Prédiction d'erreurs pour les architectures digitales : méthode et résultats

Sana Rezgui

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

N° attribué par la bibliothèque

THESE

Pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité : **«**Micro-électronique **»**

préparée au laboratoire : **TIMA**

Techniques de l'Informatique et de la Micro-électronique pour l'Architecture des ordinateurs

dans le cadre de **l'Ecole Doctorale «Electronique, Electrotechnique, Automatique, Télécommunications, Signal»**

présentée et soutenue publiquement

par

Sana REZGUI

le 8 Mars 2001

PREDICTION DU TAUX D'ERREURS D'ARCHITECTURES DIGITALES :

UNE METHODE ET DES RESULTATS EXPERIMENTAUX

Directeur de thèse *: M. Raoul Velazco*

JURY

M. Bernard COURTOIS	, Président
M. Pascal FOUILLAT	, Rapporteur
M. Mattéo SONZA REORDA	, Rapporteur
M. Raoul VELAZCO	, Directeur de thèse
M. Jean GASIOT	, Examinateur
M. Robert ECOFFET	, Examinateur

A mes parents, ma famille, mes amis,
Et tous ceux qui m'ont aidé et aimé.
A mon très cher pays…

Remerciements

Je voudrais en premier lieu remercier profondément et tout particulièrement Monsieur Raoul VELAZCO, chargé de recherches au CNRS et responsable du groupe Qualification de circuits intégrés, au laboratoire TIMA (Techniques de l'Informatique et de la Micro-électronique pour l'Architecture des ordinateurs) à Grenoble, pour ses conseils, ses critiques et ses encouragements ; sans sa compétence et sa disponibilité, cette thèse n'aurait sûrement pas eu la même qualité. Qu'il trouve toute l'expression de ma sincère reconnaissance.

Je tiens à exprimer ma profonde gratitude pour les membres du jury :

- *Monsieur Bernard COURTOIS, Directeur de recherches au CNRS et directeur du laboratoire TIMA, dans lequel j'ai effectué mes travaux de recherches, de l'honneur qu'il m'a fait en acceptant de présider le jury de ma thèse.*

- *Monsieur Pascal FOUILLAT, Professeur à l'Université de Bordeaux et chercheur au laboratoire IXL de cette université et Monsieur Matteo SONZA-REORDA, Professeur au Polytechnique de Torino, pour l'intérêt qu'ils ont bien voulu accorder à mes travaux en acceptant d'être rapporteurs de cette thèse.*

- *Monsieur Jean GASIOT, Professeur à l'Université de Montpellier II et chercheur au CEM2 (Centre d'Electronique et de Micro-électronique de Montpellier) d'avoir bien voulu faire partie du jury.*

- *Monsieur Robert ECOFFET, Responsable au CNES (Centre National d'Etudes Spatiales) du Département Environnement Spatial et Effets des Radiations, d'avoir accepté d'être membre de jury, d'avoir assuré le bon déroulement des campagnes de radiation auxquelles j'ai participé et surtout pour sa sympathie.*

J'aimerais dire merci aussi à Madame Françoise BEZERRA, Ingénieur au CNES et Madame Sophie DUZELLIER, Ingénieur au ONERA/DESP, pour leurs conseils et leurs aides durant les campagnes de radiations. Je n'oublierais pas Monsieur Paul AMBLARD, pour ses conseils, son accompagnement moral durant ma

courte expérience d'enseignement mais combien fructueuse, ainsi que Monsieur Robin ROLLAND pour ses conseils en électronique, Monsieur Jean-François PAILLOTIN pour le réconfort qu'il a su m'apporter avant de plonger à 20 mètres.

Une thèse n'est jamais l'œuvre d'une seule personne, celle-ci a été l'œuvre de toute une équipe, qui m'a entouré, aimé et encouragé. Je voudrais donc leur exprimer ma vive reconnaissance pour leur acharnement pour l'aboutissement de ces travaux, pour leur tendresse et leur soutien. Je voudrais que tout stagiaire qui a participé à cette thèse puisse s'identifier dans ce travail car c'est grâce à leur contribution que j'ai pu mener à bien ce travail. Je tiens donc à exprimer mes profonds sentiments à : Veronica, Gisela, Rosario, Feiza, Yen Min, Andrea, Stéphane, Said, Juan, Whe Khee, Ray, Yinkit, Chiewan, Jiang, Jenfung, Eng Yeow, Fernanda, Sameh,….

Je n'oublierais pas tous mes collègues du laboratoire TIMA et particulièrement ma très chère et tendre amie Gabriela NICOLESCU, Bogdan NICOLESU et Jean Baptiste RIGAUD ainsi que toutes les secrétaires du laboratoire TIMA, pour leur aide, leur disponibilité et leur humour. Un grand merci s'adresse aussi à mes très chers amis Sonia UBAGO et Jérôme JACQUET pour avoir accepté d'organiser mon pot de thèse, pour tous les merveilleux moments qu'on a passé ensemble, pour leur soutien et leur amour qu'ils n'ont jamais arrêté de me témoigner.

Enfin, je réserve mon dernier merci mais oh combien spécial, fort et sincère à mes très chers parents, ma famille et tous mes amis tunisiens particulièrement Sameh MAALEJ et Fathia MAGHREBI pour leurs encouragements, leur aide et surtout pour avoir cru en moi.

RESUME

Cette thèse est consacrée à l'étude du comportement de processeurs digitaux face à l'un des effets induits par l'environnement radiatif : le phénomène dit *SEU* ou *upset* qui se traduit par le basculement intempestif du contenu d'un élément mémoire comme conséquence de l'ionisation produite par le passage d'une particule chargée. Les conséquences de ce phénomène dépendent de l'instant d'occurrence et de l'élément mémoire affecté et peuvent aller de la simple erreur de résultat à la perte de contrôle d'un engin spatial.

Les techniques de durcissement ne pouvant pas garantir entièrement l'immunité face aux upsets des circuits candidats aux applications spatiales, des méthodes d'estimation des taux d'erreurs de ces applications par des tests sous radiation ou par injection de fautes s'avèrent nécessaires, dans le double but de choisir les circuits les moins sensibles à ces effets et d'étudier le comportement des applications de vol face aux upsets.

L'objectif de cette thèse consiste en la définition d'une méthode d'injection de fautes de type upset et de son expérimentation sur différentes architectures digitales afin d'étudier ses potentialités ainsi que son efficacité. La méthode proposée se base sur l'injection d'erreurs de type upset sur une carte digitale bâtie autour du processeur cible, comme conséquence de l'activation d'un signal d'interruption asynchrone. L'exécution de la séquence de traitement de l'interruption appelée CEU dans cette thèse (Code Emulant un Upset) provoquera la modification du contenu d'un bit sélectionné aléatoirement parmi les éléments de la zone mémoire sensible aux upsets du processeur.

L'implantation de cette technique a été réalisée par l'intermédiaire d'un système THESIC, testeur dédié à la qualification sous radiation de circuits intégrés. Ce système comporte deux cartes digitales (carte mère/carte fille), dont la configuration s'est révélée adaptée aux contraintes imposées par la technique d'injection de fautes proposée.

L'objectif final de ces recherches a été de démontrer que le taux d'erreurs d'une application peut être prédite à partir des résultats issus d'essais d'injection d'upsets et des mesures des sensibilités des éléments mémoires du processeur considéré. La confrontation de ces prédictions avec des mesures réalisées à l'aide d'accélérateurs de particules, a permis de montrer la validité de l'approche proposée pour différents types de processeurs.

MOTS CLES

ENVIRONNEMENT SPATIAL, TESTS AUX IONS LOURDS, SINGLE EVENT UPSET, INJECTION DE FAUTES, CEU, ARCHITECTURES DIGITALES.

ABSTRACT

This thesis aims at the study of the behavior of digital processors with respect to one of the effects of radiation environment - the Single Event Upset phenomenon, also called *upset* - which may modify the content of memory elements as the result of the silicon ionization resulting from the impact of charged particles. The consequences of upsets for a given application depend on both the occurrence instant and the perturbed memory element, and can go from innocuous result errors to system crashes which may provoke the loose of control of a space vehicle.

As design hardening techniques cannot completely guarantee the upset immunity for circuits devoted to space applications, error rate estimation methods, based on radiation ground testing and/or in fault injection experiments, are mandatory to choose the less sensitive circuits for a given space application.

The research presented in this thesis consists in the definition of a suitable method for upset-like fault injection and its experimentation for various digital architectures in order to assess its efficiency and put in evidence its capabilities. The proposed method is based on the injection of upsets in a digital board built around a processor, as the consequence of the activation of an asynchronous interruption. The execution of the instruction sequence associated with the interruption, called here CEU (Code Emulating an Upset), will provoke the modification (bit flip) of a target selected among the circuit sensitive area which comprises mainly registers and internal memory elements.

The CEU injection technique was implemented using THESIC, a system dedicated to the qualification of integrated circuits under radiation. This system is composed of two digital boards (a mother board and a daughter board) whose configuration revealed as being well adapted to constraints imposed by the studied fault injection approach.

Demonstrating that application error rates can be predicted from the results of CEU injection experiments combined with the measure of individual sensitivities to upsets of the processor’s memory elements obtained from radiation testing. The confrontation for different architectures and programs, of predicted error rates to measured ones proved the validity of the approach.

KEY WORDS

SPATIAL ENVIRONNEMENT, TESTS UNDER HEAVY IONS, SINGLE EVENT UPSET, FAULT INJECTION, CEU, DIGITAL ARCHITECTURES.

Table des Matières

Table des Figures

Table des Graphiques

Liste des Tables

Introduction

La détermination de l'aptitude à des applications spatiales de circuits fabriqués à partir de technologies commerciales fait actuellement l'objet d'un effort de recherche considérable. Ceci entraîne la mise en œuvre de tests au sol, dans le but de quantifier la sensibilité des circuits candidats face aux perturbations induites par les radiations. A partir de résultats de ces tests et des caractéristiques de l'environnement final de l'application, le taux d'erreurs en vol est généralement prédit pour chacun des circuits puis pour le système complet.

Parmi les effets des radiations, cette thèse s'intéresse particulièrement aux phénomènes appelés «upset » qui résultent en l'inversion de bits d'un élément mémoire comme conséquence du passage d'une particule énergétique dans des zones sensibles du circuit. Lorsque l'objet de l'étude est un circuit de type processeur, i.e. capable d'exécuter un jeu d'instructions ou de commandes (microprocesseurs, microcontrôleurs, processeurs de traitement de signal,...) les conséquences d'un upset peuvent être critiques, pouvant conduire dans les cas extrêmes à la perte du contrôle d'un véhicule spatial.

Pour déterminer la sensibilité aux upsets d'un circuit de type processeur, la stratégie communément adoptée consiste en l'exposition du processeur à un flux de particules pendant qu'il exécute un programme donné. L'incidence du programme exécuté sur le taux d'erreurs est un fait établi. Bien que l'utilisation de l'application finale lors des essais sous radiations n'est pas jugée indispensable pour justifier le choix d'un processeur pour un projet spatial, il est tout de même important d'étudier les effets des upsets au niveau du système, pour établir s'ils occasionnent une défaillance catastrophique du système embarqué ou une simple perte de service de courte durée. Les résultats obtenus par diverses méthodes d'injection de fautes effectuées par moyens logiciels et/ou matériels ont confirmé l'intérêt de telles études.

Cette thèse a pour but d'explorer une méthode originale pour la prédiction de la sensibilité d'un processeur face aux upsets qui surviendront lorsqu'il exécutera une application sous radiations. La méthode proposée se base sur des essais d'injection *matérielle/logicielle* d'erreurs de type upset sur une carte digitale construite autour du processeur cible. L'idée est d'utiliser des signaux asynchrones telles que les interruptions, disponibles pratiquement dans tout processeur, pour introduire lors de l'exécution d'un programme le changement du contenu d'un bit au sein du processeur. Ce bit est choisi au hasard parmi l'ensemble des éléments de mémorisation du processeur étudié accessibles par programme (registres, mémoire interne, …). L'instant de déclenchement de l'interruption est

également choisi aléatoirement. De cette manière, le programme se poursuivra dans les conditions typiques d'occurrence d'une erreur de type upset. Le déroulement postérieur à l'injection de l'erreur pourra donc donner des indications aussi bien sur *la nature que sur le taux des dysfonctionnements* provoqués au niveau système. L'objectif final de cette thèse est de démontrer que le taux d'erreurs d'une application, peut être prédit à partir des résultats issus d'essais d'injection d'upsets et des mesures des sensibilités des éléments mémoires du processeur considéré. La confrontation de ces prédictions avec des mesures réalisées à l'aide d'accélérateurs de particules, permettra de montrer la validité de l'approche proposée. La phase expérimentale de ces travaux a été mise en œuvre à l'aide d'un système de test dédié, le testeur THESIC développé lors de diverses collaborations entre TIMA et le Centre National d'Etudes Spatiales (CNES). Le choix de ce testeur comme plate-forme pour le développement d'une stratégie d'injection d'upsets fut motivé par :

- la facilité d'adaptation à tout type de processeur grâce à la stratégie carte mère/carte fille déjà utilisée pour la mise en œuvre d'essais au sol de divers circuits intégrés,
- la possibilité de munir la carte fille d'une architecture apte à l'installation de programmes de vol, destinés à fonctionner dans des équipements à bord de véhicules spatiaux.

Des essais sous radiations et des sessions d'injection de fautes ont été réalisés sur quatre architectures digitales différentes :

- La première est bâtie autour d'un microcontrôleur largement utilisé dans des applications embarquées, le 80C51 d'Intel. Elle permet d'illustrer les aspects relevants de la méthode proposée, grâce aux résultats de différents essais d'injection d'upsets de manière concurrente avec l'exécution d'un programme simple (multiplication de matrices).
- La deuxième reproduit certains aspects de l'architecture d'un MODEM digital destiné à fonctionner dans un satellite de télécommunications. Le programme final développé pour un processeur de traitement de signal, le DSP TMS320C50 de Texas Instruments, a été utilisé lors des essais d'injection d'upsets et des tests sous radiation pour obtenir des mesures du taux d'erreurs de l'application finale.
- Les deux dernières architectures sont bâties autour de deux circuits, le processeur de traitement de signal ADSP21060 de Analog Devices et le microcontrôleur TS6833216A de Motorola, tous deux inclus dans les instruments d'un projet satellite hispano-argentin. L'application de la méthode d'injection de fautes sur ces deux architectures montrera comment la technique proposée peut être adaptée à tout processeur à condition qu'il dispose d'un signal d'interruption matérielle.

La structure du rapport de cette thèse est la suivante :

- Dans le premier chapitre seront donnés les bases de la méthode d'injection d'erreurs de type upset sur des processeurs et les principes de son implantation sur une architecture digitale typique. Ensuite sont détaillées deux stratégies de prédiction du taux d'erreurs d'une application exécutée sur un processeur : l'une par injection d'upsets avec un choix aléatoire de l'emplacement de la cible à perturber et l'instant d'occurrence de l'erreur ; l'autre par injection exhaustive d'upsets.
- Dans le deuxième chapitre sera décrite l'implantation de cette technique au microcontrôleur 80C51 d'Intel. L'analyse des résultats, ainsi que la comparaison des taux d'erreurs prédits à ceux mesurés lors de l'exposition du 80C51 à différents faisceaux d'ions lourds, sont présentées dans la deuxième partie de ce chapitre. Ces faisceaux d'ions ont été issus de deux accélérateurs de particules : l'accélérateur linéaire Tandem de Van der Graff de l'Institut de Physique Nucléaire à Orsay et le cyclotron Cyclone de Louvain la Neuve en Belgique.
- Le troisième chapitre sera consacré aux résultats expérimentaux de l'application de la méthode d'injection d'upsets dans le cas du processeur de traitement de signal TMS320C50. Les résultats des tests sous radiation montreront une variante dans l'implantation de la méthode d'injection de fautes. La confrontation des taux d'erreurs prédits et ceux dérivés des tests sous radiation mettra en évidence un inconvénient majeur de la méthode d'injection de fautes proposée dans cette thèse : l'inaccessibilité aux registres critiques au fonctionnement du DSP.
- Dans le quatrième et dernier chapitre, nous présenterons deux exemples d'applications de l'approche d'injection de fautes à des architectures digitales inclues dans un projet satellite. La méthodologie d'implantation de la technique d'injection de fautes sur ces deux architectures sera présentée brièvement. Dans la suite du chapitre, sera décrite l'implantation de l'outil de calcul des facteurs de sensibilités, ainsi que l'estimation du taux d'erreurs d'une multiplication de matrices par l'intermédiaire de cet outil. Enfin, seront présentés les résultats des sessions d'injection de fautes sur ces deux processeurs ainsi que ceux dérivés des tests du TS6833216A avec des faisceaux d'ions lourds issus de deux accélérateurs de particules. La comparaison des taux d'erreurs prédits et mesurés pour le TS6833216A permettront de montrer la possible généralisation de cette technique de prédiction du taux d'erreurs d'une application quelconque exécutée sur ce processeur.

Finalement, les conclusions feront la synthèse de ces travaux, mettant en évidence les avantages et les inconvénients de la technique proposée d'injection de fautes pour l'étude du comportement de processeurs face aux perturbations et la prédiction du taux d'erreurs d'une application donnée.

CHAPITRE 1

UNE NOUVELLE APPROCHE POUR L'ETUDE DE LA SENSIBILITE AUX BASCULEMENTS LOGIQUES D'ARCHITECTURES DIGITALES A BASE DE PROCESSEURS

I. INTRODUCTION

Les progrès constants réalisés dans la technologie micro-électronique permettent la fabrication de circuits intégrés très complexes réalisant les opérations de cartes et même des ordinateurs des années 80. La réduction des dimensions et des paramètres électriques qui en résulte, a pour conséquence l'augmentation de la sensibilité à certains effets de l'environnement (ionisation du substrat dû aux radiations, perturbations magnétiques, effets thermiques) qui étaient considérés mineurs ou sans effet dans des technologies du passé. Par exemple, il est intéressant de noter que les neutrons atmosphériques, jusqu'à présent inoffensifs pour les circuits micro-électroniques, ont des énergies suffisantes pour faire basculer le contenu de cellules mémoires ou même perturber les opérations logiques dans des circuits fabriqués dans des technologies submicroniques. Ceci peut constituer une source sérieuse d'erreurs, particulièrement dans l'avionique (à 30 000 pieds le flux de neutrons est de 4 à 8 fois plus important qu'au sol) mais aussi dans les systèmes opérant au sol [1].

Les circuits digitaux fonctionnant en présence de radiations subissent des effets qui peuvent être permanents ou transitoires [2]. Les premiers résultent des charges piégées dans le volume des oxydes et apparaissent après de longues périodes d'exposition à des radiations (phénomènes de dose cumulée), alors que les derniers, appelés SEE (Single Event Effects) peuvent être provoqués par le passage d'une unique particule chargée à travers des zones sensibles du circuit. Deux types de SEE sont distingués [3]:

- Les *upsets* (Single Event Upset ou SEU) responsables de changements transitoires dans la valeur de signaux du basculement du contenu de bits d'information stockés dans des cellules mémoires. Les conséquences des SEU dépendent de la nature de l'information (état d'un signal ou contenu d'un bit) perturbée allant depuis l'erreur de calcul jusqu'à la perte de séquencement d'un processeur par exemple [4-8].

- Les *latchups* (Single Event Latchup ou SEL) résultent de la mise en conduction de thyristors parasites présents dans les circuits CMOS et provoquent des courts-circuits susceptibles d'endommager le composant s'il n'est pas mis hors tension rapidement [9].

Des détails sur les caractéristiques de l'environnement spatial ainsi que sur les effets produits par l'interaction des particules chargées avec la matière sont fournis dans [10-11]. Une étude très complète sur ce sujet peut être trouvée dans le chapitre 4 du rapport de thèse donné en référence [12]. Durant tout le reste de ce rapport de thèse, nous ne nous intéresserons qu'aux effets des radiations résultant en des upsets dans les cellules mémoires.

L'estimation de la sensibilité face aux SEU, de composants inclus dans des équipements destinés à fonctionner sous radiations est une tâche critique. Parmi ces composants, les mémoires et les processeurs, présents dans la plupart des systèmes digitaux, peuvent être identifiés comme ceux contribuant le plus au taux d'erreurs global. En effet, le nombre très important de cellules mémoires qu'ils comprennent, rend ces composants potentiellement sensibles aux SEU.

Notre objectif est d'étudier la vulnérabilité de ces composants électroniques face aux effets des radiations résultant en l'inversion de bits dans des cellules mémoires. Pour ce faire, nous avons exploré une nouvelle méthodologie d'injection de fautes qui sera utilisée d'une part pour obtenir des données expérimentales sur la nature des erreurs induites par les upsets sur des architectures digitales, d'autre part pour prédire les taux d'erreurs de ces architectures dans l'environnement final.

Ce chapitre est organisé comme suit : d'abord sont détaillées les stratégies communément adoptées pour la qualification de circuits intégrés face aux upsets, ainsi que les travaux existant sur l'injection de fautes pour l'étude du comportement des processeurs en présence d'erreurs. Puis, sont présentées la méthode d'injection de fautes proposée et son implantation à l'aide d'un testeur dédié, ainsi que et les principes de base de la stratégie de prédiction des taux d'erreurs.

II. ETAT DE L'ART

A. Stratégies de qualification de processeurs

L'établissement de la vulnérabilité aux upsets de composants électroniques, passe par l'exposition du circuit à tester (DUT : Device Under Test) à un flux de particules pendant qu'il effectue une "activité appropriée". De nombreux travaux ont été consacrés à ce problème depuis le milieu des années 1980, parmi lesquels les plus représentatifs sont ceux présentés dans les références [13-17]. Pour un circuit de type mémoire ceci peut être fait aisément par le chargement de son contenu avec un motif préétabli, dont la corruption par SEU est identifiée par la lecture de la zone mémoire considérée pour la comparer au motif initial. C'est la stratégie communément adoptée pour qualifier une mémoire face aux effets des radiations [18-19]. A l'opposé, pour des circuits de type processeur il n'y a pas de stratégie bien établie. En effet, les conséquences d'un SEU au niveau d'un processeur sont difficiles à prédire car elles dépendent de la nature de l'information perturbée (bit utilisé pour coder un état, une variable, une donnée intermédiaire) ainsi que de l'instant d'occurrence de cette perturbation (information perturbée déjà utilisée ou à consommer par la suite). Une étude approfondie à ce propos peut être trouvée en [20].

Les données publiées sur les sensibilités de processeurs face aux SEU sont généralement obtenues à partir d'essais sous radiations durant lesquels le programme exécuté par le DUT consiste en l'inspection séquentielle de chacune des cellules mémoires accessibles à l'utilisateur (par exécution d'une séquence adéquate d'instructions) jusqu'à détection d'erreurs. Les éléments mémoires typiquement considérés sont des registres généraux, les registres spéciaux et la mémoire interne. Des exemples typiques d'utilisation de ces stratégies, dites *statiques*, peuvent être trouvés dans les références [21-27]. L'application finale, cependant, activera ces zones d'une manière différente tout en employant des instructions inutilisées par les tests statiques, activant donc d'autres cibles potentielles aux SEU. L'idéal serait l'utilisation du programme final d'application lors des essais sous radiations, mais généralement ce programme n'est pas connu ou disponible au moment où les essais de qualification sont réalisés sur des circuits candidats aux équipements spatiaux. Dès lors, une autre méthode de test, appelée souvent « *test dynamique* », consiste en l'exécution de programmes simples, supposés comme étant en quelque sorte représentatifs, et l'observation uniquement des résultats de sortie de ces programmes, ceci en espérant activer les zones sensibles du circuit de manière proche de celle de l'application de vol conçue pour ce circuit. A partir des résultats issus de ces deux tests, seront calculées deux grandeurs :

- La section efficace individuelle $\sigma_{SEU}(R)$ d'un registre R ou la section efficace du composant σ_{SEU}, représentant la sensibilité d'un circuit intégré en termes de nombre de particules nécessaire pour provoquer un upset. Cette valeur est issue de l'exécution du test statique,
- La section efficace du processeur étudié exécutant une application donnée $\sigma_{SEU}(application)$, égale au quotient entre le nombre d'erreurs détectées et le nombre total de particules auxquelles il a été exposé. Par abus de langage et durant toute la suite de ce rapport, cette valeur sera appelée section efficace de l'application :

$$\sigma_{SEU}(application) = \frac{Nombre\ d'erreurs}{Fluence\ de\ particules} \qquad (1.1)$$

A partir de la section efficace d'une application peut être déduit le taux d'erreurs τ_{SEU}, selon la formule (1.2) :

$$\tau_{SEU} = \frac{\sigma_{SEU}(application)}{\sigma_{SEU}} \qquad (1.2)$$

En [22-25], sont présentés les résultats de l'application à divers processeurs, des deux stratégies de test (statique et dynamique) sous radiations, qui diffèrent seulement dans la nature du programme exécuté par le processeur lors des essais. Les résultats obtenus à l'aide d'accélérateurs de particules montrent que les sections efficaces issues d'essais avec des programmes *benchmark* dépendent du programme testé et sont inférieurs d'au moins un ordre de grandeur par rapport à ceux résultant de la stratégie statique. Ceci n'est pas surprenant car, en effet, pour provoquer une erreur au niveau des sorties d'un programme, un upset doit survenir durant la période où l'élément mémoire perturbé contient une information utile (i.e. qui va être exploitée par la suite). De plus, comme de nombreux registres et éléments mémoires peuvent ne pas être utilisés par un programme, les upsets les affectant n'auront pas d'effet au niveau de l'exécution du programme, et donc ne contribueront pas au taux d'erreursde l'application.

B. Facteurs de sensibilité

La section efficace dérivée d'un test statique d'un processeur est parfois considérée comme étant une mesure directe de la section efficace d'une application due aux erreurs observables résultantes des SEU dans un système. Ceci suppose que tout SEU apparaissant dans les cellules mémoires d'un processeur induit des erreurs dans les résultats du programme. Cette interprétation constitue le pire cas. En effet, les SEU ne causeront d'erreurs que lorsqu'ils surviendront durant la période de sensibilité d'un registre ; cette période appelée *duty-cycle* dans la littérature anglo-saxonne [14] est définie comme étant la somme des temps pendant lesquels l'élément considéré est

sensible aux radiations et donc durant lequel des upsets sur cet élément peuvent être observés car ils provoqueront des erreurs en sortie [14][26]. Quand cette valeur est exprimée comme un pourcentage par rapport à la durée totale du programme, elle est appelée facteur de sensibilité *(duty-factor).*

Afin de clarifier le principe de calcul des facteurs de sensibilité, nous donnons un exemple de programme (figure 1.1) pour lequel sera calculée le facteur de sensibilité de l'accumulateur A. Dans un but de simplification du cas étudié, nous supposons que chaque instruction est exécutée en 1 cycle d'horloge et que l'accumulateur n'est utilisé que durant la partie du code grisée dans la figure.

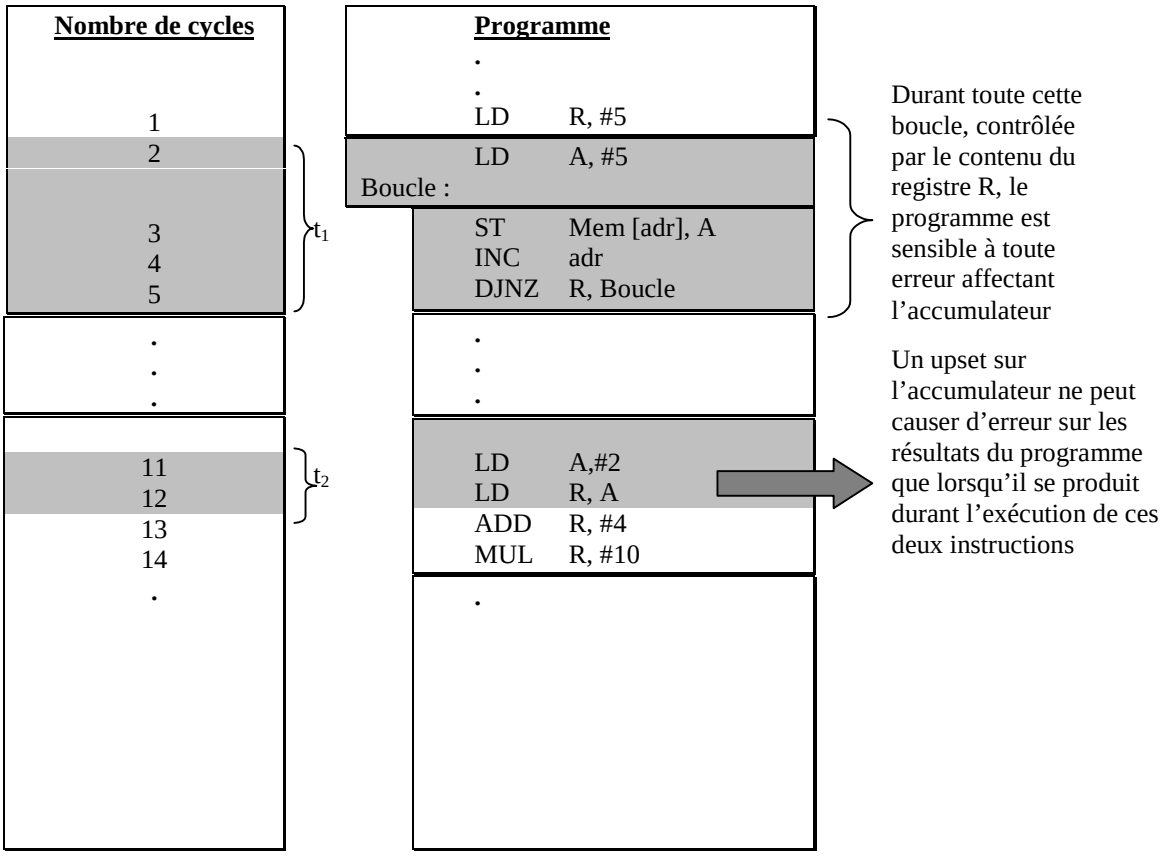


Figure 1.1 : Exemple d'étude de période et du facteur de sensibilité

A titre d'exemple, et en supposant que cette application nécessite 1000 cycles d'horloge pour être exécutée, et sachant que ces portions de code sensibles nécessitent respectivement 20 et 2 cycles pour être exécutées, le facteur de sensibilité de l'accumulateur est estimée comme étant égale à 2,2% :

$$\begin{cases} Période\ de\ sensibilité\ (A) = t_1 + t_2 = 22\ \ cycles \\ Facteur\ de\ sensibilité\ (A) = \dfrac{t_1 + t_2}{Nombre\ de\ cycles\ \ \max imum} = 2{,}2\ \% \end{cases}$$

La section efficace d'une application particulière exécutée sur un microprocesseur dépend donc des valeurs des facteurs de sensibilité. Par conséquent une mesure précise de la section efficace de l'application devrait alors être calculée comme étant la somme des sections efficaces individuelles $\sigma^{i}_{SEU}(R_i)$ par registre R_i pondérées par les coefficients fi qui correspondent aux facteurs de sensibilité des registres considérés (équation 1.3) :

$$\tau_{SEU} = \sum_{i=1}^{n} \sigma^{i}_{SEU}(R_i) f_i(R_i) \qquad (1.3)$$

où n est le nombre total de registres dans un microprocesseur. Bien entendu, le terme registre désigne ici non seulement les registres du processeur, mais aussi tout élément de mémorisation interne (mots mémoire, ports,…) accessible par exécution d'une séquence appropriée d'instructions.

Pour vérifier ce modèle de calcul de la sensibilité de processeur face aux SEU, les auteurs de la référence [25] ont considéré un programme qui utilise la majorité des éléments fonctionnels (accumulateur, compteur de programme, pointeur de pile, registres de données et d'interruption) et les instructions de base d'un microprocesseur 8 bits, le H80C85. Une analyse de ce programme a permis de déterminer les facteurs de sensibilités des différents éléments, et de calculer la sensibilité globale du circuit à partir des sensibilités des différents éléments le constituant.

Les résultats mesurés en utilisant ce même programme comme séquence de test pendant des essais sous radiation, sont en bonne corrélation avec les résultats issus du calcul effectué avec la formule 1.3. Cette étude montre alors qu'il est possible de déterminer la sensibilité d'un circuit pour un programme donné à partir de celle de ses divers éléments sensibles et notamment pour le programme d'application final. Sachant que les programmes sont généralement complexes au point que le séquencement des instructions ne peut pas toujours être prévu à l'avance (présence de boucles et de sauts conditionnels, dépendance des données extérieures) l'estimation des facteurs de sensibilité devient une tâche difficile. Les auteurs proposent la solution de partager le programme en modules où l'analyse peut être réalisée de manière simple, de calculer les facteurs de sensibilité des éléments mémoires pour chacun de ces modules, puis de faire la somme sur tous les modules pour déduire la section efficace globale d'une application. Ce travail a été réalisé manuellement. A notre connaissance, depuis sa première publication en 1988 aucun outil automatique pour le calcul des facteurs de sensibilité d'une application donnée n'a été développé, à l'exception de la tentative faite par les auteurs de [27].

En effet, dans la référence [27], il a été développé un outil de calcul des facteurs de sensibilités dans les positions de mémoire interne et les registres du 80C51 exécutant des programmes simples. Durant le traitement des

programmes et grâce aux fonctionnalités du débogueur[1] du 80C51, les instructions exécutées, l'état de tous les registres ainsi que les positions de la mémoire interne sont automatiquement sauvegardés dans le buffer du simulateur. L'idée est d'utiliser cette fonctionnalité pour l'enregistrement du contenu de ce buffer dans un fichier afin qu'il puisse être analysé pour la détermination des facteurs de sensibilités des zones sensibles du microcontrôleur 80C51. Pour ce faire, un programme a été écrit en langage de haut niveau (Visual C^{++}), pour le traitement et l'analyse automatiques de ce fichier, afin d'extraire les valeurs des facteurs de sensibilité qui seront enregistrées dans un deuxième fichier [15].

Sachant que le buffer ne peut stocker des informations que sur les 256 dernières instructions exécutées, l'exécution doit être arrêtée manuellement après chaque paquet de 256 instructions pour éviter la réécriture de cette information, ce qui constitue une limitation sérieuse lorsque la taille du programme étudié est importante, comme c'est le cas de la plupart des applications spatiales. Depuis cette tentative, aucun outil de calcul de facteurs de sensibilités des zones sensibles d'un processeur n'a été présenté dans la littérature spécialisée.

C. Méthodes d'injection de fautes

Les impératifs de sûreté de fonctionnement des systèmes destinés à fonctionner dans l'espace ne cessent d'augmenter avec l'accroissement de la complexité des traitements effectués à bord de véhicules spatiaux et l'utilisation de circuits intégrés de plus en plus miniaturisés. Des études approfondies des méthodes les plus adaptées à la qualification d'architectures digitales bâties autour de processeurs candidats aux applications spatiales, deviennent de nos jours nécessaires même pour des applications fonctionnant dans l'atmosphère terrestre. La dépendance du programme exécuté, des résultats dérivés des tests sous radiation nécessite des études minutieuses du comportement de microprocesseur face à ces effets. Une telle exigence implique la description du fonctionnement du microprocesseur en présence d'erreurs. Compte tenu que le temps moyen entre l'occurrence des fautes dans des environnements radiatifs est long et que les expériences de radiation sont coûteuses, différentes techniques d'injection de fautes ont été développées pour l'évaluation des sections efficaces de différentes applications à moindre coût. L'état de l'art montre l'existence de quelques techniques d'injection de fautes, basées sur des outils logiciels et/ou matériels.

Dans la référence [28] a été présenté un environnement global pour l'injection de fautes qui comporte, quelque soit l'application étudiée, trois principaux modules : module de génération de listes de fautes, module d'injection de

[1] DS51 du simulateur de Keil Elektronix

fautes et module d'analyse des résultats du programme. Le module d'injection de fautes, comme son nom l'indique, s'occupe de l'injection de fautes à une position mémoire et à un instant sélectionnés à partir d'une liste fournie par le module de génération de fautes qui peut être exhaustive ou aléatoire. Durant l'exécution de l'application, ce module contrôle l'avancement du programme étudié, en activant simultanément, à un instant donné le système d'injection de fautes, celui de l'observation des résultats ou celui du contrôle du chien de garde logiciel dans le cas où le système perd le contrôle (cas de perte de séquencement). Le module d'analyse des résultats permet de produire un rapport sur l'ensemble des résultats issus de la campagne d'injection de fautes, comportant essentiellement les taux d'erreurs sur les sorties du programme et le taux de perte de séquencement du processeur étudié. Les différentes versions de l'approche d'injection de fautes partagent les trois modules présentés précédemment (génération, injection et analyse de fautes) et peuvent être classifiées en :

- solutions logicielles se basant sur le fonctionnement en mode instruction par instruction «mode *trace*» [29],
- de solutions hybrides (logicielle/matérielle) : utilisant des cartes digitales dédiées à l'injection de fautes [30-32], exploitation du mode de débogage (BDM) existant sur certains microprocesseurs Motorola [29]),
- de techniques de simulation (simulateurs/émulateurs de processeurs [33], injection en langage haut niveau [34] et [35]),

Compte tenu que les méthodes basées sur les moyens purement logiciels sont spécifiques à quelques applications, nous nous contenterons dans la suite de présenter des exemples d'application des deux dernières techniques (utilisant des moyens hybrides ou se servant de simulateurs).

1) La version hybride matérielle/logicielle

Cette technique est basée sur l'injection de fautes par activation d'une interruption par un système matériel extérieur. Tout le système d'injection de fautes est implanté sur une carte extérieure qui permet de calculer le nombre d'instructions exécutées en observant l'état des broches du processeur. Cette carte interagit en temps réel avec l'application étudiée ce qui évite tout ralentissement du système. Tout de même, à chaque étude d'un nouveau circuit, une carte spécifique au DUT doit être conçue et développée pour l'injection de fautes. Cette méthode a été récemment évoquée dans [32] parmi la liste de méthodes d'injection de fautes pour la validation d'applications critiques. Mais à notre connaissance, aucun résultat de son application à des circuits n'a été publié jusqu'à présent.

