Contribution à la synthèse des circuits asynchrones Quasi Insensibles aux Délais, application aux systèmes sécurisés

Bertrand Folco

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

N° attribué par la bibliothèque
|_|_|_|_|_|_|_|_|_|_|

THÈSE

pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité : Micro et Nano Électronique

préparée au laboratoire **TIMA** dans le cadre de
l'École Doctorale d'Électronique, d'Électrotechnique, d'Automatique et de Traitement du Signal

présentée et soutenue publiquement par

Bertrand FOLCO

le 4 Octobre 2007

Titre :

CONTRIBUTION À LA SYNTHÈSE DE CIRCUITS ASYNCHRONES QUASI INSENSIBLES AUX DÉLAIS, APPLICATION AUX SYSTÈMES SÉCURISÉS

Directeur de thèse : M^r^. Marc RENAUDIN
Co-directeur : M^r^. Gilles SICARD

Jury

M^r^. Régis LEVEUGLE,	Président
M^r^. Christian PIGUET,	Rapporteur
M^r^. Bruno ROUZEYRE,	Rapporteur
M^r^. Marc RENAUDIN,	Directeur de Thèse
M^r^. Gilles SICARD,	Co-directeur
M^r^. Philippe MAURINE,	Examinateur
M^r^. Jacques SONZOGNI,	Examinateur
M^r^. Jean-Baptiste RIGAUD,	Examinateur

Remerciements

Ce mémoire de thèse est la synthèse de quatre années de recherches effectuées au sein du laboratoire TIMA sur le site Viallet de l'Institut National Polytechnique de Grenoble.

Tout d'abord, je souhaite remercier M. Bernard COURTOIS, directeur du laboratoire TIMA au début de cette thèse, ainsi que Mme Dominique BORRIONE, actuelle directrice, pour leurs accueils.

Toute ma gratitude est adressée à Marc Renaudin, mon directeur de thèse, professeur à l'Institut National Polytechnique de Grenoble, pour m'avoir accueilli lors de mon stage de licence en Informatique et conseillé de faire cette thèse. Tous mes remerciements pour toutes ces années passées ensemble, ces courses de kart effrénées et cette nouvelle aventure qui commence !

Je remercie vivement mon co-encadrant Gilles Sicard, maître de Conférence à l'Université Joseph Fourier, pour ses conseils, son humour et pour m'avoir donné mes premières armes en analogique.

Je souhaite remercier les membres de mon jury de thèse :

M. Christian PIGUET, directeur de recherche au Centre Suisse d'Electronique et de Microélectronique à Neuchâtel (CSEM), et M. Bruno ROUZEYRE, professeur à l'Université Montpellier II, pour m'avoir fait l'honneur de participer à mon jury de thèse en tant que rapporteurs.

M. Régis LEVEUGLE, professeur à l'institut National Polytechnique de Grenoble, pour m'avoir fait l'honneur de présider mon jury de thèse.

M. Philippe MAURINE, maître de conférence à l'Université Montpellier II, M. Jacques SONZOGNI, design manager à STMicroelectronics à Rousset et M. Jean-Baptiste RIGAUD, maître de conférence à l'Ecole Nationale Supérieure des Mines de Saint-Etienne, pour avoir accepté d'être examinateurs de ma thèse.

Je remercie également toutes les personnes extérieures qui m'ont aidé pour ce travail de thèse. Pascal VIVET du CEA-Leti pour son aide précieuse sur les cellules de bibliothèque. Marc THOURET, Pascal GILGENKRATZ et Sylvain LANDELLE de ST Microelectronics à Crolles pour leur patience et leur aide pour la caractérisation des cellules.

Une pensée particulière pour Laurent FESQUET qui m'a aidé dés mon arrivée au TIMA lors de mon premier stage. Laurent merci pour ton aide continue pendant toutes ces années et pour tes réponses toujours précises.

Je remercie toutes les personnes du TIMA et du CMP qui m'ont épaulé pendant ces années : Isabelle, Patricia, Chantal, Ahmed, Joëlle, Anne-Laure, Sophie M. (qui a été notre voisine de bureau et qui a réussi à nous supporter), Fred et Nicolas, Greg 'pti Dip' et Sébastien. Tous mes remerciements à Kholdoun et

à Jean-François pour leur aide.

Je remercie vivement les administrateurs du CIME, Alexandre Chagoya, Robin Rolland et Bernard Bosc pour leur patience, leur aide et leur disponibilité. Vous avez changé ma vision des administrateurs systèmes.

Je souhaite remercier chaleureusement tous les membres de cette si particulière équipe CiS. Tout d'abord les anciens, avec JB pour sa gentillesse, son aide et ses fameux coups de gueule, Anh Vu pour nos rares et précieuses discussions, Kamel et Amine pour m'avoir accueilli dans leur « bureau », Dhanistha pour sa gentillesse et ses manies, Manu pour sa manière unique de cacher l'odeur de kebab dans notre bureau, Fabien et ses peluches, Joao pour son aide et cette sortie inoubliable à l'Alpe d'Huez, Arnaud pour Arnaud, Jérôme pour ses péripéties, Salim pour sa bonne humeur, Nicolas pour sa bonhommie. Les nouveaux en commençants par Cedric, ses éternelles tongs et son string de boxe thaï, David pour ses idées de pari inimaginables (et sa prise de risques sur un certain pari que j'ai injustement perdu), Alin pour sa bonne humeur et son aide, Yann pour son honnêteté et son humour noir, Estelle pour son aide en DEA, sa joie de vivre et ses précieux conseils, Vivian pour son aide et ses fameux t-shirts, Greg pour nos parties endiablées de badminton, sa répartie et sa gentillesse, Fraidy pour sa sagesse, ses conseils précieux et notre périple en Australie. Les petits derniers avec Taha, Jérémie, Eslam et les autres ...

Merci à Yannick pour tous ses souvenirs impérissables : en Allemagne, en Italie et en Australie ; pour toutes les soirées et ces moments passés ensemble qui nous ont vraiment rapprochés. Je suis heureux de poursuivre cette aventure avec toi. A Aurélien et Audrey merci pour tous ses précieux moments passés ensemble ; de nos parties inoubliables et enjouées à n'importe quels jeux, jusqu'à notre saut en élastique (le mien était mieux ...). A Livier ma mexicaine préférée, à qui j'ai essayé d'apprendre les particularités de la langue française, pour tous ses bons moments passés ensemble en soirées, au labo et sur les pistes de ski.

Je remercie énormément ma famille, mon frère et sa charmante femme Claudia (heureusement que je suis là ...), mon père et en particuliers ma mère et mon grand-père sans qui rien n'aurait été possible. Une pensée particulière à Jean-Marc qui a toujours été là.

Enfin, je remercie affectueusement ma Sophie, pour son soutien indéfectible, sa gentillesse, notre complicité et tout ce qu'elle m'apporte. Son sourire m'a apporté un bonheur de tous les instants.

Résumé

Les travaux présentés dans cette thèse portent sur le développement d'une méthodologie de conception de circuits asynchrones Quasi Insensibles aux Délais (QDI) et son application à des circuits sécurisés. Contrairement aux circuits synchrones, les circuits asynchrones se caractérisent par l'absence de signal d'horloge. Ces circuits sont séquencés par un mécanisme de communication et de synchronisation local. En plus des nombreuses propriétés des circuits asynchrones telles que la robustesse, une faible consommation, un faible bruit et une excellente modularité, les propriétés de la logique QDI apparaissent également particulièrement intéressantes pour sécuriser l'implantation des circuits intégrés contre les attaques par analyse de courant. Cependant, le manque de méthode et d'outil de conception est un frein à leur adoption. C'est dans ce contexte que se situe ce travail de thèse, qui contribue au développement d'un outil de conception de circuits asynchrones développé au laboratoire TIMA : TAST.

Dans un premier temps, nous avons développé une bibliothèque de cellules asynchrones, nommée TAL, conçue pour diminuer la surface et la consommation des circuits asynchrones QDI. Nous avons ensuite élaboré des algorithmes de projection technologique adaptés aux circuits asynchrones QDI, c'est-à-dire n'introduisant aucun aléa logique lors des étapes de décomposition, de partitionnement et de couverture du réseau logique. Ces algorithmes ont été implantés dans l'outil TAST et validés par la conception d'un ensemble de circuits asynchrones. Dans un deuxième temps, nous avons développé un outil d'analyse de la sensibilité des circuits asynchrones face aux attaques par analyse en courant. Cet outil permet d'identifier, pendant la phase de conception, les zones du circuit comportant une dissymétrie électrique susceptible de fournir des informations lors d'une attaque par analyse en courant. Les données ainsi obtenues peuvent être utilisées pour modifier la topologie du circuit afin d'améliorer sa robustesse vis-à-vis de l'attaque DPA.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Contexte et motivations

La prépondérance historique des circuits synchrones est un fait indéniable. La conception des circuits numériques emprunte aujourd'hui de manière quasi-systématique l'hypothèse synchrone. L'utilisation de cette hypothèse réductrice de synchronisation a permis pendant des années d'exploiter le potentiel des technologies d'intégration et de concevoir des circuits de plus en plus complexes et rapides. Au début de l'intégration des circuits, le but était l'intégration des circuits et non la recherche de performances : le mot d'ordre était d'intégrer le plus de fonctions possibles dans le moins de surface. L'hypothèse synchrone était la plus adaptée et l'horloge permettait d'organiser facilement le séquencement des circuits. Les outils de CAO s'orientèrent alors vers ce type de méthodologie. Les langages de description de matériel offrirent le moyen de modéliser au niveau RTL, permettant de séparer la description fonctionnelle des performances temporelles. Les flots de conception de type ASIC étaient nés : en utilisant une logique CMOS, des cellules standard et les outils de synthèse sur les langages HDL, il devint possible de concevoir avec beaucoup de productivité des systèmes relativement complexes.

Dans le développement actuel de circuits intégrés, les performances deviennent un argument stratégique. Dans la course au MHz, la période d'horloge est une ressource de plus en plus précieuse, l'horloge n'est plus une « alliée ». En particulier pour la conception de circuits hautes performances, la diminution de la période d'horloge impose de nouvelles méthodes de conception : utilisation de logique dynamique, arbres d'horloge complexes, microarchitectures super-pipelinées. Les outils de CAO, toujours basés sur les langages de haut niveau, sont contraints à tous les niveaux par les performances : optimisation des arbres d'horloge, analyse statique de timing, insertion de délais, synthèse logique contrainte par le placement (et réciproquement), analyse du bruit de couplage, arrêt de l'horloge afin de réduire la consommation ou le bruit électromagnétique.

Avec les technologies actuelles, arrivent de nouveaux challenges : dispersions technologiques de plus en plus fortes au sein d'une même puce, prédominance des interconnexions sur la logique (délais, bruit), consommation de plus en plus élevée (aléas, horloge) et enfin problématique d'intégration des systèmes complexes tels que les « systèmes sur puce » (SoC). Ce dernier point devient prédominant. La multipli-

cité des horloges entre les différents blocs fonctionnels souvent issus de concepteurs différents et fonctionnant à des fréquences différentes ; la propagation d'un signal d'horloge sur une puce de plusieurs millimètres de côté ; mais également les problèmes de synchronisation dus aux dispersions technologiques augmentent le coût et la complexité de conception de ces circuits. Dans ce contexte, l'hypothèse de synchronisation par horloge globale apparaît trop réductrice pour offrir un niveau d'abstraction élevé.

Les circuits asynchrones apparaissent comme une solution naturelle au problème et viable aux limitations liées à la présence d'une synchronisation globale, c'est pourquoi de plus en plus d'acteurs s'intéressent à ce style de circuits. De forts potentiels peuvent être attendus : robustesse, temps de calcul moyen, consommation conditionnelle, faible bruit, faible émission électromagnétique, modularité, niveau d'abstraction plus élevé. Les circuits asynchrones ont montré leurs potentiels intéressants dans plusieurs domaines d'application : la conception de microprocesseurs, de cartes à puce, de circuits à faible consommation et de circuits résistants aux attaques par analyse en courant ou par injection de fautes.

Malgré les avantages et le potentiel des circuits asynchrones, de nombreux facteurs expliquent pourquoi les flots de conception asynchrones ne brillent que par leur absence - ou presque. Tout d'abord le fait que peu d'applications industrielles en asynchrone aient vu le jour jusqu'à présent souligne bien le fait que la motivation industrielle pour l'asynchrone n'en est qu'à ses débuts. Or le développement d'un outil complet de synthèse asynchrone est une entreprise conséquente. Elle nécessite un apport important de la part de l'industrie afin de passer de méthodologies complexes inhérentes à la recherche, à un flot complet exploitable à grande échelle. Par ailleurs, la complexité des calculs mis en jeu dans un tel outil constitue elle aussi un obstacle majeur, d'autant plus que les compétences réutilisables depuis le domaine synchrone sont limitées.

D'autre part, la conception VLSI synchrone a atteint un niveau de maturité et d'automatisation tel que le secteur n'est pas prêt à un retour en arrière en termes de méthodologie (outils) et de productivité. La conception de circuits asynchrones reste encore le domaine de spécialistes utilisant des outils faiblement automatisés. Sa diffusion dans le secteur industriel ne sera pas possible tant que les outils des circuits asynchrones n'auront pas atteint le même niveau d'automatisation que ceux des circuits synchrones, ni tant que des ingénieurs n'auront pas été formés pour leur utilisation.

Dans ce contexte, le travail de thèse s'inscrit dans la volonté du laboratoire TIMA (« Techniques de l'Informatique et de la Microélectronique pour l'Architecture des ordinateurs ») de développer un environnement de conception dédié à la conception de circuits asynchrones. Cet environnement de conception actuellement en cours de développement se nomme TAST pour « TIMA Asynchronous Synthesis

Tool ». Il est principalement composé d'un compilateur-synthétiseur capable de cibler des circuits de type QDI à partir d'une description haut niveau exprimée dans un langage proche de CHP (Communicating Hardware Processes). Après analyse, les programmes sont représentés en interne par des réseaux de Petri associés à des graphes de flot de données. Ce type de représentation, est particulièrement adapté dans le domaine des circuits asynchrones pour la vérification formelle de propriétés et la validation par simulation comportementale. Une seconde forme intermédiaire basée sur les diagrammes de décision multi-valués (MDD), généralisation des diagrammes de décision binaires utilisés en synchrone, est utilisée pour synthétiser les circuits QDI et réaliser l'étape de projection technologique. Le langage de description haut niveau utilisé pour modéliser nos circuits nous permet de définir les protocoles de communication, la définition du codage des données, les choix déterministes et indéterministes, la gestion de la hiérarchie et la génération de traces. La figure 1 présente les flots de conception supportés par l'environnement TAST.

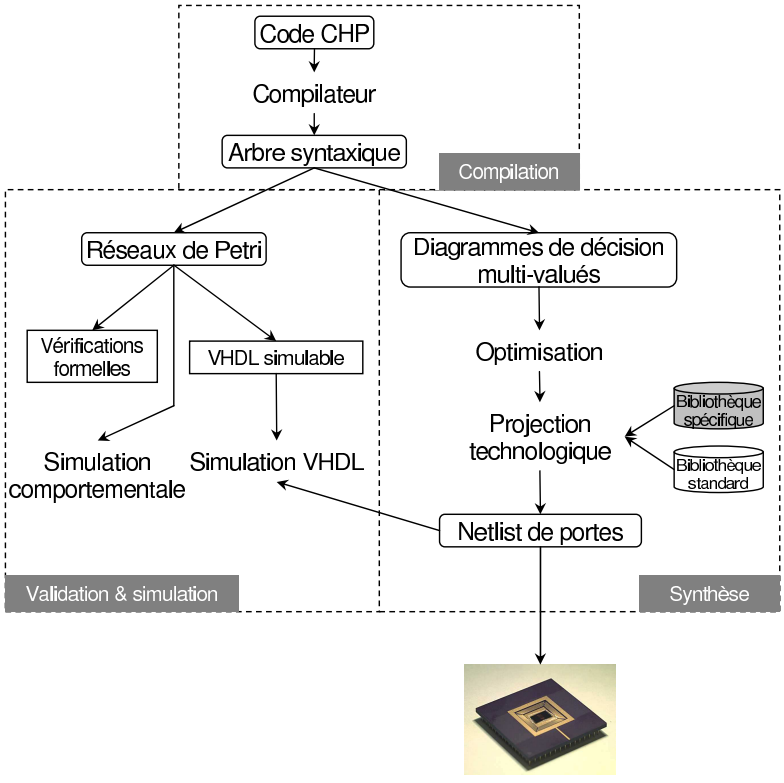


FIGURE 1 – Flots de conception supportés par TAST.

Dans ce contexte, le premier objectif de ce travail de thèse est de mettre au point des algorithmes de projection technologique dédiés aux circuits asynchrones Quasi Insensibles aux Délais (QDI). Les algorithmes communément utilisés dans le domaine synchrone ne sont pas directement applicables pour ce type de circuits car ils ne permettent pas de contrôler l'introduction d'aléas dans la logique. Une nouvelle méthode a donc été étudiée pour permettre de projeter des circuits asynchrones sans aléa de

type quasiment insensibles aux délais sur des bibliothèques cibles. Cette méthode a ensuite été implantée dans la partie back-end de l'outil TAST. Afin d'améliorer les performances des circuits obtenus, une bibliothèque de cellules asynchrones a été développée en partenariat avec le LIRM de Montpellier. Cette bibliothèque, du nom de TAL (« TIMA Asynchronous Library »), a pour objectif de diminuer la surface des circuits tout en améliorant leur vitesse et leur consommation en comparaison de l'utilisation de portes standard synchrones.

Dans le cadre de notre partenariat avec STMicroelectronics à Rousset, nous avons cherché à valider l'insertion des circuits asynchrones dans un flot de conception standard pour produits sécurisés. Les propriétés intrinsèques de la logique asynchrone (notamment des circuits QDI) offrent de manière globale un niveau de sécurité plus élevé que leur équivalent synchrone face à des attaques non intrusives. L'absence de signal d'horloge, la redondance des chemins logiques et l'utilisation d'un codage de données de type « 1 parmi n » associé à un protocole de communication de type « poignée de main » de type « 4 phases », sont des atouts incontestables pour résister aux attaques non intrusives. Une étude a donc été menée, en collaboration avec STMicroelectronics, avec pour double objectif de valider l'insertion d'un circuit asynchrone au sein d'un produit synchrone sécurisé existant, mais également de valider l'intégration de notre bibliothèque de cellules asynchrones dans un flot de conception standard synchrone basé sur des outils de synthèse commerciaux. Un dernier aspect de ce travail a été d'étudier la sensibilité des circuits asynchrones aux attaques par analyse en courant. Pour cela, un outil d'analyse de netlist a été intégré à TAST afin d'identifier les zones du circuit susceptibles de laisser fuir des informations.

Le premier chapitre du manuscrit présente tout d'abord le mode de fonctionnement asynchrone ainsi que ses nombreux avantages par rapport au mode synchrone. Les concepts de base des circuits asynchrones sont ensuite définis. Les communications entre les blocs sont localement synchronisées grâce au codage des données et au protocole de communication de type « poignée de main » choisi. Pour implanter un tel protocole, la notion de porte de Muller est introduite. Le contrôle du circuit étant réalisé de façon locale par l'utilisation d'événements sur les canaux de communication, les circuits asynchrones doivent être exempts d'aléa. Cette notion d'aléa est donc présentée dans ce chapitre. Finalement, suivant les hypothèses faites sur le modèle de délais appliqué aux portes et aux interconnexions, une classification des circuits asynchrones est réalisée.

Le deuxième chapitre présente une partie des méthodologies de conception de circuits asynchrones et les outils associés. Chaque méthodologie est présentée avec ses avantages et ses inconvénients. Principalement deux méthodes de spécification des circuits asynchrones sont étudiées : la première basée sur un langage de description de haut niveau et la seconde basée sur un graphe. Enfin nous présentons l'environnement de conception de circuits asynchrones TAST développé au laboratoire TIMA.

Le principe et les algorithmes standard de la projection technologique sont présentés dans le chapitre trois. Les trois phases de la projection technologique sont décrites séparément : la décomposition, le partitionnement et la couverture. Nous expliquons ensuite les restrictions de ces algorithmes vis-à-vis des circuits asynchrones. Nous proposons par la suite une description des différentes approches existantes de projection technologique pour les circuits asynchrones. Finalement une vue d'ensemble des bibliothèques de cellules asynchrones existantes est présentée, avec leurs avantages et leurs inconvénients.

Le chapitre 4 présente la bibliothèque de cellules asynchrones TAL pour « TIMA Asynchronous Library » développée durant cette thèse. Les caractéristiques ainsi que les plans et les choix de conception sont présentés. Une comparaison en surface, vitesse et consommation de la bibliothèque TAL est réalisée, par rapport à l'utilisation de portes de Muller à base de portes standard AO222. Les résultats obtenus sont analysés pour expliquer le fonctionnement des topologies de cellules choisies. Une seconde version de la bibliothèque TAL est présentée par la suite. Cette nouvelle version, a été conçue à partir de la technologie 65nm de STMicroelectronics, et des choix différents ont été pris pour les topologies des cellules. Une comparaison de cette nouvelle bibliothèque est également réalisée par rapport à l'utilisation de portes de Muller à base de AO222.

Les algorithmes de projection technologique pour les circuits asynchrones QDI, mis au point durant cette thèse, sont présentés dans le cinquième chapitre. Dans un premier temps, nous présentons les diagrammes de décisions multi-valués, la phase de décomposition étant basée sur ces derniers. La méthode de décomposition présentée permet de conserver la propriété d'insensibilité aux délais des circuits en assurant qu'aucun aléa n'est introduit. Nous présentons ensuite les algorithmes des phases de partitionnement et de couverture. Chaque algorithme est décrit successivement et illustré par des exemples. Finalement les résultats obtenus sur des circuits tests, en utilisant ces algorithmes, sont présentés.

Afin de valider notre bibliothèque de cellules asynchrones TAL et son intégration dans un flot de synthèse utilisant des outils commerciaux, nous avons développé plusieurs circuits cryptographiques de type DES. Ces circuits et les résultats associés sont présentés dans le sixième chapitre. Dans un premier temps, une étude est menée pour évaluer les gains apportés par la bibliothèque TAL au niveau d'un circuit complet. Dans un deuxième temps, nous présentons les résultats obtenus en collaboration avec STMicroelectronics, sur l'intégration d'un DES asynchrone dans un produit smartcard 8 bits existant. Finalement, diverses modifications sont proposées sur l'architecture du DES afin de prouver la viabilité des circuits asynchrones.

Le dernier chapitre est consacré au développement d'un outil d'analyse de la sensibilité des circuits asynchrones aux attaques par analyse de courant. L'objectif de cet outil est d'analyser formellement les

circuits en logique asynchrone QDI afin de pouvoir, dans un premier temps identifier l'origine des fuites d'information résiduelles observées et de proposer par la suite de nouvelles méthodes de conception en vue de supprimer ces fuites ou de les rendre complètement inexploitables par toutes attaques en puissance. L'analyse porte sur les dissymétries logiques et électriques dans la netlist du circuit. Une fois l'analyse terminée, les zones susceptibles de laisser fuir une information secrète sont localisées, et une modification de la netlist peut être effectuée afin d'augmenter la résistance de ces zones.

Enfin, la conclusion dresse le bilan du travail effectué et propose des perspectives à ce travail.

Partie I

ÉTAT DE L'ART

Chapitre 1

Les circuits asynchrones

1.1 Introduction

Lorsque la conception de circuits numériques a débuté, il n'existait pas de distinction entre circuit synchrone et asynchrone. Les circuits synchrones correspondent à une classe restreinte de circuits qui sont séquencés par un signal périodique uniformément distribué : l'horloge. Au contraire les circuits asynchrones sont des circuits dont le contrôle est assuré par une toute autre méthode que le recours à un signal d'horloge global. Le contrôle se fait de manière locale par une synchronisation entre blocs fonctionnels. Très vite le style de conception synchrone s'est imposé pour répondre à des besoins de calcul croissants et pour s'adapter à une technologie encore bridée.

Pourtant l'étude des circuits asynchrones a commencé au début des années 1950 pour concevoir des circuits à relais mécaniques. En 1956, Muller et Bartky, de l'université de l'Illinois, ont travaillé sur la théorie des circuits asynchrones. Huffman est le premier à concevoir des machines à états asynchrones en 1968 avec ses travaux en « switching theory ». Ces travaux ont ensuite été étendus par Huffman lui-même, Muller, Unger et Mac Cluskey. Muller fut le premier à proposer d'associer un signal de validité aux données, introduisant ainsi un protocole de communication quatre phases. En 1966, le « Macromodule Project » lancé par W.A. Clark de l'université de Washington, St Louis [24] montre qu'il est possible de concevoir des machines spécialisées complexes par simple composition de blocs fonctionnels asynchrones. Par la suite, Seitz introduit un formalisme proche des réseaux de Petri pour concevoir des circuits asynchrones, ce qui aboutit à la construction du premier calculateur « dataflow » (DDM-1) [29]. Enfin, en 1989, Yvan Sutherland a largement contribué à l'intérêt croissant porté par les institutions académiques mais aussi industrielles à la conception de circuits asynchrones en publiant un article maintenant célèbre intitulé « Micropipeline » [89]. Depuis, les travaux sur la conception de circuits asynchrones ne cessent de s'intensifier [48].

Aujourd'hui, malgré la prépondérance des circuits synchrones, des outils de conception associés à ce type de circuits qui ne cessent de progresser, ainsi qu'une formation d'ingénieurs uniquement consa-

crée à ce style de conception, de plus en plus d'acteurs du domaine du semiconducteur s'intéressent à la conception asynchrone. En effet, le rôle de plus en plus important des variations de process, de température et de tension dans la conception de circuits en technologie submicronique, rend extrêmement complexe la conception de circuits synchrones [25]. Les circuits asynchrones sont donc l'objet d'un regain d'intérêt et d'activité de recherche du fait de leurs bénéfices potentiels en faible consommation, faible bruit, robustesse aux variations (technologiques ou d'utilisation) et modularité [11].

Ce chapitre est largement inspiré d'un rapport écrit par Marc Renaudin sur l'état de l'art de la conception de circuits asynchrones [78].

1.2 Avantages des circuits asynchrones

1.2.1 Absence d'horloge

L'avantage évident des circuits asynchrones, en raison de l'absence de signal d'horloge global, est que tous les problèmes liés à la manipulation de l'horloge sont supprimés. Ces problèmes sont dans les technologies actuelles de plus en plus présents car ces technologies autorisent des circuits de plus en plus complexes, tant sur le plan fonctionnel qu'au niveau de leur fabrication, et de plus en plus rapides. Aujourd'hui, la conception des circuits d'horloge est devenue une question de toute première importance puisqu'ils peuvent directement limiter les performances du système synchrone. Par exemple pour les « systèmes sur puce » (SoC), la multiplicité des horloges entre les différents blocs fonctionnels souvent issus de concepteurs différents et fonctionnant à des fréquences différentes, la propagation d'un signal d'horloge sur une puce de plusieurs millimètres de côté, mais également les problèmes de synchronisation dus aux dispersions technologiques augmentent le coût et la complexité de conception de ces circuits. D'une manière générale, l'approche synchrone a des conséquences à tous les niveaux de la conception : estimation des capacités d'interconnexion avant la phase de synthèse logique, caractérisation des timings des éléments de bibliothèque, en particulier les temps de setup et de hold, estimation de timing après extraction sur layout, confiance dans la caractérisation de la technologie, conception électrique et placement physique des arbres d'horloge pour réduire les problèmes de gigues, génération de vecteurs de test pour valider / tester les chemins critiques et trier les circuits après fabrication en fonction de leurs performances. Les techniques et outils de conception de circuits synchrones évoluent avec les technologies pour estimer de plus en plus précisément les temps d'arrivée de l'horloge sur les bascules d'un circuit.

Au contraire les circuits asynchrones n'utilisent pas d'horloge globale. Les éléments de synchronisation sont distribués dans l'ensemble du circuit, leur conception est plus facile à maîtriser. Comme pour certains circuits asynchrones, le fonctionnement est indépendant des retards qui peuvent être introduits,

le problème de temps d'arrivée d'un signal d'un bout à l'autre du circuit n'influence pas la correction fonctionnelle du système. Le principe de synchronisation locale des circuits asynchrones constitue alors directement un outil d'aide à la conception de systèmes complexes.

1.2.2 Faible consommation

Par rapport aux circuits synchrones, plusieurs facteurs de réduction de la consommation sont à prendre en compte. La première conséquence de la suppression de l'horloge est la suppression de la consommation associée à celle-ci. Dans les circuits rapides, la consommation de l'horloge et des éléments de mémorisation peut représenter jusqu'à plus de 50% de la consommation du circuit. Cette consommation est due au chargement des circuits d'horloge et aux transitions dans les bascules. De plus, dans les circuits numériques, une part non négligeable de la consommation provient de transitions non utiles en raison de la présence d'aléas dans les blocs de logiques combinatoires. En synchrone, ces aléas ne sont pas gênants fonctionnellement car ils doivent avoir disparus à l'arrivée du prochain front d'horloge, cependant ils représentent une consommation relativement importante. Dans les circuits asynchrones, leur conception doit supprimer tout type d'aléas afin d'obtenir des circuits corrects fonctionnellement (cf. 1.4.1). La part de consommation due à ces aléas est donc supprimée.

Un autre aspect important de la réduction de la consommation concerne la mise en veille de la logique asynchrone à tout niveau de granularité. Dans un circuit synchrone, tous les blocs de logique se trouvent alimentés en données et commutent à l'arrivée de l'horloge, que ces transitions soient utiles dans le bloc ou non. Par exemple dans un microprocesseur, la plupart des blocs n'ont pas besoin de travailler pour toutes les instructions : le multiplieur est utile pour une multiplication, un additionneur pour une addition et non le contraire. Comme le mécanisme de synchronisation à horloge est simplifié, tous les blocs fonctionnent alors que fonctionnellement ils n'apportent rien. Au contraire, grâce au mécanisme de synchronisation locale des circuits asynchrones et à leur fonctionnement flot de données, seuls les blocs utiles reçoivent des données et donc consomment. Cet argument de réduction de la consommation des circuits asynchrones est donc particulièrement intéressant dans le cas d'architectures irrégulières. Des optimisations sont cependant possibles en synchrone pour contrôler l'horloge (« gated-clock »), mais leur conception reste difficile à mettre en œuvre, alors que la mise en veille de la logique à tous les niveaux de granularité est gratuite dans le cas asynchrone. Un autre avantage de cette activité conditionnelle de la logique est que le redémarrage de la logique est instantané et géré au niveau matériel à tous les niveaux de l'architecture : il n'est pas nécessaire de concevoir un logiciel, parfois complexe, pour contrôler l'activation et la désactivation de tout ou partie du système.

Une dernière et intéressante propriété des circuits asynchrones pour la faible consommation est leur robustesse et leur adaptation aux conditions de fonctionnement. Comme la puissance varie avec le carré

de la tension, il est aisé de réduire la tension d'alimentation pour limiter la puissance consommée. Cette réduction de la tension d'alimentation des circuits asynchrones peut se faire avec un matériel et un temps de conception minimum. Contrairement aux circuits synchrones, il n'y a pas besoin d'adapter et de caractériser la fréquence d'horloge aux différentes conditions de fonctionnement car la correction fonctionnelle est garantie quels que soient les délais dans les cellules élémentaires. De manière statique, il est ainsi aisément possible de choisir entre performance et faible consommation dans un circuit asynchrone : le circuit sera d'autant moins consommant à tension réduite que performant à tension nominale. De plus, il est également possible de faire varier dynamiquement la tension d'alimentation d'un circuit en fonction de son activité pour réduire la consommation [80].

Malgré toutes ces propriétés intéressantes, il est difficile de conclure de façon certaine sur la faible consommation des circuits asynchrones. En effet, en raison des mécanismes de synchronisation locale, et de l'utilisation de codages des données particuliers (double-rail, ...), les circuits asynchrones coûtent chers à implanter. Les protocoles asynchrones représentent plus de transitions que le mécanisme de synchronisation à horloge. Même si la suppression de l'horloge permet de supprimer toute consommation liée à celle-ci, il est difficile aujourd'hui de conclure de manière absolue [85].

1.2.3 Faible bruit, circuits intrinsèquement résistants

Une autre conséquence de la suppression de l'horloge et de la distribution du contrôle dans la structure du circuit est que les problèmes de pics de consommation sont inexistants. En effet, l'activité électrique d'un circuit asynchrone est bien mieux répartie dans le temps que pour un circuit synchrone. Il n'y a pas d'instants prédéfinis pour activer un opérateur comme c'est le cas avec les fronts d'horloge. Ainsi, la consommation dans les lignes du circuit est distribuée dans le temps, que ce soit au niveau de la logique ou des éléments de mémorisation. De plus, comme nous l'avons vu, l'absence de l'horloge supprime sa consommation, ainsi le bruit généré dans les lignes d'alimentation et la puissance des ondes magnétiques émises par le circuit sont limités. Les circuits asynchrones présentent alors une alternative sérieuse pour concevoir des systèmes numériques sous contraintes de limitation de bruit.

Cette propriété est également très intéressante pour l'utilisation des circuits asynchrones dans des produits sécurisés (cf. chapitre 6 et 7), afin de limiter les attaques par canaux cachés. Ce type d'attaque est basé sur la recherche et l'exploitation de toute corrélation entre les données manipulées par un circuit et ses signaux externes. Ces signaux sont soit les signaux d'alimentation, soit les signaux d'horloge, soit les signaux électromagnétiques des circuits. Ils sont appelés signaux compromettants ou canaux cachés du fait qu'ils peuvent faire fuir des informations indésirables. Les circuits asynchrones sont donc, par construction, plus résistants à ce type d'attaque [15].

1.2.4 Modulaires

La modularité des circuits asynchrones est quasi-parfaite. Elle est due à la localité du contrôle et à l'utilisation par tous les opérateurs d'un protocole de communication bien spécifié. Il est en effet très facile de construire une fonction, et même un système complexe en associant des blocs préexistants [24]. Cette modularité aide aussi à repartir les taches de conception de blocs distincts sur différentes équipes de concepteurs. La spécification explicite du protocole aux interfaces des blocs asynchrones permet de les interconnecter par simple assemblage. Il est ainsi possible de concevoir un système flot de données, tout comme il est possible d'assembler un système composé de blocs synchrones autour d'un bus de communication. Dans le cas asynchrone, la modularité entre blocs est facilitée par deux aspects. Tout d'abord, il n'y a pas besoin de spécifier un bloc en fonction du nombre de cycles d'horloge pour obtenir et échanger des données avec un bloc distant, la synchronisation est effectuée par les données (il s'agit bien d'un modèle flot de données). Ensuite, au niveau implantation d'un circuit asynchrone, lorsqu'aucune hypothèse temporelle ne garantit le fonctionnement, la conception au niveau physique est rendue plus facile. Les phases de placement routage sont moins contraintes, sauf pour des questions de performances ou de sécurité. La correction du système est garantie quels que soient les délais dans les interconnexions aux interfaces.

A l'heure actuelle, les systèmes tout intégrés (System-On-Chip) sont composés de blocs fonctionnels utilisant des horloges définies localement. Il est donc nécessaire d'avoir recours à des systèmes de synchronisation capables de garantir des communications fiables entre blocs contrôlés par des horloges distinctes. Ce type de conception est typiquement du ressort de la conception asynchrone. Il est plus aisé de concevoir des systèmes de communication implémentés dans le style asynchrone (notamment par rapport aux problèmes de traitement indéterministes), que de concevoir des systèmes de communications synchrones multi horloges [5, 75].

Pour conclure, les propriétés de robustesse et de modularité des circuits asynchrones, en raison de leur synchronisation locale, sont particulièrement intéressantes lorsque l'on souhaite promouvoir la réutilisation de blocs dans une entreprise, ou d'un point de vue plus général, l'échange de propriétés intellectuelles (IPs) dans le cadre d'implantation de systèmes complexes.

1.3 Concepts de base des circuits asynchrones

La conception de la plupart des circuits intégrés logiques est facilitée par deux hypothèses fondamentales : les signaux manipulés sont binaires et le temps est discrétisé. La binarisation des signaux permet une implantation électrique simple et offre un cadre de conception maîtrisé grâce à l'algèbre de Boole. La discrétisation du temps permet quant à elle de s'affranchir des problèmes de rétroactions et/ou

boucles combinatoires, ainsi que des fluctuations électriques transitoires. Cependant, un système fonctionnant sans ces hypothèses peut obtenir de meilleurs résultats. Les circuits asynchrones conservent un codage discret des signaux mais ne fait pas l'hypothèse que le temps est discrétisé. Ils définissent ainsi une classe de circuits beaucoup plus large car leur contrôle peut être assuré par tout autre moyen alternatif à l'horloge unique des circuits synchrones.

1.3.1 Mode de fonctionnement asynchrone

Ce paragraphe est destiné à clarifier l'utilisation du terme « asynchrone » dans le contexte de la conception de circuits numériques. Cela nous permet d'introduire le mode de fonctionnement asynchrone et ses différences avec le mode de fonctionnement synchrone.

« Asynchrone » signifie qu'il n'existe pas de relation temporelle a priori entre des événements. Dans un système intégré, ces événements sont des événements au sens large (contrôle ou données) implantés par des signaux électriques. Il faut donc définir ce qu'est un signal « asynchrone ».

Si on prend l'exemple d'un signal d'interruption appliqué à un microprocesseur (que ce dernier soit synchrone ou asynchrone), on le qualifie de signal d'interruption asynchrone par rapport au fonctionnement du microprocesseur. Cette situation est délicate à résoudre puisqu'il faut échantillonner un signal pour mesurer son niveau sous contrôle d'un événement (par exemple l'horloge du processeur) qui n'a aucune relation temporelle avec ce signal extérieur. Dans cet exemple, le terme asynchrone qualifie une indétermination sur la relation d'ordre entre le signal d'horloge et le signal d'interruption, et donc sur le niveau de ce dernier.

Quand on parle de circuits asynchrones, on qualifie des circuits qui gèrent des signaux asynchrones entre eux mais dont le comportement est parfaitement déterminé. Par exemple, supposons deux signaux qui transportent de l'information sous forme de changement de niveau. La spécification est telle qu'il n'existe pas a priori de relation de causalité entre ces deux signaux, ces signaux sont donc asynchrones. Cependant, il est garanti qu'un événement doit se produire sur ces deux signaux. Dans ce système, il y a indétermination sur les instants d'occurrence des événements de chaque signal, mais le fait que les événements aient lieu est absolument certain. Un élément de base, très utilisé dans les circuits asynchrones, permet naturellement un rendez-vous entre deux signaux asynchrones : la porte de Muller présentée dans le paragraphe 1.3.2.2. De manière générale, on parle de synchronisation d'événements : un événement est généré en sortie si et seulement si il y a événement sur les deux entrées de la porte, quels que soient les instants d'occurrence.

Les circuits asynchrones fonctionnent donc avec la seule connaissance de l'occurrence des événe-

ments, sans connaissance de l'ordre. Le fonctionnement est identique à celui des systèmes flot de données. On peut ainsi spécifier l'enchaînement des événements sous forme d'un graphe de dépendances (réseau de Petri par exemple). L'évolution du système est garantie par l'évolution conjointe (voire concurrente) des éléments qui le compose. Chaque élément évolue avec les seules informations des éléments auxquels il est connecté. L'analogie avec le modèle des processus séquentiels communicants [50] est très forte. Dans ce modèle, les processus se synchronisent par passage de messages via des canaux de communications. Pour échanger des informations, deux éléments doivent se synchroniser, ils échangent leurs informations à travers les canaux de communications puis poursuivent leur flot d'exécution de manière indépendante. Le langage CHP, que nous utilisons comme langage de haut niveau pour décrire nos circuits asynchrones, est directement issu de ce modèle [65].

Ainsi dans les circuits asynchrones, la seule connaissance de l'occurrence des événements est suffisante pour implanter la synchronisation ; il n'est pas nécessaire d'introduire de mécanisme global d'activation du système. C'est effectivement la différence avec les circuits synchrones. Dans un système synchrone, comme nous l'avons vu, tous les éléments évoluent ensemble sur un événement du signal d'horloge : l'exécution de tous les éléments se trouve synchronisée. Ce mécanisme de synchronisation introduit une contrainte temporelle globale : afin d'obtenir un fonctionnement correct, tous les éléments doivent respecter un temps d'exécution maximum imposé par la fréquence du mécanisme d'activation. A l'opposé, les systèmes asynchrones évoluent de manière localement synchronisée, le déclenchement des actions dépend uniquement de la présence des données à traiter. Ainsi la correction fonctionnelle est indépendante de la durée de traitement des éléments du système.

1.3.2 Un contrôle local

Comme nous venons de le présenter, le point fondamental du mode de fonctionnement asynchrone est que la synchronisation et le transfert d'informations sont effectués localement. Ceci est fait par une signalisation adéquate. Le contrôle local doit remplir les fonctions suivantes : être à l'écoute des communications entrantes, déclencher le traitement local si toutes les informations sont disponibles en entrée et produire des valeurs sur les sorties. De plus, pour que l'opérateur sache qu'il est autorisé à émettre de nouvelles valeurs sur ces sorties, il doit être informé que les valeurs qu'il émet sont bien consommées (reçues) par l'opérateur en aval. Ainsi, pour permettre un fonctionnement correct indépendamment du temps, le contrôle local doit implanter une signalisation bidirectionnelle. Les communications sont dites à poignées de mains ou de type requête-acquittement (figure 1.1).

Toute action doit être acquittée par le récepteur afin que l'émetteur puisse émettre de nouveau. Cette signalisation bidirectionnelle offre un mécanisme qui permet de garantir la synchronisation et la causalité des événements au niveau local et donc la correction fonctionnelle du système dans son ensemble.

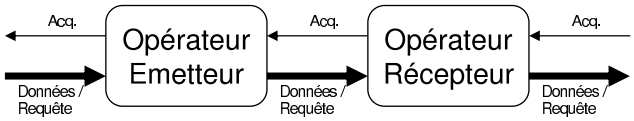


FIGURE 1.1 – Communication de type requête-acquittement entre opérateurs asynchrones.

1.3.2.1 Protocoles de communication

Pour implanter une signalisation bidirectionnelle, deux protocoles de communication sont couramment utilisés : le protocole deux phases, encore appelé NRZ (Non Retour à Zéro) ou « half-handshake » et le protocole quatre phases, encore appelé RZ (Retour à Zéro) ou « full-handshake ». Ces deux protocoles sont représentés respectivement figure 1.2 et figure 1.3. Dans les deux cas, il faut noter que tout événement sur un signal de l'émetteur est acquitté par un événement sur un signal du récepteur, et vice-versa. Ce mécanisme permet de s'assurer de l'insensibilité au temps de traitement de l'opérateur. Dans ce protocole, seul importe l'occurrence des événements, localement, entre l'émetteur et le récepteur, et non leurs temps relatifs, ni leurs ordres respectifs par rapport à l'entrée de l'émetteur ou la sortie du récepteur. Le choix du protocole de communication affecte les caractéristiques de l'implantation du circuit (surface, vitesse, consommation, robustesse, etc.).

Le protocole deux phases constitue la séquence d'échange d'information minimale nécessaire à une communication : les données sont représentées par des fronts (montants et descendants).

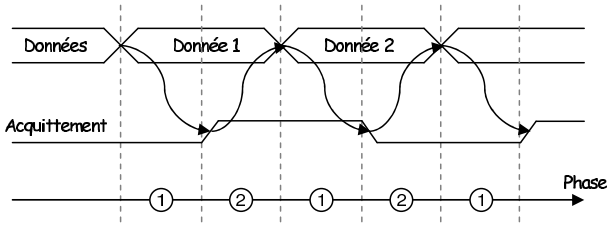


FIGURE 1.2 – Protocole deux phases.

Les deux phases sont les suivantes :

- Phase 1 : c'est la phase active du récepteur qui détecte la présence d'une donnée, effectue le traitement et génère le signal d'acquittement.
- Phase 2 : c'est la phase active de l'émetteur qui détecte le signal d'acquittement et émet une nouvelle donnée si elle est disponible.

Ce protocole, malgré son apparente efficacité, n'est que peu utilisé en raison des difficultés rencontrées dans la logique nécessaire à son implantation. L'asymétrie entre les deux phases successives (montée ou descente de l'acquittement) se révèle un obstacle important à la réalisation de ce protocole.

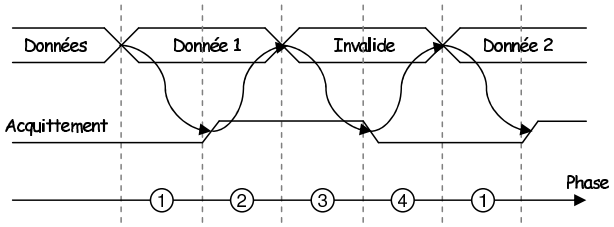


FIGURE 1.3 – Protocole quatre phases.

Les différentes phases du protocole quatre phases sont les suivantes :

– Phase 1 : le récepteur détecte une nouvelle donnée, effectue le traitement et génère le signal d'acquittement.

– Phase 2 : l'émetteur détecte le signal d'acquittement et invalide les données.

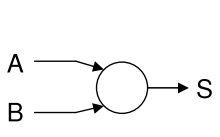
– Phase 3 : le récepteur détecte l'état invalide et désactive le signal d'acquittement.

– Phase 4 : l'émetteur détecte l'état invalide du signal d'acquittement et émet une nouvelle donnée si elle est disponible.

A l'opposé du protocole deux phases, le protocole quatre phases est le plus utilisé en raison de la symétrie de ses phases. En doublant leur nombre, l'état du système est invariablement le même au début et à la fin d'une communication.

1.3.2.2 Implantation du protocole : la porte de Muller

Pour implanter de tels protocoles, les portes logiques élémentaires ne suffisent pas. Ce paragraphe présente rapidement le fonctionnement des portes de Muller (autrement appelées « C element »), proposées par Muller dans [68]. Elles sont essentielles pour l'implantation de circuits asynchrones : elles réalisent le rendez-vous entre plusieurs signaux. La sortie copie la valeur des entrées lorsque celles-ci sont identiques, sinon elle mémorise la dernière valeur calculée. La figure 1.4 représente le symbole d'une porte de Muller à deux entrées et sa spécification sous forme de table de vérité.



