الجمهوريـــة الجزائريـــة الديمقراطيـــة الشعبيـــة

République Algérienne Démocratique et Populaire

وزارة التعلـــيم العـــالي و البحـــث العلـــمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعـــة فرحـات عبـــاس – سطيف -1-

Université Ferhat Abbas – Sétif -1-

Faculté de Technologie

Département d'Électronique

THÈSE

Présentée pour l'obtention du diplôme de

DOCTORAT EN SCIENCES

En **: ÉLECTRONIQUE**

Par

M. Abdelhakim LATOUI

Thème

CONCEPTION TESTABLE DE CIRCUITS INTÉGRÉS COMPLEXES À TRÈS HAUT NIVEAU

Date de soutenance : 12/12/2013

Devant le jury composé de :

Président :	M. A. KHELLAF	Prof.	Université Ferhat Abbas – Sétif -1-
Rapporteur :	M. F. DJAHLI	Prof.	Université Ferhat Abbas – Sétif -1-
Examinateurs :	M. D. CHIKOUCHE	Prof.	Université de M'sila
	M. R. ZIANI	M.C.A	Université M. Mammeri – Tizi Ouzou
	M. H. CHEMALI	M.C.A	Université Ferhat Abbas – Sétif -1-
	M. A. HOCINI	M.C.A	Université de M'sila

Remerciements

A l'issu de la réalisation de cette étude, je me plie à l'incontournable et aimable tradition qui est de remercier tous ceux qui ont de près ou de loin contribué ou coopéré à l'élaboration de ce travail de thèse.

Je tiens avant tout à exprimer ma très sincère reconnaissance à Monsieur Farid DJAHLI, Professeur à l'université Ferhat Abbas de Sétif, pour ses conseils toujours pertinents en tant que directeur de thèse. Je lui exprime ma profonde gratitude pour les encouragements et le soutien qu'il m'a apportés tout au long de ce travail. Qu'il trouve ici l'expression de mon intime respect.

Mes vifs remerciements vont également à Monsieur Raphaël CANALS, Maître de conférences à l'université d'Orléans (France), pour avoir facilité mon accueil au laboratoire PRISME. Je lui exprime ma vive reconnaissance pour toute l'aide qu'il m'a apportée et pour ses qualités humaines de compréhension et relationnelles pour finaliser ce travail.

Je tiens à remercier Monsieur Abdelhafid KHELLAF, Professeur à l'université Ferhat Abbas de Sétif, qui a bien voulu me faire l'honneur de présider le jury de cette thèse.

Je remercie au même titre Monsieur Djamel CHIKOUCHE, Professeur à l'université de M'sila, Monsieur Rezki ZIANI, Maître de conférences à l'université Mouloud Mammeri de Tizi Ouzou, Monsieur Abdesselam HOCINI, Maître de conférences à l'université de M'sila, et Monsieur Hamimi CHEMALI, Maître de conférences à l'université Ferhat Abbas de Sétif, qui ont tous accepté d'être membres de jury, témoignant ainsi de l'intérêt de ce travail.

Je suis également particulièrement reconnaissant à Monsieur Khaled ROUABAH, Maître de conférences à l'université Mohamed El Bachir Ibrahim de Bordj Bou Arreridj, de m'avoir mis en contact avec le laboratoire PRISME de l'université d'Orléans.

Que mon ami, Monsieur Mohamed El Hossine DAACHI, Maître de conférences à l'université Mohamed El Bachir Ibrahim de Bordj Bou Arreridj, trouve ici l'expression de ma profonde gratitude pour son aide, ses encouragements, son soutien moral et ses conseils amicaux.

Enfin, et dans le souci de n'oublier personne, que tous ceux qui m'ont aidé, de près ou de loin, que ce soit par leur amitié, leur encouragement ou leur soutien moral et matériel, trouvent ici l'expression de ma profonde gratitude.

Dédicaces

- *À ma mère, à ma mère et à ma mère*
- *À mon père*
- *À ma fille Sarra et à ma femme*
- *À mes frères et sœurs*
- *À tous ceux et celles qui me sont chers.*

A.Hakim LATOUI

Résumé

L'objectif de ce travail est de proposer une nouvelle approche de test en ligne pour les systèmes embarqués dédiés aux applications critiques. Cette approche de test en ligne est basée sur une approche BIST (*Built-In Self-Test*) dans laquelle des sondes optiques sont ajoutées à l'architecture classique BILBO (*Built-In Logic-Block Observation*). De ce fait, de nouveaux registres Optiques BILBO (OBILBO) sont proposés. La méthode de test complète développée dans cette thèse, adaptée à la technique de duplication matérielle, permet d'assurer une vérification fonctionnelle du système en exploitation afin de détecter la présence d'éventuelles erreurs, sans pour autant interrompre ou ralentir son fonctionnement normal. Elle permet, en outre, une détection immédiate de fautes de collage simples avec une latence d'erreur d'un seul cycle d'horloge, sans toutefois avoir recours à aucune opération de génération de vecteurs de test. Cette approche rend également possible la réalisation des tests hors ligne en diagnostiquant des fautes pouvant apparaître sur les registres OBILBO eux-mêmes. L'avantage de cette approche est qu'elle permet de réduire les N opérations de décalage série (N : nombre d'étages du registre OBILBO) nécessaires pour effectuer un pareil scan, en une seule opération de décalage et ainsi d'éviter le caractère série de ce registre. Outre cette possibilité, le registre OBILBO produit aussi une signature optique qui peut être analysée grâce à un système distant. L'outil VHDL-AMS est utilisé pour la modélisation, la simulation et la validation de la technique présentée dans ce manuscrit.

Mots Clés : *Conception de circuits intégrés, Synthèse de circuits, TLM, SystemC, SystemVerilog, VHDL, Test et testabilité.*

Abstract

The objective of this work is to propose a new online testing approach for embedded systems which are dedicated to critical applications. This online testing approach is based on a BIST (Built-In Self-Test) approach in which optical sensors are added to the classical architecture BILBO (Built-In Logic-Block Observation). Thus, a new Optical BILBO registers (OBILBO) are proposed. The complete test method developed in this thesis, adapted to the hardware duplication technique, ensures a functional checking of the system during its normal operating in order to detect the presence of possible errors without interrupting or slowing down its normal functioning. Besides, it allows immediate detection of simple stuck at faults with error latency of only one clock cycle without having recourse to any operation of test vectors generation. In addition, this approach also allows to perform tests in offline mode by diagnosing faults that may appear on the OBILBO registers themselves. The advantage of this approach is to reduce the number of the N shifting operations (N: number of the OBILBO register stages) necessary to perform such a scan to a single shifting operation and thus to avoid the serial nature of this register. In addition to this possibility, the OBILBO register equally produces an optical signature that can be analyzed at remote system. The VHDL-AMS tool is used to model, simulate and validate the technique presented in this manuscript.

Keywords : *Integrated Circuit Design, Synthesis of Circuits, TLM, SystemC, SystemVerilog, VHDL, Test and Testability.*

Table des matières

Liste des Abréviations et Symboles

Abréviation	Description
ALU	Arithmetic Logic Unit
ASIC	Application-Specific Integrated Circuit
ATPG	Automatic Test Pattern Generation
BILBO	Built-In Logic-Block Observation
BIST	Built-In Self-Test
BS	Boundary Scan
CA	Cellular Array
CAD	Computer-Aided Design
CAN	Convertisseur Analogique/Numérique
CAO	Conception Assistée par Ordinateur
CBIST	Concurrent Built-In Self-Test
COTS	Component-Off-The-Shelf
CUT	Circuit Under Test
DFT	Design for Testability
DGF	Data Graph Flow
DL	Defect Level
DRAM	Dynamic Random Access Memory
DSP	Digital Signal Processor
EO	Électro-Optique
FF	Franc Français
FPGA	Field-Programmable Gate Array
GSI	Giga-Scale Integration
GVT	Générateur de Vecteurs de Test
IEEE	Institute of Electrical and Electronics Engineers
IFIS	If It Fails It Stops

IP	Intellectual Property
ITRS	International Technology Roadmap for Semiconductor
LFSR	Linear Feedback Shift Register
LSSD	Levet Sensitive Scan Design
MCM	Multi Chip Module
MISR	Multiple Input Signature Register
MPU	MIDI Processing Unit
OBILBO	Optical BILBO
ORA	Output Response Analyser
PCB	Printed Circuit Board
PDA	Personnel Digital Assistant
PSA	Parallel Signature Analysis
RAM	Random-Access Memory
RF	Radio Frequency
ROM	Read-Only Memory
SAR	Signature Analysis Register
SCD	Self-Checking Design
SIA	Semiconductor Industry Association
SIP	System In Package
SISR	Single Input Shift Register
SMT	Surface Mounted Technology
SO	Sonde Optique
SOC	System On Chip
SRAM	Static Random Access Memory
SSA	Serial Signature Analysis
TAM	Test Access Mechanism
TAP	Test Access Port
TLM	Transaction-Level Modeling
TMS	Test Mode Select
TPG	Test Pattern Generator
TSC	Totally Self-Checking
TTTC	Conseil Technique de la Technologie de Test
UBILBO	Unified Built-In Logic-Block Observation
UBIST	Unified Built-In Self-test
UDL	User-Defined-Logic
VHDL	HDL-Hardware Description Language

VHDL-AMS	VHSIC-Hardware Description Language–Analog and Mixed Systems
VLSI	Very Large Scale Integration
Y	Yiel

INTRODUCTION GÉNÉRALE

Le développement des systèmes embarqués est devenu aujourd'hui une part essentielle de l'activité industrielle. Plus de 90% de l'ensemble des composants électroniques produits dans le monde sont utilisés dans les systèmes embarqués et le nombre de ces systèmes, tous domaines confondus, est en constante augmentation [1]. En effet, les systèmes embarqués sont désormais utilisés dans des applications diverses telles que le transport (avionique, espace, automobile, ferroviaire), dans les appareils électriques et électroniques (appareils photo, jouets, postes de télévision, électroménager, systèmes audio, téléphones portables), dans la distribution et la gestion d'énergie, dans l'automatisation, etc. L'attrait principal des systèmes embarqués vient du fait qu'ils permettent d'implémenter, à faible coût, des fonctions complexes précédemment considérées comme exotiques ou dont la réalisation était inimaginable il y a quelques années seulement, dans les produits de tous les jours [2].

Les systèmes embarqués sont des systèmes autonomes qui intègrent du logiciel et du matériel conçus ensemble afin de fournir des fonctionnalités bien précises. Ils peuvent contenir une variété de composants [3] tels que les processeurs, les ASICs (Application-Specific Integrated Circuit), les mémoires, les périphériques d'entrée/sortie, etc. En fait, les systèmes embarqués utilisent, souvent, des composant pris en étagère, appelés COTS (Component-Off-The-Shelf), tels que les DSPs (Digital Signal Processor), les RAMs (Random-Access Memory), qui ne sont pas conçus spécialement avec une haute fiabilité mais sont intéressants au niveau du prix et de la facilité d'intégration.

Par ailleurs, les systèmes embarqués nécessitent habituellement un faible encombrement pour pouvoir s'insérer dans les appareils dans lesquels ils vont devoir fonctionner (PDA (Personnel Digital Assistant), Internet et téléphones mobiles, ...). De plus, leur technologie fait appel à une électronique et à des applications où l'on doit minimiser aussi bien l'encombrement que la consommation électrique. Du fait de leur portabilité et de la mobilité des produits dans lesquels ils sont incorporés, les systèmes embarqués évoluent généralement dans des conditions environnementales non déterministes et souvent non maîtrisées. Ils sont donc exposés à des variations et autres contraintes environnementales susceptibles d'induire des défaillances : vibrations, chocs, variations de température et de

tensions d'alimentation, corrosion, humidité,...

Bien que leur sophistication ne cesse de s'accroître, les systèmes embarqués sont utilisés dans des applications de plus en plus critiques dans lesquelles leur dysfonctionnement peut générer des nuisances, des pertes économiques ou des conséquences inacceptables pouvant aller jusqu'à la perte de vies humaines. C'est le cas, notamment, des applications médicales ou dans les transports pour lesquelles une défaillance peut avoir un impact direct sur la vie des utilisateurs. C'est aussi le cas des applications spatiales, souterraines ou sous-marines où un défaut de fonctionnement peut entraîner des conséquences redoutables aussi bien en termes de sécurité qu'au niveau économique. Dans ces applications, le principal facteur pris en compte est alors la fiabilité. I1 est en effet indispensable d'assurer immédiatement la détection des erreurs causées par une panne dans ces systèmes, ceci afin d'intervenir rapidement et d'éviter une évolution catastrophique de la situation. Autrement dit, il faut appliquer des tests en ligne pour confirmer la validité de fonctionnement de ces systèmes, en vérifiant de manière continue les résultats produits. Ceci exige la recherche de solutions ciblées et, par conséquent, le développement de stratégies qui permettent de vérifier sans cesse l'intégrité de ces systèmes sans pour autant ralentir ou perturber leur fonctionnement normal.

Grossièrement, les techniques de test en ligne efficaces s'appuient généralement sur des stratégies qui n'affectent pas les performances de l'application en cours d'exécution. Les techniques de test en ligne idéales doivent avoir une couverture de faute de 100%, un cycle d'horloge de latence d'erreur, aucune redondance matérielle ni redondance temporelle. Au global, ces techniques n'exigent donc aucune modification majeure de la conception originale et n'imposent aucune restriction de type structurel ou fonctionnel sur le CUT (Circuit Under Test) [3]. En fait, une large gamme de techniques de test en ligne développées dans le passé continue à s'enrichir grâce à de nouvelles conceptions. On note néanmoins que la majorité des méthodes BIST (Built-In Self-Test) courantes répondent à certaines de ces contraintes sans pouvoir toutefois les adresser toutes à la fois [3,4].

Parmi ces contraintes, la réduction de la latence d'erreur est considérée, souvent, comme principal objectif dans le développement des techniques de test en ligne dédiées aux applications critiques [5]. En fait, les techniques de duplication [6-9] sont souvent adoptées car elles ont une latence d'erreur très faible par rapport à tant d'autres techniques proposées dans la littérature. Par ailleurs, la disponibilité des COTS à faible coût et leur facilité d'intégration dans les SOC (System On Chip) actuels font de la duplication matérielle une solution viable dans de nombreux domaines où la fiabilité du système est une préoccupation majeure, même au détriment du surcoût en surface nécessaire à son

implémentation [10].

Cependant, dans les SOC modernes, de nombreux cœurs sont intégrés dans une seule puce. Certains d'entre eux sont enfouis et ne sont pas accessibles directement depuis l'extérieur de la puce. Le test de ces cœurs intégrés est devenu le goulot d'étranglement dans le développement des SOC. Quoique la technique BIST est l'une des solutions de test la plus populaire adoptée pour tester les cœurs enfouis [11], elle est utilisée pour réaliser un test en ligne discontinu en activant, périodiquement, les séquences de test nécessaires. La détection de la panne est alors effectuée avec un retard par rapport à son apparition qui est liée à la périodicité du test. En outre, les techniques de test traditionnelles qui utilisent la génération automatique de vecteurs de test pour cibler les défauts simples lors du test des circuits numériques sont devenues très coûteuses et ne peuvent plus assurer une couverture de fautes suffisamment élevée. Elles ne sont pas en fait adaptées au test des systèmes futurs et présentent ainsi des limites.

Par ailleurs, la mesure de grandeurs électriques, tension ou courant, utilisant des méthodes optiques est un domaine en plein essor. Ces technologies optiques offrent de nombreux avantages et permettent, dans certaines situations, de résoudre les difficultés liées aux techniques électroniques modernes [12,13]. Ainsi, les techniques de sondage Électro-Optiques (EO) de circuits électroniques à l'aide d'un faisceau laser sont aujourd'hui des techniques développées par différentes équipes de recherche [14-17]. Une nouvelle application du sondage EO a, par exemple, été présentée dans [18-20] dont le but était de réaliser la mesure en temps réel des états haut et bas des signaux numériques d'entrées/sorties d'un circuit intégré afin de détecter toute défaillance possible durant le fonctionnement normal. Cette nouvelle application de sondage EO et l'exploitation des méthodes BIST sont des éléments clés utilisés dans l'élaboration d'une nouvelle approche pour permettre la réalisation de tests en ligne des circuits modernes avec une latence de détection d'erreur très faible.

L'idée de base développée dans cette thèse consiste à combiner la technique de duplication matérielle qui ne nécessite aucune génération de vecteurs de test et l'exploitation de l'architecture BIST (avec l'ajout de sondes optiques) pour accéder aux cœurs enfouis. Le principal objectif considéré est de réduire considérablement la latence de détection d'erreur afin d'assurer un fonctionnement fiable des dispositifs COTS. En d'autres termes, ce travail présente une approche de test en ligne pour les systèmes embarqués basée sur une approche BIST dans laquelle des sondes optiques sont ajoutées à l'architecture classique BILBO (Built-In Logic-Block Observation). De ce fait, de nouveaux registres Optiques BILBO (OBILBO) sont proposés. Nous proposons ainsi de tester optiquement et in situ

un système embarqué en cours d'utilisation, sans pour autant perturber ou ralentir son bon fonctionnement. La méthode que nous avons développée exploite les faisceaux optiques fournis par le registre OBILBO correspondant aux données présentes sur les sorties numériques de deux applications identiques. Il devient alors possible d'envoyer, en temps réel, ces signaux optiques à un système distant équipé de capteurs optiques et de convertisseurs A/N, et de les comparer pour détecter la présence éventuelle d'erreurs. De plus, notre approche peut être utilisée en mode hors ligne en examinant la signature optique du registre OBILBO.

Ce document s'articule autour de quatre chapitres. Le premier chapitre introduit les concepts de base du test numérique. Il aborde la problématique du test des systèmes intégrés et présente les nouveaux défis et exigences de test des systèmes sur les puces modernes.

Le deuxième chapitre, quant à lui, présente un état de l'art sur les techniques de test en ligne. Après la mise en évidence de la nécessité de ce genre de test dans les applications critiques, il expose en détails les concepts relatifs au test en ligne et les différentes méthodologies existantes.

Les chapitres suivants constituent une présentation détaillée du travail effectué au cours de cette thèse et des résultats que nous avons obtenus. Le troisième chapitre propose ainsi une nouvelle approche de test en ligne des systèmes embarqués. Le schéma de principe proposé à cet effet est explicité en détails, de même que la stratégie de test mise en œuvre.

Afin de valider et démontrer l'efficacité de notre approche, une modélisation du schéma proposé en langage VHDL-AMS est décrite dans le quatrième chapitre. Les résultats de simulation sont présentés et interprétés afin de confirmer nos attentes.

À l'issue de ce quatrième chapitre, nous présentons une conclusion générale sur nos travaux décrits dans ce document et la technique de test en ligne des systèmes embarqués que nous avons développée, ainsi que des perspectives sur les suites à donner à ces travaux.

Chapitre 1

CONCEPTS DE BASE DU TEST NUMÉRIQUE

Résumé *: Dans ce chapitre, les notions de base du test numérique des circuits intégrés sont présentées. Les différentes phases de test de ces derniers sont brièvement décrites. La fonction de coût du test des circuits est évaluée. Ensuite, la nécessité de la conception en vue de test est évoquée. Enfin, les exigences et défis du test des systèmes sur puce sont résumés.*

1.1 Introduction

Ce n'est un secret pour personne, la vitesse de développement en électronique va très vite. En effet, tout le monde aura remarqué ces dernières années l'inéluctable course à la miniaturisation de tout appareil électronique. Plus petit, plus de fonctionnalités, plus rapide, plus d'autonomie... Ceci est dû essentiellement à la demande croissante de mobilité et des usages de plus en plus ciblés qui sont devenus aujourd'hui les enjeux forts du marché de l'électronique. D'autres part, l'industrie des semi-conducteurs a continué à améliorer la vitesse des circuits tout en diminuant la consommation énergétique, et pour cela poursuivre la miniaturisation des transistors, en proposant ces performances accrues à coût compétitif.

En effet, un circuit intégré peut contenir, aujourd'hui, quelques dizaines de millions de transistors, voire quelques centaines de millions, ceci étant sans commune mesure avec les dizaines de milliers d'il y a quelques années. Et cette tendance n'est pas prête de s'arrêter, au vu de l'étude délivrée par la SIA (*Semiconductor Industry Association*) [21,22]. À titre d'exemple, les microprocesseurs actuellement proposés par Intel comptent aujourd'hui quelques huit cent millions de transistors par centimètre carré [23].

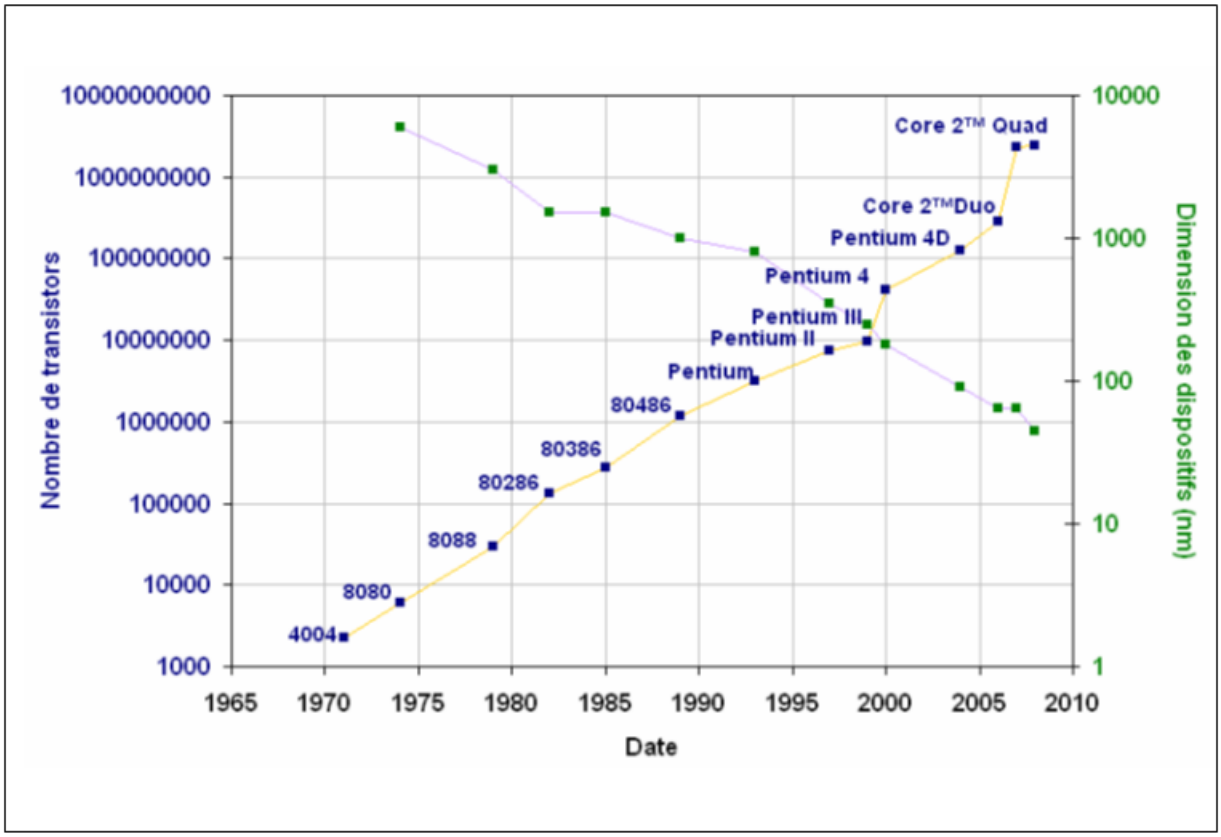


FIGURE 1.1 – Représentation de la loi de Moore avec l'exemple des µP d'Intel [25].

Ainsi, en 1965, Gordon Moore, cofondateur d'Intel prédisait que le nombre de transistors doublerait tous les deux ans dans les processeurs. Quarante ans plus tard, sa prédiction s'avère toujours exacte (Figure 1.1). Les processeurs ne cessent d'augmenter leur vitesse de traitement. Intel vient de commercialiser ses *Duo-Core* et déjà son *Quad-Core* est disponible. Jusqu'à tout récemment les processeurs étaient composés d'un seul cœur ("*Core*"), c'est-à-dire d'une seule unité de calcul avec un seul traitement à la fois. Le *Duo-core* quant à lui possède deux unités de calcul et peut donc traiter deux fonctions en parallèle. IBM a été le premier à commercialiser le processeur à deux cœurs en 2001, appelé le Power 4, ce processeur possédait 174 millions de transistors. Intel a introduit à la fin de 2006 son tout nouveau *Quad-Core*, un nouveau processeur contenant quatre cœurs. Ce processeur contient deux *Duo-core* avec une vitesse de 2,40 GHz [24].

Cependant, cette évolution exponentielle du nombre de transistors sur un même circuit intégré, a été rendue possible grâce à la diminution de la taille des dispositifs microélectroniques. Depuis la fin des années 90, une feuille de route internationale, appelée ITRS (*International Technology Roadmap for Semiconductor*) [26], est éditée par des experts afin de compléter la loi de Moore et donner un objectif global en termes de dimension,

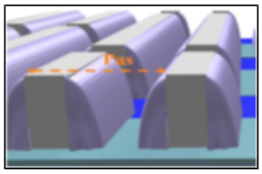
Année de production	2006	2007	2008	2009	2010	2012	2013	
DRAM ½ pas (nm)	68	59	52	45	40	36	32	
Flash ½ pas (nm)	54	45	40	36	32	28	25	
MPU longueur de la Grille (nm)	25	23	20	18	16	14	13	

FIGURE 1.2 – Extrait de l'ITRS 2008 présentant les objectifs dimensionnels de plusieurs architectures CMOS : Mémoire DRAM, Mémoire Flash. Schéma représentant le pas d'une structure type DRAM [25].

performance et coût des systèmes microélectroniques pour chaque nœud technologique. Le nœud technologique est défini comme la moitié du pas (ligne + espace pour les structures de type DRAM de la génération de composants concernée), ou *Pitch* en anglais. Le tableau de la figure 1.2 en est un extrait [26].

On atteindra ainsi, en 2013, le nœud technologique *32nm* pour lequel la longueur de grille des transistors composant les microprocesseurs (MPU), sera de *13nm* (dispositif le plus agressif en terme de dimension pour des besoins de rapidité de fonctionnement).

Toujours dans cette optique, les progrès réalisés en CAO et le besoin toujours croissant de systèmes de plus en plus performants ont conduit à la conception de circuits de plus en plus complexes. Désormais il est possible d'intégrer un système complet sur une seule puce, système qui jusque-là tenait sur une carte ou sur un MCM (*Multi Chip Module*). En effet, des systèmes complets intégrés sur silicium sont ainsi aujourd'hui devenus une réalité, et de plus en plus de constructeurs ont recours à cette approche, pour les avantages qu'elle offre [21,27]. Ces systèmes intégrés sur une puce, appelés SOC (*System On Chip*), ou SIP (*System In Package*), se développent inévitablement et se retrouvent dans des domaines variés comme les télécommunications, l'avionique, la construction automobile, etc. Ils sont composés de blocs numériques (ex. DSP : Digital Processing Unit), analogiques et mixtes (ex. convertisseurs numériques/analogiques et analogiques-numériques) et des circuits RF (ex. amplificateurs, mixeur, filtres). Ces puces sont utilisées dans différentes applications telles que les équipements biomédicaux, les téléphones mobiles, les communications Wifi, Bluetooth etc. En fait, une fois que la conception de ces circuits est validée, ils sont envoyés à la production grand volume pour en fabriquer des milliers d'échantillons. Or, pendant l'étape de fabrication dans les salles blanches, des défauts catastrophiques peuvent se produire (par exemple un court-circuit entre deux lignes métalliques parallèles ou entre les deux bornes d'un composant, un circuit-ouvert qui coupe la ligne métallique reliant

deux composants etc.). Ces types de défauts rendent le circuit défectueux, d'où la nécessité d'une étape qui suit la fabrication. Cette étape consiste à tester tous les circuits fabriqués afin de valider leur fonctionnalité ainsi que leurs spécifications prédéfinies ; c'est ce qu'on appelle le test de production. De plus, certains systèmes, fonctionnant dans des domaines d'applications critiques où la sécurité est le facteur principal pris en compte, ne tolèrent pas l'interruption de leur fonctionnement normal. Ceci exige une vérification, de manière continue, de l'intégrité de ces systèmes en cours d'exploitation sans interrompre ou ralentir leur fonctionnement normal.

Grossièrement, l'objectif général du test est d'assurer que le composant électronique mis sur le marché donnera satisfaction au client en termes de fonctionnalités et de fiabilité. Cependant, les dernières avancées technologiques au niveau des densités d'intégration, les progrès de la CAO et les besoins du marché, conduisent à une augmentation importante de la part du test dans le coût de revient du circuit. En effet, de nos jours, le coût de test d'un circuit intégré est devenu, d'une manière générale, de plus en plus important au point même de surpasser celui de la conception du circuit lui-même [28]. Les sections suivantes décrivent brièvement les enjeux du test numérique des circuits intégrés.

1.2 Notions de base du test numérique

Le test d'un circuit digital consiste à mettre en évidence son éventuel mauvais fonctionnement dû à une défaillance physique et/ou erreur de conception. Cette mise en évidence potentielle s'effectue par deux tests distincts [29]. Le premier consiste à effectuer des mesures paramétriques (courant, tension, temps de montée et de descente,...), ce test *paramétrique* fait intervenir des techniques de métrologie particulières. Le second consiste à inspecter les données logiques manipulées par le circuit et les comparer avec les valeurs attendues. C'est ce test *logique* qui sera abordé dans ce document.

En général, le test logique est un processus qui consiste à appliquer un ensemble de vecteurs de test aux entrées primaire (*Primary Inputs*) d'un circuit et vérifier (observer) si les sorties primaires (*Primary Outputs*) correspondent à la fonction attendue du circuit, dans ce cas ces sorties sont appelées "*Fault-Free Outputs*". Dans le cas contraire ces dernières sont appelées "*Faulty Outputs*" et le circuit présente, par conséquent, des fonctionnements anormaux. La présence de défauts est à l'origine du mauvais fonctionnement d'un circuit [30-33].

Un comportement inexact du circuit, c'est-à-dire non conforme au comportement attendu, est défini par l'ensemble des termes suivants : défauts, erreurs et fautes. Les définitions sont les suivantes [34].

Définition 1.1. Un *défaut* (aussi appelé *défaillance*) est la différence, non voulue, entre l'implantation réelle et l'implantation désirée du système.

Les défauts dans les circuits VLSI ("*Very Large Scale Integration*") sont typiquement les suivants [35] :

- ❒ défauts du procédé de fabrication : rupture d'oxyde, transistor parasite,...
- ❒ défauts du matériau : impureté du matériau,...
- ❒ défauts du boîtier après encapsulation : dégradation des contacts, étanchéité du boîtier,...

Les défauts de fabrication et les pannes physiques (survenant pendant la vie du circuit et non présent en sortie de chaîne de fabrication) sont appelés fautes physiques. Les fautes physiques sont observées par la manifestation d'erreurs.

Définition 1.2. Une *erreur* est la manifestation observable d'un défaut. Elle peut se produire en présence de certains stimuli.

Puisque ces fautes physiques ne sont pas facilement manipulables de façon automatisée, la solution est de traiter les fautes logiques.

Définition 1.3. Une *faute* correspond à la représentation d'un défaut ou d'une panne suivant un modèle donné.

Une faute est détectée en observant l'erreur qu'elle produit. La modélisation permet l'utilisation d'outils de simulation de fautes et génération automatique de vecteurs de test ATPG (*Automatic Test Pattern Generation*) [27,34].

Plusieurs modèles de faute ont été proposés [30-33] pour refléter les défauts et les pannes. Le modèle le plus largement utilisé est celui d'une ligne "*collée*" de façon permanente à une valeur logique 0 ou 1 (c'est-à-dire collage à 0 ("*stuck-at-0*") ou collage à 1 ("*stuck-at-1*")).

Notons que d'autres modèles de fautes peuvent être pris en compte. La diminution de la largeur des transistors, des lignes d'interconnexions et de l'espacement entre ces lignes ont permis une densité d'intégration plus importante. Cependant, ces progrès technologiques font apparaître des défaillances spécifiques. Ainsi, les circuits actuels sont sensibles à des défauts temporels (faute de délai). Ceci s'explique, d'une part, par les fréquences de fonctionnement élevées et d'autre part, par la prépondérance prise par les interconnexions sur les performances des circuits. Signalons que l'amélioration de la représentativité des fautes physiques par nouveaux de modèles ne remet pas en cause la validité du modèle de collage [27].

Les séquences de test appliquées aux circuits sont validées vis-à-vis d'un modèle de faute. On utilise pour cela la notion de taux de couverture qui est la proportion de fautes détectées par une séquence sur le nombre total de fautes dans le modèle considéré [30].

1.3 Les différentes phases de test d'un circuit

Les tests sont naturellement effectués à différents stades de la vie d'un circuit. Les puces sont testées sur les plaquettes ("*Wafers*") durant la fabrication, les circuits en boîtiers sont testés après encapsulation par le fabricant et avant l'insertion dans une carte par l'équipementier, les cartes le sont avant l'assemblage dans le système et enfin, le système lorsqu'il est terminé. Les tests appliqués ne sont pas les mêmes ; ils dépendent de l'étape où ils sont effectués. N'ayant pas la même finalité, ils s'appliquent dans des conditions tout à fait différentes [36]. Le tableau de la figure 1.3 donne les différents types de test qui peuvent être effectués suivant le moment où ils s'appliquent [29].

Chaque *Wafer* comprend un grand nombre de circuits intégrés. Le premier test effectué concerne le processus technologique. Pour cela, on utilise des motifs de test spécialement conçus qui sont placés soit entre les circuits intégrés (et dans ce cas ils sont détruits lors de la découpe), soit dans un circuit particulier, ou encore directement à l'intérieur de l'enveloppe de la chaîne des circuits intégrés. Ce test s'effectue par l'intermédiaire de machines à pointes, sur le wafer, afin de mesurer les paramètres technologiques du processus. Le deuxième test consiste à tester individuellement chaque circuit intégré directement sur le wafer toujours par l'intermédiaire d'une machine à pointes. Dans ce cas, on effectue à la fois des tests paramétriques et des tests logiques et les circuits défectueux sont marqués afin de ne pas être encapsulés.

Après encapsulation des circuits non marqués, les boîtiers résultants sont intégrés grâce à un testeur spécialisé. Les tests appliqués sont encore du type paramétrique et logique. Afin de se prémunir des défaillances de jeunesse courantes dans les dispositifs électroniques (courbes en "baignoire" de la fiabilité des dispositifs électroniques), les boîtiers sont soumis à des cycles de température ("*Burn in*") afin d'accélérer le processus de vieillissement et l'apparition de défaillances éventuelles. Enfin, certains boîtiers sont choisis comme échantillons afin d'être soumis à des tests beaucoup plus poussés qui permettent de s'assurer de la qualité du processus global de fabrication [36].

Ces circuits sont alors fournis aux utilisateurs qui, en général, leur font subir un test d'entrée avant de les intégrer dans les plaques. Ces plaques (*PCB*) sont d'abord testées sans composant, en particulier par des tests de continuité électrique. Elles sont ensuite testées toutes montées par l'intermédiaire du connecteur ou encore directement au niveau

DISPOSITIF SOUS TEST			
Wafer	**Boîtier**	**Plaque**	**Système**
▪ Test du processus technologique ▪ Test des puces	▪ Test paramétrique ▪ Test logique ▪ Vieillissement ▪ Test complet sur échantillon	▪ Test d'entrée ▪ Test de plaque nue ▪ Test de la plaque montée ▪ Vieillissement	▪ Test du système ▪ Test en utilisation

FIGURE 1.3 – Différents types de test.

des circuits en utilisant des sondes spéciales. Les plaques sont aussi soumises à des cycles de température.

En dernier lieu, le système complet est testé en général en lui faisant effectuer ses programmes d'application spécifiques. Ainsi, un ordinateur sera testé avant d'être livré en lui faisant exécuter son système opérationnel et un certain nombre d'applications permettant d'activer l'intégralité de ses ressources. Un certain nombre de programmes de test sont prévus pour tester le système en utilisation [29,36]. Ces programmes qui permettent de mettre en évidence la partie défaillante du système sont plus des programmes de diagnostic, les parties défaillantes étant alors remplacées pour être transmises au service de réparation.

1.4 Difficultés et coût de test d'un circuit

Avec l'augmentation constante de la densité d'intégration des circuits, on atteint désormais des chiffres avoisinant le milliard de transistors dans un circuit. Or, plus il y a d'éléments à tester dans un circuit, plus la liste de fautes à vérifier dans ce circuit s'allonge, et donc plus il y a de vecteurs de test. D'autre part, plus le circuit est complexe, plus il est difficile de sensibiliser certaines fautes enfouies au cœur du circuit, et, là encore, plus de vecteurs de test peuvent être nécessaires. Tous ces vecteurs additionnels se traduisent par une augmentation de la mémoire du testeur nécessaire au stockage de ces vecteurs (coût matériel relatif au testeur) ainsi que par une augmentation du temps d'application de ces vecteurs au circuit (coût en temps de production). De plus, la rapidité d'évolution de ces circuits nécessiterait une évolution similaire dans le développement des appareils de test. Or, il est impossible de le renouveler à chaque fois que la technologie des circuits intégrés progresse, un testeur coûtant plusieurs centaines de milliers d'euros [37]. Il est toutefois possible d'augmenter la profondeur mémoire d'un testeur ou bien son nombre de canaux

maximum mais cela aussi a un coût très élevé.

Par ailleurs, Pat Gelsinger commente, dans l'article [38], l'évolution de la précision des testeurs par rapport aux vitesses de fonctionnement des circuits. Les testeurs externes du début des années 80 avaient une résolution excédentaire par rapport aux exigences du composant testé. À cette époque une perte de rendement faible était due à la marge de tolérance du testeur. En une vingtaine d'années, alors que la vitesse des CUT a été multipliée par cent, la précision des testeurs n'a été accrue que d'un facteur dix. La projection de cette tendance montre que d'ici quelques années une perte de rendement inacceptable sera due seulement à l'imprécision des testeurs [39].

1.4.1 Qualité du test et fiabilité des circuits intégrés

La qualité de test dépend de la stratégie adoptée et de la technique mise en œuvre pour accomplir des tâches assurant la génération des tests et le prélèvement des résultats. Cependant, l'accomplissement de ces tâches dépend énormément de la possibilité d'accès offerte pour la réalisation de contacts avec les nœuds internes du système sous test. L'accessibilité aux circuits actuels est difficile voire impossible. Il faut donc prendre en compte d'une part, le coût du développement des tests et le coût de leur mise en œuvre qui dépendent surtout des quantités produites et, d'autre part, les coûts entraînés par les produits défectueux non repérés lors des tests. Les tests doivent être mis en œuvre en parallèle avec la conception (*"Concurrent Testing"*).

Cependant, la qualité du test a une influence directe sur la fiabilité des circuits intégrés fabriqués. C'est pour cela qu'il est indispensable d'introduire des métriques qui permettent d'évaluer la qualité de test de ces derniers. Ces métriques sont très utiles aussi bien pour le constructeur dans l'évaluation du test de production que pour le concepteur dans l'évaluation de son technique de test. Dans le cas des fautes simples et des circuits numériques le paramètre le plus utilisé est la couverture de fautes F qui désigne la probabilité de détection des circuits avec faute. Cette probabilité est estimée comme suit :

$$F = \frac{Le\,nombre\,de\,fautes\,qui\,sont\,détectées}{Le\,nombre\,total\,de\,fautes} \tag{1.1}$$

Si on désigne par Y (*Yield*) le rendement de production de circuits corrects, par T l'efficacité des tests, le taux de produits défectueux DL (*Defect Level*) non détectés sera estimé par la relation (1 .2), définie pour la première fois par *Williams* et *Brown* [40], où les fautes ont été considérées comme équiprobables [41,42] :

$$DL = 1 - Y^{(1-T)} \tag{1.2}$$

Et inversement :

$$1 - T = \frac{\log(1 - DL)}{\log(Y)} \tag{1.3}$$

Soit n le nombre de soudures ou contacts sur une carte, par exemple, et P la probabilité pour que chacune d'entre elles soit en bon état. La probabilité pour que l'ensemble soit en bon état est donnée par :

$$Y = P^n \tag{1.4}$$

Si les tests vérifient m contacts parmi les n, l'efficacité des tests est :

$$T = \frac{m}{n} \tag{1.5}$$

La probabilité que les (n-m) contacts restants soient bons vaut : $P^{(n-m)}$. La probabilité de défaillance s'écrit [41] :

$$DL = 1 - P^{n-m} = 1 - Y^{\frac{(n-m)}{n}} = 1 - Y^{(1-T)} \tag{1.6}$$

Par exemple, avec $DL = 2\%$ et $Y = 20\%$, il faut tester à 98,7%, ce qui est relativement coûteux en vecteurs de test et de plus, difficile à évaluer ; les tests devant eux-mêmes être testés à l'aide des programmes de vérification de la qualité des tests.

1.4.2 Coût du test d'un circuit intégré

Les différentes étapes de fabrication des circuits intégrés sont très complexes. Pendant leur fabrication, des défauts dus aux procédés de fabrication, aux matériaux, ou introduits lors de l'encapsulation dans le boîtier peuvent survenir. Ces défauts sont observés par la manifestation d'une erreur. Les erreurs peuvent se produire en présence de certains stimuli. Or, plus un défaut est détecté tôt dans le processus de fabrication, moins le coût qu'il va induire est élevé puisque, au niveau de leur coût, les tests suivent la règle de dix ($\times 10$).

En effet, la loi ($\times 10$) représente bien l'évolution du coût des tests au cours de l'élaboration d'un produit et en conséquence l'intérêt d'un dépistage précoce des défauts (Figure 1.4). Le coût du test d'une carte peut atteindre 50% de sa valeur [41].

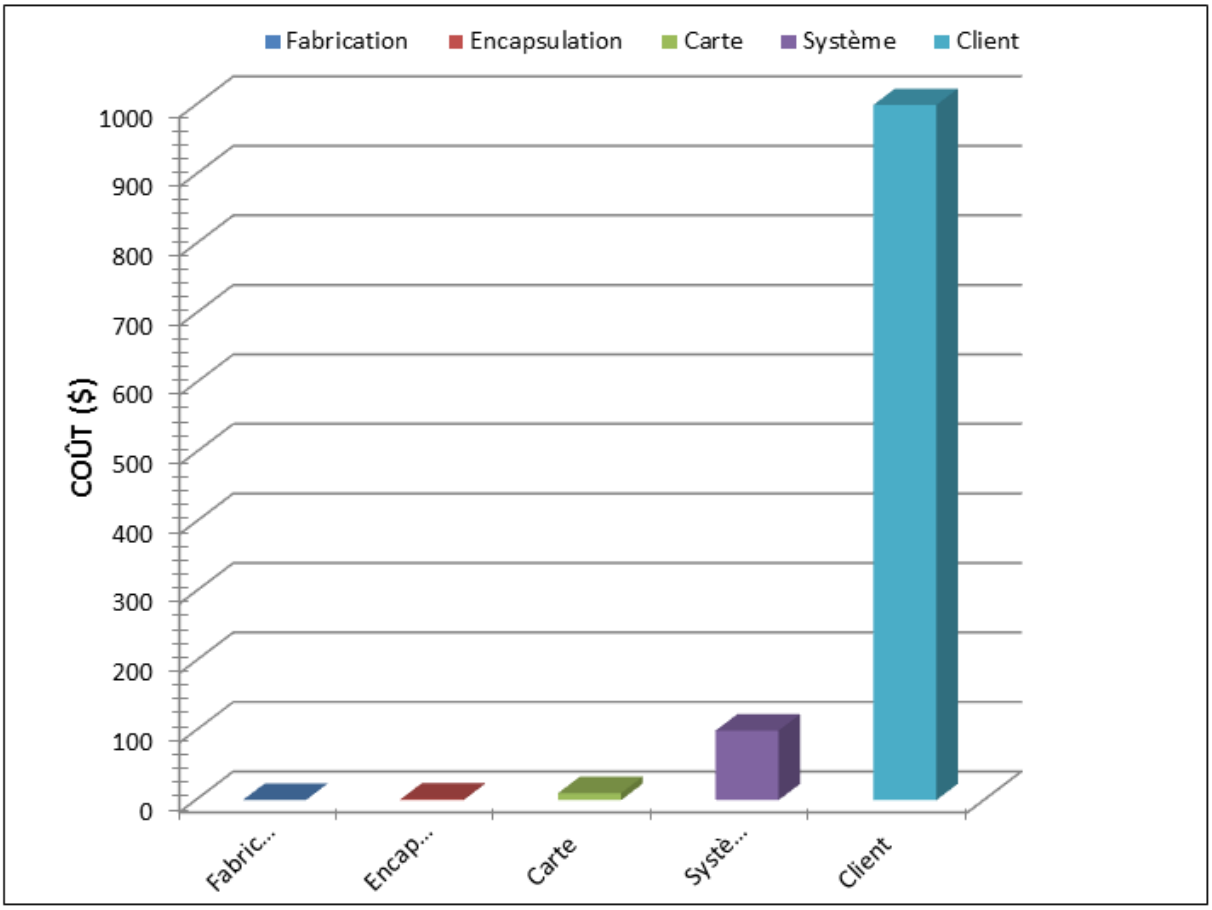


FIGURE 1.4 – Coûts de détection des défauts (règle de ($\times$10)).

Les problèmes de coût imposent donc des contraintes fortes : un matériel de test en production vaut entre quelques centaines de *KFF* et quelques dizaines de *MFF* et ne doit pas être immobilisé plus de quelques secondes sur une carte imprimée. Ceci implique un rythme de génération des vecteurs de test très rapide : une mémoire 4 kbits nécessiterait $(2^{4096} - 1)$, soit 10^{1232}, séquences de test c'est-à-dire 10^{1222} ***siècles*** avec une séquence toutes les ***ns*** [41]. Les mémoires actuelles atteignent plus de 4 Gb !

Les coûts de test, qu'ils soient matériels ou relatifs au temps du test, ont un impact direct sur le coût final des circuits et sur les marges des entreprises de fabrication de circuits sur silicium. Le test d'un circuit ou d'une carte ou encore d'un système a donc un enjeu économique important. Il est donc nécessaire d'établir un compromis entre la qualité de test requise et le coût acceptable dans les limites raisonnables. Il faudra alors penser à introduire l'aspect test dès les premières phases de conception des circuits. Autrement dit, il faudra développer des architecturées conçues en vue d'une meilleure testabilité. Cette notion fera l'objet de la prochaine section.

1.5 Conception en vue de test

La conception en vue d'une meilleure testabilité (*Design For Testability*), regroupe toutes les méthodes permettant d'améliorer, et de diminuer le coût, du test des circuits intégrés. Ces techniques tendent toutes à améliorer l'application des séquences de test citées précédemment (contrôlabilité) et leur observabilité [30,43-45].

Les premières techniques appelées techniques Ad-hoc, consistaient principalement en un respect de règles de conception particulières. Les autres techniques, appelées techniques "*Scan Path*" consistent à ramener le problème du test d'un circuit séquentiel à celui de plusieurs blocs combinatoires déconnectés et de taille plus réduite [46].

1.5.1 Méthode Ad-hoc

Ces techniques consistent en une liste de techniques permettant de résoudre les problèmes rencontrés au niveau du test du circuit. Elles se traduisent généralement par une augmentation du nombre d'entrées de contrôle et de sorties d'observation et le partitionnement du circuit. Ces techniques n'ont pas de portée générale et sont donc liées à la structure du circuit [30,46].

1.5.2 Méthode Scan Path

Dans la méthode "*Scan Path*" [47], le circuit possède deux modes de fonctionnement. Le mode de fonctionnement normal et un mode de test où tous les nœuds internes de mémorisation (bascules) sont interconnectés en formant un registre à décalage [48,49]. Le mode normal permet ensuite de transmettre ces vecteurs sur les blocs combinatoires, qui modifient alors le contenu du registre. Les résultats complets peuvent alors être extraits du circuit en mode série puis comparés à la réponse correcte.

1.5.3 Méthode LSSD

La méthode LSSD (*Levet Sensitive Scan Design*) [49,50] utilisée industriellement par la compagnie IBM, reprend le principe du "*Scan Path*" en utilisant notamment des bascules réagissant sur les niveaux. Cette méthode est moins sensible aux problèmes de délais internes du circuit.

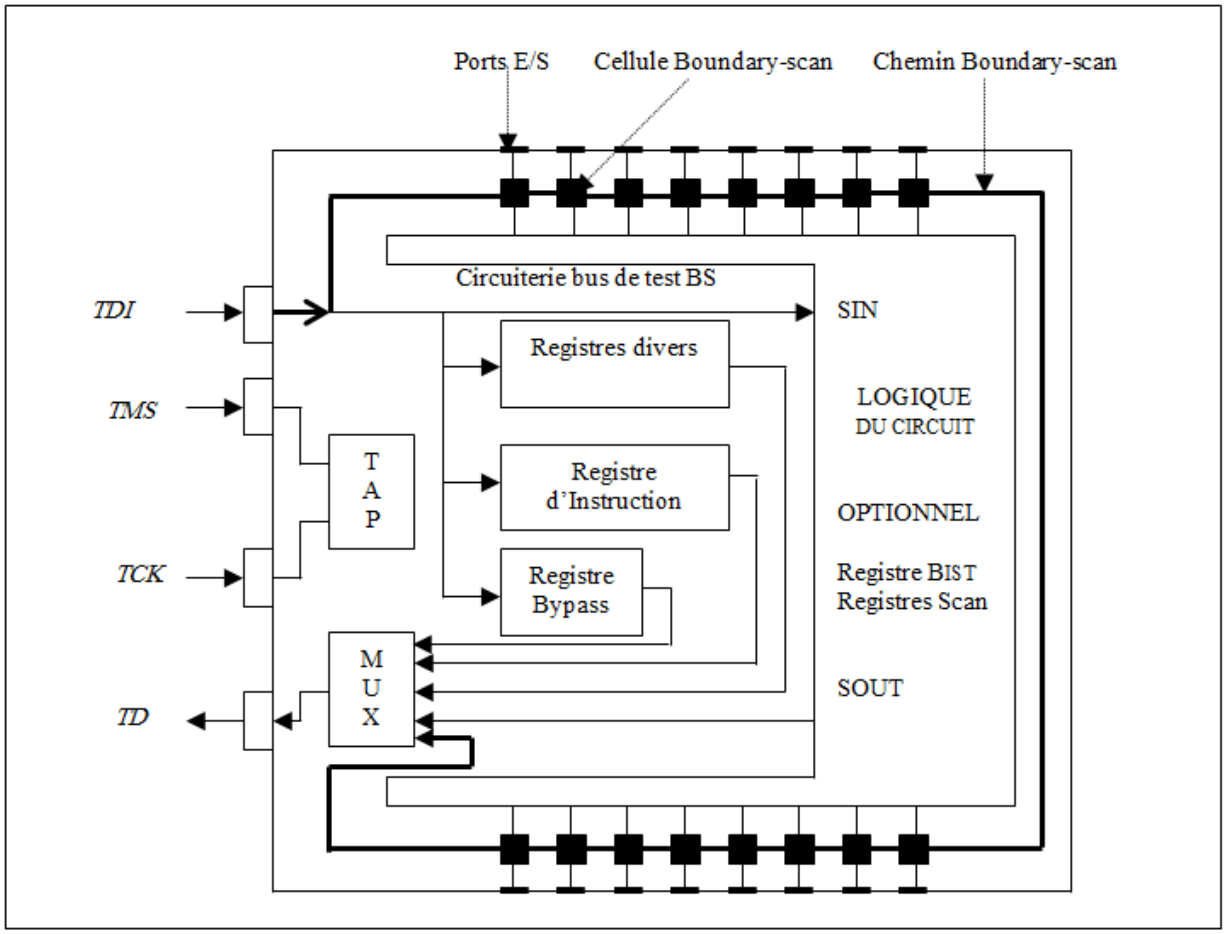


FIGURE 1.5 – Architecture conceptuelle du BS (Norme IEEE 1149.1).

1.5.4 Méthode Boundary Scan

La technique Boundary Scan (BS) est apparue à la fin des années 80. C'est une extension des techniques de Scan du niveau circuit vers le niveau carte. Face à la miniaturisation des cartes et des composants et à l'apparition des techniques de montage en surface SMT (*Surface Mounted Technology*), cette technique s'est imposée peu à peu comme standard. Elle permet l'accès sériel des vecteurs de test aux entrées/sorties de tous les circuits composants la carte. Outre le test de production, elle permet aussi d'effectuer le test de maintenance [51].

Nous proposons dans ce qui suit une très brève description de ce standard [52], cette norme (IEEE 1149.1) étant désormais bien connue dans le monde de la microélectronique.

Un composant à la norme Boundary Scan [30] intègre plusieurs éléments dédiés au test, il est principalement constitué (Figure 1.5) :

❒ d'un port d'accès au test, le TAP (*Test Access Port*), constitué de quatre signaux

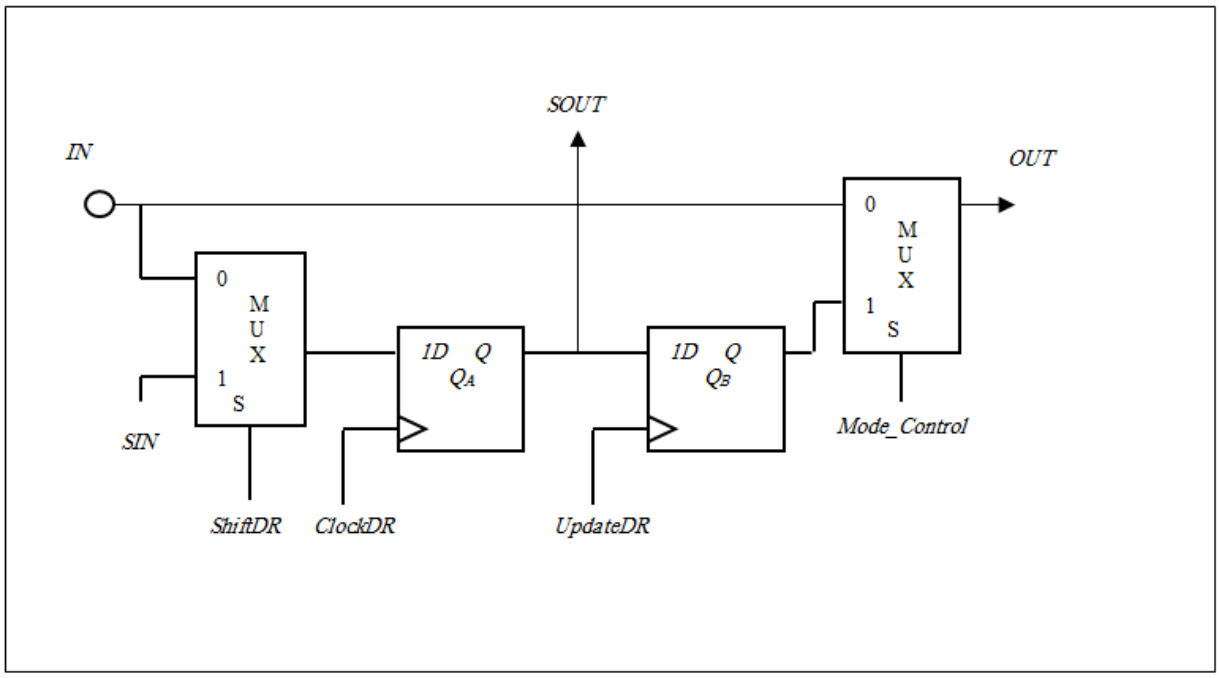


FIGURE 1.6 – Exemple d'implémentation d'une cellule BS.

obligatoires (TDI, TDO, TMS et TCK) et d'un signal optionnel TRST ;

- ❒ de registres de données. Il s'agit des registres d'identification, de *Bypass*, du registre Boundary Scan, du registre de Scan interne du circuit et du (ou des) registres utilisateurs ;
- ❒ d'un registre d'instructions ;
- ❒ d'une machine d'état contrôlant l'architecture BS, le TAP Controller.

