) [15, 52, 53, 113]. Les raisons étant que le temps de test des performances est très grand, ajouté au coût très élevé des équipements ATE utilisés. En particulier pour le cas des circuits RF, le nombre de leurs performances est très grand. Dans cette approche, les performances du circuit sous test ne sont pas directement mesurées en utilisant les méthodes conventionnelles. Mais en essayant de prédire à partir d'un ensemble réduit de mesures de test les valeurs des performances. Cette analyse est généralement effectuée en utilisant les techniques de la régression statistique (en particulier, la régression multiple non linéaire pour le cas des circuits RF). Le test alternatif est basé sur l'hypothèse que les variations des performances ainsi que celles des mesures de test dépendent des variations des paramètres physiques du circuit sous test. D'où la possibilité de prédire les variations, et notamment les valeurs, des performances à partir de celles des mesures de test. La Figure 2.11 résume ce principe.

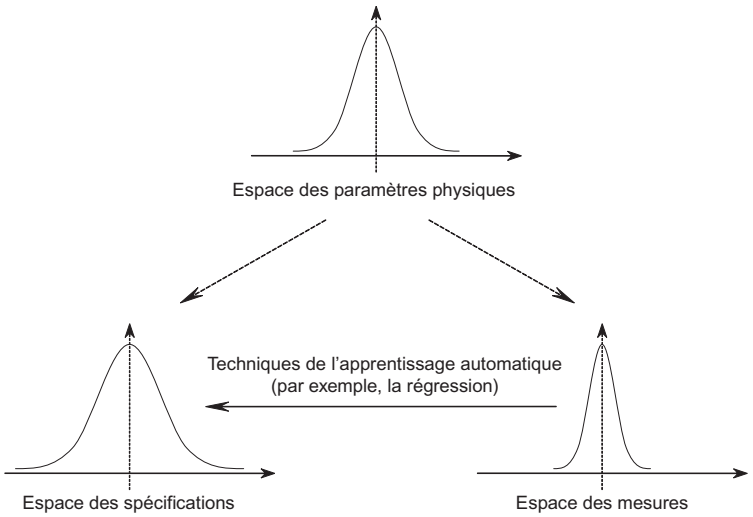


FIG. 2.11 – Principe d'un test alternatif.

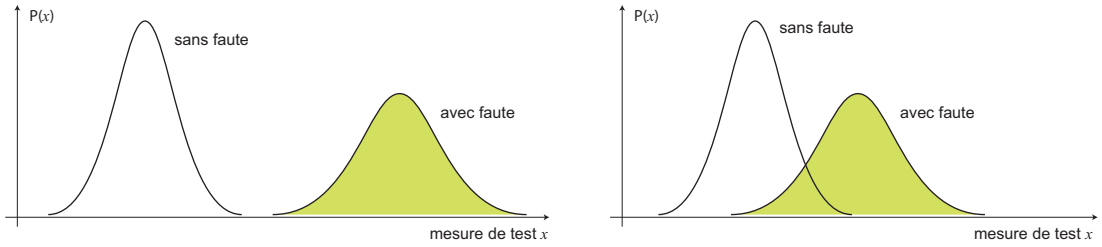
2.8 Les métriques de test

Les différents métriques de test permettant d'évaluer un test analogique seront décrites dans cette section. Certaines métriques sont utiles pour le constructeur (pour évaluer le test de production) et d'autres pour les concepteurs (pour évaluer leur technique de test). Dans le cas des fautes catastrophiques et des circuits numériques le paramètre le plus utilisé est la *Couverture de Fautes F* qui désigne la probabilité de détection des circuits avec faute. Cette probabilité est estimée comme suit :

$$F = \frac{\text{Le nombre de fautes qui sont détectées}}{\text{Le nombre total de fautes}} \tag{2.1}$$

Une application directe des concepts numériques pour le test des circuits analogiques mène à une représentation très pauvre des effets analogiques complexes. Définir comme Couverture de fautes analogiques le pourcentage des fautes catastrophiques détectées par le test est clairement une représentation non significative de la Couverture de fautes, puisque l'univers des fautes doit manifestement inclure les fautes paramétriques. Plus l'univers de fautes est grand plus la métrique de la Couverture de fautes est significative. Pour le cas des circuits analogiques les fautes paramétriques sont significativement les plus importantes, surtout puisqu'elles sont difficilement détectables et puisqu'elles dominent les fautes catastrophiques classiques (exemple, les court-circuits et les circuit-ouverts). Ainsi, un nouveau terme apparaît pour le cas des fautes paramétriques, qui représente la *Détection Partielle* d'une faute [116] du fait que les variations des paramètres peuvent mener à l'obtention d'une mesure ayant des valeurs bonnes ou mauvaises d'une des sorties du circuit à tester. Par le passé, ce genre de fautes a été considéré non détecté. Mais réellement il est partiellement détecté d'où sa nomination. La Figure 2.12 montre respectivement un exemple d'une faute totalement détectée (a) et d'une faute partiellement détectée (b).

Tenant compte du concept de faute partiellement détectable, un ensemble de métriques de test sera utilisé pour pouvoir évaluer un test analogique. La probabilité de détecter une faute dans un circuit numérique est égale soit à 1 (faute totalement détectée) ou à 0 (faute non détectée). Par contre pour les circuits analogiques, cette probabilité prend des valeurs différentes pour chaque faute et qui appartient à l'intervalle [0,1]. En plus, les fautes dans les circuits analogiques peuvent aussi bien être le résultat d'une déviation d'un seul paramètre que celui de déviations de plusieurs paramètres à la fois.



(a) (b)

FIG. 2.12 – Faute (a) totalement détectée et (b) partiellement détectée.

Il est important à noter qu'une bonne valeur de la Couverture de fautes toute seule ne garantie pas une bonne technique de test pour les circuits analogiques, car il se peut que même si la technique détecte tous les circuits défaillants, elle rejette beaucoup de circuits fonctionnels. Donc, pour évaluer correctement la qualité d'un test, il faut évaluer un certain ensemble de métriques de test.

Dans ce qui suit nous définissons quelques métriques de test en plus à la Couverture de fautes qui permettent d'évaluer d'une manière significative la qualité d'un test.

- **Le Rendement (Y) :** C'est la proportion des circuits fonctionnels, il est donné comme suit :
$$Y = \mathbf{P}(\text{Circuit soit fonctionnel})$$
- **Le Rendement de test (Y_T) :** C'est la proportion des circuits qui passent le test, il est donné comme suit :
$$Y_T = \mathbf{P}(\text{Circuit passe le test})$$
- **La Couverture de rendement** (Y_C) (respectueusement **la Perte de rendement (Y_L)**) : C'est la proportion des circuits qui passent (respectueusement qui échouent) le test parmi les circuits fonctionnels. Ces proportions sont données comme suit :
$$Y_C = \mathbf{P}(\text{Circuit passe le test/il est fonctionnel})$$
$$Y_L = \mathbf{P}(\text{Circuit échoue au test/il est fonctionnel}) = 1 - Y_C$$
- **Taux de Défauts (D) :** C'est la proportion des circuits défaillants parmi ceux qui passent le test, il est donné comme suit :
$$D = \mathbf{P}(\text{Circuit soit défaillant/il passe le test})$$

Ces métriques sont suffisantes pour avoir l'information nécessaire sur la qualité d'un test. D'autres métriques typiquement utilisées incluent :

- **La Fausse Acceptation (FA) :** Appelée aussi *erreur de type I*. C'est une autre appellation du Taux de défauts.
- **Le Faux Rejet (FR) :** Appelé aussi *erreur de type II*. Il représente la proportion des circuits fonctionnels parmi ceux qui échouent le test, il est donné comme suit :
$$FR = \mathbf{P}(\text{Circuit fonctionnel/il échoue le test})$$

Ces métriques de test peuvent être calculées pour des fautes catastrophiques et paramétriques. Il existe des relations entre ces métriques, notamment entre la Couverture de fautes et le Taux de défauts. Cette relation[7] a été définie pour la première fois par [129] où les fautes ont été considérées

[7] appelée formule de *Williams et Brown*.

comme équiprobables[8]. Ensuite, d'autres relations à base de celle-ci ont été développées afin de considérer le cas des fautes non-équiprobables (catastrophiques [118] et paramétriques [116]) ainsi que la détection partielle des fautes. Ceci fera l'objet des deux sections suivantes.

2.8.1 Calcul des métriques de test pour des fautes catastrophiques simples

Par le passé, les fautes catastrophiques sont considérées comme des fautes équiprobables et en particulier pour les circuits numériques. C'est-à-dire que la probabilité d'occurrence est la même pour toutes les fautes. Dans ce cas, la Couverture de fautes se calcule en utilisant la formule classique (2.1). Dans [129], une première relation entre le Taux de défauts et la Couverture de fautes, appelée formule de *Williams et Brown*, a été définie comme suit :

$$D = 1 - Y^{1-T} \tag{2.2}$$

où D est le Taux de défauts, Y est le Rendement et $T = F$ la Couverture de fautes donnée par (2.1).

En analysant le layout d'un circuit, on note que les fautes ne peuvent pas être équiprobables. D'où la Couverture de fautes sous l'hypothèse de la non-équiprobabilité des fautes a été définie par [118][9] en démarrant du principe de la formule de *Williams et Brown* présentée ci-dessus et qui est sous la forme suivante :

$$T = \Omega = \frac{\ln \prod_{j=1}^{m}(1 - p_j)}{\ln \prod_{i=1}^{n}(1 - p_i)} \tag{2.3}$$

où p_i représente la probabilité d'occurrence de la $i^{ème}$ faute, n est le nombre de fautes potentielles, m est le nombre de fautes détectées parmi les n.

Cette dernière équation, que nous expliquerons ci-dessous, a été utilisée dans le cas où la distribution d'occurrence des fautes est supposée poissonnienne [31] et elle est utilisée dans un cas réel où la considération de la non-équiprobabilité des fautes est nécessaire (cas des MCMs[10]) par [124, 125].

La probabilité d'occurrence d'une faute est très difficile à calculer pour le cas des fautes catastrophiques et elle se fait à base de l'analyse du layout du circuit sous test. Cependant, pour le cas des fautes paramétriques (généralement pour les circuits analogiques), la probabilité d'occurrence d'une faute représente la probabilité pour que la déviation d'un paramètre induise la violation d'au moins une des spécifications du circuit sous test. Elle se calcule à l'aide de la loi de distribution du paramètre en question, qui est souvent connue.

2.8.2 Calcul des métriques de test pour des fautes paramétriques simples

Une faute paramétrique est typiquement définie comme une déviation d'un paramètre avec un certain taux. Cependant, cette définition n'inclut pas la fonctionnalité du circuit. Dans le cadre de cette thèse, nous définirons une faute paramétrique par la déviation minimale d'un paramètre pour laquelle au moins une des spécifications du circuits sous test est violée. A partir de ce concept, la *probabilité d'occurrence* p_i^{spec} *d'une faute sur le paramètre* i peut être calculée [116]. La Figure 2.13 montre un exemple d'un paramètre ayant une distribution gaussienne où v_i^{spec} représente la faute et p_i^{spec} sa probabilité d'occurrence.

[8] chaque faute a la même probabilité d'occurrence.
[9] sous le nom de *Couverture de Fautes Pondérée*.
[10] MultiChip Modules ou Modules Multi-Puces.

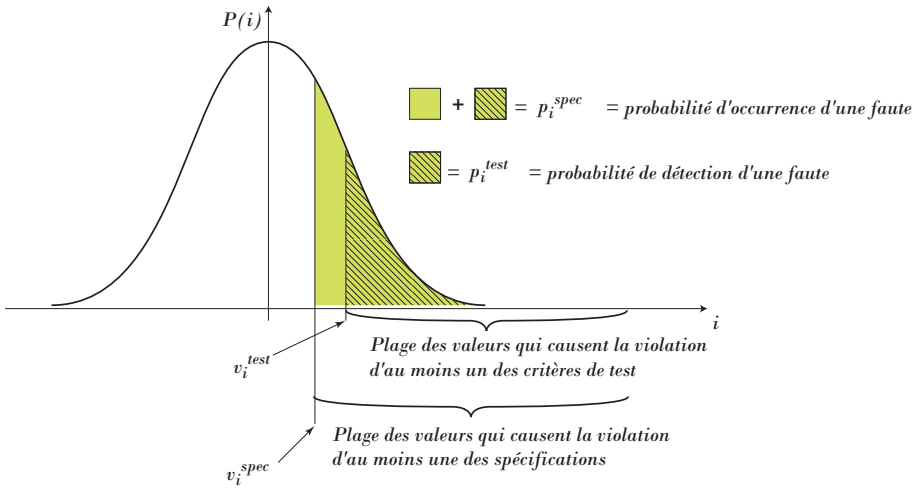


FIG. 2.13 – Probabilité d'occurrence et probabilité de détection d'une faute paramétrique.

D'une autre manière, si v_i^{spec} est la déviation minimale d'un paramètre i pour laquelle au moins une des spécifications du circuit sous test est violée, p_i^{spec} est donc la probabilité pour que ce paramètre prenne une valeur supérieure ou égale à v_i^{spec}. elle se calcule comme suit :

$$p_i^{spec} = \mathbf{P}(i > v_i^{spec}) = \int_{v_i^{spec}}^{+\infty} P(i)\ di \tag{2.4}$$

où $P(i)$ est la densité de probabilité du paramètre i.

De la même manière, la *probabilité de détection* p_i^{test} *d'une faute sur le paramètre* i peut être définie (voir Figure 2.13) qui désigne la probabilité pour qu'un paramètre dévie de sa valeur nominale d'une valeur supérieure ou égale au minimum v_i^{test} nécessaire pour détecter une faute (ou minimum permettant de violer au moins un des critères de test du circuit sous test). Cette probabilité se calcule comme suit :

$$p_i^{test} = \mathbf{P}(i > v_i^{test}) = \int_{v_i^{test}}^{+\infty} P(i)\ di \tag{2.5}$$

où $P(i)$ est la densité de probabilité du paramètre i.

Ces probabilités permettent de définir de nouvelles catégories de fautes qui sont les fautes très probables, probables et improbables. Généralement, les fautes improbables (les fautes pour lesquels $p_i^{spec} \approx 0$) ne sont jamais injectées ce qui permet de réduire la taille de l'ensemble des fautes à injecter.

Le calcul des métriques de test nécessite de connaître la distribution statistique de chaque paramètre du circuit sous test. Donc, en se basant sur les probabilités p_i^{spec} et p_i^{test}, les métriques de test précédemment définies sont calculées comme suit :

$$Y = \prod_{i=1}^{n} (1 - p_i^{spec}) \tag{2.6}$$

$$Y_T = \prod_{j=1}^{m} (1 - p_j^{test}) \tag{2.7}$$

$$Y_C = \frac{G_P}{Y} \tag{2.8}$$

$$D = 1 - \frac{G_P}{Y_T} \tag{2.9}$$

où n est le nombre total de fautes, $m < n$ le nombre fautes parmi les n qui sont détectées par le test (c'est-à-dire qu'il existe m paramètres ayant $p_i^{test} \neq 0$) et,

$$G_P = \prod_{i=1}^{n} (1 - \max(p_i^{spec}, p_i^{test})) \tag{2.10}$$

est la probabilité pour qu'un circuit soit fonctionnel et passe le test. L'utilisation de la fonction max dans (2.10) est expliquée par la Figure 2.14. Deux cas se présentent, le premier représente le cas où $v_i^{test} < v_i^{spec}$ ou $p_i^{test} > p_i^{spec}$ (voir la partie gauche (a1) et (a2) de la Figure 2.14). Le deuxième cas représente le cas où $v_i^{spec} < v_i^{test}$ ou $p_i^{spec} > p_i^{test}$ (voir la partie droite (b1) et (b2) de la Figure 2.14). Donc, $G_{P_i} = 1 - \max(p_i^{spec}, p_i^{test})$, la surface hachurée, représente la probabilité pour que le paramètre i soit fonctionnel et passe le test.

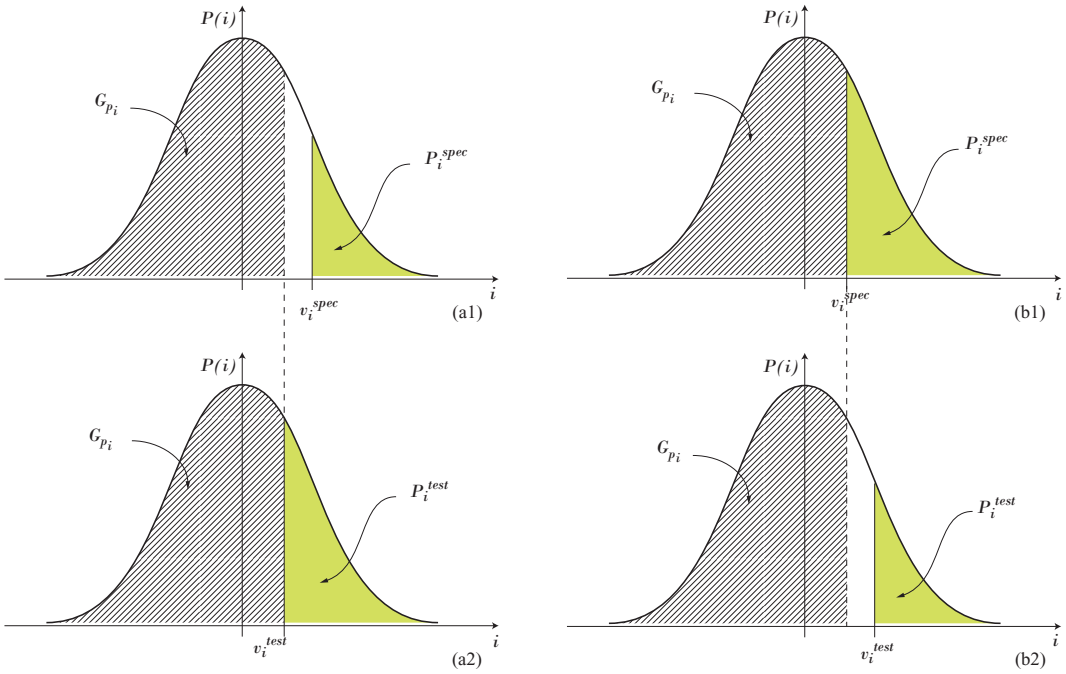


FIG. 2.14 – Probabilité de détection d'une faute (a1 et b1) et probabilité d'occurrence d'une faute (a2 et b2).

Avant de donner l'expression de la Couverture de fautes pour le cas des fautes paramétriques non-équiprobables, nous allons analyser l'Equation (2.3). C'est-à-dire étudier comment elle a été trouvée. Il est à noter que celle-ci est calculée sous l'hypothèse que la Couverture de rendement est toujours égale à 1 ($Y_C = 1$). Ceci signifie que la notion de détection partielle n'est pas considérée. Seule la probabilité d'occurrence d'une faute est considérée. Nous allons reproduire la démonstration de [118] liant le Taux de défaut D à la Couverture de fautes et le Rendement, puis nous allons refaire cette démonstration en tenant compte de la détection partielle des fautes afin de trouver la forme de la Couverture de fautes. Soit A l'événement que le circuit est fonctionnel

et B l'événement que le circuit passe le test. Alors,

$$\begin{aligned} D = \mathbf{P}(\overline{A}/B) &= 1 - \mathbf{P}(A/B) && (2.11)\\ &= 1 - \left(\mathbf{P}(B/A) \cdot \frac{\mathbf{P}(A)}{\mathbf{P}(B)}\right) && (2.12)\\ &= 1 - \left(Y_C \cdot \frac{\mathbf{P}(A)}{\mathbf{P}(B)}\right) && (2.13)\\ &= 1 - \frac{\mathbf{P}(A)}{\mathbf{P}(B)} && (2.14)\\ &= 1 - \frac{Y}{Y_T} && (2.15) \end{aligned}$$

On sait que $\forall x \in \mathbb{R}^+, x = \alpha^{log_\alpha(x)}$. Donc on peut écrire (2.15) comme suit :

$$\begin{aligned} D &= 1 - Y^{\log_Y(\frac{Y}{Y_T})} && (2.16)\\ &= 1 - Y^{\log_Y(Y) - \log_Y(Y_T)} && (2.17)\\ &= 1 - Y^{1 - \log_Y(Y_T)} && (2.18)\\ &= 1 - Y^{1 - \left(\frac{\log_e Y_T}{\log_e Y}\right)} && (2.19)\\ &= 1 - Y^{1 - \left(\frac{\ln Y_T}{\ln Y}\right)} && (2.20) \end{aligned}$$

En comparant l'Equation (2.20) avec l'Equation de Williams et Brown (2.2), nous concluons que la Couverture de fautes peut être écrite comme suit (on remplacera T par Ω pour différentier entre la Couverture de fautes sous l'hypothèse de l'équiprobabilité et celle de la non-équiprobabilité) :

$$\Omega = \frac{\ln Y_T}{\ln Y} \tag{2.21}$$

Y_T est donné par (2.7) et Y est donné par (2.6). En les remplaçant dans (2.21), nous trouverons l'Equation (2.3), qui représente la Couverture de fautes tenant compte de la non-équiprobabilité des fautes (appelée Couverture de fautes pondérée) donnée par [118].

Nous allons refaire maintenant la même démonstration, mais cette fois-ci, en plus à la probabilité d'occurrence des fautes, nous allons tenir compte aussi de la détection partielle des fautes. Nous allons nous intéresser au cas où ($v_i^{test} < v_i^{spec}$), qui engendre des déviations de paramètres détectés comme fautes, alors qu'en réalité elles ne violent aucune des spécifications du circuit (c'est-à-dire qu'elles ne sont pas des fautes). Ceci élimine donc l'hypothèse d'une Couverture de rendement Y_C est toujours égale à 1. C'est-à-dire qu'il peut y avoir des circuits qui sont fonctionnels mais qui ne passent pas le test. Donc,

$$\begin{aligned} D &= 1 - \left(\mathbf{P}(B/A) \cdot \frac{\mathbf{P}(A)}{\mathbf{P}(B)}\right) && (2.22)\\ &= 1 - \left(Y_C \cdot \frac{Y}{Y_T}\right) && (2.23) \end{aligned}$$

Nous allons donc introduire le logarithme afin d'essayer de l'écrire sous la forme de l'équation

de Williams et Brown.

$$
\begin{aligned}
D &= 1 - Y^{\log_Y(Y_C \cdot \frac{Y}{Y_T})} && (2.24)\\
&= 1 - Y^{\log_Y \frac{Y}{\left(\frac{Y_T}{Y_C}\right)}} && (2.25)\\
&= 1 - Y^{1-\log_Y \frac{Y_T}{Y_C}} && (2.26)\\
&= 1 - Y^{1-\frac{\ln \frac{Y_T}{Y_C}}{\ln Y}} && (2.27)\\
&= 1 - Y^{1-\Psi} && (2.28)
\end{aligned}
$$

En comparant avec l'équation de Williams et Brown, nous concluons que Ψ est la Couverture de fautes avec considération des probabilités d'occurrence des fautes ainsi que de la détection partielle des fautes[11]. On note que Ψ peut s'écrire de deux manières différentes, soit comme suit,

$$\Psi = \frac{\ln \frac{Y_T}{Y_C}}{\ln Y} \tag{2.29}$$

ou bien comme suit,

$$
\begin{aligned}
\Psi &= \frac{\ln Y_T - \ln Y_C}{\ln Y} && (2.30)\\
&= \frac{\ln Y_T - \ln\left(\frac{G_P}{Y}\right)}{\ln Y} && (2.31)\\
&= \frac{\ln Y_T - (\ln G_P - \ln Y)}{\ln Y} && (2.32)\\
&= 1 + \frac{\ln Y_T - \ln G_P}{\ln Y} && (2.33)
\end{aligned}
$$

Comme nous pouvons le constater, en comparant l'Equation (2.30) à l'Equation (2.21), le terme de la Couverture de rendement apparaît pour le cas de détection partielle des fautes ($Y_C \leq 1$). Ce cas n'est pas considéré si on tient compte uniquement de la non-équiprobabilité des fautes, car $Y_C = 1$.

Nous allons maintenant écrire Ψ en fonction de p_i^{spec} et de p_i^{test}. Il est à noter que dans ce cas Y_T donné par (2.7) s'écrit comme suit :

$$Y_T = \prod_{j=1}^{n} (1 - p_j^{test}) \tag{2.34}$$

où m est remplacé par n. Ceci vient du fait que la détection partielle des fautes est prise en considération. Car, en plus aux m fautes détectées, il existe d'autres paramètres qui peuvent être détectés comme des fautes alors qu'en réalité, ils ne le sont pas.

Ψ peut donc s'écrire de deux manière différentes. La première est basée sur l'Equation (2.33) et la deuxième sur l'Equation (2.29).

Pour le premier cas, si on remplace Y par (2.6) et Y_C par G_P/Y où G_P est donné par (2.10), nous obtenons :

$$\Psi = 1 + \frac{\ln \prod_{j=1}^{n}(1 - p_j^{test}) - \ln \prod_{k=1}^{n}(1 - \max(p_k^{spec}, p_k^{test}))}{\ln \prod_{i=1}^{n}(1 - p_i^{spec})} \tag{2.35}$$

[11] prendre en considération la détection partielle est une autre manière de prendre en considération la probabilité de détection des fautes.

En passant des produits aux sommes, nous pouvons écrire Ψ comme suit :

$$\Psi = 1 + \frac{\sum_{j=1}^{n} \ln(1 - p_j^{test}) - \sum_{k=1}^{n} \ln(1 - \max(p_k^{spec}, p_k^{test}))}{\sum_{i=1}^{n} \ln(1 - p_i^{spec})} \tag{2.36}$$

Pour le deuxième cas, nous allons utiliser le Y donné par (2.6) et nous allons calculer le $\frac{Y_T}{Y_C}$ en fonction de p_i^{spec} et de p_i^{test}.

$$\frac{Y_T}{Y_C} = \frac{Y_T}{\left(\frac{G_P}{Y}\right)} \tag{2.37}$$

$$= \frac{Y_T \cdot Y}{G_P} \tag{2.38}$$

Donc,

$$\frac{Y_T}{Y_C} = \frac{\prod_{j=1}^{n}(1 - p_j^{test}) \cdot \prod_{i=1}^{n}(1 - p_i^{spec})}{\prod_{k=1}^{n}(1 - \max(p_k^{spec}, p_k^{test}))} \tag{2.39}$$

Supposons maintenant que les paramètres sont classés en deux catégories. La première catégorie des paramètres (classés du 1^{er} au $n_1^{ème}$) sont ceux pour lesquels $p_i^{test} \leq p_i^{spec}$ et la deuxième catégorie des paramètres (classés du $(n_1 + 1)^{ème}$ au $n^{ème}$) sont ceux pour lesquels $p_i^{spec} < p_i^{test}$. Donc, Y, Y_T et G_P peuvent être écrits comme suit :

$$Y = \prod_{i=1}^{n_1}(1 - p_i^{spec}) \cdot \prod_{i=n_1+1}^{n}(1 - p_i^{spec}) \tag{2.40}$$

$$Y_T = \prod_{j=1}^{n_1}(1 - p_j^{test}) \cdot \prod_{j=n_1+1}^{n}(1 - p_j^{test}) \tag{2.41}$$

$$G_P = \prod_{i=1}^{n_1}(1 - p_i^{spec}) \cdot \prod_{j=n_1+1}^{n}(1 - p_j^{test}) \tag{2.42}$$

Alors, (2.39) peut être écrite comme suit :

$$\frac{Y_T}{Y_C} = \frac{\prod_{j=1}^{n_1}(1 - p_j^{test}) \cdot \prod_{j=n_1+1}^{n}(1 - p_j^{test}) \cdot \prod_{i=1}^{n_1}(1 - p_i^{spec}) \cdot \prod_{i=n_1+1}^{n}(1 - p_i^{spec})}{\prod_{i=1}^{n_1}(1 - p_i^{spec}) \cdot \prod_{j=n_1+1}^{n}(1 - p_j^{test})}$$

$$= \frac{\prod_{j=n_1+1}^{n}(1 - p_j^{test}) \cdot \prod_{i=1}^{n_1}(1 - p_i^{spec})}{\prod_{j=n_1+1}^{n}(1 - p_j^{test}) \cdot \prod_{i=1}^{n_1}(1 - p_i^{spec})} \cdot \prod_{j=1}^{n_1}(1 - p_j^{test}) \cdot \prod_{i=n_1+1}^{n}(1 - p_i^{spec})$$

Donc,

$$\frac{Y_T}{Y_C} = \prod_{j=1}^{n_1}(1 - p_j^{test}) \cdot \prod_{i=n_1+1}^{n}(1 - p_i^{spec}) \tag{2.43}$$

En fait, (2.43) représente le produit de tous les paramètres ayant $p_i^{test} \leq p_i^{spec}$ par le produit de tous les paramètres ayant $p_i^{spec} < p_i^{test}$. Donc (2.43) peut être écrit comme suit :

$$\frac{Y_T}{Y_C} = \prod_{i=1}^{n}(1 - \min(p_i^{spec}, p_i^{test})) \tag{2.44}$$

En remplaçant (2.44) dans (2.29), nous obtenons :

$$\Psi = \frac{\ln \prod_{i=1}^{n}(1 - \min(p_i^{spec}, p_i^{test}))}{\ln \prod_{i=1}^{n}(1 - p_i^{spec})} \tag{2.45}$$

Et en passant aux sommes, nous obtenons, ainsi, la formule da la Couverture de fautes donnée par [116] :

$$F^F = \Psi = \frac{\sum_{i=1}^{n} \ln(1 - \min(p_i^{spec}, p_i^{test}))}{\sum_{i=1}^{n} \ln(1 - p_i^{spec})} \tag{2.46}$$

2.8.3 Calcul des métriques de test sous des déviations multiples des paramètres process

Le cas des déviations multiples nécessite une modélisation statistique du process de fabrication. Cette modélisation peut être faite à partir de données obtenues, par exemple, en utilisant une simulation Monte Carlo. Si la densité de probabilité des performances et des critères de test des circuits sous test est connue à priori, alors les métriques de test analogiques sont calculées comme suit :

$$Y = \int_A f_S(s)\ ds \tag{2.47}$$

$$Y_T = \int_B f_T(t)\ dt \tag{2.48}$$

$$Y_C = \frac{\int_A \int_B f_{ST}(s,t)\ ds\ dt}{Y} \tag{2.49}$$

$$D = 1 - \frac{\int_A \int_B f_{ST}(s,t)\ ds\ dt}{Y_T} \tag{2.50}$$

où $A = (A_1, \cdots, A_n)$ est le vecteur des spécifications, $B = (B_1, \cdots, B_m)$ est le vecteur des limites de test, $\int_A$ représente $\int_{A_1} \cdots \int_{A_n}$, $\int_B$ représente $\int_{B_1} \cdots \int_{B_m}$, $f_S(s) = f_S(s_1, \cdots, s_n)$ est la densité de probabilité conjointe des performances, $f_T(t) = f_T(t_1, \cdots, t_m)$ est la densité de probabilité conjointe des critères de test et $f_{ST}(s,t) = f_{ST}(s_1, \cdots, s_n, t_1, \cdots, t_m)$ est la densité de probabilité conjointe des performances et des critères de test.

Les estimateurs de ces métriques pour un échantillon de N circuits générés par exemple en utilisant la simulation Monte Carlo sont donnés comme suit :

$$\hat{Y} = \frac{\text{Nombre de circuits qui sont fonctionnels}}{N} \tag{2.51}$$

$$\hat{Y}_T = \frac{\text{Nombre de circuits qui passent le test}}{N} \tag{2.52}$$

$$\hat{Y}_L = \frac{\text{Nombre de circuits qui sont fonctionnels et qui échouent au test}}{\text{Nombre de circuits qui sont fonctionnels}} \tag{2.53}$$

$$\hat{D} = \frac{\text{Nombre de circuits qui sont défaillants et qui passent le test}}{\text{Nombre de circuits qui passent le test}} \tag{2.54}$$

Il est à noter que la métrique *Couverture de fautes* n'est pas définie pour le cas des déviations process, car les circuits défaillants ne sont pas générés par injection de fautes mais à base de la simulation. Les métriques de test dans le cas des déviations process serviront à fixer les limites des critères de test. Ensuite, ces limites seront utilisées dans le calcul de la Couverture de fautes. Cette dernière sera calculée par la formule classique (2.1) sui représente le rapport du nombre de fautes détectées sur le nombre total de fautes injectées.

2.9 Conclusion

Nous avons présenté dans ce chapitre quelques concepts de base du test analogique. Ceci permettra dans les prochains chapitres de comprendre les techniques de test analogiques étudiées et intégrées dans la plateforme CAO développée dans le cadre de cette thèse. Nous présenterons dans le prochain chapitre un état de l'art sur les outils de CAT analogique, en se focalisant sur la simulation de fautes et la génération et l'optimisation de vecteurs de test analogiques.

Chapitre 3

Etat de l'art des outils de CAT analogique

3.1 Introduction

Dans ce chapitre, nous allons présenter brièvement l'état de l'art des outils de CAT analogique. Ceci nous permettra par la suite de décrire la plateforme de la CAO développée dans le cadre de cette thèse tout en le situant dans cet état de l'art. Nous distinguerons les outils et les techniques de la simulation de fautes et les outils et les techniques de l'optimisation et la génération de vecteurs de test.

3.2 Simulation de fautes

3.2.1 Environnements CAO pour la simulation de fautes

Plusieurs outils pour la modélisation, l'injection et la simulation de fautes analogiques sont apparus dans la littérature. La majorité de ces outils sont destinés pour une utilisation locale et ne sont pas commercialisés en raison de leur spécificité académique (dite maison). Nous pouvons distinguer les environnements en fonction du niveau de description du circuit pour l'injection de fautes : Comportementale, schématique, layout, etc, et leurs types de fautes considérés : Catastrophiques, paramétriques, simples, multiples, déviation process, etc. Nous distinguons aussi deux types de simulateurs de fautes. Le premier type correspond au cas où le simulateur de fautes modifie directement les paramètres de la *netlist*[1] à simuler. Le deuxième type représente des simulateurs de fautes pour lesquels la netlist est générée une fois une faute a été injectée dans une description du circuit.

Pour le premier, nous citons l'outil DRAFTS [85] dont l'architecture est présentée par la Figure 3.1. La particularité de cet outil est que l'injection de fautes se fait par des procédures permettant de modifier les fonctionnalités des différents blocs d'un circuit. Un autre outil FLYER utilisant des algorithmes de simulation de fautes basées sur la sensibilité est présenté dans [127].

Parmi les premiers outils de simulation de fautes adoptant l'approche présentée dans la Section 2.3.5 il y a FSPICE [92] et AnaFAULT [104, 105] qui est aussi utilisée pour la caractérisation de fautes [106, 117]. Le premier outil utilise le simulateur SPICE et fait partie des premiers outils utilisant le principe d'injection de fautes en modifiant directement la netlist du circuit sous test. Le deuxième outil [105] utilise le simulateur ELDO pour la simulation et l'injection de fautes (catastrophiques et paramétriques). Cet outil utilise aussi le simulateur FSPICE uniquement

[1]Une *netlist* représente une traduction textuelle de la forme schématique du circuit sous test utilisée par les simulateurs pour pouvoir effectuer une simulation.

pour l'injection de fautes. La particularité de cet outil est qu'il permet, à l'aide d'un autre outil LIFT [117], d'extraire une liste réduite de fautes à injecter à partir d'un ensemble contenant toutes les fautes possibles. Ceci permet de réduire le temps de simulation de fautes. La structure de ce simulateur est présentée par la Figure 3.2. La dernière version de cet outil permet d'effectuer une simulation de fautes parallèlement sur un réseau d'ordinateurs[2].

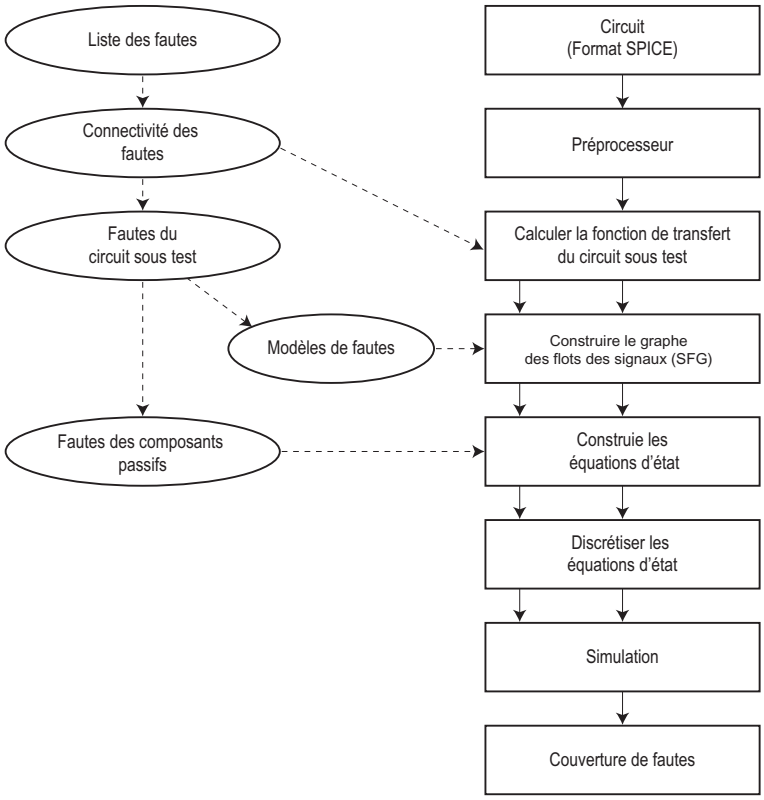


FIG. 3.1 – Architecture du simulateur de fautes DRAFTS [85].

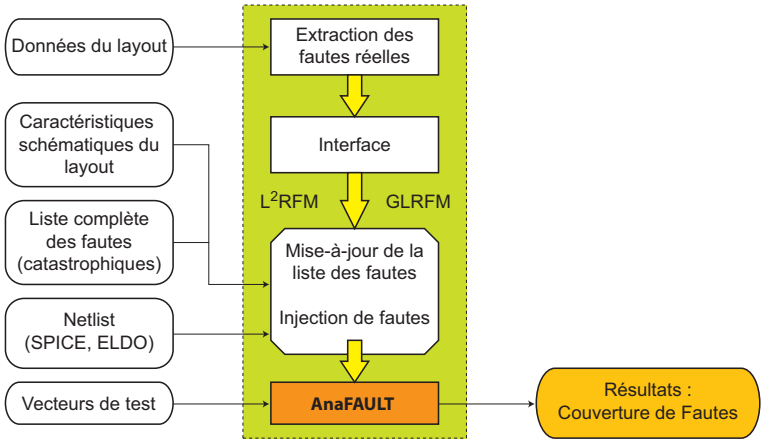


FIG. 3.2 – Simulateur de fautes AnaFault.

Les fautes catastrophiques sous des variations du process ont été aussi considérées par l'environnement de simulation de fautes ANTICS [112] dont l'architecture de base est présentée par la Figure 3.3. Ce simulateur est composé de quatre outils. Le premier outil, MC_RAND permet

[2]Workstation Cluster Environment.

de contrôler les simulations de type Monte Carlo. Le deuxième ANAFINS permet de faire l'injection de fautes. Le troisième outil ANAFAME est utilisé pour effectuer la simulation de fautes et finalement, le dernier outil ANACOV permet donc de faire toute l'analyse post-traitement (ou évaluation de test en terme de Couverture de fautes).

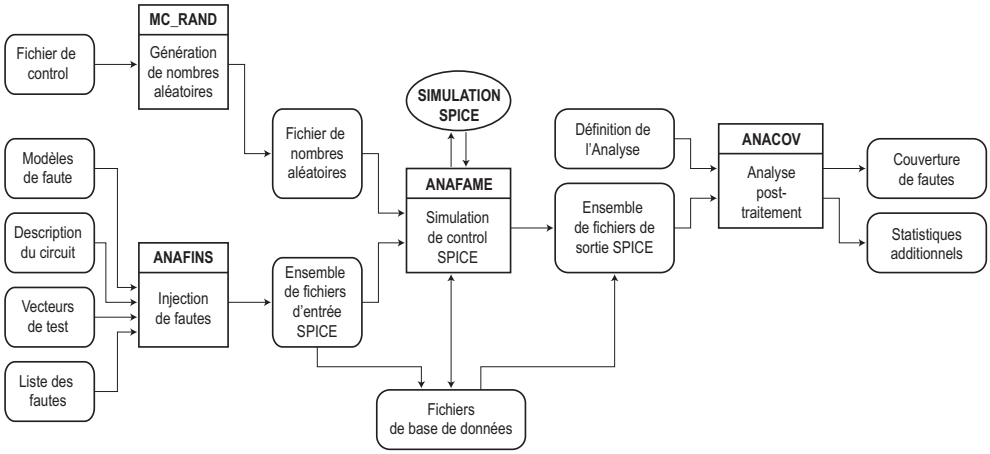


FIG. 3.3 – Environnement de simulation de fautes ANTICS [112].

Dans [133], un simulateur de fautes appelé DOTSS est utilisé pour les fautes catastrophiques, ainsi que pour les fautes paramétriques sous les variations du process. Il est aussi utilisé pour les circuits RF [32, 132]. L'architecture de cet environnement est présentée par la Figure 3.4.

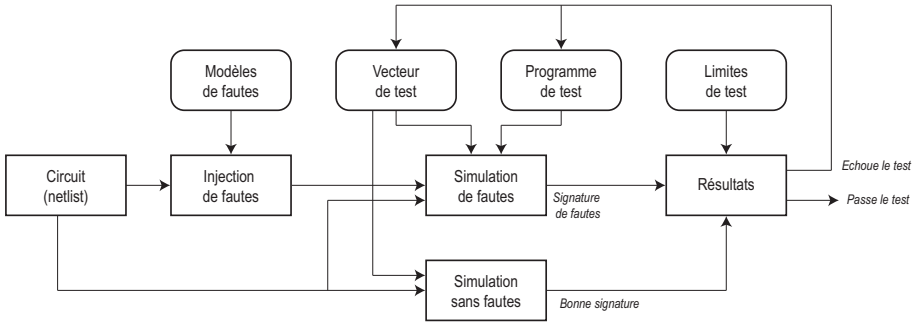


FIG. 3.4 – Architecture du simulateur de fautes DOTSS (Philips) [133].

Un autre simulateur SWITTEST est présenté dans [82]. Il est destiné pour injecter des fautes catastrophiques et paramétriques simples dans les systèmes à capacités commutées. Ce simulateur de fautes utilise le simulateur SWITCAP pour effectuer les simulations dans le domaine temporel et fréquentiel et utilise HSPICE pour effectuer des simulations dans le domaine temporel. L'injection de fautes est faite toujours au niveau interrupteur en utilisant SWITCAP. Le principe de fonctionnement de ce simulateur est montré par la Figure 3.5.

Un outil de simulation de fautes commercialisé est présenté par la Figure 3.6. Ce simulateur permet d'effectuer la simulation de fautes paramétriques et la génération de vecteurs de test utilisant l'analyse de sensibilité et la modélisation statistique [100].

Plusieurs autres outils ont été développés spécialement pour des utilisations et recherches académiques. Le point commun de tous ces outils est qu'ils modifient directement la netlist du circuit nominal pour effectuer l'injection de fautes, ce qui les rend dépendants de la netlist du simulateur utilisé. En revanche, [47] a présenté un outil de simulation de fautes où les modèles de fautes sont ajoutés, avant la génération de la netlist, directement sur le schéma du circuit

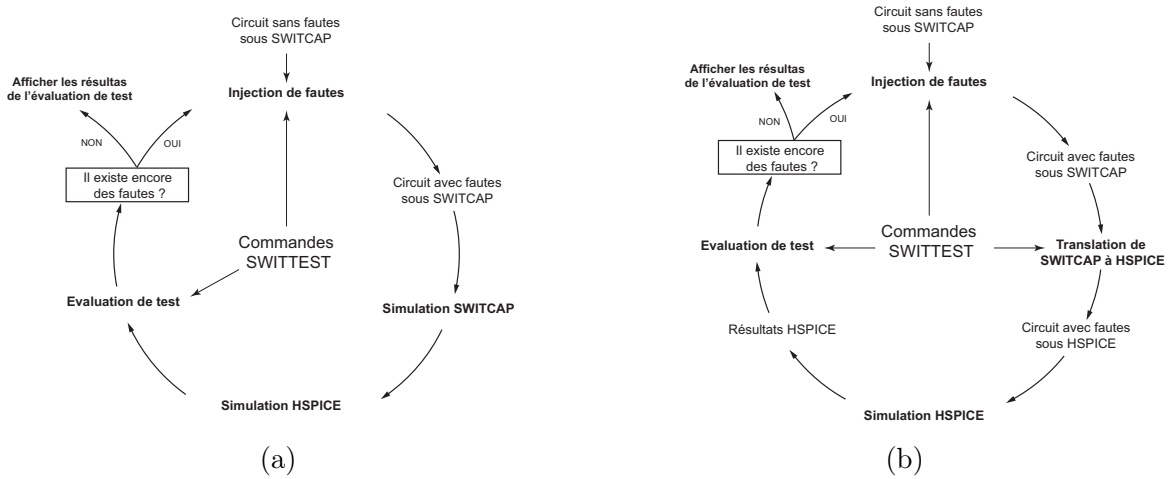


FIG. 3.5 – Simulation de fautes avec SWITTEST utilisant (a) SWITCAP et (b) HSPICE [82].

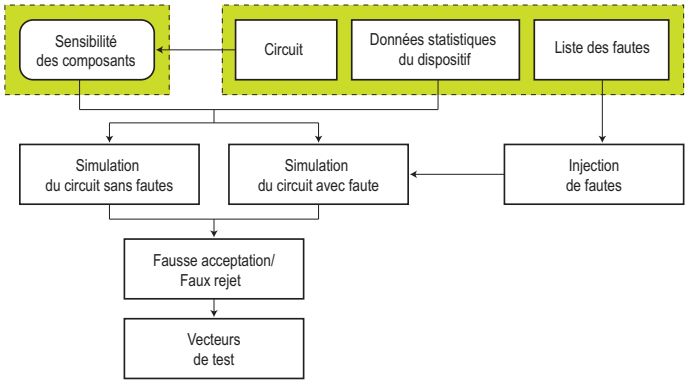


FIG. 3.6 – Environnement du simulateur de fautes dans [100].

sous test (dans Cadence). Les fautes sont ensuite injectées en modifiant les paramètres de chaque modèle de fautes. Ce même principe a été aussi utilisé par [98] dans le simulateur de fautes FaultCraft (pour la simulation de fautes sous Cadence). La netlist est ensuite créée une fois la faute est injectée, d'où l'indépendance de ces outils du simulateur utilisé. La structure de ce dernier simulateur est présentée par la Figure 3.7.

L'outil de simulation de fautes FIDESIM présenté dans le cadre de cette thèse utilise aussi ce même principe à base du langage de description de Fautes FMDL[3] décrit dans [98]. Ce principe est adopté afin de permettre d'injecter automatiquement les modèles de fautes en utilisant des Fichiers de Description de l'Injection FID (voir Section 5.2) qui remplacent les fichiers Scénarios décrits dans [98].

3.2.2 Techniques d'amélioration du temps de simulation de fautes

La simulation de fautes est l'outil principal permettant de trouver le meilleur vecteur de test structurel d'un circuit sous test. D'une manière classique, la simulation de fautes s'effectue en injectant une liste de fautes (catastrophiques ou paramétriques) puis en sauvegardant chaque sortie du circuit sous test correspondant à chaque faute injectée. Toute cette procédure doit être refaite pour chaque vecteur de test considéré. Ce qui permettra ensuite de choisir le meilleur vecteur de test pour lequel une certaine métrique de test, telle que la Couverture de fautes,

[3]Fault Model Description Language.

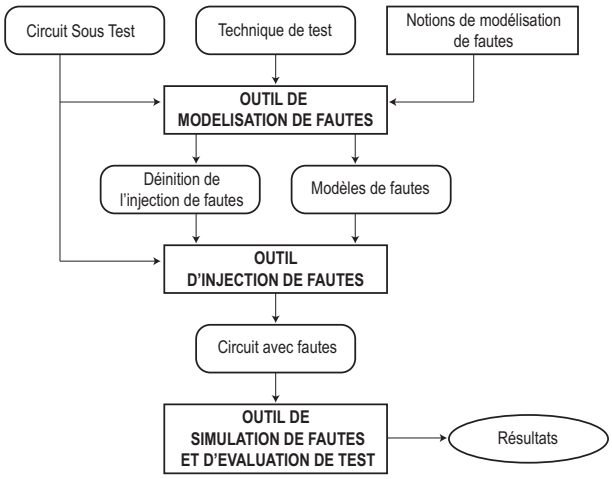


FIG. 3.7 – Structure du simulateur de fautes FaultCraft [98].

est meilleure. D'une autre manière, le meilleur vecteur de test est celui qui permet de détecter le maximum de fautes. Cependant, pour trouver ce vecteur de test, la procédure consistant à injecter toutes les fautes et de simuler le circuit sous test pour chacune d'elles est dans certains cas impossible à effectuer. Pour remédier à ce problème, plusieurs techniques ont été publiées afin de proposer une amélioration du temps de simulation de fautes.

Dans [101] la méthode utilisée pour améliorer la simulation de fautes est basée sur le principe de la simulation de fautes décrit ci-dessus. En revanche, au lieu de sauvegarder les sorties du circuit correspondant à chaque faute, seules les fautes injectées sont sauvegardées, où une faute i représente la valeur minimale R_{i_f} de la résistance R_i (d'un circuit-ouvert ou d'un court-circuit) qui cause la violation d'au moins une des spécifications du circuit sous test. La valeur de R_{i_f} est calculée comme suit :

$$R_{i_f} = \frac{\Delta_{out}}{S^{out}_{R_i}} \tag{3.1}$$

où, Δ_{out} représente les bornes pour lesquelles la sortie *out* reste vérifiée et $S^{out}_{R_i}$ est la sensibilité de la sortie *out* par rapport à la résistance R_i. Puis, une méthode appelée *Méthode du Réseau Adjoint*[4] [43] permettant de calculer toutes les sensibilités $S^{out}_{R_i}$ en parallèle est utilisée. Ainsi, les R_{i_f} peuvent être calculés parallèlement.

D'autres méthodes d'accélération du temps de simulation de fautes sont basées sur la modification des paramètres de la fonction de transfert ou du modèle décrivant le comportement, du circuit sous test, dans le domaine transitoire [13]. Pour les circuits numériques de grande taille (avec plus de 10^5 transistors et nécessitant près de $3 \cdot 10^5$ fautes) qui comportent des parties analogiques et mixtes, [102] propose de décrire tout le circuit sous test par un langage de haut-niveau à l'exception du module de la partie mixte où la faute sera injectée. Ce module sera décrit par un langage tel que Verilog afin de pouvoir lui injecter des fautes en modifiant directement son code. Le même principe est utilisé par [67, 89] pour le cas du langage VHDL-AMS.