B \ A	0	1
0	0	S^{-1}
1	S^{-1}	1

FIGURE 1.4 – Symbole et spécification de la porte de Muller.

A partir de cette spécification, la figure 1.5 décrit plusieurs implantations possibles : sous forme de portes logiques élémentaires et sous forme de transistors.

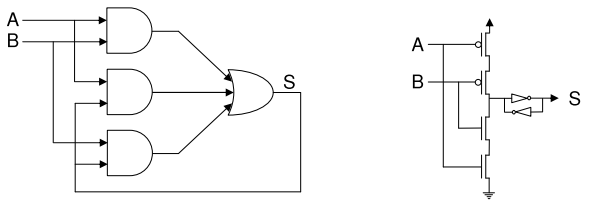


FIGURE 1.5 – Implantations de la porte de Muller.

Il existe des variantes de la porte de Muller [79]. La porte de Muller généralisée (traduit de l'anglais « generalized C element ») [62] est une porte de Muller dans laquelle les signaux qui font monter la sortie à 1 ne sont pas toujours les mêmes que ceux qui la font descendre à 0. Dans ce cas, la porte de Muller est également dite dissymétrique. A titre d'exemple, la figure 1.6 illustre une porte de Muller généralisée (son symbole, sa spécification et sa réalisation au niveau transistors). Dans cet exemple, il suffit que les entrées B et C aient la valeur 1 pour que la sortie passe à 1 et que les entrées A et B aient la valeur 0 pour que la sortie passe à 0. Sinon la sortie est mémorisée.

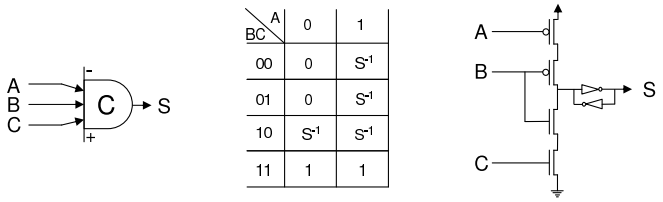


FIGURE 1.6 – Porte de Muller dissymétrique.

1.3.2.3 Codage des données

Comme nous venons de le voir, les protocoles de communication doivent détecter la présence d'une donnée sur leur entrée et leur disponibilité. Pour résoudre ce problème, il est nécessaire d'adopter un codage particulier pour les données. Il est en effet impossible d'utiliser un fil unique par bit de donnée : cela ne permet pas de détecter que la nouvelle donnée prend un état identique que la valeur précédente. Deux types de solutions sont possibles, soit la création d'un signal de requête associé aux données, soit l'utilisation d'un codage insensible aux délais bifilaire ou double-rail par bit de donnée [77].

1.3.2.3.1 Codage « données groupées » La façon la plus évidente de caractériser la validité des données consiste simplement à ajouter un fil au bus de données afin de spécifier si les données y sont valides (figure 1.7a). Les bus de données utilisent le traditionnel schéma de la logique synchrone : un fil par bit de données, ce qui s'appelle parfois mono-rail (« single rail ») dans la littérature. Le fil spécifiant la validité des données, dit signal de requête, est typiquement implanté avec le retard adéquat. Ce retard est conçu égal ou supérieur au temps de calcul dans le pire cas.

Efficace en terme de surface (en terme de nombre de fils et donc en nombre de portes pilotant ces fils),

ce type de codage permet une bonne réalisation des circuits asynchrones. Cependant, les inconvénients de ce codage dans certains circuits sont qu'en utilisant le retard assorti, le fonctionnement est fixé au pire cas : le fonctionnement ne dépend donc pas de la propagation réelle des données à l'intérieur d'un étage, et les circuits obtenus ne sont plus insensibles aux délais.

1.3.2.3.2 Codage insensible aux délais Contrairement au codage « données groupées » où les données et leur validité sont totalement séparées, une autre approche plus complexe consiste à intégrer l'information de la validité dans les données. Cette approche possède la propriété suivante : les données sont détectées à l'arrivée sans que cela ne repose sur aucune hypothèse temporelle. Ceci implique donc robustesse, portabilité et facilité lors de la conception.

- Codage 4 états : dans ce codage, chaque bit de donnée est représenté par deux fils. Parmi les quatre états possibles pour ces 2 fils, la moitié est réservée à la valeur 0, l'autre à la valeur 1 (figure 1.7b). L'émission d'une nouvelle donnée se traduit par le changement d'un seul fil : celui de droite pour exprimer la même valeur que précédemment, celui de gauche pour exprimer la valeur opposée. La validité des données est donc assurée par le changement du couple de fils.
- Codage 3 états : dans ce codage également, chaque bit est représenté par 2 fils. En revanche, les valeurs représentées ne sont pas dupliquées : une seule combinaison représente chaque valeur, tandis que la troisième indique l'état invalide et la quatrième reste inutilisée (figure 1.7c). Ainsi, comme on le voit sur le schéma, le passage d'une valeur valide à l'autre se traduit nécessairement par le passage par l'état invalide.

Parmi tous ces codages, le codage 3 états, qui peut par ailleurs sembler le moins naturel, est aujourd'hui de loin le plus utilisé pour des raisons d'implantations et de sécurité. En effet les deux autres posent des problèmes en terme de logique additionnelle : dans le codage « données groupées », il faut recombiner les données avec le signal de validité et dans le codage 4 états, un étage de logique supplémentaire est nécessaire pour tester la parité du couple de fils.

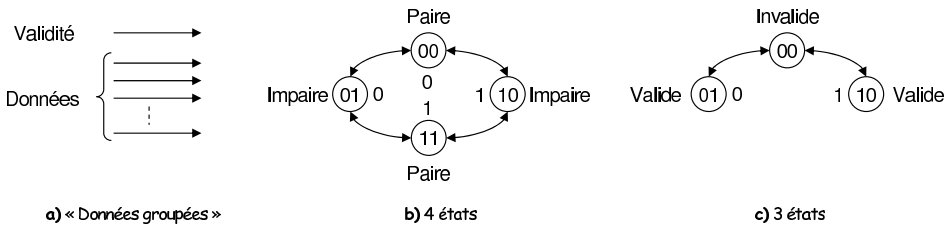


FIGURE 1.7 – Codages les plus utilisés.

Enfin, le codage 3 états, souvent appelé codage « double-rail » – en raison des deux fils par bit de données – est un cas particulier du codage « one hot ». En effet, il est possible d'étendre la représentation,

pour un digit en base N, N fils seront utilisés : chaque fil représente une valeur et l'état invalide est déterminé par la remise à zéro de tous les fils. Les combinaisons où au moins deux fils sont à 1 sont interdites. Ce codage est également appelé « 1 parmi N » ou encore « multi-rail » dans ce manuscrit. Une dernière extension possible est le codage « M parmi N ».

1.4 Méthodes de conception

1.4.1 Conception de circuits asynchrones : le problème des aléas

Dans le sens le plus général, un aléa est une activité non désirée (« glitch ») en réponse à un changement sur certaines entrées. Un aléa peut se produire en raison de la différence de délais entre les portes. La présence d'aléas dans un circuit – notamment pour un circuit asynchrone, très sensible à ces transitions – entraîne des fautes de fonctionnement. Il est possible de détecter la présence d'aléas à différentes étapes de la conception. Un séquenceur peut, par exemple, contenir des aléas fonctionnels qui trouvent leur origine dans la spécification de la fonction elle-même. Si ce problème est réglé au niveau de la spécification, il peut encore apparaître des aléas au niveau de l'implantation. Cela signifie que la conception d'un circuit ne s'arrête pas après la spécification. Il faut encore s'assurer que les techniques de réalisation choisies sont exemptes d'aléas.

Dans le cadre de la conception de circuits asynchrones, il est important de remarquer que ces circuits requièrent une conception attentive et fine de toutes les parties (combinatoires et séquentielles) du circuit, de la spécification jusqu'à l'implantation matérielle. Pour la partie combinatoire, il existe deux classes de base pour les aléas : aléas logiques et aléas fonctionnels. Chaque classe d'aléas comprend les aléas statiques et les aléas dynamiques. Nous allons présenter les différents types d'aléas dont tout circuit asynchrone ne faisant aucune hypothèse temporelle doit se prémunir.

1.4.1.1 Aléas combinatoires fonctionnels

Aléa fonctionnel statique ([19]) : f est une fonction logique booléenne possédant un aléa fonctionnel statique pour une transition des entrées du monôme A au monôme C avec $A, C \in \{0,1\}^n$, si et seulement si :

- $f(A) = f(C)$,
- il existe un état des entrées (monôme) $B \in [A, C]$ tel que $f(A) \neq f(B)$.

Aléa fonctionnel dynamique ([19]) : f est une fonction logique booléenne possédant un aléa fonctionnel dynamique pour une transition des entrées du monôme A au monôme D si et seulement si :

- $f(A) \neq f(D)$,

– il existe une paire d'états des entrées B et C ($A \neq B$, $C \neq D$) tel que $B \in [A, D]$ et $C \in [B, D]$ et $f(A) = f(C)$ et $f(B) = f(D)$.

Ainsi, les aléas combinatoires fonctionnels sont la propriété de la fonction logique. Ils peuvent être uniquement détectés et supprimés à un niveau plus élevé de la conception en étudiant et en modifiant la spécification logique de la fonction. Ces aléas ne dépendent pas de l'implantation et de la distribution des délais. Il faut une spécification exempte d'aléas fonctionnels pour obtenir une réalisation sans aléa si aucune hypothèse n'est faite sur les délais des éléments.

Il est bien connu que si une transition possède un aléa fonctionnel, aucune implantation de la fonction n'est assurée pour éviter les aléas dans cette transition, quel que soit le modèle de délais des portes et des fils ([35, 19]). Par conséquent, dans le reste de cette thèse, les transitions des entrées sont supposées exemptes d'aléas fonctionnels.

1.4.1.2 Aléas combinatoires logiques

Les aléas combinatoires logiques sont la propriété de l'implantation de la fonction logique. Leur apparition dépend de la distribution des délais dans la logique. [19] indique que si l'implantation d'une fonction logique contient des aléas fonctionnels, pour une transition des entrées donnée, alors elle ne peut pas avoir d'aléa logique pour la même transition. Cependant, si l'implantation est exempte d'aléas fonctionnels, elle peut contenir des aléas logiques.

Dans cette classe d'aléas, il existe deux types principaux d'aléas ([92]) :

– Aléas de type SIC (de l'anglais « Single Input Change ») : comme son nom l'indique, ce type d'aléas se produit au cours de la transition d'une seule entrée.
– Aléas de type MIC (de l'anglais « Multi-Input Change ») : ce sont les aléas logiques rencontrés lorsque plusieurs entrées changent.

Comme les aléas fonctionnels, les aléas logiques comprennent également les aléas statiques et dynamiques.

Aléa logique statique ([19]) : une implantation combinatoire de la fonction logique f contient un aléa logique statique pour une transition des entrées du monôme A au monôme B si et seulement si :

– $f(A) = f(B)$,
– pour un certain modèle de délais, la sortie du circuit ne reste pas monotone au cours de la transition.

Cet aléa est aussi appelé « spike » dans le jargon du concepteur. Par exemple, toute porte « AND » et « OR » est susceptible de générer ce type d'aléas si deux entrées changent simultanément (cf. figure 1.8).

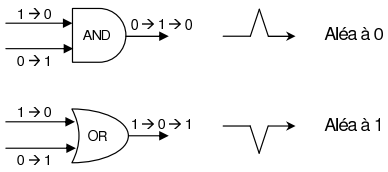


FIGURE 1.8 – Aléa statique sur des portes « AND » et « OR ».

Les aléas statiques apparaissent quand une transition n'est pas complètement couverte par une porte, en d'autres termes, chaque fois que l'implantation ne contient pas une porte qui maintient la valeur de la sortie lors de la transition.

A titre d'exemple, dans la figure 1.9, la transition de A ($xy\overline{z}$) à B (xyz) n'est pas couverte par une seule porte dans l'implantation. Il est alors possible, avec un certain modèle de délais, que les deux portes « AND » descendent momentanément, ce qui fait passer la sortie f à 0 (aléa statique 1). Cet aléa peut être éliminé en complétant la fonction f par une porte « AND » avec ses entrées xy . Cette porte maintient la sortie à 1 pendant la transition.

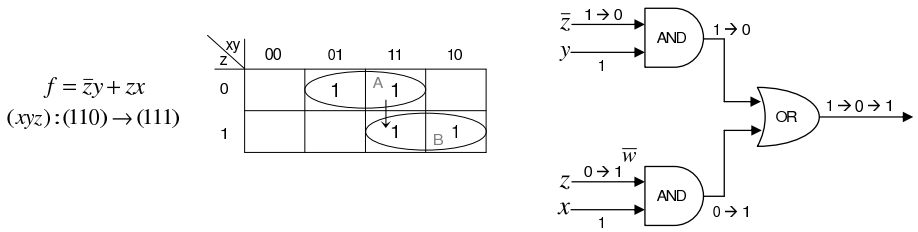


FIGURE 1.9 – Aléa logique statique SIC.

La figure 1.10 illustre un aléa logique statique de type MIC. Lors de la transition de A à B, il n'y a aucune porte qui maintient la sortie au niveau 1. Ainsi, si les portes $\overline{wx}$ et $\overline{y}$z sont suffisamment rapides et si les portes xz et $\overline{w}$y sont suffisamment lentes, alors la sortie peut momentanément prendre la valeur 0 pendant la transition (aléa statique 1).

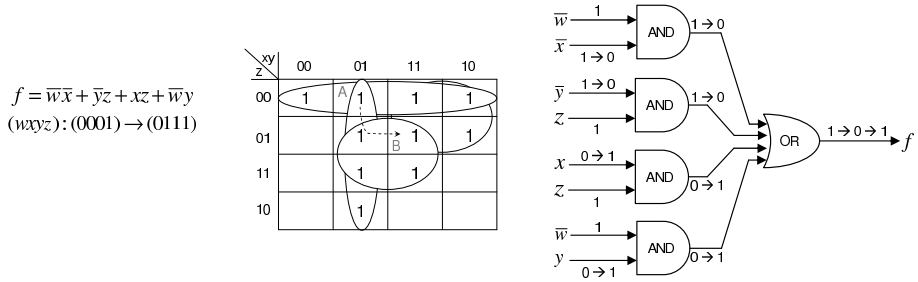


FIGURE 1.10 – Aléa logique statique MIC.

En conclusion, il a été montré dans [93] que pour une implantation quelconque sous forme SOP

(« Sum Of Product » ou somme de produits), aucun aléa ne peut survenir lors des transitions 0→0, 0→1, 1→0 sur la sortie. Ainsi, la synthèse sans aléa de circuits SOP nécessite seulement d'éliminer les aléas statiques 1. Par ailleurs, la condition nécessaire et suffisante pour éviter les aléas logiques statiques de type MIC pour les implantations SOP à deux niveaux a été démontrée dans [35]. Pour les implantations SOP multi-niveaux, les conditions sont plus complexes et sont discutées dans [18].

Aléa logique dynamique ([19]) : une implantation logique de la fonction logique f contient un aléa logique dynamique pour une transition des entrées du monôme A au monôme B si et seulement si :

- $f(A) \neq f(B)$,
- pour un certain modèle de délais, la sortie du circuit ne reste pas monotone au cours de la transition.

La figure 1.11 illustre un signal passant à 1 (respectivement 0) qui subit un aléa dynamique à la montée (respectivement la descente).

FIGURE 1.11 – Aléas dynamiques.

Les aléas logiques dynamiques s'appliquent aux deux types SIC et MIC. Dans le cas SIC, cet aléa correspond à la situation où un littéral et son complément se déploient dans plusieurs chemins de données. Dans le cas MIC, un aléa dynamique apparaît quand une porte commute momentanément pendant une transition. A titre d'exemple, un aléa logique dynamique de type MIC est représenté figure 1.12. Dans cet exemple, la transition de C à A peut provoquer un aléa logique dynamique.

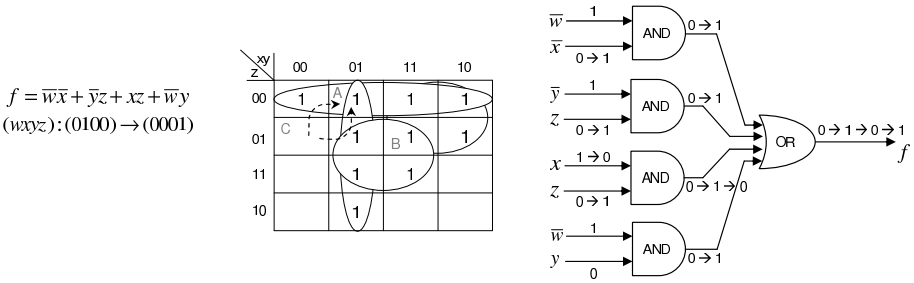


FIGURE 1.12 – Aléa logique dynamique : MIC.

Pour conclure, l'implantation est donc de toute première importance en asynchrone et nécessite la conception d'outils qui intègrent la notion d'aléas. En particulier, comme nous le verrons dans ce manuscrit, l'étape de projection technologique est spécifique aux circuits asynchrones. L'implantation d'une fonction logique à l'aide de portes logiques de base doit être contrôlée pour ne pas générer d'aléas combinatoires logiques.

La suppression d'aléas est donc un problème de couverture de tableau de Karnaugh :

- Les transitions 1→1 doivent être couvertes.
- Pour les transitions 1→0 et 0→1, un produit qui intersecte avec la transition doit contenir le monôme de départ et le monôme d'arrivée.
- Les transitions 0→0 ne génèrent pas d'aléas dans les réalisations SOP.

Ces conditions suffisent pour éliminer n'importe quel aléa de type MIC. Si il existe plusieurs aléas de type MIC, le problème de couverture peut très bien ne pas avoir de solution. Pour un ensemble de transitions MIC (fonctions de transition d'une machine à états) l'existence d'une couverture de tableau de Karnaugh conduisant à une réalisation SOP sans aléa n'est pas assurée.

1.4.1.3 Aléas séquentiels

Ces aléas trouvent leur origine dans les signaux bouclés. Ces aléas sont détectables lors de la spécification d'un problème, par exemple lors de l'étude d'un tableau de flots (« flow table » [92]). On cite souvent les aléas séquentiels qui peuvent être détectés en faisant changer la même entrée une fois puis trois fois. Il y a un aléa séquentiel si l'état final n'est pas le même dans les deux cas.

1.4.2 Classification des circuits asynchrones

Les circuits asynchrones sont communément classifiés suivant le modèle de circuit et d'environnement adopté. La figure 1.13 présente la terminologie habituellement utilisée pour qualifier les circuits asynchrones.

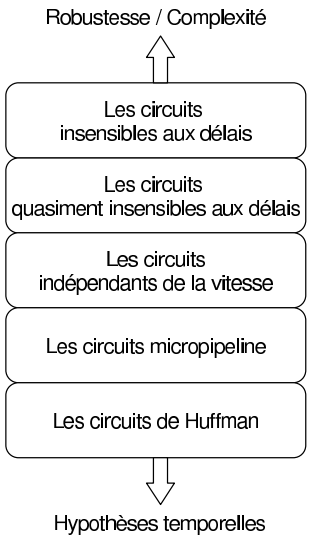


FIGURE 1.13 – Classification des circuits asynchrones.

Plus le fonctionnement du circuit respecte fondamentalement la notion d'insensibilité aux délais, plus le circuit est robuste et plus il est complexe. Dans la suite, nous présentons brièvement ces catégories de circuits en commençant par les circuits qui respectent fondamentalement la notion d'asynchronisme, puis des circuits dans lesquels sont introduites des hypothèses temporelles de plus en plus fortes. Cette approche va donc des circuits insensibles aux délais vers des circuits s'approchant du synchrone.

1.4.2.1 Les circuits insensibles aux délais

Les circuits de ce type utilisent un modèle de délai non borné dans les fils et les portes. Aucune hypothèse temporelle n'est introduite lors de la conception de ce type de circuit, cela signifie qu'ils sont fonctionnellement corrects quels que soient les délais introduits dans les éléments logiques et les portes. Basés sur des travaux de Clark Wasley, et re-formalisés dans [91], les circuits DI sont supposés répondre toujours correctement à une sollicitation externe pourvu qu'ils aient assez de temps de calcul. Ceci impose donc que chaque récepteur d'un signal informe toujours l'expéditeur que l'information a été reçue. Les contraintes de réalisation pratique imposées par ce modèle sont très fortes. De plus la plupart des circuits conçus aujourd'hui utilisent des portes logiques qui ne possèdent qu'une seule sortie. L'adoption du modèle de délai « non borné » ne permet par leur utilisation. En effet, les portes logiques standard ont leurs sorties qui peuvent changer si une seule de leurs entrées change. Comme nous l'avons souligné, toutes les composantes de ce type de circuits doivent s'assurer du changement des entrées pour produire une sortie. Si la sortie change alors qu'une seule des entrées change, seul le changement de l'entrée active est acquitté, sans pouvoir tester l'activité des autres entrées. La seule porte qui respecte cette règle est la porte de Muller (cf. 1.3.2.2). Malheureusement, les fonctions réalisables avec seulement des portes de Muller sont très limitées [65]. La seule solution est d'avoir recours à un modèle de circuit de type « portes complexes » pour les composants élémentaires. Dans ce cas, la construction de ces circuits se fait à partir de composants standard plus complexes que de simples portes logiques et qui peuvent posséder plusieurs entrées et plusieurs sorties.

1.4.2.2 Circuits quasi-insensibles aux délais (QDI)

Comme son nom l'indique, la classe des circuits QDI est un sur-ensemble de la classe DI. Cette classe adopte le même modèle de délai « non borné » pour les connexions et les portes mais ajoute la notion de fourche isochrone (« isochronic fork ») [65, 7].

Une fourche est un fil qui connecte un émetteur unique à au moins deux récepteurs. Elle est qualifiée d'isochrone lorsque les délais entre l'émetteur et les récepteurs sont identiques. Cette hypothèse a des conséquences importantes sur le modèle et les réalisations possibles. Elle résout notamment le problème de l'utilisation de portes logiques à une seule sortie, causé par la classe DI. Car si les fourches sont isochrones, il est possible de ne tester qu'une seule branche d'une fourche et de supposer que le signal s'est

propagé de la même façon dans les autres branches. En conséquence, on peut autoriser l'acquittement d'une des branches de la fourche isochrone. Alain Martin a montré dans [63] que l'hypothèse temporelle de fourche isochrone est la plus faible à ajouter aux circuits DI pour les rendre réalisables à partir de portes à plusieurs entrées et une seule sortie. Les circuits QDI sont donc réalisables avec les mêmes portes que celles utilisées pour la conception de circuits synchrones.

Finalement, dans les circuits QDI, les signaux peuvent mettre un temps arbitraire à se propager dans les portes ainsi que dans les connexions (avec l'hypothèse de fourche isochrone) sans que cela ne perturbe la correction fonctionnelle du circuit. De telles réalisations sont bien entendu très délicates à mettre en œuvre étant donné les contraintes imposées à la conception. En contrepartie, ces circuits présentent un degré de robustesse quasi parfait. En théorie, aucune erreur inhérente au circuit ne peut se produire. En pratique, la contrainte de fourche isochrone est assez faible et est facilement remplie par une conception soignée, en particulier au niveau du routage et des seuils de commutation. Il suffit, finalement, que la dispersion des temps de propagation jusqu'aux extrémités de la fourche soit inferieure aux délais des opérateurs qui lui sont connectés (au minimum une porte et un fil dont une sortie interagit avec la fourche [65, 7]).

1.4.2.3 Circuits indépendants de la vitesse (SI)

Les circuits SI (« Speed Independent ») font l'hypothèse que les délais dans les fils sont négligeables, tout en conservant un modèle « non borné » pour les délais dans les portes. Ce modèle qui n'est pas réaliste dans les technologies actuelles, est en pratique équivalent au modèle QDI à l'exception des fourches qui sont toutes considérées comme isochrones. Même si pendant longtemps, la communauté a tenté de cerner les différences entre ces deux modèles, il y aujourd'hui un consensus pour considérer les modèles QDI et SI équivalents. Dans [48], on trouve un schéma montrant comment une fourche isochrone peut être représentée par un circuit SI (figure 1.14).

Cependant, il semble plus pertinent d'utiliser le modèle QDI car celui-ci différencie les fourches isochrones de celles qui ne le sont pas. Ceci offre le moyen de vérifier les connexions du circuit qui ne sont pas insensibles aux délais au cours des différentes étapes de conception, et seulement celles-ci.

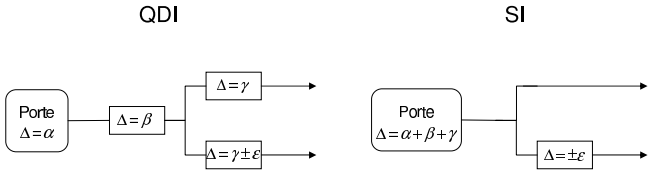


FIGURE 1.14 – Équivalence entre les modèles SI et QDI.

1.4.2.4 Les circuits micropipeline

La technologie micropipeline a été initialement introduite par Ivan Sutherland [89]. Il faut en fait considérer que les circuits micropipelines correspondent plutôt à une classe d'architecture de circuits asynchrones, que directement à un modèle de délais de circuits asynchrones. Les circuits de cette classe sont composés de parties de contrôle insensibles aux délais qui commandent des chemins de données conçus en utilisant un modèle de délais bornés. La structure de base de cette classe de circuits est le contrôle d'une file (FIFO). Elle se compose de portes de Muller connectées tête bèche.

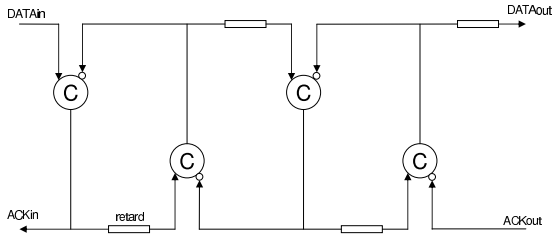


FIGURE 1.15 – Structure de base des circuits micropipelines.

Cette structure implémente un protocole deux phases. Le circuit réagit à des transitions de signaux et non pas d'états. On parle également d'une logique à événements : chaque transition étant associée à un événement. Ainsi, tous les signaux sont supposés à zéro initialement. Une transition positive sur DATAin provoque une transition positive sur ACKin qui se propage également à l'étage suivant. Le deuxième étage produit une transition positive qui d'une part, se propage à l'étage suivant mais qui d'autre part, revient au premier étage l'autorisant à traiter une transition négative cette fois. Les transitions de signaux se propagent donc dans la structure tant qu'elles ne rencontrent pas une cellule occupée. Ce fonctionnement de type FIFO est bien insensible aux délais.

Cette structure est une structure de contrôle simple. Elle peut être utilisée pour contrôler un chemin de données possédant des opérateurs de mémorisation et des opérateurs de traitements combinatoires. Dans ce cas, le codage des données et le protocole associé sont du type deux phases « données groupées ».

La motivation première pour le développement de cette classe de circuits était de permettre un pipeline élastique. En effet, le nombre de données présentes dans le circuit peut être variable, les données progressant dans le circuit aussi loin que possible en fonction du nombre d'étages disponibles ou vides. Cependant ce type de circuits révèle un certain nombre d'inconvénients. Tout d'abord, il faut remarquer que les problèmes d'aléas ont été écartés en ajoutant des retards sur les signaux de contrôle. Cela permet en fait de se ramener à un fonctionnement en temps discret dans lequel la mémorisation des données est autorisé seulement lorsqu'elles sont stables (à la sortie des portes combinatoires).

De nombreuses méthodes de conception s'attachent à concevoir des circuits micropipelines avec des modèles de délais plus ou moins différents. L'idée générale de ces approches est de séparer dans la spécification le contrôle des données et de les implanter séparément dans des styles différents. En particulier, les contrôleurs des registres de pipeline peuvent être obtenus avec un grand nombre de méthodes et avec des hypothèses de délais plus ou moins fortes.

1.4.2.5 Circuits de Huffman

Cette catégorie regroupe des circuits qui utilisent un modèle de délais identique aux circuits synchrones. Ils supposent que les délais dans tous les éléments du circuit et dans les connexions, sont bornés ou même de valeurs connues. Les hypothèses temporelles sont donc du même ordre que pour la conception de circuits synchrones. Leur conception repose sur l'analyse des délais dans tous les chemins et les boucles de façon à dimensionner les temps d'occurrence des signaux de contrôles locaux. Ces circuits sont d'autant plus difficiles à concevoir et à caractériser qu'une faute de conception ou de délai les rend totalement non fonctionnels.

1.5 Conclusion

Les problèmes liés à l'horloge dans la conception de circuits synchrones de plus en plus complexes, permettent d'ouvrir une voie vers la nouvelle approche de la conception de circuits asynchrones. Contrairement à la synchronisation globale utilisée dans les circuits synchrones, la synchronisation dans les circuits asynchrones est effectuée localement grâce à une signalisation bidirectionnelle entre tous les éléments du circuit. Cette signalisation est implémentée via des canaux de communications utilisant un protocole et un codage de données spécifiques. De nombreux types de circuits asynchrones existent : ils sont classés en fonction du modèle de délais pour les portes et les fils.

Comme nous l'avons vu, les circuits quasi-insensibles aux délais sont des circuits asynchrones très robustes. Pour ces circuits, l'hypothèse temporelle de fourches isochrones permet l'utilisation de portes élémentaires à une seule sortie lors de leur conception. C'est pour ces raisons que ce type de circuits est la cible de nos outils de conception présentés dans cette thèse.

Chapitre 2

Les outils de conception existants pour les circuits asynchrones

2.1 Introduction

Un des principaux obstacles au développement des circuits asynchrones dans l'industrie est le manque d'outils de conception automatiques pour ce type de circuits. Cependant, il existe aujourd'hui quelques méthodes et outils de synthèse pour les principaux types de circuits asynchrones. Peu de ces méthodes et outils sont commercialisés. Ce chapitre ne se veut pas exhaustif sur l'étude de l'ensemble des méthodologies et outils de synthèse de circuits asynchrones (cela serait beaucoup trop long), mais se veut un reflet des principales méthodes existantes, ainsi que des outils implémentant ces méthodes.

2.2 Méthode Haste

Philips travaillait depuis longtemps sur le développement de réalisations asynchrones basées sur un système propriétaire Tangram [8], dont les succès étaient bien connus dans la communauté asynchrone [9, 10, 44, 52]. En 2000-2001, la firme Handshake Solutions fut créée depuis l'incubateur de Philips. Cette firme développe des circuits asynchrones de type micropipeline à partir du langage Tangram, dont le nom s'est transformé en Haste. Haste est en fait un ensemble comprenant un langage de description de haut niveau, également appelé Haste, et un compilateur associé qui permet de traduire les structures du langage en des structures de composants de type « poignée de mains » (cf. figure 2.1).

Haste est un langage de description de haut niveau, dont la syntaxe ressemble aux langages de programmation traditionnels tels que C ou Pascal. Utilisant les concepts de CSP, Haste sert à modéliser des processus concurrents communicants par passages de messages synchrones via des canaux de communication point à point.

Haste utilise une compilation dirigée par la syntaxe. Cette méthode de synthèse par traduction commence par une spécification du circuit sous forme d'un programme dans un langage de description à

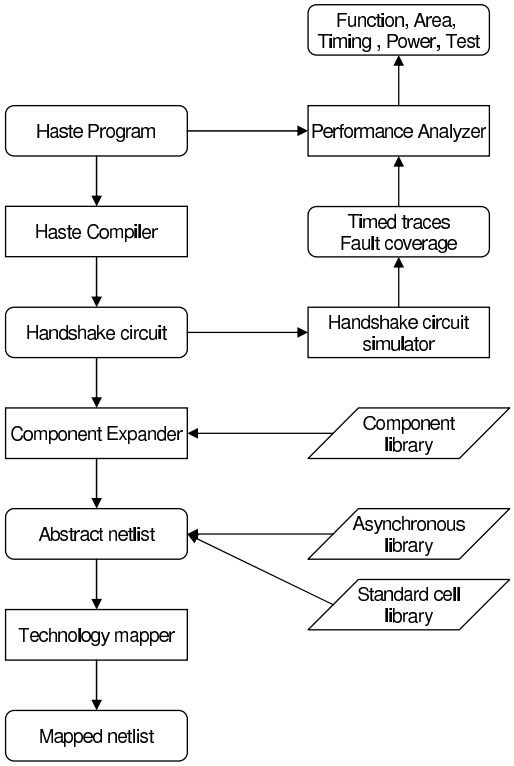


FIGURE 2.1 – Méthode Haste.

base de processus concurrents communicants. Ensuite, les structures du langage sont directement traduites en blocs de circuit. En plus de sa simplicité et de sa transparence, l'avantage de cette approche est sa capacité à décrire élégamment à haut niveau des systèmes concurrents, complexes et hiérarchiques et donc d'éviter le problème d'explosion d'états en traduisant des structures de langages directement dans des blocs de circuits fixes. Cependant, les circuits générés par cette méthode peuvent être inefficaces et redondants puisque les optimisations sont difficiles à appliquer avec une synthèse dirigée par la syntaxe. En effet, comme cette approche repose sur des transformations locales, elle manque de flexibilité en vue d'une optimisation globale. De plus, il n'existe pas de place pour la créativité : une conception pauvre peut facilement donner le même résultat qu'une conception élégante. Il faut que le concepteur ait une bonne connaissance de la conception des circuits asynchrones et ceci même s'il est bon concepteur dans le domaine synchrone. Finalement, il est important de noter que la synthèse dirigée par la syntaxe préserve la correction par construction, mais n'assure pas que le circuit global soit correct. Il est possible que le circuit présente un blocage (« deadlock ») par exemple.

La méthode de synthèse Haste utilise un format intermédiaire basé sur les circuits de type « poignée de mains » [8]. Les circuits de type « poignée de mains » sont ceux qui communiquent avec les autres circuits par des canaux directs utilisant le protocole de donnée 4 phases « données groupées ». Grâce à la compilation dirigée par la syntaxe, un circuit asynchrone décrit en langage Haste est traduit de façon directe en une structure de composants « poignée de mains ».

Le back-end du flot de conception implique de disposer d'une bibliothèque de circuits de type « poignée de mains » que le compilateur utilise comme cible et aussi certains outils comme un analyseur de performance et un simulateur fonctionnel. De nombreuses bibliothèques de circuits existent, ce qui permet d'essayer des implantations utilisant différents protocoles (4 phases « données groupées », 4 phases double-rails) ainsi que différentes technologies cibles (cellules standard CMOS, FPGA). Les circuits de type « poignée de mains » dans les bibliothèques peuvent être spécifiés et conçus de plusieurs façons : manuellement, en utilisant les méthodes par STG (§2.6), ou encore en utilisant les étapes de la méthode de Caltech (§2.4).

Comme nous l'avons vu, il est difficile d'optimiser un circuit obtenu à partir d'une telle approche de compilation. L'optimisation dans la méthode Haste se fait au niveau primaire : il s'agit de remplacer les structures de composants « poignée de mains » par des équivalents plus efficaces.

2.3 Méthode Balsa

Cette méthode est développée par l'université de Manchester. Ressemblant à la méthode Haste, Balsa – étant un cadre pour la conception de circuits asynchrones et un langage de description pour ce type de circuits – adopte une approche de la compilation dirigée par la syntaxe : les structures de langage sont mappées directement sur des composants communicants de type « poignée de mains ». Le flot de conception de cette approche est donné figure 2.2. Comme Haste, le langage Balsa [2, 1] fait partie de la famille des langages de programmation concurrents. Il se base sur la communication synchronisée par canaux et le style de description parallèle de CSP.

Le format intermédiaire utilisé principalement dans le flot Balsa est le format Breeze qui définit le réseau de circuits « poignée de mains ». Breeze sert également, aux outils finaux, à fournir des implantations pour les spécifications Balsa, ce qui permet d'avoir des outils finaux indépendants des outils frontaux. Dans le système Balsa, la simulation fonctionnelle est effectuée par LARD [36], un simulateur développé dans le cadre du projet Amulet. La simulation après synthèse est réalisée grâce à des outils commerciaux.

Bien que le succès de cette méthode de conception ait été illustré par la conception du microprocesseur Amulet3i, il apparaît que, comme les autres méthodes basées sur la synthèse dirigée par la syntaxe, l'implantation des circuits n'est pas efficace.

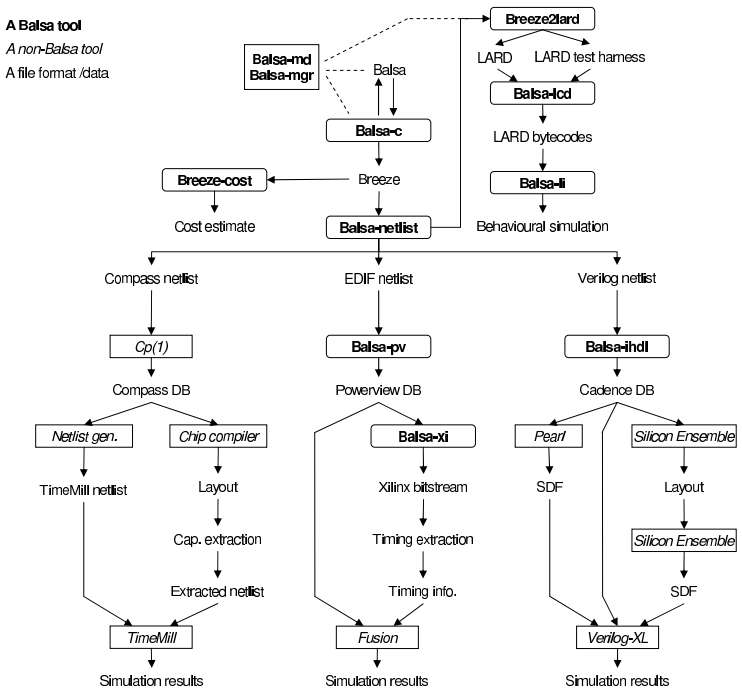


FIGURE 2.2 – Méthode Balsa.

2.4 Méthode Caltech

Universellement reconnue dans la communauté asynchrone comme étant l'une des voies de développement les plus probantes, cette méthode repose aussi sur une description de haut niveau sous forme de processus concurrents communicants. Basée sur le langage CSP ([50]), cette modélisation CHP [63] garantie dès le début du développement (« top ») jusqu'à la fin (« down ») la préservation des clauses d'insensibilité aux délais imposées par le modèle : les processus dialoguent entre eux sans jamais faire d'hypothèse concernant la propagation des signaux le long des canaux. Le langage CHP a été principalement développé pour décrire un circuit correct (en termes d'hypothèses temporelles autant que de fonction implantée) et préserver cette correction tout au long des transformations appliquées. La nécessité de modéliser l'ensemble des contraintes depuis le plus haut niveau à toutes les étapes du développement étant maintenant une certitude, CHP s'est imposé comme un langage idéal : rigoureux et complet.

Le principe de la méthode repose sur la modélisation de chaque processus en termes de communication : au plus haut niveau, le protocole est implicite, seules les actions de communication (lecture, écriture) apparaissent. Le développement du code permet de faire intervenir différents types de protocole, de codage et de conventions (choix du processus actif sur le canal, séquencement de la communication) pour aboutir, par étapes successives, à une description explicite des signaux. Sophistiquées et complexes, les étapes de développement préservent la sémantique mais sont pour la plupart effectuées manuellement (figure 2.3).

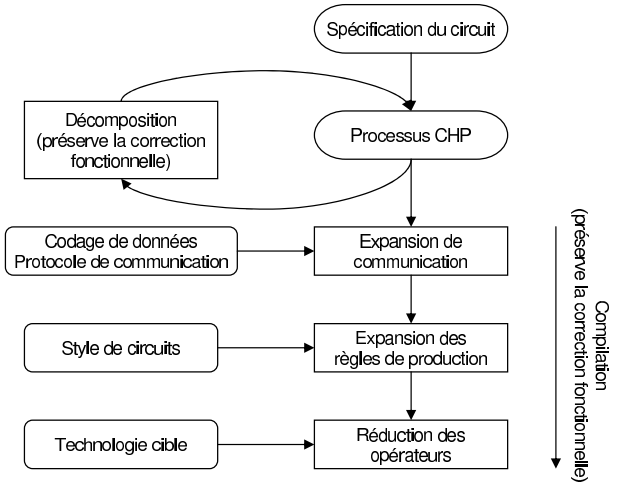


FIGURE 2.3 – Méthode Caltech.

Étape de la décomposition de processus : chaque processus est itérativement affiné en une collection de processus interactifs simples.

Étape de l'expansion des communications (HSE ou « HandShake Expansion ») : chaque canal de communication est remplacé par les fils explicites conformément au codage choisi. Egalement dans cette étape, chaque action de communication est substituée par des séquences de transitions de signaux respectant le protocole utilisé. Une lecture de canal « E ?x », par exemple, est remplacée par une séquence de transition de signaux du type :

$$[E^{req}]; x \ := donnes; E^{ack} \uparrow; [\neg E^{req}]; E^{acq} \downarrow$$

Cela signifie : « attente de la requête » puis « lecture des données » puis « acquittement du canal » puis « attente de la fin de la requête » et finalement « libération du canal ». A cette étape, il est possible d'ajouter des variables et/ou de re-ordonnancer des transitions de signaux afin d'obtenir une spécification satisfaisant la condition de CSC (cf. §2.6).

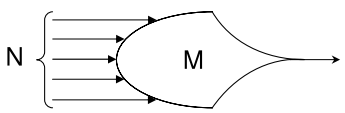
Étape de l'expansion des règles de production : une règle de production (ou PR de l'anglais « Production Rule ») se compose d'une condition (garde) et d'une action : l'action est exécutée seulement quand la condition qui la garde devient vraie. Dans cette étape, chaque expansion de communication est remplacée par un ensemble de règles de production. A titre d'exemple, $E^{req} \wedge S^{ack} \mapsto S^{req} \uparrow$ et $\neg E^{req} \wedge \neg S^{ack} \mapsto S^{req} \downarrow$. Les règles de production – étant elles-mêmes des processus concurrents simples – spécifient le comportement des signaux internes et signaux de sortie d'un processus. Les gardes doivent assurer que les transitions de signaux se produisent dans l'ordre du programme : la sémantique du programme CHP initial est conservée. Pour cela, il est parfois nécessaire de renforcer les gardes ou d'introduire des variables locales. En outre les gardes peuvent être modifiées et être réalisées de façon symétrique afin d'obtenir une implantation basée sur des cellules standard.

Étape de la réduction des opérateurs : c'est l'étape dans laquelle les PR sont regroupées et associées pour former des éléments matériels comme les portes de Muller. Les deux PR ci-dessus, par exemple, peuvent être rassemblées pour former une porte de Muller à deux entrées.

Appliquée à plusieurs conceptions dont le microprocesseur asynchrone très connu MIPS R3000 [64], la méthode de synthèse du professeur Alain Martin génère des circuits QDI avec le protocole de communication quatre phases. Elle génère des circuits plus optimisés que ceux générés par les autres méthodes dirigées par la syntaxe car l'optimisation peut être appliquée au niveau bas, lors de la génération des portes. Cependant, elle n'est pas totalement automatisée à notre connaissance.

2.5 Méthode NCL

NCL acronyme de « Null Convention Logic » est une logique propriétaire proposée et brevetée par Theseus Logic [37, 59]. Elle est basée sur le codage 3 états (cf. §1.3.2.3.2) et l'utilisation de portes à seuil (de l'anglais « threshold gates »), représentées figure 2.4, qui sont en fait des portes de Muller généralisées [22]. La « valeur ajoutée » se situe au niveau de l'optimisation logique : bien que la synthèse ne soit pas sérieusement vérifiée, il semble que les possibilités de projection technologique soient supérieures avec ces portes généralisées.



Entrées	Sortie
M (ou plus) entrées valides	1
Toutes les entrées sont nulles	0

FIGURE 2.4 – Fonctionnement d'une porte à seuil (M $\leq$ N).

La philosophie des circuits générés par la méthode NCL repose sur le motif de base représenté figure 2.5. Il s'agit de séparer fortement la partie combinatoire (proche du circuit synchrone équivalent) et la partie acquittement/gestion de la validité des sorties (spécifique à l'asynchrone). Ainsi, l'apposition d'une « barrière de registres » en sortie de la partie combinatoire permet d'éviter de propager des données non valides. La génération de l'acquittement se fait dans des blocs logiques distincts (blocs de détection de complétion ou CD) dont la synthèse n'est pas automatisée.

Avec l'intention d'adapter la description de circuits asynchrones aux langages et outils existants, la description du circuit n'est pas basée sur un nouveau langage, pour lequel il serait nécessaire de former des concepteurs, mais en VHDL (pour lequel il sera possible de tirer profit des compétences et outils existants déjà sur le marché). Il faut néanmoins ajouter quelques directives spécifiques à l'approche Theseus pour modéliser des circuits asynchrones (complétion des données, codage).

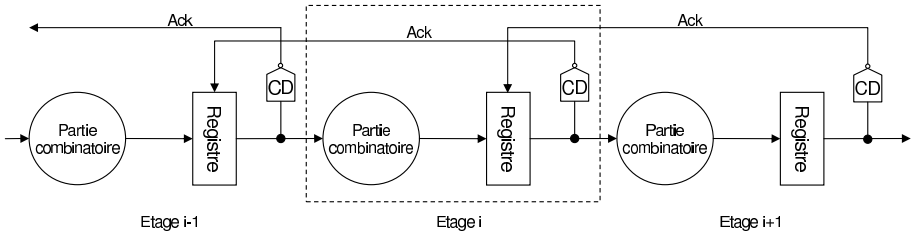


FIGURE 2.5 – Circuits obtenus par la méthode NCL.

