Le test unifié de cartes appliqué à la conception de systèmes fiables

Marcelo Lubaszewski

Marcelo LUBASZEWSKI

pour obtenir le titre de **DOCTEUR**

de l'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE
(arrêté ministériel du 30 mars 1992)

(Spécialité: Microélectronique)

Le Test Unifié de Cartes appliqué à la Conception de Systèmes Fiables

Date de soutenance: 20 Juin 1994

Composition du Jury:

MM.	**COURTOIS**	**Bernard**	**Président**
	BOREL	**Joseph**	
	HUERTAS-DÍAZ	**José Luis**	
	LANDRAULT	**Christian**	**Rapporteur**
	SOMA	**Mani**	**Rapporteur**
	TEIXEIRA	**Paulo**	

Thèse préparée au sein du Laboratoire TIMA / INPG.

A Andréa et Natasha

AVANT-PROPOS

Cette thèse a été réalisée au sein du Laboratoire TIMA/INPG. Les travaux dont elle témoigne ont été partiellement effectués dans le cadre du projet ARCHIMEDES (#7107 ESPRIT III - CCE). J'ai personnellement bénéficié d'une bourse d'études dans le cadre de la coopération scientifique franco-brésilienne CAPES-COFECUB.

Je tiens en outre à remercier :

Mr Bernard COURTOIS, Directeur de Recherche au CNRS et Directeur du Laboratoire TIMA, pour m'avoir acueilli au sein de l'équipe, pour avoir dirigé mes travaux et pour me faire l'honneur de présider le jury de cette thèse.

Mr Christian LANDRAULT, Directeur de Recherche au CNRS, pour avoir accepté d'être rapporteur de ce travail et membre de ce jury, ainsi que pour ces nombreux et précieux conseils.

Mr Mani SOMA, Associate Director of the Electrical Engineering Department of the University of Washington at Seattle, for accepting to be an examiner of this thesis. I hope this french document has not been too tedious for him.

Mr Joseph BOREL, Vice-Président de la Division Recherche et Développement CAD de SGS-Thomson, pour avoir accepté de faire partie du jury de cette thèse.

Sr José Luis HUERTAS DÍAZ, Director del Departamento de Diseño Analógico del Centro Nacional de Microelectrónica en España, por haber aceptado formar parte de este tribunal de tesis.

Sr Paulo TEIXEIRA, Professor do Instituto Superior Técnico de Lisboa e Pesquisador do Instituto de Engenharia de Sistemas e Computadores, por aceitar o convite para participar da banca desta tese.

Mes remerciements les plus chaleureux vont à Bernard, qui m'a fait et m'a donné confiance, et à Meryem, Hakim, Vladimir, Heidi et Momo, pour la lecture et la correction de ma thèse.

Mes travaux ont largement bénéfié de discussions avec les membres de TIMA et les participants du projet ARCHIMEDES.

Enfin, je ne pourrais pas citer tous ceux avec qui j'ai partagé les bons et mauvais moments de mon séjour. Qu'ils trouvent tous, ici, ma gratitude et ma reconnaissance.

AVERTISSEMENT

Il est parfois difficile de trouver des mots français pour traduire les termes anglais utilisés dans le domaine de recherche traité dans cette thèse. Etant donné ce problème, le lecteur trouvera dans cet ouvrage la traduction française de certaines expressions, et les termes anglais d'origine dans d'autres cas. Chaque fois qu'une expression en langue anglaise est utilisée, elle est mise entre guillemets et sa traduction est parfois donnée entre parenthèses.

RÉSUMÉ

Si on veut assurer de façon efficace les tests de conception, de fabrication, de maintenance et le test accompli au cours de l'application pour les systèmes électroniques, on est amené à intégrer le test hors-ligne et le test en-ligne dans des circuits. Ensuite, pour que les systèmes complexes tirent profit des deux types de tests, une telle unification doit être étendue du niveau circuit aux niveaux carte et module.

D'autre part, bien que l'intégration des techniques de test hors-ligne et en-ligne fait qu'il est possible de concevoir des systèmes pour toute application sécuritaire, le matériel ajouté pour assurer une haute sûreté de fonctionnement fait que la fiabilité de ces systèmes est réduite, car la probabilité d'occurrence de fautes augmente.

Confrontée à ces deux aspects antagoniques, cette thèse se fixe l'objectif de trouver un compromis entre la sécurité et la fiabilité de systèmes électroniques complexes. Ainsi, dans un premier temps, on propose une solution aux problèmes de test hors-ligne et de diagnostic qui se posent dans les étapes intermédiaires de l'évolution vers les cartes 100% compatibles avec le standard IEEE 1149.1 pour le test "boundary scan". Une approche pour le BIST ("Built-In Self-Test") des circuits et connexions "boundary scan" illustre ensuite l'étape ultime du test hors-ligne de cartes. Puis, le schéma UBIST ("Unified BIST") - intégrant les techniques BIST et "self-checking" pour le test en-ligne de circuits, est combiné au standard IEEE 1149.1, afin d'obtenir une stratégie de conception en vue du test unifié de connexions et circuits montés sur des cartes et modules. Enfin, on propose un schéma tolérant les fautes et basé sur la duplication de ces modules sécuritaires qui assure la compétitivité du système résultant du point de vue de la fiabilité, tout en gardant sa sûreté inhérente.

Mots clé:

Sûreté de fonctionnement, Fiabilité, Tests en-ligne/hors-ligne unifiés, Cartes et systèmes "self-checking", "Boundary scan", Systèmes "fail-safe" fiables

ABSTRACT

On one hand, if the goal is to ensure that the design validation, the manufacturing and the maintenance testing, along with the concurrent error detection are efficiently performed in electronic systems, one is led to integrate the off-line and the on-line testing into circuits. Then, for complex systems to make profit of these two types of tests, such unification must be extended from the circuit to the board and module levels.

On the other hand, although the unification of off-line and on-line testing techniques makes possible the design of systems suiting any safety application, the hardware added for increasing the application safety also decreases the system reliability, since the probability of occurrence of faults increases.

Faced to these two antagonist aspects, this thesis aims at finding a compromise between the safety and the reliability of complex electronic systems. Thus, firstly we propose a solution to the off-line test and diagnosis problems found in the intermediate steps in the evolution towards boards which are 100% compliant with the IEEE standard 1149.1 for boundary scan testing. An approach for the BIST (Built-In Self-Test) of boundary scan circuits and interconnects then illustrates the ultimate step in the board off-line testing. Next, the UBIST (Unified BIST) scheme - merging BIST and self-checking capabilities for circuit on-line testing, is combined with the IEEE standard 1149.1, in order to obtain a design strategy for unifying the tests of interconnects and circuits populating boards and modules. Finally, we propose a fault-tolerant scheme based on the duplication of these kind of modules which ensures the competitivity of the resulting system in terms of reliability at the same time as preserving the inherent module safety.

Keywords:

Dependability, Reliability, Unified on-line/off-line testing, Self-checking boards and systems, Boundary scan, Reliable fail-safe systems

TABLE DES MATIÈRES

LISTE DE FIGURES

INTRODUCTION

A partir de sa conception, et tout au long de sa vie, un circuit intégré subit un certain nombre de tests pour vérifier son bon fonctionnement : la validation des premiers prototypes, le tri de fin de fabrication, le test de maintenance, etc.

Aujourd'hui, au niveau de la conception de circuits intégrés de grande complexité il devient indispensable de prendre en compte les problèmes de test de maintenance après assemblage sur une carte. La solution consiste à intégrer dans le circuit les mécanismes nécessaires à son propre test. Ces techniques sont connues sous le nom de BIST ("Built-In Self-Test") ou autotest intégré : des générateurs de vecteurs de test et des analyseurs de signatures sont insérés dans le circuit afin de produire des séquences de test et de compacter les réponses du circuit aux stimuli appliqués. Le problème du test de maintenance de circuits sur la carte est ainsi résolu et par voie de conséquence, le test de fin de fabrication en est réduit à sa plus simple expression.

Pourtant, pour que le test de la carte soit complet, la vérification des connexions entre circuits doit être accomplie. La technique "boundary scan" est à présent la plus répandue pour résoudre ce problème. L'accès aux connexions est purement électronique et basé sur le décalage de données de test à travers une chaîne de registres. Ces registres d'entrées et sorties sont intégrés à l'interface de chaque circuit sur la carte. Puisque sur une même carte on retrouve normalement des circuits fabriqués par différentes sociétés, un standard IEEE pour le "boundary scan" existe. Le but du standard IEEE 1149.1 est d'assurer la compatibilité de ces circuits au niveau test de connexions externes.

Durant le fonctionnement des systèmes qui contrôlent des processus sécuritaires, on peut utiliser des logiciels ou mécanismes spécifiques pour assurer la détection des défaillances avant qu'un accident désastreux se produise. Une telle détection en-ligne d'erreurs peut être assurée par le matériel, en utilisant la technique de circuits "self-checking". Cette technique est basée sur le codage des sorties des blocs fonctionnels et sur la vérification de ces sorties par l'intermédiaire de contrôleurs spécifiques.

Si on veut assurer d'une façon efficace tous les types de tests nécessaires pour les circuits intégrés, on est amené à intégrer à la fois le test hors-ligne (techniques BIST) et le test en-ligne (techniques "self-checking"). L'unification de ces deux types de test mène à la technique de BIST unifié - nommée UBIST, qui peut être appliquée indistinctement aux tests de conception, de fabrication, de maintenance et au test accompli au cours de l'application.

Bien que le niveau technologique déjà atteint par la conception de circuits intégrés testables soit très important, depuis le début de cette thèse nous nous sommes fixés comme objectif d'apporter des réponses à des problèmes qui restent ouverts :

• Même si la technique "boundary scan" représente l'une des meilleures solutions pour le test hors-ligne des cartes aujourd'hui, la disponibilité sur le marché de circuits intégrant cette technique n'est pas très élevée. Ainsi une méthode efficace pour le test et le diagnostic de cartes partiellement "boundary scan" s'avère nécessaire. D'autre part, malgré la disponibilité sur le marché des processeurs de test "boundary scan", l'autotest d'une carte où tout circuit intègre le standard IEEE 1149.1 et du BIST ne sera effectif que si le partage des fonctions de test parmi les circuits fonctionnels et le processeur est assuré. La décentralisation du test de connexions et de l'analyse des signatures des circuits peut faire que l'architecture du processeur de test est grandement simplifiée.

• Pour que la conception de systèmes complexes à haute sûreté de fonctionnement soit faisable, les propriétés "self-checking" de leurs unités doivent être étendues du niveau circuit au niveau carte et module électronique. Etant donné que des techniques pour le test hors-ligne et en-ligne de circuits et pour le test hors-ligne de cartes existent, la propagation et la compression en-ligne des résultats de test et l'intégration hiérarchisée de toutes ces techniques sont à assurer dans de tels systèmes.

• L'intégration des techniques de test hors-ligne/en-ligne fait qu'il est possible maintenant de concevoir des systèmes pour toute application demandant une haute sûreté de fonctionnement. Pourtant, le matériel ajouté pour assurer une haute sûreté de la fonction réalisée fait que la fiabilité de ces systèmes est réduite, car la probabilité d'occurrence de fautes augmente. La tolérance aux fautes peut faire qu'on obtient le compromis souhaité entre la sécurité et la fiabilité.

Les recherches menées visant à la solution des problèmes mentionnés ci-dessus ont été surtout motivées par l'objectif de développer, à long terme, une vision unifiée des différents types de test nécessaires aux systèmes électroniques. La figure i.1 montre le parcours de nos activités de recherche vers ce but et indique précisément où se placent les contributions de cette thèse dans ce cadre.

Ceci étant notre contexte de travail, l'organisation de cette thèse se présente comme suit (figure i.1) :

Le chapitre I est consacré à l'introduction des principes et des aspects théoriques de trois techniques pour la conception de circuits intégrés en vue du test en-ligne : les circuits "self-checking", les circuits UBIST et les interfaces "fail-safe".

Le chapitre II présente l'étude du test hors-ligne de cartes basées sur le standard IEEE 1149.1.

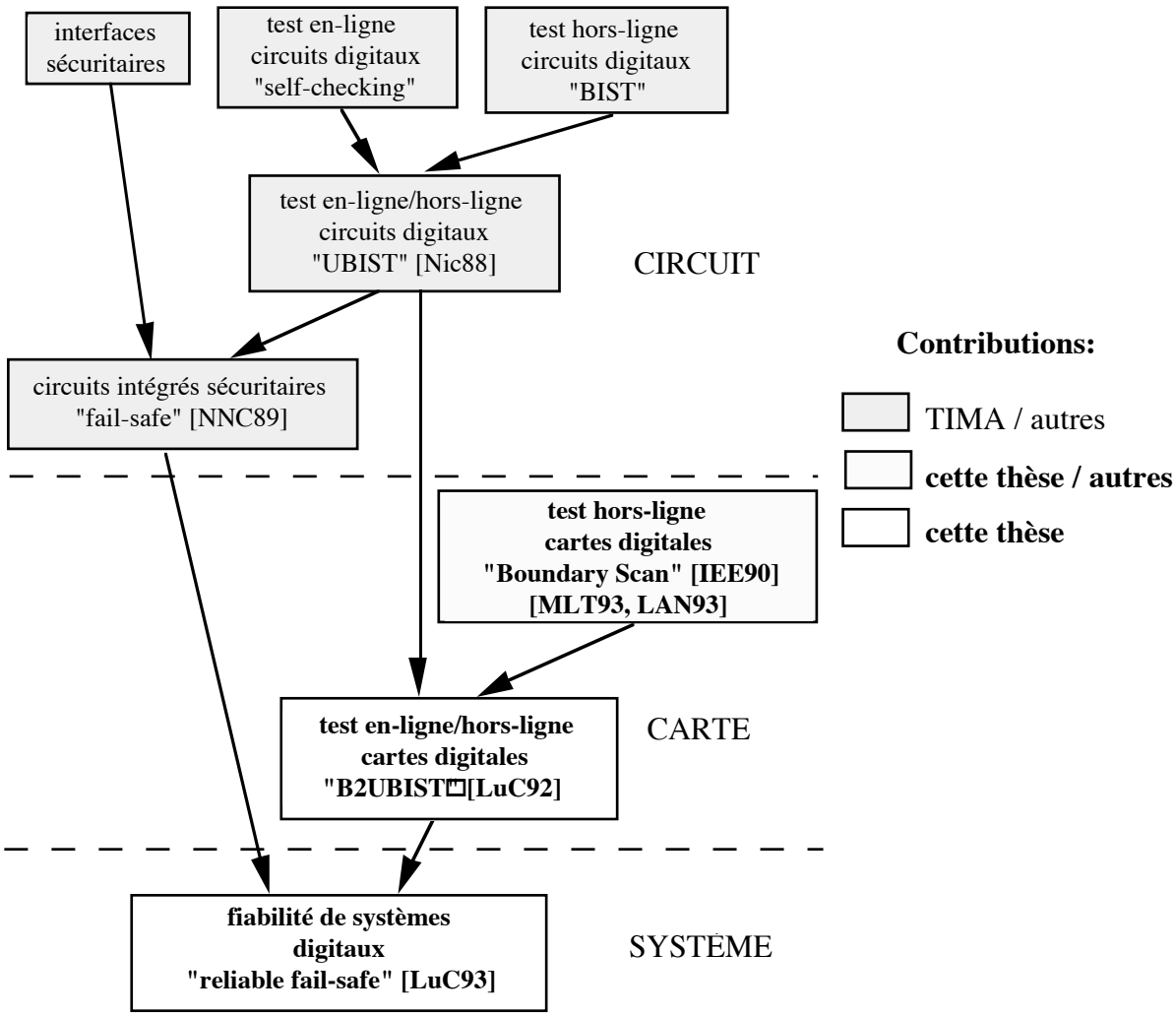


Figure i.1. Le test unifié appliqué à la conception de systèmes digitaux fiables.

Dans le chapitre III, il est tenu compte de l'évolution de la disponibilité du test "boundary scan", en proposant dans un premier temps une méthode pour le test et le diagnostic de cartes partiellement "boundary scan", et dans un deuxième temps une approche pour la vérification de signatures des circuits et pour le BIST des connexions "boundary scan".

Dans le chapitre IV, le schéma UBIST est combiné au standard IEEE 1149.1 afin d'obtenir une stratégie de conception en vue de l'unification des tests hors-ligne et en-ligne de connexions et circuits montés sur des cartes et modules. Ceci résulte dans l'approche nommée B^2UBIST ("Boundary scan Board Unified BIST").

Puis, dans le chapitre V on propose un schéma tolérant les fautes basé sur la duplication des modules sécuritaires étudiés au chapitre IV. Cette approche assure la compétitivité du système résultant du point de vue de la fiabilité, en même temps que la sûreté inhérente à ces modules est préservée.

Finalement, nous concluons sur les perspectives à courte, moyenne et longue échéance pour les activités de recherche démarrées dans cette thèse et pour l'unification des différents aspects du test de systèmes électroniques.

CHAPITRE I

LES CIRCUITS "SELF-CHECKING", LES CIRCUITS UBIST ET LES INTERFACES "FAIL-SAFE"

I.1 Introduction

Durant le fonctionnement d'un système on peut utiliser des logiciels ou des mécanismes spécifiques pour vérifier en permanence la validité des opérations effectuées. Dans les systèmes informatiques utilisés pour contrôler des processus susceptibles de produire des accidents désastreux, par exemple, il est crucial d'avoir la propriété de détection de propres défaillances immédiatement avec l'apparition des premiers résultats erronés. Ce type de test en-ligne, concurrent avec le déroulement d'un programme d'application, peut être toujours assuré par codage logiciel. Pourtant, cette technique amène, d'une part, à une dégradation très importante des performances, et d'autre part, les modèles de défaillances couverts sont à un niveau d'abstraction bien éloigné des défaillances réelles des circuits intégrés [Wad78] [GCV80].

Une solution alternative est de prendre en compte le test en-ligne dès la conception des circuits. Cette solution aurait l'avantage de couvrir des modes de défaillances réels bien connus actuellement. Une telle détection d'erreur peut être assurée par la technique de circuits "self-checking" [CaS68] à travers un apport de matériel. Cette méthode est basée sur des techniques de codage et, pour assurer que la première manifestation d'une erreur soit signalée par le circuit, ses blocs doivent vérifier certaines propriétés qui sont formalisées au niveau mathématique [And71, SmM78, NJC84].

Toutefois, le test en-ligne d'un circuit "self-checking" ne peut être effectif que si certaines conditions très restrictives - concernant, entre autres, le programme d'application, sont satisfaites. Vu qu'en pratique il est souvent très difficile, voire impossible de vérifier ces conditions, une solution consiste à combiner le test hors-ligne intégré et les mécanismes "self-checking" pour le test en-ligne, afin d'obtenir une stratégie pour le test unifié de circuits. L'unification de ces deux types de test mène à la technique de BIST unifié - nommée UBIST [Nic88], qui peut être appliquée indistinctement aux tests de conception, de fabrication, de maintenance et au test accompli en cours de fonctionnement.

Bien que la conception de circuits intégrant des capacités pour le test unifié vise surtout les applications sécuritaires (comme celles de l'industrie nucléaire, ou des transports ferroviaire et aérien), la commande de systèmes électromécaniques utilisés dans la plupart des installations critiques ne peut être assurée que par l'intermédiaire d'interfaces dites "fail-safe". Des études récentes [NNC89] ont abouti à la combinaison des parties "self-checking" pour le calcul et des interfaces "fail-safe" responsable du contrôle d'actionneurs, de manière à obtenir que l'ensemble du système critique soit entièrement "fail-safe".

Dans ce contexte, ce chapitre se consacre à introduire brièvement les trois aspects de la conception en vue du test en-ligne mentionnés ci-dessus: les circuits "self-checking", les

circuits UBIST et les interfaces "fail-safe". L'objectif est de présenter surtout les principes de ces techniques et les aspects théoriques que l'on va appliquer tout au long de cette thèse.

I.2 Les hypothèses de fautes

Etant donné que les mécanismes de défaillances liés à la durée de vie des circuits sont en général très lents, la probabilité d'occurrence simultanée de plusieurs fautes est négligeable. Bien que ce soit réaliste de considérer que pour le test en-ligne les fautes matérielles sont simples, la réussite de la méthode dépend surtout de l'utilisation d'un modèle de faute simple qui représente bien les défaillances réelles.

Pendant longtemps, le modèle pris en compte pour le test des circuits intégrés a été celui du collage logique [Fri71]. Dans ce modèle on représente les circuits par des portes logiques et on considère qu'une entrée ou une sortie d'une porte prend constamment la valeur logique '0' ou '1'. Pourtant ce modèle n'est pas représentatif de toutes les fautes réelles des circuits intégrés, comme discuté en [Wad78]. Par conséquent, de nouveaux modèles sont apparus dans lesquels les circuits sont représentés au niveau électrique, par des réseaux des transistors MOS et on considère comme fautes possibles des collages des lignes à une valeur logique, des collages de MOS passant ou ouvert ("stuck-on", "stuck-open") et des courts-circuits [GCV80].

En ce qui concerne le niveau circuit intégré, on considère dans cette étude l'implémentation réelle telle que décrite par le dessin des masques. C'est-à-dire qu'on tient compte des lignes de diffusion, de polysilicium et d'aluminium, des croisements entre une ligne de polysilicium et une ligne de diffusion pour former un transistor MOS, des contacts ou précontacts entre deux lignes de niveaux différents, etc… Ce sont donc les défaillances de ces éléments qui nous intéressent□une coupure d'une ligne, un court-circuit entre deux lignes, un contact ou un précontact défaillant, un transistor MOS défaillant, etc… A ce niveau, on dispose de la classification de mécanismes de fautes proposée dans [Cou81], où les fautes étudiées sont celles dûes à des phénomènes physiques de la matière (transport de particules, claquage de diélectrique,…). Cette étude a abouti à regrouper les fautes possibles en trois classes données figure I.1, seule la classe I étant à considérer ici dû à l'hypothèse de faute simple mentionnée ci-dessus. Puisque les mécanismes de défaillances étudiés sont directement liés à la durée de vie des circuits, une telle classification correspond tout-à-fait aux besoins du test en-ligne intégré. En plus, ces hypothèses sont aussi bien valables pour les technologies NMOS et CMOS, étant donné que les mécanismes de défaillances sont les mêmes dans les deux cas.

classe 0	**classe I**	**classe II**
un défaut simple	classe 0 +	classe I +
un contact coupé		
un MOS "stuck-on"	courts-circuits entre 2	courts-circuits entre 2
un MOS "stuck-open"	alu les plus proches	alu quelconques
un alu coupé	géographiquement	
un poly coupé		de même pour la
une diff coupée	de même pour la	diffusion
une grille flottante	diffusion	défauts multiples

Figure I.1. Classes d'hypothèses de fautes.

I.3 Les circuits "self-checking"

La structure générale des circuits "self-checking" est donnée dans la figure I.2. Les sorties des blocs fonctionnels sont codées et des contrôleurs sont utilisés pour les vérifier. Au cas où un bloc fonctionnel génère un mot de sortie en dehors du code, son contrôleur produit une indication d'erreur [CaS68]. Cette indication sera passée au contrôleur global qui signalera à l'extérieur du circuit l'occurrence d'une erreur. Le contrôleur global est donc responsable du compactage des indications d'erreur des différents blocs en une indication d'erreur globale.

Le but recherché par les circuits "self-checking" est appelé "totally self-checking (TSC) goal". Le but du TSC consiste à assurer que la première sortie erronée d'un bloc fonctionnel est signalée par le réseau de contrôleurs [And71].

Pour assurer ce but, les blocs fonctionnels et les contrôleurs doivent vérifier les propriétés qui sont données ci-dessous. Les conventions utilisées pour présenter ces propriétés sont les suivantes:

Soit un circuit,

X est l'ensemble des vecteurs d'entrée;

Y est l'ensemble des vecteurs de sortie;

A X est l'ensemble des vecteurs du code d'entrée;

B Y est l'ensemble des vecteurs du code de sortie;

F est l'ensemble de fautes pris en compte et f une faute telle que f∈F;

G(x,f) est la valeur de sortie d'une entrée x en présence d'une faute f∈F; et,

G(x,∅) est la valeur de sortie d'une entrée x en l'absence de faute.

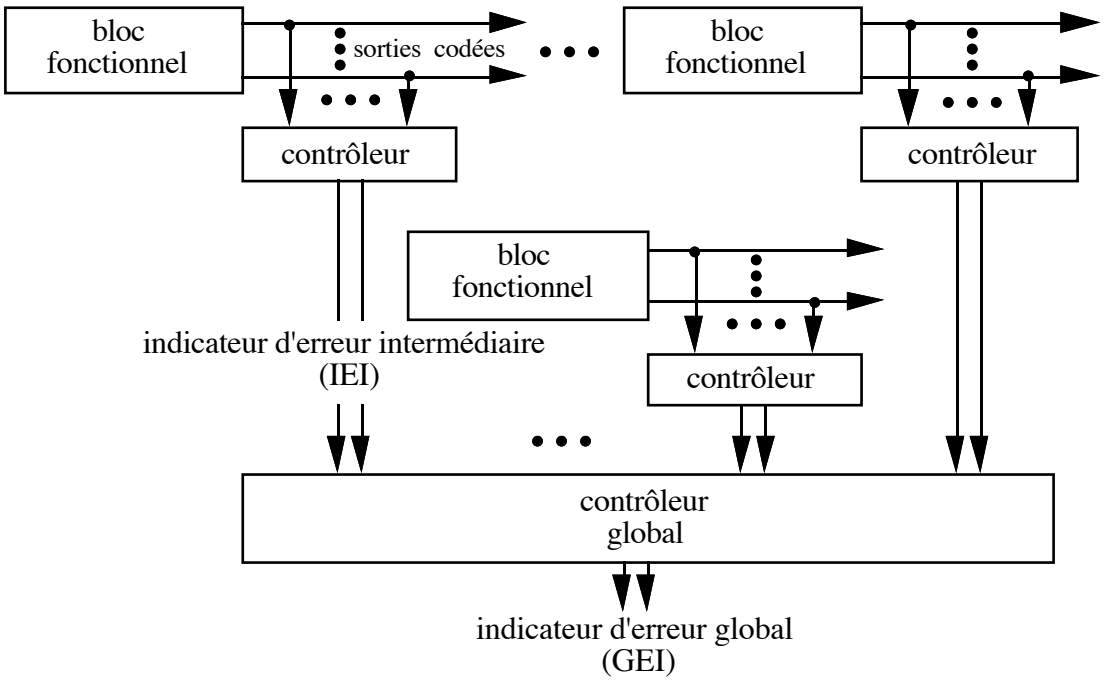


Figure I.2. Structure générale des circuits "self-checking".

Les définitions qui suivent sont dues à [And71] □

Définition I.1 □ Propriété "self-testing" (ST)
G est ST pour F, si $\forall f \in F, \exists a \in A \mid G(a,f) \notin B$.

Définition I.2 □ Propriété "fault secure" (FS)
G est FS pour F, si $\forall f \in F, \exists a \in A \mid G(a,f)=G(a,\varnothing)$ $G(a,f) \notin B$

Définition I.3 □ Propriété "totally self-checking" (TSC)
G est TSC pour F, s'il est ST et FS pour F.

A partir de la définition I.3, on peut déduire qu'un bloc fonctionnel G accomplit le but du "totally self-checking" si la première sortie erronée due aux défauts de l'ensemble F est en dehors du code de sortie. Le but du TSC ne peut cependant être atteint que si l'hypothèse I.1 est respectée□

Hypothèse I.1 □ Entre l'occurrence de deux fautes quelconques de F, il s'écoule un laps de temps suffisant pour que tous les vecteurs du code d'entrée soient appliqués aux entrées du circuit G.

La plus grande classe de circuits fonctionnels capables d'assurer le but du TSC, en respectant l'hypothèse I.1, est la classe des circuits "strongly fault secure" définie en [SmM78]□

Définition I.4 Propriété "strongly fault secure" (SFS)

G est SFS pour F, si

a) soit G est TSC;

b) soit G est FS et si une nouvelle faute survient on retombe sur le cas a) ou b) pour la faute résultante de la combinaison des deux.

En ce qui concerne les contrôleurs, les propriétés suivantes sont à prendre en considération

Définition I.5 Propriété "code disjoint" (CD)

G est CD, si $\forall a \in A$, $G(a,\varnothing) \in B$; $\forall x \in (X-A)$, $G(x,\varnothing) \notin B$.

Définition I.6 Contrôleur TSC

G est un contrôleur TSC, s'il est TSC et CD.

En fait, la propriété "totally self-checking" n'est pas forcément nécessaire pour qu'un contrôleur assure sa mission dans la mesure ou celui-ci reste "code disjoint". A partir de cette observation, le concept de contrôleur "strongly code disjoint" a été proposé [NJC84]

Définition I.7 Propriété "strongly code disjoint" (SCD)

G est SCD pour F, si

- avant l'occurrence d'une faute $f1 \in F$, G est CD; et,
- après l'occurrence d'une faute f1, on a

a) soit G est ST;

b) soit G transpose les vecteurs d'entrée hors-code en vecteurs de sortie hors-code et si une nouvelle faute f2 survient, la combinaison de fautes f1f2 entraîne à nouveau vers le cas a) ou le cas b).

Les contrôleurs "strongly code disjoint" représentent la plus large classe de contrôleurs qui, associés à des blocs fonctionnels "strongly fault secure", peuvent atteindre le but du "totally self-checking".

La figure I.3 montre le fonctionnement d'une cellule de base d'un circuit qui atteint le but du "totally self-checking". Cette cellule est composée d'un bloc fonctionnel SFS et d'un contrôleur SCD. Pour l'ensemble bloc fonctionnel plus contrôleur, l'hypothèse I.2 est adoptée afin de tenir aussi compte du contrôleur.

Hypothèse I.2 : Après l'occurrence d'une faute dans le contrôleur, il s'écoule un laps de temps suffisant pour que tous les vecteurs du code d'entrée A soient appliqués au bloc fonctionnel avant qu'une deuxième faute ne survienne

dans le bloc fonctionnel ou dans le contrôleur.

Après l'occurrence d'une faute dans le bloc fonctionnel, il s'écoule un laps de temps suffisant pour que tous les vecteurs du code d'entrée B soient appliqués au contrôleur avant qu'une deuxième faute ne survienne dans le bloc fonctionnel ou dans le contrôleur.

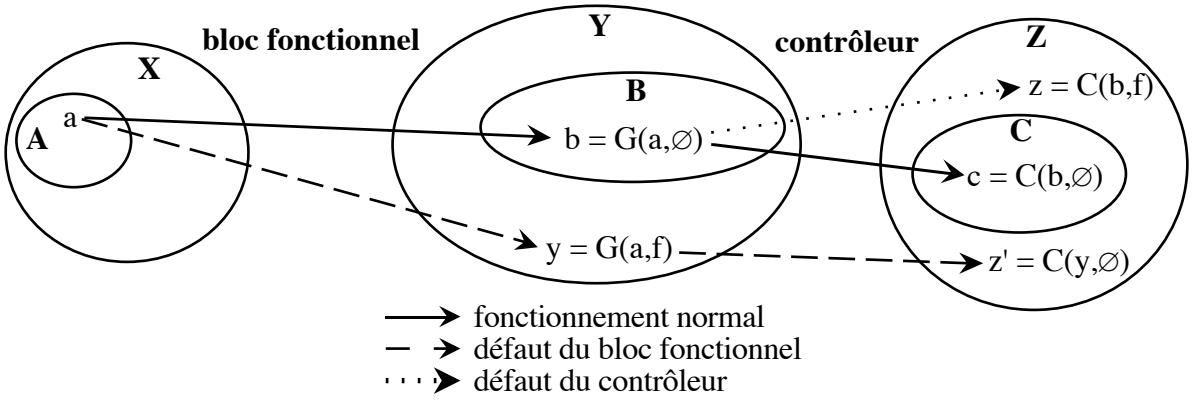


Figure I.3. Fonctionnement théorique d'un circuit "self-checking".

Sous condition de respecter l'hypothèse I.2, le comportement d'une telle cellule est le suivant :

- Si une faute survient dans le bloc fonctionnel,
  soit elle ne modifie pas la fonction du circuit (faute latente),
  soit elle provoque des erreurs qui seront en dehors du code de sortie B.
  Le contrôleur qui est "code disjoint" fournira alors une indication d'erreur avant que la faute suivante se produise.

- Si une faute survient dans le contrôleur,
  soit le contrôleur reste "code disjoint" (faute latente),
  soit il existe au moins une valeur d'entrée du contrôleur qui provoquera un signal d'erreur avant que la faute suivante ne se produise.

I.4 Les erreurs et les codes détecteurs

Dans le cadre des circuits "self-checking", les trois types d'erreurs les plus couramment considérées sont [Cro78] :

Définition I.8 Erreur simple

C'est une erreur qui n'affecte qu'un seul bit d'un mot.

Définition I.9 : Erreur unidirectionnelle

C'est une erreur qui affecte un nombre quelconque de bits d'un mot, mais tous dans le même sens (les changements sont tous d'un seul type 0 -> 1 ou 1 -> 0).

Définition I.10 : Erreur multiple

C'est une erreur qui modifie un nombre quelconque de bits dans les deux sens.

Puisque la capacité "self-checking" passe par l'utilisation d'un code capable de mettre en évidence aux sorties du circuit l'occurrence de tout type d'erreur produite par les fautes du modèle considéré, le choix du code le mieux adapté à chaque cas doit être fait. Aux trois types d'erreurs présentés ci-dessus correspondent différents types de codes, pouvant être séparables ou non, ordonnées ou non [Ber61, PeW72, Cro78].

Définition I.11 : Code séparable

C'est un code où les bits d'information et les bits de codage sont distincts et accolés.

Définition I.12 : Code non-ordonné

C'est un code où il n'existe pas deux vecteurs tels que l'un couvre l'autre. (un vecteur a='$a_1 \ldots a_i \ldots a_n$' couvre un vecteur b='$b_1 \ldots b_i \ldots b_n$' si a_i prend la valeur '1' sur tous les bits où b_i prend la valeur '1')

Définition I.13 : Code de parité (imparité)

C'est un code séparable permettant la détection d'erreurs simples. Il compte le nombre de '1' contenu dans un mot de k bits et affiche le résultat sur un bit supplémentaire. Si le nombre de '1' est pair, le bit de parité est égal à '1' ('0' pour le code d'imparité); et, si le nombre de '1' est impair, le bit de parité est égal à '0' ('1' pour le code d'imparité).

Définition I.14 : Code de Berger [Ber61]

C'est un code séparable non-ordonné permettant la détection d'erreurs unidirectionnelles. Il est obtenu par la concaténation des n bits d'information et de k bits de codage (k= $\lceil \log_2(n+1) \rceil$) représentant le complément du nombre de '1' contenus dans les n bits d'information.

Définition I.15 : Code de duplication

C'est un code capable de détecter des erreurs multiples. Il est obtenu en utilisant soit un bloc fonctionnel identique qui génère la même information

(code dupliqué), soit un bloc fonctionnel dual qui délivre en parallèle le complément de l'information (code "double-rail").

Il y a bien d'autres codes moins utilisés dans les circuits "self-checking" et qui, pour cette raison, ne seront pas présentés ici.

En pratique, pour contrôler le code de parité on calcule le bit de codage en effectuant le OU-exclusif de tous les bits d'information et on le compare avec la parité ajoutée à l'information. Un contrôleur de Berger, par contre, n'est pas aussi simple que cela. Il est composé de deux éléments, un générateur de code normalement conçu à partir d'additionneurs, et un contrôleur "double-rail" qui compare les bits de contrôle de l'information à ceux (complémentés) délivrés par le générateur de code. Dans la figure I.4, on représente la cellule de base d'un contrôleur "double-rail", dont la mise en cascade peut être utile pour obtenir un contrôleur possédant un nombre plus important d'entrées. Le codage des sorties du contrôleur est également de type "double-rail", c'est-à-dire que les sorties $(f0,f1) \in \{(1,0),(0,1)\}$ représentent le bon fonctionnement et que les sorties $(f0,f1) \in \{(0,0),(1,1)\}$ sont données en cas de mauvais fonctionnement.

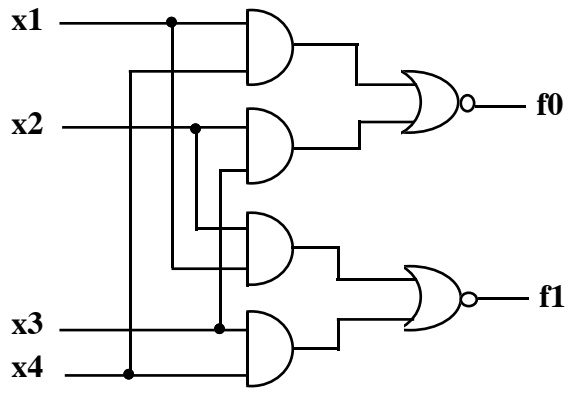


Figure I.4. Cellule de base d'un contrôleur "double-rail".

Toutes ces techniques de codage sont couramment employées pour la conception de circuits "self-checking". Leur inconvénient majeur provient de l'augmentation de surface engendrée et on peut remarquer qu'elle est d'autant plus importante que l'on veut une protection plus élevée.

I.5 La conception de circuits "strongly fault secure"

Dans ce paragraphe, on présente la procédure à suivre pour la conception de circuits possédant les propriétés "self-checking" étudiées dans les paragraphes précédents.

D'abord, pour concevoir un circuit "strongly fault secure", on doit :

- modifier sa fonction afin qu'il génère des sorties appartenant à un code détecteur d'erreurs. Ce code est, par exemple, l'un des codes présentés dans le paragraphe précédent. Le codage des sorties est en général un problème simple de calcul d'une fonction logique.

- s'assurer que toute faute du modèle considéré ne provoquera aux sorties du circuit que des erreurs détectables par le code (propriété "fault secure"). Ce problème est beaucoup plus complexe en raison du grand nombre de fautes possibles dans un circuit intégré et des comportements différents de chaque faute pour des vecteurs d'entrée différents.

- s'assurer que chaque faute est détectable ou dans le cas contraire que, en présence de la faute, le circuit reste "fault secure" (voir la définition I.4 - propriété "strongly fault secure"). Comme le problème précédent, ce problème est d'une complexité importante.

Malgré la difficulté de ces problèmes, la littérature est riche en propositions de circuits "strongly fault secure" ou "totally self-checking" (ces derniers représentant une sous classe des circuits "strongly fault secure"). La plupart de ces propositions sont basées sur le modèle du collage logique mais plus récemment des solutions basées sur des défaillances au niveau électrique ont été développées. On peut en citer : [MAD82] pour la conception des PLAs, [FuA84] pour la conception des ROMs, [HaB84] et [Nic85] pour la conception des unités arithmétiques et logiques, [Nic87] pour la conception des décodeurs, etc...

En ce qui concerne cette thèse, on est intéressé par la conception de circuits "strongly fault secure" basée sur les hypothèses de fautes discutées au paragraphe I.2. Pour ces hypothèses de fautes, des règles générales permettant la conception de circuits "strongly fault secure" ont été déterminées et sont présentées dans [NiC85] et [NiC86]. L'intérêt de ces règles est qu'elles sont valables pour n'importe quel type de circuit et qu'elles sont basées sur un modèle de fautes très proche des défaillances réelles. En se basant sur ces règles, on a pu confirmer ou compléter les propositions données pour d'autres hypothèses de fautes. Ces règles ont permis aussi de concevoir des circuits "strongly fault secure" basés sur des structures non régulières telles que la logique anarchique. Pour conclure, tous les éléments de base existent pour concevoir des circuits "self-checking" complexes tels que les microprocesseurs, par exemple.

Evidemment dans un circuit "self-checking" on doit utiliser des circuits fonctionnels "strongly fault secure" et des contrôleurs "strongly code disjoint". La conception de tels contrôleurs présente aussi de grandes difficultés. Malgré tout, on ne discutera pas de ce problème ici, car il sera résolu d'une façon radicale par la méthode proposée dans le paragraphe suivant.

I.6 La méthode UBIST

Si l'on voulait assurer d'une façon efficace tous les types de tests nécessaires aux circuits intégrés, on serait amené à unifier le test hors-ligne (techniques BIST) et le test en-ligne (techniques "self-checking") dans les circuits. Or les tests intégrés en-ligne et hors-ligne ont été étudiés indépendamment l'un par rapport à l'autre. L'intégration de deux types de test nécessite le développement de nouvelles techniques, afin de pouvoir exploiter au maximum les avantages que l'un pourrait apporter à l'autre. Les études présentées dans [Nic88] ont apporté des résultats très prometteurs dans ce domaine en aboutissant à la technique de BIST unifié, nommé UBIST ("Unified Built-In Self-Test").

Entre autres, cette technique permet de résoudre d'une façon efficace certains problèmes du test en-ligne difficiles à résoudre par les circuits "self-checking"□

- comme il est annoncé dans l'hypothèse I.2 le circuit fonctionnel et le contrôleur doivent recevoir de façon régulière l'ensemble des vecteurs du code d'entrée. Cette hypothèse est très difficile à assurer dans les circuits "self-checking" conventionnels;

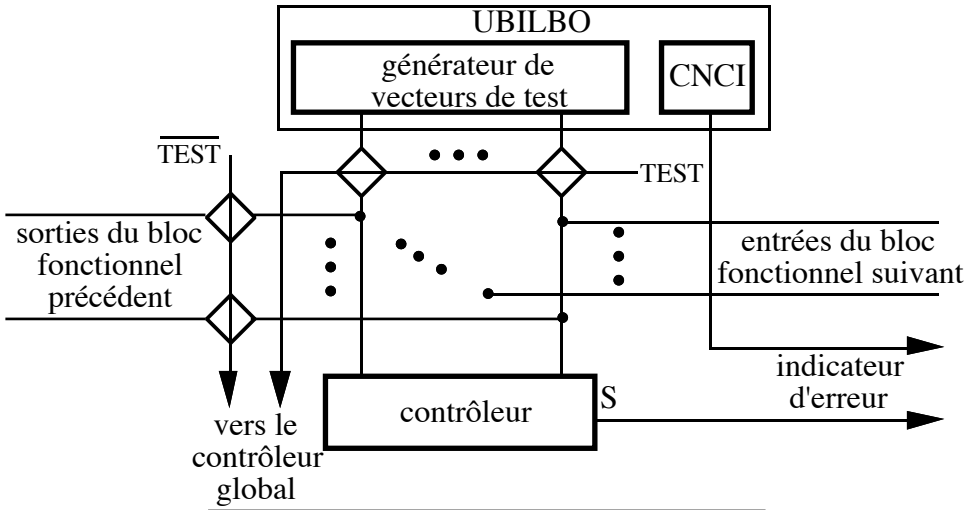
- dans les circuits "self-checking" les contrôleurs et les blocs fonctionnels sont le plus souvent conçus pour couvrir des fautes simples mais, pendant la phase de fabrication, des circuits présentant des fautes multiples peuvent être produits. Il est donc nécessaire d'assurer que toute faute multiple sera détectée avant l'utilisation de ces circuits. Cependant, ceci n'est pas toujours possible dans le cas des circuits "self-checking" conventionnels (par exemple une faute multiple dans un contrôleur peut être détectable uniquement par des vecteurs en dehors du code);

- la conception des contrôleurs "strongly code disjoint" est possible pour des modèles de fautes restreints (e.g. le modèle de collage logique) et sous la condition que le bloc fonctionnel génère un ensemble de vecteurs de sortie prédéterminé. Cette condition n'est pas toujours vérifiée.

Dans la méthode UBIST, le test hors-ligne est alterné avec l'opération normale du circuit de telle sorte qu'une phase de test est activée à des périodes d'une durée inférieure au temps moyen entre fautes pour le système. Les blocs fonctionnels "strongly fault secure" sont testés par l'intermédiaire de générateurs de vecteurs de test et d'analyseurs de signature. La propriété "strongly code disjoint" des contrôleurs est assurée par l'application à leur entrées de vecteurs de test appartenant au code et aussi de vecteurs de test en dehors du code de sortie du bloc fonctionnel. Les structures de test sont partagées entre les tests hors-ligne et en-ligne.

Les mécanismes pour la génération de test et l'analyse de signature sont basés sur des

structures de type UBILBO ("Unified Built-In Logic Block Observer", figure I.5). Ces éléments sont composés d'un registre BILBO [KMZ79] et d'un circuit qui indique la nature du vecteur appliqué (CNCI "Code/NonCode Indicator"). L'utilisation de contrôleurs à deux sorties ne se fait plus nécessaire pour assurer la détection des fautes affectant les sorties, car des mots du code et en dehors du code seront appliqués aux contrôleurs pendant la phase de test hors-ligne produisant ainsi les deux valeurs logiques possibles ('0' et '1') sur la sortie S.



entrée du contrôleur	phase de test S	phase de test CNCI	application S	application CNCI
mot du code	1	0	1	0
mot hors-code	0	1	0	0

Figure I.5. Contrôleur "self-exercising".

Dans le schéma UBIST, la propriété "strongly code disjoint" de la partie contrôle est assurée par l'association d'un UBILBO au contrôleur, l'ensemble étant nommé contrôleur "self-exercising". Cette classe de contrôleurs diffère des contrôleurs classiques par rapport à la structure et aussi à son comportement. Etant donné que les fautes dans les contrôleurs peuvent aussi être détectées pendant une phase de test, leur propriété "self-testing" est redéfinie comme suit :

Définition I.16 : Propriété "self-testing"
Un contrôleur "self-exercising" est "self-testing" pour un ensemble de fautes F, si pour chaque faute de F soit il reçoit pendant l'opération normale une entrée du code qui produit une sortie en dehors du code, soit un mot hors-code est produit aux sorties (S, CNCI) lors de la phase de test.

Il est à noter que dans l'ensemble de fautes F mentionné dans la définition I.16, les fautes dans le contrôleur et aussi celles dans l'UBILBO sont comprises.

Dans ce contexte, pour obtenir les propriétés des contrôleurs "self-exercising" il s'impose que dans les définitions de contrôleurs "totally self-checking" (définition I.6) et "strongly code disjoint" [illegible] abandonne la définition I.1 de "self-testing" en faveur de la nouvelle défi[illegible]I.

Afin d'a[illegible] la propriété "strongly fault secure" pour les blocs fonctionnels par rapport aux fautes dan[illegible] de BIST, les signaux de test qui commandent leur partie opérative (TEST et [illegible]ure I.5) doivent être contrôlés par le contrôleur global [Nic94].

Dans sa globalité, le test hors-ligne de la méthode UBIST est réalisé en trois phases distinctes, TEST1, TEST2 et TEST3 (figure I.6).

