

## TABLE DES MATIÈRES

	Page
INTRODUCTION .....	1
CHAPITRE 1 ÉTAT DE L'ART .....	9
1.1 Introduction.....	9
1.2 Notions de base.....	10
1.2.1 Environnements radiatifs .....	10
1.2.2 Les effets des radiations sur les circuits électroniques .....	15
1.2.3 Évaluation de la sensibilité des circuits intégrés à l'égard des radiations .....	19
1.2.4 Architecture des FPGA à base de SRAM.....	31
1.2.5 Configuration du FPGA Virtex-5 .....	37
1.3 Mise en contexte.....	38
1.3.1 Techniques d'injection des pannes .....	38
1.3.2 Optimisation de la procédure d'injection des pannes .....	50
1.3.3 Injection des pannes dans les LUT .....	53
1.4 Conclusion .....	56
CHAPITRE 2 ENVIRONNEMENT D'ÉMULATION.....	57
2.1 Introduction.....	57
2.2 Outil d'injection des pannes : <i>SEU Controller</i> .....	58
2.3 Procédure de protection du <i>SEU Controller</i> .....	64
2.3.1 Détermination des bits de configuration du <i>SEU Controller</i> .....	66
2.3.2 Détermination des bits essentiels du <i>SEU Controller</i> .....	67
2.4 Aperçu général des montages de test.....	69
2.5 Conclusion .....	72
CHAPITRE 3 PROCÉDURE EFFICACE D'ÉMULATION DES EFFETS DES SEU DANS LES FPGA À BASE DE SRAM.....	75
3.1 Introduction.....	75
3.2 Différence de sensibilité relative des bits de configuration à 1 et ceux à 0 .....	76
3.3 Procédure automatisée d'injection des pannes .....	80
3.4 Procédure d'injection des pannes basée sur la différence de sensibilité relative : Approche de génération des séquences de test considérant la sensibilité relative.....	85
3.5 Résultats de validation .....	87
3.5.1 Résultats de validation pour le design des RO implémentés dans les CLB du FPGA.....	87
3.5.2 Résultats de validation pour le design des RO implémentés dans les IOB du FPGA .....	92
3.5.3 Résultats de validation pour un design plus conventionnel .....	94
3.5.4 Discussion .....	95
3.6 Conclusion .....	97

CHAPITRE 4	OPTIMISATION D'ÉMULATION DES SEU DANS LES FPGA À BASE DE MÉMOIRE SRAM.....	99
4.1	Introduction.....	99
4.2	Méthodologie d'injection des pannes proposée.....	100
4.3	Évaluation de la sensibilité des différents ensembles des bits de configuration.....	101
4.4	Détermination de l'erreur d'estimation du nombre des bits critiques (CBEE).....	103
4.5	Résultats en fonction du critère d'optimisation .....	105
4.5.1	Optimisation du NFCB .....	105
4.5.2	Amélioration de la valeur de CBEE (en comparaison avec l'injection aléatoire) .....	106
4.5.3	Optimisation du CBEE .....	107
4.5.4	Gain obtenu par la connaissance du type des ressources .....	107
4.6	Conclusion .....	108
CHAPITRE 5	MÉTHODOLOGIE D'INJECTION DES PANNES AUTOMATISÉE POUR L'ÉVALUATION DE LA ROBUSTESSE DES LUT À L'ÉGARD DES SEU DANS LES FPGA À BASE DE SRAM.....	111
5.1	Introduction.....	111
5.2	Mise en contexte .....	112
5.3	Méthodes d'identification des bits de configuration des LUT.....	113
5.3.1	Identification de la totalité des bits de configuration des bits de LUT ...	113
5.3.2	Identification des bits de configuration des bits de LUT utilisés.....	116
5.4	Procédure automatisée d'injection exhaustive des pannes ciblant les LUT utilisées .....	118
5.5	Résultats de validation .....	120
5.6	Conclusion .....	121
CONCLUSION.....		123
RECOMMANDATIONS .....		127
ANNEXE I	FICHIERS EBD ET EBC .....	129
ANNEXE II	INTERFACE D'INJECTION DES PANNES.....	133
ANNEXE III	FICHIERS GÉNÉRÉS PAR L'INTERFACE LABVIEW.....	135
LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES.....		137

## LISTE DES TABLEAUX

	Page
Tableau 1.1	Différents types d'événements singuliers (SEE) non destructifs.....19
Tableau 1.2	Sites adoptés pour l'expérience Rosetta .....22
Tableau 2.1	Description des commandes à envoyer au <i>SEU Controller</i> via le mode <i>UART Control</i> .....61
Tableau 3.1	Résultats de la campagne d'injection des pannes ciblant les bits à '1' (CB1).....77
Tableau 3.2	Résultats de la campagne d'injection des pannes ciblant les bits à '0' (CB0).....77
Tableau 3.3	Résultats de validation des tests pour le design des RO implémentés dans les CLB.....88
Tableau 3.4	Résultats de validation statistique de l'approche d'injection des pannes proposée .....91
Tableau 3.5	Résultats de validation des tests pour le design des RO implémentés dans les IOB.....93
Tableau 3.6	Résultats de campagnes aléatoires d'injection des pannes le circuit B08.....95
Tableau 4.1	Résultats d'injection des pannes dans les différents sous-ensembles du design des RO .....102
Tableau 4.2	Comparaison des résultats des différentes approches.....104



## LISTE DES FIGURES

		Page
Figure 1.1	Effet du vent solaire sur la magnétosphère terrestre (Photo de NASA) ....	11
Figure 1.2	Éruption solaire en anneau (Photo de NASA) .....	12
Figure 1.3	Illustration des ceintures de radiations de Van Allen .....	13
Figure 1.4	Illustration de la douche des particules et la génération des particules secondaires .....	15
Figure 1.5	Thyristor parasite dans un circuit CMOS causé par un SEL .....	17
Figure 1.6	(a) Illustration du voisinage d'un MBU dans les cellules mémoires d'un même mot logique (b) MBU à trois bits flippés .....	18
Figure 1.7	Plateforme de test Rosetta.....	22
Figure 1.8	Principe d'un accélérateur (a) linéaire de type Van de Graaf (b) circulaire.....	24
Figure 1.9	Bascule D avec SCAN .....	29
Figure 1.10	Chaîne SCAN.....	30
Figure 1.11	Éléments internes du FPGA.....	32
Figure 1.12	Architecture d'une cellule SRAM, d'un FPGA .....	32
Figure 1.13	Couche opérative et couche de configuration du FPGA.....	34
Figure 1.14	Architecture interne d'un CLB d'un FPGA Virtex-5 .....	35
Figure 1.15	Architecture interne d'un IOB.....	36
Figure 1.16	Architecture d'une cellule SRAM de configuration.....	37
Figure 1.17	Architecture de la plateforme Thesic+.....	39
Figure 1.18	FLIPPER: plateforme matérielle/logicielle.....	40
Figure 1.19	Architecture de la plateforme FLIPPER.....	42
Figure 1.20	Architecture de la plateforme FT-UNSHADES .....	43

Figure 1.21	Architecture de la solution d'injection des pannes proposée .....	45
Figure 1.22	Architecture de la solution d'injection des pannes proposée .....	46
Figure 1.23	Contrôleur d'émulation des pannes .....	47
Figure 1.24	Architecture de l'injecteur de pannes proposé .....	48
Figure 1.25	Architecture de l'injecteur des pannes FIRED .....	49
Figure 1.26	Architecture du système d'injection d'erreurs par émulation .....	52
Figure 1.27	Méthodologie de protection des LUT .....	55
Figure 2.1	Entrées et sorties du <i>SEU Controller</i> .....	59
Figure 2.2	La primitive FRAME_ECC .....	62
Figure 2.3	La primitive ICAP du Virtex 5 .....	63
Figure 2.4	Position des bits non-utilisés dans une trame .....	63
Figure 2.5	Distinction des bits de configuration et de ceux du <i>SEU Controller</i> et de ceux potentiellement critiques.....	65
Figure 2.6	Procédure de détermination des bits de configuration du <i>SEU Controller</i> .....	66
Figure 2.7	Procédure de détermination des bits essentiels du <i>SEU Controller</i> .....	67
Figure 2.8	Premier montage expérimental (RO dans les CLB) .....	70
Figure 2.9	Deuxième montage expérimental (RO dans les IOB).....	72
Figure 3.1	Procédure d'injection des pannes.....	83
Figure 3.2	Comparaison de $Z_T$ obtenu à partir de l'estimation, ainsi qu'à partir des différentes campagnes d'injection des pannes.....	89
Figure 5.1	Design concaténant tous les LUT du FPGA Virtex-5 .....	114
Figure 5.2	Design concaténant les 14 400 LUT du Virtex-5 VLX50T (a) routage dans la moitié basse (b) placement dans la moitié basse (c) routage dans la moitié haute (d) placement dans la moitié haute .....	115
Figure 5.3	Procédure proposée pour l'extraction des bits de configuration des LUT utilisés par un design.....	116

Figure 5.4 Les adresses des slices du (a) Artix-7 A200T (b) Virtex-5 VLX50T.....118

Figure 5.5 Procédure automatisée d'injection des pannes ciblant les LUT .....119

Figure 5.6 Illustration des multiplexeurs dans les RO .....120

ClicCours.com





## LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ACM	Auto Correction Mode
CEU	Code Emulated Upsets
CLB	Configurable Logic Block
CMOS	Complementary Metal-Oxide-Semiconductor
COTS	Commercial Off-The-Shelf
CPLD	Complex Programmable Logic Device
CRC	Cyclic Redundancy Check
DOM	Detection Only Mode
DUT	Device Under Test
FAST	Fault Simulator for Transients
FPGA	Field Programmable Gate Array
GeV	Gigaelectron-Volt
IOB	Input/Output Block
JTAG	Joint Test Action Group
LUT	Look Up Table
MBU	Multiple Bits Upset
MCU	Multiple Cell Upset
MEFISTO	Multi-level Error/Fault Injection Simulation Tool
MeV	Megaelectron-Volts
MPTB	Micro-electronic and Photonic TestBed
NCD	Native Circuit Description
ODC	Observable Delay Change
RO	Ring Oscillator
ROB	Ring Oscillator Break
RTL	Register Transfer Level
SDRAM	Synchronous Dynamic Random-Access Memory
SEB	Single Event Burnout
SEE	Single Event Effect

## XXIV

SEFI	Single Event Functional Interrupt
SEGR	Single Event Gate Rupture
SEL	Single Event Latch-up
SER	Soft Error Rate
SET	Single Event Transient
SEU	Single Event Upset
SHE	Single event Hard Error
SoC	System on Chip
SOI	Silicon On Insulator
SRAM	Static Random Access Memory
STRV	Space Technology Research Vehicles
TCL	Tool Command Language
TMR	Triple Modular Redundancy
UART	Universal Asynchronous Receiver Transmitter
VERIFY	VHDL-based Evaluation of Reliability by Injecting Faults efficiently
VHDL	VHSIC Hardware Description Language
XDL	Xilinx Design Language

## INTRODUCTION

### **Motivation : contexte général**

Depuis le début des missions spatiales, les effets des radiations cosmiques sont une préoccupation importante pour les acteurs principaux travaillant dans le domaine de l'aérospatial. En effet, le milieu de déploiement de leurs systèmes, l'espace, est un milieu hostile caractérisé par la diversité des particules énergétiques qui parcourent l'univers. La Terre aussi est la cible de ces particules. Ce phénomène est causé principalement par les activités régulières du soleil ainsi que par les radiations cosmiques provenant d'ailleurs.

Malgré que l'existence des radiations a été démontrée dès le début du vingtième siècle et que les premières mesures de flux de particules en orbites ont été effectués vers la fin des années cinquante, le phénomène de l'inversion du contenu des cellules mémoire causé par l'interaction avec ces particules n'a été rapporté que vers la fin des années soixante-dix (Guenzer, Wolicki et Allas, 1979; May et Woods, 1979). Suite à cela, les techniques de durcissement des circuits destinés aux applications aérospatiales ont commencé à être explorées et développées.

La taille relativement grande des transistors à l'époque permettait de réduire l'effet des particules sur l'électronique déployée dans les environnements radiatifs hostiles. En effet, seules des particules très énergétiques pouvaient générer des anomalies. La course à la miniaturisation des circuits électroniques s'est traduite par la diminution incessante de la taille des transistors et par l'augmentation de leur nombre sur une même puce. Cette augmentation de complexité a, quant à elle, mené à une augmentation de la probabilité d'apparition des pannes et à un besoin grandissant de s'assurer de la robustesse des circuits face aux radiations.

Les effets des radiations cosmiques ne sont pas observés seulement dans l'espace mais aussi, dans une moindre mesure, au niveau du vol ainsi qu'au niveau du sol. Ceci impose la concentration des efforts pour l'étude de la vulnérabilité des circuits destinés à des applications aéronautiques ainsi que des applications critiques fonctionnant dans la surface

terrestre. Ces études deviennent plus complexes surtout avec l'apparition de l'électronique à grand public COTS (*Commercial Off-The-Shelf*) et l'adoption de ces derniers pour des applications critiques. Ceci est dû à l'utilisation de puces plus complexes qui intègrent de plus grandes quantités de mémoires et transistors que ceux durcis pour les applications aérospatiales (Bocquillon, 2009).

Parmi ces COTS figurent les FPGA (*Field Programmable Gate Array*). Certains types de FPGA sont des plateformes idéales pour le prototypage rapide ainsi que pour les applications requérant la reconfiguration dynamique. Ces caractéristiques rendant ces FPGA de plus en plus demandés par la communauté de l'aérospatial viennent avec l'inconvénient de la vulnérabilité de leurs mémoires de configuration face aux radiations. Ces radiations peuvent modifier les designs implémentés dans les FPGA. Beaucoup de travaux dans la littérature cherchent à trouver des solutions pour augmenter la robustesse de ces circuits, en trouvant tout d'abord des moyens simples, efficaces et fiables pour évaluer la sensibilité de ces circuits face aux radiations et en prenant, par la suite, les mesures nécessaires pour les durcir en adoptant les techniques de mitigation convenables. C'est dans ce contexte que se présente cette thèse.

### **Motivation : cadre de la thèse**

Cette thèse s'inscrit dans le cadre du projet global CRIAQ AVIO403, dont l'objectif général consistait à intervenir sur le processus d'intégration (de la conception au test) des circuits électroniques tout en agissant sur les méthodologies existantes afin de considérer l'impact des radiations cosmiques. Ce projet global voulait offrir aux concepteurs les stratégies et les techniques nécessaires pour valider, vérifier et tester leurs designs à tous les niveaux d'abstraction et tout au long du processus d'intégration, en vue d'augmenter les chances de réussir le test en certification des systèmes embarqués.

Le but de ce projet global était de développer des stratégies d'évaluation et de protection contre les effets des radiations. Vu que ces derniers doivent être étudiés dans toutes les étapes du flot d'intégration, l'objectif global a été divisé en quatre sous-objectifs :

- l'élaboration et la mise en œuvre d'une stratégie de validation précoce permettant l'exploration du design à haut-niveau du flot de conception;
- l'élaboration et la mise en œuvre d'une stratégie de vérification à niveaux d'abstraction multiples supportant la simulation des pannes liées aux effets des radiations;
- l'élaboration d'une stratégie de conception facilitant les tests en pré-certification pour la robustesse à l'égard des radiations;
- l'exploration des stratégies de tolérance de pannes appliquées sur des systèmes industriels complexes.

Afin de fournir des résultats réels servant comme point de référence pour les intervenants dans le projet CRIAQ AVIO403 ainsi que pour générer des résultats de validation, deux campagnes de tests accélérés sous faisceaux de protons ont été faites au laboratoire TRIUMF. Comme pour cette thèse, les circuits ciblés par ce projet global sont les FPGA à base de mémoire SRAM.

L'objectif spécifique de cette thèse, qui s'inscrit à l'intérieur du troisième sous-objectif du projet AVIO403, est l'élaboration et la mise en œuvre d'une stratégie de pré-certification versus l'impact des radiations sur les systèmes électroniques embarqués. L'idée consiste à faciliter l'injection volontaire des pannes dans certains éléments de mémoire du système cible pour émuler la présence d'événements singuliers (*Single Event Upset*, SEU) ou multiples (*Multiple Bit Upset*, MBU) causés par les radiations et observer la réaction du système. Une stratégie de validation automatique du comportement attendu doit être mise en place. L'injection est un premier défi. La validation en est un autre. Ce test en pré-certification est mis en place pour identifier les systèmes ne rencontrant pas les spécifications de robustesse face aux radiations et pour éviter ainsi de soumettre ces systèmes à la phase, très coûteuse, de bombardement de particules. Le test en pré-certification, objet de notre travail de thèse, constitue l'un de ces moyens effectués juste avant la certification.

La stratégie d'injection des pannes à implémenter doit tirer profit de l'infrastructure déjà existante dans les composants électroniques ciblés ainsi que des certaines fonctionnalités internes pour faciliter les opérations de test. Cette stratégie doit fournir des résultats très proches de ceux obtenus par les tests de certification et par la suite donner aux concepteurs

une estimation réaliste sur la sensibilité des circuits électroniques. Cette estimation permettra de valider le choix de la technique de mitigation. Finalement, la stratégie doit minimalement affecter le design, pouvoir être appliquée une fois le système mis en œuvre avec un minimum d'infrastructures externes, et permettre d'identifier les parties non suffisamment robustes.

Dans ce contexte, les défis à relever pendant les travaux de thèse gravitent autour du choix de l'infrastructure de test en pré-certification à mettre en place et de son influence sur le flot de conception, de l'identification d'un modèle représentatif des pannes à injecter et de l'injection automatisée de ces pannes.

### **Problématique de recherche**

L'espace n'est pas le seul environnement où l'on retrouve des radiations. En effet, il a déjà été prouvé que les circuits intégrés peuvent être également affectés par les radiations au niveau des altitudes de vols commerciaux (Normand, 1996a). Au niveau du sol, l'effet des radiations peut se faire sentir, même si cet effet est beaucoup moindre (Normand, 1996b). De ce fait, le domaine de l'aéronautique se préoccupe depuis un certain temps des effets des radiations cosmiques. Les études menées pour explorer la sensibilité des dispositifs électroniques destinés à l'avionique face au bombardement des particules énergétiques sont de plus en plus nombreuses (Bagshaw, 2008).

La criticité potentielle d'un mauvais fonctionnement dû aux radiations incite les industriels à investir de plus en plus dans le développement des outils de conception permettant d'une part le choix des stratégies de mitigation des effets des radiations et d'autre part leur vérification via l'estimation de la fiabilité résultante. Cet intérêt est également alimenté par la tendance de la communauté aérospatiale à l'utilisation plus marquée des composants COTS dans leurs applications, au détriment des composants dits durcis, plus résistants aux radiations mais très coûteux. Cette plus grande utilisation est motivée par la nécessité de réduire le coût des systèmes. Les composants COTS sont en effet beaucoup moins coûteux que les composants durcis, tout en offrant une complexité et une performance accrues (Bocquillon, 2009), en raison du fait qu'ils bénéficient des dernières technologies et qu'ils sont donc en avance de 2 ou 3 générations technologiques par rapport aux composants durcis.

Ces observations s'appliquent également aux FPGA à base de mémoire SRAM qui constituent la technologie cible pour cette thèse. Le domaine aéronautique est donc à la recherche de solutions offrant un compromis acceptable entre la fiabilité face aux radiations et le coût des systèmes. La recherche de telles solutions a des répercussions sur le flot complet de conception et développement des systèmes embarqués. En effet, ceci demande aux concepteurs de vérifier tout au long de ce flot la robustesse aux radiations de leur design. Il existe à cet effet des outils de simulation permettant l'injection de pannes. Cette forme de vérification a ses limites, surtout que cet effort supplémentaire vient s'ajouter à celui de la vérification conventionnelle, déjà considérée comme le goulot d'étranglement du processus de conception. Le système, une fois mis en œuvre, devrait être de plus soumis à des tests de certification, très coûteux, effectués en laboratoire à l'aide d'accélérateurs de particules.

Considérant les coûts de cette certification et les limitations de la simulation de pannes, il devient impératif de maximiser les chances de succès de cette étape ultime. Les tests dits de pré-certification constituent une avenue intéressante à explorer. Dans le processus de développement d'un système devant être certifié, ces tests sont effectués une fois que le système est mis en œuvre mais avant qu'il ne soit envoyé à la certification. La mise en place de tels tests, qui se doivent d'être représentatifs de ceux réalisés pendant la certification, aura des répercussions sur le flot de conception et de développement, dans la mesure où ce flot devra être modifié afin d'en faciliter l'application.

### **Contributions de la thèse**

Cette thèse a pour objectif de comprendre les effets des radiations sur la procédure de conception des systèmes avioniques et spatiaux en validant le concept de la pré-certification. En effet, le travail effectué est orienté vers l'étude par émulation des effets des pannes sur la mémoire de configuration des FPGA à base de mémoire SRAM. En particulier, les effets des SEU représentent l'intérêt majeur de ce travail de thèse qui permet de générer des résultats les plus proches possible de ceux donnés par les tests de certification et par la suite d'éviter d'envoyer des circuits non suffisamment robuste à la phase coûteuse de la certification.

Cette thèse présente des méthodes d'optimisation de la procédure d'émulation des SEU dans la mémoire de configuration des FPGA à base de mémoire SRAM. Vu que cette tâche demande une connaissance précise de l'architecture du FPGA ainsi que le système d'adressage des bits et des frames de configuration, on a eu recours aux outils fournis par le fournisseur du FPGA afin d'extraire cette information qui est décrite dans cette thèse.

Plus précisément, les contributions principales de cette thèse sont les suivantes :

- une nouvelle approche d'injection des pannes tenant en compte la différence de sensibilité relative entre les bits de configuration mis à '0' et ceux à '1' afin de reproduire le plus précisément possible les résultats des tests accélérés. En effet, une nouvelle approche d'injection des pannes reproduisant les résultats obtenus à partir des tests accélérés sous radiations en étudiant l'effet des SEU sur les FPGA à base de mémoire SRAM est présentée. Selon les tests sous radiation effectués sur un Virtex-5 à TRIUMF, les bits de configuration à '1' sont à peu près deux fois plus sensibles que ceux à '0'. Ceci est exploité lors de la génération des séquences de test utilisées pour les expérimentations d'émulation par injection des pannes dans l'objectif de les rendre plus réalistes. L'efficacité de l'approche proposée est validée en comparant ces résultats avec ceux obtenus par les tests sous faisceaux de protons. Cette contribution fait l'objet d'un article de revue publié (Souari et al., 2016);
- une nouvelle approche priorisant l'injection des pannes dans des ensembles de bits de configuration spécifiques selon leurs contenus ainsi que le type des ressources qu'ils configurent pour les FPGA de Xilinx. Cette approche offre une meilleure estimation des bits critiques, permettant ainsi d'inverser plus de bits critiques lors d'injection de pannes et aussi de minimiser le nombre de *bit flips* (inversion d'état des éléments de mémoire) requis pour valider la robustesse d'un design en comparaison avec la méthode d'injection des pannes traditionnelle aléatoire. Cette contribution a fait l'objet d'un article présenté dans un symposium IEEE (Souari et al., 2015b);
- une méthode d'injection des pannes automatisée permettant d'évaluer les bits de configuration des LUT utilisés par les designs implémentés dans les FPGA de Xilinx. L'approche proposée permet d'identifier tous les bits de configuration utilisés par les



LUT pour un design spécifique et puis d'y injecter des SEU et des MBU. L'approche proposée ne requiert aucun outil externe pour identifier les bits des LUT. Elle est efficace offrant 100% de couverture de pannes et elle est applicable pour les nouvelles générations de FPGA de Xilinx. Cette contribution a fait l'objet d'un article présenté dans un symposium IEEE (Souari et al., 2015a).

En travaillant sur ces contributions principales, des contributions d'ordre secondaire ont été mises en œuvre. Elles incluent :

- l'automatisation de la procédure d'injection des pannes;
- la protection de l'autodestruction de l'outil d'injection des pannes;
- l'identification des bits de configuration des LUT dans les FPGA de Xilinx.

### **Organisation de la thèse**

La thèse est structurée comme suit. Le chapitre 1 présente l'état de l'art. En premier lieu, les notions de base du domaine de recherche d'intérêt sont introduites. En second lieu, une revue de littérature est effectuée afin de lier les travaux existants aux objectifs de la thèse et par la suite distinguer l'originalité des contributions présentées. Le chapitre 2 introduit l'environnement d'émulation adopté pour tous les travaux de tests effectués. La carte à base de FPGA cible, les outils de mesure et de contrôle et les designs sous test sont présentés. De plus, l'outil d'injection des pannes ainsi que la méthode de sa protection contre l'autodestruction sont décrits. Le chapitre 3 établit le lien entre le contenu des bits de configuration et la vulnérabilité de ces derniers afin de l'exploiter pour générer des résultats d'injection des pannes par émulation se rapprochant de ceux obtenus par les tests accélérés. Dans le chapitre 4, une nouvelle approche optimisant la procédure d'émulation des SEU est présentée. Elle permet de maximiser le nombre des bits critiques inversés pour un nombre déterminé des *bit flips* ou encore minimiser l'erreur d'estimation du nombre des bits critiques. Une approche d'évaluation des bits de configuration configurant les LUT d'un design spécifique est détaillée dans le chapitre 5. La simplicité et l'efficacité de l'approche proposée sont discutées en comparaison avec d'autres approches trouvées dans la littérature. Une conclusion générale clôture cette thèse.



# CHAPITRE 1

## ÉTAT DE L'ART

### 1.1 Introduction

La revue de littérature décrite dans ce chapitre a pour objectif de présenter tout d'abord le contexte général dans lequel se positionnent les travaux décrits par cette thèse. Plus particulièrement, elle a pour objectif principal de situer la problématique et les travaux de recherche par rapport à l'état de l'art déjà existant.

Dans le cadre de notre problématique de recherche et vu que ce projet se fait en collaboration avec la France, une attention particulière a été portée sur les travaux faits au sein du groupe ARIS du laboratoire TIMA. Le but était de se familiariser avec les travaux de nos collaborateurs français afin de déployer leur expertise dans le projet AVIO403. L'état de l'art décrit dans ce chapitre s'inspire essentiellement des travaux décrits dans (Foucard, 2010) et (Bocquillon, 2009).

Notre revue de littérature a pris en considération quatre axes essentiels, à savoir les différents environnements radiatifs, les effets des radiations sur les circuits électroniques, les divers moyens d'évaluation de la sensibilité de ces circuits aux radiations et la structure générale des FPGA à base de mémoire SRAM.

Ce chapitre se divise en deux parties principales. La première, présentée dans la section 1.2, présente les concepts de base, où les données préliminaires nécessaires pour situer le cadre général de la thèse sont résumées. En effet, les quatre axes précédemment mentionnés sont décrits, respectivement, dans les sous-sections allant de 1.2.1 jusqu'à 1.2.4. La sous-section 1.2.5 décrit brièvement comment configurer un FPGA Virtex-5. La deuxième partie, donnée par la section 1.3, présente la mise en contexte des travaux de la thèse. En effet, une revue de littérature liée plus aux contributions de la thèse a été faite afin de les distinguer des autres travaux. De ce fait, la sous-section 1.3.1 se focalise sur les techniques d'injection des pannes, la sous-section suivante présente les différentes approches permettant d'optimiser ces

derniers et la sous-section 1.3.3 s'intéresse aux injections des pannes ciblant les LUT. Enfin, la section 1.4 conclut ce chapitre.

## **1.2 Notions de base**

### **1.2.1 Environnements radiatifs**

Les circuits intégrés modernes dans des environnements radiatifs différents. Ces derniers sont généralement classés en deux catégories : l'environnement radiatif spatial et l'environnement radiatif atmosphérique.

#### **A. Environnement radiatif spatial**

Cet environnement est divisé en quatre milieux principaux de sources de radiations à savoir le rayonnement cosmique, le vent solaire, les éruptions solaires et les ceintures de radiations. Ces environnements émettent des particules énergétiques différentes qui comprennent essentiellement des électrons, des protons et des ions lourds. Les caractéristiques de ces différents milieux sont présentées dans ce qui suit.

##### **1) Le rayonnement cosmique**

Les traces du rayonnement cosmique ont été détectées pour la première fois en 1912. L'origine de ce rayonnement, malgré qu'elle ne soit pas tout à fait connue, est supposée provenir des sources galactiques ainsi que d'autres sources extragalactiques. Le rayonnement cosmique est composé essentiellement de 1 % d'ions, qui sont les particules les plus énergétiques et dont leur énergie varie de quelques MeV jusqu'à plusieurs GeV. Les protons représentent 87 % du rayonnement cosmique tandis que les noyaux d'hélium en constituent 12 %.

Les flux du rayonnement cosmique sont influencés par l'activité du soleil. En effet, lors d'un vent solaire, le soleil est en pleine activité, ce qui entraîne une diminution de la densité des particules générées par le rayonnement cosmique dans le système solaire, vu que les flux cosmiques s'opposent au vent solaire.

## 2) Le vent solaire

Le vent solaire est issu de la couronne solaire; la partie atmosphérique extérieure du soleil. Ce vent émet des flux de plasma remplissant tout le système solaire. Le plasma est formé principalement par des électrons, des protons et noyaux d'hélium. La densité des particules du plasma varie de  $10^{12}$  particules/cm<sup>3</sup> au niveau du soleil jusqu'à 10 particules/cm<sup>3</sup> au niveau de l'orbite terrestre.

Le vent solaire, dans son interaction avec les planètes du système solaire en général et avec la Terre en particulier, crée des cavités magnétosphériques. En effet, la magnétosphère terrestre est une cavité naturelle créée par les interactions du vent solaire avec le champ géomagnétique, afin de protéger la Terre des attaques radiatives spatiales. La figure 1.1 montre la forme de la magnétosphère terrestre sous effet du vent solaire.

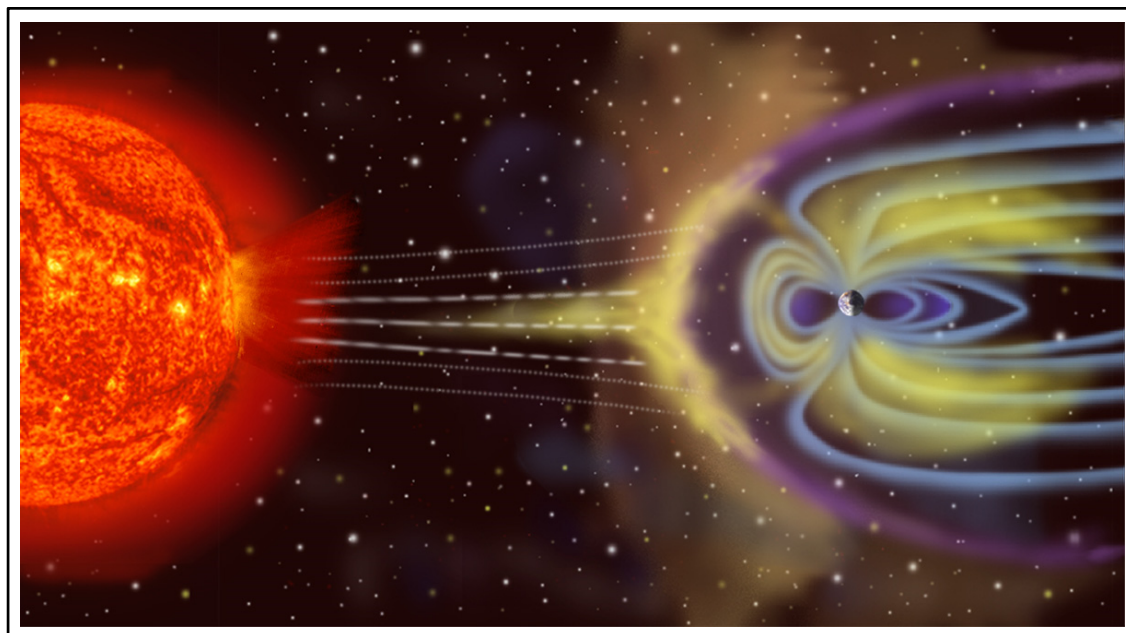


Figure 1.1 Effet du vent solaire sur la magnétosphère terrestre (Photo de NASA)  
Tirée de Foucard (2010)

### 3) Les éruptions solaires

Les activités solaires varient d'une façon cyclique, chaque cycle durant en moyenne 11 ans et pendant lequel le soleil passe des années d'activité maximale et des années de calme.

Durant un cycle solaire, on distingue deux périodes essentielles; la première est la période de forte activité qui dure en moyenne 7 années et la période de faible activité durant en moyenne 4 années. La période de forte activité est marquée par les éruptions solaires qui se manifestent par la diffusion intensive des particules énergétiques. Selon la nature de ces particules, on distingue deux types d'éruptions solaires:

- les éruptions solaires à protons où les particules émises sont essentiellement des protons. Cette émission de protons dont leur énergie peut atteindre quelques centaines de MeV dure de quelques heures à quelques jours;
- les éruptions solaires à ions lourds, comme le nom l'indique, correspondent à une émission d'ions lourds de durée maximale de quelques heures.

La figure 1.2 montre une éruption solaire avec panaches émis en anneau.

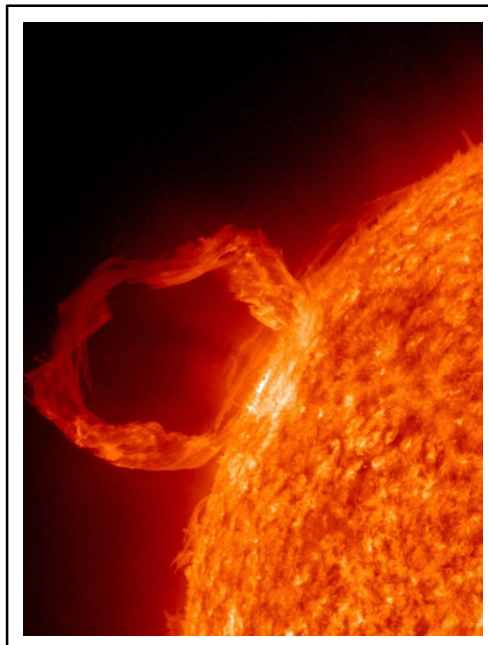


Figure 1.2 Éruption solaire en anneau (Photo de NASA)

#### 4) Les ceintures de radiations

Elles sont appelées aussi les ceintures de Van Allen par référence à James Van Allen; le découvreur de ces ceintures en 1958. Ces ceintures sont formées par des électrons, protons et ions lourds qui sont piégés dans la magnétosphère de la Terre à cause du champ magnétique terrestre.

Ces particules ont une trajectoire hélicoïdale ce qui donne la forme de tores aux ceintures de radiations. La figure 1.3 montre la forme des ceintures de Van Allen. Les ceintures de radiations sont composées de deux ceintures principales, nommées ceinture intérieure et ceinture extérieure. La première est située entre les altitudes de 700 km et 10000 km, elle est constituée essentiellement des protons ayant des énergies allant jusqu'à plusieurs centaines de MeV. La deuxième est placée entre les altitudes de 13000 km et 65000 km composée essentiellement d'électrons ayant quelques MeV d'énergie. Le décalage entre l'axe magnétique et l'axe de rotation de la terre influence la distribution des particules dans la zone des ceintures de radiations, ce qui crée des zones sur terre où l'impact de ces ceintures est senti le plus, notamment la zone de l'anomalie de l'Atlantique sud (SAA).

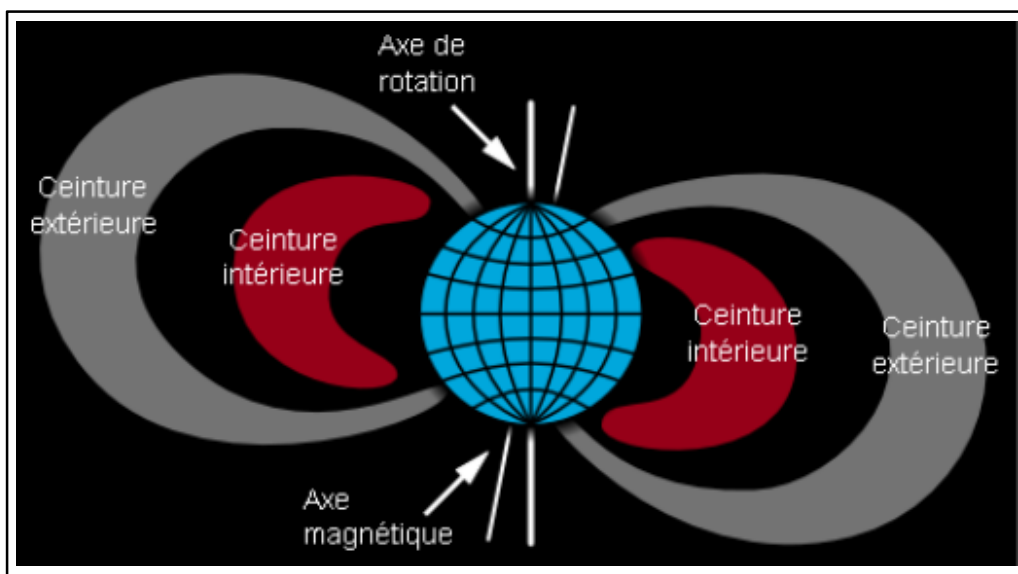


Figure 1.3 Illustration des ceintures de radiations de Van Allen  
Tirée de Foucard (2010)

## **B. Environnement radiatif atmosphérique**

Au niveau de l'environnement atmosphérique, les circuits intégrés sont affectés par deux types de radiations; les radiations artificielles causées par les impuretés radioactives et les radiations naturelles.

### **1) Les radiations artificielles**

Ce type de radiation est dû à la contamination de la chaîne de production des circuits électroniques par des impuretés radioactives. En effet, en 1987, IBM a remarqué un taux d'erreur élevé lors du test de quelques produits (Ziegler et al., 1996). L'origine du problème provient d'une contamination par le polonium qui est un élément  $\alpha$ -émetteur artificiel.

### **2) Les radiations naturelles**

L'atmosphère terrestre joue un rôle primordial dans la protection de la terre de l'exposition directe aux radiations spatiales. Elle filtre une grande partie des radiations. Seule une petite partie, appelée radiations atmosphériques, peut entrer dans l'atmosphère terrestre. Cette partie est générée à cause des interactions des radiations cosmiques avec la composition chimique de l'atmosphère à savoir l'azote et l'oxygène.

Deux types de réactions peuvent se produire au niveau de l'atmosphère : soit les particules s'ionisent en perdant une partie de leurs énergies, soit des nouvelles particules sont générées à cause des réactions nucléaires, ce dernier phénomène s'appelant « la douche des particules ». La figure 1.4 illustre ce phénomène ainsi que les différents types de particules formés, notons essentiellement les neutrons, les protons, les électrons, les muons, les pions et les photons.

Plusieurs études ont montré que l'énergie des flux de radiations varie selon l'altitude et la latitude. En effet, elle diminue de 300 fois quand on descend de l'altitude de 12 km jusqu'au niveau de la mer; aussi elle augmente de 4 fois quand on passe de l'équateur vers les pôles.



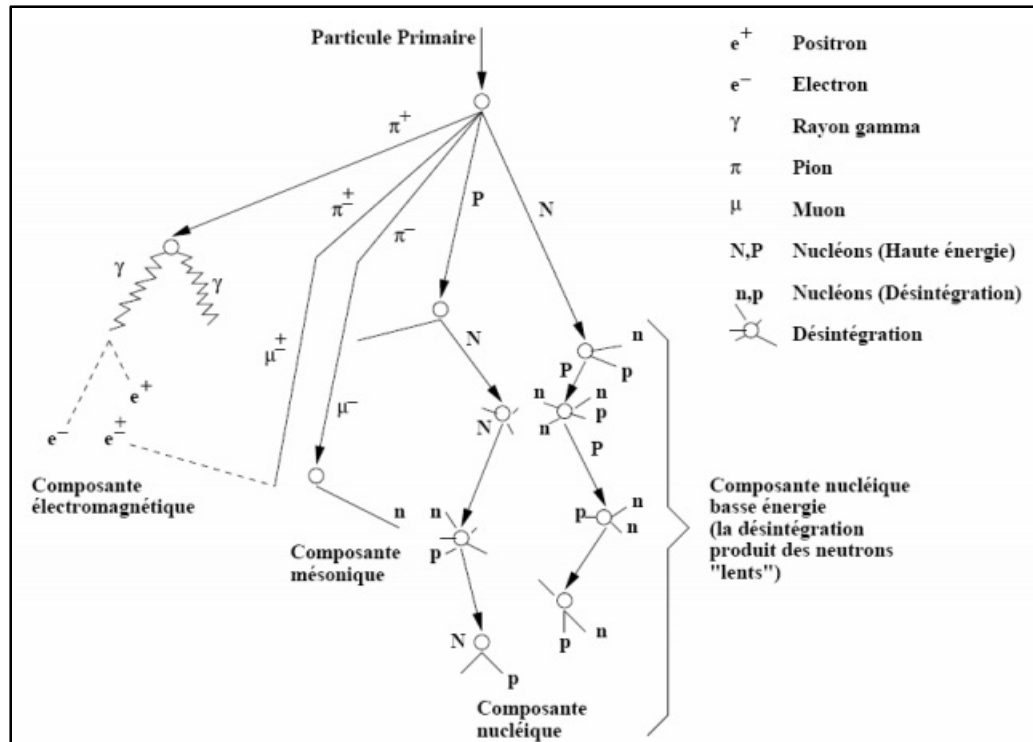


Figure 1.4 Illustration de la douche des particules et la génération des particules secondaires  
Adaptée de Ziegler et al. (1996)

Ce qui nous intéresse le plus sont les altitudes des vols commerciaux, où les particules dominantes sont les neutrons. Ces derniers peuvent causer des défaillances au niveau des circuits intégrés de par leurs interactions avec les atomes constituant ces circuits. Au niveau de la mer, le flux des neutrons atmosphériques est estimé à 10 particules/cm<sup>2</sup>/heure, tandis qu'au niveau des vols commerciaux, un flux de 10<sup>4</sup> particules/cm<sup>2</sup>/heure peut être obtenu (Foucard, 2010). Ces valeurs, données à titre indicatif, sont influencées par l'activité solaire.

