Tolérance aux pannes des circuits FPGAs à base de mémoire SRAM

Farid Lahrach

Thèse de doctorat de l'UTT

Farid LAHRACH

Tolérance aux pannes des circuits FPGAs à base de mémoire SRAM

Spécialité :
Optimisation et Sûreté des systèmes

2016TROY0028

Année 2016

utt
université de technologie
Troyes

THESE

pour l'obtention du grade de

DOCTEUR de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

Spécialité : OPTIMISATION ET SURETE DES SYSTEMES

présentée et soutenue par

Farid LAHRACH

le 30 septembre 2016

Tolérance aux pannes des circuits FPGAs à base de mémoire SRAM

JURY

Mme L. MOKDAD	PROFESSEUR DES UNIVERSITES	Présidente
M. M. AYAIDA	MAITRE DE CONFERENCES	Examinateur
M. J. BEN OTHMAN	PROFESSEUR DES UNIVERSITES	Rapporteur
M. É. CHÂTELET	PROFESSEUR DES UNIVERSITES	Directeur de thèse
M. S. NIAR	PROFESSEUR DES UNIVERSITES	Rapporteur

Farid Lahrach
Tolérance aux pannes des circuits FPGAs à base de mémoire SRAM .
Manuscrit soumis à l'Université de Technologie de Troyes (UTT) pour l'obtention du titre de Docteur de l'Université de Technologie de Troyes.
Ce mémoire ainsi que les figures qu'il contient peuvent être utilisés librement.
Manuscrit rédigé sous LaTeX avec la classe *classicthesis*.
E-mail : farid.lahrach.2013@utt.fr
Nombre de pages : 152.

Ces travaux de thèse ont été réalisés grâce au financement d'une allocation de recherche du département champagne-ardenne.

Remerciements

Cette thèse a été effectuée au sein du Laboratoire de Modélisation et Sûreté des Systèmes (LM2S) de l'Université de Technologie de Troyes (UTT). Je tiens à remercier les membres de ce laboratoire avant de commencer ce mémoire. Sans faire une liste exhaustive, je tiens à remercier particulièrement monsieur Pierre Koch, directeur de l'Université de technologie de Troyes et monsieur Régis Lengellé, directeur de l'école doctorale pour sa disponibilité et ses conseils précieux. Je remercie madame Véronique Banse et madame Marie-José Rousselet, secrétaires du pôle ROSAS. Je remercie aussi madame Pascale Denis et madame Isabelle Leclercq secrétaires de l'école doctorale.

Je tiens à remercier particulièrement monsieur Eric Châtelet, professeur des universités à l'UTT, directeur du projet MBSIE dont lequel s'inscrivent ces travaux. Je le remercie pour avoir dirigé cette thèse je le remercie pour sa disponibilité. Qu'il soit assuré de ma reconnaissance.

J'exprime ma profonde gratitude envers monsieur Smaïl Niar, professeur des universités, Université de Valenciennes et du Hainaut Cambrésis, UVHC et monsieur Jalal Ben-Othman, professeur des universités, Université Paris 13 pour avoir accepté de rapporter le présent mémoire. Leur rapports témoignent du temps qu'ils y ont consacré.

J'exprime ma reconnaissance envers madame Lynda Mokdad professeur des universités, Université de Paris-Est et monsieur Marwane Ayaida, maître de conférences, Université de Reims d'avoir accepté d'examiner cette thèse.

Il serait déplacé de ma part de ne pas remercier très sincèrement monsieur Driss Moussaid, monsieur Elhassan Chadli et monsieur Abdelhak Ziyyat, professeurs à la Faculté des sciences d'Oujda (Maroc), qui m'ont ouvert les portes du monde fascinant de l'électronique. Je remercie mes amis Chakib, Abdelghani, Mohamed et Mickael.

Un remerciement chaleureux à celle qui a ensoleillé ma vie, à la plus belle rencontre de ma vie, à toi *Hiba*.

Je remercie ma petite sœur *Nassima*, merci ma sœur d'avoir été à mes côtés dans les moments les plus difficiles.

Last, but not least, j'exprime ma plus sincère sympathie à l'ensemble des personnes qui m'ont accompagné durant cette thèse.

À mes parents,

À mes frères et sœurs,

À mon fils Malek, Malek tu as changé ma vie !

Résumé

Les circuits FPGAs à base de mémoire SRAM, ont la réputation de permettre à l'utilisateur une conception rapide, fiable et simple. La régularité de leur architecture ainsi que leur capacité de reconfiguration en temps d'exécution fournit des opportunités intéressantes à la fois pour la détection des défauts et le développement de nouvelles méthodes de tolérance aux pannes. Cette thèse possède trois principaux axes de recherche :

- **Le premier axe** porte sur deux techniques de tolérance aux pannes permanentes des FPGAs à base de mémoire SRAM. Le principe de la première technique repose sur le développement d'une méthode qui permet de tolérer plusieurs pannes dans plusieurs blocs de logique configurables (CLB). Cette méthode est construite à partir de la technique TMR combinée avec la technique master-slave. L'élément clé de notre approche pour la tolérance aux pannes est la reconfiguration partielle du FPGA en réponse à une ou plusieurs pannes. Le principe de la seconde technique repose sur la tolérance aux pannes des cellules SRAM dans les trames de configuration. L'idée consiste à réarranger les trames de configuration afin d'éviter les cellules SRAM défectueuses.
- **Le deuxième axe** de recherche porte sur la connaissance des bits critiques de configuration d'un design implémenté dans un FPGA en vue de prendre la décision adéquate sur la bonne méthode d'atténuation des SEUs. En effet, la réduction des dimensions des transistors entraîne une augmentation de la sensibilité des FPGAs face aux particules énergétiques chargées présentes dans les environnements d'exploitation de ces circuits. Si on ajoute à cela la tendance actuelle favorisant l'utilisation des FPGAs pour des applications spatiales et aéronautiques, on comprend alors la nécessité de trouver des méthodologies permettant de durcir les applications implémentées dans ces circuits pour faire face à ces particules chargées.
- Les FPGAs à base de mémoire SRAM sont configurables via leur mémoire de configuration constituée de trames. **Le troisième axe** de recherche de cette thèse porte sur le développement d'une méthodologie d'auto-test intégré. Cette méthodologie est basée sur des tests spécifiques, appelés algorithmes March, qui permettent la détection de pannes intra-mots dans les trames de configuration d'un FPGA à base de mémoire SRAM. Ces tests présentent l'avantage de bénéficier d'une implémentation rapide et d'obtenir un taux de couverture élevé.

MOTS CLEFS :

Réseaux logiques programmables par l'utilisateur, Tolérance aux fautes (ingénierie), Redondance (ingénierie), Autotest intégré, Fiabilité.

Abstract

From the outset, SRAM-based FPGAs, had the reputation to provide to the user a fast, reliable, and simple design. The regularity of their architecture and their ability of run-time reconfiguration provide interesting opportunities for defect detection, diagnosing and fault tolerance. This thesis focuses on three principal areas of research :

- **The first research area** concerns two techniques of permanent fault tolerance of SRAM-based FPGAs. The first technique's principle is based on the development of a method that permits tolerating multiple logic faults in different configurable logic blocks (CLB). This method is based on the TMR technique combined with master-slave technique. The key element of our approach to fault-tolerance is partially reconfiguring the FPGA to an alternate configuration in response to faults. The second technique's principale is based on the fault tolerance of SRAM cells in configuration frames. The main idea here is to rearrange the configuration frames in order to avoid faulty SRAM cells.
- Reducing the dimensions of transistors results in increased sensitivity of SRAM-based FPGAs face to energetic particles present in the operating environments of these circuits. This combined with the current trend favouring the use of these components for space and aeronautic applications has spurred much research on methodologies to harden applications implemented in these circuits against radiation. **The second topic** concerns on knowledge of critical bit information of a design implemented in an SRAM-based FPGA in order to make the appropriate decision on the chosen SEU mitigation technique.
- SRAM-based FPGAs are configurable via its configuration memory array which consists of configuration frames. **The third part** of this thesis focuses on developing a built-in self-test based on specific tests, called March algorithms, which detect intra-word faults in configuration frames. These tests have the advantage to benefit from fast implementation and achieve high fault coverage.

KEYWORDS :

Field programmable gate arrays, Fault tolerance (Engineering), Redundancy (Engineering), Built-In Self-Test, Reliability.

Table des matières

Table des figures

Liste des tableaux

Glossaire des notations

ALU	Arithmetic Logic Unit	13
ASIC	Application Specific Integrated Circuit	8, 10, 12
ATE	Automatic Test Equipment	34
BIST	Built-In Self-Test, Autotest intégré	84, 89
BOM	Bit-Oriented Memories	88, 96
CAO	Calcul Assister par Ordinateur	37
CF	Coupling Faults	95
CIP	Configurable Interconnect Point	21, 35, 72
CISC	Complex Instruction Set Computer	10
CLB	Configurable Logic Block	125
CMA	Configuration Memory Array	4, 35, 70, 88, 89, 108, 114
COTS	Components Off-The-Shelf	70, 75
DB	Data Background	96
DSM	Deep Sub-Micron	31
DSP	Digital Signal Processor	10
DUT	Device Under Test	122
EPROM	Erasable Programmable Read Only Memory	125
FPGA	Field Programmable Gate Arrays, Réseaux logiques programmables par l'utilisateur	1, 8, 28–30, 34, 35, 37, 39, 41, 43–45, 119
FT	Fault Tolerance	2
HDL	Hardware Description Language	119

HEP	High Energy Physics	70
ICAP	Internal Configuration Access Port	4, 69, 82–84, 88, 108
IEEE	Institute of Electrical and Electronics Engineers	32
JTAG	Joint Test Action Group	32
LET	Linear Energy Transfert	76
LUT	Look-Up Table	3, 13, 19, 29
MCM	Multi-Circuits Model	32
MFA	Model-Free Adaptive	122
MIPS	million d'opérations par seconde	11
MMU	Memory Management Unit	121
MST	Master-Slave Technique	3, 55, 67
MSU	Master-Slave Unit	3, 67
OTP	One Time Programmable	10, 71
PAR	Place And Route	18, 120
PID	Proportional Integral Derivative	122
RISC	Reduced Instruction Set Computer	10
RSoC	Reconfigurable System on Chip	9
RTR	Run-Time Reconfiguration	7, 23
SB	Switch Blocks	20, 30, 72, 89
SEE	Single event effect	71
SEUs	Single Event Upsets	3, 4, 35, 52, 70
SoC	System on Chip	1
SRAM	Static Random Access Memory	1
TMR	Triple Modular Redundancy	1, 53
VDSM	Very Deep Sub-micron	32, 71

VHDL	Very-High-Speed Integrated Circuit Hardware Description Language	119
VLSI	Very Large Scale Integration	31
WOM	Word-Oriented Memories	88, 95, 99, 100

1 Introduction générale

« . . . the perception of testability is highly dependent on one's involvement with testing. For example, a design engineer might consider the complexity of test vector generation as a measure of circuit testability. To a test engineer, testability may mean compatibility of design with the test equipment. Quality engineers often relate testability to fault coverage. . . . Testability is the property of a circuit that makes it easy (and sometimes possible !) to test.»
– Agrawal and Seth [1][2]

Dans un monde caractérisé par des perpétuels exigences et soumis à une concurrence aveugle, les entreprises comme les laboratoires de recherche ont recours à l'innovation, catalyseur de la recherche et du développement, pour assurer leur croissance voir leur existence. Le domaine de l'électronique avec toutes ses branches constitue la pierre angulaire de toute innovation quelque soit le domaine d'application.

Depuis l'invention du transistor bipolaire, l'électronique a gagné une place très importante dans une large gamme de domaines scientifiques et technologiques. Cette propriété a beaucoup contribué à renforcer l'évolution de l'électronique en particulier l'électronique numérique, une partie importante de l'électronique qui correspond au cœur des systèmes embarqués.

Les systèmes embarqués monopuces (System on Chip, SoC) sont des systèmes agrégeant une partie électronique et logiciel formant un système autonome et temps réel (RT). Ils sont formés essentiellement de plusieurs processeurs et mémoires pour répondre aux exigences des applications modernes comme la téléphonie et les réseaux de télécommunication. En un temps minime, les systèmes embarqués sont devenus la boussole qui dirige l'industrie des semi-conducteurs. Pour réaliser de tels systèmes, toute en respectant les contraintes et les objectifs de la fiabilité de ces SoCs, les concepteurs disposent des processeurs ou des circuits ASICs. Ces deux moyens sont inefficaces devant la conception des SoCs dotés de la reconfiguration dynamique, caractéristique permettant aux SoCs de fonctionner efficacement dans des conditions changeables.

Une troisième solution est apparue il y a une trentaine d'années rassemblant les avantages des microprocesseurs et ceux des circuits ASICs pour la réalisation des SoCs. Il s'agit des circuits reconfigurables représentés par les FPGAs (Field Programmable Gate Array) à base de mémoire SRAM. Les FPGAs à base de mémoire SRAM disposant de la technologie de reconfiguration dynamique partielle sont vite sortis du cercle de conception pour devenir un outil indispensable dans divers domaines grâce à leur caractéristique principale qui est la reconfiguration. La reconfiguration et la reconfiguration partielle statique ou dyna-

mique ont constitué la base des applications infranchissables par les microprocesseurs et les ASICs.

Le contenu de ce mémoire est quelque peu hétéroclite des travaux présentés, positionnés entre la tolérance aux pannes des FPGAs à base de mémoire SRAM et les tests et diagnostic des pannes dans la mémoire de configuration de ces circuits configurables. La tolérance aux pannes des circuits FPGAs à base de mémoire SRAM porte sur les pannes permanentes et les pannes transitoires (SEUs).

1.1 Contexte et problématique

La tolérance aux pannes des FPGAs à base de mémoire SRAM est la capacité de ces derniers à fonctionner comme prévu malgré la présence des défauts (*defect*). Ce type de fiabilité est généralement atteint grâce à la redondance modulaire. Or cette solution ne présente pas de rentabilité sur le plan de la consommation en puissance et sur celui des ressources logiques ajoutées au design pour réaliser la redondance modulaire. En effet, la première méthode utilisée pour la tolérance aux pannes des FPGAs à base de mémoire SRAM est la redondance modulaire triple (TMR) [3]-[7]. Cette méthode ne constitue pas un intérêt pour la tolérance aux pannes des FPGAs étant donnée sa forte consommation en puissance et en ressources logiques [8]. Les FPGAs de technologie SRAM fournissent un moyen de grande valeur à la conception des systèmes sur puce (SoC), mais ces architectures ne sont pas dépourvues d'inconvénients. En effet, les pannes sont devenues de plus en plus prononcées dans les applications émergentes à base de FPGAs de technologie SRAM, qu'il s'agisse de pannes permanentes dues aux processus d'intégration super scalaire et au vieillissement du circuit ou de pannes transitoires dues à l'exploitation de ces architectures dans des environnements nécessitant un grand degré de fiabilité.

Les FPGAs de technologie SRAM sont dotés de propriétés intrinsèques telles qu'un parallélisme matériel réel et plusieurs modes de reconfiguration, comme la reconfiguration partielle, la reconfiguration dynamique, la reconfiguration statique ou une combinaison de ces modes de reconfiguration. Ces caractéristiques ont favorisé l'utilisation de nouvelles méthodes et techniques de tolérance aux pannes des FPGAs de technologie SRAM. Il suffit par exemple de reconfigurer partiellement un FPGA-SRAM en une nouvelle configuration alternative qui contourne une ou plusieurs pannes. Le domaine de la tolérance aux pannes (FT) des FPGA-SRAMs est un domaine dont l'apparition a accompagné l'apparition des FPGA-SRAMs dans les années 1980. Il est devenu une partie intégrante de la conception ou l'utilisation des FPGA-SRAMs, c'est un domaine en évolution proportionnelle à celle de ces architectures reconfigurables. De ce fait les termes utilisés dans ce domaine ne décrivent pas les mêmes concepts que dans d'autres domaines. Il est donc judicieux d'inclure dans l'état de l'art de ce manuscrit des définitions des termes comme "*défaut*" ou (defect) en anglais ou comme le terme "*panne*" ou (fault).

Le procédé de fabrication et la densité d'intégration des FPGAs de technologie SRAM est maintenant proche d'un point où il est impossible de produire des FPGA-SRAMs exempts de défauts. En effet, le processus de fabrication des composants semi-conducteurs est en constante évolution en termes de diminution de la géométrie des transistors, d'alimentation et d'augmentation de la vitesse et de la densité logique. À titre d'exemple, la densité d'intégration des FPGA-SRAMs de la famille Virtex-7 a atteint 6,8 milliards [9]. De ce fait les FPGAs de technologie SRAM sont devenus plus vulnérables aux pannes permanentes et

transitoires (Chapitre 5). Le but fixé lors du commencement de cette thèse était tout d'abord l'étude des techniques utilisées dans le domaine du test et de la tolérance aux pannes des FPGA-SRAMs. Ensuite, présenter de nouvelles techniques FT ainsi que des méthodes de test des FPGA-SRAMs tout en respectant les contraintes suivantes :

1. le coût matériel (Area overhead) et le coût en timing de la technique de tolérance aux pannes (FT)
2. l'impact des performances sur le système FT implémenté dans le FPGA-SRAM
3. l'efficacité de la technique FT en présence de plusieurs pannes
4. le taux de couverture de pannes

La tolérance aux pannes est un paramètre essentiel dans le domaine des FPGAs de technologie SRAM vu qu'aucun composant électronique n'est fiable à 100%. Le fabricant des composants électroniques, en l'occurrence les FPGA-SRAMs, définit, après des tests comparables à des utilisations plus ou moins sévères, des critères de tolérance aux pannes du FPGA-SRAM. L'étude évoquée dans [10] a prédit que les futures FPGAs de technologie SRAM à échelle d'intégration de moins de 45 *nm* auront un rendement moins faible et la tolérance aux pannes sera donc indispensable pour ces circuits reconfigurables. La tolérance aux pannes des architectures reconfigurables se divise en deux catégories selon [11]-[13] : tolérance aux pannes au niveau matériel et tolérance aux pannes au niveau configuration.

1.2 Description du problème abordé

La vie moderne dépend de plus en plus des systèmes automatisés. Lorsque ces systèmes ne parviennent pas à fonctionner "normalement", la vie et les systèmes peuvent être en danger. Les menaces incluent les catastrophes naturelles, les erreurs humaines et les attaques intentionnelles ainsi que les défaillances matérielles. Dans tous les cas, l'attention aux modes de défaillance à travers le cycle de vie du système peut permettre à l'utilisateur de faire face à ces menaces [14]-[16]. Devant cette réalité, la tolérance aux pannes est devenue un paramètre essentiel pour assurer la sûreté de fonctionnement des systèmes, y compris les systèmes à base des FPGAs de technologie SRAM. Ainsi, l'utilisation croissante des FPGAs à base de mémoire SRAM est sujette aux questions suivantes :

1. La capacité de calcul a augmenté à un rythme exponentiel [17]. L'augmentation des capacité des FPGAs à base de mémoire SRAM et les exigences au niveau puissance/performance imposent de combiner plus de fonctionnalités dans un seul FPGA à base de mémoire SRAM. Est il possible de trouver une technique de tolérance aux pannes qui permettra de tolérer plusieurs pannes dans des régions différentes d'un FPGA de technologie SRAM avec un moindre coût possible ?
2. La géométrie des transistors est en rétrécissement croissant dans les processus d'intégration des FPGAs à base de mémoire SRAM. Autrement dit, le seuil de l'énergie d'une particule qui pourrait changer un ou plusieurs bits de configuration a diminué. Comment combiner entre les techniques d'atténuation des SEUs et l'architecture des FPGAs à base de mémoire SRAM pour faire face à ces pannes transitoires ?
3. Toutes les techniques de tolérance aux pannes requièrent des techniques de test afin de localiser les ressources défectueuses. Une technique d'auto-test intégré s'avère donc indispensable pour un taux de couverture élevé de pannes.

1.3 Organisation du manuscrit

Outre le chapitre 2 qui présente brièvement les principales architectures reconfigurables et un état de l'art critique sur l'architecture et le fonctionnement des FPGAs à base de mémoire SRAM, le présent mémoire est constitué des quatre principaux chapitres suivants :

- Le chapitre 3 est dévolu à l'étude de différentes techniques et méthodologies de test des FPGAs à base de mémoire SRAM, ainsi qu'aux techniques de tolérance aux pannes de ces circuits. La tolérance aux pannes des FPGAs à base de mémoire SRAM est un paramètre essentiel à la fiabilité d'un système à base de FPGAs de technologie SRAM. Un état de l'art sur la tolérance aux pannes et techniques de test des FPGAs à base de mémoire SRAM est développé. Les principaux modèles de pannes dans les circuits FPGAs sont présentés. L'objectif de cet état de l'art est de pointer principalement l'exploitation très limitée de chaque méthode proposée. Une comparaison des méthodes de test ainsi que des méthodes de tolérance aux pannes des FPGAs est présentée dans l'annexe B.
- L'exploitation pratique des méthodes et tests des FPGAs de technologie SRAM étudiées dans le chapitre 3 est sujette à la connaissance de l'architecture ciblée dans un FPGAs (la partie logique ou les ressources de routage). La première approche proposée dans le chapitre 4 consiste à combiner la technique de la *redondance modulaire triple* (TMR) avec la technique *master-slave* (MST) afin de tolérer plusieurs pannes dans plusieurs CLBs et donc tolérer un nombre élevé de LUTs. Le design physique est partitionné en un ensemble d'unités master-slave (MSUs). Contrairement aux méthodes de conception dans les ASICs et dans les microprocesseurs, qui résultent à des structures statiques, cette technique permet d'obtenir plusieurs configurations de la même fonction implémentée dans une même partie physique d'un FPGA-SRAM. Pour valider les résultats théoriques de la méthode proposée, des expériences ont été menées sur des designs ITC'99 implémentés sur la carte d'évaluation ML507 de Xilinx montrent la fiabilité de la méthode avec un coût raisonnable en terme de timing et ressources logiques.

  La deuxième approche proposée dans ce chapitre est une technique de tolérance aux pannes qui est basée sur le réarrangement des cellules SRAM dans les trames de configuration afin de tolérer les cellules SRAM défectueuses. Cette approche permet la réutilisation partielle des ressources logiques, la tolérance aux pannes des ressources logiques ainsi que de routage. Une analyse de la fiabilité de cette méthode est également réalisée.
- Afin de permettre une atténuation des effets singuliers (SEUs) rencontrés dans un environnement où règnent des particules chargées, une analyse rigoureuse de la partie matérielle et logicielle, dans le but d'extraction de l'information sur les bits critiques, est nécessaire. Le chapitre 5 propose donc une étude d'une stratégie de prédiction et d'extraction des informations sur les bits critiques dans la mémoire de configuration d'un FPGA-SRAM. L'utilisation conjointe des méthodes d'atténuation des pannes transitoires et d'exploitation des informations sur les bits potentiellement critiques basée sur une étude statistique et matérielle des designs implémentés dans le FPGA-SRAM Virtex-5 XC5VFX70T dans la carte d'évaluation ML507, permet de faire la distinction entre les cellules SRAM qui sont potentiellement critiques pour un design et les cellules

SRAM qui ne le sont pas. Cette distinction permet aux techniques d'atténuation des SEUs de se focaliser sur les cellules SRAM critiques dans le but de garder le contenu de ces cellules statiques vis-à-vis des pannes SEUs.

– Enfin, nous proposons dans le chapitre 6 une exploitation des tests *March* pour le test et le diagnostic de la mémoire de configuration (CMA) d'un FPGA à base de mémoire SRAM. En effet, les FPGAs utilisent des millions de cellules SRAM, qui forment la mémoire CMA, pour implémenter un design. Étant donné leur complexité linéaire, leur couverture complète des pannes ciblées et leur facilité de conversion vers un test orienté mot, les tests *March* sont les plus adaptés pour le test de la mémoire CMA. La conversion consiste à concaténer les tests *March* pour les pannes inter-mots et les tests *March* pour les pannes intra-mots. Le circuit BIST proposé est construit d'une logique fixe comprenant une conception qui utilise le port ICAP pour écrire et lire les données dans les trames de configuration.

Les travaux présentés se veulent avoir une finalité applicative. Aussi quelques résultats numériques validant les résultats établis sont présentés et discutés.

Remarque 1.1.

1. *Les travaux présentés dans ce manuscrit de thèse s'intéressent uniquement aux FPGAs à base de mémoire SRAM. Dans tout le reste de ce manuscrit les FPGAs à base de mémoire SRAM, les FPGAs de technologie SRAM ou les FPGAs désignent la même technologie. Ils seront souvent notés FPGA-SRAM.*
2. *Aussi dans ce manuscrit une conception FPGA, un design, un design physique, une conception ou une conception FPGA désignent un projet logiciel codé en un langage de description matériel (VHDL, Verilog, SystemC, ...) et qui sera implémenté dans un FPGA-SRAM.*
3. *Dans ce manuscrit nous nous focalisons sur les architectures Xilinx pour étudier les architectures des FPGA-SRAMs.*

Première partie

LES FPGA À BASE DE MÉMOIRE SRAM ET LA TOLÉRANCE AUX PANNES

Les FPGA-SRAMs : un élément poromoteur de l'informatique reconfigurable

2

« *Understanding why FPGAs can be efficient and where they are most efficient provides additional insight into where we should use FPGAs and how to fully exploit their strengths.*»
André DeHon, Department of Electrical and Systems Engineering University of Pennsylvania [18].

Sommaire

Introduction au chapitre :

L'évolution exponentielle de l'informatique moderne a confronté les systèmes embarqués au *challenge* d'adaptabilité à cette évolution. Cette adaptabilité dépasse les performances des circuits spécifiques et la flexibilité des processeurs et exige un système embarqué capable de permettre la redéfinition de ses fonctionnalités. Les architectures reconfigurables en l'occurrence les FPGA-SRAMs répondent à cette exigence grâce à l'aspect configurable de ces architectures. Le milieu des industriels ayant été convaincu de leur intérêt, des champs applicatifs, de l'automobile à l'exploration spatiale, se sont ouverts pour exploiter les avantages des FPGA-SRAMs. Cette omniprésence des FPGAs à à développer ces plateformes en ajoutant plusieurs fonctionnalités afin que ces dernières restent adaptables à leur environnement d'exploitation et aux nouveaux standards.

Notre conviction que l'étude des FPGA-SRAMs et de leurs points forts ainsi que de leurs lacunes est la brique de base pour commencer notre thématique de recherche, nous avons débuté ce manuscrit par un chapitre explorant ces architectures. Ce chapitre se veut donc un état de l'art sur les architectures reconfigurables et en particulier les FPGA-SRAMs.

2.1 Motivation

Le domaine de l'électronique constitue une condition sine qua non pour toute innovation scientifique et technologique. En effet, tous les axes de recherches scientifiques et technologiques ont besoin des plateformes électroniques pour l'acquisition et le traitement de données de façon précise et rapide. De ce fait, toute réussite scientifique nécessite une innovation dans les systèmes électroniques, voir une révolution.

L'informatique reconfigurable se positionne, depuis une dizaine d'années, au cœur des systèmes embarqués. En effet, ce nouveau domaine qui se différencie des microprocesseurs et les solutions spécifiques (ASIC) par la propriété de doter le circuit électronique d'une forme de reconfiguration matérielle (hardware reconfiguration). L'idée innovante dans le domaine des architectures reconfigurables est l'utilisation du même circuit électronique pour implémenter et exécuter des algorithmes différents sur une même plateforme électronique. Les FPGAs à base de mémoire SRAM se situent en tête des architectures reconfigurables car ils offrent aux concepteurs la possibilité de concevoir des systèmes électroniques de façon rapide et moins coûteuse par rapport aux autres alternatives grâce à leur caractéristique de reconfiguration matérielle universelle.

Depuis leur invention dans les années quatre vingt par la société *Xilinx*, les FPGA-SRAMs occupent de plus en plus tous les domaines scientifiques et technologiques. Leur omniprésence est dû essentiellement aux propriétés suivantes :

1. **La reconfiguration universelle** : Les FPGA-SRAMs sont capables de modifier leur fonctionnalité en changeant leur configuration décrite par un langage de description matériel (HDL).

2. **La reconfiguration partielle** : Les travaux de recherches ont beaucoup avancé sur l'aspect reconfiguration des FPGA-SRAMs. En effet, les FPGA-SRAMs sont capables d'effectuer une reconfiguration partielle statique ou dynamique pour accomplir certaines tâches avec un coût minimum au niveaux Timming et ressources matérielles.
3. **Le parallélisme** : les FPGA-SRAMs permettent l'exploitation du parallélisme potentiel de l'application implémentée dans le FPGA-SRAM.

Ces trois caractéristiques font parti des caractéristiques qui ont rendu les FPGA-SRAMs omniprésents dans le champs de la recherche scientifique ainsi que dans le champs de l'industrie. En effet, nous vivons le début d'un ère où tous les moyens de la vie quotidienne ont des capacités de calculs et forment un ensemble connecté, ce qui concrétise le concept de l'*Evrywhere* [19][20].

Cependant la communauté des concepteurs des FPGA-SRAMs reste limitée par rapport à celle des concepteurs sur les microprocesseurs ou ASICs. Les FPGA-SRAMs récents dépassent la fréquence de 500 MHz et disposent d'une grande densité d'intégration, cela fait de ces circuits un choix important dans les domaines scientifiques et technologiques.

Dans ce chapitre nous cherchons à étudier les FPGAs à base de mémoire SRAM sous tous les angles dans le but de préparer le terrain pour aborder notre thématique de recherche. Nous étudions leurs architectures ainsi que leur évolution. Nous abordons aussi dans ce chapitre l'aspect reconfigurable des FPGA-SRAMs, vu l'importance qu'il occupe dans le domaine des architectures reconfigurables.

2.2 L'informatique reconfigurable

L'informatique reconfigurable (RC) est devenue une nouvelle technologie innovatrice pour accélérer le calcul parallèle. Que ce soit pour la communauté scientifique ou pour l'industrie, le calcul parallèle est devenu le paradigme dominant pour surmonter la haute complexité du calcul imposé par les algorithmes des nouveaux standards. Dans cette partie, nous explorons l'informatique reconfigurable, nous étudions ses avantages par rapport aux microprocesseurs et aux solutions spécifiques (ASICs) et nous terminons par discuter les limites de cette nouvelle technologie.

2.2.1 Informatique reconfigurable : Architectures

Dans le but de démocratiser l'informatique reconfigurable, plusieurs recherches ont été menées afin de trouver une plateforme électronique pour cette nouvelle informatique. Les alternatives proposaient, dans ce contexte, variaient entre autres, les FPGA-SRAMs vu leurs avantages économiques et les solutions spécifiques vu leurs rapidité dans le calcul. Les axes guidant ces recherches pour le choix de la plateforme électronique la plus adaptée à l'informatique reconfigurable et les effets de ce choix sur les performances de cette technologie sont basées sur les points suivants [18] :

1. Traiter le réseau configurable comme un coprocesseur avec un processeur qui contrôle les données pour effectuer des opérations spécifiques, ou configurer le réseau configurable comme une unité fonctionnelle [21].
2. Quelle est le type de la granularité adéquat à l'informatique reconfigurable ?

Le premier point traite le caractère reconfigurable de ces architectures qui peuvent être statiquement ou dynamiquement reconfigurables. Selon [18] une configuration statique n'intervient qu'entre les cycles d'exécution des applications, tandis qu'une reconfiguration dynamique permet la mise à jour complète ou partielle durant l'exécution de l'application.

Une architecture reconfigurable est caractérisée par une symétrie architecturale, et donc peut être organisée sous forme des cellules (*Tiles*) ayant les mêmes fonctionnalités comme les CLBs pour les FPGA-SRAMs de la technologie *Xilinx* et les LEs (Logic Elements) pour les FPGA-SRAMs de la technologie *Altera*. Donc, la granularité utilisée pour les architectures reconfigurables est la pierre angulaire du second axe abordé dans cette partie. Dans la suite, nous étudions les types de granularité utilisés dans les architectures reconfigurables.

2.2.2 Granularité des architectures reconfigurables

Comme nous l'avons évoqué dans le paragraphe précédent, la granularité des ressources de traitement est une caractéristique permettant la classification des architectures reconfigurables. [22] et [18] classifient les architectures reconfigurables selon le type de granularité des ressources de traitement suivants :

1. **Ressources de traitement à gros grain** : Les ressources de traitement à gros grain sont caractérisées essentiellement par le traitement massif des données tels que les ALUs et les MACs. Les architecture reconfigurables à gros grain sont utilisables dans des applications mobiles et bénéficient du caractère de la reconfiguration dynamique [22] [23]. Les architectures reconfigurables à ressources de traitement sont aussi utilisées dans des applications de contrôle de flux (Image, Vidéo, DSP, ...). En effet, une équipe du laboratoire CSAIL du l'Institut Massachusetts de Technologie (MIT) [24] a développé une architecture pour le domaine de contrôle du flot de données. L'architecture développée est un réseau 2-D de cœurs exécutants les mêmes tâches. Chaque cœur communique avec son voisin par quatre matrices de routage. En optimisant le parallélisme de l'architecture et en exploitant le pipelining dans l'exécution des instructions, l'équipe a démontré que l'architecture développée est 11,2 plus fois rapide qu'un seul cœur fonctionnant individuellement.

2. **Ressources de traitement multi-grains** : L'étude présentée dans [25] combine plusieurs types de grain pour des applications réseau. L'architecture rassemble des processeurs et des FPGA-SRAMs en exploitant la reconfiguration dynamique pour le déchargement des tâches de calcul intensif d'un processeur de réseau. L'architecture permet le remplacement dynamique de modules matériels ainsi que l'établissement d'une structure de communication interne. Plusieurs exemples d'architectures multi-grains ont été étudié dans [22].

3. **Ressources de traitement à grain fin** : Ce type d'architectures permet la manipulation des opérations *bit-à-bit* en augmentant ainsi le rendement au niveau ressources reconfigurables. Les FPGAs à base de mémoire SRAM sont les circuits les plus représentants de ce domaine d'architectures. Depuis l'invention des FPGA-SRAMs, les sociétés *Xilinx* et *Altera* ont multiplié les efforts afin d'élargir leur domaine d'application qui était restreint au prototypage des circuits ASICs.

Il est évident que la première remarque qu'on peut faire, c'est que les types des architectures reconfigurables présentées sont divergentes les unes par rapport aux autres. Néan-

moins, la caractéristique commune entre elles est la capacité de la reconfiguration de ces architectures. Dans ce qui suit, les FPGA-SRAMs seront l'objet de notre étude ainsi que leurs caractéristiques notamment la reconfiguration avec toutes ses aspects.

2.2.3 Évolution technologique des systèmes reconfigurables

L'étude menée dans [26] montre un écart grandissant entre la loi de densité d'intégration de Moore et la loi de complexité algorithmique de Schannon pour les applications mobiles. En outre, la capacité des batteries, autrement dit, la consommation de l'énergie ne suit pas l'évolution des processeurs ce qui exige d'autres alternatives. L'apparition des circuits spécifiques ASICs semblait résoudre le problème car ils offrent un maximum de performance et une consommation réduite de l'énergie [27]. Or, les ASICs sont des circuits dédiés, ils ne peuvent pas suivre l'évolution des standards technologiques. Cette absence de flexibilité pousse vers une nouvelle alternative qui est les architectures reconfigurables.

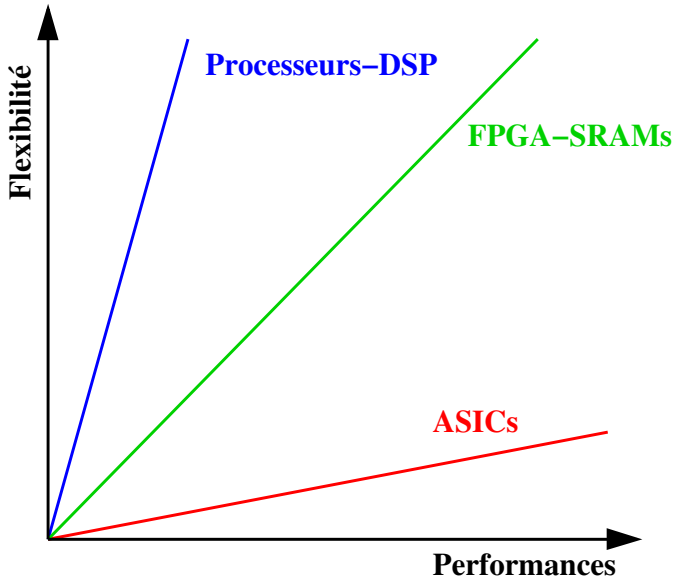


FIGURE 2.1: Schéma montrant le compromis entre les solutions programmables et les solutions spécifiques.

La figure 2.1 - adaptée de [22] et [26] - montre la position des architectures reconfigurables dans le domaine de l'électronique numérique. Ce compromis entre la flexibilité des processeurs et les performances des solutions spécifiques a attiré la grande préoccupation de la société *Xilinx* et *Altera*. En effet, depuis leur invention par *Xilinx*, les FPGA-SRAMs ont beaucoup évolué sur toutes les échelles : reconfigurabilité, performances, consommation d'énergie, parallélisme, coût, densité d'intégration, etc.

2.3 Architecture des FPGA à base de mémoire SRAM

Le progrès technologique que les FPGAs à base de mémoire SRAM ont connu a attiré l'attention de tous les chercheurs dans ce domaine. Cette attention particulière est justifiée se traduisait par un grand nombre de documents techniques et scientifiques étudiant leurs architectures, leurs modes de configuration et leur positionnement dans le domaine de

conception des systèmes C-SoCs [18]-[32]. Dans ce chapitre nous étudierons l'architecture des FPGA-SRAMs afin de présenter suffisamment de détails pour aborder notre thématique de recherche.

2.3.1 Architecture des éléments reconfigurables d'un FPGA-SRAM

Un FPGA à base de mémoire SRAM peut être vu théoriquement comme un réseau de blocs logiques configurables (CLBs). Comme montré dans la figure 2.2, les CLBs sont arrangés dans un réseau à deux dimensions (2D). La connexion entre les CLBs se fait à l'aide des lignes de connexion que se soit horizontalement ou verticalement. La connexion est gérée par des blocs de connexion (CB) et des matrices de routage (SB). Chaque lignes de connexion est constituée par un ensemble de segments qui relient les CLBs avec les blocs de connexion [33]. Ce réseau de CLBs est entouré par des blocs d'entrées/sorties (IOBs).

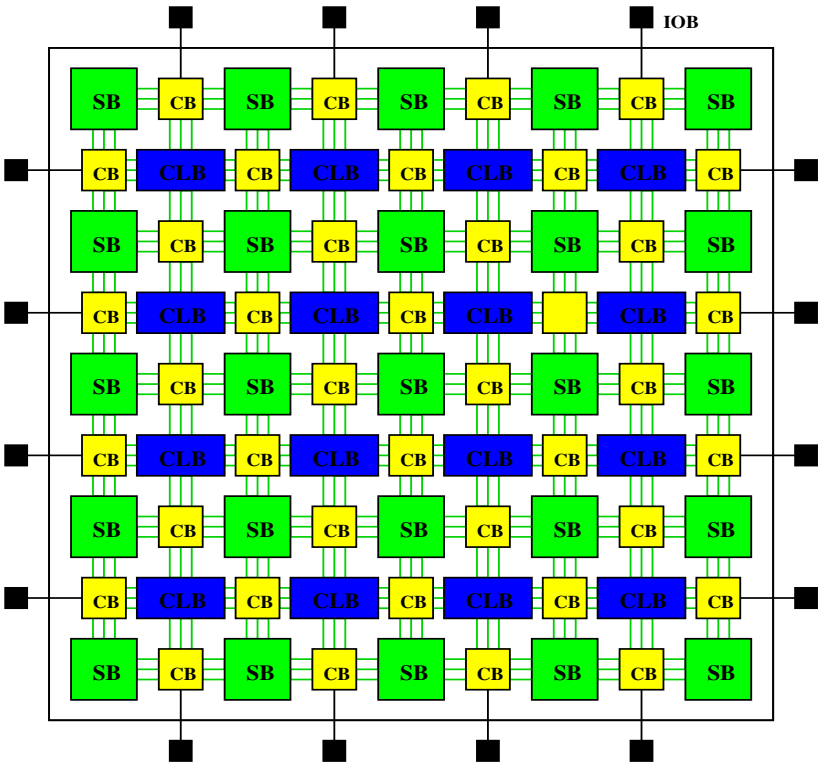


FIGURE 2.2: Structure générique de base d'un FPGA à base de mémoire SRAM constituée de 4×4 CLBs entourés par des blocs de connexion (CBs) et des matrices de routage (SBs).

Dans un FPGA-SRAM, plus de 80% de transistors sont dédiés au réseau de routage configurable comme les matrices de routage et les buffers [34]. Le FPGA-SRAM communique avec l'extérieur via les blocs d'entrées/sorties (IOBs). Un bloc de connexion (CB) est un élément de connexion permettant d'établir des connexions entre les CLBs et les autres ressources du FPGA-SRAM.

Les CLBs sont des éléments logiques d'un FPGA-SRAM. Ils implémentent les circuits combinatoires ainsi que les circuits séquentiels. Chaque CLB - selon la technologie *Xilinx* - dispose de deux *Slices*. Chaque *Slice* contient quatre tableaux combinatoires (LUTs) d'entrées allant jusqu'aux six entrées, quatre bascules de type DFFs et des multiplexeurs. La

figure 2.3 - adaptée de [35] - montre une architecture de base d'un élément reconfigurable (*Slice*) constituant un CLB dans FPGA-SRAM de la technologie *Xilinx*. Dans un FPGA à base de mémoire SRAM, les LUTs sont construites à base de cellules mémoire SRAM pour assurer l'aspect configurable du circuit et réagissent comme des générateurs de fonctions.

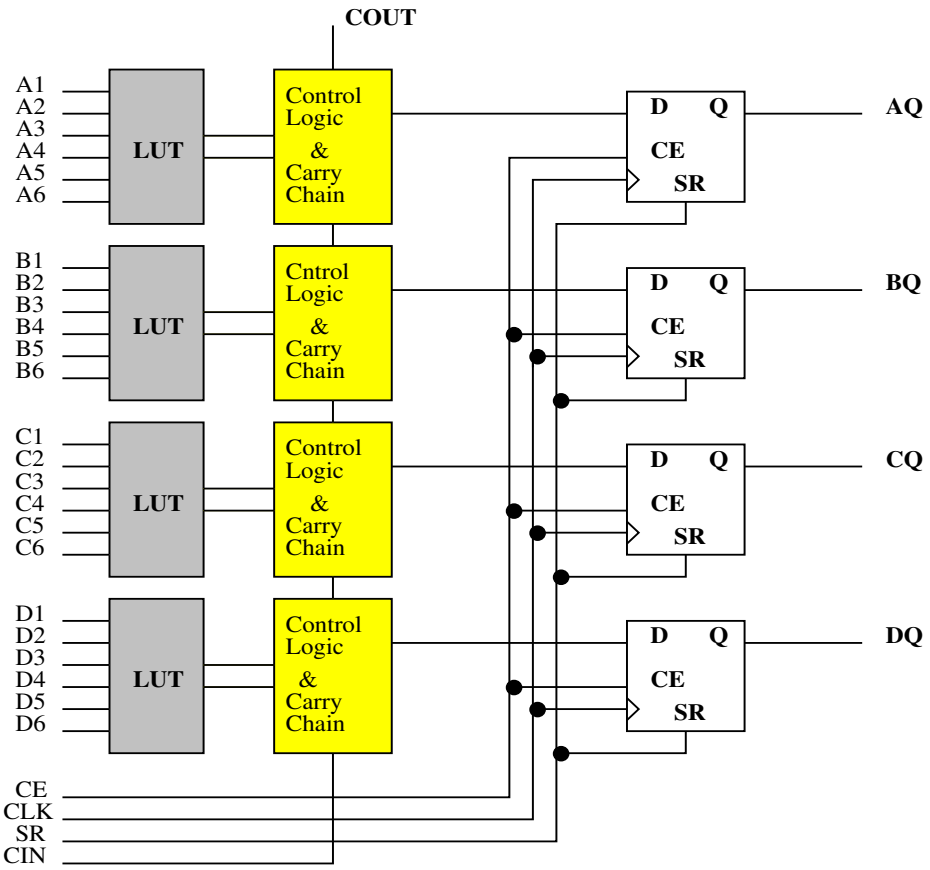


FIGURE 2.3: Architecture de base d'un élément reconfigurable d'un *Slice* constituant un CLB de la technologie *Xilinx*. Une connectivité locale et rapide peut être assurée en regroupant ces LUTs dans un *Slice*.

# Slices	# 6-LUTs	# DFFs	# chaînes de propagation	# RAM distribuées	# registre à décalage
2	8	8	2	256 bits	128 bits

TABLE 2.2: Les ressources logiques constituant un CLB.

L'élément configurable consiste en un bloc combinatoire (LUT) de six entrées pour les technologies les plus récentes. Une LUT réagit comme une mémoire de 6 et $2^6 = 64$ cases mémoire. Pour implémenter une fonction donnée, la table de vérité doit être dans cette mémoire. L'élément configurable possède deux sorties, une combinatoire et une seconde séquentielle via une bascule DFF. Le multiplexage entre les deux sorties selon la spécificité de la fonction implémentée se fait par le biais d'un multiplexeur MUX [35].

Le tableau 2.2 [35] présente un inventaire des ressources logiques constituants un CLB dans le FPGA-SRAM de *Xilinx* Virtex-5 [35]. Chaque CLB contient deux *slices* ayant chacun une chaîne de propagation de la retenue indépendante. Chaque *slice* dispose de quatre LUTs

de six entrées chacune et deux sorties. Les LUTs peuvent être configurées pour opérer comme des registres à décalage. Plusieurs LUTs peuvent être combinées de différentes manières pour former une mémoire RAM distribuée pour des applications qui requièrent des mémoires de ce type.

Type du FPGA-SRAM	**# CLBs** *Ligne* × *Colonne*	**# LUTs**	**# DFFs**	**# max de E/S**
Xilinx Virtex-7	68450	547600	1095200	300
XC7VH870T	-	6-LUTs	-	-
Xilinx Virtex-6	44280	354240	708480	720
XC6VHX565T	-	6-LUTs	-	-
Xilinx Virtex-5	25920	207360	207360	960
XC5VLX330T	240 × 108	6-LUTs	-	-
Xilinx Virtex-4	16128	126336	126336	896
XC4VFX140	192 × 84	4-LUTs	-	-

TABLE 2.3: Évolution de l'architecture du FPGA-SRAM de la famille Xilinx Virtex.

Le tableau 2.3 [36] montre l'évolution technologique de l'architecture du FPGA-SRAM. En effet, les FPGA-SRAMs les plus récents disposent d'un grand nombre de CLBs tirants profit d'une nouvelle technologie d'intégration. En efft, comme prédit par [37], la miniaturisation des systèmes micro-électroniques a atteint ses limites. Pour répondre aux exigences des nouvelles applications technologiques, les systèmes électroniques étaient contraints de changer la technologie d'intégration vers la technologie (3-D IC) [38]. Les concepteurs ont profité de cette technologie d'intégration pour donner naissance à des FPGA-SRAMs de la famille *Virtex-7 UltraSCALE* ou la famille *Kintex UltraSCALE* avec une densité d'intégration de 20*nm* [39].

2.3.2 Architecture du système de routage dans un FPGA-SRAM

Lors de l'implémentation d'une conception FPGA, le FPGA-SRAM utilise les CLBs pour implémenter les fonctions logiques ainsi que des blocs mémoires (BRAMs), des blocs DSPs, des blocs IOBs, etc. Le FPGA-SRAM a besoin donc d'une architecture de routage pour établir une communication entre les composants utilisés dans cette conception ainsi que de fournir une connexion entre les blocs d'E/S (IOBs) et les ressources logiques utilisées.

Les FPGA-SRAMs utilisent des cellules SRAMs pour configurer un réseau de routage. Les ressources de routage occupent une place importante dans les architectures reconfigurables en l'occurrence les FPGAs à base de mémoire SRAM [33]. En effet, les ressources logiques dans un FPGA-SRAM sont entourées par des lignes horizontales et verticales. Cette hiérarchie de connexion est configurable grâce au cellules mémoires SRAMs qui forment les matrices de connexion. Par la configuration de ces cellules SRAMs via le *Bitstream*, on établira des connexions entre les différentes ressources logiques utilisées dans une conception FPGA. La formation d'une ligne de connexion se fait par la connexion de plusieurs segments via des points d'interconnexion configurables (CIP).
La figure 2.4 montre un schéma de base d'une matrice de routage. Elle est constituée essentiellement par des points CIP qui assurent à leurs tour le routage. Chaque point CIP

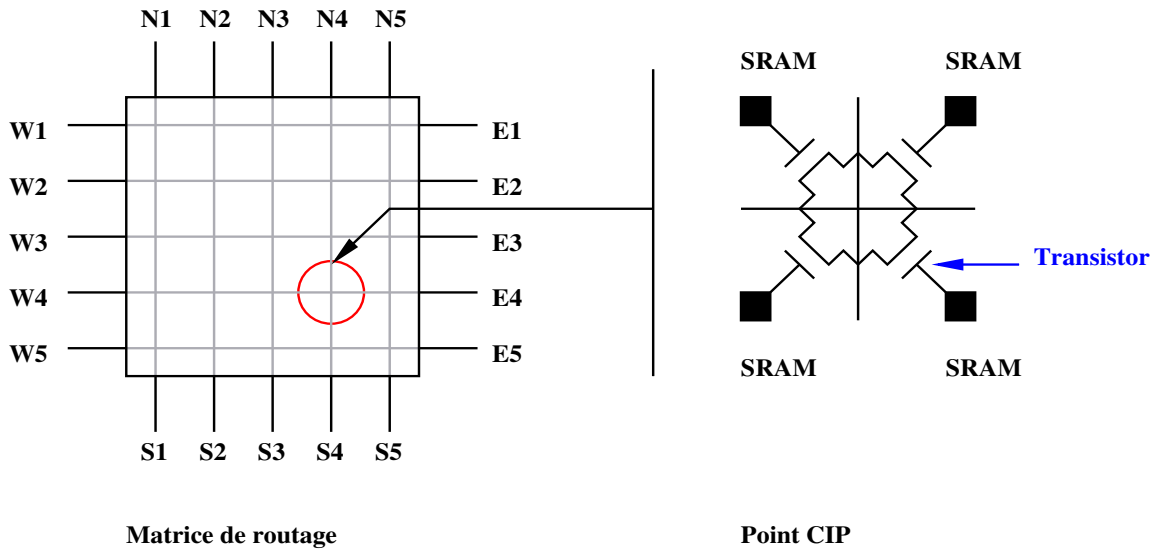


FIGURE 2.4: Structure générique de base d'une matrice de routage et un point CIP. Un point CIP est utilisé pour connecter deux lignes.

présente plusieurs possibilités de connexion grâce aux cellules SRAM qui en dispose. L'activation de plusieurs points CIPs dans certains ordre permet de relier plusieurs segments et donc la constitution de longs chemins. Pour palier les retards que font les signaux à travers les longs chemins constitués pas des courts segments et des points CIPs, les FPGA-SRAMs disposent aussi des longs segments qui relient plusieurs CLBs [33].