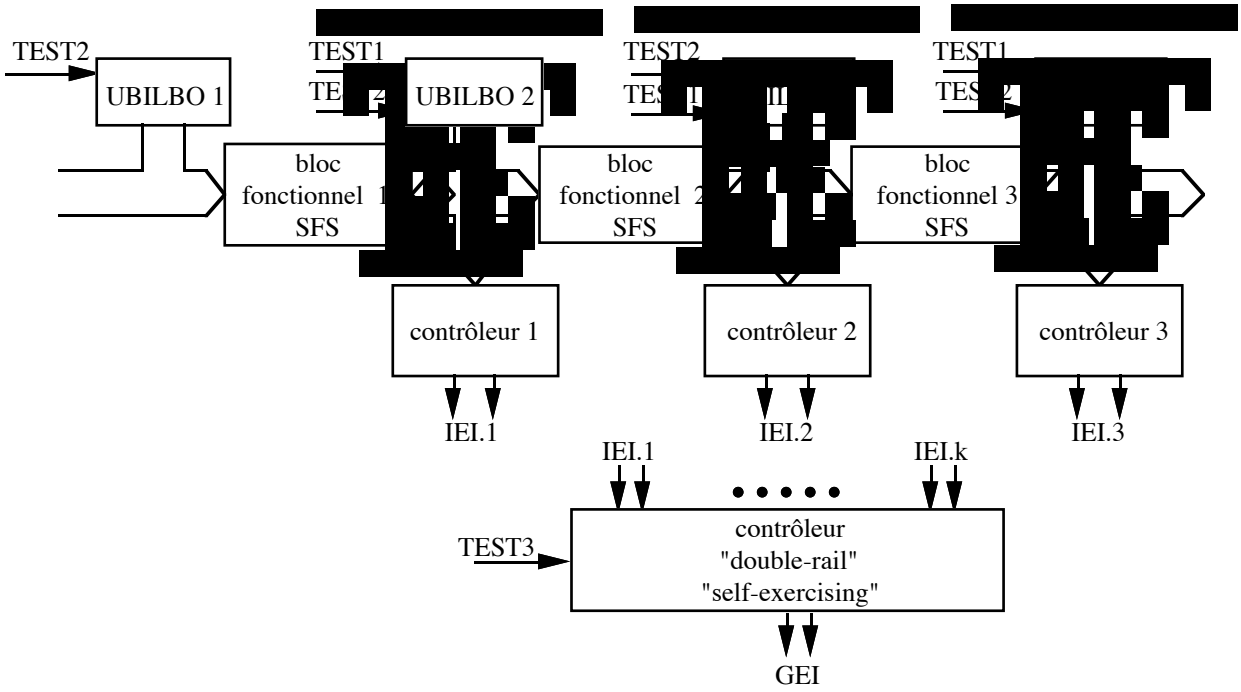


Figure I.6. Le principe de la technique UBIST.

Pendant TEST1 (respectivement TEST2), les UBILBOs pairs (resp. impairs) fonctionnent dans le mode génération de vecteurs de test et les UBILBOs impairs (resp. pairs) fonctionnent dans le mode analyse de signature. Les blocs fonctionnels pairs (resp. impairs) et les contrôleurs impairs (resp. pairs) sont testés pendant cette phase. Les sorties des blocs fonctionnels sont vérifiées par les contrôleurs pairs (resp. impairs) et compactées par les analyseurs de signature. Pour ces deux phases, des réponses de test partielles sont livrées au

contrôleur global du circuit à partir des sorties des contrôleurs intermédiaires (IEI.i). L'analyse de l'ensemble des indicateurs d'erreur intermédiaires résulte dans une indication d'erreur compacte produite à la sortie du contrôleur global (GEI).

La génération de signatures peut être conçue de sorte que le contrôleur global "double-rail" signale lui aussi toute erreur résultante de l'analyse de signature. Dans le chapitre IV, on proposera une approche qui atteint cet objectif.

Pendant la phase TEST3, la fonctionnalité du contrôleur global est vérifiée par l'intermédiaire de son propre générateur de vecteurs de test. Ce générateur joue le rôle d'un fournisseur d'indications d'erreur intermédiaires. Les réponses de test sont toujours données par l'indicateur d'erreur global.

En ce qui concerne le test en-ligne, les blocs fonctionnels et les contrôleurs sont vérifiés de la même manière que dans les circuits "self-checking" conventionnels.

Le test en-ligne et les phases de test hors-ligne TEST1 et TEST2 mettent le signal CNCI du contrôleur global à '0', tandis que le signal S (figure I.5) prend soit la valeur '1' pour le contrôleur sans faute, soit la valeur '0' lorsqu'une faute est présente. D'autre part, en supposant que durant la phase TEST3 l'opération du circuit est correcte, S est mis soit à '1', si un vecteur appartenant au code se présente aux entrées du contrôleur global, soit à '0', pour un vecteur en dehors du code "double-rail", tandis que CNCI oscille de '0' à '1' respectivement.

L'unification des tests en-ligne et hors-ligne mise à part, la technique UBIST permet d'amener à 100% la couverture pour le test hors-ligne des fautes simples et à une augmentation importante de la couverture des fautes multiples. D'autre part, on peut prédire que la technique UBIST permet de diminuer la surface et d'augmenter la vitesse des contrôleurs, la raison étant que certaines restrictions sur la conception des contrôleurs sont éliminées. En général, cette diminution de surface des contrôleurs contrebalance la surface nécessaire pour assurer le test hors-ligne et permet l'implémentation de la technique UBIST à peu de frais par rapport à une implémentation "self-checking" conventionnelle. Une première application de cette technique concernant l'implémentation UBIST pour la partie contrôle du microprocesseur MC 68000 confirme ces affirmations [Nic90].

I.7 Les systèmes "fail-safe"

D'une part, les systèmes "self-checking" (avec ou sans BIST) fournissent la possibilité de signaler l'apparition d'une erreur. S'il n'y a pas de signal d'erreur, la valeur est la bonne. D'autre part, il n'y a que les systèmes "fail-safe" qui puissent fournir la possibilité de surveiller les systèmes critiques, en donnant des sorties qui n'engendreront pas des situations

dangereuses, même si elles sont erronées (sorties d'état sûr). Pourtant, contrairement aux circuits "self-checking", il est difficile d'intégrer des circuits "fail-safe".

Des études préalables [NNC89] ont réussi à généraliser la théorie des circuits "fail-safe" et à faire la liaison avec les circuits "self-checking", tout en proposant des implémentations VLSI entièrement "fail-safe" et capables de commander des actionneurs électromécaniques.

Dans ce paragraphe, nous allons rappeler quelques unes des définitions fondamentales données dans la théorie généralisée de systèmes "fail-safe". En particulier, deux définitions de système "fail-safe" nous intéressent.

Définition I.17 Propriété "fail-safe" inconditionnelle

Un système G est "fail-safe" pour un ensemble de fautes F, un ensemble de vecteurs d'entrée X et un ensemble d'états sûrs O_s si :

$\forall x \in X, \forall f \in F : G(x,f)=G(x, \varnothing) \quad G(x,f) \in O_s$

Dès à présent nous pouvons remarquer qu'un circuit "fault secure", conçu de telle sorte que l'occurrence de vecteurs de sortie erronés ne provoque pas de situation dangereuse, peut être considéré comme un circuit "fail-safe".

Une autre définition de système "fail-safe" peut être énoncée pour faciliter la prise en compte de certaines situations dès la conception. Elle sera valable pour les circuits dont les sorties sont partagées en plusieurs groupes $O^1,\ldots,O^n$ où pour chaque groupe de sorties O^i la notion d'état sûr est considérée indépendamment des autres groupes de sorties O^j.

Définition I.18 Propriété "fail-safe" partagée

Un système G aux sorties partagées est "fail-safe" pour un ensemble de fautes F, un ensemble de vecteurs d'entrée X et un ensemble d'états sûrs $O^1_s,\ldots,O^n_s$ si :

$\forall x \in X, \forall f \in F, \forall i \in \{1,2,\ldots,n\} : G^i(x,f)=G^i(x,\varnothing) \quad G^i(x,f) \in O^i_s$,

où :

$G^i(x,\varnothing)$ est le vecteur de sortie obtenu normalement pour le sous-groupe de sorties O^i, et

$G^i(x,f)$ est le vecteur erroné présent en sortie de ce même sous-groupe après l'occurrence d'une faute.

Supposons l'interconnexion des deux sous-systèmes "fail-safe" A et B (ou B1,…,Bk) comme montré dans la figure I.7. Lorsque plusieurs sous-systèmes "fail-safe" sont utilisés pour concevoir un système plus complexe, pour que le système global soit "fail-safe", les ensembles de vecteurs d'entrée et de sortie des différents sous-systèmes doivent satisfaire des

conditions bien définies.

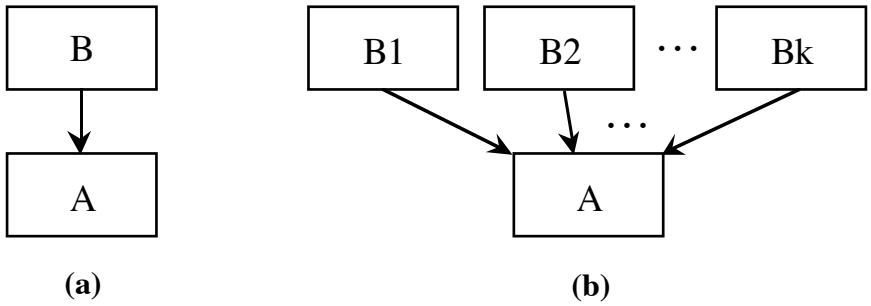


Figure I.7. Interconnexion de sous-systèmes "fail-safe".

Tout d'abord, notons I_s l'ensemble de vecteurs d'entrée d'état sûr, ensemble pour lequel un système "fail-safe" qui fonctionne normalement délivre des vecteurs de sortie qui appartiennent à O_s. En se reportant à la figure I.7(a), supposons que $O^b{}_s$ est l'ensemble des vecteurs de sortie d'état sûr du sous-système B. De même, $I^a{}_s$ est l'ensemble des vecteurs d'entrée d'état sûr du sous-système A. Pour concevoir un système glo[redacted]fe" lorsque deux sous-systèmes A et B vérifient la définition I.17, la co[redacted]dition à respecter est la suivante□

Condition I.1□ Les deux sous-systèmes A et B vérifient la définition i[redacted]le système global est "fail-safe" si la relation $O^b{}_s$ [redacted] $I^a{}_s$ est respectée.

Examinons maintenant le cas du sous-système B de la figure I.7(a) qui vérifie la définition I.18 ou, ce qui est équivalent, le cas des sous-systèmes B1,…,Bk (figure I.7(b)) qui vérifient chacun la définition I.17. Admettons que le sous-système A vérifie la définition I.17 et que le sous-système B aux sorties partagées est connecté à ce dernier à travers plusieurs groupes de sorties $O^{b1},\ldots,O^{bn}$. Pour chaque sous-groupe de lignes O^{bi} les vecteurs de sortie appartiendront à des ensembles de vecteurs d'état sûr ou non-sûr qui sont indépendants de l'état des vecteurs des autres sous-groupes de lignes O^{bj}. Soit $I^{a1},\ldots,I^{an}$ les groupes de lignes d'entrée de A qui correspondent aux différents groupes de sorties de B. Nous définissons $\{I^{a1},\ldots,I^{an}\}$ comme l'ensemble de sous-ensembles des vecteurs d'entrée d'état sûr pour lesquels chaque fois qu'un vecteur erroné e^{ai} du sous-ensemble $I^{ai}{}_s$ est appliqué au groupe de lignes d'entrée I^{ai} et que les autres vecteurs appliqués aux autres groupes des lignes d'entrée I^{aj} sont corrects, alors le sous-système A qui fonctionne normalement délivre soit le vecteur de sortie attendu, soit un vecteur de sortie d'état sûr. La condition à respecter pour que la globalité du système composé de A et B soit "fail-safe" est la suivante□

Condition I.2 Le sous-système B vérifie la définition partagée (définition I.18), le sous-système A vérifie la définition inconditionnelle (définition I.17), le système global réalisé à l'aide des deux sous-systèmes A et B est "fail-safe" s'il existe [illegible] $\{I^{a1}_s,\ldots,I^{an}_s\}$ pour lequel les relations $O^{bi}_s \; I^{ai}_s$, $\forall i\in\{1,\ldots,n\}$, sont satisfaites.

Concevoir un système "fail-safe" complexe n'est pas une tâche facile. Le problème est pourtant simplifié si nous adoptons une solution qui consiste à découper le système global en sous-systèmes. Dans la suite, nous donnons un théorème qui reporte la difficulté au niveau de sous-systèmes, chacun devant être conçu de telle sorte à satisfaire la définition I.17 ou I.18.

Théorème I.1 Soit G un système tel que:

- il est composé de n sous-systèmes S1,…,Sn "fail-safe" conformément aux définitions I.17 et I.18 et pour les ensembles de fautes respectifs F1,…,Fn; et,
- les n sous-systèmes sont connectés les uns aux autres en respectant les conditions I.1 et I.2 suivant aux définitions qu'ils vérifient.

Alors G est "fail-safe" pour l'ensemble de fautes F = F1 x … x Fn.

Dans ce théorème, l'ensemble de fautes possibles Fi comprend aussi la faute fi=∅. Alors, une faute f∈F est composée de n fautes (f1,…,fn)∈F=F1x…xFn, ce qui est pareil à dire que le système est "fail-safe" pour les fautes affectant simultanément n'importe quel ensemble de sous-systèmes de G.

I.8 Le but du "totally fail-safe" et les systèmes "strongly fail-safe"

Si un système G est "fail-safe" pour un ensemble de fautes F, et si l'occurrence d'une première faute f1 appartenant à F ne génère pas de vecteur de sortie erroné non sûr, alors la sécurité est assurée. Cependant, le vecteur sûr ainsi généré peut appartenir à l'ensemble des vecteurs sûrs obtenu en fonctionnement normal. Dans ce cas, la première faute n'est pas signalée, puisque de manière générale, les systèmes "fail-safe" ne possèdent pas nécessairement de moyens de détection de fautes. Si le système fonctionne pendant un temps assez long, une seconde faute f2 appartenant à F peut apparaître et la faute double (combinaison de f1 et f2) créée peut alors ne plus appartenir à l'ensemble F. La propriété "fail-safe" est alors perdue. Il est donc essentiel de détecter une faute dès son apparition et empêcher le système défaillant de fonctionner. La détection d'erreur s'inspire des propriétés qui avaient été établies pour les circuits "self-checking". Si un système ne possède aucun moyen de détection d'erreur, il suffit de lui rajouter un mode de test du type hors-ligne. Pour assurer la généralité des définitions qui suivent, redéfinissons encore une fois la propriété "self-testing".

Définition I.19 Propriété "self-testing"

Un circuit est "self-testing" pour un ensemble de fautes F, si pour chaque faute f appartenant à F il existe au moins un mode durant lequel la faute est détectée.

De manière semblable aux systèmes "self-checking", le but du "totally fail-safe" (TFS) est associé aux systèmes "fail-safe". Ce but consiste à assurer que les sorties erronées du système appartiennent à l'ensemble des vecteurs de sortie d'état sûr. A condition qu'une nouvelle hypothèse I.3 plus générale soit valable, le but du TFS sera atteint par les circuits "totally fail-safe" □

Hypothèse I.3 Les fautes apparaissent une à une et, entre l'occurrence de deux fautes successives, il s'écoule un laps de temps assez long pour que le circuit soit contrôlé avec les moyens mis en oeuvre pendant les différents modes de fonctionnement possibles.

Définition I.20 Propriété "totally fail-safe"

Un circuit est "totally fail-safe" s'il est à la fois "fail-safe" et "self-testing".

Si l'hypothèse I.3 est vérifiée, la plus grande classe de circuits "fail-safe" capables d'atteindre le but du "totally fail-safe" est celle des circuits "strongly fail-safe"□

Définition I.21 Propriété "strongly fail-safe"

Un circuit G est "strongly fail-safe" pour un ensemble de fautes F, si pour toute f appartenant à F□

a) G est "totally fail-safe"; ou

b) G est "fail-safe" et si une nouvelle faute f2 appartenant à F survient, la faute double ainsi créée redonne le cas a) ou le cas b).

Si on ne dispose pas de moyen de détection, la condition nécessaire et suffisante pour que le système soit "strongly fail-safe" est qu'il doive être "fail-safe" pour l'ensemble des fautes multiples. Les circuits "fail-safe" décrits dans la littérature sont généralement conçus en utilisant des composants dupliqués [MiK67] ou des composants dont le temps moyen entre fautes est très élevé pour les fautes considérées comme "dangereuses" (technique d'évitement des fautes, [FSM88]). En fait, la propriété "fail-safe" ne peut pas être assurée pour des fautes multiples lorsque l'on utilise la duplication à cause de l'apparition possible de fautes doubles dans deux composants dupliqués. La technique d'évitement n'est pas non plus envisageable car certaines fautes ont des conséquences contraires. De plus, la probabilité d'occurrence de telles fautes ne peut pas être considérée comme négligeable puisque les fautes peuvent se cumuler durant la vie

du système. Ces techniques ne sont donc pas compatibles avec l'utilisation de systèmes intégrés VLSI "strongly fail-safe".

En conséquence, nous allons nous intéresser à la conception de systèmes "fail-safe" utilisant une technique de codage (circuit "self-checking") et une interface qui sera "fail-safe" grâce à l'intégration de composants "fail-safe" et même "strongly fail-safe" puisque l'on utilise la technique BIST qui assurera la détection de la première faute.

Dans le paragraphe suivant, des aspects pratiques liés à l'application des définitions présentées précédemment à des implémentations "self-checking" seront abordés.

I.9 Les interfaces "fail-safe" et "strongly fail-safe"

De manière générale les processus critiques demandent que chaque signal de sortie d'une fonction soit "fail-safe" individuellement. Pour cela la plupart de ces systèmes génère des sorties codées en fréquence.

Cette technique consiste à faire correspondre à un état non sûr le passage d'une fréquence (état '1') et à considérer tout autre état électrique (état '0') comme état sûr (notamment le blocage du passage de la fréquence). Cette solution a aussi l'avantage de s'adapter à la commande des éléments électromécaniques (actionneurs) qui se trouvent généralement placés en bout de chaîne des fonctions critiques [FSM88].

La figure I.8 présente le principe de réalisation d'une interface "fail-safe" pour un système "self-checking". Dans la figure, les blocs fonctionnels sont "strongly fault secure" et les contrôleurs "strongly code disjoint". Le circuit délivre n sorties (I1,…,In) fonctionnelles codées avec k bits (C1,…,Ck). Les signaux d'indication d'erreur intermédiaires sont interprétés par le contrôleur "double-rail" global pour donner une indication d'erreur globale (f0,f1). Comme les blocs fonctionnels sont SFS et l'ensemble des contrôleurs est SCD, chaque vecteur erroné sera donc un vecteur hors-code. Si le vecteur (I1,…,In,f0,f1) appartient au code, alors l'interface fournit une fréquence F∅ correspondant à l'état non sûr (état '1') aux sorties (O1,…,On) en fonction de l'autorisation de passage délivré par le vecteur (I1,…,In).

La propriété "fail-safe" de l'interface est assurée en dupliquant certains transistors (T1 et T2), mais comme par hypothèse seule l'occurrence d'une faute simple est possible, il n'est pas nécessaire de dupliquer la porte OU-exclusif en sortie du contrôleur "double-rail" global.

La partie du système de la figure I.8 qui génère le vecteur (I1,…,In,f0,f1) est "strongly fault secure" et par conséquent considérée comme "strongly fail-safe". Si nous voulons que le système global soit "strongly fail-safe", l'interface doit aussi avoir cette propriété.

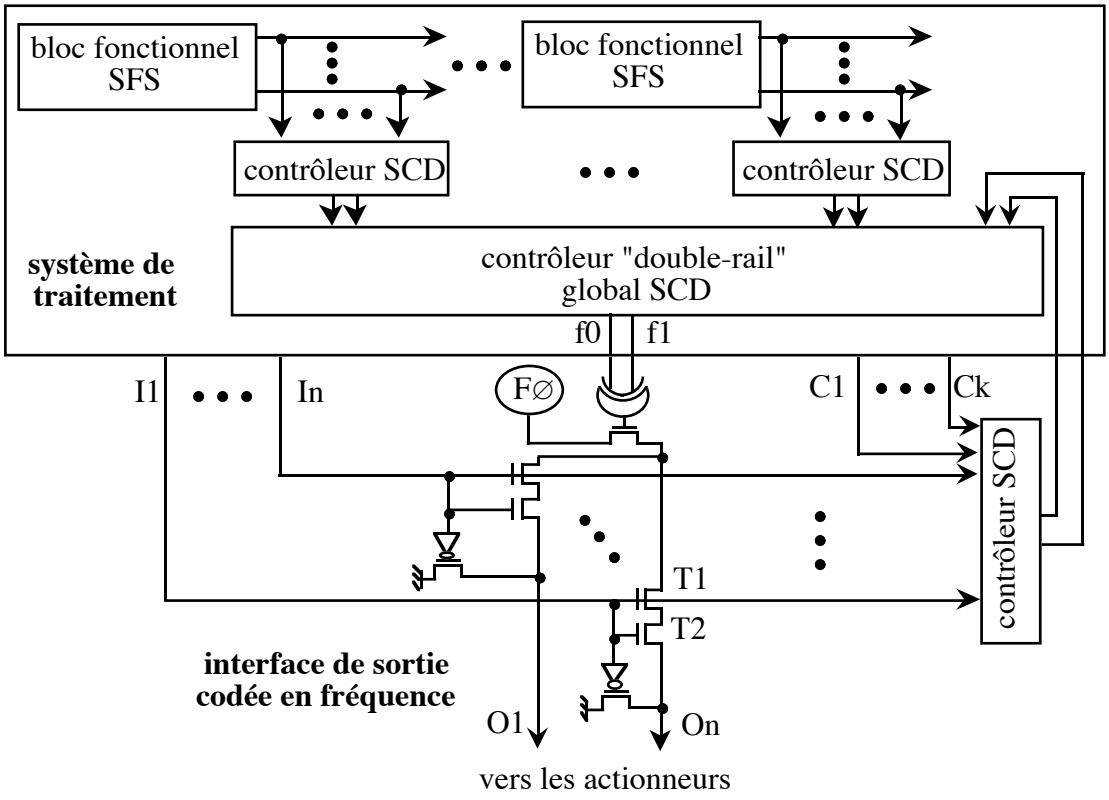


Figure I.8. Principe d'une interface "fail-safe".

Toutefois l'interface de la figure I.8 ne reste pas "fail-safe" en présence de certaines fautes multiples (par exemple, les transistors T1 et T2 "stuck-on", le collage à 1 de la sortie de la porte OU-exclusif combiné à un collage à 1 d'une entrée Ii, etc…). Par conséquent, pour assurer la propriété "strongly fail-safe" de l'interface, il faut prévoir des mécanismes pour détecter une première faute avant l'occurrence d'une deuxième.

Ceci peut être obtenu en utilisant des techniques BIST. Comme montre la figure I.9, où une seule branche de l'interface est représentée, un générateur de vecteurs de test délivre la séquence utile pour détecter les fautes simples dans l'interface. Un analyseur de signature prend en compte l'ensemble de réponses du bloc sous test et la vérification du contenu de cet analyseur sera faite par le contrôleur global à la fin de la phase de test hors-ligne [illegible] dans la technique UBIST).

Deux remarques sont importantes par rapport à l'interface "strongly fail-safe" qu[illegible]s venons de présenter□

1) Comme les systèmes électromécaniques (de type actionneur) sont sensibles [illegible] fréquences de l'ordre de la milliseconde, le mode de test hors-ligne (TEST='1', [illegible])

ayant une durée de que[redacted]des ne perturbe pas le fonctionnement normal de l'interface.

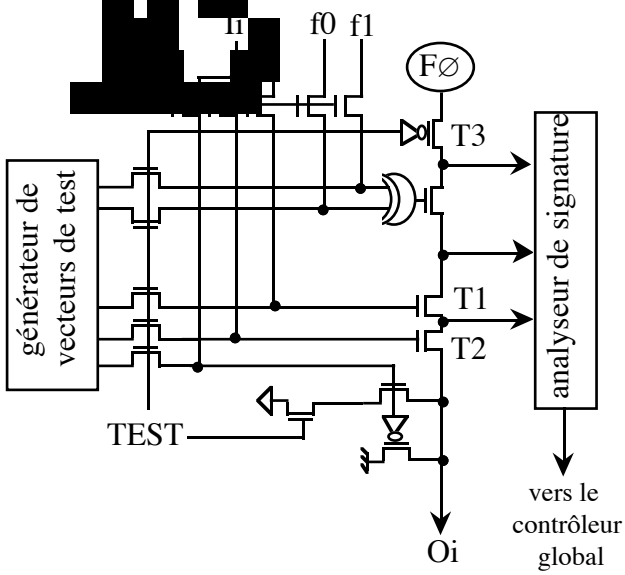


Figure I.9. Interface "strongly fail-safe".

2) Pour que l'interface donnée en figure I.9 soit "strongly fail-safe", la seule contrainte à satisfaire par la génération de test est que toute faute pour laquelle l'interface pourrait perdre la propriété "fail-safe" (par exemple, transistor T3 "stuck-on") soit détectée. Les fautes qui n'altèrent en rien la propriété désirée (par exemple, transistor T3 "stuck-open") ne doivent pas être obligatoirement signalées.

Certes, cette interface ne pourra pas assurer la propriété "strongly fail-safe" dans son intégralité si le système n'est pas arrêté indéfiniment dans un état sûr de fonctionnement. Dans la figure I.10 nous présentons le schéma proposé dans [NNC89] pour la mise du système à un état irréversiblement sûr□la coupure du courant et, par conséquent, l'impossibilité de délivrer la fréquence F∅ aux actionneurs même si une nouvelle faute survient.

Ce schéma utilise l'indicateur asynchrone d'erreur donné dans [Gai85] aux sorties du contrôleur global du système (f0 et f1 dans les figures I.8 et I.9). Ce circuit est capable de mémoriser l'occurrence d'une indication d'erreur à ses entrées et de détecter (mais ne pas mémoriser) à ses sorties ses propres fautes. Afin de lui donner aussi la capacité de mémoriser ses propres fautes, on duplique l'indicateur lui même et on rajoute au circuit deux contrôleurs "double-rail" comme le montre la figure I.10. Ainsi lorsque l'un est défaillant l'autre mémorise les mots en dehors du code produits par le premier. On génère ensuite une sortie supplémentaire pour le circuit, Oe. Les sorties des indicateurs d'erreur sont alors utilisées pour

isoler indéfiniment la sortie Oe de la fréquence Fe lorsqu'une faute est détectée. La sortie Oe commande une circuiterie réalisée au niveau de la carte en utilisant des composants "fail-safe"□ le relais, par exemple, a une très faible probabilité d'avoir une défaillance qui lui mettrait dans l'état passant (ce genre de composant discret est disponible dans le marché). Alors, si la fréquence Fe n'est pas présente sur la sortie Oe pendant quelques millisecondes, le courant du circuit est coupé. Finalement, pour assurer la propriété "strongly fail-safe" pour ce coupe courant on y ajoute du BIST comme l'on a déjà fait pour l'interface de la figure I.9.

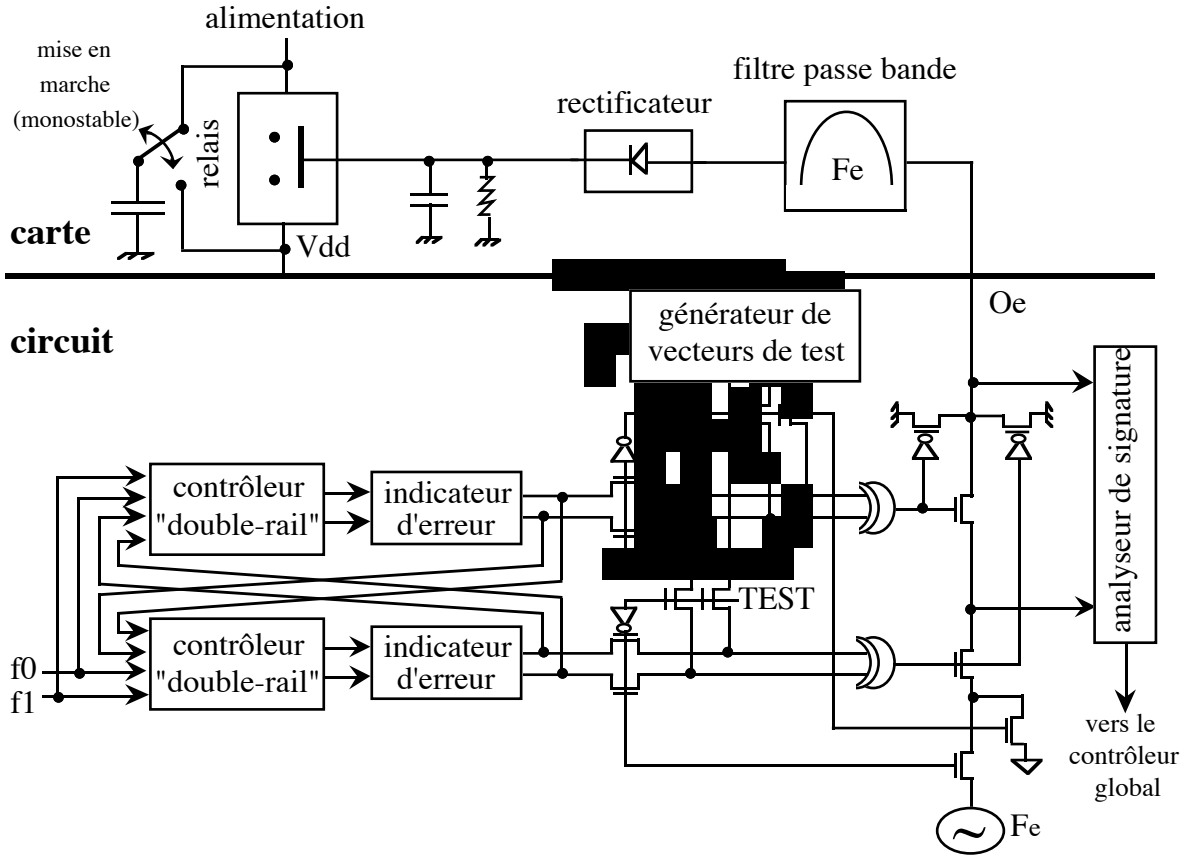


Figure I.10. Coupe courant "strongly fail-safe".

I.10 Conclusion

Dans ce chapitre, nous avons étudié la technique "self-checking" dont le but est de signaler à travers un test en-ligne l'apparition du premier résultat erroné dans un circuit intégré. Cette technique est basée sur le codage de données et sur certaines propriétés à respecter pour chaque bloc composant le circuit.

Vu que cette méthode n'est effective que sous certaines conditions très restrictives, nous avons introduit une technique qui combine harmonieusement la méthode "self-checking" pour

le test en-ligne et la méthode BIST pour le test hors-ligne. L'utilisation de cette technique, nommée UBIST, offre une grande souplesse de conception et une couverture très élevée de fautes de fabrication et d'usage à des coûts raisonnables.

Même si les systèmes "self-checking" ou UBIST fournissent la possibilité de signaler l'apparition d'une erreur, ceci n'est pas suffisant pour assurer une haute sûreté de fonctionnement lorsqu'il s'agit de commander les actionneurs électromécaniques d'un système critique. Pour résoudre ce problème, nous avons présenté une technique qui associe à un sous-système de traitement "self-checking" une interface "fail-safe" donnant des sorties pour le système global qui n'engendreront pas de situations dangereuses (même si ces sorties sont erronées). Les avantages apportés par l'ajout à cette interface d'une phase de test hors-ligne (tout comme dans UBIST) ont été également mis en évidence.

Vu que l'objectif de cette thèse est d'étendre l'unification du test et la sécurité du niveau circuit au niveau système, tout en passant par les cartes, tous les principes de base rappelés ici seront d'une grande utilité dans les chapitres à suivre.

CHAPITRE II

LE TEST "BOUNDARY SCAN"

II.1 Introduction

Il est connu que l'augmentation incessante de la complexité de circuits intégrés amène à des coûts de test qui ne sont pas acceptables pour la plupart des applications modernes. Cette situation nous oblige donc à prévoir le test dès la conception des circuits.

Dans le cas du test de cartes la réalité est encore plus dure, vu que la complexité et le coût du test d'un circuit VLSI se cumulent avec ceux d'autres circuits placés sur la carte et des connexions qui atteignent actuellement l'ordre de grandeur de 2000 à 3000 noeuds.

L'observabilité et la contrôlabilité étant évidemment difficiles à assurer dans un tel contexte, le test fonctionnel à partir des connecteurs de la carte perd du terrain dans les cartes à moyenne et grande complexité.

A cela s'ajoute le problème de la miniaturisation, introduit par les nouvelles technologies d'assemblage, comme par exemple la CMS (composants montés en surface), dont l'utilisation devient de plus en plus importante dans des produits commerciaux. L'objectif étant d'augmenter la densité des circuits et des connexions sur la carte, l'accès aux noeuds internes devient extrêmement limité.

De même, le test "in-circuit" perd aussi du terrain dans les cartes à moyenne et grande complexité, car la conception de lits à clous avec des espacements très réduits crée des problèmes techniques et économiques difficiles à résoudre.

Il s'avère donc nécessaire de prévoir le test de l'ensemble de la carte dès la conception des circuits qui la composent. Les connexions étant le moyen de communication entre circuits, seule l'adoption d'une méthode standard pourra résoudre le problème du test des cartes qui utilisent des circuits conçus et fabriqués par différentes sociétés.

La standardisation du test de cartes passe évidemment par le choix d'une méthode particulière de test qui est déjà assez mûre et bien acceptée par la plupart des industriels concernés. Ensuite, afin d'encourager son intégration dans tout produit, le rapport coût-bénéfice doit être le plus avantageux possible.

La technique du "boundary scan" (BS) remplit les conditions du côté bénéfice, tout en gardant un coût raisonnable□l'augmentation de la surface du circuit et du nombre de broches restent abordables pour la plupart des produits et des fabricants. Cette technique est la base du standard IEEE 1149.1 pour le test de cartes [IEE90] qui, malgré son approbation récente, connaît déjà le succès et la reconnaissance mis en évidence à travers une multitude de produits disponibles sur le marché [Mau94].

Ce chapitre portant sur le test de cartes sera donc consacré à la présentation au lecteur de ce standard et son utilisation [MaT90].

II.2 Le modèle de fautes

Bien que le modèle du collage logique [Fri71] ne soit pas capable de représenter toute faute réelle d'un circuit intégré [Wad78], il est toujours largement utilisé pour le test hors-ligne et donne dans la plupart de cas des résultats satisfaisants. Etant donné que dans ce chapitre et le suivant nous allons étudier le test hors-ligne de cartes, le modèle de collage sera donc adopté pour les fautes survenant à l'intérieur de leur circuits.

Le modèle de fautes pour une carte doit, tant que possible, bien représenter les défaillances observées dans les connexions. Les défaillances résultantes du processus de fabrication sont dûes d'une part à l'assemblage et d'autre part au soudage: l'assemblage est à l'origine de broches cassées, collées les une contre les autres, manquantes, etc; le soudage, quant à lui, est à l'origine de circuits-ouverts dûs à un manque éventuel de soudure, ou de courts-circuits dûs à des écoulements de soudure, etc.

De manière générale, les fautes des connexions peuvent être groupées dans les classes suivantes [JMF91]:

1) court-circuit: cette faute met en contact physique des connexions différentes. Le comportement obtenu dépend des caractéristiques de broches de sortie qui alimentent les connexions court-circuitées. Dans cette étude seuls les comportements déterministes, dont la détection est possible au niveau digital, seront considérés:

(a) court-circuit de type ET: la valeur résultante est un ET logique des valeurs de chacune des connexions concernées, la valeur logique '0' étant dominante;

(b) court-circuit de type OU: la valeur résultante est un OU logique des valeurs de chacune des connexions concernées, la valeur logique '1' étant dominante; et,

(c) court-circuit de type LE PLUS FORT ("strong driver"): la valeur logique imposée par la broche de sortie la plus puissante en termes de "fanout" va dominer.

Les technologies qui sont utilisées détermineront le(s) type(s) de court(s)-circuit(s) possible(s) sur la carte [JMF91].

2) collage logique: quoi qu'on fasse, la connexion défaillante reste collée à la valeur logique '0' ou à '1'.

3) circuit-ouvert : cette faute fait qu'une connexion est coupée en plusieurs parties qui restent déconnectées. Dans ce cas, la valeur logique appliquée à une extrémité de la connexion n'atteindra pas les autres extrémités. En général, les capacités parasites des extrémités déconnectées seront chargées (ou déchargées) à une valeur constante par la circuiterie de protection des broches, ce qui équivaut à un collage.

Puisque toutes ces fautes affectent différemment les connexions, leur détection demande des stratégies de test différentes. Par exemple, un court-circuit ou un collage affecte nécessairement la totalité d'une connexion, de telle sorte qu'il est suffisant d'appliquer des vecteurs de test à partir d'une seule sortie et d'observer une seule entrée de la connexion. Un circuit-ouvert par contre peut affecter tout simplement une partie d'une connexion, ce qui fait que le test doit être accompli en tenant compte de toute extrémité (entrée ou sortie) appartenant à la même connexion.

II.3 Du "scan path" au "boundary scan"

La technique du "scan path" [EiW78], étant à l'origine de la méthode "boundary scan" (BS), utilise la notion de chemin de balayage tant pour l'observabilité que pour la contrôlabilité de points internes à un circuit séquentiel. C'est-à-dire que les bascules du circuit sont reliées en registre à décalage pour permettre l'opération de décalage des données en vue de l'injection de vecteurs et de la récupération de réponses aux séquences de test appliquées.

L'observabilité et la contrôlabilité de points de test au niveau de la carte peuvent être obtenues par l'extension du chemin de balayage interne aux broches des circuits. La figure II.1 montre l'utilisation de la technique du "boundary scan", où on remarquera que pour chaque broche une cellule "boundary scan" est rajoutée [LeB84]. De manière semblable à la technique du "scan path", une cellule BS sera constituée de bascules et de multiplexeurs et toutes les cellules BS seront connectées en série pour former, dans un premier temps, le chemin de balayage du circuit et, ensuite, le chemin de balayage de la carte. Il devient donc possible de décaler des données de test par l'intermédiaire de broches d'entrée et de sortie BS tant au niveau des circuits qu'au niveau de la carte.

Comme chaque cellule BS est insérée entre une broche et la logique interne du circuit, on peut tout simplement observer des signaux passant (provenant de la logique interne ou des connexions), ou bien injecter des valeurs logiques aux entrées du circuit ou dans les connexions de la carte à travers le chemin de balayage "boundary scan". Vu que l'accès électronique est assuré, le test de la carte peut donc se faire sans se soucier des problèmes d'accès physique.

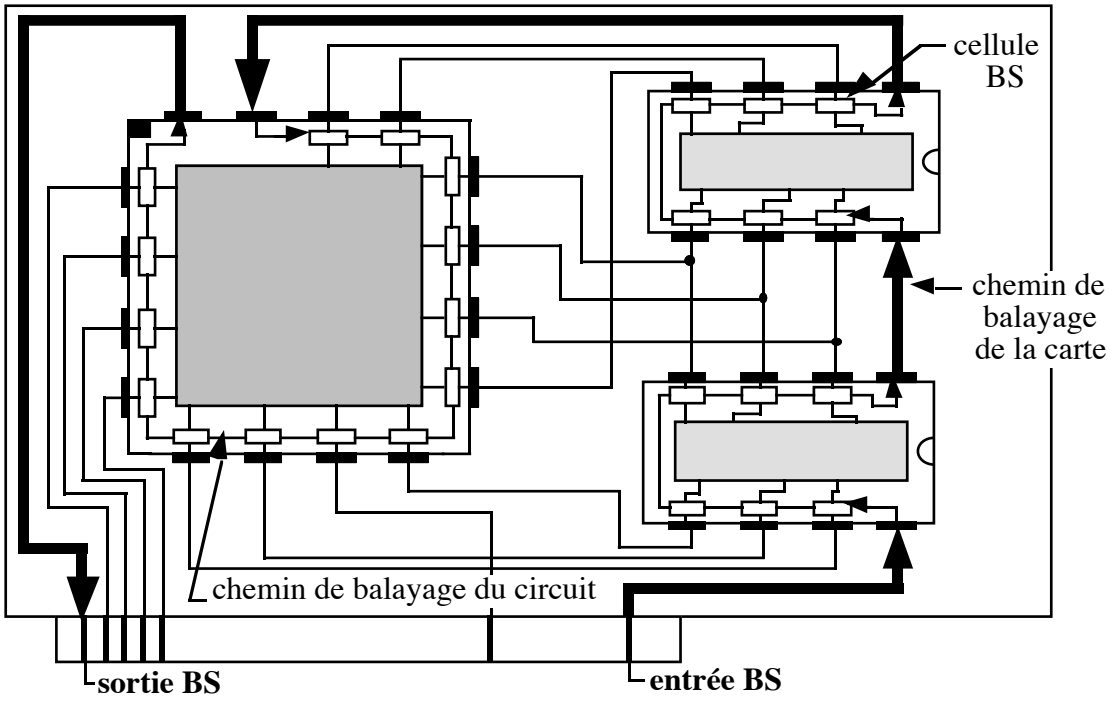


Figure II.1. Carte "boundary scan".

II.4 Les tests "boundary scan"

Dans un circuit intégré, les cellules BS sont connectées en série pour former un registre à décalage cheminant le long de son périmètre. La figure II.1 montre le chemin de balayage du circuit et ses broches d'entrée et de sortie. Pour contrôler les opérations à effectuer sur ce chemin, il est nécessaire aussi d'avoir des broches de contrôle et d'horloge. Celles-ci ne sont pas représentées dans la figure II.1.

La figure II.2 montre un exemple de cellule BS, celle-ci peut être intégrée à une broche unidirectionnelle d'entrée ou de sortie (à 2-états). A ce niveau-là les signaux de contrôle sont représentés. La cellule se compose de deux multiplexeurs et deux bascules. Les multiplexeurs font le routage à l'intérieur de la cellule□l'un contrôle le signal d'entrée et l'autre contrôle l'entrée de la bascule de décalage. Grâce au chargement du signal de sortie dans cette bascule et le décalage de son contenu vers la sortie de décalage, l'observabilité d'une broche est assurée par la cellule. La contrôlabilité est liée au fait qu'il est possible de remplacer le signal d'entrée par le contenu de la bascule de rétention auparavant chargée par l'intermédiaire de la broche d'entrée de décalage. En effet, cette deuxième bascule existe pour assurer que, pendant le décalage, la valeur de contrôle imposée au signal de sortie ne va pas changer. Pour les broches de sortie à 3-états deux cellules BS semblables à celle-ci sont nécessaires, l'une pour la sortie et l'autre pour le contrôle de troisième état. Dans le cas de broches bidirectionnelles trois cellules

sont nécessaires l'une pour l'entrée, l'autre pour la sortie et la troisième pour le contrôle de direction.

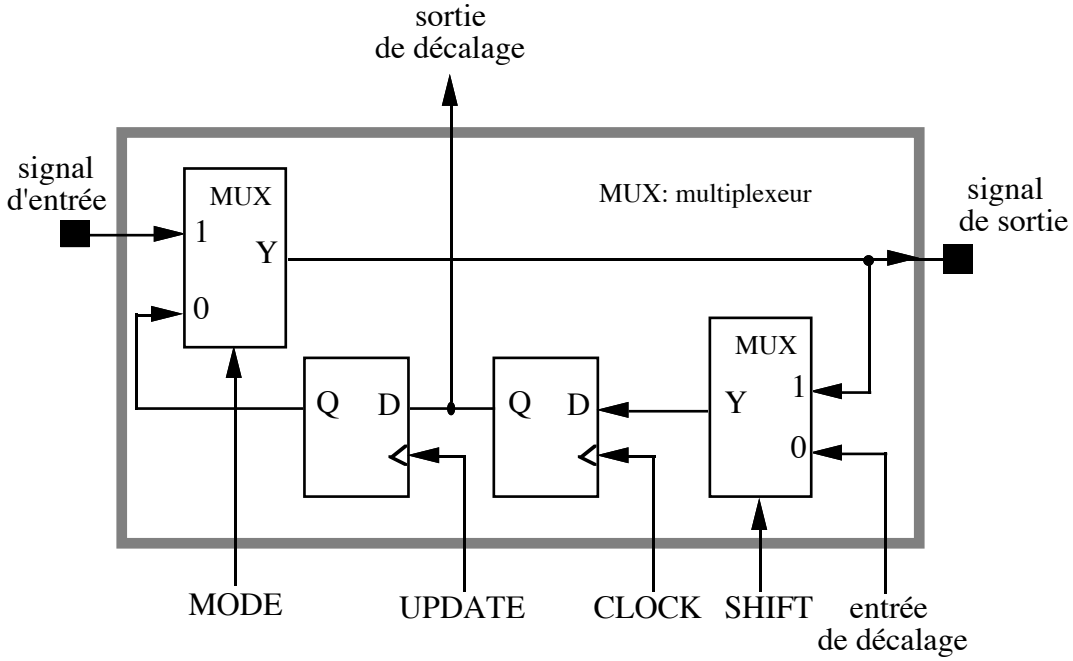
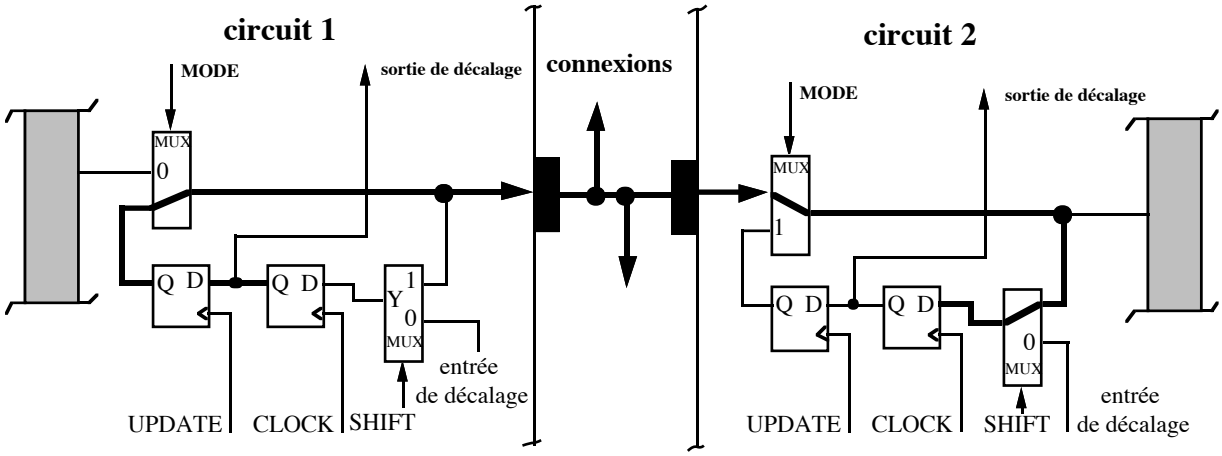


Figure II.2. Exemple de cellule BS.

