INJECTION DE FAUTES SIMULANT LES EFFETS DE BASCULEMENT DE BITS INDUITS PAR RADIATION

F. Faure

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

N° attribué par la bibliothèque
|__/__/__/__/__/__/__/__/__/__|

T H E S E

pour obtenir le grade de

DOCTEUR DE l'INPG

Spécialité : Microélectronique

préparée au laboratoire **TIMA** dans le cadre de
l'Ecole Doctorale d'« Electronique, Electrotechnique, Automatique, Télécommunications, Signal »

présentée et soutenue publiquement

par

Fabien Faure

Le 14 novembre 2005

Titre :

INJECTION DE FAUTES SIMULANT LES EFFETS DE BASCULEMENT DE BITS INDUITS PAR RADIATION

Directeur de Thèse : Raoul Velazco

JURY

M. Pierre Gentil	, Président
M. Jean Gasiot	, Rapporteur
M. Pacal Fouillat	, Rapporteur
M. Raoul Velazco	, Directeur
M. Robert Ecoffet	, Examinateur
M. Nicolas Renaud	, Examinateur

« Celui qui au départ insiste pour savoir où il va, quand il doit partir, et par où il passera n'ira jamais loin. »

Napoléon 1er.

Remerciements

Cette thèse a été réalisée au sein du groupe Qualification (QLF) du Laboratoire Techniques de l'Informatique et de la Microélectronique pour l'Architecture des Ordinateurs (TIMA). Au terme de cette étude, je souhaitai remercier :

Monsieur Bernard Courtois, Directeur de Recherche au CNRS et Directeur du Laboratoire pour m'avoir accueilli au sein de TIMA.

Monsieur Raoul Velazco, Directeur de Recherche au CNRS et Directeur du groupe QLF, pour m'avoir proposé ce sujet de thèse. Je lui exprime ma profonde gratitude pour son ouverture d'esprit, pour m'avoir fait confiance et m'avoir laissé les « coudées franches ». Je n'oublie pas sa disponibilité, ses conseils et critiques, son optimisme et sa légendaire bonne humeur.

Monsieur Pierre Gentil, Directeur de l'Ecole Doctorale, pour m'avoir fait l'honneur d'accepter de présider le jury de cette thèse.

Messieurs Jean Gasiot, Professeur de l'Université Montpellier II, et Pascal Fouillat, Professeur à l'ENSEIRB, pour l'honneur qu'il m'ont fait en acceptant d'être rapporteurs de ce travail. Leurs remarques judicieuses furent d'une aide conséquente.

Messieurs Robert Ecoffet, Ingénieur au CNES, et Nicolas Renaud, Ingénieur chez ATMEL, pour leur participation en tant qu'examinateurs du jury.

Je voulais aussi remercier l'équipe du groupe QLF. Tout d'abord Paul, qui m'accompagne depuis mes débuts à QLF, pour ses longues heures partagées de débogage, de tests sous radiations et j'en passe... Et Gilles, le « petit nouveau ». Je leur souhaite bonne chance.

J'adresse toute mon amitié aux membres du TIMA-CMP, et qu'ils me pardonnent le « squat » intensif de la machine à café.

Un grand merci à la communauté RADECS, ils sont trop nombreux pour les citer tous.

Finalement, j'adresse une pensée toute particulière à Sophie, mon adorable compagne, et toute ma famille, pour leur présence de chaque instant et pour leur soutien que le moral n'y est plus.

Résumé

OBTENIR une estimation du taux d'erreurs induit par les phénomènes de basculement de bit (*soft error rate*, SER) des équipement électroniques est d'un intérêt grandissant : alors que le SER d'un bit peut être extrêmement bas, la quantité toujours croissante de mémoire embarquée dans les circuits modernes, combinée avec leur utilisation dans des applications nécessitant une fiabilité élevée fait de l'évaluation du SER un pas important, si ce n'est obligatoire, avant l'introduction d'une nouvelle technologie sur le marché ou son utilisation dans un milieu « aggressif ».

Pour mesurer le SER d'un circuit numérique, la stratégie communément adoptée consiste à exposer le composant soit à un faisceau de particules (test accéléré), soit à l'environement de destination (test temps-réel) pendant qu'il effectue une certaine activité.

Le principal problème est d'utiliser le circuit d'une façon aussi représentative que possible de l'activité à laquelle il sera soumis dans l'application finale. Pour les circuits de type mémoire, ce but est atteint en écrivant un motif pré-déterminé dans les cellules mémoires, dont la corruption par basculement de bit sera identifiée lors de la relecture après exposition.

Plusieurs standards ont été publiés, définissant à la fois les pré-recquis et les procédures d'évaluation du SER des circuits intégrés.

Cependant ces documents traitent principalement de la qualification des circuits de type mémoire. Il n'y a pas d'accord sur les méthodes de qualification des circuits numériques complexes tels que les microprocesseurs. Il a été observé par plusieurs auteurs que l'utilisation directe des méthodes de test de mémoires dans le cas de processeurs (appelées test statiques) conduit à une surévaluation significative du SER.

Dans ce contexte, cette thèse s'attache à définir une methodologie permettant de mesurer puis de prédire le SER d'un processeur à l'aide d'une approche en trois étapes :

- En définissant une méthode de test statique correcte, permettant d'obtenir de façon précise la sensibilité du circuit au rayonnement ionisant ;
- En présentant une analyse détaillée de mesures effectuées sous radiation, dont le but est d'extraire un modèle statistique d'un test accéléré, c'est à dire *où* et *quand* les basculements de bits apparaissent ;
- En utilisant cette empreinte statistique pour reproduire au laboratoire, à l'aide d'injection de fautes, le comportement du circuit étudié sous radiations afin d'analyser et prédire le comportement d'une application quelconque exécutée par le processeur.

Les aspects théoriques sont discutés et illustrés à l'aide de mesures sous radiations obtenues sur un processeur SPARC exposé à un faisceau d'ions lourds. Le matériel et le logiciel développés durant cette thèse sont aussi présentés.

Abstract

ESTIMATING the *soft error rate* (SER) of digital equipment is a major concern : while the SER of a single bit can be extremely low, the increasing amount of bits per device of modern chips combined with their use in safety-critical applications makes the SER evaluation an important (if not mandatory) milestone before introducing a new technology to the market or using it in a space application.

To derive the SER of a device, the commonly adopted strategy consists in exposing the tested part to either a particle beam (accelerated test) or to its natural environment (real-life test) while it carries on a given activity.

The difficult point is to exercise the tested chip in a way as representative as possible of the one that will be used in the final environment. For memories this is easily done by loading them with a given pattern, whose corruption by soft-errors is identified by reading out the memory area and comparing it with the expected values.

Standards have been published, defining requirements and procedures for SER testing of integrated circuits.

However, the procedures presented in these texts apply primarily to memory devices. There is not such an agreement on the SER evaluation of complex digital devices like microprocessors. It has been shown that using methodologies such as the ones exposed above (called *static strategies* in the following) leads to overestimate the SER of processors.

In this context, the work done in this Ph.D. defines a methodology to measure and predict the SER of a processor using a three steps approach :

- By defining a correct static test strategy allowing to measure the cross-section of the processor's memory elements ;
- By presenting a detailed analysis of radiation ground testing data, aiming at extracting a statistical model of an accelerated radiation ground test, that is *where* and *when* SEUs do occur in the studied processor ;
- By using this statistical footprint and *fault injection* techniques to study the

behaviour of any application executed by the processor.
Theoretical aspects are discussed and illustrated with real radiation ground testing results obtained on a SPARC microprocessor. Hardware and software tools developed during this Ph.D. are also presented.

Sommaire

Remerciements iii

Résumé v

Abstract vii

Introduction xix

1 Radiations : Environnement et Effets sur les Circuits Intégrés 1
- 1.1 Origines . . . 1
  - 1.1.1 Soleil . . . 1
  - 1.1.2 Magnétosphère terrestre . . . 7
  - 1.1.3 Rayons Cosmiques . . . 11
- 1.2 Environnements Radiatifs . . . 13
  - 1.2.1 L'Héliosphère . . . 13
  - 1.2.2 Ceintures de radiations . . . 14
  - 1.2.3 Pôles et Anomalie Sud-Atlantique . . . 16
  - 1.2.4 Atmosphère . . . 17
- 1.3 Effets des Radiations sur les Circuits Intégrés . . . 19
  - 1.3.1 Mécanismes d'interactions . . . 19
  - 1.3.2 Dose . . . 21
  - 1.3.3 Evénements Singuliers . . . 25

2 Qualification de Microprocesseur face aux SEUs 31
- 2.1 Qualification de Microprocesseurs . . . 31
  - 2.1.1 Modélisation de Mémoires SRAM . . . 32
  - 2.1.2 Qualification de Mémoires SRAM . . . 33
  - 2.1.3 Extension aux Microprocesseurs . . . 34
- 2.2 Injection de fautes . . . 36
  - 2.2.1 Niveau transistor . . . 37
  - 2.2.2 Niveau Comportemental . . . 37

2.2.3 Emulation FPGA . . . 37
2.2.4 Emulation Logicielle . . . 38
2.2.5 Laser . . . 38
2.2.6 Méthode CEU . . . 39
2.3 Résumé . . . 40

3 Modèle d'Injection de Fautes 43
3.1 Motivations . . . 43
3.2 Méthode d'Injection de Faute Etendue . . . 45
3.2.1 Ciblage du cache donnée . . . 45
3.2.2 Ciblage du cache instruction . . . 46
3.2.3 Amélioration générale . . . 47
3.3 La Section Efficace : Définition . . . 47
3.3.1 Le Jeu de Fléchettes . . . 48
3.3.2 Formulation Mathématique . . . 48
3.4 Interprétation Probabiliste . . . 50
3.4.1 Définition Formelle d'une Probabilité . . . 50
3.4.2 Application à la Section Efficace . . . 50
3.5 Modélisation suivant la Loi Binomiale . . . 51
3.5.1 Modélisation . . . 51
3.5.2 Dérivation d'un intervalle de Confiance . . . 52
3.5.3 Résumé et Critiques . . . 54
3.6 Modélisation suivant une loi de Poisson . . . 55
3.6.1 Distribution de Poisson . . . 55
3.6.2 Intégration du paramètre temps . . . 55
3.6.3 Processus de Poisson . . . 56
3.7 Bilan du modèle temporel proposé . . . 56
3.8 Localisation de l'injection de fautes . . . 58
3.8.1 Cas Simple . . . 58
3.8.2 Cas Général . . . 58

4 Implantation et Premiers Résultats 61
4.1 Présentation du Véhicule d'Etude . . . 61
4.1.1 Unité de Calcul Entier . . . 62
4.1.2 Mémoire Cache . . . 62
4.1.3 Détails sur le Prototype . . . 63
4.2 Obtention des Sections Efficaces . . . 65
4.2.1 Organisation des Programmes . . . 65
4.2.2 Contrôle du Flot . . . 66
4.2.3 Remarques sur l'Ecriture des Programmes . . . 68
4.2.4 Dimension de la Boucle d'Exposition . . . 70

4.2.5 Protocole et Analyse des Résultats 71
4.3 Injection de Fautes 75
4.3.1 Implantation de la Routine CEU 75
4.3.2 Génération des Paramètres 77
4.3.3 Génération de Nombres Aléatoires 82
4.4 Système de Test 86
4.4.1 Version Préexistante 87
4.4.2 Amélioration 88
4.5 Résultats et Discussion 90
4.5.1 Résultats Radiations 90
4.5.2 Vérification de l'Uniformité 94
4.5.3 Vérification de l'Estimation de l'Erreur 96
4.5.4 Résultats d'Injection de Faute et Comparaisons 97

5 Etude de Programmes Applicatifs en Présence de SEUs 101
5.1 Taux de SEU journaliers 101
5.2 Présentation des Programmes 102
5.2.1 Définitions 102
5.2.2 Multiplication de Matrices 103
5.2.3 Tri 103
5.2.4 Sélection des Tailles 103
5.3 Mesures sous Radiations et Injection de Fautes 106
5.3.1 Comportements Observés 106
5.3.2 Résultats 107
5.3.3 Discussion 107
5.4 Explications 110
5.5 Calcul des $\tau_{inj,i}$ 114
5.6 Vérification de la Correction Apportée à la Méthode Classique 114
5.7 Conclusions 118

6 Conclusions et Perspectives 119

A Publications et Activités pendant la Thèse 123

Références 127

Table des figures

1.1 Coupe simplifiée du Soleil. 2
1.2 Effet du vent solaire sur la magnétosphère terrestre. 9
1.3 Exemple de trajectoire des particules piégées dans les ceintures de radiation. 12
1.4 Répartition des protons ($E > 10\ MeV$) dans les ceintures (Modèle AP8-MAX). Abscisses et ordonnées sont exprimées en rayons terrestres. 15
1.5 Répartition des électrons ($E > 1\ MeV$) dans les ceintures (Modèle AE8-MAX). Abscisses et ordonnées sont exprimées en rayons terrestres. 16
1.6 Position de UOSAT-3 lors de la détection d'erreurs dans sa mémoire. 17
1.7 Illustrations des réactions de cascades possibles lorsque les rayons cosmiques pénetrent l'atmosphère (adapté de [8]). 18
1.8 Diagramme en bandes illustrant les processus physiques mis en jeu lors de l'irradiation d'une structure MOS (d'aprés [23]). 24
1.9 Mécanismes de collection de charge : drift, funneling et diffusion (d'aprés [27]). 27
1.10 Illustration du phénomène SEU sur un point mémoire 0.25 μm. *Coin supérieur gauche* : Schéma électrique de la cellule. *Coin supérieur droit* : Layout correspondant. *Coin inférieur gauche* : Injection d'une charge collectée de 25 fC sur le drain du transistor N1. La charge collectée n'est pas suffisante pour faire basculer le point mémoire. *Coin inférieur droit* : Injection d'une charge de 27 fC dans le drain du transistor N1. La charge collectée est suffisante pour renverser le contenu du point mémoire (VL bascule). Les injections de charge sont faites en utilisant le modèle présenté dans [28] et [29]. 30

2.1 Principe d'injection de fautes selon la méthode CEU 39

3.1 Schéma de l'expérience. 49
3.2 Illustration du théorème central limite. La probabilité de réalisation d'un événement est arbitrairement fixée à $p = 0.3$. 53

4.1 Système de fenêtres de l'architecture SPARC. W_n représente la fenêtre n. 63
4.2 Implantation de la file de registre. L'instruction exécutée est : `add %g1, %g2, %g3` (%g3 = %g1 + %g2). 64
4.3 Structure de la base de code commune aux benchmarks. 67
4.4 Découpage de l'exécution en phases et compteurs associées. 68
4.5 Pourcentage de temps passé avec les registres exposées en fonction du nombre d'exécutions de la boucle d'attente. 72
4.6 Exemples de valeur du compteur `active - running` lors d'un test sous radiation. L'ordonnée est mesurée en cycle d'horloge. 74
4.7 Etapes de la routine CEU. 78
4.8 Illustration de la méthode de *rejection sampling*. 80
4.9 Découpage des temps d'injections en injections relatives à l'exécution. 83
4.10 Illustration de la technique de *self-reseeding* à l'aide d'un générateur linéaire congruentiel. La graine initiale est $X_0 = 0$. 85
4.11 Schéma d'implantation de la fonction `my_rand()`. La partie *self-reseeding* est masquée. 86
4.12 Synthèse du modèle d'injection de faute. T_{CEU} est la durée de la routine d'injection, T_{prog} est la durée du benchmark, f la fréquence du DUT, N_{bits} le nombre de bit cibles, Φ le flux de l'injection de faute, σ la section efficace des bits considérés et F la fluence de l'injection de fautes. 87
4.13 Architecture du testeur THESIC$^+$. 89
4.14 Courbe de section efficace de la file de registres du LEON. 92
4.15 Courbe de section efficace du cache données du LEON. 93
4.16 Distribution géographique des upsets dans le cache données du LEON (Adressage logique, 1500 upsets). 95
4.17 Démonstration de la convergence de la section efficace obtenue par injection de faute vers celle obtenue sous radiations, et estimation de l'erreur commise. La section efficace simulée est de 1 10^{-3} $cm^2/device$. 96
4.18 Abaque de calcul de la fluence à choisir en fonction de la section efficace et de la précision voulue. 97
4.19 Comparaison des taux d'upsets par seconde enregistrés sous radiations avec le taux d'injections de fautes par seconde obtenu grâce au modèle. La cible est la file de registres. Lors du test sous radiation, le flux était de 1 10^4 $p/cm^2/s$ et la section efficace mesurée est 2.30 10^{-4} $cm^2/device$. Ces paramètres ont été utilisés pour l'injection de faute. La section efficace obtenue après injection de fautes est 2.50 ± 0.3 $10^{-4} cm^2/device$. 99
4.20 Courbe quantile-quantile de la distribution temporelle des faute injectées en fonction de l'apparition des SEUs sous radiation. 100

5.1 Accès réussis et manqués au cache données en fonction de la taille des matrices. (1) correspond à une taille de 18×18, où les matrices résident entièrement dans le cache. Entre (1) et (2), les coefficients de C recouvrent ceux de A et B. Après (2), A, B et C se recouvrent mutuellement. 104
5.2 Accès réussis et manqués au cache données en fonction de la taille des listes. (1) correspond à une taille de 1024/2, où les deux listes résident entièrement dans le cache. Entre (1) et (2), les coefficients de la liste triée recouvrent ceux de la liste d'entrée. Après (2), une même liste se recouvre. 105
5.3 Taux d'accès réussis pour chaque programme en fonction de la charge équivalente. 106
5.4 Programme de tri. Histogrammes représentant le nombre d'injections par exécution selon les deux méthodes d'injection de faute. Pour le modèle présenté dans ce manuscrit, un flux de 10^4 particules par seconde est appliqué. 111
5.5 Programme de tri. Histogramme représentant le nombre d'injections par exécution selon le modèle présenté dans ce manuscrit, pour un flux de 10^3 particules par seconde. 112
5.6 Soit $\tau_{inj,1} = p$. A la première injection, il y a p chances que le système soit en erreur, et $1 - p$ chance que le système reste correct. A la deuxième injection, si le système était en erreur, il y reste. Il ne peut pas se réparer. S'il était correct, il y a p chances qu'il devienne incorrect, et $1 - p$ chances qu'il reste correct. Etc... 115
5.7 Résultats d'injection de faute avec i injections par exécution comparés à l'estimation fournie par l'équation 5.4. 117

Liste des tableaux

2.1 Exemples de pseudo-codes CEU pour le 8051, d'après [47]. L'exemple pour le compteur programme (PC) suppose que PC est emplilé dans une pile logicielle lors d'une interruption et dépilé lors du retour d'interruption. 40

4.1 Exemple de durée de chaque phase du benchmark `regfile`. La durée « atomique » de la boucle d'attente est de $4.1\mu s$, et N représente le nombre de fois où cette boucle est exécutée. 71

4.2 Test des différents générateurs de nombres aléatoires 83

4.3 Résultats obtenus sous radiation 91

4.4 Vérification de l'indépendance vis-à-vis du flux. 91

4.5 Vérification de l'indépendance vis-à-vis de la boucle d'attente. 94

4.6 Mesures sous radiations et injections de fautes 98

5.1 Exemple de taux d'upset journalier en orbite. 102

5.2 Mesures sous radiations et injections de fautes sur le programme de tri. Pour chaque expérience la fluence est de 10^6. Les valeurs moyennes de l'injection de fautes ont été calculées sur 30 échantillons. Les mesures sous radiations ont été prises avec des ions ayants un LET égal à 55.9 $MeV/cm^2/mg$. L'injection de fautes classique porte la mention « Pas applicable » dans la colone flux car ce paramètre n'est pas pris en compte dans cette méthode. 108

5.3 Mesures sous radiations et injections de fautes sur le programme de matrices. Pour chaque expérience la fluence est de 10^6. Les valeurs moyennes de l'injection de fautes ont été calculées sur 30 échantillons. Les mesures sous radiations ont été prises avec des ions ayants un LET égal à 55.9 $MeV/cm^2/mg$. L'injection de fautes classique porte la mention « Pas applicable » dans la colone flux car ce paramètre n'est pas pris en compte dans cette méthode. 109

5.4 Table de calcul du τ_{inj} pour le programme de tri, dans le cas ou le flux Φ est de 10^4 particules par secondes à partir de la distribution de Poisson de paramètre $\sigma \times \Phi \times t_{exec}$ et de la connaissance de $\tau_{inj,1}$. 117

Introduction

LE commencement de l'ère spatiale va de pair avec le lancement du satellite Russe Sputnik le 4 octobre 1957. Le 31 janvier 1958, le satellite Américain Explorer I, conçut et construit par le Jet Propulsion Laboratory décolle de Cap Canaveral. Il embarque un compteur Geiger proposé par Van Allen et destiné à mesurer le niveau ambiant de rayons cosmiques présents dans l'espace.

Cependant, alors que le satellite gagne en altitude, le détecteur cesse apparemment de compter. Van Allen comprend que cette « anomalie » est due à une saturation du compteur : l'appareil mesure en fait d'intenses radiations entourant la Terre. Il annonce sa découverte le 1er Mai 1958 [1], découverte qui sera confirmée lors du lancement d'Explorer III le 26 Mars 1958.

Quelques années plus tard, le 10 Juillet 1962, le premier satellite actif de télécommunication Telstar, construit par les Laboratoires Bell, est lancé. Le jour précédent, les Etats Unis d'Amérique procèdent à l'explosion d'une bombe nucléaire, Starfish, à plus de 200 km d'altitude. L'explosion injecte assez d'électrons dans les ceintures de Van Allen pour augmenter les flux d'un facteur 100. Les niveaux de radiations extrêmement élevés induits par cette explosion conduisent à la perte prématurée de Telstar le 21 Février 1963 [2].

L'enquête menée par une équipe des Laboratoires Bell rendit une augmentation des courants de fuites collecteur-base des transistors bipolaires comme responsable de la perte du satellite [3]. Cet événement est la première perte d'un satellite due aux radiations.

En 1975, un nouveau phénomène est observé. Binder *et al.* rapportent 4 inversions spontanées de bit dans des bascules bipolaires J-K durant les 17 années de fonctionnement d'un satellite de communication [4]. Dans cet article, les auteurs rapportent que ces basculements de bits pourraient être causés par des atomes de fer présents dans le vent solaire.

En 1978, le fabriquant Intel observe ces mêmes basculements de bits sur des mémoire de type DRAM [5]. Ce phénomène, appelé « Soft Errors » dans l'article,

est causé par une contamination radioactive des matériaux du boîtier des mémoires. Le passage d'un particule alpha dans la capacité de la cellule mémoire corrompt la charge enregistrée et conduit à une perte de l'information stockée.

Alors que Binder *et al.* attribuent les basculements de bits à un mécanisme d'ionisation directe, un article publié en 1979 rapporte la possibilité de basculement de bit par ionisation indirecte (protons et neutrons) [6]. Cet article est le premier à dénommer ce basculement de bit « Single Event Upset » (SEU).

La même année Ziegler prédit que les rayons cosmiques peuvent engendrer un problème de fiabilité au niveau du sol et aux altitudes avioniques [7].

Les années 80 voient la multiplication des méthodes de durcissement face aux SEU. Durant les années 90 on assiste à la convergence de deux marchés historiquement dissociés : d'un côté la communauté spatiale et militaire et de l'autre les fondeurs de circuits commerciaux.

Les premiers doivent affronter des réductions de coût ainsi que la disparition des fondeurs de circuits durcis et envisagent d'utiliser des circuits commerciaux (« Component Off The Shelf », COTS) pour des raisons de performance, de disponibilité et de prix. Les seconds, suite à différents problèmes coûteux sur site [8] et quelques mauvaises publicités [9], ont intégré la problématique Soft Error à la feuille de route ITRS [10].

Plusieurs standards ont été publiés, apportant méthodes et conseils pour la qualification de circuits face aux événements transitoires induits par les radiations [11], [12] et [13]. Cependant ces standards se concentrent sur la qualification de circuits de type mémoire vive. Les procédés de qualification de circuits comme les microprocesseurs ne sont pas discutés, si bien que chacun a sa propre « méthode ». De plus, il a été observé maintes fois que le taux d'erreur d'un système à base de microprocesseur est dépendant de l'application exécutée [14] [15].

C'est dans ce contexte que s'inscrit cette thèse. La nécessité de qualifier les microprocesseurs commerciaux face aux basculements de bits, tant dans les domaines de haute fiabilité (militaire, spatial, aéronautique, automobile) que dans les applications grand public, n'est plus à démontrer.

Le travail effectué s'articule autour de deux axes. Tout d'abord, il définit une méthode permettant de mesurer le comportement pire-cas d'un microprocesseur soumis à un faisceau de particules provoquant des basculements de bits. Finalement, à partir de ces mesures pire-cas une méthode réaliste d'injection de faute est établie, permettant d'étudier le comportement du système dans son entier (microprocesseur exécutant une application quelconque) en présence de basculements de bit sans utiliser de faisceau de particules.

Le premier chapitre se concentre sur la description des radiations : origines, mécanismes d'interaction avec la matière et effets sur les circuits intégrés.

Le second chapitre se propose de résumer les techniques de qualification de microprocesseurs face aux SEU. Nous introduisons alors la notion d'injection de fautes, et nous présentons les méthodes d'injection de fautes les plus utilisées pour simuler un basculement de bit.

Dans le troisième chapitre, nous identifions le principal défaut des méthodes d'injection de fautes actuelles : elles décrivent comment injecter une faute, jamais quand l'injecter de manière réaliste. Cette absence justifie ce travail de thèse et nous proposons un développement théorique permettant de déterminer des instants d'injection de faute réalistes.

Le quatrième chapitre présente l'implantation du modèle décrit précédement. Le matériel et le logiciel nécessaire pour conduire l'étude sont présentés. Les protocoles de prise de mesure et d'analyse sont présentés. Les premières confrontations entre les mesures sous radiations et celles obtenues par injection de fautes sont données.

Le cinquième et dernier chapitre propose une étude sur le comportement de programmes quelconques sous radiations. Nous montrons que les techniques classiques d'injection de fautes ne sont pas en mesure d'expliquer certaines données obtenues sous radiations, alors que la technique présentée dans ce mansucrit en est capable. Aprés quelques considérations sur l'adéquation entre test sous radiation et environnement final, nous montrons comment conserver l'approche classique d'injection de fautes, tout en permettant une comparaison des résultats avec les mesures prises sous radiation.

Finalement nous concluons ce manuscrit par un bilan général des travaux effectués au cours de cette thèse, et nous proposons des suites logiques à cette étude.

Chapitre 1

Radiations : Environnement et Effets sur les Circuits Intégrés

DANS ce chapitre, nous proposons une courte introduction aux radiations, leurs origines, ainsi que leurs effets sur les circuits intégrés.

La première section établit les origines des radiations dans le système solaire, et les acteurs qui participent aux particularités de ce milieu.

La deuxième section découpe l'espace Soleil-Terre en niveaux correspondants aux différents environnements radiatifs, et présente chacun d'eux.

La troisième et dernière section discute brièvement des effets des radiations sur les circuits intégrés.

1.1 Origines

Cette section a pour but de présenter les acteurs qui entrent en jeu dans la formation de ce que l'on appelle l'environnement radiatif : les populations de particules que l'on rencontre suivant le secteur où l'on se place.

1.1.1 Soleil

Le Soleil est une étoile, ce qui veut dire qu'il émet sa propre lumière. La question de savoir d'où venait ce rayonnement s'est longtemps posée, et nombreuses furent les théories pour y répondre. Ce n'est que durant les années 20 qu' A. S. Eddington suggéra que l'énergie des étoiles était d'origine nucléaire [16].

Plus précisément, c'est une étoile naine de type spectral G2 qui évolue sur la séquence principale du diagramme de Hertzsprung-Russell. Les modèles d'évolution stellaire prédisent que le Soleil, âgé de 4,6 milliards d'années, est approximativement à la moitié de son existence et qu'il évoluera vers un corps céleste de type

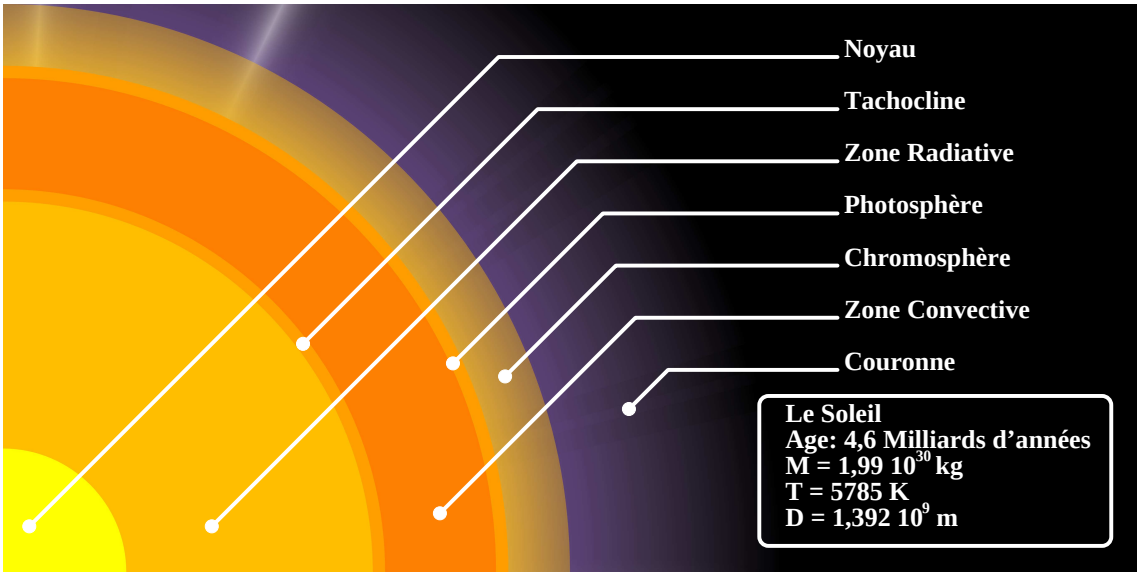

FIG. 1.1 – Coupe simplifiée du Soleil.

naine blanche.

A lui seul, le Soleil représente 99,8 % de la masse totale du système solaire, les 0,2 % restants incluant les planètes, dont la Terre. Le Soleil est principalement constitué d'hydrogène (90 %) et d'hélium ($\sim$ 8 %). Au centre du Soleil, des réactions thermonucléaires de fusion convertissent l'hydrogène en hélium. L'énergie produite à cette occasion est émise sous forme de lumière, de radiations et de particules.

Le Soleil est défini par des paramètres globaux tels que sa masse $m_\odot = 1,99.10^{30}\ kg$, sa magnitude $M_\odot = 4.8$, son diamètre $D_\odot = 1,392.10^9\ m$ et sa température effective $T_e = 5785\ K$.

Afin de mieux appréhender la complexité de la structure solaire, la présentation qui suit est divisée en deux parties, l'une présentant les couches internes du Soleil, et l'autre son atmosphère. Une coupe simplifiée du Soleil est représentée sur la figure 1.1.

Le lecteur désirant plus de détails pourra se référer à [17] et [18].

Structure Interne

L'intérieur du Soleil peut à son tour être découpé en quatre régions dans lesquelles les processus physiques qui y ont lieu sont totalement différents : le noyau ou coeur, la zone radiative, la tachocline et la zone convective.

Noyau La température au centre du Soleil est de $15,43.10^6\ K$ et la masse volumique y est voisine de $145,7\ g.cm^{-3}$. C'est là que se déroulent les réactions nucléaires de fusion de l'hydrogène en hélium qui fournissent l'énergie du Soleil. Chaque seconde, environ 4 millions de tonnes de matière se transforment en photons très énergétiques (photons γ) qui vont s'échapper vers l'extérieur, repoussant les couches de gaz qui, sous l'effet de l'attraction gravitationnelle tendent à tomber vers le coeur du Soleil. C'est ainsi que se préserve l'équilibre de la structure. Ces réactions de fusion s'accompagnent d'émission de neutrinos, extrêmement difficiles à détecter puisqu'ils sont capables de traverser toutes les couches du Soleil, de même que la Terre, sans être perturbés.

Actuellement, environ 40 % de l'hydrogène du noyau a été consumé. Température et densité décroissent rapidement lorsque l'on s'éloigne du centre. Par exemple à la limite du noyau ($\sim$ 175000 km du centre), la température a diminué de moitié et la masse volumique est de l'ordre de 20 $g.cm^{-3}$.

Zone Radiative Cette zone s'étend de $0,25\ R_\odot$ à $0,75\ R_\odot$ avec une température passant de $7.10^6\ K$ à $2.10^6\ K$ et une densité variant de 20 à $0,2$. En raison de la densité de cette couche, le rayonnement issu du noyau ne peut se propager sur une grande distance sans « heurter » les atomes du milieu. Il est ainsi absorbé pour être émis de nouveau, avec une énergie différente, dans une direction quelconque. L'énergie se transporte dans cette zone par collisions électrons-photons, de choc en choc, de façon aléatoire. On appelle cette région de l'intérieur solaire « zone radiative » en raison du mode de transport de l'énergie sous forme de rayonnement.

Un photon émis dans le noyau va mettre environ un million d'années pour traverser la zone radiative et en émerger sous forme de photon UV ou visible. On considère en général que l'ensemble constitué du noyau et de la zone radiative est en rotation rigide autour de l'axe nord-sud, la période de rotation étant de 26 jours.

Tachocline La tachocline est l'interface entre la zone radiative et la zone convective. Les mouvements de gaz que l'on trouve dans la zone convective disparaissent à mesure que l'on s'enfonce dans ses profondeurs, jusqu'à retrouver les conditions « calmes » qui caractérisent la zone radiative.

Cette fine couche (d'épaisseur estimée à 3000 km) est l'objet d'un intérêt accru depuis qu'a été émise l'hypothèse selon laquelle le champ magnétique solaire serait issu d'une dynamo présente dans cette zone. En effet, les variations de vitesse des gaz seraient à l'origine de phénomènes d'étirement et d'amplification des lignes de champ magnétique. Il semblerait qu'il y ait aussi des changements brutaux de composition chimique dans cette zone.

Zone Convective La zone convective s'étend de la tachocline à la surface visible du Soleil ($\sim 2.10^5\ km$). La densité à la surface ($\sim 2.10^{-7}$) est 10^6 fois inférieure à la densité à la base de la zone de convection. Elle est donc beaucoup moins dense que la zone radiative, ce qui fait que des mouvements locaux de la matière y sont possibles. La température variant de $2.10^6\ K$ à la base de la zone de convection à $5785\ K$ à la surface, cette chute permet à l'hydrogène atomique de se former.

Les photons qui l'atteignent ont perdu beaucoup d'énergie au cours de leur périple dans la zone radiative, et ils sont absorbés par cet hydrogène neutre et par l'ion H^- résultant de la capture d'un électron par un atome d'hydrogène. Cette absorption chauffe et dilate le gaz qui s'élève, pour retomber ensuite à cause de l'attraction gravitationnelle et de la baisse de température. C'est à ce phénomène de convection que cette région solaire doit son nom. Les mouvements convectifs engendrés par les forts gradients de densité et de température sont observables à la surface solaire sous forme de granules ou de super-granules.

Cette zone tourne sur elle-même avec une vitesse qui varie selon la latitude. Il faut aux couches supérieures du Soleil entre 25 jours à l'équateur et 33 jours près des pôles pour faire un tour complet : c'est ce que l'on appelle la rotation différentielle.

La présence d'ions lourds (e. g. carbone, oxygène, calcium, fer) rend le milieu plus opaque et les radiations ont plus de difficultés à traverser cette couche. Néanmoins, l'énergie arrivant de la zone radiative du Soleil ne mettra que deux mois à émerger. Cette opacité rend impossible les observations directes des couches internes du Soleil.

Atmosphère

L'atmosphère solaire comprend trois sous-couches présentées ci-dessous : la photosphère, la chromosphère et la couronne.

Photosphère Parce que le Soleil est une énorme boule de gaz, il n'a pas de surface solide. Cependant, il existe une profondeur à partir de laquelle le gaz n'est plus transparent aux photons : c'est ce phénomène qui donne une surface visible à notre étoile. La photosphère est définie formellement comme la région où la profondeur optique devient égale à l'unité. Cette surface représente une frontière « virtuelle » entre l'intérieur du Soleil (noyau, zone radiative et convective) et son atmosphère.

La photosphère correspond à une couche de quelques centaines de kilomètres d'épaisseur et de température moyenne d'environ $5800\ K$. Elle émet un rayonnement dont le maximum de puissance est situé dans le vert ($\lambda = 500\ nm$), mais aussi dans le bleu et le rouge et paraît ainsi blanche.

Les principales figures caractéristiques de la photosphère sont les taches solaires, les facules, les granules et les super-granules.

Les granules sont composées de jets de gaz chaud émergeant du centre et de gaz plus froid retombant sur les cotés. Elles ont une forme irrégulière et ont une taille

d'environ 1500 km. Leur temps de vie est de l'ordre de 10 minutes. Les granules se regroupent en super-granules dont la taille est de 30000 km, avec une durée de vie de plus de 24 heures. Granules et super-granules sont représentatives des cellules de convection sous-jacente.

Les taches solaires sont quant à elles des points noirs sur le Soleil associés à des régions possédant de fortes valeurs du champ magnétique (quelques milliers de Gauss). Le champ magnétique est le plus intense et presque vertical dans l'ombre des taches (au centre), et le moins intense et presque entièrement horizontal dans la pénombre (sur les extrémités). La température au centre des taches chute vers 3700 K. Elles durent en général quelques jours, bien que les plus grosses puissent être observables pendant plusieurs semaines. Les taches sont utilisées pour tracer l'activité magnétique solaire.

Le nombre de Wolf (nombre de taches et groupes de taches solaires) a permis de mettre en évidence un cycle d'activité magnétique solaire de l'ordre de 11 ans. Cette période n'est qu'une moyenne, et peut varier de 9 à 14 ans. Généralement, le nombre de Wolf augmente pendant 4 ans pour atteindre son maximum, et décroît ensuite pendant 7 ans. Si l'on prend en compte la polarité des tâches dans chaque hémisphère, c'est un cycle magnétique de 22 ans que l'on observe : une inversion est observable d'un cycle à l'autre. Ce cycle de 11 ans peut aussi être observé en étudiant les variations en latitude du nombre de tâches (diagramme papillon). Comme nous le verrons par la suite le cycle solaire a des répercutions importantes sur la composition du milieu interplanétaire.

Les facules sont des zones brillantes qui sont plus aisément observables prés du bord du disque solaire. Elles ont, comme les taches solaires, une origine magnétique mais le champ est beaucoup plus concentré. Alors que les taches solaires tendent à assombrir le Soleil, les facules l'éclaircissent : durant le maximum du cycle solaire (maximum de taches), le soleil apparaît environ $0,1$ % plus brillant que lors du minimum.

Chromosphère La chromosphère est une couche irrégulière entourant la photosphère d'une épaisseur avoisinant les 10000 km. La densité continue à y décroître rapidement alors que la température s'élève de 6000 K à la frontière avec la photosphère à plus de 20000 K à son sommet.

A ces hautes températures, l'hydrogène émet une lumière rouge que l'on peut observer dans la raie H_α à $656,3$ nm (d'où le nom de cette couche, littéralement la « sphère colorée »).

La raie H_α est la plus utilisée pour caractériser les figures principales de la chromosphère (les plages, les filaments, les protubérances, les fibrilles et les spicules) bien que le réseau chromosphérique créé par le champ magnétique soit observable dans la raie K du calcium ionisé (partie violette du spectre lumineux à $393,4$ nm).

Les plages sont des régions brillantes caractérisant les forts champs magnétiques des taches solaires et de leurs polarités magnétiques associées.

Les spicules sont de petits jets de matière dirigés de la chromosphère vers la couronne avec une vitesse de l'ordre de 20 $km.s^{-1}$, la matière éjectée retombant ensuite. La durée du phénomène est d'environ 10 minutes. Les fibrilles sont sensiblement similaires, avec une durée de vie doublée.

Les protubérances (vues au limbe) ou les filaments (vus sur le disque solaire) sont des structures magnétiques plus denses et plus froides que leur environnement. Ces structures sont principalement observées dans des raies chromosphériques mais le corps de ces structures se situe dans la couronne. Les protubérances peuvent être éruptives lorsqu'elles éjectent de la matière dans l'espace, elles ressemblent souvent à des ponts aux arches de plusieurs dizaines de milliers de kilomètres de portée. Les éruptions solaires mises à part, car moins fréquentes, les protubérances sont les plus spectaculaires phénomènes solaires.

Couronne La couronne solaire est la partie de l'atmosphère du Soleil située au-delà de la chromosphère. Elle s'étend sur plusieurs unités astronomiques (1UA = 1.510^{11} m).En se diluant dans l'espace, elle provoque le vent solaire.

On ne peut l'observer en lumière visible que pendant les éclipses totales ou à l'aide d'un coronographe, instrument inventé par Bernard Lyot en 1930, son éclat étant extrêmement plus faible que celui de la photosphère. Cependant elle est constituée de plasma (comme nous le verrons ensuite) qui est un émetteur naturel d'ondes radioélectriques allant des longueurs d'onde centimétriques ($\sim$ 10 GHz) aux longueurs d'onde métriques ($\sim$ 160 MHz) si bien que l'on peut aussi l'observer en permanence en rayonnement X ou ultra-violet.

Elle se caractérise par une température élevée et une densité faible ($\sim$ 10^{-15}).

Les premières observations du spectre visible de la couronne révélèrent des lignes d'émission brillantes à des longueurs d'ondes qui ne correspondaient à aucun matériau connu. Ces observations conduisirent les astronomes à proposer l'existence d'un nouvel élément appelé *coronium*. La composition réelle de la couronne resta un mystère jusqu'à ce qu'il soit déterminé que les gaz coronariens sont chauffés à des températures supérieures à 2.10^6 K. A ces températures, l'hydrogène et l'hélium sont complètement ionisés. Même les composants mineurs tels que le carbone, l'azote et l'oxygène ont leurs électrons arrachés. Seuls les éléments les plus lourds comme le fer et le calcium retiennent certains électrons de leurs couches internes dans cette chaleur intense. Ce sont les émissions de ces éléments très ionisés (qui forment ce que l'on appelle un plasma) qui produisent les raies d'émissions particulières qui furent si mystérieuses pour les astronomes.

Une particularité intéressante de la couronne est qu'elle est beaucoup plus chaude que la photosphère. Les mécanismes exacts qui conduisent à ce chauffage sont encore

aujourd'hui débattus, mais deux hypothèses sont actuellement retenues : un phénomène d'induction provenant du champ magnétique solaire ou un chauffage par les ondes de pression soniques sous jacentes.

Les structures caractéristiques sont variées par leurs dimensions : les boucles coronales, les sigmoïdes, les trous coronaux, les plumes, les points brillants et les *Helmet Streamers*. Les régions actives regroupent des figures à la fois photosphériques (tâches), chromosphériques (plages, filaments) et coronales (boucles, sigmoïdes).

Les boucles coronales et les sigmoïdes sont des lignes de champ magnétique reliant les deux polarités d'une région active.

Les Helmet Streamers sont des structures magnétiques de grandes dimensions ($> 2\ R_\odot$) recouvrant une région active. En période de minimum d'activité solaire, les « Helmet Streamers » sont symétriques par rapport à l'équateur. De ces régions provient le vent solaire lent.

Les plumes polaires sont des serpentins longs et fins apparaissant aux pôles et associés à des structures magnétiques ouvertes.

Les points brillants sont de petits dipôles magnétiques ayant une durée de vie limitée (entre quelques minutes et quelques heures). Ils contribuent au chauffage à micro-échelle de la couronne.

Les trous coronaux sont des zones sombres apparaissant aux pôles (parfois sur le disque) et associés à des lignes de champ magnétique ouvertes.

Des événements éruptifs sont souvent associés aux structures coronales. Les éruptions (*flares*) correspondant à des embrillancements observés dans la raie H_α se produisent dans les régions actives et se caractérisent par une réorganisation du champ magnétique et une modification des structures coronales de la région active. Les éruptions peuvent engendrer des éjections de matière coronale dans le milieu interplanétaire (*Coronal Mass Ejection*, CME).

Suivant la direction d'éjection, les particules ainsi libérées interagissent avec l'atmosphère terrestre (aurores boréales, perturbations électrostatiques).

1.1.2 Magnétosphère terrestre

La magnétosphère terrestre est une cavité naturelle dans le milieu spatial dans laquelle la Terre est relativement protégée des influences extérieures. La magnétosphère est compressée du côté faisant face au Soleil, et étirée de l'autre.

