

Amélioration des méthodes d'émulation des pannes par les radiations cosmiques sur les FPGA

par

Mariem LAOUEJ

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE
AVEC MÉMOIRE EN GÉNIE ÉLECTRIQUE
M. Sc. A.

MONTRÉAL, LE 29 JUILLET 2021

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC



Mariem Laouej, 2021



Cette licence Creative Commons signifie qu'il est permis de diffuser, d'imprimer ou de sauvegarder sur un autre support une partie ou la totalité de cette oeuvre à condition de mentionner l'auteur, que ces utilisations soient faites à des fins non commerciales et que le contenu de l'oeuvre n'ait pas été modifié.

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE:

M. Claude Thibeault, directeur de mémoire
Département de génie électrique à l'École de technologie supérieure

M. Naïm Batani, président du jury
Département de génie électrique à l'École de technologie supérieure

M. Frédéric Nabki, membre du jury
Département de génie électrique à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 16 JUILLET 2021

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Je dédie ce travail à l'âme de mon père, cet homme formidable et dévoué qui a toujours cru en moi.

Je tiens à remercier M. Claude Thibeault, mon maître de mémoire, pour m'avoir confié la tâche de mener ce projet à bien ainsi que pour ses précieux conseils et recommandations judicieux sur ce projet et pour son support financier.

Un grand merci à ma famille et à mon ami Soufiene qui m'ont aidé à me remettre sur pied dans les moments de faiblesse.

Je remercie également les membres du jury pour leur acceptation de l'évaluation du mémoire.

Finalement, je tiens à adresser ma sincère gratitude au programme de bourses d'études mondiales du gouvernement canadien Mitacs qui m'a permis de faire partie de cet incroyable réseau de recherche.

Amélioration des méthodes d'émulation des pannes par les radiations cosmiques sur les FPGA

Mariem LAOUEJ

RÉSUMÉ

La terre est bombardée par un flux de particules chargées énergétiques appelées rayons cosmiques. Les effets des rayons cosmiques sur la microélectronique sont connus et sont de plus en plus préoccupants, en particulier pour les systèmes embarqués des avions volant à des altitudes de plus en plus élevées. En effet, la diminution de la taille des transistors a fait en sorte que les circuits sont en général plus sensibles aux radiations cosmiques.

Dans ce contexte, il est important d'investiguer la réaction de ces systèmes embarqués face aux radiations. Si les tests accélérés sous faisceaux de particules demeurent la manière la plus réaliste de faire cette investigation, d'autres techniques, telle l'émulation, permettent d'initier cette tâche. L'objectif de ce projet est l'amélioration d'une méthode d'injection de pannes par émulation en terme de représentativité ainsi qu'en terme de rapidité pour la reproduction la plus fidèle possible des résultats des tests sous faisceaux de neutrons obtenus au laboratoire TRIUMF de Vancouver.

Dans ce mémoire, une série d'expérimentations d'injection de pannes est effectuée tout en ciblant des parties précises de la mémoire de configuration des circuits utilisés. Les circuits utilisés par nos expérimentations sont les FPGA à base de mémoire SRAM, qui représentent un choix populaire dans les environnements aéronautiques. Le type des pannes ciblées, très fréquemment rencontré dans les composants de mémoire SRAM, est le SEU (single event upset), qui constitue l'effet habituel du passage d'un neutron dans une région sensible d'un microcircuit.

Nous proposons une nouvelle méthode qui consiste à attaquer les parties essentielles des bits des LUT (Look Up Table), représentant la plus petite entité configurable du FPGA permettant la mise en œuvre des fonctions logiques simples, ainsi que des bits Non LUT, représentant les autres bits de configuration, tout en considérant la différence de sensibilité entre ces bits. Cette approche est efficace, elle contribue à la réalisation de notre objectif, de plus, elle permet une estimation de nombre des bits critiques.

Mots-clés: FPGA à mémoire SRAM, SEU, injection de pannes, effet de radiations, benchmarks

Improvement of fault injection methods under the effects of cosmic radiations

Mariem LAOUEJ

ABSTRACT

The earth is bombarded by a stream of energetic charged particles called cosmic rays. The effects of cosmic rays on microelectronics are known and are of increasing concern, especially for on-board systems in airplanes flying at ever-higher altitudes. Indeed, the decrease in the size of transistors has resulted in circuits that are generally more sensitive to cosmic radiation.

In this context, it is important to investigate the reaction of these embedded systems to radiation. If accelerated tests under particle beams remain the most realistic way to do this investigation, other techniques, such as emulation, allow to initiate this task. The objective of this project is the enhancement of a method of fault injection by emulation in terms of representativeness as well as in terms of speed for the most faithful reproduction of the results of the tests under neutron beams obtained at the TRIUMF laboratory in Vancouver.

In this thesis, a series of fault injection experiments are performed while targeting specific parts of the configuration memory of the circuits used. The circuits used in our experiments are SRAM-based FPGAs, which are a popular choice in aerospace environments. The type of targeted failures, very frequently encountered in SRAM memory components, is the SEU (single event upset), which is the usual effect of a neutron passing through a sensitive region of a microcircuit.

We propose a new method, which consists in attacking the essential parts of the LUT (Look Up Table) bits, representing the smallest configurable entity of the FPGA allowing the implementation of simple logic functions, as well as the Non-LUT bits, representing the other configuration bits, while considering the difference in sensitivity between these bits. This approach is efficient, it contributes to the achievement of our objective, and it allows an estimation of the number of critical bits.

Keywords: SRAM FPGA, SEU, fault injection, radiation effect, benchmarks

TABLE DES MATIÈRES

	Page
INTRODUCTION	1
CHAPITRE 1 NOTIONS DE BASE	5
1.1 Introduction	5
1.2 Environnements radiatifs	5
1.2.1 Environnement radiatif spatial	5
1.2.2 Environnement radiatif atmosphérique	8
1.3 Les erreurs causées par les radiations cosmiques sur les circuits électroniques	10
1.3.1 Effets destructifs	11
1.3.2 Effets non destructifs	11
1.4 Architecture des FPGA à mémoire SRAM	12
1.4.1 Couche applicative(opérative)	14
1.4.2 Couche de configuration	16
1.5 Configuration du FPGA ARTIX-7	16
1.6 Conclusion	17
CHAPITRE 2 ÉTAT DE L'ART	19
2.1 Introduction	19
2.2 Techniques d'injection de pannes	19
2.2.1 Injection de pannes dans des circuits physiques	20
2.2.2 Injection de pannes par simulation	21
2.2.3 Injection de pannes par émulation	23
2.3 Optimisation de la procédure d'injection de pannes	25
2.4 Signatures	26
2.4.1 Le concept de signature	26
2.4.2 Plateforme de génération de signatures	27
2.4.3 Classification des signatures	31
2.5 Conclusion	31
CHAPITRE 3 MÉTHODOLOGIE PROPOSÉE	33
3.1 Introduction	33
3.2 Circuits de référence utilisés	33
3.3 Outil d'injection de pannes : SEM <i>Controller</i>	34
3.4 Classification des bits de configuration selon leur niveau logique	37
3.5 Identification des bits de LUT de l'ARTIX-7	39
3.6 Méthode d'identification des bits essentiels de LUT de l'ARTIX-7	43
3.7 Procédure d'injection de pannes	45
3.8 Conclusion	48

CHAPITRE 4	AMÉLIORATIONS DES MÉTHODES D'INJECTION DE PANNES	49
4.1	Introduction	49
4.2	Injection de pannes aléatoires	50
4.3	Injection de pannes basée sur la sensibilité relative entre les bits de configuration par rapport à leur contenu	50
4.4	Injection de pannes dans les LUT basée sur la différence de sensibilité relative en fonction du contenu et du type de ressource	52
4.5	Injection de pannes dans les LUT essentiels et les Non LUT essentiels basée sur la sensibilité globale	55
4.6	Comparaison et discussion des résultats des différentes méthodes	56
4.7	Conclusion	60
CONCLUSION	61
RECOMMANDATIONS	63
ANNEXE I	SEM CONTROLLER	65
ANNEXE II	STRUCTURE FICHIER EBC ET EBD	67
ANNEXE III	INTERFACE <i>INJECTOR</i>	69
ANNEXE IV	PROCÉDURE D'IDENTIFICATION DES BITS CRITIQUES	71
ANNEXE V	EXEMPLE DE FICHIER DE CONTRAINTES XDC	73
ANNEXE VI	VÉRIFICATION DE L'EXACTITUDE DES ADRESSES GÉNÉRÉES	75
ANNEXE VII	ALGORITHME DE GÉNÉRATION DE SÉQUENCE DE TEST	77
BIBLIOGRAPHIE	79

LISTE DES TABLEAUX

	Page
Tableau 3.1	Répartition des bits de configuration des trois circuits 39
Tableau 3.2	Nombre des bits essentiels et non essentiels des LUT et Non LUT du circuit B01 44
Tableau 3.3	Nombre des bits essentiels et non essentiels des LUT et Non LUT du circuit B05 45
Tableau 3.4	Nombre des bits essentiels et non essentiels des LUT et Non LUT du circuit B12 45
Tableau 4.1	Nombre des bits inversés dans les fichiers de relecture après bombardement..... 53
Tableau 4.2	Facteurs de sensibilité globale 56
Tableau 4.3	Résultats d'injection de B01 vs TRIUMF 57
Tableau 4.4	Résultats d'injection de B05 vs TRIUMF 57
Tableau 4.5	Résultats d'injection de B12 vs TRIUMF 57
Tableau 4.6	Racine carrée de l'Erreur quadratique Moyenne moyenne, RCEQM, de trois circuits de test vs TRIUMF 58
Tableau 4.7	Nombre des bits inversés pour les trois circuits de test 60

LISTE DES FIGURES

	Page
Figure 1.1	La magnétosphère et ses déformations 6
Figure 1.2	Image d’une éruption solaire 7
Figure 1.3	Représentation des ceintures de radiations 8
Figure 1.4	Illustration de la production de particules secondaires dans l’atmosphère suite à l’interaction d’un proton de 10^{15} eV à une altitude de 35 km donnant naissance à une cascade de particules..... 9
Figure 1.5	Éléments génériques du FPGA 13
Figure 1.6	Cellule SRAM 14
Figure 1.7	Partitionnement d’un FPGA en deux couches 15
Figure 1.8	La structure d’un bloc logique configurable (CLB) 15
Figure 1.9	Architecture de la cellule SRAM 16
Figure 2.1	Bascule instrumentée pour la méthode Scan chain 24
Figure 2.2	Plateforme de génération : partie DUT 28
Figure 2.3	Plateforme de génération : partie de référence 30
Figure 3.1	Schéma global de communication 36
Figure 3.2	Relation hiérarchique de la configuration avec les bits essentiels 37
Figure 3.3	Procédure de génération des listes de bits de configuration à 0 et à 1 38
Figure 3.4	Connexion des LUT de design du FPGA ARTIX-7 41
Figure 3.5	Plage des adresses des tranches de l’ARTIX-7 A200T 42
Figure 3.6	Design initialisant les LUT de FPGA ARTIX-7 de la Région1_B, (b) (c) (d) adresses non accessibles 43
Figure 3.7	Algorithme de classification des bits de configuration 44
Figure 3.8	Procédure d’injection de pannes 47

Figure 4.1	Algorithme de sélection de la liste des adresses avec sensibilité relative 1,8	52
------------	--	----

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ATPG	Automatic Test Pattern Generation
BRAM	Block Random Access Memory
CLB	Configurable Logic Block
DEL	Diode ÉlectroLuminescente
DUT	Device Under Test
ÉTS	École de Technologie Supérieure
FF	Flip-Flop - Bascule
FPGA	Field Programmable Gate Array
IOB	Input/Output Block
LL	Lockup Latch
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
NCD	Native Circuit Description
RTL	Register Transfer Level
SEB	Single Event Burnout
SEGR	Single Event Gate Rupture
SEE	Single Event Effect
SEL	Single Event Latch-Up
SEM	Soft Error Mitigation
SET	Single Event Transient

SEU	Single Event Upset
SHE	Single event Hard Error
SRAM	Static Random Access Memory
SPARC	Scalable Processor Architecture
UART	Universal Asynchronous Receiver/Transmitter
XDC	Xilinx Design Constraint

LISTE DES SYMBOLES ET UNITÉS DE MESURE

UNITÉS ÉLECTRIQUES

eV	electron-volt
KeV	kiloelectron-volt
MeV	Megaelectron-volt
GeV	Gigaelectron-volt

AUTRES UNITÉS

s	seconde
cm	centimètre
km	Kilomètre

INTRODUCTION

Contexte global

Dans le but de minimiser la consommation de carburant de leurs avions, les compagnies aériennes visent à faire voler leurs appareils à des altitudes plus élevées qu'actuellement. Cependant, malgré les avantages que cela représente, les avions, et en particulier pour ce projet, les composants électroniques embarqués, sont alors moins protégés par l'atmosphère et plus exposés aux radiations cosmiques.

Dans ce cadre, ce projet de maîtrise s'inscrit dans la suite de travaux effectués dans deux autres projets : 1) le projet CRIAQ AVIO403, dont l'objectif général consistait à développer des stratégies pour caractériser l'influence des rayons cosmiques sur les circuits électroniques et plus précisément sur les divers éléments de FPGA à technologie SRAM, et 2) le projet international Electromagnetic Platform for lightweight Integration/Installation of electrical systems in Composite Aircraft (EPICEA), qui se concentrait, entre autres, sur l'étude des effets des rayonnements sur l'électronique embarquée dans les avions.

Nos travaux se concentrent sur l'amélioration des résultats du projet CRIAQ AVIO403 tout en utilisant la plateforme de génération des signatures développée dans le projet EPICEA.

Contexte du projet

Les FPGA utilisant la technologie SRAM font office de choix populaire pour l'électronique embarqué dans les environnements aéronautiques et spatiaux. Ils peuvent être fabriqués avec les procédés de fabrication les plus récents (ex. CMOS 10 nm pour les FPGA d'Intel), ce qui leur confère l'avantage d'un ou plusieurs nœuds technologiques, offrant des circuits plus complexes, plus rapides et moins énergivores, par rapport aux autres technologies programmables tel que la technologie Flash (à 28 nm en ce moment) et celle des antifusibles (à 150 nm). Ces

avantages ont poussé les fabricants de produits aéronautiques à les adopter dans leurs systèmes électroniques embarqués.

Cependant, dans le contexte du rayonnement cosmique, les FPGA basés sur la technologie de mémoire SRAM présentent un inconvénient majeur par rapport aux autres technologies programmables : la sensibilité à ce rayonnement. En effet, ils sont plus fragiles face aux perturbations induites par le rayonnement cosmique, appelées événements singuliers (Single Event Upset, SEU). Ces derniers sont en mesure d'inverser les bits de mémoire SRAM des FPGA, ce qui peut affecter la fonctionnalité du design sous-jacent.

Face à cela, des nombreux travaux dans la littérature ont proposé divers moyens en vue d'évaluer la sensibilité de ces circuits par rapport aux perturbations des radiations cosmiques. Cette évaluation constitue la première étape dans la quête de solutions pour accroître la robustesse de ces circuits. Parmi ces moyens, notons les techniques d'injection de pannes.

Objectif du projet

Le projet a pour ambition d'améliorer une technique d'injection de pannes récemment proposée, tant au niveau de sa représentativité par rapport à des tests accélérés de bombardement de particules qu'au niveau de la rapidité avec laquelle les injections sont effectuées. De manière plus spécifique, l'objectif ultime de ce projet est la reproduction de manière aussi précise que possible les résultats des tests sous faisceaux de neutrons au laboratoire national canadien de physique des particules, de physique nucléaire et de science fondée sur les accélérateurs nommé TRIUMF à Vancouver (Canada), tout en préservant une telle représentativité et réalisme dans les approches effectuées et en garantissant une amélioration dans la vitesse de la procédure d'injection de pannes. La nouvelle technique proposée sera validée à l'aide d'une série d'injection de pannes par émulation sur le FPGA tout en ciblant des parties précises de la mémoire de configuration. Le montage expérimental du projet EPICEA sera utilisé à cette fin.

Plan du mémoire

Le mémoire est organisé comme suit : dans le premier chapitre, on retrouve des notions de base sur différents aspects, tels les environnements radiatifs, l'effet des radiations cosmiques sur les circuits électroniques, la structure des FPGA basé sur la mémoire statique et la configuration de FPGA ARTIX-7 utilisé dans nos expérimentations. Le deuxième chapitre présente les principales techniques d'injection de pannes décrites dans la littérature, ainsi que les principales approches d'optimisation des procédures d'injection de pannes. Il inclut également une description de la plateforme d'encapsulation d'EPICEA utilisée pour la génération des signatures afin de détecter le comportement fautif du circuit. Le troisième chapitre introduit la méthodologie proposée dans ce mémoire. Dans le quatrième chapitre, on retrouve une description des différentes approches implémentées ainsi qu'une présentation et validation des résultats obtenus. Enfin une conclusion générale clôture ce mémoire.

Contributions du projet

Le projet a mené aux contributions suivantes :

- une nouvelle approche rapide d'injection ciblant des sous-ensembles précis des bits de configurations dits essentiels, en fonction du type de ressources à configurer, de leur valeur logique ('0' ou '1'), et de la sensibilité relative de ces sous-ensembles face aux radiations ;
- l'identification des bits de configuration servant à programmer le contenu des tables de conversion du FPGA ARTIX-7 de Xilinx utilisé durant les expérimentations.

CHAPITRE 1

NOTIONS DE BASE

1.1 Introduction

L'objectif de ce chapitre, composé de différents axes, est de couvrir quelques notions de base utiles pour la compréhension de ce projet, telles que les environnements radiatifs, dont l'intérêt est de saisir comment ils agissent sur les circuits électroniques, la structure des circuits FPGA, plus précisément ceux à mémoire SRAM, ainsi que la configuration de FPGA ARTIX-7 de la compagnie Xilinx.

1.2 Environnements radiatifs

L'environnement radiatif est composé par plusieurs types de rayonnements ionisants comme les rayons X et les particules alpha. Toutefois, ceux qui perturbent le plus l'électronique embarquée sont les ions lourds, les protons et les neutrons. De façon générale, l'environnement radiatif est divisé en deux types : spatial et atmosphérique.

1.2.1 Environnement radiatif spatial

Dans l'environnement spatial on distingue quatre principales sources de rayonnement : le vent solaire, les éruptions solaires, le rayonnement cosmique ainsi que les ceintures de radiation. Une émission des particules énergétiques diverses, à partir de ces environnements, est composée principalement des photons, des protons, d'électrons et d'ions lourds (Boudenot, 1996).

- 1) **le vent solaire** : le monde interplanétaire est en permanence balayé par un vent de charges électriques composés d'électrons et d'ions de densité de l'ordre de 10^{12} particules/cm³ sur le plan solaire (et de 10 particules/cm³ sur le plan terrestre) (Luu, 2009). La vitesse moyenne de vent solaire est de 400km.s⁻¹, et elle peut être augmentée par les éruptions solaires.

À son contact avec la Terre et son champ magnétique, une cavité centrale magnétosphérique est créée, jouant le rôle de bouclier protégeant la Terre contre les émissions radiatives.

L'interaction du vent solaire et son effet sur la magnétosphère terrestre sont présentés dans la figure 1.1.

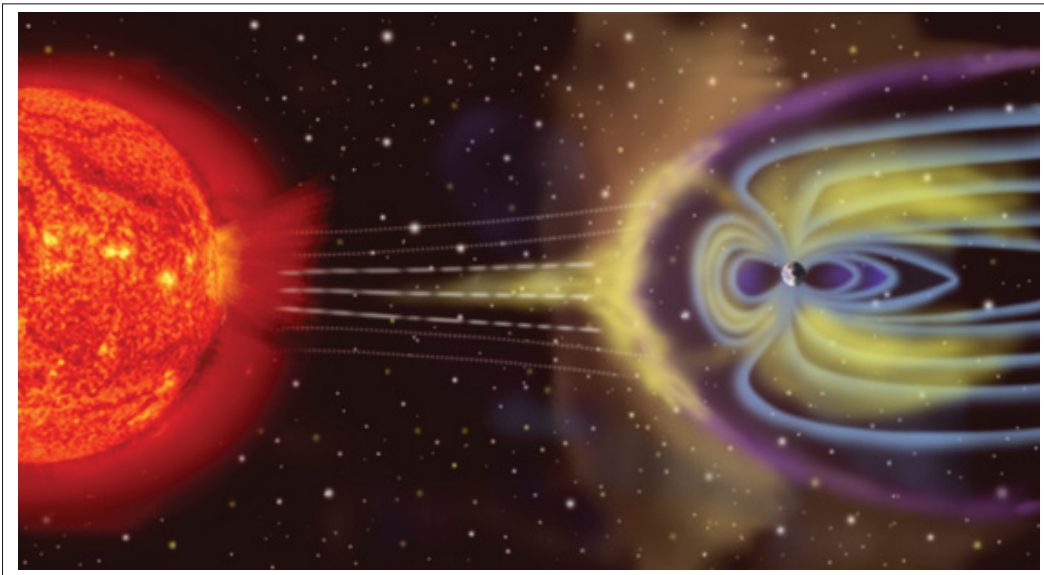


Figure 1.1 La magnétosphère et ses déformations
Tirée de Fabrice (2018)

- 2) **les éruptions solaires** : l'activité solaire est cyclique. La durée d'un cycle varie entre 9 et 13 ans. Chaque cycle est donc constitué de périodes d'années actives et d'autres calmes. La période d'activité maximale se distingue par le nombre d'évènements observés sur la surface du soleil, principalement l'apparition d'éruptions solaires (Figure 1.2).

Les éruptions solaires présentent un phénomène fondamental dans l'activité solaire, qui apparaît à la surface de la photosphère (Luu, 2009). Elles se manifestent dans une libération soudaine d'énergie sous forme d'ondes électromagnétiques sur un large spectre d'un intense rayonnement (des ondes radio aux rayons gamma passant par des rayons UV, rayons X, etc.) et des particules énergétiques (essentiellement des protons).

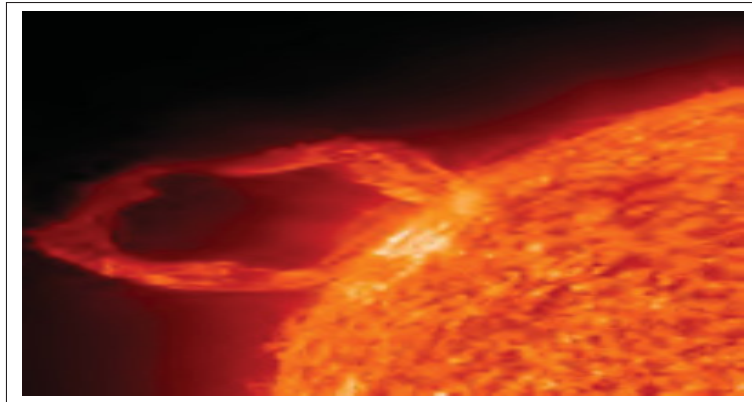


Figure 1.2 Image d'une éruption solaire
Tirée de Nasa (2010)

Deux sortes d'éruptions solaires qu'on peut distinguer selon la nature des matières qui sont émises :

- les éruptions solaires à ions lourds, de valeur énergétique entre 1 et 10 MeV et d'un nombre atomique qui peut dépasser 44. Ces éruptions durent quelques heures au plus ;
- les éruptions solaires à protons (particules d'hydrogène ionisées) dont les particules émises sont principalement des protons d'énergie importante franchissant des centaines de MeV. Ces éruptions durent de quelques heures à quelques jours.

