Conception Automatique de Chemins de Données en Logique Asynchrone QDI

J. Fragoso

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

N° attribué par la bibliothèque
|__/__/__/__/__/__/__/__/__/__|

THESE

pour obtenir le grade de

DOCTEUR DE l'INPG

Spécialité : Micro Nano Électronique

préparée au laboratoire **TIMA** dans le cadre de
l'Ecole Doctorale d'« Electronique, Electrotechnique, Automatique, Télécommunications, Signal »

présentée et soutenue publiquement

par
João Leonardo FRAGOSO

Le 16 novembre 2005

Titre :

CONCEPTION AUTOMATIQUE DE CHEMINS DE DONNEES EN LOGIQUE ASYNCHRONE QDI

Directeur de Thèse : Marc RENAUDIN
Codirecteur : Gilles SICARD

JURY

M. Olivier SENTIEYS	, Président, Rapporteur
M. Christian PIGUET	, Rapporteur
M. Marc RENAUDIN	, Directeur de thèse
M. Gilles SICARD	, Co-directeur de thèse
M. Ricardo REIS	, Examinateur

Résumé

Ces dernières années, les circuits asynchrones sont apparus comme une solution naturelle aux problèmes de conception des circuits synchrones lies aux technologies submicroniques. En s'affranchissant d'une horloge globale et en utilisant un mécanisme de synchronisation locale, les circuits asynchrones se montrent plus fiables, robustes et modulaires que leurs équivalents synchrones. En plus, l'absence de horloge globale permet d'adresser des contraintes de faible consommation, faible bruit et sécurité. Cependant, l'intérêt croissant dans les circuits asynchrones se heurte au manque actuel de méthodes et outils d'aide à la conception de tels circuits.
Dans ce cadre, ce travail de thèse porte sur l'étude de la conception de chemins de données asynchrones QDI (de l'anglais, « quasi-delay insensitive »). Initialement, cette thèse propose et évalue une méthode de comparaison de différentes implémentations des circuits asynchrones. Par la suite, les deux principaux opérateurs arithmétiques sont étudiés : les additionneurs et les multiplieurs. Dans cette étude, plusieurs architectures ont été évaluées et l'impact de différents codages de données ont été examinés. La méthode de comparaison et la génération d'opérateurs arithmétiques ont été automatisées de façon à permettre aux concepteurs de circuits de choisir l'implémentation plus adéquate aux contraintes de conception.
L'expertise obtenue par l'étude d'opérateurs arithmétiques a aussi permis de généraliser certaines recommandations à la conception de tous les chemins de données asynchrones. Ces recommandations sont à l'origine d'une méthodologie de conception de chemins de données asynchrones. Les résultats de ce travail enrichissent l'outil de conception qu'aide à combler l'espace entre les concepteurs et les circuits asynchrones.

Mots clés : circuits asynchrones, chemins de données, opérateurs arithmétiques, additionneurs, multiplieurs, CAO, synthèse automatique.

"DATA PATHS AUTOMATIC DESIGN IN QDI ASYNCHRONOUS LOGIC"

Abstract

In recent years, asynchronous circuits have become a natural solution for synchronous circuit design problems related to the use of submicron technologies. By using a local synchronization mechanism instead of a global clock, asynchronous circuits are more reliable, robust and modular than their equivalent synchronous form. Additionally, by not using a global clock, design constrains such as low-power consumption, low noise and security may be addressed. However, the ever-increasing interest in asynchronous circuits faces an important drawback, which is the lack of methods and tools that help designing such circuits.
In this context, this thesis focuses on designing QDI (quasi-delay insensitive) asynchronous data paths. Initially, this thesis proposes and evaluates a method for comparing different asynchronous circuit implementations. Next, the two main arithmetic operators, which consist in adders and multipliers, are studied. In this study, not only several architectures were evaluated but also the impact of different data coding was analyzed. The comparison method and the arithmetic operator generation were automated, allowing designers to choose an implementation, which fits best the design constraints.
The expertise earned by studying arithmetic operators has also led to the generalization of some guidelines concerning the design of all asynchronous data path. These guidelines are the result of a methodology for designing asynchronous data path. The results of this work improve the design tool, helping to reduce the gap between designers and asynchronous circuits.

Keywords: asynchronous circuits, data-paths, arithmetic operators, adders, multipliers, CAD, automatic synthesis.

ISBN : 2-84813-076-8
ISBNE : 2-84813-076-8

À Mariana

Pour ses bras je ne serais jamais
un étrange étranger.

Résumé

Ces dernières années, les circuits asynchrones sont apparus comme une solution naturelle aux problèmes de conception des circuits synchrones lies aux technologies submicroniques. En s'affranchissant d'une horloge globale et en utilisant un mécanisme de synchronisation locale, les circuits asynchrones se montrent plus fiables, robustes et modulaires que leurs équivalents synchrones. En plus, l'absence de horloge globale permet d'adresser des contraintes de faible consommation, faible bruit et sécurité. Cependant, l'intérêt croissant dans les circuits asynchrones se heurte au manque actuel de méthodes et outils d'aide à la conception de tels circuits.

Dans ce cadre, ce travail de thèse porte sur l'étude de la conception de chemins de données asynchrones QDI (de l'anglais, « quasi-delay insensitive »). Initialement, cette thèse propose et évalue une méthode de comparaison de différentes implémentations des circuits asynchrones. Par la suite, les deux principaux opérateurs arithmétiques sont étudiés : les additionneurs et les multiplieurs. Dans cette étude, plusieurs architectures ont été évaluées et l'impact de différents codages de données ont été examinés. La méthode de comparaison et la génération d'opérateurs arithmétiques ont été automatisées de façon à permettre aux concepteurs de circuits de choisir l'implémentation plus adéquate aux contraintes de conception.

L'expertise obtenue par l'étude d'opérateurs arithmétiques a aussi permis de généraliser certaines recommandations à la conception de tous les chemins de données asynchrones. Ces recommandations sont à l'origine d'une méthodologie de conception de chemins de données asynchrones. Les résultats de ce travail enrichissent l'outil de conception TAST (de l'anglais « TIMA Asynchronous Synthesis Tool ») qu'aide à combler l'espace entre les concepteurs et les circuits asynchrones.

Mots-clés : circuits asynchrones, chemins de données, opérateurs arithmétiques, additionneurs, multiplieurs, CAO, synthèse automatique.

DATA PATHS AUTOMATIC DESIGN IN QDI ASYNCHRONOUS LOGIC

Abstract

In recent years, asynchronous circuits have become a natural solution for synchronous circuit design problems related to the use of submicron technologies. By using a local synchronization mechanism instead of a global clock, asynchronous circuits are more reliable, robust and modular than their equivalent synchronous form. Additionally, by not using a global clock, design constrains such as low-power consumption, low noise and security may be addressed. However, the ever-increasing interest in asynchronous circuits faces an important drawback, which is the lack of methods and tools that help designing such circuits.

In this context, this thesis focuses on designing QDI (quasi-delay insensitive) asynchronous data paths. Initially, this thesis proposes and evaluates a method for comparing different asynchronous circuit implementations. Next, the two main arithmetic operators, which consist in adders and multipliers, are studied. In this study, not only several architectures were evaluated but also the impact of different data coding was analyzed. The comparison method and the arithmetic operator generation were automated, allowing designers to choose an implementation, which fits best the design constraints.

The expertise earned by studying arithmetic operators has also led to the generalization of some guidelines concerning the design of all asynchronous data path. These guidelines are the result of a methodology for designing asynchronous data path. The results of this work improve the design tool, helping to reduce the gap between designers and asynchronous circuits.

Keywords: asynchronous circuits, data paths, arithmetic operators, adders, multipliers, CAD, automatic synthesis.

Remerciements

Ces travaux de thèse ont été financés par l'agence CAPES (« Coordenação de Aperfeiçoamento de Pessoal de Nível Superior ») du Ministère de l'Éducation du gouvernement brésilien. Donc, je voudrais tout d'abord remercier le gouvernement brésilien pour le soutien financier apporte à cette thèse.

Cette thèse a été réalisée au sein du laboratoire des Techniques de l'Informatique et de la Microélectronique pour l'Architecture des ordinateurs (TIMA), sur le site Viallet de l'Institut National Polytechnique de Grenoble (INPG). Je remercie son directeur, Bernard Courtois, pour son accueil au sein de ce laboratoire.

Je remercie Marc Renaudin, mon directeur de thèse, Professeur à l'École Nationale Supérieure d'Électronique et de Radioélectricité de Grenoble (ENSERG), et responsable du groupe « Concurrent Integrated Systems » (CIS), de m'avoir accepté au sein de son groupe de recherche, de m'avoir accordé sa confiance et de m'avoir fait partagé sa passion pour les circuits asynchrones. Ces quatre années ont été très enrichissantes sur le plan technique et sur le plan humain.

Je remercie également Gilles Sicard, mon co-directeur de thèse, Maître de Conférence à l'Université Joseph Fourier (UJF), pour sa disponibilité, son esprit critique, et ses éclairages techniques pendant ces quatre années.

Je remercie Olivier Sentieys, Professeur à l'École Nationale Supérieure des Sciences Appliquées et de Technologie (ENSSAT) de Lannion, et Christian Piguet, Professeur à l'École Polytechnique Fédérale de Lausanne (EPFL) en Suisse, d'avoir accepté d'être les rapporteurs de mes travaux de thèse. Je remercie également Ricardo Reis, Professeur à l'Université Fédérale du « Rio Grande do Sul » (UFRGS), d'avoir participé à mon jury de thèse.

Je remercie Alexandre Chagoya pour son extrême gentillesse, Bernard Bosc, Robin Rolland et plus généralement tout le personnel du CIME. Je remercie également tout le personnel du TIMA et du CMP pour les formidables années passées ici.

J'adresse mes sincères remerciements à l'ensemble des membres du groupe CIS pour leur bonne humeur et leurs « compétences » dans de si nombreux « domaines » qui m'ont été utiles au cours de cette thèse. Je voudrais remercie tout particulièrement: Manu (le roi de la guitare et « je m'excuse d'avance »), Fabien (qu'a le mérite d'avoir rédigé une thèse tout en partageant le bureau avec moi), Jérôme et Pascale (pour leur amitié et nos innombrables sorties de ski), Dhanistha (la éternelle mascotte du groupe CIS et ma professoresse de français), Yann (l'homme à la banane en cuir et avec plus d'option que GCC par ligne de commande), Fraidy (le cryptologe à la grosse banane), Salim (l'homme sans banane), Estelle et Christophe (la toujours future mascotte du groupe CIS et son mari), Kamel (le playboy de la bibliothèque municipale de Grenoble), Amine (mon « maître nageur »), Jean-Baptiste (l'ancienne mascotte du groupe CIS au mauvais caractère, roi de la porte de Muller, et dernier fan de Michel Sardou), Gregory (le testeur officiel de fixations de ski), Sébastien C. (un autre « maître nageur »), Sébastien F. & Frédéric (les gardiens du réseau informatique), Patrick (le roi de la Chartreuse sur les pistes enneigées), Ahmed (le bassiste du groupe TIMA).

Je voudrais aussi remercie tous mes amis avec le quelles j'ai eu le « plaisir » de partager l'aventure de cette thèse : Renaud et Virginie (toujours disponible pour garder Mariana pendant la rédaction du manuscrit), Stephane R. (« on peut descendre à Montpellier ? »), Benoît et Mari-Ange (le couple « Vietnamien »), Stephane et Christelle (pour les soirées au « basse terre » et en Bretagne), Thomas et Céline (pour les visites à Le Pont-de-Claix), Wander et Shirley (pour son aide à l'intégration en France), Adriano et Andrea (le meilleur accent brésilien) et à tous les autres que j'ai certainement oublié de mentionner, mais qui sauront me pardonner.

Je voudrais remercie toute ma famille : mon père Antonio, ma mère Maria Vanda, mes sœurs Valeska et Variluska, et ma belle famille : Leila, Gilberto, Liana, Cecília, Maria et Geremias. Tous ont su partager et alléger le déchirement de la distance.

Finalement, je voudrais remercier ma femme Mariana qu'a accepté de faire « le voyage » et d'être toute ma famille pendant ces quatre années de thèse. Sans son soutien et son amour, cette thèse n'aurait jamais vu le jour…

Table de Matières

Liste de Figures

Chapitre VI Additionneurs

Chapitre VII Multiplieurs

Liste de Tableaux

Lexique

CAO	Conception Aidé par Ordinateur
CHP	Communicating Hardware Processes
CIS	Concurrent Integrated Systems
DTL	Data Transfer Level
FPGA	Field Programmable Gate Array
GALS	Globally Asynchronous Locally Synchronous
GSLA	Globally Synchronous Locally Asynchronous
QDI	Quasi Delay-Insensitive
RTL	Register Transfer Level
SI	Speed-Independent
SoC	System on Chip
TAST	TIMA Asynchronous Synthesis Tool
TIMA	Techniques of Informatics and Microelectronics for Computer Architecture
VHDL	VLSI Hardware Description Language
VLSI	Very Large Scale Integration

Chapitre I

Introduction

Les circuits asynchrones ne sont pas étranges pour les concepteurs des circuits intégrés numériques vu que les premiers travaux concernant ces circuits datent de la fin des années 50. Cependant, à cette époque, les circuits synchrones, avec une horloge globale synchronisant les actions, offraient une simplification de la conception face aux circuits asynchrones. Pour cette raison les circuits synchrones se sont popularisés au point que circuit intégré numérique est devenu synonyme de la présence d'une horloge globale. Dû à cette popularité, les efforts de recherche et développement se sont consacrés à l'élaboration des outils et des environnements à l'aide de la conception de circuits synchrones.

En contre partie, la complexité croissante des circuits intégrés sur une même puce confronte la conception de circuits synchrones à des problèmes majeurs liés directement à l'existence d'une horloge globale. La réduction du bruit et de la consommation d'énergie et la réutilisation de blocs déjà conçus sont rendus difficiles par la présence d'une horloge globale et la propre distribution du signal d'horloge dans une puce est devenue un problème dont les contraintes sont telles qu'elle doit être réalisée exclusivement par des spécialistes et des outils développés pour cette tâche. Les circuits asynchrones représentent donc la solution naturelle aux inconvénients d'une horloge globale vu que les circuits asynchrones ne sont pas synchronisés par une horloge mais localement par un protocole d'échange d'informations.

Nonobstant les potentialités et les avantages inhérents aux circuits asynchrones démontrés par plusieurs prototypes conçus dans le cadre académique, divers facteurs expliquent le petit nombre d'applications commerciales asynchrones qui actuellement ont vu le jour. Entre ces facteurs, le plus préoccupant est le manque d'outils, méthodologies et

environnements d'aide à la conception des circuits asynchrones. Malgré les efforts que la communauté asynchrone a fait ces dernières décennies, la conception asynchrone est encore aujourd'hui l'affaire de spécialistes travaillant avec des outils faiblement automatisés. En plus, la conception numérique synchrone est arrivée à un niveau de maturité et d'automatisation qui n'est pas encore atteint par les outils asynchrones et l'industrie n'est pas disponible à un changement qui risquera d'augmenter son temps d'accès au marché. En conséquence, une large diffusion et adoption des circuits asynchrones passe par le développement et la mise à disposition d'outils aussi fiables et performants que les outils synchrones actuellement existants.

Le laboratoire TIMA et, en particulier, le groupe CIS (pour « *Concurrent Integrated Systems* ») contribuent à la recherche asynchrone en particulier par le développement de l'environnement TAST, acronyme de « TIMA Asynchronous Synthesis Tools ». L'environnement TAST, présenté par le flot de conception de la Figure I.1, vise à offrir des outils simples d'utilisation pour la conception de circuits intégrées asynchrones.

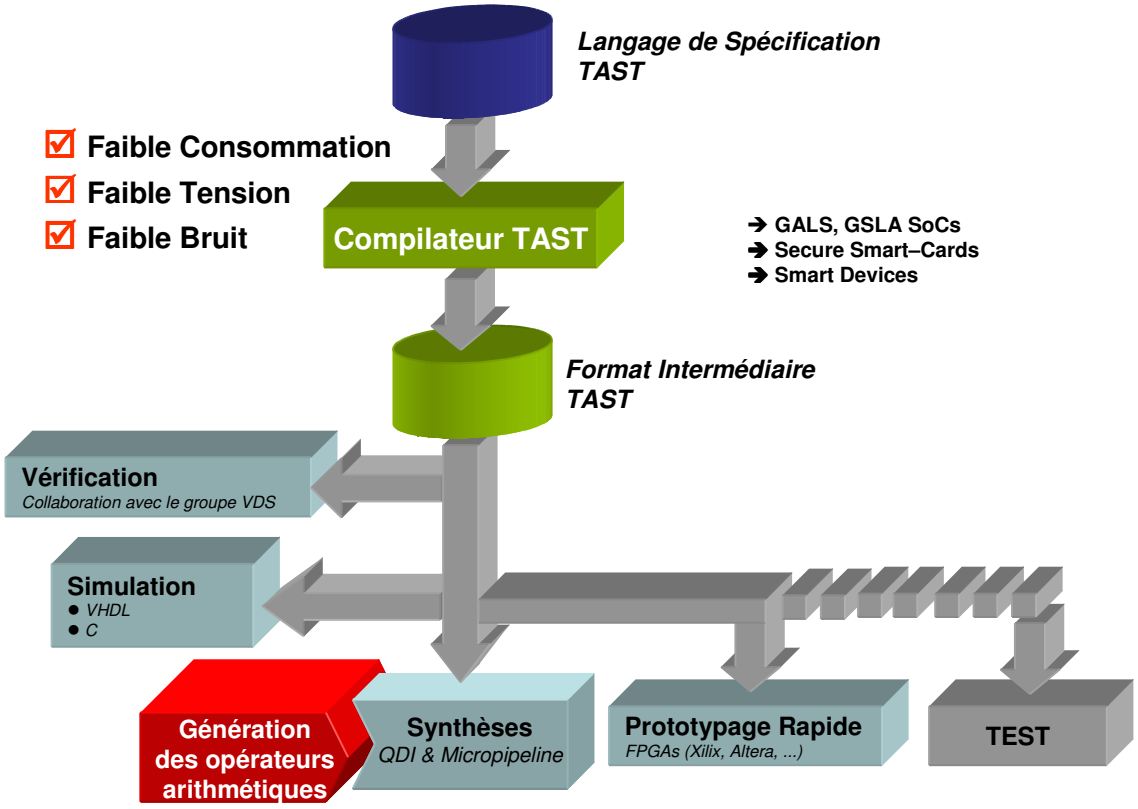


Figure I.1 – Flot de conception TAST.

L'environnement TAST est un ensemble d'outils capables de cibler plusieurs styles de circuits asynchrones en partant des langages de description de haut niveau. Le compilateur

TAST produit un format intermédiaire de description des circuits asynchrones à partir d'une spécification du circuit en langage CHP. L'utilisation d'autres langages de description de circuits, comme par exemple SystemC, sont en cours d'études au sein du groupe CIS.

Le format intermédiaire de TAST est basé sur les réseaux de Petri et les graphes de flot de données. À partir de ce format, plusieurs cibles sont possibles : (1) la génération d'une description comportementale VHDL ou C pour une simulation rapide sur des outils standard ; (2) la vérification formelle de la description ; (3) le prototypage rapide sur des FPGAs ; (4) la synthèse des circuits par la génération de descriptions matérielles au niveau portes logiques en VHDL ; et, finalement, (5) l'automatisation de la génération des vecteurs de test (en cours).

Dans le processus de synthèses deux classes de circuits asynchrones sont ciblées : la classe des circuits quasiment insensibles aux délais (QDI) et les circuits micropipeline. Le résultat de la synthèse peut, par la suite, être utilisé par les outils standard commerciaux pour effectuer les dernières étapes de la conception, comme par exemple la génération du *layout* final du circuit.

Dans ce contexte, cette thèse contribue au flot principal de l'environnement TAST, plus spécialement à la synthèse des circuits QDI. Cette thèse fait une étude de la conception automatique de chemins de données asynchrones en logique QDI. Les chemins de données de circuits micropipeline sont similaires aux circuits synchrones et leur synthèse peut être réalisée par les outils standard utilisés dans la conception de circuits synchrones. Ainsi, seulement les circuits QDI avec différents codages de données sont abordés au cours de cette thèse.

La première contribution de cette thèse est l'élaboration d'une métrique permettant la comparaison de différentes implémentations de circuits QDI au niveau des portes logiques. Cette métrique, qui permet de comparer surface, retard et consommation d'énergie, peut être utilisée au moment de la synthèse des circuits pour évaluer rapidement la performance des différentes possibilités d'implémentation.

Par la suite, cette thèse effectue une comparaison de plusieurs styles de circuits asynchrones actuellement existants et propose un ensemble de recommandations qui seront intégrés aux versions futures de l'outil de synthèse TAST. Les résultats de cette comparaison permettront d'améliorer les résultats de synthèse des circuits QDI et pourront guider les concepteurs dans l'exploitation architecturale. Cette thèse propose aussi une méthode pour la décomposition de portes logiques à plusieurs entrées de façon à limiter l'entrance maximale des portes logiques et en même temps, de réduire la consommation d'énergie et d'augmenter la vitesse d'arbres de portes logiques.

Comme tout les outils de conception de circuits, l'environnement TAST a besoin de modules dédiés à la génération de circuits spécifiques, comme par exemples les opérateurs arithmétiques. Le développement de modules de génération d'opérateurs arithmétiques est extrêmement important pour la conception, puisque ces opérateurs sont largement utilisés dans le flot de conception de circuits numériques. De plus, ces circuits, de part leur régularité de construction et quelques propriétés mathématiques, peuvent être optimisés ou modifiés pour améliorer soit l'occupation en surface, soit la performance, soit la consommation. Les résultats obtenus par des générateurs spécifiques sont difficilement atteints par les outils de synthèse généraux.

Ainsi, l'autre contribution de cette thèse porte sur la généralisation et l'automatisation de la conception de plusieurs architectures d'additionneurs et multiplieurs QDI utilisant différents codages de données. Un module de génération automatique de ces opérateurs a été développé et intégré au flot de conception TAST. Ainsi, le synthétiseur de circuits QDI en présence d'un opérateur arithmétique fait appel à ce module pour la conception de ces opérateurs. Le module de génération automatique peut aussi être utilisé indépendamment par le concepteur.

I.1. Plan du Manuscrit

Ce manuscrit est organisé comme suit.

Tout d'abord, le chapitre II présente les principes des circuits asynchrones, leurs avantages et aussi leurs inconvénients par rapport aux actuels circuits synchrones. L'anatomie des circuits synchrones est dévoilée et les caractéristiques principales des circuits asynchrones sont abordées. En absence d'une horloge globale, la synchronisation est réalisée localement. Ainsi, les protocoles de communication permettant l'échange d'information entre plusieurs blocs d'un circuit asynchrone sont présentés. Ce chapitre présente aussi une discussion sur les différents codages de données qui permettent l'implémentation des protocoles de communication et la synchronisation entre différentes parties d'un circuit.

La robustesse d'un circuit asynchrone est directement dépendante des hypothèses temporelles faites au moment de la conception. Ces hypothèses permettent de classifier les différents circuits. Cette classification et l'impact sur la robustesse des circuits sont aussi présentés au cours du chapitre II de cette thèse.

Le troisième chapitre présente avec plus des détails le flot de conception TAST et le langage CHP actuellement utilisé par le compilateur. Le chapitre III permet de mieux situer le contexte des principales motivations de cette thèse,

Le chapitre IV présente tout d'abord l'étude et l'élaboration d'un modèle de surface, retard et consommation d'énergie qui permet de comparer de façon équitable plusieurs circuits asynchrones. Ce modèle est employé pour l'évaluation de circuits asynchrones décrits au niveau des portes logiques. Le modèle développé est indépendant de la technologie cible, mais il intègre plusieurs facteurs liés exclusivement à des technologies CMOS. Ainsi, même si le modèle reste indépendant de la technologie, il est possible d'ajuster facilement le modèle pour refléter des changements technologiques et/ou pour améliorer les résultats par rapport à une technologie spécifique. Ensuite, le chapitre IV propose une évaluation du modèle élaboré par l'étude de cas du dimensionnement de portes logiques et une métrique est proposée visant à intégrer le retard et la consommation d'énergie dans une même mesure de performance des circuits asynchrones.

Le chapitre V présente l'étude comparative des styles de circuits asynchrones QDI existants. Cette étude définit l'intérêt de chaque style de circuit QDI et dans quelles conditions ces styles peuvent être employés et optimisés. Le résultat de cette étude est un ensemble de recommandations qui, ultérieurement, serviront à améliorer les outils de synthèses de l'environnement TAST. Le chapitre V présente aussi l'algorithme de décomposition des portes logiques développé au cours de cette thèse. L'algorithme proposé décompose les portes logiques à plusieurs entrées visant à limiter l'entrance maximale de chaque porte, mais l'algorithme fournit aussi une solution qui réduit la consommation d'énergie et qui augmente la vitesse des arbres de portes logiques face aux arbres équilibrés qui sont normalement utilisés par les concepteurs de circuits intégrées.

Le sixième chapitre de cette thèse est relatif à l'étude de la conception automatique des additionneurs QDI utilisant le codage *1-parmi-N*. Les additionneurs étudiés ont été généralisés pour tout l'ensemble des codages quasiment insensibles aux délais existants dans le langage CHP. Les cellules de base des différents additionneurs ont été développées permettant la conception des plusieurs structures d'additionneurs. La génération de ces additionneurs a été aussi automatisée permettant l'exploitation architecturale par le concepteur qui peut maintenant concevoir et évaluer rapidement plusieurs structures d'additionneurs. Au cours de cette thèse, plusieurs architectures pour l'addition ont été évaluées et les résultats présentés dans le chapitre

VI montrent l'intérêt de mettre à la disposition des concepteurs différentes formes de codage de données dans les langages de description de circuits asynchrones.

Le chapitre VII réalise une étude similaire pour les multiplieurs. L'algorithme de multiplication et la génération de certains multiplieurs ont aussi été automatisés. L'outil de génération des opérateurs arithmétiques – addition et multiplication – a été intégré au flot de conception TAST et le synthétiseur QDI fait appel à cet outil au moment de la synthèse de circuits contenant ces opérateurs.

Finalement, le dernier chapitre présente un bilan du travail réalisé et dresse les conclusions obtenues à partir de l'élaboration de cette thèse. Ce chapitre identifie aussi les possibles travaux futurs et les perspectives de cette thèse.

Chapitre II

Principes des Circuits Asynchrones

Les systèmes numériques synchrones sont composés des blocs logiques combinatoires interposés par des registres (bascules) comme montre la Figure II.1. L'activité du circuit est contrôlée par une horloge globale qui effectue la mémorisation de l'état actuel du circuit dans les registres. Dès qu'un état est échantillonné et donc mémorisé dans les bascules, les circuits combinatoires commencent le calcul du prochain état qui sera échantillonné au prochain coups d'horloge. Cette construction est conforme avec le formalisme de spécification RTL (de l'anglais, « register transfer level ») utilisé par la plupart des concepteurs et outils actuels.

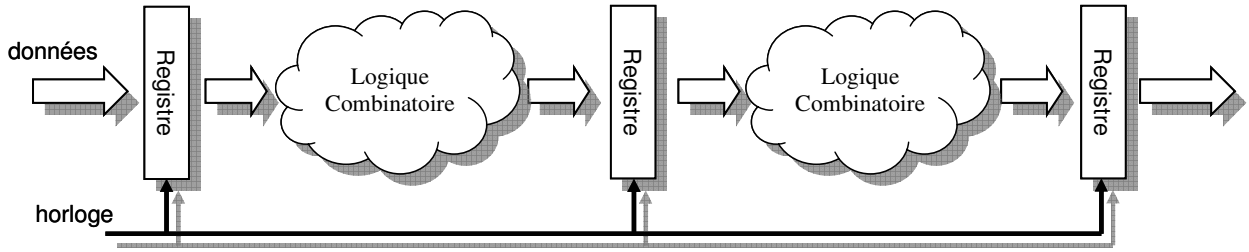


Figure II.1 – Structure de base de systèmes synchrones.

L'adoption de tel modèle entraîne trois conséquences importantes :

- ➔ **simplification de la conception** de blocs logiques vu que le concepteur peut ignorer les aléas logiques et les fluctuations électriques transitoires ;
- ➔ le **mécanisme de communication** entre les différents blocs est trivial ;
- ➔ il y a **une seule contrainte à respecter** pour assurer le bon fonctionnement du circuit : le chemin critique n'excède pas la période d'horloge.

Grâce à la simplicité de réalisation du contrôle et de la logique, les circuits synchrones dominent l'industrie de microélectronique depuis des années. Cependant, avec la miniaturisation des composants, l'augmentation continue du niveau d'intégration de systèmes de plus en plus complexes sûr une même puce et la demande de systèmes de plus en plus performants, la conception de circuits synchrones se confronte à des problèmes inhérents à leur architecture :

- ➔ **la performance est limitée par le bloc le plus lent** ; même si certains composants peuvent réaliser leur traitement dans un délai inférieur à la période d'horloge ou même si le chemin critique est rarement emprunté;
- ➔ **la distribution du signal d'horloge** : si le signal d'horloge n'est pas uniformément distribué, les différences de temps de propagation du signal (de l'anglais « skew ») peuvent causer un disfonctionnement du circuit. Ainsi dans les actuels systèmes synchrones, la distribution du signal d'horloge est un problème à part entière;
- ➔ **la consommation d'énergie** : dans les circuits synchrones, l'occurrence des coups d'horloge est source d'une importante consommation d'énergie qui croît rapidement avec l'augmentation de la fréquence du signal. Enfin, même si un système synchrone ne traite pas des nouvelles données, la présence du signal d'horloge continue à provoquer une consommation du circuit ;
- ➔ **la modularité** est un des principaux problèmes d'intégration des systèmes synchrones. Il peut s'avérer difficile de composer un système complexe en utilisant différents blocs logiques qui ont été conçus avec des fréquences d'horloge distinctes. La modularité est un aspect important pour la conception de systèmes complexes et pour la réutilisation de blocs déjà conçus ;
- ➔ les bascules échantillonnant des signaux asynchrones (notamment dans les interfaces des circuits synchrones) peuvent rester dans un état métastable. la **métastabilité** implique que le signal à la sortie du bascule restera indéterminé pour un temps non prévisible [II.1] et, donc, un temps suffisant long (période d'horloge) pour provoquer le mauvais fonctionnement du circuit ;
- ➔ **les émissions électromagnétiques** sont concentrées dans les harmoniques de la fréquence d'horloge vu que la plus forte activité, et donc les pics de courant, des circuits synchrones se produisent au moment de la mémorisation de l'état du circuit.

Confronté aux problèmes des circuits synchrones, la conception des circuits asynchrones apparaît comme la solution naturelle. En effet, l'horloge globale des circuits synchrones est remplacée par un contrôle local distribué dans les circuits asynchrones. Ainsi, la suppression de l'horloge permet de s'affranchir des problèmes de l'architecture synchrone et la conception asynchrone a gagnée en popularité ces dernières années.

II.1. Anatomie des Circuits Asynchrones

Les circuits asynchrones sont composés de blocs communicants et le fonctionnement des blocs est contrôlée par la présence de données dans les entrées et sorties de chaque bloc. Leur fonctionnement est similaire à celui des systèmes « flot de données ». En considérant un module asynchrone de base, comme le montre la Figure II.2, le module reçoit les données par son canal d'entrée, il les traite et il envoie le résultat par son canal de sortie. La communication n'est pas régie par un signal externe – comme un signal d'horloge – mais par un protocole de communication implémenté dans le module. Un protocole de communication requiert un échange d'informations entre l'émetteur et le récepteur. Ces protocoles sont appelés protocoles à poignée de main (de l'anglais « handshake protocol ») et la communication se fait par des canaux contenant l'ensemble des signaux (données, requête et acquittement).

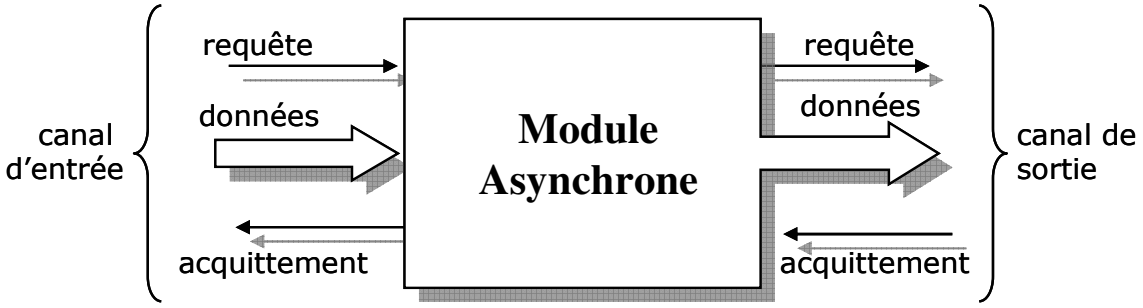


Figure II.2 – Un module asynchrone.

Le protocole de communication est la base des règles séquentielles qui régissent les circuits asynchrones. Un module commence son calcul si et seulement si toutes les données requises au calcul sont disponibles aux canaux d'entrées du module. Dès que le module peut stocker le résultat, le module libère ses canaux d'entrées. Le module envoie le résultat au canal de sortie si et seulement si le canal de sortie est disponible, c'est-à-dire, si le canal à été libéré par le module récepteur de l'information. L'implémentation dans chaque module du protocole de communication est le prix à payer pour s'affranchir d'une horloge globale.

Les modules asynchrones peuvent implémenter des fonctions avec différents niveaux de granularités : fonctions au niveau des bits, fonctions au niveau des mots, fonctions arithmétiques ou même d'algorithmes complexes. Cependant, ces modules ont toujours trois caractéristiques principales :

- ➔ tous leurs canaux respectent un protocole de communication uniforme ;
- ➔ la latence directe est minimale et elle est égale au temps nécessaire au module pour effectuer le calcul et placer le résultat dans le canal de sortie. C'est une importante différence par les circuits synchrones. Dans les circuits asynchrones, la latence directe peut varier avec les données à calculer ou avec le nombre de données stockées dans un pipeline.[II.2][II.3][II.4][II.5]
- ➔ le débit de sortie est maximal et il correspond à la vitesse maximale du module à calculer les donnés reçues. Le débit caractérise le temps maximum de cycle. Cette caractéristique n'est pas l'inverse de la latence directe parce qu'un module doit libérer ses canaux d'entrées pour recevoir des nouvelles données [II.5].