La structure complexe de la magnétosphère est le résultat des interactions entre les particules chargées provenant des couches supérieures de l'atmosphère terrestre, dont le mouvement est guidé par le champ magnétique de la Terre, et les particules du vent solaire transportant le champ magnétique interplanétaire. Une présentation détaillée de la magnétosphère est donnée dans [19].

Origine

Le champ magnétique terrestre résulte de la superposition du champ interne, du champ crustal et d'un champ magnétosphérique.

Le champ magnétique interne trouve son origine dans les courants magmatiques du noyau terrestre. Il est généralement modélisé par un développement en harmoniques sphériques dont le premier terme est celui d'un dipôle magnétique (comparable au champ d'un barreau magnétisé).

Cependant cette première approximation n'est pas tout à fait rigoureuse. Il est plus approprié de considérer que le centre du dipôle est décalé de 500 km vers le Pacifique Ouest, et que son axe est incliné par rapport à l'axe de rotation de la Terre. D'autres modèles plus réalistes basés sur des développement en séries harmoniques de Gauss existent.

A cause de ce décalage et de cette inclinaison, il existe une zone où le champ magnétique est plus faible. Cette région est située au niveau du Brésil et a pour nom l'anomalie sud-atlantique (*South Atlantic Anomaly*, SAA). Nous verrons par la suite que cette anomalie engendre une présence de radiations à une altitude moins élevée au dessus du Brésil.

Jusqu'à une altitude de 20000 km, le champ interne est la principale composante du champ magnétique terrestre.

Le champ magnétique crustal est dû à la présence de roches magnétiques dans l'écorce terrestre qui perturbent localement le champ magnétique à la surface de la Terre. L'influence du champ crustal s'affaiblit rapidement avec l'altitude.

Les champs magnétosphériques sont créés en dehors de la Terre, dans la magnétosphère. Ils résultent de l'interaction du champ magnétique terrestre avec le vent solaire et du mouvement de particules chargées (plasma) dans la magnétosphère.

Structure

Limite La nature de la magnétosphère est illustrée par l'analogie suivante. Imaginons un bateau se déplaçant sur la mer. Devant ce bateau une vague est formée : cette vague délimite la région dans laquelle le bateau perturbe l'écoulement de l'eau. L'eau derrière la vague est obligée de couler lentement autour de la coque du bateau. Derrière le bateau un sillage se forme.

L'interaction entre le vent solaire et la magnétosphère est très semblable à l'écoulement de l'eau autour du bateau. Le vent solaire s'infiltre par le champ magnétique interplanétaire. Une onde de choc est formée devant la magnétosphère de la Terre, délimitant la région où le flux du vent solaire est affecté par la présence de la Terre.

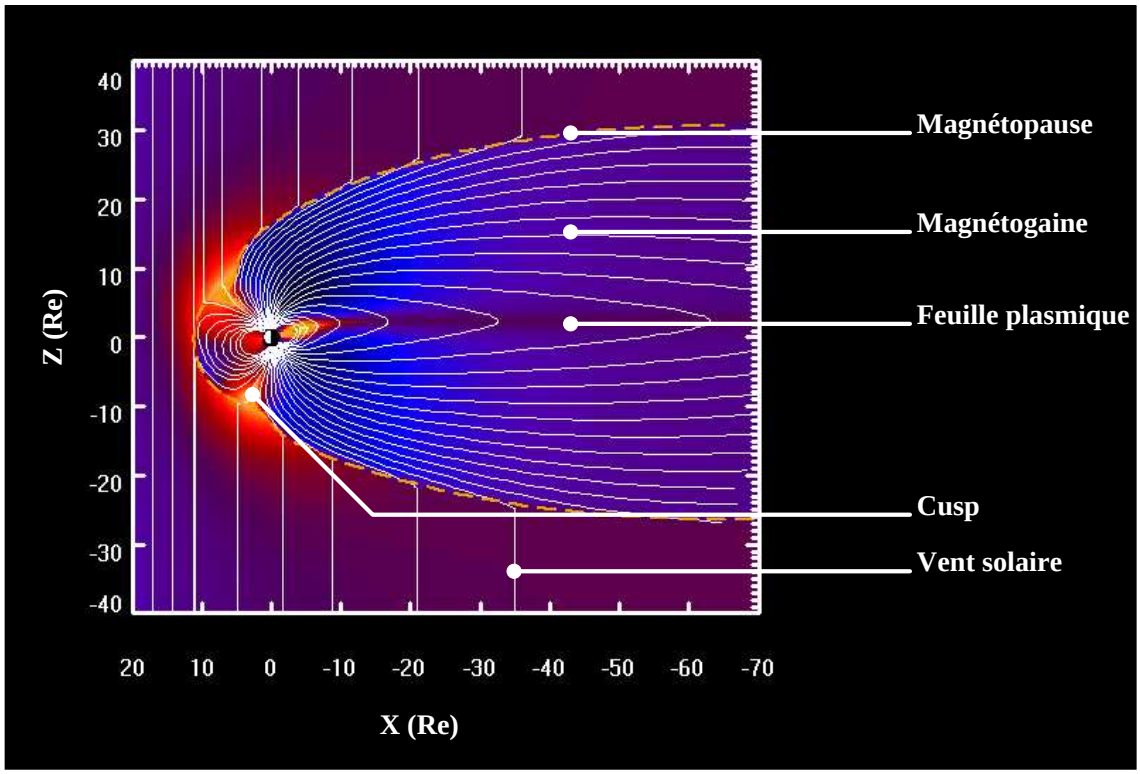

FIG. 1.2 – Effet du vent solaire sur la magnétosphère terrestre.

A hauteur de l'onde de choc, le vent solaire supersonique est ralenti et devient subsonique. Le vent solaire dans la magnétogaine, la région entre le bouclier et la magnétosphère de la Terre, est obligé de couler autour de la magnétosphère de la Terre et est comprimé.

La surface externe imperméable de la magnétosphère, où la pression totale (la somme de la pression dynamique et des pressions thermiques et magnétiques) du vent solaire comprimé équilibre précisément toute la pression à l'intérieur de la magnétosphère, s'appelle la magnétopause.

Comme représentée sur la figure 1.2, la magnétopause a une forme ovale et s'étire dans la direction opposée au Soleil, formant une longue queue, qui dans un sens est semblable au sillage derrière le bateau.

Le point subsolaire, également connu officieusement sous le nom de « nez » de la magnétopause, est normalement situé à une distance ascendante d'environ 10 rayons terrestres (R_E), alors que la distance du bouclier est en général de 15 R_E. La longueur de la queue s'exprime en centaines de R_E.

Notez les régions particulières au-dessus des pôles magnétiques (cusp), où le vent

solaire peut entrer relativement facilement dans la magnétosphère.

L'image ci-dessus est une représentation simplifiée de la réalité. En effet, la magnétosphère n'est pas une structure statique. Elle est en mouvement constant, car l'orientation du dipôle magnétique de la Terre change avec la rotation quotidienne de celle-ci et sa révolution annuelle autour du Soleil, alors que le vent solaire se caractérise par une forte variabilité temporelle sur des échelles de temps, allant de secondes à des années. Les dimensions et les formes des régions peuvent changer dans le temps à cause de la variabilité naturelle. Par exemple, lorsque la matière d'une éruption de la couronne solaire (une éjection de masse coronale) se propage à travers l'espace interplanétaire et atteint la Terre, la pression dynamique du vent solaire est fortement augmentée, de telle sorte que le bouclier et la magnétopause sont poussés vers l'intérieur, produisant un orage magnétosphérique. Durant ces orages les modifications peuvent être telles qu'un satellite en orbite géostationnaire peut se trouver temporairement en dehors de la magnétopause.

Un des effets des fluctuations mineures continues de la pression dynamique du vent solaire est le mouvement d'oscillation de la magnétopause. Des fluctuations spatio-temporelles sont également connues pour rendre celle-ci semi-perméable, et permettre au plasma de la magnétogaine de la traverser, formant donc la couche limite magnétosphérique.

Intérieur Tandis que le comportement du plasma dans les régions externes de la magnétosphère est dominé par les conditions du vent solaire, l'intérieur de la magnétosphère est fortement lié à l'ionosphère de la Terre. On peut facilement comprendre que la région interne, appelée plasmasphère, qui se compose d'un plasma dense et froid d'origine principalement ionosphérique, doit plus ou moins tourner avec la Terre. La plasmasphère est située sur les lignes terrestres fermées du champ magnétique. Sa limite externe s'appelle la plasmapause. La *plasmathrough* est située en dehors de la plasmapause ; elle se trouve également sur les lignes fermées du champ. Elle est très fine et ne tourne pas avec la Terre.

La queue de la magnétosphère est principalement constituée de deux lobes, de polarité magnétique opposée. Chaque lobe est l'extension magnétosphérique des lignes du champ magnétique provenant de l'ionosphère polaire. Les lobes sont séparés par le feuillet plasmique qui contient le plasma chaud. Le feuillet plasmique est une région très dynamique. Les changements dans le champ magnétique interplanétaire peuvent déclencher un comportement instable dans cette région (selon un processus connu sous le nom d'orage magnétosphérique). Le feuillet de plasma est situé sur les lignes du champ magnétique qui ont leur point d'ancrage ionosphérique dans l'ovale auroral ; les particules chaudes du feuillet plasmique sont donc responsables de l'apparition des aurores.

Les Ceintures de Radiation Les ceintures de radiation ou ceintures de Van Allen, sont situées au sein de la magnétosphère. Elles sont donc situées en dehors de l'atmosphère terrestre mais toujours dans la zone de l'espace influencée par le champ magnétique de la Terre. Elles sont constituées d'électrons et d'ions (principalement des protons) très énergétiques dont le mouvement autour de la Terre est fortement contraint par le champ magnétique terrestre.

Toute particule chargée en présence d'un champ électromagnétique est soumise à la force de Lorentz. Si le champ magnétique est grand et que l'énergie des particules est élevée (et par conséquent leur vitesse), le champ électrique peut être ignoré et la force de Lorentz se réduit à

$$\vec{F} \approx q(\vec{v} \wedge \vec{B})$$

Dans ces conditions, le mouvement peut se décomposer en trois composantes :

- Un mouvement rapide de rotation générant une trajectoire spiralée (*Gyration*). La particule chargée tourne autour des lignes de champ ;
- Un mouvement de va-et-vient entre les deux hémisphères (*Bounce*). Afin de conserver son moment magnétique constant, une particule partant de l'équateur vers l'un des pôles voit la composante radiale de la vitesse augmenter jusqu'à annulation de la composante tangentielle. La particule atteint alors un point miroir et un léger gradient du champ magnétique lui permet de repartir vers l'autre hémisphère ;
- Un mouvement lent de dérive autour de la Terre (*Drift*). Ce mouvement est dû à la nature radiale du gradient du champ magnétique.

Une particule chargée soumise à ces 3 mouvements périodiques va se mouvoir sur une surface de forme toroïdale appelés *drift shells* et ainsi être piégée. La figure 1.3 illustre la trajectoire compliquée suivie par les particules des ceintures de Van Allen. Les particules confinées dans ce type de trajectoire peuvent y rester durant plusieurs années d'où le terme particules piégées.

1.1.3 Rayons Cosmiques

Ce rayonnement est issu de sources présentes dans notre galaxie ainsi qu'à l' extérieur. Les interactions avec la matière interstellaire, les ondes de choc ainsi que les champs électromagnétiques dispersent et accélèrent ce rayonnement. De ce fait, à l'échelle de notre système solaire, il apparaît avec une distribution angulaire isotropique et les particules qui le constituent sont complètement ionisées. Les rayons cosmiques (*Galactic Cosmic Rays*, GCR) constituent un bruit de fond constant.

On y trouve des protons ainsi que tous les ions lourds. Leurs énergies sont très hautes et peuvent atteindre plusieurs GeV. Dans le système solaire, les flux caractéristiques de cette population sont modulés par le cycle d'activité solaire : lorsque le

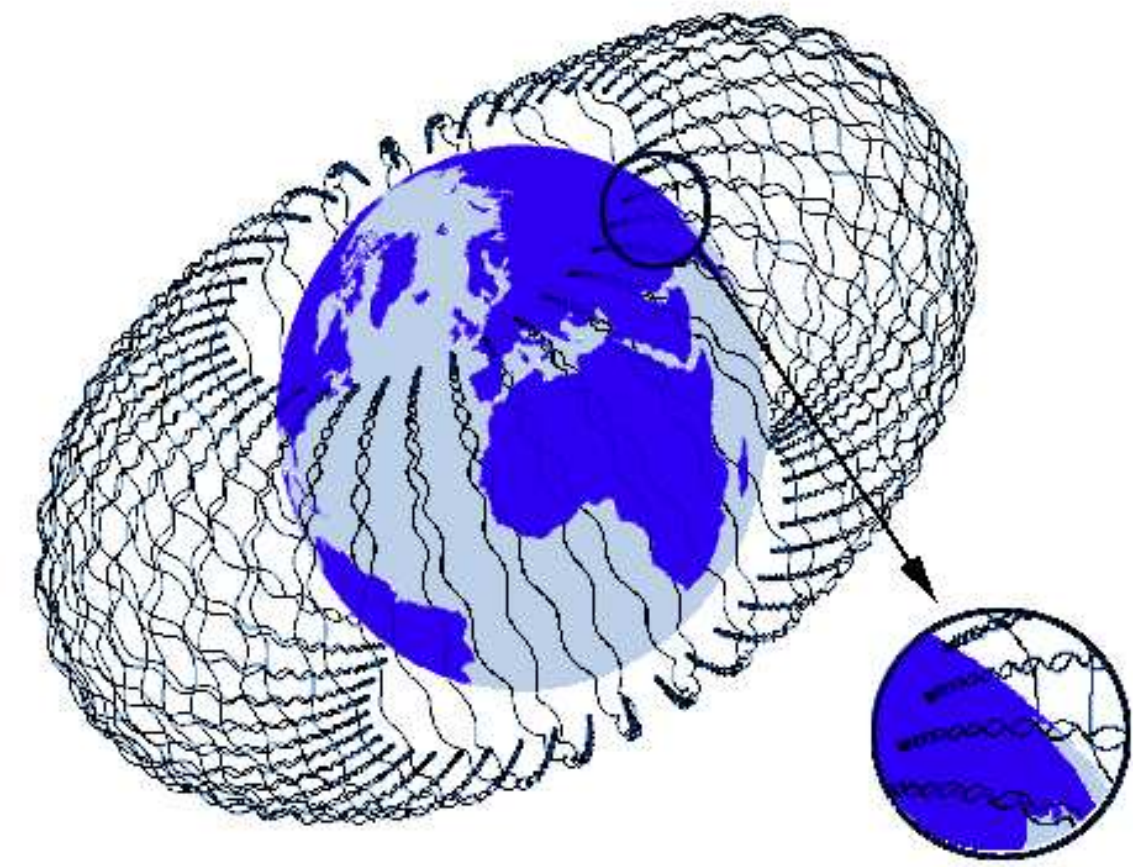

FIG. 1.3 – Exemple de trajectoire des particules piégées dans les ceintures de radiation.

soleil est en période d'activité maximale, le vent solaire s'oppose au flux de rayons cosmiques. Les populations de particules d'origine cosmique qui peuplent le système solaire sont donc réduites durant ces périodes.

1.2 Environnements Radiatifs

Suivant leur origine, on distingue plusieurs types de radiations :

- Les radiations transitoires issues des rayons cosmiques, ou d'événements solaires ;
- Les radiations piégées dans certaines zones de la magnétosphère terrestre ;
- Les radiations présentes dans les couches de l'atmosphère ;
- Les radiations au niveau du sol.

Les quatre sections suivantes introduisent plus en détail chaque zone. De plus amples détails sur l'environnement spatial sont donnés dans [20] et [19]. Le lecteur plus interessé par l'atmosphère terrestre pourra consulter [8] et [21].

1.2.1 L'Héliosphère

La composition de ce milieu est dictée par deux populations : le vent solaire (ainsi que les événements solaires tels que les CME et les flares) et les rayons cosmiques.

La couronne, selon son activité et la topologie de son champ magnétique, régule les caractéristiques du flux de particules qui s'échappent du soleil. Durant les périodes calmes du cycle solaire, deux composantes principales peuvent être observées : Une « lente » dont la vitesse est de l'ordre de 350 km/s avec une densité au niveau de la Terre de 10 protons par cm^3, et une « rapide » dont la vitesse est d'environ 700 km/s pour une densité de 3 protons par cm^3. La première vient de la couche magnétique neutre de la couronne, et la seconde des trous coronariens.

Lorsque le vent solaire s'échappe de la couronne, il emporte avec lui les lignes de champ magnétique. Parce qu'il quitte le Soleil dans des directions radiales, et du fait que le Soleil tourne d'environ 13.5° par jour, les lignes de champs qui restent ancrées au Soleil acquièrent une forme de spirale à l'autre bout.

Ce sont ces lignes de champ qui vont moduler les flux de rayons cosmiques. Leur composition approximative est de 83 % de protons, 13 % d'ion ^4He, 3 % d'électrons et 1 % d'ions plus lourds. Deux zones d'enrichissement en éléments légers (Li, Be, B) et moyens (Se, V, Cr, Mn) peuvent être observées. Elles sont attribuées aux interactions nucléaires avec la matière stellaire.

Evènements Solaires

Cette population est générée lors d'événements solaires tels que les CME et flares. On y trouve des protons, des particules α ainsi que des ions lourds et des électrons. Ces événements sont imprévisibles.

1.2.2 Ceintures de radiations

Ces radiations sont piégées dans les ceintures de Van Allen. On y trouve des protons et des électrons principalement, ainsi que les éléments les plus légers de la table de classification périodique des éléments.

La population de particules piégées est constituée principalement de protons d' énergies comprises entre 100 KeV et plusieurs centaines de MeV et d' électrons d'énergie comprises entre quelques dizaines d'eV et 10 MeV.

Des indices sembleraient indiquer la présence d'une fine région aux alentours d'un rayon terrestre d'altitude peuplée d'ions lourds pouvant venir de la décélération de rayons cosmiques anormaux (atomes non-ionisés arrivants dans le système solaires et qui sont ionisés par la suite par effet photo-électriques ou par collision). Cependant l'intensité de cette population est bien moindre que celle des protons et électrons piégés, de sorte que ces ions ne sont généralement pas pris en compte.

Les compositions et localisations des particules piégées ne sont pas statiques, elles résultent d'un équilibre entre :

- Les mécanismes de création : injection venant de la queue de la magnétosphère et réactions nucléaires entre la haute atmosphère et les ions à haute énergie (solaire ou cosmiques) ;
- Les mécanismes de perte : précipitation dans la haute atmosphère et échanges de charges avec les composants de l'exosphère.

Protons

Les protons sont répartis sur une ceinture unique avec un maximum de densité à $L = 1.7$ pour les protons de 10 MeV ou plus. Les flux sont relativement stables. La répartition des protons dans la ceinture est représentée sur la figure 1.4.

Electrons

Les électrons sont répartis de manière plus complexe sur deux ceintures :

- L'intérieure. Elle est centrée sur $L = 1.4$ et s'étend jusqu'à $L = 2.8$. La population est stable et l'énergie des particules peut atteindre 30 MeV ;
- L'extérieure. Elle est centrée sur $L = 5$ et s'étend de $L = 2.8$ à $L = 10$. Les densités sont très variables et les énergies peuvent atteindre les 7 MeV.

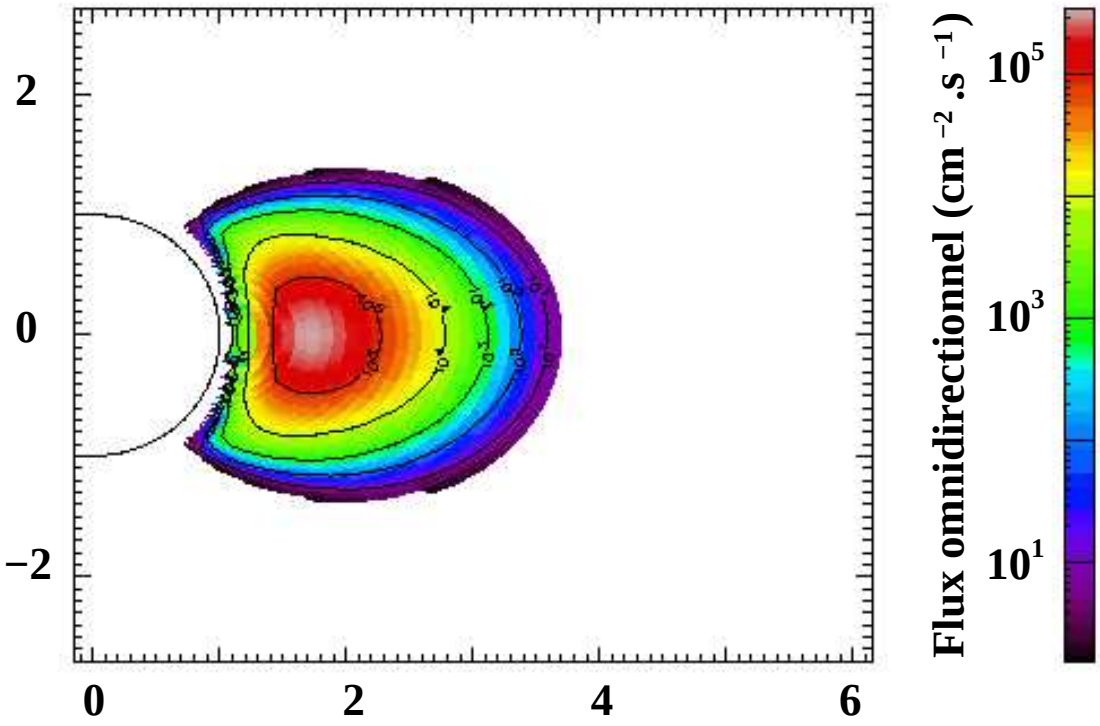

FIG. 1.4 – Répartition des protons ($E > 10\ MeV$) dans les ceintures (Modèle AP8-MAX). Abscisses et ordonnées sont exprimées en rayons terrestres.

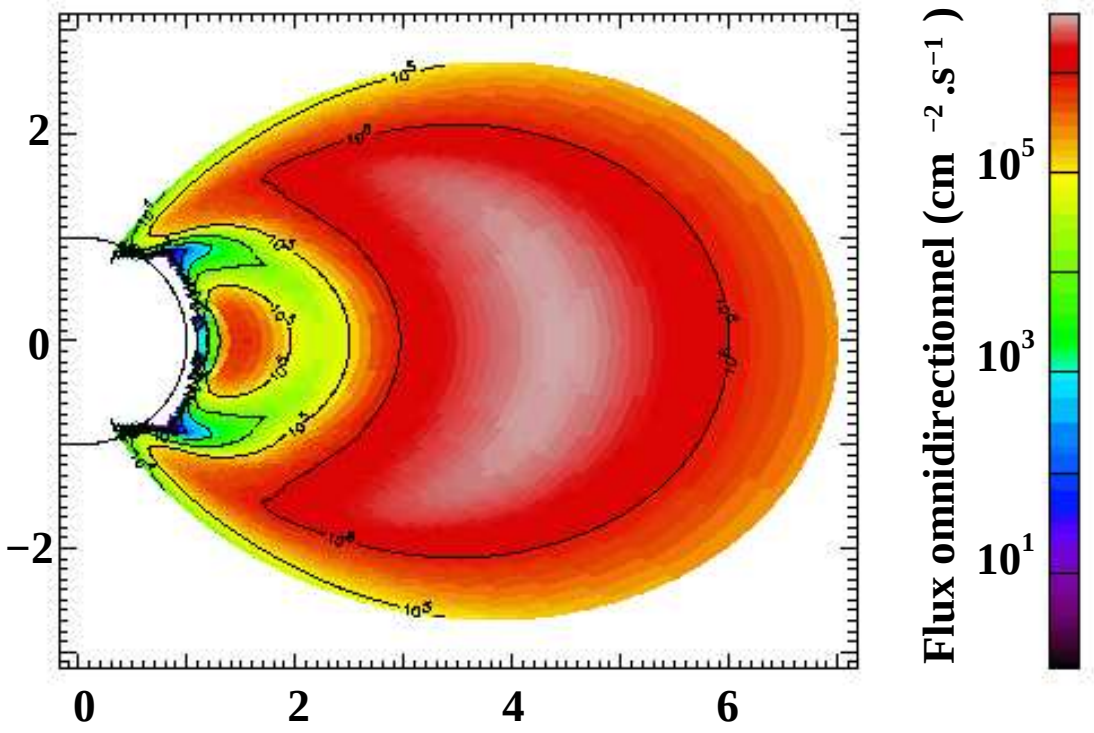

FIG. 1.5 – Répartition des électrons ($E > 1\ MeV$) dans les ceintures (Modèle AE8-MAX). Abscisses et ordonnées sont exprimées en rayons terrestres.

La répartition des électrons dans les deux ceintures est représentée sur la figure 1.5.

1.2.3 Pôles et Anomalie Sud-Atlantique

Nous avons vu dans la section 1.1.2 qu'à cause des particularités du dipôle magnétique terrestre, les ceintures de radiations sont plus basses au niveau du Brésil. Un satellite en orbite basse (*Low Earth Orbit*, LEO) verra une asymétrie dans l'exposition aux radiations qu'il recevra. De même, la position de Kourou fait que les lanceurs doivent être préparés à cette exposition.

De la même façon, les zones moins protégées au dessus des pôles (cusp) induisent une présence élevée d'ions d'origine cosmique, de protons (pour une altitude supérieure à 1000 km) et d'électrons. Un satellite, toujours en orbite LEO, devra aussi être préparé à cette exposition.

A titre d'exemple la figure 1.6 montre la position du satellite UOSAT-3 (780 km d'altitude) pour chaque erreur détectée dans sa mémoire. On reconnaît clairement l'anomalie sud-atlantique et les zones polaires.

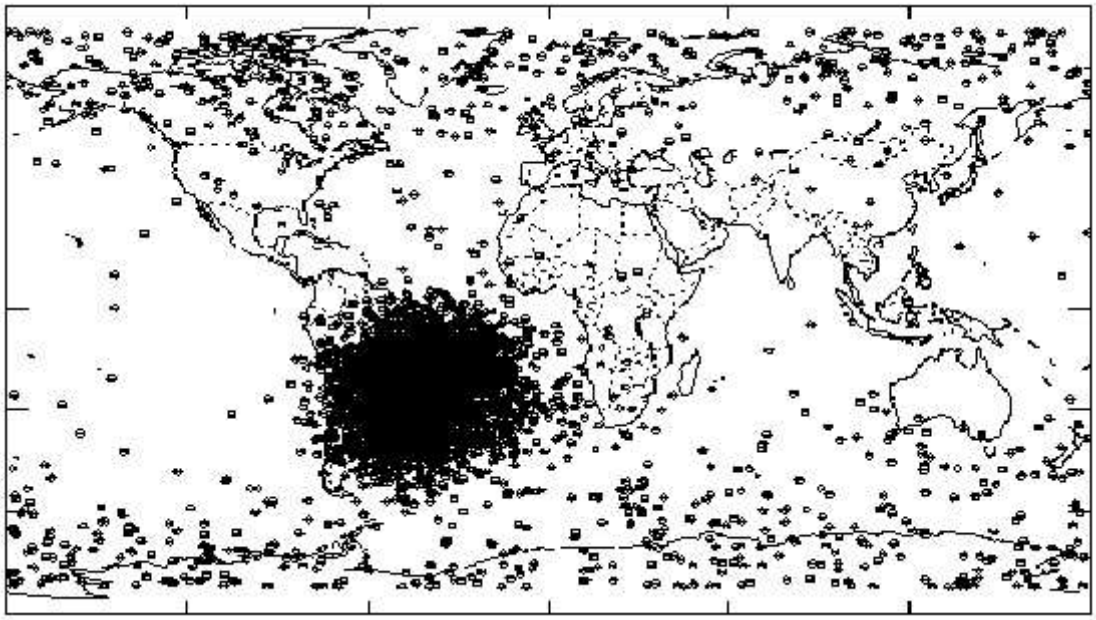

FIG. 1.6 – Position de UOSAT-3 lors de la détection d'erreurs dans sa mémoire.

1.2.4 Atmosphère

Les rayons cosmiques et les protons pénètrent l'atmosphère et interagissent avec l'azote et l'oxygène de l'atmosphère pour créer une douche de particules secondaires (protons, électrons, neutrons, ions lourds, muons et pions). Les neutrons sont les hadrons que l'on retrouve en plus grande quantité : ils apparaissent à 330 km d'altitude et leur densité croit jusqu'à environ 20 km, pour réduire jusqu'au sol où elle n'est plus que 1/500 du flux maximum. Le mécanisme de douche est illustré sur la figure 1.7.

Au sol les radiations d'origine cosmiques sont issues de particules de 6ème ou 7ème génération. La population a une direction hautement verticale. De par leur origine, elles sont soumises au cycle d'activité solaire. Cependant, les événements solaires peuvent augmenter localement ce flux de plus de 5000 %.

Le flux de neutrons au sol est théoriquement constitué de trois pics distincts en énergie.

Le premier, entre 10 MeV et 1000 MeV contient les rémanents des particules de haute énergie. Lorsque les neutrons ralentissent, ils sont plus facilement absorbés, entrainent une réduction de la population sous 20 MeV. Ce pic a un flux de l'ordre de 12 neutrons $/cm^2.hr$.

Lorsque des neutrons énergétiques subissent des réactions dites de spallation,

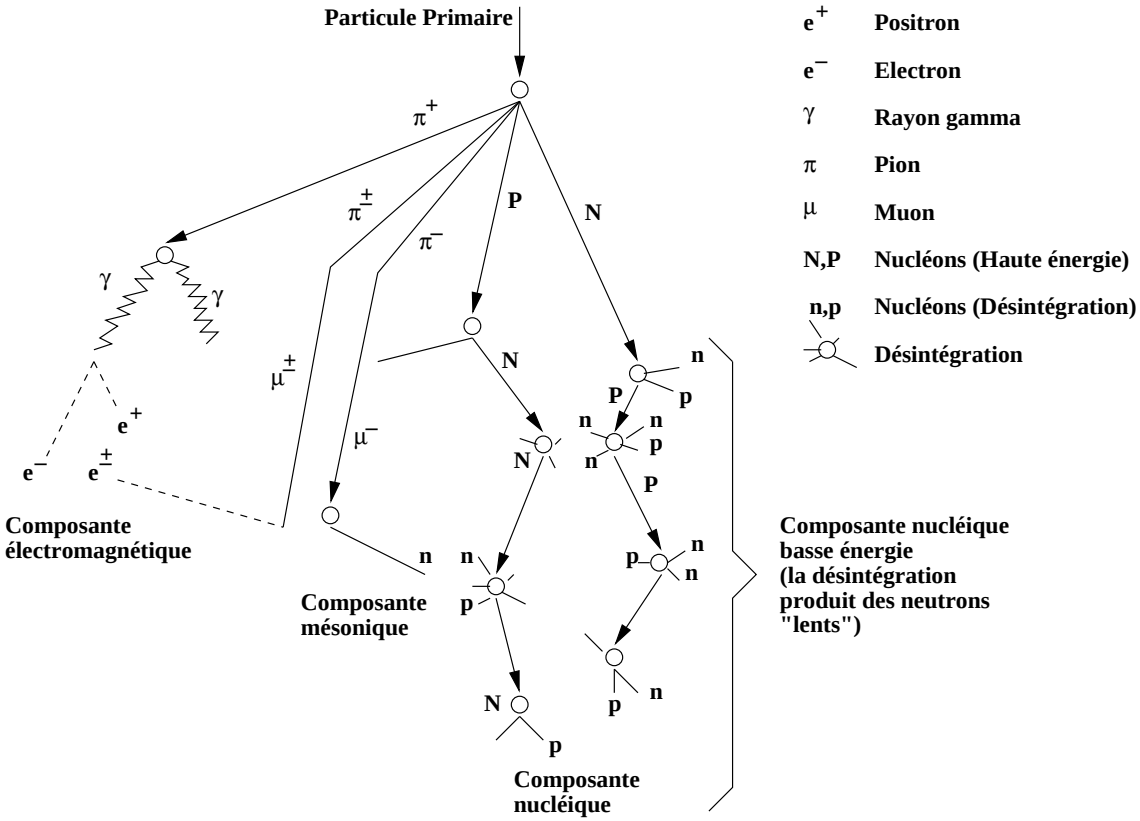

FIG. 1.7 – Illustrations des réactions de cascades possibles lorsque les rayons cosmiques pénetrent l'atmosphère (adapté de [8]).

ils produisent à leur tour un flux de neutrons secondaires de moindre énergie, aux alentours de quelques MeV. Associé à ce phénomène, il existe un seuil de réaction nucléaire autour de 5 MeV : beaucoup d'atomes ne réagissent plus avec les neutrons en dessous de ce seuil. De ce fait, les neutrons qui atteignent la barre des 5 MeV ont une durée de vie bien plus longue. Ces deux mécanismes associés font que l'on trouve un second pic entre 0.1 MeV et 5 MeV. Ce pic a un flux d'environ 20 neutrons $/cm^2.hr$.

Finalement les neutrons continuent à ralentir jusqu'à atteindre la vitesse des atomes à température ambiante (les neutrons « thermiques », à 25 meV). Ces neutrons ne peuvent plus perdre d'énergie et ils forment le troisième et dernier pic avec un flux d'environ 4 neutrons $/cm^2.hr$.

Théoriquement le flux de neutrons au sol devrait être de l'ordre de 36 particules $/cm^2.hr$, mais plusieurs facteurs le modifient :

- L'altitude. La population croit avec l'altitude. A 3700 m, le flux est multiplié par un facteur 13.
- La latitude géomagnétique ;
- Le cycle solaire ;
- Le blindage environnant. Jusque dans les années 90, seuls les neutrons d'énergie supérieure à 100 MeV étaient une menace pour les circuits intégrés. Pour ces neutrons, l'atténuation est exponentielle et se calcule aisément. Mais depuis les dix dernières années, les neutrons d'énergie inférieure à 100 MeV posent eux aussi des problèmes (ce phénomène est la conséquence de la réduction de la finesse de gravure). Ces particules sont souvent le résultat des étages finaux de la douche de particule, de sorte que le bureau, la chaise ou même le corps humain peuvent en être la source.

1.3 Effets des Radiations sur les Circuits Intégrés

Les radiations peuvent causer des effets transitoires ou des dommages permanents dans les matériaux qu'elles traversent. Cette section a pour but de présenter les mécanismes d'interaction radiation-matière. Suivant l'interaction, deux types d'effets se dégagent : les effets de dose (qu'elle soit ionisante ou non) et les événements dits singuliers (causé par une seule et unique particule). Ces deux effets feront respectivement l'objet d'une sous-section.

1.3.1 Mécanismes d'interactions

Les dommages, qu'ils soient permanents ou transitoires, sont très souvent proportionnels à la quantité d'énergie perdue par la particule incidente dans la matière

qu'elle traverse. Cette quantité s'appelle le pouvoir d'arrêt total dE/dx et exprime la quantité d'énergie perdue par unité longueur de matériau traversé. Le pouvoir d'arrêt total caractérise la façon dont la particule incidente est ralentie puis absorbée.

La matière étant constituée d'électrons et de noyaux, une distinction doit être faite selon que la particule incidente interagit avec l'un ou l'autre.

Le pouvoir d'arrêt total peut être scindé en trois composantes suivant la nature du transfert énergétique (par collision, perte radiative ou réaction nucléaire) :

$$\frac{dE}{dx}\Big)_{total} = \frac{dE}{dx}\Big)_{col} + \frac{dE}{dx}\Big)_{rad} + \frac{dE}{dx}\Big)_{nuc}$$

où $(dE/dx)_{col}$ est appelé pouvoir d'arrêt par collision avec le nuage électronique cible et caractérise les pertes par interaction Coulombienne (ionisation et excitation), $(dE/dx)_{rad}$ est le pouvoir d'arrêt radiatif qui caractérise les pertes de type Bremsstrahlung et Cerenkov et $(dE/dx)_{nuc}$ le pouvoir d'arrêt qui caractérise les réactions nucléaires.

L'interaction avec un électron peut conduire ce dernier à quitter son orbitale atomique, avec possibilité de création de paires électrons / trous dans les matériaux semiconducteurs. Parfois un électron éjecté peut à son tour produire une ionisation. Ce phénomène s'appelle une production de rayons delta.

L'interaction entre une particule chargée et le champ électrique des atomes conduit à l'émission de radiations γ. Ce flot de photons, appelé Bremsstrahlung peut à son tour ioniser le milieu. L'énergie associée au Bremsstrahlung est proportionnelle à $1/m^2$, de ce fait ce sont les particules légères, notamment les électrons, qui sont assujéties à ce type de phénomène. Le rayonnement Cerenkov se produit lorsqu'une particule traverse un milieu avec une vitesse supérieure à celle qu'aurait la lumière dans ce milieu. En général il y'a émission de photons ayant un spectre continu en fréquence.

Le résultat de l'interaction avec un noyau peut conduire à un transfert d'énergie avec comme conséquence possible l'éjection de ce dernier du réseau cristallin. Des défauts interstitiels sont alors crées. La perte d'énergie associée à ce type de dommage est appelée perte d'énergie non ionisante (*Non Ionising Energy Loss*, NIEL) :

$$NIEL = \frac{dE}{dx}\Big)_{nuc}$$

lorsque l'on néglige les phonons.

Dans la communauté spatiale, pour qualifier l'ionisation du milieu, on utilise le terme transfert linéique d'énergie (*Linear Energy Transfert*, LET). Le LET ou pouvoir d'arrêt par collision restreint (*restricted collision stopping power*) est défini

de manière formelle comme le pouvoir d'arrêt par collision lorsque la perte d'énergie par collision est considérée **locale**, c'est à dire que l'on ne prend pas en compte l'ionisation du milieu par les rayons delta.

Un autre type d'interaction est la réaction nucléaire de spallation : la particule incidente disparaît lors de l'interaction avec un atome du milieu pour créer une ou plusieurs particules filles qui, à leur tour, vont être ralenties, absorbées et / ou subir d'autres réactions nucléaires de spallation.

1.3.2 Dose

Une grande partie des effets des radiations que l'on peut observer sur les satellites est regroupée dans ce qui est appelé les effets de dose. En général, lorsque l'on parle de dose, c'est à la dose ionisante que l'on se réfère. La dose absorbée s'exprime en Gray (Gy). Un Gray correspond à l'absorption d'un Joule par kilogramme de matière.

Pour un échantillon de matière de surface S, d'épaisseur dx, de masse volumique ρ, irradié par une fluence de particules Φ déposant une énergie dE par ionisation, la dose s'exprime :

$$\begin{aligned} D_i &= N_{particules}\frac{dE}{masse} \\ &= \Phi.S.\frac{dE}{S.dx.\rho} \\ &= \Phi\frac{1}{\rho}\left(\frac{dE}{dx}\right)_{col} + \frac{dE}{dx}\Big)_{rad} \end{aligned}$$

Par analogie, on définit la dose non ionisante par :

$$D_{ni} = \Phi.NIEL$$

Afin de compléter le vocabulaire, le taux de dose ($Gy.s^{-1}$) est défini par :

$$D' = \frac{dD}{dt}$$

Dose Ionisante

Nous avons vu précédemment que lors d'une irradiation, l'énergie perdue par les particules incidentes peut être cédée aux électrons du matériau traversé. Selon le matériau, ces électrons peuvent en retour atteindre la bande de conduction et libérer un trou dans la bande de valence. Les références [22] et [23] décrivent de manière

détaillée les interactions rayonnement ionisant - matière.

On considère qu'il faut en moyenne 3.6 eV pour faire migrer un électron de la bande de valence à la bande de conduction dans du silicium. Dans l'oxyde de silicium, cette valeur devient 18 eV. Pour fixer un ordre de grandeur, 1 Gy dans 1 cm^3 de silicium génère 7.3 10^{14} paires d'électrons / trous et 1.5 10^{14} paires dans l'oxyde de silicium (SiO_2).

Comparés aux densités typiques des métaux, ces porteurs sont négligeables et vont se recombiner très rapidement. Pour du silicium non-dopé, ou très légèrement (*bulk*), la génération de charge est sensiblement équivalente à la densité de porteurs de charges à l'équilibre. L'effet sera donc transitoire et disparaîtra avec l'irradiation. L'effet est plus dramatique dans les isolants : la charge générée ne peut migrer à cause de la très faible mobilité et reste donc piégée sur des niveaux parasites introduits par les impuretés ou les défauts dans le large gap.

La dégradation induite par les effets de dose ionisante est l'effet de plusieurs mécanismes que nous allons présenter rapidement par la suite :

- La génération de charge, que nous avons cité précédemment ;
- La recombinaison initiale ;
- Le transport des porteurs excédentaires ;
- Le piégeage des trous ;
- La formation d'états d'interface.

Recombinaison Dès que les paires sont générées dans l'oxyde, les électrons évacuent l'oxyde (meilleure mobilité). Cependant une certaine fraction a le temps de se recombiner avec les trous. La fraction de trous qui échappe à la recombinaison dépend de deux facteurs :

- Le champ électrique qui sépare les paires ;
- La densité de la colonne de charge excédentaires : plus les composants de la paire sont proches, plus la probabilité qu'ils se recombinent est grande.

Transport des trous Les trous qui échappent à la recombinaison sont soumis à un transport très lent au travers de l'oxyde. L'hypothèse la plus probable pour expliquer ce phénomène est un transport via des niveaux permis profonds dans le gap de l'oxyde sous l'action du champ électrique.

Piégeage L'interface Si/SiO_2 n'est pas « nette ». Parfois un atome d'oxygène manquant laisse une liaison $Si-Si$ faible qui constitue un piège à trous. La fraction de trous qui peuvent être piégés à cette interface varie de 1 % à 80 %. Les électrons sont censés avoir quitté l'oxyde lors de la génération. Une faible fraction est tout de

même piégée dans cette interface peu de temps après, mais elle ne compense pas la charge positive des trous piégés.

La population de trous piégés ne l'est pas *ad vitam eternam*. Les trous disparaissent sur une échelle de temps qui s'étale de la microseconde à plusieurs années grâce à deux phénomènes :

- Par effet tunnel : des électrons peuvent rejoindre les trous piégés et se recombiner ;
- Par excitation thermique : les trous sont excités de manière à rejoindre la bande de valence.

Formation d'états d'interface Plusieurs modèles ont été proposés pour expliquer la formation de ces états, et la controverse continue. Il est tout de même admis que le précurseur de cet état est un atome de silicium lié à 3 autres ainsi qu'un atome d'hydrogène. Lorsque le lien $SI - H$ est brisé, il laisse une *liaison pendante* (tout comme l'orbitale atomique non saturée à la surface d'un semiconducteur qui conduit à ce qu'on appelle des états de surface).

Dans tous les cas, ces états peuvent se charger suivant la tension appliquée (négativement pour les MOSFETs de type N sous une tension positive, positivement pour les MOSFETs de type P polarisés négativement).

Effets sur les transistors MOS Les trous piégés dans l'oxyde induisent une charge négative sous la grille. La création du canal en sera facilitée pour les transistors NMOS, entrainant une diminution de la tension seuil. Pour les transistors PMOS, on aboutit à une augmentation de la tension seuil. La figure 1.8 résume les mécanismes participants à la réponse d'une structure MOS en présence de rayonnements ionisants.

Les états d'interface modifient la mobilité dans le canal, la transconductance et la vitesse de recombinaison. Lorsqu'ils sont chargés (positivement pour les transistors PMOS), ils contribuent à l'élévation de la tension seuil. Pour les NMOS (dont les états se chargent négativement), ils compensent l'abaissement de la tension seuil.

Effets sur les transistors Bipolaires Les effets de la dose ionisante induisent une dégradation du gain en courant : le courant de base augmente (à cause des recombinaisons due aux états d'interface) alors que le courant du collecteur reste relativement constant.