Une autre méthode, utilisant les signaux d'exception présents dans certains processeurs, a rendu possible l'injection de fautes [30]. Celle-ci est basée sur l'exécution du code associé à une exception dont le rôle est

d'injecter une erreur à une position quelconque des zones sensibles d'un processeur. Cette technique matérielle/logicielle permet l'injection de fautes avec le minimum d'interférence avec l'application cible, qui n'est pas modifiée de point de vue matériel et fonctionne à la fréquence nominale. Les principaux avantages sont ici les faibles coût et encombrement matériel. L'inconvénient majeur est qu'elle suppose que le processeur étudié est équipé avec des routines d'exception particulières, ce qui constitue une sérieuse limitation pour la généralisation à d'autres processeurs. Dans la référence citée, l'implantation de cette approche s'est limitée à un processeur de la famille Power PC (le MPC601).

2) Utilisation des techniques de simulation

Les auteurs de la référence [28] ont implanté l'injection de fautes à l'aide d'un simulateur[2] où ils ont utilisé un outil spécial qui leur a permis d'écrire un fichier script capable de piloter l'exécution du programme à étudier (charger le code, exécuter le programme, insérer des points d'arrêt, modification des variables). Les transformations apportées au code sont mineures. Tout de même, la lenteur des simulations constitue une grande limitation pour ce type de technique d'injection de fautes.

En [33] a été explorée une méthode de prédiction du taux d'erreurs d'un système basée sur l'injection logicielle d'upsets à l'aide d'un simulateur[3] du microcontrôleur 80C51, commercialement disponible. Dans le cadre de cette étude, une application de vol d'un satellite de l'agence DERA (Grande Bretagne), conçue autour du microcontrôleur 80C51, a été testée. Le simulateur du 80C51 fut utilisé pour injecter des upsets sur les zones mémoires internes du 80C51 de manière concurrente avec l'exécution du programme d'application.

Même si les résultats obtenus ont été en bonne corrélation avec les résultats des essais au sol, cette technique a été considérée comme n'étant pas généralisable principalement à cause de la durée des simulations, des difficultés d'automatisation, et du fait qu'elle ne permet pas (du moins de manière simple) la simulation d'upsets sur des registres critiques (compteur de programme par exemple) ou sur le code du programme lui même. De plus il est nécessaire de connaître le programme en détail et l'architecture interne du microprocesseur pour être capable d'utiliser efficacement le modèle de simulation. Enfin, pas tous les simulateurs de processeurs sont munis d'un mode d'exécution simultanée de deux programmes, permettant ainsi d'injecter des fautes au même temps que l'exécution du programme, ce qui limite l'utilisation de cette méthode à quelques processeurs.

[2] FLEX développé par Noral Micrologics Ltd
[3] Le Keil Elektronix

Dans la référence [35] sont décrites des expériences basées sur la simulation d'upsets et réalisées à l'aide d'une description de haut niveau (HDL) d'un microprocesseur 16 bits ayant un jeu d'instructions réduit. De telles simulations permettent d'injecter un upset à des instants et dans des cellules mémoires choisis de manière pseudo-aléatoire, et par conséquent d'étudier d'une manière représentative les effets des upsets sur le comportement d'un processeur. La non- disponibilité de descriptions HDL de tout processeur et la complexité de les piloter sans la coopération des ingénieurs systèmes ou les concepteurs (l'auteur a mentionné la période de 6 mois de traitement des simulations et d'analyses des résultats) rend difficile l'utilisation de telles descriptions pour la construction d'une méthode générale de quantification des sections efficaces de différentes applications.

En conclusion de cette étude de l'état de l'art, on peut dire qu'aucune des méthodes citées ne constitue une solution globale, puisqu'elles sont orientées principalement vers des applications spécifiques ayant des caractéristiques particulières qui peuvent être difficilement appliquées à d'autres cas de figures. D'un autre côté, aucune parmi elles n'a été ciblée aux systèmes embarqués bâtis autour de microprocesseurs, d'où **l'objet d'étude de cette thèse.**

Cette thèse présente une étude de l'application d'une technique hybride d'injection de fautes, qui est automatisable et qui a pour but de caractériser et de quantifier les effets des upsets sur le fonctionnement des architectures bâties autour de microprocesseurs. Cette technique consiste en l'injection de fautes, simultanément avec l'exécution d'un programme dans le processeur testé, grâce à l'utilisation de la réponse du circuit à l'activation d'une interruption. Elle peut être appliquée à toute architecture à base de microprocesseurs avec une transformation matérielle minimale (addition de quelques nouvelles fonctionnalités), à condition que ce microprocesseur dispose d'un signal asynchrone de type interruption.

Nous avons défini une méthodologie d'estimation de la section efficace d'une application face aux SEU, basée sur un test sous radiation (ayant pour but l'évaluation de la section efficace aux SEU d'un processeur) et sur des expériences d'injection de fautes pour évaluer statistiquement le nombre d'upsets qui résulteront en des erreurs lorsque le programme étudié est exécuté.

Dans la suite de ce chapitre, nous exposerons les principes de base, les avantages, les inconvénients et la stratégie d'implantation, de la méthode d'injection de fautes objet d'étude de cette thèse. Ces fautes sont exclusivement de type "inversion intempestive de bits" d'un élément de mémorisation. La principale caractéristique de cette approche est la possibilité de prédire la section efficace d'une application sans recours aux tests sous

radiation (sauf pour la détermination de section efficace aux SEU). La méthodologie de prédiction de la section efficace d'une application donnée est décrite à la fin de ce chapitre. Un article synthétisant l'ensemble de ces travaux, qui ont été réalisés en collaboration avec le Centre National d'Etudes Spatiales [36], a été présenté à la conférence NSREC-2000 et publié dans la revue *Transactions on Nuclear Sciences* [37]. Une demande de brevet[4] CNRS, portant sur certains aspects de l'approche développée au cœur de cette thèse, a été déposée.

III. APPROCHE PROPOSEE POUR L'INJECTION D'UPSETS

A. Principe

Nous supposerons dans la suite, que l'objet de l'étude est une architecture digitale organisée autour d'un processeur (machine de type Von Neumann) capable d'exécuter une séquence d'instructions ou de commandes stockées dans une mémoire (externe ou interne) et de répondre à des signaux d'entrée asynchrones telles que les interruptions [38-39]. En principe, ce processeur est considéré comme pouvant être programmé pour réaliser de manière directe ou indirecte, des opérations de lecture et d'écriture de chacun des emplacements de la mémoire externe, ainsi que des registres et zones mémoires internes. Par contre, sauf cas particulier, le jeu d'instructions n'offre pas de moyens de lecture/écriture pour un ensemble d'éléments mémoire internes (des bascules et latches de la partie contrôle du processeur par exemple).

Des upsets peuvent être provoqués dans des processeurs comme conséquence de l'exécution d'un code adéquat dont la structure dépendra essentiellement de certaines caractéristiques de la cible (fonctions, accès en lecture/écriture). Par exemple, pour provoquer un basculement des bits d'un registre *R* d'usage général ou d'un emplacement mémoire interne ou externe accessible par adressage direct, seules sont nécessaires quelques instructions simples pour réaliser les tâches suivantes:

- lecture du contenu de la cible de l'upset (mot mémoire ou registre à modifier),
- opération de type *Ou Exclusif* de la valeur lue avec une valeur appropriée (masque possédant un "1" à l'emplacement des bits qui doivent être inversés et "0" ailleurs),
- écriture de la valeur corrompue dans le récipient cible.

Pour la plupart des processeurs, les jeux d'instructions permettent la réalisation de ces tâches en seulement quelques instructions écrites en assembleur, qui une fois compilées donneront lieu à des codes n'accédant pas une dizaine de mots. L'aspect crucial pour simuler un upset est l'insertion de ces codes dans le programme en cours

[4] R. Velazco, S. Rezgui, *Procédé pour l'estimation du taux d'erreurs d'architectures digitales exposées à des radiations*, Demande de brevet CNRS déposé le 16 janvier 2001.

d'exécution, pour qu'il soit exécuté à l'instant souhaité pendant l'exécution d'un programme quelconque. Pour que la faute injectée reproduise fidèlement les upsets survenant comme conséquence de l'effet des radiations, l'exécution de ce code doit avoir pour seul effet la corruption de la valeur de la cible de l'upset, laissant inchangés les éléments de mémorisation restants du processeur.

Une solution simple et puissante consiste à se servir du mécanisme d'interruption disponible dans la presque totalité de processeurs. Pour ce faire, le code provoquant l'upset terminé par une instruction de type "retour d'interruption" doit être enregistré à une adresse déterminée dans la mémoire, afin d'être utilisé comme programme de traitement de l'interruption. En effet si le processeur est configuré de manière appropriée il exécutera typiquement les étapes données ci-dessous en réponse à l'activation d'une interruption :

- arrêt de l'exécution du programme en cours après avoir complété l'exécution de l'instruction courante,
- sauvegarde du contexte (au moins la valeur du compteur programme PC) dans une pile,
- branchement vers le programme de traitement de l'interruption provoquant donc l'inversion de(s) bit(s) cibles choisis,
- restauration du contexte depuis la pile, puis continuation de l'exécution du programme interrompu.

Il suffit donc d'activer l'interruption à différents moments de l'exécution d'un programme pour provoquer l'injection d'upsets. La méthode entre donc dans la catégorie hybride (matériel/logiciel).

Pour clarifier la présentation de cette méthode, dans la suite de ce rapport les séquences d'instructions provoquant de tels basculements seront appelés codes *CEU (Code Emulant un Upset)*. De même, l'emplacement mémoire perturbé suite à l'exécution du code CEU sera appelé la *cible du CEU*. Enfin, pour établir les limitations de la méthode d'injection d'erreurs étudiée face aux effets singuliers de type SEU survenant sous radiations, la modification de la cible du CEU qui résulte de l'exécution du code CEU sera également appelé *CEU (Code Emulated Upset)*. A titre anecdotique, il doit être noté que cette apparente confusion des termes adoptés respecte la non différentiation qu'est aussi faite généralement entre *cause* (l'impulsion de courant résultant du passage d'une particule chargée) et *effet* (le basculement du contenu d'un point mémoire) tous deux appelés SEU dans la plupart des publications portant sur les effets des radiations sur les circuits et les systèmes. Dans notre cas, nous provoquerons des CEU (basculements des bits) dans des processeurs, comme conséquence de l'injection de CEU (codes émulant des upsets) dont l'exécution sera déclenchée par une interruption. Dans la table 1.1 sont donnés les codes CEU (en assembleur générique) pour des cas typiques de cibles de CEU.

Table 1.1 : Exemples en langage assembleur générique de codes CEU correspondant à diverses cibles.

XOR R, #masque
RETI
	PUSH R
	LD R, mem(adr)
	XOR R, #masque
	LD mem(adr), R
	POP R
	RETI
		DEC SP
		POP R ;R :=PC
		XOR R, #masque
		PUSH R
		RETI
			LD R,#code Jump
			ST mem(adr-1)
			DEC SP
			POP R ;R :=PC
			ST R, mem(adr)
			XOR SP, #masque
			JUMP adr-1

(a) Registre général corruption directe	**(b) Mot mémoire corruption via R**	**(c) Compteur Programme corruption via la pile**	**(d) Pointeur de Pile (SP) corruption via la pile**

*- **R** désigne l'un des registres,*
*- **adr** représente une adresse mémoire (adressage direct via R)*
*- **masque** désigne une constante comportant un 1 (à la place du bit à perturber) parmi des 0*

La dernière étape après avoir injecté un CEU est l'observation de ses effets au niveau de l'exécution du programme en cours. La manière de mettre en œuvre cette étape est fortement liée au type du programme exécuté. Dans le cas des programmes simples tels qu'une multiplication de matrices, les variables d'entrée sont fixées depuis le début et le bon fonctionnement du programme est jugé par l'exactitude des résultats obtenus.

Dans le cas des applications complexes, on ne peut pas se contenter de fournir des données statiques aux entrées du programme car celles-si sont variables en fonction du temps et des paramètres du système global. Une solution consiste à faire varier les variables d'entrée lors de l'exécution du programme en stockant différents stimuli dans la mémoire externe. Quand la vérification uniquement des résultats de sortie s'avère insuffisante, il est nécessaire de rajouter des points d'observation au sein du programme lui-même. Ceci donne la possibilité de contrôler tout au long de l'exécution du programme le fonctionnement de l'application face aux perturbations. Bien évidemment, les rajouts nécessaires pour la contrôlabilité des entrées et l'observabilité des sorties d'un programme dépendent du type du programme étudié. Dans le contexte de cette étude nous allons considérer que la "justesse" dans l'exécution d'un programme peut être déterminée en comparant les résultats obtenus aux valeurs attendues en absence d'injection de fautes.

B. Avantages et limitations de la méthode proposée

Nous rappelons que dans l'approche étudiée les upsets sont injectés comme conséquence de l'exécution d'un programme particulier de traitement d'interruption, le code CEU, pouvant contenir a priori des instructions accédant à n'importe quelle région de la mémoire. Ceci donne deux avantages principaux face à d'autres méthodes, en particulier à celles basées sur l'utilisation d'émulateurs de processeurs.

Le premier résulte de la possibilité qu'elle offre de prendre en compte *l'aléatoireté* de l'instant d'occurrence des upsets. Des dispositifs permettant l'activation de l'interruption de manière pseudo-aléatoire peuvent facilement être

conçus et implantés à l'extérieur de la carte testée, à l'aide de timers ou de compteurs. L'intrusivité matérielle de la méthode est donc minimale.

Le deuxième réside dans le mécanisme même d'injection utilisé, qui est basé exclusivement sur la perte de séquencement du flot d'exécution en cours vers un programme préalablement chargé dans une zone mémoire adéquate. L'intrusivité logicielle est à nouveau minimale (les seules contraintes sont la configuration du système d'interruptions du processeur et le stockage d'un code CEU préparé à l'extérieur de l'architecture testée ou résidant une fois pour toutes dans celle-ci). La diversité des cibles qui peuvent être perturbées par l'approche CEU, permet l'étude des conséquences d'erreurs critiques. Deux exemples typiques sont :

- la corruption du code du programme en cours d'exécution. L'occurrence de ce type d'erreur est fréquente dans l'espace dans tous les cas où le programme est exécuté dans une mémoire statique (interne ou externe).

- la possibilité d'injecter des upsets dans des registres critiques, parfois inaccessibles par programme, par corruption des données sauvegardées dans la pile après activation de l'interruption.

Parmi les registres critiques accessibles par ce mécanisme se trouvent principalement le registre compteur programme (PC), le pointeur de pile (SP) et parfois les registres des indicateurs (flags). La cible « apparemment inaccessible » la plus parlante est certainement PC. En effet, étant utilisé pour adresser l'instruction en cours d'exécution, le contenu de PC n'est pas lisible directement à travers le jeu d'instructions. Comme l'injection d'upsets implique une opération XOR entre sa valeur courante et un masque, cette limitation exclurait PC de l'ensemble des cibles de CEU. Cependant, l'étude des conséquences des upsets sur PC est indispensable car ils sont fréquemment à l'origine de pertes de séquencement. Pour perturber le contenu du PC, il faut pouvoir lire sa valeur et la corrompre. Cette valeur du compteur de programme contenant l'adresse de retour est systématiquement sauvegardée dans la pile, une fois que le processeur a reçu l'interruption matérielle. Une manière simple de la perturber consiste à la reprendre de la pile, la modifier, puis la retourner là où on l'a prise (à la même adresse de la pile). La corruption de PC via la pile est illustrée par le code donné dans la troisième colonne de la table 1.1-c.

Enfin, il faut noter que la procédure d'injection de CEU utilise le registre pointeur de pile (SP), élément clef pour garantir le retour au programme cible. Ce registre étant également utilisé dans la plupart des programmes pour gérer les appels et retours de procédures, il doit aussi pouvoir être cible de l'injection de fautes. Nous avons résolu ce « cercle vicieux » d'une manière relativement simple. En effet dans le cas où SP est choisi comme cible d'un CEU, la première instruction du code CEU est l'écriture à une adresse déterminée (notée *adr* dans la table 1.1), du

code opération correspondant à un saut inconditionnel. A l'adresse suivante est sauvegardée la valeur de PC (adresse de retour au programme en cours d'exécution) à partir de celle inscrite dans la pile. SP est ensuite corrompu (provoquant la perte de l'adresse de la pile). La séquence se termine par un branchement inconditionnel à l'adresse *adr*. Ce cas, illustré dans la table 1.1-d, est à priori le seul où le code CEU ne se termine pas par l'instruction de retour d'interruption.

En conclusion, l'une des caractéristiques principales qui fait de cette méthode une stratégie adaptée à l'étude des conséquences des upsets sur des applications digitales, est son intrusivité temporelle minimale provenant du fait qu'il s'agit d'une *auto-injection* d'erreurs. En effet, le temps nécessaire à l'injection d'un CEU est d'une part relativement court (quelques dizaines de cycles d'horloge) et d'autre part indépendant de la complexité de l'application. Ceci, plus la possibilité d'automatisation, rend possible l'itération de cette approche autant de fois que nécessaire pour en dériver des statistiques relatives aux effets des upsets sur le fonctionnement de programmes complexes. En effet, par simple réinitialisation (ou modification) de la zone mémoire dans laquelle est stocké le code CEU, pour qu'il corresponde à la corruption d'une autre cible, le système est prêt pour une nouvelle injection d'upset.

Parmi les principales limitations de la méthode d'injection d'upsets basée sur des interruptions, doit être mentionnée l'impossibilité de corrompre toute cible dont le contenu est non contrôlable/observable à travers le jeu d'instructions. Il est évident que de telles cibles existent dans tout processeur aussi bien dans la partie contrôle (bascules, registres zones mémoires internes,…) que dans la partie opérative (registres tampons de l'unité arithmétique et logique,….) et que leur nature et nombre font partie des informations réservées aux concepteurs. Cependant, il peut être supposé que pour des processeurs complexes, disposant de mémoires internes de taille importante (plusieurs dizaines voire de centaines de Kbytes) le nombre de cibles inaccessibles est bien inférieur (voire négligeable) face à celui des cibles accessibles par CEU. Ceci donnerait aux sections efficaces des programmes dérivées de l'application de cette approche d'injection de fautes une certaine représentativité par rapport à celles obtenues suite à des essais en ambiance radiative.

Une deuxième limitation réside dans l'instant de prise en compte de l'interruption, qui en général a lieu après la fin de l'exécution de l'instruction en cours. Ceci implique que des upsets survenant durant l'exécution d'une instruction ne pourront pas être simulés par l'approche CEU.

Il est donc clair que les conséquences des SEU dépendent fortement de l'instant d'occurrence et de la cible affectée. Il existe certainement de très nombreuses configurations *(instant* x *cible)* d'upsets ayant des conséquences critiques et non simulables par l'approche CEU. C'est pour cette raison que la méthode proposée ne doit en aucun cas se substituer aux tests sous radiations. Cependant, les cas étudiés que nous exposerons par la suite, montreront que les sections efficaces des applications issues d'essais d'injection de CEU pourraient être utilisées conjointement avec des essais sous radiations réalisés avec une stratégie statique, pour estimer la section efficace d'un programme quelconque sans qu'il soit jamais utilisé lors d'essais sous radiations au sol. Cette potentialité, qui reste à prouver par l'expérimentation, constitue l'un des intérêts majeurs de l'approche CEU.

Pour une plus grande clarté des exemples présentés dans la suite, il est supposé que tous les mots mémoires et registres ont le même nombre d'octets et que SP pointe vers le premier mot libre d'une pile qui contient à son sommet la valeur de PC.

C. Quelques considérations pour l'implantation de la méthode d'injection de CEU

Tel qu'il a été exposé précédemment, l'injection de CEU implique, une fois la cible du CEU et son instant d'occurrence déterminés :

(a) l'élaboration du code CEU,
(b) l'enregistrement du code CEU dans une zone mémoire adéquate,
(c) l'activation d'une interruption,
(d) la saisie et la comparaison des résultats à des valeurs de référence.

Pour ce qui est de la tâche (a) *l'aléatoireté* du choix de la cible implique le développement d'une routine paramétrable permettant la génération de valeurs pseudo - aléatoires comprises entre certaines limites. Par exemple, la sélection d'une cible de CEU parmi les mots d'une zone mémoire implique la génération d'une valeur aléatoire de l'adresse, qui doit être donc comprise entre les valeurs de début et de fin de la zone considérée. Le choix aléatoire d'une cible parmi un certain ensemble de registres (auxquels on associe par exemple des numéros de codes successifs), peut également être fait à l'aide de la même procédure avec des bornes correspondantes. La procédure de génération aléatoire de nombres doit donc être de « bonne qualité » aussi bien pour de petits que pour de grands nombres. Une fois la cible choisie, la composition du code objet CEU peut être réalisée par attribution de valeurs aléatoires préalablement déterminées à des parties variables du code générique associé à chaque type de cible. A titre d'exemple, si on traite le cas d'injection d'un CEU dans une zone mémoire d'un microcontrôleur Intel 80C51

(chemin de données 8 bits avec 128 mots de mémoire interne), seuls sont nécessaires les affectations de trois octets dans une séquence CEU générique, correspondant à l'adresse cible et au masque indiquant les bits à corrompre. La table 1.2 illustre cet exemple, les autres cas (injection de CEU dans des registres généraux ou des registres de contrôle) différent seulement par le nombre et la nature de paramètres à affecter (valeurs minimale et maximale entre lesquelles ils sont compris, nombre de bits).

Table 1.2 : Obtention du code objet CEU pour perturber la mémoire interne du 80C51

Code CEU assembleur		Code objet	
		Code opération du…	Opérande
PUSH	R	PUSH R	
LD	R, mem(adr)	LD(R←mem(adr))	adr
XOR	R, #masque	XOR	masque
LD	mem(adr), R	LD(mem(adr)←R)	adr
POP	R	POP R	
RETI		RETI	

La carte digitale responsable de l'activation de l'interruption de déclenchement de l'injection de CEU, peut facilement être adaptée pour réaliser les étapes (b) et (d) par une programmation adéquate. Seule contrainte : le code CEU et les résultats de l'exécution du programme testé doivent être enregistrés dans une mémoire accessible à la carte digitale bâtie autour du DUT et à la carte d'injection de CEU. La figure 1.2 illustre le principe de base de la méthode d'injection de CEU par l'intermédiaire du mécanisme d'interruption.

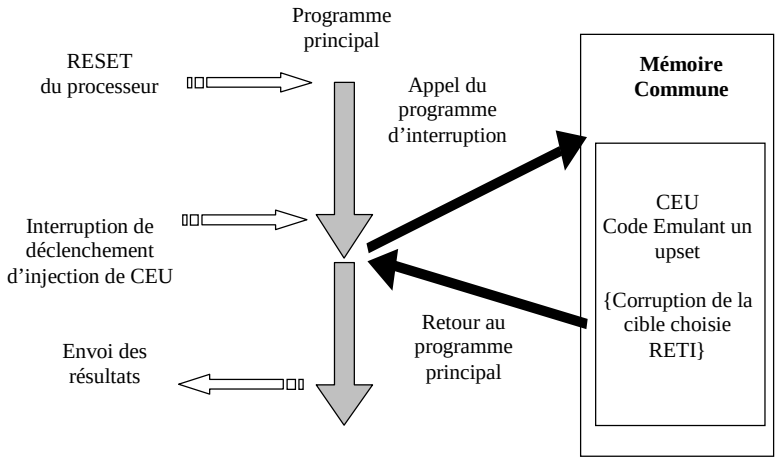


Figure 1.2 : Principe de fonctionnement de la méthode d'injection de CEU

IV. IMPLANTATION DE LA METHODE D'INJECTION DE CEU A L'AIDE DU TESTEUR THESIC

A. Introduction

L'implantation de l'approche d'injection de CEU proposée dans des architectures digitales à base de processeur, entraîne l'implantation de dispositifs matériels/logiciels permettant de réaliser de façon itérative les fonctions suivantes :

(a) préparation et chargement du code CEU dans une zone mémoire accessible au processeur test,
(b) activation d'un signal d'interruption à des instants aléatoires,
(c) comparaison des résultats du programme à ceux de référence.

L'adjonction de ces fonctions à une carte digitale existante (cartes d'émulation du commerce disponibles pour la plupart de processeurs) n'est pas généralement une tâche complexe. Elle implique cependant, des modifications matérielles et/ou logicielles qui peuvent s'avérer délicates compte tenu des limitations dans l'espace d'adressage et des contraintes dans les protocoles d'échange de signaux.

Pour ces raisons, nous avons choisi de valider l'approche CEU à l'aide d'un testeur dédié, développé en collaboration avec le CNES. La flexibilité de ce système de test, appelé THESIC (*Testbed for Harsh Environment studies of Integrated Circuits*) [40], qui s'adapte a priori à tout processeur grâce à une carte d'interface adéquate (dite *carte fille*), et son architecture organisée en deux cartes (carte mère/carte fille) communiquant à travers une zone mémoire commune via un protocole simple qui permet la gestion des échanges de signaux asynchrones indispensables à la méthode proposée, ont été les principales raisons justifiant ce choix.

B. Le système de test THESIC

Des essais sous radiations sont nécessaires pour prédire le taux d'erreurs qu'un circuit subira une fois placé dans l'environnement de l'application spatiale. Ils consistent en l'exposition du circuit, placé en général dans une enceinte sous vide, à un faisceau issu par exemple d'un accélérateur de particules. Des développements matériels et logiciels pour la mise en œuvre de tels essais doivent prendre en compte que les événements doivent être détectés en ligne (instant d'occurrence aléatoire) et que certains parmi eux peuvent avoir des conséquences sérieuses (pertes de séquencement d'un processeur), voire destructives (latchup). Des dispositifs adéquats pour la détection et le recouvrement de ces événements (circuits anti - latchup et chien de garde), sont implantés dans des cartes de test.

La plupart des testeurs fonctionnels du commerce offrent des solutions puissantes pour la réalisation de tests de SEE (*Single Event Effects*) pour tout type de circuits [41-43]. Cependant, ils présentent deux désavantages importants :

- impossibilité d'être placés dans les enceintes sous vide généralement utilisées pour les essais avec sources de rayonnement. S'ils sont installés en dehors de ces enceintes, en les connectant au dispositif à tester, des problèmes délicats de propagation de signaux apparaissent.
- difficulté de définition des stimuli de test (ensemble de séquences binaires qui correspondent aux valeurs des pattes des circuits à chaque période d'horloge) dans le cas de circuits intégrés complexes.

Le système de test THESIC a été conçu pour surmonter ces inconvénients. Le dispositif sous test (DUT) est immergé dans un environnement digital « naturel », au lieu de recevoir des vecteurs de test qui miment l'activité sur ses entrées [43]. Lors de la conception de l'architecture du système THESIC les contraintes suivantes ont été considérées :

- Flexibilité : le système doit permettre une conception simple pour un circuit donné à tester, en réduisant le temps et le coût de développement nécessaires à l'évaluation de nouveaux dispositifs.
- Dimensions réduites : pour permettre de placer l'équipement de test dans les enceintes sous vide où sont effectués les essais.
- Contrôle en temps réel de l'activité du composant sous test.

L'architecture adoptée [44], donnée dans la figure 1.3, comporte :

- Une carte fille (carte DUT) qui a l'architecture nécessaire pour permettre la stimulation du DUT pendant l'irradiation.
- Une carte mère qui réalise les fonctions de contrôle de la carte fille.
- Un ordinateur qui sert d'interface avec l'utilisateur et permet la gestion du test en ligne et l'enregistrement des événements pour une analyse ultérieure.

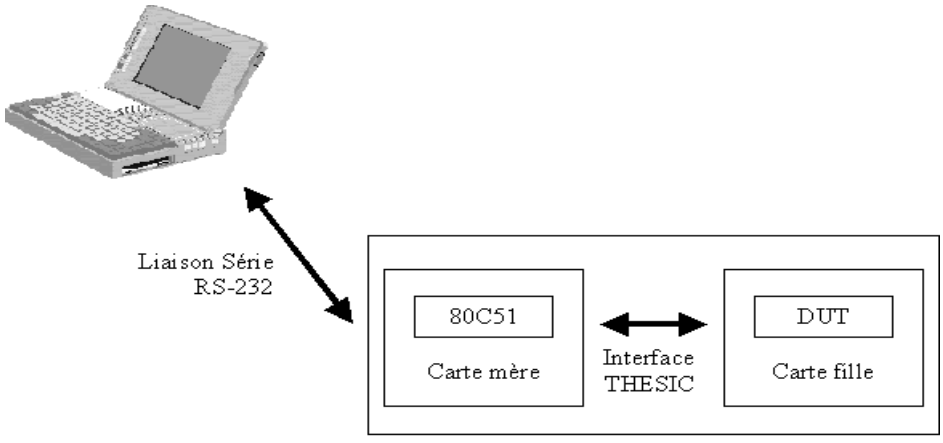


Figure 1.3 : Principe de base du testeur THESIC

1) Interface THESIC

Les signaux qui définissent l'interface entre la carte mère et la carte fille sont détaillés dans la table 1.3. La communication entre les deux cartes est réalisée de manière asynchrone grâce à une mémoire commune appelée MMI (*Memory Mapped Interface*). Il est intéressant de préciser la fonction de certains des signaux :

- BD_SEL* Il indique que la carte mère initie un accès de lecture ou d'écriture sur la MMI.
- RESET* Il sert pour initialiser le test sur la carte fille ou pour le réinitialiser lors d'une perte de séquence détectée sur le système sous test. Ce signal est normalement connecté au signal d'entrée Reset du processeur présent sur la carte DUT.
- INT* (signal d'interruption vers la carte mère) et INT_RESET* (un signal d'interruption sur la carte fille). Ils permettent l'établissement d'une communication asynchrone.
- RXD_DUT et TXD_DUT Ils permettent une communication série asynchrone entre les deux cartes.
- ADDR[10 : 0] Onze lignes d'adresse qui permettent l'adressage depuis la carte mère de 2Koctets de MMI.
- DATA[7 : 0] Huit lignes de données.
- Des alimentations et masses que la carte mère fournit à la carte fille. Ceci permet le contrôle de la consommation (détection de latchups) lors des essais de radiation.

Table 1.3 : Signaux de l'interface carte mère / carte fille THESIC

	A	**B**	**C**
1	Non assigné	Non assigné	Non assigné
2	Réservé	Non assigné	Non assigné
3	Réservé	Non assigné	Non assigné
4	ADDR 13	Non assigné	Non assigné
5	ADDR 12	Réservé	Non assigné
6	ADDR 11	Réservé	Non assigné
7	ANALOG 1	GND	D/A V
8	GND	GND	GND
9	GND	GND	GND
10	-15V	-15V	-15V
11	+15V	+15V	+15V
12	GND	GND	GND
13	GND	GND	GND
14	-5V	-5V	-5V
15	-5V	-5V	-5V
16	-5V	-5V	-5V
17	GND	GND	GND
18	GND	GND	GND
19	+5V	+5V	+5V
20	+5V	+5V	+5V
21	+5V	+5V	+5V
22	GND	GND	GND
23	GND	GND	GND
24	RXD_DUT	RESET*	TXD_DUT
25	BD_SEL	INT_RESET *	DATA 7
26	ADDR 6	INT*	DATA 6
27	ADDR 5	WR*	DATA 5
28	ADDR 4	RD*	DATA 4
29	ADDR 3	ADDR 10	DATA 3
30	ADDR 2	ADDR 9	DATA 2
31	ADDR 1	ADDR 8	DATA 1
32	ADDR 0	ADDR 7	DATA 0

2) Carte mère

La carte mère est organisée autour du microcontrôleur 80C51 et dispose de circuits nécessaires pour la communication avec la carte fille et un PC. Ses dimensions sont de 233 x 160 mm, taille qui lui permet d'être installée dans la plupart des enceintes utilisées pour faire les tests en environnement sévère. Ses alimentations sont $\pm 15V$ (tension régulée par la source parce que la carte ne dispose pas de régulateurs pour le faire) et +5V (tension régulée sur la carte mère). La communication avec le PC est réalisée via une liaison série RS-232 et l'interface avec la carte DUT est le bus THESIC. Les fonctions de la carte mère sont le contrôle de la carte fille : contrôle de latchup, démarrage - arrêt du test, téléchargement des programmes et données de test, réception et transmission de données de et vers la carte fille, ainsi que la réception et la transmission des données de et vers le PC d'interface avec l'utilisateur.

Le microcontrôleur 80C51 ne dispose pas de mémoire de programme interne. La figure 1.4 montre l'organisation du plan mémoire externe.

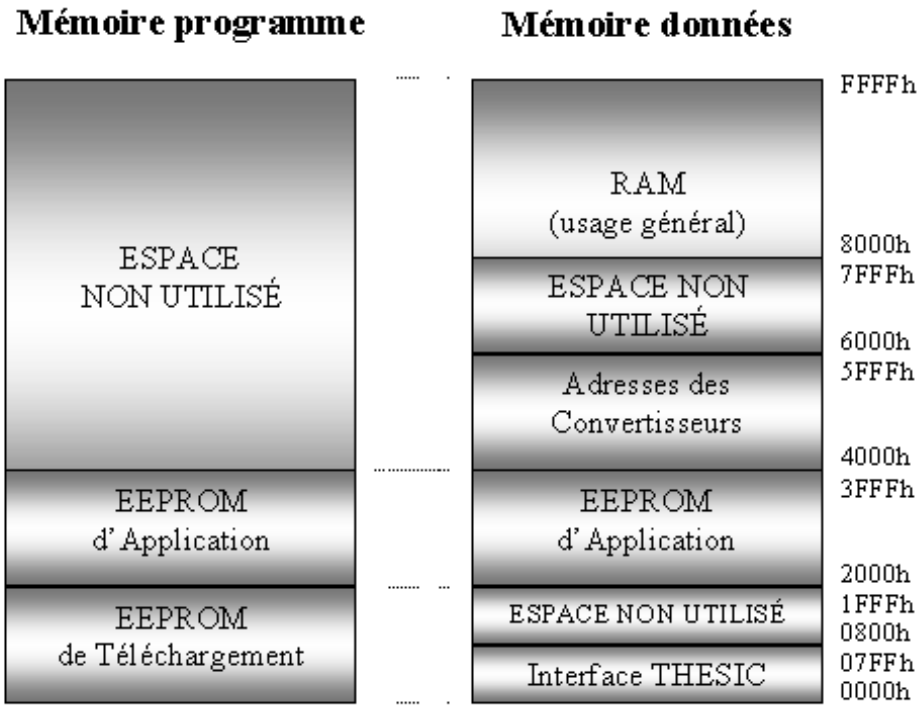


Figure 1.4 : Plan mémoire du 80C51

L'espace de mémoire programme externe est composée de deux mémoires EEPROM de 8K octets. L'EEPROM de Téléchargement occupe les adresses 0000h à 1FFFh et contient le logiciel dit de téléchargement, qui permet de reprogrammer l'autre EEPROM. L'EEPROM d'application occupe les adresses 2000h à 3FFFh et contient le logiciel principal gérant l'interface avec la carte fille.

Quant à la mémoire de données externe, elle se compose de quatre zones différentes :

- Les 2Koctets qui vont de 0000h à 07FFh sont réservés à la mémoire commune pour l'interface avec la carte fille. Cet espace peut être utilsé pour une mémoire physiquement localisée sur la carte fille ou bien pour l'accès direct à un dispositif placé dans la carte fille comme une PAL ou un DUT.
- Entre les adresses 2000h et 3FFFh, se trouve l'EEPROM d'application, accessible seulement en écriture. Ceci permet au programme de téléchargement de programmer cette mémoire avec le programme de contrôle de test à réaliser.
- Le convertisseur A/N de 12 bits, utilisé pour l'acquisition de la voie analogique provenant de la carte fille, occupe en lecture, l'adresse 4000h (8 bits de poids faibles) et 4001h (4 bits de poids forts). Le seuil de latchup programmé par l'utilisateur qui fournira le convertisseur N/A de 8 bits aux circuits de contrôle de latchup se trouve aussi à l'adresse 4000h en écriture.
- Les adresses 8000h à FFFFh sont utilisées pour le stockage des données du programme d'application et des fichiers à télécharger à la carte fille.

3) Carte fille

La carte fille permet l'irradiation d'un circuit, simulant le fonctionnement du DUT en environnement *naturel*, le plus proche possible de celui de l'application en vol. Cette carte doit avoir une taille physique lui permettant d'être installée dans les enceintes utilisées pour les tests. La communication avec la carte mère doit être réalisée en accord avec le protocole et les signaux de l'interface THESIC. On peut choisir entre trois architectures différentes selon la nature du composant à tester (Figures 1.5, 1.6 et 1.7) :

Test de mémoires

Dans ce cas, le DUT peut être placé dans l'espace d'adressage de la carte mère, le test étant directement assuré par celle-ci suite à l'écriture d'un programme adéquat. Le principal désavantage de cette architecture est la fréquence de travail, limitée à celle du microcontrôleur présent sur la carte fille. De plus, sachant que le bus d'adresses de THESIC comporte seulement 11 lignes d'adresses, les mémoires auxquelles une telle stratégie est appliquée pourront être testées très partiellement (un bloc de 2 K octets).

Test de microprocesseurs

Pour le test de circuits capables d'exécuter un jeu d'instructions ou de commandes (microprocesseurs, DSPs, etc.) nous adoptons l'architecture donnée dans la figure 1.6 où l'utilisation de la mémoire partagée (MMI) offerte par THESIC est indispensable.

Dans ce cas, la carte mère télécharge à la MMI le programme qui doit être exécuté par le composant pendant le test. Quand le microprocesseur reçoit le Reset depuis la carte mère, il exécute le programme de test téléchargé. La carte mère obtient les résultats du test en lisant la MMI, préalablement remplie par le DUT.

Test de circuits périphériques de microprocesseurs

Il s'agit de faire fonctionner le périphérique dans son environnement « naturel ». Le DUT travaille comme «esclave» d'un microprocesseur.

