ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE ÉLECTRIQUE
M. Ing.

PAR
Nik RENQUINHA

AMÉLIORATION DE LA TECHNIQUE DE TEST ET DIAGNOSTIC CDIDDQ

MONTRÉAL, LE 10 JANVIER 2014

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. Claude Thibeault, directeur de mémoire
Département de génie électrique à l'École de technologie supérieure

M. Ghyslain Gagnon, codirecteur de mémoire
Département de génie électrique à l'École de technologie supérieure

M. Nicolas Constantin, président du jury
Département de génie électrique à l'École de technologie supérieure

Mme Catherine Laporte, membre du jury
Département de génie électrique à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 12 DECEMBRE 2013

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

J'aimerais tout d'abord remercier Claude Thibeault, mon directeur de maîtrise, pour sa patience exceptionnelle et sa détermination à avoir bien voulu me ramener sur le bon chemin tout au long de ma maîtrise. Il a été d'un très grand support moral et m'a toujours poussé à vouloir me dépasser et perfectionner mes compétences actuelles. Je suis, en grande partie, grâce à lui, un professionnel qui a toujours envie d'en savoir plus sur les sujets qui me passionnent. Son soutien m'a été d'une aide inestimable. Je le remercie aussi de ne pas avoir perdu confiance en la rédaction de ce mémoire malgré mon incapacité à organiser et formuler mes idées correctement. Il a toujours cru en mes idées et je lui dois une grande reconnaissance. Merci également à mon co-directeur, Ghyslain Gagnon, pour ses commentaires et suggestions.

J'aimerais aussi remercier ma copine Lysanne Arseneault-Fontaine pour l'appui moral, les encouragements et, bien sûr, la correction des fautes d'orthographe et de syntaxe de ce mémoire. Elle a grandement contribué à la bonne compréhension de cet écrit et elle a su passer un grand nombre d'heures à s'assurer que ce mémoire soit impeccable et représentatif de mes recherches.

Finalement, merci à Christelle Hobeika pour avoir rendu mes journées de rédaction et de recherche plus agréables tout en restant dans un environnement professionnel. Elle m'a donné plusieurs sources et idées qui m'ont été essentielles à la rédaction de ce document et elle a su me motiver tout au long de mes travaux de recherche. Son aide et son appui ont su me donner envie de me surpasser et d'en découvrir davantage dans ce domaine.

AMÉLIORATION DE LA TECHNIQUE DE TEST ET DIAGNOSTIC CDIDDQ

Nik RENQUINHA

RÉSUMÉ

Ce mémoire a pour objectif l'amélioration de la technique de test et diagnostic CDIDDQ. Le travail réalisé consiste au perfectionnement de l'application du test CDIDDQ sur des circuits intégrés de type FPGA précédemment développée par Haithem (2011) afin d'accroître la capacité à détecter des défectuosités de type court-circuit. Ce mémoire débute par une présentation des notions de base ainsi qu'une revue de littérature où sont présentées différentes notions liées aux pannes de circuits intégrés, l'émulation de pannes sur FPGA, les techniques de tests basées sur le courant ainsi que les travaux passés dans l'application de la technique de test CDIDDQ.

Par la suite, la technique de génération des patrons de test CDIDDQ ainsi que les vecteurs associés à ceux-ci sont présentés. Nous avons utilisé la simulation afin de valider le fonctionnement des patrons de test pour ensuite passer à leur application sur des circuits intégrés de type FPGA. Comme première application expérimentale, nous avons choisi de refaire celle employée par Haithem (2011) en utilisant toutefois un différent système de mesure, ce qui nous a donné des résultats similaires.

Suite à cela, nous avons apporté des modifications sur le montage afin de réduire au maximum le bruit sur l'alimentation. Nous avons utilisé le même FPGA, mais cette fois-ci, sur une autre carte de développement, ce qui nous a permis d'augmenter notre capacité à détecter des pannes de court-circuit. Nous nous sommes aussi intéressés à l'application de la technique sur des circuits plus énergivores allant de 70mA à 3A.

Pour finir, nous présentons en détail la technique d'application expérimentale avec laquelle nous avons pu obtenir les meilleurs résultats. À ce stade, nous utilisons un différent montage, une différente architecture pour le testeur CDIDDQ, une différente technique d'acquisition et un logiciel spécialement mis au point pour les tests CDIDDQ. Ces modifications nous ont, entre autres, permis de détecter des pannes de court-circuit induisant des courants cinq fois plus faibles.

Mots-clés: FPGA, CDIDDQ, court-circuit, test

AMÉLIORATION DE LA TECHNIQUE DE TEST ET DIAGNOSTIC CDIDDQ

Nik RENQUINHA

ABSTRACT

This thesis aims to improve the CDIDDQ test and diagnostic technique. The experiments consist in improving the CDIDDQ application, previously developed by Haithem, to detect short-circuit type defect in a FPGA circuit. This thesis begins with an introduction to basic concepts as well as a literature review describing integrated circuit fault concepts, FPGA fault emulation, current-based test techniques, and previous work on the CDIDDQ test technique applications.

Subsequently, the CDIDDQ test pattern generation technique as well as its associated vectors is presented. Simulation was used to validate test patterns in order to apply them on FPGA integrated circuits. We redid Haithem's (2011) experiment using a different measurement system, which produced almost identical results.

Afterwards, modifications were done on the setup in order to reduce power induced noise as much as possible. The same FPGA was used but on a different prototype board, which increased the ability to detect short circuit defects. We also looked into the application of the technique on energy-hungry circuits from 70mA to 3A.

Finally, the applied experimentation techniques with which we obtained the best results is laid out in detail. At that point, a different setup was used, as well as a different architecture for the CDIDDQ tester, a different acquisition technique and a software custom made for CDIDDQ tests. These modifications allowed us to detect bridge faults with a current five times smaller.

Keywords: FPGA, CDIDDQ, short-circuit, testing

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ASIC	Application-specific integrated circuit
ATPG	Automatic test pattern generation
CMOS	Complementary Metal-Oxide-Semiconductor
CUT	Circuit under test
CDIDDQ	Complementary delta direct drain quiescent current
DC	Direct current
DIDDQ	Delta direct drain quiescent current
DFT	Design for testability
ÉTS	École de technologie supérieure
FPGA	Field-programmable gate array
IDDQ	Direct drain quiescent current
IO	Input/output
JTAG	Joint test action group
PC	Personal computer
PROM	Programmable read-only memory
RTL	Register to transistor level
TDF	Transition delay fault
USB	Universal serial bus
VBA	Visual Basic for applications
VHDL	Very high speed integrated circuit hardware description language

INTRODUCTION

Au cours des dernières décennies, le développement de la microélectronique s'est fait à un rythme exponentiel soutenu. Ce rythme de développement suit la fameuse « loi de Moore », selon laquelle les circuits intégrés voient leur densité d'intégration doubler tous les 18 mois (Moore 1965). La miniaturisation des transistors qui lui est associée a permis de développer des systèmes de plus en plus petits et performants. Cependant, cette miniaturisation vient également avec son lot de défis à relever, défis qui évoluent avec la technologie et les procédés de fabrication de plus en plus complexes. Par exemple, ces procédés permettent l'intégration d'un nombre croissant de niveaux de métal, avec des traces de plus en plus rapprochées. Ceci a entre autres mené à l'augmentation des défectuosités dans la fabrication des circuits, particulièrement au niveau des courts-circuits entre les traces de métal (Zorian et Mourad 2000). Ainsi, même avec une conception qui respecte toutes les règles de design du manufacturier, il est possible d'avoir des circuits intégrés qui soient défectueux. Dans ce contexte, il est important que les méthodes de test utilisées évoluent avec la technologie et les défectuosités introduites par les procédés de fabrication.

L'objectif du test est la détection de manière efficace de tout problème pouvant affecter un circuit intégré. Cela permet de garantir que le système est fonctionnel et répond aux spécifications. C'est une étape importante, car elle permet d'assurer la qualité du produit. La complexité grandissante des circuits intégrés a rendu cette tâche plus ardue. Face à cette complexité grandissante, on a pu voir l'apparition de différentes techniques de test.

Parmi ces techniques on retrouve le test dit logique, qui constitue la technique la plus utilisée. Sur la base de la représentation au niveau porte, le test logique vérifie le niveau logique, 0 ou 1 (respectivement représenté par les tensions 0 et Vdd), d'un ou plusieurs nœuds d'un circuit intégré afin de vérifier le bon fonctionnement de celui-ci. Sans apport d'informations supplémentaires, cette technique s'avère peu efficace pour détecter les pannes de court-circuit (Rajsuman 2000). En effet, lorsqu'on utilise comme information seulement le niveau porte, le nombre de courts-circuits potentiels est de l'ordre de N^2 où N représente le nombre

de noeuds dans le circuit (Zorian et Mourad 2000). De plus, afin de procéder au test logique, on doit propager les valeurs des nœuds internes sur les sorties du circuit. Cela vient donc complexifier la génération des vecteurs de test. Aussi, puisque le niveau de tension du court-circuit dépend de la résistivité du court-circuit et des dimensions des transistors, il est difficile de savoir le résultat logique équivalent, qui, dans le pire cas, pourrait être une valeur mitoyenne (Rajsuman 2000). Aussi, notons que même si un court-circuit n'affecte pas la fonctionnalité, il cause la dégradation de la fiabilité du circuit, ce qui est préoccupant dans la majorité des cas (Rajsuman 1991).

Pour pallier les faiblesses du test logique pour la détection des courts-circuits, une autre technique s'est imposée comme alternative. Il s'agit du test basé sur le courant de consommation statique (« IDDQ testing »), présentée plus en détail au chapitre suivant. Le test IDDQ, qui offre de manière inhérente une très grande observabilité, a été développé en tirant profit du fait que les circuits CMOS, en mode statique, ne consommaient à l'époque qu'une minime quantité de courant (Riezeman 1991), et que les défectuosités telles les courts-circuits causaient en général une augmentation significative du courant. Ainsi, un court-circuit devenait facilement détectable. Malheureusement, la diminution de la taille des transistors, ainsi que l'augmentation de la taille des circuits intégrés ont contribué à augmenter le courant de fuite des circuits sans pannes, ce qui a réduit l'efficacité du test IDDQ (Engelke 2006). En effet, l'utilisation de la technique nécessite que le courant de fuite d'un circuit sans panne soit beaucoup plus petit que celui d'un circuit avec panne, ce qui de nos jours n'est plus le cas (Powell 2000).

Afin de prolonger l'utilisation du courant de consommation statique durant le test, un certain nombre d'approches ont été proposées, la plupart basées sur le post-traitement des mesures de courant. Une des approches très utilisées est Delta-IDDQ (DIDDQ) (Thibeault 1999), qui sera également décrite de manière un peu plus détaillée au chapitre suivant. Cette approche fait la différence entre deux mesures consécutives de courant statique. Cela permet d'éliminer le courant d'arrière-plan (plus précisément sa valeur moyenne), qui varie d'une puce à l'autre, dû aux variations inhérentes des procédés de fabrication, et donc de réduire

l'effet néfaste de grands courants de fuite. Comme son prédécesseur, DIDDQ voit néanmoins son efficacité diminuer avec la réduction de la taille des transistors, en raison des variations qui apparaissent d'une mesure de courant à l'autre.

Une nouvelle approche a été récemment proposée afin d'éliminer l'effet des variations affectant DIDDQ. Cette approche, appelée test CDIDDQ (Thibeault et Hariri 2009) et décrite en détail au chapitre 1, s'appuie sur une nouvelle façon de générer les vecteurs de test et d'effectuer le post-traitement des mesures de courant, qui permet d'éliminer l'effet de toutes les variations (d'une puce à l'autre, d'une mesure à l'autre), sauf celles liées au bruit de la mesure du courant. Cette élimination facilite la détection de courts-circuits, pourvu que le niveau de courant supplémentaire soit suffisant pour être distingué du bruit de mesure.

La validation expérimentale de CDIDDQ sur des circuits ASIC émulés sur FPGA (Haithem 2011) a démontré l'efficacité de cette technique de test. L'application a été faite sur deux approches distinctes : le test indépendant de l'application et celui dépendant de l'application. Les tests permettent de détecter des courts-circuits dans le FPGA pour l'approche indépendante ou dans l'ASIC émulé dans le FPGA pour l'approche dépendante. Les résultats expérimentaux ont démontré qu'il était possible de détecter des pannes de courts-circuits d'une résistivité d'environ 80k ohms et moins sur des circuits émulés sur le FPGA à faible consommation de courant.

Ce projet de maîtrise vise à obtenir de meilleurs résultats dans la détection de pannes de court-circuit pour les ASIC, émulés sur FPGA. La principale contribution de ce mémoire est l'amélioration de la technique de test et diagnostic CDIDDQ précédemment utilisée, grâce 1) à l'utilisation d'une meilleure méthodologie d'application du test CDIDDQ, 2) au développement de logiciels d'acquisition, et 3) à l'amélioration du montage. Ces améliorations permettent d'obtenir des résultats validant la possibilité de détecter des pannes de court-circuit à haute résistivité sur des circuits intégrés émulés sur FPGA, et ce, en présence d'une grande consommation de courant. Le mémoire est structuré comme suit :

1- Le chapitre 1 présente les notions de base nécessaires à la compréhension de ce mémoire ainsi que l'état de l'art dans le domaine du test des circuits. De plus, nous expliquons tout ce qui traite du processus de génération des vecteurs CDIDDQ.

2- Dans le chapitre 2, nous expliquons en premier lieu la génération des patrons de test CDIDDQ qui saura par la suite les valider en simulation. En second lieu, nous expliquons le fonctionnement de l'application expérimentale de CDIDDQ sur des circuits FPGA. Enfin, nous procéderons à la reproduction des résultats déjà obtenus (Haithem 2011) afin de revalider l'application de la méthode CDIDDQ.

3- Dans le chapitre 3, nous appliquons les patrons de test CDIDDQ sur une autre carte de développement comportant le même FPGA. Nous présentons en détail le nouveau montage et explorons les différentes possibilités d'alimentation de la carte.

4- Le chapitre 4 s'intéresse à l'application des patrons de test CDIDDQ sur des montages plus énergivores. On y présente le montage ainsi que les techniques qui ont été utilisés afin d'obtenir les résultats présentés.

5- Dans le chapitre 5, nous présentons les différentes améliorations appliquées pour la méthode CDIDDQ ainsi que les résultats expérimentaux obtenus grâce à ceux-ci. Nous terminerons avec un tableau récapitulatif affichant les résultats de différents patrons de test CDIDDQ pour différents montages.

CHAPITRE 1

NOTIONS DE BASE ET REVUE DE LA LITTÉRATURE

1.1 Introduction

Dans ce chapitre, nous allons dans un premier temps présenter les notions de base nécessaires à la compréhension de ce mémoire. Les notions couvertes seront les tests, IDDQ, Delta-IDDQ, CDIDDQ et l'émulation d'ASIC sur FPGA. Par la suite, nous allons procéder à la revue de la littérature qui nous positionnera face aux autres recherches existantes et justifiera notre contribution au domaine de la détection de courts-circuits dans les ASICs de type CMOS. Nous nous attarderons à la littérature traitant de la mesure de courant et l'utilisation de deux sources. Pour finir, nous présenterons les résultats obtenus avec le montage d'Haithem (Haithem 2011).

1.2 Notions de base du test

Bien que la simulation d'un circuit intégré indique que ce dernier est fonctionnel, il est possible qu'il n'effectue pas la fonction désirée une fois implémenté. En effet, il est possible, par exemple, qu'il y ait des défectuosités introduites par le procédé de fabrication qui engendrent des dysfonctionnements. Dans le but de détecter les défectuosités et d'assurer la fiabilité d'un circuit intégré, il est impératif de le tester (Breuer 1976). Le test est défini comme étant la série d'actions entreprises après la fabrication d'un circuit intégré menant à la détection de tout problème, comparativement au diagnostic qui permet de connaître la source du problème. Pour tester le circuit, nous devons générer des vecteurs de test. Les vecteurs comprennent la valeur à injecter à l'entrée ainsi que leurs sorties correspondantes. Avec la complexité des systèmes courants, il est nécessaire de faire appel à des logiciels de génération automatisée de vecteur (ATPG). Afin de simplifier le processus de génération, les vecteurs sont produits pour des modèles abstraits de panne. Selon (Mourad, S. and Y. Zorian 2000) il y a quatre principaux modèles de pannes : la panne collé-à, la panne ouverte, la panne de délais et la panne de court-circuit.

La panne collé-à (stuck-at), qui est le modèle le plus utilisé dans le test de circuit (Bushnell, M. L. et V. D. Agrawal 2000), représente un nœud collé à une valeur. La valeur du nœud comportant la défectuosité est soit toujours à 1 (VDD) ou soit toujours à 0 (GND), ce qui vient de ce fait altérer le fonctionnement du circuit.

La panne ouverte (stuck open) représente une panne où un nœud est laissé non connecté au reste du circuit. Nous avons donc un nœud qui est flottant, ce qui va aussi changer le fonctionnement de notre circuit.

Une panne de délais représente une défectuosité temporelle affectant des signaux qui, dès lors, se propagent trop lentement ou rapidement dans le circuit. L'objectif de tout test ciblant ce genre de défectuosité est de détecter les délais qui divergent de ceux nécessaires à la bonne synchronisation du circuit. Ces pannes sont donc testées à des fréquences élevées (i.e. à des fréquences supérieures ou égales à la fréquence nominale) afin de permettre de les détecter. Il existe deux modèles de pannes de délais : les délais de transitions et les délais des chemins (Smith 1985). Le délai de transition est une transition lente à monter (0 à 1) ou lente à descendre (1 à 0), appliqué au niveau d'une porte logique de manière ponctuelle sur un nœud, tandis que le délai de chemin est appliqué de façon distribuée sur un chemin complet.
La panne court-circuit (bridge faults) représente une défectuosité ou une connexion physique non désirée entre deux nœuds comme représenté dans la figure 1.1. La miniaturisation des circuits a causé l'augmentation des pannes de court-circuit étant donné la réduction de l'espace entre les traces (Mourad, S. and Y. Zorian 2000). Le lien résistif établi entre les deux signaux peut modifier le niveau des sorties des portes logiques associées. Le niveau logique est difficile à prédire puisqu'il dépend alors de la dimension des transistors, ce qui rend plus ardue la détection de ce type de faute (Kuen-Jong Lee 1991). De plus, les courts-circuits hautement résistifs sont davantage difficiles à détecter par le test logique puisque ceux-ci n'influencent pas la valeur logique. Notons que, comme mentionné dans la problématique, nous nous concentrerons dans ce mémoire sur la détection des pannes de court-circuit. L'objectif est de détecter tous les courts-circuits, causant ou non une modification du fonctionnement du système.

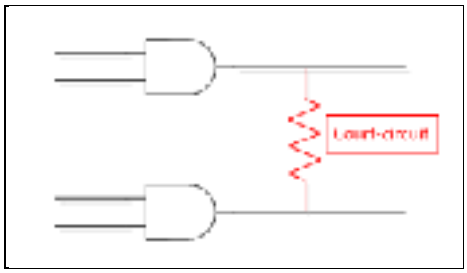

Figure 1.1 Court-circuit

1.3 IDDQ

La technique de test IDDQ permet, grâce à la lecture du courant statique, de détecter une anomalie causant une augmentation de ce courant dans un circuit intégré. Cette technique a été développée en s'appuyant sur le fait que les circuits CMOS en statique ne consommaient qu'un petit courant de fuite provenant des transistors (Riezeman 1991). Ainsi, une surconsommation de courant provenant du circuit indiquait la présence d'une défectuosité. IDDQ permet de détecter certaines défectuosités manquées par les autres techniques de test, par exemple un faible (i.e très résistif) court-circuit ne causant qu'un petit délai supplémentaire, insuffisant pour mener à une erreur de synchronisation. Un autre avantage du test par le courant est que nous ne nous soucions pas de l'observabilité (Shinogi, T. 1998). L'observabilité est un paramètre qui définit la facilité à pouvoir lire la valeur d'un nœud dans un circuit. Puisque nous lisons directement le courant fourni via le bloc d'alimentation au circuit, nous n'avons pas à nous soucier de ce paramètre. Cela est un grand avantage pour la détection des pannes de court-circuit. Ceci a contribué au départ à la popularité d'IDDQ pour la détection de ce type de panne.

1.3.1 DIDDQ

La diminution de la taille des transistors a réduit l'efficacité d'IDDQ (Engelke 2006). En effet, la technique implique que le courant de fuite d'un circuit intégré sans panne est beaucoup plus petit que celui d'un circuit avec panne. Malheureusement, la diminution de la taille des transistors ainsi que l'augmentation de la taille (et densité) des circuits intégrés ont

contribué à augmenter le courant de fuite des circuits sans panne. IDDQ s'avère donc plus sensible face au bruit et aux variations des procédés de fabrication qui se traduisent par une variance des mesures d'une puce à l'autre et d'un vecteur de test à l'autre. Grâce à des analyses expérimentales (Thibeault 1998), on a pu démontrer qu'en soustrayant la mesure de courant de deux vecteurs IDDQ consécutifs pour une même puce, on arrive à diminuer la variance des distributions de courant et à réduire le décalage du courant de fuite des transistors. Par exemple, si la première mesure donne 100mA et la deuxième 110mA et que l'on place un seuil à 5mA, on considérera la puce comme défectueuse. Cette technique de post-traitement des mesures de courant statique, appelée Delta IDDQ (DIDDQ), permet en effet de réduire l'effet des variations de courant d'une puce à l'autre, en éliminant le courant statique moyen consommé par chaque circuit intégré.

1.3.2 CDIDDQ

Comme son prédécesseur, DIDDQ voit son efficacité diminuer avec la réduction de la taille des transistors. Cette diminution est liée aux variations grandissantes des mesures de courant observées d'un vecteur de test à l'autre. Afin d'étendre la vie de l'utilisation du test par consommation de courant Thibeault et Hariri (2009) ont mis au point une nouvelle technique de test appelée CDIDDQ, expliquée en détail au chapitre suivant. Cette technique est basée sur la génération adaptée des vecteurs de test en sous-ensembles de 4 vecteurs dont la combinaison des mesures de courant donne, en théorie, 0. L'équation 1.1 décrit la combinaison des mesures.

$$CDIDDQ = 0 = I(Vy_{1-1}) + I(Vz_{0-0}) - (\ I(Vw_{1-0}) + I(Vx_{0-1})\ ) \qquad (1.1)$$

Où $I(V_j)$ est la mesure de courant statique prise au vecteur V_j. L'objectif est de pouvoir, grâce aux quatre mesures, éliminer la variation du courant de fuite des transistors d'un vecteur à l'autre. De ce fait, si le résultat CDIDDQ est supérieur à un certain seuil, déterminé en fonction de la variation due à la mesure de courant, on considère le circuit défectueux. Il est à noter que cette dernière variation est la seule qui demeure, les autres (d'une puce à l'autre et

d'un vecteur à l'autre) ayant été éliminées. Tel que décrit plus loin, cette technique a été validée (Haithem 2011) en l'appliquant sur différents circuits du banc d'essai ISCAS89 (Brglez 1989). L'objectif de ce projet est d'améliorer l'application de cette technique pour détecter des courts-circuits plus faibles (résistifs) pour des circuits consommant plus de courant.

1.3.3 Émulation d'ASIC à l'aide de FPGA

Les ASICs sont des circuits intégrés dédiés. Une puce dédiée est conçue et fabriquée dans le but de répondre à des spécifications particulières. L'intérêt de ce genre de circuits intégrés est dans la production de masse où les coûts par puce sont faibles, dans la mesure où les frais non récurrents sont amortis par le très grand volume de vente. Le design et la fabrication de puces dédiées ASIC coûtent très cher et nécessitent beaucoup de temps (Maxfield 2004). De plus, une fois le design complété et la pièce implémentée, nous faisons face à un système gelé (*Frozen system*), ce qui signifie qu'il est impossible de le modifier. Il nous apparaît alors l'importance d'avoir un circuit parfaitement fonctionnel pour la fabrication. Afin de pouvoir a priori valider la fonctionnalité du design avant la fabrication, il est possible d'utiliser des FPGAs (Courtoy, M. 1995).

Les FPGAs comportent, de nos jours, plusieurs millions de portes logiques équivalentes et permettent de réaliser pratiquement n'importe quel système numérique par le biais d'une simple programmation (Maxfield 2004). Il y a plusieurs avantages au prototypage sur FPGA. Notons entre autres que la conception sur FPGA a l'avantage d'être peu coûteuse et rapide en plus d'être modifiable une fois implémentée, contrairement aux ASICs. En plus de pouvoir être utilisée pour les tests de fonctionnalité, l'émulation d'ASICs sur FPGA peut aussi servir pour l'injection d'un grand nombre de pannes (De Andres 2006) puisque l'on peut générer un circuit avec des défectuosité logique. Le FPGA étant déjà testé par le manufacturier, nous supposons ce dernier sans défectuosité. Ainsi, il est possible d'implémenter sur le FPGA le design original modifié comportant des fautes, en connectant par exemple deux nœuds ensemble pour créer un court-circuit. L'objectif étant soit de vérifier le comportement du

design face à la faute ou bien de valider la détection de la faute par le système de test. Pour ce projet, nous allons émuler un ASIC sur FPGA afin d'expérimenter la technique de test CDIDDQ. Nous ne nous servirons pas de l'émulation dans le but de test fonctionnel puisque nous utiliserons les circuits ISCAS-89 qui ont déjà été vérifiés. Nous utiliserons plutôt l'émulation pour insérer des pannes de type court-circuit, afin de procéder à leur détection grâce à CDIDDQ. Nous nous intéressons donc à la détection de la panne plutôt qu'à son impact sur la fonctionnalité du circuit.