Dose Non Ionisante

L'introduction de défaut dans le réseau cristallin engendre des niveaux d'énergie permis dans le gap du semiconducteur. Ces niveaux parasites ont 5 types de contribution :

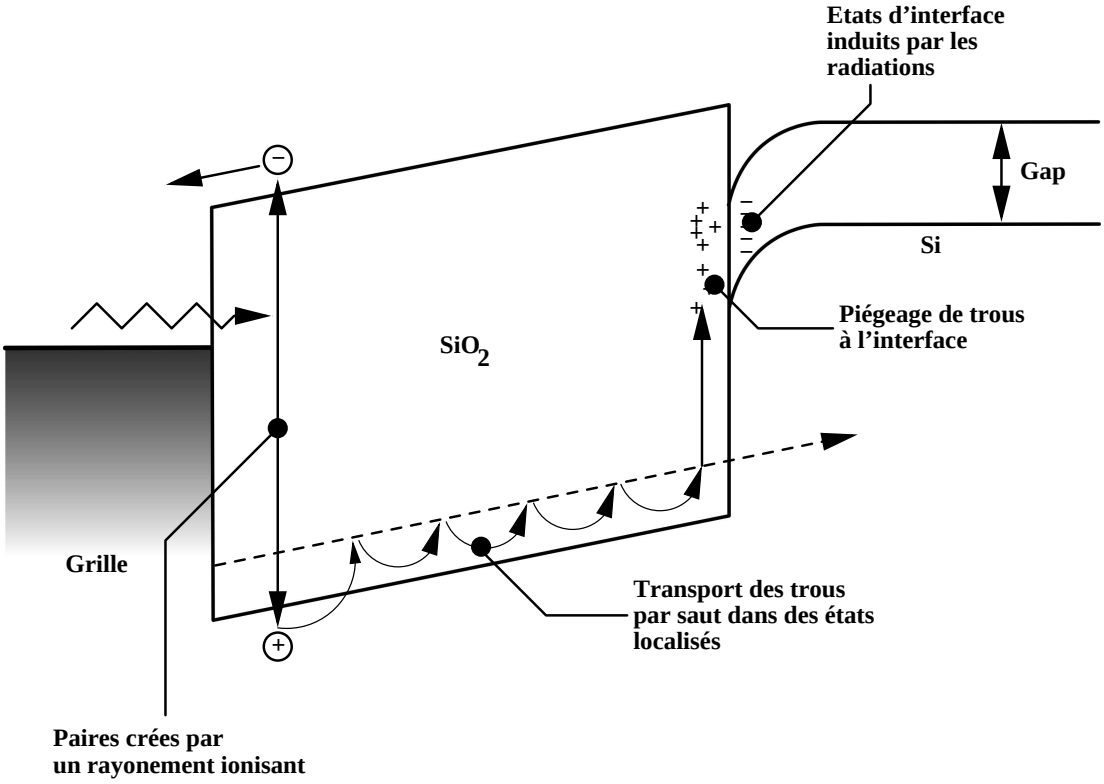

FIG. 1.8 – Diagramme en bandes illustrant les processus physiques mis en jeu lors de l'irradiation d'une structure MOS (d'aprés [23]).

- Génération de paires, et donc augmentation des courants de fuite ;
- Recombinaison de paires, et donc réduction de la durée de vie des porteurs minoritaires ;
- Piégeage de porteurs ;
- Compensation des dopants ;
- Effet tunnel.

Il faut noter que ce sont surtout les composants opto-électroniques et les transistors bipolaires qui sont sensibles à ce type de dommage.

1.3.3 Evénements Singuliers

Dans cette section, nous présentons les événements singuliers (*Single Event Effects*, SEE). Même si l'origine du phénomène est la même que pour la dose ionisante, c'est à dire la création de paires de porteurs, les conséquences, et donc le traitement sont différentes : les SEE sont caractérisés par une injection de charge **élevée** et **localisée** dans l'espace et le temps.

Les événements singuliers résultent principalement de la déposition et de la collection de charges à un noeud sensible du circuit. Ce noeud sensible est une jonction en polarisation inverse : à l'intérieur de la zone de déplétion, le champ électrique permet une collection efficace des charges excédentaires. Pour simplifier les calculs ce noeud est modélisé par un parallélépipède rectangulaire (*Rectangular Parallele-Piped*, RPP) [24].

Les ions énergétiques qui traversent ce volume produisent *directement* une colonne d'électrons-trous qui peuvent provoquer un SEE.

Les protons et les neutrons transfèrent une partie ou la totalité de leur énergie lors de collisions élastiques / inélastiques. Les sous produits de ces collisions peuvent à leur tour ioniser la matière et déclencher un SEE, à la manière d'un ion. C'est un mécanisme *indirect*.

Mécanismes

Le déclenchement d'un SEE se passe en deux temps : une période de génération de charge, et une période de collection de charge. Plus de détails sont fournis dans [25] et [26].

Génération De même que pour la dose ionisante, on utilise le LET pour quantifier la charge déposée par le passage de l'ion. La courbe qui trace le LET d'une particule en fonction de sa distance de pénétration dans la matière est appelée *courbe de Bragg*.

Sur la majeure partie du trajet de la particule dans la matière, le LET reste relativement constant (et donc l'ionisation provoquée). Vers la fin du trajet, lorsque la particule a perdu une grande partie de son énergie, le LET augmente subitement pour s'annuler brutalement lorsque la particule incidente est au repos. Ce pic est appelé le *pic de Bragg*.

En première approximation, on peut calculer l'énergie déposée dans un volume de densité ρ, connaissant la longueur du trajet d de la particule :

$$E_{dep} \approx LET \times d \times \rho$$

Pour fixer les idées, une particule avec un LET de 97 $MeV.cm^2/mg$ dépose environ 1 pC par μm de silicium parcouru.

Ces calculs sont basés sur les approximations suivantes :
- Le LET de l'ion est constant dans le volume considéré ;
- Il n'y a pas de dépendance envers la trajectoire de la particule.

Ce type d'approximation est beaucoup moins bien justifié dans le cas d'ionisation indirecte : les produits des réactions nucléaires ont une énergie et une pénétration relativement faible.

Collection Afin d'illustrer le mécanisme de collection, nous allons décrire le cas d'école : une ionisation directe dans la zone de déplétion d'une jonction P-N polarisée (drain d'un transistor N dans l'état bloqué) en inverse. C'est un pire cas car :
- Les électrons ont une mobilité supérieure à celle des trous ;
- La zone de déplétion est étendue ;
- La zone de déplétion est désertée, de ce fait le profil de création de charge est net.

Lors du passage de la particule, deux phénomènes sur des échelles de temps différentes participent à la collection de charges :
- Conduction (*drift*). Les électrons, sous l'influence du champ électrique régnant dans la zone de déplétion, vont se déplacer vers le drain du transistor N. Ce courant est responsable de la réponse initiale du noeud touché, et la durée du phénomène est inférieure à la nanoseconde ;
- Diffusion. Les électrons générés dans le substrat peuvent diffuser vers le drain et ainsi être collectés. La diffusion s'exerce sur de plus longues distances et peut s'étendre sur plusieurs nanosecondes.

Un troisième phénomène peut aussi rentrer en jeu : le *funneling*. Lorsque l'ion sort de la zone de déplétion, il distord le champ électrique et l'étire en dehors de la zone de déplétion proprement dite : des électrons générés en dehors de la zone de déplétion, mais proches d'elle peuvent de ce fait être collectés par le champ électrique plutôt que par diffusion. La longueur de funneling n'est cependant pas toujours bien définie. La figure 1.9 illustre ces trois mécanismes.

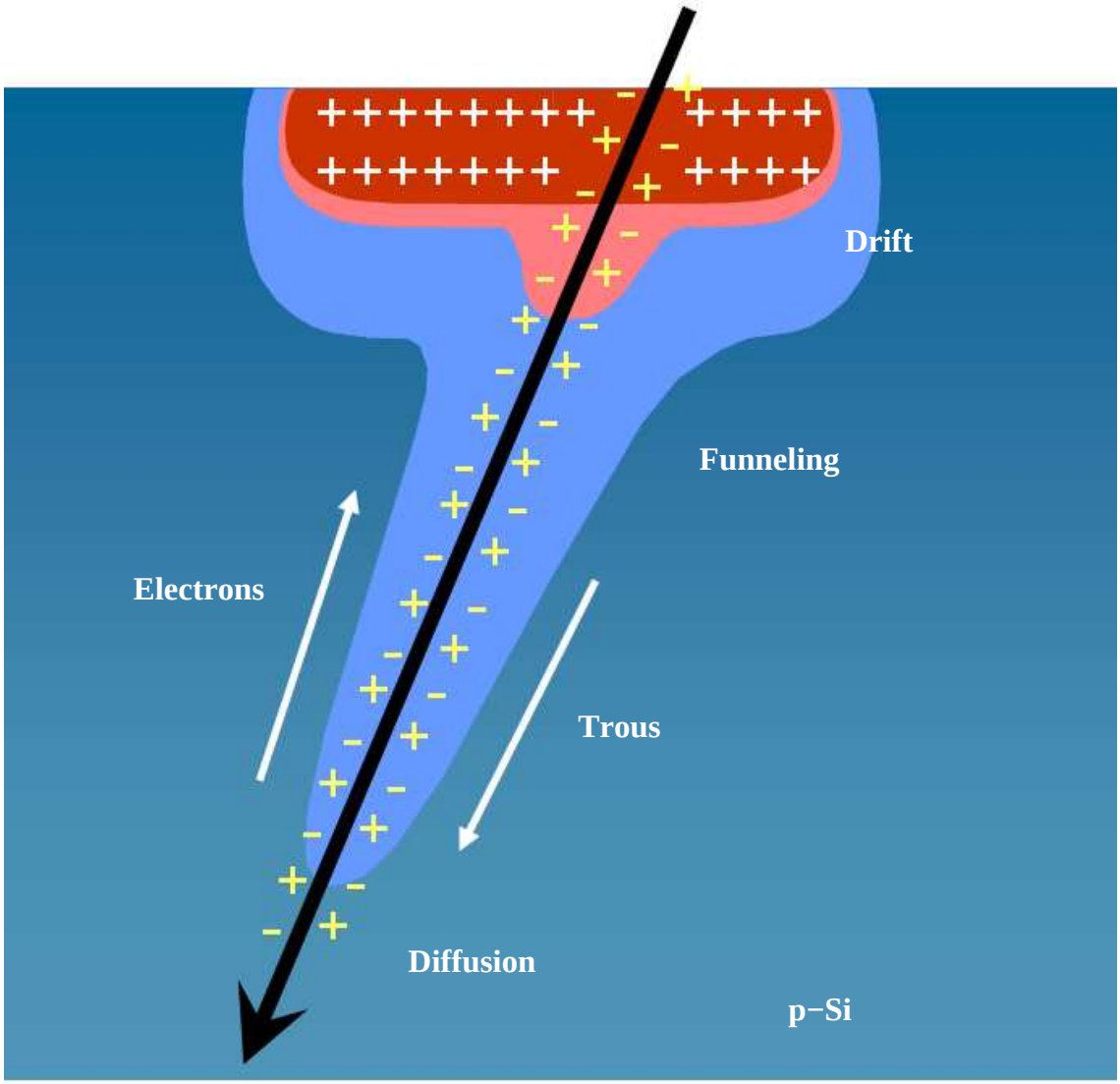

FIG. 1.9 – Mécanismes de collection de charge : drift, funneling et diffusion (d'aprés [27]).

D'autres mécanismes de collection existent. Lorsqu'un ion traverse à la fois la source et le drain d'un transistor, les charges générées sont multipliées. L'amplification bipolaire peut apparaître lorsque des charges sont injectées de la source vers le canal, et conduit à une collection excessive.

Single Event Transient

Le Single Event Transient (SET) est la conséquence directe d'une collection de charge dans un noeud sensible. C'est un pic de courant qui se propage (ou non) dans le circuit logique. Selon la variation de tension (en amplitude et en durée) il peut être amorti ou pas. Il peut aussi être annulé selon les portes logiques qu'il rencontre. Dans le pire cas, il rejoint l'entrée d'une bascule où il peut être capturé si :

- La variation de tension induite atteint un niveau logique correct (V_{IH} ou V_{IL}) de la bascule ;
- La largeur du pic est supérieure au temps de *setup* et de *hold* ;
- Il arrive aux alentours du front d'horloge.

Si ces conditions sont réunies, la bascule peut capturer un état logique incorrect. Si le SET viole le temps de setup ou de hold, la bascule peut rentrer dans un état métastable. Ce mécanisme de capture est largement dépendant de la fréquence du circuit cible.

Les SET sont pour l'instant surtout observés dans les circuits analogiques.

Single Event Upset

Le Single Event Upset (SEU) est un basculement de bit dans une bascule ou un point mémoire. Il est aussi appelé Soft Error car la perte d'information n'est pas destructrice pour l'élément contenant le bit corrompu.

Dans une cellule de type SRAM, le début du déclenchement suit celui du SET : l'apparition d'un pic de courant à un noeud du circuit. Le pire cas est toujours celui du drain du transistor N dans l'état bloqué. Lors de l'apparition du pic de courant, le transistor P de l'inverseur va fournir du courant pour compenser le courant excédentaire induit par le passage de l'ion. Une chute de tension apparaît alors, propagée à l'entrée de l'autre inverseur. Dans le cas où l'amplitude est trop faible, rien ne se passe. Sinon, deux cas se présentent alors :

- Le temps de réponse du second inverseur est court par rapport à la largeur de la chute de tension transitoire. Le SET se propage jusqu'à l'entrée du premier inverseur, alors que le transistor N touché est toujours perturbé. Dans ce cas, le transistor N touché devient passant, et le point mémoire bascule.

– Si le temps de réponse est trop long, le point mémoire ne bascule pas et le SET est amorti. Cependant, une lecture effectuée pendant cet amortissement peut conduire à ce que l'on appelle un *Single Event Disturb* (SED) : un état de fonctionnement métastable où le pic est capable de modifier temporairement le contenu du point mémoire, mais pas de le verrouiller dans la position corrompue.

On caractérise souvent la possibilité d'apparition d'un SEU par un paramètre dépendant du point mémoire : la charge critique (Q_{crit}). Si la charge collectée (Q_{col}) est supérieure à Q_{crit}, un upset est déclenché. Cette « barrière » est illustrée sur la figure 1.10.

Autres Evenements Singuliers

La famille des événements singuliers regroupe de nombreux autres effets, parmi lesquels certains sont destructifs et d'autres non [26]. Parmi les non-destructifs, citons :

– Les *Multiple Bit Upsets* (MBU). Analogue aux SEU mais plusieurs bits sont corrompus par la même particule. Ce cas de figure se présente lorsqu'un ion traverse plusieurs noeuds sensibles, ou qu'il passe entre deux cellules adjacentes (la diffusion déplaçant ensuite les charges excédentaires dans les zones sensibles). Protons et neutrons peuvent aussi engendrer ce type de phénomènes lorsque plusieurs noyaux fils sont crées par collision inélastique.
– *Single Event Functionnal Interrupt* (SEFI). Les mécanismes conduisant à cet état sont nombreux, et dépendent du circuit cible. Il se caractérise par un comportement erratique du circuit, souvent associé à une augmentation de la consommation. La récupération de l'état normal du circuit passe souvent par un reset complet, voir par un redémarrage de l'alimentation.
– *Single Hard Error* (SHE). Ce phénomène a pour symptôme un bit collé, qu'il est impossible de reprogrammer. Il est associé à une déposition de charge locale dans les oxydes de grilles, ou les oxydes d'isolation. Il est amplifié par la dose reçue par le composant.

Parmi les événements destructifs, l'on trouve :

– *Single Event Latchup* (SEL). Ce phénomène se caractérise par le déclenchement du thyristor parasite inhérent à la structure CMOS. Un court-circuit entre l'alimentation et la masse du circuit est créé, et peut aboutir à la destruction du composant si l'alimentation n'est pas coupée.
– *Single Event Burn-out* (SEB). Ce phénomène s'observe dans les MOSFETs de puissance. Si le transistor est à l'état bloqué et que le courant transitoire est capable de rendre passant le transistor bipolaire parasite formé par la source (émetteur) / le body-p (base) / la couche épitaxiale-n (collecteur), une grande quantité de courant résultante peut détruire le composant par effet thermique.

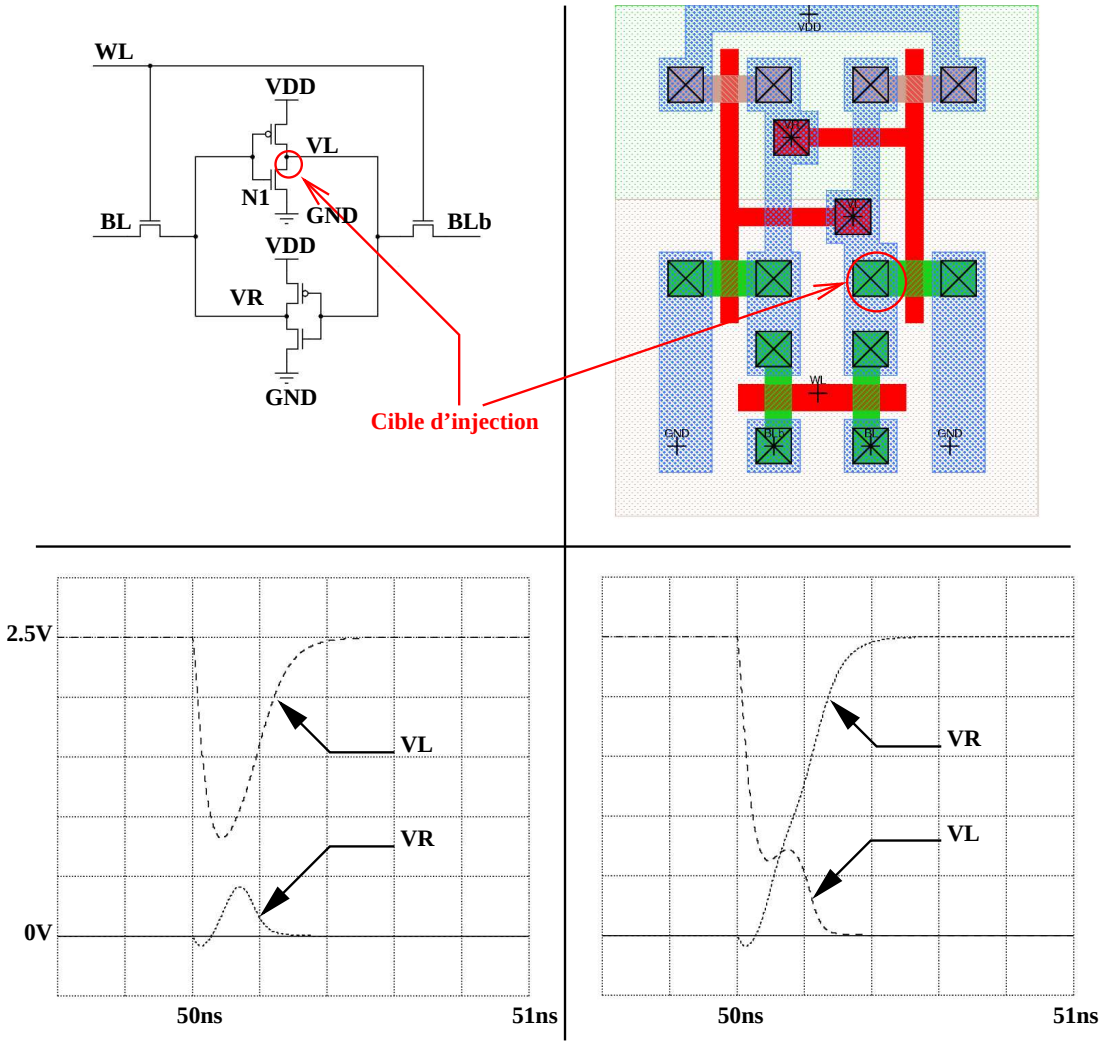

FIG. 1.10 – Illustration du phénomène SEU sur un point mémoire 0.25 μm. *Coin supérieur gauche* : Schéma électrique de la cellule. *Coin supérieur droit* : Layout correspondant. *Coin inférieur gauche* : Injection d'une charge collectée de 25 fC sur le drain du transistor N1. La charge collectée n'est pas suffisante pour faire basculer le point mémoire. *Coin inférieur droit* : Injection d'une charge de 27 fC dans le drain du transistor N1. La charge collectée est suffisante pour renverser le contenu du point mémoire (VL bascule). Les injections de charge sont faites en utilisant le modèle présenté dans [28] et [29].

Chapitre 2

Qualification de Microprocesseur face aux SEUs

CE chapitre se propose d'établir l'état de l'art en matière de qualification de microprocesseurs face aux SEU, ainsi que de présenter les méthodes d'injection de fautes les plus utilisées.

La première section est dédiée à la qualification de microprocesseurs. Historiquement, ce sont les mémoires de type SRAM qui ont été le plus étudiées. Nous présenterons le modèle généralement retenu pour l'exploitation des mesures faites sur SRAM, ainsi que les méthodes de qualification. Nous décrirons ensuite comment ces stratégies et modèles ont été adaptées aux microprocesseurs. Le principal problème engendré par cette adaptation sera mis à jour, introduisant la nécessité d'utiliser l'injection de fautes.

La deuxième section rappelle succinctement les différentes méthodes utilisables pour simuler la présence de basculement de bits dans les circuits numériques.

2.1 Qualification de Microprocesseurs

Dans le chapitre 1, nous avons décrit succinctement ce qu'était un Single Event Upset (SEU). La manière la plus simple de le représenter est un simple basculement du contenu d'un bit mémoire d'un niveau logique vers son complémentaire. C'est un effet induit par le passage d'une particule ionisante dans ou proche d'un des noeuds sensibles d'un élément de mémorisation. Sans schéma de protection adéquat, capable au minimum de signaler la corruption (utilisation d'un code de parité), voir de la corriger (code de détection et de correction d'erreur, EDAC, de type Hamming par exemple) il va de soi que l'utilisation d'un bit dont la valeur est corrompue peut avoir de sérieuses conséquences sur le comportement d'un système électronique.

Avant d'employer un composant dans une application destinée à fonctionner en

présence de rayonnement ionisants, il est donc nécessaire de connaître la sensibilité face aux SEUs du dit circuit. La meilleure façon d'obtenir cette information serait d'évaluer le comportement du circuit dans son environnement de destination, mais il est évident que cette méthode est impraticable : les environnements sont bien trop nombreux, les temps de développement sont bien trop longs.

C'est la raison pour laquelle l'on conduit des tests dits accélérés. En exposant le composant à un faisceau de particules adéquat, représentant le type d'environnement que le circuit aura à affronter, l'expérimentateur essaie d'obtenir l'information la plus précise sur le comportement du composant.

2.1.1 Modélisation de Mémoires SRAM

Quantifier la sensibilité d'une mémoire aux basculement de bits revient à exposer le composant à un faisceau de particules et compter le nombre d'évènements. Nous avons vu dans la section 1.3.3 qu'un SEU est déclenché par la création d'une colonne de charges excédentaires, et que l'on mesure la capacité d'une particule à ioniser la matière par son LET (section 1.3.1). Le nombre de SEU mesuré sera donc compté en fonction du LET de la particule incidente.

Etant donné que les accélérateurs de particules disponibles ne sont pas en mesure de délivrer des énergies équivalentes à celles rencontrées dans l'espace, on utilise d'autres ions, avec des énergies moindres, mais un LET équivalent.

Finalement, l'expérimentateur ne peut pas déterminer quelle particule parmi le faisceau a déclenché un SEU, de ce fait on mesure la valeur moyenne du nombre de SEU par particules. En pratique on divise le nombre de SEU par la *fluence* de l'expérience, c'est à dire le nombre de particules incidentes par unité de surface (usuellement le cm^2). Cette moyenne s'appelle la *section efficace macroscopique* du composant :

$$\sigma = \frac{N_{SEU}}{F}$$

avec N_{SEU} le nombre de SEU mesuré, et F la fluence.

Le produit de la mesure sous radiation de la sensibilité d'une mémoire est donc la courbe de section efficace :

$$\sigma = f(LET)$$

Une courbe de section efficace a deux caractéristiques essentielles :

- Le LET seuil (LET_{th}). C'est le LET minimum à partir duquel des SEU commencent à apparaître.
- La section efficace de saturation (σ_{sat}). Elle représente une mesure de la surface sensible du circuit.

On utilise souvent la distribution de Weibull pour ajuster les points de mesure. Cette distribution est ajustée par deux paramètres : W (même unité que le LET) et

S (sans dimension). Dans ce formalisme, l'expression de la section efficace est donnée par [26] :

$$\sigma = \begin{cases} 0 & \text{si } LET < LET_{th} \\ \sigma_{sat}\left(1 - exp\left(\left(-\frac{LET - LET_{th}}{W}\right)^{S}\right)\right) & \text{sinon} \end{cases}$$

Pour une mémoire, une hypothèse raisonnable est que chaque cellule a la même sensibilité. On divise alors la section efficace par le nombre de bits et l'on obtient une estimation de la surface sensible de chaque point mémoire. Nous avons vu (section 1.3.3) que la collection de charge se déroule en profondeur, d'où la notion de volume sensible (surface sensible + profondeur de collection de charge). Un modèle permettant de dimensionner le volume sensible est le modèle RPP (Rectangular ParallelePiped) [24] : le volume sensible est défini par un parallélépipède rectangulaire de dimensions :

$$\begin{aligned} L &= \frac{\sqrt{\sigma_{sat}}}{N_{SV}} \\ l &= \frac{\sqrt{\sigma_{sat}}}{N_{SV}} \\ h &= \frac{E_c}{\rho \times LET_{th}} \end{aligned}$$

ou N_{SV} est le nombre de volumes sensibles (nombre de bits), ρ la masse volumique du matériau traversé et E_c l'énergie critique (l'énergie que la particule incidente doit apporter pour créer un nombre suffisant de paires électrons/trous tel que la charge collectée soit supérieure à la charge critique du circuit). En réalité, dans ce modèle on admet que toute charge déposée dans le volume sensible est collectée : h est alors la profondeur de la zone de déplétion, additionnée de la profondeur de collection de charge par funneling.

2.1.2 Qualification de Mémoires SRAM

Afin de mesurer la section efficace d'une mémoire SRAM, il faut compter le nombre d'erreurs survenues, connaissant la fluence durant l'expérience. Il faut donc écrire un motif dans la mémoire, et comparer la relecture de ce motif après irradiation avec celui attendu. Quatre motifs sont généralement retenus [30] :

- L'échiquier ;
- Tout à 0, écriture unique, lectures multiples (*all-0, Write Once, Read Many*, WORM-0) ;
- Tout à 1, écriture unique, lectures multiples ; (*all-1, Write Once, Read Many*, WORM-1) ;
- Etats préférés complémentaires.

L'échiquier La mémoire est remplie d'un motif alternant les 0 et 1 logiques. A chaque passage de lecture, le complément est réécrit dans la mémoire. Cette méthode utilise toutes les fonctionnalités du circuit.

WORM La mémoire est cette fois remplie avec une seule valeur logique (0 ou 1). Elle est ensuite lue continuellement. Cette méthode maximise l'utilisation de l'alimentation par le circuit.

Etats Préférés Certaines mémoires ne sont pas automatiquement mise à 0 lors de la mise sous tension : chaque cellule contient alors sont état « préféré », celui vers lequel elle tend naturellement. En lisant son contenu, et en écrivant son complémentaire, on place la cellule dans son état le plus faible.

Autres Considérations Les méthodologies citées ci-dessus s'attachent à tester le fonctionnement du circuit dans son ensemble : cellules mémoire et circuits périphériques. L'expérimentateur voulant déterminer la section efficace des points mémoire sans entacher la mesure d'erreurs potentielles venant de la périphérie doit minimiser les accès lecture / écriture. La méthode consiste donc à écrire le motif de test dans la mémoire, exposer le circuit au faisceau de particules, puis relire le contenu de la mémoire [31] [32].

Le temps d'exposition doit être ajusté en fonction du flux de particules (et donc du nombre moyen de SEU mesuré par cycle de lecture). Un bon compromis est de l'ordre d'un SEU mesuré tous les 10 cycles de lecture. De cette manière, une accumulation anormalement élevée de SEU lors d'un même cycle de lecture peut-être écartée. Ce genre d'accumulation est normalement l'indice d'un dysfonctionnement du circuit de lecture, ou d'un MBU. Tenter de discerner l'un des deux cas requiert la connaissance du placement physique des points mémoire.

2.1.3 Extension aux Microprocesseurs

Il n'existe pas de consensus sur la méthode de qualification des processeurs face aux SEU. En effet il est aisé de comprendre que le comportement du processeur en présence de SEU dépend de l'utilisation faite de ses ressources internes : par exemple, si un bit est corrompu mais jamais utilisé par l'application exécutée par le processeur, il n'aura aucune conséquence sur le fonctionnement du système [33].

Le cas idéal serait de pouvoir tester le processeur dans son environnement final (entouré des périphériques et exécutant l'application finale). Cependant cette situation est rarement rencontrée. Deux types d'approches sont alors adoptées :

– L'approche dite *statique* ;
– l'approche *dynamique*.

Approche Statique

Cette méthodologie est directement héritée du test de mémoires. Elle consiste donc à écrire un motif dans tous les éléments mémoires du processeur (registres, mémoires cache, mémoire interne), et à les relire suite à ou pendant l'irradiation. Des exemples typiques d'utilisation de cette méthode peuvent être trouvés dans [34, 35, 36, 14, 15].

Certains processeurs offrent des moyens indirects d'accéder à leurs éléments mémoires (contrôleur JTAG, unité de débogage intégrée), mais pour la plupart des composants un programme *ad-hoc* écrit par l'expérimentateur doit être utilisé. Dans ce cas, les zones observées se limitent à celles que le jeu d'instruction du processeur permet d'adresser.

L'approche statique offre la possibilité de distinguer les SEU dans différentes zones du processeur (registres généraux, mémoire caches) et donc de mesurer leurs sections efficaces respectives en fonction du LET des particules incidentes.

Approche Dynamique

Cette approche consiste à exposer le composant à un flux de particules pendant qu'il exécute une « activité appropriée ». En général, le microprocesseur exécute des applications simples, supposées en quelque sorte représentatives. On observe alors les résultats produits par l'exécution de ces programmes. Les programmes sont écrits de sorte à activer les zones sensibles du circuit de la manière la plus proche possible de celle de l'application de vol. On ne mesure plus alors la section efficace du circuit, mais le taux d'erreur par particule du système microprocesseur + application, ceci toujours en fonction du LET.

Remarques

Les références [35, 36, 14] présentent les résultats de l'application à divers processeurs des deux stratégies citées ci-dessus. Les résultats obtenus montrent que les taux d'erreurs issus de l'exposition de programmes dynamiques dépendent de celui-ci, et sont inférieurs d'au moins un ordre de grandeur par rapport aux section efficaces obtenues lors d'un test statique (pour effectuer cette comparaison, on associe arbitrairement l'apparition d'un SEU en sortie du test statique à une erreur).

Ceci n'a rien de surprenant. Un programme de test statique est écrit de sorte à observer le maximum de points mémoire et à rapporter toute variation, de sorte

qu'on peut le considérer comme un pire-cas. Au contraire, pour qu'un upset provoque une erreur en sortie d'un programme dynamique, il faut qu'il survienne durant la période où le point mémoire contient une information qui va être utilisée par la suite.

Le problème qui se pose est le suivant. Les modèles qui permettent de déterminer le taux de SEU journalier (CREME96 [37] par exemple) pour un environnement radiatif donné nécessitent la connaissance à priori de la section efficace du point mémoire. Mais les résultats obtenus à l'aide de programmes dynamiques montrent que le choix de la section efficace comme métrique de la sensibilité d'une application quelconque constitue un pire-cas.

D'un autre côté, les programmes dynamiques ne permettent pas de déterminer la section efficace statique des éléments mémoires du processeur. Par contre, ils donnent un taux d'erreur plus représentatif d'une application quelconque. Cependant on note que ce taux d'erreur est dépendant de l'application exécutée : il faudrait donc à priori exposer l'application finale qui sera exécutée par le microprocesseur pour connaître son taux d'erreur.

C'est dans ce contexte qu'émerge comme solution potentielle la notion d'**injection de faute**. L'expérimentateur cherche à déterminer le taux d'erreur journalier ξ d'une application exécutée par un microprocesseur dans un environnement donné. Ce taux peut se décomposer en :

$$\xi = \frac{Nombre\ d'erreurs}{Nombre\ de\ SEU} \times \frac{Nombre\ de\ SEU}{Nombre\ de\ particule} \times \frac{Nombre\ de\ particules}{Nombre\ de\ jours}$$

Nous voyons apparaître ici la définition de la section efficace σ. Utilisée comme paramètre d'entrée des modèles d'environnement, nous obtenons le taux d'upset journalier. Le rôle de l'injection de fautes est alors de fournir le taux d'erreur par upset τ du système microprocesseur + application.

De cette façon, il est possible de déterminer le taux d'erreur d'une application quelconque en présence de SEU.

2.2 Injection de fautes

Cette section présente les différentes méthodes existantes d'injection de fautes et applicables aux microprocesseurs.

Comme énoncé dans la section précédente, le rôle de l'injection de fautes est de déterminer une estimation du taux d'erreur par upset. Le modèle de faute est le suivant : un SEU est représenté par l'inversion du contenu d'un point mémoire (d'où le terme *bitflip* dans la littérature anglo-saxonne).

Le principe de l'injection de faute est simple : durant l'exécution du programme par le microprocesseur, l'expérimentateur provoque le basculement du contenu d'un point mémoire et observe le résultat de cette perturbation sur le déroulement du programme.

Dans la littérature, on trouve un certain nombre de méthodes permettant d'effectuer ce basculement de bit durant l'exécution de l'application. Elles sont présentées ci-dessous.

2.2.1 Niveau transistor

La simulation se situe au niveau du schéma électrique du microprocesseur. On injecte un pic de courant dans le point mémoire afin de le faire basculer. Ces pics de courant sont généralement modélisés suivant une double exponentielle [28, 29] et injectés à l'aide de logiciels de type SPICE. Il faut évidement calibrer le pic de courant pour qu'il déclenche un upset « à coup sûr ».

La référence [38] présente une approche complète de tentative de qualification à partir du schéma électrique. Les auteurs ne se limitent pas à obtenir le taux d'erreur par upset à l'aide de l'injection de faute, ils tentent aussi de déterminer la section efficace statique en modulant le pic de courant en fonction du LET de la particule simulée.

En utilisant cette approche le modèle de faute peut donc être étendu au niveau analogique. Les deux principales limitations de cette méthode sont tout d'abord la disponibilité d'une telle représentation électrique du microprocesseur, et ensuite la lenteur des simulations.

2.2.2 Niveau Comportemental

Cette approche utilise la description comportementale (en langage VHDL ou Verilog) du microprocesseur. Durant la simulation, des commandes propres au simulateur employé sont utilisées pour corrompre les éléments mémoires du processeur. La référence [39] donne un exemple de l'emploi de cette méthode. Toutefois, la lenteur des simulations est encore une fois pénalisante.

2.2.3 Emulation FPGA

Cette méthode tente d'apporter une réponse à la lenteur des simulations au niveau comportemental. La description du processeur est synthétisée, puis placée et routée pour être émulée sur un FPGA (*Field Programmable Gate Array*). Au cours de ces étapes, les éléments mémoire sont « instrumentés » de façon à ce que leur contenu soit aisément accessible et modifiable durant l'exécution du programme par le microprocesseur.

Dans [40], les auteurs rapportent une accélération variant d'un facteur 20 à 45 par rapport à la méthode au niveau comportemental, pour des résultats de taux d'erreur comparables.

2.2.4 Emulation Logicielle

Cette approche est similaire à celle au niveau comportemental. Ici, au lieu de recourir à une description en langage HDL (*Hardware Description Language*), l'expérimentateur utilise un simulateur de jeu d'instruction (*Instruction Set Simulator*, ISS) et utilise les commandes disponibles pour modifier les registres et autres emplacements mémoires du microprocesseur.

L'utilisateur est cependant limité par les possibilités offertes par le simulateur [41] : le simulateur peut ne pas être totalement conforme au microprocesseur, notamment en présence d'erreurs et dans la façon dont le processeur y réagit.

2.2.5 Laser

De la même manière que pour la simulation au niveau transistor, le principe de l'injection de faute par laser consiste à générer un pic de courant transitoire dans le point mémoire ciblé afin d'en inverser son contenu [42]. L'injection se fait cette fois-ci sur un prototype physique du circuit. Ce *photo-courant* est généré par l'apport d'énergie localisé induit par le faisceau laser. L'énergie du laser doit être calibrée de manière à déclencher un upset à chaque impact.

Le laser n'est bien sûr pas limité à ce type d'injection de fautes. La possibilité de faire varier l'énergie, et donc la quantité de charges générée en fait un outil capable de sonder les valeurs de section efficaces. Des modèles existent, permettant d'associer l'énergie du laser à un LET équivalent [42] d'une manière plus ou moins fiable [43, 44].

La principale limite de cette méthode est la profondeur de pénétration du faisceau laser, réduite par l'opacité des couches métalliques supérieures du circuit (pour une longueur d'onde de 800 nm, la profondeur est estimée à environ 12 μm). Une façon de s'affranchir de cette limitation est l'exposition par le substrat du circuit (*Backside Testing*). Cependant, dans la plupart des cas le boîtier du circuit doit être ouvert, et le substrat aminci, ce qui limite les applications.

Les problèmes d'amincissement du substrat pourraient disparaître avec l'emploi de la technique TPA (*Two-Photon Absorption*) [45] : l'énergie du laser est située sous le gap du semiconducteur, de ce fait le matériau est transparent vu du laser. Cependant, à l'endroit où le laser est concentré, l'énergie est suffisante pour libérer des porteurs excédentaires.

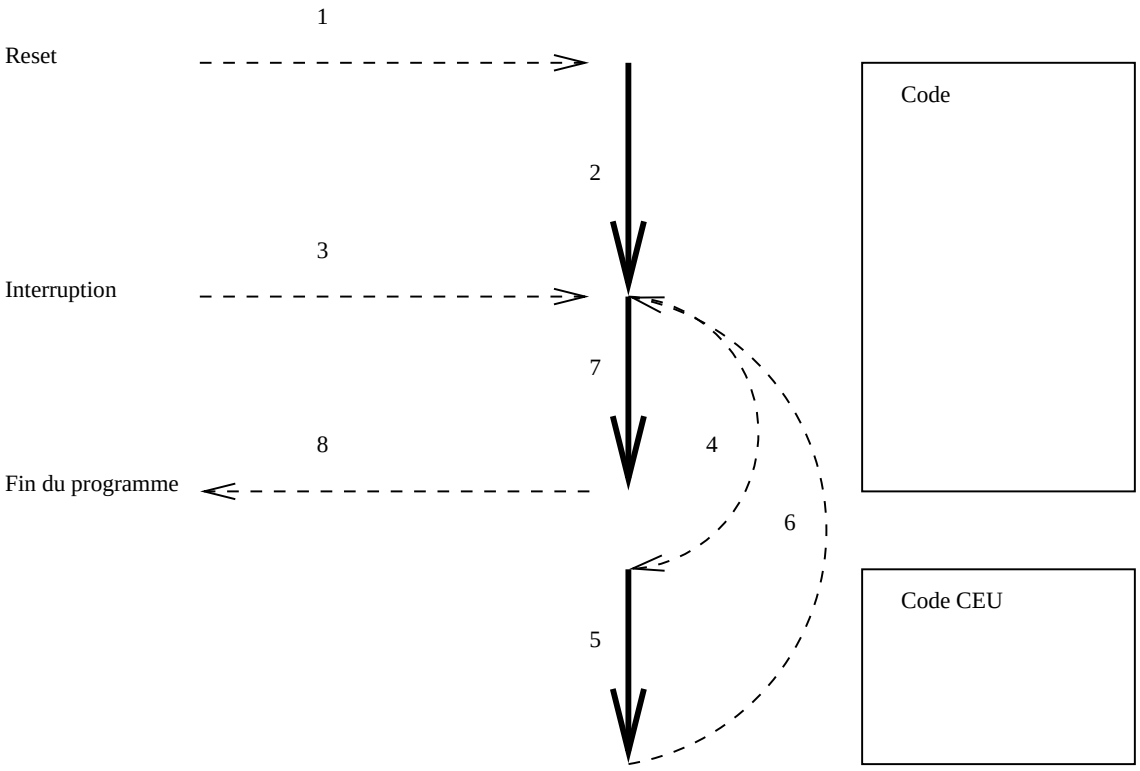

FIG. 2.1 – Principe d'injection de fautes selon la méthode CEU

2.2.6 Méthode CEU

Cette méthode repose sur le fait qu'il est possible de simuler l'inversion d'un bit par l'exécution par le processeur d'un programme adéquat [46, 47] :

- Lecture du contenu du registre cible ;
- Corruption d'un bit du registre à l'aide d'un masque et d'un OU exclusif ;
- Ecriture du registre cible avec la valeur corrompue.

Afin que ce morceau de programme puisse être exécuté à n'importe quel moment, il est lié à un vecteur d'interruption : le déclenchement de l'interruption en question provoque l'exécution du programme corrupteur, et donc l'inversion de bit. A la fin de l'interruption, le processeur retourne automatiquement à l'exécution de l'instruction interrompue. Ce mécanisme est représenté sur la figure 2.1 :

1. Lancement du processeur par activation du signal de reset ;
2. Le processeur démarre l'exécution du programme ;
3. Activation de l'interruption à un instant choisi de manière adéquate ;
4. Le processeur interrompt le flot d'exécution normal et saute vers la routine d'interruption ;

Cible	Pseudo-code	Description
Registre généraux	`xor R, masque`	`R` ← `R` ⊕ `masque`
Mot mémoire	`push R`	`[SP]` ← `R` ; `SP` ← `SP + 1`
	`load R, addresse`	`R` ← `[addresse]`
	`xor R, masque`	`R` ← `R` ⊕ `masque`
	`store R, addresse`	`[addresse]` ← `R`
	`pop R`	`SP` ← `SP - 1` ; `R` ← `[SP]`
Compteur programme	`pop R`	`SP` ← `SP - 1` ; `R` ← `[SP]`
	`xor R, masque`	`R` ← `R` ⊕ `masque`
	`push R`	`[SP]` ← `R` ; `SP` ← `SP + 1`

TAB. 2.1 – Exemples de pseudo-codes CEU pour le 8051, d'après [47]. L'exemple pour le compteur programme (PC) suppose que PC est emplilé dans une pile logicielle lors d'une interruption et dépilé lors du retour d'interruption.

5. Exécution du code corrupteur ;
6. Retour vers le programme ;
7. Le processeur termine l'exécution du programme, avec un bit corrompu ;
8. Fin de l'exécution. Les résultats de l'exécution du programme sont disponibles.

Dans cette approche, le contenu du code CEU est dépendant de la cible de l'injection de faute, et doit être généré pour tout changement de cible. Le tableau 2.1 présente quelques cas typiques de code CEU appliqués au microprocesseur 8051.

2.3 Résumé

Dans ce chapitre ont été introduit les méthodes de qualification de processeurs. La première méthode, appelée test statique, et directement héritée de la qualification de mémoire, permet de mesurer la sensibilité intrinsèque du composant. La seconde, nommée test dynamique, permet d'observer le comportement du processeur en situation plus où moins réelle.

Cependant, il a été montré qu'aucune des deux méthodes n'est pleinement satisfaisante. Cette constatation mène à l'introduction de la notion d'injection de fautes.

Les techniques classiques d'injection de fautes ont ensuite été passé en revue.

Le prochain chapitre tentera d'expliquer les défauts communs aux méthodes d'injection de fautes traditionnelles, et expliquera l'élaboration d'un nouveau modèle d'injection de fautes.

Chapitre 3

Modèle d'Injection de Fautes

DANS ce chapitre, nous proposons une nouvelle approche pour l'injection de fautes de type SEU sur microprocesseurs.

Ce chapitre s'articule autour de trois axes. Nous commencerons par discuter des motivations : pourquoi développer un nouveau modèle d'injection de fautes. De cette question, deux nouveaux axes apparaîtront : comment injecter une faute, et surtout quand.

La première réponse sera apportée par une extension de la méthode CEU, la seconde par la construction d'un modèle statistique permettant de déterminer des instants d'injection de fautes de la manière la plus réaliste possible. Un des objectifs sera aussi que ce modèle puisse estimer sa propre erreur, c'est à dire la vitesse à laquelle il converge vers une expérience réelle de mesure sous radiation.

Nous commencerons par définir ce qu'est la section efficace, et montrer qu'elle a une interprétation probabiliste. Un premier modèle sera ensuite proposé, permettant d'obtenir une estimation de l'erreur commise lors de son utilisation.

De ce premier modèle, nous dériverons un second qui permet de déterminer de manière aléatoire des instants réalistes d'injection de fautes. Nous terminerons le chapitre par un récapitulatif du modèle proposé.

3.1 Motivations

Nous avons vu dans le chapitre précédent que la connaissance du comportement d'un système microprocesseur + application en présence de SEU passe par trois étapes :

- La mesure de la section efficace du processeur σ ;
- En combinant σ avec les propriétés de l'environnement radiatif, on obtient un taux d'upset journalier ;
- Finalement, l'injection de fautes apporte le taux d'erreur par upset.

Ces trois étapes permettent de déterminer le taux d'erreur par jour du système dans un environnement donné.