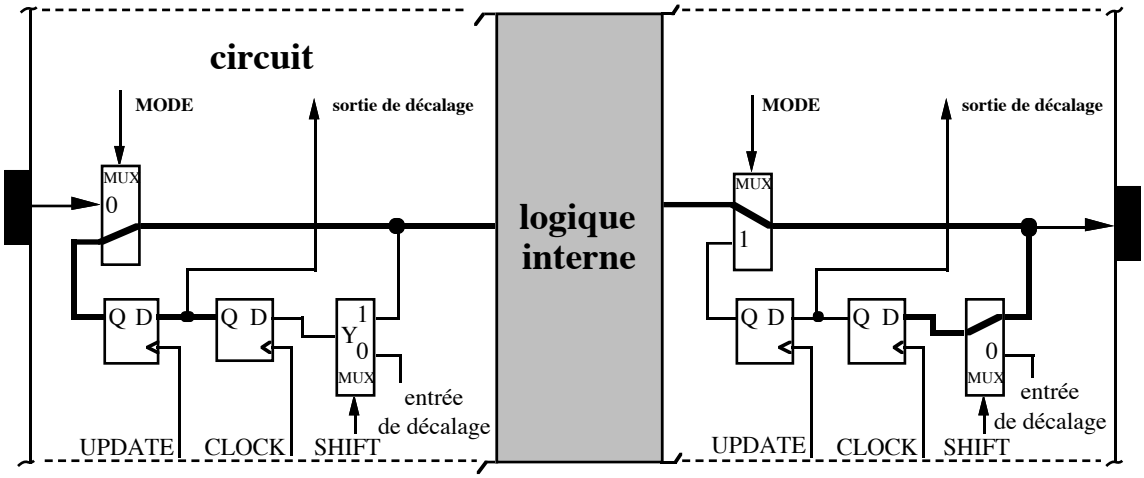
La disponibilité de ces cellules au long du périmètre du circuit fait qu'il est possible de réaliser trois types de test différents au niveau de la carte: le test externe, le test interne et le test d'échantillonnage. Ces trois tests sont décrits ci-après:

Le test externe consiste à vérifier les connexions entre les circuits. La figure II.3(a) montre les différents modes d'opération pour les multiplexeurs des broches d'entrée et de sortie des circuits. Dans ce mode les vecteurs de test sont fournis par les cellules BS de l'interface de sortie des circuits et capturés par les cellules BS de l'interface d'entrée des circuits.

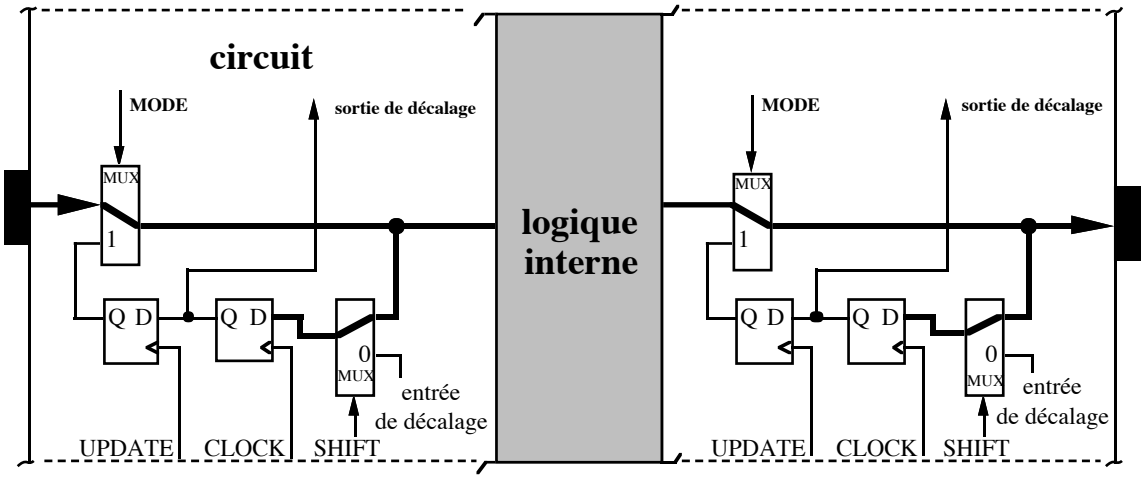
Le test interne, à son tour, consiste à appliquer des vecteurs de test pour vérifier la fonctionnalité de la logique interne des circuits. Dans ce cas les multiplexeurs sont programmés comme le montre la figure II.3(b), où on remarquera que les vecteurs de test sont fournis par les cellules BS de l'interface d'entrée du circuit et les réponses capturées par les cellules BS de son interface de sortie. Ce mode de test n'est pas très efficace surtout parce que les opérations de balayage sur la carte sont généralement très lentes dû au nombre de cellules BS et l'utilisation d'une fréquence d'horloge de décalage normalement plus faible que celle d'opération du système. L'avantage se trouverait donc dans les facilités d'accès et dans l'élimination des problèmes de conflits lorsqu'un testeur impose des tensions aux entrées d'un circuit sans tenir compte des valeurs fournies par les sorties des ses prédécesseurs [MaT90].



(a) test externe



(b) test interne



(c) test d'échantillonnage

Figure II.3. Modes de test "boundary scan".

Dans le troisième et dernier type de test possible, le test d'échantillonnage, les multiplexeurs sont programmés de façon à permettre le passage direct du signal d'entrée à la sortie de chaque cellule BS (figure II.3(c)). A part le délai introduit par le multiplexeur de la cellule BS placé sur le chemin de données, le test d'échantillonnage n'affecte pas le fonctionnement du circuit. Contrairement aux modes de test précédents, qui sont plutôt structurels, celui-ci a un caractère tout-à-fait fonctionnel. En effet, ce mode de test se rend très utile au moment de vérifier, à la vitesse d'opération normale du système, les interactions et les performances dynamiques des diverses parties composant la carte [Lef90].

II.5 Introduction au standard IEEE 1149.1-1990

Bien que la description technique du standard IEEE 1149.1-1990 détaille les caractéristiques de base des différents blocs de test, la réalisation pratique reste en partie ouverte à chaque concepteur. L'objectif de la description présentée est donc d'aider à la compréhension et de définir les règles d'implémentation à suivre pour assurer la compatibilité des protocoles de test entre tous les circuits montés sur une carte.

L'architecture du standard IEEE 1149.1-1990 [IEE90] est composée d'une partie contrôle (le port TAP et son contrôleur) et d'une partie opérative (registre d'instruction de test, registre de données et logique de routage), comme le montre la figure II.4. Dans les paragraphes suivants on va décrire chacune des sous-parties de l'architecture globale de ce standard.

II.5.1 Le port TAP et son contrôleur

C'est à travers le port de test TAP ("Test Acces Port") que tous les signaux d'horloge, de contrôle, les instructions et les données de test venant du monde extérieur atteignent et ressortent du standard intégré dans le circuit. Ces signaux y arrivent par l'intermédiaire de cinq broches nommées:

TDI ("serial Test Data Input"): entrée série d'instructions et de données de test;

[redacted] Output"): sortie série d'instructions et de données de test. Il s'agit [redacted]ts commandée par le signal ENABLE;

TCK [redacted]st [redacted]"): horloge de test, indépendante de l'horloge du système (SCK);

TMS [redacted] Select"): signal d'ordonnancement d'état du contrôleur TAP.

[redacted] input"): signal de reset asynchrone (optionnel).

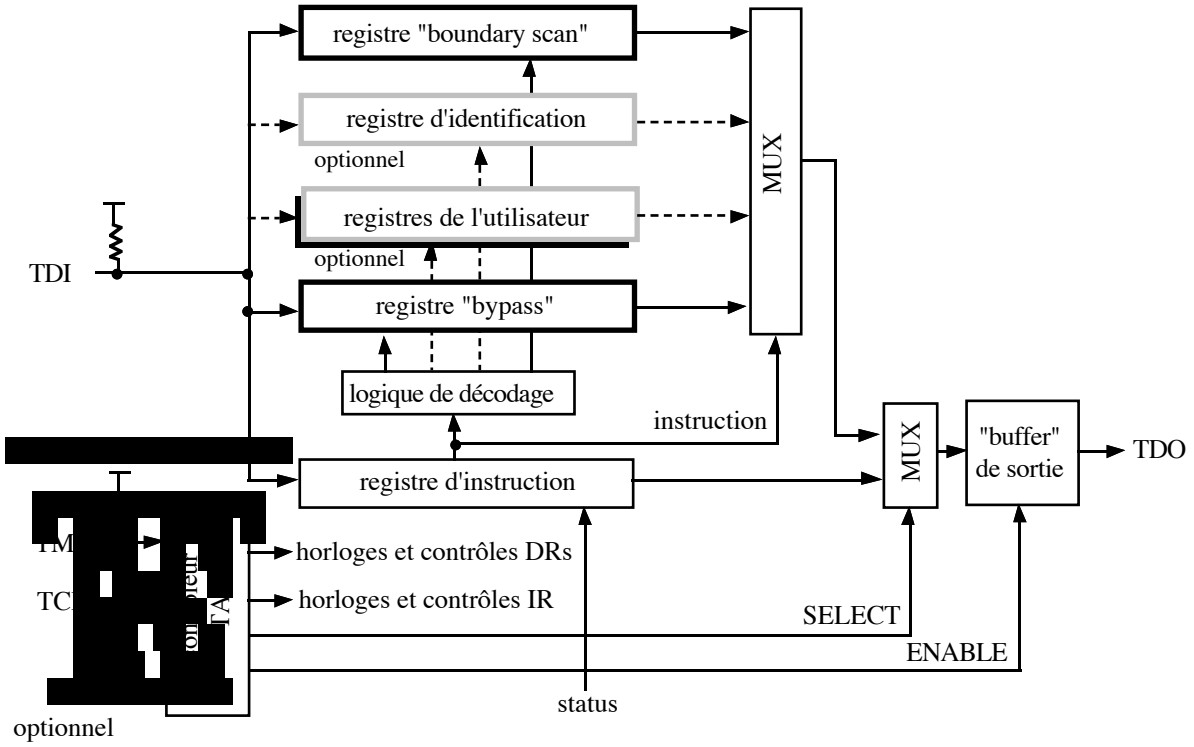


Figure II.4 L'architecture du standard IEEE 1149.1.

Tous les registres du standard sont des registres à décalage dont le contenu peut être modifié à partir des broches TDI et TDO et par l'intermédiaire de l'activation adéquate des signaux de contrôle TMS et TCK. Les signaux TDI et TMS sont échantillonnés sur le front montant de l'horloge TCK, alors que TDO atteint la sortie série sur le front descendant. Les broches TDI, TMS et TCK possèdent une résistance "pull-up" interne pour assurer l'état logique '1' sur les broches éventuellement déconnectées. Comme on verra par la suite, cela fait que le contrôleur TAP est initialisé dans un état sûr (reset synchrone) si le signal TMS reste à '1' pendant au moins cinq cycles d'horloge.

Les broches TDI et TDO servent au décalage de données et d'instructions, les registres du standard étant classés selon leur contenu (donnée ou instruction) et étant sélectionnés par le signal SELECT. Dans une carte, une broche TDO d'un circuit est connectée à la broche TDI du circuit suivant. Toutes les broches TMS d'une carte ainsi que les broches TCK sont normalement reliées ensemble. Cette liaison en parallèle permet le contrôle simultané de tous les circuits d'une carte intégrant le standard, c'est-à-dire que tous les contrôleurs TAP reçoivent toujours la même commande.

Le contrôleur TAP est une machine d'état dont les signaux TMS et TCK sont les entrées et les signaux de contrôle internes montrés figure II.4 sont les sorties. Le diagramme d'états de

ce contrôleur comporte 16 états, comme le montre la figure II.5. Ces états peuvent être groupés en trois classes selon les opérations réalisées : les états principaux, les états d'instruction et les états de données.

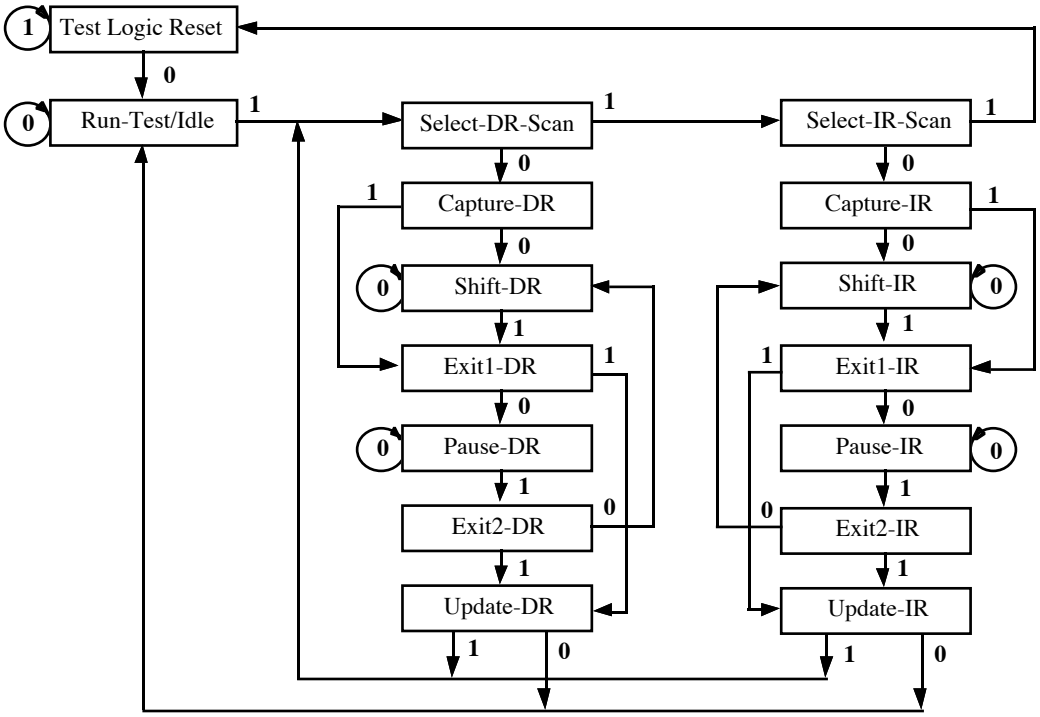


Figure II.5. Diagramme d'états du contrôleur TAP.

Les états principaux du contrôleur TAP sont quatre :

Test Logic Reset : Dans cet état la circuiterie de test est initialisée et le contrôleur est mis dans un mode qui n'affecte pas le fonctionnement du circuit. Où qu'il se trouve, le contrôleur atteindra cet état si TMS est maintenu à '1' pendant au moins cinq cycles d'horloge TCK, et il restera dans cet état tant que TMS restera à '1'. Si la broche TMS du circuit est par une raison ou une autre déconnectée, l'entrée se trouvera à la valeur logique '1' grâce à la résistance "pull-up" montrée figure II.4, menant le contrôleur vers cet état, où il restera indéfiniment.

Run-Test/Idle : Cet état peut servir à deux propos : soit la logique de test se met carrément en repos (état "Idle") entre, par exemple, deux opérations de décalage, soit une logique de test intégrée dans le circuit se met en marche et exécute une instruction d'autotest (état "Run-Test").

Select-DR-Scan : C'est un état où l'on choisit le type de registre qui va subir une opération de décalage. Si TMS se met à '0', le choix d'un registre de données (celui sélectionné par le registre d'instruction) est fait.

Select-IR-Scan : C'est aussi un point de décision : si TMS se met à '0', le registre d'instruction entre en mode décalage; si TMS reste à '1', le contrôleur TAP retourne à l'état Test Logic Reset.

Mis à part le registre utilisé, une opération de décalage sera accomplie de manière identique qu'il s'agisse d'une instruction ou de données. Il est donc suffisant que nous décrivions un seul des groupes d'états restant (soit celui d'instruction, soit celui de données). Dans la suite nous allons noter XR un registre pouvant être le registre d'instruction (IR) ou un registre de données de test (DR). Les états du contrôleur TAP qui portent sur l'opération de décalage sont douze, six dans le groupe d'instruction et six dans le groupe de données :

Capture-XR : Dans cet état le registre X sélectionné est chargé en parallèle avec le contenu qui sera décalé vers l'extérieur du circuit (observation). C'est-à-dire que si XR est l'IR, l'état du test réalisé précédemment sera chargé dans IR ("status" dans la figure II.4), ou bien si XR est un DR, la réponse due au test sera chargé dans DR.

Shift-XR : Dans cet état s'effectue l'opération de décalage entre TDI et TDO du registre sélectionné.

Exit1-XR : C'est un état où la décision de terminer ou non l'opération de décalage est prise.

Pause-XR : Cet état permet d'arrêter momentanément le décalage. Ceci est très utile dans le cas où la synchronisation avec un testeur est nécessaire. Le contrôleur TAP reste dans cet état tant que TMS est maintenu à '0'.

Exit2-XR : C'est encore un état où la décision de terminer ou non l'opération de décalage est prise.

Update-XR : Dans cet état le registre sélectionné X est chargé en parallèle avec le contenu qui a été décalé à l'intérieur du circuit (contrôle). C'est-à-dire que si XR est l'IR, une nouvelle instruction de test sera activée, ou bien si XR est un DR, un nouveau vecteur de test sera chargé dans DR.

D'après tout ce que nous avons vu jusqu'à maintenant, nous pouvons conclure qu'une instruction de test va sélectionner un registre de la partie opérative du standard et certaines

opérations seront déclenchées et accomplies. Dans les paragraphes suivants on va étudier chacun des registres que intègre le standard et les instructions associées à ces registres.

II.5.2 Le registre d'instruction

Le registre d'instruction est utilisé pour sélectionner un registre de données de test et, par conséquent, le type de test à réaliser. Au début du cycle de décalage d'instruction, il sera chargé avec l'état du test précédent. Ensuite, cet état sera décalé vers TDO en même temps qu'arrivera la nouvelle instruction par TDI.

Le registre d'instruction doit compter au moins 2 cellules élémentaires (2 bits). Chaque cellule élémentaire sera composée d'un registre à décalage acceptant des charges parallèles et d'un "latch" de sortie. La figure II.6 montre une implémentation possible où le registre à décalage reçoit un mot d'état en parallèle et des instructions en série à partir de la broche TDI.

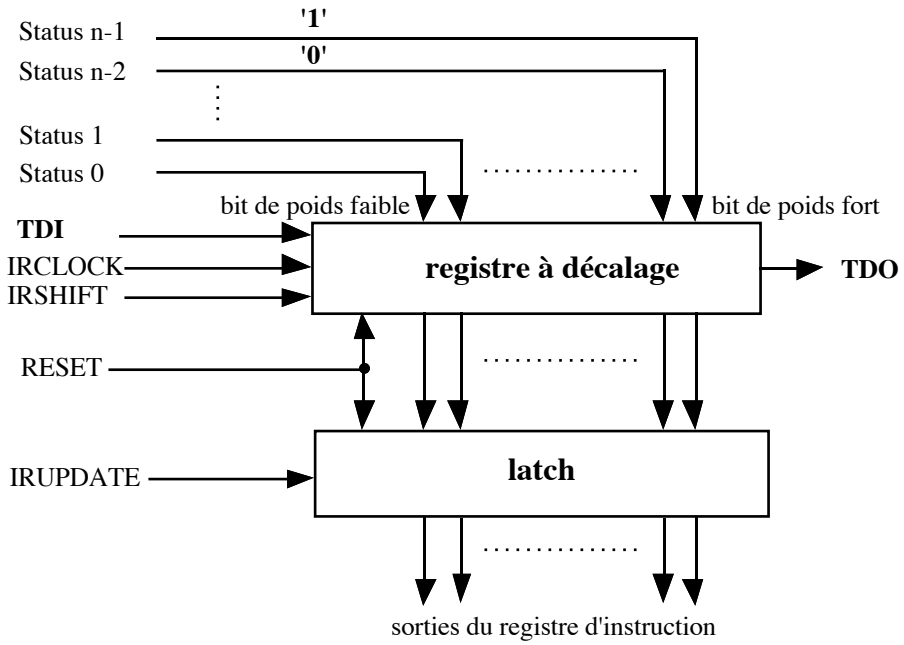


Figure II.6. Le registre d'instruction.

Les deux bits d'état définis dans la figure sont d'utilisation obligatoire et doivent toujours contenir les valeurs '01' indiquées. Comme démontre la figure II.7, cela permet la détection et la localisation de certaines fautes sur le chemin de balayage de la carte à partir de l'observation de sa broche de sortie TDO.

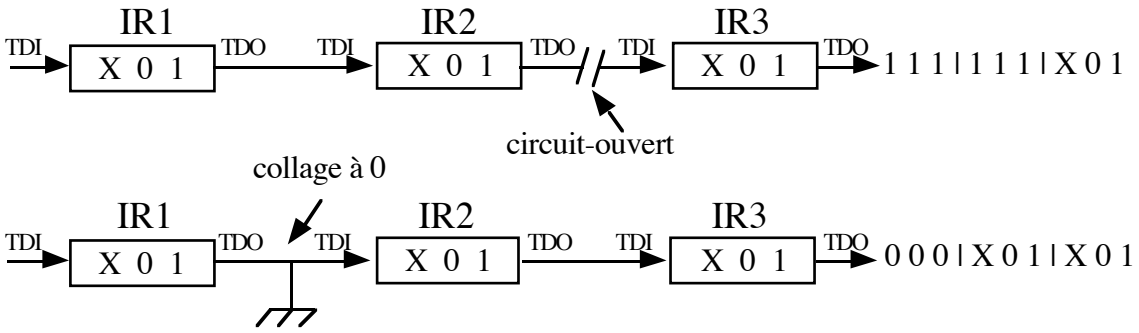


Figure II.7. Localisation d'une faute sur le chemin de balayage de la carte.

Finalement, le "latch" dit de sortie sera chargé avec l'instruction à exécuter dans l'état "Update-IR" du contrôleur TAP et deviendra insensible à toute opération de décalage jusqu'à ce qu'une nouvelle instruction soit disponible à ses entrées.

II.5.3 Les registres de données de test

Chaque instruction de test programmée dans le registre d'instruction doit sélectionner un registre de données pour son exécution. Deux registres sont d'utilisation obligatoire dans le standard : le registre "bypass" et le registre "boundary scan". Un registre d'identification et d'autres peuvent, en option, être implémentés par l'utilisateur.

II.5.3.1 Le registre "bypass"

Le registre "bypass" court-circuite le circuit de façon à l'isoler du chemin de balayage de la carte. Ce registre sera composé d'une seule bascule intercalée entre les broches TDI et TDO et sera chargée à '0' dans l'état "Capture-DR" du contrôleur TAP. On associe à ce registre une instruction de même nom. Une fois programmée dans le registre d'instruction, BYPASS peut être très utile pour le partitionnement de la carte en vue du test ou du diagnostic. Dans la figure II.8 on remarquera que l'accès à un circuit cible de la carte peut se faire en parcourant un chemin de balayage réduit par l'activation du registre et de l'instruction BYPASS.

II.5.3.2 Le registre d'identification

Ce registre est un registre optionnel qui comporte l'identification du constructeur, du circuit, de la version du circuit utilisée, etc, et auquel l'instruction IDCODE est associée. Il sera sélectionné par défaut dans l'état "Test Logic Reset" du contrôleur TAP et, au cas où il n'est pas disponible dans le circuit, le registre "bypass" prendra sa place. Etant donné que, de la même façon que dans le registre d'instruction, le bit de poids fort du registre d'identification doit obligatoirement contenir un '1' logique, l'identification de ce registre par rapport au registre "bypass" est assurée, car ce dernier est toujours chargé à la valeur logique '0'.

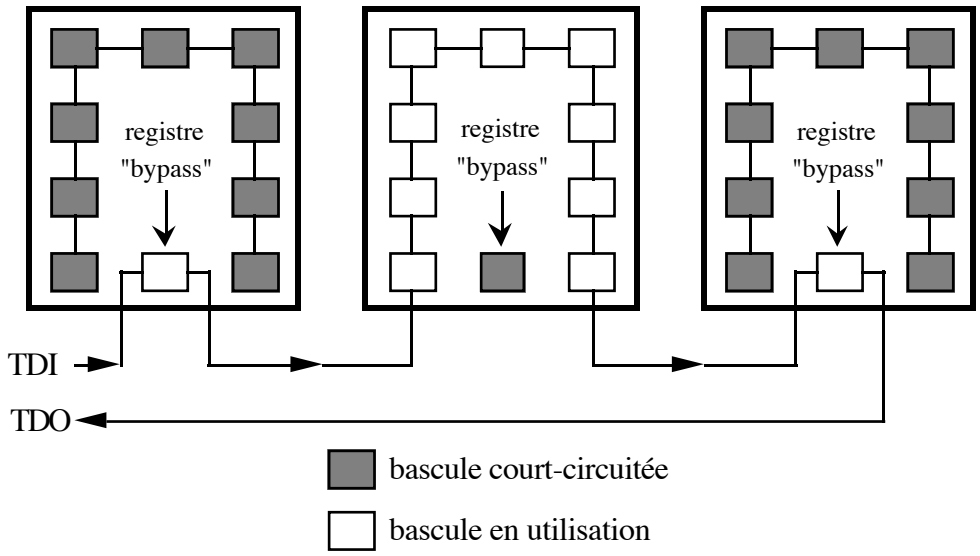


Figure II.8. L'utilisation du registre "bypass".

L'intégration d'un tel registre dans le circuit est très utile pour le réglage de l'exécution du programme de test en fonction de la version utilisée du circuit, pour la vérification de l'assemblage des circuits sur la carte, pour l'identification de la carte à tester lorsqu'il y a, dans la chaîne de test, une famille de cartes compatibles par rapport aux connecteurs, etc.

II.5.3.3 Le registre "boundary scan"

Comme le montre la figure II.1, le registre "boundary scan" consiste en l'enchaînement de plusieurs cellules BS chacune connectée à une broche de l'interface d'entrée ou de sortie du circuit. Ces cellules BS peuvent être connectées à des broches de données, de contrôle ou même d'horloge, à des broches unidirectionnelles ou bidirectionnelles, à des broches de sortie à 2-états, à 3-états ou collecteur-ouvert [IEE90]. On peut aussi associer des cellules BS à des signaux internes au circuit lorsqu'une interface analogique-digital, par exemple, y est intégrée [MaT90]. Dans cette thèse nous allons nous limiter à l'exemple de cellule BS donné figure II.2.

Quelle que soit la réalisation adoptée, il est obligatoire qu'une cellule BS puisse accomplir les modes de test externe et d'échantillonnage, le test interne étant optionnel. A ces types de test on associe respectivement les instructions EXTEST, SAMPLE/PRELOAD et INTEST□

EXTEST□ Cette instruction permet l'application à partir des registres BS de sortie d'une séquence de vecteurs pour le test de connexions ou d'un groupe de composant n'ayant pas le standard. La récupération des résultats est faite par l'intermédiaire des registres BS d'entrée. L'exécution d'EXTEST ne doit pas affecter la logique interne des circuits.

SAMPLE/PRELOAD□ Cette instruction permet la prise d'un échantillon de l'état de la carte (SAMPLE) suivie de la précharge d'une valeur donnée sur le registre BS (PRELOAD). Côté SAMPLE, cette instruction se rend utile à la réalisation d'un test fonctionnel; coté PRELOAD, la précharge d'un état sûr dans le registre BS avant de déclencher n'importe quel type de test est assurée. L'exécution de SAMPLE/PRELOAD ne doit pas affecter le fonctionnement normal de la carte.

INTEST□ Cette instruction permet l'application à travers les registres BS d'entrée d'une séquence de vecteurs pour le test de la logique interne des circuits. La récupération des résultats est faite par l'intermédiaire des registres BS de sortie. D'une part l'exécution d'INTEST doit assurer un état sûr aux connexions de sortie des circuits par l'intermédiaire de leur registres BS de sortie. D'autre part son exécution ne doit pas être affectée par les tensions présentes aux entrées des circuits□les registres BS d'entrée, chargés à travers le chemin de balayage de la carte, doivent donc rester insensibles à leur entrées parallèles liées aux connexions extérieures.

Chacune de ces instructions fait que l'application du test concerné est réalisée pas-à-pas□ l'injection d'un vecteur de test est suivie de l'observation du résultat donné, puis le vecteur suivant est appliqué et ainsi de suite. Tout balayage de la carte correspondra donc au décalage simultané du résultat de l'application d'un vecteur de test (vers l'extérieur) et du vecteur à appliquer ensuite (vers l'intérieur).

II.5.3.4 Les registres de l'utilisateur

Les registres de l'utilisateur sont des registres de données spécifiques permettant l'accès à toute ressource de test de la logique interne du circuit. Ces ressources peuvent être un "scan path" interne ou des registres du type autotest intégré ("built-in self-test"), par exemple.

Mis à part les décalages, toute autre opération sur un registre de l'utilisateur doit se passer dans l'état "Run-Test/Idle" du contrôleur TAP. Ainsi, afin de permettre l'organisation des divers types de test de circuits, il est important d'informer le concepteur de la carte du nombre de cycles d'horloge nécessaires pour réaliser chacun de ces tests.

Dans une carte où plusieurs circuits s'autotestent en même temps, par exemple, il faut que tous les contrôleurs TAP restent dans l'état "Run-Test/Idle" assez longtemps pour permettre au circuit demandant le plus de cycles d'horloge de test de finir son autotest. Finalement, il faut

assurer que le résultat de l'autotest d'un circuit qui a fini avant les autres ne change pas jusqu'à ce que le décalage vers l'extérieur de tous les résultats ait abouti.

En ce qui concerne ce type de registre de données, le standard propose l'instruction optionnelle RUNBIST qui doit fournir le résultat d'un autotest dans un registre défini par le concepteur. Ce registre sera connecté au chemin de balayage de la carte dans l'état "Shift-DR". Finalement, RUNBIST doit garantir que le circuit soit isolé du reste de la carte et que les connexions soient mises dans un état sûr pendant son exécution.

II.6 Le test de cartes 100% "boundary scan"

L'intégration du standard IEEE 1149.1 étant assurée dans tout circuit intégré, on ne trouvera que des cartes 100% "boundary scan" dans des produits. Il est clair qu'il se passera encore longtemps jusqu'à ce que cette situation idéale se présente, ce qui n'invalide absolument pas les objectifs du standard. Celui-ci est prêt tout-de-même à servir dans des étapes intermédiaires de l'évolution de la conception de circuits en vue du test de cartes.

Dans ce contexte, ce paragraphe sera consacré à l'étude du cas idéal, tandis que dans le chapitre suivant nous proposerons, entre autres, une méthode pour le test et le diagnostic de cartes partiellement "boundary scan".

Une séquence capable de vérifier toutes les pièces composant une carte "boundary scan" est proposée dans [TuY90]. Il s'agit d'un test structurel pour la carte composé de trois grandes étapes : l'initialisation et la vérification de la logique de test, le test des connexions et le test des circuits. Un certain nombre de pas composent chacune de ces étapes, ils sont résumés ci-après :

1. initialisation et vérification de la logique de test :
    1.1. atteindre l'état "Test Logic Reset" dans tous les contrôleurs TAP sur la carte (5 cycles de TCK avec TMS à '1');
    1.2. tester le chemin de balayage de la carte à travers les registres d'instruction (la séquence attendue est de type 'X…X01X…X01…X…X01');
    1.3. précharger les registres BS de sortie avec un état sûr pour éviter des dommages causés par des conflits de bus (utilisation de l'instruction PRELOAD);
    1.4. examiner l'assemblage des circuits sur la carte (utilisation de l'instruction IDCODE);
    1.5. vérifier l'opération des registres de donnés (utilisation des instructions BYPASS, etc);

2. test des connexions□
    2.1. appliquer à toutes les connexions une séquence de test pour la détection des fautes du modèle (utilisation de l'instruction EXTEST);
    2.2. en cas de faute, appliquer aux connexions suspectes une séquence de test pour le diagnostic de fautes (utilisation conjointe des instructions EXTEST et BYPASS);

3. test des circuits□
    3.1. déclencher l'autotest sur les circuits qui présentent cette fonctionnalité (utilisation conjointe des instructions RUNBIST et BYPASS);
    3.2. vérifier la logique interne des autres circuits (utilisation conjointe des instructions INTEST et BYPASS).

Dans les paragraphes qui suivent nous allons, d'une part, regarder plus profondément les problèmes associés à chacune de ces étapes et, d'autre part, présenter quelques unes des solutions existantes pour ces problèmes.

II.6.1 Le test de l'infrastructure "boundary scan"

L'infrastructure nécessaire au test "boundary scan" consiste en un certain nombre d'éléments□les lignes d'alimentation, le standard intégré dans chaque circuit et les réseaux TCK-TMS, TDI-TDO et TRST de la carte (ce dernier n'est pas toujours disponible).

Dans [JoH91] les fautes affectant ces éléments sont étudiées dans un contexte où seules les fautes simples peuvent se produire. En présence d'une faute, chacun de ces éléments réagissant de manière particulière demande des stratégies de test différentes.

Une faute dans les lignes d'alimentation, par exemple, va faire que le comportement de la carte est non déterministe, que des oscillations parviennent à sa broche TDO, etc. Seules plusieurs mesures de voltage et de courant sur la carte peuvent aider à l'identification de la vraie raison du mauvais fonctionnement.

En ce qui concerne les fautes affectant le standard BS intégré dans les circuits, on compte normalement les détecter par l'intermédiaire de leur propre utilisation pendant le test de la carte. Il est sûrement raisonnable de croire que l'utilisation pure et simple du standard est un test en soi qui donne une bonne couverture de fautes dans ses registres et qui fournit beaucoup de données pour le diagnostic à ce niveau. Pourtant l'utilisation du standard n'est-elle peut-être pas suffisante pour assurer un test convenable à son contrôleur TAP et en même temps assurer le minimum de données pour permettre que la tâche de localisation de fautes ne soit pas trop

compliquée. C'est pourquoi des travaux sont apparus qui proposent des solutions au problème du test du contrôleur TAP [DUY89, Esc92].

Les fautes dans les réseaux TCK-TMS, comme toute faute affectant une horloge, mènent en général à un comportement qui indique que rien ne marche sur la carte. Pourtant, il se peut qu'un court-circuit entre une connexion quelconque et l'un de ces signaux ne soit pas détecté si ces derniers sont fournis par un testeur externe (des signaux forts). Dans ce cas on espère avoir la chance de détecter la faute pendant le test de l'autre connexion. D'autre part, la détection d'un court-circuit entre TCK et TMS sera assurée par la méthode de reset avec TMS à '1' pendant 5 cycles de TCK.

En ce qui concerne le test des réseaux TDI-TDO, il peut se faire à l'aide du décalage du registre d'instruction. Malheuresement on ne teste pas la connexion TDI d'entrée de la carte et on n'assure pas la détection des courts-circuits entre les diverses branches du chemin de balayage. L'injection d'une séquence de test complémentaire à celle des registres d'instruction s'avère donc nécessaire. Une séquence de type '…11110000…', où le nombre de '0's et de '1's sont égaux au nombre de bits du registre d'instruction le plus long dans le chemin de balayage de la carte, assure la détection de toute faute dans les réseaux TDI-TDO.

Finalement, quand le réseau TRST est disponible sur la carte la vérification de la connexion des broches TRST des circuits à ce réseau demande une procédure particulière de test. La détection de ce genre de faute peut être assurée par une procédure basée sur la sélection correcte ou incorrecte d'un registre de l'architecture du standard. Le registre d'identification ou le registre "bypass" sera sélectionné en cas de reset réussi, sinon un autre registre (par exemple le registre d'instruction, si le contrôleur TAP se trouve dans la branche d'instruction lorsque le signal TRST est activé) sera sélectionné. Les contenus décalés vont révéler si la bonne sélection de registre a été faite.

Une procédure de test couvrant tous les aspects mentionnés ci-dessus plus la vérification d'assemblage de la carte est proposée dans [JoH91] et donnée ci-après.

1. reset en utilisant la méthode TMS à '1' pendant 5 cycles de TCK;

2. décalage des registres d'instruction suivi d'une séquence complémentaire;

3. décalage des registres d'identification;

4. test du réseau TRST.

L'ordre d'exécution de ces pas ne peut pas être modifié, vu que le test réalisé dans chacun des pas ne pourra pas fournir d'information utile au diagnostic si les éléments qui ont été activés dans les pas précédents ne fonctionnent pas correctement.

II.6.2 Le test de connexions en prévision du diagnostic

Le test des connexions étant sans doute l'application la plus importante de la technique du "boundary scan", de nombreuses publications sont apparues dans ce domaine. Dans la majorité de ces travaux un modèle considérant les fautes multiples est utilisé.

Pendant plusieurs années, le diagnostic de courts-circuits s'est présenté comme un thème de recherche très attirant, au détriment des autres types de fautes. Ceci s'explique surtout du fait que les courts-circuits étant des fautes d'interaction entre connexions, ils s'avèrent plus difficiles à diagnostiquer. Les collages logiques et les circuits-ouverts (dans la plupart des cas pouvant être modélisés comme des collages) affectent les connexions de manière isolée, ce qui fait que ces fautes sont facilement détectées et diagnostiquées par l'application de n'importe quelle séquence qui assure au moins un '0' et un '1' pour chacune des connexions de la carte.

Afin d'illustrer le genre de problème trouvé lors du diagnostic de fautes de type court-circuit analysons d'abord l'exemple donné figure II.9. Le court-circuit représenté dans la figure est de type ET. Supposons l'application à cet ensemble de connexions de la séquence de comptage binaire [Kau74] indiquée à gauche dans la figure. En présence d'un court-circuit entre les connexions 3 et 6, la séquence réponse qu'on obtient est montrée à droite. On remarquera que pour l'exemple, bien que la connexion 2 ne participe pas au court-circuit, elle appartiendra au groupe de connexions candidates à la faute. Ce phénomène est connu comme "aliasing" syndrome [JaY89] dans la littérature. La séquence de comptage binaire assure la détection de courts-circuits mais ne peut pas assurer le diagnostic de "aliasing" syndromes.

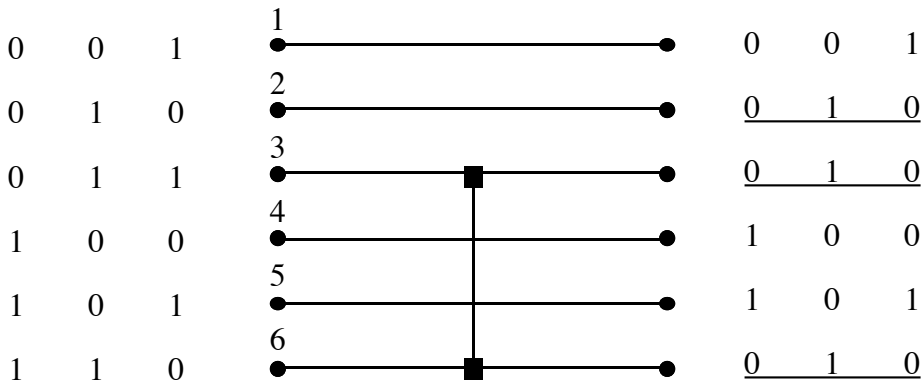


Figure II.9. "Aliasing" syndrome.

Prenons ensuite l'exemple donné figure II.10. Les courts-circuits représentés dans la figure sont tous les deux de type ET. Supposons l'application à ces connexions de la séquence de comptage binaire suivie de son complément [Wag87] indiquée à gauche dans la figure. En présence d'un court-circuit entre les connexions 1 et 4, et d'un autre court-circuit entre les connexions 2 et 3, la séquence réponse qu'on obtient est montrée à droite dans la figure. On remarquera que, bien que les deux courts-circuits soient indépendants l'un de l'autre, il est impossible de les distinguer d'après les réponses obtenues au test appliqué. Ce phénomène est connu comme "confounding" syndrome [JaY89] dans la littérature. La séquence de comptage binaire suivie de son complément assure la détection de courts-circuits et le diagnostic de "aliasing" syndromes, mais n'assure pas le diagnostic de "confounding" syndromes.

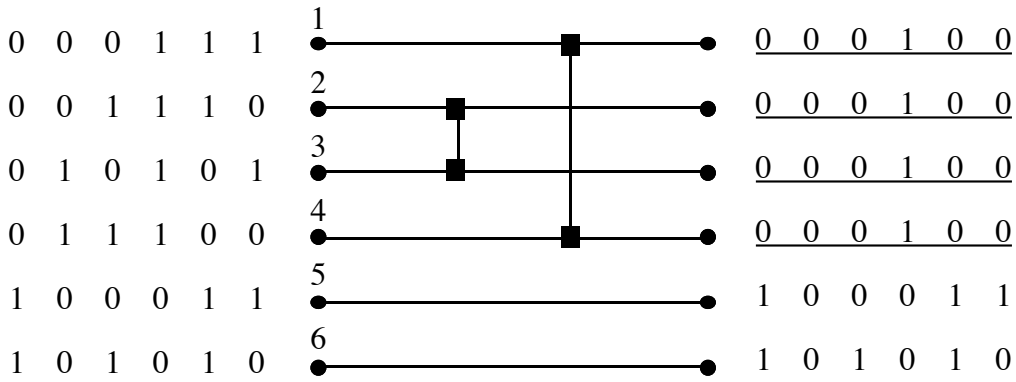


Figure II.10. "Confounding" syndrome.

Diverses séquences de test présentant des longueurs et des capacités de diagnostic différentes ont été proposées par différents auteurs. Celles présentées dans [CLW90], par exemple, mènent à des faibles temps d'application de test tout en assurant le diagnostic de "aliasing" syndromes, mais ne couvrent pas les "confounding" syndromes. Les séquences proposées dans [YaJ89], par contre, vont de la minimisation du nombre possible de syndromes à leur couverture à 100%, passant d'une approche à l'autre au prix d'un temps croissant d'application de test. Finalement, les séquences présentées dans [HRA88], de type '01'-baladeur, couvrent les deux types de syndromes sans avoir besoin d'informations additionnelles sur la structure du réseau de connexions (contrairement à [YaJ89]), mais le nombre de vecteurs de test est égal au nombre de connexions sur la carte. Nous reviendrons aux séquences de type '01'-baladeur un peu plus loin dans ce même paragraphe.

En plus des problèmes liés aux syndromes de type "aliasing" et "confounding", des fautes peuvent se produire pour lesquelles le diagnostic est impossible avant une première

réparation [LiB91]. Ceci est dû au masquage de certaines fautes par d'autres, comme le montre la figure II.11. Pour la configuration (a), le circuit ouvert ne peut même pas être détecté avant que l'un de deux courts-circuits ne soit réparé; de même, pour la configuration de fautes (b), le court-circuit ne peut être détecté avant qu'au moins l'un des deux circuits-ouverts ne soit réparé.

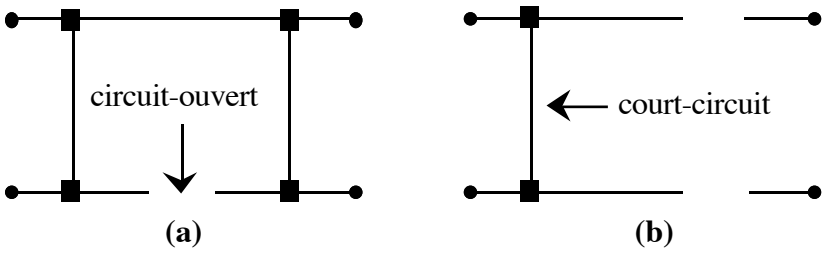


Figure II.11. Des fautes non diagnosticables.

Malgré tous les obstacles que nous venons de mentionner, Lien et Breuer [LiB91] ont déterminé une séquence universelle minime pour le diagnostic en un pas de toute faute diagnosticable dans les connexions. La figure II.12 montre une telle séquence pour un ensemble de trois connexions.

1	1	1	0	0	0	0	1
1	1	0	1	0	0	1	0
1	0	1	1	0	1	0	0

court-circuit ET court-circuit OU

Figure II.12. Séquence universelle minime (3 connexions).

La première moitié de cette séquence correspond aux courts-circuits de type ET, tandis que la deuxième adresse les courts-circuits de type OU. Chacune de ces moitiés est composée respectivement d'un vecteur "tout-à-'1'" et une séquence '0'-baladeur, et d'un vecteur "tout-à-'0'" et une séquence '1'-baladeur, ce qui totalise une longueur de 2(N+1) vecteurs (où N est le nombre de connexions sur la carte). On remarquera qu'une séquence X-baladeur reproduit tous les mots de type 1(X)-parmi-N(N) séquence-baladeur (resp. '1'-baladeur) assure le diagnostic des divers courts-circuits possibles de type ET (resp. de type OU), tandis que les vecteurs tout-à-'1' (resp. tout-à-'0') permet le diagnostic entre un court-circuit de type ET (resp. de type OU) et des collages-à-'0' (resp. collages-à-'1') affectant, tous les deux, toutes les connexions de la carte.

Vu que les séquences permettant le diagnostic de toute faute dans les connexions sont très longues et que celles n'assurant que la détection présentent des longueurs beaucoup plus faibles, on pourrait envisager de réaliser le diagnostic en deux étapes. Ceci est en effet proposé dans la procédure [TuY90] discutée au début du paragraphe II.6. On utilise une première séquence pour la détection des connexions suspectes et une deuxième séquence pour le diagnostic parmi les connexions candidates à la faute. Bien qu'une telle approche mène de manière générale à une application plus rapide du test, les algorithmes associés sont normalement beaucoup plus complexes car la deuxième séquence est une fonction des résultats de l'application de la première séquence (algorithmes adaptatifs). Le lecteur trouvera des propositions d'algorithmes pour le diagnostic de connexions en deux pas dans [GoM82, JaY89, CLW90].

II.6.3 L'association BIST-"boundary scan"

Les techniques dites d'autotest intégré ou BIST fournissent la possibilité d'intégrer du test dans les circuits. Plusieurs schémas différents existent qui offrent une couverture élevée de fautes à des faibles surcoûts en surface, assurant ainsi des moyens efficaces pour le test de fin de fabrication de circuits intégrés.

Afin de simplifier la détection et le diagnostic de fautes, l'association BIST-"boundary scan" devient crucial pour le test de circuits sur la carte [LeB84]. On veut par cette association réduire le temps nécessaire à l'application du test ainsi que la capacité mémoire nécessaire au testeur.

La proposition du standard IEEE 1149.1 a été suivie de plusieurs travaux de recherches où le BIST est convenablement associé à la technique du "boundary scan" [GlB88, UsE89, NKL91, TRB91].

D'une part, le standard encourage les concepteurs à intégrer dans les circuits des capacités pour leur autotest. L'instruction RUNBIST crée les moyens pour le démarrage d'une procédure d'autotest et pour l'accès à des registres de l'utilisateur afin de vérifier les résultats du test appliqué. Des recommandations sont faites pour que tous les autotests intégrés dans un même circuit soient réalisés dans l'état "Run-Test/Idle" du contrôleur TAP en réponse à l'activation de RUNBIST. Le décalage de données pour l'initialisation des circuits d'autotest ne doit pas être prévu, tout doit se faire localement selon la philosophie du test 100% intégré.

D'autre part, l'architecture BIST-BS proposée dans tous les travaux mentionnés précédemment (figure II.13) est basée sur l'utilisation du registre "boundary scan" d'entrée comme générateur de vecteurs de test et le registre "boundary scan" de sortie comme analyseur de signature. Un chemin série ("scan path") est aussi prévu pour le contrôle et l'observation de

noeuds internes au circuit. Le partage des structures de test entre le BIST et le BS mène à une réduction évidente du surcoût en surface de silicium.

