Titre: Title:	Conception d'un système de test et de configuration numérique tolérant aux pannes pour la technologie WAFERIC
Auteur: Author:	Yan Basile-Bellavance
Date:	2009
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Basile-Bellavance, Y. (2009). Conception d'un système de test et de configuration numérique tolérant aux pannes pour la technologie WAFERIC [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/235/

UNIVERSITÉ DE MONTRÉAL

CONCEPTION D'UN SYSTÈME DE TEST ET DE CONFIGURATION NUMÉRIQUE TOLÉRANT AUX PANNES POUR LA TECHNOLOGIE WAFERIC

YAN BASILE-BELLAVANCE
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION DU
DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉLECTRIQUE)
DÉCEMBRE 2009

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

CONCEPTION D'UN SYSTÈME DE TEST ET DE CONFIGURATION NUMÉRIQUE TOLÉRANT AUX PANNES POUR LA TECHNOLOGIE WAFERIC

présenté par : BASILE-BELLAVANCE Yan
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. DAVID, Jean-Pierre, Ph.D., président
M. SAVARIA Yvon, Ph.D., membre et directeur de recherche
M. BLAQUIÈRE Yves, Ph. D., membre et codirecteur de recherche
M. BOIS, Guy, Ph. D., membre

DÉDICACE

Je dédie ce mémoire à mes parents qui n'ont jamais cessé de m'encourager à poursuivre mes études et à faire preuve de persévérance.

REMERCIEMENTS

En premier lieu, je veux remercier Yvon Savaria mon directeur de recherche et directeur du département de génie électrique de l'École Polytechnique Montréal. Je garde une grande reconnaissance envers Yvon Savaria, car il a su me faire confiance pour faire partie de l'équipe du projet DreamWafer. En deuxième lieu je veux remercier Yves Blaquière, mon codirecteur, pour sa contribution dévouée et généreuse à ma formation d'ingénieur. M. Blaquière était toujours disponible pour discuter de mon travail même s'il a un emploi du temps chargé en tant que directeur des programmes en microélectronique de l'Université du Québec à Montréal.

Je me suis senti privilégié d'avoir 2 excellents mentors durant mes études à la maîtrise. Yves Blaquière et Yvon Savaria ne m'ont pas seulement formé au niveau technique, mais au niveau humain pour m'aider à atteindre un niveau d'excellence qui fera une différence dans ma vie professionnelle.

En troisième lieu, je tiens à remercier Richard Prytula, président de TechnoCap Inc., le partenaire industriel du projet DreamWafer, pour deux raisons. Premièrement, au niveau humain pour sa gestion exigeante et audacieuse du projet, pour les nombreux conseils qu'il nous a donnés sur le démarrage et la gestion de projet technologique. Deuxièmement pour sa contribution financière et son courage pour avoir eu foi en l'équipe DreamWafer sans quoi le projet n'aurait jamais pu voir le jour.

Il ne faut pas oublier Richard Norman qui, le premier, a eu la vision du projet DreamWafer. Sans M. Norman, le projet n'aurait jamais vu le jour aussi rapidement. Sa vision ambitieuse à long terme du projet DreamWafer nous a permis de rester motivés même dans les moments les plus difficiles et incertains. Plusieurs discussions passionnantes avec R. Norman m'ont permis de développer une vision unique et avant-gardiste de la microélectronique.

De plus, je veux remercier mes principaux coéquipiers et amis : Étienne Lepercq, Olivier Valorge et Nicolas Laflamme-Meyer. Il faut spécialement remercier Étienne Lepercq, car son incroyable efficacité au travail m'a poussé à me surpasser sans cesse. Et un merci à Étienne pour toutes les fois qu'il m'a aidé à résoudre certains problèmes techniques rencontrés sur mon chemin. Je dois remercier Olivier Valorge pour les conseils techniques en microélectronique avancée qu'il m'a donnés depuis le tout début du projet et Nicolas Laflamme-Mayer pour son support ponctuel et indispensable qu'il m'a offert généreusement dans le domaine des PCBs.

L'équipe DreamWafer a vu passer beaucoup de monde. Je tiens à remercier tous les étudiants qui ont contribué au projet : Youssef El Alaoui, Marc-André Daigneault, Moufid Radji et Anh Tuan Nguyen.

Un grand merci à Réjean Lepage pour son soutien technique informatique rapide et efficace. Réjean Lepage a aussi été un ami et je voudrais le remercier pour son humour et sa gentillesse. Sans oublier Ghyslaine Éthier qui a toujours été là pour s'occuper de nous. Je ne veux pas oublier de remercier les collègues du département de génie électrique avec qui j'ai passé mes études à la maîtrise : Gilbert Kowarzyk, Sébastien Éthier, Jaouad El Fouladi, Philippe Aubertin, David Bafumba-Lokilo, Moufid Radji, Louis-Francois Tanguay, Étienne Boulais, Roula Ghannoum et Mame Maria Mbaye.

Finalement, je dois remercier mes parents Donald Bellavance et Marie-Jeanne Basile pour leur soutien moral constant, eux qui ont toujours su m'encourager à aller loin dans mes études.

RÉSUMÉ

L'objectif principal du projet de recherche est de concevoir, implanter et vérifier un système de programmation JTAG tolérant aux pannes pour un circuit intégré à l'échelle de la tranche (WSIC, *Wafer Scale Integrated Circuit*). Le projet comprend la conception de l'interface logicielle/matérielle, l'implantation en VHDL du système, la conception de l'environnement de vérification SystemC ainsi qu'une étude sur la "diagnosabilité" du WaferIC un circuit WSIC au cœur d'un système configurable applicable au prototypage rapide.

Une nouvelle approche face à la conception de bancs de test programmable pour le test de circuits numériques est en cours de développement dans plusieurs universités québécoises, dont l'École Polytechnique Montréal dans le cadre du projet "DreamWafer^{TM}". Ce nouveau système de prototypage rapide de circuits numériques a pour but de mettre au point un réseau intégré d'interconnexions configurables, nommé WaferNet. Ce réseau d'interconnexions est déployé sur un circuit intégré à l'échelle de la tranche. Ainsi, le projet "DreamWafer^{TM}" vise à développer un système équivalent à un "PCB reconfigurable" permettant de prototyper des circuits intégrés numériques discrets (FPGA, processeurs, DSP…), ceux-ci étant déposés à sa surface. Ce circuit intégré contient une matrice comportant des milliers de cellules identiques, chacune comportant un centre de contrôle logique, un crossbar configurable et un ensemble de "plots" de quelques dizaines de micromètres de large (points de contact avec les composants déposés). Cette matrice de cellules se nomme le WaferIC.

Ce projet de maîtrise porte spécifiquement sur la conception d'un système de configuration tolérant aux pannes pour le WaferIC, en la mise au point d'un environnement de simulation et de vérification matérielle codé en SystemC et en VHDL, à concevoir l'interface logicielle/matérielle pour le contrôle de la configuration basée sur le protocole JTAG et la conception d'une méthodologie de test et de diagnostic du système de configuration et du

WaferNet. La tolérance aux pannes est importante dans le cadre de cette application spécifique pour des raisons économiques et pour atteindre le niveau de qualité requis pour cette application.

Parmi les résultats publiés dans ce document figurent le bilan du plan de vérification, les résultats de test et diagnostic du premier prototype “ASIC” du circuit fabriqué par l’entremise de CMC Microsystems, réalisé en technologie CMOS 0.18 μm. La réalisation et la vérification de ce premier prototype ASIC a été possible grâce à un système de test de circuit adapté à notre situation particulière de test. Ce système a été mis en place par une équipe de chercheurs dont l’auteur a fait parti.

ABSTRACT

The goal of this master project is to design, implement and validate a new system able to control the WaferIC, a Wafer Scale Integrated Circuit (WSIC). More specifically, the project objective was to design the software/hardware interface, design and implement an embedded fault-tolerant control system and implement from scratch an environment in SystemC for functional verification. Moreover, the ASIC synthesis is applied on the VHDL code to fabricate a test chip to validate the circuit.

A new approach for rapid prototyping of digital systems is in development at several universities, including École Polytechnique de Montréal, through the "DreamWafer™" project. The goal of this new system is to interconnect all the digital pins of a set of discrete chip at the system level by using a reconfigurable network called WaferNet. This interconnection network is deployed over the active surface of a whole wafer. This wafer scale integrated system called WaferIC aims at implementing a form of reconfigurable PCB that is able to reconnect the digital pins of discrete chips at will. User's ICs deposited on the active surface of the wafer are detected by an array of tiny reconfigurable "NanoPads" that can redirect the signals in the WaferIC's internal network or feed the user's IC pins with data and power.

The specific contribution of this master project consists of designing a fault-tolerant system to test and configure the WaferIC, to implement a verification environment coded in a mixed language SystemC/VHDL. This environment implements a software/hardware interface for the WaferIC and the design of a new test and diagnosis methodology for the reconfigurable network. Fault tolerance is an important issue for this class of circuit for economic reasons, and to reach the quality required for this application.

This document reports results obtained while testing and validating a test chip (1x1 mm^2) that has been fabricated. Those results proved that the WaferNet concept works properly and the fault tolerant test and configuration system works as expected.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

ASIC	Application Specific Integrated Circuit
BIST	Built-In Self-Test
BSDL	Boundary Scan Description Language
CDSP	Circuit Discret (sur la surface du WaferIC) Sous Prototypage
CLB	Configurable Logic Block
CUT	Cell Under Test
DFD	Design For Diagnosis
DFT	Design For Testability
DUT	Device Under Test
FPGA	Field Programmable Gate Array
JTAG	Joint Test Action Group
LFSR	Linear Feedback Shift Register
MISR	Multiple Input Signature Register
MTG	Multi-Threaded Graph
NoC	Network on Chip
NoW	Network on Wafer
PCB	Printed Circuit Board
RTL	Register Transfer Level
SA0/1	Stuck-At zero/one
SiP	System in Package
SoC	System on Chip
SoW	System on Wafer
TC	Test Controller
ULSI	Ultra Large System Integration
VHDL	Very high speed integrated circuits Hardware Description Language
VLSI	Very Large System Integration
WSI	Wafer Scale Integration

LISTE DES ANNEXES

INTRODUCTION

Il est bien connu que la miniaturisation et la performance des circuits intégrés s'accroît à un rythme exponentiel. Cette progression s'est maintenue pendant plusieurs décennies, mais depuis quelques années, un ensemble d'obstacles et de problèmes se pointer à l'horizon. Ces obstacles pourraient compromettre cette progression. Beaucoup de ces nouveaux problèmes ne sont pas causés par les circuits intégrés eux-mêmes, mais par les PCB (*printed circuit board*), c'est-à-dire des cartes imprimées qui supportent et relient les circuits intégrés. En effet, la conception d'un système électronique composé de circuits intégrés à la fine pointe de la technologie n'est pas une tâche facile. Plusieurs problèmes attendent les concepteurs qui s'aventurent dans le domaine de la conception de cartes électroniques à haute fréquence et à grande densité. Parmi ces problèmes figure la densité d'interconnexion qui amène des problèmes de routage et de validation.

Présentement, le problème de routage est géré de trois façons qui consistent à :

1- Amincir les traces d'interconnexions sur le PCB. Cette solution est rapidement limitée par les procédés de fabrication. En effet, la vitesse de miniaturisation des circuits intégrés dépasse celle de la miniaturisation des interconnexions des PCB.

2- Augmenter le nombre de couches d'interconnexion sur le PCB, ce qui augmente le coût de fabrication des systèmes électroniques.

3- Augmenter la fréquence des signaux propagés sur les lignes pour augmenter la quantité d'information qui peut transiter entre les circuits. Cette solution est aussi très coûteuse, car bien que possible, cela s'avère requérir une grande expertise de concepteurs de circuits hyperfréquences qui est rare et dispendieuse.

L'augmentation de la densité d'interconnexions des PCB crée un autre problème, celui de la validation. Puisque la densité et le nombre de broches des boîtiers de circuits intégrés ne cesse d'augmenter, la densité des interconnexions des PCB, donc le nombre de couches augmente, complique le test quand il est temps d'accéder à des points de test particuliers

qui sont sous des boitiers de circuits intégré de type BGA (Ball Grid Array) ou profondément enfouis dans des PCB pouvant comporter plus de 20 couches.

Une autre difficulté qui émerge de la densification des cartes électroniques vient du principe que plus un design est complexe, plus il est difficile de résoudre les bogues matériels et logiciel ainsi que ceux qui résultent de leur interaction. Ce problème est aggravé par l'accroissement de l'utilisation de systèmes embarqués gérés par des systèmes d'exploitation de complexité croissante. Par exemple, pour certains designs dans le domaine des télécommunications, il est possible de rencontrer des bogues qui n'apparaissent qu'après plusieurs milliards de cycles exécutés, ce qui rend le déverminage encore plus ardu. Les problèmes peuvent être autant au niveau logiciel que matériel et il faut être capable de diagnostiquer efficacement un problème pour le résoudre.

Ainsi, le temps de test et de validation de chaque prototype soumis à la fabrication amène des coûts et des efforts non négligeables pour l'industrie électronique. Sans compter les coûts associés au fait qu'il faut souvent plusieurs itérations avec le fabricant de PCB pour arriver à un système fonctionnel. Ceci est problématique quand une ronde de prototypage d'un PCB coûte plus de 10000$, qu'elle implique des délais de fabrication d'un mois à chaque ronde de correction d'erreur et que ces PCB sont *populés* de composants coûteux et souvent rares dans la phase de développement d'un système. C'est pour répondre à ces problèmes qu'il existe sur le marché toute une gamme de produits visant à permettre la conception rapide et efficace d'un prototype de systèmes électroniques sur PCB. Parmi ces produits figurent les cartes de développement pour CPU ou microcontrôleur, les "*breadboard*", les FPGA et les équipements spécialisés de validation et de test. Un système de prototypage rapide de systèmes numériques idéal aurait les caractéristiques suivantes :

1- Capacité à tester le circuit sous test à la fréquence nominale d'opération du circuit prototypé.

2- Le circuit prototypé peut être modifié à volonté pendant qu'il est en fonction.

3- Le circuit prototypé doit être facile et rapide à modifier.

4- Une observabilité et une contrôlabilité de tous les points d'opérations d'intérêt du circuit.

Ainsi, une équipe de conception de système électronique à haute densité et à haute vitesse aurait un avantage indéniable en ayant à sa disposition un système de prototypage rapide de systèmes numériques capable de répondre à toutes ces exigences. Par contre, il n'existe encore aucun système de prototypage rapide de système numérique idéal capable de répondre à toutes les exigences citées plus haut.

Puisque "la nécessité est la mère de toute invention", un nouveau type de circuit intégré a été inventé pour répondre aux besoins grandissants du marché des PCB et fait l'objet de ce mémoire de maîtrise. Cette invention de monsieur Richard Norman [27] apporte une solution élégante et viable au problème du prototypage de PCB. La conception de ce circuit est réalisée dans le cadre d'un projet de recherche auquel contribuent plusieurs universités (École Polytechnique Montréal, UQAM, UQO et McGill) et soutenu par un partenaire industriel, Gestion TechnoCap Inc. Ce projet se nomme DreamWafer. Le brevet écrit par Richard Norman[27], qui participe intensivement au projet, décrit en détail un nouveau type de circuit intégré à l'échelle de la tranche qui pourrait à terme avoir une application commerciale dans le domaine du prototypage rapide de systèmes numériques. Le circuit intégré à l'échelle de la tranche est un circuit qui se déploie sur toute la surface active d'une tranche de silicium plutôt que sur un seul dé comme avec les circuits classiques. Le propriétaire du brevet est la compagnie Gestion TechnoCap Inc. Cette technologie brevetée se nomme WaferBoard^{TM}.

L'idée de base consiste à créer un réseau reprogrammable sur substrat à l'échelle de la tranche. Cela signifie qu'un circuit permet de reprogrammer les interconnexions d'un réseau pour créer un circuit unique correspondant au schéma du circuit ("netlist") soumis par l'usager. Pour interconnecter entre eux les circuits intégrés d'un système, un système de

configuration doit être conçu avec un pont de communication entre le logiciel et le substrat programmable.

Le circuit intégré à l'échelle de la tranche qui a été conçu et en voie d'être breveté dans le cadre du projet DreamWafer a été baptisé WaferIC. Il contient une matrice de plusieurs millions de plots (appelés NanoPads) de quelques dizaines de micromètres de dimension. Ces plots sont en fait des étages d'entrées/sorties configurables connectés entre eux par un réseau d'interconnexion (WaferNet) reconfigurable. La grande densité de plots évite d'avoir à aligner les broches des circuits intégrés avec les NanoPads. Cela permet d'installer facilement et rapidement un ensemble de puces sur la surface du WaferIC et de configurer dynamiquement leurs interconnexions.

À terme, le WaferBoard^{TM} permettra de rabaisser à quelques jours de travail ce qui prenait plusieurs mois avec des prototypes sur PCB conventionnels. Non seulement on y trouve un gain de productivité, mais il y a une économie sur les coûts de prototypage. En effet, sous-traiter la fabrication d'un prototype sur PCB peut coûter très cher. Chaque aller-retour avec le fabricant de PCB implique un coût. De ce fait, on abaisse sensiblement le coût de développement d'un nouveau produit électronique (économie sur les salaires d'ingénieurs et sur l'utilisation d'équipement de test). De plus, le temps d'arrivée sur le marché ("*time-to-market*") est un aspect économique important qui est pris en compte par les acteurs de l'industrie de l'électronique.

Une petite équipe interuniversitaire (Polytechnique-UQAM-UQO) s'est formée pour la conception et la réalisation d'un circuit prototype à petite échelle (4 étudiants à la maîtrise, 4 professeurs, 1 post-doctorant). Certains membres de l'équipe se concentrent plus sur l'aspect analogique. Olivier Valorge qui vient de compléter son stage postdoctoral a travaillé à résoudre des problèmes au niveau de l'alimentation en puissance, de l'intégrité du signal, de la propagation rapide des signaux et du design physique. Nicolas Laflamme-

Mayer, étudiant à la maîtrise, travaille sur un système distribué de régulateurs de tension et de puissance pour le WaferIC. Étienne Lepercq, étudiant à la maîtrise, contribue au WaferNet, principalement au niveau logiciel sur la méthodologie de travail nécessaire pour automatiser le travail des concepteurs matériels qui utiliseront le DreamWafer. Moufid Radji, aussi étudiant à la maîtrise, travaille sur l'aspect micro-fabrication et "post-traitement" du WaferIC.

Lesujet de recherche de cette maîtrise couvre la conception du système de communication numérique entre le logiciel et le WaferIC. La partie spécifique de mon projet de maîtrise consiste en la conception d'un système de configuration tolérant aux pannes pour le WaferIC, en la mise au point d'un environnement de simulation et de vérification matérielle codé en SystemC et en VHDL, à la conception de l'interface logicielle/matérielle pour le contrôle de la configuration basé sur le protocole JTAG[28] et la conception d'une méthodologie de test et diagnostic du système de configuration.

Pour accéder aux fonctionnalités du WaferIC, il faut un système de programmation tolérant aux pannes, que l'on peut reconfigurer en fonction des zones mortes du circuit. Une zone morte est un endroit défini dans le circuit où il est connu qu'une défaillance fonctionnelle existe. Cette défaillance crée par effet domino d'autres défaillances sur des zones déterminées du circuit. Tout élément contenu dans ces zones ne doit donc pas être utilisé par le système. Ce système a été basé sur le protocole IEEE 1149.1 JTAG. Pour commander le WaferIC à partir d'un ordinateur, il faut mettre au point un émulateur JTAG adapté aux besoins spécifiques du circuit. Un environnement de vérification a été codé en C++ et en SystemC pour tester et vérifier l'interface logicielle/matérielle et pour vérifier la validité des fonctionnalités internes de chaque cellule. Un plan de vérification a été rédigé et plusieurs cas de test ont été développés pour valider le WaferIC, ainsi que l'émulateur logiciel du JTAG. De plus, il est important de rendre le WaferIC testable et diagnosticable, en prenant en compte le protocole JTAG.

Parmi les résultats publiés dans ce mémoire figure le bilan du plan de vérification, ainsi que les résultats de test et diagnostic du premier prototype “ASIC” d’un sous-ensemble du WaferIC fabriqué par l’entremise de CMC Microsystèmes en technologie CMOS 0.18 µm. Ce premier prototype ASIC a été testé grâce à un système de test de circuit adapté à notre situation particulière de test. Mes contributions au développement du système de test du circuit sont décrites dans le reste du mémoire.

Le mémoire est organisé en 5 chapitres :

- Une revue de la littérature et un bref résumé du brevet de R. Norman sur le WaferBoard et le WaferIC sont présentés au chapitre 1.
- De nouvelles solutions matérielles au problème de la configuration JTAG tolérante aux défectuosités sont présentées au chapitre 2.
- La méthode de test et de diagnostic à base de logiciel du WaferNet sont décrites au chapitre 3.
- L’implantation matérielle de la solution et sa vérification sont présentées au chapitre 4.
- Les résultats de la validation de notre solution sur un ASIC de test et divers résultats connexes sont présentés au chapitre 5.

CHAPITRE 1. REVUE DE LITTÉRATURE

Le présent chapitre est composé de 5 parties. Les deux premières parties visent à décrire les tendances dans l'évolution et les limites des systèmes sur puce (SoC) et des systèmes sur PCB. Ensuite, la troisième partie vise à décrire la technologie et le paradigme des systèmes sur tranche (SsT) qui devraient à terme dépasser en complexité les deux précédentes familles de systèmes électroniques (SoC et PCB). En quatrième partie, un effort particulier doit être investi dans la description du WaferIC qui se veut une nouvelle famille de circuit intégré. Le WaferIC est un concept en voie d'être breveté par Richard Norman. En effet, un brevet a été déposé avant le début du projet DreamWafer. Finalement en cinquième partie, les fondements technologiques sur lesquels se basent les contributions de ce mémoire sont décrits. Ces fondements sont la tolérance aux pannes dans les chaînes de balayage, et le diagnostic de pannes dans un réseau sur puce.

1.1 L'évolution des systèmes sur puce (SoC)

Dès les débuts de la commercialisation des processeurs intégrés sur puce comportant des périphériques intégrés sur le même substrat, l'industrie est entrée dans l'ère des systèmes sur puce, généralement connue selon son nom anglais de "System-On-Chip" (SoC)[30]. Les systèmes sur puce intègrent sur un même substrat de semiconducteur les fonctionnalités de plusieurs types de circuits, pour créer un composant capable d'effectuer des fonctions complexes, effectuées auparavant par des systèmes électroniques entiers comportant plusieurs puces discrètes. Les systèmes sur puces peuvent contenir des circuits numériques, analogiques/mixtes et souvent ils contiennent des circuits radio-fréquence.

L'inclusion de tous ces types de modules sur un seul circuit intégré a nécessité un effort de recherche et développement très intensif. Cet effort continue de se maintenir pour améliorer plusieurs aspects des SoC. Avec leur évolution, il se dessine des limitations techniques

difficiles à contourner qui devraient ralentir la progression de cette technologie dans les années à venir.

Voici une liste non exhaustive de ces limitations :

1- ***La consommation en puissance :*** À mesure que les procédés de fabrication avancent vers des niveaux de miniaturisation de plus en plus proches du nanomètre, de nouveaux problèmes physiques apparaissent. À des échelles bien plus petites que le micromètre (inférieures à 45 nm), le phénomène de courant de fuite devient de plus en plus significatif et même nuisible. Ceci crée des circuits plus rapides et plus petits, mais parfois beaucoup moins efficaces énergétiquement. Des solutions peuvent s'appliquer pour contourner le problème. Par exemple, des techniques modernes de gestion dynamique de la puissance permettent d'adapter dynamiquement le niveau de tension ou de fréquence en fonction des besoins.

2- ***L'accroissement de la complexité :*** La surface maximale occupée par un circuit intégré est actuellement limitée par un grand nombre de facteurs, parmi lesquels figurent les erreurs de lithographie associées à la méthode de fabrication des circuits, la pureté des matériaux et des produits chimiques, etc. Pour rendre cette technologie rentable, il est obligatoire de couper la tranche de silicium en "dés" et d'encapsuler ces "dés" dans un boîtier pour être fixés sur un PCB et pour être connecté à d'autres circuits intégrés.

3- ***Le rendement de production :*** La taille d'un circuit intégré n'affecte pas seulement le coût de fabrication en utilisant plus de silicium, mais elle affecte surtout la probabilité de retrouver une défectuosité de fabrication sur sa surface. À mesure que la surface du circuit s'accroît pour atteindre la grandeur de la tranche de silicium qui le constitue, la probabilité de trouver une défectuosité s'accroît rapidement vers un.

C’est pour cette raison que les méthodes standards de conception de circuit ne permettent pas de fabriquer des circuits à l’échelle de la tranche (WSI : wafer scale integration).

4- ***La photo-répétition et la taille des réticules :*** Une autre limitation de la grandeur des circuits intégrés vient de la méthode de la photolithographie elle-même. Les circuits intégrés sont fabriqués à partir de la répétition d’image essentiellement carrée sur une tranche de silicium par le principe de la photo-lithographie [23]. La structure ‘imprimée’ de façon répétitive sur la surface de la tranche est communément appelé un réticule et son empreinte sur la surface de la tranche n’excède typiquement pas 2,5 cm de coté. La taille du réticule proprement dit est généralement plus grande que l’image réticulaire par un facteur déterminé (comme 5X par exemple). Par conséquent, tout design de circuit intégré à l’échelle de la tranche de silicium doit normalement être composé d’un agencement de cellules régulier à l’échelle de l’image d’un réticule, sans quoi il faudrait disposer d’autant de jeux de masque que d’images distinctes, ce qui bien que théoriquement possible devient prohibitif.

1.2 Les limitations des systèmes sur PCB

Les systèmes sur PCB ne cessent de se complexifier. La bande passante demandée entre les circuits intégrés ne cesse d’augmenter, à mesure que les capacités de calcul et de traitement des circuits intégrés s’améliorent.

Cette augmentation de la complexité cause plusieurs problèmes, dont les deux cités ici.

1- La complexité croissante des PCB rend le test et le débogage au niveau système ardu. La miniaturisation des boitiers et des broches («pins» en anglais)

qu'ils possèdent pour relier des puces au reste d'un système électronique ne cesse de s'améliorer avec le temps. Le diamètre d'une des balles métalliques qui sertit un boîtier de circuit intégré atteint 200 µm en 2009. Cette progression entraîne une augmentation de la densité d'interconnexion des traces (lignes électriques) sur le PCB pour atteindre une largeur de 85 à 100 µm cette année (2009) [17]. Les circuits de la classe FPGA utilisent des matrices de plots d'une densité linéaire de 45 plots par cm et cette densité ne cesse d'augmenter [33]. L'augmentation de la densité des traces entraîne des difficultés pour les concepteurs de PCB, car les méthodes traditionnelles de test telles que le "bed of nails" (test sous pointe) deviennent de plus en plus difficiles et coûteuses à appliquer. Dans certains types de circuits sur PCB très dense, des portions du circuit deviennent même impossibles à tester avec ces méthodes, car les points d'accès sont parfois cachés par plusieurs couches de métallisation et d'isolation et le réseau d'interconnexions du circuit qu'ils implémentent est très dense.

2- La complexité croissante des PCB rend de plus en plus important l'accroissement de productivité des concepteurs. Beaucoup de technologies alternatives ont été développées pour améliorer quelques-uns des inconvénients. Parmi ces solutions alternatives, il y a les MCM (multi-chip module) pour les systèmes à haute performance. Une autre solution est d'empiler des puces ("*stacked chip*"), mais elle ne peut supporter aujourd'hui qu'un nombre limité de puces empilées.

Une foule d'outils ont été conçus pour tenter de résoudre ces problèmes. Parmi les solutions existantes, deux ont retenu notre attention comme étant suffisamment proches de la technologie WaferBoard pour être décrites dans ce document.

1- Le modèle Zebu XXL

Le ZebuXXL **Erreur ! Source du envoi introuvable.** est un émulateur de système capable de tester des FPGA ou des ASIC en relation avec le système global qui les entoure. Ainsi, le ZebuXXL peut être configuré rapidement et facilement pour émuler le comportement du système global sous test. Il a été conçu pour être utilisé au niveau de la vérification logiciel-matériel à l'étape finale de l'intégration de toutes les parties d'un système. De cette façon, les concepteurs logiciels et matériels peuvent partager un même environnement de développement et de vérification, ce qui permet d'améliorer la productivité d'une équipe de travail.

1-1 La plateforme Zebu-XXL.

Cette technologie utilise une approche différente de la technologie WaferBoard pour faire du prototypage rapide de systèmes numériques. Le ZebuXXL peut émuler le comportement numérique d'un SoC à partir d'une description RTL synthétisable, si cette description RTL a une complexité inférieure à ce que l'émulateur peut gérer.

2- Le modèle NanoBoard

Le NanoBoard est un des premiers systèmes de prototypage rapide de système numérique qui a été offert sur le marché [13]. Ce système n'est pas un émulateur comme le modèle ZebuXXL, mais un véritable système permettant d'augmenter la vitesse de prototypage de système numérique. La technologie a été développée par Altium, une compagnie très innovatrice et ancienne qui a, entre autres, été la première à commercialiser un logiciel CAO pour la conception de circuits imprimés. Continuant à innover dans son domaine

d'excellence, Altium a développé un système d'aide à la conception pour les systèmes intégrés qui comprend à la fois un logiciel CAO et un environnement de développement matériel qui permettent d'accélérer la conception de systèmes électroniques complexes.

1-2 La plate-forme de développement NanoBoard^{TM}.

Les circuits intégrés qui sont au cœur du système à concevoir sont soudés sur une carte imprimée spéciale dite "daughter-board" qui est capable de se connecter à l'environnement de développement. Une fois connectés, tous les circuits qui sont sur le "daughter-board" peuvent être testés et vérifiés avec l'aide des logiciels CAO fournis avec le NanoBoard pour accélérer dramatiquement le développement du produit. Le NanoBoard peut ainsi déverminer des circuits numériques reprogrammables (FPGA) et de petites cartes de développement tout en supportant une interaction temps réel avec le système en cours de conception.

1.3 L'avènement des systèmes sur wafer (SoW)

La demande pour une intégration croissante de modules hétérogènes dans des volumes de plus en plus petits a amené la création du principe de SiP (System in Package) [22]. Ce type de puce est dite "3D" parce qu'elle permet d'empiler verticalement des circuits intégrés dans un même boîtier. Ce type de circuit est principalement utilisé dans le domaine des cellulaires et autres circuits électroniques portables. Puisque le marché des cellulaires est en pleine croissance, il existe un grand nombre d'usines de fabrication capables de produire à

grande échelle des SiP et la croissance de ce segment du marché se mesure avec un taux de croissance à 2 chiffres.

Comme alternative au SiP (ou comme solution compétitrice), un nouveau concept a été proposé en 2009 sous le nom de “System on Wafer” (SoW) par une équipe de recherche française, le LETI-CAE [29]. Cette approche consiste à souder des circuits intégrés “flip-chip” sur un wafer complet avant de le découper et l’enfermer dans un boîtier. Cette méthode permet d’empiler 2 étages de circuits. Le principal avantage de cette approche est d’utiliser un substrat de silicium comme interconnexions et support mécanique entre les “flip-chip”. Cette technologie permet d’ajouter une foule de circuits actifs CMOS au système existant. Ces circuits peuvent servir comme circuit de test, circuit de reconditionnement de signal, etc. Même si on n’intègre pas de circuit actif CMOS dans le substrat, le procédé de fabrication peut tout de même atteindre des résolutions d’un ordre de grandeur supérieur à ce qui peut se faire de mieux dans le domaine des PCB, ce qui permet de faciliter les routages denses entre les flip-chips du SoW.

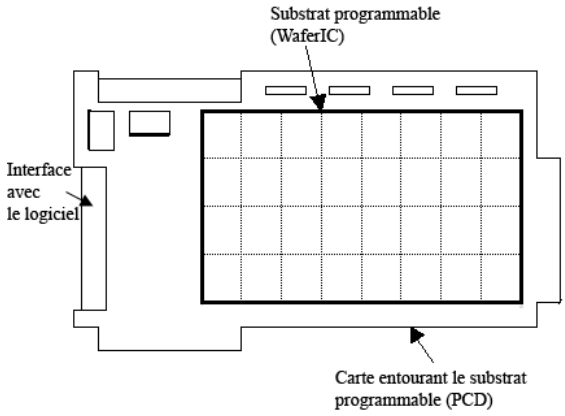
C’est dans ce contexte qu’est apparu le brevet de R. Norman [27] décrivant les bases de la technologie WaferIC. La section suivante vient décrire la technologie telle qu’elle a été conçue par R. Norman en 2007. De plus, une revue des plus récents développements du projet et des articles publiés sur le sujet est effectuée.

1.4 Le WaferBoard

Le WaferBoard^{TM} est un système de prototypage rapide pour les circuits numériques. Le système est conçu à partir d’un SsT et permet de tester, valider rapidement des circuits au niveau système. Une carte électronique comprend principalement un substrat programmable (le WaferIC) et une interface rapide pour se connecter à un ordinateur (voir fig. 1-3). Le substrat programmable est un circuit électronique de type “*wafer scale IC*”, c'est-à-dire un circuit intégré au niveau de la tranche de silicium. Cette technologie permet la communication de données entre plusieurs réticules d’une même tranche. Ces données

sont propagées directement par les couches de métallisation intégrées sur la tranche de silicium. Le substrat programmable est connecté à une carte PCI express par le dessous de la tranche et échange des données avec un ordinateur hôte pour l'auto-test, la configuration, la programmation, le test du système conçu par l'usage en cours de prototypage. Les données fournies par l'ordinateur sont des bits de configuration pour permettre au substrat programmable d'adapter sa topologie de connexions internes pour implémenter une "*netlist*" représentant les interconnexions voulues par un ingénieur procédant au design d'un système numérique.

Par exemple, les données transmises à l'ordinateur sont des lectures de points de contact entre les broches des circuits discrets et la surface du substrat. Les circuits discrets déposés à la surface du WaferIC seront appelés dans ce document des CDSP (circuit discret sous prototypage).



1-3 Exemple de WaferBoard. Image tirée du brevet [27].

Le WaferBoard est par conséquent un système de prototypage rapide capable d'interconnecter rapidement un ensemble de composants électroniques. C'est en quelque sorte un PCB reprogrammable supportant une densité d'interconnexion supérieure aux PCB multicouches conventionnels. Le *Wafer Integrated Circuit* (WaferIC) est le nom donné au substrat actif qui peut être beaucoup plus grand qu'un circuit intégré conventionnel. Ce

substrat actif dépasse la taille limite d'un réticule et il peut occuper la surface complète d'une tranche de silicium. Une expression utilisée couramment dans ce document est "substrat programmable" pour exprimer l'idée qu'un substrat qui peut couvrir une tranche entière de silicium, peut être programmé pour réaliser les interconnexions désirées à l'échelle du système. À partir d'un WaferIC, il est possible de concevoir une nouvelle famille de circuits intégrés capable d'attaquer des problèmes difficiles à résoudre avec les familles de circuits intégrés de la taille d'un réticule assemblés sur PCB ou MCM .

Le WaferBoard peut réduire le temps de développement et les coûts qui sont associés au développement de systèmes impliquant un PCB et plusieurs circuits digitaux discrets. Par analogie, les FPGA ont contribué à réduire le temps de développement de circuits intégrés par rapport aux ASIC en permettant l'utilisation d'un tissu logique programmable pré-manufacturé, capable d'émuler le comportement d'une fonction logique complexe.

L'analogie entre les FPGA et le WaferBoard peut être poussée jusqu'au niveau de la production. Tout comme les ASIC ont un coût de production inférieur aux FPGA pour de grands volumes, un design fait à l'aide d'un PCB aurait un coût de production inférieur à un WaferBoard pour un design vendu en grande quantité. Par contre, il est connu qu'un système réalisé à l'aide de FPGA peut rivaliser en termes de coût de production avec les ASIC pour les produits complexe vendus en volume modéré.

Pour être compétitif sur le marché, les systèmes électroniques (avec PCB ou MCM) actuels doivent incorporer des circuits intégrés discrets d'une complexité croissante, tout en rencontrant des contraintes de plus en plus exigeantes au niveau de la consommation de puissance et de la taille. Quelques-unes des principales difficultés ne viennent pas des composants discrets mais de leurs interconnexions. En effet, les outils de simulation et de CAO dédiés au domaine du design de PCB sont tous très matures et la fiabilité des designs à ce niveau est très bonne.

Dans le cas du WaferBoard, les composants électroniques sont simplement déposés manuellement sur une surface active qui a la capacité de détecter les contacts entre elle-même et ces composants. Cela signifie que le WaferIC permet un placement insensible à l'alignement contrairement aux PCB dont les circuits discrets doivent être soigneusement alignés sur des plots avant de les souder. Le WaferIC envoie les informations sur le placement des circuits intégrés ce qui permet par la suite au système d'afficher le résultat de cette détection sur un écran d'ordinateur. La "netlist" représentant l'interconnexion des broches du système à concevoir est envoyée au WaferIC, qui les interconnecte pour permettre le transfert des signaux. Ces connexions sont programmées par un port JTAG d'une manière similaire aux FPGA. Les signaux passent à travers des répéteurs pour en préserver l'intégrité sur les longues interconnexions.

Une approche d'intégration à l'échelle de la tranche ("WSI, *Wafer Scale integration*") offre deux avantages : 1- elle permet d'avoir une grande densité de contacts, capables de supporter l'interconnexion des puces les plus miniaturisées ; 2 – elle permet une densité d'interconnexions suffisante pour mettre en place des systèmes de grande complexité. Cette approche permet donc de créer une plate-forme de prototypage de circuits digitaux unique en son genre.

Le principal avantage du WaferIC n'est pas d'atteindre une plus grande puissance de calcul, comme proposé dans certains travaux antérieurs sur le WSI [8][19]. Il s'agit en fait d'obtenir une surface suffisamment grande pour interconnecter de manière dense un ensemble de puces discrètes pour créer un système électronique de complexité supérieure en un temps de conception raccourci. Déjà en 1989, des chercheurs tels que Brewer [8], pionnier du domaine du WSI, ont démontré la faisabilité du concept d'intégration à l'échelle de la tranche et plusieurs règles de conception ont été définies dans un travail ultérieur [20]. Beaucoup de ces règles de conception font appel au concept de régularité de

la structure pour gérer le problème de la limitation en surface causée par les masques lithographiques, mais aussi pour gérer le problème des défectuosités inévitables en utilisant des stratégies de tolérance aux fautes.

La technologie CMOS 180 nm est considérée en date de dépôt de ce mémoire comme une technologie mature et les coûts de fabrication qu'elle implique rendent envisageable de concevoir une tranche de silicium entièrement dédiée à un seul système. Les usines qui offrent une technologie CMOS 180 nm produisent des tranches de silicium de grandeur 6 à 8 pouces.

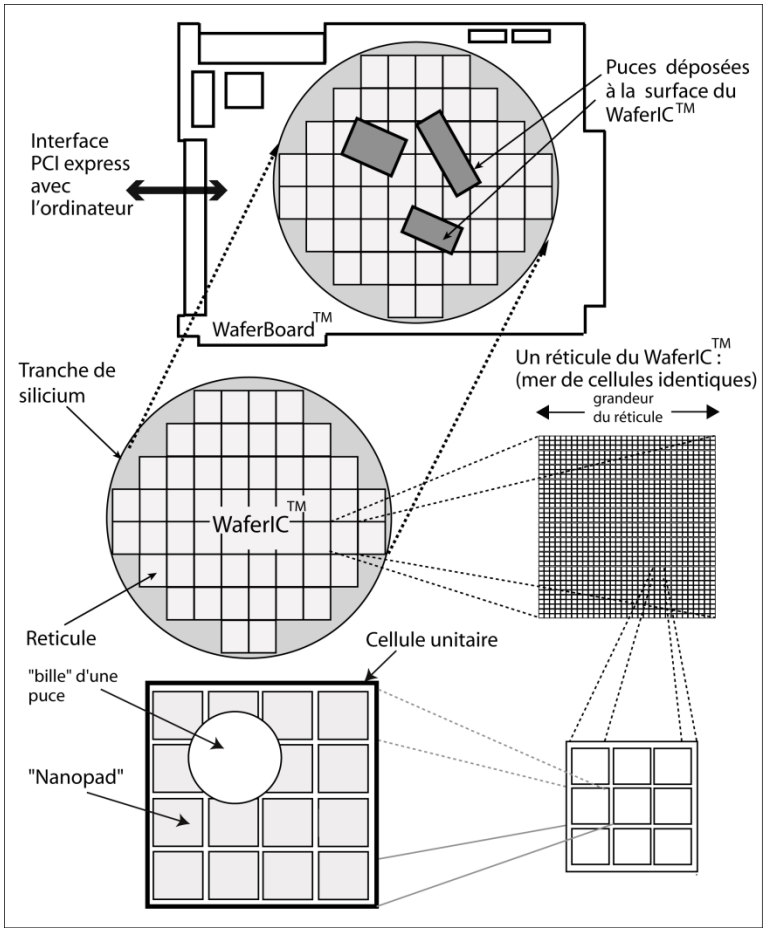
La structure de la surface du WaferIC comprend une matrice de plots très fins (de l'ordre de 100 μm) réalisée avec la dernière couche de métallisation du circuit intégré. Une couche spéciale est ajoutée par-dessus le circuit intégré. Cette couche est assimilable à un tissu qui protège la surface de l'usure. Ce mince tissu composé à la base d'un polymère flexible qui permet la conduction électrique selon son épaisseur, aussi appelé axe Z (d'où le nom "Z-axis film"). Les meilleurs tissu-axe-Z présentement disponibles sur le marché ont une épaisseur d'environ 300 microns Chaque plot sur le WaferIC est appelé "NanoPad". La matrice de plots est suffisamment dense pour permettre le contact entre les NanoPads et les broches d'une puce déposée sur la surface du tissu-axe-Z. Les NanoPads sont connectés au réseau interne reconfigurable du WaferIC appelé WaferNet.