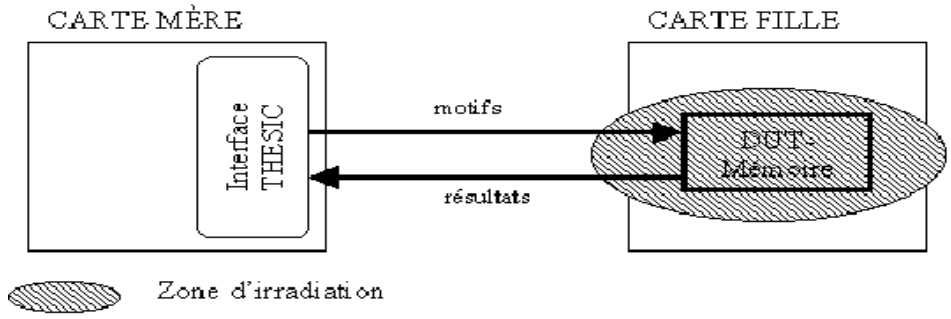


Figure 1.5 : Carte fille THESIC pour le test de circuits de mémoires

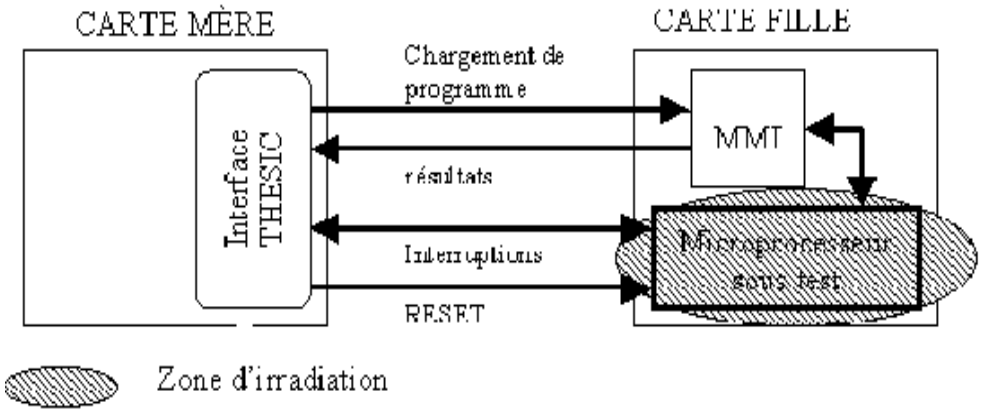


Figure 1.6 : Carte fille THESIC pour le test de circuits de type processeurs

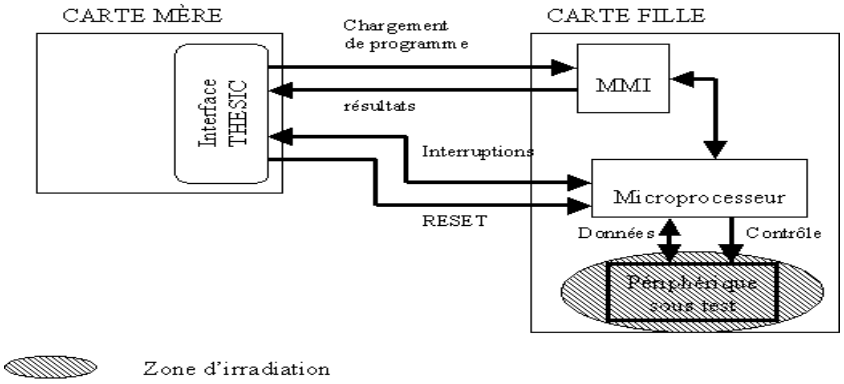


Figure 1.7 : Carte fille THESIC pour le test de circuits périphériques du processeur

La carte mère télécharge aussi un programme de test qui sera exécuté par le microprocesseur présent sur la carte fille, qui communiquera via la MMI les événements détectés sur le DUT lors de l'exécution de ce programme. La figure 1.7 donne le schéma du principe de cette architecture.

Il faut noter que dans le cas où le circuit sous test est un processeur, le mode (b) présente quelques limitations de diagnostic. En effet les erreurs affectant le séquencement d'un programme peuvent avoir des conséquences difficiles à prévoir ou à comprendre à partir de l'analyse des données corrompues. A titre d'exemple, un SEU affectant le compteur programme ou le code des instructions, peut conduire à un « black out » au niveau de la carte mère (laquelle durant le test est conçue pour attendre des interruptions de la carte fille lui indiquant que des erreurs ont été détectées). Pour faire face à ce type d'erreur critique, un « chien de garde » logiciel programmable a été implanté sur la carte mère.

Les capacités du système THESIC ont été largement prouvées durant plusieurs campagnes d'essais sous radiations réalisées ces quatre dernières années. La versatilité de ce testeur a été mise en évidence par la diversité des composants testés. A titre d'exemple, peuvent être mentionnés des microprocesseurs et des micro-contrôleurs à usage général (le transputer T225 de INMOS, le 80C51 d'Intel), des co-processeurs dédiés (circuit neuronal LNEURO 1.0 de Philips, contrôleur fuzzy WARP 2.0 de SGS-Thomson) et des SRAM's de 64 KOctets (de Hitachi et de Matra Harris Semiconductors). Des détails sur le développement de ces cartes filles THESIC réalisées et utilisées sous radiations peuvent être trouvés dans [46-47].

La photo dans la figure 1.8 montre l'ensemble carte mère /carte fille THESIC dans la chambre sous vide du cyclotron 68" du LBL [48]. La carte mère, montrée à l'arrière-plan, est fixée à un support mobile qui permet de réaliser l'alignement entre le DUT et le faisceau. La communication vers un PC externe est réalisée à travers d'une connexion série. La carte fille, montrée au premier plan, est destinée à évaluer le comportement sous radiation d'une architecture conçue autour d'un Transputer, un co-processeur neuronal et différentes SRAM's. Ce système constitue le modèle d'une expérience, actuellement embarquée à bord d'un satellite scientifique, opérationnelle en orbite depuis la fin 1997 [49].

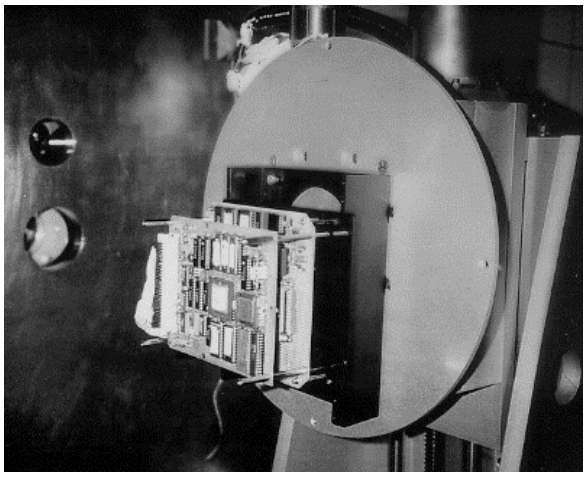


Figure 1.8 : THESIC à l'intérieur de la chambre sous vide du cyclotron du LBL (Berkeley-California)

C. Implantation de la technique d'injection de CEU à l'aide du testeur THESIC

Rappelons que les étapes principales d'injection de CEU impliquent (§ III. C) :

(a) l'élaboration du code CEU,

(b) l'enregistrement du code CEU dans une zone mémoire adéquate,

(c) l'activation d'une interruption,

(d) la saisie des résultats et leur comparaison à des valeurs de référence.

L'interface THESIC offre la possibilité d'utiliser différents signaux asynchrones pour dialoguer avec la carte fille, l'un d'entre eux, INT_RESET, inutilisé dans la configuration actuelle, a été choisi pour être utilisé pour activer l'interruption qui provoque l'exécution du code CEU. Cette activation peut être déclenchée par le passage à 0 d'un compteur initialisée par un programme simple à une valeur aléatoire. Le microcontrôleur 80C51 au cœur de la carte mère dispose de registres Timers programmables tout à fait adéquats pour ce but.

Pour réaliser les étapes (a) (b) (c) et (d), un programme spécial a été écrit en assembleur du microcontrôleur 80C51 de la carte mère THESIC. Le code CEU et les résultats de l'exécution du programme testé sont enregistrés dans la mémoire MMI, accessible à la carte mère et à la carte fille et qui est donc parfaitement adaptée à cette fonction

THESIC offre donc une plate-forme appropriée pour l'injection de CEU, sans modifications ni développements matériels supplémentaires. Seulement il y a certains inconvénients à mentioner, tels que la fréquence de fonctionnement du timer du microcontrôleur 80C51 (responsable du comptage du nombre de cycles de l'application

étudiée) qui est limitée à 1 MHz. En effet, cette limitation nous restreint à injecter au maximum un CEU par microseconde. Pour des processeurs très rapides, ceci peut constituer une sérieuse limitation. D'un autre côté, sachant que le 80C51 est un microcontrôleur 8 bits, le calcul de nombres aléatoires peut être difficile à implanter lorsque le nombre de cibles est grand. Tout de même, ces deux limitations peuvent être évitées par le remplacement du microcontrôleur 80C51 de la carte mère par un autre processeur ayant un jeu d'instructions plus riche et une fréquence de fonctionnement élevée. Mis à part ces deux restrictions, la seule ressource logicielle manquante dans le système THESIC actuel est la génération de nombres aléatoires, objet du paragraphe suivant.

D. Algorithme utilisé pour la génération de nombres pseudo-aléatoires

Comme vu précédemment l'une des tâches à réaliser pour implanter la méthode d'injection d'erreurs est celle liée à la génération de nombres aléatoires qui seront utilisés dans le programme d'injection, pour le choix de la cible à modifier et de l'instant de l'injection de l'erreur.

Plusieurs algorithmes existent pour produire des nombres pseudo-aléatoires. Ils se différencient par leur vitesse, leur complexité et la qualité (« aléatoirieté ») des résultats qu'ils produisent. Nous avons utilisé une stratégie appelée *méthode linéaire de congruence* (introduite par D. Lehmer en 1951) [50]. Cet algorithme, qui produit une suite de nombres sans aucune répétition à partir d'une valeur initiale appelée « graine » (SEED), est défini par la formule (1.4) suivante :

$$Rndnum(n) = (Rndnum(n-1) * MULT + INC) \bmod [M] \qquad (1.4)$$

Où : Rndnum(n) = le nombre aléatoire actuel

Rndnum(n-1) = le nombre aléatoire précédent

MULT = multiplicateur (constante à définir)

INC = incrément (constante à définir)

M = module

Le choix des paramètres MULT et INC est crucial pour la qualité de la distribution des nombres générés. Nous nous sommes basés sur des études préalables pour réaliser ce choix. Dans la suite, nous illustrons le choix de constantes effectué pour générer des adresses aléatoires pour injecter des fautes dans une mémoire de 64 K. Une mémoire de cette taille constitue la mémoire interne du processeur pour le traitement de signaux traité dans le chapitre suivant.

- M : étant un module, il définit l'étendue des nombres générés. L'algorithme renverra donc un nombre compris entre 0 et M-1. Dans l'exemple de la mémoire 64 K, M doit être 65536 pour garantir que toutes les adresses soient parcourues.
- SEED : Ce premier nombre aléatoire dans la séquence est une constante arbitraire entre 0 et M-1.
- MULT : Basé sur la théorie des nombres aléatoires, ce nombre devrait être tel que ses trois derniers chiffres soient une paire-2-1 (tels que xx821, x421, etc.). Le numéro 31821 a été choisi dans le cas de la mémoire de 64 K mots de 16 bits.

Après avoir effectuer un tour complet, l'algorithme répète l'un des nombres déjà générés, entrant donc en boucle. Pour éviter les problèmes que ceci entraînerait au niveau de l'injection de CEU, un changement de SEED est réalisé de manière systématique.

Un effort particulier [51], a été investi dans le développement d'une interface utilisateur conviviale, susceptible de fournir à l'opérateur des résultats en ligne sur l'évolution du test. Cette interface a été développée et mise au point pour permettre l'automatisation d'expériences d'injection d'upsets à l'aide du testeur THESIC. Deux aspects importants et originaux de cette interface utilisateur sont : (a) la visualisation en ligne d'indicateurs représentatifs de la sensibilité du DUT à l'environnement et (b) la possibilité de choisir la «qualité» de la session d'injection de CEU : aléatoire ou exhaustive (Figure 1.9). Dans une session d'injection de CEU, on peut fixer notre choix du bit ou la position mémoire ou les deux à la fois. De même, pour l'instant au moment où on voudrait injecter le CEU, le choix peut être aléatoire ou déterministe (à partir du cycle d'horloge initial jusqu'au nombre maximum de cycles d'horloge que nécessite l'exécution entière de l'application étudiée). Des détails supplémentaires à propos de cette interface sont donnés en annexe 1.

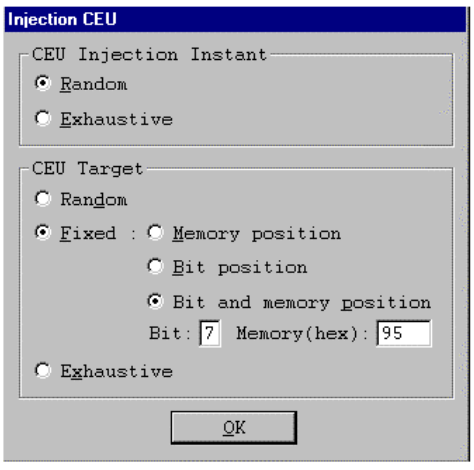


Figure 1.9 : Interface utilisateur THESIC pour l'injection de CEU

V. PREDICTION DE LA SECTION EFFICACE D'UNE APPLICATION

Comme les basculements de bits injectés sont aléatoires dans la position et l'instant d'occurrence, cette méthode peut être utilisée pour prédire la section efficace d'une application complexe. En effet, l'hypothèse faite à la base de cette thèse est qu'à partir du taux d'erreurs moyen dû aux CEU et de la section efficace de SEU d'un test statique, peut être prédit le taux d'erreurs des SEU dus aux radiations. L'idée est la suivante, sachant qu'on utilise la nomination # pour désigner le nombre :

- à partir du test statique peut être déterminé (formule 1.5), le nombre moyen de particules nécessaires pour provoquer un upset dans une cible donnée (section efficace face aux SEU, σ_{SEU}).

$$\sigma_{SEU} = \frac{\#Erreurs}{Fluence\ de\ particules} \qquad (1.5)$$

- à partir des essais d'injection de CEU (formule 1.6), on peut déterminer le nombre moyen de CEU (taux d'erreurs face aux CEU, τ_{CEU}) nécessaires pour provoquer une erreur dans le programme.

$$\tau_{CEU} = \frac{\#Erreurs}{\#CEUs\ injectés} \qquad (1.6)$$

- Le produit de la section efficace et du taux de CEU donne la contribution de la cible considérée à la section efficace globale du programme considéré dû aux radiations.
- La somme de toutes les contributions constitue un estimateur de la section efficace de l'application τ_{SEU} sous radiations.

  Cette démarche est illustrée par la formule (1.7):

$$\tau_{SEU} = \sum_i \tau_{SEU}^i(R_i) = \sum_i \sigma_{SEU}(R_i) * \tau_{CEU}(R_i) \qquad (1.7)$$

Où :

- Ri désigne la cible de l'upset (mémoire ou registre).
- τ_{SEU}^i : Section efficace du registre Ri durant l'exécution du programme étudié

Comme l'ensemble des cibles des SEU n'est pas en général complètement connu et accessible à l'utilisateur, l'équation (1.7) ne donne qu'une estimation de la section efficace de l'application. L'erreur commise dépendra de l'ampleur de la zone sensible inaccessible.

A. Estimation de la section efficace d'une application par calcul des faceturs de sensibilité

L'un des avantages de la méthode d'injection de CEU consiste en la possibilité d'injection de fautes dans une cible donnée de manière exhaustive dans les instants d'occurrence. En effet, en choisissant une cible donnée dans la zone sensible du circuit étudié et en injectant un CEU dans cette position à chaque cycle d'horloge du 80C51 (de la carte mère), nous pouvons déterminer le facteur de sensibilité de la cible durant l'application étudiée. Ainsi, en se basant sur la formule (1.7), on peut déduire le calcul de la section efficace d'une application à partir du produit des sections efficaces individuelles pondérées par les facteurs de sensibilité correspondants, calculées à partir des résultats d'essais d'injection de fautes exhaustifs (formule 1.8) :

$$\tau_{SEU} = \sum_i \sigma_{SEU}^i(R_i) * f_{CEU}^i(R_i) \quad (1.8)$$

Où :

- σ_{SEU}^i : La section efficace individuelle aux SEU du registre Ri
- f_{CEU}^i : Fecteur de sensibilité d'un registre Ri

B. Estimation de la section efficace d'une application par injection aléatoire

Si on suppose que toutes les cibles (positions de mémoire interne et registres) ont la même section efficace individuelle, et si on injecte des CEU aléatoirement à des proportions égales dans toutes les zones du processeur étudié, la formule (1.7) peut être simplifiée et devient :

$$\tau_{SEU} = \sigma_{SEU} * \tau_{CEU} \quad (1.9)$$

Où :

- σ_{SEU} : La section efficace aux SEU du composant
- τ_{CEU} : Taux d'erreurs global

Ces deux approches d'estimation de la section efficace d'une application, soit par injection aléatoire dans toutes les cibles du processeur soit par calcul des facteurs de sensibilités, seront appliquées à différents processeurs afin de tester leurs validités dans la prédiction de la sensibilité aux SEU d'une application donnée.

VI. CONCLUSION

Dans ce chapitre, nous avons présenté une approche d'injection de fautes de type upset, permettant d'étudier les effets de SEU sur le fonctionnement de microprocesseurs. La stratégie proposée est basée sur des expériences d'injection logicielle de fautes, dans un bit du processeur et à un instant d'occurrence choisis aléatoirement afin de s'approcher des SEU survenant en environnement radiatif. Les upsets sont injectés durant l'exécution du programme par activation d'une interruption externe à l'aide d'un système dédié, le testeur THESIC, développé au laboratoire TIMA. Cet outil compact et flexible, a été conçu pour des tests de qualification (radiation, température, E.M.C) des circuits intégrés complexes.

Du fait de *l'aléatoireté* des paramètres d'injection de CEU, cette technique servira non seulement pour l'étude du comportement de processeurs en environnement radiatif mais aussi pour la prédiction de la section efficace d'une application quelconque exécutée sur le processeur étudié. En effet, l'objectif principal est qu'à partir du taux d'erreurs prédit suite à des sessions d'injection de CEU dans un processeur et de la section efficace aux SEU de ce composant, la sensibilité de cette application pourrait être déduite, et ceci sans devoir l'exposer aux radiations.

Afin de prouver la validité de la méthode d'injection de CEU et d'évaluer la précision de la prédiction du taux d'erreurs qui en découle, nous allons appliquer cette technique à différentes architectures digitales bâties autour de divers processeurs (le microcontrôleur 80C51 d'Intel, les processeurs de traitement de signal, le TMS320C50 de Texas Instruments, le ADSP21060 d'Analog Devices et le microprocesseur TS6833216A). Les résultats expérimentaux de ces expériences d'injection de fautes ainsi que les taux d'erreurs prédits pour les applications étudiées seront présentés dans les chapitres suivants.

CHAPITRE 2

INJECTION DE CEU ET PREDICTION DU TAUX D'ERREURS POUR UNE ARCHITECTURE DIGITALE BATIE AUTOUR DU 80C51

I. INTRODUCTION

Nous avons choisi le microcontrôleur d'Intel 80C51 comme véhicule d'expérimentation pour valider la technique d'injection de CEU, mettre en évidence ses limitations éventuelles et explorer les possibilités de généralisation et d'automatisation de cette approche. En effet, la disponibilité d'abondants résultats de tests sous radiations de ce circuit, qui fait partie de plusieurs instruments embarqués dans des projets des agences spatiales européennes et américaines, devrait permettre à terme, de confronter les prédictions du taux d'erreursfaites à partir des expériences d'injection de CEU sur des modules de programmes de ces instruments à des mesures effectuées en vol.

Après une description des grandes lignes de l'implantation à l'aide du testeur THESIC de la méthode CEU pour ce microcontrôleur, ce chapitre sera consacré à l'analyse des résultats expérimentaux issus de son application au 80C51 lors de l'exécution de deux programmes simples. Des essais réalisés avec un choix aléatoire des paramètres du CEU (cible, instant d'occurrence) permettront l'estimation de la sensibilité aux SEU pour les différentes applications. Des essais avec sélection déterministe de ces paramètres, mettront en évidence deux nouvelles potentialités de la méthode : possibilités d'analyse détaillée des conséquences des CEU sur le fonctionnement d'un programme donné et détermination automatique des facteurs de sensibilité des cibles de CEU. Des essais d'injection de CEU pseudo-aléatoires permettront l'estimation des taux d'erreurs des programmes considérés. Pour évaluer la précision de la prédiction du taux d'erreurs, des campagnes de radiation sont nécessaires. Les détails et les résultats de ces campagnes seront présentés à la fin de ce chapitre.

Une carte fille THESIC a été développée autour du 80C51 pour réaliser des essais d'injection de fautes et des tests aux ions lourds. Les détails de cette architecture n'étant pas nécessaires à la compréhension des résultats présentés ici, ils ont été reportés dans l'Annexe 2.1.

Les différents codes CEU contenant des paramètres correspondant à la position mémoire et au bit cibles, sont stockés dans la mémoire externe de la carte mère du testeur THESIC. La détermination de ces codes passe par l'identification des cibles de CEU et leur classification en familles selon le mode d'adressage nécessaire à l'accès (lecture/écriture) à leur contenu. Cette étape est issue de l'analyse des caractéristiques du circuit à tester : architecture interne et modes d'adressage. L'étude sera limitée à l'injection de CEU sur des cibles internes au 80C51. Son extension à des cibles externes (pour étudier les conséquences des upsets sur le code d'un programme par exemple) ne soulève pas de difficultés majeures. L'occurrence du CEU résultera du passage à 0 d'un timer de la carte mère THESIC préalablement chargé à une valeur aléatoire.

II. CIBLES DE CEU DANS LE 80C51

Les principales caractéristiques de la famille de contrôleurs MCS 51 sont les suivantes [52] :

- CPU de 8 bits optimisé pour le contrôle d'applications
- Processeur booléen permettant le calcul sur un bit
- 64k octets d'espace mémoire programme
- 64k octets d'espace mémoire de données
- Mémoire programme intégrée d'une taille supérieure à 32 KOctets
- Mémoire interne RAM de 128 octets
- Lignes d'entrées/sorties individuelles et bidirectionnelles
- Communication bidirectionnelle UART

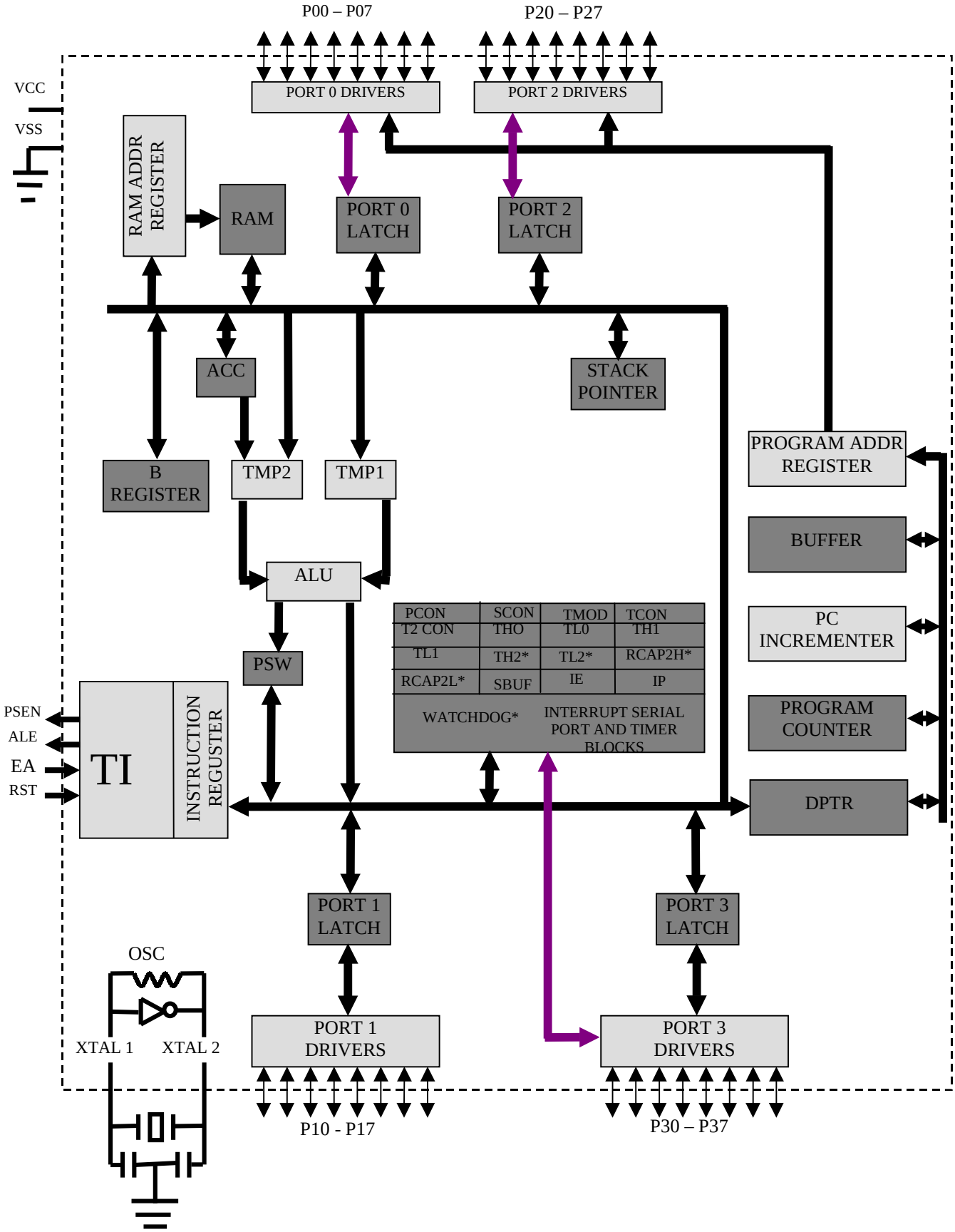


Figure 2.1: Architecture Interne du 80C51

Dans la figure 2.1 sont donnés les principaux blocs de l'architecture interne du 8051. Pour déterminer l'ensemble de cibles des CEU parmi ces blocs, il suffit d'identifier ces zones mémoires qui sont directement accessibles par le jeu d'instructions. Dans le cas du 80C51, ces zones (en gris foncé dans le schéma de la fig. 2.1) comprennent essentiellement les ports d'entrée / sortie (Port 0 - Port 3), les accumulateurs A et B, les registres spéciaux (SFRs), le registre compteur de programme (PC) et la mémoire RAM interne (128 octets). Les registres SFRs sont implantés dans la mémoire interne dans laquelle ils sont accessibles par adressage direct. La figure 2.2 donne l'organisation de cette partie de la mémoire interne.

8 Octets

F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8									CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL					PCON	87

Bit accessible

Figure 2.2 : Plan mémoire des SFRs du 80C51

Dans le schéma de la figure 2.1 sont également indiquées (en gris clair) les cibles potentielles de SEU qui ne pourront pas être perturbées par des CEU. Elles incluent des registres d'entrée de l'U.A.L., des latches, des registres d'adresses, etc. L'évaluation exacte du nombre d'éléments mémoire inaccessibles aux CEU n'est pas à la portée d'un usager. Dans la figure 2.3 est indiquée à titre indicatif la proportion de cette zone (évaluée à 7%) par rapport à la zone globale accessible aux CEU.

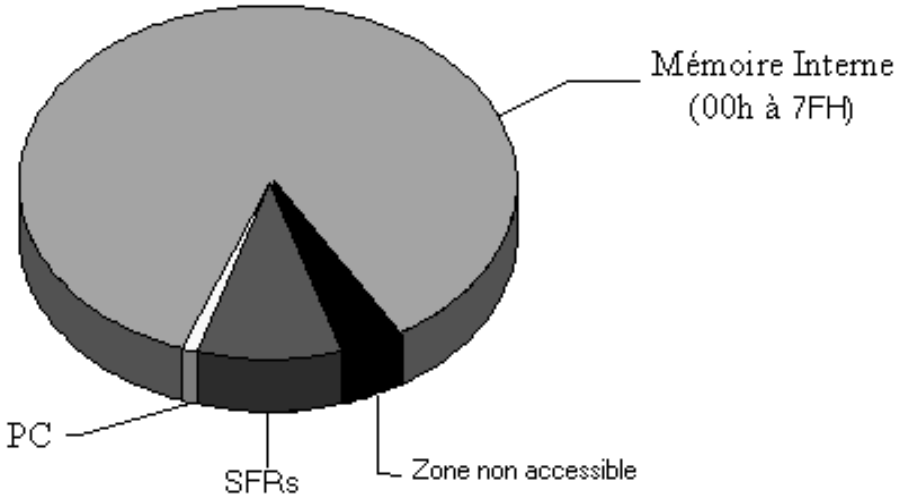


Figure 2.3 : Cibles des CEU du 80C51

III. STRATEGIE ADOPTEE POUR LA DETERMINATION DES CODES CEU

L'analyse de la nature de l'accès ainsi que des fonctionnalités des éléments mémoires détermine des familles de cibles de CEU, auxquelles seront associés des codes CEU génériques. Ces codes comporteront des parties variables dont l'affectation, préalable à l'injection du CEU, permettra de déterminer le(s) bit(s) d'un registre ou d'un mot mémoire à perturber. Dans le cas du microcontrôleur 80C51, la taille de la zone accessible aux CEU comprend 151 octets, qui peuvent se répartir comme suit :

- L'accumulateur A, dont l'adressage est implicite.
- Les registres spéciaux (SFRs) autres que SP, situés dans la partie supérieure de la mémoire interne (adresses hautes), mais accessibles par adressage direct.
- 128 positions mémoires, accessibles par adressage direct via l'accumulateur.
- Le pointeur de pile, toujours pointant vers la mémoire interne (1 octet).
- Les 2 octets du compteur programme (PC).

Une fois les familles de cibles CEU déterminées, un code CEU doit être associé à chacune d'entre elles. L'exemple le plus simple est celui de l'accumulateur, pour qui le code CEU se limite à deux instructions : l'instruction OU exclusif entre la valeur du registre et celle indiquant le bit à modifier (notée PosBit par la suite) suivie de l'instruction de retour d'interruption (RTI).

Le but étant de corrompre exclusivement la cible choisie, une attention spéciale doit être portée à la sauvegarde du contenu de tout élément mémoire différent de la cible qui doit être utilisé au sein du code CEU. Dans le cas du 80C51 ceci concerne typiquement l'accumulateur, utilisé pour l'adressage de la mémoire interne, et le registre R0, indispensable pour l'injection de CEU dans les registres de contrôle SP et PC. Les tables 2.1, 2.2 et 2.3 recensent les cibles des CEU, donnant pour chacune d'entre elles le code CEU associé dans un exemple précis.

Table 2.1 : Codes CEU associés à l'Accumulateur, la RAM interne et les SFRs

Cible du CEU	Code CEU (assembleur 8051)		Code CEU (hexadécimal)	Commentaires
Accumulateur			*;si ZMI = 80h* *;si PosBit = 02*	
	xrl	*a,* ***PosBit***	*65* ***02***	Changement d'un bit dans l' ACC
	rti		*32*	Retour au programme principal
Mémoire Interne SFRs sauf (acc, sp)			*; si PosBit = 08h* *; si ZMI = 6Fh*	
	push	*acc*	*C0 E0*	Sauvegarde de ACC
	mov	*a,* ***Zmi***	*E5* ***6F***	Chargement du contenu de la cible dans l'ACC
	xrl	*a,* ***PosBit***	*65* ***08***	Changement d'un bit dans la cible
	mov	***Zmi****, a*	*F5* ***6F***	Chargement du contenu de l'ACC dans la cible
	pop	*acc*	*D0 E0*	Reprise de l'ACC
	rti;		*32*	Retour au programme principal

- ***ZMI (Zone Mémoire Interne)*** *désigne l'adresse d'une cible dans la mémoire interne,*
- ***Posbit*** *désigne une constante comportant un 1 (à la position correspondante au bit à perturber) parmi des 0*

Dans le cas de la mémoire interne, il faut remarquer la nécessité d'utiliser l'accumulateur A comme registre d'adresse. Son contenu doit être donc sauvegardé (dans la pile par exemple) pour préserver son état. Notez que la syntaxe de l'assembleur 80C51 désigne par « a » ce registre dans toute instruction sauf dans le cas des instructions push et pop (où il est désigné par « acc »). Compte tenu de leurs fonctions, le compteur programme et le pointeur de pile constituent des cas particuliers pour la détermination des codes CEU.

Dans les cas où l'emplacement de la pile est connu et accessible à l'utilisateur, la méthode décrite dans la table 1.1-c du § III-C du chapitre 1 peut être adoptée pour injecter des CEU dans PC : OU exclusif de la valeur de PC stockée dans la pile avec la valeur correspondante au bit à modifier. Dans le cas du 80C51, SP pointe vers le sommet de la pile où se trouve l'octet des poids faibles de PC. Le code CEU pour les deux octets de PC, commencera donc par le décrément de SP (table 2.2).

Table 2.2 : Code CEU du compteur programme PC

Cible du CEU	Code CEU (assembleur 8051)		Code CEU (hexadécimal)		Commentaires
Compteur programme PC Low	*; si PosBit = 20h* *; si Zmi = 96h*				
	push	*0*	*C0*	*00*	Sauvegarde de R0
	push	*acc*	*C0*	*E0*	Sauvegarde de l'ACC
	mov	*r0, sp*	*F8*	*81*	Pointage vers l'endroit où est stocké PCL
	mov	*a, @r0*	*E6*		Chargement du contenu de PCL dans l'ACC
	xrl	*a, PosBit*	*65*	*20*	Changement d'un bit dans l'ACC (PCL)
	mov	*@r0, a*	*F6*		Chargement du contenu de l'ACC dans PCL
	pop	*acc*	*D0*	*E0*	Reprise de l'ACC
	pop	*0*	*D0*	*00*	Reprise de R0
	rti		*32*		Retour au programme principal
Compteur programme PC High	*; si PosBit = 08h* *; si Zmi = 95h*				
	push	*0*	*C0*	*00*	Sauvegarde de R0
	push	*acc*	*C0*	*E0*	Sauvegarde de l'ACC
	dec	*sp*	*15*	*81*	Décrémentation de SP
	mov	*r0, sp*	*F8*	*81*	Pointage vers l'endroit où est stocké PCH
	mov	*a, @r0*	*E6*		Chargement du contenu de PCH dans l'ACC
	xrl	*a, PosBit*	*65*	***08***	Changement d'un bit dans l'ACC (PCH)
	mov	*@r0, a*	*F6*		Chargement du contenu de l'ACC dans PCH
	inc	*sp*	*05*	*81*	Inrémentation de SP
	pop	*acc*	*D0*	*E0*	Reprise de l'ACC
	pop	*0*	*D0*	*00*	Reprise de R0
	rti		*32*		Retour au programme principal

Enfin, le cas du pointeur de pile (SP) doit être analysé avec attention. En effet les fautes survenant dans SP ont des chances d'être critiques pour tout programme réalisant des opérations avec une pile (appel de procédures, traitement d'interruptions,…). La méthode d'injection de CEU étant basée sur des interruptions, la modification du registre SP utilisé pour restaurer le contexte du programme (adresse de retour) à la fin du traitement de l'interruption doit être effectuée en prenant des précautions particulières. Nous avons adopté la solution « élégante » et précédemment présentée dans la table 1.1-d du § III-C du chapitre 1 consistant à composer à une adresse particulière, une instruction de saut inconditionnel à l'adresse de retour au programme interrompu récupérée à partir de la pile avant de perturber SP.

En effet, avant de perturber le registre SP, l'adresse de retour au programme d'application du DUT (stockée dans les adresses supérieures de la pile) est sauvegardée dans des adresses prédéfinies de la MMI (0C751H et 0C752H). A l'adresse (0C750H) on écrit le code de l'instruction JMP (02H), de telle façon qu'on fasse, après inversion d'un bit du registre SP, un saut à l'adresse (0C750H), où on retournera systématiquement par exécution de l'instruction du saut, à l'adresse où le programme a été interrompu. Le code lui correspondant est donné dans la table 2.3.

Table 2.3 : Code CEU du pointeur de pile SP

Cible du CEU	Code CEU (assembleur 8051)		Code CEU (hexadécimal)			Commentaires
Pointeur de pile SP	*; si PosBitAl = 04h*					
	push	*0*	*C0*	*00*		Sauvegarde de R0
	push	*acc*	*C0*	*E0*		Sauvegarde de l'ACC
	push	*dpl*	*C0*	*82*		Sauvegarde de DPL
	push	*dph*	*C0*	*83*		Sauvegarde de DPH
	mov	*dptr, 0C750H*	*90*	*C7*	*50*	Sauvegarde du code de JMP dans 0C750H
	mov	*a, #02H*	*74*	*02*		
	movx	*@dptr, a*	*F0*			
	mov	*dptr, 0C752H*	*90*	*C7*	*52*	Sauvegarde de PCH dans 0C752H
	mov	*r0, sp*	*F8*	*81*		
	mov	*a, @r0*	*E6*			
	movx	*@dptr, a*	*F0*			
	mov	*dptr, 0C751H*	*90*	*C7*	*51*	Sauvegarde de PCL dans 0C751H
	dec	*sp*	*15*	*81*		
	mov	*r0, sp*	*F8*	*81*		
	mov	*a, @r0*	*E6*			
	movx	*@dptr, a*	*F0*			
	inc	*sp*	*05*	*81*		
	mov	*a, sp*	*E5*	*81*		Changement d'un bit dans sp
	xrl	*a, PosBitA*	*65*	***04***		
	mov	*sp, a*	*F5*	*81*		
	pop	*dph*	*D0*	*00*		Reprise de DPH
	pop	*dpl*	*D0*	*E0*		Reprise de DPL
	pop	*acc*	*D0*	*82*		Reprise de l'ACC
	pop	*0*	*D0*	*83*		Reprise de R0
	jmp	*0C750H*	*02*	*C7*	*50*	Saut vers adresse sauvegarde -1

Des tables 2.1, 2.2 et 2.3, il peut être constaté que la taille des codes CEU va de 3 octets (dans le cas de l'accumulateur) à quelques dizaines (49 octets) dans le cas du SP. Sachant qu'un cycle machine dans la carte 80C51 utilisé pour faire ces essais est d'une microseconde, la durée de l'injection du CEU s'étale donc entre 3 et 49 microsecondes. A ce temps, doivent être ajoutés les temps de préparation du code CEU et de son transfert dans la MMI. Ces deux opérations sont réalisées par la carte mère et dépendent donc de la fréquence de fonctionnement du microprocesseur de celle-ci. Enfin, doit être également compté le temps d'exécution du programme d'application lui-même.