Une approche d'accélération de la simulation de fautes dans le domaine transitoire a été présentée dans [50]. Le cas de la simulation de fautes concurrente a été présentée dans [70, 134]. Elle est aussi présentée pour le cas de la simulation dans le domaine transitoire par [66] où les fautes considérées sont catastrophiques simples et dans [57] où les fautes catastrophiques ainsi que paramétriques simples sont considérées et qui est destinée particulièrement pour les circuits non linéaires. La simulation concurrente pour les circuits non linéaires a été aussi abordée dans [131].

[4] Adjoint Method Network.

Finalement, l'amélioration de la vitesse de la simulation de fautes dans le domaine fréquentiel a été présentée dans [128] et dans [119] dans le cas où les déviations paramétriques multiples sont considérées. L'amélioration des simulations de type DC a été décrite par [107].

3.2.3 Critère de Discrimination et de Détection de Fautes

Un critère de discrimination (appelé aussi critère de séparation ou de détection) de fautes doit être utilisé pour pouvoir décider si une faute est détectée ou non. En raison de la nature continue des circuits analogiques, la distinction entre les circuits avec fautes et sans fautes n'est pas claire comme dans le cas des circuits numériques où les effets des modèles de fautes classiques (stuck-at) sont vus comme 0 ou 1 dans le mode d'observation. Pour le cas des circuits analogiques, où les variations des paramètres process du circuit doivent être prises en considération, la détection de fautes s'avère difficile parce qu'une faute peut être soit totalement détectée, partiellement détectée ou bien non détectée (voir Section 2.8). Il est nécessaire de trouver un critère de discrimination permettant de détecter le maximum de fautes. De la même manière que pour les circuits numériques, la détection d'un maximum de fautes est basée sur le choix d'un vecteur de test efficace. Dans ce qui suit, différentes méthodes décrivant les critères de discrimination de fautes seront présentées.

Par détection totale et partielle

L'approche présentée par [29, 30] simplifie le principe de détection de fautes. Son principe est très proche de celui du cas des circuits numériques. Une faute n'est détectée que si toutes les valeurs d'au moins une des spécifications se trouvent à l'extérieur des valeurs acceptables. La Figure 3.8 présente ces différents cas. Dans cette méthode, la spécification du cas de la Figure 3.8(c) seule permet de détecter une faute.

Une autre méthode basée sur ce même principe est présentée par [112]. Elle tient compte des fautes partiellement détectables (Figure 3.8(b)) une fois que toutes les fautes totalement détectables sont déterminées.

Lorsque les fautes totalement détectables sont les seules qui sont prises en considération, la valeur de la Couverture de fautes, dans ce cas, n'est pas significative. Cependant, lorsque la détection partielle est considérée, une Couverture de fautes plus précise et plus significative est obtenue.

Calcul de la Probabilité de Détection de Faute (FDP)

Soit une mesure de test x. Pour un certain type de faute, la Probabilité de Detection de Fautes (FDP[5]) représente la probabilité pour que cette mesure de test avec faute soit en dehors de l'intervalle de tolerance de cette même mesure de test sans faute. Si $f_f(x)$ est la densité de probabilité de x avec une faute alors la FDP est donnée comme suit [63, 65, 64] :

$$FDP = 1 - \int_I f_f(x)\ dx \tag{3.2}$$

où I est l'intervalle de tolérance de la mesure de test x sans faute.

La Figure 3.9 montre les différentes valeurs que peut prendre une FDP. La Figure 3.9(a) représente le cas d'une détection totale d'une faute. La probabilité pour que les valeurs de la mesure de test avec faute correspondante soit en dehors de son intervalle de tolérance sans faute est égale à 1 (faute détectée). Le cas contraire est représentée par la Figure 3.9(d) où la FDP

[5]Fault Detection Probability.

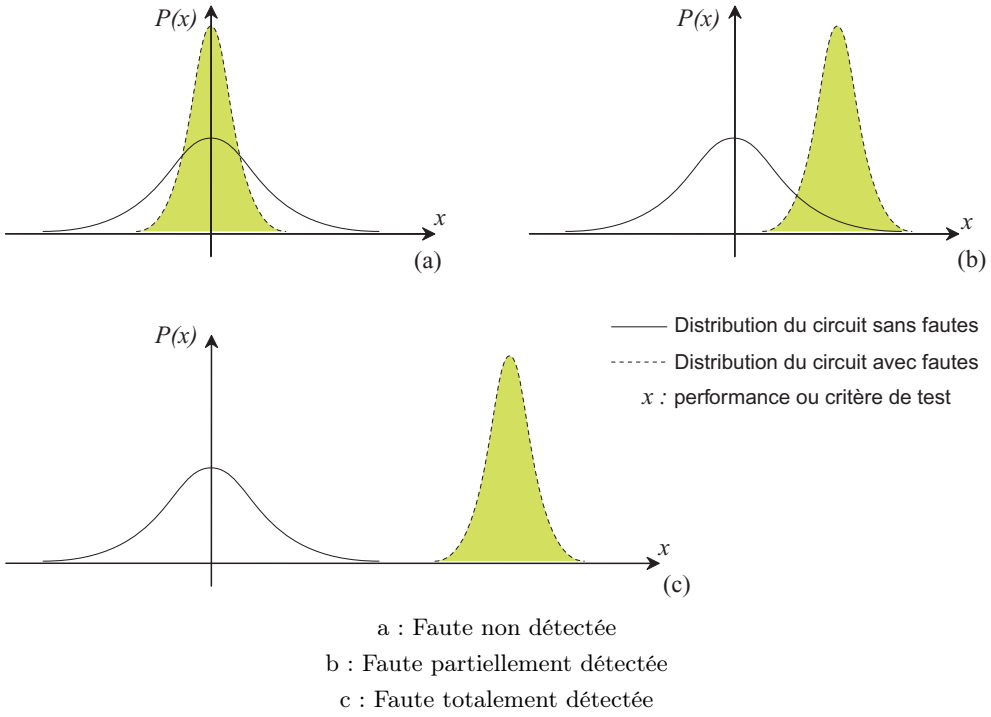


FIG. 3.8 – Performance d'un circuit sans faute et avec faute.

est égale à zéro. C'est-à-dire que la faute n'est pas détectée. Finalement, les deux cas qui restent (Figure 3.9(b) et (c)) représente les cas d'une détection partielle de faute. Donc, la FDP prend des valeurs entre 0 est 1.

Fixation du faux rejet

Une autre technique pour distinguer entre une sortie avec faute d'une sortie sans faute est présentée dans [100]. Le but principal est de minimiser à la fois le faux rejet (probabilité pour qu'un circuit soit considéré comme défaillant alors qu'il est fonctionnel) et la fausse acceptation (probabilité pour qu'un circuit soit considéré comme fonctionnel alors qu'il est défaillant). Cette minimisation multiobjectif est toutefois impossible du fait que les deux paramètres sont dépendants, car la minimisation de l'un peut engendrer la maximisation de l'autre. Le concept utilisé pour résoudre ce genre de problèmes est de trouver un compromis entre ces deux décisions. Dans [100], ce compromis est basé premièrement sur la fixation du faux rejet en fixant une certaine valeur seuil du critère de test. Puis la détection de fautes se fait selon si la valeur du critère de test x est inférieure ou supérieure à ce seuil (Figure 3.10). Ce qui engendre des erreurs de décision (faux rejet et fausse acceptation) différentes pour chaque seuil fixé. En effet, si le circuit a été accepté comme étant sans faute alors qu'en réalité il est défaillant, ceci pose un problème de qualité. Cependant, si le circuit a été accepté avec faute alors que ce n'est pas le cas, ceci réduit le Rendement de test. Le faux rejet (ou le seuil) est donc fixé selon l'utilisateur et selon le type de l'application en question. Le seuil fixé peut être optimisé en choisissant le meilleur vecteur de test. Par exemple, dans le cas de [100] où le faux rejet est fixé à zéro, le vecteur de test est choisi de telle sorte que la fausse acceptation soit la plus petite possible.

Calcul de la Fonction de Discrimination

La fonction de discrimination est un autre moyen permettant de décider si une faute est détectée ou non. La fonction de discrimination utilisée par [133], venant de l'idée utilisée par [100]

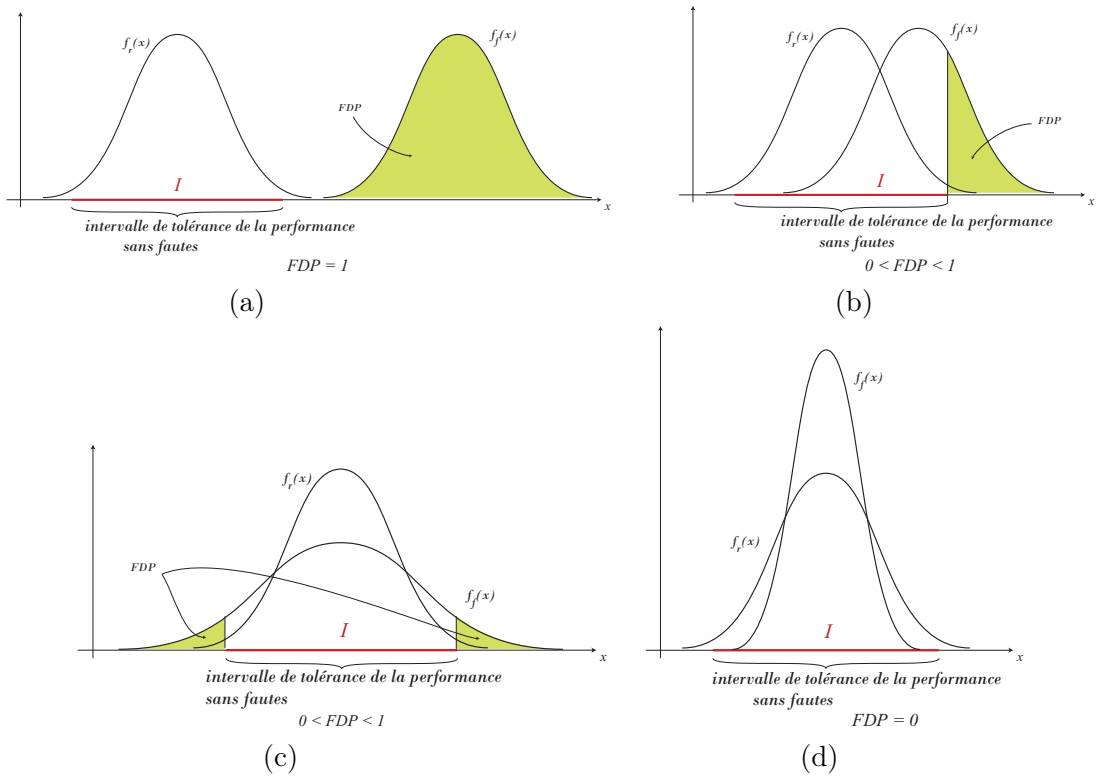


FIG. 3.9 – Les différentes valeurs de la Probabilité de Détection de Fautes FDP.

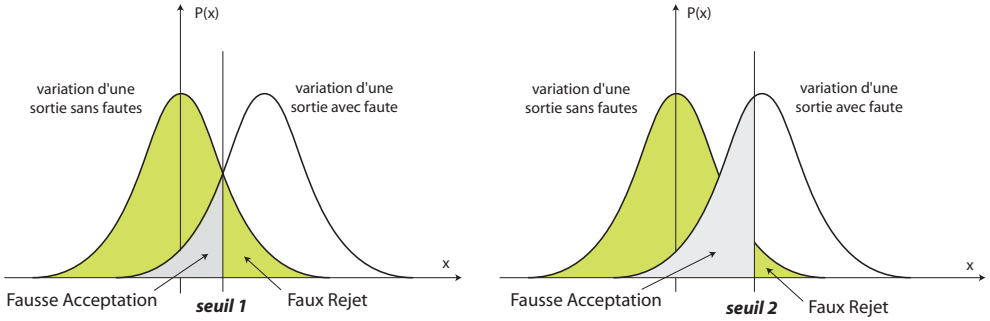


FIG. 3.10 – Fixation du seuil pour le faux rejet et la fausse acceptation.

précédemment décrite, est basée sur le calcul du rapport de vraisemblance[6]. Ce dernier représente le rapport de la densité de probabilité de la sortie d'un circuit sans faute sur la densité de probabilité de la sortie du même circuit avec faute. Le calcul de ce rapport est basé sur la règle de décision de test ϕ_i de Neyman-Pearson pour chaque faute i. Elle est définie comme suit :

$$\phi_i(x) = \begin{cases} 1 \text{ (passe le test)} & \text{si } l_i(x) \geq \lambda_i \\ 0 \text{ (échoue le test)} & \text{si } l_i(x) < \lambda_i \end{cases} . \tag{3.3}$$

où x est la mesure de test, $l_i(x)$ est le rapport de la fonction densité de probabilité $f(x/G)$ d'un circuit fonctionnel sur la fonction densité de probabilité $f(x/F_i)$ d'un circuit avec la faute i. Il est défini comme suit :

$$l_i(x) = \frac{f(x/G)}{f(x/F_i)} \tag{3.4}$$

Le seuil λ_i est calculé en fixant arbitrairement la probabilité de fausse acceptation α_i à une

[6]Likelihood ratio.

certaine valeur α_0 comme suit :

$$\alpha_i = \mathbf{P}(G/F_i) = \int_{\Gamma_G} f(x/F_i)\ dx = \alpha_0 \tag{3.5}$$

où Γ_G est l'intervalle de tolérance de la mesure x sans fautes (voir Figure 3.11). La probabilité $\mathbf{P}(G/F_i)$ représente la probabilité pour qu'un circuit soit accepté comme fonctionnel sachant qu'il est avec la faute i.

Une fois $\mathbf{P}(G/F_i)$ est fixé à α_0, il est donc possible de calculer la valeur de λ_i, qui permettra de décider si la faute i est détectée ou non, à l'aide de l'Equation (3.3).

Enfin, la valeur $\beta_i = \mathbf{P}(F_i/G)$[7] représente la probabilité pour qu'un circuit avec la faute i échoue au test sachant qu'il est fonctionnel. Elle est donnée par :

$$\beta_i = \mathbf{P}(F_i/G) = \int_{\Gamma_{F_i}} f(x/G)\ dx \tag{3.6}$$

où Γ_{F_i} est l'intervalle des valeurs de la mesure x avec la faute i (voir Figure 3.11).

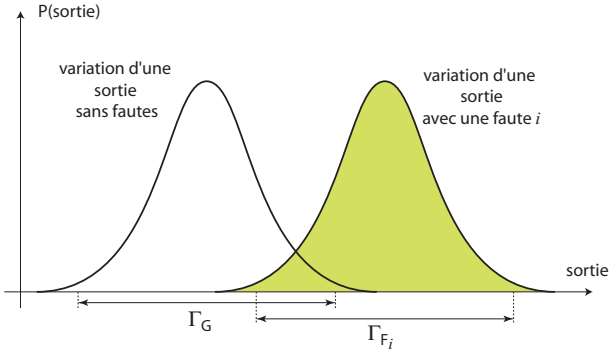


FIG. 3.11 – Les valeurs d'un critère de test d'un circuit sans fautes et avec une faute.

Calcul de la métrique Séparation de Fautes

La *Séparation de Fautes* e est utilisée dans [33] comme critère permettant de décider si une faute est détectée ou non, dans le but de générer des vecteurs de test (voir la première technique de la Section 3.4.6). La Figure 3.12 montre un exemple de la distribution y_g d'un paramètre sans faute et de la distribution y_f de ce même paramètre avec faute. La métrique de séparation de fautes pour une valeur v de ce paramètre est égale à la somme des surfaces $S1(v)$ et $S2(v)$ et se calcule théoriquement comme suit :

$$e(v) = S1(v) + S2(v) \tag{3.7}$$

$$= \mathbf{P}(y_g \geq v) + \mathbf{P}(y_f \leq v) \tag{3.8}$$

La détection d'une faute sur ce paramètre en utilisant cette métrique se fait en cherchant la valeur v_{min} de ce paramètre pour laquelle la séparation de fautes e est minimale, qu'on notera par e_{min}. Cette dernière valeur est située entre 0 et 1. Si elle est très proche de zéro, cela signifie que les deux distributions (avec et sans faute) sont très séparables, donc, la faute est considérée détectée. Si elle est proche de 1, cela signifie que les deux distributions sont très proches l'une de l'autre, donc la faute n'est pas détectable.

[7] Faux rejet.

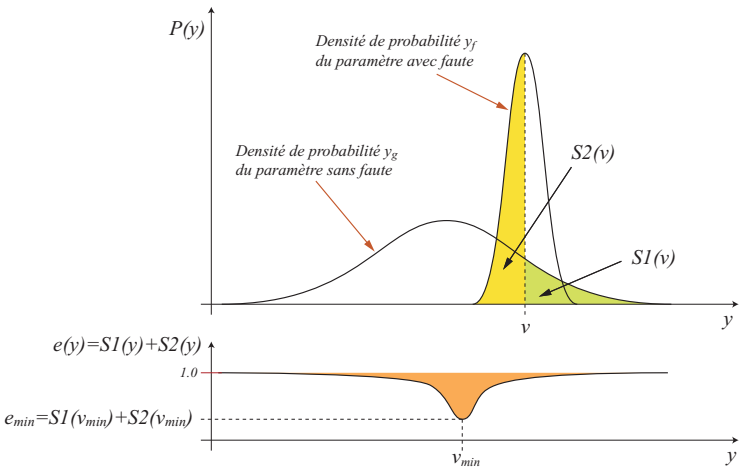


FIG. 3.12 – La distance métrique entre un paramètre sans faute et un paramètre avec faute.

Calcul de la détectabilité d'une faute

La *détectabilité d'une faute* F_i est un paramètre lié aux vecteurs de test f_j, où, $j = 1, ..., N$, et elle est donnée par D_{F_i} comme suit :

$$D_{F_i} = \sum_{j=1}^{N} E_i(f_j) = \sum_{j=1}^{N} |T_g(f_j) - T_{F_i}(f_j)| \tag{3.9}$$

La formule (3.9) représente la somme sur l'ensemble des vecteurs de test des différences entre la mesure de test T_g du circuit fonctionnel et T_{F_i}, celle du circuit avec la faute F_i. Pour un certain seuil D_{Th}, si $D_{F_i} > D_{Th}$, alors la faute F_i est considérée détectée, sinon, si $D_{F_i} \leq D_{Th}$, alors la fautes F_i est considérée non détectée. Ceci s'explique par le fait que si la détectabilité d'une faute est supérieure à un certain seuil D_{Th} alors, il existe un sous-ensemble de l'ensemble des vecteurs de test qui permet de détecter cette faute. Il est à noter que la différence entre la mesure de test du circuit fonctionnel et celle du circuit avec la faute F_i, sous le vecteur de test j est notée par $E_i(j)$ et est appelée *Erreur de la réponse*. C'est à base de cette erreur que les vecteurs de test sont générés dans [60] (voir la deuxième technique de la Section 3.4.6).

3.3 Génération de vecteurs de test fonctionnels

Dans cette section, nous présenterons brièvement quelques techniques de génération et d'optimisation de vecteurs de test fonctionnels analogiques.

3.3.1 Par analyse de la réponse impulsionnelle

L'analyse de la réponse impulsionnelle fait partie de l'une des techniques classiques pour le test fonctionnel des circuits linéaires. Elle est aussi utilisée pour générer des vecteurs de test analogiques et numériques. Dans ce qui suit, nous présenterons la méthode de génération optimale des vecteurs de test analogiques.

Génération de vecteurs de test analogiques par analyse de la réponse impulsionnelle

La technique de [123] permet de générer des vecteurs de test analogiques en maximisant la *distance totale* entre la réponse impulsionnelle (voir Figure 3.13) d'un circuit fonctionnel et celle d'un circuit défaillant.

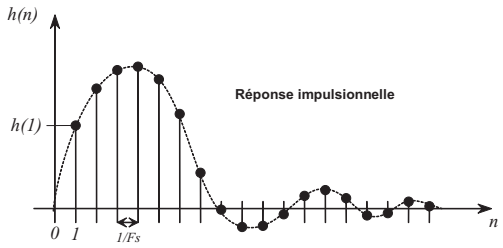


FIG. 3.13 – Réponse impulsionnelle d'un circuit linéaire.

Pour expliquer comment calculer la distance totale entre deux réponses impulsionnelles, nous allons représenter dans ce qui suit les comportements des circuits sous test, dans le cas discret, par leurs réponses impulsionnelles ($h(n)$ pour un circuit fonctionnel et $h'(n)$ pour un circuit défaillant), où n est le $n^{ème}$ point de la réponse considérée.

Pour un signal d'entrée x, le $n^{ème}$ point de la sortie du circuit fonctionnel $y(n)$ et celui de la sortie du circuit défaillant $y'(n)$ s'écrivent par convolution comme suit :

$$y(n) = \sum_{k=0}^{n} x(k)\ h(n-k) \tag{3.10}$$

$$y'(n) = \sum_{k=0}^{n} x(k)\ h'(n-k) \tag{3.11}$$

La différence $\Delta y(n)$ entre ces deux points est donnée comme suit :

$$\begin{aligned} \Delta y(n) &= y(n) - y'(n) && (3.12)\\ &= \sum_{k=0}^{n} x(k)\ [h(n-k) - h'(n-k)] && (3.13)\\ &= \sum_{k=0}^{n} x(k)\ \Delta h(n-k) && (3.14) \end{aligned}$$

Donc, la distance totale $D(x)$ entre le circuit fonctionnel et un circuit défaillant, pour un signal d'entrée x, est donnée comme suit :

$$\begin{aligned} D(x) &= \sum_{i=0}^{n} \Delta y^2(i) && (3.15)\\ &= \sum_{i=0}^{n} \left[\sum_{k=0}^{i} x(k)\ \Delta h(i-k) \right]^2 && (3.16) \end{aligned}$$

Le vecteur de test de cette technique est généré comme suit. Soient deux réponses impulsionnelles, $h(n)$ du circuit fonctionnel et $h'(n)$ du circuit défaillant. Le signal d'entrée x^* pour lequel D est maximal sera donc choisi comme étant le vecteur de test optimal.

$$x^* = \underbrace{\max}_{x}\ D(x) \tag{3.17}$$

Ce problème a été formulé comme un programme quadratique. L'inconvénient de cette technique est qu'il n'existe pas de méthodes permettant de résoudre un tel programme. Dans [123], une heuristique a été présentée pour le circuit sous test considéré afin de pourvoir trouver une solution proche de l'optimale. Les circuits défaillants dans cette technique sont générés par modification des valeurs des paramètres de la réponse impulsionnelle du circuit sous test fonctionnel.

Génération de vecteurs de test numériques par analyse de la réponse impulsionnelle

La Figure 3.14 montre une technique de génération de vecteurs de test numériques pour stimuler des circuits analogiques linéaires LTI[8]. Cette structure est utilisée dans [87] et elle se base sur la réponse impulsionnelle (voir Figure 3.13).

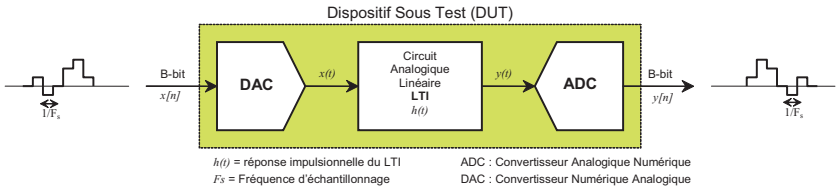


FIG. 3.14 – Structure d'un test numérique pour les circuits analogiques.

Les critères de test représentent un ensemble des échantillons de la réponse impulsionnelle $h(0), h(1), ..., h(n)$ (les points noirs de la Figure 3.13). La Figure 3.15 montre un exemple où les critères de test sont les deux échantillons $h(1)$ et $h(2)$. Sur cette figure les + représentent les circuits fonctionnels et les ∘ représentent les circuits défaillants (par rapport aux spécifications). Trouver un hyperplan (ou un ensemble d'hyperplans) permettant de séparer la région des + et celle des ∘ permettra de bien classifier les circuits. Comme ces hyperplans sont représentés dans un espace d'échantillons de la réponse impulsionnelle à k dimensions, par conséquent, chacun de ces hyperplans est représenté par la formule suivant :

$$\sum_{i=0}^{d-1} c_i \cdot h(i) - c_d = 0 \tag{3.18}$$

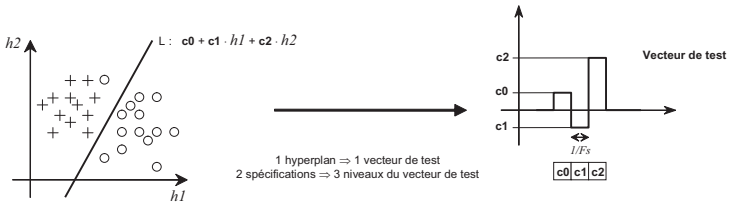


FIG. 3.15 – Vecteur de test dérivé des hyperplans.

La génération de l'ensemble des vecteurs de test est basée sur les hyperplans permettant de séparer les circuits défaillants des circuits fonctionnels. Le problème consiste, donc, à trouver les coefficients c_i $(i = 0, 1, ..., d)$ de ces hyperplans permettant de faire une bonne séparation des circuits. Ce problème peut être résolu par un algorithme du type *Perceptron*. Chaque hyperplan représente un vecteur de test et l'ensemble des coefficients de chaque hyperplan représente les différents niveaux du vecteur de test correspondant (comme exemple, voir Figure 3.15).

L'idée de cette technique vient du fait que la sortie $y(n)$ d'un circuit LTI peut être représentée par une convolution entre les signaux d'entrée $x(n)$ et la réponse impulsionnelle $h(n)$ sous la supposition qu'un circuit LTI est causal (c'est-à-dire que $h(n) = 0$ pour $n < 0$). La sortie peut

[8]LTI : Linear Time-Invariant, circuits linéaires invariants dans le temps.

être écrite ainsi comme suit :

$$y(n) = \sum_{m=0}^{\infty} x(n-m) \cdot h(m) \tag{3.19}$$

$$\approx \sum_{m=0}^{d-1} x(n-m) \cdot h(m),\ n = 0, 1, ..., \infty \tag{3.20}$$

où, $d = F_S/BW$ (F_S est la fréquence d'échantillonnage et BW est la bande passante).

Nous concluons de (3.18) et de (3.19) qu'un hyperplan dans l'espace des réponses impulsionnelles est donné par l'équation suivante :

$$\sum_{m=0}^{d-1} x(n-m) \cdot h(m) - c_d = 0 \tag{3.21}$$

Par conséquent, dans ce cas, le problème de trouver les bons coefficients c_i dans (3.18) des hyperplans de bonne classification est équivalent à trouver les signaux d'entrée $x(n-m)$ dans (3.21) qui seront, par conséquent, les vecteurs de test.

3.3.2 Par analyse de sensibilité

L'analyse de sensibilité est l'une des techniques les plus utilisées pour la génération de vecteurs de test analogiques. Dans [126], la génération de vecteurs de test est basée sur le calcul de la sensibilité pour calculer les intervalles de tolérance Δc_j des paramètres c_j du circuit sous test pour lesquels le circuit reste fonctionnel (à base des performances), où, $j = 1, \cdots, l$ représente le paramètre considéré et l le nombres de paramètres du circuit sous test. De la même manière, l'analyse de sensibilité est utilisée pour calculer les intervalles de tolérance $\Delta c'_j$ du $j^{ème}$ paramètre pour lesquels le circuit passe le test (à partir des mesures de test).

Une fois que ces paramètres sont calculés, ils seront utilisés pour calculer la fonction C donnée par l'Equation (3.22). C dépend du vecteur de test considéré étant donné que les intervalles de tolérances Δc_j et $\Delta c'_j$ dépendent de ce vecteur de test.

$$C = \sum_{j=1}^{l} \frac{(c_j)^2}{\left(|\Delta c_j| - |\Delta c'_j|\right)^2} \tag{3.22}$$

Le vecteur de test de cette technique (Figure 3.16) représente une série de pulsations. Chaque pulsation i est caractérisée par sa largeur PW_i et son temps d'échantillonnage t_i. Ceci parce que la sortie du circuit sous test considérée dans le domaine transitoire est sensible à ces paramètres.

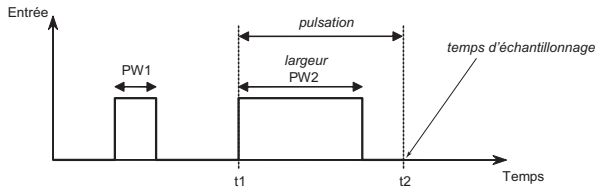


FIG. 3.16 – Vecteur de test composé d'une série de pulsations.

L'algorithme de génération du vecteur de test optimal présenté par cette technique consiste à trouver le nombre optimal de pulsations, la largeur optimale et le temps d'échantillonnage

Algorithme 3.1 : Algorithme de génération d'un vecteur de test optimal sous forme de pulsations

```
ENTREES et INITIALISATIONS :
   Soit TMAX le temps de test maximal
   Soit Ttot le temps de test total : Ttot = 0
   Soit C la fonction coût : C = 0
   Soit E l'ensemble des pulsations du vecteur de test : E = ensemble vide
TANT QUE ((Ttot<TMAX) OU (C n'est pas amélioré)) FAIRE
   Ensemble du pulsations temporaire Et = ensemble vide
   TANT QUE (PWi<Tmax1) FAIRE
     TANT QUE (ti<Tmax2) FAIRE
       Calculer C
       SI  (C est amélioré) ALORS
         Et = Et union {(PWi,ti)}
       FIN SI
     FIN TANT QUE
   FIN TANT QUE
   E = E union Et
FIN TANT QUE
SORTIES : Les pulsations du vecteur de test : E
```

optimal de chaque pulsation permettant de minimiser la fonction C donnée par (3.22). Il est présenté par l'Algorithme 3.1.

`Tmax1` et `Tmax2` représentent la largeur maximale d'une pulsation et le temps d'échantillonnage maximal d'une pulsation, respectivement.

Le vecteur de test sera donc celui qui maximisera la détection de fautes. Les fautes considérées par cette technique sont les fautes paramétriques simples.

3.3.3 Par ordonnancement des tests fonctionnels

Une approche de minimisation du *temps de test* de production en minimisant l'ensemble des vecteurs de test proposée dans [75, 76] est basée sur l'ordre des performances qui maximise la Couverture de fautes. Les raisons de cet ordre viennent du fait que certains tests doivent être privilégiés pour les raisons suivantes :

- ces tests ont une très grande probabilité d'échouer
- ces tests nécessitent un temps très court
- ces tests sont presque indépendants des tests faits dans des phases précédentes

En effet, le Rendement Y_i d'un test d'une performance i dépend de la position de la performance par rapport aux autres. C'est-à-dire, tester une performance en premier engendre un rendement différent que si elle est testée en dernier.

A partir d'un certain ordre, le sous-ensemble de performances qui donne une bonne Couverture de fautes, pour lequel le temps de test (3.23) est minimisé, remplacera l'ensemble de performances de départ. Le problème est modélisé sous forme de recherche du plus court chemin dans un graphe orienté (Figure 3.17), où les sommets représentent l'ensemble des performances considérées et les arêtes reliant le sommet i au sommet j représentent le coût (Equation (3.23)) engendré par l'ajout de la performance j à l'ensemble des performances i.

$$temps\ de\ test = \sum_{i \in T} W_i \left[\prod_{j=1}^{i-1} Y_j \right] \tag{3.23}$$

où T représente l'ensemble des indices des performances avec un certain ordre de sorte à atteindre la Couverture de fautes souhaitée.

Cependant, le plus court chemin trouvé représente l'ordre des performances, qui minimise le temps de test. L'un des algorithmes utilisés pour résoudre ce problème est l'algorithme de

Djikstra. Un autre algorithme basé sur la programmation dynamique a été proposé par [59], où ce problème a été modélisé comme problème d'ordonnancement du rang[9].

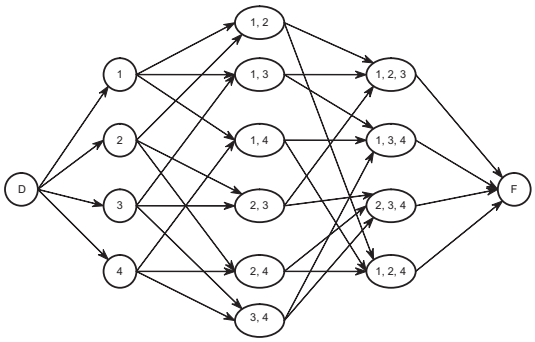


FIG. 3.17 – Graphe pour ordonner quatre performances.

3.4 Génération de vecteurs de test structurels

D'une manière classique, le test analogique est une procédure qui nécessite la vérification des spécifications d'un circuit, ce qui est le principe d'un test fonctionnel. Celui-ci est le plus utilisé et il est dans la majorité des cas l'unique solution complète de test. En revanche, son coût, en terme de ressources et collection d'une importante quantité de données, est de plus en plus inacceptable. L'une des alternatives d'un test fonctionnel est le test basé sur les fautes (ou structurel). Les tests visent la détection des fautes à la place de la validation des spécifications. Dans cette section, nous présenterons brièvement quelques techniques de génération de tests visant la détection de fautes.

3.4.1 Par projection dans le domaine des mesures de test

La technique présentée dans [77] a comme objectif l'optimisation de l'ensemble des vecteurs de test structurels. Elle considère les fautes catastrophiques simples sous un test paramétrique DC lors de la phase de production. Cette technique est basée sur la détermination de l'ensemble des signatures permettant de distinguer les circuits fonctionnels des défaillants. La Figure 3.18 montre un exemple d'une signature qui est représentée par un polyèdre de tolérances des mesures de test. Elle est calculée à base de la matrice de sensibilité (paramètres du circuit et critères de test).

Pour un vecteur de test donné, les différentes signatures pour un circuit sans fautes ainsi que pour tous les circuits avec fautes sont calculées. Une faute est donc détectée si sa signature ne s'intersecte pas avec celle du circuit sans faute. La Figure 3.19 montre un exemple d'une faute détectée (a) et d'une faute non détectée (b).

L'algorithme de génération de vecteurs de test présenté dans [77] permet donc de trouver un ensemble minimal de vecteurs de test permettant de détecter toutes les fautes. Il se résume comme suit :

Soit Δy_{ij}^t la différence entre la valeur d'un critère de test j d'un circuit sans faute et celle d'un circuit avec la faute i en considérant un vecteur de test t. Soit :

$$\Delta Y^t = \sum_{i=1}^{N} \sum_{j=1}^{M} \Delta y_{ij}^t \tag{3.24}$$

[9]Rank Ordering Problem.

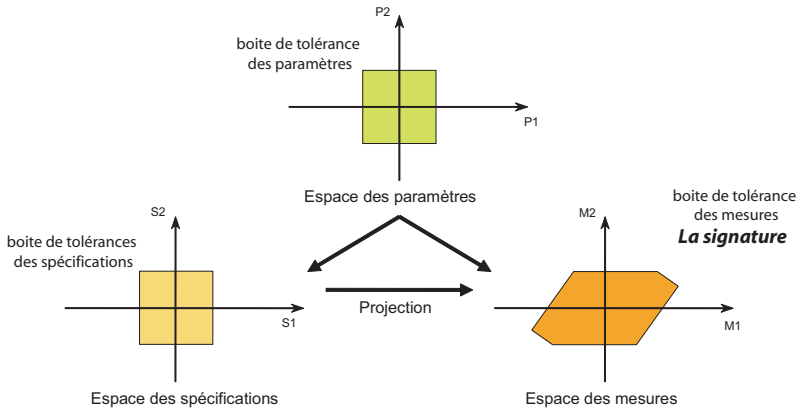


FIG. 3.18 – Calcul de l'espace de la signature.

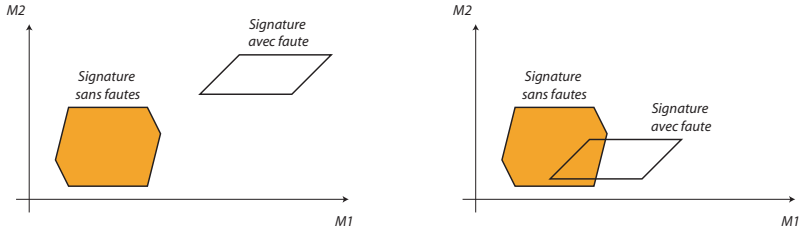


FIG. 3.19 – Signature d'un circuit sans faute et d'un circuit avec (a) faute détectée et (b) faute non détectée.

où, N est le nombre de fautes et M le nombre de critères de test. Le meilleur vecteur de test t^* est celui pour lequel la valeur ΔY^{t^*} est maximale. C'est-à-dire :

$$\Delta Y^{t^*} = \underbrace{max}_{t \in T} \Delta Y^t \tag{3.25}$$

La forme algorithmique de cette technique est présentée par l'Algorithme 3.2.

Algorithme 3.2 : Algorithme de génération de vecteurs de test par projection sur le domaine de la signature

ENTREES et INITIALISATIONS :
Soit F = ensemble de toutes les fautes
Soit T = ensemble de tous les vecteurs de test
Soit T^* = l'ensemble minimal de vecteurs de test
Soit $f = 0$, la Couverture de fautes courante
Soit f^* la Couverture de fautes objectif
TANT QUE $f < f^*$
Soit t^* le meilleur vecteur de test sur T
$T^* = T^* \cup \{t^*\}$
Soit F_i l'ensemble de fautes détectées par t^*
$F = F - F_i$
Calculer f
FIN TANT QUE
SORTIES : T^* est donc l'ensemble minimal de vecteurs de test

3.4.2 Par calcul des paramètres de détection dans le domaine temporel

La technique présentée dans [14] vise à générer des vecteurs de test en observant les paramètres[10] des différentes réponses dans le domaine transitoire du circuit sous test, tels que

[10]Ces paramètres seront considérés comme mesures de test.

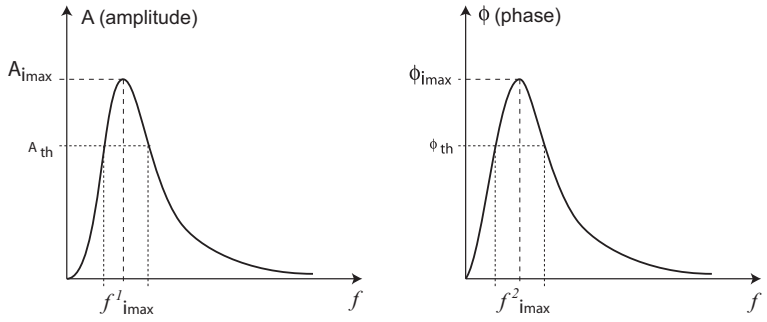


FIG. 3.20 – La différence entre l'amplitude (a) et la phase (b) d'un état stationnaire d'un circuit dans le domaine fréquentiel.

l'amplitude, la phase du signal de sortie et la rampe de sortie. Deux types de vecteurs de test sont considérés. Le premier type représente les signaux sinusoïdaux (ou signaux multi-fréquences) et le deuxième type représente les signaux à rampe[11]. Le but de cette technique est de détecter un maximum de fautes catastrophiques et paramétriques simples.

Concernant le cas des vecteurs de test du premier type, les mesures de test observées sont au nombre de deux. La première mesure représente la différence entre l'amplitude du signal de sortie du circuit sous test avec faute et celui sans faute. Cette différence est appelée aussi *erreur d'amplitude* du circuit sous test. La deuxième mesure de test représente la différence entre la phase du signal de sortie du circuit sous test avec faute et celui sans faute. Cette différence est appelée aussi *erreur de phase* du circuit sous test.

L'algorithme de génération de vecteur de test pour ce genre de mesures de test consiste à balayer une certaine plage de fréquences pour chaque faute injectée, puis calculer l'erreur de l'amplitude pour chaque fréquence. Si la fréquence du signal, pour laquelle cette erreur est maximale, est supérieur au seuil fixé, alors elle sera choisie comme fréquence du vecteur de test détectant cette faute. Si toutes ces fréquence (des vecteurs de test) ne détectent pas 100% de fautes, alors la même démarche sera refaite pour le cas des erreurs de la phase. Pour chaque faute injectée i, l'amplitude maximale $A_{i_{max}}$ (Figure 3.20(a)) de sortie est comparée à l'amplitude seuil ce qui permet de déterminer si la faute est détectée ou non. Si elle est détectée, alors la fréquence $f^1_{i_{max}}$ correspondante à $A_{i_{max}}$ est ajoutée à l'ensemble des fréquences du premier ensemble de vecteurs de test V_1. Les fréquences du deuxième ensemble de vecteurs de test V_2 sont déterminées de la même manière en observant la phase max $\phi_{i_{max}}$ (Figure 3.20(b)) au lieu de $A_{i_{max}}$.

Un seul ensemble peut être considéré si l'autre ne permet pas d'améliorer la Couverture de fautes. La forme algorithmique de cette technique est présentée par l'Algorithme 3.3.

Concernant le cas des vecteurs de test du deuxième type, la mesure de test observée est la rampe de sortie. Ces rampes sont choisies arbitrairement (en choisissant différents temps de descente). Elle seront ensuite injectées dans le circuit sous test pour chaque faute injectée, puis calculer les différents paramètres de la rampe de sortie tels que l'état stationnaire, le retard, le temps de descente et le dépassement[12]. Si l'un de ces paramètres varie au delà d'un pourcentage seuil toléré, alors la faute injectée est considérée détectée.

Des travaux en perspective ont été cités dans ce papier [14] proposant de considérer comme mesures de test les différents paramètres de la réponse impulsionnelle du circuit sous test.

[11] Rampe waveform.
[12] Overshoot.

Algorithme 3.3 : Algorithme de génération de vecteurs de test par calcul calcul de deux paramètres de détection à la sortie du circuit : L'amplitude et la phase

ENTREES et INITIALISATIONS :
Soit N fautes
Soit V_1 un ensemble de fréquences de vecteurs de test initialement vide ($V_1 = \varnothing$)
Soit V_2 un autre ensemble de fréquences de vecteurs de test initialement vide ($V_2 = \varnothing$)
Soit $f^1_{i_{max}}$ la fréquence pour laquelle l'amplitude de sortie $A_{i_{max}}$ est maximale sous la faute i
Soit $f^2_{i_{max}}$ la fréquence pour laquelle la phase de sortie $\phi_{i_{max}}$ est maximale sous la faute i
Soit A_{th} l'amplitude maximale tolérée (seuil). Donc pour une faute i, si $A_{i_{max}} > A_{th}$ alors la faute i est détectée
Soit ϕ_{th} la phase maximale tolérée (seuil). Donc pour une faute i, si $\phi_{i_{max}} > \phi_{th}$ alors la faute i est détectée
POUR i=1..N **FAIRE**
SI la faute i est détectée par l'amplitude $A_{i_{max}}$ **ALORS** : $V_1 = V_1 \cup \{f^1_{i_{max}}\}$
SI la faute i est détectée par la phase $\phi_{i_{max}}$ **ALORS** : $V_2 = V_2 \cup \{f^2_{i_{max}}\}$
FIN POUR
SI la Couverture de fautes donnée par l'un des vecteurs $V1$ ou $V2$ est égale à 100% **ALORS**
un seul vecteur de test suffit.
SINON
soient CF_1 la Couverture de fautes données par $V1$
CF_2 la Couverture de fautes donnée par $V2$ et $CF_{1,2}$ la
Couverture de fautes données par V_1 et V_2. Le (ou les
vecteurs) de test est
donc celui correspondant au $max\{CF_1, CF_2, CF_{1,2}\}$
FIN SI
SORTIES : V_1 et V_2

Faute	**Mesures**			
	M1	M2	M3	M4
F1	1	0	1	1
F2	0	1	0	0
F3	1	0	0	1
F4	0	1	1	0
F5	1	0	0	1

1 : Faute détectée, 0 : Fautes non détectée

TAB. 3.1 – Les fautes détectées et non détectées pour quelques mesures de test d'un circuit.

3.4.3 Par minimisation du nombre de fréquences de détection et de diagnostic

Une fois que l'ensemble de mesures de test de départ est minimisé, la technique proposée par [81] vise à minimiser le nombre de fréquences du vecteur de test destiné soit à la détection soit au diagnostic de fautes. Elle permet de considérer les fautes catastrophiques ainsi que les fautes paramétriques simples. Ce problème de minimisation a été modélisé comme un problème de recouvrement (voir l'Annexe A). L'un des avantages de cette méthode est que les simulations sont rapides étant donné qu'elles sont effectuées dans le domaine fréquentiel (simulations AC).

Pour la détection de fautes, l'algorithme présenté se résume comme suit : Premièrement, pour chaque faute injectée et pour chaque mesure de test considérée, un intervalle de fréquences permettant de détecter cette faute est déterminé. Ensuite, l'ensemble de mesures de test minimal ainsi que l'ensemble minimal des fréquences permettant de détecter toutes les fautes sera calculé.

Comme exemple, le Tableau 3.1 présente un ensemble de fautes et un ensemble de mesures de test permettant de détecter ces fautes. On déduit que les mesures M1 et M2 (ou M1 et M4) suffisent pour détecter toutes les fautes. L'algorithme de génération de vecteurs de test est présenté par la Figure 3.21. Un autre exemple montrant comment minimiser l'ensemble de fréquences du vecteur de test est présenté en Annexe A.2.

Cette technique permet non seulement la détection de fautes, mais aussi le diagnostic. Un même principe analogue à celui qui est utilisé pour la détection de fautes est considéré aussi pour le cas du diagnostic de fautes avec une légère modification [81].

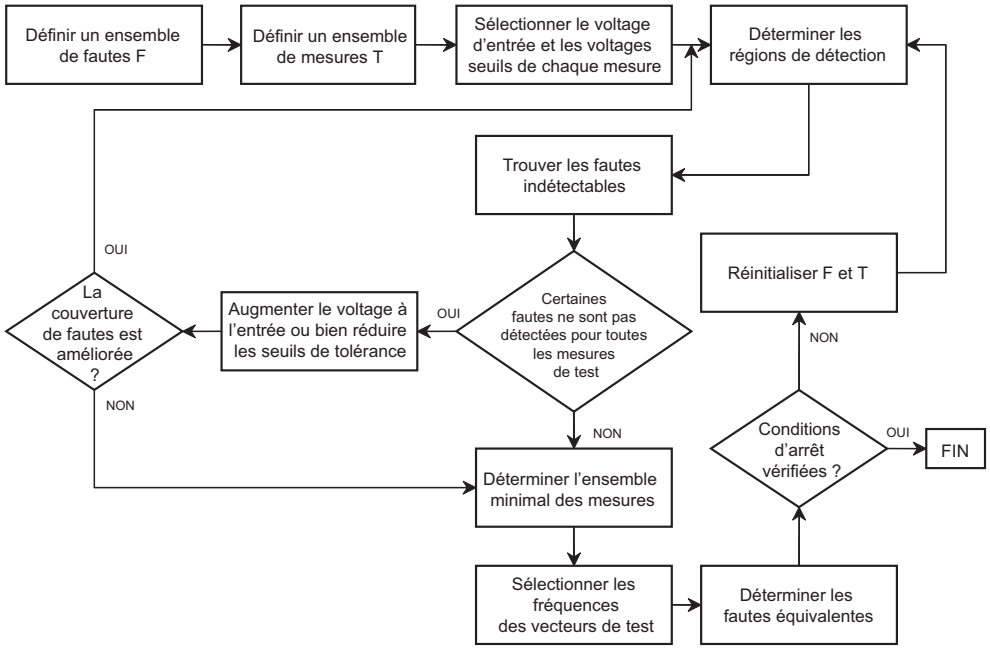


FIG. 3.21 – Génération de vecteurs de test par minimisation de l'ensemble des mesures de test et de fréquences de détection de fautes.

3.4.4 Par reproduction de la fonction de transfert

La technique présentée dans [110] permet de générer des vecteurs de test, pour détecter les fautes catastrophiques et paramétriques simples et multiples, en se basant sur la fonction de transfert. Une seule simulation dans le domaine fréquentiel du circuit sous test est nécessaire.

Le principe de cette technique est expliqué par les Figures 3.22. La première figure (3.22(a)) représente la réponse fréquentielle (ou la fonction de transfert) d'un filtre à variable d'état[13]. Les vecteurs de test initiaux représentent les différentes fréquences correspondant aux dérivées premières et secondes, ajoutées aux extrémités (premier et dernier point) de la fonction de transfert sans fautes. Ces fréquences sont montrées par les points noirs de la Figure 3.22(b). Ensuite, en utilisant la technique de l'interpolation, la fonction de transfert originale sera reproduite à l'aide des points initiaux. Comme le montre la Figure 3.22(b), ces points ne permettent pas de reproduire exactement la fonction de transfer originale du filtre, parce que la différence maximale Δ_{max} entre la fonction reproduite est l'originale est supérieure à la différence maximale tolérée Δ_{th}. Par conséquent, d'autres points (fréquences) seront ajoutés pour améliorer la précision de reproduction. La Figure 3.22(c) montre ces nouveaux points. Il est clair que ces points suffisent pour reproduire précisément, par interpolation, la fonction de transfert du filtre. En se basant sur ces fréquences (qui seront les fréquences des vecteurs de test), les fautes qui causent des déviations supérieures à Δ_{th} de la sortie du filtre seront détectées. La Figure 3.23 montre le diagramme de l'algorithme de cette méthode.

3.4.5 Par analyse de sensibilité

La technique présentée dans [111] permet de générer plusieurs vecteurs de test monofréquences (sinusoïdes) à l'aide de la sensibilité des mesures de test par rapport aux paramètres des composants du circuit sous test dans le domaines fréquentiel.

[13]State variable filter.

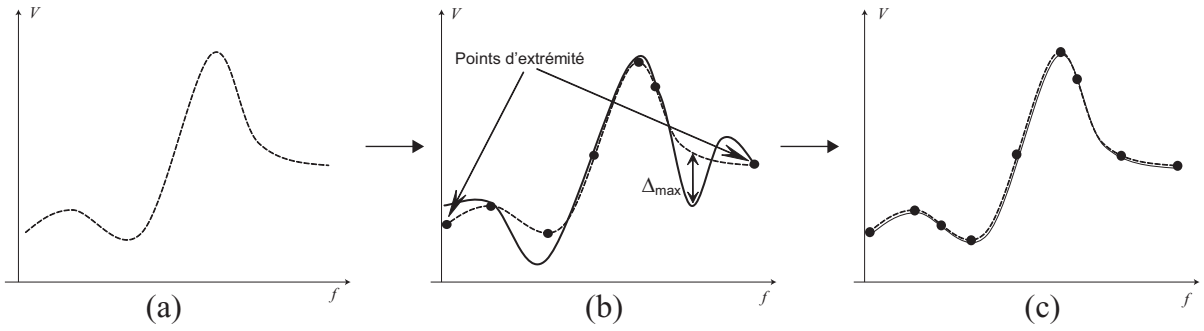


FIG. 3.22 – Reproduction de la fonction de transfert par interpolation.