La partie combinatoire est décrite de manière classique (ou presque) : la seule distinction repose dans les nouveaux types correspondant au codage de données (3 états) et les nouvelles définitions de portes (correspondant aux portes « AND », « OR », etc.). C'est par la suite que le logiciel optimisera de manière automatique la fonction désirée tout en respectant les critères de circuits asynchrones (absence d'aléa, contraintes QDI, etc.). Toutefois, le code VHDL tel qu'il est à ce stade ne peut être simulé sans introduire la notion de « rendez-vous ». Pour pallier ce problème, un nouveau type de portes est utilisé : les portes à seuil avec « hysteresis ». Ces portes possèdent des fonctions de set et de reset qui ne sont pas complémentaires. Une fois que la porte a commuté, la sortie de la porte ne change pas jusqu'à l'arrivée du signal de reset ; une fois que la porte a été remise à 0, la sortie ne change pas jusqu'à l'arrivée des conditions de commutation. Les registres sont explicitement spécifiés dans le code afin de réaliser des étages de pipeline.

Enfin, la génération des signaux d'acquittement – élément crucial des circuits asynchrones – doit se faire « à la main », c'est-à-dire en instanciant les fonctions logiques dans le code VHDL sans bénéficier de garanties que les conditions d'acquittement sont correctement gérées. Ce point faible est crucial dans la mesure où la grande majorité des méthodologies de circuits asynchrones gèrent d'une manière ou d'une autre les signaux d'acquittement selon des règles bien définies : souvent, cela représente plus de travail que pour la partie calculatoire.

La synthèse est réalisée en deux étapes [59] grâce à un outil de synthèse commercial :

– La description VHDL est synthétisée et optimisée par l'outil de synthèse. Le résultat de cette étape, est un réseau de portes de base qui décrit le flot de données du circuit.

– Tous les signaux et les portes sont codés en double-rail en utilisant toujours un outil de synthèse commercial. Cette étape effectue une optimisation et une projection technologique sur une bibliothèque de cellules de portes à seuil.

Une fonction très intéressante, pour laquelle Theseus pourrait réellement apporter quelque chose de nouveau sur le marché, concerne la vérification des fourches isochrones. Il s'agit de détecter les nœuds susceptibles de constituer des fourches isochrones. La tâche est bien connue comme étant complexe et

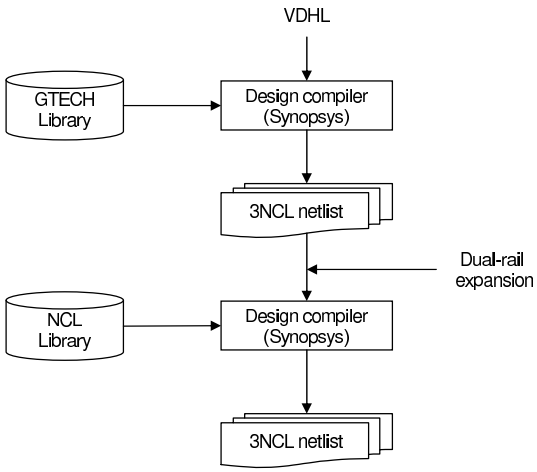


FIGURE 2.6 – Flot de synthèse de la méthode NCL.

nécessitant beaucoup de ressources de calcul, si bien que peu de techniques permettent à ce jour de donner rapidement la liste de toutes les fourches isochrones d'un circuit (cela est d'autant plus vrai que le circuit est complexe). Cependant, la méthode fournie par Theseus est incomplète. Tout d'abord, cette analyse n'est possible que sur les parties calculatoire et non les acquittements. Cette restriction est plutôt inattendue car la méthode est sensiblement la même dans les deux cas. En effet, les contraintes QDI sont cruciales autant dans la partie calculatoire que dans la partie acquittement. Il faut sans doute supposer que, comme pour la synthèse, le modèle de Theseus garantit des arbres d'acquittement simplistes et QDI par construction.

Bien que cette méthode de conception soit la seule avec Haste à être utilisée et développée par une société privée, elle comporte un certain nombre de limitations.

Le style de conception proposé est très typé. Il apparaît que non seulement cette description générique n'est pas universelle, mais de plus est limitée à plus d'un point de vue. En effet, si la distinction opérée entre logique directe et logique d'acquittement semble permettre une relative simplification de la conception, il s'agit tout de même d'une très forte restriction : il n'est dès lors plus possible de générer des acquittements partiels en vue de réaliser des optimisations (n'acquitter que certaines branches du circuit), ni de procéder à toutes les combinaisons complexes où l'acquittement dépend du chemin suivi par les données dans la partie calculatoire. Les implantations du type « lecture conditionnelle » (comme la fonction de multiplexage) ne sont pas synthétisables par cette méthode. Contrairement à Haste, Balsa ou Caltech, où les méthodes d'acquittement sont bien définies, ce principe de communication simplifiée dans laquelle le concepteur implante manuellement les acquittements, se révèle surtout un manque de flexibilité et de robustesse vis-à-vis de la spécification « haut niveau ». En effet, il n'est nullement possible de vérifier si la génération des signaux d'acquittement est correcte, ni si elle réalise le schéma de

communication souhaité.

2.6 Méthode basée sur les STG (Petrify)

Cette méthode, proposée par [27], traite de la conception des circuits asynchrones de type « indépendant de la vitesse » (SI) spécifiés par un graphe de transitions de signaux (STG). Ce type de synthèse basée sur un graphe, spécifie le comportement des circuits asynchrones avec un niveau d'abstraction faible (fréquemment au niveau signal). Pour cette raison, l'écriture de spécifications par cette méthode est souvent sujette aux erreurs surtout si la taille du circuit à concevoir est conséquente.

La figure 2.7 représente le flot de conception de la méthode basée sur les STG.

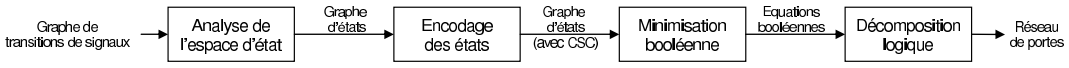


FIGURE 2.7 – Flot de conception de la méthode STG.

Basé sur les réseau de Petri [74, 94], le modèle STG [23], servant à spécifier le circuit asynchrone, est une re-formalisation du diagramme temporel (un diagramme temporel sert à spécifier les relations de causalité entre des événements de signaux). Il permet de modéliser la concurrence et une version limitée de choix entre les données. Un STG, dont un exemple est donné figure 2.8, est en fait un réseau de Petri limité par les caractéristiques suivantes :

- Liberté de choix : la sélection entre des alternatives doit être seulement contrôlée par les entrées mutuellement exclusives.
- 1-borné : il n'existe jamais plus de deux jetons dans une place.
- Vivacité : il ne peut pas y avoir de blocage.[95].

Dans cette méthode, un circuit SI à concevoir est décrit par un STG qui possède les caractéristiques suivantes :

- Cohérence : les transitions d'un signal doivent strictement alterner entre la montée et la descente dans n'importe quelle exécution du STG.
- Persistance : si une transition de signal est autorisée, elle doit avoir lieu, c'est-à-dire qu'elle ne peut pas être désactivée par une autre transition. La persistance des signaux internes et des signaux de sortie doit être garantie par le STG, tandis que celle des signaux d'entrée est garantie par l'environnement.

La spécification STG décrit des relations de causalité entre les événements (les transitions de signaux). Pour calculer la fonction de chaque sortie – la tâche de synthèse de circuit – l'espace d'états at-

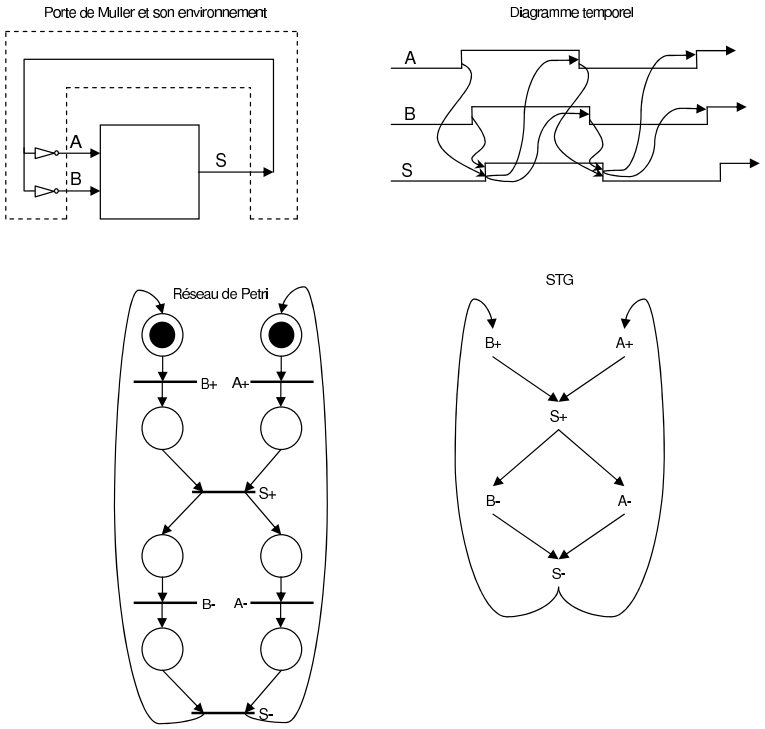


FIGURE 2.8 – Porte de Muller et son environnement.

teignables doit être produit. Le nombre d'états de cet espace augmente exponentiellement avec le nombre de signaux du STG. C'est le point faible principal de cette méthode, puisque le nombre de signaux est limité en pratique à une vingtaine. En utilisant la théorie des régions sur cet espace d'états, les équations logiques booléennes des signaux internes et des signaux de sortie sont calculées.

Un problème d'ambiguïté peut se poser lors du calcul des fonctions de signaux : il peut exister des états différents ayant le même codage. En fait, ce phénomène apparaît quand les signaux seuls ne suffisent pas à identifier tous les états. Dans ce cas, le système viole la propriété CSC (« Complete State Coding ») [26]. Il est difficile pour l'outil de synthèse de garantir la propriété CSC. La solution consiste à ajouter des variables d'état supplémentaires de façon à ce que les différentes combinaisons de codage correspondent de façon unique aux différents états.

L'implantation matérielle du circuit est ensuite déduite en respectant des contraintes technologiques. Dans le style semi-custom, les circuits doivent être construits avec les portes d'une bibliothèque spécifique de cellules, tandis que dans le style custom, les circuits sont composés de portes complexes et la complexité de ces portes est déterminée par des contraintes telles que l'entrance maximale. Chaque équation booléenne générée après optimisation doit être implantée par un ensemble de portes. Il faut alors résoudre le problème de la décomposition logique tout en préservant le fonctionnement correct du circuit. Ceci est une tache très difficile à réaliser dans les outils de synthèse des circuits asynchrones. Cela demande beaucoup de ressources en termes de temps de calcul.

En résumé, la conception de circuit asynchrones SI avec cette méthode basée sur les STG implique les étapes suivantes.

- Décrire le comportement souhaité du circuit et de son environnement par un STG.
- Vérifier si cette représentation satisfait les caractéristiques d'un STG décrivant les circuits SI.
- Vérifier si ce STG est synthétisable (CSC).
- Choisir un modèle d'implémentation et calculer les équations booléennes. A partir de ces équations, l'implémentation du circuit est réalisée en utilisant des portes atomiques (telles que la porte complexe AOI) ou des portes de base plus simples.
- Modifier cette implantation pour que le circuit puisse être forcé dans l'état initial souhaité en utilisant un signal de reset ou des signaux d'initialisation.

Cette méthode est automatisée par un outil de synthèse du nom de Petrify développé par Jordi Cortadella. Cet outil prend en entrée un STG décrivant le circuit désiré, saisi sous forme texte ou graphique. La synthèse dans cet outil est cependant limitée à des circuits de faible complexité du fait de l'explosion combinatoire lors de l'exploration des états possibles du STG.

2.7 Méthode basée sur ASM (Minimalist)

Cette méthode, proposée par Nowick dans [71], traite de la conception de contrôleurs asynchrones fonctionnant en mode rafale. Dans le mode rafale, l'environnement est autorisé à changer plusieurs entrées du circuit (l'ordre du changement n'a pas d'importance) lorsque le circuit est stable. Il n'est pas autorisé à changer de nouveau les entrées avant que le circuit ne produise les sorties et revienne à l'état stable. Ce mode est une extension du mode fondamental dans lequel l'environnement n'est autorisé à changer qu'une seule entrée lorsque le circuit est redevenu stable et a produit une sortie. Puisque l'environnement ne connaît pas l'état stable du circuit, il doit respecter le délai le plus long nécessaire pour stabiliser le circuit. Ceci implique de borner le délai des portes et des fils du circuit. Cette limitation sur l'environnement est aussi formalisée comme une contrainte temporelle absolue.

La spécification utilisée dans cette méthode est une machine à états de type Mealy. Du point de vue du calcul, cette spécification se base sur les états : pour chaque état, la machine peut recevoir des entrées, génère des sorties et avance jusqu'à l'état suivant (une telle spécification peut également être décrite dans un tableau de flots [92]). A titre d'exemple, la figure 2.9 illustre la spécification d'un contrôleur fonctionnant en mode rafale. Cette spécification est plus concise que le STG, notamment quand la concurrence du système est importante. Dans cet exemple, nous voyons que, dans l'état 1, si des fronts montants se propagent sur les entrées A et B, cela va provoquer des fronts montants sur les sorties X et Y et l'état

suivant sera l'état 2. Puis, dans l'état 2, si un front montant arrive sur l'entrée de C, la sortie X va redescendre et l'état suivant sera l'état 3. Et ainsi de suite, jusqu'à décrire entièrement le fonctionnement de notre contrôleur.

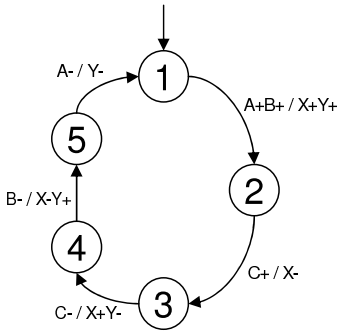


FIGURE 2.9 – Exemple de spécification en mode rafale.

Des contraintes sur la spécification sont définies pour que le contrôleur fonctionne correctement. D'abord, à chaque état donné, aucun changement des entrées ne peut être couvert par d'autres changements des entrées puisque dans ce cas le comportement du contrôleur pourrait être ambigu. La deuxième restriction est que chaque état a un point d'entrée unique, ce qui simplifie la minimisation et garantit une implémentation sans aléa. Une dernière restriction est qu'il n'y a pas de changement d'états sans un changement sur des entrées. Cela signifie que le système reste dans un état stable si aucune entrée ne change. Nowick a également proposé une méthode de synthèse à partir de cette spécification en mode rafale, basée sur une implémentation de type machine à états synchronisée localement (« locally-locked state machine »). Intrinsèquement, cette machine est une machine de Huffman ajoutant une unité d'auto-synchronisation, qui fonctionne comme une horloge locale sur des verrous. Cette machine est dite auto-synchronisée (« self-synchronized ») (figure 2.10). Les caractéristiques suivantes font que la machine à états proposée par Nowick est différente des machines de Huffman :

– L'horloge est générée de manière sélective : certaines transitions ne nécessitent pas d'horloge.

– L'unité d'horloge n'utilise pas de modèle de délais « inertiel » pour éliminer les aléas.

– La logique combinatoire sans aléa est requise pour obtenir certaines transitions.

La méthode de synthèse se compose des étapes suivantes. La première étape est de générer, à partir de la spécification, un tableau de flots pour les sorties et les prochains états. Les états dans ce tableau sont ensuite minimisés et affectés de codes d'état uniques. La dernière étape consiste à générer les fonctions booléennes pour l'horloge, chaque sortie et chaque variable d'état. Ces fonctions booléennes subissent une minimisation logique, afin d'obtenir une implantation sans aléa.

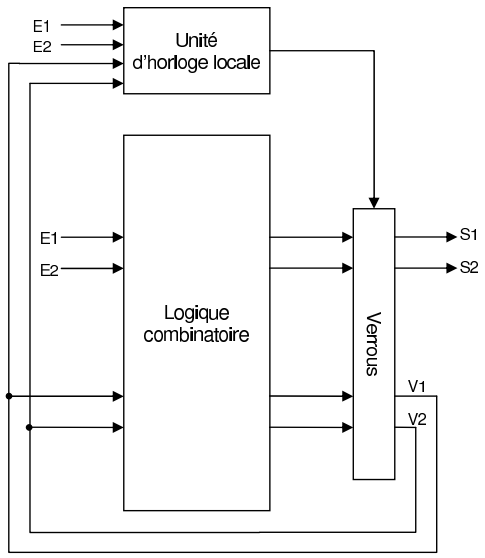


FIGURE 2.10 – Schéma de la machine auto-synchronisée.

Un outil de synthèse appelé Minimalist [43], a été développé pour prototyper cette méthode. Grâce à cet outil, les contrôleurs asynchrones peuvent être générés en utilisant des portes de Muller généralisées.

Bien que cette méthode ait été appliquée dans la conception de contrôleurs STRiP, il reste encore des problèmes à résoudre :

- Le fonctionnement des circuits générés est limité par le mode rafale.
- Il est difficile de décrire un système complexe à partir de la spécification en mode rafale.
- Le fonctionnement sophistiqué et complexe des circuits générés nécessite un test. Tester une telle implantation consiste à tester la logique combinatoire mais également les délais dans les différents chemins afin de s'assurer que les contraintes temporelles sont respectées. Ce type de test est relativement complexe à mettre en place.

2.8 Méthode TAST

Afin de pallier le manque d'outils de conception pour les circuits asynchrones de type QDI, le laboratoire TIMA a développé un environnement de conception : TAST ([33, 31, 32]). Cet environnement comporte l'ensemble des outils d'aide à la conception des circuits asynchrones QDI. L'objectif final de ces outils, associés à un langage de conception de haut niveau tel que le langage de description des processus communicants CHP, est de générer le réseau de portes d'un circuit asynchrone à partir de sa description comportementale de haut niveau. Ce réseau de portes va permettre de réaliser le dessin des masques du circuit. Les méthodes de projection technologique et d'analyse de netlist, présentées dans ce manuscrit, représentent la partie « back-end » de cet environnement de conception.

Un concept important sur lequel s'appuie largement toute la conception numérique, est celui de la hiérarchie. La nécessité d'évoluer depuis une description haut niveau vers la réalisation matérielle reflète précisément le degré de complexité élevé qui ne permet pas de considérer simultanément toutes les caractéristiques d'un circuit dans son ensemble. Le modèle de conception « top-down », largement utilisé aussi bien en synchrone qu'en asynchrone, correspond au développement d'un circuit en commençant par les aspects comportementaux pour évoluer au fur et à mesure vers les aspects les plus techniques et enfin aboutir à la réalisation. C'est dans ce cadre que la description basée sur le langage CHP s'affranchit des caractéristiques d'implantation pour ne considérer que les éléments de type communication ou fonction macroscopique.

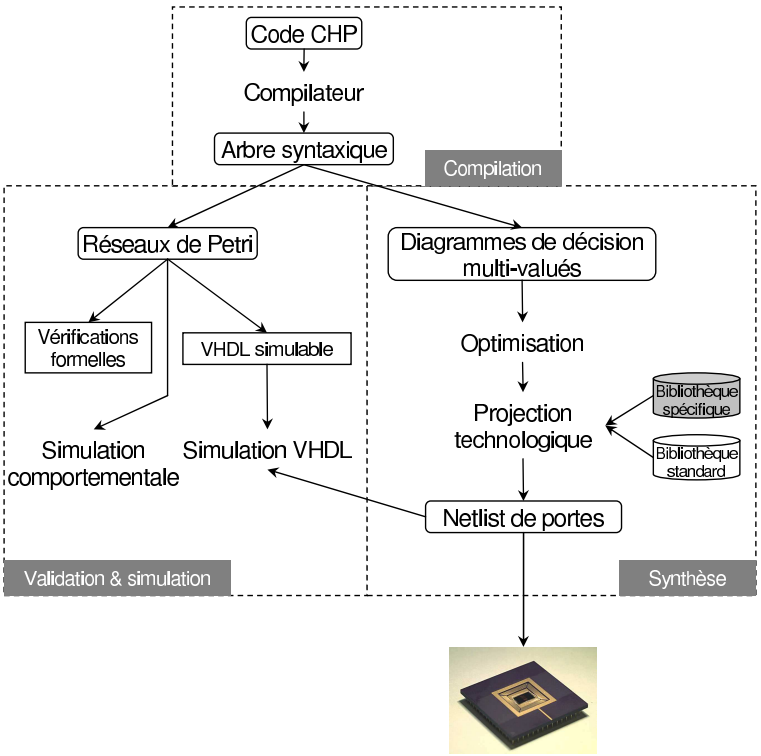


FIGURE 2.11 – Flot de conception TAST.

La figure 2.11 présente le flot de conception des circuits asynchrones implanté dans TAST. Initialement, le circuit est décrit en CHP. Le langage décrit à haut niveau les communications entre les différents processus qui composent le circuit. A partir de cette description, l'outil génère un ensemble de structures composées :

– de réseaux de Petri et de graphes flot de données. Les réseaux de Petri permettent de modéliser les états fonctionnels d'un système et les relations qui les lient avec une complexité algorithmique réduite. L'utilisation de ces graphes va permettre de réaliser des vérifications formelles sur le circuit [14], mais également de simuler fonctionnellement le circuit très tôt dans la phase de conception.

– de diagrammes de décision multi-valués. Ces diagrammes, présentés dans les chapitres suivants, vont permettre de réaliser les étapes de synthèse des circuits asynchrones tout en conservant leurs propriétés d'insensibilité aux délais.

Ainsi TAST apporte des contributions sur plusieurs points :

– un environnement complet se composant des outils d'aide à la conception de circuits asynchrones, de la description haut niveau jusqu'à l'implantation matérielle,
– une description haut niveau des circuits asynchrones avec une complexité importante,
– une méthode de conception permettant d'explorer différents styles de conception suivant le protocole de communication et le type de codage des données choisis,
– une co-simulation des implantations fonctionnelles et matérielles à l'aide d'outils commerciaux.

2.9 Conclusion

Dans ce chapitre, différentes méthodologies de conception de circuits asynchrones sont présentées. Ces méthodologies sont déterminées par la méthode de spécification et de synthèse. Deux méthodes principales existent pour spécifier des circuits asynchrones : la première basée sur un langage de description de haut niveau ; la seconde basée sur un graphe. Il apparaît dans la présentation qui précède, qu'aucune méthodologie n'adresse aujourd'hui tous les problèmes de la conception asynchrone.

Comme nous l'avons vu précédemment, puisque les langages de description de matériel sont utiles pour concevoir hiérarchiquement les circuits complexes, et que les algorithmes d'analyse temporelle et de synthèse sont plus facilement appliqués aux représentations sous forme de graphes, c'est une combinaison de ces approches qui a été utilisée pour le développement de notre outil de synthèse. Dans TAST, développé au sein du laboratoire TIMA, un circuit asynchrone est modélisé par un programme écrit dans un langage de haut niveau appelé CHP étendu qui combine le langage CHP et le VHDL. Puis ce programme est transformé en graphes (combinaison de réseau de Petri et de graphes flots de données), qui sont utilisés pour la validation et la vérification formelle ; et en diagrammes de décision multi-valués qui sont utilisés pour la synthèse. Le but de notre approche est donc de tirer parti des avantages d'un langage de haut niveau pour la spécification, des avantages des réseaux de Petri pour la validation et des avantages des diagrammes de décision multi-valués pour la synthèse.

Chapitre 3

Projection technologique : un état de l'art

3.1 Introduction

La phase de projection technologique consiste à transformer un réseau logique indépendant en un réseau logique dépendant d'une technologie, c'est-à-dire transformer un réseau de portes génériques en une interconnexion de composants qui sont des instances d'éléments d'une bibliothèque. Cette étape est très importante notamment pour la conception de circuits « semicustom » basés sur des cellules standard, puisqu'elle fournit une représentation structurelle complète du circuit logique qui servira d'interface pour les outils de conception physique.

La projection technologique nous permet de cibler différentes implantations et différentes technologies pour un même circuit logique. Cette étape est donc cruciale pour mettre à jour et perfectionner un circuit.

3.2 Le principe de la projection technologique

La projection technologique est aussi appelée en anglais « library binding ». Ce terme, peut être plus adapté que projection technologique, indique clairement que les paramètres technologiques d'un circuit sont entièrement caractérisés par les caractéristiques d'une bibliothèque en terme de surface, de vitesse et de consommation ; mais également que la tâche essentielle de la projection technologique est de trouver une relation entre un circuit et une bibliothèque, et de trouver une interconnexion possible des cellules de cette bibliothèque.

Une bibliothèque contient un ensemble de primitives logiques comportant des éléments combinatoires et séquentiels. Chaque élément est caractérisé par sa fonction logique, et des paramètres spécifiques tels que son aire, son délai et sa charge capacitive.

En règle générale, l'objectif de la projection technologique est de trouver une solution qui permettra de diminuer la surface (en utilisant dans certains cas des contraintes de délais) ou de diminuer le délai

(en utilisant dans certains cas des contraintes de surface). Malheureusement, ce problème est difficile en terme de calcul. En effet, simplement vérifier l'équivalence entre un réseau dépendant et un réseau indépendant (tautologie) est un problème très complexe.

Une approche traditionnelle de ce problème est de se restreindre à un remplacement de sous réseaux (du réseau original) par des instances d'éléments d'une bibliothèque [53, 60, 30]. On appelle ce problème la couverture de réseau par des éléments d'une bibliothèque. Il faut donc identifier une portion du réseau logique qui pourra être remplacée par une cellule de la bibliothèque, et trouver un ensemble de cellules pour couvrir tout le réseau logique en optimisant certains critères comme la surface et/ou le délai. Une cellule pourra remplacer un sous-réseau s'ils sont fonctionnellement équivalents. Dans ce cas, on dit que la cellule correspond au sous-réseau. Un exemple simple de couverture de réseau de portes est donné (figure 3.1).

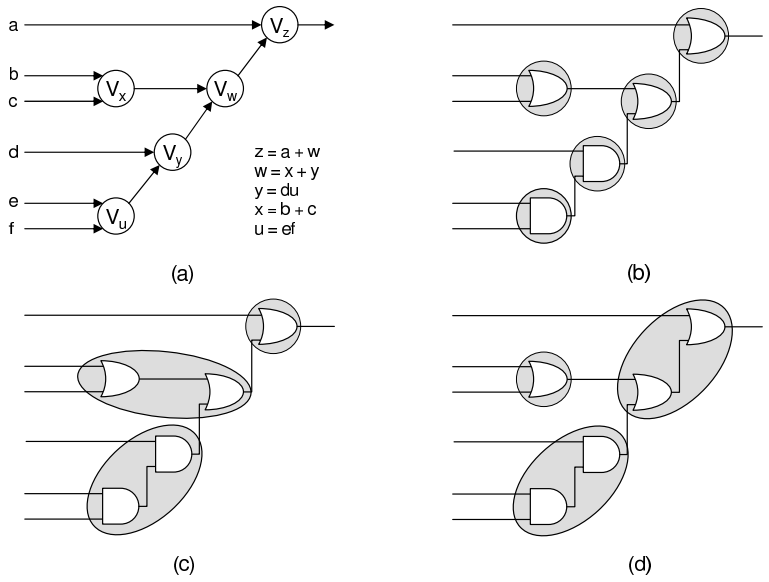


FIGURE 3.1 – (a) Réseau indépendant simple. (b) Projection triviale : une porte par sommet. (c) Couverture utilisant une bibliothèque de portes AND et OR à deux ou trois entrées. (d) Autre couverture possible.

Un réseau logique indépendant, dans lequel chaque nœud correspond à une cellule de bibliothèque, peut être transformé en réseau dépendant de façon directe en remplaçant simplement chaque sommet par la cellule correspondante. On dit cette projection triviale. Cependant, même si le réseau indépendant est initialement optimal en termes de surface et de délai, cette projection triviale ne le sera pas nécessairement. En effet, l'optimisation d'un réseau indépendant se fait en utilisant des modèles simplifiés ne prenant pas en considération les paramètres d'une technologie. Ainsi, la phase de projection technologique implique souvent une restructuration du réseau logique initial.

Décrivons maintenant le problème de couverture d'un réseau indépendant $G_n(V, E)$ plus précisément. $G_n(V, E)$ est un graphe dirigé acyclique ayant V comme ensemble de sommets et E comme ensemble d'arêtes. Un sous-réseau enraciné est un sous-graphe de $G_n(V, E)$ ayant un seul sommet qui ne possède pas d'arête de sortie. Dans le problème de couverture d'un réseau, nous associons à chaque sommet v de $G_n(V, E)$ un sous-ensemble M_v de cellules qui correspondent au sous-réseau de racine v. On dit qu'une cellule de M_v couvre v et les autres sommets du sous-réseau correspondant. Soit M l'ensemble des cellules associées aux sommets du réseau, c'est-à-dire $M = \bigcup_{v \in V^G} M_v$.

La couverture d'un réseau peut être modélisée par la sélection de suffisamment de cellules de M pour couvrir tous les sommets du réseau indépendant. Pour chaque cellule choisie, nous devons nous assurer que ses entrées sont associées aux sorties des autres cellules sélectionnées. Ainsi le choix d'une cellule implique le choix d'autres cellules. Une solution optimale sera celle qui permet de minimiser un coût associé aux cellules choisies : la somme des surfaces des cellules, la somme de leur consommation,...

Une condition nécessaire pour que le problème de couverture d'un réseau ait une solution, est que chaque sommet du réseau initial ait au moins une cellule qui lui corresponde. Cette condition est remplie en général, en décomposant au préalable le réseau initial (avant de chercher une couverture).

3.3 Les algorithmes de projection technologique

Les premiers travaux sur la projection technologique ont été commencés aux AT&T Bell Laboratories par Keutzer [53], dans lesquels il reconnu une similarité entre les problèmes de projection technologique et la phase de génération de code d'un compilateur.

Comme nous l'avons vu précédemment, le problème de la projection technologique peut se ramener à un problème de couverture de réseau. Il existe deux approches principales pour résoudre le problème de couverture :

- Dans l'approche booléenne, les cellules de la bibliothèque et une partie sélectionnée d'un réseau sont décrites sous forme d'équations booléennes. Cette approche a été utilisée dans le programme CERES [60] et résumée dans [6].
- Dans l'approche structurelle, on utilise des graphes pour représenter la décomposition algébrique des fonctions booléennes. Cette approche est celle que l'on retrouve dans DAGON [53].

Du fait de son implantation plus simple, l'approche structurelle est la plus utilisée. Cependant, que ce soit par une approche booléenne ou structurelle, ce problème de couverture est NP-complet. En effet, l'approche booléenne se ramène au problème complémentaire de tautologie, ou problème de satisfiabilité (autrement appelé SAT) problème très connu pour être du type NP (c'est-à-dire qu'il n'est pas soluble en

un temps polynomial). L'approche structurelle quant à elle est équivalente au problème d'isomorphisme de graphes, également du type NP.

Il est donc nécessaire d'appliquer des heuristiques afin de trouver une solution. Ces heuristiques s'appliquent en général dans des phases préliminaires à la couverture : la décomposition et la partition [53, 30].

3.3.1 La décomposition

La décomposition est nécessaire pour garantir que chaque sommet de notre réseau initial sera couvert par au moins une cellule, et ainsi garantir une solution au problème de couverture. L'objectif de cette décomposition dans ce contexte, est donc de transformer les fonctions locales de chaque sommet en fonctions de base : NOR et NAND à deux entrées par exemple (figure 3.2, inspirée de [83]). Evidemment, le choix des fonctions de base va dépendre de la bibliothèque cible. Dans tous les cas, si les fonctions de base sont présentes dans la bibliothèque cible et que la décomposition est possible, une projection triviale existe.

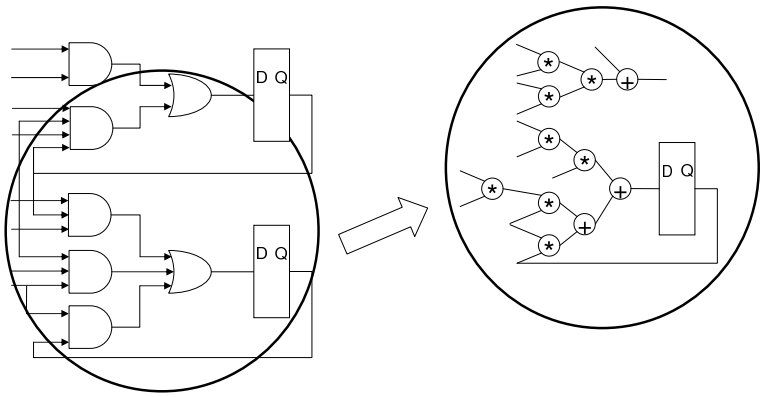


FIGURE 3.2 – Décomposition d'un réseau en portes de base (AND et OR à deux entrées).

De plus, la décomposition d'un réseau n'est pas forcément unique. Il faut donc faire attention aux fonctions de base utilisées car elles jouent un rôle important dans la qualité de la solution. Par exemple, si notre objectif est d'obtenir une projection qui minimisera le délai, la décomposition devra faire traverser un nombre minimum d'étages aux entrées les plus lentes.

3.3.2 La partition

La partition permet à la phase de couverture de considérer un ensemble de réseaux à plusieurs entrées et une seule sortie, à la place d'un réseau unique à plusieurs entrées et plusieurs sorties. Les sous-réseaux obtenus lors de cette phase de partition s'appellent des graphes sujets. On cherchera donc une couverture pour chaque graphe sujet séparément. Le gain majeur de cette phase de partition est que les réseaux sur

lesquels sont appliqués les algorithmes de couverture sont plus petits. Dans le cas de circuits complets, la phase de partition est également utilisée pour séparer la partie combinatoire d'un circuit, de sa partie séquentielle et de ses plots d'entrées/sorties (figure 3.3, inspirée de [83]).

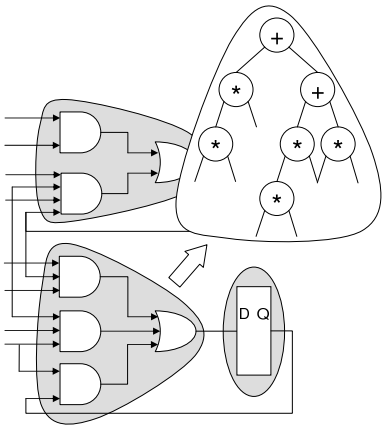


FIGURE 3.3 – Partition d'un réseau en graphes sujets.

Plusieurs méthodes sont possibles pour faire cette partition. Si on considère un circuit combinatoire, les arêtes comportant une fourche peuvent être marquées, et les sommets sur lesquels arrivent ces arêtes forment une frontière. Cette méthode implique que la phase de couverture ne pourra pas modifier la structure du réseau au-delà de ces fourches.

Une autre méthode consiste à regrouper en une partition tous les sommets qui se trouvent sur un chemin se terminant par une sortie primaire et de les enlever du graphe. Bien que cette méthode permette d'avoir des blocs de partition plus importants en taille, elle est dépendante de l'ordre du choix des sorties.

Il est important de comprendre que, malgré le fait que ces deux étapes préliminaires soient des heuristiques permettant de réduire la complexité du problème de couverture, elle contribue à la qualité de la solution trouvée.

3.3.3 La couverture

Comme nous l'avons vu, notre objectif est de couvrir notre réseau par une interconnexion de cellules de bibliothèque. Pour chaque portion de notre réseau, toutes les cellules d'une bibliothèque vont être essayées pour une couverture. Si une cellule correspond et est sélectionnée, cette portion de réseau sera remplacée par cette cellule avec ses propriétés de surface et de délai. Un choix devra être fait quant aux critères de sélection des cellules (figure 3.4, inspirée de [83]).

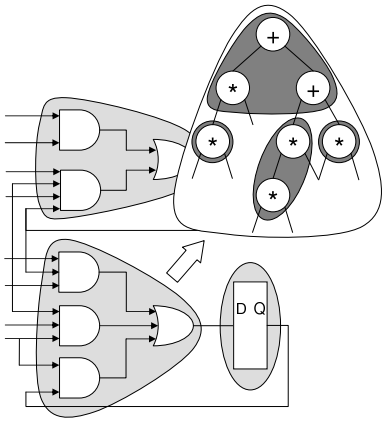


FIGURE 3.4 – Couverture du graphe sujet.

3.4 Le cas des circuits asynchrones

3.4.1 La décomposition dans les circuits asynchrones

Un des principaux challenges dans la conception et l'optimisation de circuits asynchrones, est d'assurer la correction fonctionnelle de l'implantation. Comme nous allons le voir, certaines étapes de l'approche traditionnelle synchrone de la projection technologique ne sont pas directement applicables pour les circuits asynchrones. En effet, la phase de décomposition synchrone appliquée arbitrairement peut provoquer un fonctionnement incorrect d'un circuit asynchrone.

La figure 3.5 présente une décomposition d'une porte de Muller en utilisant une méthode de décomposition propre au synchrone.

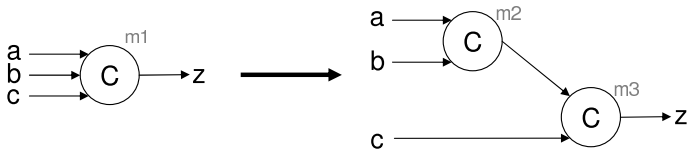


FIGURE 3.5 – Décomposition arbitraire d'une porte de Muller 3 entrées.

Dans cette figure, supposons que notre circuit contienne une porte de Muller à 3 entrées *m1*. Supposons maintenant que nous disposions d'une bibliothèque dans laquelle les portes de base sont des portes de Muller à 2 entrées. Un algorithme de projection technologique classique (comme pour le cas synchrone) pourrait décomposer une porte de Muller à 3 entrées en un réseau de portes de Muller à 2 entrées (par exemple *m2* et *m3* dans la figure 3.5). Finalement, après la phase de couverture, supposons que chaque porte de Muller à 2 entrées est couverte directement par des portes de base de notre bibliothèque.

Ainsi, dans le circuit original, lorsque $a = 1, b = 1, c = 0$, la porte *m1* ne commute pas. Cependant, dans le circuit couvert figure 3.5, *m2* commute sans que *m3* ne commute (figure 3.6). Nous nous

retrouvons dans une situation où la porte *m2* n'est pas acquittée par une sortie primaire, on dit qu'un orphelin est présent en sortie de la porte *m2*. Cela pose un réel problème pour la correction fonctionnelle du circuit. En effet, il est facile de voir qu'un signal non désiré peut apparaître sur la sortie de la porte *m3*, suivant le changement de valeur des entrées.

Par exemple si les entrées du circuit deviennent $a = 0, b = 0, c = 1$, la sortie de la porte *m1* restera inchangée. Par contre, dans le cas du circuit couvert, la porte *m2* ayant commutée précédemment, la valeur de sa sortie vaut toujours 1 puisque la porte n'a jamais été acquittée. La porte *m3* va donc commutée et ainsi provoquer un disfonctionnement du circuit.

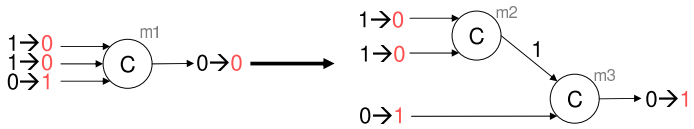


FIGURE 3.6 – Séquence de commutation provoquant un fonctionnement incorrect.

La décomposition classique d'un réseau de portes, comme celle utilisée pour les circuits synchrones, peut modifier la correction fonctionnelle du circuit en introduisant des aléas et n'est donc pas directement applicable.

3.4.2 La couverture dans les circuits asynchrones

Comme nous l'avons vu, le problème de projection technologique peut se ramener à un problème de couverture d'un graphe dirigé acyclique (DAG). [57] présente une méthode de projection technologique directement sur un DAG, dans laquelle le réseau de portes non couvert n'est pas partitionné en arbres. Nous allons voir que cette méthode peut être dangereuse car il est possible de générer des orphelins et donc obtenir un circuit incorrect vis-à-vis de l'insensibilité aux délais. Les algorithmes basés sur les DAG se caractérisent par le fait qu'un même sommet d'un graphe sujet peut être couvert par plusieurs cellules dans le circuit.

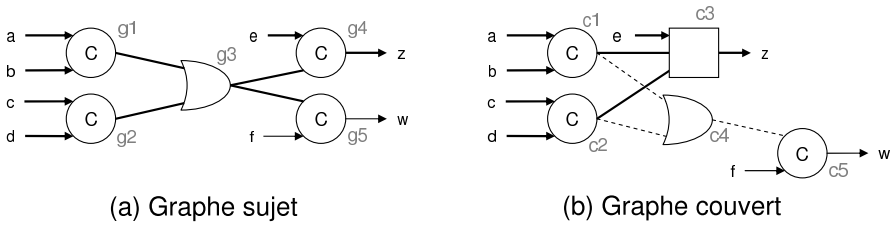


FIGURE 3.7 – Exemple de projection technologique basée sur un DAG.

Pour illustrer ce problème de couverture, référons nous à la figure 3.7. Dans cette figure, le graphe de gauche est le graphe sujet sous forme de DAG, et nous supposons qu'il ne contient pas d'aléa. Le sommet

g3 du graphe va être couvert par deux cellules (*c3* et *c4*) lors de la phase de couverture. La figure 3.7b montre le circuit couvert ; les autres sommets *g1*, *g2* et *g5* sont couverts de façon triviale respectivement par les portes *c1*, *c2* et *c5*. Pour la combinaison d'entrées $a = b = c = d = e = 1$ et $f = 0$, tandis que le graphe sujet ne contient pas d'orphelin, dans le circuit couvert un orphelin apparaît en sortie de *c4*. Cet orphelin peut provoquer un aléa en faisant commuter par erreur le fil *w* si la combinaison d'entrées suivante est $a = b = c = d = e = 1$ et $f = 1$.

Nous voyons dans ce cas également, que l'utilisation des méthodes de projection technologique synchrones peut provoquer l'apparition d'orphelins dans un circuit asynchrone. La présence de ces orphelins peut résulter en une implantation incorrecte de notre circuit.

3.4.3 Les approches existantes pour les circuits asynchrones

Nous avons vu que les méthodes classiques de projection technologique ne sont pas directement applicables aux circuits asynchrones. Il existe plusieurs approches suivant le type de circuits asynchrones étudié.

Depuis le début des années 90, des efforts importants ont été fournis dans le domaine de la projection technologique pour les circuits asynchrones. Pour les circuits de contrôle fonctionnant en mode fondamental, tels que les circuits en mode rafale synthétisés par l'outil Minimalist (cf. §2.7), une première approche de projection technologique sans aléa a été introduite par Siegel et al. dans [82]. Cette méthode consiste à modifier les algorithmes existant pour les circuits synchrones afin de les adapter aux circuits asynchrones en mode rafale. Pour ce type de circuit asynchrone, il est possible de séparer la partie combinatoire de la partie séquentielle comme pour un circuit synchrone. Ainsi il est montré que seule la phase de couverture est susceptible d'introduire des aléas dans le circuit. Pour éviter cela, il est nécessaire de détecter les aléas dans les éléments de la bibliothèque cible. Les résultats de cette analyse sont utilisés dans la phase de couverture pour assurer que la correction fonctionnelle du circuit est conservée.

Par la suite, Kudva et al. [56] développèrent une méthode pour implanter des circuits en mode rafale sans aléa en utilisant des portes CMOS complexes. L'utilisation de ces portes complexes permet de ne pas introduire d'aléas lors de la couverture du circuit.

Dans [4], Beerel et al. s'intéressent au problème d'optimisation des circuits asynchrones en mode rafale lors de la phase de projection technologique, contrairement aux autres méthodes qui se focalisent uniquement sur les problèmes d'aléas. L'optimisation des circuits asynchrones se fait sur le temps moyen d'exécution du circuit, contrairement aux circuits synchrones où le concepteur essaye d'optimiser le temps critique du circuit. Dans la méthode proposée, la spécification du circuit est d'abord analysée

pour évaluer de façon stochastique la fréquence d'occurrence des transitions d'états du STG décrivant le circuit asynchrone. Ensuite, les algorithmes de projection technologique prennent en considération la fréquence relative des transitions d'états pour minimiser leur temps de cycle.

Pour les circuits asynchrones indépendants de la vitesse, de nombreux travaux existent. [3] introduit la décomposition et la projection technologique de circuits indépendants de la vitesse décrits sous forme d'architecture dite standard-C. Dans ce type d'architecture, le circuit initial est composé de portes de Muller et de blocs logiques combinatoires. Chaque bloc logique, modélisé sous forme de sommes de produits, peut être représenté de façon équivalente par un réseau de portes AND et OR (figure 3.8). Chaque bloc évalue la valeur d'une seule sortie du circuit.

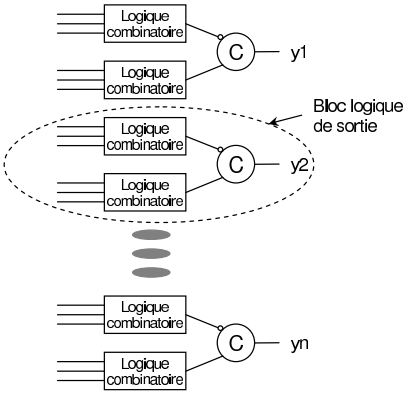


FIGURE 3.8 – Circuit indépendant de la vitesse sous forme standard-C.

Ces travaux ont par la suite été complétés par Siegel et De Micheli dans [84]. Dans les circuits indépendants de la vitesse, c'est la phase de décomposition qui peut introduire des aléas. Les phases de partition et de couverture ne posent pas de problème. Pour réaliser une décomposition sans aléa, il est nécessaire d'analyser et de caractériser le comportement de l'environnement du circuit. En effet, l'ordre d'occurrence des signaux doit être connu pour contrôler l'apparition des aléas dans le circuit. La figure 3.9 permet d'illustrer l'apparition d'un aléa statique 0 dans un circuit indépendant de la vitesse suite à une mauvaise décomposition.

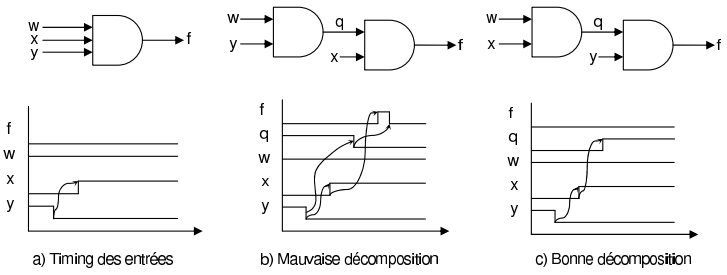


FIGURE 3.9 – Exemple de décomposition.