Dans un FPGA-SRAM on trouve typiquement trois techniques pour établir un réseau d'interconnexion entre les ressources logiques utilisées dans une conception FPGA [33][34] :

1. **Lignes d'interconnexion à usage général** : Ce sont des lignes de connexion verticaux et horizontaux formant ainsi un réseau de connexion entre les ressources logiques et les IOBs. Les matrices de routage permettent la configuration des lignes nécessaires pour établir le réseau d'interconnexion nécessaires pour la conception FPGA. Le reste du réseau n'est pas configuré donc ne participe pas à l'implémentation de la conception FPGA. Des ressources logiques spéciales sont déployées afin d'améliorer les performances des lignes d'interconnexion à usage général.

2. **Lignes de connexion directe** : Dans le but d'augmenter les performances d'une conception, les FPGA-SRAMs utilisent des lignes de connexion directe. Ces lignes permettent d'établir une connexion directe entre les IOBs et les CLBs adjacents ou entre CLBs adjacents. Ces lignes réduisent la consommation de puissance car elles sont directes.

3. **Longues lignes de connexion** : Ce type de connexion est destiné à relier des CLBs distants les uns aux autres. Les FPGA-SRAMs optent pour les longues lignes pour éviter tout retard de signaux causé par la configuration de plusieurs petits segments par des point CIPs afin d'établir une connexion entre des CLBs distants les uns aux autres. Les longues lignes utilisent peu de points CIPs réduisant ainsi les ressources de routage et le retard dû aux points CIPs [29]. Toutefois, les longues lignes limitent la flexibilité du routage et par conséquent une conception FPGA peut devenir non-routable [40].

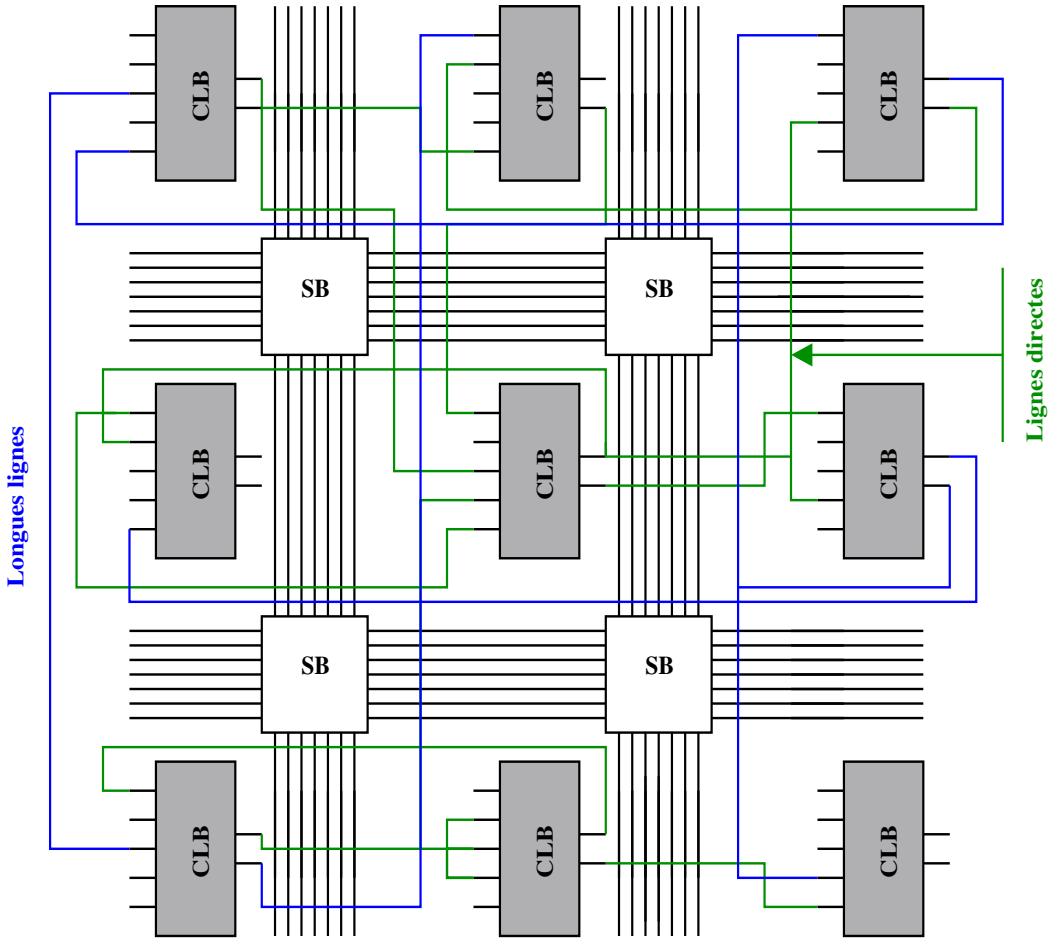


FIGURE 2.5: Schéma conceptuel montrant les trois types de connexion déployés dans les FPGA-SRAMs.

La figure 2.5 montre les trois façons pour établir une communication entre les différentes ressources logiques dans un FPGA-SRAM. Ces types de connexion sont choisis selon la disposition des ressources logiques dans le FPGA-SRAM et aussi pour augmenter les performances de la conception FPGA. En plus des CLBs, les FPGA-SRAMs disposent de différentes ressources logiques nécessaires pour les applications modernes. Dans ce qui suit, nous allons abordé certaines de ces ressources.

Remarque 2.1.
La distribution des lignes de connexion ainsi que les coordonnées des autres ressources dans les FPGA-SRAMs de Xilinx sont accessibles à l'aide de l'outil FPGA Editor du kit de développement FPGA Xilinx.

2.3.3 Les mémoires BRAMs et autres ressources logiques

Les FPGAs à base de mémoire SRAM disposent en outre des blocs CLBs des blocs mémoires BRAMs et des ressources arithmétiques. En effet, des blocs mémoires BRAMs sont intégrés dans les FPGA-SRAMs afin de réduire la surface requise par l'implémentation du circuit et d'augmenter ses performances. Les BRAMs sont configurables (mise en cascade de plusieurs BRAMs) pour être adaptées à une conception FPGA. Ces blocs mémoires

dans la famille Xilinx virtex-5 peuvent mémoriser jusqu'à 36 Kilobits de données [35]. À titre d'exemple, les BRAMs dans le FPGA Virtex-5 sont configurables en deux mémoires RAMs indépendantes de 18 Kilobits chacune, une mémoire de taille modifiable, synchrone RAM ou asynchrone RAM, etc.

Les constructeurs des FPGA-SRAMs ont intégrés dans les FPGA-SRAMs récents des blocs arithmétiques dédiés pour les opérations arithmétiques complexes. Les FPGA-SRAMs intègrent des blocs DSPs pour des applications de traitement de signal. Les FPGA-SRAMs disposent aussi des blocs arithmétiques pour exécuter des opérations de multiplication et de multiplication-accumulation MAC, des DCM (Digital Clock Manager), etc.

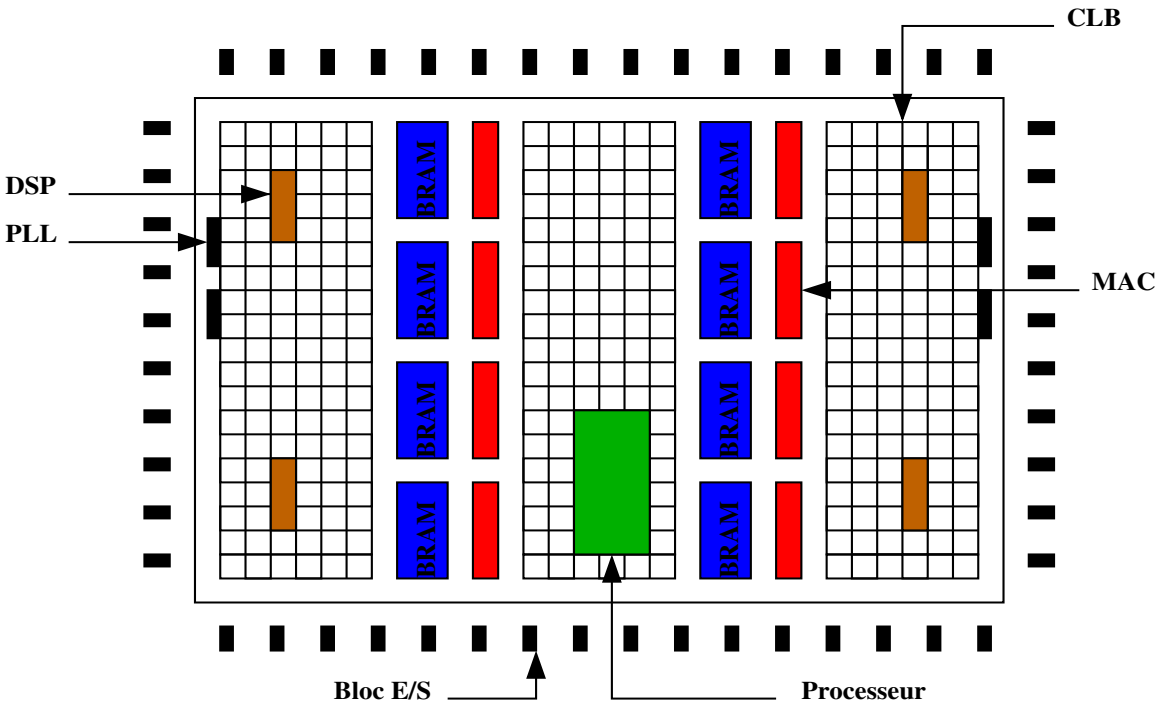


FIGURE 2.6: Architecture d'un FPGA-SRAM CSoC.

La figure 2.6 donne une idée générale sur les FPGA-SRAMs récents. En effet, les FPGA-SRAMs récents sont devenus des systèmes CSoCs grâce au progrès dans les technologies d'intégration. Certains FPGA-SRAMs de la famille Xilinx Virtex-5 disposent d'un ou deux processeurs embarqués. À titre d'exemple, le FPGA Virtex-5 XC5VFX70T intègre le processeur PowerPC 440 de IBM pouvant fonctionner à plus de 400 MHz[41]. Il existe des IPs logiciels développées en langage VHDL qui permettent l'assemblage de tous les périphériques nécessaires pour l'exploitation du PowerPC 440 comme le PLB (Processor Local Bus), MMU (Memory Management Unit), etc.

2.3.4 La configuration d'un FPGA-SRAM

La configuration d'un FPGA à base de mémoire SRAM consiste à télécharger le *Bitstream* dans le FPGA-SRAM. Les FPGA-SRAMs disposent de millions de cellules SRAMs formant les cellules de la mémoire de configuration (CMA). La figure 2.7 montre un schéma concep-

tuel d'un FPGA-SRAM dans lequel la couche de configuration reçoit le *Bitstream* qui définit par la suite les tâches de la couche opérative.

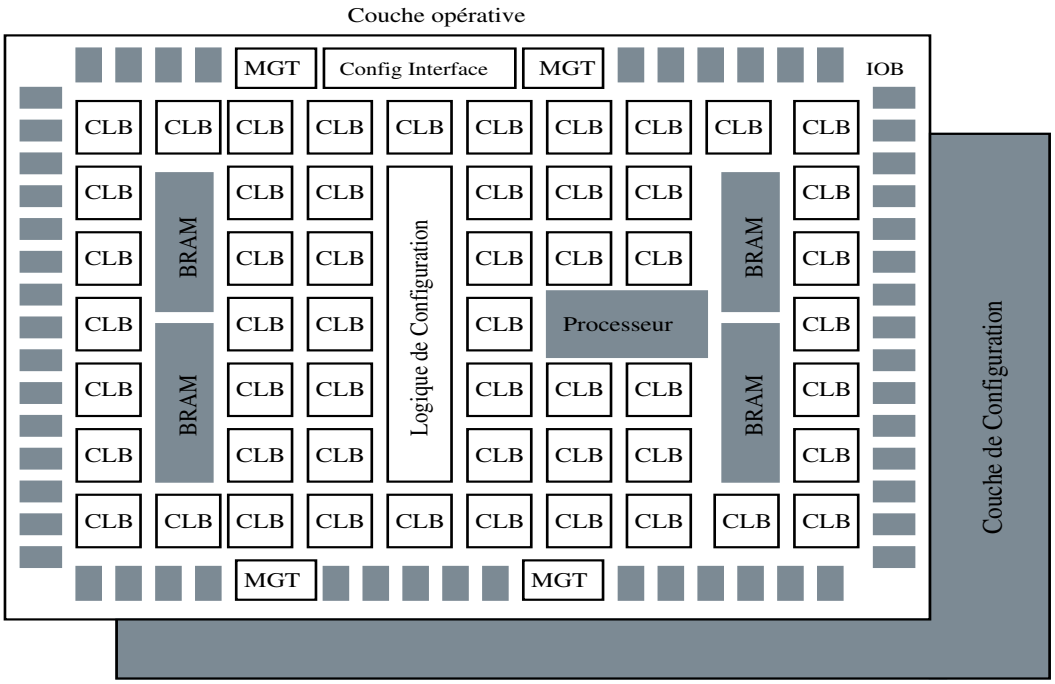


FIGURE 2.7: Schéma conceptuel de la couche opérative et couche de configuration des FPGAs à base de mémoire SRAM.

La couche de configuration CMA est constituée de trames de configuration (configuration frames). Une trame de configuration est la plus petite unité configurable dans un FPGA-SRAM. Les outils de développement des conceptions FPGA ainsi que les FPGA-SRAMs modernes permettent l'accès à chaque trame de configuration pour pouvoir effectuer des modifications. Dans la suite, nous détaillons les étapes et les types de la reconfiguration des FPGA-SRAMs.

2.4 Configuration et reconfiguration des FPGAs à base de mémoire SRAM

Les FPGA-SRAMs modernes sont essentiellement des CSoCs intégrants des processeurs embarqués, des blocs DSP, des cœurs IPs pour des applications diverses. Ce progrès technologique permet aux FPGA-SRAMs de devenir une alternative concrète aux ASICs. Les FPGAs à base de mémoire SRAM sont la technologie la plus répondue parmi les architecture reconfigurables. Dans cette partie nous étudions les différents types et méthodes de la configuration des FPGA-SRAMs, nous montrons leur utilité et nous abordons aussi les limites de la reconfiguration partielle.

2.4.1 La reconfiguration complète des FPGA-SRAMs

La reconfiguration complète d'un FPGA-SRAM consiste à utiliser le FPGA-SRAM pour implémenter une nouvelle conception FPGA. La conception FPGA est initialement décrite

par un langage de description matériel HDL (VHDL, Verilog, System C, ...) ou avec un éditeur de schémas. Celle-ci est transformée par l'intermédiaire d'un outil de conception FPGA-SRAM en *Bitstream* de configuration. La figure 2.8 - adaptée du [42] - montre les étapes standards de déroulement de la conception et la configuration d'un FPGA-SRAM. Un *Bitstream* est un fichier binaire destiné à configurer la mémoire de configuration du FPGA-SRAM (CMA) par l'intermédiaire d'une interface de configuration (JTAG, SelecMAP, Interface de configuration série, ...) [35]. Il contient toutes les informations de gestion de la configuration (positionnements des circuits ainsi que les interconnexions) et les données de la configuration (fonction de la conception FPGA). Les données de la gestion de la configuration d'un *Bitstream* rendent celui-ci propre au FPGA-SRAM ciblé, autrement dit un *Bitstream* n'est pas portable.

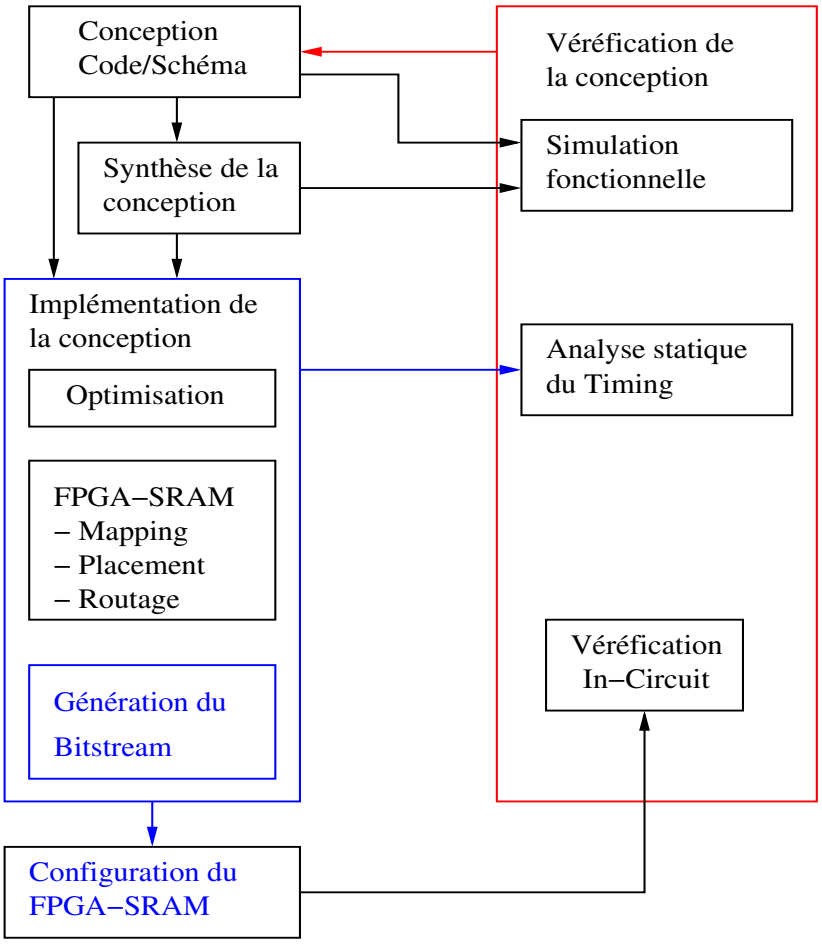


FIGURE 2.8: Diagramme standard de déroulement de la conception sous les FPGA-SRAMs de la technologie *Xilinx*.

La nature volatile de la mémoire CMA fait que les données de configuration de la conception FPGA d'avant sont effacées laissant la place aux données de la nouvelle conception FPGA. Pour un FPGA de technologie SRAM, en l'occurrence les FPGAs de technologie SRAM de la famille Xilinx, le *Bitstream* de la configuration complète gardera la même taille quelque soit l'application à implémenter dans le FPGA-SRAM.

2.4.2 La reconfiguration partielle statique

La reconfiguration partielle (PR) est par définition : la capacité de reconfigurer une partie d'un FPGA à base de mémoire SRAM tandis que le reste du système implémenté dans les autres parties du FPGA-SRAM reste inchangé sans recours à la modification de la plateforme électronique du circuit (partie matériel du circuit)[43]. Pour les FPGAs à base de mémoire SRAM de Xilinx la reconfiguration partielle consiste tout d'abord à développer et mettre en œuvre une conception FPGA qui est partiellement reconfigurable en utilisant une technique de conception modulaire appelée partitionnement [44]. Chaque module est défini par un fichier binaire partiel (partial *Bitstream*) qui représente la fonction de chaque module. En se servant de la flexibilité des FPGA-SRAMs, la PR permet la modification d'une partie déterminée d'une conception FPGA par le chargement d'un fichier de configuration partielle.

Après avoir configuré un FPGA de technologie SRAM, les fichiers binaires partiels peuvent être utilisés pour configurer les parties reconfigurables d'une conception FPGA. Cette opération se fait sans compromettre l'intégrité des fonctions exécutées dans les autres parties du FPGAs à base de mémoire SRAM qui ne sont pas concernées par la reconfiguration partielle.

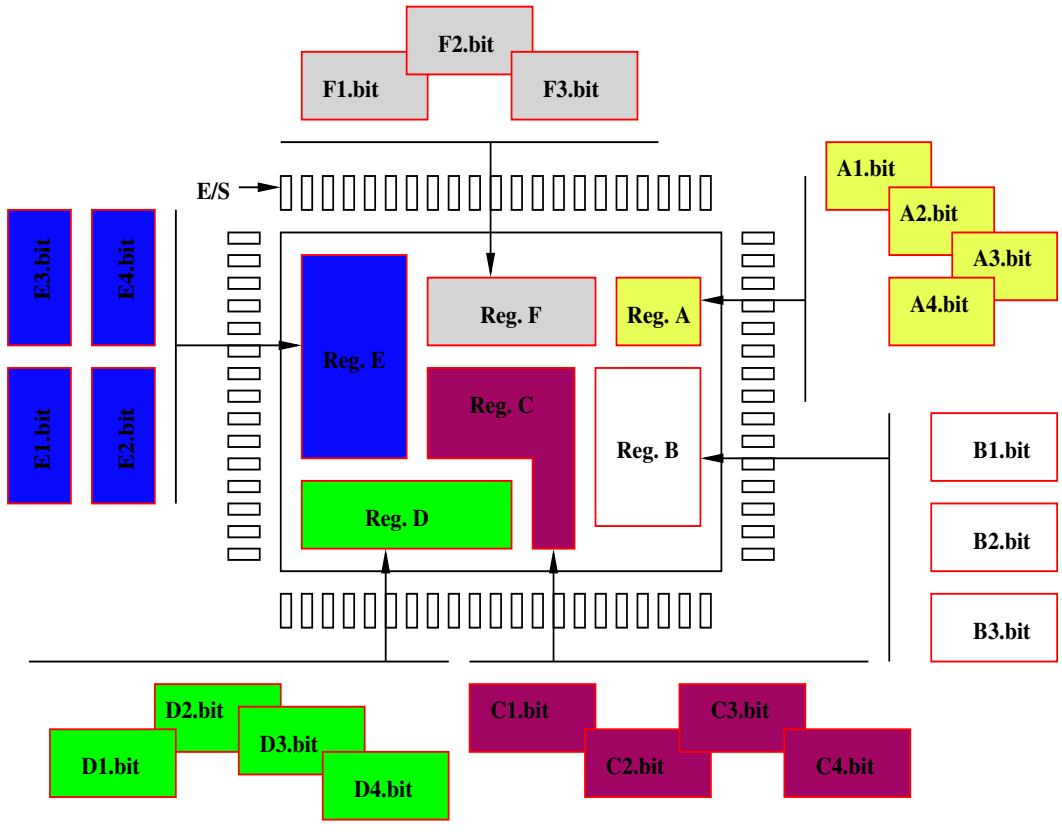


FIGURE 2.9: Schéma conceptuel de la reconfiguration partielle statique basée sur la technique du partitionnement d'un FPGA-SRAM

La figure figure 2.9 illustre le principe de la reconfiguration partielle utilisé par les FPGAs à base de mémoire SRAM de la technologie Xilinx. Ce principe est basé sur le partitionne-

ment d'un FPGA en plusieurs régions reconfigurables. Comme illustré dans la figure 2.9, la logique reconfigurable du FPGA-SRAM utilisée par la conception FPGA est partitionnée en régions A, B, C, D, E et F. Ces régions sont indépendantes les unes par rapport aux autres au sens configuration. Cependant, elles échangent des données entre elles pour faire fonctionner l'application globale (le design FPGA). Chaque fonction implémentée dans une région est modifiable par l'un des fichiers binaires. Par exemple, la fonction implémentée dans la région E est modifiable par l'un des fichiers binaires suivant : $E_1.bit$, $E_2.bit$, $E_3.bit$ ou $E_4.bit$. Le choix du fichier binaire (*partial Bitstream*) dépend de plusieurs critères.

Les FPGAs à base de mémoire SRAM de Xilinx possèdent un certain nombre d'interfaces de configuration qui sont utilisés pour télécharger les données de configuration *(Bitstream)* dans le but de pouvoir changer complètement une conception FPGA ou pour la reconfigurer partiellement. Pour les FPGAs à base de mémoire SRAM de la famille Virtex, la reconfiguration complète ou partielle peut être accomplie à travers l'interface SelectMAP, JTAG ou ICAP [45].

2.4.3 La reconfiguration partielle dynamique

La reconfiguration partielle dynamique (PDR) permet une implémentation séquentielle (en cours d'exécution) des fonctionnalités matérielles dans une même région de logique configurable d'un FPGA de technologie SRAM. La figure 2.10 montre un exemple d'utilisation de la reconfiguration partielle dynamique, où une conception FPGA est partitionnée en deux partitions A et B. La partition B de la conception FPGA totale constitue la partie statique qui reste inchangée en cours d'exécution de l'application sur FPGA-SRAM.

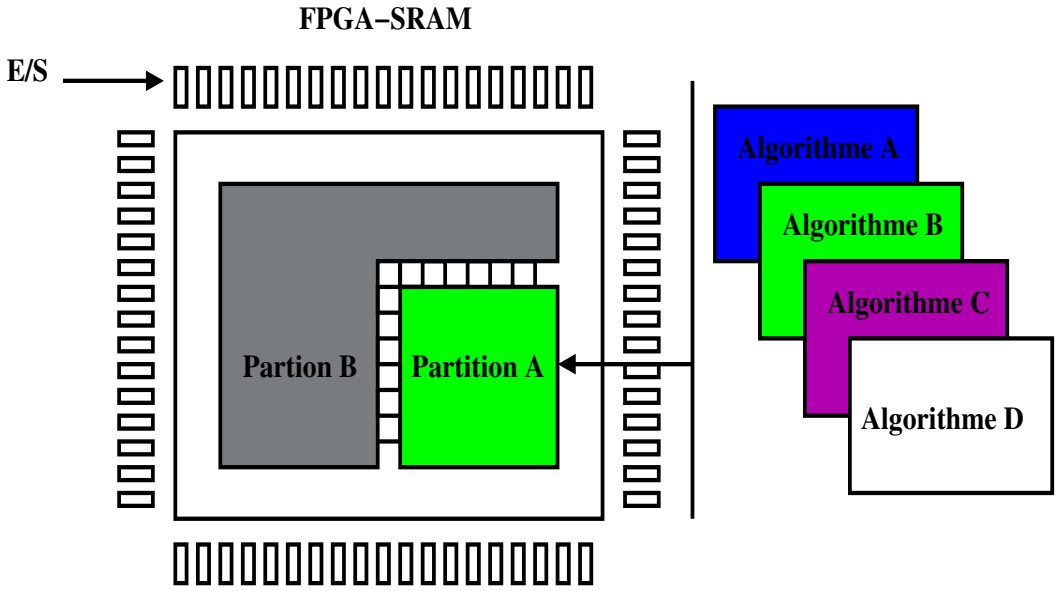


FIGURE 2.10: Schéma conceptuel de la reconfiguration partielle dynamique d'un FPGA-SRAM

En contre partie, la partition A de la conception FPGA forme la partie dynamique de cette dernière. Elle est donc destinée à être implémenter et exécuter, par l'intermédiaire d'un contrôleur de configuration plusieurs, tâches matérielles en l'occurrence l'algorithme A, B, C et l'algorithme D. L'exécution séquentielle des tâches matérielles représentées par les algorithmes A, B, C et D se fait selon les besoins de la conception FPGA.

La reconfiguration dynamique partielle ajoute une flexibilité supplémentaire aux architectures FPGAs de technologie SRAM. Elle permet une réduction considérable de la consommation de puissance et de surface (hardware overhead).

La reconfiguration partielle avec ses différents types et techniques apportent aux FPGAs à base de mémoire SRAM plusieurs avantages [45] :

1. Amélioration de la tolérance aux pannes des FPGAs à base de mémoire SRAM.
2. La reconfiguration partielle permet une flexibilité dans le choix d'algorithmes ou des protocoles utilisés dans une conception FPGA.
3. Permettant de nouvelles techniques de sécurité au sein d'un FPGA-SRAM.
4. Permettant l'accélération de calcul.
5. Permettant la réduction de la taille des ressources d'un FPGA de technologie SRAM nécessaire pour l'implémentation d'une même application, et donc la réduction de la puissance de consommation et le coût de fabrication.
6. Une reconfiguration en temps d'exécution et à distance.
7. Une optimisation des ressources du FPGA-SRAM.

Nous pouvons dire que le concept de la reconfiguration dynamique dans les FPGAs à base de mémoire SRAM permet de réduire la consommation de l'énergie, augmenter la fiabilité des systèmes à base des FPGA-SRAM ainsi que de réduire les coûts de ces derniers. Ces avantages ne sont réalisables qu'à une condition sine qua non qui est l'efficacité dans la conception et la gestion précise de ce concept.

2.4.4 Limites de la reconfiguration partielle dynamique

La flexibilité des FPGAs à base de mémoire SRAM combinée avec les techniques de la reconfiguration partielle ont permis de nouvelles méthodes de conception qui ne sont pas applicables sur des circuits spécifiques (ASICs) ou les circuits programmables (processeurs, DSP). Or, ces techniques ne sont pas pourvues d'inconvénients. En effet, la reconfiguration partielle doit être exploitée judicieusement, *i. e.*, le partitionnement de la zone reconfigurable implémentant l'application est un problème complexe qui prend en compte le partitionnement spatial et temporel [46].

Le mauvais choix des zones reconfigurables se répercute sur le coût temporel qui peut impacter les performances de la conception FPGA. En effet, un partitionnement non optimisé en zones reconfigurables se répercute sur les performances du FPGA-SRAM par l'augmentation du sur-coût de surface (ressources non exploitables). Les zones configurables sont de tailles fixes et de bords figés, l'optimisation de ces zones consiste à répondre au maximum aux besoins des tâches qui seront implémentées dans ces zones. Pour une meilleure optimisation des zones reconfigurables, les zones reconfigurables doivent implémenter des tâches de tailles à peu près identiques.

Les outils permettant de mettre en œuvre la reconfiguration partielle exigent que le concepteur partitionne manuellement le FPGA-SRAM, ce qui demande une maitrise rigoureuse de l'architecture physique du FPGA-SRAM manœuvrée pour des performances optimales. Ces facteurs ont rendu la technique de la reconfiguration partielle moins attractive par les

concepteurs FPGA-SRAM. Dans le but de surmonter le problème de partitionnement manuel du FPGA-SRAM et rendre la reconfiguration partielle accessible à une large communauté d'ingénieurs, les auteurs du [47] proposent une méthode de partitionnement automatique d'un FPGA-SRAM, supportant la reconfiguration partielle, qui peut être fusionnée avec les outils de développement FPGA-SRAM et qui surmonte les limitations existantes des autres méthodes. Les résultats de leur étude montrent que la méthode proposée n'atteint pas une utilisation à 100 % des ressources de chaque zone reconfigurable. L'optimisation du surcoût de surface et de la flexibilité de la zone reconfigurable, en fonction des applications à exécuter, reste le problème majeur de la reconfiguration partielle.

Remarque 2.2.

Xilinx, Altera et les autres constructeurs des FPGA-SRAMs ne fournissent pas des informations concernant le format du Bitstream ainsi que l'architecture de leurs FPGA-SRAMs. Ce manque d'informations engendre les obstacles suivants [48][49] :

1. *La recherche scientifique dans le domaines des FPGA-SRAMs ne donne pas des résultats rigoureuses à cause de ce manque d'information.*
2. *Ce manque de détails limite l'utilisation des FPGA-SRAMs dans certains domaines.*
3. *Ce manque d'informations empêche aussi de nouvelles utilisations de l'informatique reconfigurable de voir le jour.*

2.4.5 La reconfiguration partielle ponctuelle

La reconfiguration partielle ponctuelle (Difference-based partial reconfiguration) permet au concepteur FPGA-SRAM d'apporter des modifications ponctuelles sur une conception FPGA [50]. Ce type de reconfiguration partielle n'agit donc que sur la différence entre les deux versions de la conception FPGA, et donc ne sert pas à reconfigurer les parties du FPGA à base de mémoire SRAM implémentant les mêmes fonctionnalités. La taille du *Bitstream* utilisée par cette méthode est réduite par rapport à celle du *Bitstream* de la conception complète, ce qui permet une réduction du temps d'indisponibilité de la ressource reconfigurée. Les éléments éligibles par la reconfiguration partielle ponctuelle sont l'équation d'une LUT, le contenu d'un bloc RAM, les multiplexeurs dans un slice et les blocs IOBs. En contre partie, l'application de cette technique pour changer des valeurs dans les ressources de routage n'est pas recommandée car il risque de créer une discorde interne [50].

Conclusion :

Les FPGAs à base de mémoire SRAM sont devenus omniprésent. Cette vérité est basée sur le fait que les FPGA-SRAMs agrègent les caractéristiques des microprocesseurs et celles des circuits ASICs. Si les microprocesseurs exécutent les tâches en mode séquentiel et de façon concurrente et cherchent à atteindre le parallélisme dans leur fonctionnement. Les FPGA-SRAMs sont technologiquement dotés d'un vrai parallélisme. En effet, les fonctions d'une conception implémentée dans un FPGA-SRAM ne se retrouvent pas en situation de concurrence entre elles pour être exécutées. Chaque fonction est implémentée dans une région du FPGA-SRAM ce qui donne une autonomie et une indépendance dans l'exécution des tâches. Grâce à l'évolution des FPGA-SRAMs et grâce à la capacité de ces architectures de se reconfigurer dynamiquement, l'espace de conception délimité par les FPGAs à base de mémoire SRAM est en expansion.

Ce chapitre a montré que les architecture reconfigurables ont réduit les Gaps entre les FPGA-SRAMs et les circuits ASICs d'un côté et les microprocesseurs d'un autre côté, ce qui leurs permet de devenir une alternative importante pour la conception de nouvelles applications et un omniprésence dans tous les domaines.

La grande densité d'intégration et l'omniprésence des FPGA-SRAMs font que ces derniers ont besoin d'un degré de fiabilité élevé. Les FPGA-SRAMs dans leurs environnement de travail sont exposés à des **pannes**, que ce soit des pannes permanentes ou des pannes transitoires. Les caractéristiques particulières des FPGA-SRAMs, telle que la notion de la reconfiguration partielle dynamique ou leur architecture modulaire, permettent de faire face à ce problème. Un état de l'art sur la tolérance aux pannes des circuits FPGA-SRAMs constituera le sujet du chapitre suivant.

Techniques de test et de tolérance aux pannes des FPGAs de technologie SRAM

3

« The most certain and effectual check upon errors which arise in the process of computation, is to cause the same computations to be made by separate and independent computers ; and this check is rendered still more decisive if they make their computations by different methods »

Dionysius Lardner dans "Babbage's calculating engine 1834" [51][52]

Sommaire

Introduction au chapitre :

Avec la croissance continue de la densité d'intégration des circuits VLSIs et l'utilisation des processus de plus en plus complexes, il est de plus en plus difficile de concevoir et de produire des circuits intégrés sans défauts. En outre, un circuit intégré est sensé être capable de préserver un degré de fiabilité permettant à ce dernier de fonctionner correctement (sans pannes) dans des environnements à conditions variables.

Les FPGAs à base de mémoire SRAM ne font pas l'exception de cette vérité. Ils sont exposés à des pannes permanentes et à des pannes transitoires vue leur nature configurable. La tolérance aux pannes post-fabrication est devenue donc une étape importante pour assurer le bon fonctionnement des systèmes implémentés sous FPGA-SRAMs.

L'objectif de ce deuxième chapitre est de donner les concepts agrégeant les importantes techniques et les approches de la tolérance aux pannes des circuits FPGA-SRAMs. Nous présentons les techniques de test et de tolérance aux pannes pour les FPGA-SRAMs ainsi qu'une comparaison entre ces méthodes. Nous discuterons aussi l'importance et le rôle de la reconfiguration partielle dans l'amélioration des techniques de la tolérance aux pannes des circuits FPGA-SRAMs. Finalement, ce chapitre est introduit dans une optique d'une analyse des techniques de test et de tolérance aux pannes du point de vu utilisateur et ne pas particulièrement du côté fabricant.

3.1 Motivation

Les processus d'intégration des FPGA-SRAMs sont devenus de plus en plus complexes dans l'objectif d'augmenter les performances et de réduire la consommation en puissance de ces derniers ainsi que de produire des FPGA-SRAMs adaptables aux nouveaux standards. Dans ce but, la société *Xilinix* a utilisé une échelle d'intégration de 28 *nm* pour la famille *Virtex-7* [53] et tend à utiliser des échelles d'intégration inférieurs dans un future proche. Cette échelle nanométrique d'intégration des circuits FPGA-SRAMs rend ceux-ci susceptibles à des défauts pré et post fabrication. De ce fait, les pannes sont devenues de plus en plus prononcées dans les applications émergentes à base des FPGA-SRAMs, des pannes permanentes aux pannes transitoires.

La tolérance aux pannes apparaît donc comme un paramètre essentiel pour assurer un degré élevé de fiabilité d'un système à base de FPGA-SRAMs. En effet, depuis 1993 des laboratoires ont commencé à s'intéresser à ce nouveau domaine [12] en introduisant des algorithmes en parallèle des systèmes implémentés sur FPGA-SRAM afin de bénéficier d'une sûreté de fonctionnement permettant le bon fonctionnement du système implémenté. La recherche dans le domaine de la tolérance aux pannes des FPGAs de technologie SRAM a fixé deux axes principaux :

1. La capacité de la technique de tolérance aux pannes de détecter les pannes lors du fonctionnement du FPGA-SRAM ainsi que de localiser la partie défectueuse du FPGA-SRAM responsable des pannes engendrées.

2. Le FPGA-SRAM sera capable après réparation des pannes de retourner à son état normal et de délivrer les fonctions attendues du système implémenté sur FPGA-SRAM.

Une technique de tolérance aux pannes complète devra respecter ces deux conditions avec un minimum de coût matériel (area overhead) et une gestion efficace de la réparation des pannes.

Ce troisième chapitre se veut être un état de l'art sur les techniques de test et de la tolérance aux pannes des FPGA-SRAMs. Nous le commençons par étudier les différentes importantes méthodes et approches de la tolérance aux pannes des FPGA-SRAMs. Ensuite, nous étudions également les plus importantes techniques de test des FPGAs.

3.2 Tolérance aux pannes et modèles de pannes

Réussir à reconstruire une structure logique (partie matérielle) fiable à partir d'un FPGA-SRAM défectueux constitue le point commun entre toutes les méthodes et techniques développées dans le cadre de la tolérance aux pannes des FPGAs de technologie SRAM.

3.2.1 Importance de la tolérance aux pannes des FPGA-SRAMs

Les FPGA-SRAMs occupent de plus en plus une place importante dans les systèmes ordinaires et systèmes critiques à haute performance. Les FPGA-SRAMs sont des composants d'une plus grande complexité et font partie des composants pris sur étagère (COTS). Cette complexité les rends vulnérables vis-à-vis de la fiabilité. Pour améliorer le niveau de fiabilité des FPGA-SRAMs, des techniques de plus en plus formelles de tolérance aux pannes sont utilisées dans le but de surveiller si les conditions de la fiabilité sont assurées. Nous remarquons donc, que la tolérance aux pannes est naît parallèlement avec le choix de l'intégration des FPGA-SRAMs. Il est important donc de donner une définition à la tolérance aux pannes des FPGA-SRAMs afin de fixer les objectifs de cette thèse. En se basant sur la définition donnée à la tolérance aux pannes en [12], nous donnons la définition suivante :

Définition 3.1. *Nous définissons la tolérance aux pannes des FPGA-SRAMs comme étant l'ensemble des techniques et approches qui permettent de prolonger la durée de vie d'un ou de plusieurs systèmes implémentés sous un FPGA-SRAM. Une approche de tolérance aux pannes devra être capable de gérer d'une manière autonome et efficace l'apparition de toutes sortes de pannes que ce soit des pannes permanentes ou transitoires. Un système de tolérance aux pannes complet comportera les phases suivantes :*

1. *La phase de test : Tester le FPGA-SRAM à fin de détecter des éventuelles pannes.*
2. *la phase du diagnostic : cette phase consiste à localiser les pannes et donner leurs natures.*
3. *La phase de tolérance aux pannes : utilisation d'une technique de tolérance aux pannes pour faire fonctionner le système correctement.*

Un mécanisme de tolérance aux pannes des FPGA-SRAMs devra prendre en compte un rapport de qualité entre la fiabilité du système implémenté sous le FPGA-SRAM et les ressources logiques utilisées par ce mécanisme, le retard en temps et la consommation en puissance du mécanisme.

Dans la définition 3.1 nous avons défini la tolérance aux pannes des FPGA-SRAMs comme étant un processus complet qui rassemble le test, le diagnostic ainsi que la phase de répa-

ration. Il se peut que cette définition ne soit pas la même définition donnée à la tolérance aux pannes des FPGA-SRAMs par d'autres chercheurs car ces derniers considèrent que les phases de test, du diagnostic et la tolérances aux pannes des FPGA-SRAMs comme des phases distincts [54][55].

Le tableau 3.2 [2] montre une statistique d'apparition des défauts dans les circuits intégrés. La fréquence d'apparition des défauts renforce l'idée de développer des systèmes tolérants aux pannes.

Type du défaut	**Fréquence d'apparition en (%)**
Court-Circuits (Shorts)	51
Ouvertures (Opens)	1
Composants manquants	6
Mauvais composants	13
Composants inversés	6
Mauvaises spécification analogiques	5
Défauts dans les performances	5
Bents Leads	8

TABLE 3.2: Statistique montrant la fréquence d'apparition de certains défauts dans un circuit intégré typique.

Une technique de tolérance aux pannes des FPGA-SRAMs, rassemblant les trois étapes de la définition 3.1 restera le but de notre thématique de recherche. En effet, dans la littérature nous trouvons des travaux qui se focalisent sur le test et la détection des pannes dans les ressources logiques des FPGA-SRAMs [56] tandis que d'autres se focalisent sur les ressources de routage.

3.2.2 Importance de la tolérance aux pannes vis-à-vis des FPGA-SRAMs

Comme nous l'avons mentionné dans la partie 3.2.1 de ce chapitre, les FPGA-SRAMs sont des composants COTS et sont aussi de complexité de plus en plus élevée. En plus, les FPGA-SRAMs sont devenus omniprésents dans la majorité des systèmes embarqués. Ces caractéristiques font que les FPGA-SRAMs sont devenus difficiles à gérer aux niveaux matériel et logiciel et donc ils ont besoin de maintenir un niveau de fiabilité assurant leur bon fonctionnement notamment pour les applications critiques à haute performance.

Les FPGA-SRAMs sont en évolution croissante, cette évolution est accompagnée d'une évolution des défauts qui peuvent compromettre le fonctionnement normal de ces ICs. Il est donc primordiale d'introduire des techniques de tolérance aux pannes pour les applications s'appuyant sur des FPGA-SRAMs. Des techniques de tolérance aux pannes qui adoptent des méthodologies d'analyse de vulnérabilités menant à identifier les pannes dans une architecture FPGA-SRAM.

Les techniques de tolérance aux pannes des FPGA-SRAMs peuvent être regroupées en deux catégories [12] :

1. Techniques de tolérances aux pannes qui traitent les pannes au niveau matériel d'une architecture FPGA-SRAM. Généralement ces techniques tentent de construire une

architecture FPGA-SRAM sans défauts à partir d'une architecture contenant des ressources défectueuses.

2. Techniques de tolérance aux pannes qui tolèrent les pannes au niveau configuration d'un FPGA-SRAM sans tenir compte de l'architecture matérielle de celui-ci.

Sélectionner la technique adéquate pour la tolérance aux pannes selon [12] repose sur plusieurs critères. Parmi ces critères, on trouve :

1. La vitesse de traitement des pannes dans l'architecture FPGA-SRAM.
2. La taille occupée par la technique de tolérance aux pannes dans un FPGA-SRAM.
3. Le coût - de la puissance consommée - généré par la technique de tolérance aux pannes.
4. Le type d'utilisateur du FPGA-SRAM : le constructeur opte pour des techniques de tolérance aux pannes au niveau matériel du FPGA-SRAM [57], tandis que l'utilisateur final utilise des technique de tolérance aux pannes au niveau configuration du FPGA-SRAM.

Dans cette partie de ce chapitre, nous avons montré l'importance de la tolérance aux pannes vis-à-vis des FPGA-SRAMs notamment les technologies récentes. Dans la suite, nous étudions les concepts de ce domaine et nous définissons les importantes métriques sur lesquelles la tolérance aux pannes s'appuie.

3.2.3 Concepts de la tolérance aux pannes

Dans cette partie de ce chapitre, nous allons définir et étudier les concepts fondamentaux de la tolérance aux pannes des FPGA-SRAMs en se basant essentiellement sur la définition de la fiabilité publiée dans [52]. Dans cette optique, un FPGA-SRAM est vu comme plusieurs systèmes en interaction mutuelle.

La communication d'un système FPGA-SRAM avec son environnement extérieur signifie la mise en œuvre de *fonctions* qui fournissent à ce dernier des *services*. D'après la terminologie utilisée dans [52], un système FPGA fournit un service correct quant il implémente la fonction attendue de ce dernier. Une déviation entre le service délivré et le service correcte résulte d'une défaillance du système FPGA-SRAM. Une défaillance de service est un événement qui se produit lorsque le service délivré dévie du service correct [52]. Puisqu'un service est une séquence d'états internes du système, une panne de service signifie qu'au moins un état interne du système s'écarte de l'état de service correct [52]. Cela nous amène à définir les deux importants termes suivants en adoptant les définitions contenues dans [11] et [52] :

Définition 3.2. ***Défaut*** *: C'est un problème qui se produit lors de la fabrication d'un circuit FPGA-SRAM ou qui résulte du vieillissement du circuit. Il peut être caractérisé par une modification au niveau logique selon sa localisation dans le circuits FPGA-SRAM.*

Définition 3.3. ***Panne*** *: C'est une modification de la structure de la fonction implémentée dans le FPGA-SRAM, elle représente l'effet d'un défaut sur le comportement nominal d'un FPGA-SRAM. Une panne est active quand elle produit une erreur, sinon elle est latente.*

Comme dans [11, 58, 54][59]-[61] nous distinguons entre la tolérance aux défauts et la tolérance aux pannes des FPGAs de technologie SRAM. Le premier concept est utilisé

lors de la phase de fabrication des circuits FPGA-SRAMs [57], tandis que le deuxième concept est utilisé lors de la mise en fonctionnement par l'utilisateur [11]. Dans la suite, nous abordons les techniques de prévention des pannes dans les FPGA-SRAMs, *i.e.*, les techniques de test des FPGA-SRAMs.

3.3 Techniques de test des FPGA-SRAMs

Dans cette partie de ce chapitre, nous allons présenter les principaux concepts du domaine du test des FPGA-SRAMs. Cette partie détaille aussi les différentes méthodes de test des FPGA-SRAMs durant l'utilisation de ces architectures.

3.3.1 Processus et approches de tests des FPGA-SRAMs

On peut regrouper les tests des circuits FPGAs à base de mémoire SRAM en deux approches [62] :

1. Approche fonctionnelle.
2. Approche structurelle.

Un test fonctionnel, comme son nom l'indique, teste le FPGA-SRAM par rapport à la fonction destinée à être implémentée dans le FPGA-SRAM. Donc, à l'inverse du test exhaustif qui teste toutes les combinaisons possibles d'une LUT d'un FPGA-SRAM par exemple. Comme dans la plupart des cas, on ne connait pas à l'avance la conception qui va être implémentée dans le FPGA-SRAM. Le test fonctionnel ne s'intéresse qu'aux combinaisons d'une LUT qui vont être déployées dans la conception qui va être exécutée sur un FPGA-SRAM.

Concrètement, le test fonctionnel pour un FPGA-SRAM consiste à analyser le cahier des charges ou la fiche technique de la fonction implémentée dans le FPGA-SRAM afin d'élaborer un banc de vecteurs de test selon le cahier des charges spécifié. Cette approche ne teste, donc, que la partie du FPGA-SRAM qui implémente la fonction spécifiée. La figure 3.1 montre le protocole du test fonctionnel pour une conception implémentée dans un FPGA-SRAM.

Dans le cas d'un test fonctionnel, le processus vérifie la combinaison utilisée par la conception implémentée et non pas toutes les combinaisons possibles d'une LUT. Cette approche n'est pas pourvue d'inconvénients :

1. L'approche fonctionnelle n'est pas un test *"portable"* ,*i.e.*, une reconfiguration du FPGA-SRAM par une conception différente de l'ancienne configuration implique un changement dans les vecteurs du test et par conséquent dans le processus du test.
2. L'approche fonctionnelle ne permet pas un diagnostic des pannes.