### 1.2.2 Les effets des radiations sur les circuits électroniques

Les effets des radiations sur les circuits électroniques sont habituellement classés en deux catégories principales (Foucard, 2010): les effets de la dose cumulée et les effets dits "singuliers" causés par la collision d'une seule particule dans une zone sensible du circuit.

Dans la suite, les deux types d'effets sont détaillés.

## A. Les effets de dose

Les effets de dose cumulée consistent à une accumulation des charges au niveau des oxydes isolants des circuits intégrés. En effet, après que les particules ionisées soient entrées en collision avec le circuit intégré, elles cèdent une partie ou la totalité de leurs charges au circuit. L'accumulation de ces charges a un effet nuisible sur les circuits intégrés, comme par exemple une variation de la tension de seuil d'un transistor. Effectivement, elle peut mener au dysfonctionnement de ces derniers et même à leur destruction (Bocquillon, 2009; Foucard, 2010).

## B. Les évènements singuliers

Les évènements singuliers (*single event effect*, SEE) (Gaillard, 2011) sont générés par une seule particule entrant en collision avec le circuit. Ces évènements sont classés en deux catégories selon leurs effets sur les composants électroniques, selon que ces effets soient destructifs ou non. Les différents types d'évènements singuliers sont définis dans ce qui suit, selon leur catégorie respective.

### 1) Effets destructifs

- **SEL (*Single Event Latchup*)** : Les ions lourds peuvent causer ce type de panne en activant la conduction d'une structure parasite npnp ou pnpn qui forme un thyristor. Cet évènement est suivi d'une augmentation de la consommation du courant, pouvant détruire le composant électronique. La détection d'un tel évènement exige la coupure d'alimentation afin de protéger le circuit. Les SEL ont déjà été observés dans des processeurs (Moran et al., 1996) et dans des microprocesseurs (Buchner et al., 1998). La figure 1.5 illustre l'effet d'un SEL à savoir l'enclenchement d'un thyristor.

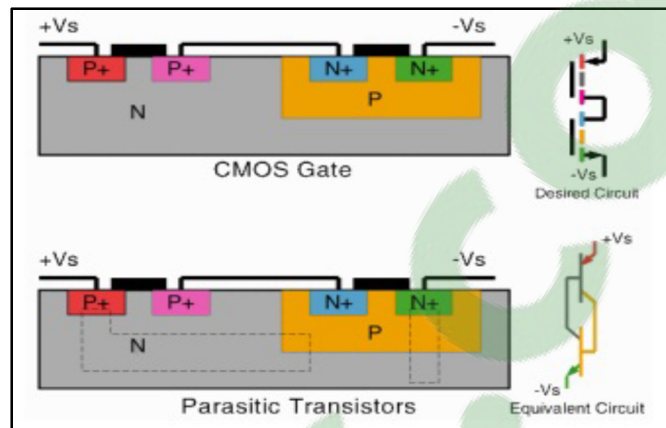


Figure 1.5 Thyristor parasite dans un circuit CMOS causé par un SEL  
Tirée de Bocquillon (2009)

- **SEB (*Single Event Burnout*)** : Ce type de panne est causé par un passage de courant fort dans les transistors de puissance, ce qui cause leur destruction.
- **SEGR (*Single Event Gate Rupture*)** : Il consiste en la création d'un chemin conducteur dans les transistors de puissance avec un courant d'énergie suffisante pour mener à une destruction de l'oxyde de grille.
- **SHE (*Single event Hard Error*)** : C'est un changement irréversible dans une cellule du composant causé par un SEU.

## 2) Effets non destructifs

- **SEU (*Single Event Upset*)** : C'est le type de pannes le plus fréquent dans les cellules mémoire de types SRAM et SDRAM. Il est généré par une particule impactant une zone sensible du circuit intégré. La panne consiste à un basculement de bit d'un état logique à son complémentaire.

Le SEU n'a pas un effet destructif et une simple réécriture dans la cellule mémoire peut habituellement corriger le problème;

- **SET (*Single Event Transient*)** : Une particule peut générer une charge qui se transforme en un pulse de courant se propageant dans le circuit et engendrant une erreur. Le SET peut devenir un SEU dans le cas où il est mémorisé;

- **MCU (*Multiple Cell Upset*)** : Ce type de défaillance consiste en un basculement de plusieurs points mémoires voisins causés par une seule particule. La probabilité des MCU dans les circuits intégrés augmente avec la diminution de la taille de transistors.

La diminution de la finesse de gravure agit sur deux paramètres permettant d'augmenter la probabilité des MCU dans les circuits intégrés.

- 1) La diminution de la charge critique par bit. En effet, la charge critique nécessaire pour faire basculer un bit de mémoire diminue à cause de la diminution des tensions d'alimentation et des capacités parasites dans la cellule mémoire;
- 2) La diminution de la distance séparant deux zones sensibles, ce qui augmente la possibilité de collecter une charge déposée dans plusieurs zones sensibles au même moment.

- **MBU (*Multiple Bit Upset*)** : Ce type de panne est un cas particulier de MCU.

Il mène à plusieurs erreurs provoquées par une particule dans un même mot logique. Plusieurs études ont été menées sur les MBU (Gasiot, Giot et Roche, 2007; Graham et al., 2005; Tipton et al., 2008). La figure 1.6 illustre l'effet d'un MBU sur les cellules adjacentes.

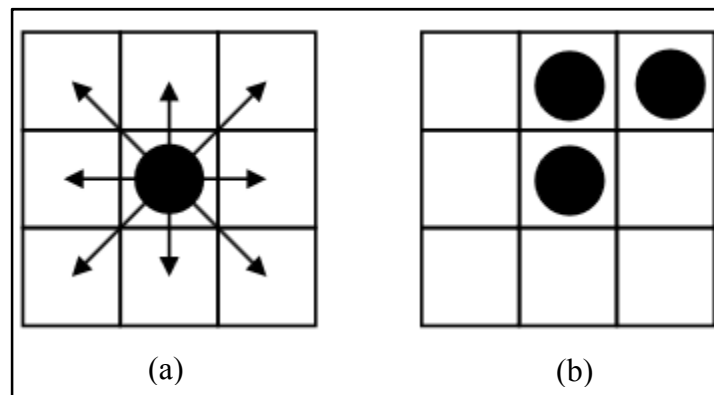


Figure 1.6 (a) Illustration du voisinage d'un MBU dans les cellules mémoires d'un même mot logique (b) MBU à trois bits flippés  
Tirée de Quinn et al. (2005)

- **SEFI (*Single Event Functional Interrupt*)** : Ce type de défaillance se manifeste par un fonctionnement inapproprié du composant. Un SEFI est souvent lié à un SEU causant

l'interruption du fonctionnement normal du circuit. Pour faire face à ce type de panne, il suffit de réinitialiser le système.

Le tableau 1.1 résume les différents types d'évènements singuliers non destructifs.

Tableau 1.1 Différents types d'évènements singuliers (SEE) non destructifs

<b>Type de SEE</b>	<b>Description</b>
SEU ( <i>Single Event Upset</i> )	Modification de l'information stockée dans une cellule mémoire ou dans une bascule
MBU ( <i>Multiple Bit Upset</i> )	Modification simultanée de l'information dans des cellules mémoires adjacentes d'un même mot logique
SET ( <i>Single Event Transient</i> )	Impulsions transitoires perturbant les réseaux combinatoires
MCU ( <i>Multiple Cell Upset</i> )	Modification de l'information stockée dans plusieurs cellules mémoires ou flips-flops
SEFI ( <i>Single Event Functional Interrupt</i> )	Interruption du fonctionnement normal du composant

Dans le cadre du projet AVIO403 en général et de cette thèse en particulier, les évènements qui nous intéressent sont les évènements singuliers ayant des effets non destructifs. Principalement, ce sont les SEU (*Single Event Upset*) et les MBU (*Multiple Bit Upset*) qui nous intéressent le plus car ils constituent les pannes les plus communément rencontrées dans les FPGA à base de mémoire SRAM (Foucard, 2010), sachant que c'est le circuit cible de nos travaux.

### **1.2.3 Évaluation de la sensibilité des circuits intégrés à l'égard des radiations**

Les composants électroniques sujets aux évènements singuliers causés par les radiations doivent être validés avant qu'ils soient embarqués et utilisés dans leurs milieux de

fonctionnement finaux. Plusieurs techniques ont été développées pour estimer la sensibilité de ces composants.

Les principales techniques d'évaluation (Velazco et Franco, 2007) sont :

- les tests en environnement réel;
- les tests accélérés;
- les techniques analytiques;
- l'injection matérielle/logicielle des pannes.

### **A. Les tests en environnements réels**

Ils sont appelés aussi *life tests*. Ce type de test consiste à exposer le composant électronique dans son milieu de fonctionnement final. Pour assurer des résultats fiables, plusieurs prototypes sont utilisés. Les *life tests* offrent des résultats réalistes et fiables mais ils sont coûteux et demandent beaucoup de temps de préparation. Plusieurs expériences ont été menées pour tester les composants électroniques dans l'environnement réel qui peut être spatial, atmosphérique ou au niveau de sol.

#### **1) Les tests en orbite**

Des agences spatiales, dans le cadre de leurs projets, ont validé la robustesse de leurs composants en ayant recours à des tests des composants électroniques dans un milieu radiatif sévère, à savoir le milieu spatial. Plusieurs expériences ont été documentées et publiées (Caffrey et al., 2009; Duzellier et al., 1997; Falguere, Duzellier et Ecoffet, 1994; Quinn et al., 2012). Le laboratoire TIMA, à Grenoble en France, avec qui nous collaborons dans le cadre de cette thèse ne manque pas d'expertise dans ce type de tests. En effet, il a marqué sa présence dans les deux projets suivants :

- 1) Micro-electronic and Photonic TestBed (MPTB): Ce projet conduit par le Naval Research Lab comportait 24 expériences testées au niveau de l'orbite. Une de ces expériences développée au sein du laboratoire TIMA avait pour objectif de tester des réseaux de neurones artificiels (Velazco, Cheynet et Ecoffet, 1998; Velazco et al., 2000).

2) Space Technology Research Vehicles (STRV): TIMA a développé une expérience sur la logique floue qui a été embarquée dans un satellite britannique du *Defence Research Agency*. Cette expérience n'a pas donné de résultats dû à un problème de communication. Les tests en orbites sont très coûteux et aussi très longs. De trois à cinq années peuvent effectivement s'écouler avant d'obtenir des résultats expérimentaux.

## **2) Les tests en haute altitude**

Ce niveau de test est destiné généralement pour les circuits avioniques. Les composants sont implémentés à bord des avions ou envoyés aux altitudes des vols commerciaux en utilisant des ballons stratosphériques. Vu que le domaine aéronautique est un domaine critique, on trouve plusieurs travaux (Chee, Braby et Conroy, 2000; Goldhagen, 2000; Normand, 2001; Peronnard, Velazco et Hubert, 2009) sur ce type de test dans la littérature.

## **3) Les tests au niveau du sol (Lesea et al., 2005)**

Ce type de test consiste à placer plusieurs composants électroniques dans différents emplacements au niveau du sol en changeant toutefois les altitudes et les latitudes afin d'observer l'influence de ces deux derniers sur la génération des SEU dans les circuits électroniques. Dans ce contexte, Xilinx a dirigé un projet nommé Rosetta (Lesea et al., 2005). Ils ont développé des plateformes de tests qui se composent des FPGA Virtex 1 jusqu'à Virtex 5 et les ont placés dans des endroits de différentes latitudes et altitudes. La plateforme de test de Rosetta est illustrée dans la figure 1.7 et le tableau 1.2 donne la liste des endroits où Rosetta a été installée.

Quelques résultats de Rosetta sont fournis dans (Lesea et Fabula, 2008). Les taux d'erreurs dans la mémoire de configuration et le block RAM des FPGA Virtex-II, Virtex-II Pro, Virtex-4 et Virtex-5 sont données. La comparaison de ces derniers avec les résultats obtenus par la simulation et les tests accélérés suggère que des améliorations doivent être apportées à ces techniques afin d'obtenir de meilleures estimations.

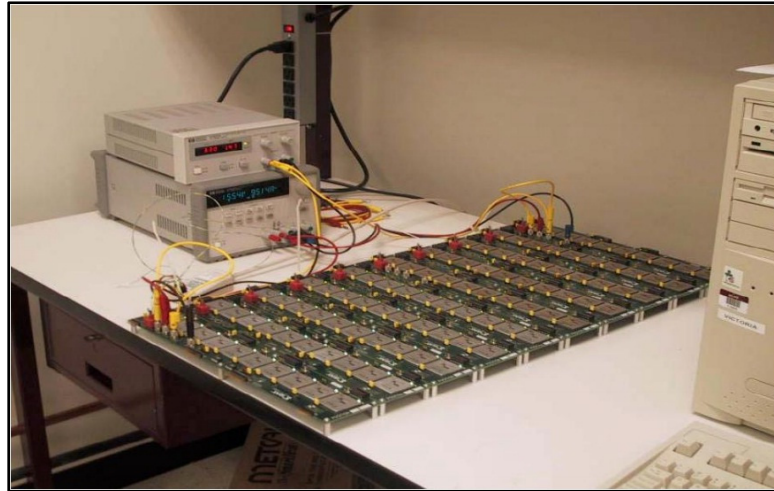


Figure 1.7 Plateforme de test Rosetta  
Tirée de Foucard (2010)

Tableau 1.2 Sites adoptés pour l'expérience Rosetta  
Adapté de Foucard (2010)

Établissement d'accueil	Emplacement	Altitude (m)
Xilinx SJ	San José, CA, USA	0
Xilinx ABQ	Albuquerque, NM, USA	1554
White Mountain Research Station	White Mountain, CA, USA	750
Caltech Submillimeter Observatory	Mauna Kea, Hawaii, USA	975
Institut Matériaux Microélectronique Nanoscience de Provence	Marseille, France	124
Institut de Radioastronomie Millimétrique	Pic de Bure, France	2552
Laboratoire Souterrain Bas Bruit de Rustrel- Pays d'Apt	Rustrel, France	-550

## B. Les tests accélérés

Le principe de ce type de test consiste à augmenter la densité des particules qui entrent en collision avec les composants électroniques testés afin de reproduire l'effet des particules dans un environnement réel tout en gagnant du temps. Cette technique est adoptée pour les applications industrielles malgré ses inconvénients, à savoir la préparation complexe des



tests, la différence entre les particules émises et les particules naturelles en termes de composition et de spectre d'énergie et le nombre restreint d'établissements de tests dans le monde. Généralement, les tests accélérés sont élaborés en utilisant des accélérateurs de particules ou des faisceaux laser.

### 1) Les accélérateurs de particules

Ces accélérateurs sont capables de générer plusieurs faisceaux de particules énergétiques telles que les ions lourds, les neutrons et les protons. Ils peuvent produire des faisceaux avec des énergies variables. Il y a différents types d'accélérateurs mais les plus utilisés sont :

- **les accélérateurs linéaires** où les particules sont accélérées sous l'effet d'un champ électrique, tels les accélérateurs de Van de Graaff (GANIL et IPN en France). La figure 1.8(a) montre le schéma d'un accélérateur de particules linéaire;
- **les accélérateurs circulaires** où les particules sont accélérées sous l'effet d'un champ électrique et un champ magnétique à la fois : les accélérateurs de type cyclotron (TRIUMF au Canada et HIF en Belgique). Un aperçu d'un accélérateur circulaire est donné par la figure 1.8(b).

Les accélérateurs circulaires offrent des niveaux d'énergie plus élevés que ceux donnés par les accélérateurs linéaires et par la suite une capacité de pénétration de particules dans le silicium plus importante.

Dans le cadre du projet AVIO403, deux campagnes de test sous neutrons ont été effectuées au laboratoire TRIUMF. Les résultats de ces deux campagnes ont servi comme un moyen de validation des approches présentées dans cette thèse.

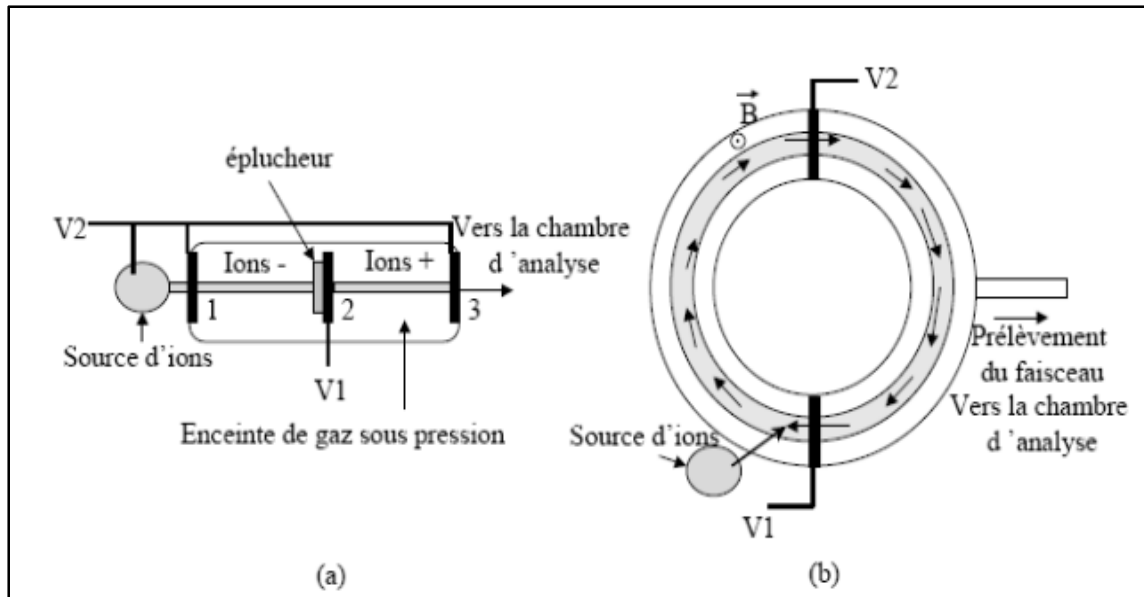


Figure 1.8 Principe d'un accélérateur (a) linéaire de type Van de Graaf (b) circulaire  
Tirée de Bocquillon (2009)

## 2) Les tests sous faisceaux Laser

Les accélérateurs de particules produisent des faisceaux de particules de grands diamètres. Il est ainsi extrêmement difficile de localiser la panne. Le laser, grâce à sa précision, peut fournir une cartographie précise des zones sensibles d'un circuit électronique.

Malgré sa précision, les tests sous faisceaux laser restent incapables de fournir des énergies élevées. Aussi, la réflexion du faisceau par les couches de métallisation oblige les opérateurs à procéder à des attaques par la face arrière du composant (Pouget, 2001).

## C. Les techniques analytiques

Ces techniques (Asadi et Tahoori, 2005a; Asadi et Tahoori, 2005b; Cai et al., 2015; Ebrahimi et al., 2015a; Sterpone et Violante, 2005; Thibeault et al., 2013; Velazco et Franco, 2007) consistent à étudier et estimer la sensibilité du design dans le flot de conception avant qu'il soit implémenté dans le matériel (*hardware*). Elles permettent la prédiction de la probabilité des SEU à partir des ressources utilisées ainsi que l'identification des sections de circuit nécessitant ou non un durcissement, ce qui assure une réduction de la taille finale du design.

## **D. L'injection matérielle/logicielle des pannes**

La technique d'injection de pannes est une technique efficace pour évaluer la sensibilité des composants électroniques. Elle permet de donner une estimation de la fiabilité des composants à l'égard des pannes injectées.

Plusieurs approches d'injection de pannes sont présentées dans la littérature (Civera et al., 2001; Jenn et al., 1994; Leveugle, 2000; Mueller-Gritschneider et al., 2014; Robache et al., 2013; Ziade, Ayoubi et Velazco, 2004). Une classification est faite pour ces différentes approches selon le niveau d'abstraction dans le flot de conception où l'injection des pannes est effectuée (Vanhouwaert, 2008). En effet, plusieurs approches permettent d'injecter des pannes dans le circuit déjà fabriqué; essentiellement, on peut citer l'injection des pannes au niveau des broches du circuit, par corruption de la mémoire ou par perturbation de l'alimentation. Pour la prédiction antérieure de la réaction d'un circuit sujet aux radiations, une injection de pannes dans des niveaux d'abstraction élevés dans le processus de conception d'un circuit intégré est établie. Les deux approches principales adoptées pour cette catégorie d'injection des pannes (la prédiction antérieure) sont la simulation et l'émulation.

Dans ce qui suit, les trois approches d'injection matérielle/logicielle des pannes sont présentées, à savoir :

- 1) approches physiques d'injection des pannes;
- 2) injection des pannes par simulation;
- 3) injection des pannes par émulation.

### **1) Approches physiques d'injection des pannes**

Ces approches consistent à utiliser le composant physique pour y injecter des pannes. Différentes approches associées à cette catégorie sont parues dans la littérature :

- **Injection des pannes à travers les broches d'entrées/sorties:** Cette approche est avantageuse parce qu'elle n'influence pas le fonctionnement temporel du circuit. Le développement des modèles de pannes ainsi que le bruit généré par la ou les sondes

représentent les deux grands problèmes rencontrés par cette technique. Plusieurs outils ont été développés pour faciliter l'injection des pannes au niveau des broches. MESSALINE (Arlat, 1990) est considéré comme le premier outil dans cette catégorie. La panne est injectée directement sur l'une des bornes du circuit intégré où une sonde force un niveau logique bas ou haut dans la broche sélectionnée. Il a été développé au sein du laboratoire LAAS en France. Cet outil a été amélioré par la suite pour donner naissance à d'autres versions comme RIFLE (Madeira et al., 1994) et AFIT (Martínez et al., 1999).

Actuellement, considérant la complexité croissante des circuits intégrés, cette méthode est très limitée et est utilisée seulement pour tester les aspects externes de fiabilité, tels les vibrations et le bruit électrique, mais elle n'est plus utilisée pour l'injection par émulation des SEU (Entrena et al., 2011).

- **Corruption de la mémoire (Michel, 1994):** Cette approche est destinée à l'injection des pannes dans la mémoire programme des processeurs. Habituellement, DEFI (Michel et al., 1994), un injecteur de pannes commercial, est utilisé pour ce type de test. L'injection des pannes à travers la corruption de la mémoire peut être assurée d'une manière matérielle en utilisant du matériel spécifique, telles que les sondes ou d'une manière logicielle à l'aide de l'insertion des segments du code. Cette approche d'injection des pannes perturbe le comportement de quelques composants du circuit cible tels que les registres et les mémoires.
- **Perturbation de l'alimentation (Karlsson et al., 1991):** Cette approche consiste à ajouter un transistor MOS entre la source d'alimentation et le nœud  $V_{CC}$  afin d'injecter des pannes tout en agissant sur la tension de grille de ce transistor. Cette approche est similaire au phénomène des chutes de tension qui cause des fautes transitoires.

## 2) Injection des pannes par simulation

Cette approche consiste à injecter des pannes aux différents niveaux d'abstraction dans la procédure de conception des circuits intégrés. Dans ce qui suit, les différentes approches d'injection des pannes à base de simulation selon chaque niveau d'abstraction sont présentées.

- **Simulation des pannes au niveau système** (Bolchini et al., 2001; Lajolo et al., 2000; Lu et Radetzki, 2011; Robache et al., 2013; Rothbart et al., 2005): l'idée consiste à analyser le plus tôt possible dans le processus de conception la fiabilité des systèmes. L'objectif est de minimiser le risque des pannes transitoires en modifiant les choix de design. Pour mettre en évidence cette technique d'injection des pannes, les langages de co-design, tel que SystemC, sont adoptés vu qu'ils représentent l'environnement de choix idéal pour la conception matérielle/logicielle. Cette approche est destinée essentiellement aux *Systems-on-Chip* (SoC). En effet, les modèles de pannes développés ciblent des blocs indépendants dans la puce. Malgré l'avantage de la vitesse de simulation, l'inconvénient majeur de cette approche est le manque de précision de l'estimation de la sensibilité des systèmes cibles.
- **Simulation des pannes au niveau RTL** (Berrojo et al., 2002; Bombieri, Fummi et Guarnieri, 2011; Chen, Mishra et Kalita, 2012; Corno et al., 2000): l'injection des pannes se fait lors de la simulation RTL selon deux approches: la modification de la description RTL ou le forçage de quelques signaux internes à des valeurs bien définies en utilisant les commandes offertes par l'outil de simulation. Les outils d'injection des pannes en se basant sur les commandes des simulateurs sont nombreux, tels que MEFISTO (*Multi-level Error/Fault Injection Simulation Tool*) (Jenn et al., 1994). On peut citer aussi un autre outil développé par l'université de Turin qui permet la simulation des pannes en se basant sur le langage TCL (*Tool Command Language*) (Corno et al., 2000). Le principe d'injection des pannes de (Jenn et al., 1994) consiste à arrêter la simulation dans un cycle, nommé cycle d'injection des pannes, en utilisant la commande *wait* du VHDL, puis le signal où on veut injecter la panne est choisi, déconnecté de sa source et enfin forcé à la valeur souhaitée pendant une durée bien définie. Dans le cas d'injection des *bit flips* dans une variable, on procède de la même manière. Pour injecter les pannes en modifiant la description RTL, le laboratoire LAAS présente l'outil MEFISTO-L (Boué, Pétilion et Crouzet, 1998) permettant d'ajouter des saboteurs et des sondes afin d'assurer les modifications nécessaires sur les signaux cibles et de contrôler la réaction du design. L'avantage de cette approche consiste à estimer tôt dans le flot de conception la vulnérabilité du design. Son inconvénient majeur est que la couverture des pannes donnée

par cette approche n'est pas habituellement en concordance avec la couverture des pannes pour des niveaux logiques plus bas (Thaker, Agrawal et Zaghoul, 2000).

- **Simulation des pannes au niveau des portes logiques:** Le principe de cette approche est similaire à celle au niveau RTL. Plusieurs outils existants sont capables d'injecter des pannes au niveau des portes logiques. L'outil FAST (*Fault Simulator for Transients*) (Cha et al., 1996) se compose de deux simulateurs; le premier assure la simulation jusqu'à l'injection de la panne tandis que le deuxième simulateur est utilisé pour observer l'effet de la panne sur les sorties. VERIFY (Sieh, Tschache et Balbach, 1997) (*VHDL-based Evaluation of Reliability by Injecting Faults efficiently*) est aussi un outil d'injection des pannes non seulement au niveau portes logiques mais aussi au niveau RTL. Un autre outil intéressant appelé ROBAN (Vanhouwaert, 2008) et développé par *iRoc technologies* sert à injecter des pannes dans les circuits combinatoires afin de faire une analyse probabiliste sur la fréquence des erreurs transitoires dans les registres du circuit. Plusieurs travaux récents (Amaricai et al., 2014; Sootkaneung et Saluja, 2010) adoptent l'injection des pannes au niveau des portes logiques pour évaluer la robustesse de leurs designs. Cette approche est plus précise que la simulation au niveau RTL dans la modélisation des pannes à simuler mais elle est lente quand appliquée sur les circuits intégrés modernes de larges tailles.
- **Simulation des pannes au niveau transistor :** Cette approche consiste à injecter des pannes dans le *netlist* d'un circuit (Sunter, 2015). Un outil développé à l'université de Téhéran nommé INJECT (Zarandi, Miremadi et Ejlali, 2003) sert à injecter des pannes dans la description au niveau transistor pour en extraire un modèle de défaut capable d'être simulé par ModelSim. L'inconvénient majeur de l'injection des pannes au niveau transistor est le temps long de simulation.

### 3) Injection des pannes par émulation

L'apparition des circuits reconfigurables, tels les CPLD et les FPGA, amène au monde des composants électroniques des caractéristiques qui rendent leur manipulation plus facile et plus flexible. Parmi les caractéristiques les plus importants des FPGA, on cite la reconfigurabilité qui rend possible l'injection des pannes par émulation (Ziade, Ayoubi et

Velazco, 2004). De plus, grâce à leur rapidité, performance et faible coût, l'injection des pannes à base des FPGA devient la technique la plus utilisée de nos jours (Khatri et al.). Dans l'approche d'injection des pannes par émulation à base des FPGA, on distingue principalement, selon (Khatri et al.), l'injection des pannes par instrumentation et l'injection des pannes par reconfiguration.

Dans ce qui suit, les deux techniques sont décrites.

### - Injection des pannes par instrumentation

Le principe de l'injection des pannes par instrumentation consiste à ajouter des circuits additionnels au circuit original dans le système ciblé.

L'instrumentation des bascules (Bushnell et Agrawal, 2000; Juneau, 2010), nommée aussi méthode SCAN, est l'une des méthodes d'injection des pannes par instrumentation. Elle consiste à ajouter des registres à décalage et de la logique combinatoire afin de faciliter l'injection des pannes. En effet, les bascules sont liées pour former un registre à décalage avec un multiplexeur à l'entrée de chaque bascule (Figure 1.9).

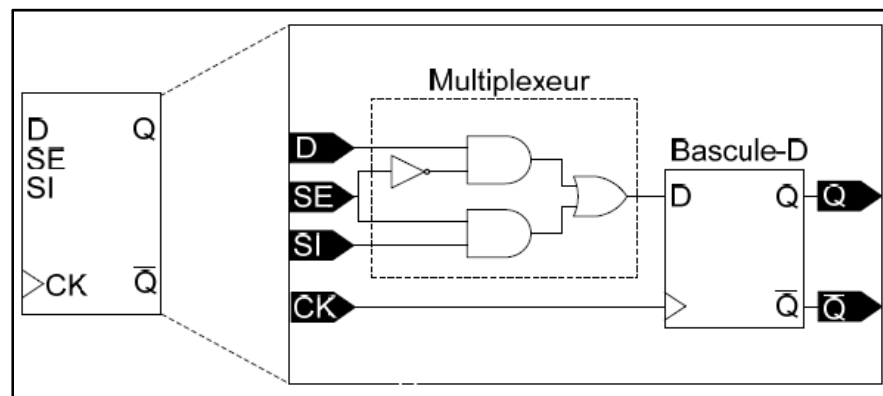


Figure 1.9 Bascule D avec SCAN  
Tirée de Juneau (2010)

La méthode SCAN (Figure 1.10) consiste à activer la broche scan puis insérer le vecteur de test à travers la broche scan in. La lecture de l'état des bascules se fait sur la broche de sortie scan out (Bushnell et Agrawal, 2000).

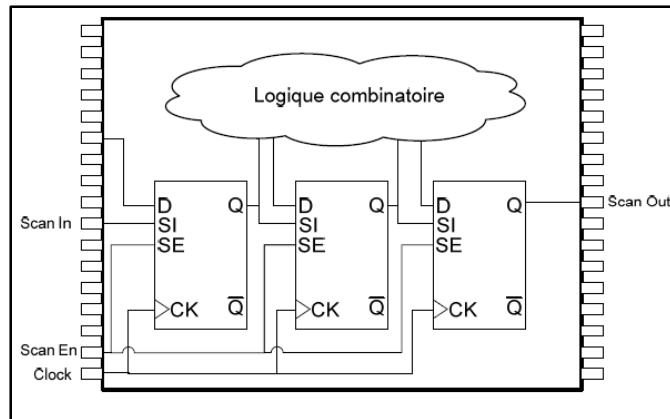


Figure 1.10 Chaîne SCAN  
Tirée de Juneau (2010)

Cette méthode, malgré sa bonne couverture de pannes (Huang et al., 2008), souffre de plusieurs limitations (Wu et al., 2009). Elle est gourmande en termes des ressources utilisées à cause du déploiement des multiplexeurs additionnels, les ressources d'interconnexions entre les bascules ainsi que la circuiterie de contrôle. Elle est caractérisée aussi par sa lenteur vu la nature sérielle de la propagation de l'information ainsi que le temps de test dépend de la longueur des chaînes de balayage.

#### - Injection des pannes par reconfiguration

Le principe de l'injection des pannes par reconfiguration consiste à injecter les pannes en changeant le fichier de configuration (*bitstream*) nécessaire pour la configuration du FPGA. Ceci dit, le FPGA doit être reconfiguré à chaque injection de panne.

Principalement, il existe deux types de configurations des FPGA à base de mémoire SRAM à savoir la configuration statique et la configuration dynamique. La première consiste à re-synthétiser le design et reconfigurer le FPGA à chaque fois où on veut apporter une modification. La deuxième technique permet de reconfigurer le FPGA totalement ou partiellement pendant l'exécution de l'application. La configuration dynamique offre un temps de reconfiguration plus court que celui offert par la reconfiguration statique.



Plusieurs approches (Antoni, Leveugle et Fehér, 2003; Di Carlo et al., 2014; Ghaffari et al., 2014; Gokhale et al., 2006; Gosheblagh et Mohammadi, 2013; Sterpone et Violante, 2007) utilisent la reconfiguration dynamique partielle pour injecter des pannes. Cette fonctionnalité offerte par les FPGA modernes à base de mémoire SRAM permet de changer dynamiquement des portions du circuit alors que le reste du design reste inchangé et totalement fonctionnel. Principalement, à chaque injection de panne une trame de la mémoire de configuration est lue par l'infrastructure de test adopté, modifiée selon le modèle de panne choisi et réécrite dans la mémoire de configuration.

L'injection des pannes par reconfiguration ainsi que différentes plateformes d'injection seront détaillées à la section 1.3.1. En fait, cette technique est celle adaptée dans le cadre de cette thèse pour injecter les pannes.

#### **1.2.4 Architecture des FPGA à base de SRAM**

Les FPGA sont des circuits intégrés programmables pouvant être reconfigurables ce qui les rend les circuits électroniques les plus adéquats à assurer un prototypage rapide et par la suite réduire considérablement les temps de conception. Quatre catégories de FPGA peuvent être distinguées selon leurs architectures internes :

- 1) architecture en tableau symétrique (*Symmetrical Array*);
- 2) architecture en colonnes (*Row-based*);
- 3) architecture en mer de portes (*Sea of Gates*);
- 4) architecture en PLD hiérarchiques (*Hierarchical PLD*).

Malgré les différences dans l'architecture, toutes ces catégories sont constituées des mêmes éléments de base définissant un FPGA, tel que montrée par la figure 1.11:

- les blocs logiques;
- les blocs d'entrées/sorties;
- les réseaux d'interconnexion.

Tous ces éléments sont programmables et assurent l'implémentation des différents circuits dans le FPGA.

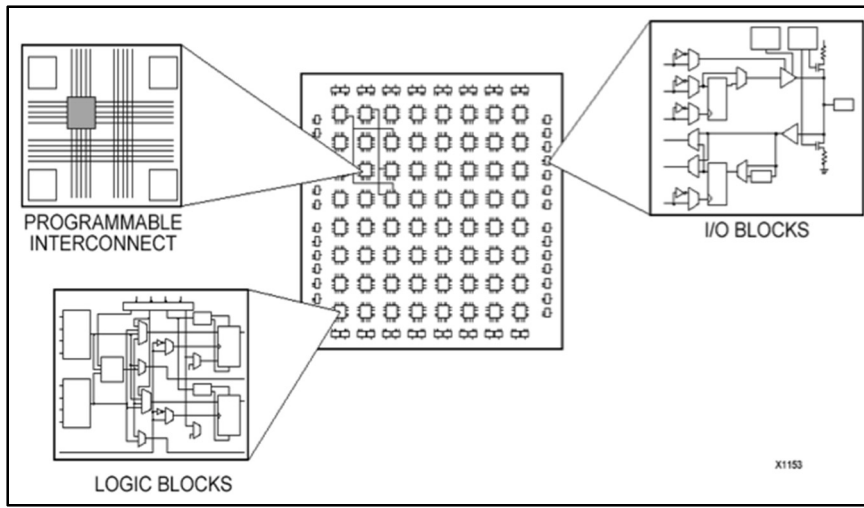


Figure 1.11 Éléments internes du FPGA  
Tirée de National Instruments (2012)

Trois principales technologies sont utilisées pour assurer la fabrication des circuits FPGAs :

### 1) FPGA à base de mémoire SRAM (*Static Random Access Memory*)

Ce type de FPGA permet de sauvegarder les données de configuration obtenu à partir d'un fichier de configuration (*bitstream*) dans des cellules mémoires à base de SRAM. La nature de ces dernières fait que les FPGA à base de mémoire SRAM sont volatiles et ne conservent leurs données de configuration que s'ils sont sous tension. Une cellule SRAM de base, donnée par la figure 1.12, est composée par 6 transistors.

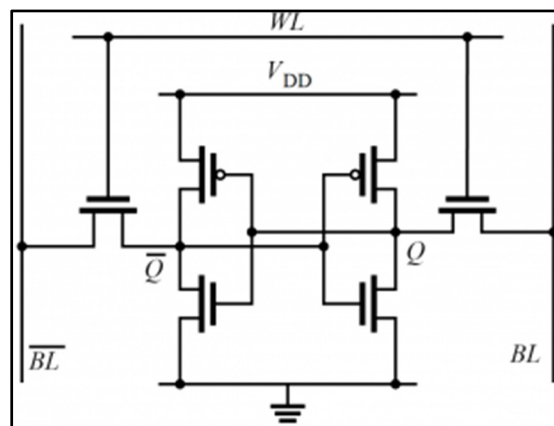


Figure 1.12 Architecture d'une cellule SRAM, d'un FPGA  
Tirée de Barth (2015)

Les deux inverseurs en tête-bêche servent à conserver la valeur du bit mémoire tant que la cellule est alimentée. Les données sont écrites ou lues à partir d'une cellule SRAM grâce aux transistors Q et Qb. Le WL active toutes les cellules mémoires cibles sur la même ligne d'adresse;

## **2) FPGA à base de mémoire Flash**

Tel que son nom l'indique, ce type de FPGA utilise la mémoire Flash pour sauvegarder les données de configuration. Cette technologie a l'avantage d'être moins gourmande en termes de consommation d'énergie ainsi que plus tolérante face aux radiations en comparaison aux FPGA à base de mémoire SRAM. Ces circuits sont reprogrammables et non-volatiles. Ce qui veut dire qu'ils sont capables de garder leurs configurations même quand ils sont hors tension;

## **3) FPGA à base d'anti-fusibles**

Ces FPGA diffèrent de ceux précédemment mentionnés vu qu'ils ne sont programmés qu'une seule fois. L'anti-fusible est un élément qui ne conduit le courant que s'il est court-circuité. Par la suite, le FPGA à base d'anti-fusible ne peut pas être reprogrammée vu qu'il n'y a aucune méthode permettant de remettre un anti-fusible court-circuité à son état initial. Ce type de FPGA est plus robuste face aux radiations quand comparé aux FPGA à base de mémoire SRAM ou Flash.

Dans le cadre du projet AVIO403, le FPGA choisi comme cible d'expérimentations est le FPGA à base de mémoire SRAM (principalement Virtex-5 et Artix-7 de Xilinx). Le choix de ce type de FPGA revient à leur rapidité et capacité d'être reconfiguré sur site, ce qui incite les industriels en aérospatial à les adopter dans leurs projets. Les deux inconvénients majeurs de ce type de FPGA sont la sensibilité aux radiations et la nécessité de reconfiguration à chaque coupure d'alimentation.

Afin de faciliter la compréhension du FPGA à base de SRAM, on peut le décomposer en deux couches (Figure 1.13):

### 1) Couche opérative (applicative)

Cette couche a pour rôle de réaliser les fonctions logiques (combinatoires ou séquentielles) de l'application à implémenter. Elle est constituée essentiellement des blocs logiques configurables, nommés CLB pour les FPGA de Xilinx. Ces derniers sont généralement entourés par des blocs d'entrées/sorties, nommés IOB pour les FPGA de Xilinx, afin de faciliter l'interfaçage des circuits implémentés à l'intérieur du FPGA aux modules externes.

### 2) Couche de configuration

Cette couche consiste en cellules mémoire SRAM permettant d'assurer l'activation des ressources de la couche opérative. En effet, cette couche permet d'interconnecter les CLB et les IOB afin de véhiculer les signaux générés par l'application implémentée dans le FPGA.