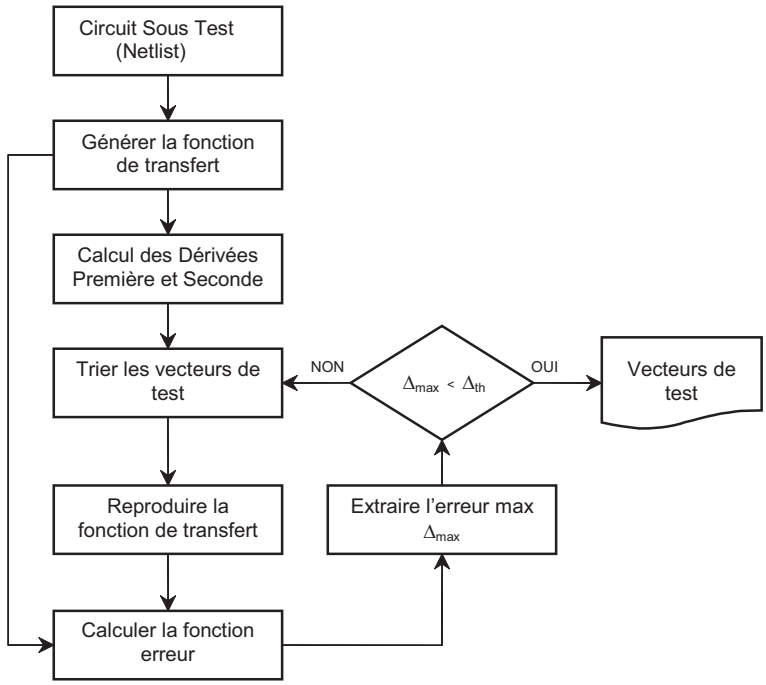


FIG. 3.23 – Génération de vecteurs de test par interpolation.

Les mesures de test considérées sont le *Gain* (ou l'amplitude) et la *Phase* du signal à la sortie des différents nœuds du circuit. L'algorithme de génération de vecteurs de test permet premièrement, de trouver pour chaque nœud, l'ensemble des fréquences des vecteurs de test permettant de détecter le maximum de fautes paramétriques. Puis, de choisir un ensemble minimal de nœuds permettant de détecter toutes les fautes. Les fréquences des vecteurs de test, pour un nœud et une mesure de test donnés, sont obtenues comme suit. Pour chaque paramètre, la fréquence pour laquelle la sensibilité de la mesure de test considérée par rapport à ce paramètre est maximale sera ajoutée à l'ensemble des fréquences du vecteur de test optimal. Cet algorithme est présenté par l'organigramme de la Figure 3.24.

Cette technique prend en considération les fautes simples ainsi que les fautes multiples, car elle n'utilise pas l'injection de fautes pour décider si une faute est détectée ou non, mais sur l'analyse de sensibilité et l'*observabilité de fautes*. L'observabilité représente l'existence ou non d'une fréquence permettant d'observer la sensibilité d'une mesure de test par rapport à un paramètre. Si cette fréquence existe, alors toute faute sur le paramètre en question sera détectée par cette fréquence, sinon les fautes sur ce paramètre seront considérées comme non détectées.

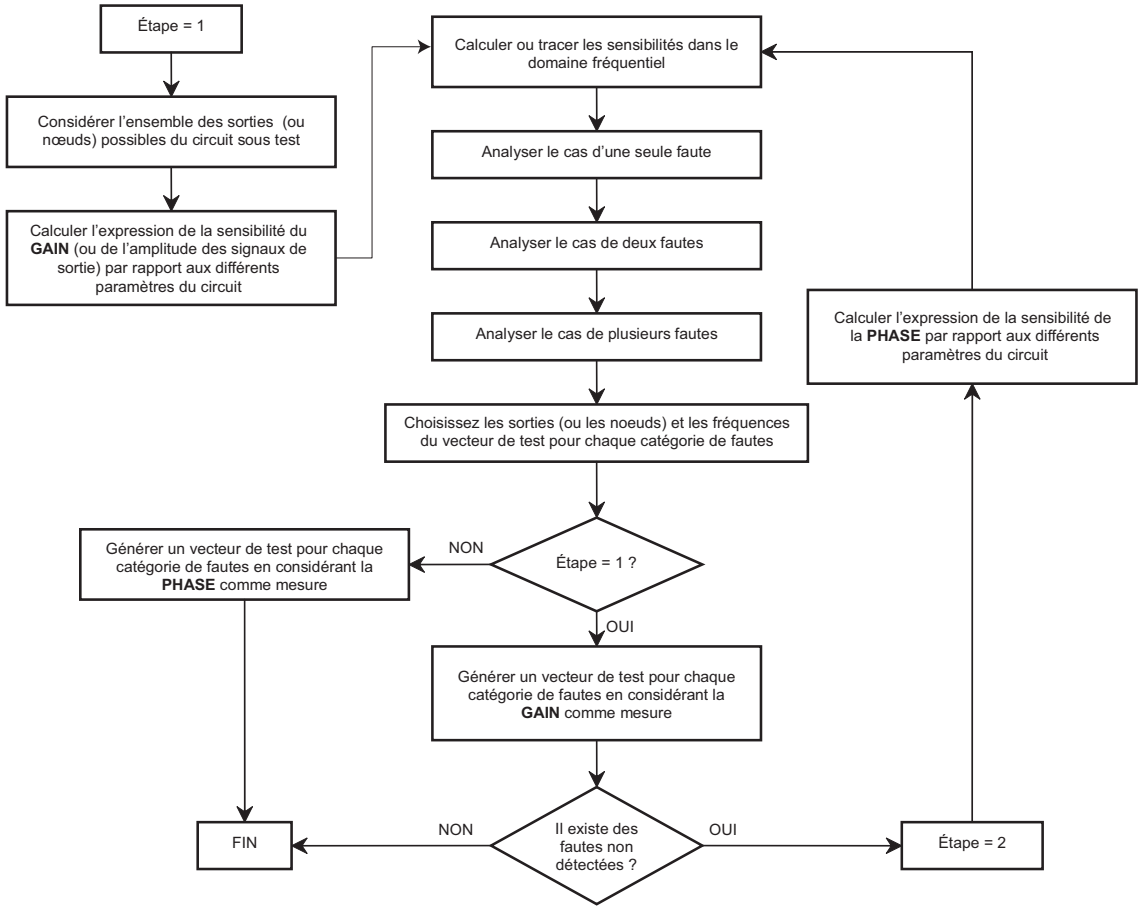


FIG. 3.24 – Algorithme de génération de vecteurs de test structurel en considérant l'analyse de sensibilité.

Une autre technique [83] utilise le même principe précédent. Elle permet de générer un ensemble de vecteurs de test minimal dans le domaine fréquentiel. Le vecteur de test choisi pour une faute (qui peut être paramétrique ou catastrophique) est celui qui correspond à la fréquence pour laquelle la sensibilité de la sortie par rapport à cette faute est maximale. Cette technique a été aussi utilisée dans [84] où l'ensemble des vecteurs de test est remplacé par un seul vecteur de test multi-fréquence. En effet, il suffit juste de trouver pour chaque signal monofréquence, l'amplitude et la phase optimales afin que ce nouveau vecteur (multi-fréquence) permette de garantir la même Couverture de fautes que celle des vecteurs de départ.

3.4.6 Par optimisation de la différence entre les réponses

Une technique, qui prend en considération les variations du process et qui est destinée au test des circuits non linéaires est présentée dans [33]. Les vecteurs de test considérés sont les voltages d'entrée du circuit sous test pour lesquels l'erreur E, représentant la différence entre la réponse du bon circuit et celle du circuit avec fautes, est maximale (3.26).

$$\underbrace{max}_{x \in X}(E) \tag{3.26}$$

où X représente le vecteur des intervalles de tolérance (valeur maximale et valeur minimale) des différents vecteurs de test de x.

Cette erreur est définie par (3.27) comme suit :

$$E = E(x,p) = \underbrace{min}_{p \in P}(E') \tag{3.27}$$

$$E' = E'(x,p) = y_g - y_f = g(x,p_g) - f(x,p_f) \tag{3.28}$$

où,

y_g : la réponse du circuit sans faute,
y_f : la réponse du circuit avec faute,
p_g : vecteur des paramètres du circuit sans faute,
p_f : vecteur des paramètres du circuit avec faute,
p : le vecteur (p_f, p_g),
P : vecteur des intervalles de tolérance des paramètres du circuit sous test,
x : vecteur de vecteurs de test (exemple, les voltages d'entrée),
g : fonction de transfert non linéaire du circuit sans faute,
f : fonction de transfert non linéaire du circuit avec faute.

L'algorithme de génération de vecteurs de test démarre d'un ensemble définit de fautes et de vecteurs de test. Il est décrit comme suit. Soit m_j la valeur de la métrique de test totale (donnée par l'Equation (3.29)) en considérant un vecteur de test j.

$$m_j = \alpha_1 \frac{n_{f_j}}{N} + \alpha_2 \frac{1}{N} \sum_{i=1}^{N} q_{ij} \tag{3.29}$$

où n_{f_j} représente le nombre de fautes détectées sous le vecteur de test j et α_1 et α_2 déterminent l'importance relative ($\alpha_1 + \alpha_2 = 1$) des deux termes sommés de (3.29). Le terme q_{ij} représente la séparation de fautes entre le circuit fonctionnel et le circuit avec la faute i (voir la partie sur la métrique séparation de fautes de la Section 3.2.3) et N est le nombre de fautes.

Le vecteur de test pour lequel cette métrique m_j est maximale sera choisi et sera éliminée de l'ensemble des vecteurs de test de départ. Puis, toutes les fautes détectées par ce vecteur de test seront éliminées de l'ensemble de fautes de départ. S'il reste des fautes non détectées par ce vecteur de test, alors, la procédure sera refaite jusqu'à ce que toutes les fautes seront détectées.

Le critère de détection de fautes utilisé est basé sur le calcul de la métrique séparation de fautes présentée en Section 3.2.3. L'algorithme final de cet approche est donné par l'Algorithme 3.4.

Une autre approche d'un test structurel dans le domaine fréquentiel a été présentée dans [60], qui permet de détecter les fautes catastrophiques et paramétriques. Le vecteur de test optimal est choisi de telle sorte que l'*Erreur de la réponse* E du circuit sous test soit maximale. Cette erreur est définie pour chaque faute F_i et pour une fréquence du vecteur de test f_j comme étant la différence entre la mesure de test T_g du circuit fonctionnel et celle du circuit défaillant T_{F_i}. Elle est donnée par $E_i(f_j)$ comme suit :

$$E_i(f_j) = |T_g(f_j) - T_{F_i}(f_j)| \tag{3.30}$$

Les vecteurs de test sont choisis pour chaque faute de telle sorte que si une faute est détectée (en mesurant sa détectabilité par (3.9)) par la mesure de test T, alors la meilleure fréquence pour laquelle l'erreur de la réponse est maximale sera choisie comme étant la fréquence du vecteur de test permettant de détecter cette faute.

Algorithme 3.4 : Algorithme de génération de vecteurs de test par optimisation de la différence entre les réponses du circuit fonctionnel et du circuit avec faute

ENTREES et INITIALISATIONS :
Soit t_i le vecteur de test i
Soit n le nombre total de fautes à injecter
Soit k le nombre de vecteurs de test
Soit $F = \{f_1, f_2, ..., f_n\}$ un ensemble initial de fautes
Soit $T = \{t_1, t_2, ..., t_k\}$ un ensemble initial de vecteurs de test
Soit $S = \varnothing$ l'ensemble optimal de vecteurs des test
TANT QUE (T non vide et F non vide) **FAIRE**
POUR chaque ti de T
calculer m_i *//la métrique de test totale*
FIN POUR
Soit M = $\{m_1, m_2, ...\}$
m_jMAX = max(M)
Choisir t_jMAX pour lequel m_jMAX est obtenu
S = S U t_jMAX
Eliminer toutes les fautes détectées par le vecteur t_jMAX
FIN TANT QUE
SORTIES : S

3.4.7 Par génération aléatoire des fréquences

Dans la technique présentée par [103], les vecteurs de test sont générés dans le domaine fréquentiel et de manière aléatoire en utilisant les algorithmes génétiques[14] classiques [51]. Cette technique est utilisée pour la détection et le diagnostic des fautes paramétriques simples.

L'idée de l'algorithme est de générer un certain nombre d'ensembles de quatre vecteurs de test (ou quatre fréquences). Puis lancer l'injection de fautes dans le but de choisir, en utilisant les algorithmes génétiques, les meilleurs ensembles permettant de détecter ou de diagnostiquer le maximum de fautes. L'organigramme de la Figure 3.25 résume cet algorithme.

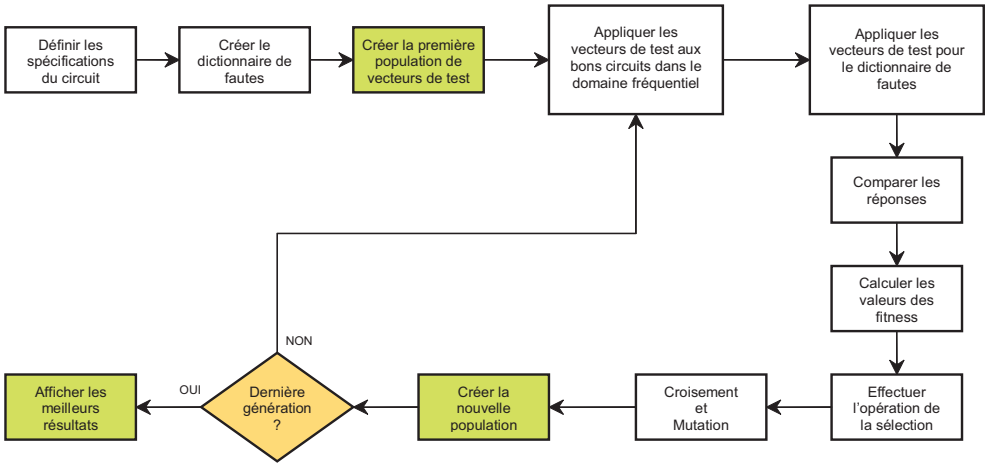


FIG. 3.25 – Génération de vecteurs de test par les Algorithmes Génétiques.

3.4.8 Par reconfiguration en mode oscillateur

Ce type de test ne nécessite pas de génération de vecteurs de test. Le principal avantage de ce test est la réduction du coût de test due à la suppression de la phase de génération de vecteurs de test [10]. Durant la phase de test, le circuit sous test peut être partitionné en plusieurs blocs

[14]Le fitness représente la moyenne quadratique des différences entre les points (fréquences) de la réponse du circuit fonctionnel et des points correspondants de la réponse du circuit défaillant. Plus ce fitness est grand, plus la probabilité de sélectionner le vecteur de test correspondant est grande.

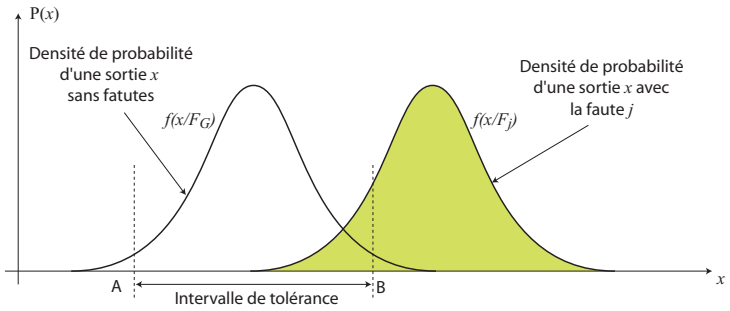


FIG. 3.26 – Densités de probabilité d'un circuit sans faute et d'un autre avec faute.

fonctionnels. Chaque bloc est configuré en oscillateur. La fréquence des oscillations et l'amplitude sont les paramètres de test.

Pour ce genre de test, un exemple de la Couverture de fautes (catastrophiques et paramétriques) obtenue est présentée dans [9].

3.4.9 Par calcul de la distance de séparation

La méthode présentée par [112] permet de trouver les vecteurs de test permettant de détecter des fautes catastrophiques avec un maximum de Couverture de fautes sous des déviations process. Elle est composée de deux étapes principales : La première étape consiste à déterminer toutes les fautes totalement détectées, puis fixer leur probabilité de détection POD[15] correspondante à 100%. La deuxième étape consiste à calculer la métrique $DIST_{i,j}$[16] donnée par (3.31) pour chaque faute j qui n'est pas déjà détectée dans la première étape et pour un certain vecteur de test i.

$$DIST_{i,j} = \left. \frac{|x_{F_j} - x_{F_G}|}{\sigma_G} \right|_i \tag{3.31}$$

où x_{F_j} représente la valeur moyenne de la mesure de test x sous la faute j, x_{F_G} est la valeur moyenne de la mesure de test x sans faute et σ_G représente l'écart-type de la distribution de cette mesure de test sans fautes. Ces valeurs peuvent être obtenues en effectuant la simulation Monte Carlo du circuit sous test.

Pour chaque faute j, le vecteur de test i maximisant la distance métrique $DIST_{i,j}$ sera choisi. Si elle n'est pas nulle (c'est-à-dire que la faute j est partiellement détectable sous le vecteur de test i) alors la probabilité de détection $POD_{i,j}$ (voir Equation (3.32)) sera calculée. C'est le critère permettant de décider si la faute j est détectée ou non par le test i (en la comparant à un certain seuil).

$$POD_j = \int_{-\infty}^{A} f(x/F_j)\ dx + \int_{B}^{+\infty} f(x/F_j)\ dx \tag{3.32}$$

où A et B sont les bornes de l'intervalle de tolérance de la mesure de test x sans fautes et sont représentés sur la Figure 3.26, et $f(x/F_j)$ représente la densité de probabilité d'un circuit avec la faute j.

L'algorithme de génération de vecteurs de test qui se compose de deux étapes se résume donc comme suit : Dans la première étape, l'ensemble des fautes totalement détectées ainsi que les vecteurs de test correspondants seront éliminés des ensembles de départ.

La deuxième étape de l'algorithme concerne les fautes qui sont partiellement détectées pour lesquelles leur valeur $DIST$ correspondante n'est pas nulle. Pour chacune de ces fautes, le vecteur

[15] La probabilité de détection POD est aussi utilisée par [63] sous le nom de FDP.
[16] Appelée *Distance de séparation*.

Algorithme 3.5 : Algorithme de génération des vecteurs de test par calcul de la distance de séparation et de la probabilité de détection

```
ENTREES et INITIALISATIONS :
Démarrer d'un ensemble de M vecteurs de test.
POUR chaque vecteur de test i=1..M FAIRE
   TANT QUE il existe des fautes non injectées FAIRE
      Effectuer les simulation Monte Carlo pour le circuit
      sans faute pour déterminer les bornes inf et sup de la
      mesure de test.
      POUR chaque faute j=1..N qui n'est pas déjà injectée FAIRE
         Injecter la faute j
         Classer la faute (Totalement détectée ou non détectée)
         SI la faute j est totalement détectée ALORS
            éliminer la faute j de la liste des fautes
            fixer sa POD_j à 100%
            fixer STIM_j à i
         SINON
            Calculer la métrique DIST_ij
         FIN SI
      FIN POUR
   FIN TANT QUE
FIN POUR
POUR chaque faute j=1..N avec j n'est pas détectée FAIRE
   REPETER
      Trouver le meilleur vecteur de test i pour lequel
      DIST_ij=max(DIST_ij) sur tous les i
      SI DIST_ij != 0 ALORS
         effectuer la simulation Monte Carlo en utilisant le
         vecteur de test i pour calculer POD_j
         fixer STIM_j à i
         fixer DIST_ij à 0
      FIN SI
   TANT QUE max(DIST_ij) sur tous les i != 0 et la
   mesure de test ne suit pas une loi Gaussienne.
FIN POUR
SORTIES : POD_j, STIM_j pour chaque faute j
```

de test, pour lequel la valeur $DIST$ correspondante est maximale, sera choisi. La sortie de l'algorithme sera pour chaque faute injectée, sa probabilité de détection ainsi que le vecteur de test permettant de la détecter. La moyenne des POD sur toutes les fautes représentera donc la valeur de la Couverture de fautes.

L'algorithme de cette méthode est donné par l'Algorithme 3.5, où $STIM_j$ représente le vecteur de test permettant de détecter la faute j.

3.4.10 Par calcul du risque de Bayes

Le choix d'un vecteur de test de la méthode présentée par [100] est basé sur le calcul du *risque de Bayes* R donné comme suite :

$$R = C_{10} \cdot \mathbf{P}(H_1/H_0) \cdot \mathbf{P}(H_0) + C_{01} \cdot \mathbf{P}(H_0/H_1) \cdot \mathbf{P}(H_1) \tag{3.33}$$

où C_{10} est le coût engendré par un circuit défaillant qui passe le test et C_{01} est le coût engendré par un circuit fonctionnel qui échoue au test, $\mathbf{P}(H_1/H_0)$ est la probabilité pour qu'un circuit passe le test sachant qu'il est défaillant et $\mathbf{P}(H_0/H_1)$ est la probabilité pour qu'un circuit échoue au test sachant qu'il est fonctionnel. Ces deux probabilités sont présentées par la Figure 3.10.

Dans cette méthode, le risque de Bayes R est calculé pour chaque faute injectée et pour chaque vecteur de test utilisé. Pour une faute donnée, le vecteur de test pour lequel R est minimal sera donc choisi pour celle-ci. L'ensemble des vecteurs de test trouvé à la fin sera donc considéré comme étant l'ensemble de vecteurs de test permettant de détecter le maximum de fautes.

3.4.11 Génération sous des variations paramétriques multiples

La méthode présentée dans [5] permet de trouver l'ensemble minimal de fréquences de test permettant de détecter le maximum de fautes paramétriques simples tout en considérant les déviations process du circuit sous test. Cette méthode est basée sur la détermination d'un nouvel intervalle de tolérance de chaque paramètre du circuit de telle sorte que si ce paramètre prend une valeur en dehors de cet intervalle alors au moins une des spécifications du circuit sera violée. Ceci permettra d'éliminer l'effet de masquage des fautes. La Figure 3.27 montre un exemple d'un paramètre x_i, où, $[x_{i_l}, x_{i_u}]$ représente l'intervalle de tolérance de ce paramètre (défini par la technologie du circuit) et $[x^-_{i_{th}}, x^+_{i_{th}}]$ représente son intervalle maximal pour lequel toutes les spécifications sont vérifiées. En dehors de cet intervalle, au moins une des spécifications du circuit sera violée, on appellera désormais cet intervalle, *intervalle de test*. L'intervalle $[\overline{x}_{i_l}, \overline{x}_{i_u}]$ représente les valeurs réelles que peut prendre ce paramètre. C'est-à-dire, que ce paramètre ne peut pas prendre en réalité des valeurs en dehors de cet intervalle.

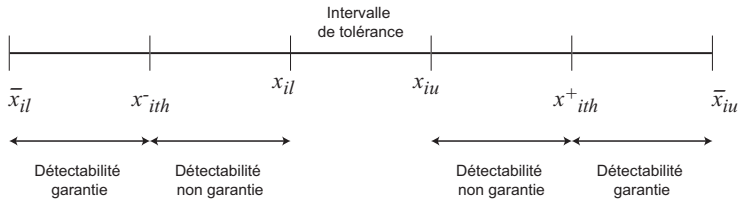


FIG. 3.27 – Intervalles de tolérance, de détectabilité et acceptables d'un paramètre du circuit.

L'algorithme de génération de vecteurs de test permet de déterminer pour chaque paramètre x_i la fréquence pour laquelle l'intervalle de test $[x^-_{i_{th}}, x^+_{i_{th}}]$ sera le plus serré. Il se résume comme suit. Soient un ensemble de m paramètres $x = (x_1, x_2, \cdots, x_m)$, un ensemble de n fréquences de test $f = (f_1, f_2, \cdots, f_n)$ et l'ensemble des p performances du circuit $T = (T_1, T_2, \cdots, T_p)$. Soit $x^+_i(k, j)$ la valeur maximale du paramètre x_i pour laquelle la performance T_k reste vérifiée (ou non vérifiée si $x_i > x^+_i(k, j)$) sous le vecteur de test (ou la fréquence) f_j. Soit $x^-_i(k, j)$ la valeur minimale du paramètre x_i pour laquelle la performance T_k reste vérifiée (ou non vérifiée si $x_i < x^-_i(k, j)$) sous le vecteur de test (ou la fréquence) f_j. Pour un paramètre x_i donné, on calcule son intervalle de test $[x^-_i(k_1, j_1), x^+_i(k_2, j_2)]$ pour tous les f_{j_1} et f_{j_2} de f et pour tous les T_{k_1} et T_{k_2} de T. Soit $f_{j^*_1}$ et $T_{k^*_1}$ la fréquence pour laquelle $x^-_i(k_1, j_1)$ est maximal et soit $f_{j^*_2}$ et $T_{k^*_2}$ la fréquence pour laquelle $x^+_i(k_2, j_2)$ est minimal. Alors les deux fréquences $f_{j^*_1}$ et $f_{j^*_2}$ représentent les vecteurs de test nécessaires pour détecter les fautes paramétriques sur le paramètre x_i. De la même manière, les autres vecteurs de test permettant de détecter les fautes sur les autres paramètres seront déterminés. L'ensemble final de ces vecteurs représentera l'ensemble de vecteurs de test optimal permettant de détecter toutes les fautes paramétriques simples du circuit sous test. Cet algorithme est décrit par l'Algorithme 3.6.

3.5 Génération de vecteurs de test alternatifs

Le test alternatif (voir la Section 2.7.4) exploite des techniques de régression pour prédire les spécifications d'un circuit à partir des mesures de test simples. La procédure de génération de vecteurs de test est utilisée non pas pour améliorer la Couverture de fautes ou bien les métriques de test analogiques, mais pour améliorer l'erreur sur la prédiction des spécifications.

Le test alternatif a été appliqué en particulier aux systèmes RF. Dans [15], les vecteurs de test sont choisis de telle sorte à ce qu'ils aient une faible puissance, car ceci permet de réduire l'erreur dans les prédictions. Deux algorithmes ont été présentés. Ils permettent de générer deux

Algorithme 3.6 : Algorithme de génération de vecteurs de test sous des variations paramétriques multiples

ENTREES et INITIALISATIONS :
Soit $x = (x_1, x_2, \cdots, x_m)$
Soit $f = (f_1, f_2, \cdots, f_n)$
Soit $T = (T_1, T_2, \cdots, T_p)$
Soit $V = \varnothing$ l'ensemble de départ des vecteurs de test
POUR chaque paramètre x_i de x **FAIRE**
POUR chaque fréquence f_i de f **FAIRE**
POUR chaque performance T_k de T **FAIRE**
Calculer $x_i^-(k,j)$
Calculer $x_i^+(k,j)$
FIN POUR
FIN POUR
Soit k_1^* et j_1^* la $k^{ème}$ performance et la $j^{ème}$ fréquence pour laquelle $x_i^-(k,j)$ est maximale
Soit k_2^* et j_2^* la $k^{ème}$ performance et la $j^{ème}$ fréquence pour laquelle $x_i^+(k,j)$ est minimale
V = V $\cup\{f_{j_1^*}, f_{j_2^*}\}$
FIN POUR
SORTIES : V est l'ensemble de vecteur de test optimal

vecteurs de test pour un système émetteur/récepteur. Le premier vecteur de test est destiné pour l'émetteur. Il représente un signal numérique périodique choisi à partir d'un ensemble de trains binaires donné. Chacun des trains binaires sera injecté dans l'émetteur puis une métrique TRM[17], donnée par l'Equation (3.34), sera calculée à partir de la réponse fréquentielle de l'émetteur. Le train binaire ayant une valeur petite du TRM et qui permet de minimiser l'erreur sur le prédiction des spécifications sera choisi comme étant le vecteur de test optimal. L'algorithme de cette méthode est résumé par la Figure 3.28.

$$\mathrm{TRM} = \sum_{k=1}^{N} |V_k| \tag{3.34}$$

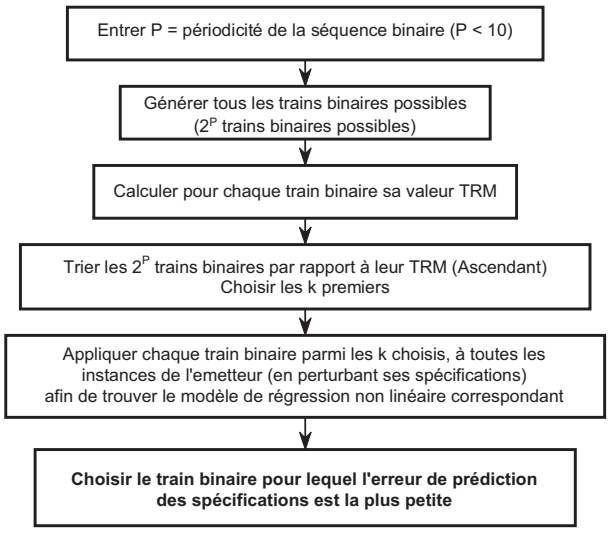


FIG. 3.28 – Génération d'un vecteur de test pour le test altérnatif d'un émetteur RF.

Pour ce qui est du deuxième vecteur de test, qui est destiné à tester le récepteur, il est sous forme d'un signal analogique multi-fréquence $X(t) = \sum_{k=1}^{N} V_k \cos(2\pi f_k t)$. L'algorithme de génération de ce vecteur démarre d'un signal à N fréquences et détermine les valeurs de

[17]Test Ranking Metric, il représente la somme des amplitudes des fréquences du signal de sortie d'un système.

ces fréquences ainsi que de leurs amplitudes correspondantes (V_k, f_k) permettant de minimiser l'erreur de prédiction des spécifications. Il est à noter que la phase de chaque fréquence du vecteur de test est supposée nulle $(\phi_k = 0, \forall k)$, mais il est possible de considérer ce paramètre aussi pour l'optimisation du vecteur de test.

La technique de [15] est une extension des approches *Loopback Test*[18] présentées par [52, 53, 113]. Dans la première approche, le système émetteur/récepteur est considéré comme étant un seul système, et uniquement un seul vecteur de test (train binaire périodique) est considéré à l'entrée de l'émetteur qui est lié directement au récepteur. Dans les deux dernières approches, la technique de génération d'un vecteur de test analogique multi-fréquence expliquée ci-dessus a été utilisée juste pour un récepteur.

3.6 Conclusion

Ce chapitre a été consacré à un état de l'art des outils de CAT analogique. La première partie présente les différents outils et techniques de simulation, de modélisation et d'injection de fautes ainsi que les différentes techniques permettant d'améliorer la simulation de fautes. Dans la deuxième partie, nous avons présenté quelques techniques d'optimisation et de génération de vecteurs de test analogiques. Cet état de l'art nous permettra de comprendre les choix qui ont été faits en termes de techniques et d'outils qui ont été développés dans la plateforme de CAT qui sera présentée dans les chapitres suivants.

[18] La technique Loopback est appelée aussi boucle de retour, consiste à boucler le récepteur avec l'émetteur et permet de considérer l'ensemble émetteur/récepteur comme étant un seul système.

Chapitre 4

Évaluation des métriques de test sous des déviations process

4.1 Introduction

Lors de la phase de design d'un circuit, il s'avère nécessaire d'envisager les techniques de test à utiliser lors de la production ou lors de l'application. Nous avons vu dans les chapitres précédents que nous pouvons utiliser soit les performances du circuit lors du test, soit des mesures ou des critères de test qui permettent le test structurel ou le test alternatif. Ces deux derniers cas aboutissent souvent sur des techniques à bas-coût, mais il est nécessaire de fixer les limites de test afin d'optimiser les métriques de test.

Nous allons présenter une nouvelle technique pour l'évaluation des métriques de test sous des déviations process. Cette technique nous permettra de fixer les limites des critères de test. Une fois les limites de test sont fixées en considérant les déviations process, les métriques de test seront calculées en présence des fautes catastrophiques et paramétriques simples pour les différents critères de test potentiels. Ceci, afin d'évaluer la capacité de la technique pour détecter les fautes catastrophiques et paramétriques simples et de trouver l'ensemble minimal des performances pour lequel la Couverture de fautes est meilleure. La Figure 4.1 résume tout ce processus. Généralement, il n'est pas possible d'estimer les métriques de test directement sur un échantillon de circuits généré par la simulation Monte Carlo, étant donné que les circuits défaillants sont rarement générés. D'où, donc, la nécessité de générer un échantillon de circuits plus large. La technique utilisée pour cette génération sera présentée dans ce chapitre.

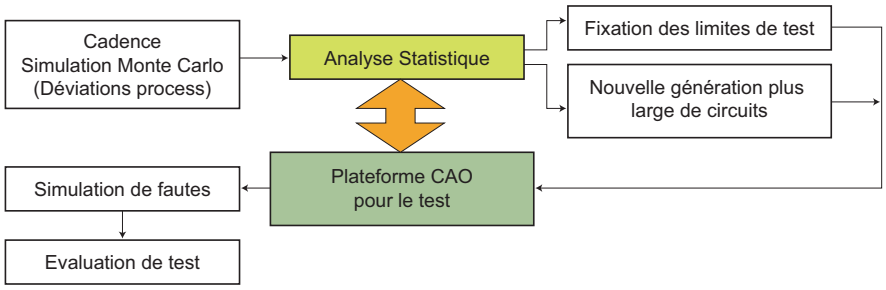


FIG. 4.1 – Méthode d'évaluation de test par modélisation statistique du circuit sous test.

L'évaluation de test sous des déviations process nécessite l'utilisation des techniques statistiques. Les techniques les plus utilisées sont celles qui permettent de faire la discrimination géométrique entre les circuits fonctionnels et les circuits défaillants. Le travail présenté dans [69]

permet de faire une classification linéaire des circuits en utilisant l'analyse de discrimination logistique. Une autre technique présentée par [114, 115] permet d'utiliser une technique d'apprentissage à base des réseaux de neurones qui permet de faire une discrimination non linéaire des circuits.

Les approches statistiques que nous allons considérer ici permettent de discriminer les différents types de circuits à base des distributions de probabilité de leurs performances et de leurs critères de test. Au passage, notons que l'analyse que nous allons faire par la suite pour estimer les métriques de test sous des déviations process, est aussi applicables pour l'analyse des fautes paramétriques multiples. La considération de fautes simples (catastrophiques ou paramétriques) est sensible pour des techniques de test de production si le design est robuste. En revanche, dans le cas où les limites de test de production sont très serrées, le Taux de défauts resultant des déviations paramétriques multiples peuvent être importants. En outre, le processus de vieillissement provoque des défaillances qui sont souvent causées par des déviations multiples.

Nous allons présenter dans ce qui suit trois approches permettant d'estimer les métriques de test analogiques afin de pouvoir fixer correctement les limites de test, surtout dans le cas ou le design est robuste, c'est-à-dire, le cas où la probabilité d'avoir un circuit défaillant dû à des déviations process est rare, soit inférieur à 100dppm[1]. Ces approches seront ensuite comparés en calculant leurs précisions sur différents échantillons du circuit simulé.

La première approche qui sera présentée est basée sur la régression statistique, la deuxième approche est basée sur la distribution multinormale des performances et des critères de test et la dernière approche est basée sur le même principe de la deuxième approche avec quelques modifications permettant de considérer la distribution multinormale uniquement pour les performances. Avant de présenter ces techniques, nous commençons d'abord par donner quelques généralités sur la loi multinormale, puis nous présenterons notre cas d'étude, qui représente un amplificateur opérationnel différentiel.

4.2 La loi multinormale

4.2.1 Généralités

Dans cette section nous allons définir quelques concepts de base sur la loi multinormal. Soit un vecteur $X = (X_1, X_2, ..., X_p)^T$ composé de p variables aléatoires, où X_j pour $j = 1, 2, ..., p$, est une variable aléatoire sur une dimension. La covariance de X_i et X_j représente le taux de dépendance de ces deux variables et elle est définie comme suit :

$$\nu_{X_iX_j} = Cov(X_i, X_j) = E(X_i \cdot X_j) - E(X_i) \cdot E(X_j) \tag{4.1}$$

où $E(.)$ représente l'espérance. Si X_i et X_j sont indépendantes, la covariance $\nu_{X_iX_j}$ est nécessairement égale à zéro. L'inverse n'est pas vrai. La covariance de la variable aléatoire X_i avec elle même est la variance :

$$\nu_{X_iX_i} = Cov(X_i, X_i) = \nu_{X_i} \tag{4.2}$$

Le coefficient de corrélation entre deux variables X_i et X_j est défini à partir de la covariance comme suit :

$$\rho_{X_iX_j} = \frac{\nu_{X_iX_j}}{\sigma_{X_i}\sigma_{X_j}} \tag{4.3}$$

où l'écart-type σ_{X_i} est égal à $\sqrt{\nu_{X_i}}$

L'avantage du coefficient de corrélation est qu'il est indépendant de l'échelle utilisée. C'est-à-dire, changer l'échelle de mesure des variables ne change pas la valeur du coefficient de corrélation.

[1] defective part per million, qui représente le nombre de circuits défaillants sur un million.

Par conséquent, le coefficient de corrélation est plus utile comme mesure de l'association entre deux variables que la covariance. Le coefficient de corrélation en valeur absolue est toujours inférieur ou égal à 1. Elle est proche de zéro si les variables aléatoires X_i et X_j sont indépendantes.

L'estimation empirique de ces quantités nécessite un certain nombre d'observations. Supposons que $\{x_i\}_{i=1}^n$ est un ensemble de n observations du vecteur de variables X dans $\mathbb{R}^p$. Chaque observation x_i a p dimensions : $x_i = (x_{i_1}, x_{i_2}, ..., x_{i_p})$, et elle correspond à une valeur observée du vecteur $X \in \mathbb{R}^p$. La covariance entre deux variables aléatoires est ainsi estimée par :

$$V_{X_iX_j} = \frac{1}{n-1}\left(\sum_{k=1}^{n} x_{i_k}\ x_{j_k} - n\overline{x}_i \cdot \overline{x}_j\right) \tag{4.4}$$

et la variance d'une variable aléatoire est estimée par :

$$V_{X_i} = \frac{1}{n-1}\left(\sum_{k=1}^{n} x_{i_k}^2 - n\overline{x}_i^2\right) \tag{4.5}$$

Le coefficient de corrélation entre deux variables aléatoires est ainsi donné par :

$$r_{X_iX_j} = \frac{V_{X_iX_j}}{s_{X_i}s_{X_j}} \tag{4.6}$$

avec $s_{X_i} = \sqrt{V_{X_i}}$.

Les covariances théoriques entre toutes les variables aléatoires peuvent être représentées par une matrice Σ appelée la matrice Variance-Covariance :

$$\Sigma = \begin{pmatrix} \nu_{X_1} & \cdots & \nu_{X_1X_p} \\ \vdots & \ddots & \vdots \\ \nu_{X_1X_p} & \cdots & \nu_{X_p} \end{pmatrix} \tag{4.7}$$

L'estimation (empirique) de la matrice variance-covariance est alors donnée par :

$$S = \begin{pmatrix} V_{X_1} & \cdots & V_{X_1X_p} \\ \vdots & \ddots & \vdots \\ V_{X_1X_p} & \cdots & V_{X_p} \end{pmatrix} \tag{4.8}$$

Soit X une variable aléatoire à p dimensions de moyenne $\mu = (\mu_{i_1}, \mu_{i_2}, ..., \mu_{i_p})^T$ et de matrice de variance-covariance Σ. Si la distribution de X suit une loi multinormale, alors la fonction densité de probabilité $f(x)$ est définie par :

$$f(x) = \frac{1}{\sqrt{det(2\pi\Sigma)}} \cdot exp\left[-\frac{(x-\mu)^T\Sigma^{-1}(x-\mu)}{2}\right] \tag{4.9}$$

La probabilité d'un ensemble $A \in \mathbb{R}^p$ est donnée par la formule d'intégrales multiples suivantes :

$$\mathbf{P}(A) = \frac{1}{\sqrt{det(2\pi\Sigma)}} \int_{A_1} \cdots \int_{A_p} exp\left[-\frac{(x-\mu)^T\Sigma^{-1}(x-\mu)}{2}\right] dx_1 dx_2 \cdots dx_p \tag{4.10}$$

4.2.2 Estimation de la moyenne par intervalle de confiance

Estimer une valeur d'un paramètre inconnu θ par intervalle de confiance représente l'une des approches les plus préférées dans la pratique car elle introduit la notion d'incertitude. On cherche à déterminer l'intervalle $[a, b]$ centré sur la valeur numérique estimée du paramètre inconnu contenant la valeur vraie avec une probabilité α fixée a priori. Cette probabilité permet de s'adapter aux exigences de l'application.

$$\mathbf{P}(a < \theta < b) = \alpha \tag{4.11}$$

L'intervalle $[a, b]$ est appelé *intervalle de confiance* et α est le *coefficient de confiance*. Une estimation par intervalle de confiance sera d'autant meilleure que l'intervalle sera petit pour un coefficient de confiance grand.

La donnée de départ, outre l'échantillon, sera la connaissance de la loi de probabilité du paramètre à estimer.

On supposera dans ce qui suit que la moyenne suis une loi normale. En outre, si le nombre de réalisations dans l'échantillon est supérieur à 30, cette supposition est vérifiée en utilisant le théorème central limite (TCL) [62]. L'estimation de la moyenne d'une variable aléatoire par intervalle de confiance dépend de si la variance σ^2 de cette variable est connue ou non.

Cas où la variance σ^2 est connue

Soit une variable aléatoire X suivant une loi normale de moyenne μ inconnue et d'écart-type σ. On dispose d'un échantillon de n réalisations x_i de cette variable aléatoire.

Dans ce cas où la variance est connue, la valeur σ est une constante dans la formule de l'intervalle de confiance et la nouvelle variable aléatoire $Y = \frac{(m-\mu)\sqrt{n}}{\sigma}$ suit une loi normale centrée et réduite. La valeur de t est lue dans une table de la loi normale centrée et réduite. L'intervalle de confiance sur la moyenne est :

$$\mathbf{P}(a < \mu < b) = \mathbf{P}\left[m - t\frac{\sigma}{\sqrt{n}} < \mu < m + t\frac{\sigma}{\sqrt{n}}\right] = \alpha \tag{4.12}$$

où m est la moyenne arithmétique calculée à partir de l'échantillon.

Cas où la variance σ^2 est inconnue

Dans ce cas, σ est une variable aléatoire. Soit s^2 l'estimateur sur l'échantillon de σ^2 que l'on obtient par :

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - m)^2 \tag{4.13}$$

Comme X suit une loi normale, on sait que la quantité $n\frac{s^2}{\sigma^2}$ suit une loi du χ^2 à $n-1$ degrés de liberté. La nouvelle variable aléatoire $Y = \frac{(m-\mu)\sqrt{n}}{s}$ suit alors une loi de Student à $n-1$ degrés de liberté. L'intervalle de confiance est alors :

$$\mathbf{P}(a < \mu < b) = \mathbf{P}\left[m - t\frac{s}{\sqrt{n}} < \mu < m + t\frac{s}{\sqrt{n}}\right] = \alpha \tag{4.14}$$

où t correspondant au risque α est lue dans une table de Student pour $n-1$ degrés de liberté.

4.3 Cas d'étude : amplificateur opérationnel différentiel

Nous allons présenter dans ce qui suit un amplificateur opérationnel complètement différentiel qui nous servira de véhicule de test dans le but de valider les techniques proposées dans le cadre de cette thèse. Cet amplificateur a été conçu sous la technologie $0.18\mu m$ CMOS de ST Microelectronics [95]. Il est formé de quatre blocs principaux : le circuit d'alimentation[2], le circuit de déclenchement[3], le circuit de contrôle du mode en commun[4] et l'amplificateur lui même. La Figure 4.2 illustre ce circuit.

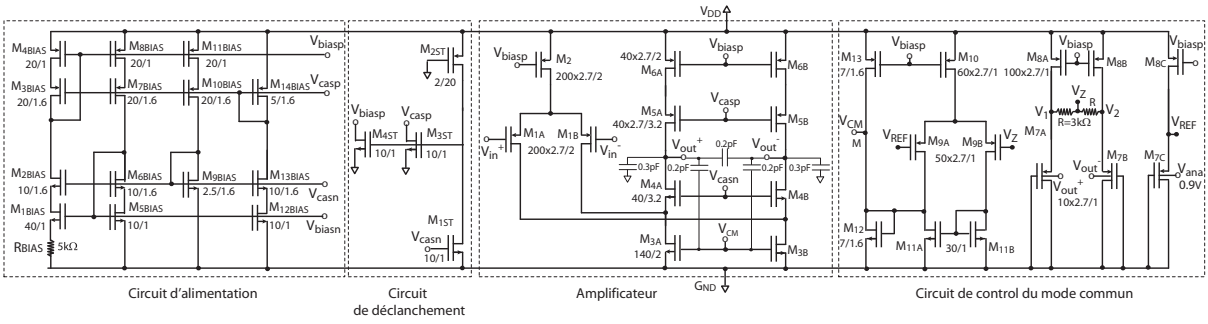


FIG. 4.2 – Amplificateur opérationnel complètement différentiel. Les dimensions de chaque transistor W/L sont de multiples de la taille unité de transistor ($W_{UNIT} = 0.28\mu m$ et $L_{UNIT} = 0.18\mu m$).

Dans la majorité des cas, les lois de probabilité des variations des paramètres process d'un circuit sont connues. En effectuant les simulations Monte Carlo, les différentes lois de probabilité des variations des performances ainsi que celles des critères de test pourront être calculées, en conséquence les différentes bornes des spécifications ainsi que celles des limites des tests peuvent être déduites. Seulement, cette opération nécessite un grand nombre de simulations pour assurer une précision de l'ordre de ppm pour les métriques de test. Dans le cas où les simulations prennent un temps trop grand, cette méthode semble impossible à utiliser. Il est possible d'effectuer un nombre réduit de simulations, puis utiliser les tests d'ajustement statistiques pour confirmer la distribution des différentes lois estimées, tels que le test de Kolmogorov-Smirnov ou bien le test du Khi-deux. Les techniques à base de la sensibilité peuvent aussi être utilisées pour déterminer ces bornes.

Pour ce cas d'étude, nous avons fixé les spécifications des performances à $4.3\sigma_i$ de leur moyenne μ_i, σ_i étant l'écart-type de chaque performance i calculé en utilisant des simulations de types Monte Carlo (ce qui est le cas dans le cadre de cette thèse). Si N simulations Monte Carlo sont effectuées, alors ces paramètres sont estimés comme suit : Soit $x_{i,(i=1,\cdots,N)}$ la valeur de la mesure ou de la performance considérée pour la $i^{ème}$ simulation. Nous estimons la moyenne $\hat{\mu}$ des x_i par :

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{4.15}$$

Ensuite, nous estimons l'écart-type $\hat{\sigma}$ des x_i comme suit :

$$\hat{\sigma} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \hat{\mu})^2} \tag{4.16}$$

[2]Bias circuit.
[3]Start-up circuit.
[4]Common-mode contrôle circuit.

Les bornes de la spécification considérée seront donc : $[\mu - t\sigma, \mu + t\sigma]$ où t est choisi selon l'application. Pour notre cas, nous avons effectué 1000 simulations Monte Carlo ($N = 1000$) et l'ensemble des performances considéré est donné dans le Tableau 4.1, où $a_1 = \mu - 4.3\sigma$ et $a_2 = \mu + 4.3\sigma$ représentent les bornes de chaque spécification. Les spécifications de l'amplificateur ne sont pas connues à priori, car l'application de ce circuit n'est pas considérée dans ce travail. Donc, nous avons fixé les bornes des spécifications nous mêmes de sorte à obtenir un très grand Rendement de design $Y = 99.99\%$ quand toutes les performances sont considérées à la fois. Ce qui explique les intervalles des spécifications fixés à $\mu \pm 4.3\sigma$. Pour ce qui est des critères de test présentés par le Tableau 4.2, les limites de test seront fixées à l'aide des techniques d'estimation des métriques de test présentées ci-dessous.

Performance	**Banc de test**	μ	σ	**Spécifications**	
				a_1	a_2
1. *Gain* A_D $(W = 100Hz)$		76.60dB	0.493dB	74.49dB	78.71dB
2. *Gain BandWidth* GBW_D	1	330MHz	18.14MHz	252.36MHz	407.64MHz
3. *Marge de phase*		63.33°	0.45°	61.40°	65.26°
4. $CMRR$	2	-42.76dB	1.02dB	-47.13dB	-38.39dB
5. $PSRR$ (G_{ND})	3	-29.99dB	3.65dB	-45.61dB	-14.37dB
6. $PSRR$ (V_{DD})	4	-28.21dB	3.75dB	-44.26dB	-12.16dB
7. THD	5	66.19dB	2.38dB	56.00dB	76.38dB
8. *Courant* (I_{DD})		2.48mA	0.21mA	1.58mA	3.38mA
9. *Intermodulation*	6	67.57dB	1.09dB	62.90dB	72.24dB
10. *Slew Rate SR+*	7	73.14V/μs	5.55V/μs	49.38V/μs	96.88V/μs
11. *Slew Rate SR−*		73.14V/μs	5.55V/μs	49.38V/μs	96.88V/μs
12. *Bruit*	8	39.22μV	0.5μV	37.08μV	41.36μV

TAB. 4.1 – Les performances de l'amplificateur, les paramètres statistiques de leurs Gaussiennes et leurs spécifications ajustées et fixées à 4.3σ.

Critère de test	**Banc de test**	μ	σ	**Limites de test**	
				b_1	b_2
1. SNDR	9	68.85 dB	2.19 dB	à déterminer	à déterminer
2. Offset		0 μV	7.69 μV	à déterminer	à déterminer

TAB. 4.2 – Les critères de test considérés pour l'amplificateur et les paramètres statistiques de leurs Gaussiennes.

La Figure 4.3(a) montre un exemple de la distribution du *Gain* et la Figure 4.3(b) montre un exemple de la distribution du THD[5].

4.4 Estimation des métriques de test par régression : *Méthode 1*

Pour un certain nombre typique de simulations Monte Carlo, les métriques de test ne peuvent pas être estimées à partir de l'échantillon en utilisant les équations (2.51) à (2.54), car il n'y a pratiquement aucun circuit défaillant dans l'échantillon. Nous avons envisagé donc d'élargir l'espace des valeurs des performances en augmentant l'écart-type de chaque performance. Ceci, peut se faire en augmentant les valeurs des paramètres physiques qui se trouvent dans le fichier technologie du circuit sous test à simuler. Dans notre cas d'étude, les paramètres physiques de

[5]Total Harmonic Distortion.