1.4 Techniques de mesure de courant pour le test

L'utilisation d'une technique de test basé sur la mesure de courant nécessite une attention particulière à l'endroit de la prise et du paramétrage de la mesure. Selon (Chakravarty et Thadikaran 1997), il existe trois endroits où cette mesure peut être effectuée. En effet, il est possible de mesurer le courant à l'intérieur du circuit à tester à l'aide de senseurs (built-in current monitors), avec un circuit spécialisé externe inséré entre le bloc d'alimentation et le circuit (loadboard monitors), ou bien directement dans le bloc d'alimentation (tester based monitors). La mesure à l'intérieur de la puce, grâce à sa position rapprochée du point de mesure, permet d'atteindre des vitesses de mesure très élevées. En général, la lecture de courant est exécutée à des endroits spécifiques ce qui permet d'éviter les grands courants DC à l'intérieur du circuit et ainsi augmenter la précision des mesures. De plus, la mesure de courant interne est comparée à une valeur prédéfinie et seulement le résultat de la comparaison est retourné (succès ou échec) et non la mesure. Pour leur part, les circuits spécialisés externes se trouvent en général très près de l'unité à tester, ce qui leur permet une vitesse de mesure relativement élevée. Ils ont la possibilité d'être dédiés à des applications particulières et on retrouve une grande diversité sur le marché de ce type de produit. Le principal inconvénient est que cette approche nécessite l'ajout d'un module supplémentaire sur la plateforme de test déjà existante. Pour finir, la mesure qui provient du bloc d'alimentation, se trouvant à la plus grande distance du circuit à tester, est en général moins rapide que les deux autres. Cela rend aussi ce type de mesure plus flexible que les autres puisqu'elle est moins dédiée à une application en particulière.

Dans le cadre de ce projet, nous ne pouvons pas utiliser la mesure de courant à l'intérieur du circuit, car de tels modules n'existent pas dans les FPGAs. Nous avions donc le choix entre la mesure de courant provenant du bloc d'alimentation ou d'un circuit externe spécialisé. Selon (Hans Manhaeve 2005), il est préférable d'utiliser un circuit externe spécialisé, comme le QD-1011HC de Q-Star, qui permet d'avoir une erreur moyenne plus basse et une vitesse d'acquisition plus grande par rapport au bloc d'alimentation. L'erreur moyenne est un paramètre important dans la mesure de courant puisqu'il nous indique quelle variation de courant maximum peut être détectée de façon répétable. Ainsi, pour un système de mesure dont l'erreur moyenne est de 20µA, on ne peut détecter de façon répétable, donc à chaque mesure, que des défectuosités qui génèrent une variation de courant de plus de 20µA (Hans Manhaeve 2005). Le bloc d'alimentation, le N6705 d'Agilent par exemple, possède une erreur de 30µA tandis que le module QD-1011HC de Q-Star est de 4µA, tous deux pour une plage de 100mA. Aussi, le QD-1011HC peut effectuer les mesures bien plus rapidement que le bloc d'alimentation. Malgré que le QD-1011HC possède de meilleures caractéristiques, nous n'avons pas choisi ce module. Il nécessitait l'ajout d'un module supplémentaire au montage, ce qui le complexifiait et en augmentait les coûts. Finalement, ce module offre une plage de lecture limitée à 2A contrairement à 3A pour le bloc d'alimentation, qui fut par conséquent notre choix. Il est clair, cependant, que le module de Q-Star pourrait être mis à contribution dans l'application de CDIDDQ pour les circuits intégrés dont la consommation ne dépasse pas 2A. L'utilisation de ce module pourrait faire l'objet de travaux futurs.

1.4.1 Utilisation de deux sources

Dans notre montage, afin de pouvoir garder une erreur moyenne de 30 µA pour une plage plus élevée que 100mA, par exemple 3A, nous utilisons deux sources pour nos tests. En effet, l'augmentation de la plage de lecture entraîne aussi l'augmentation de l'erreur moyenne, ce qui est néfaste pour la précision de nos mesures de courant. Le montage comporte une source de courant et une source de tension qui alimentent le circuit sous test, comme affiché à la

figure 1.2. La source de courant est utilisée pour fournir le courant de base au circuit afin que la source de tension ne soit utilisée que pour mesurer la variation du courant.

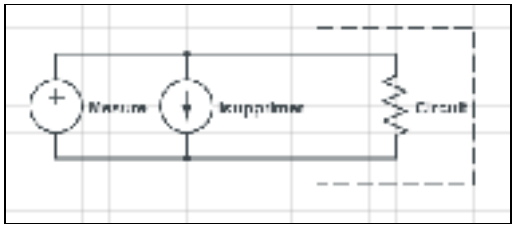

Figure 1.2 Utilisation de 2 sources pour l'alimentation et mesure du courant

Ainsi, pour un circuit consommant 3A, la source courant pourrait fournir 2.99A afin que la source de tension ait à fournir moins de 100mA pour rester dans la plage de 100mA et d'avoir une erreur moyenne de 30 µA. Cette méthode permet donc de conserver une erreur moyenne plus basse que si nous utilisions la plage de 3A, qui possède une erreur moyenne de 1350µA. L'utilisation de deux sources, comme affiché à la figure 1.3, est souvent employée pour retirer un courant d'arrière-plan afin de cibler la mesure de courant pour l'application. C'est entre autres le cas pour des mesures de petits courants et grandes résistances.

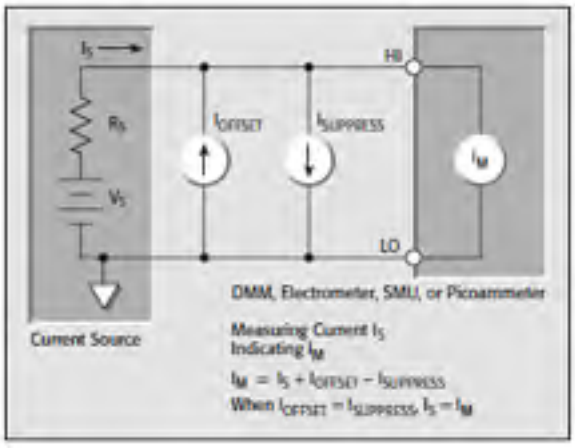

Figure 1.3 Ajout d'une source courant pour supprimer un courant de décalage
Tirée de Making Precision Low Current and High Resistance Measurements (2012, p.5)

Comme on peut le constater, l'ajout d'une source de courant externe ($I_{SUPRESS}$) peut permettre d'éliminer un courant de consommation supplémentaire (I_{OFFSET}) si ceux-ci sont égaux. Ainsi le courant mesuré est celui consommé par I_M.

1.5 Résultats CDIDDQ antérieurs

Dans le cadre d'un travail de maîtrise (Haithem 2011), on a pu démontrer l'efficacité de CDIDDQ à détecter des pannes de courts-circuits d'une résistivité moyenne sur des circuits émulés sur un FPGA à faible consommation de courant. Les résultats ont en effet montré la capacité à détecter des pannes de court-circuit de l'ordre de 80k Ω sur un montage comprenant uniquement le circuit émulé sur FPGA qui consomme environ 70mA. Notons que l'application de CDIDDQ utilisée, l'ensemble des résultats associés et le montage expérimental sont présentés dans le chapitre 2.

Sachant que le bruit est un grand préjudice aux tests utilisant la mesure de courant (Chakravarty, S. et P. J. Thadikaran 1997) nous allons dans le cadre de ce projet le réduire de manière considérable en vue d'obtenir de meilleurs résultats que précédemment. Nous parviendrons à ce point en épurant le montage sur lequel les mesures de courant sont faites en plus d'un meilleur paramétrage et utilisation de l'appareil de mesure. Nous allons par la suite ajouter au montage une surconsommation afin de voir l'efficacité de CDIDDQ pour des circuits intégrés plus énergivores. L'objectif est donc d'approfondir et élargir les résultats pour le test CDIDDQ. À l'extérieur du laboratoire, ces résultats tentent de fournir des résultats solides pour prouver aux compagnies la validité et l'étendue de la technique.

1.6 Conclusion

Dans ce chapitre, nous avons présenté les notions de base des modèles de pannes, de l'émulation des ASICs et des résultats expérimentaux avec CDIDDQ déjà obtenus. Nous avons pu constater que les modèles de pannes existants qui couvrent les courts-circuits possèdent certaines lacunes, lorsqu'utilisés par des techniques de test basées sur les valeurs logiques (ou tension), d'où l'utilisation des tests par mesure de courant. La perte d'efficacité

de ces derniers a mené à la création de techniques de post-traitement des mesures de courant, auxquelles appartient CDIDDQ. Pour les circuits à haute consommation, il est possible d'ajouter une source de courant supplémentaire afin que celle utilisée pour les mesures ne fournisse qu'un petit courant, ce qui permet de garder une échelle de mesure avec une bonne résolution. Aussi, l'expérimentation de CDIDDQ sur des ASIC émulés sur FPGA a permis de prouver le fonctionnement de cette technique pour laquelle nous visons à obtenir de meilleurs résultats. Dans le prochain chapitre, nous validons les vecteurs CDIDDQ générés en simulation. Ensuite, nous expliquons le fonctionnement de l'application expérimentale de CDIDDQ sur des circuits FPGA. Enfin, nous procéderons à la reproduction des résultats déjà obtenus (Haithem 2011) afin de revalider l'application de la méthode CDIDDQ.

CHAPITRE 2

LES PATRONS DE TEST CDIDDQ

2.1 Introduction

Dans ce chapitre, nous allons en premier lieu donner un peu plus de détails sur le fonctionnement et la génération des patrons de test CDIDDQ. Par la suite, nous allons expliquer comment procéder pour appliquer ces patrons de test. Pour finir, nous allons présenter les premiers résultats obtenus de l'application d'un de ces patrons de test sur un montage comportant un ASIC émulé sur FPGA.

2.2 Les patrons et vecteurs

Comme expliqué dans le chapitre précédent, CDIDDQ est une technique de test basée sur la lecture du courant. Afin d'effectuer le calcul CDIDDQ, nous devons avoir la lecture du courant consommé par le circuit lorsque les deux nœuds testés sont respectivement aux niveaux logiques 0-0, 0-1, 1-0 et 1-1. Afin d'obtenir ces niveaux logiques, nous devons fournir au circuit les bons stimuli aux entrées de celui-ci. Il nous est donc nécessaire de générer ces stimuli qui seront sous la forme de vecteurs afin de les appliquer au circuit.

2.3 Génération des patrons de CDIDDQ

Il n'existe présentement aucun outil ATPG commercial qui permet de générer directement l'ensemble des vecteurs de test CDIDDQ. Une méthode (Hariri 2009) a été mise au point pour générer ces vecteurs de test. Nous n'allons ici résumer que l'essentiel en expliquant les modifications que l'on doit apporter au circuit en vue du test ainsi que la génération des vecteurs de test CDIDDQ.

2.3.1 Modification en vue du test

Pour créer les vecteurs de test, on doit premièrement avoir un circuit qui est testable. Cette première étape consiste conventionnellement à l'insertion de chaînes de bascules à balayage (*scanchains*). Ceci permet d'augmenter la contrôlabilité et l'observabilité du test. On rappelle que la contrôlabilité exprime la facilité à pouvoir appliquer une valeur à n'importe quel nœud du circuit tandis que l'observabilité exprime la facilité à pouvoir capturer une valeur de n'importe quel nœud du circuit. Cette insertion nécessite d'abord le remplacement des bascules comprises dans le circuit par des bascules à balayage (*scan flip-flops*), illustré à la figure 2.1. Il est important de noter que le remplacement des bascules modifie le circuit et augmente le nombre de transistors nécessaire à la réalisation du circuit.

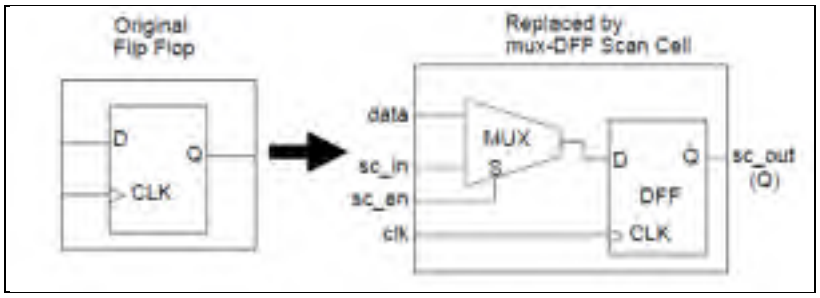

Figure 2.1 Remplacement d'une bascule normale par une bascule à balayage
Tirée de Scan and ATPG Process Guide (2008 p.66)

Comme on peut le constater avec les bascules à balayage, il est désormais possible de choisir le signal d'entrée des bascules par le biais d'un multiplexeur. En connectant ensuite les bascules en série (voir figure 2.2), on crée un (ou plusieurs) registre(s) à décalage à l'intérieur de notre circuit intégré. Ainsi, il est possible de lire ou d'écrire la valeur de chacune des bascules. Nous pourrons donc placer les bascules de notre circuit dans un état précis en vue du test.

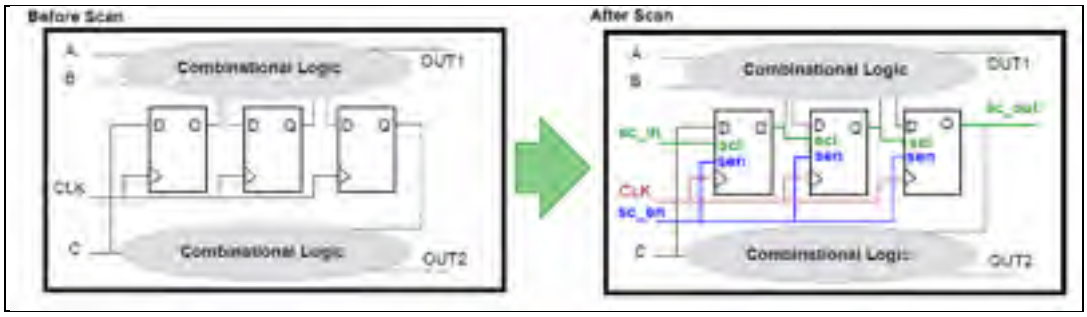

Figure 2.2 Chaîne de bascule à balayage
Tirée de Scan and ATPG Process Guide (2008 p.41)

Les bascules à balayage nécessitent donc l'utilisation de trois signaux, soit Sc_en, Sc_in et Sc_out. Le signal Sc_en permet de sélectionner entre le signal originalement connecté à la bascule et le signal Sc_in. Le signal Sc_in sera utilisé pour injecter les valeurs désirées aux bascules, tandis que Sc_out sera utilisé pour lire la valeur des bascules.

Afin de procéder à l'insertion des bascules de scan, nous utilisons le logiciel DFTadvisor. Comme illustré à la figure 2.3, on doit fournir au logiciel DFTadvisor la description du circuit (sous format Verilog ou VHDL) dans lequel on désire faire l'insertion des chaînes de bascules à balayage ainsi que la librairie qui définit ces bascules. En retour, DFTadvisor produira le circuit avec les bascules à balayage ainsi que le fichier de la configuration APTG qui sera utilisé par le logiciel de génération de patron de test (ATPG) expliqué un peu plus loin.

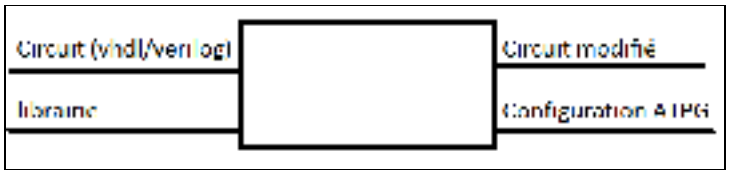

Figure 2.3 Entrées et sorties du logiciel DFTadvisor

2.3.2 L'extraction des capacités parasites

La prochaine étape à compléter afin de générer les vecteurs de test est d'obtenir le dessin des masques de notre circuit et d'en extraire les sites comportant des capacités parasites. Les sites à capacités parasites représentent les endroits du circuit où les traces sont les plus rapprochées. De ce fait, ces sites ont une plus grande probabilité d'être victimes de pannes de courts-circuits. Ne cibler que les sites probables réduit le nombre de vecteurs CDIDDQ à produire. L'extraction des capacités parasites est une étape obligatoire dans l'application de CDIDDQ pour le test d'un circuit ASIC. Dans le cadre de ce projet, comme nous émulons les circuits ASIC à l'aide d'un circuit FPGA, nous considérons que les sites potentiels de courts-circuits ont déjà été identifiés, soit par cette extraction effectuée à partir du dessin des masques ou encore de manière aléatoire. Pendant nos expérimentations, nous allons tenter de détecter un nombre restreint de pannes de court-circuit via le test de CDIDDQ.

2.3.3 Génération des vecteurs

Nous avons maintenant tous les prérequis pour la génération des vecteurs. Rappelons l'équation de CDIDDQ :

$$CDIDDQ = I(Vy_{1-1}) + I(Vz_{0-0}) - (\ I(Vw_{1-0}) + I(Vx_{0-1})\ ) \tag{2.1}$$

En ce sens, nous devons donc être en mesure de produire des sous-ensembles de quatre vecteurs qui appliqueront les valeurs 1-1, 0-0, 0-1 et 1-0 sur les deux nœuds où l'on teste une panne de court-circuit. Ce processus se divise en deux étapes qui nous fourniront chacune deux des quatre vecteurs.

2.3.4 Génération des deux premiers vecteurs

Comme mentionné précédemment, aucun outil commercial ne peut générer l'ensemble des vecteurs CDIDDQ. Toutefois, il est possible d'utiliser le logiciel FastScan pour la génération

des deux premiers vecteurs CDIDDQ de chaque sous-ensemble. Ce dernier nous permet de générer des vecteurs de tests pour plusieurs modèles de pannes, comme ceux présentés au chapitre 1. Nous allons générer les patrons de tests de pannes de délais de transition (TDF) pour l'ensemble du circuit.

Un patron de test comprend un ou des vecteurs à injecter au circuit afin de faire apparaître la panne sur un ou des nœuds spécifiques ainsi que la réponse attendue du nœud face à ces vecteurs. La figure 2.4 affiche un exemple d'un patron de test généré par FastScan pour le modèle de panne collé-à.

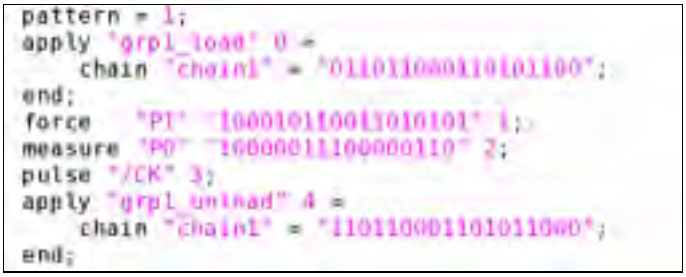

Figure 2.4 Patron de test FastScan

Le paramètre chain1 représente au départ la suite de valeurs qui sera injectée via Sc_in dans la chaîne de registres à balayage qui a été au préalable insérée, PI les valeurs à injecter aux entrées et PO les valeurs à mesurer aux sorties du circuit. La deuxième apparition du paramètre chain1 représente la suite de valeurs attendues via Sc_out. Rappelons toutefois que dans le cas de CDIDDQ, seuls les vecteurs d'entrée nous importent, les sortie n'étant pas vérifié puisque le test est basé sur le courant et non sur le niveau logique résultant des vecteurs d'entrée.

Pour obtenir la liste des pannes de transition, nous devons fournir à FastScan le circuit comportant les bascules à balayage, la librairie que nous avions utilisée précédemment avec DFTadvisor et la configuration ATPG fournie par DFTadvisor. Nous aurons alors une liste de patrons de tests qui, une fois les vecteurs injectés, créeront des transitions pour des nœuds

spécifiques du circuit. Nous allons utiliser cette liste de patrons dans FastScan afin de connaître les nœuds couverts par chacun des patrons. Finalement, nous allons utiliser la liste des nœuds couverts pour obtenir la liste de ceux couverts par les pannes de court-circuit. Les étapes sont affichées à la figure 2.5.

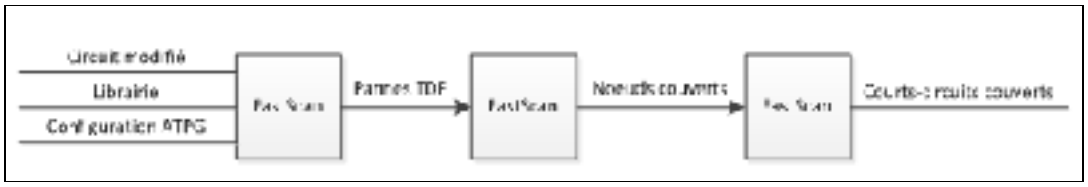

Figure 2.5 Génération des deux premiers vecteurs

Suite à l'analyse de cette liste, on connait maintenant quels patrons couvrent quels courts-circuits dans le circuit. C'est à ce moment que l'on doit comparer les courts-circuits couverts avec les sites à capacités parasites afin de déterminer quels vecteurs nous sélectionnerons pour le test. À titre informatif, les pannes TDF générées par FastScan pour le circuit s1196 d'ISCAS-89, que nous utiliserons dans nos tests, comprennent 420 patrons qui couvrent 6054 pannes comme affiché dans la figure 2.6.

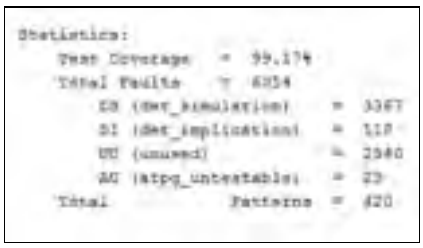

Figure 2.6 Statistiques des pannes TDF pour le circuit S1196

2.3.5 Génération des deux derniers vecteurs

Nous possédons, pour chaque patron, deux vecteurs qui généreront des transitions sur deux ou plusieurs nœuds. Nous devons créer les deux autres vecteurs manquants pour chacun des

sous-ensembles de quatre vecteurs. La figure 2.7 est un exemple simplifié, dans lequel on a omis les bascules, où sont appliqués des vecteurs TDF sur les nœuds M et N d'un circuit.

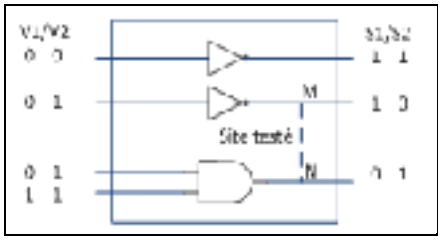

Figure 2.7 Vecteurs TDF Vw et Vx

On peut remarquer que, dans l'exemple, on obtient, grâce aux deux transitions créées par l'application de vecteurs, les niveaux 1-0 et 0-1 sur les nœuds M et N qui sont respectivement les vecteurs Vw et Vx de CDIDDQ. Notons que les deux vecteurs qui sont générés par FastScan ne sont pas toujours utilisables comme vecteurs Vw et Vx. On doit en effet s'assurer, à l'aide un test d'indépendance, de pouvoir séparer chacune des transitions sur les deux nœuds dans l'optique de créer les autres vecteurs manquants, soit ici Vy (1-1) et Vz (0-0) qui sont respectivement les vecteurs Vy et Vz de CDIDDQ. Dans notre exemple, on constate facilement que les transitions sont indépendantes, car il est possible de les exécuter séparément et sans influencer les autre partie du circuit, comme la porte inverseur qui n'a pas de lien avec notre court-circuit. Cette indépendance est nécessaire, car les 2 vecteurs supplémentaires doivent être créés de telle sorte que la combinaison des 4 mesures de courant élimine les courants de fuite statique des transistors. Pour commencer, nous allons créer une transition seulement sur le nœud N comme illustré à la figure 2.8.

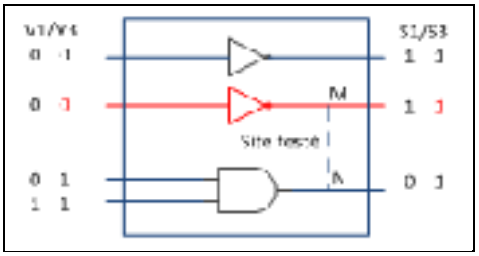

Figure 2.8 Création du vecteur Vy

Nous avons alors le troisième vecteur, Vy, qui nous permet d'obtenir le niveau 1-1. On remarque que l'on ne modifie pas l'état de la porte inverseur qui n'est pas en lien avec la panne. Il ne nous reste plus qu'à procéder de la même façon pour le nœud M.

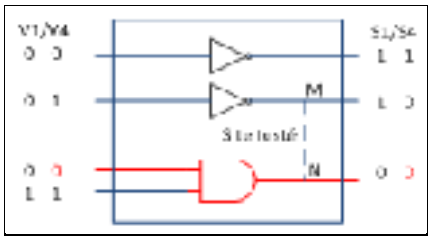

Figure 2.9 Création du vecteur Vz

Comme affichés dans la figure 2.9, nous avons créé le dernier vecteur, Vz, qui nous permet d'obtenir le niveau 0-0. Dans ce cas de figure, les vecteurs CDIDDQ seraient comme dans le tableau 2.1.

Tableau 2.1 Exemple de vecteurs CDIDDQ

Nom du vecteur	**Valeur des Vecteurs**	**Nœud M**	**Nœud N**
Vx	0111	0	1
Vy	0011	1	1
Vz	0101	0	0
Vw	0001	1	0

Avec les mesures de courant pour chaque vecteur, on peut alors appliquer CDIDDQ.

$$CDIDDQ = I(Vy_{1-1}) + I(Vz_{0-0}) - (\ I(Vw_{1-0}) + I(Vx_{0-1})\ ) \qquad (2.2)$$

Il est important de comprendre que, pour un circuit plus complexe, il est difficile et long de déterminer manuellement si les transitions sont indépendantes et de créer les vecteurs

manquants qui mènent à l'élimination des courants de fuite. C'est pourquoi un logiciel a été mis au point (Thibeault et Hariri 2009) permettant de vérifier l'indépendance des transitions et de générer les deux vecteurs manquants aux patrons de test des pannes de transition fournis par FastScan.

2.4 Présentation des patrons de test CDIDDQ

Chaque patron (sous-ensemble de 4 vecteurs) de test CDIDDQ permet de tester (au moins) une panne de court-circuit à un endroit distinct. Précédemment, seulement un patron, généré pour le circuit S1196 d'ISCAS-89, a été testé expérimentalement (Haithem 2011) : le patron p17. Dans le cadre de ce projet, nous allons expérimenter la technique de test CDIDDQ avec trois autres patrons (p5, p6 et p29) pour le même circuit afin de confirmer sa capacité à détecter des courts-circuits à divers endroits du circuit. Le nom des patrons réfère aux numéros de patrons de la liste de test TDF générée pour le circuit s1196 par FastScan, donc p17 est la 17^{e} patron de la liste. Le choix des patrons n'a pas été fait au hasard, il n'aurait pas été possible de prendre, par exemple, les patrons 1, 2, 3 et 4, car ils ne couvraient pas des courts-circuits et/ou les quatre transitions n'étaient pas indépendantes. Les tableaux 2.2, 2.3, 2.4 et 2.5 affichent les quatre patrons de test CDIDDQ pour le circuit s1196 d'ISCAS-89. On peut voir dans ce tableau les valeurs pour les signaux d'entrée Gx et Sc_en nécessaire pour obtenir chacun des vecteurs CDIDDQ.