La figure 3.9a montre une porte réalisant la fonction logique $f = wxy$. Suivant le temps d'arrivée des entrées, une transition descendante sur *y* sera suivie d'une transition montante sur *x* ; *w* reste constante à 1. Supposons que les entrées *w* et *y* soient placées sur une porte AND séparée comme dans la figure 3.9b. Quand *y* descend, la sortie de la porte intermédiaire va descendre peu de temps après. Avant que ce changement ne soit pris en compte par la porte AND finale, *x* peut monter, générant un aléa sur la sortie. Dans la figure 3.9c, *w* et *x* sont connectées à la même porte et *y* est connectée à la porte AND la plus proche de la sortie. Dans ce cas, *y* change en premier, présentant un 0 en entrée de la dernière porte AND. Le changement de valeur de *x* va induire un changement de la porte intermédiaire, mais la sortie finale restera stable, du fait que *y* est déjà à l'état bas. Ainsi aucun aléa n'apparaît sur la sortie avec cette décomposition.

Burns présente dans [22] les conditions générales et les algorithmes pour une décomposition robuste des éléments séquentiels. Cependant les algorithmes proposés ne permettent pas d'obtenir systématiquement l'optimalité.

Dans [28, 55], Cortadella et al. présentent un ensemble de méthodes pour la décomposition et la projection technologique de circuits indépendants de la vitesse synthétisés avec Petrify (§2.6). Les méthodes proposées sont basées sur l'utilisation et l'analyse des STG décrivant le circuit asynchrone. Les portes complexes composant le circuit initial sont décomposées en portes de base à deux entrées. Des signaux internes sont ensuite ajoutés au circuit, afin de contrôler l'apparition d'aléas.

Pour les circuits basés sur l'utilisation de portes à seuil comme la logique NCL (§2.5), peu de travaux existent. Les premières recherches se trouvent dans [37, 59]. Récemment, Jeong présente dans [51] une nouvelle méthode de projection technologique pour ce type de circuits. Dans un premier temps, le circuit est généré en utilisant les outils fournis par Theseus logic. Le circuit obtenu est composé de porte NCL à 3 entrées. Ces portes à 3 entrées peuvent être décomposées en portes NCL à 2 entrées sans modifier la correction fonctionnelle du circuit. Il suffit ensuite de regrouper les portes à 2 entrées suivant les portes disponibles dans la bibliothèque cible, pour obtenir le circuit projeté. Les gains obtenus par cette méthode sont importants aussi bien en termes de surface que de délai.

Les différentes méthodes de projection technologique des circuits asynchrones existantes reposent sur l'utilisation soit d'une bibliothèque composée de cellules de base standard (AND et OR) comme pour les circuits en mode rafale et indépendant de la vitesse ; soit d'une bibliothèque spécifique comme pour la logique NCL. Il existe d'autres types de bibliothèques spécifiques qui ont été utilisées pour la conception des différents types de circuits asynchrones, comme nous allons le voir dans la suite.

3.5 Les bibliothèques de cellules asynchrones

Il existe en fait peu de bibliothèques de cellules dédiées à la conception de circuits asynchrones. En règle générale, les bibliothèques utilisées ont été développées pour des circuits synchrones et sont adaptées aux circuits asynchrones.

Une bibliothèque de cellules asynchrones, basées sur l'utilisation d'un protocole de communication de type PCHB (Precharged Half Buffer), a été développée par Ozdag et Beerel ([72]). Le protocole PCHB permet d'optimiser les communications entre processus dans un circuit asynchrone, en autorisant la remise à 0 de la sortie d'un bloc avant la remise à 0 de ses entrées. L'utilisation de cette bibliothèque permet d'obtenir des circuits asynchrones QDI relativement rapides. Cependant cette méthode de conception avec cette bibliothèque n'est pas automatisée.

[38, 40, 39] illustrent l'utilisation d'une bibliothèque de cellules asynchrones dédiée aux circuits de type micropipeline. Cette bibliothèque nommée single-track, comporte un ensemble de cellules dans lesquelles la gestion du protocole de communication micropipeline est incluse. Cette bibliothèque permet d'obtenir des circuits micropipeline ayant de bonnes performances en vitesse.

Dans [79], Jean-Baptiste Rigaud a développé une bibliothèque de cellules spécifiques aux circuits asynchrones, à partir de cellules standard AND et OR (figure 3.10). Cette bibliothèque comporte un ensemble de portes logiques propres aux circuits asynchrones : des portes de Muller à plusieurs entrées, des portes de Muller dissymétriques, ... Plusieurs circuits asynchrones utilisant cette bibliothèque ont été développés au sein du laboratoire TIMA. Cette bibliothèque a également permis de projeter des circuits asynchrones sur FPGA ([49]).

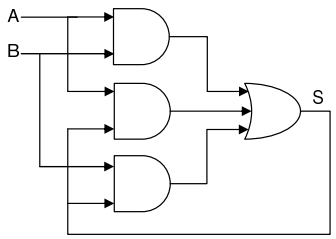


FIGURE 3.10 – Modélisation d'une porte de Muller à partir de portes standard AND et OR.

Plusieurs bibliothèques ont été développées pour augmenter la sécurité des circuits synchrones. Dans [58], les auteurs utilisent des portes logiques spécifiques de type SABL (Sense Amplifier Based Logic) développées par Kris Tiri [90]. Ces portes ont été initialement développées pour augmenter la sécurité des circuits synchrones, en consommant la même quantité d'énergie durant un cycle de calcul (charge + décharge) quelles que soient les données. Ces portes sont utilisées dans [58] pour réaliser des circuits asyn-

chrones micropipeline. Pour fonctionner, le signal d'horloge des cellules est remplacé par les signaux d'acquittements du micropipeline. Dans la même optique, Sylvain Guillet a proposé une bibliothèque de cellules double-rail dont l'objectif est d'améliorer la résistance des circuits à des attaques différentielles en puissance (DPA) ([47, 46]). Utilisables pour la conception de circuits asynchrones, ces cellules ont néanmoins un gros problème de surface. Une autre bibliothèque de cellules double-rail a été développée au LIRMM par Alin Razafindraibe ([76]). L'objectif de cette bibliothèque est également d'augmenter la résistance des circuits aux attaques DPA, en équilibrant la consommation des portes quelles que soient les données traitées. La surface des cellules de cette bibliothèque est inférieure à celle présentée par Sylvain Guillet, pour un résultat en sécurité presque identique. Finalement, Fabien Germain présente dans [45] une méthodologie de conception de cellules intrinsèquement résistantes aux attaques DPA. Par cette méthodologie, toutes les fonctions logiques à deux entrées sont réalisables. La combinaison de ces portes rend théoriquement possible la conception d'un circuit intégré, asynchrone ou non, résistant aux attaques DPA.

3.6 Conclusion

Nous avons présenté dans ce chapitre les différentes étapes de la projection technologique. Afin de rendre ce problème soluble en un temps polynomial, il est nécessaire d'utiliser des heuristiques : dans un premier temps le circuit est décomposé en portes de base (portes AND et OR par exemple) pour assurer une solution au problème. Puis le réseau initial, modélisant le circuit, est partitionné en sous-réseaux afin de diminuer la complexité de la couverture du réseau par une bibliothèque.

Les méthodes classiques de projection technologique utilisées dans le synchrone n'étant pas directement applicables aux circuits asynchrones, nous avons présenté rapidement les différentes solutions existantes à ce jour. Nous avons également présenté les différentes bibliothèques de cellules utilisées pour la conception de circuits asynchrones.

Aucune des méthodes proposées ne permet d'adresser la classe de circuits asynchrones qui nous intéresse dans cette thèse, les circuits quasi-insensibles aux délais. Notre objectif est donc de projeter des circuits asynchrones quasi-insensibles aux délais sur une bibliothèque de cellules, spécifiques ou non. Cette phase de projection technologique devant se faire de manière automatisée.

Pour obtenir des circuits optimisés, que se soit en termes de surface, de vitesse ou de consommation, nous avons également développé une bibliothèque de cellules dédiée aux circuits asynchrones du nom de TAL.

Partie II

CONTRIBUTION À UN FLOT DE SYNTHÈSE DE CIRCUITS QDI BASÉ SUR DES CELLULES PRÉCARACTÉRISÉES

Chapitre 4

Une bibliothèque de cellules asynchrones : TAL

4.1 Introduction

Comme nous venons de le voir, il existe quelques rares bibliothèques de cellules utilisables pour la conception de circuits asynchrones. Cependant aucune de ces bibliothèques n'est composée de portes standard asynchrones pour les circuits QDI utilisables quel que soit le type de protocole de communication choisi. Par conséquent, en règle générale le concepteur est contraint d'implanter les fonctions booléennes dont il a besoin à partir de portes AO222.

Toutefois cette approche produit des circuits sous optimaux en terme de surface, vitesse et consommation. Dans ce contexte, nous avons développé une bibliothèque de cellules dédiée à la conception de circuits asynchrones QDI, en partenariat avec le LIRM de Montpellier. Ces travaux ont pour base les travaux de Philippe Maurine publiés dans [67, 66]. L'objectif de ce chapitre est de présenter cette bibliothèque, au travers des choix que nous avons faits pour la développer.

4.2 La bibliothèque TAL

4.2.1 Présentation

L'objectif premier de la bibliothèque TAL, acronyme de « Tima Asynchronous Library », est de permettre une diminution de la surface des circuits asynchrones QDI. En effet, comme nous le verrons dans le paragraphe 4.2.3.1 et dans le chapitre 6, l'utilisation de portes optimisées, spécialement conçues pour les circuits asynchrones permet de réduire significativement leur surface. Pour cela, nous avons étudié les motifs les plus utilisés dans la conception de circuits QDI et une trentaine de fonctionnalités ont été sélectionnées pour concevoir les cellules de la bibliothèque. Au final, la bibliothèque mise au point comporte un ensemble de 165 cellules.

Un objectif moindre est de concevoir une bibliothèque de cellules sécurisées. Nous avons donc essayé

lors de la conception des cellules de conserver cette notion de sécurité lors du dimensionnement des portes. Nous verrons un peu plus tard dans ce chapitre les implications de ce choix.

Nous voulions également que cette bibliothèque soit compatible Corelib, et que nous puissions l'insérer dans un flot de conception standard. La bibliothèque a donc été développée à partir de la technologie 130nm de ST Microelectronics ; chaque cellule possède un ensemble de vues pour que la bibliothèque soit utilisable dans les outils de conception existants : outils de synthèse, de placement routage ou de simulation.

Chaque cellule possède les vues suivantes :

- Une vue schématique modélisant la cellule au niveau transistors. C'est à partir de cette vue que la cellule est spécifiée (fonctionnement, surface, consommation, sécurité, ...) par le choix de la topologie et le dimensionnement des transistors. Le dimensionnement des cellules sera explicité dans la suite.
- Une vue symbole, afin de pouvoir utiliser une porte comme bloc de base lors de la conception d'un circuit.
- Une vue layout, représentant la cellule au niveau masque de fabrication. Le dessin des cellules de la bibliothèque a été réalisé conjointement avec le LIRMM, afin de ne pas perdre trop de temps sur cette étape fastidieuse.
- A partir de la vue layout, une vue abstract est générée. Cette vue est utilisée lors de la phase de placement routage.
- Des vues VHDL et Verilog, décrivant le fonctionnement de la cellule et permettant une simulation fonctionnelle.
- Pour finir, une caractérisation complète de la bibliothèque a été réalisée en collaboration avec ST Microelectronics. Cette caractérisation a abouti à une évaluation précise en temps et en consommation de chaque cellule, permettant d'optimiser les circuits lors de la phase d'IPO (« In Place Optimization ») du placement routage des circuits.

Intéressons nous maintenant à la question du dimensionnement des cellules, en étudiant les choix de conception que nous avons fait.

4.2.2 Caractéristiques / Plan de conception

Comme nous l'avons vu, cette bibliothèque a pour objectif de cibler les circuits asynchrones QDI. Pour ce type de circuits, le transfert des données entre un bloc émetteur et un bloc récepteur débute par

l'envoi d'un signal de requête encodé dans les données, et s'achève par l'émission d'un signal d'acquittement par le récepteur (cf. 1.3.2). L'intervalle de temps entre l'envoi de la requête et la réception de l'acquittement a une durée inconnue a priori. Il est donc nécessaire de maintenir les données d'entrée pendant cet intervalle de temps pour garantir la propriété d'insensibilité aux délais.

Ainsi, l'utilisation de portes incluant un circuit de maintien du niveau de la sortie est obligatoire pour garantir le bon fonctionnement des protocoles de communication. Généralement ces circuits de maintien sont réalisés par des latches ou des boucles de rétroaction. L'objectif étant d'implanter la bibliothèque en utilisant une technologie CMOS, il résulte que les portes QDI sont des portes composites ou des portes complexes. En effet, elles peuvent être décomposées en une ou plusieurs portes dynamiques avec un élément de maintien de niveau (figure 4.1c,f).

Du fait de cette structure composite des portes QDI, plusieurs politiques de dimensionnement peuvent être envisagées. Il a été défini un ensemble de 5 règles qui permettent de résumer les choix de développement.

1. Équilibrer les possibilités en courant des plans N et des plans P afin d'équilibrer les durées de phases actives et de retour à zéro.
2. Dessiner au moins chaque fonctionnalité avec un nombre important de sortance afin d'accommoder une large gamme de charges. En général, chaque cellule comporte 6 sortances différentes.
3. Dessiner l'étage de sortie des portes QDI de drive Xi de sorte qu'indépendamment de leur structure, c'est-à-dire de leur poids logique, elles offrent les mêmes possibilités en courant que l'inverseur équivalent (Xi).
4. Dimensionner les portes de sorte que l'accommodation de la charge se fasse en deux étages. Cela signifie que seuls les deux derniers étages vont être dimensionnés de manière à piloter la charge de sortie. Ceci revient à cibler des implantations physiques ayant de faibles capacités d'entrée. Cette stratégie va permettre l'utilisation la plus fréquente possible de portes de sortance minimale, sans compromettre trop les performances en surface.
5. Éviter toute décomposition logique pour laquelle le nœud de sortie est contrôlé par un élément de maintien. Ce choix est préférable pour des raisons de performances en vitesse et en consommation ([67]).

Afin de garantir des performances en vitesse élevées et l'obtention d'un comportement fonctionnel correct, il est nécessaire de tailler judicieusement les cellules. Ce travail de dimensionnement a été réalisé au LIRMM par Philippe Maurine et Alin Razafindraibe et publié dans [67, 66, 76]. Sans entrer dans les détails, ce dimensionnement a été effectué à l'aide des modèles au premier ordre des performances des

cellules CMOS. Ces modèles sont basés sur une modélisation au premier ordre du courant drain source circulant dans les transistors submicroniques.

La figure 4.1 représente une porte de Muller à 2 entrées, une porte de Muller à 2 entrées avec un signal de reset et une porte complexe MO22 (deux portes de Muller à 2 entrées suivies par une porte OR), décrites à partir de portes standard puis au niveau transistors.

Afin de dimensionner les portes de la bibliothèque TAL, nous souhaitons prendre avantage de la structure composite des portes QDI pour minimiser la surface des cellules tout en conservant de bonnes performances en vitesse. L'application de la stratégie définie plus haut, et notamment la règle 4, conduit à ne dimensionner que les deux derniers étages des décompositions des cellules QDI. Cependant le choix de la décomposition des cellules peut avoir un impact important sur les performances obtenues. Il est donc nécessaire de sélectionner une décomposition parmi celles possibles. La règle 5 fournit alors un critère fiable pour sélectionner une décomposition en éléments simples d'une porte QDI, celle susceptible d'offrir les meilleures performances. La régularité des chemins électriques, la somme des poids logiques et le poids logique du dernier étage mis en jeu dans les décompositions permettent alors de faire un choix parmi les décompositions restantes ([88]).

Cependant, cette règle 5 peut être appliquée à une grande majorité des cellules de la bibliothèque mais elle est inadaptée pour les cellules de faible complexité qui ne comportent pas assez d'étages lors de la décomposition. Par exemple, une porte de Muller se décompose en une porte de Muller dynamique suivie par un étage de maintien. Ainsi les portes de Muller sont les seules portes de la bibliothèque TAL qui ne respectent pas cette règle 5.

Bien entendu, il aurait été possible d'accroître la profondeur logique des portes de Muller en insérant un étage de buffer sur le nœud de sortie. Toutefois cette solution se montre trop coûteuse en terme de surface et n'a donc pas été retenue.

Comme nous l'avons vu l'application des règles de dimensionnement, et notamment la règle 4, conduit à ne tailler que les deux derniers étages de la décomposition des cellules QDI. Par conséquent, les étages précédents sont taillés à surface minimale.

4.2.3 Bilan

Nous présentons dans ce paragraphe une comparaison des principales caractéristiques de notre bibliothèque en terme de surface, vitesse et consommation, par rapport à l'utilisation de portes standard AO222. Bien entendu nous ne pouvons faire une présentation exhaustive des performances et du coût en

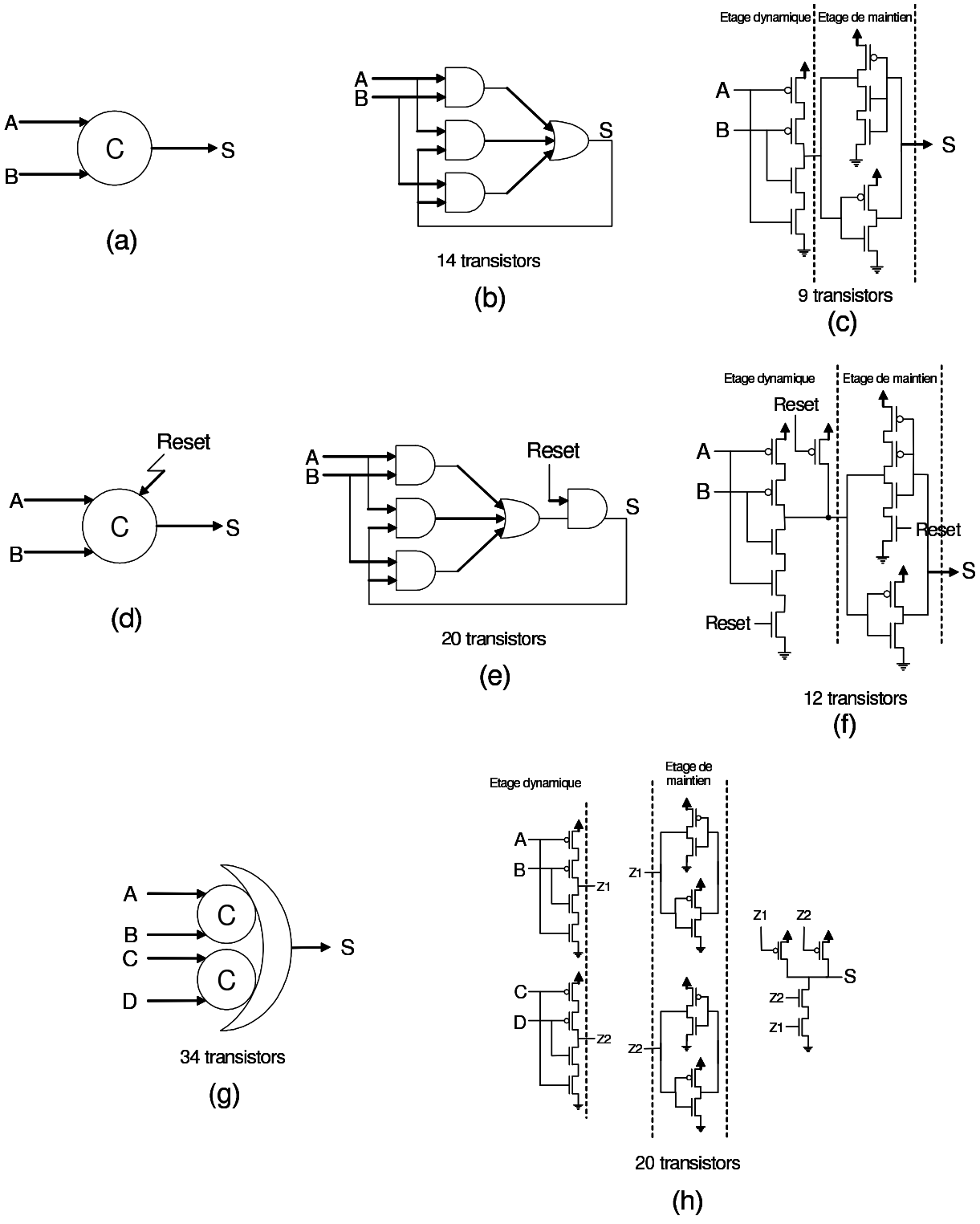


FIGURE 4.1 – (a ,d, g) Symbole des portes de Muller à 2 entrée, à 2 entrées avec reset et de MO22, (b,e) implantation à partir de portes AO222, (c, f ,h) vue schématique des portes.

surface de toutes les cellules ; nous nous limiterons à reporter les caractéristiques de quelques cellules représentatives de l'ensemble de la bibliothèque.

Les cellules que nous avons choisies pour représenter l'ensemble de la bibliothèque sont les suivantes : une Muller à 2 entrées standard, une Muller à 3 entrées, une Muller à 2 entrées avec un signal de reset et une porte complexe MO22. Ce choix se justifie d'une part, parce que ces cellules sont les plus couramment utilisées dans les circuits QDI que nous générons ; et d'autre part parce que les chemins électriques mis en jeu lors de la commutation de ces portes apparaissent très fréquemment dans d'autres portes.

4.2.3.1 Surface

Le premier gain apporté par la conception de cellules spécifiques pour l'asynchrone est de pouvoir réduire le nombre de transistors pour une fonctionnalité donnée. En effet, comme nous pouvons le voir dans la figure 4.1, le nombre de transistors de chaque cellule est bien moins important pour les cellules de la bibliothèque TAL, comparé aux cellules implantées à partir de portes AO222.

D'autre part, le dessin à la main des masques des cellules de la TAL nous a permis de réduire encore leurs surfaces. Dans le tableau ci-dessous, nous donnons le facteur de réduction en surface entre les cellules de la bibliothèque TAL et les implantations correspondantes à partir de portes standard AO222, pour quelques sortances disponibles (X1, X2 et X4). Les portes standard utilisées font partie de la bibliothèque en technologie 130nm fournie par ST Microelectronics.

TABLE 4.1 – Facteurs de réduction de surface obtenus par rapport à des implantations à base de AO222.

Portes	Facteur de réduction en surface (en fonction de la sortance)		
	X1	X2	X4
M2	1,1	1,4	1,5
M3	2	2	2,1
M4	3,3	3,3	2,6
M2RB	1,2	1,2	1,4
MO222	1,9	1,3	1,3

Le facteur de réduction de surface constaté en moyenne est de 2 sur toute la bibliothèque comme l'illustre le tableau 4.1. Le gain majeur étant obtenu pour les portes de grande taille (portes à plus de 3 entrées et portes complexes). En effet, comme nous pouvons le voir dans le tableau suivant, le gain en surface est beaucoup plus important pour les portes complexes et celles comportant un grand nombre d'entrées. Le facteur de réduction moyen étant de 3,24 pour les portes Muller à 4 entrées (sur les différentes sortances disponibles) ; tandis que ce facteur est de 1,48 pour les Muller à 2 entrées.

TABLE 4.2 – Facteur de réduction moyen en fonction du type de porte.

Portes	Facteur de réduction moyen pour chaque fonctionnalité
M2	1,48
M3	2,1
M4	3,24
M2RB	1,27
MO222	1,59

4.2.3.2 Vitesse et consommation

Si la comparaison des coûts en surface est directe, il est nécessaire de définir un protocole de comparaison des performances en vitesse et consommation. Il consiste à appliquer en entrée des portes un signal carré, et de comparer les temps de transition de toutes les entrées vers les sorties, ainsi que la consommation. La donnée brute des temps de propagation, des temps de transition et de la consommation des cellules n'ayant que peu de signification, nous ne reportons que le rapport des cellules TAL aux cellules implantées à partir de portes A0222.

L'ensemble des simulations a été réalisé sous Spectre dans l'environnement Cadence. Les mesures en délai, temps de transition et consommation ont été faites en utilisant une charge de sortie de 4fF, qui correspond à la valeur minimale de charge en technologie 130nm. Les mesures de délai sont faites entre les commutations à 50% des entrées et des sorties ; dans ces mesures, nous avons fait varier toutes les entrées d'une porte en même temps. Finalement, la consommation statique a été obtenue en appliquant une tension de 0V aux entrées.

TABLE 4.3 – Gain en délai des cellules TAL.

Portes	Gain en délai des cellules TAL					
	Montée			Descente		
	X1	X2	X4	X1	X2	X4
M2	0,99	0,85	0,71	0,88	0,94	0,79
M3	1,10	1,12	1	0,8	0,8	0,87
M4	0,79	0,79	0,82	0,81	0,8	0,79
M2RB	1,14	0,79	0,8	0,87	1,10	1,11

Bien que les cellules aient été dessinées pour obtenir un gain important en surface, nous pouvons constater dans le tableau 4.3, un gain en vitesse de propagation. Sur l'ensemble de la bibliothèque, le gain en vitesse est d'environ 10% sur les temps de montée et de descente. Pour les portes complexes du type Muller-Or, le nombre d'étages dans les portes de la TAL est plus important à cause de la mémorisation,

le délai est donc plus important pour ce type de porte, et le gain en délai plus faible. Par exemple pour la porte MO22, la profondeur logique est de 3 étages dans la bibliothèque TAL contre 2 pour la même porte implantée à partir de portes AO222. Le phénomène inverse se produit pour les portes de type M2 et M2 avec reset, pour lesquelles le gain est plus grand.

Le tableau 4.4 présente le gain des temps de transition en montée et en descente des portes de la TAL par rapport aux portes à base de AO222. Les temps de transition correspondent à la pente des signaux en sortie des portes et dépendent donc principalement de leur dernier étage. Les temps de transition sont beaucoup plus courts (environ 30% en moyenne sur la bibliothèque) pour les portes de la TAL. Il est donc possible d'accommoder un plus grand nombre de charges en sortie des portes de la TAL, la dynamique de sortie étant plus importante.

TABLE 4.4 – Gain en temps de transition des cellules TAL.

Portes	Gain en temps de transition					
	Montée			Descente		
	X1	X2	X4	X1	X2	X4
M2	0,91	0,85	0,76	0,85	0,94	0,79
M3	0,43	0,42	0,6	0,64	0,64	0,7
M2RB	0,49	0,49	0,55	0,69	0,68	0,71
MO22	0,95	0,97	0,89	0,85	1,33	1,19

Dans le tableau 4.5, nous illustrons le gain moyen en énergie des portes de la TAL qui se situe autour de 13%. Rapporté à l'ensemble de la bibliothèque, le gain en énergie est de 11%. Toutefois, nous pouvons remarquer que le gain en consommation des portes de type Muller à 3 entrées est plus important que pour les autres types de portes. En effet, pour les portes de Muller à 2 entrées pour lesquelles ce gain en énergie est faible, le driver de sortie est une structure à rapport. Or ce type de structure va dissiper une importante quantité de puissance lors du changement de valeur de toutes les entrées de 0 à 1 (ou de 1 à 0) en provoquant un conflit au niveau du nœud interne situé entre la partie dynamique et la mémorisation (nœud X dans la figure 4.2).

Cette consommation de puissance est assez importante pour diminuer le gain apporté par la diminution de la complexité de la porte, la diminution de consommation d'énergie étant de 8%. En revanche, pour une Muller à 3 entrées, le gain de consommation induit par la réduction de la complexité de la structure devient suffisant pour obtenir un gain substantiel (45%) bien que l'étage de sortie soit un étage de maintien. Ceci s'explique par l'utilisation d'un seul élément de maintien dans la cellule dessinée au lieu de deux boucles de rétroaction dans la structure à base de AO222.

TABLE 4.5 – Gain en énergie des cellules TAL.

Portes	Gain en énergie					
	Montée			Descente		
	X1	X2	X4	X1	X2	X4
M2	1,42	1,36	1,04	0,56	0,57	0,6
M3	0,58	0,58	0,5	0,55	0,55	0,56
M2RB	1,32	1,20	1,16	0,63	0,62	0,69
MO22	1,22	1,32	1	0,85	0,87	0,84

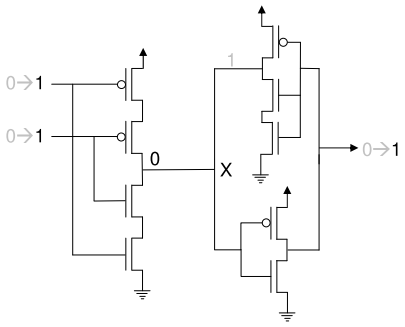


FIGURE 4.2 – Conflit dans une structure de Muller à 2 entrées.

La dissymétrie que l'on observe pour le gain en énergie en montée et en descente, est principalement due à une dissymétrie au sein des portes de la bibliothèque standard de ST Microelectronics. Ces dernières consomment beaucoup plus d'énergie sur les fronts descendants que sur les fronts montants, tandis que les cellules de la TAL consomment une énergie presque identique quel que soit le front (règle 1).

TABLE 4.6 – Gain en consommation statique des cellules TAL.

Portes	Gain en consommation statique		
	X1	X2	X4
M2	0,67	0,75	0,79
M3	0,53	0,53	0,55
M2RB	1,4	1,35	1,09
MO22	0,76	0,78	0,75

Finalement le tableau 4.6 présente le gain en consommation statique des cellules de la TAL. En règle générale, pour limiter le courant de fuite entre le drain et la source d'un transistor, on augmente la longueur « L » du canal artificiellement en ajoutant des transistors en série. Ainsi, nous voyons dans la figure 4.1, que le transistor qui va fuir le plus est, en général, le transistor N qui se trouve dans l'inverseur avant la sortie de la porte. Dans la TAL, ce transistor est dimensionné de telle sorte que le courant de fuite soit le plus petit possible. Pour les cellules à base de portes AO222, ce transistor n'a pas fait l'objet d'optimisations particulières. C'est pour cette raison que le gain en consommation statique dans la TAL

est important (de l'ordre de 30%).

Nous constatons également que les portes de type Muller à 2 entrées avec Reset consomment plus statiquement que les autres types de portes. L'explication de ce phénomène se trouve dans l'inverseur de retour de l'étage de mémoire (transistor N1 sur la figure 4.3). L'utilisation du signal Reset pour piloter ce transistor N1 permet d'accélérer la remise à 0 de la porte par le signal Reset, en désactivant l'inverseur de retour pendant la phase de remise à 0. Cette optimisation conduit à réduire le temps de remise à 0 par un facteur 2, tout en diminuant également l'énergie consommée de 11% pendant la phase de Reset. Malheureusement, dans le cas où toutes les entrées sont à 0 et que le signal de reset est inactif (Reset vaut 1), le transistor N1 piloté par le signal Reset devient passant. Le réseau N de l'inverseur de retour est donc équivalent au seul transistor N2 qui se trouve en position bloqué avec sa source à 0 et son drain à 1. Cette situation étant le pire cas pour les courants de fuite entre drain et source, la consommation statique est très importante.

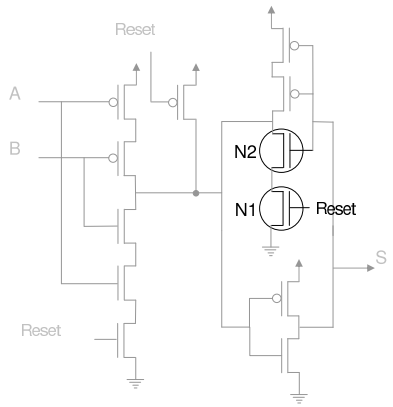


FIGURE 4.3 – Muller à 2 entrées avec Reset.

4.3 TAL2 : Une nouvelle version de TAL

Suite à l'expérience acquise par le développement et l'utilisation de la première version de TAL, une nouvelle bibliothèque TAL2 a été développée au sein du groupe CIS. Pour développer cette nouvelle version, plusieurs objectifs ont été fixés :

– Les circuits asynchrones par leur mode de fonctionnement sont très résistants aux variations de process de fabrication, mais également aux conditions extérieures d'utilisation. Les variations de température, de tension d'alimentation ou de lumière ne constituent pas d'obstacles à la correction fonctionnelle d'un circuit asynchrone. Cependant, la réalisation de circuits basés sur la première version de la bibliothèque TAL (cf chapitre 6), a montré que la baisse de la tension d'alimentation (en dessous de 0,8V) entraînait un mauvais fonctionnement de nos circuits. Ce disfonctionnement était en partie dû à la mauvaise résistance des inverseurs rebouclés aux baisses de tension. L'objec-

tif est donc de modifier les cellules de la bibliothèque afin de garantir la correction fonctionnelle des cellules à faible tension d'alimentation.

– L'utilisation de certains protocoles de communication plus complexes, dans lesquelles les politiques d'acquittement de signaux sont optimisées (par exemple en permettant la remise à 0 des sorties d'un bloc avant la remise à 0 de ses entrées), permet d'augmenter considérablement la vitesse des circuits asynchrones. L'utilisation de ces protocoles nécessite des portes complexes qu'il est possible d'optmiser considérablement par un dessin à la main. Nous avons donc ajouté de nouvelles fonctionnalités intégrant le protocole de communication de type PCHB.

La technologie utilisée pour le design de cette nouvelle version est la technologie 65 nm de ST Microelectronics. Les mêmes vues que celles de la version précédente ont été développées. De plus, une caractérisation complète de la bibliothèque a été réalisée en collaboration avec ST Microelectronics.

Au niveau du dimensionnement des portes, nous avons conservé les mêmes concepts que pour la première version. Cependant afin de garantir le fonctionnement correct de nos cellules à faible tension, nous les avons dessinées de façon entièrement statique. Figure 4.4 nous illustrons une porte de Muller à 2 entrées telle qu'elle se présente dans la nouvelle version de la bibliothèque TAL.

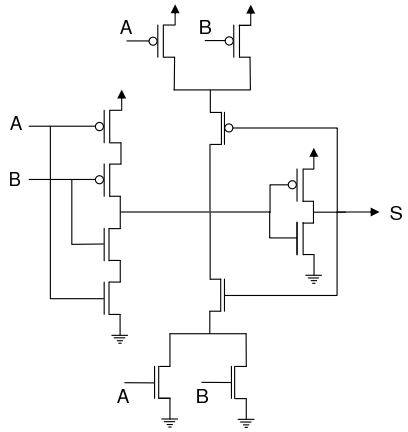


FIGURE 4.4 – Implantation d'une porte de Muller à 2 entrées en version statique.

L'utilisation de cette structure statique nous permet de diminuer la tension d'alimentation des portes de la bibliothèque TAL2 jusqu'à 0,5V. Un second avantage de cette structure entièrement statique est le gain au niveau de la vitesse de fonctionnement des portes. En effet, comme nous l'avons vu plus haut, sur une structure composée d'un étage dynamique suivi d'un étage de maintien, le changement de valeur de toutes les entrées de 0 à 1 (ou de 1 à 0) va provoquer un conflit au niveau du nœud interne située entre la partie dynamique et la mémorisation (nœud X dans la figure 4.2). Le temps de commutation de ce nœud X sera donc plus lent, augmentant le temps de traversée de la porte. Dans la nouvelle structure, le

changement de valeur des entrées de 0 à 1 (ou de 1 à 0) va couper la boucle de retour, évitant le conflit sur le nœud interne et augmentant ainsi la vitesse de commutation de la porte.

4.3.1 Bilan

Dans ce paragraphe, nous allons comparer les portes de cette nouvelle version de la bibliothèque TAL par rapport à leurs implantations à base de portes AO222 en technologie 65nm.

4.3.2 Surface

Du fait que les portes de cette nouvelle bibliothèque soient dessinées de façon statique, le nombre de transistors pour chaque cellule est plus important. Si nous faisons une comparaison du nombre de transistors pour des fonctionnalités identiques par rapport à la version précédente de TAL, nous obtenons :

- Pour la porte de Muller à 2 entrées : 12 transistors pour la version statique contre 9 dans la version dynamique avec un étage de maintien.
- Pour la porte de Muller à 2 entrées avec reset : 14 transistors pour la version statique contre 12 pour la version dynamique avec un étage de maintien.
- Pour la porte de Muller à 3 entrées : 16 transistors pour la version statique contre 11 dans la version dynamique avec étage de maintien.
- Il n'existe pas de portes complexes de type MO22 dans cette version de la bibliothèque.

Le nombre de transistors est légèrement supérieur pour une fonctionnalité identique dans la version statique. Pour les portes avec un nombre d'entrées plus important, cette différence en nombre de transistors va s'accroître dans la version statique du fait de la complexité des derniers étages.

Le tableau 4.7 présente les facteurs de réduction de surface entre les cellules de la TAL2 et les cellules implantées à base de portes AO222. Contrairement à la version précédente de TAL, nous pouvons constater que ce facteur n'est pas toujours supérieur à 1 ; mais également que les écarts de surface sont beaucoup plus faibles.

TABLE 4.7 – Facteurs de réduction de surface obtenus par rapport à des implantations à base de AO222.

Portes	Facteur de réduction en surface (en fonction de la sortance)		
	X4	X18	X35
M2	1,1	0,8	1
M3	1,2	1	1,4
M4	1,6	1,1	1,6
M2RB	1	0,8	1,1

Le dessin des cellules en statique a donc un impact non négligeable sur la surface, le facteur de réduction étant de 1,2 en moyenne sur toute la bibliothèque TAL2 (contre un facteur de réduction de 2 dans TAL1). Cependant, le gain en vitesse est lui aussi beaucoup plus important dans cette version de la bibliothèque, comme nous allons le voir.

4.3.3 Vitesse et Consommation

L'utilisation d'une structure statique, en plus de nous permettre de diminuer la tension d'alimentation des cellules, nous a permis de rendre les cellules de TAL2 plus rapides. Le tableau 4.8 présente le rapport de délais de fonctionnement des portes de la bibliothèque TAL2 et des portes à base de AO222. Le gain moyen sur cette nouvelle version de la TAL est de 33%. Le gain en vitesse est donc 3 fois plus important comparé à l'ancienne version de la TAL.

TABLE 4.8 – Gain en délai des cellules TAL2.

Portes	Gain en délai					
	Montée			Descente		
	X4	X18	X35	X4	X18	X35
M2	0,69	0,57	0,59	0,64	0,58	0,63
M3	0,81	1	0,91	0,87	1,1	1,05
M2RB	0,51	0,50	0,49	0,45	0,39	0,42

Les temps de transition sont également plus courts pour les cellules de la TAL2, avec un gain moyen sur la bibliothèque de 30% (tableau 4.9).

TABLE 4.9 – Gain en temps de transition des cellules TAL2.

Portes	Gain en temps de transition					
	Montée			Descente		
	X4	X18	X35	X4	X18	X35
M2	0,77	0,67	0,67	0,61	0,53	0,57
M3	0,68	0,76	0,66	0,47	0,72	0,72
M2RB	0,78	0,83	0,84	0,80	0,87	0,89

Pour le gain en énergie, une fois de plus les cellules de TAL2 sont plus performantes avec une diminution de l'énergie de 34% par rapport aux cellules à base de AO222.

Finalement, nous évaluons dans le tableau 4.11, le gain en consommation statique des cellules de la TAL avec une tension de 0v en entrée des portes. Le gain moyen pour les 3 fonctionnalités présentées ici est de 30%. Si nous considérons l'ensemble de la bibliothèque, la consommation statique a diminué de

27% par rapport à l'utilisation de cellules AO222.

TABLE 4.10 – Gain en énergie des cellules TAL2.

Portes	Gain en énergie					
	Montée			Descente		
	X4	X18	X35	X4	X18	X35
M2	0,87	0,81	0,77	0,84	0,79	0,69
M3	0,73	0,69	0,53	0,64	0,58	0,43
M2RB	0,61	0,53	0,55	0,67	0,70	0,66

TABLE 4.11 – Gain en consommation statique des cellules TAL2.

Portes	Gain en consommation statique		
	X4	X18	X35
M2	1,28	1,15	0,99
M3	0,52	0,43	0,38
M2RB	0,5	0,51	0,51

Pour les cellules de type Muller à 2 entrées, la consommation statique est plus importante. Dans cette porte, le nombre de transistors en série est moins important en entrée de la porte (2 transistors N en série et 2 transistors P en série), conduisant à un courant de fuite plus important que les cellules possédant plus d'entrées.

4.4 Conclusion

Nous avons présenté dans ce chapitre notre bibliothèque de cellules dédiée à la conception de circuits asynchrones : TAL. Nous avons explicité les choix de développement que nous avons fait, et présenté les plans de conception des principales fonctionnalités de la bibliothèque. Nous avons également présenté les principales caractéristiques en surface, vitesse et consommation de notre bibliothèque, en comparaison des cellules implantées à partir de portes AO222.

Pour la première version de notre bibliothèque, le gain majeur est en surface, puisque l'aire moyenne des cellules de la TAL est 50% plus petite que celle des cellules à base de AO222. Malgré cette forte diminution de la surface, les portes de la TAL sont plus rapides (+10%), consomment moins dynamiquement (-11%) et statiquement (-30%).

Pour la deuxième version de la bibliothèque, nous avons dessiné les cellules de manière statique afin de pouvoir diminuer fortement la tension d'alimentation des cellules. Bien que ces structures statiques

soient plus complexes et comportent un nombre plus grand de transistors, comparé aux cellules de TAL1, la surface moyenne de la bibliothèque est toujours plus petite que les cellules AO222 (-20%). Les autres avantages de ces structures statiques sont une augmentation de la vitesse (+33%), une diminution de la consommation d'énergie (-34%) et une diminution de la consommation statique (-27%).

De plus, les deux versions de la bibliothèque TAL ont été entièrement caractérisées afin d'être utilisables dans un flot de conception standard utilisant des outils commerciaux. Des validations de cette intégration seront présentées dans le chapitre 6.

Comme nous l'avons vu dans les chapitres précédents, les caractéristiques d'un circuit dépendent des caractéristiques de la bibliothèque sur laquelle le circuit a été projeté. Nous venons de présenter la bibliothèque que nous avons développée, intéressons nous maintenant aux algorithmes de projection technologique que nous avons mis au point pour les circuits asynchrones, et leur utilisation avec cette bibliothèque.

Chapitre 5

Projection technologique des circuits Asynchrones QDI

5.1 Introduction

Maintenant que nous avons présenté notre bibliothèque de cellules asynchrones TAL, nous allons présenter les algorithmes de projection technologique que nous avons développé pour les circuits asynchrones QDI, et implantés dans le flot de conception de TAST (figure 5.1).

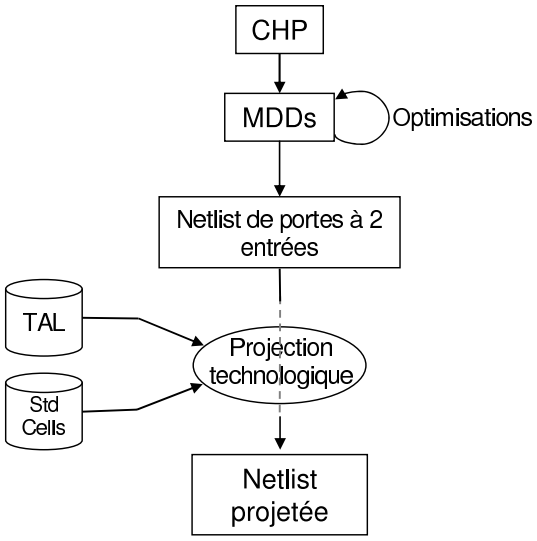


FIGURE 5.1 – Flot de conception de TAST.

Dans un premier temps, nous introduirons les diagrammes de décision multi-valués. C'est à partir de ces diagrammes, généralisation des diagrammes de décision binaires utilisés en synchrone, que nous pourrons obtenir une décomposition de nos circuits qui respecte les critères d'insensibilité aux délais. Ces diagrammes de décision multi-valués sont générés à partir d'un programme CHP décrivant le circuit asynchrone à synthétiser.

Dans un deuxième temps, nous présenterons les algorithmes de décomposition à partir de ces dia-

grammes de décision multi-valués, les algorithmes de partitionnement et les algorithmes de couverture, qui permettront de projeter la netlist décomposée de notre circuit sur une bibliothèque cible. Cette bibliothèque cible pourra être notre bibliothèque TAL ou toute autre bibliothèque de cellules standards.

Enfin nous présenterons des résultats expérimentaux obtenus à partir de ces algorithmes, mais également leur conséquence sur les caractéristiques des circuits.

5.2 Les diagrammes de décision multi-valués

La première étape de la phase de projection technologique consiste à réaliser une décomposition des portes de notre circuit. Comme nous l'avons vu dans le chapitre 3, cette décomposition est nécessaire pour rendre possible l'étape de projection technologique. Cependant, les algorithmes utilisés pour la décomposition de circuits synchrones n'étant pas utilisables avec les circuits asynchrones, nous avons développé notre propre méthode de décomposition.

Afin de garantir l'insensibilité aux délais de nos circuits lors de la phase de décomposition, nous avons élaboré un modèle basé sur les diagrammes de décision multi-valués (MDD). Ce modèle a été développé en collaboration avec Vivian Brégier [41] et présenté dans [21].

5.2.1 Présentation

L'utilisation de ce modèle basé sur les MDDs nous a permis d'élaborer une méthode de synthèse automatique des circuits asynchrones QDI [41]. La structure de MDD est utilisée pour modéliser les processus QDI que l'on souhaite synthétiser. Cette structure a pour principal avantage d'assurer la mutuelle exclusion des signaux codés en multi-rail dans le circuit. Cette propriété est essentielle pour montrer la quasi insensibilité aux délais de nos circuits. Il est également possible de choisir le niveau de décomposition des portes de la netlist à partir de cette modélisation sous forme de MDDs, comme nous le verrons dans les paragraphes suivants.

Cette structure est une généralisation de la structure de BDD (Binary Decision Diagram) [34], qui est utilisée dans de nombreux outils de synthèse de circuits synchrones. La structure d'un MDD est semblable à celle d'un graphe qui spécifie, pour chaque combinaison de valeurs des entrées, la valeur que doit prendre la sortie et en même temps l'acquittement des entrées. Pour cela, un MDD est composé de plusieurs composants qui sont utilisés conjointement comme l'illustrent les figure 5.3 et figure 5.4.