II.2. Protocoles de Communication

Un protocole de synchronisation (ou protocole de « poignée de main ») est un ensemble de règles régissant l'échange des informations au travers d'un canal entre plusieurs blocs asynchrones. Ainsi, chaque bloc asynchrone implémente le protocole de communication, ce choix est essentiel à la conception de tels circuits et ce choix conditionne les caractéristiques de l'implémentation du circuit (surface, vitesse, consommation, robustesse, etc.).

Un protocole de communication entre deux blocs asynchrones via un canal peut être caractérisé par :

- ➔ la séquence de transfert des informations ;
- ➔ le codage utilisé pour les données ;
- ➔ l'activité des portes : lequel entre émetteur ou récepteur démarre la communication.

Les différents protocoles proposés dans la littérature peuvent être divisés en deux grandes classes : protocoles deux phases – où la communication s'effectue en deux étapes – et protocoles quatre phases – où la communication s'effectue en quatre étapes.

II.2.1. Protocole 2-phases

La Figure II.3 montre un protocole deux phases aussi appelé NRZ pour « Non Retour à Zéro ». Il constitue la séquence minimale permettant l'échange d'informations entre deux blocs.

Les deux phases du protocole sont :

- **Phase 1** : le récepteur détecte la requête de l'émetteur, puis, reçoit et traite l'information et enfin envoie le signal d'acquittement ;
- **Phase 2** : le signal d'acquittement est détecté par l'émetteur indiquant que le canal de communication est libre et qu'une nouvelle information peut être émise.

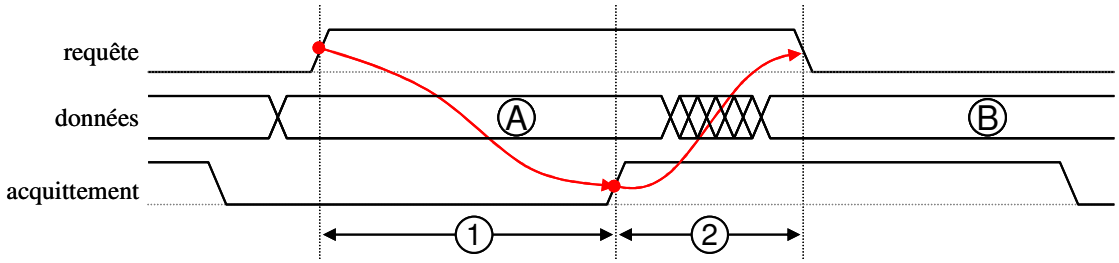


Figure II.3 – Protocole 2-phases.

Ce protocole est appelé de NRZ parce que chaque transition dans le signal de requête correspond à l'émission des nouvelles données. Donc, la requête aussi bien que l'acquittement sont détectés par les fronts (montant ou descendant) de ces signaux.

II.2.2. Protocole 4-phases

La Figure II.4 montre un protocole quatre phases aussi appelé RZ pour « Retour à Zéro ». Les quatre phases du protocole sont :

- **Phase 1** : le récepteur détecte la requête de l'émetteur, reçoit et traite l'information et envoie le signal d'acquittement ;
- **Phase 2** : l'émetteur détecte le signal d'acquittement et remet le signal de requête à zéro ;
- **Phase 3** : le récepteur détecte la remise à zéro de la requête et remet aussi le signal d'acquittement à zéro ;
- **Phase 4** : l'émetteur détecte la remise à zéro de l'acquittement indiquant que le canal est libre et qu'une nouvelle communication peut commencer.

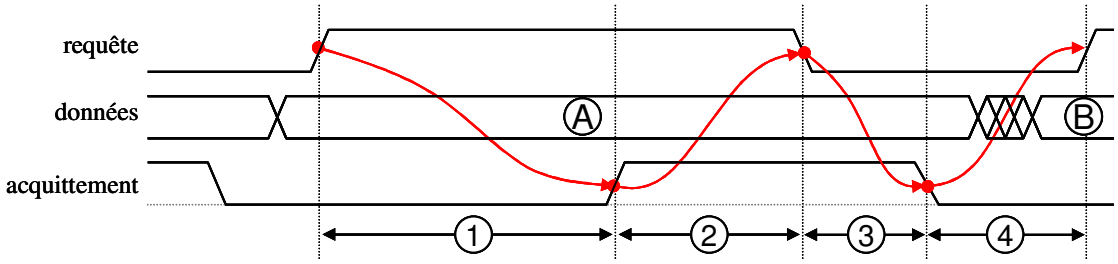


Figure II.4 – Protocole 4-phases.

Clairement, le nom de « Retour à Zéro » pour ce protocole exprime le fait que les signaux de requête et acquittement doivent être remis à zéro avant qu'une nouvelle communication puisse démarrer.

À première vue, le protocole 2-phases est plus rapide et consomme moins (moins de transitions sont nécessaires à la communication). En réalité dans la plupart des cas cette impression est fausse. Une logique basée sur des événements (détection de fronts d'ondes) est plus complexe à implémenter en utilisant la technologie CMOS qu'une logique basée sur des niveaux. De plus, de nombreuses optimisations peuvent être réalisées au niveau logique et architecturale qu'en utilisant le protocole 4-phases. De façon générale, le protocole 2-phases est préféré pour la conception de composants lents ayant grande latence directe, contrairement, le protocole 4-phases est largement utilisé lorsque le concepteur possède la liberté d'équilibrer la latence des composants afin d'optimiser les performances du circuit [II.6]. À titre d'exemple, le processeur ASPRO développé au sein du groupe CIS [II.6] utilise le protocole 4-phases dans le cœur et un protocole 2-phases pour son lien série avec l'extérieur.

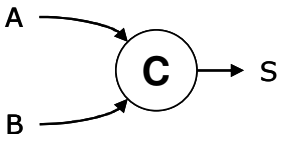
II.2.3. La Porte de Müller

Pour réaliser la synchronisation entre deux signaux nécessaire aux protocoles de communication, une implémentation utilisant des portes logiques élémentaires est très coûteuse. En [II.7] la porte de Müller (ou « C-Element ») a été introduite pour réaliser le rendez-vous de plusieurs signaux asynchrones. Une telle porte est donc essentielle à la conception de circuits asynchrones. Cette section présente succintement le fonctionnement des ces portes nécessaire pour la suite de cette thèse.

La sortie de la porte de Müller copie la valeur de ses entrées lorsque celles-ci sont identiques, sinon la dernière valeur de la sortie reste mémorisée. La Figure II.5 montre le symbole d'une porte de Müller à deux entrées et sa spécification.

À partir de la spécification de la porte de Müller plusieurs implémentations sont envisageables. La Figure II.6 montre deux implémentations possibles pour la porte de Müller.

Il est clair que l'implémentation en utilisant de portes logiques élémentaires est très coûteuse en surface et qu'une implémentation au niveau transistors est souhaitable.



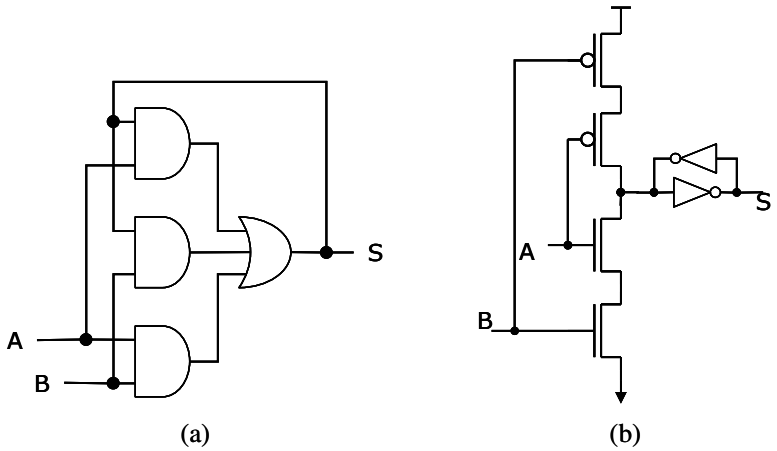
si A=B alors S←A

A	B	S
0	0	0
0	1	S
1	0	S
1	1	1

Figure II.5 – Symbole et spécification de la porte de Müller à deux entrées.

En [II.8], Alain Martin a proposé une porte de Müller généralisée (de l'anglais « Generalized C-Element »). La porte de Müller généralisée, aussi appelée dissymétrique, est une porte dans laquelle l'ensemble de signaux qui provoquent une commutation vers « 1 » (ensemble E_1) ne sont pas forcement les mêmes que ceux qui provoquent une transition vers 0 (ensemble E_0). Le fonctionnement d'une telle porte peut être décrit par :

- ➔ si tous les signaux appartenant à E_1 sont égaux à « 1 », alors la sortie est égale à « 1 » ;
- ➔ si tous les signaux appartenant à E_0 sont égaux à « 0 », alors la sortie est égale à « 0 » ;
- ➔ si aucune des conditions ci-dessus n'est pas vraie, la sortie reste inchangée.

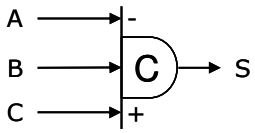


(a) (b)

Figure II.6 – Deux implémentations possibles pour la porte de Müller : (a) en utilisant portes logiques et (b) en transistors.

Pour illustrer, la Figure II.7 montre le symbole, la spécification et une possible implémentation d'une porte de Müller généralisée à trois entrées [II.9]. Dans cet exemple, les entrées A et B doivent être égales à « 0 » pour faire S commuter vers « 0 » alors que B et C

doivent être égales à « 1 » pour que S puisse commuter vers « 1 ». Dans tous les autres cas, la sortie S reste mémorisée.



A	B	C	S
0	0	X	0
X	1	1	1
autres combinaisons			S

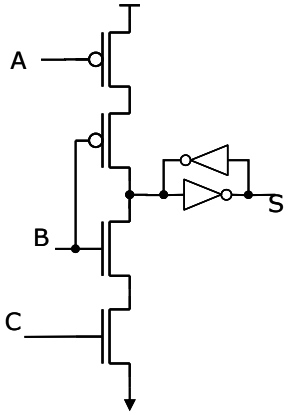


Figure II.7 – Exemple d'une porte de Müller dissymétrique.

II.3. Signalisation et Codage

Comme décrit avant, l'activité dans les circuits asynchrones est rythmée par la présence des données, requêtes et acquittements. Dans cette section la détection des nouvelles données et la génération des acquittements sont adressées. La détection des nouvelles données est aussi appelée de requête de calcul. De la même façon, la génération des acquittements implique que le calcul est fini et que les canaux peuvent être libérés. Ainsi, les acquittements sont aussi appelés « signaux de complétion ».

En conséquence, pour assurer la synchronisation entre les différents blocs, l'information de validité des signaux doit être associée avec chaque canal de communication. Ce principe nécessite un codage particulier des informations dans les circuits asynchrones. Aujourd'hui, il existe deux grands groupes des codages : les codages « données groupées » et les codages insensibles aux délais [II.10].

II.3.1. Codage « Données Groupées »

Signaler une requête de calcul ainsi que la validation des donnés se fait par un signal explicite (signal requête dans la Figure II.2). Les valeurs de données transmises peuvent donc être codées avec le schéma binaire traditionnel des circuits synchrones. Dans ce cas, chaque bit d'information est codé par un fil, ce qui parfois est appelé *mono-rail* (de l'anglais « single rail ») dans la littérature.

Ce type de codage peut être utilisée avec un protocole 2-phases (cf. Figure II.3) ou bien avec un protocole 4-phases (cf. Figure II.4). Comme cette technique est similaire à la génération d'une horloge locale à chaque canal de communication [II.11][II.12], le signal de requête doit être implémenté avec un retard adéquat pour assurer un bon fonctionnement du

circuit [II.13] [II.14] [II.15]. Cette codage est appelée données groupées (de l'anglais « bundled data ») parce que les valeurs valides de données se trouvent empaquetées dans un retard.

Efficace en terme de surface d'implémentation (en terme de nombre des fils utilisés et donc, en nombre de portes), ce codage permet une bonne réalisation de circuits asynchrones. Toutefois, comme les retards sur les signaux des requêtes sont fixes – et bien sûr égaux aux pires cas – la vitesse de propagation de l'information ne dépend pas de la propagation réelle des informations dans chaque bloc logique.

II.3.2. Codage Insensible aux Délais

Une alternative plus sophistiquée que le codage de données groupées est lorsque l'information de validité est incluse dans les données. Cette approche permet de détecter les données à leur arrivée sans que cela repose sur aucune hypothèse temporelle (retard du signal de requête) et, donc, insensible aux délais. De cette façon, le signal de requête peut être supprimé et un nouveau codage est requis pour les données. Ceci implique également plus de robustesse et de portabilité car la propre donnée porte la requête et l'information de validité.

II.3.2.1. Codage 4 états

Lorsqu'un protocole 2-phases est utilisé, la signalisation est faite par transitions. Dans ce cas, le récepteur doit être capable de détecter l'émission d'une nouvelle donnée même si une valeur identique est transmise deux fois. Ainsi, pour un bit d'information, quatre transitions sont possibles pour les données (0→0, 0→1, 1→0, 1→1) et deux bits – ou fils – sont nécessaires pour coder l'information. La Figure II.8 illustre une possible codage 4 états avec deux bits. Ici, la valeur de la donné transmise est déterminée par le bit moins significatif et une nouvelle requête est détectée par le changement de la parité du canal d'entrée.

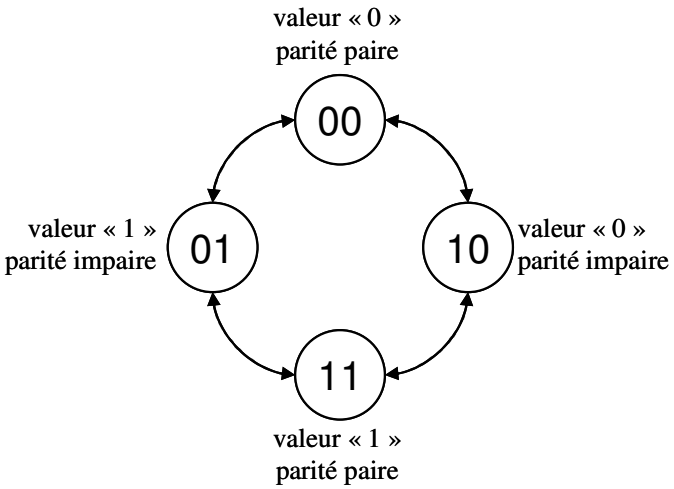


Figure II.8 – Codage 4 états.

II.3.2.2. Codage 3 états

Quand un protocole 4-phases est utilisé (signalisation est faite par niveau) et en supposant la transmission d'un bit d'information, le canal entrée peut assumer trois valeurs distinctes : valeur « 0 », valeur « 1 » et la valeur « invalide » pour la remise à zéro de la requête. Comme pour le codage 4 états, le codage 3 états nécessite l'utilisation de deux bits, par contre, un état restera interdit ou inutilisé. La Figure II.9 montre une possible codage 3 états.

Sur la figure, il est évident que le passage d'une valeur valide à une autre valeur valide se fait nécessairement par un passage par l'état invalide. À cause des difficultés d'implémentation d'un protocole 2-phases – implémentation d'une logique sensible au transitions – et du au fait que l'utilisation d'une codage « données groupées » implique à fusionner la requête avec les données, les codages 3 états est de loin le plus utilisé dans la conception asynchrone.

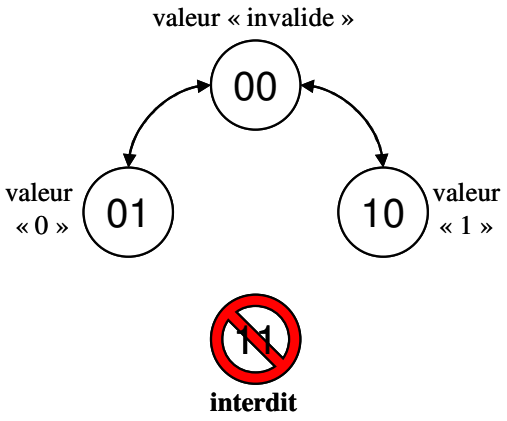


Figure II.9 - Codage 3 états.

Le codage 3 états, aussi appelé codage « *double rail* » en raison de l'utilisation de deux fils, est un cas particulier du codage « *one hot* ». Il est possible d'étendre cette représentation pour coder un valeur en base *N* et, donc, *N* fils sont nécessaires. En supposant les fils numérotés de 0 jusqu'à *N-1*, une valeur *x* est transmise en établissant le fil *x* égal à « 1 ». La remise à zéro se fait quand tous les fils sont égaux à zéro. Les combinaisons avec plus d'un fil égal à « 1 » sont interdites. Cette forme généralisée du codage est appelée codage « *1-parmi-N* ».

Il vrai que dans un codage *1-parmi-N*, le coût en nombre de fils est très élevé si *N* est grand. Donc, il existe aussi les codages « *M-parmi-N* », où *M* fils sont égaux à « 1 » pour coder une valeur. Ce type de codage réduit le nombre de fils nécessaires pour encoder une certaine base *N*, mais, malheureusement, implique une augmentation de la complexité des blocs logiques.

II.3.3. Génération des Acquittements

La plus simple génération d'un signal d'acquittement consiste à retarder le signal de requête avec un délai temporel correspondant au temps de calcul du module. Le signal retardé peut être directement utilisé comme le signal d'acquittement du module. Cette technique repose sur une hypothèse majeure : le retard du chemin du signal de requête jusqu'à l'acquittement est plus grand ou égal que le retard des chemins de données. En effet, le retard de l'acquittement est surévalué pour compenser les conséquences du routage et les variations du procès de fabrication, de température et de tension d'alimentation. La Figure II.10 illustre ce principe quand le protocole de données groupées est utilisé.

Une autre alternative pour les circuits CMOS est de détecter la fin du calcul par une mesure du courant consommé par le module [II.16]. Plusieurs circuits ont été proposés, mais cette méthode s'est avérée inefficace et difficilement réalisable.

Une technique plus robuste et plus fiable est l'utilisation du codage de données. En considérant un module conçu pour traiter des données codées avec le codage 3 états et pour respecter le protocole 4-phases, ce module produit des sorties en double rail et l'état des sorties peut être facilement détecté par une porte OR comme le montre la Figure II.11.

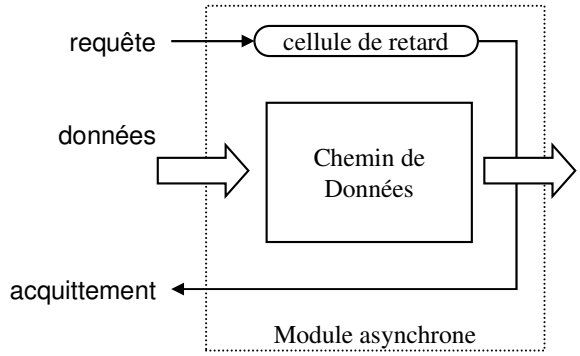


Figure II.10 – Génération de signal d'acquittement avec une cellule de retard.

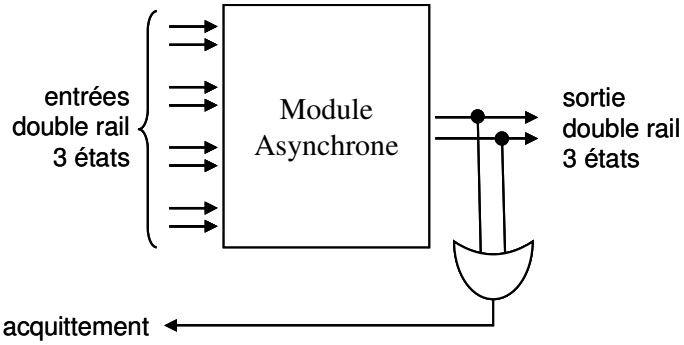


Figure II.11 – Génération de signal de complétion avec le codage 3 états.

La Figure II.12 présente la même technique pour la détection de fin de calcul quand un module asynchrone utilise le codage 4 états. Dans ce cas, une porte XOR (ou exclusif) est employée pour détecter l'émission de nouvelles données par le module.

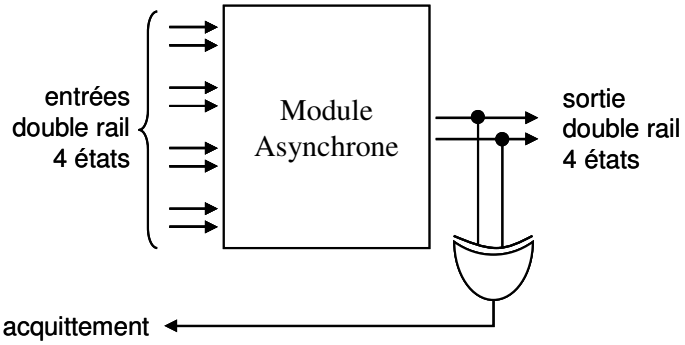


Figure II.12 – Génération de signal de complétion avec le codage 4 états.

II.4. Classification de Circuits Asynchrones

Les circuits asynchrones sont classifiés par leur modèle de délai et par la façon de communiquer avec leur environnement, c'est-à-dire, leur mode de fonctionnement.

II.4.1. Modèles de Délais

La notion de délai est, donc, très importante lors de l'implémentation de circuits asynchrones et les délais sont inhérents à la réalisation physique des circuits. Le délai correspond au temps de propagation d'un signal au travers d'une porte logique voire d'un fil. Un modèle de délai exprime la façon par laquelle le concepteur prend en compte les retards au moment de concevoir un circuit.

Les délais peuvent être caractérisés par leurs modèles temporels. Dans un modèle « *non borné* », le retard sur un fil et/ou une porte logique peut prendre n'importe quelle valeur positive et, donc, aucune borne supérieure est imposée à la valeur du délai. Dans un modèle dit « *borné* », la valeur du retard reste inconnue, mais la plage de variation de cette valeur est connue par le concepteur. Finalement, dans un modèle de délais « *fixes* », les valeurs de retards de portes et des fils sont fixées à des valeurs connues.

Les délais peuvent aussi être caractérisés par leur interaction avec le front d'onde. Il existe deux modèles courants : (i) le modèle de « *délai pur* » et le modèle de « *délai inertiel* » [II.17] [II.18]. Un modèle de délai pur ne change pas la forme d'onde des signaux, il retarde simplement le front d'onde temporellement. Au contraire, un modèle de délai inertiel peut

également modifier la forme d'onde des signaux. En particulier, le délai inertiel possède un période de seuil et toutes les impulsions avec une durée inférieure à la période de seuil sont filtrées.

II.4.2. Modes de Fonctionnement

Le mode de fonctionnement définit comment le circuit interagit avec son environnement. Le circuit et son environnement forment un « circuit complet » [II.7] ou un système fermé. Selon les terminologies employées par [II.19], les modes de fonctionnement sont les suivants :

- ➔ **Mode fondamental** : l'environnement doit attendre que le circuit soit dans un état stable (tous les signaux internes et les sortie du circuit sont dans un état stable) avant de répondre. Quand le circuit est stable, l'environnement est autorisé à changer les entrées, mais une seule entrée à la fois.
- ➔ **Mode rafale** : équivalent au mode fondamental, mais dans le mode rafale l'environnement, après que le circuit est atteint un état stable, peut changer plusieurs entrées à la fois. Après le changement des entrées, l'environnement doit attendre à nouveau que le circuit soit stable.
- ➔ **Mode entrée-sortie** : l'environnement répond aux requêtes du circuit sans aucune contrainte de délai. L'environnement ne possède aucune connaissance de l'état du circuit, c'est-à-dire, si le circuit est stable ou non. La seule restriction imposée à l'environnement est de respecter le protocole de communication avec le circuit.

II.4.3. Catégories de Circuits Asynchrones

Le terme de circuits auto séquencés (de l'anglais « self-timed circuits ») qui normalement est utilisé pour désigner les circuits asynchrones, ne correspond à aucune définition précise. En fait, à partir de leur modèle de délai et de leur mode de fonctionnement, les circuits sont classifiés. Clairement, le choix du modèle de délai et du mode de fonctionnement est un compromis entre la complexité et la robustesse des circuits [II.20]. La Figure II.13 présente la terminologie habituellement utilisée pour classifier les circuits asynchrones [II.21]. Plus le nombre d'hypothèses temporelles est élevé (contraintes sur les délais relâchées), plus simple sera le circuit. Cependant, dans ce cas, le circuit sera moins robuste.

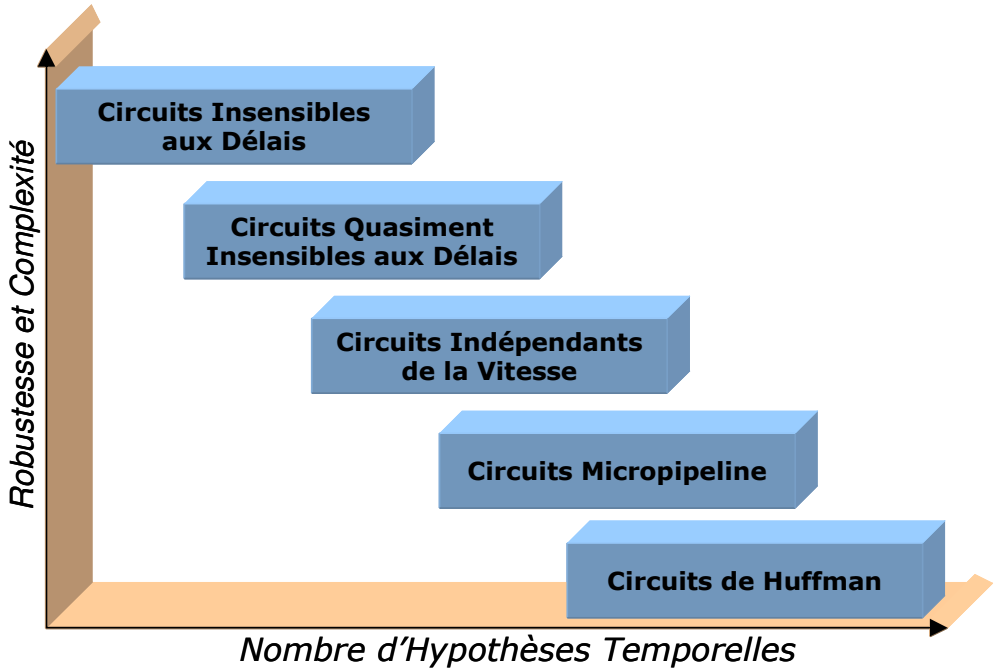


Figure II.13 – Catégories de circuits asynchrones.

II.4.3.1. Circuits de Huffman

Les circuits de Huffman [II.22] supposent un modèle de délai borné et un mode de fonctionnement fondamental. La séquence des actions est assurée par les délais dans le circuit et par le respect de l'environnement d'une attente minimum avant d'envoyer une nouvelle requête. Donc, la conception et le bon fonctionnement de tels circuits repose sur l'analyse détaillée des délais dans tous les chemins et boucles de façon à déterminer le temps d'occurrence des signaux de contrôles locaux. Il est évident que ces circuits sont très sensibles à leur caractérisation. Une faute de conception ou d'évaluation des délais rend ces circuits non fonctionnels et, en conséquence, ces circuits sont très peut robustes vis-à-vis de variations du procédé de fabrication, de la température et de la tension d'alimentation.

II.4.3.2. Circuits Micropipeline

Les circuits micropipeline ont été introduits par Sutherland [II.12]. Les circuits micropipeline sont plutôt une classe d'architecture de circuits asynchrones qu'un modèle de délai et de mode de fonctionnement. Un pipeline, synchrone ou asynchrone, est une structure qui traite et stocke des données. Les structures de stockage (registres) et de traitement (bloc logiques) sont placées de façon alternée dans la longueur du pipeline. En regardant simplement les éléments de stockage, un pipeline se comporte comme une FILE traversée par les données, où la première donnée à rentrer dans le pipeline sera la première donnée à sortir (FIFO – de l'anglais « *first-in-first-out* »). Donc, dans les circuits micropipelines, l'horloge globale est remplacée par une structure de synchronisation locale qui sert à contrôler la file de données. La

Figure II.14 montre la structure de base pour le contrôle de circuits micropipelines. Elle se compose d'éléments identiques connectés tête-bêche.

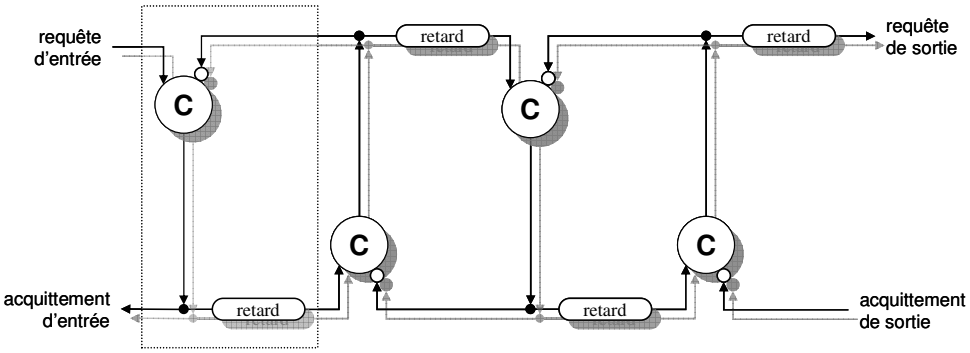


Figure II.14 – Structure de base pour le contrôle de circuits micropipeline.

La structure de la Figure II.14 implémente un protocole 2-phases vu que le circuit réagit à des transitions dans les signaux et non pas à des niveaux. En supposant tous le signaux initialement égaux à zéro, une transition positive dans la requête d'entrée produit une transition positive dans l'acquittement d'entrée qui est également propagée au étage suivante. Le deuxième étage produit une transition positive qui est propagée à l'étage suivant, mais aussi envoyée au premier étage pour l'autoriser à traiter une transition négative cette fois. Les transitions avancent dans la structure tant qu'elles ne rencontrent pas une cellule occupée.

Maintenant, la structure simple de contrôle de la Figure II.14 peut être utilisée pour contrôler un pipeline. La Figure II.15 illustre un pipeline de quatre étages avec le codage « données groupées » pour les données.

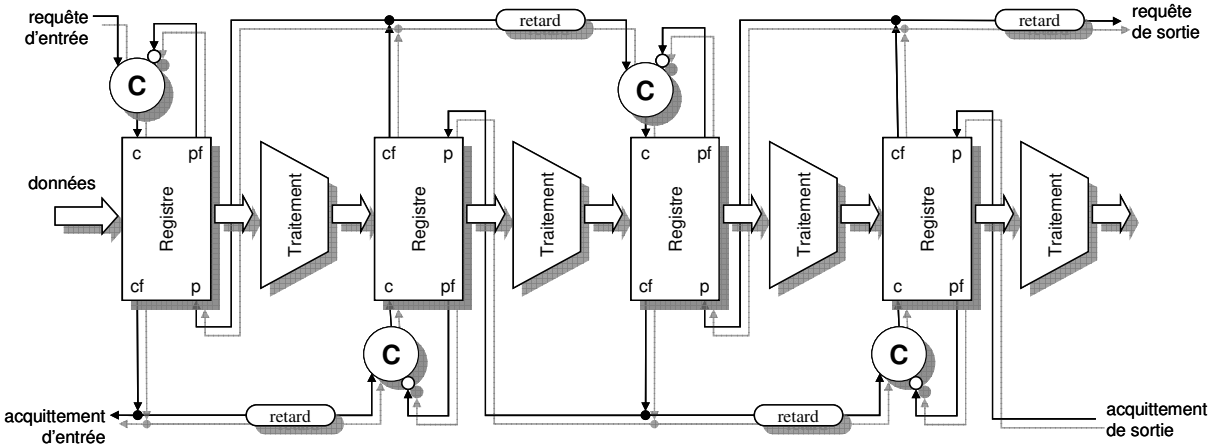


Figure II.15 – Micropipeline avec traitement.

Dans cette structure, les registres mémorisent leurs entrées à l'occurrence du signal « capture » (entrée *c* du registre) et produisent le signal « capture faite » (sortie *cf* du registre) à la fin de la mémorisation. En pratique, le signal « capture faite » est le signal « capture » retardé du temps nécessaire au registre pour effectuer la mémorisation. À l'arrivée du signal « passe » (entrée *p* du registre), les registres placent leur données à la sortie et produisent le signal « passe fait » (sortie *pf* du registre) pour indiquer que les données ont été envoyées.

La principale caractéristique de cette structure et la principale motivation pour son utilisation est le fait que la FIFO est élastique. C'est-à-dire que le nombre de données présentes dans le pipeline peut être variable et les données progressent dans la chaîne aussi loin que possible en fonction du nombre d'étages disponibles. Malgré cela, ce type de circuit révèle un certain nombre d'inconvénients. Pour s'affranchir du problème d'aléas dans la logique de traitement, un retard est ajouté dans la chaîne de contrôle. Donc, le temps est discrétisé et la mémorisation s'effectue lors que le circuit de traitement est stable. Comme les retards sont fixes dans la proposition initiale de Sutherland, chaque étage réalise le traitement dans le pire cas. Ainsi, avec cette type de structure, il n'est pas possible de tirer parti de la variation dynamique du temps de calcul des logiques de traitement.

II.4.3.3. Circuits Indépendants de la Vitesse

Les circuits indépendants de la vitesse (SI – de l'anglais « speed-independent »), ou circuits de Müller, ont été initialement décrits par [II.7]. Ces circuits utilisent un modèle de délais non bornés pour les portes logiques, mais il néglige les délais dans les fils. Dans ces circuits toutes les fourches sont *isochroniques*. Une fourche est un signal qui est envoyé à plusieurs portes. Comme il n'y a pas de retards sur les fils, quand un signal change de valeur, toutes les portes connectées au signal perçoivent le changement instantanément. Comme les portes dans la fourche reçoivent le signal au même instant, la fourche est dite isochronique.