Tableau 2.2 Patron de test 5

Vecteurs	**G 0**	**G 1**	**G 10**	**G 11**	**G 12**	**G 13**	**G 2**	**G 3**	**G 4**	**G 5**	**G 6**	**G 7**	**G 8**	**G 9**	**Sc_en**
Vw	0	1	0	0	0	0	0	0	1	0	1	1	0	1	1
Vy	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0
Vw	0	1	0	0	0	0	0	0	1	0	1	1	0	1	1
Vz	1	1	0	0	0	1	0	1	1	1	1	1	0	1	0
Vw	0	1	0	0	0	0	0	0	1	0	1	1	0	1	1
Vx	1	1	0	0	0	1	0	1	1	1	1	1	0	1	0
Scan-In	1101100000010010001														

Tableau 2.3 Patron de test 6

Vecteurs	**G 0**	**G 1**	**G 10**	**G 11**	**G 12**	**G 13**	**G 2**	**G 3**	**G 4**	**G 5**	**G 6**	**G 7**	**G 8**	**G 9**	**Sc_en**
Vy	0	0	0	1	0	1	0	1	1	1	0	0	0	0	1
Vz	0	1	1	1	1	0	1	0	1	0	0	1	1	1	0
Vy	0	0	0	1	0	1	0	1	1	1	0	0	0	0	1
Vx	0	1	0	1	1	0	1	0	1	0	0	1	0	1	0
Vy	0	0	0	1	0	1	0	1	1	1	0	0	0	0	1
Vw	0	1	1	1	1	0	1	0	1	0	0	0	1	0	0
Scan-In	0010110111110110101														

Tableau 2.4 Patron de test 17

Vecteurs	**G 0**	**G 1**	**G 10**	**G 11**	**G 12**	**G 13**	**G 2**	**G 3**	**G 4**	**G 5**	**G 6**	**G 7**	**G 8**	**G 9**	**Sc_en**
Vw	1	1	1	0	1	0	0	0	1	0	1	0	0	0	1
Vx	0	1	1	1	1	1	1	0	1	0	1	1	0	1	0
Vw	1	1	1	0	1	0	0	0	1	0	1	0	0	0	1
Vy	0	1	1	0	1	1	1	0	1	0	1	1	0	0	0
Vw	1	1	1	0	1	0	0	0	1	0	1	0	0	0	1
Vz	0	1	1	1	1	1	1	0	1	0	1	0	0	1	0
Scan-In	0010011101010101 10														

Tableau 2.5 Patron de test 29

Vecteurs	**G 0**	**G 1**	**G 10**	**G 11**	**G 12**	**G 13**	**G 2**	**G 3**	**G 4**	**G 5**	**G 6**	**G 7**	**G 8**	**G 9**	**Sc_en**
Vz	0	1	0	0	0	0	1	0	1	0	0	1	0	0	1
Vy	0	1	0	0	1	1	1	0	1	1	0	1	1	0	0
Vz	0	1	0	0	0	0	1	0	1	0	0	1	0	0	1
Vw	0	1	0	0	1	0	1	0	1	1	0	1	1	0	0
Vz	0	1	0	0	0	0	1	0	1	0	0	1	0	0	1
Vx	0	1	0	0	0	1	1	0	1	1	0	1	1	0	0
Scan-In	1111100000000101011														

Ces 4 tableaux exposent les patrons de tests CDIDDQ contenant les paires de vecteurs que l'on doit mettre aux entrées du circuit s1196 pour obtenir les niveaux voulus sur le nœud à tester en plus de la suite de valeurs qui doit être injectée dans la chaîne de bascules à balayage afin de placer les bascules à un état prédéterminé.

2.5 Simulation des patrons de test CDIDDQ

Il est impératif de valider les vecteurs des patrons de test CDIDDQ présentés précédemment. En effet, nous devons nous assurer que ces vecteurs placent les nœuds à tester au niveau désiré. On doit premièrement identifier les deux nœuds testés par chaque patron grâce à la liste des nœuds couverts fournie par FastScan. La source et le nom des nœuds testés pour chaque patron sont écrits dans le tableau 2.6.

Tableau 2.6 Source des nœuds testés

Patron	**M**	**N**
5	Sortie du NOT_71, nœud G501	Sortie du NAND_56, nœud G166
6	Sortie du NOT_11, nœud G536	Sortie du NOT_10, nœud G533
17	Sortie du NAND2_12, nœud G74	Sortie du NOT_37, nœud G275
29	Sortie du U28, nœud n188	Sortie du U27, nœud n189

Il est toutefois important de noter qu'un patron de test peut tester plus d'un nœud à la fois. Par exemple, le test CDIDDQ du patron 6 crée simultanément les transitions requises sur plusieurs nœuds. Comme nous créerons pour notre test un seul court-circuit résistif, nous n'utiliserons que deux nœuds parmi ceux possédant les transitions nécessaires. Le choix de la paire de nœuds fournissant les transitions voulues a été fait aléatoirement.

La prochaine étape consiste à la mise au point d'un banc de test qui injectera les vecteurs aux entrées du circuit à tester, soit le S1196. Ce banc de test agit comme un registre à décalage : il effectue la lecture d'un fichier texte comportant les valeurs à assigner aux broches d'entrée et les décale vers les broches d'entrées du circuit sous test à chaque coup d'horloge. La figure 2.10 affiche un fichier texte comportant les valeurs d'entrée à injecter au circuit.

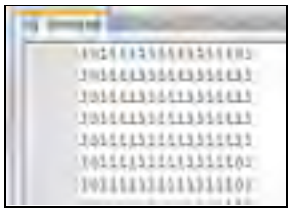

Figure 2.10 Fichier texte des vecteurs

Plus précisément, on commence par initialiser l'horloge aux niveaux bas et on procède à la lecture d'une ligne du fichier texte. Par la suite, on attend ¼ de la période d'horloge avant d'affecter les valeurs des entrées lues dans le fichier texte. Le délai entre la lecture et l'assignation a uniquement été mis en place afin de faciliter la lecture du chronogramme. On attend alors un autre ¼ de la période d'horloge avant de créer un front montant sur le signal d'horloge. Pour finir, on attend ½ de la période d'horloge, ce qui complète notre cycle, avant de boucler jusqu'à ce que le fichier texte ne comporte plus de nouvelle ligne. Dans ce cas, on procède à la fermeture du fichier texte et on arrête la simulation. La figure 2.11 affiche la machine à état du banc de test.

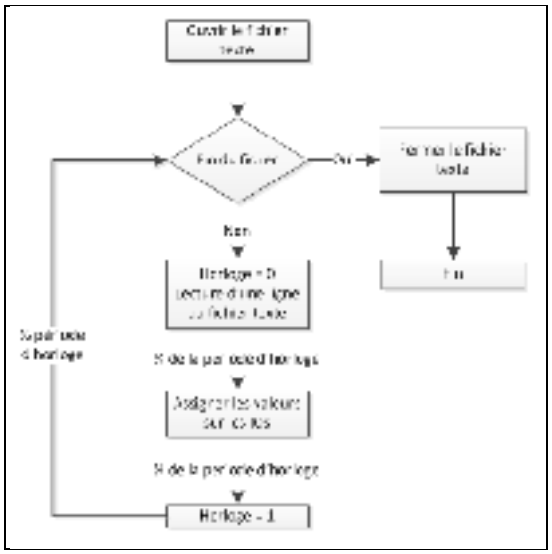

Figure 2.11 Machine à état du banc de test

En concevant le banc de test ainsi, il est possible de l'utiliser sans modification pour les quatre patrons de tests, puisque seul le fichier texte comportant la valeur des entrées à injecter doit être modifié.

Afin d'éviter d'avoir à écrire manuellement les valeurs dans un fichier texte, un programme Excel a été conçu. L'objectif principal de ce programme est de pouvoir créer le fichier texte nécessaire au banc de test directement avec les vecteurs CDIDDQs. La figure 2.12 affiche l'onglet du logiciel permettant de générer le fichier texte nécessaire afin de tester le patron 19 du s1196 sur notre banc de test.

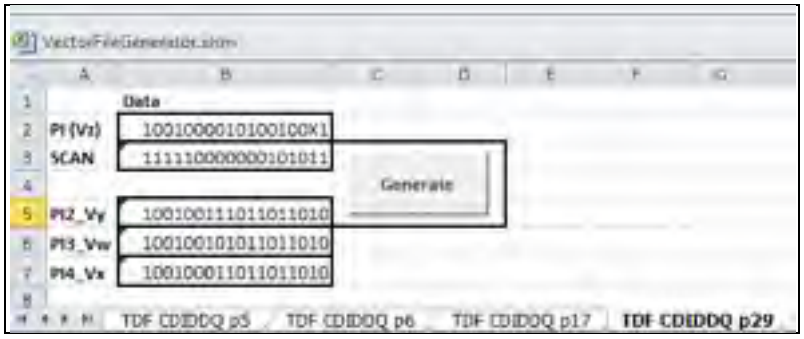

Figure 2.12 Programme de génération de vecteurs pour le banc de test

Les vecteurs transposés dans un fichier texte et le banc de test nous permettent de passer à la simulation des vecteurs CDIDDQ. Pour la simulation, nous avons utilisé ISIM de Xilinx. Les figures 2.13, 2.14, 2.15 et 2.16 affichent le résultat des simulations pour les quatre patrons de test présentés précédemment.

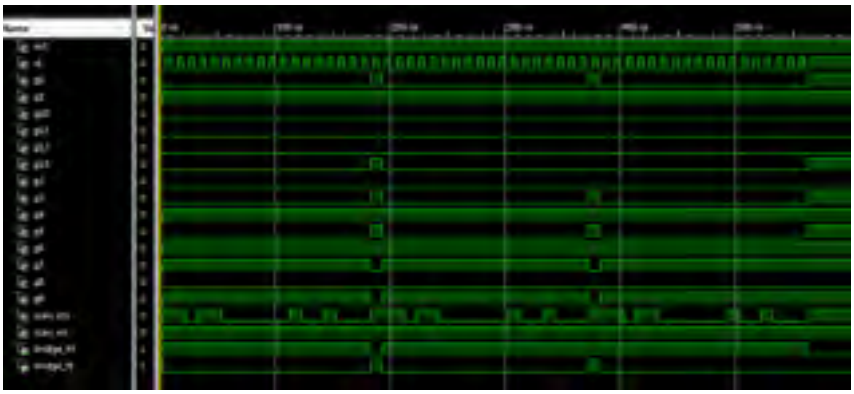

Figure 2.13 Simulation du patron 5

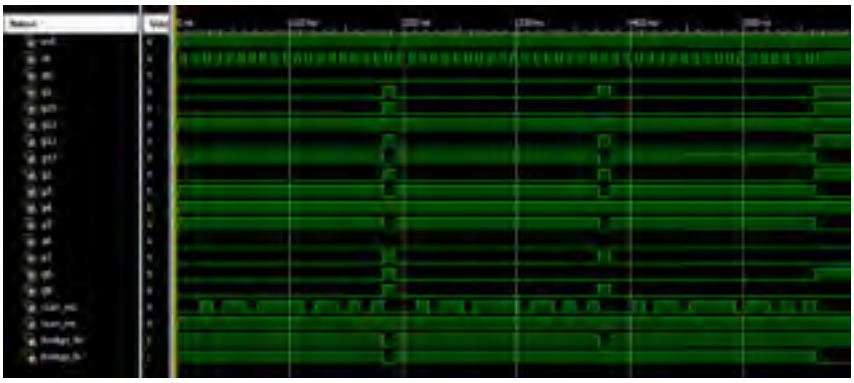

Figure 2.14 Simulation du patron 6

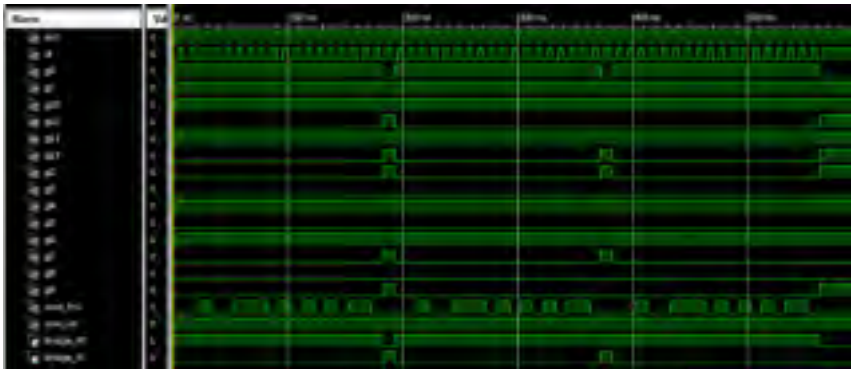

Figure 2.15 Simulation du patron 17

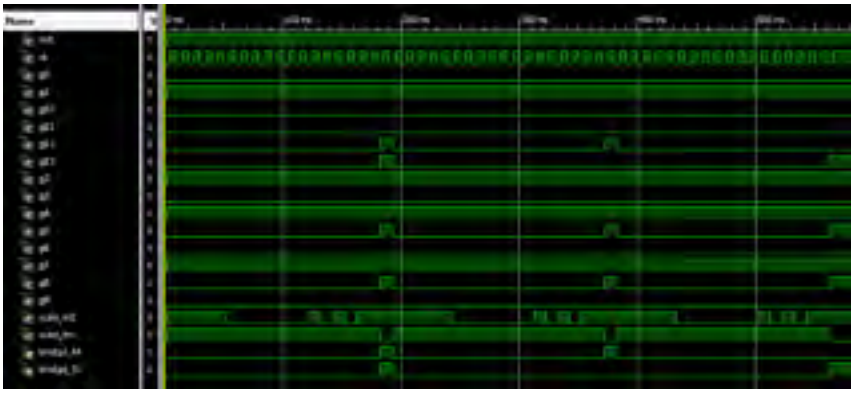

Figure 2.16 Simulation du patron 29

Comme on peut le constater, la simulation nous confirme que nous avons les transitions voulues sur les nœuds M et N lors de l'application des vecteurs CDIDDQ. En observant les chronogrammes, on remarque que l'application des deux premiers vecteurs crée une transition sur les deux nœuds tandis que les deux autres vecteurs créent une seule transition sur un seul des nœuds. On remarque aussi que l'ordre de ces transitions diffère d'un patron à l'autre. Ainsi, la première paire de vecteurs du patron 6 crée une transition 1 à 0 sur M et N, alors que la première paire de vecteurs du patron 29 crée une transition 0 à 1 sur M et N. Comme expliqué précédemment, ceci est causé par FastScan qui crée des transitions montantes ou descendantes.

En résumé, la simulation des patrons de test CDIDDQ sur le circuit s1196 nous a permis de valider que les vecteurs plaçaient les nœuds à tester au bon niveau. Nous sommes donc en mesure d'affirmer que les patrons et les circuits correspondant à ceux-ci sont fonctionnels et prêts à l'implémentation.

2.6 Application expérimentale de CDIDDQ

Grâce à la simulation, nous avons pu confirmer que les vecteurs de test CDIDDQ appliquaient les niveaux requis sur les nœuds à tester. Puisque nous désirons être en mesure d'appliquer les tests CDIDDQ pour détecter une panne de court-circuit, nous devrons en insérer une entre les deux nœuds testés. Le court-circuit inséré devra être résistif afin de pouvoir déterminer jusqu'à quel point nous sommes en mesure de détecter la panne. Pour procéder à l'insertion d'un court-circuit résistif variable, nous allons implémenter le circuit sur un FPGA en dirigeant les nœuds à tester sur des broches de sortie afin d'y placer la résistance voulue pour créer un court-circuit résistif. La figure 2.17 reflète l'approche décrite.

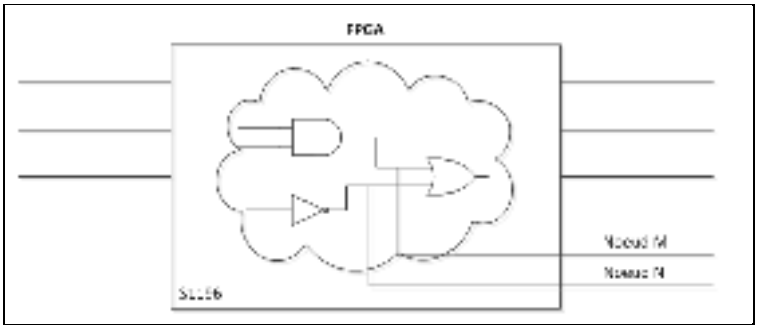

Figure 2.17 Redirection des nœuds testés

Puisque chacun des sous-ensembles de quatre patrons de test couvre un endroit différent du circuit et que nous désirons tester 4 sites de courts-circuits différents, nous devrons donc générer quatre circuits modifiés afin de pouvoir diriger les nœuds testés vers des broches du FPGA.

2.6.1 Montage expérimental

Afin d'injecter les vecteurs au FPGA qui émule le circuit à tester, nous allons utiliser un deuxième FPGA qui agira comme testeur CDIDDQ ; nous allons donc avoir un FPGA qui teste un autre FPGA. Nous allons alimenter le FPGA sous test avec le bloc d'alimentation duquel nous lirons la consommation de courant. Bien que la configuration exacte du montage ait changé à travers le temps, le principe de base, lui, n'a pas changé. La figure 2.18 résume notre montage expérimental.

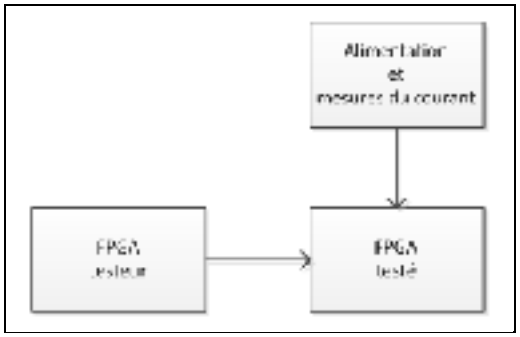

Figure 2.18 Montage expérimental

2.7 La mesure de courant de consommation

Nous avons les vecteurs CDIDDQ qui exerceront les transitions sur les nœuds à tester. Comme mentionné précédemment, nous allons utiliser une résistance externe afin de simuler un court-circuit résistif entre les nœuds M et N. Pour procéder au calcul de CDIDDQ afin de détecter la défectuosité, nous devrons lire la consommation de courant du circuit face aux quatre niveaux logiques (0-0, 0-1, 1-0 et 1-1) appliqués aux nœuds à tester. Comme on a pu le constater plus tôt, on doit appliquer trois paires de vecteurs pour obtenir les quatre niveaux désirés. Chaque paire de vecteurs crée deux niveaux logiques. Le premier vecteur de chaque paire est toujours le même, donc on le mesure une seule fois. Ainsi, pour l'application de la première paire nous allons procéder à deux mesures de courant contrairement aux deux

autres paires de vecteurs pour lesquels on procédera à une seule mesure. La figure 2.19 affiche, dans le chronogramme, le moment de la capture des mesures.

Figure 2.19 Mesure du courant

Il sera très important de connaître l'ordre des vecteurs testés afin d'affecter correctement la lecture du courant aux bons vecteurs. Cela est d'autant plus important, puisque chaque patron de test peut présenter les vecteurs dans un ordre différent. En ce sens, une mauvaise affectation peut engendrer une erreur dans le calcul de CDIDDQ.

2.7.1 Sélection de l'appareil de mesure

Comme mentionné précédemment, afin d'obtenir le courant consommé par le circuit sous test, nous allons procéder à la lecture du courant fourni par le bloc d'alimentation directement connecté à celui-ci. Pour ce faire, nous avons utilisé l'analyseur de puissance N6709 d'Agilent comme affiché à la figure 2.20.

Figure 2.20 L'analyseur de puissance N6709

Notons que c'est ce même appareil qui a été utilisé dans le cadre des expérimentations antérieures (Haithem 2011). Cet appareil a donc deux fonctions : il alimente notre circuit sous test et il mesure le courant consommé.

En résumé, il comporte quatre sorties DC pouvant fournir jusqu'à 3A chacune pour lesquelles il affiche la tension et le courant moyen sur son écran. De plus, cet appareil est apte à fournir une mesure de courant comportant jusqu'à 512000 échantillons avec une plage de mesure de 3000, 100 ou 0.2mA à un pas d'échantillonnage aussi basse que 10µs. Pour finir, il est possible de transférer les données acquises par le biais d'une connexion USB ou Ethernet. En raison de sa précision, en plus de sa capacité à communiquer avec l'ordinateur par un lien USB ou Ethernet, nous avons décidé de continuer à utiliser cet appareil.

2.7.2 Logiciel d'acquisition des mesures de courant

Dans l'expérimentation de CDIDDQ précédente, la mesure de courant utilisée pour le calcul de CDIDDQ était celle affichée à l'écran du bloc d'alimentation. Des efforts ont été déployés afin de mettre au point une meilleure méthode d'acquisition du courant. En effet, selon le manuel d'usager du N6709, le courant consommé affiché à l'écran à un taux d'échantillonnage de 60 Hertz ne nous donne qu'une approximation du courant. Dans ce projet, afin d'obtenir un maximum de précision dans notre mesure de courant, nous avons configuré l'appareil afin d'échantillonner le courant et d'enregistrer les mesures en mémoire. Une fois l'acquisition terminée, nous procéderons à la saisie des données via le port USB. Cette technique nous permettra d'obtenir une meilleure précision que celle obtenue par la lecture des valeurs affichées à l'écran. Nous croyons qu'en procédant ainsi, il sera possible d'obtenir de meilleurs résultats pour le test de CDIDDQ.

L'utilisation de cette technique requiert la mise au point d'un programme qui permettra d'établir une connexion USB entre l'appareil et l'ordinateur. En effet, Agilent ne fournit pas de gratuiciel permettant d'effectuer facilement cette tâche. Il a donc fallu mettre au point un logiciel d'acquisition spécifique à notre application. Une série d'exemples de l'utilisation de la librairie permettant la communication avec le bloc d'alimentation sur Excel est fournie par Agilent. Notre appareil ne figurant pas dans la liste pour lesquels un exemple était disponible, nous avons dû adapter le code fourni en plus d'ajouter des fonctionnalités propres à notre

application. Une fois les échantillons transférés sur Excel, il est possible d'obtenir diverses informations comme le graphique résultant, la moyenne, l'écart-type, etc. L'interface du logiciel est présentée à la figure 2.21.

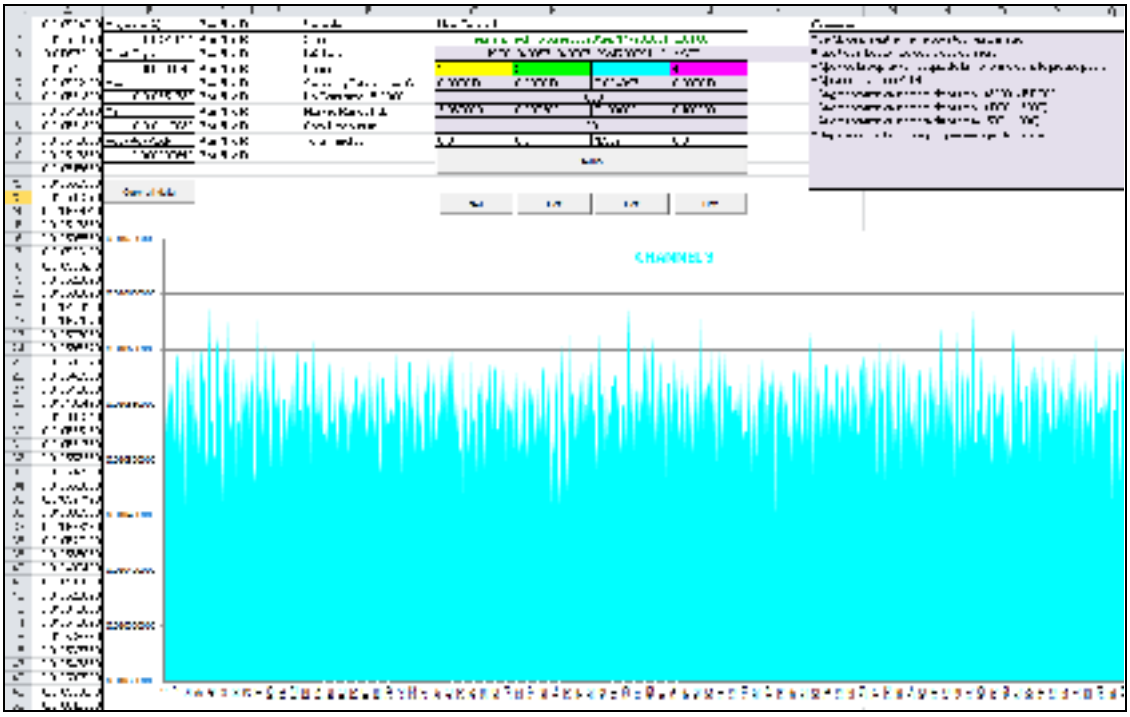

Figure 2.21 Programme d'acquisition pour le N6705A

Afin d'acquérir les données, le logiciel doit d'abord ouvrir le port USB et se connecter à l'appareil grâce à son identifiant USB. Par la suite, il faut aussi configurer l'appareil en vue de l'acquisition. On doit lui fournir la résolution, le nombre d'échantillons et la plage de la mesure, et ce, pour chaque canal que l'on désire utiliser. Il est important de vérifier que les paramètres envoyés sont à l'intérieur des capacités de l'appareil. Finalement, on peut procéder à l'acquisition des données pour la configuration que nous avons imposée. L'organigramme de l'acquisition des données par le programme Excel mis au point est représenté dans la figure 2.22.

Figure 2.22 Étapes de la mesure de courant

On remarque que le logiciel conçu ne communique avec l'appareil pour des fins d'acquisition de données. Il aurait été possible de configurer les valeurs des sorties du bloc d'alimentation en plus de pourvoir les activer/désactiver via la connexion USB, toutefois, pour des raisons de sécurité, nous avons opté pour configurer les sorties de l'appareil manuellement et nous avons choisi de procéder à l'acquisition automatiquement via l'ordinateur.

2.8 Premier résultat expérimental de CDIDDQ

Comme première application expérimentale de CDIDDQ, nous avons répété celle faite précédemment (Haithem 2011) afin de valider les résultats obtenus. Pour ce faire, nous avons utilisé le même code et le même matériel. Nous n'allons ici que décrire l'essentiel du fonctionnement de l'application des vecteurs CDIDDQ pour ce montage.