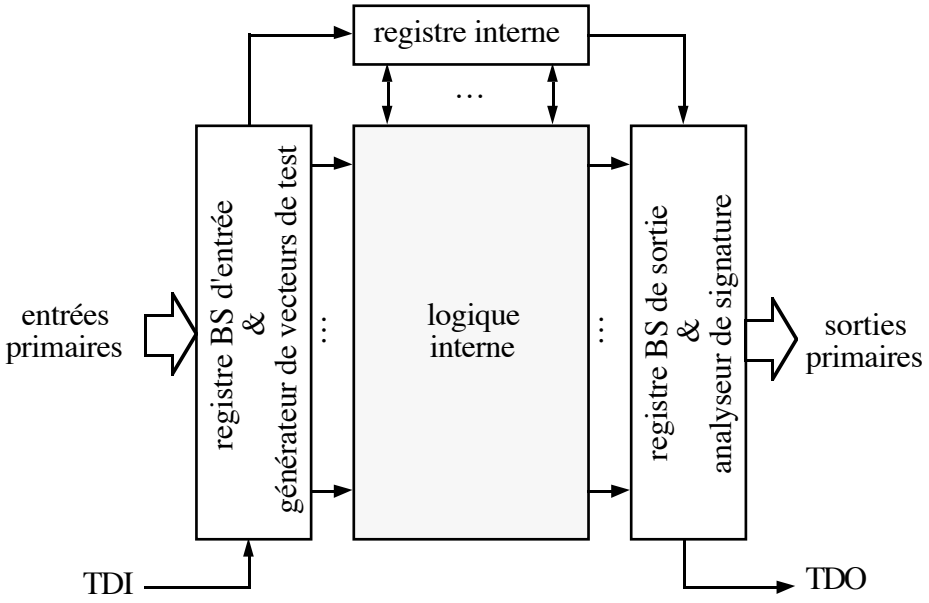


Figure II.13. L'architecture BIST-BS.

II.7 Conclusion

Prévoir le test de la carte dès la conception de ses circuits est certainement la meilleure solution pour surmonter la complexité croissante des tests suivant l'assemblage et du test de maintenance. Le standard IEEE 1149.1 basé sur la technique du "boundary scan" fait sans doute le premier pas vers l'autotestabilité des systèmes. L'efficacité de son mode de test externe est indiscutable, et son ouverture vers des solutions de l'état de l'art pour l'autotest des circuits intégrés est une réalité.

Si d'une part il est clair que l'apport de facilités de test pour la carte est incontestable dans ce standard, d'autre part la question de la rentabilité économique est à évaluer vis-à-vis de l'investissement dans la conception. Des études d'évaluation réalisées chez les industriels [TFH92] témoignent en faveur de l'intégration du standard et du BIST dans les circuits, et sont à l'origine des conclusions suivantes□

1) lorsque les fautes d'assemblage dominent le spectre de fautes de fabrication, en général l'implémentation du standard peut être justifiée facilement vis-à-vis des coûts de test;

2) lorsque des fautes fonctionnelles difficiles à diagnostiquer se présentent fréquemment, le fait de prévoir du BIST dans des circuits fait que le coût de la conception en vue du test est largement compensé par une diminution importante des temps d'isolation des fautes.

Il est évident que le rapport coût-bénéfice n'atteindra son maximum que si tous les circuits sur la carte intègrent le standard IEEE 1149.1 et des outils de support au test "boundary scan" sont disponibles. Or, depuis quelques années, le marché a été envahi par une gamme de produits "boundary scan" compatibles□ des logiciels pour le test et le diagnostic basés sur des plateformes de type ordinateur personnel, des processeurs spécifiques capables de gérer l'application des séquences de vecteurs de test pour la carte, des microprocesseurs intégrant le standard, des "latches", "buffer/driver", "transceiver" "boundary scan", etc…

Bien que cette phase de transition du test "in-circuit" au test "boundary scan" soit d'ores et déjà une réussite, les cartes mixtes BS/non-BS seront sûrement le cas le plus courant dans les années à venir. Par conséquent, il faudra trouver des procédures de test qui assurent une bonne couverture des fautes dans les parties non-BS.

Dans le cas de certains produits conçus entièrement (circuit, carte, module, système) chez le même constructeur, cette phase de transition a peut-être déjà abouti. La question que l'on peut se poser alors, est s'il est possible de faire mieux. L'intégration du BIST tant au niveau des circuits comme à celui des connexions amènera sans doute une réponse OUI à cette question.

De plus, il y a le cas des produits qui seront utilisés dans des applications sécuritaires, où les erreurs doivent être détectées en cours de fonctionnement. C'est-à-dire que le besoin d'un test en-ligne vient s'ajouter aux tests de fin de fabrication et de maintenance (tests hors-ligne) de ces produits. Etant donné que l'utilisation du mode d'échantillonnage pour le test en-ligne mènerait à une dégradation très importante des performances de la carte, l'extension du standard IEEE 1149.1 est à envisager pour assurer aussi la détection concurrente d'erreur par le matériel.

CHAPITRE III

DES TECHNIQUES HÉTÉROGÈNES AU BIST DE CARTES "BOUNDARY SCAN"

III.1 Introduction

Le standard IEEE 1149.1 pour le test "boundary scan" (BS) de cartes [IEE90] représente sans aucun doute une réponse adaptée aux besoins des technologies actuelles de mise en boîtier et de connectique.

Bien que la plupart des recherches dans ce domaine assument une disponibilité totale de ressources BS, l'industrie doit affronter à présent un marché qui commence seulement à intégrer le standard dans ses produits. Ceci implique que la grande majorité des cartes et systèmes contiennent, et contiendront dans un avenir proche, des parties BS ainsi que d'autres n'intégrant pas le standard IEEE 1149.1 (ces parties non-BS sont connues dans la littérature sous le nom de "clusters"). Ainsi, les problèmes de contrôlabilité et d'observabilité des noeuds internes aux "clusters", d'observabilité des cellules BS de sortie qui alimentent les entrées des "clusters" et de contrôlabilité de cellules BS d'entrée qui sont alimentées par les sorties des "clusters", sont encore à considérer dans les cartes partiellement "boundary scan".

Malgré cette situation de transition, on retrouve d'ores et déjà sur le marché, des processeurs de test BS qui, étant capables de gérer entièrement l'application des séquences de vecteurs et l'analyse des réponses de la carte (même celles partiellement BS), peuvent accomplir une partie de la tâche du testeur externe. Ce genre de circuit intégré représente une avance importante vers l'autotest à 100% de systèmes électroniques.

Même si tous les circuits montés sur la carte intègrent le standard BS et du BIST, l'utilisation d'un tel processeur pour l'autotest de cartes complexes impose l'utilisation de beaucoup de mémoire pour stocker le programme de test, la séquence à appliquer aux connexions, la réponse attendue à cette séquence, l'ensemble de signatures attendues pour chacun des circuits intégrés et la procédure de diagnostic. A ceci vient s'ajouter sans doute un nombre excessif d'opérations de décalage dû au fait que, excepté pour l'autotest de circuits, toute autre fonction de test et diagnostic concernant l'ensemble de la carte est centralisée dans ce processeur.

Dans ce contexte évolutif et étant donnés les problèmes de chacune des étapes de ce processus, ce chapitre vise à proposer des méthodes pour le test hors-ligne et le diagnostic de cartes partiellement "boundary scan", dans un premier temps, et entièrement "boundary scan" / autotestable dans un deuxième temps. Pour cela, nous allons commencer par une proposition d'unification de la détection de fautes et du diagnostic dans les cartes BS partiel. La structure et l'utilisation des processeurs existants pour le test "boundary scan" seront ensuite brièvement discutées. Par la suite, une approche pour le BIST de connexions sera présentée. Cette approche est basée sur les séquences de type '01'-baladeur étudiées au chapitre II. Enfin, et afin

de compléter le BIST de cartes et de simplifier le plus possible l'intégration du diagnostic dans un processeur de test "boundary scan", nous proposons une approche autotestable de vérification de signature des circuits.

III.2 L'unification du test et du diagnostic de cartes BS partiel

La conception en vue du test "boundary scan" de cartes est marquée par deux étapes intermédiaires principales: la première dominée par l'utilisation des "clusters" non-BS complexes (figure III.1), avec des noeuds internes qui sont soit non observables, soit observables par l'intermédiaire de broches de test ou à travers le connecteur de la carte [RoD90]; la deuxième étape dominée par des "clusters" non-BS beaucoup plus simples, auxquels l'accès est possible à travers les ressources BS des circuits voisins [HAR89].

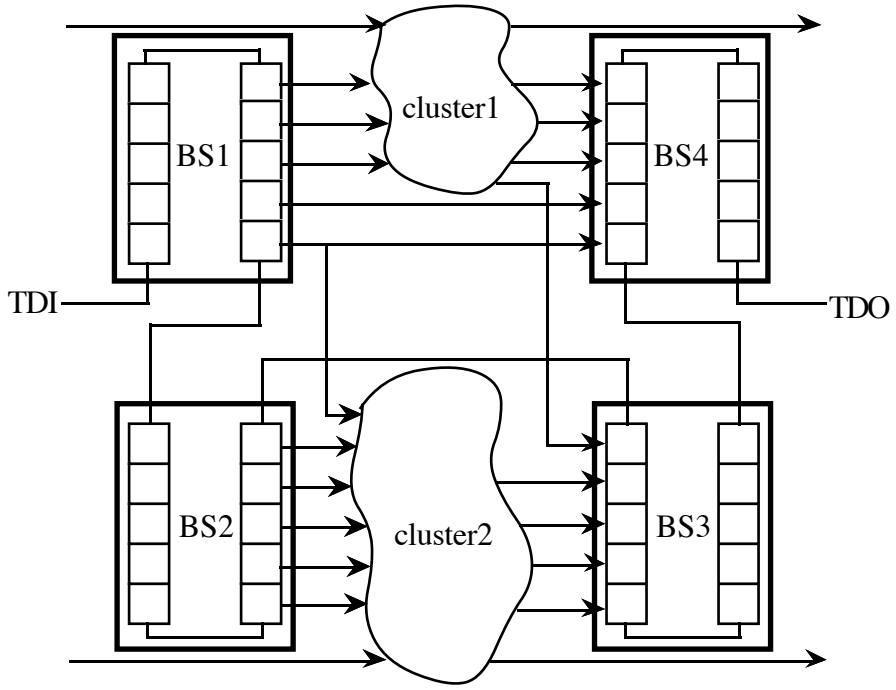


Figure III.1. Carte "boundary scan" partiel.

Quel que soit le cas, une procédure de test pour une carte partiellement "boundary scan" doit être capable d'identifier:

1) des circuits défaillants à l'intérieur des "clusters";

2) des courts-circuits et circuits-ouverts dans les "clusters";

3) des courts-circuits entre "clusters" différents;

4) des courts-circuits entre des "clusters" et des connexions entièrement BS;

5) des courts-circuits et circuits-ouverts dans les connexions entièrement BS;

6) des circuits BS défaillants.

Lors du test d'un "cluster", on doit considérer qu'une partie ou la totalité des circuits qui le composent n'ont probablement pas la faculté d'inhiber leurs sorties. Par conséquent, le contrôle des noeuds internes à travers un testeur "in-circuit" peut endommager les composants, et est donc abandonné. D'autre part, si la contrôlabilité est limitée aux cellules BS, le test des connexions (notamment des noeuds internes et des sorties des "clusters") ne peut pas être accompli que par l'activation de la fonction réalisée par le "cluster". Ainsi, l'unification des procédures de test des circuits et des connexions paraît convenable. On remarquera que le test unifié de circuits et connexions va servir à plusieurs propos□ la fabrication des cartes (où les circuits sont censés ne pas avoir de fautes) ou à la validation de prototypes et à la maintenance (où aucune hypothèse n'est faite, ni sur l'état des circuits, ni sur l'état des connexions).

Puisqu'une carte est composée normalement de plusieurs "clusters" partageant l'accès électronique des mêmes registres BS, le temps de test est réduit dans le cas de l'application en parallèle des tests de tous les "clusters". Une configuration optimale pour le chemin de balayage de la carte pourrait à priori aider à accélérer l'application du test [ChJ92]. Pourtant, afin d'assurer la détection de courts-circuits d'interaction (points 3 et 4 dans la liste de fautes donnée ci-dessus), les sorties de tout "cluster" et de toute connexion entièrement BS doivent être observées de façon continue durant le test [RoD90]. De cette manière, pour éviter des accès inefficaces aux registres BS des circuits [Han89], le test des connexions BS doit être réalisé le plus souvent possible en même temps que la vérification des "clusters". La partie restante du test des connexions BS est donc accomplie dans la phase suivante, et les circuits BS sont vérifiés en phase finale, car les mêmes moyens d'accès sont nécessaires au test des "clusters" et des connexions BS.

Dans ce contexte, nous proposons une méthodologie pour le test unifié de "clusters" et connexions. Ce test s'effectue selon une procédure à deux étapes□

La première étape est responsable de la détection de fautes et d'un premier diagnostic de fautes. A partir d'une description hiérarchisée de la carte, les parties entièrement BS et les "clusters" sont d'abord identifiés et une partition de la carte en résulte. Des séquences de test pour tous les "clusters" sont ensuite générées et ordonnées, et des vecteurs pour le test des connexions entièrement BS sont associés à l'ensemble des tests des "clusters". Ensuite, l'application de ces séquences de test fournit à la procédure de diagnostic les valeurs observées aux sorties primaires des "clusters" et des connexions BS. Ainsi, un premier diagnostic est

obtenu en utilisant un système à base de connaissance. Ce logiciel produit un ensemble de candidats à la faute et une liste de noeuds des "clusters" à observer dans l'étape suivante.

Le deuxième pas est responsable d'un diagnostic approfondi de fautes. En effet, l'application en parallèle de séquences de test indépendantes à chaque candidat va confirmer la validité ou l'invalidité des hypothèses résultantes de la première phase de test. Par exemple, si un court-circuit entre un noeud d'un "cluster" et une connexion BS inconnue est l'un des candidats à la faute, des vecteurs de test seront appliqués afin d'identifier la connexion BS affectée. De cette façon, ce qui reste de la première étape pour le test des connexions BS peut être réalisé à ce moment-là, en même temps qu'on observe les sorties du "cluster" défaillant. Pour le diagnostic des connexions internes aux "clusters", l'approche de génération automatique de vecteurs de test proposée dans [MeB92], par exemple, peut être utilisée pour sélectioner les candidats.

III.2.1 La détection de fautes

Etant donné que la détection de courts-circuits doit aussi être assurée, le test simultané de tous les "clusters" d'une carte BS partiel fait que la complexité de la génération de vecteurs de test est importante. Afin de détecter des courts-circuits d'interaction on doit assurer, d'abord, que le même vecteur de test n'est pas appliqué à deux noeuds appartenant à des "clusters" différents ou des parties entièrement BS et, ensuite, que les erreurs sont propagées aux sorties primaires des "clusters" et/ou des connexions BS.

Le problème ne pouvant être traité d'une manière globale, nous procédons à son partitionnement en sous-tâches comme suit□

1) des séquences de test individuelles sont obtenues (à partir d'un outil de génération de vecteurs de test, d'un ingénieur de test, ou d'une bibliothèque de test) qui assurent une couverture élevée de fautes internes aux "clusters" (des collages, des circuits-ouverts et des courts-circuits);

2) afin de garantir une couverture maximale des courts-circuits d'interaction entre "clusters", on manipule l'ordre et la répétition de vecteurs de test disponibles. La contrainte que nous cherchons à satisfaire est d'assurer que la séquence finale du test de la carte ne dépasse pas la longueur de la plus longue des séquences de l'ensemble des "clusters";

3) à partir de la séquence de test de l'ensemble des "clusters" de la carte, la séquence de vecteurs à appliquer aux connexions BS est générée. L'idée principale ici est que ces vecteurs garantissent la propagation aux sorties primaires des "clusters" des courts-circuits d'interaction entre les noeuds des "clusters" et les connexions BS, et que le test de courts-circuits dans les

parties BS commence par la génération d'une première partition de l'ensemble total de leur connexions.

L'ordonnancement des séquences de test et la génération des vecteurs pour les connexions BS sont basés sur une matrice représentant la couverture de collages logiques pour la séquence à appliq[redacted]btenue automatiquement à travers la combinaison des résultats produits par la simulation logique et par la simulation de fautes des "clusters". Chaque élément de la matrice M(i,j) donn[redacted]la couverture des collages sur le noeud i par le vecteur j comme suit:

M(i,j)=D [redacted] le vecteur [redacted] détecte un collage-à-'0' dans le noeud i;

M(i,j)=[redacted]ur j détecte un collage-à-'1' dans le noeud i;

M(i,j)='0'/'1'[redacted]

Un exemple de matrice est do[redacted]

noeuds	j1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
i1	D	0	[redacted]				
i2	D	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
i3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	D	0
i4	1	D	1	[redacted]	[redacted]	0	D
i5	1	0	0	[redacted]	0	[redacted]	[redacted]
i6	1	[redacted]	[redacted]	[redacted]	D	[redacted]	[redacted]

Figure III.2. Exemple de matrice de couverture de collages.

Toute information nécessaire à l'ordonnancement de séquences de test des "clusters" et au calcul des vecteurs de test des connexions BS peut être obtenu à partir de cette matrice, étant donné que tous les chemins [redacted] primaires des "clusters" peuvent être identifiés. La détection d'un cou[redacted]circuit ent[redacted]eux noeuds i_1 et i_2 appartenant à des "clusters" différents, par exemple, imp[redacted]s de la matrice $[M(i_1), M(i_2)]$, il existe un vecteur j tel que:

$[M(i_1,j)=D, M(i_2,j)=$[redacted]$]$ po[redacted] tout type de court-circuit;

$[M(i_1,j)='0', M(i_2,j)=D]$ [redacted] $[M($[redacted]$,j)=$[redacted]$(i_2,j)='0']$ [redacted]ur un court-circuit de type ET;

$[M(i_1,j)='1', M(i_2,j)=$[redacted]$]$ pour un court-circuit de type OU.

En outre, afin de détecter n'importe quel court-circuit entre un nœud i interne à un "cluster" et une connexion BS, on doit assurer qu'il existe j_1, j_2, $j_1 \neq j_2$, tel que:

[M(i,j_1)=D, toute_connexion_BS='0'] [M(i,j_2)=D, toute_connexion_BS='1'].

Dans le cas de courts-circuits avec des connexions BS, le problème consiste donc à déterminer deux ensembles de vecteurs de test tels que:

1) l'application simultanée du premier (nommé ensemble D) et des '0's à toutes les connexions BS assure la propagation de courts-circuits d'interaction de type ET à travers le "cluster";

2) l'application simultanée du deuxième (nommé ensemble [illegible]) et des '1's à toutes les connexions BS assure la propagation de courts-circuits d'interaction de type OU à travers le "cluster".

Notre algorithme de calcul de ces deux ensembles est donné dans la figure III.3.

III.2.2 Le diagnostic de fautes

Suite à la détection d'une faute, un premier diagnostic consiste à générer une liste de candidats à la faute. Ce processus utilise un système à base de connaissance, nommé PESTICIDE [Mar91], qui a été développé pour le diagnostic de circuits VLSI complexes. La réutilisation de ce système a été possible grâce à sa versatilité notamment du point de vue de la modélisation des dispositifs et des fautes.

Trois types de composants sont utilisés pour modéliser le circuit: les blocs, les connexions et les interfaces de blocs. Les bases de connaissance sont bâties à partir de ces éléments. La modélisation est fortement hiérarchisée: les différents niveaux d'abstraction et de complexité peuvent être considérés, de la description fonctionnelle d'une carte jusqu'à la description structurelle d'un simple composant.

L'observation et le contrôle sont faits à travers les interfaces des blocs et non sur les connexions. Ceci amène deux conclusions importantes:

1) cette caractéristique est spécialement adaptée au concept de cellule "boundary scan", car ces cellules représentent les principaux sites d'observabilité et de contrôlabilité;

2) ce genre de modélisation fait qu'il est facile d'unifier le diagnostic de composants et de connexions en utilisant la même méthodologie et les mêmes résultats observés.

```
main ()
{créer_ensemble (X); créer_ensemble (Y);
 créer_matrice_candidats (D); créer_matrice_candidats (Ø);
 if (aucun D_candidat)
 then if (Ø_candidat) {générer_combinaisons (Ø,Ø);}
 else if (aucun Ø_candidat)
      then générer_combinaisons (D,Ø);
      else générer_combinaisons (D,Ø);
 sortir_ensembles ()
}
```

```
créer_ensemble (X)
{for (toute ligne de matrice_couverture)
 if (un seul X) {X_ensemble = X_ensemble  colonne correspondante;}
}
```

```
créer_matrice_candidats (X)
{éliminer les lignes de matrice_couverture où un X est associé à un élément de X_ensemble;
 repeat until (la réduction de matrice_candidats_X n'est plus possible)
 {éliminer les lignes et colonnes de matrice_candidats_X où il n'existe pas de X;
  éliminer les lignes de matrice_candidats_X contenant d'autres lignes avec la même
                                                                    couverture_X;
 }
}
```

```
générer_combinaisons (X,Y)
{indice = 1; intersec_min = max (X_candidat, Y_candidat);
 while ((indice ≤ max (X_candidat, Y_candidat)) and (intersec_min > 0))
 {if (X_candidat) {X_comb = combinaison_suivante (matrice_candidats_X, indice);}
  if (Y_candidat) {Y_comb = combinaison_suivante (matrice_candidats_Y, indice);}
  if ((X_candidat) and (aucun Y_candidat) and (X_comb))
  {intersec = cardinal (X_comb ∩ X_ensemble);
   if (intersec < intersec_min) {intersec_min = intersec;}
  }
  if ((Y_candidat) and (aucun X_candidat) and (Y_comb))
  {intersec = cardinal (Y_comb ∩ Y_ensemble);
   if (intersec < intersec_min) {intersec_min = intersec;}
  }
  if ((X_candidat) and (Y_candidat)
  for (toute X_comb)
   for (toute Y_comb)
   {intersec = cardinal ((X_comb ∩ X_ensemble) ∪ (Y_comb ∩ Y_ensemble)
                                                  ∪ (X_comb ∩ Y_comb));
    if (intersec < intersec_min) {intersec_min = intersec;}
   }
 }
 if (X_candidat) {X_ensemble = X_ensemble ∪ (X_comb ∩ intersection minimum);}
 if (Y_candidat) {Y_ensemble = Y_ensemble ∪ (Y_comb ∩ intersection minimum);}
}
```

Figure III.3. L'algorithme pour le calcul des ensembles D and Ø

Puisqu'aucun modèle de fautes n'est assumé, le processus du diagnostic est déclenché lorsqu'une valeur incorrecte est exhibée sur l'une ou plusieurs interfaces de bloc.

L'outil de diagnostic vérifie la cohérence des bases de connaissance, permet la détection de défauts dans les connexions et la localisation des blocs candidats à la faute [Mar91]. L'efficacité du diagnostic sur les connexions et circuits des "clusters" est extrêmement dépendante de l'observabilité disponible, ceci étant plutôt un problème de qualité de conception pour la testabilité de la carte qu'une déficience de l'outil.

Après l'application des séquences de test, les valeurs obtenues sur les noeuds observés sont collectées et les bases de connaissance sont construites. Les bases de connaissance structurelle et fonctionnelle décrivent les blocs et les connexions, les deux étant définis par leur noms et une liste d'interfaces de bloc associée. [redacted] bloc est décrite par des attributs structurels (nom, type, etc…). La base de [redacted] comportementale contient des données sur la session de test courante, et décrit les i[redacted]tances d[redacted] interfaces de bloc. On peut générer autant de bases de connaissance comporteme[redacted]tale que le [redacted]ombre de comportements erronés des dispositifs exhibés par le vecteur appliqué. [redacted]tant donné [redacted]ue le comportement d'une interface de bloc est exprimé par son attribut d'état (e[redacted]oné, co[redacted]ct ou inconnu) et non par sa valeur logique, la détection soit d'un D, soit d'un [redacted]rétée de la même manière par PESTICIDE.

Les résultats du diagnostic PESTICIDE permettent de classer (par rapport au vecteur de test appliqué) les blocs des clusters en trois catégories: 1) blocs corrects, 2) blocs défaillants et 3) blocs présumés défaillants. Tous les blocs sur le chemin conduisant à une sortie primaire erronée seront présumés défaillants, car ils peuvent être eux-mêmes défaillants, ou simplement propager l'erreur.

A la fin de cette première classification de candidats à la faute, il est probable que le niveau d'ambiguité reste encore très important. Ceci est dû au fait que le diagnostic PESTICIDE est principalement basé sur les erreurs observées et sur la connectivité structurelle dans le "cluster". Aucun modèle de fautes n'est assumé, et la fonction interne des blocs n'est pas prise en compte. PESTICIDE établit des chemins de causalité de la propagation d'erreur dans le "cluster", et dans notre cas, les noeuds à observer (les sorties primaires) se situent au bout de tous les chemins possibles. Cela signifie qu'une ambiguité acceptable pour le diagnostic de "clusters" complexes ne pourra être atteinte que si des sites additionnels d'observabilité sont disponibles par l'ajout des composants comme les "TI's BS SCOPE™ OCTALS" [TII89] par exemple. Pourtant, afin de réduire cette ambiguité, PESTICIDE applique une heuristique additionnelle basée sur l'information "known-as-good". On peut alors enlever de la liste de candidats les blocs qui alimentent les noeuds qui présentent des valeurs correctes, et qui n'alimentent pas les noeuds avec des valeurs erronées. L'heuristique sera évidemment plus efficace dans les cas où un nombre élevé de noeuds corrects est généré par une session de test donnée.

Le système à base de connaissance fournit un premier diagnostic à travers la génération de candidats à la faute. Si les "clusters" ne sont pas très complexes, c'est-à-dire que la carte BS partiel présente un bon niveau de testabilité, ce premier diagnostic peut être suffisant. Sinon, la granularité du diagnostic n'étant pas assez fine, un processus de diagnostic approfondi doit avoir lieu. Celui-ci résultera dans la sélection des candidats à la faute.

Afin de permettre l'application de séquences de test indépendantes à chaque candidat à la faute identifiée à la fin du premier pas de notre méthodologie, il est nécessaire d'assurer l'interaction automatique entre PESTICIDE, un générateur de vecteurs de test et un simulateur. A partir d'une telle interaction, la discrimination de candidats sera possible au point où les syndromes de connexions court-circuitées (chapitre II) sont déchiffrées, où un circuit et une connexion défaillants sont distingués l'un de l'autre, etc.

Une stratégie à considérer dans les travaux de diagnostic approfondi consiste à ajouter d'autres heuristiques au raisonnement du système à base de connaissance. Des informations sur la structure globale de la carte et sur les technologies utilisées, par exemple, peuvent se rendre utiles pour la sélection des candidats à la faute.

III.2.3 Implémentation et résultats

Les principaux éléments et l'organisation générale de l'outil résultant de l'implémentation de la méthodologie proposée sont donnés par la figure III.4.

Ce logiciel a été utilisé pour le test et le diagnostic de cartes [MLT93] et MCMs ("Multi-Chip Modules) [LMT94] partiellement "boundary scan". Des expérimentations ont été réalisées en utilisant les "benchmarks" combinatoires donnés dans [BrF85]. Les résultats obtenus sont détaillés dans [MLT93]. L'ordonnancement des tests des "clusters" et l'algorithme de calcul des vecteurs de test pour les connexions BS ont permis l'obtention de séquences de test finales qui ajoutent à la couverture des fautes de types 1 et 2 la détection des fautes de type 3 et 4 (conformément à la classification donnée au début du paragraphe III.2). En comparaison avec les tests initialement obtenus en utilisant un générateur de vecteurs de test conventionnel, nos séquences présentent un surcoût en nombre de vecteurs qui varie de 6 à 50% (dû à la répétition de vecteurs). Ces séquences ont permis le démarrage du test en vue de la détection des fautes de type 5. Les fautes de type 6 sont traitées dans un pas ultérieur à travers l'utilisation du chemin de décalage de la carte pour le test de la logique interne des circuits BS. Par rapport à la génération de candidats à la faute, les sessions de diagnostic ont permis des réductions du nombre de blocs des "clusters" qui varient entre 44 et 67%.

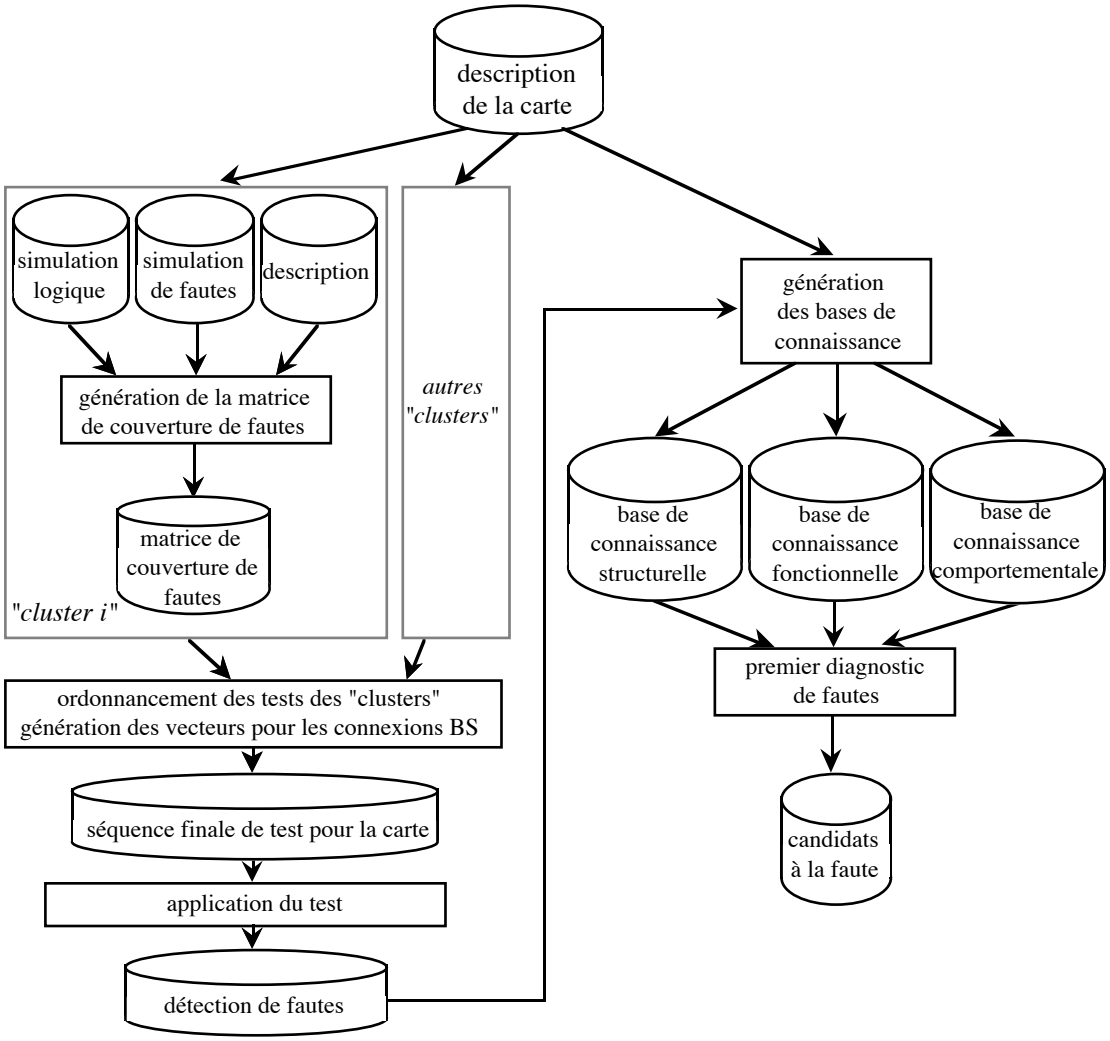


Figure III.4. Scénario de test et diagnostic.

III.3 Les processeurs de test "boundary scan"

Etant donné que la technologie "boundary scan" est à l'origine de l'environnement de test que l'on a vu s'installer de manière solide ces dernières années, cette méthode constitue un sérieux candidat au développement de fonctions d'autotest utiles aux tests de fabrication et de maintenance de cartes.

En effet, plusieurs processeurs de test BS ont été proposés [LiB89, Vin89, YaJ90, Der91, JaY91, FMP92]. Leur objectif est soit de réduire le prix des testeurs, soit de servir à l'autotest de cartes. Dans tous ces processeurs un nombre d'opérations de base ont été définies: l'initialisation de la logique BS, les transitions d'état dans la logique de contrôle de chaque composant, le décalage de chaînes de bits à travers le chemin de balayage BS, la comparaison entre séquences de bits, etc…

De manière générale, l'architecture d'un processeur capable d'exécuter les opérations de base mentionnées ci-dessus est composée d'un sous-ensemble des blocs montrés dans la figure III.5. Chacun de ces blocs à un rôle très précis dans une telle architecture:

- l'interface parallèle d'entrée permet la communication du processeur de test avec un processeur principal qui serait responsable de la gestion des tests appliqués par un ensemble donné de processeurs;

- d'une part, le port TAP système fait du processeur un circuit compatible avec le standard IEEE 1149.1, et d'autre part, il permet la communication série du processeur avec le processeur principal;

- le bloc BIST contient la circuiterie nécessaire à l'autotest du processeur de test lui-même;

- l'interprète de commandes est le coeur du processeur de test; il va accepter des commandes venant du processeur principal, il va chercher des commandes dans le programme de test et, en fonction de la commande à exécuter, il va contrôler sa propre partie opérative et l'opération de tous les autres blocs du processeur;

- le décaleur/"buffer" est l'endroit de stockage et l'interface de conversion parallèle-série/série-parallèle pour les données reçues, pour celles à traiter à l'intérieur du processeur et pour les données à transmettre via les bus parallèles et les chemins série disponibles;

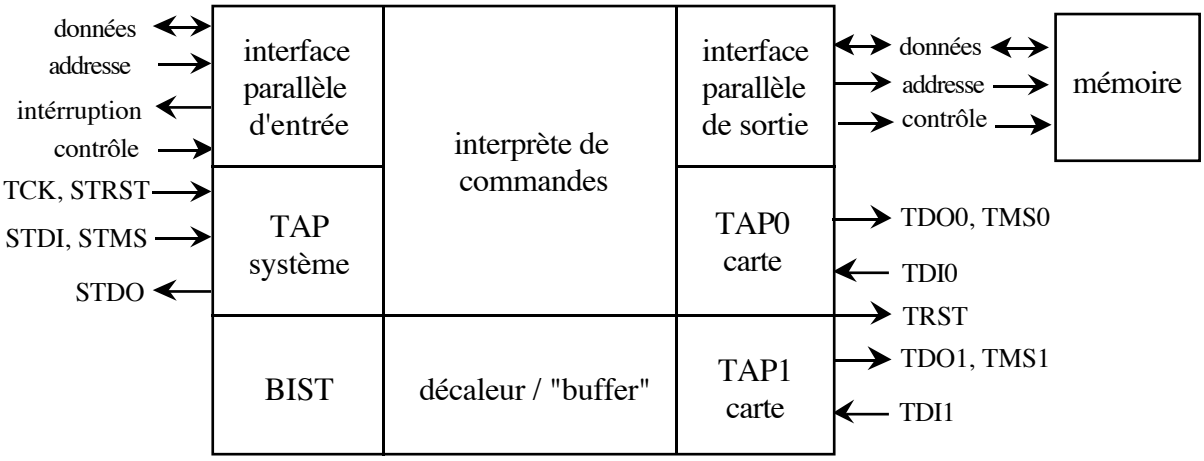


Figure III.5. Processeur de test BS général.

- l'interface parallèle de sortie permet l'accès du processeur à une mémoire externe qui sert au stockage du programme de test, des séquences à appliquer aux connexions et des

réponses attendues, des signatures d'autotest des circuits et, éventuellement, d'une procédure de diagnostic; évidemment, si cette mémoire est intégrée dans le processeur lui-même, l'interface parallèle de sortie disparaîtra;

- finalement, les ports TAP0 et TAP1 représentent la possibilité d'avoir plus d'un chemin de balayage "boundary scan" dans une même carte, ces chemins étant commandés ou non par des signaux TMS indépendants.

D'après la description de l'architecture générale d'un processeur de test BS, nous pouvons conclure que toutes les ressources nécessaires à l'implémentation de l'autotest hiérarchisé y sont intégrées, et que les ressources n'étant pas nécessaires dans un certain cas peuvent être supprimées.

Un tel processeur étant disponible sur la carte, il ne nous reste donc qu'à lui fournir un programme à exécuter accompagné des séquences de test adéquates à l'environnement où il est inséré. Une approche pour la génération automatique du programme de test basée sur un jeu d'instructions donné est présentée dans [FMP92].

III.4 Le BIST de connexions "boundary scan"

Dans ce paragraphe nous allons présenter l'approche BIST proposée dans [HAN92]. Cette approche pour le test des connexions de la carte comporte des avantages similaires à ceux du BIST de circuits intégrés [WiP82]: simplicité et efficacité temporelle dans la génération d'un ensemble de vecteurs de test, et simplicité de la procédure de détection et diagnostic.

Pourtant, comme pour n'importe quelle approche BIST, le surcoût en surface ne peut pas être évité et se manifeste dans ce cas comme des extensions des cellules BS décrites dans [IEE90]. On remarquera que dans plusieurs autres travaux [Bha91, GlB88, NWA91, etc] les cellules BS telles que données dans [IEE90] ont été adaptées à différents besoins.

Le schéma BIST présenté dans la suite utilise un processeur de test BS pour le contrôle du bus de test et la réalisation des fonctions BIST. Les vecteurs de test sont obtenus en utilisant une séquence de type '01'-baladeur étudiée dans le chapitre précédent. Nous allons assumer que le nombre total de cellules BS sur la carte, N, est toujours pair. Si N est impair le processeur de test BS pourra régler la longueur du chemin de balayage de la carte de telle sorte qu'il devienne pair.

III.4.1 Extensions des cellules "boundary scan"

La figure III.6 reprend le schéma d'une cellule BS présenté dans le chapitre précédent (figure II.2) et rajoute des fonctionnalités pour le BIST□

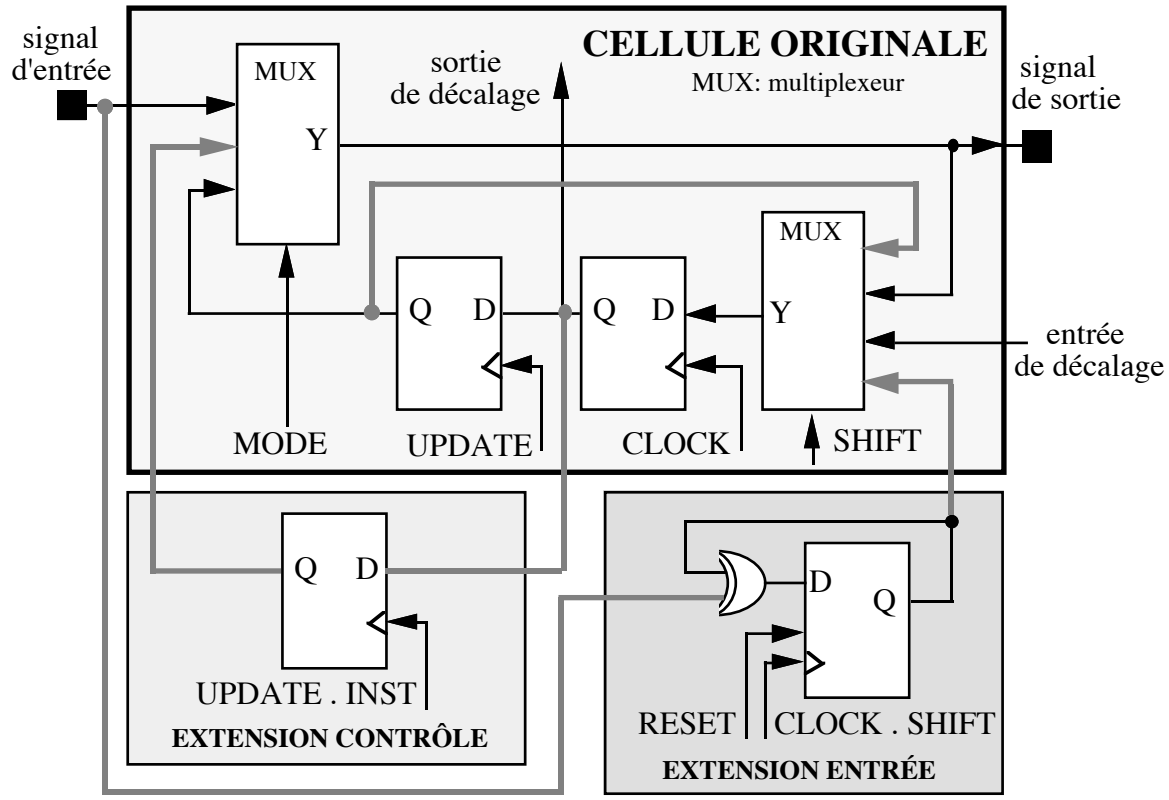


Figure III.6. Cellules BS modifiées.

En s'agissant d'une cellule d'entrée, une bascule et une porte XOR forment la circuiterie de contrôle de parité (définition I.13) pour le compactage des réponses de test. On remarque que la cellule finale peut accomplir le décalage d'un vecteur à travers le chemin BS en même temps et de manière indépendante du compactage des réponses de la connexion concernée.

La cellule BS de sortie est capable de recouvrer sa valeur précédente lors du test des connexions. Puisque les vecteurs de la séquence '01'-baladeur sont appliqués en utilisant un cycle "Shift-DR/Update-DR/Capture-DR" (figure II.5), cette capacité assure que la séquence décalée ne sera pas détruite à cause d'une opération de chargement dans l'état "Capture-DR".

La cellule BS de contrôle, quant à elle, a une capacité de type "bypass". C'est-à-dire qu'une fois que la bascule ajoutée est chargée par l'instruction de test qui convient, les opérations de décalage peuvent se passer de manière transparente au contrôle.

Finalement, dans le cas d'une cellule bidirectionnelle, les trois types de cellules modifiées et décrites ci-dessus sont tout simplement mis ensemble.

III.4.2 La génération de test et le compactage des réponses

Considérons que le vecteur binaire '100…0' (séquence de N bits avec un seul '1' suivi de (N-1) '0's), est appliqué au chemin BS d'une carte où la cellule la plus proche de la broche TDO du processeur de test (broche TDI de la carte) est une cellule BS de sortie. Alors dans chaque cellule BS d'entrée, exceptées celles appartenant à la connexion sur laquelle le '1' est appliqué, le bit perçu par la circuiterie de compactage est un '0'. Si, ensuite, ce vecteur est décalé d'une position, le vecteur '010…0' sera obtenu et un '1' ne sera appliqué qu'aux cellules BS d'entrée connectées à la deuxième cellule BS de sortie du chemin BS. Ainsi, par l'intermédiaire du décalage de ce vecteur à travers le chemin BS, une séquence de type '1'-baladeur est appliquée et toutes les connexions reçoivent, l'une après l'autre, une valeur logique '1'.

De manière semblable, par l'intermédiaire du décalage d'un vecteur '011…1' à travers le chemin BS, une séquence de type '0'-baladeur est appliquée et toutes les connexions reçoivent, l'une après l'autre, une valeur logique '0'. Comme il a été montré dans le chapitre précédent, un test constituée de ces deux séquences de type '01'-baladeur associées à deux autres vecteurs de type tout-à-'0' et tout-à-'1' assure le diagnostic maximum qu'on peut obtenir pour les connexions d'une carte BS. En plus, la génération d'une telle séquence par le matériel est triviale et fait que le nombre de ressources internes nécessaires au processeur de test BS est considérablement réduit.

Comme il a été mentionné précédemment, les réponses de test sont compactées localement dans chaque cellule BS d'entrée. La bascule ajoutée à une cellule BS d'entrée est initialisée à '0'. Le premier bit de sortie venant de la connexion en observation et le contenu présent de la bascule sont les entrées de la porte XOR de la cellule modifiée. Sa sortie est enregistrée dans la bascule et renvoyée à l'une des entrées de la porte XOR. Cette procédure est répétée N fois pour une séquence de N vecteurs de test et la réponse finale est compactée dans la propre bascule.

Pour une séquence de type '1'-baladeur de N bits (N est pair par définition), un seul '1' est appliqué à chaque connexion. Pour une séquence de type '0'-baladeur de N bits, (N-1) '1's sont appliqués à chaque connexion. Dans les deux cas la porte XOR reçoit un nombre impair de '1's, ce qui fait que la réponse attendue pour chaque séquence '01'-baladeur est '1'.

Les réponses compactées sont décalées à travers le chemin BS et compactées encore une fois dans le processeur de test de la carte. La circuiterie de compactage pour la carte est un compteur du nombre de '1's, qui est utilisé indépendamment pour les comptages suivant l'application de la séquence '1'-baladeur et puis de la séquence '0'-baladeur.

Quant au temps d'application du test, on remarque que pour chacune des séquences '01'-baladeur, le premier vecteur étant chargé dans le chemin de balayage de la carte, il ne reste qu'à réaliser N fois l'opération de décalage tout en injectant simultanément un '0' ou un '1' et jamais les deux. Ensuite, après le compactage des réponses, N bits sont décalés vers le processeur de test pour la détection et le diagnostic. De cette façon, la complexité de ce schéma au niveau du temps est O(N), ce qui fait qu'il est plus performant que toute autre séquence de test existante dans le cas d'une programmation du mode de test externe prévu dans le standard IEEE 1149.1 [HAN92].

III.4.3 La détection et le diagnostic de fautes

Dans [HAN92] les auteurs prouvent que, pour tout court-circuit et tout collage n'affectant pas simultanément la même connexion, le nombre de '1' dans les bascules de compactage ajoutées aux cellules BS d'entrée est toujours erroné.

Pour le diagnostic, d'autre part, le compactage des réponses dans le compteur de '1's n'est pas nécessaire. A partir du compactage des réponses dans les cellules BS d'entrée, une paire de bits est obtenue, chaque bit correspondant à l'une des séquences '01'-baladeur. La figure III.7 montre que la paire de bits compactés pour le cas sans faute est '11'. Pour toute faute de type court-circuit ou collage la paire de bits produit '01', '10' ou '00'. Ainsi, chaque connexion peut être déclarée défaillante ou sans faute en fonction des réponses compactées et décalées dans le processeur de test de la carte.

type de faute	'1'-baladeur	'0'-baladeur
sans faute	'1'	'1'
collage-à-'1'	'0'	'0'
collage-à-'0'	'0'	'0'
court-circuit OU (impair)	'1'	'0'
court-circuit OU (pair)	'0'	'0'
court-circuit ET (impair)	'0'	'1'
court-circuit ET (pair)	'0'	'0'

Figure III.7. Le compactage des réponses aux séquences '01'-baladeur.

III.4.4 L'indépendance de la structure et le rôle du processeur de test

Dans l'approche en discussion, les signaux de contrôle des sorties qui alimentent une même connexion doivent être dûment appliqués avant que les séquences de type '01'-baladeur soient décalées à travers le chemin de balayage de la carte. Ceci requiert que le processeur de test BS ait des informations structurelles sur la connectique de la carte et que toute cellule BS de contrôle soit dessinée selon le schéma modifié donné au paragraphe III.4.1. Une fois ces signaux de contrôle appliqués, la génération des vecteurs de test et le compactage des réponses pour la détection et le diagnostic deviennent indépendant de la complexité de la structure des connexions, de l'ordre des connexions et de cellules BS, etc.