Le WaferIC est constitué d'une mer de cellules identiques et régulières. Concevoir l'architecture du WaferIC à base de cellules identiques permet d'en simplifier dramatiquement le design. La cellule de base du WaferIC est appelée dans ce document simplement la "cellule unitaire". Ces cellules sont collées les unes à côté des autres en une matrice à l'intérieur de la surface maximale limitée par le réticule. Contrairement aux procédés usuels de fabrication, les "dés" (les cellules) constituant la tranche ne sont pas coupés et empaquetés dans un contenant de plastique. Les "dés" sont plutôt interconnectés

pour créer diverses fonctionnalités essentielles au WaferBoard. D'où le nom "*wafer scale integrated circuit*." Pour assurer une communication entre les réticules, une nouvelle étape du procédé de fabrication doit être mise au point pour tracer des lignes électriques entre les réticules ("reticle stitching"). Ainsi, le WaferIC est une mer de cellules connectées ensemble par un réseau reconfigurable d'interconnexions couplé à une matrice très dense de NanoPads.

Le WaferIC est en fait une matrice de cellules identiques au niveau physique. Les cellules sont différenciées par le contenu de leur mémoire interne une fois le circuit configuré. La mémoire interne sert à configurer un crossbar permettant de définir la topologie du système. Cela permet de configurer le mode NanoPads (VDD, IN, OUT, GND) et d'autres fonctionnalités. Parmi ces fonctionnalités figurent des senseurs de courants, tensions, structure de test, etc. Cela suffit pour permettre à une matrice de cellules de créer un "PCB reconfigurable". Cette méthode de conception est conforme aux méthodes proposées par les travaux antérieurs [8][20] de pionniers du domaine WSI, où la tolérance aux pannes et la régularité structurale doivent être maintenues.

La figure 1-4 montre la hiérarchie du système. Chaque cellule possède une dimension de 540x540 µm. Elle contient 16 NanoPads, ce qui donne une résolution suffisante pour isoler chaque broche de chaque puce d'un système électronique et évite les courts-circuits entre les broches des puces.

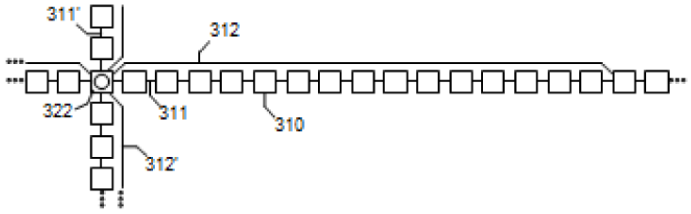


1-4 Structure physique hiérarchique du WaferBoard™ et du WaferIC™.

Voici les fonctionnalités internes supportées par chaque cellule du circuit :

1- Le système de configuration tolérant aux pannes des NanoPads et du crossbar commandé par le protocole JTAG.
2- Le système de détection de contact.
3- Le crossbar.
4- Support interne au diagnostic pour la tolérance aux pannes.
5- Les pads bidirectionnels configurables (les NanoPads).

Le réseau d'interconnexion sur Wafer : le WaferNet (figure 1-5) **:** Il existe plusieurs types de réseaux configurables tels que le réseau "hypercube" [15] ou bien le réseau de type "Benes" [9]. Le WaferNet est différent(voir figure 1-5). Il ne s'agit pas d'un simple maillage, mais d'un maillage multidimensionnel régulier. Chaque cellule est connectée à ses 4 voisins, mais aussi à ses voisins distants de plusieurs cellules. Chaque cellule inclut un crossbar de grande dimension pour connecter les liens entrants provenant des 4 directions vers les liens sortants se dirigeant vers les 4 directions. Les cellules sont connectées aux 4 voisins immédiats dans les 4 directions, mais sont aussi en connexion directe avec les voisins distants de 2, 4, 8, 16…, etc. cellules. Dans la première implantation, des lignes relient directement des cellules éloignées jusqu'à 64 cellules. Chaque cellule unitaire contient des répéteurs pour propager efficacement les signaux numériques. Ces longues interconnexions nécessitent des répéteurs distribués dans chaque cellule pour minimiser le temps de propagation des signaux.



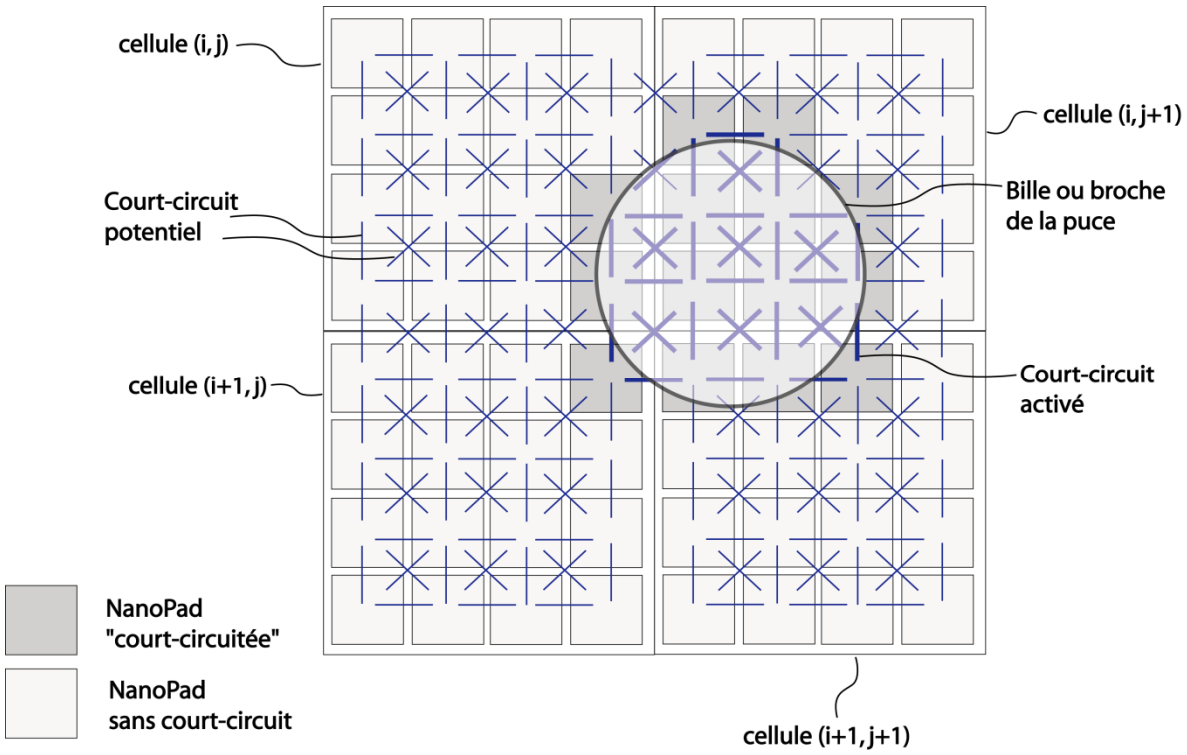
1-5 Structure du réseau WaferNet, extrait du brevet [27] .

L'architecture en maille multidimensionnelle permet non seulement d'améliorer la densité de routage, mais elle permet aussi d'obtenir une tolérance aux pannes dans le réseau. En effet, puisqu'il y a plus de liens que nécessaire, il est possible d'utiliser les ressources d'interconnexions non utilisées comme liens de remplacement. Par contre, pour être capable d'éviter une panne, il est essentiel de connaître son emplacement. Pour les trouver, il faut des techniques de diagnostic du réseau.

La détection de contact : Une fonction essentielle du WaferIC est de détecter un contact entre une broche de circuit intégré et un ou plusieurs NanoPads. Il existe plusieurs techniques qui ont été documentées dans le brevet de Norman [27] et les papiers antérieurs [12]**Erreur ! Source du renvoi introuvable.**. Ces techniques se basent sur deux lasses de méthodes. La première utilise le changement de capacité observé sur le NanoPad lorsqu'il est en contact avec une broche du circuit intégré. Cette technique n'a pas encore été validée expérimentalement.

La deuxième classe de méthodes se base sur l'utilisation de détection de court-circuit entre les NanoPads. Puisque les broches des puces doivent être obligatoirement conductrices d'électricité et que les NanoPads sont plus petits que les broches, alors une broche qui repose sur le WaferIC créera obligatoirement un court-circuit. Il s'agit donc de créer un circuit analogique de détection de court-circuits entre les NanoPads pour détecter les points de contact "broches-NanoPads" du circuit. Une illustration de la mécanique de détection de contacts est montrée à la figure 1-6. Une bille de puce est montrée sur la figure 1-6. Les lignes représentent des liens potentiels reliant les NanoPads entre eux. Tous les liens potentiels reliant les NanoPads entre eux ne sont pas représentés.

Un court-circuit entre 2 NanoPads est montré avec une ligne plus foncée sur la figure. Les NanoPads flottants sont montrés en blanc. Après l'étape de détection de contact effectué par les circuits de détection de court-circuit, le système transmet le résultat du test vers l'extérieur, d'où une image des contacts peut être recréée pour donner à l'utilisateur la possibilité d'interconnecter les "broches" des circuits intégrés du système à prototyper.



1-6 Exemple de l'effet de la procédure de détection de courts-circuits.

Tolérance aux pannes par contrôle externe

Le concept illustré à la figure 1-7 est le suivant : tous les cellules peuvent être contrôlées par les cellules voisines. Cela permet d'obtenir une configuration redondante nécessaire au remplacement des cellules défectueuses. Cette propriété améliore la tolérance globale aux pannes, car les fonctions des cellules saines voisines peuvent être utilisées par une cellule défectueuse pour lui permettre de rester fonctionnelle. En reliant les cellules voisines, les fonctionnalités non utilisées des cellules voisines peuvent être considérées comme des modules de rechange en cas de panne.

Une autre méthode de tolérance aux pannes bien connue est la tolérance basée sur l'utilisation des cellules voisines comme le montre la figure ci-dessous tirée du brevet de R. Norman [27] :

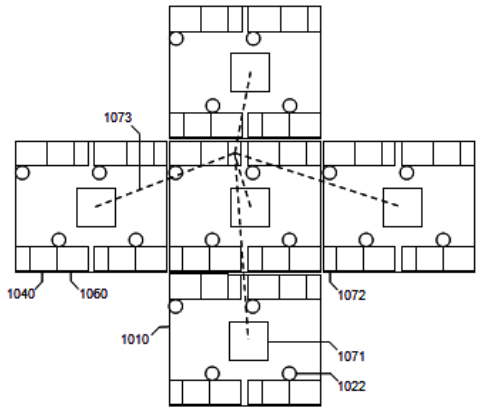


FIG. 10B

1-7 Utilisation des ressources des cellules voisines (indiquées en pointillé)[27].

1.5 Les bases technologiques des contributions de ce mémoire

Cette section présente les bases technologiques des principales contributions originales de ce mémoire qui se rapporte audiagnostic de pannes dans un réseau de maillage multidimensionnel (le WaferNet) et une méthodologie de tolérance aux fautes.

1.5.1 Les systèmes de chaîne de balayage "niveau système"

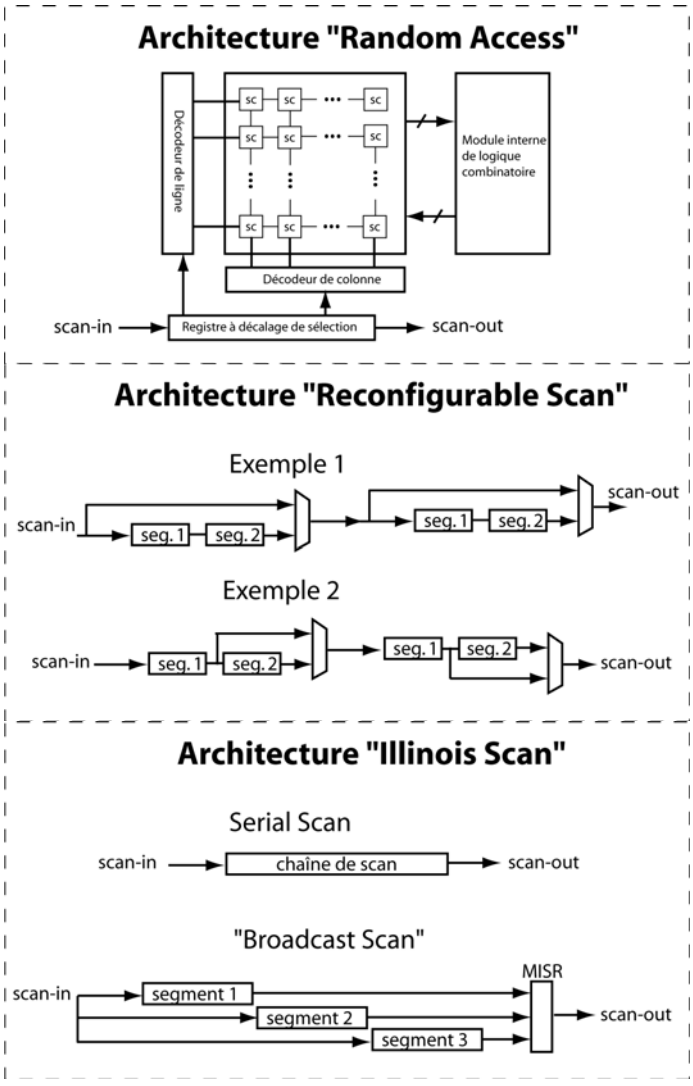
Le design pour la testabilité ("*DFT, design for testability*") est la discipline de la microélectronique s'intéressant aux techniques de test de circuits numériques modernes. Une bonne stratégie de test permet de tester le circuit à un coût et avec une surface de silicium tous deux minimum. De plus, il faut que la couverture du test soit maximale et qu'elle soit effectuée le plus rapidement possible. Une des principales techniques pour tester les circuits numériques est l'insertion de registres à décalage (appelés dans ce document chaîne de balayage) dans le circuit pour obtenir une contrôlabilité ou une observabilité des modules numériques internes. Tel qu'expliqué précédemment, le

diagnostic des défectuosités est à la base des techniques de tolérance aux pannes privilégiées dans la technologie “WaferIC”. Or, diagnostiquer un circuit, c’est l’art de localiser les pannes détectées par le test du système numérique. Les techniques de test sont donc à la base du diagnostic. Puisque les techniques de diagnostic pour la tolérance aux pannes se basent sur une architecture spéciale de chaîne de balayage il est donc pertinent de faire une revue des techniques d’insertion de chaîne de balayage.

Il existe un grand nombre de types de chaîne de balayage. La chaîne de balayage la plus élémentaire est une simple suite de registres reliés entre eux pour créer un registre à décalage. La figure 1-8 montre schématiquement les principaux types de système de chaîne de balayage. Par exemple, avec une chaîne de balayage de type “RASC : *random access scan chain*” il est possible de créer une matrice de registres accessibles à l’aide de décodeurs de colonnes et de lignes [3]. L’avantage de ce type d’architecture est d’accélérer le temps de test et la puissance consommée durant le test. Par contre, il nécessite beaucoup de ressources matérielles et de routage pour être réalisé dans un circuit numérique. Il est aussi possible de créer un système de chaîne de balayage à l’aide de chaînes de balayage reconfigurables [25]. L’idée principale est d’utiliser des multiplexeurs configurables pour contourner les registres les moins souvent utilisés durant une séquence de test pour minimiser la longueur de la chaîne de balayage. Cette technique permet d’accélérer le temps de test avec l’utilisation d’un minimum de ressources matérielles supplémentaires.

Une autre méthode est la technique d’insertion de chaînes de balayage de type “Illinois”, aussi connue sous le nom de *broadcast scan chain* [16]. Ce type de chaîne de balayage permet d’appliquer parallèlement la même séquence de vecteurs de test sur une structure régulière indépendante. La chaîne de type “Illinois” peut passer d’une chaîne de balayage élémentaire en une chaîne de balayage de type “*broadcast*”. Ce changement de mode est utilisé pour gérer les situations où il est possible d’appliquer des cas de test unique sur une série de modules parallèles (mode “*broadcast*”) et le mode élémentaire est utilisé pour les

situations où un vecteur de test unique doit être appliqué à chaque module sous test. Une MISR (*Multiple Input Signature Register*) est ajoutée à la sortie des chaînes de balayage pour comprimer le volume de données en sortie du circuit sous test.



1-8 Principales classes d'architecture de scan.

1.5.2 Architecture de test

Le présent mémoire porte principalement sur le système de programmation tolérant aux défectuosités du WaferIC et sur le diagnostic du WaferIC. Il est donc pertinent de procéder

à un survol rapide des principales architectures de test de circuits intégrés et de PCB. Le système de programmation et de diagnostic du WaferIC est basé sur le protocole de test IEEE 1149.1.

La norme 1149.1 définit un protocole d'accès pour le test [28]. Elle définit un ensemble d'architectures "*boundary scan*" pour le test de circuits numériques. La figure 1- montre un exemple de *boundary scan* sur un DUT ("*device under test*") où chaque entrée/sortie de la puce est reliée à une cellule "*boundary scan*" qui est chaînée à ses cellules voisines pour former une chaîne globale accessible par un seul port JTAG plus étroit. Chaque cellule est en fait un registre de capture et de contrôle des broches de la puce. À travers cette chaîne, les entrées/sorties deviennent contrôlables et observables. Ce qui permet de tester à la fois la logique interne du circuit et les interconnexions sur le PCB.

En comparaison du test d'un système, le test de circuit intégré pris d'une manière indépendante est relativement facile, car toutes les entrées/sorties du circuit sont accessibles. Nous insistons ici sur le mot relatif, car en fait, ce problème relativement facile peut s'avérer très difficile dans l'absolu! Par contre, une fois que la puce est intégrée et soudée sur un PCB, le problème du test de la puce devient plus complexe. Les entrées/sorties des puces du système sur PCB ne sont généralement pas directement accessibles. Une technique populaire consiste à utiliser un protocole de communication sériel pour obtenir un accès à toutes les broches de tous les circuits intégrés d'un système au travers d'un port d'accès de largeur minimale. Le protocole IEEE 1149.1 propose une solution élégante à ce problème. Ce protocole de test porte plusieurs noms. Il peut être appelé selon le contexte "*boundary scan*" ou bien simplement "JTAG" (*Joint Test Access Group*). Initialement, cette norme a été conçue pour le test de cartes électroniques (PCB). Mais le succès et l'élégance du protocole de communication en ont permis un élargissement rapide à une gamme d'applications complexes reliées au test de circuits intégrés de grande dimension, à la programmation de circuits intégrés et au diagnostic des systèmes

électroniques, etc. Une description détaillée du protocole de communication et de test JTAG serait trop volumineuse et est facilement disponible au lecteur intéressé à approfondir le sujet [28][31]. De plus, une connaissance de base du protocole JTAG est requise pour comprendre les concepts et principes énoncés dans les sections et chapitres ultérieurs de ce document. Une petite annexe résumant le protocole JTAG a été ajouté en annexe G. Plusieurs architectures de test sont possibles. Voici une énumération des principales architectures tirée du livre de Wang [31] :

Architecture JTAG de base : Dans chaque système JTAG, il y a le contrôleur de test (TC, Test controller) et l'unité sous test (UUT, unit under test). Une unité sous test peut être n'importe quelle puce, PCB, SoC ou bien un système WSI sous test. La même architecture de base peut être utilisée pour une unité sous programmation. Par conséquent, l'expression "unité sous test" réfère à tout système configurable ou bien programmable avec un port JTAG. Cette simplification permet d'alléger le texte. Les UUT sont stimulés à l'aide de vecteurs de test et la réponse est évaluée par le contrôleur de test pour décider si le système est fonctionnel ou pas. Le contrôleur de test peut être un logiciel ou bien un système embarqué capable de contrôler l'émission et la réception des flux de bits JTAG et aussi d'interpréter les résultats. L'interprétation des résultats peut être un simple constat d'échec ou de réussite à un test électronique ou bien le TC peut être capable à l'aide d'un logiciel approprié d'émettre des diagnostics.

Architecture JTAG en anneau : La figure 1-9 représente une autre architecture JTAG bien connue sous le nom de "daisy chain" où chaque chaîne de balayage de chaque UUT est reliée pour former un anneau. Un seul signal TMS est envoyé à chaque cellule pour contrôler le test. Le premier et le dernier UUT sont les seules cellules en contact direct avec le contrôleur de test. Cette architecture offre un accès efficace tout en utilisant un minimum de broches dédiées uniquement aux tests. À noter que les différentes puces peuvent recevoir des instructions de test différentes même s'ils ont le signal de contrôle TMS. Puisque la position de chaque puce dans la chaîne est connue, il est possible de donner une instruction

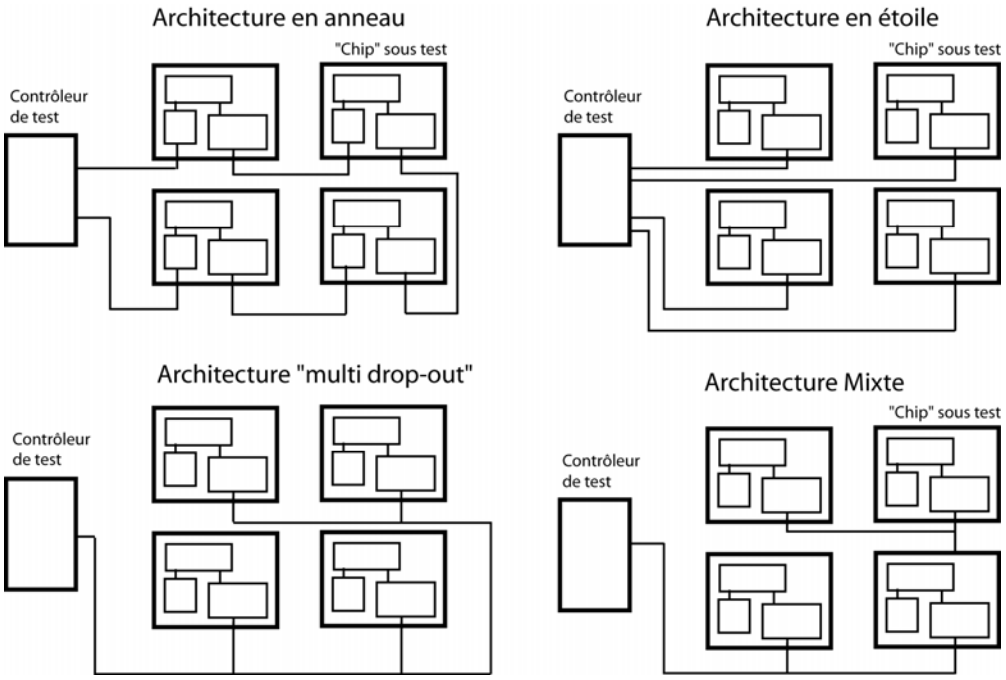
JTAG et des vecteurs de tests uniques à chaque puce faisant partie de la chaîne. Par contre, la structure en anneau possède une vulnérabilité majeure : si seulement un seul UUT n'est pas fonctionnel, l'anneau en entier devient impossible à tester, impossible à configurer.

Architecture JTAG en étoile : La figure 1-9 représente une autre architecture bien connue où chaque TC est directement en contact avec chaque UUT. Par conséquent, s'il y a "*n*" UUT dans le système sous-test, il faut entre 4*n* et 5*n* ports dédiés au test dans le système. Si le signal JTAG de reset trst*(optionnel) est utilisé dans chaque cellule le nombre de ports dédiés au test sera de 5*n*. Cette architecture apporte un accroissement de la rapidité du test, car le test des cellules se fait en parallèle au prix d'un plus grand nombre de ports de test. De plus, cette architecture résout la vulnérabilité apportée par les chaînes en anneau.

Architecture JTAG de type "Multi drop-out module" : Une solution hybride entre la solution en anneau (sériel) et la solution en étoile (parallèle) est la solution décrite à la figure 1-9. Dans une architecture "multi drop-out", chaque cellule est reliée par un bus de communication bidirectionnel et relié en une seule colonne de cellules de mémoire, comme dans une puce de mémoire RAM. Une adresse est spécifiée à un décodeur d'adresse donne accès aux données qui peuvent transiter directement entre le TC et l'UUT choisis. De même, les résultats de test provenant des UUT peuvent être acheminés vers le TC à partir du même bus en un schéma de communication "one-to-one". Cette solution est très désirable, car elle offre un accès étroit aux UUTs et elle permet d'accélérer le test. De plus, cette solution apporte un certain niveau de tolérance aux pannes. Si un seul UUT fait défaut, le système entier peut rester testable et diagnosticable, donc réparable. Par contre, si une panne arrive sur le bus de communication, les composants connectés sur ce bus deviennent non-testables, donc dysfonctionnels.

Architecture JTAG mixte : Une architecture de test JTAG mixte [7] utilise et combine de manière hiérarchique les 3 méthodes précédentes (anneau, étoile, multi-drop out). Plusieurs combinaisons d'architecture deviennent possibles. Par exemple, un PCB complexe pourrait

utiliser un système multi-drop out pour accéder à plusieurs puces du système, mais un seul SoC du même PCB pourrait ensuite contenir une architecture interne de test de type “anneaux” pour accéder aux fonctionnalités internes du SoC. La figure 1- montre un exemple d’architecture mixte hiérarchique. De plus, le brevet [21] vient décrire en détail et protéger une architecture mixte particulièrement adaptée aux tests de systèmes complexes qui doivent être testés rapidement et efficacement.



1-9 Exemples d'architectures JTAG.

Les architectures JTAG tolérantes aux fautes : Un dernier type de chaîne de balayage étudié dans ce mémoire est la chaîne de balayage tolérante aux fautes conçue par Lu et Savaria en 2003 [24]. Cette technique utilise l’approche connue sous le nom de “TMR, Triple Modular Redundancy” pour ajouter une tolérance aux pannes. Cette approche consiste à utiliser en parallèle 3 systèmes qui sont évalués par un quatrième module qui a pour but de sélectionner en sortie des données fiables sans fautes. Ce quatrième module fait “voter” les 3 systèmes et seulement la réponse de la majorité (2 sur 3) est acceptée comme

réponse en sortie. Si une panne ponctuelle arrive dans le système, la probabilité qu'elle arrive en même temps sur 2 des 3 systèmes est très basse, ce qui améliore la fiabilité du module global. De même, si une panne permanente existe dans l'un des 3 modules, les 2 autres pourront prendre le relais et permettre au système de fonctionner en présence de cette faute. La technique proposée dans [24] consiste à renforcer la chaîne de balayage en appliquant un chemin de balayage qui est triplé.

1.5.3 Le diagnostic de pannes dans un système d'interconnexions

En partant d'un circuit qui a échoué à un test fonctionnel, le diagnostic logique (simplement appelé diagnostic pour alléger le texte) consiste à cerner dans une région de plus en plus petite du circuit les régions possibles où peut se trouver une défectuosité, ce qui permet d'en estimer les causes possibles. En cernant autant que possible les causes de défectuosités, on peut souvent trouver la cause première d'un syndrome de panne, soit au niveau du design ou bien au niveau de la production en série. Le diagnostic efficace de circuits intégrés peut donc contribuer à raccourcir leur temps de développement (déverminage matériel) et à améliorer les rendements de fabrication. Cette sous-section présente un résumé des principales techniques de diagnostic qui sont reliées à celles exploitées dans ce document. Toutes les méthodes de diagnostic originales présentées dans les chapitres 2 et 3 représentent soit des améliorations ou des adaptations à la technologie WaferIC inspirées de techniques de diagnostic qui ont été appliquées à des PCB ou à des FPGA.

Méthodes de diagnostic de réseau d'interconnexion exploitées dans les PCBs : le diagnostic des interconnexions dans un PCB consiste non seulement à tester, mais à localiser exactement quelles traces sont défectueuses et à localiser les défauts de soudure entre les puces et le PCB. Ce genre de test peut être effectué à partir d'une chaîne de balayage pouvant soit observer ou contrôler les signaux de toutes les broches d'un PCB. Cette méthode a été standardisée à travers le BSDL (Boundary scan description langage) et le JTAG. Les architectures JTAG pour le test et le diagnostic ont déjà été couvertes dans ce

chapitre, mais pas la méthode pour diagnostiquer la défectuosité. Si l'ordre des cellules de balayage est connu par un logiciel de test et si ce logiciel peut générer des modèles de vecteurs de test, il devient possible de localiser et de détecter les principales pannes, comme les courts-circuits, les collages ("SA, stuck-at") et les circuits ouverts. Le modèle de test le plus connu et le plus facile à générer par du matériel embarqué sur la carte sous test est l'envoi d'un "*walking-one*" [14]. Pour ce faire, il est possible d'utiliser un simple compteur connecté en sortie à une porte NOR (OR). La sortie résultante génère automatiquement une série de bits 10000… (01111…). Cette série de bits est ensuite décalée dans la chaîne JTAG pour faire ressortir les pannes d'interconnexion du PCB. En connaissant l'emplacement exact de chaque pin dans les chaînes de balayage, il devient possible de faire un diagnostic très précis en un temps d'ordre $O(n^2)$.

Une autre méthode utilisée pour détecter les défectuosités dans les traces de PCB est la méthode connue sous le nom de "*checkerboard*". elle nécessite un temps d'ordre $O(n\log(n))$. Cette méthode consiste à utiliser le balayage de bordure associé à chaque broche de chaque puce du PCB et d'appliquer successivement des vecteurs de test de la forme d'un damier. Par exemple, appliquer un damier sur un système contenant 16 broches de sortie connecté à une ou plusieurs entrées donnerait la séquence de 4 vecteurs : "1111111100000000, 1111000011110000, 1100110011001100, 1010101001010101". La même séquence en inversant tous les bits permet de détecter tous types de court-circuits ou de collages.

*Méthode de diagnostic des pannes d'interconnexion dans les FPGA*s :

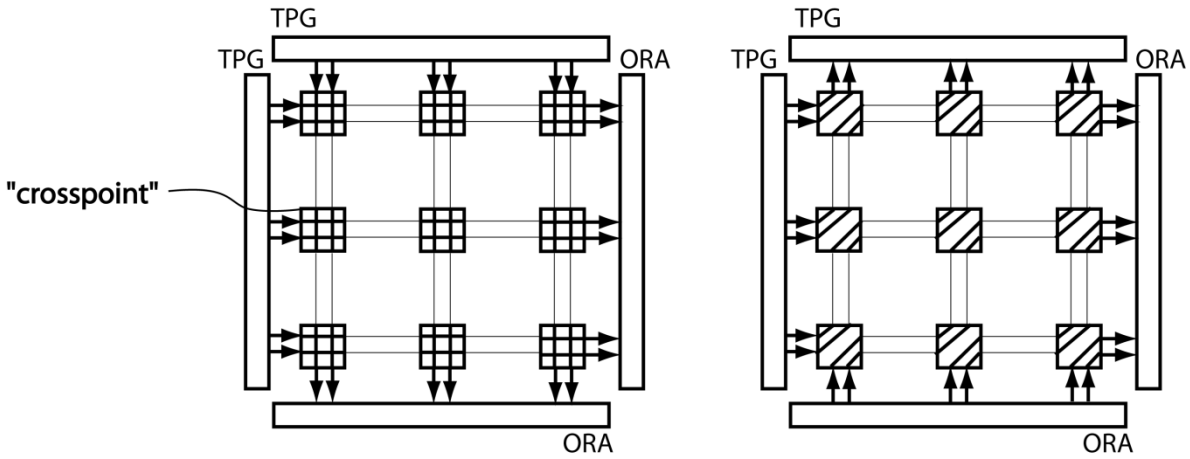
Le test et le diagnostic des interconnexions dans les FPGA sont similaires au test et au diagnostic du WaferNet et ils sont similaires à tout type d'architecture de type RRNoC

(Regular and Reconfigurable Network on Chip). Toutefois, il y a une différence basée sur une prémisse qui est à la base des techniques détaillées dans ce document : les diagnostics se font uniquement à partir d'un port JTAG et non en utilisant toutes les entrées/sorties. Par conséquent, le test et diagnostic peut se faire chez l'utilisateur. Cette capacité technique ouvre la porte à de nouvelles possibilités dans le domaine de la tolérance aux pannes, comme nous le verrons dans les sections ultérieures.

Il existe un grand nombre de techniques de diagnostic des interconnexions des FPGA. Parmi ces techniques, il y a 2 classes fondamentales [11] : (1) le diagnostic des pannes dans les CLBs (CLB, Configurable Logic Block) disponibles dans le FPGA de XILINX; (2) le diagnostic dans les interconnexions. Seulement le diagnostic des pannes dans les interconnexions sera revu dans ce chapitre.

Méthode de diagnostic des interconnexions en utilisant les capacités de reprogrammation du FPGA :

Il existe un grand nombre de brevets et d'articles couvrant l'approche du diagnostic de pannes en utilisant la capacité de reprogrammabilité des FPGA [11]. L'idée de base inhérente à cette classe de méthode est d'utiliser l'infrastructure matérielle du FPGA dédiée à la reprogrammation (CLB) pour autovalider et autogénérer des vecteurs de tests à partir de l'intérieur du FPGA. Le but est de créer un chemin de données couvrant un aussi grand nombre d'interconnexions que possible. Chaque interconnexion reliée à un chemin devient en fait une interconnexion sous-test ("IUT, interconnect under test"). Il est avantageux de tester de manière concurrente le plus grand nombre possible d'interconnexions en même temps comme le montre la figure 1-10 suivante :



1-10 Exemple de diagnostic en utilisant les capacités de reconfiguration du FPGA.

Il est impossible de tester toutes les interconnexions en une seule passe. Il faut donc reprogrammer la topologie interne du réseau d'interconnexions. Il s'agit donc de tester le plus grand nombre d'interconnexions, avec la plus grande résolution possible, tout en reprogrammant le moins souvent possible le FPGA, car chaque reprogrammation a un coût en temps de test. Dans le cas du diagnostic, il faut utiliser la même technique, mais avec un algorithme de recherche commandé par un contrôleur de test.

Parmi tous les articles ou brevets existant à ce jour, c'est le brevet [1] qui se rapproche le plus de la méthode de diagnostic présentée dans ce mémoire. Le brevet protège principalement des techniques de diagnostic "on-the-fly", c'est-à-dire un diagnostic fait pendant qu'un FPGA est en cours d'utilisation soudé sur son PCB. La technique consiste à générer à l'interne des données sur des chemins créés dans les ressources inutilisées du FPGA. Ces chemins sont recréés à plusieurs endroits du FPGA (clonés) par groupe de trois. Si le résultat sortant d'un des 3 chemins est différent des 2 autres, c'est la preuve que quelque part dans ce chemin, il existe une interconnexion fautive. Par un algorithme de recherche d'interconnexion fautive, il devient possible de localiser une panne avec précision et de ce fait de compléter le diagnostic de tout le réseau interne d'interconnexion.

Notre approche pour le test et le diagnostic est similaire, parce que nous utilisons aussi l'idée d'un algorithme de recherche dans le réseau d'interconnexion, mais les algorithmes proposés dans le chapitre 3 diffèrent significativement dans les détails comme nous allons le montrer.

1.6 Conclusion

Ce chapitre de revue de la littérature a porté sur les types de chaînes de balayage existantes et sur les architectures de test niveau système. De plus, il a été décidé d'inclure dans la revue de littérature le travail antérieur dédié au WaferIC et au WaferBoard. Ces travaux ont été effectués par l'équipe dont le présent projet de maîtrise fait parti, mais ils décrivent de manière générale la technologie de prototypage rapide en cours de développement. L'architecture du WaferIC et du WaferBoard et sa différenciation des systèmes courants tels que les MCM, les SiP ou les SoC existant. Cette différence doit être montrée pour comprendre correctement le contexte associé aux méthodes de diagnostic et de tolérance aux pannes des systèmes proposés dans ce document.

Le brevet de Norman [27] décrit d'une manière générale comment un WSI peut générer un système de prototypage rapide de systèmes numériques. Il ne décrit pas exactement comment un tel système peut être implanté à l'aide des technologies existantes. Ainsi, aucune description originale de système de configuration/programmation tolérante aux pannes n'est proposée dans le brevet. De plus, aucune description de système de diagnostic de pannes dans le WaferIC n'est décrite dans le brevet. Ce qui laisse de la place pour apporter des contributions innovatrices.

CHAPITRE 2. COMMUNICATION SÉRIELLE TOLÉRANTE AUX PANNES

2.1 Introduction

Le système de communication sérielle est l'infrastructure logicielle et matérielle permettant la communication bidirectionnelle par flux de bits sériels entre le WaferIC et le logiciel embarqué sur l'ordinateur de l'utilisateur. Le système de communication sérielle qui a été choisi pour la conception du circuit est compatible avec le protocole "JTAG" [28]. La logique qui le supporte est répétée de manière uniforme sur toutes les cellules du WaferIC pour respecter la contrainte de design qui exige que chaque cellule soit autant que possible uniforme. Un module de contrôle logique uniforme doit donc supporter la communication sérielle bidirectionnelle entre les cellules et le logiciel. Chaque cellule doit être accessible le plus rapidement possible.

Le système de communication sérielle peut sembler à première vue un module plus ou moins important du WaferIC, mais il n'en est rien. Si le système de communication sérielle est défectueux, c'est tout le WaferIC qui n'est pas utilisable. Le WaferIC est destiné à être commercialisé, il est d'une importance capitale d'y ajouter une grande tolérance aux pannes pour obtenir le rendement et la robustesse désirée. Le coût de fabrication d'un WaferIC est assez important (dépendant des options technologiques, son coût de production brut peut se situer entre 2000 et 10000$ lorsque manufacturé en volume), car il s'étend sur toute la surface de la tranche de silicium. Il serait inapproprié d'être obligé d'exclure fréquemment de la chaîne de production un WaferIC entier de plusieurs milliers de dollars à cause d'une seule défectuosité dans le circuit qui couvre une tranche de silicium complète.

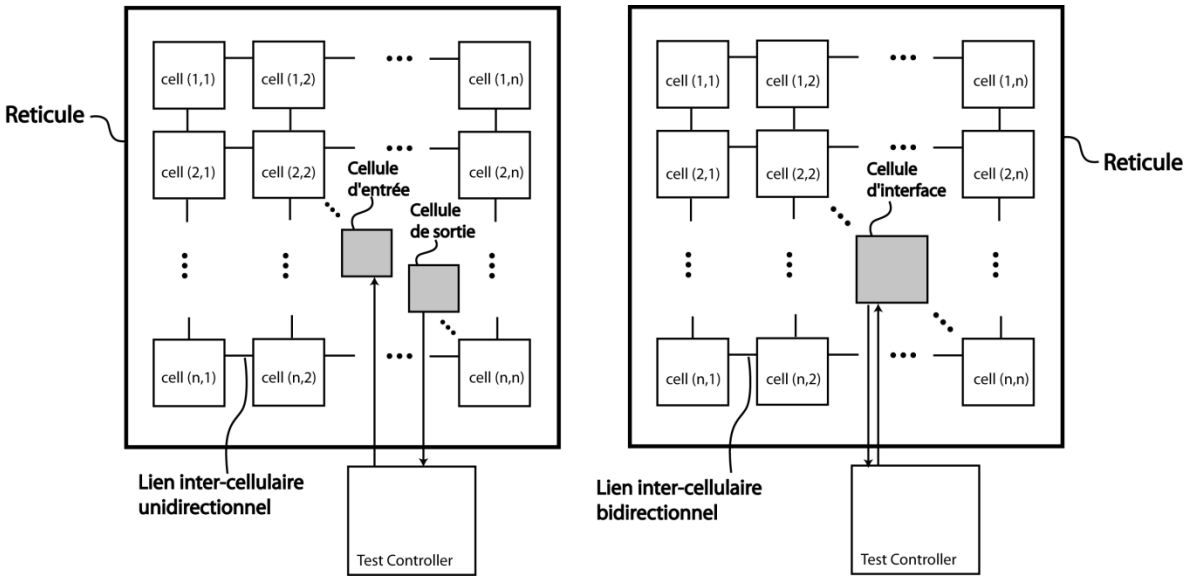
Le système de communication sérielle doit être conçu pour rediriger aux bons endroits les commandes provenant du logiciel. En effet, le WaferIC étant un circuit actif, mais il est sans intelligence propre. Ainsi, le WaferIC ne contient pas de CPU et possède peu ou pas d'*intelligence* locale pour minimiser la surface nécessaire au contrôle logique. En somme, il

ne fait que répondre passivement aux commandes numériques issues du logiciel. Par contre, les commandes doivent être acheminées par un système de communication bidirectionnelle. Chacune de ces commandes peut être catégorisée dans l'un des 5 types principaux d'opérations qui seront détaillées aux chapitres 4 et 5.