A titre indicatif, dans l'état actuel de THESIC pour la carte fille 80C51, l'injection de CEU peut être réalisée à une fréquence d'environ 1 par seconde. Dans le cadre de cette étude nous n'avons pas cherché à optimiser cette fréquence d'injection, conservant la carte mère dans son état. Il est clair que des améliorations de la carte mère (en utilisant un microprocesseur à haute fréquence de fonctionnement) permettraient de réduire considérablement le temps d'injection de CEU.

IV. MISE EN ŒUVRE DE L'INJECTION DE CEU

La procédure d'injection d'erreurs dans le 80C51 va s'exécuter en grande partie dans la carte mère. La chronologie des différentes étapes dans l'injection de CEU, numérotées et réparties entre la carte mère et la carte fille est donnée dans les figures 2.4 et 2.5. Les étapes qui se déroulent à cheval entre carte mère et carte fille tel que l'envoi du signal INT_RESET sont représentées en trois parties : programme de test du DUT, Interface THESIC et programme de DUT.

Côté carte mère

- **Etape n°1 :** Calcul du temps d'exécution de l'application exécutée par le DUT placé sur la carte fille comptage dans un Timer du nombre de cycles machine se déroulant entre l'envoi du signal Reset (lancement de l'application) et la réception du signal INT (indiquant la fin du programme).
- **Etape n°2 :** Détermination des paramètres du code CEU (remplacement des données en gras dans les tables 2.1 et 2.2),

- une cible (position mémoire ou registre),
- un bit de cette position mémoire,
- un instant d'injection.

- **Etape n°3 :** Placement du code CEU dans la MMI afin qu'il puisse être exécuté par l'application implantée dans le 80C51,
- **Etape n°4 :** Lancement de l'application sur la carte fille (envoi du signal Reset) et du décomptage du Timer,
- **Etape n°5 :** Activation du signal INT_RESET,
- **Etape n°10 :** Récupération, traitement et affichage des résultats.

Côté carte fille

- **Etape n°4 :** Réception de l'ordre d'exécuter le programme d'application (réception du signal RESET),
- **Etape n°5 :** Réception de l'ordre d'interruption INT_RESET : la carte fille arrête son programme d'application et pointe vers le code CEU,
- **Etape n°6 :** Exécution du code CEU implanté dans la MMI,
- **Etape n°7 :** Poursuite du programme d'application jusqu'à la fin,
- **Etape n°8 :** Comparaison des résultats obtenus à ceux attendus,
- **Etape n°9 :** Envoi du signal INT pour indiquer la fin du programme.

Figure 2.4 : Etapes de la procédure d'injection d'erreurs

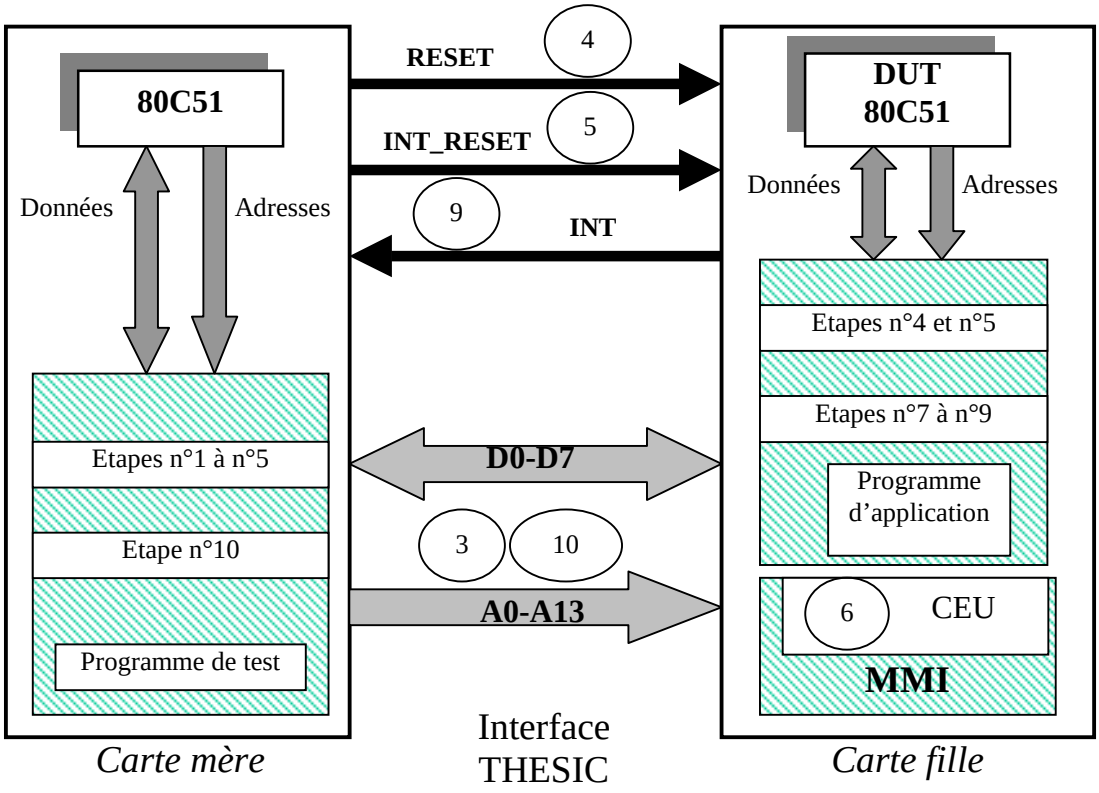


Figure 2.5 : Schéma de fonctionnement de la procédure d'injection d'erreurs

V. EXEMPLE D'ILLUSTRATION

Les programmes correspondant aux étapes décrites dans le §IV se déroulant sur la carte mère ont été développés en assembleur 80C51. Nous illustrerons dans un premier temps les résultats obtenus lors de leur utilisation pour injecter des CEU durant l'exécution d'un programme simple. Les résultats de ces essais d'injection de CEU ont été présentés dans [53] et [54].

A. Programme d'application étudié

Ce programme effectue le remplissage d'une zone de 256 octets de la mémoire externe de la carte fille avec un motif donné. Dans la suite, nous détaillerons quelques cas typiques d'injection de CEU sur deux cibles particulières : l'accumulateur A et le compteur programme PC. L'organigramme du programme est donné dans la figure 2.6. Il consiste essentiellement en une boucle contrôlée par un registre, donc son implantation dans le 80C51 pourra être faite avec seulement une douzaine d'instructions (table 2.4) avec pour cibles des CEU les registres suivants : l'Accumulateur, les Ports, le registre R0, les registres DPTR (DPL et DPH) et le compteur programme PC.

Le but de ce premier essai d'injection de fautes est d'une part l'estimation de la sensibilité face aux CEU d'un programme utilisant un nombre minimal de ressources, d'autre part la description des mécanismes permettant d'expliquer la tolérance face à certaines fautes a priori critiques. Enfin, nous tenterons de mettre en évidence le bien

fondé de l'approche et de son implantation, par le biais de l'analyse du flot de données résultant de la corruption de cibles tels que l'accumulateur et le compteur de programme, conduisant à différents types d'erreurs selon l'instant d'occurrence de la faute.

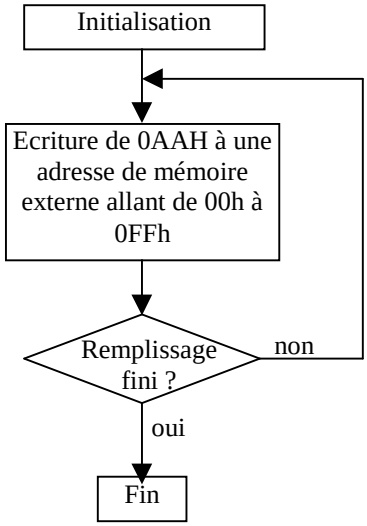


Figure 2.6 : Ecriture d'un motif dans la mémoire externe du 80C51

Le code et les informations relatives au programme sont donnés dans la table 2.4 pour permettre de suivre l'évolution des erreurs injectées dans des cas significatifs et en interpréter les conséquences.

Table 2.4 : Programme d'écriture d'un motif (0AAH) dans une zone mémoire de 256 octets

PC	Code HEX	#ligne	Code source assembleur	Commentaires
0043	75A801	29	MOV IE, #01H	Autorisation de l'interruption INT_RESET dans le registre d'interruption
0046	D2AF	30	SETB EA	
		31		
		32		
0048	900000	33	MOV DPTR, #0000h	dptr← 0000H
004B	74AA	34	MOV A, #0AAh	acc ← 0AAH
004D		35	Write_AA:	
004D	F0	36	MOVX @DPTR,A	acc ← Mem [adr]
004E	A3	37	INC DPTR	adr ← adr + 1
004F	A883	38	MOV R0, DPH	
0051	B807F9	39	CJNE R0, #01h,Write_AA	Itération de la boucle jusqu'au remplissage de 256 octets dans la mémoire externe
		40		
0054		41	First_Int:	
0054	C297	42	CLR INT	Envoi de l'interruption INT vers la carte mère
0056	00	43	NOP	
0057	D297	44	SETB INT	

B. Résultats globaux d'une session d'injection de CEU

Une session d'injection d'une durée d'environ 20 minutes a été réalisée de manière complètement automatique. Ceci a permis l'injection de 1150 CEU de manière concurrente avec l'exécution du programme. L'analyse des résultats peut être résumée comme suit :

- CEU tolérés : 1118
- CEU provoquant des erreurs dans le résultat : 15
    - 5 provenant de CEU injectés dans l'Accumulateur
    - 3 provenant de CEU sur l'octet de poids faibles de PC
    - 7 sur le registre DPL
- CEU provoquant la perte de séquencement : 17
    - 2 provenant de CEU sur le port P3
    - 5 provenant de CEU sur l'octet de poids faibles de PC
    - 10 provenant de CEU sur l'octet de poids forts de PC

Seuls 32 parmi les 1150 CEU injectés ont donc eu des conséquences sur le fonctionnement du programme testé. Le taux d'erreursde l'application peut donc être estimé comme étant d'environ 3%.

C. Quelques exemples significatifs des conséquences de CEU injectés

Dans ce paragraphe nous tenterons de montrer des cas de comportements représentatifs face aux CEU, en donnant pour chacun d'entre eux une description de l'évolution du programme en présence de la faute expliquant l'erreur obtenue. La figure 2.7 montre un exemple des informations affichées après injection d'un CEU. On y trouve les paramètres relatifs au CEU :

- ***Pmem*** désigne la position mémoire cible du CEU,
- ***Pbit*** indique, par un bit à '1', la position du bit perturbé,
- ***Nbre de Cycles*** donne le nombre de cycles s'ayant écoulé jusqu'à l'occurrence du CEU,
- ***Temps max*** spécifie la durée du programme en termes du nombre de cycles,
- ***PC*** indique l'adresse de retour au programme d'application stockée dans le compteur de programme.

Ensuite, il peut être trouvé le résultat de la comparaison, faite par la carte mère, donné sous forme d'un ou exclusif (XOR) entre les résultats attendus et les résultats obtenus. Dans le cas du programme effectuant l'écriture de 256 octets, 8 lignes de 32 octets à 00 correspondent à une exécution sans erreur. Dans ce qui suit, de l'ensemble de cette séquence il ne sera montré que la partie la plus relevante pour la compréhension des résultats.

Enfin, l'adresse de retour prévue dans le programme perturbé est affichée. Il est évident que le retour pourra être fait à des adresses différentes de celle-ci en cas de perte de séquencement produits par certains CEU.

Pmem = PCL ; Pbit = 01 ; Nbre de Cycles = 000019 ; Temps max = 003840

00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000

PC = 004e

Figure 2.7 : Affichage après injection d'un CEU sur PC n'ayant de conséquences sur l'exécution du programme

Analyse:

Dans les résultats montrés dans la figure 2.7, la cible du CEU est l'octet de poids faible du compteur programme (PCL) qui valait 4EH au moment de l'occurrence du CEU. Le bit inversé est le bit de poids faibles, la valeur de PCL passe donc de 4EH à 4FH. Cette erreur n'a pas de conséquences au niveau de l'exécution du programme. Ceci est dû au fait que le PC passe de l'adresse 4DH directement à l'adresse 4FH sans exécuter l'instruction d'incrémentation de DPTR (ligne #37 dans la table 2.4). La même adresse est donc écrite deux fois avec le motif correct 0AAH.

Lors de l'injection de CEU dans l'exemple donné dans la table 2.4, les cas typiques d'erreurs considérées sont :

- Erreurs critiques : perte de séquencement suite à un CEU sur PC,
- Résultats « aberrants » suite à une perte de séquencement provoquant une mauvaise interprétation du code par le 80C51,
- Résultats « trompeurs » : à première vue l'analyse des résultats pourrait conduire à déduire qu'un upset s'est produit dans une autre cible.
- Erreurs fines suite à une perte de séquencement conduisant à l'exécution deux fois d'une instruction,
- CEU sur PC ayant des conséquences semblables à celles sur l'accumulateur,
- CEU tolérés .

1) Exemples de perte de séquencement

- Exemple n°1 :

Pmem = PCL ; Pbit = 10 ; Nbre de Cycles = 00172a ; Temps max: 003840

****Lost sequence****
PC = 0051

Figure 2.8 a): Affichage après injection d'un CEU sur PC résultant en une perte de séquencement (dépassement du temps limite du chien de garde)

Analyse:

Du fait du changement de la valeur de PCL de 51H à 41H (Figure 2.8 a), toute la boucle est ré-exécutée ce qui provoque le dépassement du temps maximum attribué à l'exécution de ce programme. La carte mère ne reçoit pas l'interruption (INT) à temps et son *chien de garde* signale la perte de séquencement.

- Exemple n°2 :

Pmem = PCL ; Pbit = 10 ; Nbre de Cycles = 001c66 ; Temps max = 003840

****Lost sequence****

PC = 004e

Figure 2.8 b) : Affichage après injection d'un CEU sur PC résultant en une perte de séquencement total du programme

Analyse :

La valeur de PCL passe de 4EH à 5EH (Figure 2.8. b), se branchant hors du programme, sautant donc les instructions correspondantes à l'envoi de l'interruption. La carte mère ne reçoit pas d'interruption d'où la perte de séquencement.

2) Exemple de résultat aberrant

Pmem = PCL ; Pbit = 01 ; Nbre de Cycles = 0003bd ; Temps max = 003840

00000000000000000000000000000000000000000055555555555555555555
…

PC = 0051

Figure 2.9 : Affichage après injection d'un CEU sur PC donnant des erreurs de résultats

Analyse:

Changer la valeur de PCL de 51H à 50H (Figure 2.9), est le cas typique de coupure d'instruction conduisant à une fausse interprétation d'un code non prévu de la part du microcontrôleur. Par conséquent, à partir d'une adresse donnée (liée à l'instant d'occurrence du CEU), le programme arrête d'écrire la valeur 0AAH dans la mémoire initialisée à 0FFH, d'où la valeur 055H résultat du « ou exclusif » entre (0FFH et 0AAH).

3) Exemple de résultat trompeur

Pmem = PCL ; Pbit = 01 ; Nbre de Cycles = 000033 ; Temps max = 003840

00000000**aa**00000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000

PC = 004f

Figure 2.10 : Affichage après injection d'un CEU sur PC donnant des erreurs de résultats particuliers

Analyse:

Changer la valeur de PCL de 4FH à 4EH (Figure 2.10), fait exécuter deux fois l'instruction d'incrémentation de DPTR et par conséquent on n'écrit pas à l'adresse 04H, d'où la valeur de cette adresse erronée.

4) Cas de CEU sur l'accumulateur

L'importance de l'instant d'occurrence du CEU apparaît clairement quand l'accumulateur A est choisi comme cible. En effet, selon le type de programme, le contenu de ce registre peut être soit très changeant, soit constant durant une longue période. Dans le premier cas, il est évident que la plupart de CEU seront tolérés par la réécriture ultérieure d'une nouvelle valeur. Un bon exemple du deuxième cas est celui du programme analysé dans cette section. En effet, l'accumulateur est initialisé à la valeur du motif à écrire dans la mémoire, valeur qu'il conservera durant toute la boucle de remplissage de la mémoire. La moindre perturbation causera donc des erreurs. Le facteur de sensibilité devrait être donc proche de 100%.

Les deux exemples suivants illustrent des injections exhaustives de CEU à deux cycles successifs d'exécution de ce programme. La même stratégie (injection exhaustive de CEU à tous les cycles d'exécution de l'application) sera adoptée pour la détermination du facteur de sensibilité d'une cible donnée.

Pmem = Acc ; Pbit = 01 ; Nbre de Cycles = 000017 ; Temps max = 003840

010101010101010101010101010101010101010101010101010101010101010101
…..

PC = 004d

a) Affichage après injection d'un CEU sur l'accumulateur au cycle d'horloge n°17 h

Pmem = Acc ; Pbit = 01 ; Nbre de Cycles = 000018 ; Temps max = 003840

000101010101010101010101010101010101010101010101010101010101010101
…
PC = 004e

b) Affichage après injection d'un CEU sur l'accumulateur au prochain cycle d'horloge n°18 h

Figure 2.11 : Exemple d'expérience d'injection de CEU exhaustive dans l'accumulateur

Analyse :

Changer la valeur de l'accumulateur au 23ème (17H) cycle ou au 24ème (18H) cycles (Figure 2.11 cas a et b), a un impact différent sur le résultat. En effet, on peut constater dans la première séquence de résultats, que tous les premiers bits de la valeur écrite dans la mémoire sont inversés, alors que dans la deuxième séquence ceci ne se reproduit qu'à partir de l'adresse « 01 **»**. Ceci s'explique par le fait que dans le premier cas on avait perturbé l'accumulateur avant l'exécution de l'instruction : *MOVX @DPTR, A.* (ligne #36 de la table 2.4) vu que le PC est

égal à 004dH (figure 2.11 a). Or dans le deuxième cas, l'accumulateur a été perturbé après cette instruction puisque PC est égal à 004eH (figure 2.11 b), par conséquent, l'instruction à exécuter dès le retour au programme d'application sera « *INC DPTR* », ce qui fait qu'on n'écrit pas dans l'adresse 00H la valeur erronée de l'accumulateur.

Notez que si des CEU sont injectés dans les cycles suivants, le motif « 00 » progressera, jusqu'à ce qu'il n'y ait plus d'erreur. Quand ceci se produit, le CEU aura été injecté à la fin du programme, n'ayant donc pas d'effet. Un exemple intermédiaire est donné dans la figure 2.12 (l'accumulateur est perturbé à un instant aléatoire durant l'exécution de la boucle).

Pmem = ACC ; Pbit = 01 ; Nbre de Cycles = 000344 ; Temps max = 003840

00000000000001010101010101010101010101010101010101010101010101

PC = 0051

Figure 2.12 : Exemple d'expérience d'injection de CEU exhaustive dans l'accumulateur à un cycle d'horloge avancé

Analyse :

Après l'instant d'occurrence du CEU sur l'accumulateur, la valeur perturbée est conservée durant tout le reste du programme et par conséquent elle se manifestera sur le reste de la séquence de résultats.

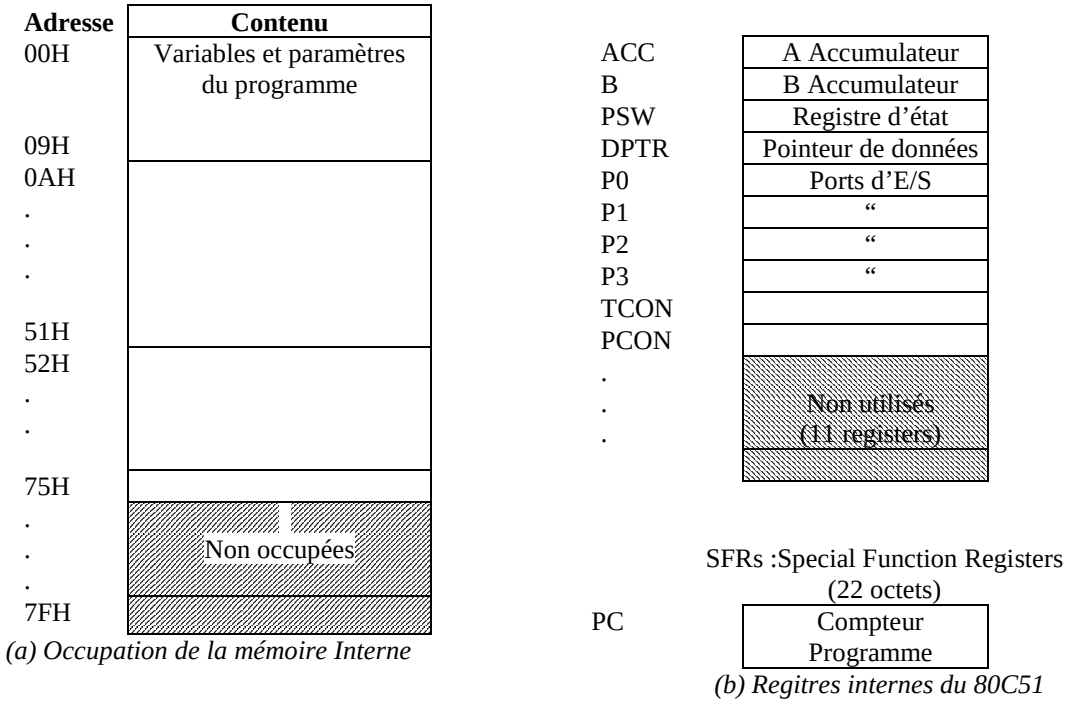
VI. ETUDE D'UN « PIRE CAS »: INJECTION DE CEU

Des expériences d'injection de CEU ont été faites sur le microcontrôleur 80C51 pendant qu'il exécutait un programme de multiplication de deux matrices (6*6). Les matrices opérandes, ainsi que celle des résultats sont stockées dans la mémoire interne. Les variables et les données de cette application occupent la majeure partie de cette mémoire, ce qui expose la presque totalité de la zone sensible du 80C51 aux effets des CEU. En effet, une fois les éléments des matrices opérandes (A et B) et de la matrice résultat (C= A*B) stockés dans la mémoire interne, il ne reste plus que 20 octets inutilisés : 10 correspondants à des registres spéciaux (SFRs) et 10 octets de la mémoire interne. Les CEU injectés sur cette zone mémoire inutilisée seront bien évidemment sans aucun effet sur le résultat final du programme.

A. Implémentation du programme

Les dix premières positions de la mémoire interne sont utilisées pour stocker les variables du programme. Les adresses à partir de 0AH jusqu'à 051H sont utilisées par les données des deux matrices A et B. Les éléments de la

matrice C sont stockés dans les adresses situées entre 052H et 075H. En plus de la mémoire interne cette application utilise les registres spéciaux SFRs du 80C51 suivants : ACC, B, PSW, DPTR, P0, P1, P2, P3, TCON, PCON et le compteur programme PC. La figure 2.13 décrit l'organisation de la mémoire. La source assembleur du programme est donnée dans l'annexe 2.2.



(a) Occupation de la mémoire Interne

(b) Regitres internes du 80C51

Figure 2.13 : Cibles de CEU : zone mémoire interne et registres

B. Discussion des résultats des essais d'injection de CEU

Nous avons injecté 12245 CEU durant l'exécution du programme de multiplication de matrices. Comme dans les cas précédents, les résultats sont répartis en trois groupes : erreurs tolérées, erreurs de résultats et pertes de séquencement. La table 2.5 constitue un récapitulatif des résultats obtenus.

Table 2.5 : Distribution des erreurs

	Erreurs tolérées	Erreurs de résultat	Perte de séquencement
Mémoire Interne (00H =< x <= 09H)	4890	359	98
Mémoire Interne (0AH =< x <= 051H)		3878	16
Mémoire Interne (052H =< x <= 07Fh)		1463	76
SFRs	1227	84	154
Total = 12245	6117 49.95%	5784 47.25%	344 2.8%

Environ la moitié des CEU injectés n'ont pas causé d'erreur sur la matrice résultats. Parmi les CEU provoquant des erreurs, un faible nombre a conduit à des pertes de séquencement. Les CEU restants donnent lieu à des erreurs dans la matrice résultats. La tolérance face aux CEU relativement faible de ce programme est étroitement liée à

l'occupation volontairement « défavorable » de la mémoire interne. En effet, si les matrices opérandes et résultat étaient stockées dans la mémoire externe et si nous considérons comme cible des CEU l'ensemble de cette mémoire, la sensibilité est réduite de presque un ordre de grandeur (table 2.6).

Table 2.6 : Distribution des erreurs

		Erreurs tolérées	Erreurs de résultat	Perte de séquencement
Mémoire Interne	10780	10323	359	98
SFRs	1465	1227	84	154
Total	= 12245	11550	443	252
		94.4%	3.6%	2%

Cependant, bien que les données de la table 2.6 pourraient être comparées directement aux résultats des radiations (les particules affectent indifféremment les positions mémoire du microcontrôleur), nous nous devons d'évaluer la sensibilité par rapport à la taille de la zone mémoire occupée, pour en tirer des chiffres objectifs. La table 2.7 résume les résultats extrapolés à partir de ceux de la table 2.6, en ne considérant que ces CEU ayant affecté des emplacements de la mémoire occupés par des variables ou par des données du programme. La sensibilité augmente d'environ 8 fois aussi bien pour les erreurs de résultats, que pour les pertes de séquencement.

Table 2.7 : Distribution des erreurs

		Erreur tolérées	Erreurs de résultat	Perte de séquencement
Mémoire Interne (00H =< x <= 09H)	937	480	359	98
SFRs	784	546	84	154
Total	= 1721	1026	443	252
		60%	**25.7%**	**14.3%**

Ces travaux rentrent dans le cadre d'une collaboration[5] réalisée dans le but d'évaluer la qualité d'une version du microcontrôleur 80C51 durcie par inclusion du code Hamming afin que chaque erreur sur la mémoire puisse être détectée et corrigée [54-58].

La qualité de *l'aléatoireté* peut être analysée suivant la distribution des CEU injectés par cible. Pour chacune des cibles a été déterminé le nombre d'erreurs provoquant l'un des 3 types d'erreurs considérées dans le but d'analyser l'existence éventuelle de cibles plus critiques que d'autres. Dans la table 2.8 sont recensés les résultats de cette analyse. Ils permettent de tirer quelques conclusions sur la sensibilité de certains des registres :

- l'accumulateur A est responsable dans 50% des cas, des erreurs de résultat, et jamais de pertes de séquencement.

[5] avec l'équipe de recherche du département d'informatique de l'université UFRGS, Porto-Algre au Brésil

- PC, PSW, DPH et surtout P0 sont à l'origine des pertes de séquencement. En effet les perturbations de ce dernier, qui véhicule le contenu du registre d'instruction et du PC, résulteront en une erreur critique dans près de 80% des cas.
- les perturbations sur PC n'occasionnent pas toujours des pertes de séquencement. Dans le cas du PCL, le taux de perte de séquencement est de seulement 11 % pour ce registre, qui cause surtout des erreurs de résultats.
- Les registres SFR ayant 100% de tolérance face aux CEU n'ont en réalité pas été utilisés par les programmes étudiés.

Table 2.8 : Distribution des erreurs dans les SFRs

SFRs +PC	Nombre d'erreurs injectées	Erreurs tolérées (%)	Erreurs de résultat (%)	Perte de séquencement(%)
ACC	45	51.11	48.89	0.00
B	49	87.76	12.24	0.00
PSW	81	76.21	7.93	15.86
SP	68	100.00	0.00	0.00
DPL	60	98.33	1.67	0.00
DPH	81	90.98	7.54	1.48
P0	66	13.64	7.58	78.78
P1	61	100.00	0.00	0.00
P2	63	100.00	0.00	0.00
P3	54	72.22	0.00	27.78
IP	47	100.00	0.00	0.00
IE	49	100.00	0.00	0.00
TMOD	61	100.00	0.00	0.00
TCON	81	99.22	0.00	0.78
TH0	50	100.00	0.00	0.00
TL0	81	100.00	0.00	0.00
TH1	81	100.00	0.00	0.00
TL1	60	100.00	0.00	0.00
SCON	47	100.00	0.00	0.00
SBUF	81	100.00	0.00	0.00
PCON	81	73.11	0.00	26.89
PCh	54	38.89	0.00	61.11
PCl	64	29.69	59.38	10.94

La table 2.9 synthétise le même type de résultats cette fois-ci pour la mémoire interne. L'inspection de cette table nous révéla l'existence d'un emplacement mémoire (l'adresse 6) qui n'avait jamais été cible d'un CEU. Ce fait, qui est lié à la méthode de génération de nombres aléatoires, met en évidence la rigueur potentielle de la méthode CEU. En effet, cette « carence » fruit du hasard de *l'aléatoireté*, peut être compensée par un essai déterministe dans lequel la cible est le mot non atteint. Dans la table 2.9, la ligne en grisée montre le résultat d'un tel essai complémentaire d'injection de CEU.

Table 2.9 : Distribution des erreurs dans la mémoire interne

Pmem Mém. Int.	Nombre d'erreurs injectées	Erreurs tolérées	Erreurs de résultat	Perte de séquencement
0	109	90.83	9.17	0.00
1	98	100.00	0.00	0.00
2	114	28.07	64.91	7.02
3	103	35.92	56.31	7.77
4	123	54.47	45.53	0.00
5	113	79.65	20.35	0.00
6	63	20.63	79.37	0.00
7	52	13.46	84.62	1.92
8	49	28.57	4.08	67.35
9	113	20.35	37.17	42.48
0a	126	86.51	13.49	0.00
0b	89	74.16	25.84	0.00
0c	81	82.95	17.05	0.00
0d	117	76.07	23.93	0.00
0e	53	71.70	28.30	0.00
0f	43	81.40	18.60	0.00
10	109	63.30	36.70	0.00
11	81	68.05	31.95	0.00
12	61	75.41	24.59	0.00
13	56	80.36	19.64	0.00
14	117	60.68	39.32	0.00
15	50	64.00	36.00	0.00
16	115	53.04	46.96	0.00
17	57	47.37	52.63	0.00
18	52	59.62	40.38	0.00
19	60	48.33	51.67	0.00
1a	73	60.27	39.73	0.00
1b	81	46.55	53.45	0.00
1c	118	43.22	56.78	0.00
1d	102	40.20	59.80	0.00
1e	105	40.95	59.05	0.00
1f	124	38.71	61.29	0.00
20	100	45.00	55.00	0.00
21	67	40.30	59.70	0.00
22	53	32.08	67.92	0.00
23	112	25.00	75.00	0.00
24	81	19.99	80.11	0.00
25	48	31.25	68.75	0.00
26	94	23.40	76.60	0.00
27	57	35.09	64.91	0.00
28	100	15.00	85.00	0.00
29	52	9.620	90.38	0.00
2a	52	15.38	84.62	0.00
2b	113	9.73	90.27	0.00
2c	104	10.58	89.52	0.00
2d	81	12.50	87.50	0.00
2e	105	20.00	80.00	0.00
2f	122	20.49	79.51	0.00
30	81	17.17	82.83	0.00
31	97	14.43	85.57	0.00

32	64	18.75	81.25	0.00
33	119	19.33	80.67	0.00
34	48	35.42	64.58	0.00
35	67	32.84	67.16	0.00
36	81	35.38	64.62	0.00
37	62	19.35	80.65	0.00
38	117	23.08	76.92	0.00
39	123	20.33	79.67	0.00
3a	105	26.67	73.33	0.00
3b	121	20.66	79.34	0.00
3c	103	18.45	81.55	0.00
3d	95	15.79	84.21	0.00
3e	81	16.75	83.25	0.00
3f	81	17.42	82.58	0.00
40	106	46.23	53.77	0.00
41	54	50.00	50.00	0.00
42	55	49.09	50.91	0.00
43	81	39.28	60.72	0.00
44	103	45.63	54.37	0.00
45	59	42.37	57.63	0.00
46	53	28.30	71.70	0.00
47	84	25.00	75.00	0.00
48	52	23.08	76.92	0.00
49	113	23.89	76.11	0.00
4a	51	15.69	84.31	0.00
4b	59	8.470	91.53	0.00
4c	41	31.71	68.29	0.00
4d	100	22.00	78.00	0.00
4e	120	32.50	67.50	0.00
4f	67	28.36	71.64	0.00
50	108	27.78	72.22	0.00
51	118	27.12	72.88	0.00
52	102	14.71	85.29	0.00
53	46	17.39	82.61	0.00
54	131	15.27	84.73	0.00
55	107	28.97	71.03	0.00
56	129	23.26	76.74	0.00
57	112	25.00	75.00	0.00
58	56	26.79	73.21	0.00
59	53	26.42	73.58	0.00
5a	99	35.35	64.65	0.00
5b	116	31.90	68.10	0.00
5c	50	34.00	66.00	0.00
5d	56	37.50	62.50	0.00
5e	121	39.67	60.33	0.00
5f	129	41.09	58.92	0.00
60	72	34.72	65.28	0.00
61	44	47.73	52.27	0.00
62	106	39.62	60.38	0.00
63	106	58.49	41.51	0.00
64	57	52.63	47.37	0.00
65	134	58.21	41.79	0.00
66	104	56.73	43.27	0.00

67	56	62.50	37.50	0.00
68	57	68.42	31.58	0.00
69	81	65.81	34.19	0.00
6a	55	72.73	27.27	0.00
6b	81	69.00	31	0.00
6c	61	70.49	29.51	0.00
6d	50	60.00	40.00	0.00
6e	81	75.4	24.6	0.00
6f	58	79.31	20.69	0.00
70	50	70.00	30.00	0.00
71	81	82.80	17.20	0.00
72	60	86.67	13.33	0.00
73	53	88.68	11.32	0.00
74	57	89.47	10.53	0.00
75	81	92.53	7.47	0.00
76	81	100.00	0.00	0.00
77	81	100.00	0.00	0.00
78	81	100.00	0.00	0.00
79	81	100.00	0.00	0.00
7a	81	100.00	0.00	0.00
7b	81	100.00	0.00	0.00
7c	81	100.00	0.00	0.00
7d	81	100.00	0.00	0.00
7e	94	100.00	0.00	0.00
7f	112	100.00	0.00	0.00

Les deux tables précédentes présentent la contribution de chaque registre ou position mémoire interne dans la détermination du taux d'erreurs déterminé par injection dans un bit et un cycle sélectionnés tous les deux aléatoirement. Une potentialité intéressante de l'approche d'injection de CEU est la détermination des facteurs de sensibilité des registres et des zones mémoires. Elle consiste en l'observation du contenu du registre ou de la position mémoire considérée durant toute l'exécution du programme étudié ce qui revient à injecter un CEU durant chaque cycle (coup d'horloge) d'exécution du programme.

Nous avons réalisé un essai d'injection (exhaustive dans tous les cycles d'exécution du programme) sur un élément d'une des matrices données. Des extraits relevants des résultats obtenus sont donnés dans l'annexe 2.3. Les résultats obtenus montrent bien la période de sensibilité de l'élément mémoire considéré. En effet, l'erreur sur la matrice résultat apparaît à partir d'un certain moment pour disparaître une fois que la cible ne contribue plus au calcul final. Ceci a permis d'estimer à environ 80% le facteur de sensibilité de cette cible.

Cette méthode, généralisable à toutes les cibles de CEU, est automatisable et permet donc la détermination de facteurs de sensibilité à prendre en compte dans le calcul de la section efficace de l'application. Une section entière

a été consacrée au calcul de ces facteurs de sensibilité dans le chapitre suivant dans le cas du microprocesseur TS6833216A pour la prédiction de la section efficace d'une application de multiplication de matrices.

VII. RESULTATS DES TESTS SOUS RADIATION

Afin d'étudier la précision de la prédiction du taux d'erreurs par la méthode d'injection de CEU, nous avons effectué des expériences préliminaires de test sous radiation. Par l'intermédiaire du testeur THESIC, le microcontrôleur 80C51 a été exposé aux effets d'une source radioactive (Cf. 252), à l'aide de l'équipement CIRIL, développé à l'ONERA - DESP (Toulouse, France) [59].

L'approche que nous avons adoptée suit les deux étapes essentielles de prédiction, présentées dans le chapitre 1:

- La section efficace intrinsèque du 80C51 est déterminée par l'exécution du test statique (ciblant des SFRs et la mémoire interne).
- La section efficace prédite du programme de multiplication de matrices est calculée selon la formule (1.9) :

$$\tau_{SEU} = \sigma_{SEU} * \tau_{CEU}$$

Où :

- σ_{SEU} : Section efficace aux SEUs
- τ_{CEU} : Taux d'erreurs global obtenu par injection de CEU
- τ_{SEU} : Section efficace du programme de multiplication de matrices

La section efficace du microcontrôleur 80C51, évaluée pour des flux de particules issus du californium, a été mesurée comme étant 3.32×10^{-3} cm²/composant. Des sessions d'injection de CEU ont été effectuées durant lesquelles le nombre de CEU injectés est approximativement le même que celui des basculements de bits détectés durant les tests de radiation. La table 2.10 donne les sections efficaces mesurées et prédites, pour chaque type d'erreur, dérivées de l'ensemble des tests de radiation réalisés.