- 3) **le rayonnement cosmique** : les premières traces des rayons cosmiques ont été découvertes au début du 20^e siècle à travers des mesures relevées à l'aide de ballons sondes (Souari, 2016). Les origines de ce rayonnement sont mal connues et sont supposées être des sources galactiques produisant des ions à différents niveaux d'énergie.

Les rayons cosmiques sont composés de différents éléments énergétiques de l'ordre de 10^2 eV jusqu'à plusieurs GeV. Les protons constituent en général 87% de leur composition, les ions lourds 1% et les 12% restant sont des noyaux d'hélium ;

- 4) **les ceintures de radiations** : les ceintures de radiations ou les ceintures de Van Allen ont été découvertes en 1958 (Souari, 2016). Celles-ci sont composées de particules (électrons, protons, ions lourds) d'énergie varie entre d'une dizaine de KeV à des centaines de MeV. Ces particules sont piégées d'une façon plus moins stable dans la magnétosphère terrestre à cause du contact avec le champ magnétique de la Terre.

On distingue trois ceintures de Van Allen (Figure 1.3) :

- une ceinture intérieure principalement constituée par des protons à haute énergie, à une altitude qui varie entre 700 km et 10000 km ;
- deux ceintures extérieures d'électrons d'énergie (> 5 MeV) centrées aux altitudes plus large de 13000 km et 65000 km.

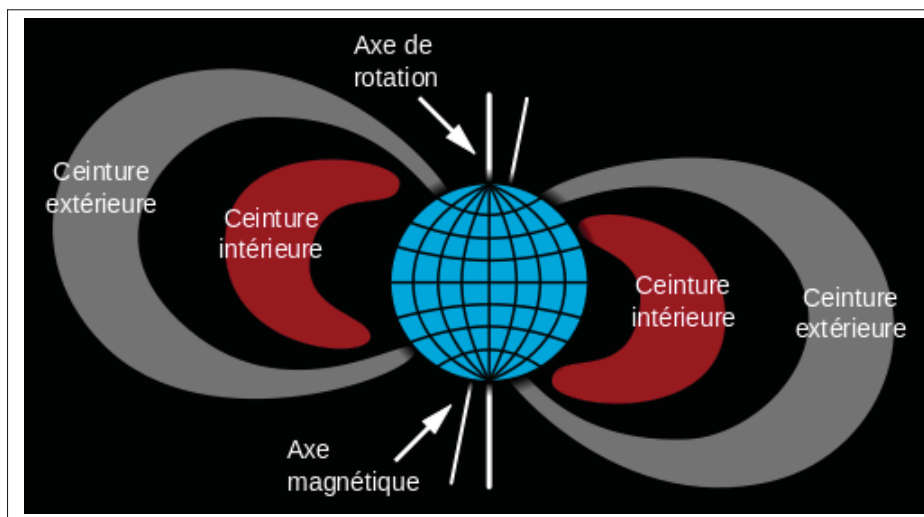


Figure 1.3 Représentation des ceintures de radiations
Tirée de Foucard (2010)

1.2.2 Environnement radiatif atmosphérique

Les effets de cet environnement sont moins dangereux que ceux de l'environnement spatial, mais ils ne sont pas potentiellement moins néfastes (Luu, 2009). En effet, plus les particules entrent dans l'atmosphère, plus il y aura une diminution des niveaux énergétiques et une désintégration des rayonnements cosmiques en différentes particules. Cela est dû à leur contact avec les atomes d'oxygène et d'azote (Bolchini & Sandionigi, 2010).

Il existe deux types de rayonnement atmosphérique : le rayonnement artificiel produit par l'homme et causé par des impuretés radioactives et le rayonnement naturel, qui est composé de diverses particules provenant du soleil ou d'origine extra galactique.

- 1) **les rayonnements artificiels** : ce genre de rayonnement cause la contamination de la fabrication des circuits électriques à la suite des impulsions radioactives. La compagnie IBM a constaté en 1987 une valeur d'erreur importante pendant les tests de différents produits (Ziegler *et al.*, 1996), le problème était dû à la contamination causée par la présence d'un émetteur artificiel, le polonium ;
- 2) **les rayonnements naturels** : les particules de haute énergie (des centaines de GeV/nucléon) venant des rayons cosmiques ne sont pas capturées par le champ magnétique de la Terre (Luu, 2009).

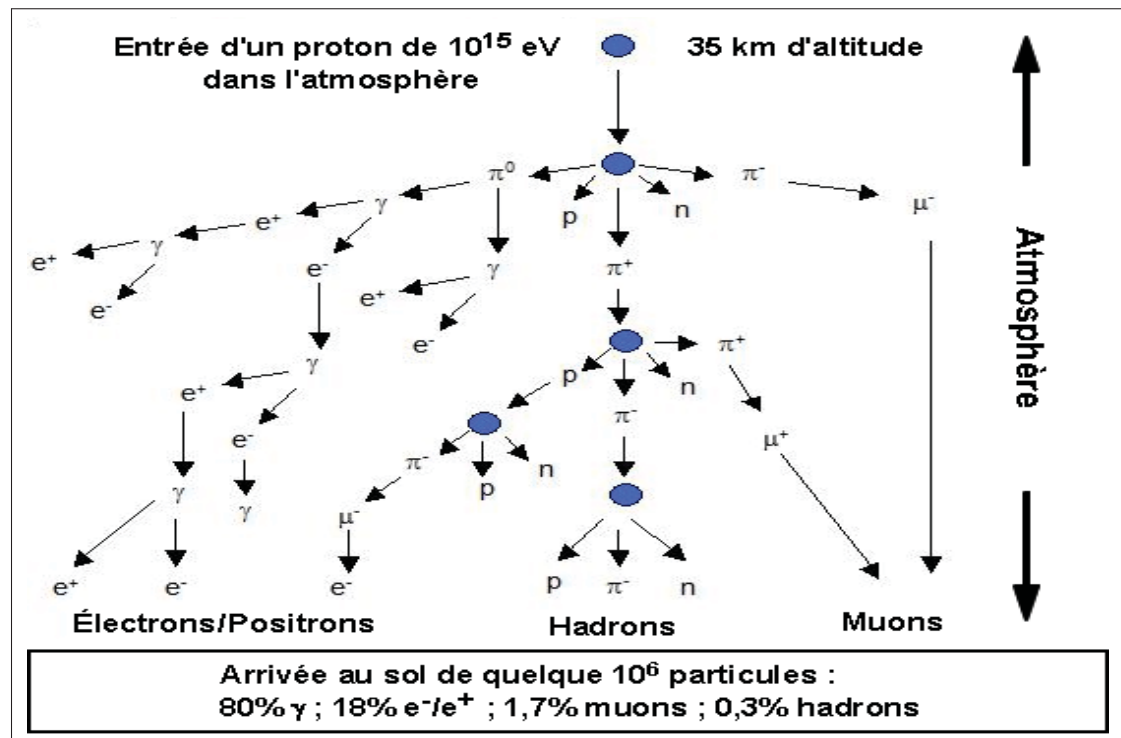


Figure 1.4 Illustration de la production de particules secondaires dans l'atmosphère suite à l'interaction d'un proton de 10^{15} eV à une altitude de 35 km donnant naissance à une cascade de particules
Tirée de Artola (2011)

La collision de ces particules avec les atomes dans l'atmosphère a deux principales conséquences : premièrement, elles ionisent des éléments de l'atmosphère, ce qui leur fait perdre une certaine quantité d'énergie. Deuxièmement, elles induisent des réactions nucléaires

sur ces éléments en chaîne, constituant une série de particules secondaires, ce phénomène étant nommé « la douche des particules ».

La figure 1.4 montre la génération de particules secondaires. Notons principalement les électrons, les protons, les neutrons, les pions, les muons et les photons (Luu, 2009). Toutes ces différentes particules sont capables d'interagir avec les molécules présentes dans l'atmosphère. En raison de ces interactions et des diminutions d'énergie qui en résultent, le flux de rayonnement décroît à mesure qu'il se rapproche du sol. Ainsi, son intensité est environ 300 fois moindre au niveau de la mer qu'à une altitude de 12 km. De plus, il est notamment variable en termes de latitude et est quatre fois plus fort aux pôles qu'à l'équateur.

La figure 1.4 illustre également la production en série de ces particules à partir d'un seul rayon cosmique d'énergie.

Dans ce projet, ce sont les altitudes des vols commerciaux qui nous intéressent le plus, où les particules principales sont les neutrons. En effet, les neutrons sont capables de provoquer des pannes dans les dispositifs intégrés en raison de leurs interactions avec les atomes qui composent ces derniers. La valeur de flux de neutrons atmosphériques est dépendante de l'activité solaire. Elle est estimée à $10 \text{ particules/cm}^2/\text{h}$ au niveau de la mer, et d'une valeur de $10^4 \text{ particules/cm}^2/\text{h}$ au niveau des vols commerciaux (Foucard, 2010).

1.3 Les erreurs causées par les radiations cosmiques sur les circuits électroniques

Ces erreurs interviennent dans l'environnement radiatif atmosphérique et spatial. Elles peuvent être causées : 1) par une unique particule en collision avec une partie sensible de circuit, ceci correspondant aux effets/événements dits singuliers, ou 2) par un phénomène cumulatif des charges au niveau d'oxydes isolants des circuits électriques qui peut mener à leurs dysfonctionnements, ce qui correspond aux effets de dose.

Un projet, élaboré par Xilinx (Lesea *et al.*, 2005), a montré l'influence des radiations cosmiques sur les cartes FPGAs. Pour cela, afin de quantifier et mesurer ces effets, une centaine des

cartes FPGA de différentes technologies et gammes ont été configurées et placées à plusieurs altitudes.

Dans ce qui suit, nous allons détailler les événements singuliers qui sont répartis en deux catégories : les effets destructifs et les effets non destructifs.

1.3.1 Effets destructifs

Les effets destructifs correspondent à l'augmentation d'un paramètre d'état (tension, courant, etc.) entraînant des erreurs non réparables qui mènent à une perte totale de fonctionnalité.

- **SEL (Single Event Latchup)** : c'est un événement qui entraîne le déclenchement d'un thyristor parasite, créant un courant de fonctionnement élevé, supérieur aux spécifications du circuit ; ce courant peut détruire le composant électronique et ne peut être annulé que par une coupure de l'alimentation pour protéger le circuit (Moran *et al.*, 1995) ;
- **SEB (Single Event Burnout)** : un événement dans lequel une particule énergétique induit un état localisé de courant élevé dans un transistor de puissance, ce qui entraîne sa défaillance ;
- **SEGR (Single Event Gate Rupture)** : un événement dans lequel une particule énergétique entraîne une rupture et un chemin conducteur subséquent à travers l'oxyde de grille d'un MOSFET. Un SEGR se manifeste par une augmentation du courant de fuite de la grille et peut entraîner soit la dégradation soit la destruction complète du dispositif ;
- **SHE (Single event Hard Error)** : modification irréversible du fonctionnement résultant d'un événement de rayonnement et généralement associée à des dommages permanents causés à un ou plusieurs éléments d'un dispositif.

1.3.2 Effets non destructifs

Les effets non destructifs sont principalement dus à une modification transitoire (changement de courant) pour laquelle les défaillances générées sont réparables.

- **SEU (Single Event Upset)** : cette panne intervient plus fréquemment dans les mémoires de type Random Access Memory (RAM), telles que la SRAM et la SDRAM. Elle est générée à la suite d'un bombardement d'une zone sensible de la cellule mémoire. La panne se manifeste dans le changement de l'état logique d'un bit par son inverse. Cet événement est réversible ; en effet, une simple réécriture de ce bit peut généralement remédier au problème ;
- **SET (Single Event Transient)** : celui-ci consiste en une perturbation de tension ou de courant qui peut se propager (mais pas obligatoirement) dans le circuit. Ce dernier est provoqué par les charges collectées pendant la propagation d'une particule ionisante dans la cellule. Le courant parasite qui en résulte peut provoquer des perturbations dans la réponse du composant et dans le système entier ;
- **MCU (Multiple Cell Upset)** : un événement unique qui provoque la défaillance simultanée de plusieurs bits dans un circuit intégré. Les bits en erreur sont généralement, mais pas toujours, physiquement adjacents. La probabilité de ce type des pannes est plus élevée lorsque la taille du transistor diminue ;
- **MBU (Multiple Bit Upset)** : ceci présente un cas particulier de MCU, i.e. un dérangement dans lequel deux ou plusieurs bits en erreur se produisent dans le même mot logique. Un MCU ne peut pas être corrigé par un simple ECC (Error Correcting Code) à un seul bit.

Dans ce mémoire, les événements à effets non destructifs de type SEU et MBU sont principalement les événements qui nous intéressent en raison de leur responsabilité des défaillances les plus fréquemment confrontées dans les FPGA basés sur la SRAM dans l'environnement radiatif atmosphérique (Foucard, 2010).

1.4 Architecture des FPGA à mémoire SRAM

Les FPGA sont des circuits intégrés qui peuvent être programmés et reconfigurés, permettant la conception des circuits numériques composés de centaines de milliers (voire millions) de cellules logiques. Ces dernières déterminent leur fonctionnalité et sont organisées dans une

matrice et connectées entre elles à travers un large réseau d'interconnexions configurables (Seifoori *et al.*, 2018).

Depuis la création des FPGA, ces circuits sont fortement utilisés dans le domaine de systèmes électroniques, Ils permettent de synthétiser différents types de circuits numériques : des processeurs intégrés au calcul parallèle de haute performance afin d'assurer un prototypage rapide ainsi qu'un temps de conception minimal.

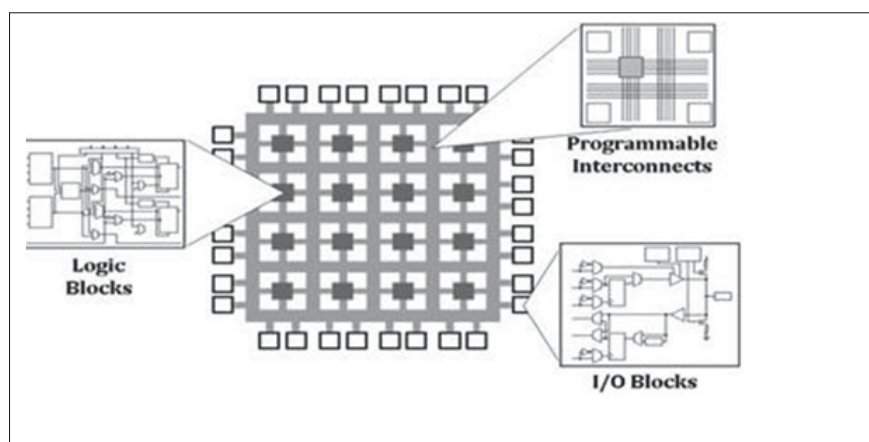


Figure 1.5 Éléments génériques du FPGA
Tirée de Seifoori *et al.* (2018)

Un FPGA est constitué essentiellement de trois types de composants génériques qui implantent les circuits, comme le montre la figure 1.5 :

- les blocs logiques configurables,
- le réseau d'interconnexions programmables,
- les blocs d'entrée/sortie.

L'architecture de style îlot est une architecture typique de FPGA, utilisée par les principaux fournisseurs commerciaux de FPGA (tels que Xilinx et Intel/Altera). Dans cette architecture, les blocs logiques sont organisés sous la forme d'un tableau bidimensionnel au sein d'un ensemble de ressources de routage. Notons qu'en plus des composants génériques, les FPGA

peuvent également contenir des modules dédiés tel des unités de calculs arithmétiques et des processeurs embarqués.

Dans le contexte de notre projet, le FPGA utilisé dans nos expérimentations est le ARTIX-7 de Xilinx basé sur la SRAM dont la cellule mémoire peut contenir 5 ou 6 transistors.

La figure 1.6 montre un exemple de cellule SRAM à 6 transistors

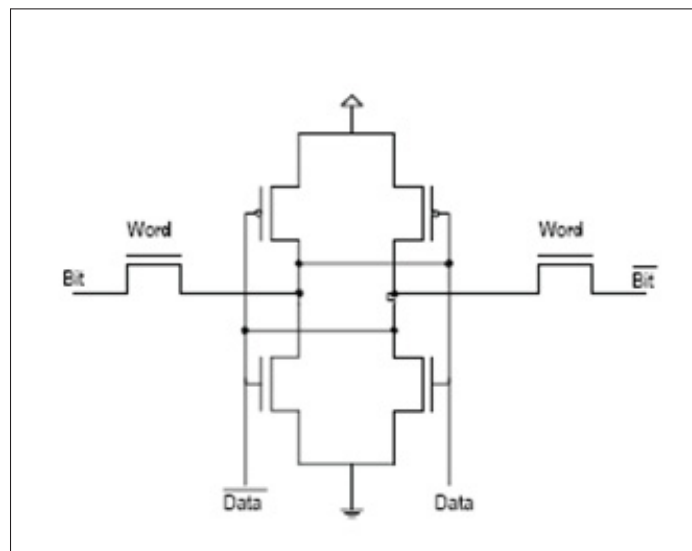


Figure 1.6 Cellule SRAM
Tirée de Ian *et al.* (2008)

Pour en simplifier la compréhension, le FPGA basé sur la SRAM peut être divisé en deux couches : une couche applicative (opérative) et une couche de configuration (Figure 1.7).

1.4.1 Couche applicative(opérative)

Utilisée pour réaliser les fonctions logiques, la couche applicative est notamment composée de blocs logiques configurables, appelés CLB (Configurable Logic Block) dans la terminologie de Xilinx. Autour de ces blocs, on trouve le réseau d'interconnexions programmables et les blocs d'entrée/sortie, appelés IOB (input output Block) selon la même terminologie.

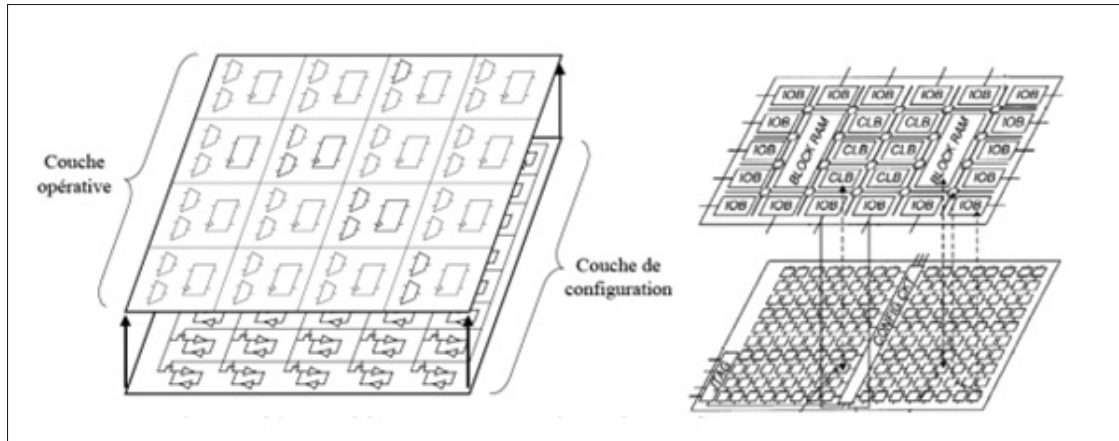


Figure 1.7 Partitionnement d'un FPGA en deux couches
Tirée de Bocquillon (2009)

- **CLB** : principalement composé de LUT (Look-Up Table) à K entrées, représentant la plus petite entité configurable du FPGA permettant la mise en œuvre des fonctions logiques simples (Figure 1.8) .

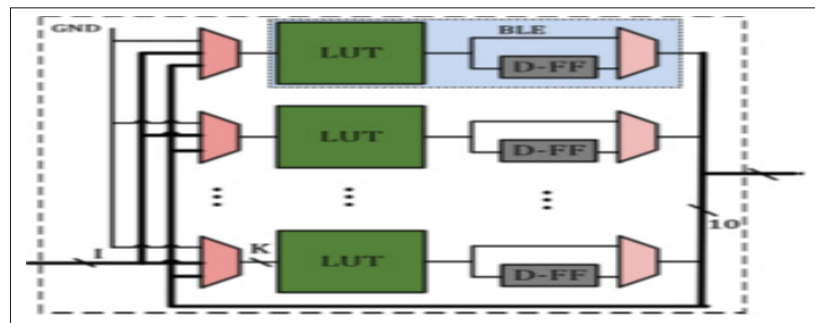


Figure 1.8 La structure d'un bloc logique configurable (CLB)
Tirée de Seifoori *et al.* (2018)

Les LUT, dans les séries 7 des FPGA Xilinx et particulièrement l'ARTIX-7, peuvent être configurés soit comme une fonction de 6 entrées avec une seule sortie, soit comme deux fonctions de 5 entrées avec des sorties séparées mais des adresses ou des entrées logiques communes. En plus des LUT, les CLB contiennent des multiplexeurs, des bascules, des portes logiques et des portes logiques dédiées aux opérations arithmétiques ;

- **réseau d'interconnexions programmables** : présente la partie intelligente de FPGA, il permet la connexion des blocs logiques les uns aux autres. Il constitue une grande partie de son espace et il permet de programmer à volonté le routage des blocs logiques ;
- **IOB** : constitue l'interface entre les broches du FPGA et ses ressources internes. Un bloc d'entrée/sortie peut être configuré en tant qu'entrée, sortie ou entrée et sortie, selon le standard d'entrée/sortie choisi, avec les niveaux logiques et les tensions d'alimentation correspondants.

1.4.2 Couche de configuration

Cette couche est un ensemble de millions de blocs mémoire SRAM configurables de différentes manières. L'architecture de la cellule mémoire (Figure 1.9) associée à la configuration est composée de deux inverseurs sous forme de têtes bœches et d'un transistor permettant d'accéder aux inverseurs pour des opérations de lecture ou d'écriture (Bocquillon, 2009).

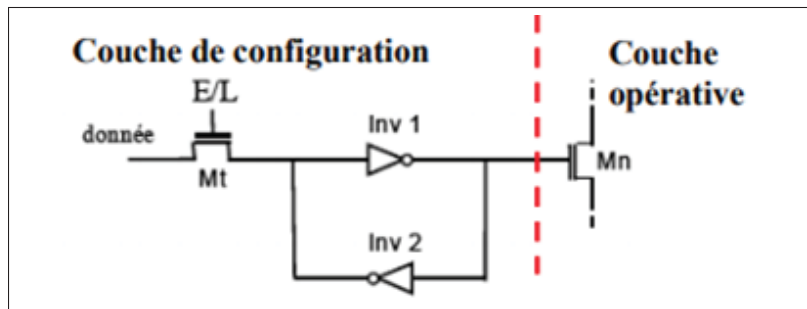


Figure 1.9 Architecture de la cellule SRAM de Bocquillon (2009)

1.5 Configuration du FPGA ARTIX-7

Le processus de configuration du FPGA basé sur la mémoire SRAM est commandé par les informations contenues dans le flux binaire générées par les outils du FPGA. Ce flux est transmis à la mémoire de la carte à travers une interface de configuration.