Le registre BS correspond à la concaténation des cellules BS. Son rôle est de contrôler et d'observer les entrées/sorties du circuit. La cellule BS (Figure 1.6) possède quatre modes de fonctionnement [30,51] :

1. Le mode normal où les données sur l'entrée IN sont directement dirigées vers la sortie OUT. Cela correspond au mode de fonctionnement normal du circuit.
2. Le mode de décalage appelé mode SCAN qui permet le décalage de données dans le registre BS, d'une cellule BS à la suivante, le chaînage s'effectuant grâce aux entrées/sorties de scan SIN et SOUT.
3. Le mode de capture, appelé aussi mode d'échantillonnage, qui permet de stocker la donnée présente sur IN dans la bascule Q_A.
4. Le mode de mise à jour (update) qui permet l'application des données présentes sur $Q_A$vers la sortie OUT.

Le TAP Controller est une machine à états finis comportant seize états, pilotée par le signal TMS (*Test Mode Select*) et l'horloge TCK. La transition d'un état à l'autre dépend de la valeur de TMS [30,52].

La procédure de test d'un circuit BS est basée sur la succession d'actions du type suivant :

- ❒ Initialisation de la logique de test
- ❒ Chargement d'une instruction
- ❒ Application de vecteurs de test
- ❒ Lecture de réponses aux stimuli

Le chargement d'une instruction, la lecture et l'application des vecteurs de test se fait par décalages sur TDI et TDO.

Il faut noter que, trois modes de test au niveau carte sont disponibles : le test externe, le test interne et le test d'échantillonnage. De plus amples informations sont disponibles dans [30,52-56].

1.5.5 Test intégré

Le BIST (*Built In Self Test*) est une des solutions de test qui doivent être pensées lors de la conception du circuit [57,58]. Il s'agit d'intégrer sur le circuit de la logique supplémentaire pour le tester : générateurs de vecteurs de test, analyseurs de réponses, systèmes de distribution pour transmettre les vecteurs de générateur vers le circuit et les réponses du circuit vers l'analyseur, et enfin un contrôleur pour le contrôle de la structure complète durant le mode test. Suivant les besoins et les contraintes, un compromis temps de test/surface additionnelle/taux de couverture peut être envisagé.

Plusieurs chercheurs se sont penchés sur l'intégration du BIST pendant la synthèse du circuit [59,60]. En effet, concevoir pendant la synthèse les parties du circuit qui permettront le test autonome permet d'optimiser la surface ajoutée et de mieux contrôler sa synthèse. Le BIST apparaît comme l'une des solutions au problème du test des systèmes sur puce [61] puisqu'il permette de rendre les noyaux accessibles via les entrées/sorties primaires du système. Dans la section suivante, nous mettons en évidence les exigences et défis du test des systèmes sur puce.

1.6 Exigences et défis du test des systèmes sur puce

Les systèmes sur puce sont de plus en plus complexes car les concepteurs intègrent de plus en plus d'unités de traitement IP (*Intellectual Property*) afin de répondre aux

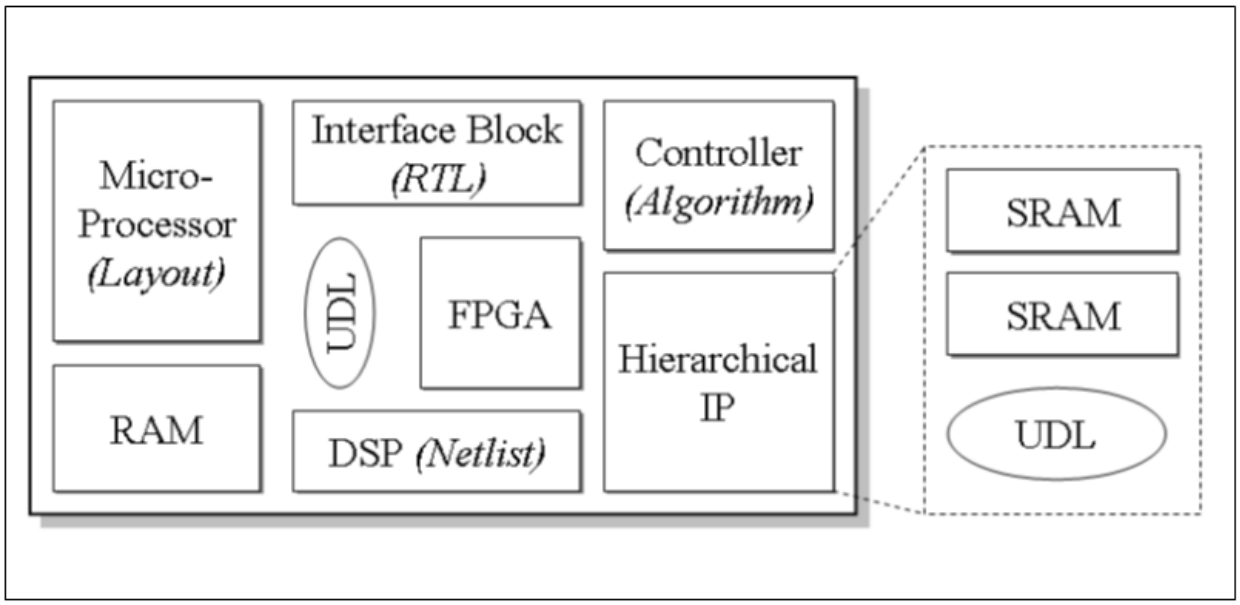


FIGURE 1.7 – Un système sur puce avec plusieurs types d'IP [62].

besoins des applications. Ceci a changé les méthodologies de conception et de test. On a vu apparaître la conception des systèmes sur puce à base d'IP (*Core-based SoC Design*). Avec cette approche, les IP précédemment conçues peuvent être réutilisées pour les conceptions suivantes afin de réduire les temps de conception. Malheureusement, la conception à base d'IP engendre de nouveaux défis pour le test des systèmes. Pour tester ces systèmes, il faut trouver des solutions de test efficaces pour chaque IP et ensuite réaliser le test au niveau système [62].

De plus, pour augmenter la productivité de conception et réduire le temps d'arrivée sur le marché, la réutilisation des IP n'est pas limitée à l'intérieur d'une seule entreprise, les IP peuvent être conçues par d'autres fournisseurs. Comme un système peut inclure plusieurs types d'IP (Figure 1.7), et les IP peuvent aussi inclure d'autres IP. Ces IP doivent être testées en utilisant différentes approches avant que le test du système entier puisse être réalisé. Le test des interconnexions entre les IP doit aussi être pris en compte. Or, plusieurs exigences et défis sont soulevés [63-66] pour le test des systèmes sur puce à base d'IP. Les exigences principales pour ce test peuvent être résumées comme suit :

- ❒ Le test des IP embarquées ;
- ❒ L'accès aux IP pour les tester ;
- ❒ L'isolation d'une ou de plusieurs IP pour les tester en même temps ;
- ❒ Le test des interconnexions entre les IP ;
- ❒ Le test de la logique utilisée pour assembler les IP (*UDL : User-Defined-Logic*).

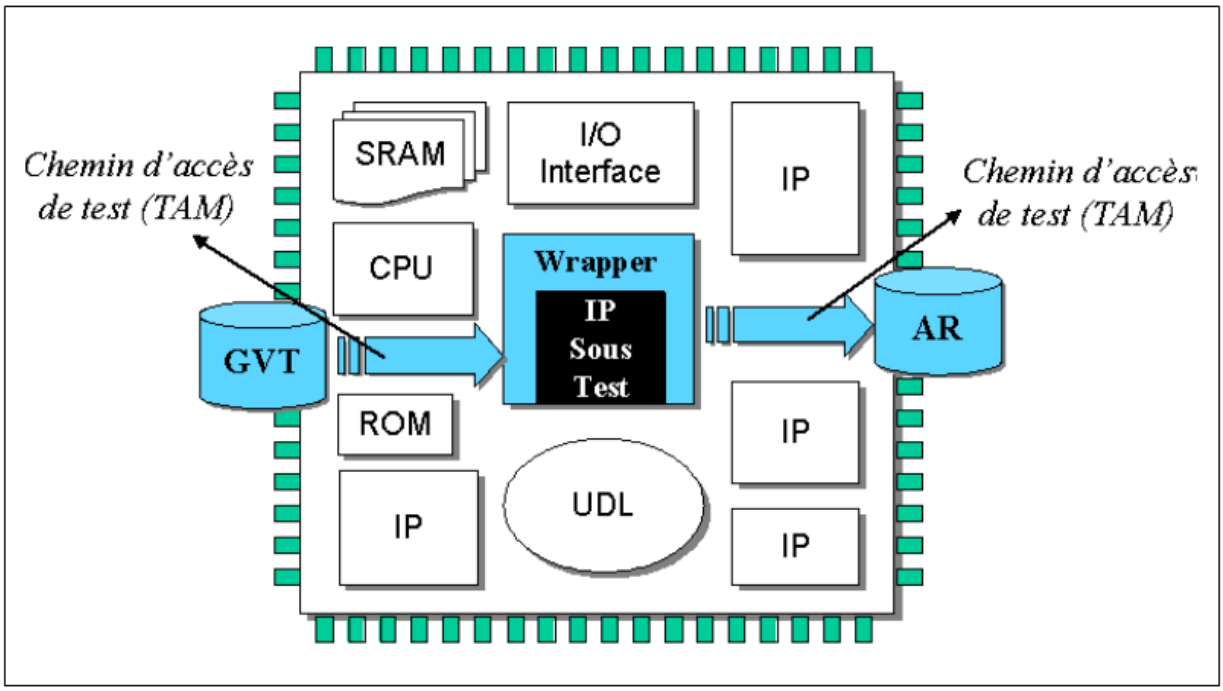


FIGURE 1.8 – Architecture de test des IP [62].

À partir des exigences du test des systèmes sur puce, on peut conclure que le test des IP joue un rôle très important dans le test des systèmes. Dans le paragraphe suivant, nous allons présenter comment réaliser le test d'une IP dans un système sur puce.

1.7 Test des IP dans un système sur puce

Pour tester une IP embarquée dans un système, il est nécessaire de trouver une solution pour accéder aux entrées de l'IP afin d'appliquer les jeux de test et accéder aux sorties de l'IP afin de récupérer les réponses. Zorian [67] a proposé une architecture générique pour tester des IP embarquées, illustrée dans la figure 1.8. Dans cette architecture, les IP embarquées sont entourées par une enveloppe de test, souvent appelée coquille (*Wrapper*) de test, afin d'améliorer la contrôlabilité et l'observabilité de ces IP. On voit qu'il y a trois parties principales dans cette architecture : (1) le générateur de vecteurs de test et l'analyseur de la réponse, (2) le chemin d'accès de test, et (3) le *Wrapper* de test.

Le générateur de vecteurs de test (GVT) génère des jeux de test. L'analyseur de la réponse (AR) compare les réponses obtenues avec les réponses prévues.

Le chemin d'accès de test, également appelé mécanisme d'accès de test TAM (*Test Access Mechanism*), transporte les jeux de test du générateur au circuit sous test et transporte les réponses du circuit sous test à l'analyseur.

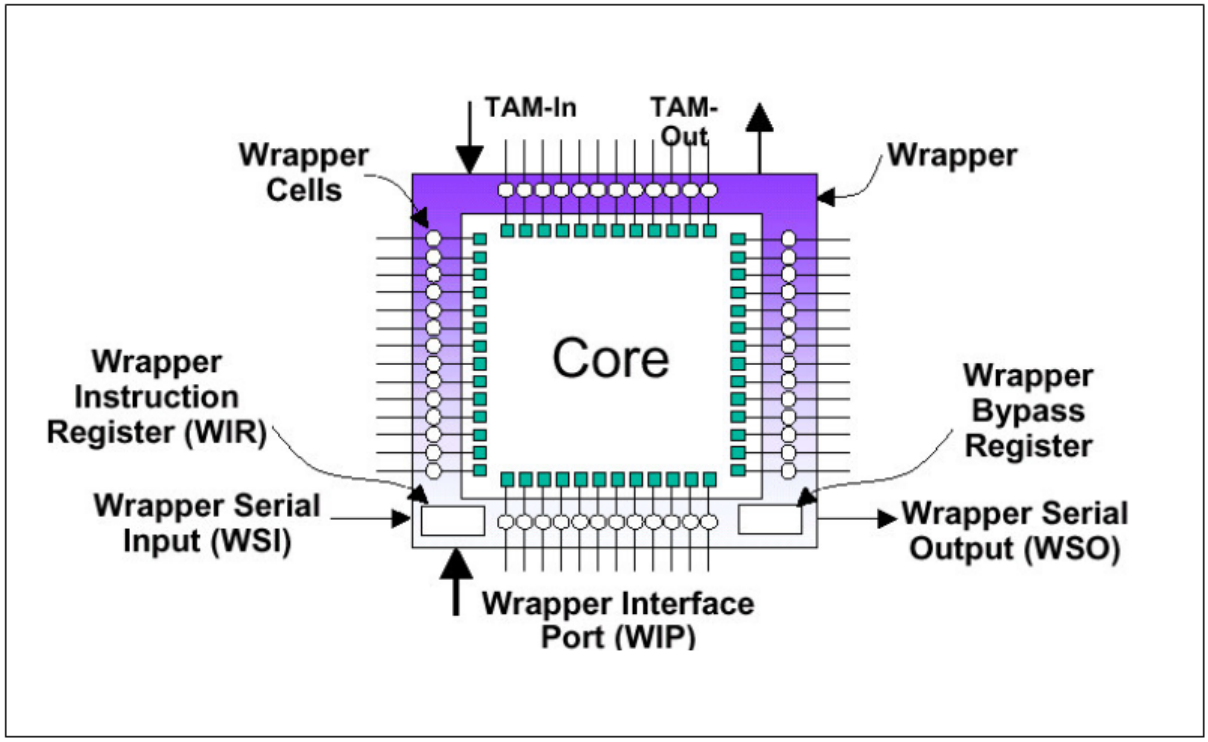


FIGURE 1.9 – Architecture d'un Wrapper P1500 [68].

Le *Wrapper* de test [68] forme l'interface entre l'IP sous test et l'environnement (Figure 1.9). Il relie les entrées/sorties de l'IP sous test aux autres blocs du système et aux mécanismes d'accès de test. Dans le mode normal, le *Wrapper* de test est transparent au fonctionnement du système. Dans le mode test, le *Wrapper* de test applique les vecteurs de test au CUT et recueille les réponses.

Vu que les systèmes sur puce intègrent, souvent, différents types d'IP de différents partenaires, chaque IP possède sa stratégie de test et ses vecteurs de test individuels. Les intégrateurs de système n'ont souvent aucune connaissance sur les structures des IP et traitent donc les IP comme des boites noires. Pour faciliter l'interopérabilité du test des IP de plusieurs vendeurs, il est nécessaire de standardiser le test des IP embarquées [59].

En effet, en septembre 1995, le Conseil Technique de la Technologie de Test (TTTC) de l'IEEE a lancé une action afin d'identifier les besoins pour standardiser le test des IP embarquées. L'objectif de ce comité était de développer une norme qui permet de tester les IP facilement. Ce travail a été normalisé en 2005 sous la norme IEEE 1500 [69].

1.8 Conclusion

Dans ce chapitre, nous avons mis en évidence le principe et la problématique du test de circuits et systèmes intégrés. Nous avons ainsi présenté les techniques de test standard les plus utilisées dans l'industrie. Ces techniques prévoient le test des circuits dès leurs premières phases de conception pour en réduire le coût inhérent au test. Nous avons vu que le test prend une part, de plus en plus importante, dans le prix de revient d'un circuit intégré. En effet, avec l'augmentation incessante de la densité d'intégration des transistors sur une même surface de silicium, le test de ces circuits intégrés devient toujours plus complexe et plus long. De plus, les testeurs sont des appareils extrêmement coûteux, qu'il est difficile d'améliorer à un rythme suivant la même évolution technologique que les circuits. Le problème majeur est de faire passer un volume de données toujours plus grand du testeur au circuit sous test tout en essayant de maintenir le temps de test le plus bas possible. Les concepteurs de systèmes doivent donc avoir des moyens d'automatiser la tâche de conception en vue de test et d'évaluer rapidement plusieurs solutions.

Le challenge est que le test des circuits intégrés est devenu, aujourd'hui, une étape critique lors de la fabrication des puces dédiées à des applications critiques où la fiabilité est le principal facteur pris en compte. De plus, certaines applications critiques nécessitent une vérification continuelle des fonctionnalités de ces systèmes sans pour autant ralentir ou perturber leur fonctionnement normal. Ceci rend la problématique de test plus délicate. Les techniques usuelles, adoptées dans de ce genre de test, seront présentés dans le prochain chapitre.

Chapitre 2

ÉTAT DE L'ART DES TECHNIQUES DE TEST EN LIGNE

Résumé *: Ce chapitre présente un tour d'horizon détaillé des principales techniques de test en ligne, proposées dans la littérature, en mettant en évidence les objectifs visés par le test en ligne. Nous concluons ce chapitre en exposant les avantages, les limitations et les problèmes actuels de test des systèmes numériques.*

2.1 Introduction

Le test de circuits intégrés est mis en œuvre pour déceler la présence de dysfonctionnement dans le circuit sous test. La testabilité des circuits complexes pose un problème important depuis un certain temps déjà. Cela a conduit à l'adoption des techniques de test hors ligne telles que les techniques Ad-Hoc, Full-Scan, Boundary Scan et Built-In-Self-Test (BIST).

Par ailleurs, certains domaines d'applications (applications critiques en médecine, transports, procédés chimiques, etc.) ne peuvent pas tolérer l'interruption de leur fonctionnement, même quelques secondes par année, et demandent de montrer l'état des systèmes au cours de leur fonctionnement normal. Dans ces conditions, il devient pratiquement impossible de tester les systèmes en utilisant les techniques de test hors ligne. Il devient donc de plus en plus évident que les systèmes du futur devraient inclure des moyens d'autotest donnant un avantage évident aux techniques de test en ligne [70].

Ceci exige la mise au point de nouvelles stratégies et le développement de nouvelles techniques qui assurent le bon fonctionnement global, avec une très haute fiabilité, et sachant que toutes les différentes applications mises en jeu répondent aux critères de sûreté

prédéfinis pour chacune. Il s'agit donc de réaliser une vérification de la conformité des résultats produits par ces systèmes en service. Les objectifs visés par ce test dit "en ligne" sont multiples et parfois spécifiques. En effet, une variété de techniques ont été développées pour satisfaire ces besoins et réaliser ces objectifs. Ces dernières peuvent être regroupées en trois grandes catégories [6] à savoir : le monitoring des paramètres électroniques analogiques ("*Analogue Electronic Parameters Monitoring*"), le "*Self-Checking Design*" et le "*On-line BIST*".

En général, ces techniques introduisent de nouveaux hardwares et/ou softwares pour vérifier certaines propriétés invariantes de la réponse du système sous test étant donné que les stimuli et les réponses ne sont pas connus d'avance. Les propriétés des codes détecteurs d'erreurs et correcteurs d'erreurs, le code de parité par exemple, sont parmi les propriétés les plus utilisées dans ces techniques. Néanmoins, d'autres techniques ont eu recours à la duplication du module sous test pour assurer la comparaison. Bien que ces techniques augmentent la fiabilité, elles demeurent toujours onéreuses. Ceci a donné naissance à d'autres formes de duplication pour contourner ce problème. De ce fait, les principales techniques adoptées dans ce genre de test seront détaillées dans les paragraphes suivants de ce chapitre.

2.2 La nécessite du test en ligne des applications critiques

L'utilisation des circuits intégrés dans des systèmes sécuritaires devient de plus en plus importante. De telles applications requièrent la détection concurrente d'erreurs car une erreur de données pendant la période de fonctionnement normal peut produire de graves effets. C'est le cas, par exemple, des applications médicales ou celles de transport où une défaillance a un impact direct sur la vie de l'être humain. Ou encore le cas de systèmes ayant une mission dans des environnements critiques tels que les applications spatiales, souterraines ou sous-marines où la défaillance peut entraîner des conséquences graves en sécurité et au niveau économique.

Le test en ligne cible donc les fautes physiques qui apparaissent pendant le fonctionnement normal du système, par opposition aux erreurs de fabrication [30]. Une faute physique peut être le résultat d'une corrosion physique en fonction du temps, ou de facteurs environnementaux tels que : la température, les radiations, la pression etc. La détection et l'éventuelle suppression (ou actions de correction) de telles fautes, sans pour autant perturber ou interrompre le fonctionnement normal du système, augmente sans doute la

fiabilité du système [71,72]. Par ailleurs, des pénalités hardwares et parfois des dégradations de performances sont observées après adoption des ressources de test en ligne. Les modifications apportées aux conceptions originales s'accompagnent d'une augmentation significative du coût.

Cependant, on peut argumenter que le test en ligne est une nécessité pour les applications critiques en assurance et sécurité humaines telles que l'avionique, l'espace, l'automobile ou l'électronique médicale ; là où il n'est pas toléré d'avoir des fautes et en conséquences, ce test peut être combiné avec les mécanismes d'auto restauration et d'autoréparation.

Une technique usuelle considère périodiquement le système en mode *Off-line* et réalise à ces intervalles des tests *On-line*. Le test en ligne est une alternative attractive quand les performances et les dégradations résultantes du test Off-line sont inacceptables. Les résultats consignés dans [73] démontrent l'importance de la couverture de fautes en ligne pour améliorer la fiabilité dans les applications critiques. En outre, ce type de test est bénéfique dans des environnements très hostiles, tels que l'environnement industriel, où les défaillances physiques sont très fréquentes et parfois certaines.

Enfin, il peut être argumenté et prouvé [73,74] que la progression rapide de la technologie et l'intégration très poussée (submicronique) ainsi que la réduction de la consommation de puissance et l'augmentation de la fréquence d'utilisation, ont rendu les circuits de plus en plus susceptibles aux défaillances physiques et, en conséquence, le test en ligne devient un impératif même pour les applications de commodité quotidiennes.

Il est à noter que le test en ligne, probablement, n'a pas d'intérêt dans les applications familiales telles que les jeux vidéo par exemple et certainement là où le prix est le facteur déterminant et donc le concepteur peut raisonnablement décider de ne pas l'impliquer.

En pratique [75], le test en ligne est réservé aux industries technologiques de pointe, aux systèmes de contrôle de qualité et particulièrement là où il y a susceptibilité d'affecter durement les humains.

2.3 Principe du test en ligne

Les fautes sont des défauts physiques ou logiques dans la conception ou l'implémentation d'un circuit digital. Sous certaines conditions, ces fautes conduisent aux erreurs qui sont des états incorrects d'un système. Les erreurs induisent des défaillances et des déviations du comportement attendu du système en question.

Les fautes peuvent être classées en trois groupes : des fautes de conception, de fabrication et d'exploitation [3]. Les fautes de conception sont commises par les concepteurs

eux-mêmes ou par les outils software CAD (simulateurs, générateurs de fonctions de routage,...) durant le processus de conception. Les défauts de fabrication résultent de l'imperfection des processus de fabrication, par exemple, le circuit ouvert et le court-circuit sont les défauts de fabrication les plus courants. Tandis que les fautes opérationnelles sont dues essentiellement à des perturbations environnementales durant l'exploitation normale du système. Ces perturbations englobent l'interférence électromagnétique, l'extremum de température et les vibrations. En outre, certains défauts de conception et de fabrication échappent à la détection et se combinent avec d'autres défauts dus aux perturbations environnementales et peuvent causer au système en utilisation de sérieux problèmes. Les fautes opérationnelles sont souvent classées selon leur durée d'apparition [3].

- ***Fautes permanentes :*** Elles demeurent indéfiniment présentes si aucune action de correction n'est prise en charge. Plusieurs d'elles sont des fautes résiduelles de conception ou de fabrication. Le reste des fautes apparaît durant les changements de modes d'opération du système tels que le démarrage, l'arrêt du système ou encore le résultat des perturbations environnementales catastrophiques telle que la collision.
- ***Fautes intermittentes :*** Elles apparaissent, disparaissent et réapparaissent de manière répétitive. Il est difficile de les prédire mais leurs effets sont fortement corrélés. En leur présence, le système continue à fonctionner correctement, dans la plupart du temps, mais échoue sous des conditions environnementales atypiques.
- ***Fautes transitoires :*** Elles apparaissent et disparaissent rapidement et ne sont pas corrélées entre elles. Elles sont souvent induites par des perturbations environnementales aléatoires.

L'objectif du test en ligne est donc, de détecter l'effet de ces fautes ou erreurs et d'entreprendre les actions de correction appropriées. Par exemple, dans certaines applications, le système s'arrête après la détection d'erreurs. Dans d'autres applications, la détection d'erreur fait appel à un mécanisme de reconfiguration qui permettra au système de continuer à opérer mais avec une certaine dégradation de performances.

Le test en ligne peut prendre la forme externe ou interne en utilisant du hardware et/ou du software. Le monitoring interne appelé Autotest (*Self-Testing*), prend place sur le même substrat que le circuit sous test (CUT), ceci signifie que le test est implanté au sein du circuit intégré et fait partie des fonctions du système.

Quatre paramètres sont à considérer dans la conception d'un schéma de test en ligne. Ces paramètres sont les suivants [3,76] :

- ***Couverture de faute :*** C'est la fraction détectée d'erreurs modélisées, elle est souvent exprimée en pourcentage. Les applications critiques et à haute disponibilité nécessitent une haute couverture de fautes pour minimiser la probabilité de défaillance.
- ***Latence d'erreur :*** C'est le temps qui s'écoule entre le moment où l'erreur devient active et le moment où elle est détectée. La latence d'erreur dépend du temps pris pour réaliser un test et du nombre de fois où celui-ci est exécuté. Un paramètre lié à la latence d'erreur est la Latence de faute, définie par la différence de temps entre le moment de sa première apparition et le moment de sa détection. Il est clair que la latence de faute est plus grande ou égale à la latence d'erreur. Néanmoins, quand la latence d'erreur est difficile à déterminer, les concepteurs considèrent la latence de faute plutôt que la latence d'erreur.
- ***Redondance de surface ("Space redundancy") :*** C'est l'extra hardware ou firmware nécessaire pour le test en ligne.
- ***Redondance de temps ("Time redundancy") :*** C'est l'extra temps nécessaire pour le test en ligne.

Idéalement, les concepteurs de test veillent à la mise en œuvre des techniques de test en ligne ayant une couverture de faute de 100%, un cycle d'horloge de latence d'erreur, aucune redondance de type "*Space redundancy*" et aucune redondance de temps "*Time redundancy*". Ainsi, elles n'exigent aucune modification majeure de la conception originale et n'imposent aucune restriction de type structurel ou fonctionnel sur les CUT. En général, la considération de ces quatre paramètres dans une conception engendre malheureusement des conflits. Notons qu'une couverture de faute élevée exige une latence d'erreur, une "*Space redundancy*" et/ou une "*Time redundancy*" élevées. Les techniques dotées de moyens de détection immédiate (latence d'erreur =1) minimisent le "*Time redundancy*" mais nécessitent du hardware en plus. Tandis qu'un schéma de détection avec retard (latence d'erreur >1) réduit le "*Time*" et "*Space redundacy*" au détriment d'une augmentation de la latence d'erreur. La majorité des méthodes BIST courantes répondent à certaines de ces contraintes sans pouvoir toutefois les adresser ensemble. Par ailleurs, plusieurs techniques de détection avec retard, proposées dans la littérature [3,77], considèrent des combinaisons d'entrées équiprobables et essaient d'établir un extremum probable de la latence d'erreur. Les résultats qui en découlent ont montré que certaines fautes restent non détectées pour une longue période de test car les vecteurs qui les détectent apparaissent rarement aux entrées des CUT. Il est donc fortement recommandé de fixer, à priori, les

paramètres déterminant dans la conception de telles techniques.

2.4 Types de test en ligne

Pour couvrir toutes les fautes opérationnelles évoquées dans le paragraphe précédent, les ingénieurs de test utilisent deux modes de test en ligne différents : test en ligne concurrent et test en ligne non concurrent [3,30]. Le test concurrent prend place durant l'opération normale du système, tandis que le test non concurrent prend place quand l'opération normale du système est suspendue temporairement. Il est à noter qu'on peut combiner les deux modes, pour permettre la mise au point, à coût raisonnable, d'une stratégie efficace. La figure 2.1 montre une taxonomie des méthodes de test en ligne pour les systèmes embarqués ("*Embedded systems*").

2.4.1 Test en ligne non concurrent

Cette forme de test est soit sporadique ("*Event-triggered* ") soit périodique ("*Time-triggered*") et elle est caractérisée par une faible redondance *espace-temps*. Le test sporadique est initialisé par des évènements clés ou par des changements d'états tels que le démarrage ou l'arrêt du système. Son but est la détection ou la possibilité de correction des fautes permanentes très tôt. Ce type de test ressemble beaucoup au test de fabrication. De tels tests peuvent être appliqués aussi longtemps que les ressources nécessaires sont disponibles. Typiquement, le hardware est partitionné en composants, chaque composant est activé par un test spécifique [78].

Le test périodique apparaît à des intervalles de temps prédéfinis au cours de l'opération normale du système. Ce test permet la détection des fautes permanentes et, souvent, utilise les mêmes mécanismes de test appliqués par le test sporadique. L'approche périodique est spécialement utile dans les systèmes opérant sur des périodes très longues. Le test périodique est aussi essentiel pour la détection des fautes intermittentes. Ces dernières se comportent comme des fautes permanentes sur une courte durée. Notons que ces tests sont partitionnés et répartis sur des intervalles de manière qu'une seule partie de test est appliquée durant chaque période [3].

2.4.2 Test en ligne concurrent

Contrairement au test non concurrent, qui ne peut détecter des fautes transitoires dont les effets apparaissent et disparaissent rapidement, le test concurrent vérifie continuellement l'occurrence d'erreurs dues à de telles fautes. Cependant, le test concurrent n'est

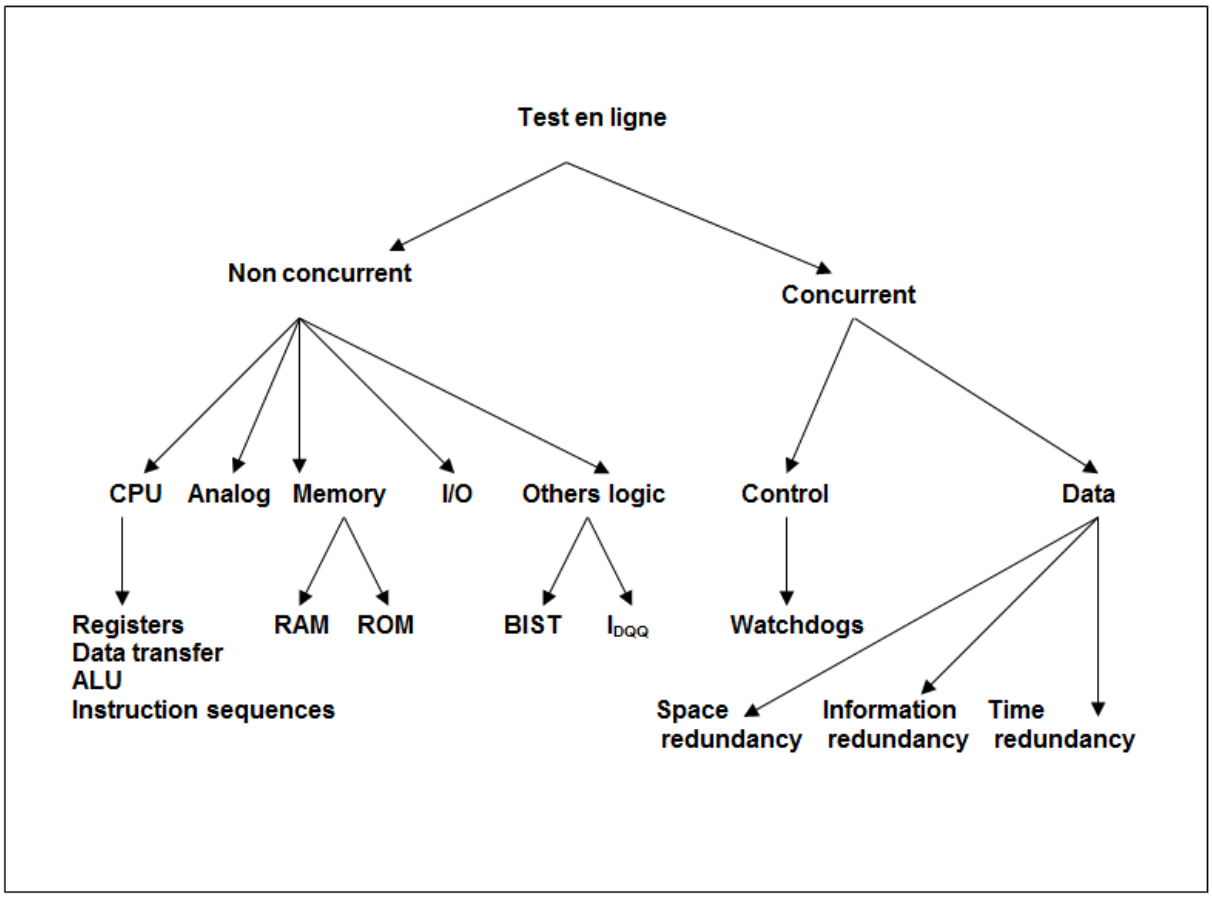


FIGURE 2.1 – Taxonomie des méthodes de test en ligne pour les systèmes embarqués

pas particulièrement utile dans le diagnostic de la source de l'erreur c'est pourquoi les concepteurs de tests le combinent avec un diagnostic software [6]. Ils peuvent également combiner les tests concurrents et non concurrents pour détecter des fautes complexes de tout type [3].

La méthode la plus courante est le "*Watch-dog Timer*". Elle est destinée, particulièrement, à détecter les erreurs de contrôle en exploitant un support hardware spécial pour la réalisation du test concurrent [79]. Ce support utilise un compteur, initialisé par le système de manière répétitive, pour indiquer que le système fonctionne correctement.

L'élément clé dans le test concurrent est la redondance concernant les erreurs de données ("*Data errors*"). Par exemple, la technique de duplication avec comparaison [80] permet la détection de toute faute simple à un coût de 100% de "*Space redundancy*". Cette technique nécessite deux copies du CUT, qui opèrent avec les mêmes entrées et la divergence de leurs sorties indique la présence d'erreurs. Cependant, dans certaines applications, la redondance hardware due à la duplication est inacceptable. Une technique alternative, avec un faible coût, en l'occurrence "*Time redundancy*" appelée "*Double exé-*

cution" ou encore "*Retry*" ; consiste à exécuter les opérations critiques plusieurs fois, à des moments différents, et à comparer les résultats obtenus. Les fautes transitoires affectent, éventuellement, un seul exemple de ces opérations et par conséquent, peuvent être détectées.

Une troisième forme de redondance, largement utilisée, est la redondance de l'information ("*Information redundancy*") qui consiste à l'ajout d'information codée tels que les bits de contrôle de parité [80]. Ces techniques ainsi que d'autres sont traitées avec plus de détails dans le paragraphe suivant.

2.5 Les techniques de test en ligne

Les techniques de test en ligne, les plus utilisées, peuvent être regroupées en trois catégories [6] :

1. Techniques du monitoring des paramètres électroniques analogiques dénommées "*Monitoring Analogue Electronic Parameters* " ;
2. Techniques dénommées : "*Self- Checking Design* (SCD)" ;
3. Techniques dites : "*On-line BIST*" et "*DFT*".

Le "*Monitoring Analogue Electronic Parameters*" est utilisé pour détecter des erreurs dans les changements des propriétés électroniques des signaux d'information qui donnent naissance à des fautes très difficiles à détecter. Elles induisent l'apparition de fautes logiques après un certain temps (par exemple : test de délai). Un suivi rigoureux des propriétés électroniques des signaux réduit la difficulté de détection de genre de fautes.

Le "*Self-Checking Design*" quant à lui, consiste à coder les sorties d'un module en utilisant les codes correcteurs d'erreurs et ensuite vérifier certaines propriétés invariantes de ces codes. Tandis que, le "*On-line BIST*", exploite, souvent, l'existence des ressources hors-ligne pour réaliser des tests durant l'état de repos d'un module donné ou encore pendant certains intervalles où l'opération normale est temporairement suspendue.

Il est à noter, que cette classification n'est pas exhaustive. En fait, il y a des techniques qui combinent plusieurs éléments des catégories susmentionnées.

2.5.1 Monitoring des paramètres analogiques

Ces techniques consistent à détecter des fautes grâce à l'impact induit sur des caractéristiques analogiques plutôt que sur les valeurs logiques. Parfois, des fautes sont détectées parce que les caractéristiques analogiques sont corrompues en même temps que les valeurs

logiques, parfois l'effet d'une faute sur les caractéristiques analogiques valide leur détection avant que les valeurs logiques soient corrompues. Ainsi, une prédiction de défaillance du système est alors possible. Cependant, l'inconvénient majeur de ces techniques réside dans la difficulté d'intégrer des capteurs dans le circuit, qui permettent le monitoring de ces paramètres, et la fixation du seuil de ces paramètres. Enfin, l'insertion de ces capteurs a un impact direct sur les performances de la conception originale et donc mérite considération [6].

2.5.1.1 Monitoring du paramètre courant

C'est la technique la plus développée de sa famille. Elle est basée sur le concept que les défauts physiques dans les VLSI résultent en une consommation anormale de courant. Les classes importantes des défauts dans la technologie CMOS sont : le circuit ouvert, le circuit-fermé, les fautes en pont etc. Ces derniers sont difficiles à détecter par le monitoring logique du fait qu'ils induisent des valeurs intermédiaires [6,81].

2.5.1.2 Monitoring du paramètre Température

La température dans un circuit est supposée ne pas dépasser un certain seuil prédéfini dans les conditions normales. Cependant, suite à une défaillance, la dissipation d'énergie et par conséquent la température peut s'élever substantiellement. Un capteur thermique, placé dans une position stratégique dans le circuit, détectera et signalera cette anomalie [6,82].

2.5.2 Technique du "*Self-Checking Design*"

Le "*Self-Checking Design*" (SCD) est utilisé pour réaliser la détection concurrente d'erreurs au moyen de la redondance hardware. Un circuit complexe est partitionné en blocs fonctionnels et chacun de ces blocs est implémenté selon la structure illustrée sur la figure 2.2. Cette structure réalise des blocs fonctionnels qui délivrent des sorties appartenant à un code détecteur d'erreurs, et ainsi introduisant une propriété invariante qui peut être vérifiée concurremment [6].

Le but désiré que doit réaliser un circuit SCD est souvent étudié comme un but " *Totally Self- Checking*" (TSC). Ce but exige, sous n'importe quelle faute modélisée, que la première sortie erronée d'un bloc fonctionnel soit signalée sur les sorties du contrôleur (" *Checker*"). Pour cela, certaines propriétés devraient être vérifiées aussi bien par le bloc fonctionnel que par le contrôleur. Ces propriétés ont été introduites par Carter [83] et formalisées par Anderson [84].

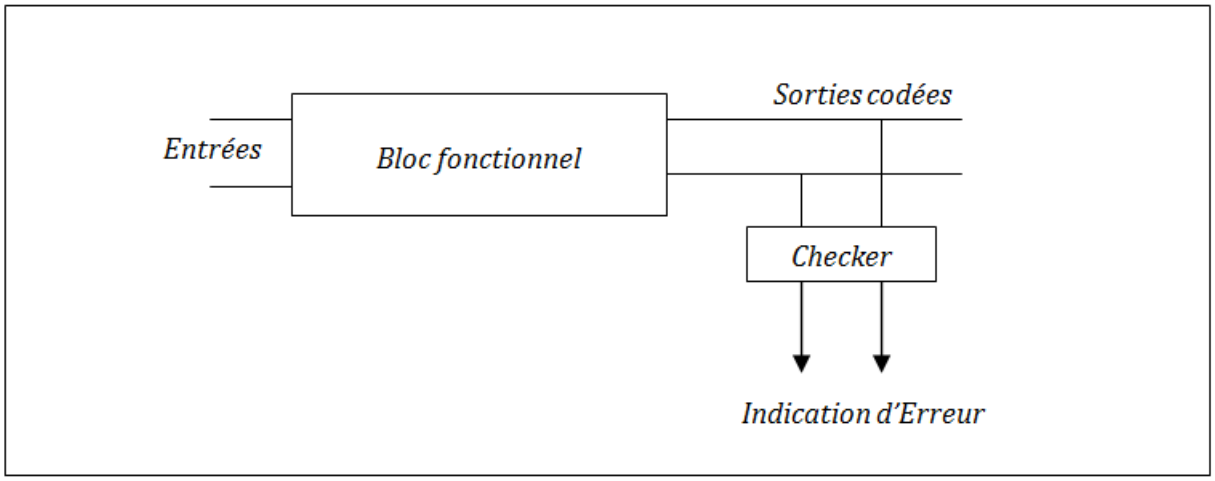


FIGURE 2.2 – Structure générale d'un circuit Self-Checking

2.5.2.1 Conception du Bloc Fonctionnel

Les propriétés suivantes sont nécessaires pour la conception du bloc fonctionnel [6] :

■ ***Propriété 1 : "Fault Secure"***

Sous n'importe quelle faute modélisée, les sorties produites erronées ne doivent pas appartenir aux sorties codées (mots codes). La raison pour laquelle cette propriété est évidente est que si une sortie erronée appartient aux codes de sorties, l'erreur est indétectable et le but TSC n'est plus réalisé [6,85]. Ainsi, la propriété "*Fault Secure*" est la plus importante propriété nécessaire pour le bloc fonctionnel. Une autre propriété utile est la propriété " *Self-Testing*". Celle-ci stipule que pour chaque faute, il existe au moins un vecteur d'entrée qui la détecte durant l'opération normale du circuit. En fait, cette propriété évite l'existence de fautes redondantes. De telles fautes restent indétectables et peuvent être combinées avec de nouvelles fautes qui apparaissent plus tard dans le circuit, résultant ainsi en fautes multiples qui pourraient détruire la propriété "*Fault Secure*". La combinaison des propriétés "*Fault Secure*" et "*Self-Testing*" offre un haut niveau de protection. Ces propriétés peuvent être exprimées comme suit :

■ ***Propriété 2 : "Self-Testing"***

Pour chaque faute modélisée, il existe un vecteur d'entrée qui apparaît durant l'opération normale du circuit et produit un vecteur de sortie qui n'appartient pas au code considéré [6].

■ ***Propriété 3 : "Totally Self-Checking (TSC)"***

Le circuit est TSC s'il est à la fois *"Fault Secure*" et "*Self-Testing*" [86]. Comme résultat, avec la propriété *"Fault Secure*", on assure que la première faute générée est toujours

détectable. Cependant, la propriété "*Self-Testing*" assure que la première faute est détectée avant que la seconde apparaisse dans un system " *Self-Checking*". De cette façon, la propriété "*Totally Self-Checking*" est vérifiée. Bien que la propriété "*Fault Secure*" soit la plus importante d'une part, elle est la plus difficile à réaliser d'autre part. Contrairement, la propriété "*Self-Testing*" peut être aisément réalisée, particulièrement, quand il s'agit des fautes de collages simples. Quant à la propriété *"Fault Secure*", la meilleure façon de la réaliser est de dupliquer le bloc fonctionnel et d'utiliser un comparateur pour vérifier l'égalité des sorties produites. Puisque cette solution est coûteuse en matière d'extra hardware nécessaire, pratiquement plus de 100%, des techniques plus élaborées ont été développées pour faire face à cet inconvénient. Elles utilisent des codes correcteurs d'erreurs avec un coût minimum par rapport aux techniques faisant appel à la duplication.

2.5.2.2 Application des codes détecteurs d'erreurs dans le SCD

L'utilisation de codes valide le circuit à concevoir, de manière que certaines classes d'erreurs seront automatiquement détectées ou corrigées à travers l'exploitation d'un contrôleur hardware de vérification, tel que le montre la figure 2.3. Plusieurs codes ont été développés pour les conceptions "*Self- Checking*". Néanmoins, le type de code utilisé peut dépendre du type du circuit [30]. Les codes les plus couramment utilisés sont :

2.5.2.2.1 Code de parité

Il permet la détection de toutes les fautes simples, et plus généralement, toutes les fautes de *multiplicité impaire*. Il est moins coûteux puisqu'il ajoute un seul bit de contrôle à l'information. Ce bit est calculé de façon à ce que la parité du mot code soit constante.

2.5.2.2.2 Code m/n

C'est une variété des codes non-séparables (bits information et contrôle fusionnés). Il est composé par des mots codes qui ont exactement *m-1* parmi les n-bits. Par exemple, le code 2/4 est constitué par des mots code : 1100, 1010, 1001, etc.

2.5.2.2.3 Code Berger

Ce code a deux parties : une partie information et une partie contrôle. La partie contrôle représente le nombre de 0 de la partie information [6,30]. A titre d'exemple, pour l'information $I = 10010$, la partie contrôle, dans ce cas, est alors $C = 011$. Une seconde variété de ce code définit la partie contrôle comme étant le complément du nombre de "1" de l'information. Pour le même exemple, la partie contrôle est donc, $C = 100$. On note que, pour une information de n-bits, le nombre de bits de contrôle est égal à $[Log_2\ (n+1)]$.

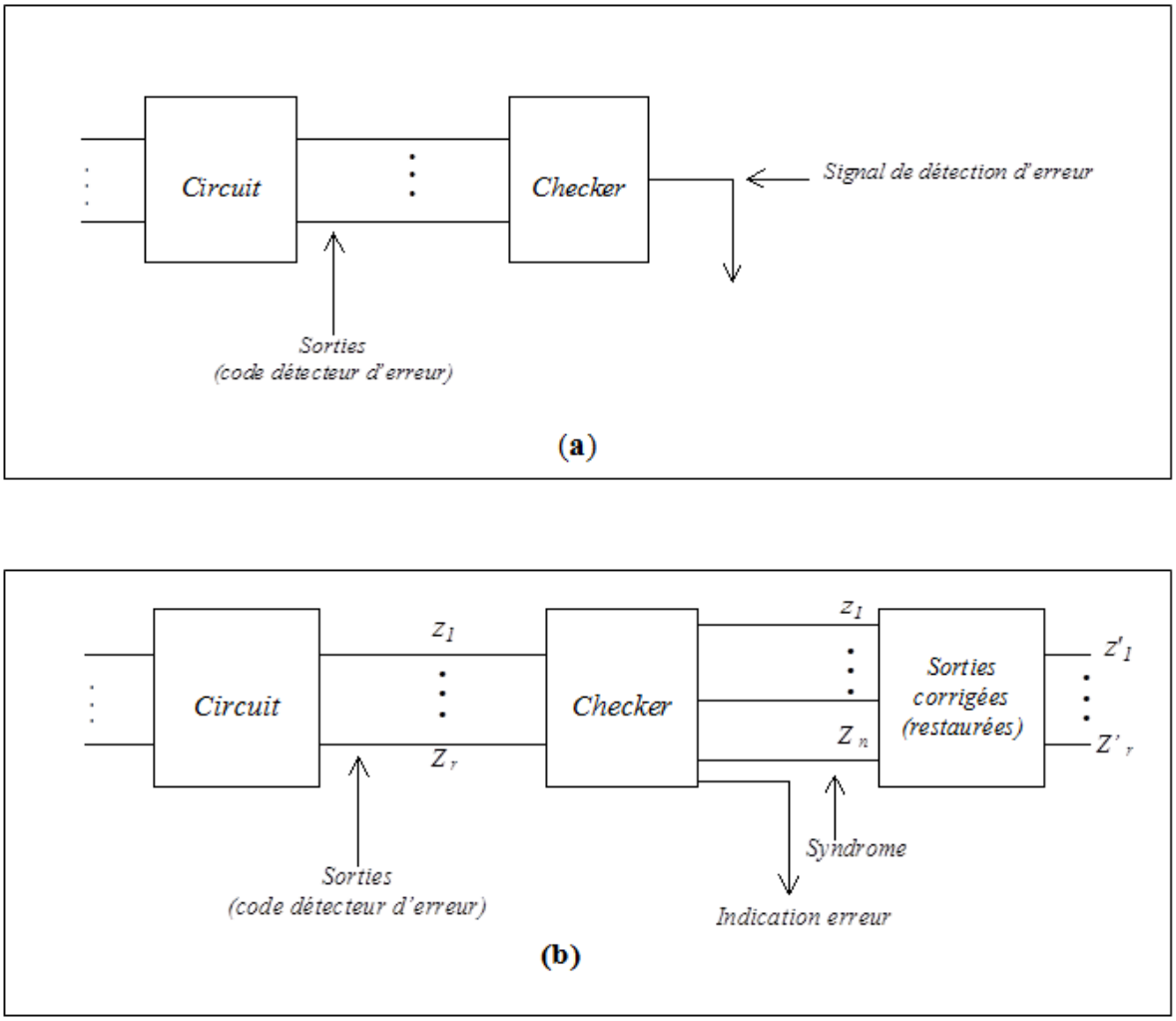


FIGURE 2.3 – **(a)** Circuit détecteur d'erreur **(b)** Circuit correcteur d'erreur

2.5.2.2.4 Code et distance de Hamming

La *distance* de Hamming d'un code est le nombre de bits dans lesquels deux mots codes diffèrent [30]. Une *distance* constante de Hamming a été utilisée [87] pour coder les signaux d'une machine à états finis "*Self-Checking*". Les signaux d'état sont codés de façon à ce que la *distance* de Hamming entre deux états consécutifs soit constante. Ce codage détecte non seulement des erreurs résultant en non-mots codes, mais aussi les mots codes incorrects. Dans [30], un schéma de code détecteur d'erreurs générique, basé sur la vérification de parité et se référant au code Hamming est défini analytiquement. S'il y a q bits information, c bits de contrôle sont alors nécessaires tout en maintenant la relation $2^c \geqslant q + c + 1$. Le mot résultant consiste en $(q + c)$ bits et peut être représenté comme $b_{q+c}.....b_2b_1$. Les bits b_2^i ; $0 \leq i \leq c-1$ sont les bits de contrôle.

Considérons un entier n et $b_j(n)$ la valeur du j^{ieme} bit de n (représenté en binaire).

On définit $P_j = \{I \ / \ b_j(I) = 1\}$, où P_j est l'ensemble des entiers ou leur représentation binaire a un (1) dans la position j. Alors on définit les équations de contrôle sous la forme :

$$\sum_{k \in P_j} b_k = 0;\ i = 1, ..., c \tag{2.1}$$

Remarque 2.1. Dans l'equation (2.1), $\sum$: Somme modulo 2.

A partir de ces équations, les bits de contrôle peuvent être déterminés.

Le code Hamming a été utilisé aussi pour vérifier des cellules SRAM [88]. Quand une opération d'écriture est exécutée, les bits de contrôle sont calculés et sauvegardés ensemble avec les données utiles. Alors que, lors de l'exécution d'une opération de lecture, le mot sauvegardé est, premièrement, vérifié puis la partie information est isolée et donc utilisée. Le schéma proposé [88] est équipé de ressources BIST qui permettent un test des cellules mémoires par l'accomplissement des opérations de lecture et d'écriture des cellules tout en préservant, temporairement, leur contenu dans des registres.

Une application intéressante du code Hamming est présentée dans [89,90], où le code Hamming est employé pour coder les états d'une machine à états finis, et la détection d'erreur est réalisée à travers l'utilisation du BIST et du MISR (*Multiple Input Signature Register*). Le processus a été codé comme une étape de prétraitement dans le processus de synthèse, produisant ainsi une conception testable en ligne par la modification automatique de la description VHDL.

2.5.2.2.5 Code arithmétique

L'avantage de ces codes est qu'ils se préservent sous les opérations arithmétiques.

Trois catégories des codes arithmétiques sont reportées dans la littérature [6,91].

2.5.2.2.5.1 Codes résiduels : ce sont des codes séparables. Si A est la base du code, alors la partie contrôle est *modulo* A.

2.5.2.2.5.2 Codes résiduels inverses : leur partie contrôle est (A – *modulo* A).

2.5.2.2.5.3 Codes AN : ce sont des codes non séparables. Les mots codes sont égaux au produit des mots non-codes. Il est clair que, les mots non-codes nécessitent une opération *modulo* pour qu'ils soient codés selon les codes arithmétiques, tandis que les sorties fonctionnelles du circuit nécessitent une opération *modulo* pour être vérifiés. Comme intérêt particulier, le cas où $A = 2^k\text{-}1$, l'opération *modulo* est réduite à une simple opération d'*addition*, ainsi le hardware implémentant ces codes n'est pas coûteux.

2.5.2.3 Conception du contrôleur ("*Checker*")

La mission du contrôleur est de signaler l'occurrence des mots codes entrée, par la génération en sa sortie une indication d'opération correcte, et l'occurrence des mots non-code, par génération d'une indication d'erreur [6]. L'ensemble des mots de sortie indiquant une opération correcte forme l'espace des sorties codes du contrôleur et, l'ensemble des mots de sortie indiquant l'occurrence d'erreurs forme l'espace des mots non-codes de celui-ci. Comme implication de cette mission, le contrôleur doit vérifier la propriété code disjoint.

■ ***Propriété Code-disjoint***

Cette propriété stipule que le contrôleur doit traduire les mots codes d'entrées en des codes de sorties et les mots non-codes d'entrées en des sorties non-codes. Il est à noter, que la propriété *code–disjoint* n'est pas liée à la testabilité du contrôleur, elle reflète tous simplement une propriété fonctionnelle. Cependant, l'occurrence d'une faute dans le contrôleur peut altérer son aptitude à produire une sortie indicatrice d'erreur due à une entrée non-code. Si cette faute n'est pas détectée, une seconde peut apparaître plus tard dans le bloc fonctionnel et une sortie erronée est alors produite par ce bloc qui, éventuellement, ne sera pas signalée par le contrôleur dû à sa propre faute. Pour résoudre ce problème, le contrôleur doit vérifier la propriété "*Self-Testing*".

Néanmoins, la conception d'un contrôleur "*Self-Testing*" est une tâche difficile car elle nécessite la détection de toutes fautes dans le contrôleur par application d'une seule entrée code. Heureusement, dans la pratique, on ne considère qu'un nombre limité de classes de contrôleurs correspondants aux codes détecteurs d'erreur les plus utiles.

2.5.2.4 Contrôleur de parité

Afin de concevoir une unité fonctionnelle "*Fault-Secure*" avec codage de parité, il est essentiel, que les fautes internes affectent un nombre impair des bits de sorties. Pour réaliser ce but, une cellule additionneur complet a été introduite [92] avec la possibilité de prédiction de parité. Cette cellule (Figure 2.4) est utilisée comme un bloc constitutif pour les opérateurs arithmétiques.

Dans la figure 2.4, A, B et C_{in} représentent les entrées usuelles, S, C et C_p sont, respectivement, la *somme*, le *Carry* et le *Carry redondant* utilisé pour la prédiction de parité. Les bits C_p de tous les blocs constitutifs sont *X-ORed* pour produire le bit de prédiction de parité globale.

Il a été prouvé [92,93] que, les ALUs, les multiplieurs et les diviseurs utilisant ces cellules sont "*Fault Secure*" et nécessitent ainsi un extra hardware entre 35% à 50% avec une dégradation de performance de 18% à 36%.