Ce chapitre a pour objectif de décrire les stratégies utilisées pour rendre tolérant aux défectuosités le système de communication du WaferIC. Ces systèmes sont logiquement indépendants et "orthogonaux" et ils peuvent être combinés pour obtenir un système complet et optimal en terme de nombre de ressources logiques nécessaires pour les réaliser. Le choix de la solution optimale finale pour l'implantation sera le sujet du chapitre 4 sur l'implantation matérielle.

La solution proposée dans ce document peut s'appliquer non seulement au WaferIC, mais aussi à tout type de PCB ou MCM complexe. D'une manière générale, l'architecture de communication qui relie le module de contrôle (le logiciel) et le WaferIC est montrée à la figure 2-1. Dans ce schéma, deux types de systèmes de communication sont montrés : le premier à gauche s'appelle le système UCIC pour Unidirectional Configurable InterCellular. Le deuxième à droite s'appelle BCIC où B signifie "bidirectional". Dans les deux systèmes, les cellules font partie d'un système plus large (le réticule dans le cas du WaferIC). Le contrôleur de test est montré aussi sur la figure, ainsi que 2 types de cellules : (1) les cellules d'interface qui peuvent soit recevoir et/ou transmettre les données et (2) les cellules internes qui ne sont pas en contact direct avec l'extérieur du circuit. Dans le cas du WaferIC, les cellules d'interface peuvent être situées à n'importe quel endroit dans le réticule, mais préférablement, elles se retrouvent au centre pour maximiser la tolérance aux pannes. Le schéma 2-1 montre des liens intercellulaires sur 2 réticules. Ces liens peuvent être (unidirectionnels ou bidirectionnels). Les deux possibilités qui sont caractérisées par des coûts et des robustesses associés différentes seront analysées dans ce chapitre. Les structures fonctionnelles qui en résulte sont activées dynamiquement durant les phases de

test, diagnostic et programmation. Seulement la méthode unidirectionnelle (UCIC) a été réalisée et testé (voir chapitre 4 et 5).



2-1 Schéma bloc niveau réticule pour des systèmes de communication uni/bidirectionnel

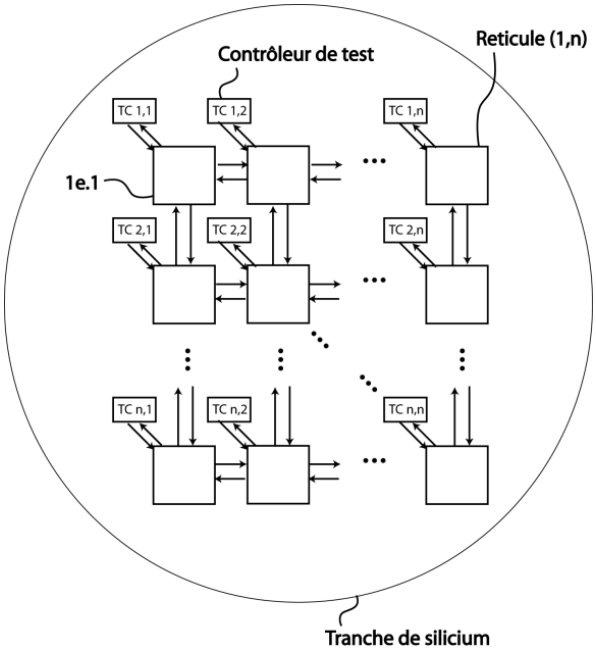
La figure 2-2 montre l'architecture de communication entre les réticules et le contrôleur de test. La figure a été dessinée de telle sorte qu'elle soit générique, pour tout type de communications entre un ensemble de réticules et un ou plusieurs contrôleurs de test. Les contrôleurs de test sont placés en communication parallèle. Cette capacité de traitement parallèle offre non seulement une meilleure vitesse de communication, mais une meilleure tolérance aux pannes. Les contrôleurs de test peuvent être implantés comme une unité indépendante embarquée sur le système ou bien être un module logiciel.

La méthodologie générale utilisée pour créer un système tolérant aux fautes est de commander par logiciel la création de liens inter-cellulaires dynamiques et reconfigurables permettant d'éviter les fautes dans le système de communication. Si une cellule du système est défectueuse, ou bien si seulement un seul des liens intercellulaires ne fonctionne plus, il

faut pouvoir le diagnostiquer par un test et ensuite éviter le lien en utilisant un chemin de contournement. La même stratégie doit être faite face aux cellules globalement défectueuses.

La clé qui permet la tolérance aux pannes vient de la *diagnosticabilité*, mais aussi du fait qu'il y a un nombre de liens et de cellules plus grand que nécessaire, ce qui permet de remplacer facilement une cellule par une autre si elle est défectueuse. Le défi consiste donc à créer une méthodologie de diagnostic et un système de liens reconfigurable qui occupe le moins d'espace possible.

Les liens UCIC ou BCIC montrés sur la figure 2-2 peuvent être dynamiquement activés en cas de défaillance d'un lien de communication. Par conséquent, si un lien de communication est défectueux, alors les réticules adjacents peuvent prendre le relais et rediriger l'information vers le réticule isolé. Cette redirection est contrôlée par logiciel. Plus de détails seront donnés dans ce chapitre sur les mécanismes de partage d'arbres d'horloge et de données entre réticules.

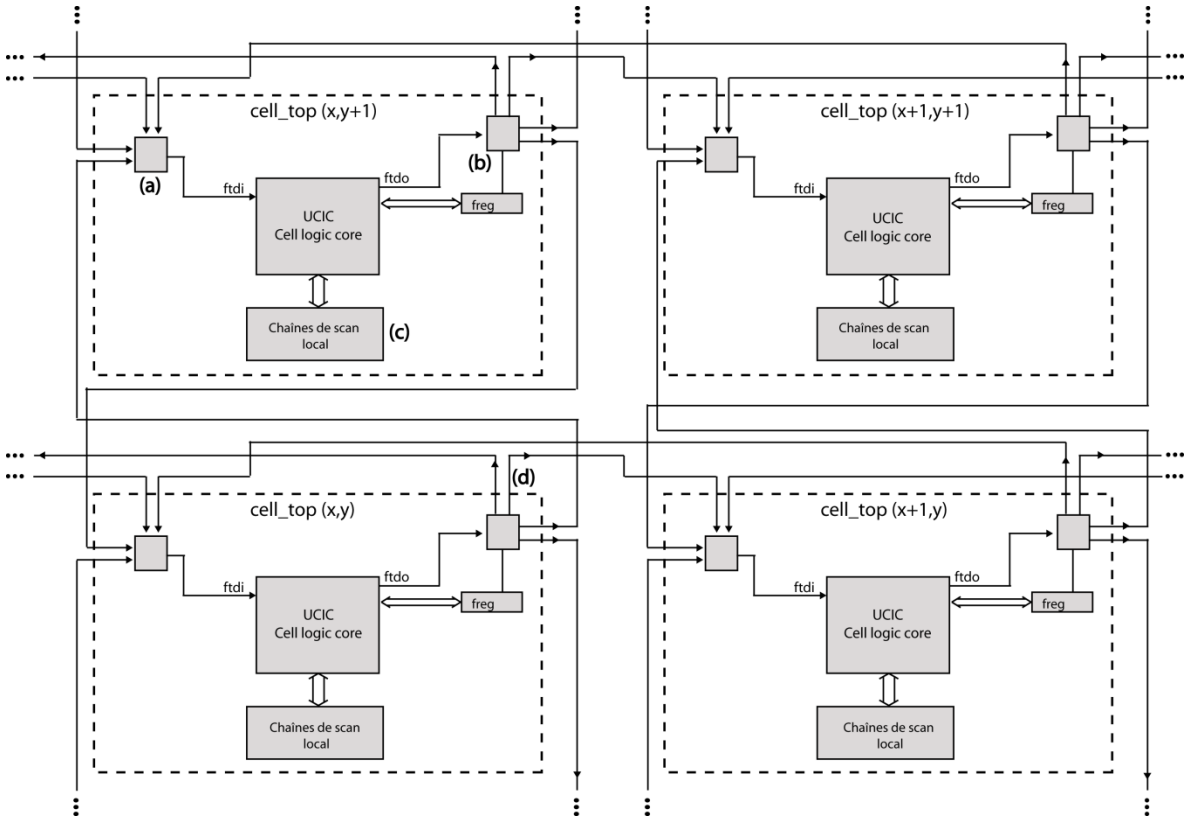


2-2 Vue globale du système de communication WaferIC avec les contrôleurs de test

Un contrôleur de test et un pont de communication indépendant sont associés à chaque réticule, car la communication doit être totalement indépendante d'un réticule à l'autre. Les contrôleurs de test peuvent être soit intégrés dans la tranche ou bien à l'extérieur par logiciel ou par un circuit numérique de contrôle dédié. De plus, un mécanisme de tolérance aux pannes permet aux réticules de communiquer avec leurs voisins nord, sud, est et ouest. Les images de réticule sont toujours imprimées sur une tranche de silicium comme une matrice ou un damier. Chaque case du damier doit être identique, car seulement un seul jeu de masques est utilisé par tranche de silicium pour imprimer des motifs. C'est pourquoi il est obligatoire d'avoir au moins un lien de communication par réticule. Chaque lien de communication doit être indépendant et doit aussi fournir une horloge et un signal de "reset" indépendant. En l'occurrence, le protocole JTAG contient un signal d'horloge (tck), un signal de reset (trst*), un signal de contrôle (tms) et deux signaux de données (tdi, tdo).

2.2 L'architecture à base de chemins inter-cellulaires reconfigurables unidirectionnels

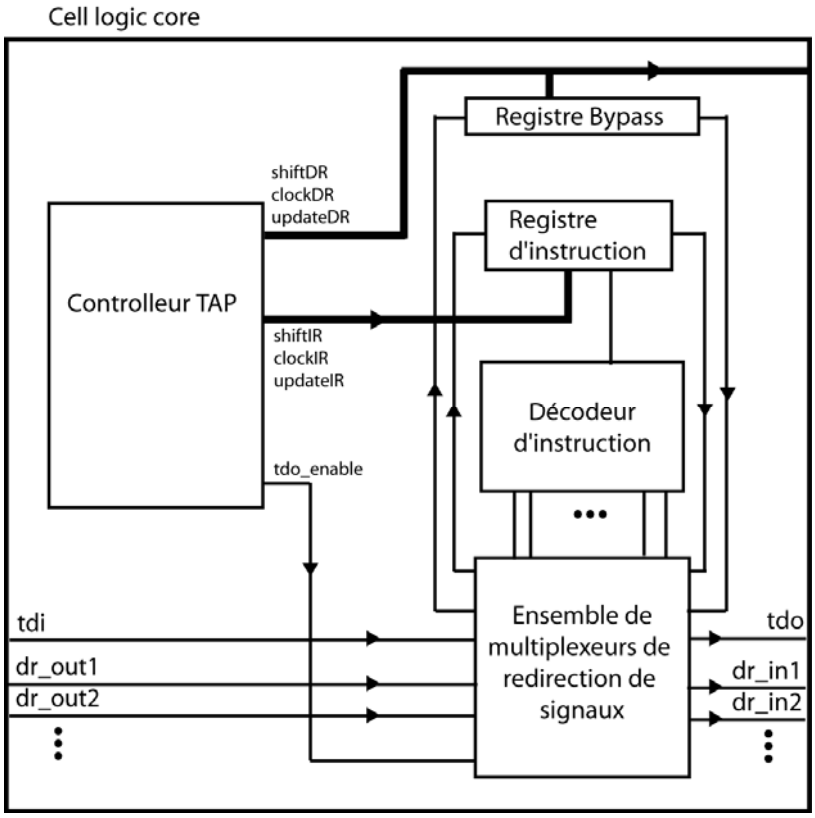
La figure 2-3 montre en détail l'architecture interne d'un système de lien "UCIC, Unidirectional Configurable Intercell Chain". Seulement quatre cellules voisines sont montrées pour illustrer le système d'interconnexion et pour montrer comment les cellules interagissent. Cette figure détaille seulement le système de communication sérielle pour la programmation et le test du système, mais non la fonctionnalité de base du système. La fonctionnalité de base du système n'est pas montrée pour simplifier le schéma. De plus, cela permet d'illustrer le caractère générique du système de communication. Ce système peut être utilisé sur le WaferIC, mais aussi sur tout type de systèmes massivement parallèles nécessitant un système de communications qui n'interfère pas avec le chemin de données principal. L'interaction avec la fonctionnalité principale du circuit se fait à l'aide des registres à décalage montrés sur la figure 2-3 (c). Par exemple, un système WSI peut contenir une matrice de CPU travaillant en parallèle. Dans ce cas, chaque CPU serait vu comme une cellule et pourrait avoir besoin d'un système de communications pour la configuration, le test et le diagnostic qui soit indépendant des CPU eux-mêmes.



2-3 Architecture du système de communication unidirectionnel intercellulaire UCIC.

Chaque cellule contient un sous-module appelé “cell logic core” contrôlant le flux de bits dans toutes les directions accessibles par la cellule. Ce module permet d’accéder aux registres spécifiques à l’application donnée ou bien permet de configurer le registre “freg” qui active le lien à la prochaine cellule à l’aide d’un multiplexeur(b) vers une cellule voisine en particulier (nord, sud, est, ouest). Par exemple, le lien (d) est un lien qui relie la cellule x,y à sa cellule voisine de l’est. La fonction du démultiplexeur d’entrée montré en (a) est de recevoir et de sélectionner les bons flux de bits provenant des cellules précédentes de la chaîne intercellulaire. De même, les multiplexeurs de sortie en (b) ont pour fonction de diriger le flux de bits vers une prochaine cellule à l’aide du registre freg. Soulignons un élément important : puisqu’il n’existe qu’un seul registre freg par cellule, il n’est pas

possible pour une chaîne de balayage de revenir sur elle-même. C'est pourquoi cette architecture est appelé UCIC, c'est-à-dire une architecture unidirectionnelle intercellulaire. Elle ne supporte qu'une seule direction de communication à la fois.



2-4 Architecture interne du "cell logic core" pour le système UCIC

La figure 2-4 illustre l'architecture interne du "cell logic core" (cœur logique) associée à l'architecture UCIC. Le cœur logique contient les modules JTAG standard, c'est-à-dire : (1) la machine à états "TAP controller", (2) un décodeur d'instruction et (3) un registre bypass. De plus, le cœur logique contient un ensemble de multiplexeurs ("register select mux") qui prend comme entrée les données sérielles (tdi, dr_out1, dr_out2, etc.) et les redirige vers la ligne tdo de la cellule. Le contrôleur TAP est contrôlé par un signal externe appelé "tms" provenant du port JTAG (voir annexe G). Le registre d'instruction permet d'activer à l'aide du décodeur d'instruction le mode désiré par l'utilisateur pour la cellule. L'activation du

mode permet d'activer le multiplexeur dans le module "Register Select Mux". Selon la norme JTAG, le registre bypass sert à passer directement à la prochaine cellule de la chaîne intercellulaire.

Le but des registres de configuration locaux comme ceux montrés à la figure 2-3 est de les utiliser comme capteur ou contrôleur pour le test interne des circuits ou pour la programmation "on-the-fly" du système. Une fois la chaîne intercellulaire mise en place, ces registres peuvent devenir accessibles de la même manière qu'un registre interne dans une chaîne en anneau (introduit au chapitre 1).

La figure 2-5 apporte des explications supplémentaires sur le mécanisme de création de chemins entre la cellule de départ et d'arrivée. Le but de la création de chemins est d'accéder aux registres "uReg". La figure illustre en 4 étapes (A,B,C,D) un exemple de création d'un chemin entre 4 cellules d'un réticule. Le même principe s'applique pour la création d'un chemin d'un nombre arbitraire de cellules. L'interface avec le contrôleur de test est montrée par un point gris pâle (tdi) ou foncé (tdo). Le point foncé est la cellule de départ qui transmet les données (tdo) vers le contrôleur de test. Le point pâle est en contact avec la cellule de départ qui reçoit et réachemine les données vers les cellules successives de la chaîne inter-cellulaire en cours de création. Chaque étape est associée à une série de commandes sérielles JTAG envoyées à la matrice de 2x2 cellules.

L'étape A consiste à accéder au registre "freg" qui détermine le chemin activé vers la prochaine cellule, en l'occurrence la cellule de l'ouest. L'étape suivante (B) montre qu'une fois que le registre "freg" de la cellule de départ active le lien vers l'ouest, il devient possible de configurer le même registre "freg" de la cellule suivante. La cellule suivante devient accessible à travers la cellule de tête qui est configurée en mode bypass. En C, le même processus se répète pour en arriver à accéder au port tdo situé dans la cellule

d’arrivée. L’étape C est atteinte une fois que toutes les cellules de la matrice 2x2 sont reliées, incluant la cellule en contact avec le contrôleur de test. À ce moment, il devient possible de voir apparaître un signal de sortie sur le port tdo de la cellule. C’est à ce moment qu’on a la preuve que le chemin inter-cellulaire fonctionne correctement. La dernière étape (D) de cet exemple montre qu’une fois un chemin intercellulaire créé, il devient alors possible d’accéder aux registres internes de configuration, de test, de programmation en mode anneau (“daisy chain”).

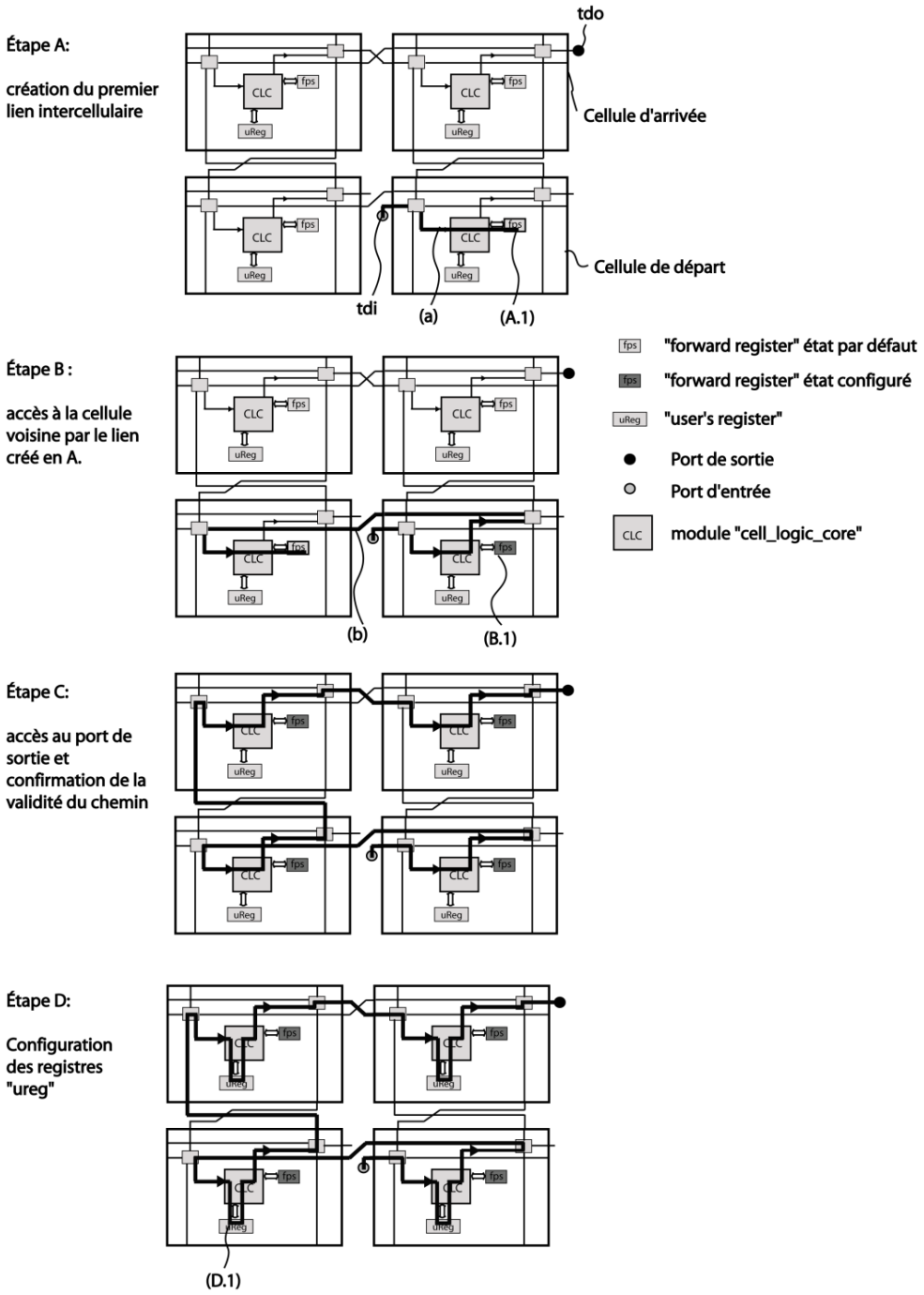


Figure 2-5 Exemple de création d'une chaîne de balayage intercellulaire

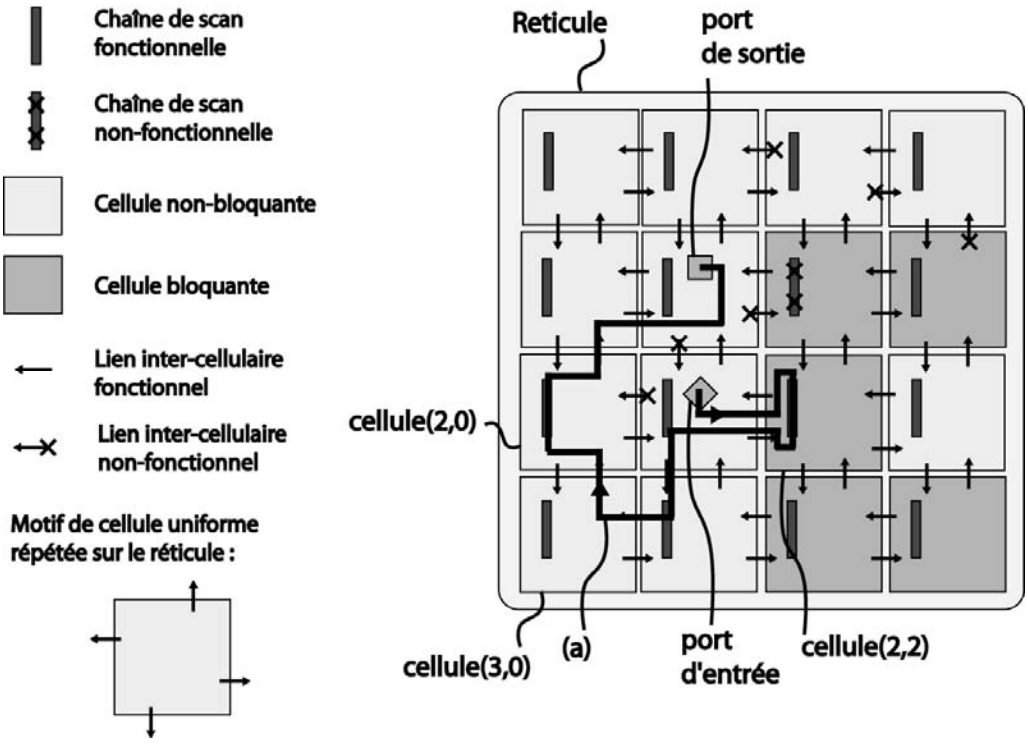
2.2.1 La tolérance aux pannes du système UCIC

Cette section montre les stratégies de tolérance aux pannes applicables avec des liens intercellulaires créés dynamiquement par configuration. La première stratégie est l'utilisation d'un contrôle externe tel que vu par R. Norman [27]. Ce concept a déjà été expliqué à la figure 1- du chapitre 1 de ce document. Le contrôle externe a la capacité d'accéder aux chaînes de scan d'une cellule à partir d'une cellule voisine. Il devient donc possible de contrôler l'état d'une chaîne de balayage d'une cellule défectueuse à partir des cellules voisines qui fonctionne bien. Un exemple de contrôle externe est montré à la figure 2-6. En effet, la cellule (2,2) n'est pas accessible directement, car c'est une cellule bloquante. Une cellule est bloquante si cette cellule empêche la création d'une chaîne de balayage intercellulaire peut importe la cause de se blocage. Du point de vue du logiciel qui gère le diagnostique, une cellule bloquante est une cellule qui peut laisser passer un signal du WaferNet, mais qui empêche la création d'une chaîne intercellulaire. Une cellule défectueuse est une cellule qui contient une défaillance logique dans le WaferNet ou dans le système de configuration.

Par exemple, puisque les registres de la cellule (2,2) sont fonctionnels, il devient possible d'y accéder à partir d'une cellule voisine, en l'occurrence, la cellule (2,1). Bref, avec le concept de contrôle externe, les registres peuvent encore être accessibles même si le centre de contrôle de la cellule est défectueux.

La deuxième stratégie est la stratégie d'évitement qui consiste à éviter d'utiliser des cellules bloquantes ou des liens défectueux dans le circuit. Cette méthode est plus complexe, car elle implique un diagnostic préalable du circuit. Pour éviter un lien défaillant, il faut connaître préalablement son emplacement. La figure 2-6 montre un exemple particulier d'évitement où le but est de configurer la cellule (2,0) et la cellule(2,2). Les cellules (2,3) et (3,2) ont par exemple une couleur plus foncée que les cellules (0,2) ou (0,0). La teinte

foncée exprime l'idée que le cœur logique de la cellule est dysfonctionnel donc ces cellules bloquent les flux de bits qui essaient de la traverser d'où le nom "cellule bloquante". Dans chaque cellule, il y a une chaîne de balayage qui est représentée avec un rectangle vertical. Si aucun "X" n'est superposé au registre, alors le registre est accessible pour le test ou pour la programmation. Le "X" superposé sur le registre exprime l'idée que la chaîne de balayage interne est dysfonctionnelle. Les liens unidirectionnels reliant les cellules sont représentés par une flèche. Certaines flèches sont des liens fonctionnels et d'autres flèches ont un "X" pour exprimer l'idée que ces liens sont défectueux. Une chaîne de balayage intercellulaire est montrée en 2-6 (a). Cette chaîne commence à l'interface avec le contrôleur de test (tdi) et elle finit à la cellule qui transmet les données (tdo) vers le contrôleur de test.



2-6 Les capacités de tolérance aux fautes pour l'architecture UCIC

À remarquer que toutes les cellules peuvent mettre leur système de communication dans 4 états distincts :

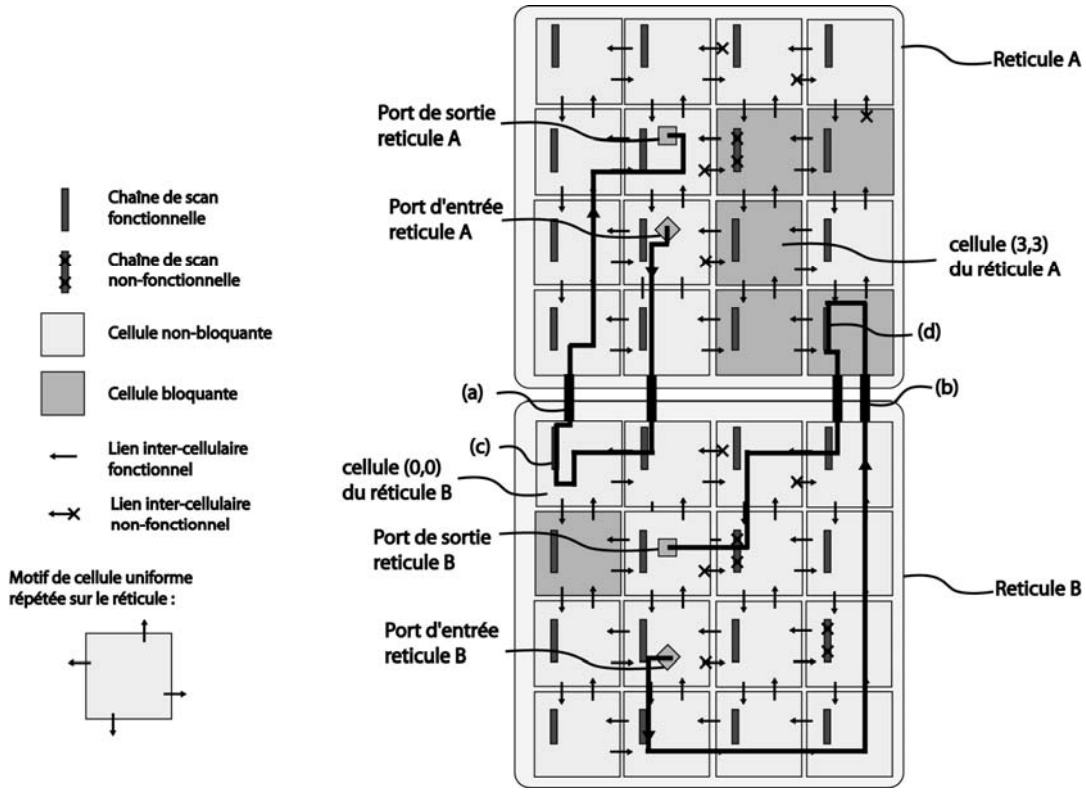
1- état inactif, la cellule ne participe pas à la communication dans la matrice de cellules (par exemple la cellule (0,0) de la figure 2-6).
2- l'état "bypass" où aucun accès aux registres chaînés de la cellule n'est fait, les données sont seulement redirigées vers la prochaine cellule (ex : la cellule (3,0) de la figure 2-6).
3- l'état "scan in" où un décalage est fait vers un ensemble de registres internes de la cellule (ex : la cellule (2,0) de la figure 2-6).
4- l'état "external scan" où la cellule actuelle envoie à une cellule voisine les données destinées à ses registres à décalage (ex : la cellule (2,1) de la figure 2-6).

2.2.2 Les liens inter-réticulaires et la tolérance aux pannes de UCIC

Si la cellule d'entrée ou bien la cellule de sortie de la figure 2-6 sont dysfonctionnelles, alors le système entier devient inopérant. De plus, une panne majeure arrive si tous les liens reliant la cellule d'entrée ou la cellule de sortie avec leurs cellules voisines sont dysfonctionnels. Pour passer à travers cette vulnérabilité, une autre fonctionnalité doit être ajoutée au système.

La figure 2-7 est une version plus détaillée de la figure 2-2 où un ensemble de réticules dans un système WSI est montré avec tous les réticules qui ont leur propre accès indépendant au contrôleur de test. La figure 2-7 montre en détail comment il est possible de faire communiquer deux réticules d'une tranche de silicium. Dans cette section, il est pris pour acquis que le procédé de microfabrication permet des interconnexions entre réticules, par exemple grâce à la technique de "*interreticle stitching*". La technique de "*interreticle stitching*" consiste à interrelier les réticules voisins par une couche de métal directement définie à partir des masques de photolithographie. La figure 2-7 montre avec les mêmes

symboles pictographiques, les 2 mêmes méthodes de tolérance aux pannes s'appliquent à des liens de communications inter-réticulaires. Dans cette figure, il y a deux réticules avec chacun 4x4 cellules (4x4), dont une cellule d'entrée et une cellule de sortie. De plus, chaque réticule a déjà configuré et activé sa propre chaîne de cellules inter-cellulaires.



2-7 Exemple de communication sérielle entre deux réticules voisins

Le but de la figure 2-7 est de montrer comment il est possible d'arriver à configurer les registres des cellules (3,3) du réticule A et la cellule (0,0) du réticule B. Une première stratégie pour accéder à la cellule (0,0) du réticule B est de créer un chemin inter-réticulaire partant de la cellule d'entrée du réticule A. Pour ce faire, il faut utiliser les liens inter-réticulaires indiqués par l'étiquette (a) . On peut voir que même si la cellule (0,0) du réticule B est fonctionnelle, il est impossible de l'atteindre avec un chemin intercellulaire à partir du réticule B.

Une autre méthode consiste à ajouter des liens inter-réticulaires dédiés à la gestion du contrôle externe. La figure 2-7 montre aussi qu'il est possible d'atteindre la cellule (3,3) (coin sud-est) du réticule A en utilisant le contrôle externe (b).

Logiquement, les liens configurables inter-réticulaires existent seulement sur la périphérie du réticule en contact direct avec d'autres réticules voisins. Il est possible d'utiliser ces liens de plusieurs façons. On peut entrer dans un réticule et sortir les données sérielles par un autre réticule. On peut aussi aller parcourir un certain nombre de cellules d'un réticule voisin, mais faire entrer et sortir les informations sérielles par le même réticule.

Il y a deux principales différences entre les liens inter-cellulaires et les liens inter-réticulaires. Premièrement, les liens inter-réticulaires se font entre réticules, par définition. Or, ce qui commande la création de liens intercellulaires ou inter-réticulaires est un logiciel de contrôle. Le logiciel de contrôle doit différencier ces deux types de liens, car ils sont gérés différemment. Pour un lien intercellulaire, il suffit d'un seul flux de bits JTAG et pour un lien inter-réticulaire, il faut 2 flux de bits JTAG. Par exemple, sur la figure 2-7, le tdi provient du réticule A et le tms, tck provient du réticule B. Par conséquent, lorsqu'une chaîne inter-réticulaire est créée entre deux réticules, le tdi est partagé par les 2 réticules, mais pas les autres signaux de contrôle tels que tck et tms. Cela peut donc engendrer des problèmes de synchronisation. Cette synchronisation doit être gérée à partir du contrôleur de test externe. Deuxièmement, au niveau physique, les liens intercellulaires peuvent être créés naturellement par la définition RTL d'un circuit, mais un lien inter-réticulaire n'est pas à la portée de toutes les usines de microfabrication. Il faut être capable de faire du *interreticle-stitching*.

2.2.3 Diagnostic des liens inter-cellulaires UCIC

Reconfigurer une chaîne inter-cellulaire ou inter-réticulaire n'est pas suffisant pour qu'un circuit puisse se remettre d'une défectuosité. Il faut aussi un système de diagnostic des pannes. Il existe une différence entre une panne et une défectuosité. Une défectuosité est un

problème physique. Une panne est généralement la conséquence d'une défectosité dans le circuit. Il existe plusieurs types de pannes et de modèles de pannes. Certaines défectuosités engendrent des pannes permanentes ou statiques tels que des collages (Stuck-At, SA) et des pannes dynamiques.

Le contrôleur de test joue un rôle primordial dans la détection et le diagnostic des pannes. Il doit avoir accès à des ressources logicielles permettant de gérer les séquences de test et d'en interpréter les résultats afin d'effectuer un diagnostic précis des liens inter-cellulaires défectueux. Il faut donc un algorithme de diagnostic. Un exemple d'application de ce diagnostic est montré à la figure 2-8. De plus la figure C-3 de l'annexe C présente l'organigramme d'un algorithme de diagnostic. L'algorithme est très général et efficace et il peut également être appliqué aux systèmes d'interconnexions UCIC, CICB.

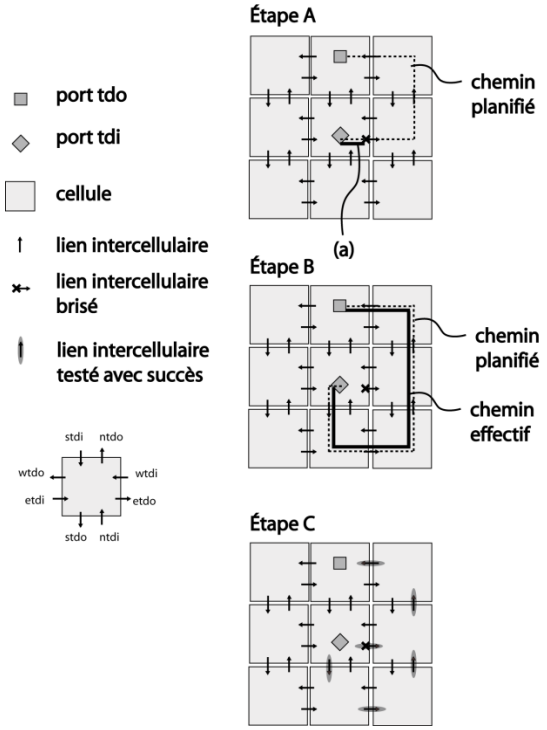


Figure 2-8 Exemple de diagnostic pour le système UCIC

Au début du processus de diagnostic, l'état des liens intercellulaires de toutes les cellules de la matrice est inconnu. La figure 2-8 montre un exemple d'application de l'algorithme de diagnostic pour une matrice de 3x3 cellules. La première étape A (illustrée à la figure 2-8) de l'algorithme consiste à planifier un chemin particulier entre la cellule d'entrée et de sortie. Le chemin planifié est montré avec une ligne pointillée. L'étape B du schéma 2-8 consiste à créer le chemin intercellulaire planifié. Le test consiste à créer le chemin et à envoyer des données dans ce chemin pour faire ressortir les données par le port tdo. Le contrôleur de test peut alors recevoir des données en provenance du WaferIC qui passeront par les cellules choisies. Or, puisqu'il y a une panne dans l'exemple sur un des liens du chemin planifié (indiqué par un X), le flux de données ne peut pas sortir tel que prévu. Donc, l'algorithme peut arriver à la conclusion par une règle d'inférence logique simple.

1- Soit il existe au moins un lien dysfonctionnel dans le chemin planifié;
2- Soit, un dysfonctionnement quelconque dans la circuiterie interne d'une cellule rend impossible la propagation du signal. Dans ce cas, le qualificatif "cellule bloquante" est donné à cette cellule. Rappel : une cellule bloquante peut potentiellement être configurée par contrôle externe, mais elle ne peut pas participer à la création d'un chemin intercellulaire.
3- Soit le lien JTAG est dysfonctionnel.

À cette étape, il n'est pas possible d'arriver à une conclusion finale sur l'état des liens ou des cellules. L'étape B (Fig. 2-8) montre une deuxième tentative où il n'y a aucun lien défectueux sur le chemin planifié, donc le signal arrive à se propager en sortie. De ce résultat, il est possible d'arriver à une série de conclusions. Ces conclusions découlent de l'application d'inférences logiques :

1- Les liens utilisés et choisis pour le chemin sont fonctionnels;
2- Le lien JTAG est fonctionnel;

3- Les cellules qui font partie du chemin sont fonctionnelles, donc elles sont non-bloquantes;
4- Connaissant les liens fonctionnels et les cellules fonctionnelles, il est possible d'arriver à la conclusion que le lien défectueux est responsable du résultat du test fait à l'étape A.

L'organigramme de la figure C.4 de l'annexe C montre comment il est possible de diagnostiquer tous les liens intercellulaires d'un réticule dans un cas général. La même méthode de diagnostic d'essai et erreur est utilisée pour tout le réticule. L'algorithme décrit dans ce mémoire n'a pas été optimisé.

Pour caractériser les liens et l'état de la cellule (bloquante ou non-bloquante) il faut appliquer des règles d'inférence. Le but des règles d'inférence est de décider si un lien ou une cellule sont fonctionnels ou pas. La première règle d'inférence peut être exprimée ainsi : "Si un signal se propage sur un chemin configuré et connu, alors toutes les cellules et les liens participant à ce chemin sont non-bloquants".

Une deuxième règle d'inférence est très importante. Cette règle d'inférence porte le nom de la règle d'inférence A. Elle se décrit par l'affirmation suivante : "Supposons qu'un chemin est composé de "n" cellules et de "n-1" liens. Le chemin est composé uniquement de cellules préalablement diagnostiquées comme non-bloquantes : $\{C_1, C_2, \ldots, C_n\}$. Le chemin est aussi composé d'un ensemble de liens préalablement diagnostiqués comme non-bloquants $\{L_1, L_2, \ldots, L_{n-2}\}$ et d'un autre ensemble de liens $\{U_1, U_2, \ldots, U_n\}$ dont l'état est indéterminé. Les liens de l'ensemble U sont les seuls liens non caractérisés du chemin. Si le signal arrive à se propager dans le chemin, alors l'état des liens de l'ensemble {U} est non bloquant, autrement (si le signal ne se propage pas) l'état des liens reste indéterminé."

Il est possible de caractériser l'état "bloquant ou non-bloquant" d'une cellule en utilisant la règle d'inférence suivante : "si une cellule possède 4 liens bloquants en entrée et 4 liens bloquants en sortie alors, c'est une cellule bloquante".

En appliquant ces 3 règles d'inférence, il est possible de diagnostiquer une matrice entière de cellules et les liens qui la constituent. D'autres règles d'inférence peuvent être appliquées pour accélérer le processus de diagnostic du système. De plus, la règle d'inférence A peut être appliquée de plusieurs façons. Par exemple, dans une situation où les pannes sont très rares, il est plus avantageux d'utiliser un ensemble {U} qui contient un grand nombre d'éléments. Dans le cas contraire où les pannes sont nombreuses, il est plus avantageux d'utiliser un ensemble U de petite taille.