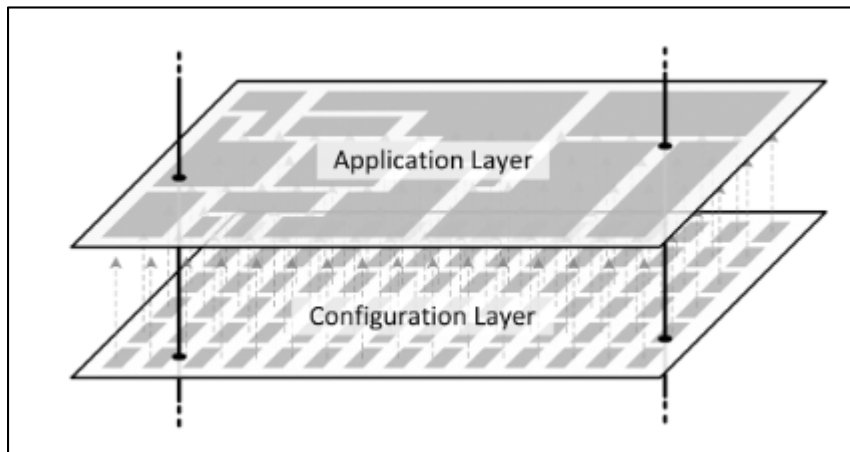


Figure 1.13 Couche opérative et couche de configuration du FPGA  
Adaptée de Herrera-Alzu et Lopez-Vallejo (2013)

## Couche opérative

### 1) CLB

Les blocs logiques de configuration sont les éléments de base du FPGA. Ils sont constitués essentiellement de LUT (*Look Up Tables*) qui sont les plus petites entités configurables du FPGA permettant d'implémenter des fonctions combinatoires.

Les CLB contiennent, en plus des LUT, des bascules, des multiplexeurs, des circuits dédiés aux opérations arithmétiques et des portes logiques.

La figure 1.14 montre l'architecture d'un CLB du FPGA Virtex-5. Ce dernier est composé de deux slices identiques où chacun contient 4 LUT à 6 entrées et 4 bascules ainsi que des multiplexeurs et portes logiques. Ces éléments sont utilisés pour assurer des fonctions logiques, arithmétiques et de mémoire ROM. Dans les nouvelles générations des FPGA, le nombre des CLB augmente considérablement ce qui rend ces circuits capables d'abriter des designs très complexes.

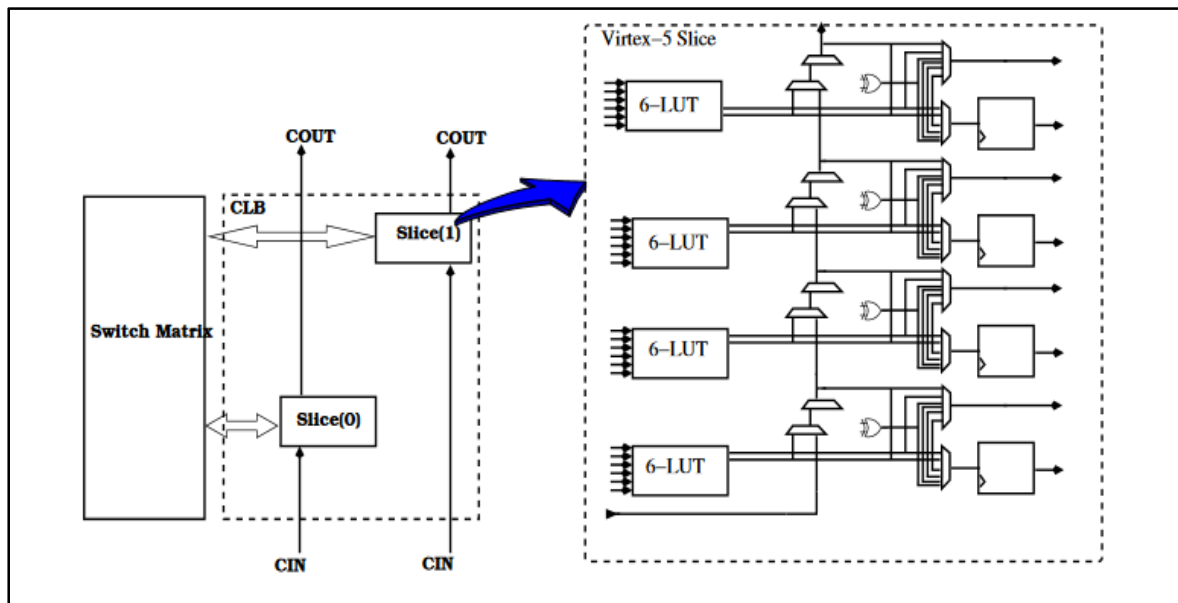


Figure 1.14 Architecture interne d'un CLB d'un FPGA Virtex-5  
Tirée de UG190 (2009)

## 2) IOB

Les blocs d'entrée/sorties (IOB) sont les éléments de base permettant l'interfaçage des ports internes du circuit implémenté à l'intérieur du design aux composants externes. Chaque IOB peut être configuré comme entrée, sortie, entrée et sortie, ou inutilisé. Tel que montré par la figure 1.15, un bloc d'entrée/sortie est composé entre autres de buffers d'entrées et sorties d'impédance programmable, de bascules, un contrôleur de temps de transition (*slew rate controller*), de résistances et d'un bloc de délai programmable.

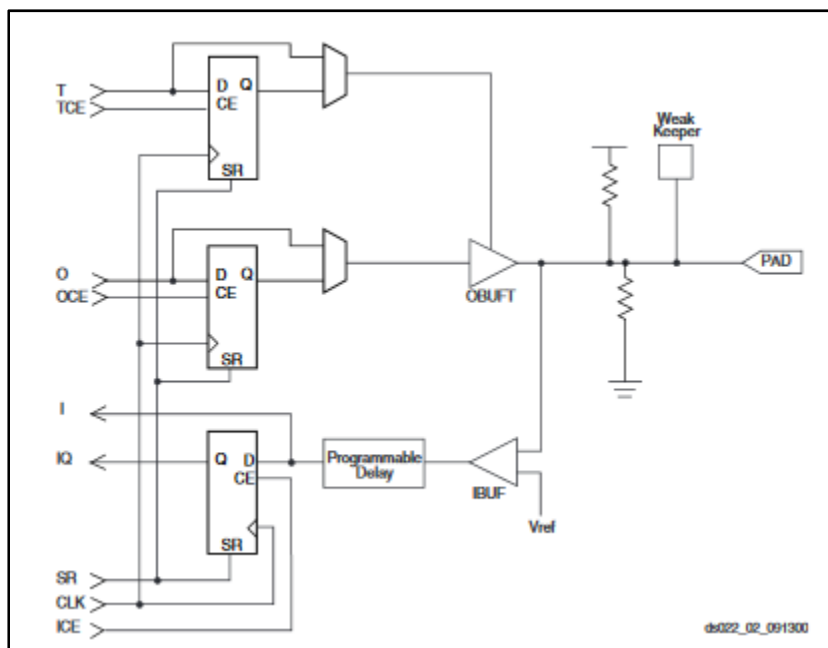


Figure 1.15 Architecture interne d'un IOB  
Tirée de Datasheet (2002)

### Couche de configuration

La couche de configuration consiste en mémoire de configuration du FPGA. Cette dernière est constituée de millions de cellules SRAM. L'architecture des cellules mémoire SRAM dédiée à la configuration est différente de celle d'une cellule classique (donnée par la figure 1.12). En effet, celle dédiée à la configuration consiste à deux inverseurs en têtes bêtes et un seul transistor d'accès permettant l'accès aux inverseurs afin d'assurer les opérations d'écriture (configuration) ou de lecture (readback) (Bocquillon, 2009).

La figure 1.16 montre l'architecture d'une cellule SRAM dédiée à la configuration. Elle est essentiellement constituée de 5 transistors sachant que le contenu de la cellule sert à commander un sixième transistor appartenant à la couche opérative.

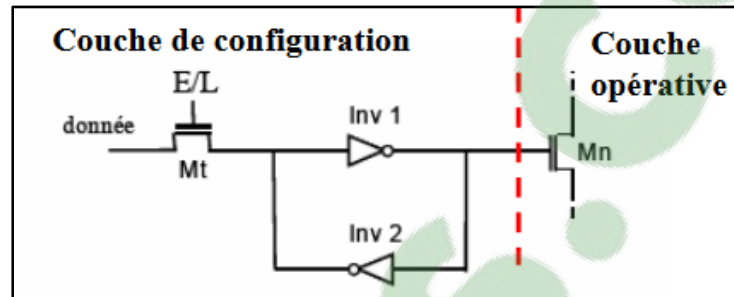


Figure 1.16 Architecture d'une cellule SRAM de configuration  
Tirée de Bocquillon (2009)

### 1.2.5 Configuration du FPGA Virtex-5

La configuration des FPGA à base de mémoire SRAM en général et du Virtex-5 en particulier se fait à travers un fichier de configuration, appelé *bitstream*, envoyé à la mémoire de configuration de la puce à travers une interface de configuration. Le *bitstream* contient les instructions contrôlant la configuration ainsi que les données de configuration.

La mémoire de configuration du FPGA Virtex-5 est divisée en colonnes élémentaires de mémoire appelés trames (*frames*) où chacune représente la plus petite entité capable d'être reconfigurée individuellement dans le FPGA. Chaque trame est composée de 1312 bits ou plus précisément de 41 mots de 32 bits. La division de la mémoire de configuration en trames adressables rend possible la reconfiguration partielle consistant à reconfigurer une ou plusieurs trames.

La configuration des FPGA de Xilinx peut se faire d'une façon externe à travers des interfaces (Virtex, 2012) telles que SelectMap ou JTAG ou d'une façon interne à travers le port de configuration interne ICAP (*internal configuration access port*). ICAP est une interface existant physiquement à l'intérieur du FPGA et permet l'accès interne à la mémoire de configuration pour effectuer des opérations d'écriture et de lecture. Le contrôle de l'ICAP

peut être effectué à travers le processeur intégré dans le FPGA. L'interface ICAP est l'interface utilisée par l'outil d'injection des pannes adopté pour les travaux de cette thèse afin d'assurer l'injection des pannes dans la mémoire de configuration. Plus de détails sont donnés au chapitre 2.

### **1.3 Mise en contexte**

Dans cette section, une revue de littérature orientée vers les objectifs particuliers définis par cette thèse est présentée. La section 1.2.3 a présenté une revue de littérature globale situant le contexte général de cette thèse. Dans cette section, les travaux en lien direct avec les nôtres sont rapportés et discutés. En effet, une exploration plus détaillée des outils et techniques d'émulation d'injection des pannes est nécessaire afin de justifier nos choix. De plus, la littérature a été explorée pour trouver les travaux concernant les méthodes d'optimisation d'injection des pannes ainsi que l'injection des pannes dans les LUT afin de distinguer nos contributions des leurs.

#### **1.3.1 Techniques d'injection des pannes**

Tel que décrit dans la section 1.2.3, plusieurs techniques d'injection des pannes existent dans la littérature pour évaluer les effets de radiations dans les FPGA. Ces catégories peuvent être classées en tests accélérés, tests sous faisceaux laser, tests d'injection des pannes par simulation, test d'injection des pannes par émulation et les techniques analytiques.

Parmi toutes ces techniques, nous nous sommes intéressés aux tests d'injection des pannes par émulation vu que cette catégorie est imposée par la définition de ce projet de thèse lui-même. En effet, la stratégie de pré-certification visée ne peut être implémentée que par cette technique, vu que c'est la phase qui vient directement avant les tests de certification, assurés d'habitude par les tests accélérés. Cette phase de pré-certification vient directement après l'implémentation du design dans le FPGA dans le flot de conception. Par conséquent, toutes les techniques de haut niveau (par simulation ou techniques analytiques) ne sont pas concernées.



Les techniques d'émulation par injection des pannes consistent à utiliser un module supplémentaire (matériel ou logiciel) dans une plateforme de test afin de contrôler la procédure d'injection des pannes. De ce fait, les plateformes de test peuvent se catégoriser selon s'ils utilisent un ou plusieurs FPGA.

### A. Plateformes de test à plusieurs FPGA

Dans cette catégorie, les principales plateformes sont : Thesic+ (Faure et al., 2002), Flipper (Alderighi et al., 2009a; Alderighi et al., 2009b; Alderighi et al., 2010; Alderighi et al., 2007), FT-UNSHADES (Guzman-Miranda, Aguirre et Tombs, 2008) et FT-UNSHADES2 (Guzmán-Miranda, Barrientos-Rojas et Aguirre, 2016; Mogollon et al., 2011).

#### 1) Thesic+

La plateforme de test Thesic+ (Faure et al., 2002) est développée au sein du laboratoire TIMA. Elle est composée essentiellement de deux FPGA à base de mémoire SRAM (Fig. 1.17). Le premier nommé COMFPGA organise la communication entre l'ordinateur contrôlant le banc laser et la plateforme de test à travers une connexion Ethernet. Il contrôle aussi la consommation en courant du DUT. Le deuxième nommé Chipset contient le fichier de configuration de l'application à implémenter dans le DUT.

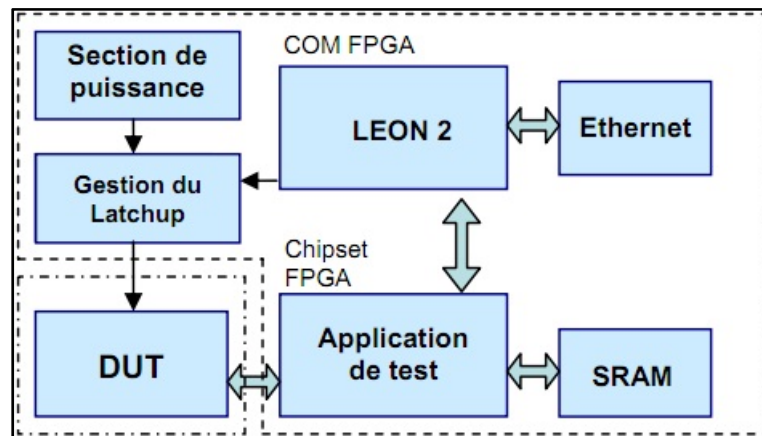


Figure 1.17 Architecture de la plateforme Thesic+  
Tirée de Faure et al. (2002)

Malgré le fait qu'elle soit générique, l'inconvénient majeur de cette plateforme est qu'elle est coûteuse vu que du matériel spécifique est déployé, entre autre deux FPGA, pour assurer le test du DUT.

## 2) Flipper

Flipper (Alderighi et al., 2009b; Alderighi et al., 2010; Alderighi et al., 2007) est une plateforme développée par l'Institut National Italien de l'Astrophysique. Il a pour objectif l'évaluation de la sensibilité des designs implémentés dans des FPGA à base de mémoires SRAM.

Flipper permet l'insertion de *bit flips* dans la mémoire de configuration du FPGA au moyen de la reconfiguration partielle. Le système (Fig. 1.18) consiste en une plateforme matérielle et une application logicielle sur PC.

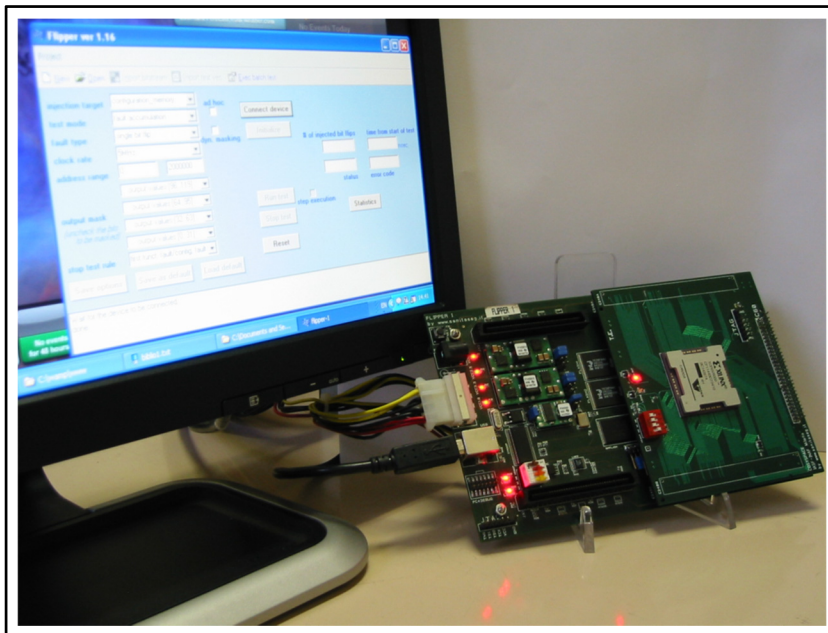


Figure 1.18 FLIPPER: plateforme matérielle/logicielle  
Tirée de Alderighi et al. (2009a)

Flipper offre :

- une caractérisation quantitative de la robustesse du design;

- une analyse de la sensibilité des bits de configuration;
- une comparaison entre différentes techniques de mitigation;
- une mise au point de la protection d'un design en comparant entre deux versions du design, avec et sans mitigation.

L'injection des pannes avec Flipper se fait par manipulation du *bitstream*. L'outil est en mesure d'injecter des SEU et des MBU par reconfiguration partielle. L'injection des MBU se fait par insertion simultanée de *bit flips* dans un nombre de bits de configuration adjacents. L'injection des pannes dans le *bitstream* se fait d'une manière aléatoire, séquentielle ou déterministe. Une injection séquentielle veut dire que l'accès aux bits de configuration à modifier se fait d'une manière séquentielle tandis que l'injection déterministe permet à l'utilisateur de choisir les emplacements d'injection via un fichier texte.

Le système se compose de trois parties principales (Figure 1.19):

- 1) une plaquette de contrôle qui gère la procédure d'injection de pannes;
- 2) une plaquette DUT qui contient le FPGA à tester;
- 3) un PC.

La plaquette de contrôle basée sur un FPGA Virtex-II Pro est connectée à la plaquette contenant le DUT. Un logiciel implémenté dans le PC communique avec le contrôleur à travers une connexion USB. Une seule copie de DUT existe et sa réponse est comparée à des valeurs connues déjà enregistrées.

Les limitations de la plateforme FLIPPER sont:

- elle ne supporte pas les différentes familles de FPGA. Elle cible principalement les FPGA des familles Virtex II ainsi que Virtex II pro et Virtex 4 de Xilinx (Alderighi et al., 2007);
- elle nécessite l'utilisation d'un matériel spécifique vu que c'est une plateforme matérielle/logicielle, ce qui la rend une solution coûteuse;
- elle est incapable de faire un diagnostic complet. En effet, l'étude de l'impact d'une panne sur le design est limitée à la classification des pannes en deux catégories; celles qui causent des erreurs aux sorties du DUT et celles sans effet (Bolchini, Castro et Miele, 2009).

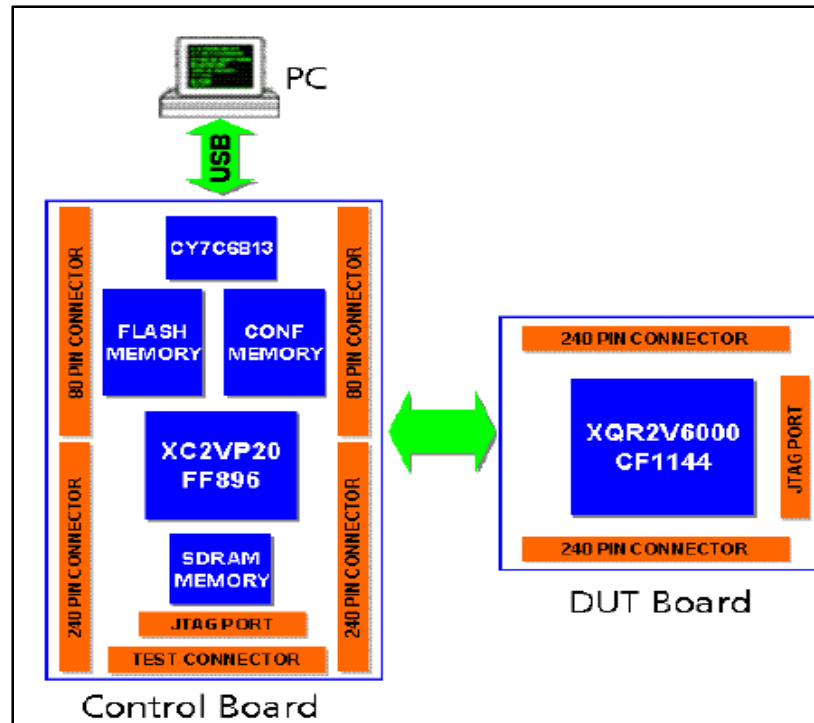


Figure 1.19 Architecture de la plateforme FLIPPER  
Tirée de ESA

### 3) FT-UNSHADES et FT-UNSHADES2

FT-UNSHADES (Guzman-Miranda, Aguirre et Tombs, 2008) est une plateforme de test à base de FPGA développée par des chercheurs de l'université de Séville et de l'Agence Spatiale Européenne. Elle est dédiée à l'étude de la fiabilité des circuits numériques via la reconfiguration partielle. FT-UNSHADES est une plateforme logicielle/matérielle basée sur un FPGA Virtex-II. Les SEU sont injectées sous forme de *bit flips* dans les registres du design.

L'architecture de test de FT-UNSHADES est donnée à la figure 1.20. Le module sous test, nommé Module Under Test (MUT), est instancié deux fois. Seule une copie sera l'objet d'injection des pannes et l'autre va servir de référence pour la comparaison. La comparaison des deux configurations se fait à chaque relecture en utilisant la commande *readback*, ce qui permet de détecter si les pannes ont été activées ou pas.

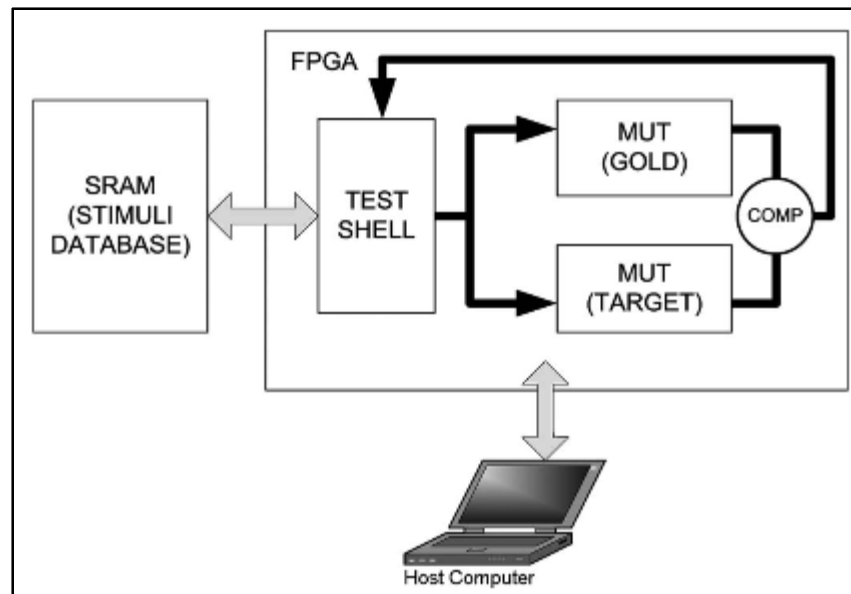


Figure 1.20 Architecture de la plateforme FT-UNSHADES  
Tirée de Guzman-Miranda, Aguirre et Tombs (2008)

Chaque cycle d'injection des pannes passe par les étapes suivantes :

- 1) le cycle d'injection est sélectionné et un ou plusieurs registres sont identifiés pour qu'ils soient modifiés quand le cycle d'injection est atteint;
- 2) le système est programmé pour qu'il arrête au cycle d'injection;
- 3) quand le système est arrêté, les *bit flips* sont insérés dans les registres sélectionnés. Le système continue son fonctionnement jusqu'à ce qu'un évènement erroné soit détecté à la sortie ou le test soit complété sans erreur;
- 4) les pannes sont classées.

Les limitations de la plateforme FT-UNSHADES sont pratiquement les mêmes que celles du FLIPPER :

- elle cible juste les FPGA à base mémoire SRAM de Xilinx;
- elle est coûteuse puisqu'elle est une plateforme matérielle/logicielle;
- l'observabilité par FT-UNSHADES est limitée à contrôler l'effet des pannes sur les sorties du DUT et les classer en deux catégories; dommage et latent (Bolchini, Castro et Miele, 2009).

FT-UNSHADES2 (Guzmán-Miranda, Barrientos-Rojas et Aguirre, 2016; Mogollon et al., 2011) est la version évoluée de FT-UNSHADES. La nouvelle plateforme est basée sur une carte mère qui peut gérer deux cartes filles contenant chacune un DUT. FT-UNSHADES2 est composée au total de 5 FPGA : un Virtex XC5VFX130T nommé CONTROL FPGA est localisé dans la carte mère, les quatre autres sont localisés dans les cartes filles où deux sont nommés SERVICE FPGA et les deux autres TARGET FPGA. Cette plateforme permet, outre l'injection des pannes dans les registres, de cibler la mémoire de configuration. La plateforme permet aussi de supporter les tests accélérés en comparant les signaux de deux copies de DUT; une exposée aux particules et une servant de référence.

Les plateformes d'injection des pannes mentionnées dans cette section, malgré leur capacité d'évaluer la sensibilité des circuits sous test, sont coûteuses (utilisent du matériel et des FPGA supplémentaires) et non disponibles (développés pour des organismes bien définis dans le cadre de projets spécifiques). De plus, certaines d'entre elles supportent des familles limitées et anciennes de FPGA. Un des objectifs implicites de cette thèse consiste à adopter un outil d'injection de pannes non coûteux et applicable sur le FPGA cible de nos travaux, à savoir le Virtex-5, ainsi que sur les générations subséquentes pour des raisons de portabilité vu que le projet AVIO403 cible aussi le Artix-7. Donc, les plateformes à plusieurs FPGA n'étaient pas une solution envisageable par nos travaux.

## **B. Plateformes de test à un seul FPGA**

Dans cette catégorie, on retrouve les plateformes développées par (Bernardi et al., 2004), (Lee et Sedaghat, 2008), (Ebrahimi et al., 2014) et (Nunes et al., 2015).

(Bernardi et al., 2004) propose une solution d'injection des pannes par émulation. Le système d'injection des pannes proposé est composé de :

- ***Fault List Manager (FLM)***: cette entité génère la liste des pannes à injecter dans le DUT.
- ***Fault Injection Manager (FIM)***: cette entité permet la gestion de la procédure d'injection des pannes. Elle sélectionne une panne parmi la liste des pannes, l'injecte

dans le DUT et enfin observe et analyse les résultats obtenus afin de classifier la panne selon l'effet généré.

La figure 1.21 donne une idée de l'architecture de la solution d'injection des pannes proposée. Le FLM est une entité logicielle implémentée sur le PC alors que le FIM est une entité logicielle/matérielle. Sa partie logicielle tourne sur le PC et sa partie matérielle est implémentée dans le FPGA avec le DUT. Les deux parties communiquent à travers un port parallèle (EPP).

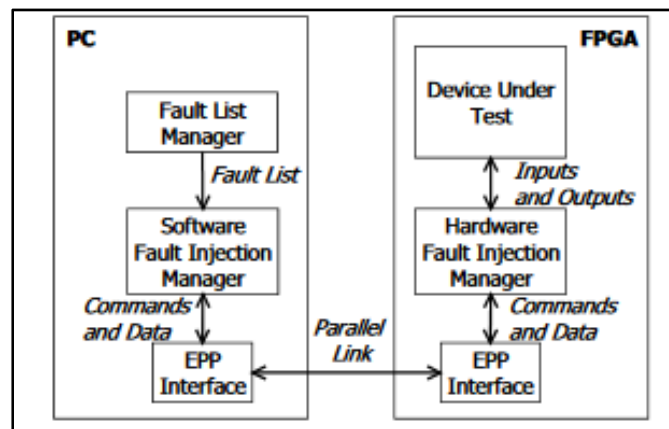


Figure 1.21 Architecture de la solution d'injection des pannes proposée  
Tirée de Bernardi et al. (2004)

Le FLM est optimisé de telle sorte qu'il considère seulement les bits utilisés par le design pour générer la liste des pannes. Ceci est fait en se basant sur une base données permettant de décoder les *bitstream* de DUT. Il n'est quasiment plus possible de créer cette base de données avec les familles récentes de FPGA de Xilinx. L'injection des pannes assurée par le FIM est faite par la génération et la configuration du FPGA par des *bitstreams* erronés générés à partir de la liste des pannes obtenue à partir du FLM.

La limitation majeure de l'approche proposée par (Bernardi et al., 2004) est la lenteur. En effet, le processus d'injection de pannes prend 6 secondes pour chaque panne injectée. Ce temps est essentiellement dû au chargement du *bitstream* erroné pour chaque panne.

Un autre outil interne d'injection des pannes a été présenté dans (Lee et Sedaghat, 2008). Les auteurs proposent une nouvelle approche d'injection des pannes basée sur la reconfiguration dynamique partielle par module. Cette approche est basée sur le partitionnement du DUT en différents modules. Les pannes sont injectées dans les partitions cibles à travers la reconfiguration dynamique sans reconfigurer les modules non ciblés.

Le système est implémenté sur un FPGA Virtex-II embarqué dans la carte de développement V2EC1000.

Pour appliquer cette approche, le circuit est divisé en trois parties telles que montrées dans la figure 1.22. La partition dynamique (DP), est la partition cible d'injection des pannes et doit avoir la capacité d'être reconfigurée dynamiquement à chaque injection de panne. La partition statique (SP) est la partition qui ne peut pas être dynamiquement reconfigurable et par la suite elle ne peut pas être la cible d'injection des pannes. Finalement, le contrôleur d'émulation des pannes (EC) sert à contrôler tout le flot d'émulation.

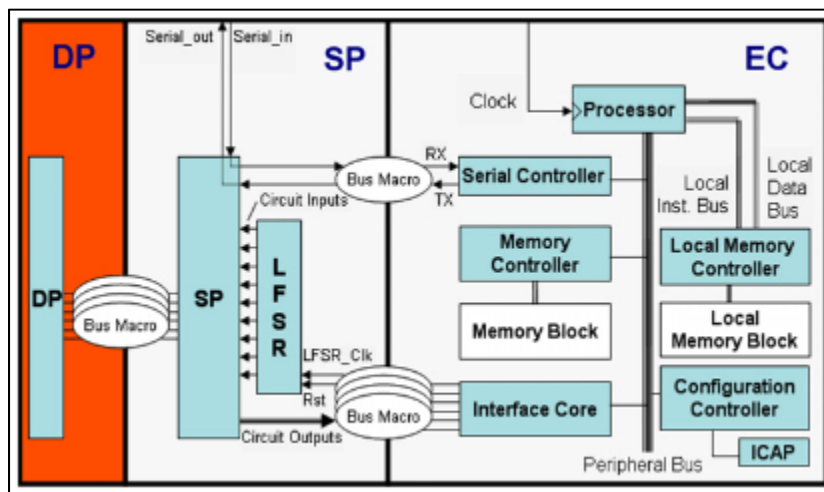


Figure 1.22 Architecture de la solution d'injection des pannes proposée  
Tirée de Lee et Sedaghat (2008)

Le contrôleur d'émulation des pannes est implémenté dans un processeur Microblaze d'architecture RISC à 32 bits. La figure 1.23 montre l'architecture du contrôleur. Une interface assurant la communication entre le DUT et le processeur ainsi qu'entre ce dernier et



le générateur des vecteurs de test est requise. Une mémoire de 16 KB contenant les instructions et les données ainsi qu'une BRAM de 32 KB pour sauvegarder les *bitstreams* partiels et les bonnes sorties du système sont fournis. L'outil permet une reconfiguration autonome à travers l'ICAP (*Internal Configuration Access Port*).

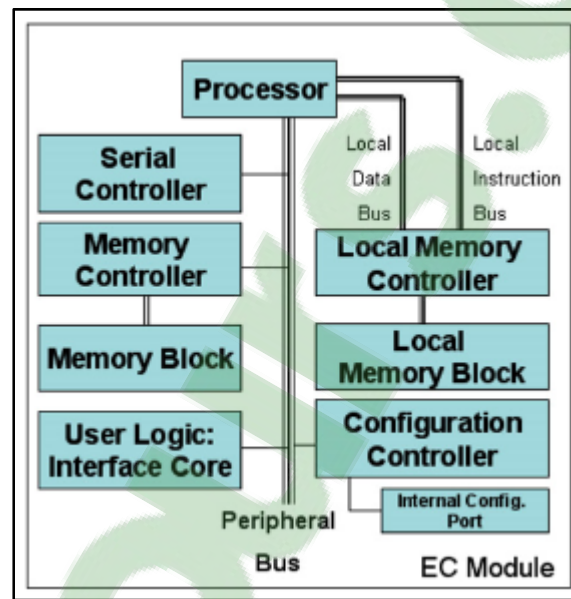


Figure 1.23 Contrôleur d'émulation des pannes  
Tirée de Lee et Sedaghat (2008)

La limitation majeure de l'approche d'injection des pannes proposée par (Lee et Sedaghat, 2008) est qu'elle n'est applicable que sur les circuits capables d'être partitionnés en modules indépendants. De plus, la partie statique du circuit sous test ne peut pas être testée ce qui minimise l'efficacité de l'approche proposée dans l'évaluation de la sensibilité des circuits sous test.

Dans (Ebrahimi et al., 2014), les auteurs proposent une nouvelle technique d'injection des pannes dans les flips-flops ainsi que dans la mémoire de configuration des FPGA d'Altera. Cette technique utilise les éléments de débogage déjà existants sur la puce et offerts par Altera afin d'injecter les SEU et les MBU. Vu que la technique proposée utilise déjà des

éléments existants dans le FPGA, la solution proposée vient avec un coût supplémentaire en termes de matériel et une dégradation négligeable de la performance du système.

La technique utilise l'infrastructure déjà existante afin d'augmenter la contrôlabilité et l'observabilité dans le design. Notamment, le *In-System Sources and Probes* (SAP) et le *In-System Memory Content Editor* (MCE) sont utilisés pour contrôler et observer les valeurs des registres de contrôle et les cellules mémoires.

La figure 1.24 montre le diagramme de la solution d'injection des pannes proposée. Les deux composants SAP et MCE sont contrôlés par un contrôleur JTAG assurant l'interfaçage entre l'infrastructure de débogage à l'intérieur du FPGA et le logiciel sur le PC commandant le test.

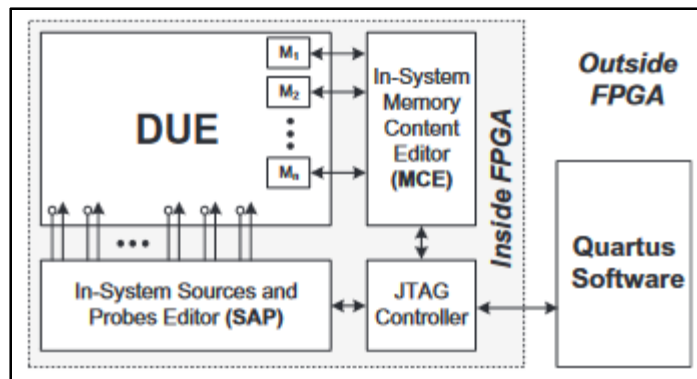


Figure 1.24 Architecture de l'injecteur de pannes proposé  
Tirée de Ebrahimi et al. (2014)

L'inconvénient de cette approche, outre le coût supplémentaire en termes des ressources utilisées, est qu'elle n'est applicable que sur les FPGA d'Altera. Cette approche ne peut donc pas être utilisée dans le cadre du projet AVIO403 en général et dans le cadre de ce projet de thèse en particulier vu que les FPGA ciblés sont ceux de Xilinx.

(Nunes et al., 2015) propose un injecteur de pannes nommé FIRED ciblant les FPGA à base de mémoire SRAM. Cet outil permet d'injecter des pannes en temps réel dans les cellules SRAM à travers la reconfiguration dynamique partielle et de contrôler le comportement du DUT en état d'exécution.

FIREDD (figure 1.25) est composé de deux composants principaux :

- 1) **Injection Runtime Controller (IRC)**: responsable de l'injection des pannes dans la mémoire de configuration du DUT. Afin d'injecter une panne dans une cellule mémoire, l'entité a accès au port de configuration interne l'ICAP (*Internal Configuration Access Port*) permettant d'assurer la reconfiguration dynamique partielle;
- 2) **Experiment Management Environment (EME)**: est l'interface d'injection des pannes s'exécutant sur un PC. Cette entité est responsable de la définition et l'exécution de l'expérimentation et l'analyse des résultats.

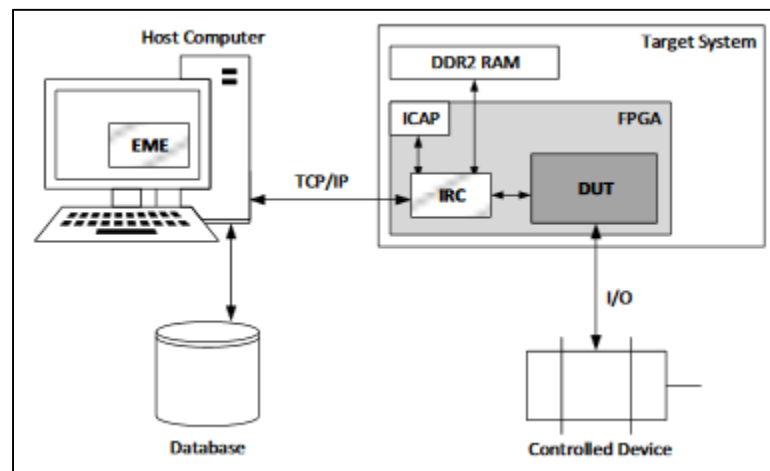


Figure 1.25 Architecture de l'injecteur des pannes FIREDD  
Tirée de Nunes et al. (2015)

FIREDD déploie une mémoire externe DDR2 RAM pour enregistrer localement les données d'entrées/sorties obtenues du module IRC afin de réduire la communication avec le module EME.

L'architecture et le principe de fonctionnement de FIREDD sont similaires à l'outil d'injection des pannes adopté pour les travaux effectués dans le cadre de cette thèse et décrit dans le chapitre 2. La valeur ajoutée de FIREDD est qu'il permet d'injecter, en plus des *bitflips*, des modèles de pannes *stuck-at-0* et *stuck-at-1*. De plus, la durée d'injection des pannes ainsi que la plage des adresses de trames à cibler peuvent être définies comme paramètres d'entrée au module EME pour définir l'expérimentation d'injection des pannes. L'avantage de l'outil que

nous avons choisi par rapport à FIRED est qu'il est fourni par le fabricant des FPGA cible de nos tests lui-même. De plus, au moment où le choix de l'outil d'injection des pannes, FIRED n'existait pas.

En général, la lenteur, la dégradation de l'efficacité d'évaluation de la sensibilité des DUT et l'inapplicabilité sur les FPGA ciblés par nos travaux rendent ces plateformes un choix inapproprié pour nous. L'outil d'injection choisi dans le cadre de cette thèse est le meilleur candidat en comparaison avec les plateformes basées sur un seul FPGA décrites dans cette sous-section. Ses caractéristiques sont détaillées dans le chapitre 2.

### 1.3.2 Optimisation de la procédure d'injection des pannes

La procédure d'injection des pannes par émulation consiste à apporter intentionnellement des modifications sur le DUT et observer son comportement en leur présence afin d'évaluer sa sensibilité. Le réalisme des résultats en comparaison avec les tests de certification et la rapidité sont des caractéristiques de performance désirables pour ce type de test. Plusieurs travaux dans la littérature visent à améliorer et optimiser la procédure d'émulation par injection des pannes dans ces deux directions.

(Faure, Velazco et Peronnard, 2005) décrit une approche statistique basée sur la loi de Poisson pour déterminer les instants exacts d'injection de SEU dans les microprocesseurs dans le but d'avoir des résultats similaires à ceux obtenus par les tests accélérés. Le DUT (*Design Under Test*) utilisé dans ce travail est un cœur de processeur SPARC V8. Les auteurs proposent, tout d'abord, un modèle, *Particle Domain Fault Injection*, basé sur le processus de Bernouilli pour déterminer le temps d'injection de fautes à partir d'une section efficace  $\sigma$  et une fluence  $F$  donnés, avec  $\sigma = N_{SEU}/F$  et  $N_{SEU}$  est le nombre de pannes détectées lors d'une expérimentation.

Cette approche cherche à ajouter du réalisme à la procédure d'injection des pannes par émulation afin de produire des résultats proches de ceux obtenus par les tests accélérés. Ceci est l'un des objectifs majeurs de la présente thèse sauf que l'approche proposée par (Faure, Velazco et Peronnard, 2005) est orientée essentiellement aux applications microprocesseurs. De plus, l'approche que nous proposons utilise la différence de sensibilité relative selon le

contenu des bits de configuration pour optimiser les résultats sans prendre en compte le temps d'injection.

Dans (Lopez-Ongil et al., 2007), les auteurs proposent un mécanisme autonome d'injection des pannes basé sur l'idée de minimiser la communication entre l'émulateur et le PC hôte durant l'exécution de l'injection des pannes et d'optimiser les tâches d'injection et de classification des pannes exécutés dans le FPGA. Ceci est accompli en insérant le contrôleur d'injection des pannes dans le FPGA cible. Ce dernier a pour fonctions de générer et sauvegarder les stimuli et la liste des pannes ainsi que de classifier les pannes selon leurs effets. Implémenter un tel système permet de limiter la communication avec le PC hôte et par la suite ce dernier n'a accès au FPGA qu'au début de processus pour envoyer le *bitstream* et vers la fin pour extraire les résultats.