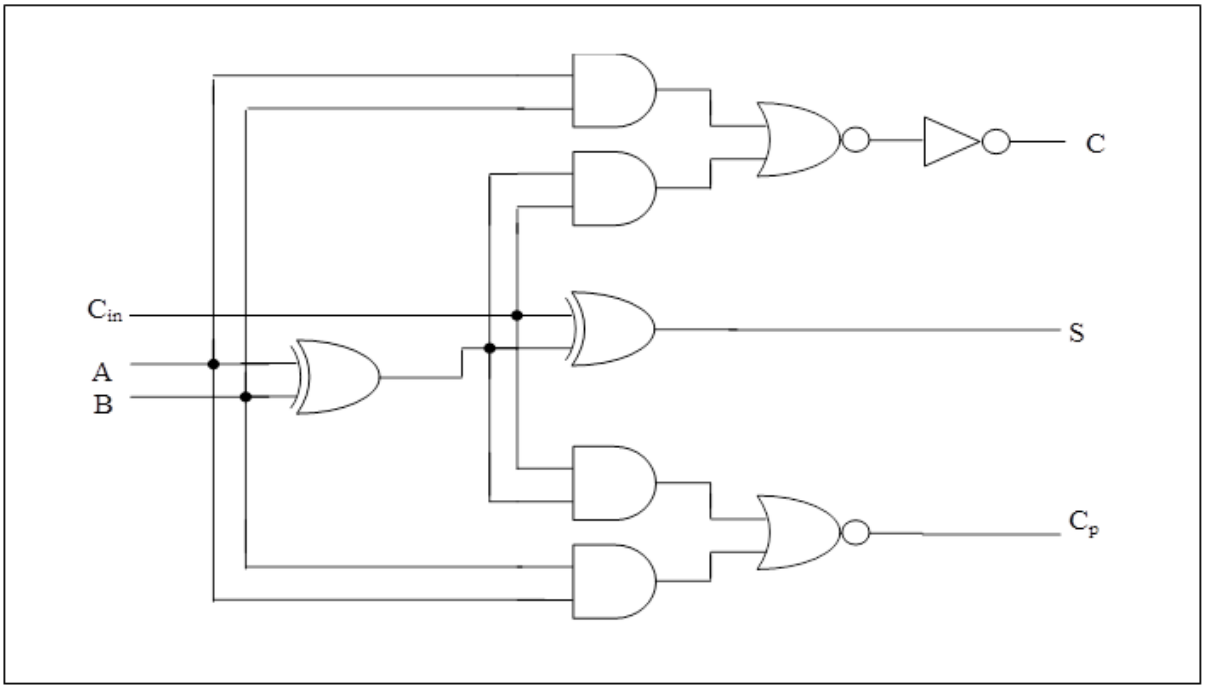


FIGURE 2.4 – Cellule additionneur complet avec prédiction de parité

2.5.2.5 Contrôleur "*Dual-rail*"

Le contrôleur "*Dual-rail Self-Checking*" peut être conçu comme un arbre de parité où chaque porte *XOR* est remplacée par une cellule "*Dual–rail*". Une cellule contrôleur "*Dual-rail*" est montrée sur la figure 2.5. Le contrôleur résultant est facilement testable, puisque seuls quatre mots codes entrées sont nécessaire au test de celui-ci et ce, quelle que soit la taille du circuit [6]. Ce circuit est très utile dans la conception "*Self-Checking*" car il peut être utilisé pour vérifier des blocs fonctionnels duaux et aussi les blocs dupliqués. Son utilisation principale est la compression des signaux d'indication d'erreurs délivrés par les différents contrôleurs d'un circuit complexe "*Self-Testing*". Et comme de tels contrôleurs délivrent des paires de sorties en "*Dual-rail*", alors le contrôleur "*Dual-rail*" peut compacter les paires "*Dual-rail*" délivrées par les différents contrôleurs du système en une seule paire "*Dual-rail*" qui sert d'indication d'erreur globale du système [94].

2.5.2.6 Contrôleur code Berger

La structure générale d'un contrôleur *Self-checking* (pour code séparable) est illustrée sur la figure 2.6. Le circuit *N1* génère les bits de contrôle à partir des bits information. Le contrôleur (Circuit *N2*) compare les bits de contrôle des mots d'entrées avec les bits de contrôle générés par le circuit *N1* [6,30,95,96].

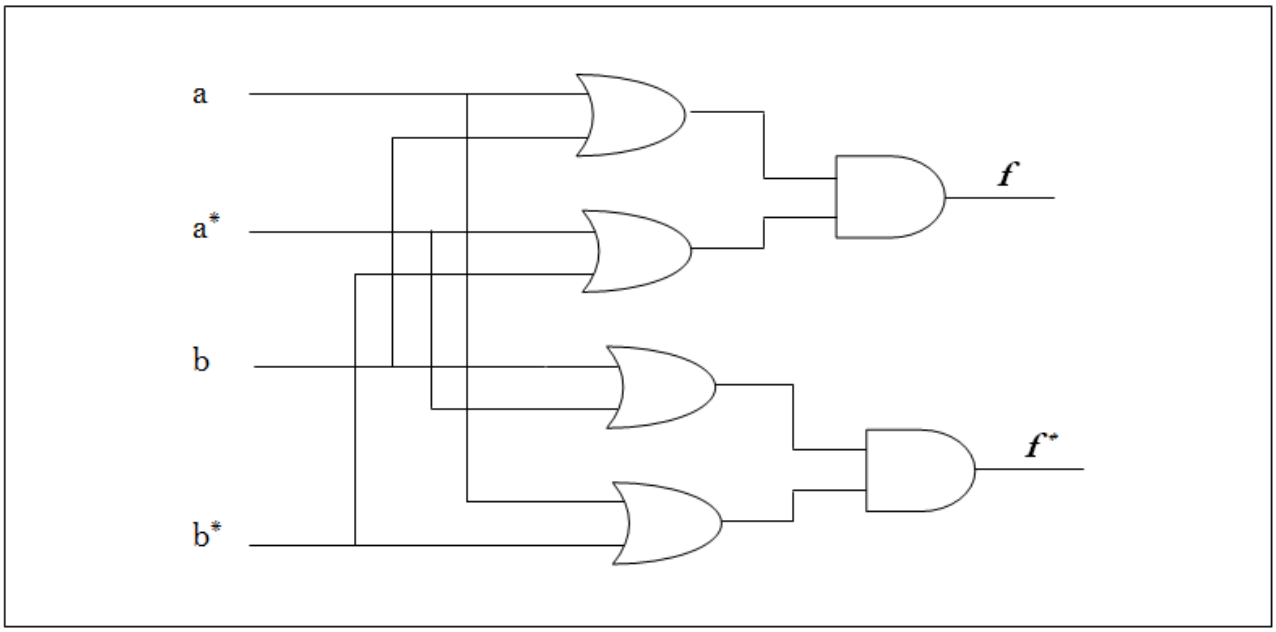


FIGURE 2.5 – Cellule Contrôleur "*Dual-rail*"

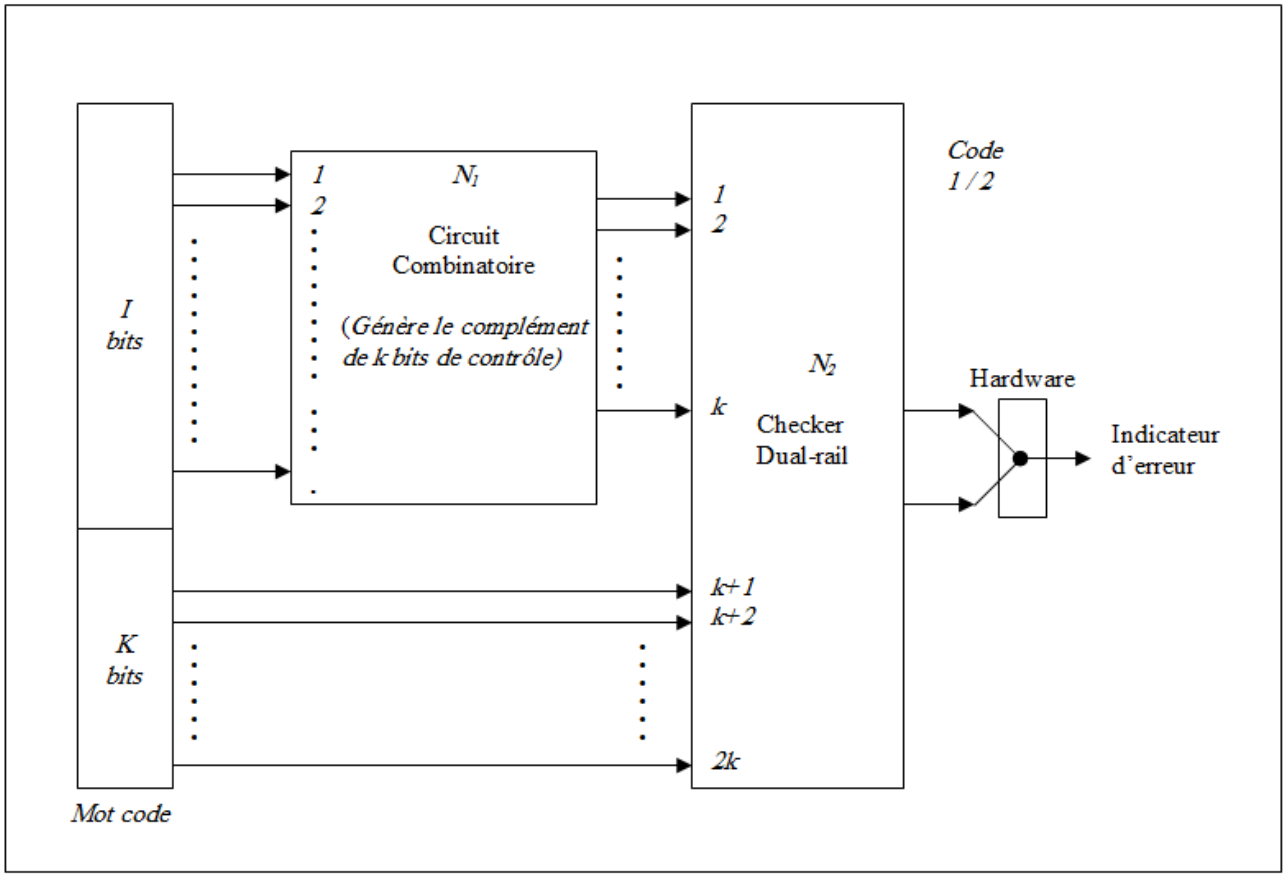


FIGURE 2.6 – Contrôleur TSC (pour code séparable)

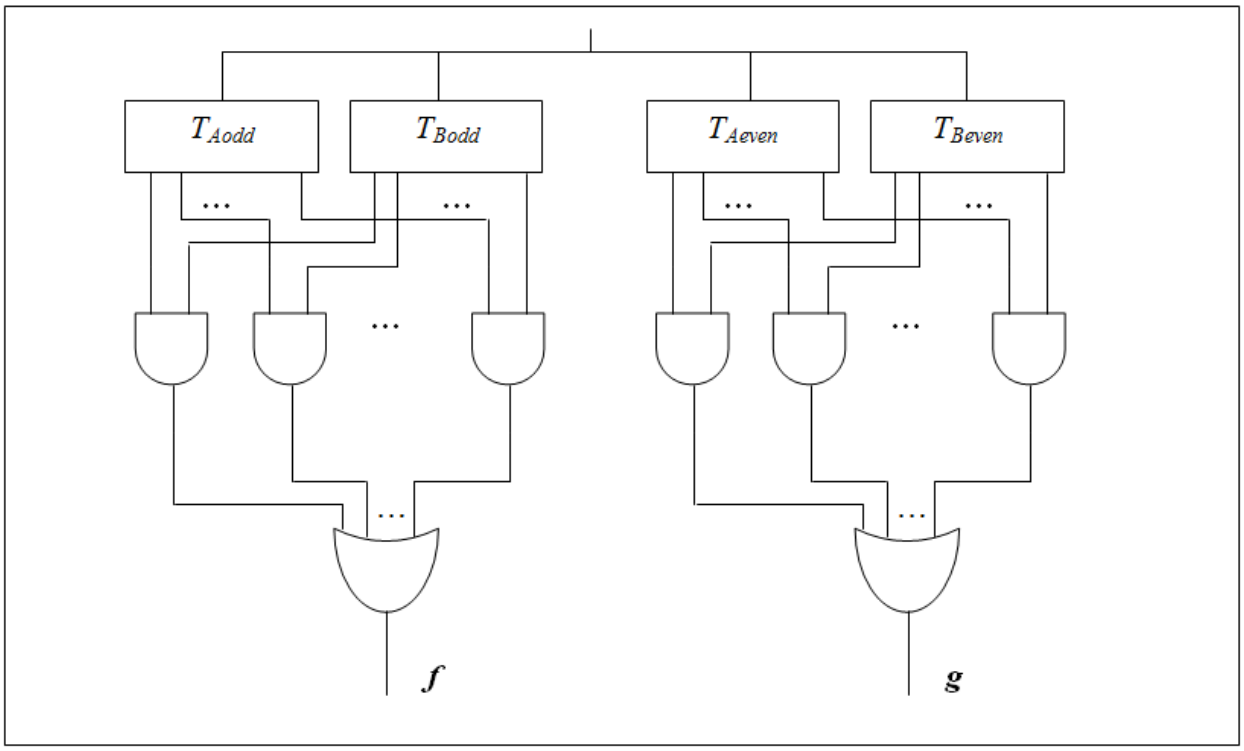


FIGURE 2.7 – Contrôleur TSC (Code $k/2k$)

2.5.2.7 Contrôleur code m/n

Parmi ces contrôleurs, on trouve le contrôleur *k/2k*. Ce dernier a deux sorties f et g et les $2k$ entrées sont divisées en deux sous-ensembles disjoints x_A et x_B, de k entrées chacun. La fonction f est définie pour avoir la valeur 1 si et seulement si i ou plus des variables dans x_A ont la valeur 1, et k-1 ou plus des variables dans x_B ont la valeur 1 et ce pour i *impair*. De manière similaire, la fonction g est définie pour avoir la valeur 1 si et seulement si i ou plus des variables dans x_A ont la valeur 1, et k-1 ou plus des variables dans x_B ont la valeur 1 et ce pour i *pair*. Pour des entrées codes (c.-à-d. exactement k parmi $2k$ variables ont la valeur 1), alors $f = 1$ et $g = 0$ ou $f = 0$ et $g = 1$, tandis que pour les entrées non-codes, avec moins de k 1-bits, $f = 0$ et $g = 0$ et pour des entrées non-codes avec plus de k 1-bits, $f = g = 1$. La forme générale du circuit est montrée sur la figure 2.7 où T_{Aodd} ont des sorties pour chaque valeur de i *impaire*, et la i^{ieme} sortie prend la valeur 1 si i ou plus des variables d'entrées ont la valeur 1. Les sous-circuits T_{Aeven}, T_{Bodd}, et T_{Beven} sont définis similairement. Il a été montré dans [30] que, ce circuit est à la fois "*Fault Secure*" et "*Self-Checking*" et par conséquent, il est TSC.

Le contrôleur $k/2k$ peut être aussi utilisé pour réaliser un contrôleur où $k \neq 2k$. La forme générale du circuit est alors celle montrée sur la figure 2.8. La matrice *AND* consiste

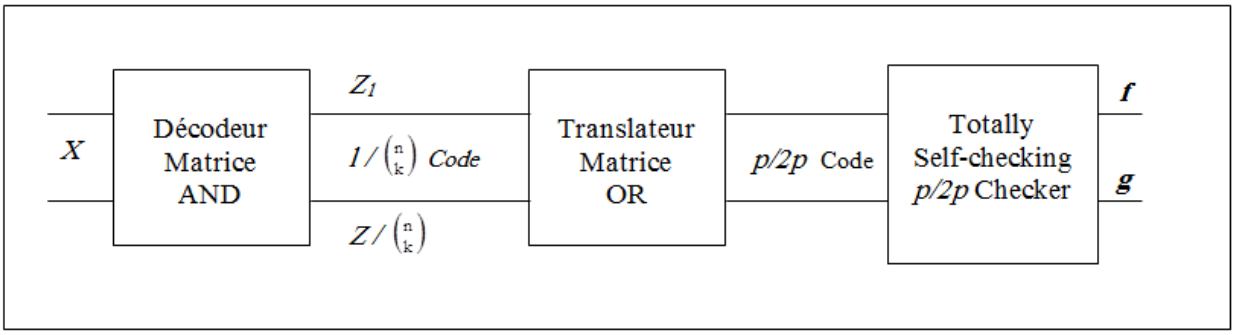


FIGURE 2.8 – Conception du contrôleur code k/n

en un niveau simple de portes *AND* de k entrées, une pour chaque sous-ensemble de k variables, qui génère tous les produits possibles de k des n variables. Tandis que la matrice *OR* consiste en un simple niveau de portes *OR*, qui convertit le code $1/\binom{n}{k}$ en Z signaux de sortie à un code $p/2p$, où p doit satisfaire la contrainte :

$$2p \leq \binom{n}{k} \leq \binom{2p}{p} \tag{2.2}$$

Néanmoins, ce contrôleur présente une solution onéreuse en matière de hardware car il nécessite tous les mots-codes valides, comme ensemble de vecteurs de test, afin de détecter toutes les fautes de collage simple. Un contrôleur plus efficace [30] peut être conçu comme le montre la figure 2.9, consistant en trois sous-circuits C_1, C_2 et C_3. Le circuit C_1 a n entrées et Z sorties où, $Z = 4$ pour des codes $m/(2m+1)$, $Z = 5$ pour des codes $2/n$, et Z = 6 pour n'importe quel autre code m/n.

Durant l'opération normale, le circuit C_1 reçoit les mots m/n codes comme entrées et produit un code $1/Z$ en ses sorties. Le circuit C_2 convertit le code $1/Z$ à ses entrées en un code $2/4$ à ses sorties, et le circuit C_3 est un contrôleur de code $2/4$. Notons que, tous les trois sous-circuits sont conçus comme des circuits "*Totally Self-Checking*". Ce type de réalisation montre, en général, qu'il nécessite moins de vecteurs de test pour détecter les fautes de collage simples par rapport à celui de la figure 2.8.

Remarque 2.2. L'utilisation des codes et le concept "*Self-Checking*" sont aussi applicables aux circuits séquentiels. À titre d'exemple, considérons le modèle de la figure 2.10, où les sorties d'états sont codées. Le "*Self-Checking*" peut être obtenu par la conception de la logique combinatoire telle que [30] :

1. Pour n'importe quelle faute interne de C, et pour n'importe quelle entrée, soit la sortie et l'état futur sont corrects ou la sortie et/ou l'état futur est un mot non-code.

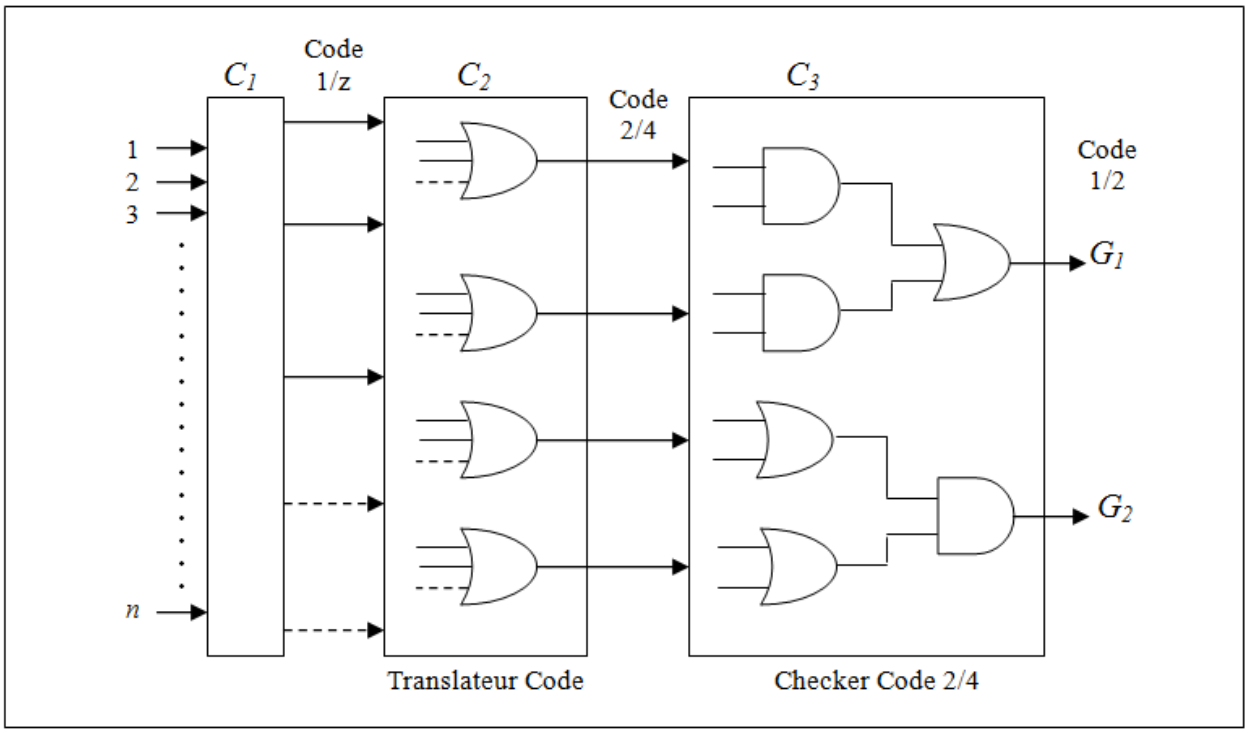


FIGURE 2.9 – Contrôleur code m/n

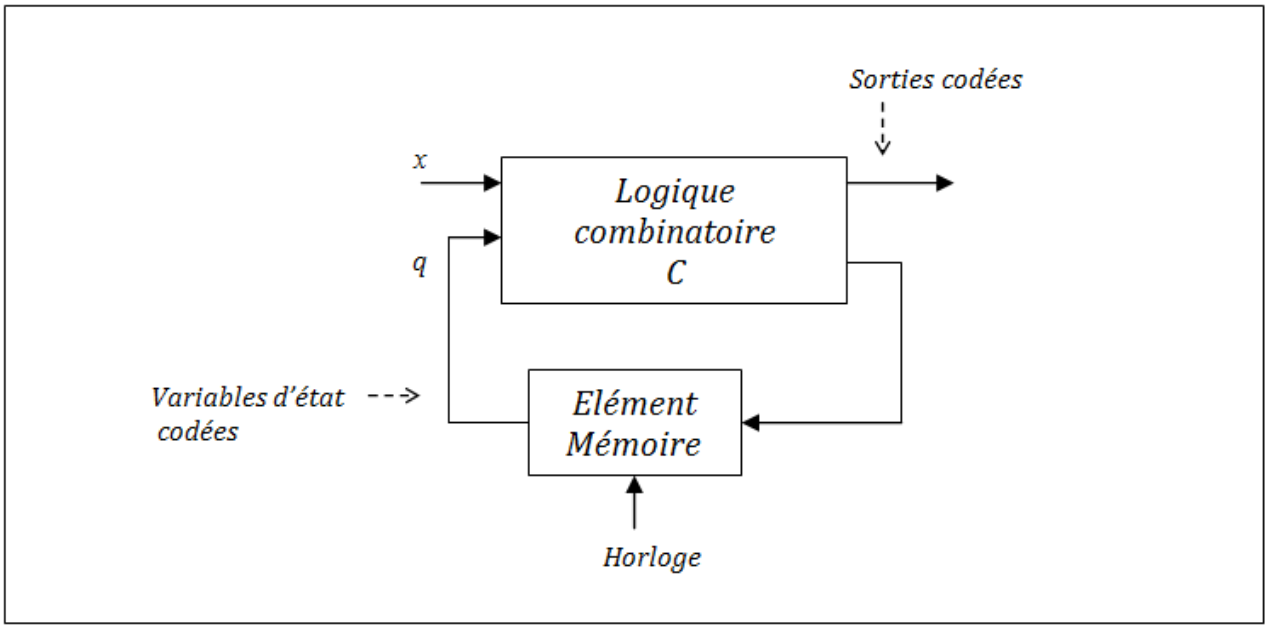


FIGURE 2.10 – Modèle d'un circuit séquentiel

2. Pour n'importe quel état correspondant à un mot non-code résultant d'une faute f dans C, et pour n'importe quelle entrée, l'état futur généré par C avec la faute f est un mot non-code et la sortie est aussi un mot non-code.

À partir de ces conditions, le circuit C sert donc comme un contrôleur de variables d'états codées.

Une conception similaire est illustrée sur la figure 2.11. Les sorties du circuit séquentiel C_1 sont codées en code k/n, qui sont des entrées pour un contrôleur k/n. Ce dernier génère un signal R_s, qui est nécessaire pour la génération du prochain cycle d'horloge (ou du signal d'entrée suivant dans le cas des circuits séquentiels). Ainsi, pour autant de fautes, le système s'arrête aussitôt lorsqu'une sortie erronée est générée [30].

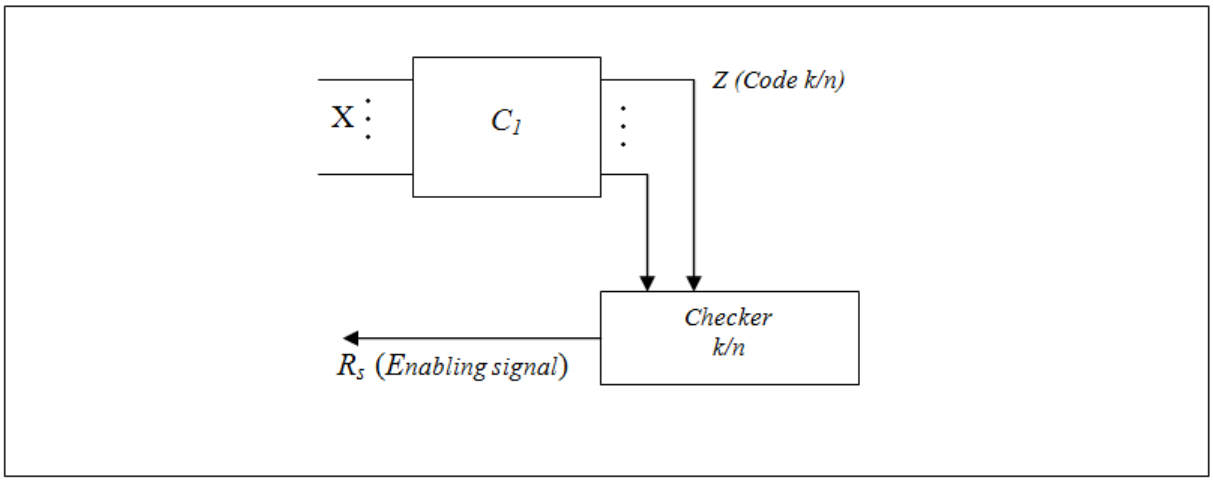


FIGURE 2.11 – Utilisation des codes détecteurs d'erreurs dans les circuits séquentiels

2.5.2.8 Techniques de Duplication

2.5.2.8.1 Duplication Physique

Le schéma de base de la duplication physique est présenté sur la figure 2.12. Pour cette technique, le circuit augmenté consiste en deux copies du circuit sous test (CUT) original. La seconde copie (CUT*) devrait être fonctionnellement équivalente à CUT ; si elle est structurellement équivalente on parle alors d'une duplication *identique*. Autrement, on parle d'une duplication *diverse* [71].

La duplication physique est à "*Fault Secure*" par sa nature pour des fautes de collage simples. Néanmoins, elle est très onéreuse en matière de surface silicium. Pratiquement, cette conception nécessite une introduction plus de 100% d'espace, à cause de la duplication du CUT et des lignes de connexions [71,97,98].

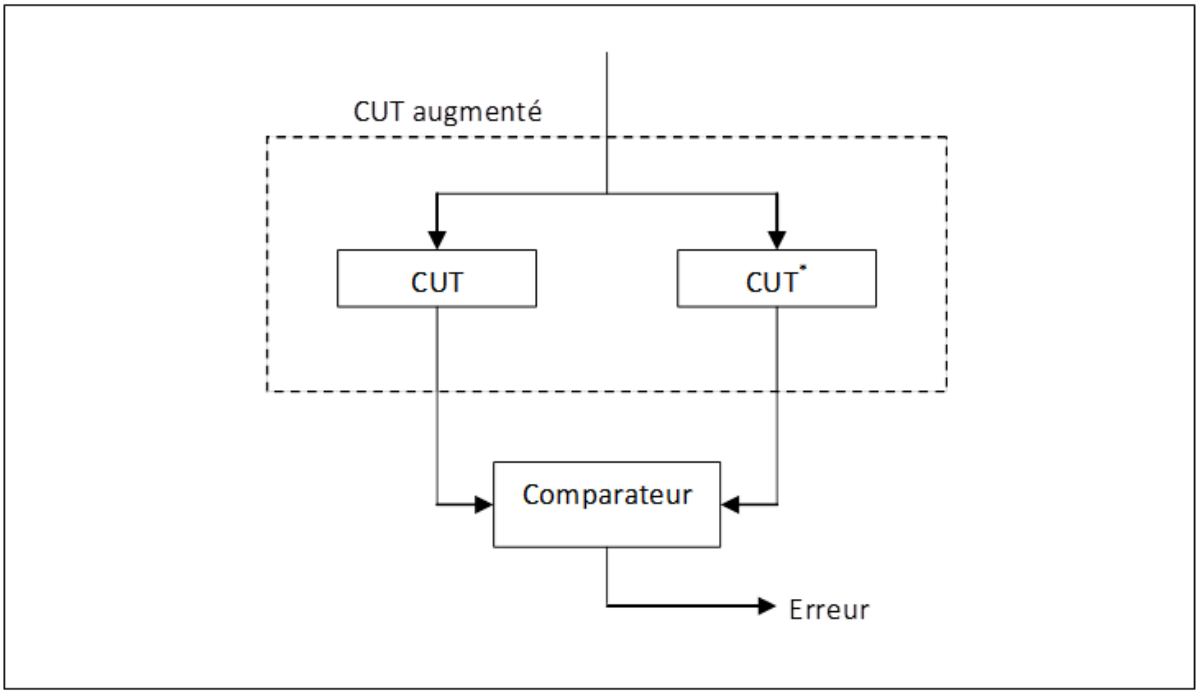


FIGURE 2.12 – Duplication physique

2.5.2.8.2 Technique de duplication des sorties fonctionnelles du CUT

Pour remédier au surplus de surface additionnelle que nécessite la duplication physique, Vitaly [8] a proposé une nouvelle variante de duplication physique où seules les sorties fonctionnelles du CUT sont dupliquées. La figure 2.13 montre un exemple d'application de cette variante à une ALU. Les opérandes d'entrée $A = (a_0, \ldots a_{n-1})$ et $B = (b_1, \ldots b_{n-1})$ sont appliqués aux entrées des cellules ALU $A_0, \ldots A_{n-1}$ qui réalisent des opérations d'addition (ADD) et des opérations logiques (OU , $ET\,et\,OU - Exclusif$) à la sortie $S = (s_0, \ldots . s_{n-1})$ et leur inverses ($\overline{ADD}$, NOR, $NAND$ et $XNOR$) à la sortie inversée $\overline{S} = (\overline{s_{s_0}}, \ldots . \overline{s_{n-1}})$. Les opérations sont sélectionnées au moyen des signaux de contrôle, $m_1, \ldots . m_k$. Chaque cellule ALU, A_i, implémente le retenu propagé,$p_i = a_i \oplus b_i$. Une opération *OU-Exclusif* est appliquée aux retenus propagés de toutes les cellules pour générer la parité, $p(a \oplus b) = p_0 \oplus, \ldots ... \oplus p_{n-1} = a_0 \oplus b_0 \oplus \ldots . \oplus a_{n-1} \oplus b_{n-1}$. Les opérandes d'entrée a et b sont supposés être codés avec code de parité tels que,$p_a = a_0 \oplus \ldots a_{n-1}$ et $p_b = b_0 \oplus \ldots b_{n-1}$ respectivement. Les erreurs dans l'ALU seront détectées [8] en comparant la sortie, S, à la sortie inversée, $\overline{S}$, et la parité, $p(a \oplus b)$, à la parité, $p_a \oplus p_b$.

2.5.2.8.3 Duplication *"Dual-rail"*

La duplication "*Dual-rail*" est une variété de la duplication *diverse*. Dans cette technique, le module CUT (Figure 2.12) ne produit pas la même sortie que le CUT* mais, son

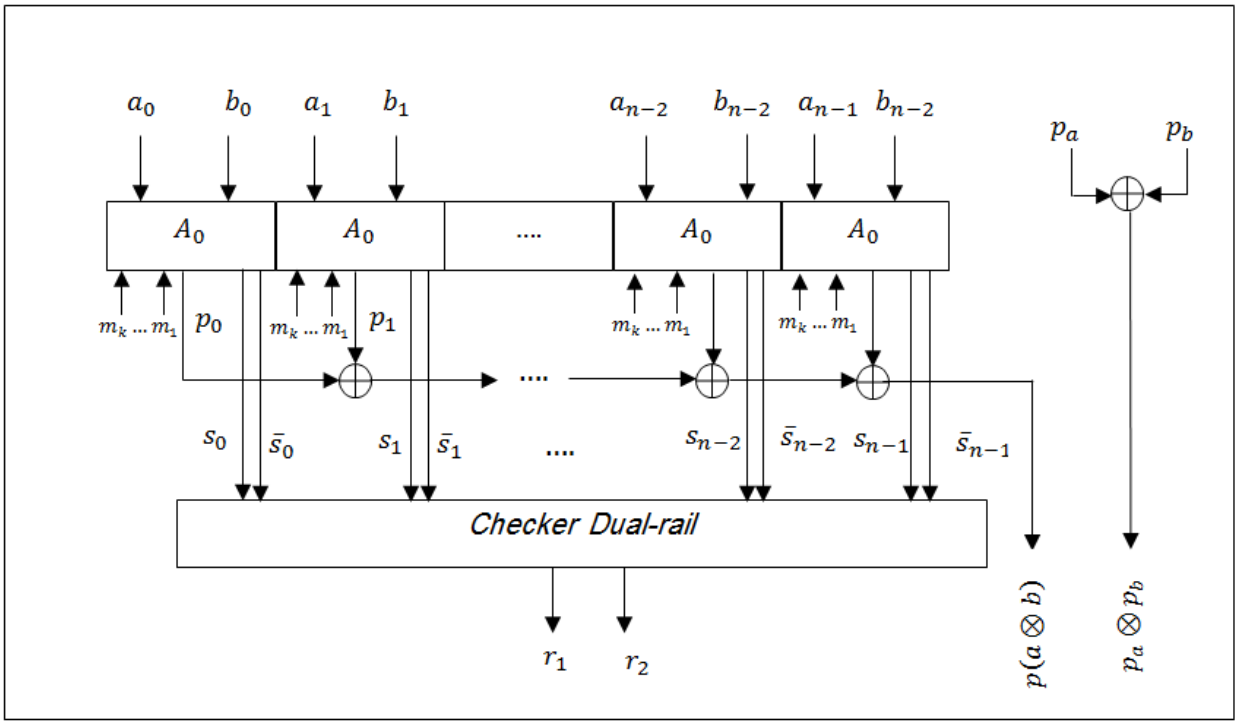


FIGURE 2.13 – Technique de duplication des sorties fonctionnelles du CUT

complément logique (dans le cas correct du système ("*Fault-Free*"). On utilise à cet effet un codeur pour produire les sorties complémentaires [6].

Une autre technique de la même famille plus au moins différente, appelée IFIS (*If It Fails It Stops*), utilise le codage *Dual-rail* des données quoiqu'elle n'obéisse pas au schéma général de la duplication [99,100]. La figure 2.14 représente une portion du système utilisant cette technique. Le système est divisé en éléments IFIS simples, chaque élément est implémenté en utilisant le codage *Dual-rail*. Sous l'opération "*Fault-Free*", exactement un parmi les deux signaux (constituant la sortie *Dual-rail*) change à chaque cycle d'horloge. En particulier, un parmi les deux signaux correspond à la donnée utile, tandis que le second change quand le premier reste stable, de manière à avoir toujours changement dans la sortie "*Dual-rail*". Comme le montre la figure 2.14, chaque élément reçoit le retour de son successeur et les entrées de son prédécesseur. Si un élément échoue de changer sa valeur, à cause de la présence de fautes, alors son successeur et son prédécesseur la détectent et restent également inchangés. Ainsi, de cette façon, le système complet s'arrête tôt [101].

2.5.2.8.4 Duplication Algorithmique

La duplication physique et la conception "*Dual-rail*" sont effectivement "*Fault-Secure*" mais restent très onéreuses. Récemment, une investigation du temps de repos d'un module

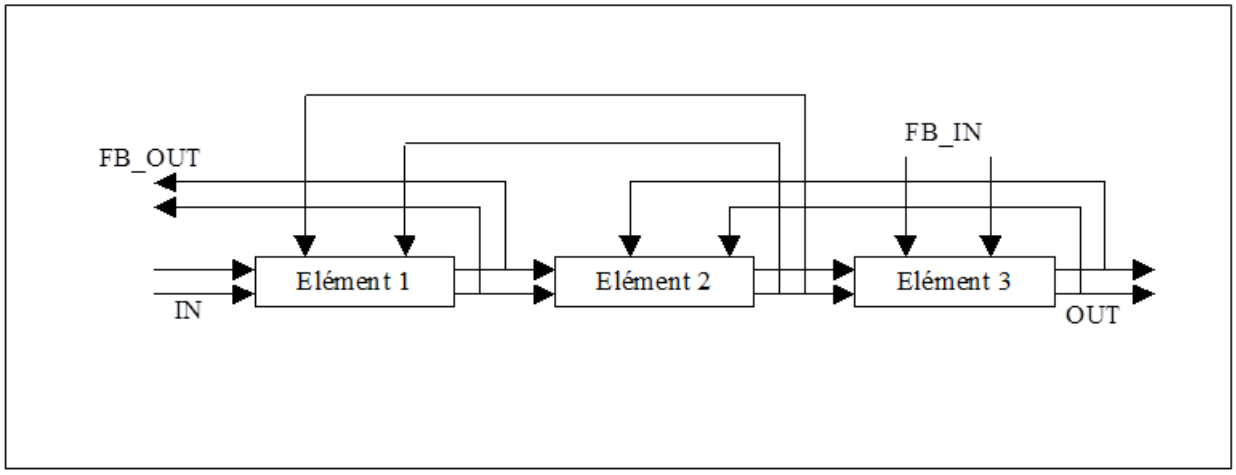


FIGURE 2.14 – La technique IFIS

dans le "*Data-path*" et la possibilité de l'exploiter pour des fins de test, a été menée par plusieurs concepteurs de test [76,102]. Le terme temps de repos réfère aux cycles d'horloge durant lesquels un module fonctionnel ne traite pas des données utiles et ne délivre pas des sorties valides.

A titre d'exemple, considérons le graphe de flot de données DGF (*Data Graph Flow*) de la figure 2.15. Les opérations +1 et +2 sont attribuées au module A_1 (Additionneur), l'opération +3 est attribuée au module A_2 (Additionneur) et l'opération *1 est attribuée au module M_1 (Multiplieur). Le "*Data-path*" nécessite donc deux additionneurs et un multiplieur. Physiquement, dupliquer ces modules résulte, évidemment, en quatre additionneurs et deux multiplieurs. Cependant, le module A_2 est au repos durant l'étape de contrôle 1 et 3, alors que le module A_1 est aussi au repos à l'étape 3. Donc, on peut utiliser le module A_2 durant l'étape 1 et dupliquer l'opération +1. De manière similaire, on peut dupliquer les opérations +2 et +3 durant l'étape 3 en exploitant les modules A_1 et A_2 respectivement. Ceci introduit une latence de faute d'un cycle d'horloge sans l'ajout du hardware. Afin de vérifier l'opération *1, le seul choix qui reste est d'introduire un nouveau multiplieur. Ainsi, le résultat de vérification est accompli avec l'utilisation de deux additionneurs, deux multiplieurs, quelques multiplexeurs, des registres et des interconnexions, avec une latence de faute d'un cycle d'horloge.

Cet exemple illustre les propriétés de la duplication algorithmique. Dans cette technique, c'est l'opération qui est dupliquée et non l'opérateur. Il est à noter que l'opération originale et sa duplication seront programmées sur des hardwares disjoints, pour garantir la propriété "*Fault-Secure*". De plus, il sera avantageux qu'elles soient attribuées, dans la mesure du possible, à des hardwares divers. Quelques latences de fautes seront aussi atten-

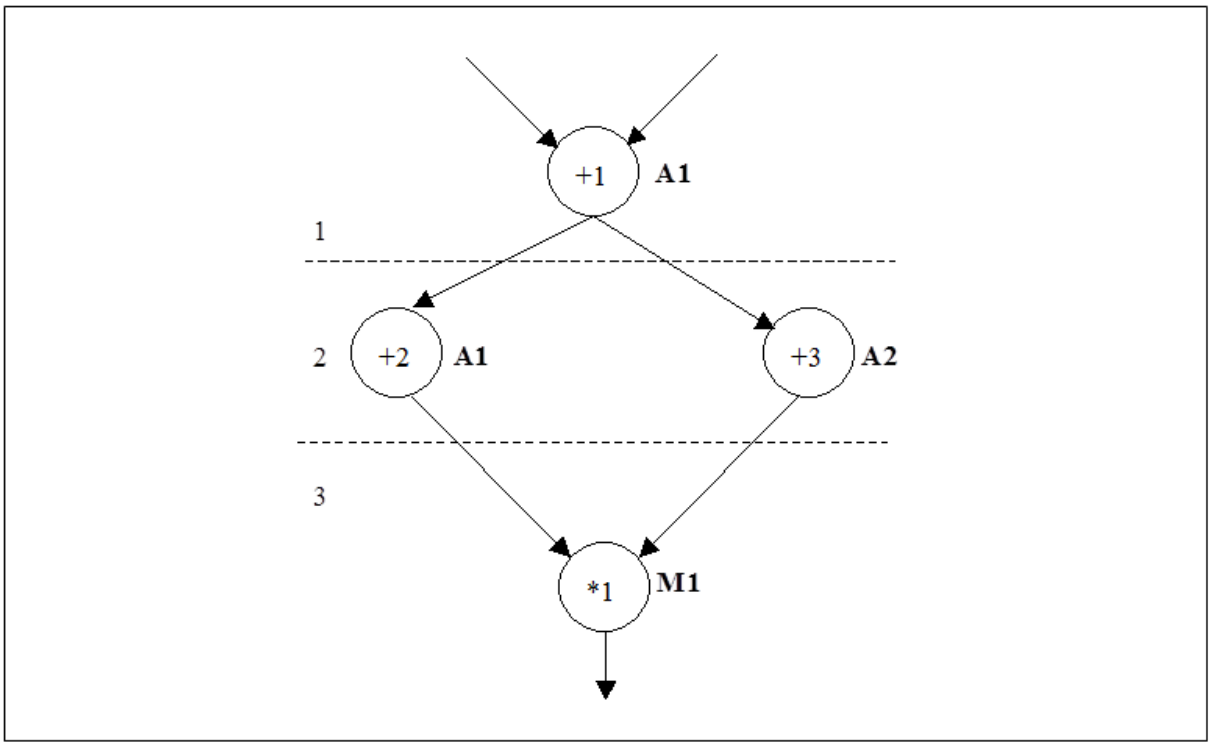


FIGURE 2.15 – Exemple.1 de DFG

dues si le temps de repos n'est pas suffisant. Dans ce cas, il faudra soit ajouter du hardware, soit tolérer une dégradation de performances. Par conséquent, l'outil permettant l'implémentation de ces duplications devrait être capable de prendre de telles décisions (ajouter du hardware ou tolérer une dégradation de performances, ou encore introduire une latence d'erreur [71]).

2.5.2.8.5 Technique de duplication Inverse ("*Inversion Testing*")

Contrairement à la duplication physique, dans cette technique, le module INV(CUT) réalise l'opération inverse (Figure 2.16). L'opération et son inverse peuvent être arithmétiques, telles que l'addition et la soustraction, ou des opérations au niveau registres telles que le décalage à droite ou à gauche. Les sorties fonctionnelles sont comparées aux sorties inversées, et dans le cas "*Fault-Free*", elles devraient être égales. Dans le cas contraire, le contrôleur signale la présence de défauts [71,103].

La technique d'inversion est "*Fault secure*", par sa nature, pour des fautes de collage simple. En général, inverser un module à des fins de tests n'apporte aucun avantage, par opposition à la duplication physique. Cependant, si le module INV(CUT) est au repos durant quelques cycles d'horloge particuliers, alors cette technique conduit à une implémentation hardware efficace et ainsi, moins de silicium est nécessaire. Un cas simple pour

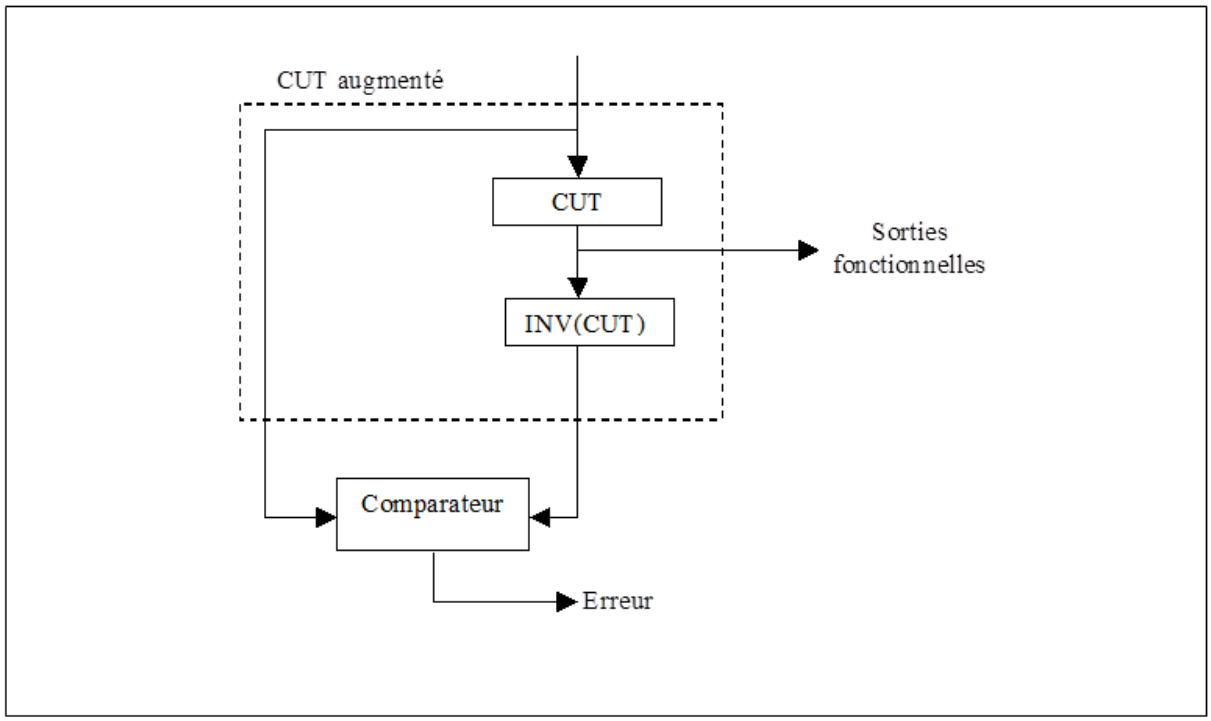


FIGURE 2.16 – La technique d'inversion

illustrer ce concept est montré sur la figure 2.17. Le DFG consiste en une opération addition, une opération soustraction et en deux opérations de multiplication. Le circuit est implémenté par un additionneur A_1, un soustracteur S_1 et deux multiplieurs M_1 et M_2. On constate que l'additionneur A_1 est au repos durant l'étape de contrôle C_2 et C_3, tandis que le soustracteur S_1 est repos durant C_1 et C_2. L'opération +1 peut être inversée par le soustracteur S_1 à l'étape C_2, alors que l'opération –1 peut être aussi testée durant $C2$ par A_1 (les opérateurs sont supposés à boucle fermée pour qu'après l'étape C_3, le contrôle revient à l'étape C_1). Ainsi, les résultats de chaque opération sont vérifiés sans le besoin d'introduire d'autres additionneurs ou soustracteurs. Ce qui n'est pas le cas dans la duplication physique qui impose la duplication de tous les modules (Additionneur, Soustracteur, et les deux multiplieurs). Dans ce cas précis, la technique d'inversion réduit la duplication des modules fonctionnels par un facteur de 50% [71].

Néanmoins, l'opération inverse est réalisée après l'opération originale, contrairement à la duplication physique, qui se fait simultanément. Cependant, moyennant quelques cycles d'horloge de relaxation et un outil de synthèse puissant, l'opération et son inverse peuvent être réalisés dans la même étape de contrôle [71].

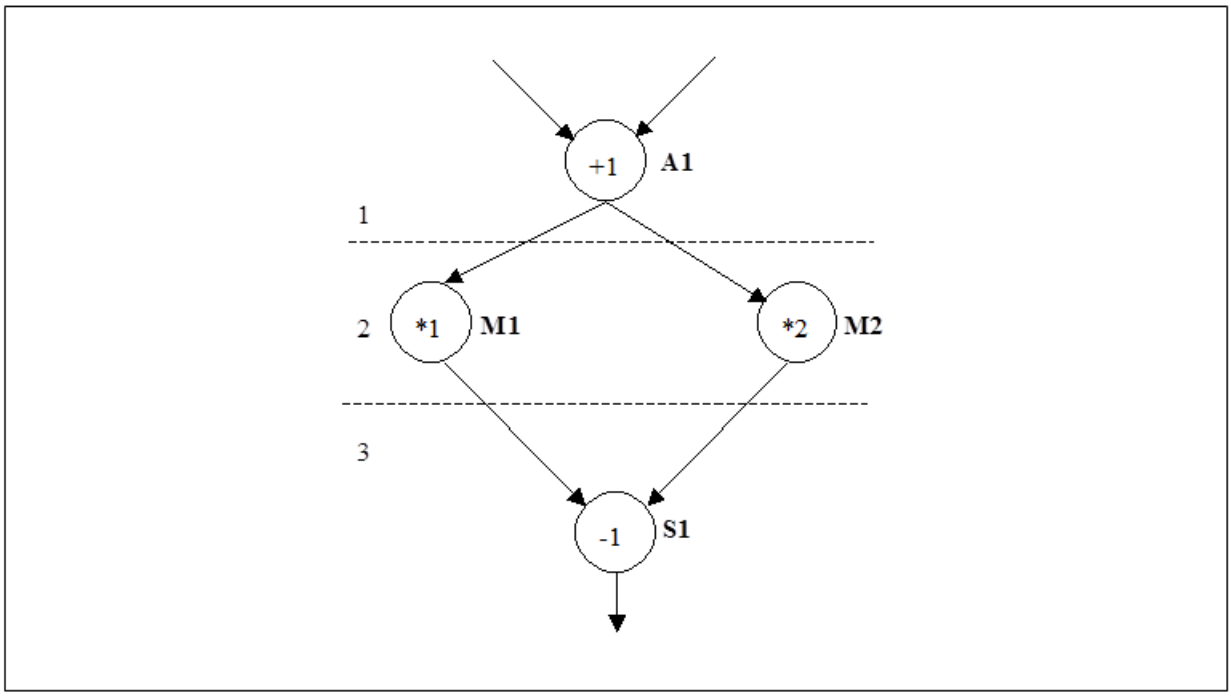


FIGURE 2.17 – Exemple.2 de DFG

2.5.3 On-line BIST et DFT

Les techniques BIST et DFT ne sont pas de nouveaux concepts. Une variété d'approches a été développée dans littérature [3,4,6,30,104,105]. En général, des ressources dédiées pour le test *Off-line* sont aussi exploitées pour réaliser des tests en ligne, en se basant sur le principe du partage des ressources hardwares.

2.5.3.1 On-line BIST

Pour des systèmes critiques et à hautes disponibilités, une approche exhaustive qui couvre toutes les fautes permanentes, intermittentes et transitoires est essentielle. Durant cette dernière décennie, le BIST (*Built-In Self-Test*) ou test intégré, a émergé comme une technique importante de test [3,88,106] valable aussi bien pour le test de production que pour le test en ligne.

Le BIST est un concept DFT [3] qui consiste à intégrer, sur le CUT, de la logique supplémentaire dédiée au test du CUT lui-même, comme le montre la figure 2.18.

Durant l'opération normale, le CUT reçoit ses entrées à partir des autres modules et réalise la fonction pour laquelle il a été conçu. Dans le mode test, un générateur de test applique une séquence de test au CUT et, le moniteur de réponse évalue la réponse. Dans les BIST courants, le moniteur de réponse compacte les résultats pour former la signature

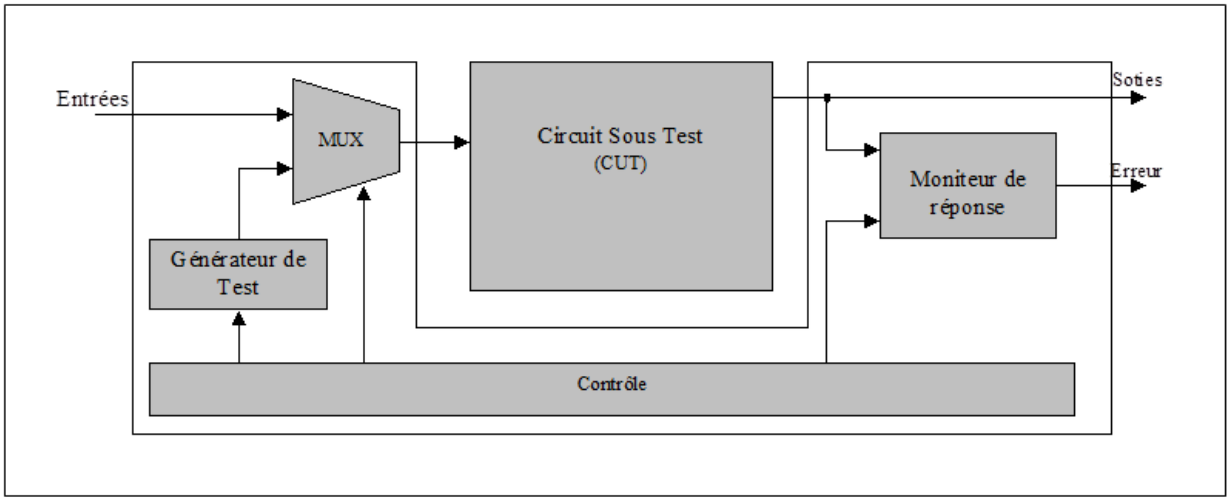


FIGURE 2.18 – BIST générique

qui sera comparée à une signature référence.

Les concepteurs de systèmes peuvent utiliser le BIST pour le test en ligne non concurrent des systèmes logiques et mémoires [4]. Ils peuvent configurer le hardware BIST pour des tests sporadiques conduisant ainsi le contrôle BIST au mode *RESET* du système tel que ce test apparaît dans la phase de démarrage ou à l'arrêt. Toutefois, le BIST peut être conçu pour des tests périodiques avec une latence de faute faible. Ceci nécessite l'incorporation de processus de test [3] garantissant la détection de fautes cibles dans un temps fixe.

Le test intégré présente de ce fait les avantages suivants :

- ✓ Suppression de la nécessite d'un testeur externe couteux ;
- ✓ Possibilité de test à haute fréquence et à la vitesse nominale de fonctionnement du circuit,
- ✓ Taux de couverture élevé ;
- ✓ Temps de test court en utilisant à la fois le parallélisme et la hiérarchie du circuit avec une vitesse de test égale à la vitesse nominale du circuit ;
- ✓ Possibilité de tester le circuit en phase de maintenance mais aussi en opération en utilisant les temps de repos du système (cas des systèmes temps réel en particulier).

En revanche, l'utilisation du test intégré conduira obligatoirement à un surcoût en matériel s'accompagnant d'une perte de performance.