Comme nous l’avons déjà mentionné dans nos expérimentations, nous nous concentrons sur le FPGA ARTIX-7 qui dispose d’une mémoire de configuration divisée en trames de taille constante. Chaque trame est segmentée en 101 mots de 32 bits, de sorte qu’au total, chaque trame contient 3232 bits regroupés en lignes, colonnes et sous-colonnes. Cette décomposition en trames adressables de la mémoire facilite la configuration partielle des puces.

Selon la documentation de Xilinx (Xilinx, 2018b), les trames sont les plus petites parties adressables de la mémoire de configuration. Par conséquent, si l’utilisateur veut effectuer un changement de bit dans le flux binaire, il doit manipuler la trame complète.

Les FPGA Xilinx peuvent être programmés d’une façon externe via les interfaces comme JTAG ou SelectMap, ainsi que d’une façon interne par l’intermédiaire du port ICAP (internal configuration access port) permettant un accès en lecture et écriture à la mémoire de configuration.

L’outil d’injection de pannes le SEM *Controller* adapté par nos expérimentations utilise l’interface ICAP pour simuler les erreurs douces, plus des détails dans le chapitre 3.

1.6 Conclusion

Ce chapitre avait pour objectif de présenter, d’une part, une introduction sur les radiations cosmiques, leurs sources ainsi que leurs effets sur les circuits électroniques, et d’autre part, une description de l’architecture des FPGA à base de la technologie de mémoire SRAM.

Maintenant que nous avons vu comment les radiations agissent sur l’électronique, il faut savoir comment peut-on émuler ces effets ? Nous devons donc étudier les techniques ainsi que les procédures d’injection de pannes qui existent.

CHAPITRE 2

ÉTAT DE L'ART

2.1 Introduction

Le présent chapitre a pour objectif d'expliquer les différentes techniques d'injection de pannes à savoir celles effectuées dans des circuits physiques, par simulation et par émulation afin d'évaluer les techniques existantes et justifier celle qu'on adopte dans notre projet. De son côté, la littérature est mise à profit afin de distinguer les méthodes d'injection qui permettent à la fois une optimisation de la procédure d'injection de pannes en termes de vitesse et de représentativité. Enfin, la plateforme d'encapsulation permettant la génération des signatures afin de détecter le comportement fautif de circuit est présentée.

2.2 Techniques d'injection de pannes

Dans le contexte de ce projet, une panne est définie comme étant l'abstraction, au niveau logique, de l'effet des radiations cosmiques sur les valeurs logiques des signaux en général, et sur celles des bits de mémoire de configuration et des tables de conversion en particulier. Dans la littérature, on distingue plusieurs techniques d'injection de pannes pour évaluer les effets du rayonnement cosmique au niveau du circuit physique et ainsi établir la réaction des composants face aux pannes injectées.

Selon (Vanhouwaert, 2008) les différents critères qui peuvent être utilisés pour évaluer une technique d'injection de pannes sont : le degré d'intrusion, le contrôle des sites d'injection, la gestion de la durée d'injection, la reproductivité (qui présente la capacité à produire des résultats équivalents en respectant la configuration donnée), l'accessibilité (définie par la faculté de parvenir aux points d'injection désirés), et la flexibilité (qui représente la facilité de modifier le circuit ciblé).

Les techniques d'injection peuvent être classées comme suit : injection de pannes dans des circuits physiques, injection par simulation et injection par émulation.

2.2.1 Injection de pannes dans des circuits physiques

Il y a diverses approches utilisant ce type d'injection où les pannes sont injectées en se servant de composants physiques, en changeant la tension ou en bombardant avec des ions, etc. Parmi ces approches on peut citer :

- **injection de pannes au niveau des broches d'entrée/sortie** : cette méthode consiste à introduire des pannes au niveau des broches d'entrées/sorties par l'intermédiaire des sondes qui forcent le niveau logique bas ou haut. L'outil général d'injection de pannes MESSA-LINE (Arlat *et al.*, 1989) est reconnu en tant que premier outil qui a été développé pour faciliter l'injection. Cet outil est amélioré dans une autre version AFIT (Martínez *et al.*, 1999);
- **injection de pannes par corruption de la mémoire** : cette approche se traduit par l'introduction des pannes dans la mémoire d'un processeur, en perturbant par exemple les registres mémoires afin de tester des mécanismes de détection d'erreurs. Le système commercial DEFI (Michel *et al.*, 1994) est un modèle d'outil qui permet ce type d'injection de pannes pouvant se produire d'une manière matérielle en utilisant les sondes ou bien au niveau logiciel par l'insertion des codes ;
- **injection de pannes par laser** : cette méthode nécessite un spot laser très fin permettant l'injection de pannes transitoires (i.e. qui disparaissent après un certain temps). Le laser génère des paires électrons/trous lorsqu'il traverse le substrat, ce qui produira un pic de tension si les paires d'électrons/trous sont à proximité d'une jonction PN. Ce type des tests sous faisceaux laser reste limité au niveau d'énergies fournies (Lewis *et al.*, 2001);
- **injection de pannes par perturbation de l'alimentation** : ce type d'injection implique plusieurs méthodologies. Parmi lesquelles on peut citer l'exemple d'injection proposée dans (Karlsson *et al.*, 1991) qui repose sur l'ajout d'un transistor dans le circuit entre l'alimenta-

tion et le nœud Vcc dans la puce. En variant la tension de la grille de transistor, l'alimentation électrique peut être perturbée ;

- **injection de pannes via accélérateurs de particules** : ce type d'injection se base sur l'utilisation de particules accélérées, telles les ions lourds, les protons et les neutrons. Il s'agit du type d'injection qui reproduit le plus fidèlement l'effet des radiations. Il a également l'avantage d'accélérer l'exposition aux radiations par plusieurs ordres de grandeurs. Ce type d'injection est habituellement utilisé pour estimer la robustesse de circuit dans des situations critiques. Dans le contexte de notre projet, les résultats des tests de faisceaux de neutrons effectués précédemment au laboratoire TRIUMF ont constitué un moyen pour valider les approches proposées.

Ces technologies d'injection présentent plusieurs inconvénients. En effet, certaines peuvent être destructives. De plus, leur contrôlabilité, accessibilité et reproductibilité sont réduites. Finalement, elles nécessitent du matériel pour réaliser une campagne d'injection, ce qui peut être restrictif.

2.2.2 Injection de pannes par simulation

Cette technique correspond à introduire les pannes à plusieurs niveaux d'abstraction dans le design, et cela a pour effet de permettre la vérification du circuit à différents stades du flot de conception. Selon le niveau choisi, on peut distinguer trois approches d'injection par simulation.

- 1) **au niveau système** : pour la réalisation de cette technique, divers langages tels que SystemC ou POLIS sont adoptés dans la conception matérielle/logicielle afin de décrire les circuits au niveau système (Souari, 2016). Cette méthode est principalement dédiée aux System-on-Chip (SoC). Son avantage est la rapidité de simulation et son inconvénient est le faible niveau de précision dans la détermination de la valeur de sensibilité des systèmes concernés ;

- 2) **au niveau RTL** : à ce niveau, selon (Berrojo *et al.*, 2002), il est possible d'injecter des pannes de deux manières : la première consiste à renforcer les valeurs de certains nœuds internes à l'aide des commandes fournies par l'outil de simulation, et la seconde consiste à modifier la description RTL. Un exemple de la première façon est l'utilisation de l'outil MEFISTO (Multilevel Error/Fault Injection Simulation Tool) (Jenn *et al.*, 1995), qui manipule les signaux et les variables pour injecter des pannes ; par exemple, la commande wait du VHDL peut être utilisée pour arrêter la simulation pendant un cycle puis forcer la valeur du signal désirée pendant un temps bien précis. On peut mentionner comme exemple l'outil développé par l'Agence Spatiale Européenne par utilisation de la commande « force » de ModelSim, les langages TCL (Tool Command Language) et Perl. Un exemple de la seconde façon est l'utilisation de l'outil MEFISTO-L (Boué *et al.*, 1997), développé par le laboratoire LAAS, qui permet l'injection de pannes tout en modifiant la description RTL en utilisant les saboteurs, mutants et sondes afin de modifier les signaux et contrôler le design. L'avantage présenté d'injection à ce niveau est d'estimer tôt la vulnérabilité dans le processus de la conception et son principal inconvénient est que la couverture des pannes ne correspond généralement pas à la couverture des pannes des bas niveaux logiques (Thaker *et al.*, 2000) ;

- 3) **au niveau des portes logiques** : plusieurs outils ont été développés pour introduire les pannes dans les portes logiques. Le premier outil apparu est FAST (Fault Simulator for Transients) développé par (Cha *et al.*, 1996). Il se distingue des autres outils par l'utilisation des modèles réalistes de pannes et il se décompose en deux simulateurs : un simulateur de pannes de timing pour définir la durée de l'impulsion et un simulateur de pannes parallèles pour examiner les conséquences de la panne sur les sorties. Le deuxième outil, VERIFY (VHDL-based Evaluation of Reliability by Injecting Faults efficiently), développé par (Sieh *et al.*, 1998), supporte l'injection de pannes dans les portes logiques et de la description RTL. Il permet aussi l'inversion des bits et les collages à '1' ou '0'. On peut aussi citer d'autres outils tel que ROBAN (Alexandrescu *et al.*, 2004) permettant l'injection rapide des pannes transitoires, et LIFTING, un outil gratuit permettant la simulation de pannes ainsi que l'analyse de ses résultats.

Dans la modélisation des pannes, la simulation des pannes au niveau de la porte logique est plus fiable et précise qu’au niveau RTL. Par contre, elle est lente lorsqu’on l’applique aux grands circuits modernes.

2.2.3 Injection de pannes par émulation

Cette technique a été conçue afin de minimiser le temps d’injection par rapport à l’injection par simulation. Cela offre au concepteur la particularité d’étudier le fonctionnement du circuit en fonction des interactions temps réel. Avec la technique d’émulation, on peut injecter des pannes d’une façon plus rapide que celle par simulation en recourant au prototypage grâce l’apparition des circuits reconfigurables comme les FPGAs basés sur la SRAM, qui constituent un choix attrayant pour les industries spatiale et aéronautique. Leurs caractéristiques telles que la flexibilité et la reconfigurabilité sur site, ainsi que leur faible coût rendent leur utilisation croissante pour les applications critiques. D’après (Khatri *et al.*, 2020) on distingue essentiellement les injections de pannes par instrumentation et celles par reconfiguration.

- **injection de pannes par instrumentation** : son principe est d’ajouter au circuit original des composants tels de la logique combinatoire et des registres à décalage. Avec cette approche, la communication entre le concepteur et le FPGA a besoin de transférer peu de données pour optimiser les performances. On peut citer par exemple l’instrumentation des bascules ou la méthode SCAN (Bushnell & Agrawal, 2004). Cette méthode se distingue par l’ajout d’un multiplexeur à l’entrée de chacune des bascules (Fig.2.1), ces dernières étant liées afin de former un registre de décalage ce qui facilite l’injection de pannes et l’observation de leurs effets. L’outil FIFA (Fault Injection by means of FPGA) (Civera *et al.*, 2001) utilise ce type d’instrumentation. À partir de la liste d’interconnexions du circuit, il génère un rapport de pannes et il comporte notamment des modules pour surveiller et analyser l’injection de pannes. Cette méthodologie offre une bonne contrôlabilité mais le nombre des entrées/sorties reste limité par la taille de FPGA. De plus, un délai est rajouté sur les lignes suite à l’ajout des instruments.

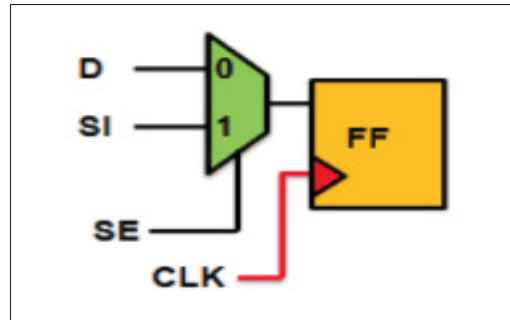


Figure 2.1 Bascule instrumentée pour la méthode Scan chain
Tirée de Savanoor & Guruprasad (2020)

- **injection de pannes par reconfiguration** : cette méthode correspond à introduire les pannes par modification du fichier de configuration (*bitstream*). Ce dernier contient les informations pour la configuration du FPGA. Cette méthode exige la reconfiguration du circuit à chaque série d'injection de panne, cette reconfiguration peut se faire d'une façon statique (Compile Time Reconfiguration ou CTR) ou dynamique (Real Time Reconfiguration ou RTR).

La configuration statique consiste à compiler et synthétiser le design pour appliquer des changements produits au code afin de reconfigurer le FPGA et régénérer le système complet. La configuration dynamique est plus rapide que la statique. Elle permet de modifier en temps réel la configuration du FPGA pendant l'exécution de l'application. Cette reconfiguration peut être d'une façon partielle (Local RTR) ou entière (Global RTR). Actuellement, les méthodes d'injection de pannes basées sur la reconfiguration partielle dynamique (DPR) des FPGA à base de SRAM ont beaucoup attiré l'attention vu qu'elles permettent le changement des portions du circuit sans avoir changé le reste du design. La reconfiguration dynamique permet aussi l'injection de pannes par la méthode de collage "1" ou "0" au sein des blocs logiques du circuit. En outre, il est aussi possible à l'aide du module *SEM Controller* d'introduire des SEU (détaillé au chapitre 3), ce qui entraîne une modification de l'état des éléments de stockage tels que les cellules mémoires. Cette méthode d'injection est jusqu'à 30 fois plus rapide que l'injection de pannes par simulation.

Parmi toutes les techniques d'injection mentionnées précédemment, nous sommes intéressés dans notre projet à celle par émulation. En effet, celle par instrumentation a été utilisée au

niveau de la plateforme d'encapsulation de génération des signatures (décrite plus loin) pour distinguer le comportement fautif et celle par configuration a été adaptée dans notre projet en émulant le design suite à l'injection des SEU induites par les particules de rayonnement.

2.3 Optimisation de la procédure d'injection de pannes

Plusieurs scénarios d'optimisation et d'amélioration sont proposés dans la littérature. Les objectifs habituels des procédures d'injection de pannes sont : 1) la minimisation du temps d'émulation, et 2) la reproduction la plus fidèle possible des résultats obtenus sous les tests accélérés sous faisceaux de particules.

Afin de minimiser le temps d'injection, une approche proposée par (Entrena *et al.*, 2011) pendant l'exécution de l'injection, est basée sur la réduction de communication dans la structure de système d'émulation avec le PC hôte. Cette réduction se fait en insérant un contrôleur qui permet d'optimiser le processus d'injection ainsi de classer la liste des pannes générées selon leurs effets. Un gain important du temps d'injection a été assuré par l'arrêt de processus de test chaque fois que l'injection et la classification de panne sont faites.

Dans la même optique d'accélération de la vitesse d'injection, plusieurs autres méthodes ont été proposées comme par exemple dans (Ziade *et al.*, 2011), où l'idée est de diviser le DUT en sous-ensembles fonctionnels et d'injecter les pannes dans chaque ensemble tout en observant et sauvegardant le comportement fautif détecté dans chaque ensemble puis dans le système au complet. Avec cette proposition, la vitesse d'injection de pannes est sensiblement augmentée et le temps nécessaire à la réalisation de l'expérimentation est raccourci. En effet, avec cette méthode, il est possible d'injecter simultanément plusieurs pannes dans les bits de configuration des FPGA à base de SRAM lors de sa reconfiguration avec le flux binaire (*bitstream*) défectueux, alors que d'autres techniques étaient capables d'insérer une seule panne à chaque reconfiguration.

Dans (Faure *et al.*, 2005), une méthode basée sur la loi de poisson a été proposée afin de préciser les instants d'injection de SEU. Les auteurs ont utilisé le processeur SPARC comme DUT avec

un modèle basé sur le processus de Bernoulli, appelé Particle Domain Fault Injection, afin de calculer le temps d'injection représentatif d'un bombardement réel.

En vue de maximiser le nombre d'erreurs causées par les pannes injectées dans la mémoire de configuration et minimiser ainsi le temps d'émulation, l'auteur (Souari *et al.*, 2016) a proposé deux approches : 1) une approche d'injection de pannes qui priorise un sous-ensemble spécifique de bits de configuration, dont les bits de LUT. Cette approche est appliquée pour évaluer la criticité de différentes ressources FPGA et l'impact global de l'inversion des bits critiques. Une réduction significative au niveau de l'estimation du nombre de bits critiques était également signalée tout en atteignant un facteur d'accélération de 108 fois comparativement avec l'injection de pannes aléatoire. 2) une approche d'injection de pannes en considérant la différence de sensibilité relative entre les bits de configuration réglés à "0" et à "1" pour reproduire les résultats des tests accélérés aussi précis que possible.

Nos travaux sont au départ orientés vers la reproduction la plus représentative des expérimentations en bombardement en exploitant davantage le concept de la sensibilité relative. De plus, nous proposons une méthode permettant d'accélérer l'émulation tout en conservant cette représentativité.

2.4 Signatures

2.4.1 Le concept de signature

Une signature est une représentation qui reflète le comportement défectueux d'une conception à plusieurs niveaux d'abstraction à travers les techniques d'injection de pannes. L'objectif de l'utilisation des signatures est de comprimer les résultats collectés à partir des rapports d'injection de pannes générés par un outil d'injection de pannes et de réutiliser les informations à un niveau d'abstraction plus élevé. Une réduction du nombre de paramètres dans les signatures est visée autant que possible pour minimiser leur taille et complexité (Robache *et al.*, 2013).

Les signatures peuvent avoir différentes interprétations et formats :

- **signatures arithmétiques** : présentent une distribution des erreurs arithmétiques modélisant la différence entre la sortie parfaite d'un circuit de référence et celle d'un circuit défectueux (Hobeika *et al.*, 2014).

$$Erreurs\ arithmetiques = Result_{golden} - Result_{faulty} \quad (2.1)$$

Ce type de signatures peut être facilement utilisé à des niveaux d'abstraction plus élevés tels que Matlab/Simulink, etc ;

- **signatures logiques** : présentent une distribution d'erreurs binaires modélisant la différence en termes de bits (XOR) entre la sortie d'un circuit de référence et d'un circuit défectueux (Hobeika *et al.*, 2014). Ce type de signature préserve l'information au niveau du bit, comme la position du bit en erreur ainsi que le nombre de bits inversés. Elle se calcule comme suit :

$$Binary\ errors = Result_{golden} XOR Result_{faulty} \quad (2.2)$$

Ce type de signatures peut être plus représentatif à un niveau d'abstraction inférieur.

- **signatures brutes** : présentent une réponse directe de montage en fonction des vecteurs d'entrées. Lorsqu'une panne est détectée dans le circuit lors de l'inversion de la valeur logique d'un bit de la mémoire de configuration, une séquence de données est générée.

Pour ce projet, nous utilisons les signatures brutes car c'est ce type de signatures qui est fourni par la plateforme matérielle utilisée, décrite ci-après. Notons qu'un post-traitement sera effectué sur les données collectées pour les analyser.

2.4.2 Plateforme de génération de signatures

Les deux schémas ci-dessous présentent la plateforme utilisée afin de générer les signatures pour les circuits, tant au laboratoire TRIUMF que pendant les émulations. Cette plateforme a été initialement conçue par Florian Jacquet dans le cadre de son projet de maîtrise (JACQUET, 2017) et par la suite améliorée par Simon Pichette, le tout sous la supervision du professeur

Claude Thibeault. Le premier schéma (Fig.2.2) montre la partie DUT (Device Under Test) de la plateforme design(celle dans laquelle on retrouve les circuits sous test, dont le nombre varie selon le circuit utilisé) et met en évidence les signaux qui doivent être partagés avec la partie référence de la plateforme (contenant, entre autres, le circuit de référence), partie qui est présentée dans le deuxième schéma (Fig.2.3).

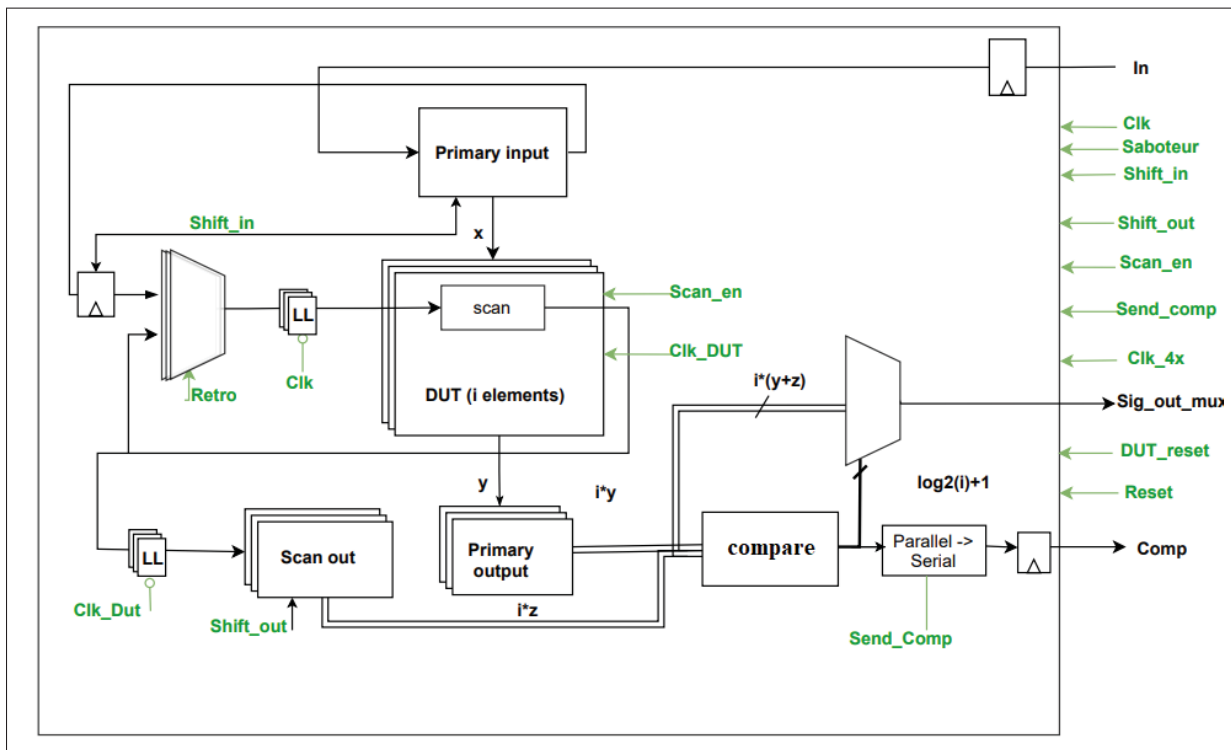


Figure 2.2 Plateforme de génération : partie DUT

Globalement, la plateforme fonctionne de la manière suivante :

Au départ, elle est placée dans un premier mode, appelé mode de détection. C'est dans ce mode que la plateforme détecte la présence d'erreurs causées par l'injection de pannes. Dans ce mode, la plateforme utilise des vecteurs de type ATPG (Automated Test Pattern Generation) faisant levier sur la présence de chaînes de balayage (scan chain) insérées dans chacun des DUT. Ces chaînes de balayage sont activées lorsque les DUT sont en mode test en formant un seul registre dans chaque DUT (représenté par le bloc scan, Fig.2.2) à partir de ses bascules. Ces vecteurs sont emmagasinés dans la partie référence de la plateforme et sont acheminés de manière sérielle aux DUT via l'entrée « In » (Fig.2.2) . Ces vecteurs contiennent le contenu de la chaîne de balayage ainsi que les valeurs à fournir aux entrées de chaque DUT. La procédure suivante est suivie en mode détection :

1. Les DUT sont placés en mode test ;
2. Les vecteurs ATPG sont chargés dans les DUT ;
3. Les DUT sont sorti du mode test et placés dans le mode fonctionnel. Ils sont activés pour un coup d'horloge ;
4. Les DUT sont replacés en mode test ; le contenu de la chaîne de balayage de chaque DUT est récupéré dans un registre externe (représenté par le bloc « scan out », Fig.2.2), et les sorties de chaque DUT sont aussi récupérés dans un registre externe (représenté par le bloc « primary output », Fig.2.2) ;
5. Les DUT sont organisés en groupe de 3 et la réponse (i.e. sorties et contenu de la chaîne) d'un DUT est comparé à la réponse des deux autres au même groupe. Si une différence est notée dans la réponse d'un DUT, celui-ci est étiqueté défectueux. On considère alors qu'une erreur est détectée et on passe en mode génération ;
6. Si aucune erreur n'est détectée, on retourne à l'étape 2.