Malheureusement, dans les technologies actuelles, le retard dû au routage est de plus en plus critique. Donc, un modèle qui néglige ces retards n'est pas souhaitable.

II.4.3.4. Circuits Insensibles aux Délais

Les circuits insensibles aux délais [II.23] [II.24] (DI – de l'anglais « delay-insensitive ») n'imposent aucune restriction aux délais des portes logiques et aux délais de fils. Pour l'environnement, il doit simplement respecter le protocole de communication. Donc, ce type de circuit utilise un modèle de délai non borné pour les portes et les fils et, donc, il se comporte conformément à sa spécification indépendamment des délais insérés par les portes et fils.

Malheureusement, il n'est pas toujours possible de concevoir des circuits DI. Dans les cas où des portes logiques à une seule sortie sont utilisées, seulement des circuits conçus avec des inverseurs et des portes de Müller seront DI. L'ensemble de fonctions réalisables avec ces portes est très limité [II.25].

II.4.3.5. Circuits Quasi Insensibles aux Délais

Les circuits quasi insensibles aux délais (QDI – de l'anglais « quasi delay-insensitive ») est un sous ensemble des circuits insensibles aux délais où la notion de fourche isochronique a été insérée. Dans [II.26], Alain Martin a montré que l'hypothèse temporelle de fourche isochronique est la plus faible contrainte à ajouter aux circuits DI pour les rendre réalisables en utilisant des portes à plusieurs entrées et une seule sortie (portes logiques élémentaires). Les circuits QDI sont donc réalisables avec des cellules standard utilisées par les circuits synchrones et par ce fait l'intérêt pour classe de circuits est très important.

Même si pendant longtemps la communauté asynchrone a tenté de cerner les différences entre le modèle QDI et le modèle SI, il y a aujourd'hui un consensus pour considérer les deux modèles équivalents. Dans [II.27], Hauck a démontré avec le schéma de la Figure II.16 comment les fourches isochrones sont perçues par les deux modèles.

Malgré leurs similitudes, le modèle QDI est plus pertinent parce que, à la différence du modèle SI où toutes les fourches sont isochrones, dans le modèle QDI seules certaines fourches, identifiables par le modèle, doivent être isochrones.

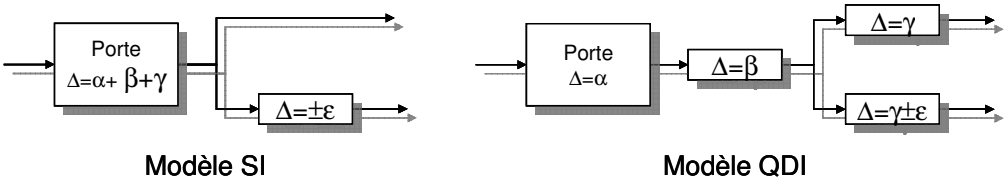


Figure II.16 – Équivalence entre le modèle SI et le modèle QDI.

Finalement, dans les circuits QDI, les signaux peuvent mettre un temps arbitraire à se propager dans les portes ainsi que dans les connexions sans que cela remette en cause la fonctionnalité du circuit. De telles réalisations sont, bien entendu, très délicates à mettre au point vu les contraintes imposées à la conception. En contrepartie, elles offrent un dégrée de robustesse quasi-parfait. En théorie, aucune erreur inhérente au circuit ne peut se produire et, en pratique, la contrainte de fourche isochrone est assez faible et facilement remplie par une conception soignée. Il suffit que la dispersion des temps de propagation jusqu'aux extrémités de la fourche soit inférieure aux délais des opérateurs qui lui sont connectés – au minimum une porte et un fil dont une sortie interagit avec la fourche [II.25] [II.28]– pour rendre le circuit fonctionnel.

II.5. Conclusions

La présentation des principes des circuits asynchrones vient d'être faite et celles-ci proposent une alternative à la conception synchrone. Différentes classes de circuits asynchrones ont été présentées avec différentes hypothèses temporelles et modes de fonctionnement donc résultent des circuits plus ou moins robustes et plus ou moins complexes.

Malgré les avantages et les gains potentiels apportés, surtout avec la suppression d'horloge globale, des problèmes subsistent. Les difficultés de l'adoption de tels circuits résident principalement dans leur conception. Quand le temps est discrétisé, le concepteur peut s'affranchir des problèmes liés aux aléas ou aux phénomènes transitoires. Dans les circuits asynchrones où le temps n'est pas discret, toute transition d'un signal peut être interprétée comme un événement porteur d'information. En conséquence, tous les aléas sont susceptibles de causer un mauvais fonctionnement. Dans ce cadre, les principales difficultés à l'adoption des circuits asynchrones sont : (i) d'assurer la correction du fonctionnement par des méthodes de conception qui puissent garantir une implémentation sans aléas et, en conséquence, (ii) un manque d'outils de CAO capables de réaliser une telle conception.

De ce fait, la communauté asynchrone réalise beaucoup d'efforts pour offrir des solutions à la conception de circuits asynchrones. Même si aujourd'hui, il n'existe pas une vision unificatrice, d'innombrables méthodologies de conception ont été proposées pour les différentes classes des circuits asynchrones reflétant la large gamme de solutions possibles. Dans ce contexte, cette thèse essaie de contribuer à combler l'espace entre le concepteur et les circuits asynchrones.

II.6. Références

[II.1] T. J. Chaney, C. E. Molnar, "Anomaloues behavioural of synchronizer and arbiter circuits", IEEE Transaction on Computers, C-22(4):421-422, April, 1973.

[II.2] U.V. Cummings, A.M. Lines, A.J. Martin, "An Asynchronous Pipelined Lattice Structure Filter", in Proceedings of the International Symposium on Advanced Research in Asynchronous Circuits and Systems, Salt Lake City, Utah, pp. 126-133, November 3-5, 1994.

[II.3] B. El Hassan, A. Guyot, M. Renaudin et V. Levering , "New self timed ring and their application to division and square root extraction", ESSCIRC'95, Lille, France, Sept. 1995.

[II.4] T.E. Williams, "Self timed rings and their application to division", Ph.D dissertation, Stanford university, May 1991.

[II.5] T.E. Williams, "Performance of iterative computation in self timed rings", Journal of VLSI signal processing, N°7, pp 17 - 31, Feb.1994.

[II.6] M. Renaudin, P. Vivet, F. Robin, "ASPRO-216 : a standard-cell Q.D.I. 16-bit RISC asynchronous microprocessor", Proc. of the Fourth International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC'98), San Diego - USA, 1998, p. 22-31.

[II.7] R. E. Miller. Sequential Circuits and Machines, volume 2 of Switching Theory. John Wiley & Sons, 1965.

[II.8] Alain J. Martin. Compiling communicating processes into delay-insensitive VLSI circuits. Distributed Computing, 1(4):226-234, 1986.

[II.9] J-B. Rigaud, "Spécification de bibliothèques pour la synthèse de circuits asynchrones", PhD Thesis, INP Grenoble, 2002.

[II.10] M. Renaudin. Asynchronous circuits and systems: a promising design alternative. Journal of microelectronic engineering, 54: 133-149, 2000.

[II.11] K. Y. Yun, P. A. Beerel, and J. Arceo. "High-performance asynchronous pipeline circuits". In Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems. IEEE Computer Society Press, March 1996.

[II.12] I.E. Sutherland, "Micropipelines", Communication of the ACM, Volume 32, N°6, June 1989.

[II.13] Stephen B. Furber, James D. Garside, Peter Riocreux, Steven Temple, Paul Day, Jianwei Liu, and Nigel C. Paver. AMULET2e: An asynchronous embedded controller. Proceedings of the IEEE, 87(2):243-256, February 1999.

[II.14] Joep Kessels and Paul Marston. Designing asynchronous standby circuits for a low-power pager. Proceedings of the IEEE, 87(2):257-267, February 1999.

[II.15] Lars S. Nielsen and Jens Sparsø. Designing asynchronous circuits for low-power: An IFIR filter bank for a digital hearing aid. Proceedings of the IEEE, 87(2):268-281, February 1999.

[II.16] M. Dean, D.Dill, M. Horowitz, "Self timed logic using current sensing completion detection", Journal of VLSI signal processing, N°7, pp 7 - 17, Feb.1994.

[II.17] S. H. Unger. Asynchronous Sequential Switching Circuits. Wiley-Interscience, John Wiley & Sons, Inc., New York, 1969.

[II.18] S. H. Unger. Asynchronous sequential switching circuits with unrestricted input changes. IEEE Transactions on Computers, 20(12):1437-1444, December 1971.

[II.19] J. A. Brzozowski and J. C. Ebergen. Recent developments in the design of asynchronous circuits. In J. Csirik, J. Demetrovics, and F. Gécseg, editors, Fundamentals of Computation Theory, FCT'89, volume 380 of Lecture Notes in Computer Science, pages 78-94, FCT'89, Szeged, Hungary, 1989. Springer-Verlag.

[II.20] P. Vivet, "Une Méthodologie de Conception de Circuits Intégrés Quasi-Insensibles aux Délais: Application à l'Etude et à la Réalisation d'un Processeur RISC 16-bits Asynchrone", Thèse de Doctorat, Institut National Polytechnique de Grenoble, 2001.

[II.21] Chris Myers. Asynchronous Circuit Design. John Wiley & Sons, 2001.

[II.22] D. A. Huffman. The synthesis of sequential switching circuits. In E. F. Moore, editor, Sequential Machines: Selected Papers. Addison-Wesley, 1964.

[II.23] Wesley A. Clark. Macromodular computer systems. In AFIPS Conference Proceedings: 1967 Spring Joint Computer Conference, volume 30, pages 335-336, Atlantic City, NJ, 1967. Academic Press.

[II.24] Jan Tijmen Udding. A formal model for defining and classifying delay-insensitive circuits. Distributed Computing, 1(4):197-204, 1986.

[II.25] Alain 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.

[II.26] Alain J. Martin. Programming in VLSI: From communicating processes to delay-insensitive circuits. In C. A. R. Hoare, editor, Developments in Concurrency and Communication, UT Year of Programming Series, pages 1-64. Addison-Wesley, 1990.

[II.27] Scott Hauck. Asynchronous design methodologies: An overview. Proceedings of the IEEE, 83(1):69-93, January 1995.

[II.28] Kees van Berkel. Beware the isochronic fork. Integration, the VLSI journal, 13(2):103-128, June 1992.

Chapitre III

TAST

L'outil TAST (de l'anglais « TIMA Asynchronous Synthesis Tool ») est un outil de conception asynchrone développé au sein du groupe CIS (de l'anglais « Concurrent Integrated Systems ») au laboratoire TIMA. Cet outil s'encadre dans les efforts de la communauté asynchrone et, en spécial, du groupe CIS d'offrir un environnement adapté à la conception de circuits asynchrones. L'outil TAST est un environnement intégré visant le projet, l'exploitation architecturelle et l'entraînement des concepteurs dans le domaine asynchrone.

Les circuits asynchrones sont modélisés dans TAST comme un ensemble de processus communicants. Chaque processus reçoit les données par ses ports d'entrée, effectue les traitements adéquats et envoie les résultats à ses ports de sortie. Dans les prochaines sections, le flot de conception TAST est décrit.

III.1. Flot de Conception

La Figure III.1 présent le flot de conception TAST [III.1] [III.2]. Le langage CHP est utilisé comme langage de spécification. En fait, ce langage est une version enrichie du langage CHP (de l'anglais « Communicating Hardware Processes ») initialement proposé par Alain Martin [III.3]. La spécification CHP est compilée dans un format intermédiaire basé sur les réseaux de Petri. La décomposition des différents processus et les éventuels raffinements est effectuée sur des réseaux de Petri formalisés. Ensuite et en partant d'un réseau de Petri, plusieurs solutions architecturales peuvent être adressées : circuits micropipelines, circuits quasi insensibles aux délais et même des circuits synchrones. Les moteurs de synthèse dédiés produisent un réseau de portes logiques dans le langage VHDL qui, ensuite, est simulé et

synthétisé physiquement avec les outils de CAO commerciaux. De plus, un modèle en VHDL comportemental ou en langage C peut être aussi généré en vue de valider la spécification initiale en CHP par simulation.

Ce flot a déjà été employée comme support, jusqu'à la réalisation physique, à la conception des deux microprocesseurs asynchrones – ASPRO [III.4] et MICA [III.5] – et au développement d'un processeur DES de cryptographie [III.6].

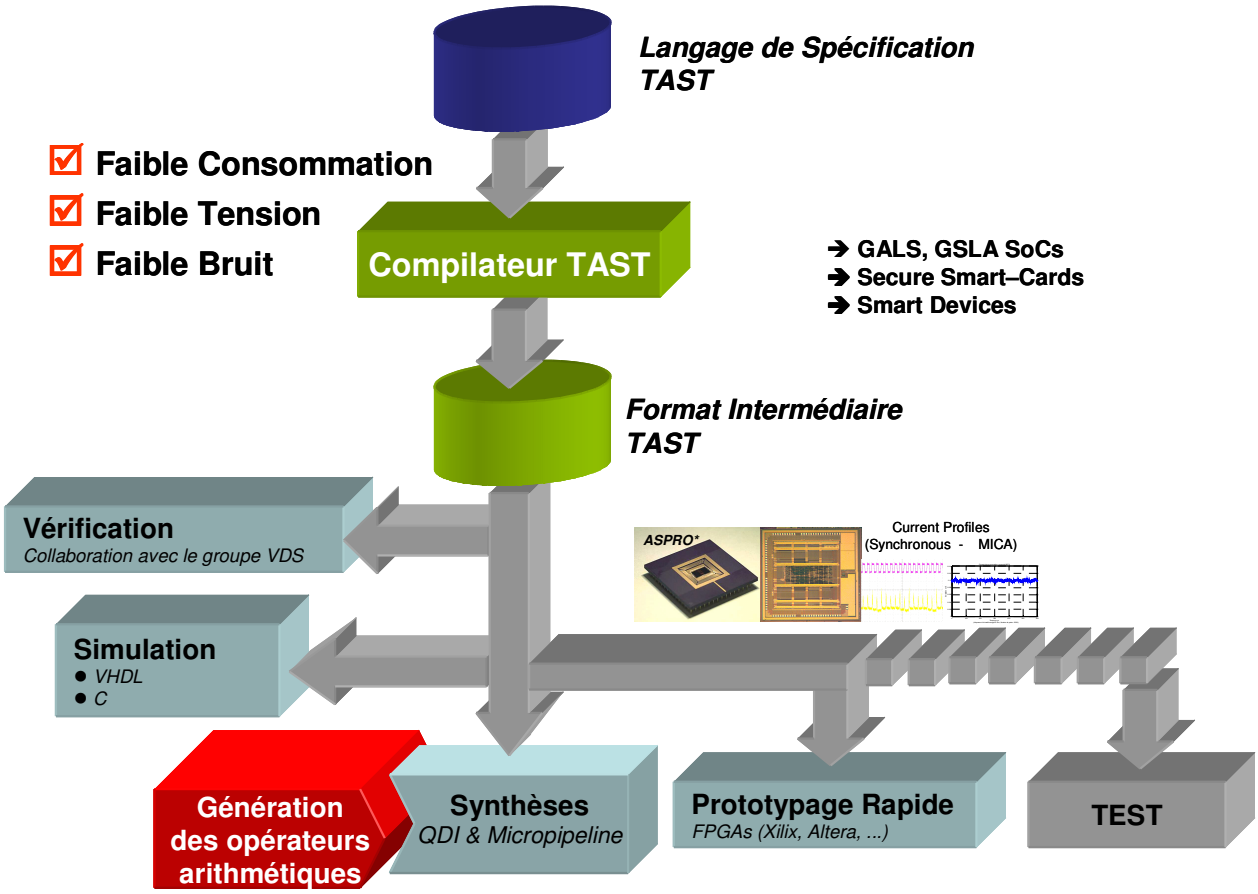


Figure III.1 – Flot de conception TAST.

La vérification de la spécification CHP peut être aussi effectuée en utilisant un outil dédié développé en collaboration avec le groupe VDS (de l'anglais « Verification and Modeling of Digital Systems ») du laboratoire TIMA [III.7].

III.2.Le Langage CHP

Dans cette section le langage CHP est brièvement introduit afin de fournir les bases nécessaires aux lecteurs pour comprendre les principales motivations de cette thèse.

III.2.1. Littéraux

Les littéraux, ou les constants entiers, sont notés dans le langage en précision fixe avec la syntaxe suivante : "<digit>.<digit>. …"[B][L] qui spécifie un vecteur de longueur L dans la

base B. Chaque digit a une valeur dans l'intervalle [0, B-1], et B et L sont des valeurs naturelles non nulles.

Exemples :

"1.9.7.5"[10] est évidement la valeur 1975.

"1.2.3"[4] est égal à $1\times4^2+2\times4^1+3\times4^0$ et, donc, égale à 27.

Pour faciliter l'écriture, les notations dans les bases communément utilisées – base 2 et 10 ainsi que – les notations standard sont supportées pour les nombres entiers et binaires. Dans la suite de cette thèse, une discussion plus approfondie de la représentation de nombres en différentes bases sera présentée.

III.2.2. Types de Données

Dans le langage CHP plusieurs types de données existent pour la déclaration de variables. Les types de données peuvent être repartis dans deux ensembles : les types signés et les types non signées.

III.2.2.1.Types Non Signés

- **MR[B] :** type multi-rail dans la base B. Ce type représente un nombre dans l'intervalle [0, B-1] et le nombre est codé en utilisant un codage insensible aux délais « *1-parmi-B* ».
- **MR[B][L] :** ce type correspond à un vecteur de longueur L d'un multi-rail dans la base B. Une variable de ce type peut contenir valeurs dans l'intervalle $[0,B^L-1]$.
- **DR :** type double rail – équivalent au type MR[2].
- **DR[L] :** un vecteur de longueur L du type double rail et, évidement, équivalent au type MR[2][L].
- **BIT :** représente un nombre binaire (0 ou 1) dans un codage double rail. Cela est aussi équivalent au type MR[2].
- **BIT[L] :** un vecteur de bits de longueur L.
- **BOOLEAN :** spécifie un valeur booléenne (vrai ou faux) aussi dans un codage double rail et, donc, équivalent à MR[2].
- **NATURAL [max] :** représente un nombre naturel allant de zéro à « max » - équivalent au type MR[2][$\lceil \log_2(\max+1) \rceil$] .

➔ **SR :** type mono rail (de l'anglais « single rail »). Ce type sert uniquement à la déclaration de signaux de synchronisation entre différents processus communicants. En conséquence, ce type ne peut pas être utilisé dans la déclaration de variables.

III.2.2.2.Types Signés

Tous les types non signés, à l'exception du type SR, possèdent leur équivalent signé. Le systèmes de numérotation pour les nombres signés et non signés sera discuté dans les chapitres suivants.

➔ **SMR[B]** : type multi-rail signé dans une base B. il représente un nombre soit dans l'intervalle $\left[-\frac{B}{2}, \frac{B}{2}-1\right]$ si B est paire, soit dans l'intervalle $\left[-\frac{B-1}{2}, \frac{B-1}{2}\right]$ si B est impaire.

➔ **SMR[B][L] :** un vecteur de longueur L du type multi rail signé dans une base B. les variables de ce type peuvent contenir soit des valeurs dans l'intervalle $\left[-\frac{B^L}{2}, \frac{B^L}{2}-1\right]$ si la base est paire, soit de valeurs dans l'intervalle $\left[-\frac{B^L-1}{2}, \frac{B^L-1}{2}\right]$ si la base est impaire. Les valeurs négatives sont codées en utilisant le système de complément à la base.

➔ **SDR :** type double rail signé équivalent au type SMR[2].

➔ **SDR[L] :** un vecteur de longueur L du type SDR. Ce type est équivalent au SMR[2][L].

➔ **INTEGER[max] :** représente un entier allant de –(max+1) à +max. Ce type est équivalent à SMR[2][$\lceil \log_2(2\times \max + 2) \rceil$] .

III.2.3. Opérateurs

Les opérateurs suivants sont disponibles dans le langage CHP :

➔ Comparaisons : < (inférieur), <= (inférieur ou égal), > (supérieur), >= (supérieur ou égal), = (égal), /= (inégal).

➔ Arithmétiques : +, -, *, mod, neg, abs, sll, sla, srl, sra, rol, ror.

➔ Logiques: not, nand, and, nor, or, xor, xnor.

- ➔ Communications: ! (écriture), ? (lecture), # (sonde).
- ➔ Affectations : **:=**
- ➔ Composition : « ; » (séquence) et « , » (concurrence). L'opérateur de séquence I1;I2 indique que l'instruction I2 doit commencer après l'exécution de I1. Au contraire, l'opérateur concurrence I1,I2 indique que les instructions I1 et I2 doivent être exécutées en parallèle.

Tous les opérateurs sont définis pour tous les types de données – signés et non signés – du langage CHP.

III.2.4. Structures de Contrôle

Les structures de contrôles sont basées sur des blocs de commandes gardées. Les gardes sont dénotées « guard_*i* » et les blocs de commandes sont dénotés « block_*i* ». Chaque garde est finie par une commande « loop » ou « break ».

III.2.4.1. Sélection Déterministe

Cette structure attend jusqu'à ce qu'une seule garde soit vraie. Quand la garde est vraie, le bloc correspondant est exécuté. Si la garde finie par une commande « loop » la structure est réévaluée sinon la sélection est terminée. La syntaxe pour une sélection déterministe est la suivante :

```
@[
     guard_1 => block_1; break | loop
     guard_2 => block_2; break | loop
     …
     guard_n => block_n; break | loop
 ]
```

III.2.4.2. Sélection Non Déterministe

Cette structure attend jusqu'à ce qu'une ou plusieurs gardes soient vraies. Si plusieurs gardes sont vraies, un tirage aléatoire est fait pour déterminer la garde à exécuter. Le bloc correspondant à la garde choisie est exécuté. Si la garde finie par une commande « loop » la structure est réévaluée sinon la sélection est terminée. La syntaxe pour une sélection non déterministe est la suivante :

```
@@[
     guard_1 => block_1; break | loop
     guard_2 => block_2; break | loop
     …
     guard_n => block_n; break | loop
  ]
```

III.2.5. Structure d'un Programme CHP

La modélisation de la structure de programme en CHP a été inspirée par les langages commerciaux de description matérielle existants, comme le VHDL. La hiérarchie est assurée par la possibilité de déclarer des processus et/ou d'instancier d'autres composants. Ainsi, il est possible avec le langage CHP de gérer les niveaux de complexités exigés par les systèmes VLSI.

III.2.5.1.Composant

Un composant est la vue globale d'un circuit. Un composant est constitué par une interface de communication, par une partie déclarative et par son corps. L'interface de communication est, comme dans VHDL, la liste de ports et leurs directions. La partie déclarative sert à la déclaration des objets locaux – canaux de communication internes et constants. Le corps du composant est constitué d'instructions concurrentes telles que des instances d'autres composants et des déclarations de processus. Les canaux de communications déclarés dans la partie déclarative servent à connecter les différents processus et instances de composant et seulement les communications point à point sont autorisées.

La syntaxe de déclaration d'un composant est la suivante :

```
component name
     port (port list)
     {declaration part}
begin
     component body
end name ;
```

III.2.5.2.Processus

Contrairement aux langages de description matérielle, un processus CHP doit déclarer explicitement ses ports de communication, tout comme un composant. Ceci permet de déclarer proprement les attributs des canaux (codage, direction et protocole) pour ensuite, pendant la synthèse de circuits cibles, effectuer les vérifications nécessaires. Donc, comme le composant, un processus est composé par une interface de communication, une partie déclarative (déclaration des objets locaux) et un corps (ensemble des instructions). Le corps du processus fini par une instruction « break », le processus est exécuté une seule fois, sinon le processus est toujours actif. La syntaxe de déclaration d'un processus est la suivante :

```
process name
     port (port list)
     {declaration part}
begin
[
```

```
        instruction list;
        break | loop
    ];
    end;
```

III.2.6. Un Exemple

La Figure III.2 montre un exemple de code CHP pour un sélecteur. Comme dans cet exemple il n'existe qu'un seul processus, la liste de ports du processus est identique à la liste de ports du composant. Dans l'exemple de la Figure III.2, le canal d'entrée « C » est lu dans la variable local « ctrl » du processus. Ensuite, la variable « ctrl » est évaluée par intermédiaire d'une sélection déterministe. Si la valeur de « ctrl » est égale à « 0 », le canal d'entrée « E » est lu et envoyé au canal de sortie « S1 ». Si la valeur de « ctrl » est égale à « 1 », le canal d'entrée « E » est lu et envoyé au canal de sortie « S2 ». Finalement, si la valeur de « ctrl » est égale à « 2 », le canal d'entrée « E » est lu et envoyé parallèlement aux canaux « S1 » et « S2 ».

```
component selector
    port (
        E: in DR;
        C: in MR[3][1];
        S1, S2 : out DR
    )
begin
    process main
        port(
            C: in MR[3][1];
            E: in DR;
            S1, S2 : out DR
        )
        variable x : DR;
        variable ctrl : MR[3][1];
    begin
    [
        [ C?ctrl ;
        @[
            ctrl = "0"[3] => E?x; S1!x; break
            ctrl = "1"[3] => E?x; S2!x; break
            ctrl = "2"[3] => E?x; S1!x, S2!x; break
            ];
        loop
        ];
    ];
    end;  --process main
end; -- selector
```

Figure III.2 – Code CHP pour un sélecteur.

Il est important de souligner que le canal d'entrée « C » aussi bien que la variable « ctrl » sont déclarés en utilisant le type MR[3][1]. Ceci implique que le canal « C » est définie comme un type insensible aux délais et codé avec trois fils. Dans la structure de contrôle, il est possible d'identifier les trois valeurs possibles pour le canal « C ».

III.3.Circuits Synthétisables

La forme DTL (de l'anglais « Data Transfer Level ») proposée par A. Dinh-Duc [III.1] [III.8] définit des règles d'écriture de programmes pour assurer la synthèse des circuits asynchrones. De façon équivalente aux règles RTL (de l'anglais « Register Transfer Level ») existent pour les circuits synchrones, il est possible d'établir une méthodologie générale pour la synthèse des circuits asynchrones à partir de la forme DTL.

Les règles DTL définissent une forme pour l'écriture de programmes, dans n'importe quel langage approprié, qui assure, donc, une synthèse systématique des circuits asynchrones. Même si aujourd'hui seulement le langage CHP est supporté par le compilateur TAST, la forme DTL est indépendante du langage d'entrée choisi. Actuellement, le groupe CIS travaille pour offrir des supports à d'autres langages et les principaux langages commerciaux sont envisagés comme VHDL, Verilog et System-C.

La forme DTL ne fait aucune hypothèse sur les circuits cibles et, dans ce contexte, la forme DTL apparaît comme une forme généralisée du RTL vu que les circuits synchrones peuvent aussi être adressés. Ainsi, la forme DTL permet d'automatiquement cibler n'importe quelle classe de circuits asynchrones avec les différents protocoles et, donc, les circuits cibles QDI [III.9] [III.10] et micropipeline [III.11] proposés dans le flot TAST.

Pour qu'un processus soit conforme à l'écriture DTL, ce processus doit respecter les règles suivantes :

1. les variables partagées ne sont pas autorisées sauf dans le cas où les variables sont uniquement consultées;
2. une variable doit être affectée avant d'être lue
3. les valeurs envoyées aux canaux de sortie doivent exclusivement dépendre des valeurs reçues sur les canaux d'entrée ;
4. un même canal ne peut être accédé séquentiellement que dans le cas où les canaux de sortie sont initialisés au début du processus ;
5. un même canal ne peut pas être accédé concurremment ;

6. deux instructions séquentielles doivent exprimer une dépendance vraie.

Les règles DTL visent l'élimination de toute mémoire fonctionnelle d'un processus. Si ces règles sont respectées, les sorties d'un processus dépendent exclusivement des signaux d'entrées et, donc, un processus DTL unique ne peut pas être une machine séquentielle. Chaque processus DTL est un processus combinatoire communicant et les machines séquentielles sont obtenues par la composition des différents processus.

Dans [III.8], il est démontré comment transformer un processus que ne respectent pas les règles en un processus conforme aux règles DTL. Il est aussi démontré qu'un circuit asynchrone quelconque peut être décrit dans n'importe quel langage qui respecte ces règles.

III.4.Insertion du Présent Travail

L'intérêt particulier dans les circuits QDI et micropipeline découle de la possibilité de leur réalisation en utilisant une bibliothèque de cellules standard. Pour cette raison, le flot TAST préconise ces deux styles de conception qui, à partir d'un réseau de portes logiques produit par les moteurs de synthèse TAST, peuvent facilement être insérés dans des outils commerciaux pour l'étape suivante de synthèse physique. Dans ce contexte, les travaux de A. Dinh-Duc [III.8], A. Rezzag [III.12] et P. Vivet [III.13] proposent des méthodologies de conception pour ces deux classes de circuits.

Cependant, comme pour les circuits synchrones, les résultats de la synthèse logique des circuits peuvent être améliorés par l'utilisation de compilateurs dédiés à la génération de certains modules spécifiques comme les mémoires, les opérateurs arithmétiques, etc. Un compilateur dédié exploite la régularité et la connaissance de l'architecture du module pour optimiser le circuit généré. Cette connaissance est difficilement intégrée dans des méthodologies de conception plus générales. Ce fait a pu être constaté dans le travail de P. Vivet [III.13] où un microprocesseur a été réalisé. P. Vivet a identifié la nécessité de concevoir certaines cellules spécifiques pour optimiser les résultats de synthèse.

Dans ce cadre, cette thèse réalise une étude des deux principaux opérateurs arithmétiques présents dans les chemins de données : les additionneurs et les multiplieurs. Comme les circuits micropipelines utilisent le même codage binaire que les circuits synchrones, ces circuits peuvent utiliser les mêmes outils pour la synthèse de leurs chemins de données. Donc, les circuits micropipelines ne présentent aucun intérêt ici vu l'énorme travail déjà réalisé dans le domaine synchrone. Ainsi, seulement la classe des circuits quasi insensibles aux délais est adressée dans cette thèse.

L'autre facteur important est l'existence de plusieurs types de données dans la langage CHP, c'est-à-dire, que plusieurs codages peuvent être utilisés pour les signaux de données. De ce fait, il est nécessaire de concevoir un chemin de données et tous ses opérateurs dans le codage qui a été choisi par le concepteur afin d'éviter au maximum l'insertion des interfaces de conversion des types. Ces interfaces peuvent dégrader la performance des circuits générés et leur implémentation représente un coût additionnel en surface et en consommation.

Malheureusement, tous les résultats de l'étude des deux opérateurs arithmétiques ne peuvent pas être insérés dans les méthodologies plus générales de synthèse logique qui sont au cœur des moteurs de synthèse TAST. Ainsi, la génération des opérateurs arithmétiques s'est fait par l'appel d'un module développé au cours de cette thèse (voir Figure III.1) et attaché au moteur de synthèse QDI.

Cependant, certains des résultats de cette thèse ont pu être généralisés. Les conclusions obtenues ici, et qui permettent d'optimiser les circuits QDI, peuvent, par la suite, être intégrées dans les moteurs de synthèse afin d'améliorer les résultats de ces outils.

Enfin, cette thèse propose une méthodologie de comparaison rapide des circuits asynchrones QDI. La comparaison des différentes solutions a été un élément important au cours de la réalisation de ce travail et dans l'absence d'une méthodologie adaptée, une métrique permettant de comparer les différents aspects de l'implémentation d'un circuit – surface, énergie consommée et retard – a été élaborée.

Dans les chapitres suivants, la métrique qui permet de comparer les différentes solutions et son évaluation est d'abord présentée pour légitimer les résultats exposés dans les autres chapitres de cette thèse. Ensuite, les résultats plus généraux sont discutés avant la présentation des études sur les opérateurs arithmétiques.

III.5.Références

[III.1] A. Dinh-Duc, L. Fesquet, M. Renaudin, "Synthesis of QDI Asynchronous Circuits from DTL-style Petri-Net" IWLS-02, 11th IEEE/ACM Internat. Workshop on Logic & Synthesis, New Orleans, Louisiana, June 4-7, 2002.

[III.2] M. Renaudin, J.B. Rigaud, A. Dinh-Duc, A. Rezzag, A. Sirianni, J. Fragoso : "TAST CAD Tools", ASYNC'02 TUTORIAL, ISRN: TIMA--RR-02/04/01—FR, 2002.

[III.3] A.J. Martin, "Programming in VLSI: from communicating processes to delay-insensitive circuits", in C.A.R. Hoare, editor, Developments in Concurrency and Communication, UT Year of Programming Series, 1990, Addison-Wesley, p. 1-64.

[III.4] M. Renaudin, P. Vivet, F. Robin, "ASPRO-216 : a standard-cell Q.D.I. 16-bit RISC asynchronous microprocessor", Proc. of the Fourth International Symposium on

Advanced Research in Asynchronous Circuits and Systems (ASYNC'98), San Diego - USA, 1998, p. 22-31.

[III.5] A. Abrial, J. Bouvier, M. Renaudin, P. Senn and P. Vivet, "A New Contactless Smart Card IC using On-Chip Antenna and Asynchronous Microcontroller", Journal of Solid-State Circuits, Vol. 36, 2001, pp. 1101-1107.

[III.6] G.F. Bouesse, M. Renaudin, A. Witon, F. Germain, "A Clock-less low-voltage AES crypto-processor", European Solid State CIRcuits Conference, ESSCIRC 2005, Grenoble, September 12-16, 2005.

[III.7] D. Borrione, M. Boubekeur, L. Mounier, M. Renaudin, A. Sirianni "Modeling CHP descriptions in Labeled Transitions Systems for an efficient formal validation of asynchronous circuit specifications". In Proc FDL'03, Frankfort, Germany, 23-26 September 2003.