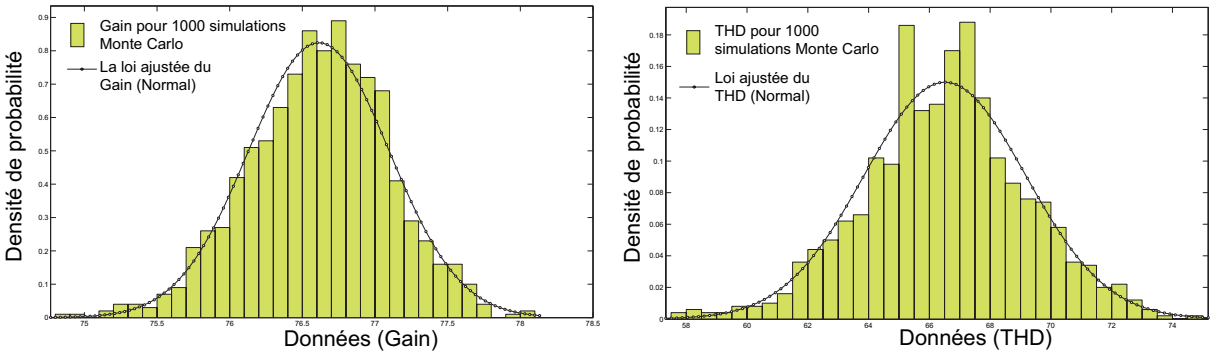


FIG. 4.3 – La distribution (a) du *Gain* et (b) du THD de l'amplificateur.

la technologie ont des distributions Normales. Donc, nous avons augmenté les écart-types de ces distributions par des valeurs allant de $\alpha = 2$ à $\alpha = 3$ (avec un pas de 0.1). Nous avons ensuite calculé les métriques de test pour chaque valeur de l'écart-type. Une fois toutes ces métriques calculées, nous avons effectué une régression afin d'obtenir les valeurs des métriques de test pour le cas nominal (ie. $\alpha = 1$). La Figure 4.4(a) montre la valeur du Taux de défauts trouvée par régression et qui est de $1.28\% = 12800ppm$ et celle de la métrique Perte de rendement est montrée par la Figure 4.4(b) qui est de $0.03\% = 300ppm$. Sachant que les spécifications sont fixées pour avoir un Rendement de $100dppm$, ces valeurs ne sont pas acceptables.

L'inconvénient de cette méthode est que les valeurs des métriques de test dépendent du modèle de régression utilisé et des paramètres de ce modèle aussi. Pour un modèle donné, une légère modification de ses paramètres entraîne un changement important des valeurs des métriques de test. Ce qui fait que les métriques de test ne sont pas calculées avec une grande précision. En plus, si le calcul doit être fait en ppm, cette méthode est fortement déconseillée. D'où le développement de la technique de la section suivante (*Méthode 2*).

4.5 Estimation des métriques de test sous l'hypothèse multinormale : *Méthode 2*

Dans cette technique, nous considérons que la distribution (ou la fonction densité de probabilité) des performances et des critères de test est multinormale. Quelques notions de base sur la loi multinormale sont définies en Section 4.2.

Il est à noter que la détermination du modèle statistique du circuit sous test ne nécessite pas forcément l'utilisation des techniques de type Monte Carlo. Il existe d'autres approches qui sont basées sur l'utilisation de l'analyse de sensibilité pour identifier d'une manière déterministe les bornes sur les valeurs des paramètres du circuit [16]. L'information sur le process et la sensibilité des composants principaux du circuit a été récemment considérée dans [100] pour générer un modèle statistique des circuits sans fautes et avec fautes, qui est ensuite utilisée pour la génération de vecteurs de test. Ces modèles sont obtenus en utilisant une approche statistique et l'estimation linéaire, au lieu des simulations Monte Carlo, mais ils peuvent manquer de précision pour le cas de déviations paramétriques importantes. Une autre approche statistique est considérée dans [133]. Dans celle-ci, par contre, les fautes paramétriques sont injectées par un échange des transistors, un seul à la fois, par d'autres transistors ayant des paramètres process décalés de 3σ et l'analyse de sensibilité est effectuée uniquement dans le domaine DC. Le problème avec cette approche est que le mauvais comportement résultant de plusieurs petites déviations ne peut pas être évalué précisément.

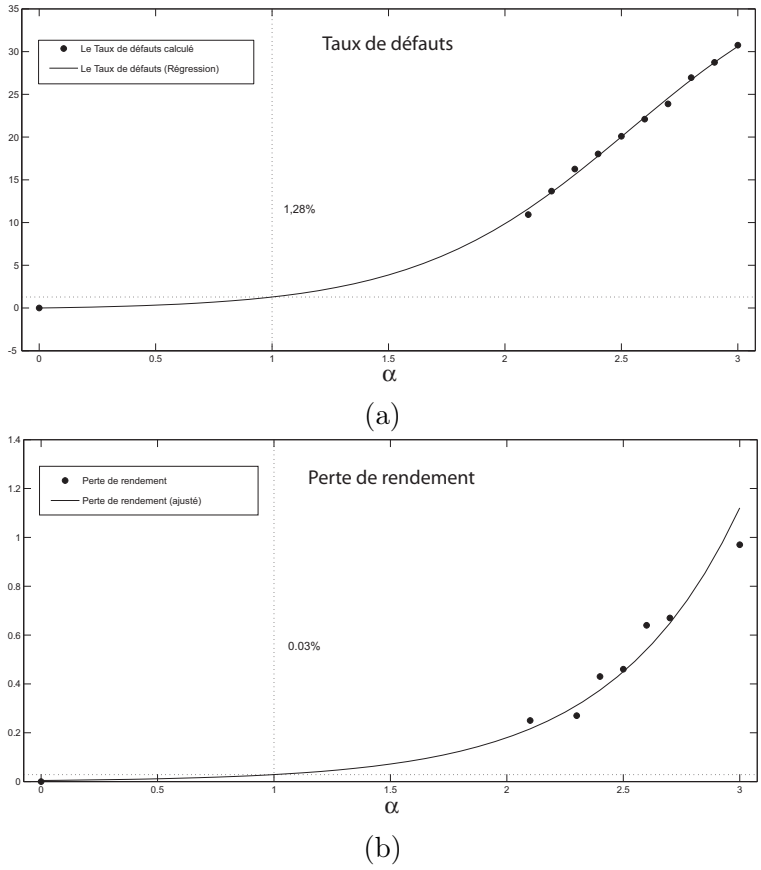


FIG. 4.4 – Calcul des métriques (a) Taux de défauts et (b) Perte de rendement par régression.

4.5.1 Validation de l'hypothèse multinormale

Validation graphique et par tests classiques

L'une des méthodes utilisées pour valider une distribution multinormale est la méthode graphique. Elle nécessite de trouver graphiquement la loi qui s'ajuste à chaque performance et à chaque critère de test. Il est aussi possible d'utiliser d'autres tests tels que le test de Kolmogorov-Smirnov ou le test du Khi-deux qui permettent de valider la loi normale pour chaque performance ou critère de test séparément après décorrélation.

Validation par facteur de corrélation

Pour trouver la plage de validité de l'hypothèse que la densité de probabilité des performances et des critères de test est une multinormale nous avons effectué plusieurs simulations Monte Carlo et, pour chaque cas, l'écart-type des paramètres physiques du circuit est augmenté par un facteur de $\alpha > 0$ par rapport à l'écart-type du cas nominal (ou initial). Ce qui permettra d'augmenter l'écart-type des performances et des critères de test. La Figure 4.5 montre un exemple d'une performance élargie. L'écart-type initial σ_1 a été augmentée à $\sigma_2 = \alpha\sigma_1$.

La simulation Monte Carlo du circuit a été effectuée pour des valeurs différentes de α entre 1 et 3 (avec un pas de 0.1), montrant que le facteur de corrélation entre les performances et les critères de test reste la même que celui du cas initial.

Les Figures 4.6(a-j) montrent le coefficient de corrélation entre chaque performance et le THD et la Figure 4.7 montre le coefficient de corrélation entre le critère de test SNDR et la performance THD, pour les valeurs de α situées entre 1 et 3. Pour le cas nominal (ie. $\alpha = 1$), comme exemple,

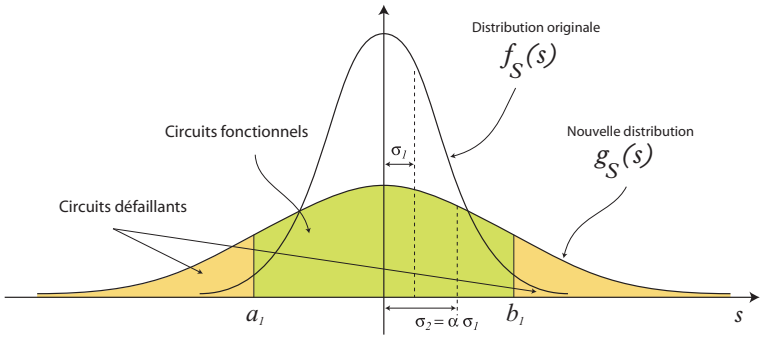


FIG. 4.5 – Exemple d'une performance élargie.

	AD	$GBWD$	PM	$CMRR$	$PSRR1$	$PSRR2$	THD	I_{DD}	$Intermod$	SR	$Noise$
AD	1.0000	0.7126	0.2728	0.3575	0.4419	0.4979	0.8646	0.3085	0.7563	0.7838	0.4644
$GBWD$	0.7126	1.0000	0.3878	0.7721	0.7808	0.7979	0.9550	0.8285	0.7664	0.9914	0.0628
PM	0.2728	0.3878	1.0000	0.1843	0.2188	0.2360	0.3760	0.2470	0.3334	0.4072	0.3281
$CMRR$	0.3575	0.7721	0.1843	1.0000	0.9714	0.9578	0.7258	0.8938	0.5280	0.7569	0.4419
$PSRR1$	0.4419	0.7808	0.2188	0.9714	1.0000	0.9971	0.7649	0.8047	0.5831	0.7782	0.2945
$PSRR2$	0.4979	0.7979	0.2360	0.9578	0.9971	1.0000	0.7968	0.7830	0.6152	0.8006	0.2324
THD	0.8646	0.9550	0.3760	0.7258	0.7649	0.7968	1.0000	0.7113	0.8195	0.9802	0.1585
I_{DD}	0.3085	0.8285	0.2470	0.8938	0.8047	0.7830	0.7113	1.0000	0.4988	0.7908	0.4765
$Intermod$	0.7563	0.7664	0.3334	0.5280	0.5831	0.6152	0.8195	0.4988	1.0000	0.7958	0.2424
SR	0.7838	0.9914	0.4072	0.7569	0.7782	0.8006	0.9802	0.7908	0.7958	1.0000	0.0991
$Noise$	0.4644	0.0628	0.3281	0.4419	0.2945	0.2324	0.1585	0.4765	0.2424	0.0991	1.0000

(4.17)

$PSRR1 = PSRRG_{ND}$ et $PSRR2 = PSRRV_{DD}$.

TAB. 4.3 – Matrice de corrélation en valeurs absolues.

le Tableau 4.3 montre la matrice variance-covariance en valeurs absolues contenant les coefficients de corrélation entre les différentes performances présentées par le Tableau 4.1. Puisque les valeurs du $SR+$ sont exactement les mêmes que celles du $SR-$ nous avons éliminé de cette matrice le $SR-$, ainsi la matrice contient 11 lignes et 11 colonnes. Ces figures montrent clairement que pour un rapport $\alpha < 2$ l'hypothèse de la multinormale est vérifiée. Par contre, pour $\alpha \geq 2$ cette hypothèse est non vérifiée (voir par exemple la Figure 4.6(i)). Ceci correspond au cas quand les paramètres physiques sont sujets à des perturbations importantes, mais ceci est peu probable en réalité pour des paramètres multiples. Pour le cas d'un seul paramètre, ceci correspond à une faute paramétrique simple. Nous analyserons ces fautes dans le chapitre suivant.

Les Figures 4.8(a) et (b) montrent les distributions des performances augmentées par rapport aux distributions normales, il est clair que les nouvelles lois sont différentes, pour le cas (a) où la nouvelle distribution représente une loi *Lognormale* et pour le cas (b) où la nouvelle distribution représente une loi *Valeur extrême*. Les déviations très larges peuvent être proprement considérées en utilisant la supposition de fautes paramétriques simples.

Il est à noter que dans la Figure 4.6(j) la corrélation diminue légèrement avec l'augmentation du rapport α, mais ceci est causé par le fait que les performances THD et $Bruit$ sont pratiquement non corrélées. En résumé, ces résultats indiquent que les petites déviations paramétriques multiples peuvent être étudiées proprement par un modèle multinormal pour le cas d'étude considéré.

4.5.2 Précision sur le calcul des métriques de test

Exemple de comparaison avec un calcul théorique

Nous avons besoin de trouver les limites de test séparant les circuits défaillants des fonctionnels, en fonction des métriques de test requises. Un compromis entre le Taux de défauts et la Perte de rendement doit être considéré. Ceci permettra de fixer les limites de test. Les limites

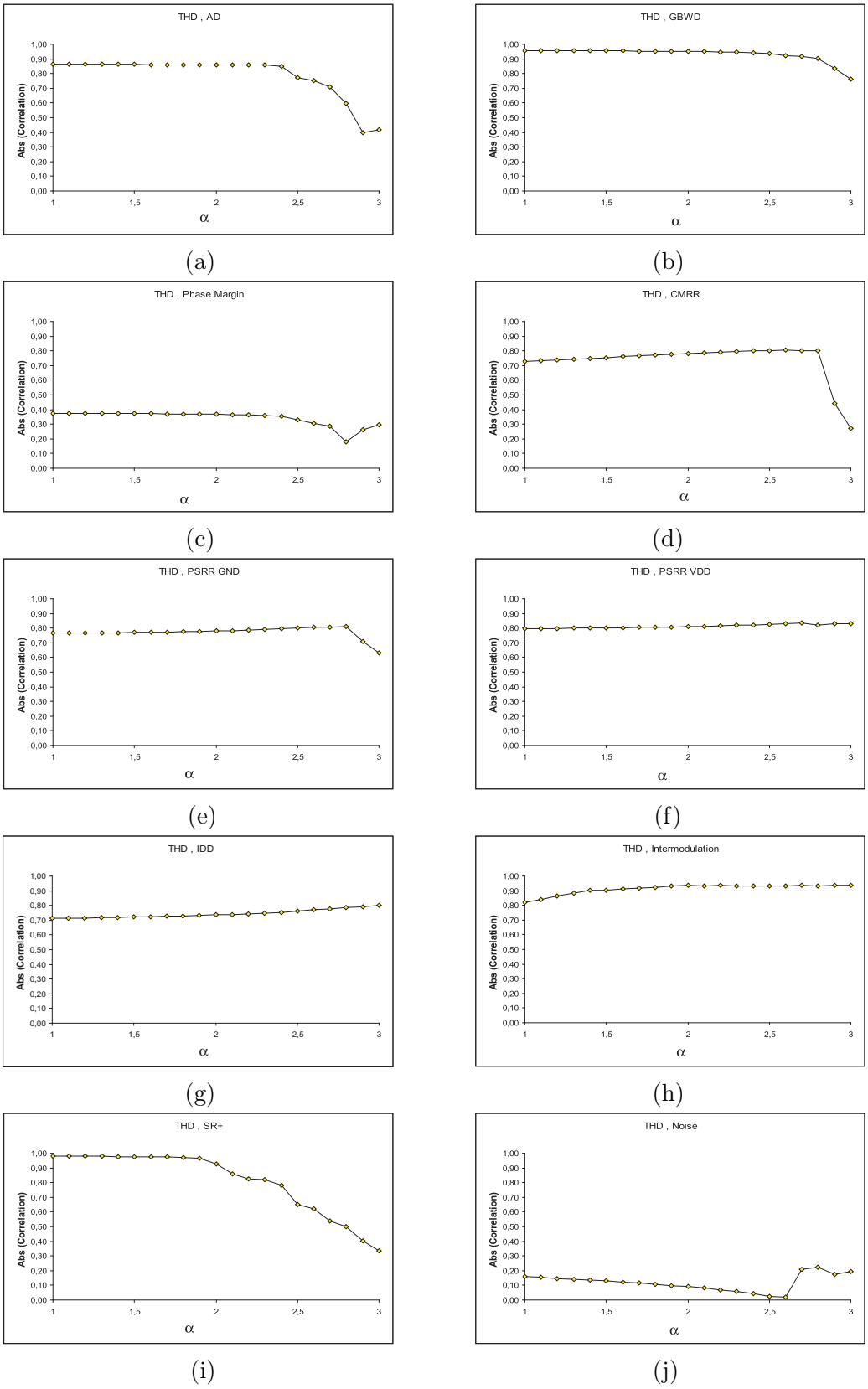


FIG. 4.6 – Coefficients de corrélation en valeurs absolue entre le THD et les autres performances du circuit en fonction du rapport d'augmentation de l'écart-type α.

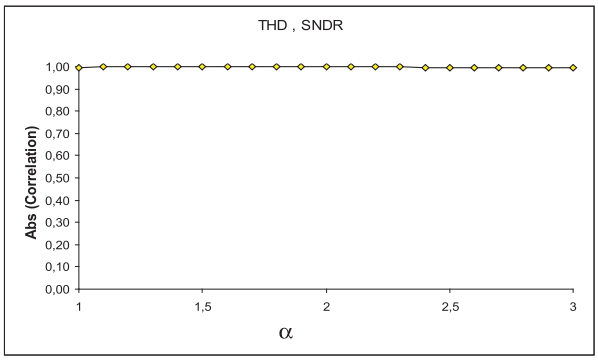
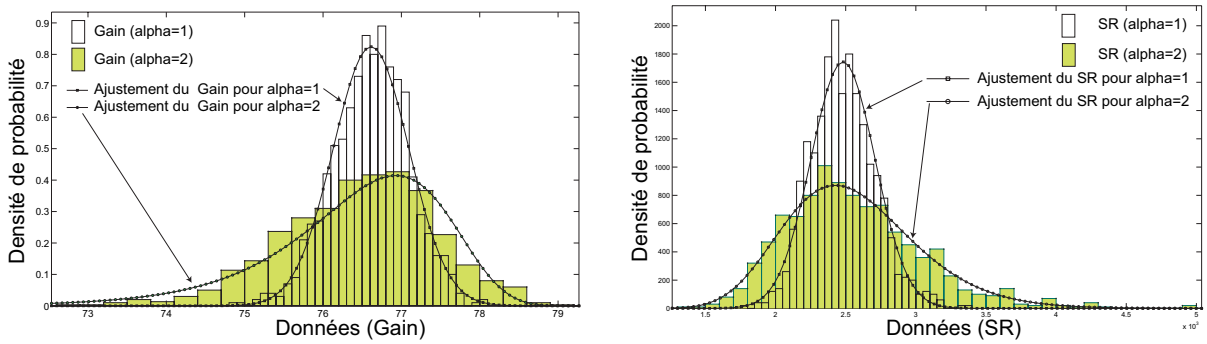


FIG. 4.7 – Corrélation absolue entre le THD et le critère de test $SNDR$ en fonction du rapport d'augmentation de l'écart-type α.



(a) (b)

FIG. 4.8 – La distribution (a) du *Gain* et (b) du SR de l'amplificateur pour $\alpha = 1$ qui représente le cas normal et $\alpha = 2$ qui représente le cas d'un élargissement.

d'un critère de test sont données par un intervalle $\mu_i \pm k_i\sigma_i$. Pour des raisons de simplicité, nous représenterons désormais les limites de test uniquement par k_i.

Pour illustrer la précision de notre approche, nous devrions comparer les résultats obtenus par celle-ci aux résultats théoriques, ie. par calcul direct des intégrales des équations (2.47)-(2.50), où le nombre des performances considérées est 11 et uniquement un seul critère de test est considéré. Ceci nécessite de résoudre des équations intégrales multiples d'ordre 12 (12 intégrations). Il n'est pas possible d'effectuer l'intégration avec un tel nombre d'intégrales. Nous illustrerons un cas simple où seulement deux performances. La *Marge de phase* PM et le THD, et un critère de test SNDR sont considérées. Ainsi, les métriques de test peuvent être calculées directement par les équations (2.47), (2.48), (2.49) et (2.50) comme suit :

$$Y = \int_{A_3} \int_{A_7} f(s_1, s_2)\ ds_1\ ds_2 \tag{4.18}$$

$$Y_T = \int_{B_1} f(t_1)\ dt_1 \tag{4.19}$$

$$Y_L = 1 - \frac{J}{Y} \tag{4.20}$$

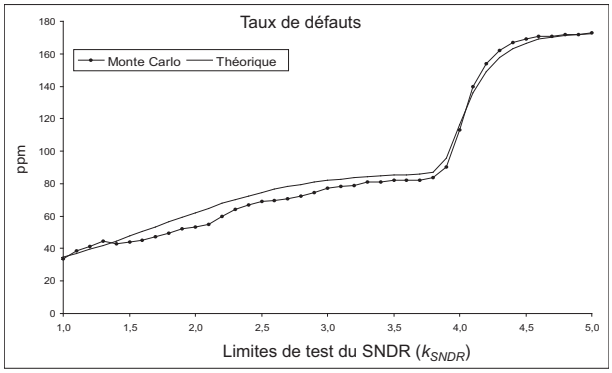
$$D = 1 - \frac{J}{Y_T} \tag{4.21}$$

où,

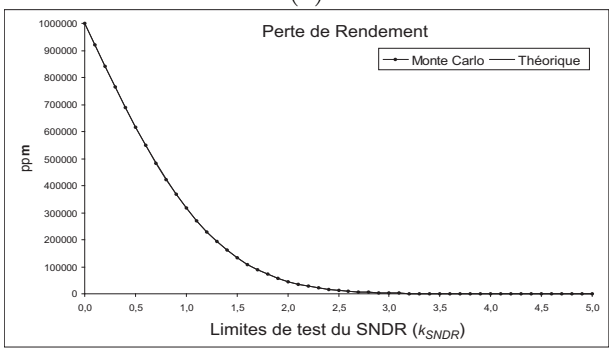
$$J = \int_{A_3} \int_{A_7} \int_{B_1} f(s_1, s_2, t_1)\ ds_1\ ds_2\ dt_1 \tag{4.22}$$

$s_1 = PM$, $s_2 = THD$, $t_1 = SNDR$, A_i représente la $i^{ème}$ spécification, B_1 est la limite de test du $SNDR$ et $f(.)$ est calculée par (4.9).

Pour calculer ces métriques de test en utilisant notre approche basée sur la loi multinormale des performances et des critères de test, un programme écrit sous Matlab[6] est utilisé pour générer des millions d'instances suivant la même densité de probabilité multinormale, à partir desquelles les métriques de test peuvent être directement estimées en utilisant leurs estimateurs donnés par (2.51), (2.52), (2.53) et (2.54). Les Figures 4.9(a) et (b) montrent que les valeurs estimées et les valeurs théoriques sont très proches.



(a)



(b)

FIG. 4.9 – Comparaison des métriques de test estimées et théoriques pour le cas de deux spécifications (*Marge de phase* et THD) et un seul critère de test (SNDR) : (a) Taux de défauts et (b) Perte de rendement.

En considérant toutes les performances et les critères de test, nous n'allons pas appliquer l'intégration directe qui est impossible. Nous allons utiliser l'estimation sur un million de circuits générés directement d'une loi multinormale. Pour illustration, la Figure 4.10 montre la distribution de la *Marge de phase* et du $SNDR$ pour le cas de 1000 circuits générés par la simulation Monte Carlo et de 1000 circuits générés par la loi multinormale. Cette figure montre clairement que les deux distributions sont quasiment identiques. Le même résultat a été obtenu pour les autres performances et critères de test.

La Figure 4.11 montre la génération de 1000 et d'un million d'instances de circuits ayant une loi multinormale. Il est clair qu'avec un million d'instances, la précision à l'ordre de ppm peut être atteinte.

[6]On peut aussi utiliser le logiciel R ou un autre logiciel statistique.

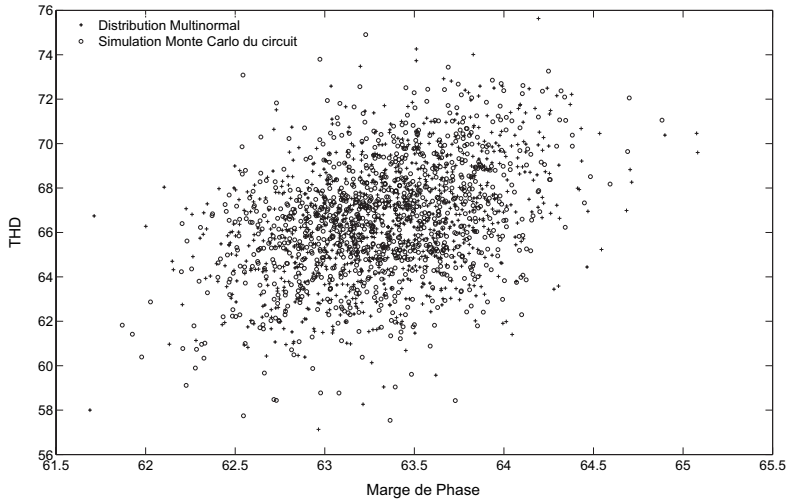


FIG. 4.10 – Distribution de 1000 circuits générés par la simulation Monte Carlo du circuit et de la loi multinormale.

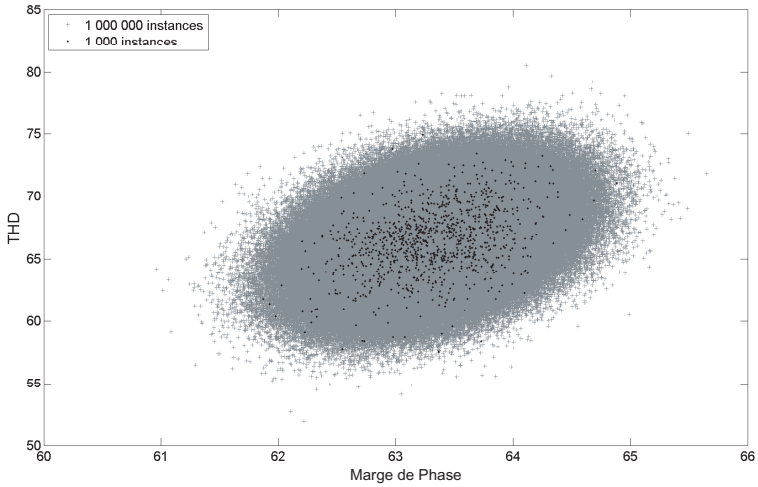


FIG. 4.11 – Génération de 1000 et 1 million de circuits ayant une distribution multinormale.

Calcul des intervalles de confiance des métriques de test

Nous allons présenter dans ce qui suit les précisions sur les calculs des métriques de test. Cette précision sera donnée sous forme d'inférences sur les valeurs estimées. C'est-à-dire, assurer que la valeur calculée ne dévie pas de plus d'un certain pourcentage (par exemple de 5%) de la valeur réelle qui n'est pas connue. Nous allons présenter des intervalles de tolérance de telle sorte que les valeurs estimées des métriques de test se trouveront à $99,73\%$ de confiance dans ces intervalles. Mais avant tout, il est à noter qu'avant de calculer les précisions sur les calculs des métriques de test, nous allons d'abord calculer les précisions sur la moyenne des performances et des critères de test estimés à base des 1000 simulations Monte Carlo effectuées pour l'amplificateur. La technique utilisée pour calculer les intervalles de confiance est présentée en Section 4.2.2. Les intervalles de confiance des moyennes des performances (du Tableau 4.1) estimées sont donnés par le Tableau 4.4.

Performances de l'amplificateur	**Intervalle de confiance sur la moyenne μ (niveau de confiance de 99.73%)**
1. *Gain* A_D	$[76.56, 76.65]$
2. *Gain BandWidth* GBW_D	$[3.28 \cdot 10^8, 3.31 \cdot 10^8]$
3. *Marge de phase* ϕ_D (PM)	$[63.26, 63.35]$
4. $CMRR$	$[-42.87, -42.68]$
5. $PSRR$ (G_{ND})	$[-30.41, -29.73]$
6. $PSRR$ (V_{DD})	$[-28.67, -27.97]$
7. THD	$[66.25, 66.76]$
8. *Courant* (I_{DD})	$[2.45 \cdot 10^{-3}, 2.50 \cdot 10^{-3}]$
9. *Intermodulation*	$[67.53, 67.76]$
10. *Slew Rate* $SR+$ $(C_L = 1pF)$	$[7.19 \cdot 10^7, 7.31 \cdot 10^7]$
11. *Slew Rate* $SR-$ $(C_L = 1pF)$	$[7.19 \cdot 10^7, 7.31 \cdot 10^7]$
12. *Bruit*	$[3.92 \cdot 10^{-5}, 3.93 \cdot 10^{-5}]$

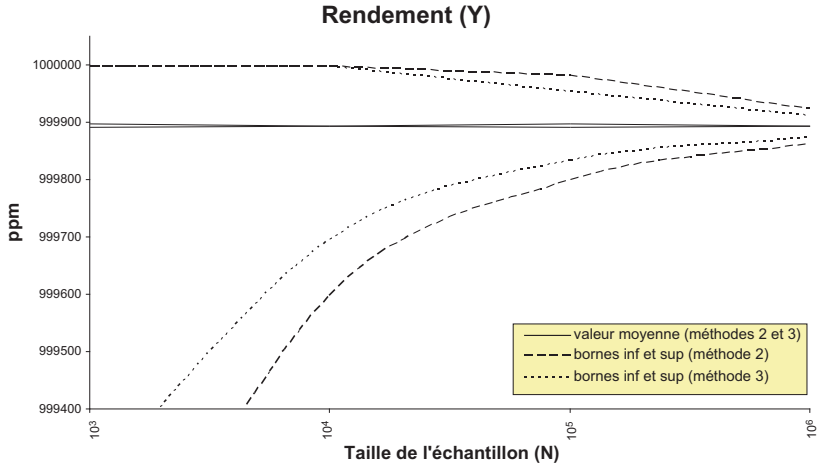
TAB. 4.4 – Les intervalles de confiance des valeurs moyennes estimées des performances de l'amplificateur, avec un niveau de confiance de 99.73%.

Pour ce qui est de la précision sur le calcul des métriques de test, nous allons nous baser sur la variance. Plus elle est petite, plus la valeur estimée est proche de la valeur réelle. Pour calculer cette variance pour chaque métrique de test, nous avons générer plusieurs fois (1000 fois) des échantillons de circuits (amplificateurs) de tailles différentes (1000, 10^4, 10^5 et 1 million) et pour chaque génération, nous avons estimé les métriques de test (Rendement, Taux de défauts et Perte de rendement). Nous avons calculé la moyenne de chaque métrique et puis fixé ensuite les intervalles de tolérance à $\pm 3\sigma$ de cette moyenne. Ce qui signifie que, les valeurs estimées des métriques de test appartiennent à 99.73% à ces intervalles de confiance. Les intervalles de confiance calculés sont présentés par les courbes à traits discontinus des Figures 4.12(a), (b) et (c).

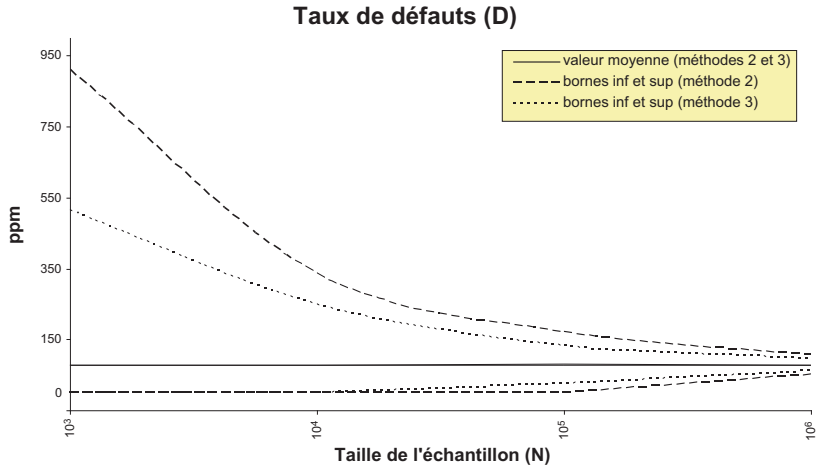
4.5.3 Fixation des limites de test

Les limites de test peuvent être fixées comme étant un compromis entres les métriques de test comme le Taux de défauts et la Perte de rendement. Nous allons considérer dans ce qui suit deux critères de test, nous allons ajouter à l'$SNDR$, le courant I_{DD} étant donné qu'il permet d'améliorer la Couverture de fautes comme il sera présenté dans la Section 5.4.5. La Figure 4.13(a) montre le Taux de défauts en fonction des limites de test k_{SNDR} de l'$SNDR$ et les limites de test $k_{I_{DD}}$ de l'I_{DD}. La Figure 4.13(b) montre la Perte de Rendement en fonction des limites de test k_{SNDR} et $k_{I_{DD}}$.

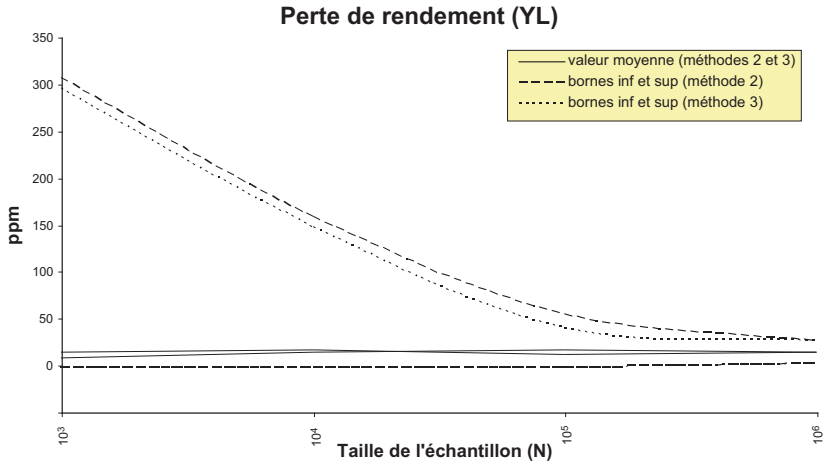
La fixation des limites de test est toujours un compromis entre le coût de test et la qualité de test. Ici, juste un exemple pour notre cas d'étude, nous considérerons les limites de test où le Taux de défauts est égal à la Perte de rendement. La Figure 4.14 montre le Taux de défauts et le Perte de rendement ensemble en fonction des limites de test de l'$SNDR$ et de l'I_{DD}. Nous nous nous intéresserons à leur intersection. Cette intersection (points où le Taux de défauts est égal à la Perte de rendement) est tracée dans la Figure 4.15. Nous avons choisi comme compromis des limites de test de l'$SNDR$ et de l'I_{DD} le minimum de ces points qui est égal à $55ppm$. Ceci résulte des limites de test $k_{SNDR} = 4.0$ pour l'$SNDR$ et $k_{I_{DD}} = 4.1$ pour l'I_{DD}.



(a)

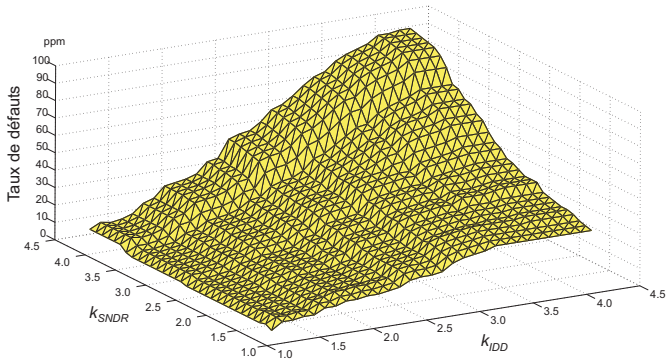


(b)

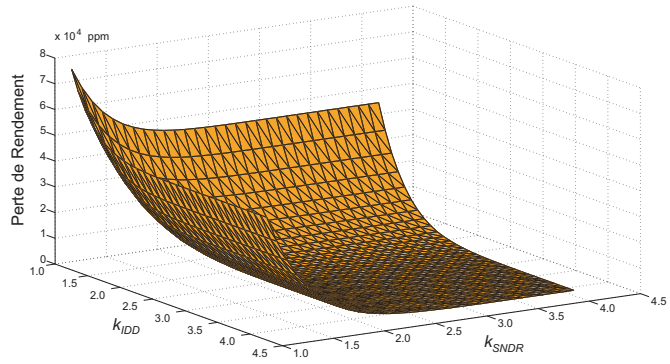


(c)

FIG. 4.12 – Le Rendement moyen, Le Taux de défauts moyen et la Perte de rendement moyen et leurs intervalles de confiance calculés pour des échantillons de tailles différentes et par les deux méthodes : *Méthode 2* et *Méthode 3*.



(a)



(b)

FIG. 4.13 – Le Taux de défauts (a) et la Perte de rendement (b) vs. les limites de test.

4.6 Estimation des métriques de test par élargissement des performances : *Méthode 3*

La technique présentée précédemment (*Méthode 2*) en Section 4.5 pour calculer les métriques de test est basée sur l'estimation des paramètres statistiques à l'aide des simulations Monte Carlo du circuit sous test. La précision de la technique dépend de la précision de ces paramètres estimés. Surtout, si les spécifications du circuit sont très larges, ce qui résulte en une génération très petite, voire nulle, de circuits défaillants quand le nombre de simulations Monte Carlo est faible (par exemple dans notre cas, pour un nombre de 1000 simulations Monte Carlo qui n'est pas considéré comme petit, aucun circuit défaillant n'a été généré).

La précision de ces calculs est donc obtenue en calculant la variance de ces métriques de test pour plusieurs échantillons générés. Il est donc intéressant de recourir aux techniques de réduction de la variance (telles que des techniques de l'échantillonnage d'importance[7]). Afin de réduire l'intervalle de tolérance de chaque métrique de test estimée, pour se rapprocher le plus possible de sa valeur réelle. Nous proposons d'élargir les distributions (supposées Gaussiennes) des performances et des critères de test afin d'augmenter la probabilité de génération de circuits défaillants (voir Figure 4.5). Ceci permettra de générer ce type de circuits pour un nombre petit de simulations Monte Carlo. Ensuite, la formule permettant de calculer les vraies valeurs des métriques de test sera calculée à partir des valeurs obtenues à base des nouvelles Gaussiennes

[7]Importance Sampling.

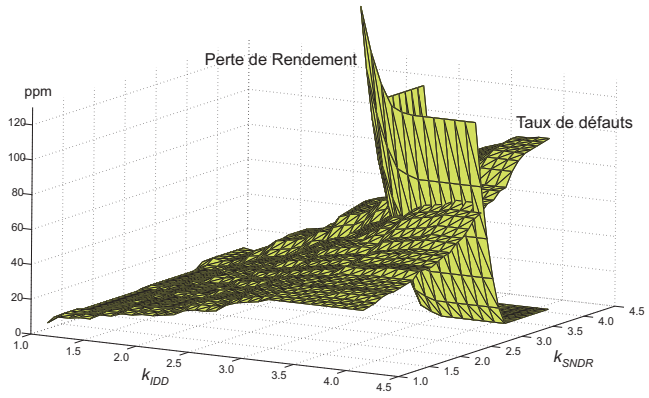


FIG. 4.14 – Le Taux de défauts et la Perte de rendement vs. les limites de test.

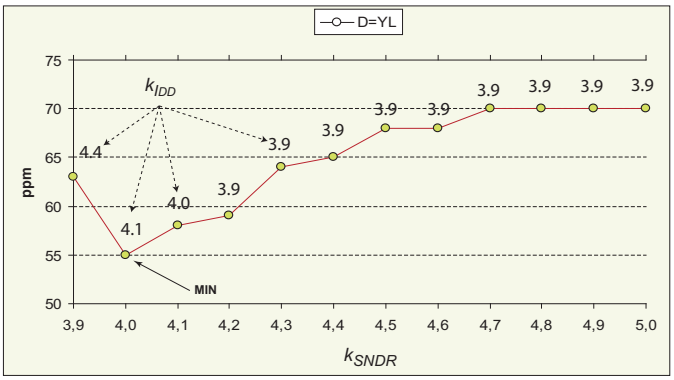


FIG. 4.15 – Points où le Taux de défauts est égal à la Perte de rendement vs. les limites de test.

élargies.

Pour exposer cette méthode d'une manière simple, nous supposerons que nous avons une seule performance x avec une spécification A et que nous voulons calculer la probabilité pour que le circuit soit fonctionnel (le Rendement Y_1). C'est-à-dire, la probabilité pour que les valeurs x de la performance soient dans A. Ceci se calcule en intégrant la densité de probabilité $f(x)$ de cette performance dans le domaine A comme suit :

$$Y_1 = \int_A f(x)\ dx \tag{4.23}$$

Supposons maintenant que l'estimation de Y_1 n'est pas possible pour un échantillon de circuits donné, car dans cet échantillon aucun circuit défaillant n'est généré. Alors, nous allons changer la densité de probabilité de x de $f(x)$ à $g(x)$ de telle sorte à pouvoir générer des circuits défaillants pour cet échantillon. Donc, le Rendement sur cette nouvelle densité se calcule comme suit :

$$Y_2 = \int_A g(x)\ dx \tag{4.24}$$

Mais il est à noter que $Y_1 \neq Y_2$, car, les deux Rendements se calculent pour des densités (ou loi de probabilité) différentes. Notre objectif est donc, de chercher s'il existe une transformation ϕ telle que :

$$Y_1 = \phi(Y_2) \tag{4.25}$$

Dans l'Annexe B nous présenterons une analyse permettant de trouver cette transformation pour le cas général (en considérant toutes les performances et tous les critères de test). Cette

transformation est basée sur la relation entre les deux densités de probabilité, l'initiale et l'élargie, suivante :

$$f_{ST}(s,t) = \frac{f_S(s)}{g_S(s)} \times g_{ST}(s,t) \tag{4.26}$$

où, $s = (s_1, s_2, \cdots, s_n)$ est le vecteur des performances, $t = (t_1, t_2, \cdots, t_m)$ est le vecteur des critères de test, $f_{ST}(s,t)$ est la densité de probabilité conjointe initiale des performances et des critères de test, $g_{ST}(s,t)$ est la densité de probabilité conjointe élargie des performances et des critères de test, $f_S(s)$ est la densité de probabilité conjointe initiale des performances et $g_S(s)$ est la densité de probabilité conjointe élargie des performances.

A partir de cette relation, les métriques de test peuvent être calculées et prédites pour le cas de la loi originale des performances et des critères de test comme suit (voir Annexe B) :

$$Y = \frac{1}{N} \sum_{i=1}^{N_g} \psi_{S_i} \tag{4.27}$$

$$Y_T = \frac{1}{N} \sum_{i=1}^{N_p} \psi_{S_i} \tag{4.28}$$

$$Y_L = 1 - \frac{\sum_{i=1}^{N_u} \psi_{S_i}}{\sum_{i=1}^{N_g} \psi_{S_i}} \tag{4.29}$$

$$D = 1 - \frac{\sum_{i=1}^{N_u} \psi_{S_i}}{\sum_{i=1}^{N_p} \psi_{S_i}} \tag{4.30}$$

où,

$$\psi_{S_i} = \frac{f_S(s^i)}{g_S(s^i)} = \frac{f_S(s_1^i, s_2^i, \cdots, s_n^i)}{g_S(s_1^i, s_2^i, \cdots, s_n^i)} \tag{4.31}$$

et, N le nombre de circuits générés, N_g le nombre de circuits fonctionnels, N_p le nombre de circuits passant le test et N_u le nombre de circuits fonctionnels et qui passent le test.

L'un des avantages de cette technique n'est pas seulement la précision, mais aussi, il n'est pas nécessaire de connaître la densité de probabilité conjointe des performances ainsi que celle des critères de test.

Pour notre cas d'étude, nous avons choisi de prendre comme nouvelle loi, une autre gaussienne avec un écart-type plus grand que celui d'origine. Le nouveau écart-type est augmenté d'un facteur de 1.5. Ensuite, pour une génération d'échantillons de tailles différentes, nous avons calculé les valeurs des métriques de test obtenues par la méthode 2 présentée précédemment et cette méthode (par les équations (4.27), (4.28), (4.29) et (4.30)). La courbe à points de la Figure 4.12(a) montre pour des échantillons de tailles différentes de circuits, les valeurs du Rendement obtenues par les deux méthodes ainsi que leurs intervalles de confiance. Il est clair qu'il y a moins de variations pour le cas de la méthode 3. Ainsi, pour une génération d'un échantillon d'un million de circuits sous test que nous avons répété 1000 fois, le Rendement calculé par la méthode 2 se trouve avec un niveau de confiance de 99.9% dans l'intervalle $[99.9866\%, 99.9927\%]$ de taille $\pm 5ppm$ et celui calculé par la méthode 3 se trouve avec un niveau de confiance de 99.9% dans l'intervalle $[99.9877\%, 99.9915\%]$ de taille $\pm 3ppm$. Les valeurs moyennes ainsi que les intervalles de confiance sont aussi calculées pour les autres métriques de test, telles que le Taux de défauts (la courbe à points de la Figure 4.12(b)) avec une valeur de $78 \pm 5ppm$ (méthode 2) et $78 \pm 3ppm$ (méthode 3) et la Perte de rendement (la courbe à points de la Figure 4.12(c)) qui est le même pour les deux méthodes ($16 \pm 2ppm$). Il est à noter que pour une précision de 10^{-5} la moyenne du Rendement

donnée par la méthode 3 pour des échantillons de tailles différentes est toujours la même est qui est de 99.989%.

Une fois que les différents intervalles de confiance sont calculés, nous avons calculé en utilisant cette approche les métriques de test directement à partir des 1000 circuits générés par la simulation Monte Carlo sous Cadence de l'amplificateur. Nous avons trouvé les valeurs suivantes : le Rendement est de $99,99\%$, le Taux de défauts est de $72ppm$ et la Perte de rendement est de $0ppm$. Comme on peut l'observer sur les Figures 4.12(a), 4.12(b) et 4.12(c), ces valeurs appartiennent bien aux intervalles de tolérances (pour le cas de 1000 générations). Nous voyons bien qu'avec cette méthode, les métriques de test peuvent être prédites directement à partir de la simulation Monte Carlo. Ceci n'est pas possible par estimation directe, car aucun circuit défaillant n'est généralement observé. Ces résultats ne sont pas toujours assurés pour des simulations Monte Carlo à 1000 itérations, d'où la valeur nulle obtenue pour la Perte de rendement. Il faut donc un nombre important de simulations pour générer un ensemble très important de circuits. Cette génération est basés sur la connaissance de la densité de probabilité des performances et des critères de test. Ceci présente l'inconvénient principal des méthodes 2 et 3. Il est donc nécessaire d'utiliser d'autres techniques permettant d'estimer la densité de probabilité telles que les techniques d'estimation non paramétriques dont la méthode du noyau est actuellement la méthode la plus conseillée.

4.7 Conclusion

Dans ce chapitre nous avons présenté une approche statistique pour évaluer les métriques de test durant l'étape design. Cette évaluation permettra ensuite de fixer les limites de test sous des déviations process. L'approche présentée est l'une des contributions principales de cette thèse. Les données utilisées sont obtenues à partir des simulations Monte Carlo du circuit sous test. Pour un ensemble de critères de test donnés, les métriques de test sont évaluées en prenant en considération les spécifications du circuit. Les limites de test sont fixées comme un compromis entre le coût et la qualité de test. Pour ce faire, trois méthodes ont été présentées : La première est basée sur la régression et les deux autres utilisent les distributions de probabilité théoriques pour calculer les différentes métriques de test. Ceci permet d'avoir des résultats précis si la distribution des performances est connue à priori. C'est le cas de la loi multinormale qui est utilisée dans le cadre de ce travail. Par contre, il n'est pas possible de calculer directement les métriques de test quand toutes les performances et les critères de test sont considérés tous à la fois en raison de la difficulté de résoudre les intégrales multiples présentes dans les équations. Nous avons proposé une solution qui remédie à ce problème en générant une grande population d'instances de la loi multinormale, et en estimant directement les métriques de test sur cette population. L'extension de cette technique pour d'autres types de distributions sera considérée en perspectives.

Chapitre 5

Simulation de fautes

5.1 Introduction

Dans ce chapitre, nous allons présenter les outils de simulation, d'injection et de modélisation de fautes analogiques et mixtes composant la plateforme de CAT développée dans le cadre de ce travail, et qui est intégrée dans le logiciel de conception de circuits microélectroniques Cadence.

Ces outils seront ensuite utilisés pour évaluer les métriques de test sous les fautes catastrophiques et paramétriques simples, une fois, les limites de test, des critères de test potentiels, sont fixées par les approches présentées précédemment (Chapitre 4) pour le cas des déviations process.

5.2 Modélisation de fautes analogiques

Les modèles de fautes sont des circuits qui représentent le comportement d'un défaut réel durant une simulation. Les modèles de fautes simples représentatifs des défauts réels sont fondamentaux pour développer une stratégie de test efficace. En effet, la mesure de l'efficacité d'un ensemble de vecteurs de test en terme de la capacité de détecter des défauts réels est basée sur le modèle de fautes utilisé. Si pour un ensemble de tests donné, le taux de Couverture de fautes est de 100%, cela ne signifie pas que tous les défauts physiques seront détectés. L'efficacité d'un ensemble de tests dépend aussi de la représentativité du modèle de fautes utilisé, plus le modèle de fautes est représentatif de la majorité des défauts physiques, plus il y aura de défauts détectés. En général, comme pour les circuits numériques, les modèles de fautes analogiques supposent que si la faute existe alors elle est unique (faute simple). Cependant certaines techniques prennent en compte le cas des fautes multiples, mais elles sont rarement applicables aux circuits actuels car ces derniers sont trop complexes.

Les fautes catastrophiques engendrent un fonctionnement complètement différent du circuit, elles sont alors plus facilement détectables. Aussi, comme la plupart des fautes catastrophiques affectent la plupart des performances du circuit, le problème du choix des meilleurs paramètres à mesurer en sortie (que soit un ensemble des performances ou bien des critères de test) est plus simple. En général, un test DC ou un test en courant IDDQ peut détecter la majorité des fautes catastrophiques [55]. Le test en courant IDDQ est basé sur la mesure du courant d'alimentation du circuit, en général pour les fautes catastrophiques le courant d'alimentation mesuré est différent du courant d'alimentation du circuit correct. Par contre, les fautes paramétriques engendrent des déviations des paramètres des sorties du circuit et ces déviations peuvent être plus au moins grandes suivant le paramètre du circuit considéré, il est donc plus difficile de détecter ces fautes. En effet, il ne suffit pas de trouver les vecteurs de test qui activent la faute, mais il faut aussi trouver les meilleurs paramètres qui permettent d'avoir une déviation en sortie du circuit en dehors de la plage de tolérance acceptable.

Le test des circuits intégrés analogiques et mixtes diffère d'une manière très importante du test des circuits numériques. La différence principale provient du besoin de considérer des signaux continus et les déviations paramétriques, en plus aux fautes catastrophiques (circuit-ouverts et court-circuits), qui sont les seules fautes considérées pour le cas des circuits numériques. Pour ces derniers, le test structurel représente une solution très efficace du point de vue coût de test, où seules les fautes catastrophiques sont prises en compte sans aucune considération du test de la fonctionnalité du circuit. Ainsi, la Couverture de fautes est la métrique de test majeure dans ce domaine, et elle est, en fait, indépendante des spécifications. Pour le cas des circuits analogiques, le besoin de considérer les déviations paramétriques mène à la définition de nouvelles métriques de test qui tiennent compte aussi des fonctionnalités du circuit. D'une autre manière, même si une approche de test est basée sur les fautes paramétriques, une métrique de test telle que la Couverture de fautes ne peut pas être calculée sans connaître les spécifications des performances [116].