L'approche structurelle vise à tester la structure interne d'un FPGA-SRAM, il est complètement indépendant de la conception implémentée dans le FPGA-SRAM. Cette approche teste tous les modes de fonctionnement des éléments constituant l'architecture du FPGA-SRAM. Un test structurel vise à partitionner l'architecture FPGA-SRAM en blocs indépendants les uns des autres. Ce partitionnement permet, d'un côté, d'atteindre un niveau élevé de diagnostic et d'un autre côté, il permet l'accès à chaque bloc en vue de le tester. Le

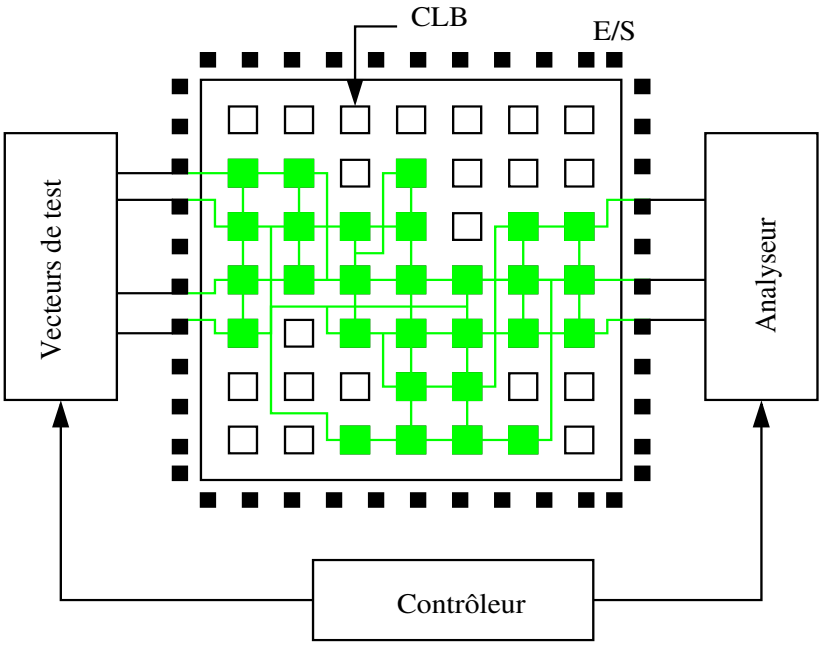


FIGURE 3.1: Schéma basique illustrant le protocole de test fonctionnel pour une conception implémenté dans un FPGA-SRAM.

test structurel vise à détecter et localiser les pannes dans les circuits FPGA-SRAMs et donc déterminer les défauts engendrant les pannes détectées. Par conséquent, l'élaboration d'un test structurel pour un FPGA-SRAM requiert une étude de l'architecture de ce dernier comme le nombre d'entrées d'une LUT, l'architecture du réseau de routage, etc.

Cette étude permettra une analyse rigoureuse des effets des défauts sur la structure de chaque élément du FPGA-SRAM, ce qui permettra de définir des vecteurs du test pour la détection de chaque panne engendrée par ces défauts. La figure 3.2 - schéma adaptée de [63] - montre les étapes principales d'une approche structurelle.

Les deux approches peuvent utiliser soit des techniques de test en ligne ou des techniques de test hors ligne.

3.3.2 Méthodes de test des FPGA-SRAMs

Les FPGA-SRAMs comme les CPLDs utilisent des millions de cellules mémoire SRAMs formant ainsi la mémoire de configuration (CMA) pour implémenter les fonctions d'une conception FPGA. En se basant sur la possibilité d'accès à la mémoire CMA, les auteurs dans [55] ont développé un circuit de test et de validation des cellules SRAMs d'un FPGA-SRAM ou d'un CPLD. L'idée du brevet [55] est de développer une technique BIST [1] en utilisant les éléments matériel du FPGA-SRAM pour construire le circuit BIST.

Les données de configuration sont transmises à la mémoire de configuration en une trame [2] à la fois via l'interface de configuration. La technique est basée sur un circuit qui utilise le

1. Built-In Self-Test est un circuit électronique implémentant une technique de test permettant à un FPGA-SRAM de s'autotester.

2. Une trame de configuration est une mémoire de largeur un bit étendue du haut de la mémoire CMA jusqu'au bas. Pour le FPGA-SRAM Virtex-5 de Xilinx une trame de configuration a une taille de 1312 bits.

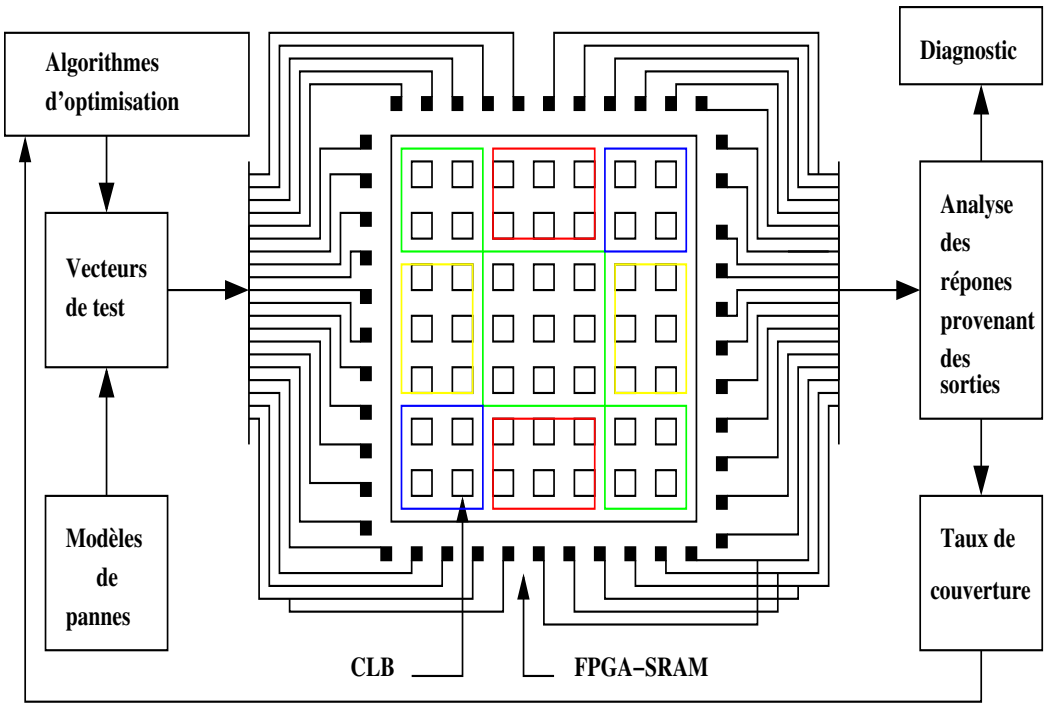


FIGURE 3.2: Schéma basique illustrant les principales étapes du test structurel d'un FPGA-SRAM.

circuit ICAP pour écrire des modèles récurrents de 1 et de 0 dans les trames de configuration et de lire ensuite dans ces trames. Le principe de la méthode consiste à écrire des modèles récurrents de 0 et 1 dans une trame de configuration donnée puis lire dans cette trame et comparer ce qui été lu avec ce qui été écrit et noter la différence. L'adresse de la trame de configuration contenant des cellules SRAM défectueuses sera stockée dans un registre de localisation.

1. La technique BIST proposée détecte les pannes de type "stack-at 0/1" dans les cellules SRAM de la mémoire de configuration d'un FPGA-SRAM.
2. La technique BIST teste le FPGA-SRAM, cellule SRAM par cellule SRAM, en considérant cette dernière comme l'unité à grain fin.
3. La technique BIST proposée vise les pannes permanentes ainsi que les pannes transitoires.
4. La technique BIST proposée permet aussi l'injection des pannes pour analyser la vulnérabilité de la mémoire CMA aux SEUs.
5. La technique BIST proposée utilise la reconfiguration partielle pour tester la mémoire CMA.
6. Techniquement, la méthode proposée relève un important travail pour implémenter le circuit BIST.
7. Les BISTs sont connues par le fait quelles effectuent le test à la fréquence de fonctionnement du circuit.

Malgré l'innovation de la technique proposée dans [55], la technique BIST est limitée à tester que les pannes de type collage à 0/1 (stack-at 0/1). Or, comme les dimensions des transistors sont devenues plus petites et les tensions d'alimentation plus basses en outre

d'une forte densité d'intégration, la mémoire CMA est devenue vulnérable à plusieurs pannes [64]- [71].

3.3.3 Test et diagnostic en ligne des FPGAs de technologie SRAM : technique BIST

Nous nous intéressons dans cette partie à la technique BIST. En effet, les techniques BISTs constituent un moyen efficace pour assurer un niveau élevé de fiabilité pour un FPGA-SRAM. Aussi, les techniques BISTs forment un processus automatisé pour la tolérance aux pannes des FPGA-SRAMs. En effet, les techniques BISTs remplacent l'intervention directe de l'ingénieur pour réparer les pannes dans un FPGA-SRAM. Ceci est important quand il s'agit des missions où l'intervention directe de l'ingénieur est difficile. Dans la suite, nous étudions quelques travaux dans ce domaine.

Dans [72], les chercheurs proposent une technique de test BIST en ligne et hors ligne pour le test et le diagnostic des pannes permanentes dans un FPGA-SRAM de façon périodique. La méthode propose trois circuits BISTs améliorés pour un diagnostic précis. La contribution principale de [72] est une continuité des travaux publiés dans [73] sont deux circuits BISTs pour le diagnostic appelés 1- et 2-diagnostiquable[3]. La technique BIST proposée est implémentée en premier lieu dans les deux colonnes du FPGA-SRAM et qui va par la suite survoler tout le réseau FPGA-SRAM. La détection des pannes par la méthode BIST proposée dans [72] consiste à comparer la sortie du CLB sous test avec la sortie d'un autre CLB de même configuration. La non concordance dans les deux réponses indique un CLB défectueux dans le groupe de CLBs sous test. Dans le cas de détection et localisation les auteurs proposent les techniques développées dans [74][75] pour la tolérance aux pannes des CLBs défectueux. Le circuit BIST développé dans [76] - prouvé dans [72] qu'il est 0-diagnostiquable - est le noyau sur lequel les auteurs ont développés les deux circuits BISTs 1- et 2-diagnostiquable.

1. Les FPGA-SRAMs ciblés sont ceux qui supportent la reconfiguration partielle.
2. Cette technique alloue deux colonnes de CLBs comme ressources de rechanges pour le processus ROTE (ROving TEster) et une troisième pour la reconfiguration des pannes.
3. La technique BIST proposée est adaptable aux FPGA-SRAMs récents selon le critère rapport entrées/sorties [72].

1. La technique BIST proposée ne teste pas les ressources de routage.
2. Un contrôleur extérieur est nécessaire pour contrôler les protocoles du processus BIST.
3. La méthode BIST proposée exige la connaissance à l'avance du circuit qui configurera chaque CLB afin de ne pas effectuer un test exhaustif.
4. Pour des raisons de simplification, les auteurs supposent qu'un même circuit configurera tous les CLBs du FPGA-SRAM et affirment que l'extension à plusieurs circuits est simple.

3. Un circuit BIST k-diagnostiquable est un circuit si en présence de $m \leq k$ CLBs défectueux, le circuit détectera correctement les m CLBs défectueux parmi les $n \geq k$ CLBs qui teste [72].

5. La technique BIST proposée ne détecte que les pannes fonctionnelles, lorsque, le circuit qui configure un CLB, ce dernier délivre à sa sortie une réponse incorrecte.

Dans ce travail, les chercheurs ont conçu deux circuits BIST fonctionnels, *i.e.*, un circuit BIST qui se base sur la vérification complète de toutes les fonctions implémentées dans chaque CLBs d'un FPGA-SRAM. Les résultats de simulation appliquée sur un modèle FPGA-SRAM de 32×32 montrent un taux de couverture élevé par rapport aux techniques [77][78] en fonction de la densité de pannes. Les résultats de simulation montrent aussi un temps de latence (temps mis pour le diagnostique d'une panne) réduit en fonction de la densité de pannes. Cependant, la technique BIST proposée reste incapable de détecter des pannes autres que les pannes de collages à 1/0 (stack-at 1/0) sachant que la densité d'intégration des FPGA-SRAMs récents cause une grande probabilité d'apparition de plusieurs autres types de pannes comme nous le verrons dans le chapitre 6.

3.3.4 Test de la structure de routage des FPGA-SRAMs

Plus de 80% des transistors dans un FPGA-SRAM sont dédiés au réseau de routage [33]. De ce fait, il est clair que le réseau de routage est plus exposé aux pannes que la partie logique dans un FPGA de technologie SRAM. Plusieurs travaux ont été réalisés dans le but de tester les interconnexions dans ces ICs.

Dans [79], les auteurs proposent une procédure de test au niveau constructeur de la partie routage du FPGA-SRAM. L'approche du test différencie entre deux types d'entrées : les entrées de fonctionnement pour l'application des vecteurs d'entrées $(IV_1 \dots IV_n)$ pendant le fonctionnement normal de la conception FPGA et les entrées de configuration qui servent à transmettre le *Bitstream* au FPGA-SRAM sous forme de vecteurs de configuartion $(CV_1 \dots CV_n)$. La procédure de test consiste à configurer successivement le FPGA en utilisant les entrées de configuration et d'appliquer par la suite des séquences de test (TS) en utilisant les entées de fonctionnement. Dans [79], les chercheurs distinguent aussi entre les pannes redondantes qui sont indétectables par une configuration et qui le sont par une autre configuration et les pannes non-redondantes. L'évaluation de la procédure de test se fait suivant une métrique de qualité appelée *taux de couverture du test de configuration* (C^{TC}) définie par le rapport du nombre des pannes non-redondantes (n_{nr}) divisé par le nombre total des pannes (n_T) suivant la formule 3.1.

$$C^{TC} = \frac{n_{nr}}{n_T} \tag{3.1}$$

La structure utilisée pour l'élaboration de la procédure de test dans [79] consiste en un réseau de $m \times m$ de blocs de connexion. Chaque bloc de connexion est relié à son voisin par k lignes. Les chercheurs utilisent tous les modèles traditionnels de pannes à l'extérieur des blocs de connexion comme les pannes de collage (stuck-at-0/1), ouverture dans une ligne, un pont entre deux lignes etc, et à l'intérieur des blocs de connexion où ils distinguent entre les lignes connectées et non-connectées comme une connexion permanente (PC) ou une déconnexion permanente (PD).

La figure 3.3 - schéma adaptée de [79] - montre la structure de routage adaptée dans l'élaboration des tests pour le réseau de routage ainsi les trois configurations pour détecter toutes les pannes, *i.e.*, rendre toutes les pannes non-redondantes.

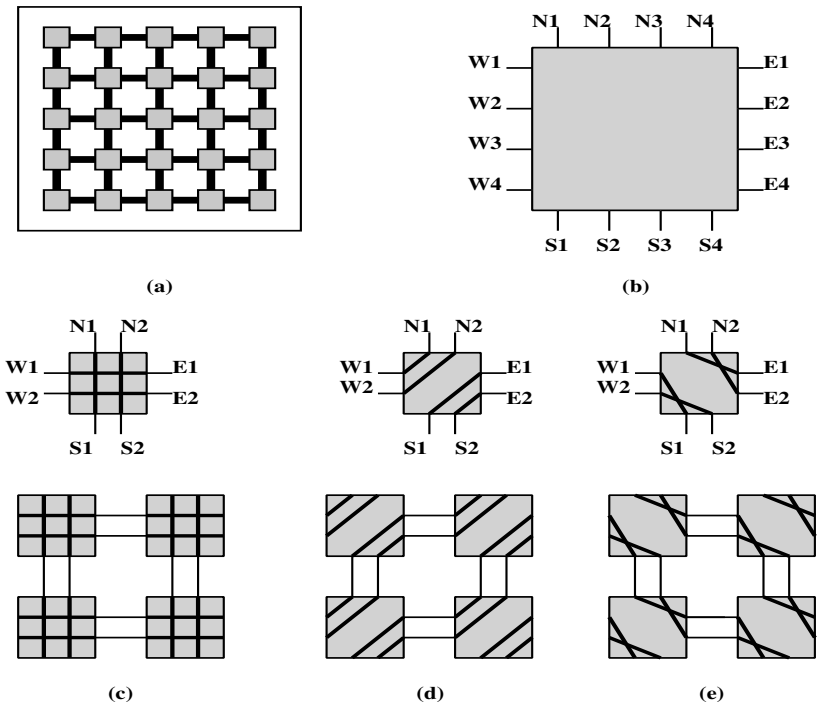


FIGURE 3.3: Schéma montrant la structure de l'architecture du routage d'un FPGA-SRAM (a) et (b), ainsi les trois configurations permettant de rendre toutes les pannes non-redondantes : Configuration orthogonale (c), Configuration diagonales 1 (d) et Configuration diagonale 2 (e).

Les trois configurations ainsi que leurs séquences de test correspondantes ont été appliquées sur les FPGA-SRAMs de Xilinx série 3000 et 4000. La procédure de test a été modifiée en vue de la structure de ces architectures. Toutefois, il n'y a pas de résultats expérimentaux montrant l'efficacité de cette procédure de test de l'architecture du routage d'un FPGA-SRAM.

3.3.5 Taux de couverture des tests des FPGA-SRAMs

Un modèle de panne représente l'impact du défaut provocant cette panne sur le FPGA-SRAM. la liste des modèles de pannes ainsi que d'autres paramètres comme la vue topologique du FPGA-SRAM (layout) constituent la phase préliminaire d'un protocole de test d'un FPGA-SRAM. Ensuite, ces pannes sont injectées d'une manière séquentielles afin d'être simulées avec tous les vecteurs de test. L'analyse des résultats de ce test se base sur le calcul du taux de couverture, *i. e.*, le rapport des pannes détectées par le test avec le nombre total des pannes injectées.

Comme dans [63], nous définissons le taux de couverture T_C par la formule suivante :

$$T_C = \frac{P_d}{P_{inj}} \tag{3.2}$$

Où P_d représente le nombre de pannes détectées et P_{inj} définit le nombre de pannes injectées pendant le test.
Le taux de couverture d'un test est un paramètre essentiel dans l'évaluation de la qualité d'un test. En effet, plus le T_C est proche de 1 plus le test est fiable. Au taux de couverture,

il faut ajouter la capacité de diagnostic d'une panne, *i.e.*, la détecter et la localiser ainsi que donner des informations sur le type de la panne. Ces informations sont issues d'une analyse rigoureuse des défauts qui ont causés la panne.

3.4 Techniques de tolérances aux pannes des FPGA-SRAMs

Plusieurs travaux bibliographiques ont été réalisés dans le champs de la tolérance aux pannes des architectures FPGA-SRAMs [12][11]. Dans cette partie, nous allons donner un état de l'art de ces techniques afin de fournir une continuation de ce qui a été déjà réalisé. Les techniques FTs seront évaluées selon les critères suivants [12] :

1. Le matériel nécessaire pour l'implémentation de la technique FT.
2. L'impact de la technique FT sur le système.
3. La complexité de l'implémentation de la technique FT (côté reconfiguration).
4. Les modèles de pannes tolérables par la technique ou le taux de couverture.

3.4.1 Tolérance aux pannes par allocation de ressources

Une technique de FT basée sur l'allocation des ressources logiques dans un FPGA-SRAM a été introduite dans [80][11]. La technique propose deux distributions de ressources logiques appelées **King-Shifting** et **Horse-Allocation**. La méthode proposée consiste en une modification de la mémoire de configuration (CMA) du FPGA-SRAM ainsi que l'allocation de certains CLBs comme des CLBs de rechange distribués de façon régulière dans le FPGA-SRAM. Cette modification permet au FPGA-SRAM le décalage - verticalement par une ligne ou horizontalement par une colonne - des données de configuration sans les modifier. Dans le cas de la détection et la localisation d'un CLB défectueux, la technique exécute un décalage des données de configuration en parallèle avec les possibilités proposées par les CLBs de rechange. La technique proposée restructure un FPGA à base de mémoire SRAM en un réseau 2-D d'éléments unitaires. Chaque élément unitaire est constitué d'un CLB et les ressources de routage qui l'entoure. Cet élément unitaire est assimilé à un registre à décalage. Les étapes de la technique proposée sont illustrées dans l'exemple de la figure 3.4 - schéma adapté de [12] et [58] -. Les auteurs prétendent que leur technique FT apporte les avantages suivants :

1. Les auteurs prétendent que la technique est valable pour le fabriquant des FPGA-SRAMs ainsi que l'utilisateur final.
2. La faisabilité des deux distributions (**King-Shifting**, **Horse-Allocation**) augmente avec la taille du réseau FPGA-SRAM.

Malgré les avantages apportés par cette technique, celle ci n'est pas dépourvue de certains inconvénients :

1. La technique considère un FPGA-SRAM comme un réseau 2-D de $N \times N$ de blocs identiques (CLBs), or les FPGA-SRAMs sont des architectures complexes et contenant des ressources logiques variées.
2. La technique proposée ne couvre pas la partie test et diagnostic du FPGA-SRAM. Les auteurs citent [81] comme technique pour le test et la localisation des pannes dans les LUTs.

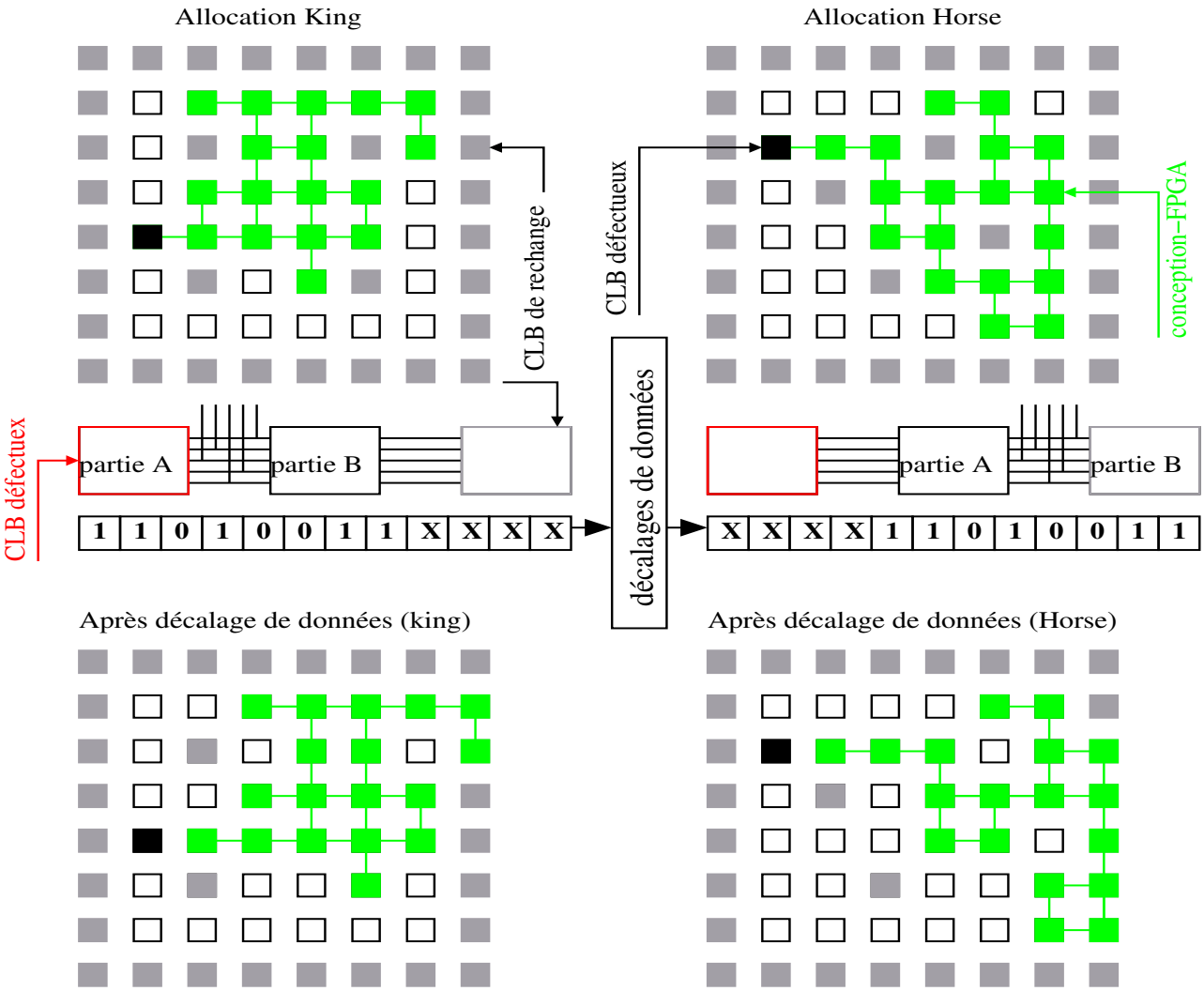


FIGURE 3.4: Schéma montrant un exemple de base de décalage de données par les deux méthodes : *King-shifting* et *Horse-allocation*.

3. Les auteurs supposent que la mémoire de configuration du FPGA-SRAM est composée de registres de décalage série.
4. La technique proposée ne traite pas les pannes dans les blocs d'entrées/sorties d'un FPGA-SRAM.

L'inconvénient majeur des techniques FTs basées sur l'allocation des ressources, c'est le fait de considérer que la colonne ou la ligne contenant un CLB défectueux comme une colonne ou ligne respectivement complètement défectueuse. Ceci engendre une baisse considérable au niveaux rendement. Une des proposition qui permet de contourner cet obstacle est la reconfiguration partielle pour réduire le temps de retard et les ressources logiques.

3.4.2 Tolérance aux pannes d'un FPGA-SRAM par partitionnement d'une conception-FPGA

Dans le but de minimiser les effets des techniques de la tolérance aux pannes sur le fonctionnement d'un système FPGA-SRAM, une technique FT intitulée *Tiling* a été proposé

dans [54][59]. Cette technique stipule un partitionnement de la conception-FPGA en un ensemble de tuiles *"tiles"*. Chaque *tile* est composé de CLBs, des ressources de routage, une interface qui spécifié la connectivité avec les *tiles* voisins et une portion de la *net-list* [4] global. Chaque *tile* contient un nombre déterminé de CLBs de rechange capables de remplacer les CLBs défectueux dans le *tile*. Le principe de la tolérance aux pannes de la méthode *Tiling* est basé sur l'approvisionnement de chaque *tile* de plusieurs configurations de la même portion de la conception-FPGA. Ces configurations sont stockées dans une mémoire. Quand une panne apparait dans un CLB d'un *tile*, celui ci utilise une configuration où le CLB défectueux n'interviendra pas. La figure 3.5 - schéma adaptée de [54] et [12] - montre un exemple de base illustrant la tolérance aux pannes d'un FPGA-SRAM par la méthode *Tiling*.

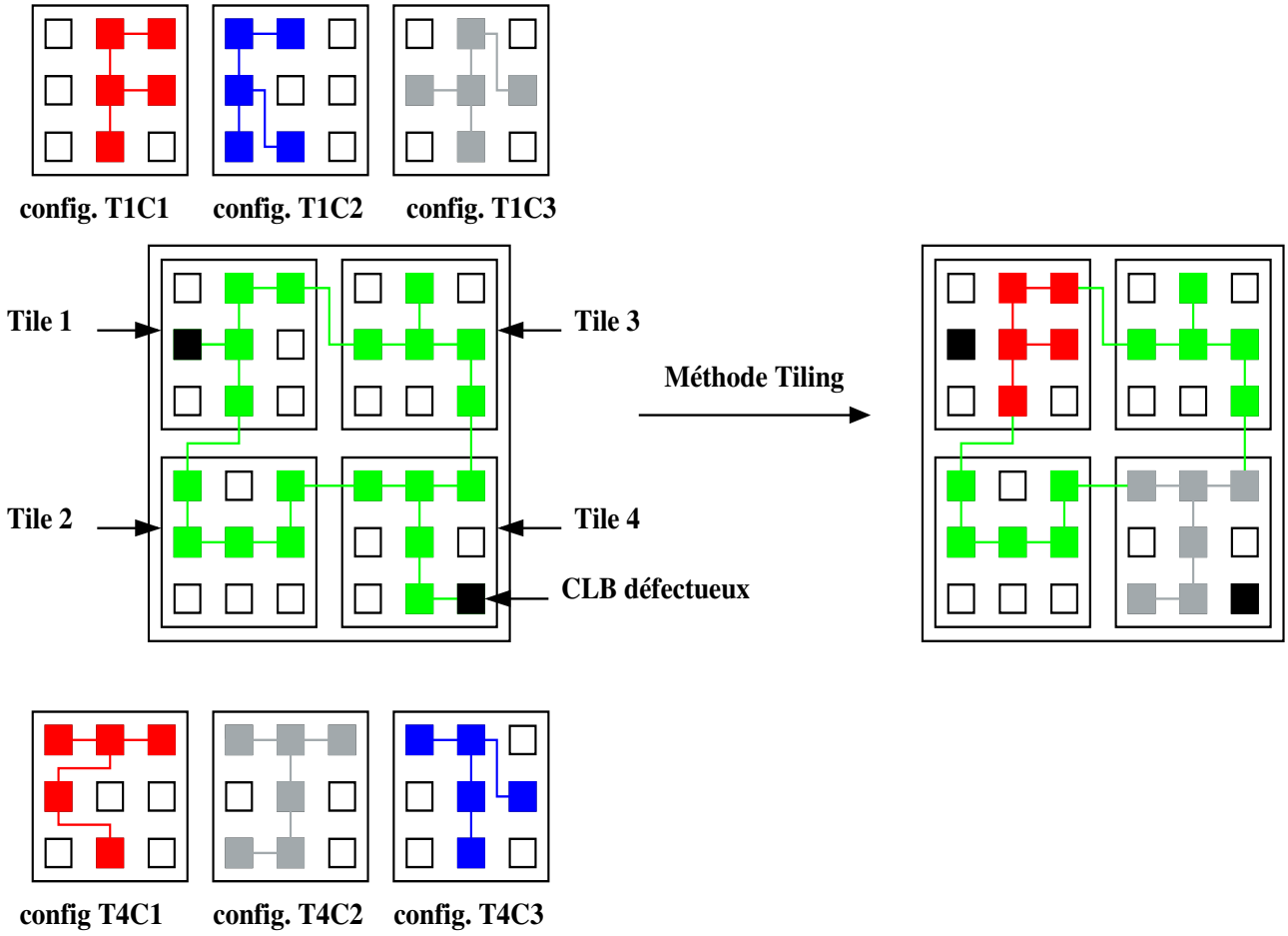


FIGURE 3.5: Schéma basique illustrant la tolérance aux pannes d'un FPGA-SRAM (réseau de 6×6 CLBs) par la méthode *Tiling* où la technique change les configurations dans le tile 1 et le tile 2 afin de ne pas utiliser les CLBs défectueux utilisés dans les configurations initiales.

La technique proposée dans [54] est caractérisée par les avantages suivants :

1. La technique *Tiling* vise à augmenter la fiabilité d'un système FPGA-SRAM.

4. Une *netlist* est définie dans [62] comme étant un fichier contenant une description textuelle d'une conception-FPGA, ce fichier contient des descriptions des sous-circuits, des portes logiques, des des blocs mémoires,... Les outils CAD de Xilinx par exemple permettent la conversion d'une netlist d'un FPGA-SRAM en un schéma.

2. La fiabilité est mesurée par le nombre de configurations qu'un tile pourrait supporté.
3. Comme une configuration d'un *tile* est significativement plus petite que la configuration globale d'une conception-FPGA, la méthode *Tiling* est plus rapide et moins coûteuse en terme de ressources.
4. La méthode ne requière pas une allocation des ressources comme des lignes ou colonnes de CLBs, elle consacre quelques ressources de rechange pour chaque *tile*, ce qui réduit le coût matériel.
5. La technique n'agit que sur le tile contenant des pannes sans reconfigurer le reste du FPGA-SRAM, ce qui réduit le temps de reconfiguration.

La technique *Tiling* souffre des inconvénients suivants :

1. La technique *Tiling* ne dispose pas d'un moyen de détection et de localisation des pannes dans les *tiles*.
2. Chaque tile ne dispose que d'un nombre limité de ressources de rechange (CLBs, ressources de routage), donc la technique *Tiling* ne peut tolérer qu'un nombre limité de pannes dans chaque *tile*.
3. La technique *Tiling* est applicable en premier lieu sur les CLBs et les ressources de routage qui entourent les CLB. Elle reste inefficace pour le reste des ressources de routage qui constituent une grande partie de l'architecture d'un FPGA-SRAM.

La pierre angulaire de la technique *Tiling* est sa flexibilité dans le processus de tolérance aux pannes. La technique vise dans son principe général à utiliser la reconfiguration partielle comme outil pour tolérer les pannes dans une région défectueuse d'un FPGA de technologie SRAM sans impacter le reste de ce dernier. Cette contribution a attiré toute l'intention des chercheurs afin de développer des techniques de tolérance aux pannes des FPGA-SRAMs basées sur la reconfiguration partielle.

3.4.3 Tolérance aux pannes des FPGA-SRAMs par reconfiguration

Les FPGA-SRAMs sont dotés de la reconfiguration dynamique et partielle, cet atout constitue un moyen précieux pour la tolérance aux pannes de ces circuits. En effet, dans [82][83] un FPGA-SRAMs est utilisé pour assurer la tolérance aux pannes d'un réseaux de processeurs grâce à la reconfiguration dynamique.

Les techniques émergentes de la reconfiguration partielle et dynamique permises par les FPGA-SRAMs récents ont été utilisées pour améliorer la tolérance aux pannes des FPGA-SRAMs [84]. La réimplantation partielle du *bitstream* (PBR) est parmi ces technique permettant la tolérance aux pannes en temps réel des FPGA-SRAMs par reconfiguartion partielle/dynamique. En effet, la réimplantation partielle du *bitstream* consiste à lire une partie du *bitstream* stocké dans une mémoire interne ou externe du FPGA-SRAM dont le contenu est généré pour une région partiellement reconfigurable (PBR) dans le circuit FPGA-SRAM et modifié selon le besoin pour une région différente du FPGA-SRAM [84]. La technique propose une réimplantation partielle du *bitstream* d'une manière non intrusive d'une région active (source) dans le FPGA-SRAM et la écrire dans une région différente (destination). La réimplantation partielle du *bitstream* est privilégiée pour la tolérance aux pannes des FPGA-SRAMs dans le cas où le temps de remplacement de la partie défectueuse de la

conception-FPGA par une autre partie fonctionnant correctement est critique comme le reparamétrage des noyaux dans le domaine de l'aviation et le traitement d'images dans les satellites [85]-[86]. La figure 3.6 - schéma adapté de [84] - montre les principales étapes de la technique PRR-PRR.

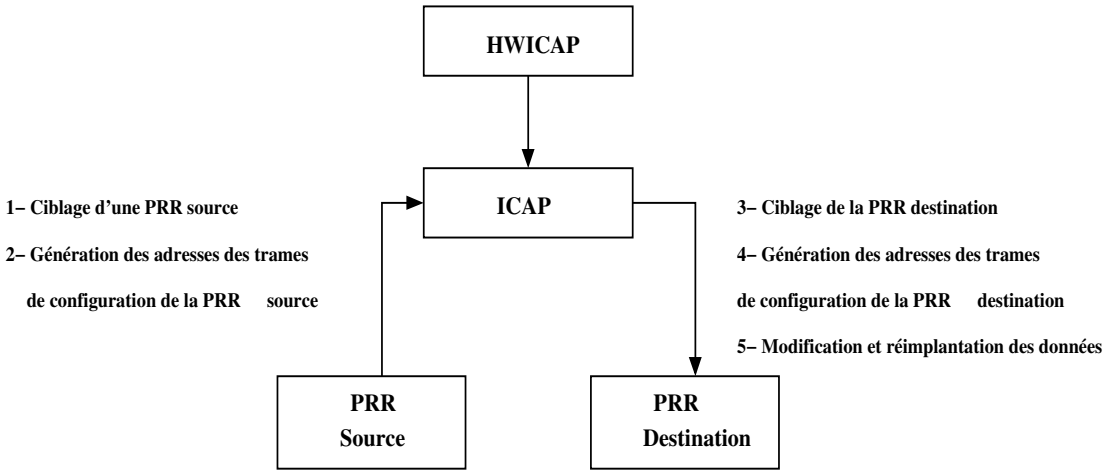


FIGURE 3.6: Schéma illustrant les principales étapes de la réimplantation partielle du *bitstream* PRR-PRR.

L'algorithme PRR-PRR proposé par les auteurs lit les données d'une trame de données d'une région PRR et la réimplanter dans une trame de données de la PRR destination à la volée,*i. e.*, sans passer par l'utilisation d'une mémoire. Le modèle d'analyse proposé par [84] pour tester l'algorithme a montré une précision de 95% pour 7 cas d'étude et 153 fois plus rapide que la méthode [87].

La capacité de modification ou de reconfiguration de la mémoire CMA d'un FPGA-SRAM rend la reconfiguration partielle et la reconfiguration dynamique un atout essentiel pour la tolérance aux pannes des FPGA-SRAMs [88]-[91].

3.4.4 Comparaison des approches de tolérance aux pannes des FPGA-SRAMs

De nombreux travaux dans la littérature ont suivi la progression des recherches concernant le test et la tolérance aux pannes des FPGA-SRAMs [12, 11, 92]. Dans le tableau 3.3 quelques travaux qui représentent des idées innovantes dans le domaine du test et la tolérance aux pannes des FPGA de technologie SRAM sont comparés.

Les systèmes auto-réparables (BISR) sont devenus un challenge pour les FPGA-SRAMs utilisés dans des environnements difficilement accessibles par l'humain ou dans des longues missions. Une technique BISR consiste en un système qui permet à un FPGA-SRAM de réaliser un auto-test (BIST) de localiser les pannes et donc de reconfigurer le FPGA-SRAM de telle manière à exclure les éléments défectueux dans l'architecture FPGA-SRAM et assurer ainsi un fonctionnement correcte de la fonction implémentée dans le FPGA-SRAM.

Comme nous l'avons indiqué dans la définition 3.1, une technique BISR constitue un système complet pour la tolérance aux pannes des FPGA à base de mémoire SRAM. Elle consiste généralement en un module d'auto-test (BIST) qui teste la partie logique ou le réseau de connexion d'un FPGA-SRAM ou les deux en même temps.

Technique FT	Système auto-réparable	Surcoût matériel	Détection et diagnostic	Pannes tolérées
TMR [93]	Oui	200%	Non	pannes logiques
SRAM Shifting [11]	Non	$\geq$ 10%	Non	pannes logiques et routage
Tiling [54]	Non	11%	Non	pannes logiques et routage
Fast-TAD [72]	Oui	Non déterminé	oui	pannes logiques
STAR [56]	Oui	$\leq$ 20%	limité	pannes dans les CLBs
[82]	Oui	ROM externe	oui	pannes logiques et routage

TABLE 3.3: Comparaison des approches de test et de tolérance aux pannes des FPGA-SRAMs.

Plusieurs contributions scientifiques ont été réalisées afin de proposer une technique BISR complète. Parmi ces travaux on trouve [94]-[56], ces méthodes exploitent l'architecture modulaire des FPGA-SRAMs ainsi que l'aspect reconfigurable de ces ICs pour la tolérance aux pannes.

3.5 Discussion

Dans ce chapitre nous avons donné un état de l'art des techniques de test et de tolérance aux pannes des FPGA-SRAMs. Les principaux points qu'on peut tirer de ce chapitre sont :

1. Les techniques de tests proposées dans la littérature sont destinées à détecter un ensemble précis de pannes, or plusieurs études ont montré l'existence de plusieurs modèles de pannes. Cette lacune exige des approches qui prennent en compte plusieurs modèles de pannes.
2. La majorité des techniques de tolérance aux pannes des FPGA-SRAMs supposent que les pannes ont été précédemment détectées et diagnostiquées et donc la technique de tolérance aux pannes n'a pour tâche que de tolérer ces pannes.
3. Dans la plupart des travaux qui traitent les tests et le diagnostic des FPGA-SRAMs, on remarque qu'il y a une absence des techniques de test qui détectent les pannes intermittentes, *i.e.*, des pannes qui apparaissent que si certaines conditions sont réunies. Nous traiterons ce genre de problématique dans le chapitre 6.
4. Concernant les techniques de tolérance aux pannes des FPGA-SRAMs qui visent la modification des layouts de ces derniers, il est possible que cette modification provoquera des répercussions négatives sur le bon fonctionnent des FPGA-SRAMs.
5. Les travaux de recherche menées dans le domaine de la tolérance aux pannes des FPGA-SRAMs sont dans la plupart des cas absorbés par une vision binaire dans le traitement des pannes. En effet, comme nous l'avons mentionné précédemment et dans le tableau 3.3 ces travaux visent les ressources logiques dans un FPGA de technologie SRAM en considérant les ressources de routage comme des ressources sans défauts et vice versa.

Sur la lumière de ces cinq points, notre contribution dans ce mémoire visera à s'échapper de cette vision binaire de la tolérance aux pannes des FPGA-SRAMs. Nous essayerons dans les chapitres qui suivent de regarder le circuit FPGA-SRAM sous un angle homogène afin d'établir une technique qui teste l'ensemble du FPGA-SRAM et donne par la suite une méthode de tolérance aux pannes regroupant les ressources logiques et de routage.

Conclusion :

Le test des circuits FPGA-SRAMs est un paramètre essentiel qui servent d'assurer un degré de fiabilité permettant le bon fonctionnement de ces architectures quelques soient les conditions et les environnements parfois critiques de déploiement des FPGA-SRAMs. La densité croissante d'intégration des FPGA-SRAMs et leurs complexités font que le domaine de test des FPGA de technologie SRAM est en pleine croissance. Le fabricant ainsi que l'utilisateur cherchent à développer des techniques de test qui prennent en considération la croissance rapide de la complexité des FPGAs à base de mémoire SRAM afin d'améliorer le rendement de ces circuits.

Le domaine de la tolérance aux pannes des FPGA-SRAMs - avec tout ce qui rassemble le terme "tolérance aux pannes" de techniques et approches - est un champs en perpétuelle mutation. Pour cette raison, nous avons cherché à donner une contribution à l'état de l'art sur le test, le diagnostique et la tolérance aux pannes des FPGA-SRAMs.

La mise en œuvre des FPGA-SRAMs dans la plupart des domaines exige que ces derniers soient dotés d'un système de tolérance aux pannes efficace.

Deuxième partie

MÉTHODES DE TOLÉRANCES AUX PANNES ET DE TEST DES FPGAS À BASE DE MÉMOIRES SRAM

Tolérance aux pannes des FPGA-SRAMs : Techniques et Analyse de la Fiabilité

4

« Current . . . (VLSI) technology allows the manufacture of large-area integrated circuits with submicrometer feature sizes, ... However, imperfections in the fabrication process result in yield-reducing manufacturing defects, whose severity grows proportionally with the size and density of the chip. Consequently, the development and use of yield-enhancement techniques at the design stage, to complement existing efforts at the manufacturing stage, is economically justifiable. Design-stage yield-enhancement techniques are aimed at making the integrated circuit "defect tolerant," i. e., less sensitive to manufacturing defects. They include incorporating redundancy into the design, modifying the circuit floorplan, and modifying its layout »

– I. Koren, Z. Koren 1998 paper "Defect Tolerance in VLSI Circuits : Techniques and Yield Analysis" [57]

Sommaire

Introduction au chapitre :

La pierre angulaire des techniques de tolérance aux pannes citées dans le chapitre 3 est la reconfiguration du FPGA-SRAM. En effet, cette dernière permet d'exclure les ressources défectueuses dans la nouvelle implémentation de la conception FPGA. Dans le présent chapitre, nous proposons deux techniques de tolérance aux pannes qui visent les pannes permanentes en premier degré. Ces techniques se basent uniquement sur les capacités de la reconfiguration des FPGA-SRAMs.

La première technique appelée méthode MST permet :
– la tolérance aux pannes de plusieurs pannes permanentes
– l'atténuation des pannes transitoires (SEUs) dans les régions CLB-M

La deuxième technique agit sur les cellules SRAM de la mémoire de configuration et permet :
– la tolérance aux pannes des trames de configuration de la mémoire CMA
– l'utilisation partielle des CLBs défectueux

Contrairement aux méthodes de conception dans les ASICs et les microprocesseurs qui aboutissent à des structures statiques, ces techniques permettent à un design physique de fournir différentes configurations alternatives à celle utilisant des ressources défectueuses. Les résultats expérimentaux effectués sur un ensemble de circuits ITC'99 démontrent un haut niveau de fiabilité accompagné d'un faible coût matériel.

4.1 Première technique : La méthode Master-Slave Technique

4.1.1 Motivation

La complexité des FPGA-SRAMs a atteint 6,8 milliards de transistors et sa fréquence d'horloge a dépassé un gigahertz [9]. En utilisant la technologie 2.5D (SSI) [1], les FPGA-SRAMs de Xilinx intègrent 2 millions de cellules logiques et son échelle d'intégration a atteint 28 nm. En contrepartie, ce progrès technologique rend les FPGA-SRAMs plus vulnérables autant aux pannes permanentes qu'aux pannes transitoires provenant de l'exposition de ces circuits à des particules chargées (Chapitre 5). Devant cette échelle d'intégration, les techniques de blindage des circuits FPGA-SRAMs demeurent inefficaces pour l'atténuation des SEUs. Par conséquent, ces tendances technologiques rendent les FPGA-SRAMs moins fiables.

1. La technologie Stacked Silicon Interconnect (SSI) permet l'empilement de multiples puces sur puces ou l'empilement de wafer sur wafer, avec des techniques d'amincissement de puces, dans le but d'aller vers des intégrations plus poussées. Ces techniques 3D permettent d'augmenter les performances (la bande passante entre un processeur et une mémoire), de réduire la consommation en puissance en remplaçant une longue connexion horizontale par une courte connexion verticale, de baisser les coûts de production en choisissant la technologie adaptée à la fonction recherchée et de réduire le facteur de forme [96].

La tolérance aux pannes est devenue donc un paramètre essentiel dans la conception des designs à implémenter sur un FPGA-SRAM [8, 13]. Parmi les méthodes de tolérance aux pannes, il y a la redondance modulaire triple (TMR). Or, comme nous le verrons dans la suite de ce chapitre, la redondance modulaire triple reste une technique coûteuse en terme de ressources et de consommation en puissance [8]. La réplication des circuits FPGA-SRAMs pour exécuter la même application est également peu attractive étant donné le coût élevé de la technique [54][59].

L'aspect flexible des FPGA-SRAMs - qui sont des architectures reconfigurables - fournit un support solide pour les techniques de tolérance aux pannes. En se basant sur la nature reconfigurable des FPGA-SRAMs, le principe de la tolérance aux pannes de ces circuits consiste en leur reconfiguration. La reconfiguration a pour but de ne pas utiliser les ressources défectueuses [54]. Cette stratégie est généralement accompagnée de l'allocation d'un pourcentage de ressources comme des ressources de rechange (spare allocation). Avec cette stratégie, le système reste fiable en dépit de l'existence de ressources défectueuses. Cependant, cette approche se traduit par un temps d'arrêt significatif du système FPGA-SRAM [54]. Elle exige également que l'utilisateur final possède les outils de placement et routage du fabricant, ce qui n'est pas généralement le cas. Ainsi, la technique ne sera pas efficace pour des applications qui ont des contraintes critiques sur le temps.

Nous proposons dans ce chapitre une technique de tolérance aux pannes d'un FPGA à base de mémoire SRAM. Cette technique est applicable à une large gamme d'architectures FPGA-SRAMs puisque elle ne dépend pas d'une architecture donnée. Nous proposons le partitionnement physique du circuit implémenté en un ensemble d'unités appelées Master-Slave Unit (MSU). Chaque MSU est composée d'un ensemble de ressources logique et de routage (CLBs et SBs), d'une spécification d'interface qui dénote la connectivité des MSUs voisines et d'une partie de la netlist. La fiabilité est obtenue en fournissant de multiples configurations à chaque MSU. En outre, en déterminant les interfaces de chaque MSU, les effets de changement de la configuration d'une MSU ne se propagent pas à d'autres MSUs, fixant ainsi la région reconfigurable du FPGA-SRAM.

La figure 4.1 montre un exemple d'implémentation d'une partie d'un design dans une MSU. La figure 4.1 (I) montre une configuration où la technique TMR est appliquée sur le circuit 1. La figure 4.1 (II) illustre une configuration alternative du MSU dans le cas où les CLBs implémentant les circuits 4 et 5 sont défectueux. Chaque reconfiguration est interchangeable avec le MSU initiale. En effet, étant donnée que l'interface entre chaque MSU et les autres est fixe, la fonction du MSU reste inchangeable. Le timing de la conception implémentée pourrait alors varier à cause du changement dans le routage.

Dans le cas de pannes dans plusieurs CLBs, la méthode publiée dans [54][59] n'est pas applicable puisque le nombre de CLBs de rechange est inférieur à celui des CLBs défectueux. Nous résolvons ce problème en reconfigurant l'architecture MSU pour remplacer les CLBs défectueux. L'approche proposée nécessite une étape de prétraitement de détection des pannes et de diagnostic. Dans sa forme actuelle, notre approche ne tient pas compte de la tolérance aux pannes des interconnexions. Les pannes dans les interconnexions locales seront considérées comme étant dans le CLB en question.