Pour quantifier la qualité des prévisions issues d'une telle chaîne de traitement, et donc de la pertinence de l'injection de faute, il faudrait comparer ces résultats avec des mesures prises en vol. Cependant :

- A notre connaissance aucune étude n'a été publiée sur le sujet ;
- Les taux d'erreurs en vol sont généralement faibles, ce qui pose problème au niveau de la qualité statistique des résultats ;
- L'introduction des modèles d'environnement dans la chaîne de traitement entache la comparaison : il est reconnu que les modèles d'environnement ne sont pas parfaits [26].

La meilleure façon de qualifier l'injection de faute reste donc la comparaison avec des mesures prises sous faisceau, lors de tests accélérés. Une série de deux mesures (pour un LET donné) permet de :

- Déterminer σ à l'aide d'un programme statique ;
- Mesurer un taux d'erreur par particule Σ pour un programme quelconque.

Le taux d'erreur par upset τ_{rad} peut donc être déterminé par la relation :

$$\tau_{rad} = \frac{\Sigma}{\sigma}$$

avec σ la section efficace du composant (le taux d'upset par particule) et Σ le taux d'erreur par particules. Il permet une comparaison directe avec le taux d'erreur par upset τ_{inj} obtenu par injection de fautes. Dans la pratique c'est souvent le produit $\tau_{inj} \times \sigma$ que l'on compare avec Σ.

La référence [48] est une bonne illustration de cette démarche, avec pour stratégie d'injection de fautes la méthode CEU. Cependant, plusieurs points viennent entacher la qualité des résultats :

1. Les auteurs n'ont pas injecté de fautes dans les mémoires caches du processeur. Quand celles-ci sont activées, les résultats de prédiction s'écartent de manière significative des mesures prises sous radiation ;
2. Les auteurs ont choisi, lors de la détermination par injection de fautes du taux d'erreur par upset, d'injecter une seule et unique faute par exécution du programme, distribuée aléatoirement durant le déroulement du programme. Ce choix n'est à aucun moment justifié ;
3. Le problème de cibles avec des sections efficaces différentes n'est pas abordé. Comme nous le verrons par la suite, la section efficace est l'expression directe de la probabilité qu'un point mémoire bascule. Une injection de faute réaliste devrait prendre ce phénomène en compte.

Alors que le point 1 peut sembler être une limitation inhérente à la méthode CEU (les autres méthodes citées au chapitre 2 ne partagent pas cette limitation), le point 2 est un problème partagé. En général, les méthodes d'injection de fautes s'attachent à la façon d'injecter une faute, pratiquement jamais à l'instant choisi pour injecter une faute. A notre connaissance aucune publication n'adresse le point 3.

Ce sont ces constatations qui ont motivé le travail de cette thèse. Tout d'abord étendre la méthode CEU afin de la rendre compatible avec des microprocesseurs avec mémoire cache. Enfin, développer un modèle permettant de déterminer des instants d'injection de fautes réalistes vis-à-vis d'un test accéléré sous radiation. Afin de permettre une comparaison avec les mesures prises sous radiation, cet ensemble de méthodes devra de plus fournir une estimation de sa propre précision.

3.2 Méthode d'Injection de Faute Etendue

Nous rappelons ici le principe de base de la méthode CEU : le code corrupteur est placé à l'adresse d'un vecteur d'interruption du processeur. Lorsque cette interruption est déclenchée, l'exécution du code CEU provoque la corruption d'un bit et d'un seul du processeur, avant de rendre le contrôle au programme principal, à l'exécution même de l'instruction interrompue.

Il va de soi qu'idéalement le code CEU doit être aussi « silencieux » que possible, c'est à dire ne modifier que le bit visé, et rien d'autre.

Les sous-sections suivantes ne donnent que des principes généraux. Un code CEU est écrit au plus bas niveau, c'est à dire en assembleur. Il est donc complètement dépendant du jeu d'instruction du processeur et de ses possibilités. Enfin, écrire un code CEU « silencieux » nécessite une connaissance intime de l'architecture du processeur testé, et beaucoup d'astuce.

Plus le code CEU est « affiné », moins son exécution se remarque (mis à part le bit ciblé). Dans tout les cas, son écriture et son exécution doivent être parfaitement analysées : par exemple, dans le cas où le code CEU exige un vidage des mémoires caches, le temps d'exécution du programme peut être grandement rallongé. Ceci doit être pris en compte pour ne pas déclarer le processeur « perdu » alors qu'il est simplement en train de remplir ses caches.

3.2.1 Ciblage du cache donnée

Deux cas se profilent ici :
- Le jeu d'instruction du processeur permet un accès direct au cache donnée ;
- L'accès direct est interdit.

Dans le premier cas, la technique est triviale : il suffit d'accéder directement au cache donnée et de le modifier. Cette modification passe souvent par un saut en mode « super utilisateur » du processeur afin de pouvoir exécuter ce genre d'instruction. Il faut bien évidement prendre en compte l'exécution du code CEU : celui ci peut parfaitement venir remplacer des lignes d'instruction dans le cache instruction. Cependant, la plupart des microprocesseurs actuels permettent un « gel de cache » pendant les interruptions. Ce mode particulier assure qu'aucun des caches ne sera modifié de manière indirecte (c'est à dire qu'il ne peut être modifié que par l'exécution d'instructions spécifiques) pendant toute la durée de l'interruption.

Dans le deuxième cas, le principe est un peu plus compliqué :

- La première étape consiste à désactiver l'accès aux caches. Dans cette configuration, les caches ne sont plus mis à jour ;
- Ensuite, on vient invalider le cache donnée (*flush*). La mémoire externe du processeur est mise à jour avec le contenu du cache afin d'en assurer la cohérence ;
- La corruption du bit cible se fait alors en mémoire externe ;
- Une fois cette modification effectuée, le cache données est réactivé ;
- Les données sont relues par le processeur, et donc copiées dans le cache donnée ;
- Finalement le cache instruction est réactivé.

Les modifications accidentelles sur le cache instruction sont minimes voir nulles. L'impact sur le cache donné est lui plus ou moins dépendant de l'algorithme de gestion des caches : pour un cache *direct mapped* la correspondance bloc de cache - mémoire externe est aisée à déterminer, dans le cas de cache *set assiocative* elle devient plus délicate.

De manière générale, quand le microprocesseur est prévu pour fonctionner en environnement multiprocesseur, les concepteurs implémentent des protocoles de cohérence de cache qui permettent une modification aisée de leurs contenus. Ces mécanismes ouvrent une porte primordiale pour l'utilisation de la méthode CEU.

3.2.2 Ciblage du cache instruction

Les deux cas de la mémoire cache donnée sont ici aussi applicables. Nous ne parlerons donc que de la situation où le cache instruction n'est pas accessible.

Dans ce cas, on peut employer une technique baptisée *self-modifying code* ou encore *polymorphic code*. Cette technique a été mise en pratique au début des années 90 dans certains virus pour échapper aux tentatives de détection par calcul d'une signature sur un fichier binaire.

Lors de l'interruption, l'instruction est modifiée à la volée, en mémoire vive, comme une donnée, avant d'être exécutée. Une seconde interruption doit être lancée juste après que l'instruction modifiée entre dans le cache, de manière à restaurer le programme modifié (et donc conserver un code correct en mémoire vive et une

instruction corrompue en mémoire cache). C'est une technique délicate à mettre en place.

3.2.3 Amélioration générale

Dans la méthode traditionnelle, le code CEU est dépendant de la cible visée. A chaque nouvelle exécution, le code CEU correspondant doit être copié en mémoire vive par un opérateur externe au processeur de façon à refléter la nouvelle injection.

Nous avons préféré lorsque c'est possible, une approche plus générique qui consiste à écrire une routine monolithique, résidente en mémoire morte avec le programme à exécuter. Cette routine exporte tout le contexte mémoire du processeur en mémoire vive, modifie le bit ciblé en mémoire vive, puis restaure le contenu corrompu. De cette manière la routine CEU n'accepte que deux paramètres : une adresse et un bit à modifier. L'exécution de la routine dure le même nombre de cycles quelle que soit la cible, ce qui rend le traitement des données post-injection plus aisé.

3.3 La Section Efficace : Définition

La prédiction du taux d'erreurs dues aux SEU nécessite de connaître la réponse du circuit à l'impact d'une seule particule. Comme mentionné plus haut, ceci requiert l'exposition et l'étude du comportement du circuit face à un type de radiation. Cependant, une installation d'irradiation n'est pas capable d'une telle granulosité : les particules sont toujours délivrées sous forme d'un faisceau. De plus, dans la plupart des cas, le composant a de nombreux volumes sensibles. L'expérimentateur ne peut déterminer quelle particule va interagir avec quel volume et à quel instant.
Ceci a pour conséquence que les données obtenues sous radiations ont une nature statistique et sont dépendantes des caractéristiques du faisceau (type de particule, flux, angle d'incidence) et du composant cible (dimensions et nombre de volumes sensibles entre autres). L'expérimentateur a donc besoin d'un concept permettant de masquer ces effets géométriques afin d'observer la dynamique d'interaction entre une particule et le circuit. C'est le concept de *section efficace*.

Afin d'introduire ce concept, nous présenterons deux exemples. Le premier, tiré de la vie réelle, fait appel au sens commun. Armé de ce bon sens, le second exemple permettra de dériver un formulation mathématique simple de la section efficace.

3.3.1 Le Jeu de Fléchettes

Supposons un joueur de fléchettes irlandais qui lance au hasard N_T fléchettes en direction d'une cible. A la fin du jeu, on compte :

- N_P fléchettes plantées dans la cible ;
- Parmi ces N_P fléchettes, N_C sont plantées au centre.

Le bon sens nous dit qu'il doit exister une relation de proportionnalité entre N_C/N_P (la fraction de fléchettes plantées et au centre) et le rapport de la surface du centre à celle de la cible S_C/S :

$$\frac{N_C}{N_P} \simeq K\frac{S_C}{S} \tag{3.1}$$

K étant une variable, sensiblement proche de l'unité, mais dépendante entre autre de la chance et de l'état d'ébriété du tireur.

3.3.2 Formulation Mathématique

Gardant en tête la relation 3.1, imaginons une version simplifiée d'une expérience de mesure de section efficace type. L'expérience est représentée sur la figure 3.1. Les conditions expérimentales sont supposées être les suivantes [49] :

- Le circuit sous test est composé d'une nombre fini et connu N_{SV} de volumes sensibles identiques ;
- La surface d'un volume sensible vue par les particules est notée σ ;
- La surface totale du circuit est notée S_T, et S_S est la surface sensible du circuit. Dés lors, $S_S = N_{SV} \times \sigma$ et $S_S \leq S_T$: les volumes sensibles ne sont pas forcément accolés ;
- Les particules arrivent à incidence normale à la surface du circuit, une à une ;
- Les particules délivrées par le faisceau ont un LET suffisant pour déclencher un SEU si elles traversent un volume sensible. Ceci revient à supposer que la charge collectée dans le volume sensible Q_{coll} sera toujours supérieure à la charge critique du noeud Q_{crit} ;
- Le flux de particules (nombre de particules arrivant par unité de temps et surface) Φ est connu ;
- L'appareillage est capable de mesurer le nombre de SEU survenus par unité de temps λ_{SEU}.

Si l'on note λ_P le nombre de particules traversant S_T par unité de temps, par analogie avec 3.1 (les particules sont les fléchettes, le composant la cible et les volumes sensibles les centres),

$$\frac{S_S}{S_T} = \frac{\lambda_{SEU}}{\lambda_P}$$

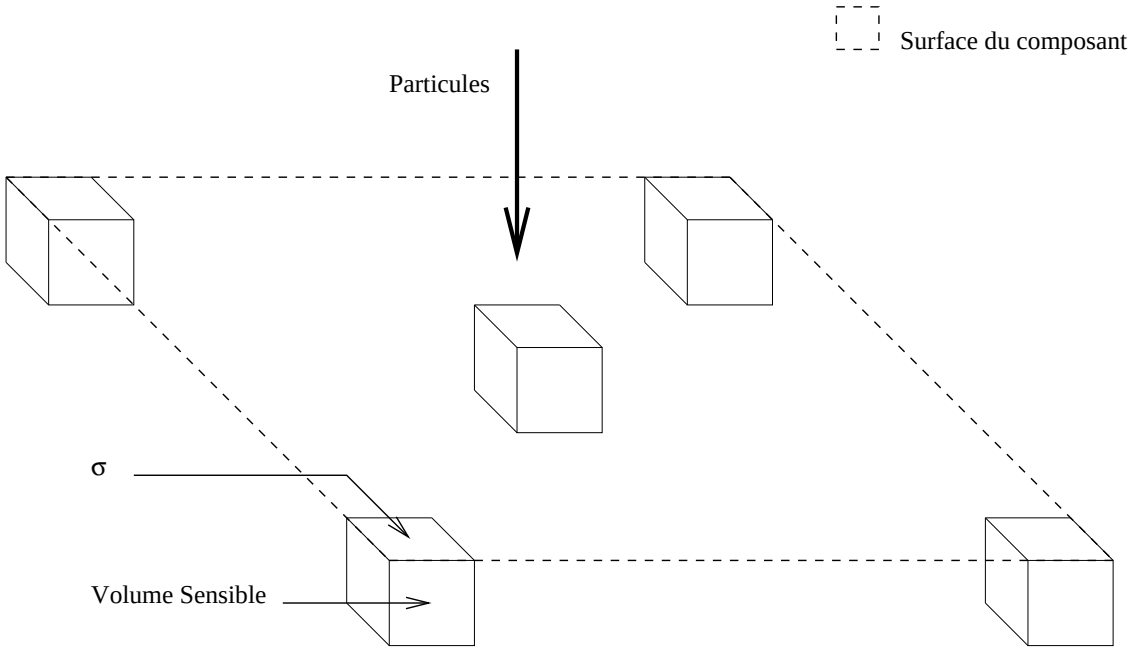

FIG. 3.1 – Schéma de l'expérience.

En remplaçant S_S par $N_{SV} \times \sigma$ on obtient

$$\sigma = \frac{\lambda_{SEU}}{\frac{\lambda_P}{S_T} \times N_{SV}}$$

λ_P/S_T n'étant autre que l'expression du flux Φ, σ s'exprime par la relation

$$\sigma = \frac{\lambda_{SEU}}{\Phi \times N_{SV}} \tag{3.2}$$

Finalement, si l'on introduit la durée T de l'expérience,

$$\sigma = \frac{\lambda_{SEU} \times T}{\Phi \times N_{SV} \times T} = \frac{N_{SEU}}{F \times N_{SV}}$$

où N_{SEU} est le nombre de SEU observés durant l'expérience, et F la fluence de l'expérience.

σ est appelée la section efficace d'interaction, et est une mesure directe de la surface supérieure du volume sensible. Elle est usuellement exprimée en cm^2 ou en barns (1 barn $= 10^{-24}\ cm^2$). Quand le nombre de volumes sensibles N_{SV} est inconnu, on l'exprime en tant que section efficace d'interaction par composant ou par bit :

$$\sigma_{dev} = \frac{N_{SEU}}{F} \tag{3.3}$$

$$\sigma_{bit} = \frac{N_{SEU}}{F \times N_{bit}} \tag{3.4}$$

Pour résumer, la mesure de la section efficace d'interaction revient à compter le nombre de SEU apparus pendant un temps T, et de diviser ce nombre par la quantité de particules envoyées sur le circuit par unité de surface pendant ce même temps T.

3.4 Interprétation Probabiliste

3.4.1 Définition Formelle d'une Probabilité

Supposons que l'on observe un événement E répété N_e fois, on dit que l'on prend un échantillon de N_e événements. Dans n cas, cet événement est caractérisé par une marque distinctive a (appelée aussi caractère). Si les événements dans cette suite sont indépendants, alors la probabilité $\mathcal{P}(a)$ que la marque a se manifeste est définie comme

$$\mathcal{P}(a) = \lim_{N_e \to +\infty} \frac{n}{N_e} \tag{3.5}$$

Rappelons enfin qu'une probabilité $\mathcal{P}(a)$ suit la propriété suivante : $0 \leq \mathcal{P}(a) \leq 1$

3.4.2 Application à la Section Efficace

Pourvu que l'on reste dans le cadre strict de la mesure de SEU, c'est à dire qu'une particule déclenche au plus un SEU, on peut mettre en parallèle les équations 3.3 et 3.4 avec 3.5. L'évènement E devient l'observation de l'interaction d'une particule avec le circuit, répétée F fois (nombre de particules lancées sur le circuit), la marque distinctive a est l'apparition d'un SEU, et la probabilité $\mathcal{P}(SEU)$ qu'une particule déclenche un SEU s'exprime par

$$\mathcal{P}(SEU) = \lim_{F \to +\infty} \frac{N_{SEU}}{F} = \lim_{F \to +\infty} \sigma \tag{3.6}$$

Deux cas limites des équations 3.3 et 3.4 se profilent :

- Le composant n'est pas sensible aux SEU. Dans ce cas $N_{SEU} = 0$ et $\sigma = 0$;
- Le composant est très sensible aux SEU. Dans ce cas limite, chaque particule déclenche un SEU, $N_{SEU} = F$ et $\sigma = 1$ *u.a.*.

Le reste du temps, $N_{SEU} \leq F$, et par conséquent $\sigma \in [0; 1]$. La section efficace d'interaction peut donc être considérée comme une mesure de la probabilité qu'a une particule de déclencher un SEU, pourvu que le nombre de particules utilisées pour mesurer cette section efficace soit grand.
Dans la suite du chapitre, nous supposerons la probabilité $\mathcal{P}(SEU)$ connue (par exemple mesurée lors d'un test sous radiation).

3.5 Modélisation suivant la Loi Binomiale

Cette section, ainsi que la suivante ont fait l'objet d'une publication [50]. Nous souhaitons établir un modèle permettant de déterminer des instants d'injection de fautes. La première étape consiste à modéliser numériquement l'expérience de mesure de section efficace présentée en 3.3.2 . Nous emploierons ensuite le théorème de la limite centrée pour dériver un intervalle de confiance du modèle. Finalement, nous adresserons quelques critiques du modèle présenté.

3.5.1 Modélisation

On considère une expérience dont le résultat de chaque épreuve ne peut prendre que deux valeurs appelées par convention *succès* ou *échec*.
A une expérience de ce type est associé une variable aléatoire X prenant la valeur 1 pour le succès, et la valeur 0 pour l'échec avec les probabilités respectives p et $(1-p) = q$. Cette variable est appelée *variable de Bernoulli*. La *loi de probabilité* de X est définie par :

$$\begin{aligned} \mathcal{P}(X=1) &= p \\ \mathcal{P}(X=0) &= 1-p \\ &= q \end{aligned}$$

Si l'on réalise n épreuves indépendantes de la même expérience telles que :

- chaque épreuve ne peut avoir que deux résultats, s'excluant mutuellement (succès ou échec) ;
- la probabilité p de succès est constante à chaque épreuve, la probabilité d'échec est également constante et égale à $1 - p = q$,

alors la variable aléatoire X qui compte le nombre de succès au cours de n épreuves est appelée *variable binomiale*.
La loi de la variable X est appelée *loi binomiale de paramètre* (n, p), notée $\mathcal{B}(n; p)$.
La probabilité d'obtenir k succès lors de ces n épreuves est définie par :

$$\mathcal{P}(X=k) = \left(\begin{array}{c} n \\ k \end{array} \right) p^k (1-p)^{n-k} \tag{3.7}$$

Si l'on applique ce modèle à l'expérience 3.3.2, la probabilité d'obtenir k SEUs s'exprime par :

$$\mathcal{P}(N_{SEU}=k) = \left(\begin{array}{c} F \\ k \end{array} \right) \mathcal{P}(SEU)^k (1-\mathcal{P}(SEU))^{F-k} \tag{3.8}$$

3.5.2 Dérivation d'un intervalle de Confiance

En théorie des probabilités, le terme *théorème limite central* désigne toute proposition affirmant que sous certaines conditions la fonction de répartition d'une somme de variables aléatoires individuellement petites converge vers la fonction de répartition d'une loi normale lorsque croît le nombre de termes. Le théorème limite central doit son importance à l'explication théorique qu'il fournit de l'observation suivante, confirmée du reste par la pratique : si l'issue d'une expérience aléatoire est définie par un grand nombre de facteurs aléatoires, chacun d'eux exerçant une influence arbitrairement petite, cette expérience se laisse bien approximer par une loi normale d'espérance mathématique et de variance convenablement choisies.
Nous rappelons ici la formulation du théorème dite de Lévy-Lindeberg :

Théorème 1 (Lévy-Lindeberg) *Si* $\{X_k, k \geq 1\}$ *est une suite de variables aléatoires deux à deux indépendantes identiquement réparties, la fonction de répartition* $F_n(x)$ *de la somme normée* $\eta_n = \frac{\sum(X_k - na)}{\sigma\sqrt{n}}$ *vérifie la relation*

$$\lim_{n\to\infty} F_n(x) = \Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-z^2/2} dz \tag{3.9}$$

a étant l'espérance et σ^2 *la variance des variables aléatoires* X_k*.*

Le théorème suivant est un cas particulier important du théorème de Lévy-Lindeberg formulé pour des variables aléatoires binomiales X_k [51, 52].

Théorème 2 (Moivre-Laplace) *Si* v_n *est le nombre d'apparitions d'un événement de probabilité* p*,* $0 < p < 1$*, dans une suite de* n *épreuves indépendantes, alors la fonction de répartition* $F_n(x)$ *de l'écart normé par rapport au nombre moyen* $\eta_n = \frac{v_n - np}{\sqrt{np(1-p)}}$ *d'apparitions vérifie la relation 3.9.*

Il est équivalent de dire que la variable η_n est asymptotiquement normale réduite. La figure 3.2 illustre cette convergence pour plusieurs valeurs de n.
Pour étudier la convergence de la fréquence d'apparition, nous pouvons évaluer la probabilité que l'écart entre la fréquence $\frac{v_n}{n}$ de l'apparition d'un événement dans une suite d'épreuves de Bernoulli et la probabilité p, $0 < p < 1$, de cet événement soit inférieure à un réel ϵ positif quelconque.

$$\begin{aligned}
\mathcal{P}\left(\left|\frac{v_n}{n} - p\right| \leq \epsilon\right) &= \mathcal{P}\left(\left|\frac{v_n - np}{\sqrt{np(1-p)}}\right| \leq \epsilon\sqrt{\frac{n}{p(1-p)}}\right) \\
&= \mathcal{P}\left(|\eta_n| \leq \epsilon\sqrt{\frac{n}{p(1-p)}}\right) \\
&= F_n\left(\epsilon\sqrt{\frac{n}{p(1-p)}}\right) - F_n\left(-\epsilon\sqrt{\frac{n}{p(1-p)}}\right)
\end{aligned}$$

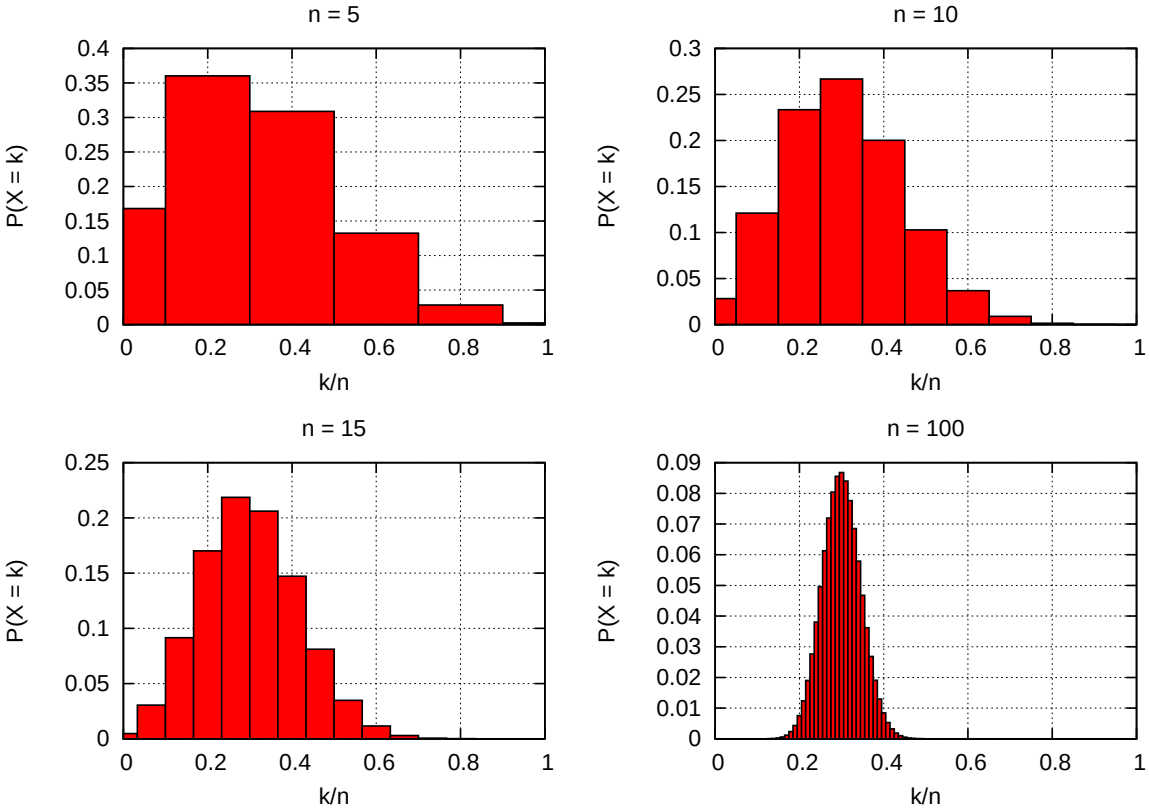

FIG. 3.2 – Illustration du théorème central limite. La probabilité de réalisation d'un événement est arbitrairement fixée à $p = 0.3$.

L'application directe du théorème de Moivre-Laplace nous donne

$$\lim_{n\to\infty} \mathcal{P}\left(\left|\frac{v_n}{n} - p\right| \leq \epsilon\right) = \frac{1}{\sqrt{2\pi}} \int_{-\epsilon\sqrt{\frac{n}{p(1-p)}}}^{\epsilon\sqrt{\frac{n}{p(1-p)}}} e^{-z^2/2} dz$$

Il est intéressant de noter que cette inégalité est une reformulation de celle de Chebyshev, et qu'elle conduit à ce que l'on nomme la « loi des grands nombres » : Si l'on répète un grand nombre de fois une même expérience aléatoire, qui a comme résultat une valeur numérique, alors la moyenne des résultats obtenus tend à se rapprocher de l'espérance mathématique de l'expérience.
De nombreux travaux de l'école russe du début du 20^{eme} siècle ont été consacrés à la détermination de l'erreur commise lors de l'utilisation d'une loi normale à la place de la loi binomiale. Nous utiliserons les résultats dûs à Uspensky [53].
Le résultat le plus fin concerne l'approximation obtenue en effectuant la « correction de continuité ». Dans cette approximation, chaque probabilité binomiale $\mathcal{P}(X = k)$ est représentée graphiquement par le rectangle élémentaire de base $[k - 1/2, k + 1/2]$

et de hauteur $\mathcal{P}(X = k)$. Dans ces conditions Uspensky montre que

$$\mathcal{P}\left(\left|\frac{v_n}{n} - p\right| \leq \epsilon\right) = \frac{2}{\sqrt{2\pi}} \int_0^{\delta} e^{-z^2/2} dz + \frac{1 - \theta_1 - \theta_2}{\sqrt{2\pi\sigma}} e^{\delta^2/2} + \Omega$$

avec, si $[x]$ désigne la partie entière de x (l'entier le plus proche inférieur ou égal à x),

$$\begin{aligned} \delta &= \frac{\epsilon n \sigma}{\epsilon n + 1/2} \\ \theta_1 &= (nq + \epsilon n) - [nq + \epsilon n] \\ \theta_2 &= (np + \epsilon n) - [np + \epsilon n] \\ \sigma &= \sqrt{npq} \end{aligned}$$

et le terme d'erreur Ω satisfait

$$|\Omega| < \frac{0.20 + 0.25\,|p - q|}{\sigma^2} + e^{-3\sigma/2}$$

pour $\sigma \geq 5$.

Connaissant n, p il est désormais possible de calculer un intervalle de confiance pour les résultats (principalement le nombre de succès) obtenus par utilisation du modèle binomial proposé.

3.5.3 Résumé et Critiques

Nous avons d'abord montré que, moyennant quelques restrictions, la section efficace obtenue lors d'un test sous radiation peut-être considérée comme une mesure de la probabilité qu'une particule déclenche un upset. Partant de l'hypothèse que cette section efficace a été obtenue correctement, nous proposons ensuite de modéliser l'expérience par une loi binomiale $\mathcal{B}(n, \sigma_{SEU})$. Enfin l'utilisation du théorème de la limite centrée nous permet de dériver un intervalle de confiance pour les résultats issus de cette modélisation.

Cependant ce modèle présente un inconvénient majeur : le temps d'injection de faute n'est référencé à aucun moment. Généralement, lors d'une expérience de mesure de section efficace, on fixe un flux f, et on mesure le nombre d'upsets apparus pendant un temps T. La fluence de l'expérience s'exprime alors par $F = f \times T$. L'utilisation de la fluence dans notre modèle « gomme » le paramètre temps. Il permet d'obtenir un nombre de fautes à injecter connaissant le nombre total de particules utilisé pendant l'expérience, mais pas quand les injecter. Nous pourrions supposer que le temps d'arrivée de chaque particule est équi-distribué selon $\Delta t = \frac{F}{f}$, mais cette hypothèse nous semble trop peu réaliste. Le but des sections suivantes est de relâcher la contrainte sur le temps d'arrivée de chaque particule et de l'intégrer à notre modèle.

3.6 Modélisation suivant une loi de Poisson

3.6.1 Distribution de Poisson

Soit une variable aléatoire X suivant la distribution binomiale $\mathcal{B}(n,p)$. Si l'on écrit l'équation 3.7 en terme d'espérance mathématique (le nombre attendu de succès) $v = np$, la probabilité d'obtenir k succès devient :

$$\mathcal{P}(X=k) = \frac{n!}{k!(n-k)!}\left(\frac{v}{n}\right)^k\left(1-\frac{v}{n}\right)^{n-k}$$

Lorsque le nombre de répétitions devient grand, la distribution approche

$$\lim_{n\to\infty}\mathcal{P}(X=k) = \frac{v^k e^{-v}}{k!}$$

qui est appelée distribution de Poisson.
Il ne faut pas voir la convergence de la loi binomiale vers une distribution de Poisson comme une contradiction du théorème limite central. En effet, pour les grands n, la distribution de Poisson tend elle aussi vers une loi normale. Ce « passage » vers une distribution de Poisson n'est qu'une étape intermédiaire de la convergence vers le loi normale. Cette approximation, connue sous le théorème de Poisson, est d'autant plus justifiée lorsque $n \gg 1$, $p \ll 1$ mais $np \sim 1$. C'est la loi dite des « événements rares ». Elle est utilisée pour modéliser des événements de faible probabilité lorsque les calculs dûs aux factorielles dans la loi binomiale deviennent intractables. Comme nous le verrons par la suite, les sections efficaces (et par prolongement les probabilités) considérées dans ce manuscrit sont toujours très petites, de sorte que cette condition $np \sim 1$ est parfaitement remplie.

3.6.2 Intégration du paramètre temps

Dans notre modèle, nous avons considéré chaque particule comme une répétition de l'expérience. Si l'on suppose que le flux f de l'expérience est constant, on peut écrire n comme :

$$n(t) = f \times t$$

Dès lors, notre modèle représente un processus de comptage, et il représente la probabilité pour qu'à l'instant t nous ayons obtenus k succès :

$$\mathcal{P}(X(t)=k) = \frac{(p\times f\times t)^k e^{-(p\times f\times t)}}{k!}$$

On nomme la constante $\lambda = p \times f$ la « cadence mathématique » du processus de comptage. Ce n'est rien d'autre que la fréquence moyenne d'apparition des événements d'intérêt.

3.6.3 Processus de Poisson

Nous avons montré que moyennant certaines conditions (la principale étant que le flux reste constant) ; la loi du processus de comptage $Xt(t)$ suit la distribution de Poisson. Si l'on rajoute les trois hypothèses suivantes, à savoir que :

- Les événements attendus se produisent indépendamment les uns des autres. C'est un processus sans mémoire.
- La loi $X(t)$ du nombre d'événement réalisés entre l'instant t et $t + \Delta t$ ne dépend que de Δt, il ne dépend pas du nombre d'événement réalisés au cours des intervalles antérieurs.
- Deux événements ne peuvent pas se produire simultanément, donc la probabilité que l'évènement se produise plus d'une fois lors de l'intervalle Δt petit est négligeable.

alors le processus de comptage est un « processus ponctuel de Poisson ». De plus, comme le flux est supposé constant, il est dit « homogène ». Cette propriété est très importante dans le cas qui nous intéresse, car elle nous permet d'affirmer que la durée séparant deux événements consécutifs est distribuée exponentiellement. La probabilité pour qu'aucun événement ne se réalise pendant l'intervalle Δt est :

$$\mathcal{P}(X(t + \Delta t) - X(t) = 0) = e^{-\lambda \Delta t}$$

Nous avons donc obtenu un moyen de déterminer les instants d'injection de fautes.

3.7 Bilan du modèle temporel proposé

Nous nous proposons dans cette section de résumer les hypothèses qui permettent de définir un espace valide d'application au modèle proposé dans ce chapitre. Nous nous plaçons dans le cadre de l'approximation RPP du volume sensible. La section efficace est définie comme :

$$\sigma = \frac{N_{SEU}}{F}$$

avec N_{SEU} le nombre d'upsets comptés lors de l'exposition du circuit à F particules distinctes mais de même nature (même L.E.T.). Sous peine que $F \gg 1$, on peut considérer la section efficace comme une mesure de la probabilité $\mathcal{P}(SEU)$ qu'à une particule de déclencher un upset :

$$\lim_{F \to \infty} \sigma = \mathcal{P}(SEU)$$

Si l'on rajoute l'hypothèse selon laquelle les particules arrivent une à une, la variable aléatoire qui compte le nombre de fautes à injecter N_{inj} suit une loi binomiale $\mathcal{B}(n, \sigma_{rad})$, où n est le nombre virtuel de particules pseudo-utilisées dans l'expérience d'injection de fautes.

En vertu du théorème limite central, nous pouvons déterminer l'erreur ϵ séparant la section efficace mesurée sous radiation σ_{rad} du résultat de la session d'injection de faute avec n particules virtuelles :

$$\sigma_{inj} = \frac{N_{inj}}{n}$$

par la relation :

$$\mathcal{P}\left(|\sigma_{inj} - \sigma_{rad}| \leq \epsilon\right) = \frac{2}{\sqrt{2\pi}} \int_0^{\delta} e^{-z^2/2} dz + \frac{1 - \theta_1 - \theta_2}{\sqrt{2\pi\xi}} e^{\delta^2/2} + \Omega$$

avec, si $[x]$ désigne la partie entière de x (l'entier le plus proche inférieur ou égal à x),

$$\begin{aligned} \delta &= \frac{\epsilon n \xi}{\epsilon n + 1/2} \\ \theta_1 &= (nq + \epsilon n) - [nq + \epsilon n] \\ \theta_2 &= (np + \epsilon n) - [np + \epsilon n] \\ \xi &= \sqrt{n\sigma_{rad}(1 - \sigma_{rad})} \end{aligned}$$

et le terme d'erreur Ω satisfait

$$|\Omega| < \frac{0.20 + 0.25\,|2\sigma_{rad} - 1|}{\xi^2} + e^{-3\xi/2}$$

pour $\xi \geq 5$.
Si l'on admet de plus que l'expérience est sans mémoire, et que le flux de particules est constant, la loi de la variable aléatoire $N_{inj}(t)$ qui compte le nombre de fautes à injecter au cours du temps t est un processus ponctuel homogène de Poisson :

$$\mathcal{P}(N_{inj}(t) = k) = \frac{(\sigma_{rad} \times \Phi \times t)^k e^{-(\sigma_{rad} \times \Phi \times t)}}{k!}$$

où Φ est le flux moyen virtuel de l'injection de fautes.
En vertu des propriétés du processus ponctuel homogène de Poisson, le temps Δt qui doit séparer deux injections de fautes est distribué selon une loi exponentielle. La probabilité pour qu'aucune injection de faute ne doive être effectuée pendant l'intervalle Δt est :

$$\mathcal{P}(N_{inj}(t + \Delta t) - N_{inj}(t) = 0) = e^{-\sigma_{rad}\Phi\Delta t}$$

Pour résumer, l'utilisation du modèle est la suivante :
– la connaissance à priori de la section efficace du composant est requise.

- l'utilisateur définit une fluence n qui représente le nombre de particules virtuelles lancées sur le circuit.
- il peut dès lors connaître la précision pour un seuil de confiance donné de la session d'injection de fautes, c'est à dire l'écart absolu entre la section efficace qu'il va mesurer en sortie d'injection de fautes et celle réelle du composant.
- l'utilisateur doit ensuite définir un flux moyen de particules virtuelles Φ.
- Il suffit alors d'injecter des fautes à des temps distribués selon la loi exponentielle pour simuler les conditions réelles d'un test sous radiation.

L'implantation de ce modèle sera le sujet du chapitre suivant.

3.8 Localisation de l'injection de fautes

Le but de cette section est de présenter la façon dont est choisie la cible d'injection de fautes. Nous étudierons deux cas distincts :

- Le cas simple où toutes les cibles ont la même section efficace (cas typique d'une SRAM par exemple) ;
- Le cas où plusieurs familles de cibles avec des sections efficaces différentes coexistent (cas d'un microprocesseur avec un banc de registre et de la mémoire cache).

3.8.1 Cas Simple

Puisque tous les bits ont la même section efficace, c'est à dire la même probabilité d'être corrompus, il suffit d'en choisir un au hasard. C'est un tirage d'événement équiprobables. On peut assimiler ce cas à un tirage au sort avec remise de boules identiques.

3.8.2 Cas Général

Dans ce cas, on utilise les propriétés de superposition des processus de Poisson. Soit $\mathcal{P}_1$ et $\mathcal{P}_2$ deux processus de Poisson ayant pour paramètres respectifs λ_1 et λ_2. Alors le processus $\mathcal{P}$ combinant $\mathcal{P}_1$ et $\mathcal{P}_2$ est un processus de Poisson de paramètre $\lambda_1 + \lambda_2$.

Concrètement, pour chaque « famille » de bits (une famille représentant un ensemble de bits de même section efficace), on génère une liste de temps d'injection de fautes selon la loi exponentielle avec pour paramètre leurs sections efficaces respectives. Il suffit ensuite de combiner les listes en une seule. La liste ainsi obtenue contiendra des temps d'injection de fautes pour chaque famille de bits. Lorsque la liste est déroulée pendant l'injection de fautes, elle donne les temps d'injection pour

chaque famille. Une fois celle ci identifiée, on se ramène au choix d'un bit quelconque parmi cette famille (cas simple).

Chapitre 4

Implantation et Premiers Résultats

CE chapitre décrit les dévelopements techniques réalisés au cours de cette thèse ainsi que les premiers éléments de comparaison entre les mesures prises sous radiations, et les résulats d'injection de fautes.

Nous présentons tout d'abord le processeur qui a servi de véhicule de test pendant ces travaux.

La méthode d'obtention des sections efficaces statiques du circuit est ensuite décrite.

Une grande partie du chapitre est ensuite dédiée à l'implantation de l'injection de fautes. Le logiciel ainsi que le matériel supportant ce type de manipulation est décrit.

Finalement, nous terminons ce chapitre par une discussion sur les premiers résultats d'injection de fautes et de mesures sous radiation. Ces données confortent la validité du modèle élaboré dans le chapitre précédent.

4.1 Présentation du Véhicule d'Etude

Le composant qui a servi de démonstrateur pendant la plupart des expériences de cette thèse est le processeur LEON. Il est disponible en téléchargement sous la forme d'une description en langage VHDL synthétisable. Ce coeur, conçu et maintenu par Gaisler Research [54] sous contrat avec l'agence spatiale européenne (*European Space Agency*, ESA), est placé sous licence LGPL [55].

Le processeur appartient à la famille RISC (*Reduced Instruction Set Computer*) et a été certifié conforme à l'architecture SPARC V8 (*Scalable Processor Architecture*). Le jeu d'instruction SPARC a été conçu par David Patterson (inventeur du

RISC) et William Joy (responsable en grande partie de l'implantation de la version *Berkeley* d'UNIX et co-fondateur de Sun microsystems). L'architecture SPARC est une norme IEEE-1754 [56].

Le LEON embarque les fonctionnalités suivantes : caches instructions et données séparés (architecture Harvard), contrôleur d'interruption, unité de débogage, deux timers, deux UARTs, un port d'entrées sorties génériques de 16 bits et un contrôleur de mémoire.

4.1.1 Unité de Calcul Entier

L'unité de calcul entier est en mesure d'exécuter toutes les instructions SPARC V8. Elle comporte un *pipeline* de cinq étages (*Fetch*, *Decode*, *Execute*, *Memory*, *Write*).

Une des particularités de l'architecture SPARC est la façon dont sont accessibles les registres du processeur (l'ensemble des registres est appelé *Register File*, ou *Regfile*). Ceux-ci sont organisés en groupes de 24 registres de 32 bits chacun. Un groupe est appelé fenêtre.

Une fenêtre contient 8 registres d'entrée (*ins*, `%i`), 8 registres locaux (*locals*, `%l`) et 8 registres de sortie (*outs*, `%o`). La particularité vient du fait que les registres `%o` d'une fenêtre sont les `%i` de la fenêtre précédente. Seuls les 8 registres locaux appartiennent totalement à une fenêtre.

Des instructions spécifiques (`save`, `restore`, *return from trap* `rett`), des accès directs au registre de fenêtre (*Current Window Pointer*, CWP) ou les exceptions et interruptions (*traps*) permettent de passer d'une fenêtre à l'autre. A ces fenêtres s'ajoutent 8 registres globaux (*globals*, `%g`), accessibles de n'importe quelle fenêtre (en héritier de la tradition RISC, le registre $\texttt{\%g}_0$ contient toujours 0). Cette organisation en fenêtre est illustrée sur la figure 4.1.

4.1.2 Mémoire Cache

La correspondance cache - mémoire externe se calcule selon l'algorithme *direct mapped* : à une adresse correspond un emplacement unique en mémoire cache :

Adresse en cache = (Adresse en mémoire externe) *modulo* (taille du cache)

La cohérence du cache données dépend de l'action en cours : lecture ou écriture. Lors d'une lecture :

- Si la donnée n'est pas présente en mémoire cache (*cache-miss*), elle est lue en mémoire externe et copiée en mémoire cache ;

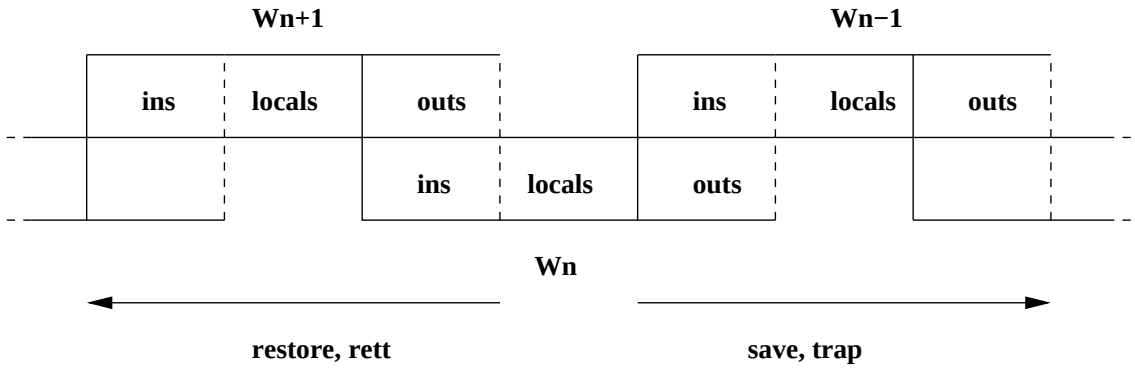

FIG. 4.1 – Système de fenêtres de l'architecture SPARC. W_n représente la fenêtre n.

– Si la donnée est en mémoire cache (*cache-hit*) elle est lue à partir du cache.

Lors d'une écriture :

– Si la donnée vient remplacer une donnée déjà en mémoire cache, la nouvelle valeur est copiée en mémoire cache et en mémoire externe (*write-through*) ;
– Si la donnée ne remplace pas une donnée déjà en cache, la nouvelle donnée est uniquement écrite en mémoire externe (*no allocate on write-miss*).

4.1.3 Détails sur le Prototype

Le circuit a été synthétisé, placé et routé au laboratoire TIMA. Il a été fondu au moyen du service CMP du laboratoire sur une technologie ST microelectronics 0.25 μm. La fréquence de fonctionnement standard est de l'ordre de 40 MHz pour une tension d'alimentation coeur et entrées-sorties de 2.5 V.