Un MDD est un graphe dirigé acyclique enraciné composé d'un ensemble de nœuds et d'arcs, tel que :

– Trois types de nœuds composent un MDD : les nœuds terminaux, les nœuds non-terminaux et les nœuds fourches.
– Un nœud terminal ne possède pas d'arc en sortie. Il est étiqueté par une variable de sortie et par la valeur de cette sortie.
– Un nœud non-terminal est étiqueté par une variable v d'entrée. Il possède un nombre n d'arcs en sortie. Ce nombre n va dépendre du nombre de valeur que peut prendre la variable v, chaque arc est associé à une valeur.
– Une fourche a un nombre quelconque de fils (éventuellement 0) et n'a pas d'étiquette. Les fourches sont utilisées pour générer la valeur de plusieurs sorties à la fois, ce qui est nécessaire pour réaliser des optimisations sur les MDDs.

On définit un chemin d'un MDD comme une suite de nœuds depuis un nœud racine jusqu'à un nœud terminal. La valeur de la sortie correspond à la valeur du circuit en fonction de la valeur de toutes ses entrées.

5.2.2 Contraintes

La structure de MDD définie ci-dessus est assez générale et ne nous permet pas de contrôler complètement les différentes étapes de synthèse. Il est donc nécessaire de contraindre ce modèle pour obtenir des circuits qui soient insensibles aux délais.

Par exemple, il serait possible, dans cette structure générale, d'émettre plusieurs valeurs à la fois sur un canal de sortie ce qui est bien entendu interdit. De même, il serait possible qu'un nœud soit activé par deux signaux qui ne sont pas mutuellement exclusifs. Le premier signal qui arrive va provoquer une transition en sortie, et le second ne sera donc pas acquitté. Or nous avons vu dans la section 3.4.1 que toute transition doit être acquittée pour que le circuit soit insensible aux délais.

Afin de pallier ces problèmes il est nécessaire de garantir, lors de la construction du modèle, que les rails d'un même canal de sortie sont mutuellement exclusifs, c'est-à-dire qu'un seul d'entre eux peut être à 1 à un instant donné. Définissons tout d'abord la mutuelle exclusion de deux arcs :

Définition 1 : Deux arcs a et b sont mutuellement exclusifs si et seulement si ils respectent une des conditions suivantes :

– a et b sont des arcs distincts sortant d'un même nœud non terminal.
– a est un arc sortant d'un nœud n, b est mutuellement exclusif avec tous les arcs entrants dans n.

Deux chemins sont dits mutuellement exclusifs si leurs terminaux sont mutuellement exclusifs.

Nous pouvons maintenant énoncer la propriété que doivent respecter les MDDs pour assurer que les rails d'un même canal de sortie soient mutuellement exclusifs. Dans ce cas, on dit que le MDD est bien formé.

Propriété : Un MDD est bien formé si et seulement si :

- les arcs entrants d'un nœud sont mutuellement exclusifs.
- Les terminaux étiquetés de la même variable sont mutuellement exclusifs (leurs valeurs doivent différer).

5.2.3 Sémantique des MDD

Maintenant que nous avons décrit le modèle des MDDs, il reste à faire le lien entre la structure présentée et le comportement du processus que nous avons modélisé. C'est ce que l'on appelle la sémantique des MDDs.

Un MDD modélise donc un processus QDI. Chaque arc de ce MDD correspond à un fil dans le circuit. Les entrées, sorties et variables intermédiaires sont codées en utilisant un codage insensible aux délais, modélisant à la fois la requête et la donnée. Les valeurs possibles sont donc « valide avec une valeur » ou « invalide ». Cependant, dans le cas de variable d'acquittement, il n'y a qu'une seule valeur valide.

Définition 2 : Notion d'activation des nœuds et des arcs.

- Les racines d'un MDD sont toujours activées.
- Un nœud qui n'est pas racine est activé si un de ses arcs entrants est activé. Lorsqu'un nœud est activé, il active certains de ces arcs sortants. Supposons le nœud n activé :
  - Si n est un nœud non-terminal, la variable de son étiquette est lue. Si la valeur de cette variable est valide, avec une valeur i, l'arc sortant i est activé.
  - Si n est un nœud fourche, tous ses arcs sortants sont activés.
  - Si n est un nœud terminal, la sortie prend la valeur de n.

En procédant à l'activation successive des différents nœuds du MDD, il est possible de décrire le fonctionnement de notre processus. Ainsi, l'ensemble des variables d'entrées de notre MDD correspond à l'ensemble des canaux d'entrées de notre processus ; l'ensemble des variables de sortie correspond à l'ensemble des canaux de sorties.

5.2.4 MDD direct et MDD d'acquittement

En utilisant les MDDs, nous pouvons modéliser les chemins de données du circuit, c'est-à-dire les chemins qui évaluent les valeurs des sorties et émettent les requêtes.

Cependant, nous voulons également modéliser les chemins d'acquittements des processus. Pour cela, nous évaluons l'acquittement des canaux d'entrée en fonction de l'acquittement des canaux de sortie, et éventuellement de la valeur des canaux d'entrées. Il faut donc ajouter les acquittements des canaux de sorties dans l'ensemble d'entrées de nos MDDs, et ajouter les acquittements des canaux d'entrée dans l'ensemble de sortie des MDD.

5.2.5 Exemple

Afin d'illustrer les définitions que nous venons de présenter, nous proposons de suivre la transformation d'un programme CHP simple en MDD. Le programme CHP commenté suivant permet de modéliser une petite unité arithmétique et logique :

Un canal de sélection Sel permet de définir l'opération entre les deux canaux d'entrée A et B. Si Sel vaut 0, la sortie de l'alu prend la valeur de $A\ ET\ B$. Si Sel vaut 1, la sortie de l'alu prend la valeur de $A\ OU\ B$.

```
process alu_simple
port (Sel: in DI MR[2],   -- déclaration d'un port d'entrée double rail
      A: in DI MR[2],     -- déclaration d'un port d'entrée double rail
      B: in DI MR[2],     -- déclaration d'un port d'entrée double rail
      S: out DI MR[2];)   -- déclaration d'un port de sortie double rail

variable:   sel : MR[2];
            a   : MR[2];   -- declaration de variables
            b   : MR[2];   -- double rail

*[
Sel?sel;                    -- lecture de la variable sel sur le canal Sel
@[
    sel = '0' => A?a; B?b; S!a and b;   -- Si sel vaut 0, S vaut <<a and b>>
    sel = '1' => A?a; B?b; S!a or b;   -- Si sel vaut 1, S vaut <<a or b>>
]
]
end
```

FIGURE 5.2 – Programme CHP décrivant une petite UAL.

Les variables, les canaux d'entrée et de sortie sont définis comme des double-rail. En CHP, nous les décrivons de manière équivalente comme des canaux multi-rail de taille 2. L'opérateur « ? » correspond à la lecture sur un canal ; ainsi la notation « A ?a » signifie que la variable a prend la valeur lue sur le canal A. L'opérateur « ! » correspond à l'écriture sur un canal ; ainsi la notation « S !a and b » signifie

que la valeur de l'expression « $a\ and\ b$ » est émise sur le canal S.

Intéressons nous maintenant aux MDDs correspondants à ce programme CHP. La figure 5.3 illustre le MDD direct, qui évalue la valeur des sorties en fonction de la valeur des entrées.

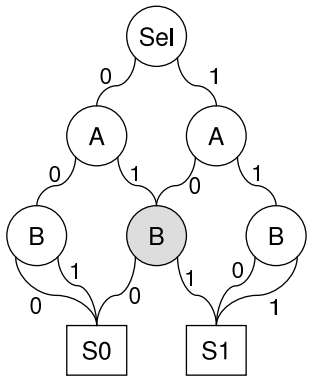


FIGURE 5.3 – MDD direct correspondant au programme CHP.

Il est facile de comprendre le fonctionnement de ce MDD. Par exemple, si Sel vaut 0, la sortie S vaut la valeur de « $a\ and\ b$ ». La sortie vaut donc 1 si a et b valent 1 et 0 sinon.Nous obtenons un MDD identique à un BDD, du fait de l'utilisation d'un codage double-rail. Les valeurs des arcs en sortie de nœud ne pouvant prendre que les valeurs 0 et 1. Cela aurait été différent en utilisant un codage multi-rail d'une taille supérieure à 2, les arcs de sortie des nœuds pouvant prendre des valeurs supérieures à 1.

Il est également intéressant de noter qu'une optimisation a été faite sur ce MDD. Dans deux branches du MDD, deux nœuds étaient identiques, ils ont donc été fusionnés (nœud grisé). Cette optimisation va permettre de diminuer le nombre d'états de notre MDD, et ainsi diminuer le coût des algorithmes qui lui seront appliqués par la suite.

Comme nous l'avons dit précédemment, il est également nécessaire de calculer les acquittements des entrées en fonction de l'acquittement des sorties. Dans cet exemple, ce calcul est très simple. En effet, nous voyons que dans chaque chemin du MDD, toutes les variables sont lues. Nous pouvons en conclure que quel que soit le chemin emprunté, l'acquittement du canal de sortie va permettre d'acquitter tous les canaux d'entrées, puisque tous ces canaux ont participé à l'évaluation de la sortie. Nous obtenons le MDD d'acquittement de la figure 5.4, dans lequel le signal d'acquittement du canal de sortie S_{ack} va provoquer l'acquittement de tous les canaux d'entrées : A_{ack}, B_{ack} et Sel_{ack}.

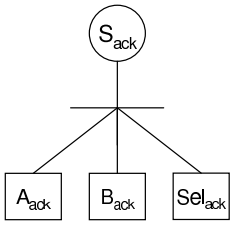


FIGURE 5.4 – MDD d'acquittement.

5.3 Projection technologique des circuits QDI

Dans ce chapitre, nous présentons les algorithmes de projection technologique que nous avons développés pour les circuits asynchrones QDI. L'objectif est de présenter précisément les trois phases de la projection technologique que nous avons présentées dans le chapitre 3 : une phase de décomposition, une phase de partitionnement et une phase de couverture.

5.3.1 Décomposition

La phase de décomposition que nous avons développée, est basée sur l'utilisation des MDDs que nous venons de présenter. Les algorithmes présentés permettent de synthétiser un circuit à partir de sa représentation sous forme de MDD. C'est lors de cette phase de synthèse qu'il est possible d'influer sur la décomposition des portes suivant un degré de granularité fixé.

5.3.1.1 Algorithmes de synthèse

Voici une première description, sous forme de règles, de l'algorithme de synthèse qui nous permet d'obtenir une netlist de portes d'un circuit à partir de sa description sous forme de MDDs :

- Chaque arc contenu dans le MDD correspond à un rail de notre circuit.
- Chaque chemin du MDD des entrées vers une sortie correspond à une porte de Muller.
- Toutes les portes de Muller participant à l'évaluation de la même sortie sont regroupées par une porte OR.

Ce premier algorithme de synthèse permet de synthétiser des circuits de type DIMS pour « Delay Insensitive Minterms Synthesis » [86]. Les circuits DIMS sont des circuits asynchrones QDI par construction. Bien que robustes, ces circuits sont lents. En effet, le fonctionnement de ces circuits consiste à évaluer en entrée toutes les combinaisons des variables communément appelées minterms. Chacune de ces combinaisons correspond aux entrées d'une porte de Muller. Ensuite les différentes portes de Muller sont combinées par des portes OR en fonction de leur participation à l'évaluation d'une sortie. Cette configuration permet de ne faire commuter à chaque instant qu'une seule porte de Muller, et ainsi de faciliter l'acquittement des entrées.

Reprenons l'exemple d'alu précédent pour illustrer cet algorithme de synthèse.

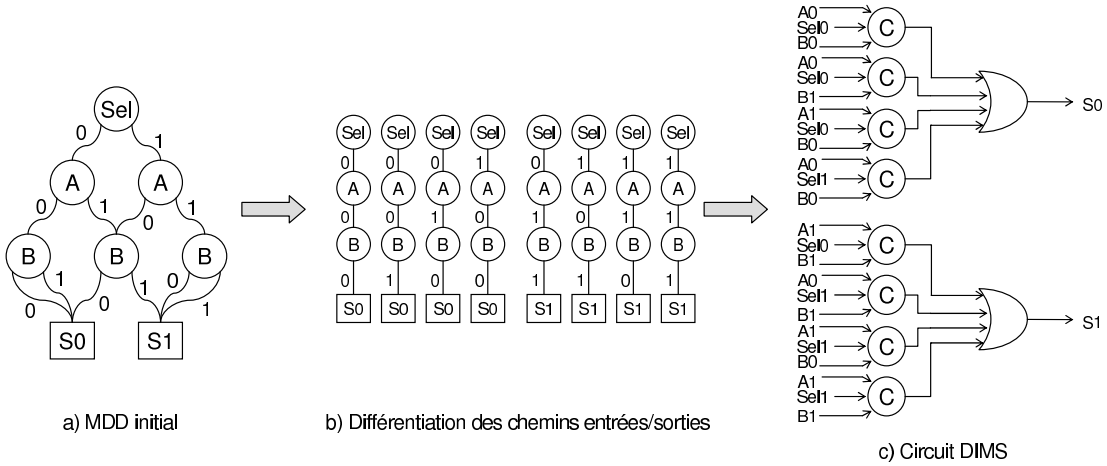


FIGURE 5.5 – Génération d'un circuit asynchrone QDI de type DIMS.

Dans la figure 5.5, nous représentons le passage du MDD vers un circuit de type DIMS. Dans un premier temps, nous différencions les différents chemins entre les entrées et les sorties du circuit dans le MDD (figure 5.5b). Une fois que cette différentiation est faite, nous associons à chaque chemin une porte de Muller. Les portes de Muller sont ensuite regroupées par une porte OR en fonction de la sortie qu'elles participent à faire commuter (figure 5.5c).

Il apparaît évident qu'une telle méthode de synthèse ne permet pas de résoudre les problèmes de décomposition. En effet, s'il est possible de décomposer directement les portes OR sans introduire d'aléa, ce n'est pas le cas des portes de Muller qui possèdent plus de deux entrées (comme nous l'avons vu dans 3.4.1). Cependant, il est intéressant de visualiser cette forme DIMS, pour comprendre le fonctionnement de nos algorithmes de synthèse.

5.3.1.2 Décomposition

Une adaptation de l'algorithme de synthèse va donc permettre de générer directement un circuit décomposé qui respecte par construction l'insensibilité aux délais. Pour cela, il suffit de modifier les règles définies précédemment pour décrire notre algorithme :

- Chaque arc contenu dans le MDD correspond à un rail de notre circuit.
- Les arcs dirigés vers un même nœud sont regroupés par une porte OR.
- Chaque nœud non-terminal est représenté par un ensemble de portes de Muller à deux entrées qui synchronisent les arcs entrants du nœud. Les arcs sortants correspondent aux sorties de ces portes de Muller.

La figure 5.6 illustre de façon très simple ces dernières règles, en considérant un nœud de MDD comportant 3 arcs en entrée et 3 arcs en sortie.

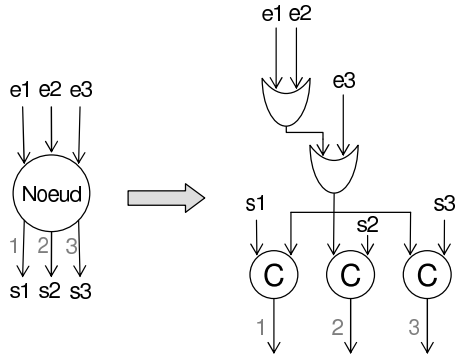


FIGURE 5.6 – Règle de synthèse d'un nœud de MDD.

Avec cette méthode, les portes OR mais également les portes de Muller sont limitées à 2 entrées, lors de la phase de synthèse. De plus, la propriété de mutuelle exclusion des rails du MDD assure que la construction conserve son insensibilité aux délais. Nous avons donc réussi à obtenir une décomposition qui n'insère pas d'aléa dans le circuit.

Illustrons maintenant l'application de ces règles supplémentaires sur le circuit présenté précédemment.

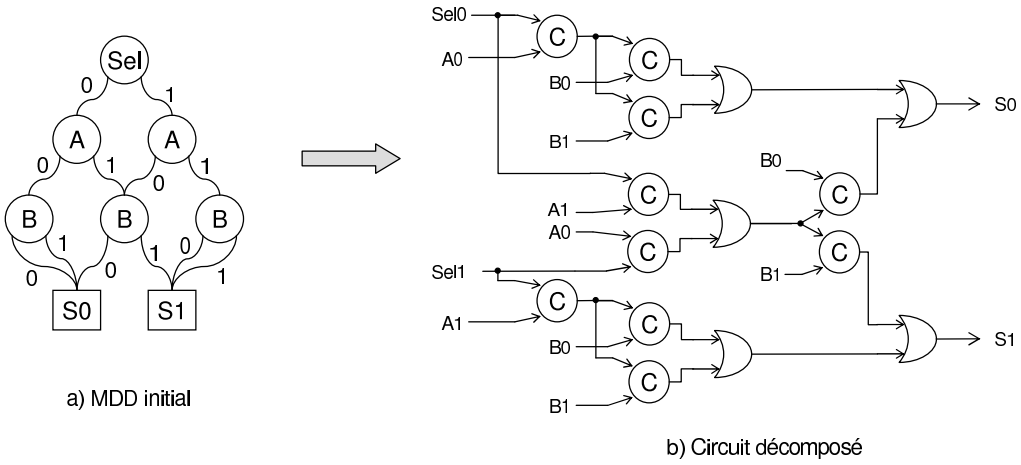


FIGURE 5.7 – Génération d'un circuit asynchrone QDI décomposé.

Le circuit obtenu figure 5.7b est composé uniquement de portes de Muller à 2 entrées et de portes OR à 2 entrées. La décomposition obtenue respecte l'insensibilité aux délais du circuit par construction.

5.3.2 Partitionnement

Pour la phase de partitionnement, l'idée est de découper le circuit décomposé en sous-circuits comportant plusieurs entrées et une seule sortie. Dans nos algorithmes, nous avons choisi de définir des cônes logiques à partir des sorties primaires. Nous regroupons donc les portes logiques qui précèdent une sortie primaire en un ensemble que nous enlevons ensuite de la liste des portes. Il y a donc une dépendance de la solution vis-à-vis de l'ordre du choix des sorties. Cependant cette solution a l'avantage d'être facile à implanter.

Toutefois, il est nécessaire de s'attarder sur le cas des fourches. Dans le circuit décomposé, si une fourche se trouve en sortie d'une porte, il est évident que cette fourche ne pourra pas être incluse dans une porte complexe lors de la phase de couverture, puisqu'elle deviendrait un nœud interne de la cellule « couvrante » et il ne serait plus possible d'y accéder. L'algorithme de partition devra également réaliser une découpe au niveau des fourches pour pallier ce problème.

Afin d'illustrer cette phase de partitionnement, nous continuons sur notre exemple et montrons les étapes successives de la partition du circuit.

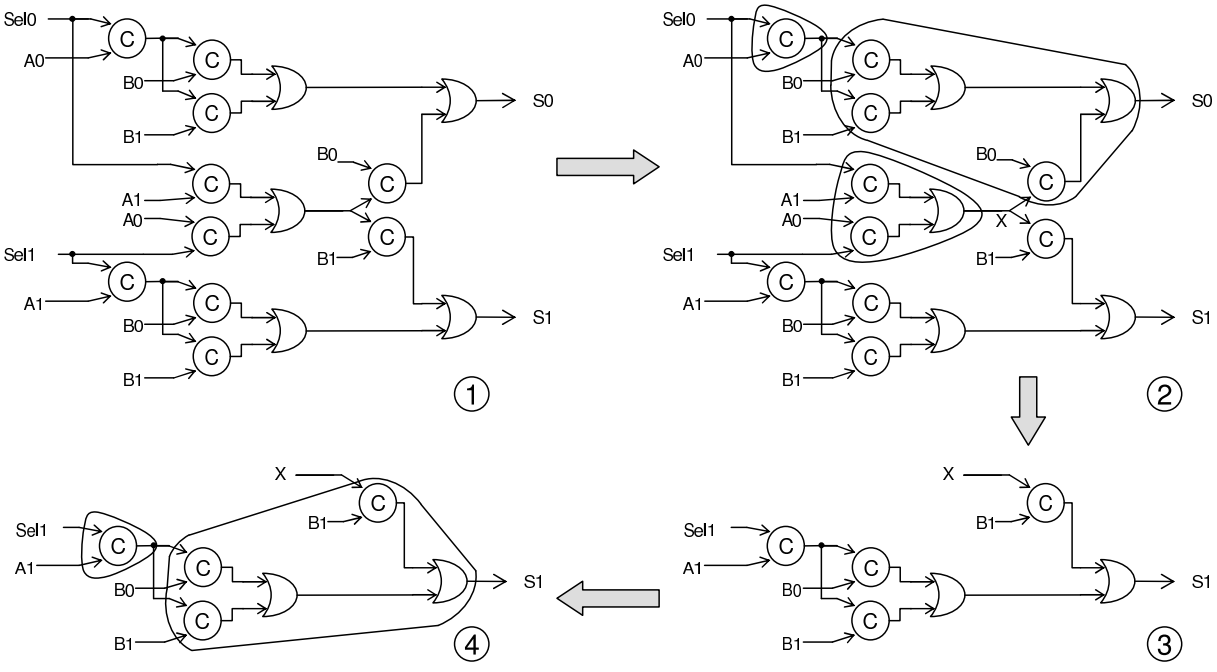


FIGURE 5.8 – Description des différentes étapes de la partition.

La figure 5.8 représente les différentes étapes du partitionnement de notre circuit. A partir du circuit initial, nous choisissons une des sorties primaires, ici S_0. A partir de cette sortie, nous regroupons le plus grand nombre de portes possible. Comme nous l'avons signalé, il faut limiter les ensembles en s'arrêtant

aux fourches rencontrées. Nous voyons à l'étape 2 que 3 ensembles ont été définis. Ils correspondent aux plus grands regroupements de portes possibles, en tenant compte des fourches qui se trouvent dans le circuit.

Une fois que nous avons définis ces premiers ensembles, nous enlevons du circuit les portes qui y sont regroupées. L'étape 3 est une représentation du circuit dans laquelle nous avons enlevé les portes regroupées à l'étape 2.

Finalement nous relançons l'algorithme de partitionnement sur les sorties restantes. L'étape 4 représente les regroupements de portes à partir de la sortie S_1, en prenant toujours en compte les fourches.

L'algorithme se termine lorsque toutes les portes font partie d'un ensemble. Évidemment, c'est sur ces ensembles que nous venons de créer que va se dérouler l'étape de couverture par une bibliothèque. Nous voyons clairement que le gain en performances ne sera pas négligeable, les sous-circuits obtenus étant bien plus petits en nombre de portes.

5.3.3 Couverture

Il reste à présenter la dernière étape de la projection technologique, la phase de couverture. De nombreuses recherches ont été menées sur cette étape de la projection technologique dans la conception des circuits synchrones. L'objectif de ces recherches étant bien entendu d'augmenter les performances des algorithmes.

Nous présentons clairement, dans ce paragraphe, les algorithmes de couverture que nous avons développés. Ces algorithmes sont adaptés des travaux de M. Zhao et S. Sapatnekar présentés dans [96]. En effet, toutes les recherches menées sur la couverture des circuits synchrones a permis d'obtenir des algorithmes très performants, qu'il a fallut modifier pour les adapter aux circuits asynchrones.

5.3.3.1 Présentation de l'algorithme

Comme nous l'avons vu dans le chapitre 3.3.3, il existe deux approches pour la couverture d'un circuit : une approche booléenne et une approche structurelle. Nous avons développé des algorithmes simples et rapides basés sur l'approche structurelle. L'idée générale de l'approche structurelle est de décomposer le circuit initial et la bibliothèque cible en un ensemble commun de fonctions simples, appelées *fonctions de base*. L'inconvénient majeur de cette approche structurelle est que la décomposition d'une bibliothèque en fonctions simples n'est pas unique. Il peut y avoir un nombre important de décompositions pour chaque cellule, particulièrement pour les cellules ayant beaucoup d'entrées. Le temps passé à couvrir le réseau initial par ces nombreuses décompositions peut devenir très important. La tendance

actuelle de développer des cellules comportant beaucoup d'entrées augmente drastiquement le nombre de décompositions, et la phase de couverture devient extrêmement coûteuse en temps de calcul.

La méthode développée ici est simple en terme d'implantation et efficace en termes de vitesse et d'occupation mémoire. Le principe de cette méthode est de créer une relation entre la couverture d'un nœud et la couverture de ses fils dans le réseau. Etant donné un nœud du réseau à couvrir, une couverture valide de ce nœud peut être trouvée à partir des couvertures valides de ses fils et du type du nœud. Cette propriété peut être utilisée pour limiter de nombreuses redondances dans les algorithmes classiques de couverture basés sur une approche structurelle. Pour cela, chaque structure canonique d'une cellule (ou d'une partie d'une cellule) qui peut être représentée sous forme d'arbre, est abstraite sous forme d'index, appelé *index modèle*. La relation structurelle entre toutes ces structures (ou sous-structures) est représentée sous forme de table, appelée *table modèle*. L'utilisation de tables modèles basées sur des représentations canoniques permet de dépasser les problèmes dus au nombre important de décompositions.

Pour l'instant, la méthode que nous utilisons ne permet de traiter que les cellules qui peuvent être représentées sous forme d'arbre. Les cellules qui ne peuvent être représentées que sous forme de DAG ne sont pas encore considérées. Dans notre approche, la décomposition des cellules de la bibliothèque se fait suivant les fonctions de base AND, OR et Muller.

Nous allons présenter les différentes étapes de l'algorithme de couverture : dans un premier temps, nous présenterons les tables modèles et la manière de les générer à partir de la bibliothèque cible ; nous utiliserons ensuite ces tables modèles pour réaliser la couverture du réseau initial.

5.3.3.2 Tables modèles

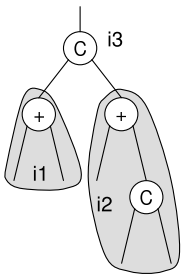
Le but des tables modèles est de présenter sous forme simple les relations structurelles entre les cellules d'une bibliothèque. Les tables modèles sont constituées de 5 parties : une table modèle pour chaque fonction de base AND, OR et portes de Muller ; deux tableaux distincts *estPorte* et *est PorteInv*. Nous avons choisi ces fonctions de base suivant nos besoins, cependant il est possible de les changer en fonction des nécessités.

Pour chaque cellule de la bibliothèque cible pour laquelle une représentation sous forme d'arbre est possible, nous créons un arbre unique à plusieurs entrées. Par la suite, nous allons assigner un index modèle unique pour chacune de ces structures (ou sous-structures) d'arbre. Ainsi chaque index modèle va correspondre à une cellule ou la sous-structure d'un arbre correspondant à une cellule. Comme notre modèle ne prend en compte que les cellules qui peuvent être représentées sous forme d'arbre, il est nécessaire que les entrées des cellules soient logiquement indépendantes. Dans le cas contraire, c'est-

à-dire que les entrées sont logiquement dépendantes et que la structure de la cellule est un DAG, nous n'obtenons pas une représentation canonique de la cellule.

Les tables AND, OR et Muller sont les parties les plus importantes des tables modèles. Elles illustrent le fait qu'un arbre peut être construit en appliquant des opérations AND, OR ou Muller sur deux autres structures d'arbre. Dans ces tables, les colonnes et les lignes sont des index modèles. Toute entrée (i, j) dans une table AND (OR, Muller) correspond à l'index modèle de l'arbre construit en appliquant une opération AND (OR, Muller) sur les structures d'arbre représentées par les index modèle i et j. Dans le cas où un tel arbre n'existerait pas, c'est-à-dire qu'il ne correspondrait à aucune structure ou sous-structure d'arbre des cellules, l'entrée dans la table aurait la valeur invalide. Ces tables ont une taille de $M \times M$, où M est le nombre d'index modèles. Dans la suite, nous utiliserons la notation $[x, y, z]$ pour représenter une entrée dans les tables modèles. x est une variable indiquant si l'entrée se trouve dans la table AND, OR ou Muller. y et z indiquent la position de l'entrée dans la table. Physiquement, la notation $[x, y, z]$ correspond à un arbre dont la racine est de type x et dont les fils sont y et z. De plus, les tables AND, OR et Muller sont symétriques, du fait de la propriété de symétrie des opérateurs AND, OR et Muller.

Prenons un exemple de table Muller (figure 5.9). Dans cet exemple, la composition par une Muller de deux arbres d'index modèle i1 et i2, crée un arbre i3. Nous avons donc une modification de l'entrée $[Muller, i1, i2]$ à la valeur i3. Nous pouvons remarquer que l'entrée $[Muller, i2, i2]$ est invalide (ici noté « - ») car cette structure ne correspond à aucune structure ou sous-structure de cellules.



Muller	i1	i2
i1		i3
i2	i3	-

FIGURE 5.9 – Exemple de table de Muller.

Comme nous l'avons vu, nous créons un arbre unique à plusieurs entrées pour chaque cellule de la bibliothèque cible. Cependant les tables que nous utilisons sont à deux dimensions, il est donc nécessaire de décomposer le nœud racine de l'arbre que l'on considère en nœud à 2 entrées. Bien entendu, plusieurs décompositions peuvent être obtenues pour un même nœud racine. Une énumération exhaustive des décompositions de chaque nœud est faite et les éléments correspondants sont insérés dans les tables modèles. L'exemple de décomposition présenté figure 5.10 (inspirée de [96]) illustre l'énumération des

décompositions pour un arbre i5. Le nombre de décompositions est de 5, il y aura donc 5 entrées avec i5 dans la table OR. En effet, nous pouvons remarquer que les nœuds i2 et i3 sont identiques, ce qui permet de réduire le nombre de décompositions. Dans le cas où i2 et i3 seraient différents, le nombre d'énumérations aurait été de 7.

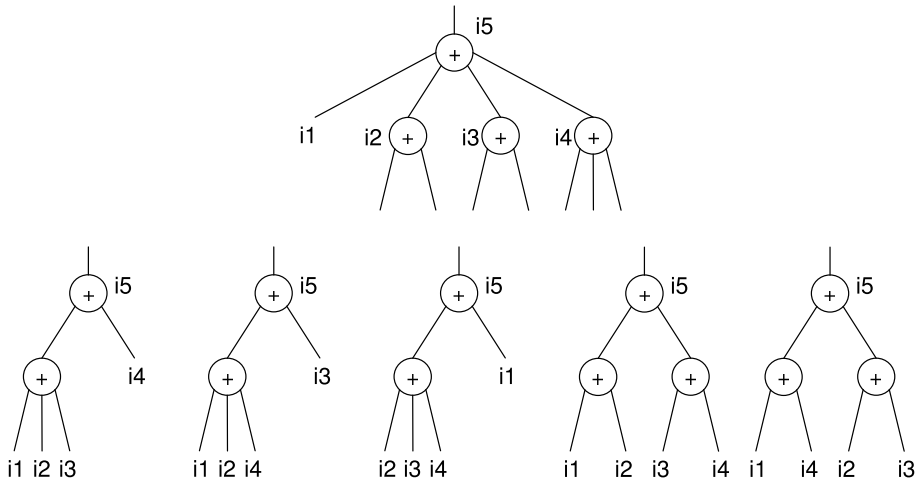


FIGURE 5.10 – Exemple de décomposition.

Les tableaux *estPorte* et *estPorteInv* complètent les tables modèles, ils permettent d'indiquer la relation entre une structure et une cellule. En effet, il est possible que certaines structures identifiées lors du remplissage des tables AND, OR et Muller, ne correspondent ni à une cellule, ni à une sous-structure de cellule. Le tableau *estPorte* est un tableau de dimension M, utilisé pour indiquer si l'index modèle d'une structure correspond à une cellule ou non. De plus, si la cellule en cours d'analyse contient des inverseurs, ces derniers sont d'abord poussés vers la racine, puis les inverseurs restants sont poussés vers les feuilles. Le tableau de dimension M, *estPorteInv,* permet de signaler la présence d'un inverseur à la racine d'un arbre.

5.3.3.3 Algorithmes

Les tables modèles sont remplies en prenant chaque porte d'une bibliothèque une à une ; les structures d'arbre des cellules de la bibliothèque sont créées et analysées et les entrées correspondantes des tables sont ajoutées. La figure 5.11 présente l'algorithme de génération de ces tables modèles. Une fois qu'une structure d'arbre est créée pour une cellule, nous cherchons l'index modèle qui lui correspond. Pour cela, nous utilisons la procédure *tableModèle* décrite figure 5.12.

L'objectif de la procédure *tableModèle* est de renvoyer un index modèle pour un arbre fourni en entrée, et de remplir si nécessaire les tables modèles avec cet index.

L'index modèle d'une feuille est appelé *indexFeuille*, si la feuille est précédée d'un inverseur, l'in-

```
Algorithme de génération des tables modèles
Entrée : Une librairie
Sortie : Table modèle AND, OR et Muller ; tableaux estPorte, estPorteInv.
1.  Initialisation de toutes les entrées des tables modèles avec la valeur invalide.
2.  Pour toutes les cellules de la librairie
3.      Créer un arbre A à base de portes AND/OR/Muller/NOT
4.      indexModèle = tableModèle(A)
5.      estPorte[indexModèle] = 1
6.      S'il y a un inverseur à la racine de la structure
7.          estPorteInv[indexModèle] = 1
```

FIGURE 5.11 – Algorithme de génération des tables modèles.

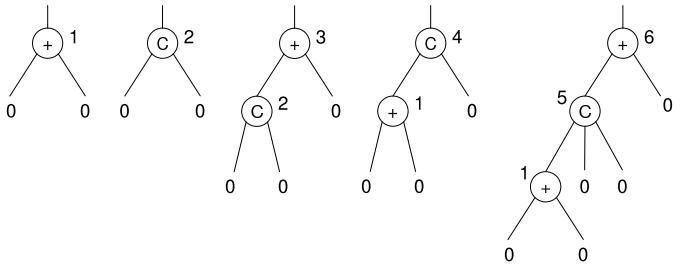
dex s'appelle *invIndexFeuille*. Ces deux index sont les seuls réservés, tous les autres étant évalués par l'algorithme. Pour chaque décomposition à deux entrées d'une cellule de la bibliothèque, la procédure *tableModèle* est appelée récursivement afin de remplir les tables modèles. Dans le cas où un arbre A a déjà été visité, la ligne 6 de la figure 5.12 est utilisée et l'index correspondant à l'arbre est retourné. Dans le cas où un arbre isomorphe à l'arbre A a déjà été visité, une seule décomposition de A est prise en compte (lignes 12-13). Dans tous les autres cas, un nouvel index modèle est assigné à A, et l'énumération des décompositions de A est réalisée (lignes 8-14).

```
Procédure tableModèle(arbre A)
Entrée : Arbre A AND/NOT/Muller/NOT
Sortie : Index modèle de l'arbre en entrée. Entrées dans la table modèle.
Variable globale : nouvelIndex (index modèle)
01.  Si T est une feuille
02.      retourner indexFeuille
03.  Si T est un inverseur avant une feuille
04.      retourner invIndexFeuille
05.  Si indexModèle(A) est une valeur valide
06.      retourner indexModèle(A)
07.  typeNoeud = type du nœud racine de A
08.  Pour chaque décomposition à 2 entrées du nœud racine A appelée A'
09.      indexG = tableModèle(filsGauche(A'))
10.      indexD = tableModèle (filsDroit(A'))
11.  Si [typeNoeud, indexG, indexD] est une valeur valide
12.          indexModèle(A) = [typeNoeud, indexG, indexD]
13.          retourner indexModèle(T)
14.      indexModèle(T) = [typeNoeud, indexG, indexD] = nouvelIndex
15.      incrémenter nouvelIndex
16.      retourner indexModèle(T)
```

FIGURE 5.12 – Procédure tableModèle.

5.3.3.4 Exemple de bibliothèque

Pour illustrer les algorithmes de génération des tables modèles, nous présentons figure 5.13 (inspirée de [96]) une bibliothèque composée de 5 cellules, et les tables modèles associées.



C	0	1	2	3	4	5
0	2	4	-	-	5	-
1	4	-	5	-	-	-
2	-	5	-	-	-	-
3	-	-	-	-	-	-
4	5	-	-	-	-	-
5	-	-	-	-	-	-

OR	0	1	2	3	4	5
0	1	-	3	-	-	6
1	-	-	-	-	-	-
2	3	-	-	-	-	-
3	-	-	-	-	-	-
4	-	-	-	-	-	-
5	6	-	-	-	-	-

estPorte	
1	1
2	1
3	1
4	1
5	0
6	1

estPorteInv	
1	0
2	0
3	0
4	0
5	0
6	0

FIGURE 5.13 – Exemple de génération de tables modèles.

Dans l'exemple, l'*indexFeuille* est noté « 0 ». Toutes les structures correspondent à une porte, excepté la structure 5 : tous les éléments du tableau *estPorte* sont à « 1 », excepté l'index 5. Aucune de ces portes n'est une porte inverseuse : le tableau *estPorteInv* ne contient que des « 0 ». Dans l'exemple, l'index modèle 5 peut correspondre à la combinaison d'un arbre d'index modèle 1 et d'un arbre d'index modèle 2 par une porte de Muller, mais également à une combinaison des arbres d'index modèles 4 et 0 par une porte de Muller. Nous fixons donc les valeurs des entrées $[Muller, 1, 2]$, $[Muller, 2, 1]$, $[Muller, 4, 0]$ et $[Muller, 0, 4]$ à 5.

5.3.3.5 Couverture

Nous avons présenté les étapes préliminaires dans lesquelles nous avons traité les relations structurelles entre les cellules cibles. Nous allons maintenant présenter les algorithmes de couverture qui utilisent les relations structurelles entre les cellules mais également la relation entre la couverture d'un nœud du réseau initial et la couverture de ses fils.

Dans les méthodes traditionnelles de couverture, chaque nœud du réseau initial est traité individuellement causant des opérations redondantes et une baisse des performances. Dans cette approche, la couverture d'un nœud va dépendre de la couverture de ses fils.

L'ensemble de couverture d'un nœud N d'un réseau sujet est composé de tous les index modèles

correspondant aux arbres isomorphes au réseau sujet de racine N. Etant donné le type NT d'un nœud du réseau sujet, G l'ensemble de couvertures du fils gauche et D l'ensemble de couverture du fils droit, l'index modèle de chaque élément du produit croisé $G \times D$, (g, d) est vérifié par rapport à l'entrée $[NT, g, d]$ des tables modèles. Si un élément a un index valide, son index fait partie de l'ensemble de couverture du nœud N. De plus, tous les nœuds du réseau sujet peuvent être vus comme des feuilles, nous ajoutons donc systématiquement l'*indexFeuille* à l'ensemble de couverture de chaque nœud. L'ensemble de couverture d'un nœud est donc constitué de deux types d'index modèles : les index qui représentent une cellule et ceux qui représentent une sous-structure d'une cellule. Dans le premier cas, il est nécessaire d'évaluer le coût associé à cette cellule : surface, consommation, vitesse…; c'est cela que réalise la fonction *evalCoutCellule*. Dans le deuxième cas, l'index correspondant à une sous-structure est ajouté à l'ensemble de couverture car il pourra être utile pour trouver un ensemble plus important de couvertures pour le père du nœud.

Afin de compléter cette description de la méthode de couverture, la figure 5.14 décrit l'algorithme de couverture.

Procédure Matching(nœud N)
Entrée : Ensemble de couverture des fils du noeud N ; les tables modèles **Sortie :** Ensemble de couverture du noeud N.
1. NT = type de N. 2. Pour tous les matchs du fils gauche, G 3. Pour tous les matchs du fils droit, D 4. k = [NT, g, d] 5. si (k est valide) 6. insérer k dans l'ensemble de couverture de N 7. si estPorte[k] == 1 8. evalCoutCellule(k) 9. insérer indexFeuille et invIndexFeuille dans l'ensemble de matchs de N

FIGURE 5.14 – Algorithme de couverture.

Pour terminer la présentation de nos algorithmes, illustrons cette étape de couverture en utilisant notre exemple d'alu des paragraphes précédents. Les réseaux utilisés pour la phase de couverture sont ceux obtenus après la phase de partitionnement (figure 5.15).

Pour effectuer la couverture de notre réseau, nous utilisons la même bibliothèque que celle présentée dans la section 5.3.3.4. Nous avons reporté les tables modèles figure 5.15a, les réseaux sujets obtenus après la phase de partitionnement se trouvent figure 5.15b. Les éléments entre accolades correspondent aux ensembles d'index modèles de chaque nœud du réseau. Comme nous l'avons vu, chacun de ces ensembles comporte l'*indexFeuille*, ici représenté par la valeur « 0 ». Pour obtenir les ensembles de

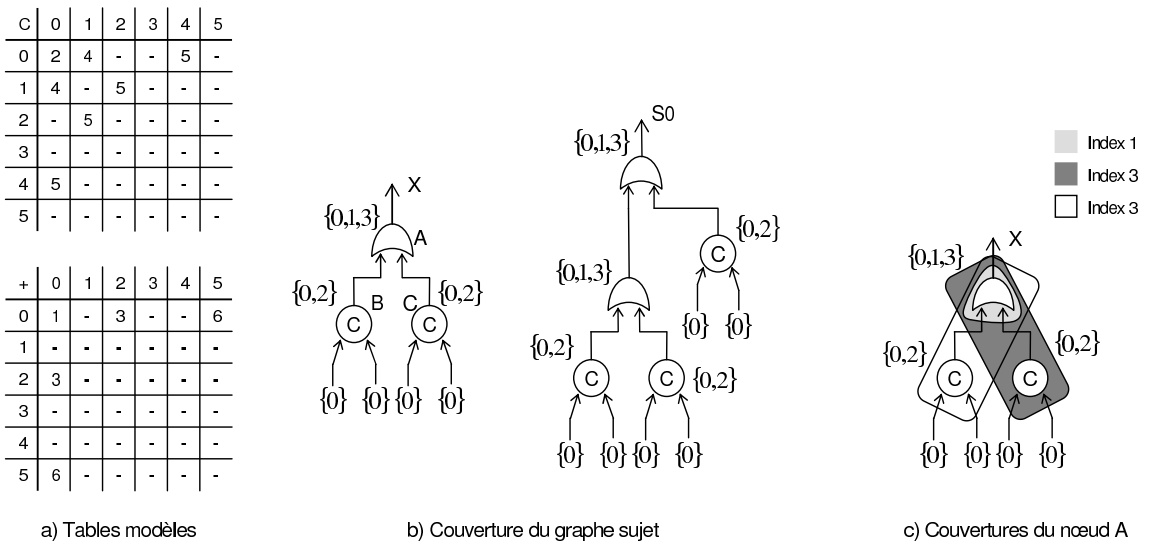


FIGURE 5.15 – Exemple de la méthode de couverture.

couverture des nœuds, nous réalisons un parcours depuis les entrées primaires vers les sorties primaires, en appliquant l'algorithme de la figure 5.14. Par exemple, pour le nœud A, en recherchant dans la table OR les éléments de l'ensemble $\{0, 2\} \times \{0, 2\}$, nous obtenons l'ensemble de couverture $\{0, 1, 3\}$. En effet, l'entrée $[OR, 0, 2]$ retourne l'index modèle 3, tandis que l'entrée $[OR, 0, 0]$ retourne l'index modèle 1 ; les autres combinaisons retournant la valeur invalide. Finalement, comme ce nœud A peut être vu comme une feuille, nous ajoutons « 0 » à l'ensemble de couverture.

De plus, nous pouvons voir figure 5.15c les possibilités de couverture du nœud A, à savoir les index modèle 1 et 3. Pour l'index modèle 3 il existe deux possibilités de couverture puisque les deux entrées $[OR, 0, 2]$ et $[OR, 2, 0]$ retournent le même index modèle. Il apparaît donc nécessaire de connaître à tout instant, pour un même index modèle, les différentes couvertures possibles, en différenciant systématiquement les couvertures du fils gauche et du fils droit.

Pour finir, nous devons définir un critère afin de départager plusieurs solutions de couverture, ce qui est réalisé par la fonction *evalCoutCellule* dans l'algorithme figure 5.14. Nous avons choisi dans nos algorithmes de privilégier la surface, mais la méthode utilisée pour évaluer la surface serait la même pour des critères différents tels que la consommation ou la vitesse. L'évaluation du coût pour une solution de couverture peut être optimisée grâce à notre méthode de couverture. Lors du calcul de la couverture d'un nœud, il est possible d'évaluer son coût en fonction du coût de la couverture de ses fils, et ainsi éviter la redondance des calculs.

Par exemple, dans la figure 5.16, le coût de la couverture W choisie pour le nœud A est : $C(Y) + C(X) + C(Z) + C(W)$ où $C(Y) + C(X) + C(Z)$ correspond au coût des fils de A, et $C(W)$ correspond

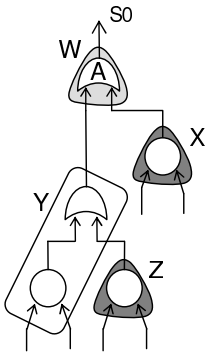


FIGURE 5.16 – Exemple d'évaluation du coût d'une couverture.

au coût de la couverture de A. Ainsi le coût de la couverture W peut se noter : $C_{fils\_gauche} + C_{fils\_droit} + C(W)$, évitant ainsi de dupliquer le calcul de $C(Y) + C(X) + C(Z)$ déjà réalisé lors de l'évaluation du coût de Y et de X. Par contre, dans le cas où le critère de choix dépend des pins ou de la charge, et que le calcul du coût d'un nœud ne dépend pas du coût des fils, il est difficile de réaliser une telle optimisation.

5.4 Résultats

Les algorithmes ayant été présentés, nous allons maintenant nous intéresser à la validation de nos travaux. Pour cela nous présentons dans cette section les résultats obtenus sur deux circuits : une unité arithmétique et logique en base 4 et une unité logique d'un MIPS 4k 32 bits.