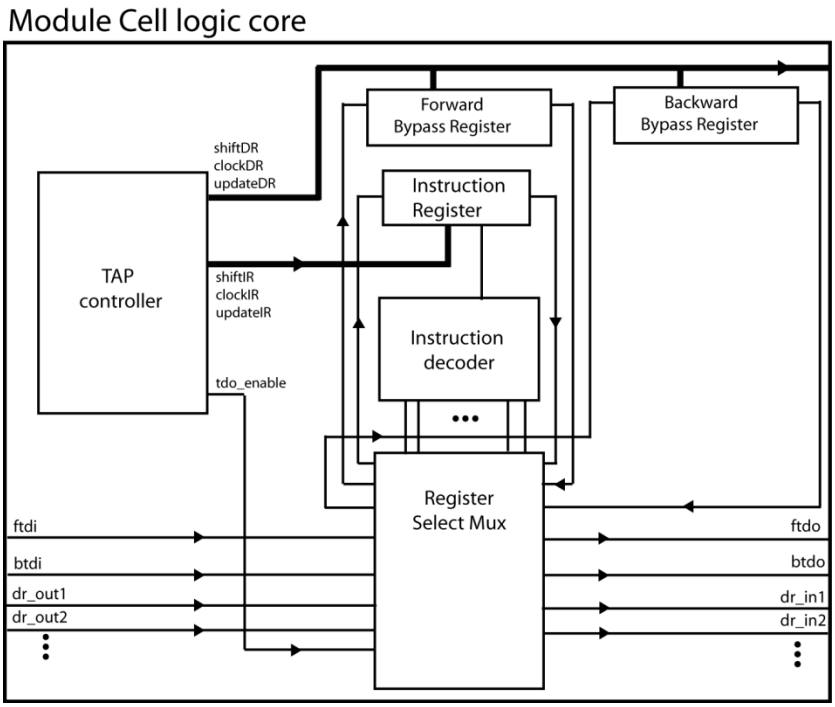
2.3 Variation : bidirectionnalité

Le second type d'architecture de liens inter-cellulaires reconfigurables proposé dans ce document est appelé : "bidirectional configurable inter-cellular (BCIC) links architecture". Cette architecture est légèrement différente de l'architecture unidirectionnelle de la précédente section du chapitre. La seule différence réside dans la capacité de communiquer dans les 2 directions pour créer un chemin intercellulaire qui puisse revenir sur ses pas en passant par le même chemin. La figure 2-1 montre des vues à haut niveau de ces architectures. On voit sur la figure 2-1 que chaque cellule peut gérer 2 liens, ainsi une seule cellule est nécessaire pour jouer le rôle de cellule d'entrée et de sortie.

La figure C.1 de l'annexe C montre en détail comment sont interconnectées les cellules dans une architecture BCIC. L'architecture BCIC possède les mêmes sous-modules qu'une architecture UCIC sauf qu'il y a une différence : il y a un multiplexeur configurable de plus (indiqué par (b)) qui redirige lui aussi les flux de bits. Ce multiplexeur configurable est associé à un registre de configuration appelé breg (pour "backward register"). De plus, un démultiplexeur (a) est ajouté par rapport au UCIC. Ces ajouts permettent de créer deux

chemins intercellulaires : un qui avance vers la prochaine cellule et un autre qui revient vers la cellule précédente.

La figure 2-9 montre en détail la structure interne du cœur logique de l'architecture BCIC. Le "*Tap controller*" est commandé par un signal tms JTAG standard. Par contre, plutôt que d'avoir un seul registre "*bypass*", il faut 2 registres nommés : "*forward bypass*" et "*backward bypass*". Ces 2 registres permettent de gérer deux flux de bits en même temps permettant la création de 2 chemins configurables dans une seule cellule. Cela a pour effet d'augmenter la tolérance aux pannes du système mais surtout de permettre à un chemin de revenir sur lui-même en passant pas la même cellule. Un nouvel ensemble d'instructions JTAG doit être inclus dans le décodeur pour gérer les cas où les flux de bits avant et arrière sont gérés en même temps. Les deux flux de bits peuvent utiliser les liens btdo (*backward* tdo), ftdo (*forward* tdo), btdi (backward tdi), ftdi (*forward* tdi) pour rediriger les données.



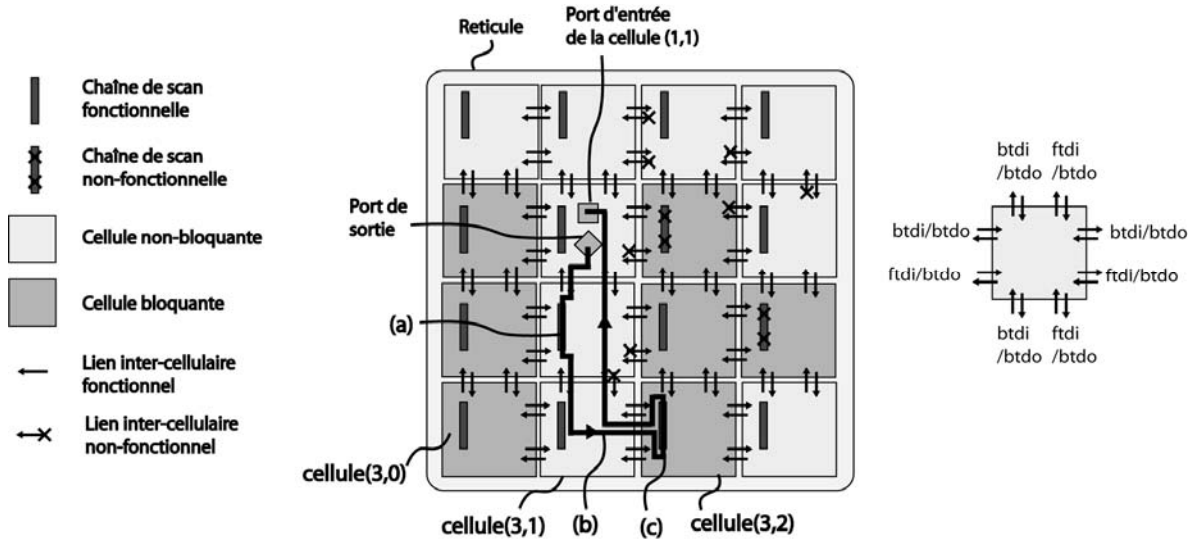
2-9 L'architecture interne du "cell logic core" pour le système BCIC

Un exemple impliquant un réticule de 4x4 cellules et permettant d'illustrer les capacités supérieures de la structure bidirectionnelle en termes de tolérance aux pannes est montré à la figure 2-10. Chaque cellule possède un jeu de 8 entrées et 8 sorties. Plusieurs fautes ont été ajoutées dans le réticule à la fois au niveau des liens bloquants et des cellules bloquantes. Le but est d'arriver à configurer un registre situé dans la cellule (2,1) et la (3,2). La cellule (2,1) est simplement configurée à l'aide de la création d'un chemin qui la traverse. L'état de la cellule est donc "*scan in*". La cellule (3,2) est configurée grâce à l'utilisation d'une cellule voisine (la cellule (3,1)) par contrôle externe.

L'architecture avec liens BCIC offre plusieurs avantages par rapport au UCIC :

1- Un plus grand nombre de liens entre les cellules signifie plus de chemins potentiels pour éviter les liens bloquants ou les cellules bloquantes. Un exemple de cet accroissement de la robustesse est montré à la figure 2-10 où il est possible de créer un chemin qui revient sur lui-même et qui passe entre des cellules bloquantes. Le chemin intercellulaire débute à la cellule (1,1) et revient à la même cellule. Ce chemin aurait été impossible à créer avec une architecture UCIC avec comme conséquence que les cellules (2,0), (3,0), (3,1), (2,2,), (3,2) ne sont pas configurables, donc dysfonctionnelles.
2- Les algorithmes de diagnostic sont plus simples à coder sur un logiciel. Il est toujours possible de revenir sur des cellules déjà visitées par un chemin, les algorithmes de routage sont plus faciles et directs à implanter sur ordinateur. Il devient donc possible d'explorer les liens d'un réticule une cellule à la fois.

D'un autre côté, un inconvénient évident de l'architecture BCIC est qu'elle utilise un plus grand nombre de ressources matérielles. Pour des systèmes très denses composés de petites cellules, cette méthode n'est pas recommandée.



2-10 Les capacités de tolérance aux fautes de l'architecture BCIC

La figure C.3 de l'annexe C montre comment il est facile et direct de procéder à un diagnostic complet d'une matrice de cellules. En quatre étapes, la figure illustre comment il est possible d'appliquer l'algorithme de diagnostic sur une petite matrice de 3x3 cellules. Le même algorithme de diagnostic que pour l'architecture UCIC s'applique pour l'architecture BCIC. Par conséquent, l'organigramme de la figure C.4 s'applique directement. L'étape A de la figure C.3 montre en pointillé le chemin planifié pour le test incluant un ensemble de liens non explorés (non testés). Ce chemin inclut un lien bloquant. Ainsi, à l'étape suivante (B), une fois le chemin créé, le signal ne peut pas atteindre le port de la cellule qui interface avec le contrôleur de test. S'il n'y a pas de signal qui sort par le port de sortie, alors au moins un des liens visités ou une des cellules utilisées par le chemin est bloquant. Ainsi, un deuxième chemin est planifié et créé pour le test en C sur la figure C.3. Cette fois-ci, le signal peut se propager à travers le chemin ce qui prouve que les liens visités fonctionnent. En (D), les résultats du test sont mémorisés dans une base de données et ils peuvent être utilisés pour effectuer des inférences logiques à partir des résultats d'autres tests.

2.4 Conclusion

Ce chapitre a porté sur le système de communication sérielle tolérant aux pannes pour les cellules du WaferIC. Les méthodes proposées ne sont pas seulement applicables aux besoins spécifiques du WaferIC, mais elles peuvent s'appliquer à tout système sur puce (SoC) ou système sur tranche (SoW) qui exploite un système de communication sérielle JTAG tolérant aux pannes. Deux architectures de communication ont été proposées dans ce chapitre. Les deux méthodes s'appellent respectivement l'architecture avec liens UCIC, c'est-à-dire "unidirectional configurable inter-cellular" et l'architecture avec liens BCIC (bidirectional configurable inter-cellular). Cette méthode permet de créer dans le WaferIC des liens configurables dynamiques en fonction des besoins de l'utilisateur et de la liste des pannes du circuit (*fault map*).

CHAPITRE 3. LE DIAGNOSTIC DU WAFERNET

3.1 Introduction

Dans ce chapitre, un ensemble de méthodologies sont proposées pour procéder efficacement au diagnostic des fautes dans le WaferNet. Ces méthodes ont été spécialement conçues pour les réseaux réguliers et reconfigurables se déployant sur la totalité de la surface active d'une tranche de silicium. Ce type de réseau est appelé dans ce mémoire RRNoC pour "Regular, Reconfigurable Network on Chip" ou bien RRNoW pour "Regular, Reconfigurable Network on Wafer". Les réseaux reconfigurables et réguliers sur la surface de la tranche de silicium contiennent inévitablement des fautes dues à la nature du procédé de microfabrication. Les fautes doivent être localisées autant que possible avec beaucoup de précision pour activer les chemins de données qui puissent éviter justement ces fautes et leur cône d'influence. Le cône d'influence est la zone où est propagée logiquement une faute. Tout signal inclus dans le cône d'influence est affecté par une faute distante.

Le contexte d'utilisation du test et du diagnostic proposé dans ce document diffère des méthodes habituelles. Normalement, en industrie, le test est utilisé pour éliminer le plus rapidement possible de la chaîne de production les "chips" défectueux pour réduire les coûts. Le diagnostic qui permet de localiser la source d'une faute est utilisé pour améliorer les processus de production et pour le débogage. Dans notre cas, le diagnostic est utilisé pour la tolérance aux fautes de circuits intégrés à l'échelle de la tranche comme le WaferNet. Le WaferNet est un réseau programmable utilisé pour interconnecter des puces déposées à la surface du WaferIC. Pour ce faire un algorithme de routage est utilisé. Ainsi, un diagnostic fait avec une bonne précision permet de générer une liste de fautes utilisable par des algorithmes de placement/routage pour éviter les fautes. Ce paradigme pour faire face aux défaillances des circuits se nomme "Defect-aware design flow" [18].

Il existe déjà une littérature extensive sur le diagnostic de court-circuit et de collage dans un réseau électrique [14][11][32]. Par contre, la structure interne du réseau du WaferIC nommé le WaferNet contient des différences clés qui justifient l'utilisation de nouvelles techniques de diagnostic qui diffèrent significativement du diagnostic de fautes dans le contexte des PCB. Les méthodes proposées arrivent non seulement à localiser les fautes, mais arrivent à optimiser la vitesse de diagnostic pour les réseaux très larges couvrant toute une gaufre de silicium, comme le WaferNet.

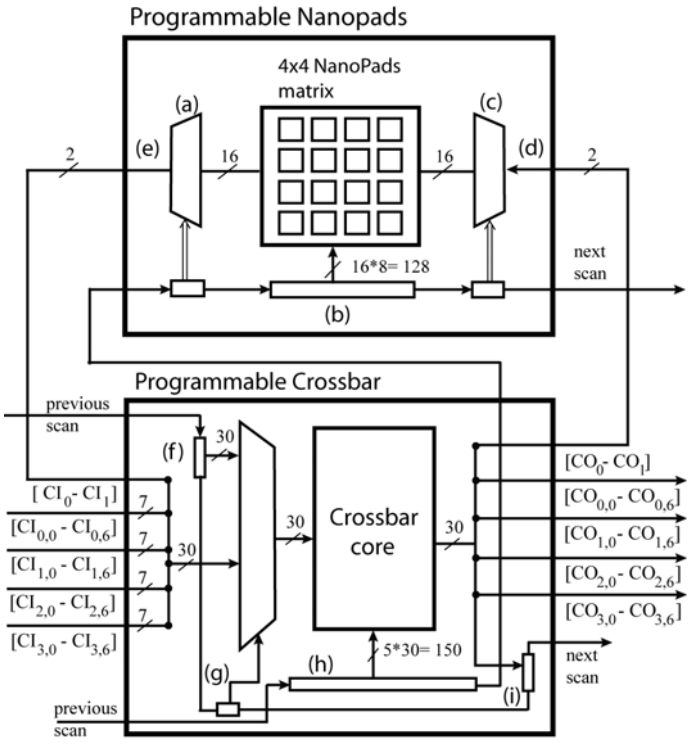
L'approche proposée utilise exclusivement un nombre limité de ports sériels JTAG et n'utilise pas les NanoPads comme points d'entrée et sortie. De plus, la méthodologie de diagnostic est à base de logiciel où les vecteurs de stimulation changent pour optimiser la recherche de défectuosités dans le réseau selon les résultats de test.

Ce chapitre est partitionné en 3 sections. La première section vise à détailler la structure interne du WaferNet, la deuxième section explique l'algorithme conçu par l'auteur de ce document et présenté à Newcas/Taisa [4] et finalement troisième section propose une méthode d'optimisation permettant d'améliorer la vitesse de diagnostic. Cette méthode d'optimisation fait l'objet d'une divulgation de brevet en cours d'approbation. D'autres méthodes de diagnostic ont été écrites par l'auteur dans cette divulgation, mais ces méthodes vont au-delà de la matière couverte par ce mémoire de maîtrise.

3.1.1 Description détaillée de la structure du WaferNet

Chaque cellule du WaferIC contient un crossbar programmable de 30x30 et une matrice de 4x4 NanoPads tels que montrés à la figure 3-1. Le crossbar a 7 liens entrants ($CI_{0,[0..6]}$, $CI_{1,[0..6]}$, $CI_{2,[0..6]}$, et $CI_{3,[0..6]}$ respectivement dans les directions N-E-S-O) et 7 liens sortants par côté de cellule pour un total de 4x7 = 28 entrées et sorties. En plus, il y a 2 liens entrants et 2 liens sortants qui proviennent de la matrice de NanoPads pour un total de 30 liens. Ces 2 liens servent à connecter les tampons d'entrée/sortie des NanoPads au réseau

WaferNet. La longueur des interconnexions augmente en suivant la suite géométrique (1,2,4,8,16,32,64) où la longueur est déterminée en fonction du nombre de cellules.

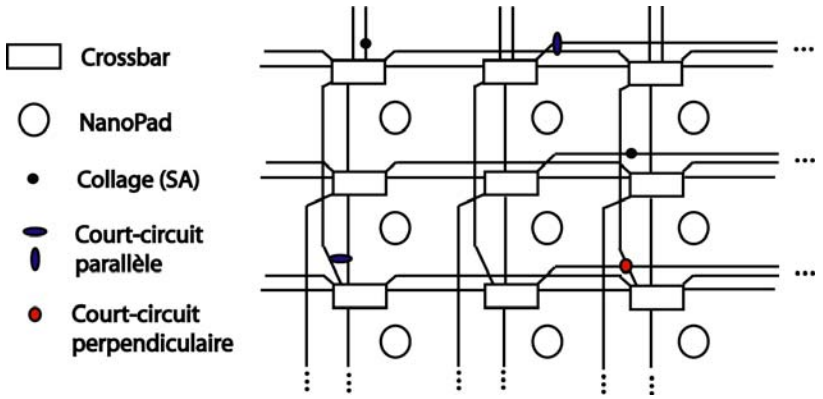


3-1 Schéma bloc de l'interaction NanoPads vs Crossbar.

Par conséquent, la principale fonction du WaferNet est d'interconnecter les circuits déposés à la surface du WaferIC pour émuler la topologie d'un circuit spécifié par l'utilisateur. Chaque cellule du WaferIC est interconnectée avec son voisinage immédiat dans les quatre directions (nord, sud, est, ouest). La figure 3-1 représente deux structures programmables : le crossbar et les NanoPads. Au plus, deux signaux peuvent être capturés sur les broches en contact avec le WaferIC et être redirigés vers le WaferNet.

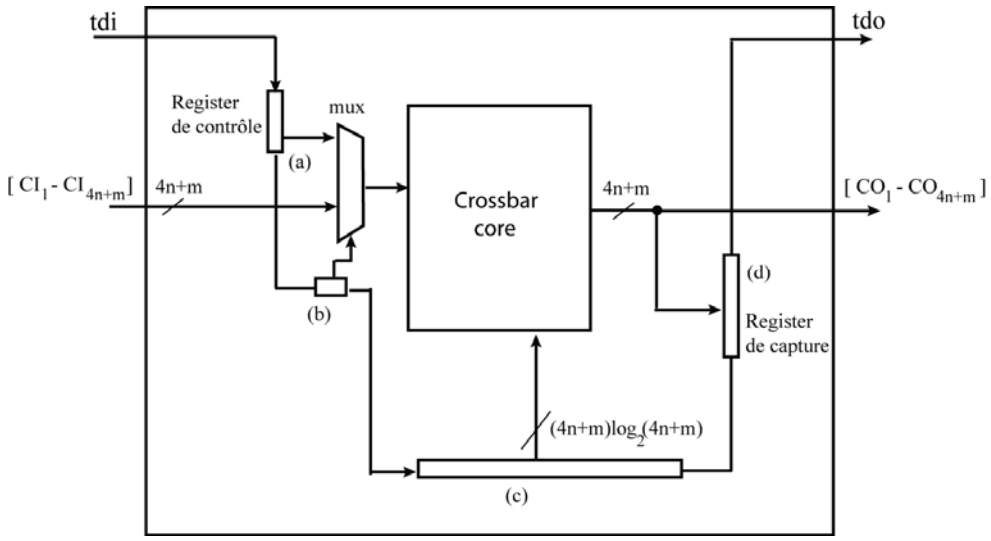
La figure 3-2 montre un exemple de fautes qui peuvent possiblement être observées dans le WaferNet. Les NanoPads sont représentés dans le schéma, mais ils ne sont pas considérés comme faisant partie du WaferNet. Par conséquent, seulement les fautes internes au

WaferNet sont détectées. Cela signifie que les algorithmes de diagnostic présentés dans ce chapitre ne couvrent pas la détection et la localisation de court-circuit entre un NanoPad et un fil du WaferNet. Cette situation, quoique improbable pourrait tout de même arriver. Les fautes de type bloqué à "stuck-at (SA)" sont représentées par un point noir et les courts-circuits par des ellipses touchant 2 fils. Parmi les types de court-circuit, il y a les courts-circuits entre 2 fils parallèles et les courts-circuits entre fils perpendiculaires.



3-2 Exemple de défauts dans le WaferNet.

La structure interne du crossbar tel qu'implanté pour le TestChip v1 est montrée à la figure 3-3. Cette structure interne comprend le noyau du crossbar ("crossbar core") entouré de ressources matérielles dédiées au test et au diagnostic. À la sortie du crossbar, il y un registre de capture de données de sortie du crossbar qui a la même largeur que le crossbar. À l'entrée du crossbar, un registre de contrôle et un multiplexeur configurable permettent de contrôler l'état de chaque entrée du crossbar. Le registre 3-3 (b) configure le mode "test" ou le mode "normal" et de ce fait permet de forcer des valeurs sur les entrées du crossbar. Le registre 3-3 (c) permet de configurer le crossbar dans un état particulier. Les registres à décalage (a,b,c,d) peuvent être chaînés ensemble ou faire partie d'une ou plusieurs chaînes de registres à décalage.

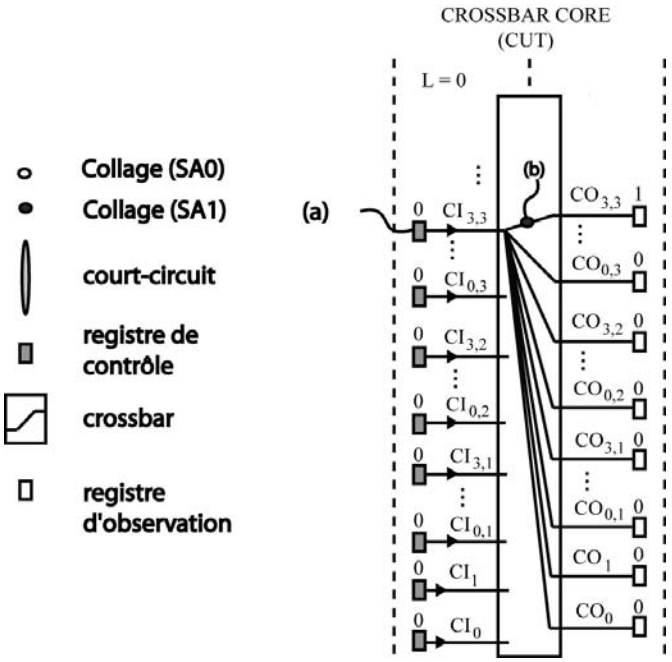


3-3 Architecture de test et de diagnostic pour le crossbar.

Il est important de clarifier les différences clés entre les algorithmes de détection classiques de courts-circuits dans le contexte des PCB et l'approche proposée dans ce mémoire. Dans le cas d'un circuit sur carte imprimée, la méthode classique utilise des registres "boundary scan" compatibles avec la norme IEEE 1149.1 dans chaque puce intégrée de la carte imprimée. Dans cette norme, chaque registre "boundary scan" est associé à une seule broche. Ces registres sont capables de capturer l'état logique appliqué sur la broche ou bien de forcer une valeur sur cette même broche. Ce sont des registres de lecture/écriture. Appliquer cette solution sur les entrées et sorties des crossbars du réseau est une solution trop coûteuse pour le WaferIC et peut aussi être une solution trop coûteuse pour tout autre type de réseau régulier de crossbar. L'approche suggérée dans ce document utilise seulement un seul registre de contrôle par port d'entrée du crossbar et un seul registre d'observation par sortie du crossbar, comme le montre la figure 3-3. Cette architecture a donc besoin d'algorithmes spéciaux de diagnostic qui sont plus complexes que les méthodes de diagnostic appliquées aux PCBs.

3.2 Description de l'algorithme "walking-one" appliqué au WaferNet

La méthode de test et de diagnostic "walking-one" permet de diagnostiquer les collages et les courts-circuits dans un réseau électrique en appliquant un vecteur de test qui contient une structure ressemblant à un '1' qui avance à travers le réseau. En appliquant un seul '1' logique sur une maille d'un réseau et en forçant un zéro sur toutes les autres mailles, il est possible de rendre observables des courts-circuits. Le même principe s'applique en utilisant un seul '0' forcé sur une seule maille du réseau. Ensuite, le '1' ou le '0' unique est appliqué à la prochaine maille du réseau à travers une chaîne de balayage d'où le nom "walking-one" ou bien "walking-sequence". La méthode proposée de diagnostic basée sur le "Walking one" adaptée pour le WaferNet s'applique en 3 phases. Un type de test spécifique est appliqué à chaque phase. Ces types sont simplement appelés : A, B et C. Le type A est illustré à la figure 3-4 et le type B à la figure 3-5.

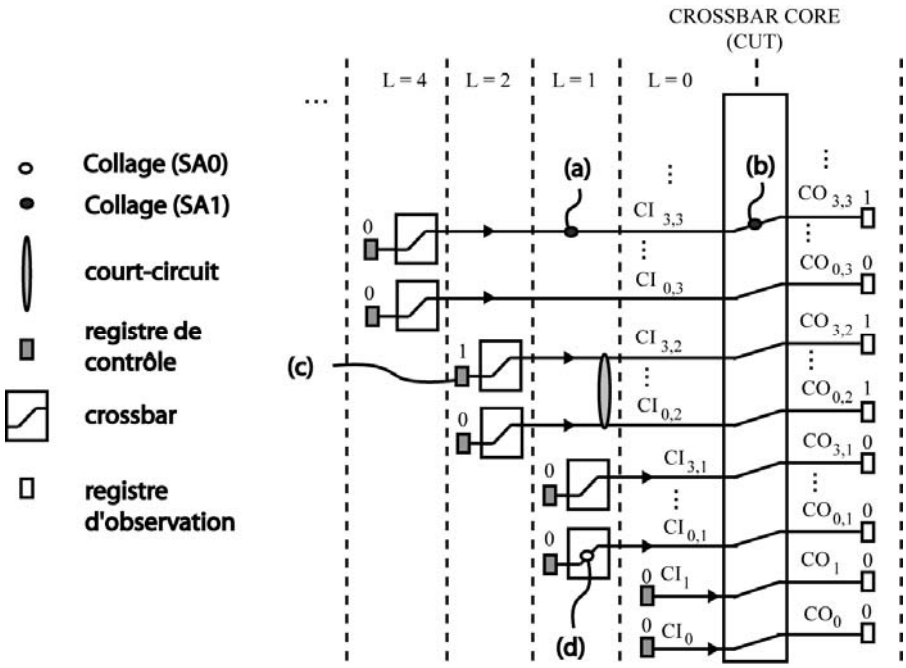


3-4 Diagnostic avec le test de type A.

Le type de test A : l'exemple à la figure 3-4 illustre un exemple de SA1 ("Stuck-At 1" ou collage à 1) identifié par un point noir (b). Ce schéma représente un crossbar configuré dans un mode particulier nommé "broadcast". C'est un test qui est appliqué très rapidement et directement, car il tire avantage des registres de contrôle locaux du crossbar (CUT, pour "crossbar under test") montré au centre de la figure et des registres d'observation locaux. La première étape du type de test A consiste à configurer un crossbar en mode "broadcast" pour la première entrée du crossbar et de configurer le crossbar en mode de test. Ensuite, en appliquant un 0 sur le registre de contrôle, il est possible de faire ressortir toutes les fautes SA1 vers les registres de capture. Le résultat du test est décalé vers la sortie et le prochain test en mode broadcast est appliqué. Le même test est répété pour toutes les entrées du crossbar. Un avantage de ce test est qu'il est relativement rapide. Il peut s'exécuter en parallèle et simultanément sur tous les crossbars du WaferIC. De plus, pour compléter le test, il faut refaire 2 fois le test sur chaque entrée en appliquant un 0 (pour détecter les SA1) et un 1 (pour détecter les SA0).

Le type de test B : un exemple de ce test est illustré à la figure 3-5. Cette figure illustre un crossbar sous test (CUT) et les autres crossbars distants reliés par des lignes de longues distances à ses entrées. Seulement un échantillon des liens du crossbar sont représentés pour simplifier le schéma. La même représentation logique et la même représentation formelle sont utilisées pour la prochaine figure 3-. De plus, une notation telle que L= (0,1,2,4,…) est utilisée pour différencier les interconnexions de longues distances entre elles. Par exemple, un lien de longueur L = 0 (les liens internes de la cellule) est représenté dans le schéma dans la zone délimitée à cet effet et les liens de longueur 4 sont représentés dans la zone L=4, etc. À la source de chaque lien est représenté un registre de contrôle où seront appliqués les vecteurs de la séquence de test et à la fin de chaque lien sont montrés les registres de capture qui peuvent échantillonner les données à la sortie du crossbar. Des exemples de fautes sont ajoutés au WaferNet : des fautes de type SA1 ont été ajoutées (a) et (b), une faute de type SA0 et une faute de type court-circuit entre la ligne $CI_{3,2}$ et $CI_{0,2}$. Le

test de type B peut faire ressortir les courts-circuits entre les liens inter-cellulaires du WaferNet. Les entrées du CUT ne sont pas reliées à n'importe quel crossbar distant. Chaque entrée est appariée avec le premier crossbar rencontré. Par exemple, le lien $CI_{3,3}$ est relié au crossbar situé à 4 cellules vers l'ouest.



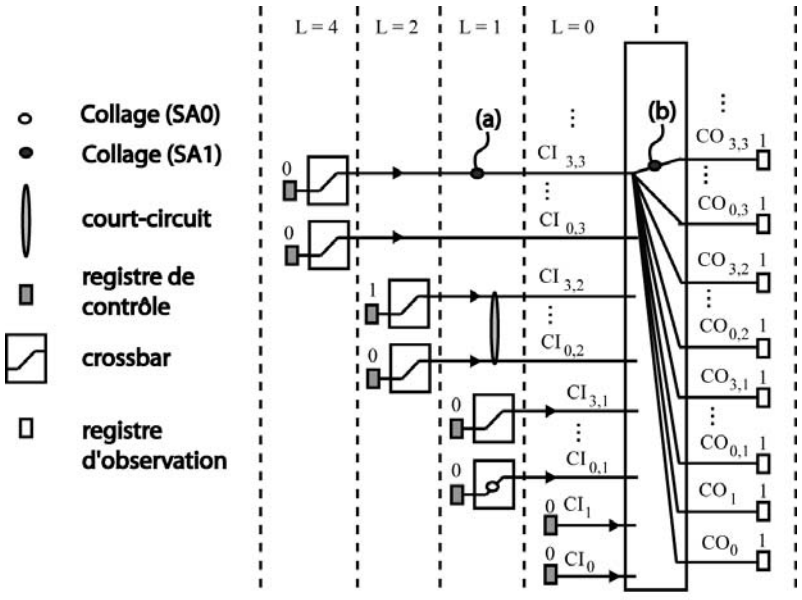
3-5 Diagnostic avec le test de type B.

Le test de type B consiste à appliquer le concept du "walking-one" à chaque registre de contrôle du WaferNet pour atteindre une couverture de test de 100% et une bonne précision dans le diagnostic des fautes. Chaque séquence de test appliquée à chaque registre de contrôle de chaque crossbar du WaferNet implique un décalage des données capturées en sortie du crossbar (les ports $CI_{x,y}$).

Notons que pendant l'application d'un '1' sur un registre de contrôle en particulier, il est important de forcer un '0' sur tous les autres registres de contrôle du WaferNet. Cette précaution permet un diagnostic sur chaque paire (perpendiculaire et/ou parallèle) d'interconnexions. Il est possible de rendre concurrent ce type de test. Également, plus de

détails seront donnés à la prochaine section sur les optimisations possibles. La figure 3-6 illustre l'effet sur les registres de capture. Par exemple, le '1' du "Walking-one" fait ressortir un court-circuit entre les interconnexions associées aux ports $CI_{3,2}$ et $CI_{0,2}$. Un dernier détail doit être ajouté : il est insuffisant de n'appliquer qu'une séquence de "Walking-one" sur chaque registre de contrôle du réseau. Il faut aussi appliquer la même séquence de "walking-zero" tout en forçant un '1' sur tous les autres registres de contrôle. Pour plus de détails sur l'algorithme permettant le diagnostic avec le test de type "B", il faut consulter le diagramme C.4 de l'annexe C.

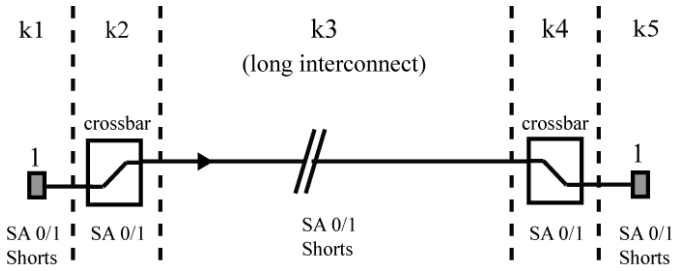
Le type de test C : la faute (b) aux figures 3-5 et 6 masque la détection d'une autre faute (a) dans les longues interconnexions du WaferNet. Le type de test C est nécessaire pour pouvoir localiser ce genre de fautes, illustré à la figure 3-. Les tests de type C s'appliquent donc à une étape subséquente au test de type B et s'appliquent seulement sur les interconnexions suspectées de contenir des fautes "masquées". Une cellule suspecte est une cellule où il a été détecté une faute SA1 ou SA0.



3-6 Diagnostic avec le test de type C.

Dans le test de type C, la même configuration en mode “broadcast” est utilisée, toutefois le registre de contrôle n’est pas situé localement, mais plutôt à l’entrée du crossbar distant qui est associé au lien sous test. Par exemple, la figure 3- illustre un cas particulier où un test est appliqué sur l’interconnexion sous test L=4 associée au port $CI_{3,3}$ du crossbar. Puisque le signal provient de l’extérieur de la cellule en provenance d’une cellule distante, une faute telle qu’un SA1 située dans les interconnexions L=4 doit se propager sur toutes les sorties. Les registres de capture peuvent donc enregistrer le résultat du test et décaler les résultats vers le contrôleur de test.

Pour récapituler, il existe plusieurs types de zones dans le réseau WaferNet. La figure 3- montre cette structure où les zones sont identifiées par k1 à k5. Ces zones définissent la résolution maximale du circuit en termes de localisation des fautes (diagnostic). Une analyse exhaustive de la résolution du diagnostic en présence de court-circuit multiple va au-delà de la portée de ce mémoire. Si la densité de fautes est suffisamment basse, par exemple pour des procédés de fabrication matures, la probabilité d’avoir des fautes multiples sur une même ligne devient aussi très basse. Dans une telle situation, la méthode proposée permet de diagnostiquer des fautes de court-circuit situées dans les zones k1, k3 et k5. Par contre, une fois un court-circuit identifié, il n’est pas possible de différencier entre des courts-circuits trouvés dans les zones k1, k3 et k5. Les fautes SA peuvent être facilement différenciées dans toutes les zones k1 à k5. Les zones k2 et k4 peuvent contenir des fautes de court-circuit, mais elles sont plus difficilement localisables.



3-7 La résolution du diagnostic.

3.3 Méthodes d'optimisation appliquées au diagnostic du WaferNet

En [4] nous avons proposé (cette référence se trouve aussi en annexe E) une méthode de test, à base de séquences de vecteurs ("walking-one"). Cette section propose donc des méthodes d'optimisation permettant d'accélérer par un facteur d'environ 50 la méthode initiale.

3.3.1 Modèle de performance du diagnostique du WaferNet

Pour pouvoir évaluer le facteur d'accélération d'un algorithme, il faut pouvoir le comparer à un algorithme de base qui est le même qu'à la référence [4]. Cet algorithme consiste à utiliser une chaîne de balayage multiple avec les tests de type A, B et C de la section 3.2. La structure des chaînes de test est montrée à la figure 3-3. Dans cette structure, il y a 3 chaînes qui entrent dans le crossbar. Pour l'algorithme de base, chacune des 3 chaînes (tci, tco et cr) sont chaînées de manière statique entre elles. Aucun chemin reconfigurable n'est possible. Dans le test de type B, il y a 3 sous-étapes de l'algorithme qui prennent relativement plus de temps à s'exécuter :

1- L'insertion du '1' dans les registres de contrôle du "walking-one".
2- La reconfiguration du crossbar à mesure que le diagnostic s'exécute.
3- L'extraction des résultats du test par décalage des registres de capture. Pour améliorer grandement la performance de l'algorithme de diagnostic, il faut améliorer cette étape.

Temps de diagnostic pour l'insertion : en tenant pour acquis que les registres de capture des crossbars soient tous chaînés en anneaux ("*daisy chained*"), l'ensemble des registres d'un réticule doivent être décalés pour chaque insertion de séquence de test. Ainsi, le temps d'insertion (T_i) d'une séquence pour le test de type B est donné par :

$$T_i = (n_c n_f)^2 T \tag{3.1}$$

où n_f est le nombre de cellules dans un réticule, n_c est la taille du registre de contrôle (le nombre de ports d'entrée des crossbars du réseau, $n_c = 30$) et T est la période de l'horloge (100ns @10MHz). Le facteur $n_c n_f$ est le nombre de cycles d'horloge nécessaire pour placer la séquence de test par décalage dans les registres de contrôle. Puisqu'il faut appliquer un vecteur de test pour chaque entrée de tous les crossbars du réticule, alors, il faut appliquer une puissance de 2 à la formule pour trouver le nombre total de cycles d'horloge et le temps total T_i pour l'insertion.

Temps de diagnostic pour la reconfiguration des crossbars : durant l'application des séquences de vecteurs de test, il faut changer la configuration du réseau à chaque fois qu'un nouveau crossbar sous test est rencontré. Il y a autant de crossbars sous test qu'il y a de cellules (n_f) et le nombre de registres de configuration pour chaque crossbar correspond à n_g. Ainsi, le temps de diagnostic pour la reconfiguration des crossbars est donné par :

$$T_r = (n_g n_f) n_f T \tag{3.2}$$

Temps de diagnostic pour l'extraction des données : Le même principe s'applique pour calculer le temps (T_e) pour extraire les résultats des tests de diagnostic. Par contre, il faut prendre en compte le nombre de réticules à inclure dans le cône d'influence pour éviter des interférences entre les tests appliqués sur le WaferIC. Les tests de type B peuvent être faits en concurrence, mais en gardant une distance correspondant à l'étendue du cône d'influence qui est défini en fonction du court-circuit possible entre 2 liens de grandeur maximale (L_{max}). Ainsi, le cône d'influence permet de définir une zone où 2 tests de type B peuvent être appliqués sans aucun risque d'interférence mutuelle. L'équation estimant le temps d'extraction des données du test peut s'exprimer par la formule suivante :

$$T_e = (2\frac{L_{max}}{n} + 1)^2 (n_o n_f)^2 T \tag{3.3}$$

où n_o est le nombre de bits dans le registre d'observation correspondant au nombre de sorties du crossbar et n_f est le nombre total de cellules dans le réticule. Le facteur $(2L_{max}/n+1)$ correspond à un facteur de ralentissement. Le diagnostic du réseau peut se faire de manière indépendante et concurrente sur plusieurs réticules du circuit à la fois. Mais, pour éviter les chevauchements et les interférences provenant des tests voisins, il faut définir une zone de sécurité où deux tests ne peuvent pas co-exister. Cette zone de sécurité vient ralentir le test, car elle affecte négativement la paralélisation. Cette zone de sécurité est définie en fonction du cône d'influence qui est un carré de plusieurs réticules, d'où la puissance de 2 appliqué à ce facteur de ralentissement. Par exemple, si un circuit est composé de réticule 32x32 (n=32) cellules et que L_{max} = 64, alors une zone de sécurité de 5x5 réticules doit être définie dans le circuit pour chaque réticule testé, ce qui ralenti le diagnostic d'un facteur 25.