En plus, afin d'assurer un gain important en termes du temps d'injection des pannes, le processus de test est arrêté dès qu'une panne est injectée et classifiée. En effet, dans plusieurs cas, l'effet d'une panne peut être observé quelques cycles après l'injection de la panne donc cette dernière peut être classifiée et le circuit peut être directement restauré à l'état qui précède l'injection de panne et donc on n'a pas besoin d'attendre jusqu'à la fin de tous les cycles du design pour recommencer l'injection. Le temps requis pour l'émulation est améliorée de deux ordres de grandeur par cette approche en comparaison avec les approches existantes.

La figure 1.26 montre la structure générale du système d'émulation proposé. La carte d'émulation contient un FPGA, une mémoire RAM et une interface au PC hôte. Le FPGA inclut le DUT, le module d'injection des pannes, le module de classification et le contrôleur d'émulation. Ce dernier est le cœur du mécanisme proposé vu qu'il gère tout le processus, à savoir, l'initialisation du design, l'application des vecteurs de test, l'injection des pannes et la comparaison entre les sorties du design référence et les sorties du design erroné.

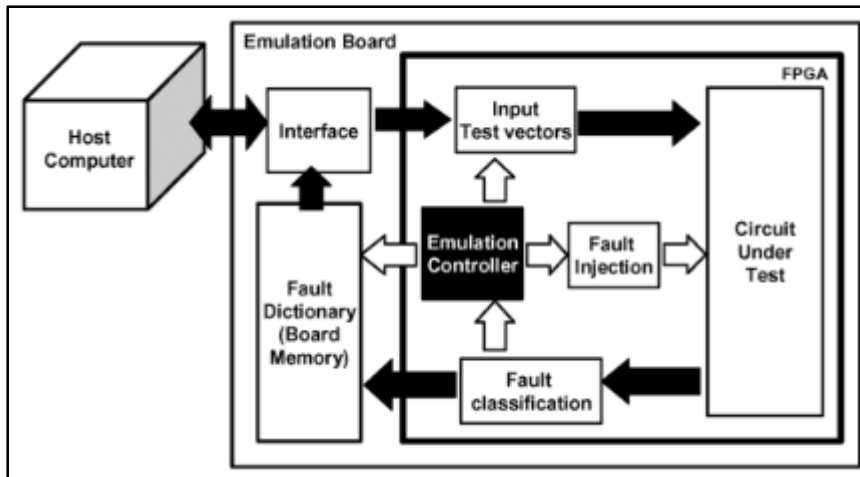


Figure 1.26 Architecture du système d'injection d'erreurs par émulation  
Tirée de Lopez-Ongil et al. (2007)

Une autre façon d'accélérer la procédure d'injection des pannes est présentée par (Cieslewski, George et Jacobs, 2010). L'idée derrière ce travail consiste à injecter plusieurs pannes au même temps d'opération. Dans les travaux précédents, un seul bit de configuration est modifié à la fois. En adoptant l'injection simultanée de plusieurs pannes jumelée à la reconfiguration partielle, la procédure d'injection peut être accélérée par plus d'un ordre de grandeur.

Afin d'implémenter leur approche, les auteurs considèrent que la probabilité de deux pannes ou plus masquant leurs effets et menant à un circuit fonctionnant correctement est infiniment petite et donc n'a aucun effet sur les résultats des tests. Pour renforcer cette hypothèse, les auteurs ont appliqués des contraintes sur les endroits où les pannes seront injectées conjointement. En effet, aucune des pannes injectées conjointement ne cible des bits occupant un même CLB.

L'idée d'injecter plusieurs pannes simultanées pour accélérer la procédure d'injection des pannes a été aussi traitée par (Ziade et al., 2011). Le DUT est divisé en plusieurs modules fonctionnels. Tout d'abord, la première étape consiste à injecter une panne dans chaque module puis d'observer et détecter le comportement erroné de chaque module. La deuxième étape consiste à observer les effets des modules erronés sur tout le système. En utilisant

l'approche proposée, il est possible d'injecter plusieurs pannes en reconfigurant le FPGA avec un seul *bitstream* erroné, ce qui accélère la procédure d'injection des pannes et diminue le temps requis par les expérimentations d'émulation.

Une autre approche d'accélération de la procédure d'injection des pannes est présentée par (Ebrahimi et al., 2015b). L'idée consiste à éviter l'injection des pannes dans des intervalles de temps non critiques où le traçage de leurs effets peut durer des millions de cycles sans la détection d'un effet à cause du masquage. Pour ce faire, une technique basée sur l'analyse *Architecturally Correct Execution* (ACE) est utilisée pour identifier les intervalles de temps non vulnérables dans les mémoires et pour éviter l'injection des pannes dans ces intervalles sans sacrifier la précision de l'évaluation de la sensibilité du DUT. L'approche proposée permet d'accélérer la procédure d'injection des pannes d'un facteur moyen de treize fois.

La plupart des travaux décrits dans cette sous-section cherchent à optimiser la procédure d'injection des pannes en termes du temps d'émulation requis tout en gardant le même niveau de précision des résultats obtenus par une injection des pannes conventionnelle. Afin de se distinguer de ces travaux et pour répondre à l'objectif ultime de cette thèse, à savoir l'élaboration d'une stratégie de pré-certification efficace, nos travaux sont orientés à rendre la procédure d'injection des pannes plus réaliste et à maximiser la précision d'estimation de la sensibilité des circuits ciblés par nos tests par rapport à la procédure conventionnelle d'injection des pannes et en comparaison avec les résultats de test de certification.

### **1.3.3 Injection des pannes dans les LUT**

Les LUT sont les plus petites entités configurables du FPGA permettant d'implémenter une fonction combinatoire. L'identification des LUT vulnérables face aux pannes injectées permet aux concepteurs de prendre les mesures nécessaires pour les protéger et par la suite augmenter la robustesse du DUT en entier. Dans ce contexte, plusieurs travaux cherchent à maximiser le taux de couverture des pannes dans les LUT sachant que l'obstacle majeur pour les familles récentes des FPGA consiste à identifier les adresses des bits configurant les LUT.

Dans (Kenterlis et al., 2006), les auteurs présentent une plateforme automatique d'injection des pannes permettant d'injecter des SEU dans les bits de configuration du FPGA. Plus spécifiquement, ils optimisent l'injection des pannes dans les LUT en évitant l'injection dans les LUT non utilisés par le design. L'identification des bits de configuration configurant les LUT ainsi que d'autres ressources est faite en utilisant l'outil Jbits (Singh et James-Roxby, 2001). Développé par Xilinx, ce dernier permet de concevoir des circuits dans un niveau élevé d'abstraction et permet l'accès aux différentes parties matérielles du FPGA. Le développement de cet outil est arrêté depuis une décennie et par la suite il ne supporte plus les nouvelles versions des FPGA de Xilinx.

Dans (Jing et al., 2011), les auteurs proposent un outil d'évaluation des SEU basé sur la simulation. Il quantifie la sensibilité des différents bits de configuration groupés selon les ressources du FPGA qu'ils configurent. L'évaluation de la criticité des bits de configuration est assurée par une analyse des pannes. Dans ce contexte, la criticité d'un bit de configuration consiste à la probabilité d'observation d'une ou plusieurs erreurs sur les sorties du DUT à cause d'un SEU dans ce bit. Les auteurs utilisent l'outil VPR afin de placer et router leurs designs. Cet outil donne accès aux plus bas niveaux des ressources du FPGA. L'inconvénient de cette approche est qu'elle ne peut pas être appliquée directement sur les FPGA commerciaux sans avoir accès à leurs architectures détaillées.

Une autre approche analytique prouvant l'existence des pannes non testables dans les LUT est proposée dans (Bernardeschi, Cassano et Domenici, 2012); en effet ces pannes ne peuvent pas être détectées soit parce que les erreurs générées ne se propagent pas vers les sorties du DUT ou parce que l'activation de ces pannes ne se fait que par une combinaison de signaux qui ne peut pas se produire et c'est cette deuxième catégorie qui a été étudiée par (Bernardeschi, Cassano et Domenici, 2012). Ce travail utilise les techniques de vérification formelle afin de prouver l'incapacité de quelques bits de configuration d'être excités.

Dans (Das, Venkataraman et Kumar, 2013), les auteurs proposent une nouvelle technique de protection des LUT en maximisant les zéros ou les uns. L'identification des bits de configuration des LUT est assurée par l'outil Rapidsmith (Lavin et al., 2011). La figure 1.27 montre le flot de l'implémentation du design modifié et proposé par les auteurs. Le flot



conventionnel adopté par les fabricants des FPGA est décrit par les boîtes en blanc tandis que les boîtes colorées montrent les étapes ajoutées permettant de protéger les LUT. La première étape LUTXtract consiste à extraire les LUT et leurs contenus à partir de la *netlist* générée par l'outil de synthèse. Dans le cas d'un flot de design de Xilinx, cette information est donnée par le fichier NCD (*Netlist Circuit description*). L'extraction des LUT est effectuée par l'outil LUTXtract de Rapidsmith. L'étape suivante consiste à la restructuration logique (LR) qui maximise le nombre de zéros ou uns dans les LUT en exploitant les XOR et XNOR des slices. Elle est suivie de la décomposition logique (LD) des LUT pour augmenter la granularité afin de faciliter le masquage de pannes. Finalement, le bloc *Resynth* modifie les portes logiques de la *netlist* en faisant les connexions nécessaires pour la génération du bitstream.

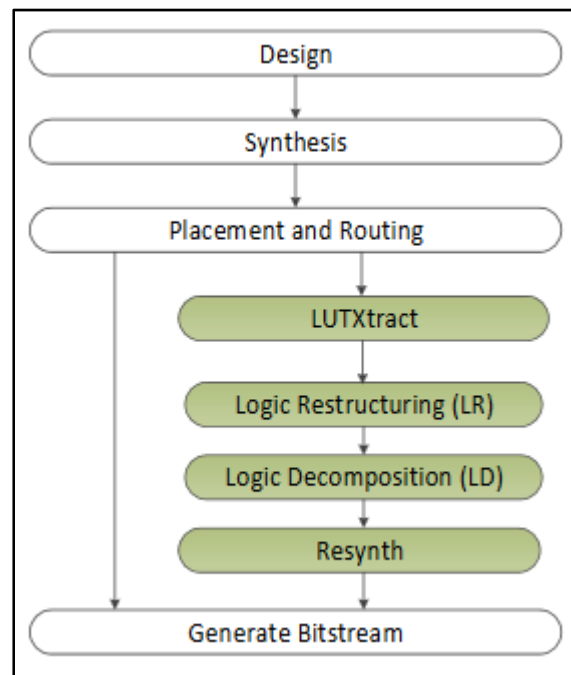


Figure 1.27 Méthodologie de protection des LUT  
Tirée de Das, Venkataraman et Kumar (2013)

À partir des travaux susmentionnés, on peut constater que les techniques d'injection des pannes dans les LUT proposées ont toujours recours à un outil spécialisé pour identifier les bits à cibler ce qui diminue leur portabilité et les rend peu flexibles. En effet, sans recours à

ces outils, il n'est pas possible de déterminer les adresses des bits des LUT. C'est dans cette perspective que dans cette thèse est proposée une nouvelle approche efficace, simple à implémenter et flexible sans aucun recours à des outils externes indispensables à l'extraction des adresses des bits des LUT. En fait, la simplicité de l'approche proposée provient de la simplicité de la méthode de détermination des bits des LUT en manipulant des fichiers générés par l'outil de synthèse lisibles par l'utilisateur humain. N'importe quel outil générique de gestion des fichiers textes peut être utilisé pour cette fin.

#### **1.4 Conclusion**

Dans ce chapitre, une revue de littérature concernant le domaine d'intérêt de la thèse a été faite à savoir les radiations et leurs effets sur les circuits intégrés en général et sur les FPGA en particulier. En premier lieu, des connaissances primordiales concernant les sources des radiations, leurs effets sur les circuits intégrés et les différentes méthodes d'évaluation de la sensibilité de ces circuits déployés dans les milieux radiatifs ont été présentées. En second lieu, une mise en contexte des objectifs de la thèse en les liants aux travaux existants dans la littérature a été faite.

L'état de l'art présenté dans ce chapitre permet de situer les travaux de la thèse dans leur cadre général du domaine de recherche étudiant l'effet des radiations sur les circuits intégrés. Ceci permet d'évaluer les techniques déjà existantes et montrer les niches d'innovations qui peuvent être explorées.

## CHAPITRE 2

### ENVIRONNEMENT D'ÉMULATION

#### 2.1 Introduction

Dans le chapitre précédent, une revue de littérature générale sur les différentes techniques d'évaluation de la sensibilité des circuits intégrés face aux radiations ainsi que sur les techniques de mitigation a été faite. Plusieurs techniques d'injection des pannes ont été détaillées. En effet, ces dernières sont variées et sont développées pour répondre aux différents besoins de tests. Elles peuvent être implémentées à l'intérieur ou à l'extérieur du DUT (*Device Under Test*), et elles peuvent être des solutions logicielles ou matérielles, requérant des plateformes spécifiques ou non. Elles dépendent aussi des circuits cibles à tester et des modèles de pannes adoptées. Plusieurs variétés de techniques d'injection de pannes se présentent donc et le choix d'un outil convenable répondant aux besoins définis par cette thèse est un défi en lui-même. Un compromis a été fait entre le coût, la simplicité d'implémentation et d'utilisation et les performances des différents outils pour choisir la solution optimale.

Ce chapitre présente l'outil d'injection des pannes adopté tout au long des travaux de la présente thèse. En effet, l'objectif ultime de cette thèse est de développer des stratégies visant essentiellement à créer un environnement de pré-certification permettant d'assurer une évaluation fiable de la robustesse des circuits cibles. L'outil d'injection des pannes constitue le noyau autour duquel les diverses stratégies élaborées ont été construites. Ce noyau doit offrir un environnement de test permettant d'émuler les effets des radiations. Les motifs de ce choix ainsi que les caractéristiques de l'outil choisi sont détaillés dans ce qui suit. En effet, le principal motif consiste à utiliser un outil simple qui nécessite un minimum d'efforts de développement et d'implémentation et offrant un environnement d'émulation convenable pour la reproduction des effets de radiations ciblés. Le *SEU Controller* de Xilinx était le

meilleur candidat répondant à nos besoins pour diverses raisons qui sont détaillées dans ce chapitre.

Le chapitre 2 est organisé comme suit. L'outil d'injection des pannes adopté est décrit par la section 2.2. La section 2.3 introduit une nouvelle technique de protection du *SEU Controller*. Cette technique peut être considérée comme une contribution secondaire de cette thèse vu que la technique présentée est efficace, sans coûts et générique. Dans la section 2.4 les montages de test sont décrits. Ces derniers vont servir de plateformes d'implémentation des stratégies proposées et de validation au même temps. Finalement, la conclusion apparaît à la section 2.5.

## **2.2 Outil d'injection des pannes : *SEU Controller***

Le *SEU Controller* (Chapman, 2010a; 2010b), développé par Xilinx, est une macro qui est instanciée avec le design dans le FPGA. Elle offre deux fonctionnalités principales:

- 1) la correction des erreurs de configuration causées par des SEU dès qu'elles sont détectées en effectuant un balayage de la mémoire de configuration, ce qui permet de détecter et corriger les SEU;
- 2) l'émulation des SEU dans les FPGA Virtex-5 en injectant des pannes dans la mémoire de configuration, ce qui permet l'évaluation de la robustesse des composants face aux événements singuliers.

Le *SEU Controller* permet un suivi de toutes les activités de la macro, y compris des rapports détaillés de tout SEU injecté, des rapports d'état contenant le nombre de balayage des trames de la mémoire de configuration effectués (*scans*) afin de détecter les erreurs ainsi que le nombre d'erreurs de configuration qui ont été réparées. La réparation d'une erreur de configuration consiste à inverser le bit de configuration où cette erreur a été détectée. Il est aussi possible d'afficher le contenu de n'importe quelle trame (*frame*) de bits de configuration pour confirmer l'injection de pannes ainsi que les corrections apportées. Les ports d'entrées et sorties du *SEU Controller* sont illustrés à la figure 2.1.

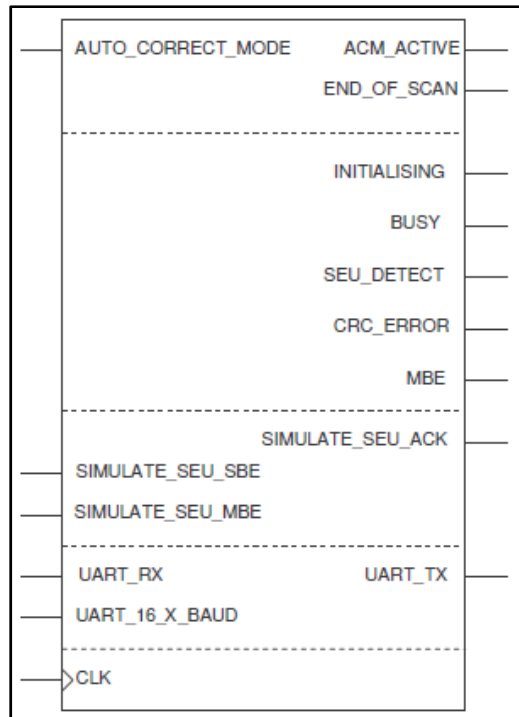


Figure 2.1 Entrées et sorties du *SEU Controller*  
Tirée de Chapman (2010a)

Les entrées/sorties de la macro sont décrites comme suit (Chapman, 2010a):

- ***AUTO\_CORRECT\_MODE***: spécifie le mode opérationnel de la macro. Deux modes sont supportés : le *Detect Only Mode (DOM)* ou *Automatic Correction Mode (ACM)*;
- ***ACM\_ACTIVE***: confirme que le *SEU Controller* fonctionne en mode *Automatic Correction Mode (ACM)*;
- ***END\_OF\_SCAN***: est utilisé pour confirmer que le circuit *Readback CRC* est en train de balayer le composant;
- ***INITIALISING***: indique que le *SEU Controller* va entrer en phase d'initialisation. Ce signal est actif après la configuration du composant;
- ***BUSY***: indique que la macro est en train d'exécuter une tâche telle que la correction ou l'injection d'erreur;
- ***SEU\_DETECT***: signifie qu'une erreur de configuration a été détectée ou non par le circuit *Readback CRC*;

- **CRC\_ERROR:** indique qu'une erreur de configuration a été détectée par le circuit *Readback CRC*;
- **MBE:** indique qu'une erreur sur multiple bits (*Multiple Bit Error*) est détectée dans la mémoire de configuration et qu'elle ne peut pas être corrigée;
- **SIMULATE\_SEU\_SBE & SIMULATE\_SEU\_MBE:** indiquent que la macro simule (émule) une erreur simple (*Single Bit*) ou multiple (*Multiple Bit Error*), respectivement;
- **SIMULATE\_SEU\_ACK:** désigne que la procédure d'injection d'une erreur dans le composant a commencé;
- **UART\_TX:** donne des informations utiles concernant l'état du composant ainsi que les tâches exécutées;
- **UART\_RX:** sert aussi à obtenir de l'information ainsi que contrôler le *SEU Controller* en lui envoyant des commandes;
- **CLK:** signal d'horloge.

Le *SEU Controller* peut être contrôlé de deux façons (Chapman SEUC, 2010):

- **Pin Control:** la macro est contrôlée en utilisant les ressources offertes par la plaquette sur laquelle se retrouve le FPGA, telles que les boutons, les interrupteurs, etc;
- **UART Control:** la macro est commandée par un ordinateur à travers une interface UART en lui envoyant des caractères bien définis. Le tableau 2.1 résume les caractères qui peuvent être envoyés au *SEU Controller* via ce mode ainsi que la description des commandes à effectuer.

L'émulation des SEU avec *SEU Controller* a été implémentée d'une manière à rapprocher le plus possible le modèle de panne de la réalité. L'injection des pannes dans la mémoire de configuration du FPGA peut être aléatoire ou déterministe, auquel cas les bits à changer sont choisis par l'utilisateur. Pour une injection aléatoire avec *SEU Controller*, la macro détermine les adresses cibles à l'aide de deux générateurs de nombres pseudo-aléatoires : le premier détermine l'adresse de la trame et l'autre la position du bit dans la trame. Afin d'injecter un SEU, la macro lit la trame sélectionnée aléatoirement puis inverse le bit cible et enfin, la trame modifiée est écrite à la même adresse de la mémoire de configuration à partir de laquelle la trame originale est lue.

Tableau 2.1 Description des commandes à envoyer au *SEU Controller*  
via le mode *UART Control*  
Adapté de Chapman (2010a)

Caractère	Description de la commande à effectuer
*	Basculer au mode <i>UART Control</i>
#	Basculer au mode <i>Pin Control</i>
S	Générer le rapport d'état
D	Enclencher le <i>Detect Only Mode</i> (DOM)
A	Enclencher le <i>Auto Correction Mode</i> (ACM)
1	Simuler aléatoirement une erreur simple ( <i>Single Bit Error</i> )
2	Simuler aléatoirement une erreur double ( <i>Double Bit Error</i> )
R	Lire et afficher une trame de configuration spécifique
Q	Convertir le numéro d'une trame spécifique à son adresse
T	Inverser la valeur d'un bit de configuration situé à une position spécifique

Toutefois, en cas de détection d'un SEU, l'opération de balayage de la mémoire de configuration doit être interrompue. Ceci a deux effets potentiels. Le premier est que la détection des pannes n'est pas possible lors du processus d'injection des pannes alors qu'en pratique, la détection doit être une opération continue. Le deuxième effet est spécifique au mode ACM (*Auto Correction Mode*) du *SEU Controller* où les SEU sont corrigés dès qu'ils sont détectés et consiste à reprendre toujours l'opération de détection des pannes à partir de la première trame de la mémoire de configuration. Le temps de détection des pannes est ainsi proportionnel à la position de la panne dans la mémoire de configuration. En effet, les pannes dans les premières trames seront détectées pratiquement immédiatement alors que ceux dans les dernières trames prendront environ le temps consommé par un balayage de toute la mémoire de configuration pour qu'elles soient détectées.

Concernant l'architecture interne du *SEU Controller*, il se compose essentiellement des primitives FRAME\_ECC et ICAP du Virtex 5 (Dutton et Stroud, 2009; Gajjar et al., 2011).

La primitive FRAME ECC (*Error Correcting Code*) sert à détecter et identifier les erreurs simples (*Single Bit Errors*) et doubles (*Double Bit Errors*) dans les différentes trames des bits

de configuration. Durant le Readback de chaque trame à partir de la mémoire de configuration, la primitive calcule les bits de Hamming (nommés aussi bits de parité et existent avec les bits de configuration dans chaque trame afin de permettre d'identifier les bits erronés en cas d'erreur) ainsi que la parité totale de la trame relue, et elle en fait la comparaison. En se basant sur cette dernière, la primitive indique la présence ou non d'erreurs et si l'erreur est simple ou double. Un signal nommé syndrome indiquant l'emplacement des erreurs simples est alors généré.

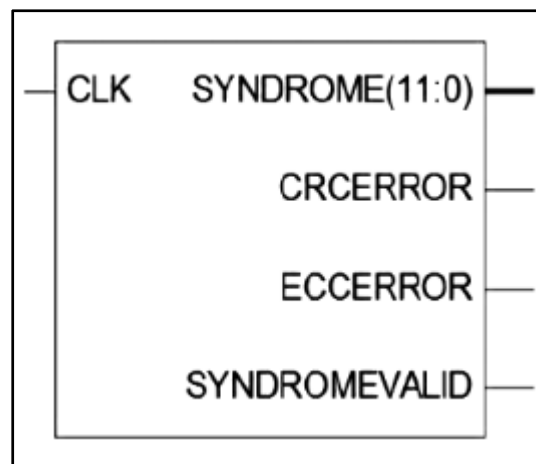


Figure 2.2 La primitive FRAME\_ECC  
Tirée de Gajjar et al. (2011)

Le *SEU Controller* fait aussi appel à la primitive ICAP (*Internal Configuration Access Port*) qui permet l'accès à la mémoire de configuration du FPGA et le cas échéant l'inversion de bits de configuration ciblés. La figure 2.3 montre les signaux d'entrée et de sortie de la primitive ICAP du FPGA Virtex 5. La primitive ICAP fonctionne comme l'interface de configuration externe SelectMAP sauf qu'elle a deux bus séparés pour la lecture et l'écriture dans la mémoire de configuration, à savoir O et I respectivement. Le signal *WRITE* contrôle la fonction effectuée par la primitive; lecture ou écriture. CE est un signal d'activation et CLK est un signal d'horloge. Le signal *BUSY* est activé lors des opérations de lecture (Tambara et al., 2016).



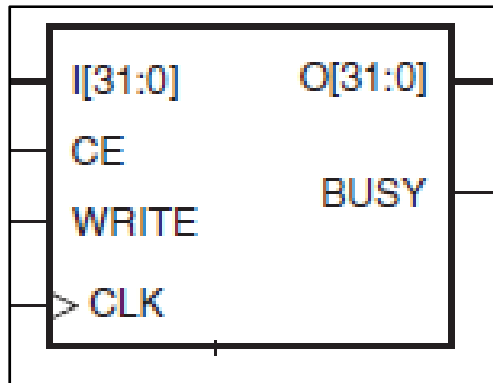


Figure 2.3 La primitive ICAP du Virtex 5  
Adaptée de Chapman et Jones (2010)

Il faut noter que dans certains cas, les demandes d'inversion de bits de configuration n'engendrent pas d'erreurs. Dans un premier cas, certaines adresses mènent à des bits qui ne sont pas utilisés (Chapman, 2010a). Tel que montré par la figure 2.4, il y a 16 bits inutilisés par trame. Dans les faits, ces bits n'existent tout simplement pas physiquement dans le FPGA. Notons également que quelques trames peuvent avoir des bits non utilisés supplémentaires. Dans un deuxième cas, le *SEU Controller* n'a pas accès aux bits des tables de conversion (*LUT*) utilisées comme mémoire distribuée dans un design, ni à ceux des *Block RAM* (Chapman, 2010a). Le *SEU Controller* a en effet été conçu pour éviter les bits de mémoire considérés comme des données d'utilisateur.

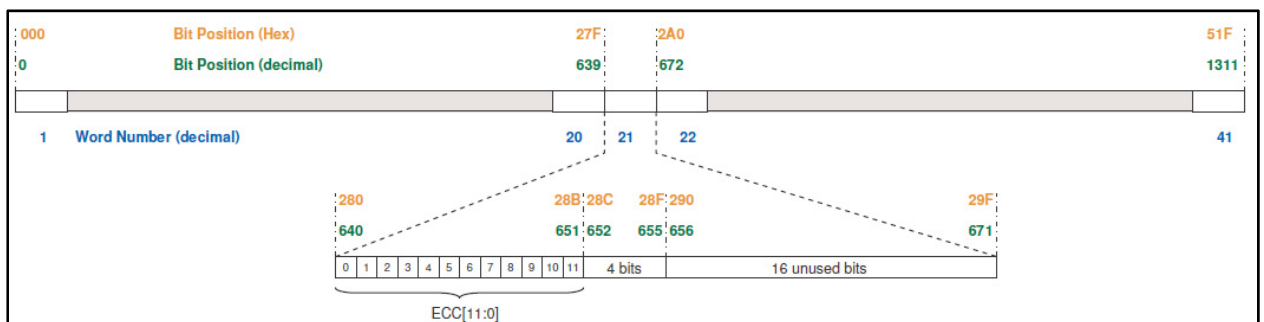


Figure 2.4 Position des bits non-utilisés dans une trame  
Tirée de Chapman (2010a)

La macro du *SEU Controller* est elle-même susceptible aux SEU. Plus spécifiquement, deux types d'erreurs peuvent causer le dysfonctionnement du *SEU Controller*: les erreurs affectant ses bits de configuration ou celles affectant sa mémoire de données. Les bits les moins critiques peuvent être corrigés par la macro vu qu'elle a la capacité de s'auto-réparer. Les erreurs dans le ICAP ou dans le FRAME\_ECC peuvent mener à écrire des trames incorrectes dans la mémoire de configuration.

### 2.3 Procédure de protection du *SEU Controller*

Tel que mentionné dans la section 3.2, la macro du *SEU Controller* occupe une partie de la mémoire de configuration, ce qui la rend vulnérable aux injections des pannes qui peuvent mener à son dysfonctionnement. Pour la protéger, plusieurs techniques de mitigation existant dans la littérature peuvent être appliquées, telles que la redondance triple (TMR). Cette dernière est très efficace comme technique de mitigation. Cette technique est par contre très coûteuse vu qu'elle est gourmande en termes des ressources utilisées. Adopter le TMR pour le *SEU Controller* peut de plus influencer considérablement le placement et le routage du design cible. Comme il fallait minimiser l'impact de son insertion dans le montage expérimental initial, qui a été soumis aux radiations sans la présence du *SEU Controller* (qui fut rendu accessible après la première expérimentation à TRIUMF), le TMR n'a pas été considéré dans le cadre de cette thèse.

Comme contribution secondaire, nous proposons une procédure de protection du *SEU Controller* ne nécessitant aucune ressource matérielle supplémentaire. Cette procédure permet de minimiser le risque d'autodestruction du *SEU Controller* et de faciliter l'automatisation de la procédure d'injection des pannes en minimisant les interruptions des expérimentations d'injection des pannes à cause de la cassure du *SEU Controller*. L'idée consiste à éviter l'injection des pannes dans les bits de configuration qui peuvent influencer le bon fonctionnement du *SEU Controller*. La plupart des bits de configuration du *SEU Controller* peuvent être extraits des fichiers EBC et EBD (*Essential Bits Files*) générés par l'outil *Bitgen* de Xilinx. EBC et EBD (voir ANNEXE I) sont, respectivement, les fichiers contenant les données de configuration du design et les données de masque pour les bits de

configurations essentiels (potentiellement critiques) où quelques bits de cet ensemble configurent le *SEU Controller*. Les définitions des bits de configuration, bits critiques et bits potentiellement critiques sont données comme suit (Le, 2012):

- **bit de configuration**: information contenue dans un élément de mémoire SRAM de configuration du FPGA;
- **bit critique**: bit de configuration causant une défaillance fonctionnelle s'il change d'état;
- **bit potentiellement critique**: bit de configuration changeant la circuiterie du design s'il change d'état. Toutefois, son inversion peut ne pas affecter le fonctionnement du design.

La figure 2.5 illustre les bits importants à éviter lors de la procédure d'injection déterministe des pannes. Les bits de configuration du design ainsi que les bits potentiellement critiques sont, respectivement, les ensembles des bits contenus dans les fichiers EBC et EBD. Les deux ensembles de bits contiennent des bits en commun.

L'ensemble des bits de configuration du *SEU Controller* et l'ensemble des bits potentiellement critiques du *SEU Controller* sont extraits des fichiers EBC et EBD, respectivement. L'union de ces deux ensembles doit contenir la plupart (si non pas tous) des bits qui peuvent influencer le comportement du *SEU Controller* quand inversés.

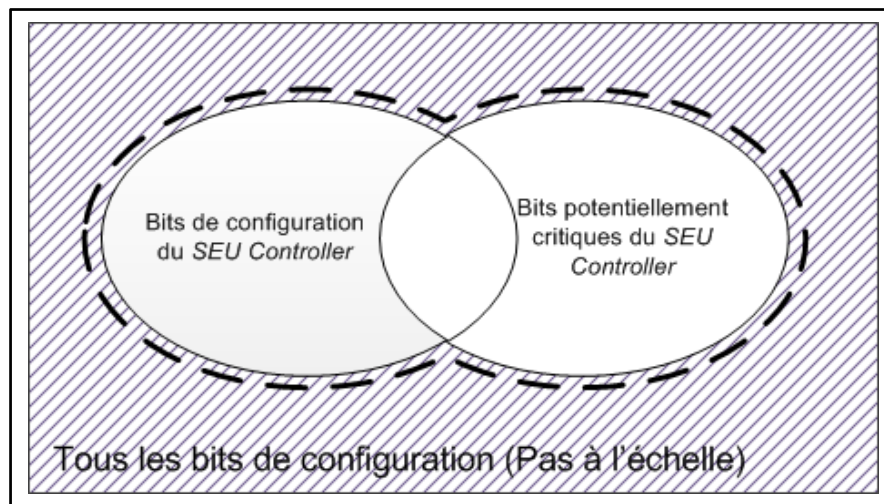


Figure 2.5 Distinction des bits de configuration et de ceux du *SEU Controller* et de ceux potentiellement critiques

Le principe de la procédure de protection du *SEU Controller* est basé sur l'exclusion des adresses des bits configurant le *SEU Controller* de la liste des adresses de tous les bits de configuration. La nouvelle liste, incluant tous les bits de configuration sauf ceux du *SEU Controller*, devient la liste cible à partir de laquelle les séquences de test sont générées. Les étapes à suivre pour déterminer les bits de configuration ainsi que les bits essentiels du *SEU Controller* sont détaillées dans les prochaines sections.

### 2.3.1 Détermination des bits de configuration du *SEU Controller*

La première étape de la procédure de protection du *SEU Controller* consiste à trouver les adresses des bits qui le configurent. Cette étape est illustrée à la figure 2.6.

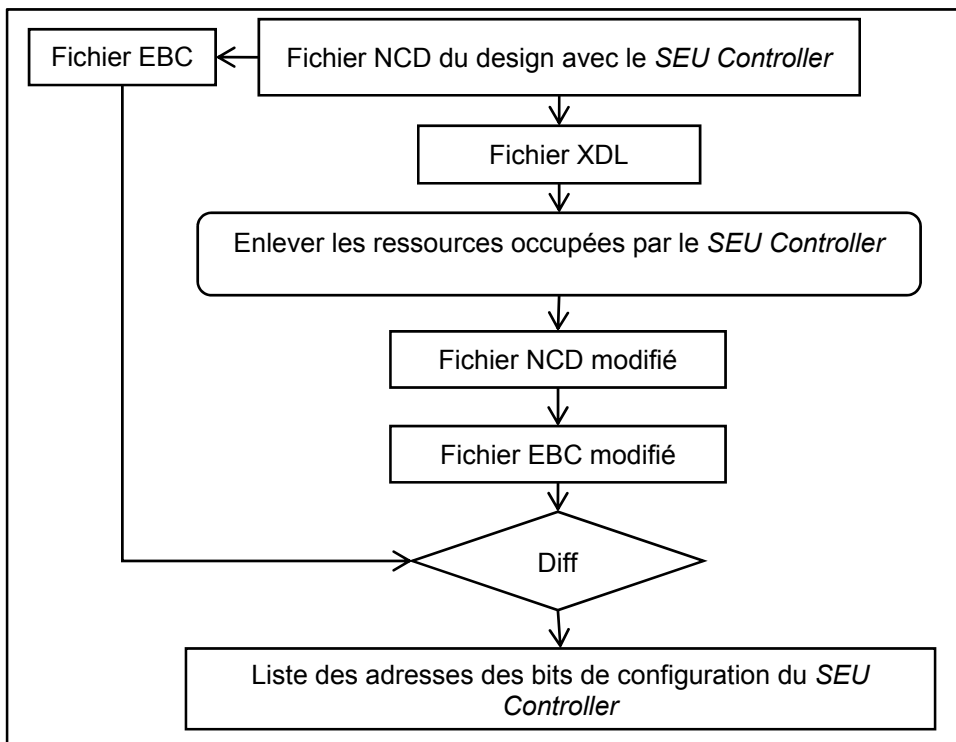


Figure 2.6 Procédure de détermination des bits de configuration du *SEU Controller*

- 1) le fichier NCD (*Native circuit Description*) de tout le design sous test et le *SEU Controller* est généré. Le fichier EBC est par la suite créé;

- 2) le fichier XDL (*Xilinx Design Language*) est généré par *Bitgen*. Ce fichier texte lisible par l'humain décrit le design synthétisé placé et routé. À l'aide d'un script, les ressources occupées par le *SEU Controller* sont automatiquement enlevés du fichier XDL selon le préfixe *SEU Controller* dans les noms des instances et nets;
- 3) les fichiers NCD et EBC modifiés sont produits à partir de la version altérée du fichier XDL;
- 4) les adresses des bits de configuration configurant le *SEU Controller* sont extraites par une comparaison différentielle des deux fichiers EBC, à l'aide d'un autre script.

Une limite importante de cette procédure est que les bits critiques définis à leur valeur initiale par défaut (n'ayant pas changé de valeur) ne sont pas détectés. Cette limitation est donc résolue par la seconde procédure décrite par la suite.

### 2.3.2 Détermination des bits essentiels du *SEU Controller*

La deuxième étape de la procédure de protection du *SEU Controller* consiste à extraire les adresses des bits essentiels du *SEU Controller*. La procédure suivie est illustrée à la figure 2.7.

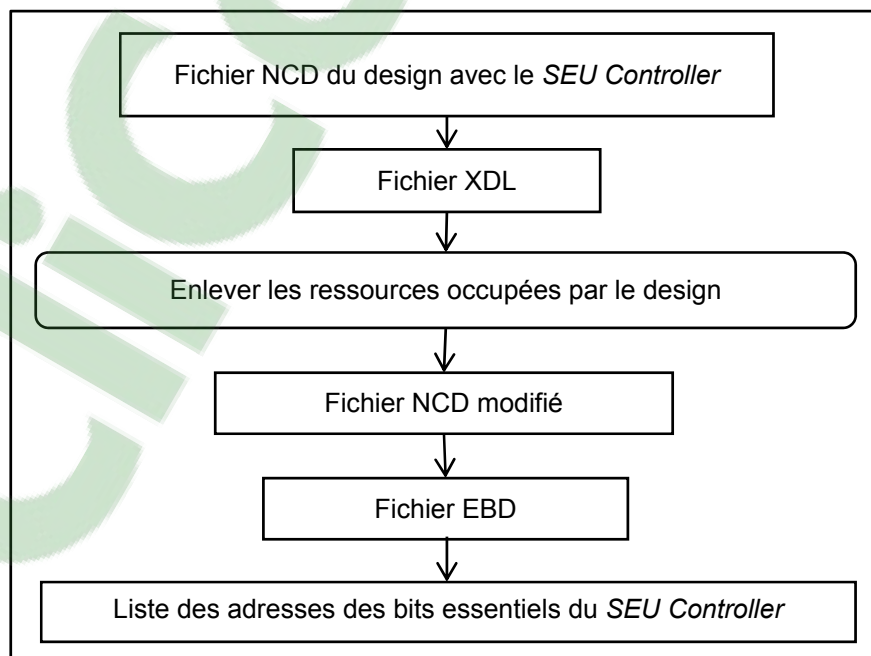


Figure 2.7 Procédure de détermination des bits essentiels du *SEU Controller*

- 1) après la création du fichier NCD de tout le design sous test et le *SEU Controller*, le fichier XDL est généré;
- 2) les ressources occupées par le design cible sont enlevées automatiquement du fichier XDL, à l'aide d'un script en se basant sur le nom du design dans les noms des instances et nets;
- 3) le fichier NCD modifié est généré à partir de la version altérée du fichier XDL;
- 4) le fichier EBD contenant les adresses des bits essentiels du *SEU Controller* est produit à partir de la version altérée du fichier NCD;
- 5) les adresses des bits essentiels du *SEU Controller* sont extraites du fichier EBD en déterminant les positions des bits ayant la valeur 1, à l'aide d'un autre script.

Vu que les bits essentiels sont définis par Xilinx comme les bits associés à la circuiterie du design (Le, 2012), on pourrait penser qu'éviter l'injection des pannes dans ces bits essentiels est suffisant pour protéger le *SEU Controller*, c'est-à-dire que *Bitgen* et son fichier EBD peuvent être utilisés directement pour identifier les bits de configuration appartenant au DUT. Toutefois, nos expérimentations avec *Bitgen* et son fichier EBD montrent que la liste des bits essentiels fournie par *Bitgen* est loin d'être parfaite. Nous avons en effet identifié de vrais bits critiques (dont l'inversion de leur état (*bit flips*) causent des erreurs) qui n'apparaissent pas dans la liste de bits essentiels (ces résultats sont présentés au chapitre 4). De plus, un nombre important de ces bits dits essentiels ne sont pas critiques tel est le cas du design présenté dans la section 4.2 où tous les bits configurant les LUT sont identifiés comme essentiels par *Bitgen* alors que seulement 2 bits des 64 bits configurant chaque LUT sont critiques.

Les inefficacités de *Bitgen* sont aussi rapportées par (Hobeika et al., 2014). C'est pour cette raison que dans ce travail, les bits potentiellement critiques générés par *Bitgen* sont seulement utilisés dans la procédure de protection du *SEU Controller* comme une mesure supplémentaire afin de l'améliorer.

Dans notre cas, l'application de notre procédure de protection du *SEU Controller* réduit l'occurrence des interruptions fonctionnelles du *SEU Controller* d'un facteur de 48, de 0.1075% à 0.0022% (ces valeurs sont obtenues en calculant le ratio des *bit flips* injectés

généralisant des erreurs sur le nombre total des *bit flips* injectés). Des expérimentations supplémentaires peuvent être effectuées afin de déterminer un facteur d'amélioration générique. D'autres adresses de vrais bits critiques peuvent être observées lors de l'occurrence d'une interruption de fonctionnement du *SEU Controller* durant les expérimentations d'injection des pannes. Ces adresses peuvent être enlevées de la liste globale des adresses des bits de configuration une fois détectées. La procédure de protection du *SEU Controller* a été utilisée avec succès avec le FPGA Virtex-5. Elle pourrait aisément être appliquée aux FPGA récents (séries 6 et 7) de Xilinx qui utilisent la nouvelle version du *SEU Controller* (SEM IP). L'approche proposée de protection du *SEU Controller* est une méthode efficace de protection sans coût supplémentaire en termes de ressources internes au FPGA.