2.8.1 Le montage

Le montage consiste en deux cartes de développement Nexys2 qui comportent chacune un FPGA Spartan 3E. L'une d'elles est le testeur, l'autre, le circuit sous test. Les deux cartes sont connectées entre elles par le biais d'un adaptateur personnalisé. Elles sont reliées à l'adaptateur grâce à un connecteur comportant 40 broches du FPGA, celui-ci se trouvant à l'extrémité droite de la carte de développement. L'adaptateur nous permet aussi d'avoir accès aux broches connectées entre les deux cartes. Cela nous permet donc d'insérer notre défectuosité résistive. De plus, il est possible de connaître l'état logique des broches avec un analyseur logique afin de confirmer le bon fonctionnement du circuit. Le montage du testeur CDIDDQ est illustré à la figure 2.23.

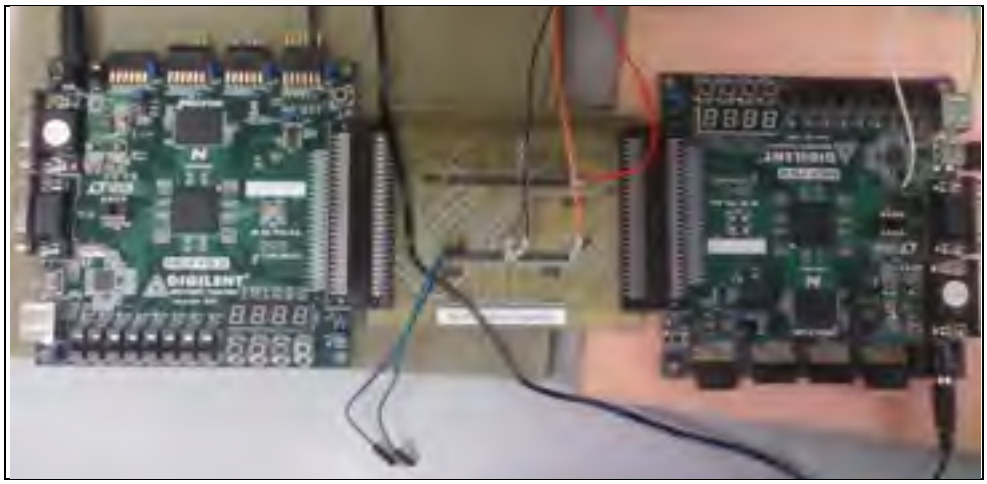

Figure 2.23 Montage d'Haithem

Pour la carte de développement comportant le circuit sous test, le cristal relié au FPGA ainsi que celui connecté au circuit intégré permettant la programmation par le port USB ont été retirés. L'objectif visé par ces modifications (Haithem 2011) matérielles est la réduction du bruit sur l'alimentation afin d'avoir des mesures de courant comportant moins de variations.

2.8.2 Validation

Avant de procéder aux mesures pour le test CDIDDQ, il est important de valider que nous ayons bel et bien les transitions nécessaires comme obtenues en simulation. Pour ce faire, nous allons utiliser un analyseur logique afin de vérifier le niveau des deux broches constituant le court-circuit testé. Les résultats sont affichés dans les figures 2.24, 2.25, 2.26 et 2.27. Notons que, comme Haithem (2011), nous n'utiliserons que le patron de test 17.

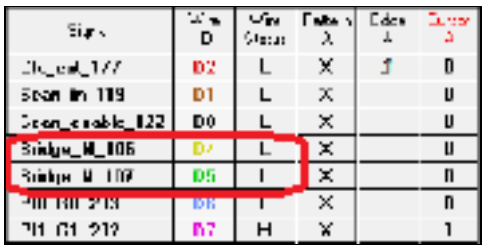

Figure 2.24 Vz (0-0)

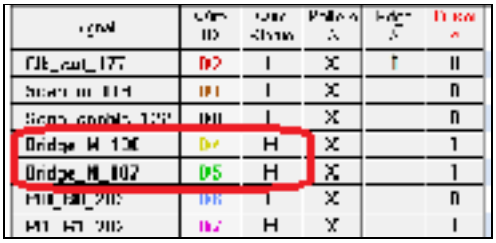

Figure 2.25 Vy (1-1)

Figure 2.26 Vx (0-1)

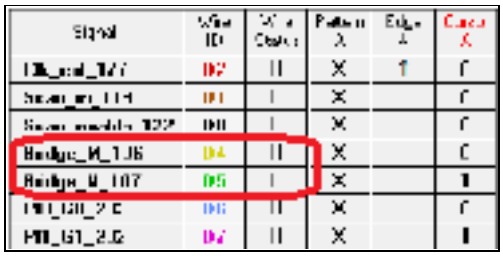

Figure 2.27 Vw (1-0)

Comme on peut le constater, les vecteurs du patron de test effectuent correctement les transitions sur le nœud à tester. On peut alors procéder à la mesure des courants consommés par la carte via le bloc d'alimentation pour chacun des vecteurs.

2.8.3 Application des vecteurs et résultats

Afin de mesurer le courant consommé, nous devons connecter l'alimentation de la carte qui comportera le circuit sous test directement au bloc d'alimentation. Pour sa part, la carte qui comportera le testeur sera connectée à l'alimentation murale. Il ne reste plus qu'à appliquer les étapes décrites à la figure 2.28 pour obtenir la valeur CDIDDQ.

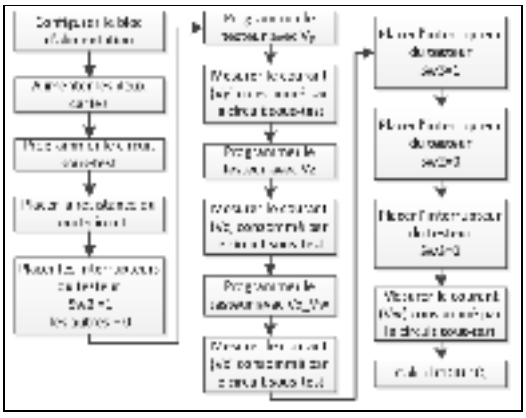

Figure 2.28 Procédure de mesure CDIDDQ

Pour ne pas surcharger le diagramme, nous n'avons pas inclus le débranchement du cristal qui a lieu avant chaque mesure. Nos résultats des calculs CDIDDQ ainsi que ceux

précédemment obtenus (Haithem 2011) sont affichés dans les tableaux 2.7 et 2.8 et sur les graphique 2.29 et 2.30. Dans notre tableau de résultats, on affiche pour chacun des vecteurs la moyenne des courants mesurés après leur application. Pour ce faire, nous avons mesuré 100 points à une vitesse d'échantillonnage de 10ms sur une plage de 100mA.

Tableau 2.7 Résultats obtenus (A)

R	**Vx**	**Vw**	**Vy**	**Vz**	**|Vx+Vw-Vy-Vz|**
infini	0.0743912	0.0743145	0.0748128	0.0737849	0.0001079
0.0994k	0.0859598	0.0863086	0.0747247	0.0738200	0.0237237
1k	0.0770793	0.0771775	0.0748574	0.0739862	0.0054131
2k	0.0758599	0.0757118	0.0748863	0.0740232	0.0026621
12.85k	0.0745436	0.0745609	0.0748245	0.0739187	0.0003613
38.6k	0.0744933	0.0744434	0.0745222	0.0736341	0.0007803
73.7k	0.0747304	0.0742246	0.0747457	0.0742272	0.0000178
81.2k	0.0741744	0.0744231	0.0749062	0.0734554	0.0002358
90.8k	0.0743299	0.0743481	0.074756	0.0739472	0.0001447

Tableau 2.8 Résultats d'Haithem (mA)
Tiré du mémoire d'Haithem (2011, p. 54)

R	**Vx**	**Vw**	**Vy**	**Vz**	**CDIDDQ**
Infini	80.15	80.14	80.15	80.16	0.02
100	80.17	80.14	84.34	84.35	8.38
1000	80.17	80.15	84.11	84.12	7.91
2000	80.18	80.14	83.73	83.75	7.16
13000	80.17	80.15	80.24	80.25	0.17
39030	80.17	80.14	80.19	80.2	0.08
80200	80.17	80.14	80.19	80.19	0.07
84200	80.15	80.14	80.15	80.16	0.02

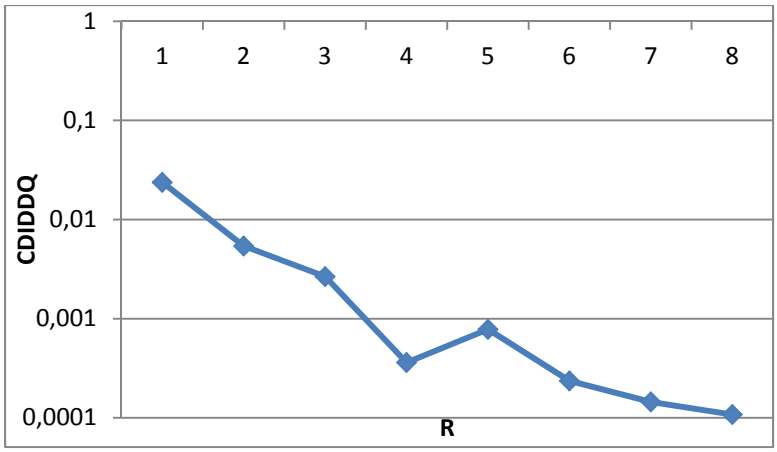

Figure 2.29 Graphique de CDIDDQ obtenus (A)

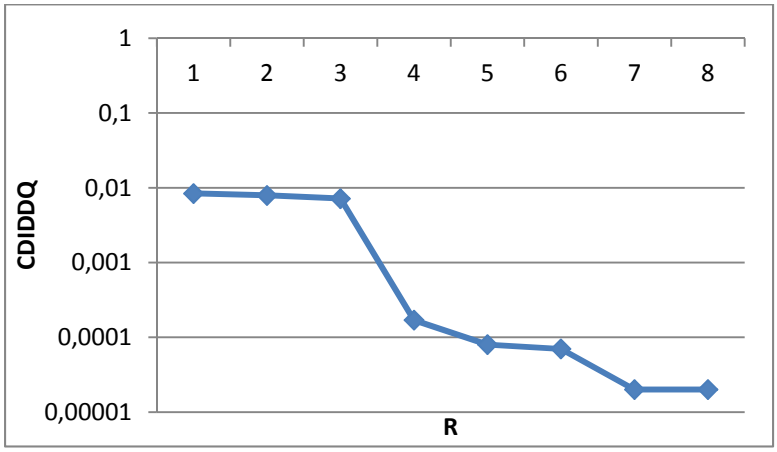

Figure 2.30 Graphique de CDIDDQ d'Haithem

2.8.4 Analyse des résultats

Les résultats obtenus sont légèrement différents (d'environ 8%) bien que nous ayons utilisé les mêmes patrons de test, montage, code et appareil de mesure. On remarque que, même sans résistance (R=infini), cette légère différence est présente. Cela pourrait être lié, en partie, à l'emploi d'une échelle de mesure différente que celle nous avons utilisée. Puisque nous ne possédons pas d'information précise sur la configuration ou le moyen d'acquisition des données dans l'expérimentation antécédente, il est impossible d'en arriver à expliquer cette différence. Néanmoins, on remarque que, dans les deux cas, on détecte facilement un

court-circuit de 100Ω, puisque CDIDDQ (100Ω) >> CDIDDQ (infini) et, qu'avec l'augmentation de la valeur de la résistance, il est de plus en plus difficile de discerner le court-circuit, car CDIDDQ (R) ≈ CDIDDQ (infini). Comme il a été indiqué précédemment, le seuil de détection est fonction du bruit de la mesure. Supposons, par exemple, que ce seuil est fixé à 0.2mA. Dans le premier tableau, on remarque que la résistance de 73.7kΩ n'aurait pas été détectée tandis que la 81.2kΩ l'aurait été. Si on refait la prise de mesure, il est possible que cette dernière résistance soit détectée. Ceci est causé par l'oscillation du courant de consommation dans la carte de développement. La carte de développement Nexys 2 comporte beaucoup de circuits intégrés qui utilisent l'alimentation. En effet, tous ces composants viennent vraisemblablement ajouter au bruit de consommation et, puisque l'on veut tester uniquement notre FPGA, cela est néfaste. Suite à ces observations, il a été convenu d'utiliser une carte comportant moins de composants et générant moins de bruit sur l'alimentation afin d'obtenir de meilleurs résultats quant à la lecture du courant.

2.9 Conclusion

Dans ce chapitre, nous avons présenté les patrons de test CDIDDQ nécessaires à notre expérimentation. Nous avons commencé par expliquer la génération des patrons et nous avons validé leur fonctionnement par le biais de la simulation. Par la suite, nous avons expliqué l'approche utilisée pour l'application expérimentale de CDIDDQ. Nous avons aussi démontré le moyen utilisé pour la mesure du courant de consommation. Finalement, nous avons fait le test avec le même montage qu'Haithem (Haithem 2011). Suite à l'analyse des résultats, nous avons constaté des différences dans les valeurs dont nous pensons qu'elles viennent de la technique pour la mesure de courant. Afin d'obtenir de meilleurs résultats, nous avons conclu que l'utilisation d'une carte générant moins de bruit sur l'alimentation serait requise.

CHAPITRE 3

APPLICATION DES VECTEURS CDIDDQ SUR LA CARTE DE DÉVELOPPEMENT SPARKFUN

3.1 Introduction

Dans ce chapitre, nous allons appliquer les vecteurs de test CDIDDQ sur la carte de développement Sparkfun. L'objectif de cette manipulation est d'obtenir de meilleurs résultats que ceux obtenus avec la carte Nexys II. En effet, la carte Sparkfun comporte beaucoup moins de composants, ce qui devrait vraisemblablement diminuer le bruit sur nos lectures de courant et de ce fait même augmenter notre précision. Dans un premier temps nous allons présenter la carte Sparkfun et la comparer avec la Nexys II. Ensuite, nous allons expliquer le montage qui sera utilisé pour l'application des vecteurs de test CDIDDQ et nous allons présenter les résultats obtenus. Finalement, nous allons appliquer le test CDIDDQ sur différentes alimentations du FPGA et analyser ces résultats.

3.2 La carte de développement Sparkfun

Comme mentionné précédemment, nous pouvons raffiner nos mesures en utilisant une carte comportant moins de composantes ce qui diminuerait le bruit sur l'alimentation. Afin de valider notre hypothèse, nous avons fait l'acquisition d'une carte de développement de chez Sparkfun. Contrairement à la Nexys 2, cette carte présente peu de composantes. En effet, elle ne contient que trois régulateurs de tension, un transmetteur récepteur RS-232, un cristal, une mémoire PROM et, bien sûr, un FPGA. Aussi, il est possible de contourner les régulateurs afin de fournir nous-mêmes les niveaux d'alimentation nécessaires au FPGA. Le FPGA sur cette carte de développement est un Spartan-3E identique à celui de la carte Nexys2. Ainsi, les modifications pour la migration de carte de développement sont minimes.

3.2.1 Test de consommation de la carte Sparkfun

Pour valider notre hypothèse, nous avons alimenté les deux cartes et nous avons comparé leur consommation de courant. Afin de réduire davantage le bruit sur l'alimentation, nous avons décidé de dessouder le cristal et le transmetteur-récepteur présents sur la nouvelle carte. Les figures 3.1 et 3.2 permettent de visionner la lecture du courant sur les deux cartes. Les informations sur le taux d'échantillonnage, le nombre de points et autres apparaissent aussi dans ces images.

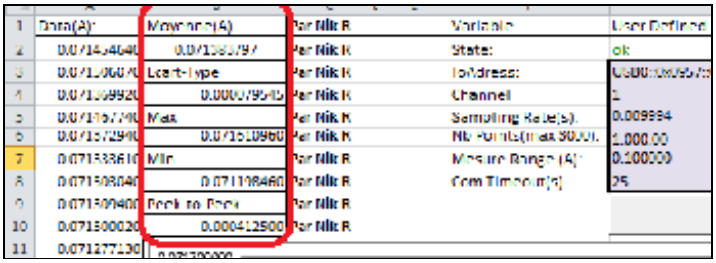

Figure 3.1 Carte Sparkfun

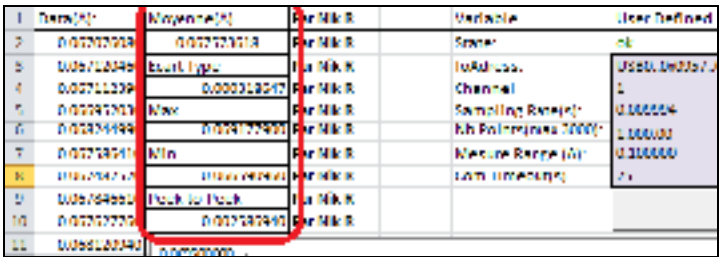

Figure 3.2 Carte Nexys 2

La carte Sparkfun, qui comporte moins de composantes, consomme légèrement plus que la carte Nexys 2. Nous croyons que la différence réside dans les régulateurs utilisés. En effet, la carte Nexys 2 comporte des régulateurs à découpage tandis que la carte Sparkfun comporte des régulateurs linéaires. Cela n'a pas vraiment d'importance, puisque l'objectif final est d'avoir une carte qui génère moins de bruit sur l'alimentation. Ainsi, on remarque que la carte Sparkfun a un écart-type du courant quatre fois plus petit que celui de la carte Nexys 2.

L'écart type étant moindre, il y aura donc un plus petit écart entre nos échantillons, ce qui augmente la précision de notre mesure. Nous pouvons donc conclure que, pour notre circuit à tester, la carte Sparkfun est supérieure à la carte Nexys 2. À partir de maintenant, nous allons utiliser la carte Nexys 2 comme testeur et la carte Sparkfun comme circuit sous-test.

3.2.2 Modification du montage pour la carte Sparkfun

Afin de pouvoir utiliser la carte Sparkfun comme circuit sous-test dans notre montage expérimental, nous avons dû modifier celui-ci. En effet, nous avons procédé au design et à la réalisation du circuit imprimé qui permet la connexion entre les cartes de développement Sparkfun et Nexys 2 puisque l'adaptateur utilisé pour le montage précédent ne le permet pas. L'adaptateur réalisé permet de connecter les 40 broches d'entrée-sortie, utilisées initialement pour la connexion Nexys 2 à Nexys 2, à la carte Sparkfun. De ce fait, l'utilisation des mêmes broches d'entrée-sorties nous permet d'éviter d'apporter des modifications au code du testeur.

3.2.3 Conception de l'adaptateur

La carte Sparkfun comporte 4 rangées de 52 trous, chacune permettant l'accès à une banque de broches d'entrée-sorties du FPGA, à savoir, les banques 0,1,2 et 3. Il aurait été possible de prendre seulement une banque de broches d'entrée-sorties de la carte Sparkfun pour la connecter à l'autre carte, mais l'utilisation de deux connecteurs permet d'espacer les traces, ce qui simplifie le routage du circuit. Il faut choisir correctement les broches sur les banques du FPGA puisqu'elles ne sont pas toutes des broches d'entrée-sortie. Certaines peuvent, par exemple, servir à l'alimentation ou être utilisables seulement en entrée. L'adaptateur permet aussi de relier les réseaux JTAG des deux cartes. Le bus de 4 fils JTAG est un standard IEEE qui permet de relier plusieurs circuits entre eux. Via cette communication série, il est possible de lire ou écrire dans les circuits connectés à ce bus. De ce fait, il est possible d'utiliser ce bus pour programmer les composantes de cette chaîne. La carte Nexys 2 peut être programmée via le connecteur USB ou par le connecteur JTAG, tandis que la carte

Sparkfun ne supporte que le lien JTAG. Puisque chacune des cartes possède un bus JTAG, il est intéressant de pouvoir les lier ensemble. Ainsi, on n'utilise qu'un seul programmeur JTAG qui nous permet de configurer les deux cartes de développement. Il est possible de modifier cette chaîne en détournant les signaux nécessaires sur le connecteur de 40 broches que nous voulons utiliser pour l'adaptateur. Avec ces informations, nous avons produit notre circuit imprimé. On peut voir dans la figure 3.3 le routage de notre circuit imprimé.

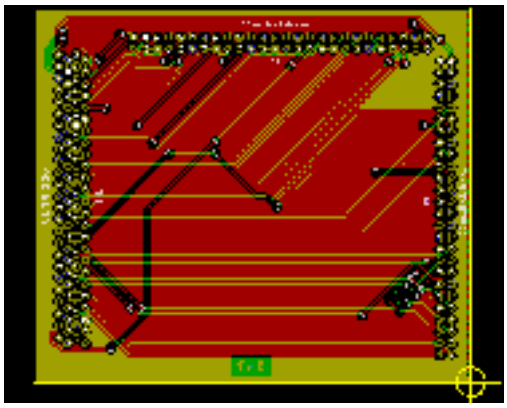

Figure 3.3 Routage de l'adaptateur

Une fois le circuit imprimé de l'adaptateur réalisé, on peut alors connecter les cartes entre elles comme dans la figure 3.4.

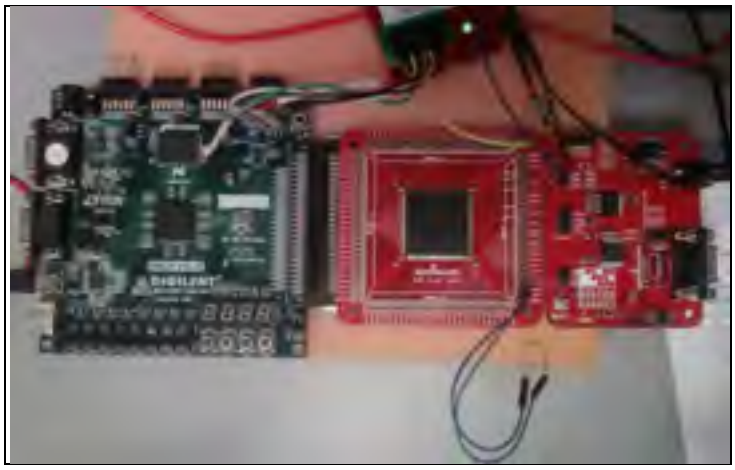

Figure 3.4 Montage avec une carte Nexyx II(testeur)
et une Sparkfun(circuit sous test)

3.2.4 Script d'automatisation de la programmation

Comme programmeur JTAG, nous avons utilisé le DS300 de Xilinx, la même compagnie qui produit notre FPGA. Via son logiciel d'interface sur l'ordinateur, on constate que notre chaîne JTAG fonctionne correctement, car les FPGAs des deux cartes sont détectés comme affiché sur la figure 3.5.

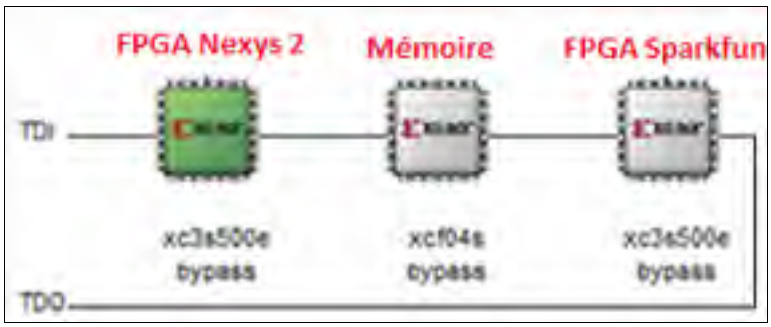

Figure 3.5 Chaîne JTAG du montage

Un ajout important de l'utilisation de ce programmeur JTAG est qu'il est désormais possible d'automatiser la programmation des deux cartes. En effet, lors de la programmation de la carte Nexys 2 via le port USB, on doit absolument utiliser le programme interface fourni par le fabriquant qui ne supporte aucune commande externe. De son côté, le programmeur DS300 peut être utilisé avec son interface ou par des lignes de commandes. L'automatisation est très intéressante dans notre cas, puisque, comme on peut le constater dans l'organigramme du testeur, on doit programmer trois fois la carte pour chaque test CDIDDQ. Nous avons donc mis au point un script qui automatise la programmation des cartes. L'organigramme du script est affiché à la figure 3.6.

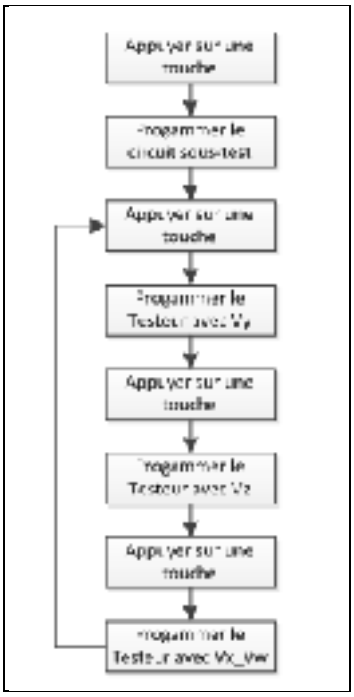

Figure 3.6 Organigramme du script de programmation

L'utilisation de ce script nous permet donc de procéder au calcul de CDIDDQ plus rapidement.

3.2.5 Résultats et discussions

La première série de mesures de courant avec la carte Sparkfun ont été prises sur l'alimentation 5 volts qui fournit les 3 régulateurs de la carte. Les résultats sont affichés aux tableaux 3.1 et 3.2.