En mode génération, on utilise un autre type de vecteurs, à savoir des vecteurs pseudo-aléatoires émulant le mode fonctionnel des DUT. Ces vecteurs sont également fournis par la partie référence de la plateforme et sont aussi acheminés de manière sérielle aux DUT via l'entrée « In ». Dans ce cas, les vecteurs ne contiennent que les valeurs à fournir aux entrées de chaque DUT,

le contenu de la chaîne de balayage étant réinjecté dans les bascules pour conserver l'état du circuit. L'utilisation d'un autre vecteur permet d'identifier les erreurs détectées par les vecteurs ATPG mais manquées lorsque les DUT sont en mode fonctionnel. En mode génération, 2047 vecteurs pseudo-aléatoires sont fournis aux DUT et les sorties du circuit étiqueté comme défectueux ainsi que son numéro et sa réponse au dernier vecteur ATPG appliqué sont emmagasinés dans des blocs de mémoire avant d'être téléversés vers l'ordinateur hôte.

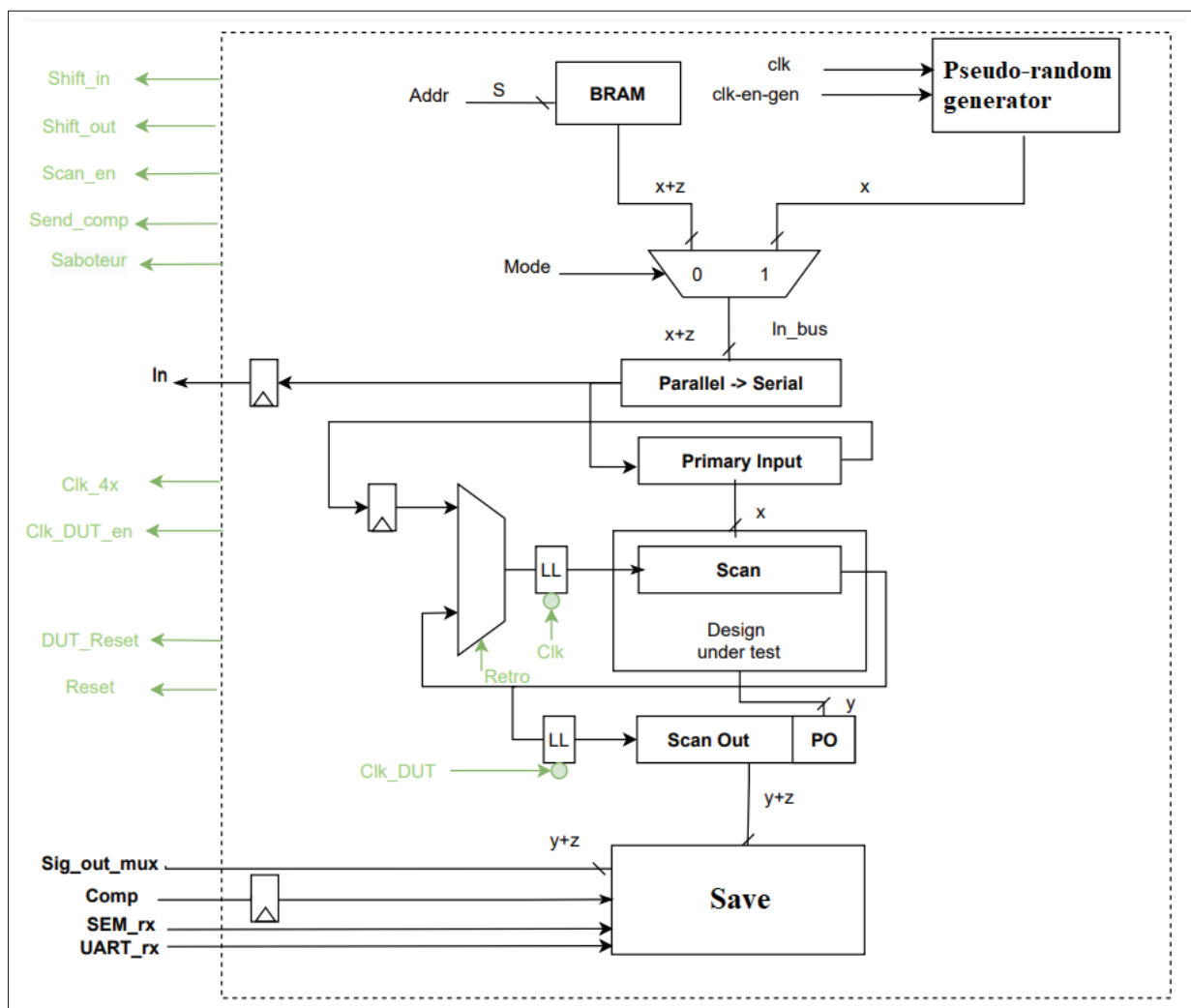


Figure 2.3 Plateforme de génération : partie de référence

La figure 2.3 illustre la partie de référence de la plateforme de génération des signatures. Comme mentionné précédemment, cette partie fournit les vecteurs aux DUT. Les vecteurs

de type ATPG sont emmagasinés dans le bloc étiqueté « BRAM » alors qu'un générateur (bloc « pseudo-random generator ») fournit les vecteurs pseudo-aléatoires. Cette partie de la plateforme contient également une copie du DUT ainsi qu'un bloc (« Save ») responsable de récupérer les informations provenant du DUT étiqueté défectueux, de les emmagasiner et les transmettre à l'ordinateur hôte. Originellement, les deux parties de la plateforme étaient sur deux cartes séparées. Cependant, dû à des problèmes de communication, les deux parties ont été implémentées sur une seule carte.

2.4.3 Classification des signatures

La fusion des deux parties de la plateforme ainsi que la présence de certains blocs de la partie DUT tel les blocs scan qui contiennent à leur tour les chaînes de balayage font en sorte que des fausses détections soient rapportées par la plateforme. Nous appelons les signatures résultantes des fausses signatures.

Une fausse détection est rapportée s'il n'y a pas de différence entre les valeurs des registres à balayage et les valeurs des sorties primaires pour les deux types de vecteurs. Un exemple de fausse signature pourrait survenir selon le scénario suivant : si un *bitflip* attaque le bloc de compare alors qu'il n'y a pas de différence dans les sorties. Une signature sera donc considérée comme bonne s'il y a au moins une différence entre les valeurs des registres à balayage et les valeurs des sorties primaires. Une bonne signature sera considérée comme manquée par les vecteurs pseudo-aléatoires lorsque les sorties primaires du DUT étiqueté comme défectueux et celle du circuit de référence sont identiques dans le mode fonctionnel. Dans ce mode, en effet, le contenu des chaînes de balayage ne sont accessibles (sauf si elles sont directement connectées à une sortie primaire). Seules les sorties primaires le sont.

2.5 Conclusion

Dans ce chapitre, nous avons présenté les techniques d'injection de pannes, qui peuvent être effectuées d'une façon matérielle ou logicielle. Ensuite, nous avons abordé les recherches pré-

cédemment effectuées en littérature pour l'optimisation de la procédure d'injection de pannes. Enfin, le concept de génération des signatures a été évoqué, comme méthode d'évaluation de la sensibilité des circuits électroniques face aux radiations.

CHAPITRE 3

MÉTHODOLOGIE PROPOSÉE

3.1 Introduction

Ce chapitre porte sur la méthodologie proposée afin d’émuler le comportement fautif des Circuits de référence (benchmarks) sous l’effet de rayonnement cosmique et dont l’objectif principal est de reproduire le plus fidèlement possible les résultats obtenus au laboratoire TRIUMF. Cette méthodologie comprend des séries d’expérimentations d’injection de pannes ciblant des parties précises, telles que les bits se trouvant dans les tables de conversion (appelés aussi bits de LUT), ce qui requiert l’identification de leurs adresses exactes dans la mémoire de configuration.

Tout d’abord, nous introduisons les bancs d’essais utilisés dans nos expérimentations. Ensuite, nous présentons l’outil d’injection de pannes par émulation, le *SEM Controller* de Xilinx, pour injecter des *bitflips* dans la mémoire de configuration. En effet, cet outil est simple, non coûteux et offre un environnement capable d’émuler les effets de rayons cosmiques qui peuvent affecter les circuits FPGA de cette compagnie. Par la suite, nous abordons une description détaillée de la procédure d’identification de quatre types bits de configuration, à savoir les bits de LUT essentiels, les bits de LUT non essentiels, ainsi que les bits hors LUT (et hors BRAM) essentiels et les bits hors LUT non essentiels. Enfin, une description de la procédure d’injection de pannes adoptée dans ce mémoire est présentée.

3.2 Circuits de référence utilisés

Divers circuits de référence ou repères (benchmarks) ont été utilisés au fil des années pour comparer différents systèmes. Ces repères peuvent être utiles aux chercheurs qui souhaitent déterminer comment les changements apportés à la technologie, à l’architecture ou au compilateur affectent les performances du système (Quinn *et al.*, 2015).

Des nombreux circuits sont réalistes par rapport à la façon dont les FPGA sont utilisés dans les environnements informatiques embarqués. En raison de toutes les qualités décrites, le jeu de référence (benchmark ITC'99) est idéal pour nos besoins de test. Ce jeu de référence est bien établi et populaire particulièrement auprès de la communauté scientifique s'intéressant au test des circuits intégrés. Cette référence répond à toutes nos exigences, y compris une variété d'algorithmes réalistes, des entrées définies, l'extensibilité et la portabilité. Plus spécifiquement, les circuits choisis pour ce projet seront adaptés pour permettre l'application des vecteurs de tests automatiquement générés et basés sur l'insertion de chaînes de registres à balayage (aussi appelés vecteurs de type ATPG), et insérés dans la plateforme d'encapsulation décrite dans le chapitre précédent.

Rappelons qu'un des objectifs derrière le développement de cette plateforme est de permettre de mesurer la proportion des pannes qui sont détectées par les vecteurs de type ATPG mais qui sont manquées quand les modules roulent en mode dit fonctionnel. Nous nous sommes concentrés sur les circuits B01, B05 et B12 dans nos expérimentations pour comparer les résultats obtenus avec ceux qui sont observés au laboratoire TRIUMF dans un environnement de rayonnement.

3.3 Outil d'injection de pannes : SEM *Controller*

Le contrôleur SEM développé par Xilinx pour la série 7 est un dispositif à configuration automatique (Xilinx, 2018a) utilisé comme solution prévérifiée pour :

- détecter les erreurs dans la mémoire de configuration des FPGA série 7 ou Ultrascale, et les corriger si besoin. Notons que les options de détection et de correction ne sont pas utilisées dans ce projet ;
- effectuer l'émulation des SEU en injectant des pannes au niveau de la mémoire de configuration. Dans ce projet, nous utilisons l'émulation, aussi appelé mode d'injection de pannes, pour investiguer la réponse du système face aux SEU. Notons que la fonction d'injection de

pannes peut également servir à évaluer et tester les capacités de mitigation des SEU d'un système. Cette dernière fonctionnalité n'est pas utilisée dans ce projet.

Dans nos expérimentations nous sommes donc intéressés au mode d'injection de pannes. Le contrôleur SEM dispose de six interfaces d'entrées-sorties (voir ANNEXE I). Les interfaces qui sont importantes pour notre procédure d'injection sont : 1) l'interface de statut pour contrôler des diodes électroluminescentes (DEL), et 2), les interfaces fournies par le fabricant pour se connecter aux primitives ICAP et FRAME_ECC. Il faut alors connecter ces interfaces correctement selon les critères décrits en documentation technique.

L'interface d'injection de pannes est sans intérêt dans la mesure où cette injection est plutôt contrôlée par l'utilisateur via l'UART de son ordinateur. Ce dernier assure la correspondance avec le module SEM moyennant l'envoi de caractères spécifiques. À titre d'exemple, dans notre procédure d'injection qui sera explicitement détaillée au présent chapitre, l'affichage des signatures générées se produit sur l'*Hyperterminal* en se servant de l'émulateur de terminal *PuTTY* à travers une interface UART moyennant une connexion série RS232 pour transmettre le caractère 's' dans une session dans le but de déclencher l'étape de surveillance BRAM. Le port USB-UART est notamment utilisé pour connecter la carte à l'ordinateur gérant le processus d'injection. Ce port, agissant comme émetteur/récepteur, intègre un double PMOD (peripheral module interface) analogique/numérique permettant la correspondance entre le cœur IP du SEM et l'ordinateur. Le contrôle de la macro se fait à partir des pins du PMOD pour diriger l'émission et la réception des signaux.

La figure 3.1 montre le schéma de communication de l'environnement de travail utilisé dans ce projet : la programmation de FPGA par un fichier « .bit » est assurée par l'outil *Vivado* via l'interface du Hardware Manager à partir d'une connexion JTAG.

L'injection de pannes est assurée par un autre outil appelé *injector*, développé par Simon Pichette. À ce stade, il est obligatoire d'insérer la liste des adresses des bits de configuration à attaquer, qui vont être flippés selon l'approche choisie, sous forme d'un fichier. Par la suite, le SEM *Controller* est mis en mode d'injection cumulative sans appliquer aucune correction

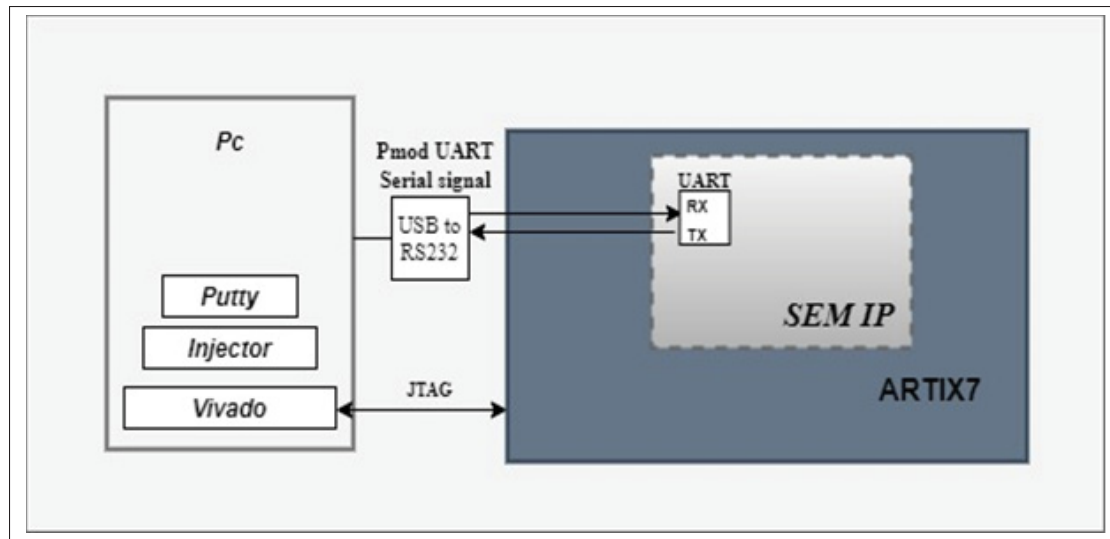


Figure 3.1 Schéma global de communication

jusqu'à la détection d'une panne, et on passe par la suite à l'étape de collecte des signatures générées. Notons que la plateforme entière d'injection de pannes a été développée avant le début de ce projet et a été utilisée sans modifications.

Généralement 20 à 40% de la mémoire de configuration présente un aspect critique pour le fonctionnement du FPGA (BEDI, 2014). Le partitionnement de la mémoire de configuration en bits essentiels et non essentiels améliore la précision et la fiabilité de la conception.

La figure 3.2 illustre la relation entre les bits de configuration, les bits essentiels, les bits essentiels priorisés et les bits critiques.

- **bits de configuration** : contenu d'un élément de mémoire de configuration ;
- **bits essentiels** : bits associés aux circuits de la conception et constituent un sous-ensemble des bits de configuration du dispositif par exemple les bits des LUT peuvent être essentiels ;
- **bits essentiels priorisés** : constituent un sous-ensemble des bits essentiels, bits associés aux parties de design que l'utilisateur définit au *SEM Controller* pour qu'il commence par ces parties lorsqu'il est en mode détection, tel qu'on a mentionné, cette fonction n'est pas utilisée dans notre projet, donc ce type des bits ne nous intéresse pas ;

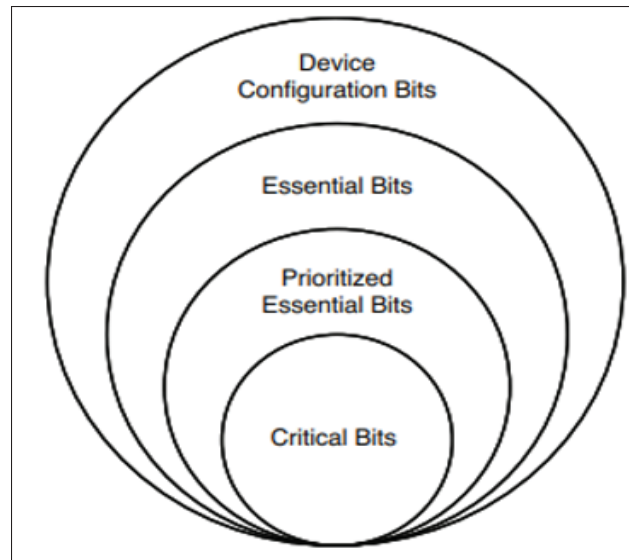


Figure 3.2 Relation hiérarchique de la configuration avec les bits essentiels
Tirée de BEDI (2014)

- **bits critiques** : bits de configuration qui causent une panne fonctionnelle s'ils subissent une inversion dans leur état logique.

Dans ce qui suit, nous allons détailler la partie d'identification des différents bits ciblés dans nos expérimentations d'émulation de SEU dans le FPGA.

3.4 Classification des bits de configuration selon leur niveau logique

Cette étape vise à créer deux listes, la première contient l'adresse des bits de configuration avec un « 0 » logique, l'autre contient celle des bits de configuration avec un « 1 » logique. La création de ces deux listes va permettre l'application des différents facteurs de sensibilité en fonction du contenu des bits de configuration, sachant par exemple que les bits contenant un « 1 » logique sont plus sensibles aux radiations que ceux contenant un « 0 » logique. Cette démarche est effectuée sur les trois montages comprenant les bancs d'essais (benchmarks) utilisés dans nos tests d'injection de pannes et durant les expérimentations au laboratoire TRIUMF.

La classification des bits est faite à partir des fichiers EBC de bancs d'essais et à l'aide d'un script codé en langage C. Ces fichiers regroupent le contenu des cellules de mémoire. L'adresse

d'un bit de configuration correspond au numéro de la trame, au numéro du mot dans cette trame, et à la position exacte du bit dans ce mot. Notons que pour le circuit FPGA utilisé, il y a un total de 61104192 bits de configuration répartis en 18906 trames de 101 mots de 32 bits.

La procédure de génération des deux listes est illustrée à la figure 3.3.

Pour générer le fichier EBC, on peut soit ajouter une contrainte dans le fichier XDC (*Xilinx Design Constraints*) ou utiliser des commandes Tcl. on a opté pour la première option. Notons que la contrainte indiquée ci-dessous permet également la génération de fichier EBD, qui contient les données de masque identifiant les bits essentiels du fichier EBC ; on aura besoin de ce fichier plus tard.

```
Set_property BITSTREAM.SEU.ESSENTIALBITS YES [current_design]
```

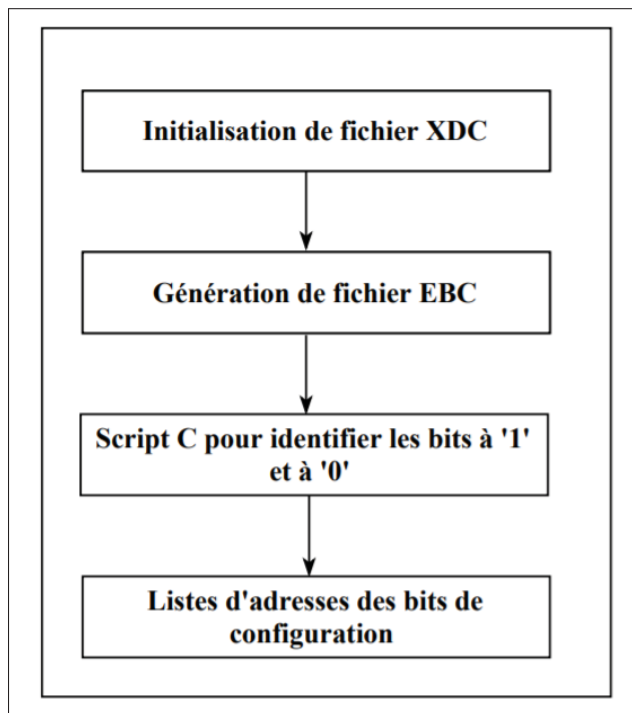


Figure 3.3 Procédure de génération des listes de bits de configuration à 0 et à 1

En appliquant cette procédure pour les trois circuits utilisés dans nos expérimentations nous avons obtenu les résultats présentés dans le tableau 3.1 suivant :

Tableau 3.1 Répartition des bits de configuration des trois circuits

Nombre des bits	B01	B05	B12
Bit à 0	58984260	59292536	58236954
Bit à 1	2116700	1808424	2864006

3.5 Identification des bits de LUT de l'ARTIX-7

Rappelons que le LUT est un tableau qui détermine la sortie d'une ou plusieurs entrées données. Il présente une entité configurable de FPGA qui permet de déterminer et implémenter des fonctions combinatoires pour les designs.