Table 2.10 : Sections efficaces mesurées et prédites
Programme exécuté: une multiplication de matrices 6x6

	Section efficace [cm² / composant]	
	Mesurée	**Prédite**
Erreurs de résultats	1.45×10^{-3}	1.57×10^{-3}
Perte de séquencement	1×10^{-4}	9.3×10^{-5}
Section efficace globale	1.55×10^{-3}	1.66×10^{-3}

La table 2.11 donne le pourcentage des erreurs, issu des expériences de radiation et d'injection de CEU, pour chaque type d'erreur. Ces résultats ont été présentés à NSREC -2000 [37].

Table 2.11 : Résultats de Radiation vs. Injection de CEU

Programme exposé: Multiplication de Matrices (6x6)

Types d'erreurs	**Radiation**	**Injection**
Erreurs de résultats (1)	43.7 %	47.25 %
Perte de séquencement (2)	3.01 %	2.8 %
Erreurs (1) + (2)	46.71 %	50.05 %

La bonne corrélation entre les taux d'erreurs mesurés et prédits, met clairement en évidence les performances de l'approche d'injection de CEU. Toutefois, ces tests de radiation au Californium, dont le flux est généralement assimilé à un flux mono - énergétique (avec un LET de 42 MeV/mg/cm²), ont permis de déterminer un seul point de la courbe complète du taux d'erreurs de cette application. Pour déterminer la courbe du taux d'erreurs aux SEU de ce programme de multiplication de matrices, sont nécessaires des essais sous radiation, effectués avec différents types d'ions lourds. La confrontation des courbes de sections efficaces du programme de multiplication de matrices prédites et mesurées constituera le meilleur moyen d'étudier les potentialités de la technique étudiée d'injection de fautes.

De telles campagnes de radiation ont été faites à l'aide de deux installations différentes : l'accélérateur linéaire de particules Tandem de Van der Graff de l'Institut de Physique Nucléaire (Orsay, France) et le cyclotron, Cyclone (Louvain la Neuve, Belgique).

A. Accélérateur de particules Tandem Van der Graff

L'accélérateur Tandem de Van der Graff de l'IPN (figure 2.14), est une machine électrostatique (tension maximale 15 MV) qui peut accélérer une importante palette d'ions, allant des particules jusqu'aux agrégats massiques [60]. Le Tandem dispose d'un site d'irradiation dédié aux études des effets des radiations et les techniques de durcissement sur les composants électroniques.

Figure 2.14 : Equipement de test sous radiation du Tandem Van der Graff

Le microcontrôleur 80C51 a été exposé aux faisceaux composés de différents types d'ions lourds dans l'enceinte sous vide de l'accélérateur de particules. Pour l'un de ces faisceaux, celui avec des ions de type Chlore, nous avons changé deux fois l'angle d'incidence de ces faisceaux de particules sur le circuit (48°, 60°) afin de faire varier la valeur du LET effectif. En effet, en faisant varier l'angle d'incidence du flux de particules, la charge déposée sur le volume sensible du composant est modifiée d'où l'augmentation du LET effectif qui est calculé comme suit :

$$LET_{eff} = \frac{LET_0}{\cos\theta} \qquad (2.1)$$

Où :

- θ est l'angle incident du faisceau d'ions,
- LET_0 : LET nominal à l'angle 0,
- LET_{eff} : LET effectif.

De même, le flux effectif d'ions atteignant la surface du composant est modifié selon la formule (2.2) :

$$Flux_{eff} = Flux_0 \cos\theta \qquad (2.2)$$

- $Flux_0$: Flux de particules nominal
- $Flux_{eff}$: Flux de particules effectif

Les caractéristiques physiques des ions utilisés sont données dans la table 2.12.

Table 2.12 : Différents faisceaux utilisés pour le 80C51

Tandem de Van der Graff, Orsay

Type d'ion lourd	ENERGIE DUT [MEV]	LET [MeV/mg/cm²]	Angle degrés	LET Effectif [MeV/mg/cm²]
Chlore (Cl)	80	12,7	0	12,7
//	//	//	48	19,5
//	//	//	60	25,4
Brome (Br)	192	40,7	0	40,7

B. Cyclotron de Louvain la Neuve

Le cyclotron Cyclone de Louvain la Neuve, est un cyclotron capable d'accélérer des protons (jusqu'à 85 MeV), des particules alpha et des ions lourds. La photo représentée sur la figure 2.15 montre le testeur THESIC (carte mère et carte fille 80C51) dans l'enceinte sous vide de l'équipement de test sous ions lourds. Les principales caractéristiques des ions lourds, auquel le processeur étudié a été exposé lors des essais réalisés à l'aide de cette installation, sont données dans la table 2.13. Des détails supplémentaires concernant cet équipement sont donnés dans la référence [61].

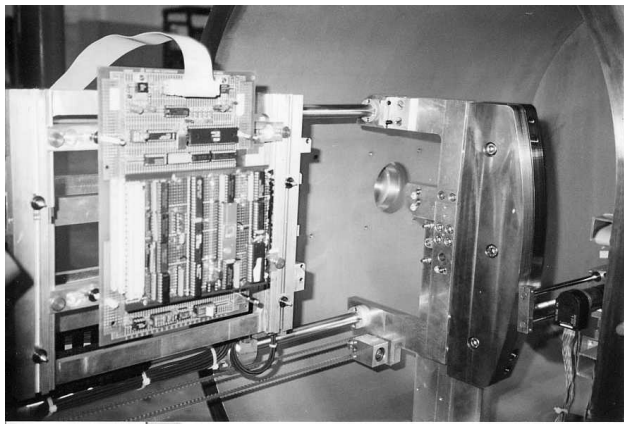


Figure 2.15 : Système THESIC dans la gamelle de l'équipement de test sous ions lourds du Cyclotron de Louvain la Neuve

Table 2.13 : Différents faisceaux utilisés pour le 80C51

Cyclone, Louvain la Neuve

M/Q=5	ENERGIE DUT [MEV]	LET [MeV/mg/cm²]
$^{40}Ar^{8+}$	150	14.1
$^{20}Ne^{4+}$	78	5.85
$^{15}N^{3+}$	62	2.97
$^{10}B^{2+}$	41	1.7
$^{84}Kr^{17+}$	316	34

- **M** la masse atomique
- **Q** est l'état de charge de l'ion

C. *Résultats expérimentaux*

Un test statique (vérification continuelle du contenu de la mémoire interne et des registres) du microcontrôleur 80C51, a été réalisé à l'aide des deux équipements cités précédemment, sous l'effet de divers types d'ions lourds dont les LET varient entre 2,97 et 40,7 MeV/mg/cm². Les sections efficaces aux SEU ont été déterminées et sont données dans la table 2.14 et représentées dans la figure 2.16.

Table 2.14 : Sensibilités du 80C51 aux SEU

Type d'ion lourd	LET [MeV/mg/cm²]	Angle degrés	LET Effectif [MeV/mg/cm²]	Section efficace aux SEU [cm² / composant]
Azote (N)	2,97	0	2,97	$4{,}00\ 10^{-6}$
Néon (Ne)	5,85	0	5,85	$3{,}10\ 10^{-4}$
Chlore (Cl)	12,7	0	12,7	$7{,}55\ 10^{-4}$
ARGON (AR)	14,1	0	14,1	$8{,}65\ 10^{-4}$
Chlore (Cl)	12,7	48	19,5	$1{,}20\ 10^{-3}$
Chlore (Cl)	12,7	60	25,4	$1{,}51\ 10^{-3}$
Krypton (Kr)	34	0	34	$1{,}77\ 10^{-3}$
Brome (Br)	40,7	0	40,7	$1{,}80\ 10^{-3}$

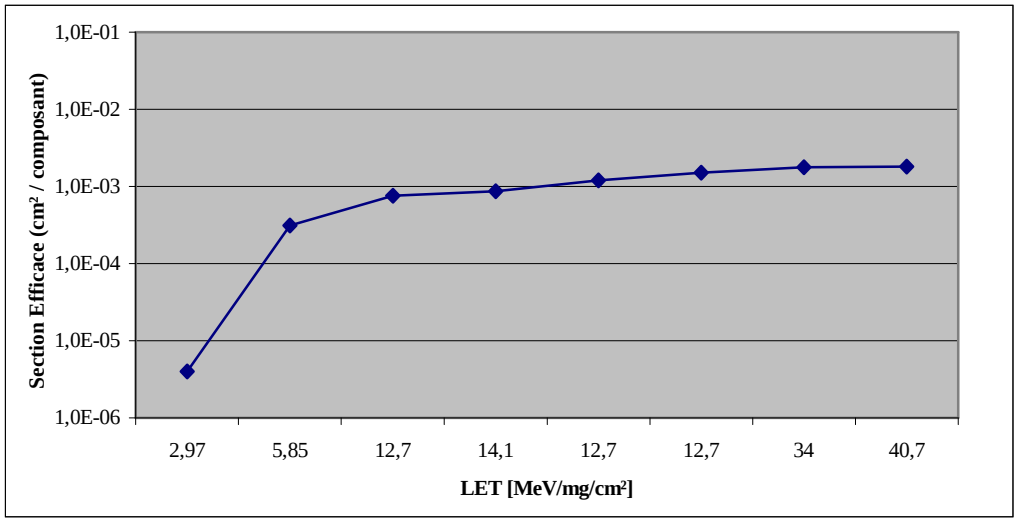


Figure 2.16 : Courbe de section efficace aux SEU du 80C51

En appliquant la formule $\tau_{SEU} = \sigma_{SEU} * \tau_{CEU}$, les valeurs des sections efficaces du programme de multiplication de matrices exécuté sur le 80C51, peuvent être estimés à partir :

- des valeurs de section efficace σ_{SEU} (cinquième colonne de la table 2.14),
- des taux de CEU injectés ($\tau_{CEU} = 50{,}05\%$).

Les sections efficaces du programme étudié mesurées et prédites, obtenues avec ces deux accélérateurs de particules sont donnés dans la table 2.15 (en gris) et représentées sur la figure 2.17. Leur comparaison met en évidence la très bonne corrélation entre les résultats prédits et mesurés, à un point tel que les deux courbes se

superposent dans tous les cas sauf celui correspondant aux ions du Néon (LET =5,85 [MeV/mg/cm²]) où la différence reste tout de même négligeable. Les sections efficaces prédites de l'application de multiplication de matrices sont parfois supérieures et parfois inférieures à celles mesurées. Si on se base sur le fait que certains éléments mémoire ne sont pas accessibles aux CEU, l'estimation devrait être a priori optimiste, ce qui n'est pas toujours le cas dans la table 2.15. En effet, on retouve parfois des cas où l'estimation de la sensibilté de l'application est pessimiste (la section efficace prédite est supérieure à celle mesurée), ceci peut être expliqué d'une part par *l'aléatorité* des basculements provoqués soit par les radiations soit par l'injection de fautes et d'autre part par le fait que les valeurs des deux sections prédites et mesurées soient très proches.

Table 2.15 : Sections efficaces mesurées et prédites
Programme exposé: Multiplication de Matrices (6x6)

Type d'ion lourd	LET Effectif [MeV/mg/cm²]	Section Efficace [cm² / composant]	
		Mesurée	Prédite
Azote (N)	2,97	$2{,}00\ 10^{-6}$	$2{,}00\ 10^{-6}$
Néon (Ne)	5,85	$1{,}02\ 10^{-4}$	$1{,}55\ 10^{-4}$
Chlore (Cl)	12,7	$3{,}96\ 10^{-4}$	$3{,}78\ 10^{-4}$
ARGON (AR)	14,1	$4{,}50\ 10^{-4}$	$4{,}33\ 10^{-4}$
Chlore (Cl)	19,5	$6{,}63\ 10^{-4}$	$6{,}00\ 10^{-4}$
Chlore (Cl)	25,4	$7{,}13\ 10^{-4}$	$7{,}55\ 10^{-4}$
Krypton (Kr)	34	$9{,}12\ 10^{-4}$	$8{,}86\ 10^{-4}$
Brome (Br)	40,7	$8{,}85\ 10^{-4}$	$9{,}00\ 10^{-4}$

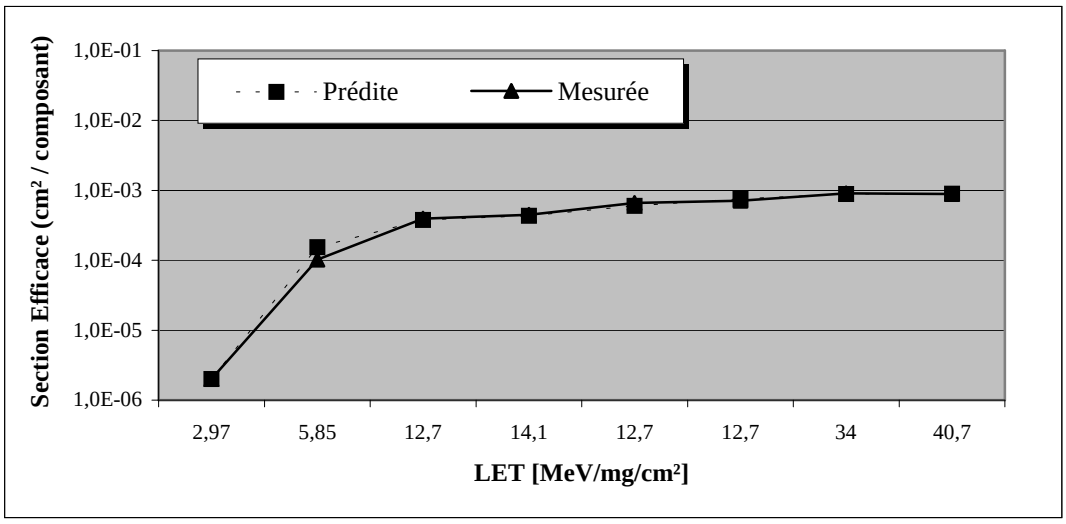


Figure 2.17 : Sections efficaces prédites et mesurées aux SEU du 80C51
Programme exposé: une multiplication de matrices

Ces premiers résultats mettent en évidence, du moins dans le cas du microcontrôleur 80C51 d'Intel, l'excellente efficacité de cette nouvelle technique pour la prédiction des sections efficaces de programmes. Tout de même, la précision de la prédiction dans le cas du microcontrôleur 80C51 est peut-être due à la simplicité du programme

exécuté sous radiation et à la possibilité offerte par le microcontrôleur 80C51 pour accéder pratiquement à toute la zone sensible (l'accès est estimé approximativement à 93%). Afin de prouver la possibilité de généralisation de cette technique d'injection de CEU, son application à d'autres processeurs exécutant différents programmes sera étudiée dans le chapitre suivant.

VIII. CONCLUSION

Nous avons présenté une application de la technique d'injection de CEU dans le cas d'une architecture digitale bâtie autour du microcontrôleur 80C51 d'Intel. Les basculements de bits dans les zones accessibles (registres, mémoire, etc.) sont injectés simultanément avec l'exécution du programme par une interruption asynchrone afin de simuler le comportement du microcontrôleur 80C51 face aux effets des radiations. Les performances de la méthode d'injection de CEU ont été évaluées, pour deux programmes simples exécutés par un microcontrôleur 8 bits. Le but du premier est l'étude de la stratégie de l'implantation de la technique d'injection de CEU ainsi que les types d'erreurs obtenus avec des extraits de résultats à l'appui.

Des milliers d'upsets ont été injectés en parallèle avec l'exécution d'un deuxième programme : une multiplication de matrices. Les résultats obtenus sont en excellente corrélation avec ceux issus sous radiation, mettant clairement en évidence l'une des principales utilisations de l'approche d'injection de CEU: la prédiction de la section efficace d'un programme donné seulement à partir de la section efficace et les résultats de simulation de CEU et ceci sans exécuter le programme étudié sous radiations.

L'objectif de notre travail futur consiste à étendre l'application de cette technique à des processeurs complexes tels que les processeurs de traitement de signal, le TMS320C50 de Texas Instruments, le ADSP21060 (SHARC), le microprocesseur Motorola TS6833216A. Le but est d'estimer les sections efficaces de différents programmes y compris d'applications de contrôle d'instruments à bord de véhicules spatiaux. La précision de la prédiction des sections efficaces de ces programmes sera évaluée par confrontation aux mesures issues de tests sous radiation effectués avec différents types d'ions lourds.

CHAPITRE 3

ETUDE DU COMPORTEMENT D'UN MODEM EN PRESENCE DE CEU ET PREDICTION DU TAUX D'ERREURS

I. INTRODUCTION

La méthode d'injection de CEU a été conçue principalement pour la prédiction de la section efficace d'une application donnée, exécutée sur un processeur. La deuxième utilité de cette nouvelle technique consiste en l'étude du comportement d'un processeur face aux erreurs transitoires provoquées par les radiations, afin d'évaluer la robustesse de l'application étudiée et de l'améliorer dans la mesure du possible. Ce chapitre mettra en évidence l'utilisation de la technique d'injection de CEU pour l'étude de comportement et la prédiction du taux d'erreurs en présence d'upsets, d'une application modem implantée à l'aide d'un processeur de traitement de signal.

Dans la première partie de ce chapitre est présentée l'architecture interne du processeur étudié décrivant les zones mémoires internes et les registres, cibles principales de la méthode d'injection de CEU. Puis sont brièvement décrites l'architecture du modem ainsi que la configuration adoptée pour l'implantation de cette application avec le DSP. La deuxième partie décrit la mise en œuvre de l'injection d'erreurs. Enfin, les résultats des essais d'injection de CEU sont discutés et confrontés à ceux obtenus lors de l'exposition de ce circuit à un faisceau d'ions lourds.

II. APPLICATION IMPLANTEE : MODEM DE COMMUNICATIONS

Les essais d'injection de CEU décrits dans le chapitre précédent ont montré la possibilité d'estimation de la sensibilité d'une architecture à base d'un 80C51 face aux upsets survenant lors de l'exécution de programmes simples. L'objectif final de notre étude est de prouver que cette approche permet de prédire la section efficace d'une application quelconque à partir de résultats de tests statiques effectués sous radiations et de ceux issus de sessions d'injections de CEU. Nous avons considéré important que cette démarche soit validée sur une application complexe, faisant partie d'un projet spatial. Une telle application nous a été proposée par l'INTA (*Instituto Nacional de Técnica Aeroespacial*, Espagne) sous forme d'un modem développé pour le projet Nanosat.

A. Projet Nanosat

L'objectif principal du projet Nanosat [62] est d'établir la communication pour la transmission et la réception des données depuis des stations scientifiques terrestres très éloignées. Le satellite obtient les données quand il passe par la station à observer, et après les avoir stockées, les dépose de nouveau au point de réception pour qu'elles soient analysées. Pour concevoir ce système de communications, appelé système de stockage et d'envoi, un programme spécifique (modem) a été conçu et implanté sur un processeur de traitement de signal, le TMS320C50 de Texas Instruments. Mais l'un des buts que poursuit le Nanosat est la validation du comportement de certains composants dans l'espace en observant leur réponse à la radiation ionisante provenant de l'espace extérieur.

C'est ainsi que ce projet a eu lieu dans le but de concevoir et de développer une carte fille destinée à réaliser le test de deux circuits candidats au projet Nanosat. Ces circuits sont : un processeur de traitement du signal (le TMS320C50 de Texas Instruments) et une SRAM de 128KB (M65608 de TEMIC) utilisée pour le stockage des données. Ces circuits ont été testés sous radiations en Juillet 1998 avec le cyclotron 68" du LBL (Lawrence Berkeley Laboratory, Berkeley, USA) pour prédire leurs taux d'erreurs une fois dans l'espace [63].

B. Description du système de communications du Nanosat

Le schéma bloc du système de communications de Nanosat est donné dans la figure 3.1. Le fonctionnement est le suivant : l'émetteur implanté sur le DSP génère les échantillons d'un signal à transmettre, à partir d'une certaine séquence d'informations binaires correspondant à un signal reçu échantillonné et démodulé par le récepteur. Cette séquence d'échantillons est remise au convertisseur D/A. Le bloc de radiofréquence (RF) génère, via une suite d'étapes de modulation et de démodulation appropriées, les signaux analogiques RxI et RxQ (en phase et en quadrature de phase), et les transmet au canal ADC (Canal Analogique/Digital). Ce dernier comporte deux convertisseurs A/D, de telle sorte que les deux flux d'échantillons en sortie sont générés simultanément à partir des signaux RxI et RxQ. Le récepteur récupère, à partir des échantillons du convertisseur A/D, l'information transmise et la livre au DSP, qui détermine la validité de celle-ci et établit, s'il le faut, les procédés pour sa retransmission.

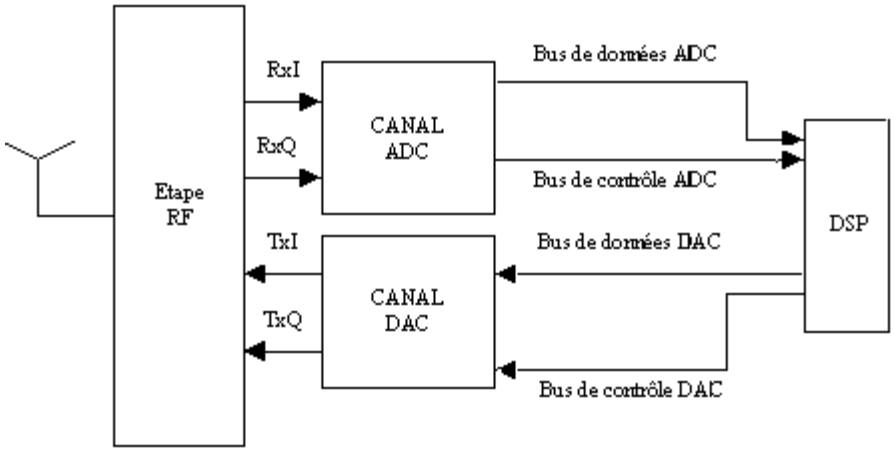


Figure 3.1 : Description du système de communications

Le DSP a été programmé par l'INTA pour fonctionner en tant que modem. Le programme correspondant comporte un récepteur et un émetteur, répartis sur des blocs schématisés dans la figure 3.2, dont des descriptions succinctes sont données dans l'annexe 3.1. Aussi bien à l'étape de transmission qu'à l'étape de réception, le modem a deux modes de fonctionnement : mode apprentissage et mode données. Le mode d'apprentissage est utilisé pour ajuster le circuit de synchronisation de la phase du récepteur. Une séquence de données connue est transmise avec pour objectif que le récepteur puisse connaître la phase du signal qui lui est présenté afin d'ajuster au mieux ses paramètres. Ceci a pour but l'amélioration du rapport signal/bruit. Une fois que le récepteur est synchronisé, on passe au mode normal de fonctionnement du modem qui est le mode données. Dans celui-ci des signaux d'entrée sont transmis après avoir été modulées et démodulées.

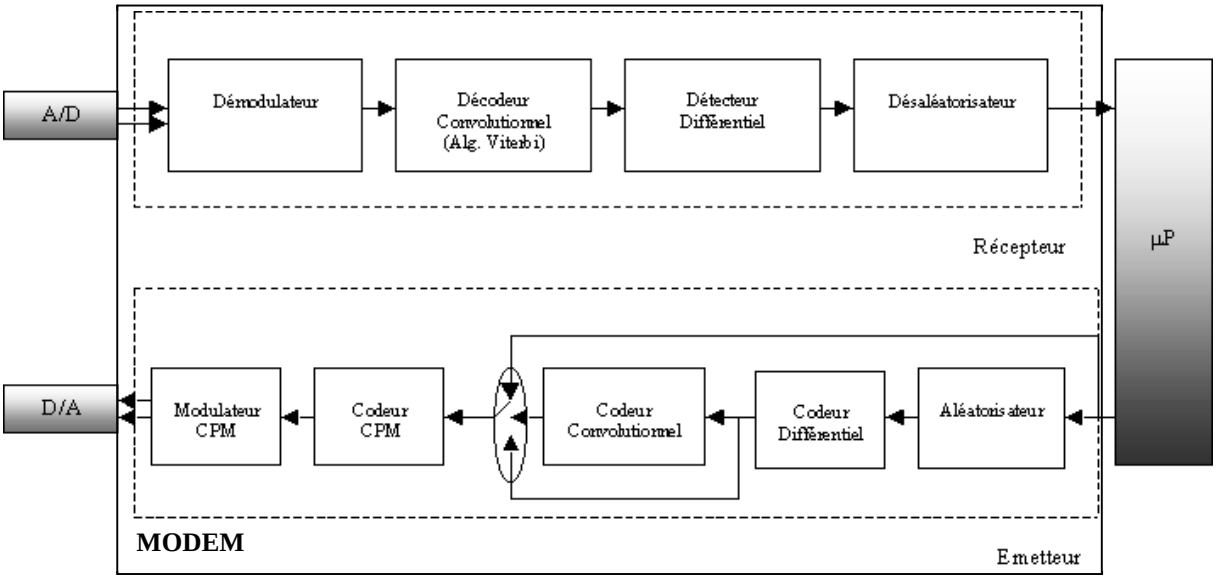


Figure 3.2 : Schéma bloc du MODEM

Nous avons implanté ce programme modem sur une carte fille THESIC disponible, construite autour du processeur de traitement de signal le TMS320C50 dans le but d'effectuer des tests sous radiations pour qualifier ce circuit. Les considérations suivantes ont été prises en compte lors de la conception de cette carte (voir Annexe 3.2) :

- Les convertisseurs D/A et A/D n'ont pas été implantés, la sortie de l'émetteur est directement bouclée sur l'entrée du récepteur.
- Les stimuli d'entrée du modem sont stockés dans la mémoire externe sous forme de séquences de mots de 16 bits. Dans ces séquences seulement un bit (celui de poids faible) est utilisé pour coder la valeur à un instant donné du signal à transmettre.
- L'émetteur et le récepteur fonctionnent de façon totalement synchronisée suivant une interruption externe pilotée par le DSP.

Mis à part le contrôle de l'interruption (qui a été effectué grâce à un timer du DSP), aucune modification matérielle n'a été apportée à la carte fille TMS320C50 existante. De point de vue logiciel, outre le rebouclage du modem sur lui même mentionné plus haut, la modification suivante a été apportée au programme modem original : trois points d'observation ont été ajoutés au programme modem sous forme de variables de contrôle « flags », afin d'obtenir des informations sur la nature des erreurs résultant de l'injection de CEU :

1) Le FLAGAPPR indique si une erreur s'est produite dans certaines variables durant le mode d'apprentissage.

2) Le FLAGVERIF indique s'il y a eu des erreurs sur des constantes qui devraient rester fixes pendant toute l'exécution du programme.

3) Le FLAGDON indique s'il y a eu une erreur sur le bit de sortie avant d'arriver à la fin du procédé de réception.

La comparaison entre les données d'entrée et celles de sortie est réalisée chaque fois qu'un signal est traité. En cas d'erreur, son adresse dans la mémoire et sa valeur ainsi que la valeur de Flagdon sont stockés dans la mémoire commune MMI. Chaque fois que le DSP a fini de traiter les données du buffer d'entrée, il envoie une interruption à la carte mère, après avoir stocker dans la MMI les valeurs des flags de contrôle, le nombre d'erreurs détectées et les transformations concernant ces erreurs.

III. IMPLANTATION DU MODEM SUR LE TMS320C50PQ

La carte fille THESIC a été initialement conçue à TIMA pour tester sous radiations le DSP TMS320C50PQ [62]. Pour faire fonctionner l'application modem sur le DSP, des modifications adéquates ont été effectuées. Les données d'entrée et de sortie du modem, les variables nécessaires au test ainsi que celles nécessaires à l'injection de fautes qu'on appellera par la suite « paramètres d'injection » sont stockées dans une mémoire externe.

A. Architecture Interne du TMS320C50

Les principales caractéristiques du DSP sont :

Espace Mémoire

- 64Kx16bit de mémoire programme,
- 64Kx16bit de mémoire données,
- 64Kx16bit de mémoire entrée/sortie,
- 32Kx16bit de mémoire globale.

Mémoire interne

- 1056 mots de mémoire RAM d'accès double (DARAM),
- 9Kx16bit de mémoire RAM d'accès unique (SARAM).

CPU

- Unité Arithmétique et Logique (UAL) 32-bit,
- Accumulateur 32-bit (ACC),
- Multiplieur parallèle de 16-bit x 16-bit,
- Unité logique parallèle (PLU) de 16-bit,
- 8 registres auxiliaires.

Contrôle de Programme

- Pile matérielle de 8 niveaux,
- Pipeline de 4 étages,
- Sauvegarde automatique des principaux registres lors du service des interruptions,
- 2 registres pour adressage circulaire.

Jeux d'instructions

- Instructions de multiplication/addition dans un seul cycle,
- Opérations pour répétitions de blocs d'instructions,

- Instructions pour le transfert de blocs de mémoire,
- Mode d'adressage indexé,
- Adressage *bit-reveresed* pour FFT.

Périphériques internes

- 16 générateurs d'états d'attente programmables,
- Port série avec multiplexage temporel (TDM),
- Port de communication série synchrone bidirectionnelle,

Le diagramme des blocs fonctionnels est donné dans la figure 3.3. Les détails concernant le principe de fonctionnement de ces blocs sont donnés dans la documentation technique du TMS320C50 donnée à la référence [65].

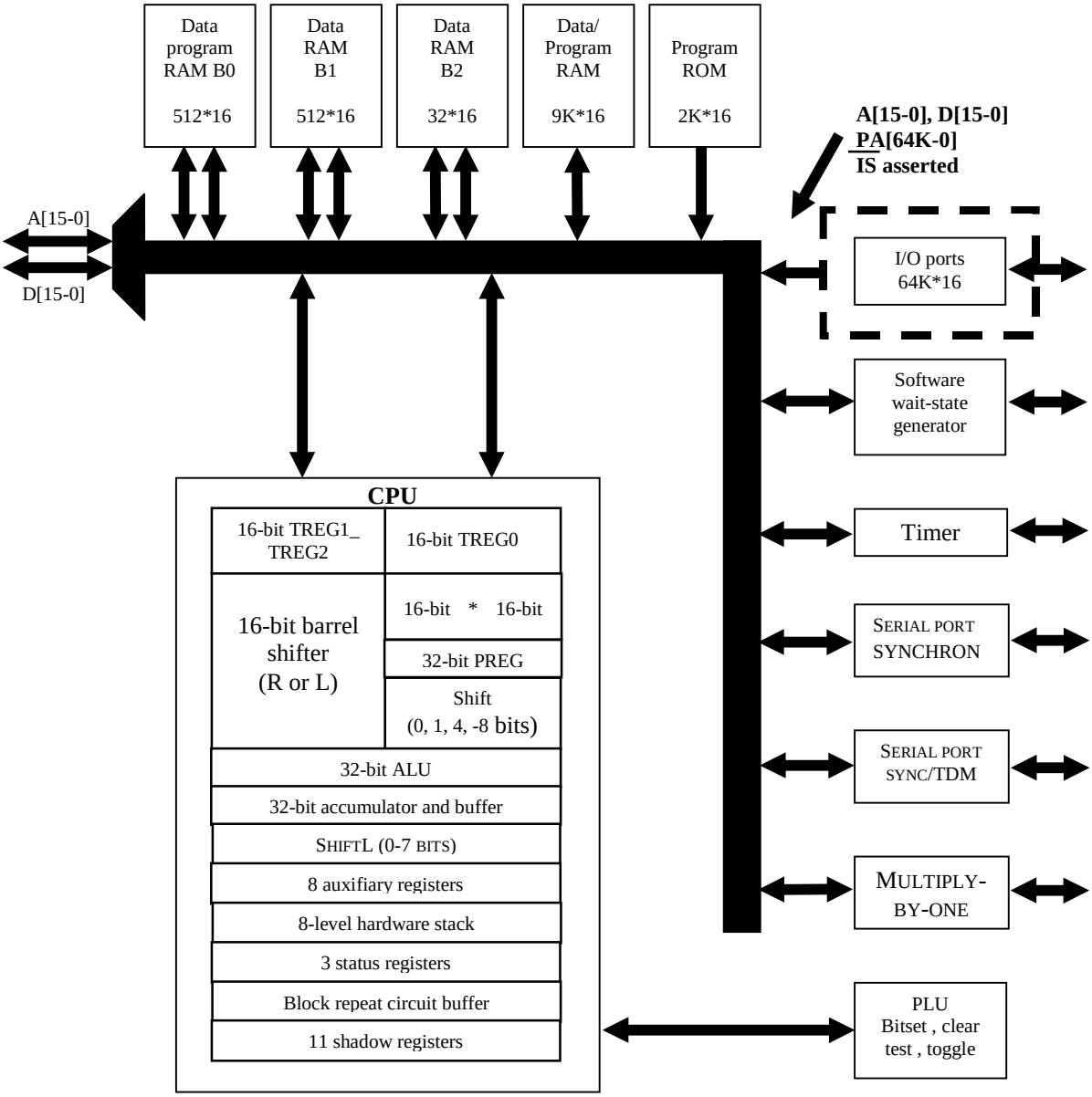


Figure 3.3 : Blocs fonctionnels du TMS320C50

B. Configuration du plan mémoire du DSP

L'espace mémoire du DSP peut être l'objet de plusieurs configurations, ce qui permet l'adaptation du processeur aux besoins spécifiques de l'application. Cependant, il y a deux caractéristiques figées dans le cas de l'utilisation de THESIC : la configuration de la MMI et la position de la mémoire EEPROM de démarrage. La figure 3.4 montre l'espace mémoire du DSP adopté pour implanter le modem.

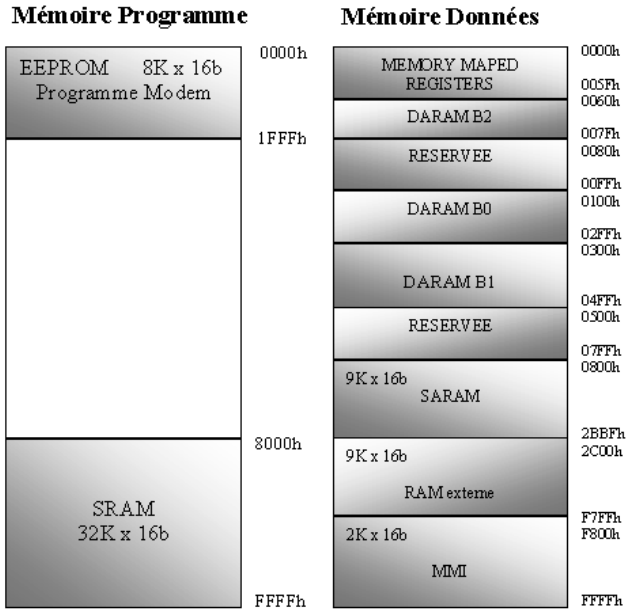


Figure 3.4 : Plan mémoire du DSP pour l'application MODEM

Le plan mémoire programme du DSP comporte principalement une mémoire EEPROM, où sera stocké le programme d'application du DUT, le modem dans notre cas. De plus, 32K de mots (16 bits) de mémoire externe sont configurés entre les adresses 8000h et FFFFh, zone qui sera utilisée pour stocker les paramètres du modem et les variables de la procédure d'injection de fautes. Quant à la mémoire données, elle est répartie ainsi:

- Les *Memory Mapped Registres* (MMR), les registres du DSP, ils sont situés entre l'adresse 0000h et l'adresse 005Fh.
- Les 1056 mots de mémoire DARAM (*Dual Acces RAM*) sont partagés sur trois blocs : les 512 mots du bloc 0 (B0), dans lesquels on stocke les variables d'état du récepteur et de l'émetteur. Quant aux paramètres de l'Algorithme de Viterbi (voir Annexe 3.1), ils sont stockés dans le bloc 1 (B1) situé entre les adresses 0300h et 04FFh. Le dernier bloc B2 est utilisé pour garder les variables d'état nécessaires à certaines procédures du programme (voir figure 3.3).
- Le bloc de mémoire SARAM (*Single Acces RAM*), les 9K x 16bit de mémoire, est configuré en tant que mémoire de données. Les variables des différents modules de réception et transmission y sont stockées.
- Le buffer des données d'entrée et de sortie, ainsi que les variables nécessaires pour faire le test et celles de l'injection sont stockées dans la mémoire externe définie entre les adresses 02C00h et 0F7FFh.

IV. L'IMPLANTATION DE LA METHODE D'INJECTION DE CEU

A. Procédure d'injection de CEU

Pour injecter des CEU au cours de l'exécution du modem par le TMS320C50 nous avons choisi une variante simplifiée de la stratégie générale présentée dans le chapitre 2. En effet, du fait que le DSP contient essentiellement un seul type de cible des CEU, des bits de la mémoire interne, il devient attirant la détermination aléatoire de la cible (adresse et bit à perturber) par l'exécution du programme d'interruption lui-même. Nous rappelons que cette étape, dans le cas du 8051, était effectuée par la carte mère. Les figures 3.5 et 3.6 donnent les principales étapes de l'implantation de l'injection de CEU. Les différences avec la méthode exposée dans la figure 2.4 de la section IV du chapitre 2, sont écrites en caractère italique, et les changements nécessaires sont grisés.

Côté carte mère

- **Etape n°1:** Calcul du temps d'exécution de l'application exécutée par le DUT côté carte fille comptage dans un Timer du nombre de cycles machine se déroulant entre l'envoi du signal Reset (lancement de l'application) et la réception du signal INT (indiquant la fin du programme).
- ***Etape n°2 :*** *Détermination des paramètres du code CEU*……Faite par la CARTE FILLE
- *une cible (position mémoire ou registre),*
- *un bit de cette position mémoire*,
- un instant d'injection.
- ***Etape n°3 :*** *Placement du code CEU dans la MMI afin qu'il puisse être exécuté,*…….. Eliminée (le code CEU est directement implanté dans la procédure de l'interruption)
- **Etape n°4 :** Lancement de l'application sur la carte fille (envoi du signal Reset) et du décomptage du Timer,
- **Etape n°5 :** Activation du signal INT_RESET,
- **Etape n°10 :** Récupération, traitement et affichage des résultats.