Tableau 3.1 Résultat avec la carte Sparkfun

R	**Vx**	**Vw**	**Vy**	**Vz**	**|Vx+Vw-Vy-Vz|**
infini	0.0562437	0.0572152	0.0567152	0.0567188	0.0000250
0.0994k	0.0562577	0.0571130	0.0795809	0.0795004	0.0457106
1k	0.0562029	0.0571437	0.0597167	0.0595810	0.0059511

R	Vx	Vw	Vy	Vz	|Vx+Vw-Vy-Vz|
2k	0.0560431	0.0570759	0.0581916	0.0580941	0.0031667
12.85k	0.0561970	0.0571178	0.0568883	0.0567638	0.0003372
38.6	0.0562336	0.0569460	0.0566763	0.0565545	0.0000512
73.7	0.0561815	0.0570563	0.0565120	0.0565000	0.0002258
81.2k	0.0560006	0.0571320	0.0564856	0.0564739	0.0001730
90.8k	0.0559561	0.0569146	0.0564800	0.0564942	0.0001036
110k	0.0560895	0.0575500	0.0565824	0.0565669	0.0004902
300k	0.0561111	0.0574320	0.0569687	0.0565567	0.0000177
510k	0.0561551	0.0575235	0.0566150	0.0566232	0.0004403

Tableau 3.2 Rappel des résultats avec la carte Nexys 2

R	Vx	Vw	Vy	Vz	|Vx+Vw-Vy-Vz|
infini	0.0743912	0.0743145	0.0748128	0.0737849	0.0001079
0.0994k	0.0859598	0.0863086	0.0747247	0.0738200	0.0237237
1k	0.0770793	0.0771775	0.0748574	0.0739862	0.0054131
2k	0.0758599	0.0757118	0.0748863	0.0740232	0.0026621
12.85k	0.0745436	0.0745609	0.0748245	0.0739187	0.0003613
38.6k	0.0744933	0.0744434	0.0745222	0.0736341	0.0007803
73.7k	0.0747304	0.0742246	0.0747457	0.0742272	0.0000178
81.2k	0.0741744	0.0744231	0.0749062	0.0734554	0.0002358
90.8k	0.0743299	0.0743481	0.074756	0.0739472	0.0001447

Pour les mesures servant au calcul CDIDDQ, nous avons échantillonné 100 points à une vitesse d'échantillonnage de 10ms sur une plage de 100mA. Comme mentionné précédemment, la mesure CDIDDQ sans défectuosité (R=infini) devrait donner près de 0, s'il n'y a pas de défectuosité dans le circuit. Comme on peut le constater, la mesure CDIDDQ de la carte Sparkfun est 4 fois plus petite que la carte Nexys 2, et, donc, plus près de zéro. Cela vient une fois de plus confirmer que nous aurons une mesure plus précise de la

consommation du courant en réduisant le nombre de composantes sur la carte testée. Afin d'obtenir des résultats plus précis, il serait possible de réduire davantage les composantes sur la carte. En effet, on peut retirer les régulateurs sur la carte si on fournit directement les alimentations au FPGA.

3.3 Exploration et utilisation des différentes alimentations du FPGA

Comme mentionné précédemment, il est possible, avec la carte Sparkfun, de contourner les régulateurs afin que l'on fournisse nous-mêmes les alimentations (3.3volts, 2.5 volts, 1.2 volts) au FPGA. Ainsi, la consommation de courant mesurée viendra uniquement du FPGA.
La première étape est de discerner, laquelle des tensions est la plus propice au test CDIDDQ. Pour ce faire, nous avons exécuté le test CDIDDQ sans et avec une résistance et nous avons comparé les résultats. Nous avons choisi une résistance basse (100ohms), afin de pouvoir facilement visualiser l'effet. Les résultats sont affichés dans le tableau 3.3.

Tableau 3.3 CDIDDQ (en ampère) pour les 3 alimentations

R	**3.3v**	**2.5v**	**1.2v**
infini	0.0000836	0.0000053	0.0000506
100	0.0468528	0.0000892	0.0000429

Comme on peut le constater, la plus grande variation apparaît sur le 3.3 volts. Ce résultat est facile à anticiper connaissant la nature des diverses alimentations du FPGA. L'alimentation 1.2 volts, appelée VCCint, est utilisée pour alimenter toute la circuiterie interne servant aux fonctions logiques. Pour sa part, l'alimentation de 2.5 volts, appelée VCCaux, est l'alimentation auxiliaire utilisée pour différents blocs du FPGA comme l'interface JTAG, les broches de configuration et autres. Finalement le 3.3 volts alimente les broches d'entrée-sortie du FPGA. Puisque la défectuosité est connectée sur les broches d'entrée-sortie du FPGA, la variation de courant sur le 3.3 volts sera élevée. En mesurant uniquement le courant sur l'alimentation 3.3volts, on néglige alors la consommation interne du circuit, élément dont il faudra tenir compte lors du choix du seuil de décision appliqué à CDIDDQ,

permettant d'obtenir l'équilibre désiré entre les pertes de rendement (dues aux puces fonctionnelles faussement déclarées défectueuses) et le taux de puces défectueuses non détectées. Le choix de ce seuil sera discuté plus en détail au chapitre suivant. On remarque par ailleurs que la valeur du courant sur l'alimentation 1.2 volts varie légèrement, mais ne semble en aucun cas être liée à la consommation introduite par la défectuosité ajoutée. Autrement dit, la variation semble plutôt être l'effet du bruit dans la mesure du courant. C'est spécifiquement ce bruit qui devra être considéré lors du choix du seuil. À partir de ce point, nous utiliserons la lecture sur le 3.3 volts pour nos tests puisqu'elle reflète plus précisément le test appliqué et qu'elle nous permet d'éviter l'utilisation des régulateurs sur la carte.

3.3.1 Mesure sur le 3.3 volts

Afin de valider que nous ayons une mesure plus précise, nous avons commencé par vérifier l'écart-type entre la mesure du courant de la carte Sparkfun programmées avec le code du circuit sous-test sur l'alimentation 5 volts (avec régulateurs) et 3.3volts (sans régulateurs).Les résultats sont affichés aux figures 3.7 et 3.8.

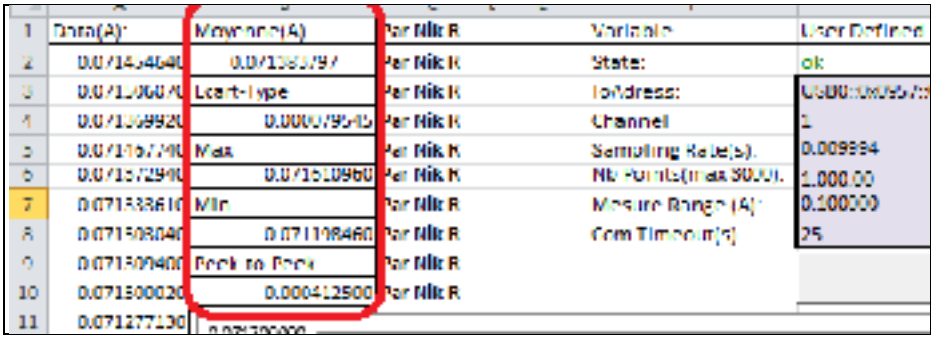

Figure 3.7 Carte Sparkfun 5 volts, avec régulateurs

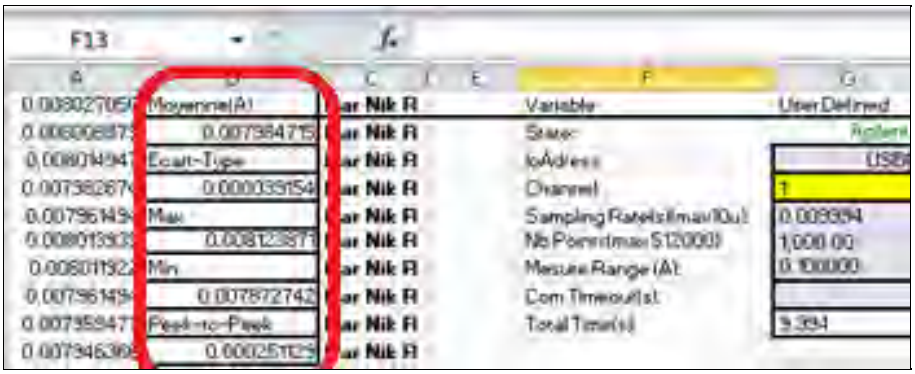

Figure 3.8 Carte Sparkfun 3.3 volts, sans régulateur

Comme on peut le remarquer, l'écart-type est réduit de moitié. De ce fait, nous obtiendrons des mesures plus précises et, donc, une meilleure capacité à détecter les défectuosités hautement résistives.

3.3.2 Analyse des résultats

Nous avons donc procédé au même test qu'auparavant mais sur le 3.3 volts. Toutefois, afin de diminuer l'impact de la variation de courant, nous avons décidé que nous allions prendre le taux d'échantillonnage le plus rapide possible, soit 10us pour l'Agilent. De ce fait, nous diminuons le temps entre les échantillons. Nous allons garder ce taux d'échantillonnage pour toutes les autres mesures qui suivront. Aussi, nous avons décidé de réduire le nombre de résistances testées de presque de moitié afin de réduire le temps nécessaire au test. Puisque la valeur exacte des résistances n'est pas ce qui nous préoccupe, nous allons donc répertorier les résistances par leurs valeurs théoriques. Les résultats obtenus ainsi que ceux précédemment obtenus sont affichés aux tableaux 3.4 et 3.5.

Tableau 3.4 Résultat obtenu avec la carte Sparkfun sur 3.3 volts, sans régulateur

R	**Vx**	**Vw**	**Vy**	**Vz**	**|Vx+Vw-Vy-Vz|**
infini	0.0075196	0.0084522	0.0079830	0.0079847	0.0000041
0.1k	0.0075260	0.0084436	0.0313751	0.0313803	0.0467857

R	Vx	Vw	Vy	Vz	|Vx+Vw-Vy-Vz|
1k	0.0075277	0.0084413	0.0111319	0.0111338	0.0062966
13k	0.0075263	0.0084446	0.0082400	0.0082425	0.0005116
39k	0.0075217	0.0084466	0.0080656	0.0080753	0.0001726
75k	0.0075228	0.0084503	0.0080248	0.0080297	0.0000813
110k	0.0075253	0.0084485	0.0080112	0.0080262	0.0000636
300k	0.0075229	0.0084434	0.0079438	0.0079989	0.0000269
510k	0.0075206	0.0084429	0.0079878	0.0079895	0.0000137

Tableau 3.5 Rappel des résultats obtenus avec la carte Sparkfun sur le 5 volts, avec régulateur

R	Vx	Vw	Vy	Vz	|Vx+Vw-Vy-Vz|
infini	0.0562437	0.0572152	0.0567152	0.0567188	0.0000250
0.0994k	0.0562577	0.0571130	0.0795809	0.0795004	0.0457106
1k	0.0562029	0.0571437	0.0597167	0.0595810	0.0059511
2k	0.0560431	0.0570759	0.0581916	0.0580941	0.0031667
12.85k	0.0561970	0.0571178	0.0568883	0.0567638	0.0003372
38.6	0.0562336	0.0569460	0.0566763	0.0565545	0.0000512
73.7	0.0561815	0.0570563	0.0565120	0.0565000	0.0002258
81.2k	0.0560006	0.0571320	0.0564856	0.0564739	0.0001730
90.8k	0.0559561	0.0569146	0.0564800	0.0564942	0.0001036
110k	0.0560895	0.0575500	0.0565824	0.0565669	0.0004902
300k	0.0561111	0.0574320	0.0569687	0.0565567	0.0000177
510k	0.0561551	0.0575235	0.0566150	0.0566232	0.0004403

Cette fois-ci, toutes les valeurs mesurées sont supérieures à la valeur R=infini. Le niveau de bruit ayant descendu nous obtenons une moins grande oscillation du courant de consommation et donc une meilleure capacité à détecter de petites variations de courant. Il faut noter cependant que le fait qu'une valeur obtenue avec une défectuosité soit supérieure à

celle obtenue sans défectuosité ne signifie pas nécessairement qu'on puisse détecter cette même défectuosité. En effet, comme souligné précédemment, nous devons considérer l'ensemble du bruit de mesure et choisir la valeur du seuil de décision en conséquence. Une autre manière de comparer les résultats obtenus avec et sans les régulateurs est de tracer un graphique des valeurs CDIDDQ en fonction de la valeur de la résistance émulant la défectuosité. Veuillez prendre note que les valeurs CDIDDQ dans les graphiques ci-dessous sont affichées sur une échelle logarithmique. Aussi, seules les neuf valeurs de résistance appartenant aux 2 tests sont affichées dans les figures 3.9 et 3.10.

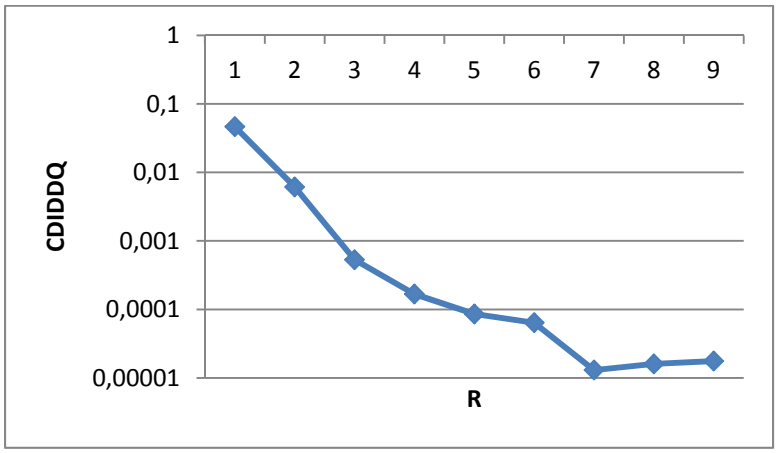

Figure 3.9 CDIDDQ 3.3 volts, sans régulateur

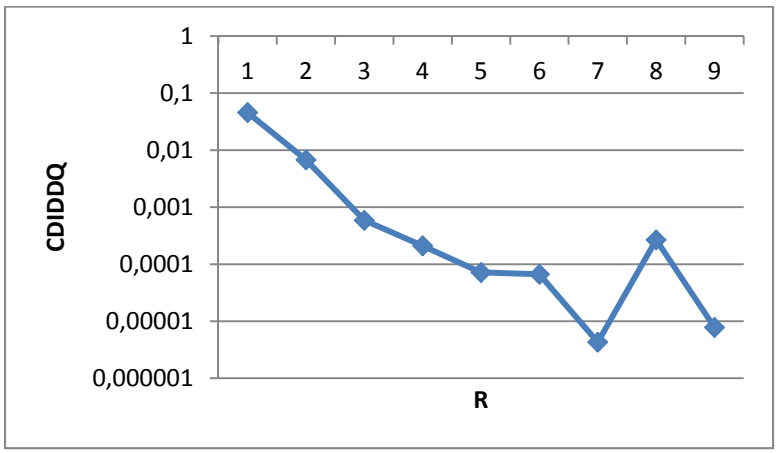

Figure 3.10 CDIDDQ 5 volts, avec régulateurs

Le graphique représentant les valeurs CDIDDQ sur le 3.3 volts décroît avec l'augmentation des valeurs ohmiques de la défectuosité insérée. C'est bien naturellement ce qu'on s'attend à avoir comme résultat, puisque la résistance tend vers l'infini, et, donc, CDIDDQ devrait tendre vers CDIDDQ(infini). On remarque toutefois que, dans l'autre graphique, les valeurs ne décroissent pas comme on se l'attendait. Ce phénomène est causé par le bruit sur la lecture qui se répercute sur notre calcul CDIDDQ.

3.4 Conclusion

À cette étape, nous sommes en mesure de détecter via CDIDDQ toutes les défectuosités que nous nous sommes fixées. De plus, les valeurs CDIDDQ coïncident avec les valeurs que nous nous attendons à obtenir, puisqu'elles décroissent avec l'augmentation de la résistivité de la défectuosité. Nous sommes satisfaits des résultats obtenus et nous allons maintenant nous intéresser à la détection des mêmes défectuosités, mais cette fois-ci sur un montage qui consomme plus de courant. Le défi sera donc de détecter les mêmes petites variations à travers un plus gros courant.

CHAPITRE 4

APPLICATION DES VECTEURS CDIDDQ SUR DES MONTAGES AVEC SURCONSOMMATION DE COURANT

4.1 Introduction

Comme mentionné précédemment, nous allons effectuer les mêmes tests CDIDDQ que ceux du chapitre précédent, en ajoutant au montage des éléments permettant d'augmenter artificiellement la consommation de courant. Afin d'émuler cette consommation supplémentaire de courant, nous ajoutons en parallèle un circuit purement résistif. Nous allons effectuer les tests CDIDDQ sur des montages consommant 200mA, 1A et 3A, cette dernière valeur étant le maximum que la source peut fournir pour un canal. Jusqu'ici, nous avons utilisé la plage de mesure de 100mA. Cette plage est évidemment insuffisante pour les niveaux de courant visés. La prochaine plage de mesure disponible sur l'appareil pour la mesure de courant excédant 100mA est de 3.06A. Si nous augmentons l'échelle de mesure, nous augmentons aussi l'erreur moyenne, ce qui entraîne la diminution de la précision de nos mesures. Afin de pouvoir garder la même échelle de mesure, soit 100mA, nous allons ajouter à notre montage actuel une source de courant. Cette source fournira le courant nécessaire afin de s'assurer que la source de tension, que nous utilisons pour faire la lecture du courant consommé, ne fournisse pas plus que son échelle de mesure, soit 100mA. Notons que l'ajout de cette source de courant n'influence pas nos calculs de CDIDDQ, puisque les niveaux DC sont naturellement filtrés par ces calculs. Deux types de source de courant sont considérés : une première source basée sur le circuit LT3080 et une seconde fournie par la source d'alimentation N6705A.

4.2 Source de courant basée sur le LT3080

La figure 4.1 affiche le circuit de la première source de courant utilisée lors de nos tests.

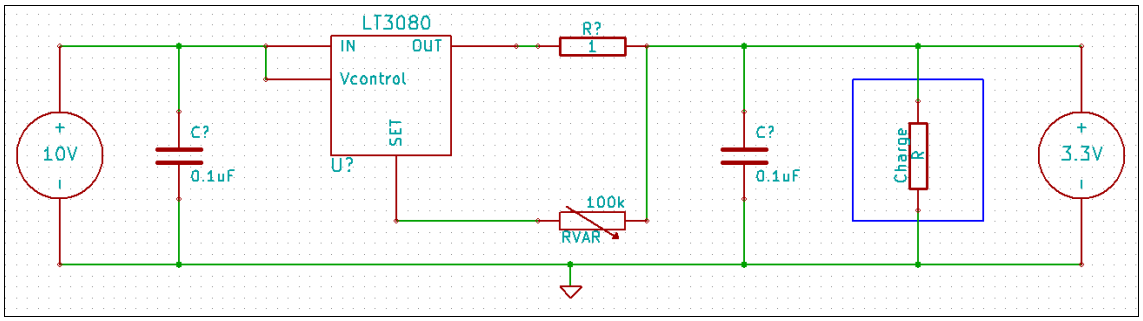

Figure 4.1 Montage avec source courant

Comme on peut le constater, on utilise désormais 2 canaux sur notre source d'alimentation du N6705A pour notre montage. Nous utilisons une source de tension du N6705A pour alimenter notre source de courant externe construite sur un platine d'expérimentation et nous utilisons une autre source de tension du N6705A qui fournit le 3.3 volts ainsi que le reste du courant non fourni par notre source de courant externe.

4.3 Le LT3080

Le cœur de la source de courant est le LT3080 qui est constitué d'une source courant et d'un ampli op suivi d'un transistor NPN, comme affiché à la figure 4.2.

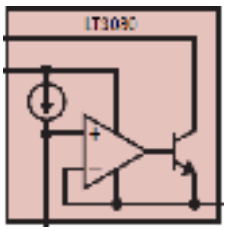

Figure 4.2 Source de courant avec le LT3080
Tirée de LT3080 – Adjustable 1.1A Single Resistor Low Dropout Regulator (2007, p.1)

Il est possible d'utiliser ce circuit intégré pour différentes applications, mais nous nous en tiendrons au montage de la source de courant. Cette source peut fournir jusqu'à 1A. Afin de pouvoir fournir plus de courant et/ou de dissiper la température de la puce, il est possible de la paralléliser, comme illustré dans la figure 4.3. Ainsi, il sera possible de construire une

source qui fournira jusqu'à 3A. Finalement, le LT3080 peut être acheté dans le package TO-220 qui nous permet de l'utiliser directement sur la platine d'expérimentation.

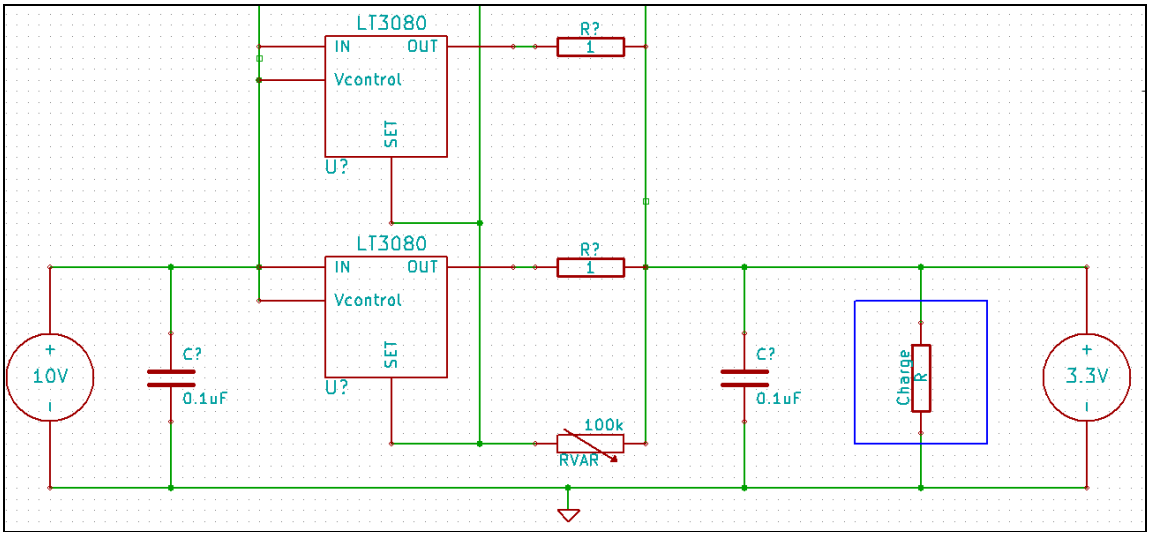

Figure 4.3 Parallélisassions de la source de courant

4.4 Ajustement de la source de courant

Pour ajuster la source, on doit mettre seulement la source de tension 3.3 volts sur le circuit pour mesurer le courant de consommation du circuit. Ensuite, on éteint la source de tension 3.3 volts, on place le potentiomètre à 0k et on allume la source de 10V. On ajuste le potentiomètre pour avoir légèrement moins (3 à 5mA) que le courant de consommation précédemment mesuré. Si l'on place la source de courant trop près du courant de consommation, l'oscillation du courant peut créer un courant négatif sur la source 3.3 volts. Une fois la source de courant ajustée, on peut alors allumer la source de tension de 3.3 volts.

4.5 Ajustement de la source de courant

On rappelle que l'ajout de la source courant nous permet de garder l'échelle de mesure de 100mA sur notre analyseur de puissance. Afin de créer une surconsommation de courant, nous avons utilisé des résistances de 8.2 ohms. Ainsi, dans notre montage nous avons comme

charge le FPGA et un agencement de résistances afin d'obtenir la consommation désirée. Les agencements utilisés sont affichés dans le tableau 4.1.

Tableau 4.1 Agencements des résistances

Courant à 3.3 volts	**Résistances utilisées**
200mA	2 résistances 8.2 Ω en série = 16.4 Ω
1A	5 groupes parallèles de 2 résistances 8.2 Ω en série = 3.28 Ω
3A	7 résistances 8.2 Ω en parallèle = 1.17 Ω

La figure 4.4 affiche le montage utilisé pour une surconsommation de 1A.

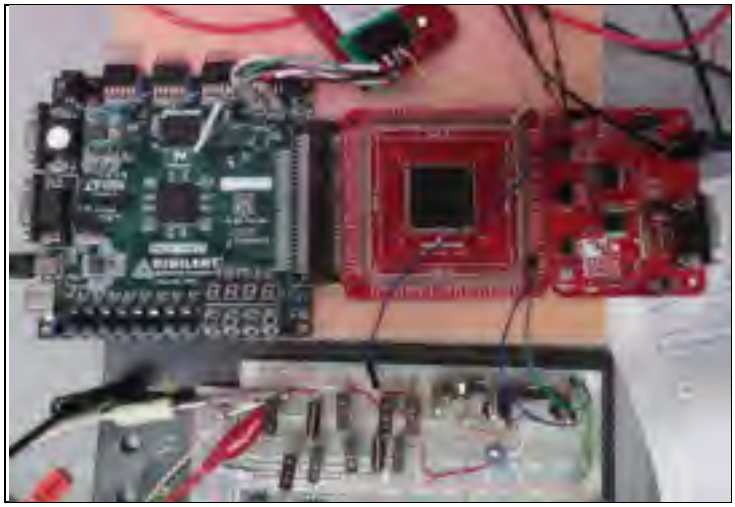

Figure 4.4 Montage

On peut observer sur cette image le testeur CDIDDQ (carte verte Nexys II), le circuit sous test (carte rouge Sparkfun), la charge résistive supplémentaire (sur la partie gauche de la platine d'expérimentation) et la source de courant (sur la partie droite de la platine d'expérimentation).

4.6 Résultats et analyse

Pour tester l'application des vecteurs de test CDIDDQ, nous avons choisi d'utiliser le même circuit sous-test ainsi que les mêmes vecteurs utilisés précédemment. Les résultats sont affichés aux tableaux 4.1, 4.2 et 4.3.