#### 2.4 Aperçu général des montages de test

Le premier montage cible de nos expérimentations d'émulations par injection des pannes est le même que celui décrit dans (Thibeault et al., 2012). Tel que montré à la figure 2.8, ce montage expérimental consiste en une carte *Genesys* de *Digilent* basée sur un FPGA Virtex-5 XC5VLX50T de Xilinx, une carte fille avec le composant 7404 (avec 6 inverseurs et un  $V_{cc}$  de 1.2 V), un analyseur de spectre Anritsu MS2721A et un PC pour contrôler les expérimentations d'injection des pannes. Une interface *Labview* est utilisée pour contrôler l'analyseur de spectre.

Le premier design cible des expérimentations d'injection des pannes consiste à deux oscillateurs en anneaux « *Ring Oscillators* » (RO) chacun ayant 1 799 inverseurs, implémentés dans les *LUT* du FPGA. Les deux RO tournent à des fréquences quasi-égales,  $F_1$  et  $F_2$ , où  $F_2 > F_1$ . La différence entre les fréquences des RO ( $F_2 - F_1$ ), donnée comme la sortie de la carte fille basée sur le composant 7404, est mesurée à un bout de la résistance de 5.1 k $\Omega$  en utilisant l'analyseur de spectre.

Selon (Thibeault et al., 2012), les fréquences des RO sont, respectivement, égales à  $F_2 \approx F_1 \approx 1.25$  MHz, et la valeur de la différence des fréquences est égale à  $F_2 - F_1 \approx 12.4$  kHz.

Pour les expérimentations d'émulation par injection des pannes, le design des RO est implémenté avec la macro de *SEU Controller* dans le FPGA. La totalité du design occupe 52 % des *slices* de la puce ainsi que 13 % de ses LUT. Le *SEU Controller* requiert moins que 4 % des ressources du plus petit FPGA de la famille Virtex-5 et moins que 1 % pour les puces les plus larges. Considérant seulement le design des RO, il consomme 3600 *slices* des 7200 disponibles dans le Virtex-5 VLX50T, ce qui signifie 50 % des *slices* de la puce et 12.5 % des LUT. Les RO requièrent 6 IOB. L'estimation du nombre des bits de configuration requis par notre design basé sur une technique d'approximation de Xilinx montre que 30 % des bits de configuration sont utilisés, sachant que le nombre total des bits de configuration du Virtex-5 VLX50T est  $14.05 \times 10^6$ .

Ce montage est principalement conçu pour détecter les variations dans la différence des fréquences des RO, appelées changements des délais observables « *Observable Delay Changes* » (ODC). Ces derniers sont dus à des capacités parasites supplémentaires (Darvishi et al., 2014).

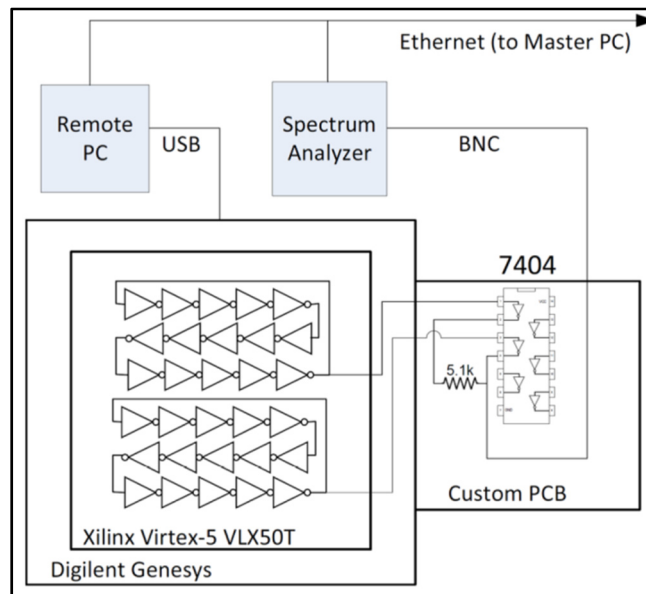


Figure 2.8 Premier montage expérimental (RO dans les CLB)



Ce montage permet la mesure des changements de délais de 40 ps et plus, et peut aussi détecter l'apparition des cassures dans les RO (*Ring Oscillator Breaks*, ROB). Ces ROB peuvent être causés par des *bits flips* dans les bits de configuration générant des pannes de type « *open* » ou « *stuck-at* ». Deux exemples fort probables sont : l'inversion des deux bits des LUT définissant chaque inverseur et la désactivation du *pass transistor* à travers lequel un chemin routé passe (en inversant son bit de configuration de '1' à '0') (Darvishi et al., 2014). Tout au long du travail décrit par cette thèse, ce premier montage est utilisé pour comparer les taux d'apparition de ces événements (ODC et ROB) obtenus avec les expérimentations d'émulations d'injection des pannes utilisant le *SEU Controller* avec ceux donnés par les tests sous radiations de protons.

Un deuxième montage de test a été utilisé pour valider les résultats obtenus durant les travaux décrits par cette thèse. Il est présenté à la figure 2.9. Ce montage est similaire à celui décrit précédemment. La seule différence est que le design des RO cible des injections des pannes est implémenté dans les IOB (les primitives IOBUF) du FPGA dans le deuxième montage au lieu des CLB. Le premier RO occupe 179 IOB alors que le deuxième nécessite 160 IOB. Certains de ces IOB sont configurés comme inverseurs alors que les autres ne le sont pas. Au total, 71 % des 480 IOB disponibles sur le FPGA cible sont utilisés par le design des RO.

Similairement au montage précédent, la différence de fréquence entre les deux RO est mesurée par l'analyseur de spectre sur la sortie de la résistance de 5.1 k $\Omega$ . Par la suite l'interface *Labview* qui contrôle l'analyseur de spectre, génère les résultats des mesures dans des fichiers textes.

Selon (Tazi et al., 2014), les fréquences des RO sont, respectivement, égales à  $F2 \approx 919$  kHz et  $F1 \approx 827$  kHz, ce qui donne une différence des fréquences égale à  $F2 - F1 \approx 92$  kHz.

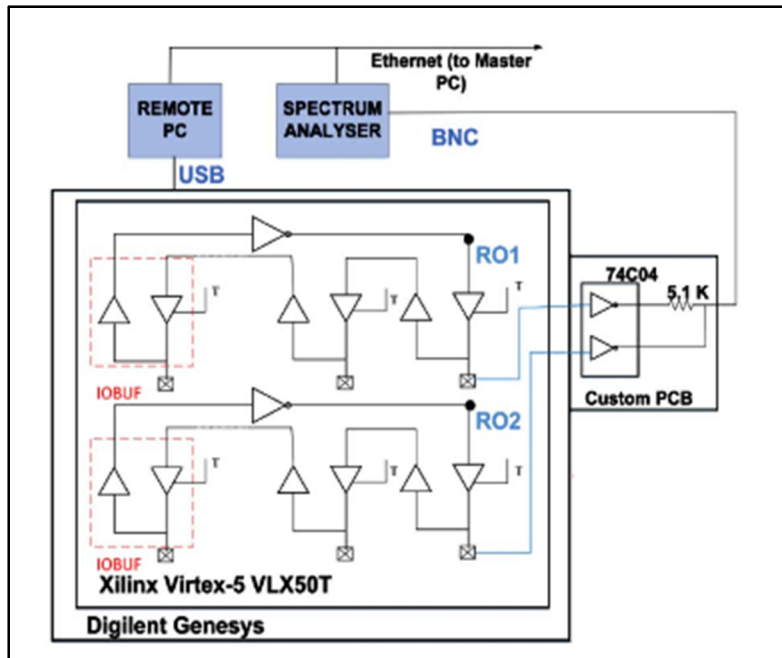


Figure 2.9 Deuxième montage expérimental (RO dans les IOB)  
Tirée de Tazi et al. (2014)

## 2.5 Conclusion

Ce chapitre a présenté l'outil d'injection des pannes adopté pour les travaux décrits par cette thèse. Cet outil permet de faciliter les expérimentations d'émulation par injection des pannes des effets de radiation. Le *SEU Controller* a été exploité comme l'élément de base de la stratégie de test en pré-certification proposée par cette thèse, vu qu'il est simple à utiliser, qu'il ne demande pas de matériel spécifique (outre le fait qu'il s'applique sur des FPGA Virtex-5 et les familles Xilinx subséquentes) et que l'accès à son code source est gratuit et ouvert à tous. En effet, cette macro permet à la fois d'émuler la présence des SEU dans un design par injection des pannes et de corriger les erreurs détectées à travers les modes *DOM* et *ACM*. Il est facile à contrôler en envoyant des commandes sous forme de caractères à travers une interface UART ou en manipulant les broches de la plaquette contenant le FPGA cible. Sa capacité d'être implémentée avec le design cible, vu qu'elle n'est qu'un code VHDL accessible gratuitement sur le site de Xilinx dans le même FPGA, la rend une solution

avantageuse par rapport aux autres outils coûteux demandant la présence de plateformes supplémentaires pour assurer l'injection des pannes.

Les caractéristiques de cet outil en font le meilleur candidat pour répondre aux objectifs de cette thèse. De plus, les montages de test utilisés durant tous les travaux de thèse sont présentés. Comme le *SEU Controller* est implémenté au sein du FPGA cible avec le DUT, nous avons élaboré dans le cadre de cette thèse une technique de protection de l'outil d'injection des pannes pour le prévenir de l'autodestruction et faciliter le déroulement des expérimentations d'injection des pannes. Cette technique qui peut être considérée comme une contribution secondaire est générique, simple à implémenter et applicable pour *SEU Controller* ainsi que pour SEM (la version plus récente du *SEU Controller*).

Le noyau d'injection des pannes a été présenté dans ce chapitre. Toutes les stratégies d'optimisation des expérimentations d'émulation ont été construites autour de ce noyau. Ces stratégies sont présentées dans les chapitres qui suivent.



## CHAPITRE 3

### PROCÉDURE EFFICACE D'ÉMULATION DES EFFETS DES SEU DANS LES FPGA À BASE DE SRAM

#### 3.1 Introduction

Le chapitre précédent a présenté l'environnement de test adopté par les travaux décrits par cette thèse. Une description détaillée de l'outil d'injection des pannes utilisé pour les expérimentations d'émulation y a été exposée. L'environnement d'émulation a aussi été détaillé, présentant les designs cibles ainsi que les ressources matérielles et logicielles utilisées, notamment celles de mesure et de contrôle. Le choix de l'outil d'injection des pannes a été influencé par les caractéristiques offertes par la macro *SEU Controller* elle-même ainsi que par les objectifs de la procédure d'injection des pannes.

L'objectif de recherche présentée dans le présent chapitre est d'élaborer une procédure d'émulation par injection de panne la plus réaliste possible dans la mémoire de configuration des FPGA à base de mémoire SRAM. En d'autres termes, nous cherchons à produire des résultats d'émulation proches de ceux obtenus par les tests de certification sous faisceaux de particules. Tel que mentionné dans (Faure, Velazco et Peronnard, 2005), la recherche du réalisme des résultats générés par l'émulation aide à prédire le comportement d'un design FPGA dans l'environnement final où il va opérer.

Le chapitre 3 est organisé comme suit. La section 3.2 présente la pierre angulaire de l'approche choisie pour obtenir des résultats réalistes à partir d'émulation par injection des pannes dans les FPGA à base de mémoire SRAM, à savoir la prise en compte de la différence de sensibilité relative entre les bits de configuration à '1' et ceux à '0'. La procédure automatisée d'injection des pannes, la présentation de la méthodologie optimisée d'injection des pannes tenant compte de la différence de sensibilité relative entre les bits de configuration et la validation des résultats sont décrits par les sections 3.3, 3.4 et 3.5, respectivement. Enfin, la conclusion de ce chapitre apparaît à la section 3.6.

### 3.2 Différence de sensibilité relative des bits de configuration à 1 et ceux à 0

Les mécanismes d'injection des pannes en général et du *SEU Controller* en particulier sont limités par une distribution uniforme des pannes sur les bits de configuration à '1' et ceux à '0' quand ils sont déployés en mode d'injection aléatoire des pannes. Sachant que les bits à '1' et ceux à '0' ont des sensibilités différentes face aux SEU, ce mode d'injection des pannes peut influencer le réalisme des résultats obtenus.

La nouvelle approche d'injection des pannes présentée dans ce chapitre prend en considération la différence de sensibilité relative entre les bits de configuration à '0' et ceux à '1'. Cette nouvelle approche a été élaborée suite à des observations faites lors d'expérimentations sous faisceau de protons à TRIUMF (Thibeault et al., 2012). Il a été en effet observé, pour des niveaux d'énergies variant entre 8 et 105 MeV, que les bits à '1' sont en moyenne 2.1 fois plus sensibles que les bits à '0' pour le FPGA Virtex-5 de Xilinx. Ceci signifie que pour un même nombre de bits, les bits à '1' tendent à générer plus d'erreurs que les bits à '0' quand ils sont exposés aux radiations. Ce constat a été fait suite à des relectures de la mémoire de configuration à la fin de chacune des expérimentations sous radiation (68 au total) et à la comparaison des fichiers relus avec le fichier de configuration original contenant  $CB0$  bits à '0' et  $CB1$  bits à '1'. Après chaque expérimentation, le nombre des *bit flips* de '0' à '1',  $BF0$ , et le nombre des *bit flips* de '1' à '0',  $BF1$ , sont rapportés, et le ratio de la sensibilité relative des bits à '1' par rapport à ceux à '0', défini comme  $N = (BF1/CB1)/(BF0/CB0)$ , est calculé. Notons que cette différence de sensibilité a aussi été observée par (Bocquillon, 2009).

Pour qu'il soit justifié de la prendre en considération, cette différence de sensibilité doit avoir un impact sur la nature et le nombre des erreurs détectées. Cet impact est démontré dans ce qui suit. Dans ce travail, une attention particulière est portée aux ODC et aux ROB par la radiation tel que décrit dans (Thibeault et al., 2012), en raison du type de design sous test utilisé dans les montages expérimentaux.

Afin de montrer l'impact de la différence de la sensibilité relative des bits de configuration, deux campagnes d'injection des pannes avec *SEU Controller* (à l'aide de la procédure décrite

à la section 4.3) sur le premier montage expérimental (RO dans les CLB, Figure 3.8) ont été effectuées : une ciblant seulement les bits de configuration à '1' et l'autre ciblant seulement les bits à '0'. De plus, à partir des résultats obtenus à partir de ces deux campagnes, le nombre des bits critiques associés à chaque valeur ('0' ou '1') et à chaque type d'évènement (ODC ou ROB), est estimé. Dans chaque campagne d'injection des pannes, quatre séries d'expérimentations ont été accomplies. Chaque série est composée de 68 expérimentations (le même nombre qu'à TRIUMF), où les pannes sont aléatoirement injectées selon le nombre total des bits à cibler. Les résultats des deux campagnes sont donnés par les Tableaux 3.1 et 3.2 où les colonnes #2, #3 et #4 donnent le nombre d'ODC, le nombre de ROB et le nombre des bits flippés (EBF) pour chaque série d'expérimentations, respectivement. Toutes les expérimentations finissent par un ROB.

Tableau 3.1 Résultats de la campagne d'injection des pannes ciblant les bits à '1' (CB1)

	<b>ODCs</b>	<b>ROBs</b>	<b>Bits Flippés <i>EBF1</i></b>
Série (1)	0	68	589
Série (2)	0	68	743
Série (3)	0	68	517
Série (4)	0	68	581
Total	0	272	2 430

Tableau 3.2 Résultats de la campagne d'injection des pannes ciblant les bits à '0' (CB0)

	<b>ODCs</b>	<b>ROBs</b>	<b>Bits Flippés <i>EBF0</i></b>
Série (1)	296	68	107 963
Série (2)	347	68	140 419
Série (3)	446	68	164 313
Série (4)	403	68	146 417
Total	1 492	272	559 112

Dans les 272 expérimentations d'injection des SEU ciblant les bits à '1' (CB1), aucun ODC n'est observé alors que le nombre de ROB est égal à 272 vu que chaque expérimentation se termine par un bris de design. L'absence de détection d'ODC en ne ciblant que des bits configurées à '1' est conforme à la modélisation du principal mécanisme des ODC mentionné au chapitre précédent, à savoir l'ajout d'une capacitance additionnelle à une interconnexion,

et au fait que cet ajout devrait principalement se produire lors du renversement de la valeur d'un bit contrôlant le routage de '0' à '1'. Pour la campagne d'injection des pannes ciblant les bits à '0' ( $CB_0$ ), 1492 ODC ainsi que 272 ROB ont été détectés.

Ces résultats montrent clairement l'effet du contenu des bits inversés. En effet, les ODC sont seulement observés quand les bits configurés à 0 sont inversés. De plus, le nombre des bits inversés nécessaire à la détection d'un ROB est significativement large lorsque les bits à 0 sont ciblés.

Sachant que le contenu du bit inversé a un impact sur la nature et le nombre d'erreurs détectées, il s'agit maintenant de déterminer si cet impact va demeurer apparent malgré le déséquilibre significatif entre les bits à '0' et ceux à '1'. En effet, dans le premier montage expérimental, il existe 82 fois plus de '0' que de '1' dans les bits de configuration. Cette seconde vérification sera effectuée à l'aide d'une métrique décrite dans ce qui suit.

Nous définissons la métrique  $Z_T$  comme le ratio de nombre d'ODC sur le nombre de ROB. En se basant sur les résultats obtenus par les deux campagnes d'injection des pannes ciblant juste des '0' et juste des '1'.  $Z_T$  peut être estimée par l'équation suivante :

$$Z_T = (CB\_ODC0 + N \times CB\_ODC1) / (CB\_ROB0 + N \times CB\_ROB1) \quad (3.1)$$

où :

- **$CB\_ROB0$** : nombre des bits critiques à '0' causant un ROB quand inversés à 1; valeur estimée = 5,247;
- **$CB\_ROB1$** : nombre des bits critiques à '1' causant un ROB quand inversés à 0; valeur estimée = 14,681;
- **$CB\_ODC0$** : nombre des bits critiques à '0' causant un ODC quand inversés à 1; valeur estimée = 28,779;
- **$CB\_ODC1$** : nombre des bits critiques à '1' causant un ODC quand inversés à 0; valeur estimée = 0;
- **$N$** : ratio de sensibilité relative entre les bits à '1' et les bits à '0'.

Le nombre de bits critiques causant un évènement (ODC ou ROB) quand inversés ('0' à '1' ou '1' à '0') peut être estimé comme  $E_i * CB_j / EBF_j$ , où  $E_i$  est le nombre d'évènements ciblés (ODC ou ROB),  $CB_j$  ( $CB_0$  or  $CB_1$ ) le nombre total des bits de configuration à la valeur cible



$j$  ('0' ou '1') et  $EBF_j$  le nombre total des bits inversés durant la campagne d'injection des pannes pour une valeur cible  $j$ . Cette estimation suppose que les bits critiques sont indépendants, c'est-à-dire, qu'il n'y a aucun effet de masquage entre eux. L'estimation des nombres des bits critiques pour chaque type d'évènement et chaque état du bit sont des mesures que notre environnement d'émulation permet d'extraire alors qu'elles sont indisponibles avec les tests accélérés. De plus, l'un des avantages de la métrique proposée est qu'elle n'est pas influencée par la procédure de protection du *SEU Controller* tant que le nombre des bits critiques reste le même. En effet, le nombre des bits enlevés par la procédure de protection du *SEU Controller* n'influence pas le calcul de la métrique vu que les paramètres de l'équation (3.1) dépendent seulement du nombre des bits critiques du design cible qui est dans notre cas celui des RO.

En utilisant les nombres estimés des bits critiques et  $N = 2.1$ , le ratio obtenu lors des expérimentations à TRIUMF, nous obtenons une valeur de  $Z_T$  égale à 0.79 qui est proche de la valeur obtenue à TRIUMF, à savoir 0.82. Considérant  $N = 1$  (injection aléatoire),  $Z_T$  devient égale à 1.44. Ces résultats décrits en détail dans la section 3.5 suggèrent que prendre en considération la sensibilité relative des bits de configuration mène à une diminution de 45% de  $Z_T$  en comparaison avec l'injection à sensibilité égale ( $N = 1$ ) des pannes. En d'autres mots, en moyenne, l'injection à sensibilité égale des pannes donne 1.8 fois plus d'ODC qu'en considérant la sensibilité relative. L'estimation de l'erreur entre les résultats de TRIUMF et ceux obtenus par l'injection aléatoire des pannes mène à une erreur de 75%.

La vérification en deux étapes aide à décider s'il est rentable ou non de prendre en considération la sensibilité relative des bits de configuration. Concernant l'injection des pannes, cette vérification à deux étapes est suffisamment générique pour qu'elle soit appliquée sur n'importe quel design où différents types d'évènements peuvent être observés. Dans notre cas, l'erreur de 45% dans la valeur de  $Z_T$  est suffisamment élevée pour justifier la considération de la sensibilité relative des bits de configuration.

### 3.3 Procédure automatisée d'injection des pannes

Cette section décrit la procédure automatisée d'injection des pannes développée dans le cadre de ce travail. Même si le but ultime est de pouvoir prendre en considération la sensibilité relative, cette procédure est suffisamment générique pour supporter d'autres modes d'injection à des fins de comparaison et de validation. Conséquemment, en plus du mode d'injection considérant la sensibilité relative, les trois modes suivants sont également supportés : le mode aléatoire (à sensibilité égale), le mode ciblant seulement des bits de configuration à '0', et celui ciblant seulement des bits de configuration à '1'. Le mode aléatoire permet la comparaison avec la technique usuelle d'émulation, alors que les deux derniers modes ont été utilisés dans la section précédente pour justifier la prise en considération de la sensibilité relative.

La procédure automatisée d'injection des pannes est générique et consiste aux trois tâches suivantes: 1) injecter des SEU dans les bits de configuration, 2) contrôler les sorties du DUT pour détecter les effets des SEU en cas d'erreurs et 3) configurer le FPGA.

Cette procédure a été appliquée sur le design des RO, ce qui prouve son efficacité. Le montage complet d'émulation construit autour de l'environnement de test pour le design des RO inclut trois parties majeures à savoir le *SEU Controller*, l'interface développée dans l'environnement logiciel *Labview* et l'outil *Digilent Adept*, ce qui permet d'automatiser la procédure d'injection des pannes tel que décrit par le paragraphe précédent. En effet, 1) l'injection des SEU dans les bits de configuration est assuré en contrôlant le *SEU Controller*, 2) le contrôle des sorties du DUT est effectué en contrôlant l'analyseur de spectre et 3) la configuration du FPGA est faite en agissant sur l'outil *Digilent Genesys*, respectivement.

Le fonctionnement de chaque partie est détaillé dans ce qui suit.

- contrôle du *SEU Controller*

Le *SEU Controller* peut être contrôlé de deux façons différentes (Chapman, 2010a):

- contrôle à travers les pins : la macro est contrôlée à travers des ressources offertes par la carte sur laquelle le FPGA est embarqué, tel que les boutons et les interrupteurs;

- contrôle à travers l'UART : la macro est contrôlée à travers une interface UART en envoyant des caractères spécifiques via la connexion RS232 à partir d'un ordinateur.

Vu que notre objectif est d'automatiser la procédure d'injection des pannes, le *SEU Controller* est contrôlé à travers l'UART avec une session Hyperterminal.

- Contrôle de l'interface *Labview*

Cette interface permet de détecter une panne, notamment un ODC ou un ROB pour le design RO décrit à la section 2.4. Afin de contrôler l'interface *Labview*, les fonctionnalités suivantes doivent être disponibles:

- activer l'interface *Labview* au début de chaque expérimentation;
- lire les fichiers de sortie de l'interface *Labview*;
- désactiver l'interface *Labview* après chaque expérimentation.

L'interface *Labview* génère deux fichiers (voir ANNEXE III), notamment :

- *Traces.log* : contenant toutes les données concernant les valeurs des fréquences lues à partir de l'analyseur de spectre;
- *Peak\_values.log* : rapporte le temps auquel les données sont lues à partir de l'analyseur de spectre, la valeur de la fréquence de la raie (*peak*) représentant la différence entre les deux fréquences des RO, l'amplitude de la raie et une valeur booléenne nommée *flag d'ODC* qui est automatiquement activée en cas de détection d'ODC, c'est-à-dire, quand la différence  $F2-F1$  mesurée avant et après l'injection des pannes est plus large (en valeur absolue) qu'un seuil prédéfini qui est à 55 Hz pour nos expérimentations.
- Configuration du FPGA

La carte de test commerciale de *Digilent Genesys* basée sur le FPGA Xilinx Virtex-5 XC5VLX50T, utilisé pour les expérimentations sous radiation à TRIUMF a aussi été utilisée pour nos expérimentations d'émulation. Cette carte vient avec un outil logiciel appelé *Digilent Adept* assurant la configuration du FPGA. Le FPGA est automatiquement reconfiguré en activant l'outil *Digilent Adept* à chaque début d'expérimentation d'injection des pannes.

La figure 3.1 représente le flot d'automatisation de la procédure d'injection des pannes. Tout d'abord, le FPGA est configuré et l'interface *Labview* est activée pour lire les données à

partir de l'analyseur de spectre. Un temps d'attente de 6 secondes est requis avant d'injecter des pannes dans le design afin que l'interface *Labview* soit initialisée et commence à acquérir les données de l'analyseur de spectre. Ensuite, le mode de contrôle du *SEU Controller* à travers l'UART est déployé en envoyant le caractère '\*' via l'Hyperterminal.

Le *Detect Only Mode* (DOM) est, par la suite, déployé afin d'injecter les pannes en mode cumulatif où aucune correction n'est appliquée. Ceci permet d'imiter la distribution des pannes lors des tests sous radiation où les pannes s'accumulent. En plus, le type des pannes ciblées à TRIUMF (les délais) montrent une tendance à s'accumuler vu que plus qu'un ODC peut se manifester avant un ROB. Après, un temps d'attente est requis afin de permettre à l'interface *Labview* d'acquérir les données de l'analyseur de spectre et de générer les fichiers de sortie *Peak\_values1.log* et *Peak\_values2.log*.

La séquence de test prédéfinie (enregistrée dans un fichier texte) est ensuite lue ligne par ligne où chaque ligne contient une adresse d'un bit de configuration à cibler. Après cela, le *SEU Controller* est mis en mode d'injection des pannes en envoyant le caractère 't' via l'Hyperterminal.

Ensuite, l'adresse de la trame et la position du bit cible d'injection dans la trame sont extraites du fichier texte contenant la séquence de test et par la suite, elles sont envoyées au *SEU Controller* à travers l'Hyperterminal.

Après l'injection d'une panne (*bit flip*), un temps d'attente de 2.5 secondes est requis pour laisser le temps à *Labview* pour mettre à jour les données reçues à partir de l'analyseur de spectre et conséquemment, mettre à jour ses fichiers de sortie. Ces 2.5 secondes sont suffisantes pour observer les effets des pannes sur les fréquences générées par les RO pour chaque adresse ciblée et mettre à jour les fichiers de sortie de l'interface *Labview*. Cette dernière peut effectuer 40 acquisitions par minute à partir de l'analyseur de spectre.

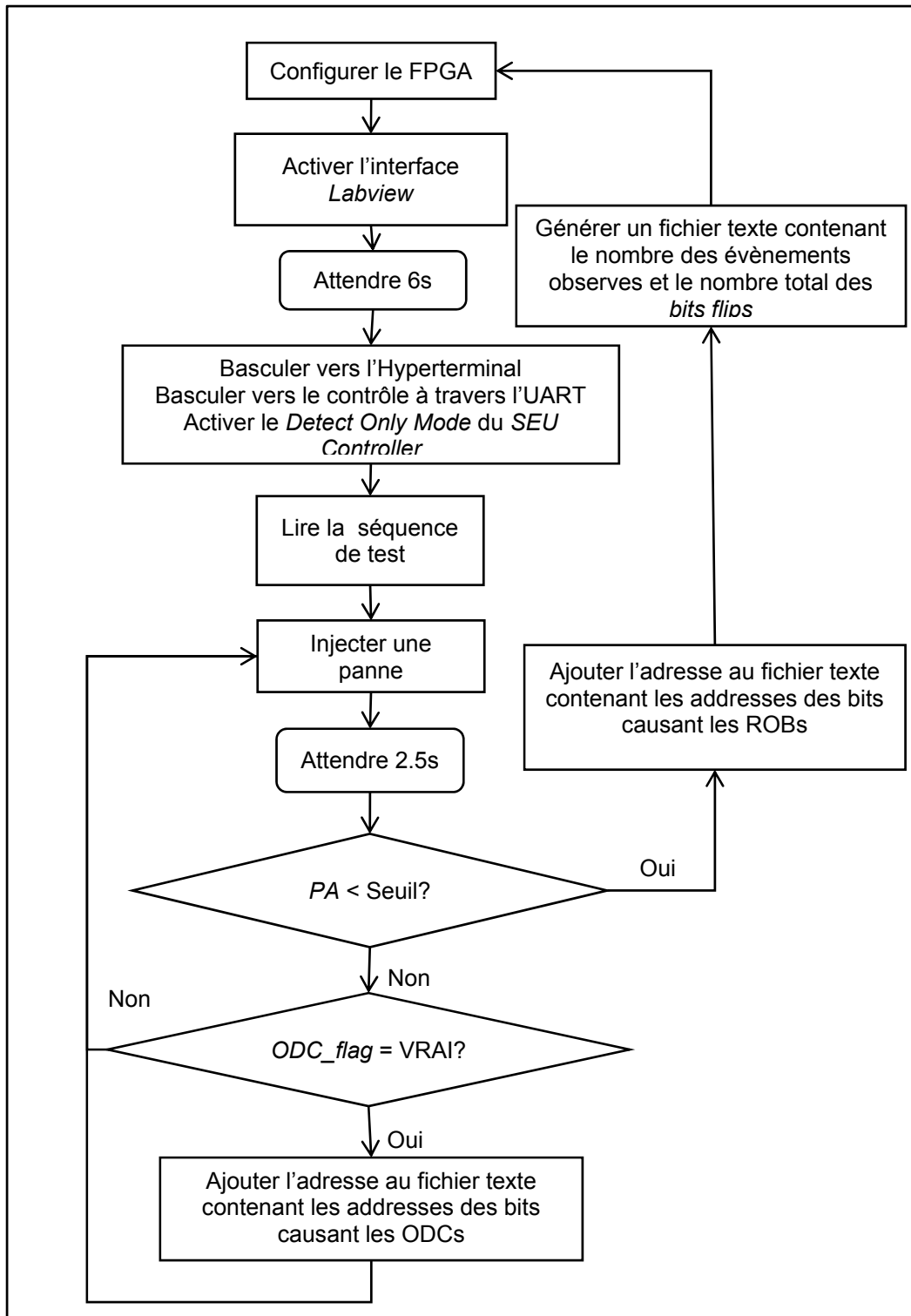


Figure 3.1 Procédure d'injection des pannes

Notons que la limitation majeure est le temps requis par le montage expérimental pour mesurer l'effet du SEU. Notons également qu'aucun effort spécial n'a été dévoué afin d'accélérer l'opération d'injection des pannes elle-même qui prend, en moyenne, 14.2 ms par injection.

Après chaque injection de panne, son effet est contrôlé en vérifiant si elle cause ou non l'un des deux types d'événement, notamment les erreurs ROB ou ODC. Comme indiqué précédemment, les ODC sont détectées si un changement de 55 Hz et plus apparaît au niveau de la différence de fréquence  $F2-F1$ . Les ROB sont, quant à eux, détectés si l'amplitude de la raie  $PA$ , donnant la différence entre les fréquences des RO,  $F2-F1$ , est inférieure à un seuil prédéfini. Lors de la détection d'un événement, l'adresse d'injection de la panne est enregistrée dans le fichier contenant les bits critiques correspondant à l'événement observé (ODC ou ROB). Tel que montré dans la figure 3.1, la procédure d'injection des pannes continue même en cas de détection d'un ODC. Dans le cas de détection d'un ROB, cette procédure s'arrête automatiquement.

L'injection des pannes continue jusqu'à la détection d'un ROB. À ce moment-là, l'interface *Labview* est désactivée dans le but d'arrêter l'acquisition des données à partir de l'analyseur de spectre. Le fichier *Peak\_values2.log* est utilisé pour déterminer le nombre d'ODC détectés durant l'expérimentation d'injection des pannes. Cette étape est critique pour la validation de l'approche proposée vu que les résultats d'émulation par injection des pannes sont comparés aux résultats de TRIUMF selon le nombre des événements observés.

Finalement, trois fichiers texte sont récupérés. Ils contiennent respectivement: 1) la liste des adresses des bits critiques causant un ROB, 2) la liste des adresses des bits critiques causant un ODC, et 3) le nombre d'ODC et le nombre total de pannes injectées durant l'expérimentation.

Parmi les trois tâches principales effectuées par le montage expérimental, seulement celui relié à la mesure (notamment, consistant à contrôler l'analyseur de spectre à distance et générer les données) dépend du design. Par conséquent, seulement cette partie a besoin d'être modifiée afin d'adapter notre procédure d'injection des pannes à d'autres designs. Cette

adaptation serait facilitée si les différents événements produits peuvent être proprement contrôlés par un équipement compatible avec *Labview*. À titre d'exemple, la procédure d'injection a aussi été déployée sur un design plus conventionnel (section 3.5.3). Dans ce cas particulier, la détection a été faite grâce à des circuits de comparaison placés à l'intérieur du FPGA, ainsi qu'à des diodes électroluminescentes sur la carte *Genesys*.

Notons que la procédure d'injection précédemment décrite demeure la même pour les 4 modes d'injection supportés. Le mode d'injection est déterminé lors de la génération de la séquence de test:

- le mode aléatoire (sensibilité égale) choisit de manière aléatoire les bits dans le fichier EBC et les applique s'ils ne font pas partie des bits critiques du *SEU Controller*;
- pour le mode ciblant seulement les bits à 0 ou à 1, les bits à la valeur ciblée sont d'abord extraits du fichier EBC et conservés s'ils ne font pas partie des bits critiques du *SEU Controller*; les bits sont ensuite choisis de manière aléatoire dans le sous-ensemble résultant.

La génération de la séquence de test pour le mode considérant la sensibilité relative est décrite dans ce qui suit.

### **3.4 Procédure d'injection des pannes basée sur la différence de sensibilité relative : Approche de génération des séquences de test considérant la sensibilité relative**

L'approche proposée consiste à générer les séquences de test pour le *SEU Controller* en tenant compte de deux paramètres :  $N$  et  $K = CB0/CB1$ , ce dernier correspondant au ratio du nombre total des bits de configuration à '0' par rapport au nombre total des bits de configuration à '1'.

L'algorithme 3.1, implémenté avec Matlab, génère les séquences de test en se basant sur la différence de sensibilité relative. Tout d'abord, le fichier EBC, généré par *Bitgen*, est lu. Après, les adresses des bits à '1' et ceux à '0' sont extraites et enregistrées dans deux listes: *List1* et *List0*, respectivement. En effet, le fichier EBC du design est lu comme une matrice contenant des '0' et des '1'. Cette dernière est balayée afin de déterminer les coordonnées des

éléments à '1' et à '0'. Une fois que ceci est fait, les coordonnées extraites sont transformées en adresses en format hexadécimal. Ces dernières sont par la suite enregistrées dans *List1* et *List0* selon leurs contenus. Ensuite, les adresses des bits critiques du *SEU Controller* ainsi que leurs bits de configuration sont identifiés et enlevés des *List1* et *List0*. La valeur de  $K$  est par la suite calculée.

Pour chaque séquence de test, *TSCB0* et *TSCB1* adresses sont choisies à partir de *List0* et *List1*, respectivement. Le nombre d'adresses des bits de configuration à '0' dans une séquence de test est défini par l'utilisateur, alors que le nombre des adresses des bits de configuration à '1' est défini selon les valeurs de  $N$  et  $K$ .

#### Algorithme 3.1 Génération des séquences de test

- *TSCB0*: Nombre d'adresses des bits de configuration à '0' dans une séquence de test (défini par l'utilisateur); ce nombre est défini à 2000 pour créer des séquences de test de tailles limitées avec des proportions des valeurs de bit appropriées
  - *TSCB1*: Nombre des adresses des bits de configuration à '1' dans une séquence de test
  - *TS*: Nombre des séquences de test à générer
- 1 Générer le fichier EBC du design
  - 2 Extraire les adresses des bits à '0' et des bits à '1' dans *List0* et *List1*, respectivement
  - 3 Appliquer la procédure de protection du *SEU Controller*
  - 4 Déterminer  $K$
  - 5 For 1 to *TS*
    - 7 Choisir aléatoirement *TSCB0* de *List0*
    - 8 Choisir aléatoirement *TSCB1* de *List1*, où
    - 9  $TSCB1 = TSCB0 \times N / K$
    - 10 Mélanger les *TSCB0* et *TSCB1* adresses des bits dans un fichier
    - 11 Générer la séquence de test en lisant les adresses des bits dans le fichier



Finalement, les *TSCB0* et *TSCB1* adresses des bits de configuration sont aléatoirement rangées dans un fichier texte et la séquence de test est par la suite générée.

Les adresses des bits de configuration dans la séquence de test générée auront alors une distribution de SEU similaire à celle expérimentalement observée à TRIUMF.

### **3.5 Résultats de validation**

La validation des résultats consiste à comparer les résultats obtenus à partir des tests accélérés sous faisceaux de protons effectués à TRIUMF avec ceux obtenus par l'injection des pannes par émulation pour les designs des RO implémentés dans les CLB et les IOB, respectivement. Des résultats obtenus par émulation pour un design plus conventionnel sont aussi présentés.

#### **3.5.1 Résultats de validation pour le design des RO implémentés dans les CLB du FPGA**

Afin de valider l'efficacité de notre approche dans la reproduction des résultats des tests accélérés, la métrique  $Z_T$  a été estimée à partir des résultats d'injection des SEU par émulation puis elle a été comparée à celle obtenu à TRIUMF. Afin de valider expérimentalement l'approche proposée, une campagne d'injection des pannes a été effectuée où les séquences de test sont générés selon la méthode décrite dans la section précédente.

Malgré que les fichiers de configuration utilisés pour les tests sous radiation ainsi que pour les tests d'injection des pannes ne sont pas les mêmes vu que le design utilisé pour les tests accélérés contient seulement le design des RO alors que celui utilisé pour les expérimentations d'injection des pannes contient en plus des RO le *SEU Controller*, plusieurs mesures ont été prises pour minimiser l'impact de la présence du *SEU Controller* avec les RO. Le placement identique des pins et les contraintes de l'emplacement relatif (RLOC) des RO sont appliqués dans les deux designs afin de préserver les mêmes ressources utilisées par les RO dans les deux DUT. En particulier, les contraintes RLOC forcent la

création d'une forme 'S' dans les ressources du FPGA, limitant ainsi les changements de placement et routage. En outre, afin de limiter les différences entre les deux designs, le *SEU Controller* a été forcé d'occuper des ressources séparées et inutilisées par les RO. Les deux RO obtenus pour les deux designs après l'application des différentes contraintes étaient quasi-identiques, avec et sans *SEU Controller*. Notez que malgré l'application de toutes les contraintes mentionnées, la présence du *SEU Controller* dans le design d'émulation impose une légère différence entre les deux designs. En effet, 3479 LUT utilisés par le design d'émulation parmi un total de 3600 LUT utilisés par le design original sont préservés, c'est-à-dire, à peu près 97% des ressources du design original ont été préservées.

Les résultats obtenus ainsi que ceux issus des tests accélérés à TRIUMF sont donnés par le Tableau 3.3. À TRIUMF, pour les tests sous faisceaux de protons, 42485 pannes sont observées dans le design. Pour les expérimentations d'émulation par injection des pannes, 82922 pannes sont injectées afin de valider l'approche proposée, ce qui signifie qu'à peu près le double des SEU détectés à TRIUMF a été injecté. Les colonnes ODC et ROB du Tableau 3.3 donnent le nombre des ODC et ROB détectés après chaque série d'expérimentations, respectivement. La colonne  $Z_T$  est calculée comme le ratio des valeurs données par les deux colonnes précédentes.