5.2.1 Conception des modèles de fautes et langage FMDL

Dans la plateforme de CAT, la modélisation de fautes s'appuie sur un langage de description de modèles de fautes (FMDL[1]). Il s'agit d'un ensemble de fonctions créées en langage SKILL permettant d'injecter des fautes.

Les ingénieurs de test conçoivent un ensemble de modèles de fautes dans l'environnement Cadence DFII[2] en utilisant des interfaces graphiques et textuelles. Ces modèles de fautes sont construits comme des cellules de design, en utilisant des primitives telles que des points de connection, des segments et des instances de sous cellules définies par l'ingénieur de test. Ces primitives ont des étiquettes qui permettent leur instantiation par les procédures d'injection. Des primitives de modélisation de fautes, qui viennent saboter les connections, sont aussi utilisées, permettant la création d'un nouveau point de connection (*intruder*) ou l'interruption ou la configuration d'une connection (*bandit*). Ces primitives facilitent la compréhension du modèle de fautes. Il existe deux versions du FMDL : la première a été décrite par [98] et nécessite un outil de modélisation de fautes supplémentaire qui permet de décrire les différentes connections d'une faute au circuit sous test ; la deuxième version a été décrite dans le cadre de cette thèse où l'injection de fautes est totalement automatisée.

Les Figures 5.1(a) et (c) montrent deux exemples pour illustrer le langage FMDL et la construction et l'injection d'un modèle de fautes. Ces modèles de fautes permettent d'ajouter une résistance à une instance ayant 3 ports. La Figure 5.1(b) montre un exemple du premier modèle injecté comme étant un circuit-ouvert ajouté dans le terminal numéro 1 d'un transistor. La Figure 5.1(d) montre un exemple du deuxième modèle injecté comme étant un court-circuit ajouté entre le terminal numéro 1 et le terminal numéro 2 d'un transistor. Pour construire ces modèles de fautes, nous devons nommer les trois ports qui seront connectés au transistor par `p#1` et les trois ports qui restent doivent être nommés `p#2`. Chaque port doit être connecté à un segment nommé `s#x` où `x` représente le numéro du port du transistor où ce segment doit être connecté. Pour pouvoir connaître le numéro d'un port d'un transistor, nous avons ajouté à l'ensemble des fonctions de Cadence, la fonction `selectPinByNumber()`.

Pour automatiser l'injection de fautes en utilisant ce principe, plusieurs types de fichiers d'injection de fautes ont été créés. Ces types sont présentés en Section 5.3.3.

[1] Fault Model Description Language.
[2] Design Framework II

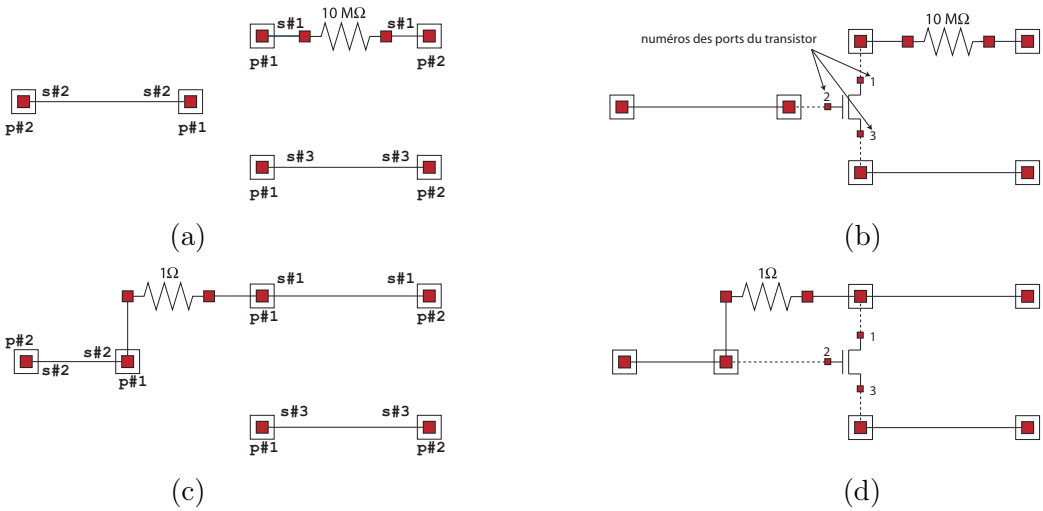


FIG. 5.1 – Construction d'un modèle de fautes : (a) et (c) représentation en format FMDL, (b) et (d) exemples d'injections des modèles de fautes pour le cas d'un transistor.

5.2.2 Modèles de fautes explicites et implicites

La famille des modèles de fautes *explicites* représentent tous les modèles dont leurs formes initiales sont, à priori, connues et qui sont prêts à être injectés directement dans le circuit sous test. Ce type de modèles inclut toutes les fautes catastrophiques ainsi que les fautes paramétriques globales. Cependant, en ce qui concerne les fautes paramétriques simples, les déviations arbitraires données à l'avance avant chaque injection ne représentent pas réellement un modèle significatif d'une faute paramétrique, étant donné qu'il n'est pas possible de savoir si cette déviation permet de violer les spécifications du circuit sous test. Donc, ce type de fautes doit être injecté *implicitement* par recherche de la déviation permettant de violer au moins une spécification du circuit sous test. Si cette déviation existe, alors, elle représente une faute paramétrique qui peut être soit partiellement ou bien totalement détectée. Sinon, elle ne sera pas considérée comme faute du moment qu'aucune déviation sur ce paramètre ne permet de violer les spécifications du circuit sous test.

5.2.3 Modèles de fautes catastrophiques

La définition des fautes catastrophiques diffère d'un auteur à un autre. Pour certain auteurs, les fautes catastrophiques sont des fautes qui correspondent à des défauts de fabrication ponctuels et aléatoires (*Spot Defect*), celles qui résultent d'une particule de poussière sur un masque photolithographique entraînant des déformations locales comme les court-circuits et les circuit-ouverts. En revanche, pour d'autres auteurs les fautes catastrophiques sont des fautes qui engendrent un fonctionnement du circuit complètement différent du fonctionnement normal, même si l'origine de cette faute n'est qu'une variation d'un paramètre du circuit. Dans le cadre de ce travail, nous avons adopté la première définition car elle permet de différencier les types des fautes non pas par rapport au fonctionnement du circuit mais suivant l'origine de la faute. En outre, pour les circuits analogiques, il n'est pas facile de définir la limite entre une petite déviation et une grande déviation pour pouvoir classer sans ambiguïté une faute selon la deuxième définition.

La Figure 5.2 montre quelques modèles de fautes catastrophiques les plus utilisés conçus sous Cadence (Virtuoso [3]) et qui sont prêts à être injectés par l'outil développé. Le modèle de faute de la Figure 5.2(a) permet de remplacer une instance à deux ports par une résistance de valeur 1Ω. Le modèle de faute de la Figure 5.2(b) permet d'injecter un circuit-ouvert à la grille d'un

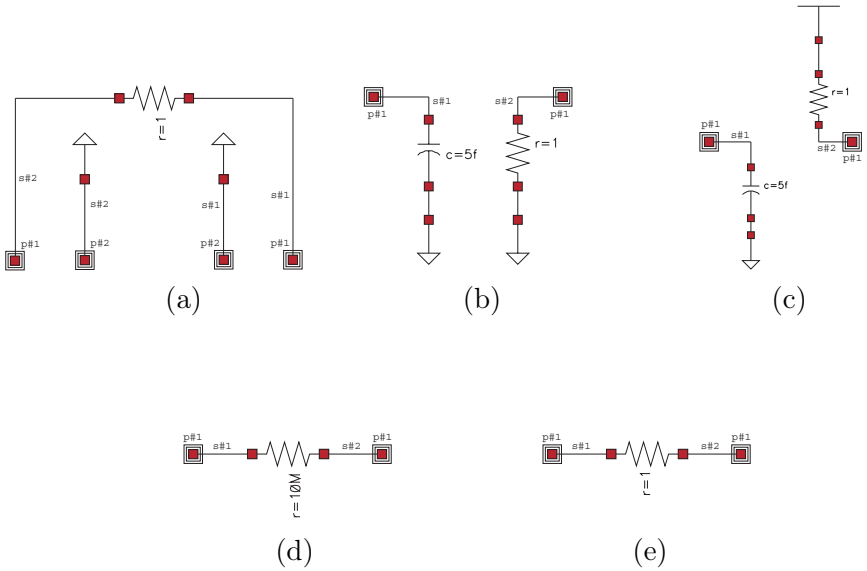


FIG. 5.2 – Modèles de fautes catastrophiques conçus sous Cadence (Virtuoso).

transistor NMOS. Le modèle de faute de la Figure 5.2(c) permet d'injecter un circuit-ouvert à la grille d'un transistor PMOS. Le modèle de faute de la Figure 5.2(d) permet d'injecter un circuit-ouvert entre deux ports d'une instance. Le modèle de faute de la Figure 5.2(e) permet d'injecter un court-circuit entre deux ports d'une instance.

5.2.4 Modèles de fautes paramétriques

Les fautes paramétriques pour certains auteurs sont considérées comme les déviations des paramètres physiques ou géométriques au delà de leur valeurs typiques. En général, ce genre de fautes n'engendrent pas un comportement complètement différent du circuit. Pour d'autres auteurs, les fautes paramétriques sont les déviations des paramètres physiques qui causent la violation d'au moins une des performances du circuit. Ce genre de fautes sont plus difficilement testables.

Modèles de fautes paramétriques locales

Classiquement, les modèles de fautes paramétriques locales représentent une certaine déviation (de ±10%, ±20% ou ±50%) d'un paramètre du circuit par rapport à sa valeur typique. La Figure 5.3 montre trois différents modèles de fautes paramétriques. Le premier modèle représente une résistance déviée de +10% de sa valeur nominale. Le deuxième et le troisième exemple montre des déviations de +10% de la largeur et de la longueur d'un transistor, respectivement. Dans le cas de [116] cette déviation n'est pas fixée à l'avance mais c'est celle qui permet la violation d'au moins une des spécifications du circuit (modèle de fautes implicite).

Modèles de fautes paramétriques globales

Un modèle de fautes paramétrique global représente une déviation sur des paramètres communs à plusieurs composants ou à tous les composants du circuit sous test tel que l'épaisseur d'oxyde t_{ox}. La Figure 5.4 montre un exemple de quelques paramètres globaux d'une technologie ainsi que leurs paramètres statistiques (valeur typique et écart-type). Si par exemple on veut varier explicitement la valeur typique du paramètre `nmoshs_vtor_lot` de 5%, il suffit juste de

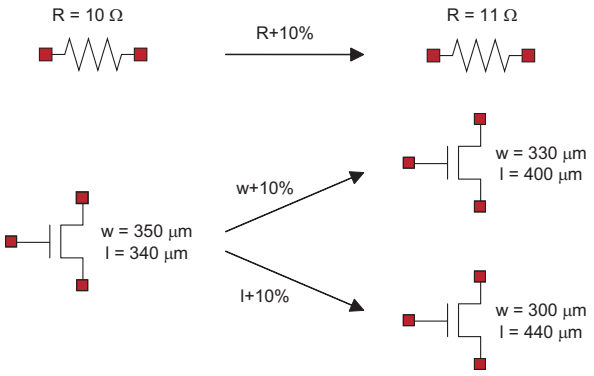


FIG. 5.3 – Modèles de fautes paramétriques locales.

changer la valeur `0.389` par `0.389+5%`. Si, cette variation est implicite, donc ce paramètre sera changé jusqu'à ce qu'il cause la violation d'au moins une des spécifications du circuit sous test.

```
parameters nmoshs_vtor_lot=0.389 parameters nmoshs_dmu_lot=0.0
parameters nmoshs_dracc_lot=0.0 parameters nmoshs_dcjb_lot=0.0
parameters nmoshs_dcjsw_lot=0.0 parameters nmoshs_dcjgate_lot=0.0
statistics {
  process {
    vary nmoshs_vtor_lot dist=gauss std=1.3e-2 percent=no
    vary nmoshs_dmu_lot dist=gauss std=0.0e-2 percent=no
    vary nmoshs_dracc_lot dist=gauss std=0.0e-2 percent=no
    vary nmoshs_dcjb_lot dist=gauss std=3.3e-2 percent=no
    vary nmoshs_dcjsw_lot dist=gauss std=6.6e-2 percent=no
    vary nmoshs_dcjgate_lot dist=gauss std=8.3e-2 percent=no
  }
}
```

FIG. 5.4 – Paramètres géométriques tirés d'un fichier de technologie d'un circuit sous spectre (Cadence).

5.3 Injection de fautes analogiques

L'injection d'une faute représente la procédure permettant d'ajouter au circuit sous test, pendant la phase de simulation, un circuit (appelé faute) dans le but de simuler un défaut réel. Dans les simulateurs de fautes existants, l'injection de fautes se fait en modifiant le fichier *netlist*[3] du circuit sous test. Cependant, l'injection de fautes dans [47, 98] se fait en modifiant directement la forme schématique du circuit sous test sous Cadence avant de générer la netlist. Ce dernier principe d'injection de fautes est utilisé dans le cadre de cette thèse. L'avantage étant que l'injection de fautes est totalement indépendante du simulateur utilisé. Dans ce qui suit, les différentes manières d'injection des fautes catastrophiques et paramétriques seront présentées.

5.3.1 Injection de fautes catastrophiques

Injecter une faute catastrophique est une procédure très simple en la comparant à l'injection d'une faute paramétrique. Une faute catastrophique s'injecte en ajoutant un circuit, appelé modèle de faute, modélisant un certain défaut réel, au circuit sous test, tel qu'un court-circuit, un circuit-ouvert, etc.

[3]Une *netlist* représente la forme textuelle de la forme schématique du circuit sous test utilisée par les simulateurs pour pouvoir effectuer une simulation.

L'injection d'une faute catastrophique ne se limite pas seulement à l'ajout d'un modèle de faute au circuit sous test, mais aussi à la manière dont celui-ci est injecté et à la représentativité du modèle de faute lui même. C'est-à-dire, le modèle de faute utilisé, doit représenter fidèlement un défaut réel.

Donc, l'injection de n'importe quelle faute catastrophique passe par deux étapes principales. La première consiste à trouver le modèle de faute modélisant le plus fidèlement possible le défaut considéré. La deuxième, consiste à trouver la manière dont ce modèle de faute sera injecté dans le circuit sous test. Par exemple, un court-circuit se modélise comme une résistance d'une valeur très petite (de 1Ω à 10Ω), et un circuit-ouvert se modélise comme une résistance d'une valeur très grande (de $10M\Omega$ à $+\infty$). Pour ce qui est de la manière dont ces modèles seront injectés, le court-circuit s'injecte en parallèle entre deux ports d'une instance ou bien entre deux connections d'un circuit. Par contre le circuit-ouvert s'injecte en série à la sortie d'un des ports d'une instance ou bien dans l'un des segments d'une connection d'un circuit.

5.3.2 Injection de fautes paramétriques

Plusieurs propositions ont été faites sur l'injection de fautes paramétriques. [133] propose de faire varier la valeur nominale d'un seul paramètre d'un taux de 3σ sous une simulation Monte Carlo en gardant les variations de tous les autres paramètres (Figure 5.5(a)). [100] propose la même idée mais l'écart type du faux paramètre est fixé à zéro (Figure 5.5(b)). [6] propose de varier la valeur nominale du paramètre à une valeur de 6σ mais sous une simulation aux valeurs typiques (Figure 5.5(c)). Comme c'est déjà expliqué précédemment, [116] définit une faute paramétrique comme étant la valeur d'un paramètre qui permet de violer au moins une des spécifications du circuit. Pour trouver une telle faute, il faut varier le paramètre en question d'un certain pourcentage jusqu'à ce qu'au moins une des spécifications soit violée tandis que les autres paramètres restent fixés à leurs valeurs nominales. En revanche, ce principe ne peut pas être utilisé pour le cas du mauvais fonctionnement d'un circuit résultant de la combinaison de plusieurs petites déviations.

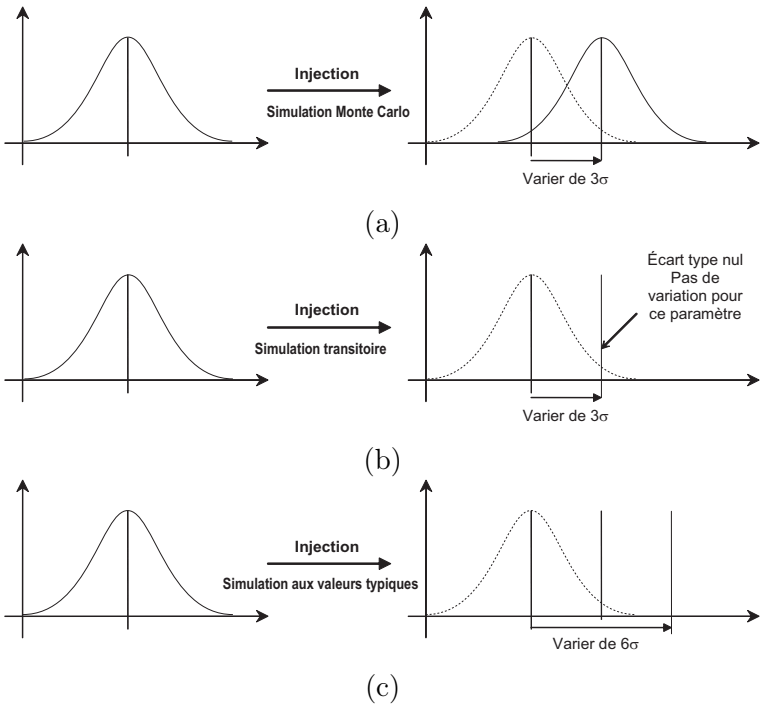


FIG. 5.5 – Différentes injections de fautes paramétriques.

5.3.3 Fichier de Description de l'Injection FID

Nous définissons dans ce qui suit le fichier décrivant l'automatisation de l'injection de fautes, ce fichier est appelé Description de l'Injection de Fautes FID[4]. Le but de ce fichier est de rendre l'injection de faute totalement indépendante de la forme schématique du circuit à simuler. Les fichiers FID sont réutilisables, c'est-à-dire qu'un fichier peut être utilisé pour plusieurs circuits différents. Une première version présentée dans [98] utilise des fichiers *Scénarios* où la faute à injecter doit être décrite précisément avec tous les points de connections, ce qui nécessite un autre outil permettant de récupérer les différentes coordonnées géométriques des différents composants et connections du circuit à simuler. En revanche, un fichier FID est décrit une seule fois et peut être utilisé pour différents circuits et surtout il est créé par une interface facilement manipulable. En cas de nécessité, la seule modification qui peut être produite sur un fichier FID est les noms des instances ou des connections du circuit où la faute doit être injectée (car chaque circuit possède ces propres noms de composants). Et aussi, une telle modification est très simple à effectuer en modifiant directement le fichier en question. Ce fichier permet aussi d'injecter plusieurs fautes de même type, ce qui permet de réduire aussi le nombre de fichiers décrivant l'injection des fautes. Un fichier FID permet d'injecter plusieurs fautes soit de manière séquentielle soit de manière parallèle.

Il existe 8 types de fichiers FID. Cinq types concernent l'injection de fautes catastrophiques, deux concernent les fautes paramétriques locales et un type pour les fautes globales. Les cinq premiers types représentent le remplacement d'une instance par une faute ou la connection d'une faute à une instance donnée, le court-circuit, le circuit-ouvert, l'injection d'une faute entre deux connections et l'insertion d'une faute dans une connection. Les deux autres types représentent la modification d'un paramètre d'une instance directement dans la forme schématique du circuit (exemple, la longueur ou la largeur d'un transistor) de manière explicite ou implicite. Le dernier type qui reste consiste à modifier un paramètre commun à toutes les instances du même type (exemple, t_{ox} pour les transistors).

La Figure 5.6 montre la fenêtre permettant de créer facilement des fichiers de descriptions d'injections FID. Cette fenêtre permet de choisir le circuit du modèle de fautes, le circuit sous test, la liste des instances ou bien des connections ainsi que leurs blocs d'appartenance, s'ils existent. Si la faute sera injectée dans une instance, alors les numéros des ports où cette faute sera injectée peuvent être choisis, sinon, si l'injection de cette faute concerne une connection ou plusieurs connections, alors la liste des connections ainsi que leurs segments concernés peut être choisie aussi. Une fois toutes ces informations sont entrées, il ne reste qu'à donner un nom au fichier FID à créer.

Après avoir créé tous les fichiers FID, ils seront donc prêt à être utilisés par l'outil de simulation de fautes FIDESIM pour effectuer une injection de fautes. Le processus automatique d'injection de fautes à base des fichiers FID est présenté par la Figure 5.7. Ce processus consiste donc à sauvegarder tous les circuits des modèles de fautes dans la base de données Cadence. Ces circuits doivent être créés à base du langage FMDL (voir Section 5.2.1). Ensuite, pour l'injection de fautes de type 1 à 7 (voir les sections suivantes) qui concernent les fautes catastrophiques et paramétriques locales, la fenêtre de la Figure 5.6 permet de les créer comme c'est expliqué ci-dessus, et pour le cas des fautes paramétriques globales, un fichier FID de type 8 (voir ci-dessous) doit être écrit manuellement.

[4]Fault Injection Description.

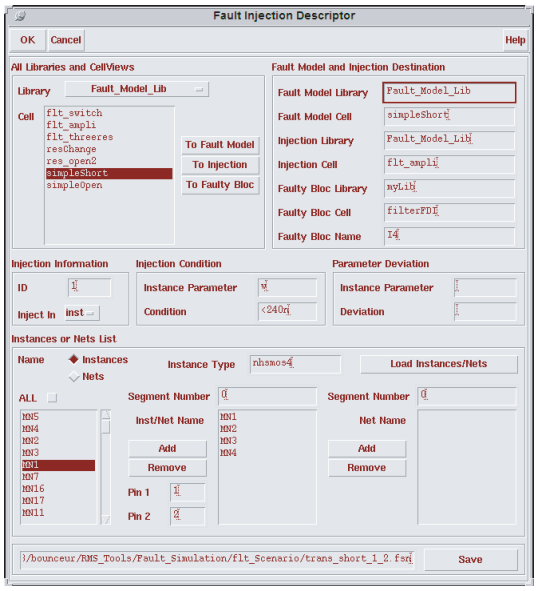


FIG. 5.6 – Fenêtre pour la création d'un fichier FID.

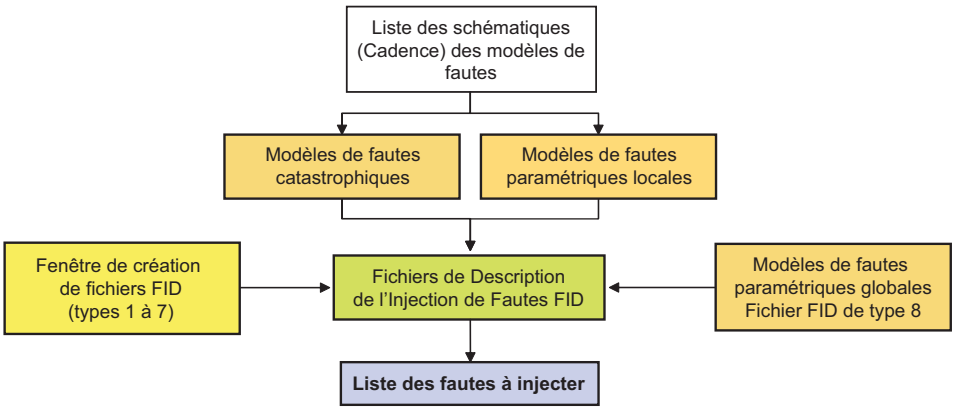


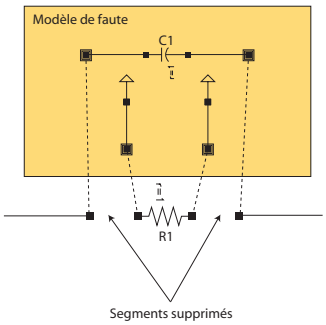
FIG. 5.7 – Mécanisme basé sur les fichiers FID pour la définition de la liste de fautes à injecter.

Fichier FID de type 1

Ce type de fichier d'injection est utilisé pour remplacer un composant du circuit par un modèle de faute. La Figure 5.8(a) montre la forme de ce fichier, où `"flt_mod_Lib"` est le nom de la librairie où la cellule `"flt_mod_Cell"` du modèle de faute est enregistrée. `"injLib"` et `"injCell"` représentent la librairie et la cellule, respectivement, où le modèle de faute doit être injecté. Ces deux informations ne sont pas obligatoires si la faute est injectée directement dans le circuit sous test. Cependant, elles sont nécessaires si la faute doit être injectée dans un bloc `"injBloc"` du circuit sous test. `"nom_inst_i"` représente le nom d'une instance `i` qui sera remplacée par le modèle de faute. Il est à noter que l'injection de fautes sur les instances `i` se fait de manière séquentielle (injection de faute simple). `"inst"` et `"type_inst"` sont ajoutés pour spécifier que le composant à remplacer est une instance de type `"type_inst"` (qui peut être `"res"` pour les résistances, `"cap"` pour les capacités, `"nmos"` pour les transistors NMOS, etc.). Si le modèle de faute sera injecté dans toutes les instances du type `"type_inst"` alors la partie `("nom_inst_1" "nom_inst_2" ...)` peut être remplacée par `"ALL"`. Et en plus le modèle de faute peut être injecté dans toutes les instances de même type `"type_inst"` avec une condition d'injection,

```
(((1 "flt_mod_Lib" "flt_mod_Cell" ["injLib" "injCell" "injBloc"]))
(((1 (1) ("inst" "type_inst"
("nom_inst_1" "nom_inst_2" ...))
(("param" "condition")) ))))
```

(a)



(b)

```
(((1 "Fault_Model_Lib" "capChange" "blcLib" "blcCell" "B1"))
(((1 (1) ("inst" "res"
("R1") )
(nil) ))))
```

(c)

FIG. 5.8 – Format d'un fichier FID de type 1 (a), exemple d'une injection représentant un remplacement d'une résistance par une capacité (b) et son fichier FID correspondant (c).

telle qu'une injection sur toutes les instances ayant un paramètre `"param"` vérifiant la condition `"condition"` (par exemple, les résistances pour lesquelles le paramètre `"r"` est `"<10K"`).

Un exemple d'une injection de ce type est montré par la Figure 5.8(b) qui représente une résistance remplacée par une capacité. Le fichier FID correspondant est montré par la Figure 5.8(c). Ce fichier montre que le modèle de faute qui se trouve dans la cellule `"capChange"` de la libraire `"Fault_Model_Lib"` sera injecté dans la résistance `"R1"` qui se trouve dans le bloc `"B1"` du circuit sous test. Ce bloc fait référence au circuit qui se trouve dans la cellule `"blcCell"` de la librairie `"blcLib"`. Le `nil`, il désigne qu'il n'y ait aucune condition sur l'injection de fautes. Par exemple, si le modèle de fautes est injecté dans toutes les instances de type `"res"` du circuit sous test (ceci est possible en remplaçant dans la Figure 5.8(c) `("R1")` par `"ALL"`) sous la condition que ce modèle doit être injecté uniquement dans les résistances inférieures à $10K\Omega$ alors le `nil` sera remplacé par `("r" "<10K")`.

Fichier FID de type 2

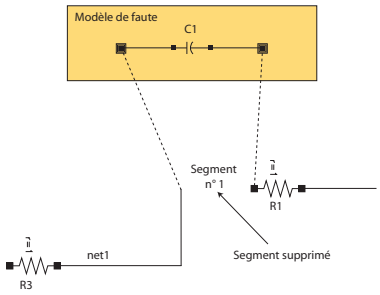
Ce type d'injection permet de remplacer un segment d'une connection par un modèle de fautes. La Figure 5.9(a) montre la forme de ce fichier, où `"flt_mod_Lib"` est le nom de la librairie où la cellule `"flt_mod_Cell"` du modèle de faute est enregistrée. `"injLib"` et `"injCell"` représentent la librairie et la cellule, respectivement, où le modèle de faute doit être injecté. Ces deux informations ne sont pas obligatoires si la faute est injectée directement dans le circuit sous test. Cependant, elles sont nécessaires si la faute doit être injectée dans un bloc `"injBloc"` du circuit sous test. `"nom_net_i"` est le nom d'une connection et `#seg_i` est le numéro de l'un des ses segments où la faute sera injectée. `"net"` représente que la faute sera injectée dans une connection et non pas dans dans une instance. `nil` est un élément qu'il faut ajouter, il est destiné

au programme.

Une injection de ce type est présentée par la Figure 5.9(b) qui montre un exemple d'une capacité insérée dans un segment d'une connection. Le fichier FID correspondant est montré par la Figure 5.9(c) où le segment numéro `1` de la connection `"net1"` est remplacé par le modèle de faute de la cellule `"capModel"` qui se trouve dans la librairie `"Fault_Model_Lib"`.

```
(((1 "flt_mod_Lib" "flt_mod_Cell" ["injLib" "injCell" "injBloc"]))
(((1 (2) ("net" nil
(("nom_net_1" #seg_1) ("nom_net_2" #seg_2) ...) )))))
```

(a)



(b)

```
(((1 "Fault_Model_Lib" "capModel"))
(((1 (2) ("net" nil
(("net1" 1)) )))))
```

(c)

FIG. 5.9 – Format d'un fichier FID de type 2 (a), exemple d'une injection représentant une capacité insérée dans le premier segment d'une connection (b) et son fichier FID correspondant (c).

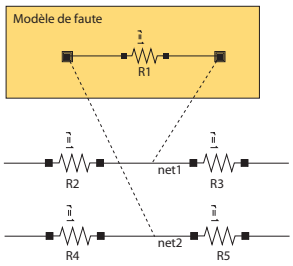
Fichier FID de type 3

Ce type d'injection permet d'insérer une faute entre deux connections d'un circuit. La Figure 5.10(a) montre la forme de ce fichier, où `"flt_mod_Lib"` est le nom de la librairie où la cellule `"flt_mod_Cell"` du modèle de faute est enregistrée. `"injLib"` et `"injCell"` représentent la librairie et la cellule, respectivement, où le modèle de faute doit être injecté. Ces deux informations ne sont pas obligatoires si la faute est injectée directement dans le circuit sous test. Cependant, elles sont nécessaires si la faute doit être injectée dans un bloc `"injBloc"` du circuit sous test. Comme ce type d'injection permet d'injecter un modèle de fautes entre deux connections, alors, `#seg1_i` représente le nom du segment de la première connection `"nom_net1_i"` et `#seg2_i` représente le nom du segment de la deuxième connection `"nom_net2_i"` entre lesquels la faute sera injectée. `"net"` représente que la faute sera injectée dans une connection et non pas dans dans une instance. `nil` est un élément qu'il faut ajouter, il est destiné au programme.

La Figure 5.10(b) montre un exemple d'une injection de type 3. Elle représente un court-circuit entre deux connections. Le fichier FID correspondant est montré par la Figure 5.10(c) où le modèle de faute (résistance d'une valeur très petite) de la cellule `"simpleShort"` de la librairie `"Fault_Model_Lib"` est injectée entre le segment numéro `1` de la connection `"net1"` et le segment numéro `1` de la connection `"net2"`.

```
(((1 "flt_mod_Lib" "flt_mod_Cell" ["injLib" "injCell" "injBloc"]))
(((1 (3) ("net" nil
((("nom_net1_1" #seg1_1) ("nom_net2_1" #seg2_1))
(("nom_net1_2" #seg1_2) ("nom_net2_2" #seg2_2))
...
(("nom_net1_n" #seg1_n) ("nom_net2_n" #seg2_n))) )))))
```

(a)



(b)

```
(((1 "Fault_Model_Lib" "simpleShort"))
(((1 (3) ("net" nil
((("net1" 1) ("net2" 1)))
)))))
```

(c)

FIG. 5.10 – Format d'un fichier FID de type 3 (a), exemple d'une injection représentant une résistance insérée entre deux connections (b) et son fichier FID correspondant (c).

Fichier FID de type 4

Ce type d'injection permet de connecter un modèle de faute entre deux ports d'une instance. La Figure 5.11(a) montre la forme de ce fichier qui ressemble quasiment au fichier FID de type 1. La seule différence est que dans le type 4 il figure deux numéros en plus `#Pin1` et `#Pin2` qui représentent les numéros des ports d'une instance `"nom_inst_i"` entre lesquels la faute sera injectée. Il est généralement utilisé pour injecter des court-circuits.

Les Figure 5.11 (b1), (b2) et (b3) montrent des exemples de court-circuits injectés dans des transistors NMOS (instance de type *"nmos"*). Le fichier FID de la Figure 5.11(c) correspond au cas de la Figure 5.11(b1), où le modèle de faute de la cellule `"simpleShort"` de la librairie `"Fault_Model_Lib"` est injecté entre les ports `1` (Grille) et le port `2` (Drain) du transistor NMOS de nom `"M1"`. Le `nil` signifie qu'il n'y a pas de condition d'injection. Afin de pouvoir connaître le numéro associé à chaque port d'une instance, nous avons développé la fonction `selectPinByNumber()` qui permet de sélectionner un port d'une instance en donnant son numéro.

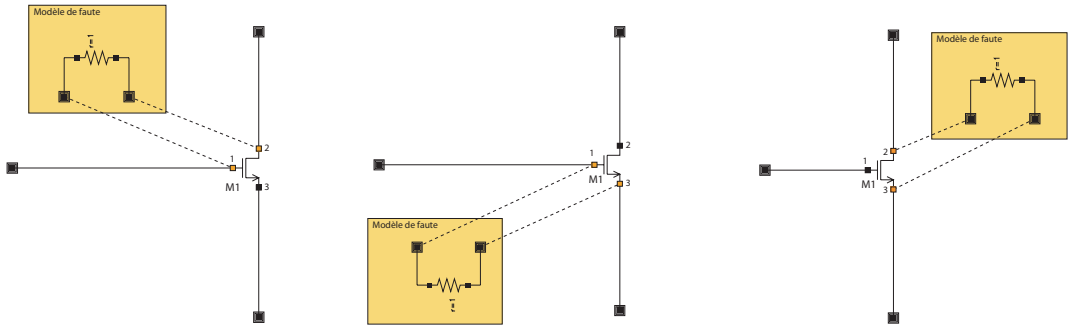
Fichier FID de type 5

Ce type d'injection permet de connecter une faute à un port d'une instance. La Figure 5.12(a) montre la forme de ce fichier qui est presque le même que le fichier FID de type 4, présenté ci-dessus. La seule différence est que dans ce type 4 il figure uniquement un seul numéros `#Pin` qui représente le numéro du port d'une instance `"nom_inst_i"` auquel la faute sera connectée. Il est généralement utilisé pour injecter des circuit-ouverts.

Les Figures 5.12 (b1) et (b2) montrent des exemples de circuit-ouverts injectés dans un

```
(((1 "flt_mod_Lib" "flt_mod_Cell" ["injLib" "injCell" "injBloc"]))
(((1 (4)("inst" "type_inst"
("nom_inst_1" "nom_inst_2" ...))
(("param" "condition") #Pin1 #Pin2) ))))
```

(a)



(b1) (b2) (b3)

(b1) Court-circuit Grille-Drain
(b2) Court-circuit Grille-Source
(b3) Court-circuit Drain-Source

```
(((1 "Fault_Model_Lib" "simpleShort"))
(((1 (4) ("inst" "nmos"
("M1"))
(nil 1 2) ))))
```

(c)

FIG. 5.11 – Format d'un fichier FID de type 4 (a), exemples d'injections représentant les différents court-circuits d'un transistor NMOS (b1, b2 et b3) et le fichier FID correspondant au premier cas, un court-circuit Grille-Drain (c).

transistor NMOS (instance de type *"nmos"*). Le fichier FID de la Figure 5.12(c) correspond au cas de la Figure 5.12(b1), où le modèle de faute de la cellule `"simpleOpen"` de la librairie `"Fault_Model_Lib"` est connecté au port `2` (Drain) du transistor NMOS de nom `"M1"`. Le `nil` signifie qu'il n'y a pas de condition d'injection.

Fichier FID de type 6

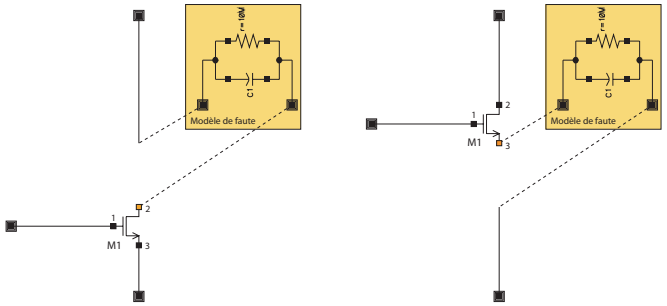
Ce type d'injection permet de modifier un paramètre d'une instance d'un certain pourcentage. La Figure 5.13(a) montre la forme de ce fichier où le paramètre `"injParam"` est dévié par un pourcentage `pourcentage`. Ce dernier peut être négatif. Les autres informations de ce fichier sont les mêmes que celles des fichiers FID précédemment définis. La Figure 5.13(a) montre un exemple permettant de dévier de 50% le paramètre `"w"` (largeur) d'un transistor `"nmos"` de nom `"M1"`. Pas de conditions d'injection dans ce fichier, d'où l'existence du `nil`.

Fichier FID de type 7

Ce type d'injection permet de trouver la déviation minimale d'un paramètre permettant de violer au moins une des spécifications du circuit sous test. Cette déviation est obtenue à base d'une recherche dichotomique (en considérant tous les bancs de test). La Figure 5.14(a) montre la forme du fichier FID de type 7, où, `"injParam"` représente le paramètre pour lequel la déviation minimale sera calculée, `typiqueVal` est sa valeur typique et `pourcentage` représente

```
(((1 "flt_mod_Lib" "flt_mod_Cell" ["injLib" "injCell" "injBloc"]))
(((1 (5)("inst" "type_inst"
("nom_inst1" "nom_inst2" ...))
(("param" "condition") #Pin)))))
```

(a)



(b1) (b2)

(b1) Circuit-ouvert au Drain d'un NMOS
(b2) Circuit-ouvert à la Source d'un NMOS

```
(((1 "Fault_Model_Lib" "simpleOpen"))
(((1 (5) ("inst" "nmos"
("M1"))
(nil 2)))))
```

(c)

FIG. 5.12 – Format d'un fichier FID de type 5 (a), exemples d'injections représentant quelques circuit-ouverts d'un transistor NMOS (b1 et b2) et le fichier FID correspondant au premier cas, circuit-ouvert au Drain (c).

sa déviation maximale. C'est-à-dire que la déviation minimale violant les spécifications ne doit pas dépasser cette déviation. Pour le détail sur les autres éléments de ce fichier, voir les fichiers FID précédemment décrits.

L'exemple montré par la Figure 5.14(b) consiste à calculer la déviation minimale du paramètre `"w"` du transistor NMOS de nom `"M1"` qui doit être entre $7.5\mu m$ et $7.5\mu m + 200\%$. Bien entendu, cette déviation est calculée en considérant un programme de configuration de test qui définit les spécifications à vérifier et les analyses à effectuer (voir Section 5.4.1).

Fichier FID de type 8

Ce type d'injection permet de modifier des paramètres communs à toutes les instances du circuit sous test (paramètres globaux). Il contient plusieurs blocs qui représentent les différentes injections à effectuer séquentiellement. Puisque les paramètres d'une technologie peuvent être décrits dans plusieurs fichiers[5], chaque bloc de ce fichier contient des sous-blocs dont chacun concerne le fichier de technologie contenant les paramètres à modifier. La Figure 5.15(a) montre la forme d'un fichier FID de type 8 qui est décrit comme suit. Le fichier `"fichier_i_1"`, où i=1,...,M représente le fichier d'origine d'un modèle d'un composant `i` (par exemple, un NMOS) destiné pour la simulation, mais dans notre cas, ce fichier sera utilisé comme référence et il ne sera pas modifié directement. Par contre, c'est sa copie `"fichier_i_2"` créée automatiquement qui sera modifiée et qui sera utilisée pour la simulation. Ensuite l'ensemble des déviations est décrit

[5]Les fichiers de technologie qui portent par exemple l'extension .scs pour le cas du simulateur spectre.

```
(((2 (("nom_inst_1" "nom_inst_2" ...) "type_inst")
"injParam" pourcentage
("param" "condition") ["injLib" "injCell" "injBloc"])))
```

(a)

```
(((2 (("M1") "nmos")
"w" 50 (nil) )))
```

(b)

FIG. 5.13 – Format d'un fichier FID de type 6 (a) avec exemple (b).

```
(
("nom_inst_1" "nom_inst_2" ...) "type_inst" "injParam"
"injLib" "injCell" "injBloc"
typiqueVal pourcentage
)
```

(a)

```
(
("M1") "nmos" "w"
"" "" ""
7.5e-6 200
)
```

(b)

FIG. 5.14 – Format d'un fichier FID de type 7 (a) avec exemple (b).

comme suit. Pour un modèle `i` donné (d'un fichier `"fichier_i"`), nous décrivons l'ensemble des déviations en donnant l'ensemble des noms des paramètres `"paramètre_i_j"` et leurs déviations `"déviation_i_j"` ainsi que leurs types de déviation `type_i_j`. Ce dernier prend la valeur `nom` pour la déviation de sa valeur nominale (ou de sa moyenne) ou bien `tol` pour la déviation de son sigma (ou de son écart-type).

Un exemple d'un fichier FID de type 8 est montré par la Figure 5.15(b). Ce FID permet d'injecter deux fautes l'une après l'autre. La première faute représente une déviation de 60% des valeurs nominales de toutes les longueurs des transistors PMOS du circuit sous test (qui sont définies dans le fichier de technologie `pmoshs.scs`). La deuxième faute représente deux déviations qui seront injectées en même temps (en parallèle), la première consiste à élargir la tolérance d'un facteur de 5 de tous les transistors NMOS (qui sont définies dans le fichier de technologie `nmoshs.scs`), et la deuxième consiste à varier la valeur nominale de -50% de toutes les largeurs des transistors NMOS du circuit sous test. Il est à noter que dans la version actuelle de la plateforme, la recherche des déviations minimales violant au moins une des spécifications du circuit sous test n'est pas encore considérée pour les fautes paramétriques globales.

```
(# Bloc 1 : Début Injection 1
   ("fichier_1_1" " fichier_1_2"
   ("paramètre_1_1" "déviation_1_1" type_1_1)
   ("paramètre_1_2" "déviation_1_2" type_1_2)
   ...
   ("paramètre_1_N1" "déviation_1_N1" type_1_N1))
   ...
   ("fichier_2_1" " fichier_2_2"
   ("paramètre_2_1" "déviation_2_1" type_2_1)
   ("paramètre_2_2" "déviation_2_2" type_2_2)
   ...
   ("paramètre_2_N2" "déviation_2_N2" type_2_N2))
   ...
   (("fichier_M_1" " fichier_M_2"
   ("paramètre_M_1" "déviation_M_1" type_M_1)
   ("paramètre_M_2" "déviation_M_2" type_M_2)
   ...
   ("paramètre_2_N3" "déviation_2_N3" type_M_N3)) )# Bloc 1 : Fin Injection 1
(# Bloc 2 : Début Injection 2
...
)# Bloc 2 : Fin Injection 2
...
(# Bloc Dernier : Début Dernière Injection
...
)# Bloc Dernier : Fin Dernière Injection
```

(a)

```
RMS Tools : Parametric Faults Injection
(
   ("pmoshs.scs" "pmoshs2.scs"
   ("l" "60" nom))
)
(
   ("nmoshs.scs" "nmoshs2.scs"
   ("l" "5" tol)
   ("w" "-50" nom))
)
```

(b)

FIG. 5.15 – Format d'un fichier FID de type 8 (a) avec exemple (b).

5.4 Évaluation des métriques de test sous la présence de fautes simples

5.4.1 Configuration des bancs de test

Un *banc de test*[6] représente une configuration du circuit sous test par laquelle un certain ensemble de performances ou de critères de test peut être calculé. Pour calculer, pour un certain banc de test, les différentes performances ainsi que les différents critères de test correspondantes, en utilisant l'outil FIDESIM, il faut entrer à l'aide de la fenêtre de la Figure 5.16 un *Programme de Configuration de Test*[7] sous forme d'un pseudo code. Dans celui-ci, l'ensemble des performances avec leurs spécifications ainsi que l'ensemble des critères de test et leurs limites sont décrits.

Cependant, pour calculer toutes les performances ainsi que tous les critères de test d'un circuit, plusieurs bancs de test sont nécessaires. Ceci nécessite donc des simulations pour chaque banc de test. Ce processus a été automatisé en utilisant un programme de configuration de plusieurs bancs de test. Un exemple de ce programme permettant de lancer une simulation de

[6]Test bench.
[7]Test Configuration Program.

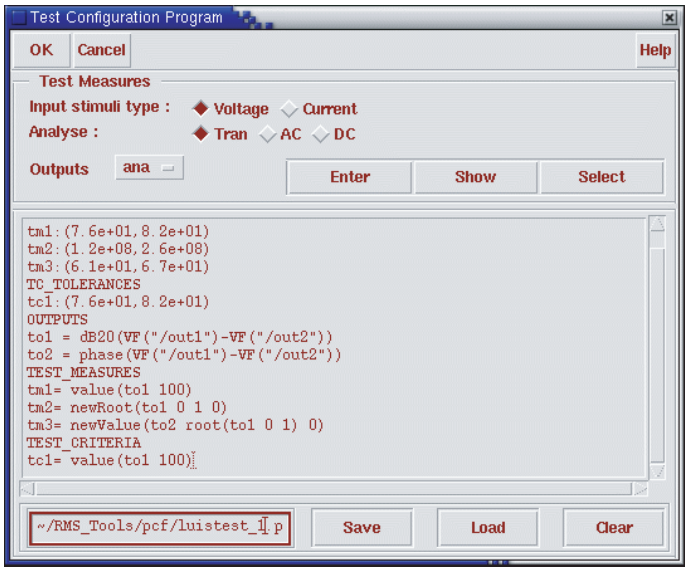


FIG. 5.16 – Exemple d'un Programme de configuration de Test.

fautes en considérant deux bancs de test (le premier et le deuxième de l'amplificateur) est présenté par la Figure 5.17. Ce programme contient l'ensemble des informations nécessaires pour lancer une simulation de fautes, telles que le circuit sous test, le programme de configuration de test, la liste des fautes à injecter, l'analyse de simulation, le type de simulation de fautes et l'ensemble des résultats à sauvegarder. Il est à noter que pour le cas de la simulation Monte Carlo, une instance générée doit être la même pour chaque banc de test.

Le processus permettant de gérer la simulation de fautes à plusieurs bancs de test est illustré par la Figure 5.18. L'ingénieur de test doit commencer par configurer manuellement chaque banc de test séparément afin d'obtenir sur un fichier toutes les informations nécessaires pour effectuer une simulation de fautes. En assemblant tous ces fichiers dans un seul, une simulation de fautes avec la considération de tous les bancs de test peut être faite.

Quelques bancs de test (le 1^{er}, le $4^{ème}$, le $6^{ème}$ et le $7^{ème}$) de l'amplificateur de la Section 4.3 sont montrés par les Figures 5.19, (a), (b), (c) et (d), respectivement. Le tableau 4.1 montre les différentes performances ainsi que leurs bancs de test correspondants. Huit bancs de test sont nécessaires pour calculer toutes les performances. Pour le test réel de l'amplificateur, la mesure du SNDR est considérée en utilisant la technique du *sine-wave fitting* décrite dans [4]. Un banc de test supplémentaire est nécessaire pour simuler cette mesure. Le temps de simulation, pour effectuer la simulation Monte Carlo de 1000 instances de l'amplificateur en considérant tous les 9 bancs de test est de 3 heures.

5.4.2 Création des modèles de fautes

Les modèles de fautes sont créés sous Cadence. Ils doivent vérifier certaines règles décrites par le langage FMDL (voir Section 5.2.1). Ces règles sont utiles pour automatiser l'injection de fautes. Les fautes à injecter dans notre cas d'étude sont les suivantes :

- les court-circuits dans les résistances,
- les court-circuits dans les capacités,
- les court-circuits entre le drain et la source des NMOS et des PMOS,
- les court-circuits entre le drain et la grille des NMOS et des PMOS,
- les court-circuits entre la grille et la source des NMOS et des PMOS,
- les circuit-ouverts sur les résistances,

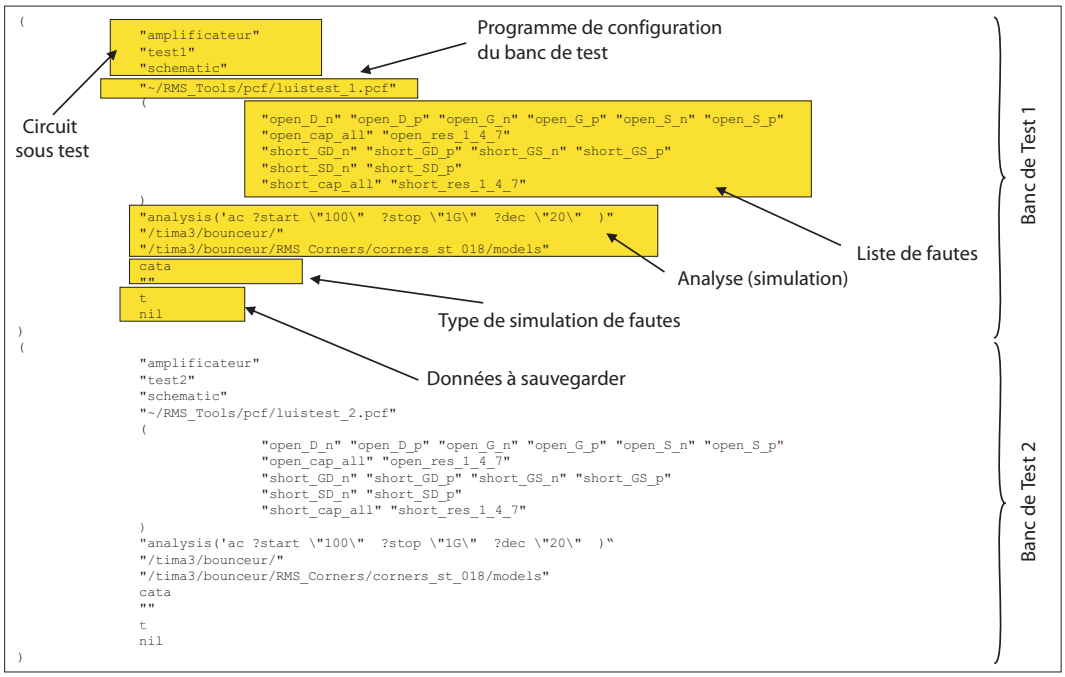


FIG. 5.17 – Exemple d'un Programme de Configuration à Plusieurs Bancs de Test pour l'amplificateur (banc de test 1 et 2).