2.5.3.2 BIST Concurrent (CBIST)

Saluja a proposé dans [77] une technique qui exploite les ressources *Off-line* pour le test en ligne. Elle se base sur l'extension du concept *Off-line* BIST pour introduire le BIST en ligne. L'architecture du circuit BIST concurrent proposée dans [30,77] est illustrée sur la figure 2.19. Le circuit CUT doit être de logique combinatoire et, le hardware BIST est séparé du circuit normal. Cette technique peut être utilisée pour des circuits séquentiels si ces derniers peuvent être divisés en blocs combinatoires. Ils peuvent être alors testés comme des entités séparées.

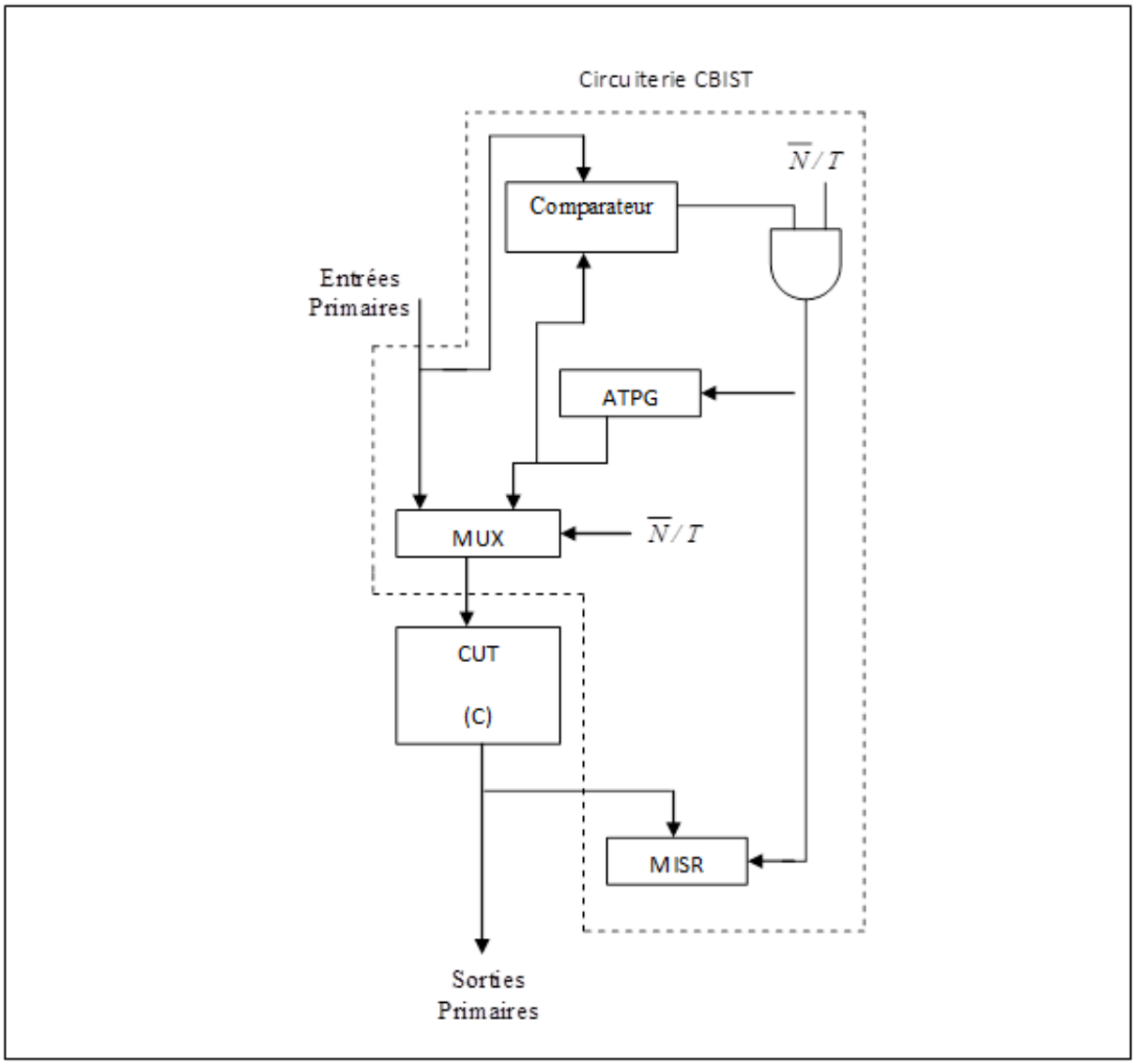


FIGURE 2.19 – Schéma synoptique de la circuiterie CBIST

Dans le mode *Off-line*, l'ATPG (*Automatic Test Pattern Generator*) et le MISR (*Mul-*

tiple Input Shift Register) réalisent leur fonction usuelle et le CUT reçoit les entrées de l'ATPG. Dans le mode en ligne, le contenu de l'ATPG et les entrées fonctionnelles sont comparés. S'ils sont égaux, le MISR avance au prochain vecteur de test ("*Test pattern*"), sinon, les ressources de test restent au repos. En effet, dans cette technique, le temps nécessaire pour que l'ATPG parcourt tous les états possibles (définissant la latence de faute), est très élevé et dépend des entrées fonctionnelles du circuit. Le CBIST est considéré, par la majorité des chercheurs, comme une approche classique et, probablement, la première tentative dans ce domaine.

2.5.3.3 BIST Unifié (UBIST)

Si l'on voulait assurer d'une façon efficace tous les types de test nécessaires aux circuits intégrés, on serait amené à unifier le test hors-ligne et le test en ligne dans les circuits. L'intégration des 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. En effet, les études présentées dans [107,108] ont apporté des résultats prometteurs dans ce domaine en aboutissant à la technique de BIST unifié, nommé UBIST (*Unified Built-In Self-Test*) dont le schéma de principe est représenté à la figure 2.20.

Cette technique "UBIST" assure la détection de pannes dans le circuit, aussi bien pendant la phase de fonctionnement normal (test en ligne) que pendant la phase de test hors-ligne, en autorisant la génération interne de séquences de vecteurs de test qui vérifient le bon fonctionnement des blocs fonctionnels des contrôleurs.

Les mécanismes qui sont capables de délivrer les séquences de vecteurs de test hors-ligne désirées peuvent être réalisés selon la procédure BILBO (*Built-In logic Block Observation*). Ces éléments ont plusieurs modes de fonctionnement dont les détails seront donnés dans le prochain chapitre. Ils sont parfaitement adaptés au type de test hors-ligne à effectuer dans le cadre d'une conception UBIST. Pour cette dernière approche, les mécanismes BILBO subissent quelques modifications de structure pour devenir des registres UBILBO [107,109,110].

Dans sa globalité, le test hors-ligne est réalisé en trois phases distinctes : *TEST1*, *TEST2* et *TEST3*. Le bon fonctionnement du contrôleur global qui compacte les signaux d'indication d'erreurs (E_1,.....E_k) est vérifié pendant la phase *TEST3*. Durant la phase *TEST1*, les UBILBO *impairs* délivrent des vecteurs de test sur les entrées des blocs fonctionnels *impairs* et sur les entrées des contrôleurs *pairs*. Les réponses de chacun des blocs *impairs* sous test sont vérifiées par les contrôleurs *impairs* correspondants mais elles sont aussi compactées, en vue d'obtenir la signature finale, par les UBILBO *pairs* associés. La défaillance d'un contrôleur *pair* pendant cette phase de test est immédiatement signalée

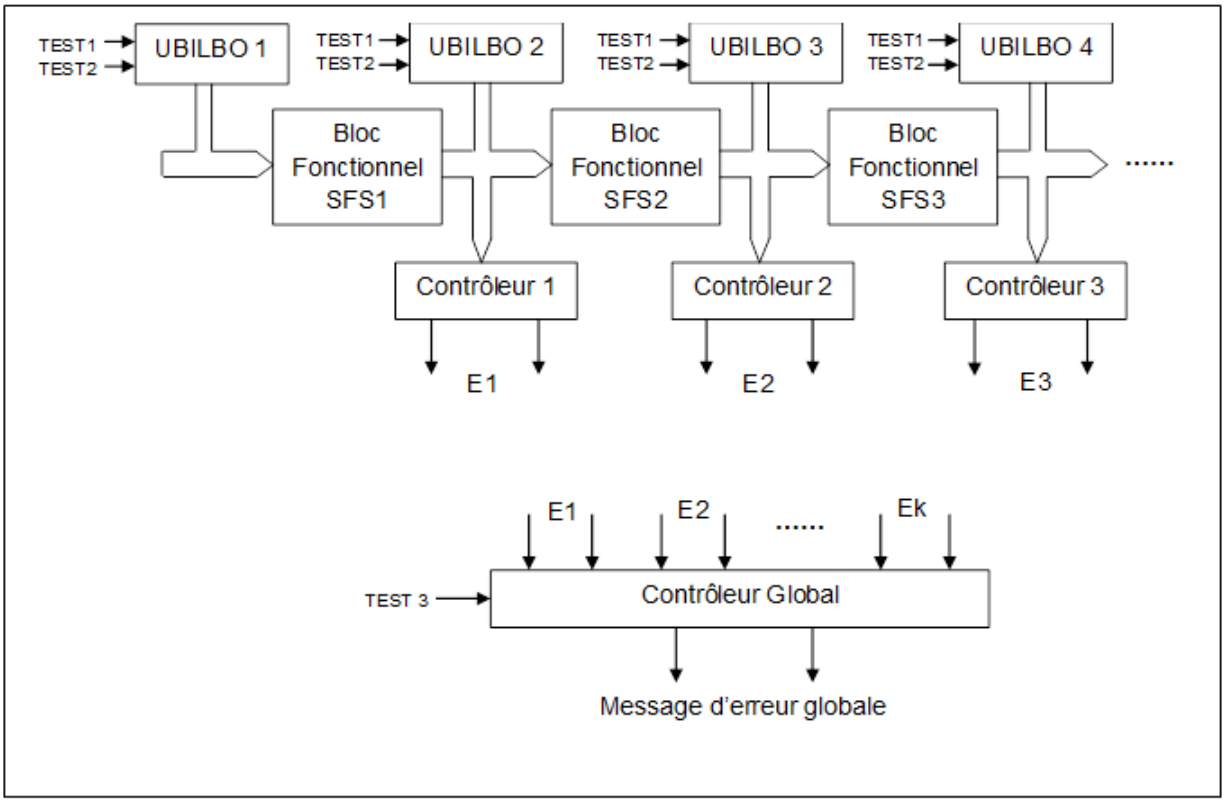


FIGURE 2.20 – Schéma de principe d'une conception UBIST

par le contrôleur *global*. A la fin de la période *TEST1* les signatures des différents UBILBO *pairs* sont contrôlées. En outre, le principe est le même pour la phase *TEST2* où les blocs fonctionnels *pairs* et les contrôleurs *impairs* sont vérifiés.

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.

2.6 Conclusion

Ce chapitre de revue de littérature a porté sur les principales techniques de test en ligne existantes. En fait, les techniques "*Self-Checking Design*" et *"On-line BIST"* sont les plus adoptées dans ce domaine. Bien que la technique "*Self-Checking Design*", qui exploite les propriétés de codes détecteurs d'erreurs pour la détection concurrente d'erreurs (avec une latence d'erreur faible), est moins couteuse en termes de surface additionnelle requise à son implémentation, elle a une faible couverture de faute. En revanche, la technique BIST offre une haute couverture de faute aux détriments d'un surplus de surface additionnelle et d'une latence d'erreur relativement grande. Une technique de test en ligne est plus efficace

si elle assure d'une part, une haute couverture de fautes avec une aptitude de détection immédiate (latence de faute faible) et si elle ne nécessite pas de redondance hardware d'autre part.

Par ailleurs, certains systèmes, fonctionnant dans des domaines d'applications critiques où la sécurité est le facteur principal, ne tolèrent pas l'interruption de leur fonctionnement normal même quelques secondes par année. Ceci exige une vérification en temps réel de l'intégrité de ces systèmes. Or, les systèmes embarqués sont devenus aujourd'hui la base de ces applications critiques. Le test de ces systèmes est de plus en plus difficile, surtout quand il s'agit de détection de fautes transitoires, car ces systèmes sont de plus en plus complexes en termes d'architecture et de méthodologie de conception.

Nous présentons, dans le chapitre suivant, une nouvelle technique dédiée au test en ligne des systèmes embarqués, qui assure une détection immédiate de fautes de collage simples et répond ainsi aux exigences du test en ligne de tels systèmes.

Chapitre 3

DÉVELOPPEMENT D'UNE NOUVELLE APPROCHE DE TEST EN LIGNE POUR DES SYSTÈMES EMBARQUÉS

Résumé *: Dans ce chapitre, une nouvelle approche dédiée au test en ligne des systèmes embarqués, basée sur l'utilisation de la norme Built-In-Self-Test est présentée et de nouveaux registres BILBO optiques sont proposés. La stratégie mise au point adaptée à la technique de test en ligne est aussi présentée.*

3.1 Introduction et motivation

Les systèmes embarqués sont des systèmes électroniques et informatiques autonomes, construits pour effectuer des tâches précises. Cette autonomie nécessite qu'ils soient capables d'interagir avec leur environnement et de gérer leurs ressources disponibles (puissance de traitement, capacité de stockage, énergie, etc.). On trouve des systèmes embarqués partout. Au-delà du militaire, de la production de l'énergie, du spatial, de la productique, de tels systèmes sont aussi largement déployés dans les transports : aérien, routier, ferroviaire, naval. Dans ces domaines où l'erreur peut avoir de graves conséquences, le processus de développement est particulièrement rigoureux — tant des composants matériels et logiciels que de leur interconnexion — mettant de plus en plus en jeu des techniques formelles. La sûreté de fonctionnement est aussi obtenue en incorporant des mécanismes de détection d'erreur et de reconfiguration en ligne permettant d'améliorer la résilience des comportements. Lorsqu'une faute survient, qu'elle ait été anticipée, simplement prévisible, ou même non prévue (perturbations physiques, activation de bogues, méprises,...) le système doit

être capable de se remettre en route [111]. De ce fait, une vérification continuelle des résultats produits et la possibilité de prédiction de défaillances augmentent la fiabilité et la sûreté de fonctionnement de ces systèmes.

Par ailleurs, la mise en œuvre d'une technique de test qui permet de satisfaire ces besoins dépend fortement des objectifs visés prédéfinis en plus de la stratégie à mettre en œuvre. Cette stratégie est efficace dans la mesure où elle permet d'appliquer des tests et de prélever des résultats sans perturber ou ralentir le bon fonctionnement de ces systèmes.

Les deux aspects clés, dans la philosophie de test, sont la contrôlabilité et l'observabilité. Aujourd'hui, les vecteurs de tests sont générés par des algorithmes très complexes en utilisant différentes méthodes (BIST, Boundary Scan,...). De plus, les concepteurs de test, devraient assurer non seulement l'activation des fautes ciblées, mais aussi leur propagation aux soties primaires pour les rendre observables. En pratique, les broches E/S dédiées au test sont peu nombreuses (car l'ajout de broches E/S, pour des fins de test, augmente dramatiquement le coût de la conception originale) [30,31,43]. Cette situation devient très délicate, notamment quand il s'agit d'assurer un test en ligne où l'intervention humaine devrait être réduite au strict minimum. C'est pour cela que depuis un certain nombre d'années, les concepteurs de test ont considéré des solutions optiques reposant sur l'insertion de sondes optiques dans les circuits sans modifier le nombre des E/S du circuit original [12,112]. Ces sondes optiques intégrées, assurent l'émission des signaux optiques caractérisant les états des signaux électriques. En effet, récemment, un modèle de sonde optique a été développé au laboratoire LCIS répondant aux exigences de test en ligne de la nouvelle génération des circuits GSI [20]. Par ailleurs, l'emplacement de ces sondes dans la structure du circuit cible doit être fait avec une attention toute particulière pour aboutir à une solution optimale en termes de surface additionnelle requise à l'intégration de ces sondes optiques. De ce fait, nous nous proposons des registres BIST spécifiques comme plate-forme d'intégration de ce type de sondes optiques afin de mettre en œuvre une solution de test pertinente. D'où, un schéma et une stratégie de test en ligne optiques sont proposés et explicités en détail dans la suite de ce chapitre.

3.2 Principe de la nouvelle approche de test en ligne proposée

Les techniques de test en ligne idéales doivent avoir une couverture de faute de 100%, un cycle d'horloge de latence d'erreur, aucune redondance de type "*Space redundancy*" ni de "*Time redundancy*". Au global, ces techniques n'exigent donc aucune modification

majeure de la conception originale et n'imposent aucune restriction de type structurel ou fonctionnel sur les CUT [3]. La majorité des méthodes BIST courantes répondent à certaines de ces contraintes sans pouvoir toutefois les adresser ensemble [4]. En général, la considération de ces quatre paramètres dans une conception engendre des conflits. Notons qu'une couverture de faute élevée exige une latence d'erreur, une *"Space redundancy"* et/ou une *"Time redundancy"* élevées. Les techniques dotées de moyens de détection immédiate (latence d'erreur =1) minimisent le "*Time redundancy*" mais nécessitent du hardware en plus. Tandis que, un schéma avec détection avec retard (latence d'erreur >1) réduit le time et "*Space redundacy*" au détriment d'une augmentation de la latence d'erreur. Plusieurs techniques avec détection avec retard proposées dans la littérature [3,77], considèrent des combinaisons d'entrées équiprobables et essaient d'établir un extremum probable de la latence d'erreur. Les résultats qui en découlent ont montré que certaines fautes restent non détectées pour une longue période de test car les vecteurs qui les détectent apparaissent rarement aux entrées des CUT.

Toutefois, il faut rappeler que la réduction de la latence d'erreur est considérée souvent comme principal objectif des techniques de test en ligne [5] surtout quand il s'agit de systèmes embarqués dédiés à des applications critiques. En fait, les techniques de duplication ou triplication sont souvent adoptées [6-9] car elles ont une latence d'erreur très faible par rapport aux techniques décrites dans le chapitre précédent. Par ailleurs, la disponibilité des composants en étalage (*Component-Off-The-Shelf* (COTS)) à faible coût et la possibilité d'intégrer facilement une grande quantité de ces derniers dans les systèmes sur puce actuels font de la duplication matérielle une solution viable dans de nombreux domaines où la fiabilité du système est une préoccupation majeure, même au détriment de la surface supplémentaire [10].

La technique de duplication matérielle consiste à dupliquer le module sous test et comparer ses sorties avec celles du module dupliqué. Cependant, si une erreur est survenue dans l'un des modules, l'erreur ne peut être détectée qu'après s'être propagée jusqu'aux sorties primaires. Or, les circuits de nos jours intègrent plusieurs modules dans leur architecture et le temps de propagation de l'erreur à travers ces modules jusqu'aux sorties primaires peut être prohibitif, ce qui augmente davantage la latence d'erreur. C'est pourquoi on a eu l'idée de combiner la technique de duplication matérielle et le BIST avec l'ajout de sondes optiques dans son architecture pour réduire considérablement la latence d'erreur et ainsi augmenter le pouvoir de détection d'erreur transitoires qui apparaissent et disparaissent rapidement, pour assurer un fonctionnement fiable des systèmes dédiées aux applications critiques. En d'autres termes, la technique proposée à l'issue de ce travail de recherche, exploite d'une part l'avantage de la technique de duplication qui ne nécessite

aucune génération de vecteurs de test car les entrées primaires sont elles-mêmes les vecteurs de test. D'autre part, on exploite le BIST, adopté par la majorité des concepteurs des circuits actuels, pour accéder aux modules internes et réaliser ainsi des opérations de scan des différents modules (*"Core"*), en exploitant les signaux optiques issus des sondes optiques intégrées dans les registres BIST.

Cette approche de test en ligne repose sur l'utilisation du fameux (*Off-line*) BIST qui est le *Block-In Logic Observation* (BILBO) et l'exploitation de l'un de ses modes de test, en l'occurrence le mode chargement parallèle (*Parallel Load mode* en anglais), qu'offre cette architecture [30,41,104,105,113,114]. De ce fait, nous proposons de nouvelles cellules BILBO Optiques (OBILBO : *Optical BILBO*). Les cellules BILBO sont modifiées afin d'intégrer l'aspect optique réalisant ainsi notre OBILBO [115].

La technique de test exploitée utilise la duplication matérielle du circuit sous test. Les signaux optiques issus des OBILBO de deux applications identiques, après une conversion en signaux électriques, sont continuellement comparés. Un contrôleur génère un signal d'erreur, indiquant ainsi l'occurrence d'erreurs, dans le cas où les réponses des deux applications diffèrent. Notons que cette technique optique est valable aussi bien pour le test en ligne des cartes que pour des systèmes supportant l'architectures BILBO et peut être étendue à d'autres techniques (telle que la duplication algorithmique). Le mécanisme de test adoptant l'approche proposée est simple et se résume comme suit :

- Configuration des registres BILBO de la carte (système) en mode de chargement parallèle ;
- Exécution de la tâche pour laquelle la carte a été conçue ;
- Comparaison concurrente des signaux issus des registres OBILBO reflétant les états logiques de données sur les E/S des modules internes de l'application et son module dupliqué ;
- En cas de présence de défauts signalés par le comparateur, une opération Scan est alors nécessaire pour vérifier l'intégrité des registres OBILBO qui peuvent être, éventuellement, la cause de défauts ;
- Sinon, la réalisation des opérations de scan de chaque module séparément, pour identifier le module défaillant, est entreprise.

Dans le paragraphe suivant, nous présentons le côté hardware permettant l'implémentation de cette stratégie de test en ligne optique.

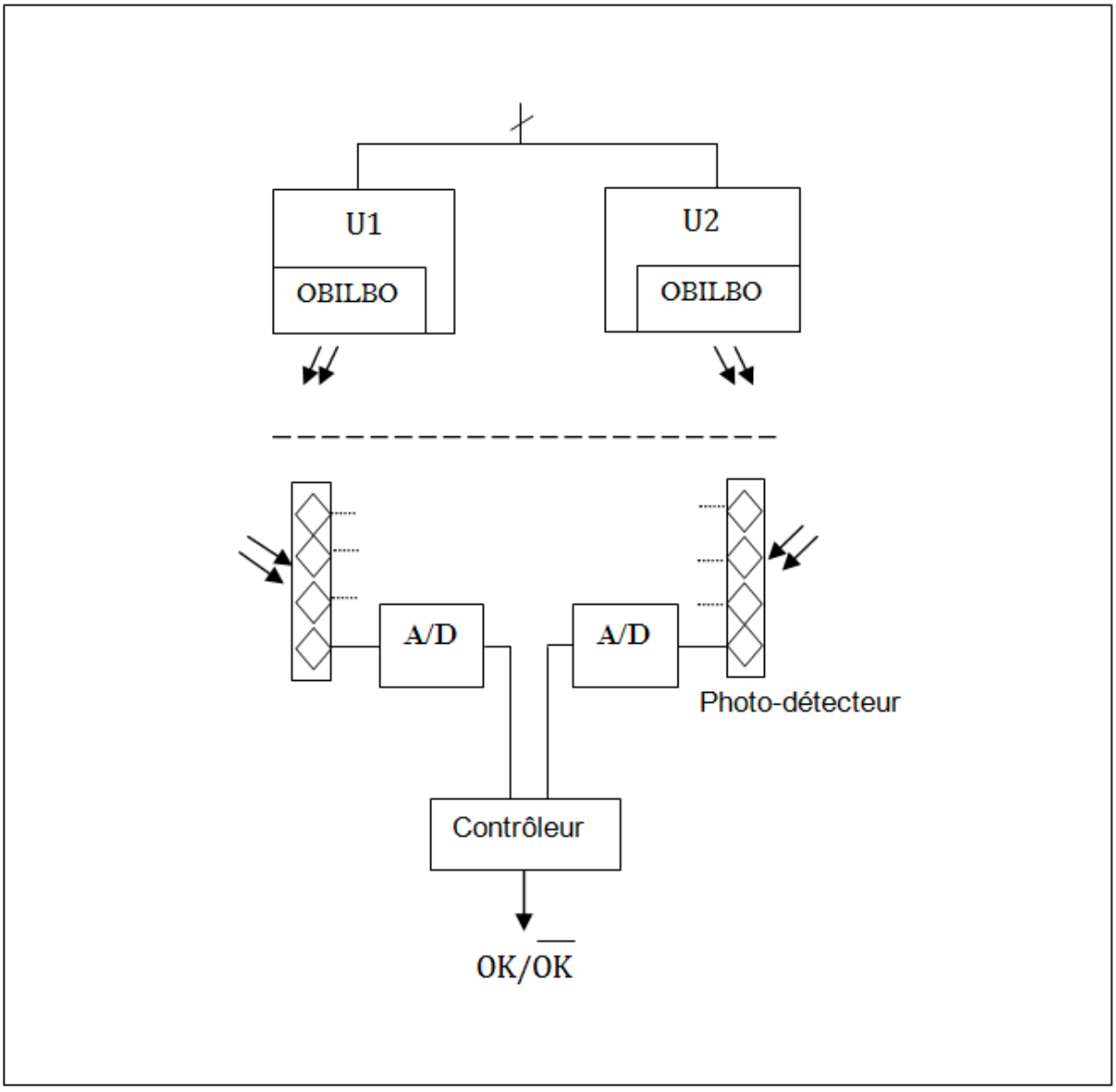


FIGURE 3.1 – Schéma de principe de la nouvelle approche de test en ligne proposée.

3.3 Description du schéma de principe de l'approche de test en ligne proposée

Le schéma de principe est illustré sur la figure 3.1 et comporte deux unités identiques recevant les mêmes entrées durant l'opération normale. Les données disponibles sur les sorties des registres OBILBO, configurées en mode de chargement parallèle, sont capturées par les Sondes Optiques (SO) considérées dans leur architecture. Selon les valeurs présentes sur les sorties de ces cellules, les OBLIBO refléteront, en conséquence, des rayons optiques

appropriés. Ces rayons optiques sont traités par des capteurs simples en l'occurrence des photodiodes. Un Convertisseur Analogique/Numérique (CAN) est alors nécessaire pour délivrer des signaux d'entrées numériques au comparateur. Dans ce qui suit, nous détaillons le rôle et la ou les fonctions réalisés par chacun des blocs fonctionnels constitutifs du système.

3.3.1 Modules U1 & U2

Les modules U1 et U2 peuvent être des circuits simples ou des cartes ou encore des systèmes supportant la nouvelle architecture BIST. Cette architecture est identique à l'architecture conventionnelle où seules les cellules BILBO sont désormais des cellules optiques. Notons que la structure BILBO est un cas particulier du BIST faisant appel à des registres spéciaux qui seront détaillés plus loin dans ce chapitre. La figure 3.2 représente l'architecture générale d'un schéma BIST [62].

Le BIST implique l'ajout de trois blocs supplémentaires (Figure 3.2) au composant devant être testé :

1. Un générateur de test (*Test Pattern Generator* : TPG) qui produit les vecteurs de test pour le circuit ;
2. Un analyseur de réponse qui vérifie que la réponse est conforme à la donnée attendue (*Response Verification*) ;
3. Un contrôleur de test qui active les différentes phases de test (*Test Controller*).

Les solutions de génération de test couramment employées sont :

- Des ROM avec microprogrammes pour un test déterministe ;
- Des compteurs binaires pour des tests exhaustifs ou pseudo-exhaustifs ;
- Des registres à décalages à rétroaction linéaire (LFSR : *Linear Feedback Shift Register*) et des automates cellulaires pour du test pseudo-aléatoire ou pseudo-exhaustif.

D'autres solutions de génération de vecteurs de test adaptés à certaines parties spécifiques d'un circuit (notamment les chemins de donnée) existent. Néanmoins, l'approche la plus employée [116] est la génération de vecteurs pseudo-aléatoires qui utilisent des LFSR.

Quand on applique un vecteur de test sur un circuit, on doit connaître la réponse correcte pour pouvoir effectuer la comparaison. On peut stocker cette réponse sur des ROM mais cela pose un problème de surplus de matériel. Une autre solution consiste à compresser ou compacter les réponses. On obtient ainsi une "*signature*" du circuit, sous forme d'un nombre restreint de données. La compaction entraine parfois des problèmes d'"Aliasing", c'est-à-dire de perte d'information, qu'il faudra minimiser.

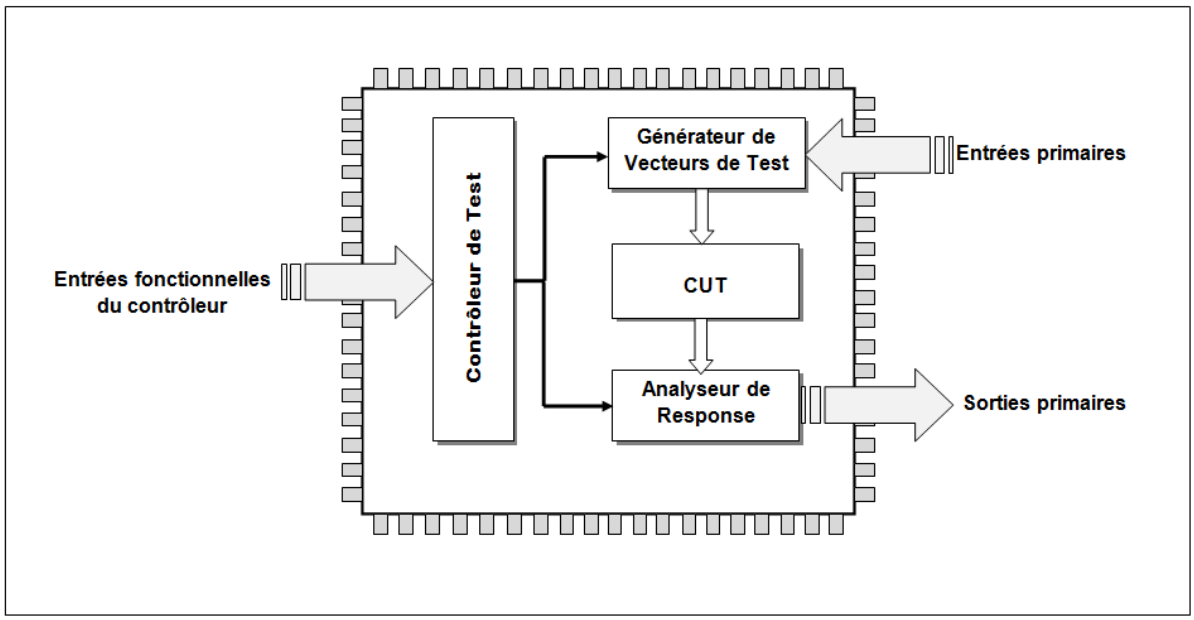


FIGURE 3.2 – Architecture BIST.

Les principales méthodes de compaction sont [30] :

- Le comptage de transition (nombre total de transition de 0 à 1 et 1 à 0) ;
- L'analyse de signature réalisée par LFSR. La longueur de la compaction est alors la longueur (nombre de bascules) du LFSR. Si la compaction se fait sur plusieurs entrées en parallèle, on parle alors de MISR (*Multiple Input Shift Register*) ;
- Le comptage de syndrome (nombre de 1 de la séquence réponse).

Vu que le LFSR est un élément clé dans un schéma BIST, nous allons l'aborder avec plus de détails dans le paragraphe suivant.

3.3.1.1 Générateur des vecteurs de test

Hormis le cas spécifique où l'on dispose d'un processeur interne au circuit permettant d'exécuter un programme de génération de vecteurs pseudo-aléatoires, la génération de vecteurs pseudo-aléatoires est réalisée en utilisant des LFSR (*Linear Feedback Shift Register*) ou des automates cellulaires (CA : *Cellular Array*). Ces automates ne seront pas détaillés ici.

Sans rentrer dans les détails, un LFSR est un registre à décalage complété par un certain nombre de portes OU Exclusif permettant de réaliser l'entrée de l'information externe et des rebouclages internes. Le nombre et la position des portes OU Exclusif dépendent du polynôme diviseur choisi. Le degré de ce polynôme diviseur correspond au nombre de bascules dans le registre [117].

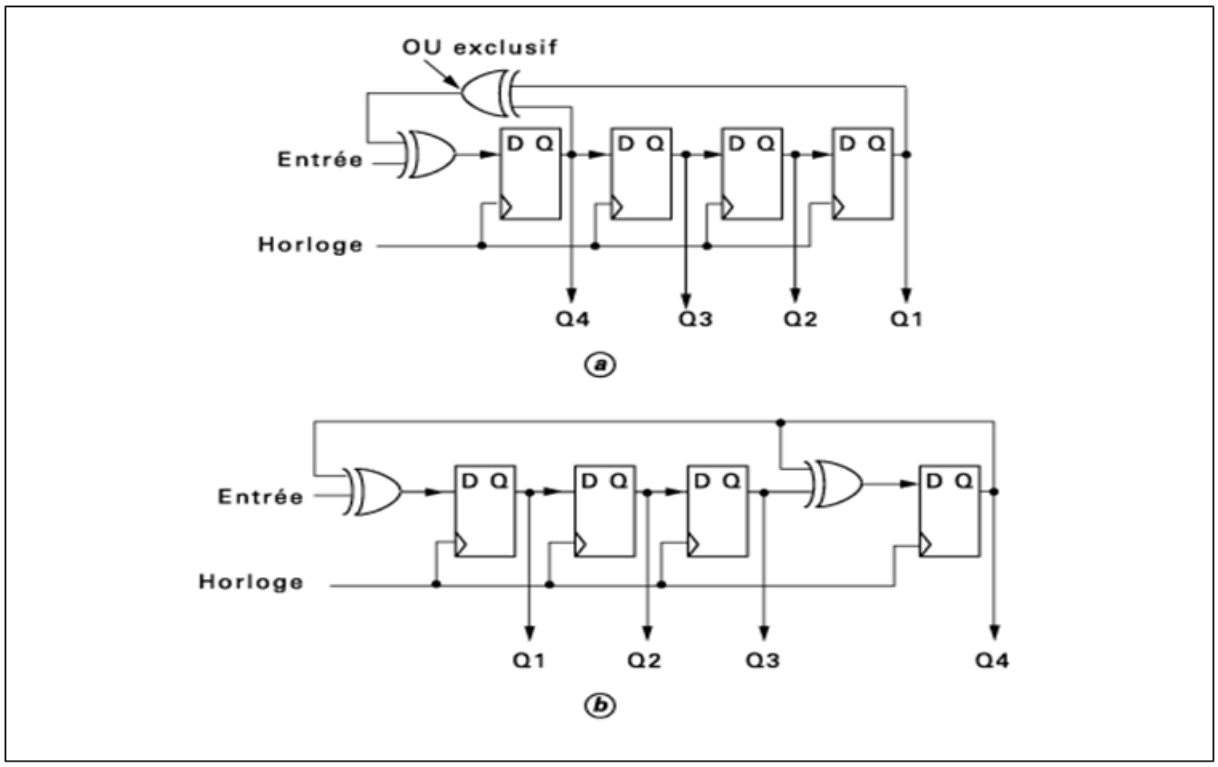


FIGURE 3.3 – Exemples d'implémentation d'un LFSR : (a) structure standard et (b) structure modulaire pour le polynôme diviseur $G(x) = x^4 + x^3 + 1$.

Pour un même polynôme, différentes structures équivalentes peuvent être utilisées, les plus communes étant la structure standard et la structure modulaire (Figure 3.3).

- Dans le cas de la structure standard, tous les rebouclages internes sont ramenés sur l'entrée du LFSR ; les portes OU Exclusif sont donc placées en dehors du registre à décalage, ce qui facilite l'implantation de celui-ci. En revanche, la fréquence de fonctionnement maximum de cette structure est limitée par le chemin de rebouclage externe (qui peut, de plus, correspondre à une interconnexion assez longue si le nombre de bascules est élevé) ;
- La structure modulaire permet d'atteindre des fréquences de fonctionnement plus élevées. En revanche, le registre à décalage doit être interrompu en certains points pour insérer les portes OU Exclusif réalisant les rebouclages internes.

Le LFSR représenté à la figure 3.4 est caractérisé par le polynôme $P(x) = x^4 + x + 1$. La séquence générée par ce circuit est composée d'une succession de sous séquences identiques constituées des 15 (2^4-1) vecteurs. 15 états est la période maximum que l'on puisse atteindre avec un LFSR à quatre étages [105]. L'état (0000) est un état absorbant puisque la sortie de l'OÙ-exclusif reste à zéro. Donc pour être sûr que le générateur fonctionne

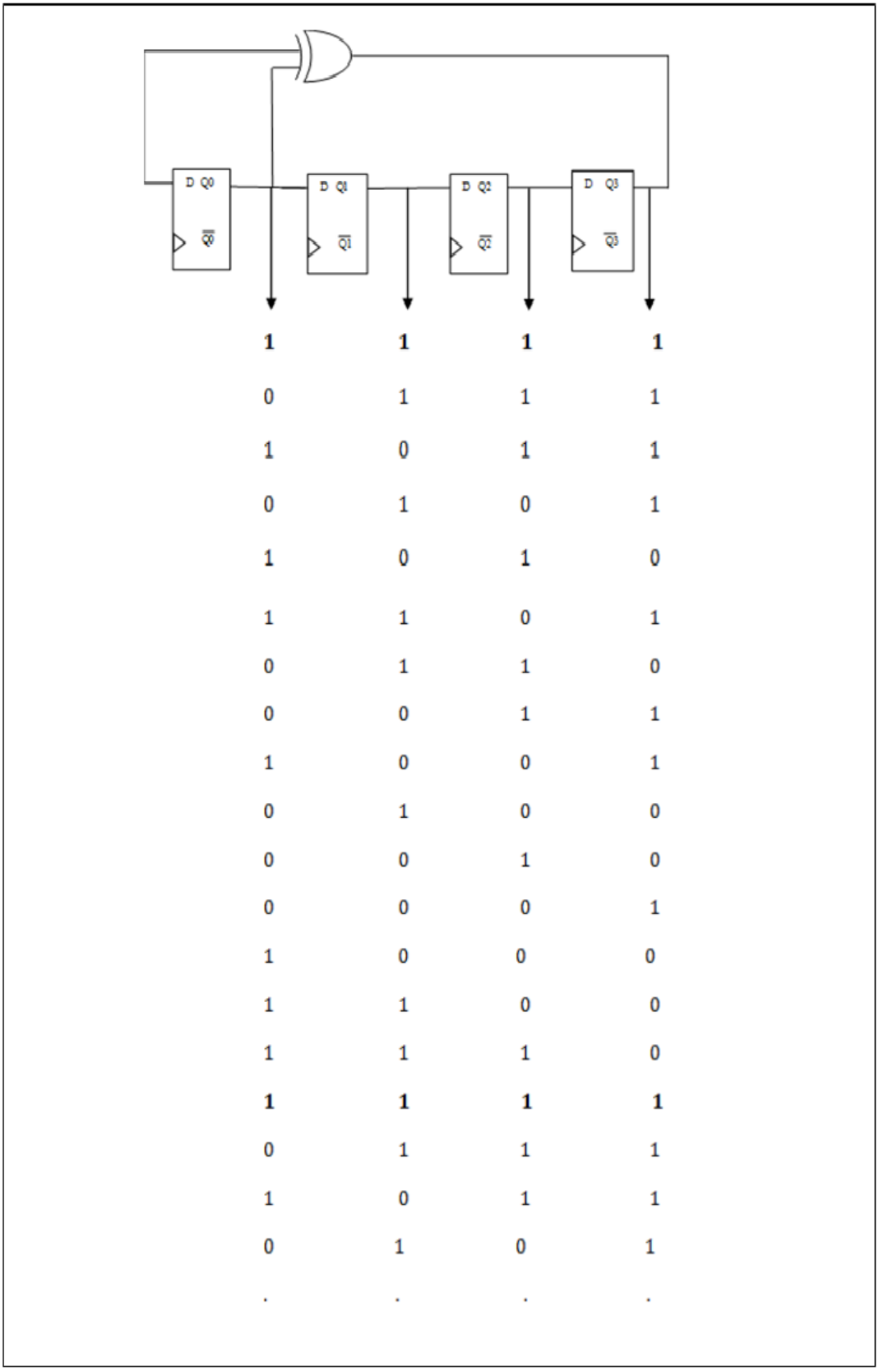


FIGURE 3.4 – Générateur de séquence de test pseudo-aléatoire utilisant un LFSR.

[118], il faut au moins une bascule initialisée à "1".

3.3.1.2 L'analyse de signature

L'analyse de signature utilise le LFSR en tant que composant de base [105,119]. Lorsque le LFSR dispose d'une entrée série, on parle alors de SISR (SISR : *Single Input Shift Registe*r) ou de plusieurs entrées parallèles dans le cas du MISR (MISR : *Multiple Input Shift Register*). Alors que le LFSR utilisé comme TPG est un système fermé (une fois initialisé), le LFSR utilisé pour l'analyse de signature a besoin de données d'entrée, en particulier la réponse de sortie du CUT. La réponse CUT peut être représentée par un polynôme, appelé le polynôme de données $K(x)$.

3.3.1.2.1 Analyse série de signature

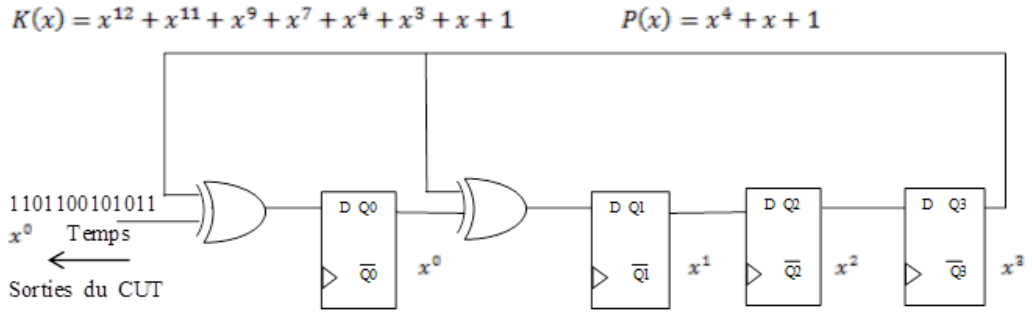
L'idée de base derrière l'analyse de signature consiste à diviser le polynôme de données $K(x)$ par le polynôme caractéristique du LFSR, $P(x)$. Le reste de cette division polynomiale, $R(x)$ est la signature utilisée pour comparer les réponses du CUT avec celles du circuit correct pour pouvoir détecter la présence d'éventuelles fautes dans le CUT à la fin de la séquence BIST. On peut alors écrire [104] :

$$K(x) = Q(x)P(x) + R(x) \tag{3.1}$$

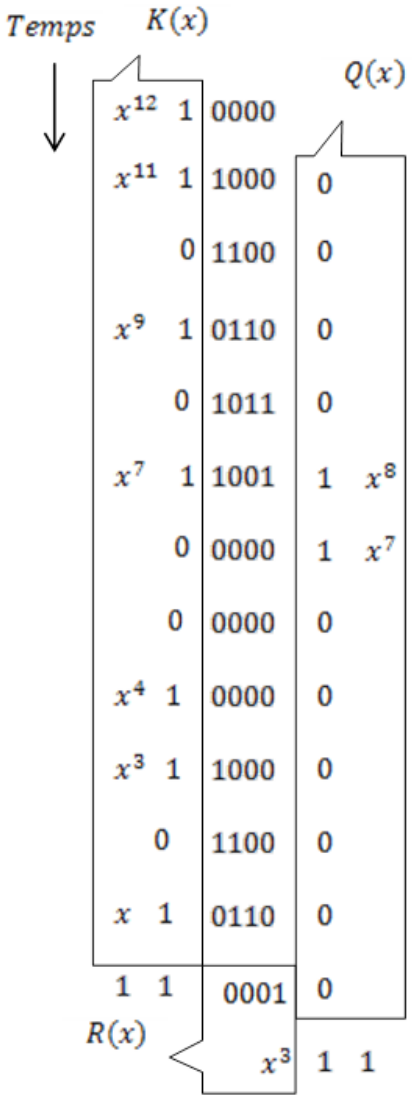
Où :
- $Q(x)$ est le quotient de la division de$K(x)$ par$P(x)$;
- $P(x)$ est le polynôme caractéristique du SISR ;
- $R(x)$ est le reste de la division de $K(x)$ par $P(x)$.

$R(x)$ est aussi l'état final du registre LFSR, configuré comme analyseur de signature SAR (SAR : *Signature Analysis Register*), après avoir entré tous les bits de la séquence d'entrées dans le SAR c.-à-d. à la fin de la séquence BIST.

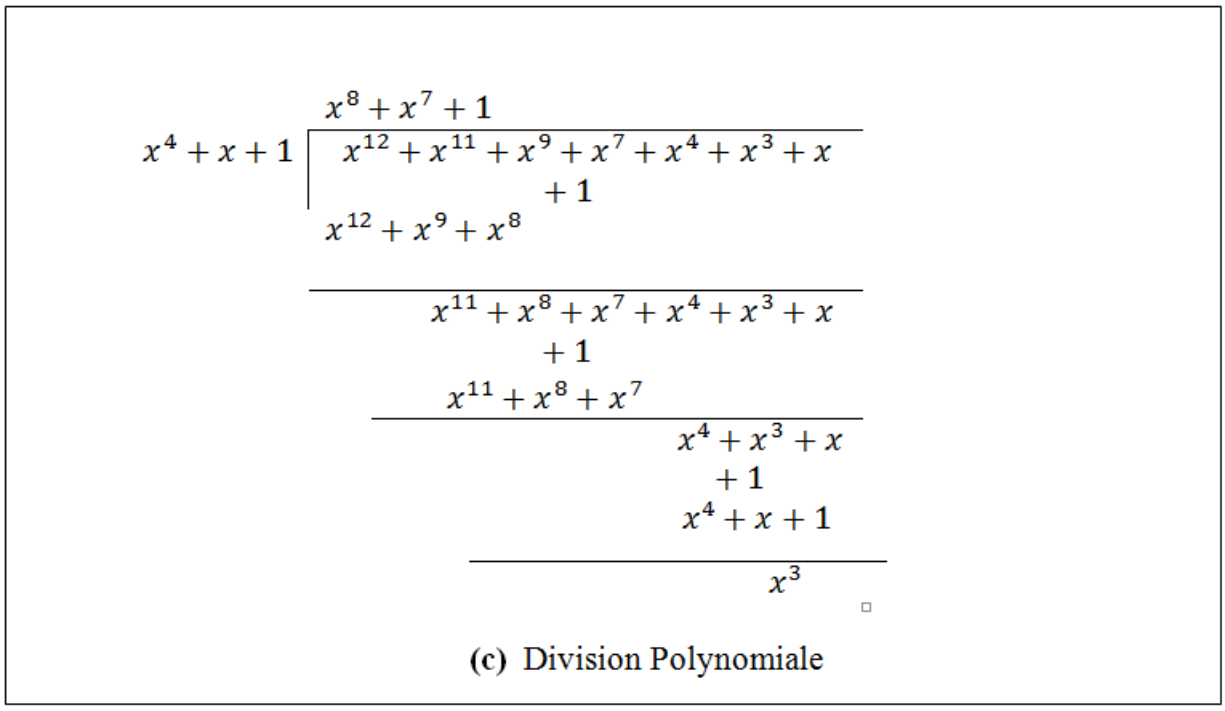
La figure 3.5 donne un exemple d'analyse de signature réalisé par un SISR configuré en SAR. Le polynôme de données pour cet exemple est donné par $K(x)$, comme le montre la figure 3.5a. Le contenu du SAR et les données d'entrée $K(x)$, pour chaque cycle d'horloge, sont illustrés par la figure 3.5b. La signature contenue dans le SAR à la fin de la séquence BIST est indiquée au bas de la figure 3.5b, elle est notée $R(x)$. Le procédé de la division polynomiale est illustré sur la figure 3.5c où la division du polynôme des données de sorties du CUT $K(x)$, par le polynôme caractéristique du LFSR $P(x)$, résulte en un quotient $Q(x)$, et un reste $R(x)$, qui est le contenu du SAR à la fin de la séquence BIST.



(a) SISR avec $P(x) = x^4 + x + 1$



(b) Opération du SISR par cycle d'horloge



(c) Division Polynomiale

FIGURE 3.5 – Exemple d'analyse de signature.

Considérons le SAR de la figure 3.5a, de polynôme caractéristique $P(x) = 1 + x + x^4$ et supposons maintenant que $K = \{10011011\}$, qui peut être exprimé sous forme de polynôme, $K(x) = 1 + x^3 + x^4 + x^6 + x^7$. En utilisant la division polynomiale, on obtient $Q(x) = x^2 + x^3$ et $R(x) = 1 + x^2 + x^3$ d'où $R = \{1011\}$. Le reste $\{1011\}$ est égal à la signature dérivée de la figure 3.5a quand le SAR est initialisé au début à l'état $\{0000\}$ appelé "*Seed* " (Figure 3.6a).

Supposons qu'une faute *F1* produit une séquence de sortie $K' = \{\mathbf{11}\mathbf{0}\mathbf{0}1011\}$, d'où $K'(x) = 1 + x + x^4 + x^6 + x^7$, comme le montre la figure 3.6b. En utilisant la division polynomiale, on obtient $Q'(x) = x^2 + x^3$ et $R'(x) = 1 + x + x^2$ d'où $R' = \{1110\}$. On constate que la signature obtenue $R^{'} = \{1110\}$, est différente de celle du circuit sein $R = \{1011\}$ et donc la faute *F1* est détectée. Cependant, pour une faute *F2* qui produit une séquence de sortie $K'' = \{\mathbf{11}\mathbf{0}\mathbf{0}11\mathbf{0}1\}$, d'où $K''(x) = 1 + x + x^4 + x^5 + x^7$, comme sur la figure 3.6c, on obtient $Q''(x) = x + x^3$ et $R''(x) = 1 + x^2 + x^3$ d'où $R'' = \{1011\}$. On constate que $R'' = R$ est la faute *F2* est non détectée [104].

La détection de fautes ou le problème d'*aliasing* ou encore de masquage d'un SAR peut être mieux mise en évidence en examinant la séquence d'erreur ou le polynôme erreur $E(x)$ de la séquence correcte K et la séquence erronée K'. Définissons $E = K + K'$ d'où $E(x) = K(x) + K'(x)$.

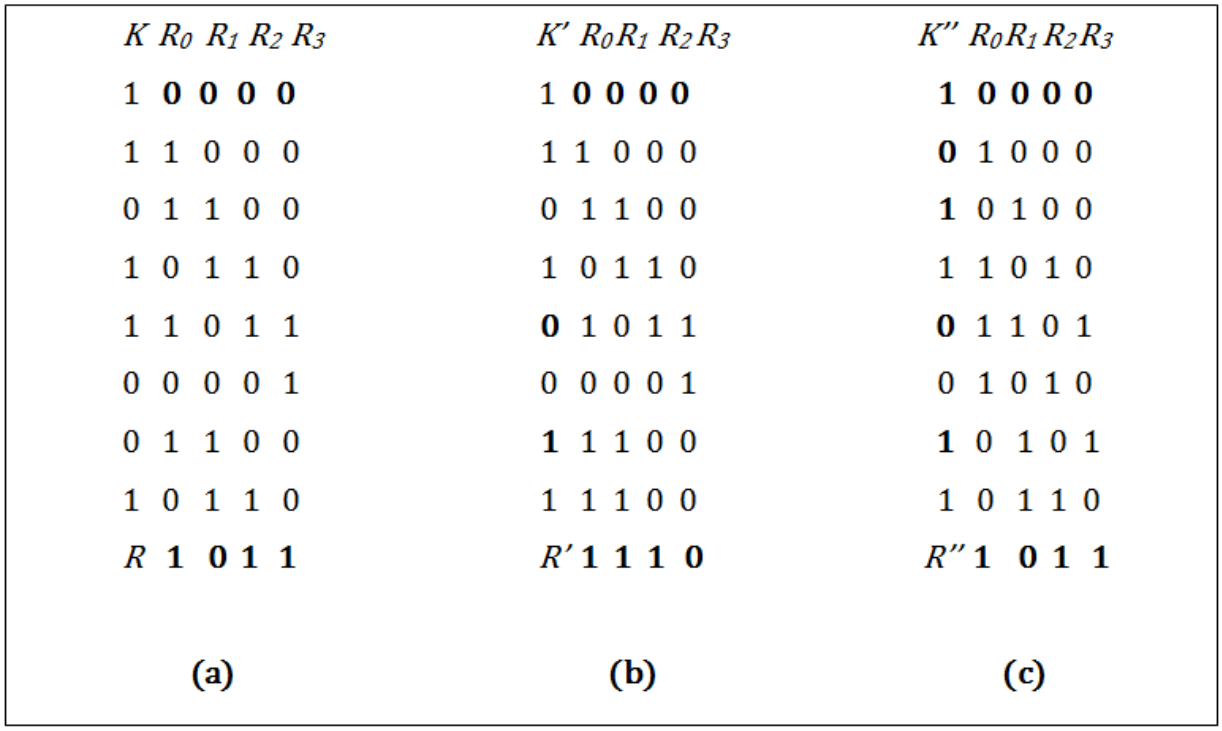


FIGURE 3.6 – Exemples d'analyse de signature pour le circuit de la figure 3.6a (a) signature du circuit sans faute ; (b) signature pour la faute *F1* et (c) signature pour la faute *F2*.

Si $E(x)$ n'est pas divisible par $P(x)$, alors toutes les fautes génerées par la séquence K' seront détectées ; autrement les fautes ne seront pas détectées [104] . Considérons la faute *F1* à nouveau. On obtient $E = \{01010000\} = K + K' = \{10011011\} + \{11001011\}$ d'où $E(x) = x + x^3$. Puisque $E(x)$ n'est pas divisible par $P(x) = 1 + x + x^4$, la faute *F1* est détectée. Considérons maintenant la faute *F2*. On a $E = \{01010110\} = K + K'' = \{10011011\} + \{11001101\}$ d'où $E(x) = x + x^3 + x^5 + x^6$. Puisque $P(x) = 1 + x + x^4$ divise $E(x)$ c.-à-d. $E(x) = (x + x^2)P(x)$, la faute *F2* n'est pas détectée [30,104].

Supposons que le SISR (*Single Input Shift Register*) se compose de N étages. Pour une séquence de $L - bit > N$, il y a $2^{(L-N)}$ possibilités de produire une signature de $N - bit$ dont l'une est correcte. Parce qu'il y a au total $2^L - 1$ séquences erronées dans la séquence de $L - bit$, la probabilité du masquage (*Aliasing*) en utilisant un SISR à N étages, pour l'analyse sérielle de signature SSA (SSA : *Serial Signature Analysis*), est donnée par l'équation (3.2) :

$$P_{SSA}(N) = 2^{(L-N)}/(2^L - 1) \tag{3.2}$$

Si $L \gg N$, alors $P_{SSA}(N) \approx 2^N$. Pour $N = 20$, $P_{SSA}(N) < 2^{-20} = 0.0001\%$.