Un autre objectif de ce projet est de tenir compte de la sensibilité des bits de configuration en fonction de leur nature. Plus spécifiquement, nous distinguons les bits de configuration contenus dans les LUT (aussi appelé bits de LUT) des autres bits de configuration (aussi appelés bits des NON LUT, excluant les bits de BRAM). Nous devons donc classifier les bits de configuration en fonction de leur nature, ce qui revient à identifier les bits de LUT dans la configuration du circuit FPGA ciblé. Rappelons que pour l'ARTIX-7, chaque tranche (slice) contient quatre LUT (A, B, C et D) à six entrées indépendantes (entrées I – I0 à I5) et une sortie (O).

L'approche utilisée pour identifier les adresses de bits de LUT est la même que celle définie dans (Souari, 2016), sauf que nous l'avons adapté à notre environnement du travail. Cette approche est fondée principalement sur la création de deux versions d'un design spécifiquement conçu pour cette tâche, où le contenu des LUT est inversé d'une version à l'autre. La différence principale par rapport à (Souari, 2016) est que nous ciblons un autre FPGA, plus complexe, et que nous avons utilisé l'outil *Vivado* pour la synthèse et l'analyse des conceptions HDL qui prend en charge les FPGA serie-7, remplaçant Xilinx ISE avec des fonctionnalités supplémentaires, ce qui nous a forcé à modifier un peu l'approche.

Le design en question est formé d'une chaîne de LUT. Un exemple d'une telle chaîne à 4 LUT est montré en figure 3.4. La stratégie utilisée est d'empêcher l'optimisation combinatoire de la chaîne, de geler le placement des LUT, d'initialiser les LUT avec une première série de valeurs et de générer le fichier EBC correspondant. Ensuite, les valeurs des LUT sont inversées et un second fichier EBC est généré.

Comme le placement demeure le même, le routage demeure également le même, ce qui fait que les seuls bits de configuration qui changent sont ceux des LUT. En comparant les deux fichiers EBC, on peut alors les identifier.

La stratégie s'appuie sur l'utilisation de contraintes définies dans le fichier XDC, comme la contrainte "DONT_TOUCH", qui empêche l'optimisation. La contrainte peut être définie dans le fichier de contraintes XDC (Voir Figure-A V-1), avec la syntaxe suivante :

```
Set_property DONT_TOUCH true [get_cells-hier-filter REF_NAME= LUT6]
```

Plus précisément, les étapes à suivre pour de identifier les bits de LUT sont les suivantes :

- création de design qui regroupe tous les LUT du FPGA par attribution d'une adresse à chaque LUT avec la syntaxe décrite en ANNEXE (Voir Figure-A V-1);
- initialisation des valeurs de LUT (à tire d'exemple 64hAAAAAAAAAAAAAAAAAAAA);

```
Set_property INIT_A 64'hAAAAAAAAAAAAAAAAAAAA [get_cells LUT6]
```
- génération de fichier EBC;
- réinitialisation des valeurs de LUT par le complément de premier exemple;

```
Set_property INIT_A 64'h5555555555555555 [get_cells LUT6]
```
- génération de deuxième fichier EBC;
- comparaison de deux fichier EBC et extraction des adresses des LUT.

La comparaison entre les deux fichiers se fait à l'aide d'un script codé en langage C, qui sert à extraire les adresses exactes des bits de configuration où une différence entre les deux fichiers EBC générés apparaît.

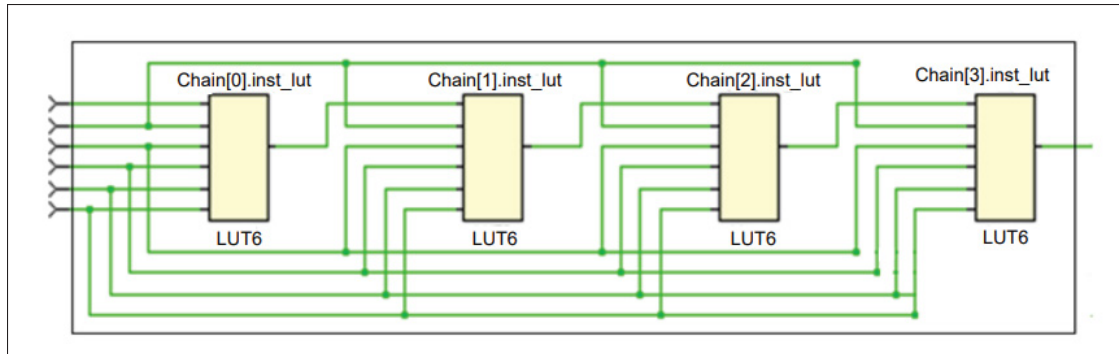


Figure 3.4 Connexion des LUT de design du FPGA ARTIX-7

Comme mentionné précédemment, nous utilisons une puce FPGA de la famille ARTIX-7, le ARTIX-7 XC7A200T, dont le fichier EBC contient 61 104 192 bits de configuration, sous la forme de 18906 trames de 101 mots de 32 bits chacun, et où le bit le moins significatif (LSB) d'un mot, considéré séquentiellement comme son premier bit, est à l'extrême droite. Notons que le fichier EBC débute par une trame vide (*padding frame*) et qu'il ne contient pas les bits de BRAM.

En appliquant les étapes décrites précédemment, nous avons constaté que la synthèse du code VHDL du design d'une chaîne regroupant tous les LUT se heurte à un problème au niveau de la mémoire et que le fichier EBC n'a pas généré à cause que Vivado est limité au niveau de nombre des itérations qu'il peut exécuter. Afin de surmonter ce problème, nous avons divisé la plage d'adresses physiques des 134600 LUT du FPGA en 5 régions, qui, à leur tour, ont été subdivisées en sous régions, définies dans le fichier XDC sous forme de *pBlock*. Ce dernier est un moyen de tracer le plan de design sur le FPGA tout en précisant l'adresse de début et celle de fin de chaque bloc comme le montre la figure 3.5.

Dans chacune de ces sous-régions, à chaque LUT a été associée une adresse fixe (technique aussi appelée placement absolu) via une contrainte dans le fichier XDC. En utilisant des scripts codés en langage C, nous avons généré automatiquement les lignes d'adresses des LUT correspondants à chaque tranche de chacune des sous régions, que nous avons insérées dans le fichier des contraintes.

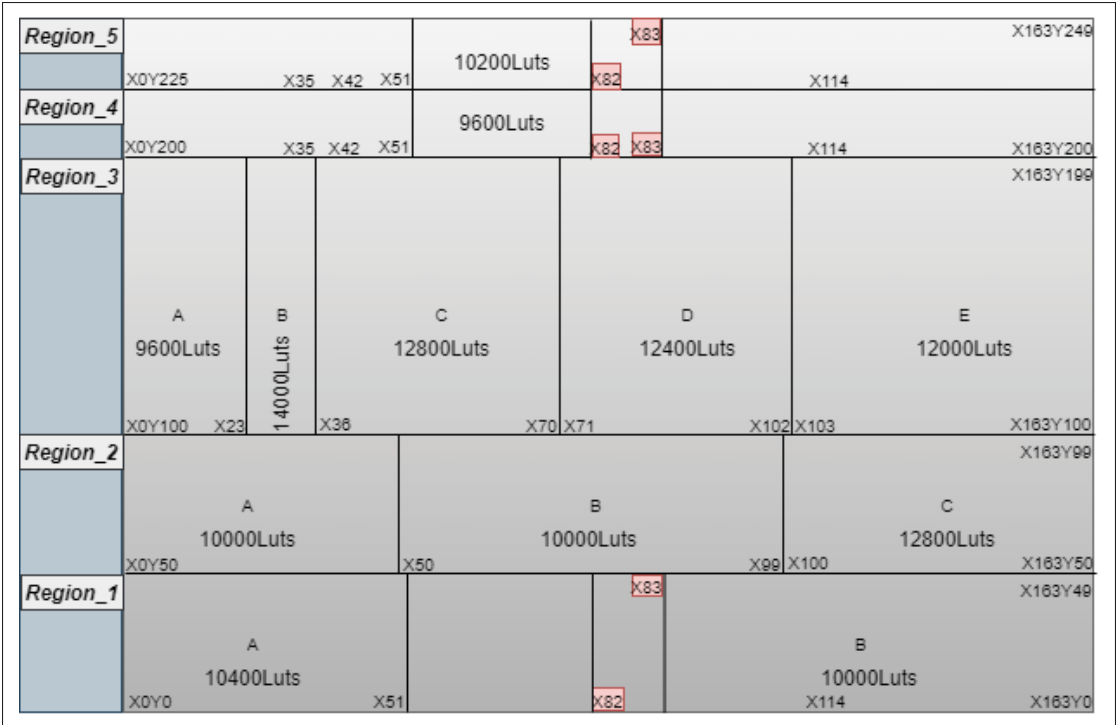


Figure 3.5 Plaque des adresses des tranches de l’ARTIX-7 A200T

Après avoir appliqué les étapes de 1 à 6 décrites ci-dessus sur chacune des sous-régions, nous avons réussi à identifier l’ensemble des adresses de bits de LUT.

Afin de nous assurer du bon fonctionnement de notre approche, nous avons vérifié le nombre des bits générés pour chaque sous-région. Nous avons au départ validé notre approche sur un exemple contenant 4 LUT seulement, pour lequel il faut retrouver 256 bits de différence. Pour s’assurer de l’exactitude des adresses extraites, nous avons suivi les étapes décrites en ANNEXE VI : Cette procédure (ANNEXE VI) à 12 étapes a été appliquée avec succès sur chacune des sous-régions. Cette vérification a permis de constater que parmi 134600 adresses des bits de LUT de l’ARTIX7, il y en a 800 qui sont non accessibles par défaut lors de l’implémentation. Ceci est illustré en figure 3.6. Les LUT non accessibles sont marqués d’un signe rouge et sont de plus en plus évidents à mesure que la figure est zoomée (de 3.6a vers 3.6d).

Par conséquent, nous avons identifié toutes les adresses des bits de LUT qui sont accessibles, ce qui correspond à 99.4% de nombre total des adresses des bits de LUT.

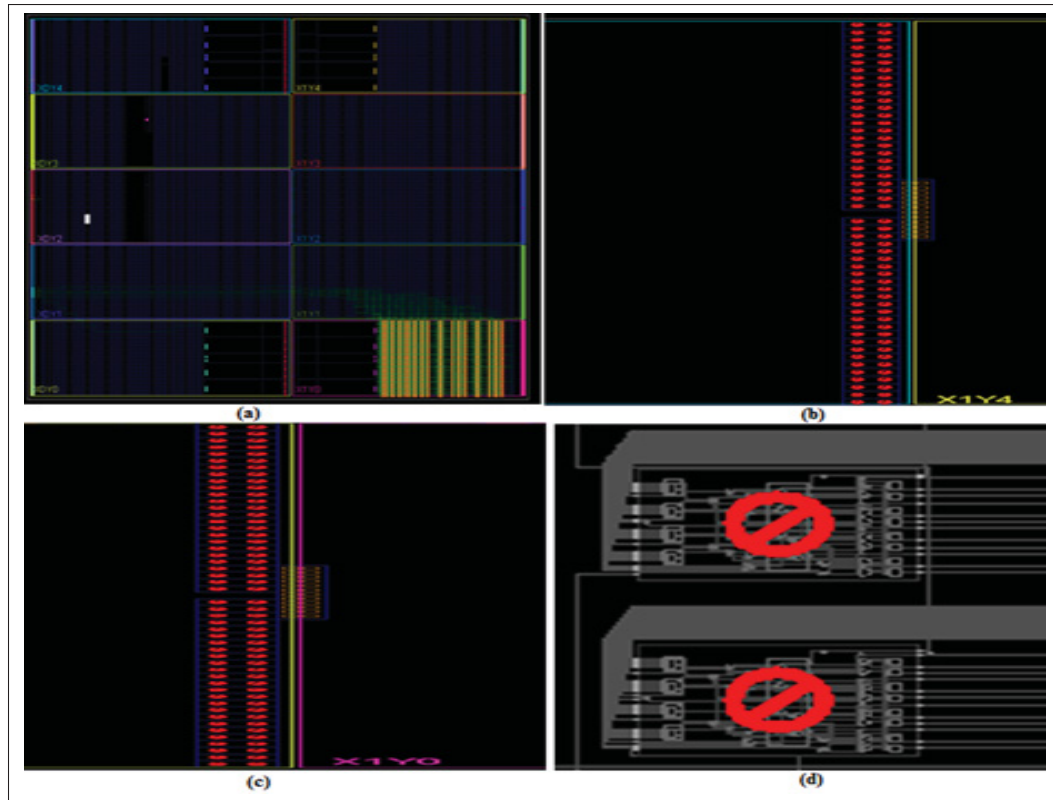


Figure 3.6 Design initialisant les LUT de FPGA ARTIX-7 de la Région1_B,
(b) (c) (d) adresses non accessibles

3.6 Méthode d'identification des bits essentiels de LUT de l'ARTIX-7

L'étape suivante dans le processus de conception est la détermination des parties essentielles des bits de LUT. La méthode d'identification des bits essentiels, illustrée en figure 3.7, consiste à transformer le fichier d'adresses de la totalité des bits de configuration des LUT décrit dans la section précédente dans un autre fichier qui a la même forme que les fichiers EBC et EBD.

À partir de fichier EBC on détermine le niveau logique de bit de configuration, s'il est à '0' ou à '1', ce qui est indiqué par l'étape ch1 dans l'algorithme. À partir de fichier EBD on précise s'il est un bit essentiel ou non essentiel, qui est indiqué par l'étape ch2 de l'algorithme. Finalement, l'étape ch3 permet de savoir si le bit fait partie des bits des LUT ou des Non LUT à partir de fichier d'adresses transformé.

Ces étapes sont appliquées sur les trois circuits de benchmarks afin de catégoriser leurs bits de configuration à '0' et à '1' dans différents ensembles : Bits des LUT essentiels, Bits des LUT non essentiels, Bits des Non LUT essentiels et Bits des Non LUT non essentiels. L'algorithme ci-dessous décrit la méthode utilisée.

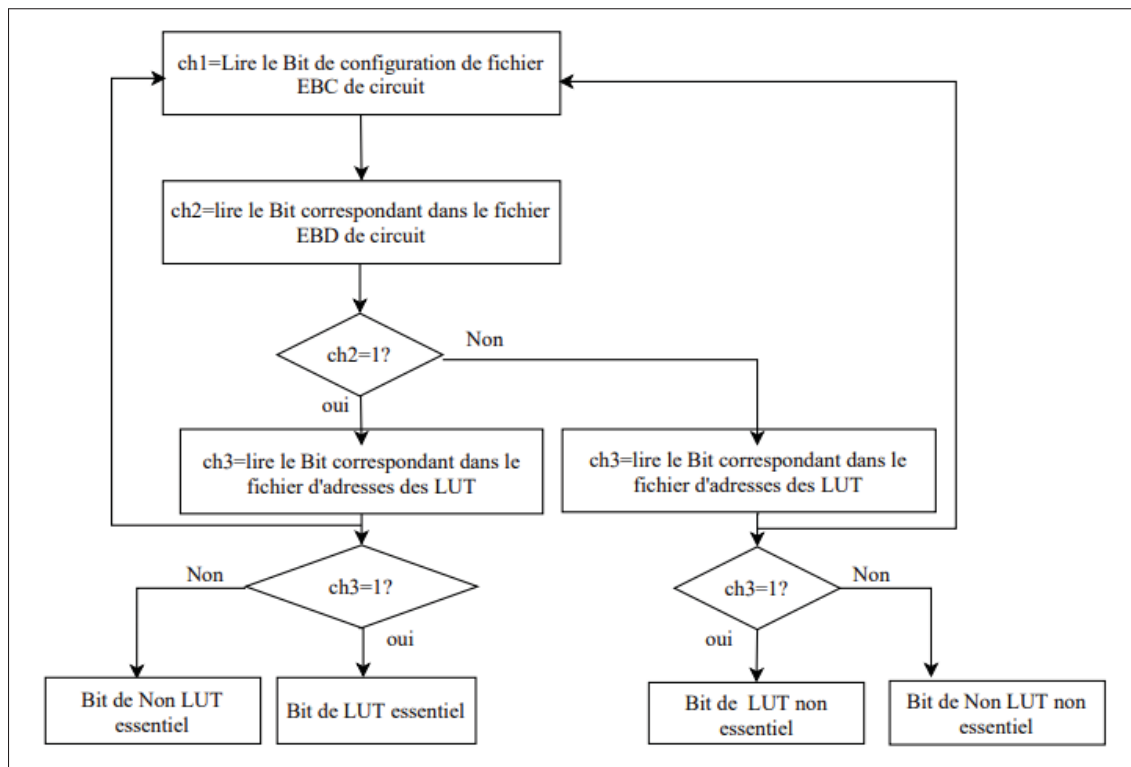


Figure 3.7 Algorithme de classification des bits de configuration

Les résultats trouvés lors de l'implémentation de cet algorithme pour les trois circuits d'essais sont présentés dans les tableaux ci-dessous 3.2, 3.3 et 3.4 :

Tableau 3.2 Nombre des bits essentiels et non essentiels des LUT et Non LUT du circuit B01

B01	essentiel LUT	essentiel Non LUT	Non essentiel LUT	Non essentiel Non LUT
Bit 1	1086697	977355	0	52648
Bit 0	1442775	2434703	6033728	49073054

Tableau 3.3 Nombre des bits essentiels et non essentiels des LUT et Non LUT du circuit B05

B05	essentiel LUT	essentiel Non LUT	Non essentiel LUT	Non essentiel Non LUT
Bit 1	893139	874177	0	41108
Bit 0	1194349	2217615	6475712	49404860

Tableau 3.4 Nombre des bits essentiels et non essentiels des LUT et Non LUT du circuit B12

B12	essentiel LUT	essentiel Non LUT	Non essentiel LUT	Non essentiel Non LUT
Bit 1	1476517	1327582	0	59907
Bit 0	1932763	3380330	5153920	47769941

3.7 Procédure d'injection de pannes

Le but principal de cette plateforme est de reproduire le plus fidèlement possible les résultats des tests d'émulation effectués au laboratoire TRIUMF de Vancouver.

La procédure d'injection présentée par la figure 3.8 se fait sur trois principales étapes : 1) la configuration de la carte FPGA, 2) l'émulation des SEU avec le SEM *Controller*, par injection de pannes dans la mémoire de configuration, et 3) la collecte des signatures générées lors de la détection d'une panne par le montage lui-même.

Configuration de FPGA : la configuration de la carte est une étape indispensable dans la procédure d'injection de pannes. Elle se fait par programmation de la puce via l'interface *Vivado* en utilisant le fichier «. bit » de circuit de benchmarks ciblé pour l'émulation . Celle-ci est réalisée à l'aide de hardware manager qui assure la connexion avec la carte.

Une reconfiguration de la puce est obligatoire avant chaque injection de panne.

Émulation des SEU et injection de pannes : l'émulation d'une SEU dans un FPGA basé sur une SRAM se produit en inversant un bit dans sa mémoire de configuration.

L'émulation peut être effectuée à un endroit bien précis et connu grâce à l'utilisation de SEM contrôler. Pour nos expérimentations, nous avons réalisé trois types d'émulation : 1) une émulation aléatoire, afin de distinguer l'efficacité des autres approches, 2) une émulation en fonction de la sensibilité relative entre les bits à '1' et les bits à '0', et 3) une émulation qui tient en compte à la fois de la sensibilité des bits selon leur contenu (0 ou 1) et selon leur nature (LUT versus Non-LUT). Dans ce dernier cas, comme Xilinx définit les bits essentiels comme les bits associés à la circuiterie du design (Robert, 2012), nous avons pensé qu'attaquer ces bits par l'injection de pannes en ciblant des sous-ensembles essentiels des bits des LUT et Non LUT pourrait provoquer plus rapidement un dysfonctionnement du circuit ciblé tout en conservant la représentativité de l'émulation. Davantage des détails seront fournis dans le chapitre suivant.

Collecte des signatures : cette étape est réalisée après que le montage ait détecté une panne. À ce moment, il y a affichage des signatures générées sur l'*Hyperterminal* à travers une interface UART par une connexion série de type RS232 tel que mentionné dans la section 3.2. Une fois les signatures sont affichées, il faut (manuellement) arrêter l'*injector* et ensuite reprogrammer la carte FPGA avec le fichier « .bit » d'origine. En ce qui concerne les signatures affichées sur le terminal après chaque détection de panne, nous devons les enregistrer dans des fichiers « .txt » distincts dont on aura besoin dans la dernière étape. Après avoir sauvegardé la signature, il faut recommencer l'injection à partir de l'adresse qui le suit et répéter la procédure d'injection de pannes jusqu'à avoir collecté le nombre nécessaire des signatures pour chaque circuit.

La dernière étape est l'analyse des signatures collectées. Celle-ci est effectuée en utilisant premièrement un script codé en python pour chaque circuit d'essais afin de générer un fichier Excel. Ce script va partitionner les signatures pour distinguer les valeurs des chaînes de balayages et des sorties primaires de la version de design sur lequel la panne a été détectée ainsi que celles de la version de référence. Deuxièmement, la différence en termes de bit (XOR) entre ces valeurs est calculée pour pouvoir déterminer par la suite les types des signatures (bonnes, fausses, bonnes manquées, définies au Chapitre 2) et leurs pourcentages.