Certaines instructions SPARC permettent 3 accès concurrentiels sur les registres. Pour permettre l'exécution de ce type d'instructions en un minimum de coups d'horloge, les registres sont implantés à l'aide d'une mémoire RAM triple port. Ce type de mémoire n'étant pas disponible dans la bibliothèque ST au moment de la conception du circuit, la file de registres a été implantée à l'aide de deux mémoires RAM double port comme le montre la figure 4.2. Cette particularité aura une importance lors de l'écriture des programmes de test statique. Finalement, notons que l'implantation choisie comporte 8 fenêtres, et les caches données et instructions ont une taille de 4 KiB chacun.

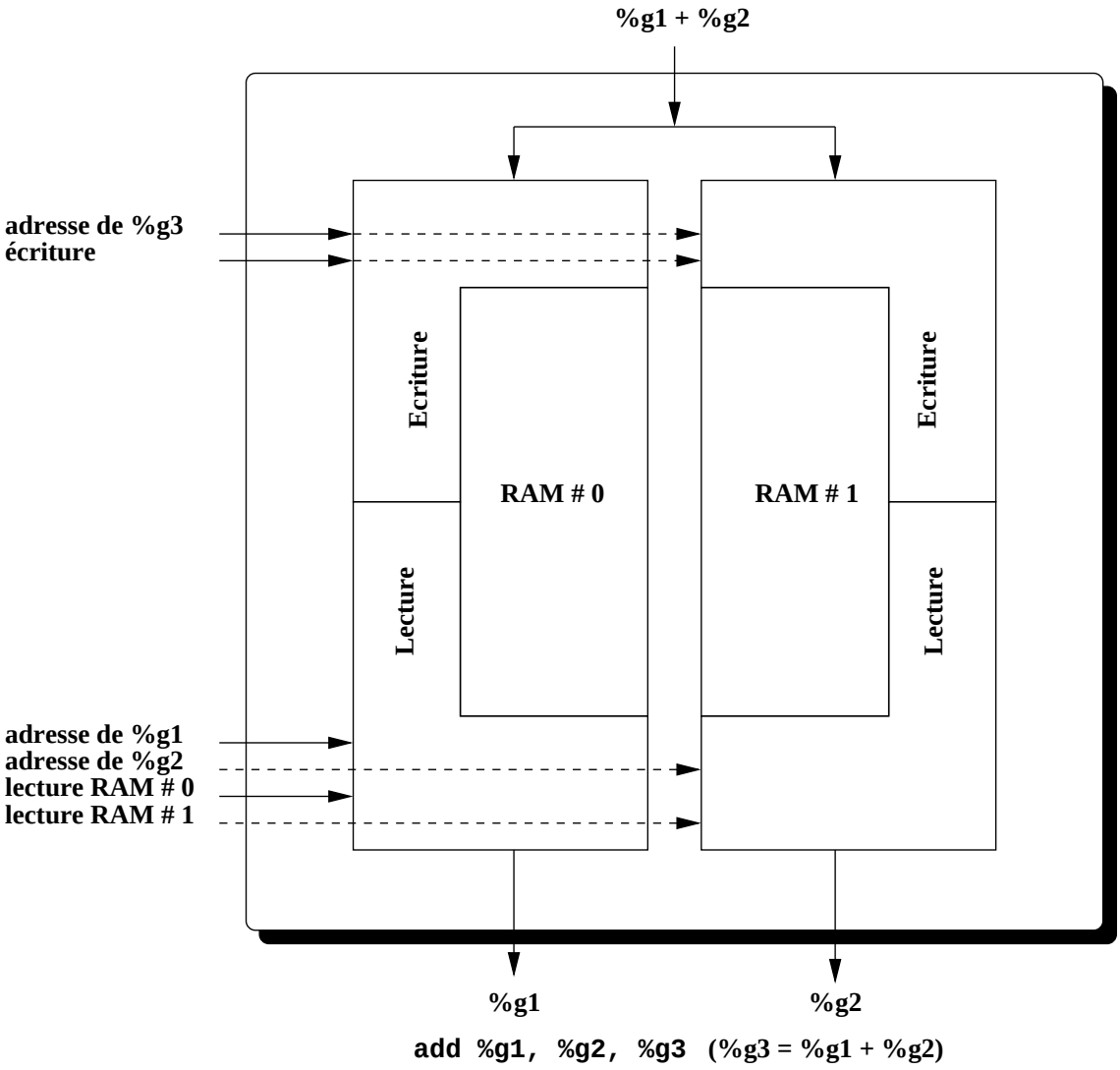

FIG. 4.2 – Implantation de la file de registre. L'instruction exécutée est : `add %g1, %g2, %g3` (%g3 = %g1 + %g2).

4.2 Obtention des Sections Efficaces

Le jeu d'instruction du processeur nous permet d'adresser et donc de tester cinq types d'éléments mémoires différents :

- La file de registres ;
- Les blocs du cache données ;
- Les index (*tag*) du cache données ;
- Les blocs du cache instructions ;
- Les index (*tag*) du cache instructions.

Les bits des caches ayant été regroupés pour des raisons pratiques, cela nous conduit à l'écriture de trois benchmarks permettant de mesurer la section efficace statique de ces différents bits :

- `regfile` : permettant de mesurer la section efficace des registres globaux et de ceux des fenêtres ;
- `dcache` : permettant de mesurer la section efficace du cache données (en distinguant celle des blocs de celle des tags) ;
- `icache` : permettant de mesure la section efficace du cache instructions (en distinguant celle des blocs de celle des tags).

Nous commencerons par discuter de l'organisation de ces programmes et des différentes phases qui les constituent, tout en notant leur similarités. Les techniques utilisées pour contrôler l'exécution correcte du programme par le processeur seront présentées ensuite. Quelques remarques propres au jeu d'instructions du processeur sont données. Nous poursuivrons par une discussion sur la méthode utilisée pour maximiser l'exposition du processeur vis-à-vis des autres phases du programme. Finalement la méthode d'analyse des résultats de mesure sous radiation est présentée.

4.2.1 Organisation des Programmes

En simplifiant à l'extrême, les trois benchmarks partagent la même philosophie :

- Tout d'abord, le processeur démarre (*boot*) : Les interfaces mémoire, la pile, etc sont configurées ;
- Le processeur commence ensuite à exécuter le benchmark :

  1. Il y a toujours une première phase où le processeur écrit un motif connu dans les bits mémoire sous test (*write*) ;
  2. Dans une seconde phase, le processeur attend pendant que ces bits sont exposés à un faisceau de particules approprié (*wait*) ;
  3. Dans la troisième et dernière phase, le processeur écrit le contenu de ces bits en mémoire externe pour vérification ultérieure par l'expérimentateur (*dump*).

– Finalement, et de manière optionnelle (mais fortement recommandée) le processeur signale au « monde extérieur » que l'exécution est terminée et que les résultats sont disponibles.

Notons que cette procédure, « lourde » bien que simple **minimise** la participation du processeur dans l'obtention des résultats. Beaucoup d'expérimentateurs sont tombés dans le piège de l'écriture d'une boucle de test plus « élégante », notamment en faisant participer le processeur à la vérification des résultats, de manière à minimiser les transferts entre le testeur et l'expérimentateur, et donc maximiser le temps d'exposition du processeur. Une telle approche n'est pas correcte, n'oublions pas le vieil adage qui stipule que *l'on ne peut être juge et partie à la fois*, surtout lorsque le « juge » est bombardé de plusieurs centaines de particules par seconde.

Cette similitude dans les programmes nous a poussé à introduire la notion de « base de code commune » (*common code base*). Les trois benchmarks partagent cette base de code et diffèrent seulement dans la boucle de test effectuée. Pour résumer, les benchmarks partagent :

– La partie configuration, en structure et parfois en paramètres ;
– L'appel à la fonction de test ;
– Le retour de la fonction de test ;
– La gestion des exceptions.

La structure de la base de code commune est représentée sur la figure 4.3

4.2.2 Contrôle du Flot

Le découpage naturellement apparu pendant la présentation du flot des benchmarks dans la section 4.2.1 nous a permis d'implanter un contrôle du flot d'exécution du processeur. Il suffit de compter le nombre de cycles pour chaque phase et de les comparer avec une valeur de référence pour déterminer si quelque chose d'anormal s'est passé pendant l'exécution du programme. Nous présentons ci-dessous chacun des compteurs utilisés :

– `booting` : ce compteur mesure le temps entre le reset du processeur et le saut vers le benchmark. Si cette valeur est incorrecte, le processeur est probablement mal configuré, et les résultats obtenus en fin d'exécution sont souvent corrompus ;
– `waiting` : contrairement aux autres compteurs, celui-ci ne mesure pas de phase du benchmark. Il sert simplement à faire attendre le processeur à la fin de sa phase d'écriture. Lorsque le compteur atteint une valeur prédéfinie, il signale au processeur que celui-ci peut commencer à écrire les résultats en mémoire externe. Il sert donc à régler le temps d'attente du processeur ;
– `running` : ce compteur mesure la durée d'exécution du benchmark (write, wait et dump hors boot) ;

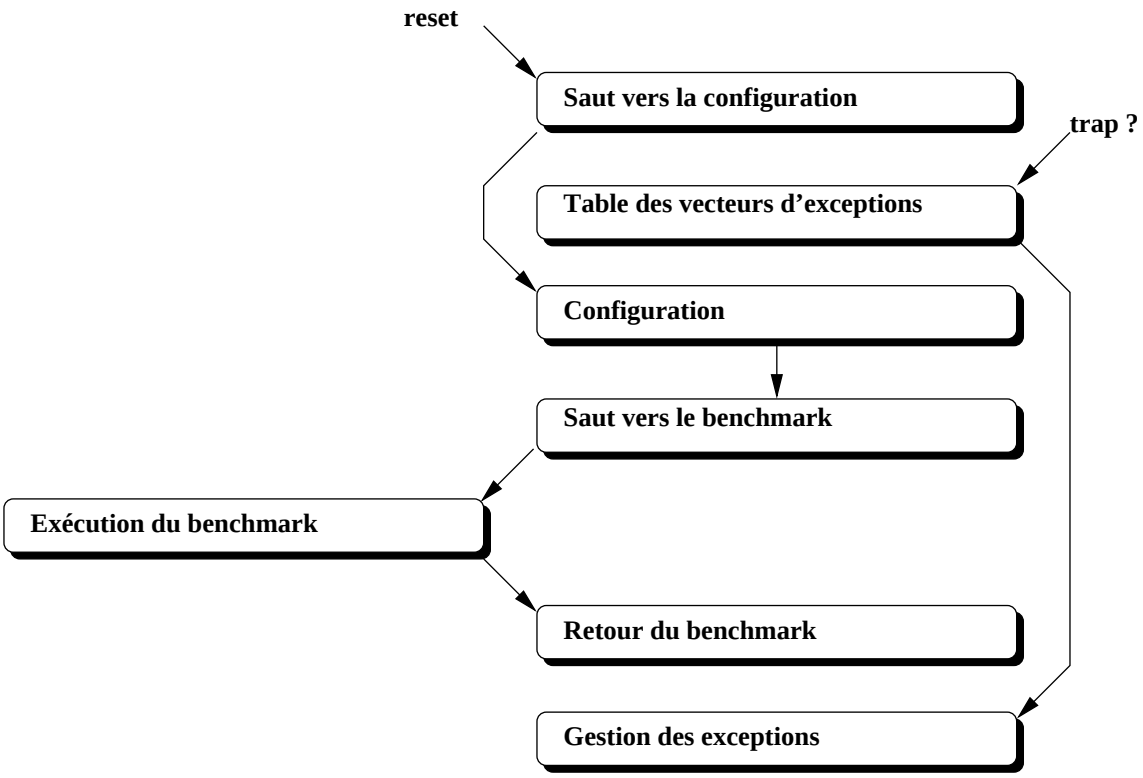

FIG. 4.3 – Structure de la base de code commune aux benchmarks.

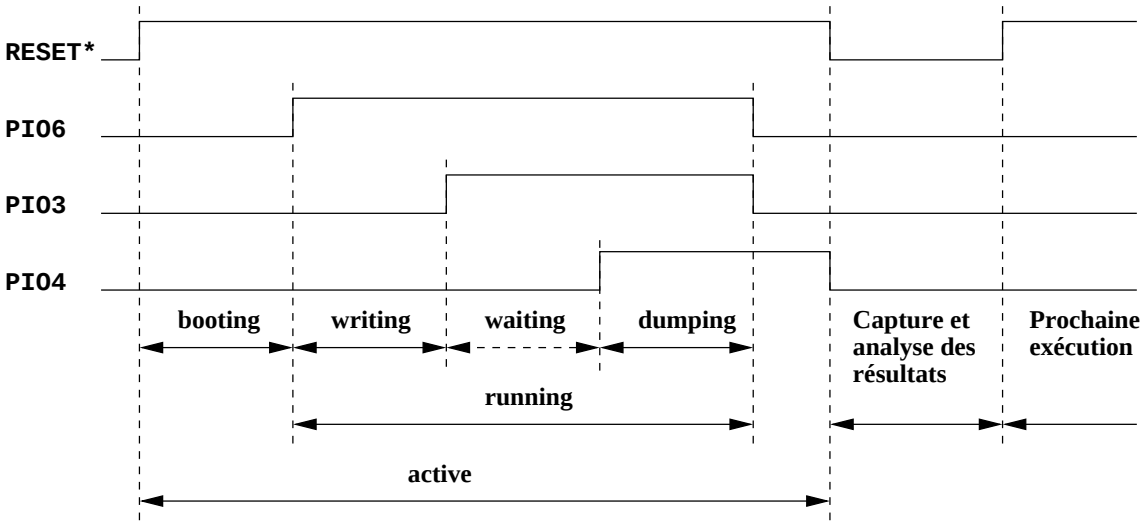

FIG. 4.4 – Découpage de l'exécution en phases et compteurs associées.

– `active` : ce compteur mesure le temps écoulé entre deux *reset* du processeur. Il correspond au temps d'exécution du benchmark, additionné du temps de configuration, et du temps où le processeur était inactif mais où le testeur récupérait les résultats de l'exécution précédente.

Comme nous le verrons par la suite, ces compteurs permettent une connaissance complète de la durée de toutes les phases du benchmark, ainsi qu'une détermination précise de la vraie fluence de l'expérience (c'est à dire la quantité de particules qu'à reçu le processeur **alors qu'il était en état de prise de mesure**).

Le port PIO (*Programmable Inputs and Outputs*) du processeur est utilisé pour activer les séquences de comptage de chaque compteur : lorsque le processeur atteint une phase du benchmark, il active une de ses broches et le compteur correspondant démarre ou s'arrête selon le cas.

De manière identique, la boucle d'attente (wait) est implantée de la façon suivante : le processeur lit le niveau logique de l'une de ses broches. Tant que ce niveau n'est pas celui attendu, il continue la lecture et ne passera à l'étape suivante que lorsque le niveau logique sera celui voulu. Cette technique de rebouclage jusqu'à ce qu'une condition soit rencontrée s'appelle *polling*.

La figure 4.4 résume ce découpage en phase et les compteurs associés.

4.2.3 Remarques sur l'Ecriture des Programmes

Nous présentons dans cette section quelques « pièges » rencontrés pendant l'écriture des programmes de test. Avant d'en discuter il convient de revenir sur la nature

du processeur et son implantation.

Le processeur LEON est un SPARC, ce qui implique que son jeu d'instruction est réduit, et que toutes les instructions ont la même taille (32 bits dans le cas du LEON). Par exemple il n'existe pas d'instruction permettant de copier le contenu d'un registre dans un autre, pas plus qu'il n'existe d'instruction pour écrire directement une valeur dans un registre (une instruction faisant 32 bits, elle ne peut pas à la fois contenir le code d'instruction, le registre cible et la valeur de 32 bits). Les concepteurs de l'architecture RISC ont alors introduit la notion d'instruction synthétique : une « instruction » qui n'existe pas dans le jeu d'instruction du processeur mais qui peut être émulée par une suite d'instructions réelles. Le programmeur emploie ces instructions synthétiques, qui sont remplacées ensuite par l'assembleur. A titre d'exemple nous donnons les instructions synthétiques `mov %r`$_s$`, %r`$_d$ et `set #VALEUR, %r`$_d$.

`mov` permet de copier le contenu du registre source `%r`$_s$ vers le registre destination `%r`$_d$. L'instruction qui est réellement exécutée est `add %r`$_s$`, %g`$_0$`, %r`$_d$ (`%r`$_d$ `= %r`$_s$ `+ 0`).

`set` permet d'écrire une valeur (`#VALEUR`) de 32 bits dans le registre de destination `%r`$_d$. Comme les instructions ont une taille 32 bits, elles ne peuvent contenir le mot entier. `#VALEUR` est découpé en un mot de 22 bits (`#VALEUR_MSB`) et un mot de 10 bits (`#VALEUR_LSB`). Les instructions réellement exécutées sont :

```
sethi  #VALEUR_MSB, %rd        (%rd[31 :10] = #VALEUR_MSB, %rd[9 :0] = 0)
or     %rd, #VALEUR_LSB, %rd   (%rd = %rd + #VALEUR_LSB)
```

La façon la plus naturelle d'écrire le motif de test dans les bits mémoires du processeur serait d'employer l'instruction synthétique `mov` :

```
...
set #MOTIF, %g1
mov %g1, %g2
mov %g1, %g3
mov %g1, %g4
...
```

Le problème de cette séquence est qu'elle assume que le contenu du registre `%g`$_1$ est fixe. Sous radiation ceci n'est plus vrai, et un upset sur `%g`$_1$ pendant la phase d'écriture conduirait à une écriture corrompue lors de l'exécution des instructions suivantes.

La façon correcte de réaliser cette écriture est :

```
...
set #MOTIF, %g1
```

```
set #MOTIF, %g2
set #MOTIF, %g3
set #MOTIF, %g4
...
```

même si l'instruction synthétique `set` est réalisée en deux instructions au lieu d'une unique pour `mov`. De même lors des boucles d'écriture, la valeur du registre contenant le motif à écrire doit être rafraîchie à chaque itération, et non fixée avant de commencer la boucle.

L'utilisateur doit aussi faire attention à la façon dont les calculs d'adresse de lecture/écriture sont faits. Il faut en général préférer les adresses calculées lors de la compilation plutôt que lors de l'exécution. Voici un exemple (`st %r`$_s$`, [%r`$_d$`]` écrit le contenu de `%r`$_s$ à l'adresse contenue dans `%r`$_d$) :

Mauvais :
```
...
set #ADRESSE, %g1
st %g2, [%g1]
add %g1, 4, %g1
st %g3, [%g1]
...
```

Bon :
```
...
set #ADRESSE, %g1
st %g2, [%g1]
set #ADRESSE + 4, %g1
st %g3, [%g1]
...
```

Pour finir, le dernier « piège » que nous avons rencontré est dû à l'implantation de la file de registre. Supposons que le registre `%g`$_1$ soit corrompu par un upset. Alors les deux additions (notez l'ordre des opérandes) :

```
add %g1, %g2, %g3          (%g3 = %g1 + %g2)
```
et
```
add %g2, %g1, %g3          (%g3 = %g2 + %g1)
```

ne donnent pas le même résultat. Selon que la RAM 0 ou 1 est touchée, l'une des deux opérations sera juste, l'autre fausse. Afin de ne pas mélanger les mesures de section efficaces d'une RAM ou de l'autre, il faut s'assurer que chaque registre soit toujours en même position dans les instructions. Idéalement, il faudrait tester les deux mémoires (même si les sections efficaces sont probablement identiques).

4.2.4 Dimension de la Boucle d'Exposition

Afin de maximiser le débit de résultats de l'expérience, il faut que le processeur soit exposé et en mode « prise de mesure » le plus de temps possible. Les phases annexes du programme de test (boot, write, dump, récupération des résultats par l'expérimentateur) doivent donc être minimisées par rapport à la boucle d'attente

Phase	Durée		
boot	t_{boot}	=	29.1 μs
write	t_{write}	=	76.7 μs
wait	t_{wait}	=	$N \times 4.1$ μs
dump	t_{dump}	=	133.4 μs

TAB. 4.1 – Exemple de durée de chaque phase du benchmark `regfile`. La durée « atomique » de la boucle d'attente est de $4.1\mu s$, et N représente le nombre de fois où cette boucle est exécutée.

(wait). Une fois cette opération effectuée, la boucle d'attente doit être ajustée de façon à ce que le processeur passe la plupart du temps exposé, avec le motif de test écrit.

A titre d'exemple, le tableau 4.1 récapitule les durée de chaque phase du benchmark pour le programme `regfile` (le DUT fonctionne à 40 MHz, les caches sont désactivés).

La fraction de temps R passé dans la boucle d'attente (et donc de mesure) s'exprime par :

$$R = \frac{t_{wait}}{t_{benchmark}} = \frac{t_{wait}}{t_{boot} + t_{write} + t_{wait} + t_{dump}}$$

La fonction $R = f(N)$ est tracée sur la figure 4.5 avec les valeurs du tableau 4.1. Pour $N > 50000$, le processeur passe plus de 99.9 % du temps dans la boucle d'attente. En raison de la nature séquentielle des phases write et dump, certains registres passent plus de temps exposés que d'autres, ce qui devrait être pris en compte dans le calcul de section efficace. En ajustant N correctement, ces effets de bords peuvent être négligés ($t_{benchmark} \simeq t_{waiting}$).

4.2.5 Protocole et Analyse des Résultats

Pour un LET donné, les mesures sous radiation se sont déroulées comme suit :
- Choix du benckmark, choix du paramètre N ;
- Quelques exécutions du benchmark sont lancées à vide, sans radiation afin de vérifier que l'expérience fonctionne parfaitement (les compteurs de contrôle de flot doivent rester relativement constants) ;
- Choix d'un flux de particules Φ_{exp} et d'une fluence F_{exp} ;
- Début d'exposition, alors que le processeur fonctionne toujours et exécute des itérations du benchmark ;
- Fin d'exposition lorsque la fluence est atteinte. On obtient de la dosimétrie du faisceau la durée de l'expérience T_{exp}.

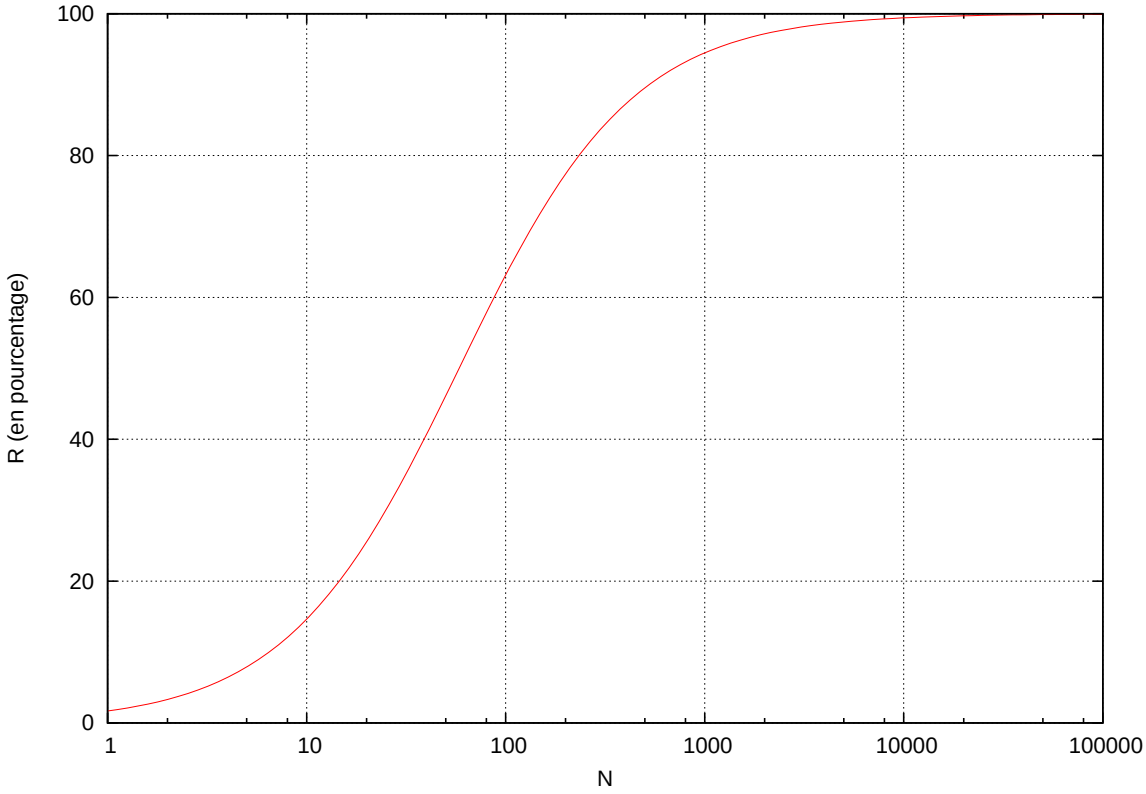

FIG. 4.5 – Pourcentage de temps passé avec les registres exposées en fonction du nombre d'exécutions de la boucle d'attente.

– Le processeur exécute toujours du code afin de vérifier qu'il retrouve un état de fonctionnement normal ;
– Arrêt du processeur.

L'intégrité du faisceau est vérifiée en calculant

$$\Phi_{moy} = \frac{F_{exp}}{T_{exp}}$$

qui doit être sensiblement égal aux flux initial choisi Φ_{exp}.

Les exécutions du benchmark hors-radiation doivent être retirées de l'analyse (on ne conserve les exécutions qu'à partir du premier upset enregistré jusqu'au dernier).

Pour chaque compteur de contrôle de flot (`booting` et `running`, `active` étant traité à part) la valeur moyenne m ainsi que la variance σ^2 sont estimées à l'aide des estimateurs sans biais $\hat{m}$ et $\hat{\sigma}^2$:

$$\begin{aligned} \hat{m} &= \frac{1}{n}\sum_{i=1}^{n} x_i \\ \hat{\sigma}^2 &= \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \hat{m})^2 \end{aligned}$$

où x_i représente la valeur du compteur pour l'exécution i.

Toute valeur de compteur en dehors de l'intervalle $[\hat{m} - \sqrt{\hat{\sigma}^2}; \hat{m} + \sqrt{\hat{\sigma}^2}]$ entraîne le retrait de l'exécution concernée de l'analyse, pour inspection manuelle ultérieure.

A la suite de ce tri, nous obtenons une liste d'exécutions où le flot d'exécution de chaque benchmark est relativement correct, et pendant lesquels le processeur a été exposé aux radiations. La dernière étape avant le calcul de la section efficace est de calculer le temps d'exposition **réel** du processeur et donc la fluence réelle de l'expérience.

En effet, entre deux exécutions du benchmark, les résultats sont envoyés sur l'ordinateur de contrôle. Ce temps est variable en fonction de la latence du lien de communication ordinateur-testeur, de la charge du système d'exploitation, et de la quantité de résultats à déplacer. Afin d'illustrer ces propos, le lecteur peut se reporter à la figure 4.6 où est tracée la valeur (`active - running`) qui correspond exactement au temps de copie des résultats du testeur vers l'ordinateur. Comme le montre cette figure, le temps de copie est hautement variable et doit être pris en considération pour déterminer le temps d'exposition réel du processeur. Nous utilisons donc le compteur `running` (qui représente le temps d'exposition lorsque la boucle d'attente a été correctement ajustée) pour déterminer le temps d'exposition utile du processeur T_{utile} et donc la fluence utile F_{utile} de l'expérience :

$$\begin{aligned} T_{utile} &= \sum_{i=1}^{n} t_{running,i} \\ F_{utile} &= \Phi_{moy} \times T_{utile} \end{aligned}$$

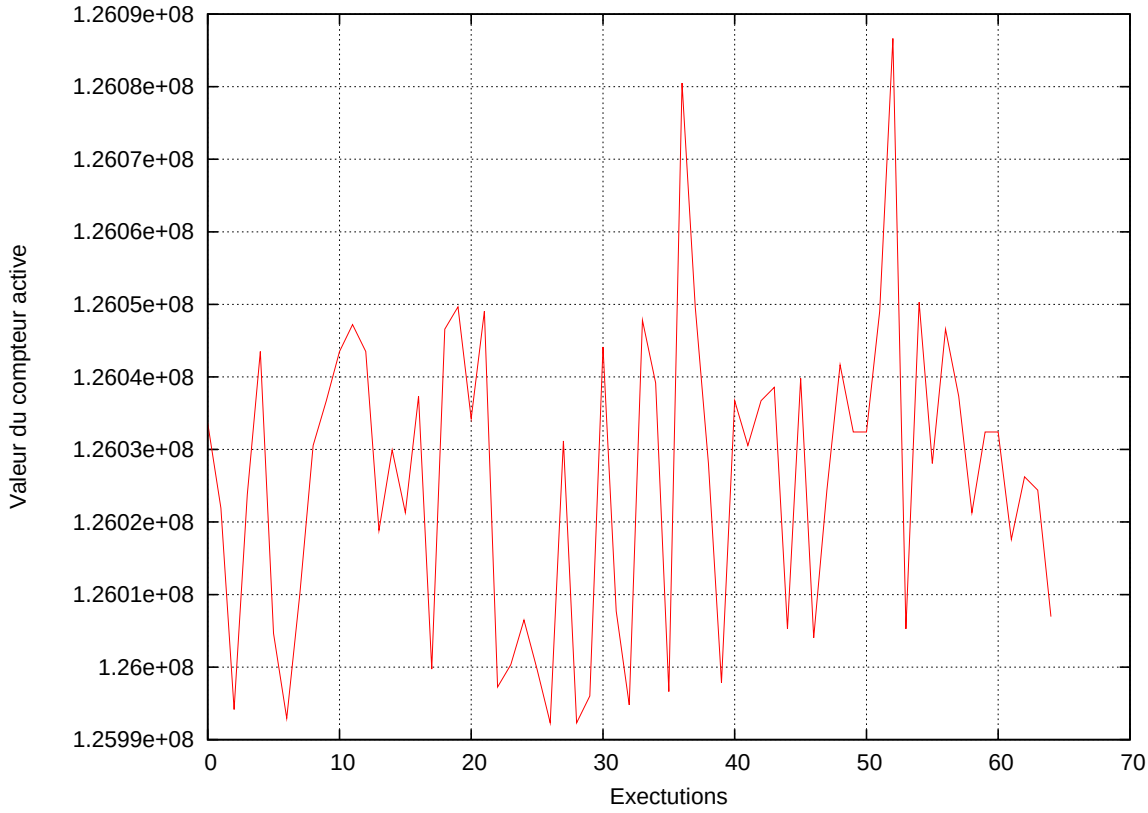

FIG. 4.6 – Exemples de valeur du compteur `active - running` lors d'un test sous radiation. L'ordonnée est mesurée en cycle d'horloge.

où n représente le nombre d'exécutions après épuration des exécutions sans particules, et celles dont le flot est non-conforme, et $t_{running}$ est la valeur du compteur `running` pour ces exécutions.

La section efficace peut alors être déterminée par :

$$\sigma = \frac{N_{SEU}}{F_{utile}}$$

où N_{SEU} est la somme des SEU enregistrés par exécutions, en ne considérant que les exécutions conformes :

$$N_{SEU} = \sum_{i=1}^{n} N_{SEU,i}$$

4.3 Injection de Fautes

Nous présentons dans cette section comment la routine CEU a été adaptée à l'architecture du processeur LEON. Connaissant les trois paramètres nécessaires à l'injection de fautes : quel mot, quel bit et quel moment, nous nous intéressons ensuite à la génération pseudo-aléatoire de ces trois variables. La dernière partie s'attache a présenter le choix d'un générateur de nombre pseudo-aléatoire, et son amélioration.

4.3.1 Implantation de la Routine CEU

Gestion des Interruptions

Avant d'expliquer plus en détail de la routine CEU, il convient de présenter le modèle d'interruption et d'exceptions (*traps*) de l'architecture SPARC. Il existe quelque légères différences entre une interruption et une exception, mais pour ce qui nous intéresse, elles sont équivalentes.

Lorsque le processeur reçoit une interruption, il doit consulter une table qui lui permet de trouver l'adresse du code gérant l'interruption. L'adresse de base de cette table est contenue dans le registre `%tbr` (*Trap Base Register*). Le déplacement (*offset*) dans cette table est donnée par la nature de l'interruption (*trap type*, `tt`) de sorte que l'adresse du vecteur d'interruption est donnée par :

$$\text{vecteur d'interruption} = \%tbr + tt \times 16$$

Le `tt` est décalé de 4 bits à gauche (multiplication par 16) car un vecteur d'interruption ne peut être composé que de 4 instructions au plus. Généralement, le

vecteur implante un saut vers une routine de traitement d'interruption qui elle n'est plus limitée en taille.

La dernière spécificité de la gestion des exceptions est que lorsque le processeur reçoit une interruption, il change de fenêtre automatiquement pour passer à la fenêtre précédente. Dans cette nouvelle fenêtre, l'adresse de l'instruction interrompue est copié dans le registre `%l1` (c'est le *Program Counter*, `PC`) et l'adresse de la prochaine instruction qui aurait dû être exécutée dans `%l2` (*next Program Counter*, `nPC`).

Gestion des Fenêtres et de la Pile

Rappelons que l'instruction `save` ainsi que les interruptions déclenchent une rotation de la fenêtre N vers la fenêtre $N-1$. L'instruction `restore` ainsi que le retour d'interruption (`rett`) font l'opération opposée.

Lorsque le processeur doit appeler une fonction, il exécute la plupart du temps un `save`. De cette façon, la fonction possède une fenêtre vierge. Selon [56], la fonction doit trouver :

- Ses paramètres d'appels dans `%i0` et jusqu'à `%i5` ;
- Une adresse de pile vierge dans `%i6`. Ce registre est renommé *Frame Pointer* (`%fp`). De ce fait, `%fp` réfère **toujours** au registre `%i6` de la fenêtre courante.

Une fonction doit passer ses valeurs de retour dans ses registres *ins* (qui sont les *outs* de l'appelant avant d'exécuter un `restore`).

Un deuxième pointeur de pile existe : c'est le *Stack Pointer* (`%sp`). Il est contenu dans le registre `%o6`. De cette façon `%sp` de la fenêtre N est `%fp` de la fenêtre $N-1$. Dans l'architecture SPARC, la pile grandit selon des adresses décroissantes. Le registre `%fp` est donc l'adresse supérieure de pile (la « base ») qu'il ne faut pas dépasser sous peine de s'aventurer dans la pile de la fonction appelante. Toute adresse sous `%fp` est donc utilisable par la fonction. Le sommet de la pile (`%sp`) doit donc toujours être à jour de manière à ce qu'il devienne une base correcte pour la pile d'une fonction appelée : ce mécanisme de pile à deux pointeurs permet de générer une pile locale à la fonction.

Puisque une interruption entraîne une rotation de fenêtre dans le même sens qu'un appel de fonction, son `%fp` (le « vieux » `%sp`) représente donc une base saine pour une nouvelle pile. Nous utiliserons cette propriété pour notre routine CEU.

Routine CEU

Nous admettons ici que l'interruption a été déclenchée (le « quand » a été choisi). Il reste à choisir le « où ». Celui-ci se décompose en un mot et un bit parmi ce mot. Puisque l'interruption peut intervenir à n'importe quel moment, il est impossible de prédire la valeur des pointeurs de piles : Les paramètres de la routine doivent donc se trouver à un endroit fixe, en mémoire externe. Avant d'explorer plus en détail la

routine, précisons que les registres %g5, %g6 et %g7 sont des registres réservés pour le compilateur : nous pouvons donc les utiliser sans risque.

La routine CEU se découpe en 4 étapes :

1. Sauvegarde des valeurs critiques, qui permettent un retour d'interruption ;
2. Sauvegarde du contexte du processeur en mémoire externe ;
3. Récupération des paramètres CEU et corruption du contexte sauvegardé ;
4. Restauration du contexte corrompu et retour au programme.

Etape 1 La première opération consiste à copier %fp dans %g6 et %g7. A partir de ce moment nous nous servirons de %g6 et %g7 à la place de %fp et %sp. La raison de cette substitution apparaitra par la suite. Le status du processeur (%psr) est sauvegardé en %g6. Il contient entre autre les codes de condition arithmétiques, et comme la routine CEU exécute des opérations arithmétiques, ils vont être modifiés. Vient ensuite la sauvegarde des valeurs de retour PC et nPC (%l1 et %l2) en %g6 - 4 et %g6 - 8. A chaque sauvegarde, %g7 est décrémenté d'autant : la pile croît, et %g7 donne l'adresse de la dernière sauvegarde.

Etape 2 Le contexte du processeur est sauvegardé, cette fois avec %g7 comme référence. Comme la routine est amenée à changer de fenêtres, nous sommes dans l'impossibilité de faire des références explicites à %sp et %fp puisque ceux-ci sont locaux à la fenêtre en cours. C'est la raison pour laquelle nous avons utilisé le couple %g6 et %g7. Lors de cette étape, %g7 est décrémenté, puis la sauvegarde est faite à l'adresse pointée par %g7. Les registres globaux 1 à 4, les fenetres 0 à 8, le cache donnée (blocs et tags) et le cache instruction (blocs et tags) sont sauvegardés. A la fin de cette étape, %g7 est au sommet de la pile de sauvegarde du contexte.

Etape 3 Le processeur va ensuite chercher les 2 paramètres mot et masque. mot est en fait un décalage par rapport à %g7 et l'adresse du mot à corrompre est donc %g7 + mot. masque est un masque avec un seul bit à 1. Le masque est appliqué sur le mot à corrompre à l'aide d'un ou exclusif, puis le mot corrompu est réécrit à son adresse.

Etape 4 L'étape 4 est une symétrie parfaite de l'étape 1 et 2. Le contexte est restauré, ainsi que les valeurs critiques, et la main est rendue au programme principal.

La figure 4.7 schématise le déroulement des étapes 1 à 3.

4.3.2 Génération des Paramètres

Nous nous intéressons ici à la génération aléatoire de l'instant d'injection, du mot et du bit cible.

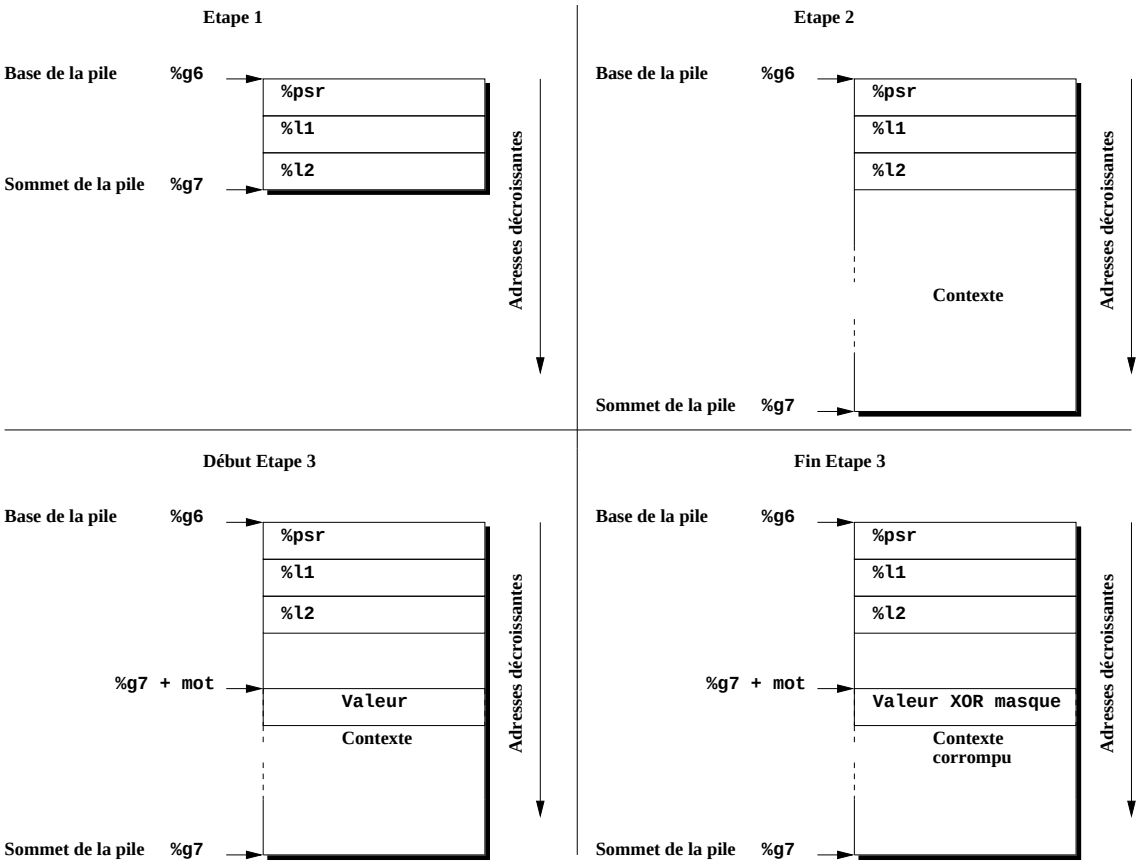

FIG. 4.7 – Etapes de la routine CEU.

Obtention du Mot Cible et du Masque

Pour N mots de même section efficace, le choix se résume à un tirage unique de 1 parmi N. Il suffit de générer un nombre aléatoire dans l'intervalle $[0 : 1)$, de le multiplier par $N-1$ et d'en prendre la partie entière. En langage C, `my_rand()` étant la fonction qui retourne un nombre entre 0 et 1, `floorf()` la fonction qui retourne la partie entière d'une nombre à virgule flottante et `(unsigned int)` l'opération qui permet de transformer le type d'une variable en entier non signé (*cast*), cela s'écrit :

```
mot = (unsigned int)floorf( my_rand() * (N - 1) ) ;
```

Une fois le mot choisi, il reste à déterminer le bit cible. Les mots du LEON ont une taille de 32 bits. Afin de déterminer le masque, on génère un nombre aléatoire dans l'intervalle $[0; 1]$, puis on le multiplie par 31, ensuite on en prend la partie entière. Cette partie entière donne le nombre de décalages à gauche que la valeur `0x1` doit subir pour placer le bit cible en position. En C, l'opérateur $\ll$ représentant le décalage à gauche, cela s'écrit :

```
masque = 0x1 << (unsigned int)floorf( my_rand() * 31 ) ;
```

Obtention du Temps selon le Modèle Binomial

Nous rappelons que le modèle binomial permet, pour une fluence F de particules p_i, $i \in [1 : F]$, d'obtenir la liste des particules qui doivent conduire à une injection de faute. Pour obtenir cette liste, il faut pour chaque particule décider si elle conduira ou non à une injection. Nous rappelons que la probabilité qu'une particule déclenche un upset est la section efficace σ.

En règle générale, pour obtenir des nombres aléatoires distribués selon une loi de probabilité $\mathcal{P}(x)$ donnée, on peut utiliser la technique de *rejection sampling* [57]. On borne alors la fonction $\mathcal{P}(x)$ en abscisse (`X_MAX`), et on génère un couple (x_1, y_1) de nombre aléatoires uniformément distribués, l'un entre 0 et `X_MAX`, l'autre entre 0 et 1. Si $y_1 < \mathcal{P}(x = x_1)$ alors le point x_1 est accepté comme réalisation de $\mathcal{P}(x)$, sinon il est rejeté. La figure 4.8 illustre cette technique.