Figure 3.5 : Changement dans les étapes de la procédure d'injection de CEU côté carte mère

Côté carte fille

- **Etape n°4 :** Réception de l'ordre d'exécuter le programme d'application (réception du signal RESET),
- **Etape n°5 :** Réception de l'ordre d'interruption INT_RESET : la carte fille arrête son programme d'application et pointe vers le code CEU,
- *Etape n°6 : Exécution du code CEU implanté dans la MMI*, ……..Le code CEU est implanté dans l'EEPROM de la carte fille. En effet, le code CEU détermine d'abord l'adresse et le bit à perturber, puis injecte l'erreur. Une fois la cible du CEU sélectionnée (à l'aide du programme de génération de nombres aléatoires décrit dans le chapitre I) les informations concernant l'adresse, le contenu de la cible et le bit modifié, sont inscrites dans la MMI pour le traitement ultérieur des résultats d'essais d'injection d'upsets
- **Etape n°7 :** Poursuite du programme d'application jusqu'à la fin,
- **Etape n°8 :** Comparaison des résultats obtenus à ceux attendus,
- **Etape n°9 :** Envoi du signal INT pour indiquer la fin du programme.

Figure 3.6 : Changement dans les étapes de la procédure d'injection de CEU côté carte fille

Le système carte mère/carte fille utilisé pour effectuer les tests d'injection est illustré par la photo donnée dans la figure 3.7. La carte mère est placée en dessous et est connectée à un PC par une liaison série.

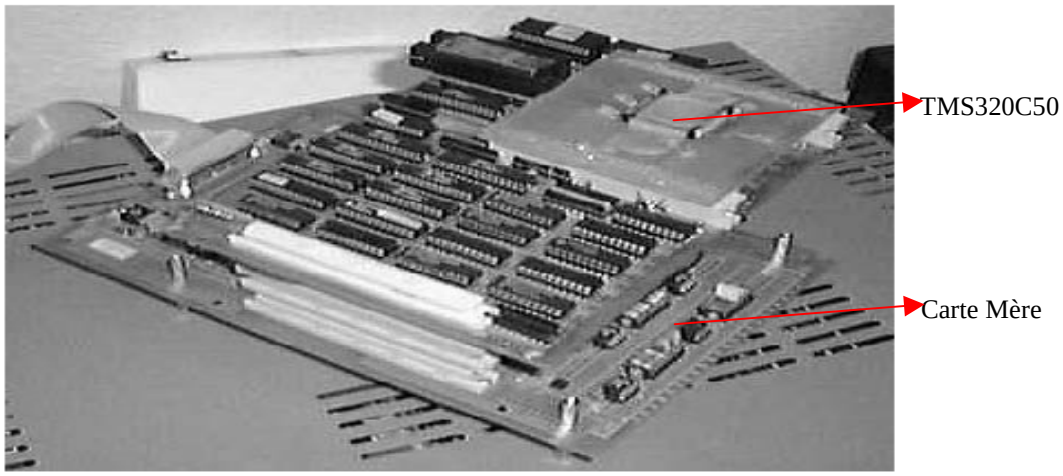


Figure 3.7 : Architecture utilisée pour injecter des CEU sur un MODEM

B. Cibles de l'injection de CEU

Le programme d'injection de CEU doit a priori pouvoir perturber n'importe quelle cible des CEU : registres, zones mémoire utilisées pour stocker des variables, des données ou des résultats de l'application modem. L'existence de zones de mémoire réservées dans le DSP, non accessibles par programme, ne permet pas l'injection de CEU dans la totalité de la zone sensible. En plus, le TMS320C50 possède une pile matérielle dont l'emplacement n'est pas connu pour l'utilisateur. Ceci peut constituer une restriction considérable dans l'étude des conséquences des erreurs. En effet, comme montré dans le chapitre 1, la pile est le seul moyen permettant d'injecter des CEU sur des registres de contrôle critiques tels que PC et SP.

La mémoire du TMS320C50 peut se diviser en quatre zones :

1. La mémoire externe, qui ne sera pas l'objet d'injection de CEU, vue qu'elle ne sera pas exposée aux faisceaux de particules lors des tests sous radiation. Des grandes portions de cette mémoire sont inoccupées.
2. La mémoire interne : accessible par adressage direct et donc modifiable par des CEU dans sa totalité. Elle est répartie en :
   - MMR : région utilisée pour mapper des registres.
   - DARAM B2
   - DARAM1 B0
   - DARAM1 B0
   - DARAM B1
   - SARAMD

   } zones contenant les variables du programme modem.

- ZONE NON UTILISEE : Afin de réduire la durée des essais d'injection de CEU, nous avons décidé de ne pas affecter les zones de mémoire non utilisées par le modem.

La figure 3.8 illustre la répartition des cibles de CEU.

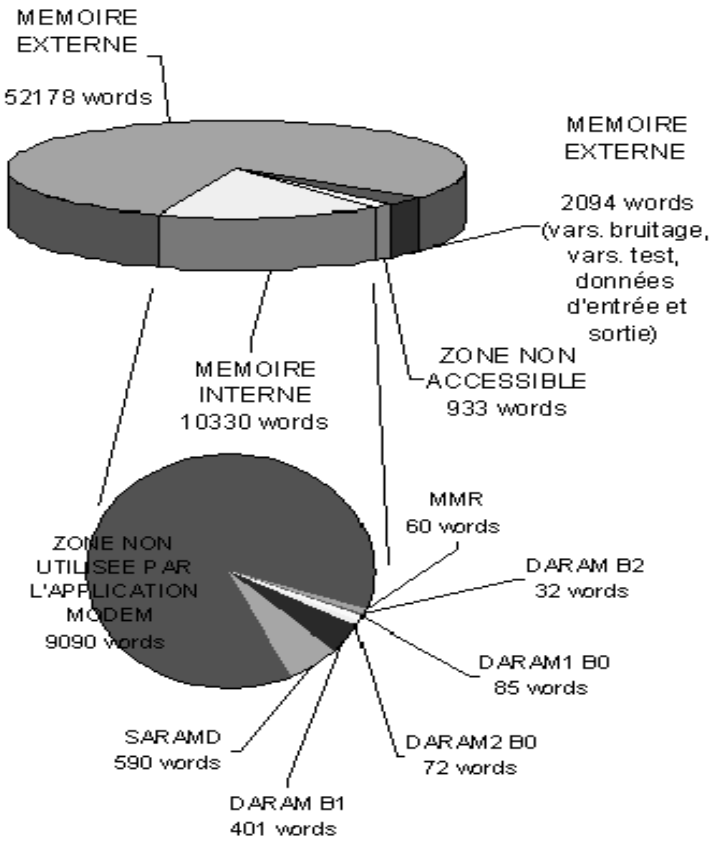


Figure 3.8 : Possibilités d'injection de CEU dans les zones mémoires du DSP

V. RESULTATS DES ESSAIS D'INJECTION DE CEU SUR LE DSP

A. Introduction

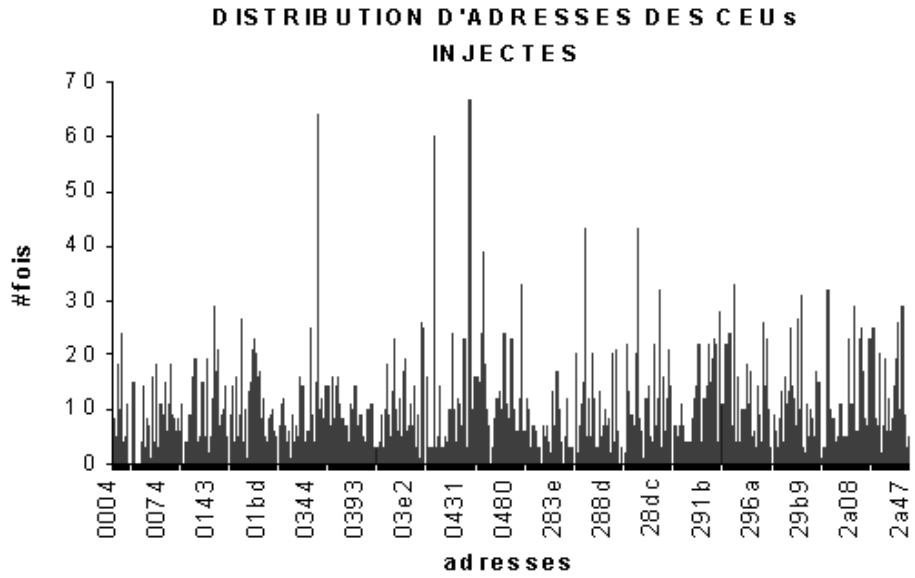
L'idée à la base de la méthode de prédiction du taux d'erreurs présentée précédemment dans le § V du chapitre 1 est qu'à partir d'informations sur le nombre d'erreurs obtenu lors des sessions d'injection de CEU, et de la section efficace face aux SEU déterminée à l'aide d'un test statique exécuté sous radiations, peut être prédit le taux d'erreursd'une application fonctionnant sous radiations. Une première validation de cette méthode, avec deux accélérateurs de particules de test sous ions lourds, a été effectuée avec succès dans le cas d'une architecture digitale bâtie autour du 80C51. L'objectif de l'expérience avec le modem est d'étendre le champ d'application de cette méthode à des architectures digitales utilisant des processeurs complexes et exécutant des modules dérivés d'un programme de vol.

Le modem implanté sur le processeur de traitement de signal TMS320C50, constitue un bon véhicule de test des potentialités de cette technique. Le choix du DSP comme véhicule de cette validation a été motivé d'une part, par le fait que ce processeur possède seulement un type de cible de CEU (la mémoire interne), simplifiant les essais. D'autre part, l'inaccessibilité de l'injection de CEU dans quelques registres et zones mémoires de ce processeur constitue une limitation de cette technique, son étude permettra de mettre en œuvre l'évaluation, dans un tel cas, de la précision du taux d'erreurs prédit.

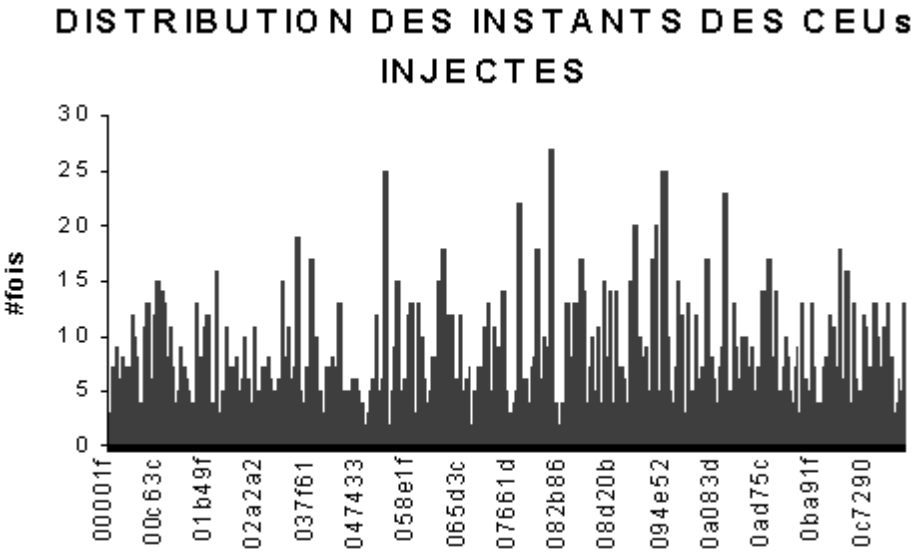
Dans cette section, nous exposerons les résultats d'une campagne d'injection de CEU effectuée sur le modem implanté sur la carte THESIC développée autour du TMS320C50. Le but de ces essais était l'estimation du nombre moyen de CEU, ayant lieu dans les zones sensibles du DSP, nécessaires pour provoquer un dysfonctionnement du programme exécuté, le programme modem dans notre cas. Les résultats déduits de la technique d'injection de fautes seront comparés aux données obtenues lors d'une campagne de radiations du TMS320C50 exécutant le programme modem, effectuée à l'aide d'un accélérateur de particules, le Tandem Van der Graff de l'IPN.

B. Présentation des résultats d'injection de CEU

Trente trois essais d'injection de CEU ont été effectués, chacun avec une durée différente pour exploiter au maximum les possibilités du programme de génération de nombres aléatoires [66]. Les graphiques 3.1 et 3.2 montrent respectivement les distributions de positions mémoire affectées par des CEU et des instants d'injection. La qualité de *l'aléatoireté* semble correcte. Les « trous » (cibles jamais modifiées) dans le Graphique 3.1 correspondent aux zones mémoires non utilisées par le modem et aux zones réservées non accessibles aux CEU. L'analyse du Graphique 3.2 permet de constater que tous les cycles machine du programme ont été l'objet d'occurrence de CEU.



Graphique 3.1 : Fréquence de répétition de chaque adresse (cibles de CEU)



Graphique 3.2 : Fréquence de répétition de chaque instant d'injection de CEU

La table 3.1 donne toutes les informations des tests d'injection de CEU : durée, nombre de CEU injectés, nombre d'erreurs du modem détectées et taux d'erreurs associés. Puis, dans le cas où cela fut possible, est indiqué un diagnostic grossier concernant la phase du modem durant laquelle s'est produit le CEU : apprentissage, données ou vérification. Dans tous les autres cas, les erreurs ont été classées sous l'appellation « incertitude ».

L'analyse du nombre d'erreurs produites pour des essais ayant une durée de session d'injection proche, pourrait faire penser à une contradiction. En effet, le nombre de CEU injectés semble pour certains cas inconsistant avec leur durée. Par exemple, lors du test n° 15, d'une durée de 2h 1min, 1828 CEU ont été injectés, alors que durant le test n°

17, ayant une durée de 17 minutes de plus, seuls 1516 CEU ont été injectés. Cette différence est due principalement au temps d'affichage des erreurs très dépendant de la nature de l'erreur provoquée. Dans le cas des rafales d'erreurs l'affichage peut en effet prendre plusieurs secondes.

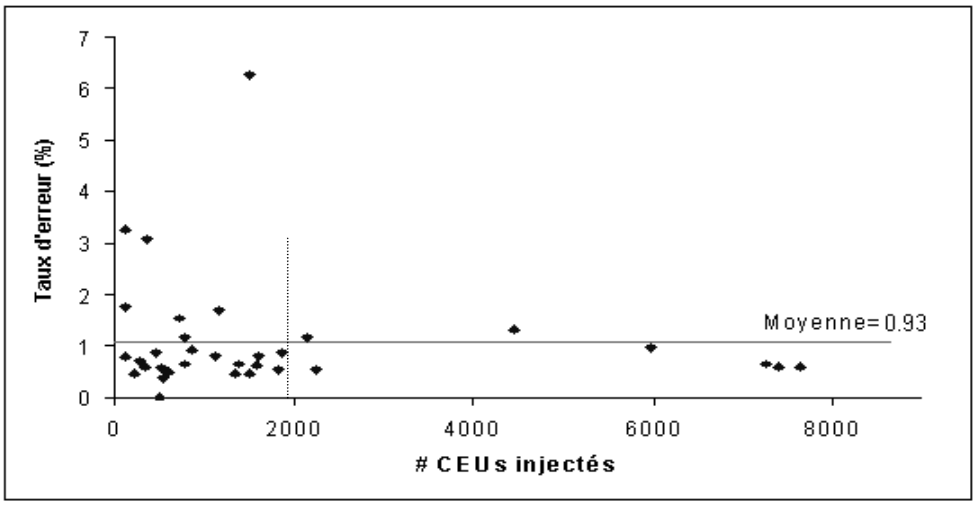
Table 3.1 : Table de relation des différents événements observés

Test	Durée d'injection	# CEU injectés	# Erreurs	Taux Erreur (%)	T.E. Ph. Appr. (%)		T.E. Ph. Don. (%)		T.E. Ph. Vérif. (%)		Incertitude (%)
1	35min	521	3	0,5758	0/2	0	1/2	50	0/1	0	0
2	52min	784	5	0,6378	0/2	0	1/1	100	0/8	0	0
3	8min	115	2	1,7391	0/1	0	1/1	100	0/0	0	0
4	8h 6min	7260	46	0,6336	2/45	4,44	23/38	60,53	9/57	15,79	7,14
5	8min	123	4	3,2520	0/1	0	4/4	100	2/3	66,67	25
6	52min	786	9	1,1450	0/9	0	3/4	75	1/4	25	0
7	1h 30min	1384	9	0,6503	0/10	0	4/6	66,67	1/5	20	4,76
8	14min	219	1	0,4566	0/2	0	0/0	0	0/2	0	0
9	35min	547	2	0,3656	0/7	0	1/1	100	0/1	0	0
10	8h 12min	7388	43	0,5820	1/48	2,08	23/33	69,70	7/43	16,28	9,41
11	35min	531	3	0,5650	0/4	0	0/2	0	0/1	0	0
12	1h 47min	1610	13	0,8075	0/18	0	6/9	66,67	2/12	16,67	5,13
13	2h 30min	2258	12	0,5314	1/16	6,25	4/6	66,67	1/12	8,33	5,88
14	2h 16min	1874	16	0,8538	1/11	9,09	9/11	81,82	3/11	27,27	12,12
15	2h 1min	1828	10	0,5470	0/14	0	4/9	44,44	1/13	7,69	0
16	1h 28min	1345	6	0,4461	0/17	0	2/2	100	2/10	20	3,45
17	2h 13min	1516	7	0,4617	0/11	0	4/7	57,14	1/12	8,33	3,33
18	40min	610	3	0,4918	0/5	0	0/0	0	0/2	0	0
19	1h 13min	1117	9	0,8057	0/11	0	5/6	83,33	2/9	22,22	7,69
20	22min	290	2	0,6897	1/13	7,69	2/2	100	1/5	20	10,00
21	1h 35min	1175	20	1,7021	2/17	11,76	10/21	47,62	7/18	38,89	13,64
22	38min	494	0	0,0000	0/14	0	0/0	0	0/7	0	0
23	37min	468	4	0,8547	0/5	0	1/6	16,67	0/3	0	0
24	9min	128	1	0,7813	0/2	0	1/2	50	0/0	0	0
25	30min	358	11	3,0726	2/12	16,67	7/8	87,50	2/7	28,57	14,81
26	7h 54min	5966	58	0,9722	7/159	4,40	39/48	79,17	13/74	17,57	6,76
27	25min	332	2	0,6024	2/17	11,76	2/3	66,67	0/4	0	8,33
28	58min	719	11	1,5299	0/20	0	7/10	70	0/9	0	0
29	1h 6min	863	8	0,9270	2/31	6,45	5/7	71,43	0/6	0	4,55
30	5h 53min	4433	59	1,3309	4/116	3,45	33/50	66	9/47	19,15	5,71
31	2h 5min	1585	10	0,6309	0/28	0	5/12	41,67	0/10	0	0
32	2h 5min	1513	95	6,2789	10/36	27,78	58/70	82,86	3/23	13,04	5,68
33	2h 49min	2154	25	1,1606	4/44	9,09	15/21	71,43	2/18	11,11	7,23
Moyenne				1,0764	3,66		62,82		12,20		

La première observation qui ressort de l'analyse de la table 3.1 concerne l'absence d'erreurs de type perte de séquencement. De telles erreurs sont en principe susceptible de se produire suite à des upsets sur le compteur de programme ou sur la pile par exemple. Rappelons que la méthode d'injection de CEU dans le PC est faite par corruption de la pile dont l'emplacement est inconnu dans le cas du DSP étudié. La deuxième remarque concerne le test n° 22. En effet, aucun CEU parmi les 494 CEU injectés n'a provoqué de dysfonctionnements dans le modem. Bien qu'exotique, un tel cas peut en effet se produire, suite à une « bonne chance » dans le tirage aléatoire des cibles et des instants d'injection des CEU. Enfin, le plus important résultat issu de la table 3.1 est l'évaluation du nombre

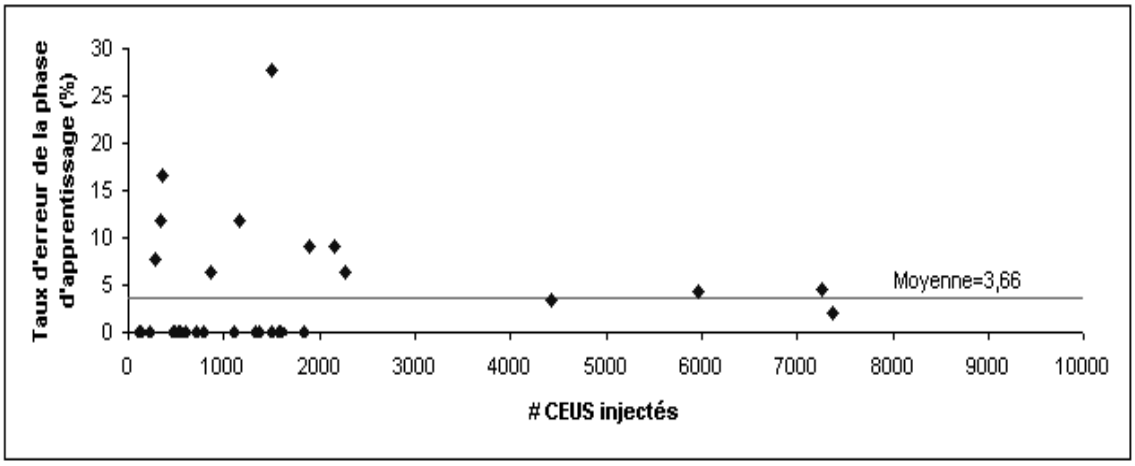
de CEU nécessaires pour causer une erreur dans l'opération du modem. La moyenne de ces taux de « CEU efficaces » permet l'évaluation de la sensibilité aux SEU de l'application.

D'après cette table, on conclut qu'approximativement 1 parmi 100 CEU injectés va provoquer une défaillance du système. L'analyse de cette table révèle qu'il y a un nombre minimum de CEU qui doivent être injectés, pour éviter l'obtention de points exotiques comme, par exemple, ceux correspondants aux tests numéros 3, 5, 22, 23 et 33. Alors, un deuxième calcul a été fait pour estimer plus précisément le taux d'erreurs, en ne prenant en compte que les tests injectant plus de 2000 CEU (contrainte représentée pointillés sur le graphique 3.3). La nouvelle moyenne obtenue, estime le taux d'erreursà 0,93. Le graphique 3.3 représente la distribution du taux d'erreursen fonction du nombre de CEU injectés.

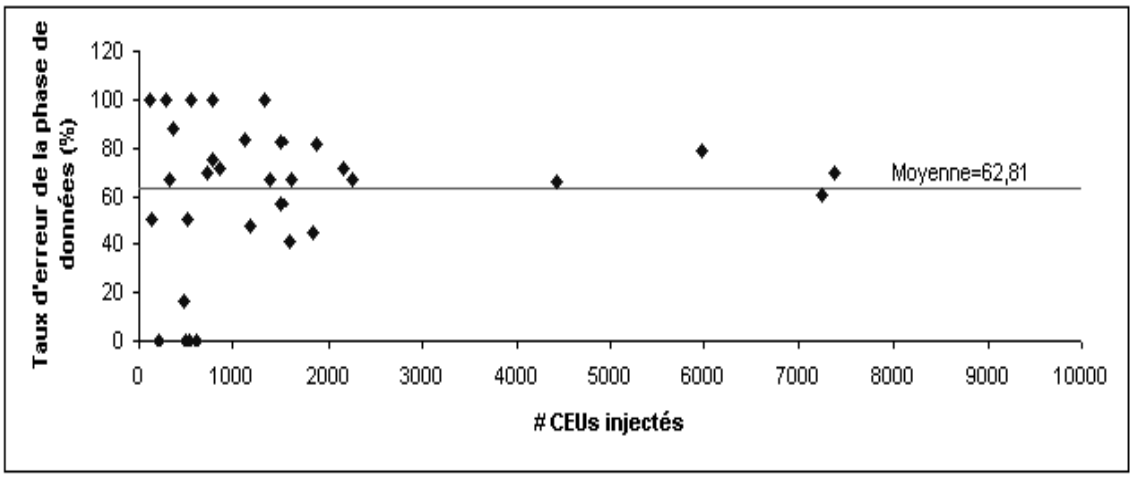


Graphique 3.3 : Evolution de la probabilité d'erreur

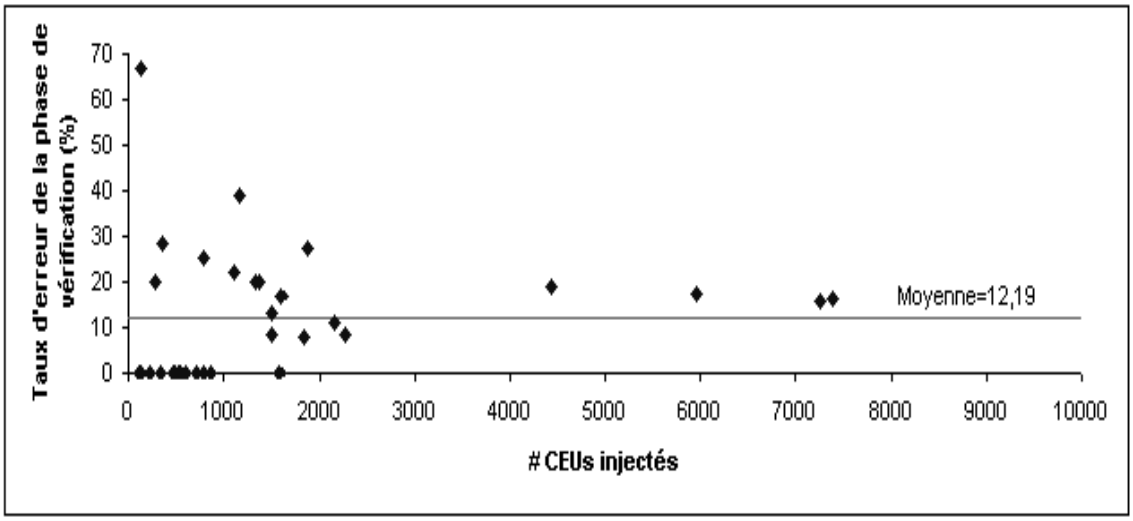
Enfin, les graphiques 3.4 à 3.6, représentent le taux d'erreurs dues aux CEU injectés sur les variables de contrôle des différentes phases du modem. Rappelons que ces erreurs sont détectées par les valeurs des flags de contrôle ajoutées au programme original. Ces graphes montrent les sensibilités estimées face aux CEU survenant pendant l'exécution des phases du modem. Une conclusion immédiate tirée de ces graphes est que la phase d'apprentissage est la moins sensible, alors que celle de données est la plus sensible (63% des CEU conduisent à des erreurs dans le fonctionnement du modem).



Graphique 3.4 : Evolution des erreurs en phase d'apprentissage



Graphique 3.5 : Evolution des erreurs en phase de données



Graphique 3.6 : Evolution des erreurs en phase de vérification

C. Analyse des modes d'erreurs identifiées

Le programme modem présente deux types de variables : les variables statiques, qui sont essentiellement des paramètres utilisés pour configurer le mode de fonctionnement du modem, et les variables dynamiques, dont le contenu varie tout au long de l'exécution du programme. Alors, si un CEU affecte une variable statique (telle NL : le nombre d'échantillons par bit ou REG3 concernant la configuration générale de la transmission et la réception), il aura de grandes chances de causer des rafales d'erreurs. Par contre, la conséquence d'un CEU sur une variable dynamique n'affectera pas forcément le fonctionnement du programme. En effet, à l'instant de l'injection du CEU, la variable peut ne pas être en cours d'utilisation et qu'elle sera écrite plus tard avec une nouvelle valeur. Celle-ci est l'une des raisons qui expliquent que la probabilité d'erreur globale du modem soit si faible.

Un autre motif de la faible sensibilité à des upsets est lié à l'instant d'occurrence du CEU. Si, par exemple, un CEU est injecté au début du programme, quand la configuration du processeur est à peine entamée, il est normal qu'il ne cause pas d'erreurs, parce que la variable ou le registre affectés seront remplis ultérieurement avec leur valeur définitive. Un tel CEU n'aura donc aucune répercussion sur le déroulement du programme. L'autre cas extrême est quand le CEU est injecté presque à la fin de l'exécution du programme quand la plupart des variables ont déjà été utilisées.

Les différents dysfonctionnements du modem observés sont :

- DES RAFALES D'ERREURS : Le CEU a eu lieu sur un paramètre du programme ou bien sur une variable importante pour le fonctionnement du modem ou simplement sur une variable qui était en train d'être utilisée à cet instant là. Ce cas apparaît par exemple lorsqu'un tel CEU a modifié la valeur des constantes des PLLs, des pointeurs des différents buffers utilisés par le modem, etc.
- DES PERTURBATIONS DANS LE MODE D'APPRENTISSAGE (Graphique 3.4) : Quand le mode d'apprentissage est fini, il y a quelques variables, comme les constantes des PLLs ou la phase du récepteur, qui doivent avoir des valeurs précises. Si le FLAGAPPR est différent de 0000h, il indique que ces valeurs ont changé. Cependant, bien que cela montre qu'un CEU est survenu sur une variable pendant le mode d'apprentissage, il ne va pas toujours causer un mauvais comportement du programme, mais seulement une durée plus longue de l'apprentissage ou des valeurs différentes de la phase, bien que proches de celle attendue de C000h. Ces erreurs sont en grande partie tolérées.
- DES PERTURBATIONS DANS LE MODE DE DONNEES (Graphique 3.5) : Cela se produit quand le CEU corrompt des données avant d'arriver à la fin du procédé de réception.

- DES PERTURBATIONS DANS LE MODE DE VERIFICATION (Graphique 3.6) : A l'aide du FLAGVERIF on vérifie s'il y a eu des erreurs sur des constantes du programme, les corriger et vérifier leur validité. C'est une procédure de détection et de correction d'erreurs mais à une position fixe du programme. Ces erreurs sont en partie tolérées.
- DES VALEURS « ETRANGES » DES DIFFERENTS FLAGS DE CONTROLE : Comme il a déjà été dit, le programme dispose de trois flags qui permettent de suivre le déroulement de l'exécution. S'il n'y a pas un comportement incorrect du programme, FLAGAPPR doit être égal à 0000h, FLAGDON à FFFFh et FLAGVERIF à 0000h. FLAGDON indique le nombre d'erreurs résultant et donc il peut avoir n'importe quelle valeur. Par contre, FLAGAPPR et FLAGVERIF ont des valeurs limites. Le premier, par exemple, ne peut pas excéder 03FFh. Dans les différents tests réalisés, des valeurs incorrectes de ces flags ont pu être observées. La cause de leur apparition est l'occurrence d'un CEU sur une variable qui provoque une perte de séquencement du système pendant une boucle de programme. Dans tous les cas traités, après un Reset du modem, le programme a été capable de fonctionner correctement.

D. Essais sous radiations

Pour valider la qualité de prédiction des taux d'erreurs dues aux radiations à partir des taux de CEU résultant d'essais d'injection d'après la technique de prédiction proposée au chapitre 1, nous avons effectué des tests sous radiations au Tandem Van der Graaff de l'IPN. Durant cette campagne, effectuée en avril 2000, le DSP a été exposé à un faisceau de Chlore (LET= 12,7 Mev/mg/cm2). La section efficace résultante σ_{SEU} a été évaluée comme étant 1,73 x 10-2 cm²/composant (336 erreurs obtenues lors d'une exposition à 19445 particules). A partir des essais d'injection, le taux d'erreurs CEU, τ_{CEU} , a été estimé à 0.93%. Le produit de la section efficace et du taux de CEU est l'estimateur de la section efficace globale τ_{SEU} du programme modem dû aux radiations calculé selon la formule (1.9) du § V chapitre I :

$$\tau_{SEU} = \sigma_{SEU} * \tau_{CEU} \qquad (1.9)$$

Vu que les erreurs sur la partie restante non utilisée sont par définition sans effet sur le fonctionnement du modem, les injections de CEU ont été faites uniquement sur la portion utilisée de la mémoire interne. Ainsi, pour évaluer le taux d'erreursdu modem, nous devons prendre en considération dans le calcul de la section efficace, le taux d'occupation de la mémoire interne du DSP pour l'application modem. Ce taux étant de 12 %, la sensibilité du

modem peut être estimée comme étant le produit de la section efficace correspondant à la mémoire interne occupée par le modem, par le taux de CEU du modem:

$$\tau_{SEU}(Modem) = 1{,}73.10^{-2} * 0{,}12 * 0{,}93.10^{-2} = 1{,}9.10^{-5}\, cm^2 \text{ / composant}$$

Pour valider cette prédiction nous avons effectué des essais sous radiations avec le faisceau de Cl durant lesquels le DSP exécutait le programme modem. Pour être dans des conditions semblables aux tests d'injection de CEU (dans lesquels un seul upset est introduit à chaque exécution d'une boucle du programme) les tests sous radiations ont été réalisés avec des flux relativement faibles (quelques centaines de particules par cm² et par seconde). Les résultats de deux de ces essais sont donnés dans la table 3.2.

Table 3.2 : Résultats des radiations avec un faisceau de Chlore

	Test Radiations n°1	**Test Radiations n°2**	**Moyenne mesurée**	**Prédiction**
Section efficace de résultats (1)	$2{,}38\ 10^{-5}$	$1{,}57\ 10^{-5}$	$2\ \ 10^{-5}$	$1{,}9\ \ 10^{-5}$
Section efficace de perte de séquencement (2)	$2{,}38\ 10^{-5}$	$1{,}43\ 10^{-5}$	$1{,}9\ \ 10^{-5}$	0
Section efficace (1)+(2)	$4{,}76\ 10^{-5}$	$3\ \ 10^{-5}$	$3{,}9\ \ 10^{-5}$	$1{,}9\ \ 10^{-5}$

La table 3.2 montre que la prédiction de la section efficace du modem à partir de la méthode proposée est bonne, malgré l'existence dans le DSP étudié de zones inaccessibles aux CEU. En effet, l'erreur relative commise par la prédiction, si on se restreint aux erreurs de résultat dans le modem, est de 5/100. Si nous considérons la prédiction par rapport à la section efficace globale (incluant erreurs de résultats et les pertes de séquencement), l'erreur relative est dix fois plus grande.

Une autre manière de présenter les résultats consiste à calculer le taux d'upsets tolérés. La table 3.3 donne les tolérances mesurées et prédites par l'approche CEU. Dans cette table, il est clairement mis en évidence la faible sensibilité aux upsets du programme modem.

Table 3.3 : Comparaison des résultats des radiations et de l'injection

	Radiations	**Injection de CEU**
% Erreurs de résultats	0,96	0.93
% Perte de séquencement	0,92	0
% Erreurs tolérées	98,12	99.07

VI. CONCLUSION

La technique d'injection de CEU a été confrontée à l'expérimentation dans le cas d'une application dérivée d'un programme de vol : un modem implanté sur un processeur de traitement de signal, le TMS320C50 de Texas Instruments. Deux aspects ont été mis en évidence et particulièrement détaillés dans ce chapitre : l'étude du comportement de ce modem en présence de fautes et la prédiction du taux d'erreurs de cette application face aux SEU.

L'étude du fonctionnement de ce modem en présence de CEU a mis en évidence la grande robustesse face aux SEU de cette application : en effet il a été prouvé qu'elle est tolérante aux fautes approximativement à 99%. L'étude des types de mauvais comportements induits par les erreurs injectées, montre la possibilité qu'offre la technique d'injection de CEU pour l'élaboration des diagnostics détaillés d'une application face aux erreurs injectées afin de pouvoir remédier à certaines de ces erreurs. Ce nouvel aspect de la méthode d'injection de CEU pourra être utilisé dans le domaine de la tolérance aux fautes.

Les données obtenues suite à la confrontation du taux d'erreurs prédit à partir des sessions d'injection et celui mesuré durant la campagne de radiations effectuée à l'aide de l'accélérateur de particules, mettent en évidence une bonne corrélation entre prédiction et mesure du taux d'erreurs du modem implanté sur le TMS320C50. Le chapitre suivant montre une nouvelle application de l'approche d'injection de CEU à des processeurs, durant l'exécution d'applications de vol, conçues dans le cadre d'un projet satellite hispano-argentin.

CHAPITRE 4

ESTIMATION DES TAUX D'ERREURS D'ARCHITECTURES BATIES AUTOUR DE PROCESSEURS COMPLEXES

- Microprocesseur TS6833216A
- DSP ADSP21060 : SHARC

I. INTRODUCTION

Les travaux présentés dans les chapitres 2 et 3 montrent l'efficacité de la technique d'injection de CEU pour la prédiction du taux d'erreurs d'une application dans le cas d'un processeur simple, le microcontrôleur 80C51 et d'un processeur de traitement de signal le TMS320C50. L'objectif de ce chapitre est l'application de cette méthode à d'autres types de processeurs ayant une plus grande complexité. Nous avons sélectionné le processeur de traitement de signal ADSP21060 de Analog Devices, nommé aussi SHARC et le microprocesseur TS6833216A de Motorola. Notre choix s'est fondé sur deux aspects : la complexité de leurs architectures internes et le fait qu'ils soient tous les deux inclus dans des instruments d'un projet satellite nommé CESAR de l'INTA (Institut National des Techniques Aérospatiales) en Espagne.

Les résultats expérimentaux des expériences de radiation et des sessions d'injection de CEU, ont été obtenus lors de l'exécution de programmes benchmark ainsi que lors de l'exécution des modules des applications de vol du projet CESAR, sur des architectures digitales bâties autour de ces deux processeurs. La comparaison des taux d'erreurs obtenus avec les applications de vol à ceux dérivés des stratégies généralement utilisées va nous permettre de conclure sur les déviations résultantes dans les taux d'erreurs mesurés et la surestimation ou sous-estimation des taux d'erreurs par rapport à l'application finale.

Ce chapitre est organisé comme suit : le principe de fonctionnement des instruments du satellite hispano - argentin CESAR sera brièvement présenté dans le paragraphe II. La méthodologie d'implémentation et de

qualification sous radiations des architectures bâties autour des deux processeurs considérés sera détaillée dans le § III. La stratégie d'injection de CEU dans les zones sensibles du processeur SHARC sera présentée dans le § IV. Les expériences d'injection de CEU seront décrites dans la section V. Dans ce chapitre, nous avons réservé la section VI, pour l'étude d'une application de calcul des facteurs de sensibilités et ceci pour le microprocesseur TS6833216A.