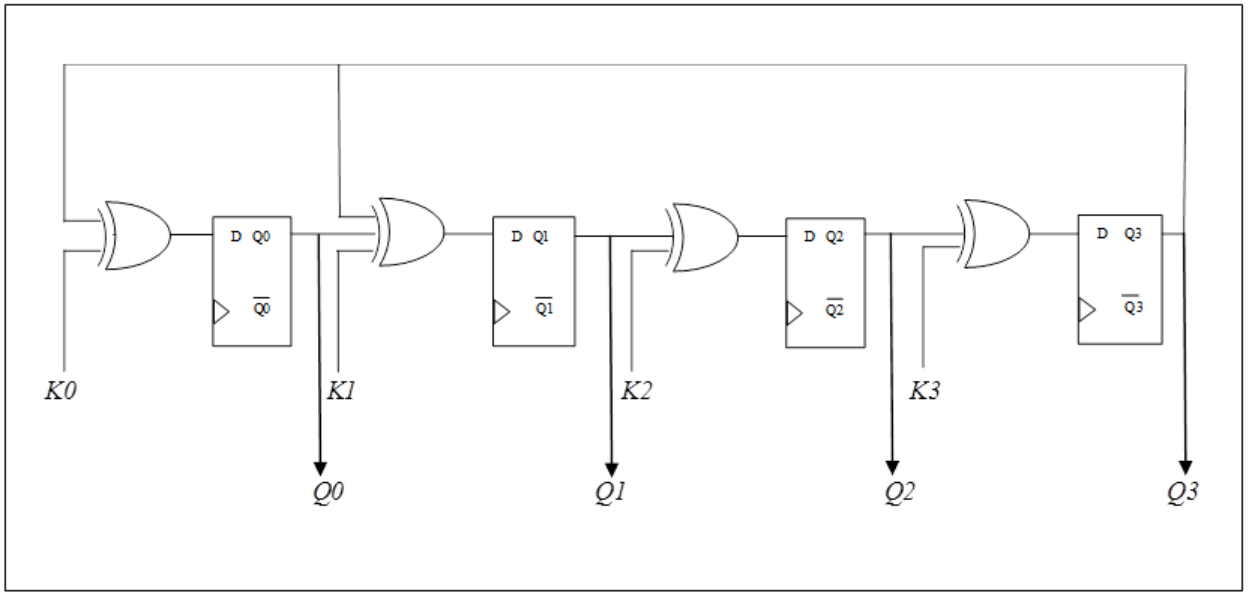


FIGURE 3.7 – Implémentation d'un MISR avec $P(x) = 1 + x + x^4$.

3.3.1.2.2 Analyse parallèle de signature

Dans le cas très courant où le circuit à tester présente plusieurs sorties, deux solutions sont envisageables :

- Multiplexer les sorties pour attaquer l'entrée unique du LFSR, cette solution est lente ;
- Utiliser un analyseur de signature parallèle MISR (MISR : *Multiple-Input Signature Analysis*).

Un MISR est implanté simplement en rajoutant à la structure du LFSR un ensemble de portes OU Exclusif permettant d'entrer simultanément un bit d'information externe au niveau de plusieurs bascules du registre. Comme le montre la figure 3.7, le plus souvent, le nombre de bits du registre correspond au nombre de bits d'information à lire en parallèle.

Dans [120], il a été montré qu'un MISR à N entrées peut-être remodelé comme un SISR avec une *séquence d'entrée efficace* $K(x)$ et une s*équence erreur efficace* $E(x)$ exprimées comme suit :

$$K(x) = K_0(x) + xK_1(x) + x^2K_2(x) + ... + x^{N-2}K_{N-2}(x) + x^{N-1}K_{N-1}(x) \qquad (3.3)$$

Et

$$E(x) = E_0(x) + xE_1(x) + x^2E_2(x) + ... + x^{N-2}E_{N-2}(x) + x^{N-1}E_{N-1}(x) \qquad (3.4)$$

K_0	1 0 0 1 0
K_1	0 1 0 1 0
K_2	1 1 0 0 0
K_3	1 0 0 1 1
K	1 0 0 1 1 0 1 1

FIGURE 3.8 – Séquence K équivalente des 4 séquences d'entrées $K0, K1, K2$ et $K3$.

Considérons le MISR à 4 entrées de la figure 3.7 avec$P(x) = 1 + x + x^4$. Supposons que $K_0 = \{10010\}$, $K_1 = \{01010\}$, $K_2 = \{11000\}$, et $K_3 = \{10011\}$. A partir de ces informations, la signature R du MISR peut être calculée comme $\{1011\}$. En utilisant $K(x) = K_0(x) + xK_1(x) + x^2K_2(x) + x^3K_3(x)$, on obtient $K(x) = 1 + x^3 + x^4 + x^6 + x^7$ d'où $K = \{10011011\}$ (Figure 3.8). Celle-ci est bien la séquence que nous avons utilisée dans l'exemple du SISR précédent et par conséquent, $R = \{1011\}$.

Supposons qu'il y'ait ML-bit séquences à compacter par un MISR à N-étages où $L > N \geq M \geq 2$. La probabilité de masquage pour un PSA (PSA : *Parallel Signature Analysis*) devient alors :

$$P_{PSA}(N) = 2^{(ML-N)}/(2^{ML} - 1) \tag{3.5}$$

Si $L \gg N$, alors $P_{PSA}(N) \approx 2^N$. Pour $N = 20$, $P_{PSA}(N) < 2^{-20} = 0.0001\%$.

À partir de ce résultat, on peut suggérer que la $P_{PSA}(N)$ depend principalement de N, quand $L \gg N$. Par conséquent, l'augmentation du nombre d'étages du MISR ou l'utilisation du même MISR avec des polynômes caractéristiques $P(x)$ différents peuvent réduire substantiellement la probabilité du masquage [120,121].

3.3.2 Le Registre BILBO Optique de base

On peut constater qu'il y a une analogie entre le générateur de vecteur pseudo-aléatoire à LFSR et un analyseur de signature également réalisé par un LFSR. C'est pourquoi on a eu l'idée de fusionner, en un seul ensemble, le générateur de séquences et l'analyseur de signature selon le schéma appelé BILBO (*Built-In Logic Block Observation*) [122].

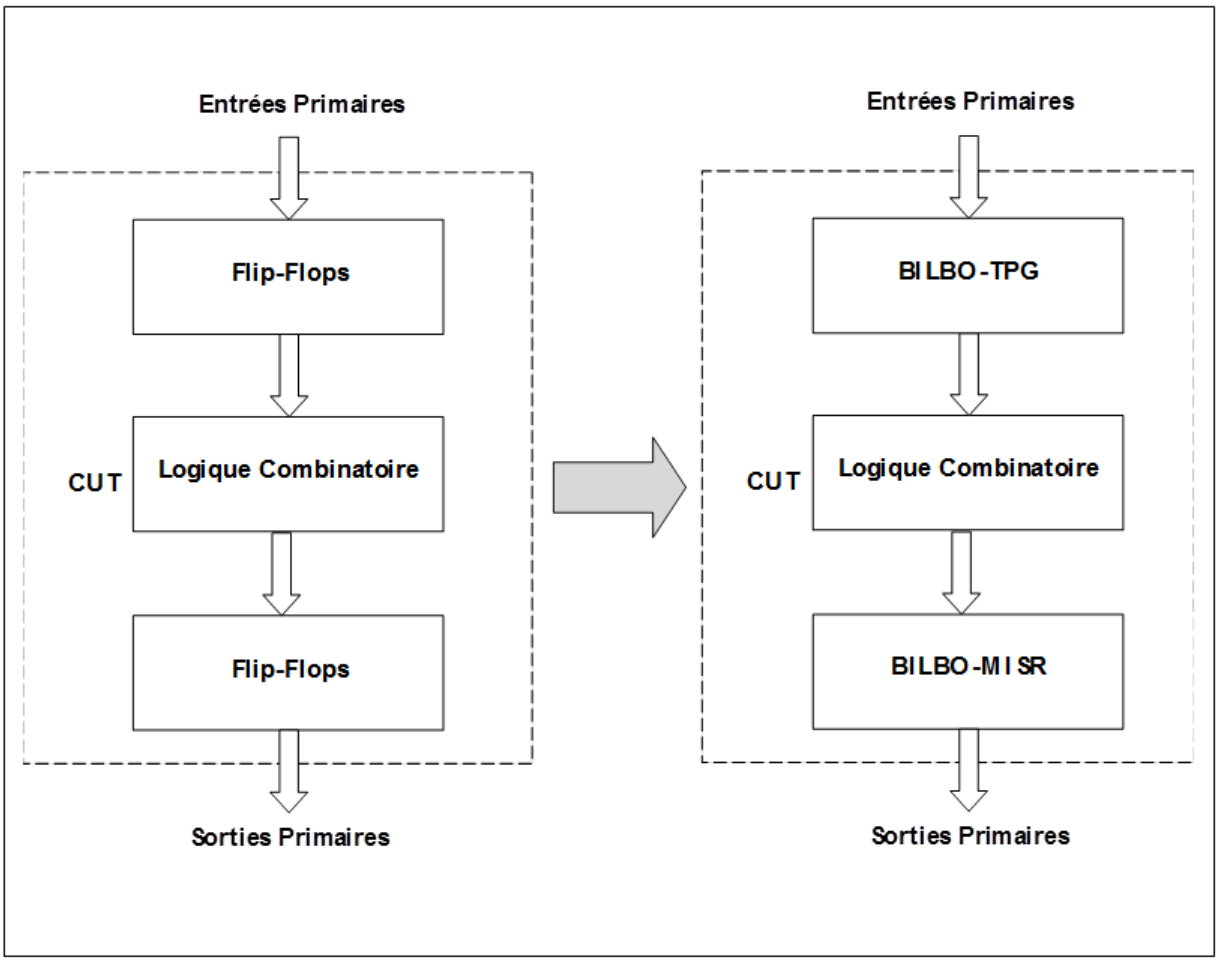


FIGURE 3.9 – Application de l'approche BILBO à un CUT.

Les structures de type BILBO sont largement utilisées dans le BIST des circuits numériques. Ces structures, basées sur l'utilisation de LFSR, permettent la génération de vecteurs de test binaires pseudo-aléatoires. Ces stimuli sont appliqués sur un CUT. La réponse de ce dernier à ces stimuli est alors compactée dans une signature de test. Cette opération est normalement réalisée au moyen de LFSR à entrées multiples appelés MISR. Les structures BILBO peuvent opérer dans les deux modes, génération et analyse de la signature [123]. Le BILBO est une architecture BIST intégrée puisqu'il utilise les bascules du CUT pour construire les fonctions TPG et l'analyseur de la réponse ORA (ORA : *Output Response Analyser*) [124].

La figure 3.9 donne un exemple d'application dans sa forme la plus simple des structures BILBO à un CUT [105], où les Flip-Flops près des entrées/sorties primaires sont utilisés pour la construction de deux cellules BILBO. Le BILBO près des entrées primaires est utilisé pour générer des séquences pseudo-aléatoires et joue alors le rôle d'un TPG, tandis

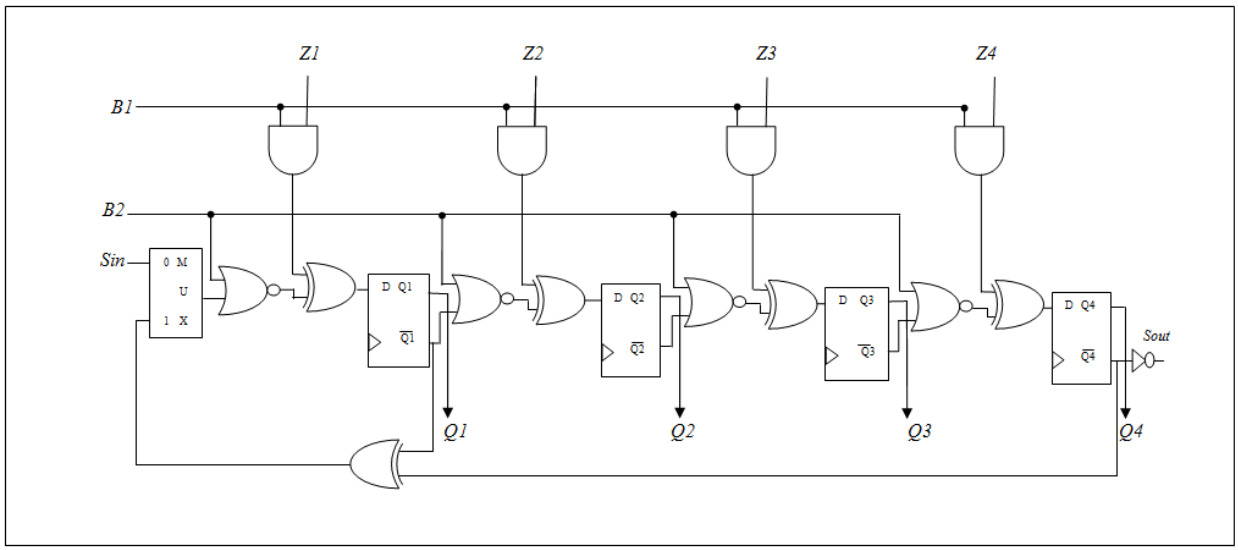


FIGURE 3.10 – Implémentation originale du BILBO [122].

que le BILBO près des sorties primaires du CUT, il fonctionne comme un analyseur de signature MISR. En conséquence, le BILBO est une approche BIST parallèle ("*Test-per-clock*" en anglais) puisque à chaque top d'horloge de la séquence BIST, un nouveau vecteur de test est appliqué au CUT et une nouvelle réponse du celui-ci est compactée dans le MISR.

Les portes logiques supplémentaires ajoutées aux Flip-Flops existants, pour créer le BILBO, sont illustrées dans figure 3.10 pour la mise en œuvre d'un BILBO 4-bit. Pour chaque bascule, une porte OU-exclusif, une porte ET, et une porte NON-OU sont ajoutées [122]. Pour chaque BILBO, un multiplexeur, un inverseur et une ou plusieurs portes OU-exclusive (nécessaire pour construire un polynôme primitif caractéristique) sont ajoutés.

Toutefois, il faut rappeler que, même si cette architecture offre un test parallèle (*Test-per-clock*), elle n'est cependant utilisée qu'en mode *Off-line* [30,41,105]. C'est pourquoi on a eu l'idée d'apporter des modifications à cette architecture pour pouvoir l'exploiter à des fins de test en mode en ligne [115]. Jusqu'à présent, ce mode de test n'a pas encore eu de solutions satisfaisantes, surtout quand il s'agit de détection de fautes transitoires, qui apparaissent durant le fonctionnement normal des systèmes, et ne peuvent donc être détectées en mode *Off-line*. De ce fait, on se propose d'introduire l'aspect optique en insérant des sondes optiques appropriées à des emplacements bien précis dans l'architecture des cellules BILBO, pouvant extraire de l'information sur l'état logique des broches de sortie de ces cellules, qui sont elles-mêmes les entrées/sorties du CUT (Figure 3.9). Ces informations sont transmises sous formes de rayons optiques à un système distant, en temps réel, pour vérifier les résultats produits par ces systèmes en cours de fonctionnement. Ceci

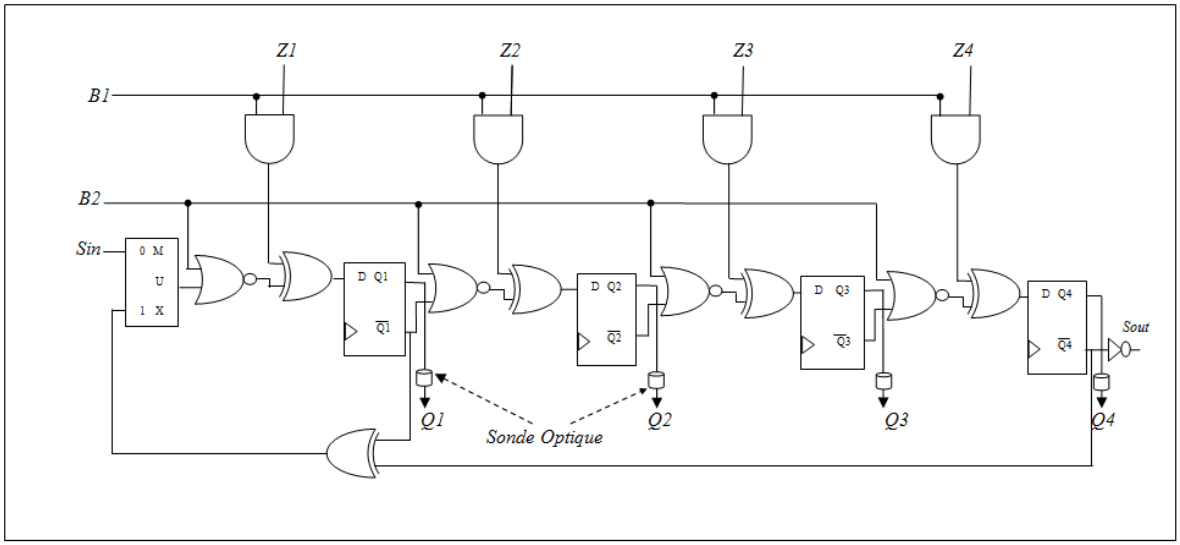


FIGURE 3.11 – Diagramme logique de l'OBILBO proposé [115].

est possible en exploitant le mode normal (*Parallel Load mode*) des cellules BILBO qui, dans ce mode, se comportent comme un registre parallèle.

La figure 3.11 montre le nouveau registre BILBO Optique 4-bit proposé (OBILBO : *Optical BILBO*). Outre le registre à décalage et l'ensemble des portes logiques supplémentaires ajoutées pour construire un registre BILBO standard, notre registre intègre dans son architecture des Sondes Optiques (SO) dont la structure est détaillée dans ce qui suit.

3.3.2.1 Technologie de la sonde optique utilisée

La structure de la sonde optique proposée dans [20] est montrée sur la figure 3.12. Elle est fabriquée sur un substrat Si sur lequel une électrode Au est déposée. Cette dernière sert de miroir optique et d'électrode inférieure, sur laquelle, un matériau optoélectronique (PLT) est déposé, puis une succession de Si et SiO_2 qui agit comme un réflecteur. Il est à noter que, la couche interne du Si sert aussi comme électrode supérieure. La tension appliquée entre la couche Au et Si crée ainsi un champ électrique nécessaire pour changer l'indice de réfraction du matériau optoélectronique PLT. Les dimensions d'une telle sonde [20] sont de l'ordre de 30×30 µm^2.

La propriété du matériau optoélectronique (PLT) employé dans cette sonde peut être considérée, dans ce cas, comme une capacité soumise à un champ électrique pour changer son indice de réfraction optique [20]. Ce dernier obéit à la relation suivante :

$$n = n_0 - \frac{1}{2} n_0^3 r_{13} E_z \tag{3.6}$$

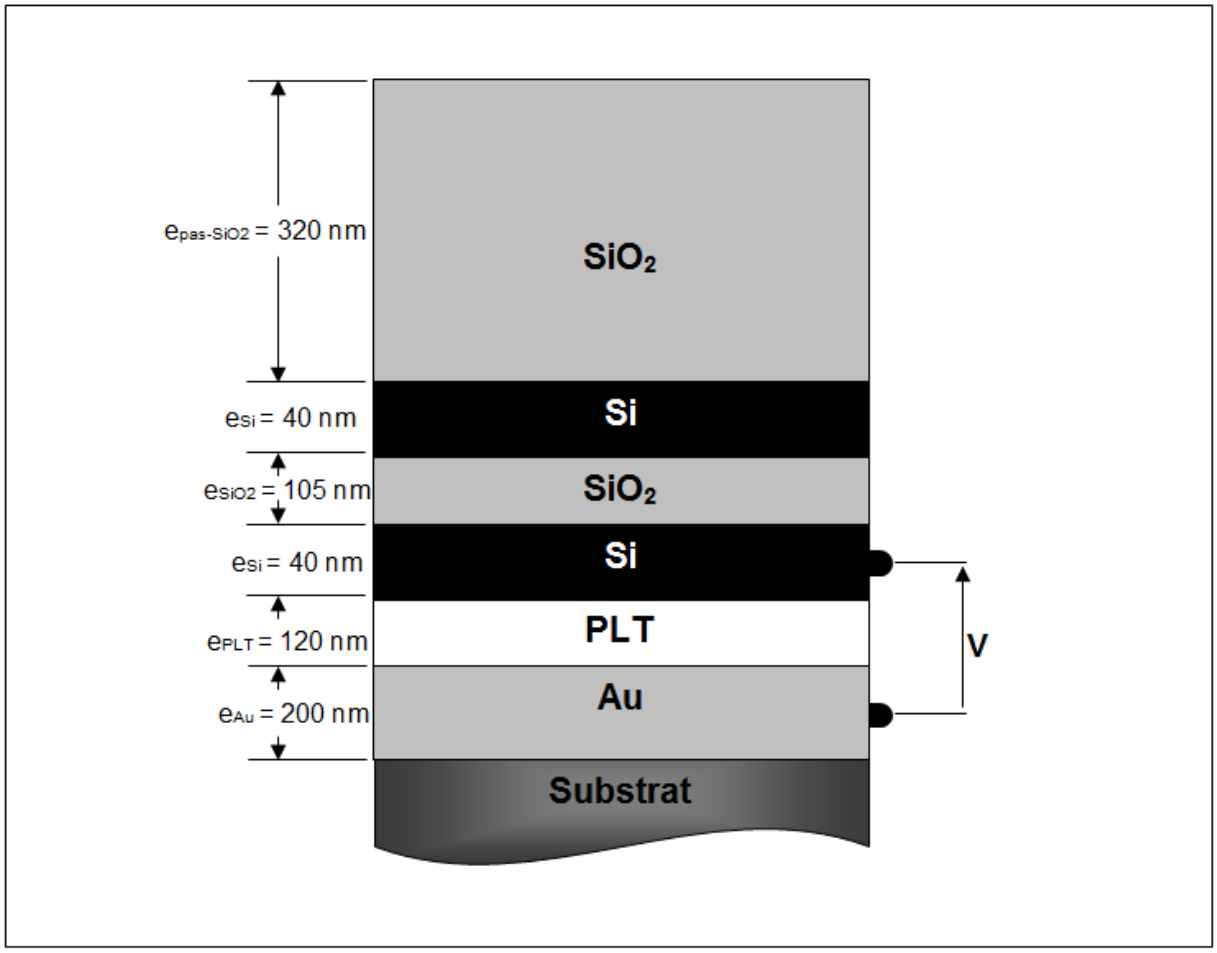


FIGURE 3.12 – Structure de la Sonde Optique [20].

Où :

- n_0 : est l'indice de réfraction en absence du champ électrique ;
- r_{13} : est un coefficient optoélectronique du matériau ;
- E_z : est l'amplitude du champ électrique (Figure 3.13) dans la direction Z.

La composante E_z est régie par la relation (3.7) suivante :

$$E_z = \frac{V}{e} \tag{3.7}$$

Si un rayon optique traverse le matériau PLT, en absence ou en présence du champ électrique correspondant à la valeur $0\,V$ ou $5\,V$ de la tension V (correspondant à la valeur logique 0, 1 respectivement), la phase de sortie ne sera pas la même comme le montre la relation (3.8) :

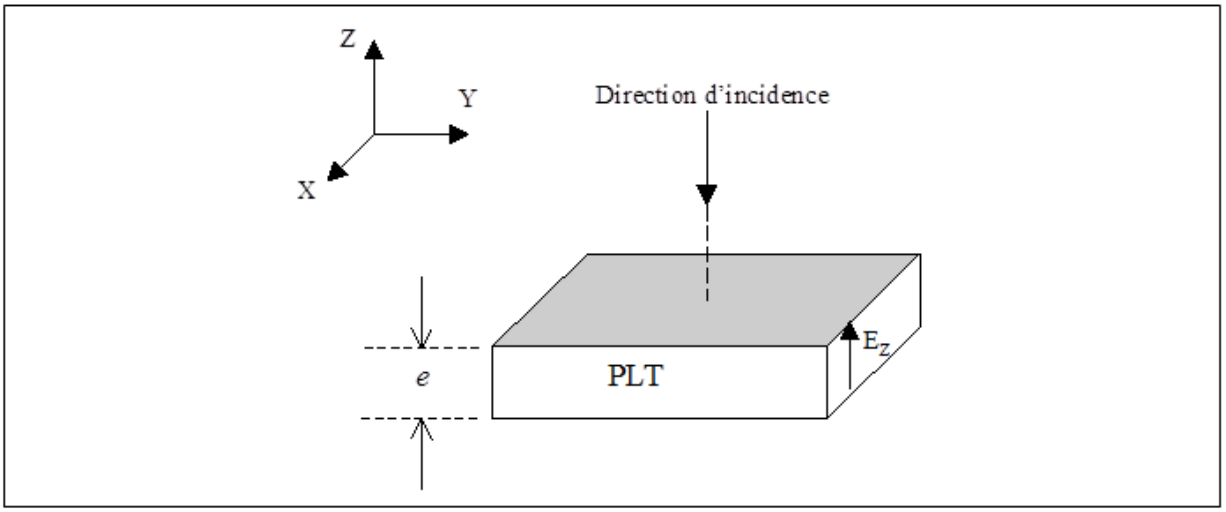


FIGURE 3.13 – Matériau EO de la Sonde Optique [20].

$$\phi = \frac{2\pi n}{\lambda_0} e \tag{3.8}$$

Avec :

- n : est l'indice de réfraction donné par l'équation (3.6) ;
- λ_0 : est longueur du rayon optique dans l'espace ;
- e : est l'épaisseur du matériau PLT.

Le changement de la phase peut être mesuré dans une configuration d'interférence [18,20]. Si une partie de signal optique est partiellement réfléchie sur la première interface du matériau PLT et, une partie sur une seconde interface (Figure 3.14 (a)), l'intensité globale réfléchie (I) du rayon optique est donnée par la relation (3.9).

$$I = I_1 + I_2 + 2\sqrt{I_1 I_2}\cos(2\phi) \tag{3.9}$$

I_1 et I_2 sont respectivement les intensités des rayons optiques réfléchis sur la première et la deuxième surface du matériau PLT. Cette intensité peut être directement convertie en signal électrique par une simple photodiode, par exemple, qui détermine l'absence ou la présence du champ électrique. Il est à noter, qu'il est possible d'augmenter le niveau de l'intensité réfléchie (I) dans une structure résonante telle que l'interféromètre fibre optique où des réflexions multiples peuvent apparaître (Figure 3.14 (b)). L'intensité globale, dans ce cas, sera donnée par la relation (3.10).

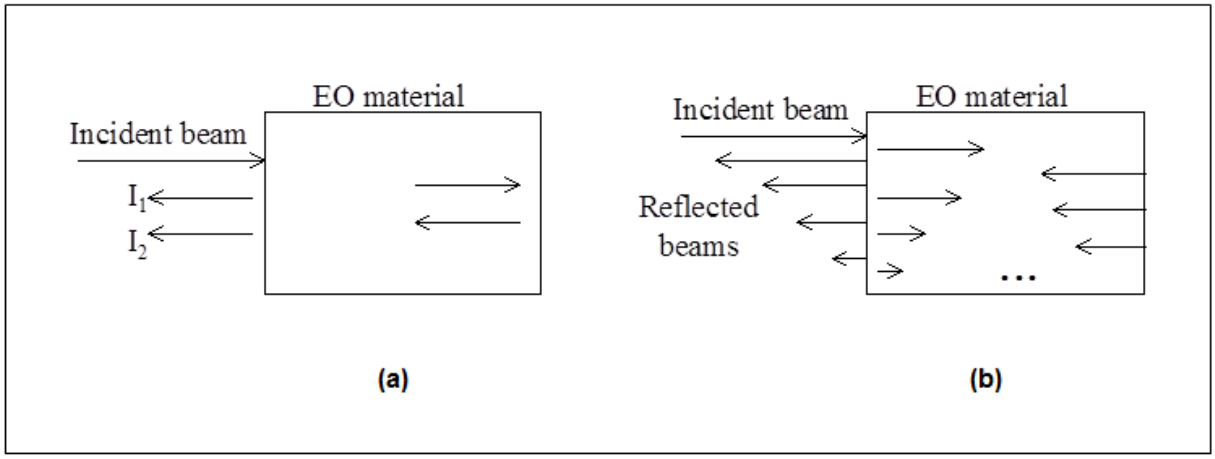


FIGURE 3.14 – Réflexion du rayon optique : **(a)** Simple, **(b)** Multiple [20].

$$I = I_{Max}\left(1 - \frac{1}{1 + M\sin 2(\frac{\phi}{2})}\right) \tag{3.10}$$

I_{Max} étant l'intensité maximale réfléchie et M est un coefficient qui est fonction de la réflectivité des deux miroirs.

3.3.2.2 Mode de fonctionnement du registre OBILBO

Comme pour le registre BILBO standard [122], l'OBILBO présentant quatre modes de fonctionnement selon les valeurs données aux commandes B1 et B2 : un fonctionnement parallèle correspondant au fonctionnement normal du système, un mode LFSR (Test) qui peut servir soit de générateur de vecteurs de test obtenu par machine d'états autonome (les états étant les vecteurs de test), soit de registre d'évaluation du résultat par compression de la réponse du circuit et enfin un mode série permettant d'inclure le registre dans une chaîne de scan.

3.3.2.2.1 Mode normal (Parallel Load mode)

Lorsque $B1 = B2 = 1$, le circuit fonctionne comme un registre parallèle dont les entrées sont les Z_i et les sorties les Q_i (Figure 3.15). Dans ce cas, et si le registre OBILBO est placé à la sortie d'un CUT (Figure 3.9), fonctionnant en mode normal, les entrées de l'OBILBO sont aussi les sorties du CUT et donc ses réponses fonctionnelles sont transmises aux sorties Q_i de l'OBILBO sans pour autant ralentir ou interrompre le fonctionnement du CUT. De ce fait, les sondes optiques placées aux sorties Q_i capturent ces données et les transmettent sous forme de signaux optiques correspondant aux états logiques des sorties Q_i voire les

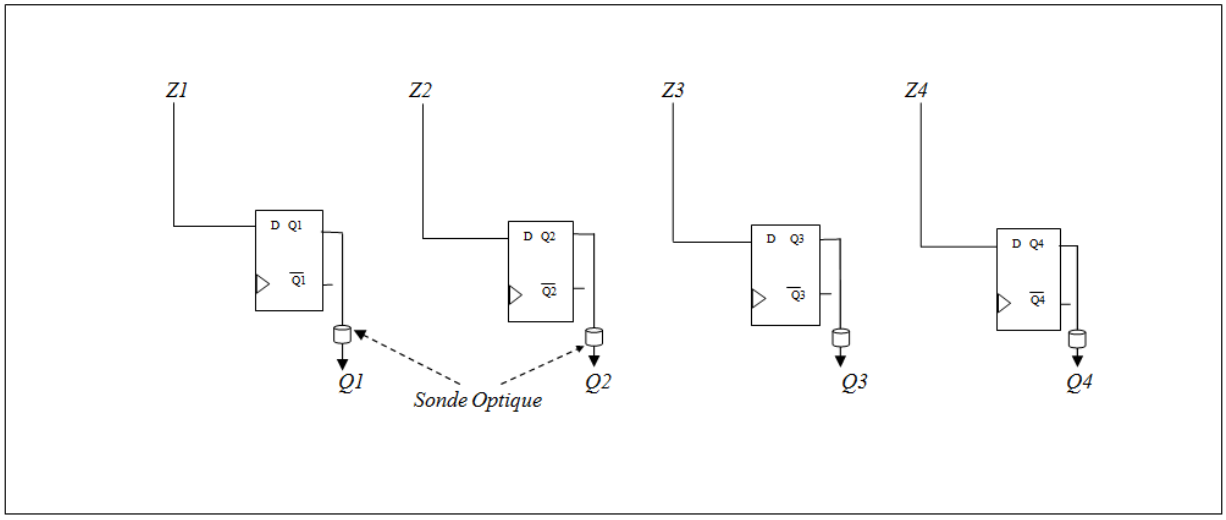


FIGURE 3.15 – Mode de fonctionnement normal de l'OBILBO.

sorties du CUT. Par conséquent, une vérification à distance de ces réponses peut être réalisée en temps réel en comparant ces dernières avec celles d'un module dupliqué.

3.3.2.2.2 Mode Scan

Lorsque $B1 = B2 = 0$, les bascules sont montées en série et le circuit fonctionne comme un registre à décalage série avec l'entrée S_{in} et la sortie S_{out} permettant la circulation des données de test comme le montre la figure 3.16. Cependant, avec les nouvelles sondes optiques intégrées dans sa structure, le registre est désormais transformé en un registre parallèle, ce qui simplifie énormément son diagnostic. En effet, pour le test des fautes de collage à 1, par exemple, en choisissant le *Seed* égal à "0000...", l'opération de scan se réduit à une seule opération de décalage alors qu'il fallait normalement N tops d'horloge (N étant le nombre de bascules formant le registre BILBO) pour accomplir la même tâche avec un BILBO standard. Pour le test des fautes de collage à 0, il suffit de choisir le *Seed* égal à "1111...", de réaliser une seule opération de décalage et de comparer les signaux émis par les sondes optiques, correspondant aux états logiques des sorties, après bien sûr une conversion A/N de ces signaux. Ainsi, le caractère série de ce registre est désormais évité.

3.3.2.2.3 Mode LFSR (Test)

Si $B1 = 1$ et $B2 = 0$, le registre OBILBO est monté en diviseur polynomial (Figure 3.17). Il peut alors servir soit de registre d'analyse de signature parallèle MISR et les réponses à compacter sont injectées sur les entrées Z_i , soit de générateur de séquences pseudo-aléatoires parallèle et les entrées Z_i sont maintenues à 0. Dans les deux cas, les

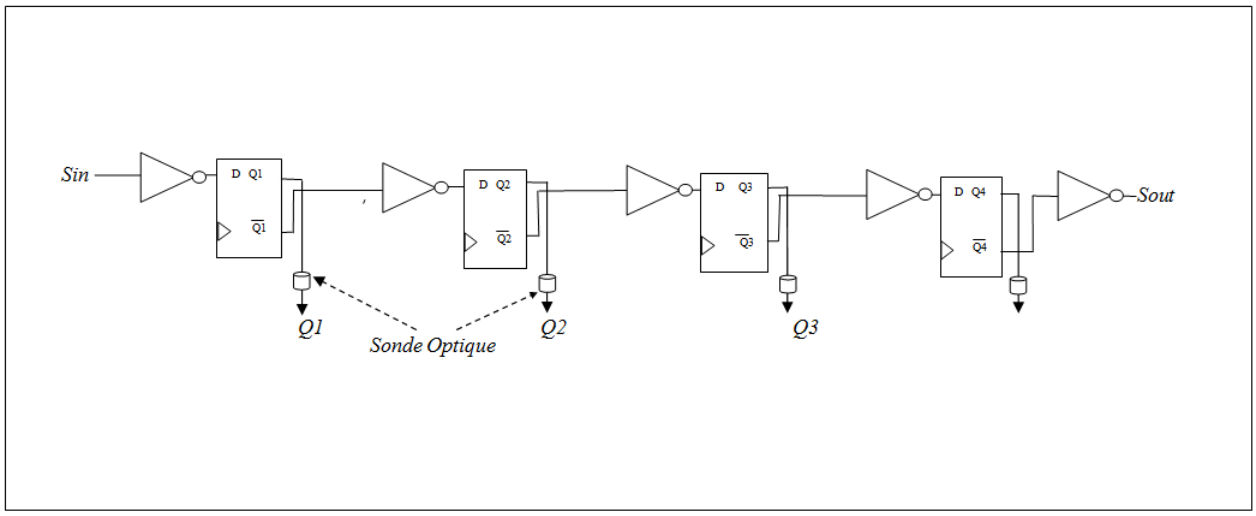


FIGURE 3.16 – Mode Scan de l'OBILBO.

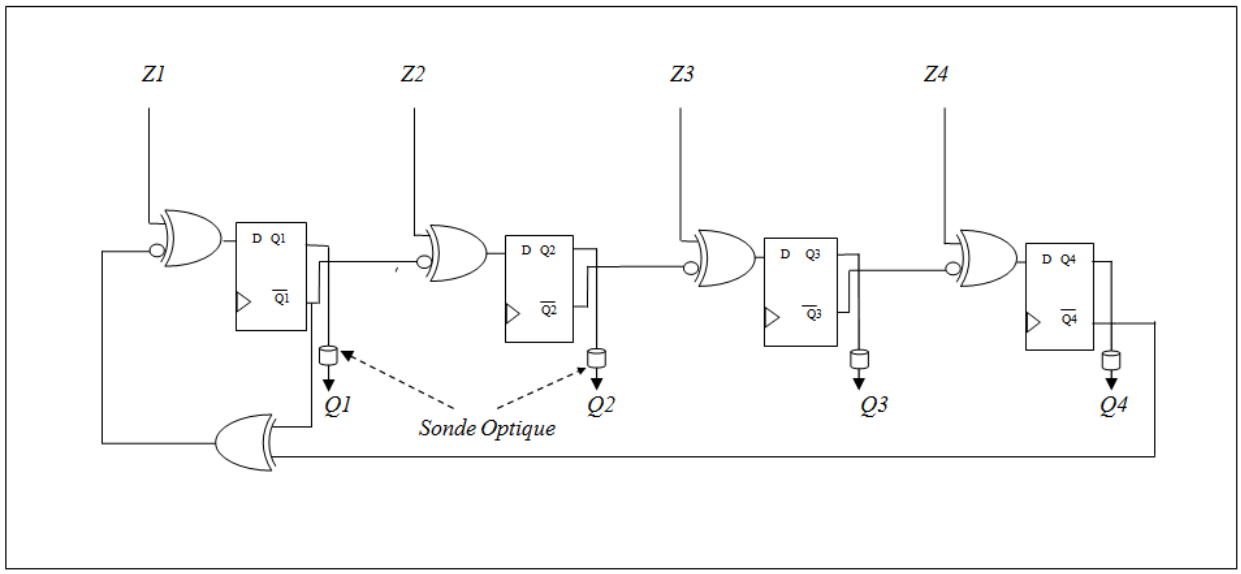


FIGURE 3.17 – Mode LFSR de l'OBILBO.

sondes optiques associées aux sorties Q_i transmettent des signaux optiques correspondant aux états logiques des sorties Q_i. Lors du mode MISR, le contenu du registre OBILBO est transmis sous forme de signature optique à un système distant pour vérification. Ceci est un autre avantage de l'intégration de ces sondes optiques dans le BILBO.

3.3.2.2.4 Mode REST

Le circuit est remis à zéro en positionnant $B1 = 0$ et $B2 = 1$. Les signaux optiques émis par les sondes optiques considérées permettent de vérifier à distance si le registre est bel et bien remis à zéro ou non.

3.3.3 Le Capteur optoélectronique

Pour pouvoir analyser et comparer les signaux optiques issus des OBILBO, des capteurs optoélectroniques sont nécessaires. Dans notre cas, de simples photodiodes sont utilisées.

3.3.4 Le convertisseur A/D

Les signaux analogiques délivrés par les photodiodes sont convertis en signaux numériques et sont appliqués aux entrées du contrôleur.

3.3.5 Le Contrôleur

Le contrôleur génère un signal *Erreur* de niveau logique haut dans le cas où les signaux issus des deux applications ne sont pas identiques et de niveau bas autrement.

3.4 Stratégie de test

La procédure de test exploite le mode normal qu'offre le standard BILBO. Dans ce mode, les données disponibles sur les broches de sorties de chaque module du CUT (Figure 3.9) des deux applications sont échantillonnées concurremment à chaque cycle d'horloge, grâce aux sondes optiques considérées dans l'architecture des registres OBILBO. Cette opération n'affecte pas le mode de fonctionnement normal du CUT puisque aucune opération de génération de vecteurs de test n'est sollicitée. Autrement dit, les entrées usuelles du CUT sont elles-mêmes les vecteurs de test. En conséquence, il y a possibilité de détection de fautes avec une latence de faute de un cycle d'horloge, grâce à la comparaison continuelle des signaux optiques émis par les OBILBO. Ceci est vrai si les registres OBILBO restent

exempts de fautes ("*Fault-free*"). Cependant, si après une comparaison, le contrôleur signale la présence d'erreur, qui peut être due soit à l'application (CUT) soit au registre OBILBO lui-même, alors dans ce cas l'intégrité des registres OBILBO doit être examinée. L'organigramme de la figure 3.18 illustre la stratégie de test définie.

Après configuration des registres OBILBO en mode normal, l'application et sa duplication peuvent opérer en mode de fonctionnement normal. Une lecture continuelle des OBILBO est réalisée en temps réel. Si le comparateur signale la présence de fautes, une opération Scan des registres OBILBO est alors recommandée et se résume ainsi :

1. Configuration des registres OBILBO en mode Scan ;
2. Choix d'un *Seed* égal à la séquence "111111..." pour tester les fautes de collage à 0 ;
3. Réalisation d'une opération de décalage d'une *seule* position ;
4. Lecture des contenus des OBILBO ;
5. Comparaison des signaux issus des convertisseurs A/D ;
6. Si le contrôleur délivre un signal d'état logique '0' (pas de fautes de collage à 0), alors refaire les étapes 2, 3, 4 et 5 pour tester les fautes de collage à 1 et ce par le choix du *Seed* égal à la séquence "00000..." ;
7. En cas où l'étape 6 est terminée sans détection de fautes dans les registres OBILBO, alors pour identifier le module défaillant on effectue des opérations Scan, de chaque application, en examinant la signature optique de chaque module par procédé de comparaison à une signature de référence ("*Fault-Free Signature*").

3.5 Conclusion

Dans ce chapitre, nous avons présenté notre approche de test en ligne des systèmes embarqués. Suivant cette approche, une architecture matérielle a été proposée afin d'améliorer la testabilité en ligne de ces systèmes et réduire ainsi la latence de détection d'erreur, qui est souvent considérée comme un paramètre déterminant dans une approche de test en ligne, notamment quand il s'agit d'application critiques.

Les nouveaux registres BILBO Optiques (OBILBO) proposés sont les éléments de base de l'architecture proposée. De ce fait, nous avons présenté les différents modes de fonctionnement de ces registres et les possibilités qu'ils offrent pour assurer une vérification fonctionnelle des réponses des systèmes en cours d'exécution, sans pour autant ralentir ou interrompre leur fonctionnement normal.

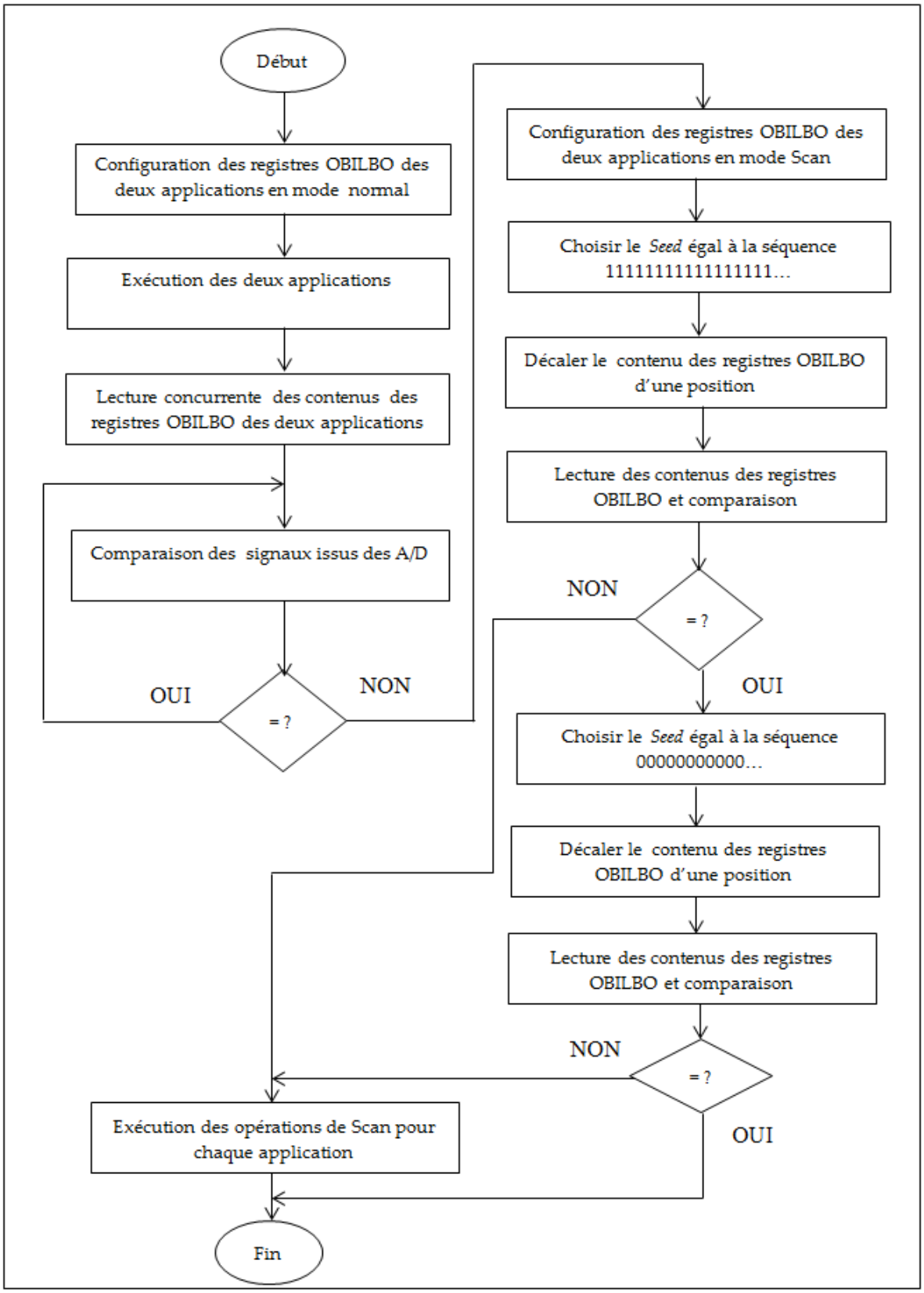


FIGURE 3.18 – Stratégie de test en ligne proposée.

Grâce à la combinaison de la technique optique proposée et de la duplication physique standard, les opérations de comparaison des réponses de l'application et de son module dupliqué sont accomplies avec une latence de faute d'un cycle d'horloge. Néanmoins, cette approche nécessite plus de hardware.

Il faut noter, que la tâche onéreuse de génération de vecteurs de test est désormais évitée car les entrées usuelles sont-elles mêmes les nouvelles données de test. La modélisation du schéma proposé et la validation de cette approche seront présentées dans le quatrième chapitre.

Chapitre 4

MODÉLISATION ET SIMULATION VHDL DE L'APPROCHE DÉVELOPPÉE

Résumé *: Dans ce chapitre, après une brève description du langage VHDL-AMS, les différents éléments constituant le schéma proposé sont modélisés et simulés sous forme d'entités simples puis, un exemple d'application, faisant appel à ces éléments de base pour former un système de test complet, est considéré. Enfin, les résultats de simulation de cette application sont présentés et interprétés.*

4.1 Introduction

Le langage VHDL est un standard IEEE (IEEE 1076-1993) pour la modélisation, la simulation et la synthèse de systèmes matériels logiques (HDL - *Hardware Description Language*). Il est aujourd'hui très largement utilisé et est supporté par tous les environnements d'aide à la conception de circuits et de systèmes électroniques (EDA - *Electronic Design Automation*) [125-126].

Le langage VHDL-AMS (*VHSIC-Hardware Description Language – Analog and Mixed Systems*) normalisé en décembre 1999 sous la référence IEEE 1076.1-1999 [127], a été développé comme une extension du langage VHDL (IEEE 1076-1993). Le VHDL-AMS permet la modélisation et la simulation de circuits et de systèmes logiques, analogiques et mixtes [128].

En tant que sur-ensemble du VHDL, le VHDL-AMS profite des capacités de ce langage à modéliser des objets abstraits manipulant des signaux quantifiés à des moments discrets dans le temps. Aux instructions concurrentes et aux instanciations des modèles du VHDL, le VHDL-AMS ajoute les instructions simultanées qui permettent de manipuler des valeurs

à temps continu transportées par les objets "*Quantity*", et le concept de synchronisation des noyaux de simulations numériques et analogiques utiles lorsque l'on veut forcer des points de simulation à des temps donnés. Les instructions simultanées mettent en jeu des équations différentielles pouvant être simples, sélectives ou conditionnelles et qui relient les quantités considérées. Le VHDL-AMS dispose aussi de l'objet "*Terminal*" jouant le rôle de nœud de connexion pour le domaine analogique. L'objet "*Terminal*" est associé à une "*Nature*" qui définit un domaine physique. Afin que l'on puisse résoudre l'ensemble des équations différentielles, le critère de solvabilité doit être validé localement et les conditions initiales doivent être spécifiées [129,130].

4.2 Organisation d'un modèle VHDL-AMS

4.2.1 Unités de conception

L'unité de conception (*Design unit*) est le plus petit module compilable séparément. VHDL-AMS offre cinq types [131] d'unités de conception (Figure 4.1) :

- La déclaration d'entité (*Entity declaration*) ;
- Le corps d'architecture (*Architecture body*), ou plus simplement architecture ;
- La déclaration de configuration (*Configuration declaration*) ;
- La déclaration de paquetage (*Package declaration*) ;
- Le corps de paquetage (*Package body*).

Les trois premières unités de conception (déclaration d'entité, architecture et déclaration de configuration) permettent la description de l'aspect matériel d'un système, alors que les deux dernières (déclaration et corps de paquetage) permettent de grouper des informations pouvant être réutilisées pour la description de plusieurs systèmes différents.

Les trois unités de conception : déclaration d'entité, déclaration de configuration et déclaration de paquetage, sont qualifiées de *primaires* (*Primary unit*s, marquées par un "P" à la figure 4.1), car elles décrivent une vue externe (le "*quoi*" de la boîte noire). Les unités primaires déclarent ce qui est accessible aux autres parties du modèle. Elles permettent le partage et la réutilisation de ressources (modèles et algorithmes). Elles ne contiennent que le minimum d'information nécessaire pour l'utilisation de ces ressources.

Les deux autres unités de conception : architecture et corps de paquetage, sont qualifiées de *secondaires* (*Secondary units*, marquées par un "S" à la figure 4.1), car elles décrivent une vue interne particulière (une réalisation ou le "*comment*" de la boîte noire).

Les unités secondaires regroupent des ressources seulement visibles par leur unité primaire correspondante (déclaration d'entité pour l'architecture et déclaration de paque-

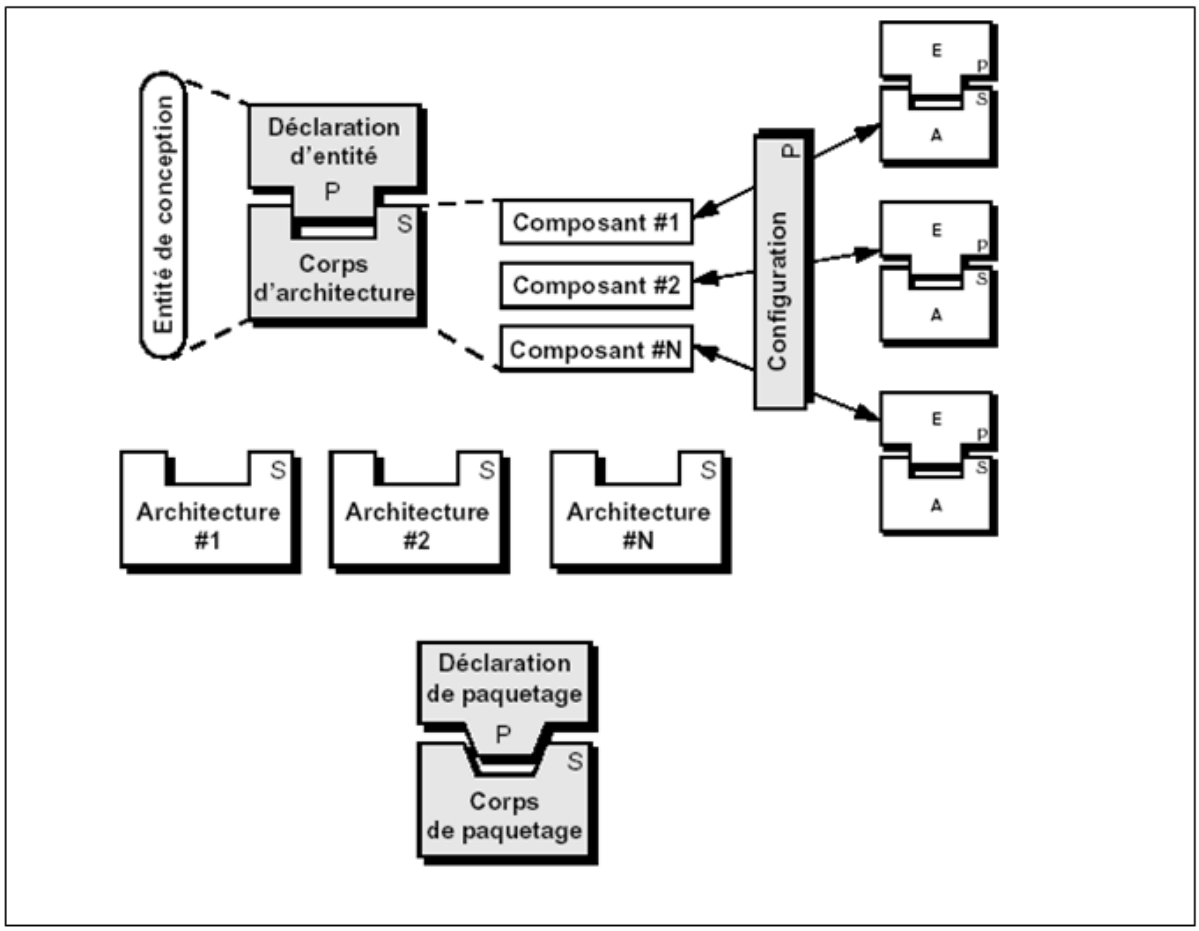


FIGURE 4.1 – Unités de conception VHDL-AMS (en gris).

tage pour le corps de paquetage). Elles contiennent des détails d'implémentation que l'on masque pour éviter de surcharger l'utilisateur avec des détails inutiles et/ou éviter des modifications malencontreuses de modèles et d'algorithmes de base. Plusieurs unités secondaires peuvent être définies pour une même unité primaire.

4.2.2 Entité de conception

L'entité de conception (*Design entity*) est l'abstraction de base en VHDL-AMS. Elle représente une portion d'un système matériel possédant une interface entrée-sortie et une fonction bien définies. Une entité de conception est constituée d'une déclaration d'entité et d'un corps d'architecture correspondant.

Une entité de conception peut représenter un système matériel à plusieurs niveaux de complexité : un système entier, un sous-système, une carte, un circuit intégré, une cellule complexe (par exemple : ALU, mémoire, convertisseur A/N, filtre, amplificateur

opérationnel, etc.), une porte logique ou un transistor.

Une *configuration* permet de définir comment plusieurs entités de conception sont assemblées pour constituer le système complet. Une entité de conception peut être décrite comme une hiérarchie de *composants* (*Component*) interconnectés. Chaque composant peut être lié à une entité de conception de plus bas niveau qui définit la structure ou le comportement de ce composant. Une telle décomposition d'une entité de conception et les liens de ses composants éventuels à d'autres unités de conception forment une hiérarchie représentant le système complet.

4.2.3 Déclaration d'entité

La déclaration d'entité définit l'interface d'un modèle avec le monde extérieur au moyen de *ports* (*Ports*). Les ports peuvent être de plusieurs classes :

- Les ports de classe "*Signal*" définissent des canaux de communication directionnels (entrées (mode IN), sorties (mode OUT) ou bidirectionnels (mode INOUT)) modélisant des signaux logiques.
- Les ports de classe "*Terminal*" définissent des points de connexions analogiques adirectionnels pour lesquels les lemmes de Kirchhoff sont satisfaits.
- Les ports de classe "*Quantity*" définissent des points de connexions analogiques directionnels (entrées (mode IN), sorties (mode OUT)) pour lesquels les lemmes de Kirchhoff ne doivent pas être satisfaits.

La déclaration d'entité peut également définir des *paramètres génériques* (*Generic parameters*) qui serviront à rendre le modèle plus général.

4.2.4 Description structurelle et configuration

Une vue interne (architecture) possible en VHDL-AMS est une description structurelle pour laquelle le modèle est une interconnexion de composants, avec éventuellement un nombre de niveaux hiérarchiques non limité.

Une architecture structurelle peut être décrite de deux manières. La première manière utilise des *déclarations de composants* pour définir les *besoins* de l'architecture. Ces déclarations sont purement locales et ne sont pas nécessairement reliées à des entités de conception particulières. Une *déclaration de configuration* est nécessaire pour établir ces liens.