Dans le cas qui nous intéresse la loi $\mathcal{P}(x)$ est la loi de probabilité d'une variable de Bernoulli. Puisque x ne peut valoir que 1 (succès) ou 0 (échec), il n'est pas nécessaire de tirer un couple de valeurs aléatoires, une seule est nécessaire. Si le nombre aléatoire x_1 compris entre 0 et 1 est inférieur à la probabilité de succès p, alors l'essai est réussi. En C, pour chaque particule p_i, le test s'exprime par :

```
return( my_rand() < p ) ;
```

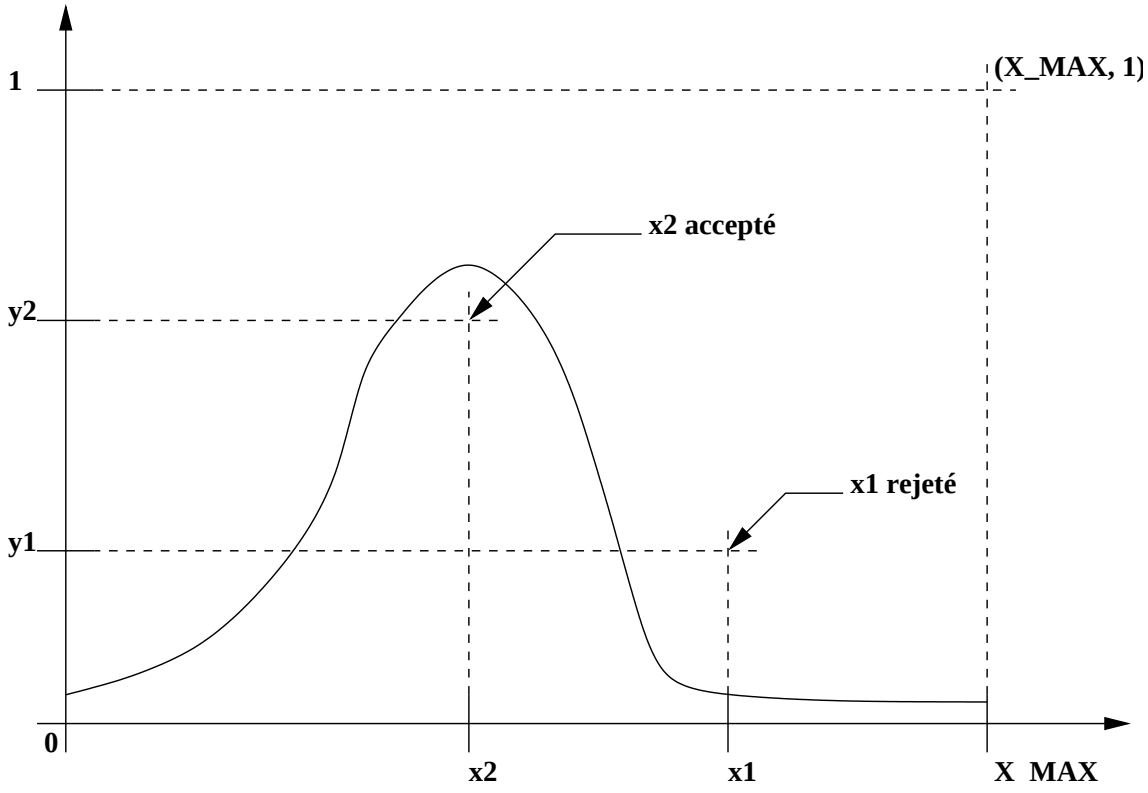

FIG. 4.8 – Illustration de la méthode de *rejection sampling*.

qui retourne 1 si la particule doit être associée à une injection de faute, 0 sinon.

Nous rappelons que le modèle binomial ne donne pas d'instant d'injection de faute, puisque nous ne connaissons pas quand arrivent les particules qui sont sélectionnées pour être associées à une injection. Afin d'utiliser ce modèle, nous créons artificiellement des temps d'arrivée.

Pour une fluence F, nous associons un temps d'expérience T. Le temps d'arrivée t_i de la particule p_i, $i \in [1 : F]$ est calculé selon :

$$t_i = i \times \frac{T}{F}$$

La génération d'instants d'injection de faute se fait de la manière suivante. Il suffit de parcourir la liste de particules, et pour chaque particule, générer un nombre aléatoire x entre 0 et 1. Si $x < \sigma$, une faute doit être injectée, et on calcule le temps d'injection comme précisé ci dessus. En bout de chaîne de traitement du modèle, nous obtenons alors une liste d'instants croissants associés à une injection de faute.

Obtention du Temps selon le Modèle Poissonnien

Lorsque la fonction inverse de la distribution de probabilité cumulée de la loi $\mathcal{P}(x)$ existe et a une forme analytique, il est plus efficace de générer des nombres aléatoires selon la méthode de la transformée inverse (*inverse transform*) qu'en utilisant la technique de *rejection sampling*.

Pour une variable aléatoire X , si

$$X = F^{-1}(U) \tag{4.1}$$

U étant distribué aléatoirement entre 0 et 1, alors X est une variable aléatoire de densité cumulée $F(x)$.

En effet, si X est tel qu'en 4.1 avec $F(x)$ une fonction de densité cumulée quelconque,

$$\begin{aligned} \mathcal{P}(X \leq x) &= \mathcal{P}(F^{-1}(U) \leq x) \\ &= \mathcal{P}(U \leq F(x) \\ &= F(x) \end{aligned}$$

qui est la définition de la fonction de densité cumulée de X.

Pour une variable aléatoire X distribuée selon la loi exponentielle de paramètre λ, $F(X)$ s'écrit

$$F(X) = 1 - e^{-\lambda X}$$

Si l'on pose $U = F(X)$, on obtient

$$X = -\frac{ln(1-U)}{\lambda} \tag{4.2}$$

En pratique, pour une section efficace σ et un flux Φ, $\lambda = \sigma \times \Phi$, on se place à l'instant $t_0 = 0$. On génère X selon l'équation 4.2, et X donne le temps $t_1 = X$ de la première injection. On se place ensuite en t_1 et on génère un nouveau X qui donne $t_2 = t_1 + X$, et ainsi de suite.

Découpage du Temps d'Injection en Cycles d'Horloge et Exécutions

Afin d'obtenir des temps d'injections relatifs à l'exécution en cours, et exprimés en cycles d'horloge, plusieurs étapes sont nécessaires.

Tout d'abord on découpe la liste des fautes à injecter (exprimée en temps absolu T) en temps relatifs à l'exécution t :

$$t = (T)\ modulo\ (\text{durée de l'exécution})$$

Ensuite, on convertit les temps t en cycles d'horloge :

$$n_{cycle} = \frac{t}{f}$$

f étant la fréquence du processeur.

Enfin, dans le cas où plusieurs injections par exécution doivent être réalisées, il faut prendre en compte la durée de la routine CEU et décaler d'autant les prochaines injections. Pour N injections à réaliser dans une même exécution, avec T_{CEU} la durée d'une injection, n_i (l'instant de la i-ème injection pour $i \in [1 : N]$) devient

$$n_i + (i-1) \times T_{CEU}$$

On obtient alors une liste d'instants d'injection de fautes par exécution et mesurée en coups d'horloge processeur. Ce traitement est représenté sur la figure 4.9

4.3.3 Génération de Nombres Aléatoires

Dans la section précédente, nous avons vu que l'obtention des paramètres d'injection de faute fait lourdement appel à la génération de nombres pseudo-aléatoires distribués entre 0 et 1, notamment par l'intermédiaire de la fonction `my_rand()`. Dans cette section, nous évaluons plusieurs générateurs. Une fois que le plus adéquat sera choisi, nous discuterons des améliorations qui lui sont apportées.

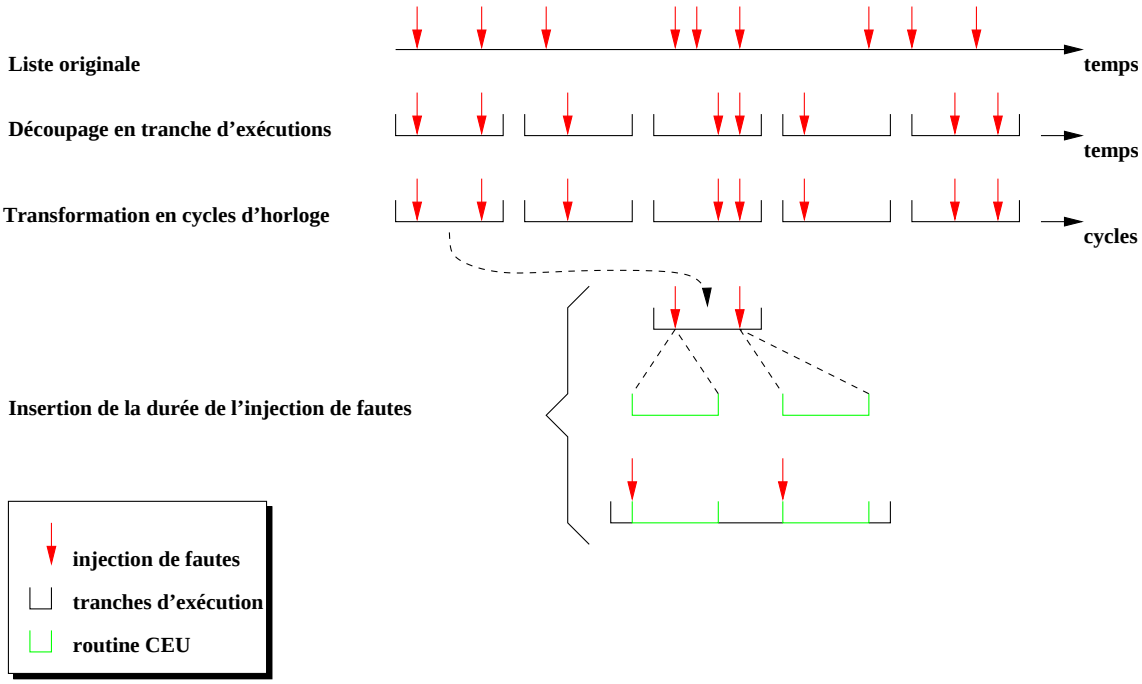

FIG. 4.9 – Découpage des temps d'injections en injections relatives à l'exécution.

Générateur	**Taux d'échec au test du χ^2 au seuil 95%**	**Taux d'échec au test du χ^2 au seuil 99%**	**Durée d'exécution**
`rand()`	5.72%	1.20%	3902 s
Lehmer	5.99%	1.26%	3945 s
Mersenne-Twister	5.87%	1.24%	3897 s

TAB. 4.2 – Test des différents générateurs de nombres aléatoires

Comparaison de Différents Générateurs

Nous avons évalué trois générateurs différents :

1. La fonction de la librairie C accompagnant le système d'exploitation Linux `rand()` [58] qui utilise un *non-linear additive feedback random number generator*. La période est estimée à $16(2^{31} - 1)$.
2. Une implantation d'un générateur de Lemer [59]. La période est $2^{31} - 1$.
3. Mersenne Twister [60]. La période est $2^{19937} - 1$.

Nous avons généré 50000 listes de 60000 nombres avec chaque générateur. Pour chaque liste, le test du χ^2 de Pearson contre la distribution uniforme a été réalisé aux seuils de 95% et 99%. La table 4.2 résume les résultats obtenus. A l'issu de ces

tests, nous avons sélectionné la fonction `rand()`. Tout d'abord parce qu'elle est la plus sûre (taux d'échec le plus bas), ensuite parce qu'elle est relativement rapide.

Cependant deux problèmes persistent :

- La période : la liste de paramètres peut être longue, et beaucoup de tirages sont effectués. Nous voulons éviter de dépasser la période du générateur ;
- Les interférences : nous voulons éviter de tirer les nombres aléatoires des différents paramètres d'injection de faute avec le même générateur afin d'éviter des interférences. En effet nous avons vérifié qu'une liste issue de `rand()` est bien uniformément distribuée, mais rien ne certifie qu'une liste générée à partir de valeurs prises toutes les N générations est conforme à la distribution.

Améliorations

Nous présentons ci-dessous les deux améliorations apportées au générateur afin d'adresser les deux problèmes mentionnés

Extension de Période Les générateurs de nombres pseudo-aléatoires sont déterministes. Si l'on possède une valeur générée aléatoirement, on peut déterminer celle qui va suivre. Cette propriété fait qu'ils sont en général inutilisables tel quel en cryptographie. Cette propriété n'est par contre pas dérangeante pour les applications de type Monte-Carlo.

Une autre propriété est qu'ils sont périodiques. Une fois la période dépassée, ils reproduisent la liste précédemment générée. Une technique permettant de s'affranchir de cette périodicité est la technique de *self-reseeding* [61]. Avant que la période ne soit dépassée, on utilise le générateur pour générer une graine qui servira à réamorcer le générateur. On étend ainsi sensiblement la période. Cette technique est illustrée sur la figure 4.10.

Parallélisation Un générateur de nombres pseudo-aléatoires est défini par sa période, son algorithme mais aussi ses variables d'état. Dans le cas du générateur linéaire congruentiel présenté sur la figure 4.10, l'algorithme s'écrit :

$$X_{n+1} = (a \times X_n + c) \; mod \; m$$

où a s'appelle le multiplicateur, c l'incrément et m le module. Ce jeu de 3 constantes définit, avec l'algorithme, le comportement du générateur. D'où l'idée de partager l'algorithme (le « code ») mais de séparer les variables d'états. Au début de la génération des paramètres d'injection, on crée un nombre suffisant de jeux de variables d'états qui sont associés à une liste particulière (instant, mot, bit). Avant de tirer des valeurs pour une des listes, on « force » le jeu de variables d'état correspondant dans l'algorithme, et on le retire pour le remplacer par le jeu de la prochaine liste.

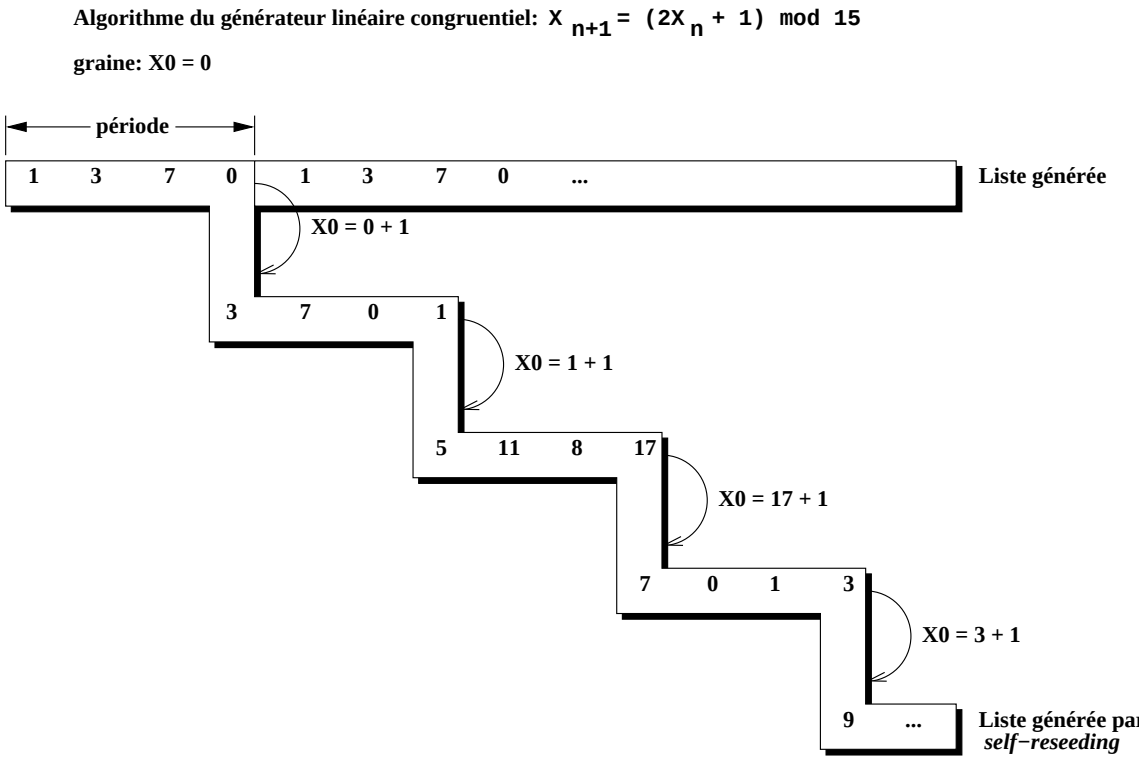

FIG. 4.10 – Illustration de la technique de *self-reseeding* à l'aide d'un générateur linéaire congruentiel. La graine initiale est $X_0 = 0$.

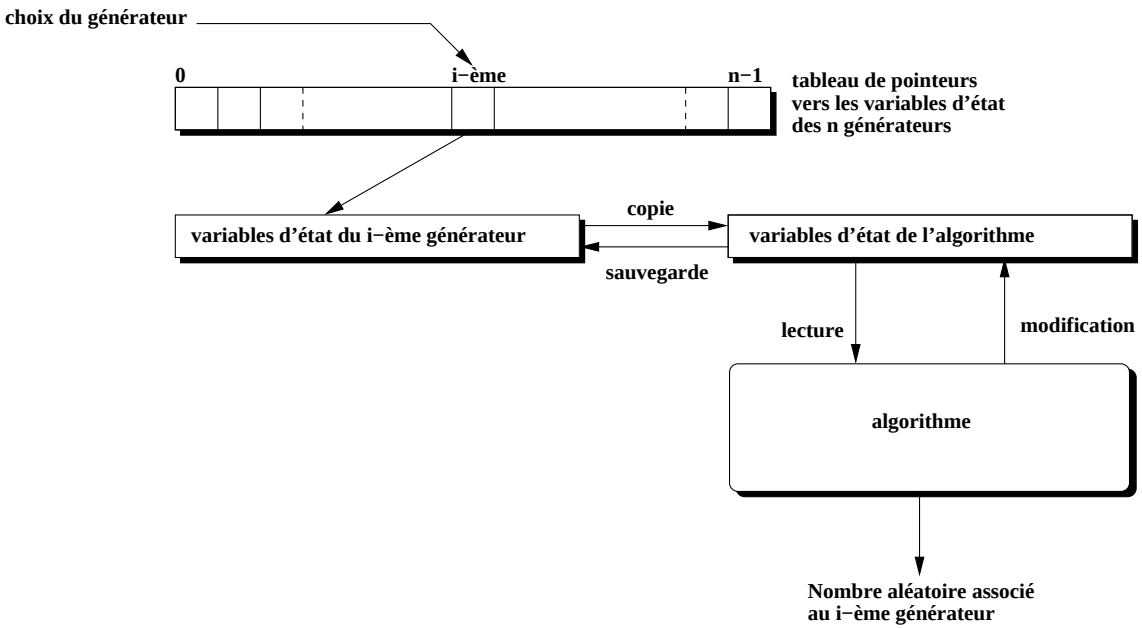

FIG. 4.11 – Schéma d'implantation de la fonction `my_rand()`. La partie *self-reseeding* est masquée.

De cette façon, nous pouvons créer des instances différentes du même algorithme et éviter les problèmes d'interférence : l'algorithme est partagé, les variables d'état ne le sont pas.

En pratique la fonction `rand()` est encapsulée dans la fonction `my_rand()` qui prend comme paramètre le numéro d'identification du générateur cible, copie le jeu de variables d'état associé au générateur, appelle le générateur, sauvegarde le jeu modifié par le générateur, et retourne le nombre aléatoire renvoyé par `rand()`. Cette implantation est schématisée sur la figure 4.11.

Synthèse Un schéma de synthèse de l'implantation du modèle est donné sur la figure 4.12.

4.4 Système de Test

Que ce soit pour conduire les mesures sous radiation ou les injections de fautes, un système de test est nécessaire. Il doit permettre au DUT (*Device Under Test*) d'exécuter son code, et d'écrire les résultats. L'expérimentateur doit être en mesure de récupérer ces résultats. Un contrôle de la consommation du DUT doit aussi être présent.

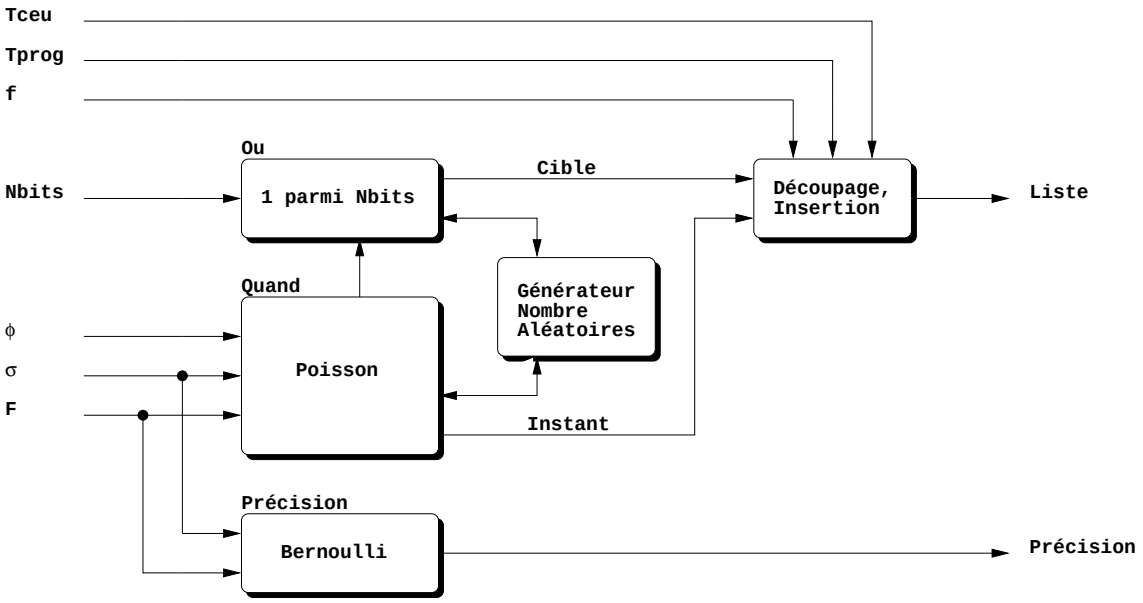

FIG. 4.12 – Synthèse du modèle d'injection de faute. T_{CEU} est la durée de la routine d'injection, T_{prog} est la durée du benchmark, f la fréquence du DUT, N_{bits} le nombre de bit cibles, Φ le flux de l'injection de faute, σ la section efficace des bits considérés et F la fluence de l'injection de fautes.

Nous présentons dans cette section la version « historique » du système de test développé au laboratoire TIMA. Après avoir identifié ses faiblesses, nous introduirons la version élaborée au cours de cette thèse.

4.4.1 Version Préexistante

Ce système, nommé THESIC (*Testbed for Harsh Environment Testing of Integrated Circuits*) a été développé en coopération avec le CNES [47]. Il se décompose en deux cartes : la carte mère et la carte fille.

La carte mère offre les fonctions suscitées. Elle est architecturée autour d'un microcontrôleur de la famille 8051. La communication avec l'ordinateur se fait au moyen d'un lien série RS232.

La liaison mécanique et électrique entre carte mère et fille se fait à l'aide d'un connecteur 96 broches, dont une soixantaine transporte effectivement des signaux électriques. Parmi ceux-ci, 11 lignes d'adresses, 8 lignes de données et quelques signaux de contrôles (lire, écrire, reset, interruption, watchdog).

Les signaux de contrôle permettent au 8051 de démarrer et d'arrêter le DUT, de lancer une interruption et de vérifier quand le DUT a fini l'exécution de son programme.

Les lignes d'adresses et de données permettent de communiquer avec la carte fille au travers d'une mémoire partagée présente sur la carte fille. Cet espace mémoire (de 2 KiB organisé en 2 $Ki \times 8$ $bits$) permet au 8051 de récupérer les résultats issus du DUT.

La carte fille a une seule contrainte : elle doit comporter une mémoire qui sera partagée entre le 8051 de la carte mère et le DUT. En général, lorsqu'elle est bâtie pour le test d'un processeur, elle est composée de plusieurs mémoires contenant le code du processeur et les zones données. Charge à l'expérimentateur de fournir un mécanisme de partage entre l'une d'entre elles et la carte mère.

Une analyse de ce système nous a conduit à identifier les points faibles suivants :

- La taille de mémoire partagée est trop faible, et son organisation en mot de 8 bits rend le partage avec un microprocesseur 32 bits complexe ;
- Le lien de communication entre le PC et le testeur est trop lent. Nous sommes incapables de rapatrier de gros volumes de données, ce qui nous conduit au prochain point ;
- En raison de la lenteur du lien série, l'analyse des résultats est faite sur le testeur, par le 8051. Un résumé synthétique est envoyé au PC comme unique trace de l'exécution, ce qui réduit la visibilité sur le comportement du DUT. Il faut donc maîtriser la programmation du 8051 et connaître parfaitement l'architecture du testeur. De plus, comme le 8051 fonctionne sans système d'exploitation, l'écriture du programme pour le 8051 est délicate ;
- Il n'est pas possible d'implanter le contrôle de flot présenté en début de chapitre ;
- La carte mère est réalisée en *wrapping* ce qui pose des problèmes de fiabilité.

Cette liste non-exaustive des problèmes inhérents à la version disponible au début de cette thèse nous a conduit à réviser en profondeur le testeur.

4.4.2 Amélioration

Cette refonte du testeur s'appelle THESIC$^+$. Quatre révisions ont été faites au cours de la thèse afin de l'adapter à des composants de plus en plus complexes. En raison de la nature « mouvante » de ces révisions, nous présentons uniquement les concepts clefs implantés dans l'architecture THESIC$^+$.

Afin de faciliter la présentation, un schéma de principe est donné sur la figure 4.13.

Le lien de communication série a été remplacé par une liaison Ethernet. Le protocole de transport est implanté à l'aide d'une pile TCP/IP, permettant de brancher le testeur sur un intranet, voire l'internet (certaines des mesures présentées dans ce manuscrit ont été prises alors que l'auteur était installé chez lui, la communication

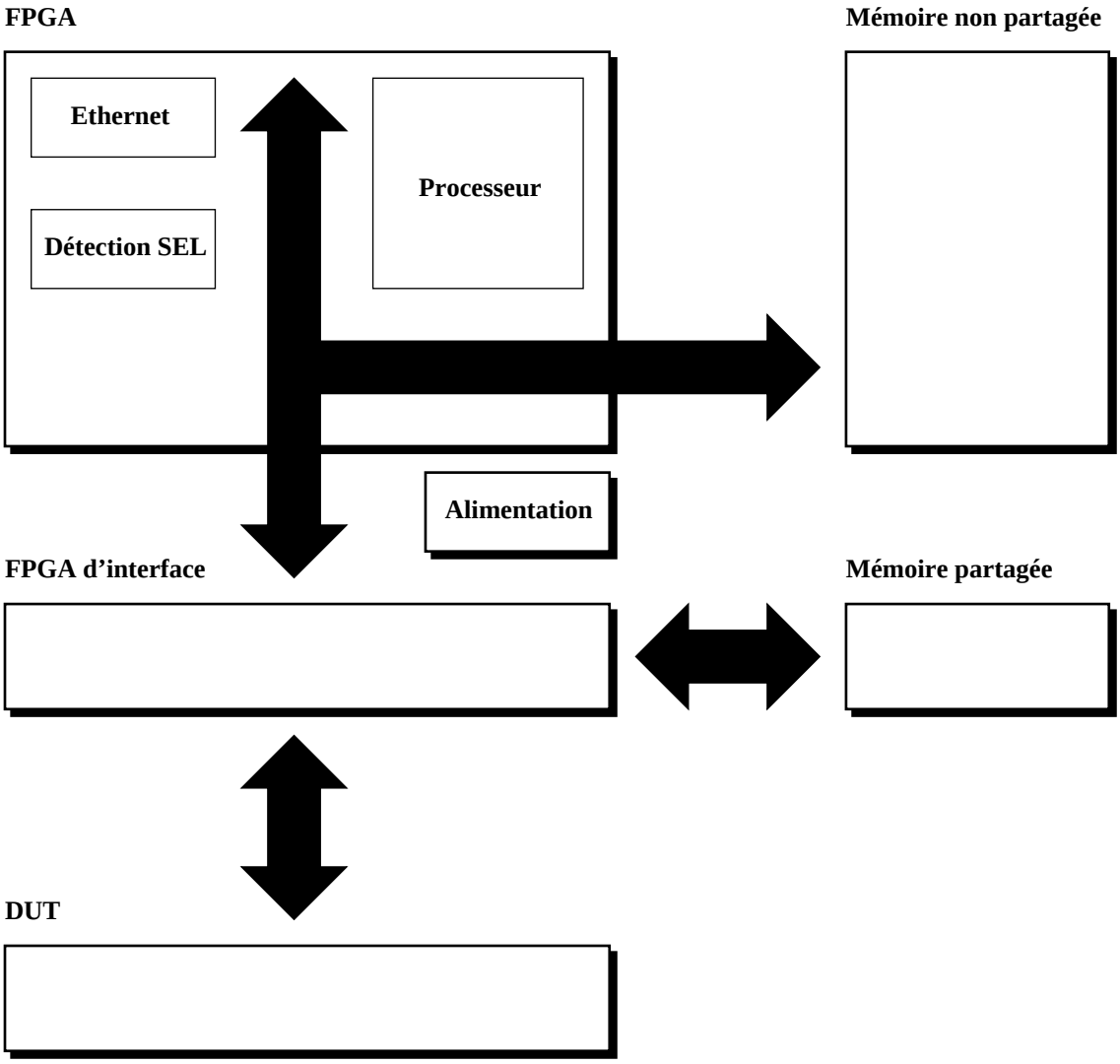

FIG. 4.13 – Architecture du testeur THESIC^{+}.

avec le testeur branché sur l'intranet du laboratoire étant assurée par une liaison ADSL, au moyen d'un tunnel crypté SSH permettant de franchir le pare-feu du laboratoire).

Le microcontrolleur 8051 a été remplacé par le processeur LEON, synthétisé et placé routé dans un FPGA.

Les deux innovations les plus importantes sont les suivantes.

Tout d'abord, le partage d'une mémoire entre le DUT et le testeur est assuré par un FPGA, présent sur la carte mère. Ce composant assure une flexibilité totale dans la création d'une interface DUT / testeur. Nous avons remplacé la conception d'une carte fille complexe par celle d'un design, décrit à l'aide d'un langage HDL, ce qui au vu de notre expérience, est bien plus aisé. Grâce à cet FPGA, les compteurs de contrôle de flot peuvent être implantés facilement.

La deuxième innovation est le déplacement du contrôle de l'expérience du testeur vers l'ordinateur. La liaison Ethernet permettant un transport de données volumineuses, l'expérimentateur peut récupérer les données brutes, et les analyser à la volée, ou plus tard, et profiter de toute la puissance de traitement de l'ordinateur. Il n'est plus nécessaire d'écrire un programme de contrôle pour le testeur. Cette opération a été remplacée par une librairie de fonctions (*Application Programm Interface*, API) de haut niveau, masquant la complexité du testeur. L'utilisateur n'a plus besoin de connaître les détails d'implantation et les protocoles. Ces fonctions sont décomposées par l'API en micro-opérations, envoyées séquentiellement par le lien Ethernet et interprétées par le LEON sur le testeur. Les résultats sont renvoyés vers l'API, qui les recompose et les fournit à l'utilisateur. Cette API peut être appelée à partir de langages de programmation variés (C, Visal Basic, ...). Le débogage est ainsi facilité (l'API a trois ans d'existence, la plupart des bogues ont été découverts et corrigés).

La version actuellement utilisée est fabriquée sur une technologie de circuit imprimé de 8 couches, classe 6.

A ce jour, THESIC$^+$ a été interfacé à un nombre sans cesse croissant de composants différents : microcontrôleurs (8051, ST10), processeurs et DSP (LEON, MIPS, Shark), mémoires (SRAM, FIFO), FPGA (Virtex II) et convertisseurs analogiques numériques.

4.5 Résultats et Discussion

4.5.1 Résultats Radiations

Les mesures de section efficaces des éléments mémoires du LEON ont toutes été prises sur l'installation HIF (*Heavy Ion Facility*), au centre de recherches du cyclotron, à l'université catholique de Louvain-la-Neuve en Belgique. La prise de mesures s'est étalée sur plus d'une année, en utilisant à la fois les cocktails standard

Cible	**Ion**	**LET** $(MeV/cm^2/mg)$	**Section Efficace** $(cm^2/device)$
Registres	$^{22}Ne^{7+}$	3.3	$1.45\ 10^{-5}$
	$^{40}Ar^{12+}$	10.1	$6.35\ 10^{-5}$
	$^{40}Ar^{8+}$	14.1	$2.30\ 10^{-4}$
	$^{58}Ni^{17+}$	21.9	$2.71\ 10^{-4}$
	$^{84}Kr^{17+}$	34	$4.30\ 10^{-4}$
	$^{132}Xe^{26+}$	55.9	$4.40\ 10^{-4}$
Cache données	$^{22}Ne^{7+}$	3.3	$9.67\ 10^{-5}$
	$^{40}Ar^{12+}$	10.1	$3.45\ 10^{-4}$
	$^{40}Ar^{8+}$	14.1	$1.02\ 10^{-3}$
	$^{58}Ni^{17+}$	21.9	$1.39\ 10^{-3}$
	$^{84}Kr^{17+}$	34	$2.11\ 10^{-3}$
	$^{132}Xe^{26+}$	55.9	$2.21\ 10^{-3}$

TAB. 4.3 – Résultats obtenus sous radiation

Ion	**LET**	**Cible**	**Flux** $(p.cm^{-2}.s^{-1})$	**Section efficace** $(cm^2/device)$
$^{40}Ar^{8+}$	14.1	Registres	10000	$2.30\ 10^{-4}$
			5800	$2.30\ 10^{-4}$

TAB. 4.4 – Vérification de l'indépendance vis-à-vis du flux.

et haute pénétration.

Ces résultats sont résumés dans le tableau 4.3. Afin de visualiser les résultats, la courbe de section efficace de la file de registre se trouve sur la figure 4.14, et celle du cache données sur la figure 4.15 (le cache instruction a la même section efficace).

Pour chaque courbe, un ajustement selon la distribution de Weibull a été effectué à l'aide d'une régression linéaire selon la droite des moindres carrés. Dans les deux cas, le coefficient de corrélation empirique r est supérieur à 0.98, ce qui indique que les points sont relativement bien alignés.

Nous avons aussi voulu vérifier la robustesse de notre protocole de prise de mesure et d'analyse. Pour un LET, la section efficace ne dépend que du nombre d'upset enregistrés pour une fluence donnée. Les paramètres de notre protocole (la dimension de la boucle d'exposition, et le flux) ne doivent pas influencer l'obtention de la section efficace. Afin de le vérifier, deux tests ont été effectués.

Le premier consiste pour un LET et une fluence donnés à mesurer la section efficace avec deux flux différents. Avec un flux supérieur nous vérifions que le système de test et le protocole ne « perdent » pas des évènements en route, et qu'ils sont au contraire tous rapportés. Le tableau 4.4 présente les résultats d'un de ces tests. Le protocole est bien insensible au flux de l'expérience.

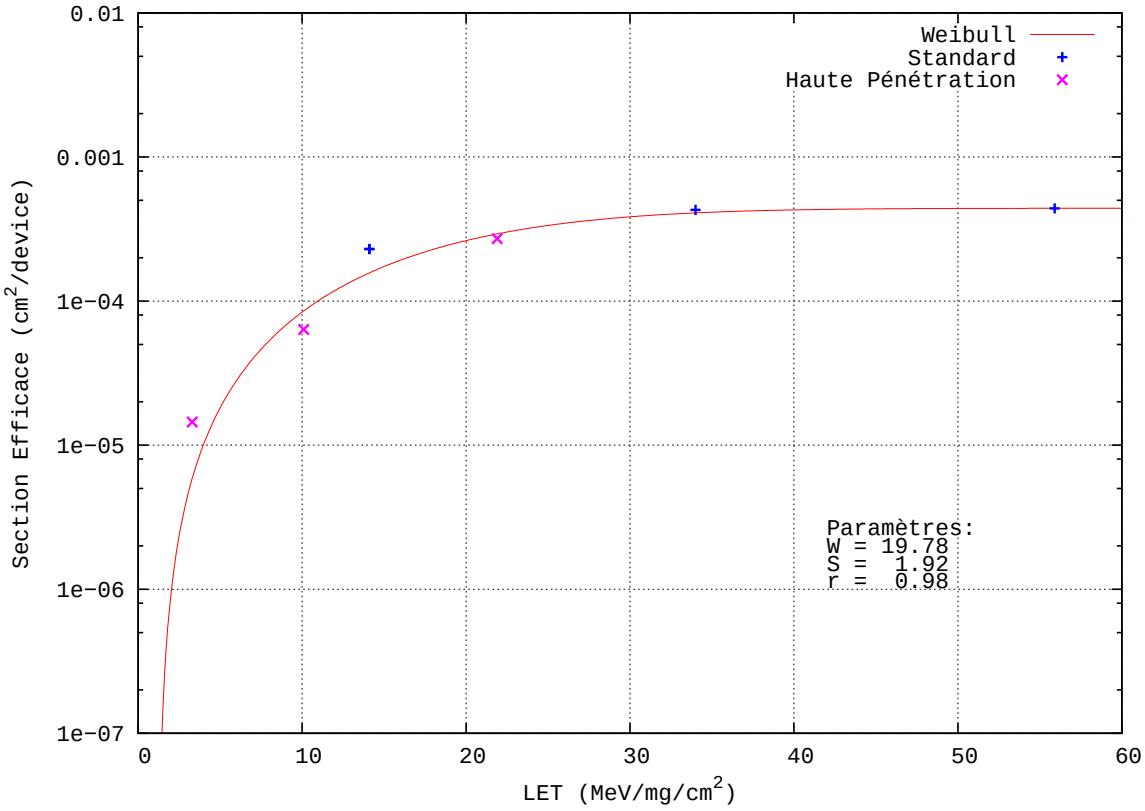

FIG. 4.14 – Courbe de section efficace de la file de registres du LEON.

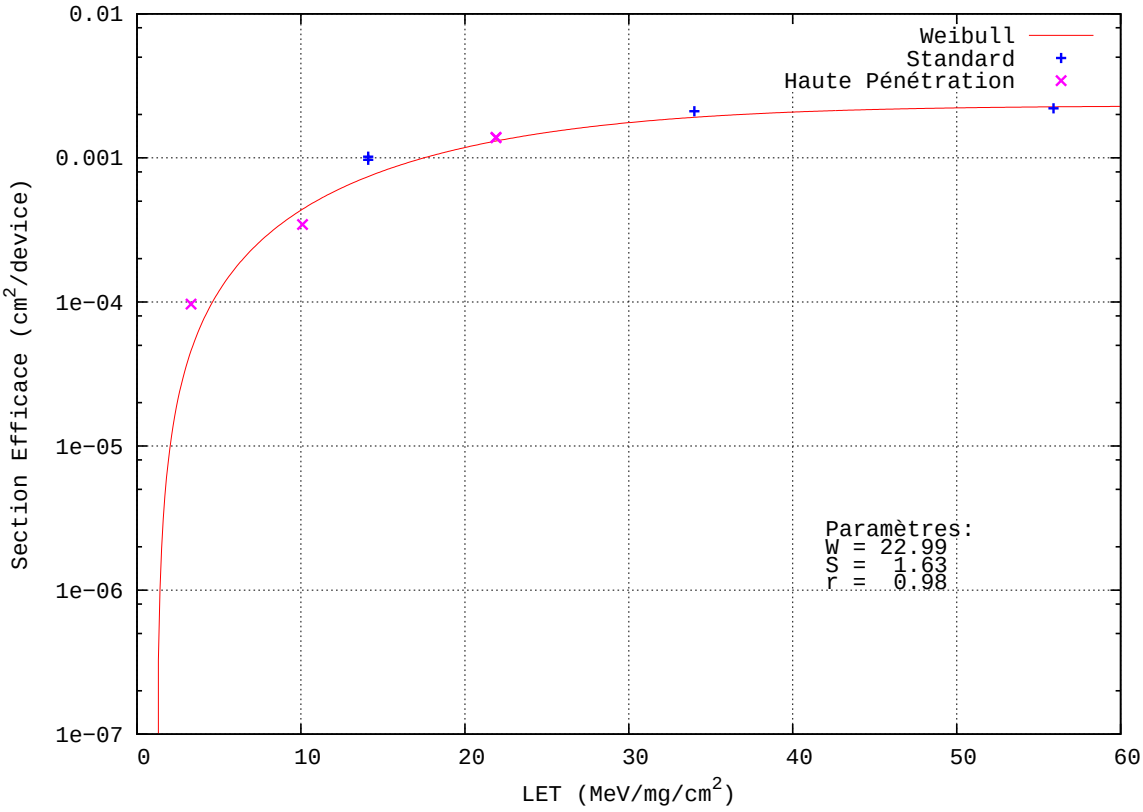

FIG. 4.15 – Courbe de section efficace du cache données du LEON.

Ion	LET	Cible	Boucle d'attente (s)	Section efficace ($cm^2/device$)
$^{58}Ni^{17+}$	21.9	Cache données	1	$1.38\ 10^{-3}$
			2	$1.39\ 10^{-3}$

TAB. 4.5 – Vérification de l'indépendance vis-à-vis de la boucle d'attente.

Le second consiste, pour un LET donné et une fluence identique, à mesurer la section efficace avec un flux identique, mais avec une boucle d'exposition différente. Ici, quand la boucle d'attente est plus grande, la quantité d'événements grandit d'autant. Le transfert des résultats prend donc plus de temps, et augmente la charge sur le lien de communication et les analyses faites à la volée. Nous vérifions que le retrait de ce temps de transfert du temps utile de l'expérience est correct, et que la section efficace reste constante. Le tableau 4.5 montre que le protocole est robuste vis-à-vis de la taille de la boucle d'exposition.

Finalement notons que la disponibilité à HIF d'un faisceau haute pénétration a permis de plus de vérifier que les section efficaces issues de mesures avec le cocktail standard ne subissait pas d'influence vis-à-vis des couches supérieures du circuit (passivation, couches métalliques et oxyde).

En conclusion de cette section, nous pouvons établir que l'ensemble du système de prise de mesure (testeur, protocole de mesure et d'analyse, contrôle du flot du processeur) permet de prendre des mesures de section efficace de bonne qualité, et surtout reproductibles.

4.5.2 Vérification de l'Uniformité

L'homogénéité de la distribution géographique des upsets a été vérifiée pour les mesures effectuées sous radiations comme pour les injections de fautes. En l'absence de la disponibilité de l'adressage physique, la vérification a été effectuée à l'aide de l'adressage logique. La figure 4.16 présente la distribution géographique dans le cache données pour un test sous radiations et pour une injection de fautes. Les paramètres d'injection de fautes ont été calqués sur le test sous radiation (même « flux », même « durée » et la section efficace obtenue sous radiation a servie de probabilité pour l'injection de fautes).

Aucun motif « suspect » ni regroupement d'upset n'apparaît, que ce soit pour le test sous radiation ou pour l'injection de faute. C'est un gage de l'homogénéité de la section efficace et l'injection de fautes la retranscrit correctement.

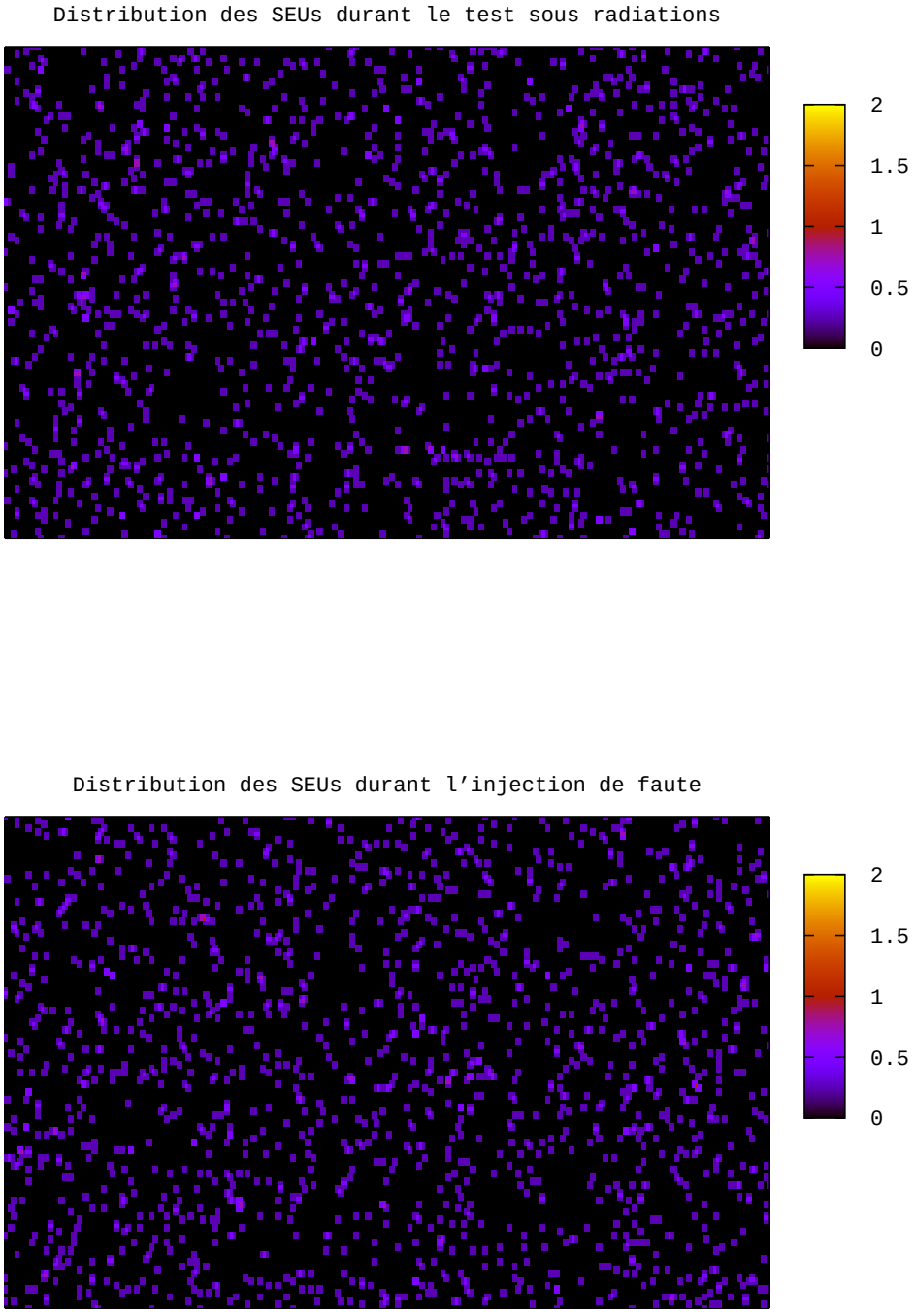

FIG. 4.16 – Distribution géographique des upsets dans le cache données du LEON (Adressage logique, 1500 upsets).