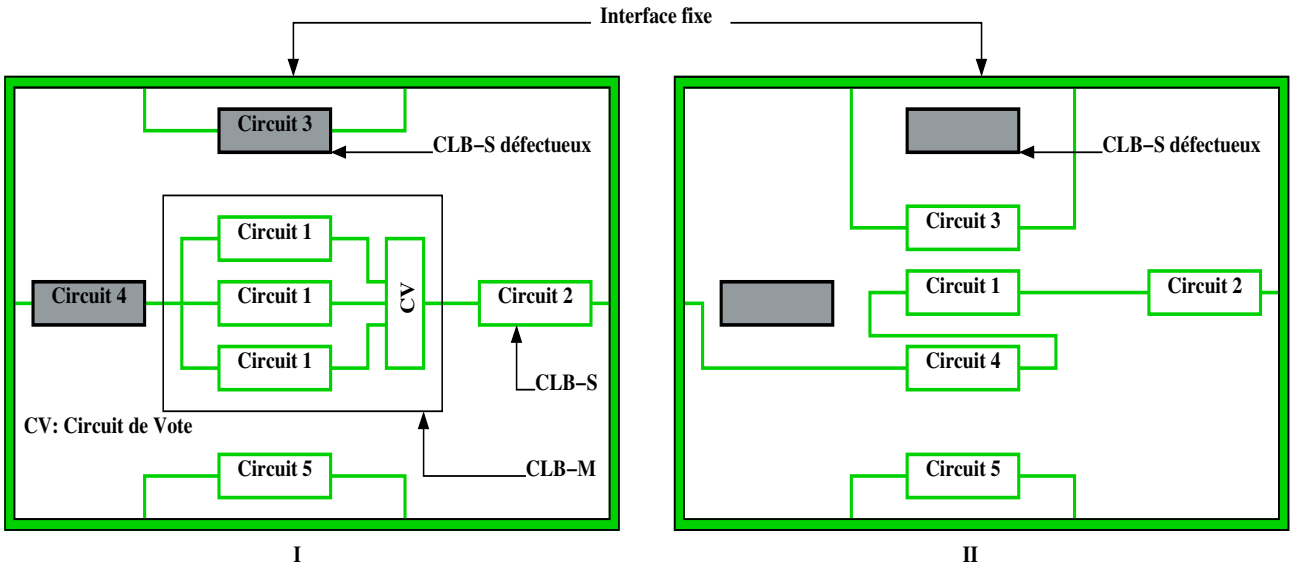


FIGURE 4.1: Illustration de la technique "Master-Slave Technique" excluant les CLBs défectueux.

4.1.2 Principe de l'approche

Le principe de la méthode MST est tel que : si la nouvelle configuration met en œuvre la même fonction que l'initiale, tout en évitant la partie défectueuse, le système peut être redémarré. L'étape difficile serait l'identification d'une configuration alternative efficace. Dans cette section, nous exposons les éléments clés de cette approche.

Principe

Dans le cas de plusieurs pannes dans plusieurs CLBs dans un groupe de CLBs, la méthode publiée dans [54][59] n'est pas applicable puisque le nombre de CLBs de rechange (spare) est inférieur à celui des CLBs défectueux. Pour faire face à ce problème de pannes multiples, nous proposons donc l'architecture illustrée dans la figure 4.2.

Cette approche a trois avantages principaux par rapport aux méthodes de tolérance aux pannes des FPGA-SRAMs. Comparée à la technique TMR [93], elle a un coût matériel faible, une gestion en temps d'exécution de la conception implémentée et une flexibilité complète. Le coût matériel nécessaire pour cette approche à grain fin peut être mesuré en ressources physiques (CLBs et ressources de routage) et en timing. La gestion en temps d'exécution d'un système basé sur un FPGA-SRAM peut être une caractéristique très importante pour les applications critiques. Cette technique de tolérance aux pannes serait capable de traiter en ligne les problèmes de pannes, minimiser le temps d'arrêt du système et éliminer la nécessité d'une intervention extérieure. La flexibilité apportée par cette approche fournit des solutions aux applications spécifiques. Le degré de la tolérance aux pannes peut en effet être modifié en fonction des contraintes de temps, des limitations de ressources ou de la fiabilité des CLBs.

Dans ce qui suit, nous continuons les explications de la méthode proposée et de ses limites. La démonstration de la nouvelle approche de tolérance aux pannes est faite en utilisant la carte de développement ML507 équipée d'un FPGA-SRAM Virtex-5 XC5VFX70T de Xilinx

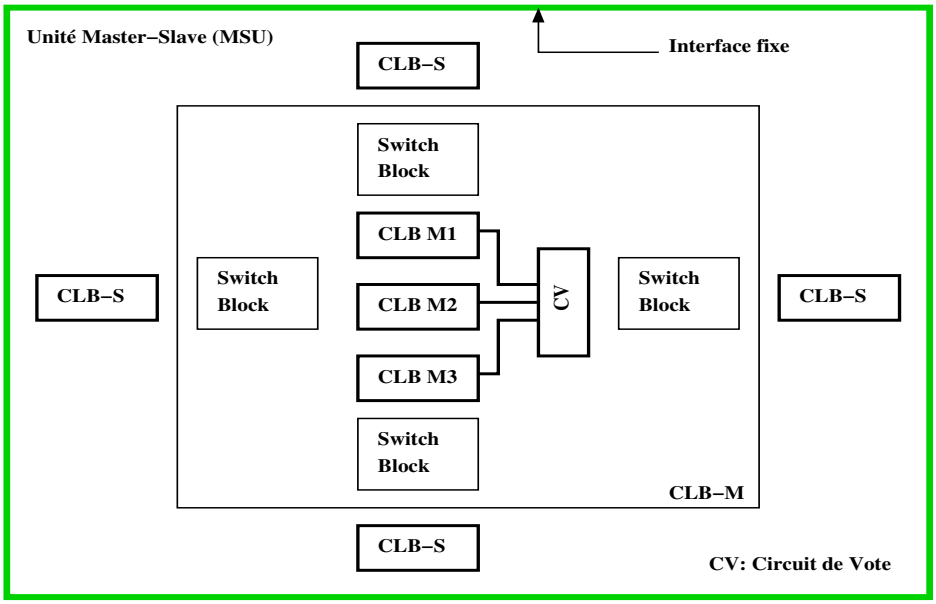


FIGURE 4.2: Illustration de l'architecture "Master-Slave Unit" (MSU), où chaque CLB-M est entouré par quatre CLB-S dans une unité Master-Slave (MSU).

[97]. Toutefois, ni le concept général, ni les algorithmes d'optimisation ne sont spécifiques à la familles Virtex-5 ou même à l'architecture Xilinx. Toute architecture FPGA-SRAM pouvant supporter plusieurs reconfigurations partielles pourrait être utilisée, comme les FPGA-SRAMs de Altera et de Lattice. Néanmoins, l'approche n'est pas applicable aux FPGAs antifusibles, telles que l'architecture Actel, puisque ce sont des technologies OTP.

Approche

Dans notre approche, nous réduisons la quantité de mémoire de configuration requise en déterminant la région à reconfigurer dans le FPGA-SRAM. Ceci est possible par le partitionnement de la logique d'un design de manière à ce que les composants puissent être indépendamment reconfigurés sans impact sur le reste de la conception. En comparaison avec d'autres solutions, cette approche permet également de réduire le temps d'arrêt d'un système à base de FPGA-SRAM. La technique est aussi capable d'améliorer la tolérance aux pannes en offrant la possibilité de tolérer des pannes multiples dans différents CLBs avec un coût matériel acceptable. Les éléments clés de la mise en œuvre de notre concept sont :

1. les unités MSUs
2. les blocs Macro-MSU
3. la technique Master-Slave (MST)

La combinaison entre ces trois éléments permet la tolérance aux pannes multiples. Nous définissons dans la suite les éléments énumérés ci-dessus.

Définition 4.1. *Une unité MSU (Master-Slave Unit) est composée de deux architectures de CLBs (CLB-M et CLB-S, désignant respectivement des CLBs masters et des CLBs slaves), de ressources de routage, d'une partie de la netlist et d'une spécification de l'interfaçage avec les MSUs adjacentes.*

La figure 4.2 illustre l'architecture d'une unité MSU où chaque CLB-M est entouré par quatre CLB-Ss.

Définition 4.2. *Une Macro-MSU (M-MSU) est composée d'un ensemble de MSUs afin de faciliter le partitionnement d'un design dans un FPGA-SRAM. Une M-MSU spécifie aussi la façon d'interfacer une M-MSU avec les autres M-MSUs adjacentes de telle sorte qu'un changement dans une unité M-MSU n'affectera pas les autres M-MSUs.*

Maintenant, nous définissons l'architecture CLB-M ainsi que leurs relations avec les CLB-Ss.

Définition 4.3. *Un CLB-M est une architecture qui regroupe trois CLBs auxquels s'ajoute un circuit de vote (CV). Le CLB-M implémente trois copies de la même fonction en appliquant la technique TMR. Dans le cas d'un CLB défectueux dans une unité MSU, une nouvelle configuration implémentant la même fonction que l'initiale et excluant le CLB défectueux, configure l'unité MSU.*

Définition 4.4. *Les CLB-S sont les CLBs qui entourent les CLB-M. En utilisant la technique MST, un CLB-S défectueux peut être remplacé par un CLB de l'architecture CLB-M dans la même unité MSU.*

Comme montré dans la figure 4.3, la fonction implémentée dans l'unité M-MSU (A) peut être implémentée suivant plusieurs fichiers binaires partiels : `E_1.bit`, `E_2.bit`, `E_3.bit`. Chaque fichier binaire `E_i.bit` avec $i \in \{1, 2, 3, \ldots\}$ présente alors la même fonction mais utilise des ressources différentes [2].

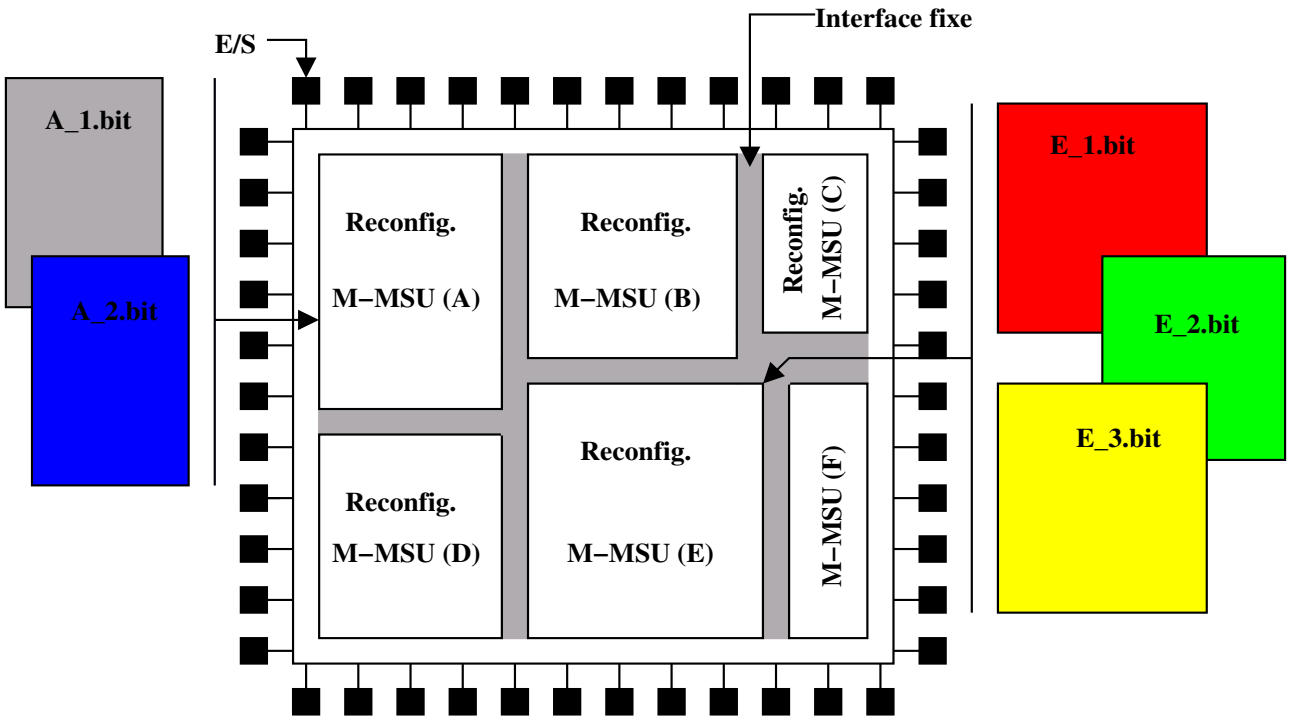


FIGURE 4.3: Schéma de base de la reconfiguration d'un bloc M-MSU.

2. Dans la figure 4.3 nous n'avons pas représenté la partie statique du FPGA-SRAM, ce qu'il faut prendre en compte lors de la reconfiguration partielle.

Considérons à nouveau la figure 4.3 et supposons que la matrice des CLBs est constituée de $N \times M$ CLBs. Supposons un design qui doit être en mesure de tolérer toute panne dans un CLB. Il est possible de subdiviser la matrice des CLBs en 3×3 M-MSUs. Supposons maintenant qu'une configuration du design complet requière **X** octets de mémoire, l'approche de tolérance aux pannes qui consiste à changer toute la configuration requière $(N \times M) * \mathbf{X}$ octets de mémoire pour tolérer une panne. Avec notre approche on décompose le design de $N \times M$ CLBs en 3×3 M-MSUs de $n \times m$ CLBs chacune. Toutefois, étant donné que chaque M-MSU (**X**/9 octets de de mémoire) est indépendante, la mémoire nécessaire pour configurer une MSU est de seulement ($n \times m$)***X** octets, soit 88,88% de réduction par rapport au système de tolérance aux pannes utilisant une configuration complète.

La technique MST améliore également la fiabilité. Dans cet exemple, un système de tolérance aux pannes peut tolérer un seul CLB défectueux dans l'ensemble du design. Cependant, la technique MST est capable de tolérer plusieurs pannes dans une unité M-MSU et jusqu'à quatre CLBs par MSU.

Le coût d'une augmentation de la tolérance aux pannes et de la réduction de la mémoire de reconfiguration est dû à l'utilisation de plus de CLBs comme CLBs de rechange. Contrairement aux autres méthodes étudiées dans le chapitre 3, notre technique de tolérance aux pannes ne réserve pas de CLBs de rechange. Les CLBs destinés à remplacer des éventuels CLBs défectueux constituent les CLB-Ms. Cette stratégie permet d'augmenter la puissance dynamique consommée mais l'architecture CLB-Ms et leur distribution dans le design permettent l'atténuation des pannes SEUs. En effet, les CLB-Ms permettent d'atténuer les pannes SEUs comme montré dans la figure 4.4.

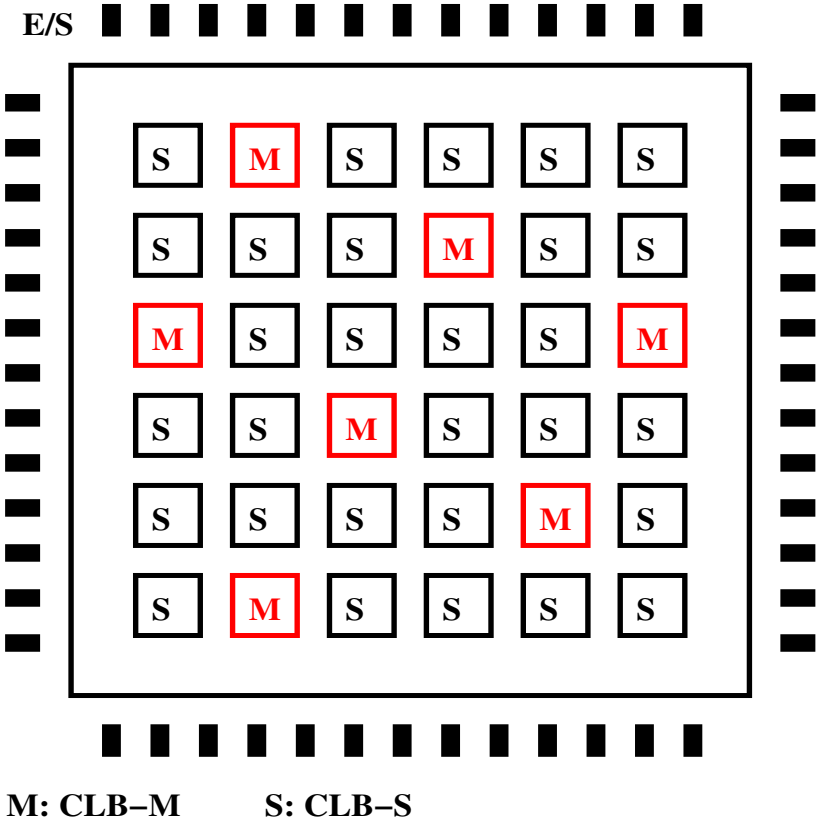


FIGURE 4.4: L'architecture et la distribution des CLB-Ms permettent d'atténuer les SEUs.

La figure 4.5 illustre la technique TMR utilisée dans l'architecture CLB-M. Cette technique implémente une fonction trois fois et les résultats générés par ces trois modules sont comparés via un circuit de vote. Les deux résultats qui sont identiques sont considérés comme étant corrects et le résultat non identique aux deux autres résultats est rejeté. La figure 4.5 contient trois modules représentant trois implémentations de la même fonction. Chaque module a une sortie connectée au circuit de vote VC3. Ce circuit implémente la fonction **(S1 AND S2) OR (S2 AND S3) OR (S1 AND S3)** et fournit le résultat à la sortie **S**.

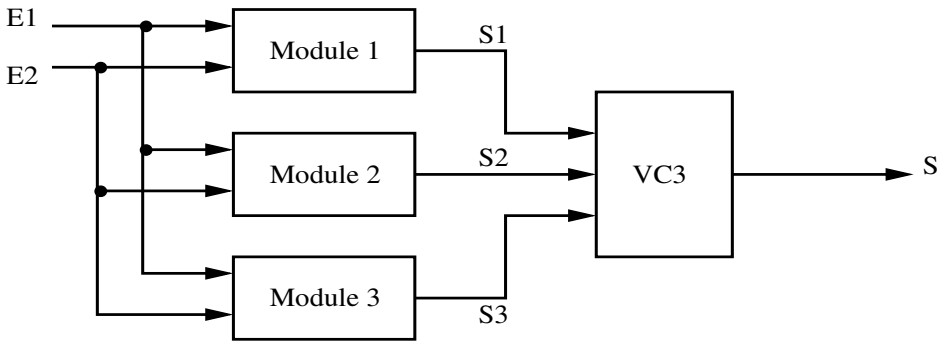


FIGURE 4.5: Illustration de la technique de redondance modulaire triple (TMR) utilisée pour l'architecture CLB-M.

De toute évidence, cette approche permet de surmonter toute panne qui affecte la fonctionnalité d'un des trois modules. Le module affecté produit une réponse erronée qui est substituée dans le circuit de vote par les deux autres modules.

D'après les trois dernières définitions (4.3, 4.4 et 4.1), l'architecture proposée pour la tolérance aux pannes des FPGAs à base de mémoire SRAM consiste au partitionnement du FPGA-SRAM en des ensembles de CLBs (MSUs). Ces derniers comportent des CLB-M qui constituent l'élément clé de la technique MST.

Étant donnée que l'unité MSU est associée à la fois à des ressources physiques et à des portions de la *netlist* complète, une conception ne peut être partitionnée en unités MSUs qu'après être passée par l'étape P&R. En fixant les interfaces entre les MSUs, nous créons la possibilité de produire de multiples configurations partielles qui satisfont la spécification fonctionnelle d'une MSU donnée, indépendamment du reste de la conception. La tolérance aux pannes est obtenue par l'introduction des CLBs du CLB-M de chaque MSU comme ressources de rechange. De cette manière, une fois une ou plusieurs pannes détectées dans un CLB-S particulier, une configuration de la fonctionnalité du MSU qui n'utilise pas le CLB défectueux peut être activée.

Pré-allocation des CLB-Ms

Soit un FPGA-SRAM défini par une matrice de $N \times M$ CLBs, chaque CLB est défini par ses coordonnées (r, c) tels que : $(r, c) \in \{1, \ldots, N\} \times \{1, \ldots, M\}$.
On pose $\delta = r \pmod 5$, la formule de pré-allocation des CLB-M est donnée par :

$$c \pmod 5 = (2\delta + 1) \pmod 5 \tag{4.1}$$

Si les coordonnées (i, j) d'un CLB dans la matrice $N \times M$ vérifient l'équation 4.1, alors le CLB de coordonnées (i, j) est alloué comme un CLB-M. La figure 4.6 montre un exemple de matrice de CLBs de taille 6×6 partitionnée en MSUs.

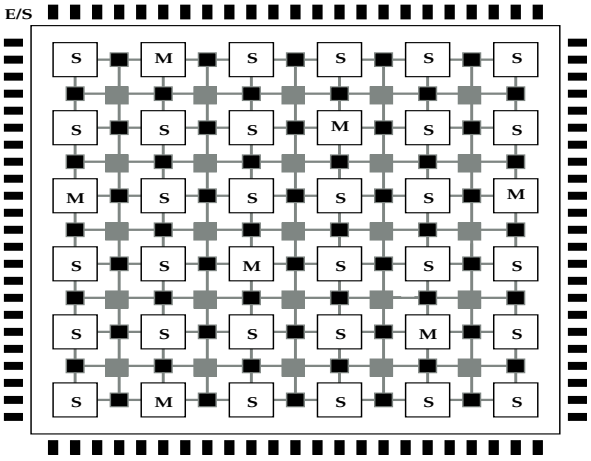


FIGURE 4.6: Partitionnement d'une matrice de CLBs de taille 6×6 en 7 MSUs, où chaque CLB-M (M) est entouré par quatre ou trois CLB-S (S).

Remarque 4.1. *Pour la pré-allocation des CLB-Ms nous avons utilisé l'équation 4.1, elle était utilisée dans [56] pour l'allocation des CLBs de rechange (spare).*

Méthode de synthèse

L'approche de synthèse est organisée en un processus itératif. Nous commençons avec un design dépourvu de tout système de tolérance aux pannes (design non-FT), puis d'une manière récursive, on procède à un partitionnement du design non-FT en M-MSUs et MSUs. Nous vérifions ensuite la faisabilité de tous les scénarios de panne avec un niveau de difficultés décroissant. L'idée est de déterminer le plus tôt possible les conceptions qui ne conduiront pas à une solution faisable. L'algorithme proposé calcule aussi la fiabilité du design. Le concepteur a alors la possibilité de mettre fin à ce calcul et continuer sur le design qui a un degré élevé de fiabilité. La synthèse est résumée dans l'algorithme montré dans la figure 4.7.

Dans cet algorithme, le processus MST est initialisé par le design non-FT. Les outils P&R créent le design non-FT et enregistrent les caractéristiques pertinentes du design. La procédure pour le calcul de fiabilité est expliquée dans le paragraphe 4.1.3.

L'algorithme de synthèse impose que la boucle se termine lors de la création d'une conception tolérante aux pannes qui réponde à toutes les spécifications de l'utilisateur. Ces spécifications comprennent le coût en ressources et en timing, le niveau de la tolérance aux pannes ainsi que la mémoire disponible. L'algorithme se termine également si l'algorithme atteint toutes les possibilités d'allocation des CLB-Ms dans une M-MSU en utilisant l'équation 4.1 et en commençant par des coordonnées de CLBs différents. Si les spécifications ne peuvent pas être satisfaites pour le design FT généré, alors l'algorithme sera répété en utilisant d'autres dimensions de la M-MSU, augmentant ainsi le nombre d'unités MSUs pour répondre aux spécifications du design.

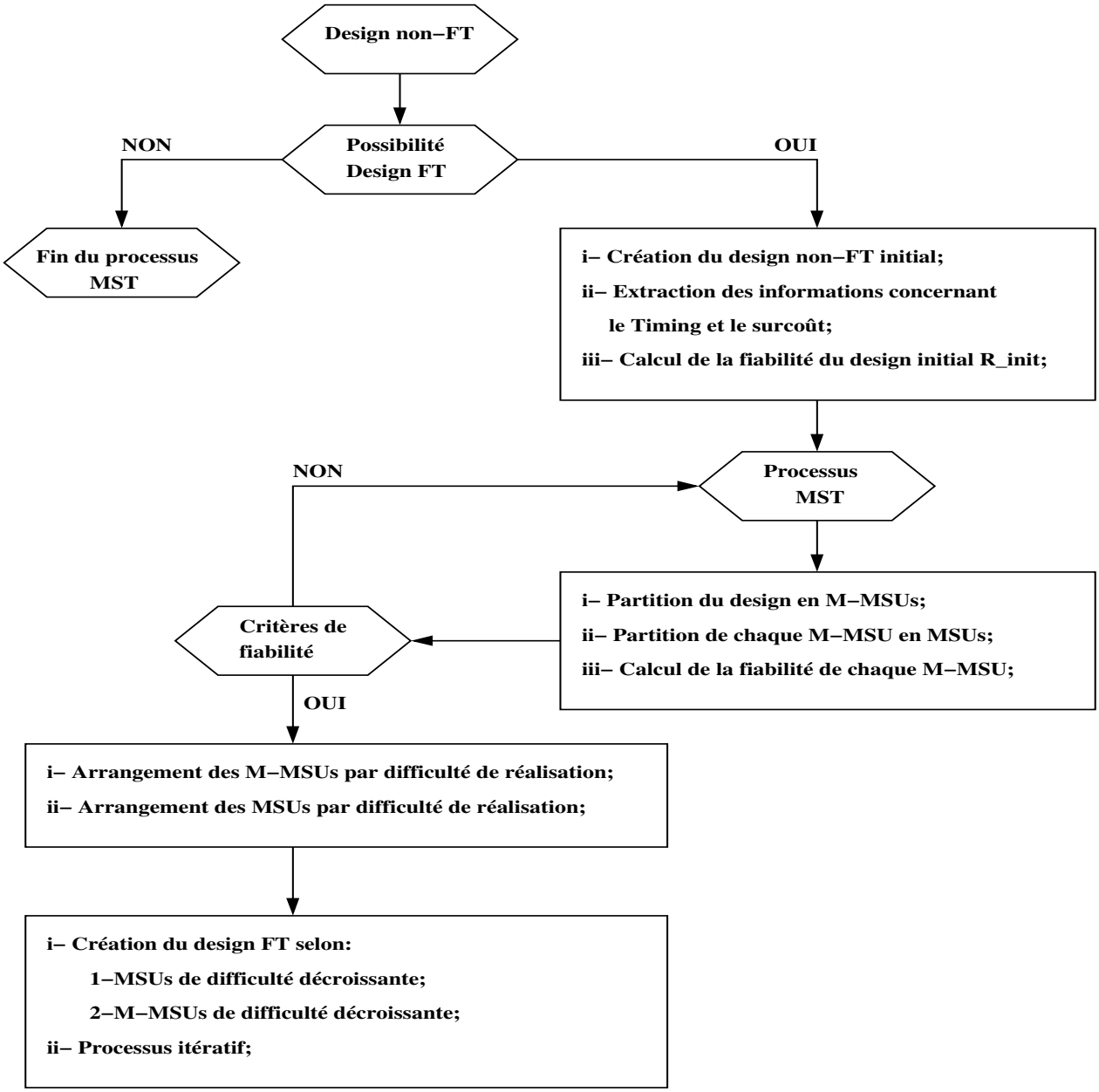


FIGURE 4.7: Principe algorithmique de la technique MST.

Dans l'étape suivante l'algorithme subdivise le design en M-MSUs comme décrit dans le paragraphe 4.1.2. Le placement et la forme des M-MSUs sont déterminés par les trois facteurs clés suivants, cités dans un ordre décroissant d'importance :

1. les ressources de routage disponibles sur les interfaces de l'unité M-MSU
2. la densité logique de la M-MSU
3. la taille de la M-MSU

La fiabilité calculée dans le paragraphe 4.1.3 est ensuite élevée, les M-MSUs étant larges. Si la méthode MST ne répond pas aux spécifications (ressources, timing), l'algorithme retourne au début de la boucle de partitionnement pour une nouvelle tentative de partition-

nement du design en unités M-MSUs. Les hard-macros et la chaîne de retenue (fast carry chains) influent également sur la forme et le placement d'une M-MSU.

L'algorithme de partitionnement qui se termine lors de la création d'un design tolérant aux pannes répondant aux spécifications de l'utilisateur ou à l'épuisement de toutes les possibilités de partitionnement. Si c'est le cas, l'algorithme retourne à l'étape critères de fiabilité.

L'algorithme veille aussi à ce que les partitions d'une M-MSU en MSUs répondent aux spécifications de la fiabilité déterminées par l'utilisateur. L'algorithme fait retourner l'algorithme au début de la boucle de partition en MSUs si les spécifications ne sont pas atteintes. Si les possibilités de créer des MSUs sont épuisées, l'algorithme relance à nouveau le processus MST.

La dernière phase de l'algorithme facilite la détection des processus de synthèse difficiles à réaliser. Les M-MSUs et MSUs qui sont moins susceptibles de réussir doivent être utilisées comme dernière solution. Ainsi l'algorithme retourne au début de la boucle si les spécifications ne sont pas respectées.

L'algorithme applique l'ordre défini par la dernière phase. L'algorithme configure les différents MSUs. Si la reconfiguration ne peut pas être atteinte ou que la configuration ne répond pas aux spécifications de timing de l'utilisateur, l'algorithme recommence une nouvelle partition.

4.1.3 Évaluation de la fiabilité apportée par la technique MST

La première étape dans le calcul de fiabilité d'un design implémenté dans un FPGA est de déterminer le modèle de pannes. Comme la technique MST vise en premier lieu les pannes permanentes, nous calculons la fiabilité des designs en se basant sur ce modèle de panne. Nous définissons le terme fiabilité comme dans [98] et [99] adapté à un système FPGA-SRAM.

Définition 4.5. *La fiabilité est définie comme étant l'aptitude d'un FPGA-SRAM à implémenter une conception FPGA, dans des conditions données, pendant une durée donnée. Le terme fiabilité est aussi utilisé comme caractéristique désignant une probabilité de succès ou un pourcentage de succès.*

Pannes indépendantes uniformément réparties

Si la probabilité qu'un CLB fonctionne correctement est $R_{CLB}(t)$, la probabilité que ce CLB tombe en panne est $(1 - R_{CLB}(t))$. La fiabilité d'un design sans technique de tolérance aux pannes utilisant n CLBs est donnée par l'équation 4.2 :

$$R_{init}(t) = (R_{CLB}(t))^n \tag{4.2}$$

Soient $R_{MST}(t)$ la fiabilité d'un design utilisant la technique MST et $R_{MSU}(t)$ la fiabilité d'une unité MSU, $R_{MSU}(t)$ est donnée par l'équation 4.3 :

$$R_{MSU}(t) = \sum_{i=0}^{n} \binom{l}{i} R_{CLB}(t)^{l-i}(1 - R_{CLB}(t))^i \tag{4.3}$$

Où n est le nombre de CLBs qui peuvent remplacer un CLB défectueux dans une MSU et l le nombre total des CLBs dans une MSU.

La fiabilité totale apportée par la technique MST est donnée par l'équation 4.4 :

$$R_{MST}(t) = \prod_{j=1}^{m} R_{MSU}(t) = (R_{MSU}(t))^m \tag{4.4}$$

Où m est le nombre de MSUs obtenu en appliquant la technique MST sur le design. L'égalité des deux termes est due au fait que chaque MSU a la même fiabilité.

Résultats

Le tableau 4.2 calcule fiabilité apportée par l'approche proposée et fait une comparaison avec celle de la méthode *Tiling* [54] ainsi qu'avec celle d'un design non-FT.

CLB	100 CLB design			1000 CLB design			5000 CLB design		
Reliability	**Orig.**	**Tiled**	**MST**	**Orig.**	**Tiled**	**MST**	**Orig.**	**Tiled**	**MST**
0,9500	0,005921	0,444669	0,977090	0,000000	0,000302	0,793137	0,000000	0,000000	0,313863
0,9750	0,079551	0,800119	0,996995	0,000000	0,107534	0,970356	0,000000	0,000014	0,860311
0,9800	0,132687	0,864375	0,998448	0,000000	0,232820	0,984595	0,000000	0,000684	0,925313
0,9850	0,220739	0,919633	0,999340	0,000000	0,432660	0,993422	0,000000	0,015161	0,967542
0,9900	0,366277	0,962643	0,999803	0,000043	0,683364	0,998031	0,000000	0,149026	0,990197
0,9950	0,606224	0,990317	0,999975	0,006704	0,907280	0,999751	0,000000	0,614762	0,998760
0,9980	0,819220	0,998429	0,999998	0,136145	0,984404	0,999984	0,000047	0,924414	0,999920
0,9990	0,905528	0,999608	0,999999	0,370696	0,996091	0,999998	0,007000	0,980610	0,999990
0,9995	0,951999	0,999903	0,999999	0,611453	0,999028	0,999999	0,085470	0,995153	0,999998
0,9999	0,990868	0.999995	0,999999	0.912346	0.999952	0.999999	0,632119	0,999762	0,999999

TABLE 4.2: Fiabilité apportée par la méthode MST contre celle de la méthode *Tiling*.

La figure 4.8 représente la fiabilité apportée par la technique MST comparée à celles de la méthode *Tiling* et d'un design sans système de tolérance aux pannes.
Dans le cas d'un design utilisant 5000 CLBs et pour des CLBs de fiabilité $R_{CLB}(t) = 0.999$, la fiabilité du design initial (sans système de tolérance aux pannes) est de moins de 1%. Celle apportée par la technique *Tiling* est de 98.00% tandis que la fiabilité apportée par la technique MST est 99.90%.

4.1.4 Résultats expérimentaux

Nous avons effectué une implémentation de la technique MST en utilisant 15 circuits de référence de ITC'99 [3]. Nous avons utilisé la carte d'évaluation ML507 équipée du FPGA

3. Les circuits de référence ITC'99 sont un ensemble de circuits fournies par des compagnies électroniques. Ils sont destinés à être utilisés pour le **Design for Testability** (DFT) et **Automatic Test Pattern Generation** (ATPG). Ils ont été présentés dans la conférence **International Test Conference** 1999 à Atlantic City, New Jersey.

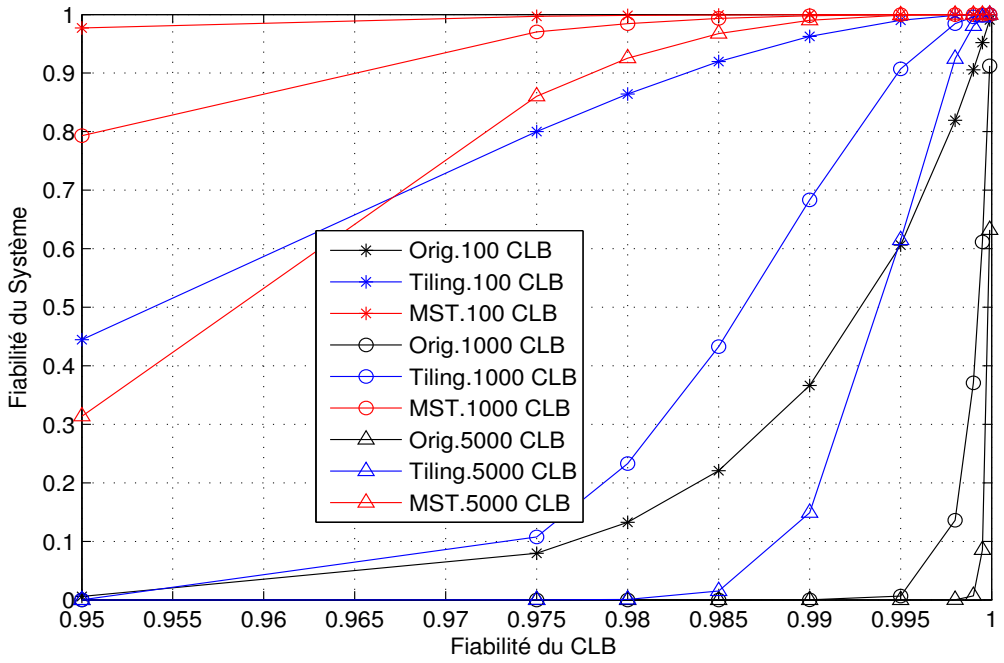


FIGURE 4.8: Fiabilité apportée par la technique MST vs celle apportée par la méthode *Tiling* et celle d'un design sans technique de tolérance aux pannes.

Xilinx Virtex-5 XC5VFX70T (figure A.2 Annexe A) ainsi que le kit ISE 11.5 pour l'implémentation de ces circuits.

Stratégie d'implémentation

Dans un premier temps, nous avons implémenté les circuits de référence ITC'99 en utilisant les outils de synthèse du kit ISE 11.5. Nous avons extrait le nombre de LUTs utilisées dans le design initial ainsi que d'autres informations. Comme les CLBs dans le FPGA Xilinx Virtex-5 sont constitués de deux *Slices* et que chaque *Slice* est constitué de quatre 6-LUTs (figure A.1 Annexe A), l'outil "`View Technology Schematic`" du ISE 11.5 ne montre que les LUTs utilisées et pas les CLBs auxquels ces LUTs appartiennent. Dans un second temps, nous avons utilisé l'outil "`View Technology Schematic`" pour extraire la fonction implémentée dans une LUT ainsi que la librairie "`Library UNSIM`" et le *package* "`use UNSIM.Vcomponents.all`" pour instancier les primitives de Xilinx.

Première implémentation de la technique MST

L'utilisation des CLBs de rechange dans une technique de tolérance aux pannes pour un FPGA de technologie SRAM facilite la tolérance aux pannes. Cependant, en règle générale, ces ressources ajoutées introduisent un coût supplémentaire. C'est pourquoi la quantité des ressources de rechange ajoutée doit être raisonnable. Dans notre étude, le nombre de ressources de rechange est mesuré par le nombre de LUTs de rechange.
Le tableau 4.3 montre une première implémentation de la technique MST tenant compte des pannes permanentes et des pannes SEUs. Le coût élevé de la technique MST revient

Design ITC'99	# initiale de LUTs	# Technique MST	# Technique TMR
b01	10	16	40
b02	4	7	16
b03	39	63	156
b04	115	184	460
b05	184	295*	736*
b06	8	14	24
b07	99	159	396
b08	23	38	92
b09	49	79	196
b10	38	62	152
b11	100	160	400
b12	261	420*	1044*
b13	80	128	320
b14	1218	1950*	4872*
b14_1	1291	2065*	5164*

TABLE 4.3: Variation des ressources (LUTs) utilisées avant et après l'application de la technique MST versus la technique TMR.

à utiliser une 3-LUT pour implémenter le circuit de vote dans chaque CLB-M. La technique TMR consomme 4 fois plus de ressources logiques tandis que la technique MST en consomme entre 1.59 et 1.75 fois plus.

Remarque 4.2. *Les ressources logiques des circuits de référence ITC'99 marquées par (*) sont des estimations des ressources logiques utilisées par l'application des méthodes MST et TMR sur les circuits originaux.*

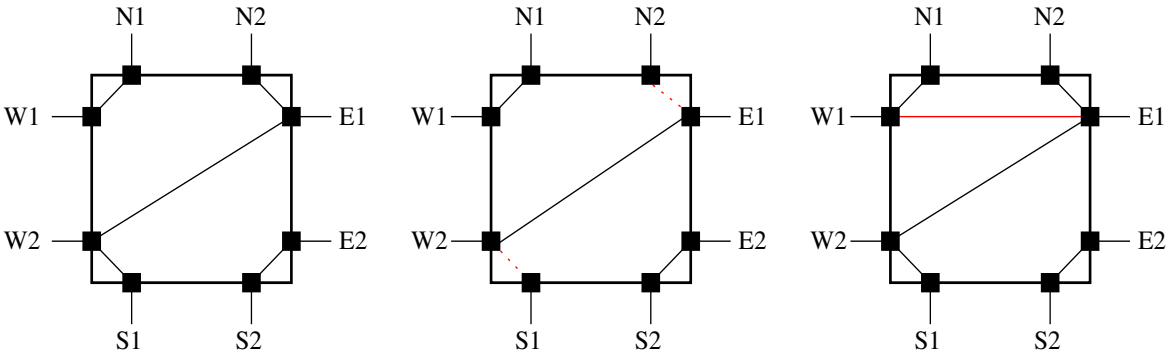
4.2 Deuxième technique : Tolérance aux pannes des FPGAs via les trames de configuration

Bien que les techniques de tolérance aux pannes des FPGA-SRAMs sont généralement plus performantes, elles ont un coût matériel élevé. Nous avons remarqué dans le chapitre 3 que la majorité des techniques de tolérance aux pannes ne traite qu'une seule partie du FPGA-SRAM, soit la partie logique soit la partie routage. Dans cette partie du chapitre 4, nous proposons une méthode de tolérance aux pannes qui agit sur les cellules SRAMs des trames de configuration pour tolérer en même temps les pannes logiques et les pannes du réseau de routage.

4.2.1 Motivation

Une panne dans une cellule SRAM (stuck-on ou stuck-off) d'une trame de configuration peut conduire à des erreurs dans le routage dans un SB :

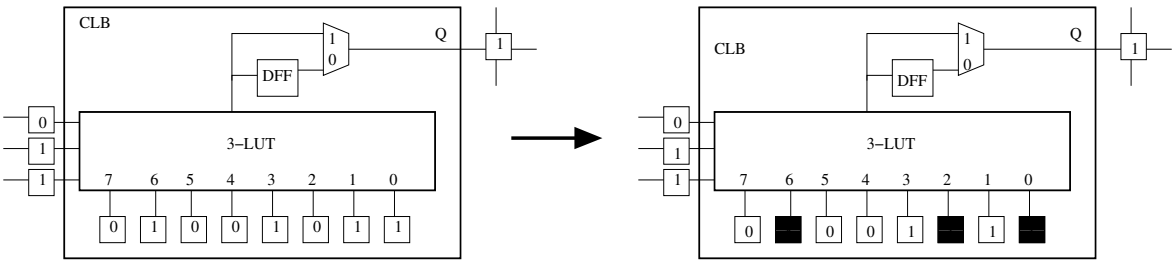
1. une panne *stuck-on* peut causer le déroutage d'un signal dans un SB (figure 4.9 (c))
2. une panne *stuck-off* peut causer une disparition d'un signal dans un SB (figure 4.9 (b))



(a) Routage correcte (b) Signal manqué (c) Signal dérouté

FIGURE 4.9: Pannes de routage dans les SBs dues aux pannes stuck-on et stuck-off.

Le principe de notre technique de tolérance aux pannes des FPGA-SRAMs est basé sur le fait de reconfigurer la trame de configuration contenant les cellules SRAMs défectueuses responsables du routage de manière à remplacer ces cellules par d'autres sans défauts permet de tolérer les pannes de routage.



(a) Une 3–LUT implémentant la fonction Q (b) Cellules SRAM défectueuses dans une 3–LUT

FIGURE 4.10: LUT partiellement défectueuse.

Remarque 4.3.

1. *Dans cette technique, la granularité de la tolérance aux pannes des FPGA-SRAMs est réduite à une cellule SRAM.*
2. *La détection et le diagnostic des cellules SRAM de la mémoire CMA sont détaillés dans le chapitre 6.*

Les cellules SRAM responsables de la configuration des LUTs causent des pannes dans ces LUTs suite à une ou plusieurs cellules SRAM défectueuses. Les techniques de tolérance aux pannes *classiques* (chapitre 3) remplacent tout le CLB par un CLB de rechange. Une reconfiguration de la trame défectueuse de manière à ce que cette trame remplace les

cellules SRAM défectueuses par des cellules de rechange permet de réutiliser le même CLB. Il est évident que la technique ne change pas le timing du circuit vu qu'elle interviendra à l'intérieur de chaque CLB.

4.2.2 Principe de la technique

La trame de configuration est la plus petite unité adressable dans la mémoire de configuration d'un FPGA de technologie SRAM. Dans le cas des FPGAs Virtex-5, une trame de configuration contient 1312 bits de configuration (1312 cellules SRAM) regroupés en 41 mots de 32 bits chacun. L'élément clé de notre approche est la capacité de changer l'adresse d'un mot dans une trame de configuration pour pouvoir remplacer les cellules défectueuses par des cellules SRAM opérationnelles.

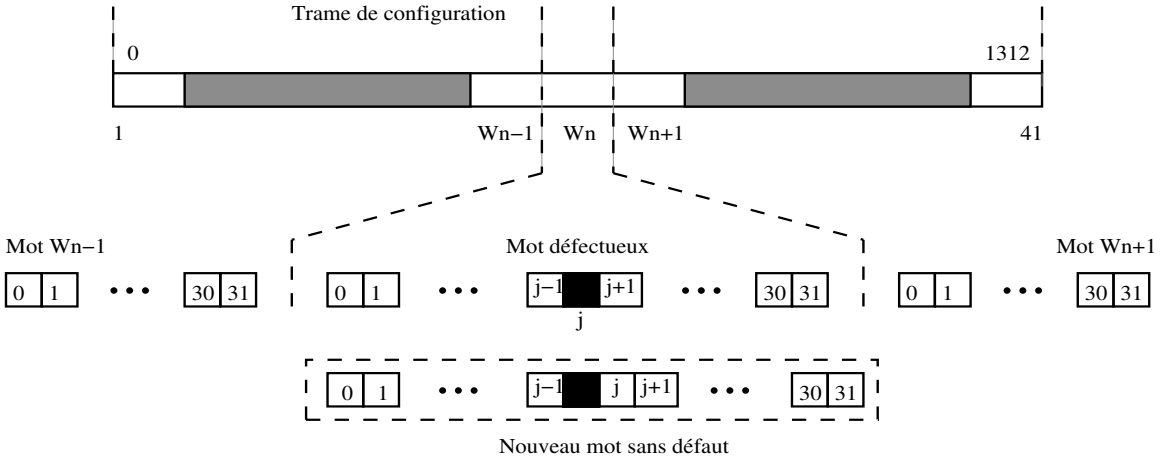
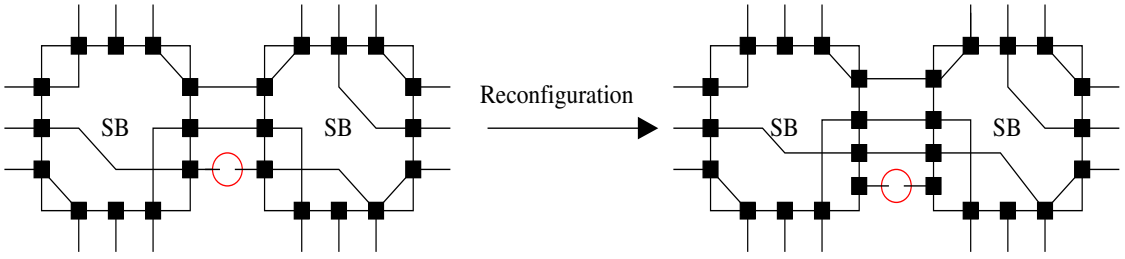


FIGURE 4.11: Reconfiguration d'une trame de configuration pour remplacer les cellules SRAM défectueuses.

La figure 4.11 illustre le réarrangement d'une trame de configuration pour remplacer les cellules SRAM défectueuses. La technique permet aussi la tolérance d'autres types de panne, tels qu'une rupture dans une ligne de connexion en agissant sur les cellules SRAM qui configurent les deux points CIPs délimitant la ligne comme illustré dans la figure 4.12.



(a) Panne dans les lignes de connexion (b) Remplacement de la ligne défectueuse

FIGURE 4.12: Remplacement des CIPs délimitant des lignes défectueuses.

4.2.3 Analyse de la fiabilité

Soit $R_{SRAM\ Cell}(t)$ la fiabilité d'une cellule SRAM, la probabilité qu'une cellule SRAM tombe en panne est $(1 - R_{SRAM\ Cell}(t))$. La fiabilité d'un mot w_i d'une trame de configuration est $R_{w_i}(t) = (R_{SRAM\ Cell}(t))^{32}$. Celle d'une trame de configuration sans technique de tolérance aux pannes est $R_{init} = (R_{w_i}(t))^{41}$.
Pour la technique proposée, la fiabilité est donnée donc par la formule suivante :

$$R_{w_i}(t) = \sum_{i=0}^{m} R_{SRAM\ Cell}(t)^{l-i}(1 - R_{SRAM\ Cell}(t))^{i} \tag{4.5}$$

où m est le nombre de cellules SRAM de rechange allouées dans la trame de configuration et $l = 32$ bits.
La fiabilité d'une trame de configuration est donc : $R_{Trame}(t) = (R_{w_i}(t))^{41}$.

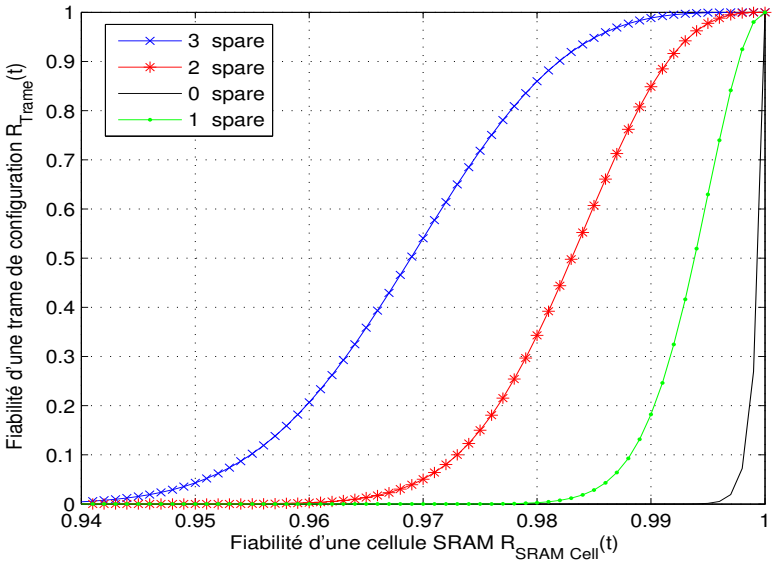


FIGURE 4.13: Fiabilité de trames de configuration avec et sans Cellules SRAM de rechange.