Tableau 3.3 Résultats de validation des tests pour le design des RO implémentés dans les CLB

	ODC	ROB	$Z_T$
<b>Campagne d'injection des pannes (82992 pannes injectées)</b>			
Série (1)	58	68	0.85
Série (2)	33	68	0.49
Série (3)	59	68	0.87
Série (4)	67	68	0.99
Total/Moyenne	217	272	0.80
<b>Tests accélérés, sous radiation de protons à TRIUMF (42485 pannes observées)</b>			
TRIUMF	56	68	0.82

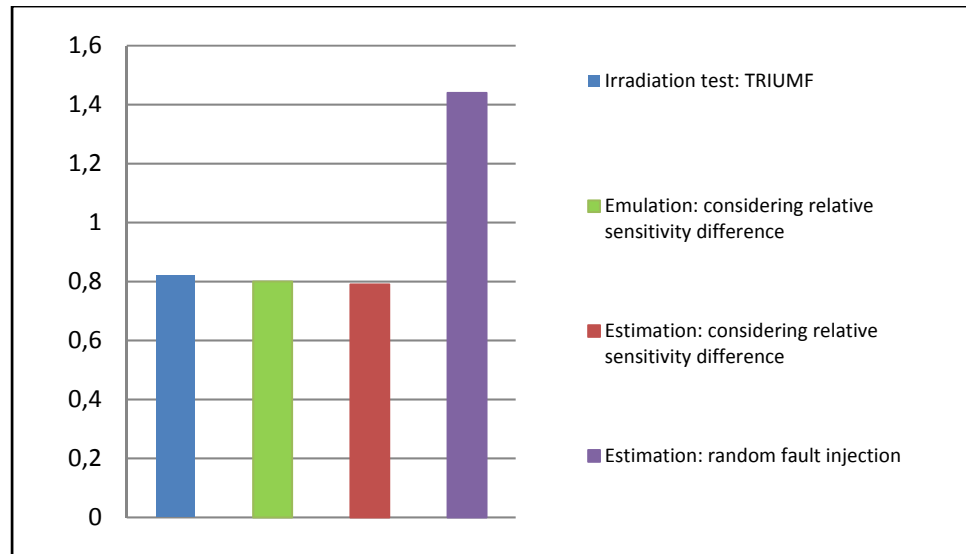


Figure 3.2 Comparaison de  $Z_T$  obtenu à partir de l'estimation, ainsi qu'à partir des différentes campagnes d'injection des pannes  
Tirée de Souari et al. (2016)

Tel que montré par la figure 3.2, la moyenne de la valeur de  $Z_T$ , obtenue à partir de la campagne des expérimentations d'injection des pannes considérant la différence de sensibilité relative, est égale à 0.80. Cette valeur est proche de celle obtenue à partir des tests accélérés qui est égale à 0.82. Ceci correspond à une erreur de -3.1 % qui est une claire indication que la procédure d'émulation proposée offre des résultats réalistes proches de ceux obtenus sous le test de bombardement par protons. La comparaison de  $Z_T$  obtenue avec celle estimée (considérant aussi la différence de sensibilité relative) donne une erreur de 1.3 %, ce qui est une autre indication de l'efficacité de l'approche proposée. De plus,  $Z_T$  estimée pour l'injection aléatoire des pannes donne une erreur de 75 % quand comparée à  $Z_T$  obtenue à TRIUMF. Rappelons ici que la valeur de  $Z_T$  déterminée par estimation est calculée en se basant sur l'équation (3.1) et les estimations des nombres des bits critiques données dans la section 3.2. La valeur de  $Z_T$  extraite par émulation consiste à calculer simplement le ratio de nombre des ODC par le nombre des ROB observés pour chaque série d'injection des pannes.

Dans l'objectif de valider les résultats obtenus par l'émulation par injection des pannes, la procédure de la validation statistique décrite dans (Quinn et Wirthlin, 2015), est suivie. Cette dernière consiste essentiellement en trois étapes: l'estimation de la sensibilité du design en

utilisant l'émulation des pannes, l'estimation de la sensibilité du design en utilisant les sources de radiations et la comparaison des résultats des deux estimations.

1) Estimation de la sensibilité du design en utilisant l'émulation des pannes

Selon (Quinn et Wirthlin, 2015), l'objectif de l'émulation est d'estimer  $r$  qui est la probabilité d'avoir une erreur fonctionnelle en effectuant plusieurs injections de pannes. L'estimateur de la probabilité maximale (*maximum likelihood estimator*),  $\hat{r}$ , est estimé selon l'équation suivante :

$$\hat{r} = k/n = 489/82922 = 0.00589 \quad (3.2)$$

où  $k$  est le nombre de pannes injectées causant un dysfonctionnement du système et  $n$  est le nombre total de pannes injectées. Selon le tableau 3.3,  $k$  est égale à 489 (217 ODC et 272 ROB) et  $n$  est égale à 82922. La taille de l'intervalle de confiance à 95 % est donnée par l'expression suivante :

$$-1.96 \sigma + \mu \leq r \leq 1.96 \sigma + \mu \quad (3.3)$$

où :

$$- \mu = \hat{r} \quad (3.4)$$

$$- \sigma = \sqrt{\frac{k}{n^2} \left(1 - \frac{k}{n}\right)} \quad (3.5)$$

La valeur calculée de l'intervalle de confiance à 95 % est alors :

$$0.00536 \leq r \leq 0.00641 \quad (3.6)$$

2) L'estimation de la sensibilité du design en utilisant les sources de radiation

Selon (Quinn et Wirthlin, 2015), cette valeur est obtenue en divisant le nombre des erreurs fonctionnelles par le nombre total de pannes injectées. Dans notre cas, le nombre

d'erreurs est 124 (56 ODCs et 68 ROBs), alors que le nombre total des pannes injectées est 21485. En effet, le nombre total de pannes détectées à TRIUMF durant les expérimentations de bombardement par protons du design des RO dans les CLB est 42485. Toutefois, vu que les BRAM ne sont pas utilisés par notre design et le mécanisme d'injection des pannes à savoir le *SEU Controller* est incapable d'injecter des SEU dans les cellules de BRAM, les pannes affectant les cellules de BRAM lors des tests sous radiation ne doivent pas être considérées dans l'estimation de la sensibilité du design sous radiation. La valeur estimée de la sensibilité du design sous radiation est 0.00567 et l'intervalle de confiance à 95 % est :

$$0.00468 \leq r \leq 0.00667 \quad (3.7)$$

### 3) Comparaison des deux estimations

Les résultats d'émulation des pannes tombent dans l'intervalle de confiance à 95 % obtenu à partir des résultats de test sous radiation. Ceci confirme que notre approche est valide.

Le tableau 3.4 résume les résultats de la validation statistique de l'approche d'injection des pannes par émulation proposée.

Tableau 3.4 Résultats de validation statistique de l'approche d'injection des pannes proposée

<b>Expériences</b>	$\mu$	<b><i>r</i> à un intervalle de confiance à 95 %</b>
Émulation en considérant la sensibilité relative	0.00589	[0.00536, 0.00641]
TRIUMF	0.00567	[0.00468, 0.00667]

### 3.5.2 Résultats de validation pour le design des RO implémentés dans les IOB du FPGA

Dans le but de renforcer les résultats de validation, l'approche proposée est appliquée sur un autre circuit testé sous radiation de protons à TRIUMF (Tazi, 2014) durant une deuxième série d'expérimentations. Tel que mentionné au chapitre précédent, ce design consiste en une paire de RO implémentés dans 340 blocs d'entrée/sortie (IOB) au lieu de dans les blocs CLB du FPGA. Trente expérimentations sous radiation de protons ont été effectuées sur ce circuit. Selon (Tazi et al., 2014), 77 ODC et 30 ROB ont été détectés pour ce circuit lors des expérimentations à TRIUMF. Durant les expérimentations d'injection des pannes par émulation, une technique automatisée de détection de seuil est utilisée pour la détection des ODC (seuil de +/- 100 Hz), vu que le niveau de bruit sur les RO dans le IOB est plus élevé que celui dans les CLB. En utilisant cette technique de détection automatique, le nombre des ROB observés à TRIUMF restera le même alors que le nombre des ODC sera réduit à 57. Cette dernière sera alors utilisée pour effectuer des comparaisons avec les résultats obtenus par l'émulation par injection des pannes.

Afin de valider notre approche, 5 séries de 30 expérimentations d'injection des pannes sont effectuées et leurs résultats sont comparés à ceux obtenus à TRIUMF. Pour les expérimentations d'injection des pannes, il a été supposé que  $N = 1.89$  vu que c'est la valeur estimée durant la deuxième série d'expérimentations à TRIUMF sur le design des RO dans les IOB. De même manière que pour les expérimentations des RO dans les CLB, la valeur de  $N$  est estimée en considérant toutes les pannes dans les bits de configuration. De plus, afin de montrer l'importance de la considération de la différence de sensibilité relative dans la procédure d'émulation, 5 autres séries de 30 expérimentations aléatoires ( $N = 1$ ) d'injection des pannes sont effectuées et leurs résultats sont comparés avec les deux autres. Le Tableau 3.5 résume les résultats obtenus à partir des trois campagnes : tests accélérés, tests par injection des pannes en considérant la différence de sensibilité relative et tests aléatoires d'injection des pannes. L'écart-type de  $Z_T$  a été aussi calculé pour la campagne d'injection des pannes en considérant la sensibilité relative ainsi que pour la campagne d'injection aléatoire des pannes.

Le ratio  $Z_T$  de la campagne d'injection des pannes en considérant la sensibilité relative est égal à 1.63, alors que la valeur obtenue à partir de la deuxième série d'expérimentations à TRIUMF est égal à 1.90. Ceci correspond à une erreur de -14%. Cette dernière est plus élevée (en valeur absolue) que celle obtenue à partir de la première série d'expérimentations (-3.1%), mais elle reste une valeur acceptable pour valider l'approche proposée surtout quand elle est comparée à l'erreur obtenue à partir de la campagne aléatoire d'injection des pannes, à savoir ( $Z_T = 2.71$ ). Les résultats obtenus montrent que la considération de la différence de sensibilité relative dans l'émulation donne des résultats plus proches de ceux donnés par les tests sous radiation quand comparés à ceux issus des expérimentations aléatoires d'injection des pannes. L'erreur élevée obtenue pour la deuxième série d'expérimentations n'a pas été investiguée en profondeur, mais elle peut être partiellement expliquée par le type des ressources servant à implémenter les RO, à savoir les IOB. En effet, il s'avère que les mécanismes de défauts menant aux ODC dans les CLB sont différents de ceux dans les IOB, où les mécanismes s'apparentant à des changements au niveau des standards d'entrée/sorties s'ajoutent à ceux liés aux capacités parasites additionnelles (Tazi et al., 2014).

Tableau 3.5 Résultats de validation des tests pour le design des RO implémentés dans les IOB

	ODC	ROB	$Z_T$	Écart-type
<b>Tests accélérés, sous radiation de protons à TRIUMF</b>	57	30	1.90	-
<b>Injection des pannes considérant la sensibilité relative</b>				
Série 1	27	30	0.90	0.48
Série 2	53	30	1.77	
Série 3	67	30	2.23	
Série 4	49	30	1.63	
Série 5	49	30	1.63	
Total/Moyenne	245	150	1.63	
<b>Injection aléatoire des pannes</b>				
Série 1	115	30	3.83	0.98
Série 2	83	30	2.77	
Série 3	43	30	1.43	
Série 4	61	30	2.03	
Série 5	104	30	3.47	
Total/Moyenne	406	150	2.71	

L'ajout de ces nouveaux mécanismes peut amener une plus grande variabilité au niveau des résultats. Dans ce contexte, le nombre plus restreint d'expérimentations en tests accélérés (30 au lieu de 68) a pu affecter l'amplitude de l'erreur.

D'autres observations intéressantes peuvent être faites en comparant les résultats de l'injection aléatoire des pannes avec ceux obtenus en considérant la différence de sensibilité relative. La première est qu'en moyenne l'injection aléatoire des pannes produit à peu près 1.7 fois plus d'ODC pour un nombre donné de ROB. Ceci est une indication claire que la différence de sensibilité relative a aussi un impact significatif sur le type des événements observés dans les IOB. La seconde observation qui peut être faite consiste au fait que l'écart-type de  $Z_T$  est deux fois plus élevé pour l'injection aléatoire des pannes (0.98) que pour l'injection en considérant la différence de sensibilité relative (0.48). Cette dernière observation indique que la différence de sensibilité relative a un impact sur la variabilité des résultats. Les résultats de ces expérimentations ne sont pas validés par la procédure de validation décrite dans (Quinn et Wirthlin, 2015) vu que l'erreur entre les résultats d'injection des pannes en considérant la sensibilité relative et ceux de TRIUMF était un peu élevée et par la suite l'intervalle de confiance à 95 % de l'estimation de la sensibilité du design en utilisant l'émulation ne tombe pas dans celui calculé pour les résultats de TRIUMF. La comparaison avec les résultats de la campagne aléatoire d'injection des pannes était nécessaire pour montrer l'efficacité de l'approche proposée.

### 3.5.3 Résultats de validation pour un design plus conventionnel

Dans l'objectif de montrer l'importance de considérer la différence de sensibilité relative entre les bits de configuration dans la procédure d'émulation et son applicabilité pour des circuits autres que ceux basés sur des RO, l'approche proposée (considérant  $N = 2.11$ ) est appliquée sur le circuit B08 des benchmarks ITC99 (Quinn et al., 2015), et les résultats sont comparés à ceux obtenus à partir d'une injection aléatoire des pannes pour le même circuit. Les résultats obtenus sont donnés par le Tableau 3.6. Dans ce cas particulier, seulement les erreurs logiques détectées sur les sorties du circuit, en appliquant une séquence fonctionnelle de test, sont considérées.



Pour le même nombre de pannes injectées, l'approche proposée avec  $N=2.11$  génère, en moyenne, à peu près 2.3 fois plus d'erreurs que par une injection aléatoire uniforme ( $N=1$ ) de pannes. Considérant un intervalle de confiance de 95 %, ce facteur peut varier de 1.36 jusqu'à 4.52.

Tableau 3.6 Résultats de campagnes aléatoires d'injection des pannes le circuit B08

	<b>Ratio de sensibilité relative entre les bits de configuration à '1' et ceux à '0'</b>	
	$N = 1$	$N = 2.11$
<b>Nombre total des bit flips (B)</b>	101 626	101 626
<b>Nombre total des erreurs observées (E)</b>	13	30
<b>Ratio (E/B)</b>	$1.27 \times 10^{-4}$	$2.95 \times 10^{-4}$

Ces résultats montrent l'importance de considérer la différence de sensibilité relative dans l'injection des pannes dans le but d'obtenir des estimations plus fiables de la sensibilité d'un circuit en comparaison avec une injection aléatoire des pannes. Les résultats obtenus suggèrent qu'adopter la dernière approche d'injection des pannes peut mener à une sous-estimation significative de la sensibilité relative d'un design face aux radiations.

### 3.5.4 Discussion

Ces résultats montrent clairement que l'approche proposée basée sur la considération de la différence de sensibilité relative dans le processus de l'émulation est efficace pour produire des résultats plus réalistes que ceux générés par une injection aléatoire. En effet, en considérant que les bits de configuration à '1' sont plus sensibles que les bits à '0' dans la procédure de génération des séquences de test, l'approche d'injection des pannes proposée donne des résultats plus proches de ceux obtenus par les tests accélérés que ceux donnés par l'injection aléatoire, surtout pour les expérimentations effectués sur le design des RO dans les CLB.

La considération de la différence de sensibilité relative entre les bits à '0' et ceux à '1' a été dans un premier temps justifiée par le type des événements d'intérêt (ODC et ROB) et par le fait que les ODC sont particulièrement sensibles aux contenus des bits de configuration, et ce

pour le type de designs ciblés lors des tests accélérés (RO dans les CLB et IOB). Mais comme il s'agit de designs synthétiques, il était important de montrer la pertinence de considérer la différence de sensibilité pour des designs plus conventionnels. Un premier pas en ce sens a été franchi avec l'utilisation du circuit B08. L'hypothèse que nous formulons à ce stade-ci pour expliquer la différence observée se situe au niveau de la répartition des bits à 1 et à 0 et de leur testabilité relative. En effet, nos résultats jusqu'ici suggèrent que la majorité de bits à 0 critiques se retrouvent dans les LUT alors que les bits à 1 critiques peuvent se retrouver tant dans les LUT que dans les cellules contrôlant le routage des signaux. Or une analyse préliminaire suggère que les bits dans les LUT sont plus difficiles à tester (i.e. détecter une inversion de bit) que ceux contrôlant le routage, car la détection de leur inversion nécessite qu'on y accède, alors que la détection d'un bit de contrôle inversé s'apparente à celle d'une panne collée-à sur le signal routé (Li et al., 2003). À titre d'exemple, considérons une porte XOR à 4 entrées réalisée à l'aide d'un LUT à 4 entrées également, dont les entrées et la sortie sont accessibles aux broches du FPGA. Pour chacun des 16 bits de mémoire de la LUT, il existe un et seul vecteur de test permettant d'y détecter une inversion de bit, chaque vecteur ne détectant que l'inversion d'un seul bit. Tous les 16 vecteurs sont donc nécessaires pour couvrir l'ensemble des 16 bits de LUT. Pour ce qui est de la détection de l'inversion d'un bit contrôlant le routage, il existe pour chaque signal routé 8 vecteurs pouvant détecter une panne collée-à-0 ainsi que 8 vecteurs pouvant détecter une panne collée-à-1, un seul vecteur pour chaque type de pannes étant nécessaire. Cet exemple simple permet d'illustrer que la difficulté relative de détection de l'inversion d'un bit de LUT tient au nombre beaucoup plus restreint de vecteurs permettant la sortie du bit à tester de la LUT et la détection éventuelle de son inversion. Notons de plus que pour les circuits plus complexes où les signaux doivent traverser des réseaux combinatoires et des registres pour apparaître en sortie, il y a risque de masquage des pannes dues aux inversions. Moins il y a de vecteurs permettant à un bit inversé de se manifester à la sortie de la ressource fautive (LUT ou interrupteur de routage), moins grandes sont les chances qu'il puisse se propager vers la sortie et être détecté. Ceci devrait normalement se traduire par une liste plus longue de vecteurs à appliquer afin de détecter toutes les inversions de bit possible dans les LUT et où certaines inversions nécessitent une séquence précise de vecteurs pour qu'elles se

manifestent et soient propagées jusqu'aux sorties. Dans le cas où il n'est possible d'appliquer tous les vecteurs ou séquences de vecteurs nécessaires, certaines inversions peuvent demeurer non détectées.

À cause de l'accès difficile aux laboratoires permettant d'assurer les tests accélérés et les coûts élevés de ces tests, la validation de l'approche proposée est limitée aux deux circuits des RO et les résultats des expérimentations relatives. En effet, ils sont les seuls circuits testés sous radiation et pour lesquels on a les résultats nécessaires pour comparer avec ceux générés par l'approche proposée. La validation de l'approche proposée avec d'autres types de circuits fait partie des recherches proposées pour le futur. De plus, des investigations de la différence de sensibilité relative sur d'autres designs plus conventionnels seraient requises afin de valider l'hypothèse liée à la testabilité. D'autres investigations permettraient de valider expérimentalement le lien entre le type des événements (ROB et ODC) et le type des ressources (LUT, routage, BRAM, etc.).

### **3.6 Conclusion**

Ce chapitre a présenté une méthodologie permettant d'obtenir des résultats plus réalistes par la procédure d'injection des pannes par émulation dans les FPGA à base de mémoire SRAM. En effet, une approche d'injection des SEU par émulation, considérant la différence de sensibilité relative entre les bits à '1' et ceux à '0', est proposée. La considération de cette information est justifiée en investiguant l'impact des valeurs des bits inversées sur le comportement erroné observable.

L'approche d'injection de pannes considérant la différence de sensibilité relative a été validée expérimentalement et statistiquement par des tests accélérés sous des faisceaux de protons à TRIUMF. La comparaison des résultats des deux campagnes donne une erreur relative atteignant 3.1 %. La comparaison des résultats d'une injection conventionnelle aléatoire de pannes avec ceux de TRIUMF donne une erreur pouvant s'élever à 75 %. Une autre valeur ajoutée de l'approche proposée a été montrée quand elle est appliquée sur le circuit B08 des Benchmarks ITC99. Elle consiste à détecter 2.3 fois plus d'erreurs qu'une

injection aléatoire appliquée sur le même circuit, ce qui suggère qu'ignorer la différence de sensibilité relative dans la procédure d'injection des pannes peut mener à une sous-estimation de la sensibilité du DUT. À notre connaissance, nous sommes les premiers à considérer la différence de sensibilité relative entre les bits de configuration dans la procédure d'émulation afin d'obtenir un tel réalisme dans les résultats en comparaison avec ceux obtenus par les tests accélérés.

L'idée de l'obtention des résultats plus réalistes d'injection des pannes dans ce chapitre consiste à exploiter le fait que le contenu d'un bit de configuration a un impact sur la variation de la sensibilité des différents bits face aux SEU. Dans le même contexte, le chapitre suivant explorera l'optimisation de l'injection des pannes par émulation en exploitant non seulement l'effet des contenus des bits de configuration sur la variation de la sensibilité de ces derniers mais aussi celui dépendant des types des ressources configurées.

## CHAPITRE 4

### OPTIMISATION D'ÉMULATION DES SEU DANS LES FPGA À BASE DE MÉMOIRE SRAM

#### 4.1 Introduction

Dans le chapitre précédent, une nouvelle approche d'injection des pannes par émulation permettant de générer des résultats les plus réalistes possibles a été présentée. L'idée consistait à considérer la différence de sensibilité relative entre les bits de configuration à '1' et ceux à '0' lors de la génération des séquences de test. L'efficacité de l'approche proposée a été validée en comparant les résultats obtenus avec les résultats issus des expérimentations sous radiation ainsi que ceux obtenus par les expérimentations d'injection aléatoire des pannes par émulation.

Dans ce chapitre, on présente une nouvelle approche d'optimisation pour la procédure d'injection des pannes dans les FPGA à base de mémoire SRAM. L'approche proposée privilégie l'injection de pannes dans des sous-ensembles spécifiques des bits de configuration définis en fonction de leur contenu ainsi que des types des ressources du FPGA qu'ils configurent. La nouvelle approche permet de maximiser le nombre de bits critiques inversés durant la procédure d'injection des pannes. Elle permet aussi de maximiser la précision d'estimation du nombre de bits critiques. Le circuit cible des tests pour valider notre approche est celui des RO dans les CLB. Ce choix est justifié par l'intérêt de notre groupe par l'étude de l'impact des SEU en termes d'ODC. Le design des RO était un circuit idéal pour atteindre cet objectif vu que c'était facile de détecter les ODC en contrôlant simplement la différence entre les fréquences des deux RO. Néanmoins, l'approche proposée peut être applicable sur n'importe quel design. Notez donc que dans notre cas, un bit de configuration est déclaré critique s'il cause soit un ODC ou un ROB quand son contenu est inversé.

Le chapitre 4 est organisé comme suit. La section 4.2 présente la méthodologie d'injection des pannes proposée basée sur l'injection des SEU dans des sous-ensembles spécifiques des bits de configuration. Les résultats montrant la pertinence d'identifier ces sous-ensembles et

évaluant leurs sensibilités face aux SEU sont présentés dans la section 4.3. La méthode utilisée pour estimer l'erreur dans le calcul du nombre de bits critiques est détaillée dans la section 4.4. La comparaison des résultats obtenus se fait dans la section 4.5. Enfin, la section 4.6 conclut ce chapitre.

## 4.2 Méthodologie d'injection des pannes proposée

Le nombre de bits de configuration des FPGA croît constamment. Il a d'ailleurs franchi la barre du milliard dans la dernière famille de Xilinx. Face à ce constat, il devient impératif d'accélérer le processus aléatoire d'injection des pannes dans les FPGA. À cette fin, nous proposons de prioriser l'injection des pannes dans les parties les plus critiques du FPGA. Afin de prioriser l'injection des pannes, les trois étapes suivantes sont à suivre:

1) classification des bits de configuration selon les ensembles suivants:

- **(LUT, 1)/(LUT, 0):** bits à 1/0 configurant les LUT utilisés par le design cible;
- **(!LUT, BG, 1)/(!LUT, BG, 0):** bits à 1/0 configurant des ressources (autres que les LUT utilisés par le design cible) identifiées comme potentiellement critiques par *Bitgen* (BG);
- **(!LUT, !BG, 1)/(!LUT, !BG, 0):** bits à 1/0 configurant des ressources (autres que les LUT utilisés par le design cible) non identifiées comme potentiellement critiques par *Bitgen*. Notez qu'en se basant sur les résultats obtenus, tous les bits configurant les LUT utilisés par le design cible sont identifiés comme potentiellement critiques par *Bitgen*. En réalité, pas tous ces bits sont critiques. En effet, pour le design particulier des RO où les LUT à 6 entrées du Virtex-5 contiennent chacun un inverseur seulement, seulement 2 des 64 bits de chaque LUT sont réellement critiques. Nous avons de plus observé que ces LUT implémentant un inverseur contiennent 32 '1' et 32 '0', même si un seul '1' serait nécessaire;

2) estimation du nombre de bits critiques de chaque ensemble en injectant de façon aléatoire des pannes dans les bits de chaque ensemble. Cette méthode permet d'identifier les zones les plus critiques du FPGA, ainsi que l'estimation du nombre total de bits critiques dans le FPGA ;

3) priorisation d'injection des pannes dans les zones les plus critiques identifiées.

Les fichiers EBC et EBD (générés par l'outil *Bitgen* de Xilinx) sont utilisés pour assurer l'étape de la classification des bits de configurations. Comme il a été mentionné précédemment, ces fichiers contiennent, respectivement, les données de configuration du FPGA et le masque des données des bits de configuration essentiels du design.

L'identification des bits de configuration qui appartiennent ou non aux LUT était un défi vu que Xilinx ne divulgue pas l'information sur la relation des bits de configuration avec les ressources qu'ils configurent. Pour surmonter cet obstacle, nous avons développé une méthode permettant de déterminer tous les bits de configuration configurant les LUT d'une façon précise. Cette méthode est détaillée dans le chapitre suivant.

En quelque sorte, cette méthode est similaire à la procédure de protection du *SEU Controller* décrite dans le chapitre 2.

### 4.3 Évaluation de la sensibilité des différents ensembles des bits de configuration

Les premiers résultats d'injection des pannes dans les différents sous-ensembles du design des RO sont donnés par le tableau 4.1. Les colonnes de 1 à 5 donnent, respectivement, le sous-ensemble cible, le nombre total des bits de configuration (*Total Number of Configuration Bits*, TNCB) dans ce sous-ensemble spécifique, le nombre des *bitflips* injectés (*Number of Injected Bitflips*, NIB), le nombre de bits critiques inversés (*Number of Flipped Critical Bits*, NFCB), et le ratio des nombres de bits critiques détectés (*Ratio of Observed Critical Bits*,  $ROCB = NFCB/NIB$ ). LUT correspond aux bits de configuration configurant les LUT utilisés par le design cible, tandis que !LUT correspond aux bits de configuration configurant d'autres ressources. Rappelons que BG correspond aux bits de configuration identifiés comme essentiels par *Bitgen*. !BG correspond aux bits non identifiés par *Bitgen*.

Ces résultats sont obtenus pour une injection des pannes en mode SEU où chaque bit inversé est corrigé par la suite (plus précisément inversé à nouveau pour retrouver sa valeur originale) avant l'injection d'un autre *bit flip*.

Tableau 4.1 Résultats d'injection des pannes dans les différents sous-ensembles du design des RO

Sous-ensemble cible		TNCB	NIB	NFCB	ROCB (%)	
LUT	1	115,168	1,447	44	3.0	
	0	115,232	1,406	44	3.1	
!LUT	BG	1	10,993	10,993	10 879	99.0
		0	47,945	1,425	811	56.9
	!BG	1	5,613	5,613	0	0.0
		0	10,711 417	200, 000	3	0.0015

À partir des résultats de la première campagne d'injection des pannes, on peut observer que les sous-ensembles les plus critiques sont les suivants :

- (!LUT, BG, 1): Les bits à 1 configurant d'autres ressources (autres que les LUT utilisés par le design cible) identifiés comme potentiellement critiques par *Bitgen* où 99% de ces bits sont critiques;
- (!LUT, BG, 0): Les bits à 0 configurant d'autres ressources identifiés comme potentiellement critiques par *Bitgen* où 56.9 % de ces bits sont critiques;
- (LUT, BG, 0): Les bits à 0 configurant les LUT utilisés par le design cible où 3.1 % de ces bits sont critiques; ce ratio confirme que seulement 1 des 32 '0' dans les LUT implémentant un inverseur est critique;
- (LUT, BG, 1): Les bits à 1 configurant les LUT utilisés par le design cible où 3.0 % de ces bits sont critiques; ce ratio confirme que seulement 1 des 32 '1' dans les LUT implémentant un inverseur est critique.

Ces résultats montrent que l'efficacité de *Bitgen* à identifier les bits critiques varie considérablement en fonction du type des ressources configurées ainsi que le contenu des bits de configuration. Les valeurs obtenues varient de 99 % pour les bits à 1 configurant d'autres ressources (!LUT) à 3 % pour les bits configurant les LUT utilisés par le design cible. Le ratio de 3 % obtenu pour les LUT représente le pire scénario vu que chaque LUT implémente un seul inverseur et ce ratio peut atteindre 100 % en cas d'utilisation totale des LUT, notamment en cas d'implémentation de fonctions combinatoires à 6 entrées. En se basant sur



des statistiques fournis par (Altera, 2007), on estime que le ratio moyen des bits critiques dans les LUT utilisés est d'à peu près 46 %.

Il est important de noter que le nombre des pannes injectées afin d'obtenir les résultats du tableau 4.1 n'est pas optimisé. Par exemple, performer une injection exhaustive de pannes n'est pas nécessaire pour estimer les valeurs de ROCB (comme on a fait pour les sous-ensembles (!LUT, BG, 1) et (!LUT, !BG, 1)) car nous pouvons les prédire en se basant sur les résultats donnés par *Bitgen*. On s'attend à ce que *Bitgen* donne des résultats similaires pour tous les designs, vu que les bits identifiés comme potentiellement critiques par *Bitgen* sont supposés avoir de fortes chances d'être critiques tandis que ceux non identifiés par *Bitgen* sont supposés ne pas générer d'erreurs.

Notons enfin, que les trois événements observés lors de l'injection des pannes dans le sous-ensemble (!LUT, !BG, 0) confirme que *Bitgen* n'identifie pas précisément les bits critiques. Ceci justifie l'utilisation du fichier EBC dans la procédure de protection du *SEU Controller* (chapitre 2) et vient appuyer les résultats présentés dans (Hobeika et al., 2014) montrant les limitations de *Bitgen*.

#### 4.4 Détermination de l'erreur d'estimation du nombre des bits critiques (CBEE)

Les résultats dans le tableau 4.1 sont utilisés pour déterminer les résultats d'estimation préliminaires pour les approches proposées données par le tableau 4.2, où les colonnes de 1 à 5 représentent respectivement l'approche d'injection des pannes, le nombre des *bit flips* injectés (NIB), le nombre des bits critiques inversés (NFCB), le ratio des bits critiques détectés (ROCB) et l'erreur d'estimation de nombre des bits critiques (*Critical Bit Error Estimate*, CBEE). Le paramètre CBEE est dérivé de l'intervalle de confiance binomial (BCI) qui est estimé en utilisant l'approche conventionnelle basée sur l'approximation normale de la distribution d'erreur (Faure, Velazco et Peronnard, 2005).

Dans notre cas, un niveau de confiance de 95 % est utilisé menant à un BCI de  $[p-z\sigma, p+z\sigma]$ , où  $p = \text{ROCB}$ ,  $\sigma = \text{sqrt}(p(1-p)/n)$ ,  $z = 1.96$ , et  $n = \text{NIB}$ . CBEE est égal à  $\pm z\sigma$   $(\text{TNCB} - \text{NIB}) / (\text{TNCB} * p)$ .

En utilisant les nombres donnés par le tableau 4.1, on déduit les résultats donnés par l'approche « Classification » dans le tableau 4.2, avec NIB = 220,884, NFCB = 11,781 et une erreur d'estimation des bits critiques (CBEE) de  $\pm 7.5\%$ . Il est important de noter qu'aucun effort d'optimisation en termes de maximisation du nombre de bits critiques inversés ou en termes de réduction de l'erreur d'estimation de leur nombre n'a été investi dans ce premier ensemble de résultats.

Tableau 4.2 Comparaison des résultats des différentes approches

<b>Approche</b>	<b>NIB</b>	<b>NFCB</b>	<b>ROCB (%)</b>	<b>CBEE (%)</b>
<b>Classification</b>	220,884	11,781	5.3	$\pm 7.5$
<b>Random, #1</b>	87,418	376	0.43	$\pm 10.0$
<b>Random, #2 (est.)</b>	87,418	362	0.41	$\pm 10.2$
<b>Proposed, #1 (est.)</b>	87,418	39,057	44.7	n/a
<b>Proposed, #2 (est.)</b>	87,418	38,166	43.7	$\pm 1.2$
<b>Proposed, #3 (est.)</b>	87,418	30,970	35.4	$\pm 1.1$
<b>Bitgen, #1 (est.)</b>	87,418	9,200	10.5	$\pm 3.2$

Une autre campagne d'injection des pannes a été faite avec le même montage expérimental en injectant des pannes aléatoires dans les bits de configuration. Les résultats de cette campagne aléatoire d'injection des pannes sont donnés par le tableau 4.2 par l'approche ayant comme nom "Random, #1". Ils sont obtenus pour une valeur de NIB de 87418 (choisie de manière arbitraire). Ces résultats ont été obtenus pour une campagne d'injection des pannes dans le mode d'accumulation où le bit inversé n'est pas corrigé avant d'injecter une prochaine panne et seront utilisés comme référence dans la suite de l'analyse. L'utilisation du mode accumulation (représentative des expérimentations de bombardement sous radiation) permet une comparaison avec les résultats obtenus en mode SEU. Les résultats présentés

sous le nom “Random, #2” dans le Tableau 4.2 donnent les valeurs NFCB et ROCB estimées dans le mode SEU. Ces derniers résultats sont obtenus en assumant que les pannes sont distribuées sur les bits de configuration des différents sous-ensembles selon leurs proportions respectives par rapport au nombre total des bits de configuration.

Il est remarquable que les valeurs de NFCB et ROCB soient légèrement plus élevées dans le mode accumulation que ceux obtenus dans le mode SEU. Ceci est peut-être dû au fait que quelques bits de configuration sont individuellement non critiques et deviennent critiques quand d’autres bits sont inversés, tel que rapporté par (Tazi et al., 2014). Des expérimentations additionnelles sont requises pour vérifier si le mode accumulation donne toujours un nombre plus élevé de nombre des bits critiques en considérant une valeur de CBEE de 10%. Cette partie peut être investiguée comme travail futur mais jusqu’à maintenant la différence entre les deux modes ne semble pas significative.

#### **4.5 Résultats en fonction du critère d’optimisation**

Dans les sous-sections suivantes, les résultats obtenus avec divers critères d’optimisation sont présentés, ainsi que les gains obtenus par la connaissance du type de ressources. Ces critères sont : la maximisation du nombre de bits inversés (NFCB, pour un nombre total d’injections donné), et la diminution de l’erreur d’estimation CBEE.

##### **4.5.1 Optimisation du NFCB**

La maximisation du nombre des bits critiques inversés lors des expérimentations d’injection des pannes permet d’assurer une meilleure évaluation de la sensibilité du design cible et de l’impact des SEU. En utilisant les résultats du tableau 4.1, maximiser le nombre des bits critiques inversés (NFCB) pour un temps d’émulation donné est simple. Il faut simplement prioriser les sous-ensembles les plus sensibles lors de l’injection des pannes. Par exemple, pour un temps d’émulation permettant 87418 pannes (pour être en concordance avec les résultats du Random #1 et #2), une injection exhaustive des pannes dans les 10,993 bits de (!LUT, BG, 1) et dans les 47,945 bits de (!LUT, BG, 0), suivie par une injection aléatoire dans le sous-ensemble (LUT, 0) pour les 28,480 bits restant peut être effectuée.

Les résultats obtenus par la stratégie proposée sont donnés par le tableau 4.2 sous le nom du “Proposed #1, est.”. Avec cette stratégie, à peu près 45 % des pannes ciblent des bits critiques, ce qui mène à une valeur de NFCB estimée de 39057. Obtenir ce nombre de bits critiques inversés demanderait l’injection aléatoire de 9.1 M pannes dans le mode accumulation (réduction de la durée de l’expérimentation par un facteur de 104) et 9.4 M de pannes dans le mode SEU (réduction par un facteur de 108).

#### **4.5.2 Amélioration de la valeur de CBEE (en comparaison avec l’injection aléatoire)**

L’injection des pannes peut être appliquée afin d’estimer le nombre de bits critiques inversés (NFCB) ainsi que l’erreur relative d’estimation de ce nombre (CBEE). Malheureusement, la stratégie présentée précédemment dans la sous-section 4.5.1 et nommée “Proposed #1, est.” dans le tableau 4.2 ne permet pas d’estimer directement CBEE vu qu’il n’y a pas d’échantillons pour les sous-ensembles (!LUT, !BG, 0) et (!LUT, !BG, 1). En effet, ces deux sous-ensembles n’étaient pas considérés par la procédure d’injection de pannes présentée par “Proposed #1, est.”, sachant qu’ils contiennent la majorité des bits de configuration. Afin d’extraire la valeur estimée de CBEE en préservant une valeur élevée de NFCB, l’injection aléatoire dans le sous-ensemble (LUT,0) peut être remplacée par une injection exhaustive des pannes dans les 5613 bits de (!LUT, !BG, 1) suivie par une injection aléatoire dans les 10.7 M de bits (!LUT, !BG, 0) avec les 22867 pannes restantes. Cette approche mène aux résultats donnés par “Proposed #2, est.” dans le tableau 4.2 avec des valeurs estimées de NFCB et CBEE de 38166 et  $\pm 1.2$  %, respectivement, surpassant ainsi l’injection des pannes aléatoires (avec des valeurs de NFCB et CBEE de 376 et  $\pm 10$  %, respectivement dans le mode accumulation). Ces résultats sont obtenus en se basant sur le fait qu’aucune émulation de pannes n’est requise pour estimer le nombre de bits critiques inversés dans les LUT. En fait, l’estimation du NFCB peut être effectuée par l’extraction (par exemple à partir du fichier XDL pour les FPGA de Xilinx) des fonctions et du nombre des entrées (actives) implémentées dans les LUT. Dans notre cas, cette estimation est facile à déduire vu que chaque RO contient 1799 inverseurs, chacun d’eux implémenté dans un LUT (1 entrée active), en plus d’un multiplexeur à 2 entrées et une porte logique AND à 2 entrées qui sont

implémentés dans un seul LUT (4 entrées actives). Ceci donne un total de 7228 bits critiques (3622 '0' et 3606 '1'). Notez que ces bits sont des bits critiques statiques et ils peuvent ne pas être dynamiquement critiques (ne générant pas d'erreur lors du fonctionnement du design) vu que certaines combinaisons d'entrées des LUT ne sont pas générées par le fonctionnement normal du design.

#### 4.5.3 Optimisation du CBEE

Les résultats précédents montrent que la stratégie proposée permet d'améliorer l'estimation du nombre de bits critiques. Ceci est dû au fait que la combinaison de l'estimation théorique des bits critiques des LUT jumelée à une injection exhaustive des pannes ( $10993+47945+5613 = 64551$ ) dans les bits de configuration mène à une estimation précise du nombre de bits critiques.

La seule source d'erreur pour cette estimation est reliée à l'injection des pannes dans les 10.7 M de bits à 0 non identifiés par *Bitgen*, qui configurent des ressources autres que les LUT (!LUT, !BG, 0). Le CBEE peut être réduit davantage en contrepartie d'une réduction de la valeur de NFCB.

Les résultats dans Tableau 4.2, "Proposed #3, est.", montre qu'une valeur estimée de CBEE de  $\pm 1.1\%$  est obtenue. Afin d'atteindre cette valeur, l'injection des pannes exhaustive dans les 47945 bits à 0 configurant d'autres ressources et identifiés comme critiques par *Bitgen* (!LUT, BG, 0) a été remplacée par l'injection de 35300 pannes dans ces 47945, permettant d'augmenter à 35512 le nombre d'injections dans les 10.7M bits à 0 configurant les autres ressources et non identifiés comme essentiels par *Bitgen* (!LUT, !BG, 0).

#### 4.5.4 Gain obtenu par la connaissance du type des ressources

Finalement, afin de quantifier le gain généré par la considération du type des ressources, une estimation a été faite en tenant compte seulement de la valeur et de l'identification des bits essentiels ou non par *Bitgen*. Le Tableau 4.2 montre les résultats obtenus sous ces conditions. Ces résultats, donnés par "bitgen #1, est.", donnent la valeur la plus basse de CBEE sans

considérer le type des ressources, 3.2 %, versus la valeur de 1.1% obtenue par l'approche proposée optimisant la valeur de CBEE. La valeur du NFCB dans "bitgen #1, est." est, quant à elle, inférieure d'un facteur de 3.4 quand elle est comparée aux résultats donnés par "Proposed #3, est.". Ce facteur atteint une valeur de 4.2 quand les résultats sont comparés aux résultats optimisant NFCB ("Proposed #1, est."). Ces résultats montrent la pertinence de considérer le type de ressources dans l'effort d'optimisation de l'injection de pannes.

#### 4.6 Conclusion

Ce chapitre a présenté une approche efficace pour l'optimisation de l'injection des pannes émulant les SEU dans les FPGA de Xilinx à base de mémoire SRAM. En se basant sur l'identification des bits de configuration critiques et priorisant l'injection des pannes dans ces bits, l'approche proposée permet soit de maximiser le nombre de bits critiques inversés pour un nombre bien déterminé des *bit flips* ou de minimiser l'erreur d'estimation du nombre de bits critiques. Les résultats de l'approche proposée ont été comparés aux résultats d'une injection aléatoire de pannes. L'approche proposée, maximisant le nombre des bits critiques inversés, permet d'accélérer la procédure d'injection de pannes d'un facteur atteignant la valeur de 108 quand comparée à l'approche aléatoire. Elle permet aussi de réduire l'erreur d'estimation du nombre des bits critiques à une valeur de  $\pm 1.1$  % calculée pour un intervalle de confiance de 95 % alors que la valeur d'erreur d'estimation des bits critiques générée par l'approche aléatoire d'injection des pannes pour le même intervalle de confiance est évaluée à  $\pm 8.6$  %. Finalement, la pertinence de considérer le type des ressources est clairement dévoilée ainsi que l'efficacité relative de *Bitgen* à identifier les bits critiques. En effet, la considération du type des ressources dans la procédure d'injection des pannes permet d'inverser 3.4 fois plus de bits critiques qu'une procédure ne le considérant pas. De plus, la valeur d'erreur d'estimation des bits critiques diminue de  $\pm 3.2$  % à  $\pm 1.1$  %. En résumé, la priorisation de l'injection des pannes dans les ensembles des bits de configuration les plus sensibles constitue l'aspect original de la méthode. De plus, l'amélioration des performances obtenue permet de juger de sa significativité.