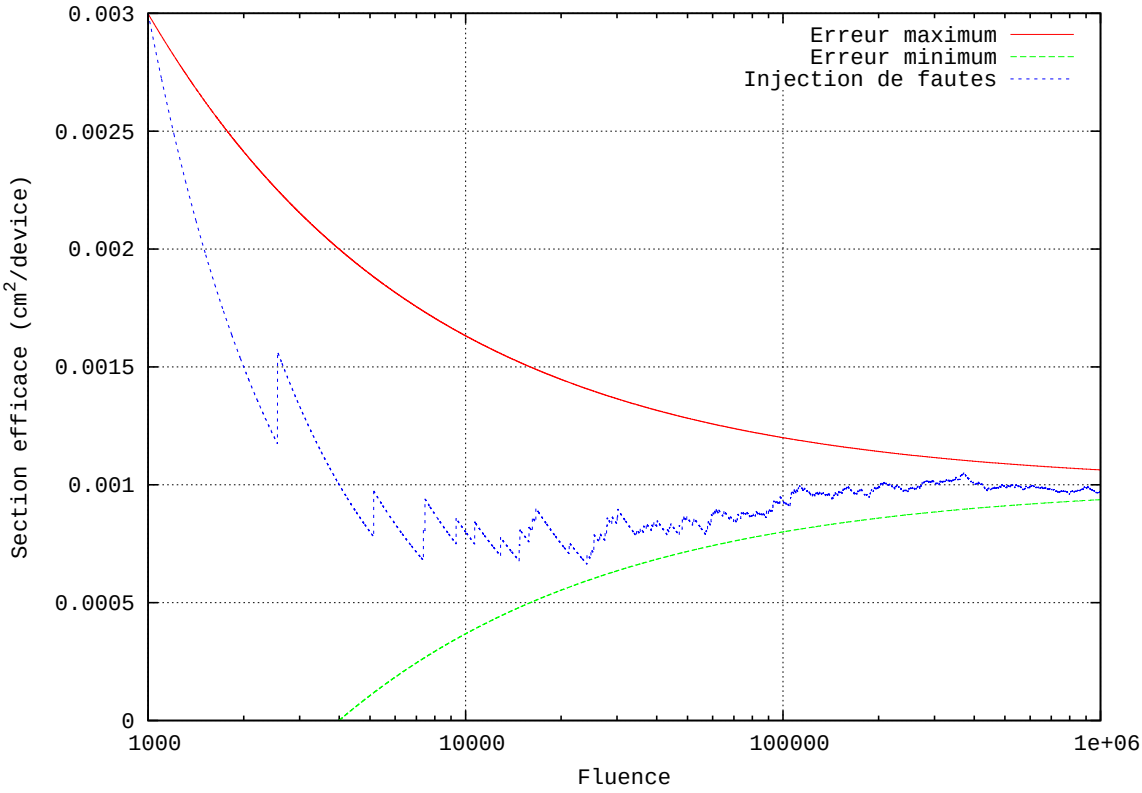

FIG. 4.17 – Démonstration de la convergence de la section efficace obtenue par injection de faute vers celle obtenue sous radiations, et estimation de l'erreur commise. La section efficace simulée est de 1 10^{-3} $cm^2/device$.

4.5.3 Vérification de l'Estimation de l'Erreur

En vertu du modèle d'erreur présenté au chapitre 3, plus la fluence de l'injection de fautes augmente, plus la section efficace simulée doit se rapprocher de la section efficace réelle. Afin de vérifier l'encadrement de la section efficace par les équations d'erreur, nous nous sommes intéressé au comportement asymptotique du modèle. Comme le montre la figure 4.17, plus la fluence augmente, plus la section efficace obtenue par injection de faute converge vers la section efficace sous radiation. Le modèle d'erreur suivant l'approximation normale de la loi binomiale lui aussi converge, de sorte que la précision de la simulation augmente avec la fluence.

Afin de faciliter le calcul de la précision de la simulation, nous avons réalisé un abaque reproduit sur la figure 4.18. Il permet pour une section efficace σ à simuler de retrouver la fluence F à appliquer pour obtenir une précision de 50%, 75%, 90%, 95%, 96%, 97%, 98% et 99%. L'utilisation est simple. Pour un σ choisi, on cherche l'intersection de la verticale passant par σ avec la courbe de précision voulue P.

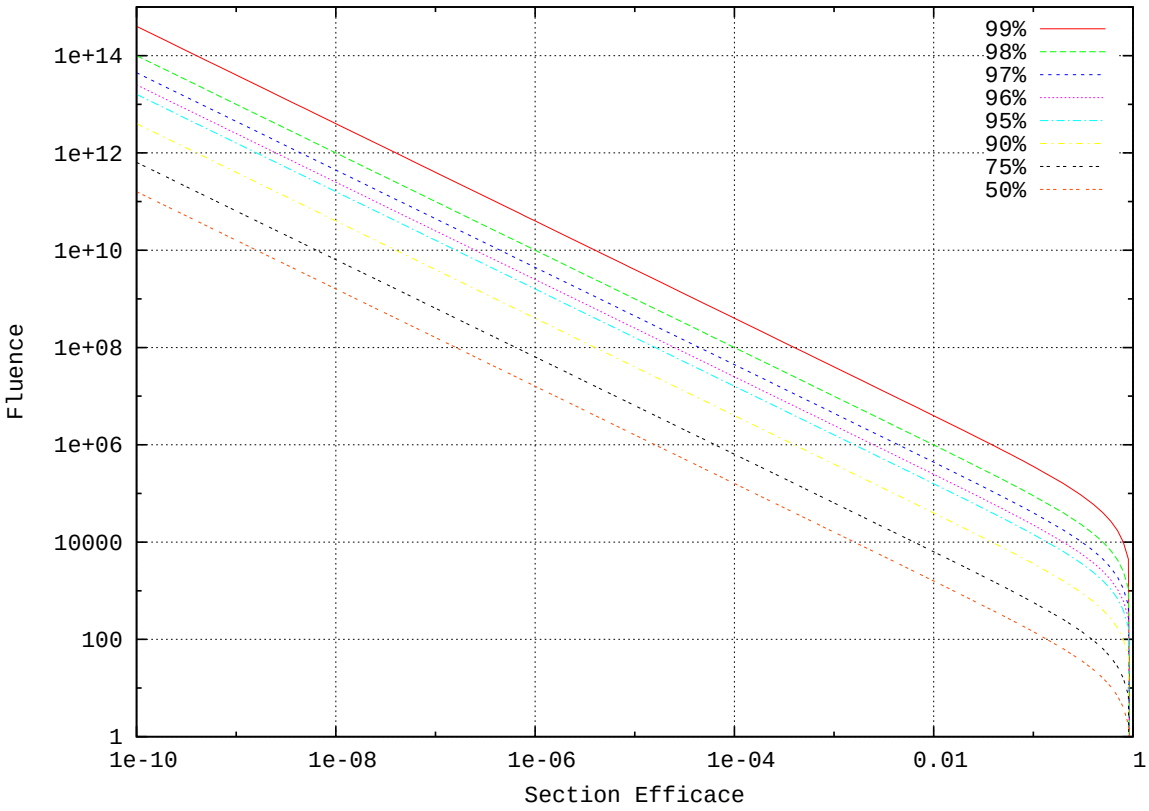

FIG. 4.18 – Abaque de calcul de la fluence à choisir en fonction de la section efficace et de la précision voulue.

L'ordonnée de ce point donne la fluence F a utiliser. La simulation sera fidèle à

$$\sigma \pm (100 - P) \times \sigma$$

près.

Par exemple pour $\sigma = 1\ 10^{-3}$, une précision de 50 % est atteinte dès lors que la fluence F vaut 10000. Pour $F = 1\ 10^{6}$, la précision est meilleure que 90 %.

4.5.4 Résultats d'Injection de Faute et Comparaisons

Nous présentons dans le tableau 4.6 les résultats d'injections de fautes, a mettre en parallèle avec les section efficaces respectives.

Les résultats d'injections de fautes ont été analysés avec les mêmes programmes que ceux obtenus sous radiation. La concordance est très bonne. Du point de vue du DUT, il n'y a pas de différence visible entre le test sous radiation et l'injection de fautes (mis à part un temps d'exécution plus long, dû à l'insertion des routines

Ion	LET	Radiations Cible	Flux ($p.cm^{-2}.s^{-1}$)	Section efficace ($cm^2/device$)	Injection de fautes Section efficace ($cm^2/device$)
Ar	14.1	Registres	10000	$2.30\ 10^{-4}$	$2.50 \pm 0.3\ 10^{-4}$
			5800	$2.30\ 10^{-4}$	$2.46 \pm 0.3\ 10^{-4}$
		Cache données	6000	$1.02\ 10^{-3}$	$1.05 \pm 0.3\ 10^{-3}$
			5500	$0.97\ 10^{-3}$	$1.00 \pm 0.3\ 10^{-3}$
Kr	34	Registres	6000	$4.30\ 10^{-4}$	$4.36 \pm 0.3\ 10^{-4}$
		Cache données	13500	$2.11\ 10^{-3}$	$2.19 \pm 0.3\ 10^{-3}$

TAB. 4.6 – Mesures sous radiations et injections de fautes

d'injection).

Mais la comparaison ne s'arrête pas là. En effet le modèle complet introduit la notion de temps, ce qui nous permet de nous intéresser à la distribution temporelle d'apparition des upsets. Afin d'illustrer cette comparaison, le lecteur peut se reporter à la figure 4.19.

Les histogrammes concordent relativement bien. Une méthode permettant de visualiser cette concordance est la courbe quantile-quantile. Elle est donnée dans la figure 4.20.

Le nuage de points est proche de la droite d'équation $y = x$ ce qui indique que les distributions sont sensiblement équivalentes. Pour quantifier cette équivalence, le test du χ^2 a été effectué. L'hypothèse nulle H_0 est que les histogrammes obtenus sont issus de la même distribution. Au seuil de confiance de 95 %, l'hypothèse nulle ne peut pas être rejetée.

Pour résumer, l'injection de fautes réalisée selon notre modèle permet de reproduire un test sous radiations non seulement vis-à-vis de la section efficace, mais aussi vis-à-vis de la distribution d'apparition des upsets dans le temps. De plus le modèle permet d'estimer la précision de la simulation. Bien évidement, le modèle ne permet pas de « deviner » la section efficace du composant, il permet de reproduire l'environnement qui à permis de l'obtenir.

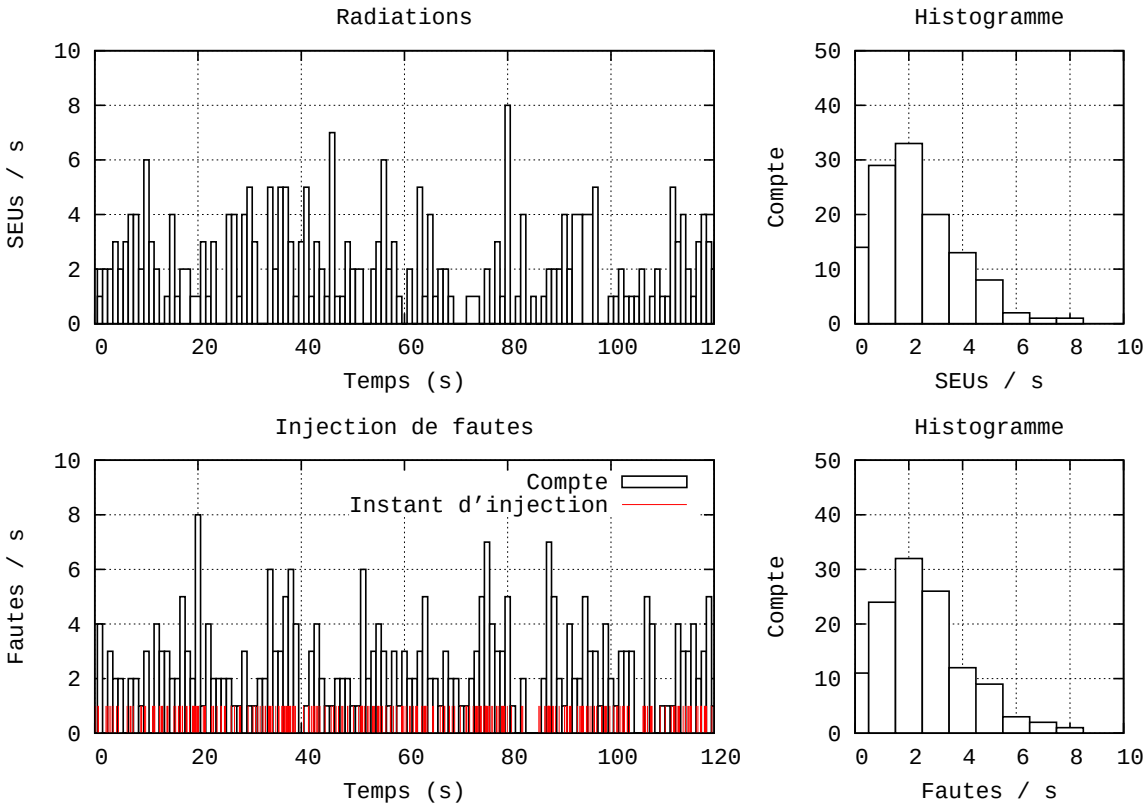

FIG. 4.19 – Comparaison des taux d'upsets par seconde enregistrés sous radiations avec le taux d'injections de fautes par seconde obtenu grâce au modèle. La cible est la file de registres. Lors du test sous radiation, le flux était de 1 10^4 $p/cm^2/s$ et la section efficace mesurée est 2.30 10^{-4} $cm^2/device$. Ces paramètres ont été utilisés pour l'injection de faute. La section efficace obtenue après injection de fautes est 2.50 ± 0.3 $10^{-4} cm^2/device$.

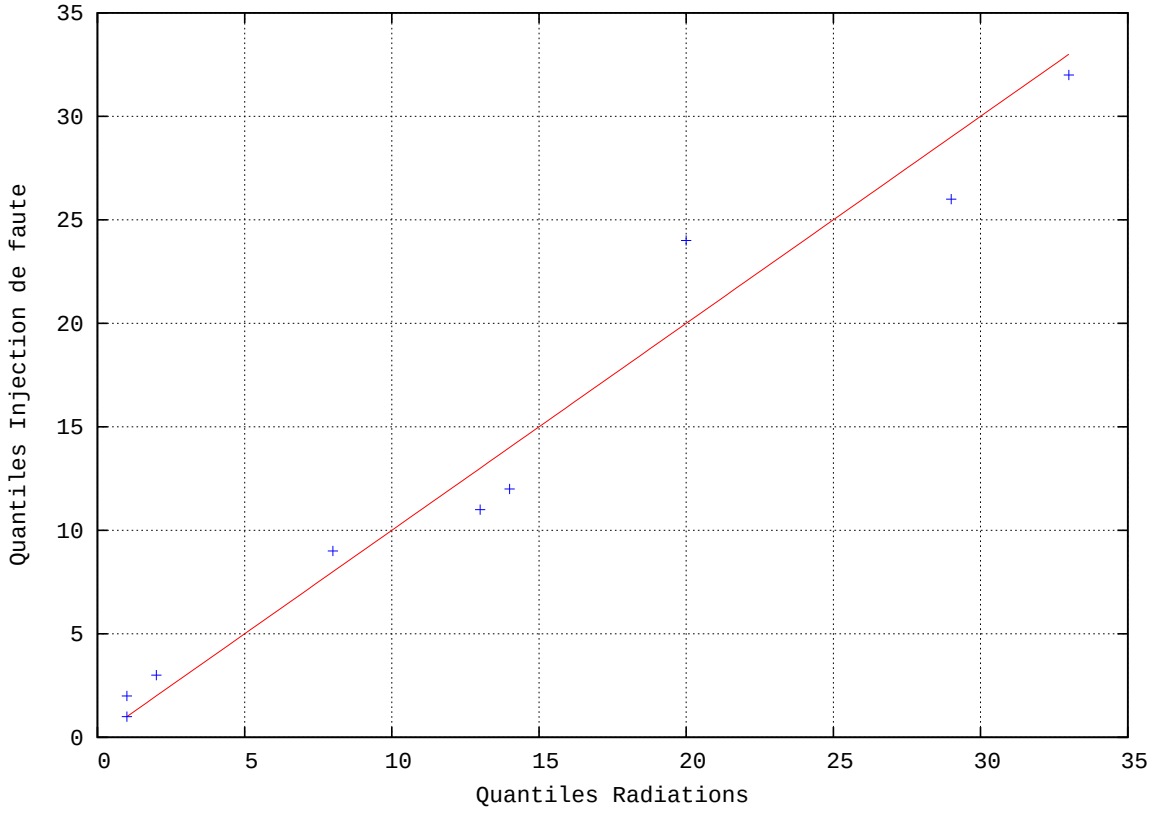

FIG. 4.20 – Courbe quantile-quantile de la distribution temporelle des faute injectées en fonction de l'apparition des SEUs sous radiation.

Chapitre 5

Etude de Programmes Applicatifs en Présence de SEUs

POUR conclure cette étude, nous nous sommes penché sur la qualification du comportement de programmmes « dynamiques » exécutés par notre véhicule d'étude en présence de SEUs induits par un rayonnement ionisant.

Nous commençons ce chapitre par un calcul des taux d'upsets journaliers que subirait le processeur LEON s'il était utilisé dans une application spatiale.

Le chapitre se poursuit par une étude des deux programmes dynamiques choisis, et le dimensionnement effectué pour maximiser l'utilisation du cache données du processeur.

Nous présentons ensuite des résultats pris sous radiation, ainsi que des mesures obtenues par injection de fautes, en utilisant à la fois le modèle présenté au chapitre 3 et l'injection de fautes « classique », telle qu'en [47]. Les données démontrent clairement que la méthode classique ne suffit pas pour expliquer les résultats obtenus, là où le modèle développé dans cette thèse y parvient correctement. Cependant, cette méthode classique offre une couverture de fautes trés interessante comparée à celle obtenue sous radiation.

Nous concluons ce chapitre en donnant une méthode permettant d'utiliser le « meilleur des deux mondes », c'est à dire profiter de la couverture de fautes de la méthode classique, tout en gardant la possibilité de comparer les résultats obtenus selon cette méthode avec ceux obtenus sous radiations.

5.1 Taux de SEU journaliers

En utilisant les paramètres de Weibull obtenus au chapitre précédent, nous avons calculé le nombre de SEU attendus par jour pour le processeur LEON en orbite géostationnaire. Les données sont présentées dans le tableau 5.1.

Condition	Cible	Taux d'upset (SEU/device/jour)
Minimum solaire	Cache données	$5.09550\ 10^{-3}$
	Cache instructions	$5.09550\ 10^{-3}$
	Registres	$8.42110\ 10^{-4}$
	Total	$1.10331\ 10^{-2}$
Flare (*Worst Day*)	Cache données	$1.35549\ 10^{0}$
	Cache instructions	$1.35549\ 10^{0}$
	Registres	$1.94675\ 10^{-1}$
	Total	$2.90565\ 10^{0}$

TAB. 5.1 – Exemple de taux d'upset journalier en orbite.

Le calcul a été effectué pour une orbite gestionnaire, en période de minimum d'activité solaire afin d'augmenter l'exposition du circuit aux rayons cosmiques et lors d'un *solar flare*. Les noyaux de $Z = 1$ à $Z = 92$ ont été inclus dans la simulation. Un blindage d'aluminium de 1 g/cm^2 protège le circuit. Finalement, suivant [26], la profondeur du volume sensible a été fixée à 1 μm sans funneling (pire-cas).

5.2 Présentation des Programmes

Nous présentons dans cette section les deux programmes applicatifs qui ont été choisis : un programme de multiplication de matrices, et un programme de tri. Le choix de ces programmes est orienté vers l'utilisation intensive des ressources du processeur (principalement les mémoire caches) le but étant d'observer un maximum d'erreurs. représentatifs d'une activité réelle [62].

5.2.1 Définitions

Afin de faciliter la présentation des deux programmes, nous clarifions le sens de certains termes :

- *Cache hit* : la donnée requise est en mémoire cache ;
- *Cache miss* : la donnée requise n'est pas en mémoire cache et doit être lue en mémoire externe ;
- *Hit rate* : Le nombre de cache hits divisé par le nombre total de requêtes cache ;
- *Miss rate* : le nombre de cache miss divisé par le nombre total de requêtes cache ;
- Charge (*Workload* : Pour un programme donné, cette valeur qualitative devrait refléter à la fois la taille des vecteurs d'entrée et sortie, ainsi que la complexité du traitement.

5.2.2 Multiplication de Matrices

Ce programme calcule la matrice C comme produit des matrices A et B. A et B sont des matrices carrées de n^2 coefficients.

Le calcul de chaque coefficient de C requiert la lecture de $2n$ coefficients. Si l'on émet l'hypothèse selon laquelle les matrices A et B sont déjà dans le cache données (pas de pénalité de lecture), il y a donc 2n accès cache réussis. Puisque la taille de C est n^2, il y a $2n^3$ cache hits.

Cette formule est un cas idéal, où la taille du cache données est infinie. Dans un cas réel, la gestion du cache doit être prise en compte. Puisque il y a 3 matrices, cette formule reste valide tant que la taille de chaque matrice reste inférieure à 1024/3 (18×18). Si la taille est supérieure, certains coefficients auront la même adresse dans le cache données, et la lecture de l'un deux écrasera l'autre. Si le coefficient écrasé est requis un peu plus tard, un cache miss aura lieu.

Nous avons calculé les nombres d'accès réussis et manqués du cache données pour plusieurs tailles de matrices. Ils sont représentés sur la figure 5.1.

5.2.3 Tri

L'algorithme de tri choisi est connu sous le nom de *bubble sort*. Ce n'est clairement pas l'algorithme le plus rapide, mais son comportement est aisément prévisible.

Pour une liste à trier de n éléments, avec l'hypothèse que la liste est déjà en mémoire cache, et que la taille du cache est infinie, le nombre d'accès réussis au cache données est $(n+1)^n$. En effet,

- n dans $(n+1)$ est le nombre de comparaisons ;
- 1 dans $(n+1)$ vient du premier élément lu dans la liste ;
- ^{n} vient de la répétition de lecture de la liste n fois.

Cette formule est valide du moment que la taille de la liste d'entrée est inférieure à 1024/2. Ensuite, la liste triée recouvre la liste d'entrée dans le cache données et des échecs d'accès au cache sont introduits.

Les nombres d'accès réussis et manqués au cache sont reportés sur la figure 5.2.

5.2.4 Sélection des Tailles

Pour facilement comparer le nombre d'accès réussis au cache pour les deux programmes, une charge « équivalente » doit être utilisée. Plusieurs règles peuvent être utilisées :

- Les vecteurs d'entrées doivent avoir la même taille : $2n^2_{matrice} \simeq n_{liste}$;
- Les vecteurs de sortie doivent avoir la même taille : $n^2_{matrice} \simeq n_{liste}$;
- La somme des tailles des vecteurs d'entrées et de sortie doivent être égales : $3n^2_{matrice} \simeq 2n_{liste}$.

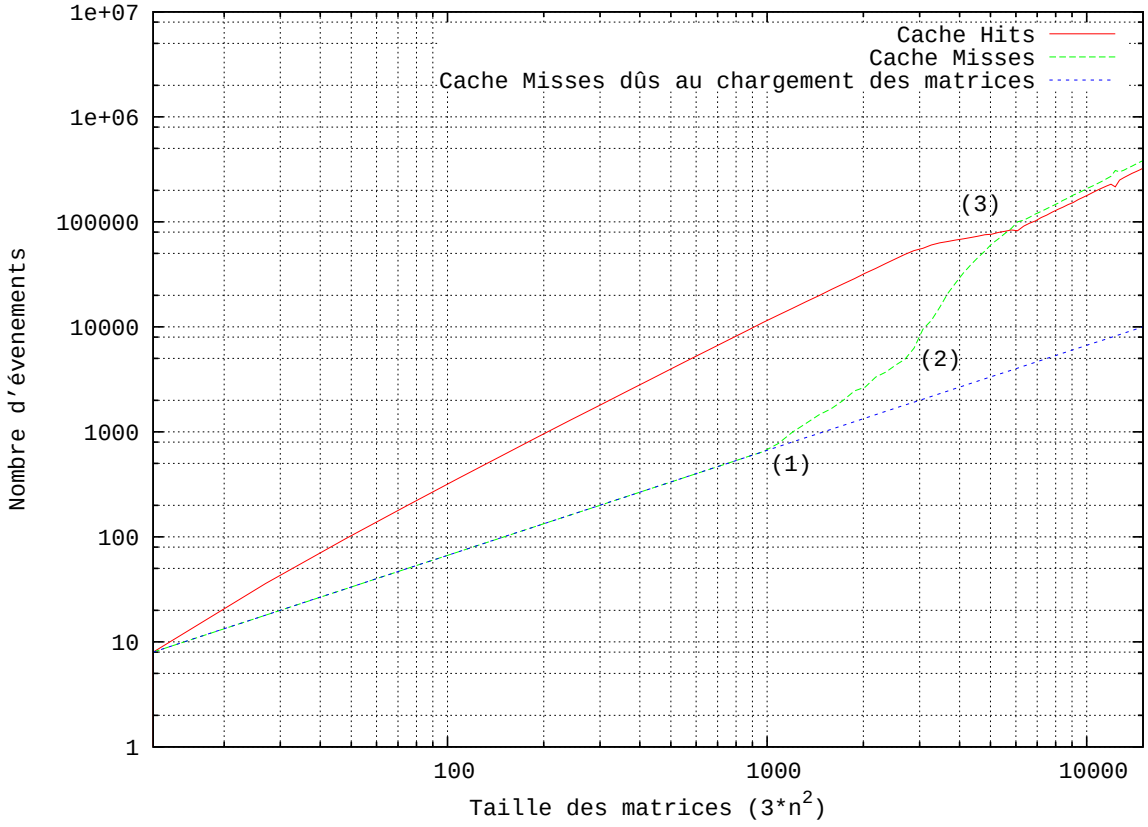

FIG. 5.1 – Accès réussis et manqués au cache données en fonction de la taille des matrices. (1) correspond à une taille de 18×18, où les matrices résident entièrement dans le cache. Entre (1) et (2), les coefficients de C recouvrent ceux de A et B. Après (2), A, B et C se recouvrent mutuellement.

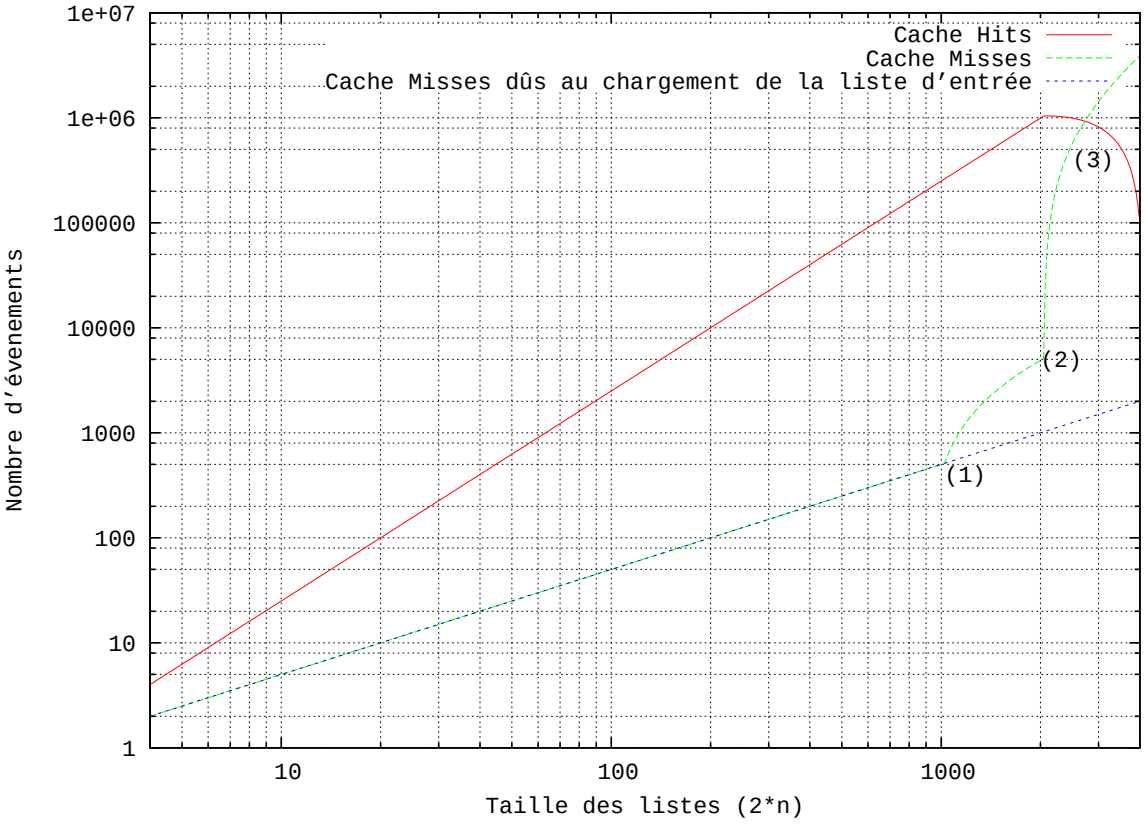

FIG. 5.2 – Accès réussis et manqués au cache données en fonction de la taille des listes. (1) correspond à une taille de 1024/2, où les deux listes résident entièrement dans le cache. Entre (1) et (2), les coefficients de la liste triée recouvrent ceux de la liste d'entrée. Après (2), une même liste se recouvre.

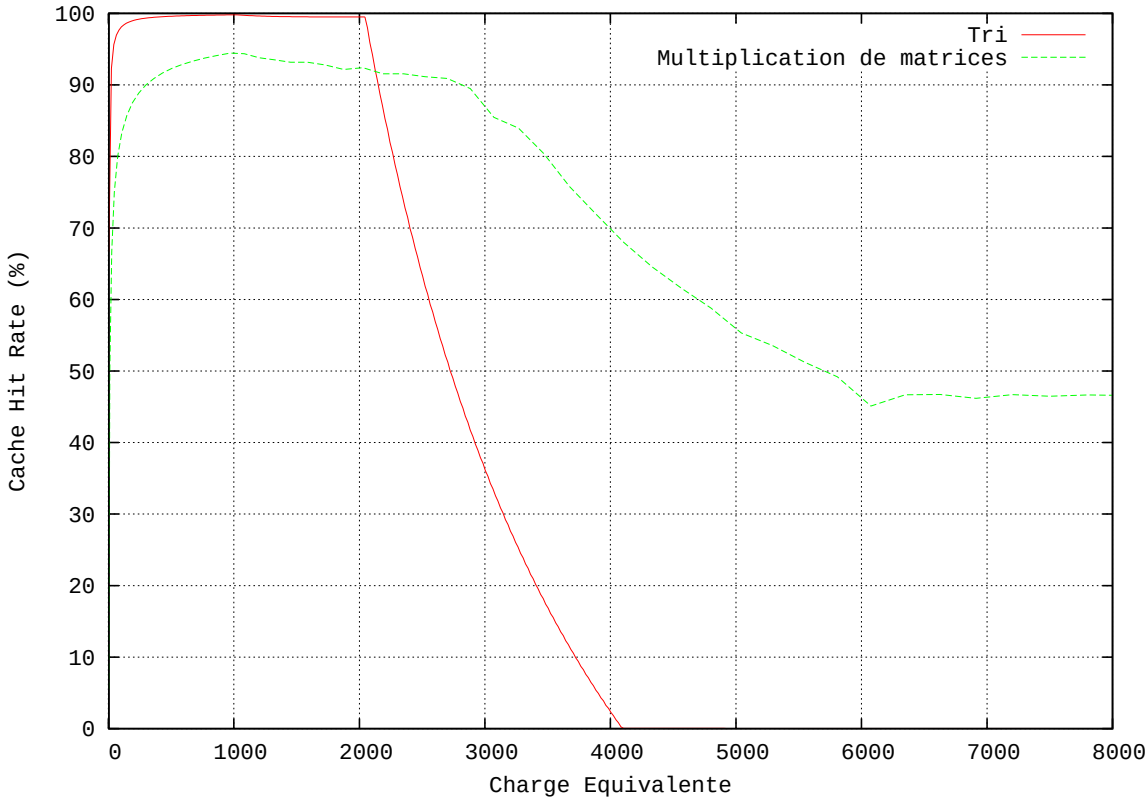

FIG. 5.3 – Taux d'accès réussis pour chaque programme en fonction de la charge équivalente.

Nous avons conservé la dernière règle. En effet elle a l'avantage de refléter ce qui se passe à l'intérieur du cache. Lorsque il y a recouvrement de données à l'intérieur du cache, ce recouvrement est identique avec les deux programmes. Nous pouvons donc comparer les taux d'accès réussis dans les deux programmes. La figure 5.3 représente les taux d'accès réussis pour chaque programme en fonction de la charge équivalente. La situation la plus « stimulante » est clairement celle où chaque matrice a une taille de 18×18, et où la taille des listes est de $1024/2$.

5.3 Mesures sous Radiations et Injection de Fautes

5.3.1 Comportements Observés

A la suite des mesures sous radiations ou de l'injection de fautes, quatre comportements ont été observés :

- Les résultats sont corrects ;
- Un ou plusieurs vecteurs de sortie sont incorrects ;

– Le processeur a fini le calcul (juste ou faux) en avance ou en retard par rapport à la durée attendue ;
– Le processeur n'a jamais atteint la fin du programme.

Afin de faciliter la présentation des résultats, aucune distinction n'est faite dans le type d'erreur. Toute disgression vis-à-vis de l'exécution attendue est considérée comme une erreur.

5.3.2 Résultats

Nous nous sommes intéressés à la capacité de notre modèle d'injection de fautes à reproduire des résultats obtenus sous radiation. A cette fin, les entrées de notre modèle seront les sections efficaces statiques obtenues précédemment, et les paramètres de flux et fluence utilisés lors de la mesure sous radiation effectuée sur les programmes dynamiques. Pour finir, nous comparons ces résultats avec l'injection de fautes « classique » présentée dans [47].

Pour rappel, cette injection classique repose sur la formule suivante :

$$\sigma_{dyn} = \tau_{inj} \times \sigma_{stat} \tag{5.1}$$

avec σ_{dyn} le taux d'erreurs par particule, τ_{inj} le taux d'erreurs par upset (obtenu par injection de faute) et σ_{stat} la section efficace du circuit.

Pour cette méthode, on injecte une faute distribuée aléatoirement dans le temps au cours de l'exécution du programme par le processeur.

Les résultats de mesure sous radiations et d'injection de fautes pour le programme de tri sont donnés dans le tableau 5.2. Le lecteur pourra trouver les données relatives au programme de matrices dans le tableau 5.3.

5.3.3 Discussion

Pour le programme de matrices (tableau 5.3) les résultats sont simples à analyser. Les résultats sous radiation sont relativement regroupés, et nous ne pouvons pas faire de distinction entre les deux méthodes d'injection de fautes.

En revanche, pour le programme de tri (tableau 5.2), les observations sont toutes autres. En effet, à flux égal, les résultats sous radiations sont bien regroupés. Par contre, les résultats varient en fonction du flux appliqué. Pourtant, nous avons utilisé le même circuit, le même programme, et la même quantité de particules. De plus, la répétition des mesures pour un même flux nous assure que ce que nous observons n'est pas dû à des variations statistiques.

L'injection de fautes en utilisant le modèle présenté au chapitre 4 reproduit très correctement ce phénomène. L'injection de fautes traditionnelle en est **incapable**.

Afin de comprendre la cause de ces variations et pourquoi l'injection de fautes « classique » ne les retranscrit pas, nous nous sommes intéressés à la seule différence

Flux (p/s)	**Type**	**Erreurs**	**Taux d'erreur par particules**
$1\ 10^4$	Radiations	448	$4.48\ 10^{-4}$
		475	$4.75\ 10^{-4}$
		450	$4.50\ 10^{-4}$
		464	$4.64\ 10^{-4}$
	Injections (Notre modèle)	462 ± 11	$4.62\ 10^{-4}$
$5\ 10^3$	Radiations	607	$6.07\ 10^{-4}$
		658	$6.58\ 10^{-4}$
		626	$6.26\ 10^{-4}$
	Injections (Notre modèle)	636 ± 14	$6.36\ 10^{-4}$
$2\ 10^3$	Radiations	755	$7.55\ 10^{-4}$
		789	$7.89\ 10^{-4}$
		769	$7.69\ 10^{-4}$
	Injections (Notre modèle)	788 ± 26	$7.88\ 10^{-4}$
$1\ 10^3$	Radiations	866	$8.66\ 10^{-4}$
		857	$8.57\ 10^{-4}$
	Injections (Notre modèle)	847 ± 26	$8.47\ 10^{-4}$
(Pas applicable)	Injections (classique)		$9.14\ 10^{-4}$

TAB. 5.2 – Mesures sous radiations et injections de fautes sur le programme de tri. Pour chaque expérience la fluence est de 10^6. Les valeurs moyennes de l'injection de fautes ont été calculées sur 30 échantillons. Les mesures sous radiations ont été prises avec des ions ayants un LET égal à 55.9 $MeV/cm^2/mg$. L'injection de fautes classique porte la mention « Pas applicable » dans la colone flux car ce paramètre n'est pas pris en compte dans cette méthode.

Flux (p/s)	**Type**	**Erreurs**	**Taux d'erreur par particules**
$1\ 10^4$	Radiations	725	$7.25\ 10^{-4}$
		691	$6.91\ 10^{-4}$
		690	$6.90\ 10^{-4}$
		672	$6.72\ 10^{-4}$
	Injections (Notre modèle)	688 ± 22	$6.88\ 10^{-4}$
$5\ 10^3$	Radiations	700	$7.00\ 10^{-4}$
		678	$6.78\ 10^{-4}$
		747	$7.47\ 10^{-4}$
	Injections (Notre modèle)	692 ± 22	$6.92\ 10^{-4}$
$2\ 10^3$	Radiations	656	$6.56\ 10^{-4}$
		676	$6.76\ 10^{-4}$
		712	$7.12\ 10^{-4}$
	Injections (Notre modèle)	694 ± 24	$6.94\ 10^{-4}$
$1\ 10^3$	Radiations	694	$6.94\ 10^{-4}$
		672	$6.72\ 10^{-4}$
	Injections (Notre modèle)	699 ± 33	$6.99\ 10^{-4}$
(Pas applicable)	Injections (classique)		$6.99\ 10^{-4}$

TAB. 5.3 – Mesures sous radiations et injections de fautes sur le programme de matrices. Pour chaque expérience la fluence est de 10^6. Les valeurs moyennes de l'injection de fautes ont été calculées sur 30 échantillons. Les mesures sous radiations ont été prises avec des ions ayants un LET égal à 55.9 $MeV/cm^2/mg$. L'injection de fautes classique porte la mention « Pas applicable » dans la colone flux car ce paramètre n'est pas pris en compte dans cette méthode.

notable entre les deux méthodes d'injection de fautes : la notion d'instant d'injection de fautes.

5.4 Explications

Alors que l'injection de fautes classique se concentre sur l'obtention de la réponse du système processeur + programme en présence d'un upset, la méthode développée dans ce manuscrit est plus générale : elle essaie de reproduire l'environnement auquel est soumis le système lors du test sous radiation. L'obtention de la réponse du système n'est qu'un produit de cet environnement.

Les deux méthodes utilisent le code CEU pour injecter la faute simulant un upset. Alors que la méthode classique injecte une faute par exécution du programme, distribuée de manière aléatoire au cours de l'exécution, notre modèle module la probabilité d'apparition d'un upset par une distribution de Poisson : lors de certaines exécutions, aucune faute ne sera injectée, alors que pour d'autres 2 ou 3 fautes seront injectées (à la manière de ce qui se passe sous radiations, ou aucun contrôle n'est possible sur l'instant d'apparition d'un upset).

Afin de clarifier notre explication, nous avons analysé les traces temporelles d'injection de fautes selon les deux méthodes. Le lecteur pourra se reporter à la figure 5.4 où sont comparées les quantités de fautes injectées par exécution selon les deux modèles.

La quantité de fautes injectées par exécution du programme est clairement différente dans les deux cas, ce qui explique pourquoi la méthode traditionnelle n'est pas en mesure de reproduire les résultats obtenus sous radiations.

Nous pouvons constater que pour le programme de tri, lorsque le flux baisse les valeurs obtenues sous radiations se rapprochent de celles obtenues par la méthode classique. La figure 5.5 représente le nombre de fautes injectées par exécution lorsque le flux est bas (10^3 particules par seconde).

L'explication est simple : lorsque le flux baisse, la quantité d'upsets apparaissant par exécution décroît aussi, de sorte que la situation « 1 upset par exécution » devient prépondérante. On se rapproche alors des conditions d'injection de fautes de la méthode traditionnelle.

Pour le programme de multiplication de matrices, le flux ne semble pas influencer les résultats. La raison est évidente : son temps d'exécution est très court (5.32 ms, à comparer avec les 171.33 ms du programme de tri), et donc la quantité de fautes injectées reste majoritairement dans le domaine $[0 : 1]$ fautes par injection quel que soit le flux.

Nous proposons une nouvelle formule pour calculer le taux σ_{dyn} (nombre d'erreur

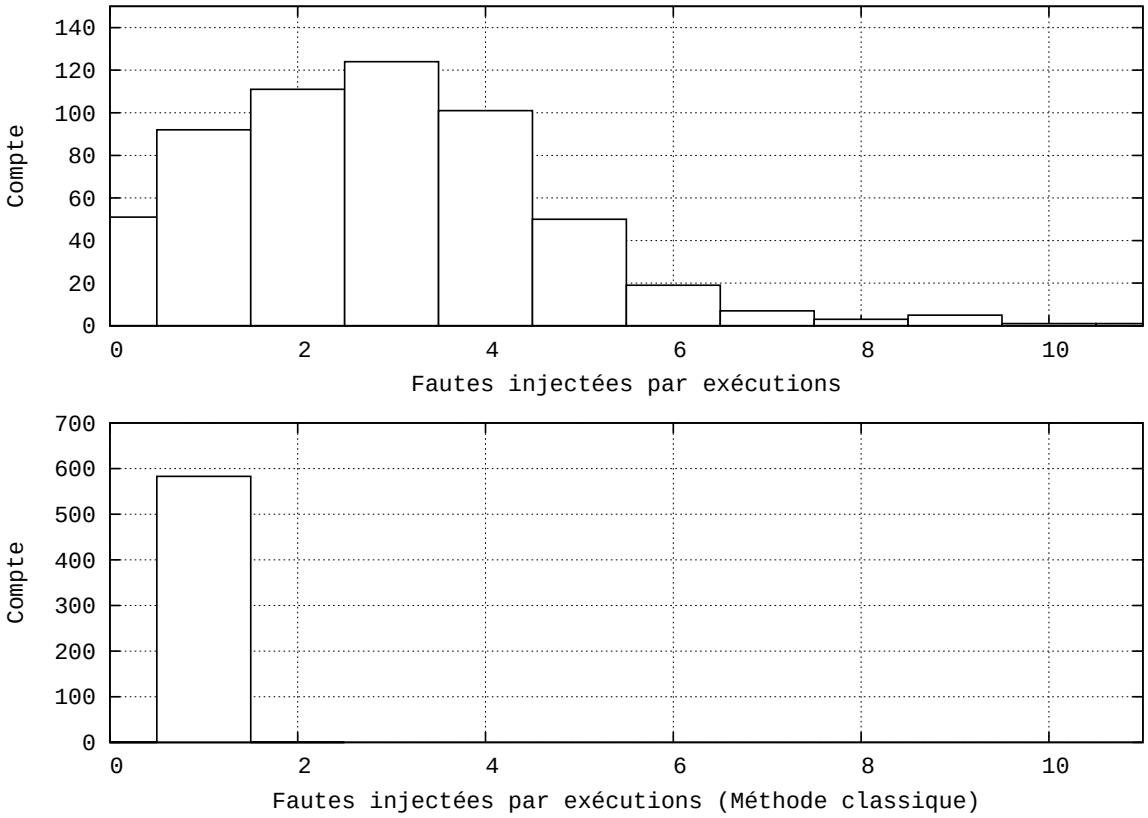

FIG. 5.4 – Programme de tri. Histogrammes représentant le nombre d'injections par exécution selon les deux méthodes d'injection de faute. Pour le modèle présenté dans ce manuscrit, un flux de 10^4 particules par seconde est appliqué.