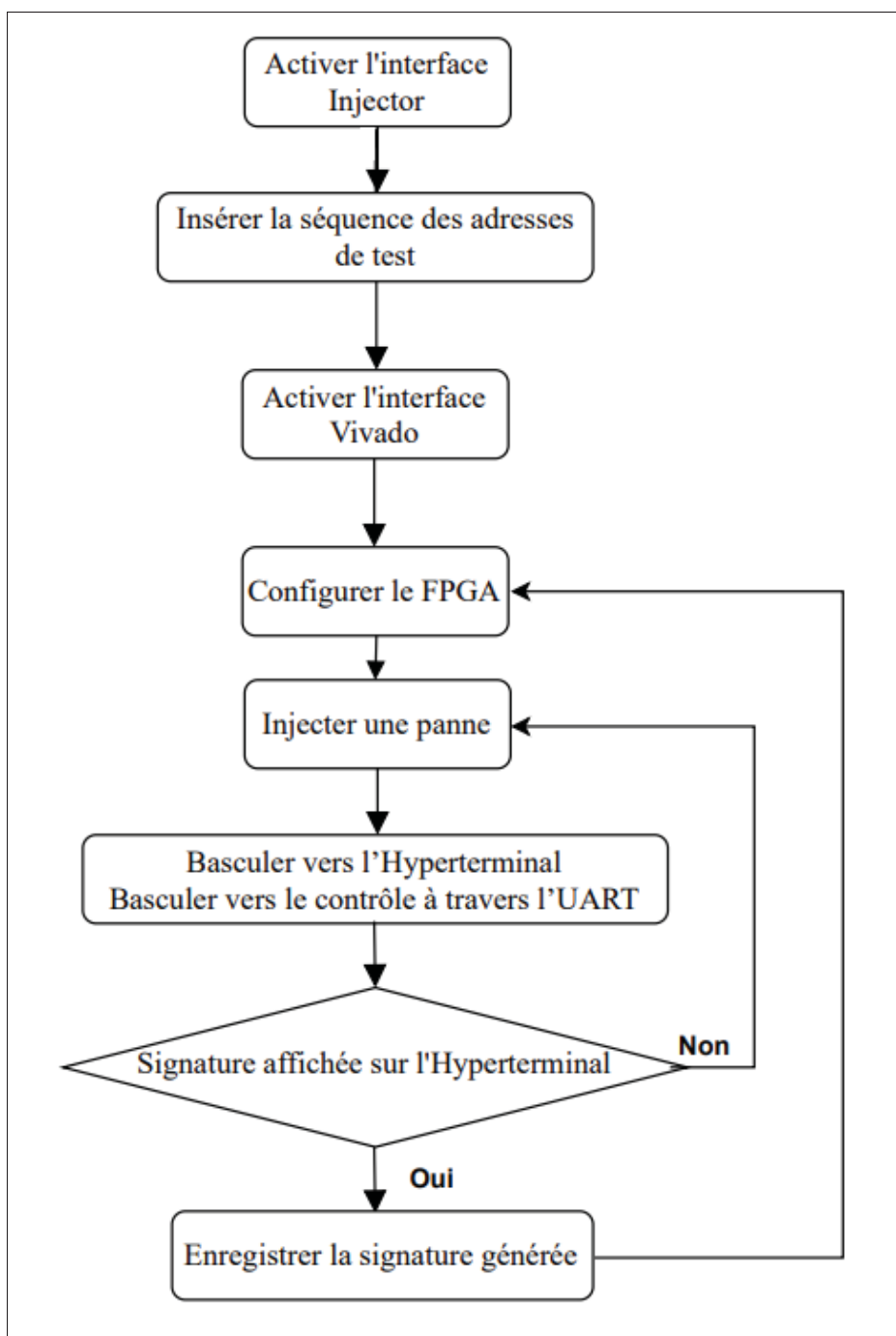


Figure 3.8 Procédure d'injection de pannes

3.8 Conclusion

Dans ce chapitre, nous avons premièrement présenté l'outil d'injection proposé par Xilinx pour la série 7 le *SEM Controller* pour injecter les pannes dans la mémoire de configuration. Bien également, les méthodes pour déterminer les bits de configuration des circuits ciblés dans nos expérimentations, les bits des LUT et Non LUT de la puce utilisée, ce qui nous a aussi permis d'identifier si les bits de configuration des circuits d'essais font partie des bits des LUT ou Non LUT de FPGA utilisé et en particulier s'ils sont des bits essentiels ou non essentiels, pour se servir des adresses générées de ces bits dans les approches d'injection de pannes. Enfin, nous avons présenté une description détaillée de la procédure d'injection de pannes.

Dans le chapitre suivant, nous allons entamer une description et une évaluation des résultats ainsi que la pertinence des approches implémentées.

CHAPITRE 4

AMÉLIORATIONS DES MÉTHODES D'INJECTION DE PANNES

4.1 Introduction

Au chapitre précédent, nous avons expliqué la méthodologie adoptée dans notre projet en présentant l'outil d'injection utilisé dans nos expérimentations ainsi que les méthodes d'identification des différents bits de configuration.

Dans le présent chapitre, nous allons détailler les approches d'injection de pannes proposées afin d'évaluer la fiabilité de la conception en émulant l'effet cumulatif que les SEU persistants ont sur les circuits FPGA à mémoire SRAM. L'idée était de prendre en compte la sensibilité relative entre les bits de configuration à "1" et à "0" ainsi que le type de bit lui-même dans le processus de génération des fichiers d'adresses de test.

Les approches proposées se concentrent sur la provocation de dysfonctionnement de circuit par injection de pannes dans des parties particulières de sa mémoire de configuration. Les nouvelles approches maximisent les inversions de bits essentiels pendant le processus d'injection de pannes. Elles permettent notamment l'estimation de la moyenne de bits critiques dans les différentes parties.

La validation des méthodes proposées a été effectuée par la comparaison des résultats obtenus avec ceux sous rayonnement ainsi qu'avec ceux fournis par l'injection de pannes aléatoires. Rappelons que pour l'ensemble des expérimentations d'émulation, nous ciblons les trois circuits définis précédemment (B01, B05 et B12 du banc d'essai ITC99). De plus, pour toutes nos expérimentations, nous avons choisi de générer 150 signatures pour le circuit B01, 200 signatures pour le circuit B05 et 250 signatures pour le circuit B12 afin d'obtenir suffisamment de bonnes signatures (au moins une centaine) et d'évaluer le nombre de ces bonnes signatures qui peuvent être manquées par le test fonctionnel.

4.2 Injection de pannes aléatoires

La première méthode avec laquelle nous allons comparer nos résultats est l'injection aléatoire des pannes. Cette méthode est utilisée comme référence afin de montrer la pertinence des autres approches qui ciblent des ensembles bien précis des bits de mémoire. Concernant cette méthode d'injection avec le *SEM Controller*, ce dernier identifie les adresses ciblées en utilisant deux générateurs de pseudo-aléas : un générateur qui identifie l'adresse de trame et un autre qui détermine la position du bit sur la trame.

En vue d'introduire un SEU, le module procède à la lecture de la trame choisie au hasard, ensuite à l'inversion du bit visé et finalement, la trame modifiée est enregistrée dans la mémoire de configuration à la même adresse que celle d'origine.

Dans ce contexte, le script injecte des pannes de manière aléatoire sur la zone de la mémoire de configuration accessible pour le SEM et utilisée par la conception en test, accumulant ces pannes jusqu'à ce qu'un dysfonctionnement soit détecté. Ainsi, jusqu'à 100000 adresses de positions binaires sont choisies au hasard à partir de la liste des bits de configuration de chaque circuit.

4.3 Injection de pannes basée sur la sensibilité relative entre les bits de configuration par rapport à leur contenu

La seconde méthode avec laquelle nous allons comparer nos résultats est l'injection de pannes tenant compte de la sensibilité relative des bits de configuration par rapport à leur contenu (Souari, 2016).

Étant donné que, face aux SEU, les bits "1" et "0" ont des sensibilités différentes, le niveau de réalisme des résultats obtenus lors de l'injection de pannes aléatoires peut en être affecté. Tel que mentionné au chapitre 2, dans (Souari, 2016), l'auteur a estimé cette différence sous faisceaux des protons avec des niveaux énergétiques appartenant à un intervalle de [8,105Mev]. Le facteur obtenu est de 1,8 pour le FPGA Virtex-5. Cette valeur a été interprétée à partir d'une

comparaison de nombre des bits inversés. Un facteur similaire a été obtenu sous faisceaux de neutrons au laboratoire TRIUMF, dans une série d'expérimentations menés par Simon Pichette et le professeur Thibeault, pour le FPGA ARTIX-7.

Nous avons utilisé ce facteur pour reproduire la méthode de (Souari, 2016) sur ce type de FPGA. Pour ce faire, nous avons extrait la liste d'adresses à injecter selon l'algorithme présenté par la figure 4.1, qui se base sur une sélection aléatoire mais biaisée par la sensibilité relative d'une adresse de bit en 3 étapes. La première étape est dédiée au choix du contenu du bit (0 ou 1), via un premier nombre aléatoire, $Random_1$, entre $]0,1[$, et on le compare à la valeur de Seuil, définie par l'équation suivante :

$$Seuil = \frac{Nbr_0}{Nbr_0 + (S * Nbr_1)} \quad (4.1)$$

Où Nbr_1 est le nombre total de bits à "1", Nbr_0 le nombre total de bits à "0", et $S (=1,8)$ le facteur de sensibilité relative des bits à "1" par rapport aux bits à "0". Si $Random_1$ est inférieur ou égal au Seuil, on cible un bit à "0" sinon on cible un bit à "1". La deuxième étape sert à identifier une ligne d'adresse à partir du fichier regroupant toutes les adresses des bits de configuration à la valeur logique ciblée. On choisit deux autres nombres aléatoires entre $]0,1[$ et on calcule la ligne d'adresse correspondante, selon la valeur ciblée :

$$Identifier_adresse_0 = Random_2 * Nbr_0 \quad (4.2)$$

$$Identifier_adresse_1 = Random_3 * Nbr_1 \quad (4.3)$$

La dernière étape est l'ajout de la ligne d'adresses identifiée dans la liste finale (fichier de sortie) d'adresses à injecter. L'exécution de l'algorithme s'arrête automatiquement après avoir collecté 100000 adresses dans le fichier de sortie.

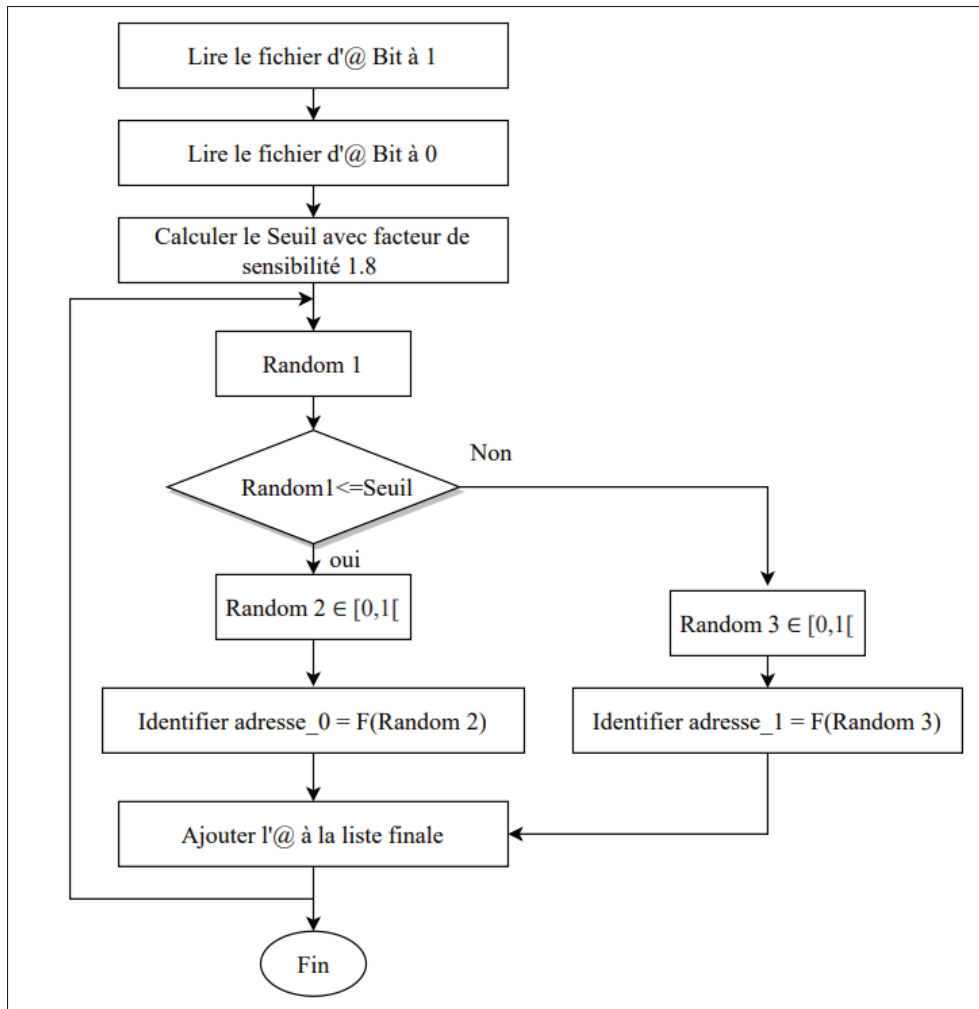


Figure 4.1 Algorithme de sélection de la liste des adresses avec sensibilité relative 1,8

4.4 Injection de pannes dans les LUT basée sur la différence de sensibilité relative en fonction du contenu et du type de ressource

La première méthode proposée dans ce mémoire est celle basée sur la différence de sensibilité aux SEU en fonction du contenu des bits de configuration et du type de ressources qui y sont associés. En effet, afin d'améliorer la procédure d'injection, nous nous sommes orientés vers l'identification des adresses des bits des LUT de design implémenté dans la carte FPGA tel que mentionné dans le chapitre précédent. Ainsi, dans le but de donner un plus grand réalisme à la

procédure d'émulation, nous avons ajouté comme critère de différenciation le fait que les bits de configurations soient contenus ou non dans des LUT.

Plus spécifiquement, nous avons choisi de cibler 4 catégories en fonction du type de ressource ciblée, LUT ou Non LUT (!LUT), et du contenu du bit, 1 ou 0 : **{LUT, 1}**, **{LUT, 0}**, **{!LUT, 1}** et **{!LUT, 0}**. Ceci implique donc une génération des séquences de bits sélectionnés en fonction de leurs sensibilités. De manière arbitraire, ces dernières ont été définies par rapport à la sensibilité de l'ensemble **{!LUT, 0}**. Cette manière de faire est similaire à celle utilisée dans (Souari, 2016), et a été utilisée pour faciliter les comparaisons avec les résultats de (Souari, 2016). Ceci dit, il est aussi possible de définir la sensibilité relative de manière globale (et non par rapport à une catégorie en particulier), ce qui sera fait plus loin dans ce chapitre.

L'étape d'identification des 4 catégories est décrite au chapitre 3. Cette identification permet le calcul des valeurs de sensibilité relative de chaque sous-ensemble. Ce calcul a été effectué sur la base des résultats d'expérimentations passées réalisées au laboratoire TRIUMF sous faisceaux de neutrons. Pour mener à bien ce calcul, nous disposons d'un fichier de relecture de référence (gold) pour chaque circuit de test avant le bombardement, et nous possédons en outre d'un ensemble de fichiers de relecture (readback) collectés après le bombardement lorsque le design est affecté par les faisceaux de neutrons à TRIUMF. Ces ensembles sont de 113 fichiers de relecture pour le circuit B01, 149 pour le B05 et 184 pour le B12.

L'idée est basée sur l'identification des numéros ainsi que des types de bits inversés dans les fichiers après bombardement en les comparant au fichier de référence. Cette identification a été implémentée à l'aide de scripts codés en langage C. Le tableau 4.1 suivant présente les valeurs relevées :

Tableau 4.1 Nombre des bits inversés dans les fichiers de relecture après bombardement

	NBI_1 LUT inversés de 1 à 0	NBI_2 !LUT inversés de 1 à 0	NBI_3 LUT inversés de 0 à 1	NBI_4 !LUT inversés de 0 à 1
B01	232	65	1654	2280
B05	178	226	1406	3267
B12	184	60	1242	1739

À partir de ces valeurs $\{NBI_i\}$ et le nombre de bit total dans chaque catégorie $\{total_i\}$ avec $i \in \{1, 2, 3, 4\}$, nous avons calculé les valeurs de sensibilité relative Sr_j avec $j \in \{1, 2, 3, 4\}$ par rapport à la sensibilité S_4 de $\{!LUT, 0\}$ qui sont définies selon l'ordre suivant $\{\{LUT, 1\}, \{LUT, 0\}, \{!LUT, 1\} \text{ et } \{!LUT, 0\}\}$ par $\{4.64, 4.81, 1.37 \text{ et } 1\}$ pour B01, $\{3.03, 3.75, 4.77, 1\}$ pour B05 et $\{3.55, 1.22, 4.95, 1\}$ pour B12 en fonction des équations suivantes :

$$S_i = \frac{NBI_i}{total_i} \quad (4.4)$$

$$Sr_j = \frac{S_i}{S_4} \quad (4.5)$$

L'étape qui suit maintenant est le calcul des nombres équivalents d'adresses dans chaque catégorie à utiliser dans la procédure d'injection de pannes en fonction des valeurs de sensibilité calculées précédemment suivant l'algorithme 7.1 en ANNEXE VII.

À partir des paramètres qui sont définis comme entrées (Input), nous commençons par le calcul du nombre des bits équivalents dans chaque catégorie en utilisant l'équation définie à la ligne 3. Un bit équivalent est défini comme un bit ayant la même probabilité d'être affecté par un SEU qu'un bit ayant une sensibilité relative égale à 1. En d'autres termes, un bit de configuration ayant une sensibilité relative égale à 2 va correspondre à 2 bits équivalents. Nous calculons ensuite la somme totale des bits équivalent en utilisant l'équation définie à la ligne 5.

Dans la suite, pour $T_F = 100000$ adresses à sélectionner à partir des fichiers d'adresses de 4 catégories, et en se servant de l'équation définie par le numéro 7, on calcule le nombre des *bitflips* attendus qui sont définis comme sorties (Output) dans chaque catégorie.

Après avoir calculé les nombres équivalents, nous passons à la génération des listes d'adresses à attaquer suivant l'algorithme 7.2 en ANNEXE VII.

Enfin, nous avons effectué l'émulation pour chaque circuit, les résultats trouvés sont présentés dans la section 4.6.

4.5 Injection de pannes dans les LUT essentiels et les Non LUT essentiels basée sur la sensibilité globale

La dernière méthode avec laquelle nous allons comparer nos résultats s'oriente dans la même direction que celle précédente. Elle se manifeste à attaquer les ensembles essentiels des LUT et !LUT. Cette méthode a pour intérêt non seulement de générer des résultats proches que celles sous faisceaux de neutrons tout en augmentant la vitesse d'émulation, mais aussi de fournir un critère de représentativité au niveau des types de bits ciblés, ce critère ayant été négligé dans les travaux présentés dans la littérature.

Cette priorisation de sous-ensembles spécifiques permet l'évaluation de l'impact de l'inversion de bit essentiel sur le fonctionnement du composant ainsi, une estimation du nombre des bits critiques peut être effectuée par la suite.

Tel qu'expliqué dans le chapitre précédent, nous avons identifié les parties essentielles des bits des LUT et Non LUT des circuits d'essais implémentés dans la carte FPGA.

Dans la section 4.4, en complément du calcul de la valeur de sensibilité relative par rapport à l'ensemble de $\{!LUT, 0\}$, les valeurs interprétées nous ont permis de relever les facteurs de sensibilité globale Sg_i pour chaque catégorie en appliquant l'équation suivante :

$$Sg_i = \frac{\frac{NBI_i}{\sum_{i=1}^4 NBI_i}}{\frac{total_i}{\sum_{i=1}^4 total_i}} \quad (4.6)$$

Cette équation a été appliquée pour chacun des trois circuits d'essai. Par la suite nous avons calculé la moyenne de sensibilité dans chaque catégorie, ces valeurs sont présentées au tableau 4.2.

Tableau 4.2 Facteurs de sensibilité globale

$Sg_1 : S_{LUT, 1}$	2,38
$Sg_2 : S_{!LUT, 1}$	1,44
$Sg_3 : S_{LUT, 0}$	3,1
$Sg_4 : S_{!LUT, 0}$	0,64

Cette méthode est basée sur l'émulation de chaque circuit par 4 catégories des parties des bits essentiels avec le facteur de sensibilité globale présentés au tableau 4.2. Les parties sont présentées sous ces ensembles :

- $\{LUT, 1, \text{essentiel}, Sg_1\}$: bits essentiels à "1" configurant les LUT utilisés dans le design ;
- $\{!LUT, 1, \text{essentiel}, Sg_2\}$: bits essentiels à "1" configurant les ressources autres que les LUT dans le design ;
- $\{LUT, 0, \text{essentiel}, Sg_3\}$: bits essentiels à "0" configurant les LUT utilisés dans le design ;
- $\{!LUT, 0, \text{essentiel}, Sg_4\}$: bits essentiels à "0" configurant les ressources autres que les LUT dans le design.

De la même manière décrite dans la section 4.4 en suivant les deux algorithmes 7.1 et 7.2 tout en changeant les paramètres d'entrées en fonction de valeurs de sensibilité globale et de fichiers des ensembles essentiels, nous effectuons le calcul des nombres des bits équivalents dans chacune des catégories mentionnées ci-dessus ainsi que la génération de fichier d'adresses à utiliser dans l'injection de pannes.

4.6 Comparaison et discussion des résultats des différentes méthodes

Dans cette section, nous présentons par les tableaux ci-dessous (4.3, 4.4, 4.5) les résultats des expérimentations des différentes campagnes d'injections décrites précédemment et effectuées pour chaque circuit d'essais, incluant les tests au laboratoire TRIUMF. Bien également, nous présentons par le tableau 4.6 la racine carrée de l'erreur quadratique moyenne des pourcentages des bonnes signatures et des bonnes manquées entre les méthodes d'injection et les résultats obtenus au TRIUMF pour les trois circuits d'essais.

Tableau 4.3 Résultats d'injection de B01 vs TRIUMF

Méthodes d'injection	Signatures			
	Bonnes	Bonnes manquées	% Bonnes	% Bonnes manquées
Injection aléatoire	119	21	79,3	17,6
Injection avec sensibilité 1,8	126	22	84	17,0
Injection dans 4 catégories avec Sensibilité !LUT 0	130	13	86,6	10,0
Injection dans 4 catégories avec Senibilités globale	129	16	86,0	12,4
TRIUMF	87	10	77,0	11,5

Tableau 4.4 Résultats d'injection de B05 vs TRIUMF

Méthodes d'injection	Signatures			
	Bonnes	Bonnes manquées	% Bonnes	% Bonnes manquées
Injection aléatoire	124	34	62,0	27,4
Injection avec sensibilité 1,8	129	33	64,5	25,5
Injection dans 4 catégories avec Sensibilité !LUT 0	125	39	62,5	31,3
Injection dans 4 catégories avec Senibilités globale	131	40	65,5	30,5
TRIUMF	91	25	61,1	27,5

Tableau 4.5 Résultats d'injection de B12 vs TRIUMF

Méthodes d'injection	Signatures			
	Bonnes	Bonnes manquées	% Bonnes	% Bonnes manquées
Injection aléatoire	118	57	47,2	48,3
Injection avec sensibilité 1,8	110	48	44,0	43,6
Injection dans 4 catégories avec Sensibilité !LUT 0	105	42	42,0	40,0
Injection dans 4 catégories avec Senibilités globale	94	41	37,6	43,6
TRIUMF	89	24	48,4	41,6

Tableau 4.6 Racine carrée de l'Erreur quadratique Moyenne moyenne, RCEQM, de trois circuits de test vs TRIUMF

Méthodes d'injection	RCEQM (%)	
	% Bonnes	% Bonnes manquées
Injection aléatoire	1,6	5,3
Injection avec sensibilité 1,8	5,2	3,6
Injection dans 4 catégories avec Sensibilité !LUT 0	6,7	2,5
Injection dans 4 catégories avec Senibilités globale	8,5	2,2

Rappelons que notre premier objectif est de reproduire le plus fidèlement possible les résultats obtenus sous faisceau des neutrons tout en étant représentatif au niveau des bits ciblés dans les expérimentations, en particulier ceux liés au pourcentage des bonnes signatures manquées par le test fonctionnel.