La figure 4.13 montre la fiabilité d'une trame de configuration avec la technique proposée versus la fiabilité d'une trame de configuration sans technique de tolérance aux pannes. Cette figure 4.13 montre la fiabilité d'une trame de configuration lorsqu'on réserve une cellule SRAM comme *spare* par mot. Le coût matériel est alors de 3.12%. Lorsqu'on réserve deux cellules SRAM par mot, ce coût est de 6.25%. Enfin, lorsqu'on réserve trois cellules SRAM par mot, il est de 9.37%.

4.2.4 Discussion

Le coût matériel de la technique proposée ne prend en compte que les cellules SRAM réservées par la technique. Nous n'avons pas une vision sur le nombre de multiplexeurs à ajouter pour permettre la flexibilité d'adressage des trames de configuration.

Le modèle de fiabilité ne représente pas la flexibilité de la technique proposée. Dans le cas de plusieurs cellules SRAM défectueuses dans le même mot, la technique utilise les

cellules SRAM allouées dans les autres mots de la trame de configuration et effectue un nouvel adressage pour remplacer les cellules SRAM défectueuses.

Conclusions du chapitre :

Dans ce chapitre, nous avons proposé deux techniques pour la tolérance aux pannes permanentes des FPGAs de technologie SRAM. Ces techniques prennent en compte trois paramètres essentiels pour la fiabilité des FPGA-SRAMs :

1. la synthèse du FPGA-SRAM
2. une conception tolérante aux pannes
3. une amélioration du rendement du FPGA-SRAM

La technique MST partitionne un design implémenté dans un FPGA-SRAM en unités MSUs. La reconfiguration partielle de chaque MSU permet la tolérance à plusieurs pannes. Nous avons réalisé une série d'implémentations de circuits de référence (benchmarks) ITC'99 pour évaluer le coût matériel de la technique MST. Par ailleurs, cette technique est capable d'atténuer les SEUs dans les noeuds CLB-Ms grâce à l'architecture de ces derniers.

La deuxième technique consiste à remplacer les cellules SRAM défectueuses dans les trames de configuration par des cellules SRAM de rechange. D'un point de vue applicatif, cette approche ouvre la voie à la conception de méthodes de tolérance aux pannes au niveau matériel plus rigoureuses et plus générales. La détection et la localisation des cellules SRAM défectueuses dans les trames de configuration du FPGA-SRAM constituent le sujet du chapitre 6.

Techniques d'atténuation des SEUs dans les FPGAs à base de mémoire SRAM

5

« Une éruption solaire . . . Elle se produit périodiquement à la surface de la photosphère et projette au travers de la chromosphère des jets de matière ionisée qui se perdent dans la couronne à des centaines de milliers de kilomètres d'altitude . . . Durant le stade précurseur, l'énergie commence à être libérée sous la forme de rayons X. Puis les électrons, protons et ions accélèrent jusqu'à approcher la vitesse de la lumière lors du stade impulsif.»

Wikipedia

Sommaire

Introduction au chapitre :

Le phénomène des effets transitoires (SEUs), bien connu et documenté, affecte considérablement les circuits électroniques. Les SEUs sont causés soit par les neutrons atmosphériques soit par les particules chargés. Les SEUs pourraient causer des erreurs sur un ou plusieurs bits dans les composants électroniques tels que les ASICs, les processeurs, les mémoires et les FPGA-SRAMs. Dans ce cinquième chapitre, nous introduisons brièvement l'effet des particules chargées sur les systèmes reconfigurables, notamment les FPGAs de technologie SRAM. Ensuite, nous décrivons les techniques et méthodes de détection et d'atténuation des SEUs dans les FPGA-SRAMs. De plus, nous discutons une stratégie d'atténuation des SEUs dans les FPGAs basée sur les bits critiques d'un design implémenté dans un FPGA-SRAM. Finalement, nous terminons ce chapitre par une technique auto-test permettant d'atténuer les SEUs.

5.1 Exploitation des FPGA-SRAMs dans l'espace

Les FPGA-SRAMs offrent de nombreux avantages tels que la reconfiguration partielle et dynamique (PDR), la reconfiguration en temps-réel et un haut débit dans les applications de traitement de données intensives, souvent utilisées dans les systèmes spatiaux ou dans les expériences de physique de hautes énergies (HEP). Ces applications sont forcées d'exploiter en même temps les circuits ASICs, les processeurs et les FPGA-SRAMs dans leurs détecteurs placés au voisinage des accélérateurs de particules comme le grand collisionneur de hadrons (Large Hardrons Collider, LHC). Les FPGA-SRAMs commerciaux (COTS) [1] sont donc inadéquats sans techniques de tolérance aux pannes transitoires dans cet environnement d'exploitation.

Les FPGA-SRAMs sont adaptés aux missions à distance car ils permettent d'apporter des modifications sur la conception FPGA en orbite et ainsi de réduire le coût de la mission en corrigeant les erreurs ou encore d'améliorer les performances du système après lancement. Contrairement aux ASICs, la propriété de reconfiguration des FPGA-SRAMs conduit à des solutions prometteuses telles que la facilité d'utilisation, une mise rapide sur le marché, des dépenses faibles d'ingénierie ainsi que la vérification de la conception sur puce. En contrepartie, les FPGA-SRAMs sont sensibles aux particules chargés provoquant des SEUs [2].

Dans un FPGA-SRAM, la mémoire de configuration (CMA) comprend généralement des millions de cellules SRAM constituant la couche de configuration du FPGA-SRAM (partitionnement hypothétique, Figure 2.7 Chapitre 2). Afin de fonctionner de manière fiable dans l'espace par exemple, la plupart des systèmes qui utilisent les FPGA-SRAMs atténuent les effets des SEUs par certaines formes de redondance. La technique la plus couramment

1. COTS : désigne les composants commerciaux, appelés également composants sur étagère.
2. Single Event Upset : type de panne lié à un impact de particule chargée dans un FPGA-SRAM et se traduisant par l'inversion logique d'un bit d'information [100].

utilisée au sein des FPGA-SRAMs est la redondance modulaire triple (TMR) [6][101]. La technique TMR[3] a un coût matériel de plus de 3 fois l'espace et la puissance initialement consommés [102].

Les particules chargés existant dans l'environnement spatial peuvent avoir des effets dramatiques sur l'électronique des engins spatiaux. Les effets singuliers (SEE) sont donc la principale préoccupation dans l'espace étant donnés les risques importants qu'ils représentent, comme la perte d'information ou la défaillance fonctionnelle [103]. Quand le silicium d'un circuit électronique est heurté par des particules chargés, ces derniers lui transfèrent suffisamment d'énergie pour provoquer un SEU dans le système [103][104].

L'évolution permanente de la technologie d'intégration des VLSIs a conduit à des architectures de plus en plus complexes, avec une grande quantité de mémoire embarquée [67]. Le procédé de fabrication est maintenant arrivé à un point où il sera impossible d'intégrer des FPGA-SRAMs *immunisés* contre les SEUs [105]. En effet, la technologie de fabrication des composants semi-conducteurs est en évolution continue en termes de rétrécissement de la géométrie des transistors, de puissance d'alimentation, de fréquence .... La protection des FPGA-SRAMs contre les SEUs est donc devenue de plus en plus impérative. En effet, les expériences présentées dans [106] indiquent que les neutrons présents dans l'atmosphère sont capables de produire des SEUs. Sur la base de la définition de la tolérance aux pannes, l'objectif est de maintenir l'exploitation des FPGA-SRAMs correctement en dépit de l'existence des SEUs.

D'un autre côté, la technologie anti-fusible est une technique de durcissement des FPGAs utilisée pour de ces circuits contre les SEUs [107]. Cependant, les FPGAs récents de technologie anti-fusible sont coûteux et ne laissent pas de place d'aporter des améliorations de la conception FPGA puisqu'il s'agit d'une technologie OTP. Les avantages offerts par les FPGA-SRAMs poussent les ingénieurs à concevoir des méthodes fiables d'atténuation des SEUs.

5.2 Effets des particules chargés sur les FPGA-SRAMs

La densité d'intégration des circuits FPGA-SRAMs continue de croitre. Par ailleurs, l'exploitation des FPGA-SRAMs dans l'environnement spatial et plus récemment au niveau de la mer peut être perturbée par des particules chargées qui génèrent également des pannes transitoires (SEUs) [108]. Les pannes transitoires provoquées par les particules chargés sont une préoccupation majeure, elles doivent être tolérées afin de garantir un niveau élevé de fiabilité des FPGA-SRAMs.

Le milieu spatial se compose de plusieurs particules produites par les activités solaires. L'interaction de ces particules avec le silicium d'un circuit produit l'ionisation qui génère un dépôt de charges qui peuvent être modélisées par des impulsions de courant transitoire [109]. Ce courant peut être interprété comme un signal dans le circuit provoquant un SEU dans la logique combinatoire, séquentielle, SBs ou dans les CIPs d'un FPGA à base de mémoire SRAM. La figure 5.1 - adaptée de [109] - montre les endroits vulnérables par les SEUs dans une LUT. Les SEUs peuvent produire l'une des trois pannes suivantes [109] :

3. TMR : la redondance modulaire triple (Triple modular redundancy) est la conception d'un système par multiplication de ses modules par trois et l'ajout d'un circuit de vote de majorité réalisant un vote majoritaire pour masquer l'effet d'un SEU sur l'un de ses modules.

1. un changement dans une bascule flip-flop (DFF)
2. un changement dans la mémoire de configuration qui provoque le changement d'une source de routage ou d'une bascule flip-flop,
3. un changement dans la demi-bascule qui a causé le maintien de la bascule flip-flop en mode *reset* ou désactivé l'horloge.

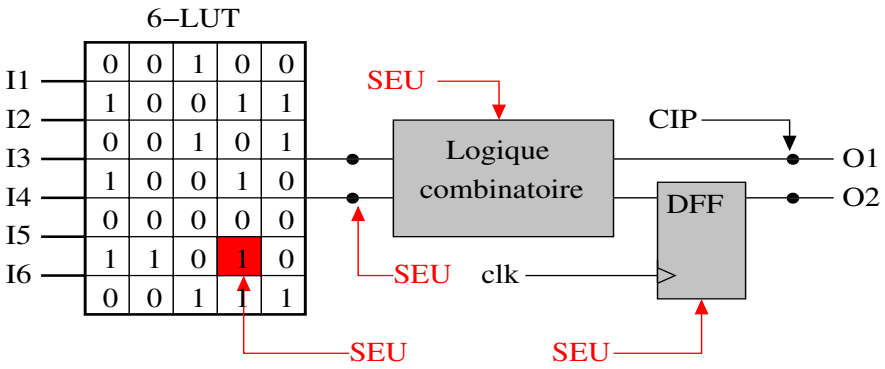


FIGURE 5.1: Les SEUs perturbent les logiques séquentielle et combinatoire ainsi que les ressources de routage programmables dans un FPGA à base de mémoire SRAM.

Considérons par exemple la figure 5.2 - adaptée de [110] - où les cellules mémoires SRAM C1, SRAM C2 et SRAM C4 contiennent la valeur logique 1 et la cellule mémoire SRAM C3 contient la valeur logique 0. Les points CIP 1 et CIP 2 sont activés, connectant les lignes de connexion LC0, LC1 et LC2. La cellule SRAM C1 est aussi activée, connectant LC3 et LC4. Le point CIP 3 est désactivé.

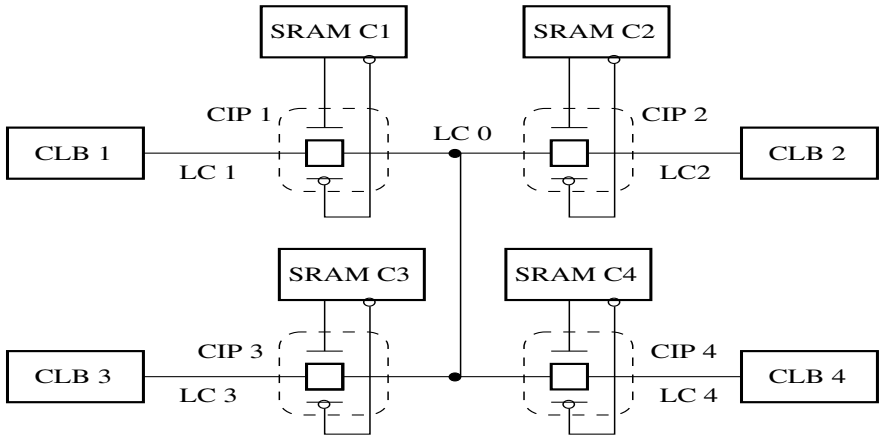


FIGURE 5.2: Une partie d'un FPGA à base de mémoire SRAM où les SEUs peuvent alterner le contenu des cellules SRAM.

Supposons qu'un SEU apparaisse dans la cellule SRAM C1 et change sa valeur logique en 0, le point CIP 1 est désactivé donc la ligne LC1 sera isolée de la ligne LC0. Supposons aussi que le CLB 1 achemine un signal à travers la ligne LC1 et le CLB 3 achemine un signal à travers la ligne LC3, si le CLB 1 est connecté au CLB 2 à travers LC0, cette liaison n'existera pas et la fonction sera erronée [110].

5.3 Méthodes de détection et de correction des SEUs au niveau configuration dans un FPGA-SRAM

Durant les dix dernières années, les chercheurs ont testé les FPGA-SRAMs de la famille Virtex de Xilinx pour la sensibilité aux rayonnements [111]. En plus d'atténuer les SEUs, il s'agit d'identifier la manière dont les SEUs agissent sur un FPGA-SRAM parmi les trois modes cités dans le paragraphe 5.2.

5.3.1 Atténuation des SEUs par la technique *Scrubbing*

Dans [112], on trouve une comparaison entre la technique *scrubbing* [4] et une autre technique d'atténuation des SEUs pour les FPGA-SRAMs de Xilinx. Comme la mémoire de configuration n'est pas durcie contre les SEUs, celle-ci a une *section efficace* [5] qui favorise les SEUs. Le but derrière les techniques d'atténuation des SEUs est la réduction de la durée des pannes provoquées par les SEUs ainsi que la prévention de l'accumulation des SEUs dans cette mémoire.

Les techniques d'atténuation des SEUs dans la mémoire de configuration d'un FPGA-SRAM par reconfiguration peuvent être classées selon deux catégories [112] :

- la première catégorie consiste à reconfigurer complètement le FPGA-SRAM. Cette catégorie de techniques exige un arrêt complet du fonctionnement du système. Le *bitstream* reconfigure une nouvelle fois le FPGA-SRAM afin de ramener le système à son état de fonctionnement.
- la deuxième catégorie utilise la technique *Scrubbing* qui est le processus de reconfiguration de quelques portions de la mémoire de configuration sans perturber l'opération ou l'ensemble des tâches exécutées par le système FPGA-SRAM [113]. L'aspect statique dominant la mémoire de configuration permet l'écriture dans cette mémoire sans perturber son fonctionnement. Par conséquent, la technique *Scrubbing* est transparente en ce qui concerne le fonctionnement opérationnel (le système ne s'arrête pas durant le *scrubbing*). La mémoire de configuration contient aussi des bits de configuration dynamiques comme les registres de décalage de 16-bits (SRL16s), les RAM distribuées ou les BRAMs. Ces bits de configuration peuvent être modifiés par le *scrubbing*, ce qui perturbe le fonctionnement opérationnel.

L'étude présentée en [112] se focalise sur les modes de défaillance relatives à l'accumulation des SEUs dans la mémoire de configuration en utilisant la deuxième catégorie comme techniques d'atténuation des SEUs. Une comparaison des performances d'implémentation de la technique *scrubbing* développée par Xilinx (Xilinx IP core scrubber) et d'une autre déve-

4. Scrubbing : Une méthode de correction des SEUs qui consiste à configurer à nouveau une trame de configuration d'un FPGA-SRAM à un instant donné. La technique *scrubbing* est moins couteuse comme technique d'atténuations des SEUs, mais ne signifie pas que la logique de configuration est susceptible d'être en mode écriture pour un grand pourcentage de temps [36][112].

5. Deux paramètres sont utilisés pour caractériser la sensibilité des circuits intégrés face aux particules chargées : la section efficace, notée σ, et le seuil de sensibilité (ou charge critique), noté LET_{th} (Linear Energy Transfer). Pour générer des événements singuliers dans un circuit intégré, c'est ce LET minimum qu'une particule doit avoir. Cette valeur dépend notamment de la technologie de fabrication du circuit. La section efficace est le nombre de perturbations divisé par la fluence reçue par le circuit. Son unité est cm^2/composant ou bien le cm^2/bit [104].

loppée par la NASA (NASA Radiation effects and analysis group REAG external scrubber) est faite. La première utilise la technique *Readback* [6] en conjonction avec la technique SEC-DED ECC pour la détection et la correction des pannes. La deuxième technique consiste à écrire le *bitstream* correct (golden configuration) périodiquement dans la mémoire de configuration. La technique REAG de la NASA a amélioré les performances du système par rapport à la technique Xilinx IP-core de Xilinx.

La technique REAG se montre inefficace pour tolérer les pannes transitoires dans les bits de configuration dynamiques ni les pannes dues aux SEFIs (Single event functionnal interrupt). La première technique (Xilinx IP-core) reste inefficace devant les MBUs (Multiple Bits Upsets).

5.3.2 Atténuation des SEUs dans les mémoires BRAMs

Les SEUs sont difficiles à détecter et à corriger lorsqu'ils affectent les éléments mémoire présents dans les FPGA-SRAMs et qui sont utilisées pour l'implémentation des machines à états finis (FSMs). L'idée centrale de l'approche proposée dans [115] est de placer une FSM dans une BRAM libre dans un FPGA-SRAM (pour la détection des SEUs en utilisant le bit de parité) et de corriger ensuite ces pannes par une réécriture du contenu de ces mémoires. L'implémentation des FSMs dans des BRAMs présente les avantages suivants [115] :

- Temps fixe quelque soit la complexité du FSM implémentée,
- Facilité de développement et de débogage par rapport à la mise en œuvre traditionnelle,
- Faible consommation en puissance comparée à l'implémentation de larges FSMs basée sur l'approche utilisant des DFF,
- Changement rapide et facile dans la fonctionnalité du FSM par changement du contenu du BRAM,
- Coût matériel minimal.

Deux techniques basées sur le bit de parité, la redondance modulaire et le *scrubbing* des BRAMs, permettent la détection et la correction des SEUs dans ces mémoires. La première technique combine la redondance modulaire double (DMR) et l'inclusion du bit de parité pour chaque mot pour la détection des MBUs. La seconde technique utilise le bit de parité de l'entrée du FSM et un *scrubbing* extérieur lors d'une éventuelle détection d'un SEU. La deuxième technique exige que la BRAM soit dual-port : chaque port doit avoir des adresses, horloge, données, set/reset séparés [115].

5.4 Méthodes de détection et de correction des SEUs au niveau matériel du FPGA-SRAM

5.4.1 Atténuation des SEUs par la technique TMR

La technique TMR [116]- [119] est une technique de redondance qui permet de masquer les pannes. Dans la technique TMR, un module est reproduit trois fois puis la sortie est extraite à partir d'un circuit de vote majoritaire. Lorsqu'un SEU apparaît, le circuit de

6. Readback : c'est une opération post-configuration de lecture des registres et de la mémoire de configuration [114].

vote majoritaire ignore la valeur erronée du module recevant le SEU et accepte la valeur correcte provenant des deux autres modules qui n'ont pas été affectés par des SEUs. Si deux modules reçoivent simultanément des SEUs, alors il n'est pas garanti d'obtenir en sortie la valeur correcte. Le système de redondance est considéré comme tolérant aux SEUs à condition que le circuit de vote majoritaire soit *immunisé* contre ces derniers. Un circuit de vote majoritaire peut être conçu avec des buffers à trois états [7].

Le principal inconvénient de la technique TMR est son coût matériel excessif. Cependant, elle peut atténuer les SEUs dans un module et a donc été utilisée dans les familles FPGA-SRAMs Virtex de Xilinx [120]. Quand le circuit TMR est placé sur un FPGA-SRAM, il est physiquement mélangé. Dans un FPGA-SRAM, la nature programmable des ressources de routage peut favoriser des pannes qui sont plus difficiles à détecter [120]. Comme le circuit TMR est physiquement mélangé dans un FPGA, les nœuds intérieurs (un nœud correspond à une interconnexion entre les éléments d'un circuit) dans deux modules différents peuvent être facilement séparés par un seul CIP inactivé. Si un SEU permet par inadvertance un tel CIP, les nœuds internes des deux modules seront court-circuités. Deux des trois modules fournissent alors des données suspectes sur le circuit de vote majoritaire. De même, un SEU qui change la valeur sauvegardée dans un CIP peut court-circuiter deux sorties. Dans ce cas, deux des trois entrées du circuit du vote majoritaire peuvent être incorrectes.

Pour répondre à ce problème, on voit dans [121] que la méthode QMR propose que les circuits quintuple modular redundancy soient mélangés dans le FPGA-SRAM. Ainsi un SEU, qui par inadvertance court-circuite deux lignes d'interconnexion, peut rendre invalide seulement deux sur cinq signaux de sortie du module. Le circuit de vote majoritaire fournit toujours la valeur correcte. Cependant, cette solution apportée par la méthode QMR souffre d'un coût excessif en ressources.

5.4.2 Effets des SEUs sur les ressources de routage

Le durcissement des designs est une des techniques employées pour l'atténuation des SEUs. Il comprend la mise en place d'une redondance matérielle et/ou logicielle. Le durcissement contre les SEUs pour les bascules latch et les cellules SRAM est obtenu par la modification de leur structure de base, en ajoutant des transistors supplémentaires pour augmenter la capacité de certains nœuds et ainsi obtenir une immunité des transistors contre les SEUs. Le latch RHBD [122], le latch *C-element* [123][124], la cellule de blocage d'information durcie [125], la cellule DICE [126, 127], la cellule SRAM-tct [128] et la cellule HIT [129] sont quelques exemples de la tolérance aux SEUs présents dans la littérature. Ces méthodes, bien que efficaces, ne peuvent malheureusement pas être facilement appliquées aux FPGA-SRAMs à base de mémoire SRAM [110] étant donné que les FPGA-SRAMs sont des PLDs COTS préfabriqués. Le cycle de conception dans son ensemble devrait être modifié, ce qui est prohibitif. Les codes correcteurs d'erreur comme les codes SEC-DED [130], les codes SEC-DED-DAEC [131], les codes SEC-DAED-SDAEC [132] et les codes de Hamming [133] peuvent également être utilisés pour protéger les FPGA-SRAMs contre les SEUs. Cependant, ces codes correcteurs d'erreur ne peuvent corriger qu'un nombre limité de bits erronés par effet des SEUs.

La méthode présentée dans [110] propose une solution aux courts-circuits des lignes d'interconnexion dans les FPGA-SRAMs affectées par les SEUs. La figure 5.3 - adaptée de [110] - représente le schéma proposé pour le routage d'un circuit TMR. La méthode proposée

attribue deux nœuds dans les différents modules du circuit TMR à deux lignes d'interconnexions séparées par deux CIPs.

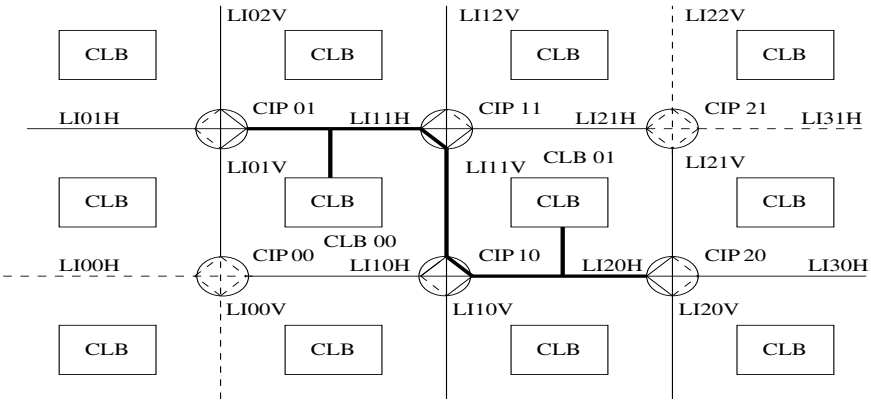


FIGURE 5.3: Schéma représentant la méthode proposée pour atténuer les SEUs dans les CIPs.

Si la cellule mémoire qui contrôle un des points CIPs change de contenu par l'impact d'un SEU, les deux lignes d'interconnexion ne peuvent pas être connectées (risque de court-circuit entre les deux lignes). Autrement dit, deux nœuds dans deux modules de TMR différents ne peuvent pas être court-circuités par un SEU [110].

5.5 Stratégie d'atténuation des SEUs

Les informations sur les bits critiques de configuration (Critical Bits Information) sont une option additionnelle pour traiter rigoureusement les SEUs [134].

5.5.1 Information sur les bits critiques de configuration

La détection et la correction des SEUs sont seulement focalisées sur les cellules mémoires qui sont associées avec les bits de configuration qui définissent une conception FPGA-SRAM [135]. Il s'agit des bits qui doivent rester statiques pendant le fonctionnement du FPGA-SRAM. Si des cellules SRAM de configuration définissent une partie du design dans un FPGA-SRAM, alors ces cellules peuvent changer d'état à cause d'un SEUs. Donc, des cellules SRAM de configuration qui définissent une partie du design pourraient être considérés comme des bits potentiellement critiques (*potentially critical bits*). Ce que l'on cherche à savoir, c'est quels sont les bits qui entrent dans cette catégorie et combien sont-ils [134]-[135].

La première approche suppose que chaque bit dans la configuration a le potentiel d'être critique [135]. Toutefois, il s'agit juste d'une hypothèse facile pour les calculs. Le nombre de trames de configuration dans la mémoire de configuration d'un FPGA-SRAM est fixe. Cependant, tous les bits de ces trames de configuration n'occupent pas une mémoire SRAM physique [135]. Autrement dit, il y a des "*trous*" dans la mémoire de configuration du FPGA-SRAM. Chaque trame de configuration dans la famille Virtex-5 de Xilinx contient 16 bits inutilisables. Par conséquent, ces bits ne peuvent pas être des bits critiques pour un design. Puisqu'il s'agit de bits inutilisables, ce sont des bits qui n'existent pas, donc ils ne peuvent

pas changer d'état par l'effet d'un SEU. Ainsi, ces bits devraient être déduits du nombre initial de bits [135].

À ce stade, nous avons besoin de déterminer quels sont les bits qui ont le potentiel d'être des bits critiques pour un design et ceux qui ne l'ont pas. Il y a par exemple les 12 bits ECC (Error Correction Code) qui servent à identifier les cellules SRAM qui ont changé d'état par un SEU [35][135]. Ces bits ne définissent aucune partie du design donc le changement d'état d'un de ces bits ne peut pas influencer sur le fonctionnement du design. Ainsi, $\sim$ 0.9% des bits d'une trame de configuration sont définitivement considérés comme des bits non-critiques [135].

Bien que nous n'avons pas une façon directe de voir ce que chaque bit de configuration contrôle dans un FPGA-SRAM, il parait évident que chaque ressource configurable soit définie par un ou plusieurs bits. Les points CIPs ont un grand potentiel de connections et peuvent être considérés comme des *'switches'* qui sont activés ou désactivés par un bit de configuration associé. Si une ligne de connexion est utilisée dans un design, alors il est raisonnable de supposer que les bits contrôlant cette connexion sont des bits critiques pour le design implémenté dans le FPGA-SRAM [135]. De même, les connections qui ne sont pas utilisées par le design contiennent des *switches* désactivés mais ils n'affecteront pas le design s'ils sont activés. Pour cette raison, les bits contrôlant des CIPs qui ne font pas partie du design ne sont pas des *bits critiques* [135].

Remarque 5.1.
La société Xilinx ne fournit aucun document sur la fonctionnalité des bits de configuration dans un FPGA. En effet, il n'y a pas des informations sur le rôles et la distribution des bits dans le fichier Bitstream.

5.5.2 Étude statistique

Les portes logiques d'un design sont généralement implémentées dans un FPGA Virtex-5 à l'aide des LUTs de type 6-LUT. Comme dans ces circuits une 6-LUT a 6 entrées, il y a 64 cellules de configuration pour définir tous les résultats possibles d'une fonction logique quelconque.

Mais quand moins de 6 entrées d'une 6-LUT sont connectées dans un design, plusieurs cellules de configuration parmi les 64 ne seront pas utilisées dans le design. Par conséquent, même lorsque une 6-LUT est utilisée dans un design, certains bits seront critiques et d'autres ne le seront pas. La figure 5.4 (a) prise du *FPGA Editor ISE 11.5* montre le placement et le routage du design *b19.vhd*. Le tableau 5.2 montre une utilisation de $\sim$ 77% des *Slices* du Virtex-5 XC5VFX70T. Une première inspection confirme que le design semble être assez complet mais en regardant de plus près la figure 5.4 (b) ainsi que la figure A.1 de l'annexe A, on voit que des ressources configurables restent inutilisées.

En effet, nous avons effectué une implémentation de 27 circuits de référence de ITC'99 de *Sun Microsystems, Inc*. Les résultats sont montrés dans le tableau 5.2. L'étude statistique du nombre d'entrées des 6-LUTs, utilisées dans l'implémentation des circuits de référence ITC'99 dans le FPGA Virtex-5 XC5VFXT70T (figure 5.5), montre que seulement 52.59% des 6-LUTs utilisent complètement leurs 6 entrées et que 47.41% des 6-LUTs utilisent moins de 6 entrées. Ces résultats montrent qu'une grande partie des 6-LUTs qui font partie de l'implémentation d'un design dans un FPGA-SRAM contiennent des bits qualifiés de non-

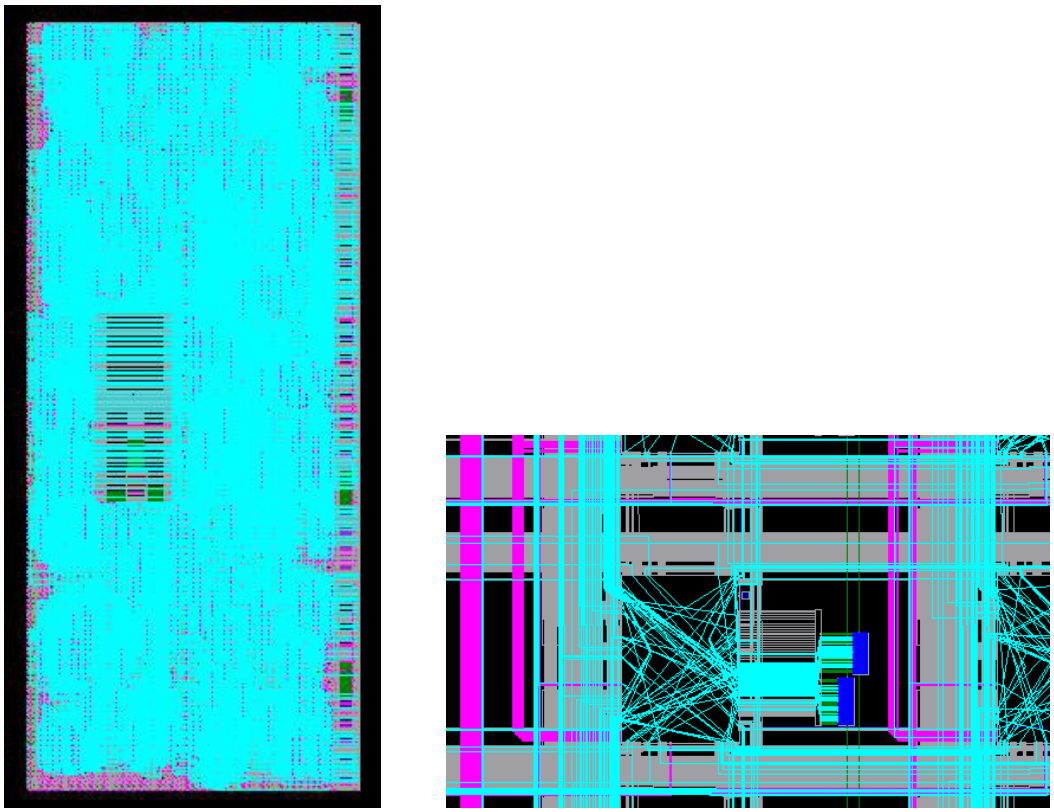
	LUT						DFF/Latches			IO Buffers	
Design	**LUT1**	**LUT2**	**LUT3**	**LUT4**	**LUT5**	**LUT6**	**FDC**	**FDP**	**FDCE**	**IBUF**	**OBUF**
b01	0	0	0	8	1	1	9	1	0	3	2
b02	0	1	0	3	0	0	3	1	0	2	1
b03	0	0	2	4	16	16	22	1	12	5	4
b04	0	14	2	40	27	31	66	1	0	12	8
b05	0	5	13	11	59	96	42	1	0	2	36
b06	0	1	2	4	1	0	7	1	0	3	6
b07	0	22	5	19	14	39	38	1	1	2	8
b08	0	2	1	1	7	12	8	1	12	10	4
b09	0	0	2	10	29	8	27	1	0	2	1
b10	0	4	7	2	6	19	17	1	1	12	6
b11	0	11	11	14	19	45	35	1	0	8	6
b12	0	26	13	25	30	161	69	2	70	6	6
b13	0	9	21	22	3	23	31	4	27	11	10
b14	27	31	85	180	309	529	55	1	161	33	54
b14_1	27	31	79	226	348	524	55	1	161	33	54
b15	205	46	88	157	212	1004	326	1	2	37	70
b15_1	236	19	94	100	259	983	327	2	97	37	70
b17	614	148	447	414	709	3109	982	4	353	38	97
b17_1	707	61	465	327	912	2850	982	4	353	38	97
b18	1256	298	908	1442	1582	6105	2006	7	889	37	23
b18_1	1442	244	853	960	1462	6535	2006	7	889	37	23
b19	2514	641	1876	2874	3078	12510	4014	13	1778	21	30
b19_1	2514	651	1812	2872	3112	12574	4014	13	1778	21	30
b20	30	100	151	418	610	1111	110	2	322	33	22
b20_1	30	99	157	450	603	1141	110	2	322	33	22
b22	57	152	307	650	988	1589	165	3	451	33	22
b22_1	57	152	314	682	951	1625	165	3	451	33	22

TABLE 5.2: Ressources utilisées pour l'implémentation des designs de banc de tests ITC'99 dans le FPGA-SRAM Virtex-5 XC5VFX70T.

critiques pour ce design. Par exemple il y a 9.70% des 6-LUTs qui n'utilisent qu'une seule entrée. Dans ce cas, pour chaque 6-LUT, 62 bits sont qualifiés de non-critique pour le design.

Nous pouvons observer que beaucoup de ressources potentielles disponibles dans le FPGA-SRAM ne sont pas utilisées, et que par conséquent, les cellules SRAM de configuration associées à ces ressources ne sont pas critiques pour le design. Malgré cela, il n'est pas évident de déterminer les bits de configuration qui sont potentiellement critiques pour un design. Nous continuons donc dans la suite d'essayer de comprendre quels sont les bits qui sont critiques pour un design FPGA-SRAM.

Remarque 5.2.
Les outils Xilinx (critical bit map ISE 12.x) contiennent une description de chaque bit de configuration. C'est un raffinement considérable par rapport à l'analyse des ressources utilisées par un design. En principe, il suffit donc de compter les bits de configuration marqués potentiellement critiques et



(a) (b)

FIGURE 5.4: Utilisation des ressources configurable dans le FPGA Virtex-5 de Xilinx.

de multiplier le nombre total par le chiffre FIT/MO pour déterminer le facteur FIT pour un design.

5.5.3 Approche

Le circuit BIST montré dans la figure 5.6 inclut une petite mémoire pour les données et les adresses des trames de configuration qui contiennent les bits critiques pour un design implémenté dans un FPGA-SRAM (trames en bleu dans la figure 5.6). Le test est effectué en calculant la valeur de syndrome soit en utilisant la trame ECC-Frame, soit un processeur comme le *PowerPC 440* intégré dans le FPGA Virtex-5 XC5VFX70T [35]. Le test peut être effectué en temps réel. Une fois qu'un SEU change un bit dans les trames de configuration, le circuit BIST change la configuration de la trame par la configuration initiale sauvegardée dans la mémoire du circuit. Pour pouvoir écrire et lire dans les trames de configuration contenant le design, le circuit BIST utilise le composant ICAP. La méthode d'atténuation des SEUs suit les étapes suivantes :

1. génération de l'adresse de la trame de configuration participant dans la configuration du design
2. lecture de la trame de configuration par le ICAP puis calcul du syndrome par le circuit BIST

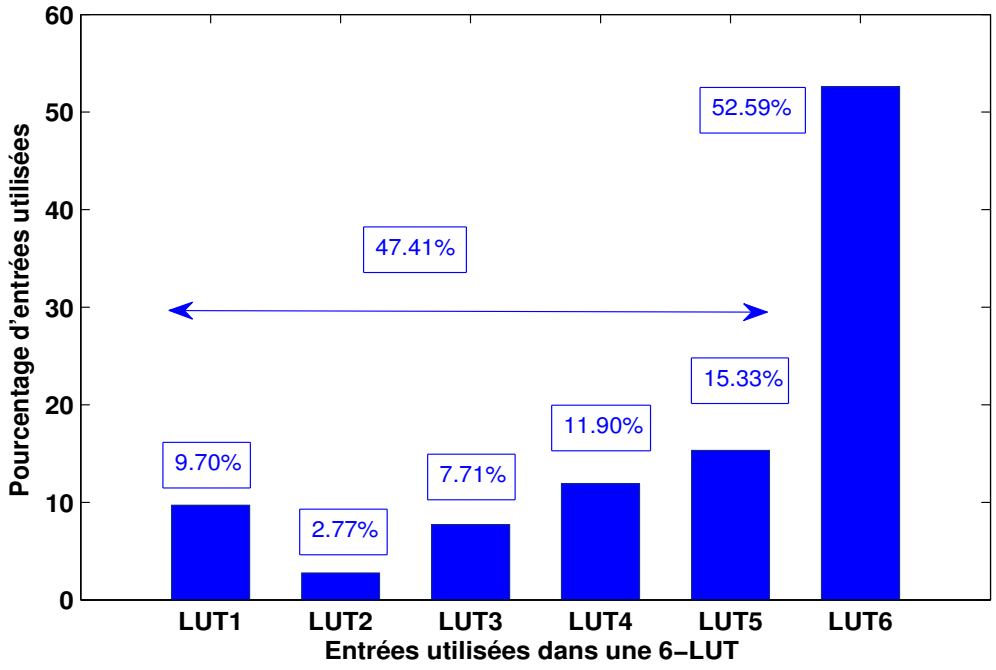


FIGURE 5.5: Étude statistique sur l'utilisation des entrées d'une 6-LUT appliquée sur 27 designs ITC'99 implémentés dans le FPGA Xilinx Virtex-5 XC5VFX70T.

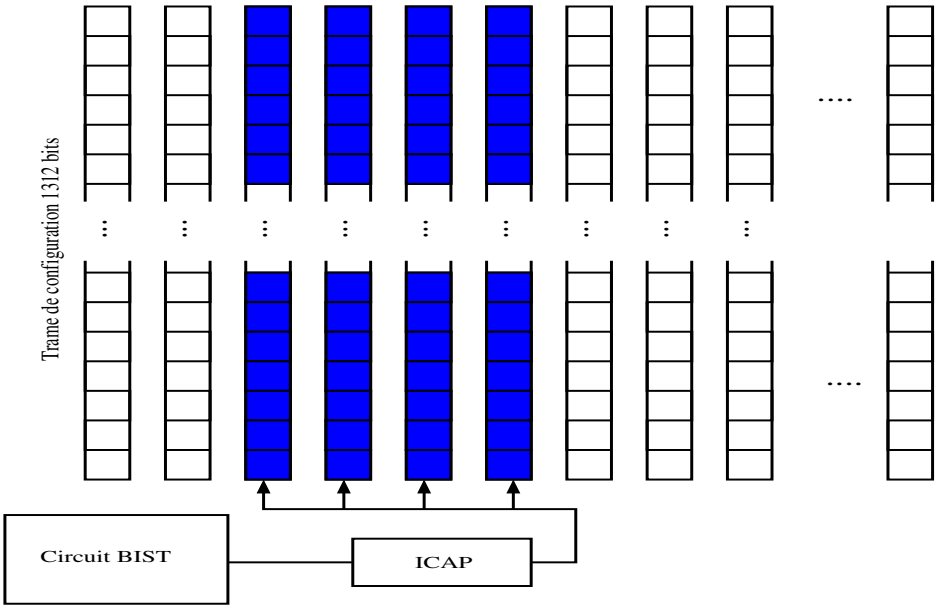


FIGURE 5.6: Schéma représentant l'approche pour l'atténuation des SEUs dans un FPGAs.

3. si une erreur est indiquée par le circuit BIST, le circuit fait le *readback* de la trame testée et remplace son contenu par le contenu initial sauvegardé dans la mémoire de circuit BIST
4. le test reprend avec l'adresse de la trame suivante

```
Read_Configuration_Frame (FRAME_ADDR){
        Write to command READ_CONFIG_MEM WCFG
        Write to Frame Address FAR FRAME_ADDR
        Read Frame Data Output FDRO 82 words
        for (i = 0; i < 41; i++){
        }
        for (i = 0; i < 41; i++){
        Read the frame used in the design;
        Compare the frame with the golden one;
        }
}
```

FIGURE 5.7: Lecture d'une trame de configuration via l'interface ICAP.

La procédure pour la lecture et l'écriture dans les trames de configuration dans le contexte de l'approche proposée est illustrée respectivement dans le pseudo-code de la figure 5.7 et dans le schéma de la figure 5.6.

Remarque 5.3.
Dans le FPGA-SRAM Virtex-5, la trame sélectionnée ne peut pas contenir de bits de configuration des LUT-RAM ou des flip-flop car ces bits sont masqués dans la logique de la trame Frame ECC durant l'opération Readback [35].

5.5.4 Discussion

Le seul inconvénient majeur de l'approche proposée est la détermination des trames de configuration qui implémentent le design dans le FPGA-SRAM. En effet, il est difficile d'identifier les trames de configuration dans le fichier *bitstream* ainsi que les ressources qu'il configure.

Conclusion du chapitre :

Peut-on vraiment éliminer tous les SEUs dans un FPGA-SRAM ? C'est une question à laquelle il est difficile de répondre mais une autre question dont nous devons nous préoccuper est comment minimiser l'impact de ces pannes transitoires sur le bon fonctionnement du FPGA-SRAM.
Finalement, il est intéressant de noter qu'il est incorrect de dire qu'une méthode d'atténuation des SEUs dans un FPGA-SRAM est fiable en testant cette méthode sur des circuits de référence qui utilisent un pourcentage médiocre des ressources du FPGA-SRAM. Il est important d'évaluer la fiabilité des FPGA-SRAMs en implémentant des designs de même dimension que les designs qui seront exploités.

6 Test, détection et diagnostic des pannes dans les FPGA-SRAMs

« . . . Advances in semiconductor memories have been very impressive. Their density . . . is ever increasing . . . algorithms with a test time of . . . order $O(n^2)$, where n is the number of bits in the memory chip, are no longer acceptable for testing current . . . multi-Mega-bit memory chips.»
– Ad J. van de Goor [68]

« La génération d'un programme de test efficace est une tâche fondamentale dévolue aujourd'hui au concepteur. Le choix des modèles de fautes (pannes) considérés et les stratégies de test mises en œuvre influencent profondément la qualité du produit livré aux clients. Par ailleurs, la part du coût de développement correspondant à la préparation des tests est en forte augmentation.»
– Régis LEVEUGLE [62]

Sommaire

Introduction au chapitre :

Plusieurs travaux de recherche ont montré que le test physique de toute mémoire, notamment la mémoire de configuration d'un FPGA-SRAM, n'est pas possible. Le seul mécanisme pour tester une mémoire est de comparer le comportement d'une mémoire défectueuse avec celui d'une mémoire sans défauts. Ceci requière la modélisation des pannes physiques en pannes logiques. Les tests doivent alors détecter et localiser les pannes physiques au sein d'une trame de configuration. La modélisation des pannes physiques en des pannes logiques rend les approches de test plus indépendantes, et par conséquent plus générales, de la technologique de fabrication.

Les tests *March* apparaissent comme un choix judicieux pour tester les trames de configuration d'un FPGA-SRAM vu leur linéarité et leur facilité d'implémentation. Or, les tests *March* sont conçus pour le test des mémoires orientées bits (BOM). Le fait de les appliquer sur des mémoires orientées mots (WOM) entraîne [137] :

1. Un temps excessif de test.
2. Un taux de couverture limité pour les pannes de couplage *intra-mot*.

Une méthode systématique de conversion des tests *March* BOM en tests *March* WOM, qui distinguent les pannes *inter-mots* des pannes *intra-mots*, apparaît nécessaire pour surmonter ces inconvénients.

6.1 Motivation

En dépit de l'utilisation croissante des FPGAs à base de mémoire SRAM, peu de travaux expérimentaux concernant le test du CMA ont été publiés. Le brevet d'invention [55] propose un circuit auto-test intégré (BIST) pour la détection des pannes de type "collage-à 0/1" dans la mémoire de configuration du FPGA-SRAM. Le circuit BIST proposé est composé d'une logique fixe comprenant une conception qui utilise le port ICAP qui permet la lecture et l'écriture dans les trames de configuration et qui permet aussi le *read-back* afin de valider chaque bit de configuration. Une mémoire non-volatile est incluse avec le circuit BIST pour la sauvegarde de l'état de la trame en cours d'évaluation. La mémoire non-volatile est ensuite accessible par le port ICAP à travers l'interface de configuration pour effectuer le diagnostic. Cette méthode n'est pas déterministe, elle n'est pas optimale et elle est consommatrice en temps de test. Cela implique qu'un seul test de mémoire est insuffisant et qu'il faut plutôt un ensemble de tests.

La méthode de diagnostic présentée dans [55] est basée sur l'écriture des 1's et des 0's dans la mémoire CMA de façon aléatoire, ce qui donne un aspect non-déterministe à la technique. En effet, la stratégie consiste à effectuer des comparaisons entre les données écrites dans les trames de configuration et les données lues à partir de ces trames. La technique proposée ne vise que les pannes de types "collage-à 0/1" bien qu'il existe divers types de pannes, ce qui se répercute sur le taux de couverture de la technique proposée.

La méthode proposée ne différencie pas les trames de données statiques des trames de données dynamiques.

Pour réaliser un taux élevé de couverture avec un coût de test industriellement acceptable, une modélisation précise de pannes et une conception des tests efficaces apparaissent importantes. Des travaux antérieurs ont largement étudié les tests de mémoire en utilisant des algorithmes d'une complexité allant de $\mathbf{O}(n^2)$ à $\mathbf{O}(n)$. Ce progrès *impressionnant* dans le test des mémoires est renforcé par les tests "*March*" [138]-[140]. Un test *March* est appliqué à chaque cellule SRAM de la mémoire avant de passer à la cellule suivante. En d'autres termes, si un modèle spécifique est appliqué à une cellule SRAM, il doit être appliqué à toutes les cellules de la mémoire. Ceci est fait soit dans un ordre croissant d'adressage de mémoire ($\Uparrow$), soit dans un ordre décroissant ($\Downarrow$) ou encore dans un ordre indifférent ($\Updownarrow$). Leur complexité $\boldsymbol{O}(n)$, leur régularité et leur symétrie font des tests *March* les plus appropriés pour le test des mémoires [2].

La méthode de conversion des tests *March* BOM en tests *March* WOM, a été proposée dans [137]. Motivé par le fait que les FPGA-SRAMs récents incluent des BRAMs, CLBs, des *hard cores*, des *soft cores*, des ICAPs, etc, nous avons adapté cette méthode pour le test et le diagnostic des trames de configuration du la mémoire CMA d'un FPGA-SRAM.

Remarque 6.1. *Dans ce chapitre tous les algorithmes et techniques du test March et les nominations des pannes ainsi que leurs modèles sont adaptés de l'article [137] afin de construire des tests pour les trames de configuration d'un FPGA-SRAM.*

6.2 Modélisation et détection des pannes dans la mémoire CMA

Plusieurs pannes dans les circuits intégrés, notamment dans les mémoires, sont causées par des particules indésirables appelées *spot defects (SDs)* . En fonction de leur conductivité, elles peuvent causer des connexions indésirables ou des déconnexions dans la mémoire. Leur impact sur la mémoire peut être classifié en trois groupes : *circuit ouvert, court-circuit et pont entre lignes*. Un *circuit ouvert* est une résistance excessive dans la ligne de connexion, un *court-circuit* est une connexion indésirable entre la tension V_{cc} et la masse tandis qu'un défaut de type *pont* est une connexion indésirable entre deux lignes [141].

6.2.1 La mémoire de configuration CMA

Les cellules SRAM de la mémoire CMA configurent les CLBs, IOBs, BRAMs et connectent les sous-circuits d'une conception FPGA en configurant les SBs. La mémoire CMA peut être visualisée comme un tableau rectangulaire de cellules SRAM (figure 2.7 chapitre 2). Les bits de configuration sont regroupés en trames de configuration de mots de largeur 1 bit qui s'étendent en colonnes du haut en bas de la mémoire CMA.