Dans le chapitre suivant, une méthodologie automatisée assurant une évaluation complète des bits de configuration configurant les LUT utilisés par un design est présentée.

**Clicours.COM**



## CHAPITRE 5

### MÉTHODOLOGIE D'INJECTION DES PANNES AUTOMATISÉE POUR L'ÉVALUATION DE LA ROBUSTESSE DES LUT À L'ÉGARD DES SEU DANS LES FPGA À BASE DE SRAM

#### 5.1 Introduction

La procédure d'émulation des effets des SEU dans les FPGA à base de mémoire SRAM et son optimisation étaient l'objet des deux chapitres précédents. En effet, deux approches permettant d'améliorer l'évaluation de la robustesse des designs implémentés dans les FPGA à base de mémoire SRAM ont été présentées. La première consistait à tenir compte de la différence de sensibilité relative dans la procédure d'injection des pannes afin de produire des résultats plus fiables et réalistes, comparables à ceux obtenus par les tests accélérés. La deuxième exploitait la différence de sensibilité relative des bits de configuration en se basant sur leur contenu ainsi que les ressources qu'ils configurent, afin de maximiser le nombre de bits critiques détectés pour un nombre bien déterminé de SEU injectés et afin de minimiser l'erreur lors de l'estimation du nombre de bits critiques.

Le travail décrit par le présent chapitre vise le même objectif que celui du chapitre précédent, soit d'optimiser la procédure d'injection des pannes, sauf qu'on cible ici une ressource bien déterminée du FPGA, à savoir les LUT. En effet, une nouvelle approche ciblant les bits de configuration des LUT des FPGA de Xilinx à base de mémoire SRAM est présentée. Cette approche permet d'identifier tous les bits de configuration utilisés par les LUT pour un design spécifique afin d'y injecter des SEU et des MBU.

Le chapitre 5 est organisé comme suit. La mise en contexte du travail proposé est donnée par la section 5.2. La section 5.3 détaille la méthodologie adoptée pour identifier les bits configurant les LUT. Cette méthodologie basée sur la rétro-ingénierie permet d'identifier tous les LUT du FPGA cible et d'identifier les bits de configuration des LUT seulement utilisés par un design. La section 5.4 présente une approche d'automatisation de la procédure

d'injection des pannes. Les résultats de validation de l'approche proposée sont divulgués par la section 5.5 et finalement, la conclusion de ce chapitre est donnée par la section 5.6.

## 5.2 Mise en contexte

Les SEU et les MBU ont des conséquences critiques sur les FPGA à base de mémoire SRAM. Ils ont aussi la capacité d'interrompre ou de changer le fonctionnement d'un design implémenté dans un FPGA. Prédire le comportement d'un circuit affecté par les SEU devient une tâche incontournable pour les concepteurs afin d'éviter d'envoyer des circuits non suffisamment robustes dans leur milieu de fonctionnement.

La mitigation et la protection des designs contre les SEU deviennent une préoccupation prioritaire pour les concepteurs. Plusieurs techniques existent dans la littérature et peuvent être utilisées tel que le *Triple Modular Redundancy* (TMR) et la duplication. Afin d'être en mesure d'appliquer ces techniques, le concepteur doit être en mesure de faire des compromis entre le coût (surface, alimentation, performance...) et la robustesse du design. La protection sélective est une solution permettant au concepteur de gérer ces compromis. La mitigation sélective demande, tout d'abord, l'identification de la sensibilité des différents bits de configuration. Ceci est assuré par la classification des bits selon leur impact sur le système au cas où leur contenu est modifié par les SEU. Auparavant, l'outil Jbits de Xilinx (Singh et James-Roxby, 2001) permettait l'identification des bits de configuration des Virtex II et des puces les plus anciennes. Pour les composantes les plus récentes de Xilinx, cette tâche devient plus difficile vu qu'aucun outil est disponible. En conséquent, la rétro-ingénierie représente une solution permettant de surmonter ce problème.

L'objectif de l'approche proposée dans ce chapitre est l'identification, via la rétro-ingénierie, de tous les bits de configuration des LUT vulnérables dans un design implémenté dans un FPGA de Xilinx, y compris les familles de circuits plus récentes. Ceci devrait permettre aux concepteurs de prendre les mesures appropriées afin de protéger seulement les parties en question et de minimiser les coûts de conception en termes de surface. La valeur pratique de l'approche proposée peut être décrite comme suit : cette approche permet une couverture des

pannes de 100% vu que tous les bits de configuration des LUT utilisés par un design spécifique sont identifiés. De plus, elle supporte l'injection des SEU ainsi que des MBU vu que l'outil d'injection des pannes adopté possède ces deux fonctionnalités. Elle permet aussi, non seulement l'identification des bits critiques mais aussi l'évaluation de leur impact sur le comportement du design une fois inversés. L'approche proposée devrait permettre de décider de l'efficacité de quelques mesures de mitigation, comme le TMR partiel par exemple où cibler seulement les bits de configuration des LUT dans la zone répliquée, permettant ainsi d'évaluer l'impact de la technique de mitigation ajoutée et de juger si elle est suffisante ou pas.

L'approche proposée dans ce chapitre, qui ne requiert aucun autre outil commercial, s'articule autour d'un processus en deux temps. Elle identifie tout d'abord les bits de configuration des LUT utilisés par un design spécifique. Cette méthode d'identification fait l'objet de la section 5.3. Ensuite, une injection des pannes est effectuée automatiquement, via un script Python, dans ces endroits spécifiques en se basant sur la reconfiguration partielle assurée par les outils de Xilinx : *SEU Controller* pour les FPGA Virtex 5 et SEM pour les versions les plus récentes. Cette procédure automatisée d'injection est, quant à elle, décrite à la section 5.4.

### **5.3 Méthodes d'identification des bits de configuration des LUT**

Dans cette section, deux méthodes d'identification sont présentées : la première ciblant l'ensemble de bits de configuration des LUT, et une seconde ne ciblant qu'un design spécifique (en partie ou en totalité). Dans les deux cas, l'identification des bits de configuration est faite en comparant les fichiers EBC d'un design spécifique, avant et après la modification du fichier XDL, en inversant les fonctions logiques implémentés dans les LUT.

#### **5.3.1 Identification de la totalité des bits de configuration des bits de LUT**

La méthode proposée pour identifier la totalité des bits de configuration des bits de LUT consiste à créer deux versions d'un design artificiel qui contient tous les LUT du FPGA, dont

chaque LUT d'une version est le complément de l'autre. Notons que cette méthode d'identification des bits de configuration a été utilisée pour identifier les bits des LUT (et par conséquent ceux configurant les autres ressources) au chapitre précédent.

Plus précisément, afin de déterminer toutes les adresses des bits de configuration des LUT, les étapes suivantes sont suivies :

- 1) créer un design contenant tous les LUT du FPGA en les connectant tous tel que montré par la figure 5.1;

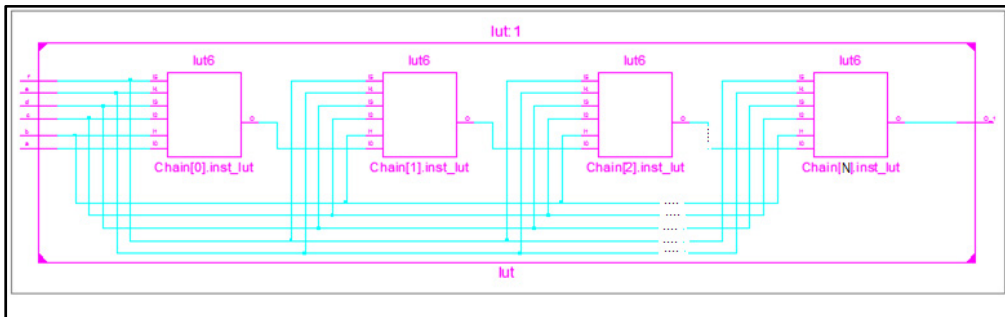


Figure 5.1 Design concaténant tous les LUT du FPGA Virtex-5

- 2) instancier tous les LUT du FPGA à une valeur spécifique (par exemple X "A5A5A5A5A5A5A5A5");
- 3) générer le fichier EBC;
- 4) ré-instancier les LUT à une valeur complémentaire (par exemple X "5A5A5A5A5A5A5A5A");
- 5) générer le second fichier EBC;
- 6) comparer les deux fichiers EBC en extrayant les adresses des bits différents.

En appliquant les étapes précédentes, il a été observé que la synthèse d'un design concaténant tous les LUT pouvait générer un problème de mémoire et que les fichiers EBC pouvaient ne pas être générés. Ce fut le cas lors de nos essais pour générer les adresses des LUT pour le FPGA cible Virtex-5VLX50T. Pour résoudre ce problème, les adresses des LUT du FPGA ont été générées en deux étapes, en appliquant les étapes de 1 à 6 décrites ci-dessus sur les 14 400 LUT de la moitié basse du FPGA en premier lieu et par la suite en les

appliquant sur la moitié haute du FPGA pour les autres 14 400 LUT restants, sachant que le Virtex-5VLX50T contient 28 800 LUT. La figure 5.2 montre le design concaténant les LUT dans les deux compartiments du FPGA Virtex-5.

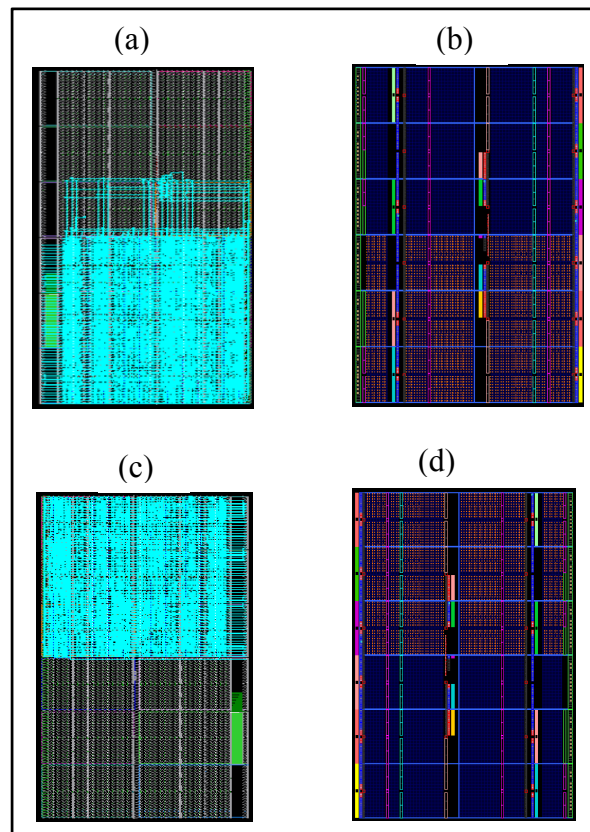


Figure 5.2 Design concaténant les 14 400 LUT du Virtex-5 VLX50T (a) routage dans la moitié basse (b) placement dans la moitié basse (c) routage dans la moitié haute (d) placement dans la moitié haute

Cette première méthode donne toutes les adresses des bits des LUT dans un FPGA. Cette méthode répondait aux besoins du chapitre précédent, étant donné la nature du design ciblé (RO dans les CLB) et du fait que les optimisations qui y sont proposées soit évitent totalement l'injection de pannes dans les LUT (en faisant levier sur la connaissance a priori du nombre de bits critiques dans les LUT) ou dans le pire des cas, procèdent à une injection partielle. Cependant, dans le cas de designs plus conventionnels, il peut s'avérer intéressant d'identifier les LUT vraiment utilisées par le design tout entier ou pour certaines parties

spécifiques. Ce genre d'identification fait l'objet de la seconde méthode, qui est présentée dans ce qui suit.

### 5.3.2 Identification des bits de configuration des bits de LUT utilisés

Afin d'optimiser la procédure d'injection des pannes pour des designs plus conventionnels, les bits des LUT non utilisés ne doivent pas être considérés durant l'injection, vu que des pannes injectées dans ces bits n'auront aucun impact. La seconde méthode d'identification, illustrée à la figure 5.3, vise donc à fournir l'adresse des bits configurant les LUT utilisés par un design spécifique et utilise le même principe que celui utilisé à la section 5.3.1, soit de créer une deuxième version du design avec des contenus de LUT complétés.

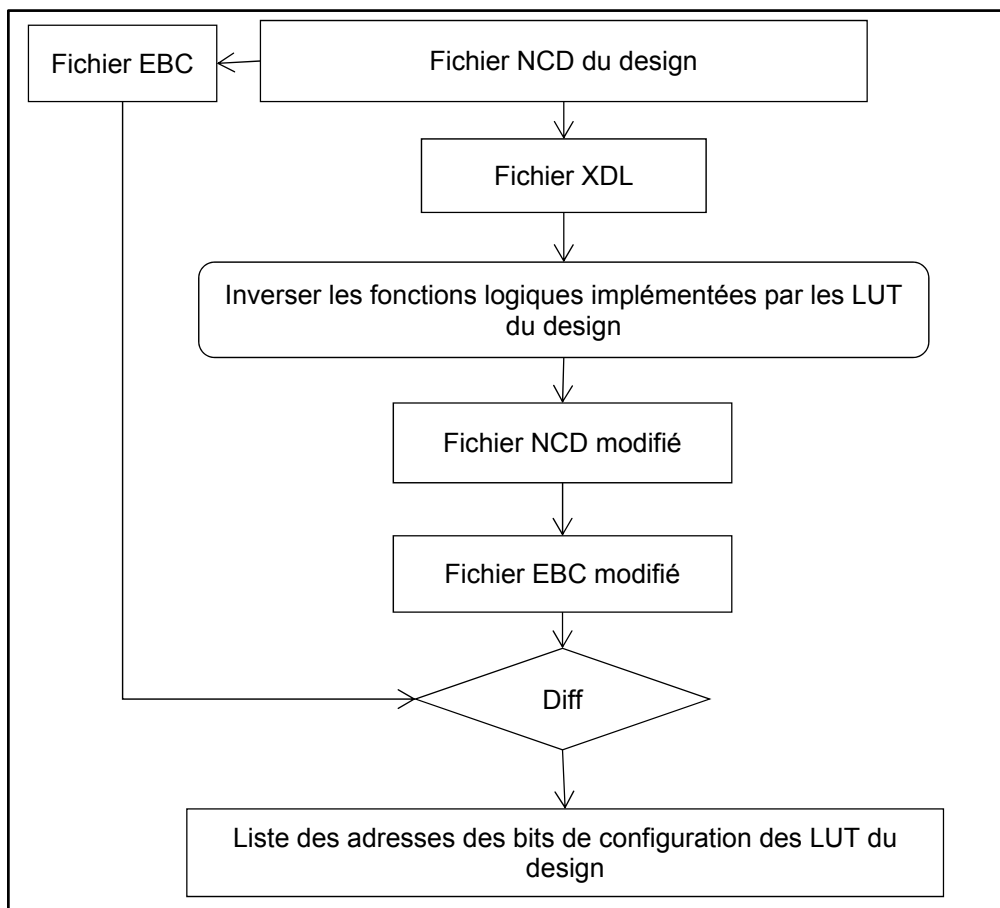


Figure 5.3 Procédure proposée pour l'extraction des bits de configuration des LUT utilisés par un design

Tout d'abord, le fichier NCD (*Native Circuit Description*) du design ainsi que son fichier EBC sont générés. Deuxièmement, le fichier XDL (*Xilinx Design langage*) de ce design est extrait par l'outil *Bitgen* de Xilinx. Les LUT du design sont par la suite identifiés et leurs fonctions logiques sont automatiquement lues et complétées par un script en Python. Troisièmement, une version modifiée des fichiers NCD et EBC est produite. Finalement, les adresses des bits de configuration des LUT utilisés par un design sont extraites à travers une comparaison différentielle entre les deux fichiers EBC.

Afin d'assurer une extraction correcte des adresses des bits de configuration, le nombre de bits par trame ainsi que le nombre total des trames doivent être correctement extraits pour chaque type de FPGA. En effet, le calcul des adresses des bits de configuration en utilisant le fichier EBC est basé sur la détermination correcte de la position du bit dans la trame ainsi que le numéro de la trame. Vu que le nombre de bits par trame ainsi que le nombre des trames varient d'un FPGA à un autre, une attention doit être prêtée à ces deux paramètres en calculant les adresses des bits.

À titre d'exemple, le fichier EBC du Virtex-5VLX50T contient 11 006 368 bits de configuration (en excluant les bits des BRAM), chaque trame étant composée de 41 mots de 32 bits chacun. Alors, au total, 8389 trames sont comptées, où chaque trame est composée de 1312 bits. Dans le cas de l'Artix-7A200T, le nombre total des bits de configuration est 61 104 192 avec 18906 trames de 3232 bits chacune. La modification du fichier XDL ainsi que le calcul des adresses des bits de configuration est fait automatiquement grâce à des scripts en Python et en Matlab, respectivement. L'identification des adresses des bits de configuration d'un seul LUT ou d'un ensemble des LUT utilisés pour une structure spécifique comme le TMR est aussi faisable. L'approche proposée est applicable sur n'importe quel FPGA de Xilinx où la génération du fichier EBC est possible.

Une attention spéciale doit être aussi prêtée à l'architecture du FPGA cible lors de la génération des fichiers UCF pour le design concaténant les LUT. En effet, le nombre des colonnes et des lignes diffère d'une famille de FPGA à une autre et par la suite les adresses des *slices* où les LUT sont implémentés peuvent varier considérablement d'un FPGA à un

autre. La figure 5.4 montre la différence des adresses des slices pour deux FPGA de différentes familles.

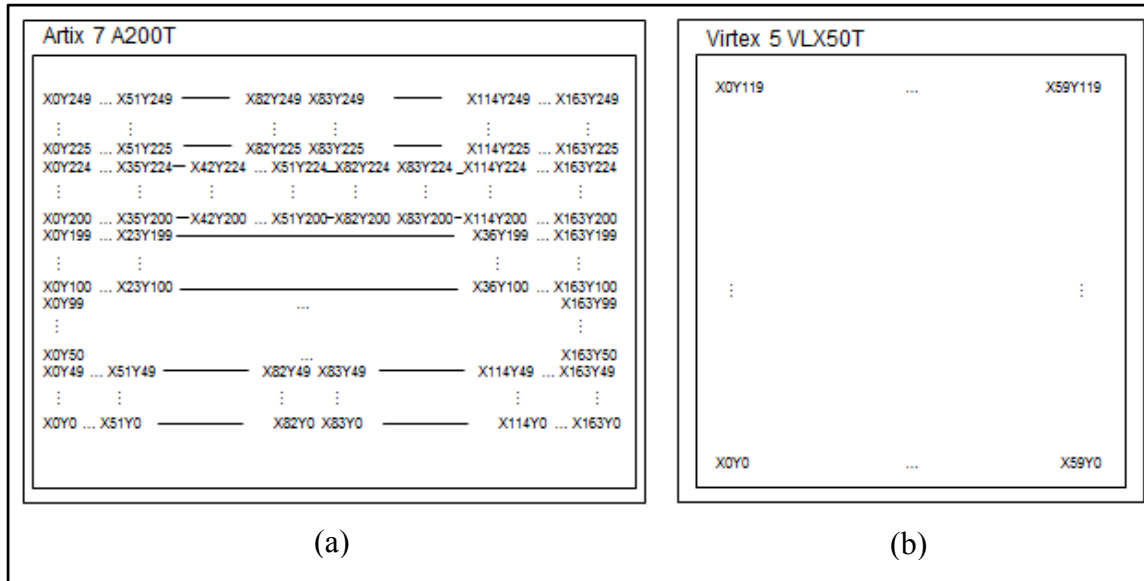


Figure 5.4 Les adresses des slices du (a) Artix-7 A200T (b) Virtex-5 VLX50T

#### 5.4 Procédure automatisée d'injection exhaustive des pannes ciblant les LUT utilisés

Contrairement au chapitre précédent (où l'injection dans les LUT était soit évitée ou soit partielle), ici l'évaluation complète de la sensibilité des LUT demande que l'injection des SEU soit effectuée d'une façon exhaustive : tous les bits configurant les LUT sont inversés, un par un, de sorte que tous les bits critiques des LUT et par conséquent tous les LUT vulnérables soient identifiés. Ce qui permet de qualifier l'approche proposée de "déterministe" permettant de tester complètement tous les LUT dans un design spécifique.

La procédure d'injection utilisée pour cibler les LUT utilisés s'appuie sur celle présentée au chapitre 3. Une des différences notoires est qu'il n'est pas nécessaire ici de protéger le *SEU Controller* car il l'est de facto. Comme c'est le cas pour les autres procédures d'injection des pannes présentées dans cette thèse, celle ciblant spécifiquement les LUT est contrôlée par l'utilisateur via l'Hyperterminal. La procédure automatisée d'injection des pannes ciblant les LUT est illustrée à la figure 5.5. Un script en python est responsable de cette tâche. Il



contrôle l'outil d'injection des pannes en envoyant des commandes via l'Hyperterminal. La procédure d'automatisation est détaillée par les quatre étapes suivantes :

- 1) les adresses des bits configurant les LUT sont lues une par une après leurs extraction par l'étape d'identification;
- 2) le mode UART du *SEU Controller* est déployé en envoyant le caractère '\*'. Le *SEU Controller* est mis au mode *Detect Only Mode* (DOM);
- 3) le caractère 't' est envoyé avec l'adresse du frame et du bit afin d'y injecter une panne;
- 4) les sorties du DUT sont lues afin d'observer l'effet de la panne et par la suite le mode *Auto Correct Mode* (ACM) est activé.

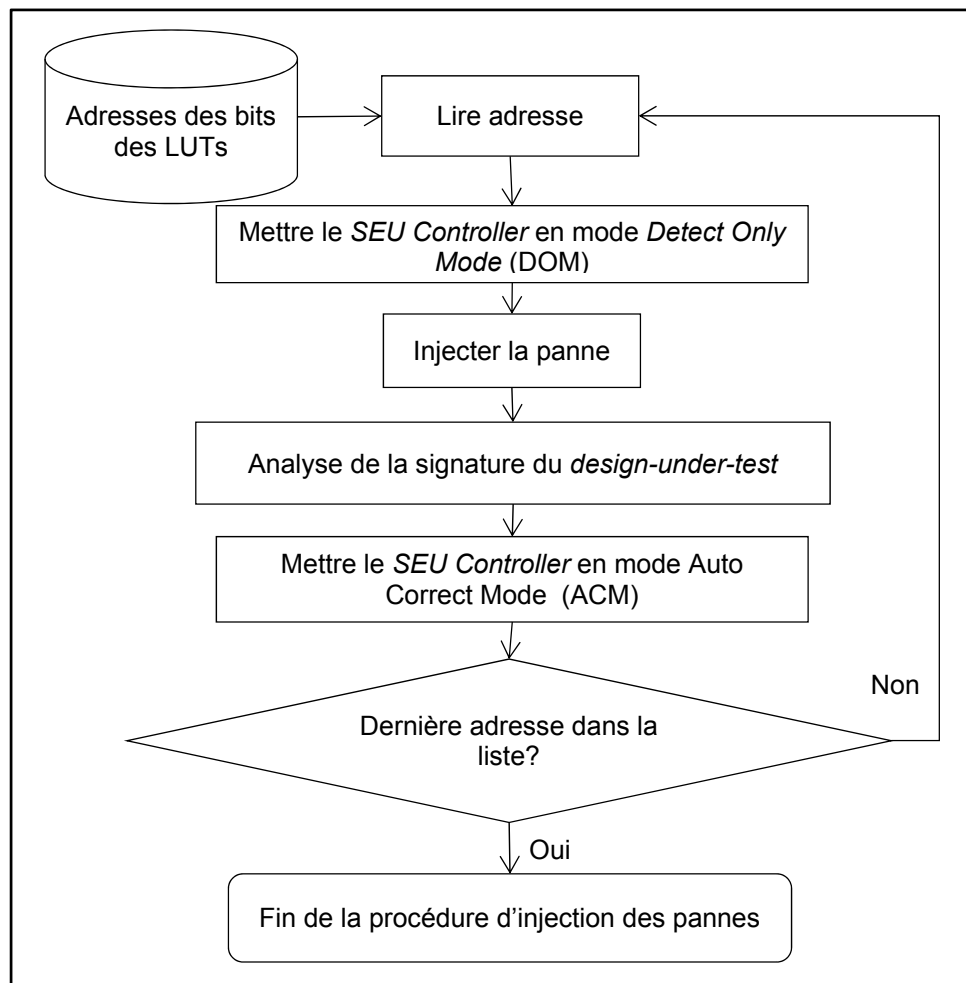


Figure 5.5 Procédure automatisée d'injection des pannes ciblant les LUT

Cette procédure est répétée jusqu'à ce qu'il ne reste aucune adresse dans la liste. Dans le cas d'injection des MBU, après l'envoi du caractère '\*', le caractère '2' doit être envoyé. Le *SEU Controller* ne peut pas corriger un MBU, alors après chaque injection de MBU, une reconfiguration du FPGA est nécessaire. Il est important de noter que l'extraction de la signature du DUT n'est pas automatique vu qu'elle dépend du design cible.

## 5.5 Résultats de validation

L'approche proposée a été validée en l'appliquant sur le design composé de deux RO implémentés dans les CLB. Rappelons que chaque RO est composé de 1799 inverseurs où chaque inverseur occupe un LUT. En plus, chaque RO a un multiplexeur permettant de varier la longueur des RO en réduisant le nombre des inverseurs utilisés à 1797, illustré à la figure 5.6.

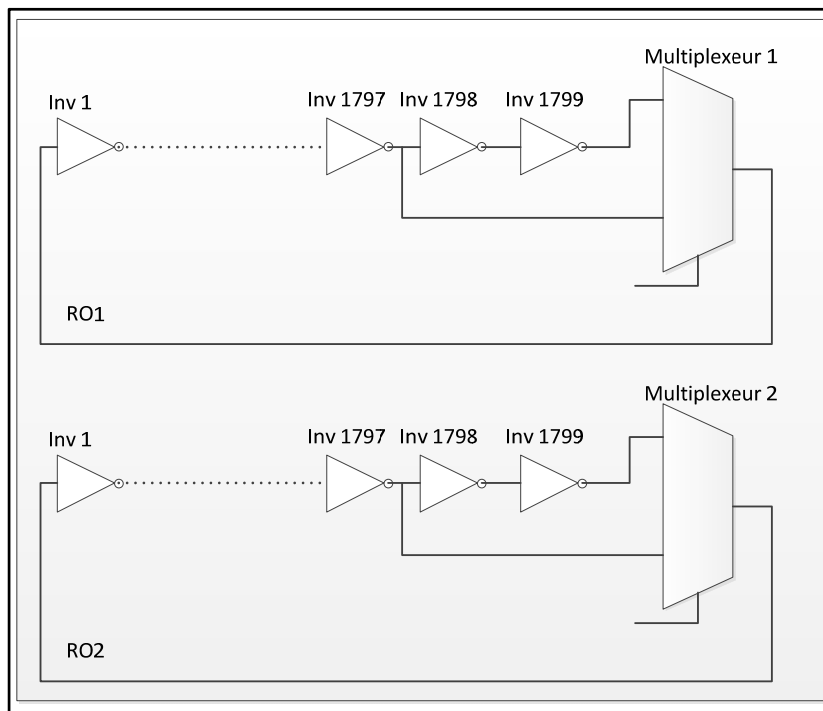


Figure 5.6 Illustration des multiplexeurs dans les RO

Les deux multiplexeurs sont implémentés chacun dans un LUT séparé. Le nombre total des LUT utilisés pour ce design est 3600. L'estimation du nombre de bits où un SEU peut causer

une erreur est simple ce qui rend les RO un excellent choix pour montrer l'efficacité de l'approche proposée.

En extrayant le nombre des entrées utilisés pour implémenter chaque composant à partir du fichier XDL, le nombre de bits critiques peut être estimé. Dans notre cas, chaque inverseur est implémenté avec deux bits de configuration de LUT. Ce qui donne un nombre de bits critique total pour notre design (en excluant les multiplexeurs) égal à 7196 (2 bits x 2 RO x 1799 inverseurs). En appliquant l'approche proposée, le nombre total des bits configurant les LUT extraits du design (ciblant les inverseurs seulement) est 230272 (64 bits pour chaque LUT utilisé). En injectant des pannes dans les bits extraits, on trouve exactement 7196 adresses dans lesquelles une panne cause une erreur. L'égalité entre les résultats d'estimation et les résultats d'injection des pannes prouve que l'approche proposée est efficace et offre une couverture des pannes de 100%.

Afin de montrer que l'approche proposée peut être aussi appliquée partiellement à un design en choisissant sélectivement des parties implémentées dans les LUT du design, nous avons fait une deuxième expérimentation ciblant seulement les multiplexeurs dans le design des RO. À partir des 128 adresses des bits configurant les 2 LUT où les multiplexeurs sont implémentés, 8 bits critiques sont détectés: 4 causent des ROB et 4 causent des ODC. Les ODC sont causés vu que les bits configurant les entrées de sélection des multiplexeurs qui sont responsables de la variation des longueurs des RO sont inversés. La deuxième expérimentation montre que l'approche proposée peut être appliquée partiellement (ou totalement) pour tester les LUT dans des structures spécifiques de mitigation telles que les parties du design où le TMR sélective est utilisé.

## **5.6 Conclusion**

Une approche automatisée d'injection des pannes évaluant la robustesse des LUT utilisés par des designs implémentés dans les FPGA à base de mémoire SRAM est proposée. L'approche permet d'extraire toutes les adresses des bits configurant les LUT utilisés par un design. Ensuite, l'injection des pannes dans les adresses extraites est effectuée automatiquement.

Cette approche permet une couverture complète des pannes où tous les LUT du design sont testés. Elle permet aussi d'évaluer la robustesse des LUT d'une structure de mitigation spécifique vu qu'elle permet d'extraire les adresses des bits des LUT d'une partie sélectionnée du FPGA. L'approche proposée est efficace vu qu'elle assure une couverture de pannes de 100 %. Elle permet d'identifier tous les bits utilisés par un design et de procéder à une injection exhaustive de pannes. De plus, en comparaison avec les autres approches proposées dans la littérature et présentées dans le chapitre 1 section 1.3.3, l'approche proposée est simple à implémenter et à appliquer et elle est applicable aux nouvelles générations des FPGA. En effet, les approches dans la littérature, afin d'identifier les bits des LUT, avaient recours à des outils tel que JBits qui est obsolète ou tel que LUTXtract qui agit sur le *netlist* d'un design pour déterminer les bits des LUT et sans lequel il est impossible de les extraire. L'approche que nous proposons agit directement sur des fichiers générés par l'outil de synthèse du design lisibles par l'utilisateur, notamment les fichiers XDL et EBC. Ceci dit que l'extraction des bits des LUT selon notre méthode ne requiert qu'une simple manipulation de fichiers texte avec n'importe quel outil permettant de le faire. L'approche proposée peut assurer un test en ligne si un module contrôlant l'injection des pannes ainsi qu'une mémoire contenant les adresses des bits à cibler peuvent être embarqués dans le système à tester dans son environnement de fonctionnement final.

## CONCLUSION

L'objectif ultime de cette thèse était de mettre en œuvre une stratégie de pré-certification permettant d'évaluer d'une manière sophistiquée la robustesse des designs implémentés dans les FPGA à base de mémoire SRAM avant de les envoyer à la phase très coûteuse de certification. La technique d'injection des pannes par émulation fut la technique adoptée pour établir la stratégie ciblée. Plusieurs outils d'émulation ont été développés dans la littérature ce qui nous a poussé à choisir un injecteur des pannes déjà existant, répondant aux besoins dictés par les objectifs de la thèse, et à diriger les efforts déployés vers la recherche des méthodologies permettant d'optimiser la procédure de pré-certification pour la rendre plus efficace et plus réaliste par rapport aux tests de certification.

La conquête de la stratégie de pré-certification a mené les travaux de recherche à trois contributions principales :

- 1) La considération de l'effet du contenu des bits de configuration sur la variation de la sensibilité de ces derniers dans la procédure d'injection des pannes par émulation afin de produire des résultats les plus proches possibles de ceux obtenus par les tests accélérés. Cette approche a aussi permis d'étudier et de confirmer l'effet du contenu des bits de configuration sur la variation de la sensibilité de ces derniers;
- 2) Une nouvelle méthodologie d'optimisation de la procédure d'injection des pannes par émulation permettant
  - la maximisation de nombre de bits critiques inversés pour un nombre de pannes bien déterminé, réduisant ainsi la durée des expérimentations;
  - l'augmentation de la précision d'estimation du nombre de bits critiques pour un design spécifique;
- 3) Une nouvelle approche facilitant l'évaluation de la sensibilité des bits des LUT utilisés par un design cible.

L'idée derrière la première contribution, qui est majeure, est venue de l'analyse des résultats des tests sous radiation à TRIUMF. En effet, une comparaison entre les fichiers de configuration relus à partir du FPGA sous test lors des bombardements par faisceaux de

protons avec le fichier de configuration de référence a montré que les bits de configuration à '1' sont pratiquement deux fois plus sensibles que les bits à '0'. Cette observation a été confirmée par d'autres travaux trouvés dans la littérature mentionnant la différence de sensibilité des bits de configuration selon leur contenu. L'étape suivante consistait à trouver une façon pour mettre en œuvre cette différence de sensibilité à travers l'émulation. Pour ce faire, des expérimentations d'injection des pannes ciblant juste des bits '1' et juste des bits '0' ont été effectuées. Les résultats obtenus ont bien illustré que le contenu des bits de configuration a un impact important sur le type des événements observés, leur nombre et le nombre de pannes nécessaires pour générer des événements. À partir de ces résultats, un ratio permettant de considérer la différence de sensibilité relative a été mathématiquement défini afin de comparer notre approche, les résultats de TRIUMF et l'approche conventionnelle d'injection aléatoire des pannes. La comparaison des résultats estimés a montré la pertinence de considérer la différence de sensibilité relative dans la procédure d'injection des pannes, ce qui justifie la proposition d'une procédure de génération des séquences de test tenant en compte cet aspect. La validation de l'approche proposée a été faite en comparant les résultats obtenus des différentes approches et la considération de la sensibilité relative dans la procédure d'émulation a donné des résultats proches de ceux obtenus des tests accélérés. Les résultats obtenus à partir d'une injection de pannes considérant la sensibilité relative, en plus qu'ils soient validés statistiquement, sont validés expérimentalement. En effet, en comparaison avec les résultats de TRIUMF, une erreur relative de 3.1 % a été enregistrée. Tandis qu'une comparaison des résultats de TRIUMF avec ceux d'une injection conventionnelle aléatoire de pannes donne une erreur atteignant 75 %. La considération de la sensibilité relative dans la procédure d'injection de pannes permet aussi d'éviter la sous-estimation de la sensibilité d'un design. En effet, une injection de pannes considérant la sensibilité relative sur le circuit B08 des benchmarks ITC99 génère 2.3 fois plus d'erreurs en comparaison d'une injection aléatoire de pannes. À notre connaissance, nous sommes les premiers à proposer une procédure d'émulation avec un tel réalisme en considérant la différence de sensibilité relative. Cette contribution a fait l'objet d'un article de revue publié :

Anis Souari, Claude Thibeault, Yves Blaquière et Raoul Velazco. 2016. « Towards an Efficient SEU Effects Emulation on SRAM-Based FPGAs ». *Microelectronics Reliability*, Elsevier (soumis).

La deuxième contribution consistait à classifier les bits de configuration en différents sous-ensembles selon leur contenu et les ressources qu'ils configurent, étudier leur sensibilité face aux SEU et ensuite prioriser l'injection des pannes dans les sous-ensembles les plus sensibles. Quelques scénarios permettant de maximiser le nombre de bits critiques inversés ou maximiser la précision d'estimation de leurs nombre ont été proposés. Une amélioration de la durée de l'expérimentation de plus de deux ordres de grandeur a été enregistrée en comparaison avec l'approche d'injection aléatoire des pannes. L'objectif visé derrière cette approche consistait à améliorer l'estimation de la sensibilité des designs sous test. En effet, elle permet d'améliorer l'estimation du nombre des bits critiques en réduisant l'erreur à une valeur de  $\pm 1.1\%$  calculée pour un intervalle de confiance de 95 % en comparaison à une valeur de  $\pm 8.6\%$  donnée par une injection aléatoire des pannes. De plus, l'approche proposée montre que la considération du type des ressources dans la procédure d'injection des pannes permet d'augmenter de 3.4 fois le nombre de bits critiques inversés et de diminuer l'erreur d'estimation de leur nombre de  $\pm 3.2\%$  à  $\pm 1.1\%$  en comparaison avec une procédure ne le considérant pas. Cette contribution fait l'objet d'un article présenté à un symposium IEEE :

Anis Souari, Claude Thibeault, Yves Blaquière et Raoul Velazco. 2015. « Optimization of SEU emulation on SRAM FPGAs based on sensitiveness analysis ». In *2015 IEEE 21st International On-Line Testing Symposium (IOLTS)* (publié).

La dernière contribution de cette thèse avait pour objectif d'assurer une évaluation simple, complète et efficace de la sensibilité des bits de configuration des LUT déployés par le DUT. Dans la littérature, la pierre angulaire de ce type d'approches consiste en l'identification des bits cibles des tests. D'habitude, des outils externes requérant un minimum d'expertise sont requis à cette fin, ce qui rend la tâche plus compliquée. De plus, ces outils ne sont plus disponibles pour les nouvelles générations des FPGA. C'est dans ce contexte qu'il fut choisi de proposer une méthode simple d'identification des bits des LUT, sans aucun recours aux

outils externes et applicable pour les nouvelles générations des FPGA de Xilinx où une génération des fichiers XDL et EBC est possible. L'approche proposée assure une couverture de pannes de 100%, ce qui la rend plus performante et moins coûteuse par rapport autres approches existantes. De plus, la simplicité de l'approche proposée provient du fait qu'elle n'a besoin que d'un simple outil permettant la gestion des fichiers texte pour extraire les adresses des LUT. Cette contribution fait l'objet d'un article présenté à un symposium IEEE :

Anis Souari, Claude Thibeault, Yves Blaquière et Raoul Velazco. 2015. « An automated fault injection for evaluation of LUTs robustness in SRAM-based FPGAs ». In 2015 IEEE East-West Design & Test Symposium (EWDTS) (publié).

Toutefois, durant les travaux de cette thèse, nous avons dû avoir recours à la rétro-ingénierie, en manipulant les fichiers générés par les outils de synthèse utilisés. En effet, la détermination des adresses d'un groupe des bits de configuration spécifique n'est pas une information accessible et divulguée par les constructeurs des FPGA cibles. Ceci dit, l'accès à cette information nous a permis non seulement l'établissement des contributions essentielles mais aussi l'implémentation des contributions secondaires. La protection de l'outil d'injection des pannes adopté, *SEU Controller*, de l'autodestruction en est une. Identifier les bits configurant la macro ainsi que ses bits essentiels et éviter d'injecter des pannes dedans a en effet permis de minimiser le risque de son dysfonctionnement.

Cette thèse a permis de valider le concept de pré-certification, tout en fournissant les outils nécessaires liés à ce concept. Le projet lié à cette thèse, vu qu'il représentait un volet à part entière du projet global AVIO403, était d'une grande importance pour atteindre l'objectif global à savoir l'intervention sur le processus conventionnel d'intégration des systèmes embarqués pour tester leur robustesse à l'égard des radiations.



## RECOMMANDATIONS

Les améliorations apportées à la procédure d'injection des pannes par émulation proposées dans cette thèse sont à la fois inspirées par des observations et analyses des résultats de tests accélérés sous faisceaux de protons et validées par ces derniers. À cause de l'accès limité aux accélérateurs des particules et des coûts élevés de ces types de test, seulement deux circuits ont été testés sous radiations à savoir les RO (un implémenté dans les CLB et l'autre dans les IOB) et ont donc pu servir comme des circuits de référence et de validation de nos travaux. Il serait donc intéressant tester sous radiation d'autres designs de différentes natures (combinatoires ou séquentiels), de différentes tailles, occupant différentes ressources du FPGA, afin de donner, entre autres, une meilleure estimation de la différence de sensibilité entre les bits de configuration à '1' et ceux à '0'. Une valeur générique de sensibilité relative pourrait ainsi être déduite et utilisée dans la génération des séquences de test, ce qui pourrait améliorer les résultats d'injection des pannes. Parallèlement à ces tests accélérés de designs plus conventionnels, poursuivre l'investigation de l'impact de la différence de sensibilité sur ce type de designs est également recommandé, afin de valider l'hypothèse de la testabilité réduite des LUT.