Commençant par l'injection aléatoire, tel que mentionné précédemment, cette méthode est principalement dédiée pour la comparaison avec les autres méthodes d'injection. Les valeurs des pourcentages trouvées pour le circuit B05 sont similaires à celles observées lors des expérimentations à TRIUMF. On a noté une moyenne RCEQM de 1,6% pour les bonnes signatures et de 5,3% pour les bonnes signatures manquées. Comme la valeur de 1,6% est la plus basse obtenue pour le pourcentage des bonnes signatures, une première constatation est que l'injection aléatoire peut s'avérer suffisante si c'est cette statistique seule qui est recherchée. Cependant, la prise en compte de la sensibilité relative des bits devient plus pertinente si on s'intéresse au pourcentage des bonnes signatures manquées.

En effet, en comparant les valeurs de RCEQM pour les pourcentages des bonnes signatures manquées pour les trois circuits, on peut constater une baisse graduelle de ces valeurs, qui partent de 5.3% pour l'injection aléatoire, à 3,6% en considérant le facteur de sensibilité de 1,8 (Souari, 2016), et à 2,2% et 2,5% en considérant la sensibilité relative des quatre catégories de nos deux méthodes proposées, avec et sans la considération du caractère essentiel des bits, respectivement.

Si la prise en compte de la sensibilité des bits a permis d'obtenir de meilleurs résultats pour ce qui est du pourcentage des bonnes signatures manquées, les résultats se sont avérés moins intéressants pour celui des bonnes signatures. Une hypothèse pouvant être avancée pour expliquer ceci s'appuie sur les limitations de l'injection via *SEM Controller*. Le *SEM Controller* a été élaboré pour ne pas considérer les bits de mémoire contenant les données des usagers, ce qui inclut. les bits des Block RAM, ceux des LUT utilisés comme mémoire distribuée dans un design et le contenu des registres. Tous ces éléments de mémoire peuvent voir leur contenu inversé sous l'effet des radiations. Cette hypothèse est que les éléments de mémoire inaccessibles à l'injection via *SEM Controller* ont davantage d'impact sur les fausses signatures. Cette hypothèse est partiellement supportée par l'observation selon laquelle de fausses signatures ont été déclenchées sans inversion de bit dans la section non BRAM des bits de configuration des DUT. Ce type de signature biaise donc la comparaison qui est faite, en augmentant le nombre des fausses signatures et, par conséquent, diminuant le pourcentage de bonnes signatures. Des investigations supplémentaires sont requises pour valider complètement cette hypothèse.

Rappelons que notre second objectif est de garantir une amélioration de la vitesse d'émulation. En parallèle de l'enregistrement des signatures, nous avons noté à chaque expérimentation le nombre des bits inversés NBI pour générer les signatures. Nous avons ainsi pu calculé la moyenne d'apparition des bits critiques MB_c (1 bit critique pour chaque x *bitflips*) en fonction du nombre des signatures générées Nb_s et le NBI ,

$$MB_c = \frac{NBI}{Nb_s} \quad (4.7)$$

Les résultats obtenus sont présentés au tableau 4.7. Cela nous permet de comparer le nombre moyen de pannes à injecter pour chaque signature.

Ces résultats (Tab 4.7) montrent qu'une certaine réduction du nombre de bits à inverser en tenant compte de la sensibilité des bits, mais surtout la pertinence de notre approche basée sur le ciblage des bits essentiels au niveau de l'accélération de la procédure d'injection de pannes.

Tableau 4.7 Nombre des bits inversés pour les trois circuits de test

	NBI			<i>MB_C</i>		
	B01	B05	B12	B01	B05	B12
Injection aléatoire	6863	10099	8620	48	51	35
Injection avec sensibilité 1,8	6278	8479	6228	42	42	25
Injection dans les 4 catégories avec sensibilité relative !LUT0	5287	6790	6440	35	34	26
Injection dans les 4 catégories essentielles avec sensibilité globale	996	1081	1988	7	5	8

La réduction du nombre de bits à inverser en tenant compte de la sensibilité des bits est dûe au fait que cette sensibilité favorise le ciblage des bits à 1, qui sont plus critiques lorsqu'ils sont hors LUT (Souari, 2016).

La réduction la plus substantielle est obtenue en ne ciblant que les bits essentiels des quatre catégories, où le nombre de bits à inverser par signature devient inférieur ou égal à 8, alors qu'il est inférieur ou égal à 51 pour l'injection aléatoire, inférieur ou égal à 42 pour l'injection avec le facteur de sensibilité de 1,8, et inférieur ou égal à 35 pour l'injection dans les quatre catégories sans cibler les bits essentiels. Rappelons que cette réduction substantielle est obtenue en conservant la représentativité associée aux sensibilités relatives qui n'était pas conservée par l'utilisation des bits essentiels dans (Souari, 2016).

4.7 Conclusion

Dans ce chapitre, nous avons présenté une description détaillée des différentes approches effectuées pour évaluer l'impact des SEU sur le FPGA ARTIX-7. Ensuite, nous avons effectué les campagnes d'injection de pannes en utilisant les séquences d'adresses dédiées pour chaque approche. Finalement, nous avons comparé les résultats trouvés avec ceux obtenus sous faisceaux de radiation à TRIUMF afin de montrer la pertinence et la fiabilité de nos choix.

CONCLUSION

L'objectif principal de ce projet était d'améliorer les méthodes d'injection de pannes par émulation afin d'évaluer la sensibilité des designs implémentés dans les FPGA à mémoire SRAM à l'égard des particules de rayons cosmiques. Les pannes qui ont été ciblées sont celles qui sont les plus fréquentes dans les circuits à mémoire SRAM, à savoir les SEU, qui changent la valeur stockée d'un zéro logique (0) à un (1) ou de un (1) logique à zéro (0).

Les approches que nous avons adopté dans notre projet avaient pour objectif d'imiter les résultats trouvés dans les tests sous faisceaux de neutrons au laboratoire TRIUMF à Vancouver, et de mettre en valeur la pertinence et l'efficacité de ces approches. En effet, nous avons visé l'amélioration de la procédure d'injection de pannes en termes de représentativité ainsi qu'au niveau de la vitesse d'émulation.

Commençant par le critère de représentativité, nous nous sommes focalisé sur le choix des bits de configuration ciblés dans les expérimentations. En effet, nous nous sommes concentrés sur trois axes principaux, à savoir la ressource contrôlée par le bit ciblé, la sensibilité et le contenu du bit ciblé en vue de mieux évaluer l'impact des SEU sur le design. L'identification des adresses des bits contenus dans les tables de conversion et ceux hors de celles-ci, pour un FPGA ARTIX-7, a été effectuée, ce qui a permis la création de quatre sous-ensembles en fonction de la ressource contrôlée (LUT ou non) et du contenu du bit (0 ou 1), et ce pour les trois circuits d'essai ciblés. Ceci a mené à l'émulation de ces sous-ensembles, en considérant la sensibilités de chacun d'entre eux face aux radiations. L'application de cette sensibilité a contribué à une meilleure reproduction des résultats de tests sous rayonnement au niveau du pourcentage des bonnes signatures manquées, la principale métrique visée par le projet. Par contre, l'ajout de cette sensibilité n'a pas amélioré les résultats au niveau du pourcentage de bonnes signatures, les meilleurs résultats étant obtenus par une émulation purement aléatoire.

Le second objectif réalisé dans l'amélioration de la procédure d'injection de pannes était l'augmentation de la vitesse d'émulation. Cet objectif a été atteint en ne ciblant que les parties dites essentielles des quatre sous-ensembles de bits, tout en considérant les valeurs de sensibilité globale de chacun. Cela nous a permis d'accélérer la procédure d'émulation tout en étant représentatif au niveau des bits inversés. De plus, ceci nous a permis de faire une estimation du nombre de bits réellement critiques de chacun des sous-ensembles.

RECOMMANDATIONS

Grâce à l'estimation du nombre de bits critiques dans les expérimentations, une estimation du nombre total de bits critiques dans le design implémenté sur le FPGA pourrait être réalisée par la suite. Une ouverture de projet réside dans la détermination de leurs adresses mémoire afin de mieux évaluer la vulnérabilité des FPGA envers les radiations cosmiques ainsi que la sensibilité des bits de configuration. Cette détermination devrait permettre une augmentation de la fiabilité de design. En effet, la collecte des informations relatives aux bits critiques dans une conception peut mener à une meilleure classification des pannes affectant le circuit et ainsi, à préciser les parties à protéger face aux SEU et de sélectionner la technique appropriée pour mitiger les effets transitoires. Cette idée peut être implémentée par l'automatisation de la procédure d'injection de pannes et par la modification de l'outil *injector* adapté dans nos expérimentations. La figure-A IV-1 résume cette procédure.

ANNEXE I

SEM CONTROLLER

Le module se présente sous la forme d’une boîte noire paramétrable et utilise plusieurs primitives d’appareil comme ICAP et FRAM_ECC et la fonction de Readback CRC pour superviser la fonction de détection des SEU. Les interfaces d’entrées-sorties de contrôleur SEM sont regroupés en 6 groupes comme ils sont représentées sur la figure-A I-1 :

- **Error Injection Interface** : permet l’injection de pannes ;
- **ICAP Interface** : est une liaison point à point entre le module de contrôle SEM et la primitive ICAP qui permet l’accès en lecture et en écriture aux registres à l’intérieur du système de configuration FPGA ;
- **FRAME_ECC Interface** : fournit une interface pour la fonction de détection d’erreurs logicielles douces dans le système de configuration du FPGA ;
- **Status Interface** : fournit un ensemble pratique de sorties décodées qui supervisent ce que fait le contrôleur ;
- **Monitor Interface** : fournit un moyen pour l’utilisateur de communiquer avec le module contrôleur à partir de la liaison série asynchrone l’UART ;
- **Fetch Interface** : fournit un dispositif permettant au contrôleur de demander des données à une source externe.

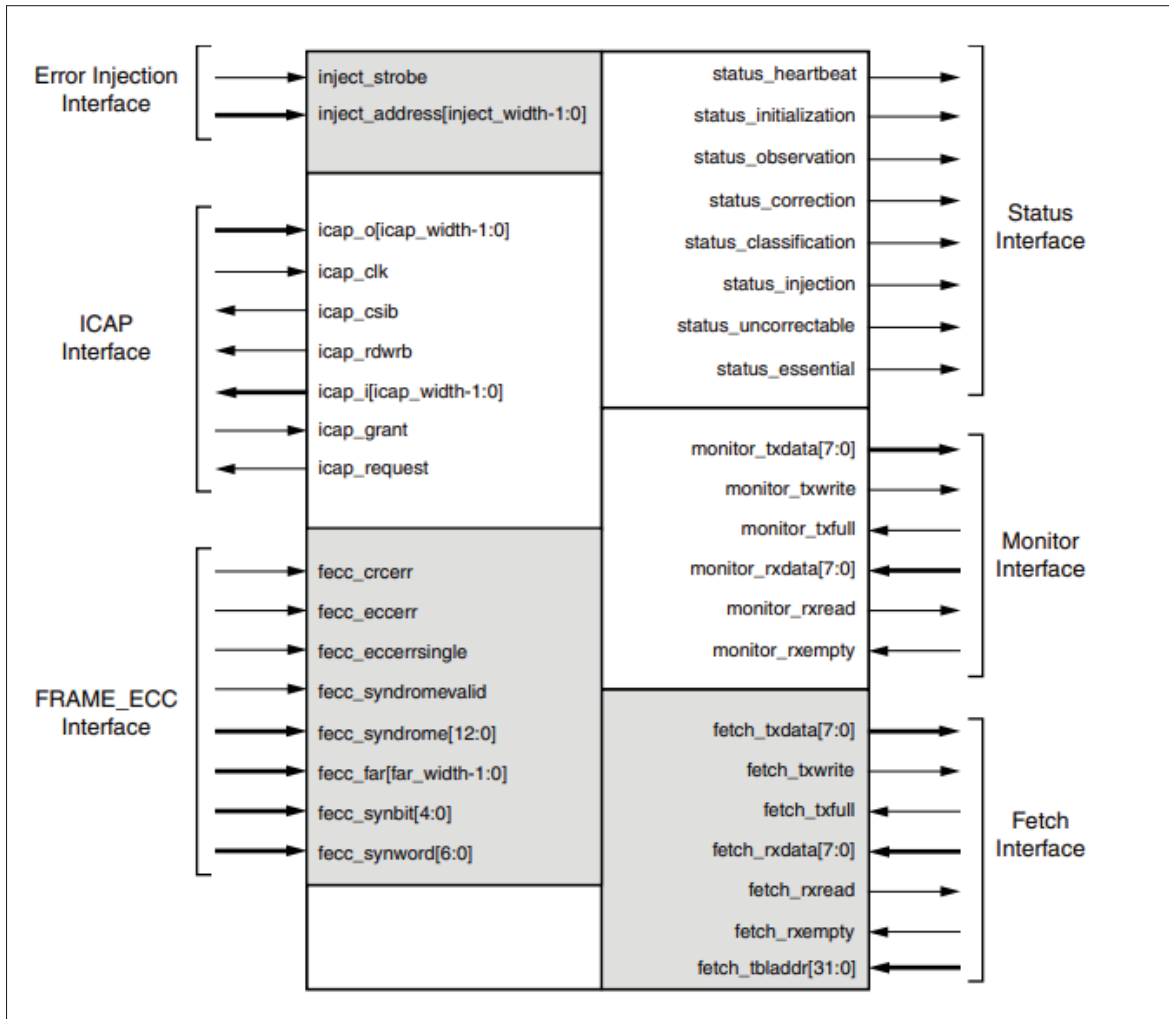


Figure-A I-1 Ports du SEM Controller
Tirée de Xilinx (2018a)

ANNEXE II

STRUCTURE FICHIER EBC ET EBD

L'ensemble du fichier (autre l'en-tête) est organisé comme un ensemble de lignes qui correspondent à des mots dans l'adresse du trame linéaire . Chaque mot contient 32 bits, ce qui correspond à la description du format de l'adresse de la trame. Chaque 101 lignes (mots) constitue une trame.

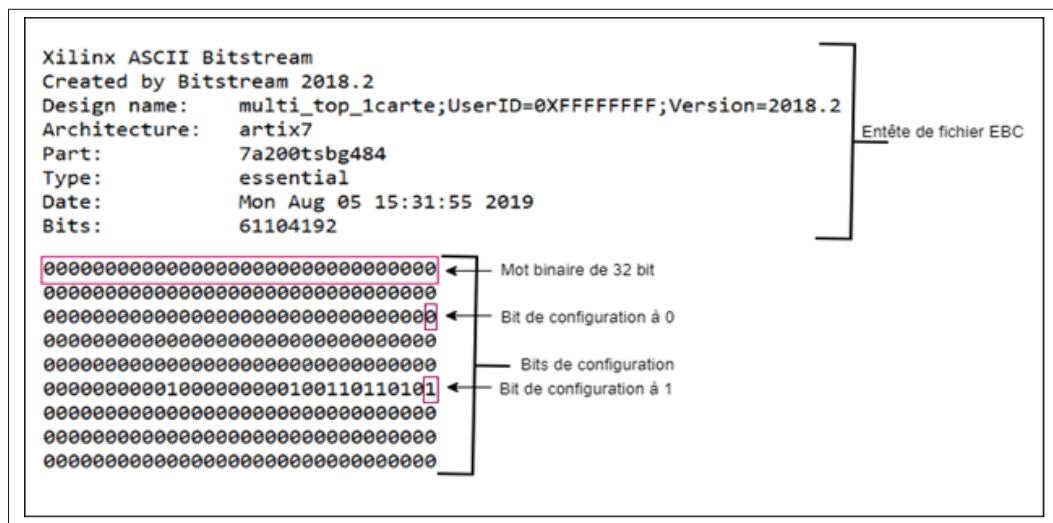


Figure-A II-1 Structure de fichier EBC

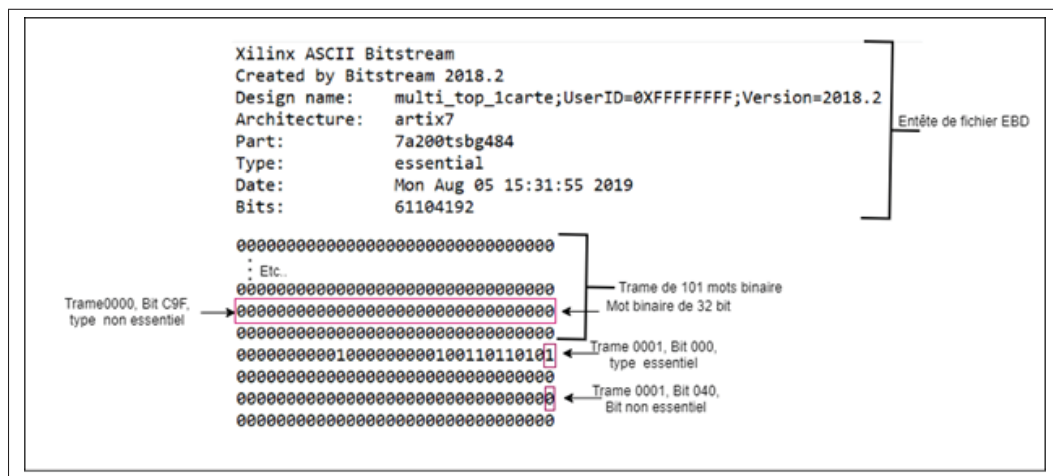


Figure-A II-2 Structure de fichier EBD

L'adressage de trame linéaire est simple, mais les adresses ne fournissent pas d'informations sur le type et l'emplacement physique de la trame. Le format de la commande d'injection d'erreur pour l'adresse de trame linéaire est illustré dans la Figure II-3 suivante, avec des bordures marquant les limites de quartet et un ombrage marquant des champs de bits séparés dans la commande.

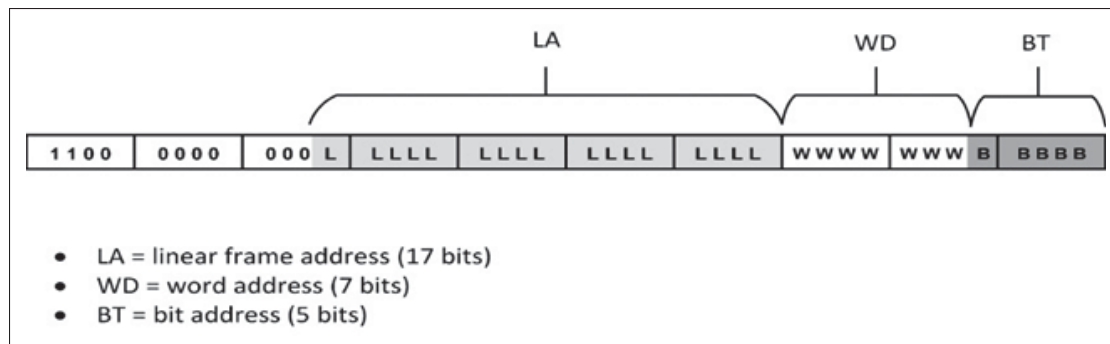


Figure-A II-3 Commande d'injection de panne
(adresse de trame linéaire)

$$LA = \frac{EBD_{line} - (EBD_{line} \bmod 101)}{101} - 1$$

$$WD = EBD_{line} \bmod 101$$

$$BT = 31 - character_{position}$$

ANNEXE III

INTERFACE *INJECTOR*

L'interface *Injector* utilisée pour assurer la communication avec SEM Controller et par la suite effectuer la fonction d'injection des pannes. La figure suivante montre les différentes fonctionnalités.