5.4.1 Une unité arithmétique et logique en base 4

5.4.1.1 Présentation de l'ALU

Dans le but d'avoir une première évaluation de nos algorithmes de synthèse de circuits asynchrones QDI et de pouvoir d'autre part, comparer une architecture asynchrone QDI à une architecture synchrone équivalente, nous avons développé un bloc de base d'une ALU en base 4 (figure 5.17). Ces résultats ont d'ailleurs fait l'objet d'une publication dans [42].

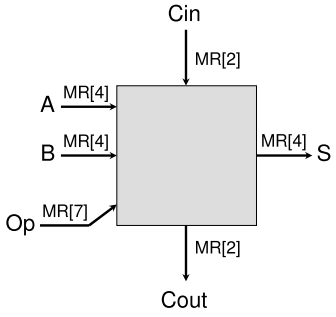


FIGURE 5.17 – Bloc ALU.

Le bloc d'ALU calcule une fonction Op entre deux opérateurs A et B, en utilisant si nécessaire les

entrées/sorties C_{in} et C_{out} (addition et soustraction).

L'utilisation d'une base 4 sert à démontrer que notre méthode est valide pour un codage différent du simple double-rail. L'ALU peut réaliser 7 opérations : addition, soustraction, et, ou, ou exclusif, négation et not ; l'opérateur Op utilise donc un codage 1-parmi-7. Le code CHP de cette ALU est présenté figure 5.18.

```
process bloc_alu
port(Op: in di MR[7], A: in di MR[4],
     B: in di MR[4], Cin: in di MR[2],
     S: out di MR[4],Cout: out di MR[2];)
begin
variable op: MR[7],a: MR[4],b: MR[4],c: MR[2];
*[
    Op?op;
    @[
        op = '0' => A?a, B?b;                          --add
        @[
            a+b<3 => Cout!0, [Cin?c; S!a+b+c];         --K
            a+b=3 => Cin?c; [Cout!c, S!(c=0?3:0)];     --P
            a+b>3 => Cout!1, [Cin?c; S!(a+b+c-4)];     --G
        op = '1' => A?a, B?b;                          --sub
        @[
            b-a<3 => Cout!0, [Cin?c; S!b-a+c];         --K
            b-a=3 => Cin?c;  [Cout!c,S!(c=0?3:0)];     --P
            b-a>3 => Cout!1, [Cin?c; S!(b-a+c-4)];     --G
        op = '2' => A?a, B?b; S!a and b;               --and
        op = '3' => A?a, B?b; S!a or b;                --or
        op = '4' => A?a, B?b; S!a xor b;               --xor
        op = '5' => A?a; S!(not a+1);                  --neg
        op = '6' => A?a; S!(not a);                    --not
        ]
    ]
end
```

FIGURE 5.18 – Code CHP du bloc ALU en base 4

La modélisation complète de ce programme CHP sous forme de MDDs (direct et d'acquittement) étant difficile à illustrer, nous ne représenterons ici, pour des raisons de clarté, que le MDD correspondant à l'évaluation directe de la sortie S (figure 5.19).

La modélisation sous forme de MDD du programme CHP initial est assez simple, nous voyons qu'en fonction des valeurs de l'opérateur Op, une opération est sélectionnée. Par exemple dans le cas où Op vaut 6, le fil 6 suivant le nœud Op devient actif et la sortie S est évaluée suivant la valeur de A en appliquant l'opérateur not. Des optimisations sont ensuite possibles sur le MDD afin d'en diminuer le nombre de nœuds et ainsi accélérer les traitements algorithmiques appliqués par la suite. Ces optimisations dépassent le cadre de cette thèse mais sont partiellement présentées dans [42] et font l'objet de la thèse à

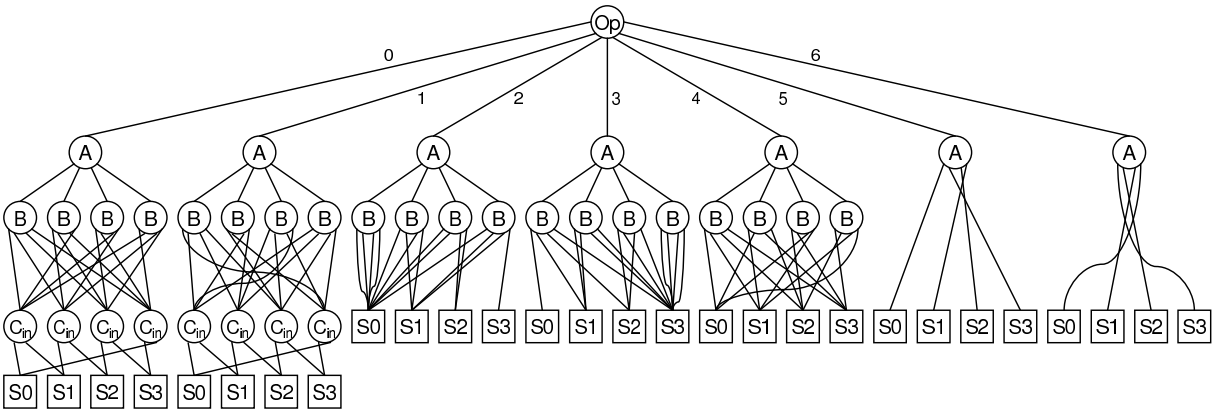


FIGURE 5.19 – MDD d'évaluation directe de la sortie S.

paraître de Vivian Brégier à l'INPG [20].

A partir du MDD présenté ci-dessus, il est possible de synthétiser un circuit décomposé en portes à 2 entrées comme nous l'avons vu précédemment. Le circuit synthétisé (illustré dans [42]) est composé de 107 portes de Muller et de 95 portes OR à deux entrées uniquement.

5.4.1.2 Résultats

Dans cette section, nous cherchons à faire des comparaisons en terme de surface, l'un des objectifs de ce travail étant de réduire l'aire des circuits asynchrones en appliquant nos algorithmes de projection technologique. Dans un premier temps, nous allons comparer une version de cette ALU composée de portes standard synchrones (portes AO222) à une version composée de portes de la bibliothèque TAL. Dans un deuxième temps, nous comparerons cette version composée de portes de la TAL à une version sur laquelle nous avons appliqué les algorithmes de projection. Cette différentiation entre les versions va nous permettre d'évaluer le gain dû à la bibliothèque, du gain associé à la projection technologique.

Le tableau 5.1 présente la différence de surface entre une version simple de l'ALU dans laquelle les portes de Muller sont basées sur des portes standard synchrones AO222, et une version dans laquelle les portes de Muller sont celles développées dans la bibliothèque TAL 130nm. Cette évaluation de la surface s'est effectuée avant la phase de placement routage.

TABLE 5.1 – Différence de surface pour une ALU à base de portes AO222 et une ALU basée sur TAL.

	Bibliothèque TAL	Cellules standard ST
Nb de transistors	1533	2068
Aire (μm^2 (avant placement routage)	2469	3116,36

Ce tableau nous permet de constater que l'utilisation, dans l'ALU, de portes de Muller provenant de la

bibliothèque TAL au lieu de portes AO222 permet de diminuer le nombre de transistors de l'architecture de 35%, tandis que la surface totale diminue de 35%. Ces résultats corroborent ceux que nous avions trouvés dans le chapitre 4.

Si nous comparons les surfaces entre la version ci-dessus basée sur TAL et une ALU sur laquelle nous avons appliqué nos algorithmes de projection technologiques, nous obtenons le tableau 5.2.

TABLE 5.2 – Résultats après projection technologique.

	Netlist TAL	Netlist TAL projetée
Nb de transistors	1533	1034
Aire (μm^2 (avant placement routage)	2469	1401,95

Les portes majoritairement utilisées lors de la phase de projection technologique sont des portes complexes de type Muller-OR. Les portes OR qui n'ont pas pu être regroupées en porte Muller-OR sont regroupées en portes OR à 3 ou 4 entrées. Nous pouvons constater que le nombre de transistors a diminué de 32% et que la surface a diminué de 43% par rapport à la version sans projection technologique. Nous obtenons ainsi un gain global d'environ 50% en terme de nombre de transistors mais également de surface par rapport à la version initiale basée sur des portes AO222.

Cependant bien que ces résultats soient intéressants dans la comparaison entre versions asynchrones, il est nécessaire de comparer cette même architecture avec une version synchrone. Pour cela, nous avons réalisé une description complète de l'architecture en VHDL, puis nous avons utilisé Design Compiler de Synopsys pour réaliser la synthèse de l'architecture. Bien entendu, il a été nécessaire de rajouter une horloge dans la description VHDL puisque dans le circuit synchrone la mémorisation se fait par l'intermédiaire de registres pilotés par le signal d'horloge. Le tableau 5.3 présente la différence de surface entre la version asynchrone et la version synchrone.

TABLE 5.3 – Comparaison avec une version synchrone.

	Netlist TAL projetée	Netlist synchrone
Nb de transistors	1034	386
Aire (μm^2 (avant placement routage)	1401,95	476,06

Si nous comparons les deux versions, la version synchrone est très avantagée du point de vue de la surface. En effet, l'ALU synchrone comporte 2,7 fois moins de transistors pour une surface 2,9 fois plus petite. Bien que cette différence de surface soit grande, il est important de comprendre que dans un

circuit de grande taille cette différence serait moindre. En effet, nous devons prendre en considération le fait que dans un circuit synchrone de taille importante, l'arbre d'horloge prend une part non négligeable de la surface totale du circuit. Dans cet exemple, l'horloge occupe une place quasi nulle contrairement au contrôle local dans le cas asynchrone. Le but dans cet exemple n'était pas de se ramener à une surface identique (ce qui est probablement impossible) entre les deux versions mais de diminuer autant que possible l'écart en appliquant nos algorithmes de projection technologique. Dans cette optique, nous passons d'un rapport de surface de 6,5 à 2,9, ce qui est un résultat correct. Il serait encore possible de diminuer la surface en ajoutant des cellules à la bibliothèque, comme illustré sur un DES dans le chapitre 6.

5.4.2 Analyse d'une unité logique

Une seconde étude a été menée sur l'unité logique d'un MIPS 4k 32 bits, développée par David Rios au sein du groupe CIS.

Cette unité logique exécute les instructions suivantes : OR, AND, NOR, NAND. La synthèse de cette architecture a été réalisée avec TAST Synthetizer. Le circuit obtenu, basé sur une bibliothèque de cellules génériques appelée AGLib pour Asynchronous Generic Library, comporte initialement 1541 portes logiques pour une surface de 22805 μm^2. Les algorithmes de projection technologique nous ont permis de diminuer cette surface de 34% (15003 μm^2), la netlist finale ne comportant plus que 544 portes logiques.

5.5 Conclusion

Nous avons présenté dans ce chapitre les algorithmes de projection technologique pour les circuits asynchrones QDI mis au point durant cette thèse.

Dans un premier temps, nous avons présenté les diagrammes de décision multi-valués. C'est à partir de ces diagrammes, généralisation des diagrammes de décision binaires utilisés en synchrone, que nous pouvons obtenir une décomposition de nos circuits qui respecte les critères d'insensibilité aux délais.

Dans un deuxième temps, nous avons présenté les algorithmes développés pour chaque phase de la projection technologique :

– La phase de décomposition est basée sur les diagrammes de décision multi-valués. L'application d'un ensemble de règles de synthèse pour chaque nœud du MDD représentant le circuit, nous permet d'obtenir une netlist de portes à deux entrées qui respecte les critères d'insensibilité aux délais par construction.

- La phase de partitionnement consistant à décomposer le circuit initial en sous-circuits de tailles inférieures permet de réduire la complexité des algorithmes de couverture. La solution que nous avons choisie pour partitionner le circuit consiste à former des cônes logiques à partir des sorties primaires du circuit. Cette solution, en plus de fournir de bons résultats, offre le grand avantage d'être facile à implanter.
- La phase de couverture présentée ici est adaptée des travaux de M. Zhao et S. Sapatnekar. La méthode proposée est efficace en termes de vitesse de traitement et d'occupation mémoire. L'idée est de créer une relation entre la couverture d'un nœud et la couverture de ses fils dans le réseau représentant le circuit, afin d'accélérer cette étape souvent coûteuse en temps de calcul.

Finalement une validation des algorithmes est proposée au travers de deux circuits test : une unité arithmétique et logique en base 4 et une unité logique d'un MIPS 4k 32 bits. D'autres circuits ont pu être optimisés en utilisant ces algorithmes, comme nous le verrons dans le chapitre suivant.

Partie III

APPLICATION AUX SYSTÈMES SÉCURISÉS

Chapitre 6

Insertion dans un flot industriel de produits sécurisés : conception d'un cryptoprocesseur DES

6.1 Introduction

Cette thèse s'inscrit dans un projet de la région PACA, consistant à étudier différentes méthodes pour concevoir des produits sécurisés. L'objectif de ce travail est de protéger les circuits intégrés, par exemple les circuits de type carte à puce, contre les attaques non intrusives de type SPA, DPA, DEMA ou encore DFA. Pour cela, nous exploitons les propriétés de la logique asynchrone pour augmenter la sécurité des circuits vis-à-vis de ce type d'attaques (SPA, DPA tout d'abord). L'idée maîtresse étant de maîtriser la consommation électrique pour qu'elle soit non corrélée aux valeurs des données traitées.

La première étape de ce travail fut la mise au point de la bibliothèque TAL, composée de cellules spécialement dessinées pour les circuits asynchrones. La seconde étape fut de développer les algorithmes de projection technologique permettant de concevoir des circuits asynchrones exploitant cette bibliothèque. Ces deux premières étapes ont été présentées dans les chapitres précédents.

Finalement, la dernière étape consiste à évaluer les méthodes mises au point afin de les valider, mais également de les intégrer dans un flot de conception standard, et de montrer les gains apportés par les circuits asynchrones. Nous présenterons donc dans un premier temps les travaux réalisés pour valider notre bibliothèque ainsi que nos méthodes de synthèse et leur intégration dans un flot de conception standard ; et dans un deuxième temps les travaux effectués pour évaluer la sécurité de nos circuits.

6.2 Un circuit de cryptage symétrique : le DES

Afin de valider notre bibliothèque et nos algorithmes de projection, nous avons décidé de les appliquer à la conception d'un circuit cryptographique de type DES. Un circuit DES, pour Data Encryption

Standard, est un circuit cryptographique symétrique à clé secrète, c'est-à-dire que la même clé est utilisée pour le chiffrement et le déchiffrement d'un message. Développé dans les années 70 par IBM, il a été adopté en 1977 par le gouvernement américain comme standard de protection d'informations non secrètes mais confidentielles [70].

L'algorithme DES transforme un bloc de 64 bits en un autre bloc de 64 bits. Il manipule des clés individuelles de 56 bits, représentées par 64 bits (avec un bit de chaque octet servant pour le contrôle de parité). Malgré son obsolescence et sa faiblesse face aux machines de calcul modernes en matière de sécurité, le DES reste très utilisé par l'industrie, notamment pour son implantation matérielle qui reste avantageuse par sa faible surface, et son utilisation en triple cryptage consécutifs (triple DES), qui reste encore sûre pour de nombreuses applications. De plus, les attaques sur le DES sont connues, et bien qu'efficaces, demandent du matériel de mesure de précision.

L'algorithme est basé sur le principe des schémas de Feistel [81] (figure 6.1). D'une manière générale, on peut dire que le circuit DES fonctionne en trois étapes :

- Permutation initiale selon une table d'un bloc de 64 bits.
- Le résultat est soumis à 16 itérations (ou rondes, d'indice i) pendant lesquelles les opérations suivantes sont réalisées :
  - Soient L_i, R_i les deux moitiés (gauche et droite) de 32 bits chacune, du bloc 64 bits à chiffrer.
  - Soit K_i la sous-clé spécifique à chaque ronde, calculée selon une table de transformation.
  - Soit $F(K_i, R_i)$ une fonction de transformation (figure 6.2).
  - A chaque ronde, on calcule :
    $L_{i+1} = R_i$
    $R_{i+1} = L_i \oplus F(K_i, R_i)$
- Le dernier résultat de la dernière ronde est transformé par la fonction inverse de la permutation initiale.

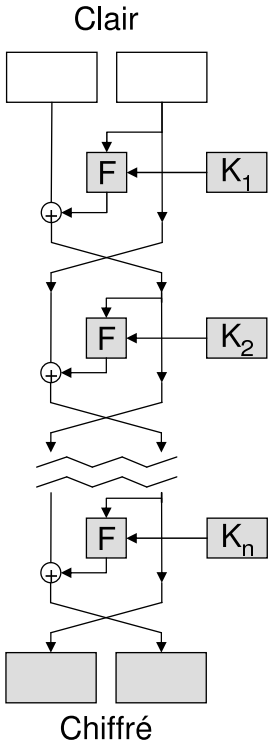


FIGURE 6.1 – Architecture schéma de Feistel.

Le DES utilise huit tables de substitution, les S-Boxes, qui contribuent à la « confusion » en rendant l'information chiffrée inintelligible. Les S-Boxes permettent de casser la linéarité de la structure de chiffrement.

Chaque S-Boxe prend une variable de 6 bits en entrée et produit une sortie de 4 bits. Les valeurs présentes dans les S-Boxes ont été choisies de manière à résister aux attaques par divers moyens comme l'utilisation de fonctions courbes. Dans le cas du DES, il a été prouvé que les tables avaient été conçues de manière à résister à la cryptanalyse différentielle (technique qui ne sera publiée que bien des années plus tard) [12].

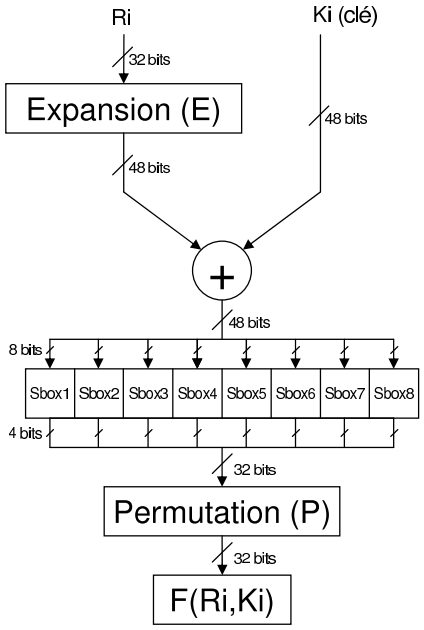


FIGURE 6.2 – Fonction F du DES.

6.3 Évaluation de la bibliothèque TAL

Afin de valider notre bibliothèque, nous avons réalisé au sein du groupe CIS un ensemble de circuits utilisant la bibliothèque TAL. La création de ces circuits nous a permis de valider l'insertion de TAL dans les outils commerciaux de placement routage, de simulation, ..., mais également de valider le fonctionnement des cellules. Afin d'évaluer les performances de notre bibliothèque, une première version d'un DES a été réalisée à partir de cellules standard synchrones, dans laquelle les portes de Muller sont implantées à partir de portes AO222. Une seconde version a été réalisée à partir des cellules de la TAL (figure 6.3). Il est nécessaire de souligner plusieurs points :

– Les netlists de ces circuits ont été réalisées à partir d'une version ultérieure de TAST et optimisées à la main, seules les phases de placement routage et de simulation ont été réalisées avec des outils commerciaux. Ces étapes nous ont permis de valider la caractérisation de notre bibliothèque puisque des optimisations et des vérifications de violation de timing ont pu être réalisées dans l'outil de placement routage SoC Encounter. Dans cet exemple, nous avons pu constater que l'intégration de TAL dans un flot de conception standard est totale.

– La projection technologique des deux circuits DES a été faite à la main. Il est probable que des optimisations plus importantes auraient pu être obtenues en utilisant notre outil de projection technologique. Par exemple, le panel des cellules utilisées aurait été plus important, et des optimisations en surface auraient pu être réalisées. Malheureusement, faute de temps, les circuits ont dû être envoyés en fabrication avant la fin du développement de l'outil de projection technologique.
– Aucune des deux architectures des circuits DES n'a été optimisée ; la comparaison avec une version synchrone d'un circuit DES serait au désavantage de l'asynchrone. Cependant nous ne cherchons pas ici à évaluer les performances intrinsèques des circuits asynchrones, mais à quantifier les gains apportés par notre bibliothèque.
– Seul le DES utilisant la bibliothèque TAL a été envoyé en fabrication. Les comparaisons qui suivent sont donc basées sur les résultats de simulations électriques réalisées sur les netlists des circuits.

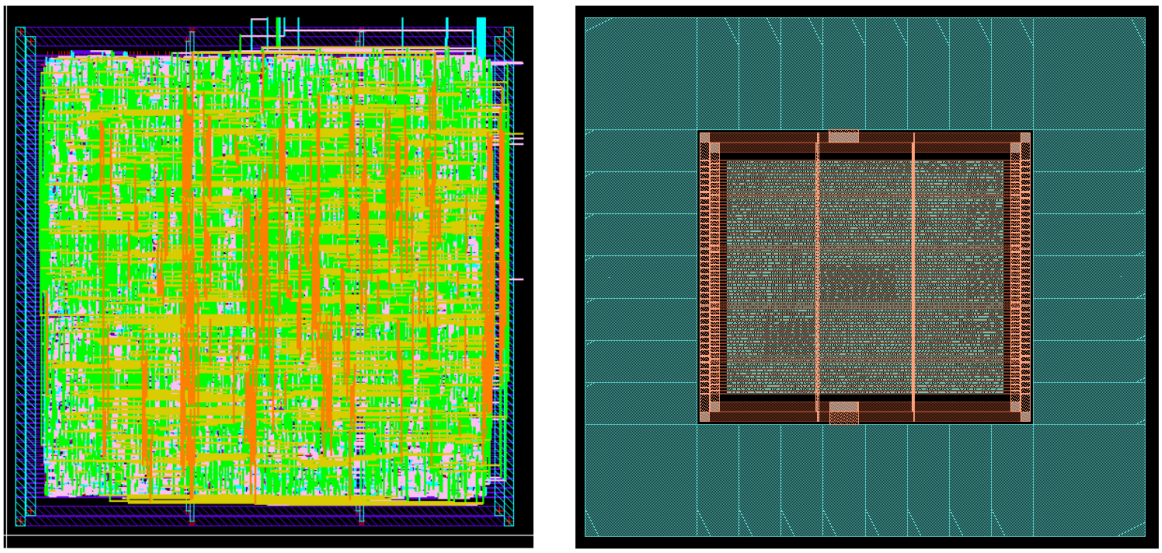


FIGURE 6.3 – Cœur du DES à base de portes AO222 (à gauche), DES TAL complet (à droite).

TABLE 6.1 – Tableau de comparaison des circuits DES en surface.

	DES utilisant les cellules AO222	DES TAL
Surface core (mm^2)	0.218	0,156
Nombre de portes	13821	8434
Nombre de portes équivalentes (porte ND2LL = 6,0516 μm^2)	30795	19404

Le tableau 6.1 fournit les différences en nombre de portes et en surface de ces deux versions du circuit DES. Le gain en surface apporté par l'utilisation de la bibliothèque TAL est important puisque la surface du cœur du DES TAL est 29% plus petite que celle de la version utilisant les portes AO222.

Quant au nombre de portes équivalentes, il est 39% moins important dans la version TAL que dans la version à base de portes AO222 du DES.

Sur le circuit global, la réduction de surface ne correspond pas tout à fait à la réduction moyenne obtenue sur la bibliothèque TAL qui était de 50%. Cela s'explique par le fait qu'en réalité une grande partie des portes utilisées dans le circuit sont des portes de Muller à 2 entrées avec ou sans reset (tableau 6.2) pour lesquelles le gain en surface est moins important ; mais également que le pourcentage de cellules de TAL dans le DES s'élève à environ 38% du nombre total de cellules.

Afin de préciser le type de portes de la TAL et leur quantité dans le DES TAL, le tableau 6.2 fournit le pourcentage en nombre de portes, des principales fonctionnalités de la TAL utilisées dans le DES.

TABLE 6.2 – Pourcentage d'occupation des cellules TAL.

Type de cellules	Pourcentage
Buffer	2,69%
M2	2,33%
M3	2,48%
M4	2,66%
M2RB	10,81%
M3RB	5,92%
MO333RB	7,59%
MO22	1,52%
MO33RB	1,33%

Le reste du DES est composé de portes provenant de la bibliothèque standard de ST dont des portes logiques simples (NOR, NAND, MUX, ...), mais également de bascules pour l'interfaçage du DES avec un environnement synchrone ; et de cellules de remplissage utilisées par l'outil de placement routage.

Le tableau 6.3 fournit les valeurs obtenues en vitesse de fonctionnement et en consommation pour les deux versions du DES. Ces valeurs ont été obtenues avant et après placement routage du circuit, en utilisant une tension d'entrée de 1,2V, à partir de l'outil de simulation électrique Nanosim.

TABLE 6.3 – Tableau de comparaison des circuits DES en vitesse et consommation.

	Avant Placement routage		Après Placement routage	
	Consommation (mA)	Temps (ns)	Consommation (mA)	Temps (ns)
DES cellules AO222	17,5	54,8	19,7	53,8
DES TAL	9	58	12	44

La consommation moyenne du DES TAL sur le temps de chiffrement après placement routage représente 61% de la consommation du DES basé sur les portes AO222. Cette réduction de la consommation très importante est due à la diminution du nombre de nœuds internes dans le DES par l'utilisation de portes complexes provenant de la TAL, et à l'utilisation de cellules qui consomment moins en moyenne que les cellules AO222.

Avant placement routage, les temps de fonctionnement des deux DES sont presque similaires. Après placement routage, le temps de fonctionnement du DES TAL diminue pour atteindre 82% du temps de fonctionnement du DES à base de portes AO222. La forte diminution du nombre de nœuds internes dans le circuit du fait de l'utilisation de portes complexes, permet d'augmenter la vitesse du DES TAL par rapport au DES implanté à base de portes AO222. De plus, la variation des valeurs de temps obtenues avant et après placement routage nous permet de conclure que la phase de placement routage, et notamment les phases d'optimisation de timing, se déroulent correctement pour le circuit DES utilisant la bibliothèque TAL. Une fois de plus, nous pouvons constater l'intégration complète de TAL dans un flot de conception standard, utilisant des outils du commerce.

Toutefois il est intéressant de se demander comment une optimisation en délai peut être réalisée sur un circuit asynchrone. En effet, l'optimisation en délai se fait dans un circuit synchrone entre deux registres pilotés par le signal d'horloge. Les algorithmes d'optimisation considèrent tous les chemins entre deux registres, vérifient que les temps de traversée de ces chemins sont plus courts que la période de l'horloge, et dans le cas contraire ajoutent des buffers ou modifient la sortance des portes pour diminuer ce temps de traversée. Dans un circuit asynchrone, il n'y a pas de registre synchrone, et encore moins de signal d'horloge. Le schéma de la figure 6.4 résume le fonctionnement d'un circuit asynchrone QDI. Il représente l'implantation de deux modules QDI avec les éléments de mémorisation appelés Half-Buffer. Le Half-Buffer permet d'implanter le protocole de communication 4 phases décrit dans le chapitre 1 (cf. §1.3.2.1). La mémorisation des données se fait ainsi localement à l'intérieur des Half-Buffers et le signal d'acquittement peut être considéré comme une sorte d'horloge locale. L'utilisation de Muller comportant un signal de reset, permet de réinitialiser les éléments mémoires du circuit lors du démarrage de celui-ci. En modifiant les fichiers de description des cellules générés lors de la caractérisation de la bibliothèque, il est possible de faire passer le signal de reset comme étant un signal d'horloge global. Cette modification va permettre à l'outil d'optimisation de considérer les chemins entre deux Half-Buffers, et donc d'optimiser les temps de traversée de ces chemins. Le comportement des algorithmes sera le même pour les circuits asynchrones que pour les circuits synchrones.

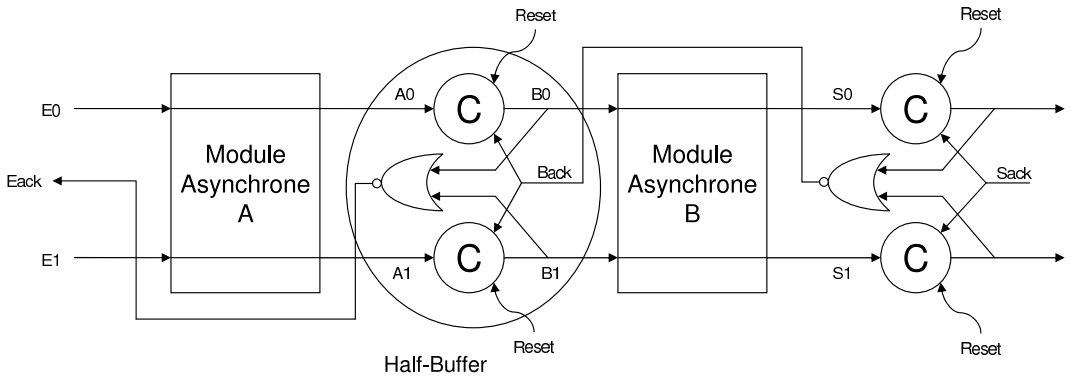


FIGURE 6.4 – Half-Buffer à un bit implantant un protocole 4 phases entre deux blocs QDI.

6.4 Intégration d'un circuit DES asynchrone dans un produit Smartcard 8 bits

6.4.1 Présentation du projet

L'objectif de ce projet était de remplacer dans un produit smartcard 8 bits existant et développé par ST Microelectronics, un module de cryptage DES synchrone par un composant de même fonctionnalité asynchrone. L'architecture et le circuit de cryptage asynchrone ont été entièrement développés au laboratoire TIMA en respectant un cahier des charges provenant de ST Microelectronics. L'idée est de remplacer le composant synchrone existant sans modification sur l'utilisation du produit final afin de valider l'intégration complète de la bibliothèque, mais également de valider les algorithmes de projection technologique. Le circuit sera ensuite finalisé, testé et validé par STMicroelectronics, en utilisant des outils de conception du commerce et le flot de conception de STMicroelectronics.

La figure 6.5 illustre une vue générale du circuit, dans lequel devra s'insérer le DES asynchrone. L'impératif de ce projet est la transparence d'utilisation du produit final quelle que soit l'architecture synchrone ou asynchrone choisie.

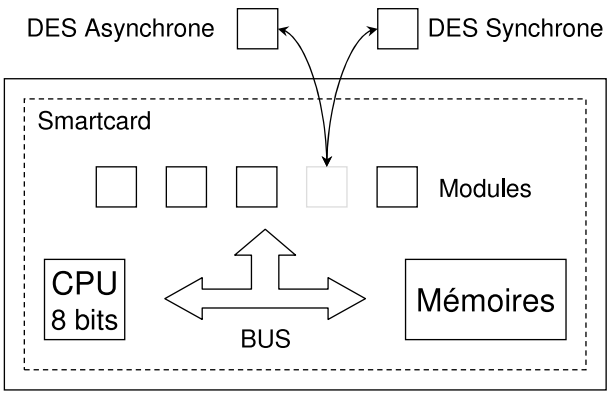


FIGURE 6.5 – Schéma général du projet.

La collaboration, entre les trois entités TIMA, LIRMM et STMicroelectronics, s'est déroulée comme suit :

- le laboratoire TIMA fournit le cœur du DES asynchrone,
- le DES est basé sur la bibliothèque TAL développée conjointement par TIMA et le LIRMM,
- STMicroelectronics est chargé d'implanter le DES asynchrone dans un produit et de valider son utilisation dans un produit Smartcard existant. Le circuit Smartcard choisi est un STM7 de STMicroelectronics.

Le cœur du DES se nomme ADES pour « Asynchronous DES », et il est relié au circuit Smartcard STM7 par une interface synchrone/asynchrone (figure 6.6).

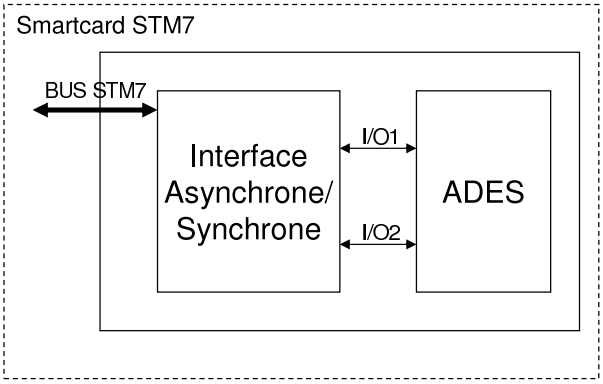


FIGURE 6.6 – Description haut niveau du circuit.

Les signaux I/O1 et I/O2 sont composés :

- de signaux de début et de fin de calcul du DES,
- de signaux de contrôle tels que cryptage/décryptage ou remise à zéro,
- la valeur de la donnée codée en double-rail,
- la valeur de la clé de cryptage encodée en double-rail, et la sortie du cœur du DES.

6.4.2 Description de l'architecture

La conception du DES a été réalisé par Fraidy Bouesse à partir de netlists de DES asynchrones existantes [15]. De nombreuses optimisations ont été réalisées afin de diminuer la consommation mais surtout la surface du DES. Dans ce chapitre nous expliquerons les concepts qui nous ont permis de diminuer la surface et la consommation de notre circuit asynchrone afin de correspondre aux attentes de STMicroelectronics.

6.4.2.1 Diminution de la surface

Au niveau de la surface, l'utilisation de l'outil de projection technologique nous a permis d'optimiser l'utilisation des cellules de la TAL, en utilisant la surface comme critère de choix des cellules pour la couverture. Cependant, il a été nécessaire de modifier la structure même de l'architecture du DES pour arriver à en diminuer fortement la surface. Pour cela, le principe est d'utiliser des portes combinatoires (comme des portes ET/OU suivant le besoin) à la place des portes de Muller dans les parties pour lesquelles les contraintes de délais ne peuvent pas être violées. Par exemple, on retrouve dans le DES en sortie de l'interface asynchrone/synchrone un multiplexeur présenté figure 6.7a.

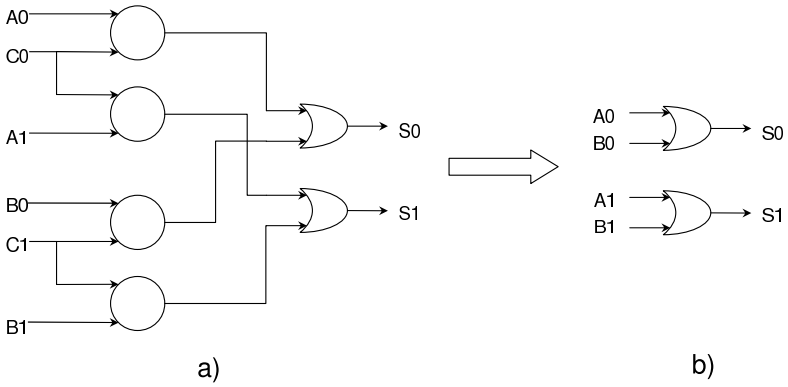


FIGURE 6.7 – Modification d'un MUX dont le fonctionnement est garanti correct par construction.

Dans ce multiplexeur, les signaux de commande C0 et C1 (avec $C0 \oplus C1 = 1$) sont couplés aux entrées A et B en double-rail par des portes de Muller pour synchroniser l'arrivée et la propagation des signaux. Cependant, il est possible de garantir par construction dans le DES que les entrées A0, A1 (et B0, B1) sont exclusives dans tous les chemins d'exécution du DES. Il devient alors possible de modifier ce multiplexeur comportant initialement des portes de Muller en une structure uniquement combinatoire (figure 6.7b). Le gain en surface apporté par cette simple modification est très important, toutefois il est nécessaire de connaître et de maîtriser parfaitement l'architecture du DES pour pouvoir le réaliser.

6.4.2.2 Diminution de la consommation

Pour la consommation, l'idée principale est de diminuer les pics de consommation du DES et de contrôler la consommation moyenne en supprimant le plus possible la concurrence des blocs de calcul. Dans [17], l'auteur nous présente différentes architectures utilisant différentes méthodes d'acquittement à l'intérieur des circuits asynchrones. Ces différentes méthodes vont permettre de jouer sur la concurrence entre les blocs comme l'illustre la figure 6.8.

Nous voyons dans la figure 6.8a que l'arrivée du signal Back va lancer la commutation des deux portes de Muller en même temps. Dans la figure 6.8b, l'exécution de chaque Muller va dépendre de

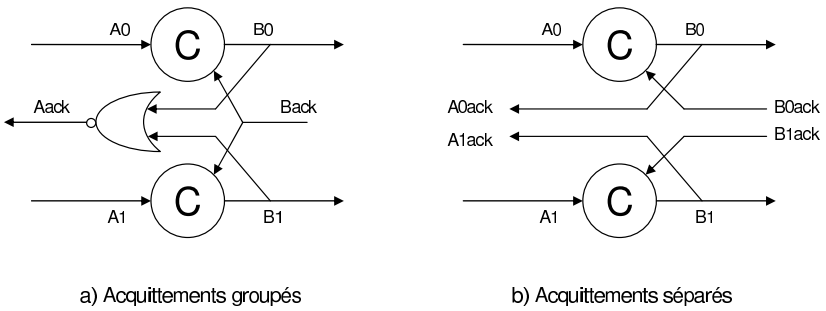


FIGURE 6.8 – Différence entre acquittements groupés ou séparés.

l'arrivée des signaux d'acquittements associés. Du fait des différences entre les deux fils B0ack et B1ack au niveau du routage, l'exécution des deux Muller va être répartie dans le temps. Le pic de consommation sera donc plus faible pour l'exécution de ce bloc.

Ainsi dans le cas où tous les signaux d'acquittement sont groupés, c'est-à-dire que chaque module qui compose le circuit ne possède qu'un seul signal d'acquittement, la consommation en courant est importante. En effet, les signaux d'acquittements qui permettent l'activation des calculs d'un module (modules de complexité plus ou moins importante), peuvent être vus comme des signaux d'horloge locaux. Cette logique offre la possibilité de lancer l'exécution de plusieurs modules de manière concurrente, la consommation électrique peut potentiellement augmenter rapidement. La figure 6.9 présente l'architecture du DES initiale, composée d'acquittements groupés entre les modules.

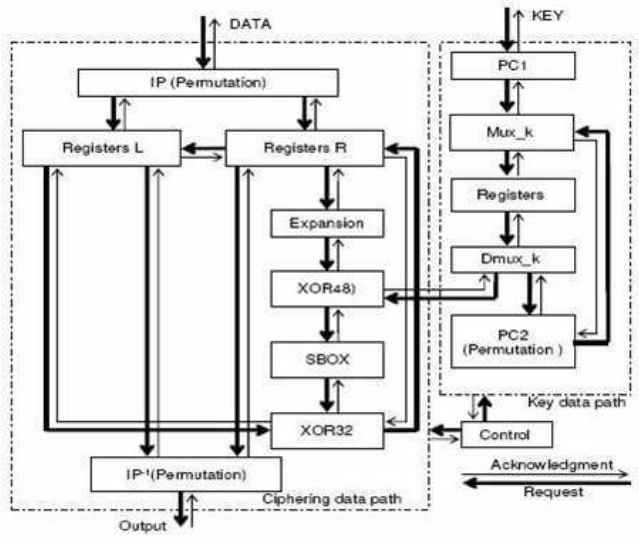


FIGURE 6.9 – DES avec acquittements groupés.

Par contre, dans le cas de signaux d'acquittement séparés, c'est-à-dire que le nombre de signaux d'acquittements d'un module dépend du nombre de bits d'entrée qui peuvent être acquittés par le module, la dépendance entre les signaux d'acquittements et les modules est réduite. Chaque signal d'acquittement

va directement dépendre des chemins de données du module. Cette approche permet de diminuer la consommation moyenne en courant, mais également de la rendre plus lisse. Les circuits obtenus par cette méthode sont également plus rapides qu'avec la méthode précédente. La figure 6.10 présente le schéma détaillé du DES réalisé avec des acquittements séparés.

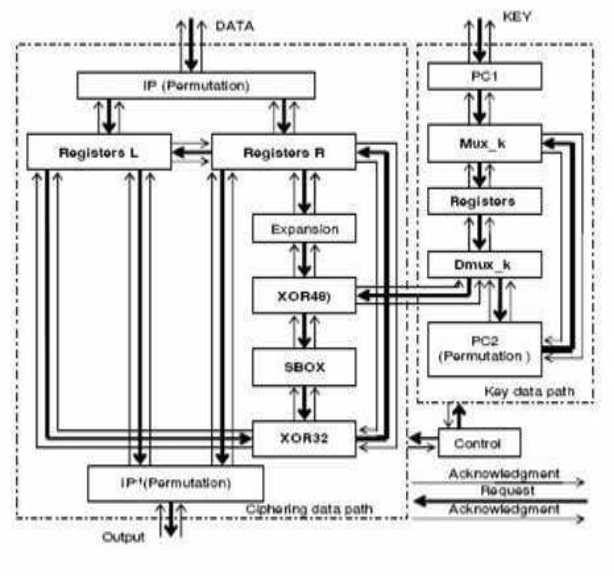


FIGURE 6.10 – DES avec acquittements séparés.

Finalement, pour diminuer encore la consommation, une analyse de la concurrence des blocs est réalisée. Cette analyse, basée sur les travaux de Yannick Monnet présentés dans [69], va permettre de connaître les blocs critiques du DES et d'identifier les blocs consommant beaucoup au même moment. De cette analyse, une machine à états est créée pour contrôler le lancement en séquence des différents blocs critiques de calcul. Par exemple, dans le DES, au lieu de lancer le calcul sur les 8 S-Boxes en parallèle, les S-Boxes vont calculer séquentiellement.

6.4.3 Bilan

Au final, le DES asynchrone développé ne comporte plus que 8986 portes équivalentes (porte équivalente NAND2 en technologie 130nm) pour une surface de 54364 μm^2. Le DES synchrone complet comporte 5286 portes équivalentes pour une surface de 31980 μm^2. Le rapport de surface entre ces deux versions du DES est de 1,74. Cette augmentation de surface est faible par rapport aux versions présentées précédemment, prouvant que des architectures asynchrones QDI optimisées peuvent être viables en terme de surface. Nous verrons dans le paragraphe suivant que d'autres optimisations peuvent encore être effectuées afin de diminuer davantage la surface et la consommation de nos circuits.

Le temps de calcul du DES asynchrone est indépendant de la fréquence d'horloge de la partie syn-

chrone. Cependant il est nécessaire de prendre en compte la phase de lecture/écriture dans les registres contenus dans les interfaces pour évaluer le temps de fonctionnement complet du DES. Cette phase de lecture/écriture est, quant à elle, dépendante de la fréquence d'horloge. Ainsi, pour une fréquence de fonctionnement de la partie synchrone de 10MHz, le DES asynchrone a un temps de calcul simple DES de 35,5 μs, et un temps de calcul triple DES de 54,5 μs. A une fréquence de fonctionnement de 20MHz de la partie synchrone, le temps de calcul simple DES asynchrone est de 19,9 μs, et le temps triple DES de 33,7 μs. Ainsi à basses fréquences, notre DES apporte un gain en vitesse par rapport au DES synchrone, tandis qu'à fréquence d'utilisation moyenne, les performances sont équivalentes.

Au niveau de la consommation moyenne, le DES asynchrone a une consommation 20% plus faible (0,19mA) pour une utilisation à 10MHz. Pour des performances équivalentes à 10MHz (le DES asynchrone étant plus rapide dans ce cas là), le gain en consommation est de l'ordre de 39%. Pour la partie numérique complète du produit smartcard, le gain en consommation total apporté par notre DES asynchrone est de l'ordre de 25%.

Au niveau des pics de consommation, les gains du DES asynchrones sont importants, la diminution du pic de consommation au niveau du produit complet étant de l'ordre de 60%. Au niveau du bloc DES, la diminution du pic de courant est de l'ordre de 69%.

6.5 Optimisations de l'architecture du DES

Suite à ces travaux, nous nous sommes rendus compte qu'il était encore possible de diminuer la surface et la consommation de notre DES asynchrone. Ne pouvant utiliser le DES synchrone de STMicroelectronics comme point de comparaison, nous avons synthétisé notre propre circuit DES synchrone. Ce dernier est composé de 4360 portes équivalentes, l'énergie qu'il consomme pour le calcul du simple DES est de 330 pJ à une vitesse est de 204 ns. Nous utiliserons dans la suite ce DES synchrone comme point de comparaison.

Afin de diminuer la surface de notre DES, nous avons modifié le codage utilisé dans les S-Boxes. Nous avons utilisé un codage 4-rail à la place du codage double-rail utilisé précédemment. De plus, une grande partie de la surface du DES asynchrone présenté ci-dessus, est occupée par des cellules Half-Buffers. Nous avons donc enrichi la bibliothèque TAL de cellules Half-Buffer optimisées en surface et en consommation.

Ces deux optimisations nous ont permis d'obtenir un DES asynchrone comportant 5489 portes équivalentes (porte équivalente NAND2 en 130 nm). L'énergie consommée par ce DES est de 410 pJ pour une vitesse de 200 ns. Au niveau de la surface, le rapport entre le DES synchrone et le DES asynchrone

est de 1,26. Au niveau de l'énergie consommée, le rapport entre le DES synchrone et le DES asynchrone est de 1,24. Cette comparaison basée sur des circuits de type DES est peu favorable aux circuits asynchrones puisque l'arbre d'horloge est quasi inexistant dans le circuit synchrone et tous les modules du DES sont actifs pendant le calcul. Cependant les résultats obtenus sont très encourageants, le surcoût en surface est plus que raisonnable, tandis que la consommation d'énergie est légèrement plus élevée pour le composant seul, sans tenir compte de la diffusion de l'horloge à l'échelle du système, de la génération de cette horloge, ni du coût des communications. Les avantages des circuits asynchrones, faible émission électromagnétique, pics de consommation beaucoup plus faibles en amplitude et une robustesse intrinsèque aux attaques par analyse en courant, apparaissent tout à fait intéressants.

6.6 Conclusion

Nous avons présenté dans ce chapitre, différentes réalisations qui nous ont permis de valider plusieurs aspects de cette thèse. D'une part, nous avons pu valider l'intégration de la bibliothèque TAL dans un flot de conception industriel utilisant des outils de « back-end » commerciaux. Au sein du laboratoire TIMA, la comparaison entre un DES basé sur des portes AO222 et un DES basé sur la TAL nous a permis de démontrer le bon fonctionnement de la bibliothèque, mais également de montrer les avantages de son utilisation : le circuit DES basé sur TAL est plus rapide (18%), il consomme moins (-39%) et sa surface est plus petite que son équivalent en portes AO222 (-29%).