Comme mentionné précédemment, le processeur de test BS est censé contrôler le bus de test et aussi fournir les commandes BIST. Ce processeur doit être capable de générer les deux séquences de type '01'-baladeur, de générer les bits de contrôle basé sur les informations structurelles fournies par un testeur externe ou stockées dans le processeur lui-même, de compacter les données pour la détection de fautes, et d'analyser les réponses de test en vue de la localisation de la faute. Un tel processeur de test fait disparaître le besoin d'un équipement de test automatique sophistiqué et par conséquent cher.

III.4.5 Le test de circuits-ouverts

Les circuits-ouverts peuvent être détectés à travers la vérification de chaque chemin entre une cellule BS de sortie et toutes les cellules BS d'entrée de chaque connexion sur la carte. Toute cellule d'entrée doit donc être initialisée à une valeur connue et la valeur opposée doit être appliquée à la cellule BS de sortie sous vérification. Dans le cas sans faute les valeurs des cellules d'entrée doivent être modifiées à travers la connexion elle-même.

En effet les séquences de type '01'-baladeur peuvent être utilisées pour la détection et le diagnostic de circuits-ouverts [HAN92]. Pourtant, ce genre de test dépendra de la structure de la carte si des connexions à multiples cellules BS de sortie y sont présentes. Ceci étant le cas et P étant le nombre maximum de cellules de sortie branchées sur la même connexion, une séquence de type '01'-baladeur sera appliquée P fois avec P configurations différentes pour les cellules BS de contrôle, afin d'assurer la détection et le diagnostic de tout circuit-ouvert possible dans les connexions sous test.

III.5 La vérification de signatures sur les cartes "boundary scan"

La longueur de la signature résultante de l'application d'une procédure de BIST est normalement proportionnelle à la complexité du circuit où elle est intégrée [TRB91]. Si une très faible probabilité d'"aliasing" s'avère nécessaire [WDG86], le nombre de bits du registre de signature sera encore plus important. En plus, il est clair que la longueur et la bonne valeur de signature vont varier d'un circuit à l'autre sur une carte.

Dans un environnement complètement BIST-BS, ces différences font que le diagnostic devient plus difficile, car une chaîne non-déterministe de dizaines, voir de centaines de longues signatures est censée être recueillie dans le testeur ou le processeur de test monté sur la carte [NWA91]. L'impact sur le temps de vérification des signatures peut être réduit par l'utilisation de processeurs distribués à travers le système. Dans ce cas, la mémoire nécessaire au stockage des signatures sera simplement repartie parmi ces processeurs.

Dans ce contexte, ce paragraphe est consacré à la proposition et l'analyse de schémas de vérification de signatures visant la localisation de façon efficace de circuits défaillants montés sur la carte. Le but est de réduire le temps de décalage des signatures des circuits, ainsi que le nombre de ressources mémoire pour le stockage des signatures correctes, et surtout, de rendre le diagnostic maximum suffisamment simple pour être facilement intégré dans un processeur de test BS. Ces objectifs doivent être atteints sans toutefois demander de gros efforts de conception.

Dans la suite nous allons assumer que tout circuit sur la carte BS intègre du BIST selon le schéma d'association étudié au paragraphe II.6.3, qu'aucune valeur initiale n'est décalée de l'extérieur dans les registres de signature (conformément aux recommandations du standard IEEE 1149.1 [IEE90]) et que toute faute affectant soit le circuit sous test, soit sa circuiterie BIST, résulte en une signature incorrecte.

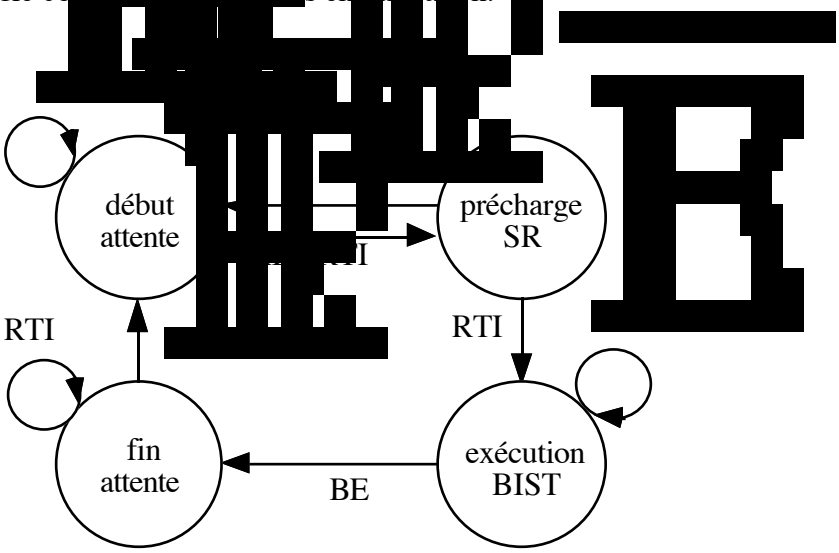
III.5.1 Vérification externe

La nature du protocole BS d'accès aux unités sous test augmente le temps de réparation, dans la mesure où le diagnostic ne peut pas se faire en un seul pas à cause du masquage de fautes. Ceci ne peut pas être évité, mais le temps utilisé peut être réduit par l'intermédiaire d'un ordonnancement adéquat du test. Par exemple, comme nous avons pu le voir dans le chapitre II, la raison pour laquelle on charge le registre d'instruction (IR) du standard IEEE 1149.1 avec 'X…X01' dans l'état "Capture-IR" du contrôleur TAP, est que les deux bits obligatoires ('01') se rendent utiles lors de la localisation des fautes dans le chemin série, avant que ce dernier soit utilisé pour d'autres tests.

Puisque le temps total d'exécution de l'autotest de la carte est sans doute beaucoup plus long que le décalage de la chaîne de signatures des circuits vers le processeur de test, il n'est pas raisonnable de courir le risque d'être empêché d'accéder à une partie des résultats des autotests dû, par exemple, à un collage logique affectant un bit d'un registre de signature. Ainsi, une approche similaire à celle utilisée pour le registre d'instruction doit vérifier le chemin série de la carte passant à travers les registres de signature.

La procédure de test de la carte proposée dans [TuY89] et étudiée dans le chapitre précédent prévoit la vérification des registres de données (DR) avant leur utilisation. Du point de vue du diagnostic, cette vérification ne peut être effective que si une valeur connue est chargée dans les registres avant que l'état "Capture-DR" du contrôleur TAP ne soit atteint. Pourtant, dans le cas du registre "boundary scan", la valeur chargée va dépendre soit de l'état des connexions de la carte, soit de l'état interne du circuit, et il n'est pas toujours possible de prédire ce que sont ces états.

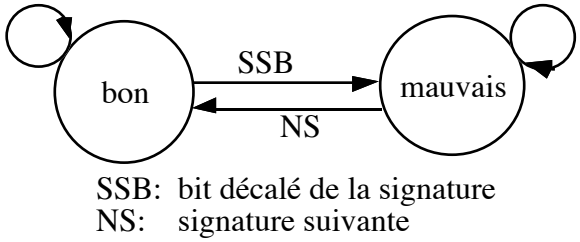
Compte tenu du fait qu'une partie du registre BS est utilisée comme analyseur de signature pendant l'exécution de l'instruction RUNBIST (paragraphe II.6.3), une alternative permettant de surmo[redacted]outer un état de précharge dans la procédure d'autotest. Dans cet état, une valeur fixe avec des bits obligatoires est chargée dans le registre de signature (figure III.8 [redacted]on, le premier passage par l'état "Run-Test/Idle" du contrôleur TAP permet [redacted] le [redacted]une chaîne de bits capable de vérifier le chemin série co[redacted]s en[redacted]n.



RB: instruction de test RUNBIST
RTI: état "Run-Test/Idle" du contôleur TAP
SR: registre de signature
BE: fin BIST

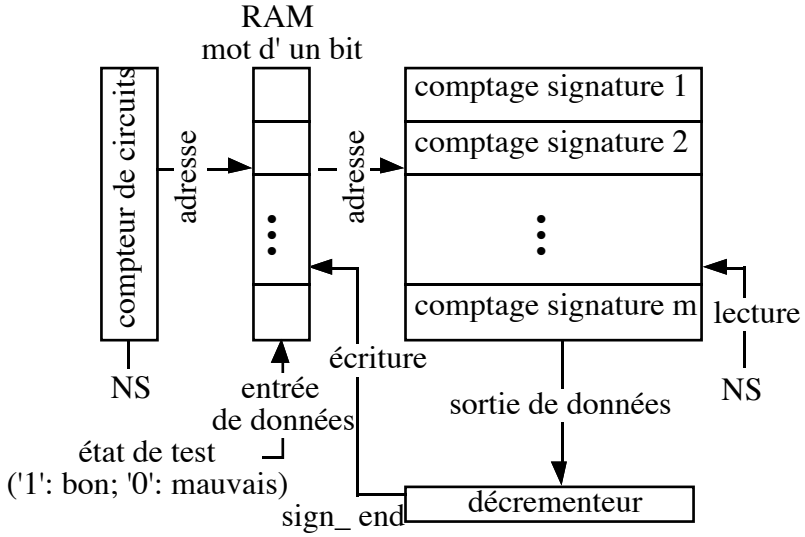
Figure III.8. Vérification des analyseurs de signature.

Dans la mesure où les registres de données de test sont testés avant l'exécution du BIST, les signatures correctes des circuits peuvent être des chaînes quelconques de bits. Pourtant, le choix approprié des signatures peut réduire la mémoire nécessaire au processeur de test de la carte et faire en sorte que l'implémentation en matériel d'une procédure de diagnostic devienne plus simple. Etant donné que les longueurs des signatures sont partout différentes dans la carte, la seule façon d'obtenir une chaîne déterministe de signatures est de choisir comme résultats sans faute de l'autotest soit des signatures tout-à-'0', soit des signatures tout-à-'1'. [McS86] présente une procédure simple [illegible] valeur initiale du registre d'analyse de signature de manière à ce qu'une signature donnée soit obtenue pour le circuit sous test. Cette valeur doit être chargée dans le registre de signature par le contrôleur du BIST lui-même. L'impact d'une telle approche sur la mémoire de stockage nécessaire au processeur de test est évident, car un mot de n bits est remplacé par un seul bit représentant le type de signature attendue ('0' tout-à-'0'; '1' tout-à-'1'), plus [illegible] représentant le comptage [illegible] '0's ou [illegible]. En effet, le premier bit peut être éliminé [illegible] seul type de signature est autorisé (figure III.9). Evidemment la standardisation des signatures [illegible] ne change pas le temps de décalage des résultats vers le processeur de test.



SSB: bit décalé de la signature
NS: signature suivante

(a) vérification d'une seule signature



(b) vérification de la chaîne de signatures

Figure III.9. Implémentation matérielle du diagnostic d'une chaîne de signatures tout-à-'0'.

Seule la standardisation de la longueur des signatures permet des améliorations en termes de mémoire de stockage et de simplicité de diagnostic. Etant donné que les longueurs des signatures sont fonction de la complexité des circuits, cet objectif ne peut être atteint par une approche de vérification externe, seule une approche intégrée peut le faire.

III.5.2 Vérification intégrée

Le but visé par l'implémentation de la vérification intégrée de signature est d'incorporer dans chaque circuit une fonction de test du niveau carte. La vérification intégrée donnant des réponses compactes de test au chemin de balayage BS, le coût de test de la carte est réduit au prix d'un surcoût très faible en surface de silicium.

La vérification intégrée de signature peut être réalisée par l'intermédiaire des circuits de type BISC ("Built-In Signature Checking circuit"). Pour que ce genre de circuit accomplisse de manière effective la tâche que lui est confiée, il doit posséder les propriétés suivantes :

Propriété III.1 : Seule une signature correcte peut produire un résultat correct de vérification pour un BISC sans faute.

Propriété III.2 : Pour un BISC défaillant, soit la faute est détectée indépendamment de la signature obtenue, soit le résultat correct de vérification sera produit si et seulement si la signature est correcte.

Dans la suite, nous allons présenter un BISC qui assure ces propriétés. Nous allons supposer que ce circuit ne peut être affecté que par un collage logique simple à la fois.

III.5.2.1 Le vérificateur "self-testing"

Notre approche pour la vérification intégrée de signature est donnée dans la figure III.10 [LAN93]. Ce BISC est basé sur l'évaluation série de la signature et sur la mémorisation de l'occurrence du premier '1' logique qui est décalé. Le détecteur de '1' est composé de la bascule C3 et d'une porte OU. Les bascules C3, C2 et C1 sont chargées de fournir au chemin série de la carte BS le résultat de la vérification de la signature.

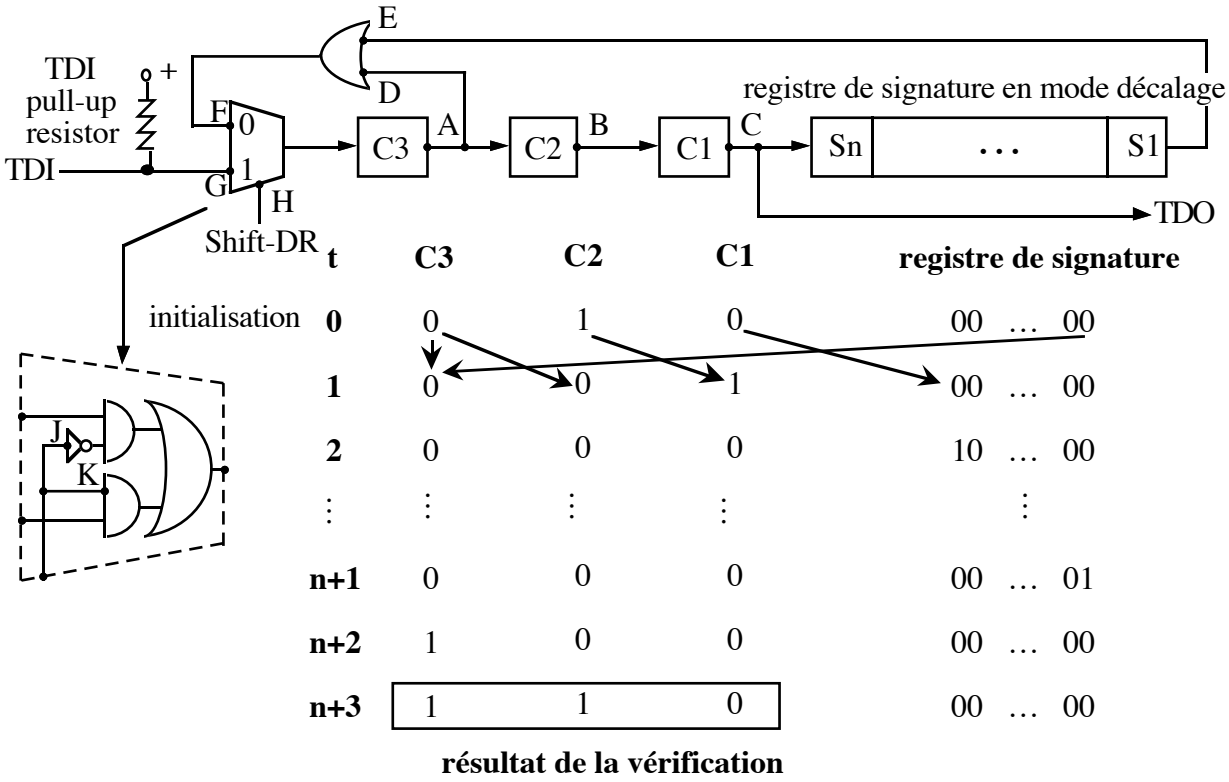


Figure III.10. Le vérificateur intégré "self-testing".

Les lemmes et le théorème qui suivent visent à démontrer que, pour le schéma proposé, les propriétés III.1 et III.2 peuvent être assurées sous certaines conditions.

Lemme III.1. En supposant que le BISC ne présente pas de faute et que C3/C2/C1 sont initialisés à '010', seule une signature de n bits de type tout-à-'0' peut produire le résultat de vérification C3/C2/C1='110' après n+3 cycles d'horloge.

Preuve. Le comportement du BISC de la figure III.10 peut être décrit par les équations logiques suivantes.

C3(n+3)=C3(0)+S1(0)+…+Sn(0)+C1(0)+C2(0)[+C3(0)]

C2(n+3)=C3(0)+S1(0)+…+Sn(0)+C1(0)+C2(0)

C1(n+3)=C3(0)+S1(0)+…+Sn(0)+C1(0)

Après simplification de ces équations, on obtient.

C3(n+3)=C2(n+3)=C1(n+3)+C2(0) [I]

C1(n+3)=C3(0)+S1(0)+…+Sn(0)+C1(0) [II]

Etant donné que C3(0)='0', C2(0)='1', C1(0)='0' et C3(n+3)='1', C2(n+3)='1', C1(n+3)='0', alors [I] est vraie. De même, à partir de [II] on obtient.

S1(0)+…+Sn(0)='0' [III]

Puisque la seule solution pour [III] est S1(0)=…=Sn(0)='0', seule une signature de type tout-à-'0' peut produire C3(n+3)='1', C2(n+3)='1', C1(n+3)='0', à partir de C3(0)='0', C2(0)='1', C1(0)='0'. □

Lemme III.2 Sous l'hypothèse de collage simple, le BISC assure la détection de toute faute interne indépendamment de la signature obtenue pour le circuit sous test.

Preuve Décomposons le circuit cible en trois blocs principaux: le registre à décalage (résultat de la vérification + signature), la porte OU et le multiplexeur.

Il est connu qu'une séquence de type '1'-baladeur est capable de détecter 100% des collages simples sur une chaîne de décalage [HRA88]. Cette séquence est appliquée au registre à décalage pendant la vérification de la signature de t(0) à t(n+2) (figure III.10). Comme tout collage affectant le registre à décalage et toute signature erronée changent le nombre de '1's qui circulent, la séquence finale appliquée sera modifiée. Dans ce cas, seuls '000' et '111' peuvent être obtenus comme résultat de la vérification (voir A, B, C et Si dans la figure III.11) grâce au circuit détecteur de '1'.

localisation	faute	signature décalée		remarques
		circuits précédents	circuit courant	
A, B, C	c_Z	… ZZZ ZZZ	ZZZ	Z est soit '0', soit '1'.
D	c_0 c_1	… XXX XXX … XXX XXX	010 111	X est "don't care".
E, F	c_0 c_1	… XXX XXX … XXX XXX	000 111	
G	c_0 c_1	… 000 000 … 111 111	XXX XXX	le diagnostic peut être faux. le diagnostic peut être faux.
H	c_0	… ZZZ ZZZ	ZXX	le diagnostic peut être faux.
H, K	c_1	… XXX XXX	111	
J	c_0	… 111 111 … 111 111 … 111 110 … 111 100 …	1XX 0XX 0XX 0XX	le diagnostic peut être faux.
Si	c_Z	… XXX XXX	ZZZ	la détection d'une combinaison de cette faute avec l'une des fautes données ci-dessus est assurée.

Figure III.11. L'analyse de fautes pour le BISC "self-testing".

Afin de tester la porte OU, l'application des vecteurs '00', '10' et '01' s'avère nécessaire. Ces vecteurs sont appliqués de t(0) à t(n) ('00'), à t(n+1) ('10') et à t(n+2) ('01'), et les réponses du circuit sont chargées dans C1, C2 et C3 respectivement à t(n+3). Bien qu'une signature erronée ou un collage dans le registre à décalage puisse empêcher l'application des vecteurs à la porte OU, aucun arrangement de vecteurs différent de celui que l'on vient de présenter ne peut produire un résultat de vérification '110' (voir D et E, F dans la figure III.11).

Pour la réalisation du multiplexeur donnée dans la figure III.10, seuls un collage-à-'0' sur J et un collage-à-'1' sur K ne peuvent pas être directement transposés en un collage affectant les entrées ou la sortie du multiplexeur. De la même façon, tout collage sur F ou sur la sortie du multiplexeur, et un collage-à-'1' sur H ou K (grâce à la résistance "pull-up" de la broche TDI du standard BS), sera signalé pendant la phase de vérification de la signature (figure III.11).

D'autre part, tout collage sur G, et un collage-à-'0' sur H ou J, sera détecté lors du décalage par la transposition des résultats de vérification des circuits suivants en une indication de mauvais fonctionnement (figure III.11).

Finalement, une initialisation incorrecte de C3/C2/C1 va donner nécessairement un résultat de vérification différent de '110', car l'ordonnancement de vecteurs de test ne correspondra pas à celui attendu.

A partir de la discussion ci-dessus et de l'analyse de fautes présentée dans la figure III.11, on conclut qu'une couverture de 100% des collages simples est assurée indépendamment de la signature obtenue, même si un faux diagnostic peut résulter dans certains cas. □

Théorème III.1 : Les propriétés III.1 et III.2 sont assurées par le BISC "self-testing".

Preuve □ Suit directement des lemmes III.1 et III.2. □

En effet, le BISC peut être amélioré par le partage de C2 et C1 avec le registre de signature. Dans ce cas, la signature correcte est '100…0' et on doit garantir qu'une faute sur le multiplexeur placé entre C3 et le registre de signature puisse être détectée par l'approche de vérification. Le temps de vérification de la signature est réduit à n+1 cycles d'horloge. On remarque que C3 ne peut pas faire partie du registre de signature, car un collage-à-'0' sur D (figure III.10) pourrait faire transposer une signature erronée ('110…0', au lieu de '010…0')

en une indication de bon fonctionnement ('110'). Même si cette amélioration n'est pas implémentée, le surcoût en surface de silicium résultant est négligeable.

III.5.2.2 Le diagnostic

Si on compare la vérification basée sur le BISC "self-testing" à l'approche de vérification externe, on remarque que le gain en termes de stockage de signatures est très important car le même résultat de vérification est attendu de la part de n'importe quel circuit sur la carte. En termes de temps, au lieu d'utiliser $\sum_{i=1}^{m} n_i$ cycles d'horloge pour décaler la chaîne complète de signatures vers le processeur de test de la carte, seuls $\{[\max(n_i) + 3] + 3m\}$ cycles sont nécessaires pour la vérification des signatures et le décalage des résultats obtenus (m est le nombre de circuits sur la carte et n_i est la longueur de la signature du circuit i).

Comme dans l'approche de vérification externe de signatures, le test des registres de résultat de la vérification intégrée doit être réalisé avant que le BIST des circuits ne soit déclenché. Sinon, les fautes à l'origine soit d'un faux diagnostic, soit du masquage d'autres résultats de vérification (figure III.11) empêcheraient la localisation de manière efficace de la faute sur la carte. Ainsi, si on combine les schémas présentés dans les figures III.8 et III.10, la procédure en deux étapes donnée ci-après pourra être utilisée pour le diagnostic :

premier pas : diagnostic de m Registres BISCs (BRs)

1. activer RUNBIST dans tous les circuits
2. précharger tout BR avec un vecteur de test
3. for (i=1 to m)
   - 3.1. décaler vers le processeur de test le contenu de BR_i en même temps que le vecteur de test pour les BRs est décalé dans les circuits fonctionnels
   - 3.2. if (le contenu de BR_i est incorrect)
     - then {tester seulement BR_{i-1} en même temps que le vecteur de test pour les BRs est décalé dans le circuit}
     - else {}
     - if (BR_{i-1} est sans faute)
       - then {remplacer circuit i et recommencer}
       - else {tester seulement BR_i en même temps que le vecteur de test pour les BRs est décalé dans le circuit}
     - if (BR_i est sans faute)
       - then {remplacer circuit (i-1) et recommencer}
       - else {remplacer circuits i and (i-1) et recommencer}
   - 3.3. next i

deuxième pas□diagnostic de l'autotest des circuits

4. appliquer le BIST et attendre la fin de l'autotest le plus long
5. for (i=1 to m)
   - 5.1. décaler vers le processeur de test le contenu de BR_i
   - 5.2. if (le contenu de BR_i est correct)
     - then {$autotest_i$ est bon}
     - else {$autotest_i$ est mauvais}
   - 5.3. next i

Le vecteur de test pour les BRs mentionné dans le premier pas peut être quelconque contenant au moins un '0' et un '1'. Par exemple, si pour l'initialisation de la vérification de signature '010' est préchargé dans le registre BISC, une séquence de type '1'-baladeur sera appliquée à chaque registre à décalage. Le décalage dans le chemin de balayage de la carte de ce vecteur de test doit être accompli afin d'assurer que le registre BISC du circuit m est complètement testé. Finalement, on remarque qu'une procédure de diagnostic aussi simple que celle présentée peu être facilement intégrée dans le processeur de test BS en utilisant le schéma montré dans la figure III.12.

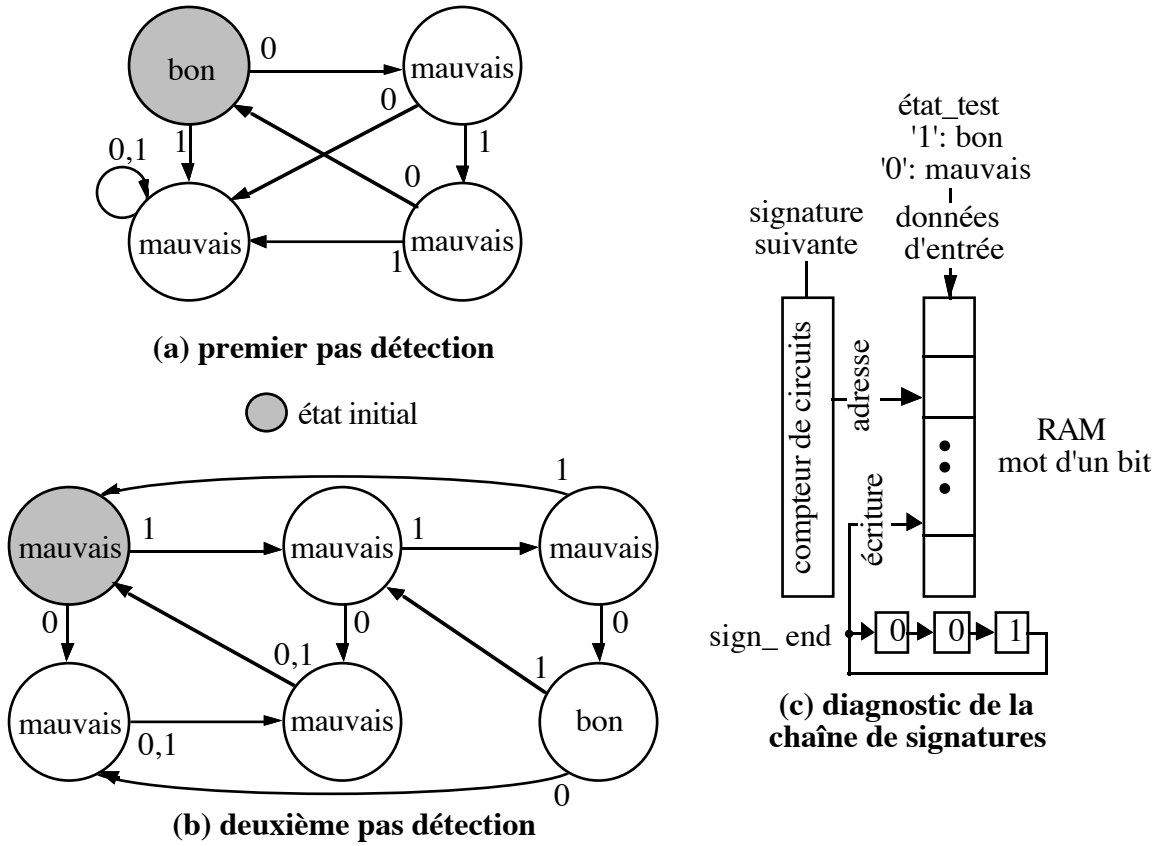


Figure III.12. La réalisation matérielle de la procédure finale de diagnostic.

III.6 Conclusion

Dans ce chapitre le contexte évolutif du test hors-ligne des cartes basées sur le standard IEEE 1149.1 a été parcouru et des solutions pour chacune des étapes ont été proposées. Nous avons étudié le test de cartes partiellement "boundary scan", cas très courant à présent chez les industriels, et le test de cartes BS entièrement autotestables, étape ultime du test hors-ligne de cartes.

Les premières expérimentations réalisées avec notre méthode d'unification du test et du diagnostic de "clusters" et connexions ont donné des résultats très prometteurs pour le cas combinatoire. Il nous reste appliquer et évaluer cette méthode pour le cas séquentiel. Suite à la validation de notre outil pour les deux cas, il pourrait constituer un complément à des produits commerciaux, comme par exemple le PM 3790 BSD de Fluke&Philips [Phi93]. Au niveau diagnostic de fautes dans les "clusters", ce dernier se limite à l'indication des entrées de contrôle et des sorties fautives des "clusters" présumés défaillants.

Les schémas pour le BIST de connexions et pour la vérification de signatures d'autotest discutés dans ce chapitre requièrent, à leur tour, que la carte intègre un processeur de test BS, que tout autre circuit fonctionnel intègre les cellules BS modifiées, le BIST basé sur le registre BS et sur un état supplémentaire de précharge, et le BISC pour la vérification de signature. Cette situation est plus acceptable dans une société qui conçoit et produit tout chez elle, du circuit à la carte, et ensuite au système qui les intègre. Cette société peut implémenter le BIST de connexions et de circuits de manière hiérarchisée afin de justifier plus facilement les coûts engendrés. D'autre part, à moins que différents constructeurs d'ASICs ("Application Specific Integrated Circuits") n'adoptent les approches proposées, le BIST en général, mais surtout le BIST de connexions, est sans doute d'utilité restreinte pour les cartes où les circuits ont des origines diverses.

En effet, les techniques étudiées dans ce chapitre ne peuvent se rendre utiles qu'au test en dehors du fonctionnement de la carte. Pourtant, dans des applications où la sécurité est prioritaire, il est possible que la détection concurrente d'erreurs soit partie intégrante de la spécification fonctionnelle du système à concevoir. Lorsque le besoin d'un test en-ligne vient s'ajouter au test hors-ligne, l'unification de ces deux types de test devient cruciale. Ceci sera présenté dans les chapitres qui suivent.

CHAPITRE IV

LES CARTES "BOUNDARY SCAN SELF-CHECKING"

IV.1 Introduction

Les techniques pour le test hors-ligne, couvrant une partie des tests nécessaires aux circuits intégrés, cartes et modules, peuvent être appliquées soit à la validation de prototypes, soit au test de fabrication, soit au test de maintenance. Ces techniques, se présentant sous la forme intégrée (BIST), ont déjà atteint un niveau élevé de maturité par rapport aux critères de couverture de fautes, de surcoût en silicium et de temps d'application de test. Au niveau carte et module, on peut, à présent, compter sur le standard IEEE 1149.1 basé sur l'architecture "boundary scan" (BS) et offrant des moyens efficaces à l'exécution du test hors-ligne de circuits et connexions. Enfin, comme on a vu aux chapitres II et III, l'association des techniques BIST et BS fait qu'il en résulte une réduction très importante de la complexité de test.

L'autre partie des tests nécessaires aux circuits, cartes et modules, est relative à la capacité de détection d'erreur en-ligne□caractéristique essentielle pour les systèmes où le mauvais fonctionnement peut, par exemple, engendrer un accident. Ceci est le cas des systèmes ferroviaires, automobiles et nucléaires, où les erreurs doivent être détectées avant la contamination d'autres unités et à un point où la réparation est encore possible. Une telle capacité de détection d'erreur en cours de fonctionnement peut être assurée à travers l'utilisation de circuits "self-checking".

Dans ce contexte, on peut conclure que seulement l'unification des tests hors-ligne et en-ligne peut amener à une stratégie qui soit applicable à toutes les phases de la vie utile d'un système. Dans le passé, quelques études ont déjà été réalisées qui adressent le problème de l'unification du test au niveau circuit intégré [Sed80, Nic88]. Ici on est plutôt intéressé à la technique UBIST, brièvement décrite au chapitre I, qui propose l'intégration du BIST à des architectures "self-checking" conventionnelles. Bien qu'on ne se trouve plus au point de départ en ce qui concerne l'unification du test, aucune stratégie existante ne considère ce problème au niveau de la carte. Dans ce chapitre une telle stratégie est proposée. Son principe est d'intégrer les techniques BS et UBIST ce qui résulte en une approche nommée B^2UBIST ("Boundary scan Board Unified BIST").

Dans le paragraphe suivant, les problèmes qui peuvent être résolus par l'intermédiaire de l'unification du test au niveau carte sont discutés. Ensuite, on présente les bases de la conception de circuits intégrés en vue du test unifié de cartes et modules. Ces capacités de test unifié sont donc utilisées pour tester en-ligne et hors-ligne les circuits et les connexions de la carte. Puis, le diagnostic de fautes dans notre environnement de test est présenté et les caractéristiques précédemment étudiées sont appliquées au test unifié de modules.

IV.2 L'unification du test au niveau carte

De manière semblable aux circuits intégrés, l'unification des tests en-ligne et hors-ligne au niveau carte peut apporter une solution efficace aux problèmes suivants□

1) les fautes de fabrication étant fréquemment des fautes multiples, les systèmes "self-checking" conçus pour la détection de fautes simples ne les couvrent pas suffisamment. Ainsi, après la fabrication, ces systèmes doivent être testés pour les fautes multiples.

2) l'hypothèse de faute simple est réaliste car les mécanismes de défaillance liés à la vie de systèmes sont normalement très lents. Pourtant, les fautes latentes peuvent amener à un cumul de fautes et à la violation de cette hypothèse. Un mécanisme permettant la détection de ces fautes est donc d'une importance capitale.

3) quand le mécanisme de détection concurrente d'erreur génère une indication d'erreur, il est très important de pouvoir distinguer entre une faute permanente et une faute transitoire.

Dans ce cadre, l'intégration des techniques BS et UBIST en vue de l'unification des tests permettrait d'utiliser les implémentations BS et BIST afin de tester la carte après fabrication. Ensuite l'activation périodique de la circuiterie BS et BIST amènerait à la détection des fautes latentes évitant donc qu'elles ne se cumulent dans la carte. Finalement, au moment où une indication d'erreur est générée au cours de l'application, l'activation des ressources BS et BIST permettrait de vérifier s'il s'agit d'une faute permanente ou bien d'une faute transitoire.

IV.3 La conception de circuits en vue du test unifié

Notre objectif étant d'intégrer d'une manière adéquate les architectures BS et UBIST, le protocole entre le niveau de la carte et celui des circuits doit être constitué, d'une part d'instructions de test (partant du processeur de test de la carte vers ses circuits fonctionnels) et, d'autre part, d'indications d'erreur (partant des circuits fonctionnels vers le processeur). Ainsi, l'interface entre ces deux niveaux doit être composée, dans un premier temps, par des registres d'instructions de test, auxquels on associe des instructions pour l'activation du test en-ligne, par des registres d'indication d'erreur et par un mécanisme permettant l'observation des indications globales d'erreur des circuits pendant l'exécution de l'application. Ensuite, pour que les connexions de la carte soient aussi prises en compte durant le test en-ligne, les registres "boundary scan" doivent être conçus de telle sorte que la compatibilité avec les capacités "self-checking" soit assurée. Dans la suite, chacun de ces points sera examiné en détail.

IV.3.1 Le registre d'indication d'erreur

Des registres de données compatibles avec le standard IEEE 1149.1 doivent être ajoutés à l'architecture UBIST, afin d'associer à chaque indicateur global d'erreur (GEI) un moyen de stockage dont le contenu sera utilisé pour le diagnostic de fautes. Ces GEIs doivent être conçus d'après un schéma qui assure la mémorisation soit d'une indication d'erreur survenant à ses entrées, soit d'un mot hors-code résultant des ses propres fautes (figure IV.1). Une approche de mémorisation d'erreur asynchrone a été présentée au chapitre I au cours de l'étude des interfaces "strongly fail-safe". Ici on s'intéresse plutôt à un schéma synchrone vu que la compatibilité avec le standard IEEE 1149.1 est recherchée. Dans l'approche donnée figure IV.1 la synchronisation est assurée, tout en gardant les principes introduits au chapitre I□la duplication des indicateurs d'erreur et le rebouclage croisé de ces indicateurs.

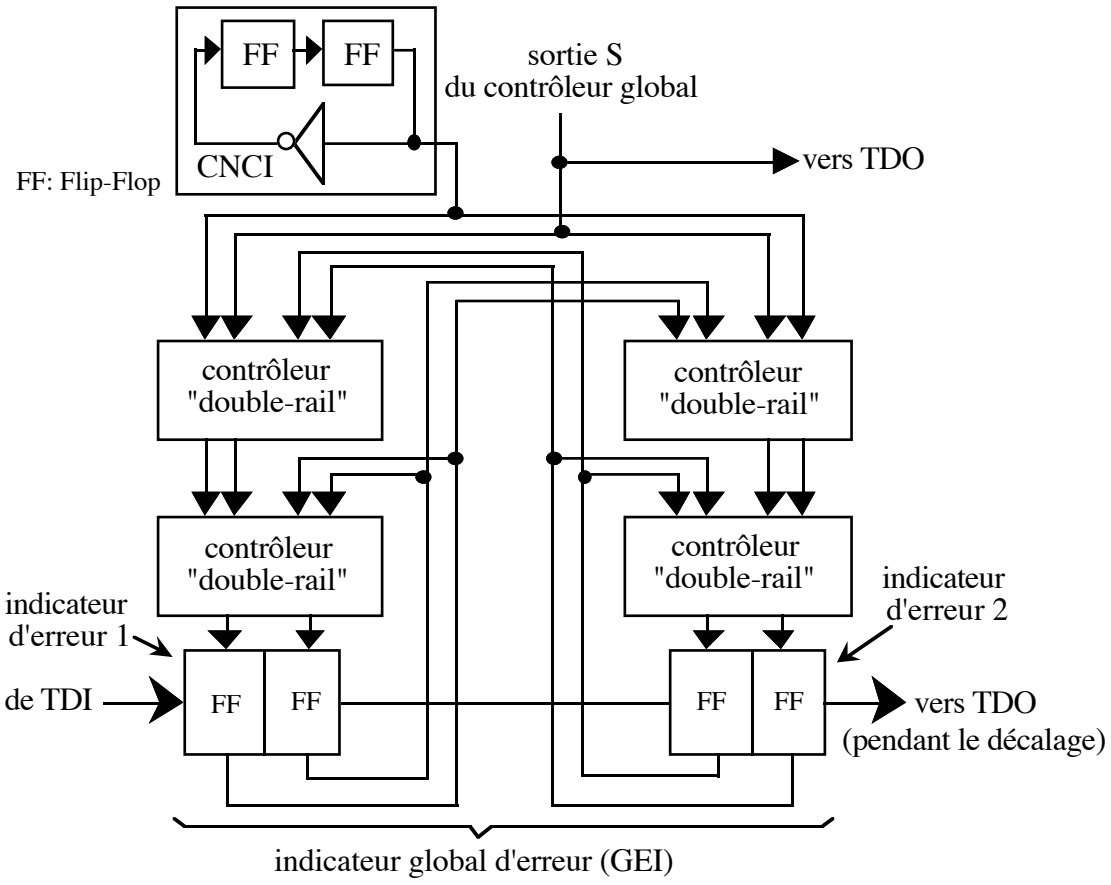


Figure IV.1. Le registre GEI et la mémorisation d'erreur.

Spécialement pour le test en-ligne, il est important qu'une indication d'erreur soit mémorisée immédiatement après sa détection, car pendant l'exécution de la procédure de diagnostic il est possible que les sorties erronées du bloc fonctionnel reprennent des valeurs appartenant au code. En fait, cette approche est aussi extrêmement utile pour le test hors-ligne,

puisqu'une erreur originaire de la phase de test UBIST (chapitre I) sera mémorisée par le registre GEI de la même façon, et l'observation de son contenu ne sera pas nécessaire avant la fin de la phase BIST.

De manière semblable aux autres contrôleurs du circuit, les contrôleurs composant la circuiterie de mémorisation doivent être proprement activés pendant la phase de test UBIST, afin d'assurer leur propriété "strongly code disjoint" (définition I.7). L'utilisation des séquences de type CNCI ("Code/NonCode Indicator", chapitre I) ='00110011…', S='11001100…' assure l'application de tous les mots du code aux entrées des contrôleurs et, par conséquent, le test qui se fait nécessaire. L'implémentation du circuit CNCI proposée figure IV.1, associée à une sous-phase de test qui applique aux entrées du contrôleur global deux mots du code suivis de deux mots hors-code et ainsi de suite, génère les séquences données ci-dessus. Une autre possibilité serait l'initialisation des analyseurs de signatures des blocs fonctionnels (méthode UBIST, chapitre I) de façon à avoir, après l'application de la séquence de test, un ensemble de signatures produisant une séquence de type '11001100…' [McS86]. La vérification de cet ensemble par le contrôleur global, associée à la configuration proposée pour le CNCI, fournirait les séquences nécessaires au test de la circuiterie de mémorisation d'erreur.

IV.3.2 Les instructions pour le test en-ligne

Deux nouvelles instructions de test permettant la détection concurrente d'erreur sont ajoutées au jeu d'instructions du standard IEEE 1149.1□CASCADE et PARALLEL. Les deux sélectionnent les registres GEIs comme registres de données, tout en les connectant au chemin de balayage de la carte à travers lequel les opérations de décalage de données sont accomplies (figure IV.2). Dans l'état "Run-Test/Idle" du contrôleur TAP (chapitre II), CASCADE ou PARALLEL étant actives, les opérations de chargement des registres GEIs seront contrôlées par l'horloge du système (SCK), permettant ainsi le diagnostic de fautes suivant la détection en-ligne d'une erreur. D'autre part, pendant la phase de test hors-ligne, les opérations de chargement et de décalage des registres GEIs seront contrôlées par l'horloge du standard (TCK).

IV.3.3 L'observabilité en-ligne des indications d'erreur

D'après ce qui a été dit au sujet de la technique UBIST au chapitre I, on peut conclure que la seule variable d'intérêt à la détection concurrente d'erreur au niveau de la carte est la sortie S des contrôleurs globaux des circuits. Etant donné que le chemin de balayage de la carte n'est utilisé que dans les états "Shift-DR" et "Shift-IR" du contrôleur TAP (quand le signal "ENABLE" est actif), une façon efficace d'observer la sortie du contrôleur global pendant

l'application serait de connecter S à la broche TDO du circuit par l'activation de CASCADE ou PARALLEL et après avoir atteint l'état "Run-Test/Idle". Afin de propager S à travers le chemin de balayage de la carte, TDI peut être connectée à l'entrée du contrôleur global du circuit par l'activation de l'instruction CASCADE. L'observabilité de la sortie S au niveau de la carte sera donc assurée par l'intermédiaire du schéma donné figure IV.3.

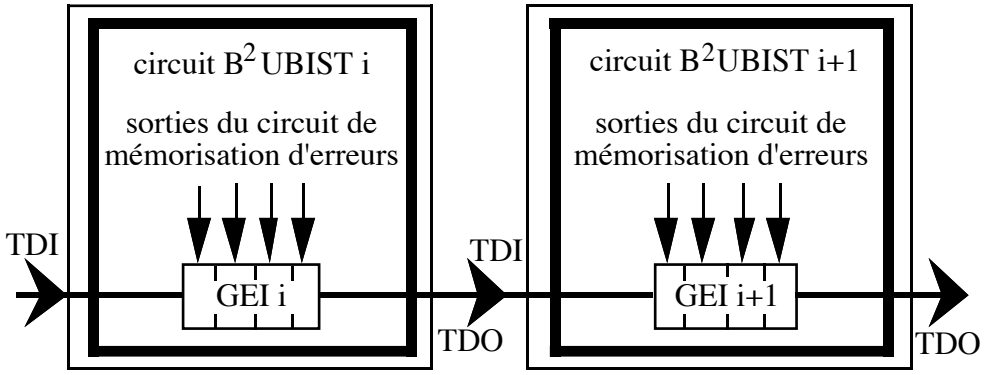


Figure IV.2. La chaîne de décalage d'indicateurs globaux d'erreur.

Afin de mieux comprendre la totalité de la circuiterie donnée figure IV.3, passons à l'examen de l'état des entrées et sortie du contrôleur global pour chacun des cas possibles. En tout il y a six possibilités□

1) dès l'activation de l'instruction CASCADE (CASCADE='1', TEST1='0', TEST2='0', TEST3='0') deux cas différents peuvent surve[redacted]

a) si le contrôleur TAP se trouve dans l'état "Run-[redacted]Idle" (RTI='1', SCAN='0', ON_LINE='1'), TDI sera connectée à une entrée du contrôleur gl[redacted]al (l'autre entrée de la paire sera mise à '1'), les indicateurs intermédiaires d'erreur, les [redacted]aux [redacted]s circuiteries BIST et BS seront connectés aux autres entrées de ce contrôleur et i[redacted]ppa[redacted] sur TDO. La phase de test en-ligne sera en cours d'exécution. La valeur logique '1' étant produite par le contrôleur en cas d'absence d'erreur dans le circuit, le bon fonctionnement de ce dernier portera, au niveau de la carte, sur la valeur '0' car un "buffer" inverseur est utilisé pour la broche TDO. Une telle convention fait que les circuits-ouverts affectant le chemin de balayage de la carte peuvent être détectés pendant l'exécution de l'application grâce aux résistances "pull-up" des broches TDI (chapitre II).

b) sinon (RTI='0', SCAN='0', ON_LINE='0'), TDI sera connectée à une entrée du contrôleur global (l'autre entrée de la paire sera mise à '0'), aux autres entrées de ce contrôleur seront imposées des valeurs codées "double-rail" et TDO sera dans un état de haute impédance. Comme on le verra au paragraphe IV.4.2.1, cette configuration s'avère utile à l'activation du

chemin de propagation de S.

Figure IV.3. Multiplexage du chemin de propagation de S et du chemin de balayage de la carte.

2) l'activation de l'instruction de test PARALLEL (PARALLEL='1', CASCADE='0', TEST1='0', TEST2='0', TEST3='0', RTI='1', SCAN='0', ON_LINE='1') fait que l'entrée du contrôleur global est isolée de la broche TDI, que les indicateurs intermédiaires d'erreur, les signaux des circuiteries BIST et BS, sont connectés aux autres entrées de ce contrôleur et que Si est connecté à la broche TDO par l'intermédiaire du "buffer" inverseur. La phase de test en-ligne sera en cours d'exécution.

3) pendant l'exécution de la phase TEST1 de la technique UBIST (TEST1='1', TEST2='0', TEST3='0'), ou bien de la phase TEST2 (TEST1='0', TEST2='1', TEST3='0'),

l'instruction RUNBIST étant active (CASCADE='0', PARALLEL='0') et le contrôleur TAP étant dans l'état "Run-Test/Idle" (RTI='1', SCAN='0'), le comportement attendu est semblable à celui décrit au point 2 ci-dessus, sauf pour la broche TDO qui devra se mettre en haute impédance.

4) pendant l'exécution de la phase TEST3 de la technique UBIST (TEST3='1', TEST1='0', TEST2='0'), l'instruction de test RUNBIST étant active (CASCADE='0', PARALLEL='0') et le contrôleur TAP étant dans l'état "Run-Test/Idle" (RTI='1', SCAN='0'), le générateur de vecteurs de test se trouve branché aux entrées du contrôleur global. TDO est dans un état de haute impédance.