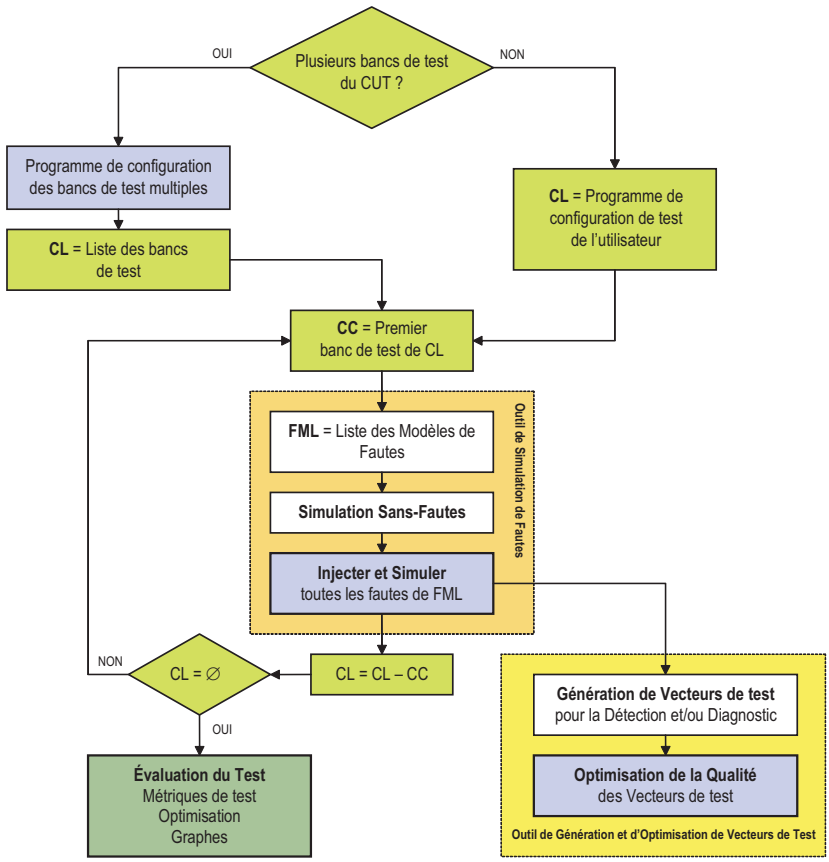


FIG. 5.18 – Procédure de simulation de fautes à plusieurs bancs de test.

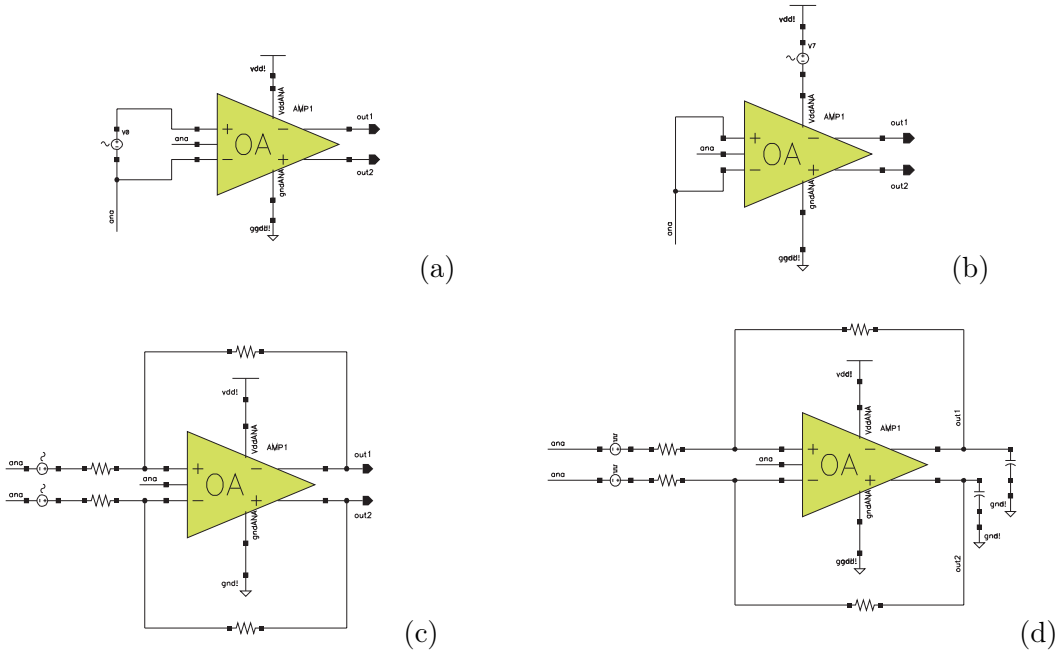


FIG. 5.19 – Exemple des configurations des bancs de test (a) : n°1, (b) : n°4, (c) : n°6 et (d) : n°7 de l'amplificateur.

- les circuit-ouverts sur les capacités,
- les circuit-ouverts au drain des NMOS et des PMOS,
- les circuit-ouverts à la grille des NMOS et des PMOS,
- les circuit-ouverts à la source des NMOS et des PMOS.

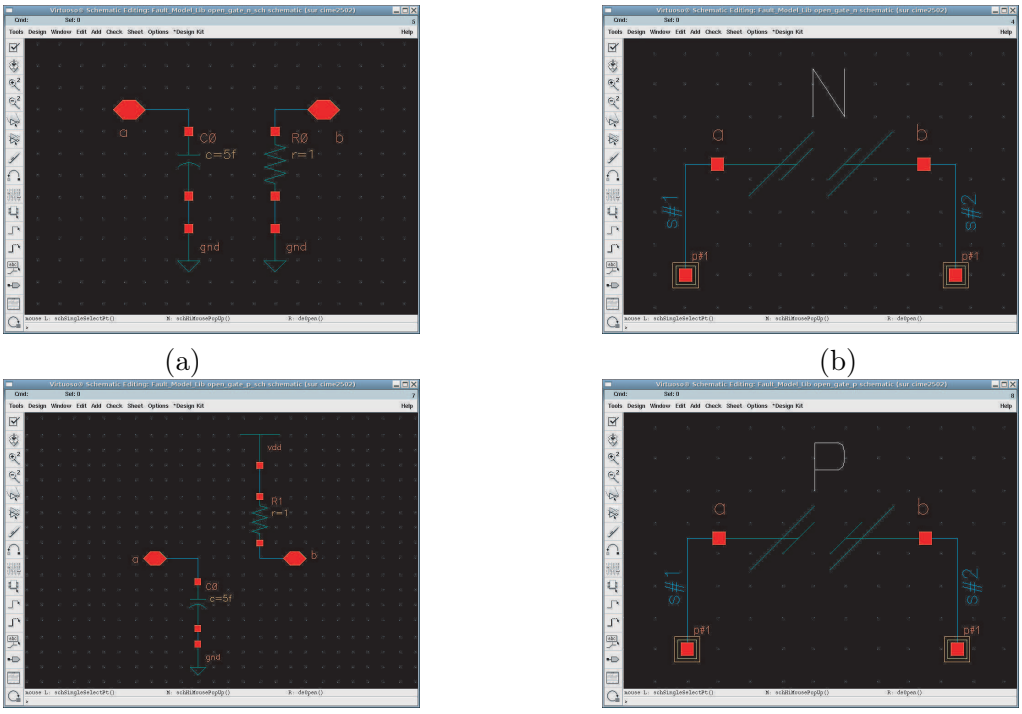
En éliminant les fautes redondantes, ceci résulte en un ensemble de 160 injections de fautes catastrophiques. Cependant, uniquement 4 modèles de fautes sont nécessaires pour injecter toutes ces fautes. La Figure 5.20 montre un exemple de deux modèles de fautes, le circuit-ouvert à la grille d'un NMOS (a) et le circuit-ouvert à la grille d'un PMOS (c), et leurs format FMDL présentés par (b) et (d), respectivement, créés sous Cadence.

5.4.3 Création des fichiers FIDs

Une fois tous les modèles de fautes sont créés sous le format FMDL et sauvegardés dans les librairies Cadence, un ensemble de fichiers permettant de décrire comment injecter ces modèles dans le circuit sous test doit être défini. Ces fichiers de type FID (voir Section 5.3.3) seront donc une des entrées principales de l'outil de simulation de fautes FIDESIM. L'interface de cet outil avec la liste des fichiers FID créés (ou choisis, car les fichiers FID sont réutilisables) est présentée par la Figure 5.21. Cette liste se trouve au milieu de cette fenêtre. La partie droite de cette fenêtre montre la liste des fautes injectées décrite par tous les fichiers FID choisis.

5.4.4 Exécution de la simulation de fautes

La procédure de simulation de fautes utilisée par l'outil FIDESIM est présentée par la Figure 5.22. D'abord, l'ingénieur de test doit décrire un programme de configuration de test (voir la Section 5.4.1). Ce programme contient toutes les informations nécessaires pour l'évaluation de test et les informations à sauvegarder dans la base de données. Ensuite, l'ensemble des fichiers FIDs seront utilisés pour définir la liste des fautes à injecter et la manière dont ces fautes seront injectées. La simulation de fautes peut donc être effectuée en simulant d'abord le circuit sans



(a) (b)

(c) (d)

FIG. 5.20 – Exemple de modèles de fautes : (a) Circuit-ouvert à la grille d'un NMOS (forme Schématique), (b) Circuit-ouvert à la grille d'un NMOS (forme FMDL), (c) Circuit-ouvert à la grille d'un PMOS (forme Schématique) et (d) Circuit-ouvert à la grille d'un PMOS (forme FMDL).

fautes puis en simulant ce circuit avec chaque faute. Les fautes sont injectées soit au niveau schématique, soit dans le fichier technologie. L'injection au niveau comportemental ou layout n'est pas encore prise en compte par la plateforme développée.

5.4.5 Évaluation des métriques de test sous des fautes catastrophiques

Pour notre cas d'étude, nous avons considéré les fautes catastrophiques resultant des court-circuits et des circuit-ouverts dans tous les transistors, toutes les résistances et toutes les capacités de l'amplificateur. Ce qui a donné 160 fautes catastrophiques, en éliminant les fautes redondantes. La Figure 5.23 montre la Couverture de fautes donnée par chaque performance et chaque critère de test avec les limites de test fixées comme c'est décrit dans le chapitre précédent. La considération de toutes les performances à la fois permet de détecter 98.12% de fautes où les fautes non détectées se situent dans le bloc d'alimentation. D'autre part, le critère de test $SNDR$ permet la détection de 89.38% de fautes. Les fautes non détectées se situent aussi dans le bloc d'alimentation. Une Couverture de faute maximale peut être atteinte si le courant de consommation I_{DD} est considéré avec le $SNDR$.

Nous présenterons dans le chapitre suivant un algorithme pour générer des tests orientés à la détection des fautes. Cet algorithme sera appliqué au cas des fautes qui ne sont pas détectées ici avec la mesure des performances ou des critères de test considérés.

5.4.6 Évaluation des métriques de test sous des fautes paramétriques

Nous considérons comme fautes paramétriques les déviations qui causent la violation d'au moins une des spécifications du circuit sous test. Les paramètres physiques considérés sont la

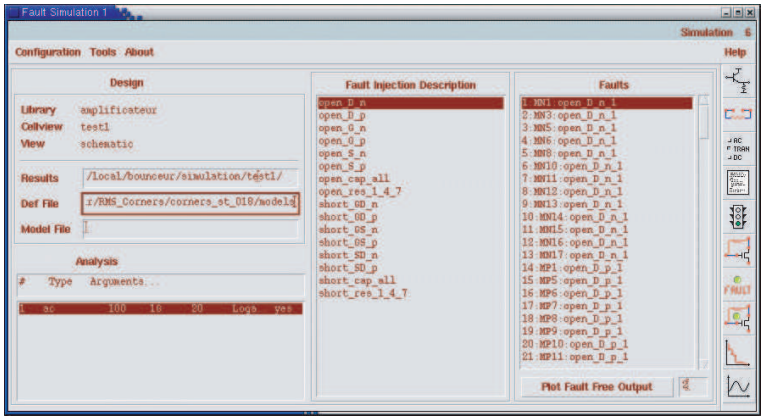


FIG. 5.21 – Interface du simulateur de fautes FIDESIM.

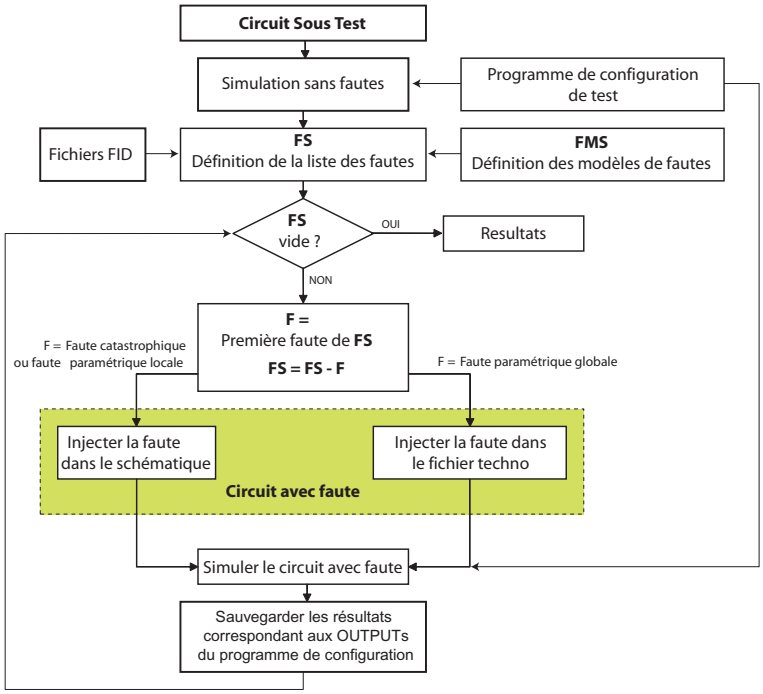


FIG. 5.22 – Procédure de simulation de fautes de l'outil FIDESIM.

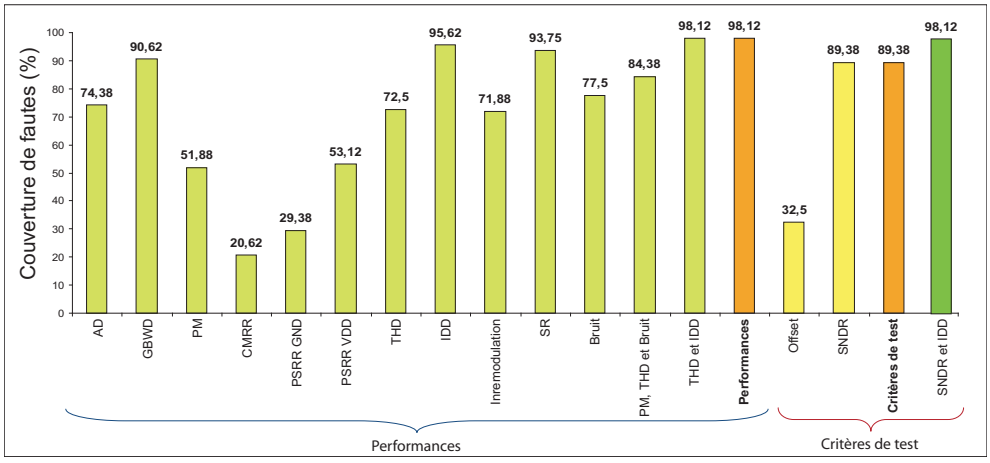


FIG. 5.23 – La Couverture de fautes catastrophiques des performances et des critères de test.

longueur (L) et la largeur (W) des transistors NMOS et PMOS, les résistances (R) et les capacités (C). Notons que d'une part, L et W des transistors ne sont pas des paramètres process, et donc leurs déviations ne sont pas incluses dans les déviations process. D'autre part, les résistances et les capacités dévient sous les variations process.

Afin de calculer la probabilité d'occurrence d'une faute p_i^{spec} et la probabilité de détection d'une faute p_i^{test} pour chaque paramètre physique potentiel, avec faute, nous avons obtenu par simulation les valeurs limites de ces déviations (en utilisant les fichiers d'injection de type 7). La distribution de chaque paramètre physique est considérée comme multinormal avec une moyenne μ égale à sa valeur typique. L'écart-type σ est fixé à $10nm$ pour L et W des transistors, et il est fixé à 5% pour les résistances et les capacités. Ainsi, pour chaque paramètre physique variant, les fichiers FID considèrent des déviations minimales et maximales de -20% et $+200\%$, respectivement.

Ce qui résulte à la considération de 180 paramètres physiques à dévier, où uniquement 13 ont permis de violer les spécifications. Ces fautes sont listées dans le Tableau 5.1 ainsi que leurs probabilités d'occurrence et de détection. Les autres fautes ont des probabilités d'occurrence négligeables ($p_i^{spec} \simeq 0$). Il est à noter que pour les transistors qui sont couplés, les déviations ne peuvent pas être prises en considération individuellement. Elles sont considérées en injectant la même déviation dans tous les transistors couplés à la fois. Nous avons observé que toutes les fautes dans les transistors couplés ont des probabilités négligeables. Afin de considérer le comportement avec fautes résultant des couplages, il est nécessaire d'effectuer une simulation Monte Carlo avec option couplage[8]. Dans le cadre de ce travail, nous avons considéré uniquement les simulation Monte Carlo avec option déviation process. Mais la même étude peut être faite en considérant les déviations couplées.

No	**Composant (Paramètre)**	**Faute**	p^{spec}	$\min(p^{spec}, p^{test})$
1	MP1 (l)	+12.21%	0.013996	0.013996
2	MP3 (l)	+7.32%	0.093690	0.093690
3	MP5 (l)	+22.95%	0.000009	0.000009
4	MP1 (l)	−3.18%	0.283305	0.283305
5	MP3 (l)	−11.70%	0.017604	0
6	MP18 (l)	−15.76%	0.002270	0
7	MN2 (l)	−16.23%	0.001736	0.001736
8	MN4 (l)	−6.46%	0.122278	0.122278
9	R1 (r)	+15.14%	0.001227	0.001227
10	R4 (r)	+17.09%	0.000308	0.000308
11	R7 (r)	+16.11%	0.000628	0.000628
12	R4 (r)	−12.79%	0.005249	0.005249
13	R7 (r)	−16.15%	0.000611	0.000611

TAB. 5.1 – Fautes paramétriques simples pour calculer les métriques de test, ayant des probabilités d'occurrence non négligeables.

Dans ce qui suit nous associerons aux métriques de test la lettre D (que nous mettons comme exposant) pour désigner que ces métriques sont estimées durant la phase design (à base des équations (2.47) à (2.50) et (2.1)) et la lettre F pour désigner que celles-ci sont estimées sous la présence de fautes paramétriques simples (à base des équations (2.6) à (2.10) et (2.46)).

Les valeurs moyennes des métriques de test calculées pour le cas des déviations process sont présentées dans la Section 4.6 avec leurs intervalles de confiance. Ces valeurs sont résumées dans le Tableau 5.2, auquel nous avons ajouté la valeur du Rendement de test Y_T^D qui n'a pas été

[8]Mismatch simulation.

donnée auparavant.

Métrique	Valeur moyenne
Y^D	99.989%
Y_T^D	99.99%
Y_C^D	99.99%
D^D	0.0078%

TAB. 5.2 – Les valeurs moyennes des métriques de test sous des déviations process.

Nous avons utilisé l'Equation (2.46) et les équations (2.6) à (2.9) pour calculer les métriques de test pour le cas des fautes paramétriques. Les valeurs ainsi obtenues sont données par le Tableau 5.3. Nous avons considéré deux cas : dans le premier cas (colonne 2), toutes les 13 fautes sont considérées. Uniquement 2 de ces fautes ne sont pas détectées par les critères de test ($SNDR$, $Offset$ et I_{DD}), c'est pour cette raison que la Couverture de fautes n'atteint pas les 100%. La Couverture de fautes F^F est élevée parce que les fautes non détectées ont une très faible probabilité de détection. Il est à noter que le Rendement Y^F est inférieure à 60%, qui est très petit par rapport au Rendement design Y^D qui a une valeur supérieure à 99% (voir Figure 4.12). Ceci parce que les déviations des paramètres physiques tels que L et W ne sont pas considérés sous les déviations process. Dans le second cas du Tableau 5.3 (colonne 3), les déviations des paramètres physiques L et W ne sont pas considérées, mais uniquement celles des résistances. Dans ce cas, la Couverture de fautes F^F atteint les 100% avec un Rendement Y^F supérieur à 99%. Puisque ces déviations sont aussi considérées comme des déviations process, nous obtenons des résultats similaires de Rendement (Y^F et Y^D), Rendement de test (Y_T^F et Y_T^D), Couverture de rendement (Y_C^F et Y_C^D) et de Taux de défauts (D^F et D^D).

Métrique	Toutes les fautes paramétriques	Les fautes sur les résistances
F^F	96.69%	100%
Y^F	54.56%	99.22%
Y_T^F	55.56%	99.22%
Y_C^F	100%	100%
D^F	1.98%	0%

TAB. 5.3 – Les valeurs des métriques de test pour des fautes paramétriques simples.

5.5 Conclusion

Dans ce chapitre, nous avons présenté les outils de modélisation, d'injection et de simulation de fautes que nous avons développés et intégrés dans la plateforme de CAT. Nous avons aussi illustré comment utiliser cette plateforme pour automatiser la simulation de fautes et la simulation Monte Carlo en tenant compte de plusieurs bancs de test du circuit sous test. Ceci afin de calculer les différentes métriques de test analogiques. En particulier, nous avons montré comment calculer la Couverture de fautes catastrophiques et paramétriques, une fois les limites de test sont fixées. Nous nous somme basé sur les techniques et le cas d'étude d'un amplificateur opérationnel différentiel que nous avons présentées dans le Chapitre 4 précédent.

L'utilisation de cette plateforme nous a permis de conclure que d'une manière expérimentale, quand les fautes paramétriques sont limitées aux paramètres physiques déjà considérés durant les déviations process, les métriques de test calculées durant l'étape design ne diffèrent pas significativement à celles obtenues sous la présence de fautes.

Chapitre 6

Optimisation et génération de vecteurs de tests

6.1 Introduction

Dans ce chapitre, nous allons présenter des outils d'optimisation et de génération des vecteurs de test analogiques et mixtes. Nous présenterons en premier, en utilisant la technique statistique présentée dans le Chapitre 4, comment réduire l'ensemble des performances d'un circuit à tester, en particulier pour le cas d'étude que nous avons défini dans ce même chapitre. Ensuite, nous introduisons l'outil de génération de vecteurs de test multi-fréquences, orienté à la détection des fautes. Nous illustrerons une application de cet outil pour un filtre biquadratique différentiel. Ceci permettra par la suite de comprendre comment les vecteurs de test sont générés, pour la détection de fautes, pour le cas d'étude des chapitres 4 et 5. Les fautes considérées sont celles qui ne sont pas détectées par les performances et les critères de test utilisées dans ces chapitres. Ceci, sera suivi par une présentation d'un outil destiné à générer des vecteurs de test numériques pour le test des circuits analogiques. Finalement, nous présenterons un outil pour la génération de vecteurs de test pseudo-aléatoires multi-niveaux pour le test et la caractérisation des circuits non linéaires.

6.2 Optimisation des tests fonctionnels

6.2.1 Introduction

Le rôle de plus en plus important joué par le test sur le coût des circuits analogiques, impose une démarche d'optimisation des méthodes de test pour réduire le coût du test et ainsi le coût global de fabrication des circuits intégrés. Le temps de test étant un facteur déterminant dans le processus de test, il est évident que le test de toutes les performances serait très exorbitant et ne serait pas économiquement applicable durant le processus de fabrication d'où la nécessité de recourir au test d'un nombre réduit de performances, qui sera effectué dans un temps raisonnable, tout en acceptant un risque d'erreur de test négligeable. Ce problème a été déjà traité par [76] où les performances sont considérées une par une, à base d'un certain ordre jusqu'à ce que la Couverture de fautes voulue est atteinte (voir 3.3.3). Une autre technique de réduction de l'ensemble de performances a été présentée dans [17] basée sur l'utilisation des techniques d'apprentissage statistique à base des SVM[1]. La méthode présentée dans [54] montre une autre technique de réduction du nombre de performances basée sur le calcul de la matrice de redondance. Celle-ci est calculée à partir de la matrice de corrélation. L'idée de cette technique est

[1]Support Vector Machine, une technique très utilisée en analyse de discrimination bi-classes.

d'observer la distribution statistique de chaque couple de performances, puis calculer la courbe ajustée de cette distribution (par exemple en faisant un ajustement polynomial) et ses courbes ajustées de tolérance. Une fois ces courbes sont calculées, on observera si la spécification d'une des performances est contenue dans l'autre. Si ce cas de figure existe, alors il est possible d'enlever une de ces performances selon un certain seuil donné, puisque la violation de la spécification la plus large entraînera forcément la violation de l'autre spécification (qui est plus serrée que la première). Ceci permettra de déterminer les intervalles de redondance de chaque performance par rapport à une autre performance et ainsi, calculer la matrice de redondance T. Ce principe a permis de définir une nouvelle forme de l'Equation (3.23) du *temps de test* décrite dans [76]. Cette nouvelle équation est définie comme suit :

$$temps\ de\ test = \sum_{i \in T} W_i \left[\prod_{j=1}^{i-1} Y_j \right] P(N_i / G_1 \cap \cdots \cap G_{i-1}) \tag{6.1}$$

d'où l'introduction de nouveaux facteurs G_k, dont chacun représente l'événement que le circuit sous test passe la performance numéro k et N_k qui représente l'événement que la performance numéro k n'est pas éliminée de l'ensemble obtenu par l'analyse de redondance. Ainsi, $P(N_i/G_1 \cap \cdots \cap G_{i-1})$ représente la probabilité d'effectuer réellement la mesure de la performance i sachant que toutes les spécifications $(1, \cdots, (i-1))$ mesurées avant celle-ci sont vérifiées.

6.2.2 Optimisation des tests fonctionnels par estimation du Taux de défauts

Dans cette section, nous présenterons une technique qui permet de réduire l'ensemble des performances d'un CUT en se basant sur le calcul du Taux de défauts. Cette technique est le fruit d'un stage de master que nous avons co-encadré. Elle est basée sur la recherche du plus petit sous-ensemble de performances de test permettant d'atteindre un Taux de défauts désiré.

Pour chaque sous-ensemble de performances de test, le Taux de défauts est estimé sur des échantillons de CUTs générés à base de la loi multinormale comme c'est expliqué au Chapitre 4. Ces échantillons pourraient être générés en utilisant d'autres méthodes telles que l'analyse de la sensibilité. L'estimation du Taux de défauts est faite en utilisant l'Equation (2.54) ou l'Equation (4.30). Une méthode permettant de réduire l'ensemble des performances de test a été présentée dans la Section 3.3.3 par [76]. Cette méthode est nécessaire quand l'ordre sur le test des performances est important. C'est-à-dire, quand le coût de tester deux performances dépend de l'ordre choisi pour les tester. Cependant, la méthode présentée dans cette section ne tient pas compte de cet ordre. Si ce dernier n'est pas exigé, alors cette méthode est adéquate en raison de sa vitesse d'exécution qui est rapide par rapport aux algorithmes de la technique présentée dans [76].

Le principe permettant de trouver le sous-ensemble optimal de performances de test est résumé par l'algorithme de la Figure 6.1. Au départ, à partir d'un ensemble de N performances, on doit composer tous les sous-ensembles de performances de test possibles formés de $N-1$ performances. Le sous-ensemble de performances de test engendrant le plus petit Taux de défauts sera donc considéré comme étant l'optimal à la première itération. Cet ensemble optimal sera considéré à la prochaine itération comme étant l'ensemble de performances de départ (avec $N-1$ performances). Ensuite, tous les sous-ensembles de performances de test ayant $N-2$ performances de l'ensemble optimal des performances trouvé à la première itération seront formés. De la même manière, le sous-ensemble engendrant le plus petit Taux de défauts sera choisi comme étant l'ensemble de performances optimal de cette itération. Cette procédure sera refaite jusqu'à ce que la valeur du Taux de défauts obtenue pour une itération donnée est supérieure au Taux de défauts seuil D_{th} (fixé à l'avance). Le principe de cette technique par rapport aux techniques

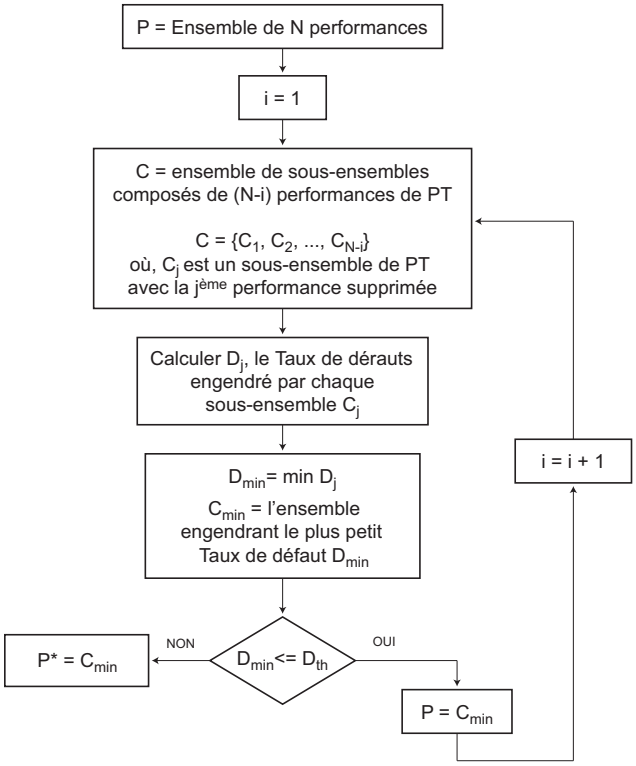


FIG. 6.1 – Algorithme de la méthode de réduction du nombre des performances.

existantes est le fait que la réduction du nombre des performances est étroitement liée au Taux de défauts. Cette métrique est essentielle pour évaluer le test, mais elle n'a pas été considérée par les autres méthodes.

6.2.3 Application au cas d'étude d'un amplificateur opérationnel différentiel

Pour le cas de l'amplificateur présenté dans le cas d'étude (Section 4.3), nous avons fixé la valeur du Taux de défauts seuil D_{th} (entre 10ppm et 100ppm), et pour chacun de ces seuils, nous avons déroulé l'algorithme de la Figure 6.1 sur 30 échantillons différents d'un million d'amplificateurs générés par simulation de la loi multinormale (voir Chapitre 4). Ce qui permet d'obtenir le nombre de performances nécessaires pour chaque Taux de défauts seuil fixé et pour chaque échantillon généré. Les résultats obtenus sont montrés par le graphe de la Figure 6.2.

A partir de ce graphe, nous pouvons calculer pour chaque nombre de performances donné, la valeur moyenne et l'intervalle de confiance engendré par celui-ci. Les résultats ainsi obtenus, pour des nombres de performances situés entre 2 et 7, sont présentés par le Tableau 6.1 pour un niveau de confiance de 95%.

Nombre de performances	D **moyen**	D **(ppm) niveau de confiance de 95%**
2	74.5	$[57, 92]$
3	58.5	$[42, 75]$
4	43.0	$[28, 58]$
5	30.5	$[19, 42]$
7	10.5	$[4, 17]$

TAB. 6.1 – Le Taux de défauts engendré par un certain nombre de performances donné.

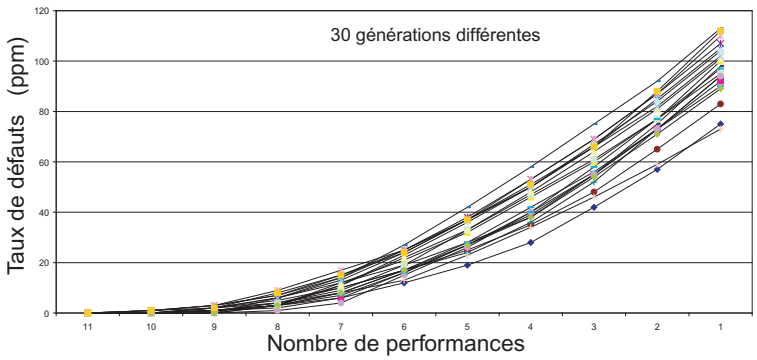


FIG. 6.2 – Algorithme de la méthode de réduction du nombre des performances.

6.2.4 Ordre d'élimination des performances

Dans ce qui suit, nous étudierons s'il existe un certain ordre sur la suppression des performances pour chaque itération de l'algorithme présenté par la Figure 6.1. C'est-à-dire, est ce que à chaque itération, c'est toujours les mêmes performances qui sont éliminées. Ceci nous aidera à connaître l'ordre de l'importance de chaque performance car la performance éliminée en premier est plus importante que la performance éliminée en dernier. Pour ce faire, nous avons généré plusieurs échantillons de CUT, et nous avons fixé une certaine valeur du Taux de défauts seuil D_{th} (entre 20ppm et 100ppm). Pour chaque D_{th} fixé, nous avons déterminé l'ensemble des performances supprimées le plus souvent en premier, puis en deuxième et ainsi de suite. Le Tableau 6.2 montre les ordres d'élimination obtenus pour chaque performance. En effet, nous avons étudié cet ordre pour des échantillons de rendements différents, et nous avons observé que les 5 premières performances données dans le Tableau 6.2 gardent toujours cet ordre, mais les 6 performances qui restent ne respectent pas toujours un ordre bien déterminé.

En conclusion, nous pouvons considérer un ensemble de 6 performances de test (SR, $PSRR\ GND$, THD, $CMRR$, $GBWD$) qui assure un Taux de défauts de $30.5 \pm 11.5 ppm$ avec un niveau de confiance de 95%.

Ordre d'élimination	**Performance**
1	SR
2	$PSRR\ GND$
3	THD
4	$CMRR$
5	$GBWD$
6	$PSRR\ VDD$
7	*Intermodulation*
8	*Gain*
9	IDD
10	*Bruit*
11	*Marge de Phase*

TAB. 6.2 – Ordre d'élimination moyen des performances.

6.3 Génération de vecteurs de test multi-fréquences orientés à la détection de fautes

Dans cette section, nous allons illustrer une approche de génération de vecteurs de test orientés aux fautes que nous avons intégré dans la plateforme de CAT. Cette approche a été développée par le passé au sein de notre laboratoire [81]. Elle est basée sur l'algorithme présenté en Section 3.4.3. Nous nous limiterons ici à illustrer son fonctionnement à base de l'outil développé. D'abord, pour un filtre biquadratique différentiel et puis, pour le cas d'étude que nous utilisons tout au long de cette thèse.

Le filtre biquadratique a été choisi pour illustration, car nous le retrouvons dans la publication originale [81] où les simulations sont considérées seulement au niveau fonction de transfert. Par contre, nous effectuerons ici des simulations au niveau circuit, exploitant la plateforme CAT. Ceci nous a permis d'obtenir des résultats plus précis.

En deuxième lieu, nous appliquerons cet algorithme pour le cas d'étude. En fait, nous avons vu dans le Chapitre 4 qu'un petit pourcentage de fautes catastrophiques (2%) ne sont pas détectées ni par les test fonctionnels, ni par les mesures de test proposées. Nous allons donc utiliser l'outil pour générer des vecteurs de test spécifiques à la détection de ces fautes.

Notons au passage que d'autres algorithmes de génération de vecteurs de test, tels que ceux décrits dans le Chapitre 3 auraient pu être utilisés (par exemple l'algorithme de la Section 3.4.4 ou 3.4.5).

Tous ces algorithmes aboutissent pratiquement aux mêmes résultats d'un point de vue théorique. Ils sont très facilement implémentables dans la plateforme CAT exploitant des outils de simulation de fautes. Nous avons choisi l'implémentation de l'algorithme de la Section 3.4.3 puisqu'il a été développé au laboratoire.

6.3.1 Application de l'algorithme pour un filtre biquadratique différentiel

Dans ce qui suit nous montrerons l'application de cet algorithme pour le cas d'un filtre biquadratique (Figure 6.3) qui est composé de 12 résistances, 4 capacités et de 3 amplificateurs opérationnels. Nous allons considérer le cas des fautes catastrophiques dans les composants passifs.

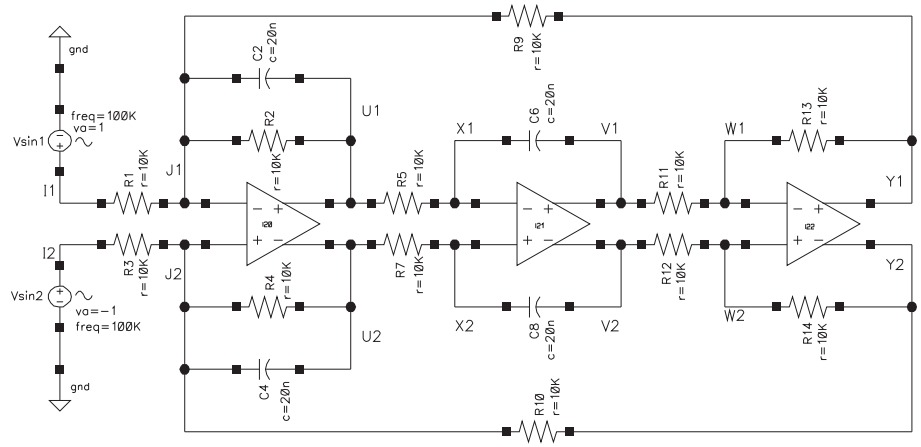


FIG. 6.3 – Filtre biquadratique.

Nous expliquerons dans ce qui suit la méthodologie utilisée pour cet amplificateur afin de générer un vecteur de test avec un nombre de fréquences optimal. Mais, nous allons tout d'abord,

présenter comment minimiser l'ensemble des mesures de test permettant d'obtenir la même Couverture de fautes que celle de l'ensemble initial.

Nous avons considéré pour cet amplificateur un ensemble de 6 mesures de test. Celles-ci sont déclarées dans l'outil en utilisant la fenêtre du programme de configuration de test. L'ensemble des intervalles de tolérance de ces mesures de test est aussi déclaré dans la même fenêtre. Les mesures de test représentent les modes communs ($MCOM$) dans les nœuds différentiels J, U, X, V, W et Y, qui sont tm1=$MCOM$(J1,J2), tm2=$MCOM$(U1,U2), tm3=$MCOM$(X1,X2), tm4=$MCOM$(V1,V2), tm5=$MCOM$(W1,W2) et tm6=$MCOM$(Y1,Y2). L'intervalle de tolérance (ou la limite de test) de chaque mesure est de $[-100mV, 100mV]$.

Une fois toutes ces informations sont entrées, une simulation de fautes doit être effectuée afin de déterminer les intervalles de fréquences permettant de détecter chaque faute. Puisque le filtre est symétrique, les fautes situées uniquement dans la partie supérieure sont considérées (6 résistances et 2 capacités) incluant les court-circuits et les circuit-ouverts dans les résistances et les capacités. Ce qui donne un nombre total de 16 fautes à injecter. Deux modèles de fautes, seulement, sont nécessaires pour injecter ces fautes. Le premier modèle, Figure 6.4(a), représente un court-circuit (la Figure 5.2(e) montre la forme schématique de ce modèle) et le deuxième modèle, Figure 6.4(b), représente un circuit-ouvert (sa forme schématique est représentée par la Figure 5.2(d)).

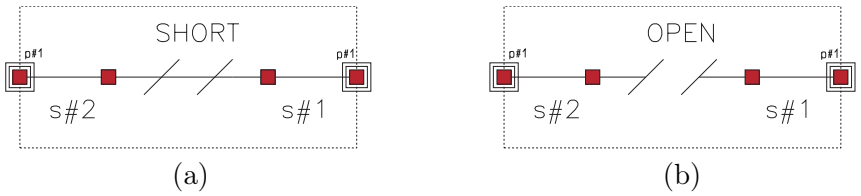


FIG. 6.4 – Modèles de fautes sous le format FMDL représentant (a) un court-circuit et (b) un circuit-ouvert.

Une fois tous ces modèles sont créés en respectant les normes du langage FMDL (Section 5.2.1), un ensemble de fichiers FID doit être créé pour déterminer comment injecter chacun de ces deux modèles (Section 5.3.3). Dans le cas de ce filtre, pour injecter toutes les fautes, uniquement 2 fichiers FID sont nécessaires (un fichier pour injecter tous les court-circuits et un fichier pour injecter tous les circuit-ouverts).

Une fois la simulation de fautes est finie, les résultats de simulation peuvent être traités. Par exemple, les différents voltages dans les différents nœuds du circuit peuvent êtres observés. La Figure 6.5 montre pour chaque mode commun ((a) le mode commun tm1, (b) le mode commun tm2, (c) le mode commun tm3, (d) le mode commun tm4, (e) le mode commun tm5 et (f) le mode commun tm6) dans la plage des fréquences $[1Hz, 100KHz]$ obtenus pour les 16 fautes injectées. L'ensemble des intervalles de fréquences permettant la détection de chaque faute est présenté par l'outil tel que le montre la Figure 6.6.

A partir de ces résultats, l'application de l'algorithme de Balas ou bien l'algorithme génétique proposé et qui est présenté dans l'Annexe A (pour résoudre le problème de recouvrement d'une grande dimension) permettra donc de trouver l'ensemble minimal des mesures de test permettant de détecter le maximum de fautes. Pour le cas de ce filtre, trois mesures de test (les modes communs dans les nœuds différentiels J, X et W) suffisent pour obtenir la même Couverture de fautes que celle obtenue pour toutes les mesures de test (qui est de 100% pour cet exemple).

Afin d'obtenir le vecteur de test avec un minimum de fréquences pour la détection de fautes, la procédure est exactement la même que la précédente (celle utilisée pour minimiser l'ensemble des mesures de test). L'outil permettra donc de trouver l'ensemble minimal de fréquences du vecteur de test permettant de détecter le maximum de fautes (la région B3=$[647Hz, 1014Hz]$

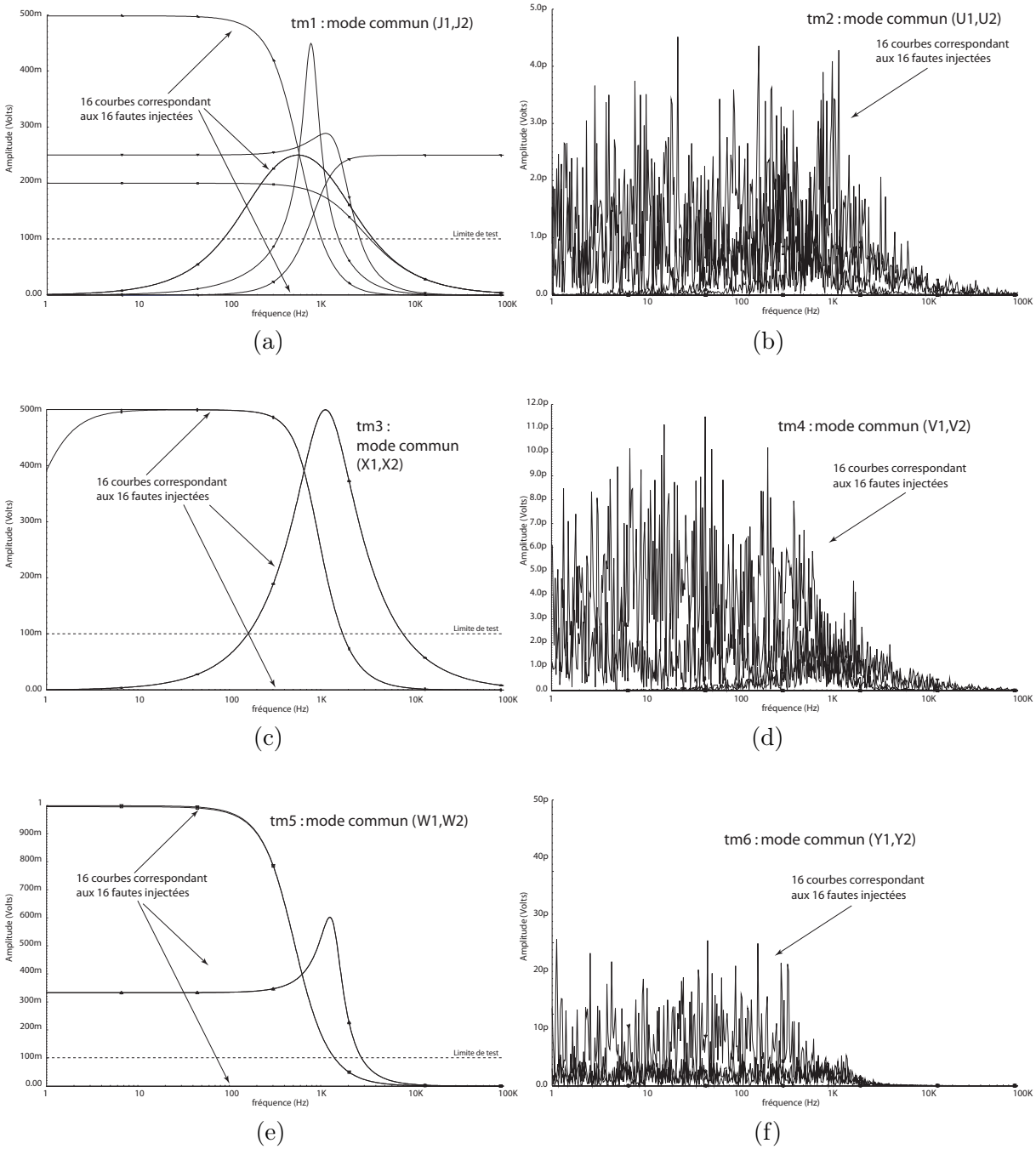


FIG. 6.5 – Les modes communs sous les effets des fautes : (a) tm1, (b) tm2, (c) tm3, (d) tm4, (e) tm5 et (f) tm6.

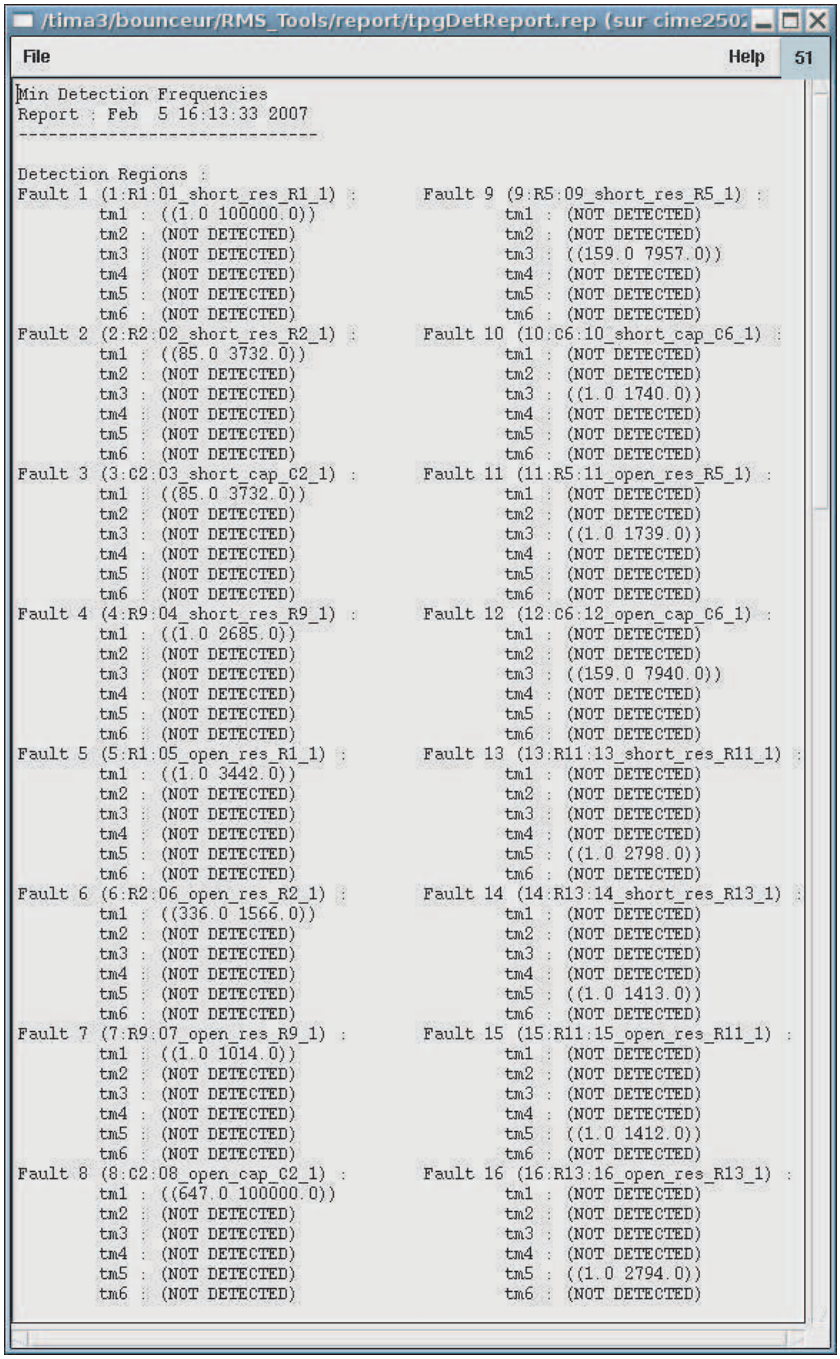


FIG. 6.6 – Intervalle fréquenciel de détection de chaque faute injectée dans le filtre.

de la Figure 6.7). Une seule fréquence de cet intervalle est suffisante pour détecter 100% de fautes. Il est à noter que cet outil permet aussi de donner les fréquences minimales permettant de diagnostiquer les fautes. Étant donné que le diagnostic de fautes ne fait pas l'objet de de cette thèse, nous n'allons pas présenter les résultats associés à cette partie.

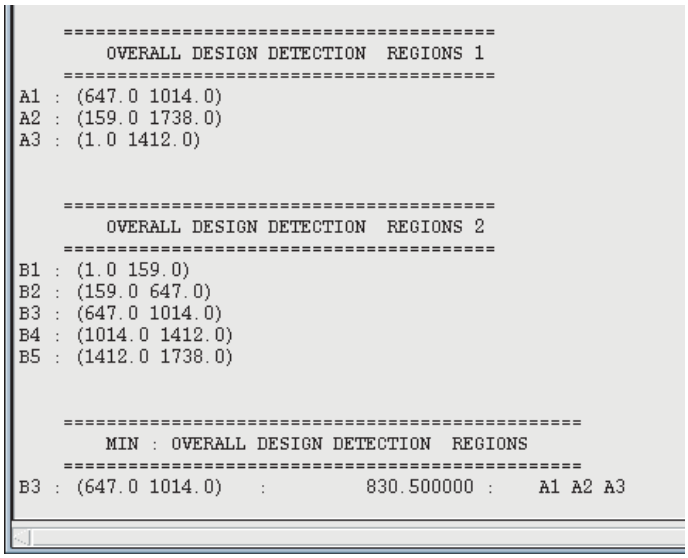


FIG. 6.7 – Les fréquences du vecteur de test utilisé pour la détection de fautes.