VHDL-AMS définit la notion de *configuration par défaut* (*Default configuration*) : une instance de composant est automatiquement associée à une entité de conception dans la bibliothèque de travail si une telle entité possédant la même interface que le composant

existe (mêmes noms d'entité et de composant, mêmes nombres, classes et types de ports, mêmes noms de ports déclarés dans le même ordre).

La spécification d'une entité de conception sans nom d'architecture est légale en VHDL-AMS. Dans ce cas, l'architecture *la plus récemment analysée* (MRA - *Most Recently Analyzed*) est implicitement considérée.

4.2.5 Terminaux et natures

Le " *Terminal*" est un objet VHDL-AMS utile pour la spécification de points de connexions pour lesquels les lois de Kirchhoff sont satisfaites. Les terminaux permettent de définir des branches qui elles-mêmes servent de support à la spécification d'équations liant les grandeurs de branches associées, usuellement la tension et le courant pour des systèmes électriques.

Un "*Terminal*" appartient à une *nature* qui représente un domaine d'énergie particulier (électrique, mécanique, thermique, etc.). Chaque domaine d'énergie est caractérisé par deux classes de grandeurs liées à des effets physiques. Les grandeurs "*entre*" (*Across*) représentent un effort (par exemple : la tension pour les systèmes électriques, la vitesse pour les systèmes mécaniques, la température pour les systèmes thermiques).

Les grandeurs "*dans*" (*Through*) représentent un flot (par exemple : le courant pour les systèmes électriques, la force pour les systèmes mécaniques, le débit de chaleur pour les systèmes thermiques). Ces deux classes de grandeurs permettent de définir la notion d'énergie et de puissance d'un système physique.

Une déclaration de nature en VHDL-AMS ne définit pas directement les grandeurs physiques relatives à chacune des classes "*entre*" et "*dans*", mais plutôt leur type, c'est-à-dire l'ensemble des valeurs (nécessairement réelles) que ces grandeurs peuvent prendre. Les grandeurs elles-mêmes seront définies au niveau des branches.

4.3 Modélisation du schéma de principe

Le schéma de principe de l'approche de test proposée dans le chapitre 3 peut être vu comme une interconnexion de composants simples qui, peuvent être modélisé par VHDL comme des simples entités de conception. Ces entités de conception seront ensuite instanciées pour former le système entier de test. Dans ce qui suit, nous présentons en détails chaque modèle VHDL-AMS correspondant à chacun de ces composants sous forme d'entité de conception dont nous présentons les résultats de simulation. Cela permettra de valider ces modèles avant de les insérer dans le schéma global.

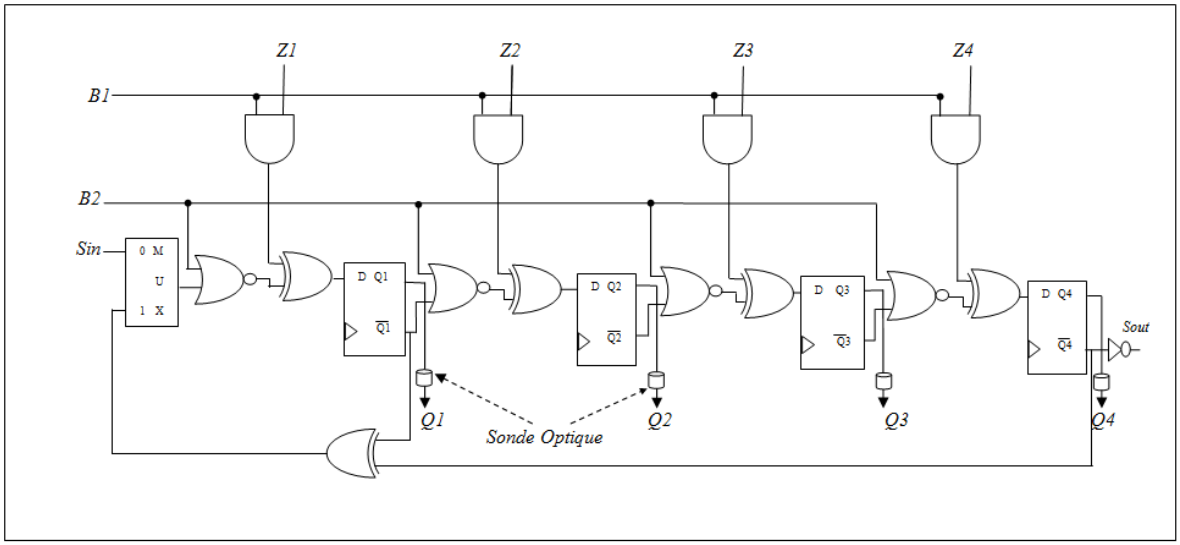


FIGURE 4.2 – Diagramme logique de l'OBILBO.

4.3.1 Modèle VHDL-AMS du registre OBILBO

Le registre OBILBO est l'élément de base du schéma proposé. Il est constitué d'un registre BILBO standard et de sondes optiques associées à ces sorties comme le montre le schéma de la figure 4.2. Nous avons développé deux modèles VHDL-AMS (un modèle pour le registre BILBO et un modèle pour la sonde optique) qui sont présentés dans ce qui suit.

4.3.1.1 Modèle VHDL du registre BILBO

Le registre BILBO (Figure 4.3) peut être vu comme une interconnexion de quatre bascules et d'un ensemble de portes logiques. On peut, à cet effet, faire appel à ces composants qui existent dans presque toutes les bibliothèques des outils VHDL d'aujourd'hui, et on aura affaire à un modèle structurel. Néanmoins, on peut le modéliser de façon comportementale, afin de ne pas cibler une technologie précise et laisser le choix au synthétiseur, pour rendre le modèle portable. La figure 4.4 donne une modélisation VHDL comportementale possible de ce registre.

Il faut noter que nous avons utilisé l'outil MAX+plus II 10.1 d'ALTERA pour simuler et ensuite implémenter le modèle sur FPGA, en prouvant ainsi sa validité.

La figure 4.5 montre le fonctionnement du BILBO en mode *Scan* ($B2B1 = B = 00$) où, on a généré une séquence d'entrée "1111111...", à partir de $t = 0ns$, puis une séquence "0000000...", à partir de $t = 100ns$, via l'entrée $SCAN\_IN$. Ces séquences ont été récupérées à la sortie $SCAN\_OUT$. Les entrées Z_i ont été maintenues à un état

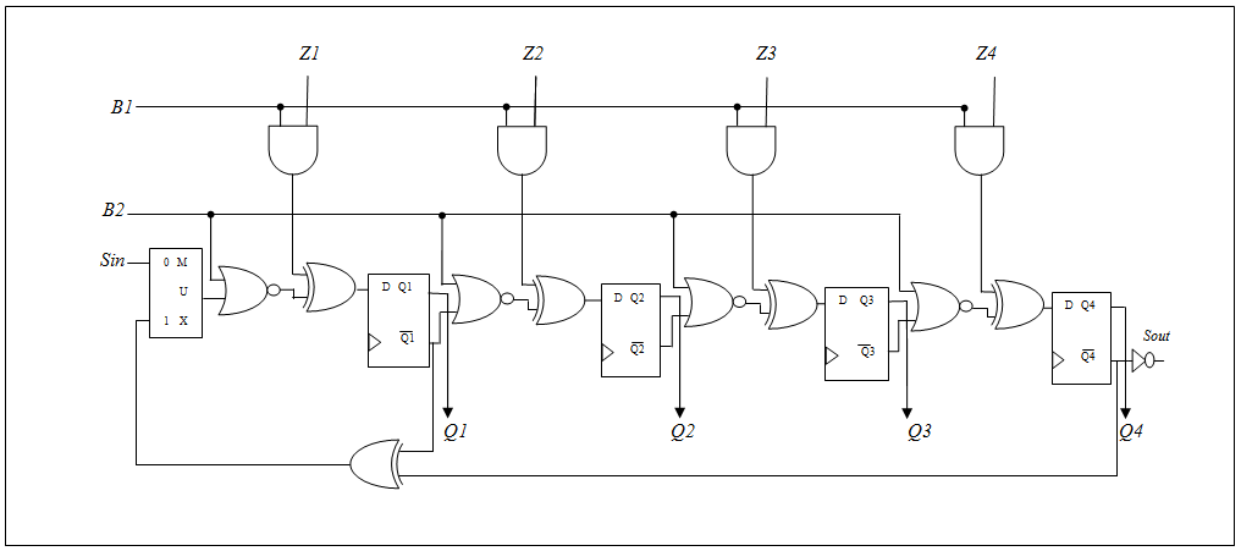


FIGURE 4.3 – Diagramme logique du BILBO.

indéterminé. Notez que, à partir de $t = 250ns$, c'est le mode RESET ($B2B1 = B = 10$) qui est simulé.

Le mode BIST (Test) est illustré par la figure 4.6. Dans ce mode de test, si les lignes de commande, $B2B1 = B = 01$, et si les entrées Z_i sont positionnées à 0 (à un état logique bas) alors le BILBO fonctionne comme un générateur de séquences pseudo-aléatoires. Pour rappel, pour que le BILBO fonctionne comme un générateur de séquences pseudo-aléatoires, l'état initial du registre doit être différent de l'état "0000" c'est pourquoi nous avons choisi arbitrairement, l'état initial comme l'état "1111". Cet état à été chargé via les lignes d'entrées Z_i. Néanmoins, on peut agir sur l'entrée *PRESET* qui n'a pas été évoquée dans les paragraphes précédents. On constate que le BILBO génère périodiquement, 15 états différents ($2^4 - 1$) comme illustré par la figure 4.6.

Cependant, pour mettre en évidence le mode de fonctionnement MISR qu'offre le BILBO, on a opté pour un exemple concret qui sera implémenté par la suite sur circuit FPGA. Cet exemple est relativement simple ; il consiste en un additionneur complet à $1 - bit$ où on a appliqué la technique BILBO pour tester son bon fonctionnement.

À cet effet, nous avons développé un modèle VHDL structurel pour un additionneur complet à 1-bit parfait (*Golden circuit*). Ce modèle a été dupliqué et sa structure a été modifiée en injectant une faute de collage-à-0 à l'entrée de la porte OU-logique en gris, représentée à la figure 4.7. Les deux circuits qui reçoivent les mêmes données d'entrées devraient en principe produire les mêmes signatures qui seront stockées dans les deux registres BILBO. Cependant, à cause de la présence de la faute susmentionnée, la signature des circuits sera différente.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY BILBO_V02 IS
PORT(
   RESET, CLK, SCAN_IN, Z1,Z2,Z3,Z0: IN BIT;
   SCAN_OUT : OUT BIT;
   B : IN BIT_VECTOR(2 DOWNTO 1);
   Q : OUT BIT_VECTOR(3 DOWNTO 0) );
END BILBO_V02 ;
ARCHITECTURE ARCH OF BILBO_V02 IS
   SIGNAL F0,F1,F2,F3:BIT;
   SIGNAL Y0,Y1,Y2,Y3:BIT;
BEGIN
PROCESS(CLK)
BEGIN
IF RESET = '1' THEN
   F0<='0';
   F1<='0';
   F2<='0';
   F3<='0';
ELSIF CLK'EVENT AND CLK = '1' THEN
CASE B IS
WHEN "00" => --SHIFT MODE
   F0 <= SCAN_IN;
   F1 <= F0;
   F2 <= F1;
   F3 <= F2;
WHEN "01" => --MISR MODE
   F0 <= Y0;
   F1 <=Y1;
   F2 <=Y2;
   F3 <=Y3;
WHEN "10" => --RESET MODE
   F0<='0';
   F1<='0';
   F2<='0';
   F3<='0';
WHEN "11" => --PARALLEL LOAD MODE
   F0 <= Z0;
   F1 <= Z1;
   F2 <= Z2;
   F3 <= Z3;
WHEN OTHERS => --SEED
   F0<='0';
   F1<='0';
   F2<='0';
   F3<='0';
END CASE;
END IF;
END PROCESS;
Q(0)<=F0;
Q(1)<=F1;
Q(2)<=F2;
Q(3)<=F3;
SCAN_OUT <= F3;
Y0<= Z0 XOR F3;
Y1<= F0 XOR Z1 XOR F3;
Y2<= F1 XOR Z2;
Y3<=F2 XOR Z3;
END ARCH;
```

FIGURE 4.4 – Modèle VHDL comportemental du BILBO.

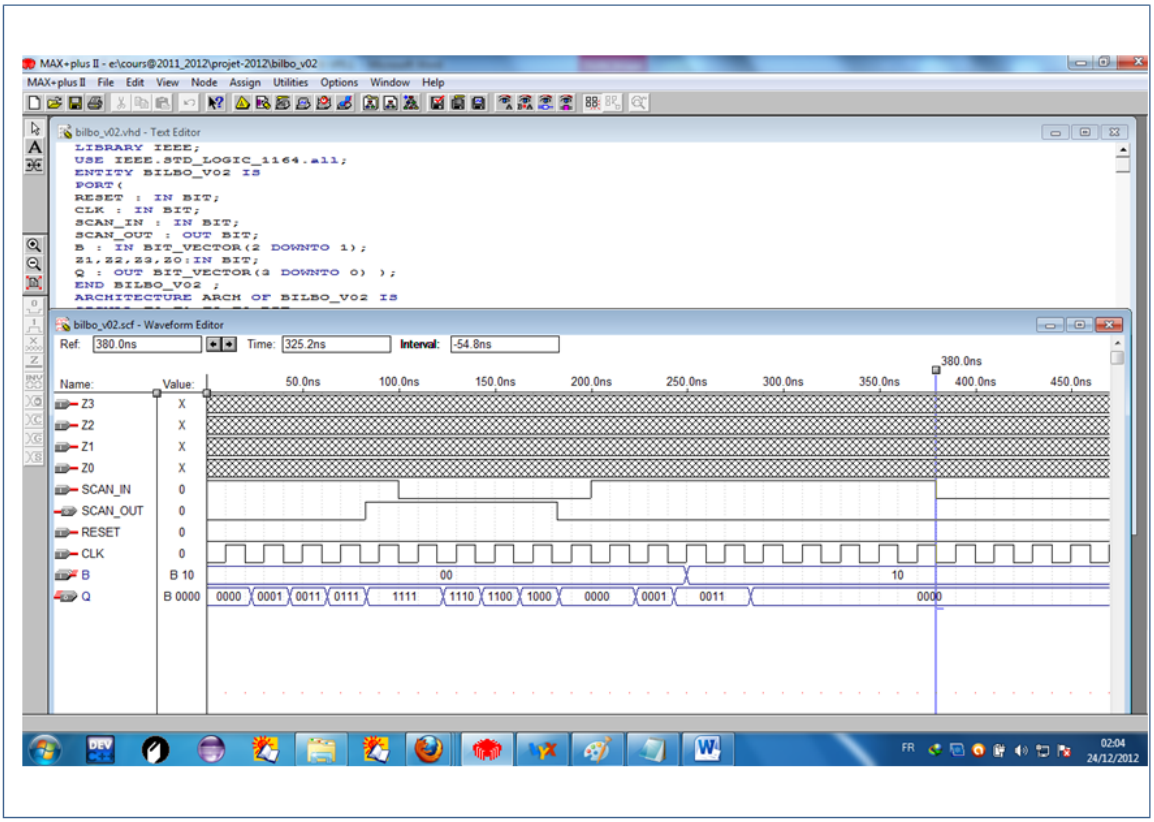


FIGURE 4.5 – Simulation du BILBO en mode Scan.

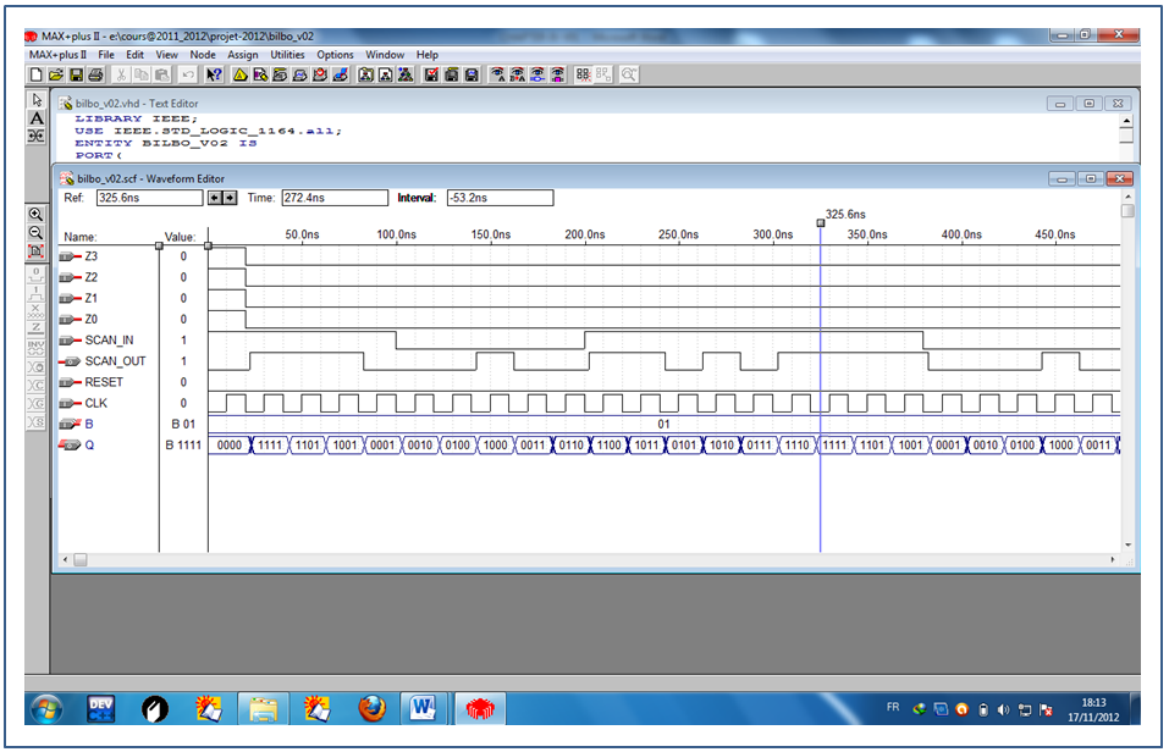


FIGURE 4.6 – Simulation du BILBO en mode BIST (Test).

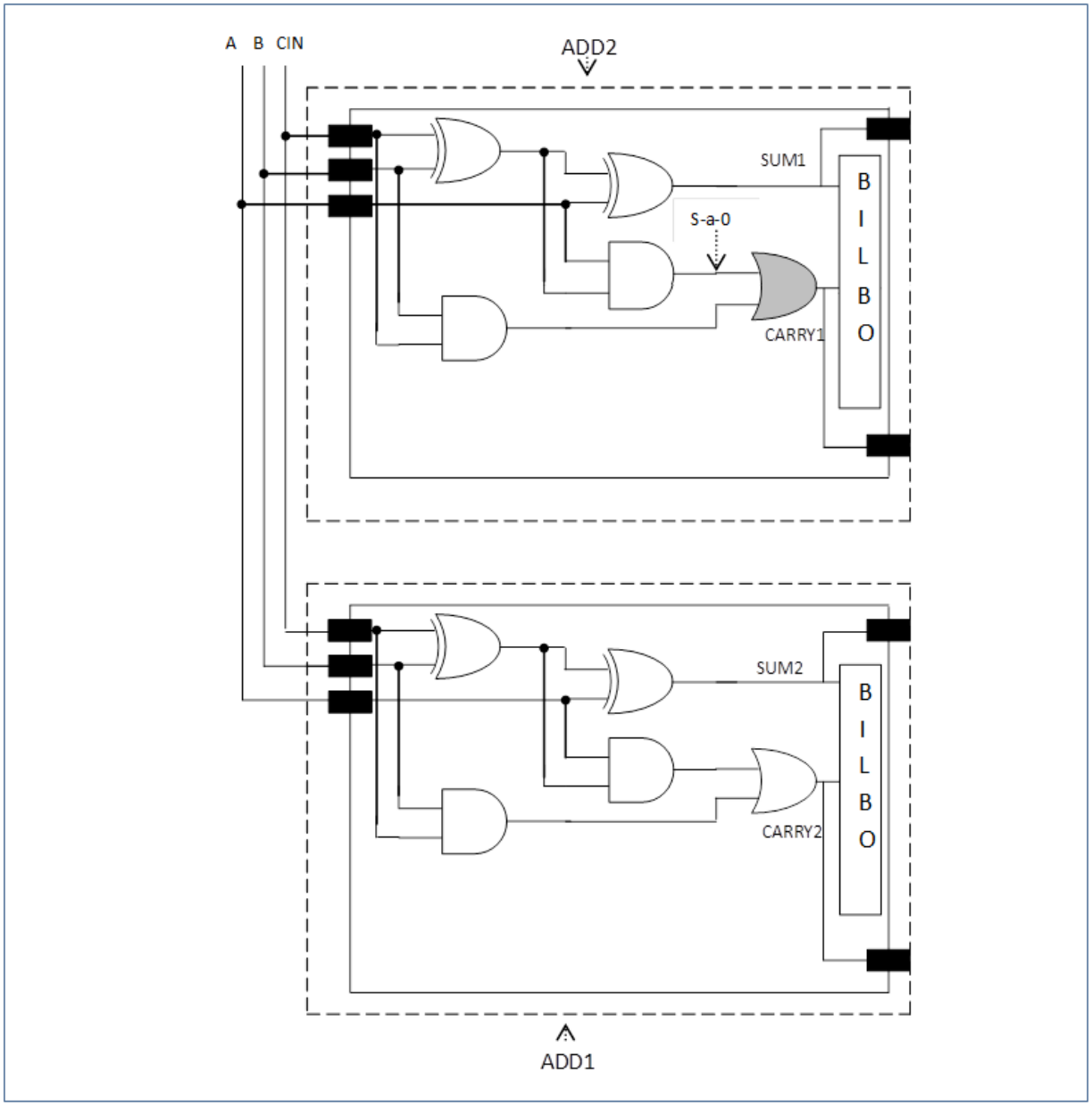


FIGURE 4.7 – Exemple d'utilisation de la technique BILBO (mode MISR).

La table de vérité de la figure 4.8 met en évidence les réponses qui devraient être produites par les deux circuits additionneurs aux différentes combinaisons des séquences d'entrée.

Par procédé de la division polynomiale, en choisissant un polynôme caractéristique $P(x) = 1 + x + x^4$, on aura dans les deux cas de figures les signatures indiquées sur la même table vérité.

A	B	CIN	SUM1 (Free)	SUM2 (Faulty)	CARRY1 (Fault Free)	CARRY2 (Faulty)	SIGNATURE
0	0	0	0	0	0	0	/
0	0	1	1	1	0	0	/
0	1	0	1	1	0	0	/
0	1	1	0	0	1	0	/
1	0	0	1	1	0	0	/
1	0	1	0	0	1	0	/
1	1	0	0	0	1	1	/
1	1	1	1	1	1	1	/
ADD1 (*FREE*)							**1011**
ADD2 (*FAULTY*)							**0101**

FIGURE 4.8 – Table de vérité du circuit de la figure 4.7.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.MATH_REAL.ALL;
------------------------------ ADD1 FAULT-FREE
ENTITY PLUS IS
PORT(A,B, CIN ,SUM1,CARRY1: IN BIT);
END ;
ARCHITECTURE STRUCTURE OF PLUS IS
SIGNAL S1 : BIT;
BEGIN
PROCESS
BEGIN
     S1 <= A XOR B ;
     SUM1 <= S1 XOR CIN ;
     CARRY1 <= (A AND B) OR (S1 AND CIN) ;
     WAIT ON A, B, CIN,COUT,S1 ;
END PROCESS;
END STRUCTURE;
```

FIGURE 4.9 – Modèle VHDL structurel de l'additionneur complet $ADD1$ (1-bit).

Le code VHDL de l'additionneur sein ($ADD1$) est illustré à la figure 4.9. Pour modéliser le deuxième additionneur ($ADD2$) qui présente dans sa structure la faute de collage-à-0 en question, le code VHDL du modèle du circuit sein, $ADD1$, a été modifié comme celui de la figure 4.10.

La simulation du circuit de la figure 4.7 a permis d'obtenir les résultats consignés à la figure 4.11 où, le signal *error* représente le résultat de comparaison entre la signature du circuit $ADD1$, $Q1$, et la signature du circuit $ADD2$, $Q2$. Celui-ci, est positionné à un état

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.MATH_REAL.ALL;
-----------------------------ADD2 FAULTY
ENTITY PLUS_F IS
PORT(A,B,CIN,SUM2,CARRY2: IN BIT);
END ;
ARCHITECTURE STRUCTURE OF PLUS_F IS
SIGNAL S1 : BIT ;
BEGIN
PROCESS
BEGIN
     S1 <= A XOR B ;
    SUM2 <= S1 XOR CIN ;
    CARRY2 <= (A AND B) OR '0';-- INJECTION DE LA FAUTE (S-a-0)
     WAIT ON A, B, CIN,COUT,S1 ;
END PROCESS;
END STRUCTURE;
```

FIGURE 4.10 – Modèle VHDL structurel de l'additionneur complet $ADD2$ (*Faulty*).

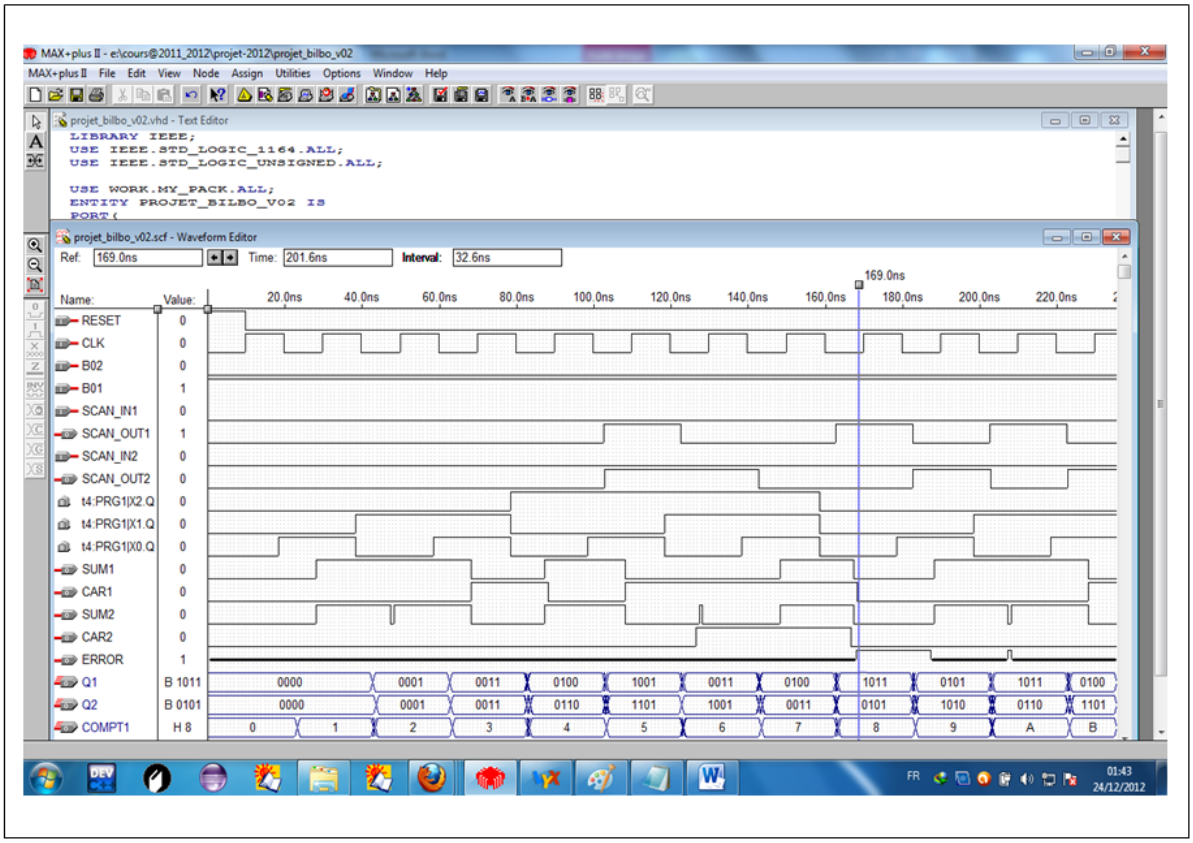


FIGURE 4.11 – Simulation du circuit de la figure 4.7.

logique bas, quand les deux signatures sont égales, et à un état logique haut dans le cas contraire. Cependant, la signature est le contenu du registre BILBO à la fin de la séquence BIST c'est-à-dire, après avoir entré tous les bits d'une séquence d'entrée (8 bits dans notre cas). Donc la lecture du registre BILBO est effectuée après le 8^{eme} top d'horloge. Nous avons alors ajouté un compteur, $COMPT1$, pour déclencher le comparateur au moment voulu. La sortie de ce dernier, le signal *error*, est en effet mise en état haute impédance en dehors des instants de lecture et comparaison des signatures. Outre ce compteur, nous avons également utilisé un deuxième pour générer les séquences d'entrées *A, B* et *CIN*, représentées sur la figure 4.11 par les signaux (*t4 :PRG1/X2.Q*), (*t4 :PRG1/X1.Q*) et (*t4 :PRG1/X0.Q*) respectivement. Les modèles de ces deux compteurs ainsi que ceux de l'$ADD1$, de l'$ADD2$ et du BILBO ont été regroupés dans package, nommé MY_PACK. Le code VHDL de ce package ainsi que le programme principal correspondant à cette application seront présentés en annexe de ce document (Annexe A).

Toutefois, il faut noter que les résultats de simulation de ce circuit (Figure 4.11) coïncident avec ceux calculés théoriquement (Figure 4.8). De plus, cette application a été implémentée sur circuit FPGA type EPF8282ALC84-4. A cet effet, nous avons exploité un kit FPGA CIC-310 doté de système de visualisation des signaux d'entrée et de sortie tels que l'afficheur sept segments et des LED, ce qui nous a permis de vérifier les réponses produites par ce circuit et ainsi prouver la validité du modèle VHDL développé.

4.3.1.2 Modèle VHDL-AMS d'une sonde optique

La sonde optique [20] peut être vue comme un composant simple comme illustré à la figure 4.12 qui a comme entrée, un signal électrique (Tension V (0-5V)) à ces bornes correspondant aux deux états logiques 0 et 1, et comme sortie une quantité réelle (puissance optique).

Pour rappel, les équations régissant ce modèle sont :

$$n = n_0 - \frac{1}{2} n_0^3 r_{13} E_z \tag{4.1}$$

Où :

- n : est l'indice de réfraction du matériau optoélectronique (PLT) ;
- n_0 : est l'indice de réfraction en absence du champ électrique ;
- r_{13} : est un coefficient optoélectronique du matériau ;
- E_z : est l'amplitude du champ électrique dans la direction Z.

La composante E_z est régie par la relation (4.2).

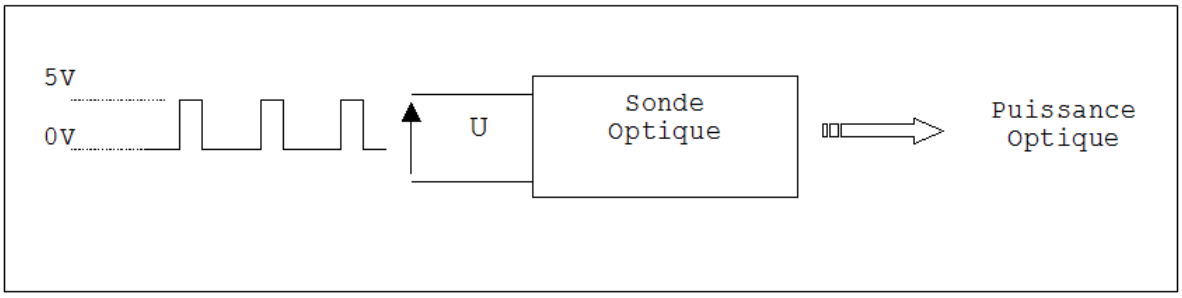


FIGURE 4.12 – Modèle de sonde optique.

$$E_z = \frac{V}{e} \tag{4.2}$$

V : est la tension appliquée aux bornes de la sonde (0 – $5V$).

La phase de sorite, dans ce cas, est régie par la relation (4.3).

$$\phi = \frac{2\pi n}{\lambda_0} e \tag{4.3}$$

Avec :
- n : L'indice de réfraction donné par l'équation (4.1) ;
- λ_0 : La longueur d'onde du rayon optique dans l'espace ;
- e : L'épaisseur du matériau PLT ;

L'intensité globale est donnée par la relation (4.4).

$$I = I_{Max} \left(1 - \frac{1}{1 + M \sin 2\left(\frac{\phi}{2}\right)} \right) \tag{4.4}$$

I_{Max} est l'intensité maximale réfléchie et M est un coefficient qui dépend de la réflectivité des deux miroirs (Voir figure 3.14).

Le modèle VHDL-AMS [132] correspondant au circuit de la figure 4.12 est présenté à la figure 4.13.

Il est maintenant possible de simuler la sonde optique décrite par ce modèle VHDL-AMS. La simulation de ce modèle donne pour u_in une tension en créneau de 0 à $5V$, les résultats obtenus sont présentés sur la figure 4.14.

La réponse de cette sonde est caractérisée par une puissance de sortie (représentée par le signal "*ligth*" sur la figure 4.14) de l'ordre de $873.7mW$ pour une tension d'entrée (représentée par le signal "u_in" sur la figure 4.5) de $5V$ et une puissance de $857.0mW$ pour une tension de $0V$ à ses bornes. Cette puissance optique peut être convertie facilement

```
LIBRARY DISCIPLINES,IEEE;
USE DISCIPLINES.ELECTROMAGNETIC_SYSTEM.ALL;
USE IEEE.MATH_REAL.ALL;
ENTITY OP_PROBE IS
PORT (SIGNAL SBIN: IN BIT ;  QUANTITY LIGHT : OUT REAL);
END OP_PROBE;
ARCHITECTURE ARCH  OF  OP_PROBE IS
QUANTITY U_IN  : REAL ;
QUANTITY EZ, N_EZ, PHI  :  REAL  ;
CONSTANT N0: REAL  := 2.415 ;
CONSTANT LAMBDA : REAL := 0.63E-6 ;
CONSTANT E :REAL := 120.0E-9 ;
CONSTANT R13 : REAL := 55.0E-12 ;
CONSTANT IMAX : REAL := 0.50 ; CONSTANT M : REAL := 0.95 ;
BEGIN
  IF  (SBIN='1') USE U_IN == 5.0 ;
  ELSE U_IN == 0.0 ; END USE ;
  EZ == U_IN/E ;  N_EZ == N0 -( 0.5* R13*EZ*N0**3 ) ;
  PHI == ((2.0*3.14*E*N_EZ)/LAMBDA);
  LIGHT  == IMAX *(1.0-((1.0/M*SIN(2.0*(2.0*PHI)))));
END ARCH;
```

FIGURE 4.13 – Modèle VHDL-AMS de la sonde optique.

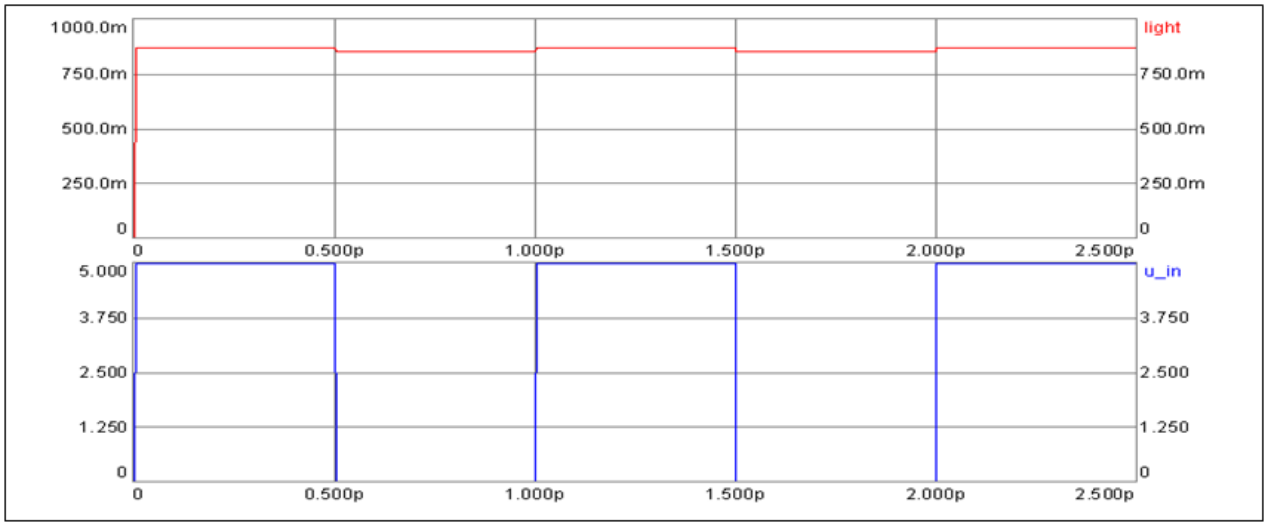


FIGURE 4.14 – Simulation de la sonde optique (U_IN : 0 – 5 V).

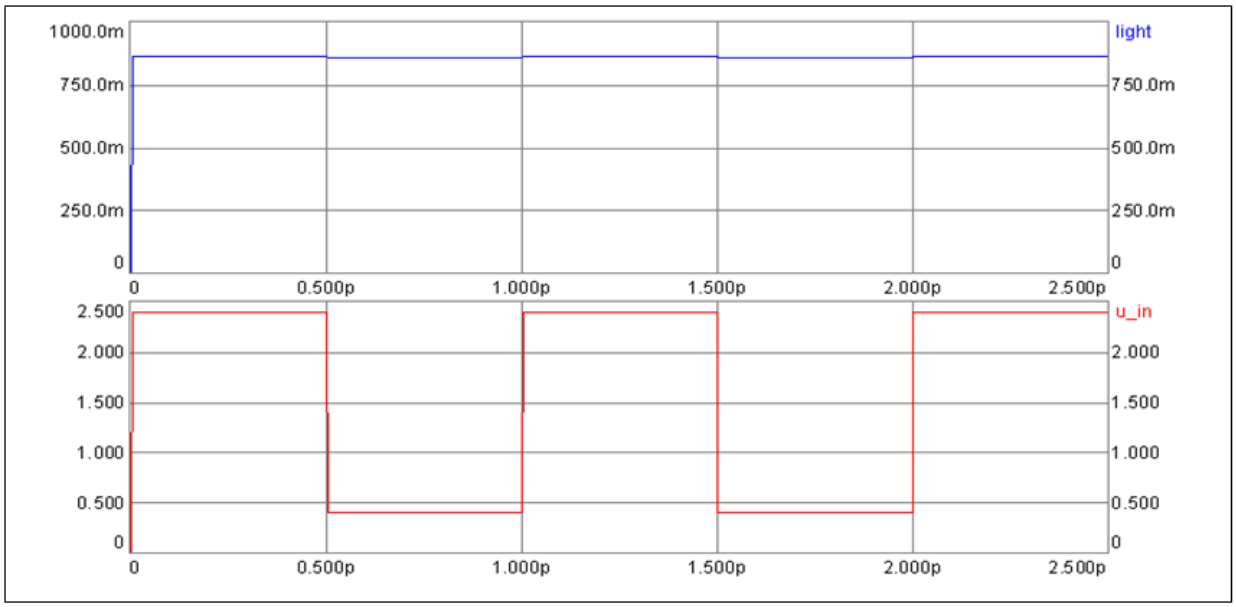


FIGURE 4.15 – Simulation de la sonde optique (U_IN : 0.4- 2.4 V).

en son équivalent électrique par des simples capteurs tels que des photodiodes. Il est à noter, que les deux niveaux de puissance différents (correspondant aux entrées 0 et $5V$) représentent la réponse aux entrées logiques '0' et '1' respectivement. Néanmoins, en pratique, les niveaux de tension comprise entre $0.4V$ et $2.4V$ sont considérés comme des états logiques indéterminés, cependant les résultats de simulation de ces niveaux ont montré que la réponse en puissance pour de telles tensions donne des niveaux de puissance différents, qui peuvent être convertis en des niveaux logiques aussi différents, alors que les états sont considérés comme indéterminés ce qui conduit à une ambiguïté. La figure 4.15 donne les résultats de simulation pour une tension u_in en créneaux de 0.4 à $2.4V$.

La réponse de la sonde, dans ce cas, est caractérisée par une puissance de sortie de l'ordre de $865.9mW$ pour $u\_in = 2.4V$ et de l'ordre de $859.2mW$ pour $u\_in = 0.4V$. Ces deux niveaux de puissance, après une conversion en des signaux électriques au moyen d'une photodiode par exemple, peuvent représenter des états logiques '1' et '0' respectivement. Il est donc impératif de prendre en compte ces états indéterminés et définir la gamme des tensions exploitables qui permet d'exclure, d'une manière simple, les états indéterminés.

4.3.2 Modèle VHDL-AMS d'une Photodiode

La photodiode est le composant dual de la sonde optique qui a comme entrée une puissance optique et comme sortie un signal électrique comme le montre la figure 4.16. Du point de vue électrique, la photodiode peut être vue comme un générateur de courant. Nous

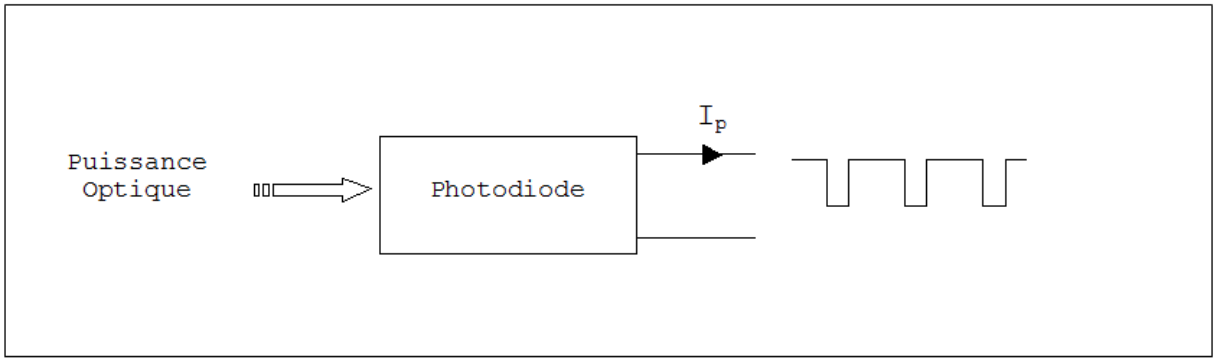


FIGURE 4.16 – Modèle de photodiode.

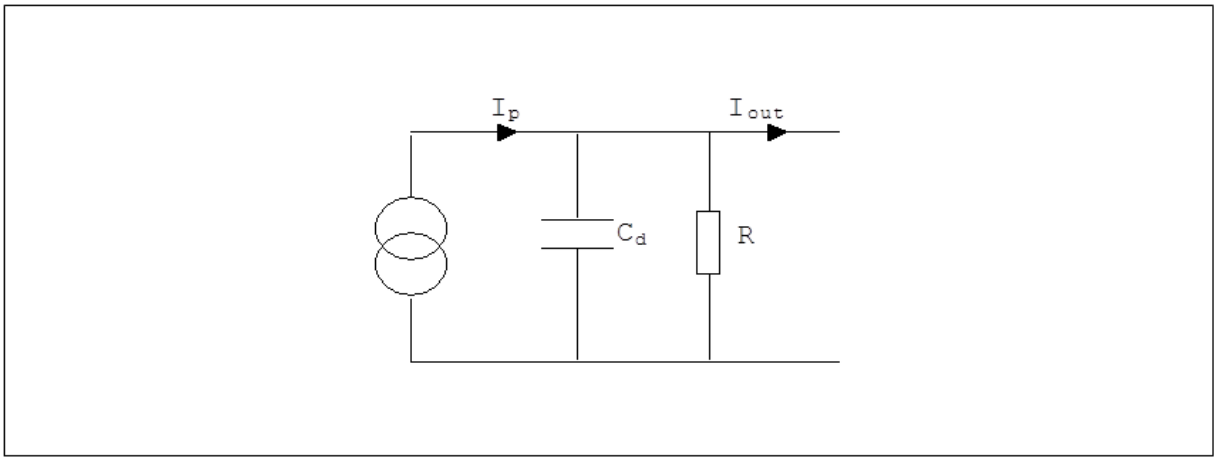


FIGURE 4.17 – Schéma équivalent d'une photodiode.

rappelons que pour faire fonctionner correctement une photodiode, elle doit être polarisée en inverse ou non polarisée, mais en dynamique, la capacité de diffusion C_d diminue le photocourant I_p . On peut donc modéliser la photodiode à l'aide du circuit de la figure 4.17.

Les équations régissant le circuit représenté sur la figure 4.17 sont :

$$I_r = \frac{V_d}{R} \tag{4.5}$$

$$I_c = \frac{C_d.d(V_d)}{dt} \tag{4.6}$$

$$I_p = -S.L \tag{4.7}$$

Où :

– S : Sensibilité de la diode

```
LIBRARY DISCIPLINES,IEEE;
USE DISCIPLINES.ELECTROMAGNETIC_SYSTEM.ALL;
USE IEEE.MATH_REAL.ALL;
ENTITY  PIN_DIODE IS
PORT (TERMINAL N1,N2 :ELECTRICAL ;
QUANTITY LIGHT : IN REAL);
END  PIN_DIODE;
ARCHITECTURE ARCH OF PIN_DIODE IS
QUANTITY VD ACROSS IP,IR,IC THROUGH N1 TO N2;
CONSTANT CD : REAL := 1.0E-15;
CONSTANT R : REAL := 1.0E3;
CONSTANT S : REAL := 0.13 ;
BEGIN
   IR == VD/R;
   IC == CD*VD'DOT;
   IP == - S*LIGHT;
END ARCH;
```

FIGURE 4.18 – Modèle VHDL-AMS d'une Photodiode.

- C_d : Capacité de diffusion
- R : Résistance de fuite

Ce circuit peut être modélisé en VHDL-AMS comme le montre la figure 4.18.

La simulation de ce modèle pour une entrée créneau de puissance de $857.0mW$ à $873.7mW$ (délivrée par la sonde optique) donne les résultats représentés sur la figure 4.19.

Nous constatons que la réponse de la photodiode, décrite par ce modèle, est caractérisée par un courant I_p (représenté sur la Fig.4.19 par le signal "*ip.pind*") de l'ordre de $120mA$ pour une entrée en puissance (représentée sur la Fig.4.19 par le signal "*light.pind*") de $873.7mW$ et de l'ordre de $107mA$ pour une entrée en puissance de $857.0mW$. Ces deux niveaux de courant différents, correspondant aux deux niveaux de puissance reçus par la photodiode (émis par la sonde optique), après une conversion analogique/numérique, présentent deux états logiques différents.

4.3.3 Modèle VHDL-AMS d'un convertisseur A/N

Comme en VHDL, plusieurs modèles de convertisseurs ont d'ores et déjà été développés. Pour notre application, nous avons choisi un modèle très simple, illustré par le code VHDL-AMS de la figure 4.20, où nous exploitons uniquement deux états correspondant aux deux états logiques standard.

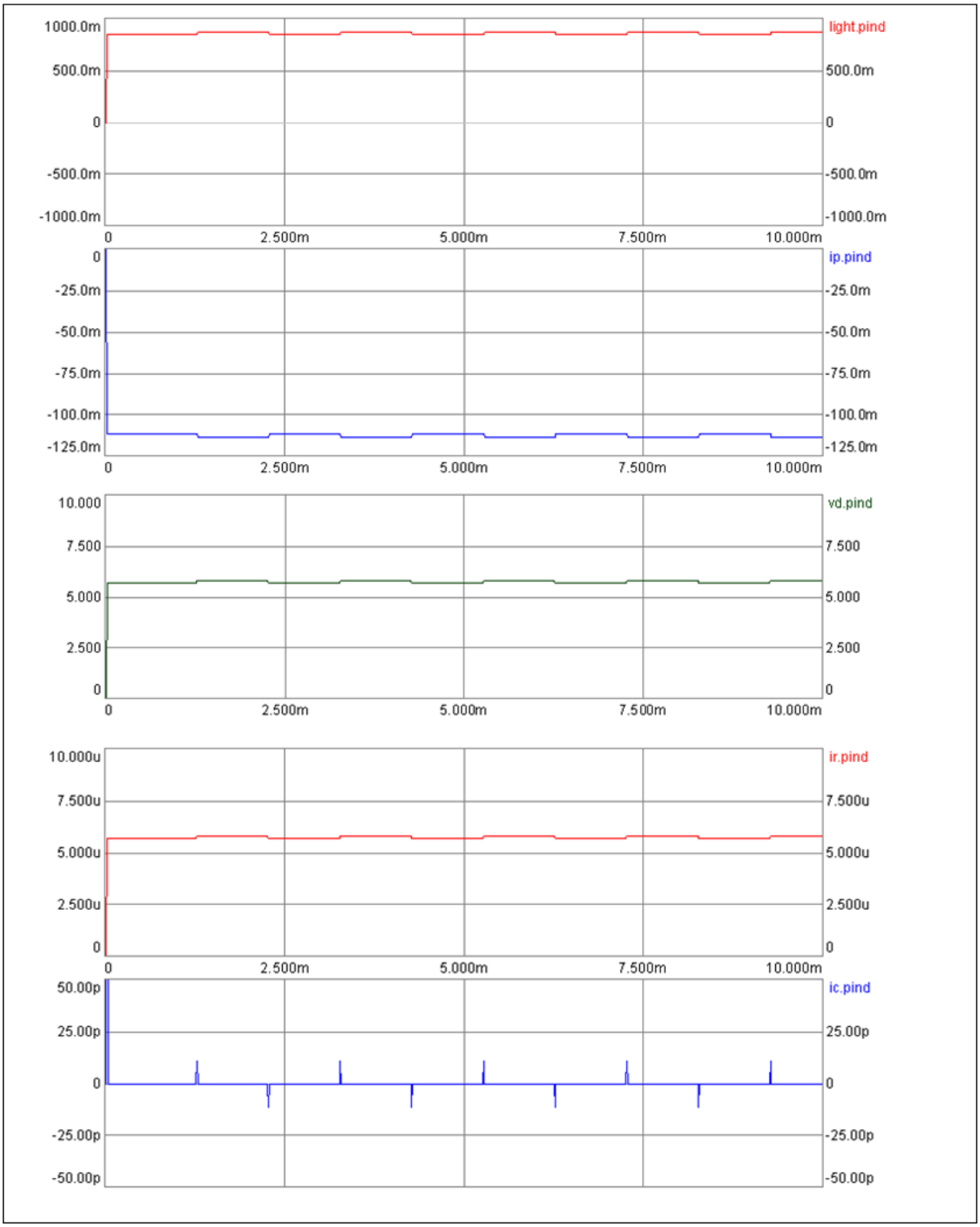


FIGURE 4.19 – Simulation de la photodiode.

```
LIBRARY DISCIPLINES,IEEE;
USE DISCIPLINES.ELECTROMAGNETIC_SYSTEM.ALL;
USE IEEE.MATH_REAL.ALL;
ENTITY  ADC  IS
GENERIC (UL: REAL:=125.1E-3; UH:REAL:=125.1E-3);
PORT (TERMINAL N1, N2: ELECTRICAL;
SIGNAL DOUT: OUT BIT:='0');
END ENTITY ADC;
ARCHITECTURE ARCH OF ADC IS
QUANTITY UIN  ACROSS  I THROUGH  N1 TO N2;
BEGIN
BREAK  DOUT=>'1' WHEN UIN'ABOVE(UL);
BREAK  DOUT=>'0' WHEN NOT UIN'ABOVE(UH);
END ARCHITECTURE ARCH;
```

FIGURE 4.20 – Modèle VHDL-AMS d'un CAN.

4.3.4 Modèle VHDL-AMS d'un banc de test de base

Après avoir développé les modules élémentaires constituant le schéma de principe de l'approche de test proposée, il est désormais possible de regrouper ces éléments pour former un banc de test de base qui permet la mise en œuvre de la technique développée. Or, la mise en œuvre de la technique de test optique développée dans ce travail exploite le moule de base (Figure 4.21) formé par deux cellules fondamentales : la cellule OBILBO et la cellule de réception et de traitement de données. Cette dernière est à son tour constituée de quatre cellules élémentaires identiques. La cellule élémentaire de réception et de traitement de données, représentée en bas de la figure 4.21, est formée d'une photodiode et d'un CAN. Ce banc de test de base est alors un système mixte, à cause de la nature des signaux mis en jeu. Ces signaux sont de types numériques, analogiques et optiques. Les signaux numériques, représentant les états logiques des broches d'E/S de la cellule OBILBO. Par ailleurs, les signaux de sortie produits par la cellule OBILBO, en réponse aux signaux d'entrée numériques, sont capturés et transmis, grâce à l'ensemble des sondes optiques insérées dans la cellule OBILBO, sous forme de signaux optiques à un système distant (cellule de réception et de traitement de données). Ce dernier, muni de capteurs optoélectroniques (photodiodes), qui convertissent les signaux optiques reçus en signaux électriques, et de CAN, qui fournissent les états logiques correspondants. Le signal récupéré aux bornes de la résistance R est l'image du courant traversant la photodiode, et donc l'image du signal optique reçu (puissance optique reçue par la photodiode). Il s'agit donc de la modélisation d'un système pluridisciplinaire présentant quatre modes de fonctionnement dictés par la cellule OBILBO.

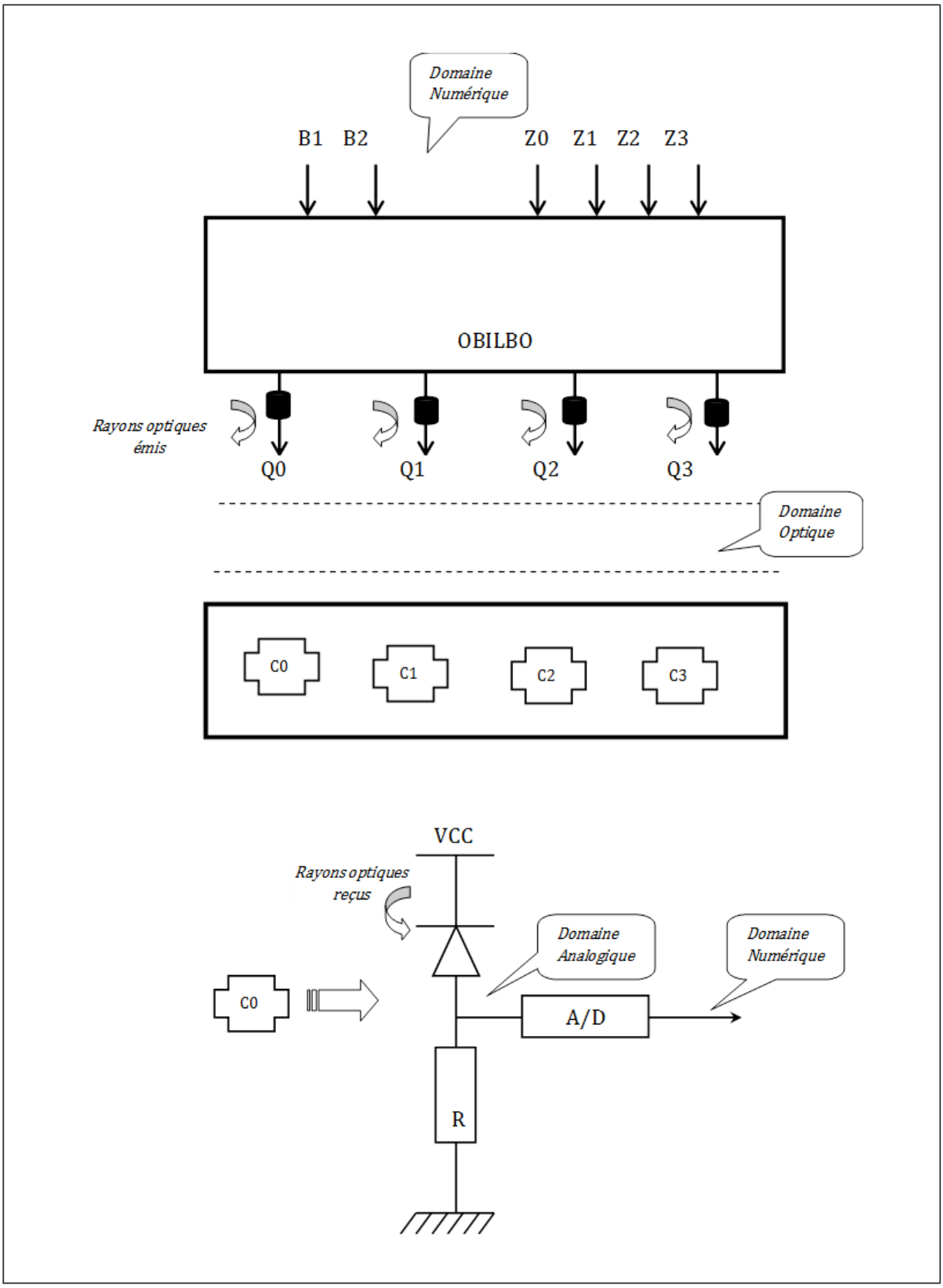


FIGURE 4.21 – Schéma de principe du système de test de base.