Une autre ouverture à ce projet réside dans l'identification des bits de configuration et la génération d'une cartographie liant chaque bit de configuration aux ressources qu'il configure. L'accès à une telle information n'est pas actuellement divulgué par les fabricants des FPGA, alors qu'auparavant, il existait des outils fournis par ces mêmes fabricants permettant d'accéder à ce genre d'informations.

La cartographie des bits de configuration pourrait contribuer à l'amélioration des procédures d'évaluation de la sensibilité des FPGA et leur protection contre les effets des radiations. Tout d'abord, ceci permettrait de mieux caractériser les différentes ressources du FPGA selon leurs vulnérabilités face aux radiations, ce qui ajouterait de la précision lors de l'estimation de la différence de sensibilité entre les bits de configuration selon ressources qu'ils configurent. Tenant en compte cette information dans la procédure d'injection de pannes par émulation considérant la différence de sensibilité des bits à '1' et ceux à '0'

ajouterait plus de réalisme et une meilleure stratégie de pré-certification serait envisageable. En outre, la caractérisation des bits de configuration selon leurs ressources permettra de mieux identifier les zones les plus sensibles et par la suite d'améliorer les techniques de mitigation afin de les rendre plus efficaces.

Finalement, la stratégie de pré-certification présentée dans cette thèse n'est que l'une des dernières étapes de vérification dans le flot de conception défini par le projet CRIAQ AVIO403, donc il serait intéressant de valider via notre stratégie les approches développées par les autres intervenants dans le projet pour des niveaux d'abstraction plus élevés.



Tel que montré par la figure figure-A I-2, la structure du fichier EBC est la même que celle du fichier EBD sauf que le premier contient les bits de configuration alors que le deuxième contient leur masque où les bits ayant la valeur 1 sont potentiellement critiques et les autres non.

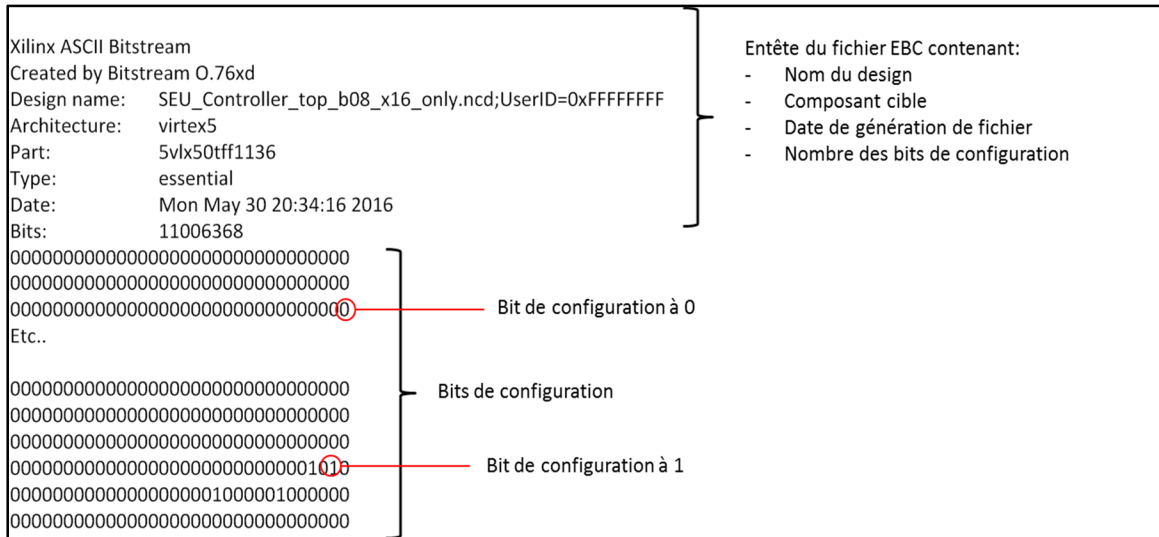


Figure-A I-2 Structure générale du fichier EBC





## ANNEXE II

### INTERFACE D'INJECTION DES PANNES

La figure figure-A II-1 décrit les différentes fonctionnalités offertes par l'interface *Injector* utilisée pour faciliter la communication avec *SEU Controller* et par la suite faciliter la procédure d'injection des pannes.

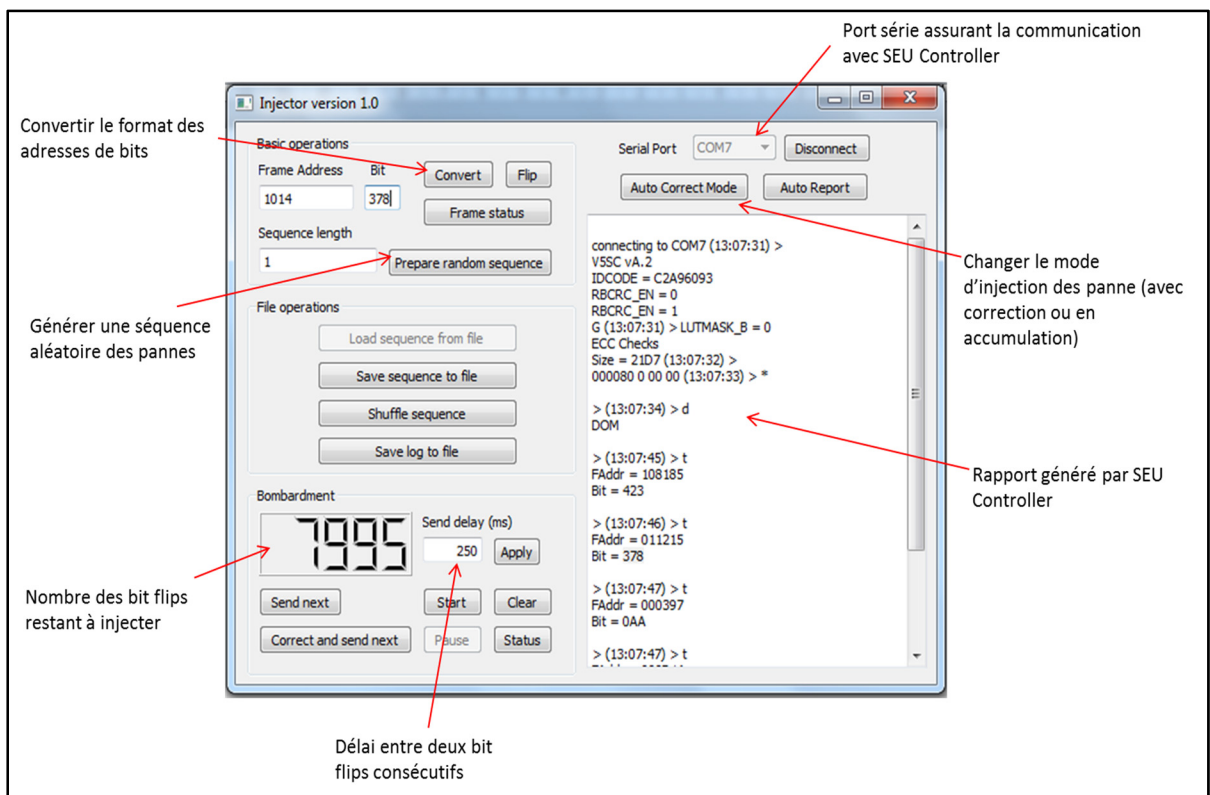


Figure-A II-1 Interface d'injection des pannes





### ANNEXE III

#### FICHIERS GÉNÉRÉS PAR L'INTERFACE LABVIEW

La figure figure-A III-1 montre la structure du fichier « *Traces.log* » généré par l'interface *labview*. Tel que montré, le fichier contient les amplitudes de toutes les fréquences mesurées par l'analyseur de spectre.

-83.624	-80.412	-80.100	-83.060	-77.852	-69.256	-63.696	-63.116	-65.136	-74.096	-82.272	-81.220	-80.172	-
81.488	-84.660	-82.524	-82.284	-83.716	-84.212	-83.256	-84.136	-90.428	-87.488	-87.420	-88.724	-90.056	-
94.356	-90.128	-85.624	-83.924	-84.100	-87.588	-85.596	-85.440	-84.128	-82.384	-83.084	-86.212	-84.140	-
81.908	-82.112	-85.300	-84.920	-86.332	-85.568	-83.184	-83.132	-85.384	-84.784	-80.704	-80.828	-87.336	-
Etc..													
80.656	-77.648	-77.300	-78.628	-82.720	-81.148	-81.188	-82.600	-85.868	-81.712	-79.988	-81.116	-77.072	-
74.636	-73.964	-73.984	-74.856	-79.452	-86.500	-85.252	-80.752	-80.568	-82.964	-81.268	-79.152	-79.124	-
80.832	-83.572	-81.140	-81.060	-80.468	-75.976	-75.408	-75.816	-78.060	-80.364	-81.408	-83.368	-80.848	-
80.820	-81.624	-82.088	-80.040	-80.316	-85.904	-81.764	-78.532	-76.900	-77.244	-79.700	-80.204	-82.000	-
82.440	-76.400	-75.392	-77.092	-81.592	-82.328	-81.888	-76.268	-75.208	-75.832				

Figure-A III-1 Extrait du fichier "*Traces.log*"

La figure-A III-2 montre un extrait du fichier « *Peak\_values.log* » avec une légende décrivant le contenu de chaque colonne dans le fichier.

04:22:05 PM	95218	-32	TRUE
04:22:07 PM	95218	-32	FALSE
04:22:08 PM	95218	-32	FALSE
04:22:09 PM	95164	-32	FALSE
04:22:11 PM	95164	-32	FALSE
04:22:12 PM	95164	-32	FALSE
04:22:13 PM	95164	-32	FALSE
04:22:15 PM	95218	-32	FALSE
Etc..			
04:22:37 PM	95164	-32	FALSE
04:22:38 PM	95164	-32	FALSE
04:22:40 PM	95164	-32	FALSE
04:22:41 PM	95164	-32	FALSE
04:22:42 PM	95164	-32	FALSE
04:22:43 PM	95164	-32	FALSE
04:22:45 PM	95218	-32	FALSE
04:22:46 PM	95164	-32	FALSE
04:22:47 PM	95218	-32	FALSE





	Temps de lecture des données
	Fréquence du Peak
	Amplitude du Peak
	Valeur booléenne de ODC_flag

Figure-A III-2 Extrait du fichier "*Peak\_values.log*"

## LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

- Alderighi, Monica, Fabio Casini, Sergio D'Angelo, Marcello Mancini, S Pastore, Giacomo R Sechi et Gabriele Sorrenti. 2009a. « Experimental Validation of Fault Injection Analyses with the FLIPPER Tool ». Switzerland: 9th ESA/ESTEC D/TEC-QCA.
- Alderighi, Monica, Fabio Casini, Mauro Citterio, Sergio D'Angelo, Marcello Mancini, Sandro Pastore, Giacomo R Sechi et Gabriele Sorrenti. 2009b. « Using FLIPPER to predict proton irradiation results for VIRTEX 2 devices: a case study ». *IEEE Transactions on Nuclear Science*, vol. 4, n° 56, p. 2103-2110.
- Alderighi, Monica, Fabio Casini, Sergio D'Angelo, Marcello Mancini, David Merodio Codinachs, Sandro Pastore, Christian Poivey, Giacomo R Sechi, Gabriele Sorrenti et Roland Weigand. 2010. « Experimental validation of fault injection analyses by the FLIPPER tool ». *IEEE Transactions on Nuclear Science*, vol. 4, n° 57, p. 2129-2134.
- Alderighi, Monica, Fabio Casini, Sergio D'Angelo, S Pastore, GR Sechi et R Weigand. 2007. « Evaluation of single event upset mitigation schemes for SRAM based FPGAs using the FLIPPER fault injection platform ». In *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2007)*. p. 105-113. IEEE.
- Altera. 2007. « Stratix III FPGAs vs. Xilinx Virtex-5 Devices: Architecture and Performance Comparison ». Altera Corporation.
- Amaricai, Alexandru, Sergiu Nimara, Oana Boncalo, Jiaoyan Chen et Emanuel Popovici. 2014. « Probabilistic gate level fault modeling for near and sub-threshold CMOS circuits ». In *Digital System Design (DSD), 2014 17th Euromicro Conference on*. p. 473-479. IEEE.
- Antoni, Lörinc, Régis Leveugle et Béla Fehér. 2003. « Using run-time reconfiguration for fault injection applications ». *IEEE Transactions on Instrumentation and Measurement*, vol. 52, n° 5, p. 1468-1473.
- Arlat, Jean. 1990. « Validation de la sûreté de fonctionnement par injection de fautes : méthode, mise en oeuvre, application ». Institut national polytechnique de Toulouse, 191 p.
- Asadi, Ghazanfar, et Mehdi B Tahoori. 2005a. « Soft error rate estimation and mitigation for SRAM-based FPGAs ». In *Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays*. p. 149-160.
- Asadi, Ghazanfar, et Mehdi Baradaran Tahoori. 2005b. « An analytical approach for soft error rate estimation in digital circuits ». In *2005 IEEE International Symposium on Circuits and Systems*. p. 2991-2994.

- Bagshaw, Michael. 2008. « Cosmic radiation in commercial aviation ». *Travel medicine and infectious disease*, vol. 6, n° 3, p. 125-127.
- Barth, Thomas. 2015. « Memory Array Architectures ». < <http://www.barth-dev.de/knowledge-corner/digital-design/memory-array-architectures/> >.
- Bernardeschi, Cinzia, Luca Cassano et Andrea Domenici. 2012. « SEU-X: A SEU unexcitability prover for SRAM-FPGAs ». In *2012 IEEE 18th International On-Line Testing Symposium (IOLTS)*. p. 25-30. IEEE.
- Bernardi, Paolo, Matteo Sonza Reorda, Luca Sterpone et Massimo Violante. 2004. « On the evaluation of SEU sensitiveness in SRAM-based FPGAs ». In *On-Line Testing Symposium, 2004. IOLTS 2004. Proceedings. 10th IEEE International*. p. 115-120. IEEE.
- Berrojo, Luis, Isabel González, Fulvio Corno, Matteo Sonza Reorda, Giovanni Squillero, Luis Entrena et Celia Lopez. 2002. « New techniques for speeding-up fault-injection campaigns ». In *Proceedings of the conference on Design, automation and test in Europe*. p. 847. IEEE Computer Society.
- Bocquillon, Alexandre. 2009. « Évaluation de la sensibilité des FGPA SRAM-based face aux erreurs induites par les radiations naturelles ». Institut Polytechnique de Grenoble, 109 p.
- Bolchini, Cristiana, Fabrizio Castro et Antonio Miele. 2009. « A fault analysis and classifier framework for reliability-aware SRAM-based FPGA systems ». In *2009 24th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*. p. 173-181. IEEE.
- Bolchini, Cristiana, Luigi Pomante, Fabio Salice et Donatella Sciuto. 2001. « Reliability properties assessment at system level: a co-design framework ». In *On-Line Testing Workshop, 2001. Proceedings. Seventh International*. p. 165-171. IEEE.
- Bombieri, Nicola, Franco Fummi et Valerio Guarnieri. 2011. « Accelerating RTL fault simulation through RTL-to-TLM abstraction ». In *2011 Sixteenth IEEE European Test Symposium*. p. 117-122. IEEE.
- Boué, Jérôme, Philippe Pétilion et Yves Crouzet. 1998. « MEFISTO-L: a VHDL-based fault injection tool for the experimental assessment of fault tolerance ». In *Fault-Tolerant Computing, 1998. Digest of Papers. Twenty-Eighth Annual International Symposium on*. p. 168-173. IEEE.

- Buchner, S, M Olmos, Ph Cheynet, R Velazco, D McMorrow, J Melinger, R Ecoffet et JD Muller. 1998. « Pulsed laser validation of recovery mechanisms of critical SEE's in an artificial neural network system ». *RADECS-97*, p. 353-359.
- Bushnell, Michael, et Vishwani D Agrawal. 2000. *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*, 17. Springer Science & Business Media.
- Caffrey, Michael, Keith Morgan, Diane Roussel-Dupre, Scott Robinson, Anthony Nelson, Anthony Salazar, Michael Wirthlin, William Howes et Daniel Richins. 2009. « On-orbit flight results from the reconfigurable cibola flight experiment satellite (CFESat) ». In *Field Programmable Custom Computing Machines, 2009. FCCM'09. 17th IEEE Symposium on*. p. 3-10. IEEE.
- Cai, Shuo, Ji-Shun Kuang, Tie-Qiao Liu et Wei-Zheng Wang. 2015. « Soft Error Susceptibility Analysis for Sequential Circuit Elements Based on EPPM ». *Journal Of Semiconductor Technology And Science*, vol. 15, n° 2, p. 169.
- Cha, Hungse, Elizabeth M Rudnick, Janak H Patel, Ravishankar K Iyer et Gwan S Choi. 1996. « A gate-level simulation environment for alpha-particle-induced transient faults ». *IEEE Transactions on Computers*, vol. 45, n° 11, p. 1248-1256.
- Chapman, Ken. 2010a. « New Generation Virtex-5 SEU Controller ». Xilinx.
- Chapman, Ken. 2010b. « Virtex-5 SEU Critical Bit Information Extending the capability of the Virtex-5 SEU Controller ». Xilinx.
- Chapman, Ken, et Les Jones. 2010. « SEU strategies for Virtex-5 devices ». *Xilinx Corporation*, vol. 20.
- Chee, P Alexander, Leslie A Braby et Thomas J Conroy. 2000. « Potential doses to passengers and crew of supersonic transports ». *Health physics*, vol. 79, n° 5, p. 547-552.
- Chen, Mingsong, Prabhat Mishra et Dhrubajyoti Kalita. 2012. « Automatic RTL test generation from SystemC TLM specifications ». *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 11, n° 2, p. 38.
- Cieslewski, Grzegorz, Alan D George et Adam Jacobs. 2010. « Acceleration of FPGA Fault Injection Through Multi-Bit Testing ». In *ERSA*. p. 218-224.
- Civera, Pierluigi, Luca Macchiarulo, Maurizio Rebaudengo, Matteo Sonza Reorda et A Violante. 2001. « Exploiting FPGA for accelerating fault injection experiments ». In *On-Line Testing Workshop, 2001. Proceedings. Seventh International*. p. 9-13.

- Corno, Fulvio, Gianluca Cumani, M Sonza Reorda et Giovanni Squillero. 2000. « RT-level fault simulation techniques based on simulation command scripts ». In *Proc. XV Conf. on Design of Circuits and Integrated Systems (DCIS'00), Montpellier, France*. p. 825-830.
- Darvishi, Mostafa, Yves Audet, Yves Blaqui re, Claude Thibeault, Simon Pichette et Fatima Zahra Tazi. 2014. « Circuit Level Modeling of Extra Combinational Delays in SRAM-Based FPGAs Due to Transient Ionizing Radiation ». *IEEE Transactions on Nuclear Science*, vol. 61, n  6, p. 3535-3542.
- Das, Anup, Shyamsundar Venkataraman et Akash Kumar. 2013. « Improving autonomous soft-error tolerance of FPGA through LUT configuration bit manipulation ». In *2013 23rd International Conference on Field programmable Logic and Applications*. p. 1-8. IEEE.
- Datasheet, Xilinx. 2002. « Virtex-E 1.8 V Field Programmable Gate Array ». *DS022-1 (v2.3), Xilinx Inc.*
- Di Carlo, Stefano, Paolo Prinetto, Daniele Rolfo et Pascal Trotta. 2014. « A fault injection methodology and infrastructure for fast single event upsets emulation on Xilinx SRAM-based FPGAs ». In *2014 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. p. 159-164. IEEE.
- Dutton, Bradley F, et Charles E Stroud. 2009. « Single Event Upset Detection and Correction in Virtex-4 and Virtex-5 FPGAs ». In *CATA*. p. 57-62.
- Duzellier, S, D Falguere, R Ecoffet et I Tsourilo. 1997. « EXEQ II and III: On-board experiments for the study of single events ». In *Radiation and Its Effects on Components and Systems, 1997. RADECS 97. Fourth European Conference on*. p. 504-509. IEEE.
- Ebrahimi, Mojtaba, Adrian Evans, Mehdi B Tahoori, Enrico Costenaro, Dan Alexandrescu, Vikas Chandra et Razi Seyyedi. 2015a. « Comprehensive analysis of sequential and combinational soft errors in an embedded processor ». *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, n  10, p. 1586-1599.
- Ebrahimi, Mojtaba, Abbas Mohammadi, Alireza Ejlali et Seyed Ghassem Miremadi. 2014. « A fast, flexible, and easy-to-develop fpga-based fault injection technique ». *Microelectronics Reliability*, vol. 54, n  5, p. 1000-1008.
- Ebrahimi, Mojtaba, Nour Sayed, Maryam Rashvand et Mehdi B Tahoori. 2015b. « Fault injection acceleration by architectural importance sampling ». In *Proceedings of the 10th International Conference on Hardware/Software Codesign and System Synthesis*. p. 212-219. IEEE Press.

- Entrena, Luis, Celia López-Ongil, Mario García-Valderas, Marta Portela-García et Michael Nicolaidis. 2011. « Hardware fault injection ». In *Soft Errors in Modern Electronic Systems*. p. 141-166. Springer.
- ESA. « Flipper Product Sheet ».  
< [http://microelectronics.esa.int/techno/Flipper\\_ProductSheet.pdf](http://microelectronics.esa.int/techno/Flipper_ProductSheet.pdf) >.
- Falguere, D, S Duzellier et R Ecoffet. 1994. « SEE in-flight measurement on the MIR orbital station ». *IEEE transactions on nuclear science*, vol. 41, n° 6, p. 2346-2352.
- Faure, F, P Peronnard, R Velazco et R Ecoffet. 2002. « Thesic+: A flexible system for SEE testing ». In *Proc. of RADECS*.
- Faure, Fabien, Raoul Velazco et Paul Peronnard. 2005. « Single-event-upset-like fault injection: a comprehensive framework ». *IEEE transactions on nuclear science*, vol. 52, n° 6, p. 2205-2209.
- Foucard, Gilles. 2010. « Taux d'erreurs dues aux radiations pour des applications implémentées dans des FPGAs à base de mémoire SRAM: prédiction versus mesures ». Institut Polytechnique de Grenoble, 136 p.
- Gaillard, Rémi. 2011. « Single event effects: Mechanisms and classification ». In *Soft Errors in Modern Electronic Systems*. p. 27-54. Springer.
- Gajjar, Nagendra, Vijay Savani, NM Devashrayee et KS Dasgupta. 2011. « self healing system to mitigate single event upset in SRAM based FPGA for space application ». *Global J. of Engg. & Appl. Sciences*, vol. 1, n° 4.
- Gasiot, G, D Giot et P Roche. 2007. « Multiple cell upsets as the key contribution to the total SER of 65 nm CMOS SRAMs and its dependence on well engineering ». *IEEE Transactions on Nuclear Science*, vol. 54, n° 6, p. 2468-2473.
- Ghaffari, Fakhreddine, Fouad Sahraoui, Mohamed El Amine Benkhelifa, Bertrand Granado, Marc Alexandre Kacou et Olivier Romain. 2014. « Fast SRAM-FPGA fault injection platform based on dynamic partial reconfiguration ». In *2014 26th International Conference on Microelectronics (ICM)*. p. 144-147. IEEE.
- Gokhale, Maya, Paul Graham, Michael Wirthlin, D Eric Johnson et Nathaniel Rollins. 2006. « Dynamic reconfiguration for management of radiation-induced faults in FPGAs ». *International journal of embedded systems*, vol. 2, n° 1-2, p. 28-38.
- Goldhagen, Paul. 2000. « Overview of aircraft radiation exposure and recent ER-2 measurements ». *Health Physics*, vol. 79, n° 5, p. 526-544.

- Gosheblagh, Reza Omid, et Karim Mohammadi. 2013. « Dynamic partial based Single Event Upset (SEU) injection platform on FPGA ». *International Journal of Computer Applications*, vol. 76, n° 3.
- Graham, Paul, Heather Quinn, James Krone, Michael Caffrey et Sana Rezgui. 2005. « Radiation-Induced Multi-Bit Upsets in SRAM-Based FPGAs ». In *IEEE NSREC*. Vol. 5. Citeseer.
- Guenzer, CS, EA Wolicki et RG Allas. 1979. « Single event upset of dynamic RAMs by neutrons and protons ». *IEEE Transactions on Nuclear Science*, vol. 26, n° 6, p. 5048-5052.
- Guzman-Miranda, H, MA Aguirre et J Tombs. 2008. « A non-invasive system for the measurement of the robustness of microprocessor-type architectures against radiation-induced soft errors ». In *Instrumentation and Measurement Technology Conference Proceedings, 2008. IMTC 2008. IEEE*. p. 2009-2014. IEEE.
- Guzmán-Miranda, H, J Barrientos-Rojas et MA Aguirre. 2016. « A Fault Injection technique oriented to SRAM-FPGAs ». In *FPGAs and Parallel Architectures for Aerospace Applications*. p. 49-59. Springer.
- Herrera-Alzu, Ignacio, et Marisa Lopez-Vallejo. 2013. « Design techniques for Xilinx Virtex FPGA configuration memory scrubbers ». *IEEE Transactions on Nuclear Science*, vol. 60, n° 1, p. 376-385.
- Hobeika, Christelle, Simon Pichette, MA Leonard, Claude Thibeault, Jean-François Boland et Yves Audet. 2014. « Multi-abstraction level signature generation and comparison based on radiation single event upset ». In *2014 IEEE 20th International On-Line Testing Symposium (IOLTS)*. p. 212-215. IEEE.
- Huang, Yu, Ruifeng Guo, Wu-Tung Cheng et James Chien-Mo Li. 2008. « Survey of scan chain diagnosis ». *IEEE Design & Test of Computers*, vol. 25, n° 3, p. 240-248.
- Jenn, Eric, Jean Arlat, Marcus Rimen, Joakim Ohlsson et Johan Karlsson. 1994. « Fault injection into VHDL models: the MEFISTO tool ». In *Fault-Tolerant Computing, 1994. FTCS-24. Digest of Papers., Twenty-Fourth International Symposium on*. p. 66-75.
- Jing, Naifeng, Ju-Yueh Lee, Zhe Feng, Weifeng He, Zhigang Mao, Shi-Jie Wen, Rick Wong et Lei He. 2011. « Quantitative SEU fault evaluation for SRAM-based FPGA architectures and synthesis algorithms ». In *2011 21st International Conference on Field Programmable Logic and Applications*. p. 282-285.
- Juneau, Marc. 2010. « Méthode d'évaluation de la sensibilité aux radiations ». École de technologie supérieure.



- Karlsson, Johan, Ulf Gunneflo, Peter Lidén et Jan Torin. 1991. « Two fault injection techniques for test of fault handling mechanisms ». In *Test Conference, 1991, Proceedings., International*. p. 140. IEEE.
- Kenterlis, P, Nektarios Kranitis, A Paschalis, Dimitris Gizopoulos et Mihalis Psarakis. 2006. « A low-cost SEU fault emulation platform for SRAM-based FPGAs ». In *12th IEEE International On-Line Testing Symposium (IOLTS'06)*. p. 7 pp.: IEEE.
- Khatri, Abdul Rafay, Manuel Milde, Ali Hayek et Josef Börcsök. « Instrumentation Technique for FPGA based Fault Injection Tool ».
- Lajolo, Marcello, Maurizio Rebaudengo, M Sonza Roerda, Massimo Violante et Luciano Lavagno. 2000. « Evaluating system dependability in a co-design framework ». In *Proceedings of the conference on Design, automation and test in Europe*. p. 586-590. ACM.
- Lavin, Christopher, Marc Padilla, Jaren Lamprecht, Philip Lundrigan, Brent Nelson et Brad Hutchings. 2011. « Rapidsmith: Do-it-yourself cad tools for xilinx fpgas ». In *2011 21st International Conference on Field Programmable Logic and Applications*. p. 349-355. IEEE.
- Le, Robert. 2012. « Soft error mitigation using prioritized essential bits ». *Xilinx XAPP538 (v1. 0)*.
- Lee, Peter Ming-Han, et Reza Sedaghat. 2008. « FPGA-based switch-level fault emulation using module-based dynamic partial reconfiguration ». *Microelectronics Reliability*, vol. 48, n° 10, p. 1724-1733.
- Lesea, Austin, Saar Drimer, Joseph J Fabula, Carl Carmichael et Peter Alfke. 2005. « The rosetta experiment: atmospheric soft error rate testing in differing technology FPGAs ». *IEEE Transactions on Device and Materials Reliability*, vol. 5, n° 3, p. 317-328.
- Lesea, Austin, et J Fabula. 2008. « Continuing experiments of atmospheric neutron effects on deep submicron integrated circuits ». *WP286 (v1. 0), Xilinx Inc*, vol. 2.
- Leveugle, Régis. 2000. « Fault injection in VHDL descriptions and emulation ». In *Proceedings-IEEE-International-Symposium-on-Defect-and-Fault-Tolerance-in-VLSI-Systems*. p. 414-19.
- Li, Zhuo, Xiang Lu, Wangqi Qiu, Weiping Shi et DMH Walker. 2003. « A circuit level fault model for resistive opens and bridges ». In *VLSI Test Symposium, 2003. Proceedings. 21st*. p. 379-384. IEEE.

- Lopez-Ongil, Celia, Mario Garcia-Valderas, Marta Portela-Garcia et Luis Entrena. 2007. « Autonomous fault emulation: a new FPGA-based acceleration system for hardness evaluation ». *IEEE Transactions on Nuclear Science*, vol. 54, n° 1, p. 252-261.
- Lu, Weiyun, et Martin Radetzki. 2011. « Efficient fault simulation of systemc designs ». In *Digital System Design (DSD), 2011 14th Euromicro Conference on*. p. 487-494. IEEE.
- Madeira, Henrique, Mário Rela, Francisco Moreira et João Gabriel Silva. 1994. « RIFLE: A general purpose pin-level fault injector ». In *European Dependable Computing Conference*. p. 197-216. Springer.
- Martínez, RJ, PJ Gil, G Martín, C Pérez et JJ Serrano. 1999. « Experimental validation of high-speed fault-tolerant systems using physical fault injection ». In *Dependable Computing for Critical Applications 7, 1999*. p. 249-265. IEEE.
- May, Timothy C, et Murray H Woods. 1979. « Alpha-particle-induced soft errors in dynamic memories ». *IEEE Transactions on Electron Devices*, vol. 26, n° 1, p. 2-9.
- Michel, T, Regis Leveugle, Gabriele Saucier, R Doucet et P Chapier. 1994. « Taking advantage of ASICs to improve dependability with very low overheads [PLC] ». In *Proceedings.-The-European-Design-and-Test-Conference.-EDAC,-The-European-Conference-on-Design-Automation.-ETC-European-Test-Conference.-EUROASIC,-The-European-Event-in-ASIC-Design-Cat.-No. 94TH0634-6*. p. 14-18. IEEE Comput. Soc. Press, Los Alamitos, CA, USA.
- Mogollon, JM, H Guzman-Miranda, J Napoles, J Barrientos et MA Aguirre. 2011. « FTUNSHADES2: A novel platform for early evaluation of robustness against SEE ». In *Radiation and Its Effects on Components and Systems (RADECS), 2011 12th European Conference on*. p. 169-174. IEEE.
- Moran, A, K LaBel, M Gates, C Seidleck, R McGraw, M Broida, J Firer et S Sprehn. 1996. « Single event effect testing of the Intel 80386 family and the 80486 microprocessor ». *IEEE Transactions on Nuclear Science*, vol. 43, n° 3, p. 879-885.
- Mueller-Gritschneider, Daniel, Petra R Maier, Marc Greim et Ulf Schlichtmann. 2014. « System C-based multi-level error injection for the evaluation of fault-tolerant systems ». In *2014 International Symposium on Integrated Circuits (ISIC)*. p. 460-463. IEEE.
- National Instruments. 2012. « FPGA Fundamentals ». < <http://www.ni.com/white-paper/6983/en/> >.
- Normand, Eugene. 1996a. « Single-event effects in avionics ». *IEEE Transactions on nuclear science*, vol. 43, n° 2, p. 461-474.

- Normand, Eugene. 1996b. « Single event upset at ground level ». *IEEE transactions on Nuclear Science*, vol. 43, n° 6, p. 2742-2750.
- Normand, Eugene. 2001. « Correlation of inflight neutron dosimeter and SEU measurements with atmospheric neutron model ». *IEEE Transactions on Nuclear Science*, vol. 48, n° 6, p. 1996-2003.
- Nunes, José Luís, Tamás Pecserke, João Carlos Cunha et Mário Zenha-Rela. 2015. « FIRED-Fault Injector for Reconfigurable Embedded Devices ». In *Dependable Computing (PRDC), 2015 IEEE 21st Pacific Rim International Symposium on*. p. 1-10. IEEE.
- Peronnard, Paul, Raoul Velazco et Guillaume Hubert. 2009. « Real-life SEU experiments on 90 nm SRAMS in atmospheric environment: measures versus predictions done by means of MUSCA SEP3 platform ». *IEEE Transactions on Nuclear Science*, vol. 56, n° 6, p. 3450-3455.
- Quinn, Heather, Paul Graham, Jim Krone, Michael Caffrey, Sana Rezgui et Carl Carmichael. 2005. « Radiation-induced multi-bit upsets in Xilinx SRAM-based FPGAs ». In *Proc. Military and Aerospace Appl. of Prog. Logic Devices Conf.*
- Quinn, Heather, Paul Graham, Keith Morgan, Zachary Baker, Michael Caffrey, Dave Smith et Randy Bell. 2012. « On-orbit results for the Xilinx Virtex-4 FPGA ». In *2012 IEEE Radiation Effects Data Workshop*. p. 1-8. IEEE.
- Quinn, Heather, William H Robinson, Paolo Rech, Miguel Aguirre, Arno Barnard, Marco Desogus, Luis Entrena, Mario Garcia-Valderas, Steven M Guertin et David Kaeli. 2015. « Using Benchmarks for Radiation Testing of Microprocessors and FPGAs ». *IEEE Transactions on Nuclear Science*, vol. 62, n° 6, p. 2547-2554.
- Quinn, Heather, et Michael Wirthlin. 2015. « Validation Techniques for Fault Emulation of SRAM-based FPGAs ». *IEEE Transactions on Nuclear Science*, vol. 62, n° 4, p. 1487-1500.
- Robache, R, J-F Boland, C Thibeault et Y Savaria. 2013. « A methodology for system-level fault injection based on gate-level faulty behavior ». In *New Circuits and Systems Conference (NEWCAS), 2013 IEEE 11th International*. p. 1-4. IEEE.
- Rothbart, K, U Neffe, Ch Steger, R Weiss, E Rieger et A Muehlberger. 2005. « A smart card test environment using multi-level fault injection in SystemC ». In *6th IEEE Latin-American Test Workshop (LATW'2005) Digest of Papers*. p. 103-108.
- Sieh, Volkmar, Oliver Tschache et Frank Balbach. 1997. « VERIFY: Evaluation of reliability using VHDL-models with embedded fault descriptions ». In *Fault-Tolerant*

- Computing, 1997. FTCS-27. Digest of Papers., Twenty-Seventh Annual International Symposium on.* p. 32-36. IEEE.
- Singh, Satnam, et Philip James-Roxby. 2001. « Lava and JBits: From HDL to bitstream in seconds ». In *Field-Programmable Custom Computing Machines, 2001. FCCM'01. The 9th Annual IEEE Symposium on.* p. 91-100. IEEE.
- Sootkaneung, Warin, et Kewal K Saluja. 2010. « Gate input reconfiguration for combating soft errors in combinational circuits ». In *2010 International Conference on Dependable Systems and Networks Workshops (DSN-W).* p. 107-112. IEEE.
- Souari, Anis, Claude Thibeault, Yves Blaqui et Raoul Velazco. 2015a. « An automated fault injection for evaluation of LUTs robustness in SRAM-based FPGAs ». In *2015 IEEE East-West Design & Test Symposium (EWDTS).* p. 1-4.
- Souari, Anis, Claude Thibeault, Yves Blaquièrè et Raoul Velazco. 2015b. « Optimization of SEU emulation on SRAM FPGAs based on sensitiveness analysis ». In *2015 IEEE 21st International On-Line Testing Symposium (IOLTS).* p. 36-39.
- Souari, Anis, Claude Thibeault, Yves Blaquièrè et Raoul Velazco. 2016. « Towards an Efficient SEU Effects Emulation on SRAM-Based FPGAs ». *Microelectronics Reliability, Elsevier.*
- Sterpone, Luca, et Massimo Violante. 2005. « A new analytical approach to estimate the effects of SEUs in TMR architectures implemented through SRAM-based FPGAs ». *IEEE Transactions on Nuclear Science*, vol. 52, n° 6, p. 2217-2223.
- Sterpone, Luca, et Massimo Violante. 2007. « A new partial reconfiguration-based fault-injection system to evaluate SEU effects in SRAM-based FPGAs ». *IEEE Transactions on Nuclear Science*, vol. 54, n° 4, p. 965-970.
- Tambara, Lucas A, Jimmy Tarrillo, Fernanda L Kastensmidt et Luca Sterpone. 2016. « Fault-Tolerant Manager Core for Dynamic Partial Reconfiguration in FPGAs ». In *FPGAs and Parallel Architectures for Aerospace Applications.* p. 121-133. Springer.
- Tazi, Fatima Zahra, Claude Thibeault, Yvon Savaria, Simon Pichette et Yves Audet. 2014. « On Delay Faults Affecting I/O Blocks of an SRAM-Based FPGA Due to Ionizing Radiations ». *arXiv preprint arXiv:1409.0736.*
- Thaker, Pradip A, Vishwani D Agrawal et Mona E Zaghoul. 2000. « Register-transfer level fault modeling and test evaluation techniques for VLSI circuits ». In *Test Conference, 2000. Proceedings. International.* p. 940-949. IEEE.
- Thibeault, Claude, Simon Pichette, Yves Audet, Yvon Savaria, H Rufenacht, E Gloutnay, Yves Blaquièrè, F Moupfouma et N Batani. 2012. « On extra combinational delays in

- SRAM FPGAs due to transient ionizing radiations ». *IEEE Transactions on Nuclear Science*, vol. 59, n° 6, p. 2959-2965.
- Thibeault, Claude, Yassine Hariri, Syed Rafay Hasan, Christelle Hobeika, Yvon Savaria, Yves Audet et Fatima Zahra Tazi. 2013. « A library-based early soft error sensitivity analysis technique for SRAM-based FPGA design ». *Journal of Electronic Testing*, vol. 29, n° 4, p. 457-471.
- Tipton, Alan D, Jonathan A Pellish, John M Hutson, Robert Baumann, Xiaowei Deng, Andrew Marshall, Michael A Xapsos, Hak S Kim, Mark R Friendlich et Michael J Campola. 2008. « Device-orientation effects on multiple-bit upset in 65 nm SRAMs ». *IEEE Transactions on Nuclear Science*, vol. 55, n° 6, p. 2880-2885.
- UG190, Xilinx. 2009. « Virtex-5 FPGA user guide ». *UG190*, vol. 5.
- Vanhouwaert, Pierre. 2008. « Analyse de sureté par injection de fautes dans un environnement de prototypage à base de FPGA ». Institut polytechnique de Grenoble, 134 p.
- Velazco, R, Ph Cheynet et R Ecoffet. 1998. « Operation in space of artificial neural networks implemented by means of a dedicated architecture based on a transputer ». In *Integrated Circuit Design, 1998. Proceedings. XI Brazilian Symposium on*. p. 162-165. IEEE.
- Velazco, R, Ph Cheynet, A Tissot, J Haussy, J Lambert et R Ecoffet. 2000. « Evidences of SEU tolerance for digital implementations of Artificial Neural Networks: one year MPTB flight results ». In *1999-Fifth-European-Conference-on-Radiation-and-Its-Effects-on-Components-and-Systems.-RADECS-99*. p. 565-568. IEEE, Piscataway, NJ, USA.
- Velazco, Raoul, et Francisco J Franco. 2007. « Single event effects on digital integrated circuits: origins and mitigation techniques ». In *2007 IEEE International Symposium on Industrial Electronics*. p. 3322-3327.
- Virtex, Xilinx. 2012. « FPGA Configuration User Guide UG191 ».
- Wu, Xiaoxia, Paul Falkenstern, Krishnendu Chakrabarty et Yuan Xie. 2009. « Scan-chain design and optimization for three-dimensional integrated circuits ». *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 5, n° 2, p. 9.
- Zarandi, Hamid R, Seyed Ghassem Miremadi et Alireza Ejlali. 2003. « Fault injection into verilog models for dependability evaluation of digital systems ». In *Proceedings of the Second international conference on Parallel and distributed computing*. p. 281-287. IEEE Computer Society.

- Ziade, Haissam, Rafic A Ayoubi et Raoul Velazco. 2004. « A survey on fault injection techniques ». *Int. Arab J. Inf. Technol.*, vol. 1, n° 2, p. 171-186.
- Ziade, Haissam, Rafic A Ayoubi, Raoul Velazco et Tareck Idriss. 2011. « A new fault injection approach to study the impact of bitflips in the configuration of SRAM-based FPGAs ». *Int. Arab J. Inf. Technol.*, vol. 8, n° 2, p. 155-162.
- Ziegler, James F, Huntington W Curtis, Hans P Muhlfield, Charles J Montrose et B Chin. 1996. « IBM experiments in soft fails in computer electronics (1978–1994) ». *IBM journal of research and development*, vol. 40, n° 1, p. 3-18.