5) pour le décalage de données de test (SCAN='1', RTI='0', TEST1='0', TEST2='0', TEST3='0'), n'importe quelle instruction de test étant active, des valeurs "double-rail" (définition I.15) seront imposées aux paires d'entrée du contrôleur global, sauf celle connectée à la sortie du multiplexeur de registres BS (dénoté BS dans la figure IV.3). Cette paire d'entrées du contrôleur global aura l'un de ses signaux mis à '1' et l'autre propagera la valeur logique à décaler (BS). Compte tenu de la fonction de transfert de ce contrôleur (chapitre I), qui produira la valeur logique '1' pour un mot du code de type '10' et la valeur '0' pour un mot en dehors du code de type '11', le complément de BS apparaîtra sur Si. Puisqu'une opération de décalage est en cours, le "buffer" inverseur de sortie est actif et le signal BS apparaîtra sur la broche TDO du circuit. Etant donné que le chemin de balayage BS et le chemin de propagation de S se superposent, l'exécution d'opérations de décalage activera les deux chemins durant la phase de test hors-ligne de la carte.

6) pour tous les autres cas (l'exécution de l'instruction EXTEST, par exemple), une valeur codée "double-rail" sera imposée à toute entrée du contrôleur global du circuit.

L'ajout de la circuiterie BS et BIST à une architecture "self-checking" ne doit pas détruire la propriété "strongly code disjoint" du contrôleur global du circuit. Ainsi, les circuiteries BS et BIST doivent être conçues de telle sorte que toute faute simple affectant un signal qui contrôle le chemin de données du contrôleur global□a) soit détectée par un mot du code pendant l'opération normale du circuit, ou, b) soit détectée par un vecteur de test généré pendant la phase de test hors-ligne, ou encore, c) ne soit pas détectable mais que le contrôleur reste toujours capable de transposer un mot hors-code d'entrée en un mot hors-code de sortie, et si une nouvelle faute survient, on retombe dans le cas a), b) ou c). Pour assurer cela, on doit utiliser une implémentation de la circuiterie de commande de test UBIST et BS qui soit "self-checking" par rapport aux signaux qui contrôlent le chemin de données du contrôleur global du circuit. Ceci est représenté en figure IV.3 par le codage "double-rail" des signaux.

IV.3.4 Le registre "boundary scan self-checking"

Afin d'accomplir le test en-ligne des connexions, un code doit être associé à chaque sous-ensemble différent de données communiquant d'un circuit à l'autre sur la carte. Les registres BS doivent donc être augmentés pour accommoder ce code. Les circuits UBIST fournissant déjà des sorties primaires codées (chapitre I), il reste à concevoir des contrôleurs à leurs entrées afin de vérifier les codages utilisés. Si le code nécessaire à la détection concurrente d'erreur sur les connexions n'est pas semblable au code disponible, un générateur du nouveau code de sortie devra être intégré à l'architecture du circuit.

Par l'utilisation d'un contrôleur intermédiaire intégré (contrôleur 0, figure IV.4), la vérification en-ligne des entrées du circuit par rapport au code reçu va générer un résultat codé "double-rail" à la sortie de ce contrôleur. Cet indicateur d'erreur sera envoyé au contrôleur global du circuit qui va signaler l'occurrence d'une faute affectant les connexions d'entrée du circuit.

Dans ce contexte, chaque sous-ensemble différent de données communiquant d'un circuit à l'autre sur la carte peut être modélisé comme présenté figure IV.4 - où on suppose que le bloc fonctionnel N génère le code nécessaire au test en-ligne des connexions.

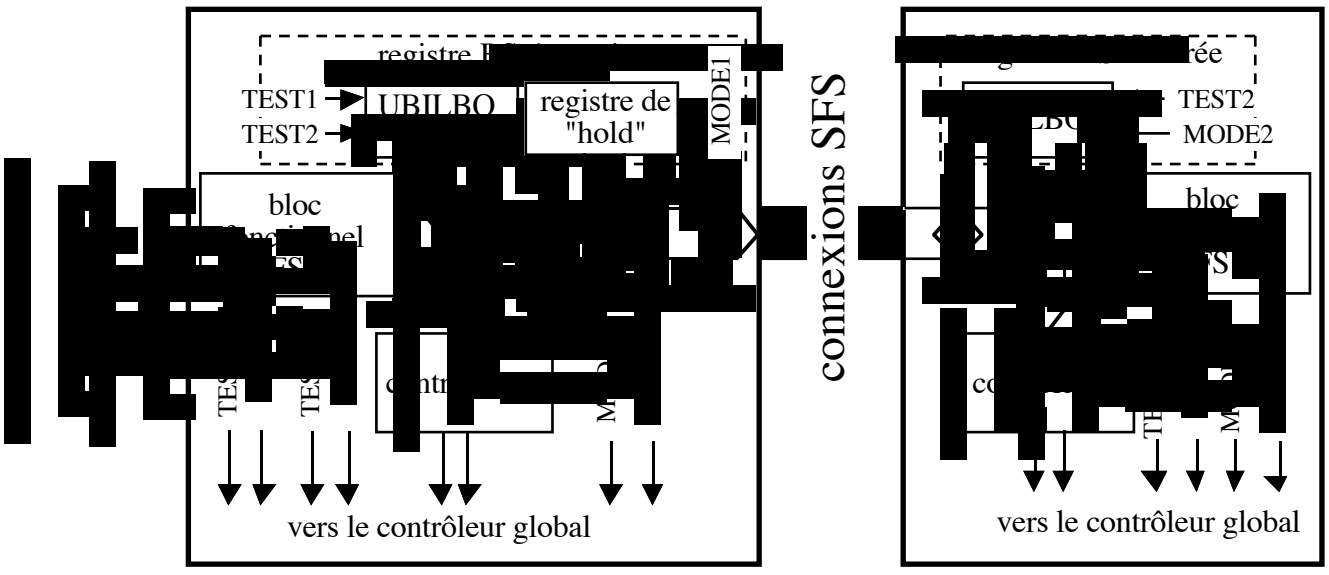


Figure IV.4. Le registre "boundary scan self-checking" et le test en-ligne des connexions.

Mise à part l'intégration de ressources "self-checking" pour le test des connexions, les circuits B²UBIST doivent maintenir, autant que possible, la compatibilité avec le standard IEEE 1149.1 (chapitre II). Les moyens d'exécution des instructions RUNBIST, INTEST,

EXTEST et SAMPLE/PRELOAD doivent donc être présents. Le tout peut être assuré par l'augmentation du registre BS, d'une part, et par la commande de ce registre à travers des signaux nommés MODE1 et MODE2 d'autre part (figure IV.4). Ainsi, l'activation d'INTEST/RUNBIST fera que le coeur du circuit sera complètement déconnecté du monde extérieur et que les connexions de la carte seront contrôlées par l'interface de sortie du circuit (MODE1='1', MODE2='1'); l'activation d'EXTEST (MODE1='1', MODE2='0') ou de SAMPLE/PRELOAD (MODE1='0', MODE2='0') fera que les connexions de la cartes seront contrôlables et/ou observables.

Concernant le te[redacted]ues comme des blocs fonctionnels qui doivent a[redacted]ES, définition I.4) pour un modèle de fautes donné. A[redacted]ourts-circuits et circuits-ouverts simples, le code de pa[redacted]le signal qui contrôle les sorties d'un circuit (MODE1, [redacted]si[redacted]qui contrôle les entrées du circuit commandé par ces so[redacted]2, [redacted]ux de commande de la circuiterie BIST (TEST1, [redacted]2, [redacted]ifiés par les contrôleurs globaux des circuits respectifs. Ceci est dû au fait qu'une faute simple qui affecte l'un de ces signaux peut produire une erreur multiple (définition I.10) aux entrées du contrôleur des connexions (contrôleur 0 dans la figure IV.4). D'autre part, l'utilisation d'un code détecteur d'erreurs unidirectionnelles (définition I.9), comme par exemple le code de Berger (définition I.14), permettrait la détection de tout type de court-circuit comme le démontre [CLW90], tandis que des erreurs multiples seraient détectées par un code dupliqué (définition I.15).

Dans le cas des connexions, la détermination du code à utiliser doit également tenir compte des contraintes d'interface des circuits et du surcoût en surface engendré pour la carte. Si le modèle de faute mène à l'utilisation du code de parité, par exemple, une seule broche additionnelle sera requise pour chaque sous-ensemble différent de connexions entre deux circuits sur la carte. Le surcoût en surface sera donc minimum du point de vue de la carte. En dépit de cela, étant donné que les sous-ensembles de données sont toujours déterminés par l'application, il est clair que cette approche demandera des versions spécifiques d'interface pour le même circuit utilisé dans des cartes implémentant des fonctions légèrement différentes. D'autre part, si un code du type duplication est utilisé, il n'y aura qu'une version pour le circuit, vu que tous les sous-ensembles possibles de données auront leur codes disponibles à son interface. Pourtant, le nombre de broches fonctionnelles pour les circuits va doubler et, par conséquent, le surcoût en surface pour la carte sera maximum.

IV.4 Le test unifié de la carte

D'une part, l'implémentation du circuit selon la technique UBIST fait qu'il est possible

de tester sa logique interne hors fonctionnement, tant avant qu'après son assemblage sur une carte. Ensuite, l'intégration dans le même circuit de la technique "boundary scan" fait que les moyens d'accès électronique à travers lesquels le test hors-ligne de ses connexions sera appliqué sont disponibles.

D'autre part, l'adoption des stratégies pour le test en-ligne présentées précédemment mène à la disponibilité sur la broche TDO de l'état de la logique interne et des connexions d'entrée du circuit et à la possibilité d'utiliser la broche TDI pour l'observation d'un indicateur d'erreur extérieur pendant l'opération normale du circuit.

Il ne nous reste donc qu'à intégrer ces deux parties et à définir dans son intégralité la procédure finale de test de la carte tout en tirant profit de leur complémentarité. Ceci fait l'objet des paragraphes suivants.

IV.4.1 Le test en-ligne

En ce qui concerne la détection concurrente d'erreur, on est plutôt intéressé à des mécanismes pour la compression et la propagation d'indicateurs globaux d'erreur à travers les circuits de la carte. Ainsi, on propose trois approches différentes pour le traitement de ces informations de test: l'approche cascadée, la vérification parallèle d'indicateurs d'erreur et l'approche mixte. Ces approches sont décrites ci-après.

IV.4.1.1 L'approche cascadée

Dans l'approche cascadée, donnée figure IV.5, le complément du signal S venant du circuit précédent atteint l'entrée du contrôleur global du circuit suivant dans la cascade à travers la broche TDI. De cette façon un résultat direct et compact pour le test en-ligne de la totalité de la carte sera disponible sur la broche TDO du dernier circuit de la cascade. Vu que les entrées TDI sont dotées de résistances "pull-up", le choix de propager le complément de S fait qu'il est possible de détecter un circuit-ouvert sur le chemin de balayage pendant le temps d'exécution du test en-ligne de la carte.

A l'exception du premier circuit de la cascade, où l'instruction de test PARALLEL sera programmée, tous les autres circuits auront le code choisi pour l'instruction CASCADE chargé dans leur registres d'instructions BS.

Pour cette approche, la fréquence maximale d'opération dépendra du temps de stabilisation de toutes les sorties des contrôleurs sur la carte. Si on suppose que le premier circuit est celui qui présente le délai le plus important de ses contrôleurs par rapport aux autres circuits qui composent la cascade, le temps d'exécution du test en-ligne pour la carte sera

donnée par :

$$t_{on} = dcn_1 + \sum_{i=2}^{NC} dgc_i, \qquad [I]$$

où, dcn_1 : délai du réseau de contrôleurs du circuit 1;
NC : nombre de circuits sur la carte;
dgc_i : délai du contrôleur global du circuit i.

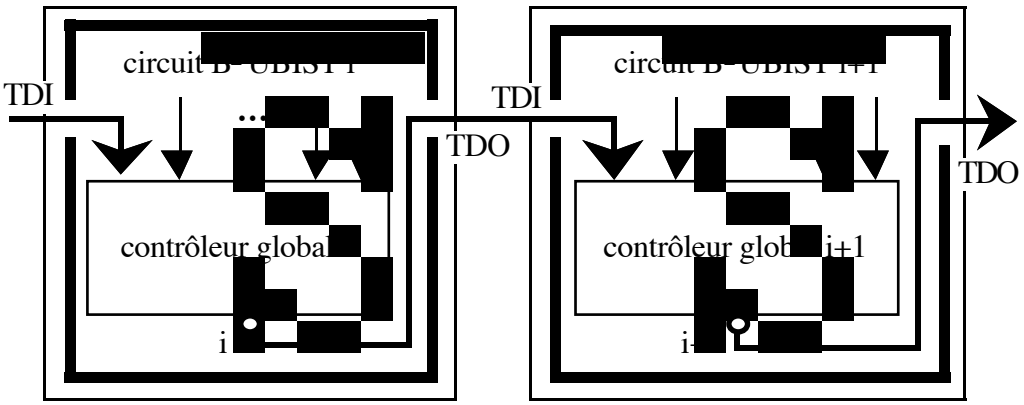


Figure IV.5. L'approche cascadée.

Si le circuit qui présente le délai maximum de son réseau de contrôleurs n'est pas celui placé sur la première position de la cascade, le temps total pour le test en-ligne de la carte sera certainement inférieur à celui donné par l'expression [I].

IV.4.1.2 L'approche parallèle

Dans l'approche parallèle, donnée figure IV.6, le complément de toute sortie S d'un contrôleur global est envoyé à l'entrée d'un contrôleur global pour la carte. Ce contrôleur est responsable de leur évaluation en parallèle, fournissant à sa sortie une indication d'erreur pour l'ensemble de la carte. On va supposer ici que le contrôleur global pour la carte est implémenté comme un circuit intégré spécifique. Pourtant, ce contrôleur pourrait être intégré dans un processeur de test BS (chapitre III), si un tel circuit était prévu pour la carte.

Quel que soit le cas, le contrôleur global pour la carte doit être implémenté selon les stratégies B²UBIST. Les capacités de test que chacune de ses entrées est censée avoir sont présentées figure IV.7. Les entrées du contrôleur originaires de TDI et BS sont implémentées d'après les mêmes principes donnés figure IV.3. En fait, concernant le test en-ligne, le contrôleur global pour la carte peut être conçu pour supporter seulement l'instruction CASCADE, profitant du fait que la connexion de sa broche TDI à la broche TDO du dernier

circuit fonctionnel du chemin de balayage sera sur la carte de toute façon.

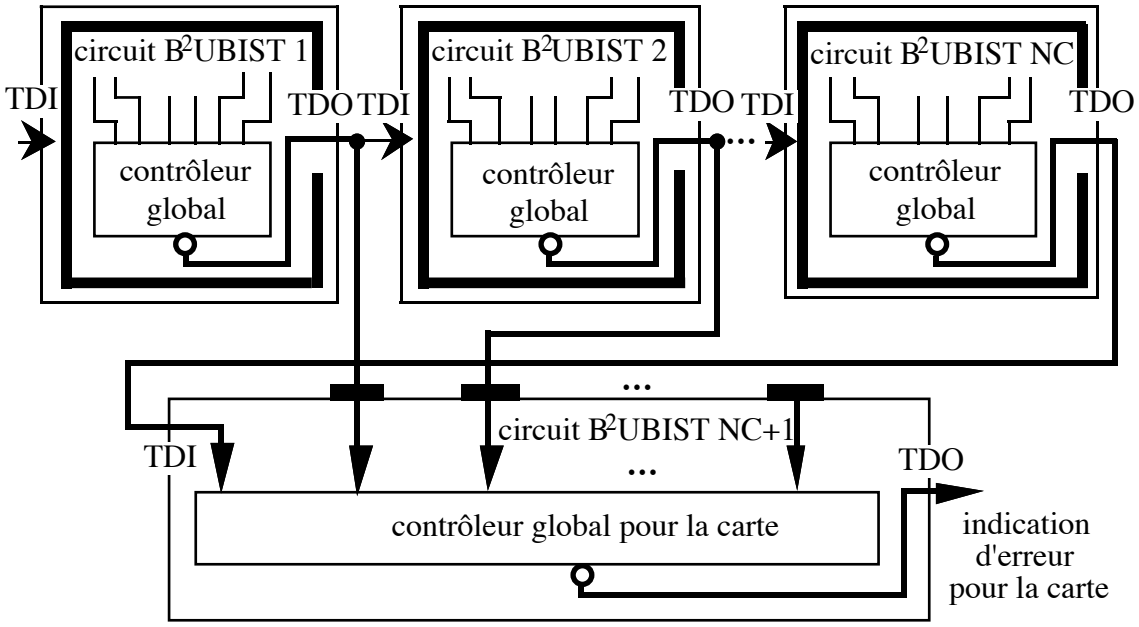


Figure IV.6. L'approche de vérification parallèle.

Afin de permettre le diagnostic de fautes du point de vue du module, la mémorisation d'erreur doit être pourvue selon ce qui a été présenté au paragraphe IV.3.1. Finalement, le circuit où le contrôleur global pour la carte sera intégré doit comporter un registre BS ordinaire (non "self-checking") dont le seul contenu correct possible sera '00…0'.

A l'exception du contrôleur global pour la carte, où l'instruction de test CASCADE sera programmée, tous les autres circuits auront le code choisi pour l'instruction PARALLEL chargé dans leur registres d'instructions BS.

Enfin, pour cette approche, on peut conclure que le temps du test en-ligne pour la carte sera donné par□

$$t_{on} = \max (dcn_i) + dgc_{NC+1}, \qquad [II]$$

IV.4.1.3 L'approche mixte

Le nombre de circuits/connexions additionnels et la longueur de ces connexions détermineront la surface finale de la carte et, par conséquent, le surcoût associé à chacune des approches pour la réalisation du test en-ligne. Le tableau donné figure IV.8 résume ces caractéristiques pour les approches cascadée et parallèle.

En ce qui concerne le temps de test en-ligne, la comparaison entre les expressions [I] et

[II] reprises figure IV.8 donne□

(a) $dcn_1 = \max(dcn_i)$, d'après la déduction de [I];

(b) $\sum_{i=2}^{NC} dgc_i >> dgc_{NC+1}$, dû à la complexité des contrôleurs;

(c) [I] >> [II], de (a) et (b)

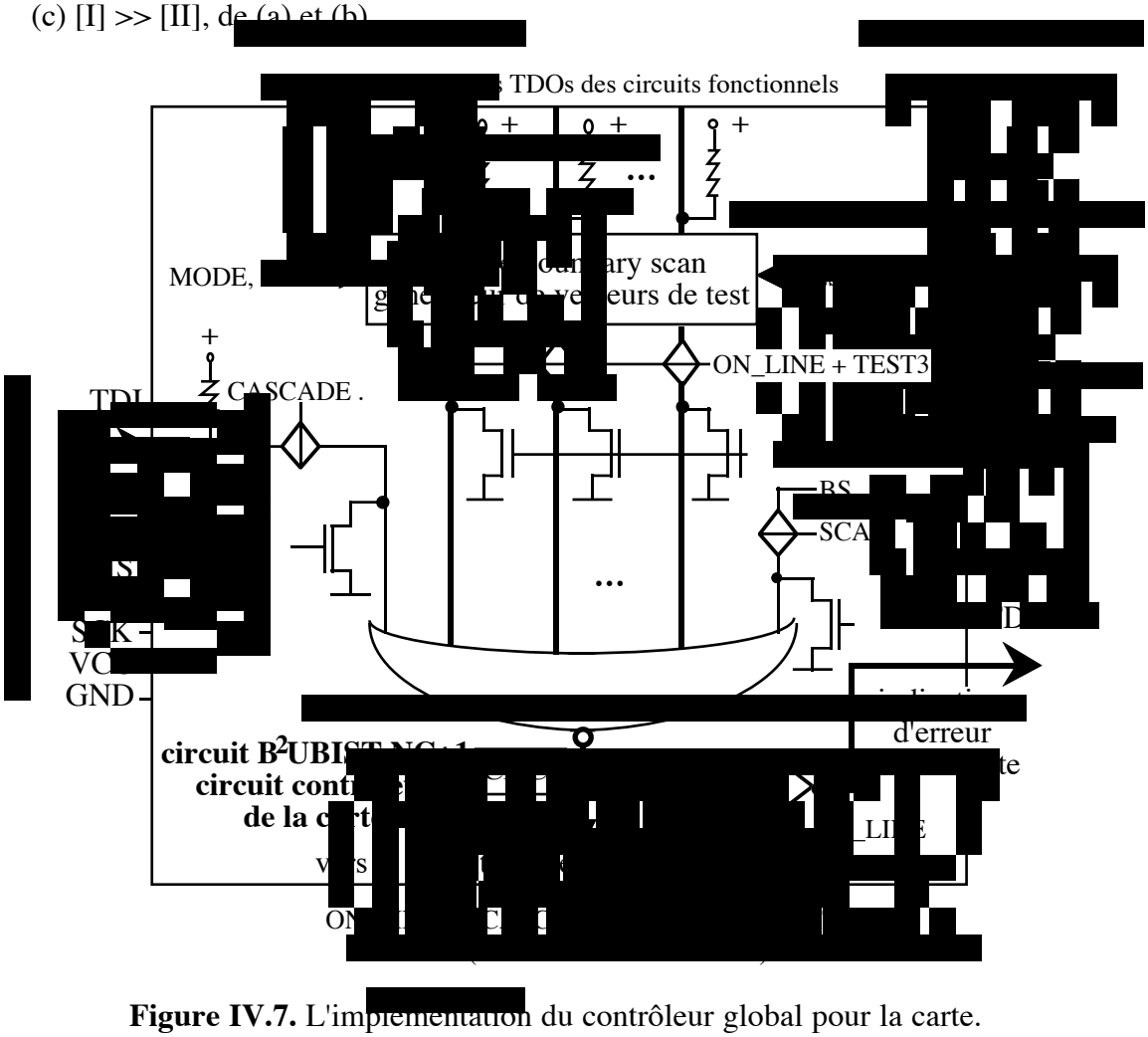


Figure IV.7. L'implémentation du contrôleur global pour la carte.

approche	circuits additionnels	connexions additionnelles	temps pour le test en-ligne
cascadée	0	0	$dcn_1 + \sum_{i=2}^{NC} dgc_i$ [I]
parallèle	1	NC + 6	$\max(dcn_i) + dgc_{NC+1}$ [II]

Figure IV.8. Comparaison entre les approches cascadée et parallèle.

Chaque fois qu'on est confronté à une application demandant une performance en fréquence, **f**, telle que **[I] > 1/f > [II]**, l'utilisation d'une approche mixte cascadée/parallèle (figure IV.9) pourra être envisagée en vue d'une diminution du surcoût en surface de la carte.

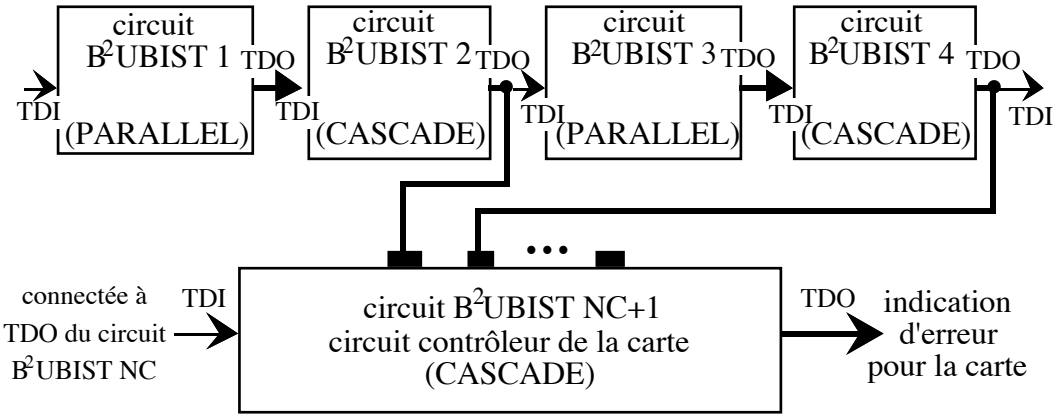


Figure IV.9. L'approche mixte.

IV.4.2 Le test hors-ligne

Le test hors-ligne joue un rôle essentiel dans la détection de fautes suivant la fabrication de la carte, et un rôle aussi important dans l'activation périodique, et alterné avec l'application, de ses circuits et connexions.

Le test hors-ligne de cartes "boundary scan" peut être réalisé comme illustré par la procédure proposée dans [TuY89] et brièvement discutée au chapitre II. Etant donné que le schéma B²UBIST ajoute des capacités de test à la carte, il devient nécessaire de vérifier à quel point les circuiteries additionnelles sont testées et peuvent être utilisées par cette procédure de test. Ainsi, examinons chacun des trois grandes lignes de la procédure [TuY89]□

IV.4.2.1 L'initialisation et la vérification de la circuiterie de test

L'initialisation vise la protection de la carte contre les dommages causés par des conflits potentiels de bus. L'état "Test-Logic-Reset" du contrôleur TAP doit être atteint et un vecteur sécuritaire doit être chargé dans le registre BS en utilisant l'instruction du standard SAMPLE/PRELOAD. Seules des ressources BS sont concernées dans cette étape.

La vérification de la circuiterie de test, à son tour, vise la vérification de l'infrastructure utilisée pour le test de la carte dans les pas suivant. Le chemin de balayage série, le registre d'instructions BS et les registres de données BS, entre autres, doivent être testés (chapitre II). A travers le décalage des contenus de registres d'identification BS, on peut vérifier si

l'assemblage de circuits sur la carte a été accompli comme il fallait.

Dans un environnement B²UBIST, un chemin de balayage spécial est partagé entre les phases de test hors-ligne et en-ligne de la carte. Comme on a vu aux paragraphes IV.4.1.1 et IV.4.1.2, le chemin de propagation de S ne comporte que des '0's pendant le test en-ligne. La faculté de TDI de propager un '1' à l'entrée du contrôleur global doit donc être vérifiée. Ce test est réalisé de manière implicite quand on quitte l'application de la carte et démarre la phase de test hors-ligne alternée. L'instruction CASCADE étant active, le déplacement de l'état "Run-Test/Idle" vers l'état "Capture-DR" du contrôleur TAP fera que le résultat d'un tel test sera chargé dans le registre GEI (paragraphe IV.3.3). Si le contenu décalé de GEI est codé en "double-rail", cette partie du circuit et le registre lui même sont corrects.

Concernant la portion du chemin de propagation de S qui est connectée à la broche TDO, sa procédure de vérification peut être la même que celle appliquée au chemin BS puisque les deux chemins se superposent à ce niveau là (figure IV.3). Du point de vue des connexions TDI/TDO de la carte, il est impératif que les courts-circuits soient détectés (chapitre II). On remarquera qu'un court-circuit du type ET (où '0' prévaut) entre ces broches connecte les sorties de contrôleurs globaux (figures IV.5, IV.6 et IV.9) et, au cas où cette faute reste non détectée, la propriété "strongly code disjoint" des contrôleurs sera détruite. L'une des séquences de test proposées dans [JoH91], qui garantissent la détection de courts-circuits affectant les connexions TDI et TDO, a été présentée au chapitre II. Bien qu'une telle séquence soit utilisée pour la détection de courts-circuits TDI/TDO, pour les approches en-ligne parallèle et mixte il reste au moins une combinaison de fautes qui ne peut toujours pas être détectée (figure IV.10) et qui détruit la propriété "strongly code disjoint" des contrôleurs globaux.

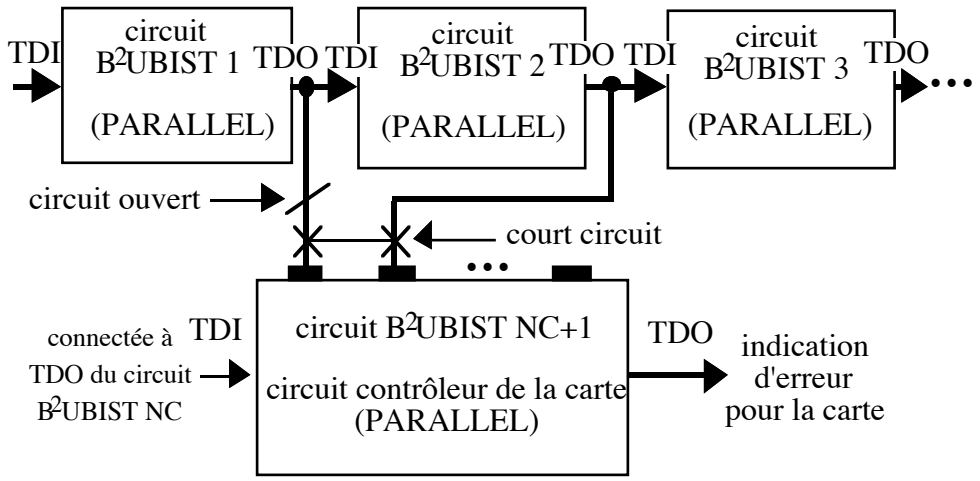


Figure IV.10. Faute multiple non détectable en-ligne.

Afin de détecter ce type de faute multiple, on doit tester de manière explicite la portion du chemin de propagation de S qui ne se superpose pas au chemin de balayage série BS (figure IV.10). Cela peut être facilement réalisé à travers la dissociation du contrôleur global de la carte des circuits fonctionnels par rapport au signal TMS, tout en gardant un seul chemin de balayage série pour la carte. De cette manière, il devient possible d'activer l'instruction EXTEST dans le circuit contrôleur global, en même temps qu'on décale une séquence pour le test des connexions à travers les registres BYPASS des circuits fonctionnels (figure IV.11). TMS0 commandera les contrôleurs TAP des circuits fonctionnels, tandis que TMS1 commandera celui du circuit intégrant le contrôleur global pour la carte. Un ensemble de vecteurs pour le test des connexions sera appliqué par l'intermédiaire des broches TDO dans l'état "Shift-DR" des contrôleurs TAP des circuits fonctionnels. Les réponses au test seront décalées hors du registre BS du circuit contrôleur de la carte dans l'état "Shift-DR", après avoir été recueillies dans l'état "Capture-DR". Pour l'approche de vérification parallèle, la synchronisation des états "Shift-DR" des circuits fonctionnels et "Capture-DR" du contrôleur de la carte peut être assurée par les séquences de commandes TMS0 et TMS1 montrées figure IV.11.

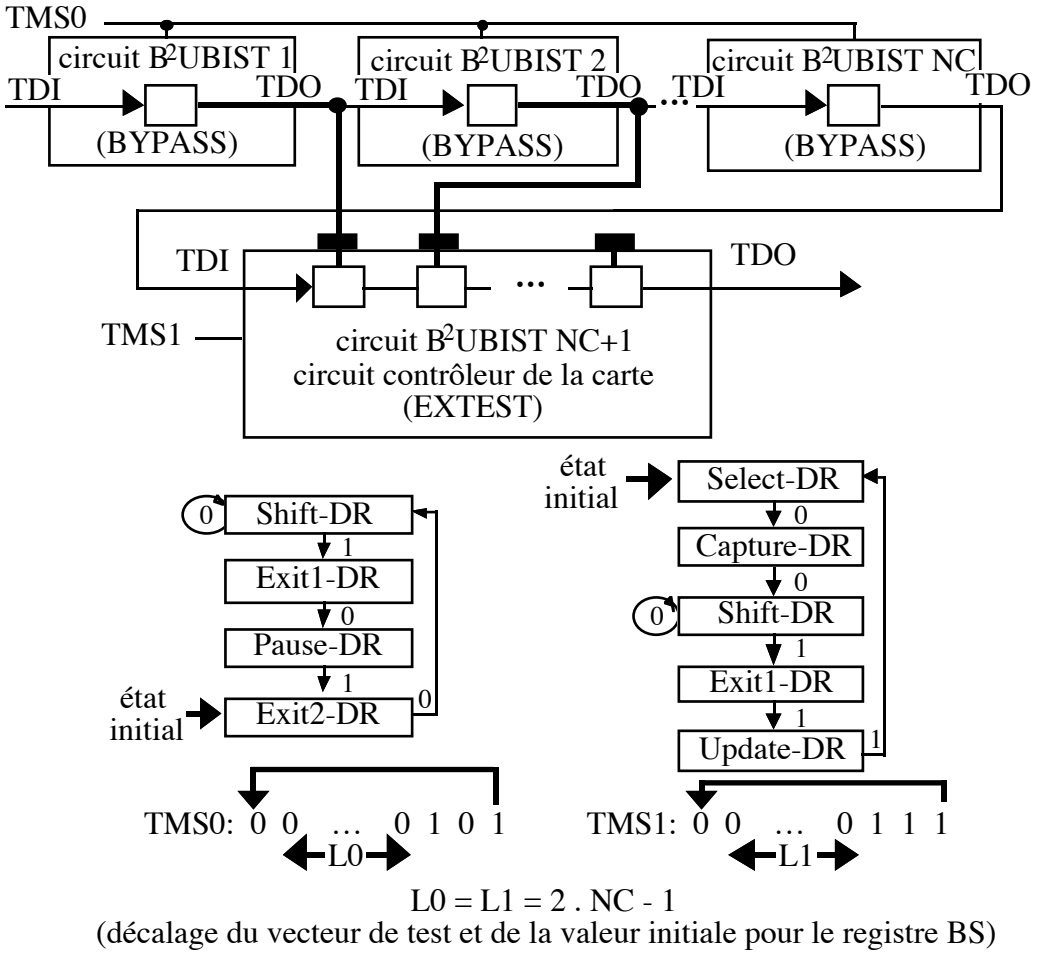


Figure IV.11. La vérification du chemin de propagation de S dans l'approche parallèle.

IV.4.2.2 Le test des connexions

Le test hors-ligne des connexions d'une carte B²UBIST doit vérifier soit les connexions de données, soit les connexions de codage. Puisque ce genre de test a un caractère structurel, même les connexions qui n'ont pas de rapport fonctionnel entre elles doivent être vérifiées les unes vis-à-vis des autres. Ainsi, toute connexion doit être testée par rapport à toutes les autres, ou au moins, par rapport à un sous-ensemble de la carte placé dans son voisinage. Ceci implique l'application de vecteurs de test générés soit par une procédure extérieure à la carte, soit par une procédure intégrée dans le processeur de test BS, ou bien par une procédure intégrée dans chacun des circuits de la carte (chapitre III). En tout cas, ce n'est que par l'activation de l'instruction BS EXTEST que cette tâche peut être accomplie. Il est crucial que les vecteurs appliqués assurent le diagnostic maximum pour les collages, les courts-circuits et les circuits-ouverts affectant les connexions de la carte (chapitre II).

IV.4.2.3 Le test des circuits

La portion UBIST des circuits B²UBIST offre les ressources d'autotest nécessaires aux circuits de la carte. L'intégration dans ces circuits d'une machine à états finis (FSM) partiellement "self-checking" - responsable du contrôle de la phase TEST1, de la vérification des signatures produites pendant TEST1, de la phase TEST2, de la vérification des signatures produites pendant TEST2 et de la phase TEST3, permet l'exécution simultanée du test hors-ligne de tous les circuits montés sur la carte. Cette FSM doit démarrer à partir de la programmation de l'instruction de test RUNBIST qui sélectionnera le registre GEI comme celui qui devra fournir le résultat final de la phase BIST au processeur de test de la carte.

L'isolation de la logique interne du circuit en vue de l'exécution de RUNBIST a été déjà discutée dans le paragraphe IV.3.4. Dans le chapitre I, les moyens requis pour l'exécution des phases UBIST TEST1, TEST2 et TEST3 ont été décrits. Il ne nous reste donc qu'à proposer une stratégie pour la vérification interne des signatures résultantes de l'exécution des phases TEST1 et TEST2, et un schéma pour l'autotest du circuit où se trouve le contrôleur global pour la carte.

L'idée de base de notre approche de vérification de signatures [LAN93] consiste à initialiser les analyseurs de signatures de manière à ce que leur état final résulte dans une séquence de type '00110011…' ([McS86] présente une procédure assez simple pour le calcul des valeurs initiales des analyseurs de signatures). Suite à l'exécution de TEST1/TEST2, les UBILBOs qui contiennent les signatures concernées seront connectés en série et leurs contenus seront décalés vers une entrée du contrôleur global du circuit (figure IV.12). L'autre entrée de la

paire sera mise à '1', de telle sorte que le complément de la chaîne de signatures (une séquence de type '11001100…') atteigne la sortie du contrôleur global et soit fournie à l'une des entrées de la circuiterie de mémorisation d'erreur du circuit. Ceci étant compatible avec l'opération de décalage décrite paragraphe IV.3.3, le simple fait de tenir compte du signal de vérification de signature (SV, figure IV.12) permet la réutilisation de la circuiterie proposée figure IV.3. Finalement, le circuit CNCI proposé paragraphe IV.3.1 sera mis en marche pour générer, sur l'autre entrée de la circuiterie de mémorisation, la séquence '00110011…' complémentaire à celle produite à [illegible].

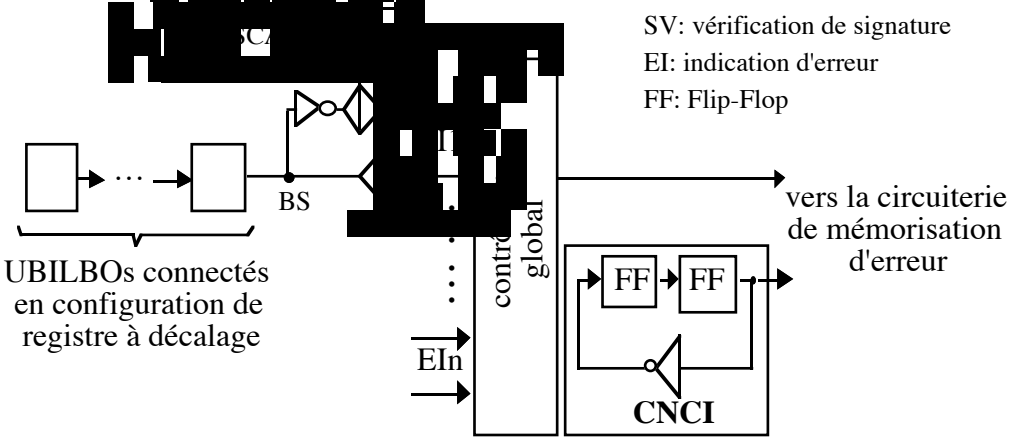


Figure IV.12. La vérification interne des signatures

Etant donné que cette paire de signaux est envoyée aux entrées de contrôleurs "double-rail", la condition sans faute (un mot codé "double-rail") ne sera mémorisée que quand les deux séquences seront correctement générées. Chaque fois qu'une signature erronée est reçue ou qu'une faute affectant le circuit CNCI, ou bien le circuit de mémorisation d'erreur, est détectée, une indication d'erreur (un mot hors du code "double-rail") sera mémorisée dans le registre GEI.

De manière semblable à ce que nous avons vu dans le chapitre III, l'utilisation de cette stratégie, d'une part, réduit le nombre d'opérations de décalage à effectuer sur la carte, car les signatures finales des circuits auront chacune une longueur de 4 bits, et seront donc extrêmement compactes. D'autre part, la mémoire nécessaire au stockage de signatures dans le processeur de test de la carte est aussi fortement réduite, puisqu'il n'y aura que quatre possibilités pour une signature dite sans faute ('0101', '0110', 1001' et '1010'). En plus, l'application des séquences données ci-dessus au circuit de mémorisation d'erreur va assurer à la fois la propriété "strongly code disjoint" de ses contrôleurs "double-rail", car tous les mots du code leur seront délivrés.

L'autotest du circuit contrôleur de la carte, à son tour, doit assurer l'exercice nécessaire à sa porte NON-OU de (NC+2) entrées et à sa circuiterie de mémorisation d'erreur. Les capacités BIST montrées figure IV.7 serviront à cet effet.

Le générateur de vecteurs de test sortira le prochain vecteur d'une séquence '1'-baladeur (chapitre II), chaque fois que le signal TEST3 aura la valeur logique '1'. Quand TEST3 est mis à '0', un vecteur tout-zéro sera appliqué aux entrées de la porte NON-OU. TEST3 sera généré par une structure identique à celle du circuit CNCI utilisé dans l'approche de vérification de signatures (figure IV.12). En supposant l'initialisation '00' pour le générateur de TEST3 et '000…001' pour le générateur de vecteurs de test, la séquence '0000…0000', '0000…0000', '0100…0000', '0010…0000', '0000…0000', '0000…0000',…, '0000…0100', '0000…0010' sera envoyée aux entrées de la porte NON-OU qui devra produire '110011…00' à sa sortie. On remarquera que la séquence produite à la sortie de la porte NON-OU est pareille à celle attendue à la sortie du contrôleur global pour la chaîne de signatures d'un circuit fonctionnel B^{2}UBIST sans faute. Ainsi, l'activation de la circuiterie de mémorisation d'erreur peut être aussi assuré par l'implémentation de CNCI comme proposé dans l'approche de vérification de signatures. Etant donné que la séquence '14-baladeur générée ne peut assurer qu'une partie du test de la porte NON-OU, les vecteurs qui restent ('1000…0000' et '0000…0001') seront générés, respectivement, quand l'instruction CASCADE sera active et le contrôleur TAP se déplacera de l'état "Run-Test/Idle" à l'état "Capture-DR" (CASCADE='1', RTI='0', SCAN='0' et CNCI='1' dans la figure IV.7), et quand un '1' sera décalé vers l'extérieur du circuit (RTI='0', SCAN='1').

IV.5 Le diagnostic de fautes

Après la détection d'une erreur au cours de l'application normale de la carte ou durant la phase de test hors-ligne, on est intéressé à la localisation de la faute responsable de l'apparition de cette erreur, pour pouvoir réparer ou remplacer le matériel endommagé.

Afin d'éviter qu'il y ait une perte d'information compromettant le diagnostic, le circuit pour la mémorisation d'occurrence d'erreur donné paragraphe IV.3.1 et constitué, entre autres, d'un registre dont le contenu est codé "double-rail" (GEI), est intégré dans chaque circuit B^{2}UBIST monté sur la carte.

Etant donné que les registres GEI fournissent des données utiles au diagnostic, analysons comment en tirer profit et comment obtenir d'avantage d'information pour la localisation de fautes□

- Quand une erreur est détectée en-ligne, la localisation de la faute dans une région précise sera fonction de la configuration de la carte pour la propagation des indications d'erreur et du placement des GEIs hors-code dans la chaîne d'indicateurs décalée vers l'extérieur de la carte.

Sous l'hypothèse de faute simple, soit un circuit, soit un sous-ensemble de connexions qui partagent le même codage, soit une partie du chemin de propagation de S, peut être fautif.

D'une part, si l'approche parallèle est implémentée sur la carte, trois cas sont possibles□

(1a) si seulement le circuit contrôleur de la carte présente un GEI hors-code, le circuit lui-même, ou bien l'une de ses connexions d'entrée présente la défaillance;

(2a) pour l'occurrence d'un seul GEI hors-code parmi les circuits fonctionnels, la région candidate à la faute sera composée du circuit qui a généré un tel GEI et de ses connexions d'entrée non partagées par d'autres circuits fonctionnels;

(3a) pour l'occurrence de multiples GEIs hors-code parmi les circuits fonctionnels, la région candidate à la faute sera restreinte aux sous-ensembles de connexions d'entrée partagées par tous les circuits produisant des GEIs hors-code.

Afin d'approfondir le diagnostic, les ressources BIST et le mode de test externe de la technique BS doivent être activés dans les cas (1a) et (2b), et seulement le mode externe BS est nécessaire dans le cas (3a).

D'autre part, quand le schéma de propagation d'indicateurs d'erreur est composé d'une cascade (approche cascadée) ou plusieurs (approche mixte), tous les circuits qui suivent celui produisant le premier GEI hors-code vont aussi mémoriser l'occurrence d'une erreur. Malgré cela, les cas possibles sont similaires à ceux donnés ci-dessus□

(1b) si un circuit contrôleur pour la carte est utilisé dans une configuration mixte, ce cas est identique au cas (1a);

(2b) quand une seule cascade est atteinte, la région candidate à la faute sera composée du circuit fonctionnel placé dans la tête de la chaîne de GEIs hors-code et de ses connexions d'entrée non partagées avec d'autres circuits et toutes celles partagées avec les circuits fonctionnels qui le suivent dans la cascade;

(3b) quand de multiples cascades sont atteintes, la région candidate à la faute sera limitée aux sous-ensembles de connexions d'entrée partagées par les circuits fonctionnels placés dans la tête de chaque cascade produisant des GEIs hors-code, et éventuellement, partagées avec d'autres circuits qui font partie de ces mêmes cascades.

Les possibilités pour approfondir le diagnostic sont essentiellement les mêmes que celles valables pour une approche purement parallèle.

- Quand une erreur est détectée pendant la phase de test hors-ligne, il n'y a que le placement des GEIs hors-code sur la chaîne d'indicateurs d'erreur décalée qui joue dans le diagnostic. Dans ce cas, un GEI hors-code n'identifiera qu'un circuit défaillant. Le mode externe BS devra donc être utilisé pour le diagnostic de fautes de connexions.

IV.6 Le test unifié de modules

Afin de tester en-ligne un module (ensemble de cartes), on peut envisager l'extension des approches proposées paragraphe IV.4.1 pour la détection concurrente d'erreur sur une carte isolée.

En ce qui concerne le diagnostic au niveau module, pour toute carte ayant un circuit contrôleur global, le contenu de son registre GEI comportera l'indication de bon ou mauvais fonctionnement de la carte. D'autre part, si une approche purement cascadée est utilisée pour la carte, vu que tous les circuits suivant celui qui produit la première indication d'erreur vont aussi mémoriser l'occurrence d'une erreur, le GEI du dernier circuit fonctionnel de la cascade fournira l'état de fonctionnement dont on aura besoin pour identifier la carte défaillante du module.

En fait, la mise en cascade de circuits fonctionnels sera sans doute restreinte aux cartes, à cause surtout de fortes contraintes de temps qu'une telle approche imposerait au niveau du module. Pourtant, la mise en cascade d'indicateurs d'erreur provenant du circuit contrôleur global de chaque carte peut être considérée comme une approche très intéressante, compte tenu du bas coût engendré□une seule broche additionnelle pour le circuit contrôleur de la carte et un seul fil ajouté pour la connexion de la broche TDI de la carte à la nouvelle broche du circuit contrôleur (figure IV.13).

Toute approche basée sur la vérification parallèle impliquera l'augmentation des connecteurs des cartes pour pouvoir transmettre ses indicateurs d'erreur aux entrées d'un circuit contrôleur pour le module.

Le test hors-ligne étant basé sur la technique BS, chacun des aspects du test de la carte parcouru dans les paragraphes et les chapitres précédents est aussi valable au niveau du module. Confronté à la complexité de réalisation de ce type de test à ce niveau là, un partitionnement pour le module doit être envisagé. Ce partitionnement peut être basé sur l'utilisation de processeurs de test BS distribués (chapitre III), afin d'améliorer de manière significative le temps d'exécution du test de tous les circuits et toutes les connexions dans le module.