Les résultats dérivés des sessions d'injection de fautes ainsi que des tests de radiations seront discutés dans le paragraphe VII. A partir de la synthèse de ces résultats et de ceux issus des tests de radiation à l'aide de différents types d'ions lourds, seront prédits les taux d'erreurs sous radiation d'un instrument dont le fonctionnement est contrôlé par le microprocesseur TS6833216A. Ceci permettra encore une fois d'évaluer les performances de la stratégie de prédiction du taux d'erreurs pour des cas complexes. Des aspects préliminaires de ces travaux ont fait l'objet d'une présentation au Workshop RADECS 2000 [67] et les résultats finaux ont été soumis à l'édition 2001 de la conférence NSREC (Nuclear and Space Radiation Effects Conference) [68].

II. PROJET CESAR

Le projet CESAR est une mission satellite d'observation de la surface terrestre, développée en coopération entre deux agences spatiales, INTA en Espagne et CONAE (Commission Nationale pour Activités Spatiales) en Argentine. La mission avec une date de lancement en 2002/2003, consiste en la conception, la construction, la mise en fonctionnement, le lancement d'un satellite ayant un poids approximatif de 400 Kg et la mise à jour d'une base terrestre en Espagne et en Argentine pour recevoir et traiter les données transmises par le satellite CESAR [69]. Ses premiers objectifs seront la cartographie, la topographie, les études thermiques et géophysiques avec un équipement comportant essentiellement (Figure 4.1) :

- Une caméra panchromatique (IRIS): dans le champ visuel du spectre. Elle sera utilisée pour la cartographie et la topographie.
- Une caméra Multi-spectre (MUS): avec six bandes dans le champ visuel et le proche infrarouge. Elle sera utilisée pour des applications de ressources naturelles.
- Un spectromètre (MEGA): utilisé pour la mesure de la concentration des gaz atmosphériques impliqués dans le procédé de destruction de la couche d'ozone.
- Une caméra panchromatique à haute résolution (PAS): dans le champ visuel du spectre, mais d'une très haute résolution. Elle sera utilisée pour prendre des images de nuages et du vortex polaire durant la nuit.

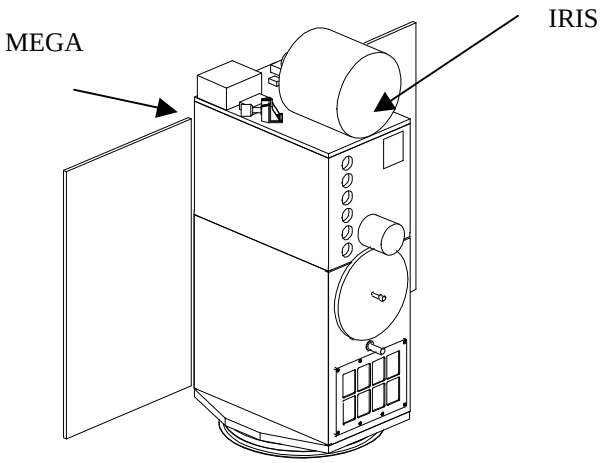


Figure 4.1 : Configuration du satellite CESAR

L'institut spatiale INTA est responsable de la conception et du développement de deux instruments du projet satellite CESAR : la caméra panchromatique IRIS et le spectromètre MEGA. De point de vue conception électronique, les deux bénéficient des technologies employées dans les architectures commerciales développées auparavant pour diverses applications.

Le programme CESAR a spécifié que de point de vue qualification, les composants électroniques doivent respecter la norme MIL-STD-883 B et les applications de vol étudiées concernent principalement le contrôle du spectromètre MEGA et la caméra IRIS. Pour ce faire, deux versions militaires de processeurs et de convertisseurs analogique / digital ont été sélectionnées :

- TS6833216A, (THOMSOM-CSF) [70]. C'est un microprocesseur de 32 bits fonctionnant à la fréquence 16 MHz. Son rôle est de contrôler l'unité centrale du spectromètre MEGA.
- AD 677, (Analog Devices). C'est un convertisseur A/D ayant pour résolution 16 bits et 100 kiloéchantillons par seconde avec une interface série, utilisé pour la lecture d'image du détecteur (1024 pixels) du spectromètre MEGA.
- ADSP21060, (Analog Devices) [71]. C'est un processeur 40-MIPS, qui sera utilisé comme une unité centrale de traitement de contrôle et d'acquisitions d'images provenant de la caméra IRIS.
- AD 9040, (Analog Devices). C'est un convertisseur avec une fréquence d'échantillonnage de 40 MHz et une résolution de 10 bits, utilisé pour le saisie des données de la caméra IRIS.
- Deux applications ont été développées autour des deux processeurs le ADSP21060 et le TS6833216A pour le contrôle du fonctionnement des instruments IRIS et MEGA. Les détails de ces deux applications sont données en annexe 4.1.

Ces processeurs dont, à notre connaissance, aucune information sur leurs sensibilités aux radiations n'a été publié auparavant, doivent être testés dans un environnement radiatif afin qu'ils soient qualifiés à ces effets. Pour ce faire, deux cartes filles THESIC ont été conçues et développées autour du processeur ADSP21060 et du microprocesseur TS6833216A pour la qualification de ces deux circuits sous radiation (ions lourds) et pour la réalisation des expériences d'injection de fautes.

III. BANC D'EXPERIMENTATION

En plus de la configuration standard de toute carte fille THESIC, celle bâtie autour du TS6833216A inclut aussi le convertisseur AD 677 nécessaire à l'adaptation des programmes développés pour l'instrument CESAR. Durant les tests de radiation, les stimuli des signaux vont activer le convertisseur, représentant en quelques sortes les mesures acquises en vol. La figure 4.2 représente le schéma bloc de la carte fille TS6833216A et le convertisseur analogique/digital AD 677.

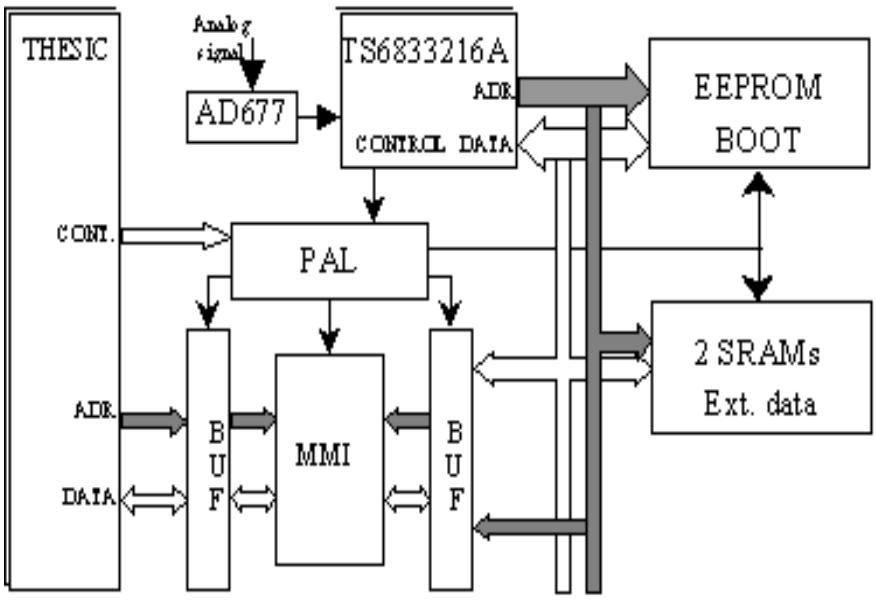


Figure 4.2 : Schéma Bloc de la carte fille TS6833216A

La carte fille SHARC a une architecture similaire à celle du TS6833216A. Etant donnée que le SHARC est un processeur qui nécessite des instructions de 48 bits et des données de 32 bits, la carte fille bâtie autour de ce DSP comporte en plus 6 EEPROM pour le stockage des programmes (ayant un bus de données de 8 bits chacune) et 4 SRAM pour les données. D'un autre côté, le DSP permet d'exécuter des programmes dans sa mémoire interne. Pour ce faire, une EEPROM de 8 bits est nécessaire pour le téléchargement des programmes à partir de cette EEPROM

(de boot) après un Reset du DSP, dans la mémoire interne de ce dernier. Le schéma bloc de la carte fille ADSP21060 est représenté sur la figure 4.3.

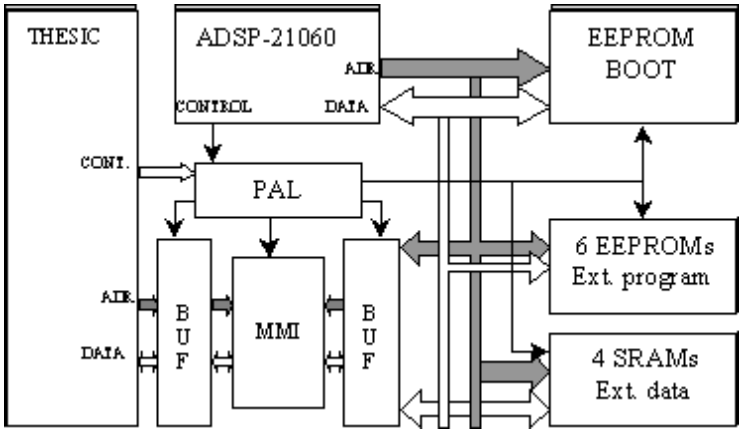


Figure 4.3 : Schéma bloc de la carte fille ADSP21060

La photo de la figure 4.4 représente la carte mère connectée à la carte fille ADSP21060. Il faut noter que le convertisseur Analogique / Digitale n'a pas été implanté sur la carte fille DSP.

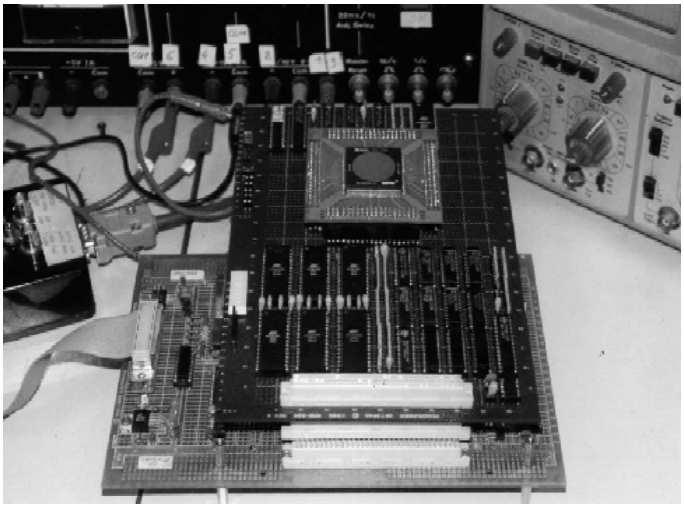


Figure 4.4 : La carte fille ADSP21060 connectée à la carte mère THESIC

Les expériences d'injection de fautes ont été réalisées sur les cartes filles du SHARC et du TS6833216A pour différents programmes : des programmes simples et les applications de vol du projet CESAR. Dans la suite, seront présentés les résultats d'expériences d'injection de fautes et des tests de radiation.

IV. STRATEGIE D'INJECTION DE CEU

L'injection de CEU pour le TS6833216A étant similaire à celle faite pour le microcontrôleur 80C51 étudiée dans le chapitre 2, alors seulement la méthodologie d'expérimentation dans le cas du SHARC sera analysée. Celle-ci diffère dans la lecture du code CEU à partir du SHARC. En effet :

- Le ADSP21060 n'exécute que des instructions codées en 48 bits,
- Le code CEU est transféré à partir de la carte mère vers la MMI sous forme de paquets de 8 bits.

Par conséquent, le code CEU sera d'abord transféré par le processeur SHARC, de la mémoire MMI vers la mémoire interne du DSP en transformant ce dernier de 8 bits en instructions de 48 bits.

Les étapes d'injection d'un CEU sont dans l'ensemble similaires à celles correspondantes du microcontrôleur 80C51, décrites dans la figure 2.4 de la section IV du chapitre 2, sauf que dans le cas du SHARC, l'étape n°6 de lecture du code CEU se répartie en deux étapes supplémentaires que nous avons appelé (Etape n°6 et Etape n°6 bis). La figure 4.5 décrit le principe de cette injection de CEU dans les zones sensibles du SHARC.

Etape n°1 : …….

⋮

Etape n°6 : Le SHARC arrête l'exécution du programme et pointe vers la procédure de traitement d'interruption (le code CEU) après sauvegarde du contexte. L'objectif de cette dernière est de transformer le code CEU en 48 bits dans la mémoire interne du DSP à partir d'une adresse prédéfinie.

Etape n°6 bis : Exécution du code CEU dans la mémoire interne du DSP SHARC.

⋮

Etape n°10: …….

Figure 4.5 : Stratégie d'injection de CEU dans les zones mémoires du SHARC

En prenant comme exemple que la position mémoire sélectionnée est un registre d'adresse (registre M5), la procédure d'injection de CEU peut être schématisée comme suit (figure 4.6):

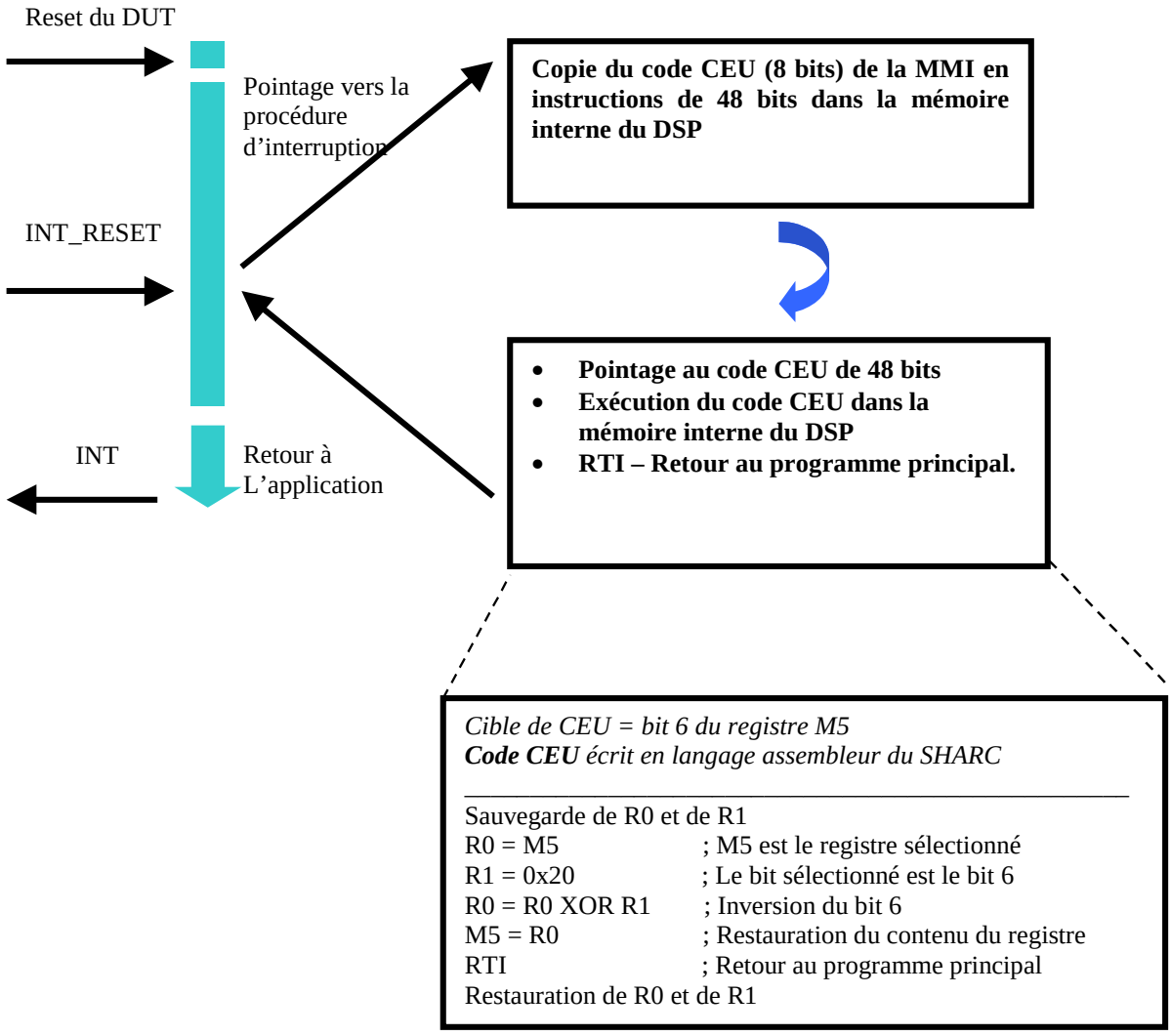


Figure 4.6 : Exemple d'injection de CEU dans un registre du SHARC

V. RESULTATS EXPERIMENTAUX : INJECTION DE CEU

Trois programmes différents ont été développés pour le ADSP21060, une multiplication de matrices 11 x 11, une Transformée de Fourrier Discrète D.F.T et les principaux modules de l'application de vol IRIS. Pour tous ces programmes, le code et les données sont résidents dans la mémoire interne du SHARC. Pour le microprocesseur TS6833216A, deux programmes benchmark (une multiplication de matrices 10 x 10 et une F.F.T) ainsi que l'application MEGA ont été étudiés.

A. Carte fille ADSP21060

1) Cibles d'injection de CEU

Les zones sensibles du SHARC sont la mémoire interne et les registres. La première est répartie sur 1.5 Mégabits pour le code (de 0x00020000 à 0x00027FFF en mots de 48 bits) et 2 Mégabits pour les données (de 0x00030000 à 0x0003FFFF en mots de 32 bits). Après un reset, le ADSP21060 prend les données de l'EEPROM de boot et les convertit sous forme de mots de 48 bits en les plaçant via la DMA (Direct Memory Access) à partir de l'adresse 0x00020080 de la mémoire interne du DSP. Ce processeur comprend aussi 172 registres d'entrée / sortie (I/O registres), situés entre 0x00 et 0x0FF dans le plan mémoire du SHARC et 101 registres de contrôle ou d'état, où chaque registre est codé sur 32 bits. La figure 4.7 représente le plan mémoire des registres I/O et l'organisation de l'espace mémoire interne dans le plan mémoire du SHARC.

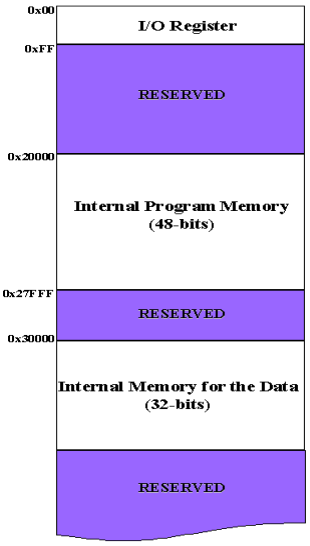


Figure 4.7 : Cibles d'injection de CEU pour le SHARC

Les divers codes CEU diffèrent selon la position de mémoire interne sélectionnée dans le DSP où on voudrait injecter le CEU. Huit cas particuliers ont été considérés et leurs codes CEU correspondants ont été calculés :

- la mémoire de données (32 bits) : Ce code concerne aussi les registres d'entrée/sortie et les vecteurs d'interruption, vu qu'ils sont mappés dans la mémoire interne du SHARC.
- la mémoire programme (48 bits) : Ce code est valable pour les adresses mémoire ayant un format de 48 bits situées entre 0x20000 et 0x27FFF du DUT. Vu que le format de ces données est de 48 bits et que la taille d'un registre (à l'aide duquel on effectue l'opération XOR pour l'inversion du bit sélectionné) est de 32 bits alors la séparation de ces données en 32 et 16 bits est nécessaire. Dès lors, deux codes différents ont été fabriqués pour ces deux cas: MSB 16 bits et LSB 32 bits.

- Les registres du CPU du SHARC (les registres de données et d'adresses : R0-R15, F0-F15, M0-M15, L0-L15, I0-I15).
- Registres R14 et R15 : Ce code CEU n'est pas valable pour les registres R14 et R15 puisqu'ils sont utilisés dans l'implantation du code lui-même pour l'inversion du bit, un autre code CEU spécifique à ces deux registres est nécessaire.
- Le compteur de programme PC.
- Le sommet de pile du compteur de programme du DSP, PCSTKP.
- Les deux registres de contrôle du mode de fonctionnement MODE1 et de l'état du calcul arithmétique ASTAT du DSP : Ces deux registres sont sauvegardés systématiquement par le SHARC dans la pile lors de la réception d'une interruption et repris après traitement de la procédure correspondante. Pour résoudre ce cas, nous avons remplacé le RTI par un RTS (retour de procédure). De cette manière, seulement le compteur programme sera repris de la pile et le programme d'application est ainsi repris là où il était interrompu.

  La structure de chaque code CEU est présentée dans l'annexe 4.2 avec un exemple détaillé pour chacun.

2) Exemples d'illustration d'injection de CEU

Les cibles d'injection de CEU sont les positions mémoires accessibles au programmeur. Parmi cette zone sensible (contenant approximativement 3.5 Mégabits), seulement 151 bits dans les registres du SHARC ne peuvent pas être accédés par l'utilisateur. Comparés à la totalité de la zone sensible accessible à l'injection de CEU, où les upsets peuvent être simulés ; le pourcentage de ces bits est de 4.10-5, ce qui est négligeable. En plus, il faut mentionner que les zones réservées dans le SHARC (dont le contenu est inconnu à l'utilisateur), représentées dans la figure 4.7, et la zone mémoire de la partie contrôle de ce processeur (registres, bascules, etc.) peuvent causer des erreurs durant les tests de radiation. Ces cibles, où les upsets ne peuvent pas être injectés, peuvent causer une différence entre le taux d'erreurs résultant des sessions d'injection et celui calculé lors des tests de radiation. En prenant en considération la grande taille de la mémoire interne où les erreurs peuvent être simulées, on peut s'attendre à ce que la différence soit négligeable.

Les effets résultant des injections de CEU diffèrent selon le taux d'occupation des zones sensibles (mémoire interne, registres, etc.) pendant l'exécution des programmes par le processeur. La table 4.1 présente les pourcentages des zones occupées, pour le code et les registres, pour chacun des programmes développés pour notre étude [72].

Table 4.1 : Taux d'occupation des zones sensibles du ADSP21060 pour chaque programme testé

	MATRICE	D.F.T.	IRIS
Code 48 bits	0.54 %	0.58 %	0.97 %
Données 32 bits	1.11 %	1.17 %	5.47 %
Registres. 32 bits	14.83 %	17.8 %	13.98 %
Zone sensible globale	0.86 %	1.02 %	3.31 %

Pour chaque programme, nous avons injecté 40000 upsets pseudo - aléatoires dans l'emplacement et l'instant d'occurrence. Comme dans le cas du 80C51, les résultats obtenus sont classifiés en trois groupes : erreurs tolérées, erreurs de résultats et perte de séquencement. Le taux d'erreurs τ_{CEU}, prédit pour chaque programme, est calculé selon la formule (1.6):

$$\tau_{CEU} = \frac{\#\,Erreurs}{\#\,CEUs\ injectés} \qquad (1.6)$$

Les résultats obtenus sont présentés dans la table 4.2 :

Table 4.2 : Résultats des sessions d'injection de CEU pour le SHARC

	MATRICE	D.F.T.	IRIS
Erreurs tolérées	99.68 %	99.36 %	97.74 %
Erreurs de résultats	0.14 %	0.03 %	2.15 %
Perte de séquencement	0.18 %	0.61 %	0.11 %
Taux d'erreurs (τ_{CEU})	0.32 %	0.64 %	2.26 %

A partir de ces expériences, on déduit qu'un faible nombre d'erreurs peuvent survenir lors du fonctionnement de l'application IRIS en environnement radiatif. Ceci est dû au faible taux d'occupation du programme IRIS de la zone sensible (approximativement 3%).

Pour les deux autres programmes considérés, le taux d'erreurs est négligeable. La différence entre les sensibilités est due au taux d'occupation de zones sensibles aux effets de SEU de chaque programme. En effet, l'application IRIS utilise approximativement trois fois (respectivement 4 fois) plus de mots de cette mémoire que les zones sensibles du DSP durant l'exécution du programme D.F.T (respectivement le programme de multiplication de matrices).

Concernant les types d'erreurs détectés, pour le programme D.F.T, les cas de perte de séquencement ont été moins fréquents que dans le cas des autres programmes. Ceci peut être expliqué par le grand nombre de registres utilisés par ce programme durant une boucle de calcul pour le stockage des paramètres critiques du programme. Les

CEU simulés sur ces cibles amènent le DSP à la perte de séquencement lors du dépassement du temps limité par le chien de garde logiciel implanté sur la carte mère.

B. Carte fille TS6833216A

1) Cibles d'injection de CEU

Les zones sensibles du microprocesseur TS6833216A sont la mémoire interne et les registres de contrôle d'état (figure 4.8). Les registres de ce circuit ont pour rôle de contrôler le fonctionnement de cinq modules :

- SIM : Module d'Intégration du Système, pour le contrôle du système (chien de garde, protection, etc.)
- CPU32 : Unité centrale de traitement du code,
- TPU : Unité de traitement dédié au Timer, fonctionnant indépendamment du CPU32,
- QSM : Module d'interface série : Il contient le bloc d'interface de communication série (SCI) et de périphérique série,
- TPURAM CTL: pour le contrôle des registres de l'émulateur de micro - code du TPU RAM.

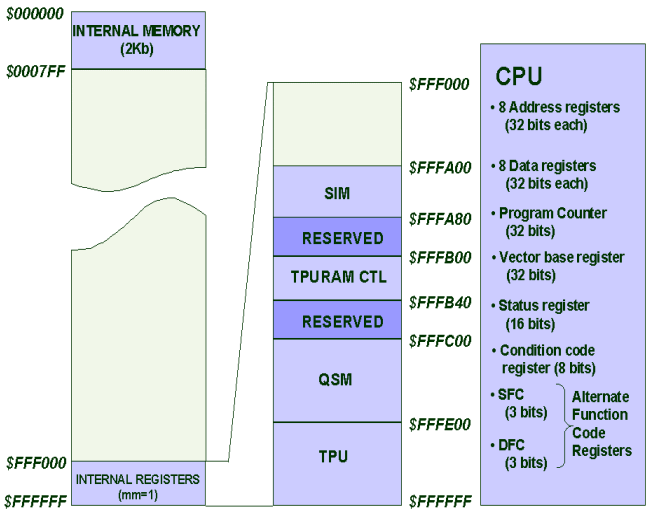


Figure 4.8 : Cibles de CEU pour le TS6833216A

Comme le montre la figure 4.8, les zones réservées de la mémoire interne représentent 320 octets. Il faut noter tout de même que les zones réservées (dont le contenu est inaccessible à l'utilisateur) et les éléments mémoire de la partie contrôle du microprocesseur peuvent causer des erreurs durant les tests de radiation. De même que dans le cas du SHARC, l'existence de ces zones, où les basculements de bits ne peuvent pas être simulés, peuvent résulter en des déviations dans le taux d'erreurs résultant comparé à celui obtenu lors des tests de radiation.

Les effets d'injection de CEU diffèrent selon le taux d'occupation des zones sensibles durant l'exécution d'un programme par le processeur à tester. La table 4.3 présente les pourcentages des zones sensibles occupées par la mémoire interne et par les registres, pour chacun des deux programmes benchmark considérés ainsi que pour l'application de vol exécutée sur ce microprocesseur.

Table 4.3 : Taux d'occupation des zones sensibles du TS6833216A pour chaque programme

	MATRICE	F.F.T	MEGA
Mémoire Interne (2048 Octets)	29,3 %	0 %	0 %
Registres (398 Octets)	16 %	21 %	40 %
Zone sensible globale	27.1 %	3.4 %	6.5%

Les données nécessaires au fonctionnement des applications F.F.T et MEGA sont stockées dans la mémoire externe. Sachant que cette mémoire n'a pas été exposée aux faisceaux d'ions lourds lors des expériences de radiation, elle ne contribuera pas au calcul du taux d'erreurs pour ces deux programmes.

C. Résultats des sessions d'injection de CEU

Pour chaque programme, nous avons injecté 40000 basculement de bits pseudo-aléatoires. La table 4.4 présente les taux d'erreurs lors des sessions d'injection de CEU pour le microprocesseur TS6833216A en exécutant les trois programmes [73].

Table 4.4 : Résultats des sessions d'injection de CEU dans le TS6833216A

	MATRICE	**F.F.T.**	**MEGA**
% Erreurs tolérées	86.48 %	97.8 %	99.3 %
% Erreurs de résultats	12.6 %	0.1 %	0 %
% Perte de séquencement	0.86 %	2.1 %	0.7 %
% Taux d'erreurs (τ_{CEU})	13.52 %	2.2 %	0.7 %

Pour l'application FFT ainsi que pour l'application MEGA, très peu parmi les CEU injectés ont causé des déviations dans les sorties du programme. Cependant, pour le programme de multiplication de matrices, où les matrices opérandes et résultat sont stockées dans la mémoire interne du microprocesseur, plus de 12% des CEU injectés ont provoqué des erreurs de résultats. Ceci peut être expliqué par le fait que les deux programmes (F.F.T et MEGA), à l'opposé du programme de multiplication de matrices, n'utilisent pas la mémoire interne pour le stockage des variables ou des constantes du programme. Globalement, la faible sensibilité aux CEU des applications F.F.T et MEGA est due certainement au faible taux d'occupation de la mémoire interne et des registres du microprocesseur, approximativement 2% pour le programme F.F.T et 0,7% pour l'application MEGA.

Finalement, seulement 0.86% des erreurs injectées dans le programme de multiplication de matrices causent des pertes de séquencement. Ceci est dû au fait que le programme n'utilise que quelques registres de contrôle. Concernant les types d'erreurs détectées, les erreurs de perte de séquencement ont été beaucoup plus fréquents pour les programmes F.F.T et MEGA, que pour l'application de multiplication de matrices. Le grand nombre de registres utilisés durant la boucle de calcul des résultats critiques pourrait expliquer ce comportement. Les upsets dans ces zones cibles peuvent amener le microprocesseur à la perte de séquencement provoquant donc le dépassement du temps fixé par le chien de garde implanté sur la carte mère.

VI. CALCUL DES FACTEURS DE SENSIBILITE DES ZONES MEMOIRE DU TS6833216A

Comme montré dans les chapitres précédents, l'un des avantages de la technique d'injection de CEU réside dans la possibilité d'effectuer des expériences déterministes d'injection de fautes dans un registre particulier ou dans des zones de données ou même de code, dont le contenu est critique pour l'exécution du programme. Les résultats d'une expérience réalisée sur le compteur de programme durant l'exécution du programme de multiplication de matrices montrent que 14,7% des CEU injectés ont été tolérés, 37,7% ont causé des erreurs de résultats et 47,7% ont provoqué des pertes de séquencement. Ceci montre que la sensibilité de ce registre est loin d'être 100% comme assumée généralement [14]. De plus, nous avons réalisé des expériences d'injection de CEU sur l'un des éléments des matrices opérandes afin de déterminer le facteur de sensibilité correspondant à cette zone mémoire. Les résultats obtenus montrent que le facteur de sensibilité des positions mémoires où sont stockés les opérandes de la matrice est d'environ 11%.

Sachant que la période d'une boucle d'exécution de la multiplication de matrices est de l'ordre de 13000 cycles et le grand nombre de registres et de positions de mémoire interne dans le microprocesseur TS6833216A (2K octets de mémoire interne et 590 registres), le calcul des facteurs de sensibilité de toute la zone sensible de ce processeur nécessiterait approximativement 74 jours (40 minutes par facteur de sensibilité calculé). A l'opposé, la simulation de 1000 CEU aléatoires, ce qui est suffisant pour prédire le taux d'erreurs d'une application donnée dans le même programme ne dépasse pas 17 minutes (1 seconde par CEU injecté). En effet, durant les tests de radiation, on se contente en général de l'obtention de quelques dizaines d'erreur d'un programme.

L'utilisation de séances exhaustives d'injection de CEU pour le calcul de la section efficace d'un programme pourrait être indispensable si les sections efficaces individuelles par registre diffèrent selon la zone testée de ce processeur. Dans ce cas, le calcul des facteurs de sensibilité sera plus judicieux et plus utile pour estimer une section efficace d'une application aussi précise que possible.

Pour illustrer l'estimation des facteurs de sensibilité, nous avons choisi un programme de multiplication de matrices, où le rang des deux matrices opérandes est de (2 X 2) ceci dans le but de limiter le nombre d'expériences d'injection de CEU. La table 4.5 donne les facteurs de sensibilité de chacune des positions mémoires (mémoire interne et registres) utilisées dans ce processeur pour l'exécution de ce programme calculés suite à l'injection exhaustive de CEU. Les facteurs de sensibilité des positions de mémoire non utilisées dans ce programme sont égaux à zéro.

Table 4.5 : Facteurs de sensibilité des positions mémoires utilisées

a) Facteurs de sensibilité des positions utilisées dans la mémoire interne

Position dans la Mémoire interne (H)	Facteurs de sensibilité (%)
400	1.643
401	1.555
402	1.580
403	1.554
404	3.028
405	3.056
406	3.144
407	3.113
408	2.434
409	2.406
40A	2.607
40B	2.441
40C	3.030
40D	2.920
40E	3.063
40F	3.082
410	2.629
411	2.715
412	2.771
413	2.633
414	2.176
415	2.286
416	2.115
417	2.090
Somme (Mémoire Interne)	**60.07**

b) Facteurs de sensibilité des registres utilisés

Registres	Facteurs de sensibilité (%)
D0	1.203
D1	1.772
D2	2.489
D3	0.48
A0	3.188
A1	2.309
A2	0.593
PC	61.47
SP	0.169
SR	14.97
Somme (Registres)	**88,64**

Total (Reg. + Mém. Int.)	**148.71**

Nous avons effectué des expériences d'injection de CEU, où les upsets sont injectés aléatoirement dans les positions mémoire et dans le cycle de l'exécution d'un programme. Le taux d'erreurs résultant est égal à 0.07%. Bien évidemment, avec cette méthode (injection aléatoire de CEU), on suppose que toutes les zones sensibles dans ce microprocesseur ont la même section efficace individuelle σ_{reg} . Si on applique la même hypothèse à la méthode basée sur le calcul du facteur de sensibilité de chaque position mémoire utilisée dans le programme suivant l'équation (1.1), on obtient :

$$\sigma_{SEU}(application) = \frac{\sigma_{SEU}}{n} \sum_{i=1}^{n} f_i \qquad (4.1)$$

Suivant les formules (IV. 1)et (I. 6), le taux d'erreurs peut être calculé selon la formule (4.2) :

$$\tau_{CEU} = \frac{\sum_{i=1}^{n} f_i}{n} \qquad (4.2)$$

D'après l'équation (5) et sachant que le nombre total de positions de la mémoire interne et des registres dans ce microprocesseur est égal à 2162, le taux d'erreurs peut être estimé alors à 0,069%. Cette valeur est en parfait accord avec le taux d'erreurs résultant de la méthode d'injection par simulation aléatoire de CEU dans l'ensemble des zones sensibles du processeur (0,07%).

VII. RESULTATS EXPERIMENTAUX: TESTS SOUS RADIATIONS

Deux campagnes des tests de radiation, durant lesquels le microprocesseur TS6833216A a été exposé aux faisceaux de différents types d'ions lourds, ont été réalisés, l'une en octobre 2000 à l'aide du Tandem Van der Graff, de l'IPN, et l'autre fin novembre 2000 avec le cyclotron Cyclone. La figure 4.9 montre l'ensemble carte mère/carte fille TS6833216A installé sur le support de l'enceinte sous vide du Cyclone. Les principales caractéristiques des ions lourds auquel le processeur étudié a été exposé sont données dans la table 4.6.

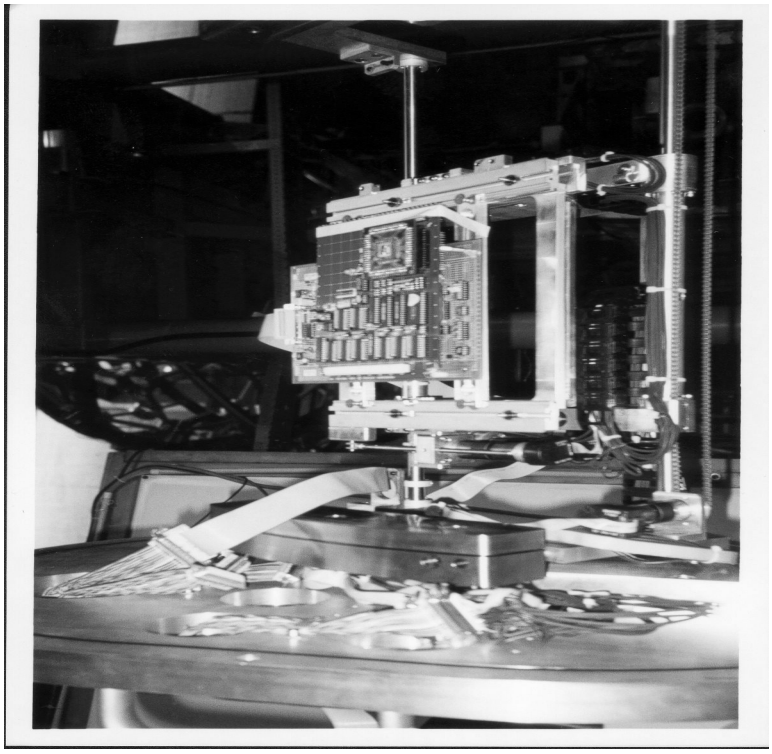


Figure 4.9 : Système THESIC placé dans l'équipement de tests sous ions lourds au cyclotron à Louvain-la-Neuve

Table 4.6 : Différents faisceaux utilisés pour le TS6833216A

a) Cyclone, Louvain la Neuve

M/Q=5	ENERGIE DUT [MEV]	LET [MeV/mg/cm²]
$^{40}Ar^{8+}$	150	14.1
$^{20}Ne^{4+}$	78	5.85
$^{15}N^{3+}$	62	2.97
$^{10}B^{2+}$	41	1.7

b) Tandem Van de Graff, Orsay

Ion lourd	ENERGIE DUT [MEV]	LET [MeV/mg/cm²]
Br	192	40.7

- **M** la masse atomique
- **Q** est l'état de charge de l'ion

Ces expériences de test sous radiation nous ont permis de mesurer les sections efficaces aux SEE (upsets et latchups) de la version militaire du microprocesseur TS6833216A en exécutant un test de mémoire (test statique) pendant l'exposition du circuit aux faisceaux d'ions lourds. Les courbes de section efficace obtenues de ce processeur sont données dans la figure 4.10.