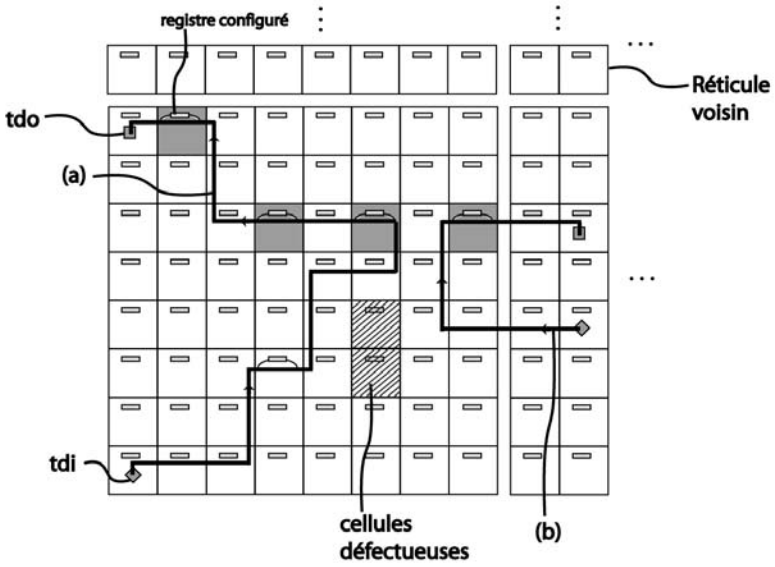
3.3.2 Optimisation de l'insertion de la séquence de test

La première stratégie d'optimisation est la plus facile à appliquer. Elle tire profit du caractère versatile et reconfigurable des chaînes de scan inter-cellulaires (UCIC et BCIC). Un exemple des capacités des chaînes de scan UCIC est montré à la figure 3-. Comme il a été expliqué au chapitre précédent, les chaînes de scan dynamiques sont reconfigurables, cela signifie qu'elles peuvent changer de chemin en configurant les cellules en en mode "bypass" ou "scan in". Le mode "bypass" ne fait que laisser passer les données et le mode "scan in" est le mode de configuration interne de la cellule. Par exemple, dans la figure 8, les cellules grises sont les cellules qui sont en cours de reconfiguration ("scan in") et les cellules blanches dans la chaîne sont en mode "bypass". Ainsi, changer la position du "walking one" dans un réseau de crossbars signifie changer dans le pire des cas l'état des registres de contrôle de 2 cellules. Au contraire, comme il a été spécifié pour l'architecture de référence de la section 3.3.1, s'il n'est pas possible de configurer les cellules en mode "bypass", alors il est obligatoire de décaler les données de toutes les

cellules du réticule pour changer l'état d'un seul bit. Donc, avec les chaînes inter-cellulaires (UCIC et BCIC), il suffit de changer l'état de l'instruction JTAG des cellules du réticule et de décaler les nouvelles données vers leurs destinations pour changer l'état d'un seul registre. L'équation 3.4 estime le temps nécessaire pour insérer les vecteurs de test de la séquence "walking-one".

$$(T_i)_{op} = (n_i\, n_f + n_f - 2 + 2n_c) n_f n_c T \tag{3.4}$$

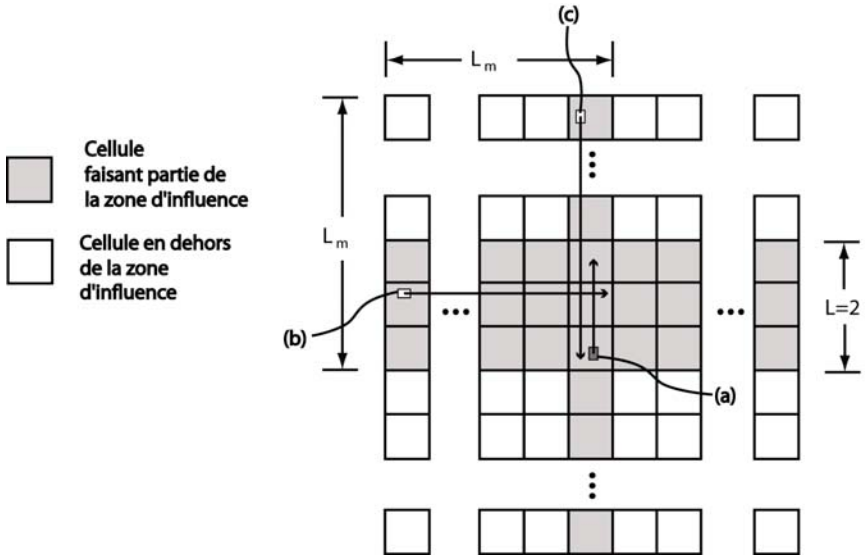
Le terme $(n_i\, n_f + n_f - 2 + 2n_c)$ décrit le temps nécessaire pour faire une seule insertion de données. Le terme $2n_c$ sert à calculer le pire cas où 2 reconfigurations de registre de contrôle a lieu à chaque insertion de nouveau vecteur de test. Le terme $n_i\, n_f$ est le nombre de bits nécessaire pour configurer les registres d'instruction JTAG et $n_f - 2$ est le nombre de registre bypass à traverser pour accéder aux cellules à reconfigurer. Puisqu'il faut tester successivement toutes les lignes d'interconnexion du réseau, un facteur de $n_f n_c$ est appliqué à l'équation 3.4.



3-8 Les capacités de la chaîne reconfigurable pour le diagnostic.

3.3.3 Optimisation de l'extraction des résultats du test

Décaler en sortie les données du test signifie que les données de tous les registres d'observation des crossbars doivent se décaler pour chaque test appliqué sur chaque entrée de crossbar. Cette étape peut être très coûteuse en temps pour de grands réseaux (voir eq. 3.3). Par conséquent, pour un réseau tel que le WaferNet, cette sous-étape doit être optimisée. Les mêmes principes de sélection de cellules pertinentes pour le diagnostic s'appliquent ici. Accélérer cette méthode consiste à utiliser le principe des cônes d'influence pour décaler seulement les données réellement utiles vers l'extérieur du circuit. Le figure 3-9montre un exemple de cône d'influence.



3-9 La dynamique de la zone d'influence des courts-circuits dans le WaferNet.

La figure 3-9 montre comment le cône d'influence se déploie sur la matrice de cellule du réticule. Les cases foncées indiquent les cellules du réticule incluses dans la zone d'influence du test appliqué en (a) tandis que les cases blanches indiquent celles en dehors de la zone d'influence. Une zone d'influence est la surface où il est logiquement possible de trouver une faute en fonction du test appliqué. Par exemple, la figure 3-9 montre un "walking one" appliqué sur un registre de contrôle (a) du WaferNet. Le registre de contrôle

est associé à un lien de longueur L=2 et est orienté vers le nord. La zone d'influence est la zone où peut se déployer un court-circuit entre l'interconnexion sous test et toutes les autres interconnexions qui recoupent perpendiculairement ou parallèlement l'interconnexion sous test. En assumant qu'il n'y pas de fautes multiples (plus d'un court-circuit sur une interconnexion), alors logiquement, il est impossible de retrouver une faute de court-circuit sur une ligne du WaferNet qui se trouve dans une des cellules blanches de la figure. Il faut donc aller interroger les registres des cellules foncées du schéma. Par exemple, les registres d'observation (b) et (c) font partie de la zone d'influence, mais à la périphérie générée par le registre de contrôle (a). Pour ce faire, la même technique de sélection des registres des cellules pertinentes est utilisée.

Chaque nouvelle application d'un vecteur "walking-one" amène la création d'une nouvelle zone d'influence. Par conséquent, un algorithme de diagnostic doit être en mesure de prédire avec précision la forme et l'extension de cette zone. La forme de croix de la zone d'influence telle que représentée sur la figure 3-9 est générique. Il suffit alors d'appliquer le même modèle pour toutes les interconnexions du WaferNet. En somme, la zone d'influence grandit avec la grandeur de l'interconnexion sous test et change d'orientation en fonction de l'orientation de l'interconnexion sous test.

L'équation générale exprimant une estimation du temps d'extraction des données provenant des cônes d'influence $(T_e)_{op}$ pour extraire les résultats du test de diagnostic correspond à l'équation suivante :

$$(T_e)_{op} = \sum_{k=1}^{m} (L_k L_m + 2L_m - L_k) n_o n_f T + n_i n_f n_c T \qquad (3.5)$$

où $m = 7$, $L_k = \{1, 2, 4, 8, 16, 32, 64\}$ pour k variant entre 1 et 7. L_m est la longueur du lien de grandeur maximale qui correspond à 64. La grandeur étant définie en termes de

nombre de cellule. Le premier terme de l'équation 3.5 représente la somme de toutes les extractions de données des registres inclus dans un cône d'influence où il faut extraire l'information. L'extraction des résultats du test n'est limitée qu'à la zone grise de la figure 3-9. Ainsi, il suffit de connaitre le nombre de cellule à visiter $(L_k L_m + 2L_m - L_k)$ et multiplier ce nombre par $n_o n_f T$ pour calculer le temps d'extraction associé à l'emplacement d'une seule séquence '1' ou '0' du "*walking-one*". Le deuxième terme de l'équation 3.5 représente le temps nécessaire pour changer l'état des registres d'instruction pour la configuration des cellules en mode "bypass" ou en mode "scan in".

3.3.4 Estimation du gain de performance

Le tableau suivant montre les gains de performance associés à l'application des méthodes d'optimisation suggérées dans ce chapitre. La variable n_f est le nombre de cellules dans un réticule.

Estimation du gain de performance.

n_f	T_i (s)	$(T_i)_{op}$ (s)	T_r (s)	$(T_r)_{op}$ (s)	C_f (s)	T_e (s)	$(T_e)_{op}$	T (s)	$(T)_{op}$	Facteur d'accélération
64	0.37	0.07	0.13	0.01	289	113.64	1.82	114.14	1.90	60.00
256	5.90	1.03	2.06	0.05	81	509.61	7.29	515.51	8.37	61.59
1024	94.37	15.91	32.93	0.59	25	2516.58	29.17	2610.95	45.67	57.17
4096	1509.95	252.40	526.80	8.65	9	14495.51	116.68	16005.46	377.72	42.37

La colonne « facteur d'accélération » du tableau montre que la complexité du temps de diagnostic de l'algorithme de diagnostic de base T et celui optimisé $(T)_{op}$ sont du même ordre de complexité quadratique $O(n^2)$. Par contre, le temps de diagnostic pour la version optimisée $(T)_{op}$ est plus court, car l'ordre de complexité est défini seulement en fonction de l'ordre du polynôme décrivant le temps maximal de l'algorithme, tel que défini à l'équation 3.5. L'algorithme optimisé ne fait que minimiser le temps pour extraire les données. Les données sont extraites en n'allant chercher que les informations nécessaires

dans le réticule, mais l'ordre de complexité d'extractions reste le même. Cela est équivalent à appliquer un facteur x << 1 sur le terme d'ordre $O(n^2)$. Ainsi, l'algorithme d'optimisation de change pas la nature quadratique du diagnostic.

3.4 Conclusion

Ce chapitre a traité du diagnostic des courts-circuits et des collages dans les interconnexions du WaferNet et de ses possibilités d'optimisation. Il a été traité de la "diagnosabilité" du WaferNet étant donné l'infrastructure de test qui a été ajoutée au crossbar. Ces algorithmes s'appliquent seulement aux collages (SA1 et SA0) et aux courts-circuits entre les interconnexions du réseau. Aucun diagnostic n'a été proposé en présence de fautes dynamiques telles que les fautes de délais, de "crosstalk", etc. De plus, l'importance du diagnostic a été soulignée pour améliorer la tolérance aux fautes dans le WaferNet. Il a été démontré qu'avec un système de chaîne de balayage inter-cellulaire configurable, il est possible d'optimiser le temps de diagnostic. De plus, la connaissance de la forme exacte et de l'extension de la zone d'influence des fautes est la base conceptuelle qui permet de dramatiquement augmenter la vitesse de diagnostic des courts-circuits du réseau. La méthode décrite dans ce chapitre peut être adaptée avantageusement à d'autres réseaux reconfigurables et réguliers sur puces (NoC) ou sur gaufres (NoW). Une version plus détaillée des méthodes d'optimisation montrées aux sections 3.3.2 et 3.3.3 fera l'objet d'une divulgation.

CHAPITRE 4. IMPLANTATION ET VÉRIFICATION DU SYSTÈME

Les chapitres 2 et 3 ont décrit des solutions possibles pour implanter un véritable système de configuration tolérant aux fautes. Parmi ces solutions, il a été démontré que le système UCIC est celui qui occupe la plus petite surface tout en offrant une tolérance aux fautes équivalente. C'est pourquoi le système UCIC a été choisi dans l'implantation du WaferIC. Ce chapitre détaille l'implantation de l'architecture RTL de la solution retenue. De plus, les détails d'implantation de l'environnement de vérification développé spécifiquement pour la partie numérique du WaferIC sont inclus en deuxième partie du chapitre.

4.1 Description fonctionnelle du WaferIC

Cette section vise à introduire et détailler les principaux modes d'opération du WaferIC. Ces modes d'opération (diagnostic, programmation, contrôle externe et détection de contact) sont essentiels à comprendre pour se faire une idée juste du système de communications du WaferIC.

4.1.1 Le système de configuration interne par le protocole JTAG

Un protocole de communication sérielle entre le WaferIC et la carte de contrôle est utilisé afin de minimiser le nombre de connexions. Le protocole JTAG a été choisi pour faire le test de circuit imprimé. Il a ensuite été étendu pour permettre la configuration interne de circuit électronique. Depuis, les FPGA et d'autres types de systèmes utilisent le protocole JTAG pour supporter la configuration par flux de données sérielles.

Dans les pages suivantes, les modifications apportées au protocole JTAG et au système de configuration interne pour permettre la tolérance aux fautes sont expliquées. Le système JTAG adapté pour le WaferIC supporte 4 modes :

1- **Mode diagnostic du WaferIC** Il localise les fautes pour en permette l'évitement (voir le chapitre 3).

2- **Mode de programmation du WaferNet** Ilaccède les registres "scan" (les registres accessibles par décalage) d'un ensemble de cellules pour configurer le WaferNet et pour interconnecter les billes des puces déposées sur le WaferIC.

3- **Mode de contrôle externe** Il prend en charge les fonctionnalités des cellules voisines. Cette fonctionnalité est utilisée comme stratégie pour la tolérance aux fautes et elle est illustrée aux figures 1- et 2-10. Chaque cellule contient un CLC (cell logic core) qui redirige les flux de données aux bons endroits dans le système. Chaque CLC peut diriger l'information vers sa propre cellule ou la rediriger vers l'extérieur, c'est-à-dire vers les cellules voisines du nord, sud, est, ouest. Ainsi avec ce mode, un CLC peut prendre en charge la chaîne "scan" d'un CLC défectueux voisin, donc peut prendre le contrôle des unités commandées par le registre de la chaîne "scan".

4- **Mode de détection de contacts** Un contact entre la surface du WaferIC et les broches d'un circuit intégré déposé sur le WaferIC est détectable par court-circuit puisque les broches sont composées de matériaux conducteurs d'électricité. Plus de détails sur ce mode d'opération seront ajoutés à la section 4.4.

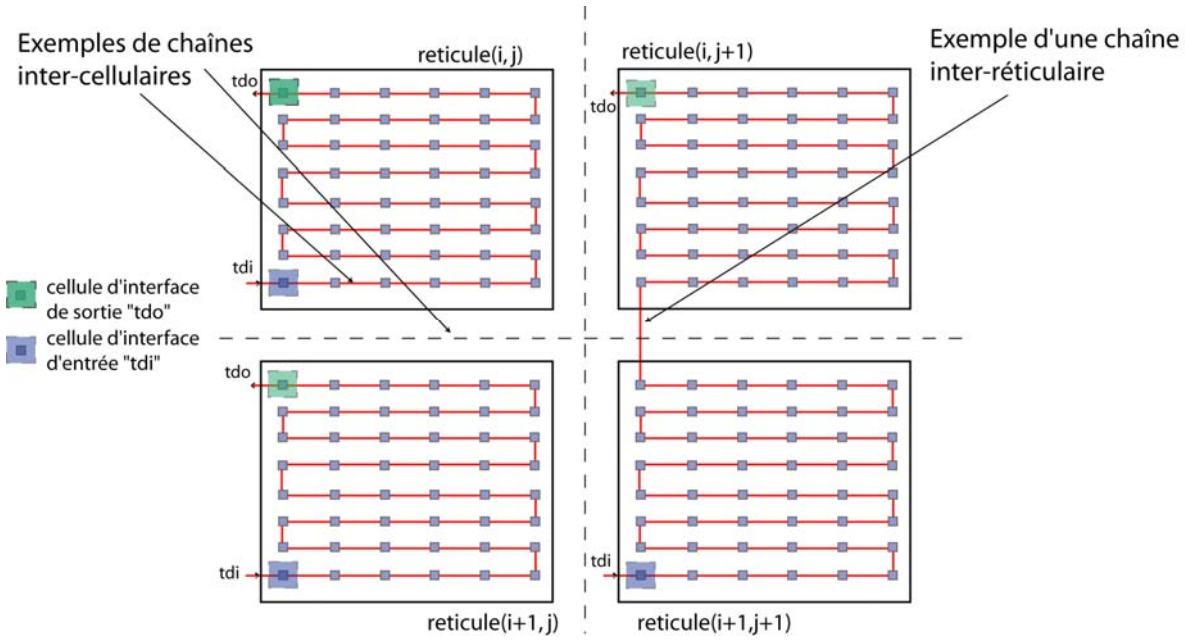
4.2 Description détaillée de l'architecture matérielle

Cette section a pour but de décrire la structure interne du WaferIC. Chaque niveau hiérarchique du WaferIC (niveau WaferIC, niveau réticule jusqu'au niveau de la structure interne RTL de la cellule) est aussi détaillé dans cette section.

4.2.1 Le niveau WaferIC

Le WaferIC est un quadrillage de réticules sur la totalité de la surface active de la gaufre de silicium. Il est bien connu que la fabrication d'un circuit intégré se fait à l'aide du principe de photo-répétition d'une même image de réticule sur une plaque de silicium. Cette image est répétée pour former un quadrillage. Les interconnexions en bordure du réticule sont connectées à celles des réticules voisins par une technique de *stitching* développée par Tower Semiconductor Inc. Ainsi, ce niveau hiérarchique doit être supporté. Cette description matérielle au niveau WaferIC n'est pas faite pour être synthétisée, car elle transcende le réticule qui est le plus haut niveau d'abstraction synthétisable. Il y a beaucoup de fonctionnalités à valider et vérifier au niveau WaferIC. La figure 4-1 montre les fonctions qui nécessitent plus d'un réticule pour être implantées dans le silicium.

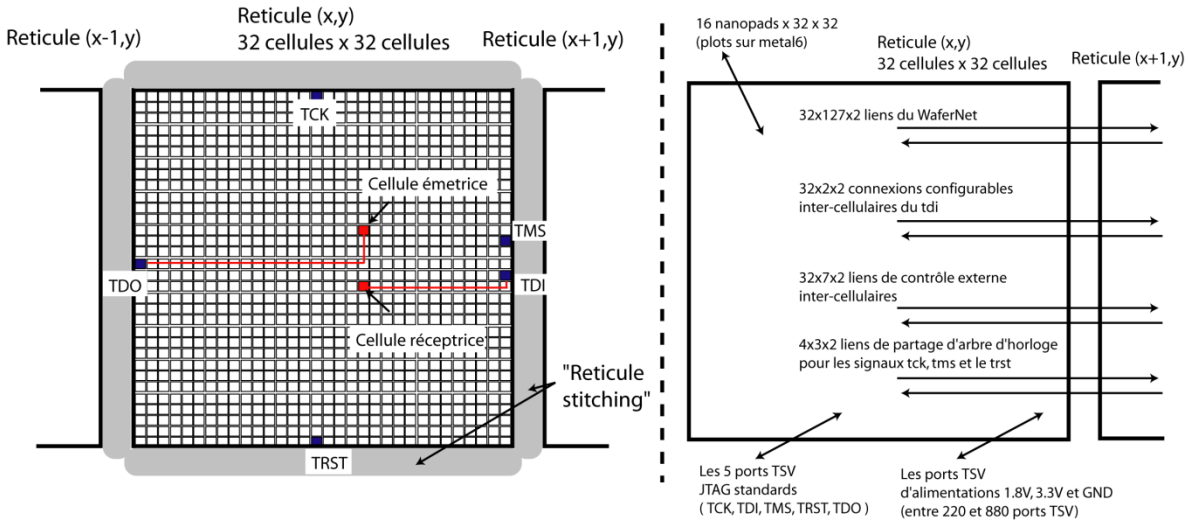
Comme il a été expliqué au chapitre 2, il existe 2 types de chaînes intercellulaires. Il y a les chaînes intercellulaires "inter-réticulaires" et les chaînes "intra-réticulaires". Ces dernières sont limitées à un seul réticule. Les cellules d'entrée et de sortie du flux de données sont incluses dans un seul et même réticule. La chaîne ne visite aucun autre réticule. La chaîne inter-réticulaire relie un ou plusieurs réticules. La chaîne peut entrer par un réticule et sortir dans un réticule voisin à partir des liens inter-réticulaires. Cette méthode n'est pas triviale à réaliser en pratique, car les réticules sont indépendants et communiquent par le biais d'un pont de communication qui ne garantie aucune synchronisation entre les signaux de deux réticules voisins.



4-1 Structure de la chaîne inter-cellulaire.

4.2.2 Le niveau Réticule

Dans les procédés standards de fabrication de circuits intégrés, il existe un écartement (“scribe line”) entre les réticules pour permettre le cisaillement de la gaufre. Pour avoir un circuit de la taille d’une gaufre, cet écartement peut être éliminé et les réticules doivent être reliés par des traces de métal. Par conséquent, ces traces de métal font partie des ports d’entrée/sortie du réticule, en plus des ouvertures (NanoPads) du dessus. L’autre façon d’interfacer avec le réticule est par des trous gravés dans le silicium (Through Silicon Via, TSV) par le dessous. C’est par ces TSV que l’alimentation en puissance (VDD3.3, VDD1.8 et GND) et les ports JTAG sont connectés. La figure 4-2 montre comment sont connectés les ports JTAG avec les cellules d’entrée et de sortie du réticule. La figure 4-2 de droite est un schéma bloc du nombre d’entrées et sorties au niveau des répéteurs du WaferNet.



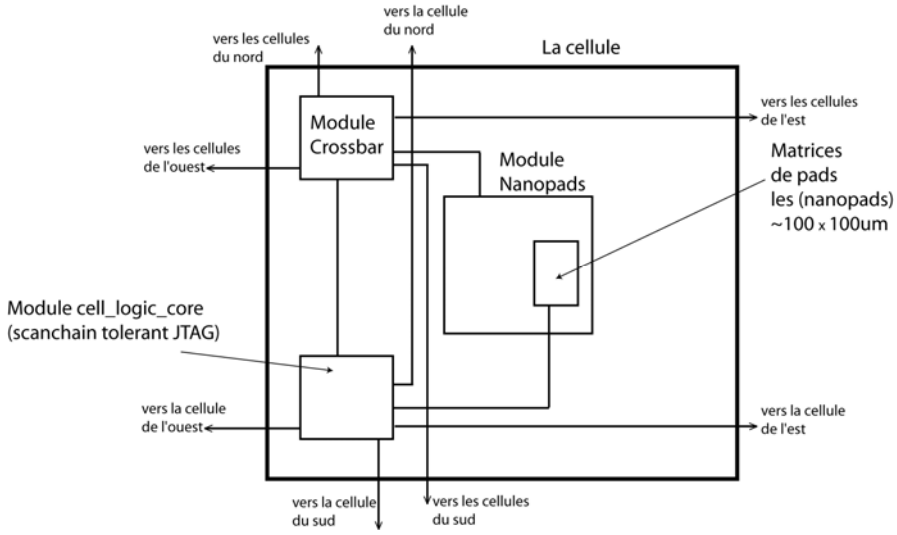
4-2 Structure interne du réticule.

La structure du fichier VHDL du réticule est générique. Cela signifie qu'il suffit de définir comme paramètre la grandeur "*n*" en terme de nombre de cellules dans une dimension du réticule pour simuler automatiquement un réticule d'une dimension *n*x*n* avec tous les ports d'entrée/sortie des cellules connectés entre eux. L'équipe a eu besoin d'un réticule générique parce qu'il était impossible de savoir exactement combien de cellules allait contenir le réticule final pour le premier prototype. De plus, l'aspect générique du réticule a bien servi, car le TestChip v1 contient 3x3 cellules, le TestChip v3 contient 8x8 cellules et le WaferIC en contiendra 32x32.

4.2.3 Le niveau Cellule

Le module "cell_top" est celui qui comprend la définition de l'architecture de tous les modules analogiques et numériques contenus dans une cellule. Les modules analogiques sont définis en tant que boîtes noires pour le placement/routage. Cela inclut aussi les connexions aux cellules voisines pour acheminer les signaux JTAG et le contrôle externe. Le schéma de la figure 4-3 montre comment est relié le module "cell_logic_core" avec le

reste du système. Les lignes des répéteurs des crossbars ne sont pas représentées pour simplifier la représentation. Les lignes ont plutôt été représentées au niveau wr_cell_top, qui est l'enveloppe du cell_top.



4-3 Architecture interne de la cellule.

Une description détaillée du crossbar et du WaferNet a déjà été faite au chapitre 3. Par contre, le lien entre le CLC (cell logic core) et le crossbar n'a pas été expliqué. En fait, le CLC gère la redirection des données de configuration provenant du logiciel de contrôle du WaferIC. Par conséquent, si le contrôleur externe désire programmer le crossbar, il doit utiliser les instructions JTAG pour configurer le CLC pour qu'il puisse être en mode décalage du crossbar courant. Une fois le mode activé, les données de configuration provenant de l'extérieur du WaferIC sont automatiquement redirigées vers la chaîne "scan" du crossbar.

4.2.3.1 Le module "cell_logic_core"

Le même principe s'applique pour la configuration des NanoPads. La figure D.1 de l'annexe D montre de manière détaillée ce qui avait été esquissé aux figures 2-3 et 2-4 du chapitre 2. La figure montre les registres standards utilisés dans le protocole IEEE 1149.1.

Le registre d'instruction est très important. Il permet d'activer des modes à l'aide du décodeur d'instructions. Ce registre définit le mode d'accès aux chaînes de scans. L'accès au registre bypass peut également être activé par le décalage d'une instruction dans le registre d'instructions. Le registre bypass sert à accélérer l'accès aux cellules suivantes dans la chaîne inter-cellulaire.

Il y a 4 chaînes de "scan" incluses dans chacune des cellules. Deux chaînes sont dédiées au crossbar. La chaîne nommée "tc" sert au test du crossbar. La chaîne "cr" permet de configurer le crossbar. Deux autres chaînes ("na" et "cd") servent à configurer respectivement les NanoPads et à activer le mode de détection de contact. De plus, la chaîne "cd" sert à capturer l'information résultant de la détection d'un contact entre les billes des circuits intégrés en contact avec le WaferIC et les NanoPads. Le Tap controller sert à contrôler le décalage des données dans les registres de scan. Il y a donc 4 chaînes de scan à l'intérieur de chaque cellule, mais chaque cellule peut accéder aux chaînes de scan des 4 cellules voisines. Il y a 5 cellules accessibles au CLC (4 cellules voisines et la cellule courante) avec 4 chaînes de "scan" par cellule pour un total de 20 chaînes de "scan" accessibles par le CLC. Chaque accès à une chaîne de "scan" doit être activé par une instruction décalée dans le registre d'instructions.

Le registre de destination est représenté sur le schéma par la boîte "dest_register". Ce registre sert à configurer le multiplexeur jtag_out_mux. Ce multiplexeur permet de contrôler la destination du flux de données pour la configuration de la prochaine cellule de la chaîne inter-cellulaire (décrit au chapitre 2).

Les modules "ext_ctrl_out_mux" et "ext_ctrl_in_mux" servent à gérer la sortie et l'entrée des flux de bits allant vers des cellules voisines ou en provenance des cellules voisines. Ce sont des boîtes du schéma bloc qui contiennent des multiplexeurs programmables à travers les modes activés dans la cellule.

4.2.4 Le niveau NanoPad

Le NanoPad inclut un plot et les circuits qui permettent la communication avec l'extérieur du WaferIC. Un module VHDL a été codé pour représenter les NanoPads comme un tout. C'est le module wr_nanopads schématisé à l'annexe D (fig. D.2). Dans le module "NanoPads" sont inclus les circuits analogiques de l'étage d'entrée et de sortie nommé le "io_buffer". Chaque cellule possède une matrice de 4x4 NanoPads. Les NanoPads communiquent directement avec le crossbar en passant par un étage de multiplexeurs configurables.

Les multiplexeurs, les chaînes de scan et la matrice de NanoPads font partie d'une entité VHDL nommée "wr_nanopads". Une seule cellule ne peut utiliser que 2 NanoPads en même temps pour le routage des signaux de données. Par conséquent, les multiplexeurs sélectionnent au plus 2 signaux d'entrée provenant des NanoPads ou bien 2 signaux de sortie en direction des NanoPads. Les multiplexeurs d'entrée "in_bx_mx" (Fig.D.2) et les multiplexeurs de sortie "out_bx_mux" sont configurables par l'utilisateur. En effet, le signal de sélection est contrôlé par des registres "scan" connectés au système de configuration UCIC. Le module "wr_nanopads" fournit donc 2 signaux "out_bx" au crossbar local. Celui-ci redirige ces signaux vers le "WaferNet". De plus, le module "wr_nanopads" reçoit 2 signaux "in_bx" en provenance du crossbar local pour les rediriger en sortie vers les NanoPads.

La figure D.2 de l'annexe D illustre les NanoPads identifiés par un numéro d'instance qui est fonction de l'ordre de chaînage des NanoPads dans la cellule. Pour économiser des ressources de routage, il a été décidé d'entrelacer les NanoPads entre eux. Au niveau logiciel, les NanoPads sont identifiés d'une manière plus intuitive par un système de coordonnées (x,y). C'est au niveau logiciel que se fait la conversion entre les coordonnées et l'ordre d'apparition du NanoPad dans la chaîne de balayage.

Le module “wr_nanopads” s’occupe aussi du multiplexage d’un autre signal, nommé “oe_wr_na”. Ce signal permet de contrôler la fermeture de l’étage de sortie des NanoPads. Cette fonctionnalité permet d’obtenir un certain niveau de bidirectionnalité entre les broches des circuits intégrés déposés sur le WaferIC. Encore une fois, au maximum 2 signaux de contrôle peuvent être redirigés vers les NanoPads.

4.2.4.1 La structure interne du NanoPad

Les 16 NanoPads de la cellule sont chaînés ensemble par 2 chaînes de “scan”. La première chaîne “na” permet de configurer l’état du module “io_buffer”. La seconde chaîne “cd” permet de capturer les données provenant des “boundary scan” intégrés dans chaque NanoPad et les résultats du test de court-circuit entre le NanoPad et la broche d’un circuit intégré (“na_rd_tpt_reg”). La structure interne du NanoPad est montrée à l’annexe D, figure D.3.

Sur la chaîne “na”, il y a un total de 8 bits de configuration pour le NanoPad, ce qui donne un total de 256 états possibles au niveau combinatoire. Par contre, ces 256 possibilités ne sont pas toutes utiles. Il y a 2 bits pour l’état du NanoPad indiqué par le registre “state_reg” sur la figure D.3. Les NanoPads peuvent être configurés en mode “data input”, “data output”, “GND” et “VDD”. Ces deux derniers états (GND, VDD) permettent de configurer le NanoPad pour qu’il puisse alimenter les circuits intégrés déposés sur le WaferIC. Le registre “level_reg” permet de configurer la transmission de données ou la puissance à un niveau de tension 3.3V ou 1.8V. Le “na_wt_tpt_reg” permet de configurer le mode de détection de contact entre le NanoPad et la broche du circuit intégré. Les registres log_level_reg, static_oe_reg et dyna_oe_reg permettent de gérer le signal “oe, output_enable” de l’étage entrée/sortie du NanoPad.

4.3 L’environnement de covérification

Il ne suffit pas de simplement coder la structure interne d’un modèle numérique avec un langage de description matérielle. Il faut aussi procéder à la vérification matérielle qui

consiste à vérifier si les résultats générés par le modèle numérique correspondent aux résultats attendus, c'est-à-dire aux spécifications. Dans le cas du design du WaferIC, il faut faire encore plus. Il faut s'assurer que le matériel soit capable de recevoir les commandes provenant du logiciel et il faut s'assurer que le logiciel soit capable de bien commander au matériel. En d'autres mots, il s'agit de développer des routines bas-niveau logiciel avant la fabrication du circuit et il faut tester le matériel qui n'existe pas encore physiquement face aux exigences d'un logiciel en cours de développement. Ce processus s'appelle la covérification.

Pour être plus précis, le système de vérification dédié au WaferIC doit être capable de valider trois types de fonctionnalités qui impliquent à la fois une maîtrise logicielle et matérielle du processus :

1- Validation au niveau logiciel des algorithmes de diagnostic Comme il a été expliqué aux chapitres 2 et 3, il est essentiel de tester et diagnostiquer le circuit pour permettre la tolérance aux fautes. Les techniques de diagnostic proposées dans ce document sont basées sur un "test controller" qui peut être implanté à l'aide d'un logiciel embarqué dans un processeur ou bien sur un ordinateur externe en communication constante avec le WaferIC. Ainsi, les commandes envoyées par le contrôleur de test peuvent être codées et validées avant la fabrication du circuit. Ainsi, l'environnement de vérification permet la validation de ces algorithmes.

2- Validation au niveau logiciel des routines de configuration Le WaferIC doit être configuré de manière précise pour relier entre eux les circuits intégrés déposés à sa surface. Il faut configurer adéquatement les NanoPads et les crossbars, et mettre en place la chaîne inter-cellulaire pour atteindre les cellules intérieures du réticule. La configuration est contrôlée par des routines bas-niveau JTAG qui émulent le comportement JTAG du système. Ainsi, le logiciel génère les flux de bits JTAG nécessaires pour configurer les registres internes du système.

3- **Validation des routines de test du TestChip V1.0** Si un circuit n'est pas testable, il ne peut pas être fonctionnel. Il est très important d'être capable de tester la testabilité du WaferIC avant que le premier prototype soit fabriqué. Puisque le contrôle des séquences de test provient lui aussi d'un logiciel, il faut utiliser la covérification dans la stratégie de développement.

4- **Validation des routines de détection de contact** Il existe un algorithme optimisé de détection de court-circuit entre les NanoPads. La description détaillée de cet algorithme va au-delà du sujet couvert par ce mémoire.

4.4 Conclusion

Dans ce chapitre, la structure hiérarchique du code VHDL utilisé pour réaliser le TestChip V1.0 a été expliquée ainsi que la méthode de covérification. Pour plus de détails sur la méthode de vérification utilisée, il faut consulter deux documents. Le premier document est un article de conférence publié par l'auteur de ce mémoire, qui a été ajouté à l'annexe E. Le deuxième document est l'annexe A qui montre en détail le flot de travail pour la vérification matérielle.

CHAPITRE 5. RÉSULTATS DU TESTCHIP V1.0

5.1 Introduction

La conception du WaferIC a été réalisée par une équipe de concepteurs composée principalement d'étudiants à la maîtrise et un post-doctorant. De plus, le projet DreamWafer est un projet complexe incluant plusieurs phases s'échelonnant sur plus de 2 ans et il continue d'être actif en date de publication de ce mémoire. Mes contributions à ce projet se situent à plusieurs niveaux :

1- conception et réalisation du PCB pour le test du "testChip V1.0" (le circuit d'essai V1);
2- conception de l'architecture tolérante aux fautes du système de programmation du WaferIC;
3- programmation de routines logicielles bas-niveau pour la configuration JTAG du WaferIC;
4- conception d'algorithmes de diagnostic du WaferNet;
5- écriture du logiciel de génération des vecteurs de test pour TestChip V1 et V3;
6- conception de l'environnement de vérification.

Toutes ces contributions font partie intégrante du système. La première partie de ce chapitre présente les résultats apportés par l'environnement de vérification. Ensuite la deuxième partie de chapitre présente le résultat principal du travail dans ce projet de maîtrise qui est la réalisation du circuit d'essai V1 en tant que prototype physique, testable et complet.

5.2 Environnement de vérification pour les technologies associées au WaferIC

La vérification fonctionnelle consiste à vérifier chaque fonctionnalité (caractéristiques) par un ou plusieurs cas de test ("*test case*"). Les détails d'implantation de la méthode de vérification fonctionnelle ont été expliqués au chapitre 4. Cette sous-section a pour but

d'expliquer et de commenter les résultats de la vérification, puisque beaucoup d'efforts ont été investis dans cette partie du projet.

Un total de 379 tests de vérification a été effectué pour tester le système de programmation JTAG avec un ensemble de routines logicielles bas-niveau pour le contrôle JTAG. Chaque test a été intégré dans un script de régression. À chaque fois qu'une modification est effectuée dans le code source au niveau RTL ou bien au niveau des routines logicielles de base, le script de régression est ré-exécuté pour s'assurer que la modification ne fait pas apparaître d'autres bogues ailleurs dans le système. Plus de détails ont été intégrés dans l'annexe A sur le flot de travail associé aux tests de régression.

Une synthèse de tous les tests de vérification a été incluse à l'annexe F. Cette annexe présente un tableau synthétique de tous les tests de vérification effectués pour le circuit d'essai V1.

Notons que la conception et la réalisation d'un cas de test pour le WaferIC et pour le circuit d'essai v1 consiste en 5 tâches :

1- Il faut commencer par mettre en place une routine de génération automatique de vecteurs de test pour ce cas.
2- L'étape suivante consiste à s'assurer que le cas de test est exécutable dans l'environnement de vérification. Il faut être capable d'appliquer de manière synchrone les vecteurs de test sur le modèle RTL et de capturer les résultats générés par le modèle.
3- La troisième étape consiste à ajouter des routines d'auto-vérification du test. Si l'environnement de vérification a accès aux données générées suite à l'application d'un cas de test sur le modèle, il devient possible de prouver automatiquement qu'un test est appliqué avec succès sur le modèle.

4- Ensuite, invariablement, des bogues apparaissent. Cette étape consiste donc à "déverminer" le modèle RTL suite aux problèmes détectés par le cas de test. La source des bogues peut être la routine de génération de vecteur de test ou bien peut provenir de l'environnement de vérification lui-même ou de l'auto-vérification.
5- La dernière étape consiste à intégrer le cas de test dans le test de régression. Une fois le test intégré dans le script de régression, ce script est ré-exécuté.

À remarquer que chacune des étapes est identifiée par une colonne dans le tableau de l'annexe F. Un point est donné pour la complétion d'une étape de développement d'un cas de test pour un total de 5 points par cas de test. Cette méthode permet d'évaluer l'état d'avancement du projet de vérification.

Leçons apprises :

1- Un principe bien connu par les spécialistes en vérification matérielle est le suivant : l'effort que met un développeur matériel pour faire le design d'un modèle ne représente qu'environ 25% du temps total. Le reste du temps étant dédié à la vérification matérielle et au débogage [6]. Par conséquent, en moyenne, 3 fois plus de temps est dédié à la vérification qu'au design du système. Ce principe a été observé durant le projet pour la réalisation de la partie matérielle.
2- Il a été possible durant le projet de noter la provenance du bogue une fois trouvé et corrigé, ce qui a permis d'accumuler des statistiques. Il a été remarqué qu'environ 10% des bogues provenaient du modèle VHDL lui-même. De plus, 2 bogues importants ont été trouvés dans la version post-synthèse Verilog. Environ 30% des bogues proviennent du générateur de vecteur de test, 30% de l'environnement de vérification lui-même et 30% proviennent des routines d'auto-vérification. Parmi les 10% des bogues trouvés dans le RTL, tous étaient des bogues subtils et difficiles à détecter. Même si la plupart des bogues détectés par le script de régression ne venaient pas du matériel, mais du

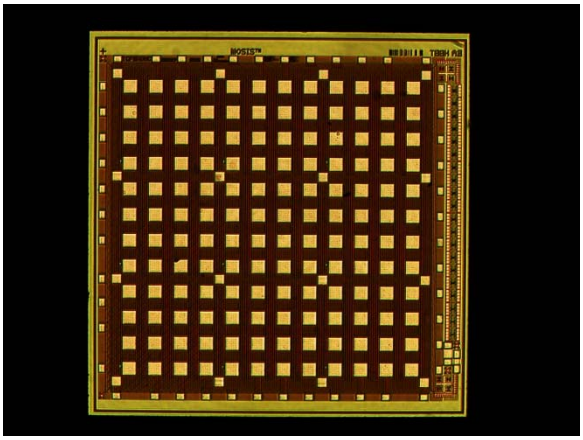
logiciel (90%), les bogues trouvés dans le matériel étaient d'une importance primordiale. De plus, le script de régression avait pour but de tester les routines de bas niveau JTAG qui génèrent les vecteurs de test. Cette bibliothèque de routines sera réutilisée durant le projet DreamWafer par la partie logicielle de l'équipe. C'est pourquoi l'utilisation d'un système de vérification est très importante pour s'assurer d'un design d'une grande qualité au niveau logiciel. Cette même bibliothèeque permet d'accélérer le temps de développement de la partie logicielle en commençant l'effort de test logiciel avant que le circuit physique n'existe effectivement.

3- Même si le test de régression contient un grand nombre de cas de test, le circuit d'essai v1 a fait ressortir 1 seul bogue qui provenait d'une erreur non détecté dans le modèle RTL. Tous les détails au sujet des bogues détectés dans le circuit d'essai V1 sont expliqués dans la section 5-3. La raison pour laquelle le bogue n'a pas été détecté est la suivante : tous les tests de régression étaient auto-vérifiants. Or, le test qui aurait dû détecter le bogue n'observait pas les bons signaux. En d'autres mots, les signaux dans le cône d'influence généré par le défaut n'étaient pas observés par le test de vérification. Cette erreur est humaine. Il aurait fallu être plus vigilant et s'assurer que tous les signaux importants soient observés.