[III.8] A. Dinh-Duc. "Synthèse Automatique de Circuits Asynchrones QDI". PhD Thesis. INP de Grenoble, 2003.

[III.9] K. Yun, P. Beerel, J. Arceo, "High Performance Asynchronous Pipeline", In "International Symposium on Advanced Research in Asynchronous Circuits and Systems", ASYNC'96 , Aizu-Wakamatsu, Japan, 18 - 21 April, pp 17-28, 1996.

[III.10] A.J. Martin, "Synthesis of Asynchronous VLSI Circuits", Internal Report, Caltech-CS_TR-93-28, California Institute of Technology, Pasadena, 1993.

[III.11] I.E. Sutherland, "Micropipelines", Communication of the ACM, Volume 32, N°6, June 1989.

[III.12] A. Rezzag. "Synthèse Logique de Circuits Asynchrones Micropipeline". PhD Thesis. INP de Grenoble, 2004.

[III.13] P. Vivet. "Une méthodologie de conception de circuits intégrés quasi-insensibles aux délais: application à l'étude et à la réalisation d'un processeur RISC 16-bit asynchrone". PhD Thesis. INP de Grenoble, 2001.

Chapitre IV

Modèles d'Énergie, Retard et Surface

Cette thèse ambitionne de proposer une méthode pour la conception des chemins de données asynchrones. Donc, avant de continuer, il est important d'établir une métrique d'appréciation permettant, initialement, de comparer les différentes solutions possibles et de confronter les solutions choisies à d'autres solutions existantes. Une métrique de qualité doit couvrir les principaux critères de conception des circuits intégrés : consommation, vitesse ou encore, la surface nécessaire à l'implémentation. Évidemment, pour des questions de coûts et de délais, il est impossible de concevoir, fabriquer et tester l'ensemble des solutions envisagées. En conséquence, les simulations et évaluations des circuits au travers des modèles sont nécessaires.

La littérature propose de nombreux modèles permettant d'évaluer les circuits intégrés. Chaque modèle est applicable à un certain niveau d'abstraction des circuits intégrés et chaque modèle comporte des approximations et simplifications, soit relatives au propre modèle, soit inhérentes au niveau d'abstraction. Dans le cadre de ce travail, il a été choisi de représenter les circuits par des réseaux de portes logiques. À ce niveau d'abstraction, plusieurs facteurs technologiques et physiques liés a l'implémentation finale d'un circuit ne peuvent être considérés. En contrepartie, l'omission des aspects physiques permet d'adresser une complexité supérieure aux niveaux d'abstraction plus proche de l'implémentation réelle. Ainsi, la métrique doit être applicable à une description d'un circuit par un réseau de portes logiques.

Il est important de souligner que l'objectif recherché n'est pas de définir une métrique avec le minimum d'erreur, mais d'établir un moyen équitable et juste de comparer différentes implémentations d'une fonction donnée.

IV.1. La Surface

La surface en silicium n'est pas un facteur négligeable vu qu'elle impacte directement le coût de production et la fiabilité d'un circuit. Plus la surface est grande plus le coût est augmenté et plus le risque d'erreur dans la fabrication est aggravé. Cependant, l'implémentation des circuits asynchrones demande encore plus de surface que les versions synchrones (contrainte d'insensibilité au délai – cf. §II.4.3). En conséquence, il ne s'agit pas ici à nouveau d'estimer précisément la surface, mais seulement de la comparer de façon juste.

En ayant une description d'un réseau de portes logiques, la première idée est d'utiliser le nombre de portes contenues dans la description. Deux problèmes invalident cet estimateur. Tout d'abord, le simple comptage des portes ne prend pas en compte la complexité de chaque porte. En effet, deux réseaux contenant le même nombre de portes peuvent être largement différent en complexité, et, donc, en surface d'implémentation. La Figure IV.1 montre deux possibles réseaux qui contiennent le même nombre de portes logiques.

Ensuite, il est souvent impossible de connaître le nombre de portes logiques existantes dans les solutions présentées dans la littérature.

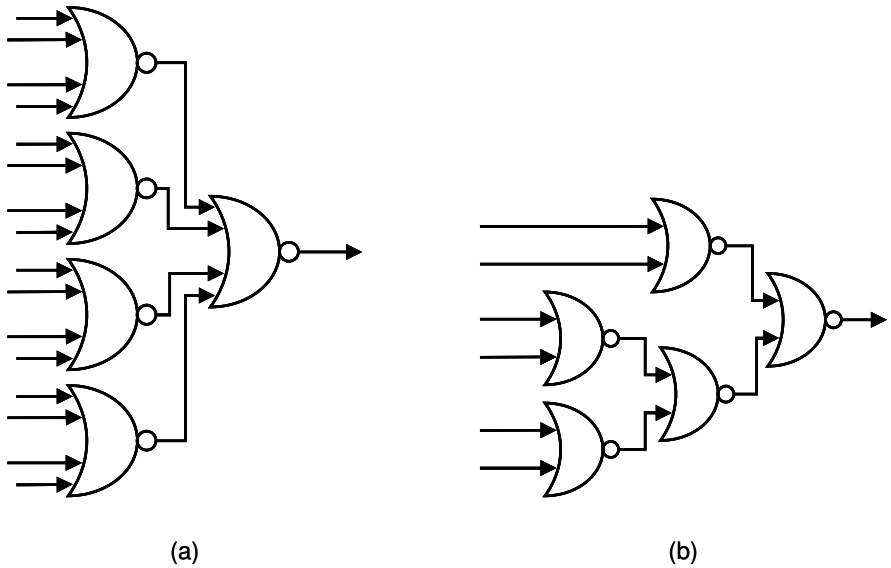


Figure IV.1 – Deux réseaux de portes logiques contenant cinq portes logiques.

Cependant, il est possible de capturer la complexité d'une porte en déterminant le nombre minimum de transistors nécessaires à son implémentation. En conséquence, il est facile de définir le nombre total de transistors nécessaire à l'implémentation d'un réseau de portes logiques donné. De nouveau, le nombre de transistors n'exprime pas la complexité de la fonction logique, mais la complexité du réseau de portes. En supposant une densité constante

de transistors par millimètre carré [xtors/mm^2] pour une certaine technologie et une méthodologie utilisées, la surface est proportionnelle au nombre de transistors.

Il est vrai que la densité de transistors pour différents circuits n'est pas constante, mais le nombre de transistors possède une plus forte corrélation avec la surface que le nombre de portes d'un réseau, particulièrement si le nombre de portes distinctes dans le réseau est grand. Le Tableau IV.1 présente des résultats de synthèse de plusieurs circuits très souvent présents dans les chemins de donnés. La densité en transistors par millimètre carré présente une plus petite variation que la densité de portes logiques (cellules). Donc, le nombre de transistors peut être utilisé comme une première approximation de la surface des circuits.

Tableau IV.1 – Résultats de synthèse de circuits utilisant la technologie HCMOS9 et les outils CADENCE®.

Circuits	Nombre de Transistors	Nombre de Cellules	Surface (μm^2)	Densité de Transistors (K/mm^2)	Densité de Cellules (K/mm^2)
Comparateur 32-bit	**758**	**120**	**11216**	**67,58**	**13,37**
Égalité 2 valeurs 64-bit	810	86	11079	73,11	9,70
Sklanski Adder 32-bit	**1020**	**126**	**15060**	**67,73**	**10,46**
Égalité 3 valeurs 64-bit	1620	171	22136	73,18	9,66
Égalité 2 valeurs 128-bit	3178	538	49140	64,67	13,69
Booth 16-bit	6486	675	87110	74,46	9,69
Mult 16-bit	8312	947	110975	74,90	10,67
Mult 32-bit	34536	3608	452748	76,28	9,96
			Moyenne (écart type)	71,49 (4,22)	10,90 (1,67)

IV.2. L'Énergie

L'énergie consommée, ou dissipée sous forme de chaleur, pour un circuit est donnée par :

$$E = V_{dd} \cdot \int i(t) \partial t \qquad \text{(IV-1)}$$

où V_{dd} est la tension, ici supposée constante, et $i(t)$ le courant dans la source d'énergie. L'énergie dissipée dans les circuits CMOS provient principalement des trois sources suivantes : les courants dynamiques, les courants de court-circuit et les courants de fuite (« leakage »). L'énergie peut être réécrite comme étant :

$$E = E_{dynamique} + E_{court-circuit} + E_{fuite} \qquad \text{(IV-2)}$$

Le courant de court-circuit est présent dans le circuit quand il y a un chemin direct entre VDD et GND. Dans un inverseur CMOS statique, il n'y a pas de court-circuit en l'absence de

transitions des entrées. Cependant, durant le changement des valeurs d'entrée, il existe un court temps pendant lequel les deux transistors, NMOS et PMOS, sont actifs simultanément, si $V_{dd} > V_{Tn} + |V_{Tp}|$, où V_{Tn} et V_{Tp} sont, respectivement, les tensions de seuil du transistor NMOS et PMOS. Donc, tant que $V_{Tn} < V_i < V_{dd} - |V_{Tp}|$ (V_i est la tension du signal d'entrée), il existe un courant de court-circuit.

Plusieurs travaux ont été faits pour évaluer la contribution du courant de court-circuit dans l'ensemble de l'énergie dissipée. Ces travaux montrent que l'énergie de court-circuit est d'environ 1% de toute l'énergie consommée [IV.1][IV.2][IV.3][IV.4]. Basé sur ces résultats l'énergie de court-circuit sera, pour l'instant, négligée ($E_{court-circuit} = 0$).

Les circuits CMOS connaissent deux types de courants de fuite : (1) courant sous le seuil dans les canaux des dispositifs bloqués (« off devices ») et (2) courant de drains des diodes en polarisation inverse (« reverse-bias diode leakage on transistor drains »). Les amplitudes de ces deux courants sont déterminées pour divers aspects de la technologie de fabrication. L'amplitude du courant sous le seuil est dépendante des dimensions des dispositifs et elle est proportionnelle à $e^{\frac{q(V_{gs}-V_T)}{kT}}$ où V_{gs} est la tension grille→source, q est la charge d'électron, k est la constante de Boltzmann et T est la température [IV.5]. Le courant de fuite de la diode (i_o) se produit quand un transistor est éteint et un autre transistor actif charge (ou décharge) le drain du premier transistor. L'amplitude de ce courant est définie par :

$$i_o = i_s(e^{\frac{qV}{kT}} - 1) \qquad \text{(IV-3)}$$

où i_s est le courant de saturation inverse et V est la tension de diode [IV.6].

Avec l'évolution des technologies de fabrication et l'apparition de dispositifs aux dimensions inférieures à 1µm, les courants de fuites augmentent leur contribution dans la somme des énergies dissipées, principalement dans les technologies avec tensions réduites de seuil pour la conception de dispositifs à très grande vitesse. Par contre, dans les technologies pour la faible consommation qui gardent une relative haute tension de seuil, les courants de fuite peuvent encore être négligées. Donc, pour une première approximation, l'énergie découlant de courants de fuites sera aussi négligée ($E_{fuite} = 0$).

Il reste l'énergie dynamique et différents travaux ont montré que l'énergie dynamique est la principale responsable pour l'énergie dissipée dans une vaste classe de circuits CMOS [IV.7] [IV.8] [IV.9] [IV.10] [IV.11] [IV.12].

L'énergie dynamique est la conséquence des courants de charge et décharge des capacités des nœuds d'un circuit CMOS au cours de son fonctionnement. Cette énergie peut être exprimée par :

$$E_{dynamique} = \frac{1}{2} V_{dd}^2 \sum_{i=1}^{k} C_i n_i \qquad \text{(IV-4)}$$

où V_{dd} est la tension d'alimentation, k est le nombre de nœuds du circuit, C_i est la capacité totale du nœud i et n_i est le nombre de transitions (0→1 ou 1→0) sur le nœud i pendant la période de mesure.

La Figure IV.2 montre une représentation des capacités d'un transistor MOS. Dans ce modèle et dans les analyses suivantes, la superposition de la grille sur les régions de drain et source est supposée nulle. Cette simplification est valide pour un auto alignement du premier ordre dans le processus de fabrication. Dans la Figure IV.2, les capacités suivantes sont indentifiables :

- ***C_{gs}, C_{gd}*** : capacités entre la grille et le canal qui sont connectées, respectivement, aux régions de source et drain du canal.
- ***C_{sb}, C_{db}*** : capacités entre les diffusions de source et drain et le substrat (du anglais « bulk »).
- ***C_{gb}*** : capacité entre la grille et le substrat.

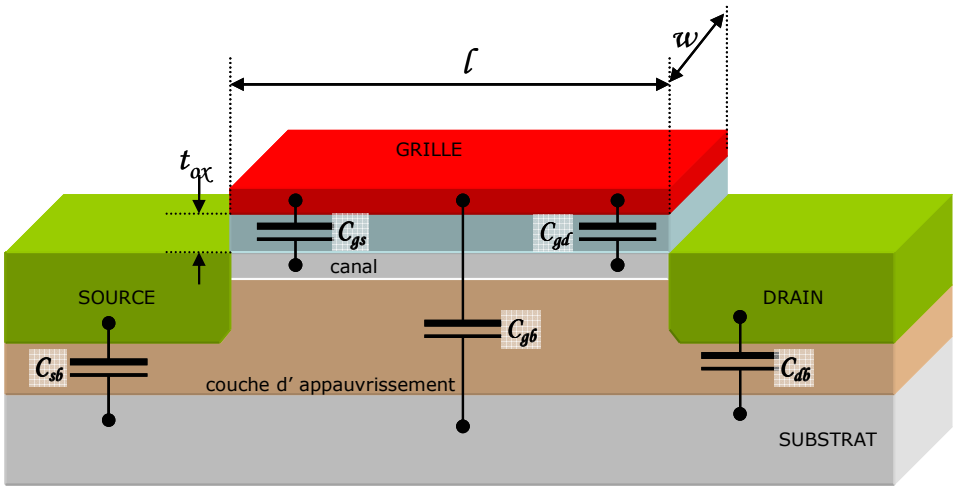


Figure IV.2 – Représentations des capacités parasites d'un transistor MOS.

La Figure IV.3 présente le circuit équivalent pour ce modèle d'un transistor MOS avec ses capacités parasites. La capacité totale de la grille est définie par la somme de ses trois

capacités : $C_g = C_{gb} + C_{gs} + C_{gd}$. La capacité C_g est fonction des dimensions du transistor et des paramètres technologiques et est calculée par :

$$C_g = \frac{\varepsilon}{t_{ox}} \cdot l \cdot w \qquad \text{(IV-5)}$$

où ε est la constante diélectrique de l'isolant, t_{ox} est l'épaisseur de l'isolant et l et w les dimensions de la grille.

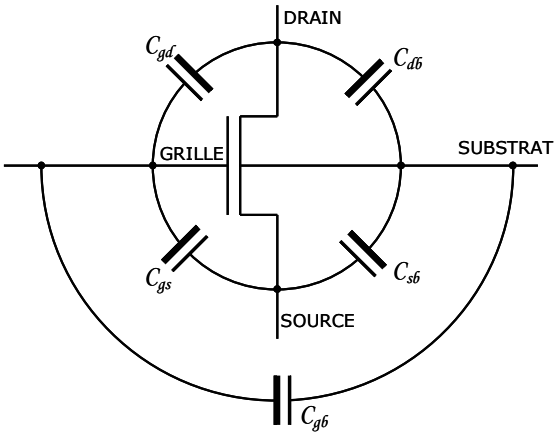


Figure IV.3 – Le circuit équivalent avec les capacités parasites.

À partir de ce modèle équivalent pour le transistor, il est possible de spécifier les capacités connectées à un nœud d'un circuit CMOS au niveau portes logiques. Trois capacités distinctes participent à la charge d'un nœud :

- ➔ C_d : les capacités des drains connectés au nœud piloté par la porte logique qui le pilote.
- ➔ C_w : la capacité relative à la connexion (« wiring ») du nœud piloté par la porte logique.
- ➔ C_g : les capacités de grille de toutes les portes logiques qui sont pilotées pour le nœud.

La Figure IV.4 montre le schéma électrique et le réseau de portes logiques d'une fraction d'un circuit CMOS avec les capacités correspondantes à la connexion entre les deux portes. Malheureusement, au niveau de description du circuit par un réseau de portes logiques et en l'absence d'informations de routage physique, il est presque impossible d'estimer correctement la capacité C_w. En conséquence, la capacité de routage sera omise dans les évaluations et la capacité d'un nœud sera considérée comme étant, pour l'instant, simplement la somme des capacités de grille des transistors connectées au nœud étudié. La valeur de C_w pourra être ajoutée après l'étape de placement et routage.

Sans perte de généralité, tous les transistors seront supposés de longueur minimale permise pour une technologie donnée ($l_{\min}$). À part quelques cellules spéciales, toutes les cellules numériques possèdent la longueur minimale qui conduit à la réduction des capacités de grille et l'augmentation de la vitesse.

De plus, il sera adopté que les cellules sont dimensionnées pour avoir le même temps de monté et descente du signal de sortie. Cela implique que pour l'inverseur de la Figure IV.4 le transistor PMOS soit, approximativement, deux fois ($\gamma = 2$) la largeur du transistor NMOS pour compenser la différence de mobilité entre les porteurs des deux types de transistors.

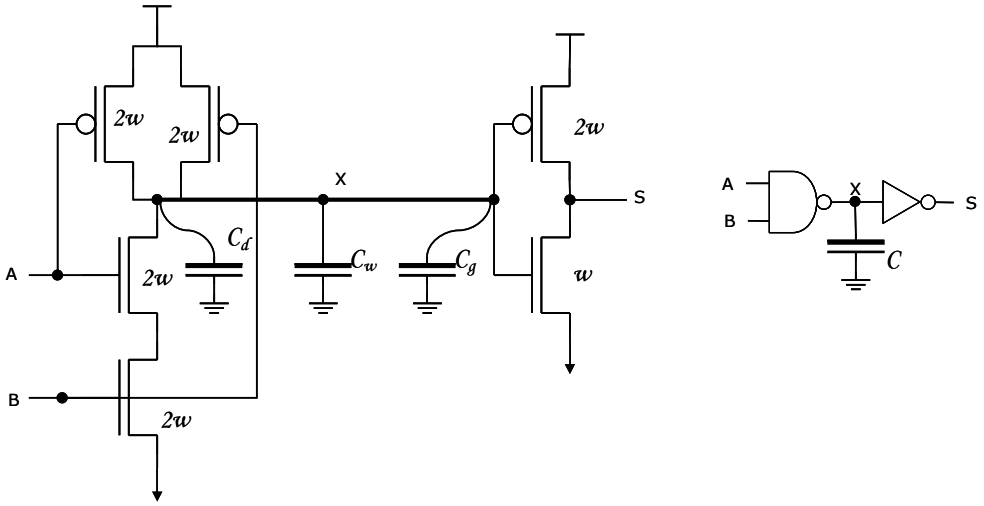


Figure IV.4 – Capacités pour un nœud.

Ainsi, la capacité de grille d'un inverseur type est égale à :

$$C_g^{INV} = (\gamma+1)\cdot\frac{\varepsilon}{t_{ox}}\cdot l_{\min}\cdot w = 3\cdot\frac{\varepsilon}{t_{ox}}\cdot l_{\min}\cdot w \qquad \text{(IV-6)}$$

et l'énergie nécessaire pour piloter l'entrée de l'inverseur est égale à :

$$E^{INV} = \frac{1}{2}V_{dd}^2 C_g^{INV} \qquad \text{(IV-7)}$$

Pour les autres portes logiques, il sera adopté qu'elles sont dimensionnées pour avoir la capacité de délivrer le même courant de sortie qu'un inverseur type. Ceci dit, les portes possédant des transistors connectés en série doivent ajuster la largeur de ces transistors pour contrebalancer l'augmentation de résistance. Pour les transistors en parallèle dans la cellule, le pire cas où seulement un transistor est saturé est considéré et, donc, ces transistors gardent les mêmes dimensions qu'un inverseur type. Dans ce cas, la porte NAND-2 (non-et à deux entrées) de la Figure IV.4 possède deux transistors NMOS en série de largeur $2\cdot w$ et les transistors en parallèle aussi de largeur $2\cdot w$. La capacité de chaque entrée est égale à :

$$C_g^{NAND-2} = 4 \cdot \frac{\varepsilon}{t_{ox}} \cdot l_{\min} \cdot w = \frac{4}{3} C_g^{INV} \quad \text{(IV-8)}$$

et l'énergie nécessaire pour piloter une entrée d'une porte NAND-2 est égale à :

$$E^{NAND-2} = \frac{1}{2} V_{dd}^2 C_g^{NAND-2} = \frac{4}{3} E^{INV} \quad \text{(IV-9)}$$

Donc, il est possible de traduire les capacités de grille et, par la suite, l'énergie pour piloter une entrée d'une porte logique en fonction de l'inverseur type de la technologie choisie. Le Tableau IV.2 rapporte l'énergie consommée pour piloter chaque entrée, et ceci pour différents types de portes logiques en fonction du nombre d'entrées.

Tableau IV.2 – Énergie consommée pour piloter chaque entrée de portes logiques normalisées en fonction de l'énergie nécessaire pour piloter l'entrée d'un inverseur type (E^{INV}).

	Nombre d'entrées				
Type de Porte	1	2	3	4	*N*
INV	1				
NAND		4/3	5/3	6/3	(*n*+2)/3
NOR		5/3	7/3	9/3	(2*n*+1)/3
C-ELEMENT		2	3	4	*N*

Actuellement, avec la continuelle réduction des dimensions des transistors, la contribution des énergies parasites devient de plus en plus importante, même si jusqu'à maintenant leur participation a été négligée. Donc, un modèle qui ne prend pas en compte la présence de ces éléments parasites dans le circuit perdra rapidement son intérêt. Malheureusement, l'énergie provenant des fuites et des courants de court-circuit aussi bien que les capacités de diffusions sont fortement dépendantes de la technologie cible choisie.

Comme il a été déjà dit, le but de cette thèse n'est pas de créer un nouveau modèle d'estimation de la consommation d'énergie pour les circuits CMOS, mais de choisir une métrique, entre les existantes, pour comparer différents circuits entre eux. Donc, les effets parasites d'une certaine technologie doivent être ajoutés au modèle. Pour cela, une capacité équivalente à la contribution totale parasite est ajoutée à la sortie de chaque porte. La valeur de cette capacité dépend, donc, de la porte et de la technologie.

Dans le contexte de cette thèse, la technologie HCMOS9 de ST-Microelectonics® sera utilisée. Plusieurs portes logiques ont été simulées pour déterminer les valeurs des capacités. L'énergie pour piloter leurs entrées et l'énergie parasite ont été évaluées. La Figure IV.5 montre le circuit de test utilisé pour évaluer une porte logique.

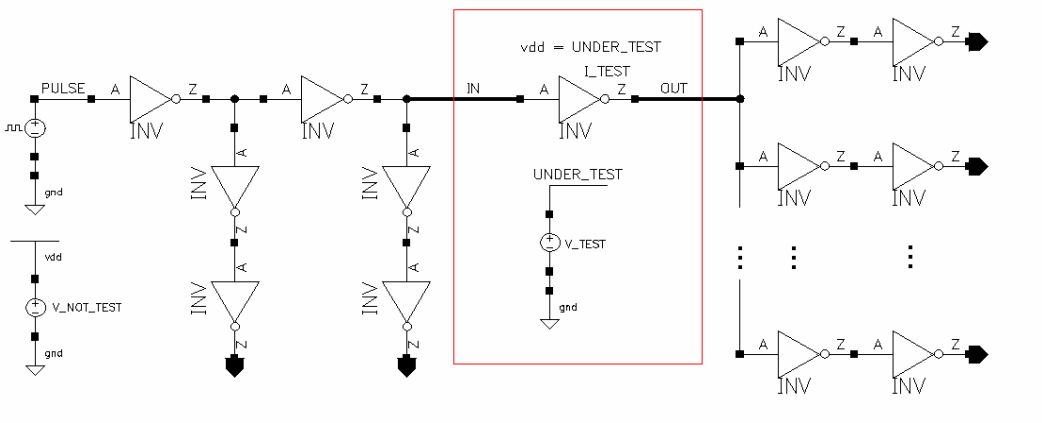


Figure IV.5 – Le circuit de test pour évaluer le comportement d'une porte logique.

Dans l'exemple de la Figure IV.5, un inverseur (instance « I_TEST ») est évalué pour plusieurs charges de sortie. La charge de sortie est augmentée en ajoutant des inverseurs similaires au nœud « OUT ». Les deux premiers étages servent à créer une forme d'onde plus réaliste à l'entrée de l'inverseur sous test. En mesurant l'énergie consommée pour l'inverseur « I_TEST », il est possible de déterminer l'énergie nécessaire pour piloter l'entrée d'un inverseur et l'énergie parasite du même inverseur.

La Figure IV.6 présente les résultats de consommation de l'inverseur « I_TEST » en fonction de la sortance (dans ce cas, le nombre de portes identiques connectées à la sortie). Par régression linaire, il est possible d'extraire la pente de la courbe qui correspond à l'énergie pour piloter un inverseur et l'ordonnée à l'origine qui correspond à tous les effets parasites de l'inverseur (capacités de diffusions, courants de court-circuit, courants de fuites, etc.).

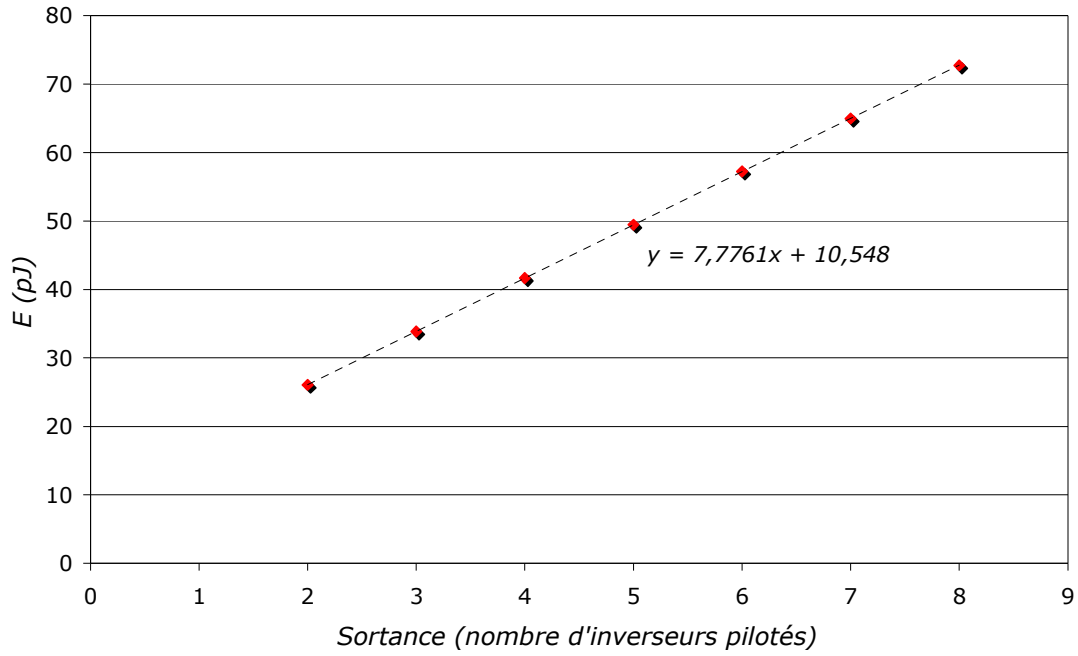


Figure IV.6 – Consommation d'un inverseur type en fonction de la sortance

Le même exercice a été réalisé pour plusieurs portes et les résultats de consommation ont été normalisés en fonction d'un inverseur type. Les simulations ont été réalisées en utilisant le V_{dd} typique de la technologie et avec des signaux d'entrées dans le domaine de rampes rapides. Le Tableau IV.3 compile ces résultats pour quelques portes.

A partir du Tableau IV.3, il est possible de vérifier que l'énergie pour piloter une porte logique donnée est proportionnelle à la capacité d'entrée de cette porte et que cette capacité est, avec une bonne approximation, proportionnelle aux largeurs de transistors.

Tableau IV.3 – Résultats de simulation de l'énergie consommée pour piloter chaque entrée de portes logiques normalisées en fonction de la énergie nécessaire pour piloter l'entrée d'un inverseur type (E^{INV}).

	Énergie			
Type de Porte	Entrée		Parasite	Rapport Parasite/Entrée
	Modèle (Tableau IV.2)	Simulation		
INV	1	1,00 (référence)	1,36	1,36
NAND-2	4/3	1,35	1,95	1,45
NAND-3	5/3	1,69	2,57	1,52
NAND-4	2	2,03	3,21	1,58
NOR-2	5/3	1,64	2,55	1,55
NOR-3	7/3	2,28	3,70	1,62
NOR-4	3	2,92	4,82	1,65

Aussi, il est possible de mettre en évidence la contribution des énergies parasites dans la technologie choisie. Les résultats démontrent que l'énergie parasite d'une porte est, en une première approximation, équivalent à 150% de l'énergie moyenne nécessaire pour piloter une entrée de la même porte. Donc, il est évident qu'il est impossible de négliger les effets parasites (capacités de diffusions, courants de court-circuit, etc.) pour les technologies actuelles.

Pour améliorer le modèle proposé, une capacité est ajoutée à la sortie de chaque porte. La valeur de cette capacité sera égale à 1,5 fois (une fois et demie) la valeur moyenne de la capacité d'entrée de la porte. Cette capacité prend, donc, en charge tous les effets parasites liés à la commutation d'une porte. La Figure IV.7 montre les modèles de capacités pour les trois portes principales avec les valeurs de chaque capacité en fonction de la capacité de l'inverseur type de la technologie.

Finalement, la porte de « Müller » (C-Element) possède un élément de mémorisation à la sortie. L'élément de mémorisation consiste en deux inverseurs connectés en anneau (Figure IV.8). Pour la porte de Müller, il est adopté que l'inverseur de sortie est dimensionné comme l'inverseur type et que l'inverseur de retour est beaucoup plus petit pouvant être négligé. Chaque fois que la sortie de la porte de Muller commute, l'inverseur de sortie commutera

aussi. En conséquence, les nœuds de sortie des portes de Müller ont leur énergie augmentée d'un inverseur type (entrée + parasite). La Figure IV.8 présente le schéma de la porte de Muller et le modèle équivalent avec les capacités d'entrées et sortie.

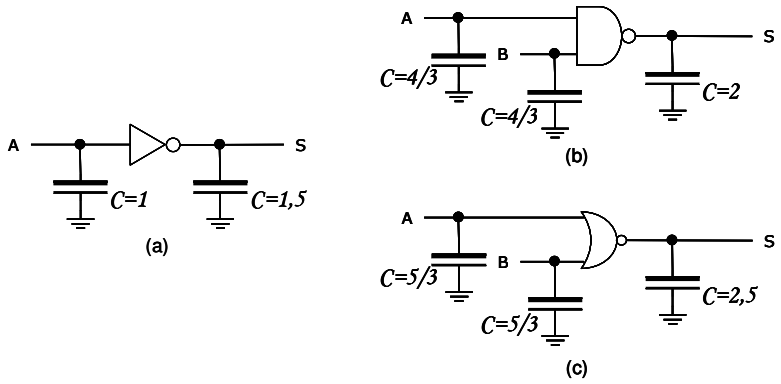


Figure IV.7 – Modèle pour les capacités des portes logiques : (a) inverseur, (b) NAND-2 et (c) NOR-2.

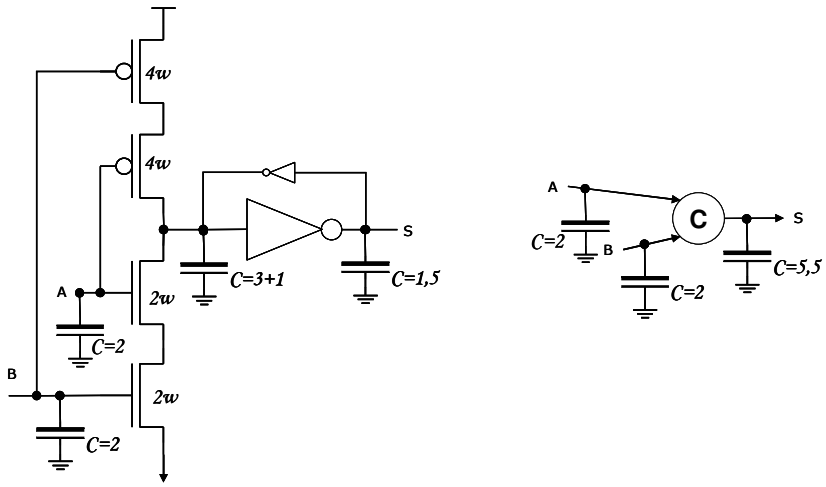


Figure IV.8 – La porte de Müller statique avec l'élément de mémorisation à la sortie et les capacités pour chaque nœud.

Pour les chemins de données asynchrones, l'intérêt est de pouvoir déterminer l'**énergie moyenne** consommée pour réaliser une opération (un cycle du protocole, cf. §II.2). L'énergie moyenne pour une opération peut-être calculée par l'énergie totale consommée divisée par le nombre d'opérations effectuées. Cette formulation est aussi valable pour calculer la consommation moyenne d'un nœud du réseau de portes.

Par les contraintes imposées aux circuits asynchrones, toutes les transitions d'un nœud d'un circuit doivent être observables à la sortie du circuit et la logique ne contient pas d'aléas. Donc, pour effectuer une opération chaque nœud réalise, au maximum, une seule transition.

En supposant une seule transition par nœud et par opération, et la réalisation de la totalité des opérations (intégralité des combinaisons des valeurs pour les signaux d'entrées), le

nombre de transitions d'un nœud pendant la totalité des opérations divisé par le nombre d'opérations réalisées est égal au nombre moyenne (α) des transitions d'un nœud. Si toutes les opérations sont équiprobables et qu'il y a une seule transition par nœud et par opération, alors $0 \le \alpha \le 1$.

$$\alpha_i = \frac{n_i}{N} \tag{IV-10}$$

oú n_i est le nombre de commutations du nœud i et N le nombre total d'opérations réalisées.