6.3.2 Application au cas d'études d'un amplificateur opérationnel différentiel

Pour le cas de l'amplificateur présenté dans la Section 4.3, nous avons effectué la procédure décrite dans la section précédente sur les mesures du *Gain* et de *Phase*. Ces mesures sont effectuées à base de simulations dans le domaine fréquentiel. Les tolérances considérées sont données par le Tableau 4.1. Ces tolérances diffèrent d'une plage de fréquences à une autre. Pour les fréquences situées entre $1Hz$ et $10KHz$, ces tolérances sont de $\pm 2.11dB$ pour le *Gain* et de $\pm 10°$ pour la *Phase*. Par contre, pour les fréquences situées entre $10KHz$ et $1GHz$ ces tolérances sont de $\pm 5dB$ pour le *Gain* et de $\pm 30°$ pour la *Phase*.

La Figure 6.8 montre un exemple de *Gain* sans faute et celui avec faute. Cette faute représente un court-circuit injecté entre la Grille et le Drain du transistor MP20 de l'amplificateur (voir Figure 4.2). Il est clair qu'à la fréquence $W = 100Hz$, cette faute ne peut pas être détectée, car la différence entre le *Gain* sans faute et celui avec faute est de $0.07dB$, inférieure à la tolérance de $+2.11dB$. Au contraire, pour une fréquence supérieure à $10KHz$, par exemple à $2MHz$, cette faute peut être détectée puisque la différence du *Gain* est largement supérieure à $+5dB$.

L'application de l'algorithme de génération de vecteurs de test présenté ci-dessus a permis de trouver un vecteur de test ayant 4 fréquences, qui sont $1KHz$, $4KHz$, $85KHz$ et $3MHz$. Ce vecteur de test permet d'améliorer la Couverture de fautes obtenue par le *Gain* par exemple, de 74.38% à 96.88%. En considérant ce vecteur de test, la Couverture de fautes obtenue en considérant toutes les performances de l'amplificateur est de 100%.

6.4 Génération de vecteurs de test numériques pour le test de circuits analogiques

Afin de programmer la technique de BIST harmonique de circuit mixtes présentée par [90], un outil OPTEGEN a été développé. Cet outil permet la génération d'un vecteur de test numérique

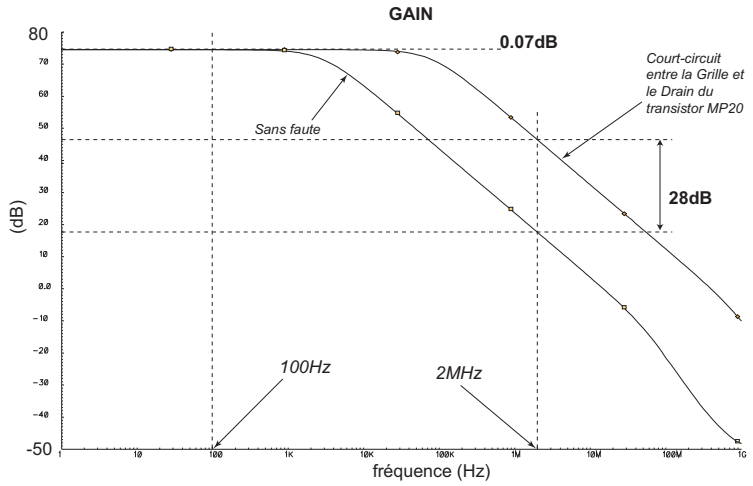


FIG. 6.8 – La performance *Gain* avec une fréquence qui ne permet pas de détecter une faute (a) et une fréquence qui permet de détecter cette faute (b).

qui code le signal de test analogique désiré sur la puce (typiquement un signal multi-fréquences) de qualité optimale par rapport à des critères tels que le THD, $SFDR$, la déviation d'amplitude ($ADev$) ou le facteur de crête[2] du signal. Cet outil a été aussi utilisé pour optimiser les signaux de test numériques dans le BIST de convertisseurs de signal $\Sigma\Delta$ présenté dans [93].

La Figure 6.9 montre le principe utilisé pour générer un signal analogique sur-puce (on-chip) à partir d'un signal numérique qui a été introduit par [44, 45]. Le vecteur de test programme la longueur du registre à décalage et le facteur de division d'horloge. Il contient aussi le train binaire à charger sur le registre. En effet, pour générer un signal analogique d'une fréquence f à partir d'un signal numérique codant ce signal, dans un premier temps T1, le signal numérique sera chargé dans le registre à décalage cadencé à une fréquence f_s qui est la fréquence d'échantillonnage du signal numérique, puis durant le temps T2, le registre à décalage sera bouclé pour périodiser le signal numérique, puis en utilisant un filtre passe-bas, le signal analogique codé sera ainsi récupéré.

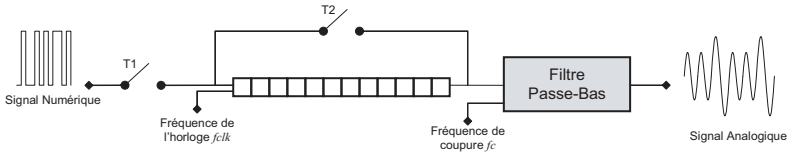


FIG. 6.9 – Génération sur-puce d'un signal analogique à partir d'un signal numérique.

Le train binaire contient une ou plusieurs périodes du signal analogique codé, comme le montre la Figure 6.10. Puisque le train binaire est périodiquement répété à la sortie du registre à décalage d'une longueur N et à une fréquence f_s, le spectre du signal d'entrée est discret et contient un nombre limité de fréquences cohérentes données par[72] :

$$f = f_s \frac{M}{N} \tag{6.2}$$

où, $M = 1, 2, ..., \frac{N}{2}$.

Uniquement les fréquences données par l'Equation (6.2) peuvent être générées par le train binaire tant que le nombre de périodes du signal de test désiré est entier. Il est évident qu'en

[2] $Peack - to - RMS$.

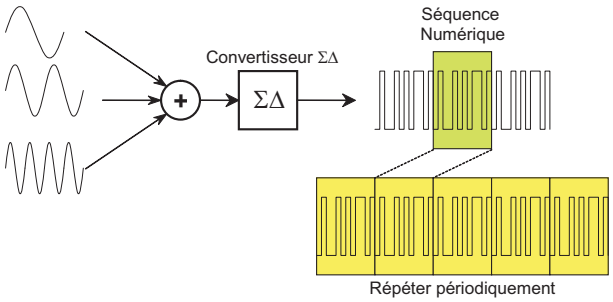


FIG. 6.10 – Génération d'un vecteur de test numérique d'une bonne qualité.

programmant la fréquence d'échantillonnage et la longueur du registre à décalage, il est possible d'avoir des fréquences avec une résolution importante (f_s/N) [80].

En pratique, la fréquence d'échantillonnage est obtenue à partir de la fréquence de l'horloge f_{ck} en utilisant un simple diviseur de fréquences comme suit :

$$f_s = \frac{f_{ck}}{2^B} \tag{6.3}$$

où, B est le facteur de division.

Comme exemple, pour générer un signal d'une seule fréquence, qui est codé par la fondamentale ($M = 1$), il faut donc utiliser la fréquence f cohérente suivante :

$$f = \frac{f_{ck}}{2^B} \cdot \frac{1}{N} \tag{6.4}$$

Un premier prototype de l'outil développé dans l'équipe [97] a été amélioré lors du stage de master que nous avons effectué [18] afin d'inclure la génération de signaux multi-fréquences au lieu de monofréquences uniquement, l'ajout d'algorithmes d'optimisation multiobjectifs et génétiques et la représentation de la sortie dans un format directement acceptable par un testeur numérique[3] [22]. La Figure 6.11 montre l'interface de cet outil qui a été utilisé pour la première fois dans [96].

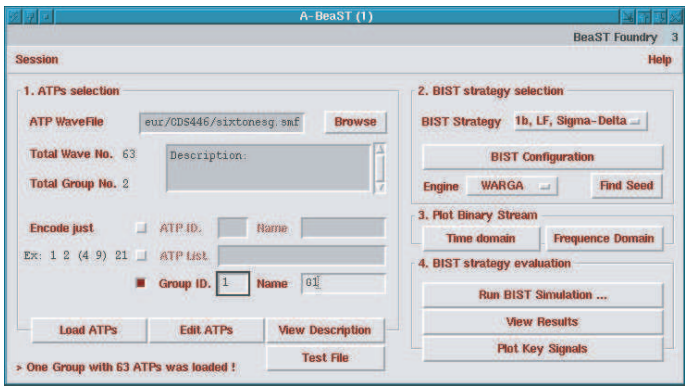


FIG. 6.11 – Interface de l'outil de l'optimisation de vecteurs de test.

Le but principal de cet outil est d'améliorer la qualité du train binaire de test à injecter sur-puce. La qualité de ce train binaire dépend des phases de chaque fréquence et du nombre de fois que ce train binaire est décalé. Cet outil utilise des algorithmes de type Monte Carlo et des Algorithmes Génétiques pour l'optimisation multiobjectif non linéaire [22]. Ces algorithmes

[3]IMS.

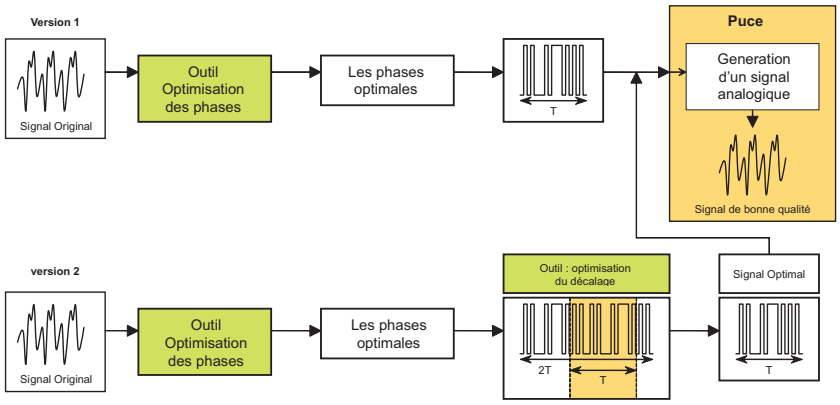


FIG. 6.12 – Technique d'optimisation de la qualité de vecteurs de test numériques.

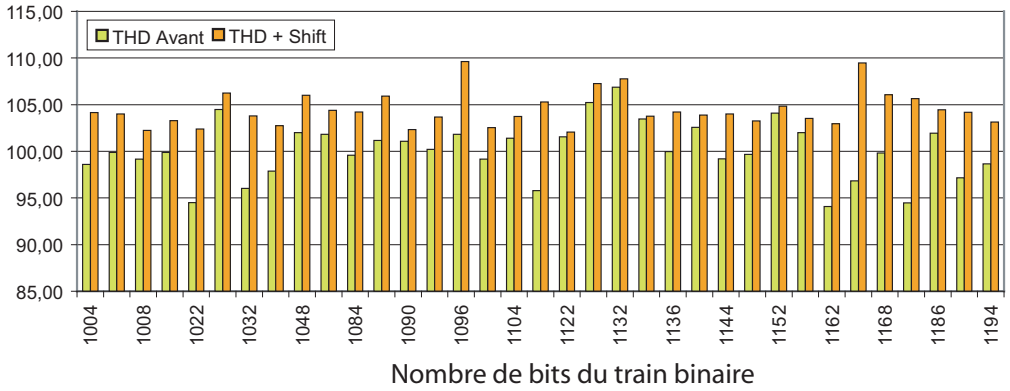


FIG. 6.13 – Les résultats de l'amélioration de la méthode d'optimisation de la qualité de vecteurs de test.

permettent donc de trouver les valeurs optimales de phases et des décalages pour lesquels le train binaire considéré est meilleur.

Dans le cadre de cette thèse cet outil a été amélioré en modifiant légèrement l'algorithme d'optimisation original. La technique d'optimisation utilisée est présentée par la Figure 6.12. Dans la première version, seuls les phases étaient considérées comme étant les paramètres nécessaires à optimiser, mais dans la deuxième version, une fois le train binaire est optimisé à base des phases, un autre algorithme basé sur le principe de Monte Carlo est utilisé pour trouver le décalage optimal de ce train binaire.

Pour des trains binaires de longueurs différentes, les $THDs$ optimisés, correspondants, sont calculés par les deux méthodes. Les résultats obtenus sont donnés par la Figure 6.13. Il est évident que les résultats sont améliorés. Cette amélioration a permis de valider une technique de BIST pour les convertisseurs Analogiques/Numériques $\Sigma\Delta$ de second ordre, et ayant une précision de 16 bits [93, 94].

6.5 Génération de vecteurs de test pseudo-aléatoires multi-niveaux

Dans le domaine du test de circuits mixtes, il n'existe pas à notre connaissance des algorithmes de génération de vecteurs de test dédiés à la caractérisation des systèmes non linéaires en utilisant des stimuli dans le domaine temporel. L'analyse de la réponse impulsionnelle permet la génération

de ces vecteurs pour le cas des circuits linéaires (voir Section 3.3.1). Cependant, l'extension de cette technique pour le systèmes non linéaires est complexe. L'analyse de sensibilité (voir Section 3.3.2) ne sont pas valables pour ces systèmes.

Notre équipe de recherche a travaillé ces dernières années sur la génération de ces stimuli pour le test et la caractérisation des dispositifs non linéaires. Cette technique est appliquée en particulier aux microsystèmes [40, 41, 42]. Une technique de BIST pseudo-aléatoire numérique a été développée par le passée pour les dispositifs linéaires [99, 35] et analysée aussi pour les dispositifs faiblement non linéaires [36]. Le BIST pseudo aléatoire, présenté dans [35, 36] ne peut pas traiter les circuits non linéaires d'une manière générale quand ils comportent à la fois des comportements linéaires et non linéaires. La raison de ceci est que le signal binaire pseudo aléatoire n'est pas totalement stimulant. C'est-à-dire qu'il ne peut pas stimuler toutes les fonctionnalités du système. Il a été démontré qu'un signal pseudo aléatoire est totalement stimulant pour un système non linéaire d'ordre N, si et seulement si, il possède au moins $N+1$ valeurs distinctes [86]. Sinon, il y aura un manque d'information sur la fonctionnalité du CUT contenue dans la réponse de test. Cette idée a été utilisée par [34] pour généraliser la méthode du BIST pseudo aléatoire pour l'appliquer aux circuits non linéaires. Dans cette nouvelle méthode, la séquence pseudo aléatoire sera remplacée par une autre séquence pseudo aléatoire multi-niveau totalement stimulante qui peut être générée sur puce de façon relativement facile. En général, les autres stimuli utilisés pour tester des circuits non linéaires sont difficilement générés, comme le cas d'un balayage de signaux sinusoïdaux afin d'effectuer une analyse spectrale. D'où la proposition de générer un type spécifique de vecteurs de test pseudo aléatoire multi-niveau. Ces vecteurs sont adaptés pour simplifier la procédure de traitement du signal pour calculer les noyaux de Volterra. Ceci est possible grace à l'utilisation du modèle de Wiener [91]. Les Noyaux de volterra représentent des fonctions qui caractérise le comportement d'un système. Le premier noyau caractérise le comportement linéaire (il est analogue à la réponse impulsionnelle dans les systèmes linéaires), le deuxième noyau décrit le second ordre du comportement non linéaire et ainsi de suite pour les autres noyaux. Cette technique est utilisable uniquement si les outils de traitement du signal (DSP[4]) sont présents sur-puce.

Dans cette section nous présenterons brièvement un outil destiné à la génération des vecteurs de test multi-niveaux pour des circuits analogiques non linéaires et au calcul des noyaux de Volterra. Nous avons développé cet outil sous C++ Builder (version indépendante) et sous SKILL pour pouvoir l'intégrer dans l'environnement Cadence.

En général, un système non linéaire invariant dans le temps peut être caractérisé par une série finie de Volterra avec une précision arbitraire selon l'Equation suivante :

$$y(k) = h_0 + \sum_{r=1}^{N} \sum_{m_1=0}^{M-1} \cdots \sum_{m_r=0}^{M-1} h_r(m_1, \cdots, m_r) \times \prod_{j=1}^{r} x(k - m_j) \tag{6.5}$$

où x et y sont, respectivement, l'entrée et la sortie du système, N est l'ordre maximal de la non linéarité, M est la mémoire du système et h_i est le $i^{ème}$ noyau de Volterra (une matrice de dimension r) qui contient toute l'information sur l'ordre du $i^{ème}$ comportement non linéaire du système.

La Figure 6.14 montre l'interface de l'outil développé pour générer un vecteur de test multi-niveau MLS[5] afin de calculer les noyaux de Volterra. Cinq paramètres doivent être entrés avant toute exécution. Ces paramètres sont listés dans le Tableau 6.3.

Durant l'exécution, une séquence MLS_{mod} à un seul niveau arbitraire de longueur M est générée en utilisant le registre à décalage approprié [91]. Une séquence MLS multi-niveau donnée

[4]Digital Signal Processing
[5]Multi Level Sequence.

FIG. 6.14 – Interface de l'outil pour la génération de vecteurs de test multi-niveaux MLS.

peut être donc générée par combinaison de plusieurs séquences MLS_i dont chacune représente la séquence MLS_{mod} avec un certain décalage approprié. La Figure 6.15 montre, par exemple, un vecteur de test multi-niveau qui correspond à une séquence MLS avec Mu=4, Order=2, Alpha=1, Sample=1 et R/F Time=1μs. Pour Mu=4, la séquence multi-niveau est courte et peut être, donc, facilement représentée.

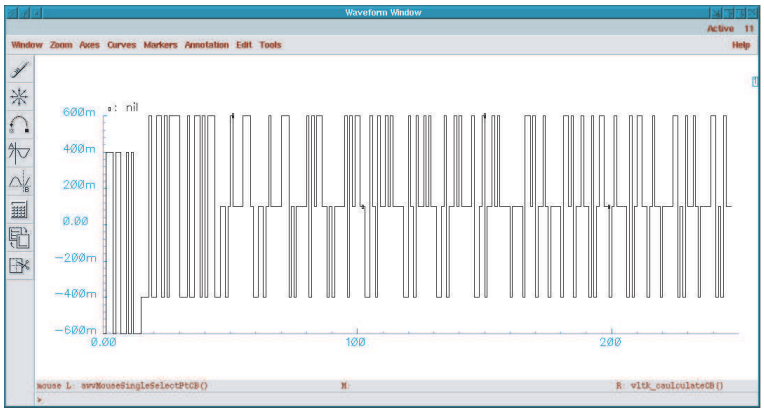


FIG. 6.15 – Vecteur de test multi-niveau (Mu=4, N=2 et Alpha=1).

La Figure 6.16 montre un exemple du $2^{ème}$ noyau de Volterra d'un accéléromètre, calculé par l'outil. Dans [34], cette méthode a été détaillée pour d'autres circuits non linéaires et dont les résultats sont obtenus à base de l'outil présenté ci-dessus.

Entrées	Signification
Mu	Longueur du registre à décalage utilisé pour générer la séquence MLS. La mémoire M est égale à $2^{Mu} - 1$.
Order	Ordre de la non linéarité N.
Alpha	Utiliser pour contrôler les amplitudes dans la séquence MLS.
Sample	Temps d'échantillonnage.
R/F Time	Temps de montée et de descente dans la séquence multi-niveau.

TAB. 6.3 – Les paramètres d'entrée de l'outil de génération de vecteurs de test multi-niveaux MLS.

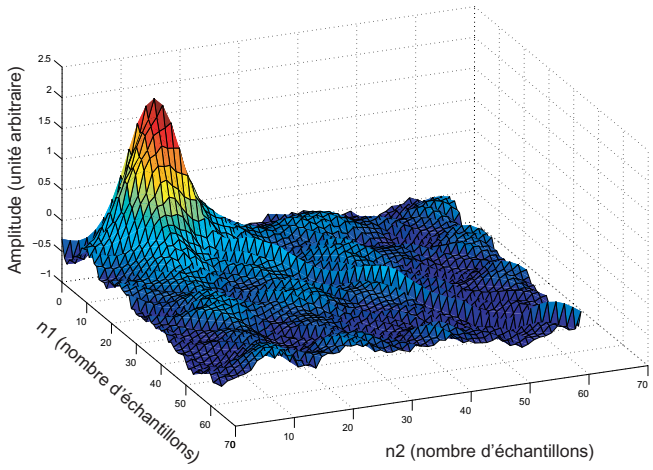


FIG. 6.16 – Le $2^{ème}$ noyau de Volterra d'un MEMS.

6.6 Conclusion

Dans ce chapitre, nous avons présenté des techniques permettant d'optimiser et de générer des vecteurs de test. Dans la première partie, nous avons présenté une nouvelle technique permettant de réduire l'ensemble des performances d'un circuit sous test engendrant un certain Taux de défauts désiré. Pour ce Taux de défauts, il peut y avoir un certain ordre sur l'élimination de certaines performances, ce qui permet donc, de choisir ces dernières comme étant les sous-ensemble de performances de test à prendre en considération. Ainsi, il est possible de réduire le temps de test. L'application de cette algorithme à notre cas d'étude a permis de réduire l'ensemble de 12 performances de départ à uniquement 6 performances.

Dans la deuxième partie de ce chapitre, nous avons présenté une application des outils développés pour générer des vecteurs de test multi-fréquences permettant de détecter le maximum de fautes. Cette application est basée sur un algorithme présenté dans [81]. Une application détaillée de cet algorithme sur un filtre biquadratique a été présentée. Ensuite, cet algorithme a été utilisé pour notre cas d'étude (amplificateur opérationnel différentiel). Un vecteur de test à deux fréquences a été trouvé pour lequel une Couverture de fautes de 100% a été obtenue, au lieu de 98% sans la considération de ce vecteur.

Nous avons aussi présenté dans la troisième partie comment optimiser un vecteur de test numérique afin de tester des circuits analogiques. Les outils développés permettent de générer des fichiers, de ces vecteurs optimaux, directement utilisables par les testeurs numériques.

La dernière partie de ce chapitre a été consacrée à la présentation de l'outil de génération

de vecteurs de test pseudo-aléatoires multi-niveaux. Ces vecteurs de test permettent ensuite de tester et de caractériser des circuits non linéaires, en calculant les noyaux de Volterra.

Chapitre 7

Conclusions et perspectives

7.1 Conclusions

Le test analogique, mixte et RF devient de plus en plus difficile à effectuer que soit du point de vue coût des équipements de test ou bien du point de vue temps de test engendrant des coûts de production très significatifs. Une des solutions les plus utilisées pour remédier à ces problèmes est de créer des circuits à l'intérieur des puces mêmes permettant d'effectuer un auto-test, ce qui est appelé BIST (Built-In-Self-Test). Une technique de BIST doit être validée avant d'être implémentée. Ceci se fait en calculant les métriques de test.

Dans cette thèse, nous proposons des techniques pour l'estimation des métriques de test analogiques permettant d'évaluer la qualité du test lors du design, avant la production. Ces techniques sont intégrées dans une plateforme de CAO pour le test (CAT[1]) integrée dans le logiciel Cadence. Elle permet aux ingénieurs de test d'effectuer les opérations suivantes :

1. Estimer des métriques de test analogiques (Couverture de fautes, Rendement, Rendement de test, Couverture de rendement et Taux de défauts) sous des déviations process, en utilisant des techniques statistiques, afin de fixer les limites des critères de test.
2. Automatiser la modélisation, l'injection et la simulation de fautes, pour estimer les métriques de test sous la présence des fautes paramétriques et catastrophiques simples.
3. Optimiser et générer des vecteurs de test, en incluant des algorithmes pour la réduction du nombre des performances de test analogiques, pour la détection de fautes, pour l'optimisation de vecteurs de test numériques codant des signaux analogiques et pour le test et la caractérisation de circuits non linéaires dans le domaine temporel.

L'utilisation des techniques statistiques, que nous avons proposées, pour évaluer les différentes métriques de test d'une manière précise, nous a conduit à l'élaboration du schéma suivant pour l'évaluation d'une procédure de test :

1. Effectuer une simulation Monte Carlo du circuit sous test sous Cadence, en considérant des déviations process.
2. Calculer les valeurs des paramètres statistiques des performances et des critères de test (μ et Σ dans le cas de la loi multinormale), ou d'une autre manière, les distributions des performances et des critères de test.
3. Estimer la densité de probabilité conjointe des performances et des critères de test (dans notre cas d'étude, cette densité est supposée multinormale).

[1]Computer Aided Test.

4. Générer une nouvelle, *large*, population de circuits (sous Matlab, R ou un autre logiciel) ayant les mêmes distributions que celles des performances et des critères de test obtenues par simulation.
5. Estimer les métriques de test analogiques, lors de l'étape du design, sur cette nouvelle population.
6. Dans le cas où l'ensemble des performances est très large, réduire l'espace des performances à base du calcul statistique du Taux de défauts, puis refaire les étapes 1 à 6 avec ce nouvel ensemble réduit.
7. A base de cette évaluation, fixer les limites de test pour chaque critère de test, par exemple, comme compromis entre le Taux de défauts et la Perte de rendement.
8. Estimer les métriques de test sous la présence de faute paramétriques et catastrophiques.
9. Générer les vecteurs de test orientés à la détection de fautes pour augmenter la Couverture de fautes.

Nous avons illustré l'application de cette plateforme sur un amplificateur opérationnel différentiel conçu sous la technologie $0.18\mu m$ de ST Microelectronics.

7.2 Perspectives

La plateforme CAO pour le test développée permet actuellement d'évaluer des techniques de test pour différents types de circuits (circuits RF : LNA [120, 121, 122] et PLL [11], et imageurs CMOS [71]).

Cette plateforme nécessite plusieurs extensions dans plusieurs domaines différents, à savoir :

1. Ajout des techniques d'évaluation de test et de simulation de fautes pour le cas des circuits RF. Ces travaux sont en cours de réalisation et ont déjà pu servir dans le cadre de test d'un LNA [122].
2. Ajout de la possibilité d'injection des fautes au niveau layout et au niveau comportemental.
3. Ajout d'autres techniques d'optimisation et d'évaluation de test basées sur la modélisation statistique des circuits sous test, en éliminant l'hypothèse de la loi multinormale des performances et des critères de test. Nous suggérons, l'utilisation des techniques de type *bootstrap* si le nombre de circuits défaillants générés pendant la simulation Monte Carlo est suffisant. Sinon, utiliser des techniques de l'estimation non paramétrique de la densité de probabilité conjointe des performances et des critères de test.
4. Souvent la génération des circuits défaillants représente un événement rare. Ce qui peut influencer sur les précisions des calculs des métriques de test et dans certain cas, l'estimation directe des métriques de test est impossible, surtout lorsque la distribution des performances et des critères de test n'est pas connue. Nous proposons donc, l'utilisation des techniques de l'échantillonnage d'importance (*Importance Sampling*). Une première application de ce type de techniques a été proposée en Annexe B et a été utilisée pour le cas de l'amplificateur opérationnel différentiel de notre cas d'étude. Les résultats présentés dans le Chapitre 4 montrent clairement une amélioration des calculs à base de cette technique.
5. Ajout des techniques permettant de réduire l'espace des performances.
6. Ajout de techniques permettant l'extraction des paramètres statistiques des performances et des critères de test, qui permettent d'éviter la simulation Monte Carlo, en particulier pour des systèmes complexes dont le temps de simulation est prohibitif.

Il est à noter que l'un des inconvénients majeurs de l'utilisation de la plateforme développée, est le temps de simulation. Ceci est dû aux simulateurs utilisés sous Cadence, donc, plus ces simulateurs seront rapides, plus l'utilisation de la plateforme est nécessaire, voire indispensable pour le test analogique. Cependant, la vitesse des ordinateurs de nos jours s'améliore d'une manière exponentielle, et par conséquent, l'amélioration de la vitesse des simulateurs est aussi exponentielle. Ce qui justifie l'utilité à long terme et le besoin continu de la plateforme présentée dans le cadre de cette thèse.

Annexe A

Algorithmes pour le problème de recouvrement

A.1 Problèmes de recouvrement

A.1.1 Problème classique

Pour illustrer le problème de recouvrement, nous prenons un des problèmes considérés en test présenté dans [81] où on cherche à trouver l'ensemble minimal de mesures détectant le maximum de fautes. Si on considère uniquement l'ensemble des fautes qui sont détectées par toutes les mesures (qui doit être généralement l'ensemble de toutes les fautes), donc ce problème est appelé problème de recouvrement. Pour modéliser ce problème en tant qu'un PL, soient F un ensemble m fautes et T un ensemble de n mesures de test tels que : $F = F_1, F_2, \cdots, F_m$ et $T = T_1, T_2, \cdots, T_n$. Soit la variable $a_{i,j}(i = 1, \cdots, m; j = 1, \cdots, n)$ qui prend la valeur 1 si la faute i est détectée par la mesure de test j et la valeur 0 sinon. Soit la variable de décision x_i qui prend la valeur 1 si la mesure de test T_i appartient à l'ensemble min de mesures de test et qui prend la valeur 0 sinon. L'objectif est donc de trouver l'ensemble minimal $T' \subseteq T$ (c'est-à-dire les valeurs des x_i) détectant le toutes les fautes. Le PL de ce problème s'écrit comme suit :

$$\begin{array}{l} \min \sum_{i=1}^{i=n} x_i \\ s.c. \end{array} \quad \begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & \geq & 1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & \geq & 1 \\ \vdots & & & & & & & & \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & + & a_{mn}x_n & \geq & 1 \end{array} \quad (A.1)$$

$$x_{i,(i=\overline{1,n})} \in \{0,1\}$$

La forme matricielle de ce problème est la suivante :

$$\begin{array}{c} \min \ \mathbf{1}X \\ AX \geq 1 \\ X \in \{0,1\}^n \end{array} \quad (A.2)$$

où $\mathbf{1}$ représente le vecteur unité à n dimension, $X = (x_1, X_2, \cdots, x_n)$ et et A la matrice ayant comme éléments à la ligne i colonne j les valuers a_{ij}.

L'algorithme idéal pour trouver une solution à ce problème est d'énumérer toutes les solutions (toutes les valeurs possibles des x_i) et choisir celle qui donne la solution minimiale. Mais pour un ensemble de 50 values (ou mesures de test), $2^{50} = 1125899906842624 \approx 1126 \cdot 10^{12}$ énumérations sont nécessaires pour trouver une solution à ce problème. Et que doit on dire si on a un 100

mesures de test ? Une des solutions existante pour résoudre ce genre de problème est l'utilisation de la technique utilisée pour résoudre un PLNE (Programme Linéaire en Nombres Entiers) qui est basée sur la technique de Branch and Bound. Une technique utilisant ce principe basée sur les tests de Ballas permet de trouver une solution rapide à ce problème. Pour exemple, soient un ensemble de 5 fautes $F = \{F_1, F_2, F_3, F_4, F_5\}$ et un ensemble de 4 mesures de test $T = \{T_1, T_2, T_3, T_4\}$. Soit la matrice A suivante :

$$A = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

La première colonne de la matrice A (ie. $A_1 = (10001)^T$) signifie que la mesure de test T_1 détecte les fautes F_1 et F_5, la quatrième colonne de $A(ie.A_4 = (10101))$ signifie que la mesure de test T_4 détecte les fautes F_1, F_3 et F_5. Donc le PL de ce problème s'écrit comme suit :

$$\begin{array}{l} \min \sum_{i=1}^{i=n} x_i \\ s.c. \end{array} \quad \begin{array}{rcrcrcrcl} & & & & & & x_4 & \geq & 1 \\ & & x_2 & & & & & \geq & 1 \\ & & x_2 & & & + & x_4 & \geq & 1 \\ & & & & x_3 & & & \geq & 1 \\ x_1 & + & x_2 & & & + & x_4 & \geq & 1 \end{array} \quad (A.3)$$

$$(x_1, x_2, x_3, x_4) \in \{0,1\}^4$$

La solution optimale de ce problème est $X = (0, 1, 1, 0)$ qui signifie que les mesures de test T_2 et T_3 suffisent pour détecter toutes les fautes.

Il est à noter que cet algorithme peut être améliorer en programmation en se basant sur la réduction de la matrice des contraintes. Ceci en éliminant chaque contrainte redondante (ie. ligne redondante) et chaque colonne redondante. Une ligne l_i (resp. colonne c_i) est dite redondante si il existe une autre ligne l_j (resp. colonne c_j) qui possède les 1 dans les mêmes colonnes (resp. lignes) et que le nombre de 1 de l_i (resp. c_i) est supérieur (resp. inférieur) ou égale au nombre de 1 de l_j (resp. c_j). La Figure A.1 montre un exemple d'une ligne redondante (ie. une contrainte redondante) et une colonne redondante (ie. cette variable est égale à 0, cette mesure de test n'est pas nécessaire).

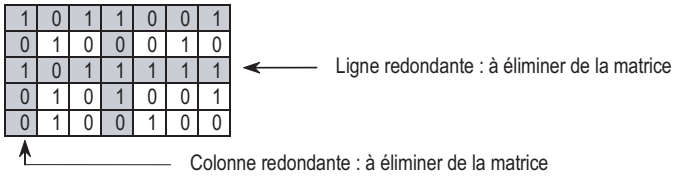


FIG. A.1 – Contraintes redondantes et variables redondantes.

Le nombre de mesure de test est généralement pas très grand. Ainsi, l'utilisation de l'algorithme de Balas qui fournit une solution optimale suffit pour résoudre ce problème. Par contre pour le cas de diagnostic de fautes où le nombre de variables est souvent très grand, nous avons développer une heuristique basée sur les algorithmes génétiques classiques [51] pour résoudre ce PL. Cet algorithme appelé MCMGA est décrit dans la section suivante.

A.1.2 Algorithme Génétique Modifié et Monte Carlo MCMGA

Résolution du problème de recouvrement en utilisant la méthode Algorithme Génétique Modifié et Monte Carlo (*Monte Carlo Modified Genetic Algorithm*)

D'abord nous commencerons par donner quelques définitions spécifiques à cet algorithme :

1. Gène : c'est l'ensemble des éléments d'un génome. Il prend une valeur binaire (0 ou 1).
2. Génome (x) : c'est une solution x qui est sous forme d'un code binaire. Exemple : $x =$ 1011010
3. K-Génome (x^k) : C'est un Génome ayant K gène 1. Exemple : $x^4 =$ **1**0**11**0**1**00 est un 4-Génome et $x^2 = 000$**1**00**1**0 est un 2-Génome.
4. Population : un ensemble de Génomes ou de K-Génomes.
5. Croisement Modifié (Modified Crossover) : c'est le croisement d'un algorithme génétique classique, seulement dans ce cas, les croisements se font avec des Génomes nuls et avec une probabilité de 1. Exemple : x_1 : **100110100** et x_2 (nul) : 000000000. Soit un croisement au $3^{ème}$ bit, le résultat sera donc : x'_1 : **100**000000 et x'_2 : 000**110100**
6. Mutation Modifiée (Modified Mutation) ou Fixation à Zéro : c'est la mutation d'un algorithme génétique classique, seulement elle ne concerne que les gènes 1 qui seront donc fixés à zéro. Les mutations multiples sont possibles et elles dépendent de la probabilité de mutation. Exemple : $x = 1001010$, voici quelques mutations possibles sur x : $x_1 =$ **0**001010, $x_2 = 100$**0**0**0**0 et $x_3 = 10010$**0**0.
7. I-Mutaion $h(x, i)$: une I-Mutation $h(x, i)$ est une mutation du $i^{ème}$ gène 1 de x. D'une autre manière, c'est la fixation à zéro du $i^{ème}$ gène 1 de x. Exemple : $x =$ 10**1**00010, $h(x, 2) = 10$**0**00010.
8. Fitness $f(x)$: c'est le nombre de gènes 1 du génome x. Exemple : $x = 0$**1**0**111**0, $f(x) = 4$.
9. La solution optimale $x*$: représente la solution optimale courante.

Chaque itération de l'algorithme MCMGA nécessite une solutions réalisable d'où l'utilisation d'un autre algorithme appelé *Algorithme de Génération Monte-Carlo (MCG)* qui permet de générer une séquence de 0 et de 1 aléatoirement de telle sorte qu'elle soit proche de la solution possible du problème de recouvrement en question. Cet algorithme ce résume comme suit :

Algorithme A.1 : Algorithme de Génération Monte Carlo (MCG)

ENTREES et INITIALISATIONS :
  $x*$ la solution courante
  Soit iMAX : le nombre maximum d'itérations
  Soit $K = f(x*)$
POUR i = 1..iMAX **FAIRE**
  $x = x^{K-1}$ // *x = génération d'un (K-1)-Génome*
  SI $f(x) < f(x*)$ **ALORS**
    $x* = x$
    $K = K - 1$
  FIN SI
FIN POUR
SORTIES : $x*$

L'Algorithme Génétique Modifié et Monte Carlo (MCMGA) est présenté par la Figure A.2.

A.2 Cas particulier sur les intervalles

Dans cette partie nous présenterons un cas particulier du problème de recouvrement pour les graphes biparties utilisé dans [81] pour minimiser les régions de détection ou de diagnostic de fautes.

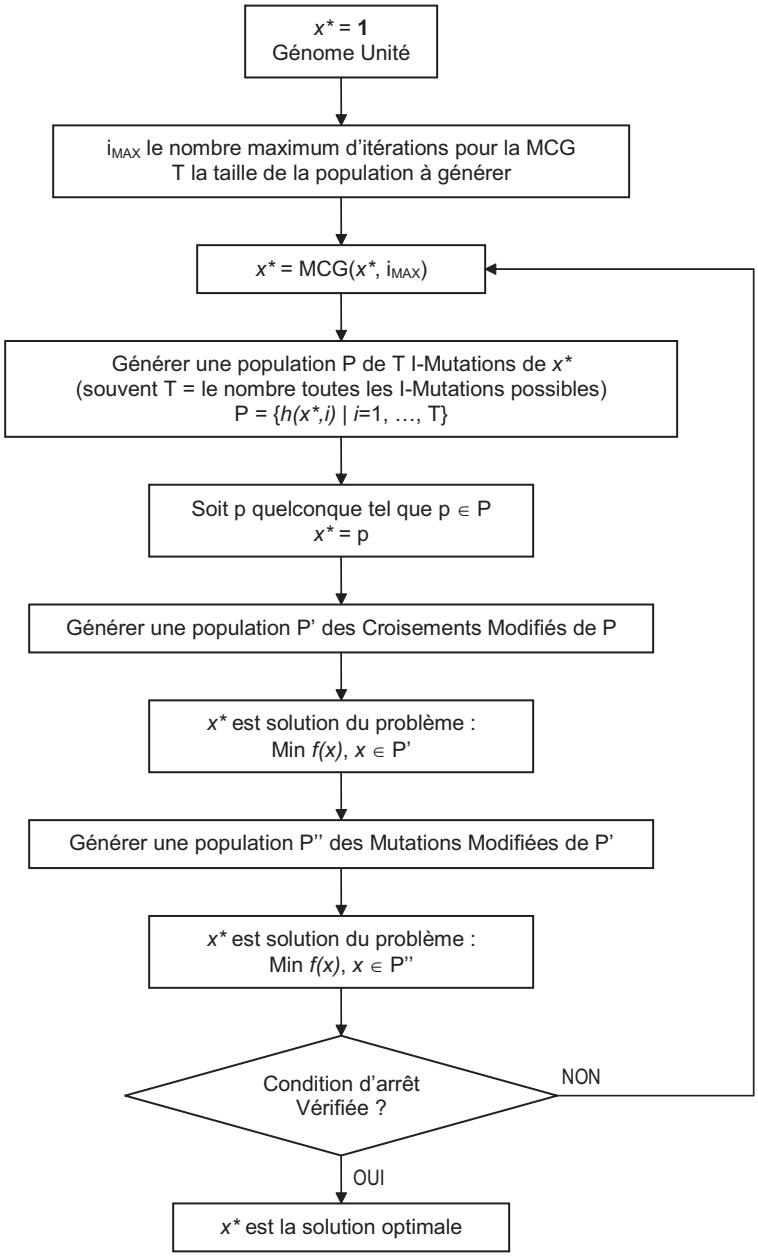


FIG. A.2 – L'algorithme MCMGA.

Soit $V2$ l'ensemble de M régions (de détection ou de diagnostic) de tout le circuit à tester. Soit $V1$ l'ensemble de N régions possibles obtenues à partir des régions de $V2$ et qu'il faut minimiser. Donc, ce problème peut être représenté comme un graphe bipartie (ie. il n'y a pas d'arcs entre les sommets de $V1$ et il n' y a pas d'arcs entre les sommets de $V2$, il y a uniquement des arcs dont une extrémité est un sommet de $V1$ et l'autre est un sommet de $V2$). On dira qu'il existe un arc d'un sommet $v_1 \in V1$ à un sommet $v_2 \in V2$ si la région v_1 se trouve à l'intérieur de la région v_2. Cette algorithme a été présenté dans [12], pour l'illustrer, soit l'exemple présenté ar la Figure A.3.

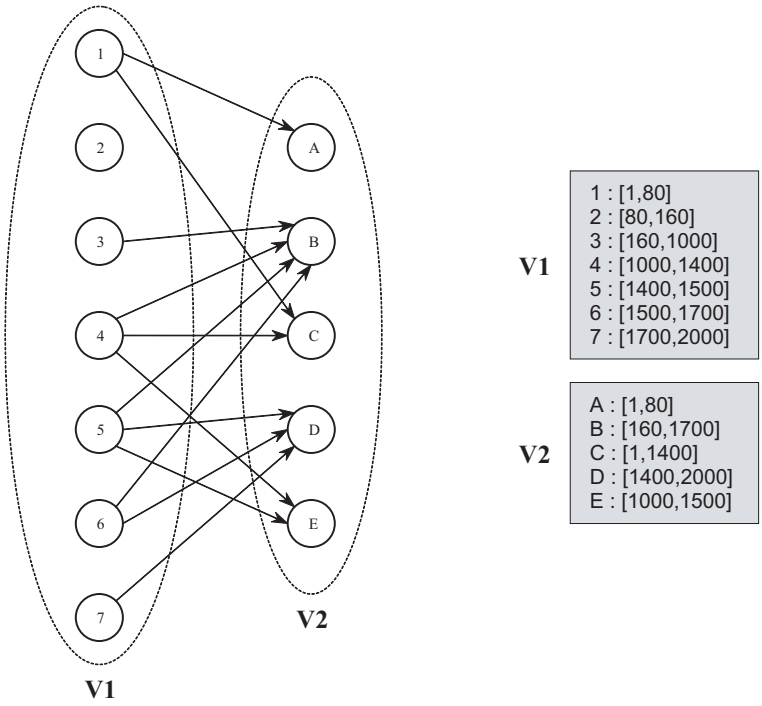


FIG. A.3 – Graphe bipartie du problème de minimum de region de détection ou de diagnostic.

Pour chaque sommet v_i de $V1$ nous affecterons une valeur $d(v_i)$ qui est égale à sont degré (ie. le nombre d'arcs qui sortent de ce sommet) puis le sommet qui possède le plus grand degré sera choisi. Dans cet exemple c'est le sommet 4 (ou 5). Donc la région numéro 4 sera prise (voir itération 1 de la Figure A.4(a)). Cette région est contenue dans les régions B, C et E. Donc tous les arcs entrants vers ces sommets seront supprimés du graphe (l'opération est illustrée Figure A.4(b) et le résultat est donné Figure A.4(c)). Le graphe ainsi obtenu (Figure A.4(c)) sera considéré comme étant le nouveau graphe et on refait la même démarche précédente pour celui-ci jusqu'à ce que tous les sommets de $V2$ seront considérés. Dans ce cas là tous les sommets de $V1$ auront un degré égal à 0 (itération 4 de la Figure A.4). Ainsi nous avons obtenu les régions 1, 4 et 5 comme solution.

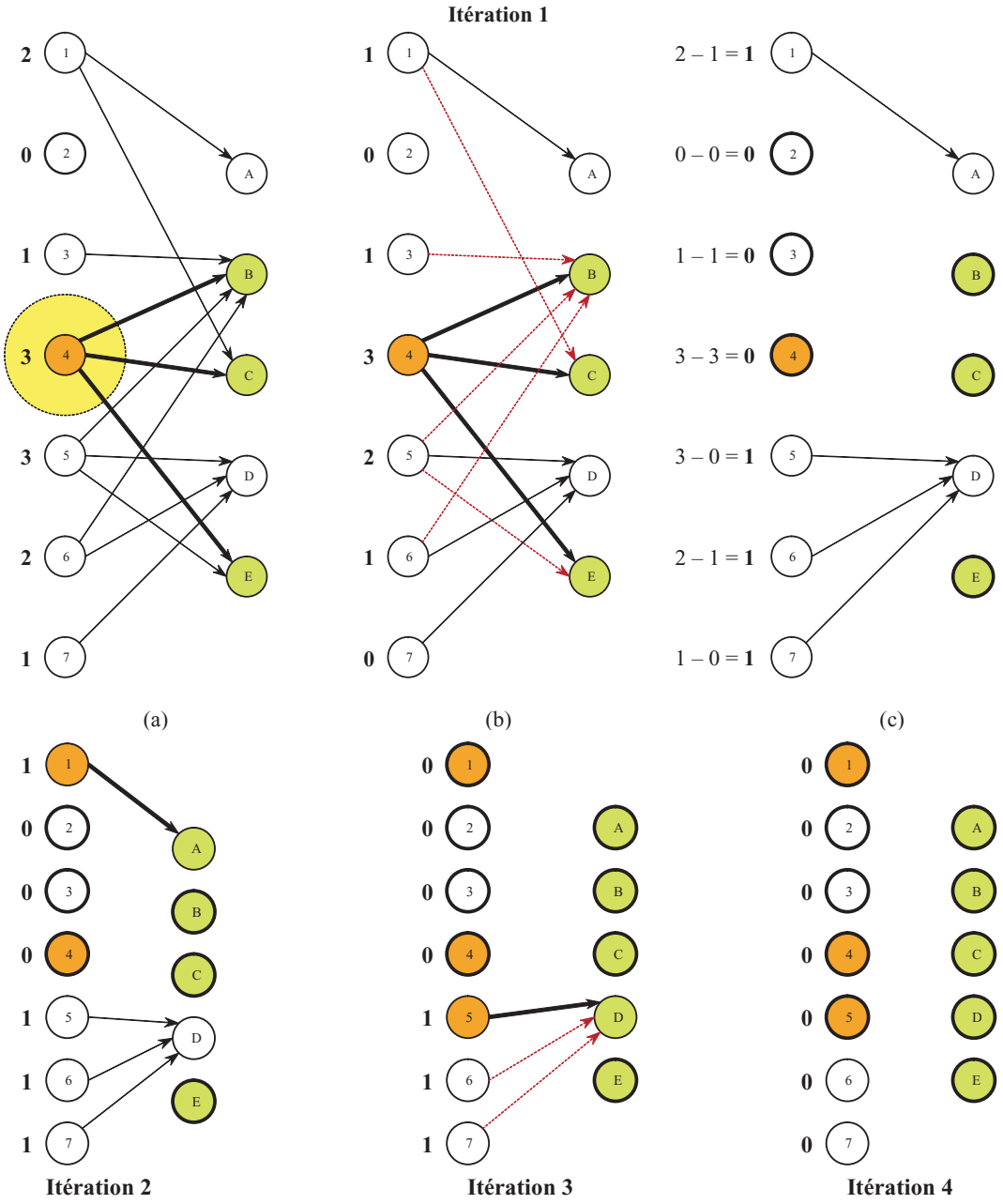


FIG. A.4 – Algorithme de résolution du problème de minimum de region de détection ou de diagnostic.

Annexe B

Amélioration de la précision pour l'estimation des métriques de test

B.1 Formulation du problème

Le chemin que nous suivons pour évaluer une technique de test, lors de la phase du design, est de générer un ensemble de circuits avec des déviations paramétriques aléatoires et tester ensuite si cette technique est efficace ou non. Évidemment, les métriques de test sont fonction du Rendement. Pour un process avec un très grand rendement (exemple, 100dppm[1]) une prédiction précise des métriques de test est difficile car, pour un nombre petit de simulations Monte Carlo, car l'ensemble de circuits défaillants généré n'est pas significatif, ou d'une autre manière, la génération des circuits défaillants représente un événements très rares. Pour remédier à ce problème, nous proposons de changer la distribution de probabilité des performances en élargissant l'écart type de chacune d'elle afin de pouvoir générer des circuits défaillants pour un nombre petit de simulations Monte Carlo. La Figure 4.5 montre un exemple d'une distribution Gaussienne élargie d'une performance.

B.2 La relation entre la distribution élargie et l'originale

Soient n performances $s = (s_1, s_2, ..., s_n)$ et m critères de test $t = (t_1, t_2, ..., t_m)$. Soit S la variable aléatoire représentant les performances et T la variable aléatoire représentant les critères de test. Soit $f_S(s)$ la densité de probabilité conjointe des performances, $f_T(t)$ la densité de probabilité conjointe des critères de test et $f_{ST}(s,t)$ la densité de probabilité conjointe des performances et des critères de test. Une nouvelle distribution des performances est considérée en augmentant par un certain rapport α ($\alpha \geq 1$) les écart-types σ_i de chaque performance i. Soit $g_S(s)$ la densité de probabilité de cette nouvelle distribution. Nous noterons la nouvelle densité de probabilité conjointe des nouvelles performances et des critères de test par $g_{ST}(s,t)$. Ces nouvelles distributions peuvent être obtenues par un changement de variables sur les perforamnces. Ce changement de variables doit être à base d'une fonction h monotone, ascendante et inversible (dans notre cas, ces conditions sont vérifiées, car, tout changement linéaire d'une gaussienne donnera une autre gaussienne). Donc, pour notre cas, ce changement de variable est donné comme suit :

$$y = h(s) = \alpha \cdot s \tag{B.1}$$

Ceci, nous permettra donc de définir $g_S(s)$ et $g_{ST}(s,t)$ comme suit :

$$g_S(s) \equiv f_Y(h(s)) = f_Y(y) \tag{B.2}$$

[1]defective part per million, qui représente le nombre de circuits défaillants sur un million.

$$g_{ST}(s,t) \equiv f_{YT}(h(s),t) = f_{YT}(y,t) \tag{B.3}$$

Nous, démontrons dans ce qui suit qu'il existe une relation entre les distributions des performances originales et les nouvelles distributions élargies.

$$\begin{aligned} g_{ST}(s,t) &= f_{YT}(y,t) && \text{(B.4)}\\ &= f_{YT}(h(s),t) && \text{(B.5)}\\ &= f_{T/Y}(t/h(s)) \times f_Y(h(s)) && \text{(B.6)}\\ &= f_{T/Y}(t/h(s)=k) \times f_Y(h(s)) && \text{(B.7)}\\ &= f_{T/Y}(t/s=h^{-1}(k)) \times f_Y(h(s)) && \text{(B.8)}\\ &= f_{T/S}(t/s) \times f_Y(h(s)) && \text{(B.9)}\\ &= \frac{f_{ST}(s,t)}{f_S(s)} \times f_Y(y) && \text{(B.10)} \end{aligned}$$

En remplaçant (B.2) dans (B.10) nous obtenons donc la formule suivante :

$$\boxed{f_{ST}(s,t) = \frac{f_S(s)}{g_S(s)} \times g_{ST}(s,t)} \tag{B.11}$$

B.3 Estimation des métriques de test

Nous présenterons dans ce qui suit comment calculer les métriques de test en utilisant l'Equation (B.11). Nous allons juste détailler le calcul du Taux de défauts D (Defect Level). Les autres métriques seront directement présentées et elles sont calculées de la même manière.