```
LIBRARY IEEE,DISCIPLINES;
USE DISCIPLINES.ELECTROMAGNETIC_SYSTEM.ALL;
USE IEEE.MATH_REAL.ALL;
USE WORK.ALL;
ENTITY OPBILBO IS
END OPBILBO;
ARCHITECTURE ARCH OF OPBILBO IS
    SIGNAL CLK,RESET,SCAN_IN,SCAN_OUT:BIT;
    SIGNAL  B :  bit_vector(2 downto 1);
    SIGNAL  Z :  bit_vector(3 downto 0);
    SIGNAL  Q :  bit_vector(3 downto 0);
    TERMINAL N1,N2,N3,N4 : ELECTRICAL;
    SIGNAL  S1,S2,S3,S4,TEST1,TEST2,TEST3,TEST4 : bit;
    QUANTITY  LIGHT1,LIGHT2,LIGHT3,LIGHT4 : real ;
 BEGIN
 S1<=Q(0);S2<=Q(1);S3<=Q(2);S4<=Q(3) ;
 OPBIL:ENTITY BILBO (ARCH)PORT MAP(RESET,CLK,SCAN_IN,SCAN_OUT,B,Z,Q);
 OPP1:ENTITY OP_PROBE PORT MAP (S1,LIGHT1);
 PIND1:ENTITY PIN_DIODE PORT MAP (N1,electrical_ground,LIGHT1);
 ADC1: ENTITY ADC PORT MAP(N1,electrical_ground ,TEST1);
 OPP2:ENTITY OP_PROBE PORT MAP (S2,LIGHT2);
 PIND2:ENTITY PIN_DIODE PORT MAP (N2,electrical_ground,LIGHT2);
 ADC2: ENTITY ADC PORT MAP(N2,electrical_ground ,TEST2);
 OPP3:ENTITY OP_PROBE PORT MAP (S3,LIGHT3);
 PIND3:ENTITY PIN_DIODE PORT MAP (N3,electrical_ground,LIGHT3);
 ADC3: ENTITY ADC PORT MAP(N3,electrical_ground ,TEST3);
 OPP4:ENTITY OP_PROBE PORT MAP (S4,LIGHT4);
 PIND4:ENTITY PIN_DIODE PORT MAP       (N4,electrical_ground,LIGHT4);
 ADC4: ENTITY ADC PORT MAP(N4,electrical_ground ,TEST4);
END  ARCH;
```

FIGURE 4.22 – Modèle VHDL-AMS du système de test de base.

La figure 4.22 montre une modélisation VHDL-AMS structurelle possible de notre système de test de base. En général, dans un modèle pareil, on peut distinguer trois parties différentes : la partie déclarative, la partie instanciation et la partie génération des stimuli.

La partie déclarative consiste en la déclaration des librairies standard (par exemple : la librairie IEEE) ainsi qu'en la bibliothèque de travail dans laquelle se trouvent les composants et les entités déjà compilées séparément et prêts à la réutilisation. La partie instanciation consiste donc à l'appel de ces composants et la manière dont ils sont interconnectés pour réaliser une fonction donnée. Enfin, la partie génération des stimuli consiste à produire les séquences d'entrées nécessaires à l'exécution de la fonction désirée. Il faut noter que, pour que ce modèle soit générique et ainsi peut être supporté par différents outils de simulation VHDL-AMS, cette dernière partie (génération de stimuli) a été supprimée du code VHDL-AMS de ce modèle puisque certains outils ont leur propre générateur de

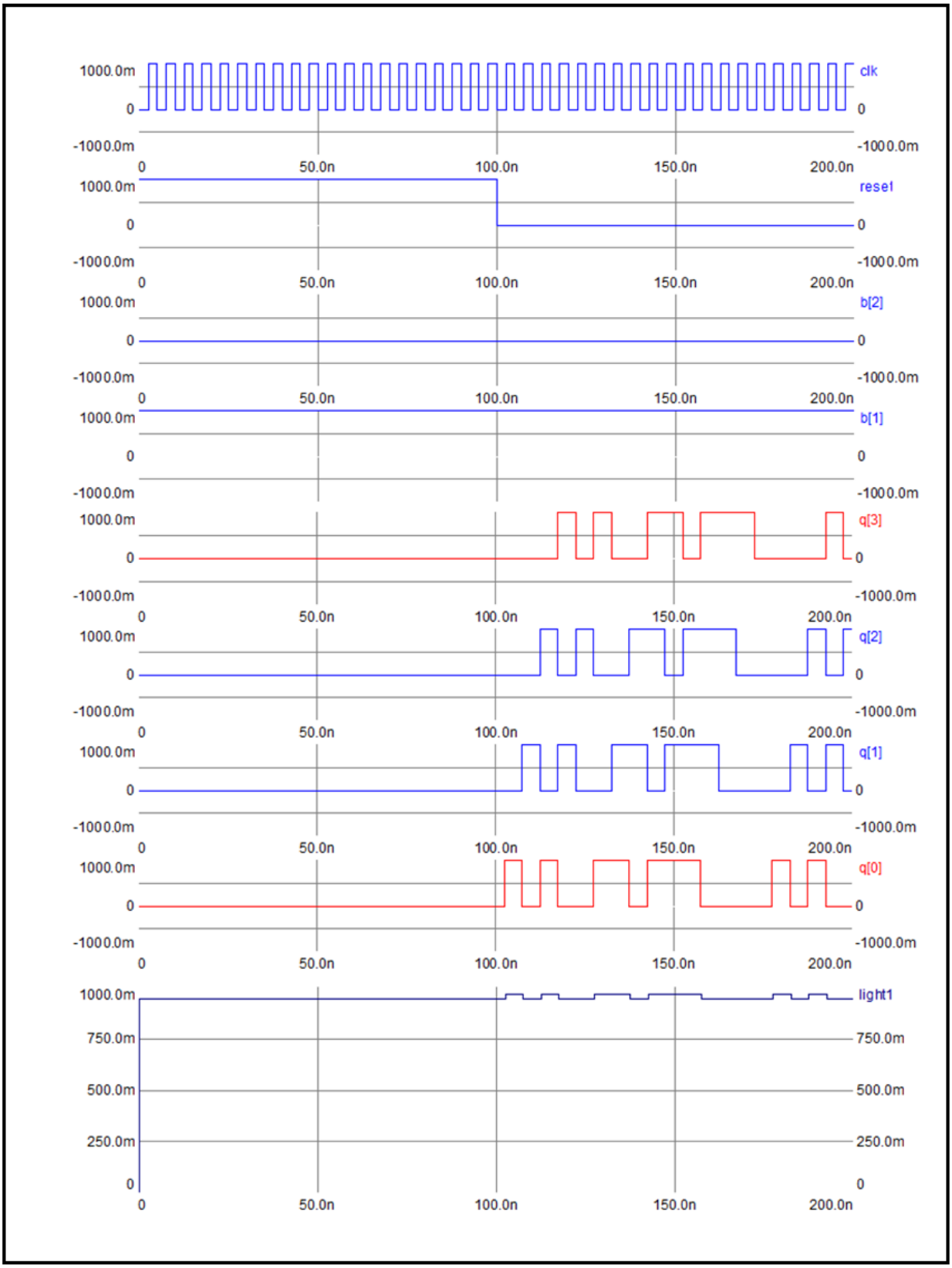


Figure 4.23 : Résultats de simulation du banc de test de base.

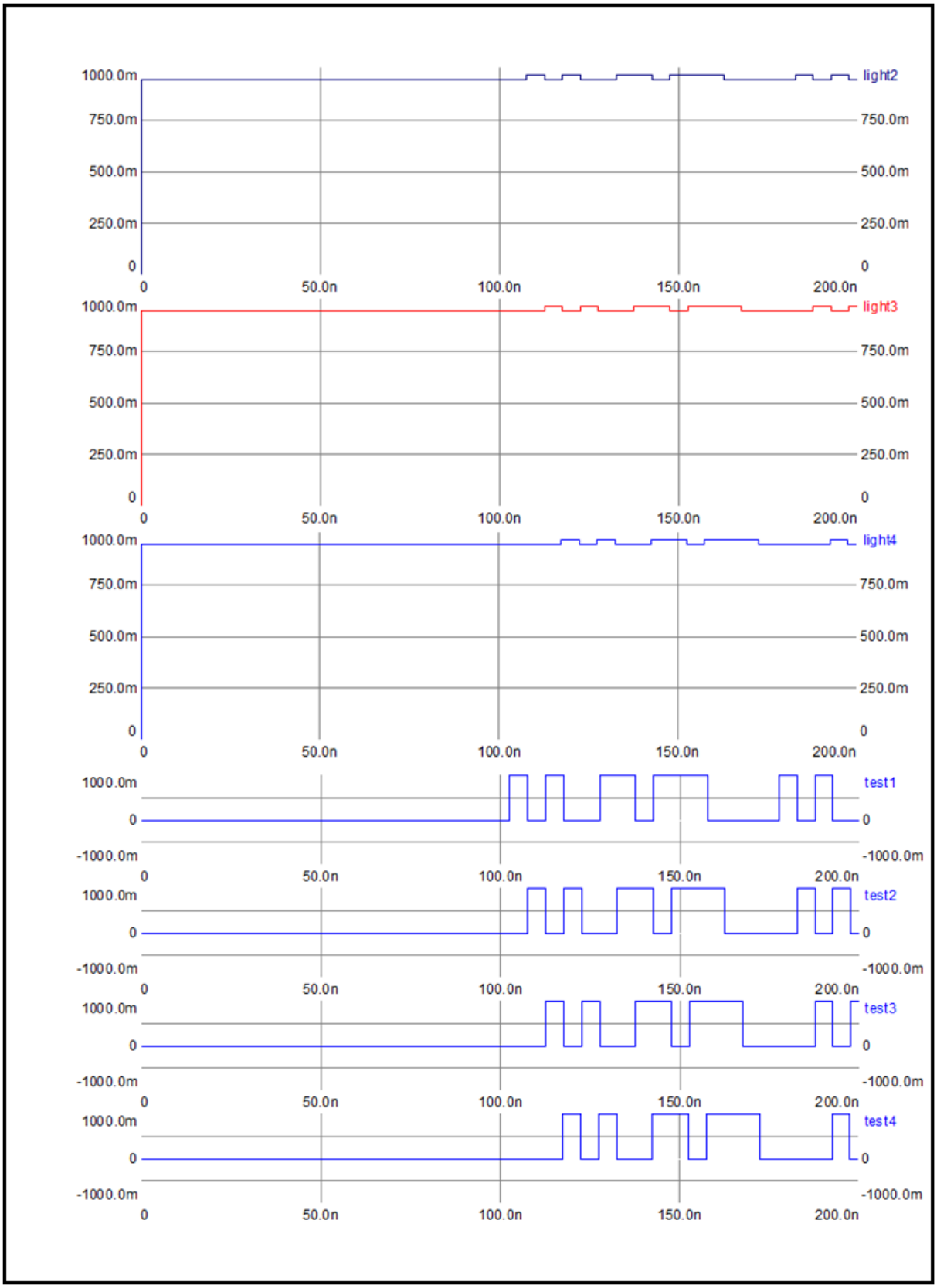


FIGURE 4.23 – Résultats de simulation du banc de test de base (Suite).

stimuli.

L'exécution du code VHDL-AMS de la figure 4.22 a permis d'obtenir les résultats consignés sur la figure 4.23 où, entre $0ns$ et $100ns$ le mode de $RESET$ est simulé ($RESET = 1$). Ensuite, entre $100ns$ et $200ns$ c'est le mode MISR. Les signaux $b[2]$ et $b[1]$ sont les lignes de commande, $q[3], q[2], ...q[0]$ sont les états logiques des signaux de sorties du registre OBILBO tandis que $light1, light2, ...light4$ sont les signaux transmis par les sondes optiques correspondant aux états de $q[0], q[1], ...q[3]$ respectivement. $light1, light2, ...light4$ sont aussi les entrées des quatre photodiodes. Les sorties des quatre CAN, placés après les quatre photodiodes, ont livré les signaux binaires $test1, test2, test3$ et $test4$, en fonction de l'état logique 0 ou 1 des sorties $q[0], q[1], ...q[3]$ de l'OBILBO respectivement. Ainsi, un traitement à un système distant est alors possible. En effet, nous pouvons désormais exploiter le mode MISR de l'OBILBO, pour transmettre la signature à système distant comme une "*Signature Optique*" [115] pour qu'elle soit analysée, et ainsi vérifier l'intégrité du CUT.

Par ailleurs, en choisissant le *Seed* égal à la séquence 1111(0000) et en réalisant une seule opération de décalage, le contenu de l'OBILBO peut être transmis, de façon parallèle, par les SO considérées. Ainsi, le test de faute de collage à 0(1) du registre OBILBO peut être effectuée de manière parallèle, afin de réduire le nombre des opérations de décalage série. Ceci nous permet de vérifier les résultats des opérations de *Scan* de l'OBILBO en une seule opération de lecture du contenu du registre OBILBO et en plus à un système distant alors qu'il fallait normalement N-tops d'horloge (N étant le nombre de bascules formant le registre OBILBO) pour accomplir la même tâche avec un registre BILBO standard. Ainsi, le caractère série du registre BILBO est désormais évité. Cependant, notre approche permet non seulement de vérifier le bon fonctionnement du CUT en mode *Off-line*, mais elle peut aussi être étendue au mode en ligne. Ceci peut être réalisé lorsque l'OBILBO est configuré dans son mode de fonctionnement normal. Dans le paragraphe suivant, nous allons considérer un exemple motivant montrant cette possibilité.

4.4 Exemple d'application

Considérons la portion de la puce DSP (*Digital Signal Processing*) TMS32010 de la figure 4.24. Diverses architectures BIST ont été appliquées à cette puce pour des fins de test *Off-line*. En effet, quatre architectures BIST ont été proposées dans [30]. Parmi ces architectures, une solution BILBO a été proposée et dans laquelle le registre $R3$ est remplacé par un BILBO pour le test de fautes de collage du multiplieur.

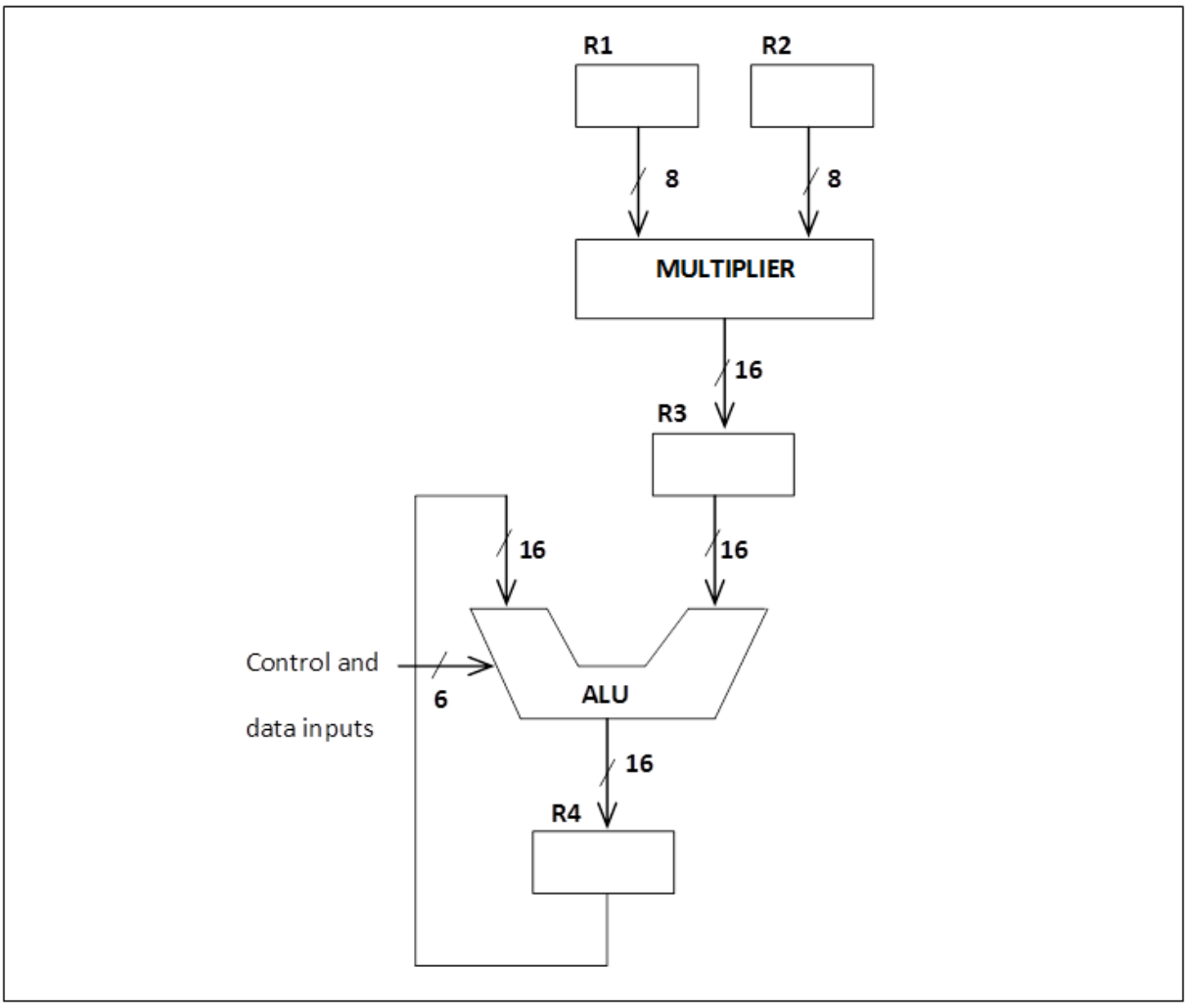


FIGURE 4.24 – Portion de la puce DSP TMS32010.

Pour appliquer notre méthode à cet exemple typique afin de tester cette puce en mode en ligne, supposons maintenant que le registre BILBO est modifié comme sur la figure 4.2 et le système est dupliqué comme le montre la figure.4.25. Le registre OBILBO devrait être configuré en son mode de fonctionnement normal (*Parallel Load Mode*). Les multiplieurs, $Mulplier1$ et $Mulplier2$, qui reçoivent les mêmes données en entrées, devraient produire, en principe, les mêmes résultats dans le cas où ils sont exempts de fautes ("*Fault-free*") et des réponses différentes si l'un des deux présente une faute ("*Faulty*"). Il s'agit donc de mettre en œuvre la stratégie de test en ligne proposée pour détecter, en temps réel, la présence de ces fautes sans avoir recours à aucune opération de génération de vecteur de test. De ce fait, on se propose de développer un modèle VHDL-AMS décrivant ce système sous test et la simulation de celui-ci.

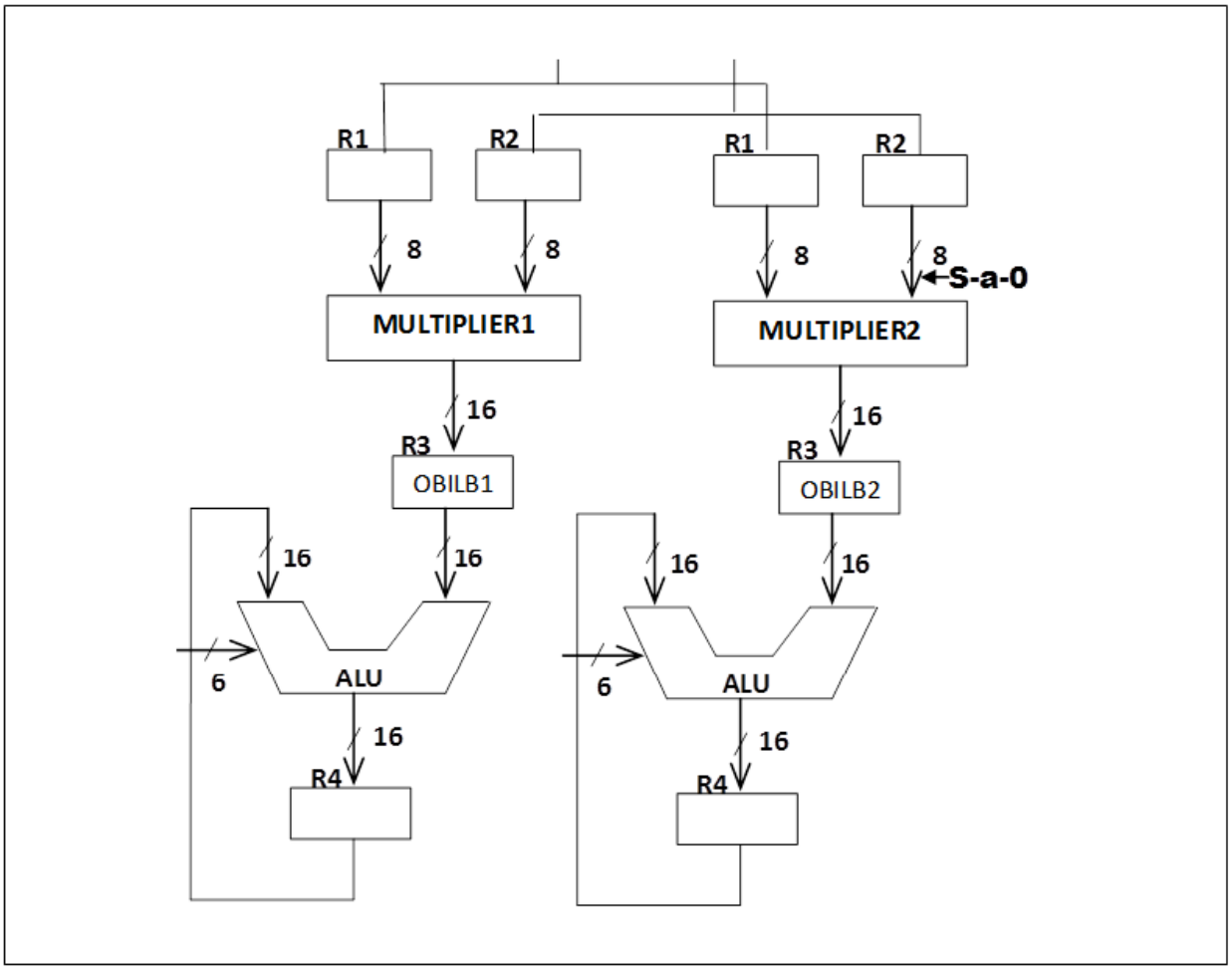


FIGURE 4.25 – Exemple de CUT avec OBILBO.

4.4.1 Modélisation

Pour valider l'approche développée à travers cet exemple type, nous supposons qu'un un multiplieur est défectueux, soit le $Multiplier2$, en raison de la faute de collage-à-0 sur la première ligne (LSB) de l'entrée de celui-ci, et le multiplieur, $Multiplier1$, est exempt de faute ("*Fault-free*"). La table de vérité de la figure 4.26 met en évidence les réponses à certaines entrées qui devraient être produites par les deux circuits multiplieurs.

Le modèle VHDL-AMS qui décrit ce système consiste donc en l'instanciation des différents composants constituant ce dernier. Outre les modèles VHDL-AMS déjà évoqués dans les paragraphes précédents, nous avons développé un modèle structurel pour le multiplieur $Multiplier1$ et ensuite nous avons déduit celui du multiplieur $Multiplier2$, en modifiant le code VHDL-AMS du modèle du $Multiplier1$ de la même façon que pour l'additionneur $ADD2$, pour injecter la faute de collage en question.

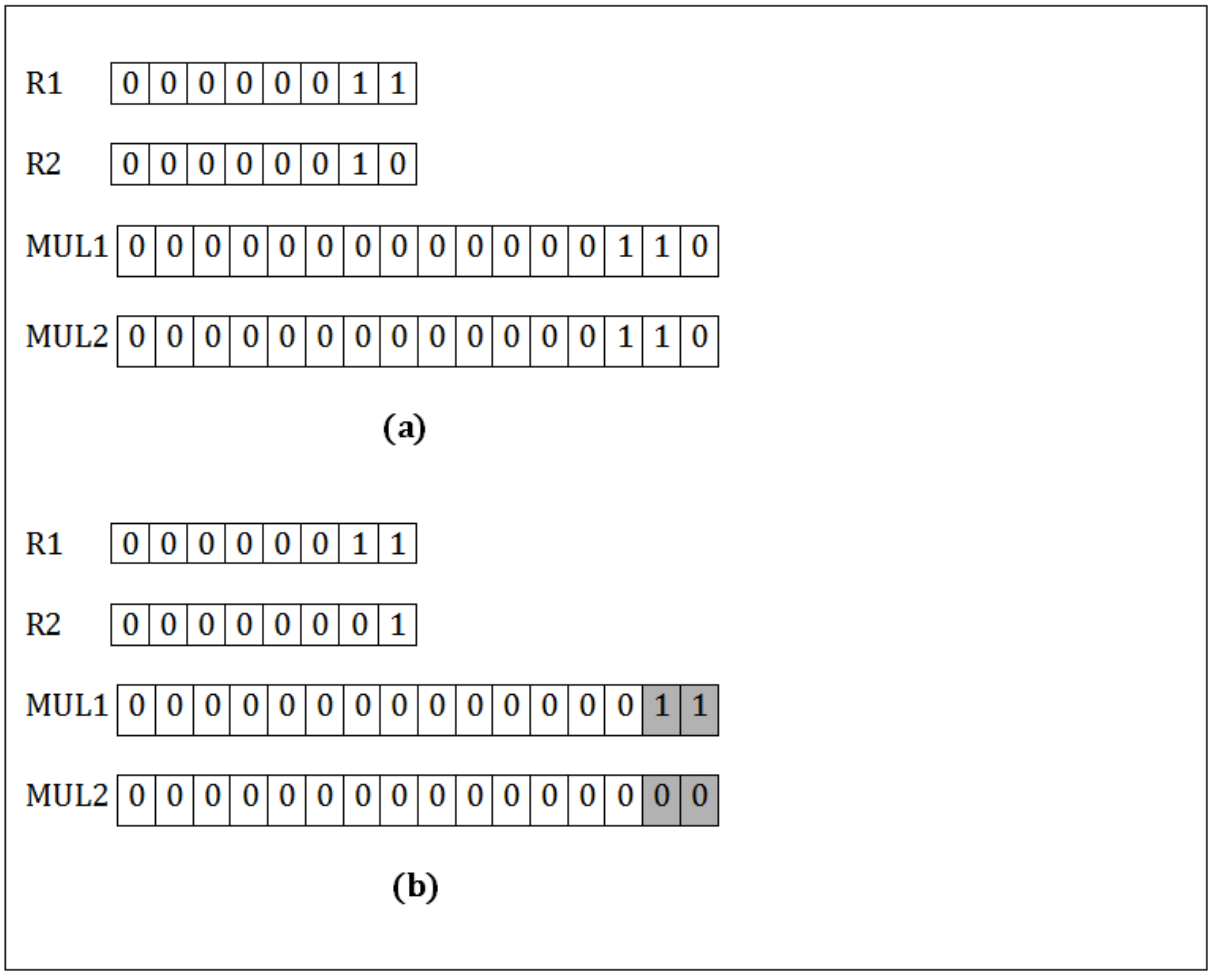


FIGURE 4.26 – Table de vérité de l'exemple de la figure 4.25.

Toutefois, il faut noter que la procédure de réception et le traitement des signaux optiques issus des registres OBILBO des deux circuits considérés est analogue à celle décrite dans le paragraphe 4.3. Néanmoins, l'ajout d'un contrôleur est nécessaire pour comparer les signaux délivrés par les CAN associés à ces applications. Ce contrôleur doit générer un signal *error* dès que les résultats produits par les deux circuits multiplieurs sont différents. Une modélisation possible de celui est illustrée par le code VHDL-AMS de la figure 4.27.

Les modèles VHDL-AMS des circuits multiplieurs, $Mul.1$ et $Mul.2$, ainsi que le modèle global résultant de l'application considérée est donné en Annexe B de ce document.

4.4.2 Simulation

La simulation du modèle global de cet exemple a été réalisée par l'outil VHDL-AMS *hAMSter* qui n'intègre pas de générateur de stimuli. À cet effet, nous avons eu recours aux

```
LIBRARY IEEE;
LIBRARY DISCIPLINES ;
USE DISCIPLINES.ELECTROMAGNETIC_SYSTEM.ALL;
USE IEEE.MATH_REAL.ALL;
USE WORK.ELECTRICAL_SYSTEM.ALL;
ENTITY CHECKER IS
PORT (CLK :  IN BIT ;SIGNAL E1,E2 : INOUT  BIT; ERROR : OUT BIT );
END ENTITY CHECKER;
ARCHITECTURE BEHAV OF CHECKER IS
BEGIN
PROCESS(CLK)
BEGIN
      IF (CLK'EVENT AND CLK='1') THEN
      IF (E1/=E2) THEN ERROR <='1' ;
      ELSIF  (E1=E2) THEN ERROR <='0' ;
      END IF ;
      END IF ;
END PROCESS ;
END ARCHITECTURE CHECKER;
```

FIGURE 4.27 – Modèle VHDL-AMS du Contrôleur.

PROCESS pour générer les séquences de contrôle et de données de cette application. Le morceau de code VHDL suivant, par exemple, permet de générer un signal horloge, *CLK*, dont la période est égal à $1.0ns$.