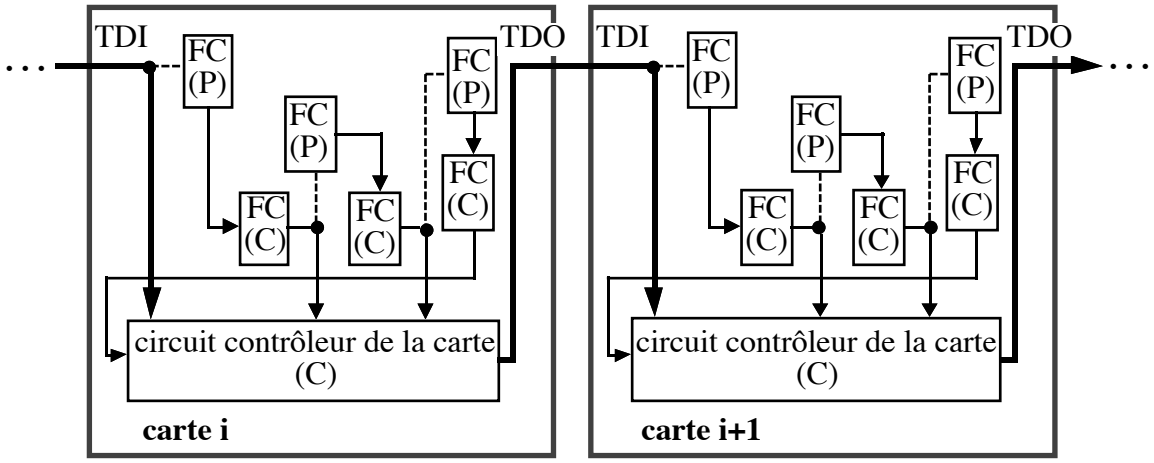


Figure IV.13. La mise en cascade d'indicateurs d'erreur des contrôleurs de cartes

IV.7 Conclusion

Dans ce chapitre, nous avons proposé une stratégie pour le test de cartes et modules nommée B²UBIST. Cette stratégie, basée sur la combinaison des techniques BS et UBIST, unifie les tests hors-ligne et en-ligne de circuits et connexions. B²UBIST est basée entre autres sur le partage du chemin de balayage série de la carte entre ces deux types de test.

D'une part, les caractéristiques de couverture de fautes et de gamme d'applications inhérentes à la technique UBIST sont préservées dans B²UBIST. D'autre part, le standard IEEE 1149.1 y joue un rôle majeur dotant la carte des moyens pour la réalisation du test hors-ligne des connexions, pour la commande d'exécution des phases de test hors-ligne et en-ligne et pour le diagnostic de fautes suivant une détection d'erreur quelconque.

Bien que la stratégie de test obtenue soit "fonctionnellement" compatible avec l'implémentation BS proposée par le standard IEEE 1149.1, quelques unes de ses règles ne sont pas vraiment respectées ici□d'abord, TDO sera active dans l'état "Run-Test/Idle" du contrôleur TAP et, ensuite, aucun registre à décalage ne sera placé entre TDI et TDO pendant l'exécution du test en-ligne. A ce sujet et concernant surtout le premier point de désaccord, l'utilisation de chemins de balayages parallèles qui partagent les connexions TDI/TDO sur les extrémités (figure IV.14, [MaT90]) ne sera pas possible, si la technique B²UBIST est adoptée. Ceci est dû au fait que tous les chemins de balayage doivent être activés en même temps pendant le test en-ligne, ce qui amène à un conflit évident entre les bus de test.

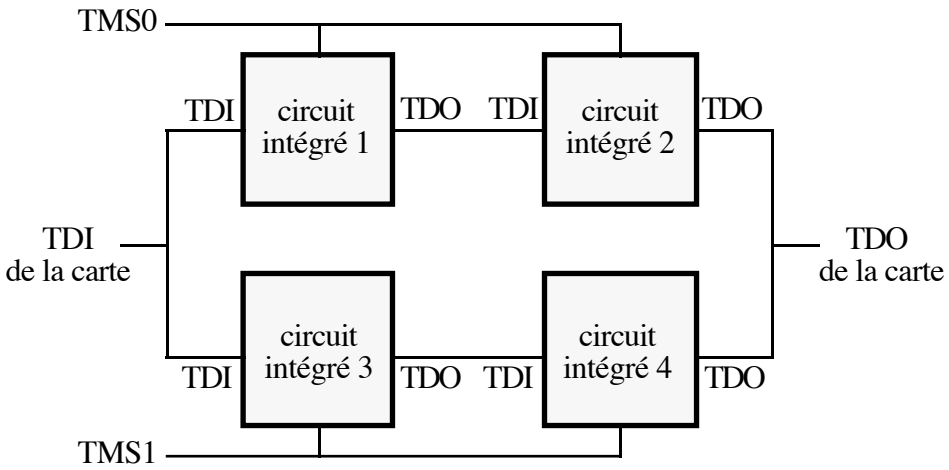


Figure IV.14 La configuration de chemins de balayage parallèles.

Finalement, le prix à payer pour l'intégration de la technique B²UBIST sera dû, d'abord, à la difficulté de conception de circuits UBIST et au manque d'outils automatiques pour réaliser cette tâche et, ensuite, au surcoût en surface engendré plutôt au niveau du silicium qu'au niveau des broches et des connexions additionnelles nécessaires aux cartes et modules. Quel que soit ce prix, il est clair que le coût engendré sera facile à justifier dans le cadre d'applications à haute sûreté de fonctionnement.

CHAPITRE V

LA CONCEPTION DE SYSTÈMES SÉCURITAIRES FIABLES

V.1 Introduction

L'utilisation de circuits intégrés dans des systèmes sécuritaires comme, par exemple, le contrôle de fonctions ferroviaires, spatiales, aéronautiques, automobiles critiques, etc, devient de plus en plus importante. De telles applications requièrent la détection concurrente d'erreurs, opération pouvant être assurée par des circuits "self-checking".

Compte tenu du fait que les fautes de fabrication se manifestent sous formes multiples, violant ainsi l'hypothèse de faute simple prise en considération dans la conception de circuits "self-checking", et que le cumul de fautes dû à leur latence peut amener à l'invalidation des propriétés "self-checking", la technique UBIST [Nic88], étudiée au chapitre I, propose l'ajout de ressources BIST aux architectures "self-checking" conventionnelles pour surmonter ces problèmes.

En dépit de l'utilisation de circuits conçus en vue des tests en-ligne et hors-ligne, la livraison de sorties erronées non détectées peut être provoquée par des courts-circuits, des fautes de type "stuck-on", ou encore des radiations spatiales, car ces phénomènes peuvent affecter en même temps les mécanismes de détection de fautes et la partie fonctionnelle du circuit. L'intégration de capteurs de courant dans une architecture UBIST peut surmonter ces problèmes, permettant ainsi la conception de circuits "self-checking" en technologie CMOS statique pour n'importe quel type d'application [LDN92].

Dans les circuits UBIST, l'activation des ressources BIST ne doit pas perturber l'opération normale du système dans lequel ils sont insérés. Afin d'empêcher que le test hors-ligne entraîne la perte de l'état du système, un schéma pour le BIST transparent [Nic92] peut être utilisé pour préserver les contenus des mémoires intégrées, permettant ainsi de récupérer l'état du système antérieur au test.

Des systèmes critiques réels doivent commander plusieurs actionneurs, délivrant des signaux qui sont soient corrects, soient sûrs. On a vu au chapitre I ce qui a été proposé dans [NNC89] concernant la possibilité d'implémenter des circuits intégrés "self-checking" avec des interfaces "fail-safe".

Pour la conception d'un système à haute sûreté de fonctionnement, les capacités "self-checking" de ses unités individuelles doivent être étendues du niveau du circuit au niveau de la carte, dans un premier temps, et au niveau du module ensuite. D'après ce qu'on a pu voir au chapitre précédent, la détection concurrente d'erreurs aux niveaux carte et module peut être assurée par la conjugaison des techniques "boundary scan" et UBIST à travers la conception B^2UBIST des circuits [LuC92].

Si l'on s'en tient à l'état de l'art, pour concevoir des systèmes qui assurent une haute sûreté de fonctionnement, on doit ajouter à la technique B^2UBIST l'intégration des capteurs de courant "self-checking", du BIST transparent de mémoires et des interfaces "fail-safe".

Pourtant, même pour l'exécution de missions de courte durée, la redondance de matériel employée pour améliorer la sécurité du système simplex contribue aussi à diminuer la fiabilité du système "self-checking" résultant. Les systèmes "self-checking" sont alors compétitifs par rapport à leur sécurité mais certainement pas par rapport à leur fiabilité.

Une solution pour surmonter ce problème réside dans une technique de tolérance aux fautes, telle qu'on assure la compétitivité de ces systèmes en termes de fiabilité en même temps qu'on préserve la sécurité inhérente des différentes parties "self-checking". Dans ce chapitre nous proposons un schéma qui atteint cet objectif [LuC93].

Le chapitre est organisé comme suit: le prochain paragraphe est consacré aux schémas matériels pour la tolérance aux fautes; ensuite, on présente un schéma tolérant les fautes basé sur deux répliques d'un système "self-checking" et sur une interface "fail-safe"; puis la sécurité de ce système est analysée, et une interface intégrant l'autotest et tolérant ses propres fautes est proposée; finalement, avant d'arriver aux conclusions de ce chapitre, un paragraphe est dédié à l'évaluation de la fiabilité du schéma proposé.

V.2 Les systèmes tolérant les fautes

Un système tolérant les fautes est construit en utilisant des techniques capables d'assurer qu'il réalise la mission pour laquelle il a été spécifié même quand une faute affecte le système. Dans ce type de système, le masquage de fautes est inconditionnellement assuré par l'intermédiaire de la redondance.

Bien que la duplication soit une technique très utile à la conception de systèmes sécuritaires, les implémentations dupliquées qui n'intègrent pas de capacités pour le diagnostic de fautes ne peuvent pas assurer la continuité de fonctionnement du système en présence d'une erreur: même s'il est toujours possible de détecter une erreur qui se manifeste comme une différence entre les résultats venant de deux modules identiques, il sera impossible d'identifier quel est le module défaillant. Probablement à cause de ce problème, les concepteurs de systèmes à haute fiabilité ont dû envisager l'utilisation de modules avec des mécanismes intégrés pour la détection de fautes, ou bien de systèmes basés sur des degrés plus élevés de redondance. Les circuits correcteurs d'erreurs illustrent le premier cas tandis que les structures N-redondantes illustrent la deuxième solution [BrF76].

L'utilisation d'un code rend possible la conception de matériel capable de détecter et/ou corriger automatiquement les erreurs produites dans un circuit. Bien que normalement les fautes du matériel correcteur ne soient pas toutes couvertes, l'utilisation de codes correcteurs d'erreurs peut masquer les sorties fonctionnelles erronées et ainsi, tolérer une grosse partie des fautes du circuit [BrF76]. Des codes correcteurs d'erreurs simples ont été largement utilisés dans des mémoires fiables, premièrement parce que la redondance engendrée est de l'ordre de 10 à 40% et, deuxièmement, parce que le délai imposé par le matériel correcteur est relativement court. Pourtant, pour des degrés plus élevés de fiabilité et d'autres circuits fonctionnels, les codes correcteurs d'erreurs ne peuvent se montrer économiquement viables que dans des cas spéciaux, car une redondance et une dégradation de performance importantes accompagnent normalement la correction d'erreurs multiples [SiS82].

La structure 3-redondante ("Triplicated Modular Redundant"□TMR) est sans doute la plus utilisée pour la tolérance aux fautes. Elle est composée de trois répliques du module simplex et d'un mécanisme de vote qui réalise la fonction de majorité M(x,y,z)=xy+yz+xz (figure V.1).

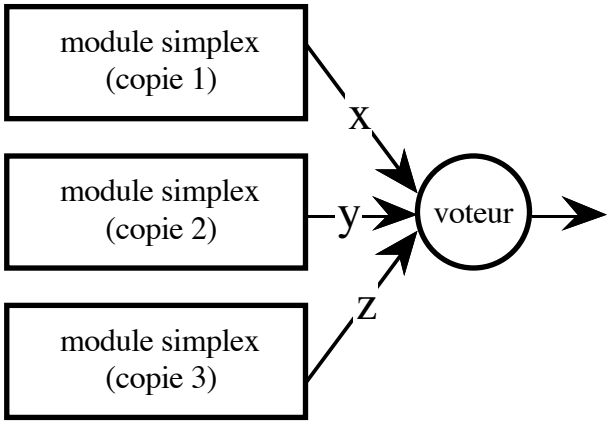


Figure V.1. La structure TMR.

D'autres structures tolérant les fautes existantes sont basées sur des degrés de redondance plus élevés que trois. Les structures redondantes hybrides, par exemple, sont composées d'un ensemble de trois modules actifs et quelques modules de réserve. Le schéma redondant bi-duplex, à son tour, est basé sur la mise en parallèle de deux structures duplex et, de la même façon que dans le cas de redondance hybride, sur un mécanisme de détection de fautes capable de reconfigurer le système de telle sorte que les sorties du module défaillant ne participent plus au vote [Cou76].

Deux caractéristiques majeures reflètent la qualité de systèmes tolérant les fautes, la sécurité et la fiabilité [LCG89]□

Définition V.1 Sécurité

La sécurité mesure la probabilité que le système n'ait pas eu jusqu'à l'instant t, de défaillance catastrophique. Ceci suppose que les défaillances dites dangereuses puissent être identifiées par rapport à la mission du système.

Définition V.2 Fiabilité

La fiabilité mesure la probabilité que le système ait fonctionné jusqu'à l'instant t. C'est l'évaluation du comportement d'un système jusqu'à la défaillance (c'est à dire jusqu'à ce que le service délivré dévie du service spécifié) dans des conditions d'environnement spécifiées.

Des études préalables sur la sécurité et la fiabilité de structures reconnues pour la tolérance aux fautes ont montré que ces deux caractéristiques sont exactement opposées [Cou76, ArL85].

V.3 Une proposition de système "self-checking" tolérant les fautes

Comme il a pu être vu dans le chapitre I, les systèmes "self-checking" fournissent des indicateurs d'erreur afin de signaler la nécessité d'ignorer les sorties de leurs parties fonctionnelles. Puisqu'un module "self-checking" défaillant peut être identifié par l'état de son indicateur d'erreur, la capacité à tolérer les fautes pourrait être assurée à travers un système composé d'une copie du module simplex et d'une copie de sa version "self-checking". Pour un module "self-checking" sans faute, la détection d'une erreur sur le module simplex serait donnée par une disparité entre les sorties fonctionnelles des deux modules. Etant donné qu'il est probable que le module "self-checking" tombe en panne avant le module simplex et que, après la défaillance du module "self-checking", la détection d'erreur du module simplex ne pourra plus être assurée, la sécurité pour le système global serait assez faible (à peu près égal à la fiabilité du système). Notre but étant aussi de préserver la sécurité du module "self-checking" dans le système total, nous proposons un schéma tolérant les fautes qui est constitué de deux copies d'un module "self-checking" et d'une interface pour le masquage des sorties du module ou du système défaillant (figure V.2).

La duplication de fonctions "self-checking" en vue de l'amélioration de la fiabilité de systèmes informatiques, a été proposée pour la première fois dans [ACL78]. Dans ce travail, les auteurs présentent un schéma basé sur la mise en parallèle de deux micro-ordinateurs à haute sûreté de fonctionnement (l'un actif et l'autre en attente), et sur la sélection des sorties via un réseau d'interrupteurs. Les problèmes d'interfaçage de deux repliques "self-checking", brièvement discutés dans [ACL78], sont l'objet d'une étude approfondie dans ce chapitre.

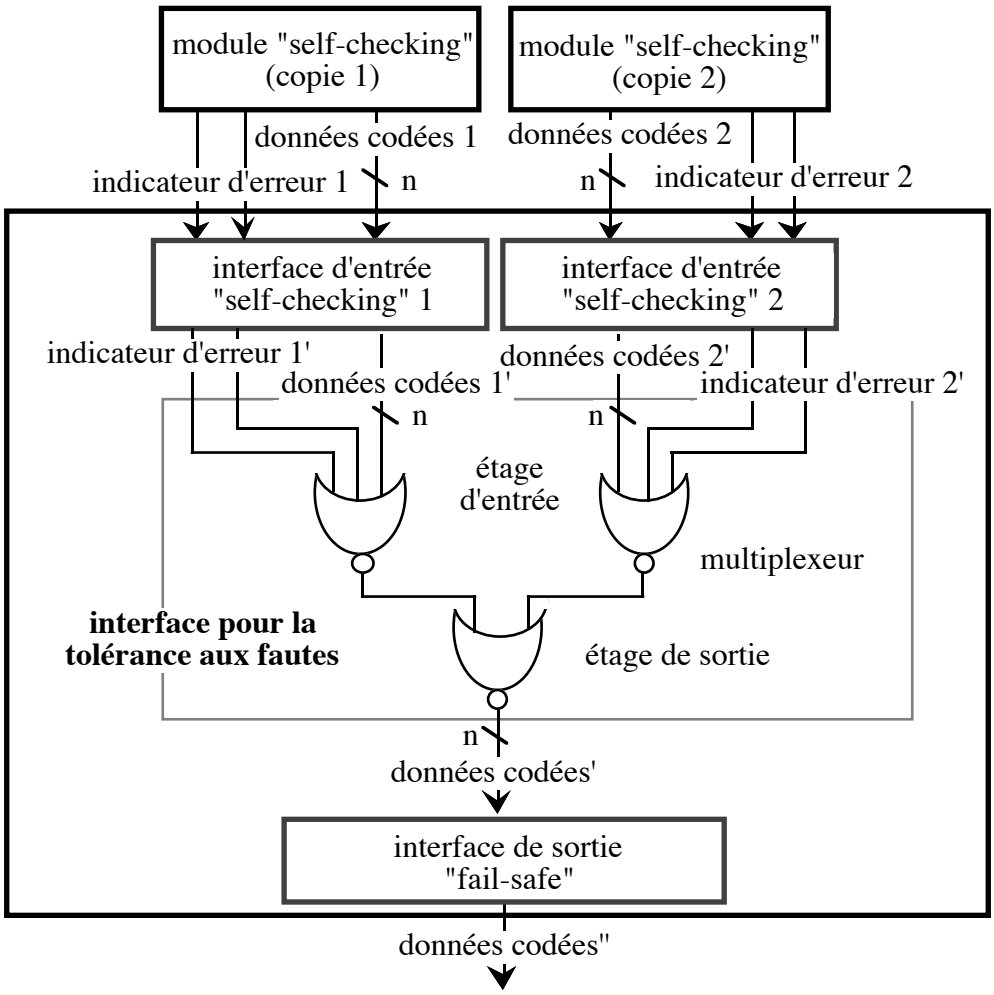


Figure V.2. Système "self-checking" tolérant les fautes.

Dans le système de la figure V.2, une faute affectant un module sera signalée par son indicateur d'erreur et une faute sur les connexions qui partent des modules vers l'interface pour la tolérance aux fautes sera observée à travers la vérification de l'état de codage du mot reçu (figure V.3). Quand une faute affecte un module et/ou les connexions de ses sorties primaires, une erreur sera indiquée par un contrôleur "double-rail" (figure V.3) et l'interface "self-checking" associée transposera les données erronées en un mot qui sera traité et masqué par le multiplexeur (figure V.2). La capacité de mémorisation d'erreur (chapitre IV) doit être assurée afin que les sorties d'un module défaillant soient arrêtées à un état sûr (figure V.3).

Le multiplexeur est censé réaliser un ET logique des sorties venant de deux copies du module "self-checking"□pour un système sans faute chaque indicateur d'erreur' sera '00' et un ET de deux mots identiques sera accompli; pour un module fautif, ses sorties seront transposées en un mot '1…1' (car son indicateur d'erreur' sera '01', '10' ou '11') et le mot venant du module sans faute atteindra la sortie du multiplexeur; pour deux modules défaillant le mot '1…1' sera pourvu par le multiplexeur. Dans le dernier cas, on s'attend à que le

multiplexeur se comporte comme un contrôleur chargé de signaler le fait que les fautes ne peuvent plus être tolérées par le système.

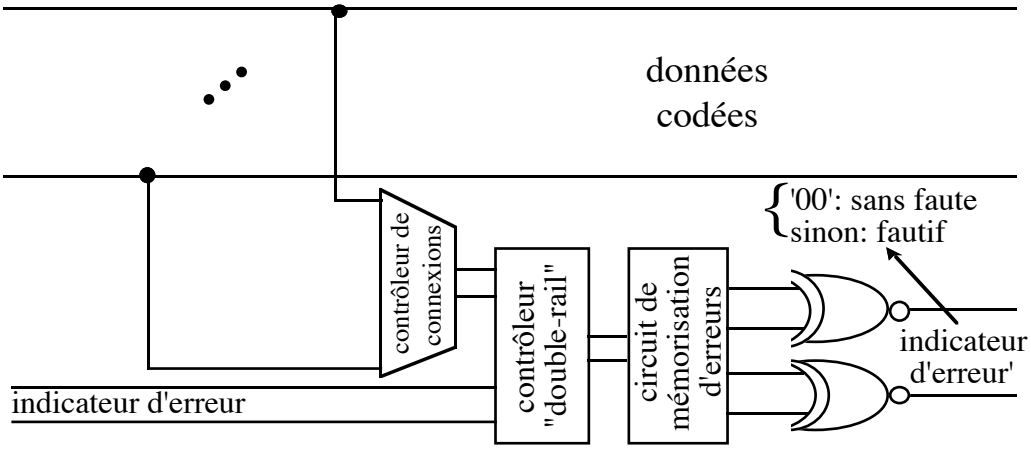


Figure V.3. L'interface d'entrée "self-checking".

Lemme V.1 Le multiplexeur sera "code-disjoint" (définition I.5), si et seulement si '1…1' est un mot hors-code.

Preuve Pour un multiplexeur sans faute et un mot d'entrée appartenant au code (c'est-à-dire que, au moins, l'un des indicateurs d'erreur' est à '00'), le multiplexeur sortira le mot venant du module sans faute (dont l'indicateur d'erreur' sera égal à '00), qui est un mot du code.

Pour un multiplexeur sans faute et un mot d'entrée hors-code (où tout indicateur d'erreur assume soit la valeur '01', soit '10', ou bien '11'), le multiplexeur sortira '1…1', qui est un mot hors-code. □

L'ensemble données-code sorti par le multiplexeur sera vérifié par l'intermédiaire de l'interface "fail-safe" de sortie (figure V.4). Cette interface est basée sur le codage en fréquence des sorties dont l'état sûr correspond à l'absence d'une fréquence predéfinie (chapitre I). Cet état sera passé aux actionneurs au moment où le système deviendra incapable d'assurer qu'une sortie correcte soit produite. Pour plus de détails concernant la conception de cette interface codée en fréquence, le lecteur est invité à se reporter au chapitre I et à [NNC89].

D'une part, notre système tolérant les fautes peut être vu comme une amélioration apportée à la structure redondante bi-duplex, car les modules "self-checking" complexes normalement ne sont pas basés sur la duplication de leurs contreparties simplex. D'autre part, cette structure est reformulée ici et ses principes sont mis en conformité avec les théories "self-checking" et "fail-safe" existantes.

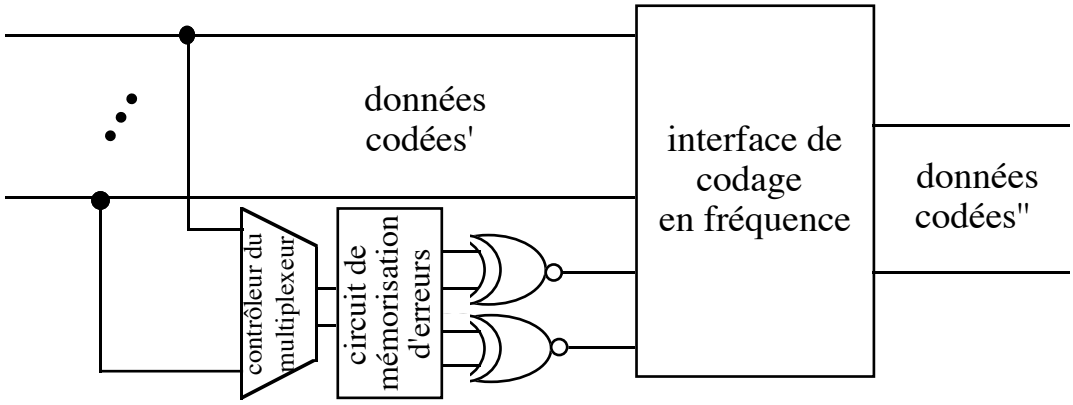


Figure V.4. L'interface de sortie "fail-safe"

Un autre point important de notre schéma est que, contrairement aux structures tolérant les fautes précédemment proposées, il est implémenté à un niveau très bas d'abstraction, car l'état de l'art est tel que l'on sait à présent comment concevoir des circuits "self-checking" et "fail-safe" en tenant compte des mécanismes physiques de défaillances (chapitre I) [Cou81, NiC85, Nic87, NNC89,…]. Ceci fait que l'on atteint une couverture de fautes réelles sans doute très élevée.

V.3.1 Analyse de la sécurité

Puisque l'interface pour la tolérance aux fautes (figure V.2) est aussi susceptible de générer des fautes, elle doit être conçue pour être "fail-safe" (chapitre I) et de cette façon assurer la sécurité du système dans son ensemble.

Lemme V.2 Le multiplexeur sera "fault-secure" (définition I.2), si et seulement si un code non-ordonné (définition I.12) est utilisé.

Preuve Le multiplexeur sans faute réalise la fonction sortie=(entrée1•entrée2). Pour que le multiplexeur soit "fault-secure" quand il est affecté par une faute simple, on s'attend à ce que

- sortie∈espace-de-codage [illegible] ulement si
  (sortie=entrée1 [illegible] reur-1'='00')
  (sortie=entrée2 [illegible] reur-2'='00');
- sinon, sortie∉espace-de-codage.

Pour n'importe quelle faute simple affectant son étage de sortie (figure V.2) le multiplexeur délivrera la sortie correcte ou effectuera la fonction spécifiée sauf pour un bit. Alors, un bit de parité (définition I.13) pourrait être utilisé pour coder

entrée1, entrée2 et sortie. Pourtant, une faute simple affectant l'un des indicateurs d'erreur de l'étage d'entrée peut permettre qu'un mot erroné, bien qu'appartenant au code, atteigne l'étage de sortie du multiplexeur. Supposons que l'étage d'entrée du multiplexeur est affecté par une telle faute et que entrée1 est le mot correcte, tandis que entrée2 est le mot erroné (appartenant ou non à l'espace de codage). De cette façon, deux cas sont possibles pour un multiplexeur "fault-secure"□

(a) si entrée2 couvre entrée1, alors sortie=entrée1; ou,

(b) si entrée2 ne couvre pas entrée1, alors sortie$\notin$espace-de-codage.

Dans les deux cas, "sortie" sera couverte par entrée1 et entrée2. Etant donné que entrée1 appartient au code, on peut conclure qu'on a besoin d'un espace de codage où aucun mot du code ne couvre un mot différent appartenant aussi au code, afin d'éviter que, dans le cas (b), sortie$\in$espace-de-codage. Un code non-ordonné peut satisfaire cette contrainte tout en couvrant, en même temps, le cas d'une faute simple sur le stage de sortie. □

Corollaire V.1 : Le multiplexeur sera à "code-disjoint", si un code non-ordonné est utilisé.

Preuve□ Pour un code non-ordonné, le mot '1…1' sera un mot hors-code, car il couvre tous les autres mots (appartenant ou non au code). Ainsi, selon le lemme V.1, la propriété "code-disjoint" sera satisfaite par le multiplexeur sans faute. □

Théorème V.1□ L'interface pour la tolérance aux fautes est "fail-safe" pour un ensemble quelconque de fautes simples affectant simultanément l'un des modules "self-checking", l'une des interfaces d'entrée "self-checking", le multiplexeur et l'interface "fail-safe" de sortie.

Preuve□ Considérons le partitionnement du système donné dans la figure V.2 en quatre blocs principaux tel que montré dans la figure V.5.

Si un circuit "self-checking" est conçu de telle sorte que l'occurrence erronée de mots hors-code ne provoque jamais de situations dangereuses, alors les mots hors-code peuvent être classés comme des mots sûrs et les définitions I.2 ("fault-secure") et I.17 ("fail-safe") se superposent pour une telle classe de circuits. Ainsi chaque module "self-checking", chaque interface "self-checking", le multiplexeur ("fault-secure" d'après le lemme V.2) et l'interface de sortie (composée d'une partie "self-checking" et d'une partie codée en fréquence comme on a pu le voir au chapitre I) sont "fail-safe" selon la définition I.17.

Si un circuit est conçu par duplication d'un sous-circuit qui est "fail-safe" par la définition I.17, alors le circuit résultant sera "fail-safe" par la définition I.18. Ainsi,

l'ensemble de deux modules "self-checking" et l'ensemble de deux interfaces d'entrée "self-checking" sont "fail-safe" d'après la définition I.18.

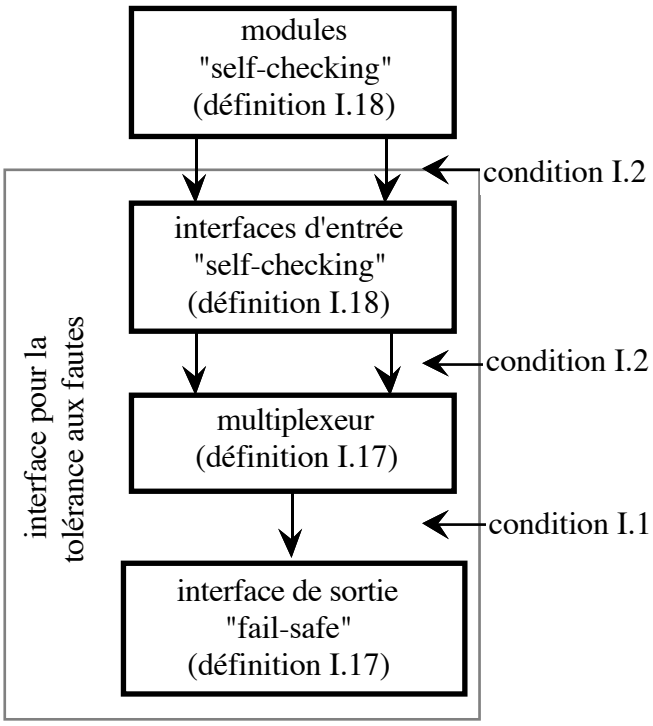


Figure V.5. Le partitionnement du système "self-checking" tolérant les fautes

Pour toute faute simple détectable affectant un circuit qui est devenu "fail-safe" selon la définition I.18 par duplication, un seul vecteur erroné e^i (i=1 ou 2) en résultera. Alors, sous l'hypothèse I.3□

(1) la condition I.2 sera satisfaite pour la connexion□(a) des modules "self-checking" aux interfaces d'entrée "self-checking", en vertu du fait que le comportement erroné d'un module sera transposé en un état sûr par le circuit de mémorisation d'erreur; (b) des interfaces d'entrée "self-checking" au multiplexeur, car pour le comportement erroné d'une branche de l'interface pour la tolérance aux fautes le multiplexeur enverra soit le mot correcte, soit un mot hors-code, à l'interface de sortie "fail-safe"; et,

(2) la condition I.1 sera satisfaite pour la connexion du multiplexeur à l'interface de sortie "fail-safe", car l'occurrence d'un mot hors-code sera mémorisée par la partie "self-checking" de l'interface de sortie, étant ainsi possible d'arrêter l'interface de codage en fréquence dans un état sûr (chapitre I).

Puisque tous les blocs qui composent notre système sont "fail-safe" et sont connectés selon les conditions I.1 et I.2, la totalité de l'interface pour la tolérance aux fautes est "fail-safe". Finalement, d'après le théorème I.1, l'interface pour la

tolérance aux fautes sera "fail-safe" pour toute collection de fautes simples affectant en même temps chacun des blocs donnés figure V.5. □

V.3.2 L'interface "strongly fail-safe"

Puisque dans un système G "fail-safe" et tolérant les fautes, le nombre de fautes f∈F, qui produisent G(a,∅) au lieu de G(a,f)≠G(a,∅) | G(a,f)∈Os n'est pas négligeable, même pour le cumul de fautes pendant sa vie utile le système doit assurer qu'un G(a,f)∉Os ne sera jamais produit avant qu'un G(a,f)∈Os ne soit délivré.

Etant donné que la propriété "self-testing" (définition I.19) ne peut pas être assurée par l'interface pour la tolérance aux fautes et que, le cumul de fautes peut amener à la perte de sa propriété "fail-safe", l'interface n'atteindra pas le but du "totally fail-safe" (définition I.20). D'autre part, des capacités BIST activées périodiquement peuvent assurer ces propriétés. Pour illustrer cela on peut voir que la propriété "self-testing" n'est pas assurée par l'étage d'entrée du multiplexeur, parce que seuls des mots appartenant au code (où l'indicateur-d'erreur'='00') sont appliqués pendant l'opération normale du système. Suite à une défaillance d'un module, la propriété "fail-safe" de l'interface d'entrée associée au module défaillant peut être perdue en raison du cumul de fautes sur son circuit de mémorisation d'erreur. Supposons que, sous ces conditions, seuls des mots couvrant les sorties du module sans faute soient délivrés au multiplexeur jusqu'à ce que ce dernier tombe en panne. A ce moment là, il est possible que le module qui était précédemment tombé en panne délivre au multiplexeur un mot incorrecte appartenant au code et ce mot sera transmis à l'interface de sortie. De cette manière, le système dans sa totalité ne sera pas "strongly fail-safe" (définition I.21), à moins qu'une phase de test soit insérée dans l'opération du système afin d'assurer la propriété "self-testing" au multiplexeur, dans un premier temps, et, ensuite, de vérifier si la branche défaillante de l'interface est toujours "fail-safe". Finalement, selon l'hypothèse I.3, ces vecteurs devront être appliqués périodiquement au circuit sous test.

L'application de tout mot du code d'entrée est normalement supposé pendant l'opération normale des systèmes "self-checking". Pourtant, pour que cela soit vraiment vérifié dans des systèmes complexes, des contraintes importantes au niveau logiciel seront imposées aux applications dans lesquelles ils sont utilisés. Encore une fois, l'activation périodique d'une phase de test pour le système ferait lever cette contrainte, assurant ainsi l'hypothèse I.3. Ceci mis à part, après l'occurrence d'une erreur pendant l'opération normale, le bloc censé être fautif pourrait être testé hors-ligne afin de vérifier s'il s'agit d'une faute transitoire, ou bien d'une faute permanente. Dans le cas d'une faute transitoire, le bloc pourrait redémarrer et reprendre les activités qu'il exécutait avant que l'erreur soit survenue.

Pour aller plus loin dans cette analyse, d'abord, définissons comme active une branche du multiplexeur dont l'indicateur-d'erreur'='00', et comme inactive une branche dont l'indicateur-d'erreur'≠'00'. Ensuite,

Théorème V.2 L'interface pour la tolérance aux fautes atteindra le but "totally fail-safe", si

(a) les branches actives du multiplexeur sont des contrôleurs "totally self-checking" (définition I.16);

(b) la branche inactive du multiplexeur est "fail-safe";

(c) la mémorisation d'erreur des interfaces d'entrée "self-checking" sont "fail-safe"; et,

(d) le matériel restant atteint le but du "totally self-checking".

Preuve Pour que le multiplexeur sans faute soit capable de signaler qu'aucun module n'est disponible, il doit être "code-disjoint". Pour que le multiplexeur soit "fail-safe" et satisfasse les conditions de connexion I.1 et I.2, il doit être "fault-secure".

D'une part, si des fautes non-détectées sont cumulées sur les branches actives du multiplexeur, la propriété "fault-secure" ne peut plus être assurée. Pourtant, si la propriété "self-testing" est assurée, les branches actives du multiplexeur se comporteront comme des contrôleurs "totally self-checking", les fautes ne se cumuleront pas et la clause (a) de la définition I.21 ("strongly fail-safe") sera satisfaite.

D'autre part, il est permis aux fautes de se cumuler sur la branche inactive du multiplexeur, à moins qu'elles interfèrent dans le fonctionnement correcte de la branche active. Alors, il est suffisant que la clause (b) de la définition I.21 ("strongly fail-safe") soit satisfaite.

Si les circuits de mémorisation d'erreur des interfaces d'entrées "self-checking" sont encore capables de mémoriser l'occurrence d'une erreur sur la branche correspondante de l'interface, alors rien ne change s'ils sont affectés par un certain nombre de fautes. Etant donné ceci, seule la propriété "fail-safe" doit être assurée et la clause (b) de la définition I.21 ("strongly fail-safe") s'applique à ce cas-là.

Pour tout le reste du matériel qui constitue l'interface pour la tolérance aux fautes, la première occurrence d'erreur doit être signalée par la circuiterie de mémorisation d'erreur associée, afin d'isoler la branche du module fautif ou de transposer les sorties du système fautif en un état sûr. De cette façon, le fait d'atteindre le but du "totally self-checking" va permettre de satisfaire la condition (a) de la propriété "strongly fail-safe". □

Dans la suite on discute le schéma de test hors-ligne représenté dans la figure V.6 et composé de cinq phases (T0, T1, T2, T3 et T4), ce qui est suffisant pour tester l'interface pour la tolérance aux fautes□

Un générateur de vecteurs de test capable de générer tous les mots appartenant au code non-ordonné utilisé [Nic88] est ajouté à l'architecture de l'interface (figure V.6) et sera activé pendant les phases T1, T2 et T4.

L'indicateur d'erreur du contrôleur du multiplexeur sera rebouclé sur les interfaces d'entrée "self-checking" pendant les phases T1 et T2. Pendant T0, T1, T2 et T3, la mémorisation d'erreur de l'interface de sortie "fail-safe" sera isolée des sorties du contrôleur du multiplexeur.

La circuiterie de mémorisation d'erreur (figure V.7-a) doit aussi intégrer des capacités pour son autotest. Dû à l'insertion de la phase de test hors-ligne, l'étage d'entrée du multiplexeur peut être simplifié en remplaçant les portes NON-OU à 3-entrées (figure V.2) par des portes NON-OU à 2-entrées. Les bascules utilisées pour la mémorisation en-ligne d'erreur seront partagées avec un générateur de vecteurs de test basé sur un compteur qui sera utilisé pour le test hors-ligne. Deux autres bascules composeront également ce générateur de vecteurs de test (figure V.7-b). Les portes restantes seront utilisées pour prédire si l'état suivant du comptage correspond à un mot codé "double-rail" (définition I.15), et pour imposer au compteur le saut de ces mots. La valeur initiale du compteur sera l'état présent des bascules de l'indicateur d'erreur et '00' pour les bascules ajoutées. Tous les 56 mots hors-code (2^6 mots possibles - 2^3 mots du code) seront générés et après 56 cycles d'horloge la mémorisation d'erreur sera remise à l'état où elle se trouvait avant que le test commence. De cette façon, du BIST transparent sera appliqué et l'état de l'interface pour la tolérance aux fautes ne sera pas perdu pendant la phase de test.

Finalement, un registre UBILBO (chapitre I) est connecté aux entrées de l'interface de sortie "fail-safe" (figure V.6). Ce registre sera utilisé comme générateur de vecteurs de test pendant la phase T0 et comme analyseur de signature pendant les phases T3 et T4.

Dans la phase T0, les contrôleurs "double-rail" des interfaces d'entrée "self-checking" (figure V.3) seront testés, par exemple, par l'intermédiaire du générateur proposé dans [Nic88]. La détection d'une faute sera signalée par la mémorisation d'erreur de ces interfaces. Dans cette même phase, l'interface de codage en fréquence et la circuiterie de mémorisation d'erreur associée recevront une séquence de test appliquée par l'UBILBO (figure V.6) et les bascules montrées dans la figure V.7-a.

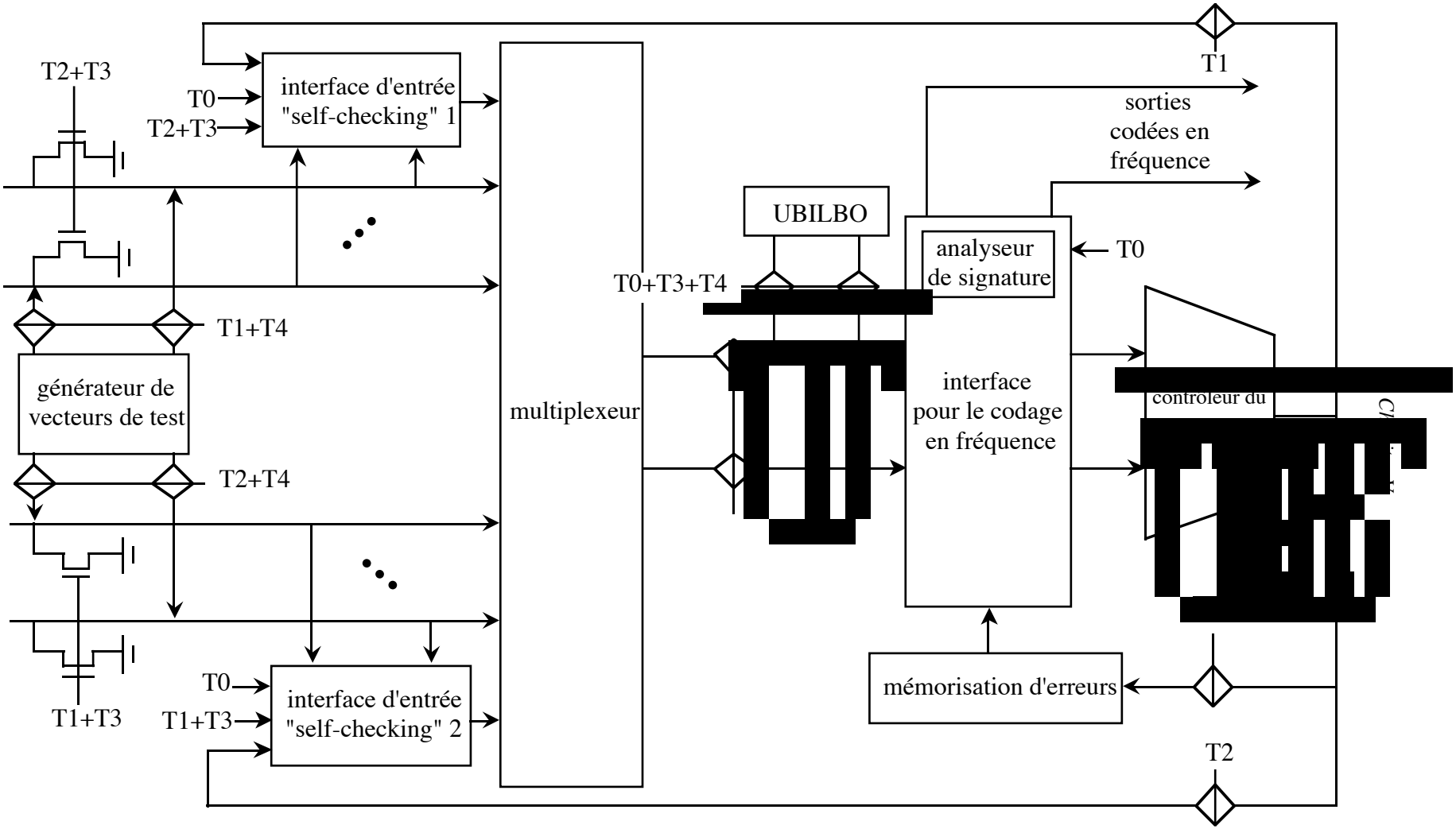


Figure V.6. Les ressources BIST de l'interface.

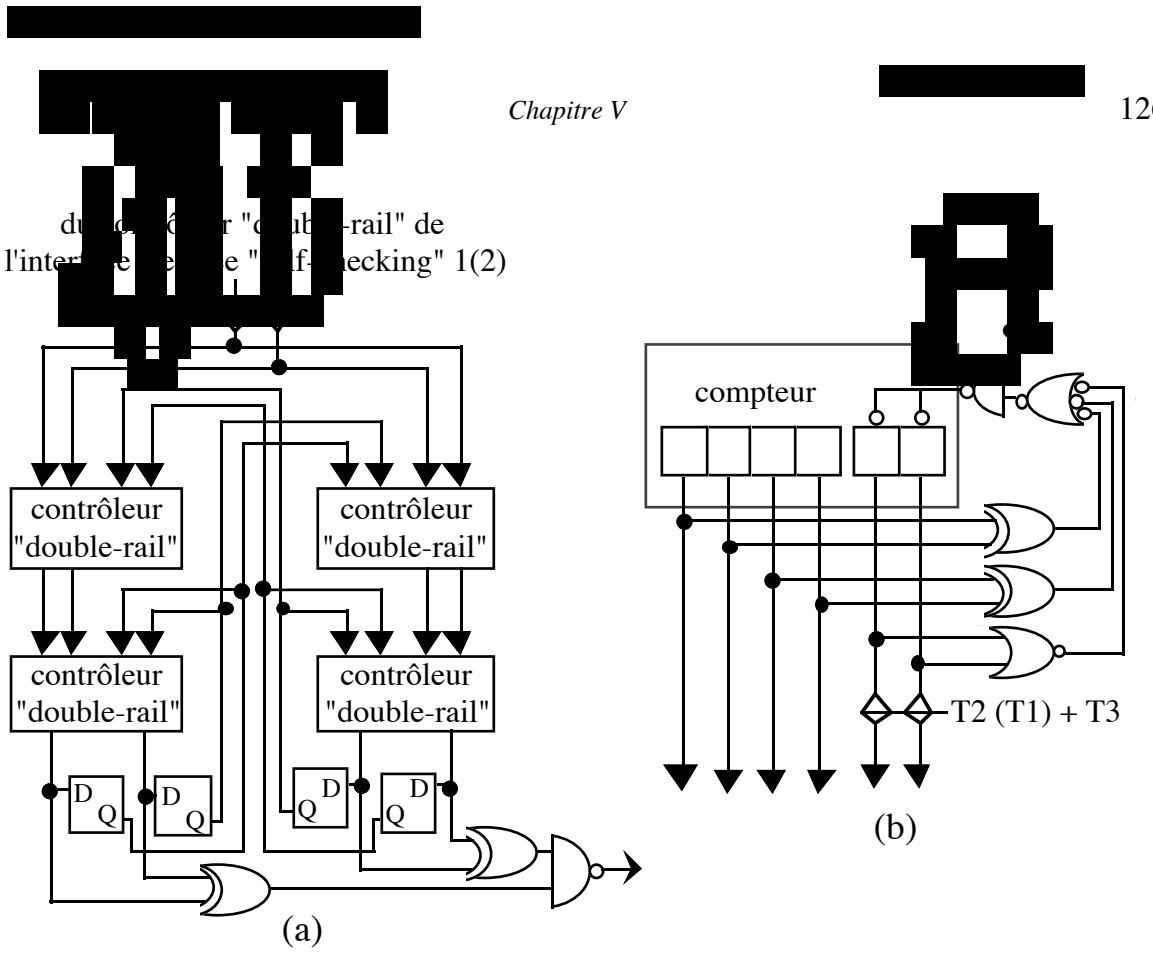


Figure V.7. Les ressources BIST de la circuiterie de mémorisation d'erreur.

Ce générateur de vecteurs de test doit être capable de générer les séquences présentées dans [NNC89] et le paragraphe IV.3.1, et de remettre les bascules de mémorisation d'erreur à l'état sans faute après l'exécution de T0. Les réponses de test seront compactées par l'analyseur de signature de l'interface de codage en fréquence (figure V.6).