D'autre part l'intégration d'un DES – spécialement conçu à partir d'un cahier des charges défini par un industriel (STMicroelectronics) – dans un produit existant, nous a permis une nouvelle fois de confirmer l'utilisation de la TAL dans des outils de conception industriels ; mais également de valider notre outil de projection technologique. En effet, les améliorations obtenues en surface et en consommation pour ce DES asynchrone sont dues à une optimisation de son architecture, mais aussi à l'utilisation de notre outil de projection pour diminuer la surface et le nombre de portes. Les résultats obtenus pour ce coup d'essai sont prometteurs pour la technologie asynchrone.

Les dernières optimisations présentées nous ont permis de montrer qu'un circuit asynchrone peut rivaliser avec un circuit synchrone dans le cas d'un cryptoprocesseur de type DES, même si ce type de circuit est peu favorable aux circuits asynchrones. Il apparaît évident que dans un circuit comportant un arbre d'horloge complexe (voire plusieurs arbres d'horloge), les circuits asynchrones seraient beaucoup plus avantageux en terme de consommation notamment.

La bibliothèque TAL a également été utilisée par le CEA-Leti dans la conception d'une partie d'un réseau sur puce (« Network-on-chip » ou « NoC ») de type GALS (« Globalement Asynchrone Locale-

ment Synchrone ») [5]. L'utilisation de la TAL a également permis d'obtenir de très bons résultats sur ce type d'architecture. Une nouvelle version de ce NoC basée sur TAL2 est en cours de développement au sein du CEA-Leti.

Le dernier aspect de ce travail de thèse est d'utiliser la technologie asynchrone pour augmenter la sécurité de produits sécurisés. Des travaux sur ce thème ont été menés au sein du laboratoire TIMA et ont aboutis à deux brillantes thèses. La première, de Fraidy Bouesse [15], présente les avantages de la technologie asynchrone pour la conception de circuits asynchrones sécurisés contre les attaques de type DPA, SPA. La technologie asynchrone est intrinsèquement résistante à ce type d'attaques du fait de l'utilisation d'un codage multi-rail. Cependant des attaques sont toujours possibles et différentes contre-mesures sont présentées pour renforcer efficacement la sécurité des circuits asynchrones. La seconde, de Yannick Monnet [69], présente une étude sur la résistance des circuits asynchrones aux attaques par injection de fautes. Les solutions présentées pour renforcer les circuits asynchrones permettent d'obtenir d'excellents résultats, prouvant la grande résistance des circuits asynchrones à ce type d'attaques.

Cependant, il serait intéressant de connaître dès la phase de conception, la résistance d'un circuit aux différents types d'attaques. L'objectif serait de pouvoir orienter les étapes de la synthèse, et notamment la phase de projection technologique, pour augmenter la résistance de certaines parties d'un circuit prouvées sensibles par analyse. Le chapitre suivant aborde ce problème.

Chapitre 7

Analyse en sécurité des circuits asynchrones QDI

7.1 Introduction

Longtemps resté un domaine réservé aux services secrets des gouvernements et des services d'espionnage (contre-espionnage) des militaires, le domaine de la cryptanalyse est devenu depuis les années 70, un domaine de recherche grand public porté par de nombreux laboratoires académiques.

En effet, le développement considérable des réseaux de télécommunication a favorisé le développement des transactions financières plus seulement sur des réseaux fermés ou privés mais sur un réseau mondial accessible par tous. Ces nouveaux types de communications nécessitent de plus en plus de moyens de transactions dits sûrs, pouvant résister à diverses attaques cryptanalytiques. De cette lutte effrénée entre les cryptographes, qui mettent en place des systèmes ou algorithmes cryptographiques et les cryptanalystes, qui développent des techniques pour briser les systèmes de sécurité, un net avantage est donné au second depuis le développement de la cryptanalyse matérielle. Ces nouvelles méthodes de cryptanalyse dites attaques par canaux cachés ont montré leur efficacité sur de nombreux produits commerciaux de type carte à puce.

Ce chapitre est consacré à la présentation d'une méthode d'analyse de la résistance de circuits asynchrones à un certain type d'attaques par canaux cachés, la DPA (« Differential Power Analysis »). Nous allons dans la première partie introduire quelques notions de base de la cryptologie avant de présenter plus précisément les attaques de type DPA. Puis nous présenterons l'outil d'analyse que nous avons développé durant cette thèse.

7.2 La cryptologie : notion de base et terminologie

Étymologiquement composée des deux mots grecs crypto' (caché) et logie' (science), la cryptologie est une branche des mathématiques englobant aussi bien l'ensemble des procédés cryptographiques que

celui des méthodes cryptanalytiques

7.2.1 La cryptographie

La cryptographie est un art et une science consacrée à la protection des données en les rendant incompréhensibles sauf pour son destinataire. Elle a pour objectif la conception et l'analyse des mécanismes permettant d'assurer l'intégrité, l'authenticité, la confidentialité et le non désaveu des données et des communications. Pour cela elle utilise des processus cryptographiques.

Un processus cryptographique est une transformation d'un message appelé texte en clair en un message incompréhensible appelé texte chiffré. Ce processus est nommé chiffrement (cryptage ou « encryption ») et lorsque le processus inverse est réalisé, on parle alors de déchiffrement (décryptage).

$$\left.\begin{array}{ll} Chiffrement & E(M) = C \\ Déchiffrement & D(C) = M \end{array}\right\} D(E(M)) = M$$

E est la fonction de chiffrement,

D la fonction de déchiffrement,

C le texte chiffré et M le texte en clair.

Les fonctions de chiffrement ou de déchiffrement représentent des fonctions mathématiques encore appelées algorithmes cryptographiques (cryptosystèmes). La sécurité d'un cryptosystème dépend de deux paramètres : la sûreté de l'algorithme et la longueur de la clé utilisée. On entend par sûreté le fait qu'il n'existe pas de meilleur moyen de casser l'algorithme que d'utiliser une attaque exhaustive (essai de toutes les clés possibles).

Le concept de clé cryptographique a été créé pour augmenter la robustesse des algorithmes cryptographiques. Elle est généralement mixée avec les données d'entrées des algorithmes cryptographiques par l'opérateur logique OU EXCLUSIF. Plus la longueur de la clé est importante, plus il est difficile de la briser par une attaque exhaustive. Pour une clé de 56 bits, il faut en moyenne 3,6x10^{16} tentatives. En faisant l'hypothèse qu'un superordinateur peut tester 1 million de clés par seconde, il faudra environ 1000 ans pour trouver la bonne clé. De nos jours, les cryptosystèmes utilisent au minimum des clés de 128 bits.

La notion de clé cryptographique permet de définir deux schémas de base des cryptosystèmes ou algorithmes cryptographiques. Les algorithmes à clés secrètes ou cryptage symétrique et les algorithmes à clés publiques ou cryptage asymétrique.

7.2.2 Cryptage symétrique

Dans un système de cryptage symétrique, la même clé est aussi bien utilisée pour le chiffrement que pour le déchiffrement des messages. On parle alors de chiffrement à clés secrètes. Parmi les cryptosystèmes à clé secrète les plus utilisés, nous trouvons le DES (Data Encryption Standard) (cf §6.2).

7.2.3 Cryptage asymétrique

Dans le mode de cryptage asymétrique, deux clés sont utilisées, une clé publique et une clé secrète. On parle alors de chiffrement à clés publiques. Ce type de chiffrement a été inventé en 1976 par W. Diffie et M. Hellman [81] pour résoudre le problème de gestion des clés posé par les chiffrements symétriques. Dans ce type de chiffrement, tout le monde peut utiliser votre clé publique pour vous envoyer des messages chiffrés que seul vous pouvez déchiffrer avec votre clé secrète. Il offre également la possibilité de créer des signatures numériques.

Parmi les algorithmes asymétriques standard de type DSS (Digital Signature Standard) avec notamment l'algorithme de signature numérique DSA (Digital Signature Algorithm) et l'algorithme de signature numérique par courbes elliptiques ECDSA (Elliptic Curve Digital Signature Algorithm), le RSA reste le plus populaire et le plus utilisé.

7.3 Définition de la cryptanalyse

La cryptanalyse est le domaine de la cryptologie consacré à l'étude des méthodes ou techniques permettant de restituer un message chiffré en clair sans pour autant connaître la ou les clés utilisée(s). L'objectif de cette science est de pouvoir mettre en évidence les faiblesses d'un cryptosystème. On parle alors de tentatives de déchiffrement ou d'attaques cryptanalytiques.

Il existe cinq types d'attaques génériques présentées ici par ordre d'efficacité :

- l'attaque avec un texte chiffré : le cryptanalyste dispose du texte chiffré de plusieurs messages obtenus avec le même algorithme.
- l'attaque avec un texte en clair connu : le cryptanalyste dispose des textes chiffrés et des textes en clair correspondants.
- l'attaque avec un texte en clair choisi : le cryptanalyste a accès aux textes chiffrés et aux textes en clair, mais il peut choisir les textes en clair à chiffrer.
- l'attaque avec un texte adaptif : le cryptanalyste peut choisir les textes en clair, mais il peut également adapter ses choix en fonction des textes chiffrés précédemment.

– l'attaque avec texte chiffré choisi : le cryptanalyste peut choisir différents textes chiffrés à déchiffrer.

Ces attaques reposent sur le principe de *Kerckhoff* [87] stipulant que la connaissance complète et détaillée du cryptosystème n'est pas un secret pour le cryptanalyste. Ainsi, la sécurité et la robustesse de tout algorithme cryptographique ne repose pas sur la non connaissance de cet algorithme, mais essentiellement sur les propriétés mathématiques de l'algorithme et sur la longueur des clés utilisées.

De nos jours, la conception de tels systèmes nécessite la collaboration de concepteurs de différents domaines (informatique, mathématique, microélectronique, etc.) car le domaine de la cryptanalyse est un domaine très ouvert faisant appel à des compétences aussi larges que possible. La figure 7.1 représente les classes et les sous classes des différentes méthodes cryptanalytiques en fonction des compétences mises en œuvre. Dans chacune de ces méthodes on peut utiliser chacun des cinq modèles d'attaques génériques définis ci-dessus.

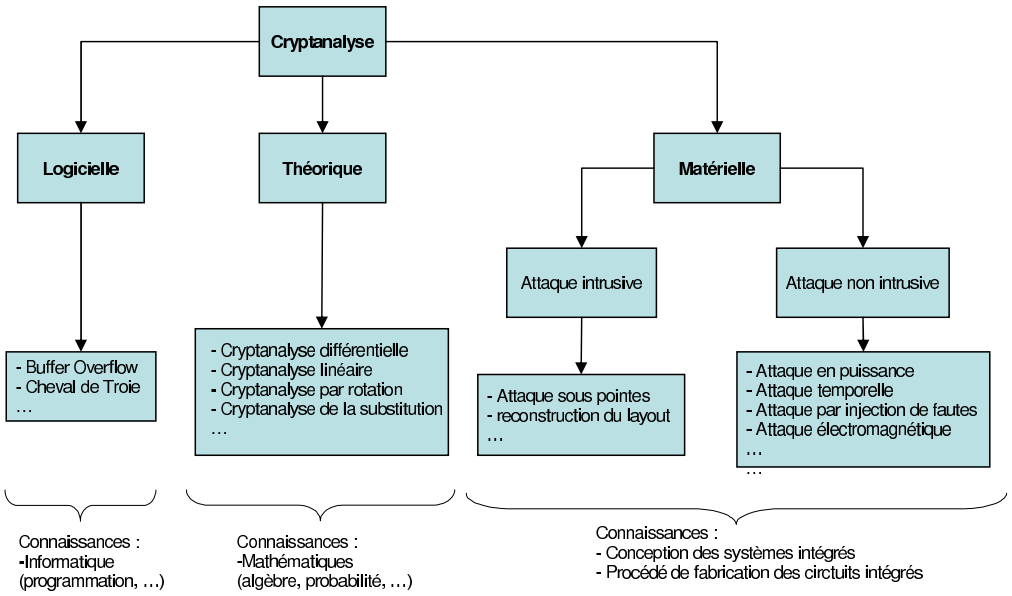


FIGURE 7.1 – Les attaques cryptanalytiques et les compétences requises.

Il existe trois grandes familles d'attaques cryptanalytiques, les attaques dites théoriques, les attaques dites logicielles (Software) et les attaques dites matérielles (Hardware).

7.4 Cryptanalyse dite théorique

Les attaques dites théoriques sont basées sur des analyses utilisant des théories mathématiques pour briser des algorithmes cryptographiques. Elles exploitent les faiblesses de conception théorique des cryptosystèmes pour générer des attaques. Ce type d'attaques permet aux cryptanalystes d'évaluer théorique-

ment la force ou la fiabilité d'un algorithme cryptographique.

Parmi les attaques les plus connues ont trouve la cryptanalyse linéaire et la cryptanalyse différentielle [81].

7.5 Cryptanalyse dite logicielle

Elle regroupe toutes les techniques qui permettent d'accéder aux informations confidentielles d'un cryptosystème en exploitant les failles des systèmes d'exploitation des composants.

Les attaques bien connues comme le cheval de Troie ou le débordement de pile (pour ne citer que celles là) développées et utilisées sur les systèmes d'exploitation des ordinateurs personnels, ont montré toute leur efficacité sur des systèmes d'exploitation moins complexes comme ceux embarqués dans des cartes à puce.

7.6 Cryptanalyse dite matérielle

Cette branche de la cryptanalyse regroupe l'ensemble des techniques cryptanalytiques exploitant les vulnérabilités matérielles des circuits intégrés dédiés à des applications sécurisées. On distingue deux catégories d'attaques matérielles : les attaques intrusives et les attaques non intrusives.

7.6.1 Les attaques intrusives

Les attaques intrusives sont directement réalisées sur le silicium du composant. Elles commencent par un désassemblage de la puce de son boîtier. La puce ainsi mise à nue est attaquée par des solvants chimiques pour retirer les différentes couches de passivation, d'isolation (etc.) utilisées dans les procédés de fabrication des circuits intégrés. Ces accès physiques sur la puce, permettent soit de reconstituer le layout de la puce, soit de réaliser des analyses sous pointes. Ces attaques sont destructrices.

7.6.2 Les attaques non intrusives

Contrairement aux attaques intrusives, les attaques non intrusives ne sont pas destructrices. Selon les méthodes d'analyses mises en place, les attaques non intrusives peuvent être rangées en deux classes : les attaques par injection de fautes et les attaques par analyse des signaux compromettants encore appelées attaques par canaux cachés. C'est cette seconde classe d'attaques qui nous intéresse plus particulièrement dans ce chapitre.

7.6.2.1 Attaques par injection de fautes

Le principe des attaques par injection de fautes consiste à générer délibérément des fautes dans un composant en fonctionnement, dont l'exploitation (des fautes générées) permettra de briser les sécurités du composant. Les fautes sont générées en faisant fonctionner le composant dans des conditions anormales. L'implantation des attaques par injection de fautes se déroule en deux phases : la génération de la faute et l'exploitation de la faute générée pour accéder aux données confidentielles du système.

Il a été montré dans [69], qu'il existe des méthodes pour sécuriser de façon extrêmement efficace les circuits asynchrones contre ce type d'attaques.

7.6.2.2 Attaques par analyse des signaux compromettants

Les attaques par canaux cachés sont basées sur la recherche et l'exploitation de toute corrélation entre les données manipulées par un circuit et ces signaux externes. Ces signaux, sont soit les signaux d'alimentation et les signaux d'horloge, soit les signaux électromagnétiques émis par un circuit. Ils sont appelés signaux compromettants ou canaux cachés du fait qu'ils peuvent faire fuir des informations indésirables. En effet, la notion de canal est définie comme un endroit par lequel transite de l'information, et il est caché quand son utilisation est détournée de son objectif initial.

Les attaques utilisant des signaux compromettants et notamment les signaux électromagnétiques sont bien connues des militaires et des agences d'espionnage depuis la seconde guerre mondiale avant de devenir un domaine de recherche académique publique dans les années 90.

De nos jours, les études sur des corrélations entre les données traitées par un système cryptographique et les signaux compromettants ont donné lieux à de nombreuses méthodes ou techniques d'attaques. En fonction du signal compromettant utilisé, on les classe en attaques temporelles, en attaques électromagnétiques et en attaques en puissance.

Nous allons dans cette partie décrire rapidement chacune de ces attaques afin d'introduire plus précisément les attaques en puissance qui seront utilisées plus loin dans ce chapitre.

7.6.2.2.1 Attaques temporelles Les attaques temporelles exploitent les dépendances temporelles entre les données manipulées et le temps nécessaire au calcul des données. Le principe de l'attaque est de déterminer toute corrélation entre le temps d'exécution d'une fonction (en terme de cycle d'horloge) et les données traitées.

Deux phases sont nécessaires à la réalisation de l'attaque. Dans la première phase, on détermine

des corrélations entre les données et le temps de calcul. Cette analyse peut être faite sur deux niveaux d'abstraction : au niveau architectural (schéma d'implantation) et au niveau algorithmique. Lorsque des corrélations sont observées, elles sont analysées dans la seconde phase afin d'examiner si elles sont exploitables pour retrouver des informations confidentielles.

7.6.2.2.2 Attaques électromagnétiques Les possibilités de capture et d'analyse des émanations électromagnétiques des systèmes électroniques (écran de télévision, ordinateur, etc.) sont bien connues des scientifiques depuis les années 50. Les recherches dans le domaine, afin de diminuer les interférences électromagnétiques des systèmes électroniques, sont restées longtemps classées secret défense par les grandes nations. Il a fallu attendre les années 85 avec la publication des travaux de Win van Eck sur les conséquences des interférences des appareils électroniques sur la sécurité des systèmes pour porter le sujet dans le milieu académique. Dans ses travaux, il démontre la possibilité de reconstruire un signal vidéo par analyse et décodage des interférences électromagnétiques émises par un récepteur vidéo.

De nos jours, les effets d'interférence des composants électroniques sur les appareils électroniques sont standardisés. Une directive de compatibilité électromagnétique (CEM ou EMC en anglais pour ElectroMagnetic Compatibility) a été rendue obligatoire depuis le 1er janvier 1996. Elle permet de caractériser l'aptitude d'un composant ou d'un équipement à fonctionner de manière satisfaisante dans un environnement et sans y occasionner lui-même des perturbations gênantes. Le terme de CEM regroupe deux concepts :

- L'émissivité (EMI pour ElectroMagnetic Interference) qui caractérise les émissions électromagnétiques produites par un appareil électronique dans un environnement.
- L'immunité (EMS ElectroMagnetic Susceptibility) qui caractérise la susceptibilité électromagnétique d'un appareil électronique dans un environnement.

Les études et les analyses des propriétés électromagnétiques ont permis aux cryptanalystes de développer des techniques d'attaques exploitant le rayonnement électromagnétique des circuits intégrés pour briser des codes cryptographiques.

Les attaques par analyse électromagnétique sur les composants dédiés à des applications cryptographiques ont connu un développement considérable depuis la publication il y a quelques années par la NSA (National Security Agency) de rapports scientifiques classés secret défense sur la cryptanalyse matérielle par effets électromagnétiques.

La majorité des circuits numériques sont cadencés par un signal global d'horloge qui impose une synchronisation des traitements dans toutes les parties du circuit. Durant chaque cycle d'horloge (front

montant ou descendant), tous les signaux sont mémorisés dans des registres (ou mémoires) et des nouveaux calculs sont alors exécutés entraînant des appels en courant non négligeables. Dans ce type de circuit, le spectre électromagnétique est essentiellement dû à l'activité électrique des éléments logiques et des points mémoires constituant le circuit. Ces éléments (logiques et mémoires) sont constitués de transistors fonctionnant comme des commutateurs commandés par une tension de grille. Lorsque la tension de grille est présente, un transfert de charge entre le drain et la source s'effectue à travers le canal du transistor. Ce transfert de charge a pour effet de favoriser un appel en courant et d'engendrer une émission électromagnétique.

L'activité électrique des circuits intégrés est due aux transferts de charges lors de la commutation des éléments logiques de 0 vers 1 et de 1 vers 0. Plus le niveau d'intégration des transistors augmente, impliquant des fréquences de fonctionnement de plus en plus élevées et des délais de portes de plus en plus faibles, plus les variations du signal d'alimentation croissent augmentant ainsi les émissions électromagnétiques [73].

Les caractéristiques du champ électromagnétique (en terme de fréquence et amplitude) sont donc relatives à l'activité électrique et à la nature des données manipulées dans le circuit. Par conséquence, le principe de l'attaque électromagnétique est d'exploiter cette dépendance afin de déterminer toute corrélation entre les données traitées et les émissions électromagnétiques.

7.6.2.2.3 Attaques en puissance ou par analyse en courant Contrairement aux attaques électromagnétiques qui exploitent l'activité électrique des circuits intégrés sous forme d'émissions électromagnétiques, les attaques en puissance exploitent l'activité électrique des composants en mesurant leur profil de courant. Elles sont focalisées sur la recherche de corrélation entre les données manipulées et les caractéristiques des signaux d'alimentation (Vdd ou Gnd) du circuit (variations en amplitudes, pics de consommation, décalage temporel, puissance, etc.). Une fois des corrélations observées, des méthodes d'analyse sont mises en œuvre afin d'exploiter ces dépendances pour briser le circuit. Les bases des attaques en puissance reposent sur l'hypothèse suivante :

En technologie CMOS la puissance consommée par un élément logique (porte) est différente selon qu'elle charge ou décharge sa capacité de sortie.

$$P_{charge} \neq P_{décharge} \Rightarrow i_{charge} \neq i_{décharge}$$

Ces différences qui sont directement liées à la nature des données manipulées sont exploitées pour établir des corrélations entre les courbes de courant et les données. Plusieurs techniques et méthodes ont été développées en vue d'exploiter cette dépendance pour être appliquées avec succès sur la majeure partie des cryptosystèmes. Ces méthodes permettent de retrouver directement les clés cryptographiques.

Les attaques par analyse du courant sont parmi les attaques matérielles les plus efficaces, les plus faciles à mettre en œuvre et les moins coûteuses. Elles sont à la portée de tout laboratoire possédant un oscilloscope, une chaîne d'acquisition du courant et un ordinateur de bureau.

7.7 Attaques en puissance ou par analyse en courant

Les premières notes techniques proposant l'exploitation des variations des signaux d'alimentation en cryptanalyse furent introduites dans les années 90 par Paul Kocher [54]. Parmi les méthodes proposées par ce dernier, on notera les attaques par simple analyse du courant (SPA pour Single Power Analysis) et les attaques par analyse différentielle du courant (DPA pour Differential Power Analysis). Excepté les différences liées aux méthodes d'analyse et de détermination des informations confidentielles misent en place dans chacune de ces approches, ces deux types d'attaques sont basés sur des analyses de corrélation entre les données manipulées et le courant consommé.

7.7.1 Mesure du courant

Cette phase correspond à la phase d'acquisition des courbes de courant. Les vecteurs utilisés pour les mesures sont choisis ou aléatoires. Le schéma de la figure 7.2 présente la plateforme utilisée pour effectuer et acquérir automatiquement les mesures de courant. Les mesures sont généralement réalisées sur la masse en effectuant une mesure différentielle aux bornes d'une résistance placée entre la masse du composant et celle du système d'acquisition. Le même type de mesure peut être fait sur le signal V_{dd} du composant, dans ce cas, ce signal doit être isolé du reste du système pour limiter des effets parasites de bruit.

Trois facteurs sont essentiels dans les mesures du courant pour la réalisation d'une attaque en puissance : l'amplitude, le nombre de pics de courant relatifs à l'activité électrique du composant et le rapport signal sur bruit.

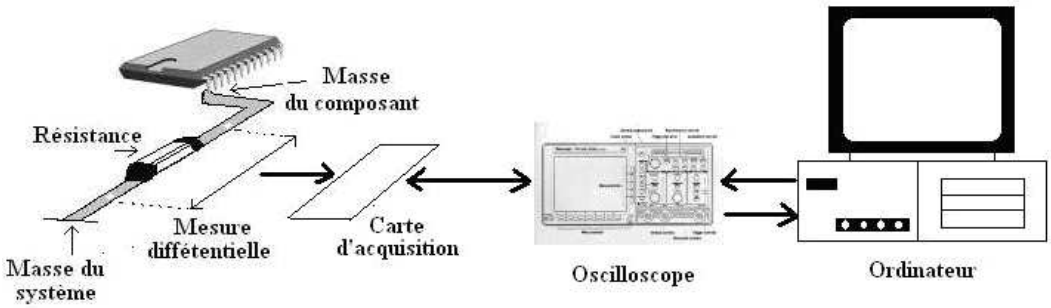


FIGURE 7.2 – Schéma de mesure et d'acquisition automatique des courbes de courant.

Afin d'améliorer la qualité du signal du courant (amplitude et nombre de pics), une résistance de

faible valeur est utilisée (inférieure à 10 ohms). Plus la résistance est faible, plus les amplitudes et le nombre de pics de courant sont importants et observables.

$$I = \frac{V}{R} \xrightarrow[R \to 0]{} \infty \quad \ldots \quad I = \frac{V}{R} \xrightarrow[R \to \infty]{} 0 \tag{7.1}$$

Une seconde approche consiste à faire varier le signal d'alimentation notamment en réduisant la tension d'alimentation du circuit. Une faible alimentation du circuit entraîne une augmentation des délais, ce qui permet d'accroître les pics relatifs à l'activité électrique des fonctions du circuit. Toutefois, la réduction de la tension d'alimentation rapproche le niveau du courant de celui du bruit. Le rapport signal sur bruit est amélioré en utilisant l'approche par moyennage des courbes. En effet, le rapport signal sur bruit est proportionnel à la racine carrée du nombre de courbes mesurées (N) :

$$SNR = \sqrt{N} \frac{S_{signal}}{\sigma_{bruit}} \tag{7.2}$$

σ_{bruit} est la variance du bruit

Ainsi, la mise en place d'un flot d'acquisition automatique des courbes de courant doit permettre de mesurer avec le maximum de précision possible, les amplitudes des pics de courant nécessaires à la réussite des attaques en puissance. Pour cela, des filtres de bruit et des amplificateurs d'instrumentation peuvent être rajoutés dans la chaîne d'acquisition.

7.7.2 Attaques par simple analyse du courant

Cette attaque permet par une simple analyse du courant, d'établir des corrélations entre le profil de la forme d'onde du courant et les données manipulées. Selon l'algorithme et le type d'implantation utilisés, elle permet de retrouver des clés secrètes, ou d'identifier la consommation (pic de courant) de blocs particuliers, d'instructions et de déterminer le poids de Hamming des bus et des registres (données et adresses).

Différents types d'implantation des algorithmes cryptographiques sont vulnérables aux attaques par analyse du courant ou à ses variantes. Les blocs de génération de sous clés des cryptosystèmes symétriques (DES et AES) sont particulièrement exposés à ce type d'attaque, car les analyses sont directement effectuées sur les valeurs de la clé. Les articles [13, 61] présentent des méthodes qui permettent de réussir des attaques sur le bloc de génération des sous clés de l'AES. La figure 7.3 présente les profils en courant récupérés d'une carte à puce lors d'un traitement DES. Les profils en courant correspondants à la permutation initiale, aux 16 rondes et à la permutation finale sont clairement définissables. Dans ce cas précis, l'attaque SPA a permis d'identifier les différentes instructions et leurs instants d'occurrence.

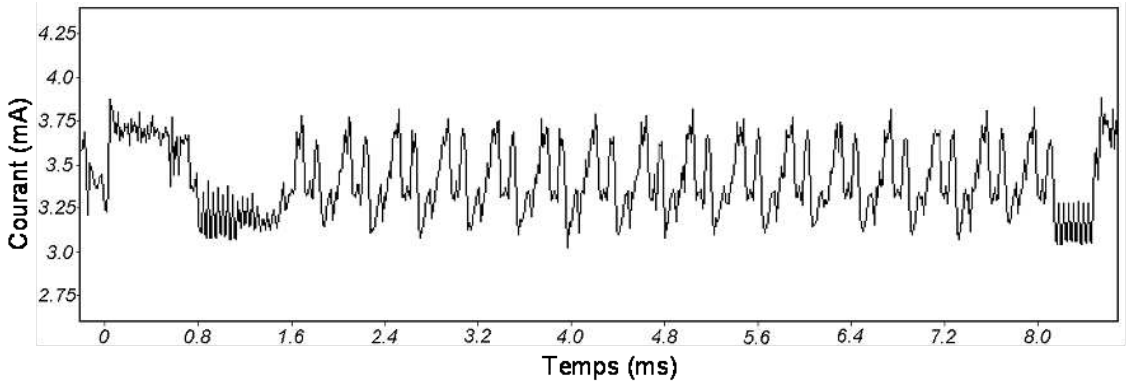


FIGURE 7.3 – Profils en courant correspondant à un traitement DES.

Notez que toute implantation naïve d'algorithme de chiffrement (TDES, AES, RSA, etc.) est vulnérable aux attaques SPA [13]. Cependant la réussite de ces attaques exige une connaissance détaillée de l'algorithme de chiffrement et de la manière dont il est implanté.

Aujourd'hui avec l'apparition de nouvelles techniques de contre-mesures, l'attaque SPA ne représente plus à elle seule une menace contre les cryptosystèmes.

7.7.3 Attaques par analyse différentielle en courant

Contrairement à l'attaque SPA, l'attaque par analyse différentielle de consommation ou DPA est beaucoup plus efficace et difficile à contrecarrer [54]. D'une part, elle ne nécessite pas de connaître dans les détails l'implantation du cryptosystème et d'autre part, par des analyses statistiques, elle est capable d'exploiter les plus petites variations du courant pour retrouver des informations secrètes contenues dans les cartes à puce comme la clé privée RSA ou la clé de chiffrement de l'algorithme DES.

Globalement, la stratégie de l'attaque DPA est d'éliminer toute information inutile (bruit de mesure, consommation des parties non sensibles du cryptosystème, ...) et d'amplifier celle corrélée avec les données secrètes.

L'attaque DPA se déroule en deux étapes : la collecte puis l'analyse des données. La collecte des données consiste à récupérer les couples cryptogramme – consommation. L'analyse des données consiste en une analyse statistique de la consommation mesurée durant les calculs cryptographiques. Pour illustrer ces différentes étapes, considérons le cas d'une attaque DPA sur un cryptosystème DES.

7.7.3.1 Notions préliminaires

Pour rappel, le DES est un algorithme de chiffrement par bloc qui repose sur des principes simples dont des permutations, des substitutions, des échanges de blocs de données et une fonction prenant en

entrée une clé intermédiaire à chaque ronde. Chaque clé intermédiaire est calculée à partir de la clé de chiffrement de 64 bits. Dans le cas d'une attaque sur texte chiffré seulement, il est important de noter que pour retrouver la totalité de la clé de chiffrement du DES, il faut au moins trouver les clefs intermédiaires des deux dernières rondes ([81]).

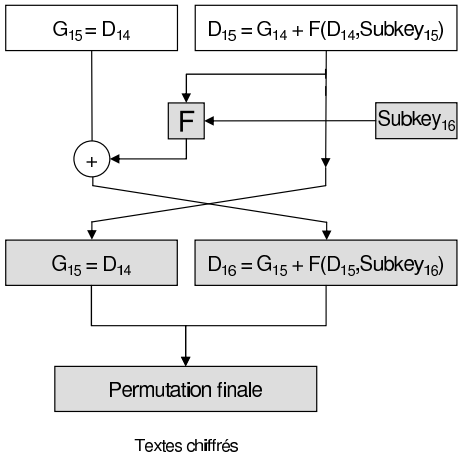


FIGURE 7.4 – Dernière ronde du DES.

La figure 7.4 présente la dernière ronde de l'algorithme DES. Si on regarde de près cette dernière ronde, et si on dispose des textes chiffrés, finalement la seule inconnue est la clé intermediaire $Subkey_{16}$. En ce qui concerne le message D_{16}, les propriétés de la fonction XOR font qu'il peut être fonction de la clé intermédiaire ($Subkey_{16}$) et du message chiffré (D_{15} et D_{16}) :

$$G_{15} = D_{16}\ xor\ F_{out} = D_{16}\ xor\ (D_{15}, Subkey_{16}) \tag{7.3}$$

7.7.3.2 Notion de fonction de sélection

Comme définie dans [54], une fonction de sélection est une fonction qui calcule une valeur intermédiaire de l'algorithme DES en fonction d'une partie de la clé intermédiaire et d'une partie du message chiffré. L'expression 7.3 est une illustration parfaite de ce qu'est une fonction de sélection.

Toutefois pour des raisons de simplification et d'efficacité dans les analyses, il est préférable de définir une fonction de sélection ciblant un bloc moins important mais tout aussi sensible du DES. Dans notre cas d'étude, nous considérerons une fonction de sélection D mettant en jeu la première table de substitution (SBOX1), 6 bits de la clé intermédiaire (k) et 6 bits du message chiffré (m) :

$$D(m, k) = SBOX1(m \oplus k) \tag{7.4}$$

Soulignons qu'une fonction de sélection est utilisée lors de l'analyse des données pour effectuer les hypothèses de clé et pour vérifier si une hypothèse de clé est la bonne. Avec la bonne hypothèse de clé, la fonction de sélection calculera la valeur intermédiaire correcte de D (4 bits) pour chaque message chiffré

m. En d'autres termes, la fonction de sélection sera corrélée avec chaque bit de D manipulé au niveau de la 16^{e} ronde du DES. Étant donné la dépendance entre la consommation et les données manipulées, des analyses de corrélation entre chaque bit de D et la consommation du circuit (mesurée pendant la 16^{e} ronde avec l'utilisation de la vraie clé k) permet alors de vérifier l'exactitude d'une hypothèse de clé.

7.7.3.3 Collecte des données

Dans cette phase, il s'agit de mesurer de façon précise la consommation en fonction du temps du cryptosystème étudié. Dans le même temps, il faut récupérer les cryptogrammes disponibles en sortie du circuit. Pour les analyses de corrélation, il faut en effet un nombre arbitrairement élevé de couples cryptogramme – consommation. En général, les cryptogrammes et les traces de courant sont mémorisés sous forme matricielle dans un ordinateur. De plus, pour limiter l'utilisation mémoire, il est nécessaire de délimiter une zone d'acquisition des traces de courant. Dans le cas d'une attaque DPA sur texte chiffré seulement, seules les traces de courant correspondant à la 16^{e} ronde sont nécessaires. Cette zone peut être facilement localisée par une attaque SPA.

Prenons le cas d'un DES pour lequel nous lançons le chiffrement de n messages aléatoires (M_i) avec la même clé k. Pour chaque message M_i, nous avons mémorisé les traces de courant dans le temps T_{ij} (j indique le numéro de l'échantillon) et le cryptogramme obtenu C_i (4 bits). Comme convenu, les données sont stockées sous forme matricielle :

$$\underbrace{\begin{pmatrix} M_1 & C_1 \\ \dots & \dots \\ M_i & C_i \\ \dots & \dots \\ M_n & C_n \end{pmatrix}}_{Messages} \quad \underbrace{\begin{pmatrix} T_{11} & \dots & T_{1j} & \dots & T_{1k} \\ & & & & \\ T_{i1} & \dots & T_{ij} & \dots & T_{ik} \\ & & & & \\ T_{n1} & \dots & T_{nj} & \dots & T_{nk} \end{pmatrix}}_{Traces\ de\ courant}$$

7.7.3.4 Analyse des données

Concrètement, cette étape de la DPA consiste à effectuer des hypothèses de clé et à déterminer l'hypothèse correcte. Pour y arriver, il faut se baser sur la dépendance entre les données traitées et les traces de courant.

Considérons la fonction de sélection présentée dans l'expression 7.4. Pour chaque hypothèse de clé k_s (s = [1-64]), elle donnera une valeur théorique notée D_i (4 bits) du cryptogramme obtenu pour chaque message en clair M_i. Pour commencer l'analyse des données, les traces de courant T_{ij} sont réparties en deux ensembles en fonction de la valeur du bit b de D_i appelé communément le bit cible. On constituera

alors deux ensembles T_0 et T_1 :

$$T_{ij} \in T_0 | D_i(M_i, k_s)[b] = 0 \tag{7.5}$$

$$T_{ij} \in T_1 | D_i(M_i, k_s)[b] = 1 \tag{7.6}$$

Pour chaque hypothèse de clé, on calcule M_0 et M_1, les moyennes respectives des ensembles T_0 et T_1.

$$M_{0j\in[1-k]} = \frac{\sum_{i=1}^{n}(1 - D_i(M_i, k_s)[b]).T_{ij\in[1-k]}}{n - \sum_{i=1}^{n}(D_i(M_i, k_s)[b])} \tag{7.7}$$

$$M_{1j\in[1-k]} = \frac{\sum_{i=1}^{n}(D_i(M_i, k_s)[b]).T_{ij\in[1-k]}}{n - \sum_{i=1}^{n}(D_i(M_i, k_s)[b])} \tag{7.8}$$

Finalement il faut déterminer la différence de ces deux moyennes qui représente la signature DPA notée S_K^{DPA}.

$$S_K^{DPA} = M_1 - M_0 \tag{7.9}$$

Pour la suite, nous nous référerons à la forme matricielle de nos données mémorisées. Si l'hypothèse de clé k_s n'est pas la bonne, le bit b de D_i calculé par la fonction de sélection sera égal au bit b de C_i (donnée réellement manipulée par le circuit) avec une probabilité de $1/2$ pour chaque message M_i. Vu le caractère aléatoire de la fonction de sélection, les traces de courant occasionnées par des transitions de différents types ([0-1] et [1-0]) peuvent se retrouver dans un même ensemble (T_0 ou T_1) et ce dans une proportion identique. Par voie de conséquence, sur un grand nombre d'échantillons n, il est fort probable que les deux moyennes M_0 et M_1 soient égales et que la courbe DPA soit plate et proche de zéro.

D'un autre côté, si l'hypothèse de clé k_s est la bonne, le bit b de D_i calculé par la fonction de sélection sera égal au bit b de C_i avec une probabilité de 1. Dans ce cas, la fonction de sélection sera corrélée à la valeur du bit b de C_i manipulé par le circuit. Par conséquent, les traces de courant occasionnées par les transitions d'un même type ([0-1] et [1-0]) se retrouveront dans un même ensemble (T_0 ou T_1). Sachant que les transitions [0-1] consomment plus par rapport à V_{dd}, que les transitions [1-0], la courbe DPA sera plate et proche de zéro aux instants où la fonction de sélection ne sera pas corrélée avec le bit cible, et affichera des pics de forte amplitude dans le cas contraire.

Dans la figure 7.5, nous présentons le résultat de deux attaques DPA réussies, pour lesquelles de bonnes hypothèses de clé permettent de voir un écart conséquent des courbes par rapport à la moyenne. Avec deux bits cibles donnant le bon résultat, nous pouvons conclure que l'attaque DPA a été menée avec succès.

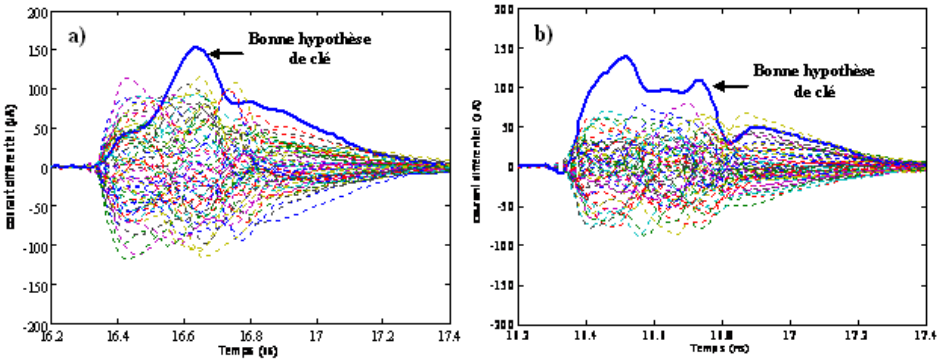


FIGURE 7.5 – Courbes d'attaques DPA faisant ressortir les bits de clé corrects.

Cependant, il est nécessaire de noter qu'il est possible d'obtenir des pics de consommation pour une mauvaise hypothèse de clé bien plus importants que ceux générés par la bonne clé. On parle alors de pics fantômes généralement occasionnés par le bruit. Ceci peut s'expliquer par un nombre trop faible d'échantillons qui ne permet pas d'éliminer le bruit par moyennage.

D'autre part, il existe une version plus efficace de l'attaque DPA qui est « l'attaque DPA de second ordre » (ou « High Order DPA »). Si précédemment les analyses de consommation sont effectuées selon la valeur d'un bit manipulé par le composant, les analyses peuvent s'effectuer à partir de plusieurs bits dans cette seconde version. Par exemple, la répartition de la consommation peut se faire selon la valeur de quelques (2,3) bits de la fonction de sélection.

La logique asynchrone quasi insensible aux délais offre une alternative performante aux circuits synchrones au niveau de la résistance des circuits aux attaques par canaux cachés. Que ce soit au niveau des injections de fautes ([69]), de la résistance aux attaques de type EMA (Electro-Magnetic Analysis) ([73, 15]) ou des attaques par analyse de courant ([16, 15]. Cependant, bien que le niveau de résistance des circuits asynchrones soit élevé par rapport à des réalisations synchrones, en terme de complexité et de mise en œuvre des attaques, des fuites d'informations peuvent subsister. Ces fuites, même si elles vont demander de déployer des efforts très importants pour être exploitées, permettent néanmoins de réussir des attaques. L'objectif dans la suite de ce chapitre est d'analyser formellement les circuits en logique asynchrone QDI afin de pouvoir, dans un premier temps identifier l'origine des fuites d'information résiduelles observées et de proposer par la suite de nouvelles méthodes de conception en vue de supprimer ces fuites ou de les rendre complètement inexploitables par toute attaque en puissance.

7.8 Analyse formelle des circuits asynchrones QDI

L'idée est d'analyser et de vérifier au niveau logique et au niveau électrique la résistance aux attaques en puissance des circuits conçus en logique asynchrone QDI. L'objectif étant de trouver les zones du circuit les plus exposées à des fuites d'information. Nous cherchons dans cette approche à contrôler le flot de conception et notamment les phases de back-end (synthèse et placement routage) afin de supprimer les dissymétries logiques et électriques découvertes dans le circuit. Le schéma suivant décrit le flot d'analyse mis en place.

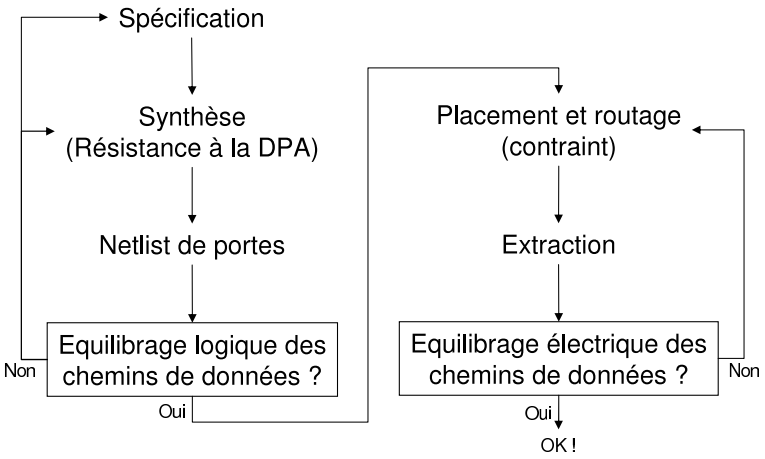


FIGURE 7.6 – Schéma du flot d'analyse.

La spécification du circuit est faite à partir du langage de description de haut niveau CHP. La synthèse des circuits, réalisée à l'aide de notre environnement de conception TAST, nous fournit une netlist de portes logiques. C'est à partir de cette netlist que débute notre travail d'analyse.

Nous présentons dans un premier temps la représentation formelle utilisée pour modéliser les circuits QDI et nous permettant de réaliser les différentes analyses. Nous étudierons ensuite les concepts et les méthodologies pour analyser les déséquilibrages logiques et électriques de nos circuits afin de diminuer leur sensibilité aux attaques.

7.8.1 Représentation formelle des circuits QDI

Notre méthode d'analyse est basée sur la représentation des circuits asynchrones sous forme de graphes orientés (ou digraphe). Ce format permet de manipuler les informations nécessaires à l'analyse de la signature électrique des modules des circuits QDI tant au niveau logique qu'au niveau électrique.

Soit $G = (V, E)$ un graphe orienté.V est l'ensemble des sommets de G et E est l'ensemble des arcs orientés de G. Un circuit représenté à partir de ce modèle possède les deux propriétés suivantes :

– toutes les portes du circuit sont des éléments de l'ensemble V des sommets du digraphe G,

– toutes les interconnexions du circuit sont des éléments de l'ensemble E des arcs orientés de G.

L'orientation des arcs entre les portes va dépendre du sens de propagation de l'information : de la sortie d'une porte p_0 vers une entrée d'une porte p_1. On dit que p_0 est le sommet ascendant de p_1.

La représentation graphique est réalisée à partir de la netlist décrivant le circuit en VHDL ou en Verilog. Cette netlist est obtenue en sortie de l'outil de synthèse TAST.

Pour illustrer le format de représentation des circuits QDI, prenons l'exemple d'une porte AND double-rail suivie par un half-buffer. Dans la figure 7.7, nous présentons la netlist décrite en VHDL de cette petite architecture.

```
ENTITY exemple IS
PORT                    (
                        Resetb: in std_ulogic;
                        A: in std_ulogic_vector (1 downto 0);
                        A_ack: out std_ulogic;
                        B: in std_ulogic_vector (1 downto 0);
                        B_ack: out std_ulogic;

                        S: out std_ulogic_vector (1 downto 0);
                        S_ack: in std_ulogic);
END exemple;
ARCHITECTURE exemple_arch OF exemple IS

 SIGNAL E0, E1, E2, E3, E4: std_ulogic;
 SIGNAL E5, E6, E7        : std_ulogic;
```

```
BEGIN
Inst_Name_00: MULLER2 port map (E0, A(1), B(1));
Inst_Name_01: MULLER2 port map (E1, A(0), B(0));
Inst_Name_02: MULLER2 port map (E2, A(0), B(1));
Inst_Name_03: MULLER2 port map (E3, A(1), B(0));
Inst_Name_04: OR3 port map (E4, E1, E2, E3);
Inst_Name_05: MULLER2Reset port map (Resetb, E5, E0,
S_ack);
Inst_Name_06: MULLER2Reset port map (Resetb, E6, E4,
S_ack);
Inst_Name_07: NOR2 port map (E7, E5, E6);

S(1) <= E5;
S(0) <= E6;
A_ack <= E7;
B_ack <= E7;
END exemple;
```

FIGURE 7.7 – Netlist de portes en VHDL décrivant une porte AND en double-rail.