5.3 Test du circuit d'essai v1

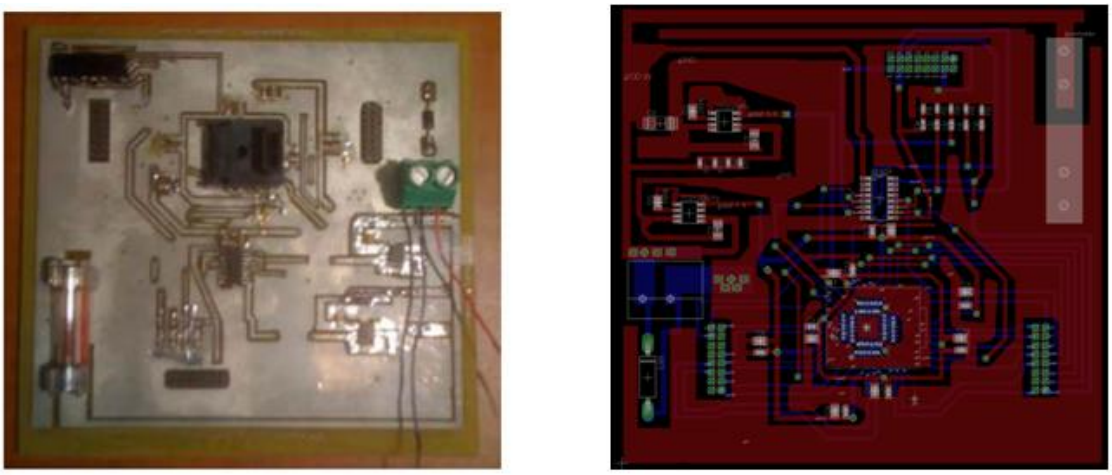
Le circuit d'essai V1 est une matrice de 3x3 cellules unitaires du WaferIC (Fig. 5.3). C'est en quelque sorte une version miniature du WaferIC où la plupart des fonctionnalités du WaferIC peuvent se valider sans être obligé d'avoir un prototype dispendieux incluant une tranche entière de silicium. Chaque cellule contient 4x4 NanoPads, ce qui donne un damier de 12x12 NanoPads. Parmi les 144 NanoPads, seulement 22 sont accessibles pour le test. Les autres ne sont pas accessibles, car ils n'ont pas été soudés sur les broches de sortie de la puce. Cinq signaux de contrôle JTAG permettent la programmation du TestChip v1 et deux niveaux de tension (3.3V et 1.8V) sont nécessaires pour alimenter en puissance le circuit intégré. Les blocs analogiques sont intégrés dans les NanoPads et sont programmables. La

partie numérique du circuit intégré utilise des cellules normalisées de la bibliothèque de la société Artisan.



5-1 Photo du dé de silicium du TestChip v1

Une carte de test dédiée au circuit de test a été conçue par l'auteur de ce mémoire avec l'aide de Nicolas Mayer. Cette carte de test (Fig. 5-2) est connectée à un micro-contrôleur qui permet de transmettre des signaux de commande JTAG au circuit d'essai v1. Le micro-contrôleur a été programmé par Étienne Lepercq et Philippe Aubertin. Il est connecté à un ordinateur et un logiciel qui permet de commander l'envoi de vecteurs de test. Le micro-contrôleur joue ainsi le rôle d'interface.



5-2 Vue XY du PCB fabriqué (gauche) et du dessin (droite)

Le circuit est placé dans une embase à insertion rapide qui permet de remplacer les circuits sous test sans endommager la carte de test. Les tests effectués sont de deux types. Premièrement, une série de tests analogiques a été effectuée pour détecter la présence éventuelle de courts-circuits dans les chips. D'autres tests analogiques plus avancés comme des mesures d'impédance sur les broches d'alimentation ont été effectués. Ensuite, les tests numériques les plus simples ont été appliqués sur le circuit.

Protocole de mesures et de tests : tous les signaux transmis au circuit sous test sont enregistrés par un analyseur logique qui donne toute l'information nécessaire au diagnostic des pannes. Un plan de test a été préalablement mis en place et testé sur l'environnement de covérification. Le test au niveau numérique a donc consisté à appliquer les vecteurs de test et à mesurer les résultats sur le circuit. Pour valider le comportement du WaferNet, des signaux ont été injectés dans les NanoPads accessibles depuis les broches du circuit sous test. Les signaux injectés dans les NanoPads ont été générés par des générateurs programmables d'ondes. Ces signaux entrant et sortant du WaferNet sont observés avec des oscilloscopes numériques. La figure 5-3 montre les résultats de ce test sur l'oscilloscope dans le laboratoire de test. Le courant passant par les broches d'alimentation (VDD) ainsi que par la mise à la terre (GND) ont été mesurés ainsi que le courant passant par les NanoPads configurés en mode VDD3.3V.

5.3.1 Résultat du test fonctionnel du TestChip

Le circuit d'essai v1 émule le comportement à petite échelle d'un WaferIC qui se déploie sur la totalité de la surface active d'une tranche de silicium. Le WaferIC étant composé d'une mer de cellules, mais seulement une matrice de 3x3 cellules est nécessaire pour émuler le comportement de base du circuit.

Tester le système JTAG de programmation implique la création d'une chaîne intercellulaire comme expliqué aux chapitres 2 et 4. Les données entrent par une cellule du WaferIC de réception des données et sortent par une cellule de transmission des données (voir figure 2-1et 2-2). Cette fonctionnalité est critique, car tout dans le système du circuit sous test doit être configuré par la chaîne JTAG avant d'être testé. Si le système de programmation JTAG ne fonctionne pas, le système entier n'est pas testable, donc non fonctionnel. Deux types de test ont été construits :

1- **Le test minimaliste intercellulaire** Il établit une chaîne de balayage inter-cellulaire partant de la cellule de réception jusqu'à la cellule de transmission contenant le port tdo. Ces cellules étant des voisines, la chaîne intercellulaire ne contient que 2 cellules.
2- **Le test de la chaîne intercellulaire prolongée** Cette chaîne passe à travers plus de 2 cellules. Cette chaîne prouve la capacité du contrôleur de test de prendre le contrôle des "cellules internes" du circuit intégré sous test et de créer un chemin à travers chacune des cellules. Pour le cas du circuit 3x3, il est géométriquement impossible de traverser toutes les cellules avec une seule chaîne intercellulaire unidirectionnelle. Il faut donc plus d'une passe pour programmer toutes les cellules du circuit intégré.

Quand le premier test minimaliste intercellulaire a été effectué, une valeur numérique '1' à 3.3V a été observée sur le port de sortie JTAG tdo à la fin du test. Par contre, l'état du port de sortie s'est avéré être collé à '1'. Après une investigation, il a été prouvé que le port tdo n'a pas été relié au bon port du module interne numérique. À la place, le tdo du TestChip v1 a été connecté à un signal interne de contrôle. En réaction à cette erreur de conception diagnostiquée, un nouveau cas de test a été développé pour contrôler le signal interne connecté au tdo. Or, l'expérience nous a montré que c'était bien le signal interne qui était connecté au port tdo, car les résultats du test simulé correspondaient exactement aux

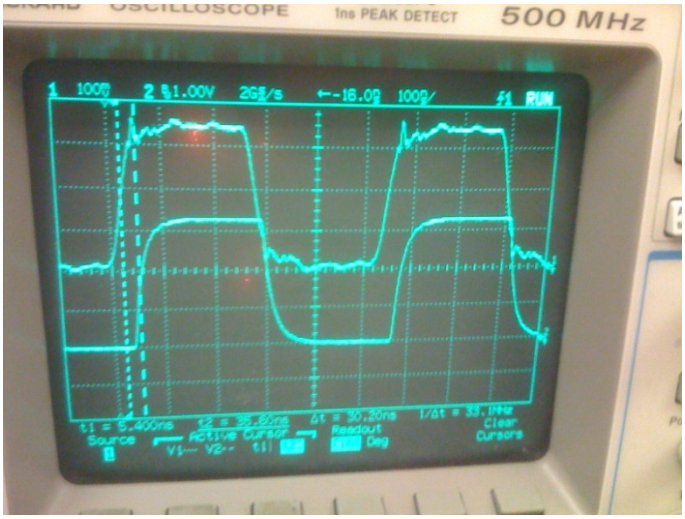
résultats expérimentaux. Pour surmonter cette erreur de design, la plupart des tests du plan de test ont été adaptés.

5.3.2 La configuration du WaferNet^{TM} et des NanoPads

Le test du WaferNet implique de configurer plusieurs crossbars et NanoPads. Au moins un NanoPad doit être configuré en entrée et un autre NanoPad doit être configuré en sortie. Les NanoPads configurés et choisis pour le test doivent être connectés aux broches du circuit sous test pour être observables avec les analyseurs logiques. Deux types de tests ont été effectués sur le WaferNet :

- Huit tests ont été effectués pour tester le WaferNet en utilisant des NanoPads spécifiques comme entrée et sortie. Un seul chemin reliant l'entrée et la sortie est créé dans le WaferNet.
- Un test a été configuré pour tester le WaferNet incluant l'activation de tous les crossbars et de tous les NanoPads configurés en sortie.

Durant le test, les NanoPads ont été configurés pour fournir la tension 3.3V. Une défectuosité a été trouvée dans le circuit d'essai v1. En effet, les NanoPads du ce circuit sont incapables de fournir une tension de 1.8V comme sortie pour les signaux et pour la tension d'alimentation. L'erreur dans le code VHDL causant cette défectuosité a été trouvée et corrigée pour les futures versions.



5-3 Propagation du signal dans le WaferNet du TestChip V1.0

Un courant de fuite a été détecté dans le circuit. Selon le diagnostic d'Olivier Valorge fait à partir des données expérimentales, le courant de fuite pourrait s'expliquer de la façon suivante. Les entrées flottantes des répéteurs situés sur les bords du réticule sont connectées à des diodes polarisées en inverse pour décharger les nœuds flottants. Le courant de fuite au travers de l'oxyde de grille des transistors pour la technologie 0.18 µm pourrait être supérieur ou du même ordre de grandeur que celui de jonctions pn. Un rapport d'impédance pourrait alors établir un potentiel suffisant aux entrées d'une grande proportion des répéteurs pour les activer dans le mode qui produit un important courant de courts-circuit.

Le test du circuit d'essai v1 est un très important jalon dans le projet DreamWafer et les résultats positifs obtenus montrent clairement qu'il est possible de créer un circuit avec un réseau configurable tel que le WaferNet. Les tests analogiques et numériques effectués sur le circuit d'essai confirment l'efficience et le bon fonctionnement du circuit en rapport avec les caractéristiques suivantes :

- Configuration et propagation au travers du Crossbar configurable numérique
- Configuration et propagation au travers du WaferNet configurable

- Étage d'entrée/sortie configurable
- Architecture JTAG interne ("full-custom") dédiée à la programmation du WaferIC.

De plus, le circuit d'essai v1 a permis de mettre en lumière les dysfonctionnements au niveau numérique et analogique, ce qui permettra à l'équipe de réparer ces défauts pour une future version du WaferIC qui occupera une tranche complète de silicium.

CONCLUSION

Ce mémoire a présenté les travaux de recherche liés au projet DreamWafer dans le but de créer un système numérique sur la totalité de la surface d'une tranche de silicium. Le WaferIC est un circuit monolithique qui utilise la totalité de la surface active de silicium pour le prototypage rapide de système numérique. Le concept est le suivant : une plate-forme de prototypage doit arriver à interconnecter les CDSP (circuit discret sous prototypage) déposés à la surface d'un circuit intégré à l'échelle de la tranche. Ce circuit géant doit interconnecter numériquement les broches des CDSP par le biais de son réseau interne d'interconnexion programmable. Ainsi, il est possible de créer une panoplie de circuits au niveau système. Le WaferIC est en quelque sorte la pièce maîtresse d'un PCB

numérique reconfigurable. L'architecture du WaferIC est très originale et est protégée par un brevet [27]. Elle consiste en l'utilisation d'une mer de cellules identiques pour créer par juxtaposition un réseau reconfigurable et un quadrillage de plots configurables nommés NanoPads. Les NanoPads sont des plots très petits qui peuvent se configurer en mode puissance (VDD,GND) ou en mode signal (Input, Output) pour permettre aux signaux du système de CDSP d'échanger des signaux ou pour alimenter en puissance les CDSP.

La technique de conception utilisée pour le WaferIC consiste à juxtaposer des cellules identiques. Les cellules sont différenciées par le contenu de leur mémoire interne lors de la programmation du système. Cette mémoire sert à configurer l'état des nanopads (VDD, GND, IN, OUT, INOUT, etc) et à configurer le réseau interne du WaferIC nommé WaferNet. La régularité structurale du WaferIC permet de simplifier le design d'un circuit qui se déploie sur toute la surface active d'une tranche de silicium. Chaque cellule possède la capacité de détecter des contacts entre les NanoPads et les CDSP et permet d'envoyer ces informations à un logiciel de contrôle qui interagit avec un usager.

Le WaferNet est un maillage multidimensionnel régulier. Chaque cellule est connectée à ses 4 voisins et à ses voisins distants de plusieurs cellules dans chaque direction. Chaque cellule contient un crossbar programmable pour rediriger les signaux en fonction des requêtes de l'usager.

Pour arriver à concevoir cette technologie, il faut pouvoir la définir, la simuler et la valider. Une contribution de ce mémoire est de démontrer la faisabilité de la simulation du WaferIC et vient montrer en détail l'architecture interne de la cellule. Une autre contribution a aussi été au niveau de la vérification fonctionnelle méthodique du WaferIC en utilisant le SystemC et dans le codage d'un émulateur logiciel du protocole JTAG adapté aux besoins spécifiques du projet. Un article de conférence a été rédigé sur ce sujet. La nouveauté de

cette contribution consiste simplement en l'application des principes de la co-vérification au processus de conception du WaferIC. La co-vérification étant la possibilité de vérifier un logiciel avant que le matériel n'existe physiquement et vérifier que le matériel peut exécuter effectivement le logiciel en cours de conception.

Pour que cette technologie soit viable commercialement, il est primordial d'utiliser des techniques de tolérance aux fautes qui soient le plus économe possible du point de vue de la consommation de surface. La tolérance aux fautes permet justement de créer un circuit fonctionnel en présence de fautes que l'on sait inévitables pour les circuits à l'échelle de la tranche. Les techniques de tolérance aux pannes font partie des principales contributions de ce mémoire. Les techniques de tolérance aux pannes proposées dans ce document sont originales et font l'objet d'une divulgation. Les techniques de tolérance aux pannes ont été appliqué principalement à 2 sous-ensembles du WaferIC : le système de configuration JTAG et le WaferNet.

Au niveau du système de communication sérielle JTAG, deux solutions originales complémentaires ont été proposées pour rendre tolérant aux défauts un système de communication sérielle JTAG. Ces solutions font l'objet d'une divulgation en cours d'approbation. Le système de communication sérielle JTAG a pour fonction de configurer le réseau interne (NoW, Network on Wafer) et les entrées/sorties du WaferIC (les NanoPads) pour émuler les circuits au niveau système défini par l'usager. De plus, ce système JTAG permet de rendre efficacement testable/diagnosticable le système. Le principe de diagnostic pour éviter les zones défectueuses du circuit a été appliqué pour le système JTAG. Il s'agit de créer un système capable de configurer à volonté une chaîne JTAG intercellulaire suffisamment versatile pour éviter les zones défaillantes du circuit. Cette solution nommée UCIC/BCIC (Unidirectional/Bidirectional Configurable

Intercellular Chain) a non seulement été simulée, vérifiée, mais a aussi été testée/validée physiquement sur un vrai circuit intégré.

Au niveau du WaferNet, il s'agit de reconfigurer intelligemment le réseau interne en fonction de la connaissance des zones dysfonctionnelles du circuit. Pour reconfigurer intelligemment le système, il faut un diagnostic le plus complet et le plus rapide possible pour trouver et éviter les fautes dans le circuit. Une méthode originale a été proposée pour le diagnostic du WaferNet qui consiste à se baser sur une adaptation de la méthode du "walking-one" pour tout type de réseau régulier de crossbar programmable. Cette méthode originale a été validée au niveau théorique, publiée dans un article de conférence [4] et implantée au niveau matériel pour la version du TestChip v3 qui a été soumise pour fabrication en septembre 2009 à Tower Inc. Ce circuit de test sera plus grand que le premier (8x8 cellules pour la version 3 contre 3x3 pour la version 1.0) et permettra de ce fait de tester plus en profondeur les algorithmes de diagnostic proposés dans ce document. De plus, plusieurs versions optimisées des algorithmes de diagnostic proposés dans ce document sont en train d'être écrites pour faire l'objet d'une divulgation. Par contre, une nouvelle méthode originale et optimisée de diagnostic du WaferNet a tout de même été proposée pour faire partie des contributions de ce mémoire.

Avec un système de configuration fonctionnel sur la puce de test, il a été possible de valider l'intégrité de la configuration du WaferNet et de la configuration des NanoPads sur un circuit de test physique. Il a été possible de tester la partie analogique de la puce de test pour détecter quelques problèmes potentiellement dangereux pour un WSIC. L'architecture cellulaire programmable a été testée physiquement avec succès sur un circuit de test fabriqué à partir de la CMC. Les crossbars programmables, les circuits de détection de contact CDSP/Nanopads ont été testés avec succès. Il a été démontré qu'il était possible de propager des signaux dans le réseau WaferNet. Quelques bogues numériques mineurs ont été détectés et corrigés.

Du point de vue académique, le projet sera un succès si l'équipe DreamWafer qui a considérablement grossi ces derniers mois arrive à faire fonctionner un premier WaferIC dans la plate-forme de prototypage WaferBoard. Il reste encore beaucoup de chemin à faire avant d'arriver à ce succès. Les travaux de recherche sur l'aspect numérique du WaferIC ne se terminent pas avec ce mémoire. Il reste un nombre considérable de découvertes et de savoir à acquérir pour maîtriser à un niveau avancé certains aspects et problématiques que soulève le design d'un système aussi unique que le WaferIC. Par exemple, en se limitant à énumérer les aspects numériques :

- Comment créer un système de minimisation du biais de synchronisation pour les arbres d'horloges qui seront "routés" dans le WaferNet?
- Comment arriver à diagnostiquer des problèmes de défauts dynamiques, causés par exemple par le crosstalk et le bruit dans le WaferNet?
- Comment créer des bus performants et bidirectionnels de signaux numériques dont le direction de transfert des signaux est encodée dans le protocole?
- Comment ajouter des assertions configurables dans le WaferBoard pour détecter des erreurs dans les systèmes sous test déposés sur le WaferIC?
- Comment ajouter d'une manière efficace de la logique programmable dans le WaferIC et quelles seraient les applications possibles?
- Comment transformer le WaferIC en un "green meter", c'est-à-dire en un système capable de mesurer le courant et la puissance consommés par un design inséré dans le WaferBoard?

Le projet évolue tellement vite que d'ici quelques mois, de nouvelles problématiques passionnantes vont peut-être émerger de la vision et de l'imagination des chercheurs de l'équipe DreamWafer. De ces réflexions émergeront peut-être les prémisses d'une nouvelle génération de circuits intégrés.

RÉFÉRENCES

[1] ABRAMOVICI M. and CHARLES E. STROUD, US Patent "Identifying faulty programmable interconnect resources of field programmable gate arrays", no. 6,966,020.

[2] Altium Corp. The Nanoboard. Consulté le 18 jan 2009, tiré de http://www.altium.com/products/the-nanoboard/en/the-nanoboard_home.cfm

[3] ANDO H., Testing VLSI with random access scan, in Proc. COMPCON, February 1980, pp. 50-52.

[4] BASILE-BELLAVANCE, Y., BLAQUIÈRE, Y., and SAVARIA, Y., "Faults diagnosis methodology for the WaferNet interconnection network," Circuits and Systems and TAISA Conference, 2009. NEWCAS-TAISA '09. Joint IEEE North-East Workshop on , vol., no., pp.1-4, June 28 2009-July 1 2009.

[5] BASILE-BELLAVANCE, Y.; LEPERCQ, E.; BLAQUIÈRE, Y. and SAVARIA, Y., "Hardware/software system co-verification of an active reconfigurable board with SystemC-VHDL," Electronics, Circuits and Systems, 2008. ICECS 2008. 15th IEEE International Conference on , vol., no., pp.1159-1162, Aug. 31 2008-Sept. 3 2008.

[6] BERGERON, J.,"Writing Testbenches: Functional Verification of HDL Models", Springer, 2nd edition, 2003.

[7] BHATTACHARYA D., "Hierarchical test access architecture for embedded cores in an integrated circuit," VLSI Test Symposium, 1998. Proceedings. 16th IEEE , vol., no., pp.8-14, 26-30 Apr 1998.

[8] BREWER, J.E., "Promises and Pitfalls of WSI", in Wafer Scale Integration, Kluwer, pp.1-29, 1989.

[9] CHANG C.and R. MELHAM R., "Arbitrary Size Benes Networks," *Parallel Processing Letters,* vol. 7, no. 3 pp.279-284, 1997.

[10] DAYTON UNIVERSITY, Eng. Faculty. Notes on JTAG. http://www.engr.udayton.edu/faculty/jloomis/ece446/notes/jtag/jtag1.html (Page consultée le 24 février 2008)

[11] DOUMAR, A.; ITO, H., "Detecting, diagnosing, and tolerating faults in SRAM-based field programmable gate arrays: a survey," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on , vol.11, no.3, pp. 386-405, June 2003.

[12] EL FOULADI, J., BLAQUIÈRE, Y., SAVARIA, Y., "Digital Measurement Technique for Capacitance Variation Detection on Integrated Circuit I/Os", *ICECS'07,* December 11-14, 2007.

[13] EVE CORP., EVE ZeBu-XXL for ASIC and SOC Emulation http://eve-team.com/products/zebu-xxl.php.

[14] HASSAN, A.; RAJSKI, J.; AGARWAL V.K., "Testing and diagnosis of interconnects using boundary scan architecture," *Test Conference, 1988. Proceedings. New Frontiers in Testing, International* , vol., no., pp.126-137, 12-14 Sep 1988.

[15] HAYES J.P., MUDGE T., **"**Hypercube supercomputers", Proceedings of the IEEE, Vol. 77, Issue 12, pp. 1829 – 1841, Dec. 1989.

[16] HSU, F.F.; BUTLER, K.M.; Patel, J.H., "A case study on the implementation of the Illinois Scan Architecture," Test Conference, 2001. Proceedings. International , vol., no., pp.538-547, 2001.

[17] IPC, Volume 1, International Technology Roadmap for Electronic Interconnections 2006 - 2007, 2006, p 31.

[18] JAIN, R.; MUKHERJEE, A.; PAUL, K., "Defect-aware design paradigm for reconfigurable architectures," Emerging VLSI Technologies and Architectures, 2006. IEEE Computer Society Annual Symposium on , vol.00, no., pp.6 pp.-, 2-3 March 2006.

[19] JALOWIECKI, I.P, HEDGE, S.J., "The WASP2 WSI Massively Parallel Processor Demonstrators", IEEE Custom Integrated Circuits Conference. 1990.

[20] KOREN, I., KOREN, Z.,"Defect Tolerance in VLSI Circuits: Techniques and Yield Analysis", IEEE, 86(9):1819-1838, 1998.

[21] LEE WHETSEL D., US Patent, "Parallel scan distributors and collectors and process of testing integrated circuits". no. *6646460* B2. Date of Patent: Nov., 2003.

[22] LENIHAN, T.G.; JAN VARDAMAN, E., "Worldwide Perspectives on SiP Markets: Technology Trends and Challenges," *Electronic Packaging Technology, 2006. ICEPT '06. 7th International Conference on* , vol., no., pp.1-3, 26-29 Aug. 2006.

[23] MADOU M., "Fundamentals of Microfabrication: The Science of Miniaturization", (2nd ed.), CRC Press (2002).

[24] MENG L.; SAVARIA Y.; BING Q.; TAILLEFER, J., "IEEE 1149.1 based defect and fault tolerant scan chain for wafer scale integration," Defect and Fault Tolerance in VLSI Systems, 2003. Proceedings. 18th IEEE International Symposium on , vol., no., pp. 18-25, 3-5 Nov. 2003.

[25] NARAYANAN S.; Breuer BREUER, M.A., "Reconfigurable scan chains: A novel approach to reduce test application time," *Computer-Aided Design, 1993. ICCAD-93.*

Digest of Technical Papers., 1993 IEEE/ACM International Conference on , vol., no., pp.710-715, 7-11 Nov 1993.

[26] NORMAN, R.; VALORGE, O.; BLAQUIÈRE, Y.; LEPERCQ, E.; BASILE-BELLAVANCE, Y.; EL-ALAOUI, Y.; PRYTULA, R.; SAVARIA, Y., "An active reconfigurable circuit board," Circuits and Systems and TAISA Conference, 2008. NEWCAS-TAISA 2008. 2008 Joint 6th International IEEE Northeast Workshop on , vol., no., pp.351-354, 22-25 June 2008.

[27] NORMAN, R., US Patent "Reprogrammable Circuit Board with Alignment-Insensitive Support for Multiple Component Contact Types" no. 11/611,263.

[28] PARKER, P.K., "*The boundary-scan handbook: analog and digital*", 2nd ed., Kluwer Academic Publishers Boston/ Dordrecht/ London, 1998, ch. 1,4.

[29] POUPON, G et al, "System on Wafer: A New Silicon Concept in SiP," *Proceedings of the IEEE* , vol.97, no.1, pp.60-69, Jan. 2009.

[30] TSAI M.K., "From PC Multimedia Chipsets to Digital Consumer SOC: Evolution and Challenges," *Solid-State Circuits Conference, 2006. ASSCC 2006. IEEE Asian* , vol., no., pp.11-13, 13-15 Nov. 2006.

[31] WANG, L., WU, C., and WEN, X. 2006 "VLSI Test Principles and Architectures: Design for Testability (Systems on Silicon)". Morgan Kaufmann Publishers Inc.

[32] WUANG, W. E., CHEN , X. T. and LOMBARDI F., "On the Diagnosis of Programmable Interconnect Systems: Theory and Application", 14th VLSI Test symposium.

[33] XILINX, Virtex-5 Packaging and Pinout Specification, 14 august 2007.

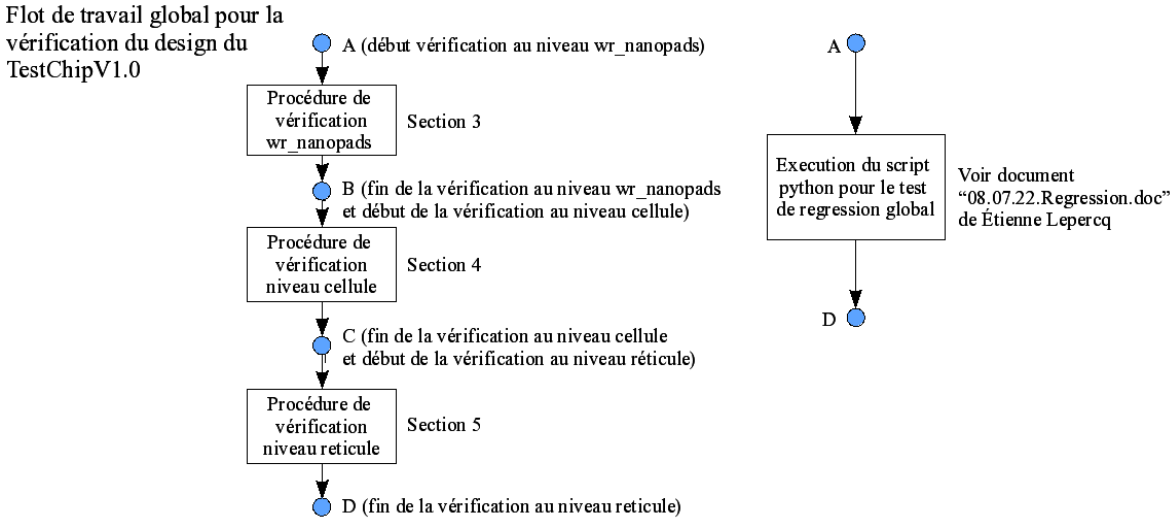
ANNEXE A – FLOT DE TRAVAIL POUR LA VÉRIFICATION DU TESTCHIP

Objet

Ce document s'adresse à toute personne désireuse de comprendre en détail la méthodologie globale de travail utilisée pour vérifier et surtout exécuter les modèles RTL et "post-layout" pour le TestChip 1. Ce document vient compléter les informations déjà données par Étienne Lepercq en montrant le flot de travail global pour le TestChip v1. Ainsi, une personne voulant reproduire les résultats déjà atteints par l'équipe de design numérique peut utiliser ce document pour s'orienter.

Flot de travail global et structure du document

La vérification se fait à partir de 3 niveaux hiérarchiques. Ces trois niveaux sont le niveau *wr_nanopad*, le niveau *cellule* (cell_top) et le niveau *réticule* (réticule). Chaque test se base sur la prémisse que le niveau hiérarchique précédent (inférieur) a été testé avec succès.



Chaque niveau hiérarchique est testé par un script de régression comprenant plusieurs test cases. Chacun de ces test cases comprend un ensemble de tests spécifiques pour une situation ou une configuration du circuit en particulier. Comme le montre la figure 1, la section 3 de ce document porte sur la vérification du niveau hiérarchique *wr_nanopads*

(lire wrapper nanopads), la section 4 porte sur la vérification et l'exécution du niveau *cellule* et la section 5 porte sur la niveau *réticule*.

Le document 08.07.22.Regression.doc décrit en détail comment utiliser le script de régression global permettant de choisir plusieurs scénarios de test possibles. Le script automatise le test de chaque niveau d'abstraction et compile les résultats dans un seul fichier.

Procédure de vérification du niveau wr_nanopads

Le niveau hiérarchique wr_nanopads se trouve dans le répertoire

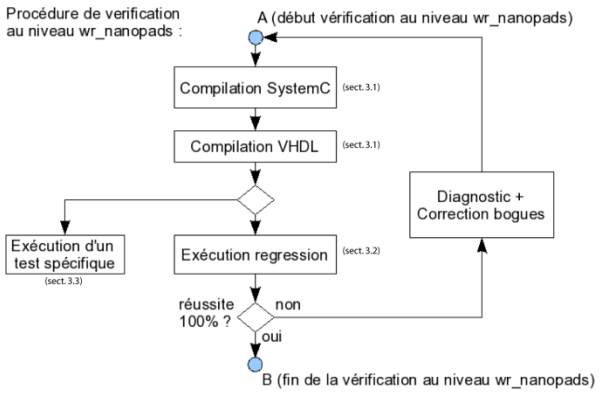
REP_PROJET$[1]/DreamWafer/trunk/nanopads/ et le code source associé au script d'exécution ModelSim se trouve dans le répertoire :

REP_PROJET$/DreamWafer/trunk/nanopads/sw/bin

La même logique d'arborescence s'applique dans les autres branches du système, par exemple :

REP_PROJET$/DreamWafer/branches/V1.0YN/nanopads/…

REP_PROJET$/DreamWafer/branches/elV1.0/nanopads/…



La figure 2 montre une procédure de vérification. Les sections 3.1 à 3.4 viennent décrire en détail comment répéter toutes les étapes montrées sur le flot de travail.

[1] **REP_PROJET$ = le répertoire où l'on a fait un «svn checkout du projet DreamWafer**

Compilation SystemC et VHDL

La compilation du système de vérification se fait en 2 étapes. La première étape est la compilation du code VHDL qui se fait à l'aide d'un script nommé compile_vhdl_wic.do. Pour exécuter le script, il faut ouvrir une fenêtre ModelSim dans le répertoire /nanopads/sw/bin et taper

>> do compile_vhdl_wic.do

Pour compiler le systemC, il faut ouvrir une fenêtre ModelSim et taper la commande suivante :

>> do compile_test_env.do

La compilation doit durer plusieurs minutes. Une fois la compilation passée avec succès, le modèle est prêt à être exécuté. Pour exécuter un test case en particulier, il faut consulter la section 3.3.

Exécution du test de régression

Un script de régression global existe, mais l'usager peut choisir d'exécuter un seul script de régression comprenant l'ensemble des test cases d'un niveau hiérarchique donné. Pour ce faire, il suffit d'utiliser un script python nommé regression_script_jt.py. Pour l'exécuter, il faut taper dans le terminal la commande suivante :

>> cd REP_PROJET$/nanopads/sw/bin

>> python regression_script_jt.py

Les résultats du test de régression s'affichent à la ligne de commande et sont sauvegardés dans le fichier REP_PROJET$/nanopads/sw/log/global_wrna_regression.log.

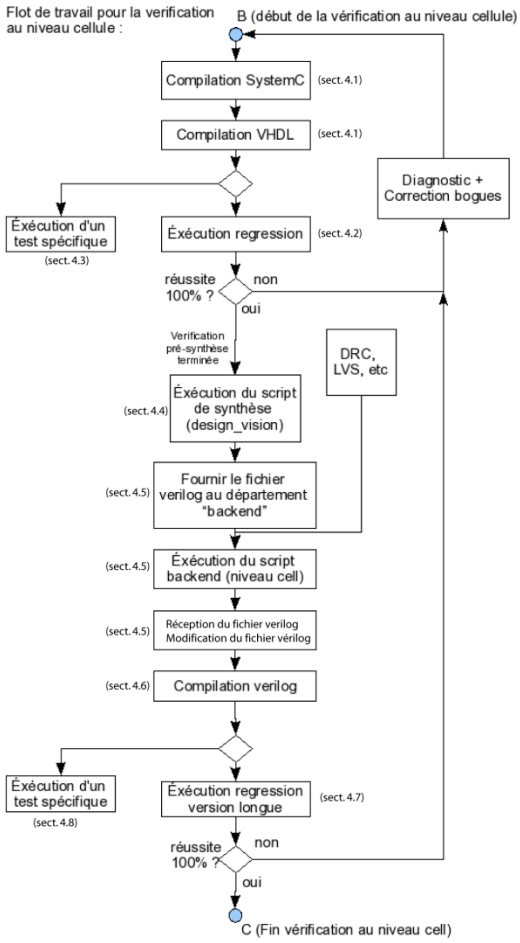
Exécution d'un test spécifique

Puisque ce sont les modules SystemC qui contrôlent le test du modèle VHDL, c'est dans un fichier SystemC que se trouve la description fonctionnelle des test cases. Pour le niveau wr_nanopads le fichier contenant la description détaillée des test cases est REP_PROJET$/nanopads/sw/src/wrna_testcases.cpp.

Pour exécuter un test case en particulier, il faut choisir le nom du "test case" dans la liste des "test cases" contenue dans le fichier REP_PROJET$/reticle/sw/src/def_system.h et recopier ce nom dans le fichier REP_PROJET$/nanopads/sw/run/testcase.txt. Ensuite, dans la ligne de commandes de ModelSim, il faut exécuter un script qui lance la simulation. De plus, le script choisit les signaux à visualiser. Par exemple, le script debug_wr_nanopads_asic.do peut lancer une simulation avec la visualisation d'un grand nombre de signaux pertinents. L'environnement de vérification lit le fichier testcase.txt et active le code du "test case" choisi.

Procédure de vérification du niveau cellule

La procédure de vérification du niveau cellule est beaucoup plus complexe que le niveau hiérarchique wr_nanopads. La vérification au niveau cellule se fait à travers des ”test cases” qui passent à travers les fonctionnalités complexes du système de contrôle JTAG et les fonctionnalités du WaferNet. Le flot de travail pour la vérification comprend la vérification post-layout du code verilog final. La figure 3 montre le flot de travail.



Les prochaines sections décrivent en détail la procédure à suivre pour refaire la vérification du code d’une cellule du DreamWafer. Le niveau hiérarchique de la cellule se trouve dans le répertoire

REP_PROJET$[2]/DreamWafer/trunk/cell/ et le code source associé au script d'exécution ModelSim se trouve dans le répertoire :

REP_PROJET$/DreamWafer/trunk/cell/sw/bin

Compilation SystemC et VHDL

La compilation du système de vérification se fait en 2 étapes. La première étape est la compilation du code VHDL qui se fait à l'aide d'un script nommé compile_vhdl_wic.do. Pour exécuter le script, il faut ouvrir une fenêtre ModelSim dans le répertoire /nanopads/sw/bin et taper

```
>> do compile_vhdl_wic.do
```

Pour compiler le systemC, il faut ouvrir une fenêtre ModelSim et taper la commande suivante :

```
>> do compile_test_env.do
```

La compilation doit durer plusieurs minutes. Une fois la compilation terminée avec succès, le modèle est prêt à être exécuté. Une autre méthode simple pour compiler l'environnement systemC est d'utiliser un script python qui vient s'assurer que tous les fichiers de configuration du modèle sont adéquatement forcés à une valeur cohérente pour la simulation niveau cellule. Pour exécuter le script, il faut se rendre dans le répertoire /bin du niveau hiérarchique de la cellule et taper sur un terminal (et non dans ModelSim)

```
>>python recompile_all.py
```

Pour exécuter un test case en particulier, il faut consulter la section 4.3.

Exécution du test de régression

Il est possible de choisir d'exécuter seulement les test cases du niveau hiérarchique de la cellule. Pour ce faire, 2 scripts de régression existent. Ce sont les scripts suivants :

REP_PROJET$/DreamWafer/trunk/cell/sw/bin/regression_script_jt.py

REP_PROJET$/DreamWafer/trunk/cell/sw/bin/regression_script_wf.py

Le premier exécute le test de régression associé au système de contrôle JTAG et le deuxième au WaferNet.

Exécution d'un test spécifique

Pour exécuter un test case en particulier, il faut choisir le nom du "test case" dans la liste des "test cases" contenue dans le fichier REP_PROJET$/reticle/sw/src/def_system.h et

---

[2] **REP_PROJET$ = le répertoire où l'on a fait un "svn checkout" du projet DreamWafer**

recopier ce nom dans le fichier REP_PROJET$/cell/sw/run/testcase.txt. Ensuite, dans la ligne de commande de ModelSim, il faut exécuter un script qui lance la simulation et choisit les signaux à visualiser. Par exemple, le script debug_cell_top.do peut lancer une simulation avec la visualisation d'un grand nombre de signaux pertinents. L'environnement de vérification lit le fichier testcase.txt et active le code du "test case" choisi.

Exécution du script de synthèse

La synthèse de la cellule est expliquée en détail dans le document 08.07.22.RTL-synthese.doc de Étienne Lepercq.

Backend et modification du netlist verilog final

Le script de synthèse de la cellule génère une netlist qui est ensuite envoyée au département backend de l'équipe DreamWafer géré par Olivier Valorge. Un script "backend" est exécuté dans l'environnement de travail d'Olivier pour générer un 2^{e} netlist verilog plus complet appelé "netlist post-layout". Ce netlist est soumis dans la structure de répertoire svn à l'emplacement suivant :

REP_PROJET$/DreamWafer/trunk/cell/sw/backend/PR_data/wr_cell_top_PR.v

Le répertoire PR_data contient aussi des fichiers d'annotations de délais pire cas, et typique (wr_cell_top_wr.sdf et wr_cell_top_typ.sdf).

Le fichier wr_cell_top_PR.v n'est pas prêt à être simulé par ModelSim. Il faut effectuer une série de modifications. Ces modifications se font manuellement ou bien automatiquement avec le script de régression global décrit dans le document 08.07.22.Regression.doc.

Compilation verilog

Une fois les modifications effectuées au fichier verilog post-layout, la compilation peut être effectuée. Pour compiler le fichier verilog, il suffit d'exécuter le script de compilation dans ModelSim. Pour ce faire, il faut se placer dans le répertoire bin du niveau cellule.

>> cd **REP_PROJET$/DreamWafer/trunk/cell/sw/bin**

Pour démarrer ModelSim :

>> stmodelt

Ensuite taper :

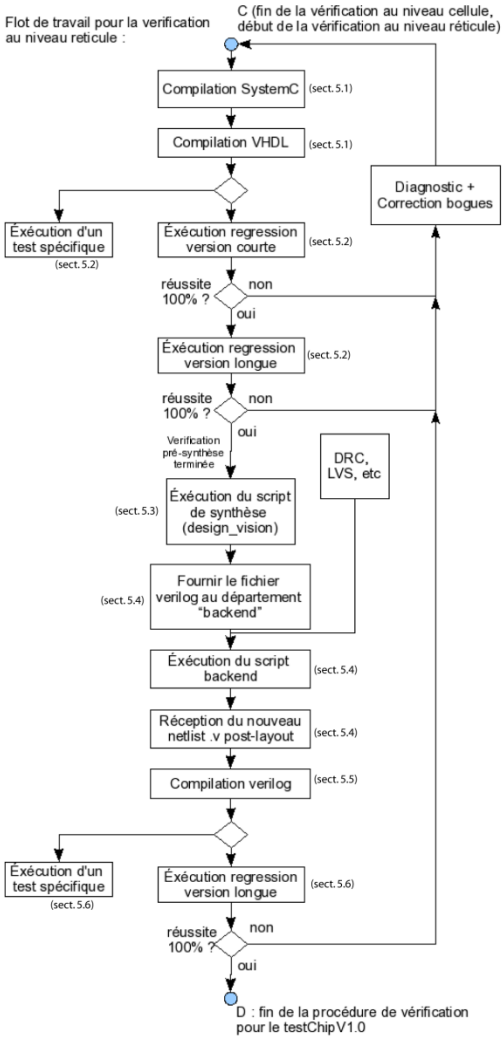
>> do compile_vlog_wic.do

Exécution d'un test spécifique ou exécution du script version longue

L'exécution d'un modèle verilog se fait de la même façon qu'avec un modèle vhdl, seulement le nom du script d'exécution diffère. Évidemment, les commandes du script

d’exécution du modèle verilog sont différentes du modèle vhdl, mais les scripts automatisent ces tâches. Le script pour exécuter un modèle peut être, par exemple, le fichier vlog_debug_cell_top_post_layout.do.