De façon équivalente, α peut être perçu comme étant la probabilité pour un nœud de changer de valeur au cours d'une opération. Si un nœud possède α=1, cela implique que le nœud commute dans toutes les opérations. Si α=0, cela implique qu'un nœud ne commute jamais pour aucune opération. Finalement si, par exemple, un nœud possède α=0,5, cela implique que le nœud commute dans la moitié des opérations réalisées pour le circuit.

En utilisant cette affirmation, l'énergie moyenne d'un nœud i peut être exprimée comme étant :

$$E_i^{MOYENNE} = \frac{1}{2}\alpha C_i V_{dd}^2 = \underbrace{\alpha \frac{C_i}{C_g^{INV}} E^{INV}}_{\text{normalisé au inverseur type}} \tag{IV-11}$$

où α est la probabilité d'activité sur le nœud et C_i est la capacité totale du nœud.

L'énergie moyenne du circuit peut-être calculée en additionnant l'énergie moyenne de tous les nœuds du circuit. En conséquence :

$$E^{MOYENNE} = \left(\sum_{i=1}^{k} \alpha_i \frac{C_i}{C_g^{INV}} \right) E^{INV} \tag{IV-12}$$

Par contre, pour les protocoles avec retour à zéro, les nœuds qui ont changé de valeur (0→1) doivent faire leur remise à zéro et, en conséquence, une deuxième transition est accumulée sur le nœud. L'énergie moyenne doit être multipliée par deux pour ces protocoles si une comparaison entre les protocoles est envisagée.

Si la probabilité de commuter des signaux d'entrées est connue, il n'y a pas besoin de simuler un circuit pour connaître la probabilité de commutation d'un nœud donné. Il est possible, en prenant comme base de simples formules de probabilité, de retrouver la probabilité que la sortie d'une porte logique change de valeur à partir des probabilités des signaux d'entrée. Les propriétés des circuits QDI asynchrones favorisent la détermination des

probabilités vu que, pour chaque opération, chaque nœud du circuit devra commuter au maximum une seule fois.

Un buffer (ou inverseur) propage la valeur d'entrée à la sortie, donc la sortie du buffer possède la même probabilité de commuter que son entrée.

Une porte ET applique aussi la fonction « et » à la probabilités de ses entrées. En supposant les événements indépendants sur les entrées, transitions 0→1 et probabilités $\rho(A)$ et $\rho(B)$ pour les entrées, la probabilité pour la sortie « S » de commuter est donnée par : $\rho(S) = \rho(A) \cdot \rho(B)$. La probabilité de sortie traduit le fait qu'il faut avoir tous les signaux d'entrées égaux à 1 pour que la sortie soit égale à 1. La Figure IV.9 montre un exemple de calcul de probabilités pour un inverseur et une porte ET. Si les événements sont mutuellement exclusifs, i.e. l'occurrence de transition d'une entrée empêche la transition de l'autre entrée, la probabilité de la sortie est nulle vu qu'il n'est pas possible d'avoir les deux entrées égales à 1 au même instant.

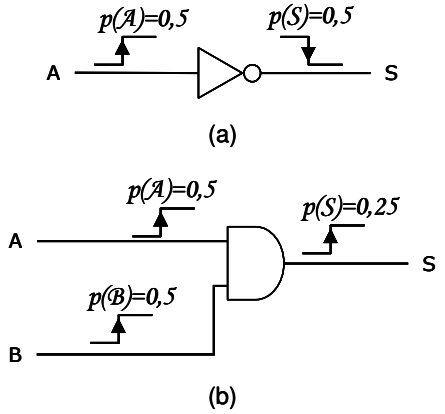


Figure IV.9 – Exemples de calcule de probabilités pour l'inverseur et la porte ET.

De façon duale, une porte OU applique la fonction « ou » aux probabilité des entrées. En supposant les événements mutuellement exclusifs, transitions 0→1, probabilités $\rho(A)$ et $\rho(B)$ pour les entrées, la probabilité pour la sortie « S » d'une porte OU de commuter est donnée par : $\rho(S) = \rho(A) + \rho(B)$. La probabilité de sortie décrit le fait qu'il suffit d'avoir A ou B pour que la sortie soit égale à 1. Si les signaux d'entrées sont indépendants, la probabilité de la sortie est définie comme étant: $\rho(S) = \rho(A) + \rho(B) - \rho(A) \cdot \rho(B)$.

Le Tableau IV.4 synthétise les calculs de probabilités pour les sorties des principales portes. Les configurations de portes plus compliquées, notamment portes complexes, peuvent être réduites à ces simples portes pour en retrouver les probabilités de la sortie.

Tableau IV.4 – Probabilité pour que la sortie d'une porte commute calculée en utilisant les probabilités de commutation de ses entrées.

	Événements des entrées			
	Mutuellement Exclusifs		Indépendants	
Type de Porte	0→1	1→0	0→1	1→0
INV	ρ(A)			
ET, NON-ET	Nulle	ρ(A)+ρ(B)	ρ(A)·ρ(B)	ρ(A)+ρ(B)-ρ(A)·ρ(B)
OU, NON-OU	ρ(A)+ρ(B)	Nulle	ρ(A)+ρ(B)-ρ(A)·ρ(B)	ρ(A)·ρ(B)
C-ELEM	Nulle	Nulle	ρ(A)·ρ(B)	ρ(A)·ρ(B)

De façon générale, les transistors en série ont besoin de tous les événements des entrées et, donc, la probabilité de la sortie est égale au produit des probabilités des entrées. Les transistors en parallèle ont besoin d'un seul événement d'entrée et, en conséquence, la probabilité de la sortie est égale à la somme des probabilités des entrées.

En sachant comment calculer la probabilité pour que la sortie d'une porte commute, les probabilités de commutation pour chaque nœud d'un circuit peuvent être retrouvées en propageant les probabilités des entrées du circuit. Dans le contexte de cette thèse, toutes les entrées sont supposées équiprobables pour le calcul de l'énergie moyenne. Cependant, la méthode proposée ici est générale et elle peut être appliquée à une quelconque distribution de probabilités des signaux d'entrée d'un circuit.

La Figure IV.10 et la Figure IV.11 montrent comment la méthode peut être appliquée pour retrouver la consommation moyenne d'un circuit. La Figure IV.10 présente une implémentation basée sur les minterms (DIMS) pour la fonction « ET » à deux entrées double rail asynchrone en utilisant le protocole quatre phases. Pour chaque nœud de la figure, il est présenté un couple ($\left[\alpha_i ; C_i / C_g^{INV} \right]$) de valeurs qui indiquent, respectivement, la probabilité de commutation du nœud et la capacité du nœud en fonction de la capacité d'un inverseur type.

L'énergie moyenne d'une opération pour l'implémentation DIMS est donnée par l'addition de tous les nœuds (équation IV-12) et elle est égale à 38,5 fois l'énergie pour piloter un inverseur. Comme l'implémentation de la Figure IV.10 utilise le protocole quatre phase, l'énergie moyenne doit encore être multipliée par deux pour prendre en charge la remise à zéro des nœuds.

De façon analogue, la Figure IV.11 propose une autre implémentation pour la même fonction « ET » à deux entrées avec les probabilités de commuter et les capacités en fonction d'un inverseur type pour chaque nœud du circuit. L'énergie moyenne pour la deuxième

architecture (addition de tous les nœuds) est égale à 54,38 fois l'énergie consommée pour piloter l'inverseur de référence. Donc, pour les tailles minima de portes logiques le circuit de la Figure IV.11 est plus consommant en moyenne par opération que l'implémentation DIMS de la Figure IV.10.

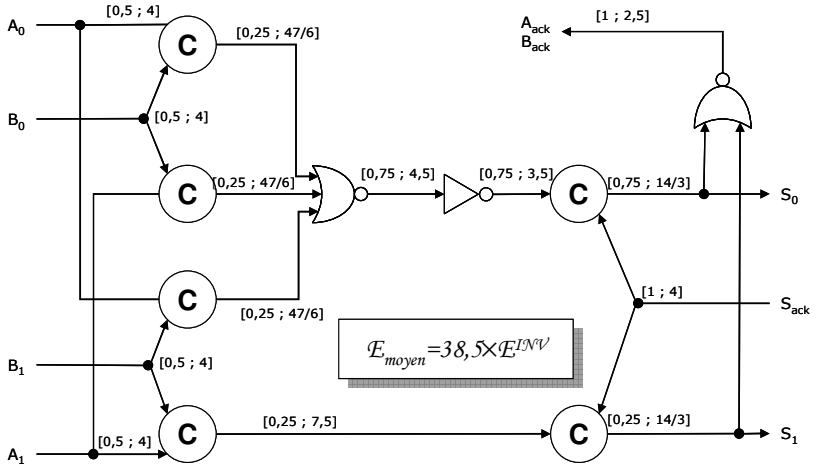


Figure IV.10 – Implémentation DIMS d'une porte AND-2 double rail avec la probabilité et la capacité en fonction d'un inverseur type pour chaque nœud.

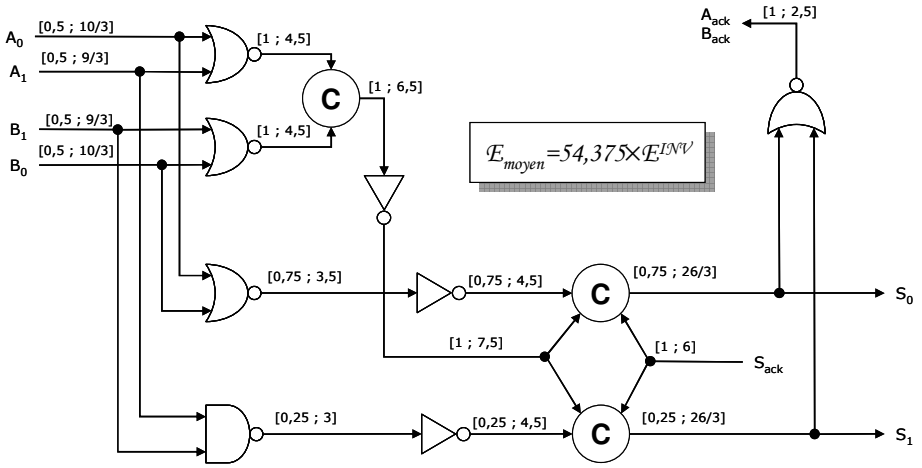


Figure IV.11 – Une autre implémentation d'une porte AND-2 double rail avec la probabilité de commuter et la capacité en fonction d'un inverseur type pour chaque nœud.

IV.3. Le Temps de Propagation

Le retard d'un signal dans un circuit est dû a deux facteurs principaux : (i) le retard du signal (front d'onde) pour se propager dans les fils et (ii) le retard des portes logiques pour commuter, i.e., le temps nécessaire aux portes pour charger/décharger les capacités connectées à sa sortie. À nouveau, la décision d'utiliser un réseau de portes logiques comme abstraction ne permet pas d'avoir une estimation de la longueur des fils, donc les retards imposés par le routage sont ignorés. Ces valeurs peuvent être ajoutées par la suite après l'étape de placement et routage.

En [IV.13], un modèle pour estimer le retard d'une porte logique appelé « effort logique » est proposé. Le modèle est basé sur un simple modèle de retard au travers d'une unique porte logique CMOS. La théorie exposée en [IV.13] semble bien adaptée au problème vu que le modèle est une première approximation du retard d'une porte logique et ce modèle est étendu pour permettre d'estimer le retard d'un chemin logique. Donc, par la suite, il sera montre que l'effort logique peut être utilisé pour estimer le retard d'un réseau de portes logiques.

Le modèle d'effort logique décrit le retard d'une porte logique comme l'effet de la charge capacitive que la porte logique pilote, mais aussi comme l'effet de la complexité de la porte. Plus la charge de sortie est importante, plus la porte nécessite de temps pour commuter, mais le retard est aussi dépendant de la fonction logique implémentée par la porte.

L'inverseur, la plus simple des portes, est le meilleur pilote des charges et est normalement choisi comme amplificateur pour piloter de grandes charges. D'autres fonctions logiques requièrent des portes plus complexes avec plus de transistors. Certains de ces transistors sont connectés en série faisant ainsi de ces portes logiques des pilotes de charge moins performants qu'un inverseur. En conséquence, une porte NAND (« non et ») présente plus de retard qu'un inverseur avec la même charge et les mêmes dimensions pour leurs transistors.

Le modèle d'effort logique exprime le retard comme une variable d sans unité. Le retard absolu d'une porte (d_{abs})est donné par :

$$d_{abs} = d\tau \qquad \text{(IV-13)}$$

où τ est le retard d'un inverseur idéal (sans capacités parasites) qui pilote un autre inverseur identique. Dans le cadre de cette thèse, il est suffisant de calculer d pour comparer plusieurs architectures. A partir d'ici, le mot retard fait référence à la variable d.

La variable d est divisée en deux parties: une partie fixe p appelée le « retard parasite » et une partie f proportionnelle à la charge appelée « effort de l'étage ». En conséquence, le retard total est donné par :

$$d = f + p \qquad \text{(IV-14)}$$

L'effort de l'étage (f) dépend des charges et de la capacité de la porte à piloter ces charges ($f = gh$). L'effort de l'étage est le produit entre l'effort logique (g) et l'effort électrique (h). L'effort électrique décrit comment l'environnement électrique de la porte

affecte sa performance et comment les dimensions des transistors déterminent la capacité de la porte à piloter ses charges. L'effort électrique est défini par :

$$h = \frac{C_{out}}{C_{in}} \qquad \text{(IV-15)}$$

où C_{out} est la capacité pilotée par la porte logique et C_{in} est la capacité de grillé de l'entrée sur laquelle le retard est calculé. Il est important de souligner que C_{in} pour l'effort électrique n'est pas la capacité totale du nœud sur lequel l'entrée est connectée, mais simplement la capacité de grille.

L'effort logique capture comment la topologie de la porte agit sur l'aptitude de la porte à piloter les charges indépendamment des dimensions des transistors. L'effort logique est défini comme étant combien de fois pire qu'un inverseur est une porte logique pour produire le courant de sortie, en supposant que la porte et l'inverseur possèdent les mêmes capacités d'entrées.

N'importe quelle topologie qui exécute une fonction logique est moins capable de délivrer du courant à sa sortie qu'un inverseur avec les mêmes capacités d'entrée. Au minimum, une porte logique possède plus de transistors et donc, pour maintenir les mêmes capacités d'entrées qu'un inverseur, les dimensions des transistors sont, en moyenne, inférieures. Si la topologie requiert des transistors en parallèle, une estimation conservatrice de la performance de la porte suppose que seulement un des transistors conduit et, en conséquence, la porte n'est pas capable de produire à sa sortie le même courant qu'un inverseur avec les mêmes capacités d'entrée. Si la topologie requiert des transistors en série, il n'est pas possible de délivrer le même courant qu'un inverseur avec des capacités d'entrées identiques.

Ainsi, l'effort logique peut être défini comme étant le rapport entre la capacité d'entrée d'une porte et la capacité d'entrée d'un inverseur qui peuvent délivrer le même courant de sortie. En utilisant l'approximation de la section précédente pour les dimensions des transistors, il est possible de vérifier que le Tableau IV.2 est équivalent à l'effort logique de chaque porte [IV.13].

Malheureusement, le retard parasite n'est pas aussi facile à obtenir que l'effort logique. La principale contribution aux capacités parasites est la capacité des régions de diffusions des transistors connectés à la sortie de la porte logique. La valeur de la capacité de diffusion prend en compte la géométrie des transistors (surface et périmètre) et les paramètres de la technologie cible. En [IV.13], une approximation grossière est présentée en utilisant le ratio entre les largeurs de transistors de diffusion connectés à la sortie d'une porte et les largeurs des

transistors d'un inverseur type. Cette approximation est très imprécise, mais comme la méthode d'effort logique est utilisée, jusqu'à présent, simplement pour le dimensionnement des transistors, elle n'a pas besoin d'une réelle estimation du retard parasite. Finalement, la formulation du retard proposée par [IV.13] est définie par:

$$d_{abs} = (gh + p) \cdot \tau \qquad \text{(IV-16)}$$

Pour déterminer par simulation l'effort logique (g) et le retard parasite (p), le même circuit de test présenté dans la Figure IV.5 est utilisé. Dans ce cas, le retard de l'instance « I_TEST » est mesuré pour plusieurs charges de sortie. La charge de sortie est augmentée en additionnant des portes identiques à la sortie de la porte qui subit le test. La Figure IV.12 présente les résultats de simulations pour quelques portes logiques dans la technologie HCMOS9.

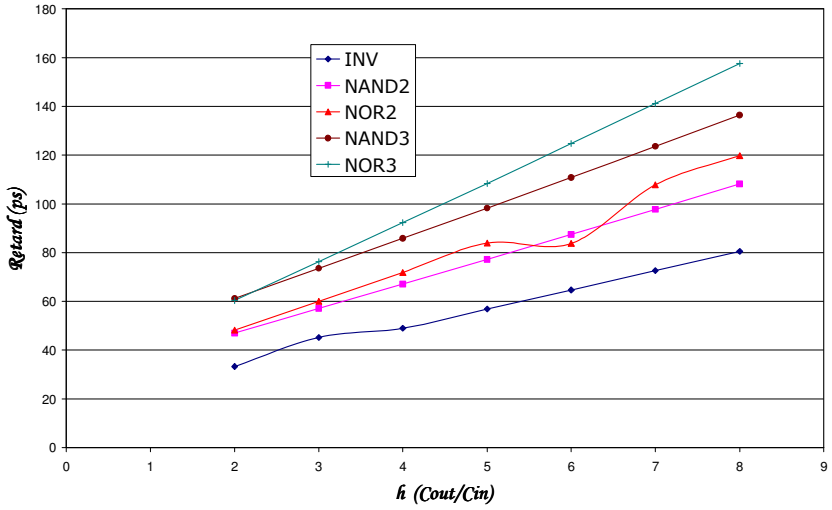


Figure IV.12 – Retard de plusieurs portes en fonction du rapport entre sortance et entrance.

A partir des courbes, il est possible de déduire les valeurs de l'effort logique et du retard parasite pour la technologie cible. Ces résultats sont résumés dans le Tableau IV.5 qui présente les valeurs normalisées par rapport à l'inverseur de référence.

Puisque l'estimation du retard parasite est très grossière et pour avoir une estimation plus précise, le retard parasite est défini comme étant égal à zéro. Le Tableau IV.5 montre que le rapport entre le retard parasite et l'effort logique est, en moyenne, égal à 1,49. Donc, il est possible d'ajouter une capacité à la sortie de la porte pour remplacer le retard parasite, vu que dans le modèle de l'effort logique, aucune charge n'est présente dans la sortie de la porte.

Ainsi, si des capacités sont ajoutées à la sortie de chaque porte comme à la Figure IV.7 et si le rapport C_{out}/C_{in} est égal à 1,5, ces capacités remplacent le retard parasite de la porte et le retard peut être défini simplement par :

$$d = gh \tag{IV-17}$$

où le rapport entre entrance et sortance (h) prend en compte cette capacité de sortie ajoutée. Cette formulation conduit à un résultat plus précis et le même modèle de capacités utilisées pour le calcul de l'énergie est utilisé pour le retard. Par ailleurs, les capacités provenant d'une étape de placement et routage ont la possibilité d'être insérées dans le modèle de retard sans aucune difficulté supplémentaire.

Tableau IV.5 – Résultats de simulation pour l'effort logique (*g*) et pour le retard parasite (*p*) en utilisant la technologie HCMOS9 de ST-Microelectronics.

Porte	Effort Logique (g)		Parasite (p)		Rapport
	Modèle (1)	Simulation	Modèle	Simulation (2)	(2) / (1)
INV	1,00 (ref)	1,00 (ref)	1,00	1,60	1,60
NAND2	1,33	1,28	2,00	2,05	1,54
NAND3	1,67	1,55	3,00	2,57	1,54
NAND4	2,00	1,83	4,00	3,02	1,51
NOR2	1,67	1,54	2,00	2,46	1,47
NOR3	2,33	2,14	3,00	3,32	1,42
NOR4	3,00	2,73	4,00	4,15	1,38

Pour déterminer le retard critique, le réseau de portes logiques est parcouru à partir de ses sorties primaires jusqu'à ses entrées. Pour un nœud i donné, f_i est définie comme étant la porte pilotant le nœud i, F_i est l'ensemble des portes f_j, tel que f_j est pilotée par le nœud i et $d(f_j)$ le retard de la porte f_j. Ainsi, le retard critique $d_i^{critique}$ d'un nœud i est défini comme étant :

$$d_i^{critique} = \max_j \left\{ d(f_j) + d_j^{critique} \,/\, f_j \in F_i \right\} \tag{IV-18}$$

Finalement, le retard critique du réseau est défini par :

$$d^{critique} = \max_j \left\{ d_j^{critique} \,/\, d_j \right\} \tag{IV-19}$$

La Figure IV.13 montre deux exemples de réseaux de portes où le retard de chaque chemin est calculé. Pour le réseau (a) de la Figure IV.13 le retard critique du nœud B_0 est égal à 12 et le retard du nœud B_1 est égal à 8. Finalement, le retard critique du réseau est donné par le maximum retard à ses entrées primaires. Donc, pour le réseau (a) de la Figure IV.13 le retard critique est égal à 12. De façon analogue, le retard critique du réseau (b) de la Figure IV.13 est égal à 9.

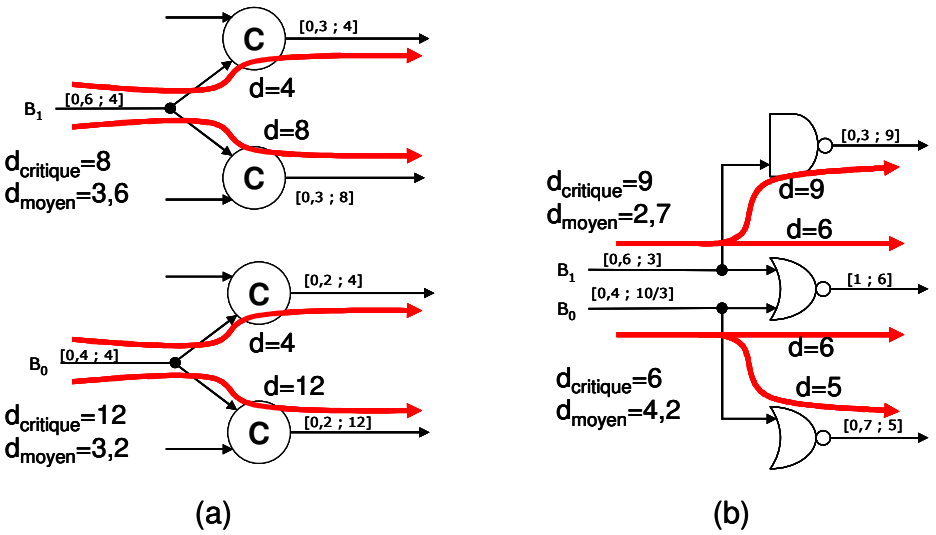


Figure IV.13 – Retard critique et retard moyen pour deux réseau de portes où (a) les nœuds contiennent de chemins exclusifs et (b) les nœud contiennent de chemins non exclusifs.

IV.3.1. Le retard Moyen

Comme pour l'énergie, il est aussi important de connaître le retard moyen dans un circuit. Pour ce faire, en utilisant les probabilités de commutation d'un nœud (i.e. probabilité de commutation de la porte qui pilote le nœud), il est possible de déterminer le retard moyen. Du aux propriétés des circuits asynchrones, l'ensemble F_i peut être divisé en deux ensembles distincts F_i^1 et F_i^0. Donc:

$$F_i = F_i^1 \cup F_i^0$$
$$F_i^1 \cap F_i^0 = 0 \qquad \text{(IV-20)}$$

En supposant une transition du nœud i, F_i^1 est l'ensemble des portes pilotées par le nœud i qui commuteront systématiquement, c'est-à-dire que la probabilité de commutation d'une porte appartenant à F_i^1 est égale à 1.

$$p(f) = 1 \forall f \in F_i^1 \qquad \text{(IV-21)}$$

En sachant que toutes les portes logiques appartenant à l'ensemble F_i^1 commutent pour toute opération réalisée, le retard moyen de cet ensemble de portes est égal au retard critique du même ensemble vu que pour toute opération le retard sur l'ensemble F_i^1 est le même. Donc, il est possible de définir le retard moyen sûr F_i^1 comme étant :

$$d^{moyen}\left(F_i^1\right) = \max_j \left\{ d\left(f_j\right) + d_j^{critique} \,/\, f_j \in F_i^1 \right\} \qquad \text{(IV-22)}$$

L'ensemble F_i^0 est l'ensemble de portes avec probabilité inférieure à 1 et une seule porte de cet ensemble commutera. Donc,

$$\begin{aligned} &p(f) < 1 \forall f \in F_i^0 \\ &p(f_j \mid f_k) = 0 \forall f_j \in F_i^0, f_k \in F_i^0, j \neq k \\ &p(F_i^0) = \left\{ \sum_j p(f_j) \right\} \forall f_j \in F_i^0 \\ &p(F_i^0) \leq 1 \end{aligned} \tag{IV-23}$$

Comme une seule porte commute par opération, le retard moyen sur l'ensemble F_i^0 doit être pondéré par la probabilité de chaque porte à commuter, c'est-à-dire, le retard moyen sur F_i^0 est la moyenne des retards des portes appartenant à cet ensemble. Ainsi, le retard moyen de l'ensemble F_i^0 est défini par :

$$d^{moyen}\left(F_i^0\right) = \left\{ \sum_j p\left(f_j\right) \cdot \left(d\left(f_j\right) + d_j^{critique}\right) \right\} \forall f_j \in F_i^0 \tag{IV-24}$$

À partir de ces définitions, le retard moyen sur un nœud peut être calculé. Le retard moyen sur un nœud est calculé par :

$$d_i^{moyen} = \left\{ \sum_j p\left(f_j\right) \cdot \max\left[\left(d\left(f_j\right) + d_j^{critique}\right); d(F_i^1)\right] \right\} \forall f_j \in F_i^0 \tag{IV-25}$$

Comme pour le retard critique, cette formulation est appliquée à partir des sorties du réseau de portes jusqu'à ses entrées primaires. Le retard moyen d'un canal d'entrée du réseau est donné par la somme de tous les retards moyens des nœuds appartenant au canal d'entrée. Ainsi, le retard moyen sur un nœud *i* n'est pas le retard moyen de ce nœud, mais la contribution de ce nœud au retard moyen du circuit. Finalement, le retard moyen du réseau est le maximum des retards moyens des canaux d'entrée du circuit.

Dans la Figure IV.13, le retard moyen du nœud B_0 du circuit (a) est égal 3,2 et ce retard est obtenu par :

$$\begin{aligned} &p\left(F_{B_0}^0\right) = 0{,}2 + 0{,}2 = 0{,}4 \\ &d^{moyen}\left(F_{B_0}^1\right) = 0 \\ &d^{moyen}\left(F_{B_0}^0\right) = 0{,}2 \times 4 + 0{,}2 \times 12 = 3{,}2 \\ &d_{B_0}^{moyen} = 0{,}2 \times \max(4;0) + 0{,}2 \times \max(12;0) = 3{,}2 \end{aligned} \tag{IV-26}$$

De façon analogue, le retard moyen du nœud B_1 pour le même circuit est égal 3,6. Donc, le retard moyen du canal B est égal à 6,8. Ce résultat peut être facilement vérifié vu que dans 50% des cas le réseau présent un retard égal à 4, dans 30% le réseau présent un retard égal à 8 et dans le 20% restant des cas le retard est égal à 12. La moyenne des retards est, donc, 6,8.

Pour le réseau (b) de la Figure IV.13, le retard moyen du nœud B_0 est donné par :

$$\begin{aligned} & p\left(F_{B_0}^0\right) = 0{,}7 \\ & d^{moyen}\left(F_{B_0}^1\right) = 6 \\ & d^{moyen}\left(F_{B_0}^0\right) = 0{,}7 \times 5 = 3{,}5 \\ & d_{B_0}^{moyen} = 0{,}7 \times \max(5;6) = 4{,}2 \end{aligned} \qquad \text{(IV-27)}$$

À nouveau et en utilisant la même formulation, le retard moyen du nœud B_1 pour le réseau (b) est égal à 2,7 et le retard moyen du canal B est égal à 6,9. Cet exemple montre l'importance de connaître le retard moyen d'un réseau de portes. Malgré, un grand écart entre les retards critiques des deux circuits de la Figure IV.13, ces circuits possèdent presque le même retard moyen et, donc, même si le réseau (a) de la Figure IV.13 présente un chemin critique plus long, la vitesse moyenne du circuit est plus grande.

La Figure IV.14 et la Figure IV.15 présentent les résultats de retard critique et de retard moyen pour deux implémentations de la fonction « ET » à deux entrées double rail asynchrone.

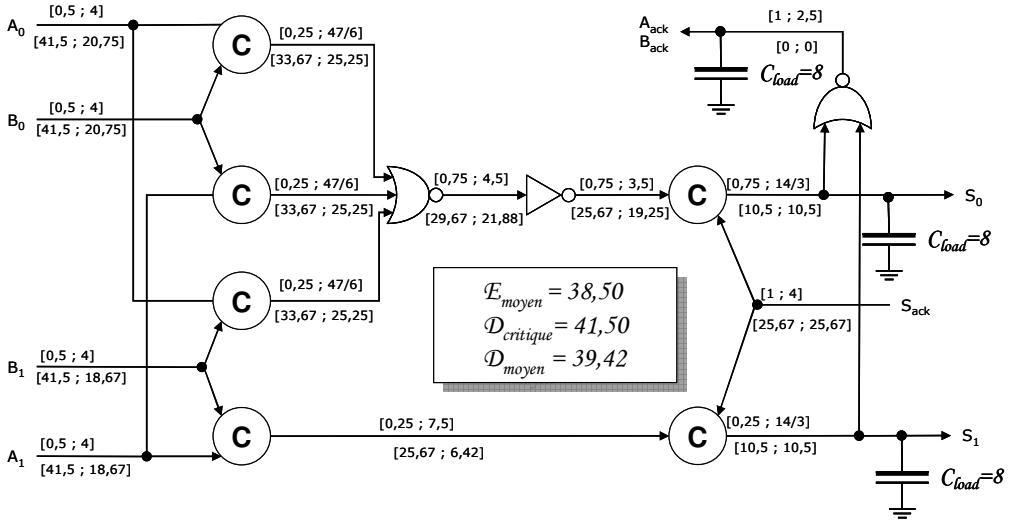


Figure IV.14 – Implémentation DIMS d'une porte AND-2 double rail avec la probabilité, la capacité, le retard critique et le retard moyen pour chaque nœud.

Pour chaque nœud des deux réseaux, les valeurs de probabilité, capacité, retard critique et retard moyen sont présentées. Une charge de sortie équivalente à 8 inverseurs de référence a été ajoutée à chaque sortie du circuit pour le calcul du retard. Ici à nouveau l'implémentation DIMS de la Figure IV.14 est plus performante en termes de retard moyen et énergie moyenne consommée que le circuit présenté dans la Figure IV.15.

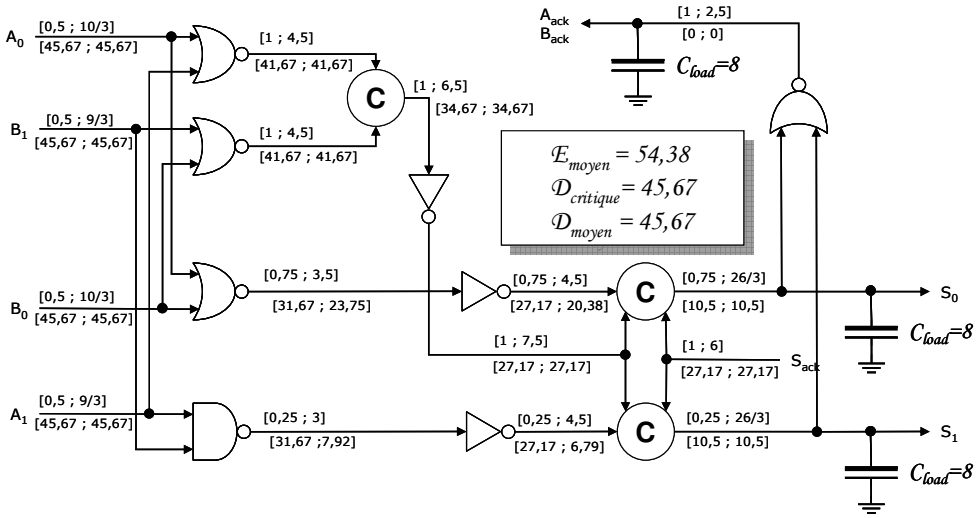


Figure IV.15 – Une autre implémentation d'une porte AND-2 double rail avec la probabilité, la capacité, le retard critique et le retard moyen pour chaque nœud.

IV.4. La Métrique Et^n

Comme pour la valeur du retard, la valeur de l'énergie moyenne obtenue par l'équation IV-12 est normalisée en fonction de l'énergie pour piloter un inverseur de référence. De cette manière, les valeurs de consommation moyenne et retard sont indépendantes de la technologie cible.

Cependant, il est important de connaître l'efficacité des architectures en terme de consommation d'énergie et de vitesse d'opération. Pour ce faire, les métriques qui combinent l'énergie et la vitesse en un unique nombre sont particulièrement intéressantes. En effet, ces métriques sont exprimées par la relation Et^n [IV.14], où n transcrit l'importance du paramètre vitesse par rapport à la consommation. Ainsi, une métrique où $n = 0$, ne prend en compte que la consommation d'énergie tandis qu'une autre où $n \to +\infty$ ne prend en compte que la vitesse.

La théorie des circuits montre que la métrique Et^2 est, en première approximation, constante quelle que soit la tension d'alimentation [IV.15][IV.16]. Cependant, il est vrai que

les concepteurs de circuits synchrones préfèrent utiliser la métrique Et compte tenu de leur prudence à propos du choix de la fréquence d'horloge.