Les données de configuration sont généralement chargées dans la mémoire CMA par trames depuis une mémoire de sauvegarde externe des données de configuration, via une interface de configuration (JTAG, Mémoire Flash, ...). Une trame de données est la plus petite partie reconfigurable de la mémoire CMA : elle a une taille de 1312 bits disposés dans 41 mots de 32 bits et elle est adressable. En d'autres termes, la plus petite unité qui peut être lue ou écrite dans la mémoire CMA est appelée trame de configuration. La figure 6.1 - adaptée de [35]- montre l'arrangement des *mots* dans le *bitstream* et dans une trame

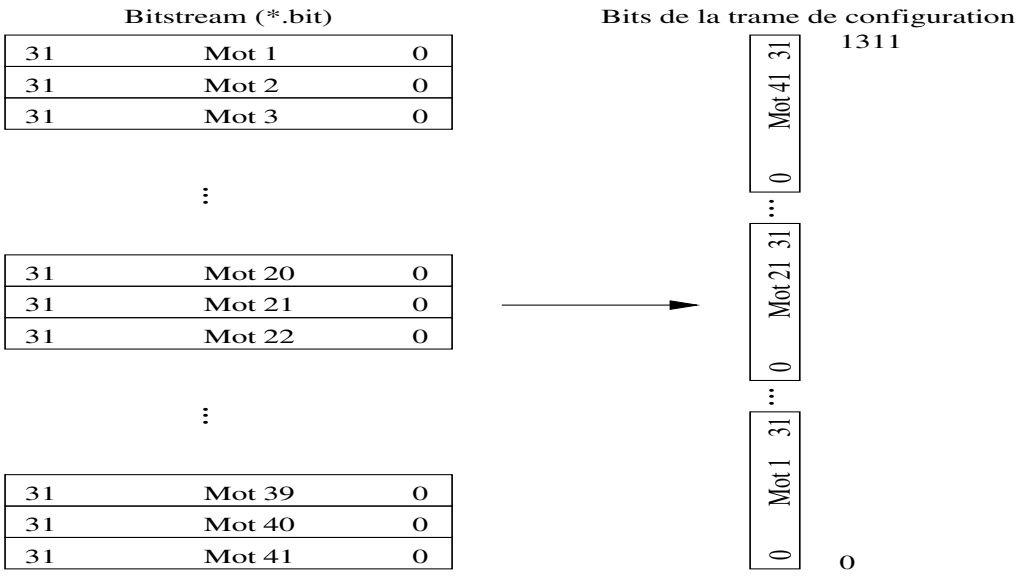


FIGURE 6.1: Arrangement des mots de configuration dans le *Bitstream* et dans une trame de configuration.

de configuration. La configuration d'un CLB est contenue dans plusieurs trames de configuration et une trame de configuration contient des parties de configuration de 20 CLBs pour la technologie Xilinx Viertex-5. Comme une trame de configuration est la plus petite partie reconfigurable de la mémoire CMA, chaque reconfiguration d'une trame change la reconfiguration d'au moins 20 CLBs.

L'adresse d'une trame de configuration est composée de cinq composantes. Elles sont sauvegardées dans le registre d'adresse de la trame (FAR)[1] qui est un registre de 32-bits. Les cinq composantes qui constituent l'adresse d'une trame sont montrées dans le tableau 6.2 [35].

Type d'adresse	Index de bit	Description
Type de block	[23 :21]	Les types de blocks valides sont : CLB, I/O, CLK (000), contenu des BRAMs (001), et CFG_CLB (010).
Bit Haut/Bas	20	Sélectionne entre la moitié supérieure de la rangée (0), et la moitié inférieure de la rangée (1).
Adresse de la rangée	[19 :15]	Sélectionne la rangée courante.
Adresse de la colonne	[14 :7]	Sélectionne une colonne principale, comme une colonne de CLBs.
Adresse secondaire	[6 :0]	Sélectionne une trame dans une colonne principale.

TABLE 6.2: Description du registre d'adresse d'une trame dans le FPGA virtex-5.

1. - Un bitstream ordinaire ne contient pas le type (010).
- Les adresses de la rangée augmentent de bas en haut.
- Les adresses des colonnes commencent de 0 sur la gauche et augmentent vers la droite.

Un FPGA-SRAM est divisé en deux parties, partie supérieure et partie inférieure. Les trames dans la partie supérieure du FPGA-SRAM sont symétriques de celles de la partie inférieure. Tous les composants sont classés en types de blocs pour faciliter l'adressage. Un FPGA-SRAM est divisé verticalement en régions d'horloge horizontales (HCLK). Ces régions sont numérotées de 0, en commençant au milieu de la puce et en incrémentant vers le haut. La partie inférieure de la puce a le même adressage, commençant au milieu de la puce et incrémentant vers le bas. Cependant, il y a le bit Haut/Bas qui différencie entre le haut et le bas du FPGA-SRAM. Comme montré dans le tableau 6.2, chaque type de bloc a sa propre adresse distribuée à partir de 0 à gauche en allant vers la droite. Cette adresse est appelée "adresse de colonne" ou "adresse principale" ou encore "colonne principale". Chaque CLB, DSP ou BRAM est composé d'un certain nombre d'adresses secondaires. Ce nombre varie suivant le type de ressource. Le tableau 6.3 montre le nombre de trames pour chaque ressource du FPGA Virtex-5 [35].

Bloc	Nombre de trames
CLB	36
DSP	28
BRAM	30
IOB	54
Colonne d'horloge	4

TABLE 6.3: Trames (adresses secondaires) pour chaque colonne dans le FPGA virtex-5.

6.2.2 Modélisation des pannes dans la mémoire CMA

Afin de spécifier certaines pannes dans la mémoire de configuration, on doit les représenter sous la forme de *panne primitive* (FP) [138] notée par $\langle S/F/R\rangle$. S décrit l'opération d'excitation de la panne ; $S \in \{0, 1, w0, w1, r0, r1, \forall\}$ où $(\forall \in \{0, 1, w0, w1, r0, r1\})$, F décrit un niveau logique de la cellule SRAM défectueuse $(F \in \{0, 1\})$ et R décrit le niveau logique à la sortie de la cellule après une opération de lecture $(R \in \{0, 1, -\})$. R a les valeurs 0 ou 1 lorsque la panne est sensibilisée par une opération de lecture tandis que "$-$" est utilisée quand c'est une opération d'écriture qui sensibilise la panne (la sortie de la donnée n'est alors pas applicable).

Remarque 6.2.
Les notations et symboles utilisées dans la section 6.2 seront utilisées dans la suite pour la représentation des tests March. Le tableau 6.4 [2] rassemble toutes les notations qui seront utilisées par la suite.

On définit un ensemble non vide de pannes FPs appelé ensemble de modèles de pannes fonctionnels *FFMs* (Functionnal Fault Models). Cet ensemble contient la classe des pannes FFM mono-cellules statiques, qui est la classe la plus importante des FFMs [138]. L'ensemble des FFMs des mono-cellules statiques est composé de FPs sensibilisées par l'exécution d'au plus une opération sur une cellule SRAM défectueuse. Les pannes mono-cellules sont les pannes qui n'influent pas le comportement des autres cellules SRAM. Autrement

Notation	Signification
r	Action sur la trame : Une opération de lecture
w	Action sur la trame : Une opération d'écriture
r0	Action sur la trame : lire les 0s de la trame
w0	Action sur la trame : écrire les 0s dans la trame
r1	Action sur la trame : lire les 1s de la trame
w1	Action sur la trame : écrire les 1s dans la trame
1**w**0	Écrire un 1 dans la cellule SRAM qui contient un 0 ou qui est en transition front-montant
0**w**1	Écrire un 0 dans la cellule SRAM contenant 1 ou la cellule en transition descendante
↕	Inverser le contenu de la cellule SRAM
⇑	Désigne un ordre d'adresse croissant (c'est-à-dire, les adresses 0,1,2, et ainsi de suite)
⇓	Désigne un ordre d'adresse croissant
⇳	Ordre d'adressage peut etre soit croissant soit décroissant.
0**w**0	Écrire un 0 dans une cellule contenant un 0
1**w**1	Écrire un 1 dans une cellule contenant un 1
x**w**x	Écrire une valeur x dans une cellule contenant déjà la valeur x
$\forall$	Désigne toutes opération d'écriture : $\forall \in \{\uparrow, \downarrow, \updownarrow, \rightarrow, \leftarrow, \Rightarrow\}$
$\langle \ldots \rangle$	Dénote une panne particulière décrite par ...

TABLE 6.4: Notations et symboles utilisées dans les tests *March*.

dit, le comportement d'une panne singulière ne peut pas changer le comportement d'une autre cellule SRAM voisine [138].

Un test March complet est délimité par des accolades "{ ... }" tandis qu'un élément du test March est délimité par des parenthèses "(...)". Les éléments d'un test March sont séparés par des points-virgules alors que les opérations dans le test March sont séparées par des virgules. Les opérations sur une cellule SRAM peuvent être soit $w0$, $w1$, $r0$ ou $r1$. Dans l'exemple 6.1, le test se compose des éléments March suivants : ⇳ $(w0)$, ⇑ $(r0, w1)$, et ⇓ $(r1, w0)$. Le premier élément du test MATS+ initialise toutes les cellules SRAM à 0*s*, le second lit les 0 suivis par une écriture des 1*s* dans chaque cellule tandis que le troisième élément procède à l'inverse du second élément [138].

Exemple 6.1. *Considérons le test MATS+ [142] (Modified Algorithm Test Sequence) suivant :*

$$MATS+ = \{⇳ (w0); ⇑ (r0, w1); ⇓ (r1, w0)\}.$$

Le test MATS+ est constitué de trois éléments March : $M0 : ⇳ (w0), M1 : ⇑ (r0, w1)$ *et* $M2 : ⇓ (r1, w0)$

Mo : { *élément March* ⇳ $(w0)$}

```
for SRAM cell := 0 to n-1 (ou tout autre ordre) do
```

```
    begin
      write 0 to SRAM cell;
    end;
```

$M1$: { *élément March* $\Uparrow (r0, w1)$}

```
    for SRAM cell := 0 to n-1 do
    begin
      read SRAM cell;  [valeur attendu =0]
      write 1 to SRAM cell;
    end;
```

$M2$: { *élément March* $\Downarrow (r1, w0)$}

```
    for SRAM cell := n-1 down to 0 do
    begin
      read SRAM cell;  [valeur attendu =1]
      write 0 to SRAM cell;
    end;
```

6.2.3 Modèles de pannes fonctionnelles (FFMs) dans la mémoire CMA

Dans cette partie, nous adoptons la classification des pannes faite dans [139] pour analyser les pannes dans la mémoire CMA. Les pannes peuvent être classées en pannes FPs-Singulières et FPs-Multiples. Une panne FP-Singulière est une panne impliquant une seule cellule SRAM tandis qu'une panne FP-Multiple est une panne impliquant plusieurs cellules SRAM. Dans ce chapitre, nous nous limitons aux FPs impliquant deux-cellules SRAM et trois-cellules SRAM. En effet, ces deux types sont les plus observés pour le test des mémoire [139] et on suppose qu'ils le sont également pour les trames de configuration de la mémoire CMA d'un FPGA-SRAM. Dans la suite, nous détaillerons ces trois types de pannes selon les définitions suivantes [2][139] :

Définition 6.1. *La panne collage-à 1/o (stuck-at 1/o fault, SAF) est une panne dans laquelle la valeur logique dans la cellule SRAM est toujours égale à* 0 *(SAF-o) ou toujours égale à* 1 *(SAF-1). La cellule est toujours dans un état défectueux qui ne peut pas être changé.*

Les pannes collage-à 0 et collage-à 1 sont décrites par les notations $\langle \forall /0/- \rangle$ et $\langle \forall /1/- \rangle$ (tableau 6.6). Quels que soient les opérations sur la cellule SRAM, la réponse est soit 0 dans le cas de la panne collage-à 0 soit 1 dans le cas de la panne collage-à 1.

Définition 6.2. *Une panne de transition (transition fault, TF) est un cas particulier des pannes collage-à o/1, dans lesquelles une cellule ne fait pas la transition* $0 \longrightarrow 1$ *(up) ou* $1 \longrightarrow 0$ *(down) lors de l'écriture.*

Une panne de transition est décrite par les notations $\langle 0w1/0/- \rangle$ et $\langle 1w0/1/- \rangle$. La figure 6.2 - adaptée de [68] - montre le diagramme d'état d'une cellule SRAM en présence de pannes collage-à o/1 et de pannes de transition.

Définition 6.3. *Une panne de pont (Bridging Fault, BF) est un court-circuit entre deux ou plusieurs cellules SRAM. C'est une panne bidirectionnelle, donc chaque cellule SRAM peut changer le contenu de l'autre.*

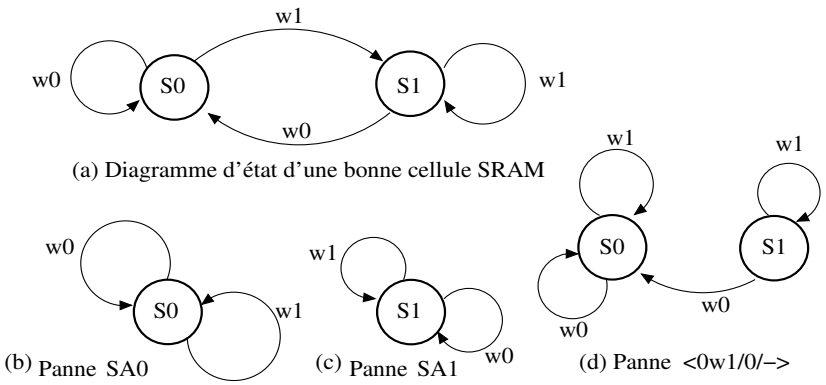


FIGURE 6.2: Modèle de diagramme de transition des états dans le cas d'une panne collage-à 0/1 (b) (c) et dans le cas d'une panne de transition (d) selon le modèle.

Définition 6.4. *On parle de pannes de couplage d'état (State coupling faults, SCF) lorsqu'une cellule mémoire SRAM C_j se trouve dans un état y et force une cellule mémoire SRAM C_i à être dans l'état x. Les quatre cas possibles d'une panne SCF sont :* $\langle 0;0\rangle$, $\langle 0;1\rangle$, $\langle 1;0\rangle$ *et* $\langle 1;1\rangle$.

Le registre FDR (Frame Data Registre) configure les trames de configuration par mot de 32 bits. Ainsi, 41 mots sont nécessaires pour configurer une trame de 1312 bits. La figure 6.1 montre les 41 mots dans le *Bitstream* et leurs correspondants dans une trame de configuration.

D'après la définition de la trame de configuration et les capacités d'accès du port ICAP [143], il va falloir changer la stratégie de test par l'algorithme March. En effet, le test March consiste à appliquer l'algorithme de test à chaque cellule SRAM avant de passer à la cellule suivante. Si un test est appliqué sur une cellule SRAM, il doit donc être appliqué sur toutes les cellules SRAM. Or, comme la configuration se fait par mots de 32 bits, nous allons appliquer le test March sur 32 bits simultanément. Cette stratégie a été publiée dans [137] pour tester les mémoires, nous l'avons adapté pour tester les trames de configuration d'un FPGA-SRAM.

6.3 Conversion des Tests March orientés bits à des tests March orientés mots

6.3.1 Position de problème

La plupart des algorithmes de test des mémoires sont optimisés pour une technologie bien précise de mémoire et notamment un ensemble particulier de modèles de panne, sous l'hypothèse que la mémoire est orientée-bit (bit-oriented). Donc, les opérations de lecture et d'écriture n'affectent qu'un seul bit (une cellule SRAM) dans la mémoire. Traditionnellement, les mémoires orientées-mot (word-oriented memories, WOMs) ont été testées par l'application répétée d'un test de mémoire orientée-bit (bit-oriented memories, BOMs) dans lequel des données (data background, DBs) [144][137] sont utilisées lors de chaque application. Il en résulte des inefficacités dans le temps de test ainsi qu'un faible taux de couverture.

Algorithm	**O**(n)	Test description
March SS	22.n	$\{\Updownarrow(w0);\Uparrow(r0,r0,w0,r0,w1);\Uparrow(r1,r1,w1,r1,w0);\Downarrow(r0,r0,w0,r0,w1);$ $\Downarrow(r1,r1,w1,r1,w0);\Updownarrow(r0)\}$
March RAW	26.n	$\{\Updownarrow(w0);\Uparrow(r0,w0,r0,r0,w1,r1);\Uparrow(r1,w1,r1,r1,w0,r0);$ $\Downarrow(r0,w0,r0,r0,w1,r1);\Downarrow(r1,w1,r1,r1,w0,r0),\Updownarrow(w0)\}$
March SL	41.n	$\{\Updownarrow(w0);\Uparrow(r0,r0,w1,w1,r1,r1,w0,w0,r0,w1);\Uparrow(r1,r1,w0,w0,r0,$ $r0,w1,w1,r1,w0);\Downarrow(r0,r0,w1,w1,r1,r1,w0,w0,r0,w1);\Downarrow(r1,r1,w0,w0,$ $r0,r0,w1,w1,r1,w0)\}$
SCAN	4.n	$\{\Uparrow(w0);\Uparrow(r0);\Uparrow(w1);\Uparrow(r1)\}$
MATS+	5.n	$\{\Updownarrow(w0);\Uparrow(r0,w1);\Downarrow(r1,w0)\}$
MATS++	6.n	$\{\Updownarrow(w0);\Uparrow(r0,w1);\Downarrow(r1,w0,r0)\}$
March C-	10.n	$\{\Updownarrow(w0);\Uparrow(r0,w1);\Uparrow(r1,w0);\Downarrow(r0,w1);\Downarrow(r1,w0);\Updownarrow(r0)\}$
PMOVI	13.n	$\{\Downarrow(w0);\Uparrow(r0,w1,r1);\Uparrow(r1,w0,r0);\Downarrow(r0,w1,r1);\Downarrow(r1,w0,r0)\}$
March SR	14.n	$\{\Downarrow(w0);\Uparrow(r0,w1,r1,w0);\Uparrow(r0,r0);\Uparrow(w1);\Downarrow(r1,w0,r0,w1);\Downarrow(r1,r1)\}$
March G	23.n	$\{\Updownarrow(w0);\Uparrow(r0,w1,r1,w0,r0,w1);\Uparrow(r1,w0,w1);\Downarrow(r1,w0,w1,w0);$ $\Downarrow(r0,w1,w0);\Uparrow(r0,w1,r1);\Uparrow(r1,w0,r0)\}$
Hammer	49.n	$\{\Uparrow(w0);\Uparrow(r0,10*w1,r1);\Uparrow(r1,10*w0,r0);\Downarrow(r0,10*w1,r1);$ $\Downarrow(r1,10*w0,r0)\}$

TABLE 6.5: Description des tests March utilisés pour la détection et le diagnostic des pannes dans les mémoires.

Une WOM contient plus d'un bit par mot. Donc on a $B \geq 2$, où B représente le nombre de bit par mot qui est généralement une puissance de 2 [137] (dans notre cas $B = 2^5$). Les opérations de lecture lisent les B bits simultanément. Les opérations d'écriture écrivent les données dans les B cellules SRAM de sorte que les données écrites peuvent être spécifiées. La méthode traditionnelle pour tester les mémoires WOMs se compose de l'application répétée d'un test BOM, par lequel un DB différent est utilisé lors de chaque application. Les tests pour détecter les pannes de couplage (CFs) [2] utilisent différents types de DBs tels que *Walking 1/o*. La même procédure a été utilisée dans [55] pour tester la mémoire CMA du FPGA-SRAM. L'inconvénient majeur de cette procédure est l'incapacité de détecter les pannes CFs intra-mot ainsi que l'incapacité de la méthode proposée de distinguer les trames configurables des non-configurables (tableau A.1).

Dans cette partie, nous utilisons une méthode systématique - basée sur les travaux publiés dans [137] - de conversion des tests *March* BOMs à des tests adaptés pour les trames de configuration dans un FPGA-SRAM, distinguant les pannes inter-mots des pannes intra-mots. La conversion consiste à la concaténation des tests BOMs et des tests pour les pannes inter-mots pour concevoir des tests WOMs pour les pannes intra-mots.

6.3.2 Tests des pannes mono-cellule et des pannes inter-mots

Les modèles de pannes pour les WOMs peuvent être divisés en deux classes :

1. Pannes Mono-Cellule : ce sont les pannes classiques de collage à 0 ou à 1 (SAFs), les pannes de transition (TFs), les pannes de conservation de données (DRFs) et les pannes de perturbation par lecture (RDFs). Elles sont détectables par les tests classiques pour les BOMs comme MATS+, MARCH C-, MATS+ . . ., le tableau 6.5 montre quelques exemples de tests *March* pour la détection des pannes dans les mémoires [2]. Le tableau 6.6 répertorie toutes les pannes de type mono-cellule [140].
2. Pannes entre les cellules SRAM : il s'agit des pannes de couplage (CFs). Elles peuvent être subdivisées en deux sous-classes :
   – pannes inter-mots : ce sont les pannes CFs classiques où la cellule SRAM *agresseur* et la cellule SRAM *victime* appartiennent à des mots différents. Elles sont détectables par les tests BOMs adéquats.
   – pannes intra-mots : ce sont les CFs où la cellule SRAM *agresseur* et la cellule SRAM *victime* appartiennent au même mot. Un test intra-mot doit être conçu pour détecter les pannes de cette sous-classe.

Les pannes intra-mots seront le sujet de ce chapitre. Selon la domination des opérations d'écriture sur les pannes CFs, les pannes CFs intra-mots peuvent ou ne peuvent pas être détectables :

1. Si l'opération d'écriture domine une panne CF, *i. e.* les valeurs spécifiées dans l'opération d'écriture sont sauvegardées dans les cellules SRAMs, la panne CF n'aura aucun effet et par conséquent ne semblera pas être présente. Un test pour détecter cette panne non-dominante ne serait pas utile car la panne ne se manifestera pas.
2. Si une panne CF domine l'opération d'écriture, la panne CF se manifestera. Il sera donc nécessaire de détecter ces pannes. Dans le cas des pannes CFids, CFdsts, CFsts [2], une panne CF ne sera pas détectable avec les algorithmes BOMs.

#	Panne	Panne primitive (FP)	Nom
1	SAF	$\langle \forall/0/- \rangle$,$\langle \forall/1/- \rangle$	stuck-at-fault
2	TF	$\langle 0w1/0/- \rangle$,$\langle 1w0/1/- \rangle$	Transition fault
3	WDF	$\langle 0w0/1/- \rangle$,$\langle 1w1/0/- \rangle$	Write destructive fault
4	RDF	$\langle 0r0/1/- \rangle$,$\langle 1r1/0/- \rangle$	Read destructive fault
5	IRF	$\langle 0r0/0/1 \rangle$,$\langle 1r1/1/0 \rangle$	Incorrect read fault
6	DRDF	$\langle 0r0/1/0 \rangle$,$\langle 1r1/0/1 \rangle$	Deceptive RDF

TABLE 6.6: Ensemble de pannes mono-cellule statiques FFMs avec les ensembles de pannes primitives (FPs) correspondants.

Tout test BOM peut être converti en un test WOM pour détecter les pannes mono-cellule et les pannes inter-mots, comme suit [137] :

1. L'opération "$w0$" doit être remplacée par "$w - data\ background$" noté "w_D" de sorte que toute *DB* soit acceptable. Par exemple : "$w00 \ldots 00$", "$w11 \ldots 11$", "$w01 \ldots 01$" (Un DB de séquence répétée de 01), tel que la taille de *DB* est de *B* bits. L'opération "$w1$" doit être remplacée par une opération d'écriture qui écrit le complément de *DB* ($w_{\overline{D}}$).
2. Les opérations "$r0$" et "$r1$" doivent être remplacées par les opérations "$r - data\ background$" (r_D) et "$r - complement - data\ background$" ($r_{\overline{D}}$) successivement.

3. Dans les équations exprimant le nombre requis d'opérations pour un test, n doit être remplacé par n/B (qui représentant le nombre de mots par trame de configuration).

Reprenons l'exemple 6.1, pour convertir le test BOM MATS+ : $\{\Updownarrow (w0); \Uparrow (r0, w1); \Downarrow (r1, w0)\}$ en un test WOM MATS+ avec $B = 32$ *bits*, en utilisant la séquence "$\underbrace{01\ldots01}_{32\text{ bits}}$" comme *DB*, le test MATS+ devient :

$$\{\Updownarrow (w\underbrace{01\ldots01}_{32\text{ bits}}); \Uparrow (r\underbrace{01\ldots01}_{32\text{ bits}}, w\underbrace{10\ldots10}_{32\text{ bits}}); \Downarrow (r\underbrace{10\ldots10}_{32\text{ bits}}, w\underbrace{01\ldots01}_{32\text{ bits}})\}$$

Le *DB* choisi est "$\underbrace{01\ldots01}_{32\text{ bits}}$" car il permet également de détecter certaines pannes CFs entre des cellules SRAM adjacentes. La complexité du test se change de $5 * n$ à $5 * n/B$. D'autres *DBs* sont possibles, par exemple :

$$\{\Updownarrow (w\underbrace{00\ldots00}_{32\text{ bits}}); \Uparrow (r\underbrace{00\ldots00}_{32\text{ bits}}, w\underbrace{11\ldots11}_{32\text{ bits}}); \Downarrow (r\underbrace{11\ldots11}_{32\text{ bits}}, w\underbrace{00\ldots00}_{32\text{ bits}})\}$$

Un autre exemple concernant la conversion du test BOM March C- $\{\Updownarrow (w0); \Uparrow (r0, w1); \Uparrow (r1, w0); \Downarrow (r0, w1); \Downarrow (r1, w0); \Updownarrow (r0)\}$ [145] en un test WOM March C-, avec $B = 32$ *bits* et utilisant le *DB* : "$\underbrace{00\ldots00}_{32\text{ bits}}$"

$$\{\Updownarrow (w\underbrace{00\ldots00}_{32\text{ bits}}); \Uparrow (r\underbrace{00\ldots00}_{32\text{ bits}}, w\underbrace{11\ldots11}_{32\text{ bits}}); \Uparrow (r\underbrace{11\ldots11}_{32\text{ bits}}, w\underbrace{00\ldots00}_{32\text{ bits}});$$

$$\Downarrow (r\underbrace{00\ldots00}_{32\text{ bits}}, w\underbrace{11\ldots11}_{32\text{ bits}}); \Downarrow (r\underbrace{11\ldots11}_{32\text{ bits}}, w\underbrace{00\ldots00}_{32\text{ bits}}); \Updownarrow (r\underbrace{00\ldots00}_{32\text{ bits}})\}$$

Remarque 6.3.
Les modèles de pannes inter-mots détectés par ce test WOM sont les SAFs, TFs, RDFs, CFsts et CFids [68].

Cependant, cette méthode de conversion des tests *March* ne garantit pas la détection de toutes les pannes CFs intra-mots [68]. La figure 6.3 montre un mot de 32 bits d'une trame de configuration. Ce mot est composé de 32 cellules SRAM : $W_{32} = (C_0, C_1, \ldots, C_{31})$ et d'une panne *CFid* entre les cellules C_i est $\langle 1w0; 1\rangle$ ou $\langle 0w1; 0\rangle$ et la cellule C_j. L'opération $0w1$ signifie que une transition d'écriture de 1 dans la cellule *agresseur* C_i cause une écriture de 1 dans la cellule *victime* C_j. Supposons que le mot contienne "$\underbrace{00\ldots00}_{32\text{ bits}}$", une opération d'écriture "$\underbrace{w11\ldots11}_{32\text{ bits}}$" peut être utilisée pour exciter les pannes CFid. Cependant, elle peut masquer le CFid entre les cellules C_i et C_j. La détection de cette panne requière une opération d'écriture "$wxxx\ldots xx1_ixx\ldots xx0_jxx\ldots xx$" avec $x \in \{0, 1\}$ de sorte que la panne *CFid* ne soit pas masquée.

6.3.3 Modèles de pannes de couplage

Une panne de couplage est une FP sensibilisée par au moins une opération ainsi que l'observation de son effet sur deux cellules SRAMs différentes. Ce modèle de panne peut être

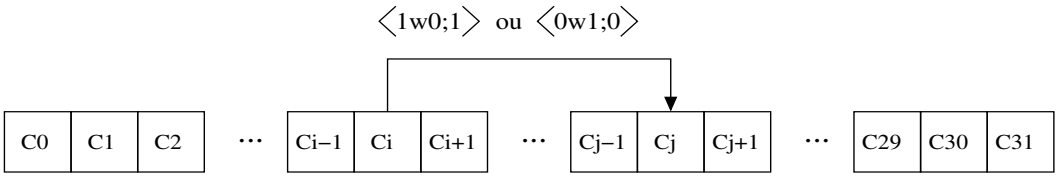


FIGURE 6.3: Panne intra-mot CFid

représenté par $\langle S/F/R\rangle = \langle S_a; S_v/F/R\rangle_{a,v}$ où S_a et S_v sont respectivement les séquences effectuées sur la cellule *agresseur* et la cellule *victime*. La cellule *agresseur* est la cellule sur laquelle l'opération de sensibilisation doit être effectuée tandis que la cellule *victime* est la cellule où la panne apparaît. Autrement dit, l'opération effectuée sur la cellule *agresseur* va exciter la panne dans la cellule *victime* (S_a et $S_v \in \{0, 1, 0w0, 1w1, 0w1, 1w0, 0r0, 1r1\}$).

6.4 Détection des pannes intra-mots dans les trames de configuration

Les *DBs* qui seront utilisés pour la détection des pannes intra-mots sont déterminés selon les modèles de panne de couplage utilisés. Dans cette section, on ne suppose aucune connaissance sur le layout des trames de configuration. Les modèles de panne reflétant cette hypothèse sont les modèles *CFs* sans restriction (unrestrected CFs models, uCFs).

6.4.1 Ensemble de DBs pour la détection des pannes uCFsts

Définition 6.5. *Une panne de couplage d'état (CFsts) est le modèle de panne qui consiste au changement forcé à un état x d'une cellule SRAM couplée c_i à une cellule SRAM de couplage c_j d'état y, où $x, y \in \{0, 1\}$ [137].*

Afin de pouvoir détecter les pannes *uCFsts* dans les trames de configuration, tous les états arbitraires des deux cellules C_i et C_j, à savoir les états $(C_i, C_j) \in \{(0,0), (0,1), (1,0), (1,1)\}$, doivent être testés. Initialement, seuls les états $(0,0)$ et $(0,1)$ doivent être testés. Les deux autres états seront testés lorsque l'algorithme est exécuté avec le *complémentaire* de *DBs*. Le nombre d de *DBs* requis pour vérifier les états $(0,0)$ et $(0,1)$ est [144] : $d = \log_2 B + 1$.

Comme il faut prendre en compte les *DBs* complémentaires, le nombre d pour tester tous les états possibles est : $d = 2\log_2 B + 2$. La nature des pannes *uCFsts* est telle que seuls les états des cellules SRAM qui comptent pour la détection des pannes uCFsts sont vérifiés. Par conséquent, les *DBs* peuvent être appliqués dans un ordre arbitraire. Chaque mot d'une trame de configuration a une taille $B = 32$ bits, les *DBs* du tableau 6.7 peuvent être appliqués.

Soit V_i le vecteur constitué des bits occupant la i^{ieme} place dans chaque *DB*. Pour un ensemble minimal de *DBs*, les deux conditions suivantes doivent être satisfaites [137] :

1. aucuns vecteurs V_i et V_j ne doivent être égaux. Si c'est le cas, les cellules SRAMs correspondantes c_i et c_j doivent être seulement vérifiées pour les états $(0,0)$ et $(1,1)$. Si il n'y a pas deux vecteurs égaux, le nombre maximal des vecteurs possibles d est déterminé par la taille du mot comme suit : $2^d = B$.

#	*DBs* normales	#	*DBs* compléments
0	00000000000000000000000000000000	1	11111111111111111111111111111111
2	01010101010101010101010101010101	3	10101010101010101010101010101010
4	00110011001100110011001100110011	5	11001100110011001100110011001100
6	00001111000011110000111100001111	7	11110000111100001111000011110000
8	00000000111111110000000011111111	9	11111111000000001111111100000000
10	00000000000000001111111111111111	11	11111111111111110000000000000000

TABLE 6.7: *DBs* pour le test des pannes *uCFsts* dans les trames de configuration.

2. aucun vecteur V_i ne doit être le complémentaire d'un vecteur V_j. Dans le cas contraire seuls les états $(0,1)$ et $(1,0)$ des cellules SRAMs c_j et c_j qui sont sensés être vérifiés. Quand il n'y a pas de vecteurs compléments, d on a $d = \log_2 B + 1$.

Pour la détection des pannes *uCFsts* : $\Updownarrow (w_{D_0}, r_{D_0}, \ldots, w_{D_{d-1}}, r_{D_{d-1}})$, avec les données D_0 jusqu'à D_{d-1} prises du tableau 6.7 de telle sorte que les valeurs normales et complémentaires soient couvertes. La figure 6.4 montre un test *March* WOM basé sur le test *March C-* pour la détection des pannes *uCFsts* dans les trames de configuration.

$$\{\Updownarrow_0 (w00\ldots00); \Uparrow_1 (r00\ldots00, w11\ldots11); \Uparrow_2 (r11\ldots11, w00\ldots00); \Downarrow_3 (r00\ldots00, w11\ldots11); \Downarrow_4 (r11\ldots11, w00\ldots00); \Downarrow_5 (r00\ldots00, w0101\ldots0101); \Uparrow_6 (r0101\ldots0101, w1010\ldots1010); \Downarrow_7 (r1010\ldots1010, w0101\ldots0101); \Uparrow_8 (r0101\ldots00, w0011\ldots0011); \Downarrow_9 (r0011\ldots0011, w1100\ldots1100); \Uparrow_{10} (r1100\ldots1100, w0011\ldots0011); \Updownarrow_{11} (r0011\ldots0011)\}$$

FIGURE 6.4: Test March *WOM* pour les pannes *uCFsts* basé sur le test *March C-*.

La complexité du test passe d'une complexité de $10 * n$ pour un test BOM à une complexité de $22 * n/32$, où $n = 1312$ bits.

6.4.2 Ensemble de DBs pour la détection des pannes uCFids

Définition 6.6. *On parle d'une panne de couplage idempotent (idempotent coupling faults, CFids) lorsqu'une transition* $1w0$ *ou* $0w1$ *dans une cellule SRAM* c_j *met le contenu d'**une** cellule SRAM* c_i *à* 0 *ou à* 1 *[137].*

Dans le but de détecter les pannes *uCFids* dans les mots d'une trame de configuration, un ensemble de *DBs* est requis [146]. Cet ensemble doit être appliqué selon des séquences particulières (*data background sequence, DBS*). Dans le cas où une panne *uCFid* domine l'opération d'écriture, les algorithmes de test BOMs doivent être modifiés en utilisant les *DBS*. Pour les pannes *uCFids* dans un mot d'une trame de configuration, quatre types de *uCFids* existent : $\langle 1w0; 0w1\rangle_{c_a,c_v}$ (c_a désigne la cellule SRAM *agresseur* et c_v désigne la cellule SRAM *victime*), $\langle 1w0; 1w0\rangle_{c_a,c_v}$, $\langle 0w1; 0w1\rangle_{c_a,c_v}$ et $\langle 0w1; 1w0\rangle_{c_a,c_v}$ où $a, b \in \{0, 1, \ldots, 31\}$. Étant donné que chaque cellule SRAM parmi B cellules peut être une cellule *agresseur* et qu'une cellule parmi (B-1) peut être une cellule *victime*, il y a $B * (B-1) = 32 * 31 = 992$ cas possible

pour chaque type de panne *uCFid*, ce qui fait que le nombre de tous les cas possibles est : $4 * B * (B-1) = 3968$. Le but de cette section est de trouver le minimum de *DBS* qui peuvent exciter les $4 * B * (B-1)$ pannes uCFids.

Remarque 6.4.
Pour toutes les cellules SRAM couplées, chaque cellule doit être lue après une série de DBS, de sorte que l'excitation d'une cellule SRAM par une panne primitive uCFids ne masque pas les autres pannes uCFids dans les autres cellules SRAM.

Comme indiqué au début de ce paragraphe, la remarque 6.4 implique l'application des *DBs* selon des *DBS*.

Pour chaque paire de cellule SRAM (c_i, c_{i+1})	
0	00000000000000000000000000000000
1	11111111111111111111111111111111
2	00000000000000000000000000000000
3	01010101010101010101010101010101
4	10101010101010101010101010101010
5	01010101010101010101010101010101
Pour chaque paire de cellule SRAM (c_i, c_{i+2})	
0	00110011001100110011001100110011
1	11001100110011001100110011001100
2	00110011001100110011001100110011
Pour chaque paire de cellule SRAM (c_i, c_{i+4})	
0	00001111000011110000111100001111
1	11110000111100001111000011110000
2	00001111000011110000111100001111
Pour chaque paire de cellule SRAM (c_i, c_{i+8})	
0	00000000111111110000000011111111
1	11111111000000001111111100000000
2	00000000111111110000000011111111
Pour chaque paire de cellule SRAM (c_i, c_{i+16})	
0	00000000000000001111111111111111
1	11111111111111110000000000000000
2	00000000000000001111111111111111

TABLE 6.8: *DBS* pour le test des pannes *uCFids* dans les trames de configuration.

Les DBS du tableau 6.8 peuvent exciter toutes les pannes uCFids dans un mot d'une trame de configuration. Il y a plusieurs versions des *DBS* du tableau 6.8 car les séquences peuvent être combinées en plusieurs façons.

Pour les pannes *uCFids* :

$$\Updownarrow (w_{D_0}, r_{D_0}, w_{D_1}, r_{D_1}, \ldots, r_{D_{d-2}}, w_{D_{d-1}}, r_{D_{d-1}})$$

$$\{\Updownarrow_0 (w00\ldots00); \Uparrow_1 (r00\ldots00, w11\ldots11); \Uparrow_2 (r11\ldots11, w00\ldots00); \Downarrow_3 (r00\ldots00, w11\ldots11); \Downarrow_4 (r11\ldots11, w00\ldots00); \Downarrow_5 (r00\ldots00, w0101\ldots0101); \Uparrow_6 (r0101\ldots0101, w1010\ldots1010); \Downarrow_7 (r1010\ldots1010, w0101\ldots0101); \Uparrow_8 (r0101\ldots00, w0011\ldots0011); \Downarrow_9 (r0011\ldots0011, w1100\ldots1100); \Uparrow_{10} (r1100\ldots1100, w0011\ldots0011); \Updownarrow_{11} (r0011\ldots0011)\}$$

FIGURE 6.5: Test March *WOM* optimisé pour la détection des pannes *uCFids* dans les trames de configuration d'un FPGA, basé sur le test *March C-*.

tel que les données D_0 jusqu'à D_{d-1} représentent les *DBS* du tableau 6.8. La figure 6.5 montre un test *March* WOM basé sur le test *March C-* pour la détection des pannes uCFids dans les trames de configuration. La complexité du test passe d'une complexité $10 * n$ pour un test BOM à une complexité $22 * n/32$, où $n = 1312$ bits.

6.4.3 Ensemble de *DBs* pour la détection des pannes *uCFdsts*

Définition 6.7. *On parle d'une panne uCFdsts (disturb coupling faults) lorsqu'une opération* $w0$, $w1$, $r0$, *ou* $r1$ *sur une cellule SRAM* c_j *produit une transition* $1w0$ *ou* $0w1$ *dans une cellule SRAM* c_i *[137].*

Un test intra-mots pour une panne *uCFdsts*, dans le cas où cette panne *uCFdst* domine l'opération d'écriture, utilise une séquence d'opérations de *DBs* (*sequence data background operations, DBOS*). Il existe huit sous-types de panne uCFdsts [146] : $\langle r0; 1w0\rangle_{c_a,c_v}$, $\langle r0; 0w1\rangle_{c_a,c_v}$, $\langle r1; 1w0\rangle_{c_a,c_v}$, $\langle r1; 0w1\rangle_{c_a,c_v}$, $\langle w1; 1w0\rangle_{c_a,c_v}$, $\langle w1; 0w1\rangle_{c_a,c_v}$, $\langle w0; 1w0\rangle_{c_a,c_v}$ et $\langle w0; 0w1\rangle_{c_a,c_v}$ avec $a, v \in \{0, 1, \ldots, B-1\}$. Chaque sous-type a $B * (B-1)$ cas possible. Le nombre total des cas *uCFdsts* est donc $8 * B * (B-1)$. D'une manière similaire aux cas *uCFids*, nous devrons chercher le nombre minimal des *DBS* qui peuvent exciter les $8 * B * (B-1)$ cas possible des *uCFdsts* et ensuite établir le nombre minimal d'opérations qui utilisent ces *DBS* (soit le nombre minimal de *DBOS*).

Le tableau 6.9 illustre les *DBOS* pour la détection des pannes *uCFdsts* dans les mots d'une trame de configuration.

Pour les pannes *uCFdsts* :

$$\Updownarrow (w_{D_0}, r_{D_0}, r_{D_0}, \ldots, w_{D_{d-1}}, r_{D_{d-1}}, w_{D_{d-1}})$$

Les *DBOS* utilisés dans le test sont ceux du tableau 6.9.
Le test pour les pannes intra-mots ci-dessus peut être modifié sans aucun impact sur le taux de couverture :

1. Des opérations de lecture supplémentaires peuvent être ajoutées, par exemple pour rendre le test plus symétrique et/ou pour détecter d'autres pannes.
2. Un élément March peut être divisé en un certain nombre d'éléments March et pour chaque élément March, l'ordre des adresses peut être choisi librement.

Ce choix libre de modifier le test intra-mot permet :

#	Operation	DB
0		00000000000000000000000000000000
1	w	11111111111111111111111111111111
2-3	$2 * r$	11111111111111111111111111111111
4	w	00000000000000000000000000000000
5-6	$2 * r$	00000000000000000000000000000000
7	w	01010101010101010101010101010101
8	w	10101010101010101010101010101010
9-10	$2 * r$	10101010101010101010101010101010
11	w	01010101010101010101010101010101
12-13	$2 * r$	01010101010101010101010101010101
14	w	00110011001100110011001100110011
15	w	11001100110011001100110011001100
16-17	$2 * r$	11001100110011001100110011001100
18	w	00110011001100110011001100110011
19-20	$2 * r$	00110011001100110011001100110011
21	w	00001111000011110000111100001111
22	w	11110000111100001111000011110000
23-24	$2 * r$	11110000111100001111000011110000
25	w	00001111000011110000111100001111
26-27	$2 * r$	00001111000011110000111100001111
28	w	00000000111111110000000011111111
29	w	11111111000000001111111100000000
30-31	$2 * r$	11111111000000001111111100000000
32	w	00000000111111110000000011111111
33-34	$2 * r$	00000000111111110000000011111111
35	w	00000000000000001111111111111111
36	w	11111111111111110000000000000000
37-38	$2 * r$	11111111111111110000000000000000
39	w	00000000000000001111111111111111
40-41	$2 * r$	00000000000000001111111111111111

TABLE 6.9: Ensemble de *DBOS* pour la détection des pannes *uCFdsts*.

1. Une réduction du temps de test. Si des éléments du test March intra-mots peuvent être rendus identiques à des éléments du test March inter-mots, les éléments intra-mots peuvent être retirés du test.

$\{\Updownarrow_0 (w00\ldots00); \Downarrow_1 (r00\ldots00, w11\ldots11); \Uparrow_2 (r11\ldots11, w00\ldots00, r00\ldots00, r00\ldots00, w11\ldots11); \Uparrow_3 (r11\ldots11, w00\ldots00); \Uparrow_4 (r00\ldots00, w11\ldots11, r11\ldots11, r11\ldots11, w00\ldots00); \Uparrow_5 (r00\ldots00, w0101\ldots0101, w1010\ldots1010, r1010\ldots1010); \Downarrow_6 (r1010\ldots1010, w0101\ldots0101, r0101\ldots0101); \Uparrow_7 (r0101\ldots0101, w0011\ldots0011, w1100\ldots1100, r1100\ldots1100); \Downarrow_8 (r1100\ldots1100, w0011\ldots0011, r0011\ldots0011); \Uparrow_9 (r0011\ldots0011, w00001111\ldots00001111, w11110000\ldots11110000, r11110000\ldots11110000); \Downarrow_{10} (r11110000\ldots11110000, w00001111\ldots00001111, r00001111\ldots00001111); \Uparrow_{11} (r00001111\ldots00001111, w0000000011111111000000011111111, w1111111100000000111111110000000, r11111111000000001111111100000000); \Downarrow_{12} (r1111111100000000, 11111111000000000w0000000011111111000000011111111, r0000000011111111000000001111111); \Uparrow_{13} (r0000000011111111000000011111111, w0000000000000000111111111111111, w11111111111111110000000000000000, r1111111111111111000000000000000); \Downarrow_{14} (r1111111111111111000000000000000w0000000000000000111111111111111, r00000000000000001111111111111111); \Uparrow_{15} (r0000000000000000111111111111111)\}$

FIGURE 6.6: Test *March WOM* pour la détection des pannes *uCFdsts* basé sur le test *March LR*.

2. Une augmentation du taux de couverture. Ceci peut être optimisé lorsque le test March intra-mots possède les propriétés suivantes :
   - Il est composé de plusieurs éléments March puisque chaque élément effectue un balayage de la trame.
   - Tous les éléments March commencent par une opération de lecture, ce qui permet la détection des pannes *CFs*.
   - L'ordre des adresses des éléments du test March doit varier autant que possible. En effet, cela maximise la probabilité de détecter des pannes dynamiques [68].

6.4.4 Corrélation des tests *March WOM*

L'analyse des tests *March* proposés pour la détection des pannes *uCFsts*, *uCFids* et *uCFdsts* montre que ces tests peuvent détecter plusieurs d'autres types de pannes simples comme les pannes SAFs, TFs, RDFs, etc.

Les tests *March* orientés *WOM* pour les pannes *uCFsts*, *uCFids* et *uCFdsts* couvrent aussi les pannes *SAFs*, *TFs* et *RDFs*. Ceci est dû au fait que l'utilisation de toute valeur *DB* (D) et de son complément $(\overline{D})$ permet de détecter ces pannes.

Les tests *WOM* pour les pannes *uCFids* couvrent aussi les pannes *uCFsts*. L'excitation des pannes *uCFids* requièrent les opérations $1w0$ et $0w1$ dans une cellule *agresseur* C_a alors que la cellule *victime* C_v est dans l'état 0 ou 1. Cela veut dire que les quatre suivant cas vont apparaître : $(a, v) \in \{(0,0), (0,1), (1,0), (1,1)\}$. Cela garantie la détection de toutes les pannes *uCFsts*.

Les tests *WOM* pour les *uCFdsts* couvrent toutes les *uCFids*. Le *DBS* (section 6.4.3) basé sur les transitions $1w0$ et $0w1$ va automatiquement exciter toutes les pannes *uCFids* [137].

6.5 Tests des pannes de couplage intra-mots restreintes dans les trames de configuration

Les tests proposés pour les pannes *uCFs* supposent que chaque cellule agresseur a dans un mot d'une trame de configuration peut influencer toute cellule dans ce même mot. Par conséquent, ces tests ne nécessitent pas d'informations sur l'organisation des cellules dans la trame de configuration. Cette section étudie les modèles de panne *CFs*, en supposant que le *layout* de la mémoire CMA est connu. Les pannes *CFs restreintes (rCFs)* sont telles que la cellule *victime* v ne peut être influencée que par une seule cellule *agresseur* a, qui est la cellule voisine de la cellule v. On parle de modèle *concurrent-restricted CFs (crCFs)* lorsque la cellule v peut être influencée seulement par l'action concurrente ou l'état des deux cellules SRAM *agresseurs* a voisines, d'un côté ou de l'autre de la cellule *victime* v. L'utilisation des modèles *rCF* et *crCF* permet une réduction du temps de test [137].

6.5.1 Pannes *CFs* restreintes (*rCFs*)

Une raison similaire à celle du [147] pour la restriction des pannes de couplage dans un mot est le fait qu'une cellule SRAM *agresseur* a ne peut influencer que sa cellule voisine. Plusieurs types de pannes *rCF* existent : *rCFsts*, *rCFids* et *rCFdsts*. Les *rCFs* requièrent la connaissance du *layout* de la mémoire de configuration. La méthode du test *WOM* pour les pannes *rCFs* dépend de l'organisation de la trame de configuration. Dans ce cas nous admettons que les cellules SRAM sont adjacentes :

1. **rCFsts** : Dans le cas où la panne *rCFsts* domine l'opération d'écriture, tous les états des cellules adjacentes doivent être testés. Ces états sont : $(i, i+1) \in \{(0,0),(0,1),(1,0),(1,1)\}$. L'ensemble des *DBs* qui satisfont cette condition est donné dans le tableau 6.10. Ce type de panne réduit la taille de l'ensemble des *DBs* de $d = 2 * (\log_2(32) + 1)$ à $d = 4$, indépendamment de B. La complexité est alors $\mathbf{O}(n) = 4$ en supposant que tous les 0s et tous les 1s sont vérifiés par le test *BOM* correspondant.
2. **rCFids** : Dans le cas des pannes rCFids qui dominent l'opération d'écriture, les *DBS* nécessaires pour sensibiliser toutes les pannes *rCFids* sont les six premiers *DBS* du tableau 6.8. Ils peuvent sensibiliser toutes les pannes *CFids* entre des cellules SRAM adjacentes, et donc tous les *rCFids*. Le modèle de panne *rCFids* réduit la taille des *DBS* de $d = 3 * (1 + \log_2(32))$ à $d = 6$ et a une complexité $\mathbf{O}(n) = 10$, indépendamment de B.
3. **rCFdsts** : Pour pouvoir sensibiliser toutes les pannes *rCFdsts*, il suffit d'appliquer les *DBOS* 0-13 du tableau 6.9. Cela est suffisant parce que ces *DBOS* peuvent exciter toutes les pannes *CFdsts* entre les cellules SRAM adjacentes. Cela réduit la taille de d à 6 et $\mathbf{O}(n)$ à 13, indépendamment de B.