Tableau 4.1 Montage avec surconsommation de 200mA

R	**Vx**	**Vw**	**Vy**	**Vz**	**|Vx+Vw-Vy-Vz|**
infini	0.0170919	0.180038	0.0175554	0.0175579	0.0000176
0.1k	0.0171241	0.0180393	0.0408346	0.0408432	0.0465144
1k	0.0166940	0.0175973	0.0202107	0.0202170	0.0061364
13k	0.0169669	0.0178964	0.0176893	0.0177047	0.0005307
39k	0.0169985	0.0179077	0.0175308	0.0175430	0.0001673
75k	0.017076	0.0179183	0.0175050	0.0175066	0.0000857
110k	0.0169984	0.0179138	0.0174852	0.0174910	0.0000640
300k	0.0170171	0.0179231	0.0174745	0.0174785	0.0000130
510k	0.0170029	0.0179208	0.0174693	0.0174704	0.0000160

Tableau 4.2 Montage avec surconsommation de 1A

R	**Vx**	**Vw**	**Vy**	**Vz**	**|Vx+Vw-Vy-Vz|**
infini	0.0339984	0.0348662	0.0344307	0.0344418	0.0000078
0.1k	0.0337728	0.0347022	0.0571032	0.0571019	0.0457301
1k	0.0338763	0.0349199	0.0377817	0.0377810	0.0067664
13k	0.0340244	0.0349235	0.0347312	0.0348043	0.0005875
39k	0.0340034	0.0349086	0.0345276	0.0345942	0.0002097
75k	0.0333583	0.0342759	0.0338634	0.0338425	0.0000716
110k	0.0340232	0.0348899	0.0344755	0.0345041	0.0000665
300k	0.0336434	0.0345562	0.0341384	0.0341384	0.0000043

R	Vx	Vw	Vy	Vz	|Vx+Vw-Vy-Vz|
510k	0.0331219	0.0341086	0.0337377	0.0337601	0.0002672

Tableau 4.3 Montage avec surconsommation de 3A

R	Vx	Vw	Vy	Vz	|Vx+Vw-Vy-Vz|
infini	0.0325169	0.0330589	0.0321963	0.0332143	0.0001652
0.1k	0.0050387	0.0328066	0.0540367	0.0544698	0.0706612
1k	0.0277718	0.0292049	0.0331739	0.0333152	0.0095124
13k	0.0262608	0.0298200	0.0289132	0.0293796	0.0022119
39k	0.0260250	0.0274845	0.0271169	0.0295451	0.0031525
75k	0.0302125	0.0318650	0.0314849	0.0314103	0.0008176
110k	0.0324545	0.0342038	0.0342016	0.0345344	0.0020777
300k	0.0339317	0.0356020	0.0354279	0.0356184	0.0015126
510k	0.0376354	0.0403018	0.0375838	0.0358280	0.0045254

Si l'on porte une attention particulière aux colonnes des vecteurs (Vz,Vy,Vx et Vw), on remarque que le courant de consommation est différent d'un montage à l'autre. La différence entre ceux-ci vient de l'ajustement de la source qui diffère d'un montage à l'autre. Puisque nous soustrayons les vecteurs, nous retirons de ce fait la valeur DC causée par l'ajustement de la source de courant. En d'autres mots, on peut ajuster la source courant comme on le désire pourvu que la source de tension n'ait pas à fournir plus de 100mA.

Comme seuil de détection, nous avons choisi de considérer comme défectueux toute valeur qui est 3 fois au-dessus de l'écart-type de CDIDDQ(infini). Nous avons choisi ce seuil puisqu'il correspond à plus 95% des valeurs sur une distribution normale. Ainsi, le seuil est de 0.000159A pour le montage à 200mA, de 0.000155A pour le montage à 1A et de 0.000489A pour le montage à 3A. Avec ces seuils, nous parvenons à détecter dans tous les cas le court-circuit causé par la résistance 39k. Malheureusement, nos calculs de CDIDDQ

oscillent lorsque la résistance augmente. Les graphiques 4.8, 4.10, 4.11 affichent les CDIDDQs pour chacun des montages.

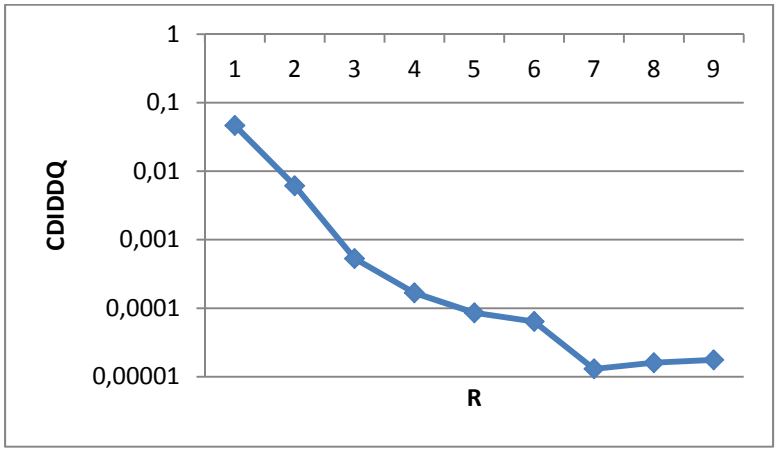

Figure 4.5 Graphique CDIDDQ à 200mA

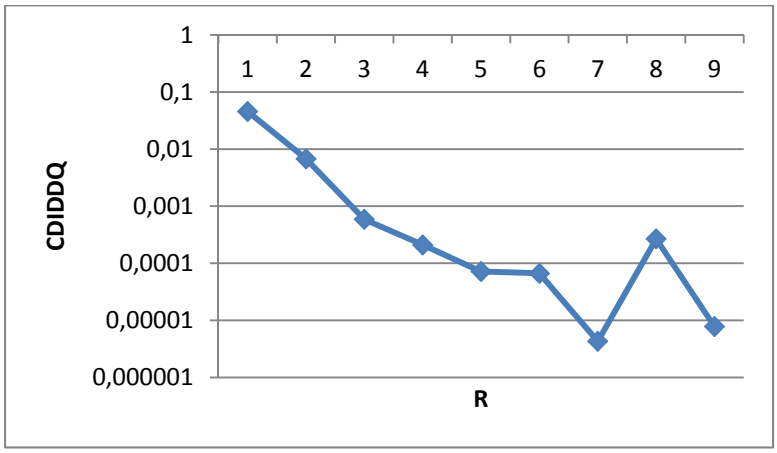

Figure 4.6 Graphique CDIDDQ à 1A

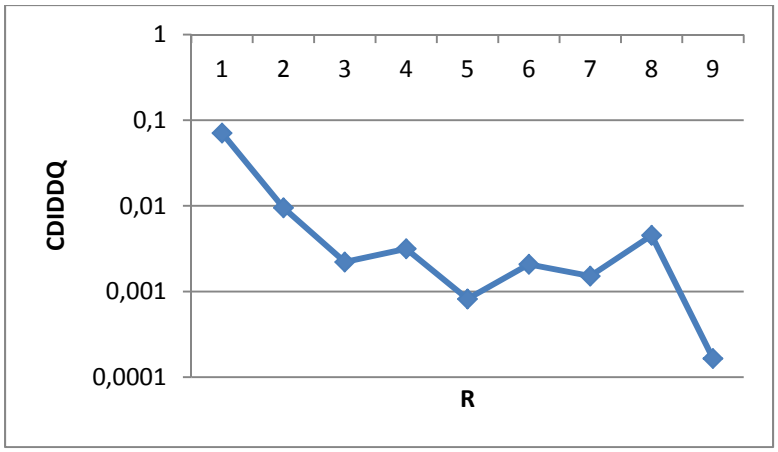

Figure 4.7 Graphique CDIDDQ à 3A

Contrairement à la courbe obtenue par les mesures sur le 3.3 volts sans surconsommation, on observe ici une nette oscillation sur nos valeurs. De plus, il y a vraisemblablement une relation entre le courant consommé et l'oscillation puisque CDIDDQ sur le montage a 3A oscille plus que sur le montage à 200mA. Ce problème vient forcément des deux nouveaux éléments ajoutés soit les résistances ainsi que la source de courant.

Nous allons débuter notre investigation par les résistances. La valeur ohmique d'une résistance est définie à une température donnée. Ainsi, une variation de la température influence la valeur de la résistance comme représenté par l'équation (Kasap, S. O. 2006) ci-dessous :

$$R = R_0[1 + \alpha(T - T_0)]. \tag{4.1}$$

Dans cette équation, R0 est la valeur de la résistance à T_0, α le coefficient de température de la résistance et T la température actuelle. Comme on peut le constater, plus la température augmente, plus la valeur de la résistance augmente. Dans notre cas, la variation des valeurs ohmiques des résistances peut avoir lieu puisque l'obtention des mesures de courant des quatre vecteurs nécessaires au calcul de CDIDDQ pour une résistance choisie prend quelques minutes. Pendant ce temps, la température des résistances peut augmenter de quelques degrés Celsius pour les montages avec une consommation de courant élevée. Afin d'éviter la

variation des valeurs ohmiques des résistances due au réchauffement de celles-ci, nous avons, pour les autres mesures subséquentes, attendu quelques minutes avant de prendre nos mesures afin que la température des résistances soit stable.

En plus d'affecter la valeur ohmique de la résistance, la température a aussi comme effet d'augmenter le niveau de bruit de celle-ci. Le bruit thermique d'une résistance est régi par l'équation (Nyquist H. 1928) suivante :

$$\overline{v_b^2} = 4k_B TR\Delta f \tag{4.2}$$

Dans cette équation, $\overline{v_b^2}$ représente la variance de la tension aux bornes de la résistance, k_B la constante de Boltzmann, T la température, R la résistance et Δf la bande passante. Ainsi, plus le courant est élevé, plus la résistance chauffe et plus le bruit augmente. Comme écrit au tableau 4.1, pour le montage de 3A, nous avons utilisé 7 résistances de 8.2 ohms. Il est aussi possible d'utiliser 15 nœuds parallèles de 2 résistances en série de 8.2 ohms pour obtenir la même consommation de courant. En modifiant le circuit ainsi, le nombre de watts par résistance diminue d'un facteur de 4, ce qui a pour effet de diminuer la température et du coup le niveau de bruit thermique. Afin de valider notre hypothèse, nous avons branché uniquement les deux circuits résistifs sur l'analyseur de puissance et nous avons effectué une lecture du courant. Les résultats sont affichés dans la figure 4.12 et 4.13.

Moyenne(A)
2.516407367
Ecart-Type
0.007987012
Max
2.530026000
Min
2.494483000
Peek-to-Peek
0.035543000

Figure 4.8 Mesure du courant, 7 résistances parallèles de 8.2 ohms

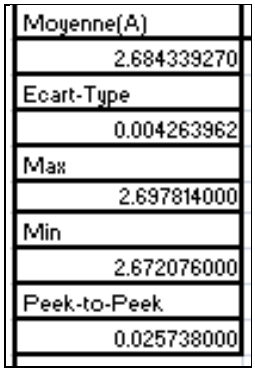

Figure 4.9 Mesure du courant, 15 nœuds parallèles de 16.4 ohms

Effectivement, en diminuant le nombre de watts appliqué aux résistances, et du même coup leur température, on arrive ici à réduire l'écart-type de presque de moitié (0.0079 contre 0.0042). De plus, on remarque la différence de consommation entre les deux montages causée par la température des résistances (2.51 contre 2.68). Ceci confirme que la température change effectivement le courant de consommation et augmente le niveau de bruit. Le seul réel inconvénient de changer notre montage ainsi est qu'on nécessite désormais 30 résistances au lieu de 7. Pour les tests futurs, nous allons utiliser le montage à 30 résistances.

Pour la source de courant, lors des manipulations, la valeur du courant fourni par la source courant oscillait. Il est certain qu'il y a une partie de cette oscillation qui provient du LT3080 lui-même, mais un autre phénomène observable lors des manipulations semble affecter l'oscillation du courant : la platine d'expérimentation. Une très légère pression sur la carte crée des changements de la source de quelque mA. La source oscille donc pendant quelques instants avant de redevenir stable. Les contacts de la platine d'expérimentation ne semblent pas adéquats pour notre application. Pour ces raisons, nous avons exploré la possibilité d'une autre source, celle de notre analyseur de puissance.

4.7 Source de courant N6705A

La deuxième source de courant considérée provient de la source d'alimentation N6705A. Cette idée avait déjà été envisagée dans le passé puisque le N6705A d'Agilent peut configurer ces canaux, soit en source de tension ou en source courant. Malheureusement, l'application de la configuration des sources, comme affichée dans le tableau 4.2, ne donne pas le résultat espéré.

Tableau 4.2 Configuration initiale de l'Agilent

Numéros de canal	**1**	**2**
Type de Source	Voltage	Courant
Voltage	3.3 volts	3.3 volts (max)
Courant	100mA (max)	Selon le montage

En connectant les sources ensemble sur un circuit purement résistif, on constate que chacune d'elle tente d'imposer un niveau. Après quelques minutes, les deux sources finissent par donner chacune la moitié du courant requis par la charge. Face à ce résultat, nous avions choisi de rejeter l'utilisation de la source courant du N6705A. Toutefois, après avoir communiqué avec Agilent, nous avons appris que pour parvenir à connecter les deux sources ensemble nous devions affecter une plus grande tension à la source de courant qu'à la source de tension. De cette façon, la source de courant est dominante face à la source de tension et impose son courant à celle-ci qui ne fournit alors que le courant restant, ce qui est le cas souhaité ici. Nous avons donc choisi les paramètres comme affichés au tableau 4.3.

Tableau 4.3 Configuration finale de l'Agilent

Numéros de canal	**1**	**2**
Type de Source	Voltage	Courant
Voltage	3.3 volts	10v volts(max)
Courant	100mA (max)	Selon le montage

Nous avons reconnecté les deux sources ensemble sur un circuit purement résistif et nous avons mesuré le courant fourni par la source voltage (canal 1). Le courant a été mesuré à toutes les secondes pendant 7 minutes sur l'échelle de 100mA. La figure 4.14 affiche le résultat du courant fourni par la source courant en ampère par rapport au temps en secondes.

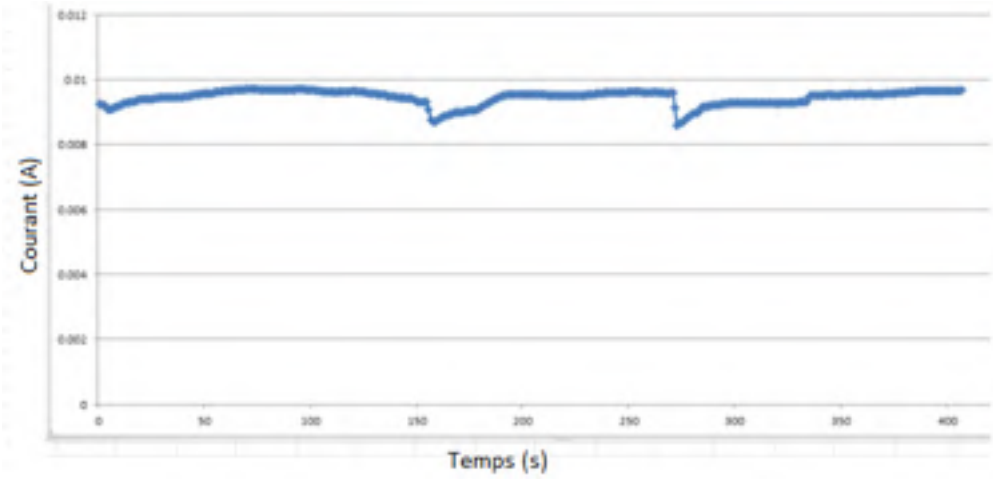

Figure 4.10 Interaction des deux sources

Avec la nouvelle configuration, l'interaction entre les deux sources est moindre. La source courant arrive à fournir sensiblement toujours le même courant, bien qui y ait de légères chutes (1mA) qui se répètent à un intervalle de quelques minutes. Ces chutes peuvent affecter nos calculs de CDIDDQ, puisque l'acquisition des valeurs de courant prend quelques minutes. Nous allons tout de même utiliser l'Agilent N6709 pour l'alimentation de notre montage puisque celui-ci est plus simple, plus stable et moins sensible.

Il faut tout de même être prudent, car, avec cette configuration, on permet à la source courant de fournir jusqu'à 10 volts au circuit afin d'obtenir le courant demandé. Puisque notre FPGA ne peut supporter plus de 3.75 volts, il faut porter une attention particulière, afin de ne pas endommager celui-ci en injectant un voltage supérieur à celui supporté. Afin de diminuer les risques d'erreur, nous avons suivi les étapes affichées à la figure 4.15.

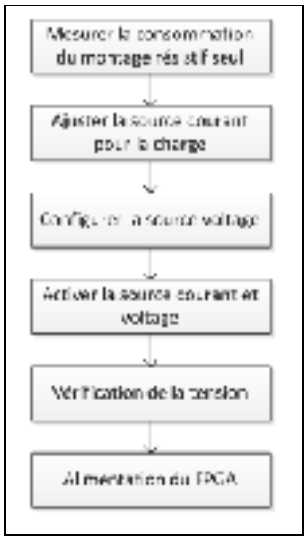

Figure 4.11 Étapes de la configuration des deux sources

La première étape est de mesurer la consommation du circuit résistif à une tension de 3.3 volts. Le FPGA ne consomme pas beaucoup de courant comparativement au circuit résistif ajouté et, donc, on veut savoir combien ce circuit nécessite de courant afin d'ajuster la source courant en conséquence. Dans un monde parfait, on voudrait que notre source de courant fournisse le courant consommé par le circuit résistif et que la source de tension fournisse le courant consommé par le FPGA. Par la suite, une fois la source courant ajustée, on configure notre source de tension comme dans le tableau 4.3 et on connecte les deux sources au circuit résistif avant de les activer. Ensuite, on mesure, avec un multimètre, la tension afin de vérifier que nous avons bien 3.3 volts comme alimentation. Une fois cette vérification faite, on fournit l'alimentation de 1.2 volts et 2.5 volts au FPGA en plus de connecter l'alimentation 3.3 volts sur celle composée des deux sources utilisées par le circuit résistif.

4.8 Conclusion

Dans ce chapitre, nous avons testé CDIDDQ sur des montages plus énergivores. Afin de garder la même échelle de mesure qu'auparavant (100mA), nous avons utilisé une source de courant en parallèle afin que la source de tension utilisée pour la mesure n'ait pas à fournir plus que 100mA. Nous avons constaté une grande oscillation dans les valeurs de CDIDDQ,

puisque les mesures de courant sont influencées par le bruit. Comme nous avons pu le remarquer, une partie du bruit est causée par la source de courant et l'autre par les résistances utilisées. Nous avons trouvé des solutions afin de réduire le bruit de ces deux sources. Toutefois, un problème subsiste, le temps pour l'acquisition des valeurs de courant nécessaire au calcul CDIDDQ prend quelques minutes, ce qui rend notre système sensible aux variations de courant entre les vecteurs. Dans le prochain chapitre, nous allons modifier le testeur afin que celui-ci soit plus rapide.

CHAPITRE 5

MONTAGE FINAL

5.1 Introduction

Dans le chapitre précédent, nous avons repéré certaines sources de problèmes qui ont pu nuire à nos mesures. Les deux changements que nous désirons apporter sont les changements de la source courant sur la platine d'expérimentation pour la source courant du N6709, ainsi que l'utilisation de 30 résistances au lieu de 7 pour le circuit avec une consommation de 3A afin de réduire le bruit thermique. Jusqu'ici, nous avons seulement exploité le patron 17 dans nos tests CDIDDQ et il serait intéressant de pouvoir tester les trois autres patrons déjà présentés, soit les patrons 5, 6 et 29. Puisque l'architecture du testeur nécessite l'utilisation de trois fichiers de programmation pour un patron de test, il nécessiterait douze fichiers de programmation pour les 4 patrons. Le testeur développé n'est pas assez versatile puisqu'il a été conçu pour envoyer un seul vecteur à la fois. Un autre point négatif du testeur présentement utilisé est que l'obtention des vecteurs CDIDDQ est lente.

Comme mentionné précédemment, pour chaque vecteur on doit reprogrammer le circuit et, même avec le script mis au point, cela s'avère assez long (quelques minutes). En plus de la reprogrammation, on doit manuellement changer les interrupteurs afin que le testeur envoie les vecteurs, ce qui rajoute du temps et augmente la possibilité d'erreur due à une mauvaise combinaison appliquer aux interrupteurs. À titre indicatif, un tableau de valeurs CDIDDQ pour chaque type de montage prend environ 1h à construire. Nous voudrions automatiser le processus afin de simplifier les manipulations et aussi de gagner du temps. Pour ces raisons, nous avons mis au point un nouveau testeur CDIDDQ plus flexible et rapide. Nous avons décidé de garder exactement la même carte de développement pour notre nouveau testeur. Il aurait été possible d'utiliser une autre carte ou même une approche totalement différente, comme l'utilisation d'une carte IO sur l'ordinateur, par exemple. Puisque le montage déjà en place répond à nos besoins nous avons décidé de conserver la carte de développement Nexys2 comme testeur. Dans notre nouveau montage, nous utiliserons un ordinateur qui

communiquera en série avec la plaquette de développement FPGA qui agira comme testeur. Celle-ci se chargera d'envoyer les vecteurs de test au circuit à tester. Une fois les vecteurs envoyés, une mesure de courant sera prise sur la carte par l'analyseur de puissance qui alimente le circuit sous-test de la même façon que dans les montages précédents. Par la suite, avec les mesures de chacun des vecteurs, nous pouvons alors calculer le CDIDDQ. La figure 5.1 montre le schéma bloc de la configuration du système.

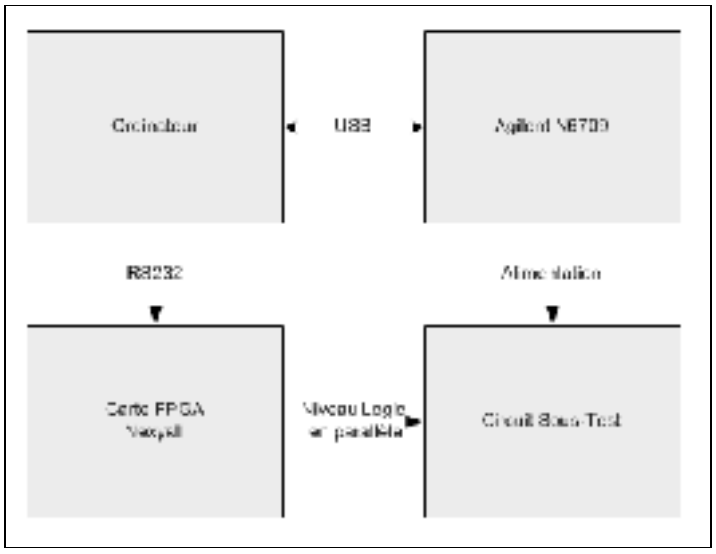

Figure 5.1 Configuration de la prise de mesure

Le nouveau testeur se divise principalement en 2 parties, soit le logiciel PC et le testeur FPGA.

5.2 Le testeur FPGA

Le testeur sur FPGA a pour fonction la stimulation du circuit avec des vecteurs de tests envoyés par l'ordinateur via une communication série. Nous voulons mettre au point un testeur simple, flexible et fiable que nous n'aurons jamais à modifier, peu importe les vecteurs de test à envoyer ou le circuit à tester. Pour ce faire, nous allons utiliser le même principe d'application des vecteurs que nous avons utilisé pour notre banc de test afin de

tester les 4 patrons. On se rappelle que le banc de test lit un fichier texte contenant des données et qu'il les affecte au port d'entrée du circuit sous test un peu avant le front montant l'horloge. Nous allons donc recevoir des données des vecteurs sous forme texte, via la communication série, que nous garderons en mémoire. Par la suite, nous utiliserons ces données pour les affecter aux sorties du testeur. Notre testeur agira en quelque sorte comme un registre à décalage. Comme indiqué précédemment, la carte de développement utilisée est la Nexys II, comme dans les montages précédents. Nous allons utiliser le connecteur de 100 broches où sont connectés 40 IOs du FPGA et un CLK-IO qui nous permet le lien au niveau logique avec le circuit à tester. De plus, puisque l'on veut ajouter la communication entre l'ordinateur et le FPGA, il nous faut donc ajouter les broches permettant cette connexion. La carte de développement utilisée par le testeur permet un seul port de communication série, soit la communication série RS-232. Ce protocole est simple à intégrer et à déverminer, bien qu'il ne soit pas très rapide (11,52ko/s dans notre cas). Le schéma bloc du testeur est affiché à la figure 5.2.

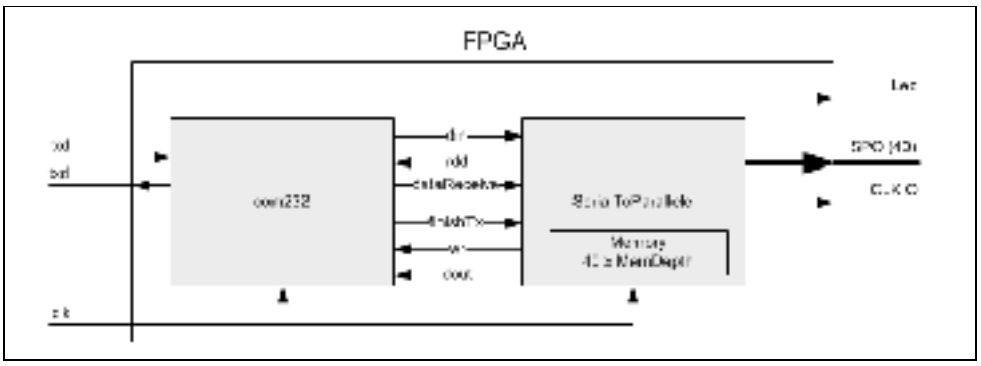

Figure 5.2 Schéma bloc du testeur FPGA

Le système se sépare principalement en 2 blocs, la communication série et le décalage des données.

5.2.1 La communication série

Comme mentionné précédemment, la communication série RS232 a été choisie pour la communication entre l'ordinateur et le testeur FPGA. Le RS-232 est un moyen de communication série asynchrone standardisé défini à la fin des années 1960 par l'industrie. C'est une communication point à point, ce qui veut dire qu'il peut y avoir seulement deux périphériques qui communiquent ensemble. Le câble standard est un DB9 que l'on peut brancher sur un ordinateur et qui nécessite l'utilisation minimale de trois fils, soit TX, RX et GND. Puisque la communication est asynchrone, il faut donc que les deux périphériques qui sont branchés ensemble aient la même configuration afin de déchiffrer correctement le message envoyé. Dans une communication série RS-232, on doit identifier le nombre de bits du message, le nombre de bits d'arrêt, la parité, ainsi que la vitesse de la communication. Une trame RS-232 de base commence toujours par un bit de départ afin d''informer l'hôte du commencement d'une trame, il est suivi d'un certain nombre de bits pour les données et d'un bit d'arrêt. Puisque les circuits numériques ne peuvent fournir les tensions nécessaires qui doivent aller de -12v à +12v, il est nécessaire d'avoir un transmetteur-récepteur RS-232 afin de fournir les tensions nécessaires. La figure 5.3 illustre une trame qui permet l'envoi du caractère 'J'.