En revanche, la métrique Et^2 est intéressante pour une large classe de circuits asynchrones de type « QDI » (quasi delay-insensitive) car ceux-ci restent fonctionnels sur une large gamme de valeurs de tension d'alimentation [IV.17].

En supposant une première version de chaque implémentation envisagée décrite utilisant les tailles minimales des transistors pour chaque porte logique, chaque implémentation est rapidement évaluée pour déterminer l'énergie et le retard avec la méthodologie présentée. Il est maintenant possible de calculer la valeur de Et^2 pour chaque solution.

Si $E_A t_A^2 < E_B t_B^2$, la solution « A » est plus performante en termes d'énergie et de retard que la solution « B ». Il faut remarquer que les valeurs de *E* et *t* sont exprimées en fonction d'un inverseur de référence. De ce fait, la comparaison est relative et donc indépendante de la technologie cible.

De plus, il est possible d'effectuer des transformations successives *T* (dimensionnement de portes, sortance, …) à chaque architecture. Après chaque transformation, un nouveau couple (*E*,*t*) peut être à nouveau rapidement calculé en utilisant la méthode. Il est donc désormais possible de déterminer *n*, tel que Et^n soit constant pour cette transformation *T*.

Plus la valeur de *n* est grande, plus l'énergie sera altérée pour une modification du retard. En conséquence, beaucoup d'énergie sera dépensée pour une anodine accélération de la vitesse d'opération du circuit. En revanche, beaucoup d'énergie sera épargnée sans grande réduction de la vitesse de l'opération.

Alors, la métrique Et^n avec la méthodologie présentée permet rapidement de :

- ➔ comparer deux solutions à l'égard d'une certaine transformation ;
- ➔ comparer l'efficacité d'une transformation pour deux solutions distinctes ;
- ➔ comparer l'efficacité de deux transformations pour une même solution.

La Figure IV.16 montre une comparaison hypothétique entre deux solutions pour une transformation *T* donnée. Les contraintes de conception détermineront la solution à adopter.

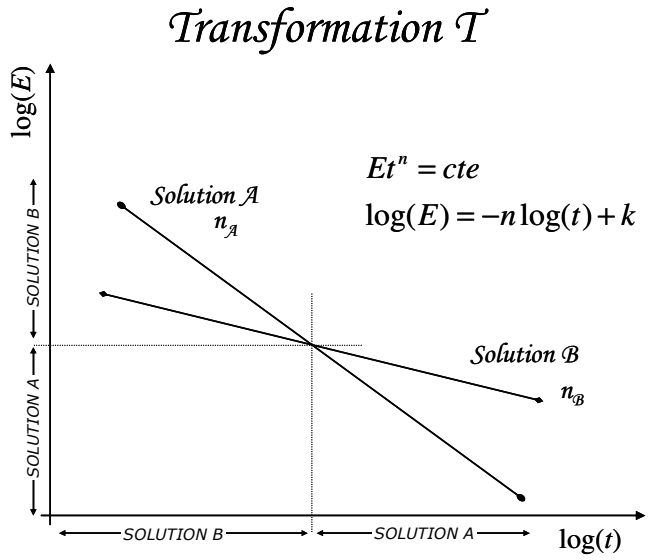


Figure IV.16 – Comparaison entre deux solutions vis-à-vis d'une transformation *T*.

IV.5. Dimensionnement de Portes – Une Étude de Cas

Pour valider la méthodologie présentée, les problèmes de détermination du retard minimum atteignable et le dimensionnement de portes ont été étudiés. L'algorithme proposé par Karandikar et Sapatnekar [IV.18] a été implémenté et il est employé pour déterminer rapidement le retard minimum atteignable pour un circuit. Le choix de cet algorithme est poussé par le fait de sa récente publication et, principalement, parce que cet algorithme utilise le même niveau d'abstraction et le même modèle de retard que la méthodologie proposée ici.

IV.5.1. Algorithme pour l'Estimation du Retard Minimum

Chaque porte possède plusieurs tailles disponibles et chaque taille correspond à une capacité d'entrée c_{in}. Un inverseur de taille 2 correspond à un inverseur avec les transistors PMOS et NMOS deux fois plus larges que l'inverseur de référence et, donc, la capacité d'entrée c_{in} deux fois plus grande. De façon équivalente, une porte NAND-2 de taille 2 est une porte NAND-2 qui possède une capacité d'entrée deux fois plus grande.

En conséquence, l'ensemble C_{in}^{F} est le groupe de toutes les valeurs possibles de capacités d'entrées de la porte F. Une porte peut piloter plusieurs autres portes et, donc, la capacité de charge pilotée (c_l) est la somme des capacités d'entrées des portes pilotés plus la capacité du nœud c_w (capacité de sortie de la porte, capacité de routage, etc.). En supposant

une porte F pilotant n portes $F_1, F_2, F_3, ..., F_n$, les valeurs possibles de la capacité de charge sont décrites par la somme de c_w avec les éléments de l'ensemble C_l^F qui est défini par :

$$C_l^F = \left\{ \sum_{j=1}^{n} c_j : \forall c_j \in C_{in}^{F_j}, j = 1 \cdots n \right\} \quad \text{(IV-28)}$$

L'algorithme présenté en [IV.18] consiste à parcourir les circuits en partant des sorties primaires jusqu'aux entrées primaires. Pour chaque porte logique, le retard maximum des entrées de la porte jusqu'à n'importe laquelle des sorties S est minimisé. Pour ce faire, la courbe $retard \times C_{in}$ est calculée. Chaque point de la courbe, représenté par $D_{F \to S}[c_{in}], \forall c_{in} \in C_{in}^F$, est défini comme le retard maximum à partir de la porte F, avec la capacité d'entrée c_{in}, jusqu'à une sortie. Pour déterminer un point, la valeur du retard de la porte – représenté par $D_F[c_{in}][c_l]$ – est d'abord calculée pour chaque capacité de charge obtenue par $c_l = c_j + c_w, c_j \in C_l^F$ et, ensuite, le résultat est additionné au retard (correspondant à la charge) du nœud de sortie de la porte jusqu'aux sorties.

Pour illustrer l'algorithme, le circuit de la Figure IV.17 est utilisé comme exemple. Toutes les portes sont des inverseurs pour simplifier la discussion, mais l'algorithme est applicable aux portes plus complexes en utilisant les valeurs appropriées pour l'effort logique.

IV.5.1.1. Sortance Simple

La sortie d'une porte pilotant une seule charge (sortance égale à 1) peut être connectée soit à une charge fixe, comme à une sortie primaire, soit à une autre porte logique. Dans le deuxième cas la charge dépend de la taille de la porte qui est pilotée. Ces deux cas sont présentés individuellement en vue de faciliter la compréhension, mais le cas de charge fixe est un cas particulier d'une charge variable avec une seule valeur de charge.

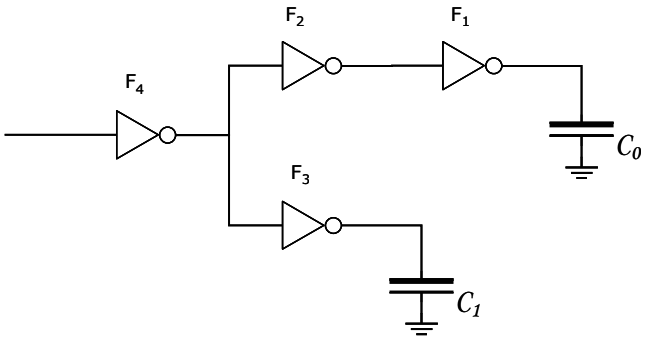


Figure IV.17 – Circuit exemple pour le calcul de la courbe *retard*xC_{in} et propagation au travers de multiples sortances.

IV.5.1.1.1. Charge Fixe

La porte F_1 pilote une charge fixe C_0 à sa sortie primaire comme le montre la Figure IV.17. Obtenir la courbe $retard \times C_{in}$ est trivial. Chaque taille possible pour F_1 correspond à une valeur de $c_{in} \in C_{in}^{F_1}$ pour la porte et le retard peut être calculé par l'équation IV-17. Dans ce cas, le retard jusqu'à la sortie primaire est égal au retard de la porte et, donc :

$$\begin{aligned} D_{F_1 \to S}[c_{in}] &= D_{F_1}[c_{in}][c_l] \\ &= \left[g_{F_1} \times \frac{c_l}{c_{in}} \right] \\ \text{avec } c_l &= C_0 \end{aligned} \qquad \text{(IV-29)}$$

Le résultat du retard pour plusieurs taille de la porte F_1 est montré sur le graphique (a) de la Figure IV.18. Comme la charge à la sortie de F_1 est fixe, le résultat est une courbe décroissant en fonction de l'augmentation de la taille de F_1.

IV.5.1.1.2. Charge Variable

En considérant maintenant la porte F_2 pilotant la porte F_1. La charge de la porte F_2 n'est pas constante, mais elle varie avec la taille de F_1. Comme la courbe $retard \times C_{in}$ pour F_1 a déjà été calculée, la courbe $retard \times C_{in}$ pour F_2 est calculée en deux étapes. Primo, pour une taille donnée de F_2, correspondant à une capacité d'entrée $c_{in} \in C_{in}^{F_2}$, toutes les tailles de F_2 sont examinées et le retard est calculé avec l'équation IV-30. Ici, la charge c_l est la somme de la capacité d'entrée de la porte F_1 et les capacités parasites à la sortie de la porte F_2. Comme il y a une seule porte connectée à la sortie de F_2, donc $C_l^{F_2} = C_{in}^{F_1}$.

En combinant le retard $D_{F_2}[c_{in}][c_l]$ de la porte F_2 avec le retard $D_{F_1 \to S}[c_j]$ qui est le retard de F_1 pour la taille considérée, le retard à partir de l'entrée de F_2 jusqu'à la sortie est calculé. Le graphique (b) de la Figure IV.18 montre la surface obtenue en combinant toutes les tailles de F_1 et F_2.

En conséquence, le retard minimum pour une taille donnée de F_2 est déterminé par l'équation IV-31.

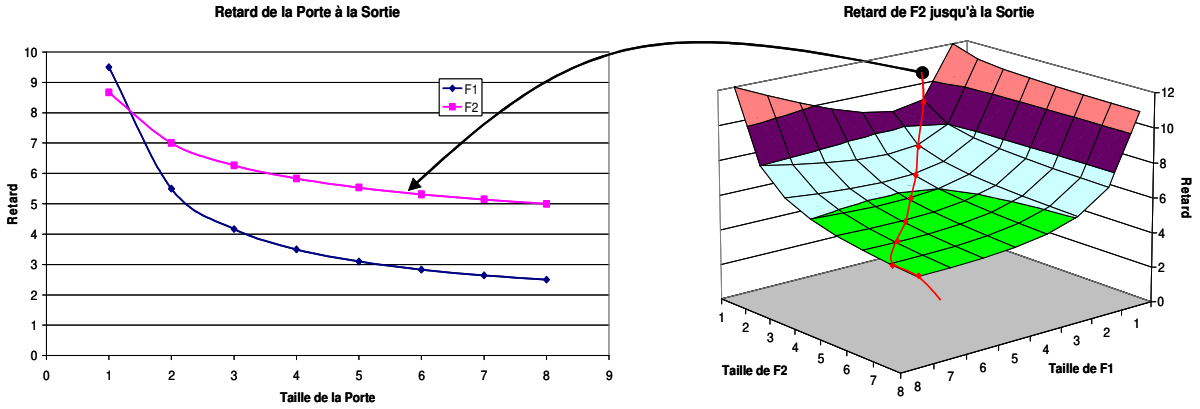
$$D_{F_2}[c_{in}][c_l] = \left[g_{F_2} \times \frac{c_l}{c_{in}} \right] \quad \text{(IV-30)}$$

$$D_{F_2 \to S}[c_{in}] = \min_{c_j \in C_l^{F_2}} \{ D_{F_2}[c_{in}][c_l] + D_{F_1 \to S}[c_j] \} \quad \text{(IV-31)}$$

$$\text{avec } c_l = c_j + c_w$$

Donc, les valeurs minima de retard pour chaque taille de F_2 (courbe sur le graphique (b) de la Figure IV.18) composent la courbe $retard \times C_{in}$ de F_2 (graphique (a)). Il faut souligner que la taille de F_1 qui minimise $D_{F_2 \to S}[c_{in}]$ n'est pas forcement la taille qui minimise $D_{F_1 \to S}[c_j]$.

Pour une taille de la porte F_2, le retard de la porte jusqu'aux sorties primaires varie en fonction de deux facteurs qui changent simultanément : (i) la charge pilotée par F_2 qui correspond aux différentes tailles de F_1 et (ii) les retards correspondants de F_1.



(a) (b)

Figure IV.18 – Courbes *retard*xC_{in} pour les portes F_1 et F_2 et leur propagations au travers de deux portes.

IV.5.1.2.Sortance Multiple

Les scénari présentés dans la section précédente sont les plus simples des cas. Maintenant, en considérant une porte pilotant plusieurs portes, comme la porte F_4 de la Figure IV.17, l'algorithme proposé par Karandikar et Sapatnekar tient compte simultanément des différentes valeurs de retard et des différentes tailles de portes pilotées. Les retards au travers des portes F_2 et F_3 peuvent présenter différentes valeurs qui dépendent des tailles de ces portes.

En supposant que les courbes $retard \times C_{in}$ ont été calculées pour les portes F_2 et F_3, donc, $C_l^{F_4} = C_{in}^{F_2} \times C_{in}^{F_3}$. La charge c_l pilotée par F_4 est l'addition des capacités d'entrées de F_2 et F_3 avec la capacité parasite à la sortie de la porte F_4. Pour une taille donnée de F_4, le retard de la porte est obtenu en utilisant l'équation IV-32 pour chaque valeur de charge.

$$D_{F_4}[c_{in}][c_l] = \left[g_{F_4} \times \frac{c_l}{c_{in}} \right] \tag{IV-32}$$

$$\begin{aligned} D_{F_4 \to S}[c_{in}] &= \min_{c_j, ck} \{ D_{F_4}[c_{in}][c_l] + D_{O \to S} \} \\ \text{avec } D_{O \to S} &= \max\{ D_{F_2 \to S}[c_j], D_{F_3 \to S}[c_k] \} \\ \text{avec } c_l &= c_j + c_k + c_w, \forall c_j \in C_{in}^{F_2}, \forall c_k \in C_{in}^{F_3} \end{aligned} \tag{IV-33}$$

Ensuite, le retard à partir de F_4 jusqu'aux sorties est calculé en combinant le valeur de $D_{F_4}[c_{in}][c_l]$ avec le maximum des retards de chaque chemin. Dans ce cas, l'identité du chemin avec le retard maximum jusqu'aux sorties peut changer en accord avec la branche critique. Pour une certaine taille de F_4, un chemin peut déterminer la valeur critique, tandis que pour une autre taille, un autre chemin peut devenir critique. L'équation IV-33 prend en compte cette formulation et, en conséquence, pour une taille donnée de F_4, l'algorithme calcule les tailles optima pour chaque porte pilotée pour déterminer le retard minimum à partir de F_4. Cette procédure est répétée pour toutes les tailles de la porte F_4 pour finalement obtenir la courbe complète $retard \times C_{in}$.

À première vue, la taille de l'ensemble $C_l^{F_4}$ est proportionnelle au produit du nombre de portes pilotées par le nombre de tailles possibles pour ces portes. En supposant une porte F pilotant quatre autres portes qui possèdent les courbes $retard \times C_{in}$ représentées par k_1, k_2, k_3 et k_4 dans la Figure IV.19. Si chaque porte possède m tailles, donc les courbes possèdent m points et la taille de $C_l^{F_4}$ est égale à m^4. Cependant, la plus grande part des valeurs de $C_l^{F_4}$ sont redondantes. Par exemple, en considérant le quadruplet T composé des premiers points $\langle t_1, t_2, t_3, t_4 \rangle$ de chaque courbe de la Figure IV.19, le quadruplet T' composé par les points $\langle t_1, t_2', t_3', t_4' \rangle$ est redondant. Deux valeurs sont extraites de T et T' pour le calcul du retard: (1) le retard maximum vers les sorties primaires et (2) la somme des capacités d'entrées représentées par leurs points (charge capacitive). Le maximum retard est le même dans les deux quadruplets (point t_1), mais la charge représentée par le quadruplet T' est plus élevée. En

conséquence, le retard de F est supérieur et, par la suite, le retard jusqu'aux sorties est supérieur dans le quadruplet T'. Comme le but est de minimiser le retard, la solution offerte par le quadruplet T' ne remplacera jamais la solution obtenue avec le quadruplet T.

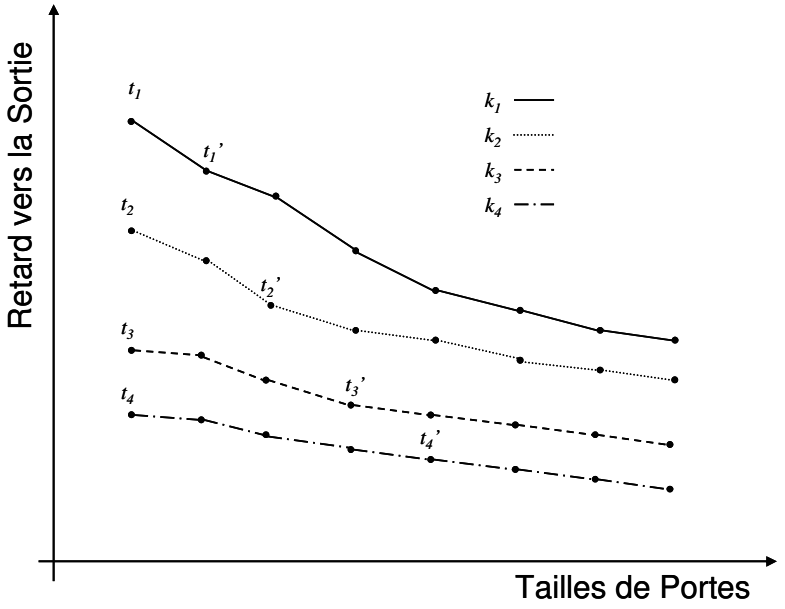


Figure IV.19 – Combinaison de courbes *retard*xC_{in} pour une sortance multiple.

Cette discussion conduit à une stratégie pour sélectionner efficacement les valeurs de c_l à partir des courbes $retard \times C_{in}$ des portes pilotées. Primo, les courbes sont stockées dans un ordre décroissant des retards (tailles croissantes). La première valeur de c_l est la capacité parasite plus les capacités correspondantes aux retards maxima de chaque courbes – le quadruplet T. La valeur suivante est obtenue en remplaçant le point avec le retard maximum (i.e., t_1 de la courbe k_1 dans T) par le point suivant de la même courbe. De cette façon, les combinaisons de t_1 avec les points restants des autres courbes sont ignorées. Ce procédé se poursuit jusqu'au retard maximum soit le dernier point de la courbe k_1. Ainsi, le nombre total de combinaisons est de l'ordre de la somme du nombre de points dans chaque courbe plutôt que le produit du nombre de points.

IV.5.1.3. L'Algorithme

La Figure IV.20 présent le pseudo code pour l'algorithme de Karandikar et Sapatnekar d'estimation du retard minimum pour une réseau de portes logiques. Le calcul est basé sur les courbes $retard \times C_{in}$ présentées dans les sections précédentes. Toutes les portes sont évaluées dans l'ordre topologique partant des sorties primaires du réseau jusqu'à ses entrées. Dans

chaque nœud d'entrée du circuit, le retard minimum est sélectionné et le maximum sur tous les nœud d'entrée est défini comme étant le retard minimum atteignable par le réseau.

En supposant qu'il existe m tailles disponibles pour chaque porte, que le circuit comportant N portes logiques et que la sortance maximum dans un nœud est $|FO|$, la boucle la plus interne est exécutée $O(m \times |FO|)$ fois. Aussi, le coût de détermination du retard maximum est $O(|FO|)$. La deuxième boucle est exécutée m fois vu qu'il existe m tailles disponibles. Finalement, comme il y a N portes, la boucle plus externe est répétée N fois. En conséquence, le temps d'exécution de l'algorithme est proportionnel à $O(N \times m \times m \times |FO| \times |FO|)$. Cependant, cette valeur est très pessimiste puisque, en réalité, très peu de portes possèdent une sortance égale à $|FO|$.

Algorithme ERM: Estimation du Retard Minimum

pour chaque porte F avec ses sorties calculées **faire**
  //calculer la courbe $retard \times C_{in}$ *pour F*
  pour toutes les $c_{in} \in C_{in}^{F}$ **faire**
    $D_{F_1 \to S}[c_{in}] = \infty$
    pour chaque $c_j \in C_l^F$ qui n'est pas redondant **faire**
      // F pilote n portes $F_1, F_2, \ldots, F_n$
      $c_l = c_w + \sum_{j=1}^{n} c_j$
      // calculer le retard de F
      $D_F[c_{in}][c_l] = \left\lceil g_F \times \frac{c_l}{c_{in}} \right\rceil$
      // déterminer le retard maximum de F à les sorties
      $temp = D_F[c_{in}][c_l] + \max_{j=1\ldots n}\{D_{F_j \to S}[c_j]\}$
      $D_{F \to S}[c_{in}] = \min(temp, D_{F \to S}[c_{in}])$
    fin pour
  fin pour
fin pour
Minimum Retard = $\max_{\text{Entrées Primaires}} \{\min\{retard\_aux\_sorties\}\}$

Figure IV.20 – Pseudo code pour l'algorithme d'estimation du retard minimum.

L'algorithme d'estimation du retard minimum est optimal pour un arbre de portes logiques. Malheureusement, la majorité des circuits sont des graphes acycliques dirigés (DAG) avec des chemins divergents et convergents à nouveau. C'est-à-dire, dans un DAG, il existe plusieurs chemins entre deux nœuds du réseau. Le problème avec l'algorithme est qu'il suppose, implicitement, que les courbes $retard \times C_{in}$ dans un nœud de sortance multiple sont indépendantes et qu'il peut choisir librement la combinaison des retards et capacités les meilleurs ajustés à une porte. Dans des chemins convergents à nouveau, les choix ne sont malheureusement pas indépendants. La sélection d'un ensemble de tailles dans une sortie

restreint les options sur les autres chemins et déterminer les relations entre plusieurs sorties est un problème difficilement soluble dans la plupart des circuits.

Cependant, l'hypothèse d'indépendance des chemins n'est pas aussi déraisonnable. Si les chemins sont complètement déséquilibrés, c'est-à-dire, si leurs structures logiques sont telles qu'un chemin a toujours un retard inférieur à l'autre, aucune erreur n'est introduite par la façon dont les courbes sont combinées. D'autre part, si les chemins possèdent des retards similaires, l'algorithme sélectionnera des valeurs de capacités d'entrées similaires. Ça peut conduire à des imprécisions vu que les valeurs des capacités d'entrées peuvent être légèrement différentes. Cependant, l'erreur dans l'estimation reste limitée comme le montre les résultats présentés en [IV.18].

IV.5.2. Dimensionnement des Portes

L'algorithme de la Figure IV.20 est utilisé pour obtenir les tailles des portes qui conduisent au retard minimum. Pendant l'exécution de l'algorithme les charges du retard minimum sont stockées. Par la suite, cette information est utilisée dans un parcours direct du circuit pour générer les tailles des portes. Une porte avec plusieurs entrées (i.e., convergence des différents chemins) possède plusieurs choix pour sa taille. Le choix est déterminé par le chemin critique, c'est-à-dire, le chemin imposant la plus grande taille. L'effet sur les chemins non critiques est l'insertion d'une charge plus grande que celle initialement prévue. Cet effet ne pose pas de problème si les chemins ne possèdent pas des valeurs de retards similaires. Par contre, si un chemin peut devenir critique par le changement de la charge, les portes sur ce chemin doivent être dimensionnées.

De plus, les chemins non critiques sont dimensionnés. La première porte d'un chemin non critique gardera sa taille minimale, par contre, les portes successives de ces chemins sont dimensionnées pour accomplir le retard minimum avec la porte initiale à taille minimum. Il faut rappeler que la vitesse des circuits asynchrones est, en fait, la moyenne sur la totalité de ces opérations. En conséquence, il est aussi important d'optimiser tous les chemins d'un circuit.

Ainsi, après un parcours, les portes du circuit sont dimensionnées pour atteindre le retard minimum. La solution proposée pour dimensionner les portes n'est pas exacte, mais elle offre une première solution qui peut être utilisée dans un outil exact de dimensionnement de portes. En partant de la solution fournie par l'algorithme qui est proche de la solution optimale, le temps d'exécution pour un dimensionnement exact peut être sensiblement accéléré.

Aussi, il faut souligner que le but cherché ici n'est pas de proposer un algorithme pour le dimensionnement, mais simplement pouvoir évaluer la métrique de comparaison entre différentes solutions.

IV.5.3. Résultats

Plusieurs circuits ont été modélisés et dimensionnés avec l'algorithme précèdent. Pour chaque circuit, la méthode d'estimation de consommation et retard moyen a été employée pour déterminer l'énergie moyenne consommée et le temps de propagation des circuits.

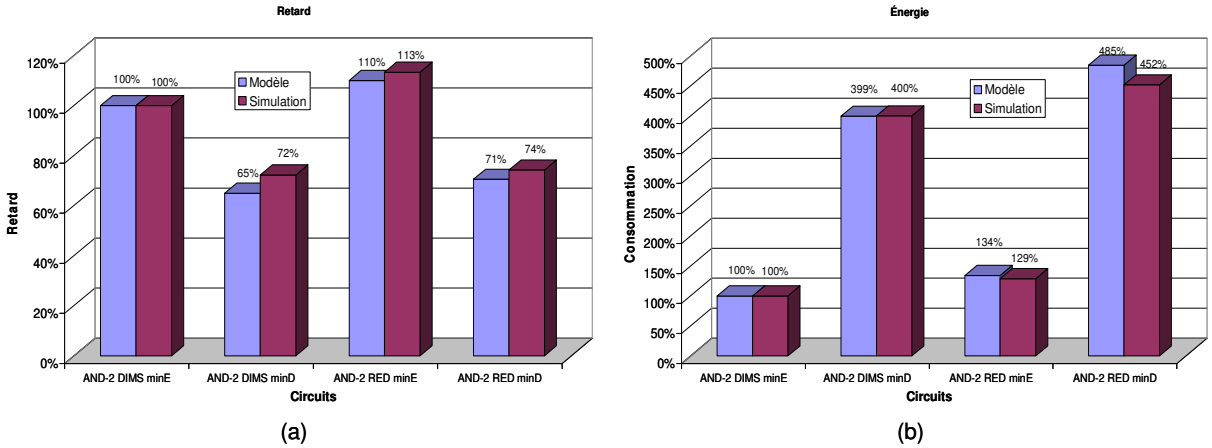


Figure IV.21 – Résultats de (a) retard et (b) énergie pour les deux implémentations de la fonction AND-2 double rail.

La Figure IV.21 montre les résultats de retard – graphique (a) – et d'énergie – graphique (b) – pour les circuits de la Figure IV.10 et Figure IV.11. Pour chaque circuit, une première version avec les tailles minimales pour les portes a été évaluée. Cette version correspond au circuit avec l'énergie minimale vu que cette version conduit aux charges minimales dans les nœuds du circuit. Ensuite, les circuits ont été taillés pour produire le retard minimum avec l'algorithme précèdent.

Ensuite, une simulation électrique au niveau transistors a été effectuée pour vérifier les résultats du modèle. Les résultats obtenus par le modèle et par simulation ont été normalisés par rapport à la version AND-2 DIMS d'énergie minimum.

Par ces résultats, l'erreur du modèle est inférieure à 10%, mais, surtout, les rapports de performances entre les circuits sont respectés. Ces résultats montrent que le modèle fournit une méthode rapide et efficace permettant de comparer plusieurs types d'implémentation des circuits asynchrones. La conclusion peut être aussi vérifiée dans la Figure IV.22, où les

résultats sont combinés dans une métrique Et^n pour déterminer les valeurs de n pour chaque circuit vis-à-vis du dimensionnement des portes.

Même si une approximation linéaire du comportement de l'algorithme peut paraître grossière, cette estimation permet déjà d'avoir une bonne idée des solutions envisageables pour atteindre les spécifications d'un projet.

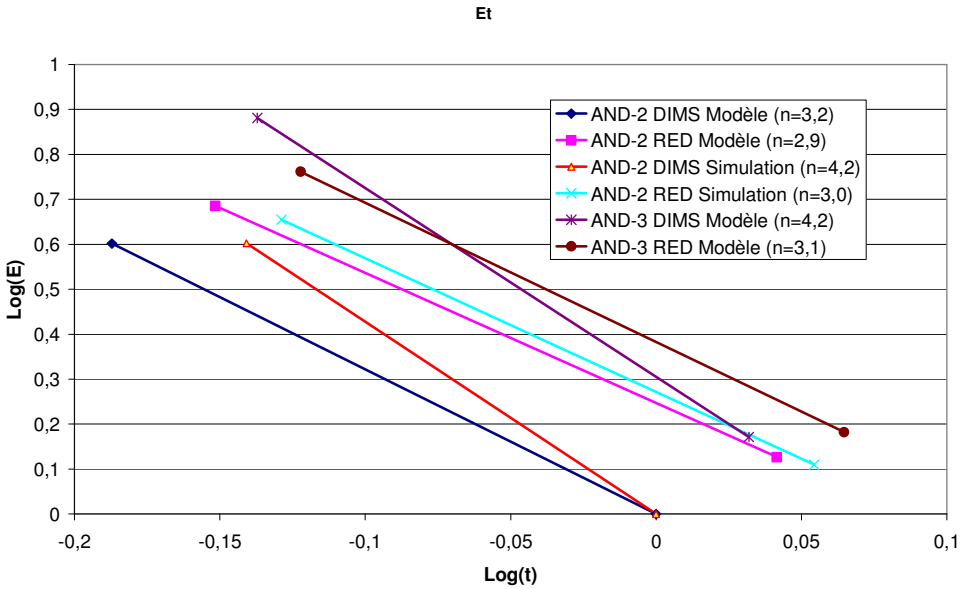


Figure IV.22 – Courbes Et pour différents circuits vis-à-vis de la transformation effectuée par l'algorithme de dimensionnement.

IV.6. Conclusions

Dans ce chapitre, une méthodologie d'estimation de la surface, de l'énergie moyenne consommée et du retard moyen d'un circuit asynchrone a été proposé. Ensuite, une métrique qui combine les valeurs de retard et énergie a été présentée. De cette façon, une métrique, aussi bien que son implémentation, permettant de comparer plusieurs solutions a été élaborée.

Le modèle a été évalué en utilisant un algorithme de détermination du retard minimum atteignable par un circuit et du dimensionnement de ses portes logiques. Les résultats du modèle ont été confrontés avec les résultats de simulations électriques. La confrontation a prouvé que le modèle peut être utilisé pour la comparaison de différentes solutions architecturales et pour la comparaison de différentes transformations dans une architecture.

Le modèle proposé est indépendant de la technologie cible utilisée, cependant, la précision du modèle peut être ajustée avec les possibles variations particulières à une technologie de fabrication. En particulière, il est possible de prendre en charge les courants de

fuite (« leakage ») qui sont de plus en plus importantes en technologies inférieures à 100nm. Avec la connaissance de la probabilité de commutation de chaque porte, il est possible d'estimer le nombre moyen de transistors commutant par cycle d'opération. Il est, donc, possible d'effectuer une correction de la valeur de l'énergie moyenne consommée en vue d'ajouter l'énergie résultante des courants de fuite.

Même si le modèle ne contient pas de nouveauté, la métrique proposée et son implémentation peuvent être insérées dans les contributions de cette thèse et les résultats ici présentés ont été publiés [IV.19].

IV.7. Références

[IV.1] T. Sakurai, A. R. Newton. Alpha-Power Law MOSFET Model and its Applications to CMOS Inverter Delay and Other Formulas. IEEE Journal of Solid-State Circuits, 25(2), April 1990.

[IV.2] N. Hedenstierna, K. O. Jeppson, CMOS Circuit Speed and Buffer Optimization. IEEE Transactions on Computer-Aided Design, CAS-6(2), March 1987.

[IV.3] S. R. Vemuru, N. Scheinberg. Short-Circuit Power Dissipation Estimation for CMOS Logic Gates. IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, 41(11). November 1994.

[IV.4] L. Bisdounis, S. Nikolaidis, O. Koufopavlou. Propagation Delay and Short-Circuit Power Dissipation Modeling of the CMOS Inverter. IEEE Transaction on Circuits and Systems-I: Fundamental Theory and Applications, 45(3). March 1998.

[IV.5] C. Mead, L. Conway. Introduction to VLSI Systems. Addison Wesley, 1980.

[IV.6] N. H. West, K. Eshraghian. Principles of CMOS VLSI Design. Addison Wesley, 1994.

[IV.7] F. Najm, R. Burch, P. Yang, I. Hajj. CREST – A current estimation tool for CMOS circuits. IEEE Int. Conf. Computer-Aided Design. Santa Clara, CA. November 7-10, 1988. pp 204-207.

[IV.8] C. Y. Tsui, M. Pedran, A. M. Despain. Efficient estimation of synamic power consumption under a real delay model. IEEE Int. Conf. Computer-Aided Design. Santa Clara, CA. November 7-11, 1993. pp 224-228.

[IV.9] C. M. Huizer. Power dissipation analysis of CMOS VLSI circuits by means of switch-level simulation. IEEE Europ. Solid-State Computer Conf. Grenoble, France, 1990. pp 61-64.

[IV.10] R. Burch, F. Najm, P. Yang, T. Trick. McPower: A Monte Carlo approach to power estimation. IEEE/ACM Int. Conf. Computer-Aided Design. Santa Clara, CA. November 8-12, 1992. pp 90-97.

[IV.11] P. I. Pénzes, A. J. Martin. An energy estimation method for asynchronous microprocessor. Design Automation and Test in Europe. March 4-8, Paris, France, 2002.