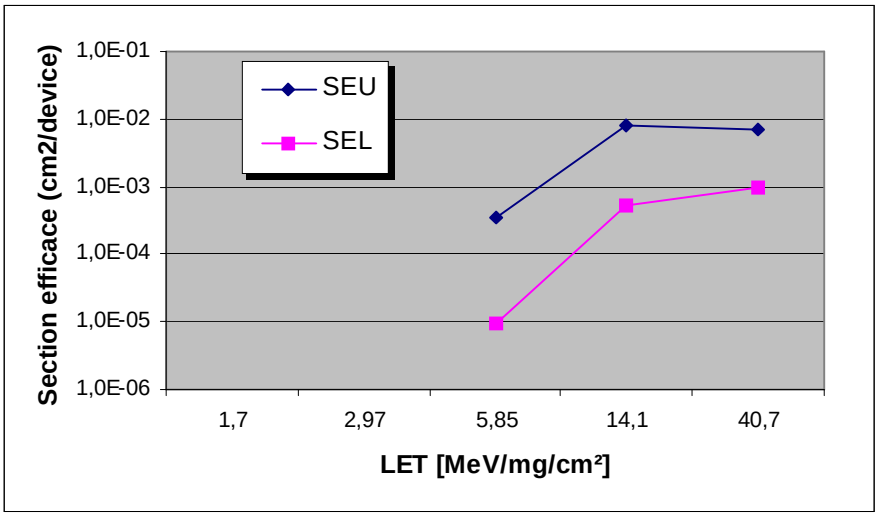


Figure 4.10 : Sensibilités aux SEE du TS6833216A

A. Estimation de la section efficace d'une application par injection aléatoire de CEU

Nous avons exécuté différents programmes sur le processeur TS6833216A (multiplication de matrices, F.F.T et MEGA) pendant l'exposition aux ions lourds de type Néon. Les sections efficaces prédites pour les différents programmes peuvent être estimés selon la formule (1.6). La table 4.7 présente les sections efficaces mesurées et prédites pour chaque programme.

Table 4.7 : Sections efficaces mesurées et prédites pour le TS6833216A sous les Ions lourds de type Néon

	MATRICE		F.F.T.		MEGA	
	Mesurée	Prédite	Mesurée	Prédite	Mesurée	Prédite
Erreurs de résultats	$3.9\ 10^{-5}$	$3.84\ 10^{-5}$	0	$3.05\ 10^{-7}$	0	0
Perte de séquencement	$0.8\ 10^{-5}$	$0.26\ 10^{-5}$	$8\ 10^{-6}$	$6.4\ \ 10^{-6}$	$2\ 10^{-6}$	$2.1\ 10^{-6}$
Section efficace	$4.7\ 10^{-5}$	$4.1\ \ 10^{-5}$	$8\ 10^{-6}$	$6.35\ 10^{-6}$	$2\ 10^{-6}$	$2.1\ 10^{-6}$

A l'exception de la multiplication de matrices, ces sections efficaces sont négligeables et montrent que si l'application exécutée par le TS6833216A n'utilise pas les éléments de la mémoire interne, les SEUs seront pratiquement sans effets. De plus, à l'exception de l'apparente sensibilité aux latchups de ce circuit, l'application de vol MEGA, même si elle occupe un espace mémoire assez important, est intrinsèquement robuste et peut par conséquent être exécutée sous radiation avec un risque mineur de perturbations. Une fois de plus, la comparaison entre les sections efficaces prédites et mesurées sont en très bonne corrélation, ce qui montre les bonnes performances de cette technique pour la prédiction de la section efficace d'une application quelconque.

Nous avons exécuté le programme de multiplication de matrices sous différents ions lourds, les sensibilités obtenues sont données dans la table 4.8, les sections efficaces correspondantes étant représentées dans la figure 4.11.

Table 4.8 : Sections efficaces mesurées et prédites pour le TS6833216A

Programme : Multiplication de Matrices (10x10)

Ion lourd	LET [MeV/mg/cm²]	Section efficace mesurée	Section efficace prédite
Bore (B)	1,70	*	*
Azote (N)	2,97	*	*
Néon (Ne)	5,85	$4{,}70.10^{-5}$	$4{,}10.10^{-5}$
Argon(AR)	14,1	$1{,}60.10^{-3}$	$1{,}06.10^{-3}$
Brome (Br)	40,7	$1{,}42.10^{-3}$	10^{-3}

* aucune erreur n'a été obtenue lors des test avec ces faisceaux d'ions.

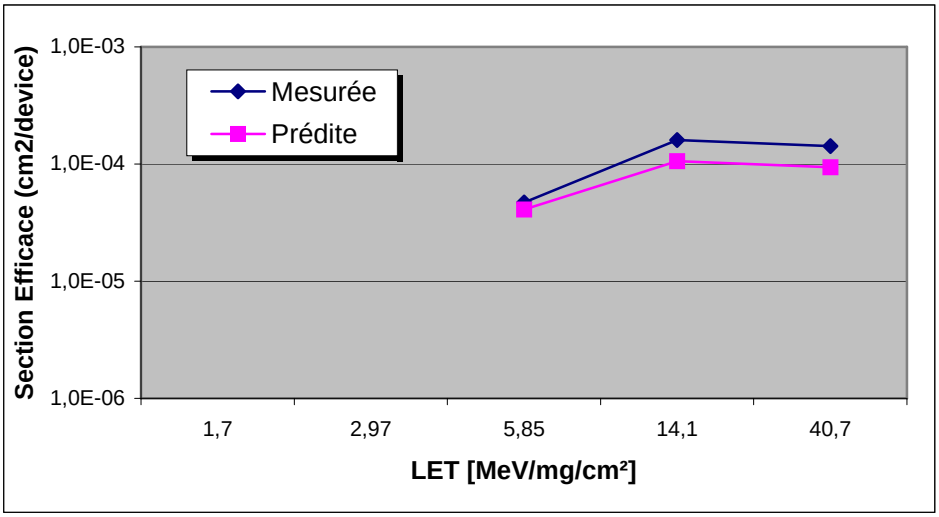


Figure 4.11 : Sections efficaces prédites et mesurées du microprocesseur TS6833216A exécutant un programme de multiplication de matrices (10x10)

La bonne corrélation entre les sections efficaces prédites et mesurées prouve l'efficacité de la technique d'injection de CEU pour prédire les sections efficaces de différentes applications avec différents types d'ions lourds, utilisant seulement les sections efficaces aux SEU, σ_{SEU} du processeur étudié. Ceci suppose que toutes les zones sensibles (registres et mémoire interne) du processeur étudié ont la même sensibilité aux SEU. Dans le cas contraire il sera nécessaire de calculer les facteurs de sensibilité et les sections efficaces pour chaque type de cible utilisée dans le DUT. Dans la prochaine section, nous allons décrire la méthodologie de calcul des sections efficaces correspondantes aux positions de mémoire interne et aux registres dans le cas du microprocesseur TS6833216A.

B. Estimation du sections efficaces d'une application par calcul des facteurs de sensibilité

Si on suppose que les positions de la mémoire interne et les registres ont tous la même section efficace. A partir de la formule (1.3), la section efficace d'une application peut être estimé comme suit :

$$\sigma_{SEU}(application) = \sigma_{reg} \sum_{i=1}^{n_{reg}} f_i + \sigma_{IntM} \sum_{i=1}^{n_{IntM}} f_i \qquad (4.3)$$

Où :

- n_{reg} est le nombre total de registres,
- n_{IntM} est le nombre de positions de mémoire interne,
- σ_{reg} est la section efficace par registre,
- σ_{IntM} est la section efficace d'une position de la mémoire interne.

Table 4.9 : Sections efficaces prédites pour le TS6833216A

Programme : la Multiplication de Matrices (2x2)

	Somme des facteurs de sensibilités	**Section efficace par registre**
Mémoire interne	60.07	$1.27\ 10^{-7}$
Registre	88,64	$2.60\ 10^{-7}$
Section efficace	$\sigma_{SEU}(application) = \sigma_{reg} \sum_{i=1}^{n_{reg}} f_i + \sigma_{IntM} \sum_{i=1}^{n_{IntM}} f_i =$	

a) par calcul des facteurs de sensibiltés

	% Taux d'erreurs par calcul aléatoire CEU Injection	**Section efficace par composant**
TS6833216A	0.07	$3{,}05\ 10^{-4}$
Section efficace	$\sigma_{SEU}(application) = \sigma_{SEU} * \tau_{CEU} = 2.1\ \ 10^{-5}$	

b) par injection aléatoire

Les résultats représentés dans la table 4.9 mettent en évidence une différence de 10^{-5} entre les sections efficaces calculés d'abord par moyen de calcul de facteurs de sensibilité et ensuite par l'injection aléatoire de CEU dans toutes les zones sensibles du processeur. Il doit être mentionné que la différence entre les taux d'erreurs mesurés et prédits obtenus pour le microprocesseur TS6833216A (figure 4.11) avec la multiplication de matrices (10X10), bien que faible est due probablement à la différence entre les sections efficaces individuelles correspondantes à une position dans la mémoire interne et à un registre particulier.

VIII. CONCLUSION

Dans ce chapitre nous avons présenté une nouvelle application de la méthode d'injection de CEU pour la prédiction des taux d'erreurs sous radiations pour deux architectures digitales bâties respectivement autour d'un processeur de traitement de signal ADSP21060 de Analog Devices et d'un microprocesseur TS6833216A de Motorola. La méthode de simulation de fautes par injection de CEU a été utilisée pour le calcul des facteurs de sensibilité des différents positions mémoires utilisées dans le programme.

Le comportement de ces architectures en présence de CEU a été étudié pour différents types de programmes. Pour la carte digitale bâtie autour du SHARC, deux programmes benchmark (une multiplication de matrices et une D.F.T) ainsi qu'un programme développé pour le contrôle d'une caméra CCD inclue dans un projet satellite scientifique. Pour la carte TS6833216A, ont été utilisés deux programmes simples (une multiplication de matrices et une F.F.T) et l'application de contrôle d'un spectromètre. Les résultats des sessions d'injection de CEU ont mis en évidence la très faible sensibilité des programmes de vol face aux upsets, qui peuvent donc être considérés robustes par rapport aux effets des particules chargées.

Pour confirmer ces résultats et déterminer les sensibilités aux SEU des processeurs étudiés, des tests sous radiations ont été effectués. Vu que la version militaire du SHARC n'était pas disponible pour les expériences de radiations, seul le TS6833216A a subi des tests sous ions lourds. Deux campagnes de radiation ont été réalisées, l'une à l'aide de l'accélérateur de particules Tandem Van der Graff et l'autre avec le cyclotron Cyclone, durant lesquelles le microprocesseur TS6833216A a été exposé à divers faisceaux d'ions lourds. La très bonne corrélation entre les sections efficaces prédites et mesurées pour chaque application apporte de nouvelles données prouvant l'efficacité de la technique proposée dans cette thèse. Les deux applications de la méthode d'injection de CEU (au TS6833216A et au SHARC) mettent en évidence sa versatilité pour la prédiction du taux d'erreurs d'un programme quelconque sous divers types d'ions lourds.

Conclusions et Perspectives

Dans cette thèse, nous avons présenté une méthodologie pour l'estimation du taux d'erreurs induit par des upsets survenant en environnement radiatif sur des architectures digitales à base de processeurs. Ces estimations sont généralement effectuées à partir des sensibilités individuelles des divers composants intégrant l'architecture étudiée, déterminées comme le résultat d'essais au sol réalisés à l'aide d'accélérateurs de particules. Dans le cas des processeurs, l'application finale est rarement utilisée lors de ces essais, rendant potentiellement imprécises les estimations du taux d'erreursen vol.

A la base de ces recherches se trouve l'injection par des moyens logiciels/matériels d'erreurs de type upset (inversion du contenu d'un bit d'un registre ou d'un mot mémoire). Ces erreurs, appelées CEU (Code Emulant un Upset) tout au long du rapport, sont injectées dans des registres ou des mots mémoires, en se servant des signaux d'interruption du processeur présent dans l'architecture cible. Un aspect crucial qui doit être pris en compte pour que les CEU miment correctement les upsets survenant en environnement radiatif, est la nature aléatoire de leur occurrence, aussi bien dans le temps que dans la cible perturbée. Nous avons résolu *l'aléatoireté* de l'instant d'occurrence par un mécanisme extérieur à l'architecture cible, basé sur un compteur initialisé à une valeur aléatoire, dont le passage à zéro provoque l'activation d'un signal d'interruption. Suite à l'activation du signal d'interruption, les bits cibles seront perturbés par des moyens purement logiciels, comme conséquence de l'exécution du programme de traitement de l'interruption. De cette manière, avec une intrusion matérielle et logicielle minimales, peuvent être simulées l'occurrence d'upsets de manière concurrente avec l'exécution d'un programme quelconque.

Pour valider cette méthode, nous l'avons appliquée à deux processeurs différents : deux microcontrôleurs (le 80C51 d'Intel et le TS6833216A de Motorola) et à deux processeurs de traitement de signal (le TMS320C50 de Texas Instruments et le SHARC de Analog Devices) en nous servant du testeur THESIC développé à TIMA en collaboration avec le CNES. Pour ce faire, un nouveau mode de fonctionnement a été implanté dans le testeur THESIC permettant d'automatiser les principales étapes correspondantes à l'injection de CEU : sélection aléatoire des paramètres du CEU, activation des interruptions provoquant son occurrence, traitement des résultats.

Ces outils ont été validés par la mise en œuvre d'expériences au cours desquelles des milliers de CEU ont été injectés. L'analyse des résultats obtenus a permis la mise en évidence de deux retombées principales de cette étude : la catégorisation des dysfonctionnements provoqués au niveau du système testé, l'estimation précise du taux

d'occurrence de chacune de ces familles d'erreurs. Comme conséquence directe de ces résultats peuvent être décidées les stratégies (redondances matérielles ou logicielles) qui doivent être adoptées pour que l'architecture fonctionne de manière fiable dans l'environnement final de l'application. De plus, une fois le système rendu fiable, la qualité de la solution adoptée peut être évaluée par une nouvelle session d'injection de CEU.

Les cas étudiés ont fourni un riche terrain d'expérimentation dont les résultats ont été discutés dans ce rapport. D'un point de vu qualitatif, l'analyse de la propagation des cas représentatifs parmi les CEU injectés, montre le bien fondé de la méthode et valide les développements matériels et logiciels effectués pour son application. D'un point de vu quantitatif, la confrontation des sections efficaces estimées avec ceux mesurées suite à la réalisation d'un test sous faisceau d'ion lourds avec les mêmes architecture et programme, met en évidence une excellente corrélation.

La sensibilité d'un processeur lorsqu'il exécute le programme final d'application peut donc être estimée à partir de résultats d'essais d'injection de CEU, en prenant en compte seulement les sensibilités de base des différentes familles de cibles de SEU. De cette manière, les essais sous radiations peuvent être limités à une unique campagne pour déterminer les courbes de sections efficaces, qui sont obtenues à l'aide de stratégies de test statique relativement faciles à mettre en œuvre. Dans le cas où des changements sont apportés au programme de l'application, l'approche d'injection de CEU évite des essais en accélérateur, toujours coûteux et de planification délicate.

Les résultats présentés dans cette thèse sont très encourageants, et ouvrent une nouvelle voie dans le domaine de la qualification de systèmes destinés à des projets spatiaux. La technique d'injection de CEU proposée, a été appliquée à différents processeurs. La corrélation entre prédictions et mesures des sections efficaces a été réalisée par l'intermédiaire de différentes campagnes d'essais sous radiations. Le développement d'outils permettant d'automatiser définitivement la préparation et la mise en œuvre d'essais d'injection de CEU est l'un des objectifs prioritaires de la nouvelle étude envisagée dans le cadre de la continuation et l'amélioration de ces travaux, qui dans le cas de succès pourraient devenir un standard.

Les travaux futurs incluent aussi l'étude d'un logiciel pour la conception automatisée de cartes filles THESIC. La description du circuit cible en langage de haut niveau (VHDL par exemple) sera l'entrée d'un outil logiciel qui générera la configuration d'un FPGA présent sur la carte fille et ayant pour fonction l'adaptation du DUT à l'interface carte mère/carte fille [74].

Références

[1] E. Normand, Single-Event Effects in Avionics, IEEE Trans. on Nuclear Science, Vol. 43, n° 2, pp. 461-474, April 1966.

[2] T. Ma, P. Dressendorfer, *Ionizing Radiation Effects in MOS Devices and Circuits*, Wiley Eds., New York, 1989.

[3] D. K. Nichols, et al., Update on parts SEE susceptibility from heavy ions, IEEE Trans. Nucl. Sci., Vol. 38, N°6, p. 1529, Déc. 1991

[4] E. L. Peterson, Single event upsets in space : basic concepts, Tutorial short course, IEEE Nuclear and Space Radiation Effects Conference, Juil. 1983.

[5] R. Koga, et al., Heavy ion induced single event upsets of microcircuits; a summary of aerospace corporation test data, IEEE Trans. Nucl. Sci., Vol. 31, N°6, p.1190, Déc. 1984.

[6] J. C. Pickel, Single event upset mechanisms and predictions, Tutorial short course, IEEE Nuclear and Space Radiation Effects Conference, Juil. 1983.

[7] E. L. Petersen, “Single event upsets in space: basic concepts”, Tutorial short course, IEEE Nuclear and Space Radiation Effects Conference, Juil. 1983

[8] D. K. Nichols, et al., A summary of JPL single event upset test data from 1982, through 1984, IEEE Trans. Nucl. Sci., Vol. 32, N°6, p. 1186, Déc. 1985.

[9] A. Dantec, Le phénomène de latchup dans les circuits intégrés CMOS, Toute l’électronique, TLE, 465, p.45, 1981.

[10] BARTH, Janet. Radiation Environment. In: IEEE NSREC Short Course, July 21, 1997. http://flick.gsfc.nasa.gov/radhome/ RPO_slides.htm.

[11] J. Bourrieau, “L’environnement spatial: flux, dose, blindage, effets des ions lourds”, Tutorial short course, RADECS, Sept. 1991.

[12] Filipe Vinci Dos Santos, Techniques de conception pour le durcissement des circuits intégrés face aux rayonnements, thèse préparée à l’Université Joseph Fourrier, 15 Oct. 1998.

[13] W. A. Kolasinski, et al., Techniques of microprocessor testing and SEU rate prediction, IEEE transactions Nuclear Science, Vol. 26, N°.6, p.5087, Déc. 1979.

[14] R. Koga, W. A. Kolasanski, M. T. Marra and W. A. Hanna, Techniques of microprocessor testing and SEU-rate prediction, IEEE Trans on Nuclear Science Vol. NS-32, N° 6, pp. 4219-4224, Déc. 1985.

[15] V. Asenek, Predicting the reliability of electronic subsystems and « Commercial-Off-Cost-Shelf » microprocessors on low-cost-small satellites, thèse préparée à l'Université de Surrey, Grande Bretagne, Juil. 1998.

[16] R. Harboe -Sorensen, et al., SEU risk assessment of Z80A, 8086 and 80C86 microprocessors intended dor the use in a low altitude polar orbit, IEEE Trans. Nucl. Sci., Vol. 33, N°6, p. 1626, Déc. 1986.

[17] R. Harboe -Sorensen, et al., Radiation pre-screening of R3000 / R3000A microprocessors, IEEE NSREC Workshop, New Orleans (USA), Juil. 13-17, 1992.

[18] R. L. Pease, et al. Radiation testing of semiconductor devices for space electronics, Proceedings of the IEEE, Vol. 76, N°.11, p.1510, Nov. 1998.

[19] W. E. Price, et al., Cosmic ray induced errors in MOS memory cells, IEEE Trans. Nucl. Sci., Vol. 28, N°6, p. 1166, Déc. 1981.

[20] S. Karoui, Etude du comportement de circuits complexes en environnement radiatif spatial, thèse préparée au sein du laboratoire de Génie Informatique, Institut National Polytechnique de Grenoble, 26 Novembre 1993.

[21] S. Karoui, et al, « SEU and Latchup results for Sparc processors », IEEE Vol. NS-40, Déc. 1993.

[22] R. Velazco, S. Karoui, T. Chapuis, D. Benezech, L. H. Rosier, Heavy ion tests for the 68020 Microprocessor and the 68882 Coprocessor, IEEE Trans. on Nuclear Science, Vol. 39, N° 3, Déc. 1992.

[23] F. Bezerra, R. Velazco, A. Assoum, D. Benezech, *SEU and Latchup results on Transputers*, IEEE Trans. of Nuclear Science, Part. I, Vol. 43, Number 3, IETNAE, pp. 893-8978, 1996.

[24] F. Bezerra, D. Benezech, R. Velazco, "Etude de la sensibilité des Transputers aux phénomènes de SEU et latchup". *Proc. of Radiation Effects on Components and Systems (RADECS)*, Arcachon, pp. 340-34, 18-23 Septembre 1995.

[25] J. H. Elder, J. Osborn, W.A. Kolasinsky, R. Koga, A method for characterizing microprocessor's vulnerability to SEU, IEEE Trans. on Nuclear Science, Vol. 35, N° 6, pp. 1679- 1681, December 1988.

[26] J. Cusick, et al., « SEU vulnerability of the Zilog Z80 and NSC microprocessor », IEEE Transactions on Nuclear Sciences, Vol. 30, N°6, p.4206, Déc. 1985.

[27] V. Asenek, C. Underwood, R. Velazco, S. Rezgui, M. Oldfield, Ph. Cheynet, R.Ecoffet, SEU induced errors observed in microprocessor systems, *IEEE Transactions on Nuclear Science*, Vol. 45, Nb. 6, IETNAE, pp. 2876-2883, Déc. 1998.

[28] M. Rebaudengo, M. Sonza Reorda, A unified environment for the fault injection in embedded microprocessor-based systems, 5 IEEE International On-Line Testing Workshop -Rhodes (Greece), Juil. 5-7, 1999.

[29] A. Benso, P. Prinetto, M. Rebaudengo, M. Sonza Reorda, A Fault Injection Environment for microprocessor-based boards, IEEE International Test Conference, pp. 768-773, 1998.

[30] J. Carreira, H. Madeira, J. G. Silva, Xception : A technique for the experimental evaluation of dependability in modern computers, *IEEE Transactions in Software Engineering*, Vol. 24, N° 2, pp. 125-136, Fév. 1988.

[31] J. Arlat, M. Aguera, L. Amat, Y. Crouzet, J. C. Fabre, J. C. Laprie, E. Martins, D. Powell, Fault injection for dependability validation: a Methodology and some applications, IEEE Transactions on Software Engineering, Vol. 16, No. 2, Fév. 1990, pp. 166-182.

[32] A. Benso, P.L. Civera, M. Rebaudengo, M. Sonza Reorda, A low cost programmable board for speeding up fault injection in microprocessor-based systems, *Annual Reliability and Maintainability Symposium*, 1999, pp. 171-177.

[33] V. Asenek, C. Underwood, M. K. Oldfield, *Predicting the rate and effects of single event upsets in satellite systems using in microprocessor simulator*, 2nd Round Table on Micro and Nano Technologies for Space, ESA - ESTEC, Noordwijk (Netherlands), 15-17 Oct. 1997, pp. 237-244.

[34] T. A. Delong, B. W. Johnson, J. A. Profeta III, A fault injection technique for VHDL behavioral-level models, IEEE Design & Test of Computers, hiver 1996, pp. 24-33.

[35] K. W. Li, J. R. Armstrong, J. G. Tront, An HDL simulation of the effects of Single Event Upsets on microprocessor program flow, *IEEE Trans on Nuclear Science Vol. NS 31, N° 6, pp. 1679- 1681*, Déc. 1984.

[36] R. Velazco, S. Rezgui, Rapport Final CNES, *Méthodes et Outils pour l'injection d'erreurs de type « UPSET » sur des systèmes à base de microprocesseurs*, Mai 2000.

[37] R. Velazco, S. Rezgui, R. Ecoffet, Predicting Error Rate for Microprocessor-Based Digital Architectures through C.E.U. (Code Emulating Upsets) Injection, IEEE Transaction of Nuclear Science, Décembre 2000. *Nuclear and Space Radiation Effects Conference (NSREC'00), Reno*, Nevada (USA), 24-28 Juil. 2000.

[38] R. Velazco, S. Rezgui, Transient Bitflip Injection in Microprocessor Embedded *Proceedings of IOLTW 2000, 6th IEEE International On-Line Testing Workshop, Mallorca*, (Espagne), pp. 80-84, 3-5 Juil. 2000.

[39] R. Velazco, S. Rezgui, R. Ecoffet, *Injecting CEU (Code Emulating Upsets) to evaluate the error rate Of microprocessor-embedded digital applications*, 2000 Single Event Effects (SEE) Symposium, Manhattan Beach, Los Angeles, 11-13 Avr. 2000.

[40] R. Velazco, Ph. Cheynet, A. Bofill, R. Ecoffet, THESIC: A testbed suitable for the qualification of integrated circuits devoted to operate in harsh environment, IEEE European Test Workshop (ETW'98), Sitges, (Espagne), pp. 89-90, 27-29 Mai 1998.

[41] F. Bezerra, D. Hardy, R. Velazco and H. Ziade, TILMICRO, a new SEU latch-up tester for microprocessors initial results on 32-bit floating point DSPs, RADECS Radiation and its effects on Components and Systems, pp. 296-301, 1995.

[42] D. Falguère, et al., "SEETEST: A system dedicated to the caracterisation of memory sensibility to SEE", RADECS, p. 479, Sept. 1991.

[43] S. Karoui, et al, « A low cost functional test system : the FUTE 16 tester », Proceedings of the International Conference on Microelectronics (ICM 92), p.511, Monastir, Déc. 1992.

[44] S. Rezgui, Ph. Cheynet, R. Velazco, N. Coderch et J. Cueto, *Un système de test pour la qualification de circuits intégrés destinés à fonctionner en environnement sévère*, colloque CAO, Aix-en-Provence, Mai 1999.

[45] R. Velazco et al., Neurone Carte Fille, Carte Mère: Recueil des documents, Mai 1996.

[46] R. Velazco, Th. Calin, M. Nicolaidis, S.C. Moss, S.D. La Laumondiere, V. T. Tran, R. Koga, SEU-hardened storage cell validation using a pulsed laser, *IEEE Transactions on Nuclear Science*, Part I, Vol. 43, pp. 2843-2848, 1996.

[47] Ph. Cheynet, R. Velazco, S. Rezgui, L Peters, K. Beck, R. Ecoffet, Digital Fuzzy Control: a Robust Alternative Suitable for Space, à paraître dans *IEEE Transactions on Nuclear Science*, Vol. 45, Nb. 6, IETNAE, pp.2941-2947, Déc. 1998.

[48] R. Koga, et al., « Advantages of the LBL cyclotron ion beam for SEP studies », RADECS, p. 586, Sept. 1991.

[49] R. Velazco, Ph. Cheynet, J-D. Muller, R. Ecoffet, Artificial Neural Network Robustness for on-board satellite image processing : Results of SEU simulations and ground tests, *IEEE Trans. on Nuclear Science*, Vol. 44, n° 6, pp. 2337-2344, 1997.

[50] D. H. Lehmer, G. H. Hardy and E. M. Wright, An Introduction to the Theory of Numbers, 5th edition. Oxford University Press. 1979.

[51] Y. Souchard , Réalisation d'une interface graphique sous Visual C++, IUT1 Grenoble, Avr-jui 1999.

[52] The 8051 Family of Microcontrollers, Richard H. Barnet,Prentice-Hall, 1995

[53] R. Velazco, S. Rezgui, H. Ziade, Assessing the soft error rate of digital architectures devoted to operate in radiation environment: a case studied, *accepté pour présentation à LATW 2001 (IEEE Latin-American Test Workshop), Cancun (Mexique)*, 11-14 Fév. 2001.

[54] E. F. Cota, L. Carro, M. Lubaszewski, R. Velazco, S. Rezgui, Synthesis of a 8051-like Microcontroller Tolerant to Transient Faults, *LATW 2000 (1st IEEE Latin-American Test Workshop)*, Rio de Janeiro, Brazil, 13-15 Mars 2000, *Reviewing en cours pour parution au journal JETTA special.*

[55] F. Gusmao de Lima, É. Cota, L. Carro, M. Lubaszewski, R. Reis, R. Velazco, S. Rezgui, Designing a Radiation Hardened 8051-like Micro-controller, Proceedings of SBCCI2000 (XIII Symposium on Integrated Circuits and Systems Design), Manaos, (Brésil), pp. 255-260, 18-24 Sept. 2000.

[56] F. G. Lima, S. Rezgui, E. Cota, L. Carro, M. Lubaszewski, R. Velazco, R. Reis, Designing and Testing a Radiation Hardened 8051-like Micro-controller, Présenté à *MAPLD 2000 Military and Aerospace of Programmable Devices and Technologies*, Laurel, Maryland (USA), Vol. 1, Session B, 26-28 Sept. 2000. *Reviewing en cours pour parution à JSR (Journal of Space Rockets).*

[57] F. G. Lima, S. Rezgui, L. Carro, R. Velazco, R. Reis, Driving Error Rate from VHDL Fault Injection in Processor-like Circuits, Soumis à *RADECS Conference 2001, Radiation Effects on Components and Systems*, Grenoble (France), 10-14 Sept. 2001.

[58] F. G. Lima, S. Rezgui, L. Carro, R. Velazco, R. Reis, High Level Hardening of Processor-like Circuits, soumis à Student Design Contest, papier DAC Nb. SDC-BLI909, pour le 38th Design Automation Conference, Las Vegas 2001.

[59] R. Velazco, et al., “Comparison between Californium and cyclotron SEU tests”, IEEE Trans. Nucl. Sci., Vol.36, No. 6, p. 2383, Déc. 1989.

[60] http://ipnsua.in2p3.fr/~tandem/machine/machine.html

[61] G. Berger, G. Ryckewaert, R. Harboe-Sorensen, L. Adams, Cyclone - a multipurpose Heavy Ion, Proton and Neutron SEE Test Site, RADECS Radiation and its effects on Components and Systems, 1997.

[62] S. Rodriguez, Spécification fonctionnelle du projet Nanosat, documentation fournie par l’INTA, Sept. 1998.

[63] A. Bofill, R. Velazco, Ph. Cheynet, M. Alvarez, S. Rodriguez, Space Qualification of an Architecture Based on a Digital Signal Processor for a Nanosat Project, *6th International Workshop on Digital Signal Processing Techniques For Space (DSP'98)*, Poster Session P.5, Noordwijk, (Pays-Bas), 23-25 Sept. 1998.

[64] A. Bofill, Qualification du TMS320C50 à l’aide du système THESIC, Diplôme d’Ingénieur en Télécommunications, Université Polytechnique de Catalunya (UPC), Barcelone, Espagne Oct 1997 - Mai 1998.

[65] TMS320C50 Users's guide, 1993, Texas Instruments.

[66] G. Picas, Essais d 'injection de fautes sur un modem à l'aide du système THESIC, Diplôme d'Ingénieur en Télécommunications, Université Polytechnique de Catalunya (UPC), Barcelone, Espagne Mars 1998 - Septembre 1999.

[67] S. Rezgui, R. Velazco, R. Ecoffet, S. Rodríguez, J. R. Mingo, *Estimating Error Rates in Processor-Based Architectures, accepté pour présentation au* Workshop RADECS 2000, Radiation Effects on Components and Systems, Louvain - la - Neuve, Belgique, 11-13 Sept. 2000.

[68] S. Rezgui, R. Velazco, R. Ecoffet, Votre titre (je suggère le titre de votre thèse ! !), soumis à Nuclear and Space Radiation Conference (NSREC 2001, Vancouver, Canada, Juil. 2001).

[69] S. Rodriguez, Spécification fonctionnelle du projet CESAR, documentation fournie par l'INTA, Janvier 2000.

[70] MC68332, Users's Manual, 1995.

[71] ADSP-2106X SHARC User's Manual (2nd Edition 7/96).

[72] S. Rezgui, R. Velazco, J.R. Mingo, *Ground Testing of Architecture Including Digital Processor Signal AD21060, soumis à* DSP-Germany 2000, European Conference, Munich, Allemagne, 11-12 Oct. 2000.

[73] S. Rezgui, R. Velazco, R. Ecoffet, S. Rodríguez, J.R. Mingo, A New Methodology for the Simulation of Soft Errors on Microprocessors : A Case Study, Présenté à *MAPLD 2000 Military and Aerospace of Programmable Devices and Technologies*, Laurel, Maryland (USA), Vol. 1, Session B, 26-28 Sept. 2000. *Reviewing en cours pour parution à JSR (Journal of Space Rockets).*

[74] R. Velazco, S. Rezgui, E. Reguer, THESIC : A flexible platform for the functional validation of integrated circuits, *soumis à IBERCHIP, 21-23 mars 2001 (Montevideo (Uruguay).*

RESUME EN FRANCAIS

Cette thèse est consacrée à l'étude du comportement de processeurs digitaux face à l'un des effets induits par l'environnement radiatif : le phénomène dit *SEU* ou *upset* qui se traduit par le basculement intempestif du contenu d'un élément mémoire comme conséquence de l'ionisation produite par le passage d'une particule chargée. Les conséquences de ce phénomène dépendent de l'instant d'occurrence et de l'élément mémoire affecté et peuvent aller de la simple erreur de résultat à la perte de contrôle d'un engin spatial.

Les techniques de durcissement ne pouvant pas garantir entièrement l'immunité face aux upsets des circuits candidats aux applications spatiales, des méthodes d'estimation des taux d'erreurs de ces applications par des tests sous radiation ou par injection de fautes s'avèrent nécessaires, dans le double but de choisir les circuits les moins sensibles à ces effets et d'étudier le comportement des applications de vol face aux upsets.

L'objectif de cette thèse consiste en la définition d'une méthode d'injection de fautes de type upset et de son expérimentation sur différentes architectures digitales afin d'étudier ses potentialités ainsi que son efficacité. La méthode proposée se base sur l'injection d'erreurs de type upset sur une carte digitale bâtie autour du processeur cible, comme conséquence de l'activation d'un signal d'interruption asynchrone. L'exécution de la séquence de traitement de l'interruption appelée CEU dans cette thèse (Code Emulant un Upset) provoquera la modification du contenu d'un bit sélectionné aléatoirement parmi les éléments de la zone mémoire sensible aux upsets du processeur.

L'implantation de cette technique a été réalisée par l'intermédiaire d'un système THESIC, testeur dédié à la qualification sous radiation de circuits intégrés. Ce système comporte deux cartes digitales (carte mère/carte fille), dont la configuration s'est révélée adaptée aux contraintes imposées par la technique d'injection de fautes proposée. L'objectif final de ces recherches a été de démontrer que le taux d'erreurs d'une application peut être prédite à partir des résultats issus d'essais d'injection d'upsets et des mesures des sensibilités des éléments mémoires du processeur considéré. La confrontation de ces prédictions avec des mesures réalisées à l'aide d'accélérateurs de particules, a permis de montrer la validité de l'approche proposée pour différents types de processeurs.

TITRE EN ANGLAIS :

ERROR RATE PREDICTION FOR DIGITAL ARCHITECTURES : A METHOD AND EXPERIMENTAL RESULTS

RESUME EN ANGLAIS

This thesis aims at the study of the behavior of digital processors with respect to one of the effects of radiation environment - the Single Event Upset phenomenon, also called upset - which may modify the content of memory elements as the result of the silicon ionization resulting from the impact of charged particles. The consequences of upsets for a given application depend on both the occurrence instant and the perturbed memory element, and can go from innocuous result errors to system crashes which may provoke the loose of control of a space vehicle.

As design hardening techniques cannot completely guarantee the upset immunity for circuits devoted to space applications, error rate estimation methods, based on radiation ground testing and/or in fault injection experiments, are mandatory to choose the less sensitive circuits for a given space application.

The research presented in this thesis consists in the definition of a suitable method for upset-like fault injection and its experimentation for various digital architectures in order to assess its efficiency and put in evidence its capabilities. The proposed method is based on the injection of upsets in a digital board built around a processor, as the consequence of the activation of an asynchronous interruption. The execution of the instruction sequence associated with the interruption, called here CEU (Code Emulating an Upset), will provoke the modification (bit flip) of a target selected among the circuit sensitive area which comprises mainly registers and internal memory elements.

The CEU injection technique was implemented using THESIC, a system dedicated to the qualification of integrated circuits under radiation. This system is composed of two digital boards (a mother board and a daughter board) whose configuration revealed as being well adapted to constraints imposed by the studied fault injection approach. Demonstrating that application error rates can be predicted from the results of CEU injection experiments combined with the measure of individual sensitivities to upsets of the processor's memory elements obtained from radiation testing. The confrontation for different architectures and programs, of predicted error rates to measured ones proved the validity of the approach.

Spécialité : MICROELECTRONIQUE

MOTS CLES : Environnement spatial, Tests aux Ions lourds, Single Event Upset, Injection de Fautes, CEU, Architectures digitales.

Laboratoire **TIMA,** Techniques de l'Informatique et de la Micro-électronique pour l'Architecture des ordinateurs, 46 Avenue Félix Viallet, 38031 Grenoble

ISBN 2-913329-58-6 Broché
ISBN 2-913-329-59-4 Format Electronique