#	Normale	#	Complément
0	00000000000000000000000000000000	1	11111111111111111111111111111111
2	01010101010101010101010101010101	3	10101010101010101010101010101010

TABLE 6.10: Ensemble de *DBs* de 32-bits pour les pannes *rCFsts*.

Les tests *March WOM* procèdent par l'écriture de $B = 32$ bits à la fois, dans le cas des trames de configuration. Cette écriture se fait d'une manière concurrente. Cela mène vers une nouvelle modélisation des pannes de couplage dans les trames de configuration.

Le test March *WOM*, basé sur le test March C-, peut s'appliquer aux pannes *rCFids* et aux pannes crCFids. La figure 6.7 montre la conversion du test March C- pour détecter les pannes *rCFids*. Les éléments M_0 jusqu'à M_4, avec l'opération $r\underbrace{00\ldots00}_{32\text{ bits}}$ de M_5, représentent le test inter-mots. Les éléments de M_5 jusqu'à M_{10} représentent le test optimisé pour les pannes *rCFids* intra-mots. Il applique les six premiers *DBS* du tableau 6.8. La complexité est : $\mathbf{O}(n) = 20 * n/B$. La figure montre la version test *WOM* du test March C- pour détecter les pannes *crCFids*. Ces éléments représentent également le test intra-mots pour les pannes *crCFids*. Ce test est basé sur le *DBS* du tableau 6.13. La complexité est : $\mathbf{O}(n) = 29 * n/B$.

$\{\Updownarrow_0\ (w00\ldots00); \Uparrow_1\ (r00\ldots00, w11\ldots11); \Uparrow_2\ (r11\ldots11, w00\ldots00); \Downarrow_3\ (r00\ldots00,$
$w11\ldots11); \Downarrow_4\ (r11\ldots11, w00\ldots00); \Downarrow_5\ (r00\ldots00, w11\ldots11); \Uparrow_6\ (r11\ldots11, w00\ldots00);$
$\Downarrow_7\ (r00\ldots00, w11\ldots11); \Uparrow_8\ (r0101\ldots0101, w1010\ldots1010); \Downarrow_9\ (r1010\ldots1010,$
$w0101\ldots0101); \Updownarrow_{10}\ (r0101\ldots0101)\}$

FIGURE 6.7: Test *WOM* pour les pannes de couplage *rCFids* pour un mot de 32-bits, basé sur le test *March C-*.

$\{\Updownarrow_0\ (w00\ldots00); \Uparrow_1\ (r00\ldots00, w11\ldots11); \Uparrow_2\ (r11\ldots11, w00\ldots00, r00\ldots00, r00\ldots00,$
$w11\ldots11); \Uparrow_3\ (r11\ldots11, w00\ldots00); \Uparrow_4\ (r00\ldots00, w11\ldots11, r11\ldots11, r11\ldots11,$
$w00\ldots00; \Uparrow_5\ (r00\ldots00, w11\ldots11, r11\ldots11); \Downarrow_6\ (r11\ldots11, w00\ldots00, r00\ldots00);$
$\Downarrow_7\ (r00\ldots00, w0101\ldots0101, w1010\ldots1010, r1010\ldots1010); \Downarrow_8\ (r1010\ldots1010,$
$w0101\ldots0101, r0101\ldots0101); \Uparrow_9\ (r0101\ldots0101, w0011\ldots0011, w1100\ldots1100,$
$r1100\ldots1100); \Downarrow_{10}\ (r1100\ldots1100, w0011\ldots0011), r0011\ldots0011); \Uparrow_{11}\ (r0011\ldots0011)\}$

FIGURE 6.8: Test *WOM* pour les pannes de couplage *rCFdsts* pour un mot de 32-bits, basé sur le test *March LR*.

6.5.2 Pannes *rCFs* concurrentielles (*crCFs*)

D'une manière conceptuelle, $B-1$ cellules SRAM peuvent agir en même temps en tant que cellules *agresseurs* pour une cellule SRAM *victime* donnée. Quand le layout des trames de configuration est connu, les tests peuvent être considérablement simplifiés quand les cellules *agresseurs* sont restreintes aux deux cellules SRAM voisines de la cellule *victime*. Une panne *crCF* peut apparaître pendant une opération de lecture ou d'écriture dans trois

cellules SRAM adjacentes du même mot (c_{a1}, c_v et c_{a2}). La figure 6.9 - adaptée de [137] - schématise une panne *crCFid* entre des cellules SRAM adjacentes.

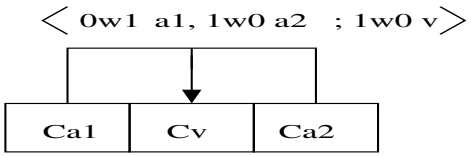


FIGURE 6.9: Panne *crCFid* entre des cellules SRAM adjacentes.

Les opérations de lecture ou d'écriture dans les deux cellules agresseurs peuvent modifier le contenu de la cellule *victime* située entre ces deux cellules [67][147]. Nous donnerons une analyse du modèle *crCF* pour les différents types de panne suivants : *crCFst*, *crCFid* et *crCFdst*.

Dans le but de détecter toutes les pannes *crCFsts*, tous les états des trois cellules SRAM adjacentes $(i-1, i, eti+1)$ doivent être vérifiés, soient les états :

$$(i-1,i,i+1) \in \{(0,0,0),(0,0,1),(0,1,0),(0,1,1),(1,0,0),(1,0,1),(1,1,0),(1,1,1)\}$$

Le tableau 6.11 regroupe les *DBs* pour la détection des pannes *crCfsts*. La taille de l'ensemble est $d = 8$ et la complexité est $\mathbf{O}(n) = 12$ (en supposant que tous les 0s et tous les 1s sont vérifiés par les tests *March BOM* correspondants). Ces paramètres sont indépendants de la taille du mot B.

#	Normale	#	Complément
0	00000000000000000000000000000000	1	11111111111111111111111111111111
2	00100100100100100100100100100100	3	11011011011011011011011011011011
4	01001001001001001001001001001001	5	10110110110110110110110110110110
6	01101101101101101101101101101101	7	10010010010010010010010010010010

TABLE 6.11: Ensemble de *DBs* de 32-bits pour les pannes *crCFsts*.

Pour le type de panne *crCFid*, il existe huit sous-types : $\langle 1w0, 1w0; 1w0\rangle_{ca1,ca2,cv}$ ($ca1$ et $ca2$ désignent les cellules *agresseurs* et cv désigne la cellule *victime*), $\langle 1w0, 1w0; 0w1\rangle_{ca1,ca2,cv}$, $\langle 1w0, 1w0; 0w1\rangle_{ca1,ca2,cv}$, $\langle 1w0, 0w1; 0w1\rangle_{ca1,ca2,cv}$, $\langle 1w0, 0w1; 1w0\rangle_{ca1,ca2,cv}$, $\langle 0w1, 1w0; 0w1\rangle_{ca1,ca2,cv}$, $\langle 1w0, 0w1; 0w1\rangle_{ca1,ca2,cv}$ et $\langle 0w1, 0w1; 0w1\rangle_{ca1,ca2,cv}$, avec $a1 = i-1$, $a2 = i+1$, $v = i$ et $i \in \{1, 2, \ldots, 30\}$.
Chaque cellule SRAM peut se comporter comme une cellule *victime* ou une cellule *agresseur*. Un ensemble de *DBS*, pour trois cellules SRAM adjacentes et qui peuvent sensibiliser toutes les pannes *crCFids*, est donné dans le tableau 6.12.

Remarque 6.5.
Un modèle de panne rcCF où une cellule SRAM agresseur ne peut pas impacter la cellule victime est noté "-" (tableau 6.12 [137]).

Pour sensibiliser toutes les pannes *crCFids* dans un mot d'une trame de configuration, les *DBS* du tableau 6.12 doivent être appliqués à chaque groupe de trois cellules adjacentes dans un mot. On peut étendre les *DBS* du tableau 6.12 à un mot de taille $B = 32$. Cela

#	DBS $c_{a1}c_vc_{a2}$	Panne détectée
0	000	
1	111	$\langle 1w0, 1w0; 0w1\rangle_{c_{a1},c_{a2};c_v}$
2	000	$\langle 0w1, 0w1; 1w0\rangle_{c_{a1},c_{a2};c_v}$
3	001	$\langle -, 1w0; 1w0\rangle_{c_{a1},c_{a2};c_v}$
4	110	$\langle 1w0, 1w0; 0w1\rangle_{c_{a1},c_{a2};c_v}$
5	001	$\langle 0w1, 1w0; 1w0\rangle_{c_{a1},c_{a2};c_v}$
6	101	$\langle 1w0, -; 1w0\rangle_{c_{a1},c_{a2};c_v}$
7	010	$\langle 0w1, 0w1; 0w1\rangle_{c_{a1},c_{a2};c_v}$
8	101	$\langle 1w0, 1w0; 1w0\rangle_{c_{a1},c_{a2};c_v}$
9	011	$\langle 0w1, -; 0w1\rangle_{c_{a1},c_{a2};c_v}$
10	100	$\langle 1w0, 1w0; 1w0\rangle_{c_{a1},c_{a2};c_v}$
11	011	$\langle 0w1, 1w0; 0w1\rangle_{c_{a1},c_{a2};c_v}$

TABLE 6.12: *DBS* de trois cellules SRAM pour les pannes *crCFids* et leurs pannes correspondantes.

n'augmente pas le nombre de lignes dans le tableau. Le tableau 6.13 donne les *DBS* des pannes *crCFids* avec les opérations à appliquer. Le nombre de *DBS* est $d = 12$ et la complexité $\mathbf{O}(n) = 19$, indépendamment de la taille du mot.

#	DBS $c_0c_1c_2\ldots c_{31}$	Opération
0	00000000000000000000000000000000	-
1	11111111111111111111111111111111	w, r
2	00000000000000000000000000000000	w, r
3	00100100100100100100100100100100	w
4	11011011011011011011011011011011	w, r
5	00100100100100100100100100100100	w, r
6	10110110110110110110110110110110	w
7	01001001001001001001001001001001	w, r
8	10110110110110110110110110110110	w, r
9	01101101101101101101101101101101	w
10	10010010010010010010010010010010	w, r
11	01101101101101101101101101101101	w, r

TABLE 6.13: *DBS* pour 32 cellules SRAM pour les pannes *crCFids*.

Pour les pannes *crCFdsts*, il y a 16 sous-types : $\langle w_x, w_y; \updownarrow\rangle c_{a1}, c_{a2}, c_v$ et $\langle r_x, r_y; \updownarrow\rangle c_{a1}, c_{a2}, c_v$, avec $x, y \in \{0, 1\}$ et $\updownarrow$ désigne une transition $1w0$ ou $0w1$. D'une manière similaire à la sec-

$\{\Updownarrow_0 (w00\ldots00); \Uparrow_1 (r00\ldots00, w11\ldots11); \Uparrow_2 (r11\ldots11, w00\ldots00); \Downarrow_3 (r00\ldots00, w11\ldots11); \Downarrow_4 (r11\ldots11, w00\ldots00); \Downarrow_5 (r00\ldots00, w11\ldots11); \Uparrow_6 (r11\ldots11, w00\ldots00); \Downarrow_7 (r00\ldots00, w0010\ldots0010, w1101\ldots1101); \Uparrow_8 (r1101\ldots1101, w0010\ldots0010); \Downarrow_9 (r0010\ldots0010, w1011\ldots10111, w0100\ldots0100); \Updownarrow_{10} (r0100\ldots0100, w1011\ldots1011) \Updownarrow_{11} (r1011\ldots1011, w0110\ldots0110, w1001\ldots1001); \Uparrow_{12} (r1001\ldots1001, w0110\ldots0110) ; \Updownarrow_{13} (r0110\ldots0110)\}$

FIGURE 6.10: Test *WOM* pour les pannes de couplages *crCFids* pour un mot de 32-bits, basé sur le test *March C-*.

tion 6.4.2, nous pouvons construire les *DBOS* pour sensibiliser et détecter toutes les pannes *crCFdsts*. Le tableau 6.14 montre les *DBOS* pour les pannes *crCFdsts*. Le nombre des *DBs* $d = 12$ et la complexité $\mathbf{O}(n) = 27$, ces deux critères de performance étant indépendants de la taille du mot.

#	DB $c_0c_1c_2c_3\ldots c_{31}$	Op.	#	DB $c_0c_1c_2c_3\ldots c_{31}$	Op.
0	0000 ... 0		14	0100 ... 0	w
1	1111 ... 1	w	15	1011 ... 1	w
2	1111 ... 1	r	16	1011 ... 1	r
3	1111 ... 1	r	17	1011 ... 1	r
4	0000 ... 0	w	18	0100 ... 0	w
5	0000 ... 0	r	19	0100 ... 0	r
6	0000 ... 0	r	20	0100 ... 0	r
7	0010 ... 0	w	21	0110 ... 0	w
8	1101 ... 1	w	22	1001 ... 0	w
9	1101 ... 0	w	23	1001 ...	r
10	1101 ... 0	r	24	1001 ... 0	r
11	0010 ... 0	w	25	0110 ... 1	w
12	0010 ... 1	r	26	0110 ... 1	r
13	0010 ... 0	r	27	0110 ... 1	r

TABLE 6.14: *DBOS* pour 32 cellules SRAM pour les pannes *crCFdsts*.

Les tests March *WOM*, basés sur le test March LR, peuvent s'appliquer aux pannes *rCFdsts* et aux pannes *crCFdsts*. La figure 6.8 montre la version *WOM* du test March LR pour la détection des *rCFdsts*. Les éléments M_0 jusqu'à M_3, avec l'opération $r\underbrace{00\ldots00}_{32\text{ bits}}$ de M_4, représentent le test inter-mots. Les éléments M_4 jusqu'à M_{10} représentent le test optimisé pour les pannes *rCFdsts* intra-mots. Le tableau 6.14 présente les *DBOS* utilisés pour la

$\{\Updownarrow_0 (w00\ldots00); \Downarrow_1 (r00\ldots00, w11\ldots11); \Uparrow_2 (r11\ldots11, w00\ldots00, r00\ldots00, r00\ldots00, w11\ldots11); \Uparrow_3 (r11\ldots11, w00\ldots00); \Uparrow_4 (r00\ldots00, w11\ldots11, r11\ldots11, r11\ldots11, w00\ldots00); \Uparrow_5 (r00\ldots00, w11\ldots11, r11\ldots11); \Downarrow_6 (r11\ldots11, w00\ldots00, r00\ldots00); \Downarrow_7 (r00\ldots00, w0010\ldots0010, w1101\ldots1101, r1101\ldots1101); \Downarrow_8 (r1101\ldots1101, w0010\ldots0010, r0010\ldots0010); \Uparrow_9 (r0010\ldots0010, w0100\ldots0100, w1011\ldots1011, r1011\ldots1011); \Downarrow_{10} (r1011\ldots1011, w0100\ldots0100), r0100\ldots0100); \Updownarrow_{11} (r0100\ldots0100, w1001\ldots1001, r1001\ldots1001); \Downarrow_{12} (r1001\ldots1001, w0110\ldots0110, r0110\ldots0110); \Updownarrow_{13} (r0110\ldots0110)\}$

FIGURE 6.11: Test *WOM* pour les pannes de couplage *crCFdsts* pour un mot de 32-bits, basé sur le test *March LR*.

détection des pannes *rCFdsts*. La figure 6.11 montre la version *WOM* du test March LR pour détecter les pannes *crCFdsts*. Les éléments M_5 à M_{13} représentent le test intra-mots pour les *crCFdsts*. La complexité est : $\mathbf{O}(n) = 42 * n/B$.

6.6 Implémentation des tests *March WOM*

L'IP XPS HWICAP [143] permet au processeur PowerPC440 [148] embarqué dans le FPGA Virtex-5 XC5VFX70T d'effectuer des opérations de lecture et d'écriture dans la mémoire CMA via le port ICAP. Il est possible d'écrire des programmes pour ce processeur qui modifient la structure du circuit implémenté dans le FPGA. Le XPS HwICAP contient un support pour la lecture et la modification des CLBs, des LUTs et des bascules DFFs. Le XPS HwICAP est caractérisé par :

- PLB v4.6 basé sur l'interface PLB
- Possibilité de charger un bitstream partiel
- Permet la lecture/écriture des CLBs LUTs
- Permet la lecture/écriture des DFFs des CLBs
- Le port ICAP fonctionne à sa fréquence maximale de 100 MHz
- Supporte les Microblazes et le PowerPC,
- Existe pour les familles Virtex-4, Virtex-5, Virtex-6, Virtex-7 et Spartan-6

Le contrôleur XPS HwICAP fournit l'interface nécessaire pour le transfert des bitstreams vers et depuis le port ICAP. Le XPS HwICAP envoie les données du bitstream directement de la mémoire de sauvegarde du bitstraem. Les nouvelles données sont sauvegardées dans une mémoire FIFO à partir de laquelle les données sont transférées au ICAP. Tous les bitstreams doivent être sauvegardés dans une mémoire (mémoire principale) avant qu'ils soient utilisés pour la reconfiguration du FPGA. Tous les bitstreams requis sont chargés dans la mémoire principale lorsque le système démarre. Le XPS HwICAP permet aussi l'opération *Readback* des états de configuration. Dans ce cas, le *readback* des trames se fait dans une mémoire FIFO de lecture, une trame à la fois. De plus, le XPS HwICAP doit être

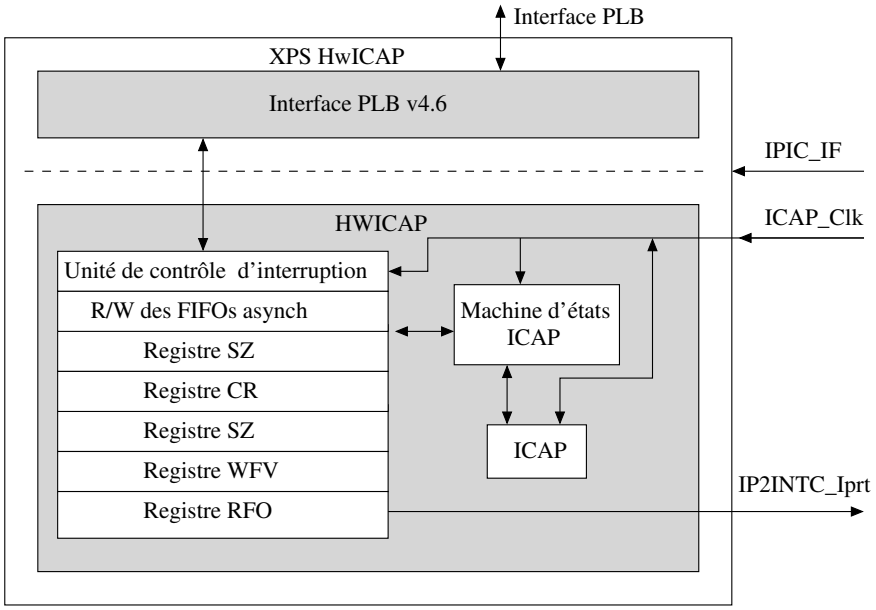


FIGURE 6.12: Schéma du diagramme du core XPS HwICAP.

capable de lire une trame de configuration directement dans la mémoire FIFO de lecture. La figure 6.12- adaptée de [143]- montre le schéma du diagramme du bloc XPS HwICAP.

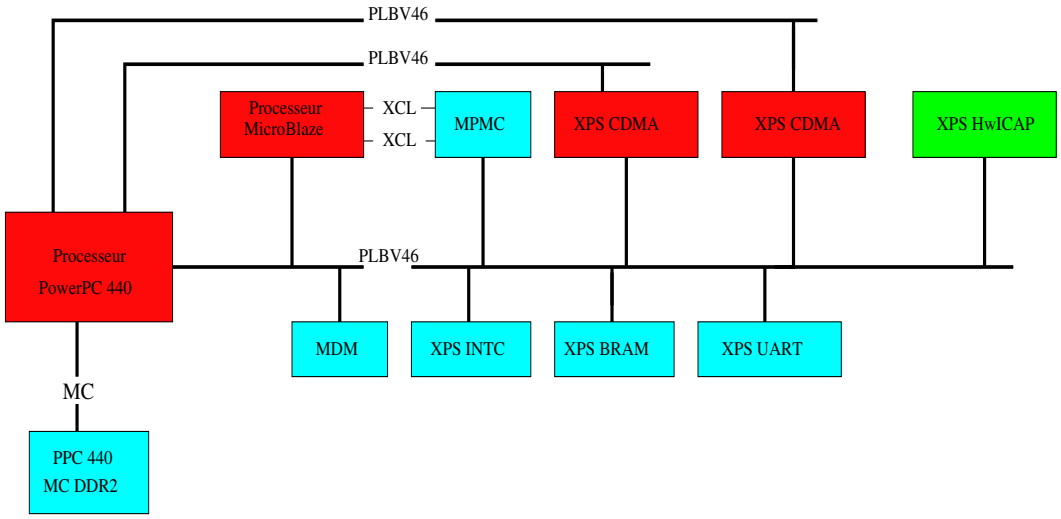


FIGURE 6.13: Le FPGA à base de mémoire Virtex-5 avec le core XPS HwICAP.

La figure 6.13 - adaptée de [143]- montre le matériel pour l'implémentation de la technique BIST proposée.
Le tableau A.1 montre qu'il y a des trames qui ne sont pas configurables. Ce détail n'était pas pris en compte dans [55] pour la configuration de la mémoire CMA.

La procédure pour écrire les tests March dans les trames de configuration dans le contexte de la méthode BIST proposée est illustrée dans le pseudo-code de la figure 6.14.

```
Write_Configuration_Frame (March_Test, FRAME_ADDR){
        Write to command RESET_CRC
        Write to ID Register DEVICE_ID
        Write to command WCFG WRITE_CONFIG_MEM
        Write to Frame Address FAR FRAME_ADDR
        Write to Frame Data Input FDRI 82 words
        for (i = 0; i < 41; i++){
        Write word(i) of March_Test
        }
}
```

FIGURE 6.14: Écriture dans une trame de configuration via l'interface ICAP.

6.7 Discussion

Ce chapitre propose un ensemble systématique de modèles de pannes de couplage intra-mot, basé sur les modèles *CFsts*, *CFids* et *CFdsts*. Selon les restrictions appliquées sur la cellule *agresseur*, on distinguent les sous-types suivants : *uCFs* (pas de restriction), *rCFs* (une seule cellule *agresseur* qui est la cellule voisine physique de la cellule SRAM *victime*) et *crCFs* (deux cellules SRAM *agresseurs* qui sont les cellules SRAM voisines de la cellule *victime*).

Le tableau 6.15 - adapté de [137] - donne les équations pour d, la complexité ainsi que les valeurs de la paire "$d, \mathbf{O}(n)$" pour un mot d'une trame de configuration, en fonction des types de panne *CFs* intra-mots. Pour les valeurs de la complexité, on suppose que les tests *BOM* ont déjà écrit et lu tous les 0*s* et tous les 1*s* de *DB*.

Type de *CF*	# de *DBs* :d	Complexété $O(n)$	d ;$O(n)$
uCFst	12	20	12;20
rCFst	4	4	4;4
crCFst	8	12	8;12
uCFid	18	34	18;34
rCFid	6	10	6;10
crCFid	12	22	12;22
uCFid	18	41	18;41
rCFid	6	13	6;13
crCFid	12	27	12;27

TABLE 6.15: Nombre des *DBs* et complexité des différents types de panne de couplage *CFs*.

La méthode classique pour la conversion d'un test *BOM* en un test *WOM* [68, 144, 149] se fait par la répétition du test *BOM* pour chaque paire complémentaire des valeurs de *DB*. Cette méthode est applicable seulement pour le modèle de panne *uCFst* [137]. En

outre, la méthode classique n'est pas fiable car le test BOM est conçu pour des pannes dans des cellules SRAM uniques et pour des pannes inter-mots. Ces tests sont répétés $d/2$ fois de telle sorte que le temps de test devienne "$(TempsdetestBOM) * d/2$". Cette méthode ne peut pas détecter les pannes intra-mots telles que les pannes *CFids* et *CFdsts* parce qu'elles nécessitent respectivement l'utilisation des séquences de *DBs* (*DBS*) ou une séquence d'opérations *DB* (*DBOS*).

Les problèmes rencontrés sont le temps de test inefficace et la couverture des pannes de couplage intra-mots. Pour y faire face et en s'appuyant sur la méthode présentée dans [137], la concaténation d'un test March inter-mot et d'un test March intra-mots permet le temps de test suivant : "(Temps de test *BOM*)+(Temps de test pour les *CFs* intra-mots)". Cette méthode permet en outre la distinction entre les différents modèles de panne.

L'écriture des séquences DBs dans les trames de configuration pourrait causer des défauts dans les cellules SRAM de la trame. En effet, l'écriture des DBs dans les trames de configuration sans placement ni routage peut créer des mauvaises connexions, ce qui pourrait entraîner une destruction des cellules SRAM. Le challenge est donc de trouver une technique permettant d'appliquer les tests *March WOM* sans mettre en danger les cellules SRAM de la mémoire CMA.

Conclusions du chapitre :

Dans la lignée directe du chapitre 3, ce chapitre est dévolu à la modélisation des pannes intra-mot dans les trames de la mémoire de configuration d'un FPGA de technologie SRAM. Le chapitre 6 présente aussi un ensemble de tests March orientés mots permettant la détection de ce type de pannes. En comparaison aux méthodes de détection de panne dans les FPGAs à base de mémoire SRAM publiées dans la littérature, l'originalité de la méthodologie proposée réside dans :

1. La formalisation du problème de la détection et le diagnostic comme une technique BIST se basant sur un ensemble de tests March
2. Un modèle de performance pour analyser l'algorithme de technique BIST proposée

Sans perdre de vue le cadre applicatif de la détection des pannes dans la mémoire CMA, cette méthode permet d'isoler les cellules SRAM défectueuses et d'utiliser partiellement les ressources logiques défectueuses. Enfin, l'utilisation de cette technique ouvre la voie à la conception des méthodes de tolérance aux pannes rigoureuses pour les FPGAs de technologie SRAM.

Synthèse, conclusion et perspectives ouvertes

7

« *Une suite de petites volontés fait un gros résultat.*»
– Charles Baudelaire, 1821 – 1867

« *Yes, there are two paths you can go by, but in the long run, There's still time to change the road you're on And it makes me wonder*»
– Robert Plant, 1948 – . . .

7.1 Synthèse des travaux présentés

Il ressort de l'état de l'art présenté dans les chapitres 2 et 3 que dans le domaine des architectures reconfigurables, celui des FPGA-SRAMs, les approches issues de la tolérance aux pannes de ces circuits n'évoluent pas aussi vite que les progrès technologiques que ce domaine connait. De même, les seules techniques de tolérance aux pannes qui ont été utilisées dans ce cadre reposent davantage sur des ressources de rechange dont l'efficacité n'est pas vérifiable (tolérance de plusieurs pannes, consommation en puissance). Les travaux présentés dans ce manuscrit ont été motivés par ce manque général, situé entre la tolérance aux pannes des FPGA-SRAMs et les techniques de test et de détection des pannes (transitoires ou permanentes).

Nous avons présenté dans le chapitre 4 un premier pas vers l'application d'une méthode de tolérance aux pannes visant la tolérance de plusieurs pannes dans plusieurs CLBs, avec un coût minimal en ressources logiques. La technique de partition d'un design en unités MSUs est abordée. L'élément clé de cette approche est la reconfiguration partielle : si la nouvelle configuration implémente la même fonction que l'originale, tout en évitant les ressources défectueuses, le système peut être redémarré. Le principal apport de l'approche proposée réside dans sa capacité à tolérer plusieurs pannes dans des CLBs différents avec un niveau élevé de redondance fourni par un seul CLB. Les principaux résultats obtenus dans ce cadre sont :

1. Les travaux de J. Lach et N. Howard visent à partitionner un design implémenté dans un FPGA-SRAM en des petites partitions *(Tiles)* et à reconfigurer la portion affectée par une panne par des configurations de manière à éviter le composant défectueux dans le tile. Dans la lignée de ces travaux, l'approche proposée peut tolérer plusieurs types de panne (logique ou dans les ressources de routage), avec toutefois un taux plus élevé que l'approche *Tiling*.

2. Lorsque plusieurs pannes sont détectées dans une MSU, donc dans des CLBs, les deux CLBs de l'unité CLB-M se reconfigurent afin de remplacer les ressources défectueuses dans les CLB-Ss.
3. Enfin, dans le cas où le nombre de pannes dans une unité MSU dépasse le nombre de pannes que peut tolérer une unité CLB-M, une approche complémentaire est présentée. Cette dernière permet de reconfigurer une autre unité MSU par la configuration de la fonction exécutée par l'unité MSU défectueuse.

Les objectifs atteints dans le cadre de cette thèse ont été multiples. Dans le chapitre 5, une étude a été développée pour déterminer les bits critiques dans un design implémenté dans un FPGA-SRAM. L'obtention des informations sur les bits critiques dans un design permet de déterminer la partie du design à protéger contre les SEUs et de choisir la technique adéquate pour l'atténuation des effets transitoires. L'implémentation de plusieurs circuits de référence ITC'99 dans le FPGA Virtex-5 ML507 a montré l'incapacité des outils de placement et de routage à utiliser toutes les entrées des 6-LUTS. Ceci montre l'importance de l'identification des bits critiques d'un design implémenté sur un FPGA afin de définir la technique d'atténuation des SEUs adéquate. Sur la lumière de cette étude, nous avons proposé une méthode pour l'atténuation des SEUs basée sur la lecture des trames de configuration contenant les bits critiques du design. La méthode réduit ainsi le temps de test et augmente la fiabilité du système.

Dans le chapitre 6, nous avons étendu l'utilisation des tests *March* pour le test des mémoires au test de la mémoire de configuration du FPGA-SRAM. Pour cela, nous avons adapté plusieurs tests *March* destinés à tester les bits d'une mémoire pour qu'ils puissent tester les mots de 32 bits des trames de configuration de la mémoire CMA. Les modèles de panne ciblés sont ceux de couplage intra-mots. Le test des trames de configuration et la localisation des cellules SRAM défectueuses permettent la réduction des ressources de rechange pour les méthodes de tolérance aux pannes ainsi qu'une augmentation du taux de couverture.

7.2 Enjeu des travaux de cette thèse

Le principal enjeu de cette thèse était de trouver une méthode originale de tolérance aux pannes pour les FPGA-SRAMs. En effet, une technique de tolérance aux pannes qui est en adéquation avec l'architecture des FPGA-SRAMs récents paraissait nécessaire. En profitant de la propriété de la reconfiguration partielle de ces circuits, la première idée était de partitionner un design en des ensembles et de configurer chaque ensemble indépendamment du reste afin de contourner les ressources défectueuses.
De plus, dans le but de réutiliser les ressources défectueuses des FPGA-SRAMs (CLBs et SBs), la seconde idée était de tolérer les cellules SRAM des trames de configuration qui configurent ces ressources. Cette fois-ci la difficulté résidait dans la manière de rendre les trames des configurations flexibles aux changements requis par la technique de tolérance aux pannes proposée.
Enfin, pour avoir des méthodes de tolérance aux pannes robustes, les techniques de tests doivent être associées à ces méthodes afin de localiser les ressources défectueuses. Le défi était alors de trouver une technique BIST qui permettra de tester les trames de configuration en utilisant plusieurs modèles de panne.

Les deux premières propositions ont d'ailleurs donné lieu à plusieurs publications et échanges avec la communauté scientifique : [150]- [153]. Pour la suite du travail, il serait intéressant de fixer plusieurs perspectives visant à améliorer et enrichir les méthodes de test ainsi que la tolérance aux pannes des FPGA-SRAMs. Ces perspectives sont détaillées dans la section suivante.

7.3 Perspectives ouvertes à la suite des travaux menés

À la suite de cette brève synthèse des travaux effectués, il semble raisonnable de se questionner sur les problèmes laissés ouverts et qu'il serait souhaitable d'aborder dans un avenir plus ou moins proche. D'autres perspectives n'ayant pas été abordées dans ce manuscrit peuvent également être présentées. Les pistes de travail ouvertes ont été schématiquement classées suivant les trois domaines de recherche abordés dans le présent mémoire :

- la tolérance aux pannes permanentes (dans les ressources logiques et de routage) des FPGA-SRAMs
- les techniques d'atténuation des effets SEUs dans les circuits FPGA-SRAMs
- les techniques de test et de diagnostic des FPGA-SRAMs, notamment de la mémoire de configuration CMA
- les machines d'émulation qui sont constituées d'un réseau de FPGAs (800 FPGAs de type Virtex-7). Les FPGAs de ces machines fonctionnent en parallèle (division du netlist du prototype à émuler). L'obstacle au bon fonctionnement de ces machines est l'incapacité d'effectuer un routage sur tous les FPGAs du réseau. C'est pourquoi augmenter le taux de routage dans les FPGAs est un point essentiel dans la réduction du temps d'émulation et le coût de développement
- le développement d'une méthode de tolérance aux pannes en ligne qui fonctionnera en parallèle à l'application implémentée dans le FPGA et qui prendra en charge la détection des pannes dans les ressources de routage ainsi que dans les ressources logiques

7.4 Conclusion

Il nous faut enfin souligner que les travaux présentés ouvrent également de nombreuses perspectives aussi bien dans le domaine de la tolérance aux pannes des FPGA-SRAMs que dans le domaine du test et diagnostic de ces circuits ou même, entre les deux, dans le domaine de la sécurité des systèmes à base de FPGA-SRAMs.

"Ma thèse de docteur ingénieur était pour moi l'occasion de goûter un peu à la recherche, mais j'ai attrapé le virus."
– Jean-Claude Laprie Chercheur en informatique 1945–2010

Troisième partie

ANNEXES

Architecture du FPGA Virtex-5

A

La plupart des FPGA-SRAMs récents sont de technologie SRAM, aussi bien pour les ressources de routage que pour les blocs logiques configurables (CLBs).
Un bloc logique configurable est constitué de manière générale de deux *slices* (figure A.1) [35]. Chaque slice est constitué de six tables de correspondance (LUT ou Look-Up-Table), de bascules DFF et d'une chaine de propagation de la retenue. Une LUT (figure A.1) sert à implémenter des fonction logiques ayant généralement 6 entrées et deux sorties. Elle peut toutefois être considérée comme une petite mémoire, un multiplexeur ou un registre à décalage. Le registre à décalage permet de mémoriser un état (machine séquentielle) ou de synchroniser un signal (pipeline). Les blocs logiques configurables sont présents en grand nombre sur la puce (6080 CLBs pour le FPGA Virtex-5 XC5VFX70T) et sont connectés entre eux par des matrices de routage (SBs) configurables [35].
Les densités d'intégration actuelles des FPGA-SRAMs ne permettent plus un routage manuel. C'est donc un outil de placement-routage automatique - du kit ISE de xilinx - qui fait correspondre le schéma logique voulu par le concepteur aux ressources matérielles du FPGA-SRAM. Comme les temps de propagation dépendent de la longueur des liaisons entre cellules logiques et que les algorithmes d'optimisation des P&R ne sont pas déterministes, les performances obtenues dans un FPGA-SRAM sont variables d'un design à l'autre.
Comme la configuration (routage et LUTs) est faite par la configuration des cellules SRAM, il est nécessaire de sauvegarder le design du FPGA-SRAM dans une mémoire non volatile externe, généralement une mémoire Flash série, compatible avec JTAG [35]. Pour éviter le recours à une mémoire externe, certaines technologies utilisent des cellules EEPROM pour la configuration ou une configuration anti-fusibles. La dernier choix n'est toutefois reconfigurable (Technologie OTP).
Quelques fonctionnalités particulières disponibles sur les FPGA-SRAMs de la famille Virtex [35][36] :

- blocs de mémoire supplémentaires (hors des LUT), souvent double-port, parfois avec le mécanisme de FIFO
- multiplieurs câblés (coûteux à implémenter en LUT) et blocs multiplieur-accumulateurs pour des traitements DSP
- cœur de microprocesseur embarqué (hard core) comme par exemple les architectures PowerPC ou ARM
- blocs PLL pour synthétiser ou resynchroniser les horloges
- reconfiguration partielle et dynamique
- chiffrement des données de configuration
- sérialiseurs/désérialiseurs dans les entrées-sorties permettant des liaisons série haut-débit
- impédance contrôlée numériquement dans les entrées-sorties évitant de nombreux composants passifs sur la carte

– couche MAC Ethernet
– couches matérielles

```
Site SLICE_X40Y79 with Comp P1/P4/m_mux0000<17> - b19_map.ncd
```

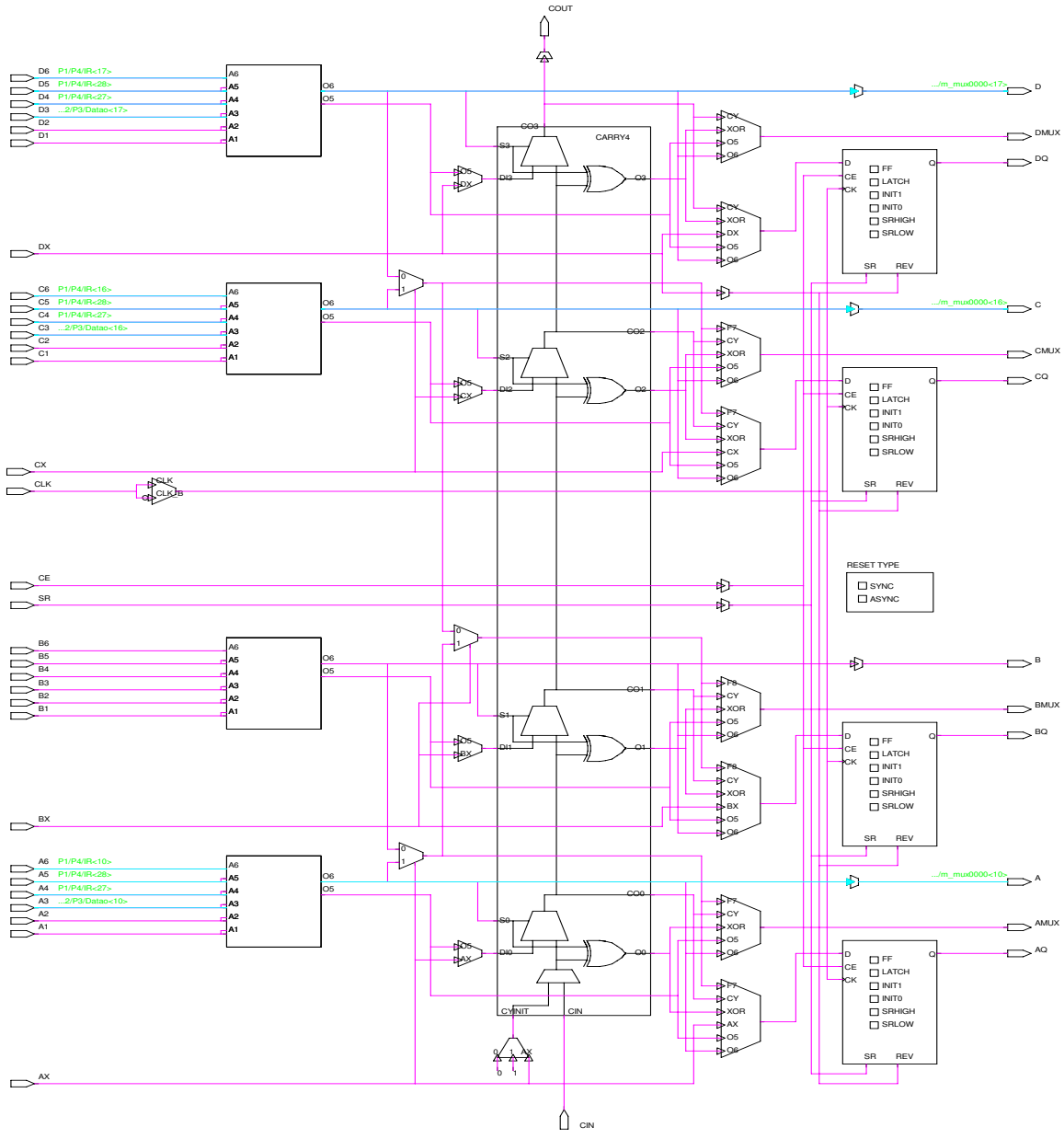


FIGURE A.1: Schéma du slice X40Y79 du FPGA Vertex-5 XC5VFX70T montrant l'utilisation partielle des LUTs.

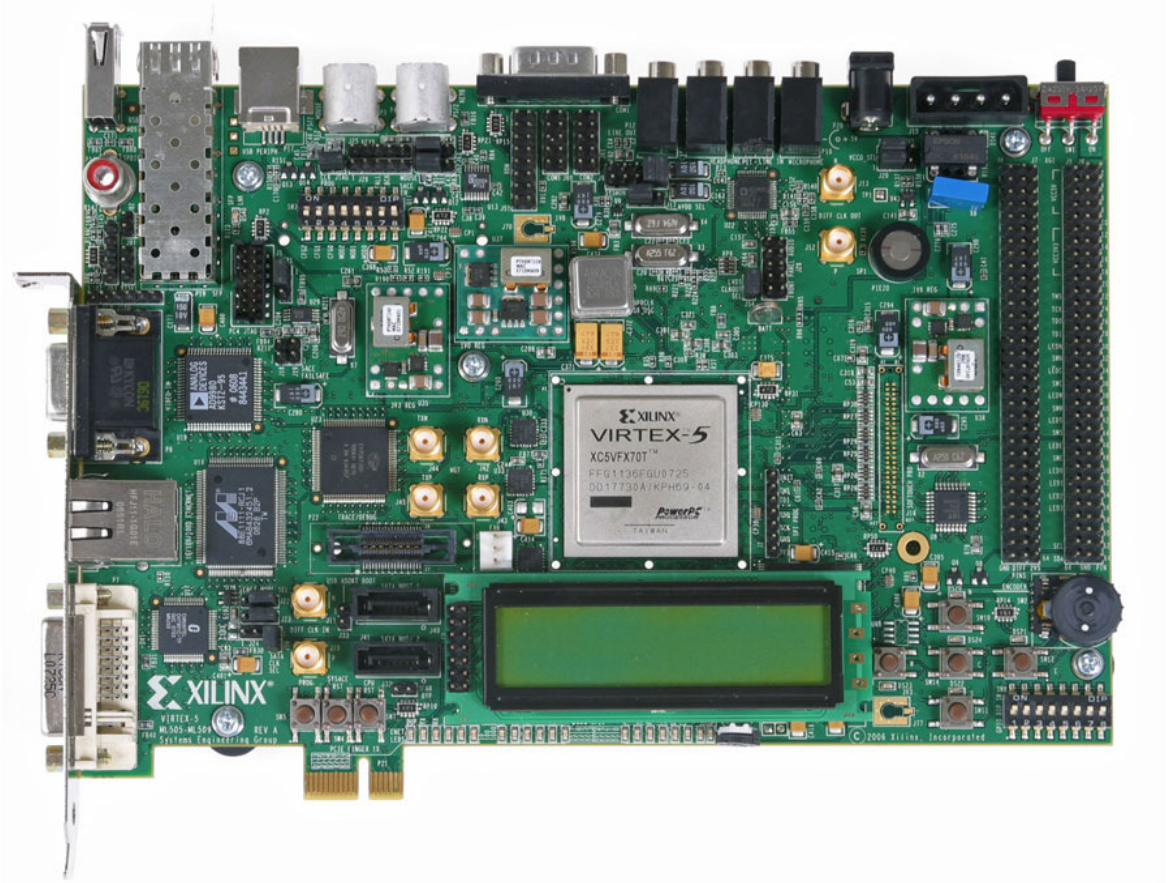


FIGURE A.2: Carte d'évaluation ML507 équipée du FPGA Xilinx Virtex-5 XC5VFX70T utilisée dans les travaux de cette thèse.

La figure A.2 montre la carte ML507 contenant un virtex-5 XC5VFX70T utilisée dans les travaux de cette thèse. Le tableau A.1 montre les caractéristique du FPGA Virtex-5 XC5VFX70T de la carte d'évaluation ML507 [36].

ID Virtex-5	Réseau CLB Rangée × colonne	Trames non-configurables	Trames configurables	# Bits de configuration
XC5VFX70T	160 × 38	488	27864	27025408

TABLE A.1: Caractéristiques du FPGA Virtex-5 XC5VFX70T de la carte d'évaluation ML507.

A.1 Reconfiguration des FPGAs de technologie SRAM

Les FPGA-SRAMs font partie des architectures reconfigurables qui possèdent la capacité de reconfigurer un ou plusieurs sous-ensembles de leurs ressources matérielles. Ces architectures se distinguent des architectures à reconfiguration complète [154]. Les architectures FPGA-SRAMs peuvent être utilisés a des fins d'augmentation de performance en exécutant les traitements intensifs au niveau des ces circuits [155]. La reconfiguration des FPGA-SRAMs se fait suivant les étapes suivantes : Programmation (VHDL, Verilog, Sys-

temC, ...), Synthèse, Mapping, Placement et routage, Configuration. Une grande partie du flot de conception et de programmation des FPGA-SRAMs est commune avec le flot de conception sur ASIC.

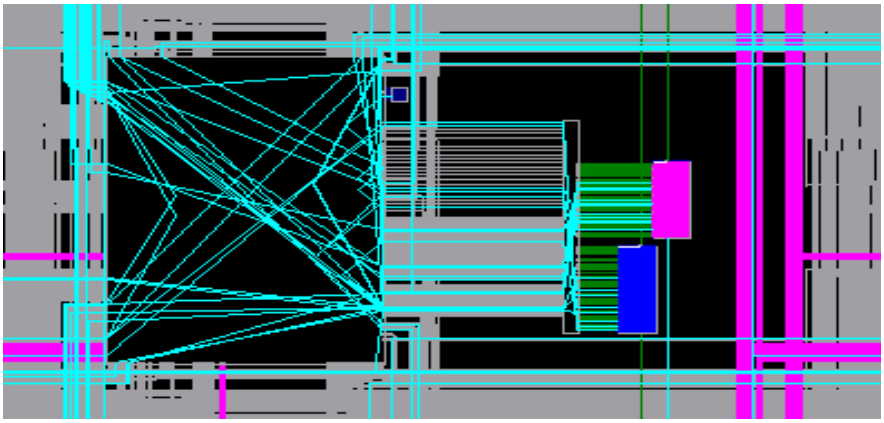


FIGURE A.3: Routage d'une partie d'un design à travers un Switch Bloc (SB).

La figure A.3 montre une partie des ressources d'un FPGA-SRAM utilisée pour le routage d'une partie d'un design-FPGA.

Techniques de Tolérances aux pannes des FPGAs de Technologie SRAM

B

Les FPGA-SRAMs définissent un champ récent qui constitue un compromis entre les champs matériel (ASIC) et logiciel (*µprocesseur*). Parmi les principaux avantages d'un FPGA-SRAM, on trouve la reconfiguration et la reconfiguration partielle dynamique. Un FPGA-SRAM est doté de la capacité d'implémenter des conceptions plus couteux au niveau ressources logiques avec une flexibilité comparable à celle d'une solution ASIC [155]. Les FPGA-SRAMs sont aussi une solution efficace pour le co-design logiciel et matériel [156].

B.1 Tolérance aux pannes des FPGA-SRAMs

La tolérance aux pannes est définie comme étant la capacité d'un système de fonctionner normalement compte tenu de la présence de ressources défectueuses [12]. Un système tolérant aux pannes possède deux axes : *la détection de pannes* et *la réparation ou contournement de pannes*. Les défauts qui peuvent impacter un FPGA-SRAM sont de types différents [157] :

- Défauts causés par la dégradation des composants [158]
- Défauts de fabrication [54]
- Pannes causées par des particules chargées (SEUs)
- ...

Les techniques de tolérance aux pannes hors-ligne interviennent avant qu'une application soit configurée dans un FPGA-SRAM, ce qui conduit à une amélioration du rendement [10, 157, 159]. Cependant, les pannes peuvent aussi avoir lieu après le déploiement du FPGA-SRAM. Les techniques de tolérance aux pannes en ligne devraient donc être couplées avec les techniques de tolérance aux pannes hors-ligne pour fournir un système robuste pour les plateformes FPAG-SRAMs [157].

B.2 Techniques de tolérance aux pannes des FPGA-SRAMs

Les FPGA-SRAMs utilisés pour des applications qui nécessitent une haute fiabilité, ont besoin d'un système tolérant aux pannes.

b.2.1 Techniques BISTs pour les FPGA-SRAMs

Les travaux [76, 56, 160] ont montré que le test BIST d'une petite portion du FPGA peut être réalisé simultanément avec le fonctionnement normal des autres parties [157]. Dans [76, 56], la technique BIST en ligne commence par une zone du FPGA-SRAM puis elle se déplace vers une autre zone de telle sorte que le fonctionnement normal n'est pas affecté.