[IV.12] A. J. Martin, A. Lines, R. Manohar, M. Nyström, P. I. Pénzes, R. Southworth, U. Cummings. The design of an asynchronous MIPS R3000 microprocessor. Proceedings of the 17th Conf. on Advanced Research in VLSI. IEEE Computer Society Press, 1997. pp 164-181.

[IV.13] I. Sutherland, B. Sproull, D. Harris. Logical effort: designing fast CMOS circuits. Morkan Kaufmann Publishers, Inc. San Francisco, CA, 1999. ISBN 1-55860-557-6. 240 pp.

[IV.14] P. Pénzes, "Energy-Delay Complexity of Asynchronous Circuits", Thesis, Caltech, Pasadena, California, USA. 2002.

[IV.15] A. Martin, "Remarks on Low-power Advantages of Asynchronous Circuits", ESSCIRC'98: Low-power Systems on a Chip. September, 1998.

[IV.16] A. Martin, "Towards an Energy Complexity of Computation", Information Pocessing Letters, 77, 2001.

[IV.17] E. Ou, M. Nyström, "Energy-Delay Tradeoffs Involving Voltage Scaling in Synchronous and Asynchronous Circuits", Fourth ACiD-WG Workshop, Turku, Finland, 2004.

[IV.18] S. Karandikar, S. Sapatnekar, "Fast Comparison of Circuit Implementations", in Proceedings of DATE'04, March 2004.

[IV.19] J. Fragoso, G. Sicard, M. Renaudin, "Comparaison Rapide de Performance de Circuits Asynchrones QDI", in Recueil de Communications de FTFC'2005, Mai, 2005.

Chapitre V

Chemins de Données Asynchrones

L'existence d'innombrables outils et méthodologies de conception expliquent la large utilisation actuelle de circuits synchrones dans l'industrie des semi-conducteurs. En opposition, même si la communauté asynchrone a fait d'énorme progrès ces dernières années et malgré les potentialités des circuits asynchrones face aux circuits synchrones, le manque d'environnements capables d'automatiser la conception de circuits asynchrones freine l'adoption de tels circuits.

Ce chapitre présente une étude comparative des différents styles de circuits asynchrones et les résultats de cette étude serviront de recommandations aux outils de synthèse asynchrone. Ce chapitre propose également un algorithme pour la décomposition des portes logiques en vue de limiter l'entrance maximale de ces dernières et de réduire la consommation des circuits générés.

V.1. Contrôle et Chemin de Données

Un circuit synchrone peut normalement être divisé en deux parties : le contrôle et le chemin de données. Même si l'anatomie d'un circuit asynchrone est différente des circuits synchrones, il est encore possible d'identifier certaines parties du circuit avec des fonctions distinctes.

La communication entre deux circuits est assurée par des bascules capables d'implémenter un protocole choisi. Il est clair donc que l'implémentation des bascules varie en fonction du protocole choisi, mais sa fonction reste de mémoriser les données (valides et invalides) dans le flux de données.

La présence d'éléments de contrôle inconditionnel de flux de données, comme la fourche, la jonction et la fusion, assure un échange correct d'information entre plusieurs sources et plusieurs récepteurs. La Figure V.1 montre les implémentations de la fourche et de la jonction pour un protocole 4-phases et les codages de « données groupées » et « double rail » QDI [V.1]. La fourche et la jonction consistent à combiner adéquatement les signaux de requête et d'acquittement pour assurer le bon déroulement de la communication. De ce fait, ces éléments n'altèrent pas le flux de données. Pour raison de simplicité, les éléments de la Figure V.1 sont conçus avec deux canaux de sorties ou deux canaux d'entrées et chaque canal avec un seul digit. Bien sûr, ces éléments peuvent être généralisés pour plusieurs canaux avec plusieurs digits. Certains circuits asynchrones pour leurs implémentations requièrent des éléments plus complexes comme les contrôles conditionnels de flux de données (multiplexeurs et démultiplexeurs) et des arbitres.

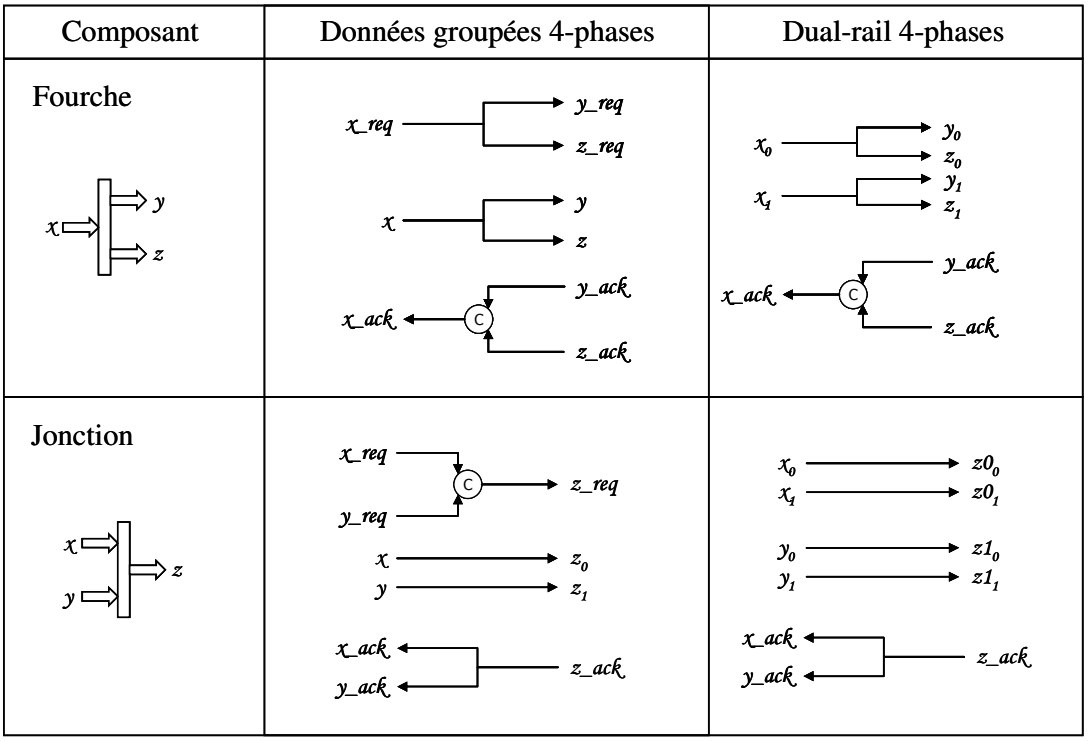


Figure V.1 – Implémentations de la fourche et de la jonction en données groupées 4-phases et double rail 4-phases.

Finalement, comme pour les circuits synchrones, il reste les parties opératives ou blocs logiques. Les blocs logiques sont responsables des calculs dans les chemins de données. Les blocs logiques sont des fonctions purement combinatoires et les sorties dépendent uniquement des valeurs d'entrées. Mais, pendant que les circuits synchrones doivent respecter une

contrainte de retard maximum, les blocs logiques asynchrones doivent être transparents aux protocoles de poignée de mains entre deux bascules.

La Figure V.2.a montre deux bascules connectées directement au travers d'un canal de communication. L'insertion d'un bloc logique entre ces deux bascules, comme montré dans la Figure V.2.b, ne doit pas altérer le fonctionnement du protocole, c'est-à-dire, du point de vue des deux bascules, la séquence de signalisation d'un canal est la même indépendamment de la présence d'un bloc logique dans le chemin de données.

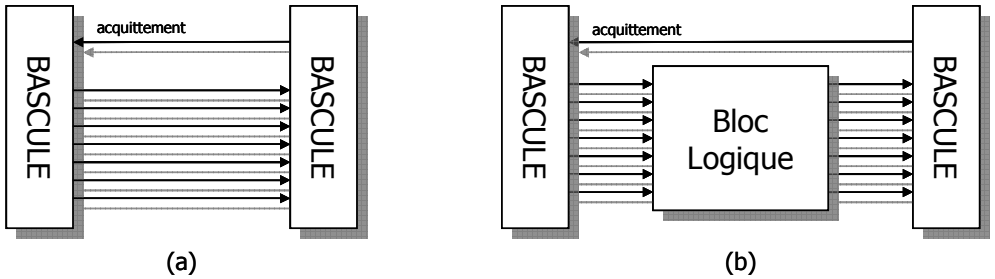


Figure V.2 – Deux bascules connectées (a) directement par un canal de communication et (b) avec un bloc logique.

Normalement, un bloc logique reçoit ses entrées par plusieurs canaux et envoie ses résultats à plusieurs canaux. L'utilisation d'entrées et sorties indépendantes implique une jonction à l'entrée du bloc logique et une fourche à la sortie. La Figure V.3 présente l'anatomie d'un additionneur asynchrone. Les éléments de jonction et de fourche peuvent être intégrés au bloc logique. Comme cette thèse s'intéresse particulièrement au chemin direct, il sera supposé que les blocs logiques reçoivent toutes leurs entrées par un même canal et produisent toutes leurs sorties dans un même canal. De plus, comme il a été discuté précédemment, seul les circuits QDI avec le codage 3-états seront abordés.

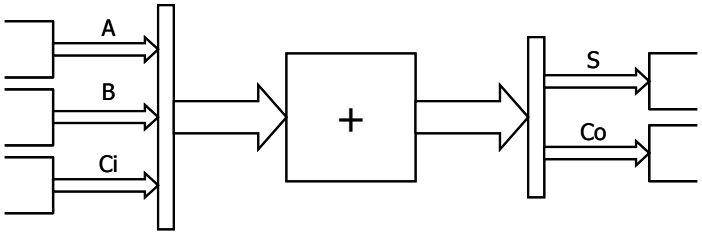


Figure V.3 – Anatomie d'un additionneur asynchrone.

V.1.1. Transparence au Protocole et Signalisation

Au-delà de réaliser une fonction définie, un bloc logique asynchrone QDI doit respecter la séquence des actions et les contraintes imposées par les circuits QDI. Comme il a été discuté

dans la section II.3, les circuits QDI utilisant un codage 3-états utilisent les données pour réaliser la signalisation, c'est-à-dire, les requêtes aux étages suivants.

Ainsi, un bloc logique QDI ne peut produire de requête à ses sorties qu'en présence de requête à ses entrées. Cela implique qu'une requête à la sortie d'un bloc logique doit impérativement indiquer que toutes les entrées sont valides et que tous les signaux internes (propriété d'insensibilité aux retards) et sorties ont été calculés –le circuit est stable. Dans les protocoles 4-phases la même contrainte est imposée à la remise à zéro du protocole. Donc, un bloc logique peut être caractérisé par sa transparence au protocole[V.1] comme étant « fortement indiqué » ou « faiblement indiqué ».

Un bloc logique *fortement indiqué* ne produit aucune sortie valide avant que toutes ses entrées soient valides. Donc, un bloc logique fortement signalé attend toutes ses entrées avant de démarrer le calcul. De façon complémentaire, le bloc logique ne produit aucune sortie invalide avant que toutes les entrées soient invalides. Donc, le bloc est fortement indiqué car n'importe quelle sortie indique la fin du calcul.

En opposition, un bloc logique *faiblement indiqué* commence à produire des sorties valides le plus tôt possible même si toutes les entrées ne sont pas encore valides et, de la même façon, il commence à produire des sorties invalides dès que possible. Cependant, pour qu'un bloc faiblement signalé fonctionne correctement, il ne doit pas produire toutes ses sorties valides avant que toutes les entrées soient valides et il ne doit pas invalider toutes ses sorties avant que toutes ses entrées soient invalidées. Ainsi, une sortie ne suffit pas pour indiquer la fin du calcul, toutes les sorties sont nécessaires pour réaliser l'indication.

En regardant l'implémentation de la fourche dans la Figure V.1, ces conditions assurent le bon fonctionnement du protocole vu que les acquittements des entrées seront envoyés en présence de tous les acquittements des canaux de sortie. Ce comportement a été qualifié par Seitz [V.2] de « condition faible ». En plus, Seitz a prouvé que toute combinaison valable de blocs logiques qui respectent une faible indication produit un nouveau bloc faiblement indiqué. Les combinaisons valables sont les structures où il n'y a pas d'entrée et/ou sortie non connectée et la structure ne crée pas de boucles. En conséquence, les blocs logiques fortement indiqués possèdent la même propriété, i.e., les combinaisons valables des blocs logiques fortement indiqués produisent un nouveau bloc logique fortement indiqué.

Les blocs logiques asynchrones présentent une hystérésis dans les transitions invalide vers valide et retour à invalide : (1) quelques – ou toutes – les sorties restent invalides jusqu'à ce que quelques – ou toutes – les entrées deviennent valides et (2) quelques – ou toutes – les

sorties restent valides jusqu'à que quelques – ou toutes – les entrées deviennent invalides. C'est ce comportement du type hystérésis qui assure la transparence vis-à-vis du protocole et en conséquence, même si les blocs logiques sont purement fonctionnels, des éléments de rétention d'état – normalement sous la forme de porte de Müller – sont nécessaires à la conception de tels modules.

Finalement, les blocs logiques fortement indiqués produisant plusieurs sorties présentent la latence de pire cas vu que toutes les entrées sont attendues avant le démarrage du calcul et donc l'utilisation de blocs logiques faiblement indiqués est nécessaire pour obtenir une latence réelle du circuit vis-à-vis de ses sorties.

V.2. Styles de Circuits

Dans la littérature, plusieurs styles de circuits asynchrones ont été proposés. Cette section compare les principales approches pour la conception asynchrone. Même s'il existe des méthodes de conception au niveau des transistors, comme les méthodes présentés en [V.1][V.3][V.4][V.5][V.6][V.7][V.8], cette section s'intéresse particulièrement aux méthodes pouvant être implémentées avec l'utilisation de cellules standard et qui ne nécessitent pas l'utilisation de compilateurs de cellules. Ainsi, les méthodes étudiées sont les méthodes qui peuvent se servir des outils de *back-end* commerciaux largement employés dans l'industrie.

V.2.1. DIMS

L'approche DIMS [V.9] – d'anglais « Delay-Insensitive Minterms Synthesis » – consiste à implémenter un bloc logique par une somme de produits. Les produits sont réalisés par des portes de Müller qui assurent la rétention de l'état et la transparence au protocole. Comme le propre nom l'indique, tous les minterms sont générés et ils sont combinés de façon adéquate pour produire les sorties. La Figure V.4.a présente l'implémentation DIMS pour un additionneur complet double rail.

Il est facile de voir par le type d'implémentation que les blocs logiques DIMS sont fortement indiqués vu que les portes de Müller commutent seulement en présence de toutes les entrées. Le style de circuits DIMS repose sur les travaux de David Müller [V.10][V.11] de la fin des années 50.

Cependant, les circuits DIMS peuvent être optimisés afin de les rendre faiblement indiqués. Dans le cas de l'additionneur de la Figure V.4.a, la valeur de la retenue de sortie est égale à zéro, si *A* et *B* sont égaux à zéro. Il est donc possible de simplifier le calcul de la

retenue en remplaçant les deux premiers minterms par une porte Müller qui teste *A0* et *B0*. Cette porte de Müller doit être ajoutée pour la fonction de calcul de la retenue, mais les minterms restent inchangés pour le calcul de la somme. Ainsi, la retenue peut être produite avant l'arrivé du signal de la retenue entrante, mais la somme est produite seulement à l'arrivé de tous les signaux. La Figure V.4.b montre cette version optimisée de l'additionneur qui transforme le comportement de l'additionneur en faiblement indiqué. Dans la Figure V.4.b, la même simplification a été faite pour la retenue sortante égale à 1.

Pour la construction correcte de circuits DIMS, une sortie doit toujours être calculée avec tous les minterms nécessaires à son calcul et cette sortie assure la transparence au protocole. Les autres sorties du circuit peuvent être simplifiés de façon à regrouper des minterms. Chaque fois qu'une nouvelle porte regroupant des minterms est insérée, les minterms regroupées sont enlevés du calcul de la sortie optimisée, mais ils restent pour le calcul de la sortie qui garantit le bon fonctionnement.

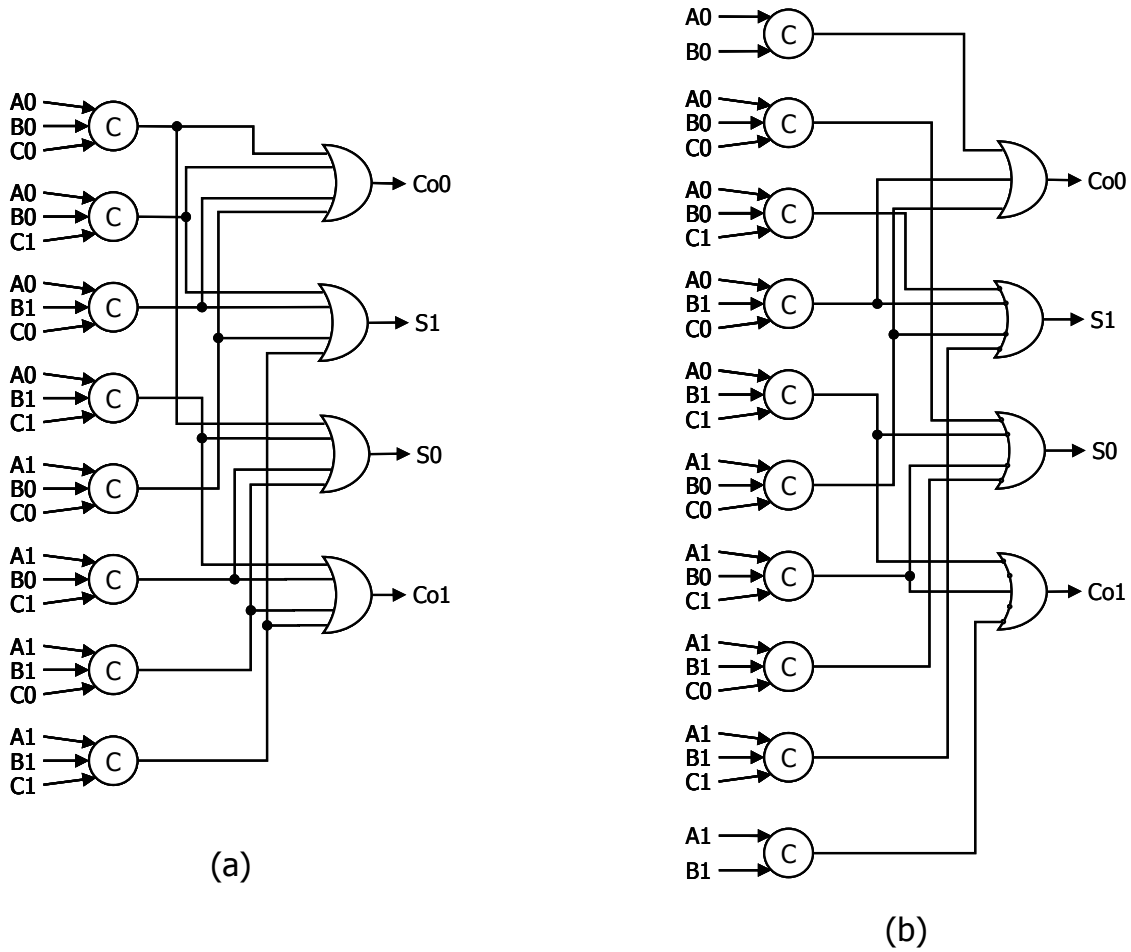


Figure V.4 – (a) Implémentation DIMS d'un additionneur complet double rail et (b) une version faiblement signalée.

Les circuits DIMS sont toujours insensibles aux délais car chaque transition d'une porte de Müller est observable à la sortie. De façon équivalente, chaque entrée d'une porte OU est exclusive à toutes les autres entrées de la même porte OU. Pour cette raison, l'optimisation d'une sortie, c'est-à-dire, le regroupement de minterms, doit respecter une seule règle : chaque nouveau regroupement doit seulement contenir des minterms qui ne sont pas couverts par d'autres portes de Müller participantes au calcul de la même sortie. Donc, chaque minterm doit être présent une seule fois dans le calcul d'une sortie et deux recouvrements ne doivent pas contenir un même minterm. Cette contrainte, qui est la seule imposée au moment de l'optimisation, est suffisante pour garantir que le circuit DIMS restera insensible aux délais car chaque transition continuera à être observable à la sortie.

La Figure V.5 présente la spécification CHP d'un décodeur 4→2 double rail et la Figure V.6.a présente son implémentation DIMS. Comme pour les circuits synchrones, pour les fonctions incomplètement spécifiées, seulement les minterms participant à la fonction de calcul sont générés. Les minterms sans influence et qui ne produisent pas de sortie – minterms « *dont care* » – n'ont pas besoin d'être implémentés.

```
process decoder
  port (
    A,B,C,D : in DR;
    S,T : out DR;
  )
  variable x,y,w,z : DR;
begin
[
  [ A?x,B?y,C?w,D?z;
    @[
      x="0" and y="0" and w="0" and z="1" => S!"1", T!"1"; break;
      x="0" and y="0" and w="1" and z="0" => S!"1", T!"0"; break;
      x="0" and y="1" and w="0" and z="0" => S!"0", T!"1"; break;
      x="1" and y="0" and w="0" and z="0" => S!"0", T!"0"; break;
    ];
    loop
  ];
];
end;
```

Figure V.5 - Spécification CHP d'un décodeur 4→2 double rail.

Cependant, les regroupements peuvent se servir de minterms non implémentés pendant l'optimisation et comme les minterms non implémentés ne sont jamais actifs, il est possible de les couvrir plusieurs fois si nécessaire. La Figure V.6.b montre la version optimisée et faiblement indiquée du décodeur 4→2. En regardant la sortie *S1*, elle est calculée avec les minterms *B1* et *D1*. Le circuit reste insensible aux délais car même si les couvertures de *B1* et *D1* ne sont pas exclusifs, l'indépendance est assurée vu que *B1* et *D1* sont exclusifs et les termes couverts plusieurs fois ne sont jamais actifs.

L'implémentation DIMS est facile à construire dû à sa régularité. En plus, la conception de circuits DIMS est facile à étendre à d'autres codages. Cependant, les implémentations asynchrones DIMS présentent un sensible surcoût en complexité quand elles sont comparées aux versions synchrones parce que chaque minterm est implémenté par une porte de Müller – élément de rétention d'état – et plusieurs fils sont utilisés pour le codage des valeurs, c'est-à-dire, la fonction et la fonction inverse doivent être implémentées dans le cas du codage double rail.

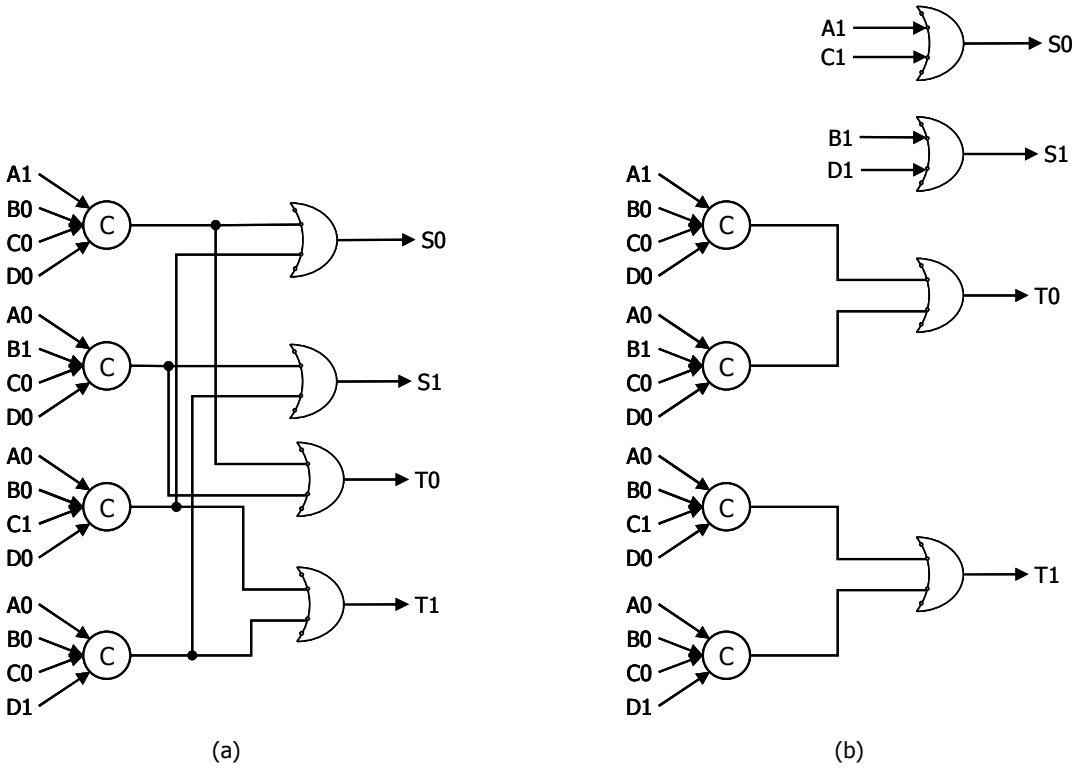


Figure V.6 –L'implémentation (a) DIMS fortement signalée pour un décodeur 4-2 et (b) la version optimisée faiblement signalée.

V.2.2. NCL

Les portes de Müller et les portes OU de la section précédente peuvent être vues comme des portes à seuil *n-parmi-n* et *1-parmi-n* avec un comportement du type hystérésis. L'idée proposée par Theseus Logic Inc., en [V.12], est d'utiliser des portes à seuil *m-parmi-n* avec hystérésis pour réduire la complexité. La sortie d'une porte à seuil *m-parmi-n* est égale à 1 quand au moins *m* entrées sont égales à 1 et la sortie est égale à zéro quand toutes les entrées sont égales à zéro, dans les autres cas, la sortie retient sa valeur.

La Figure V.7 montre un ensemble de portes logiques à seuil proposées par Theseus Logic Inc. Dans la Figure V.7, il est possible de s'apercevoir que les portes de Müller et les

portes OU sont des cas spéciaux des portes à seuil. La valeur à l'intérieur de chaque porte est le seuil de la porte et le seuil est compris entre 1 et le nombre d'entrées de la porte.

La méthodologie NCL – pour « NULL Convention Logic » – n'est pas vraiment différente du style DIMS. En fait, les portes à seuil sont les équivalentes asynchrones des portes complexes synchrones. La Figure V.8 présente l'implémentation faiblement indiquée d'un additionneur double rail en utilisant des portes logiques à seuil.

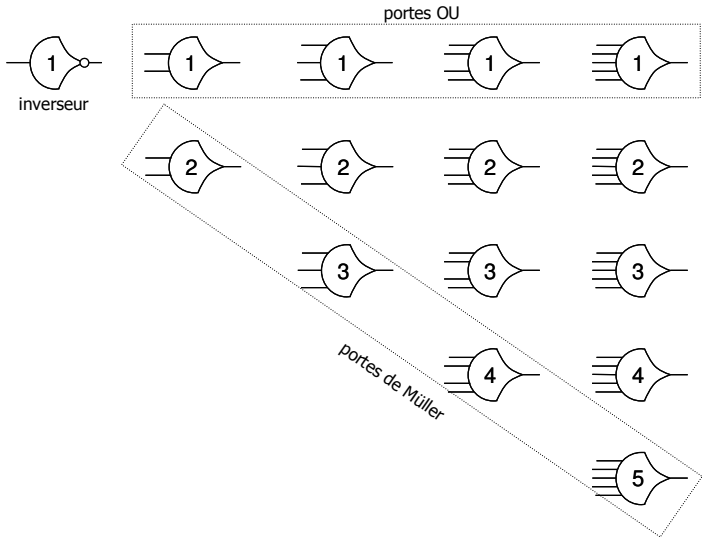


Figure V.7 – Portes logiques à seuil NCL.

Le lecteur ne doit pas se tromper avec l'apparente simplicité du circuit de la Figure V.8 parce que chaque porte à seuil comprend une certaine complexité de réalisation. La complexité du circuit de la Figure V.8 est estimée à une centaine de transistors tandis que l'additionneur complet DIMS faiblement signalée de la Figure V.4 contient 132 transistors. En plus, les portes logiques à seuil avec hystérésis ne sont pas contenues dans les bibliothèques de cellules standard.

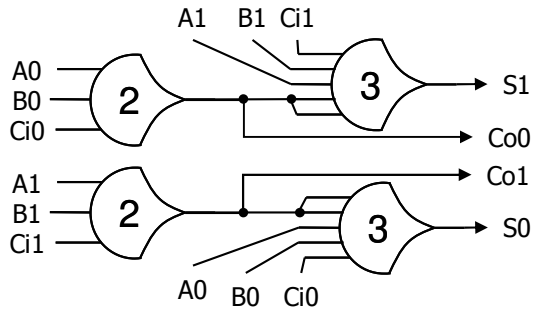


Figure V.8 – Un additionneur complet double rail avec des portes logiques NCL.

Cependant, la méthodologie NCL met en évidence le fait que la complexité d'un circuit asynchrone peut être réduite par l'utilisation de cellules complexes spécialement conçues pour leur réalisation. En plus, l'emploi des portes logiques à seuil réduit le nombre de points de rétention – points de mémoires – présentes dans le circuit. Alors que le circuit de la Figure V.4 contient 10 points de rétention (un point pour chaque sortie de porte de Müller), la version NCL contient seulement 4 points. Finalement, le style de circuit proposé par Theseus Logic Inc. n'est vraiment pas différent de la méthodologie DIMS, mais elle montre la nécessité d'effectuer une étape de projection technologique adéquate aux circuits asynchrones.

V.2.3. Signalisation Explicite

Les circuits du style ici appelé de signalisation explicite sont basés sur la méthodologie de conception présentée par David, Ginosar et Yoeli [V.13]. La Figure V.9 présente la structure générale de circuits avec signalisation explicite.

L'idée consiste à réaliser une fonction logique double rail avec les cellules standard. En fait, chaque fonction et sa version complémentaire sont implémentées par un réseau de portes logiques et ces fonctions peuvent être optimisées par les outils de conception standard vu qu'aucune contrainte n'est imposée à ce réseau logique. Le bloc « DRN » de la Figure V.9 représente la fonction combinatoire double rail.

Comme le bloc DRN tout seul n'est pas transparent au protocole vu que l'implémentation de ce bloc n'assure pas l'arrivée de tous les signaux d'entrée, cette condition doit être garantie par une signalisation explicite de la présence de tous les signaux. Le bloc « ORN » teste la validité, ou l'invalidité, de chaque signal d'entrée. Toutes les validités sont assemblées dans un même signal par le bloc « CEN » indiquant que tous les signaux d'entrée ont été reçus.

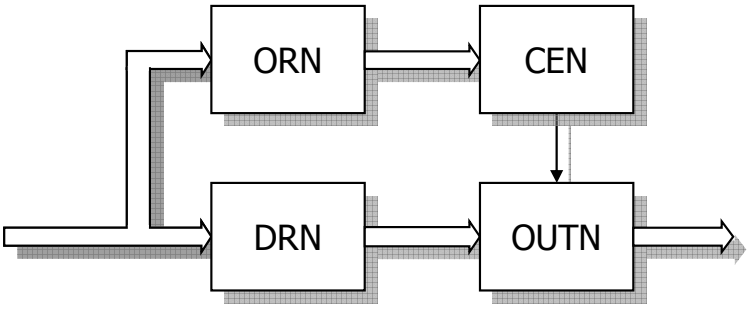


Figure V.9 – Structure générale des circuits avec signalisation explicite [V.13].

Le bloc « OUTN » bloque le passage des données à la sortie jusqu'à que le signal indiquant la présence de toutes les entrées soit produit. Le bloc « OUTN » sert aussi comme

élément de mémorisation pour créer le comportement du type hystérésis. Comme le bloc « OUTN » ne produit une sortie qu'à l'arrivé de tous les signaux d'entrée, les circuits avec cette structure sont fortement indiqués.

La Figure V.10 montre une implémentation de la fonction AND-3 double rail avec le style de circuit proposé par David, Ginosar et Yoeli. Dans ce circuit, chaque bloc de la structure générale peut être identifié.

Cette technique semble pouvoir réduire sensiblement la complexité des circuits asynchrones quand les fonctions sont complètement spécifiées et très peu de minterms sont nécessaires pour le calcul de la sortie ou une couverture efficace des minterms peut être réalisée. Les fonctions complètement spécifiées requièrent l'implémentation de tous les minterms dans les circuits DIMS et, en plus, si le circuit possède une seule sortie, il n'y a pas de place à l'optimisation pour ces derniers à part l'utilisation de cellules complexes asynchrones. L'implémentation de la Figure V.10 compte 60 transistors tandis que la version DIMS avec des portes standard nécessite 100 transistors.

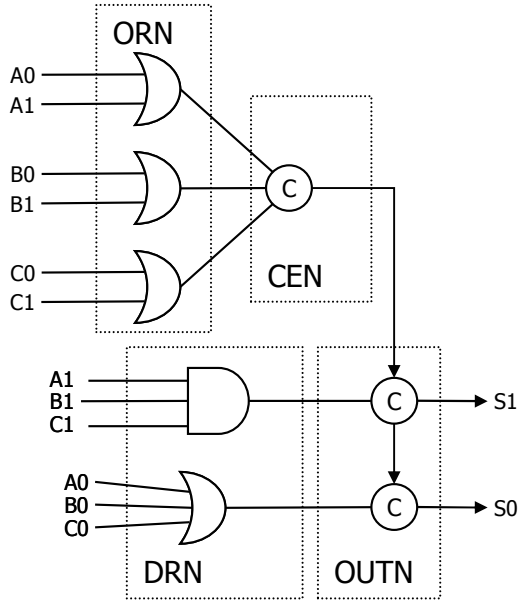


Figure V.10 – Implémentation d'un AND-3 double rail avec signalisation explicite.

Cependant, si la fonction ne peut pas être optimisée et/ou elle nécessite une grande partie des minterms, la méthodologie proposée par David, Ginosar et Yoeli cause un surcoût de complexité vu que la signalisation est réalisée par un circuit spécifique et redondant. La Figure V.11 présente l'additionneur complet double rail conçu avec signalisation explicite. Cet

additionneur compte 132 transistors et il utilise des portes logiques complexes alors que la version DIMS fortement indiquée sans aucune optimisation compte seulement 120 transistors.