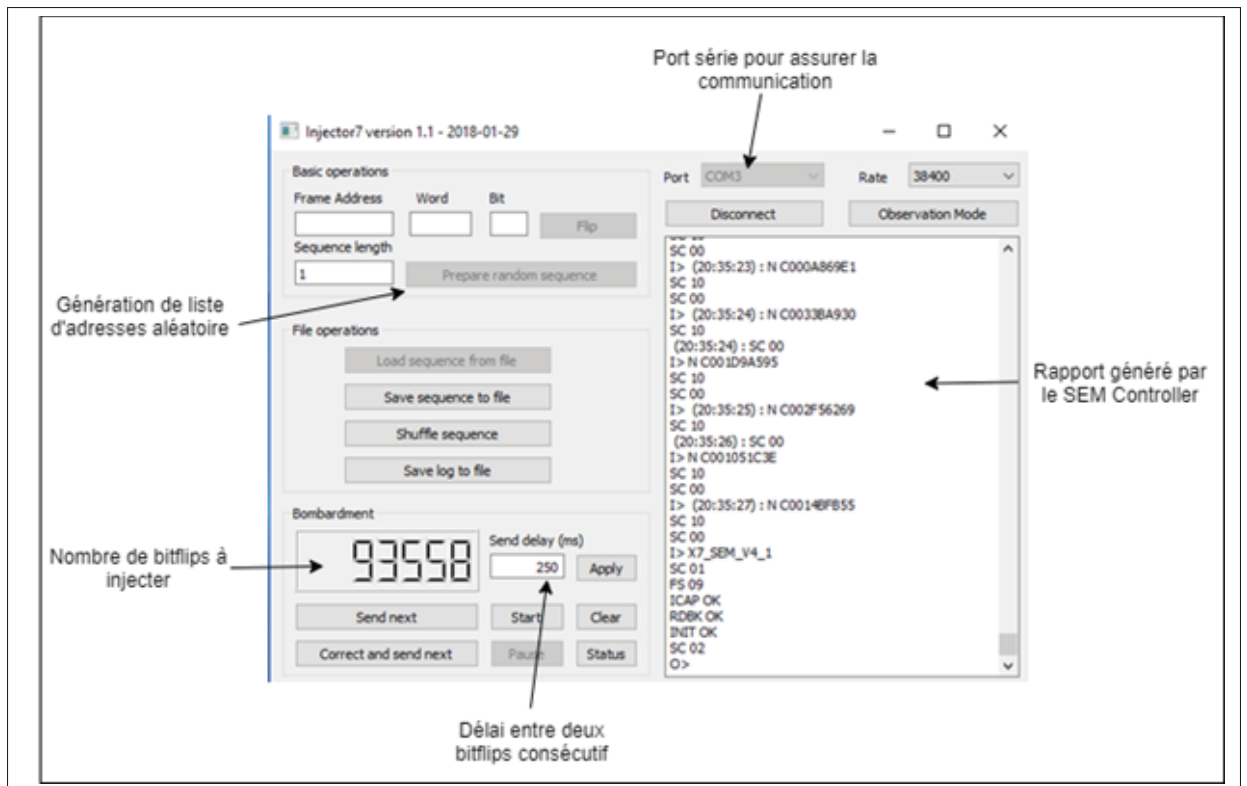


Figure-A III-1 Interface *Injector*

ANNEXE IV

PROCÉDURE D'IDENTIFICATION DES BITS CRITIQUES

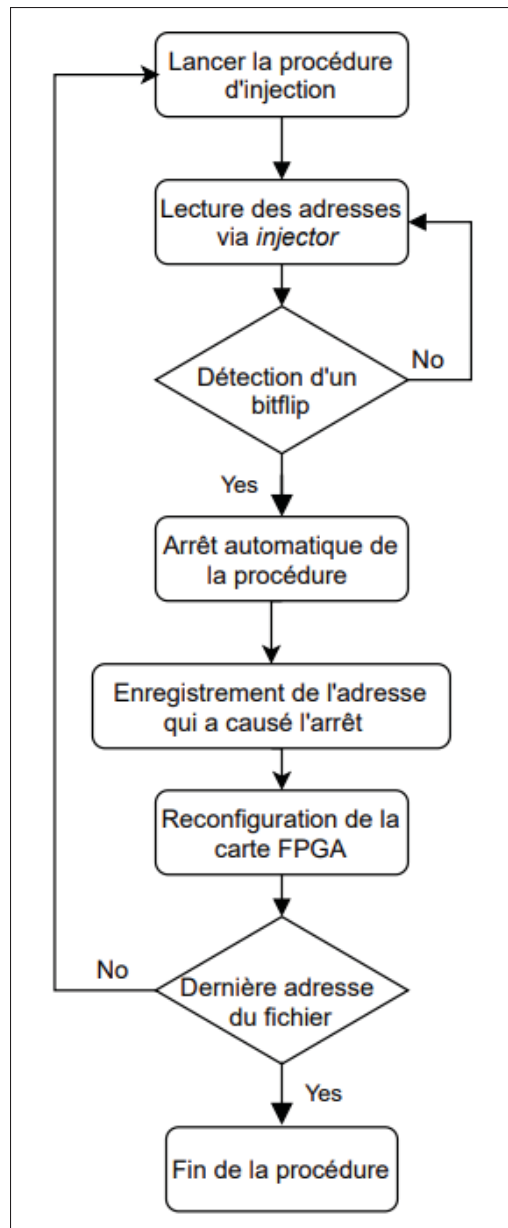
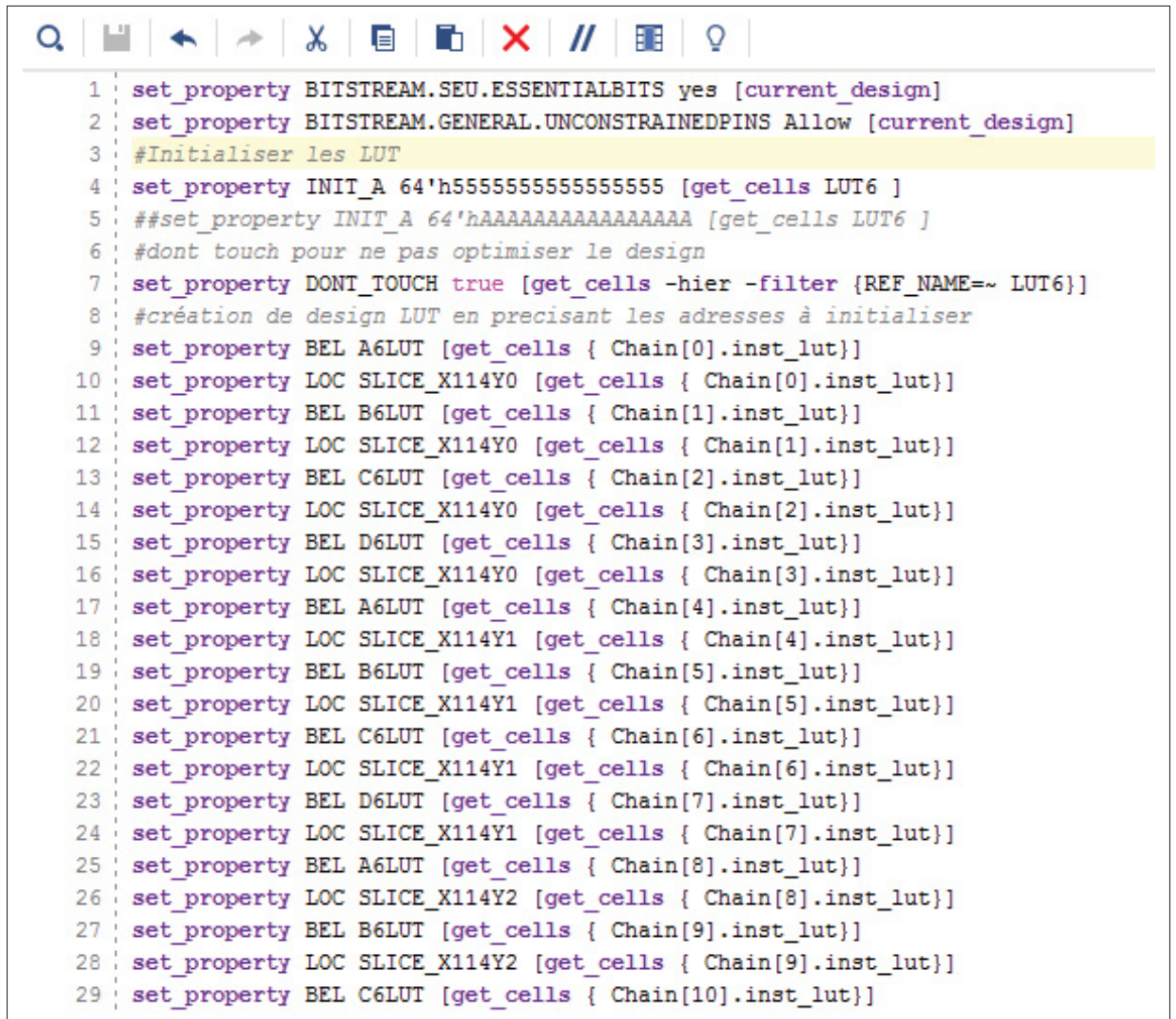


Figure-A IV-1 Procédure d'identification des bits critiques

ANNEXE V

EXEMPLE DE FICHIER DE CONTRAINTES XDC

The image shows a screenshot of a text editor window with a toolbar at the top containing icons for search, save, undo, redo, cut, copy, paste, delete, comment, and help. The editor displays a Verilog XDC file with 29 lines of code. Lines 1 and 2 are standard XDC properties. Line 3 is a comment. Lines 4 through 29 are a series of 'set_property' commands for LUTs, with some lines being commented out. The code is as follows:

```
1 set_property BITSTREAM.SEU.ESSENTIALBITS yes [current_design]
2 set_property BITSTREAM.GENERAL.UNCONSTRAINEDPINS Allow [current_design]
3 #Initialiser les LUT
4 set_property INIT_A 64'h5555555555555555 [get_cells LUT6 ]
5 ##set_property INIT_A 64'hAAAAAAAAAAAAAAAA [get_cells LUT6 ]
6 #dont touch pour ne pas optimiser le design
7 set_property DONT_TOUCH true [get_cells -hier -filter {REF_NAME=~ LUT6}]
8 #création de design LUT en precisant les adresses à initialiser
9 set_property BEL A6LUT [get_cells { Chain[0].inst_lut}]
10 set_property LOC SLICE_X114Y0 [get_cells { Chain[0].inst_lut}]
11 set_property BEL B6LUT [get_cells { Chain[1].inst_lut}]
12 set_property LOC SLICE_X114Y0 [get_cells { Chain[1].inst_lut}]
13 set_property BEL C6LUT [get_cells { Chain[2].inst_lut}]
14 set_property LOC SLICE_X114Y0 [get_cells { Chain[2].inst_lut}]
15 set_property BEL D6LUT [get_cells { Chain[3].inst_lut}]
16 set_property LOC SLICE_X114Y0 [get_cells { Chain[3].inst_lut}]
17 set_property BEL A6LUT [get_cells { Chain[4].inst_lut}]
18 set_property LOC SLICE_X114Y1 [get_cells { Chain[4].inst_lut}]
19 set_property BEL B6LUT [get_cells { Chain[5].inst_lut}]
20 set_property LOC SLICE_X114Y1 [get_cells { Chain[5].inst_lut}]
21 set_property BEL C6LUT [get_cells { Chain[6].inst_lut}]
22 set_property LOC SLICE_X114Y1 [get_cells { Chain[6].inst_lut}]
23 set_property BEL D6LUT [get_cells { Chain[7].inst_lut}]
24 set_property LOC SLICE_X114Y1 [get_cells { Chain[7].inst_lut}]
25 set_property BEL A6LUT [get_cells { Chain[8].inst_lut}]
26 set_property LOC SLICE_X114Y2 [get_cells { Chain[8].inst_lut}]
27 set_property BEL B6LUT [get_cells { Chain[9].inst_lut}]
28 set_property LOC SLICE_X114Y2 [get_cells { Chain[9].inst_lut}]
29 set_property BEL C6LUT [get_cells { Chain[10].inst_lut}]
```

Figure-A V-1 Fichier XDC

ANNEXE VI

VÉRIFICATION DE L'EXACTITUDE DES ADRESSES GÉNÉRÉES

1. Obtenir un flux binaire (*bitstream*) A «.bit » de design initialisant les 4 LUT à la valeur AAAA ;
2. Programmer le flux binaire A dans le FPGA ;
3. Faire la relecture du flux binaire A «.rbd » ;
4. Obtenir un flux binaire B «.bit » de design initialisant les 4 LUT à la valeur 5555 ;
5. Programmer le flux binaire B dans le FPGA ;
6. Faire la relecture du flux binaire B «.rbd » ;
7. Utiliser un script (en perl ou en C) qui produit la liste des adresses des bits des différences entre le «.rbd » A et le «.rbd » B ;
8. Programmer le flux binaire A dans le FPGA ;
9. Utiliser l'outil *injector* pour inverser la liste de différences obtenue à l'étape 7 ;
10. Faire la relecture du flux binaire A modifié (appelons-le C) ;
11. Utiliser le script utilisé à l'étape 7 pour produire la liste des adresses des bits des différences entre le «.rbd » C et le «.rbd » B ;
12. Vérifier qu'il y a 0 différence (sauf dans les BRAM).

ANNEXE VII

ALGORITHME DE GÉNÉRATION DE SÉQUENCE DE TEST

Algorithme 7.1 Calcul des nombres équivalents d'adresses avec la sensibilité relative en fonction du contenu et du type de ressource

Input : TF : nombre total des adresses qui est égale à 100000, $total_1$: nombre de bits LUT@1, $total_2$: nombre de bits NONLUT@1, $total_3$: nombre de bits LUT@0, $total_4$: nombre de bits NONLUT@0, Sr_1 : sensibilité relative bits LUT@1, Sr_2 : sensibilité relative bits NONLUT@1, Sr_3 : sensibilité relative bits LUT@0, Sr_4 : sensibilité relative bits NONLUT@0

Output : F_{L1} : nombre de *bitflips* attendus pour L_1 , F_{L2} : nombre de *bitflips* attendus pour L_2 , F_{L3} : nombre de *bitflips* attendus pour L_3 , F_{L4} : nombre de *bitflips* attendus pour L_4

1 L_1 : nombre de bits équivalents LUT@1, L_2 : nombre de bits équivalents NONLUT@1, L_3 : nombre de bits équivalents LUT@0, L_4 : nombre de bits équivalents NONLUT@0, N_B : nombre total de bits;

2 **for** $i \in (1, \dots, 4)$ **do**

3 | $L_i = total_i * Sr_i$;

4 **end**

5 $N_B = \sum_{i=1}^4 L_i$;

6 **for** $i \in (1, \dots, 4)$ **do**

7 | $F_{Li} = TF * (\frac{L_i}{N_B})$;

8 **end**

Algorithme 7.2 Génération de fichier d'adresses à utiliser dans l'injection de pannes

Input : F_1 : fichier d'adresses de catégorie LUT@1, F_2 : fichier d'adresses de catégorie NONLUT@1, F_3 : fichier d'adresses de catégorie LUT@0, F_4 : fichier d'adresses de catégorie NONLUT@0, F_{L1} : nombre de *bitflips* attendus pour L_1 , F_{L2} : nombre de *bitflips* attendus pour L_2 , F_{L3} : nombre de *bitflips* attendus pour L_3 , F_{L4} : nombre de *bitflips* attendus pour L_4

Output : F_o : fichier d'adresses à utiliser dans l'injection de pannes

```

1 for  $i \in (1, \dots, F_{L1})$  do
2   |  $F_o.add (F_1[\text{Random}(1, \dots, \text{length}(F_1))])$ 
3 end
4 for  $i \in (1, \dots, F_{L2})$  do
5   |  $F_o.add (F_2[\text{Random}(1, \dots, \text{length}(F_2))])$ 
6 end
7 for  $i \in (1, \dots, F_{L3})$  do
8   |  $F_o.add (F_3[\text{Random}(1, \dots, \text{length}(F_3))])$ 
9 end
10 for  $i \in (1, \dots, F_{L4})$  do
11   |  $F_o.add (F_4[\text{Random}(1, \dots, \text{length}(F_4))])$ 
12 end

```

BIBLIOGRAPHIE

- Alexandrescu, D., Anghel, L. & Nicolaidis, M. (2004). Simulating single event transients in VDSM ICs for ground level radiation. *Journal of Electronic Testing*, 20(4), 413–421.
- Arlat, J., Crouzet, Y. & Laprie, J.-C. (1989). Fault injection for dependability validation of fault-tolerant computing systems. *1989 The Nineteenth International Symposium on Fault-Tolerant Computing. Digest of Papers*, pp. 348–349.
- Artola, L. (2011). *Modeling of charge diffusion and collection mechanisms for SEE prediction in natural space environment for very scale integration devices*. (Thèse de doctorat).
- BEDI, R. (2014). *A comparison of space-grade FPGAs, Part 3*.
- Berrojo, L., González, I., Corno, F., Reorda, M. S., Squillero, G., Entrena, L. & Lopez, C. (2002). New techniques for speeding-up fault-injection campaigns. *Proceedings 2002 Design, Automation and Test in Europe Conference and Exhibition*, pp. 847–852.
- Bocquillon, A. (2009). *Evaluation de la sensibilité des FGPA SRAM-based face aux erreurs induites par les radiations naturelles*. (Thèse de doctorat, Institut National Polytechnique de Grenoble-INPG).
- Bolchini, C. & Sandionigi, C. (2010). Fault classification for SRAM-Based FPGAs in the space environment for fault mitigation. *IEEE Embedded Systems Letters*, 2(4), 107–110.
- Boudenot, J.-C. (1996). L'environnement spatial. *Que sais-je ?*
- Boué, J., Pétilion, P., Crouzet, Y. & Arlat, J. (1997). Early Experimental Verification of Fault Tolerance : The VHDL-base Fault Injection Tool MEFISTO-L. *28th IEEE Int. Symposium on Fault-Tolerant Computing (FTCS-28)*, pp. 168–173.
- Bushnell, M. & Agrawal, V. (2004). *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*. Springer Science & Business Media.
- Cha, H., Rudnick, E. M., Patel, J. H., Iyer, R. K. & Choi, G. S. (1996). A gate-level simulation environment for alpha-particle-induced transient faults. *IEEE Transactions on Computers*, 45(11), 1248–1256.
- Civera, P., Macchiarulo, L., Rebaudengo, M., Reorda, M. S. & Violante, M. (2001). FPGA-based fault injection for microprocessor systems. *Proceedings 10th Asian Test Symposium*, pp. 304–309.

- Entrena, L., López-Ongil, C., García-Valderas, M., Portela-García, M. & Nicolaidis, M. (2011). Hardware fault injection. *Soft errors in modern electronic systems*, 141–166.
- Fabrice, M. (2018). *La magnétosphère : sous l'influence de la Terre et du Soleil*.
- Faure, F., Velazco, R. & Peronnard, P. (2005). Single-event-upset-like fault injection : a comprehensive framework. *IEEE Transactions on Nuclear Science*, 52(6), 2205–2209.
- Ferron, J., Anghel, L., Leveugle, R., Bocquillon, A., Miller, F. & Mantelet, G. (2009). A methodology and tool for predictive analysis of configuration bit criticality in SRAM-based FPGAS : Experimental results. *2009 3rd International Conference on Signals, Circuits and Systems (SCS)*, pp. 1–6.
- Foucard, G. (2010). *Taux d'erreurs dues aux radiations pour des applications implémentées dans des FPGAs à base de mémoire SRAM : prédictions versus mesures*. (Thèse de doctorat, Institut National Polytechnique de Grenoble-INPG).
- Hobeika, C., Pichette, S., Leonard, M., Thibeault, C., Boland, J.-F. & Audet, Y. (2014). Multi-abstraction level signature generation and comparison based on radiation single event upset. *2014 IEEE 20th International On-Line Testing Symposium (IOLTS)*, pp. 212–215.
- Ian, K., Russell, T. & Jonathan, R. (2008). FPGA Architecture : Survey and Challenges. *Foundations and Trends® in Electronic Design Automation*, 2(2), 135-253. doi : 10.1561/10000000005.
- JACQUET, F. (2017). *Développement de bancs d'essais pour émulation et bombardement de particules*.
- Jenn, E., Arlat, J., Rimen, M., Ohlsson, J. & Karlsson, J. (1995). Fault injection into VHDL models : the MEFISTO tool. Dans *Predictably Dependable Computing Systems* (pp. 329–346). Springer.
- Karlsson, J., Gunneflo, U., Liden, P. & Torin, J. (1991). Two fault injection techniques for test of fault handling mechanisms. *1991, Proceedings. International Test Conference*, pp. 140.
- Khatri, A. R., Hayek, A. & Borcsok, J. (2020). Development and Verification of Serial Fault Simulation for FPGA Designs using the Proposed RASP-FIT Tool. *International Journal of Advanced Computer Science and Applications*, 11. doi : 10.14569/I-JACSA.2020.0110805.

- Lee, J.-Y., Feng, Z. & He, L. (2010). In-place decomposition for robustness in FPGA. *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 143–148.
- Lesea, A., Drimer, S., Fabula, J. J., Carmichael, C. & Alfke, P. (2005). The rosetta experiment : atmospheric soft error rate testing in differing technology FPGAs. *IEEE Transactions on device and materials reliability*, 5(3), 317–328.
- Lewis, D., Pouget, V., Darracq, F., Lapuyade, H. & Fouillat, P. (2001). Test de circuits intégrés par faisceau laser pulsé. *Colloque Interdisciplinaire en Instrumentation (C2I)*, pp. 1.
- Luu, A. (2009). *Méthodologie de prédiction des effets destructifs dus à l'environnement radiatif naturel sur les MOSFETs et IGBTs de puissance*. (Thèse de doctorat, Université de Toulouse, Université Toulouse III-Paul Sabatier).
- Madeira, H., Rela, M., Moreira, F. & Silva, J. G. (1994). RIFLE : A general purpose pin-level fault injector. *European Dependable Computing Conference*, pp. 197–216.
- Martínez, R., Gil, P. J., Martín, G., Pérez, C. & Serrano, J. J. (1999). Experimental validation of high-speed fault-tolerant systems using physical fault injection. *Dependable Computing for Critical Applications* 7, pp. 249–265.
- Michel, T., Leveugle, R., Saucier, G., Doucet, R. & Chapier, P. (1994). Taking advantage of ASICs to improve dependability with very low overheads/spl lsqb/PLC/spl rsqb. *Proceedings of European Design and Test Conference EDAC-ETC-EUROASIC*, pp. 14–18.
- Moran, A., LaBel, K., Gates, M., Seidleck, C., McGraw, R., Broida, M., Firer, J. & Sprehn, S. (1995). Single event effect testing of the Intel 80386 family and the 80486 microprocessor. *Proceedings of the Third European Conference on Radiation and its Effects on Components and Systems*, pp. 263–269.
- Nasa. (2010). *Ring of Fire*.
- Normand, E. (1996). Single-event effects in avionics. *IEEE Transactions on nuclear science*, 43(2), 461–474.
- Quinn, H., Robinson, W. H., Rech, P., Aguirre, M., Barnard, A., Desogus, M., Entrena, L., Garcia-Valderas, M., Guertin, S. M., Kaeli, D. et al. (2015). Using benchmarks for radiation testing of microprocessors and FPGAs. *IEEE Transactions on Nuclear Science*, 62(6), 2547–2554.
- Robache, R., Boland, J.-F., Thibeault, C. & Savaria, Y. (2013). A methodology for system-level fault injection based on gate-level faulty behavior. *2013 IEEE 11th International*

New Circuits and Systems Conference (NEWCAS), pp. 1–4.

Robert, L. (2012). *Soft Error Mitigation Using Prioritized Essential Bits* [v1.0].

Savanoor, L. F. & Guruprasad, H. (2020). Survey on Post Silicon Validation and Contribution of Scan Chain Technology.

Seifoori, Z., Ebrahimi, Z., Khaleghi, B. & Asadi, H. (2018). Introduction to emerging sram-based fpga architectures in dark silicon era. Dans *Advances in Computers* (vol. 110, pp. 259–294). Elsevier.

Siegle, F., Vladimirova, T., Ilstad, J. & Emam, O. (2015). Mitigation of radiation effects in SRAM-based FPGAs for space applications. *ACM Computing Surveys (CSUR)*, 47(2), 1–34.

Sieh, V., Tschäche, O. & Balbach, F. (1998). Comparing different fault models using verify. *DEPENDABLE COMPUTING AND FAULT TOLERANT SYSTEMS*, 11, 63–82.

Souari, A. (2016). *Stratégies facilitant les tests en pré-certification pour la robustesse à l'égard des radiations*. (Thèse de doctorat, École de technologie supérieure).

Souari, A., Thibeault, C., Blaquièrre, Y. & Velazco, R. (2016). Towards an efficient SEU effects emulation on SRAM-based FPGAs. *Microelectronics Reliability*, 66, 173–182.

Thaker, P. A., Agrawal, V. D. & Zaghloul, M. E. (2000). Register-transfer level fault modeling and test evaluation techniques for VLSI circuits. *Proceedings International Test Conference 2000 (IEEE Cat. No. 00CH37159)*, pp. 940–949.

Vanhauwaert, P. (2008). *Analyse de sûreté par injection de fautes dans un environnement de prototypage à base de FPGA*. (Thèse de doctorat, Institut National Polytechnique de Grenoble-INPG).

Wang, L., Yue, S. & Zhao, Y. (2007). Low-overhead SEU-tolerant latches. *2007 International Conference on Microwave and Millimeter Wave Technology*, pp. 1–4.

Xilinx. (2018a). *Soft Error Mitigation Controller LogiCORE IP Product Guide. PG036* [v4.1].

Xilinx. (2018b). *7 Series FPGAs Configuration, UG470* [(v1.13.1)].

Ziade, H., Ayoubi, R. A., Velazco, R. et al. (2004). A survey on fault injection techniques. *Int. Arab J. Inf. Technol.*, 1(2), 171–186.

- Ziade, H., Ayoubi, R. A., Velazco, R. & Idriss, T. (2011). A new fault injection approach to study the impact of bitflips in the configuration of SRAM-based FPGAs. *Int. Arab J. Inf. Technol.*, 8(2), 155–162.
- Ziegler, J. F., Curtis, H. W., Muhlfeld, H. P., Montrose, C. J., Chin, B., Nicewicz, M., Russell, C., Wang, W. Y., Freeman, L. B., Hosier, P. et al. (1996). IBM experiments in soft fails in computer electronics (1978–1994). *IBM journal of research and development*, 40(1), 3–18.