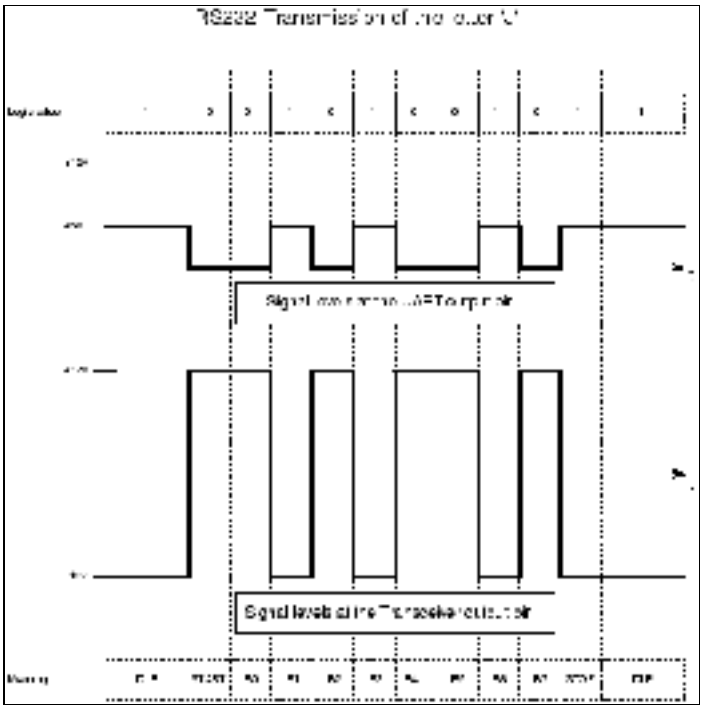

Figure 5.3 Envoi du caractère 'J'
Tirée de http://www.best-microcontroller-projects.com/how-rs232-works.html

Le fabricant de la carte Nexys 2 fournit gratuitement sur son site web le code VHDL permettant la communication pour la carte Nexys2. Bien que le code fourni compile et semble fonctionner à première vue, celui-ci comporte des problèmes majeurs qui nous empêchent de l'utiliser dans notre application. Premièrement, le module est conçu en half-duplex, ce qui signifie que l'on ne peut pas recevoir et transmettre des données en même temps. Deuxièmement, il est impossible avec ce module de communiquer en continu à la vitesse choisie (envoyer des caractères un à la suite de l'autre). Il nous apparait donc clair que le composant a été conçu pour recevoir des commandes provenant du clavier, ce pourquoi il fonctionne très bien, puisqu'il y a un délai entre chaque caractère lors de l'envoi. Puisque nous désirons automatiser le processus, les vecteurs seront envoyé sans délai. Il a donc fallu mettre au point un module VHDL de communication série qui pouvait combler les lacunes de celui offert. L'objectif ici est de pouvoir envoyer et recevoir en même temps et en continu à une vitesse de transmission de 115200 bauds. Il est important de noter qu'il n'y a pas de gestion des erreurs de communication dans le module que nous mettons au point. Il y

a 2 erreurs possibles : une erreur de trame et un écrasement. Aussi, le composant ne supporte pas le bit de parité, l'utilisation de plus d'un bit d'arrêt et des données de longueur autres que 8 bits. En bref, le composant est conçu pour recevoir des trames comme celle affichée à la figure 5.3. Nous avons donc 2 parties distinctes dans ce module, la réception et la transmission.

Pour le bloc de transmission nous aurons recours à 5 signaux. Le schéma bloc de la transmission est affiché à la figure 5.4.

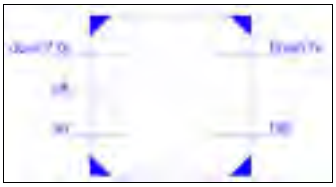

Figure 5.4 Schéma du bloc de transmission

Notre bloc comporte trois entrées (*dout*, *clk* et *wr*) et deux sorties (*finishTx* et *txd*). Le vecteur de huit bits dout représente la donnée que l'on désire envoyer tandis que le signal *wr* permet de démarrer l'envoie série de cette donnée. Pour sa part, le signal *finishTx* permet de savoir si la transmission, qui est émise via le signal *txd*, est terminée ou non. Tout l'envoi est synchronisé à un taux de 115200 bits par seconde, qui est dérivé du signal *clk*, l'horloge du FPGA. L'organigramme du bloc de transmission est affiché dans la figure 5.5.

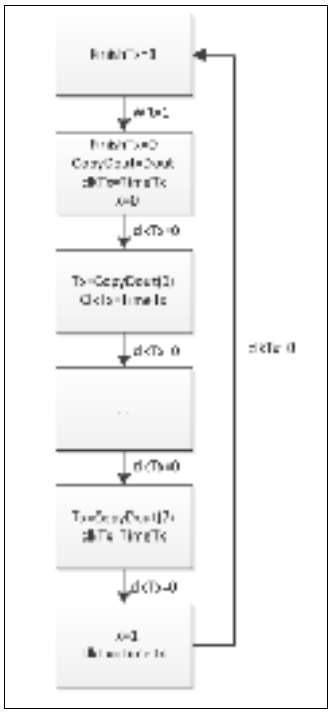

Figure 5.5 Organigramme de la transmission

Par défaut, le bloc de transmission est en attente d'un démarrage d'une transmission, finishTx est alors à '1'. Lorsque le signal *wr* est activé, le signal *finishTx* descend à zéro puisqu'une transmission débute. Une copie de la valeur de *dout* est faite dans une mémoire tampon afin de conserver les données tout le long de l'envoi. De plus, le signal *txd* est placé à '0' afin d'émettre un début de trame et le compteur qui permet de respecter les contraintes de temps pour l'envoi de chaque bit est activé. Une fois ce compteur tombé à zéro, on peut envoyer le premier bit et redémarrer le compteur. Après l'envoi des 8 bits, il ne nous reste plus qu'à envoyer le bit d'arrêt tout en redémarrant le compteur une dernière fois avant que l'on retombe en attente d'un autre envoi.

Le bloc *Rx* comporte lui aussi cinq signaux. Le schéma bloc de la réception est affiché à la figure 5.6.

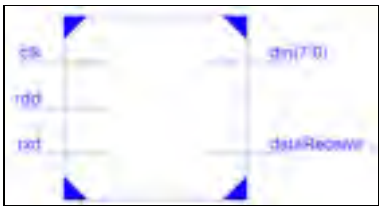

Figure 5.6 Schéma du bloc de réception

Notre bloc comporte aussi trois entrées et deux sorties. La trame rs232 est reçue en série par le signal *rxd*. Pour sa part, le signal *dataReceive* permet de savoir si un nouvel octet a été reçu. Lorsque le signal *rdd* monte à '1', le signal *Din* représente la valeur de l'octet reçu. L'organigramme du bloc de transmission est affiché dans la figure 5.7.

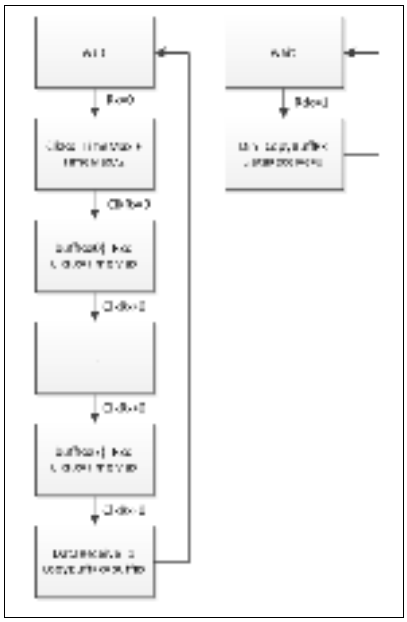

Figure 5.7 Organigramme de la réception

Le module de réception est un peu plus complexe que le module de transmission puisque l'on doit se synchroniser sur l'arrivée du bit de départ de chaque trame. Ainsi, lors de la détection d'un bit de départ, on démarre le compteur de réception. On l'initie avec une valeur 50% plus grande la première fois afin d'arriver au milieu de la réception de chacun des bits. Par la suite, chaque fois que le compteur tombe à zéro, un bit est lu et mis en mémoire et le

compteur est redémarré. Une fois rendu à la lecture du stop bit nous indiquons qu'une nouvelle donnée est disponible et nous copions la donnée dans une autre mémoire tampon avant de retourner en attente. On remarque dans l'organigramme que l'on se met attente plus tôt que l'on devrait puisque nous sommes seulement rendus à 50% du temps pour le bit d'arrêt. Cela nous permet de nous assurer que nous nous resynchroniserons parfaitement avec la prochaine trame même si elle arrive un peu plus tôt. En parallèle avec la réception, on peut activer le signal *rdd* qui copiera la mémoire tampon sur le signal *din* et placera le signal data receive à 0 afin d'indiquer qu'il n'y a plus de nouvelle valeur disponible.

Par la simulation, nous allons vérifier que nos blocs fonctionnent correctement et qu'ils répondent à nos besoins. On veut avoir la certitude que notre module est full duplex et qu'il peut recevoir et envoyer en continu. Pour ce faire, nous allons créer une boucle qui retourne toutes les trames rs-232 reçues. Il ne nous reste plus qu'à mettre au point un banc de test qui envoie en continu des caractères différents. Le résultat de la simulation est affiché à la figure 5.8.

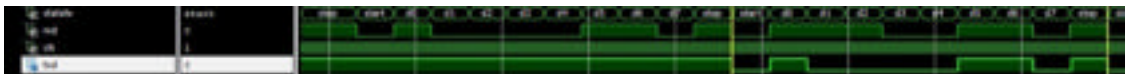

Figure 5.8 Simulation du module RS-232

La trame reçue est parfaitement retournée et on retransmet la trame pendant la réception d'une autre. Notre transmission et notre réception sont parfaitement fonctionnelles et il ne reste plus qu'à valider l'implémentation. Pour ce test, nous avons laissé en place la boucle et avons utilisé le programme RealTerm pour envoyer un fichier texte. Le résultat est affiché à la figure 5.9.

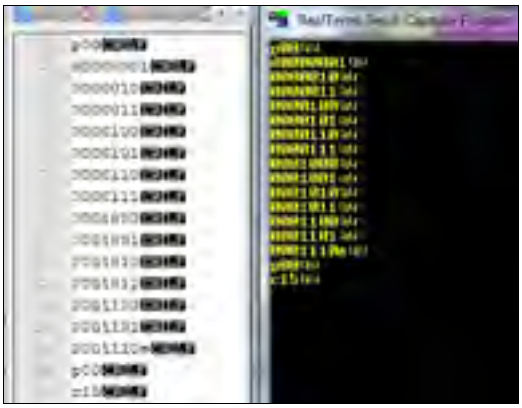

Figure 5.9 Validation de l'implémentation du module RS-232

À gauche de l'image se trouve le fichier texte envoyé tandis qu'à la droite se trouve le texte reçu. Comme on le constate, nous avons reçu exactement ce que nous avons envoyé. Nous avons donc réussi à créer un module de communication série qui répond à nos besoin.

5.2.2 Le décalage des données

Le module de décalage a principalement deux tâches : décoder les trames reçues et effectuer le décalage. Pour le décodage, le système doit analyser tous les caractères reçus afin de détecter un début d'une commande pour ensuite exécuter l'opération associée à celle-ci. Le tableau 5.1 indique les différentes commandes qu'il est possible d'envoyer au système.

Tableau 5.1 Tableau des commandes

Trame	Description	Exemple
p**	Cette commande a pour but de placer le pointeur d'écriture/lecture en mémoire. Elle commence par la lettre 'p' et est suivie de 2 caractères qui constituent l'adresse du pointeur mémoire. On doit absolument mettre 2 caractères pour l'adresse.	p05 (Place le pointeur mémoire à l'adresse 5.)

Trame	Description	Exemple
d[…]e	Cette commande a pour but de remplir la mémoire. Elle commence par la lettre ‘d’ et termine par ‘e’. On doit écrire des colonnes de 40 bits afin de remplir les 40 IOs du FPGA. Toutefois, on n’est pas obligé de remplir toute la profondeur de la mémoire. On peut donc stopper l’écriture avec l’envoi du ‘e’.	p03d01001ZZ01Z0Z110Z10Z100Z1Z0000ZZZ01Z0Z0Z101ZZ111Z1Z0Z110Z10Z100Z1Z0000ZZZ01Z0Z0Z1e (Se positionne sur la colonne 3 et écrit 2 colonnes en mémoire.)
c**	Cette commande a pour but de créer un décalage des données vers l’extérieur, donc sur les IO de la carte. Chaque donnée décalée est accompagnée d’un coup d’horloge sur la broche CLK-IO. Cette commande commence par un ‘c’ et est suivie de 2 caractères qui constituent le nombre de coups d’horloge et de données qui doivent être décalées	p07c13 (Se positionne sur la colonne 7 et génère 13 fronts montant sur CLK-IO tout en décalent 13 données (colonne 7-19) vers l’extérieur.)

Pour effectuer un décalage de données, on doit premièrement positionner le pointeur dans la mémoire et y insérer des données afin de remplir la mémoire à décalage. Ensuite, on doit placer le pointeur pour choisir l’emplacement de la lecture et envoyer le nombre de décalages que l’on désire avoir. Les décalages des données se feront comme illustré à la figure 5.10.

Figure 5.10 Décalage des données

À l’initialisation du système, les 40 broches du FPGA sont en haute impédance et resteront ainsi jusqu'à ce qu’il ait une demande de décalage de données. Une fois la demande envoyée, le système place la première donnée pointée par la commande ‘p’ et génère un front montant sur la broche CLK-IO après une demi-période d’horloges afin d’assurer la stabilité des

données. Après le dernier coup d'horloge, le système gardera la dernière valeur décalée sur les broches.

Notre partie sur le FPGA est maintenant terminée. Elle est simple à comprendre et offre une grande flexibilité qui nous donnera l'avantage de ne pas avoir à modifier le code de notre testeur, peu importe les vecteurs ou les circuits à tester. La prochaine étape est de mettre au point un logiciel qui nous permettra d'envoyer et de décaler les vecteurs automatiquement.

5.3 Le logiciel PC

Comme mentionné précédemment, le logiciel PC a pour but d'envoyer les vecteurs de test aux FPGA afin que ceux-ci les appliquent sur le circuit sous test. Une fois les vecteurs appliqués, il prendra la mesure de consommation de courant du circuit pour chacun d'eux. Le programme sera développé sur Excel, donc, programmé en VBA. Puisque le logiciel d'acquisition communiquant avec l'Agilent est développé sur Excel, nous avons choisi de garder le même logiciel pour communiquer avec notre nouveau testeur. Le programme se divise en 4 parties, soit les vecteurs de test, la communication série, la prise de mesure et l'automatisation.

5.3.1 Les vecteurs de test

Dans les lignes qui suivent, la construction et l'envoi des vecteurs de test sont expliqués. La figure 5.11 affiche l'interface de la partie de l'envoi des vecteurs de test.

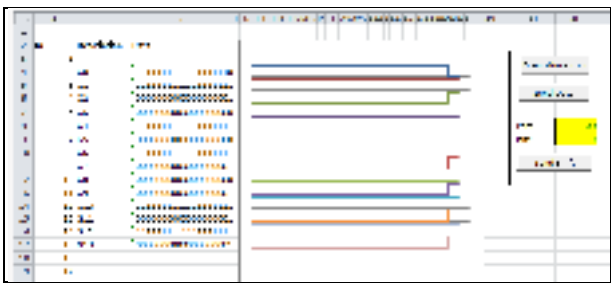

Figure 5.11 Interface de vecteurs de test

L'envoi des vecteurs se fait en 2 étapes ; écrire les vecteurs à envoyer et, par la suite, les envoyer. Pour écrire les vecteurs, on doit entrer la valeur logique que l'on désire sur les 40 IOs disponibles du testeur. Si une ligne est laissée vide, elle sera configurée en haute impédance dans le FPGA. Il est possible de visualiser la forme d'onde équivalente directement dans l'interface. Il est important de noter que les données seront décalées de gauche à droite.

Une fois les données entrées, on peut alors les envoyer au FPGA. Cette opération crée un fichier texte avec les informations des vecteurs afin que le FPGA les enregistre dans sa mémoire interne, comme expliqué précédemment. La figure 5.12 affiche le fichier texte résultant des vecteurs écrits.

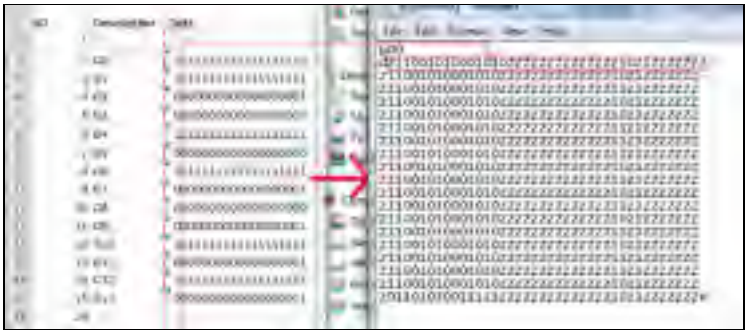

Figure 5.12 Fichier texte envoyé au testeur FPGA

La première ligne du fichier texte indique au FPGA de placer son pointeur d'adresse mémoire à l'adresse 0. Par la suite, on envoie le caractère *d* afin d'informer que les données qui suivront doivent être stockées en mémoire. Finalement, le caractère *e* à la fin du fichier texte, indique la fin de la transmission des données d'envoyées.

Une fois le fichier texte envoyé, on peut choisir où l'on veut placer le pointeur mémoire et combien de décalages on désire faire. Chaque décalage de la mémoire est accompagné par un coup d'horloge sur CLK-IO retardé d'une demi-période. Pour créer ce décalage à partir de l'interface, on doit informer le programme de l'endroit où l'on désire commencer dans la

mémoire (*From*) et combien de décalages on désire faire (*Shift*). Donc, par exemple, dans la figure 5.13, on place le pointeur à l'adresse 0 dans la mémoire et on effectue 19 décalages qui seront donc accompagnés de 19 coups d'horloges.

Figure 5.13 Configuration du décalage

Il nous faut donc remplir 4 onglets, puisque l'on désire envoyer 4 vecteurs. On peut tester individuellement chaque vecteur les envoyant séparément.

5.3.2 La communication série

Pour la communication série nous avons utilisé un code sur internet (http://cherbe.free.fr/rs232.html) qui permet d'utiliser le port série de l'ordinateur avec Excel. Nous n'avons effectué aucune modification au code qui fonctionnait parfaitement. Comme mentionné auparavant, la communication RS-232 est asynchrone, et, donc, les deux modules doivent savoir à quelle vitesse communiquer. La figure 5.14 affiche l'interface de la communication série.

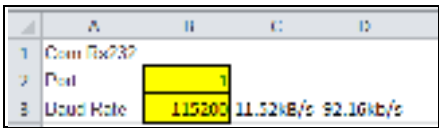

Figure 5.14 Interface de configuration de la communication série

Il est seulement possible de configurer la vitesse de communication ainsi que le numéro du port utilisé. Il est donc important de noter que la communication se fait sans bit de parité, avec 8 bits de données et un bit d'arrêt.

5.3.3 La prise de mesures

L'outil de prise de mesure est, comme on peut le constater, à la figure 5.15, une version modifiée du logiciel que nous avons mis au point et utilisé depuis le début. Son fonctionnement est identique. Certaines simplifications ont été apportées dans le but premier d'alléger le programme afin que l'acquisition des données se fasse plus rapidement. Pour ce faire, nous avons optimisé ou réécrit certaines fonctions du code déjà écrit. Nous avons enlevé tout le code lié au traitement des exceptions. Nous tenons pour acquis que les paramètres et que l'outil sera utilisé adéquatement. Pour finir, nous avons enlevé le traçage de la forme d'onde automatique, que nous avons rendue possible via un bouton.

Figure 5.15 La prise de mesure

5.4 Validation du système

Avant d'automatiser le processus du calcul du CDIDDQ, qui inclut l'envoi des vecteurs ainsi que la prise de mesures, nous devons d'abord tester notre code. Nous savons déjà que la prise de mesure est fonctionnelle puisque le code n'a été que légèrement modifié. Nous voulons donc tester ce qui est nouvellement ajouté, autrement dit, nous voulons valider que le logiciel génère bien les vecteurs et que le FPGA les envoie correctement. Pour ce faire, nous allons envoyer un vecteur de test et vérifier avec l'analyseur logique si le FPGA l'a bien appliqué.

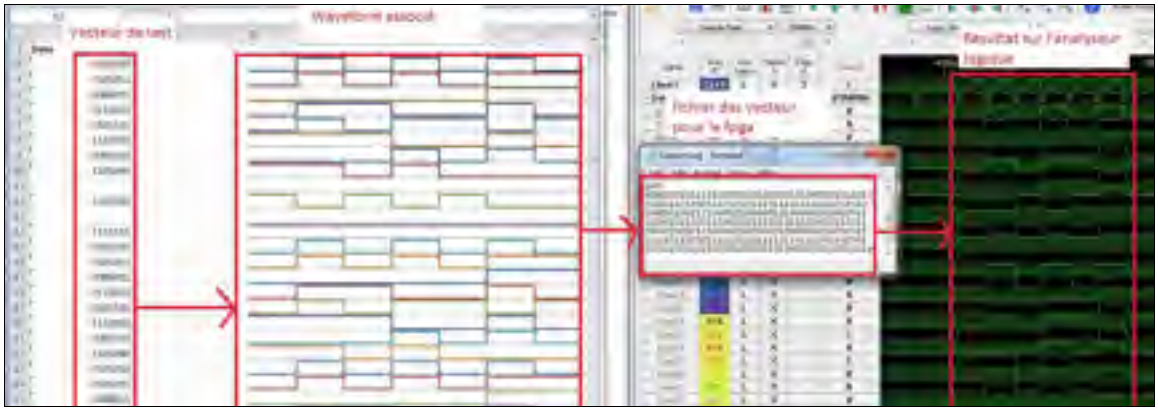

Figure 5.16 Vérification du système

Comme on peut le voir dans la figure 5.16, le texte écrit a été converti en vecteurs, puis écrit dans un fichier texte, puis envoyé vers le FPGA qui a décalé ces données sur ces 40 broches d'entrée-sortie. On en conclut que la communication série et le décalage sont fonctionnels.

La prochaine étape est d'effectuer un test avec les vecteurs CDIDDQ. On devrait obtenir les paires logiques 0-0, 1-1, 0-1 et 1-0. Les images ci-dessous affichent le résultat des 4 vecteurs de test envoyés.

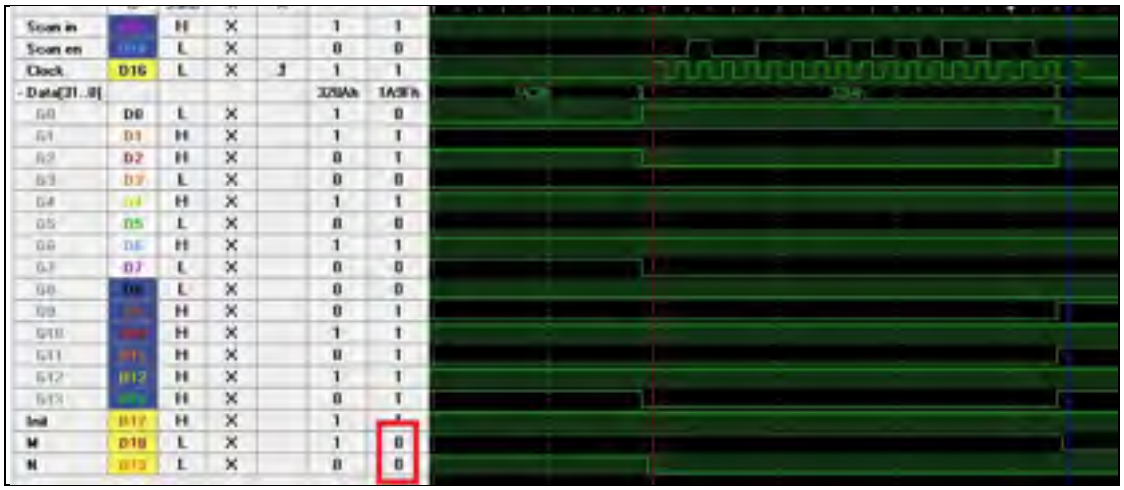

Figure 5.17 Paire 0-0

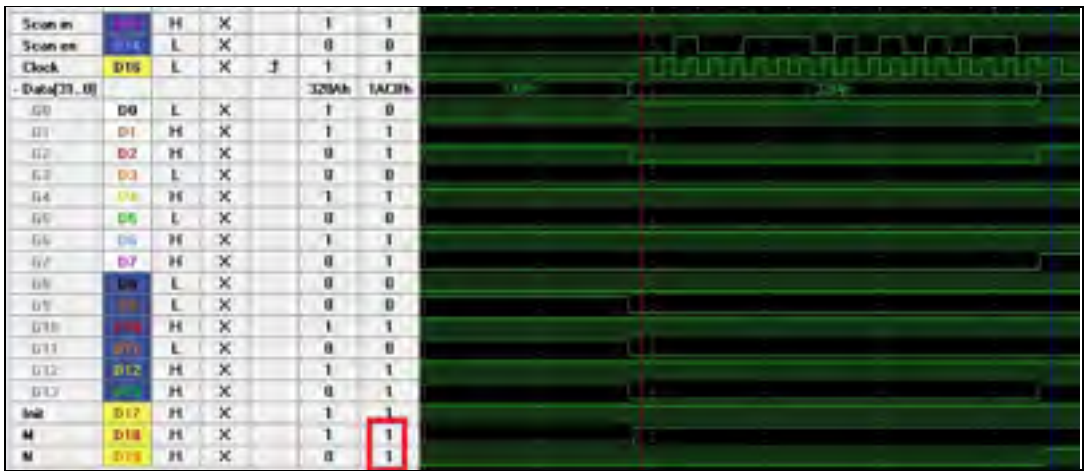

Figure 5.18 Paire 1-1

Figure 5.19 Paire 0-1

Figure 5.20 Paire 1-0

Comme l'indiquent les figures 5.17, 5.18, 5.19 et 5.20, notre système envoie correctement les vecteurs de test puisque nous avons les valeurs logiques désirées. Chacune des parties étant fonctionnelle, on peut alors passer à la dernière étape de notre système, soit l'automatisation.

5.5 L'automatisation

Afin d'accroître la vitesse du calcul CDIDDQ nous allons automatiser l'envoi et la prise de mesure des vecteurs. L'interface pour l'automatisation est affichée à la figure 5.21.