De cette netlist de portes, nous dérivons le graphe orienté présenté figure 7.8.

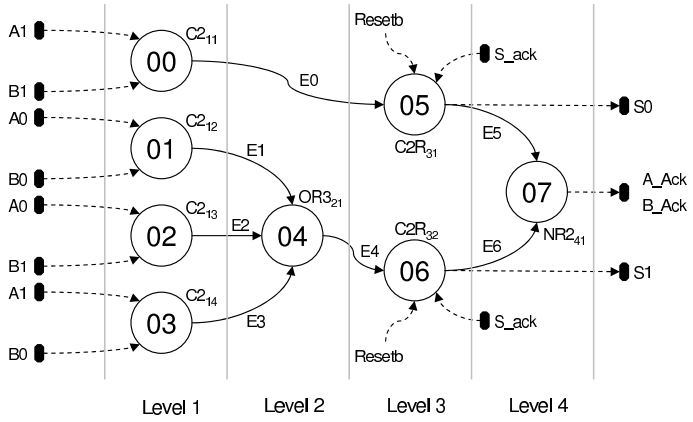


FIGURE 7.8 – Graphe orienté d'une porte AND double-rail.

7.8.2 Analyse au niveau logique : symétrie des chemins de données

La première étape de l'analyse est d'étudier l'équilibre au niveau logique des chemins d'exécution du circuit. Nous appelons chemin d'exécution d'un bloc, tout chemin de données entre les entrées et les sorties de ce bloc permettant d'évaluer une des sorties du bloc. Ces chemins d'exécution sont extraits de la représentation sous forme de graphe orienté du circuit. Pour cela, en partant des sommets de sortie, nous formons des sous-graphes orientés constitués de tous les ascendants de ces sorties.

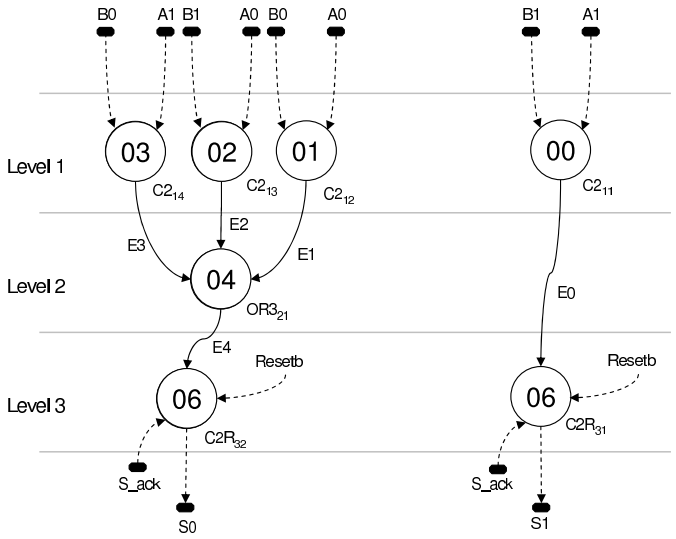


FIGURE 7.9 – Sous-graphes orientés de chacun des sommets de sortie de G_{And}.

Par exemple, figure 7.9 nous présentons les deux sous-graphes obtenus pour le graphe précédent. Ces deux sous-graphes correspondent respectivement aux sorties S_0 et S_1. Pour les sorties $A_{\_ack}$ et $B_{\_ack}$, les sous-graphes obtenus correspondent au graphe complet initial.

Pour chaque sous-graphe, il suffit ensuite d'extraire les différentes possibilités de parcours des chemins de données en fonction des portes qui le composent. Par exemple, pour une porte OR dans un chemin de données, une seule de ses entrées pourra être représentée dans un chemin d'exécution (toutes les entrées d'une porte OR étant exclusives). Par contre, une porte AND (ou une porte de Muller) dans un chemin de données verra toutes ses entrées dans un chemin d'exécution (la sortie de ce type de porte n'étant valide que si toutes ses entrées sont activées). Pour les sous-graphes de la figure 7.9, nous obtenons les chemins d'exécution de la figure 7.10.

Un bloc sera dit équilibré si ses chemins de données sont symétriques au niveau logique. Ce qui équivaut à effectuer les analyses sur les chemins d'exécution des bits de sortie de ce bloc. Il suffit donc de considérer la profondeur logique des différents chemins d'exécution pour évaluer la symétrie logique d'un bloc. Dans l'exemple précédent, il apparaît évident que les chemins d'exécution de la sortie S_0 pos-

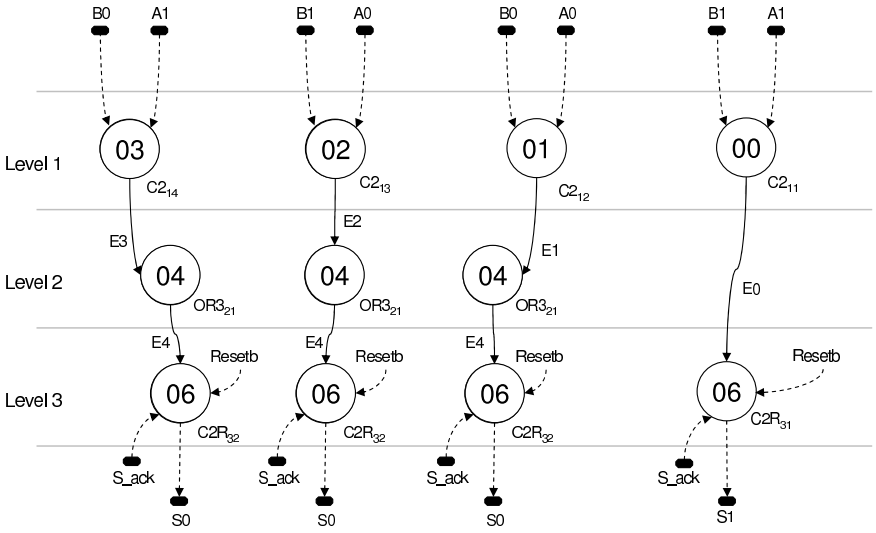


FIGURE 7.10 – Chemins d'exécution du digraphe G_{And}.

sèdent tous la même profondeur logique ; mais possèdent une profondeur logique différente du chemin d'exécution de la sortie S_1. Après avoir réalisé cette analyse, il est possible de corriger facilement cette dissymétrie en ajoutant une porte dans le digraphe associé au chemin de données de la sortie S_1. Nous obtenons le nouveau digraphe présenté figure 7.11.

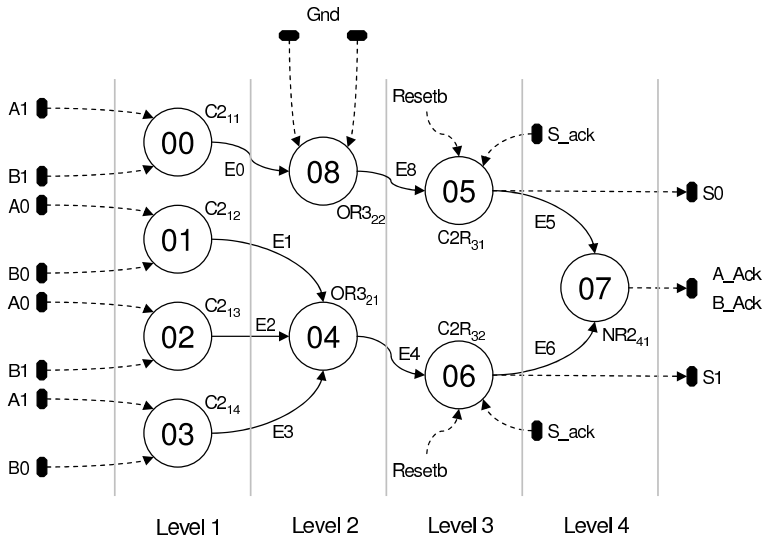


FIGURE 7.11 – Digraphe équilibré d'une porte AND double-rail.

Cette analyse de la symétrie des chemins de données au niveau logique, va nous permettre de corriger si nécessaire l'équilibrage des différents chemins de données au niveau logique, afin de conserver tous les bénéfices de l'utilisation d'un codage en multi-rail et d'un protocole 4 phases : le nombre de transitions logiques du circuit est indépendant des données manipulées.

7.8.3 Analyse au niveau électrique : identification des fuites d'information

On considère ici des circuits comportant des blocs équilibrés au niveau logique, c'est-à-dire que tous les calculs impliquent un nombre constant de transitions logiques, indépendamment de la donnée traitée. On cherche cette fois à équilibrer les chemins d'un bloc asynchrone au niveau électrique. Il a été montré dans [15] que malgré un équilibrage au niveau logique des chemins d'exécution, les capacités de charges des portes ont un effet sur la signature DPA, et rendent le circuit sensible aux attaques en puissance. En effet, la consommation d'un circuit dépend en grande partie de la charge des portes logiques, et en particuliers des capacités parasites et de routage.

L'objectif de cette analyse est de connaître précisément les zones où apparaissent les dissymétries électriques les plus importantes, c'est-à-dire les zones dans lesquelles les dissymétries capacitives sont les plus importantes. Une fois ces zones connues, il est possible de modifier le routage en le contraignant ou encore de regrouper certaines portes en portes plus complexes pour supprimer des nœuds par lesquels transitent des informations compromettantes.

Cependant, contrairement à l'analyse de la profondeur logique des différents chemins d'exécution qui est faite de manière statique, l'analyse des dissymétries électriques du circuit va se faire de manière dynamique. En effet, suivant les valeurs des entrées, des portes vont commuter dans plusieurs chemins d'exécution du circuit ; il est donc nécessaire de prendre en compte dans le calcul global des capacités parasites et de routage, les charges de sortie de toutes ces portes. Pour réaliser cette analyse dynamique de la netlist de notre circuit, nous avons développé un algorithme se comportant comme un simulateur. Le principe est le suivant : pour chaque combinaison de valeurs des entrées du bloc à analyser, une simulation logique du circuit est réalisée. Les capacités de sortie de toutes les portes qui commutent pour cette combinaison d'entrées sont sommées. La somme finale de toutes les capacités est associée aux sorties qui sont évaluées par cette combinaison d'entrées. De cette manière, il est possible de connaître pour chaque sortie la capacité totale mise en jeu pour une combinaison d'entrées donnée. En moyennant les sommes de capacités obtenues pour chaque sortie, nous sommes donc en mesure d'en déduire les chemins d'exécution dynamiques comportant la plus grande dissymétrie capacitive.

Pour illustrer cette méthode d'analyse dynamique, reprenons notre exemple de la porte AND en double-rail. Une fois que la phase de placement routage a été réalisée sur la netlist du circuit, il est possible d'obtenir un fichier de capacités indiquant la valeur des capacités parasites et de routage pour chaque fil de la netlist. Nous présentons tableau 7.1 les sommes de capacités pour chaque sortie du bloc en fonction des combinaisons des valeurs d'entrées. En analysant les valeurs obtenues, nous observons une dissymétrie dans le cas où les entrées A et B valent respectivement 0 et 1. Cette dissymétrie capacitive

peut provoquer une faiblesse du circuit par rapport à une attaque en analyse en courant.

TABLE 7.1 – Exemple d'analyse dynamique de netlist.

Valeurs des entrées	Liste des portes ayant commutées	Somme des capacités pour S0 (pF)	Somme des capacités pour S1 (pF)
A=0 B=0	Inst_01, inst_04, inst_06	0	0,0194
A=0 B=1	Inst_02, inst_04, inst_06	0	0,0228
A=1 B=0	Inst_03, inst_04, inst_06	0	0,0192
A=1 B=1	Inst_00, inst_08, inst_05	0,0189	0
	Moyenne par sortie	0,0189	0,0205

Une fois ces informations extraites, il est possible de modifier la netlist pour la renforcer. Par exemple, en contraignant le placement routage du circuit, il est possible de diminuer cet écart capacitif entre les différents chemins d'exécution.

Cependant, il peut arriver que les sommes des capacités sur plusieurs chemins d'exécution soient identiques, mais qu'il existe tout de même des dissymétries électriques. En effet, il est possible de trouver une dissymétrie capacitive en sortie d'une porte, mais que cette dissymétrie ne soit pas visible dans la somme globale. C'est pourquoi nous avons également implanté la possibilité d'analyser la netlist par niveau logique. C'est-à-dire que nous pouvons connaître et comparer la somme des capacités au niveau de chaque couche logique du circuit. Nous pouvons ainsi détecter une dissymétrie sur un nœud précis du circuit dans un chemin d'exécution donné.

TABLE 7.2 – Analyse électrique par niveau logique du circuit AND double-rail.

Valeurs des entrées	Capacités au niveau logique 1 (pF)	Capacités au niveau logique 2 (pF)	Capacités au niveau logique 3 (pF)
A=0 B=0	Inst_01 : 0,0139	inst_04 : 0,0164	inst_06 : 0,0194
A=0 B=1	Inst_02 : 0,0157	inst_04 : 0,0193	inst_06 : 0,0228
A=1 B=0	Inst_03 : 0,0132	inst_04 : 0,0157	inst_06 : 0,0192
A=1 B=1	Inst_00 : 0,0134	inst_08 : 0,0159	inst_05 : 0,0189

Nous voyons dans le tableau 7.2, que la dissymétrie capacitive se situe en sortie de la porte de Muller inst_02. Cette information nous permet de connaître très précisément les zones du circuit sensibles aux attaques en courant.

7.8.4 Présentation de l'outil d'analyse

Pour réaliser ces différentes analyses sur les netlist de nos circuits, nous avons développé un outil d'analyse de netlist qui s'insère dans le flot de conception de TAST. Le schéma de fonctionnement de cet outil est illustré figure 7.12.

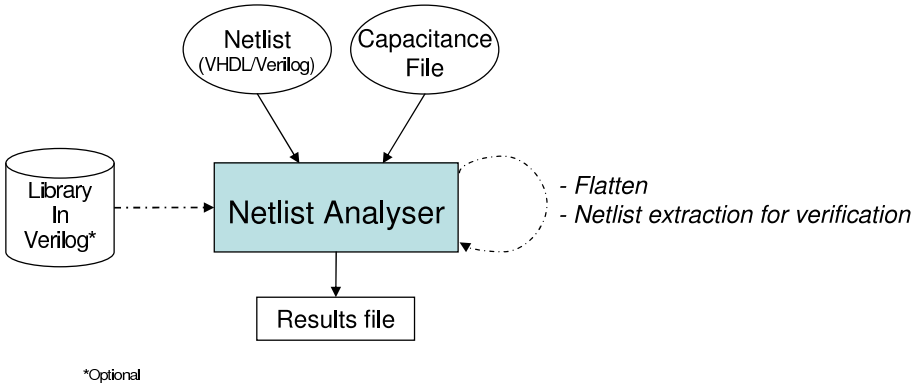


FIGURE 7.12 – Schéma de fonctionnement de l'outil d'analyse de netlist.

L'outil prend en entrée une netlist décrivant le circuit à analyser. Cette netlist (en VHDL ou en Verilog) peut être obtenue soit à partir de la synthèse du circuit dans TAST, soit à partir d'un outil extérieur. Suivant les modules instanciés dans la netlist Verilog (ou VHDL), une bibliothèque de cellules peut être nécessaire en entrée de l'outil. La dernière entrée possible correspond aux fichiers de capacités obtenus en sortie de l'outil de placement routage, nécessaires à l'analyse électrique du circuit.

Des fonctions annexes sont disponibles à l'intérieur de l'outil et permettent :

– de mettre la netlist à plat afin de pouvoir analyser la netlist globale plutôt que par module,

– d'extraire la netlist en VHDL ou en Verilog.

7.9 Conclusion

Afin d'identifier les fuites d'informations observées dans les circuits Quasi Insensibles aux Délais lors d'une attaque en puissance, nous avons présenté dans ce chapitre un outil permettant d'analyser les netlists de circuits sur deux niveaux d'abstractions : au niveau logique et au niveau électrique. Pour ce faire, les circuits QDI sont représentés sous forme de graphes orientés.

En règle générale, l'équilibrage obtenu au niveau logique n'est pas suffisant pour compenser les signatures électriques dues aux capacités de sortie des portes. Ceci notamment à cause des capacités parasites introduites pendant les différentes phases de placement routage et d'optimisation électrique du back-end. Ainsi les analyses électriques permettent de connaître de manière très précise les zones du

circuit susceptibles de laisser fuir des informations. Des modifications peuvent donc être apportées, a posteriori, à la netlist afin de la durcir.

Une validation de l'outil d'analyse est en cours et ne pourra pas être présentée dans ce manuscrit. L'idée est de corréler les résultats obtenus par une vraie attaque DPA sur un AES QDI (présentés dans la thèse de Fraidy Bouesse [15]) à l'aide de notre outil d'analyse. Malheureusement, la complexité du circuit rend l'analyse fastidieuse.

Conclusion

De nos jours, le principal frein à l'adoption des circuits asynchrones dans l'industrie est le manque d'outils de conception d'un niveau industriel pour ce type de logique. Ce travail de thèse a pour objectif de participer au développement d'un environnement de conception de circuits asynchrones Quasi Insensibles aux Délais. De plus, le fort potentiel offert par la logique asynchrone pour faire face aux attaques par analyse en courant, nous a poussé à travailler dans cette thèse sur l'application de nos méthodes de conception aux systèmes sécurisés.

Nous avons présenté dans ces travaux de recherche une méthode novatrice pour réaliser la phase de projection technologique des circuits asynchrones de type QDI. Dans la première partie de ce manuscrit, nous avons introduit les principaux concepts de la logique asynchrone ainsi que les différentes méthodologies de conception académiques existantes. Les points forts et les points faibles de chacune de ces méthodologies ont été présentés. Par la suite, nous avons abordé les concepts de projection technologique. Les algorithmes utilisés dans le domaine synchrone ne permettent pas d'adresser les circuits asynchrones, ces derniers nécessitant une conception plus rigoureuse afin d'élaborer une logique sans aléa. La phase de projection technologique nécessite, par définition, l'utilisation de bibliothèques de cellules cibles. Nous avons donc présentés dans la deuxième partie du manuscrit une bibliothèque de cellules asynchrones développée durant cette thèse : la bibliothèque TAL.

Une analyse des choix de conception de cette bibliothèque ainsi que ses principales caractéristiques ont été présentés. L'utilisation de la bibliothèque TAL permet des gains en surface, vitesse et consommation, en comparaison de l'utilisation de portes standard de type AO222. Les algorithmes développés pour la projection technologique des circuits asynchrones QDI ont ensuite été abordés. Chaque étape de la projection technologique est explicitée séparément :

- La phase de décomposition proposée – basée sur l'utilisation de diagrammes de décision multi-valués – permet de décomposer un circuit QDI en netlist de portes à deux entrées sans introduire d'aléa dans le circuit.
- La phase de partitionnement proposée découpe le circuit en partant de ses sorties primaires. Cette étape permet de définir des sous-circuits comportant un ensemble d'entrées et une seule sortie.

Aucun aléa n'est introduit dans le circuit par cette étape.

– La phase de couverture adaptée d'algorithmes provenant du domaine synchrone achève la projection technologique en ciblant la bibliothèque de cellules choisie.

Les exemples de circuits conçus à partir de cette méthodologie ont confirmé d'une part le bon fonctionnement de la méthode, les circuits obtenus respectant les propriétés d'insensibilité aux délais, et ont d'autre part permis de mettre en évidence les gains importants en terme de surface, vitesse et consommation obtenus par l'utilisation de ces algorithmes.

Finalement, nous avons présenté dans la dernière partie du manuscrit l'application de ces travaux de thèse sur des circuits sécurisés. Tout d'abord nous avons présenté une validation du fonctionnement de notre bibliothèque TAL ainsi que son intégration dans un flot industriel, au travers de la conception de circuits cryptographiques de type DES. Un premier circuit DES réalisé au laboratoire TIMA nous permet de réaliser une comparaison par rapport au même circuit conçu à partir de portes standard. Cette comparaison confirme la nécessité d'utiliser une bibliothèque de cellules spécifiques dédiée aux circuits asynchrones, les gains apportés par l'utilisation de TAL étant très intéressants. Un second circuit DES a été conçu suivant un cahier des charges défini par STMicroelectronics afin d'évaluer l'intégration d'un DES asynchrone dans un produit smartcard existant. Les contraintes de surface, vitesse et consommation ont pu être respectées grâce à des optimisations de l'architecture du DES, mais également grâce à l'utilisation de nos algorithmes de projection technologique. Les résultats obtenus sont très prometteurs. Enfin, plusieurs optimisations de l'architecture du DES ont été étudiées, et des gains en surface et consommation ont été obtenus. Ces résultats prouvent que des circuits asynchrones QDI peuvent être viables en terme de surface.

Le dernier chapitre propose un outil d'évaluation de la sensibilité des circuits aux attaques différentielles, notamment au niveau logique et électrique. Cette analyse, réalisée très tôt dans la phase de conception, localise de façon précise les zones du circuit susceptibles de laisser fuir une information secrète. Il est ensuite possible de modifier la neltist du circuit afin d'augmenter la résistance de ces zones.

Ces travaux de recherches ont également ouvert des perspectives intéressantes. Les résultats obtenus montrent l'intérêt de l'utilisation de bibliothèques spécifiques et notamment l'exploitation des propriétés de cellules en logique double-rail [76]. Des recherches préliminaires ont été menées sur des algorithmes de projection technologique sur des portes à plusieurs sorties. Peu de travaux traitent ce sujet, la plupart dans le domaine de la synthèse à base de logique Domino et dans le domaine de la synthèse sur FPGA. De nouvelles fonctionnalités de cellules seraient intéressantes à étudier pour enrichir la bibliothèque TAL.

Une étude comparative sur la sensibilité aux attaques par analyse en courant des circuits utilisant des cellules double-rail par rapport à des circuits basés sur une bibliothèque simple-rail serait également intéressante à mener avec notre outil d'analyse.

L'objectif final serait de réaliser dans TAST un flot automatisé de conception de circuits asynchrones sécurisés. Pour cela, l'insertion de contre-mesures lors de la phase de conception, telles que des jitters temporels devraient être étudiés au niveau de la phase de projection technologique.

Des projets sont actuellement menés au sein du laboratoire TIMA pour concevoir des FPGAs asynchrones. Une étude préliminaire a également été menée durant cette thèse montrant que les MDDs sont particulièrement adaptés pour la projection technologique sur FPGA.

Bibliographie

[1] A. BARDSLEY and D. EDWARDS. *Compiling the Language Balsa to Delay-Insensitive Hardware*. Hardware Description Languages and their Applications. Kloos C.D. and Cerny E., 1997.

[2] A. BARDSLEY and D. EDWARDS. The balsa asynchronous circuit synthesis system. In *Forum on Design Languages*, 2000.

[3] P.A. BEEREL and T.H.-Y. MENG. Automatic gate-level synthesis of speed-independent circuits. In *ICCAD, Proceedings of the International Conference of Computer-Aided Design*, pages 581–586, 1992.

[4] P.A. BEEREL, K.Y. YUN, and W.C. CHOU. Optimizing average-case delay in technology mapping of burst-mode circuits. In *International Symposium on Circuits and Systems*, pages 244–259, 1996.

[5] E. BEIGNÉ and P. VIVET. Design of on-chip and off-chip interfaces for GALS NoC architecture. In *12th IEEE International Symposium on Asynchronous Circuits and Systems*, pages 172–181, Grenoble, France, March 2006.

[6] L. BENINI and G. DE MICHELI. A survey of boolean matching techniques for library binding. *ACM Transactions on Design Automation of Electronics Systems*, 2 :193–226, 1997.

[7] K. Van BERKEL. Beware the isochronic fork. *Integration, the VLSI journal*, 13(2) :103–128, 1992.

[8] K.V. BERKEL. *Handshake Circuits : an asynchronous architecture for VLSI Programming*, volume 5 of *International Series on Parallel Computation*. Cambridge University Press, 1993.

[9] K.V. BERKEL, R. BURGESS, J. KESSELS, A. PEETERS, M. RONCKEN, and F. SCHALIJ. A fully asynchronous low-power error corrector for the DCC player. *IEEE Journal of Solid-State Circuits*, 29(12) :1429–1439, 1994.

[10] K.V. BERKEL, R. BURGESS, J. KESSELS, A. PEETERS, M. RONCKEN, F. SCHALIJ, and R. van de WIEL. *A single-rail re-implementation of a DCC error detector using a standard-cell library*. Asynchronous Design Methodologies. IEEE Computer Society Press, 1995.

[11] K.V. BERKEL, M.B. JOSEPHS, and S.M. NOWICK. Scanning the technology : Applications of asynchronous circuits. *Proc. IEEE*, 87(2) :223–233, February 1999.

[12] E. BIHAM and A. SHAMIR. Differential cryptanalysis of the full 16-round DES. In *Advances in Cryptology CRYPTO'91 Proceedings*, pages 487–496. Springer Verlag, 1992.

[13] E. BIHAM and A. SHAMIR. Power analysis of the key scheduling of the AES candidates. In *Second AES Candidate Conference (AES2)*, Rome, Italy, 1999.

[14] D. BORRIONE, M. BOUBEKEUR, E. DUMITRESCU, M. RENAUDIN, J.B. RIGAUD, and A. SIRIANNI. An approach to the introduction of formal validation in an asynchronous circuit design flow. In *26th Annual Hawaï International Conference on System Sciences*, Big Island, Hawaï, 2003.

[15] F. BOUESSE. *Contribution à la conception de circuits intégrés sécurisés : l'alternative asynchrone*. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble, 2005.

[16] F. BOUESSE, M. RENAUDIN, B. ROBISSON, E. BEIGNÉ, P-Y. LIARDET, and S. PREVOSTO. DPA on quasi delay insensitive asynchronous circuits : concrete results. In *XIX Conference on Design of Circuits and Integrated Systems (DCIS04)*, Bordeaux, France, 2004.

[17] F. BOUESSE, G. SICARD, and M. RENAUDIN. Efficient quasi delay insensitive asynchronous architectures for low EMI. In *EMC_Compo*, Munich, Germany, 2005.

[18] J.G. BREDESON. Synthesis of multiple-input change hazard-free combinational switching circuits without feedback. *International Journal of Electronics (GB)*, 39(6) :615–624, December 1975.

[19] J.G. BREDESON and P.T. HULINA. Elimination of static and dynamic hazards for multiple input changes in combinational switching circuits. *Information and Control*, 20 :114–224, 1972.

[20] V. BREGIER. *Synthèse automatisée de circuits optimisés prouvés Quasi Insensibles aux Délais*. PhD thesis, INP Grenoble, 2007.

[21] V. BREGIER, B. FOLCO, L. FESQUET, and M. RENAUDIN. Modeling and synthesis of multi-rail multi-protocol QDI circuits. In *International Workshop on Logic Synthesis*, 2004.

[22] S.M. BURNS. General condition for the decomposition of state holding elements. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*. IEEE Computer Society Press, 1996.

[23] T.A. CHU. *Synthesis of self-timed VLSI circuits from Graph-theoretic specifications*. PhD thesis, Massachusetts Institute of Technology, 1987.

[24] W.A. CLARK. Macromodular computer systems. In *Spring Joint Computer Conference, AFIPS*, April 1967.

[25] C. CONSTANTINESCU. Trends and challenges in VLSI reliability. *IEEE micro*, 23(4) :14–99, July 2003.

[26] J. CORTADELLA, M. KISHINEVSKY, S.M. BURNS, A. KONDRATYEV, L. LAVAGNO, K.S. STEVENS, A. TAUBIN, and A. YAKOVLEV. Lazy transition systems and asynchronous circuit synthesis with relative timing assumptions. *IEEE Transactions on Computer-Aided Design*, 21(2) :109–130, 2002.

[27] J. CORTADELLA, M. KISHINEVSKY, A. KONDRATYEV, L. LAVAGNO, and A. YAKOVLEV. *Logic Synthesis of Asynchronous Controllers and Interfaces*. Springer-Verlag, 2002.

[28] Jordi CORTADELLA, M. KISHINEVSKY, A. KONDRATYEV, L. LAVAGNO, E. PASTOR, and A. YAKOVLEV. Decomposition and technology mapping of speed-independent circuits using boolean relations. In *Proc. International Conf. Computer-Aided Design (ICCAD)*, 1997.

[29] A.L. DAVIS. The architecture and system method of DDM-1 : A recursively-structured data driven machine. In *Fifth Annual Symposium on Computer Architecture*, 1978.

[30] G. DE MICHELI. *Synthesis and Optimisation of Digital Circuits*. McGrawHill Press, 1994.

[31] A.-V. DINH-DUC, L. FESQUET, and M. RENAUDIN. Synthesis of QDI asynchronous circuits from DTL-style petri nets. In *11th IEEE/ACM International Workshop on Logic and Synthesis*, 2002.

[32] A.V. DINH-DUC. *Synthèse automatique de circuits asynchrones QDI*. PhD thesis, INP of Grenoble, 2003.

[33] A.V. DINH DUC, J.B. RIGAUD, A. REZZAG, A. SIRIANNI, J. FRAGOSO, L. FESQUET, and M. RENAUDIN. TAST CAD tools. In *Asynchronous Circuit Design*, Munich, Germany, 2002.

[34] R. DRESCHLER and B. BECKER. *Binary Decision Diagrams, Theory and Implementation.* Kluwer Academic Publishers, kluwer academic publishers edition, 1998.

[35] E.B. EICHELBERGER. Hazard detection in combinational and sequential switching circuits. *IBM Journal of Research and Development*, 9 :90–99, March 1965.

[36] F.B. ENDECOTT and S.B. FURBER. Modelling and simulation of asynchronous system using the LARD hardware description language. In *Proc. of the 12th European Simulation Multiconference*, pages 39–43. Society for Computer Simulation International, 1994.

[37] K.M. FANT. *Logically Determined Design*. John Wiley & Sons, 2005.

[38] M. FERRETTI and P.A. BEEREL. Single-track asynchronous pipeline templates using 1-of-N encoding. In *DATE*, pages 1008–1015, Paris, France, 2002.

[39] M. FERRETTI and P.A. BEEREL. High performance asynchronous design using single-track full-buffer standard cells. *IEEE Journal of Solid-State Circuits*, 41(6) :1444–1454, 2006.

[40] M. FERRETTI, R.O. OZDAG, and P.A. BEEREL. High performance asynchronous ASIC back-end design flow using single-track full-buffer standard cells. In *The 10th IEEE International Symposium on Asynchronous Circuits and Systems*, pages 95–105, Hersonissos, Crete, 2004.

[41] B. FOLCO. Rapport de DEA : Génération automatique et optimisation de circuits QDI. Technical report, TIMA, Grenoble, 2003.

[42] B. FOLCO, V. BREGIER, L. FESQUET, and M. RENAUDIN. Technology mapping for area optimized quasi delay insensitive circuits. In *IFIP International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 146–151, Perth, Australia, 2005.

[43] R.M. FUHRER, S.M. NOWICK, M. THEOBALD, N.K. JHA, B. LIN, and L. PLANA. Minimalist : An environment for the synthesis, verification and testability of burst-mode asynchronous machines. Technical report, Columbia University, July 1999.

[44] H. Van GAGELDONK, D. BAUMANN, K. Van BERKEL, D. GLOOR, A. PEETERS, and G. STEGMANN. An asynchronous low-power 80c51 microcontroller. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits ans Systems (ASYNC)*, pages 96–107, 1998.

[45] F. GERMAIN. *Sécurité cryptographique par la conception spécifique de circuits intégrés*. PhD thesis, Ecole Polytechnique, 2006.

[46] S. GUILLEY. *Geometrical Counter-measures against Side-Channel Attacks*. PhD thesis, GET/Télécom Paris CNRS-LTCI (UMR 5141), 2006.

[47] S. GUILLEY, P. HOOGVORST, Y. MATHIEU, R. PACALET, and J. PROVOST. CMOS structures suitable for secured hardware. In *Design, Automation and Test in Europe Conference and Exhibtion*, 2004.

[48] S. HAUCK. Asynchronous design methodologies : An overview. *Proceedings of the IEEE*, 83(1) :69–93, January 1995.

[49] Q.T. HO, J.B. RIGAUD, L. FESQUET, M. RENAUDIN, and R. ROLLAND. Implementing asynchronous circuits on LUT based FPGAs. In Lecture Notes In Computer Science, editor, *Proceedings of the Reconfigurable Computing Is Going Mainstream, 12th International Conference on Field Programmable Logic and Applications*, volume 2438, pages 36–46, 2002.

[50] C.A.R. HOARE. Communicating sequential processes. *Communications of the ACM 21*, 8 :666–677, August 1978.

[51] C. JEONG and S.M. NOWICK. Optmial technology mapping and cell merger for asynchronous threshold networks. In *12th International Symposium on Asynchronous Circuits and Systems*, pages 128–137, 2006.

[52] J. KESSELS, T. KRAMER, G. den BESTEN, A. PEETERS, and V. TIMM. Applying asynchronous circuits in contactless smart card. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, pages 36–44. IEEE Computer Society Press, 2000.

[53] K. KEUTZER. DAGON : technology binding and local optimization by dag matching. In *Proceedings of the 24th ACM/IEEE conference on Design automation*, pages 341 – 347, Miami Beach, Florida, United States, 1987.

[54] P. KOCHER, J. JAFFE, and B. JUN. Differential power analysis. *Advances in Cryptology CRYPTO 1999, Lecture Notes in Computer Science*, 1666 :388–397, 1999.

[55] A. KONDRATYEV, J. CORTADELLA, M. KISHINEVSKY, L. LAVAGNO, and A. YAKOVLEV. Logic decomposition of speed-independent circuits. *Proceedings of the IEEE*, 87(2) :347–362, 1999.

[56] P. KUDVA, G. GOPALAKRISHNAN, H. JACOBSON, and S.M. NOWICK. Synthesis of hazard-free customized CMOS complex-gate networks under multiple-input changes. In *ACM/IEEE 33rd Design Automation Conference*, pages 77–82, 1996.

[57] Y. KUKIMOTO, R.K. BRAYTON, and P. SAWKAR. Delay-optimal technology mapping by DAG covering. In *Design Automation Conference (DAC)*, 1998.

[58] K.J. KULIKOWSKI, M. SU, A. SMIRNOV, A. TAUBIN, M.G. KARPOVSKY, and D. MACDONALD. Dealy insensitive encoding and power analysis : A balancing act. In *11th International Symposium on Asynchronous Circuits and Systems*, pages 116–125, 2005.

[59] M. LIGHTART, K.M. FANT, R. SMITH, A. TAUBIN, and A. KONDRATYEV. Asynchronous design using commercial HDL synthesis tool. In *Proc. of International Symposium on Asynchronous Circuits and Systems*, pages 114–125, 2000.

[60] F. MAILHOT and G. DE MICHELI. Technology mapping using boolean matching and don't care sets. In *Proceedings of the EDAC*, pages 212–216, 1990.

[61] S. MANGARD. A single power analysis (SPA) attack on implementations of the AES expansion. In *Information Security and Cryptology (ICISC'02), 6th International Conference*, pages 343–358, Seoul, Korea, 2002.

[62] A. MARTIN. Compiling communicating processes into delay-insensitive VLSI circuits. *Distributed Computing*, 1(4) :226–234, 1986.

[63] A. MARTIN. *Programming in VLSI : From communicating processes to delay-insensitive circuits*. Development in Concurrency and Communication. Addison-Wesley University of Texas At Austin Year of Programming Series, 1990.

[64] A. MARTIN, A. LINES, R. MANOHAR, M. NYSTROEM, P. PENZES, R. SOUTHWORTH, and U. CUMMINGS. The design of an asynchronous MIPS R3000 microprocessor. *Advanced Research in VLSI*, pages 164–181, 1997.

[65] A.J. MARTIN. The limitations to delay-insensitivity in asynchronous circuits. In William J. Dally, editor, *Advanced Research in VLSI*, pages 263–278. MIT Press, 1990.

[66] P. MAURINE, J.B. RIGAUD, F. BOUESSE, G. SICARD, and M. RENAUDIN. Static implementation of QDI asynchronous primitives. In *PATMOS : 13th International Workshop on Power and Timing Modeling, Optimization and Simulation*, volume 2799, pages 181–191, 2003.

[67] P. MAURINE, J.B. RIGAUD, F. BOUESSE, G. SICARD, and M. RENAUDIN. TAL : une bibliothèque de cellules pour le design de circuits asynchrones QDI. In *4iemes journées Francophones d'Etudes Faible Tension, Faible Consommation*, pages 41–49, 2003.

[68] R.E. MILLER. *Sequential Circuits and Machines*, volume 2 of *Switching Theory*. John Wiley & Sons, 1965.

[69] Y. MONNET. *Etude et modélisation de circuits résistants aux attaques non intrusives par injection de fautes*. PhD thesis, Institut National Polytechnique de Grenoble, Avril 2007.

[70] NIST. Data encryption standard (DES), December 1993.

[71] S.M. NOWICK. *Automatic Synthesis of Burst-Mode Asynchronous Controllers*. PhD thesis, Standford University, 1993.

[72] R.O. OZDAG and P.A. BEEREL. A channel based asynchronous low powerhigh performance standard-cell based sequential decoder implemented with QDI templates. In *10th International Symposium on Asynchronous Circuits and Systems*, pages 187–197, 2004.

[73] D. PANYASAK. *Réduction de l'Emission Electromagnétique des Circuits Intégrés : L'Alternative Asynchrone*. PhD thesis, Institut National Polytechnique de Grenoble, 2004.

[74] C.A. PETRI. *Kommunikation mit automaten*. PhD thesis, Institut für Instrumetelle Mathematik, 1962.

[75] J. QUARTANA. *Design of Asynchronous Network on Chip : application to GALS systems*. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble, 2004.

[76] A. RAZAFINDRAIBE. *Analyse et amélioration de la logique double-rail pour la conception de circuits sécurisés*. PhD thesis, Université Montpellier II, 2006.

[77] M. RENAUDIN. Asynchronous circuits and systems : a promising design alternative. *Microelectronic Engineering*, 54(1-2) :133–149, 2000.

[78] M. RENAUDIN and J.-B. RIGAUD. Etat de l'art sur la conception des circuits asynchrones : perspectives pour l'intégration des systèmes complexes. Technical report, TIMA, Grenoble, 1998.

[79] J.B. RIGAUD. *Spécification de bibliothèques pour la synthèse de circuits asynchrones*. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble, 2002.

[80] D. RIOS-ARAMBULA, A. BUHRIG, G. SICARD, and M. RENAUDIN. On the use of feedback systems to dynamically control the supply voltage of low-power circuits. *Journal of Low Power Electronics*, 2 :45–55, 2006.

[81] B. SCHNEIER. *Cryptographie appliquée*. Vuibert Informatique, 2e edition, 2001.

[82] P. SIEGEL, M. DE MICHELI, and D.L. DILL. Automatic technology mapping for generalized fundamental-mode asynchronous designs. In *3th international conference on Design automation*, pages 61–67, 1993.

[83] P.S.K. SIEGEL. *Automatic Technology Mapping for Asynchronous Designs*. PhD thesis, Stanford University, 1995.

[84] P.S.K. SIEGEL and G. DE MICHELI. Decomposition methods for library binding of speed-independent asynchronous designs. In *International Conference on Computer Aided Design*, pages 558–565, San Jose, CA, 1994.

[85] K. SLIMANI. *Une méthodologie de conception de circuits asynchrones à faible consommation d'énergie : application au microprocesseur MIPS*. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble, 2004.

[86] J. SPARSO, J. STAUNSTRUP, and M. DANTZER-SORENSEN. Design of delay insensitive circuits using multi-ring structures. In *Proc. European Design Automation Conference (EURO-DAC)*, pages 15–20, Hamburg, Germany, 1992. IEEE Computer Society Press.

[87] D. STINSON. *Cryptographie : Théorie et Pratique*. International Thomson Publishing France, 1996.

[88] I. SUTHERLAND, B. SPROULL, and D. HARRIS. *Logical Effort : Designing Fast CMOS Circuits*. Morgan Kaufmann Publishers, INC., San Francisco, California, 1999.

[89] I.E. SUTHERLAND. Micropipelines. *Communication of the ACM*, 32(6), June 1989.

[90] K. TIRI, A. MOONMOON, and I. VERBAUWHEDE. A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In *28th European Solid-State Circuits Conference*, 2002.

[91] J.T. UDDING. A formal model for defining and classifying delay-insensitive circuits. *Distributed Computing*, 1(4) :197–204, 1986.

[92] S.H. UNGER. *Asynchronous Sequential Switching Circuits*. Wiley-Interscience. John Wiley & Sons, New York, 1969.

[93] J.F. WAKERLY. *Digital Design : Principles and Pratices*. Prentice, 2 edition, 1994.

[94] A. YAKOVLEV, L. GOMES, and L. LAVAGNO. *Hardware Design and Petri Nets*. Kluwer Academic Publisher, 2000.

[95] T. YONEDA, O. HIROOMI, and C.J. MYERS. Synthesis of speed independent circuits based on decomposition. In *The 10th IEEE International Symposium on Asynchronous Circuits and Systems*, pages 135–145, Hersonissos, Crete, 2004.

[96] M. ZHAO and S.S. SAPATNEKAR. A new structural pattern matching algorithm for technology mapping. In *The 38th Conference on Design Automation*, pages 371–376, Las Vegas, Nevada, United States, 2001.

Bibliographie de l'auteur

Chapitres de livre

– B. FOLCO, V. BRÉGIER, L. FESQUET and M. RENAUDIN. Technology Mapping for Area Optimized Quasi Delay Insensitive Circuits. In *IFIP International Federation for Information Processing*, Vol. 240, pages 55-69. M. Glesner, R. Reis, L. Indrusiak, V. Mooney, H. Eveking (Eds.), Springer, 2007.

Conférences internationales avec comité de lecture

– V. BREGIER, B. FOLCO, L. FESQUET and M. RENAUDIN. Modeling and synthesis of multi-rail multi-protocol QDI circuits. In *Proc. IWLS*, June 2004.

– A. RAZAFINDRAIBE, P. MAURINE, M. ROBERT, F. BOUESSE, B. FOLCO, M. RENAUDIN. Secured Structures for Secured Asynchronous QDI circuits. *XIX Conference on Design of Circuits and Integrated Systems*, November 2004.

– B. FOLCO, V. BRÉGIER, L. FESQUET and M. RENAUDIN. Technology Mapping for Area Optimized Quasi Delay Insensitive Circuits. VLSI Soc 2005, Perth, Australia.

– L. FESQUET, B. FOLCO, M. STEINER, M. RENAUDIN. State-holding in Look-Up Tables : application to asynchronous logic, *VLSI Soc*, 2006, Nice, France.

Conférences nationales

– B. FOLCO, M. RENAUDIN. Synthèse et projection technologique de circuits asynchrones Quasi-Insensibles aux Délais. In *8ème Journées Nationales du Réseau Doctoral de Microélectronique*, Paris, France, May 2005.

– D. RIOS-ARAMBULA, B. FOLCO, Y. MONNET, M. RENAUDIN. Analyse du profil de consommation des circuits asynchrones QDI, In *Proceedings FTFC* 2007, Paris, France.

CONTRIBUTION À LA SYNTHÈSE DE CIRCUITS ASYNCHRONES QUASI INSENSIBLES AUX DÉLAIS, APPLICATION AUX SYSTÈMES SÉCURISÉS

Résumé : Les travaux présentés dans cette thèse portent sur le développement dune méthodologie de conception de circuits asynchrones Quasi Insensibles aux Délais (QDI) et son application à des circuits sécurisés. Contrairement aux circuits synchrones, les circuits asynchrones se caractérisent par labsence de signal dhorloge. Ces circuits sont séquencés par un mécanisme de communication et de synchronisation local. En plus des nombreuses propriétés des circuits asynchrones telles que la robustesse, une faible consommation, un faible bruit et une excellente modularité, les propriétés de la logique QDI apparaissent également particulièrement intéressantes pour sécuriser limplantation des circuits intégrés contre les attaques par analyse de courant. Cependant, le manque de méthode et doutil de conception est un frein à leur adoption. Cest dans ce contexte que se situe ce travail de thèse, qui contribue au développement dun outil de conception de circuits asynchrones développé au laboratoire TIMA : TAST.

Mots clés : Circuits asynchrones, circuits quasi insensibles aux délais, projection technologique, bibliothèque de cellules asynchrones, méthodologie de conception, attaques matérielles, attaques en puissance.

CONTRIBUTION TO THE SYNTHESIS OF QUASI DELAY INSENSITIVE ASYNCHRONOUS CIRCUITS, APPLICATION TO SECURED SYSTEMS.

Abstract : The work presented in this thesis deals with the development of a design methodology for Quasi Delay Insensitive (QDI) circuits and its application to secured circuits. Contrary to synchronous circuits, asynchronous circuits are characterized by the absence of a global clock signal. These circuits are sequenced by a local mechanism of communication and synchronization. In addition to many properties of the asynchronous circuits such as the robustness, low consumption, low noise and excellent modularity, the properties of the QDI logic appear also particularly interesting to protect integrated circuits against attacks based on current analysis. However, the lack of methods and tools is a brake for their adoption in the industry. In this context, this thesis contributes to the development of a tool for the design of asynchronous circuits developed at TIMA laboratory called TAST.

Keywords : Asynchronous circuits, quasi delay insensitive circuits, technology mapping, asynchronous cells library, methodology of conception, hardware attacks, power attacks.

Intitulé et adresse du laboratoire : Laboratoire TIMA
46 avenue Félix Viallet, 38031, Grenoble Cedex, France.
ISBN : 978-2-84813-108-5