Pour activer un ”test case” en particulier, il suffit de modifier le fichier testcase.txt comme à la section 4.3.



Les prochaines sections décrivent en détail la procédure à suivre pour refaire la vérification du code d'un réticule du DreamWafer. Le niveau hiérarchique du réticule se trouve dans le répertoire

REP_PROJET$[3]/DreamWafer/trunk/reticle/ et le code source associé au script d'exécution ModelSim se trouve dans le répertoire :

REP_PROJET$/DreamWafer/trunk/reticle/sw/bin

Compilation SystemC, compilation VHDL

La compilation du système de vérification se fait en 2 étapes. La première étape est la compilation du code VHDL qui se fait à l'aide d'un script nommé compile_vhdl_wic.do. Pour exécuter le script, il faut ouvrir une fenêtre ModelSim dans le répertoire /nanopads/sw/bin et taper

```
>> do compile_vhdl_wic.do
```

Pour compiler le SystemC, il faut ouvrir une fenêtre ModelSim et taper la commande suivante :

```
>> do clean_and_compile_all.do
```

La compilation doit durer plusieurs minutes. Pour que la compilation puisse réussir, il faut que les paramètres génériques du modèle VHDL et du modèle SystemC soient cohérents. Par exemple, la grandeur du réticule doit être la même dans le modèle VHDL et dans le modèle SystemC. Pour vérifier que les valeurs sont cohérentes, il faut vérifier les fichiers /reticle/rtl/wafer_ic_constants.vhd. Toutes les variables sous le pragma "under_python_control_on" sont susceptibles d'être des variables à harmoniser avec le SystemC. Le fichier /reticle/sw/src/def_system.h est à harmoniser avec le vhdl.

Une fois la compilation terminée avec succès, le modèle est prêt à être exécuté.

Une autre méthode simple pour compiler l'environnement SystemC est d'utiliser un script python qui vient s'assurer que tous les fichiers de configuration du modèle sont adéquatement forcés à une valeur cohérente pour la simulation niveau cellule. Pour

---

[3] **REP_PROJET$ = le répertoire où l'on a fait un svn checkout du projet DreamWafer**

exécuter le script il faut se rendre dans le répertoire /bin du niveau hiérarchique du réticule (/reticle/sw/bin) et taper sur un terminal (et non dans ModelSim).

```
% python recompile_all.py
```

Pour configurer la grandeur en terme de cellule du réticule, il faut ouvrir le fichier "recompile_all.py" et choisir le nombre de cellules du réticule en forçant une valeur sur les variables python. Voici les variables python à configurer :

1 - "nb_i_wic" : le nombre de réticules en i. Cette variable doit rester égale à 1.

2 - "nb_j_wic" : le nombre de réticules en j. Cette variable doit rester égale à 1.

3 - "nb_i_ret" : le nombre de cellules en i dans le réticule. Cette variable peut être configurée à la valeur choisie.

4 – "nb_j_ret" : : le nombre de cellules en j dans le réticule. Cette variable peut être configurée à la valeur choisie.

Exécution du script version courte et longue et exécution d'un test spécifique

Il existe 43 test cases au niveau réticule, dont 321 tests spécifiques. Exécuter tous ces tests peut prendre un temps considérable. Pour faciliter le débogage du système, nous avons mis en place plusieurs scénarios de test de régression. Parmi ces scénarios, nous avons une version courte du test de régression. Cette version courte peut être exécutée de façon indépendante du test de régression globale. Pour exécuter la version courte, il faut taper à la ligne de commande

% python regression_script_jt_short_reticle.py (pour la vérification du système JTAG)

Ou

% python regression_script_wf_short_reticle.py (pour la vérification du WaferNet)

La version longue du test de régression peut être exécutée de façon indépendante du test de régression globale. Pour exécuter la version longue, il faut taper à la ligne de commande

% python regression_script_jt_long_reticle.py (pour la vérification du système JTAG)

Ou

% python regression_script_wf_long_reticle.py (pour la vérification du WaferNet)

La version longue comprend la totalité des tests du plan de test.

Pour exécuter un test spécifique, il suffit de procéder à la même suite de commandes que pour le niveau hiérarchique de la cellule (voir section 4.3). Un script facile à lancer est verif_tc_3x3.do qu'il est possible de lancer pour simuler un test case quelconque avec un réticule de 3x3. Ce script comprend la visualisation d'un grand nombre de signaux pertinents. Attention de bien compiler le SystemC et le VHDL en configurant le modèle avec un réticule de 3x3. Tout comme dans le niveau cellule, l'environnement de vérification lit le fichier testcase.txt et active le code du "test case" choisi.

Exécution du script de synthèse avec design_vision

Pour synthétiser un réticule, nous utilisons la technique suivante :

1- Compilation d'une boîte noire portant le même nom d'instance qu'une cellule boîte blanche dans design_vision.

2- Compilation et élaboration d'un réticule 3x3. Le synthétiseur fera le lien avec les boîtes noires, ce qui permet d'avoir un netlist pré-layout léger et simple de la structure interne de la cellule.

Le script de synthèse se trouve

REP_PROJET$/DreamWafer/trunk/reticle/syn/bin/synth_ret_V1p0_3x3.scr

Important : ce script doit être exécuté dans l'environnement de travail de design_vision. Premièrement, il faut faire lancer design_vision à partir du répertoire reticle/syn/run. Deuxièmement, faire file-> execute script dans l'interface design_vision. Ensuite, choisir le fichier synth_ret_V1p0_3x3.scr. Le fichier généré par le script se retrouve dans le répertoire reticle/syn/out/reticle_full_3x3.v.

Backend

Le script de synthèse du réticule génère une netlist qui est ensuite envoyée au département backend de l'équipe DreamWafer géré par Olivier Valorge. Un script "backend" est exécuté dans l'environnement de travail d'Olivier pour générer un 2^{e} netlist verilog plus complet appelé "netlist post-layout". Ce netlist est soumis dans la structure de répertoire svn à l'emplacement suivant :

REP_PROJET$/DreamWafer/trunk/reticle/sw/backend/PR_data/reticle_full_PR.v

Le répertoire PR_data contient aussi des fichiers d'annotations de délais "pire cas", et "typique" (wr_cell_top_wr.sdf et reticle_full_typ.sdf).

Le fichier wr_cell_top_PR.v du répertoire "/cell/sw/backend/PR_data" est utilisé dans le script de compilation verilog associé au réticule. Ce fichier n'est pas prêt à être simulé par ModelSim. Il faut effectuer une série de modifications, les mêmes que celles effectuées

pour la cellule. Ces modifications se font manuellement ou bien automatiquement avec le script de régression global décrit dans le document 08.07.22.Regression.doc.

Compilation verilog

Une fois les modifications effectuées au fichier verilog post-layout, la compilation peut être effectuée. Pour compiler le fichier verilog, il suffit d'exécuter le script de compilation dans ModelSim. Pour ce faire, il faut se placer dans le répertoire bin du niveau réticule.

>> cd **REP_PROJET$/DreamWafer/trunk/reticle/sw/bin**

Pour démarrer ModelSim :

>> stmodelt

Ensuite, taper :

>> do compile_vlog_wic.do

Script version longue (verilog) et exécution d'un test spécifique du modèle verilog

L'exécution d'un modèle verilog se fait de la même façon qu'avec un modèle vhdl, seulement le nom du script d'exécution diffère. Évidemment, les commandes du script d'exécution du modèle verilog sont différentes du modèle vhdl, mais les scripts automatisent ces tâches. Un script peut être par exemple le fichier vlog_debug_ret_post_layout.do dans le répertoire reticle/sw/bin.

Pour activer un test case en particulier, il suffit de modifier le fichier testcase.txt comme à la section 4.3.

ANNEXE B – PLAN DE TEST DU TESTCHIP V1.0

Table B-1 Plan de test du TestChip V1.0

Test case name	Nombre de cas de test	Nb. de cas de test passé avec succès	Description	Testé ?	Succès ?
JT_TEST_FCLING_TDO_1	1	1	Test the control signal instead of tdo	YES	YES
JT_GND_AND_VDD	2	2	Test nanopads power state configurations (GND and VDD)	YES	YES
JT_TESTWFNET_L1_3v3	4	4	Test wafernet links length of 1, at 3.3V output	YES	YES
JT_TESTWFNET_L2_3v3	4	4	Test wafernet links length of 2, at 3.3V output, 2 BALLS	YES	YES
WF_ONE_TO_ALL_3v3	1	1	Wafernet test including a max. of activated links in WF, at 3v3	YES	YES
JT_TESTWFNET_L1_1v8	2	0	Test wafernet with links length of 1, at 1.8V output	YES	NO
JT_TESTBSCAN_OUT	1	1	Test the boundary scan output state	YES	YES
JT_EXTCNTRL	6	6	Test the external control capabilities of the cells	YES	YES
WF_CHIPSELECT_1	1	1	Test nanopads for on-the-fly input/output redirection mode	YES	YES
JT_CONTACTDETECT	20	20	Test the contact detection features (see note 1)	YES	YES
JT_ALLPATH	4	4	Test all JTAG inter-cellular path of the TestChip	YES	YES
WF_FLOATING_REP	8	8	Observe unconnected repeaters (input of reticle)	YES	YES

WF_FLOATING_REP_S	1	1	Observe unconnected repeaters (input of reticle) with one output as VDD 3.3V.	YES	YES

Total number of test cases (#Total_TC) :	55
Total number of test cases passed (Passed_TC) :	53
Total number of test cases failed :	2
Passed test case ratio (#Passed_TC)/(#Total_TC)	0.96

ANNEXE C – “FLOWCHART” ET AUTRES SCHÉMAS

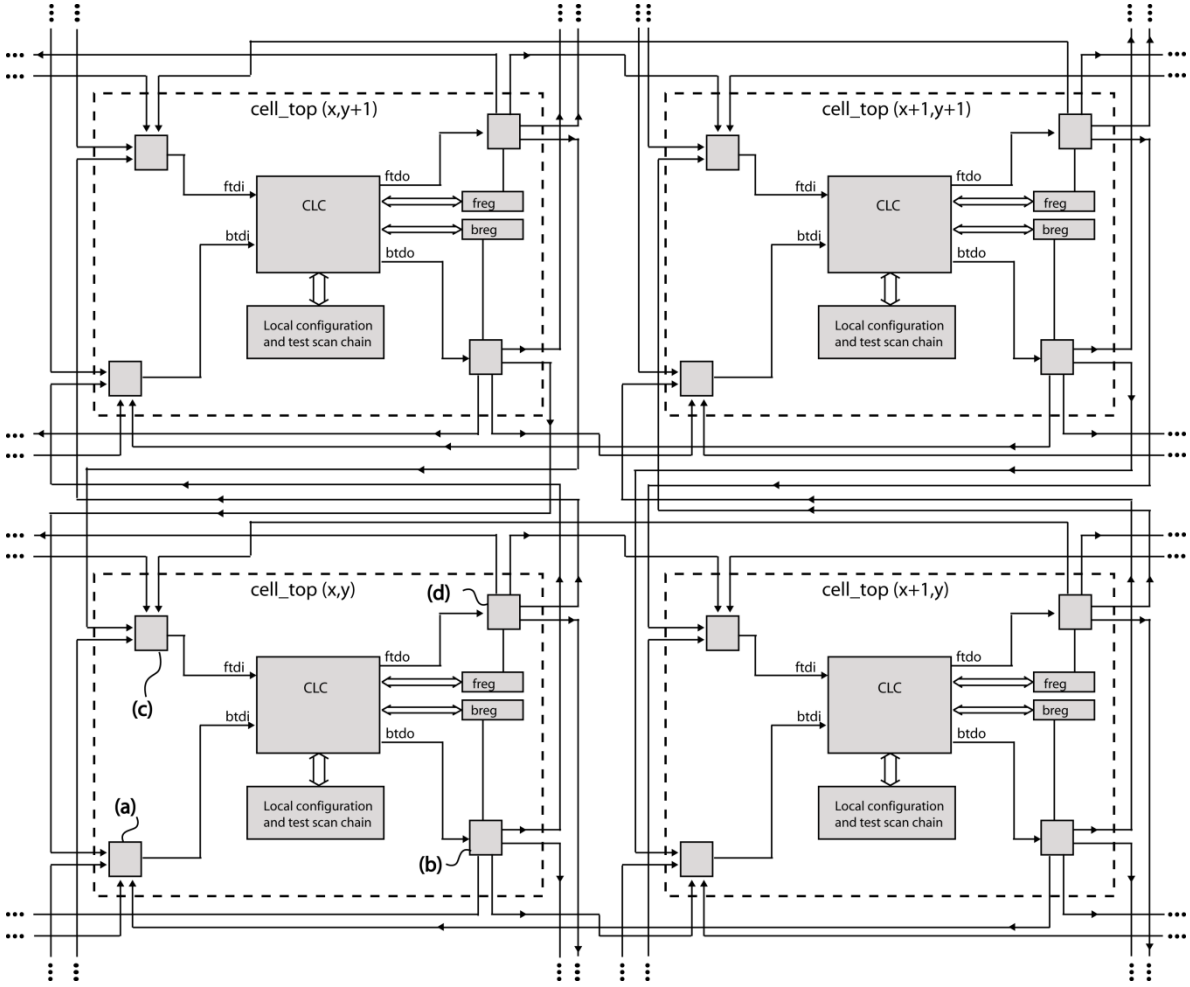


Figure C.1 Architecture intercellulaire pour le BCIC

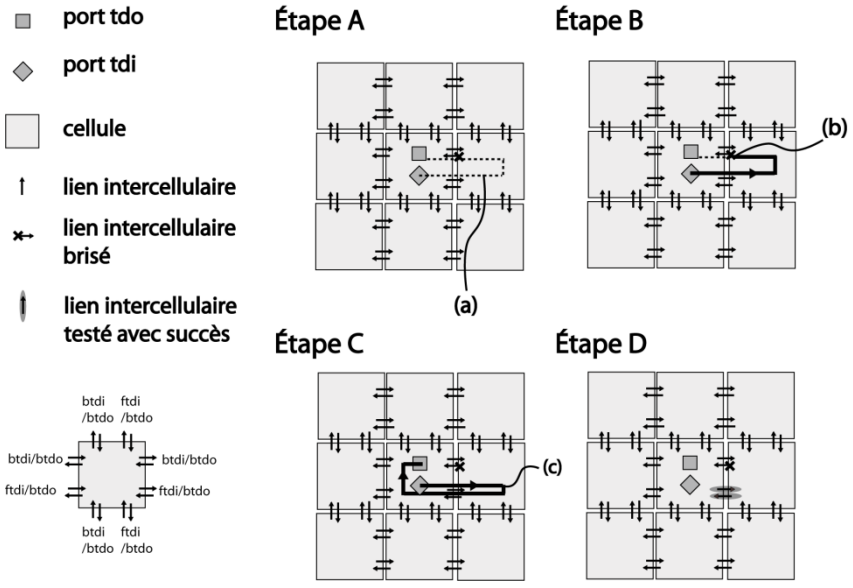


Figure C.2 Exemple de diagnostic pour l'architecture BCIC

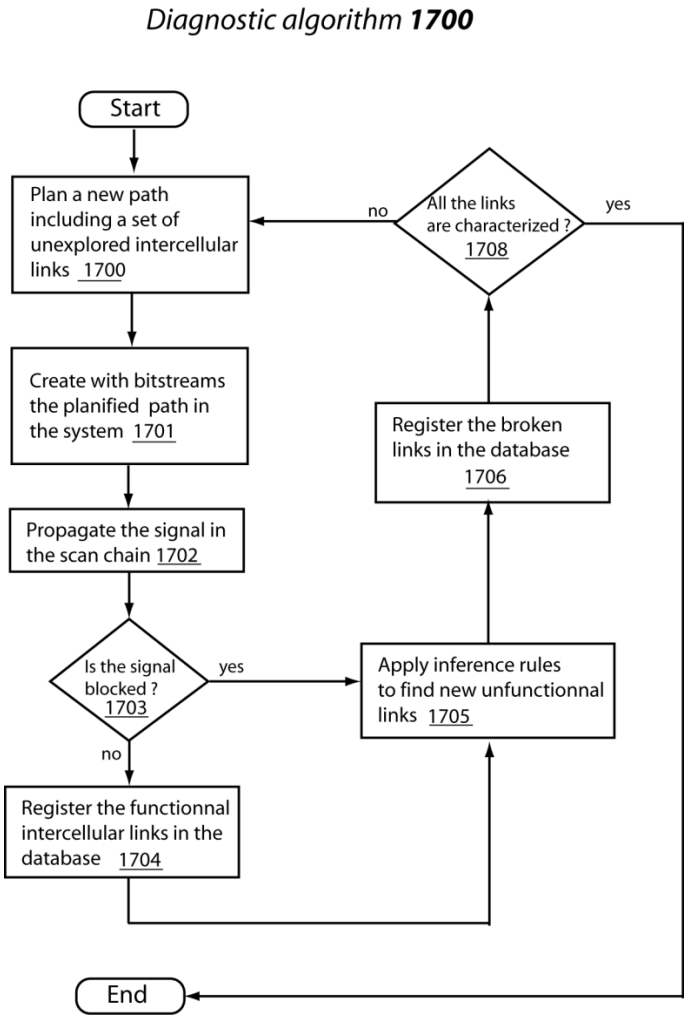


Figure C.3 Algorithme de diagnostic de fautes dans les liens intercellulaires

(La figure C.3 a été extraite de la divulgation en cours d'écriture.)

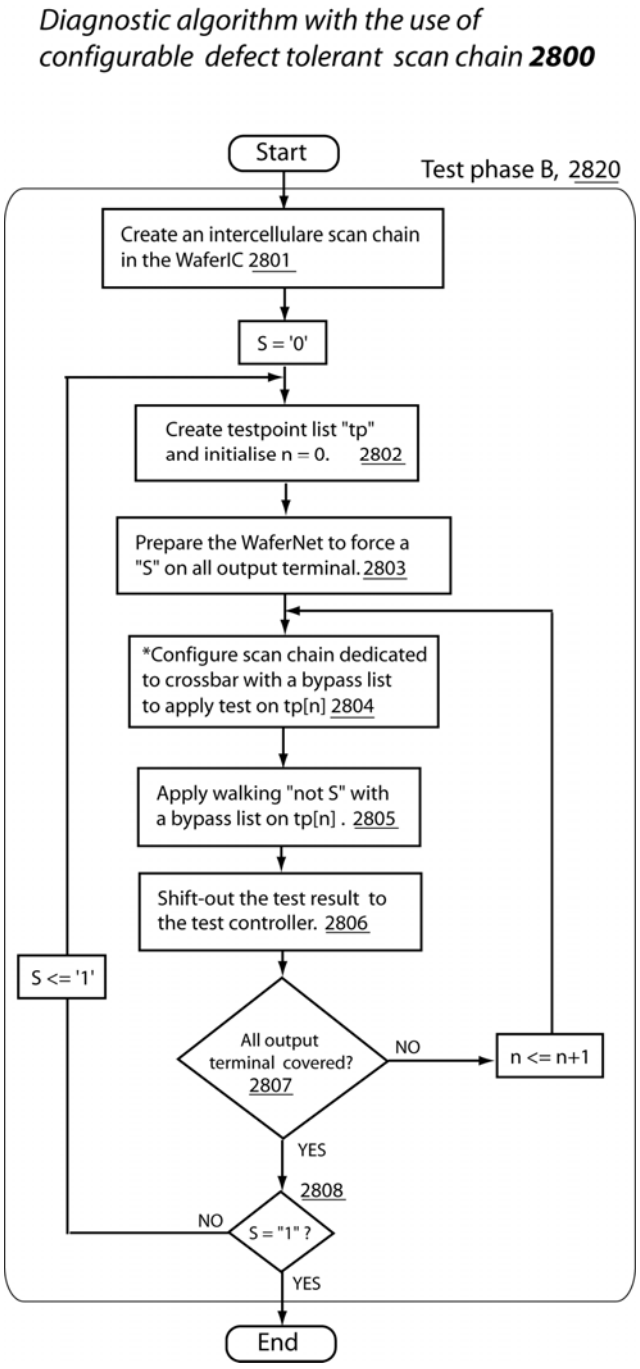


Figure C.4 Algorithme optimisé pour le diagnostic de fautes dans le WaferNet

(La figure C.4 a été extraite de la divulgation en cours d'écriture.)

ANNEXE D – PRINCIPAUX SCHÉMAS BLOCS

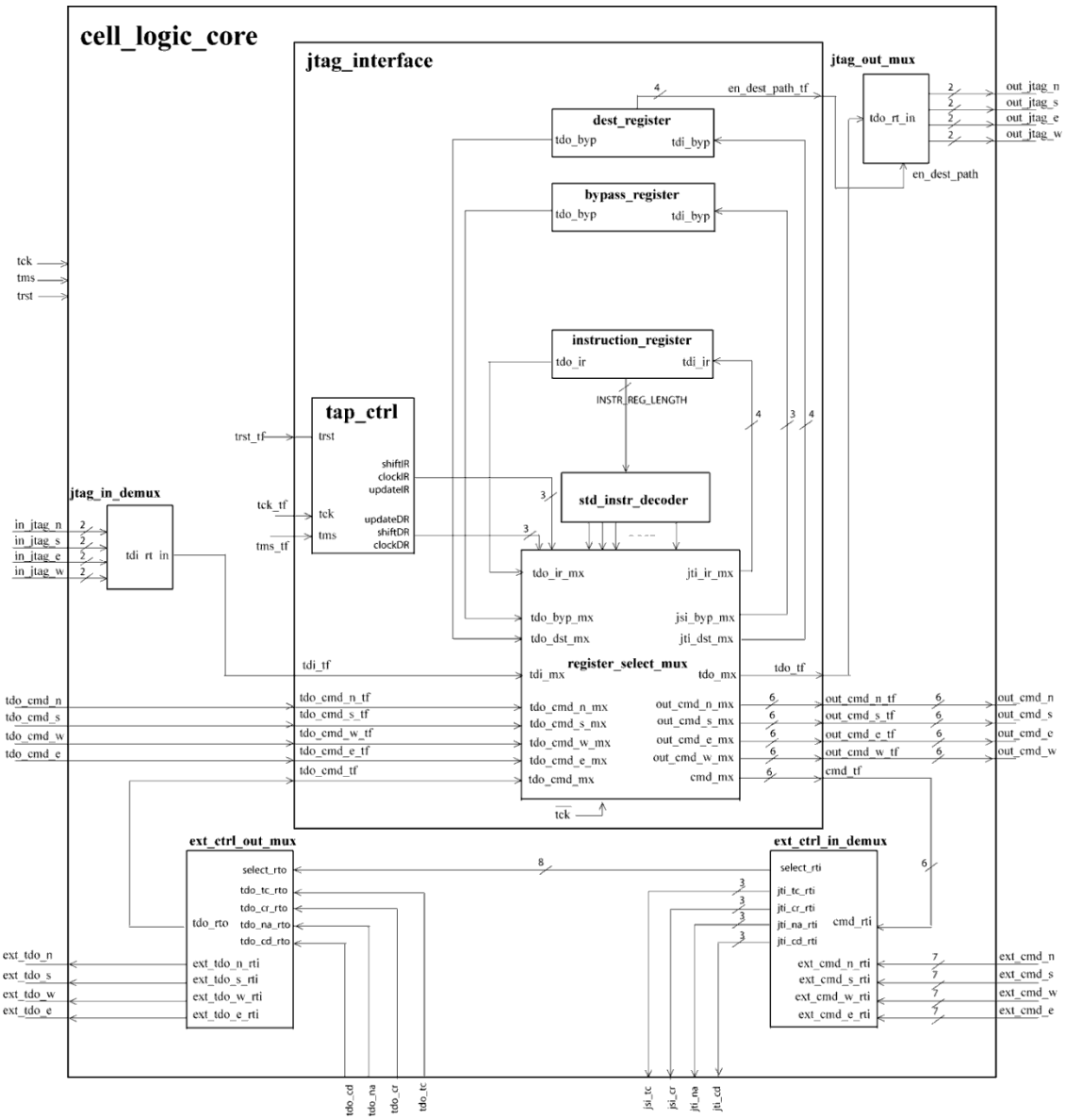


Figure D.1 Description structurale détaillée (VHDL) de l'entité cell_logic_core

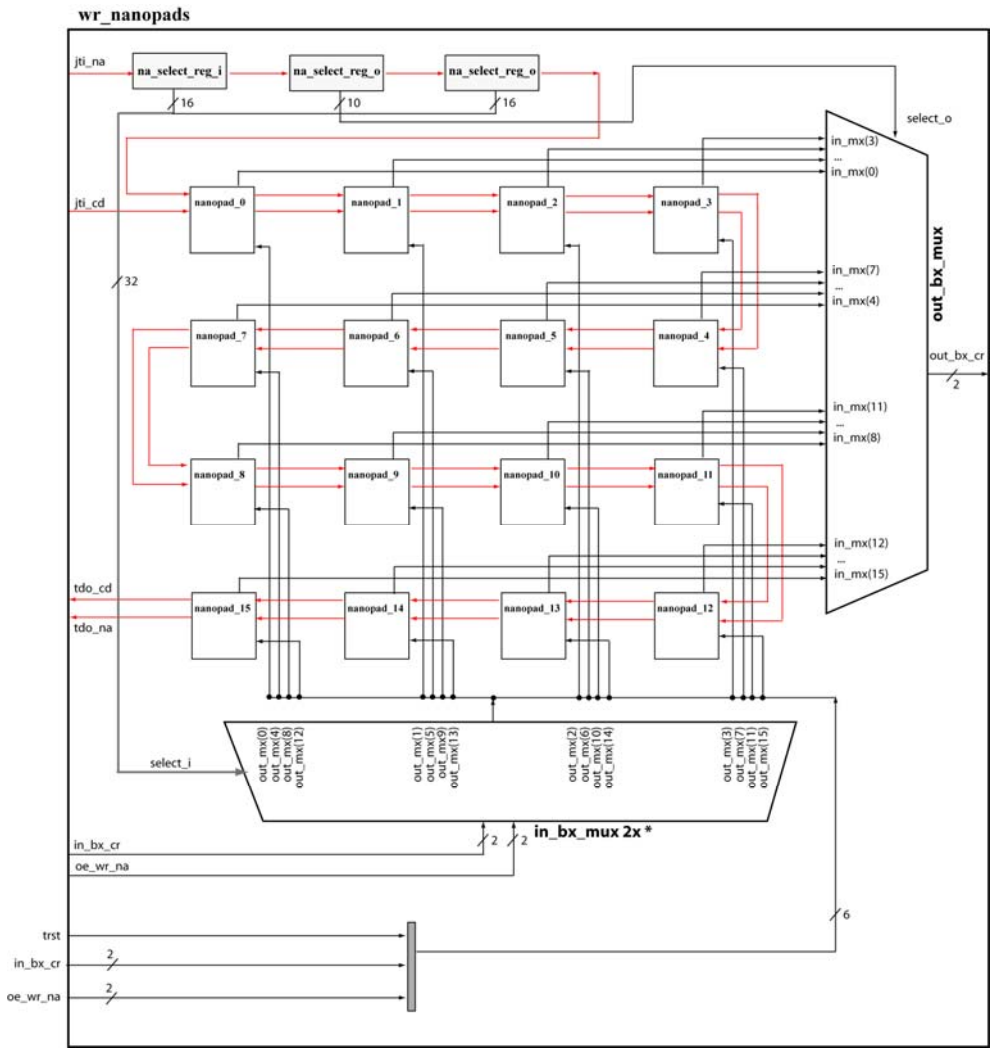


Figure D.2 Description structurale détaillée (VHDL) de l'entité wr_nanopads

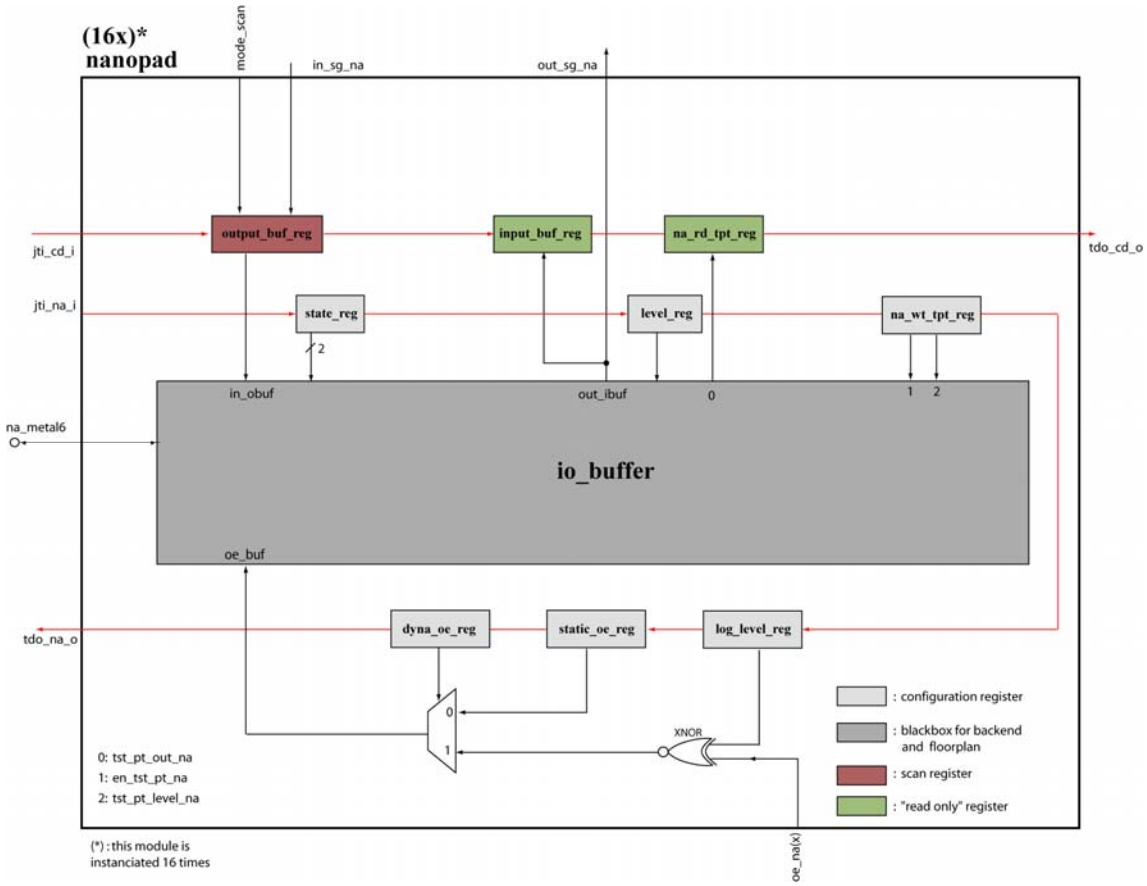


Figure D.3 Description structurale détaillée (VHDL) de l'entité NanoPad

ANNEXE E – ARTICLES DE CONFÉRENCES

Faults Diagnosis Methodology for the WaferNet Interconnection Network

Yan Basile-Bellavance[1], Yves Blaquière[2] and Yvon Savaria[1]

[1]: GR2M, École Polytechnique de Montréal,

[2]: Département d'Informatique, Université du Québec à Montréal

Abstract **- In this paper, the interconnection network (WaferNet) which is part of an active and reconfigurable prototyping board, named WaferBoard™, is analyzed to derive efficient defect diagnosis. The WaferNet structure spans an entire silicon wafer that inevitably contains defects, due to the nature of the microfabrication process, and defect management strategies are inserted in the design flow. Defects must be accurately located to efficiently reconfigure the circuit around them. Key differences between a conventional printed circuit board and WaferNet justify the proposed diagnosis methodology. A sequential walking-one algorithm and a broadcast algorithm are proposed to locate shorts or stuck-at faults in the network. It is shown that dedicated hardware architectures must be integrated in the network to locate those defects in a reasonable time. Analysis shows that the proposed diagnosis time complexity is $O(n^2)$, where n is the number of cells in the matrix. An upper bound time limit is calculated that depends on both the size and the number of faults in the circuits.**

I. INTRODUCTION

This paper proposes an architecture and methodology for diagnosis of shorts and stuck-at defect in a recently proposed wafer-scale reconfigurable mesh network called WaferNet. This network is part of the WaferBoard™ technology [1] for rapid prototyping and validation of digital systems. It embeds a reconfigurable active substrate called WaferIC™ [2] that is obtained from photo-repetition of reticles. Each reticle contains an array of cells and thousands of configurable NanoPads allowing for easy alignment-insensitive placement of integrated circuits (ICs) on its surface. Any IC pin can then be actively interconnected through the configurable WaferNet. Each cell is interconnected with its neighbor cells in the four directions (Fig. 1). Each cell contains two programmable structures: a crossbar and an array of 4×4 NanoPads, interconnected as shown in Fig. 1 and Fig. 2. Each cell can redirect at most two IC pin signals, contacted with NanoPads, to other cells. All cells are daisy-chained and configured through custom scan chains (Fig. 1(f) and (g)).

The WaferNet architecture is detailed in [3], but no work on its diagnosability has been published. The problem of WaferNet diagnosis is similar to testing FPGA interconnect resources covered in [4]. However, the WaferIC™ does not contain programmable logic blocks, but only configurable crossbars and configurable I/O buffers. Furthermore, WaferNet spreads an entire wafer rather than a large silicon die. It is well known that wafer-scale circuits require defect tolerance to make them economically feasible [5].

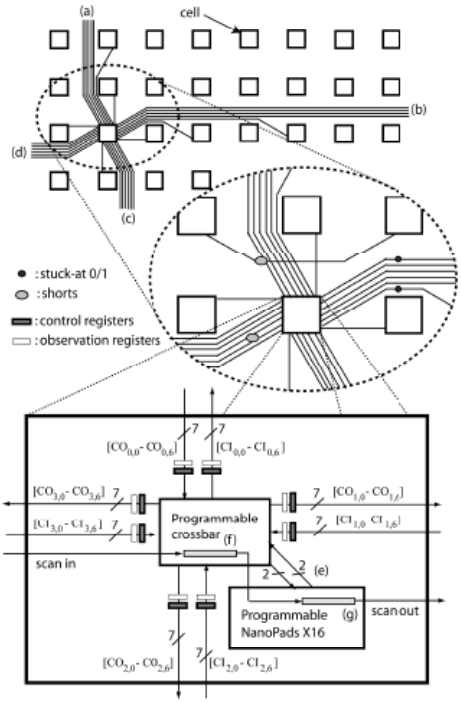


Fig. 1. Hierarchical structure of the WaferNet. The intercellular unidirectionnal network is deployed in 4 directions: (a) North, (b) East, (c) South and (d) West. A scan chain spans all the cells of each reticule of the WaferIC ((f) and (g)).

A dedicated tests and diagnosis method must be designed for the WaferNet architecture that differs from classical checker-board applied to PCB. This paper characterizes the resources added to the WaferNet to make it affordably diagnosable. It also covers the algorithms that allow detecting stuck-at 0/1 and short defects (affecting both parallel and perpendicular directions, as shown on Fig. 1) in interconnects. Only stuck-at 0/1 are diagnosed in the crossbars and no short diagnosis in the crossbar is done. Wafer-scale diagnosis is also addressed in this paper, which brings more stringent time and area constraints.

Section II details the proposed diagnosis methodology for the WaferNet. Section III analyzes its time complexity. Diagnosis time depends on both the number of scan chains

dedicated to diagnosis and on the number of faults in the network. Our main conclusions are summarized in Section IV.

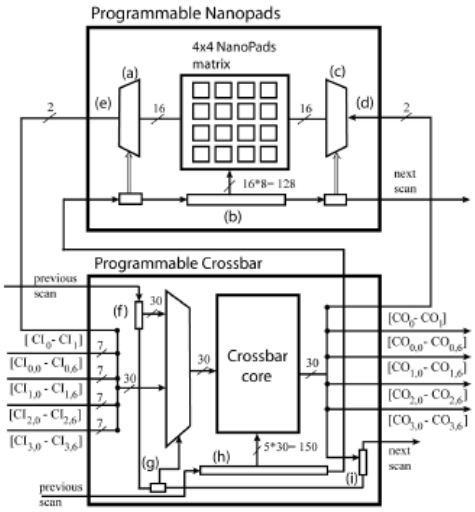


Fig. 2. Cell structure with its two scan chains : configuration scan ((h) and (b)), test and diagnosis scan ((f), (g) and (i)).

II. WAFERNET DIAGNOSIS METHODOLOGY

Defects in wafer scale integrated (WSI) systems are unavoidable, therefore defect tolerance is a must. The WaferNet defect tolerance is based on a defect-aware design flow [6]. Such design flow includes a software based, "on customer site" diagnosis step to extract a defect map. Then, a defect-aware place-and-route algorithm uses this map to avoid those defects. The test and diagnosis resources embedded in the network are described next, followed by the description of the diagnosis algorithm. In the following, it is assumed that all scan chains are defect tolerant and their design is beyond the scope of the present paper [7].

A. Implemented Test and Diagnosis Architecture

Each cell in the WaferIC™ has a programmable 30×30 crossbar and a programmable array of 4×4 NanoPads, as depicted in Fig. 2. The crossbar has 7 incoming links and 7 outgoing links in each direction, such as the incoming links $CI_{0,[0..6]}$, $CI_{1,[0..6]}$, $CI_{2,[0..6]}$, and $CI_{3,[0..6]}$ respectively in the N-E-S-W physical directions. The interconnection lengths grow according to a geometric series (1, 2, 4, 8, 16, 32, and 64 cells distances). The crossbar has 2 more inputs and outputs connected to the possible two uIC balls supported by the cell.

Each NanoPad includes a programmable I/O buffer connected to a boundary scan register. Thus, a total of 16 test and diagnosis registers are needed for the 4x4 NanoPads. The I/O buffer can be configured as power (VDD or GND), an input or output buffer. Each NanoPad has an 8 bits I/O configuration register (Fig. 2(b)) and groups of 16 NanoPads have a 20 bits state controlling multiplexers/ demultiplexers that configure the network (respectively Fig. 2(a) and 2(c)).

Due to tight WaferIC™ area constraints, a custom scan chain has been used for configuration and diagnosis rather than automatic insertion of a usual Design-For-Testability scan chain. On the other hand, Fig. 1 shows a variation of the implemented test and diagnosis architecture, where control/observation registers are inserted on both crossbar input and output. This solution can reduce the WaferNet diagnosis to a classic checkerboard algorithm [8]. At the cost of a more complex diagnosis algorithm, the implemented test and diagnosis architecture minimizes the area with a scan of 30 control registers on the crossbar input and 30 observation registers on the crossbar output. (Fig 2 (f), (g) and (i)). One-bit control register and a multiplexer (Fig. 2 (g)) are then used to configure the crossbar in normal mode or diagnosis mode. The mux-based crossbar (Fig. 2) [3] requires 5 bits per output for a total of 150 bits (Fig. 2(h)). Finally, each cell has 298 registers for configuration and 77 registers for test and diagnosis, as summarized in Table 1.

TABLE 1 : WAFERIC'S CELLS REGISTER USAGE

Cell Module	Purpose	Nb. of registers
Crossbar	Configuration	150
Crossbar	Test and diagnosis	61
NanoPads	Configuration	148
NanoPads	Test and diagnosis	16

B. Determinist Diagnostic Algorithm

The proposed diagnosis methodology is applied in 3 phases composed of tests called Type A, B, and C. Test Types A and B are depicted in Fig. 3, Test Type C in Fig. 4. For illustration, several possible fault locations (Fig. 3 (a),(b),(c),(d)) are shown and their resulting effect on the output signature is illustrated.

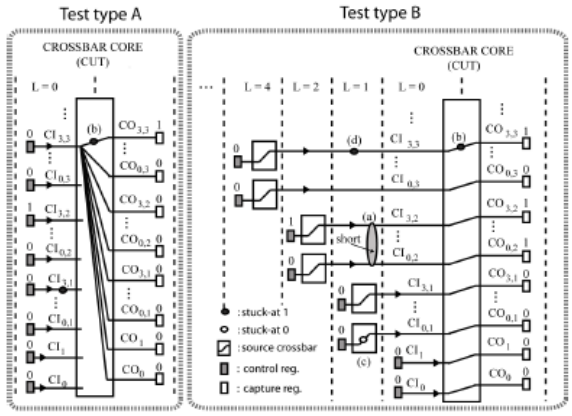


Fig. 3. The required network and CUT's configuration (CUT is in the observation state) in order to isolate faults in the networks. Several possible fault locations (a, b, c, and d) are shown in the figure for reference.

In Fig. 3 and 4, L = (0, 1, 2, 4…) is used to denote each zone of the network from the point of view of the circuit-under-test

(CUT). For example, a long interconnect between a distant crossbar source and the $CI_{3,3}$ input terminal crosses 4 cells (L=4), and the control scan coming from the input port CI_0 or CI_1 does not cross any cell (L=0) because it is included in the CUT.

1) Test Type A: This test leverages local control and observation registers, and for each '1' applied on a crossbar input port, the crossbar is configured in broadcast mode (one-to-all configuration). For example, if a fault (3 (b)) is located in zones h1, h2 of the input port $CI_{3,3}$, then the fault can be observed when the broadcast is applied to port $CI_{3,3}$. This test phase can be applied concurrently on all CUTs.