Dans la phase T1, la branche d'entrée du multiplexeur correspondant à l'interface "self-checking" 1 sera vérifiée par l'intermédiaire du générateur de vecteurs de test de la figure V.6. Cette même phase va assurer le test concurrent du contrôleur de connexions (figure V.3) de l'interface 1. L'autre branche d'entrée du multiplexeur recevra le mot hors-code '0…0' (couvert par tous les mots appartenant au code et hors-code), mais l'application simultanée de mots hors-code au circuit de mémorisation d'erreur de l'interface 2 (d'après le schéma donné dans la figure V.7) est censée masquer ce mot. La détection d'une faute sera signalée à travers la circuiterie de mémorisation d'erreur de l'interface d'entrée "self-checking" 1 (figure V.6). Finalement, T1 doit durer suffisamment pour assurer l'application de tous les mots du code et, en même temps, un multiple de 56 cycles d'horloge doit être prévu pour assurer que le circuit de mémorisation d'erreur de l'interface 2 termine cette phase au même état où il se trouvait avant le début de T1.

En T2, la même procédure de test sera appliquée à l'interface 2 (figure V.6) et la mémorisation d'erreur de l'interface 1 (figure V.7) produira les mots hors-code nécessaires au test des entrées du multiplexeur.

La phase T3 vise à rattraper une éventuelle perte de capacité de mémorisation d'erreur qui aurait échappée aux phases de test précédentes. Ceci peut être dû, par exemple, à l'activation d'une interface avec une branche "fail-safe" inactive et une branche active affectée par une faute non-détectée qui corrompt sa mémorisation d'erreur. L'UBILBO sera initialisé et fonctionnera comme analyseur de signature dans cette phase. Les circuits de mémorisation d'erreur des deux interfaces d'entrée seront vérifiés en parallèle selon le schéma de la figure V.7. Le mot hors-code '0…0' sera appliqué à tout l'étage d'entrée du multiplexeur (figure V.6) et pendant 56 cycles d'horloge le mot '1…1' sera le seul permis aux entrées de l'UBILBO.

Au démarrage de la phase T4, les bascules de mémorisation d'erreur de l'interface de sortie "fail-safe" contiendront un mot codé "double-rail" (chargé en T0) et l'UBILBO restera dans le mode d'analyse de signature. Tous les mots appartenant au code seront appliqués aux entrées du multiplexeur pendant la phase T4 (figure V.6). Si au moins l'une des branches d'entrée de l'interface est toujours en marche, ces mots seront appliqués encore une fois au contrôleur du multiplexeur. Alors l'UBILBO et la circuiterie de mémorisation d'erreur de l'interface de sortie "fail-safe" compacteront les réponses de test. En effet, dans ce cas-là, l'utilisation d'un analyseur de signature ne semble pas nécessaire. Pourtant, si aucune branche d'entrée n'est disponible, ceci peut être provoqué par une faute sur le contrôleur du multiplexeur propagée aux interfaces d'entrée dans les phases T1 et/ou T2. Puisque leurs indicateurs d'erreur sont censés être arrêtés dans un état sûr, les mots parvenant du générateur de vecteurs de test n'atteindront pas le contrôleur du multiplexeur et il est possible que le mot '1…1' ne manifeste pas la faute aux entrées du circuit final de mémorisation d'erreur. Ce fait résultera dans une signature erronée calculée par l'analyseur intégré.

Finalement, si une faute dangereuse est détectée hors-ligne, l'analyseur de signature, l'UBILBO et/ou la mémorisation d'erreur de l'interface de sortie "fail-safe" (figure V.6) va signaler ce fait. Puisqu'étant affectée par une telle faute, l'interface ne sera plus capable d'assurer un état de sortie sûr si une faute parvient, la connexion du générateur de fréquence (F∅ dans la figure I.8) avec les actionneurs doit être définitivement coupée. Afin d'assurer ceci, la circuiterie "strongly fail-safe" de coupure d'alimentation proposée en [NNC89] et étudiée au chapitre I peut être utilisée pour vérifier les contenus des registres mentionnés ci-dessus et la signature résultante de sa propre phase de test hors-ligne. A son tour, la mémorisation d'erreur de cette circuiterie de coupure de courant doit être testée comme décrit paragraphe IV.3.1.

L'exécution des phases de test hors-ligne décrites plus haut impliquera, en principe, la livraison de mots incorrects (appartenant ou non au code) aux actionneurs commandés par l'interface de codage en fréquence (figures V.4 et V.6) et par la circuiterie "strongly fail-safe" de coupure d'alimentation (chapitre I). Etant donné que normalement les systèmes électromécaniques sont sensibles à des signaux d'une durée d'au moins plusieurs millisecondes, si le test ne dure que quelques microsecondes un état non-sûr ne sera pas produit. Si le test hors-ligne dure suffisamment pour provoquer une situation dangereuse, l'interface de sortie "fail-safe" peut être légèrement modifiée afin d'imposer un état sûr (un indicateur-d'erreur='01', '10' ou '11' dans la figure V.4) pendant l'exécution des phases T1, T2, T3 et T4. De cette façon, seul le test de l'interface de codage en fréquence (pendant T0) produira des mots non-sûrs, mais sa durée sera très courte comme discuté au chapitre I.

Théorème V.3 Le schéma proposé pour le test hors-ligne assure les conditions données au théorème V.2.

Preuve La condition (a) est assurée par les phases T1 et T2, car les branches actives du multiplexeur seront testées séparément à travers l'application de tous les mots appartenant au code. Si une faute est détectée sur une branche, l'occurrence d'erreur sera mémorisée dans l'interface d'entrée "self-checking" associée et la branche sera desactivée.

La condition (b) est aussi assurée par les phases T1 et T2. Si une branche inactive ne peut pas assurer sa propriété "fail-safe", ce fait sera signalé par une sortie hors-code du multiplexeur pendant le test de la branche active. Ceci résultera, de la même façon que dans le cas précédent, dans la desactivation de la branche sous test du multiplexeur.

La condition (c) est assurée par la phase T3, car tous les mots hors-code sont appliqués aux circuits de mémorisation d'erreur des interfaces d'entrée "self-checking", vérifiant ainsi la totalité de leur capacité de mémorisation d'occurrences d'erreur.

La condition (d) est assurée pour les contrôleurs "double-rail" et l'interface de codage en fréquence par la phase T0, pour les contrôleurs de connexions par les phases T1 et T2, et pour le contrôleur du multiplexeur par la phase T4. □

Nous avons mentionné précédemment que l'insertion d'une phase de test dans l'opération normale du système pourrait être utile pour distinguer une faute permanente d'une faute transitoire. Examinons donc le cas de détection d'une faute transitoire affectant notre système tolérant les fautes.

Une telle faute peut provoquer la mémorisation d'une erreur dans les interfaces d'entrée "self-checking", dans l'interface de sortie "fail-safe" et/ou dans la circuiterie "strongly fail-safe" de coupure d'alimentation. Dans le cas de la circuiterie de coupure d'alimentation (figure I.10), seule une intervention rapide peut éviter que le courant soit coupé!

L'alimentation n'étant pas coupée, les modules peuvent être testés après sauvegarde en vue d'une restauration ultérieure de l'état. Dans la mesure où les mémoires du système intègrent du BIST transparent [Nic92], elles peuvent être utilisées pour sauvegarder ces états. Suite au test des modules, si aucune faute permanente n'est détectée, les indicateurs d'erreur des interfaces d'entrée "self-checking" peuvent être remis à un état sans faute.

Par la suite, si une faute permanente dangereuse est détectée durant le test de l'interface pour la tolérance aux fautes, les indicateurs d'erreur concernés seront encore remis à un état erroné.

D'une part, si l'interface et au moins l'un des modules sont toujours disponibles, les modules "self-checking" sans faute peuvent restaurer l'état correct du système à partir de leur propres mémoires (si le module était à un état sans faute avant le test), à partir de l'autre module (si l'autre module était et est toujours sans faute), ou bien l'activité peut être reprise à partir d'un point de repère predéfini.

D'autre part, si une faute permanente dangereuse est détectée dans l'interface, ou si aucun module n'est disponible, le courant du circuit où l'interface est intégrée sera coupé par la circuiterie de coupure d'alimentation. Par conséquent, le système sera déconnecté des actionneurs.

Finalement, cette phase hors-ligne, permettant de récupérer le système après la détection d'une faute transitoire, pourrait être activé et ordonné par un processeur de test (chapitre III) intégré dans le même circuit intégré où se trouve l'interface pour la tolérance aux fautes. Ce processeur pourrait vérifier constamment la circuiterie de coupure d'alimentation afin d'assurer qu'une faute transitoire affectant l'interface ne mène pas à une coupure de courant du circuit.

V.3.3 Evaluation de la fiabilité

Afin de pouvoir comparer des schémas tolérant les fautes qui présentent des coûts en matériel équivalents, discutons d'avantage la fiabilité et la sécurité du système TMR introduit au paragraphe V.2□

- Supposons que le mécanisme de vote ait une fiabilité R pour un temps donné de mission. Le module simplex étant T fois plus complexe que le voteur, la fiabilité de chaque

copie du module sera donnée par R^T. Puisque le système tripliqué ne sera plus disponible, si le voteur tombe en panne, ou si deux copies au moins tombent en panne, alors la fiabilité du système TMR sera donnée par□

$$R_{TMR} = [R^T.R^T.R^T + 3.R^T.R^T.(1-R^T)].R$$
$$R_{TMR} = (3.R^{2T} - 2.R^{3T}).R \qquad \textbf{[I]}$$

- Dans un module simplex, puisque aucune indication de défaillance du système n'est pourvue, la sécurité S du module sera identique à sa fiabilité R^T. A priori, la même remarque est valable pour le cas du TMR, et alors $S_{TMR}=R_{TMR}$. Afin d'améliorer la sécurité du système TMR, premièrement le voteur peut être implémenté en utilisant les mêmes mécanismes de détection de fautes que ceux utilisés dans les systèmes redondant hybrides ou bi-duplex, permettant ainsi l'identification du module fautif et l'auto-reconfiguration quand 2 modules tombent en panne. Deuxièmement, afin de couvrir ses propres fautes, le voteur peut être conçu pour être "self-checking" et, ensuite, "strongly fail-safe", de manière semblable aux techniques présentées au paragraphe V.3.2. Seule la combinaison des deux stratégies peut amener à $S_{TMR}=R$. Ce type d'implémentation résulterait dans une consommation plus importante de silicium que dans le cas de notre interface pour la tolérance aux fautes, car les sorties de trois répliques devraient être traitées.

Afin de calculer la fiabilité de notre système tolérant les fautes et le comparer à un système TMR avec le même niveau de sécurité, supposons la même fiabilité pour l'interface pour la tolérance aux fautes et pour le voteur. Bien que cette supposition soit pessimiste par rapport à la fiabilité finale de notre schéma tolérant les fautes (c[illegible]oteur "strongly fail-safe" serait plus gourmand en surface que l'interface pour la tolérance aux [illegible]utes), elle simplifie grandement l'analyse qui suit. Supposons aussi que le module [illegible]plex est T fois plus complexe que l'interface pour la tolérance aux fautes et que le module "self-checking" correspondant est [illegible] fois plus complexe que sa contre[illegible]ex. [illegible] la fiabilité de chaque copie du module "self-checking" se[illegible] R [illegible]. Puisque le système "self-checking" ne sera plus disponible, si l'interface [illegible] aux fautes tombe en panne, ou si les deux copies tombent en panne, alors [illegible] R sera donnée par□

$$R_{SC} = [R^{T}.[illegible]$$
$$R_{SC} = (2.R^{T}[illegible] \qquad \textbf{[II]}$$

Il est connu qu'un surcoût moyen en surface de l'ordre de 20.1 à 46.9% peut être obtenu pour des PLAs "self-checking" [MAD82, Ser88, TNF91, NiB91], que la conception de RAMs "self-checking" et RAMs UBIST transparent résulte dans moins de 20% de surcoût [Nic94], que l'implémentation d'additionneurs et ALUs "self-checking" peut amener à des surcoûts de 10.7 à 51.4% et de 31 à 84% respectivement [GaR68, SHB68, Nic93], et qu'un surcoût dans

l'intervalle de 23 à 60% peut être obtenu pour la conception "self-checking" de parties de contrôle de microprocesseurs [Dis81, Nic85, Nic90]. Il est donc raisonnable de s'attendre à avoir un surcoût en surface de silicium dans l'intervalle 20-60% à partir de la transformation d'un circuit complexe dans un circuit "self-checking". Au niveau de la carte et du module le surcoût en matériel ne sera sûrement pas aussi important que pour un circuit intégré, car la parité dominera le codage de données [LuC92].

Allant plus loin dans la comparaison entre les deux approches, dans la figure V.8 les fiabilités du module simplex, du TMR et du système "self-checking" (SC) sont tracées pour des missions relativement courtes (R=0,99 et R=0,999), pour variant de 1,2 à 1,6 et pour plusieurs complexités de module (10≤T≤200).

D'une part, les courbes montrent clairement la supériorité du système "self-checking" en comparaison au système TMR. D'autre part, on peut constater que la fiabilité du système "self-checking" devient inférieure à celle du module simplex seulement pour des complexités beaucoup plus importantes que dans le cas du TMR.

En fait, on s'attend à ce que varie en fonction de la complexité du module simplex que l'on veut faire devenir "self-checking". Par exemple, il est attendu que soit plus important pour les modules de faible complexité que pour ceux de haute complexité [NiB91, Nic93]. Ainsi, afin d'évaluer s'il vaut la peine de concevoir un module "self-checking" de faible complexité à utiliser dans un système tolérant les fautes, il serait utile de connaître à l'avance la limite supérieure pour laquelle l'inéquation $R_{SC}>R_{TMR}$ soit assurée.

Pour déterminer cette limite, commençons par égaler [I] et [II] par simplifier l'expression résultante

$$(R^T)^3 - 2.(R^T)^2 + 2.(R^T)^3 = 0 .$$

Après résolution de cette équation biquadratique pour $0 \le R^T \le 1$, on obtient :

$$R^T = 1 \text{ ou } \frac{1}{2}.(R^T)^2 + 2.(R^T)^3 .$$

L'isolation de donne :

$$= \sqrt{-\frac{3}{2}.(R^T)^2 + 2.(R^T)^3} \qquad [III]$$

Puisque [III] est une fonction décroissante, la valeur qui nous intéresse correspond à la valeur de pour laquelle R^T converge vers 1. Confronté à une indétermination pour $(R^T=1)$, on calcule la limite comme suit :

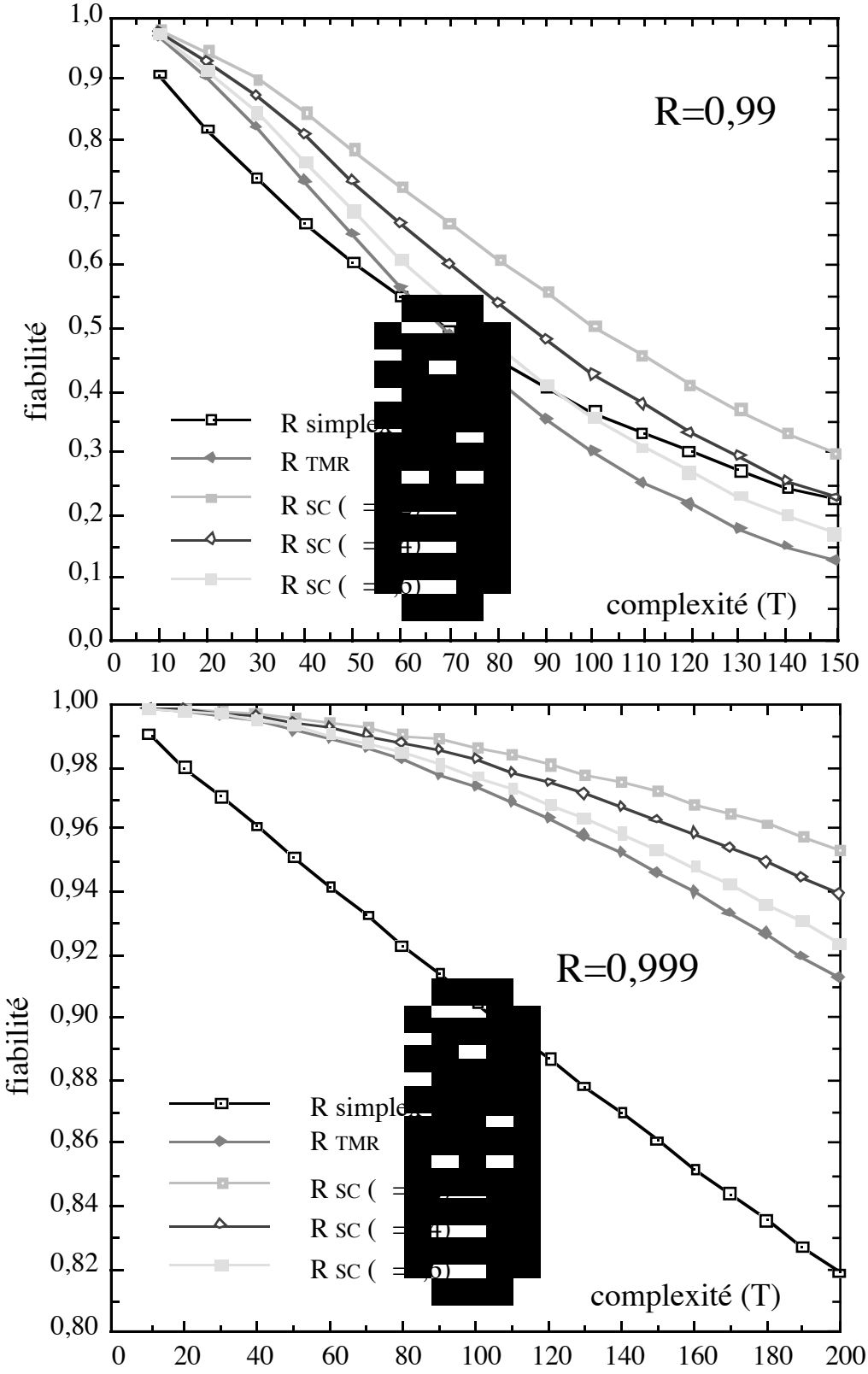


Figure V.8. Comparaison entre les fiabilités

$$\lim_{R^T \to 1} \frac{\frac{d}{dR^T}[1 - 1\sqrt{3.(R^T)^2+2.(R^T)^3}]}{\frac{d}{dR^T}[\ln(R^T)]},$$

$$\lim_{R^T \to 1} \frac{3.(R^T)^2}{[(R^T)^2+2.(R^T)^3].\sqrt{2.(R^T)+1}},$$

et on obtient :

$$\lim_{R^T \to} \sqrt{3} \quad 732 \qquad \textbf{[IV]}$$

En pratique, le résultat reporté par [IV] signifie que, pour les surcoûts de modules "self-checking" non supér 73% ([√3-1].100), notre schéma tolérant les fautes résultera dans un système plus fi le que l TMR correspondant. Evidemment, pour les surcoûts situés dans l'intervalle 1.5≤ ≤ 73, le ème "self-checking" aura une surface plus importante que celle du TMR.

V.4 Conclusion

Dans ce chapitre, nous avons présenté un schéma tolérant les fautes basé sur la duplication de modules "self-checking". Nous avons proposé aussi une interface pour la tolérance aux fautes capable de transposer les sorties erronées du système résultant en un état de sortie sûr. Nous avons montré que cette interface peut devenir "strongly fail-safe" si on y intègre les capacités BIST adéquates. De cette manière, la haute sécurité inhérente aux systèmes "self-checking" est préservée dans notre schéma.

Pour des surcoûts en matériel situés entre 20 et 60% du module simplex, situation typique pour les modules "self-checking" complexes, la fiabilité que notre schéma peut atteindre est supérieure à celle obtenue avec un système TMR ayant un niveau de sécurité équivalent. On montre que la limite théorique pour laquelle le système "self-checking" peut assurer cette fiabilité est √3-1 (surcoût de 73%). Etant donné ceci, on peut conclure que les systèmes "self-checking" peuvent non seulement assurer une haute sûreté de fonctionnement aux applications critiques, mais ils peuvent aussi être compétitifs en termes de fiabilité dans l'exécution de missions de courte durée.

La sécurité mise à part, d'autres avantages de notre système par rapport à la structure TMR sont que notre schéma est plus convenable en ce qui concerne la détection d'erreurs doubles (et, de cette manière, la détection de courts-circuits entre deux fils connectant deux modules différents à l'interface de sortie du système) et, la réparation devient plus simple car

l'unité à remplacer (un circuit, une carte ou même un module) peut être facilement identifiée à travers l'examen d'indicateurs d'erreur.

Il est vrai que, contrairement au TMR, notre schéma dépend fortement de l'hypothèse de faute simple. Pourtant, des études préalables (comme, par exemple, [Cou81]) ont démontré que cette hypothèse est réaliste pour le test en-ligne, car les mécanismes de défaillances liées à la durée de vie des systèmes sont extrêmement lents.

Il est aussi vrai qu'un module "self-checking" ne peut pas être construit à partir de composants du marché ("off-the-shelf"). Cette situation ne peut être changée qu'à travers le développement d'une vaste gamme de systèmes "self-checking", et pour que ceci soit économiquement viable, un nombre suffisant d'utilisateurs et d'outils de support à la conception doivent exister.

CONCLUSION

Partant des techniques existantes pour le BIST unifié de circuits intégrés, pour l'interfaçage "fail-safe" de ces circuits avec des actionneurs électromécaniques et pour le test "boundary scan", cette thèse a apporté des solutions à des problèmes de test hors-ligne de cartes, et a proposé l'unification des tests hors-ligne et en-ligne au niveau du circuit, de la carte et du module en vue de la conception de systèmes digitaux sécuritaires et fiables.

Nous avons présenté des méthodes pour le test hors-ligne et le diagnostic adaptées à des étapes différentes dans l'évolution du test "boundary scan" de cartes□ nous sommes partis du cas "boundary scan" partiel et sommes arrivés au traitement de l'association BIST-BS de cartes dans son intégralité. Nous avons proposé ensuite la combinaison des techniques UBIST et "boundary scan" en vue de l'unification des tests hors-ligne et en-ligne, de connexions et de circuits, appliquée aux cartes et modules. Finalement, nous avons réussi à rendre fiables des systèmes sécuritaires basés sur la duplication de modules conçus d'après notre stratégie pour le test unifié.

De manière générale, les travaux de recherche menés au cours de cette thèse se caractérisent non seulement par leur actualité, mais aussi par le fait qu'ils répondent à des besoins croissants chez les industriels. De plus, toutes les activités se placent dans des domaines sensibles et critiques, puisqu'elles mettent en jeu la fiabilité et la sécurité des machines et donc des personnes qui s'en servent.

Bien que des résultats prometteurs aient été obtenus tout au long de cette thèse, il reste encore beaucoup à faire, d'abord pour vulgariser les techniques proposées et ensuite pour mettre à jour l'unification de test par rapport à l'état de l'art des systèmes électroniques.

• Afin d'aboutir à la vulgarisation de nos méthodes, les activités de recherche doivent concerner les voies d'investigation suivantes□

A courte échéance□

Consolider les résultats obtenus jusqu'à présent par la validation des techniques proposées. Cette tâche passe surtout par l'implantation sur silicium des méthodologies développées, leur application dans des cas réels et l'analyse des performances obtenues.

L'intérêt de cette analyse dépasse le cadre d'une simple validation, puisqu'elle permettra également de définir des améliorations à apporter sur les stratégies de conception.

A moyenne et longue échéance:

Vu que l'implantation massive de nos techniques risque d'être entravée par l'effort de conception engendré, le développement d'une large gamme de systèmes conçus en vue du test unifié ne sera faisable et économiquement viable que lorsqu'il y aura un nombre suffisant d'outils de support.

Bien qu'on se soucie de ce problème depuis quelques années et qu'on l'ait déjà résolu pour des cas particuliers [TNJ88, TNF91, NiB91, NiB92, HBN94], l'intégration dans un environnement robuste de chacun des outils disponibles à présent est indispensable, dans un premier temps, afin de permettre la conception assistée par ordinateur de systèmes complexes testables.

Dans un deuxième temps, il faudra envisager la synthèse automatique basée sur la description du comportement spécifié pour ces systèmes. Par conséquent, le grand défi d'une telle synthèse de haut niveau portera sur la prise en compte des spécifications et contraintes de test, permettant la couverture de défauts physiques survenant au niveau dispositif/connectique [MCC94, HTW94].

• D'autre part, pour la mise à jour du test unifié par rapport à l'état de l'art des systèmes électroniques, il devient crucial de considérer les problèmes de test analogique: les statistiques montrent que 50% des circuits intégrés possèderont bientôt une partie analogique, dont le test a été très peu abordé jusqu'à maintenant; et n'oublions pas que la vitesse croissante d'opération des circuits fait que les parties digitales posent aussi des problèmes de test analogique.

Il faudra donc envisager le développement de nouvelles techniques pour le test unifié de systèmes hétérogènes, comme le montre la figure c.1. On remarquera que le départ a déjà été donné: le test hors-ligne [SoK94] et le test en-ligne [KLC93, LKM94, KLC94] de circuits intégrés analogiques ont été adressés séparément; et un standard IEEE pour le test de cartes mixtes analogique-digital est en préparation [Som94].

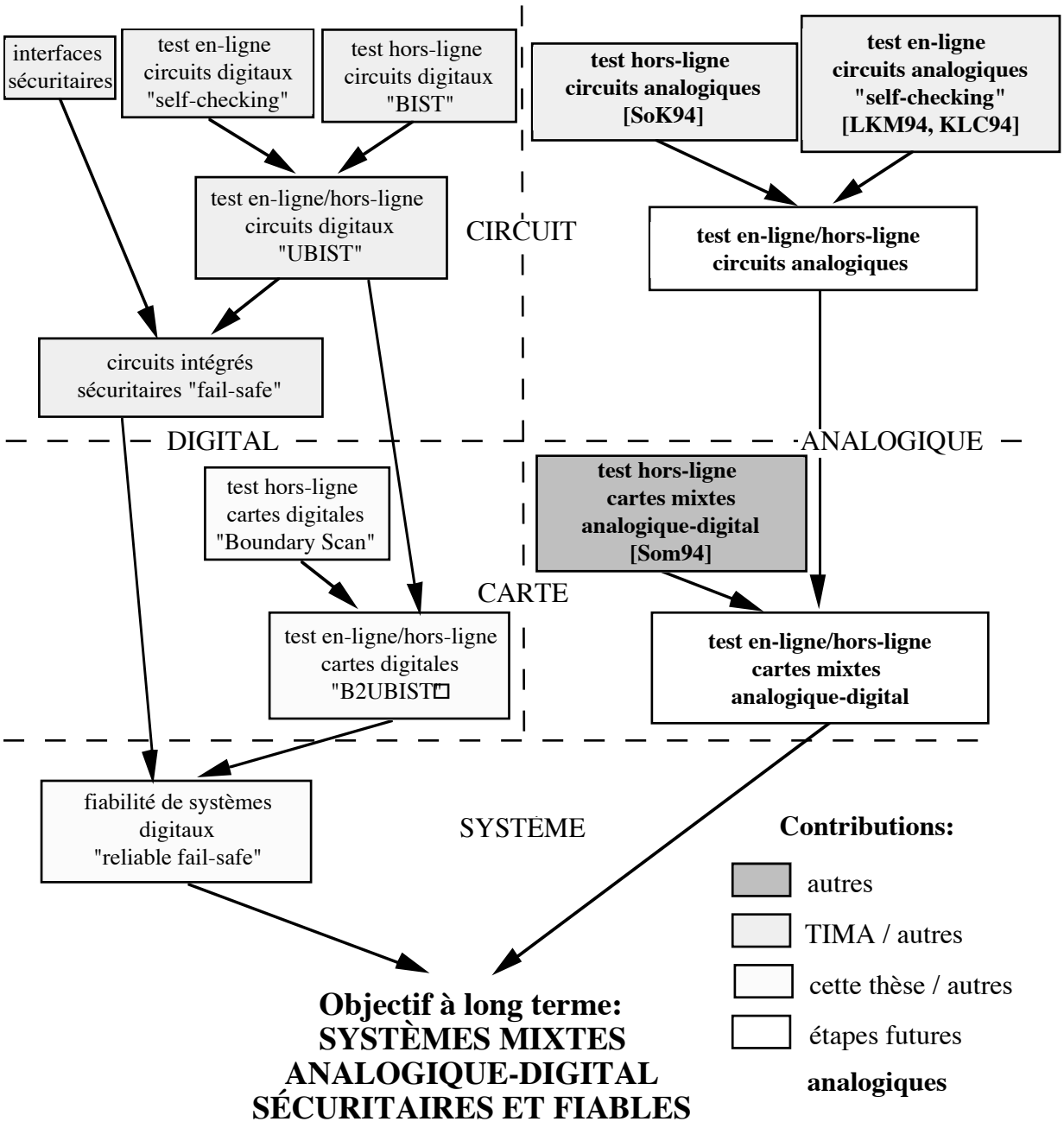


Figure c.1. Le test unifié appliqué à la conception de systèmes hétérogènes fiables.

RÉFÉRENCES

[ACL78] J. Arlat, Y. Crouzet et C. Landrault, "Operational Secure Microcomputers", Conference on Fault-Tolerant Systems and Diagnosis, 1978, 1-9.

[And71] D. A. Anderson, "Design of Self-Checking Digital Networks Using Coding Techniques", Coordinated Science Laboratory, rapport n° 527, University of Illinois, Urbana, 1971.

[ArL85] J. Arlat et C. Laprie, "On the Dependability Evaluation of High Safety Systems", International Symposium on Fault-Tolerant Computing, 1985, 318-323.

[Bha91] D. K. Bhavsar, "Testing of Interconnections to Static RAMs", Design & Test, Juin 1991, 63-71.

[Ber61] J. M. Berger, "A Note on Error Detection Codes for Asymetric Binary Channels", Inform. Contr., Vol. 4, Mars 1961, 68-73.

[BrF76] M. A. Breuer et A. D. Friedman, "Diagnosis and Reliable Design of Digital Systems", Computer Science Pres Inc., 1976.

[BrF85] F. Brglez et H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran", International Test Conference, 1985, 785-794.

[CaS68] W. C. Carter et P. R. Schneider, "Design of Dynamically Checked Computers", IFIP Congress, Edinburg 1968, Inf. Proc. 68, Amsterdam, North Holland, 1969.

[ChJ92] Y.-H. Choi et T. Jung, “Configuration of a Boundary Scan Chain for Optimal Testing of Clusters of non Boundary Scan Devices”, International Conference on Computer Aided Design, 1992, 13-16.

[CLW90] W.-T. Cheng, J. L. Lewandowski et E. Wu, "Diagnosis for Wiring Networks", International Test Conference, 1990, 565-571.

[Cou76] B. Courtois, "On Balancing Safety and Reliability of Hybrid and "Bi-Duplexed" Systems", International Symposium on Fault-Tolerant Computing, 1976, 52-57.

[Cou81] B. Courtois, "Failure Mechanisms, Fault Hypotheses and Analytical Testing of LSI-NMOS (HMOS) Circuits", VLSI Conference, 1981.

[Cro78] Y. Crouzet, "Conception de Circuits à Large Échelle d'Intégration Totalement Autotestables", thèse de Docteur-Ingénieur, Toulouse, Novembre 1978.

[Der91] B. I. Dervisoglu, "Features of a Scan and Clock Resource Chip for Providing Access to Board-Level Test Functions", Journal of Electronic Testing: Theory and Applications, Vol. 2, N° 1, Mars 1991, 107-115.

[Dis81] C. Disparte, "A Design Approach for Electronic Engine Controller Self-Checking Microprocessor", EUROMICRO Symposium, 1981.

[DUY89] A. T. Dahbura, M. U. Uyar et C. W. Yau, "An Optimal Test Sequence for the JTAG/IEEE P1149.1 Test Access Port Controller", International Test Conference, 1989, 55-62.

[EiW78] E. B. Eichelberger et T. W. Williams, "A Logic Design Structure for LSI Testability", Journal of Design Automation and Fault-Tolerant Computing, Vol. 2, N° 2, Mai 1978, 165-178.

[Esc92] B. Eschermann, "An Implicitly Testable Boundary Scan TAP Controller", Journal of Electronic Testing: Theory and Applications, Vol. 3, 1992, 159-169.

[FMP92] J. M. M. Ferreira, J. S. Matos et F. S. Pinto, "Automatic Generation of a Single-Chip Solution for Board-Level BIST of Boundary Scan Boards", International Test Conference, 1992, 154-158.

[Fri71] A. D. Friedman, "Fault Detection in Digital Circuits", Prentice Hall Inc., 1971.

[FSM88] K. Futsuhara, N. Sugimoto et M. Mukaidono, "Fail-Safe Logic Elements Having Upper and Lower Thresholds and Their Application to Safety Control", International Symposium on Fault-Tolerant Computing, 1988.

[FuA84] K. Fuchs et J. A. Abraham, "A Unified Approach to Concurrent Error Detection in Highly Structured Logic Arrays", International Symposium on Fault-Tolerant Computing, Kissemmee, USA, 1984.

[Gai85] N. Gaitanis, "A Totally Self-Checking Error Indicator", IEEE Transactions on Computers, Vol. C-34, N° 8, Août 1985.

[GaR68] O. N. Garcia et T. R. N. Rao, "On the Method of Checking Logical Operations", Princeton Conf. Inform. Sci. Sys., 1968, 89-95.

[GCV80] J. A. Galay, Y. Crouzet et M. Vergniault, "Physical Versus Logical Fault Models MOS-LSI Circuits: Impact of their Testability", IEEE Transactions on Computers, Vol. C-29, Juin 1980.

[GlB88] C. S. Gloster et F. Brglez, "Boundary Scan with Cellular-Based Built-In Self-Test", International Test Conference, Septembre 1988, 138-145.

[GoM82] P. Goel et M. McMahon, "Electronic Chip-in-Place Test", International Test Conference, 1982, 83-90.

[HaB84] M. P. Halbert et S. M. Bose, "Design Approach for a VLSI Self-Checking MIL-STD-1750 Microprocessor", International Symposium on Fault Tolerant-Computing, Kissemmee, USA, 1984.

[Han89] P. Hansen, "Testing Conventional Logic and Memory Clusters using Boundary Scan Devices as Virtual ATE Channels", International Test Conference, 1989, 166-173.

[HAN92] A. Hassan, V. K. Agarwal, B. Nadeau-Dostie et J. Rajski, "BIST of PCB Interconnects Using Boundary-Scan Architecture", Transactions on Computer-Aided Design, V. 11, N° 10, Octobre 1992, 1278-1288.

[HAR89] A. Hassan, V. K. Agarwal, J. Rajski et B. N. Dostie, "Testing of Glue Logic Interconnects using Boundary Scan Architecture", International Test Conference, 1989, 700-711.

[HBN94] B. Hamdi, H. Bederr et M. Nicolaidis, "CAD Tools for Automatic Generation of Self-Checking Datapaths", ARCHIMEDES Workshop on Test Synthesis, Février 1994.

[HRA88] A. Hassan, J. Rajski et V. K. Agarwal, "Testing and Diagnosis of Interconnects using Boundary Scan Architecture", International Test Conference, 1988, 254-265.

[HTW94] S. Hellebrand, P. Teixeira et H.-J. Wunderlich, "Synthesis for Testability - The ARCHIMEDES Approach", International Test Synthesis Workshop, Mai 1994.

[IEE90] IEEE Standard 1149.1-1990, "IEEE Standard Test Access Port and Boundary Scan Architecture", IEEE Standards Board, 345 East 47th Street, New York, NY 10017-2394, USA, 1990.

[JaY89] N. Jarwala et C. Yau, "A New Framework for Analyzing Test Generation and Diagnosis Algorithms for Wiring Interconnects", International Test Conference, 1989, 63-70.

[JaY91] N. Jarwala et C. Yau, "Achieving Board-Level BIST Using the Boundary-Scan Master", International Test Conference, 1991, 649-658.

[JoH91] F. Jong et F. V. D. Heyden, "Testing the Integrity of the Boundary Scan Test Infrastructure", International Test Conference, 1991, 106-112.

[JMF91] F. Jong, J. S. Matos et J. M. Ferreira, "Boundary Scan Test, Test Methodology, and Fault Modeling", Journal of Electronic Testing: Theory and Applications, Vol. 2, 1991, 77-88.

[Kau74] W. Kautz, "Testing for Faults in Wiring Networks", IEEE Transactions on Computers, Vol. C-23, N° 4, Avril 1974, 358-363.

[KLC93] V. Kolarik, M. Lubaszewski et B. Courtois, "Towards Self-Checking Mixed-Signal Integrated Circuits", European Solid-State CIRCuit Conference, 1993, 202-205.

[KLC94] V. Kolarik, M. Lubaszewski et B. Courtois, "Designing Self-Exercising Analogue Checkers",VLSI Test Symposium, 1994, 252-257.

[KMZ79] B. Koenemann, J. Mucha et G. Zwiehoff, "Built-In Logic Block Observer", International Test Conference, 1979, 37-41.

[LAN93] M. Lubaszewski, V. C. Alves, M. Nicolaidis et B. Courtois, "Checking Signatures on Boundary Scan Boards", European Test Conference, 1993, 339-348.

[LCG89] J.-C. Laprie, B. Courtois, M.-C. Gaudel et D. Powell, "Sûreté de Fonctionnement des Systèmes Informatiques Matériels et Logiciels", DUNOD Informatique, 1989.

[LeB84] J. J. LeBlanc, "LOCST: A Built-In Self-Test Technique", IEEE Design and Test, November 1984, 45-52.

[Lef90] M. F. Lefebvre, "Functional Test and Diagnosis: A Proposed JTAG Sample Mode Scan Tester", International Test Conference, 1990, 294-303.

[LDN92] J.-C. Lo, J. Daly et M. Nicolaidis, "Static CMOS Self-Checking Circuits", International Symposium on Fault-Tolerant Computing, 1992, 104-111.

[LiB89] J.-C. Lien et M. A. Breuer, "A Universal Test and Maintenance Controller for Modules and Boards", Transactions on Industrial Electronics, Vol. 36, N° 2, Mai 1989, 231-240.

[LiB91] J.-C. Lien et M. A. Breuer, "Maximal Diagnosis for Wiring Networks", International Test Conference, 1991, 96-105.

[LKM94] M. Lubaszewski, V. Kolarik, S. Mir et B. Courtois, “Self-Checking Analogue and Mixed-Signal Fully Differential Circuits”, BIST/DFT Workshop, 1994.

[LMT94] M. Lubaszewski, M. Marzouki et M. H. Touati, “A Pragmatic Test and Diagnosis Methodology for Partially Testable MCMs”, MultiChip Module Conference, 1994, 108-113.

[LuC92] M. Lubaszewski et B. Courtois, "On the Design of Self-Checking Boundary Scannable Boards", International Test Conference, 1992, 372-381.

[LuC93] M. Lubaszewski et B. Courtois, "Reliable Fail-Safe Systems", Asian Test Symposium, 1993, 32-37.

[MAD82] G. P. Mak, J. A. Abraham et E. S. Davindson, "The Design of PLAs with Concurrent Error Detection", International Symposium on Fault-Tolerant Computing, 1982, 303-310.

[Mar91] M. Marzouki, "Model-Based Reasoning for Electron-Beam Debugging of VLSI Circuits", Journal of Electronic Testing: Theory and Applications, Vol. 2, N° 4, 1991, 385-394.

[MaT90] C. M. Maunder et R. E. Tulloss, "The Test Access Port and Boundary Scan Architecture", IEEE Computer Society Press, 1990.

[Mau94] C. M. Maunder, "The Test Access Port and Boundary Scan Architecture: An Introduction to ANSI/IEEE Std. 1149.1 and its applications", Forum on Boundary Scan for Digital and Mixed-Signal Boards, CERN, Génève-Suisse, Janvier 1994.

[MCC94] M. Marzouki, B. Courtois et V. C. Alves, "High Level Synthesis for Testability: Where should we go from?", International Test Synthesis Workshop, Mai 1994.

[McS86] W. H. McAnney et J. Savir, "Built-In Checking of the Correct Self-Test Signature", International Test Conference, 1986, 54-58.

[MeB92] M. Melton et F. Brglez, “Automatic Pattern Generation for Diagnosis of Wiring Interconnect Faults”, International Test Conference, 1992, 389-398.

[MiK67] H. Mine et Y. Koga, "Basic Properties and a Construction Method for Fail-Safe Logical Systems", IEEE Trans. Elec. Comp., Vol. EC-16, Juin 1967.

[MLT93] M. Marzouki, M. Lubaszewski et M. H. Touati, “Unifying Test and Diagnosis of Interconnects and Logic Clusters in Partial Boundary Scan Boards”, International Conference on Computer Aided Design, 1993, 654-657.

[NiB91] M. Nicolaidis et M. Boudjit, "New Implementations, Tools and Experiments for Decreasing Self-Checking PLAs Area Overhead", International Conference on Computer Design, 1991, 275-281.

[NiB92] M. Nicolaidis et M. Boudjit, "Verification of Self-Checking Properties by means of Output Code Space Computation", European Conference on Design Automation, 1992, 169-174.

[Nic85] M. Nicolaidis, "Evaluation of a Self-Checking Version of the MC68000 Microprocessor", International Symposium on Fault-Tolerant Computing, 1985, 350-356.

[NiC85] M. Nicolaidis et B. Courtois, "Layout Rules for the Design of Self-Checking Circuits", VLSI Conference, 1985.

[NiC86] M. Nicolaidis et B. Courtois, "Design of Self-Checking Circuits Using Unidirectional Error Detecting Codes", International Symposium on Fault Tolerant-Computing, 1986.

[Nic87] M. Nicolaidis, "Shorts in Self-Checking Circuits", International Test Conference, 1987, 408-417.

[Nic88] M. Nicolaidis, "A Unified Built-In Self-Test scheme: UBIST", International Symposium on Fault-Tolerant Computing, 1988, 157-163.

[Nic90] M. Nicolaidis, "Efficient UBIST Implementation for Microprocessor Sequencing Parts", International Test Conference, 1990, 316-326.

[Nic92] M. Nicolaidis, "Transparent BIST for RAMs", International Test Conference, 1992, 598-607.

[Nic93] M. Nicolaidis, "Efficient Implementations of Self-Checking Adders and ALUs", International Symposium on Fault-Tolerant Computing, 1993, 586-595.

[Nic94] M. Nicolaidis, "Efficient UBIST for RAMs", VLSI Test Symposium, 1994, 158-166.

[NJC84] M. Nicolaidis, I. Jansch et B. Courtois, "Strongly Code Disjoint Checkers", International Symposium on Fault-Tolerant Computing, 1984, 16-21.

[NKL91] P. Nagvajara, M. G. Karpovsky et L. B. Levitin, "Pseudorandom Testing for Boundary-Scan Design with Built-In Self-Test", IEEE Design & Test of Computers, Septembre 1991, 58-65.

[NNC89] M. Nicolaidis, S. Noraz et B. Courtois, "A Generalized Theory of Fail-Safe Systems", International Symposium on Fault-Tolerant Computing, 1989, 398-406.

[NWA91] B. Nadeau-Dostie, P. S. Wilcox et V. K. Agarwal, "A Scan-based BIST Technique Using Pair-wise Compare of Identical Components", International Symposium on VLSI Design, 1991, 225-230.

[PeW72] W.W. Peterson et E.J. Weldon, "Error-Correcting Codes", deuxième édition, The MIT press, Cambridge, Massachusetts, 1972.

[Phi93] Philips Electronics, "Product Data Sheet, PM 3790 BSD Boundary Scan Diagnostics", 1993.

[RoD90] G. D. Robinson et J. G. Deshayes, "Interconnect Testing of Boards with Partial Boundary Scan", International Test Conference, 1990, 572-581.

[Sed80] R. M. Sedmark, "Implementation Techniques for Self-Verification", International Test Conference, 1980, 267-278.

[Ser88] M. Serra, "Some Experiments on the Overhead for Concurrent Checking", Workshop on New Directions for Integrated Circuits Testing, 1988.

[SiS82] D. P. Siewiorek et R. S. Swarz, "The Theory and Practice of Reliable System Design", Digital Press, 1982.

[SHB68] F. F. Sellers, M.-Y. Hsiao et L. W. Bearnson, "Error Detecting Logic for Digital Computers", McGraw Hill, 1968.

[SmM78] J. E. Smith et G. Metze, "Strongly Fault Secure Logic Networks", IEEE Transactions on Computers, Vol. C-27, N° 6, Juin 1978.

[SoK94] M. Soma et V. Kolarik, "A Design-for-Test Technique for Switched-Capacitor Filters", VLSI Test Symposium, 1994, 42-47.

[Som94] M. Soma, "IEEE P1149.4 Mixed-Signal Test Bus Standard", Forum on Boundary Scan for Digital and Mixed-Signal Boards, CERN - Génève, Janvier 1994.

[TFH92] M. V. Tegethoff, T. E. Figal et S. W. Hird, "Board Test DFT Model for Computer Products", International Test Conference, 1992, 367-371.

[TII89] Texas Instruments Inc., "Product Preview, SN54BCT8244, SN74BCT8244 Scan Test Device with Octal Buffer", 1989.

[TNF91] K. Torki, M. Nicolaidis et A. O. Fernandes, "A Self-Checking PLA Automation Generator Tool Based on Unordered Codes Enconding", European Conference on Design Automation, 1991, 510-515.

[TNJ88] K. Torki, M. Nicolaidis, A. Jerraya et B. Courtois, "UBIST Version of the SYCO's Control Section Compiler", International Conference on Computer Design, 1988, 392-396.

[TRB91] P. Thorel, J. L. Rainard, A. Botta, A. Chemarin et J. Majos, "Implementing Boundary-Scan and Pseudo-Random Built-In Self-Test in a 0.7 Micron CMOS Asynchronous Transfer Mode Switch", European Test Conference, Avril 1991, 513.

[TuY89] R. E. Tulloss et C. W. Yau, "BIST & Boundary Scan for Board Level Test: Test Program Pseudocode", European Test Conference, 1989, 106-111.

[UsE89] P. A. Uszynski et A. C. Erdar, "Hybrid Global Test Strategy", High Performance Systems, Janvier 1989, 68-74.

[Vin89] S. Vining, "Tradeoff Decisions Made for a P1149.1 Controller Design", International Test Conference, 1989, 47-54.

[Wad78] R. L. Wadsack, "Fault Modelling and Logic Simulation of CMOS and MOS Integrated Circuits", The Bell System Technical Journal, Mai 1978.

[Wag87] P. Wagner, "Interconnect Testing with Boundary Scan", International Test Conference, 1987, 52-57.

[WDG86] T. W. Williams, W. Daehn, M. Gruetzner et C. W. Starke, "Comparison of Aliasing Errors for Primitive and Non-Primitive Polynomials", International Test Conference, 1986, 282-288.

[WiP82] T. W. Williams et K. P. Parker, "Design for Testability - A Survey", Transactions on Computers, Vol. C-31, Janvier 1982, 2-15.

[YaJ89] C. W. Yau et N. Jarwala, "A Unified Theory for Designing Optimal Test Generation and Diagnosis Algorithms for Board Interconnects", International Test Conference, 1989, 71-77.

[YaJ90] C. W. Yau et N. Jarwala, "The Boundary Scan Master: Target Applications and Functional Requirements", International Test Conference, 1990, 311-315.