Cependant, les pannes transitoires (SEUs) qui apparaissent le long du cycle de test ne peuvent pas être détectées.

b.2.2 Technique de duplication et comparaison

La technique de duplication et de comparaison assure la sécurité en ligne contre les pannes dans un FPAG-SRAM. Elle est couramment utilisée dans de nombreux systèmes [161, 162, 163, 157]. Elle nécessite la duplication d'un module fonctionnel, un comparateur et un contrôleur. Les pannes permanentes sont détectées par le maintien de l'historique des résultats passés [157]. Si un module dupliqué génère plusieurs sorties erronées, il doit être séparé du système puis remplacé [157].

b.2.3 Technique de redondance modulaire multiple (NMR)

Dans cette technique, N modules (où N est impair) exécutent la même fonction. Un circuit de vote majoritaire recueille leurs résultats et choisit la valeur majoritaire en tant que résultat final. Ainsi, les pannes qui apparaissent dans la minorité des modules seront masquées [157]. La technique NMR est généralement utilisée sous la version TMR pour limiter le coût matériel [157].

B.3 Comparaison des différentes techniques de tolérance aux pannes

Le Tableau B.1 [157] survole les méthodes et techniques de tolérance aux pannes des FPGA-SRAMs. Il fait des comparaisons entre ces techniques discutées précédemment.

Paramètres FT	**BIST** [76, 56, 160]	**D&C** [161]	**NMR** [164, 161]	**CEC** [161, 165]	**WT** [166]
Mode de vérification	Généralement, les tests son effectués en mode hors ligne. [76, 56] testent les parties du FPGA pendant le fonctionnement.	En ligne, pour la détection des erreurs.	Correction en ligne par circuit de vote majoritaire	Détection en ligne correction avec informations redondantes	En ligne
Possibilité de tolérer les SEUs	Non. Dans [76, 56], le temps de détection est supérieur au temps de reconfiguration	Oui, elle peut détecter les effets transitoires.	Oui, elle peut corriger les pannes transitoires.	Oui, elle peut détecter et corriger les les pannes transitoires	Non
Surcoût A) Fonctions FT ajoutées	Circuit BIST	Circuits de duplication et de comparaison	Triplication et circuit sélecteur	Circuit pour la génération de code vérificateur et code convertisseur	Circuit WT Timer + Système d'interruption (SI)
Surcoût B) besoins de communication	Test de routage, test des data E/S par JTAG.	Entrées des ressources de routages entre modules.	Entrées des routages pour les signaux triplés et sélecteur	Pannes de routage système de récupération	Logiciel de communication entre les correcteurs et modules
Espace mémoire Exigé	Sauvegarde des signatures du circuit comme dans [76, 56].	Aucun	Aucun	CRC est stocké avec les données	Mémoire pour les timers software et SI
Temps additionnel	Temps pour le test hors ligne. Dans [76, 56], temps pour la reconfiguration et STARS.	Temps de comparaison et de propagation des signaux.	Temps de sélection entre les résultats	Temps de génération des valeurs codées rectification d'erreurs	Délai d'attente maximum
Capacité de diagnostic	Pannes de collages s-a-1/0 Taux de couverture élevé	détection deux modules défectueux	Identifie la minorité comme des pannes	détection et localisation des pannes en temps d'exécution	Programmes de diagnostic
Mécanisme de tolérance aux pannes	Contourner les blocs défectueux par reconfiguration des sorties des modules.	Esquive les opérations non sécuritaires par blocage	Les résultats majoritaires sont corrects des sorties des modules.	Esquive les opérations non sécuritaires par blocage	Relancement des modules
Commentaires	Meilleur diagnostic Temps additionnel Dissipation de puissance	Élimine les opérations non sécuritaires. Incapable de distinguer entre les modules défectueux	Redondance élevée surcoût élevé Fiabilité élevée	Redondance dans le codage des données	Utilisable pour les manipulation des pannes logiciels.

TABLE B.1: Tableau de comparaison des méthodes de tolérance aux pannes

Bibliographie

[1] V. D. Agrawal et S. C. Seth, *Tutorial test generation for VLSI chips*. IEEE Computer Society, 1988. (Cité dans la page 1.)

[2] M. Bushnell et V. D. Agrawal, *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*, vol. 17. Springer Science & Business Media, 2000. (Cité dans les pages 1, 30, 81, 83, 85, 87, et 88.)

[3] S. D'Angelo, C. Metra, S. Pastore, A. Pogutz et G. Sechi, « Fault-tolerant voting mechanism and recovery scheme for tmr fpga-based systems », *in Defect and Fault Tolerance in VLSI Systems, 1998. Proceedings., 1998 IEEE International Symposium on*, p. 233–240, IEEE, 1998. (Cité dans la page 2.)

[4] S. D'Angelo, C. Metra et G. Sechi, « Transient and permanent fault diagnosis for fpga-based tmr systems », *in Defect and Fault Tolerance in VLSI Systems, 1999. DFT'99. International Symposium on*, p. 330–338, IEEE, 1999.

[5] G. Mojoli, D. Salvi, M. Sami, G. Sechi et R. Stefanelli, « Kite : a behavioural approach to fault-tolerance in fpga-based systems », *in Defect and Fault Tolerance in VLSI Systems, 1996. Proceedings., 1996 IEEE International Symposium on*, p. 327–334, IEEE, 1996.

[6] P. Samudrala, J. Ramos et S. Katkoori, « Selective triple modular redundancy (stmr) based single-event upset (seu) tolerant synthesis for fpgas », *Nuclear Science, IEEE Transactions on*, vol. 51, no. 5, p. 2957–2969, 2004. (Cité dans la page 67.)

[7] F. Kastensmidt, L. Sterpone, L. Carro et M. Reorda, « On the optimal design of triple modular redundancy logic for sram-based fpgas », *in Proceedings of the conference on Design, Automation and Test in Europe-Volume 2*, p. 1290–1295, IEEE Computer Society, 2005. (Cité dans les pages 2 et 71.)

[8] J. Lee, Y. Hu, R. Majumdar, L. He et M. Li, « Fault-tolerant resynthesis with dual-output luts », *in Proceedings of the 2010 Asia and South Pacific Design Automation Conference*, p. 325–330, IEEE Press, 2010. (Cité dans les pages 2 et 49.)

[9] X. products, « World's Highest Capacity FPGA - Now Shipping (25 October, 2011) », *available at : http ://www.xilinx.com/products/silicon-devices/fpga/virtex-7/.*, 25 October, 2011. (Cité dans les pages 2 et 48.)

[10] N. Campregher, P. Cheung, G. Constantinides et M. Vasilko, « Analysis of yield loss due to random photolithographic defects in the interconnect structure of fpgas », *in Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays*, p. 138–148, ACM, 2005. (Cité dans les pages 3 et 115.)

[11] A. Doumar et H. Ito, « Detecting, diagnosing, and tolerating faults in sram-based field programmable gate arrays : a survey », *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 11, no. 3, p. 386–405, 2003. (Cité dans les pages 3, 31, 32, 38, 42, et 43.)

[12] J. A. Cheatham, J. M. Emmert et S. Baumgart, « A survey of fault tolerant methodologies for fpgas », *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 11, no. 2, p. 501–533, 2006. (Cité dans les pages 28, 29, 30, 31, 38, 40, 42, et 115.)

[13] A. Djupdal et P. Haddow, « Yield enhancing defect tolerance techniques for fpgas », *in MAPLD International Conference*, Citeseer, 2006. (Cité dans les pages 3 et 49.)

[14] W. House, « Cyberspace policy review : Assuring a trusted and resilient information and communications infrastructure », *Washington, DC : The White House Retrieved September*, vol. 3, p. 2009, 2009. (Cité dans la page 3.)

[15] K. Stouffer, J. Falco et K. Scarfone, « Guide to industrial control systems (ics) security », *NIST Special Publication*, vol. 800, p. 82, 2007.

[16] C. Of, H. Ronald, H. James *et al.*, « Security requirements for cryptographic modules », (Cité dans la page 3.)

[17] G. Moore, « Progress in digital integrated ceci est un test electronics », *in Electron ceci est un test Devices Meeting, 1975 International*, vol. 21, p. 11–13, IEEE, 1975. (Cité dans la page 3.)

[18] S. Hauck et A. DeHon, *Reconfigurable computing : the theory and practice of FPGA-based computation*. Morgan Kaufmann, 2010. (Cité dans les pages 9, 11, 12, et 14.)

[19] A. Greenfield, *Everyware : The dawning age of ubiquitous computing*. New Riders, 2010. (Cité dans la page 11.)

[20] S. Pillement, *Conception d'architectures reconfigurables dynamiquement : Du silicium au système*. Thèse de doctorat, Université Rennes 1, 2010. (Cité dans la page 11.)

[21] T. J. Callahan, J. R. Hauser et J. Wawrzynek, « The garp architecture and c compiler », *Computer*, vol. 33, no. 4, p. 62–69, 2000. (Cité dans la page 11.)

[22] L. Bossuet, *Exploration de l'espace de conception des architectures reconfigurables*. Thèse de doctorat, Université de Bretagne Sud, 2004. (Cité dans les pages 12 et 13.)

[23] R. David, D. Chillet, S. Pillement et O. Sentieys, « Dart : a dynamically reconfigurable architecture dealing with future mobile telecommunications constraints », *Résumé*, 2003. (Cité dans la page 12.)

[24] M. I. Gordon, W. Thies et S. Amarasinghe, « Exploiting coarse-grained task, data, and pipeline parallelism in stream programs », *in ACM SIGOPS Operating Systems Review*, vol. 40, p. 151–162, ACM, 2006. (Cité dans la page 12.)

[25] R. Koch, T. Pionteck, C. Albrecht et E. Maehle, « An adaptive system-on-chip for network applications », *in Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, p. 8–pp, IEEE, 2006. (Cité dans la page 12.)

[26] J. M. Rabaey, « Low-power silicon architecture for wireless communications : embedded tutorial », *in Proceedings of the 2000 Asia and South Pacific Design Automation Conference*, p. 377–380, ACM, 2000. (Cité dans la page 13.)

[27] S. Derrien, *Etude quantitative des techniques de partitionnement de réseaux de processeurs pour l'implantation sur circuits FPGA*. Thèse de doctorat, Rennes 1, 2002. (Cité dans la page 13.)

[28] S. Kilts, *Advanced FPGA design : architecture, implementation, and optimization*. John Wiley & Sons, 2007.

[29] U. Farooq, Z. Marrakchi et H. Mehrez, « Tree-based asif using heterogeneous blocks », *in Tree-based Heterogeneous FPGA Architectures*, p. 153–171, Springer, 2012. (Cité dans la page 17.)

[30] I. Kuon, R. Tessier et J. Rose, « Fpga architecture : Survey and challenges », *Foundations and Trends in Electronic Design Automation*, vol. 2, no. 2, p. 135–253, 2008.

[31] D. K. Iakovidis, D. E. Maroulis et D. G. Bariamis, « Fpga architecture for fast parallel computation of co-occurrence matrices », *Microprocessors and Microsystems*, vol. 31, no. 2, p. 160–165, 2007.

[32] S. TRIMBERGER, *Field-programmable gate array technology*. Springer Science & Business Media, 1994. (Cité dans la page 14.)

[33] J. HUANG, M. B. TAHOORI et F. LOMBARDI, « Fault tolerance of switch blocks and switch block arrays in fpga », *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 13, no. 7, p. 794–807, 2005. (Cité dans les pages 14, 16, 17, et 36.)

[34] J. HUANG, M. B. TAHOORI et F. LOMBARDI, « Routability and fault tolerance of fpga interconnect architectures », *in Test Conference, 2004. Proceedings. ITC 2004. International*, p. 479–488, IEEE, 2004. (Cité dans les pages 14 et 17.)

[35] XILINX, « Virtex-5 FPGA Configuration User Guide », *Xilinx User Guide (v5.3) UG190*, vol. 1, 2009. (Cité dans les pages 15, 19, 21, 73, 75, 77, 81, 82, 83, et 111.)

[36] XILINX, « Datasheets Virtex-4, Virtex-5, Virtex-6, Virtex-7. UG070, UG190, UG360, UG470 », *www.xilinx.com*, 2014. (Cité dans les pages 16, 69, 111, et 113.)

[37] Y. AKASAKA et T. NISHIMURA, « Concept and basic technologies for 3-d ic structure », *in Electron Devices Meeting, 1986 International*, vol. 32, p. 488–491, IEEE, 1986. (Cité dans la page 16.)

[38] P. RAMM, A. KLUMPP, R. MERKEL, J. WEBER, R. WIELAND, A. OSTMANN et J. WOLF, « 3d system integration technologies », *in MRS Proceedings*, vol. 766, p. E5–6, Cambridge Univ Press, 2003. (Cité dans la page 16.)

[39] *UltraScale Architecture Configuration*, vol. UG570, Decembre 16. Xilinx, 2016. (Cité dans la page 16.)

[40] A. ULLAH, « Dependable system design for reconfigurable safety-critical applications », 2015. (Cité dans la page 17.)

[41] XILINX, « Embedded Processor Block in Virtex-5 FPGAs », *Xilinx User Guide (v5.3) UG200*, vol. 1, 2010. (Cité dans la page 19.)

[42] XILINX, « Development System Reference Guide », *Xilinx Development System Reference Guide (v10.1)*, vol. 1, 2008. (Cité dans la page 21.)

[43] M. LIU, W. KUEHN, Z. LU et A. JANTSCH, « Run-time partial reconfiguration speed investigation and architectural design space exploration », *in Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, p. 498–502, IEEE, 2009. (Cité dans la page 22.)

[44] « Partial reconfiguration user guide », *UG702 (v14. 5), Xilinx Inc*, 2013. (Cité dans la page 22.)

[45] P. R. U. GUIDE, « Ug702 (v12. 3), xilinx », *Inc., October*, vol. 5, 2010. (Cité dans les pages 23 et 24.)

[46] K. B. CHEHIDA, *Méthodologie de partitionnement logiciel/matériel pour plateformes reconfigurables dynamiquement*. Thèse de doctorat, Université Nice Sophia Antipolis, 2004. (Cité dans la page 24.)

[47] K. VIPIN et S. A. FAHMY, « Architecture-aware reconfiguration-centric floorplanning for partial reconfiguration », *in Reconfigurable Computing : Architectures, Tools and Applications*, p. 13–25, Springer, 2012. (Cité dans la page 25.)

[48] J.-B. NOTE et É. RANNAUD, « From the bitstream to the netlist. », *in FPGA*, vol. 8, p. 264–264, 2008. (Cité dans la page 25.)

[49] C. BECKHOFF, D. KOCH et J. TORRESEN, « Portable module relocation and bitstream compression for xilinx fpgas », *in Field Programmable Logic and Applications (FPL), 2014 24th International Conference on*, p. 1–8, IEEE, 2014. (Cité dans la page 25.)

[50] E. Eto, « Difference-based partial reconfiguration », 2003. (Cité dans la page 25.)

[51] D. Lardner, « Babbage's calculating engine », *Edinburgh Review*, vol. 59, no. 120, p. 263–327, 1834. (Cité dans la page 27.)

[52] A. Avižienis, J.-C. Laprie et B. Randell, « Dependability and its threats : a taxonomy », *in Building the Information Society*, p. 91–120, Springer, 2004. (Cité dans les pages 27 et 31.)

[53] M. Wirthlin, D. Lee, G. Swift et H. Quinn, « A method and case study on identifying physically adjacent multiple-cell upsets using 28-nm, interleaved and secded-protected arrays », 2014. (Cité dans la page 28.)

[54] J. Lach, W. Mangione-Smith et M. Potkonjak, « Low overhead fault-tolerant FPGA systems », *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 6, no. 2, p. 212–221, 1998. (Cité dans les pages 30, 31, 40, 43, 49, 50, 58, et 115.)

[55] S. Drimer, « Total configuration memory cell validation built in self test (bist) circuit », août 5 2008. (Cité dans les pages 30, 33, 34, 80, 87, et 102.)

[56] J. M. Emmert, C. E. Stroud et M. Abramovici, « Online fault tolerance for fpga logic blocks », *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 15, no. 2, p. 216–226, 2007. (Cité dans les pages 30, 43, 55, 115, et 117.)

[57] I. Koren et Z. Koren, « Defect tolerance in vlsi circuits : techniques and yield analysis », *Proceedings of the IEEE*, vol. 86, no. 9, p. 1819–1838, 1998. (Cité dans les pages 31, 32, et 47.)

[58] A. Doumar et H. Ito, « Fault tolerance fpgas by shifting the configuration data », *in International Conference on Military and Aerospace Programmable Logic Devices*, p. 377–384. (Cité dans les pages 31 et 38.)

[59] J. Lach, W. H. Mangione-Smith et M. Potkonjak, « Algorithms for efficient runtime fault recovery on diverse fpga architectures », *in Defect and Fault Tolerance in VLSI Systems, 1999. DFT'99. International Symposium on*, p. 386–394, IEEE, 1999. (Cité dans les pages 31, 40, 49, et 50.)

[60] R. Amerson, R. Carter, W. Culbertson, P. Kuekes, G. Snider et L. Albertson, « Plasma : an fpga for million gate systems », *in Field-Programmable Gate Arrays, 1996. FPGA'96. Proceedings of the 1996 ACM Fourth International Symposium on*, p. 10–16, IEEE, 1996.

[61] W. B. Culbertson, R. Amerson, R. J. Carter, P. Kuekes et G. Snider, « Defect tolerance on the teramac custom computer », *in Field-Programmable Custom Computing Machines, 1997. Proceedings., The 5th Annual IEEE Symposium on*, p. 116–123, IEEE, 1997. (Cité dans la page 31.)

[62] R. LEVEUGLE, « Test des circuits intégrés numériques : Conception orientée testabilité », *Techniques de l'ingénieur. Electronique*, vol. 2, no. E2461, p. E2461–1, 2002. (Cité dans les pages 32, 40, et 79.)

[63] M. Dubois, *Méthodologie d'estimation des métriques de test appliquée à une nouvelle technique de BIST de convertisseur SIGMA/DELTA*. Thèse de doctorat, Université de Grenoble, 2011. (Cité dans les pages 33 et 37.)

[64] S. Hamdioui et A. J. Van de Goor, « An experimental analysis of spot defects in srams : realistic fault models and tests », *in Test Symposium, 2000.(ATS 2000). Proceedings of the Ninth Asian*, p. 131–138, IEEE, 2000. (Cité dans la page 35.)

[65] S. Hamdioui, Z. Al-Ars et A. J. Van de Goor, « Testing static and dynamic faults in random access memories », *in VLSI Test Symposium, 2002.(VTS 2002). Proceedings 20th IEEE*, p. 395–400, IEEE, 2002.

[66] S. Hamdioui, G. Gaydadjiev et A. J. Van de Goor, « The state-of-art and future trends in testing embedded memories », *in Memory Technology, Design and Testing, 2004. Records of the 2004 International Workshop on*, p. 54–59, IEEE, 2004.

[67] S. Hamdioui, *Testing static random access memories : defects, fault models and test patterns*, vol. 26. Springer Science & Business Media, 2004. (Cité dans les pages 67 et 98.)

[68] A. Van de Goor, *Testing semiconductor memories : theory and practice*. John Wiley & Sons, Inc., 1991. (Cité dans les pages 79, 85, 89, 95, et 103.)

[69] C. Papameletis, B. Keller, V. Chickermane, S. Hamdioui et E. J. Marinissen, « A dft architecture and tool flow for 3d-sics with test data compression, embedded cores, and multiple towers »,

[70] S. Hamdioui, Z. Al-Ars, A. J. Van De Goor et M. Rodgers, « Dynamic faults in random-access-memories : Concept, fault models and tests », *Journal of Electronic Testing*, vol. 19, no. 2, p. 195–205, 2003.

[71] H. Kukner, S. Khan, P. Weckx, P. Raghavan, S. Hamdioui, B. Kaczer, F. Catthoor, L. Van der Perre, R. Lauwereins et G. Groeseneken, « Comparison of reaction-diffusion and atomistic trap-based bti models for logic gates », *Device and Materials Reliability, IEEE Transactions on*, vol. 14, no. 1, p. 182–193, 2014. (Cité dans la page 35.)

[72] S. Dutt, V. Verma et V. Suthar, « Built-in-self-test of fpgas with provable diagnosabilities and high diagnostic coverage with application to online testing », *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 2, p. 309–326, 2008. (Cité dans les pages 35 et 43.)

[73] V. Verma, S. Dutt et V. Suthar, « Efficient on-line testing of fpgas with provable diagnosabilities », *in Proceedings of the 41st annual Design Automation Conference*, p. 498–503, ACM, 2004. (Cité dans la page 35.)

[74] S. Dutt, V. Shanmugavel et S. Trimberger, « Efficient incremental rerouting for fault reconfiguration in field programmable gate arrays », *in Computer-Aided Design, 1999. Digest of Technical Papers. 1999 IEEE/ACM International Conference on*, p. 173–176, IEEE, 1999. (Cité dans la page 35.)

[75] N. R. Mahapatra et S. Dutt, « Efficient network-flow based techniques for dynamic fault reconfiguration in fpgas », *in Fault-Tolerant Computing, 1999. Digest of Papers. Twenty-Ninth Annual International Symposium on*, p. 122–129, IEEE, 1999. (Cité dans la page 35.)

[76] M. Abramovici, C. Strond, C. Hamilton, S. Wijesuriya et V. Verma, « Using roving stars for on-line testing and diagnosis of fpgas in fault-tolerant applications », *in Test Conference, 1999. Proceedings. International*, p. 973–982, IEEE, 1999. (Cité dans les pages 35, 115, et 117.)

[77] M. Abramovici, C. E. Stroud et J. M. Emmert, « Online bist and bist-based diagnosis of fpga logic blocks », *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 12, no. 12, p. 1284–1294, 2004. (Cité dans la page 36.)

[78] M. Abramovici, C. Stroud, B. Skaggs et J. Emmert, « Improving on-line bist-based diagnosis for roving stars », *in On-Line Testing Workshop, 2000. Proceedings. 6th IEEE International*, p. 31–39, IEEE, 2000. (Cité dans la page 36.)

[79] M. Renovell, J. M. Portal, J. Figueras et Y. Zorian, « Testing the interconnect of ram-based fpgas », *IEEE Design & Test of Computers*, no. 1, p. 45–50, 1998. (Cité dans la page 36.)

[80] A. Doumar, S. Kaneko et H. Ito, « Defect and fault tolerance fpgas by shifting the configuration data », *in Defect and Fault Tolerance in VLSI Systems, 1999. DFT'99. International Symposium on*, p. 377–385, IEEE, 1999. (Cité dans la page 38.)

[81] W. K. Huang, F. J. Meyer, X.-T. Chen et F. Lombardi, « Testing configurable lut-based fpga's », *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 6, no. 2, p. 276–283, 1998. (Cité dans la page 38.)

[82] K. A. Kwiat, « Dynamically reconfigurable fpga apparatus and method for multiprocessing and fault tolerance », août 3 1999. US Patent 5,931,959. (Cité dans les pages 41 et 43.)

[83] K. Kwiat et S. Hariri, « Efficient hardware fault tolerance using field-programmable gate arrays », *in Proceedings ISSAT International Conference on Reliability and Quality in Design*, p. 59–64, Citeseer, 1995. (Cité dans la page 41.)

[84] A. Sudarsanam, R. Kallam et A. Dasu, « Prr-prr dynamic relocation », *Computer Architecture Letters*, vol. 8, no. 2, p. 44–47, 2009. (Cité dans les pages 41 et 42.)

[85] A. Sreeramareddy, J. G. Josiah, A. Akoglu et A. Stoica, « Scars : Scalable self-configurable architecture for reusable space systems », *in Adaptive Hardware and Systems, 2008. AHS'08. NASA/ESA Conference on*, p. 204–210, IEEE, 2008. (Cité dans la page 42.)

[86] A. Sudarsanam, R. Barnes, J. Carver, R. Kallam et A. Dasu, « Dynamically reconfigurable systolic array accelerators : A case study with extended kalman filter and discrete wavelet transform algorithms », *IET computers & digital techniques*, vol. 4, no. 2, p. 126–142, 2010. (Cité dans la page 42.)

[87] S. Corbetta, M. Morandi, M. Novati, M. D. Santambrogio, D. Sciuto et P. Spoletini, « Internal and external bitstream relocation for partial dynamic reconfiguration », *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 17, no. 11, p. 1650–1654, 2009. (Cité dans la page 42.)

[88] J. Heiner, B. Sellers, M. Wirthlin et J. Kalb, « Fpga partial reconfiguration via configuration scrubbing », *in Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, p. 99–104, IEEE, 2009. (Cité dans la page 42.)

[89] M. Straka, J. Kastil et Z. Kotasek, « Fault tolerant structure for sram-based fpga via partial dynamic reconfiguration », *in Digital System Design : Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on*, p. 365–372, IEEE, 2010.

[90] H. Tan et R. F. DeMara, « A physical resource management approach to minimizing fpga partial reconfiguration overhead », *in Reconfigurable Computing and FPGA's, 2006. ReConFig 2006. IEEE International Conference on*, p. 1–5, IEEE, 2006.

[91] H. Tan et R. F. DeMara, « A multilayer framework supporting autonomous run-time partial reconfiguration », *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 16, no. 5, p. 504–516, 2008. (Cité dans la page 42.)

[92] W. R. Moore, « A review of fault-tolerant techniques for the enhancement of integrated circuit yield », *Proceedings of the IEEE*, vol. 74, no. 5, p. 684–698, 1986. (Cité dans la page 42.)

[93] K. Morgan, D. McMurtrey, B. Pratt et M. Wirthlin, « A comparison of tmr with alternative fault-tolerant design techniques for fpgas », *Nuclear Science, IEEE Transactions on*, vol. 54, no. 6, p. 2065–2072, 2007. (Cité dans les pages 43 et 50.)

[94] N. J. Howard, A. M. Tyrrell et N. M. Allinson, « The yield enhancement of field-programmable gate arrays », *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 2, no. 1, p. 115–123, 1994. (Cité dans la page 43.)

[95] F. HANCHEK et S. DUTT, « Node-covering based defect and fault tolerance methods for increased yield in fpgas », *in VLSI Design, 1996. Proceedings., Ninth International Conference on*, p. 225–229, IEEE, 1996.

[96] M. TAOUIL, S. HAMDIOUI, K. BEENAKKER et E. MARINISSEN, « Test impact on the overall die-to-wafer 3d stacked ic cost », *Journal of Electronic Testing*, p. 1–11, 2011. (Cité dans la page 48.)

[97] X. PRODUCTS, « Virtex-5 FPGA user guide - Now Shipping (May 17, 2010) », *available at : http ://www.xilinx.com.*, May 17, 2010. (Cité dans la page 51.)

[98] M. GIRAUD, « Sûreté de fonctionnement des systèmes : Principes et définitions », *Techniques de l'ingénieur. Electronique*, vol. 5, no. E3850, 2005. (Cité dans la page 57.)

[99] A. AVIŽIENIS, J. LAPRIE et B. RANDELL, « Dependability and its threats : a taxonomy », *Building the Information Society*, p. 91–120, 2004. (Cité dans la page 57.)

[100] F. KASTENSMIDT, L. CARRO et R. da LUZ REIS, *Fault-tolerance Techniques for SRAM-based FPGAs*, vol. 32. Springer Verlag, 2006. (Cité dans la page 66.)

[101] S. BONACINI, F. FACCIO, K. KLOUKINAS et A. MARCHIORO, « An seu-robust configurable logic block for the implementation of a radiation-tolerant fpga », *Nuclear Science, IEEE Transactions on*, vol. 53, no. 6, p. 3408–3416, 2006. (Cité dans la page 67.)

[102] R. LYONS et W. VANDERKULK, « The use of triple-modular redundancy to improve computer reliability », *IBM Journal of Research and Development*, vol. 6, no. 2, p. 200–209, 1962. (Cité dans la page 67.)

[103] M. BERG, H. KIM, M. FRIENDLICH, C. PEREZ, C. SEIDLECK, K. LABEL et R. LADBURY, « Seu analysis of complex circuits implemented in actel rtax-s fpga devices », *Nuclear Science, IEEE Transactions on*, no. 99, p. 1–1, 2011. (Cité dans la page 67.)

[104] G. FOUCARD, *Taux dérreurs dues aux radiations pour des applications implémentées dans des FPGAs à base de mémoire SRAM : prédiction versus mesures*. Thèse de doctorat, Université de Grenoble Institut Polytechnique de Grenoble, Juin 2010. (Cité dans les pages 67 et 69.)

[105] H. ASADI, M. B. TAHOORI, B. MULLINS, D. KAELI et K. GRANLUND, « Soft error susceptibility analysis of sram-based fpgas in high-performance information systems », *IEEE Transactions on Nuclear Science*, vol. 54, no. 6, p. 2714–2726, 2007. (Cité dans la page 67.)

[106] E. NORMAND, « Correlation of inflight neutron dosimeter and seu measurements with atmospheric neutron model », *Nuclear Science, IEEE Transactions on*, vol. 48, no. 6, p. 1996–2003, 2001. (Cité dans la page 67.)

[107] « Actel datasheet :"rtax-s/sl radtolerant fpgas"[online] disponible : », Octobre, 2007. http://www.actel.com/documents/RTAXS_DS.pdf. (Cité dans la page 67.)

[108] M. BEN JIRAD, *Robustesse par conception de circuits implantés sur FPGA SRAM et validation par injection de fautes*. Thèse de doctorat, Grenoble, 2013. (Cité dans la page 67.)

[109] F. G. de LIMA et R. A. da LUZ REIS, « Designing single event upset mitigation techniques for large sram-based fpga devices », 2001. (Cité dans la page 67.)

[110] X. SHE et S. TRIMBERGER, « Scheme to minimise short effects of single-event upsets in triple-modular redundancy », *IET computers & digital techniques*, vol. 4, no. 1, p. 50–55, 2010. (Cité dans les pages 68, 71, et 72.)

[111] H. QUINN, G. ALLEN, G. SWIFT, C. TSENG, P. GRAHAM, K. MORGAN et P. OSTLER, « Seu-susceptibility of logical constants in xilinx fpga designs », *Nuclear Science, IEEE Transactions on*, vol. 56, no. 6, p. 3527–3533, 2009. (Cité dans la page 69.)

[112] M. Berg, C. Poivey, D. Petrick, D. Espinosa, A. Lesea, K. LaBel, M. Friendlich, H. Kim et A. Phan, « Effectiveness of internal versus external seu scrubbing mitigation strategies in a xilinx fpga : Design, test, and analysis », *Nuclear Science, IEEE Transactions on*, vol. 55, no. 4, p. 2259–2266, 2008. (Cité dans la page 69.)

[113] B. Bridgford, C. Carmichael et C. W. Tseng, « Single-event upset mitigation selection guide », *Xilinx Application Note, XAPP987 (v1. 0)*, 2008. (Cité dans la page 69.)

[114] C. Carmichael et C. Tseng, « Correcting single-event upsets in virtex-4 fpga configuration memory », *Xilinx Application Note (XAPP197)*, 2009. (Cité dans la page 70.)

[115] A. Tiwari et K. Tomko, « Enhanced reliability of finite-state machines in fpga through efficient fault detection and correction », *Reliability, IEEE Transactions on*, vol. 54, no. 3, p. 459–467, 2005. (Cité dans la page 70.)

[116] J. Von Neumann, « Probabilistic logics and the synthesis of reliable organisms from unreliable components », *Automata studies*, vol. 34, p. 43–98, 1956. (Cité dans la page 70.)

[117] R. Oliveira, A. Jagirdar et T. Chakraborty, « A tmr scheme for seu mitigation in scan flip-flops », *in Quality Electronic Design, 2007. ISQED'07. 8th International Symposium on*, p. 905–910, IEEE, 2007.

[118] C. Stroud, « Reliability of majority voting based vlsi fault-tolerant circuits », *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 2, no. 4, p. 516–521, 1994.

[119] M. Radu, D. Pitica et C. Posteuca, « Reliability and failure analysis of voting circuits in hardware redundant design », *in Electronic Materials and Packaging, 2000.(EMAP 2000). International Symposium on*, p. 421–423, IEEE, 2000. (Cité dans la page 70.)

[120] C. Bolchini, A. Miele et M. Santambrogio, « Tmr and partial dynamic reconfiguration to mitigate seu faults in fpgas », *in Defect and Fault-Tolerance in VLSI Systems, 2007. DFT'07. 22nd IEEE International Symposium on*, p. 87–95, IEEE, 2007. (Cité dans la page 71.)

[121] S. Trimberger, « Quintuple modular redundancy for high reliability circuits implemented in programmable logic devices », nov. 2 2004. US Patent 6,812,731. (Cité dans la page 71.)

[122] Z. Huang et H. Liang, « A new radiation hardened by design latch for ultra-deep-sub-micron technologies », *in On-Line Testing Symposium, 2008. IOLTS'08. 14th IEEE International*, p. 175–176, IEEE, 2008. (Cité dans la page 71.)

[123] M. Fazeli, A. Patooghy, S. Miremadi et A. Ejlali, « Feedback redundancy : a power efficient seu-tolerant latch design for deep sub-micron technologies », *in Dependable Systems and Networks, 2007. DSN'07. 37th Annual IEEE/IFIP International Conference on*, p. 276–285, IEEE, 2007. (Cité dans la page 71.)

[124] L. Wang, S. Yue et Y. Zhao, « Low-overhead seu-tolerant latches », *in Microwave and Millimeter Wave Technology, 2007. ICMMT'07. International Conference on*, p. 1–4, IEEE, 2007. (Cité dans la page 71.)

[125] M. Nicolaidis, R. Perez et D. Alexandrescu, « Low-cost highly-robust hardened cells using blocking feedback transistors », *in VLSI Test Symposium, 2008. VTS 2008. 26th IEEE*, p. 371–376, IEEE, 2008. (Cité dans la page 71.)

[126] O. Amusan, A. Steinberg, A. Witulski, B. Bhuva, J. Black, M. Baze et L. Massengill, « Single event upsets in a 130 nm hardened latch design due to charge sharing », *in Reliability physics symposium, 2007. proceedings. 45th annual. ieee international*, p. 306–311, IEEE, 2007. (Cité dans la page 71.)

[127] T. CALIN, M. NICOLAIDIS et R. VELAZCO, « Upset hardened memory design for submicron cmos technology », *Nuclear Science, IEEE Transactions on*, vol. 43, no. 6, p. 2874–2878, 1996. (Cité dans la page 71.)

[128] Y. SHIYANOVSKII, F. WOLFF et C. PAPACHRISTOU, « Sram cell design protected from seu upsets », *in On-Line Testing Symposium, 2008. IOLTS'08. 14th IEEE International*, p. 169–170, IEEE, 2008. (Cité dans la page 71.)

[129] R. VELAZCO, D. BESSOT, S. DUZELLIER, R. ECOFFET et R. KOGA, « Two cmos memory cells suitable for the design of seu-tolerant vlsi circuits », *Nuclear Science, IEEE Transactions on*, vol. 41, no. 6, p. 2229–2234, 1994. (Cité dans la page 71.)

[130] H. ZARANDI, S. MIREMADI, C. ARGYRIDES et D. PRADHAN, « Fast seu detection and correction in lut configuration bits of sram-based fpgas », *in Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, p. 1–6, IEEE, 2007. (Cité dans la page 71.)

[131] A. DUTTA et N. TOUBA, « Multiple bit upset tolerant memory using a selective cycle avoidance based sec-ded-daec code », *in VLSI Test Symposium, 2007. 25th IEEE*, p. 349–354, IEEE, 2007. (Cité dans la page 71.)

[132] A. DUTTA et N. TOUBA, « Reliable network-on-chip using a low cost unequal error protection code », *in Defect and Fault-Tolerance in VLSI Systems, 2007. DFT'07. 22nd IEEE International Symposium on*, p. 3–11, IEEE, 2007. (Cité dans la page 71.)

[133] J. SINGH, J. MATHEW, M. HOSSEINABADY et D. PRADHAN, « Single event upset detection and correction », *in Information Technology,(ICIT 2007). 10th International Conference on*, p. 13–18, IEEE, 2007. (Cité dans la page 71.)

[134] R. LE, « Soft error mitigation using prioritized essential bits », *Xilinx XAPP538 (v1. 0)*, 2012. (Cité dans la page 72.)

[135] K. CHAPMAN, « Virtex-5 seu critical bit information extending the capability of the virtex-5 seu controller », *Xilinx Documentation SEU lounge, http ://www. xilinx. com*, 2010. (Cité dans les pages 72 et 73.)

[136] U. LEGAT, A. BIASIZZO et F. NOVAK, « Seu recovery mechanism for sram-based fpgas », *IEEE Transactions on Nuclear Science*, vol. 59, no. 5, p. 2562–2571, 2012.

[137] A. Van de GOOR et I. TLILI, « A systematic method for modifying march tests for bit-oriented memories into tests for word-oriented memories », *Computers, IEEE Transactions on*, vol. 52, no. 10, p. 1320–1331, 2003. (Cité dans les pages 80, 81, 86, 87, 88, 90, 91, 93, 95, 96, 98, 103, et 104.)

[138] S. HAMDIOUI et A. Van de GOOR, « Efficient tests for realistic faults in dual-port srams », *IEEE Transactions on Computers*, p. 460–473, 2002. (Cité dans les pages 81, 83, et 84.)

[139] S. HAMDIOUI, Z. AL-ARS, A. VAN DE GOOR et M. RODGERS, « Linked faults in random access memories : concept, fault models, test algorithms, and industrial results », *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 23, no. 5, p. 737–757, 2004. (Cité dans la page 85.)

[140] Z. AL-ARS et S. HAMDIOUI, « Fault diagnosis using test primitives in random access memories », *in 2009 Asian Test Symposium*, p. 403–408, IEEE, 2009. (Cité dans les pages 81 et 88.)

[141] D. GAITONDE et D. WALKER, « Circuit-level modeling of spot defects », *in Defect and Fault Tolerance on VLSI Systems, 1991. Proceedings., 1991 International Workshop on*, p. 63–66, IEEE, 1991. (Cité dans la page 81.)

[142] J. Knaizuk Jr et C. Hartmann, « An optimal algorithm for testing stuck-at faults in random access memories », *Computers, IEEE Transactions on*, vol. 100, no. 11, p. 1141–1144, 1977. (Cité dans la page 84.)

[143] *Virtex-5 FPGA Configuration User Guide*, vol. UG191, August 14. Xilinx, 2009. (Cité dans les pages 86, 101, et 102.)

[144] R. Dekker, F. Beenker et L. Thijssen, « A realistic fault model and test algorithms for static random access memories », *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 9, no. 6, p. 567–572, 1990. (Cité dans les pages 86, 90, et 103.)

[145] M. Marinescu, « Simple and efficient algorithms for functional ram testing », *in Proc. Int. Test Conf*, p. 236–239, 1982. (Cité dans la page 89.)

[146] A. Van de Goor et I. Tlili, « March tests for word-oriented memories », *in Design, Automation and Test in Europe, 1998., Proceedings*, p. 501–508, IEEE, 1998. (Cité dans les pages 91 et 93.)

[147] M. Nicolaidis, V. Castro Alves et H. Bederr, « Testing complex couplings in multiport memories », *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 3, no. 1, p. 59–71, 1995. (Cité dans les pages 96 et 98.)

[148] *Embedded Processor Block in Virtes-5 FPGAs*, vol. UG200, February 24,. Xilinx, 2010. (Cité dans la page 101.)

[149] R. Treuer et V. Agarwal, « Fault location algorithms for repairable embedded rams », *in Test Conference, 1993. Proceedings., International*, p. 825–834, IEEE, 1993. (Cité dans la page 103.)

[150] F. Lahrach, A. Abdaoui, A. Doumar et E. Chatelet, « A novel sram-based fpga architecture for defect and fault tolerance of configurable logic blocks », *in Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2010 IEEE 13th International Symposium on*, p. 305–308, IEEE, 2010. (Cité dans la page 107.)

[151] F. Lahrach, A. Doumar, E. Châtelet et A. Abdaoui, « Master-slave tmr inspired technique for fault tolerance of sram-based fpga », *in VLSI (ISVLSI), 2010 IEEE Computer Society Annual Symposium on*, p. 58–62, IEEE, 2010.

[152] F. Lahrach, A. Doumar et E. Chatelet, « Fault tolerance of sram-based fpga via configuration frames », *in Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2011 IEEE 14th International Symposium on*, p. 139–142, IEEE, 2011.

[153] F. Lahrach, A. Doumar et E. Chatelet, « Fault tolerance of multiple logic faults in sram-based fpga systems », *in Digital System Design (DSD), 2011 14th Euromicro Conference on*, p. 231–238, IEEE, 2011. (Cité dans la page 107.)

[154] G. Sébastien, *Modélisation et contrôle formel de la reconfiguration application aux systèmes embarqués dynamiquement reconfigurables*. Thèse de doctorat, Universite de Bretagne Sud, 2012. (Cité dans la page 113.)

[155] R. David, *Architecture reconfigurable dynamiquement pour applications mobiles*. Thèse de doctorat, Rennes 1, 2003. (Cité dans les pages 113 et 115.)

[156] M. Herbordt, T. VanCourt, Y. Gu, B. Sukhwani, A. Conti et D. DiSabello, « Achieving high performance with fpga-based computing », *Computer*, vol. 40, no. 3, p. 50–57, 2007. (Cité dans la page 115.)

[157] E. Stott, P. Sedcole et P. Y. Cheung, « Fault tolerant methods for reliability in fpgas », *in Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*, p. 415–420, IEEE, 2008. (Cité dans les pages 115 et 116.)

[158] E. Stott, J. Wong, P. Sedcole et P. Cheung, « Degradation in fpgas : measurement and modelling », *in Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays*, p. 229–238, ACM, 2010. (Cité dans la page 115.)

[159] N. Goel et K. Paul, « Hardware controlled and software independent fault tolerant fpga architecture », *in Advanced Computing and Communications, 2007. ADCOM 2007. International Conference on*, p. 497–502, IEEE, 2007. (Cité dans la page 115.)

[160] J. Emmert, C. Stroud, B. Skaggs et M. Abramovici, « Dynamic fault tolerance in fpgas via partial reconfiguration », *in Field-Programmable Custom Computing Machines, 2000 IEEE Symposium on*, p. 165–174, IEEE, 2000. (Cité dans les pages 115 et 117.)

[161] A. Jacobs, A. George et G. Cieslewski, « Reconfigurable fault tolerance : A framework for environmentally adaptive fault mitigation in space », *in Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, p. 199–204, IEEE, 2009. (Cité dans les pages 116 et 117.)

[162] B. Dave et N. Jha, « Cofta : Hardware-software co-synthesis of heterogeneous distributed embedded systems for low overhead fault tolerance », *Computers, IEEE Transactions on*, vol. 48, no. 4, p. 417–441, 1999. (Cité dans la page 116.)

[163] Y. Xie et W. Wolf, « Asicosyn : Co-synthesis of conditional task graphs with custom asics », *in ASIC, 2001. Proceedings. 4th International Conference on*, p. 130–135, IEEE, 2001. (Cité dans la page 116.)

[164] B. Pratt, M. Caffrey, P. Graham, K. Morgan et M. Wirthlin, « Improving fpga design robustness with partial tmr », *in Reliability Physics Symposium Proceedings, 2006. 44th Annual., IEEE International*, p. 226–232, IEEE, 2006. (Cité dans la page 117.)

[165] W. Huang, S. Mitra et E. McCluskey, « Fast run-time fault location in dependable fpga-based applications », *in Defect and Fault Tolerance in VLSI Systems, 2001. Proceedings. 2001 IEEE International Symposium on*, p. 206–214, IEEE, 2001. (Cité dans la page 117.)

[166] A. El-Attar et G. Fahmy, « A study of fault coverage of standard and windowed watchdog timers », *in Signal Processing and Communications, 2007. ICSPC 2007. IEEE International Conference on*, p. 325–328, IEEE, 2007. (Cité dans la page 117.)

Publications

Ce mémoire reprend des concepts, des idées et des supports utilisés précédemment dans les publications suivantes :

Conférences internationales (avec actes et comité de lectures) :

1. **Farid Lahrach**, Abderrahim Doumar, and Eric Châtelet "*Fault Tolerance of Multiple Logic Faults in SRAM-based FPGA Systèmes*". in Euromicro Conference on Digital System Design (DSD), 2011 IEEE 14th International Symposium on. IEEE, Aout 2011, pp. 231-238, Oulu, Finland.
2. **Farid Lahrach**, Abderrahim Doumar, and Eric Châtelet," Fault Tolerance of SRAM-based FPGA via Configuration Frames " in Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2011 IEEE 14th International Symposium on. IEEE, Avril 2011, pp. 139-142, Cottbus, Allemagne.
3. **Farid Lahrach**, Abderrahim Doumar, Eric Châtelet, and Abderrazek Abdaoui, " Master-Slave TMR Inspired Technique for Fault Tolerance of SRAM-Based FPGA " in Proceedings of the 2010 IEEE Annual Symposium on VLSI. IEEE Computer Society, Juillet 2010, pp. 58-62, Lixouri, Grèce.
4. **Farid Lahrach**, Abderazek Abdaoui, Abderrahim Doumar, and Eric Châtelet," A Novel SRAM-based FPGA architecture for defect and fault tolerance of configurable logic blocks " in Design and Diagnostics of Electronic Circuits and Systems (DDECS), Avril 2010 IEEE 13th International Symposium on. IEEE, 2010, pp. 305-308, Vienne, Autriche.

Activités d'enseignements

Université de Technologie de Troyes UTT 2009-2012 :

- **Circuits de l'électronique analogique** (EN01) : 32 heures de TP (48 UTP)
- **Bases de la physique pour l'ingénieur** (PHYS 01) : 36 heures de TP (54 UTP)
- **Mesure physique et instrumentation** (MS11) : 30 heures de TP (45 UTP)
- **Bases de l'informatique** (LO 01) : 26 heures de TP (39 UTP)
- **Circuits de l'électronique analogique** (EN01) : 93 heures de TD (139.5 UTP)
- **Outils mathématiques pour l'ingénieur** (MATH 02) : 47 heures de TD (70.5 UTP)

UTP : Unité de Travail Pédagogique.

École supérieure d'ingénieurs Léonard-de-Vinci ESILV : 2014-2016

- **Circuits de l'électronique analogique** (Électronique II) : 52.5 heures de TD

- **Traitement de signal** (MESIGI2506) : 52.5 heures de TD
- **Eléments de mécatronique** (MESIGI1304) : 66 heures de TD
- **Systèmes électroniques et informatiques** (MESIGI1213) : 48 heures de TD

Groupe Efrei Paris-Sud : 2015-2016

- **Probabilités** : 7.0 heures de CM et TD
- **Langage C** : 64 heures de CM et TP
- **Mathématiques** : 77 heures de CM et TD

Farid LAHRACH

Doctorat : Optimisation et Sûreté des systèmes

Année 2016

Tolérance aux pannes des circuits FPGAs à base de mémoire SRAM

De nos jours, les circuits FPGAs à base de mémoire SRAM sont omniprésents dans les applications électroniques embarquées. Ainsi, ces circuits sont devenus un acteur principal dans l'amélioration du rendement de l'ensemble du spectre des systèmes-sur-puce (SoC). Néanmoins, les pannes se sont accentuées dans ces technologies émergentes, qu'il s'agisse de pannes permanentes provenant d'une forte densité d'intégration, associée à une complexité élevée des procédés de fabrication, ou de pannes transitoires découlant des particules chargées qui heurtent les FPGAs dans leurs environnements d'exploitation. La tolérance aux pannes des circuits FPGAs à base de mémoire SRAM est donc un paramètre essentiel pour assurer la sûreté de fonctionnement des applications implémentées.

Dans le cadre de cette thèse, nous proposons une stratégie de tolérance aux pannes qui s'accommode des contraintes de fiabilité pour un système implémenté dans un FPGA à base de mémoire SRAM. Cette stratégie présente une grande flexibilité et un coût faible comparé à la technique de la redondance modulaire triple (TMR), et permet la gestion en temps d'exécution qui est une caractéristique importante pour les applications critiques.

Dans cette thèse, nous proposons également des tests spécifiques, appelés algorithmes March, qui permettent de détecter les pannes intra-mots dans la mémoire de configuration d'un circuit FPGA- SRAM. Ces tests présentent l'avantage de bénéficier d'une implémentation rapide et d'obtenir un taux de couverture élevé.

Mots clés : réseaux logiques programmables par l'utilisateur - tolérance aux fautes (ingénierie) - redondance (ingénierie) - autotest intégré - fiabilité.

Fault Tolerance of SRAM-based FPGAs Circuits

Nowadays, SRAM-based FPGAs are omnipresent for embedded electronic applications. Consequently, these circuits became the key player of the overall System-On-Chip (SoC) yield enhancement. However, faults are increasingly pronounced in these emergent technologies, from permanent faults arising from circuit processing at nanometer scales to transient soft errors arising from high-energy particle hits. So fault-tolerance of SRAM-based FPGA is an important system metric to ensure the dependability of embedded applications.

The first part of this thesis exposes a comprehensive technique to cope with multiple faults in applications implemented in SRAM-based FPGA without incurring substantial area, power, or performance penalties. This approach has three main benefits compared to redundancy-based fault-tolerance: it's very low overhead, the option for runtime management, and its complete flexibility. Run-time management can be a very valuable feature of a system, particularly for mission-critical applications. This fault-tolerance approach handles runtime problems on-line, minimizing the amount of system downtime and eliminating the need for outside intervention.

The last part of this thesis is oriented toward configuration memory array of SRAM-based FPGA test and diagnostic. New fault models in configuration frames and March algorithms are proposed. These tests have the advantage to benefit from a fast implementation and achieving high fault coverage.

Keywords: field programmable gate arrays - fault tolerance (engineering) - redundancy (engineering) - Built-In Self-test – reliability.

Thèse réalisée en partenariat entre :

Aube Conseil Général

utt université de technologie Troyes

Ecole Doctorale "Sciences et Technologies"