2) Test Type B: It consists of applying the well known walking '1' algorithm [8], which reveals short and open faults on each interconnect as well as any stuck-at defects. With the proposed method, shorts can only be diagnosed between network interconnects and not between any trace outside of the network. A walking '1' (and walking '0') sequence is applied at the transceiver side of interconnects using control scan registers, signals are then captured at the crossbar outputs in the observation registers and shifted out for diagnosis.

To diagnose a short or a stuck in the network, it is necessary to shift out the content of the CUT observation register outside of the WaferIC to a test controller for each position of the walking-one. During the application of the walking '1', it is required to force a '0' on all control registers of the network. Such precaution enables the diagnosis of shorts on any pair of interconnect (parallel or perpendicular). Two other factors must be taken into account for adding concurrency to the test type B:

The concept of influence cone: for each CUT, there is a zone where it is required to force a '0' to avoid the overlap of potential multiple shorts in the network. Such influence cone is limited by the maximal extent of the incoming links times two (64 cells).

The resulting required gap between concurrent CUTs : each concurrent CUT must keep a security gap between two CUTs to avoid shorts overlap. A security gap of 2 times 64 cells is sufficient.

3) Test type C: To narrow down the possible locations of faults (shorts or stucks), further test sequences are needed. This test sequence is applied only where the CUT appears to contain faults detected with test type A or B. Test type C consists of a global broadcast test over long interconnects. The same one-to-all configuration is forced on the CUT. The difference comes from control registers. Instead of using local control register, the test uses control registers coming from the distant cells at the other end of the long interconnect. For example, there is an overlap between the two stuck-at-one defects 'd' and 'b' (Fig. 3 (d) and (b)). These overlapped faults propagate only to port $CO_{3,3}$ on test type B. To reveal fault 'd', test type C is used. If a '1' is seen on every capture register of the CUT, as shown in Fig. 4, then it is possible to conclude that a stuck-at-one exists on the interconnect (Fig. 4(d)).

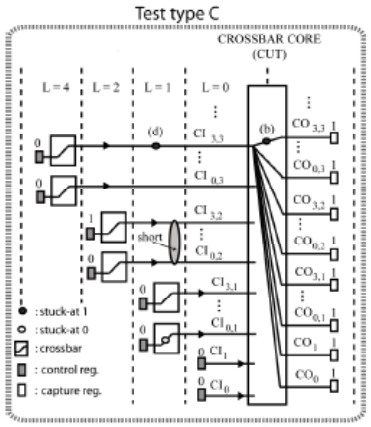


Fig. 4: Test type C is applied only on suspicious cells. An interconnect defect (d) result in capturing a '1' on any capture register.

C. Diagnosis Resolution with reduced overhead

Overhead is the ratio of the area dedicated to test and diagnosis compared to the total area used by the circuit under test. Each cell of the network occupies about 540×540 µm, and 50 % of its area is dedicated to digital circuits; the remaining area is occupied by power management and analog circuits. Significant efforts were dedicated to reducing the complexity of test and diagnosis circuits, with only 61 single-bit registers and some configurable 2 to 1 multiplexers dedicated to the task. The proposed design can be generalized to arbitrary size regular network of crossbar for SoC or wafer scale system.

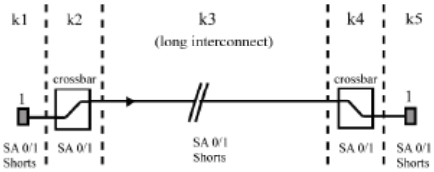


Fig. 5: a generalized definition of the diagnosis resolution in a regular network of interconnected crossbar.

Fig. 5 illustrates specific parts of the network structure (k1 to k5) under test. They define the considered granularity for diagnosis purpose. An extensive analysis of the diagnosis resolution of the proposed method in the presence of multiple shorts or multiple stuck at (SA) is beyond the scope of this paper. If the number of faults is low, for some mature microfabrication process, the probability to have multiple interacting faults on some interconnects is low. In such situation, the proposed method allows to diagnose faults located in the k1, k3 and k5 regions. The proposed diagnosis methodology is unable to differentiate shorts between k1, k3 and k5. On the other hand, SAs can be easily differentiated as shown in the test type C section. Regions k2 and k4 are only diagnosed for SA.

III. Complexity Analysis and Results

The WaferIC consists of a concatenation of regular cells in a matrix ($m \times m$) included on each reticule of a whole wafer. The number of JTAG ports on each reticule has been fixed to two. Diagnosis of reticules can be executed concurrently (i.e. star JTAG architecture). Each cell contains 4 types of registers, as shown in Table 1, for a total of 375 registers per cell. For every partitioned scan chain, all register cells are connected together, and all cells are connected together by inter-cellular scan chains forming a global reticule scan chain as shown in Fig. 1 and 2. The most efficient multiple scan chain partition (i.e. 2) is implemented as follows:

- Scan chain 1: every crossbar test registers (i.e. 61) are daisy chained together. The total number of registers spanning an entire reticle (R_1) is $R_1 = 61\ m^2$.
- Scan chain 2: the remaining configuration registers (375-61=314) are daisy chained together. The total number of registers spanning an entire reticle is $R_2 = 314\ m^2$.

Such partitioning allows configuration or diagnosis to be done independently to improve efficiency of both operations. The diagnosis time depends on the clock period p.

1) Diagnosis time for test type A: The total test time (T_A) for the "Test type A" is $T_A = 2(30)R_1 p$. There are 30 input terminals in each crossbar, thus, 30 test vectors is applied to the crossbar. The factor 2 is introduced in the diagnosis time because the broadcast algorithm includes two passes forcing '1' then '0' on each input terminal. Each test can be done concurrently because the test is local.

2) Diagnosis time for test type B : The total test time is $T_B = (2(64/m)+1)^2(R_1V_B+m^2R_2)p$. The term (R_1V_B) defines the number of '1' of the walking sequence applied to every output terminals ($V_B = 30\ m^2$) times the number of registers in the network. The term (m^2R_2p) represents the total number of crossbar re-configurations. One reconfiguration is needed for every CUT in the network. The factor $(2(64/m)+1)$ is the number of reticles included in the influence cone to avoid overlap and to keep the diagnosis integrity. The influence cone is in fact a square, thus a power of 2 is applied to this slow down factor.

3) Diagnosis time for test type C: The number of test vectors for test type C is determined by the number of faults per reticle, f, (short or SA). The number of test vector (V_C) is $30f\,R_1$. Only the worst case is considered, when there is only one fault per CUT, which required a new test for every fault discovered. Thus, $T_C = (30)2f\,(R_1+R_2)p$. Similarly to the diagnosis time for test type B, there is one crossbar reconfiguration for every CUT. This is why R_2 appears in the formula. No concurrency is applied to test type C.

The total diagnosis time $D = T_A + T_B + T_C$ for a clock period p of 10MHz is summarized in Table II. The total diagnosis time is dominated by the test type B, which grows as $O(m^4)$. And the diagnosis time for the test type A is the order of microseconds and is negligible.

TABLE II :DIAGNOSIS TIME D (@10 MHZ)

m	$T_A + T_B$ (s)	Nb. of fault short or SA per reticle (f)	T_C (s)	D (s)
8	507.7	0	0	507.7
		30	4.3	512
		60	8.6	516.3
16	2276.6	0	0	2276.6
		30	17.3	2293.9
		60	34.6	2311.2

IV. Conclusions

This paper proposed a deterministic diagnosis methodology for the new WaferNet reconfigurable network. The WaferNet needs a specific diagnosis strategy, which differs from related classical PCB diagnosis, such as the checkerboard algorithm. Time complexity analysis is demonstrated to be $O(n^2)$, where n is the number of cells in the matrix ($n = m^2$). Furthermore, the diagnosis time depends on the number of faults found with test type C. The proposed method can be useful for any type of regular network of crossbar for SoC or wafer scale integration with a relatively low diagnosis test overhead. Research is ongoing to improve diagnosis efficiency by applying test concurrency with more sophisticated algorithms.

Acknowledgements

The authors would like to thank NSERC and Gestion TechnoCap Inc. for their financial support, and CMC Microsystems for providing design tools and support.

References

[1] Norman, R., U.S. Patent Application Number 11/611,263.
[2] R. Norman, O. Valorge, Y. Blaquiere, E. Lepercq, Y. Basile-Bellavance, Y. El-Alaoui, R. Prytula, Y. Savaria, "*An Active Reconfigurable Circuit Board*", NEWCAS 2008
[3] Lepercq E., Valorge O., Basile-Bellavance Y., Savaria Y., Blaquière Y., "*An Interconnection Network for a Novel Reconfigurable Circuit Board*", NEWCAS 2008, June, pp. 129-132, 2008.
[4] A. Doumar, H. Ito, "*Detecting, diagnosing, and tolerating faults in SRAM-based field programmable gate arrays: a survey*", IEEE Trans. on VLSI Systems, vol. 11, no. 3, June 2003.
[5] Brewer, J.E., "Promises and Pitfalls of WSI", in Wafer Scale Integration, Kluwer, pp.1-29, 1989.
[6] R. Jain, A. Mukherjee, and K. Paul, *"Defect-aware design paradigm for reconfigurable architectures,"* in Proc. of the 2006 emerging VLSI technologies and architectures, Karlsruhe, Germany, Mar 2-3, 2006.
[7] M. Lu, Y. Savaria, B. Qiu, J. Taillefer, "*IEEE 1149.1 based defect and fault tolerant scan chain for wafer scale integration,*" 18 th Proc. IEEE International Symp. On Defect and Fault Tolerance in VLSI systems, 2003.
[8] A. Hassan, J. Rajski, V. K. Agarwal, "*Testing and Diagnosis of Interconnects using Boundary Scan Architecture,*" 1988, International Test Conference.

Hardware/Software System Co-Verification of an Active Reconfigurable Board with SystemC-VHDL

Yan Basile-Bellavance[1], Étienne Lepercq[1],
Yves Blaquière[2] and Yvon Savaria[1]

(1) GR2M, École Polytechnique de Montréal,
(2) Département d'Informatique, Université du Québec à Montréal

Abstract— **This paper reports on the challenges encountered, and the solutions adopted in the hardware and software co-verification of a wafer-scale integrated (WSI) system. The WaferBoard™ is designed for rapid prototyping of complex digital systems. This advanced technology embeds a smart active substrate that can be configured through a set of JTAG ports to interconnect digital integrated chips placed on its surface. The software that generates JTAG bit streams and the related hardware infrastructure embedded in the WaferBoard™ must be validated early and efficiently in the design process to improve productivity. The system co-verification methodology presented in this paper relies on a unified SystemC-VHDL environment making the design and verification tasks more effective. Results demonstrate that the simulation time complexity grows quadratically with the circuit size in number of gates if one JTAG port is used.**

Index Terms— **Co-verification, SystemC, reconfigurable circuit board, Wafer Scale Integration, JTAG, IEEE 1149.1.**

I. INTRODUCTION

To meet the requirements of the PCB/multichip module industry, designers must constantly innovate to offer increasing interconnection density and faster interconnect data rate. Moreover, MCM/PCB design time (time-to-market) must constantly decrease in order to stay competitive. This creates a requirement for innovative technologies that increase productivity and performance of products. The need to improve design productivity justifies exploring new ideas and concepts such as the reconfigurable circuit board technology proposed in [1,2], depicted in figure 1. This technology is based on a wafer scale circuit, the WaferIC™.

The WaferIC™ sits on a WaferBoard™. It is used as an active substrate that can transmit digital information between any pins of a set of conventional chips placed by the user (uICs) anywhere on its surface. Moreover, this circuit embeds an array of tiny programmable pads (NanoPads) that can make electrical contact with uIC balls through which required signal connections can be established and power can be supplied to the uICs. The uICs can be CPUs, FPGAs, memories or any other available integrated circuit whose pinout is compatible with the NanoPad array, which is true of most existing ICs. Each Unit Cell includes a programmable crossbar and 4×4 programmable NanoPads (Fig. 1(c)). The interconnection of cell's crossbars creates a global configurable interconnection network.

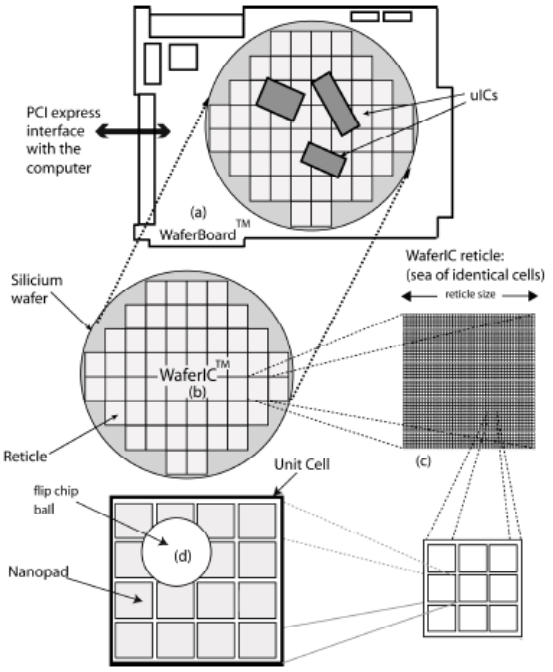


Figure 1: Illustration of the hierarchical structure of WaferBoard™. (a) WaferBoard™ is connected to a computer via a PCI-express connection. (b) WaferIC™, is a sea of identical cells(c) residing on an external pc-board. The uIC/WaferIC™ contact is represented by a ball (d).

This paper presents the hardware and software co-verification strategy used in the design of the WaferIC™. HW/SW co-verification is defined as a methodology to co-simulate hardware and software for functional verification purpose, with two goals: [3]: (1) Debug system software in conjunction with the supporting hardware infrastructure before chips and boards are available. (2) Ensure that the hardware can run the system software. Such co-simulation methodology increases productivity of hardware and software developers and avoids costly hardware design errors. The originality of the work presented in this paper is to adapt and apply a co-verification methodology in a unified SystemC-VHDL verification environment for WaferIC™ design.

Four components are integrated in the proposed SystemC-VHDL environment: (1) the interface software generating JTAG bit streams [4] for configuration, test and diagnosis; (2) the mechanisms to validate the algorithms implemented for testing and diagnosing; (3) algorithms for on-the-fly WaferIC™ reconfiguration; (4) an environment to accomplish thorough verification of the RTL features.

Previous works have already demonstrated the feasibility and the efficiency of co-verification methods for various classes of VLSI systems. For example, co-verification of complex SoC circuits that run software [5], or telecom ASIC chips [6] have already been reported, but no prior works focus on mechanisms to validate the JTAG interface for the test, diagnosis and on-the-fly reconfiguration of a wafer-scale reconfigurable board. Those specific features bring a set of particular constraints for which practical solutions are reported in this paper.

The next section summarizes previously reported results on Waferboard™ and WaferIC™. Section III explains the requirements and the internal architecture that a co-verification environment for the WaferIC™ must meet. Section IV details the internal architecture of the SystemC-VHDL environment and section V shows some results from the SystemC-VHDL simulations. The main contributions of this paper are summarized in Section VI.

II. Related Work on the Active Reconfigurable Board

WaferIC™ is to PCBs what FPGAs are to logic circuits. Software rapidly configures the WaferIC™ to interconnect several uICs according to user's instructions. In order to support any type of interconnection between uICs, it is necessary to have a reconfigurable network able to support a high density of interconnections between every configurable NanoPads. An example of reconfigurable routing between 2 distinct uICs is shown in figure 2. This is done using a network called WaferNet™ described in reference [2]. A study of itsinterconnection network presented in [7] focuses on low level characteristics like delays and silicon area. The short-circuit detection or capacitive detection [8] techniques can be used to detect uIC ball contact on NanoPads.

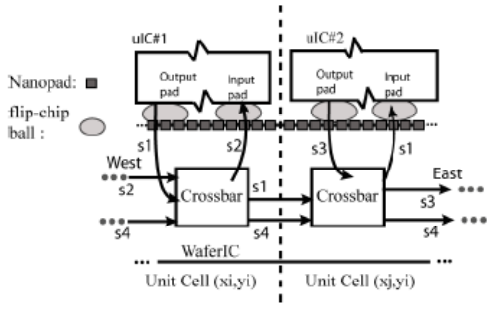


Figure 2: Routing example of a signal s1 from a NanoPad in Unit Cell(xi,yi) redirected to another NanoPad of Unit Cell(xj,yi) towards the east. Signal s4 passes through Unit Cells without interaction with the local uIC.

III. Requirements of the Hardware/Software Co-Verification Environment

The objective of our hardware/software co-verification environment is to make tractable the complex task of verifying the WaferBoard™ system. The main functionalities of the WaferIC™ can be classified in one of the three following successive steps:

A. *Step 1: Test and diagnosis*

The internal test and the diagnosis of WaferIC™ have three goals. First, it is important to make the circuit testable to detect manufacturing defects. Second, diagnostic mechanisms are essential to locate the faults in WaferIC™, and third, to improve robustness and manufacturability, it is essential to configure the system around those faults. Those essential features for wafer scale system must be efficiently verified.

B. *Step 2. Detection procedure for ball/WaferIC™ contacts*

The WaferIC™ can interact with uICs deposited on its surface through the hundreds of thousands NanoPads. Those NanoPads form a regular array spaced with a sufficient resolution to exchange I/O data with the smallest uIC packages. To implement a user specified netlist interconnecting uICs, it is mandatory to know exactly which NanoPads are in contact with a uIC ball and the ones that are floating. Therefore, a specific procedure must be developed and validated to detect electrical contacts between NanoPads and uIC balls.

C. *Step 3. Netlist upload in the WaferIC™*

This step configures the crossbars and NanoPads to create the "netlist" according to user's specifications by sending bitstreams through JTAG interfaces, similar to that available for FPGAs.

All system level functionalities listed above come from the coordinated functionalities contributed by a sea of small identical cells (Fig. 1). These identical cells are logically differentiated after the configuration procedure. To verify and validate the WaferIC™ features, a top level entity that makes connections between Unit Cells is needed. That entity is composed of an array of reticles, and each reticle contains a regular array of cells. It is only through the reticle level or the WaferIC™ level of abstraction that systematic simulation of this regular array of linked cells is possible to verify and validate the complete system that they form. The reticle top level entity can be parametrized with respect to "n", the size in term of Unit Cells in the reticle, and "m" the size of WaferIC™ in term of reticles.

IV. The System Co-Verification Environment

SystemC is a powerful modelling language often used to control and configure top-level VHDL blocks[9,10]. Such SystemC models can then be seen as testbenches that control a "device under test". This approach is somewhat restrictive, because SystemC can be a much more powerful tool to create a unified co-verification environment.

A. *The Software emulation part of the co-verification environment*

Our co-verification strategy is applied by suitably controlling the VHDL model of the WaferIC' I/O with

software routines that emulate the behaviour of a "virtual tester". This "virtual tester" is defined as any software based test and diagnostic procedure applied to the final fabricated WaferIC. Figure 3 illustrate how the software routines are interacting with the RTL model. There are 4 types of I/O in the VHDL model of WaferIC™.

1) Input JTAG port: even if an input JTAG port is 4 bits wide (tdi, tms, trst*, tck), it is not trivial to master the integrity of the serial bit stream. A single fault in a very long bit stream jeopardizes the entire command flow. A high level library written in C++ has been developed to generate the JTAG bitstreams.

2) Output JTAG port (tdo): the serial data must be adequately interpreted by C++ routines that extract the data included in the output "tdo" bit stream. Then, they must be interpreted to check the validity of the JTAG emulator and the hardware structure of the WaferIC™.

3) Data coming from the internal registers: these data allow verifying the validity of commands sent to the WaferIC™ in the form of JTAG bitstreams. It is possible to precisely define the result expected in the internal registers.

4) Data stimulation and recording coming from the NanoPads: the final objective of WaferIC™ is to interconnect the uICs, according to a "netlist" downloaded by a user. The interconnections between the NanoPads can be validated by stimulating and observing the NanoPads behavior.

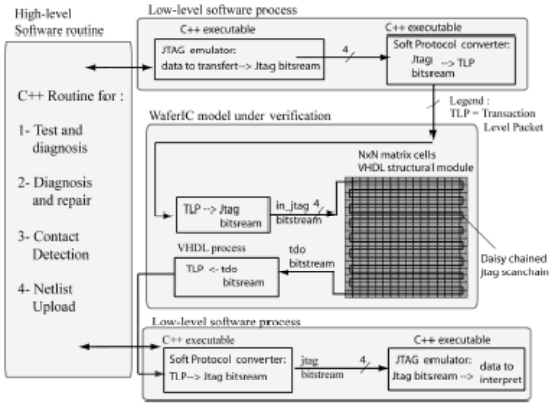


Figure 3: Relationships between the hardware and the software in the WaferIC™ co-verification environment.

Therefore, with the architecture shown in figure 3, developing a testcase for a systematic verification plan is equivalent to directly developing a reusable routine for the WaferIC™ control software. For example, the diagnosis of short-circuit between adjacent NanoPads needs special test algorithms to be validated, which are based on well known short circuit diagnosis algorithms [11]. To validate this kind of feature, one needs to validate the software control and the specific hardware infrastructure. The C++ routines of the control software generate the required bit streams. In addition, the bit streams are applied to the JTAG port through the SystemC-VHDL interface. The auto-verification is completed through software interpretation of the tdo bit streams. The former software process is re-used in the final control software.

B. Internal Architecture Model in SystemC-VHDL

Figure 4 shows how the various modules of the system co-verification environment interact. A multi-threaded graph (MTG) has been used to describe the environment. MTG is a well known representation for SystemC internal logic dependencies, structures and time dependent interactions [12]. Figure 4 shows the relationships between SystemC processes that emulate the behavior of a "virtual tester" interacting with the WaferIC™.

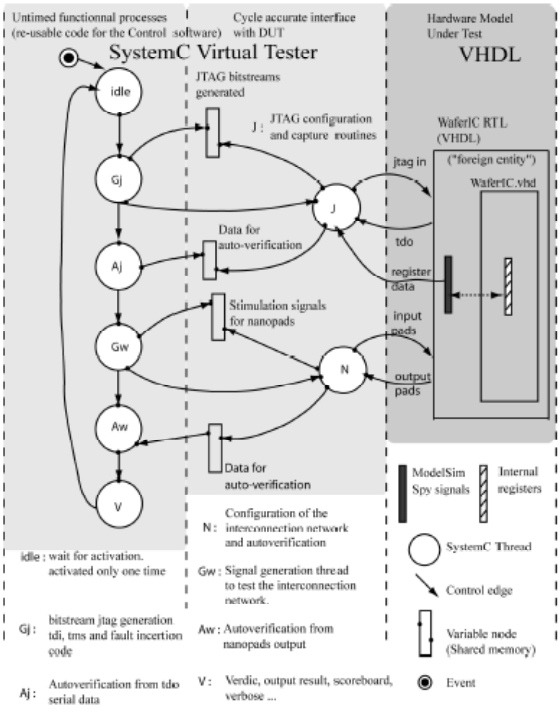


Figure 4: Multi-threaded graph flow for the SystemC-VHDL environment interacting as a virtual tester for the device under test.

The SystemC-VHDL co-verification environment is divided in three specific parts:

1) The untimed functional processes: Those processes are associated to the high level software routines called by the co-verification environment. It is not linked to any cycle accurate event, only through high-level logic conditions required by the control of the "virtual tester" that respect the test procedure of the WaferIC™. The same procedures and logic transaction must take place in the future WaferIC™ control software that will be implemented. Thus the routines implemented in the verification environment are truly reusable for the control software.

2) Cycle accurate processes: Meanwhile, the "J" and "N" processes (see Figure 4) must deal with the low level time synchronisation between cycles and signals. Those processes

are not reusable. They are placed in the systemC architecture as an interface with the device under verification.

3) RTL descriptions of the WaferIC™ : Modelsim was leveraged for concurrent SystemC/VHDL simulations. The use of "spy" variables provided by Modelsim made it possible the recopy data from inside to outside modules [13]. Thus, the proposed architecture offers as much data observability for hardware verification, as provided in "e" or System Verilog [14].

V. RESULTS

Our verification environment can simulate all the functional features of the WaferIC™. For demonstration purposes, only a small subset of the entire regression test is shown in this paper. Figure 5 presents the simulation results corresponding to a test case taken from the regression script of the verification plan. This test case uploads JTAG bit streams in each Unit-cell of the WaferIC™ in order to configure a specific interconnection net list. All cells are daisy chained together (see fig. 3) as is required by the IEEE 1149.1 communication protocol. Our test cases and RTL descriptions are generic according to the number of Unit Cells in one reticle, n^2, thus it is easy to change the value of n and observe the effect on memory consumption and simulation time.

As expected, memory used to co-simulate the test cases grows quadratically with n, the size of the Unit Cell array. Meanwhile, the simulation time complexity of one test case grows as $O(n^4)$ as shown in figure 5. Effectively, time complexity depends on both the length of the bit stream and the number of Unit Cells, which each grows as n^2. A solution to dramatically reduce the simulation time consists of increasing the number of JTAG ports with the number of Unit Cell, in which case the time complexity could remain $O(n^2)$.

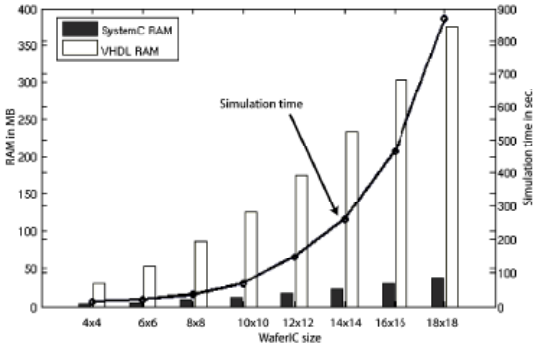


Figure 5: Simulation time and RAM used vs the size of the Reticle for one generic test case.

In spite of such time complexity, our results shows that a typical high-end desktop computer and commercially available software can handle the simulation task. Indeed, the simulations have been carried out on a Linux machine, with ModelSim SE 6.2c. The graph on figure 5shows that the simulation time and the memory used by the selected typical test case in the verification environment is tractable for a Intel DualCore running at 2.4 GHz with 3.289 GB RAM desktop computer. Therefore, the reported results demonstrate that our verification environment can handle the required verification tasks to simulate all the features of the WaferIC™.

VI. CONCLUSIONS

An innovative and promising technology for fast system prototyping of digital systems that uses Wafer-Scale Integration has been presented and validated with a model coded in SystemC-VHDL and the ModelSim simulator. Moreover, the Unit-Cell that is the cornerstone of the proposed regular architecture and its associated configurable interconnection network have been simulated, validated and verified with an unified co-verification approach. The successful simulation reported in this paper demonstrates that the level of complexity of the proposed hardware/software co-verification environment is manageable for a small team of designers.

ACKNOWLEDGMENTS

The authors would like to thank the Natural Sciences and Engineering Research Council of Canada and Gestion TechnoCap Inc. for their financial support and CMC Microsystems for providing design tools, support and associated technologies.

REFERENCES

[1] R. Norman, O. Valorge, Y. Blaquiere, E. Lepercq, Y. Basile-Bellavance, Y. El-Alaoui, R. Prytula, Y. Savaria, "*An Active Reconfigurable Circuit Board*", NEWCAS 2008, Montreal, Canada, June 2008.

[2] R. Norman, U.S. Patent Application Number 11/611,263.

[3] A. Ghosh, M. Bershteyn, R. Casley, C. Chien, A. Jain, M. Lipsie, D. Tarrodaychik, O. Yamamoto, "*A hardware-software co-simulator for embedded system design and debugging*", Asia and South Pacific Design Automation Conference, Chiba, Japan, Sept 1995.

[4] P. K. Parker, "*The boundary-scan handbook. analog and digital*", 2nd ed., Kluwer Academic Publishers Boston/ Dordrecht/ London, 1998, ch. 1,4.

[5] Y. Nakamura, "*Software Verification for System on a Chip using a C/C++ Simulator and FPGA Emulator*", International Symposium on VLSI Design, Automation and Test, April 2006.

[6] T. Wu, P. Wang, D. Jin, L. Zeng, "*Hardware/software co-verification scheme for MSTP ASIC*", ICSICT '06. 8th International Conference on Solid-State and Integrated Circuit Technology, 2006.

[7] E. Lepercq, Y. Savaria., Y. Blaquière, , "*An Interconnection Network for A Novel Reconfigurable Circuit Board*", NEWCAS, Montreal, Canada, June 2008.

[8] Y. Blaquière, Y. Savaria, J. El Fouladi, "*Digital Measurement Technique for Capacitance Variation Detection on Integrated Circuit I/Os*", *ICECS'07*, December 11-14, 2007.

[9] *SystemC Version 2 User's Guide*, http://www.systemc.org

[10] *Functional Specification for SystemC 2.0.1*- Version 2.0-P, http://www.systemc.org

[11] A. Hassan, J. Rajski, V.K. Agarwal, "*Testing and Diagnosis of Interconnects using Boundary Scan Architecture*", Internationnal Test Conference, 1988

[12] F. Thoent, F. Catthoor, "*Modeling, Verification and Exploration of Task-Level Concurrency in Real-Time Embedded Systems*". Kluwer Academic Publishers. 438p., 2000.

[13] Mentor Graphics, "*ModelSim SE User's Manual*", August 2006, ch. 21.

[14] J. Bergeron,"*Writing Testbenches: Functional Verification of HDL Models*", Springer, 2nd edition, 2003, ch 4,5.

ANNEXE F – TEST DE RÉGRESSION

Cas de test au niveau WR NANOPADS:

Cas de test	Génération de stimulis	Auto-vérification	Intégré dans la régression	Éxécuté?	Découvertes d'anomalies	Pointage	#test
WRNA_CONFIG_IOBUF_FPGA	Done	Done	Yes	Yes	7	5	12
WRNA_CONFIG_IBUF_ASIC	Done	Done	Yes	Yes	4	5	32
WRNA_CONFIG_OBUF_ASIC	Done	Done	Yes	Yes	2	5	32
WRNA_CONFIG_IOBUF_ASIC	Done	Done	Yes	No	2	5	64
						20	140
					Couverture	100.0	

Testcase at CELL_TOP level :

Cas de test	Génération de stimulis	Auto-vérification	Intégré dans la régression	Éxécuté?	Découvertes d'anomalies	Pointage	#test
JT_CTOP_CONF_NA_H	Done	Done	Yes	Yes	6	5	1
JT_CTOP_CONF_NA_L	Done	Done	Yes	Yes	6	5	1
JT_CTOP_CONF_WFNET_H	Done	Done	Yes	Yes	10	5	1
JT_CTOP_CONF_WFNET_L	Done	Done	Yes	Yes	3	5	1
JT_CTOP_CONF_CD_H	Done	Done	Yes	Yes	1	5	1
JT_CTOP_CONF_CD_L	Done	Done	Yes	Yes	2	5	1
JT_CTOP_CONF_BS_OUT_ASIC	Done	Done	Yes	Yes	3	5	2
JT_CTOP_CONF_BS_IN_ASIC	Done	Done	Yes	Yes	2	5	3
JT_CTOP_OUT_IBUF_READ	Done	Done	Yes	Yes	1	5	3
CTOP_TPT_READ_NANOPAD	Done	Done	Yes	Yes	4	5	3
JT_CTOP_CONF_TC_LH	Done	Done	Yes	No	4	5	1
JT_CTOP_CONF_TC_H	Done	Done	Yes	No	4	5	1
JT_CTOP_CONF_TC_L	Done	Done	Yes	No	4	5	1
JT_CTOP_CONF_CR_H	Done	Done	Yes	Yes	1	5	1
JT_CTOP_CONF_CR_L	Done	Done	Yes	Yes	0	5	1
JT_CTOP_FUSE_MODE	Done	Done	Yes	Yes	3	5	1
						80	23
					Couverture :	100.0	

Testcase at RETICLE level :

Cas de test	Génération de stimulis	Auto-vérification	Intégré dans la régression	Éxécuté?	Découvertes d'anomalies	Pointage
JT_ALLPATH3X3_TC	Done	Done	Yes	Yes	6	5
JT_ALLBORDER3X3_TC	Done	Done	Yes	Yes	3	5
JT_EXTCTRL_3X3_NA_H	Done	Done	Yes	Yes	10	5
JT_EXTCTRL_3X3_CR_H	Done	Done	Yes	Yes	12	5
JT_EXTCTRL_3X3_CD_H	Done	Done	Yes	Yes	1	5
JT_EXTCTRL_3X3_TC_H	Done	Done	Yes	Yes	1	5
JT_CONF3X3_NA_H	Done	Done	Yes	Yes	8	5
JT_CONF3X3_CD_H	Done	Done	Yes	Yes	0	5
JT_CONF3X3_CR_H	Done	Done	Yes	Yes	0	5
JT_CONF3X3_TC_H	Done	Done	Yes	Yes	0	5
JT_CONFIG_WFNET_2NX2N_H	Done	Done	Yes	Yes	3	5
JT_CONFIG_WFNET_2NX2N_L	Done	Done	Yes	Yes	1	5
JT_ALLPATH_2NX2N	Done	Done	No	No	0	5
JT_EXTCTRL_2NX2N_NA	Done	Done	No	No	0	5
JT_EXTCTRL_2NX2N_CR	Done	Done	No	No	0	5
JT_EXTCTRL_2NX2N_CD	Done	Done	No	No	0	5
JT_EXTCTRL_2NX2N_TC	Done	Done	No	No	0	5
JT_BYP_CONF_2NX2N_NA	Done	Done	No	No	0	5
JT_BYP_CONF_2NX2N_CR	Done	Done	No	No	0	5
JT_BYP_CONF_2NX2N_CD	Done	Done	No	No	0	5
JT_BYP_CONF_2NX2N_TC	Done	Done	No	No	0	5
JT_TESTAB_2NX2N_NA	Done	Done	No	No	0	5
JT_TESTAB_2NX2N_CD	Done	Done	No	No	0	5
JT_TESTAB_2NX2N_CR	Done	Done	No	No	0	5
JT_TESTAB_2NX2N_TC	Done	Done	No	No	0	5
JT_CONFIG_WFNET_2NX2N_H	Done	Done	Yes	Yes	3	5
JT_CONFIG_WFNET_2NX2N_L	Done	Done	Yes	Yes	1	5
JT_CONF_2NX2N_NA_H	Done	Done	No	Yes	1	5
JT_CONF_2NX2N_TC_H	Done	Done	No	Yes	1	5
JT_CONF_2NX2N_CD_H	Done	Done	No	Yes	1	5
JT_CONF_2NX2N_CR_H	Done	Done	No	Yes	1	5

143

Couverture: 100.0

Testcase at WAFERIC level :

Cas de test	Génération de stimulis	Auto-vérification	Intégré dans la régression	Exécuté?	Découvertes d'anomalies	Pointage	#test
WIC_ALLPATH2NX2N_TC	Done	Done	Yes	Yes	2	5	4
WIC_CONF_WFNET_2NX2N_H	Done	Done	Yes	Yes	3	5	1
WIC_CONF_WFNET_2NX2N_L	Done	Done	Yes	Yes	1	5	1
WIC_EXTCONF_2NX2N_NA	Done	Done	Yes	Yes	5	5	6
WIC_EXTCONF_2NX2N_CR	Done	Done	Yes	Yes	0	5	6
WIC_EXTCONF_2NX2N_CD	Done	Done	Yes	Yes	0	5	6
WIC_EXTCONF_2NX2N_TC	Done	Done	Yes	Yes	0	5	6
						35	30
					Couverture :	100.0	

Nombre total de cas de test : 379

Couverture : 100%

ANNEXE G – RÉSUMÉ DU STANDARD IEEE1149.1

En 1985, un groupe de compagnie ont formé un consortium dans le but de mettre au point un nouveau standard pour le test de PCB basé sur un protocole de communication sérielle entièrement contrôlable par logiciel. L'objectif du standard est d'accéder au contrôle et à l'observation des broches des circuits intégrés d'un circuit imprimé (PCB) dans le but de tester les interconnexions des circuits discrèts sur le PCB. Le terme « boundary scan » (chaîne de balayage) a été ainsi créé. Ce groupe s'est appellé Joint European Test Action Group (JETAG). Quelques années plus tard, le concept s'est généralisé en Amérique du Nord pour prendre le nom de JTAG et a été accepté comme standard IEEE 1149.1. La figure G.1 montre un exemple d'application de la méthodologie JTAG à un circuit composé de 3 puces intégrées.

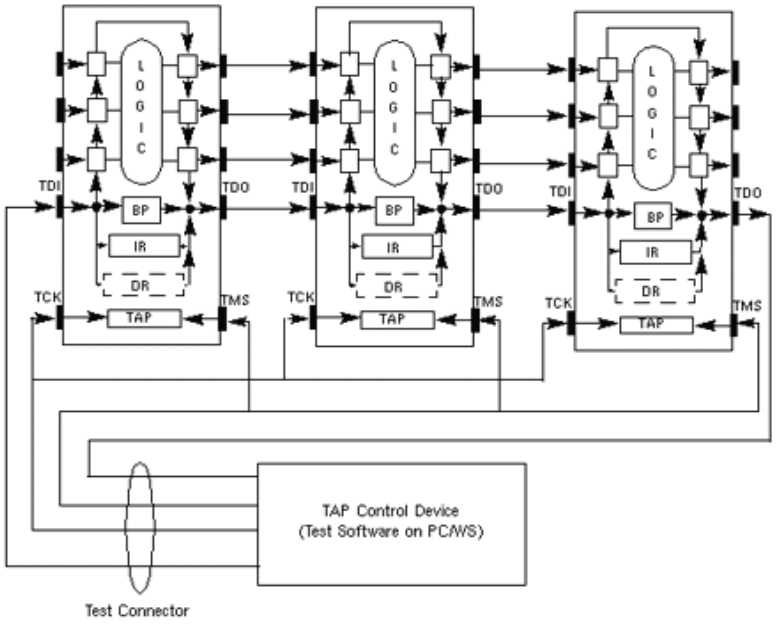


Figure G.1 Exemple d'utilisation du protocole JTAG [10]

Un système de test JTAG possède toujours un certain nombre d'éléments :

1- "Tap Controller" : une machine à état fini dont le diagramme d'état est représenté à la figure G.2.

2- "Bypass register" : un registre simple obligatoire dans le standard utilisé pour raccourcir les chemins sériels entre les circuits intégrés sous test.

3- "Instruction register" : défini le mode dans lequel la chaîne de balayage est utilisée.

4- Le port tdi: signal d'entrée des données sérielles de test à appliquer aux chaînes de balayage

5- Le port tms : signal de contrôle du "Tap Controller" (voir figure G.2).

6- Le port tck : signal d'horloge activé lors du mode test

7- Le port tdo : signal de sortie des données sérielles (résultats du test appliqué)

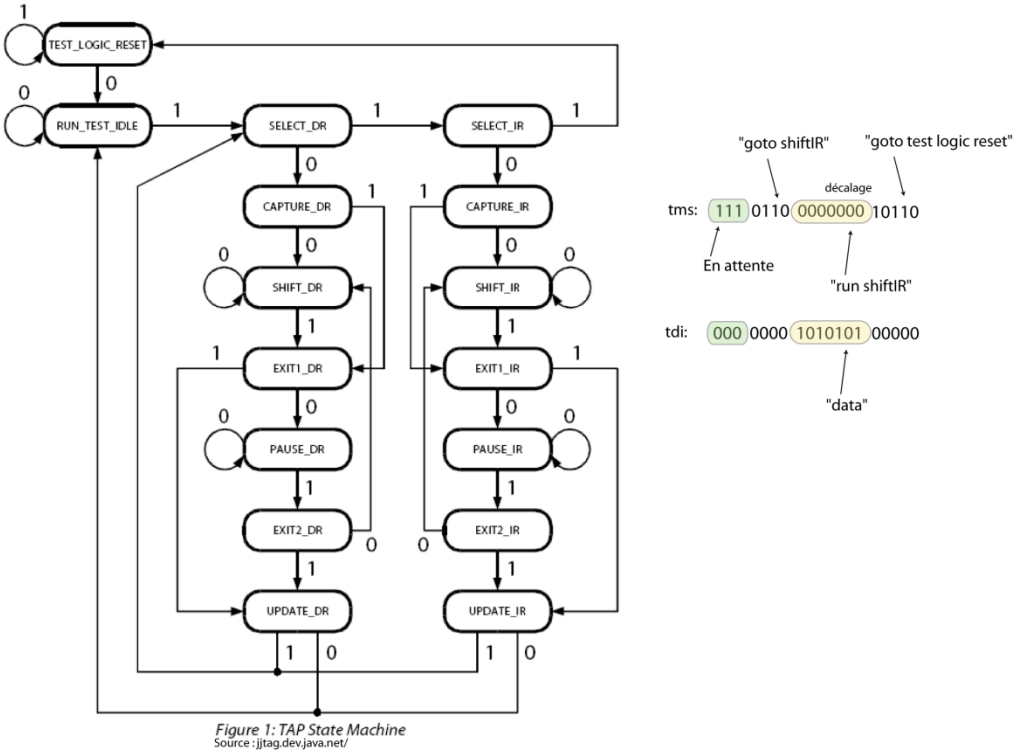


Figure G.2 Machine à état du “Tap Controller”