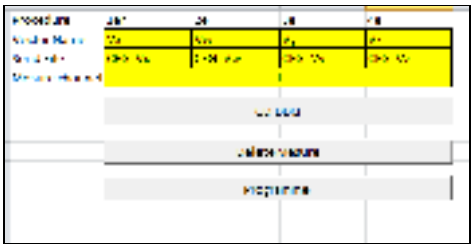

Figure 5.21 Interface d'automatisation

Cette interface nous permet de choisir l'ordre des vecteurs envoyés, de choisir le nom que portera le fichier de mesure de ce vecteur et de choisir le nom de l'onglet Excel qui comporte les données du vecteur à envoyer. Finalement, on peut choisir sur quel canal on désire effectuer les mesures. Une fois l'automatisation configurée, il ne reste plus qu'à démarrer un test CDIDDQ en spécifiant le nom du test. On utilise quatre fichiers pour générer cinq autres fichiers pour chaque test CDIDD. Les quatre fichiers nécessaires sont les données liées à chacun des vecteurs afin qu'on puisse les envoyer au testeur qui s'occupera de les appliquer au circuit sous test. Parmi les fichiers générés, on retrouve les quatre fichiers qui comportent les échantillons de mesures de courant effectuées face à chacun des vecteurs ainsi que le fichier CDIDDQ qui utilise les quatre fichiers de mesure pour procéder au calcul.

5.6 Résultats et analyse

Nous avons mis au point un nouveau testeur qui nous permet de prendre les mesures CDIDDQ plus rapidement. Nous savons déjà que les résultats obtenus sans source courant seront pratiquement les mêmes que ceux obtenus précédemment. En effet, nous avions déjà un écart type très près de la mesure à vide (sans carte), comme on peut le voir sur les figures 5.22 et 5.23.

A	B	
0.000067173	Moyenne(A)	Pa
0.000043976	0.000000802	Pa
0.000012711	Ecart-Type	Pa
0.000007460	0.000035073	Pa
0.000022588	Ecart-Type/Moyenne(%)	Pa
0.000032674	4373.747636	Pa
-0.000037717	Max	Pa
-0.000041751	0.000097429	Pa
-0.000048811	Min	Pa
-0.000049819	-0.000090162	Pa
-0.000055871	Peek-to-Peek	
0.000028640	0.000187591	

Figure 5.22 Mesures à vide

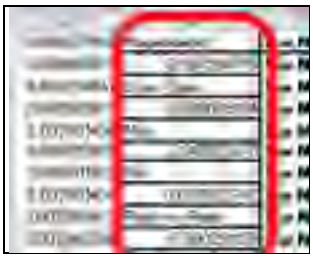

Figure 5.23 Mesure avec la carte Sparkfun

L'objectif de ce montage est donc de pouvoir améliorer nos résultats CDIDDQ sur le montage à deux sources. La configuration que nous avons utilisée pour l'acquisition de la mesure de courant de chaque vecteur est affichée à la figure 5.24. Nous avons utilisé le même taux d'échantillonnage et la même échelle de mesure que précédemment. Toutefois, nous avons réduit le nombre d'échantillons à 250 afin de réduire le temps nécessaire par mesure de vecteur.

State:	Agilent Technologies,N6705A,MY47000911,D.01.06
IoAdress:	USB0::0x0957::0x0F07::MY47000911::0::INSTR
Channel	1
Sampling Rate(s)(max10u):	0.000010
Nb Points(max 512000):	250
Mesure Range (A):	0.100000
Com Timeout(s):	20
Total Time(s):	0.0025

Figure 5.24 Configuration de l'acquisition

5.6.1 Le patron de test 17

Nous avons donc commencé par procéder au calcul CDIDDQ du patron 17 comme nous l'avons fait pour tous les autres types de montage. Afin d'avoir un maximum d'échantillons, nous avons effectué, pour chacune des résistances, 25 tests CDIDDQ avec lesquels nous avons fait la moyenne et rempli le tableau 5.2. Il nous apparaît donc que chaque valeur affichée dans le tableau est le fruit de 25000 mesures de courant.

Tableau 5.2 Résultat, patron de test P17

P17	**1 source <0.1A**	**2 sources 0.2A**	**1 source 1A**	**2 sources 1A**	**2 sources 2A**	**2 sources 3A**
0.1k	4.73E-02	4.70E-02	4.66E-02	4.58E-02	4.51E-02	4.47E-02
1k	6.14E-03	6.10E-03	6.14E-03	5.96E-03	5.87E-03	5.82E-03
13k	5.12E-04	5.06E-04	4.24E-04	4.90E-04	4.82E-04	4.98E-04
39k	1.70E-04	1.71E-04	1.63E-04	1.61E-04	1.68E-04	1.55E-04
75k	8.33E-05	9.09E-05	1.50E-04	8.79E-05	8.54E-05	9.69E-05
110k	6.12E-05	6.70E-05	7.24E-06	5.60E-05	5.06E-05	4.70E-05
300k	2.14E-05	1.86E-05	-6.56E-05	2.22E-05	3.02E-05	4.73E-05
500k	1.54E-05	1.84E-05	-4.36E-05	1.30E-05	1.47E-05	1.83E-06
infini	-2.69E-06	8.33E-06	8.33E-06	8.33E-06	8.33E-06	8.33E-06
Écart-type	7.48E-06	1.47E-05	4.61E-04	1.71E-05	2.07E-05	2.66E-05
Seuil	1.97E-05	3.86E-05	1.35E-03	4.82E-05	5.62E-05	8.56E-05

Dans le tableau, nous avons placé en rouge les valeurs au-dessus du seuil de défectuosité. Le seuil est le même que celui fixé dans le chapitre 4. On rappelle que le seuil est la valeur sans défectuosité (CDIDDQ(infini)) plus trois fois l'écart-type. Six séries de résultats CDIDDQ apparaissent dans ce tableau, de la seconde à la septième colonne : avec une seule source sans courant de fuite additionnel (via résistances en parallèle), avec 2 sources et un courant de fuite additionnel de 0.2A, avec une seule source et un courant de fuite additionnel de 1A, avec 2 sources et un courant de fuite additionnel de 1, 2 et 3A. Comme on peut le constater, plus le courant de consommation augmente, moins on détecte de défectuosités. Ceci est causé par l'écart-type des mesures de courant qui augmente plus la consommation est élevée. Aussi, on observe une plus grande variation des valeurs de CDIDDQ entre les différents montages lorsque la résistance testée est élevée. En effet, la résistance de 0.1kΩ fourni toujours un résultat autour d'environ 4.6E-02 tandis que ce n'est pas le cas pour la résistance de 510kΩ. Ceci est dû au fait que le rapport entre les variations des vecteurs CDIDDQ et le bruit de mesure diminue plus la résistance est élevée. La colonne *1 source 1A* s'avère être le calcul CDIDDQ pour un montage 1A, mais avec une seule source, et, donc, en augmentant l'échelle de mesure de 100mA à 3A. Comme on peut le constater, c'est le montage qui détecte le moins de défectuosités, ce qui confirme l'obtention de meilleurs résultats avec l'utilisation de deux sources pour les montages consommant plus de 100mA. Nous pouvons désormais détecter des défectuosités de 75kΩ et moins pour tous les montages contrairement au 39kΩ et moins obtenues au chapitre 4, ce qui a presque doublé la capacité de détection de circuit résistif. En termes de courant supplémentaire généré par un court-circuit, cette capacité de détection correspond à environ la moitié du seuil, sachant que la valeur CDIDDQ est égale à la somme des courants de court-circuit qui est activé par 2 des 4 vecteurs et supposant que les courants de ces 2 activations soient similaires. Ainsi, les résultats suggèrent qu'il est possible de détecter un delta supplémentaire de courant de 43µA sur un courant de fuite de 3A. Il est important de noter que, pour les mêmes données, si on réduit le seuil à seulement la valeur absolue de CDIDDQ(infini), comme utilisé par (Haithem 2011), on détecte, comme affiché au tableau 5.3, toutes les défectuosités à l'exception de la 500k pour le montage 3A.

Tableau 5.3 En utilisant le seuil d'Haithem (CDIDDQ(infini))

P17	**1 source <0.1A**	**2 sources 0.2A**	**1 source 1A**	**2 sources 1A**	**2 sources 2A**	**2 sources 3A**
0.1k	4.73E-02	4.70E-02	4.66E-02	4.58E-02	4.51E-02	4.47E-02
1k	6.14E-03	6.10E-03	6.14E-03	5.96E-03	5.87E-03	5.82E-03
13k	5.12E-04	5.06E-04	4.24E-04	4.90E-04	4.82E-04	4.98E-04
39k	1.70E-04	1.71E-04	1.63E-04	1.61E-04	1.68E-04	1.55E-04
75k	8.33E-05	9.09E-05	1.50E-04	8.79E-05	8.54E-05	9.69E-05
110k	6.12E-05	6.70E-05	7.24E-06	5.60E-05	5.06E-05	4.70E-05
300k	2.14E-05	1.86E-05	-6.56E-05	2.22E-05	3.02E-05	4.73E-05
500k	1.54E-05	1.84E-05	-4.36E-05	1.30E-05	1.47E-05	1.83E-06
infini	-2.69E-06	8.33E-06	8.33E-06	8.33E-06	8.33E-06	8.33E-06
Écart-type	7.48E-06	1.47E-05	4.61E-04	1.71E-05	2.07E-05	2.66E-05
Seuil	1.97E-05	3.86E-05	1.35E-03	4.82E-05	5.62E-05	8.56E-05

En traçant la courbe de CDIDDQ pour chacun des montages, on remarque que contrairement à celle obtenue au chapitre 4, il y a beaucoup moins d'oscillations. Les graphiques sont affichés dans les figures 5.25, 5.26, 5.27 et 5.28.

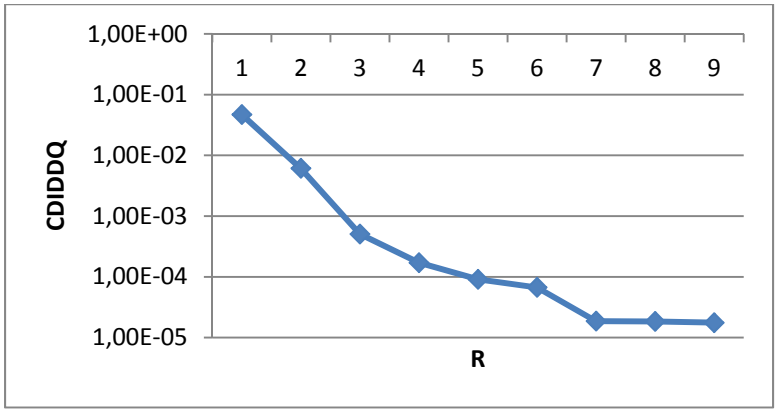

Figure 5.25 2 sources 200mA

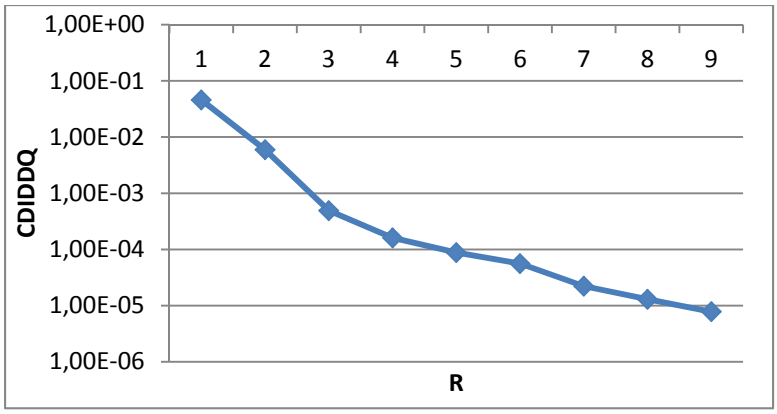

Figure 5.26 2 sources 1A

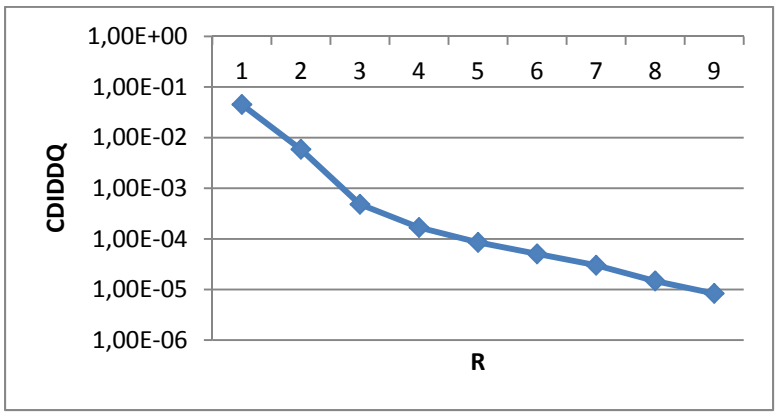

Figure 5.27 2 sources 2A

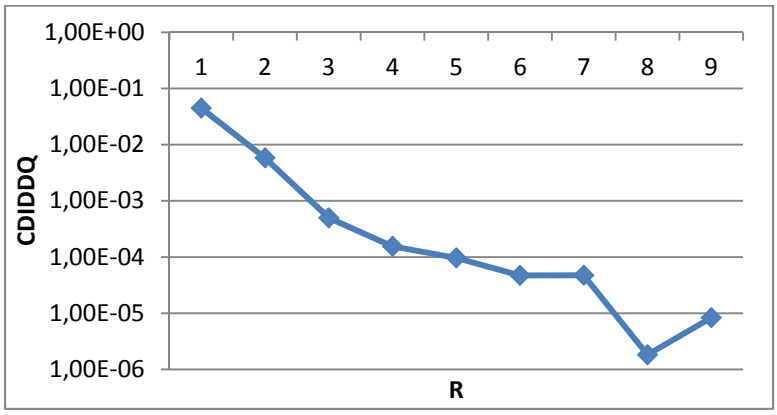

Figure 5.28 2 sources 3A

Il est important de noter que, puisque nous utilisons une échelle logarithmique, nous avons dû ajouter une valeur de 0.000008 à toutes les valeurs de CDIDDQ présentées précédemment. Cette valeur a été ajoutée dans l'objectif d'avoir toutes les données positives afin de tracer les graphiques dans lesquels on observe seulement la forme générale.

5.6.2 Les patrons de test 5, 6 et 29

Nous avons exécuté le même test CDIDDQ pour les patrons 5, 6 et 29 avec les mêmes défectuosités résistives. Les résultats de tous les patrons sont affichés dans les tableaux 5.4, 5.5 et 5.6.

Tableau 5.4 Résultat, patron de test P5

P5	**1 source <0.1A**	**2 sources 0.2A**	**2 sources 1A**	**2 sources 2A**	**2 sources 3A**
0.1k	4.52E-02	4.71E-02	4.64E-02	4.51E-02	4.40E-02
1k	6.13E-03	6.12E-03	6.03E-03	5.84E-03	5.78E-03
13k	5.14E-04	5.14E-04	5.09E-04	4.77E-04	6.08E-04
39k	1.70E-04	1.75E-04	1.63E-04	1.37E-04	1.05E-04
75k	9.17E-05	9.28E-05	9.32E-05	1.32E-05	4.48E-05
110k	6.13E-05	6.20E-05	6.36E-05	1.07E-05	4.73E-05
300k	1.78E-05	1.61E-05	2.33E-05	4.42E-06	-3.13E-05
500k	1.23E-05	1.30E-05	6.85E-06	6.57E-06	-3.20E-05
infini	-1.04E-05	9.07E-06	3.93E-07	-1.16E-04	-4.65E-06
Écart-type	9.72E-06	2.04E-05	1.85E-05	7.64E-05	3.55E-05
Seuil	1.87E-05	7.02E-05	5.58E-05	1.13E-05	1.02E-05

Tableau 5.5 Résultats patron de test P6

P6	**1 source <0.1A**	**2 sources 0.2A**	**2 sources 1A**	**2 sources 2A**	**2 sources 3A**
0.1k	4.71E-02	4.7E-02	4.61E-02	4.52E-02	4.40E-02
1k	6.17E-03	6.17E-03	6.05E-03	5.94E-03	5.79E-03
13k	5.09E-04	5.06E-04	5.01E-04	4.88E-04	6.38E-04
39k	1.72E-04	1.72E-04	1.68E-04	1.74E-04	1.24E-04
75k	8.80E-05	8.70E-05	8.56E-05	8.31E-05	7.27E-05
110k	6.08E-05	6.22E-05	6.12E-05	5.25E-05	5.12E-05
300k	2.10E-05	2.05E-05	1.22E-05	2.01E-05	-8.80E-07
500k	1.20E-05	1.10E-05	1.78E-05	7.59E-06	-9.15E-06
infini	-1.45E-06	3.60E-06	-2.12E-05	-8.21E-06	-3.01E-05
Écart-type	7.94E-06	1.84E-05	2.45E-05	2.84E-05	3.41E-05
Seuil	2.24E-05	5.89E-05	5.25E-05	7.70E-05	7.22E-05

Tableau 5.6 Résultats patron de test P29

P29	**1 source <0.1A**	**2 sources 0.2A**	**2 sources 1A**	**2 sources 2A**	**2 sources 3A**
0.1k	4.71E-02	4.70E-02	4.60E-02	4.52E-02	4.41E-02
1k	6.18E-03	6.17E-03	6.04E-03	5.93E-03	5.76E-03
13k	5.10E-04	5.16E-04	5.02E-04	4.80E-04	4.70E-04
39k	1.70E-04	1.63E-04	1.61E-04	1.59E-04	1.40E-04
75k	8.86E-05	8.26E-05	8.83E-05	8.31E-05	3.51E-05
110k	5.98E-05	5.47E-05	6.03E-05	5.91E-05	4.16E-05
300k	2.30E-05	2.24E-05	1.94E-05	2.35E-05	9.94E-05
500k	1.25E-05	1.53E-05	1.18E-05	7.71E-06	-6.33E-06
infini	3.84E-07	5.61E-06	-8.56E-07	-4.16E-06	-3.56E-06

P29	**1 source <0.1A**	**2 sources 0.2A**	**2 sources 1A**	**2 sources 2A**	**2 sources 3A**
Écart-type	8.34E-06	1.59E-05	1.92E-05	2.35E-05	3.82E-05
Seuil	2.54E-05	5.33E-05	5.67E-05	6.64E-05	1.11E-04

Comme on peut le remarquer, on obtient sensiblement le même résultat avec les autres patrons. Malheureusement, on n'arrive pas à détecter, pour le même seuil, la résistance de 75k dans tous les montages. Sachant que les trois autres patrons tests sont constitués de vecteurs différents, il nous apparaît donc que nos résultats ne sont pas uniques au patron 17.

5.7 Conclusion

Dans ce chapitre, nous avons expliqué en détail le dernier montage final utilisé. Nous avons discuté de l'architecture du nouveau testeur ainsi que la structure du logiciel PC mise en place. Par la suite, nous avons testé le système afin de valider qu'il répondait à nos besoins. Finalement, nous avons appliqué les quatre patrons de test et calculé le CDIDDQ. En utilisant le même patron et le même seuil que (Haithem 2011), nous sommes parvenus à détecter presque la totalité des courts-circuits insérés, et ce, pour presque tous les montages. Nous sommes donc parvenus, pour un circuit avec une seule source, à détecter jusqu'à une résistance de 510kΩ contrairement au 80kΩ des travaux passés, ce qui correspond à une augmentation de plus de 500%. D'un autre côté, en utilisant le même seuil que celui du chapitre 4, nous sommes presque parvenus à doubler notre capacité à détecter des défectuosités pour le patron de test 17 (75kΩ contre 39kΩ). Finalement, nous avons appliqué les autres patrons présentés et nous avons obtenu sensiblement les mêmes résultats que pour le patron 17, ce qui confirme le fonctionnement de la technique pour d'autres patrons de test.

CONCLUSION

Les pannes de court-circuit sont un problème auquel doivent faire face les fabricants de puces électroniques et ce problème continuera certainement de persister. La miniaturisation des circuits intégrés a accru ce type de panne sans toutefois rendre leur détection plus aisée, bien au contraire. La technique de test et diagnostic CDIDDQ est une solution proposée pour la détection de ce type de panne, qui cause un courant additionnel de plus en plus petit qu'il faut détecter malgré une augmentation du courant normal de fuite. Nous avons pu constater dans les travaux précédents les résultats expérimentaux prometteurs de cette technique (Haithem 2011).

Au cours de ce projet, nous avons amélioré l'application de la technique, ce qui a permis d'obtenir de meilleurs résultats. Ainsi, nous avons été en mesure de détecter un plus grand nombre de défectuosités et, ce, même avec l'utilisation d'un seuil plus contraignant et comportant un degré de certitude plus élevé.

Pour commencer, nous avons procédé à la répétition de l'application expérimentale de CDIDDQ exécutée précédemment par (Haithem 2011) afin de nous familiariser avec la technique et confirmer les résultats obtenus. Nous avons utilisé le même montage, le même patron de test CDIDDQ, le même testeur et le même circuit sous-test. Toutefois, nous avons mis au point un logiciel PC permettant une meilleure acquisition des mesures, qui, vraisemblablement, a été la source des légères différences entre les résultats originaux et les nôtres.

Par la suite, nous avons changé le montage en modifiant la partie du circuit sous test. Nous avons gardé le même FPGA, un Spartan-3E de Xilinx, mais sur une nouvelle carte de développement générant moins de bruit sur l'alimentation. Nous avons refait l'expérimentation et avons obtenu de meilleurs résultats. Par la suite, sur cette même carte, nous avons contourné l'alimentation en place afin de fournir directement celle-ci à notre

circuit sous-test. Cela a pour effet de réduire davantage le bruit sur l'alimentation et nous a permis de raffiner une fois de plus nos résultats.

Satisfaits des résultats sur un circuit comportant seulement un FPGA, nous avons élargi le champ d'application en appliquant CDIDDQ sur des circuits plus énergivores. Pour ces montages, nous avons utilisé la technique de l'alimentation par deux sources afin de pouvoir garder une échelle de mesure basse malgré la haute consommation des montages.

Finalement, nous avons créé un logiciel PC d'automatisation de test CDIDDQ qui nous a permis d'effectuer les tests plus rapidement afin de diminuer l'effet de la variation de température de notre montage. Aussi, nous avons mis en place une nouvelle architecture pour le testeur afin de répondre à nos nouveaux besoins. Pour ce montage, nous avons aussi changé notre système d'alimentation deux sources. Ces dernières améliorations nous ont permis d'obtenir nos meilleurs résultats.

En conclusion, nos travaux ont permis d'accroître l'étendue des résultats pour l'application de la technique de test CDIDDQ sur un FPGA émulant des pannes de court-circuit, et ce, avec ou sans surconsommation.

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

Andres, D and Ruiz, J.-C.; Gil, D. 2006. « Fast Emulation of Permanent Faults in VLSI Systems ». Field Programmable Logic and Applications, 2006. FPL '06. International Conference on. (Aug. 2006), p. 1-6.

Breuer, M. A. and A. D. Friedman. 1976. Diagnosis & reliable design of digital systems. Woodland Hills, Calif., Computer Science Press.

Bushnell, M. L. and V. D. Agrawal. 2000. Essentials of electronic testing for digital, memory, and mixed-signal VLSI circuits. Boston, Kluwer Academic.

Brglez, F., Bryan, D. and Kozminski, K. 1989. « Combinational profiles of sequential benchmark circuits ». IEEE International Symposium, vol. 3, p. 1929-1934.

Chakravarty, S. and P. J. Thadikaran 1997. Introduction to IDDQ testing. Boston, Kluwer Academic Publishers.

Courtoy, M. 1995. « Emulation: prototyping without the hassles of FPGA-to-ASIC conversion ». WESCON/'95. Conference record.Microelectronics Communications Technology Producing Quality Products Mobile and Portable Power Emerging Technologies.(7-9 Nov. 1995). p. 283.

Ebgelke, Piet. 2006. « Delta-IDDQ Testing of Resistive Short Defects ». Test Symposium, ATS'06. 15th Asian, p. 63-68.

Haithem, K. 2011. « ADAPTATION DE L'APPROCHE DE TEST CDIDDQ AUX CIRCUITS PROGRAMMABLES FPGA ». Mémoire de maîtrise en génie électrique, Montréal, École de technologie supérieure, 129 p.

Kasap, S. O. 2006. Principles of Electronic Materials and Devices, Mc-Graw Hill.

Kuen-Jong Lee. 1991. « Constraints for using IDDQ testing to detect CMOS bridging faults ». VLSI Test Symposium, 'Chip-to-System Test Concerns for the 90's', Digest of Paper. (15-17 April 1991), p. 303-308.

Manhaeve, Hans. 2005. « Current Testing for Nanotechnologies: A Demystifying Application Perspective ». Test Symposium, 2005. Proceedings. 14th Asian. (21-21 Dec. 2005), p. 456,456.

Maxfield, C. 2004. The Design Warrior's Guide to FPGAs: Devices, Tools and Flows, Newnes.

Moore, Gordon E. 1965. « Cramming more components onto integrated circuits ». Electronic Magazine, vol. 38, no 8, p. 114-117.

Mourad, S. and Y. Zorian. 2000. Principles of testing electronic systems. New York, John Wiley & Sons.

Nyquist, H. 1928. « Thermal Agitation of Electric Charge in Conductors ». Physical Review, vol. 32, no 1, p. 110-113.

Powell, T.J. 2000. « Delta Iddq for testing reliability ». VLSI Test Symposium 200, 18th IEEE, p. 439-443.

Rajsuman, Rochit. 2000. « Iddq testing for CMOS VLSI ». Proceedings of the IEEE 88(4), vol. 88, no 4, p. 544-568.

Riezenman, M. J. 1991, « Wanlass's CMOS circuit, IEEE spectrum », vol 28, no. 5, p. 44.

Shinogi, T., Hayashi, T. 1998. « A simple and efficient method for generating compact IDDQ test set for bridging faults ». VLSI Test Symposium, 1998, Proceedings. 16th IEEE. (26-30 Apr 1998), p.112,117.

Smith, G. L. 1985. « Models for delay faults based upon paths ». Pro. International Test Conference, p. 342-349.

Thibeault, C. 1998. « Increasing current testing resolution ». 1998 IEEE International Symposium on Defect and Fault Tolerance in Vlsi Systems, Proceedings: p. 126-134.

Thibeault, C. 1999. « On the comparison of ΔIDDQ and IDDQ testing ». 17th IEEE, *VLSI Test Symposium*, Proceedings. p. 143-150.

Thibeault, C., Hariri, Y 2009. « CDIDDQ: Improving Current-Based Testing and Diagnosis through Adapted Test Pattern Generation », IEEE Trans. on VLSI Systems, vol.19, no.1, janv. 2011. (CRSNG)