B.3.1 Taux de défauts D

Le Taux de défauts D représente la proportion des dispositifs défaillants parmi ceux qui passent le test. En d'autres termes, D est la probabilité pour qu'un circuit soit défaillant sachant qu'il passe le test. Soit $S = (S_1, S_2, ..., S_n)$ un vecteur de n performances où chaque performance S_i est une distribution gaussienne centrée avec un écart type σ_i, $(i = 1, 2, ..., n)$. Soit $A = (A_1, A_2, ..., A_n)$ le vecteur des spécifications du circuit où $A_i = [a_i, b_i]$ est la spécification de la $i^{ème}$ performance. Soit $T = (T_1, T_2, ..., T_m)$ un vecteur de m critères de test et $B = (B_1, B_2, ..., B_m)$ le vecteur de leurs limites de test. Donc,

$$\begin{aligned} D &= \mathbf{P}(\text{circuit est défaillant/il passe le test}) && \text{(B.12)}\\ D &= 1 - \mathbf{P}(\text{circuit est fonctionnel/il passe le test}) && \text{(B.13)}\\ &= 1 - \frac{\mathbf{P}(\text{circuit est fonctionnel et passe le test})}{\mathbf{P}(\text{circuit passe le test})} && \text{(B.14)}\\ &= 1 - \frac{\int_A \int_B f_{ST}(s,t)\ ds\ dt}{\int_B f_T(t)\ dt} && \text{(B.15)} \end{aligned}$$

où

$$\int_A \int_B f_{ST}(s,t)\ ds\ dt = \int_{A_1} ... \int_{A_n} \int_{B_1} ... \int_{B_m} f_{ST}(s_1, ..., s_n, t_1, ..., t_m)\ ds_1 ...\ ds_n\ dt_1 ...\ dt_m \tag{B.16}$$

et

$$\int_B f_T(t)\ dt = \int_{B_1} ... \int_{B_m} f_T(t_1, ..., t_m)\ dt_1 ...\ dt_m \tag{B.17}$$

Pour estimer D nous supposerons d'abord que nous avons seulement deux performances et deux critères de test. Donc, (B.15) sera écrit comme suit :

$$\boxed{D = 1 - \frac{D1}{D2}} \tag{B.18}$$

où

$$D1 = \int_{A_1}\int_{A_2}\int_{B_1}\int_{B_2} f_{ST}(s_1, s_2, t_1, t_2)\ ds_1\ ds_2\ dt_1\ dt_2 \tag{B.19}$$

et

$$D2 = \int_{B_1}\int_{B_2} f_T(t_1, t_2)\ dt_1\ dt_2 \tag{B.20}$$

Désormais, nous utiliserons le symbol $\int_{AB}$ pour spécifier les quatre intégrales $\int_{A_1}\int_{A_2}\int_{B_1}\int_{B_2}$. En utilisant la transformation (B.11), $D1$ sera écrit comme suit :

$$\boxed{D1 = \int_{AB} \frac{f_S(s_1, s_2)}{g_S(s_1, s_2)}\ g_{ST}(s_1, s_2, t_1, t_2)\ ds_1\ ds_2\ dt_1\ dt_2} \tag{B.21}$$

Pour transformer $g_{ST}(s_1, s_2, t_1, t_2)$ en une distribution de probabilité dans A_1, A_2, B_1 et B_2 que nous appelons u_{ST}, nous utiliserons les équations suivantes :

$$C1 = \int_{AB} g_{ST}(s_1, s_2, t_1, t_2)\ ds_1\ ds_2\ dt_1\ dt_2 \tag{B.22}$$

Nous pouvons donc créer une densité de probabilité u_{ST} comme suit :

$$u_{ST}(s_1, s_2, t_1, t_2) = \begin{cases} \frac{g_{ST}(s_1, s_2, t_1, t_2)}{C1} & \text{si } s_1 \in A_1, s_2 \in A_2, t_1 \in B_1 \text{ et } t_2 \in B_2 \\ 0 & \text{sinon.} \end{cases} \tag{B.23}$$

Donc,

$$D1 = C1 \cdot E1 \tag{B.24}$$

où

$$E1 = \int_{AB} \frac{f_S(s_1, s_2)}{g_S(s_1, s_2)}\ u_{ST}(s_1, s_2, t_1, t_2)\ ds_1\ ds_2\ dt_1\ dt_2 \tag{B.25}$$

$$= E_u\left(\frac{f_S(s_1, s_2)}{g_S(s_1, s_2)}\right) \tag{B.26}$$

représente l'espérance de $\frac{f_S(s_1,s_2)}{g_S(s_1,s_2)}$ dans le domaine A_1, A_2, B_1 et B_2.

Pour estimer $D2$ nous utiliserons les lois marginales et la transformation (B.11), nous pouvons donc écrire :

$$f_T(t_1, t_2) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_{ST}(s_1, s_2, t_1, t_2)\ ds_1\ ds_2 \tag{B.27}$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{f_S(s_1, s_2)}{g_S(s_1, s_2)}\ g_{ST}(s_1, s_2, t_1, t_2)\ ds_1\ ds_2 \tag{B.28}$$

En remplaçant (B.28) dans (B.20) nous obtenons :

$$\boxed{D2 = \int_{B_1}\int_{B_2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{f_S(s_1, s_2)}{g_S(s_1, s_2)}\ g_{ST}(s_1, s_2, t_1, t_2)\ ds_1\ ds_2\ dt_1\ dt_2} \tag{B.29}$$

Pour transformer $g_{ST}(s_1, s_2, t_1, t_2)$ en une densité de probabilité dans B_1 et B_2 que nous appellerons v_{ST}, nous utiliserons les équations suivantes :

$$C2 = \int_{B_1}\int_{B_2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g_{ST}(s_1, s_2, t_1, t_2)\ ds_1\ ds_2\ dt_1\ dt_2 \tag{B.30}$$

Nous pouvons alors créer une densité de probabilité v_{YT} comme suit :

$$v_{ST}(s_1, s_2, t_1, t_2) = \begin{cases} \frac{g_{ST}(s_1,s_2,t_1,t_2)}{C2} & \text{si } t_1 \in B_1 \text{ et } t_2 \in B_2 \\ 0 & \text{sinon.} \end{cases} \tag{B.31}$$

Donc

$$D2 = C2 \cdot E2 \tag{B.32}$$

où

$$E2 = \int_{B_1}\int_{B_2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{f_S(s_1, s_2)}{g_S(s_1, s_2)}\ v_{ST}(s_1, s_2, t_1, t_2)\ ds_1\ ds_2\ dt_1\ dt_2 \tag{B.33}$$

$$= E_v\left(\frac{f_S(s_1, s_2)}{g_S(s_1, s_2)}\right) \tag{B.34}$$

représente l'espérance $\frac{f_S(s_1,s_2)}{g_S(s_1,s_2)}$ dans le domaine B_1 et B_2.

Si N simulations Monte Carlo sont faites, c'est-à-dire que N circuits sont générés, N_p est le nombre de circuits qui passent le test et N_u est le nombre de circuits fonctionnels qui passent le test alors, le Taux de défauts D est estimé comme suit :

Estimation de $C1$:

$$C1 = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{(s_1\in A_1\ \&\ s_2\in A_2\ \&\ t_1\in B_1\ \&\ t_2\in B_2)} \tag{B.35}$$

$$= \frac{N_u}{N} \tag{B.36}$$

où $\mathbb{1}_{(s_1\in A_1\ \&\ s_2\in A_2\ \&\ t_1\in B_1\ \&\ t_2\in B_2)}$ représente la fonction indicatrice. Elle est égale à 1 si $s_1 \in A_1$, $s_2 \in A_2$, $t_1 \in B_1$ et $t_2 \in B_2$, et à 0 sinon.

Estimation de $E1$:

$$E1 = \frac{1}{N_u}\sum_{i=1}^{N_u} \frac{f_S(s_1^i, s_2^i)}{g_S(s_1^i, s_2^i)} \tag{B.37}$$

Estimation de $D1$:

$$\boxed{D1 = \frac{1}{N}\sum_{i=1}^{N_u} \frac{f_S(s_1^i, s_2^i)}{g_S(s_1^i, s_2^i)}} \tag{B.38}$$

Estimation de $C2$:

$$C2 = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{(t_1\in B_1\ \&\ t_2\in B_2)} \tag{B.39}$$

$$= \frac{N_p}{N} \tag{B.40}$$

Estimation de $E2$:

$$E2 = \frac{1}{N_p}\sum_{i=1}^{N_p} \frac{f_S(s_1^i, s_2^i)}{g_S(s_1^i, s_2^i)} \tag{B.41}$$

Estimation de $D2$:

$$\boxed{D2 = \frac{1}{N}\sum_{i=1}^{N_p}\frac{f_S(s_1^i, s_2^i)}{g_S(s_1^i, s_2^i)}} \tag{B.42}$$

Donc, D est calculé en remplaçant (B.38) et (B.42) dans (B.18).

Nous pouvons donc généraliser la formule du Taux de défauts D pour n performances et m critères de test. Ainsi,

$$\boxed{D = 1 - \frac{\sum_{i=1}^{N_u}\psi_{S_i}}{\sum_{i=1}^{N_p}\psi_{S_i}}} \tag{B.43}$$

où

$$\psi_{S_i} = \frac{f_S(s^i)}{g_S(s^i)} = \frac{f_S(s_1^i, s_2^i, \cdots, s_n^i)}{g_S(s_1^i, s_2^i, \cdots, s_n^i)} \tag{B.44}$$

Nous pouvons voir que si n'importe quel élargissement de la gaussienne d'origine est considérée, donc $f_S(s_1^i, s_2^i, \cdots, s_n^i) = g_S(s_1^i, s_2^i, \cdots, s_n^i)$. Par conséquent, $\psi_{S_i} = 1$ $(\forall i = 1, ..., n)$ et en le remplaçant dans (B.43) le Taux de défauts D est écrit comme :

$$D = 1 - \frac{N_u}{N_p} \tag{B.45}$$

qui est l'estimateur du Taux de défauts pour le cas nominal.

B.3.2 Couverture de rendement Y_C et Rendement de test Y_T

En utilisant la même procédure comme pour le calcul du Taux de défauts, la Couverture de rendement Y_C (Yield Coverage) et Rendement de test Y_T (Test Yield) sont données par les Equations (B.48) et (B.50) respectivement.

$$Y_C = \mathbf{P}(\text{circuit passe le test/il est fonctionnel}) \tag{B.46}$$

$$= \frac{\mathbf{P}(\text{circuit est fonctionnel et passe le test})}{\mathbf{P}(\text{circuit est fonctionnel})} \tag{B.47}$$

Ainsi,

$$\boxed{Y_C = \frac{\sum_{i=1}^{N_u}\psi_{S_i}}{\sum_{i=1}^{N_g}\psi_{S_i}}} \tag{B.48}$$

où N_g est le nombre de circuit fonctionnels.

$$Y_T = \mathbf{P}(\text{circuit passe le test}) \tag{B.49}$$

Ainsi,

$$\boxed{Y_T = \frac{1}{N}\sum_{i=1}^{N_p}\psi_{S_i}} \tag{B.50}$$

où ψ_{S_i} est donné par (B.44).

Bibliographie

[1] *Ocean Reference*. Cadence Design Systems, 2004.

[2] *Spectre User Guide*. Cadence Design Systems, 2004.

[3] *Virtuoso Schematic Composer SKILL Functions Reference*. Cadence Design Systems, 2004.

[4] I. S. 1057-1994. IEEE Standard for Digitizing Waveform Recorders. *IEEE Press*, December 1994.

[5] A. Abderrahman and B. Kaminska. Worst case tolerance analysis and CLP-Based multi-frequency test generation for analog circuits. *In IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18 :332–345, Mars 1999.

[6] E. Acar and S. Ozev. Defect-Based RF Testing Using a New Catastrophic Fault Model. *In IEEE International Test Conference (ITC'05)*, November 2005.

[7] H. Albustani. *Modelling Methods for Testability Analysis of analog Integrated Circuits Based on Pole-Zero Analysis*. PhD thesis, Der Fakultät Ingenieurwissenschaften der Universität Duisburg-Essen, 06 Août 2004.

[8] A. Ammari. *Analyse de sûreté des circuits complexes décrits en langage de haut niveau*. Thèse de doctorat, INPG (Institut National Polytechnique de Grenoble), 2006.

[9] K. Arabi and B. Kaminska. Parametric and Catastrophic Fault Coverage of Analog Circuits in Oscillation-Test Methodology. *In Proceedings of the 15th IEEE VLSI Test Symposium (VTS'97)*, pages 166–171, 1997.

[10] K. Arabi and B. Kaminska. Testing analog and mixed-signal integrated circuits using oscillation-test method. *In IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(7) :745–753, Juillet 1997.

[11] A. Asquini, J.-L. Carbonéro, and S. Mir. Test measurements evaluation for VCO and charge pump blocks in RF PLLs. *In SPIE International Symposium on Microtechnologies for the New Millenium, VLSI Circuits and Systems Conference*, Gran Canaria, Spain, May 2007, (to appear).

[12] A. Astran, T. Denley, and R. Häggkvist. *Bipartite graphs and their applications*. Cambridge university press, Chapter 10 : Coverings, 2004.

[13] J. S. Augusto and C.-F. B. Almeida. SpiceCache : an efficient time-domain fault simulator unsing circuit matrix caching. *In the 9th IEEE International Mixed-Signals Testing Workshop (IMSTW'03)*, pages 35–40, 2003.

[14] A. Balivada, J. Chen, and J.-A. Abraham. Analog testing with time response parameters. *In IEEE Design and Test of Computers*, 13(2) :18–25, Juin 1996.

[15] S. Battacharya, A. Halder, G. Srinivasan, and A. Chatterjee. Alternate testing of RF transceivers using optimized test stimulus for accurate prediction of system specifications. *In Journal of Electronic Testing : Theory and Applications (JETTA)*, 21 :323–339, 2005.

[16] N. Ben-Hamida and B. Kaminska. Analog circuit testing based on sensitivity computation. *In IEEE International Test Conferece (ITC'93)*, pages 652–661, Baltimore, October 1993.

[17] S. Biswas, P. Li, R.-D. Blanton, and L.-T. Pileggi. Specification test compaction for analog circuits and MEMS. *In Design, Automation and Test in Europe (DATE'05)*, pages 164–169, 2005.

[18] A. Bounceur. Réalisation d'un outil de CAO pour la validation des techniques d'autotest des circuits microélectroniques. Rapport de stage de DEA, Laboratoire TIMA, 2003.

[19] A. Bounceur, S. Mir, L. Rolíndez, and E. Simeu. *CAT platform for analogue and mixed-signal test evaluation and optimization.* Chapter in Research trends in VLSI and Systems on Chip, G. De Micheli, S. Mir, R. Reis (Eds.), Springer Science+Business Media, 2007, (à apparaître).

[20] A. Bounceur, S. Mir, L. Rolíndez, and E. Simeu. CAT platform for analogue and mixed-signal test evaluation and optimization. *In Proceedings of the 14th IFIP International Conference on Very Large Scale Integration (IFIP VLSI-SoC'06)*, pages 320–325, Nice, France, 2006.

[21] A. Bounceur, S. Mir, L. Rolíndez, and E. Simeu. A CAT platform for analogue and mixed-signal test evaluation and optimization. *In Digest of Papers of the 11th European Test Symposium (ETS'06)*, pages 217–222, Southampton, UK, 2006.

[22] A. Bounceur, S. Mir, and E. Simeu. Optimization of digitally coded test vectors for mixed-signal components. *In 19th Conference on Design of Circuits and Integrated Systems (DCIS'04)*, pages 895–900, Bordeaux, France, November 2004.

[23] A. Bounceur, S. Mir, and E. Simeu. Génération et optimisation de vecteurs de test pour des composants analogiques et mixtes. *In 7ème Journées Nationales du Réseau Doctoral de Microélectronique (JNRDM'04)*, pages 198–200, Marseille, France, 2004.

[24] A. Bounceur, S. Mir, E. Simeu, and L. Rolíndez. Estimation of test metrics for multiple analogue parametric devations. *In IEEE Desing and Test of Integrated Systems (DTIS'06)*, pages 234–239, 2006.

[25] A. Bounceur, S. Mir, E. Simeu, and L. Rolíndez. On the accurate estimation of test metrics for multiple analogue parametric devations. *In International Mixed-Signals Test Workshop (IMSTW'06)*, pages 19–26, 2006.

[26] M. Breuer and A. Friedman. *Diagnosis and reliable design of digital systems.* Computer Science Press, Woodland Hills, Calif., 1976.

[27] S.-P. Buchner, T.-J. Meehan, A.-B. Campbell, K.-A. Clark, and D. McMorrow. Characterization of Single-Event Upsets in a Flash Analog-to-Digital Converter (AD9058). *In IEEE transactions on Nuclear Science*, 47(6) :2358–2364, December 2000.

[28] M. Buehler and M. Sievers. Off-line build-in test techniques for VLSI circuits. *In Computer*, 15(6) :69–82, 1982.

[29] P. Caunegre and C. Abraham. Fault Simulation for Mixed-Signal Systems. *In Journal of Electronic Testing : Theory and Applications (JETTA)*, pages 143–152, 1996.

[30] P. Caunegre and C. Abraham. Achieving Simulation-Based Test Program Verification and Fault Simulation Capabilities for Mixed-Signal Systems. *In European Design and Test Conference*, Paris, France 1995.

[31] F. Corsi, S. Martino, and T. Williams. Defect Level As a Function of Fault Coverage and Yield. *In Electron. Lette.*, 29(8) :653–654, 1993.

[32] J. P. de Gyvez, G. Gronthoud, and R. Amine. VDD Ramp Testing for RF Circuits. *In Proceedings of the IEEE International Test Conference (ITC'03)*, pages 651–658, 2003.

[33] G. Devarayandug, P. Goteti, and S.-D. Huynh. Test set selection for structural faults in analog IC's. *In IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 18(7) :1026–1038, Juillet 1999.

[34] A. Dhayni. *Pseudo Random Built-in Self Test For Microsystems*. Ph.D. Thesis, TIMA Laboratory, 2006.

[35] A. Dhayni, S. Mir, and L. Rufer. MEMS Built-In-Self-Test Using MLS. *In Proceedings of the 9th IEEE European Test Symposium*, pages 66–71, Ajaccio, France, May 2004.

[36] A. Dhayni, S. Mir, and L. Rufer. Evaluation of Impulse Response-Based BIST Techniques for MEMS in the Presence of Weak Nonlinearities. *In Proceedings of the 10th IEEE European Test Symposium*, pages 82–87, Tallin, Estonia, May 2005.

[37] A. Dhayni, S. Mir, L. Rufer, and A. Bounceur. *On-chip Pseudorandom Testing for Linear and Nonlinear MEMS*. Chapter in IFIP VLSI-SoC, Springer, 2005, (à apparaître).

[38] A. Dhayni, S. Mir, L. Rufer, and A. Bounceur. Built-In Self-Test techniques for MEMS. *In Microelectronics Journal*, 2007, (sous évaluation).

[39] A. Dhayni, S. Mir, L. Rufer, and A. Bounceur. Nonlinearity Effects on MEMS On-chip Pseudorandom Testing. *In 11th IEEE International Mixed-Signals Testing Workshop*, pages 224–233, Cannes, France, June 2005.

[40] A. Dhayni, S. Mir, L. Rufer, and A. Bounceur. Pseudorandom Functional BIST for Linear and Nonlinear MEMS. *In Proceedings of Design, Automation and Test in Europe (DATE'06)*, pages 664–669, Munich, Germany, March 2006.

[41] A. Dhayni, S. Mir, L. Rufer, and A. Bounceur. Autotest Intégré des Microsystèmes Nonlinéaires. *In 8ème Journées Nationales du Réseau Doctoral de Microélectronique (JNRDM'05)*, pages 256–258, Paris, France, 2005.

[42] A. Dhayni, S. Mir, L. Rufer, and A. Bounceur. On-chip Pseudorandom Testing for Linear and Nonlinear MEMS. *In Proceedings of the IFIP International Conference on Very Large Scale Integration (IFIP VLSI-SoC'05)*, pages 435–440, Perth, Western Australia, October 2005.

[43] S. W. Director and R. A. Rohrer. The generalize adjoint network and network sensitivities. *In IEEE Trans. Circuit Theory*, CT-16 :318–323, 1969.

[44] B. Dufort and G.-W. Roberts. On-chip analog signal generation for mixed-signal Built-In Self Test. *In Journal of Solid-State Circuits*, 34(3) :318–330, Mars 1999.

[45] B. Dufort and G.-W. Roberts. Signal generation using periodic and multi-bit sigma-delta modulated streams. *In Proceedings IEEE International Test Conference*, pages 396–405, Washington D.C., Novembre 1997.

[46] P. Duhamel and J.-C. Rault. Automatic test generation techniques for Analog Circuits and Systems : A Review. *In IEEE transactions on Circuits and Systems*, 26(7) :411–440, Juillet 1979.

[47] Y. Eben-Aimine, A. Richardson, C. Descleves, and K. Sommacal. GDS FaultSim, a Mixed-Signal IC Computer-Aided-Test (CAT) Tool. *In IEEE Design and Test Conference in Europe*, pages 232–238, Munich, Germany, March 1999.

[48] R. Ecoffet, S. Duzellier, P. Tastet, C. Aicardi, and M. Labrunee. Observation of Heavy Ion Induced Transients in Linear Circuits. *In IEEE Radiation Effects Data Workshop*, pages 72–77, December 1994.

[49] N. Engin. *Linking Mixed-Signal Design and Test : Generation and Evaluation of Specification-Based Tests*. University of Twente in Enschede, the Netherland, 2000.

[50] N. Engin and H. G. Kerkhoff. Fast Fault Simulation for Nonlinear Analog Circuits. *In IEEE Design and Test of Computers*, pages 40–47, 2003.

[51] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, 1989.

[52] A. Halder, S. Battacharya, and A. Chatterjee. Automatic multitone alternate test generation for RF circuits using behavioural models. *In International Test Conference (ITC'03)*, pages 665–673, 2003.

[53] A. Halder, S. Battacharya, and A. Chatterjee. Alternate test generation for specification test of RF circuits. *In International Mixed-Signal Test Workshop (IMSTW'03)*, pages 7–12, 25-27 Juin 2003.

[54] A. Han, A. Halder, and A. Chatterjee. Test elimination using redundancy analysis for specification test of analog circuits. *In International Mixed-Signal Test Workshop (IMSTW'04)*, pages 69–75, 23-25 Juin 2004.

[55] R. Harvey, A. Richardson, and H. Kerhoff. Defect Oriented Test Developement Based on Layout inductive Fault Analysis. *In IEEE International Mixed-Signal Testing Workshop (IMSTW'05)*, pages 2–9, 1995.

[56] Y. Hervé. *VHDL-AMS : Applications et enjeux industriels*. Dunod-Universités - collection : Sciences-sup, préface d'Alain Vachoux.

[57] J. Hou and A. Chatterjee. Concurrent Transient Fault Simulation for analog Circuits. *In Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(10) :1385–1398, 2003.

[58] J. Huertas. Test and Design for Testability of Analog and Mixed-Signal Integrated Circuits : Theoretical Basis and Pragmatical Approaches. In *In Circuit Theory and Design : Selected Topics in Circuits and Systems*. D.G. Haigh, J.L. Huertas, P.A. Humblet, M. Kunt, Elsevier Science Publishers, 1993.

[59] S. Huss and R. Gyurcsik. Optimal ordering of analog integrated circuit tests to minimize test time. *In 28th ACM/IEEE Design Automation Conference*, pages 494–499, 17-21 Juin 1991.

[60] S. Huynth, S. Kim, M. Soma, and J. Zhang. Automatic analog signal generation using multifrequency analysis. *In IEEE Transactions on Circuits and Systems*, 46(5) :565–576, Mai 1999.

[61] B. Johnson. *Desin and Analysis of Fault Tolerant Digital Systems*. Addison-Wesley Publ., 1989.

[62] J.-M. Jolion. *Probabilités et Statistique*. http ://rfv.insa-lyon.fr/ jolion/STAT/, 2006.

[63] A. Khouas. *Simulation de Fautes et Optimisation des Tests de Production pour les Circuits Analogiques avec Prise en Compte des Tolérances*. Thèse de doctorat, Université de Paris VI, 2000.

[64] A. Khouas and A. Derieux. FDP : Fault Detection Probability Function for Analog Circuits. *In the IEEE International Symposium on Circuits and Systems (ISCAS 2001)*, 4 :17–20, 2001.

[65] A. Khouas, M. Dessouky, and A. Derieux. Optimized Statistical Analog Fault Simulation. *In 8th Asian Test Symposium (ATS'99)*, pages 227–232, 1999.

[66] Y. Kiliç and M. Zwolinski. Speed-up Techniques for Fault-based Analogue Fault Simulation. *In proceedings of the European Test Workshop (ETW'01)*, pages 27–29, 2001.

[67] Y. Kiliç and M. Zwolinski. Behavioural Fault Modelling using VHDL-AMS and Slow Transient Analysis with hAMSter Simulator to Speed-up Analogue Fault Simulation. *In proceedings of Analog Integrated Circuits and Signal*, 39(2) :177–190, 2004.

[68] K. Label, P. Marshall, C. Marshall, M. D'Ordine, M. Carts, G. Lum, H. Kim, C. Seidlick, T. Powell, R. Abbot, J. Barth, and E. Stasinopoulos. Proton-Induced Transients in Optocouplers : In-Flight Anomalies, Ground Irradiation Test, Mitigation and Implications. *In IEEE transactions on Nuclear Science*, 44(6) :1885–1892, December 1997.

[69] W.-M. Lindermeir, H.-E. Graeb, and K.-J. Antreich. Analog testing by characteristic observation inference. *In IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(9) :1353–1368, Septembre 1999.

[70] V. B. Litovski, I. V. Litovski, and M. Zwolinski. Concurrent analogue fault simulation, the equation formulation aspect. *In International Journal of Circuit Theory and Applications*, (32) :487–507, 2006.

[71] L. Lizarraga, S. Mir, G. Sicard, and A. Bounceur. Study of a BIST Technique for CMOS Active Pixel Sensors. *In Proceedings of the 14th IFIP International Conference on Very Large Scale Integration (IFIP VLSI-SoC'06)*, pages 326–331, Nice, France, 2006.

[72] M. Mahoney. DSP-based testing of analog and mixed-signal circuits. *In IEEE Computer Society Press*, Washington DC, 1987.

[73] C. Maunder. Standard Test Access Port and Boundary-Scan Architecture. *In IEEE Std 1149.1-1193a, (Maunder. (Ed)), IEEE Standards Board, New York)*, 1993.

[74] L. Milor. *Fault-driven analog testing.* Ph.d. dissertation, University of California, Berkeley, Calif., 1992.

[75] L. Milor and A.-L. Sangiovanni-Vincentelli. Minimizing production test time to detect faults in analog circuits. *In IEEE Transactions On Computer Aided Design of Integrated Circuits and Systems*, 13(6) :796–813, Juin 1994.

[76] L. Milor and A.-L. Sangiovanni-Vincentelli. Optimal Test Set Design for Analog Circuits. *Proc 1990 IEEE International Conference on Computer-Aided Design*, pages 294–297, Novembre 1990.

[77] L. Milor and V. Visvanathan. Detection of Catastrophic Faults in Analog Integrated Circuits. *In IEEE transactions on Computer Aided Design*, 8(2) :114–130, Février 1989.

[78] L.-S. Milor. A tutorial introduction to research on analog and mixed-signal circuit testing. *In IEEE transactions on Circuits and Systems-II : Analog and Digital Signal Processing*, 45(10) :1389–1407, October 1998.

[79] S. Mir. Integrated circuit testing : From microelectronics to microsystems. *In 5th IFAC Symposium on fault detection, supervision and safety of technical processes (Safeprocess), Invited Talk*, pages 13–24, Washington D.C. USA.

[80] S. Mir, C. Diedrich, C. Roman, and C. Domingues. On-chip test signal generation for acoustic and ultrasound microelectronic interfaces. *In 8th IEEE International Mixed-Signal Testing Workshop*, pages 137–144, Montreux, Switzerland, June 2002.

[81] S. Mir, M. Lubaszewski, and B. Courtois. Fault-based ATPG for linear analogue circuits with minimal size multifrequency test sets. *In Journal of Electronic Testing : Theory and Applications (JETTA)*, 9 :43–57, August/October 1996.

[82] S. Mir, A. Rueda, T. Olbrich, E. Peralías, and J. Huertas. SWITTEST : automatic switch-level fault simulation and test evaluation of switched-capacitor systems. *In Proceedings of the 34th annual conference on Design automation conference*, pages 281–286, Anaheim, California, United States, June 1997.

[83] N. Nagi, A. Chattergee, A. Balivada, and J.-A. Abraham. Fault-Based automatic test generator for linear analog circuits. *In International Conference on Computer Aided Design*, pages 88–91, 1993.

[84] N. Nagi, A. Chattergee, A. Balivada, and J.-A. Abraham. Efficient multisine testing of analog circuits. *In IEEE International Conference on VLSI Design*, pages 234–238, Janvier 1995.

[85] N. Nagi, A. Chatterjee, and J. A. Abraham. DRAFTS : Discretized Analog Circuit Fault Simulator. *In 30th ACM/IEEE Design Automation Conference*, pages 509–514, 1993.

[86] R.-D. Nowak and B.-D. Van-Veen. Evaluation of Impulse Response-Based BIST Techniques for MEMS in the presence of Weak Nonlinearities. *In IEEE Proceedings of European Test Conference*, pages 82–87, Tallinn, Estonia, May, 2005.

[87] C. Pan and K. Cheng. Test generation for linear, time-invariant analog circuits. *In IEEE Transactions on Circuits and Systems*, 46(5) :554–564, Mai 1999.

[88] C.-Y. Pan and K.-T. Cheng. Fault Macromodeling for Analog/Mixed-signal Circuits. *In proceedings IEEE International Test Conference*, pages 913–922, Washington, DC, USA, November 1997.

[89] A. J. Perkins, M. Zwolinski, C. D. Chalk, and B. R. Wlkins. Fault Modelling and Simulation Using VHDL-AMS. *In Analog Integrated Circuits and Signal Processing*, 16 :53–67, 1998.

[90] G. Prenat. *Conception d'une architecture de BIST analogique et mixte programmable en technologie CMOS très submicronique.* Thèse de doctorat, INPG (Institut National Polytechnique de Grenoble), 2006.

[91] M.-J. Reed and M.-O.-J. Hawksford. Identification of descrete Volterra series using maximum length sequences. *In IEEE Proceeding of Circuits, Devices and Systems*, 143(5) :241–248, October, 1996.

[92] M. Renovell, G. Campon, and D. Auvergne. FSPICE : a tool for fault modeling in MOS circuits. *In Integration, VLSI Jounrnal*, 3(3) :245–255, 1985.

[93] L. Rolíndez, S. Mir, A. Bounceur, and J. Carbonéro. A SNDR BIST for $\Sigma\Delta$ Analogue-to-Digital Converters. *In 24th IEEE VLSI Test Symposium*, pages 314–319, Amissville, USA, Apr 30th-May 4th, 2006.

[94] L. Rolíndez, S. Mir, A. Bounceur, and J.-L. Carbonéro. A Sine-Wave Fitting based BIST for high resolution analogue-to-digital converters. *In Journal of Electronic Testing : Theory and Applications (JETTA)*, 2006.

[95] L. Rolíndez, S. Mir, and J.-L. Carbonéro. Design of a 96-dB audio $\Sigma\Delta$ ADC including a BIST technique for SNDR testing. *In 21st Conference on Design of Circuits and Integrated Systems*, Barcelona, Spain, November, 2006.

[96] L. Rolíndez, S. Mir, G. Prenat, and A. Bounceur. A 0.18 μm CMOS implementation of on-chip analogue test signal generation from digital test patterns. *In IEEE Design and Test Conference in Europe, Paris, France*, pages 704–705, February 2004.

[97] C. Roman. Building CAT tools, for System-on-Chip. Rapport de stage de fin d'études, Laboratoire TIMA, 2002.

[98] C. Roman, S. Mir, and B. Charlot. Building an analogue fault simulation tool and its application to MEMS. *In Microelectronics Journal*, 34(10) :897–906, 2003.

[99] L. Rufer, S. Mir, E. Simeu, and C. Domingues. On-chip pseudorandom MEMS testing. *In Journal of Electronic Testing : Theory and Applications*, 21 :233–241, 2005.

[100] K. Saab, N. Ben-Hamida, and B. Kaminska. Parametric Fault Simulation and Test Vector Generation. *In IEEE Design and Test Conference in Europe*, pages 650–656, 2000.

[101] K. Saab, N. B. Hamida, and B. Kaminska. Closing the Gap Between Analog and Digital Testing. *In the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(2) :307–314, 2001.

[102] M.-B. Santos and J.-P. Teixeira. Defect-Oriented Mixed-Level Fault Simulation of Digital Systems-on-a-Chip Using HDL. *In Proceedings of Design Automation and Test in Europe (DATE'99)*, pages 549–554, 1999.

[103] C.-E. Savioli, C.-C. Szendrodi, J.-V. Calvano, and A. Filho. On the use of genetic algorithms for fault diagnosis on continuous-time analog filters. *In IEEE European Test Symposium (ETS'04)*, pages 161–163, 23-26 Mai 2004.

[104] C. Sebeke and M. Ohletz. Analogue fault simulation. *Poster on CAVE Workshop*, May 1993.

[105] C. Sebeke and M.-J. Ohletz. Simulation Models for the Fault Simulation on Single Elements of Analogue Circuits. *ESPRIT III project 7107, Report*, September 1993.

[106] C. Sebeke, J.-P. Teixeira, and M.-J. Ohletz. Automatic Fault Extraction and Simulation of Layout Realistic Faults for Integrated Analogue Circuits. *In Proceedings of the European Design and Test Conference*, page 464, 1995.

[107] C.-J. R. Shi, M. W. Tian, and G. Shi. Efficient DC Fault Simulation of Nonlinear Analog Circuits : One-Step Relaxation and Adaptive Simulation Continuation. *In IEEE trans. on Computer-Aided Desing of Integrated Circuits and Systems*, 25(7) :1392–1400, 2006.

[108] M. Singh and I. Koren. Fault Sensivity Analysis and Reliability Enchancement of Analog-to-Digital Converters. *In IEEE transactions on Very Large Scale Integration (VLSI) Systems*, 11 :839–852, Issue 5 October 2003.

[109] M. Singh and I. Koren. Reliability Enhancement of analog-to-digital converters (ADCs). *In IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pages 347–353, San Francisco, California, USA, October 2001.

[110] M. Slamani and K. Arabi. Reducing test time in the high-volume production of analog circuits using efficient test-vector generation and interpolation techniques. *In Journal of Electronic Testing : Theory and Applications (JETTA)*, 17 :417–425, 2001.

[111] M. Slamani and B. Kaminska. An integrated approach for analog circuit testing with a minimum number of detected parameters. *In International Test Conference (ITC'94)*, pages 631–640, 1994.

[112] S. Spinks, C. Chalk, I. Bell, and M. Zwolinski. Generation and Verification of Tests for Analog Circuits Subject to Process Parameter Deviations. *In Journal of Electronic Testing : Theory and Applications (JETTA)*, 20(1) :11–23, February 2004.

[113] G. Srinivasan, A. Halder, S. Battacharya, S. Goyal, and A. Chatterjee. Loopback test of RF transceivers using periodic bit sequences : An alternate test approach. *In International Mixed-Signal Test Workshop (IMSTW'04)*, pages 82–87, 23-25 Juin 2004.

[114] H.-G. D. Stratigopoulos and Y. Makris. Constructive Derivation of Analog Specification Test Criteria. *In Proceedings of the 23rd IEEE VLSI Test Symposium (VTS'05)*, pages 395– 400, 2005.

[115] H.-G. D. Stratigopoulos and Y. Makris. Nonlinear Decision Boundaries for Testing Analog Circuits. *In IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(11) :1760–1773, November 2005.

[116] S. Sunter and N. Nagi. Test metrics for analog parametric faults. *In 17th IEEE VLSI Test Symposium*, pages 226–234, 1999.

[117] J. Teixeira, I. Teixeira, C. Almeida, F. Goncalves, and J. Level. Methodology for testability Enhancement at Layout Level. *In Journal of Electronic Testing : Theory and Applications (JETTA)*, 1(4) :289–297, 1991.

[118] J. Teixeira de Sousa, F. M. Gonçalves, L. P. Teixeira, C. Marzocca, and T.-W. Williams. Defect Level Evaluation in an IC Design Environment. *In IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems*, 15(10) :1286–1293, 1996.

[119] M. W. Tian and C.-J. R. Shi. Rapid Frequency-Domain Analog Fault Simulation Under Parameter Tolerances. *In Proceedings of the 34th Design Automation Conference*, pages 275–280, 1997.

[120] J. Tongbong. Conception et test de circuits RF BiCMOS. Rapport de stage de DEA, Laboratoire TIMA et STMicroelectronics, 2005.

[121] J. Tongbong, A. Bounceur, S. Mir, and J.-L. Carbonéro. Evaluation of Test Measures for Low-Cost LNA Production Testing. *In Digest of papers of the 14th IFIP International Conference on Very Large Scale Integration (IFIP VLSI-SoC'06)*, pages 48–52, Nice, France, 2006.

[122] J. Tongbong, S. Mir, and J.-L. Carbonéro. Evaluation of test measures for LNA production testing using a multinormal statistical model. *In IEEE Design and Test in Europe Conference, Interactive presentation*, Nice, France, 2007.

[123] S.-J. Tsai. Test vector generation for linear analog devices. *In Proceedings IEEE International Test Conference*, pages 592–597, 1991.

[124] W.-D. Tseng and K. Wang. Fault Coverage Estimation Model for Partially Testable Multichip Modules. *In Proceedings of the Pacific Rim International Symposium on Fault-Tolerant Systems*, pages 72–77, 1997.

[125] W.-D. Tseng and K. Wang. Fault coverage and defect level estimation models for partially testable MCMs. *In IEE Proceedings of Circuits, Devices and Systems*, 147(2) :119–124, 2000.

[126] P. Variyam and A. Chatterjee. Test Generation for Comprehensive Testing of Linear Analog Circuits Using Transient Response Sampling. *In International Conference on Computer-Aided Design (ICCAD '97)*, pages 382–385, 9-13 Novembre 1997.

[127] P. N. Variyam. FLYER : Fast Fault Simulation of Linear Analog Circuits Using PoLYnomial Waveform and PERturbed State Representation. *In 10th International Conference on VLSI Design*, 15(10) :408–412, January 1997.

[128] T. Veiga, C. F. B. Almeida, and J. S. Augusto. Efficient Analog Fault Simulation in AC Circuits. *In Informal Digest of Papers of the IEEE European Test Symposium (ETS'05)*, pages 51–56, 2005.

[129] T. Williams and N. Brown. Defect Level as a Function of Fault Coverage. *In IEEE Transations on Computers*, C-30(12) :987–988, 1981.

[130] P. Wilson, Y. Kilic, J. Ross, M. Zwolinski, and A. Brown. Behabioural modeling of operational ampifier faults using VHDL-AMS. *Design, Automation and Test in Europe Conference (DATE'02)*, page 1133, March 4-8 2002.

[131] Z. R. Yang and M. Zwolinski. Fast, robust dc and transient fault simulation for nonlinear analogue circuits. *In proceedings of Design Automation and Test in Europe (DATE'99)*, pages 244–248, 1999.

[132] A. Zjajo and J. P. de Gyvez. Evaluation of Signature-Based Testing of RF/Analog Circuits. *In European Test Symposium*, pages 62–67, Tallin, Estonia, May 2005.

[133] A. Zjajo, J. P. de Gyvez, and G. Gronthoud. A quasi-static approach detection and simulation of parametric faults in analog and mixed-signal circuits. *In 11th IEEE International Mixed-Signals Testing Workshop*, pages 155–164, Cannes, France, June 2005.

[134] M. Zwolinski, A. D. Brown, and C. D. Chalk. Concurrent analogue fault simulation. *In proceedings of International Mixed-Signal Testing Workshop (IMSTW'97)*, pages 42–47, 1997.

Liste des publications de l'auteur

Chapitres de livre

- [1] **A. Bounceur**, S. Mir, L. Rolíndez and E. Simeu. *CAT platform for analogue and mixed-signal test evaluation and optimization.* Chapter in Research trends in VLSI and Systems on Chip, G. De Micheli, S. Mir, R. Reis (Eds.), Springer Science+Business Media (à apparaître).
- [2] [Cl.2] A. Dhayni, S. Mir, L. Rufer and **A. Bounceur**. *On-chip pseudorandom testing for linear and non-linear MEMS.* Chapter in Springer selection of best papers from VLSI-SoC'05 (à apparaître).

Revues internationales

- [3] **A. Bounceur**, S. Mir, E. Simeu and L. Rolíndez. Estimation of test metrics for the optimisation of analogue circuit testing. *In Journal of Electronic Testing : Theory and Applications (JETTA)*, (à apparaître).
- [4] A. Dhayni, S. Mir, L. Rufer, and **A. Bounceur**. Pseudorandom BIST for Test and Characterization of Linear and Nonlinear MEMS. *In Microelectronics Journal*, (sous évaluation).
- [5] L. Rolíndez, S. Mir, **A. Bounceur** and J.-L. Carbonéro. L. Rolíndez, S. Mir, **A. Bounceur** and J.L. Carbonero. A BIST Scheme for SNDR Testing of ADCs Using Sine-Wave Fitting. *Journal of Electronic Testing : Theory and Applications*, Springer Science+Business Media, 22(4-6), 2006, pp. 325-335.

Conférences internationales et Workshops avec comité de lecture

- [6] A. Dhayni, S. Mir, L. Rufer, and **A. Bounceur**. Characterization and testing of MEMS nonlinearities. *In International Design and Test Workshop (IDT'06)*, 2006.
- [7] **A. Bounceur**, S. Mir, L. Rolíndez and E. Simeu. CAT platform for analogue and mixed-signal test evaluation and optimization, *In 14th IFIP International Conference on Very Large Scale Integration VLSI-SoC*, Nice, France, October 2006, pp. 320-325.
- [8] L. Lizarraga, S.Mir, G. Sicard and **A. Bounceur**. Study of a BIST Technique for CMOS Active Pixel Sensors, *In 14th IFIP International Conference on Very Large Scale Integration VLSI-SoC*, Nice, France, October 2006, pp. 326-331.
- [9] J. Tongbong, **A. Bounceur**, S. Mir and J.-L. Carbonéro. Evaluation of test measures for low-cost LNA production testing. *In Ph.D. forum at 14th IFIP International Conference on Very Large Scale Integration VLSI-SoC'06*, Nice, France, 2006, pp. 48-52.
- [10] **A. Bounceur**, S. Mir, E. Simeu and L. Rolíndez. Estimation of test metrics for multiple analogue parametric deviations, *In IEEE Desing and Test of Integrated Systems DTIS'06*, Tunis, Tunisia, 2006, pp. 234-239.
- [11] **A. Bounceur**, S. Mir, E. Simeu and L. Rolíndez. On the accurate estimation of test metrics for multiple analogue parametric deviations. *In 12th International Mixed-Signals*

Testing Workshop IMSTW'06, Edinburgh, Southampton, UK, June 21-23, 2006, pp. 19-26.

- [12] **A. Bounceur**, S. Mir, L. Rolíndez and E. Simeu. A CAT platform for analogue and mixed-signal test evaluation and optimization. *In Informal Digest of the 11th IEEE European Test Symposium*, Southampton, UK, May 21-25, 2006, pp. 217-222.
- [13] L. Rolíndez, S. Mir, **A. Bounceur** and J.-L. Carbonéro. A Signal to Noise Ratio BIST for $\Sigma\Delta$ Analogue-to-Digital Converters. *In 24th IEEE VLSI Test Symposium*. Berkeley, California, USA, April-May 2006, pp. 314-319.
- [14] A. Dhayni, S. Mir, L. Rufer and **A. Bounceur**. Pseudorandom functional BIST for Linear and Nonlinear MEMS. *In IEEE Design, and Test Conference in Europe (DATE'06)*, Germany, 2006, pp. 664-669.
- [15] A. Dhayni, S. Mir, L. Rufer and **A. Bounceur**. On-chip Pseudorandom Testing for Linear and Nonlinear MEMS, *In 13th IFIP International Conference on Very Large Scale Integration (VLSI-SoC)*, Perth, Western Australia, October 2005, pp. 435-440.
- [16] L. Rolíndez, S. Mir, **A. Bounceur** and J.-L. Carbonero. A Digital BIST for a 16-bit audio Sigma Delta Analogue-to-Digital Converter, *In International Mixed-Signals Testing Workshop IMSTW'05*, Cannes, France, Juin 2005, pp. 45-52.
- [17] A. Dhayni, S. Mir, L. Rufer and **A. Bounceur**. Nonlinearity Effects on MEMS On-chip Pseudorandom Testing, *In International Mixed-Signals Testing Workshop IMSTW'05*, Cannes, France, Juin 2005, pp. 224-233.
- [18] **A. Bounceur**, S. Mir and E. Simeu. Optimisation of digitally coded test vectors for mixed-signal components. *In 19th Conference on Design of Circuits and Integrated Systems (DCIS'04)*, Bordeaux, France, Novembre 2004, pp. 895-900.
- [19] L. Rolíndez, S. Mir, G. Prenat and **A. Bounceur**. A 0.18 μm CMOS Implementation of On-chip Analogue Test Signal Generation from Digital Test Patterns. *In IEEE Design and Test in Europe Conference*, Interactive presentation, Paris, February 2004, pp. 704-705.

Conférences nationales

- [20] N. Akkouche, **A. Bounceur** et S. Mir. Réduction de tests fonctionnels par modélisation statistique des circuits analogiques. *In 10ème Journées Nationales du Réseau Doctoral de Microélectronique*, Lille, France, May 2007.
- [21] A. Dhayni, S. Mir, L. Rufer et **A. Bounceur**. BIST pour les microsystèmes nonlinéaires. *In 9ème Journées Nationales du Réseau Doctoral de Microélectronique*, Rennes, France, May 2006.
- [22] **A. Bounceur**, A. Dhayni, S. Mir et L. Rufer. Génération de vecteurs de test pour des MEMS purement nonlinéaires pour le calcul des noyaux de Volterra. *In 8ème Journées Nationales du Réseau Doctoral de Microélectronique*, Paris, France, May 2005, pp. 340-342.
- [23] A. Dhayni, S. Mir, L. Rufer et **A. Bounceur**. Autotest Intégré des Microsystèmes Nonlinéaires. *In 8ème Journées Nationales du Réseau Doctoral de Microélectronique*, Paris, France, May 2005, pp. 256-258.
- [24] **A. Bounceur**, S. Mir and E. Simeu, Génération et optimisation de vecteurs de test pour des composants analogiques et mixtes. *In 7ème Journées Nationales du Réseau Doctoral de Microélectronique*, Marseille, France, May 2004, pp. 198-200.

Présentations invitées

- [25] **A. Bounceur** et S. Mir. Application des méthodes et des outils de la Recherche Opérationnelle à la Micro-électronique. *Présentation invitée, Séminaire Mathématique de Béjaia*, Université de Béjaia, Béjaia, Algérie, (à présenter).

CAT platform for mixed-signal circuit testing

Abstract : The growing complexity of modern chips poses challenging test problems due to the requirement for specialized test equipment and the involved lengthy test times. This is particularly true for heterogeneous chips that comprise digital, analogue, and RF blocks onto the same substrate. Many research efforts are currently under way in the mixed-signal test domain. Theses efforts concern optimization of tests at the production stage (e.g. off-line) or during the lifetime of the chip (on-line test). A promising research direction is the integration of additional circuitry on-chip, aiming to facilitate the test application (Design For Test) and/or to perform Built-In-Self-Test. The efficiency of such test techniques, both in terms of test accuracy and test cost, must be assessed during the design stage. However, there is an alarming lack of CAT tools, which are necessary, in order to facilitate the study of these techniques and, thereby, expedite their transfer into a production setting. In this thesis, we develop a CAT platform that can be used for the validation of analogue test techniques. The platform includes tools for fault modeling, injection and simulation, as well as tools for analogue test vector generation and optimization. A new statistical method is proposed and integrated into the platform, in order to assess the quality of test techniques during the design stage. This method aims to set the limits of the considered test criteria. Then, the different test metrics (as Fault coverage, Defect level or Yield loss) are evaluated under the presence of parametric and catastrophic faults. Some specific tests can be added to improve the structural fault coverage. The CAT platform is integrated in the Cadence design framework environment.
Key words : Analogue test, CAT tools, analogue fault simulation, test evaluation, multiple parametric deviations, multifrequency test vectors, statistical modelling.

Plateforme CAO pour le test de circuits mixtes

Résumé : La complexité croissante des puces microélectroniques pose de très importants problèmes de test, avec des coûts en forte augmentation dus principalement à l'utilisation d'équipements de test très sophistiqués et à des temps de test trop long. Ceci est particulièrement vrai dans le cas des puces mixtes, intégrant simultanément des parties numériques ainsi que des parties analogiques, mixtes ou RF. De nombreuses recherches sont en cours dans le domaine du test de circuits mixtes. Ces recherches concernent des techniques permettant l'optimisation du test lors de la production ou lors de l'utilisation des puces dans leur application finale (test en ligne ou hors ligne). Certaines de ces techniques permettent d'ajouter des circuits additionnels dans la puce pour faciliter le test (conception en vue du test) et même réaliser un auto-test. Cependant, elles doivent être évaluées lors de la conception afin d'estimer la qualité des tests proposés et évaluer les avantages économiques obtenus. Ceci nécessite l'utilisation d'outils de CAO orientés au test (CAT) qui se font rares et généralement non commercialisés en raison de leur nature académique, ce qui limite leur application, ainsi, leur utilisation. Dans le cadre de cette thèse, nous avons développé une plateforme de CAT permettant de valider les techniques de test analogique, incluant des outils de modélisation, d'injection et de simulation de fautes ainsi que des outils de génération et d'optimisation de vecteurs de test analogiques. Une nouvelle méthode statistique a été proposée afin d'évaluer la qualité d'une technique de test lors de la phase design. Cette technique permet de fixer les limites des critères de test considérés. Ensuite, les différentes métriques de test (telles que la Couverture de fautes, le Taux de défauts ou la Perte de Rendement) sont évaluées sous la présence de fautes paramétriques ou catastrophiques. Des tests spécifiques à la détection de fautes peuvent être ajoutés pour augmenter la Couverture de fautes. Cette plateforme de CAT est intégrée dans l'environnement de conception microélectronique Cadence.
Mots clés : Test analogique, outils CAT, simulation de fautes analogique, évaluation de test, déviations paramétriques multiples, vecteurs de test multi-fréquences, modélisation statistique.

Thèse préparée au laboratoire TIMA (Techniques de l'Informatique et de la Microélectronique pour l'Architecture des ordinateurs), INPG, 46 avenue Félix Viallet, 38031, Grenoble Cedex 1, France.

ISBN 978-2-8481-3097-2