```
PROCESS
BEGIN
CLK <= '0' , '1' AFTER 0.5 ns;
WAIT FOR 1 ns;
END PROCESS;
```

Les résultants de simulation de cette application sont consignés à la figure 4.28 où, $r1[0], r1[1], ...r1[7]$ et $r2[0], r2[1], ...r2[7]$ sont le contenu des registres $R1$ et $R2$ respectivement, et que seulement les deux premiers bits (poids faibles) sont représentés sur la figure 4.28. Le reste des bits sont égaux à zéro et ils ne sont pas représentés. $m1[0], m1[1], ...m1[15]$ et $m2[0], m2[1], ...m2[15]$ sont les sorties des deux multiplieurs, $MULT1$ et $MULT2$, respectivement. Notons que, seuls les trois premiers bits de sorties du multiplieur $MULT1$, $m1[0], m1[1]$ et $m1[2]$ sont représentés sur la figure 4.28 et le reste des bits sont égaux à zéro et ne sont pas représentés sur cette figure. Pour le second multiplieur $MULT2$, c'est la même chose. $Light01, light11, ...light151$ sont les signaux optiques correspondants aux bits de sorties $m1[0], m1[1], ...m1[15]$, transmis par le registre OBILBO1 tandis que, $light02, light12, ...light152$ sont les signaux optiques correspondants aux bits de sorties

$m2[0], m2[1], ...m2[15]$ transmis par le registre OBILBO2. Les signaux *light*01, *light*11,... *light*151 sont également les entrées du premier ensemble des photodiodes. Cependant, *test*01, *test*11,... *test*151 sont les signaux délivrés par le premier ensemble des CAN. De même, *test*02, *test*12,...*test*152 sont les signaux délivrés par le second ensemble des CAN. $Error0$ est le résultat de la comparaison entre les signaux *test*01 et *test*02. $Error1$ est le résultat de la comparaison entre les signaux *test*11 et *test*12. Notons que, $error0$, premier bit de sortie du contrôleur (*Checker*) est égal à 1 lorsque $m1[0]$ et $m2[0]$ sont différents et égale à 0 autrement. Pour $error1, error2, ...error15$ c'est la même chose.

Le cas (a) de la figure 4.26 est simulé entre 0ns et 10ns comme le montre la figure 4.28 où les sorties des deux multiplieurs sont égaux. Le cas (b) est simulé entre $10ns$ et $20ns$, où les sorties des deux multiplieurs (bits représentés en gris sur la figure 4.26(b)) sont différents. Il faut noter que, la faute de collage-à-0 considérée à l'entrée de la première ligne (LSB) du multiplieur $MULT2$, est détectée avec une latence d'erreur très faible (après 2 cycles d'horloge), pendant que le système continue son fonctionnement normal (Fonction Multiplication) et sans recours à aucune opération de génération de vecteurs de test externe. Nous notons également que l'unité ALU a poursuivi son fonctionnement normal sans aucune interruption pendant que l'opération de test du multiplieur est effectuée. Les signaux correspondant à l'unité ALU ne sont pas représentés ici et le registre $R4$ devrait être remplacé par un OBILBO pour effectuer le test en ligne de celle-ci.

Cependant, pour la technique de duplication physique classique (sans OBILBO), les erreurs ne peuvent être détectées que lorsqu'elles se propagent vers les sorties primaires du CUT. À titre d'exemple, considérons le système de la figure 4.25, si nous supposons que les entrées des registres, $R1$ et $R2$, sont les entrées primaires du CUT et que, les sorties du registre $R4$ sont leur sorties primaires, la faute considérée à l'entrée du multiplieur $MULT2$, ne peut être détectée que lorsqu'elle se propage vers les sorties du registre $R4$. Par conséquent, l'unité ALU et le registre $R4$ vont introduire un retard comme pour le multiplieur $MULT2$ et le registre $R3$. Ainsi, plus de cycles d'horloge seront nécessaires pour détecter cette faute. En conséquence, le temps de latence d'erreur va augmenter et il va augmenter de plus en plus avec à la complexité du CUT ce qui est bien le cas des circuits actuels. Donc, la solution proposée est plus efficace car elle réduit considérablement le temps de latence de détection d'erreur et permet également une comparaison à un système distant. La latence d'erreur est réduite, puisque plus de points de test sont insérés dans le CUT via les sondes optiques considérées dans le registre BILBO qui améliorent d'avantage l'observabilité. Néanmoins, le schéma proposé exige plus de 100% de surcoût matériel.

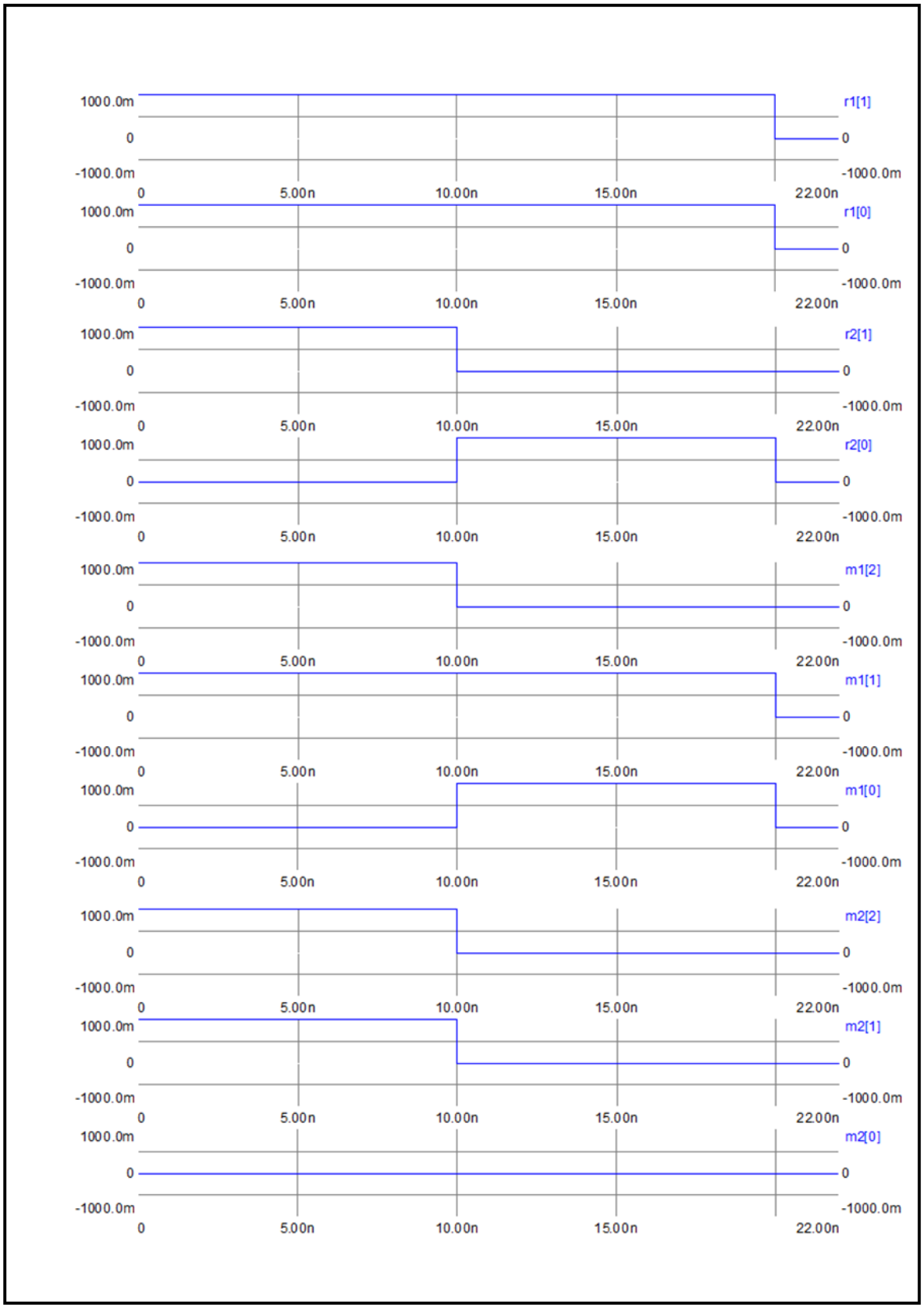


FIGURE 4.28 : Résultats de simulation du circuit de la figure 4.25.

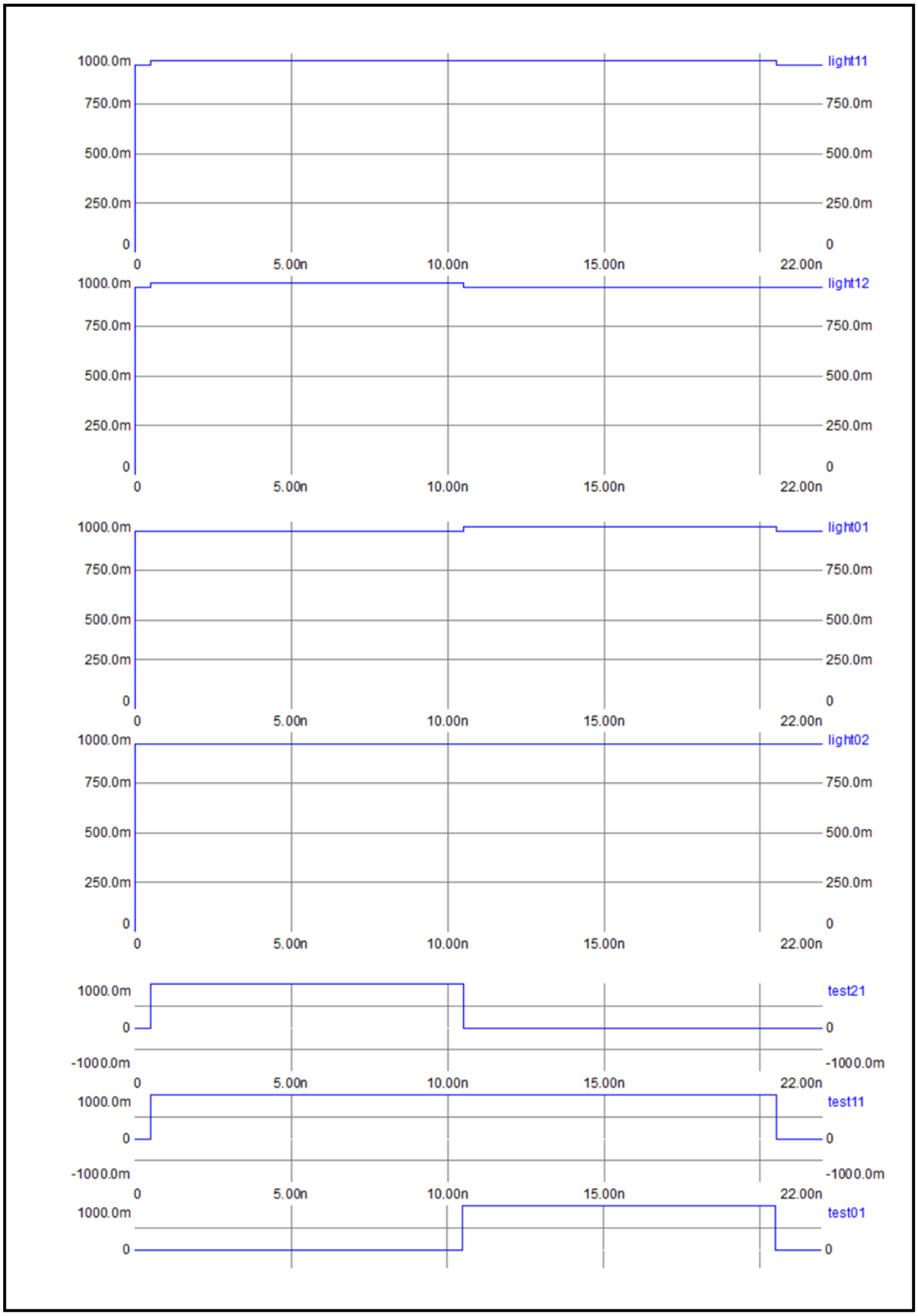


FIGURE 4.28 : Résultats de simulation du circuit de la figure 4.25 (Suite).

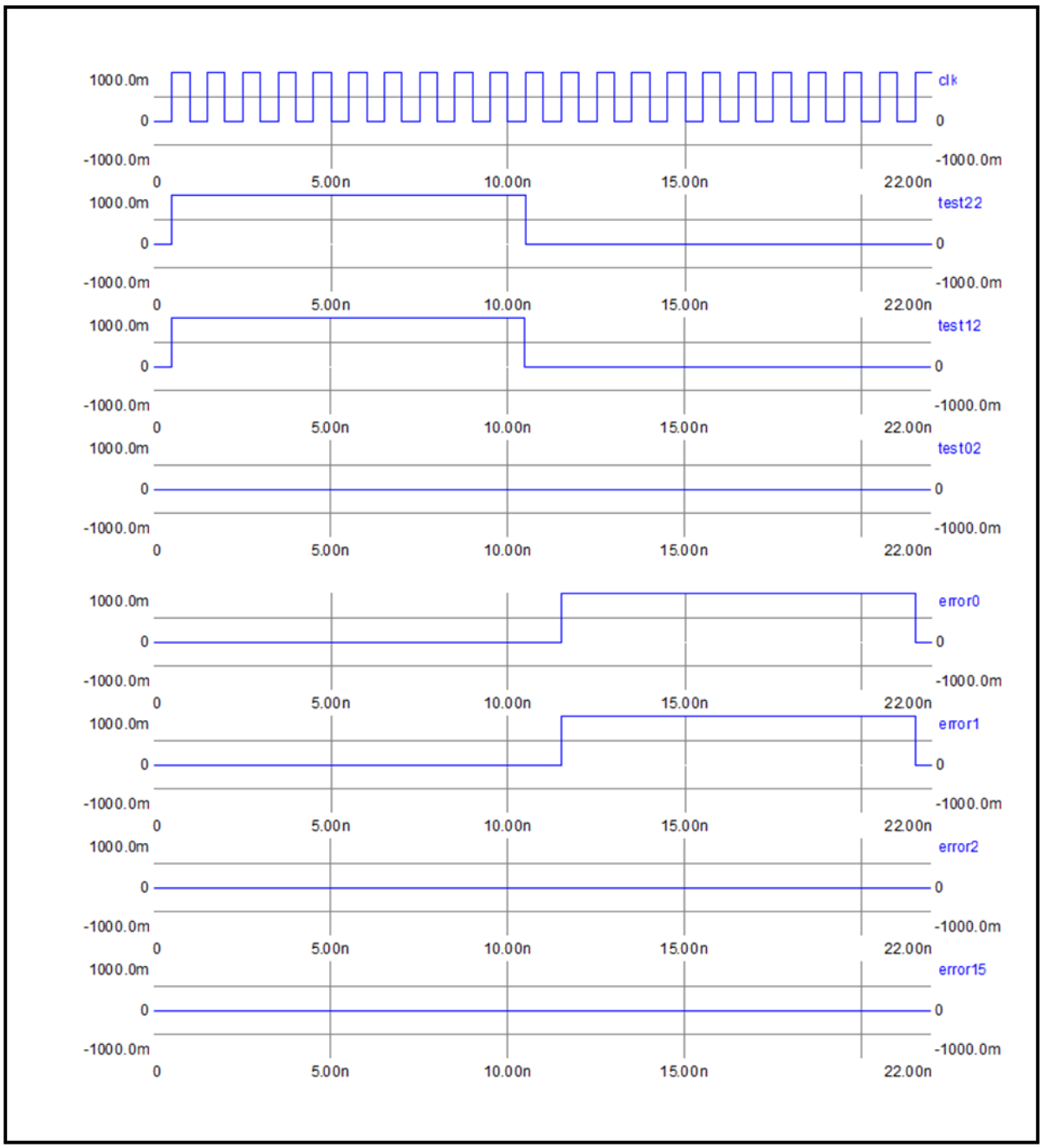


FIGURE 4.28 – Résultats de simulation du circuit de la figure 4.25 (Suite).

4.5 Conclusion

Dans ce chapitre, nous venons de présenter la mise en œuvre de l'approche développée, adaptée à la technique de la duplication physique, ainsi que les résultats de simulation obtenus. Le but était de présenter comment utiliser l'architecture BIST optique, avec ses

nouveaux OBILBO développés, pour assurer non seulement un test hors ligne mais aussi un test en ligne des systèmes embarqués et ainsi de mesurer l'efficacité de notre méthode de test.

La modélisation en langage de description de matériel normalisé VHDL-AMS du système complet de test en ligne, nous a permis de mettre en avant les facultés de ce langage à modéliser des systèmes hétérogènes et ce à plusieurs niveaux différents d'abstraction. Nous avons ainsi pu modéliser au sein d'un même modèle des composantes optiques, électroniques analogiques et électroniques numériques. Au cours des paragraphes précédents, nous avons explicité les méthodologies de modélisation des différents composants de notre système et les moyens qu'offre le VHDL-AMS pour y parvenir. Ceci nous a permis la mise au point d'une stratégie de test qui permet :

1. d'assurer un test en ligne continuel sans génération spécifique des vecteurs de test, d'où un gain substantiel, car les tâches de génération de vecteurs de test sont onéreuses et délicates,
2. d'appliquer des tests de manière concurrente à l'application en cours d'exécution sans ralentir ou perturber le fonctionnement normal du système sous test,
3. de vérifier la conformité des résultats produits en temps réel. En effet, cette stratégie a été validée sur un exemple typique, une puce DSP TMS32010.

Enfin, les résultats de simulation obtenus, ont montré que l'approche proposée offre une détection de faute de collage avec une latence d'erreur très réduite par rapport à la technique de duplication physique, qui est considérée comme l'une des techniques dont la latence d'erreur est faible par rapport à d'autres techniques de test en ligne qui existent dans la littérature. En outre, la technique optique développée peut être également exploitée dans le test hors ligne des circuits, cartes et systèmes supportant la norme BIST, avec une possibilité de réduire les opérations de Scan en une seule opération de lecture des registres OBILBO. Néanmoins, l'implémentation de notre architecture est réalisée au détriment d'un surplus de hardware.

CONCLUSION GÉNÉRALE

Le développement fulgurant des méthodes de conception des circuits intégrés a favorisé l'émergence des systèmes embarqués qui prolifèrent rapidement dans des domaines d'application très variés, allant des composants bon marché aux systèmes de très haute sécurité. Par ailleurs, les systèmes embarqués sont de plus en plus mis en œuvre dans des missions critiques. Appliqué seul, comme unique moyen de détection de fautes, le test hors ligne ne permet pas de garantir les niveaux de fiabilité et de sécurité de ces systèmes qui restent vulnérables aux défauts survenant en cours de fonctionnement. Pour un grand nombre d'applications critiques telles que celles rencontrées dans les transports, les applications spatiales et médicales, la sécurité et la sûreté de fonctionnement sont les deux principaux facteurs pris en compte. La criticité de leur mission justifie donc la nécessité de compléter le test hors ligne par des modules de test en ligne capables d'évaluer l'état d'un système en cours d'exploitation et ainsi de vérifier, de manière continuelle, ses réponses fonctionnelles.

Dans cette thèse, nous avons précisé plusieurs problématiques liées au test en ligne des systèmes intégrés, en particulier celles pour les systèmes embarqués dédiés aux applications critiques. Nous avons ensuite développé une nouvelle approche de test en ligne permettant de répondre à ces problèmes. Cette approche, adaptée à la technique de duplication matérielle, repose sur une architecture BIST (*Built-In Self-test*), approche qui a été proposée et développée afin d'améliorer la testabilité des systèmes sur puce. Dans cette architecture, nous avons suggéré l'introduction de nouveaux registres OBILBO (*Optical Built-In Logic Block Observation*). Ces nouveaux registres OBILBO sont des registres BILBO standards qui intègrent, dans leur structure, des sondes optiques. Ces OBILBO permettent l'émission, à un système distant équipé de capteurs optiques simples et de Convertisseurs Analogiques/Numériques (CAN), des signaux optiques correspondant aux états logiques des données numériques présentes sur les sorties de ces registres. Ceci rend possible la comparaison, en temps réel, des signaux récupérés aux sorties des CAN avec ceux d'un module dupliqué afin de détecter la présence d'éventuels défauts. Pour arriver à cette solution, une proposition a été étudiée et modélisée en utilisant le langage

VHDL-AMS.

L'architecture du test en ligne a finalement était intégrée dans un système sur puce typique, le DSP TMS32010, et les résultats de simulation obtenus sont très satisfaisants. Cette intégration a permis de valider la méthode de test proposée avant de l'utiliser dans de futurs circuits.

Suite au développement de l'architecture dédiée au test en ligne, nous avons également proposé une stratégie de test mettant en œuvre cette architecture. Cette stratégie permet de réaliser des tests concurremment à l'application en cours d'exécution et de vérifier les résultats produits en temps réel, sans toutefois affecter le fonctionnement normal de l'application, ceci grâce aux nouveaux registres OBLIBO considérés. La méthode de test proposée offre ainsi une détection immédiate de fautes de collage simples avec une latence d'erreur d'un seul cycle d'horloge, sans avoir recours à la génération de vecteurs de test : les entrées usuelles du système en exploitation sont, en fait, elles-mêmes les vecteurs de test, ce qui constitue un gain substantiel en matière de temps de test.

Cette approche permet également de réaliser des tests hors ligne, en particulier pour le test de fautes de collage simples. Elle offre, en effet, la possibilité de réaliser le diagnostic de fautes pouvant apparaître sur les registres OBILBO eux-mêmes qui peuvent être la cause de présence de défauts dans le système entier. L'avantage de cette approche est qu'elle engendre la réduction du nombre N d'opérations de décalage série (N étant le nombre de bascules formant le registre OBILBO) nécessaires pour effectuer un scan, à une seule opération de décalage du contenu du registre OBILBO, ce qui élimine ainsi le caractère série du registre qui constitue son inconvénient majeur. Il suffit, pour cela, de choisir d'abord le "*Seed*" (état initial) égal à la séquence 1111... (0000...), décaler ensuite le contenu du registre OBILBO d'une seule position pour tester les fautes de collage-à-0(1), et finalement réaliser la lecture parallèle du registre OBILBO. En d'autres termes, le registre BILBO série est en fait transformé en un registre parallèle grâce aux sondes optiques intégrées dans sa structure.

Dans le même contexte, les fonctions habituelles du test hors ligne des registres BILBO standards sont préservées et aucun de ses quatre modes de fonctionnement n'est modifié. Outre ses fonctionnalités standard, le registre OBILBO offre désormais une signature optique et ainsi une nouvelle possibilité d'analyse à distance.

La force de la technique que nous avons développée est caractérisée par son indépendance des outils CAD et de la structure des systèmes à tester. De plus, notre technique est facile à mettre en œuvre car la grande majorité des circuits actuels intègrent déjà le mécanisme BIST. Les exigences et les besoins de test formulés dans les nouvelles appli-

cations critiques mobiles rendent notre technique très attractive en termes d'accessibilité aux cœurs enfouis du système, en termes de temps de test et particulièrement en termes de possibilités de test en ligne sans contact. Ces aspects sont très appréciés dans de tels systèmes où l'intervention humaine doit être réduite au strict minimum.

Néanmoins, l'architecture développée présente un inconvénient en matière de surcoût en surface nécessaire à son implémentation. Pour y remédier, il serait vraisemblablement intéressant d'exploiter la signature optique des registres OBILBO pour caractériser un module dupliqué virtuel et éviter ainsi la duplication complète très onéreuse du module sous test : le module dupliqué ne serait alors vu qu'à travers ses signatures optiques spécifiques. Il faut également noter que les solutions optiques offrent des moyens de test en ligne viables mais restent tout de même liées à la technologie mise en œuvre, technologie qui s'accompagne éventuellement d'autres problèmes principalement dus aux interférences.

Ce travail de thèse propose donc une avancée pour la mise en œuvre des techniques de tests en ligne optiques dédiés aux nouvelles applications critiques (médical, spatial, transport, etc.) avec une volonté initiale de répondre de manière plus favorable aux besoins du test en ligne pour les systèmes sécurisés.

Dans la suite de nos travaux, nous souhaiterions valider notre approche en construisant un ou plusieurs prototypes matériels et le(s) tester en situation réelle; ceci permettra de valider notre méthode et notre architecture, directement sur silicium.

Une autre perspective pour nos travaux serait l'introduction de cellules *Wrapper* (*"Coquille"*) issues de la norme IEEE 1500 que nous rendrions optiques, ceci à des fins de test en ligne des IP (*Intellectual Property*) embarquées dans un système sur puce. En exploitant le mode échantillonné ("*Capture*") qu'offrent les cellules *Wrapper* standard de la norme IEEE 1500, il deviendrait alors possible d'accéder aux sorties des IP embarqués afin de récupérer leurs réponses via ces cellules *Wrapper* optiques, sans pour autant ralentir ou interrompre le fonctionnement normal du système en cours d'exploitation.

BIBLIOGRAPHIE

[1] Stephan Schulz, Jerzy W. Rozenblit, Klaus J. Buchenrieder, "Multilevel Testing for Design Verification of Embedded Systems", *IEEE Design & Test of Computer*, Vol. 19, N°.2, 2002, pp. 60-69.

[2] Emmanuel SIMEU, "Test et Surveillance Intégrés des Systèmes Embarqués", *Mémoire d'Habilitation à Diriger des Recherches de l'Université Joseph Fourier - Grenoble*, Septembre 2005.

[3] Al-Assad H., Murray.B and Hayes.J, "Online BIST for Embedded systems", *IEEE Design & Test of Computers*, Vol.15, N°.4, 1998, pp. 17-24.

[4] Papa, G., Garbolino, A.-T.T., " Optimal On-Line Built-In Self-Test Structure for System-Reliability Improvement", *IEEE Congress on Evolutionary Computation (CEC)*, pp. 222-229, 2011.

[5] Al-Asaad, H., "Efficient Techniques for Reducing Error Latency in On-line Periodic Built-in Self-Test", *IEEE Instrumentation & Measurement Magazine*, pp. 28-32, 2010.

[6] Nicolaids M. and Zorian, "On-Line Testing for VLSI- A Compendium of Approaches," *Journal of Electronic Testing Theory and Application* , N°12, 1998, pp.7-20.

[7] Johnson, J., Howes, W., Wirthlin, M., McMurtrey, D.L., Caffrey, M., Graham, P., Morgan, K., "Using Duplication with Compare for On-line Error Detection in FPGA-based Designs", *IEEE Aerospace Conference*, pp. 1-11, 2008.

[8] Vitaly Ocheretny, "Self-Checking Arithmetic Logic Unit with Duplicated Outputs", *IEEE 16th International On-Line Testing Symposium (IOLTS)*, pp. 202-203, 2010.

[9] Hunger, M., Hellebrand, S., "The Impact of Manufacturing Defects on the Fault Tolerance of TMR-Systems", *IEEE 25th International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, 2010, pp. 101-108.

[10] Grosso, M., Reorda, M.S., Portela-Garcia, M., Garcia-Valderas, M., Lopez-Ongil, C., Entrena, L., " An On-line Fault Detection Technique Based on Embedded Debug Features", *IEEE 16th International On-Line Testing Symposium (IOLTS)*, pp. 167-172, 2010.

[11] Wen-Ben Jone, Der-Chen Huang, and Sunil R. Das, "An Efficient BIST Method for Non-Traditional Faults of Embedded Memory Arrays", *IEEE Transactions on Instrumentation and Measurement*, Vol. 52, NO. 5, October 2003.

[12] Thacker, H.D., Ogunsola, O.O., Muler, A.V., Meindl, J.D., "Wafer-Testing of Optoelectonic–Gigascale CMOS Integrated Circuits", *IEEE Journal of Selected Topics in Quantum Electronics*, VOL. 17, NO. 3, pp. 659-670, 2011.

[13] J.C. Tsang, J.A. Kash and D.P. Vallett, "Time-resolved optical characterization of electrical activity in integrated circuits", *Proceedings of the IEEE*, Vol. 88,No. 09, pp. 1440-1459, 2000.

[14] D.M. Zhang, M.B. Yi, W. Sun, A.L. Hou, "External electro-optic measurement utilizing poled polymer-based asymmetric Fabry-Perot reflexion film", *Journal of Applied Physics*, 86, No 11, December 1999.

[15] G.E. Bridges, "Vector-voltage scanning force probe for non-contact MMIC measurement", *Electronic Letters*, Vol. 35, pp. 1724-1725, 1999.

[16] P.O. Muller, D. Erasme and al, "New Calibration Method in Electro-optic Probing Due to Wavelength Control and Fabry-Perot Resonance", *IEEE Transactions on Microwave Theory and Techniques*, Vol. 47, pp 308-314, 1999.

[17] K. Yang, P.B. Kateshi and J.F. Whiteker, "Electric field mapping system using an optical-fiber-based electrooptic probe", *Microwave and wireless Components Letters, IEEE*, Vol. 11, Issue 4, April 2001, pp. 164-166.

[18] B. Pannetier, P. Lemaitre-Auger, S. Tedjini, E. Dogheche, D. Remiens, "External Optical Probe for Novel Online Testing Method of Digital Circuits", *URSI-GA 2002*, Netherlands, August 2002.

[19] Aktouf .Chouki," On-line Testing in Continuous Operation of Embedded Systems: Modeling and Performance Evaluation, *Proceedings of the Seventh IEEE International On-Line Testing Workshop (IOLTW'01)*, 2001.

[20] Aktouf .Chouki, Pannetier Benoît, Pierre Lemaître-Auger & Smail Tedjini "On-line Testing of embedded Systems Using Optical Probes: System Modeling and Probing technology", *Proceedings of the Eighth IEEE International On-Line Testing Workshop (IOLTW'02)*, 2002.

[21] Mounir BENABDENBI, "Conception en vue du test de systèmes intégrés sur silicium (SOC)," *Thèse de doctorat de l'université Paris IV*, Septembre 2002.

[22] Semiconductor Industry Association, *International Technology Roadmap for Semiconductors*, 1999.

[23] Régis BOUYSSOU, "Traitements plasmas Post Gravure pour l'intégration des matériaux SiOCH poreux dans les interconnexions en microélectronique", *Thèse de doctorat de l'Université Joseph Fourier - Grenoble*, Décembre 2009.

[24] Alain Dessureault MBA," Les tendances de l'électronique des années 2010 – 2015", *Rapport du Centre d'innovation en microélectronique du Québec*, 2005.

[25] Laurène Babaud, " Développement et optimisation d'un procédé de gravure grille polysilicium pour les nœuds technologiques 45 et 32 nm", *Thèse de doctorat de l'Institut Polytechnique de Grenoble*, Avril 2010.

[26] ITRS International Technology Roadmap for Semiconductor, 2008, *http://www.itrs.net/*

[27] Julien POUGET, "Test des systèmes sur puces: Ordonnancement et exploitation de l'espace des solutions architecturales," *Thèse de doctorat, Université de Montpellier II*, Novembre 2002.

[28] L. Zaourar, J. Alami Chentoufi, Y. Kieffer, A. Waserhole, " Optimisation du partage de blocs BIST pour le test des mémoires d'un circuit intégré ", *ROADEF 2010 – Toulouse*, 2010.

[29] A. Dargelas, " Approche multi- stratégique pour la génération déterministe de vecteurs de test de circuits séquentiels synchrones," *Thèse de Doctorat, LIRMM, Université de Montpellier II*, Octobre 1997.

[30] M. Abramovici, M.A. Breuer, A.D. Friedman, "Digital systems testing and testable design," *IEEE Press, Revised Printing*, 1990.

[31] R. G. Bennetts, "Testable logic circuits," Addission –Wesley Publisher Limited, 1984.

[32] H. Fujuwara, "Logic testing and design for testability," *The MIT Press Cambridge*; March 1986.

[33] B. R Wilkins, "Testing digital circuits: An introduction," V*an Nostrand Reinheld (UK) Co. Ltd* , 1986.

[34] M.L Bushnell and V.D. Agrawal, "Essentials of Electronic Testing for Digital Memory & Mixed signal VLSI Circuits," *Kluwer Academic Publisher*, 2000.

[35] M. J. Howese and V. D Morgan, "Reliability and Degradation – Semiconductor Devices and Circuits," *Wiley-Inter-science*, 1981.

[36] H.Viallon, "Contribution au test intégré : Optimisation des générateurs de vecteurs de test et leur adaptation à la détection de fautes complexes," *Thèse de Doctorat, LIRMM, Université de Montpellier II*, Octobre 1996.

[37] Julien DALMASSO, "Compression de données de test pour architecture de systèmes intégrés basée sur bus ou réseaux et réduction du temps de test", *Thèse de Doctorat de l'Université de Montpellier II*, Octobre 2012.

[38] P. Gelsinger, "Discontinuities Driven by a Billion Connected Machines", *IEEE Design & Test of Computers*, Vol.17, N°.1, 2000, pp.7-15.

[39] Matthieu TUNA, "Auto-test logiciel des systèmes intégrés sur puce (SOC)", *Thèse de Doctorat de l'Université Paris VI*, Juin 2007.

[40] T. Williams and N. Brown, "Defect Level as a Function of Fault Coverage", *IEEE Transactions on Computers*, Vol. C-30, N°.12, 1981, pp. 987–988.

[41] S. Mourad and Y. Zorian, "Principles of Testing Electronic Systems", *John Wiley & Sons, Somerset, NJ*, 2000

[42] Ahcène BOUNCEUR, "Plateforme CAO pour le test de circuits mixtes", *Thèse de Doctorat de l'INPG*, Avril 2007.

[43] H. Fujiwara, T. Shimono, "On the acceleration of test generation algorithms," *IEEE Transaction on Computers*, Vol. C-32, No. 12,pp. 1137-1144, December, 1983.

[44] J. P. Roth, "Diagnosis for automata failures: A calculus and a method," *IBM Journal Research and Development*, Vol. 10, pp. 278-291, July, 1966.

[45] Chouki Aktouf, Hervé Fleury, Chantal Robach, "Inserting Scan at the Behavioral Level," *IEEE Design & Test of Computers*, July-September 2002, pp.2-10.

[46] Hervé BERVILLER, "Amélioration de la sûreté de fonctionnement des systèmes à commandes unidirectionnelles basée sur la conception d'interfaces en ASIC", *Thèse de Doctorat de l'Université de Metz*, Septembre 1998.

[47] M. WLLIAMS, J. ANGEL, " Enhancing testability of LSI circuits via test points and additional logic", *IEEE Transaction On Computers*, Vol.C-22, N°.l, January 1973.

[48] S. FUNATSU, N. WAKATSUKI, T. ARIMA, "Test generation in Japan", *Design Automation Conference*, June 1975.

[49] H. FUJIIVARA, "Logic testing and design for testability", *MIT Press series in computers systems, Cambridge,* 1985.

[50] E.B. EICHELBERGER, T.W. WLLIAMS, "A logic design structure for LSI testability", *Design Automation Conference*, June 1977.

[51] Bennets.RG and Osseyran. A, "IEEE Standard1149.1-1990 on Boundary scan: History Survey, and Current Status," *Journal of Electronic Testing Theory and Application,* Vol.2 N°.1, 1991,

[52] IEEE Standard Board. IEEE Std 1149.1-1990, "Standard test access port and boundary scan architecture", *345 East 47th Street, New York, NY* 10017-2394, 1990.

[53] H. Bleeker, P. van den Eijden, F. de Jong, "Boundary Scan test: a practical approach," *Kluwer Academic Press*, 1993.

[54] C.M Mounder, R.E Tulos, "An introduction to the Boundary Scan standard ANSI/IEEE std 1149.1," I*EEE Journal of Electronic*, 1991

[55] V.D. Agrawal, K.T. Cheng, D.D. Johnson and T. Lin, "Designing Circuits with Partial Scan," *IEEE Design & Test of Computers*, pp. 8-15, April 1988.

[56] Mark F. Lefebvre, "Functional Test and Diagnosis: A Proposed JTAG Sample Mode Scan Tester," *1990 International Test Conference*, pp. 294-302.

[57] Vishawni. D. Agrawal Charles R. Kime, Kewal K. Saluja, AT& Bell Laboratories, "A tutorial on Built-In-Self-Test, Part.1: principles," *IEEE Transaction on Computers,* March, 1993, pp.73-82.

[58] Vishawni. D. Agrawal, Charles R. Kime, Kewal K. Saluja, AT& Bell Laboratories, "A tutorial on Built-In-Self-Test, Part.2: applications," *IEEE Transaction on computers,* June 1993, pp.66-77.

[59] Zhe Zho Bahram Pouya, Nur A. Touba, "BETSY: Synthesizing Circuits for a specific BIST Environment," *ITC 98*, 1998, pp.144-153.

[60] Han Bin Kim, Takeshi Takahshi, Dong Sam Ha, "Test session oriental Built-In-Self-Testable data path systems," *ITC 98*, 1998, pp.154-163.

[61] A. Benso, S. Chiusano, P. Prinette, Y. Zorian, "HD-BIST: Hierarchical Distributed BSIT Architecture dir SOC," *TECS 98*, 1998, pp.241-245.

[62] Xuan-Tu TRAN," Méthode de Test et Conception en Vue du Test pour les Réseaux sur Puce Asynchrones : Application au Réseau ANOC", *Thèse de doctorat de l'Institut Polytechnique de Grenoble*, Février 2008.

[63] Y. Zorian, "Test Requirements for Embedded Core-Based Systems and IEEE P-1500", *Proceedings of the IEEE International Test Conference (ITC)*, Washington, DC, USA, November 1997. pp. 191-199.

[64] A.L. Crouch, "Design-for-Test for Digital IC's and Embedded Core Systems", *Printice Hall,* 1999.

[65] E.J. Marinissen, R. Arendsen, G. Bos, H. Dingemanse, M. Lousberg, and C. Wouters, "A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores", *Proceedings of the IEEE International Test Conference (ITC)*, October 1998, pp. 284-293.

[66] R.K. Gupta and Y. Zorian, "Introducing Core-Based System Design", *IEEE Design & Test of Computers,* Vol.14, N°4, pp.15-25, October-November 1997.

[67] Y. Zorian, "Test Embedded Core-Based System Chips", *Proceedings of the IEEE International Test Conference (ITC)*, Washington, DC, USA, October 1998, pp. 130-140.

[68] Régis POIRIER, "Compression de données pour le test des circuits intégrés", *Thèse de Doctorat de Montpellier II*, Novembre 2004.

[69] IEEE 1500 Group, *IEEE 1500 Standard for Embedded Core Test. http://grouper.ieee.org/groups/1500/*

[70] Ahmad ABDELHAY, "Test en ligne des systèmes digitaux linéaires", *Thèse de doctorat de l'INPG de Grenoble*, Avril 2001.

[71] Oikonomakos. Petros, Zwolinski Mark,"Using high-level synthesis to implement on-line testability" *IEEE Real- Time Embedded System Workshop*, Dec. 2001, pp.1-4.

[72] M. Nicolaidis, "On- line testing for VLSI : state of the art and trends," *Integration the VLSI Journal*, Vol. 26, No. 1-2, December 1998, pp. 197-209.

[73] M. Nicolaidis, "Design for Soft-Error Robustness To Rescue Deep Submicron Scaling," *IEEE International Test Conference*, 1998, pp. 11-20.

[74] M. Nicolaidis, L. Anghel, "Concurrent Checking for VLSI," *Microelectronic Engineering*, Vol. 49, No. 1-2, November 1999, pp. 139-156

[75]] C. Zeng, N. Saxena, E.J. McCluskey, "Finite State Machine Synthesis with Concurrent Error Detection," *IEEE International Test Conference*, 1999, pp.672-679.

[76] Samuel N. Hamilton and Alex Orailoglu, "On-Line Test for Fault-Secure Identification," *IEEE Transaction On VLSI Systems*, Vol.8, N°4, August 2002, pp. 446-452.

[77] K.K. Saluja, R. Sharma, and C.R. Kime, "A Concurrent Testing Technique for Digital Circuits," *IEEE Trans. Computer-Aided Design*, Vol. 7, No. 12, Dec. 1988, pp. 1250-1259.

[78] M. Nicolaidis, "Theory of Transparent BIST for RAMs," *IEEE Trans. Computers*, Vol. 45, No. 10, Oct. 1996, pp. 1141-1156.

[79] A. Mahmood and E. McCluskey, "Concurrent Error Detection Using Watchdog Processors—A Survey," *IEEE Trans. Computers*, Vol. 37, No. 2, Feb. 1988, pp. 160-174.

[80] B.W. Johnson, "Design and Analysis of Fault Tolerant Digital Systems,"*Addison-Wesley, Reading, Mass.*, 1989.

[81] A. Bogliolo, M. Favalli, M. Damiani, "Enabling Testability of Fault- Tolerant Circuits by means of IDDQ-checkable voters," *IEEE Transactions on VLSI*, Vol. 8, No. 4, August 2000, p. 415-418.

[82] K. Arabi and Kaminska, "Built in Temperature and Current Sensors for On-Line Oscillations-testing," *1996 IEEE International On-Line Testing Workshop*, Saint-Jean de Luz, Biarritz, Farnce, July 1996.

[83] W. C. Carter and P. R. Schneider, "Design of Dynamically Checker Computers," *Proc. 4th Congress IFIP, Edinburgh, Scotland*, Aug. 5-10, 1986, Vol. 2, pp. 878-883.

[84] D. A. Anderson, "Design of Self-Checking Digital Networks Using Coding Techniques Coordinates. Sciences Laboratry," *Report R/527, University of Illinois, Urbania*, Sept. 1971.

[85] J. E Smith and G. Metze, "Strongly Fault Secure Logic Networks," *IEEE Trans. On Computers*, Vol. C-27, N°.6, June 1978, pp. 71-80.

[86] E. J. McCluskey and F.W. Clegg, "Fault Equivalent in Combinational Logic Networks," *IEEE Trans. On Computers*, Vol. C-20, Nov. 1971. pp. 1286-1293.

[87] C. Aktouf, G. Al-Hayek, C. Robach, "Concurrent testing of VLSI digital signal processors using mutation based testing," *IEEE International Workshop on Defect and Fault Tolerance in VLSI Systems*, 1997, pp. 94-99.

[88] M. Lobetti-Bodoni, A. Pricco, A. Benso, S. Chiusano, P. Prinetto, "An on-line BISTed SRAM IP core," *IEEE International Test Conference*, 1999, pp. 993-1000.

[89] R. Leveugle, "Automatic Modifications of High Level VHDL Descriptions for Fault Detection or Tolerance," *Design Automation and Test in Europe (DATE),* 2002, pp. 837 - 841.

[90] C. Bolchini, R. Montandon, F. Salice, D. Sciuto, "Design of VHDL-based totally self-checking finite-state machine and data-path descriptions," *IEEE Transactions on VLSI*, Vol. 8, No. 1, February 2000, pp. 98-103.

[91] S. Tarnick, A.P. Stroele, "Embedded Self-testing checkers for low-cost arithmetic codes," *IEEE International Test Conference*, 1998, pp. 544-523.

[92] M. Nicolaidis, R.O. Duarte, S. Manich, J. Figueras, "Fault-Secure Parity Prediction Arithmetic Operators," *IEEE Design & Test of Computers*, Vol. 14, No. 2, April-June 1997, pp. 60-71.

[93] M. Nicolaidis, R.O. Duarte, "Design of Fault-Secure Parity-Prediction Booth Multipliers," *Design Automation and Test in Europe*, 1998, pp. 7-14.

[94] S. Tarnick, "Controllable Self-Checking Checkers for Conditional Concurrent Checking," *IEEE Transactions on CAD*, Vol. 14, No. 5, May 1995, pp. 547-553.

[95] A.P. Stroele, S. Tarnick, "Programmable embedded self-testing checkers for all unidirectional error-detecting codes," *IEEE VLSI Test Symposium*, 1999, pp.361-369.

[96] W.-F. Chang, C.-W. Wu, "TSC Berger-code Checker Design for 2r-1-Bit Information," *Journal of Information Science and Engineering*, Vol. 15, No. 3, 1999, pp. 429-441.

[97] S. Mitra, E.J. McCluskey, "Which concurrent error detection scheme to choose?," *IEEE International Test Conference*, 2000, pp. 985-994.

[98] S. Mitra, N.R. Saxena, E.J. McCluskey, "Fault Escapes in Duplex Systems," *IEEE VLSI Test Symposium*, 2000, pp. 453-458.

[99] J. Yeandel, D. Thulborn, S. Jones, "An on-line testable UART implemented using IFIS," *IEEE VLSI Test Symposium*, 1997, pp. 344-349.

[100] J. Yeandel, D. Thulborn, S. Jones, "IFIS: an online test methodology," *IEE Proceedings– Circuits, Devices and Systems*, Vol. 145, No. 1, February 1998, pp.1-6.

[101] J. Yeandel, D. Thulborn, S. Jones, "The design and implementation of an on-line testable UART," *Journal of Electronic Testing – Theory and Applications*, Vol. 12, No. 3, June 1998, pp. 187-198.

[102] A. Orailoglu, R. Karri, "Automatic Synthesis of Self-Recovering VLSI Systems," *IEEE Transactions on Computers*, Vol. 45, No. 2, February 1996, pp. 131-142.

[103] N.R. Saxena, S. Fernandez-Gomez, W.-J. Huang, S. Mitra, S.-Y. Yu, E.J. McCluskey, "Dependable Computing and Online Testing in Adaptive and Configurable Systems," *IEEE Design & Test of Computers*, Vol. 17, No. 1, January- March 2000, pp. 29-41.

[104] Laung-Terng Wang, Cheng-Wen Wu & Xiaoqing Wen, "VLSI Test Principle and Architectures", *Elsevier*, 2006.

[105] Charles E. Stroud "A Designer's Guide to Built-in Self-Test", *Kluwer Academic Publishers*, 2002.

[106] A.C. Williams, A.D. Brown, M. Zwolinski, "In-line test of synthesized systems exploiting latency analysis," *IEE Proceedings: Computers and Digital Techniques*, Vol. 147, No. 1, 2000, pp. 33-41.

[107] Nicolaidis, M., "A unified built-in-test scheme: UBIST", *Eighteenth International Symposium on Fault-Tolerant Computing*, 1988, pp. 157 – 163.

[108] Nicolaidis, M., “Efficient UBIST implementation for microprocessor sequencing parts”, *International Test Conference*, 1990, pp. 316-326.

[109] Serge NORAZ, “Application des circuits intégrés auto-testables à la sureté de fonctionnement des systèmes", *Thèse de doctorat de L'institut Polytechnique de Grenoble*, Novembre 1988.

[110] Ricardo Reis, Marcelo Lubaszewski, Jochen A.G. Jess, “Design of Systems on a Chip: Design and Test”, *Springer*, 2006.

[111] Paul Sabatier, "Technologie de pointe ou du quotidien : les systèmes embarqués sont au cœur d'applications sensibles", *Magazine Scientifique,* N° 11, Novembre 2007, pp 1-28.

[112] J.C. Tsang, J.A. Kash and D.P. Vallett, “Time-resolved optical characterization of electrical activity in integrated circuits”, *Proceedings of the IEEE*, Vol. 88,No. 09, pp. 1440-1459, 2000.

[113] E. J. McCluskey, “Logic Design Principles: With Emphasis on Testable Semiconductor Circuits”, *Prentice Hall, Englewood Cliffs, NJ*, 1986.

[114] V.D. Agrawal, C.R. Kime and K.K. Saluja, “A tutorial on BIST : part1” , *IEEE Design & Test of Computers*, pp. 73-82, March 1993.

[115] Abdelhakim Latoui & Farid Djahli, “An Optical BILBO for On-Line Testing of Embedded Systems”, *IEEE Design & Test of Computers*, Vol. PP, N°: 99, March 2013, pp. 1-12.

[116] Fabien Clermidy, “Amélioration de la fiabilité des calculateurs parallèle SIMD par test et tolérance aux fautes structurelle”, *Thèse de doctorat de L'INPG*, Décembre 1999.

[117] Régis LEVEUGLE, "Test des circuits intégrés numériques - Conception orientée testabilité", *Techniques de l'Ingénieur*, Référence E2461, 10 Août 2002.

[118] Michel MOREAU, "Test des VLSI numériques", *Techniques de l'Ingénieur*, Référence E2420, 10 Juin 2002.

[119] S. Golomb, "Shift Register Sequences”, *Launa Hills, California: Aegean Park, Press*, 1982.

[120] S. Z. Hassan and E. J. McCluskey, “Increased fault coverage through multiple signatures”, *Digest of Papers, Fault-Tolerant Computing Symp.*, June 1984, pp. 354–359.

[121] T. W. Williams, W. Daehn, M. Gruetzner, and C. W. Starke, “Aliasing errors in signature analysis registers”, *IEEE Design & Test of Computers*, Vol. 4, N°: 2, 1987, pp. 39–45.

[122] B. Koenmann, J Mucha, and G Zwiehoff, “Built-In Logic. Block Observation techniques”, *Proc. IEEE Test Conference, Cherry Hill, New Jersey*, 1979, pp. 37-41.

[123] Salvador Mir, “Technique de l’Informatique et de la Microélectronique pour l’Architecture des ordinateurs", *Habilitation à diriger des recherches, Laboratoire TIMA*, 2005.

[124] B. Koenemann, J. Mucha and G. Zwiehoff, “Built-In Test for Complex Digital Integrated Circuits”, *IEEE J. Solid State Circuits*, Vol. 15, No. 3, June 1980, pp. 315-318.

[125] Peter J. Ashenden & Jim Lewis, “The Designer’s Guide to VHDL, Third Edition”, *Morgan Kaufmann*, May 29, 2008.

[126] J. Weeber et M. Meaudre, “ VHDL Du langage au circuit, du circuit au langage ”, *Edition Masson*, 1997.

[127] “1076.1-1999 IEEE Standard VHDL Analog and Mixed-Signal Extensions”, *Reference Manual, IEEE Press*, ISBN 0-7381-1640-8, 1999.

[128] Hervé Yannick, " VHDL-AMS Applications et enjeux industriels ", *Dunod, Paris,* ISBN 2-10-005888-6, 2002.

[129] David GUIHAL, "Modélisation en langage VHDL-AMS des systèmes pluridisciplinaires", *Thèse de doctorat de l’Université Toulouse III*, Mai 2007.

[130] Frey.P, Shanmugasudarm.V, Mayiladuthrai. R, Chandrashekar.C and Carter.H, “Seams: simulation environment for VHDL-AMS”, *Proceeding of the 1998 Winter Simulation conference*. 1998, pp.539-545.

[131] A. Vachoux, “Modélisation de Systèmes Intégrés Analogiques et Mixtes : Introduction à VHDL-AMS", *Notes de cours à option 2ème cycle*, Version 2002.

[132] Hamimi Chemali, Abdelhakim Latoui, Chouki Aktouf, “An Optical Boundary Scan Cell for On Line Testing of Embedded Systems”. *Proc. of Embedded Systems and Applications*, 2003, pp. 155-160.

ANNEXE A : CODE VHDL DU CIRCUIT DE LA FIGURE 4.7

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
--USE IEEE.STD_LOGIC_ARITH.ALL;
ENTITY COMPT4 IS
GENERIC(N:INTEGER:=3);
PORT(CLK,RESET:IN STD_LOGIC;
COUNT: OUT STD_LOGIC_VECTOR(N DOWNTO 0));
END COMPT4;
ARCHITECTURE ARCH OF COMPT4 IS
SIGNAL X:STD_LOGIC_VECTOR(N DOWNTO 0);
BEGIN
PROCESS(RESET,CLK)
BEGIN
IF RESET='1' THEN X <= (OTHERS => '0');
ELSIF (CLK'EVENT AND CLK='1') THEN
X <= X + 1;
END IF;
END PROCESS;
COUNT <= X;
END ARCH;
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
--USE IEEE.STD_LOGIC_ARITH.ALL;
ENTITY DECOMPT4 IS
GENERIC(N:INTEGER:=3);
```

```
PORT(CLK,RESET:IN STD_LOGIC;
COUNT: OUT STD_LOGIC_VECTOR(N DOWNTO 0));
END DECOMPT4;
ARCHITECTURE ARCH OF DECOMPT4 IS
SIGNAL X:STD_LOGIC_VECTOR(N DOWNTO 0):="1111";
BEGIN
PROCESS(RESET,CLK)
BEGIN
IF RESET='1' THEN X <= (OTHERS => '1');
ELSIF (CLK'EVENT AND CLK='1') THEN
X <= X - 1;
END IF;
END PROCESS;
COUNT <= X;
END ARCH;
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY ADDER1 IS
PORT(A1,B1,CIN1:IN STD_LOGIC;SUM1,CARRY1:OUT STD_LOGIC);
END ADDER1;
ARCHITECTURE ARCH OF ADDER1 IS
BEGIN
SUM1<=(A1 XOR B1 XOR CIN1);
CARRY1<=(A1 AND B1) OR (A1 AND CIN1) OR (B1 AND CIN1);
END ARCH;
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY ADDER2 IS
PORT(A2,B2,CIN2:IN STD_LOGIC;SUM2,CARRY2:OUT STD_LOGIC);
END ADDER2;
ARCHITECTURE ARCH OF ADDER2 IS
BEGIN
SUM2<=(A2 XOR B2 XOR CIN2);
CARRY2<=(A2 AND B2) OR ('0');
END ARCH;
LIBRARY IEEE;
```

```
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY BILBO_INT IS
PORT(
RESET : IN STD_LOGIC;
CLK : IN STD_LOGIC;
SCAN_IN : IN STD_LOGIC;
SCAN_OUT : OUT STD_LOGIC;
B : IN STD_LOGIC_VECTOR(2 DOWNTO 1);
--Z : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
Z1,Z2,Z3,Z0:IN STD_LOGIC;
Q : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) );
END BILBO_INT ;
ARCHITECTURE ARCH OF BILBO_INT IS
SIGNAL FF : STD_LOGIC_VECTOR(3 DOWNTO 0);
--SIGNAL Y : BIT_VECTOR(3 DOWNTO 0);
SIGNAL Y0,Y1,Y2,Y3:STD_LOGIC;
BEGIN
PROCESS(CLK)
BEGIN
IF RESET = '1' THEN
FF <= "0000"; --SEED
ELSIF CLK'EVENT AND CLK = '1' THEN
CASE B IS
WHEN "00" => --SHIFT MODE
FF(0) <= SCAN_IN;
FF(1) <= FF(0);
FF(2) <= FF(1);
FF(3) <= FF(2);
WHEN "01" => --MISR MODE
FF(0) <= Y0;
FF(1) <= Y1;
FF(2) <= Y2;
FF(3) <= Y3;
WHEN "10" => --RESET MODE
FF <= "0000";
WHEN "11" => --PARALLEL LOAD MODE
```

```
FF(0) <= Z0;
FF(1) <= Z1;
FF(2) <= Z2;
FF(3) <= Z3;
WHEN OTHERS => --SEED
FF <= "0000";
END CASE;
END IF;
END PROCESS;
Q <= FF;
SCAN_OUT <= FF(3);
Y0<= Z0 XOR FF(3);
Y1<= FF(0) XOR Z1 XOR FF(3);
Y2<= FF(1) XOR Z2;
Y3<=FF(2) XOR Z3;
END ARCH;
--SECOND VERSION OF BILBO
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY BILBO IS
PORT(
RESET : IN STD_LOGIC;
CLK : IN STD_LOGIC;
SCAN_IN : IN STD_LOGIC;
SCAN_OUT : OUT STD_LOGIC;
B : IN STD_LOGIC_VECTOR(2 DOWNTO 1);
Z : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
Q : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) );
END BILBO;
ARCHITECTURE ARCH OF BILBO IS
SIGNAL FF : STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL Y : STD_LOGIC_VECTOR(2 DOWNTO 0);
BEGIN
PROCESS(CLK)
BEGIN
IF RESET = '1' THEN
```

```
FF <= "0001"; --SEED
ELSIF CLK'EVENT AND CLK = '1' THEN
CASE B IS
WHEN "00" => --SHIFT MODE
FF(0) <= SCAN_IN;
FF(1) <= FF(0);
FF(2) <= FF(1);
FF(3) <= FF(2);
WHEN "01" => --MISR MODE
FF(0) <= Y(2) XOR Z(0);
FF(1) <= FF(0);
FF(2) <= FF(1);
FF(3) <= FF(2);
WHEN "10" => --RESET MODE
FF <= "0000";
WHEN "11" => --PARALLEL LOAD MODE
FF(0) <= Z(0);
FF(1) <= Z(1);
FF(2) <= Z(2);
FF(3) <= Z(3);
WHEN OTHERS => --SEED
FF <= "0000";
END CASE;
END IF;
END PROCESS;
Q <= FF;
SCAN_OUT <= FF(3);
Y(2)<=Y(1) XOR Z(1)XOR FF(0) ;
Y(1)<=Y(0) XOR Z(2) ;
Y(0) <= FF(3) XOR Z(3);
END ARCH;
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY COMPARATOR IS
PORT(E1,E2:IN BIT_VECTOR(3 DOWNTO 0);
```

```
E3:IN BIT_VECTOR(3 DOWNTO 0);
ERROR:OUT STD_LOGIC);
END COMPARATOR;
ARCHITECTURE ARCH OF COMPARATOR IS
BEGIN
PROCESS(E1,E2,E3)
BEGIN
IF E3="1000" THEN
IF E1 /=E2 THEN ERROR <='1';
ELSE ERROR <='0';
END IF;
ELSE ERROR <='Z';
END IF;
END PROCESS;
END ARCH;
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
PACKAGE MY_PACK IS
COMPONENT BILBO IS
PORT(
RESET : IN STD_LOGIC;
CLK : IN STD_LOGIC;
SCAN_IN : IN STD_LOGIC;
SCAN_OUT : OUT STD_LOGIC;
B : IN STD_LOGIC_VECTOR(2 DOWNTO 1);
Z : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
Q : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) );
END COMPONENT BILBO;
COMPONENT ADDER1 IS
PORT(A1,B1,CIN1:IN STD_LOGIC;SUM1,CARRY1:OUT STD_LOGIC);
END COMPONENT ADDER1;
COMPONENT ADDER2 IS
PORT(A2,B2,CIN2:IN STD_LOGIC;SUM2,CARRY2:OUT STD_LOGIC);
END COMPONENT ADDER2;
COMPONENT COMPARATOR IS
```

```
PORT(E1,E2:IN BIT_VECTOR(3 DOWNTO 0);
E3:IN STD_LOGIC_VECTOR(3 DOWNTO 0);
ERROR:OUT STD_LOGIC);
END COMPONENT COMPARATOR;
COMPONENT BILBO_INT IS
PORT(
RESET : IN STD_LOGIC;
CLK : IN STD_LOGIC;
SCAN_IN : IN STD_LOGIC;
SCAN_OUT : OUT STD_LOGIC;
B : IN STD_LOGIC_VECTOR(2 DOWNTO 1);
--Z : IN BIT_VECTOR(3 DOWNTO 0);
Z1,Z2,Z3,Z0:IN STD_LOGIC;
Q : OUT BIT_VECTOR(3 DOWNTO 0) );
END COMPONENT BILBO_INT ;
COMPONENT COMPT4 IS
GENERIC(N:INTEGER:=3);
PORT(CLK,RESET:IN STD_LOGIC;
COUNT: OUT STD_LOGIC_VECTOR(N DOWNTO 0));
END COMPONENT COMPT4;
COMPONENT DECOMPT4 IS
GENERIC(N:INTEGER:=3);
PORT(CLK,RESET:IN STD_LOGIC;
COUNT: OUT STD_LOGIC_VECTOR(N DOWNTO 0));
END COMPONENT DECOMPT4;
END PACKAGE MY_PACK;
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE WORK.MY_PACK.ALL;
ENTITY PROJET_BILBO_V02 IS
PORT(
RESET : IN STD_LOGIC;
CLK : IN STD_LOGIC;
SCAN_IN1 :  IN STD_LOGIC;
SCAN_OUT1 :  OUT STD_LOGIC;
```

```
SCAN_IN2 :  IN STD_LOGIC;
SCAN_OUT2 :  OUT STD_LOGIC;
CAR1,CAR2:OUT STD_LOGIC;
SUM1,SUM2:OUT STD_LOGIC;
--B1B2 :  IN STD_LOGIC_VECTOR(2 DOWNTO 1);
B01,B02 :  IN STD_LOGIC;
--Z : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
Q1 :  BUFFER BIT_VECTOR(3 DOWNTO 0);
Q2 :  BUFFER BIT_VECTOR(3 DOWNTO 0);
COMPT1 :OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
--COMPT2 :OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
ERROR:OUT STD_LOGIC);
END PROJET_BILBO_V02 ;
ARCHITECTURE STRUCT OF PROJET_BILBO_V02 IS
SIGNAL Z1 :  STD_LOGIC_VECTOR(3 DOWNTO 0):="0000";
SIGNAL Z2 :  STD_LOGIC_VECTOR(3 DOWNTO 0):="0000";
SIGNAL COUNT1:  STD_LOGIC_VECTOR(3 DOWNTO 0);
--MODIFICATION
SIGNAL A1,B1,CIN1,C1:  STD_LOGIC;
SIGNAL S1:STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL A2,B2,CIN2,C2:  STD_LOGIC;
SIGNAL S2:STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL B1B2 :  STD_LOGIC_VECTOR(2 DOWNTO 1);
BEGIN
B1B2(2)<= B02; B1B2(1)<=B01;
Z1(3) <= '0';
Z1(2) <= '0';
Z2(3) <= '0';
Z2(2) <= '0';
BILBO1:BILBO_INT PORT MAP(RESET=>RESET,
CLK=>CLK,
SCAN_IN=>SCAN_IN1,
SCAN_OUT=>SCAN_OUT1,
B=>B1B2,
Z0=> Z1(0),Z1=>Z1(1),Z2=>Z1(2),Z3=>Z1(3),
Q=>Q1);
```

```
BILBO2:BILBO_INT PORT MAP(RESET=>RESET,
CLK=>CLK,
SCAN_IN=>SCAN_IN2,
SCAN_OUT=>SCAN_OUT2,
B=>B1B2,
Z0=> Z2(0),Z1=>Z2(1),Z2=>Z2(2),Z3=>Z2(3),
Q=>Q2);
ADD1:  ADDER1 PORT MAP (A1=>A1,B1=>B1,CIN1=>CIN1,SUM1=>Z1(0),CARRY1=>Z1(1));
ADD2:  ADDER2 PORT MAP (A2=>A2,B2=>B2,CIN2=>CIN2,SUM2=>Z2(0),CARRY2=>Z2(1));
--ADD2:  ADDER1 PORT MAP (A1=>A,B1=>B,CIN1=>CIN,SUM1=>Z2(0),CARRY1=>Z2(1));
--PRG
PRG1:COMPT4 GENERIC MAP(N=>3) PORT MAP (CLK=>CLK,RESET=>RESET,COUNT=>S1);
--PRG1:DECOMPT4 GENERIC MAP(N=>3) PORT MAP (CLK=>CLK,RESET=>RESET,COUNT=>S1);
PRG2:COMPT4 GENERIC MAP(N=>3) PORT MAP (CLK=>CLK,RESET=>RESET,COUNT=>S2);
--PRG2:DECOMPT4 GENERIC MAP(N=>3) PORT MAP (CLK=>CLK,RESET=>RESET,COUNT=>S2);
A1<=S1(2); B1<=S1(1);CIN1<=S1(0);C1<=S1(3);
A2<=S2(2); B2<=S2(1);CIN2<=S2(0);C2<=S2(3);
--COMPT2 <=S2;
COMPT1 <=S1;
CMP:COMPT4 GENERIC MAP(N=>3) PORT MAP (CLK=>CLK,RESET=>RESET,COUNT=>COUNT1);
COMP: COMPARATOR PORT MAP(E1=>Q1,E2=>Q2,E3=>COUNT1,ERROR=>ERROR);
SUM1 <=Z1(0);
CAR1 <=Z1(1);
SUM2 <=Z2(0);
CAR2 <=Z2(1);
END STRUCT;
```

ANNEXE B : CODE VHDL-AMS DE L'APPLICATION

```
LIBRARY DISCIPLINES,IEEE;
USE DISCIPLINES.ELECTROMAGNETIC_SYSTEM.ALL;
USE IEEE.MATH_REAL.ALL;
ENTITY OP_PROBE IS
PORT (SIGNAL SBIN: INOUT BIT ;
QUANTITY LIGHT : OUT REAL);
END OP_PROBE;
ARCHITECTURE ARCH OF OP_PROBE IS
QUANTITY U_IN : REAL ;
QUANTITY EZ, N_EZ, PHI : REAL ;
CONSTANT N0:  REAL := 2.415 ; CONSTANT LAMBDA : REAL := 0.63E-6 ;
CONSTANT E :REAL := 120.0E-9 ; CONSTANT R13 :  REAL := 55.0E-12 ;
CONSTANT IMAX : REAL := 0.50 ; CONSTANT M : REAL := 0.95 ;
BEGIN
IF (SBIN='1') USE U_IN == 5.0 ;
ELSE U_IN == 0.0 ; END USE ;
EZ == U_IN/E ; N_EZ == N0 -( 0.5* R13*EZ*N0**3 ) ;
PHI == ((2.0*3.14*E*N_EZ)/LAMBDA);
LIGHT == IMAX *(1.0-((1.0/M*SIN(2.0*(2.0*PHI)))));
END ARCH;
LIBRARY DISCIPLINES,IEEE;
USE DISCIPLINES.ELECTROMAGNETIC_SYSTEM.ALL;
USE IEEE.MATH_REAL.ALL;
ENTITY PIN_DIODE IS
PORT (TERMINAL N1,N2 :ELECTRICAL ;
QUANTITY LIGHT : IN REAL);
END PIN_DIODE;
```

```
ARCHITECTURE ARCH OF PIN_DIODE IS
QUANTITY VD ACROSS IP,IR,IC THROUGH N1 TO N2;
CONSTANT CD : REAL := 1.0E-15;
CONSTANT R : REAL := 1.0E3;
CONSTANT S : REAL := 0.13 ;
BEGIN
IR == VD/R;
IC == CD*VD'DOT;
IP == - S*LIGHT;
END ARCH;
LIBRARY DISCIPLINES,IEEE;
USE DISCIPLINES.ELECTROMAGNETIC_SYSTEM.ALL;
USE IEEE.MATH_REAL.ALL;
ENTITY ADC IS
GENERIC (UL: REAL:=125.1E-3; UH:REAL:=125.1E-3);
PORT (TERMINAL N1, N2:  ELECTRICAL;
SIGNAL DOUT: OUT BIT:='0');
END ENTITY ADC;
ARCHITECTURE ARCH OF ADC IS
QUANTITY UIN ACROSS I THROUGH N1 TO N2;
BEGIN
UIN==I;
BREAK DOUT=>'1' WHEN UIN'ABOVE(UL);
BREAK DOUT=>'0' WHEN NOT UIN'ABOVE(UH);
END ARCHITECTURE ARCH;
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY BILBO IS
PORT(
RESET : INOUT BIT;
CLK : INOUT BIT;
SCAN_IN : INOUT BIT;
SCAN_OUT : OUT BIT;
B : INOUT BIT_VECTOR(2 DOWNTO 1);
Z : INOUT BIT_VECTOR(3 DOWNTO 0);
Q : INOUT BIT_VECTOR(3 DOWNTO 0)
```

```
);
END BILBO;
ARCHITECTURE ARCH OF BILBO IS
SIGNAL FF : BIT_VECTOR(3 DOWNTO 0);
SIGNAL Y : BIT_VECTOR(2 DOWNTO 0);
BEGIN
PROCESS(CLK)
BEGIN
IF RESET = '1' THEN
FF <= "0000";--SEED
ELSIF CLK'EVENT AND CLK = '1' THEN
CASE B IS
WHEN "00" => --SHIFT MODE
FF(0) <= SCAN_IN;
FF(1) <= FF(0);
FF(2) <= FF(1);
FF(3) <= FF(2);
WHEN "10" => --MISR MODE
FF(0) <= Y(2) XOR Z(0);
FF(1) <= FF(0);
FF(2) <= FF(1);
FF(3) <= FF(2);
WHEN "01" => --RESET MODE
FF <= "0000";
WHEN "11" => --PARALLEL LOAD MODE
FF(0) <= Z(0);
FF(1) <= Z(1);
FF(2) <= Z(2);
FF(3) <= Z(3);
WHEN OTHERS =>
FF <= "0000";
END CASE;
END IF;
END PROCESS;
Q <= FF;
SCAN_OUT <= FF(3);
```

```
Y(2)<=Y(1) XOR Z(1)XOR FF(0) ;
Y(1)<=Y(0) XOR Z(2) ;
Y(0) <= FF(3) XOR Z(3);
END ARCH;
--COMPARATOR
ENTITY CHECKER IS
PORT( CLK: INOUT BIT;E1,E2:INOUT BIT ;ERROR:OUT BIT);
END CHECKER;
ARCHITECTURE ARCH OF CHECKER IS
BEGIN
PROCESS(CLK)
BEGIN
IF(CLK'EVENT AND CLK='1') THEN
IF(E1/=E2) THEN ERROR <='1';
ELSIF (E1=E2) THEN ERROR <='0';
END IF;
END IF;
END PROCESS;
-- ERROR<= E1 XOR E2;
END ARCH;
-----------------------------------------------FADDB + MUL8X8
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
PACKAGE FADDB IS
FUNCTION ADDITION (A,B: BIT_VECTOR) RETURN BIT_VECTOR;
FUNCTION "+" (L: BIT_VECTOR; R : BIT_VECTOR) RETURN BIT_VECTOR ;
END FADD;
PACKAGE BODY FADDB IS
FUNCTION ADDITION (A,B: BIT_VECTOR) RETURN BIT_VECTOR IS
VARIABLE CARRY : BIT;
VARIABLE SUM: BIT_VECTOR(8 DOWNTO 0);
BEGIN
CARRY := '0' ;
FOR I IN 0 TO 8 LOOP
SUM(I) := A(I) XOR B(I) XOR CARRY;
CARRY := ((A(I) AND B(I)) OR (A(I) AND CARRY) OR (B(I) AND CARRY));
```

```
END LOOP;
RETURN SUM;
END ADDITION;
FUNCTION "+" (L:BIT_VECTOR; R: BIT_VECTOR) RETURN BIT_VECTOR IS
BEGIN
RETURN ADDITION(L,R);
END "+";
END FADDB;
--MUL
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE WORK.FADDB.ALL;
--USE IEEE.MATH_REAL.ALL;
ENTITY MULT IS
PORT( A, B: INOUT BIT_VECTOR(7 DOWNTO 0);
C : OUT BIT_VECTOR(15 DOWNTO 0)
);
END MULT_BV_8X8 ;
ARCHITECTURE ARCH_MULT OF MULT IS
BEGIN
PROCESS(A, B)
VARIABLE A_REG: BIT_VECTOR(8 DOWNTO 0);
VARIABLE C_REG: BIT_VECTOR(17 DOWNTO 0);
BEGIN
A_REG := '0' & A;
C_REG := "0000000000" & B;
FOR I IN 0 TO 8 LOOP
IF C_REG(0)='1' THEN
C_REG(17 DOWNTO 9) := C_REG(17 DOWNTO 9) + A_REG(8 DOWNTO 0);
END IF;
C_REG(17 DOWNTO 0) := '0' & C_REG(17 DOWNTO 1);
END LOOP;
C <= C_REG(15 DOWNTO 0);
END PROCESS;
END ARCH_MULT;
--BENCH OBILBO
```

```
LIBRARY IEEE,DISCIPLINES;
USE DISCIPLINES.ELECTROMAGNETIC_SYSTEM.ALL;
USE IEEE.MATH_REAL.ALL;
ENTITY OPBILBO IS
END OPBILBO;
ARCHITECTURE ARCH OF OPBILBO IS
SIGNAL CLK,RESET,SCAN_IN,SCAN_OUT:BIT;
SIGNAL B1B2 :  BIT_VECTOR(2 DOWNTO 1);
SIGNAL Z1,Z2 :  BIT_VECTOR(3 DOWNTO 0);
SIGNAL Q1,Q2 :  BIT_VECTOR(3 DOWNTO 0);
TERMINAL N11,N21,N31,N41 :  ELECTRICAL;
TERMINAL N12,N22,N32,N42 :  ELECTRICAL;
SIGNAL S11,S21,S31,S41,TEST11,TEST21,TEST01,TEST151 :  BIT;
SIGNAL S12,S22,S32,S42,TEST12,TEST22,TEST02,TEST152 :  BIT;
QUANTITY LIGHT11,LIGHT21,LIGHT151,LIGHT01 :  REAL ;
QUANTITY LIGHT12,LIGHT22,LIGHT152,LIGHT02 :  REAL ;
SIGNAL A,B, CIN: BIT;
SIGNAL R1,R2,R3:BIT_VECTOR(7 DOWNTO 0);
SIGNAL M1,M2:BIT_VECTOR(15 DOWNTO 0);
SIGNAL ERROR0,ERROR1,ERROR2,ERROR15:BIT;
BEGIN
MUL1:ENTITY MULT PORT MAP(R1,R2,M1) ;
MUL2:ENTITY MULT PORT MAP(R1,R3,M2);
Z1(0)<=M1(0);Z1(1)<=M1(1);Z1(2)<=M1(2);Z1(3)<=M1(15);
S11<=Q1(0);S21<=Q1(1);S31<=Q1(2);S41<=Q1(3) ;
Z2(0)<=M2(0);Z2(1)<=M2(1);Z2(2)<=M2(2);Z2(3)<=M2(15);
S12<=Q2(0);S22<=Q2(1);S32<=Q2(2);S42<=Q2(3) ;
OPBIL1:ENTITY BILBO (ARCH)PORT MAP(RESET,CLK,SCAN_IN,SCAN_OUT,B1B2,Z1,Q1);
OPBIL2:ENTITY BILBO (ARCH)PORT MAP(RESET,CLK,SCAN_IN,SCAN_OUT,B1B2,Z2,Q2);

OPP11:ENTITY OP_PROBE PORT MAP (S11,LIGHT01);
PIND11:ENTITY PIN_DIODE PORT MAP (N11,ELECTRICAL_GROUND,LIGHT01);
ADC11:  ENTITY ADC PORT MAP(N11,ELECTRICAL_GROUND ,TEST01);
OPP21:ENTITY OP_PROBE PORT MAP (S21,LIGHT11);
PIND21:ENTITY PIN_DIODE PORT MAP (N21,ELECTRICAL_GROUND,LIGHT11);
ADC21:  ENTITY ADC PORT MAP(N21,ELECTRICAL_GROUND ,TEST11);
```

```
OPP31:ENTITY OP_PROBE PORT MAP (S31,LIGHT21);
PIND31:ENTITY PIN_DIODE PORT MAP (N31,ELECTRICAL_GROUND,LIGHT21);
ADC31:  ENTITY ADC PORT MAP(N31,ELECTRICAL_GROUND ,TEST21);
OPP151:ENTITY OP_PROBE PORT MAP (S41,LIGHT151);
PIND151:ENTITY PIN_DIODE PORT MAP (N41,ELECTRICAL_GROUND,LIGHT151);
ADC41:  ENTITY ADC PORT MAP(N41,ELECTRICAL_GROUND ,TEST151);
-- 2END SET
OPP12:ENTITY OP_PROBE PORT MAP (S12,LIGHT02);
PIND12:ENTITY PIN_DIODE PORT MAP (N12,ELECTRICAL_GROUND,LIGHT02);
ADC12:  ENTITY ADC PORT MAP(N12,ELECTRICAL_GROUND ,TEST02);
OPP22:ENTITY OP_PROBE PORT MAP (S22,LIGHT12);
PIND22:ENTITY PIN_DIODE PORT MAP (N22,ELECTRICAL_GROUND,LIGHT12);
ADC22:  ENTITY ADC PORT MAP(N22,ELECTRICAL_GROUND ,TEST12);
OPP32:ENTITY OP_PROBE PORT MAP (S32,LIGHT22);
PIND32:ENTITY PIN_DIODE PORT MAP (N32,ELECTRICAL_GROUND,LIGHT22);
ADC32:  ENTITY ADC PORT MAP(N32,ELECTRICAL_GROUND ,TEST22);
OPP152:ENTITY OP_PROBE PORT MAP (S42,LIGHT152);
PIND152:ENTITY PIN_DIODE PORT MAP (N42,ELECTRICAL_GROUND,LIGHT152);
ADC42:  ENTITY ADC PORT MAP(N42,ELECTRICAL_GROUND ,TEST152);
CHECK1:ENTITY CHECKER PORT MAP(CLK,TEST01,TEST02,ERROR0);
CHECK2:ENTITY CHECKER PORT MAP(CLK,TEST11,TEST12,ERROR1);
CHECK3:ENTITY CHECKER PORT MAP(CLK,TEST21,TEST22,ERROR2);
CHECK15:ENTITY CHECKER PORT MAP(CLK,TEST151,TEST152,ERROR15);
-------STIMULI
PROCESS
BEGIN
CLK <='0','1' AFTER 0.5 NS;
WAIT FOR 1 NS;
END PROCESS;
PROCESS
BEGIN
RESET <='0','0' AFTER 100 NS;
WAIT ;
END PROCESS;
PROCESS
BEGIN
```

```
SCAN_IN <='0','0' AFTER 10 NS;
WAIT FOR 20 NS;
END PROCESS;
PROCESS
BEGIN
B1B2 <="11","11" AFTER 80 NS;
WAIT ;
END PROCESS ;
PROCESS
BEGIN
R1 <="00000011","00000011" AFTER 10 NS,"00000000" AFTER 20 NS ;
WAIT FOR 30 NS;
END PROCESS;
PROCESS
BEGIN
R2 <="00000010","00000001" AFTER 10 NS, "00000000" AFTER 20 NS ;
WAIT FOR 30 NS;
END PROCESS;
PROCESS
BEGIN
R3 <="00000010","00000000" AFTER 10 NS,"00000000" AFTER 20 NS ;
WAIT FOR 20 NS;
END PROCESS;
END ARCHITECTURE BEHAV;
```

ملخـــــــــــص

الهدف من هذه الرسالة هو اقتراح نهج جديد للاختبار الحي للأنظمة المضمنة الموجهة للتطبيقات الحساسة. يستند هذا النهج، نهج الاختبار الحي، على نظام ما يسمى بـ : "BIST" (Built-In Self-Test) أو الاختبار المدمج الذاتي أين تم إضافة لاقطات بصرية إلى الهيكل الكلاسيكي المعروف بـ : "BILBO" (Built-In Logic-Block Observation). و على هذا تم اقتراح سجلات "BILBO" بصرية جديدة "OBILBO" (Optical BILBO). الطريقة الكاملة التي تم تطويرها في هذه الأطروحة تم تكييفها مع طريقة النسخ الفيزيائي تسمح بضمان الفحص الوظيفي للنظام في حالة عمل للكشف عن وجود أخطاء محتملة، دون مقاطعة أو إبطاء سير وظيفته العادية، كما تسمح بالكشف الفوري للأخطاء البسيطة بعد دورة واحدة دون اللجوء إلى أية عملية لتوليد معلومات (أشعة) الاختبار. بالإضافة إلى ذلك، فإن هذا النهج يسمح أيضا بأداء اختبارات غير حية إذ أصبح من الممكن تحقيق تشخيص الأخطاء التي قد تظهر على السجل "OBILBO" ذاته. ففي هذه الحالة هذا النهج يقلل من عدد عمليات المسح من N عملية مسح (حيث : N هو عدد طوابق السجل OBILBO) اللازمة لأداء مثل هذا الفحص إلى عملية مسح واحدة وبالتالي تجنب الطابع التسلسلي لهذا الأخير. بالإضافة إلى هذه الميزة، السجل OBILBO ينتج إمضاء بصري و الذي يمكن تحليله عن بعد. لقد تم استخدام لغة البرمجة الخاص بالدارة الكهربائية المندمجة VHDL-AMS للنموذجة والمحاكاة و إثبات فعالية التقنية المقترحة.

الكلمات المفتاح : تصميم الدارات المندمجة ، تركيب الدارات ، TLM ، SystemC ، SystemVerilog ، VHDL ، اختبـــــــــار و قابلية الاختـــــــــبار.

ABSTRACT

The objective of this work is to propose a new online testing approach for embedded systems which are dedicated to critical applications. This online testing approach is based on a BIST (Built-In Self-Test) approach in which optical sensors are added to the classical architecture BILBO (Built-In Logic-Block Observation). Thus, a new Optical BILBO registers (OBILBO) are proposed. The complete test method developed in this thesis, adapted to the hardware duplication technique, ensures a functional checking of the system during its normal operating in order to detect the presence of possible errors without interrupting or slowing down its normal functioning. Besides, it allows immediate detection of simple stuck at faults with error latency of only one clock cycle without having recourse to any operation of test vectors generation. In addition, this approach also allows to perform tests in offline mode by diagnosing faults that may appear on the OBILBO registers themselves. The advantage of this approach is to reduce the number of the N shifting operations (N: number of the OBILBO register stages) necessary to perform such a scan to a single shifting operation and thus to avoid the serial nature of this register. In addition to this possibility, the OBILBO register equally produces an optical signature that can be analyzed at remote system. The VHDL-AMS tool is used to model, simulate and validate the technique presented in this manuscript.

Keywords : *Integrated Circuit Design, Synthesis of Circuits, TLM, SystemC, SystemVerilog, VHDL, Test and Testability.*