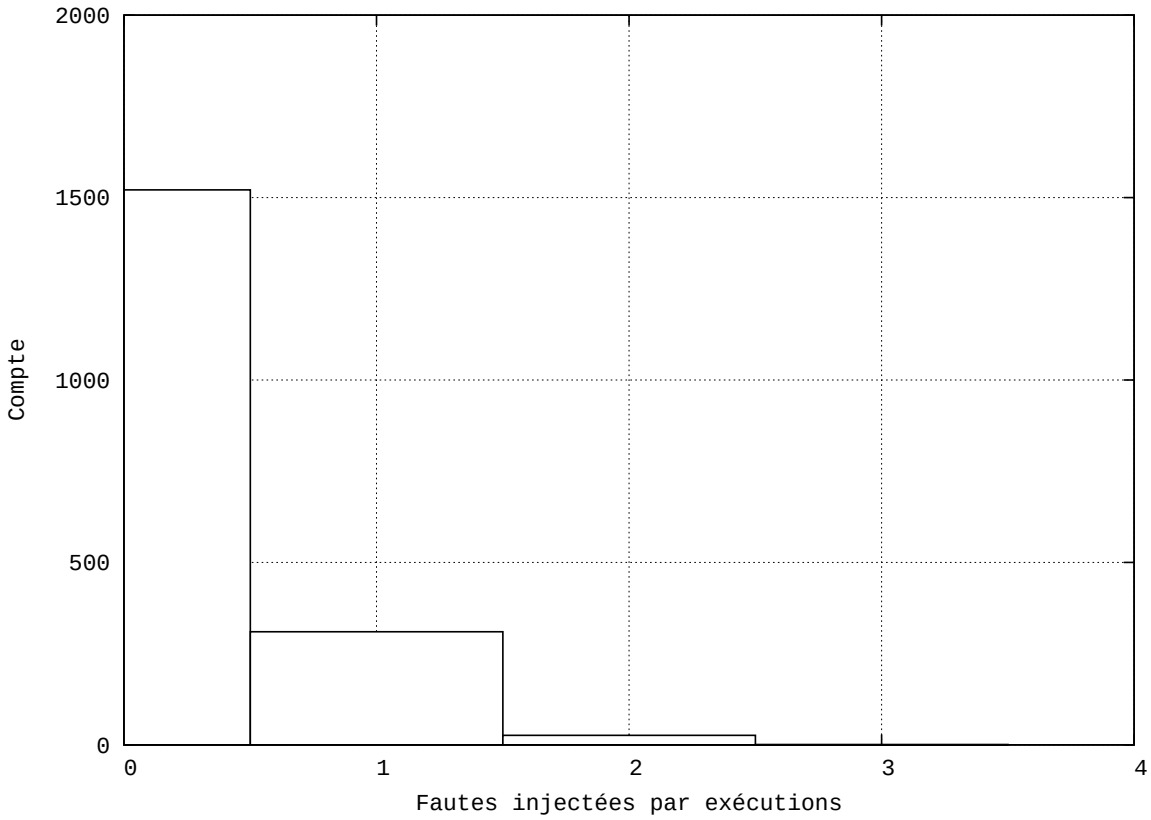

FIG. 5.5 – Programme de tri. Histogramme représentant le nombre d'injections par exécution selon le modèle présenté dans ce manuscrit, pour un flux de 10^3 particules par seconde.

par particules) :

$$\sigma_{dyn} = \frac{N_{exec}}{F} \times \sum_{i=0}^{\infty} \tau_{inj,i} \times \mathcal{P}_{exec}(N_{SEU} = i) \tag{5.2}$$

où $\tau_{inj,i}$ est le taux d'erreur par exécution en présence de i injections par exécution, $\mathcal{P}_{exec}(N_{SEU} = i)$ est la probabilité qu'il y ait i SEU au cours d'une exécution, N_{exec} est le nombre d'exécutions et F la fluence. $\mathcal{P}_{exec}$ dépend de la section efficace statique σ, du flux utilisé Φ et de la durée d'une exécution t_{exec}, et selon notre modèle est donnée par :

$$\mathcal{P}_{exec}(N_{SEU} = i) = \frac{e^{\sigma \times \Phi \times t_{exec}} \times (\sigma \times \Phi \times t_{exec})^i}{i!} \tag{5.3}$$

Nous pouvons vérifier que cette nouvelle formule dépend à la fois de la fluence et du flux de l'expérience, de la section efficace du composant et de la durée de l'exécution du programme. Ces paramètres permettent de comparer de manière précise les résultats d'injection de faute à un environnement quelconque de test sous radiation.

Dans la pratique, il n'est pas nécessaire de calculer la somme jusqu'à l'infini. Les probabilité $\mathcal{P}_{exec}$ décroissent relativement vite de sorte que l'on peut se limiter à quelques termes. Le rang i à partir duquel on néglige les termes suivant reste à la discrétion de l'expérimentateur.

Au terme de cette explication, nous nous posons la question de la représentativité du test sous radiations pour obtenir une estimation de la réponse du système en présence de SEU. En effet, si l'on se réfère au tableau 5.1, dans le pire cas, nous pouvons nous attendre à 2 upsets par jour. La probabilité pour que plusieurs upsets viennent perturber le déroulement d'une exécution est pratiquement nulle. Dès lors, il nous semble inutile de quantifier la réponse du système en présence de plusieurs upsets par exécution. Sous radiations, cela implique de réduire le flux jusqu'à ce que la probabilité $\mathcal{P}_{exec}(N_{SEU} > 1)$ soit négligeable. Dans ces conditions, pour le programme de tri avec une flux de 10^3 particules par seconde, nous notons que pour 1858 exécutions du programme, environ 80 % des exécutions ont été effectuées sans qu'aucun upset n'apparaisse. Seulement 16 % des exécutions ont été perturbées par un upset, et les 4 % restant par 2 ou plus upsets. Pour l'injection de fautes, ces 16 % représentent 310 injections avec 1 injection par exécution. Sachant que le nombre de bits perturbables par les radiations est proche de 70000, et que l'exécution d'un tri dure plus de $3.4\ 10^6$ cycles d'horloge, ces 310 upsets ne représentent que $1.26\ 10^{-7}$ % de l'espace des fautes injectables. Autrement dit, la couverture de fautes lorsque le flux est ajusté de manière à respecter l'environnement final de l'application sera forcément faible (ou le temps « faisceau » très élevé).

Si la sensibilité du composant et l'environnement final de l'application sont tels que la probabilité d'avoir plus d'un upset par exécution est négligeable, l'injection

de fautes à une faute par exécution est bien plus « rentable ». Elle permet une couverture des fautes possibles bien plus grande.

Cependant, le test sous radiations reste « l'étalon ». Il faut donc être en mesure de comparer les résultats d'injection de fautes à cette référence. A cette fin, nous pouvons utiliser la formule 5.2. Les $\mathcal{P}_{exec}$ étant donnés par les paramètres du test sous radiation, reste à déterminer les $\tau_{inj,i}$. Faut il conduire des injections de fautes avec 1, 2 ou plus injections de faute par exécutions ? Nous proposons dans la suite une méthode approximative pour déterminer les $\tau_{inj,i}$, connaissant $\tau_{inj,1}$.

5.5 Calcul des $\tau_{inj,i}$

Moyennant deux approximations les $\tau_{inj,i}$ sont dérivables de la connaissance de $\tau_{inj,1}$.

Si :

- Nous négligeons la probabilité qu'un upset en corrige un autre et
- L'ensemble des erreurs possibles est fonction d'un upset par exécution. (Ceci n'est plus vrai quand le processeur intègre des mécanismes de correction d'upset simple et de détection d'upset double),

alors le taux d'erreur par exécution en présence de n upsets peut être calculé par la formule :

$$\tau_{inj,n} = \tau_{inj,1} \times \sum_{i=0}^{n-1} (1 - \tau_{inj,1})^i \text{ , pour n} > 0 \tag{5.4}$$

L'origine de l'équation 5.4 est donnée sur la figure 5.6.

5.6 Vérification de la Correction Apportée à la Méthode Classique

L'équation 5.4 s'appuie sur la valeur de $\tau_{inj,1}$. Pour pouvoir déterminer avec précision les $\tau_{inj,n}$ il faut connaître précisément $\tau_{inj,1}$ ou du moins savoir quantifier la précision de la mesure. La mesure de la précision revient à la question suivante : Combien faut-il injecter de fautes (à 1 faute par exécution) pour connaître la valeur de $\tau_{inj,1}$ à une précision donnée ?

Ce problème bien connu en statistique est le problème de la détermination d'un intervalle de confiance pour une proportion. Nous donnons ici les résultats du traitement de ce problème.

Si p est le taux d'erreur résultant d'une campage d'injections exhaustive de fautes, et f_n le taux d'erreurs résultant de n injections de faute, alors on peut construire

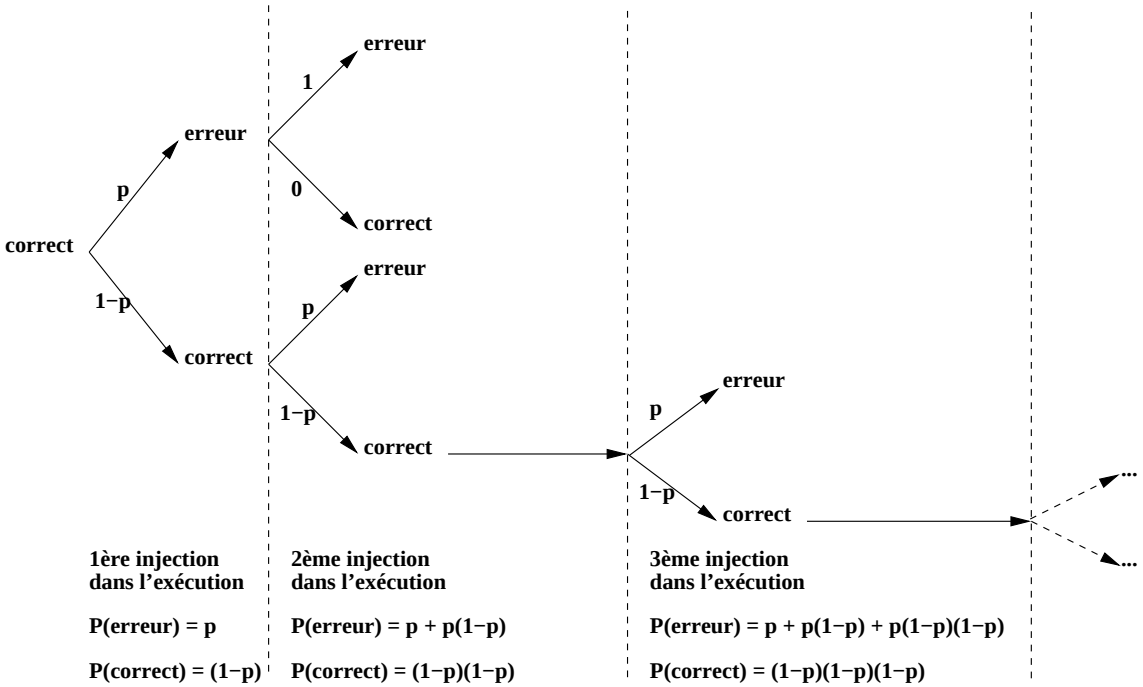

FIG. 5.6 – Soit $\tau_{inj,1} = p$. A la première injection, il y a p chances que le système soit en erreur, et $1 - p$ chance que le système reste correct. A la deuxième injection, si le système était en erreur, il y reste. Il ne peut pas se réparer. S'il était correct, il y a p chances qu'il devienne incorrect, et $1 - p$ chances qu'il reste correct. Etc...

l'intervalle (bilatéral symétrique) de confiance au seuil $1-\alpha$ par

$$f_n \pm u_{1-\alpha/2} \times \sqrt{\frac{p(1-p)}{n}}$$

où $u_{1-\alpha/2}$ est lu dans une table de valeur de la loi normale centrée réduite. Le problème est que l'on ne connaît pas p. On peut :

- Remplacer p par f_n ;
- Remplacer p par $1/2$. En effet le produit de deux nombres de somme constante (1 dans notre cas) est maximal lorsque ils sont égaux. On maximise donc l'intervalle ;
- Utiliser une méthode analytique conduisant à la résolution d'une équation du second degré.

Si l'on décide de suivre la méthode analytique, on pose l'inéquation

$$(f_n - p)^2 \leq u_{1-\alpha/2}^2 \times \frac{p(1-p)}{n}$$

La résolution de l'inéquation donne comme bornes de l'intervalle de confiance (avec $k = u_{1-\alpha/2}$) :

$$\frac{\frac{k^2}{n} + 2f_n \pm \sqrt{\frac{k^2}{n}\left(\frac{k^2}{n} + 4f_n - 4f_n^2\right)}}{2\left(1 + \frac{k^2}{n}\right)}$$

Cette dernière méthode permet de prendre le problème en sens inverse, et de dimensionner le nombre de fautes à injecter afin d'atteindre une précision voulue à un seuil de confiance donné.

En utilisant cette méthode, nous avons déterminé $\tau_{inj,1}$ par la méthode classique sur le programme de tri. Il est égal à 0.4139 ± 0.0001. Nous avons ensuite calculé les $\tau_{inj,i}$ et les avons comparés à des sessions d'injection de fautes à i injections de fautes par exécution. Le résultat se trouve sur la figure 5.7.

La formule 5.4 permet donc d'obtenir avec une bonne précision $\tau_{inj,n}$ pour tout n. Nous calculons ensuite les $\mathcal{P}(N_{SEU} = i)$ dans le cas du programme de tri, avec un flux de 10^4 particules par seconde.

Finalement nous multiplions les $\mathcal{P}_{exec}(N_{SEU} = i)$ et les $\tau_{inj,i}$ pour obtenir τ_{inj}. Le tableau 5.4 donne le résultat du calcul de τ_{inj}.

Multiplié par la section efficace du circuit, nous devons retrouver le taux d'erreur à la fois mesuré sous radiations et obtenu par le modèle d'injection de faute développé au chapitre 4. Nous obtenons un σ_{dyn} de $4.581\ 10^{-4}$ erreurs par particule (à comparer avec les $4.6\ 10^{-4}$ erreurs par particule du test sous radiation). Ce calcul a été répété pour les autres configurations de test et donne d'aussi bon résultats. La méthode est donc valable.

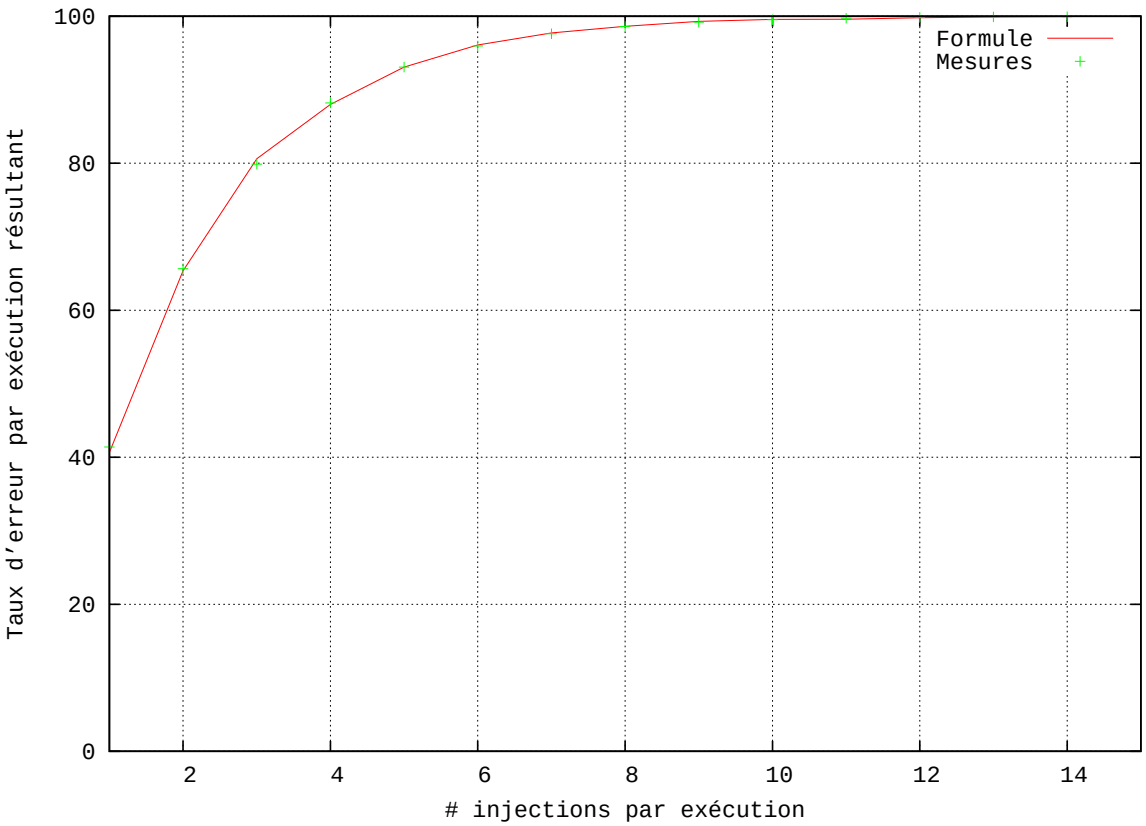

FIG. 5.7 – Résultats d'injection de faute avec i injections par exécution comparés à l'estimation fournie par l'équation 5.4.

i	$\mathcal{P}(N_{SEU} = i)$	$\tau_{inj,i}$
0	0.02	0
1	0.09	0.41
2	0.16	0.66
3	0.21	0.8
4	0.19	0.88
5	0.15	0.93
6	0.09	0.96
7	0.05	0.98
8	0.02	0.99
9	0.01	0.99
	τ_{inj}	0.79

TAB. 5.4 – Table de calcul du τ_{inj} pour le programme de tri, dans le cas ou le flux Φ est de 10^4 particules par secondes à partir de la distribution de Poisson de paramètre $\sigma \times \Phi \times t_{exec}$ et de la connaissance de $\tau_{inj,1}$.

5.7 Conclusions

Nous avons montré dans ce chapitre que l'hypothèse normalement retenue pour l'injection de fautes, c'est à dire l'injection d'une unique faute par exécution du programme étudié, peut ne pas être valalable lorsque l'on veut comparer ces résultats avec ceux obtenus par un test sous radiation. Nous avons aussi montré que le modèle présenté dans cette thèse est par contre utilisable pour effectuer cette comparaison.

Un autre résultat de ce chapitre est que l'expérimentateur doit correctement ajuster les paramètres du faisceau de particules s'il veut stimuler le circuit de manière vraisemblable vis-à-vis de l'environnement dans lequel le circuit évoluera.

Cependant, la couverture de fautes (c'est à dire le nombre de fautes qui ont perturbé le circuit par rapport à l'ensemble des fautes possibles) lorsque cet ajustement est fait reste très faible, là ou la méthode classique d'injection de fautes permet une exploration en profondeur. Une formule est donnée, permettant de dimensionner l'expérience d'injection de fautes classique afin d'obtenir la précision voulue.

Puisque le test sous radiations reste l'étalon de comparaison, nous offrons un traitement analytique permettant de rendre les résultats classiques comparables à ceux obtenus sous radiation.

Chapitre 6

Conclusions et Perspectives

Cette thèse présente une méthodologie complète permettant d'estimer le comportement d'une architecture digitale à base de processeurs en présence d'upsets induits par un rayonnement ionisant. Cette méthodologie se découpe en deux axes : l'obtention de la sensibilité intrinsèque des éléments mémoires du processeur d'une part, et l'utilisation d'une technique d'injection de fautes pour obtenir une estimation du comportement du processeur lorsqu'il exécute une application quelconque.

Un protocole de prise de mesure et d'analyse des données obtenues sous radiations a été présenté. Il permet d'obtenir de manière précise les sections efficaces statiques des éléments mémoires du processeur, en prenant en compte le temps d'exposition réelle du processeur aux radiations tout en filtrant les comportements erratiques du processeur lorsqu'il est irradié. L'invariance des mesures obtenues de cette façon a été évaluée et confirmée. Les mesures de sections efficace obtenues suite à l'utilisation de ce protocole sont de très bonne qualité : consistantes vis-à-vis des paramètres du faisceau de particules, et reproductibles.

La méthode d'injection de faute a été découpé en trois phases, chacune adressant une des trois questions que l'on est amené à se poser lorsque l'on veut injecter une faute : *où*, *quand* et *comment*. Alors que les choix du *où* et du *comment* sont abondamment adressés dans la littérature, nous n'avons *à priori* pas trouvé de références justifiant le choix du *quand*. C'est sur ce choix que la majeure partie du travail théorique a été effectuée.

Afin d'adresser le *quand*, un modèle statistique d'une prise de mesure sous radiations est développé. Une étude approfondie du modèle permet d'établir, moyennant quelques hypothèses simplificatrices, que le nombre d'upsets apparaissant sous radiations par unité de temps est distribué selon la loi de Poisson. La cadence d'apparition des upsets est déterminée par la sensibilité du composant et le flux du faisceau de

particules incidentes.

Pour répondre au *comment*, nous avons amélioré une technique existante d'injection de fautes, la méthode CEU, afin de la mettre à jour vis-à-vis des architectures des processeurs actuels, notamment ceux possédants des mémoires caches. D'un ensemble de routines, chacune adressant un type de mémoire cible, nous en avons fait une routine monolithique capable d'adresser tous les bits mémoires possibles, et ce d'une manière unique. L'implantation en est grandement simplifiée.

Finalement, pour déterminer le *où*, nous utilisons les propriétés des processus de Poisson. Si le processeur a des familles de bits possédant des sensibilités différentes, alors la cadence totale d'apparition des upsets est la somme de la cadence d'apparition dans chaque famille de bits.

L'implantation matérielle et logicielle de ce modèle a été décrite et la confrontation entre le modèle théorique et les données expérimentales a été réalisée à l'aide d'un processeur de type SPARC. Les données expérimentales sont en très bon accord avec les résultats fournis par le modèle d'injection de fautes : l'injection de fautes selon ce modèle est capable de reproduire la distribution spatiale et temporelle des upsets apparaissant sous radiations.

Nous avons finalement utilisé ce modèle pour étudier le comportement du processeur exposé à un faisceau lorsqu'il exécute deux programmes largement utilisés dans ce type d'étude. Les observations montrent que d'une part les prédictions obtenues à l'aide du modèle sont valables, et d'autre part que les hypothèses temporelles classiques d'injection de fautes ne sont pas justifiables. Cependant, nous montrons que les hypothèses classiques permettent une exploration de l'espace total des fautes bien plus poussée que ce que permet d'obtenir un test sous radiations. Nous proposons alors une méthode analytique permettant de comparer les résultats obtenus de manière classique avec ceux obtenus soit sous radiations, soit par l'utilisation du modèle d'injection de fautes présenté dans cette thèse.

Les techniques de mesures et prédictions ont été appliquées soit en partie, soit totalement, dans le cas de deux autres processeurs et donnent d'aussi bons résultats. La disponibilité dans un des deux cas de mesures de sections efficaces statiques sur des « test chips » spécialement conçus pour la mesure sous radiation a démontré le bien fondé du protocole de mesure et de l'abstraction du processeur qui en découle.

Une perspective logique de ces travaux serait la prise en considération des phénomènes émergeant suite à la réductions des tailles des transistors, notamment les upsets multiples (MBUs). Enfin, moyennant une refonte de la façon d'injecter une faute (le *comment*) et la disponibilité d'informations bas niveau sur un coeur de processeur, une tentative visant à inclure les événements de type SET pourrait être

conduite.

Annexe A

Publications et Activités pendant la Thèse

Journaux

[**NSR05**] F. Faure, R. Velazco, P. Peronnard, « Single Event Upset like Fault Injection : a Comprehensive Framework », à paraître dans *IEEE Transactions on Nuclear Science*, Décembre 2005.

[**NSR03**] F. Faure, R. Velazco, M. Violante, M. Rebaudengo, M. S. Reorda, « Impact of data cache memory on the single event upset-induced error rate of microprocessors », dans *IEEE Transaction on Nuclear Science*, Vol. 50, N°6, Decembre 2003, pp. 2101-2106.

Chapitres de Livres

[**SRC04**] R. Velazco, F. Faure, « Single Event Effects Characterization of Complex Digital Circuits », dans *Space Technology Course*, pp. 285-309, ISBN : 2.85428.654.5, Cepadues Editions, 2004.

Tutorials

[**IRPS04**] F. Faure, « Introduction to SER and Testing Challenges », Cours de 45 minutes présenté à *International Reliability Physics Symposium (IRPS'04)*, Phoenix (USA), 25-29 Avril 2004.

Conférences et Workshops

[**IOL05**] R. Velazco, R. Ecoffet, F. Faure « How to Characterize the Problem of SEU in Processors and Representative Errors Observed on Flight », dans *Proceedings of IEEE On-Line Testing Symposium (IOLTS'05)*, 06-08 Juillet 2005.

[**RAD03**] M. Alderighi, F. Casini, S. D'Angelo, F. Faure, M. Mancini, S. Pastore, G. Sechi, R. Velazco, « Proposal for a Radiation Test of Virtex-based ALU », dans *Proceedings of Radiations And its Effects on Components and Systems (RADECS'03)*, Noordwijk (The Netherlands), 15-19 Septembre 2003, pp. 341-346.

[**RAD03**] F. Faure, R. Velazco, « Single Event Upsets on a Read Only Memory Based Complex Programmable Logic Device », dans *Proceedings of Radiations And its Effects on Components and Systems (RADECS'03)*, Noordwijk (The Netherlands), 15-19 Septembre 2003, pp. 279-282.

[**IOL03**] M. Alderighi, F. Casini, S. D'Angelo, F. Faure, M. Mancini, S. Pastore, G. Sechi, R. Velazco, « Radiation Test Methodology for SRAM-based FPGAs by using THESIC+ », dans *Proceedings of IEEE On-Line Testing Symposium (IOLTS'03)*, 7-9 Juillet 2003.

[**RAD02**] F. Faure, P. Peronnard, R. Velazco, « "THESIC+ : A Flexible System For SEE Testing », dans *Proceedings of Radiations And its Effects on Components and Systems (RADECS'02)*, Padova (Italy), 19-20 Septembre 2002, pp. 231-234.

[**NSR02**] F. Faure, R. Velazco, M. Violante, M. Rebaudengo, M. Sonza Reorda, « Analysis of the effects of SEUs on the hidden parts of a pipelined microprocessor : a case study », présenté à *IEEE Nuclear and Space Radiation Effects (NSREC'02)*, Phoenix (USA), 20-24 Juillet 2002.

[**SEE02**] F. Faure, R. Velazco, « A comprehensive method for the evaluation of the sensitivity to SEUs of FPGA-based applications », présenté à *Single Event Effects Symposium (SEES'02)*, Manhattan Beach (USA), Avril 2002.

[**WRT01**] R. Velazco, F. Faure, « A flexible platform for the functional validation of programmable circuits », dans *Proceedings of WRTLT Workshop (RTL ATPG and DFT Workshop) (WRTLT)*, Nara (Japan), 21-23 Novembre 2001.

Rapports Techniques et Projets

En Cours

[**ATMEL**] « AT697 SEE susceptibility evaluation ».
Qualification SEE du processeur AT697E d'ATMEL. Conception du testeur, implantation des méthodes de test et analyse des résultats.

[**SET**] « Living With a Star / SET Project » en collaboration avec NASA et le CNES.
Expérience de mesure du SER d'un FPGA à base de SRAM en orbite. Conception de l'expérience et architecture de la carte de vol.

[**JAXA**] « MIPS core SEE susceptibility evaluation » en collaboration avec l'agence spatiale Japonaise (JAXA).
Qualification SEE implantée par JAXA sur un coeur MIPS TX-49. Architecture de la carte de test, définition des méthodes de qualification et analyse des résultats.

Terminés

[**BOS05**] F. Faure, R. Velazco, « Sub-micronic Circuits Susceptibility to Atmospheric Neutrons : Preliminary Extrapolations for Future Technologies », Robert Bosch Contract Number EPSA0292, Mars 2005.
Tentative d'extrapolation du SER des technologies futures à l'aide de GEANT4 et SPICE.

[**ST04**] F. Faure, « ST10F276 Alpha SER measurement », ST Microelectronics Contract, Novembre 2004.
Mesure du SER induit par particules alpha sur le microcontrôleur ST10F276 de ST Microelectronics.

[**JPL03**] F. Faure, R. Velazco, « THESIC+ Revision 1.1 Documentation », JPL Contract Number 1240660, Septembre 2002.
Séjour de Juin 2003 à Septembre 2003 JPL/NASA, Pasadena, California (USA).
Mise à jour du testeur THESIC+ et premières mesures du SER induit par ions lourds du VirtexII.
Supérieur Hierarchique : Gary M. Swift, Radiation Testing and Failure Analysis group.

[**JPL02**] F. Faure, R. Velazco, « THESIC+ User Manual and MAX7000 Radiations Testing Results », JPL Contract Number 1240660, Septembre 2002.
Séjour de Mai 2002 à Septembre 2002 JPL/NASA, Pasadena, California (USA).
Transfert du testeur THESIC+ à JPL.
Supérieur Hierarchique : Gary M. Swift, Radiation Testing and Failure Analysis group.

[**IRC01**] F. Faure, « ROC-S81 Qualification », IROC Contract, Novembre 2001. Qualification SEEU du processeur ROC-S81 d'IROC Technologies.

Références

[1] J. A. V. Allen, C. E. McIlwain, and G. H. Ludwig, "Radiation observations with satellite 1958 EX," *Journal of Geophysical Sciences*, vol. 64, pp. 271–286, 1959.

[2] T. P. Ma and P. V. Dressendorfer, Eds., *Ionizing Radiation Effects in MOS devices and Circuits*. New York : Wiley, 1989.

[3] D. S. Peck, R. R. Blair, W. L. Brown, and F. M. Smith, "Surface effects of radiation on transistors," *Bell Systems Technical Journal*, vol. 42, pp. 95–129, 1963.

[4] D. Binder, E. C. Smith, and A. B. Holman, "Satellite anomalies from galactic cosmic rays," *IEEE Transactions on Nuclear Sciences*, vol. 22, pp. 2675–2680, December 1975.

[5] T. C. May and M. W. Woods, "Alpha-particle-induced soft errors in dynamic memories," *IEEE Transactions on Electronic Devices*, vol. 26, pp. 2–9, February 1979.

[6] C. S. Guenzer, E. A. Wolicki, and R. G. Allas, "Single event upset of dynamic ram's by neutrons and protons," *IEEE Transactions on Nuclear Sciences*, vol. 26, pp. 5048–5053, December 1979.

[7] J. F. Ziegler and W. A. Lanford, "The effect of cosmic rays on computer memories," *Science*, vol. 206, p. 776, 1979.

[8] J. F. Ziegler *et al.*, "IBM experiments in soft fails in computer electronics (1978-1994)," *IBM Journal of Research and Development*, vol. 40, no. 1, pp. 3–18, January 1996.

[9] D. Lyons, "Sun screens," *Forbes*, November 2000.

[10] SEMATECH, "http ://www.sematech.org/."

[11] JEDEC, "Jesd57 - test procedures for the management of single-event effects in semiconductor devices from heavy ions irradiation."

[12] ——, "Jesd89 - measurement and reporting of alpha particle and terrestrial cosmic ray induced soft errors in semiconductor devices."

[13] ESCC, "Escc 25100 - single event effects test methods and guidelines."

[14] J. H. Elder, J. Osborn, W. A. Kolasinski, and R. Koga, "Method for characterizing a microprocessor's vulnerability to seu," *IEEE Transactions on Nuclear Science*, vol. 35, pp. 1678–1681, December 1988.

[15] J. Thomlinson, L. Adams, and R. Harboe-Sorensen, "The seu and total dose response of the inmos transputer," *IEEE Transactions on Nuclear Science*, vol. 34, pp. 1803–1807, December 1987.

[16] A. S. Eddington, *The Internal Constitution of Stars*. Cambridge : Cambridge University Press, 1926.

[17] P. Lantos, "The sun, the solar wind and their effects on the earth's environment," in *Space Radiation Environment and its Effects on Spacecraft Components and Systems*. Cepadues Editions, 2004, ch. I-01, pp. 13–51.

[18] S. Régnier, "Analyse des structures magnétiques solaires observrès par SOHO. modélisation magnéthohydrodynamique à 3 dimensions," Ph.D. dissertation, Université de Paris-Sud, 2001.

[19] S. Bourdarie and D. Boscher, "Space radiation environment," in *Space Radiation Environment and its Effects on Spacecraft Components and Systems*. Cepadues Editions, 2004, ch. I-02, pp. 57–82.

[20] J. L. Barth, C. S. Dyer, and E. G. Stassinopoulos, "Space, atmospheric and terrestrial radiation environments," *IEEE Transactions on Nuclear Science*, vol. 50, pp. 466–482, June 2003.

[21] J. F. Ziegler and H. Puchner, "Terrestrial cosmic rays," in *SER, History, Trends and Challenges*. Cypress, 2004, ch. IV.

[22] L. Dusseau, F. Saigné, and J. Gasiot, "Basic mechanisms," in *Space Radiation Environment and its Effects on Spacecraft Components and Systems*. Cepadues Editions, 2004, ch. II-02, pp. 145–174.

[23] T. R. Oldham and F. B. McLean, "Total ionizing dose effects in MOS oxydes and devices," *IEEE Transactions on Nuclear Science*, vol. 50, pp. 483–499, June 2003.

[24] J. C. Pickel and J. T. B. Jr, "Cosmic ray induced errors in MOS memory cells," *IEEE Transactions on Nuclear Science*, vol. 25, p. 1166, December 1978.

[25] P. E. Dodd and L. W. Massengill, "Basic mechanisms and modeling of single-event upsets in digital microelectronics," *IEEE Transactions on Nuclear Science*, vol. 50, pp. 583–602, June 2003.

[26] S. Duzellier, "Single Event Effects : analysis and testing," in *Space Radiation Environment and its Effects on Spacecraft Components and Systems*. Cepadues Editions, 2004, ch. II-05, pp. 221–242.

[27] R. Bauman, "Silicon amnesia," in *RADECS 2001 Short Course*, 2001.

[28] L. B. Freeman, "Critical charge calculation for a bipolar SRAM array," *IBM Journal of Research and Development*, vol. 40, no. 1, pp. 119–129, January 1996.

[29] P. Hazucha *et al.*, "Impact of CMOS technology scaling on the atmospheric neutron soft-error rate," *IEEE Transactions on Nuclear Science*, vol. 47, no. 6, pp. 2586–2594, December 2000.

[30] J. F. Ziegler and H. Puchner, "Methods of evaluating the SER of chips," in *SER, History, Trends and Challenges.* Cypress, 2004, ch. II.

[31] W. E. Price *et al.*, "Cosmic rays induced errors in MOS memory cells," *IEEE Transactions on Nuclear Science*, vol. 28, no. 6, p. 1166, December 1981.

[32] R. L. Pease *et al.*, "Radiation testing of semiconductor devices for space electronics," *Proceedings of the IEEE*, vol. 76, no. 11, p. 1510, November 1998.

[33] S. Karoui, "Etude du comportement de circuits complexes en environement radiatif spatial," Ph.D. dissertation, Institut Nationnal Polytechnique de Grenoble, 1993.

[34] S. Karoui *et al.*, "SEU and latchup results for SPARC processors," *IEEE Transactions on Nuclear Science*, vol. 40, December 1993.

[35] R. Velazco, S. Karoui, T. Chapuis, D. Benezech, and L. H. Rosier, "Heavy ions tests for the 68020 microprocessor and the 68882 coprocessor," *IEEE Transactions on Nuclear Science*, vol. 39, no. 3, December 1992.

[36] F. Bezerra, R. Velazco, A. Assoum, and D. Benezech, "SEU and latchup results on transputers," *IEEE Transactions on Nuclear Science*, vol. 43, no. 3, pp. 8973–8978, 1996.

[37] A. Tylka, J. H. A. Jr., P. R. Boberg, B. Brownstein, W. F. Dietrich, E. O. Flueckiger, E. L. Petersen, M. A. Shea, D. F. Smart, and E. C. Smith, "CREME96 : A revision of the cosmic ray effects on micro-electronics code," *IEEE Transactions on Nuclear Science*, vol. 44, pp. 2150–2160, 1997.

[38] K. A. Clark, A. A. Ross, H. H. Loomis, T. R. Weatherford, D. J. Fouts, S. P. Buchner, and D. McMorrow, "Modeling single-event effects in a complex digital device," *IEEE Transactions on Nuclear Science*, vol. 50, pp. 2069–2080, December 2003.

[39] A. Amendola, A. Benso, F. Corno, L. Impagliazzo, P. Prinetto, M. Rebaudengo, and M. S. Reorda, "Faulty behavior observation on a microprocessor system through a vhdl simulation-based fault injection experiment," *IEEE EURO-VHDL96*, September 1996.

[40] P. Civera, L. Macchiarulo, M. Rebaudengo, M. S. Reorda, and M. Violante, "Exploiting circuit emulation for fast hardness evaluation," *IEEE Transactions on Nuclear Science*, vol. 48, no. 6, pp. 2210–2216, December 2001.

[41] R. Velazco, A. Corominas, and P. A. Ferreyra, "Injecting bit-flips by means of a purely software approach : a case studied," in *Proc. of Defect and Fault Tolerance in VLSI Systems*, 2002.

[42] P. Fouillat, V. Pouget, F. Darracq, and D. Lewis, "The laser as a complementary tool," in *Space Radiation Environment and its Effects on Spacecraft Components and Systems.* Cepadues Editions, 2004, ch. II-07, pp. 265–284.

[43] V. Pouget, P. Fouillat, D. Lewis, H. Lapuyade, F. Darracq, and A. Touboul, "Laser cross-section measurement for the evaluation of single event effects in integrated circuits," *Microelectronic Reliability*, vol. 40, p. 1371, 2000.

[44] F. Darracq, H. Lapuyade, N. Buard, F. Mounsi, B. Foucher, P. Fouillat, M.-C. Calvet, and R. Dufayel, "Backside SEU laser testing of commercial off-the-shelf SRAMs," *IEEE Transactions on Nuclear Science*, vol. 49, p. 2977, December 2002.

[45] D. McMorrow, W. T. Lotshaw, J. S. Melinger, S. Buchner, and R. Pease, "Subbandgap laser induced single event effect : Carrier generation via two photon absorption," *IEEE Transactions on Nuclear Science*, vol. 49, p. 3002, December 2002.

[46] R. Velazco, S. Rezgui, and R. Ecoffet, "Predicting error rate for microprocessor-based digital architectures through c.e.u. (code emulating upsets) injection," *IEEE Transactions on Nuclear Science*, vol. 47, pp. 2405–2411, December 2000.

[47] S. Rezgui, "Prédiction du taux d'erreur d'architectures digitales : une méthodes et des résultats expérimentaux," Ph.D. dissertation, Institut Nationnal Polytechnique de Grenoble, 2001.

[48] S. Rezgui, G. M. Swift, R. Velazco, and F. F. Farmanesh, "Validation of an SEU simulation technique for a complex processor : PowerPC7400," *IEEE Transactions on Nuclear Science*, vol. 49, no. 6, pp. 3156–3162, December 2002.

[49] R. Velazco and F. Faure, "Single event effects characterization of complex digital circuits : Test methodology and tools," in *Space Radiation Environment and its Effects on Spacecraft Components and Systems.* Cepadues Editions, 2004, ch. II-08, pp. 285–309.

[50] F. Faure, R. Velazco, and P. Peronnard, "Single event upset like fault injection : A comprehensive framework," *IEEE Transactions on Nuclear Science*, December 2005.

[51] A. de Moivre, *The Doctrine of Chances, or, a Method of Calculating the Probabilities of Events in Play, 3rd ed.* New York : Chelsea, 2000, reprint of 1756 3rd ed.

[52] P. Laplace, *Théorie analytique des probabilités, 3ème éd.* Paris : Courcier, 1820.

[53] J. V. Uspensky, *Introduction to Mathematical Probability.* New York : McGraw-Hill, 1937.

[54] Gaisler research website. [Online]. Available : http ://www.gaisler.com/

[55] The free software foundation website. [Online]. Available : http ://www.gnu.org/copyleft/lesser.html

[56] S. I. Inc., *The SPARC Architecture Manual Version 8*. Menlo Park : SPARC Internationnal Inc., 1992.

[57] D. E. Knuth, *The Art of Computer Programming, Volume 2 : Seminumerical Algorithms*. Boston : Addison-Wesley, 1998.

[58] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C : The Art of Scientific Computing*. New York : Cambridge University Press, 1992.

[59] S. Park and K. Miller, "Random number generators : Good ones are hard to find," *Communications of the ACM*, October 1988.

[60] M. Matsumoto and T. Nishimura, "Mersenne twister : A 623-dimensionally equidistributed uniform pseudorandom number generator," *ACM Transactions on Modeling and Computer Simulation*, vol. 8, no. 1, pp. 3–30, January 1998.

[61] J. Viega, "Practical random number generation in software," in *19th Annual Computer Security Applications Conference*, 2003.

[62] F. Faure, R. Velazco, M. Violante, M. Rebaudengo, and M. S. Reorda, "Impact of data cache memory on the single event upset-induced error rate of microprocessors," *IEEE Transactions on Nuclear Science*, vol. 50, no. 6, pp. 2101–2106, December 2003.

RESUME

Obtenir une estimation du taux d'erreurs induit par les phénomènes de basculement de bit (soft error rate, SER) des équipements électroniques est d'un intérêt grandissant.
Les standards publiés traitent principalement de la qualification des circuits de type mémoire. Il n'y a pas d'accord sur les méthodes de qualification des microprocesseurs.
Dans ce contexte, cette thèse s'attache à définir une méthodologie permettant de prédire le SER d'un processeur à l'aide d'une approche en trois étapes:

- En définissant une méthode de test sous radiation permettant d'obtenir de façon précise la sensibilité du circuit au rayonnement ionisant;
- En présentant une analyse détaillée des mesures, dont le but est d'extraire un modèle statistique d'un test accéléré;
- En utilisant cette empreinte statistique pour reproduire à l'aide d'injection de fautes le comportement du circuit étudié afin de prédire le comportement d'une application quelconque exécutée par le processeur.

MOTS-CLES

Basculement de bits, Injection de fautes, Microprocesseurs.

TITLE

RADIATION INDUCED SINGLE EVENT UPSET LIKE FAULT INJECTION

ABSTRACT

Estimating the soft error rate (SER) of digital equipment is a major concern: while the SER of a single bit can be extremely low, the increasing amount of bits per device combined with their use in safety-critical applications makes the SER evaluation an important milestone before introducing a new technology to the market or using it in a space application.
To derive the SER of a device, the commonly adopted strategy consists in exposing the tested part to either a particle beam (accelerated test) or to its natural environment while it carries on a given activity.
The difficult point is to exercise the tested chip in a way as representative as possible of the one that will be used in the final environment. Standards have been published, defining requirements and procedures for SER testing of integrated circuits. However, the procedures presented in these texts apply primarily to memory devices. There is not such an agreement on the SER evaluation of microprocessors.

In this context, the work done in this Ph.D. defines a methodology to measure and predict the SER of a processor using a three steps approach:

- By defining a correct static test strategy allowing to measure the cross-section of the processor's memory elements;
- By presenting a detailed analysis of radiation ground testing data, aiming at extracting a statistical model of an accelerated radiation ground test, that is where and when SEUs do occur in the studied processor;
- By using this statistical footprint and fault injection techniques to study the behaviour of any application executed by the processor.

INTITULE ET ADRESSE DU LABORATOIRE

Laboratoire TIMA, 46 avenue Félix Viallet, 38031 Grenoble Cedex, France.

ISBN : 2-84813-052-0 (version brochée)

ISBNE : 2-84813-043-9 (version électronique)