En plus, la consommation des circuits avec signalisation explicite est plus importante que celle des circuits DIMS. Dans les circuits DIMS seulement un minterm et l'arbre de portes OU qui calcule la sortie commute, tandis que dans les circuits avec signalisation explicite toutes les portes des blocs « ORN » et « CEN » commutent en plus du minterm correspondant et des portes de Müller du bloc « OUTN » qui produisent la sortie. Les circuits DIMS consomment plus d'énergie seulement dans le cas où la réduction de complexité est très significative par l'utilisation d'une signalisation explicite. Cependant, la consommation dans les circuits DIMS se concentre principalement dans les nœuds d'entrées car la composition de tous les minterms génère une significative augmentation de la capacitance de ces nœuds.

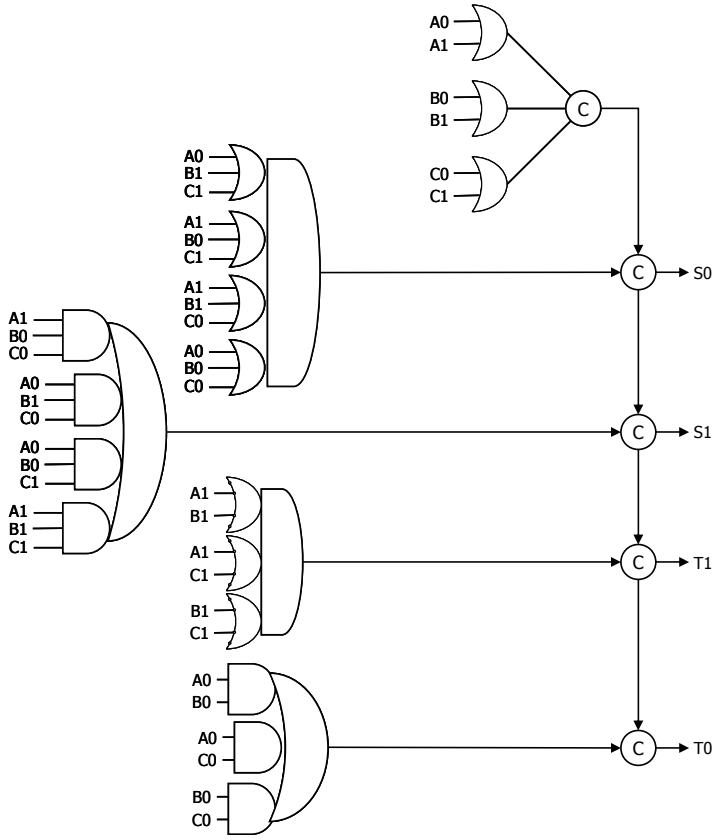


Figure V.11 – Un additionneur complet double rail avec signalisation explicite.

Également, le retard des circuits avec signalisation explicite n'est pas intéressant quand la réduction n'est pas significative. Si la fonction implémentée par le réseau combinatoire double rail est simple, cas où la réduction de la complexité peut être importante vis-à-vis des circuits DIMS, le retard du circuit est déterminé par le réseau de signalisation, c'est-à-dire, les

valeurs de sorties seront calculées avant que la signalisation s'effectue. Dans le cas opposé, le retard provoqué par le réseau double rail sera approximativement le même que le réseau DIMS, mais, pour les circuits avec signalisation explicite, il reste le bloc « OUTN » à traverser. Donc, dans les deux situations le retard attendu par les circuits avec signalisation explicite est équivalent au retard des circuits DIMS.

Deux autres problèmes affectent les circuits avec signalisation explicite : (1) les circuits sont toujours fortement indiqués et ne peuvent pas être modifiés, vu que le réseau combinatoire ne possède pas d'éléments de mémorisation pour implémenter le comportement du type hystérésis ; (2) le circuit obtenu n'est pas forcement insensible aux délais des portes logiques. En prenant l'exemple de l'additionneur double rail de la Figure V.11, ce circuit est insensible aux délais car chaque fonction est implémentée par une porte complexe. Mais le même circuit avec des portes logiques de base (portes ET et OU) n'est plus insensible aux délais. Si *A1*, *B1* et *C1* sont vrais, le résultat de la somme est égal à 1, mais les portes OU qui contiennent *A1*, *B1* et *C1* dans la fonction de *S0* commutent et les transitions de ces portes ne sont pas transmises à la sortie. Donc, la sortie valide n'assure pas que tous les nœuds internes du circuit sont stables. En [V.13], ce problème est discuté et les circuits construits par leur méthodologie sont corrects pour un mode de fonctionnement fondamental (cf. §II.4.2), mais la construction n'assure pas l'insensibilité aux délais.

D'autres travaux, comme [V.14] [V.15] [V.16], proposent des solutions aux problèmes des circuits explicitement signalés, mais l'idée est toujours la même : utiliser des signaux explicites, et redondants, pour signaler la validités des canaux. Cependant, pour transformer le circuit en faiblement indiqués, le réseau « OUTN » est enlevé du chemin du circuit et le réseau double rail, bloc « DRN », est conçus en utilisant des portes avec mémorisation – portes de Müller et portes à seuil – pour réaliser la rétention d'état.

Cette transformation crée un circuit qui n'est pas transparent au protocole car toutes les sorties peuvent être validées avant l'arrivée de toutes les entrées. Pour garantir le fonctionnement correct, la signalisation de la présence de toutes les entrées – sortie du bloc « CEN » – n'est plus utilisée pour bloquer l'avancement des données, mais ce signal est utilisé dans le chemin retour pour bloquer l'acquittement provenant de l'étage suivant. Donc, le circuit est faiblement signalé car les sorties sont produites le plus tôt possible et la transparence est assurée par la modification du chemin de retour. En plus, la stabilité de tous les signaux internes est aussi testée et combinée avec le signal de validité des entrées. Cette nouvelle

signalisation garantit la présence de toutes les entrées et que tous les nœuds internes au circuit ont été calculés et, par conséquence, le circuit produit est insensible aux délais des portes.

La Figure V.12 montre le résultat pour un décodeur 4→2 avec signalisation explicite et faiblement indiqué. Dans le circuit de la Figure V.12, les portes logiques contenant un *T* sont les portes logiques à seuil. Un signal *x.go* est un signal indiquant explicitement que le signal *x* est valide ou invalide. La validité d'un signal n'est pas générée par le bloc lui-même, mais elle doit être transmise avec les données. De ce fait, un fil supplémentaire est utilisé pour le codage des données. Le signal de sortie *done* est la signalisation du circuit.

En termes de retard, les circuits avec signalisation explicite et faiblement indiqués sont meilleurs que les autres styles car toutes les sorties sont produites le plus tôt possible, mais un contrôle doit être ajouté dans le chemin de retour. Cependant, il est facile de s'apercevoir que ces circuits sont plus complexes que les circuits obtenus par la structure de la Figure V.9 car tous les signaux internes sont aussi explicitement signalés. En plus, ces circuits présentent une consommation d'énergie plus importante que toutes les autres versions étant donné que, au-delà de la fonction de calcul, tout le chemin responsable de la signalisation doit commuter et aussi la signalisation de nœuds internes, même si les nœuds internes ne participent pas au calcul d'une opération donnée.

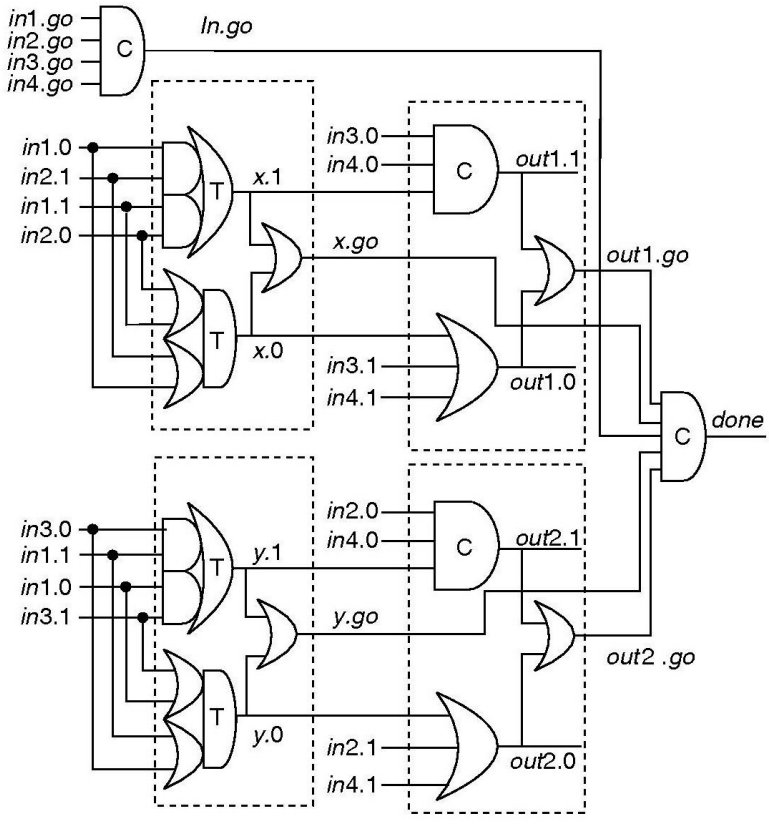


Figure V.12 – Un décodeur 4→2 avec signalisation explicite et faiblement indiqué [V.16].

Il faut aussi souligner que la combinaison des circuits avec signalisation et faiblement indiqués s'avère plus complexe que les autres styles. Dans Seitz [V.2], les circuits faiblement indiqués respectent la règle de ne pas produire toutes les sorties avant de recevoir toutes les entrées. Les circuits proposés en [V.14] [V.15] [V.16] violent cette règle et la composition de tels circuits doit prendre en charge la combinaison correcte des signaux de signalisation pour assurer le bon fonctionnement du protocole.

Finalement, le but principal de la signalisation explicite est de réduire la complexité des circuits asynchrones par l'optimisation du réseau de calcul. Cela est atteint par l'utilisation d'outils de conception synchrones standard. En fait, chaque fonction combinatoire est décrite comme pour les circuits synchrones et son implémentation est obtenue avec les outils standard. A partir du circuit combinatoire synchrone, la fonction dual est construite pour obtenir le réseau double rail. Dans [V.16], les portes ET sont remplacées par des portes de Müller et les portes complexes par des portes à seuil. En plus, chaque nœud interne du circuit combinatoire original est combiné avec sont dual pour réaliser la signalisation des nœuds internes. Ainsi, le réseau double rail peut être construit et optimisé par les outils de conception synchrones standard. Cependant, même si la structure de circuits avec signalisation explicite est applicable à tous les circuits asynchrones, il est par contre difficile d'étendre les méthodologies de conception présentées en [V.13], [V.14], [V.15] et [V.16] à tous les codages de données possibles.

V.2.4. Résultats

Plusieurs circuits ont été générés et leurs retard, consommation et complexité ont été estimés en utilisant la méthode présentée précédemment (cf. Chapitre IV). Le 0 montre la synthèse de ces résultats. Tous les circuits ont été générés sans l'utilisation de portes complexes, seulement les portes logiques de base – ET, OU, Müller – ont été employées. Pour éviter de pénaliser les circuits avec signalisation explicite, la signalisation des nœuds internes n'a pas été conçue. Ainsi, les version avec signalisation explicite – lignes SE du tableau – sont conformes à la structure générale proposée en [V.13]. Le style NCL est propriété de Thesus Logic Inc. et ils n'ont pas pu être comparés. Cependant, du à sa similitude au style DIMS, les résultats DIMS doivent être la borne supérieure de complexité et retard des circuits NCL.

La Figure V.13 présente la comparaison de complexité entre les différents styles de circuits. Comme il a été attendu, la complexité des circuits DIMS croit avec l'augmentation du nombre de minterms implémentés. Cependant, pour les fonctions avec un nombre d'entrées

réduites, le surcoût de complexité infligé par la signalisation explicite fait que les circuits DIMS restent toujours l'option souhaitable.

Tableau V.1 – Comparaison entre les styles de circuits asynchrones.

Circuit		Retard			Consommation Moyenne				#xtors
		Max	Moyen	Min[1]	Totale	Nœuds d'entrée	Nœuds Internes	Nœuds de Sortie	
AND2	DIMS	14,56	13,17	9,66	27,75	8,00	11,75	8,00	40
	SE	20,00	20,00	14,40	50,21	6,33	35,88	8,00	48
AND3	DIMS	23,33	19,17	17,84	64,75	36,00	20,75	8,00	100
	SE	21,67	21,67	15,82	67,38	11,00	48,38	8,00	60
AND4	DIMS	24,33	23,29	18,84	162,94	128,00	26,94	8,00	240
	SE	23,33	23,33	17,15	87,57	16,67	62,91	8,00	72
AND16	DIMS[2]	64,67	64,67	42,30	784,24	512,00	264,24	8,00	1200
	SE	31,67	31,67	24,82	274,79	58,67	208,13	8,00	272
AND2-4R[3]	DIMS	21,22	17,54	15,93	42,31	16,00	18,31	8,00	172
	SE	24,00	24,00	15,68	58,40	12,83	37,56	8,00	140
XOR2	DIMS	13,44	13,44	8,71	28,17	8,00	12,17	8,00	44
	SE	20,00	20,00	14,40	58,83	8,67	42,17	8,00	72

[1] Retard Minimum Atteignable après dimensionnement des portes
[2] fonction réalisée avec deux étages de AND4 DIMS
[3] fonction ET avec deux entrées et une sortie codées en *1-parmi-4*

A partir de 4 entrées et pour certaines fonctions l'implémentation avec signalisation explicite peut devenir intéressante. Mais, il faut pouvoir assurer que cette implémentation restera insensible aux délais et cette tache peut s'avérer non triviale et difficile à garantir avec des outils synchrones standard et avec l'utilisation d'autres codages pour les données.

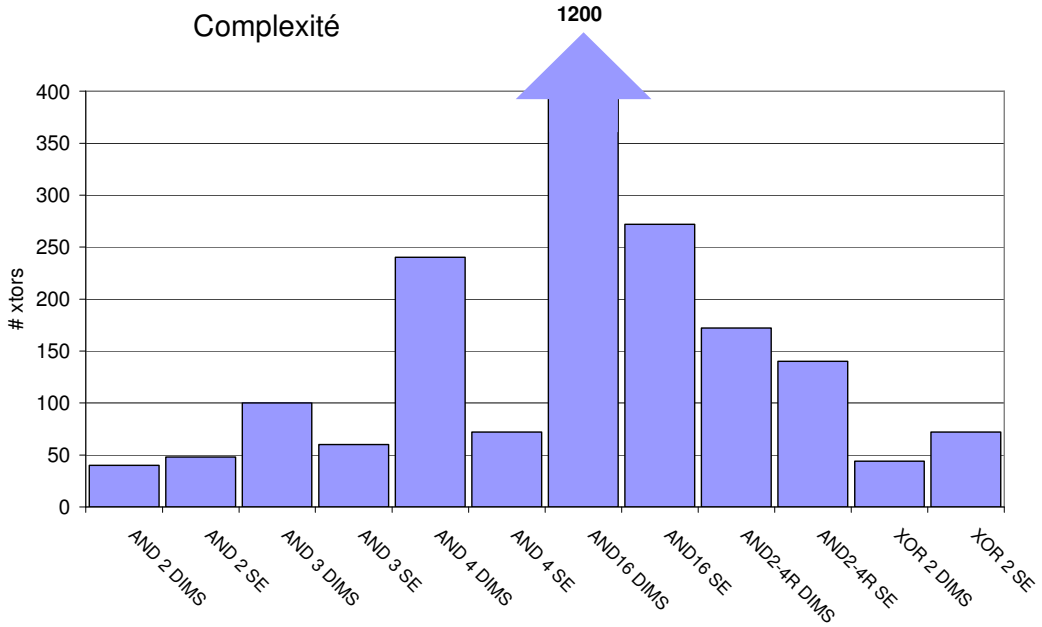


Figure V.13 – Comparaison de la complexité des différents circuits.

Comme il a été prévu, le retard des fonctions semble équivalent pour les deux styles de circuits. En plus, le retard moyen des circuits DIMS présente une valeur inférieure au retard critique vu que le retard varie en fonction des valeurs d'entrée. Pour les circuits avec signalisation explicite, le retard semble être limité par le circuit de signalisation et, donc, le retard est toujours le même indépendamment des valeurs d'entrée. La Figure V.14 montre les résultats de retard pour les différents circuits. Le circuit AND16 DIMS a été conçu avec deux étages de AND4 et cela explique la sensible augmentation du retard pour ce circuit.

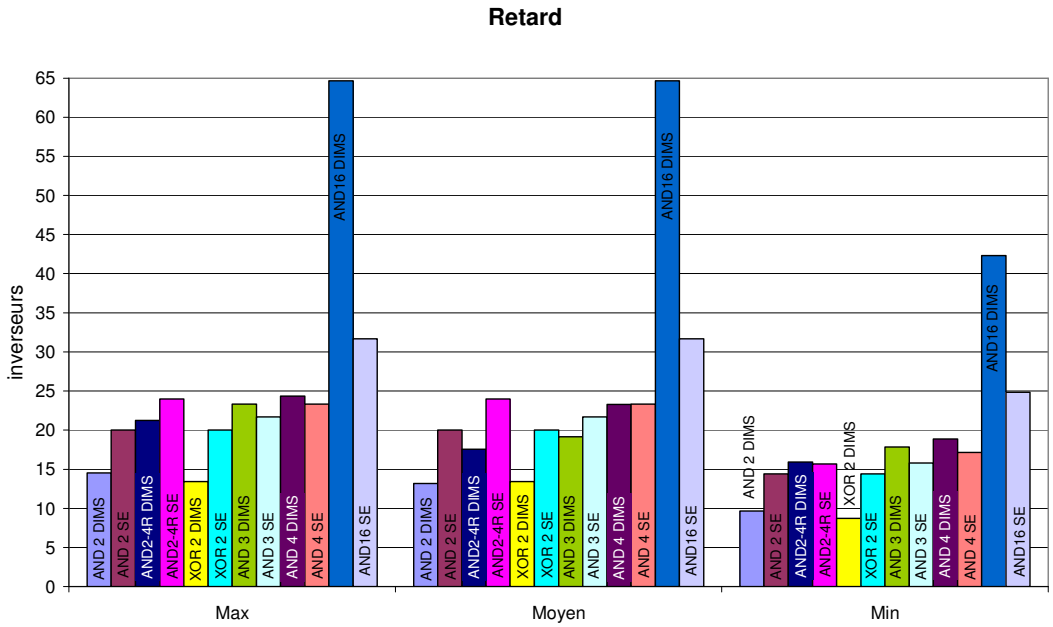


Figure V.14 – Comparaison du retard des différents circuits.

La Figure V.15 montre la consommation moyenne d'énergie pour les différents circuits. Comme il a été discuté, la consommation moyenne pour les circuits DIMS est inférieure aux circuits avec signalisation explicite, à l'exception des deux circuits où le rapport de complexité entre les circuits DIMS et les circuits avec signalisation explicite est supérieur à 3. Mais, il faut souligner que la consommation totale prend en compte la capacitance d'entrée du circuit et les capacitances de charge. En regardant seulement la consommation des nœuds internes du circuit, les circuits DIMS se montrent encore plus performants. Ces résultats montrent la seule difficulté d'utilisation des circuits DIMS. Avec l'augmentation du nombre d'entrée, au delà de 3, le nombre de minterms à implémenter par le circuit est élevé. Cela implique une sensible augmentation de la complexité, mais, plus grave et plus limitante, les capacitances des nœuds d'entrée deviennent significatives faisant des circuits DIMS des charges importantes à piloter.

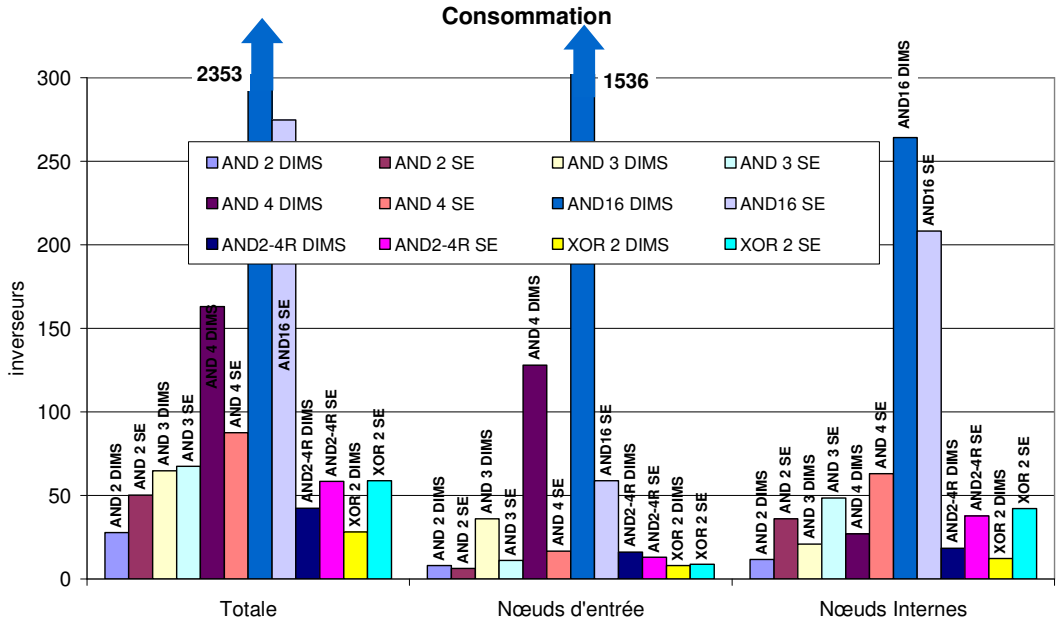


Figure V.15 – Comparaison de consommation d'énergie moyenne des différents circuits.

La Figure V.16 montre la distribution de la consommation d'énergie pour les différents nœuds des différents circuits. La charge est constante indépendamment du circuit. La charge est exprimée dans le graphique de la Figure V.16 pour permettre au lecteur d'identifier les circuits plus consommants. Plus la consommation augmente, plus la contribution de la charge de sortie diminue.

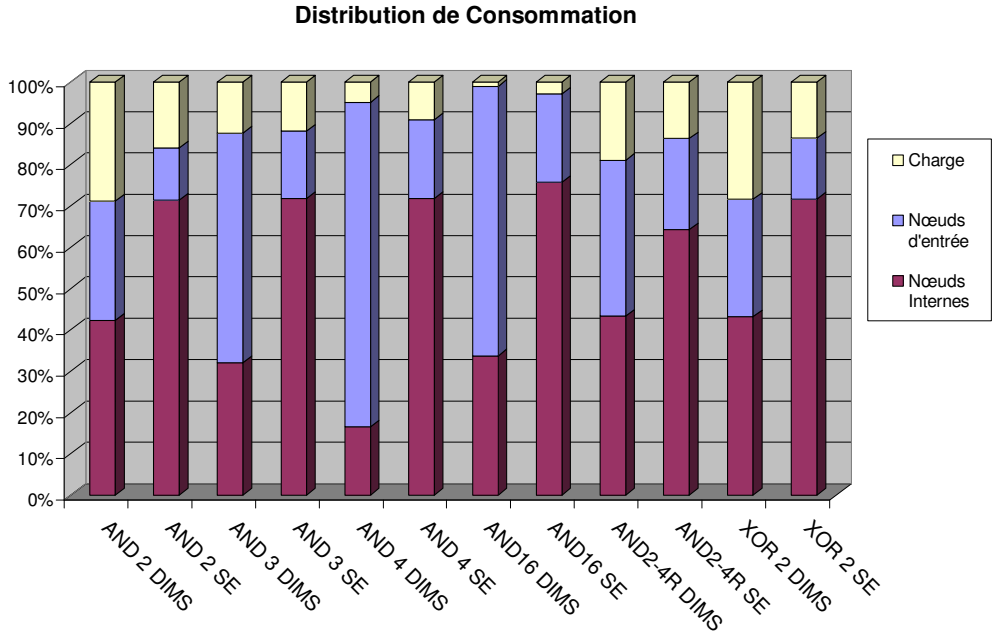


Figure V.16 – Distribution de consommation entre les différents nœuds des circuits.

Par les résultats de la Figure V.16, il est possible de conclure que la contribution majeure des circuits DIMS réside dans les nœuds d'entrée. En opposition, les nœuds internes des circuits avec signalisation explicite sont la principale source de consommation vu que le circuit de signalisation commute pour toutes les opérations réalisées.

Ces résultats laissent conclure que les circuits DIMS sont préférables à la conception asynchrone quand le nombre des entrées est limité, ou quand il est possible de décomposer la fonction à implémenter de façon à limiter la complexité des blocs logiques. En plus, les circuits DIMS sont plus faciles à concevoir du à leur régularité et les blocs logiques DIMS peuvent être facilement combinés pour obtenir des fonctions plus complexes.

Les circuits évalués ont été conçus avec des cellules standard, mais la complexité peut être réduite par l'utilisation de cellules complexes asynchrones. Même si les circuits avec signalisation explicite semblent contrôler la complexité au dépend de la consommation et du retard, cela souligne l'intérêt de concentrer les efforts pour concevoir une bibliothèque de cellules pour la conception des circuits asynchrones à faible consommation d'énergie [V.18][V.19].

V.3. Décomposition des Portes Logiques

Pour raisons électriques, le nombre des transistors en série dans une porte logique CMOS est limité à 4 transistors. Cela limite l'entrance des portes logiques et, dans les circuits, il est souvent nécessaire de décomposer une porte logique en portes plus simples capables d'être implémentées en logique CMOS. Cette section propose une méthodologie de décomposition de portes logiques visant à réduire la consommation des circuits asynchrones. Les principales applications sont les portes OU des circuits DIMS, les circuits de détection de fin de calcul et les portes de Müller avec des multiples entrées.

La discussion sera concentrée sur la décomposition d'une porte OU à plusieurs entrées où seulement une entrée commute – c'est-à-dire que les entrées sont mutuellement exclusives. Comme il sera montré par la suite, les autres situations peuvent être traitées par le même algorithme.

V.3.1. La Méthode de Décomposition

Une porte OU avec m entrées doit être décomposée par un arbre des portes OU ayant une entrance maximale f. Chaque entrée x_i possède une probabilité de commuter égale à P_i et seulement une entrée commute à la fois, donc :

$$Pi \cdot Pj = 0 \forall i \neq j \quad \text{(V-1)}$$

Le but est de minimiser l'énergie moyenne consommée par l'arbre. Ainsi, les entrées plus fréquentes doivent traverser le plus petit nombre de portes en vue de réduire le nombre de nœuds qui commutent. Il faut souligner que cette approche réduit aussi le retard moyen de l'arbre.

Ce problème est isomorphe au problème de coder un ensemble de messages à transmettre, où la probabilité de chaque message est connue. Les messages sont codés par une séquence de symboles d'un alphabet fini et aucun message ne peut être le préfixe d'un autre message. Ainsi, un message correspond à un signal d'entrée et la taille de l'alphabet correspond à l'entrance maximale de la porte logique. À partir de cette formulation, l'algorithme de codage de Huffman[V.20] peut être employé pour déterminer l'arbre optimal de portes OU.

V.3.2. L'Algorithme

Huffman a proposé un algorithme glouton qui construit un code optimal avec préfixe libre appelé Code de Huffman. L'algorithme crée un arbre T d'une façon « *bottom-up* » qui correspond au code optimal. L'algorithme commence avec un ensemble X avec $|X|$ nœuds feuilles et chaque nœud x_i avec une probabilité $P_i=p(x_i)$.

Une file de priorités Q est créée. Cette file est ordonnée de façon croissante de telle sorte que $Pi < Pj \forall i < j$. Chaque fois qu'un nœud est créé, ce nœud est inséré dans la file de priorité en gardant l'ordre de la file.

```
Algoritme_HUFFMAN ( X, f )
  Q ← X
  tant que | Q | > 1 faire
    z ← NOUVEAU_NODE( )
    P_z ← 0
    pour i de 0 jusqu'à f-1 faire
      x ← EXTRAIRE_MIN ( Q )
      fils_i [ z ] ← x
      P_z ← P_z + p(x)
    fin pour
    INSERER ( z, Q )
  fin tant
  retourner EXTRAIRE_MIN ( Q )
```

Figure V.17 - Algorithme de Huffman.

Il est facile de démontrer que le nombre de nœuds *g* de l'arbre qui sont créés par l'algorithme est égal à :

$$g = \left\lceil \frac{m-1}{f-1} \right\rceil \tag{V-2}$$

où *m* est le nombre de nœuds feuilles initial et *f* est le nombre maximal de fils d'un nœud. Donc, *g* est le nombre minimum nécessaire de portes logiques pour implémenter la fonction avec *m* entrées et avec une entrance maximale *f* pour chaque porte. La Figure V.18 montre les étapes de construction de l'arbre par l'algorithme de Huffman pour une entrance maximale égale à *4* et avec 10 nœuds. La probabilité de chaque nœud est écrite à son intérieur.

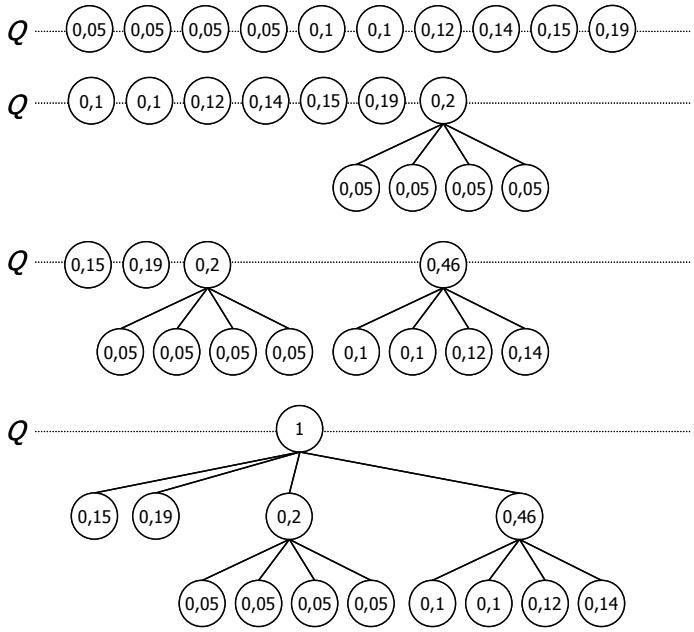


Figure V.18 – Exemple de construction de l'arbre avec l'algorithme de Huffman.

Comme il a été discuté dans le chapitre précédent, la consommation d'énergie moyenne est donnée par la somme sur tous les nœuds du produit de la capacitance et de la probabilité. Les nœuds du circuit correspondent aux arcs de l'arbre, cependant, il est possible de concentrer la consommation dans les nœuds internes de l'arbre par :

$$\begin{aligned} E_i &= \sum_{j}^{\text{fils de } i} P_j \cdot C_{in}^i + P_i \cdot C_{parasite}^i \\ P_i &= \sum_{j}^{\text{fils de } i} P_j \\ E_i &= P_i\left(C_{parasite}^i + C_{in}^i\right) \end{aligned} \tag{V-3}$$

où C_{in}^i est la capacitance d'entrée de la porte logique, $C_{parasite}^i$ est la capacité parasite et E_i l'énergie moyenne consommée par le nœud *i*. En supposant les tailles minimales de portes

logiques du chapitre précédent, les capacitances sont une fonction linéaire du nombre d'entrées de la porte logique. Ainsi, l'énergie moyenne d'un nœud peut être définie par :

$$\begin{aligned} E_i &= P_i C(n) \\ C(n) &= \alpha \cdot n + \beta \end{aligned} \qquad \text{(V-4)}$$

où *C(n)* est la fonction coût de consommation d'une porte logique et α et β sont fonction de la porte logique et *n* est le nombre d'entrées. Les paramètres α et β sont, respectivement, les capacités d'entrée et parasite de la porte. Par le modèle du Chapitre IV, les paramètres α et β sont plus grands ou égaux à 1. Le coût d'un nœud feuille est égal à zéro. En conséquence, l'énergie moyenne totale est exprimée par :

$$E = \sum_i E_i = \sum_i P_i \cdot C(n_i) \qquad \text{(V-5)}$$

Les arbres complets construits par l'algorithme de Huffman sont optimaux. Il est facile de s'apercevoir que l'arbre de la Figure V.18 est la meilleure solution. Pour tous les autres arbres l'énergie totale est supérieure ou égale à l'énergie moyenne totale consommée par l'arbre de la Figure V.18. Par exemple, si le nœud avec la probabilité 0,19 est échangé contre un nœud de probabilité 0,05, cela augmente la probabilité du nœud 0,2 à 0,34 et la probabilité de tous les autres nœuds reste inaltérée. Donc, l'énergie moyenne totale est augmentée.

Cependant, si le rapport $(m-1)/(f-1)$ n'est pas entier, l'arbre construit n'est pas complet, c'est-à-dire que certains nœuds de l'arbre posséderont un nombre de fils inférieur à *f*. Ainsi, la solution proposée par l'algorithme n'est pas forcement optimale. L'étude de l'optimisation de l'arbre initial se limitera aux cas où $f \leq 4$ car l'entrance maximale d'une porte logique de base est toujours inférieure ou égale à 4.

Si l'arbre n'est pas complet, le nombre d'entrées manquantes est égal à $(m-1) \bmod (f-1)$ et, si $f \leq 4$, le nombre maximal d'entrées non utilisées est égal à 2. Il est facile de conclure que les arbres binaires sont toujours complets. En utilisant l'algorithme de Huffman le nœud incomplet est la racine de l'arbre. En plus, un arbre optimal possède la propriété qu'un nœud dans un niveau de l'arbre est plus probable que n'importe quel nœud d'un niveau inférieur. Les niveaux sont comptés à partir de la racine de l'arbre. Cette propriété est facile de vérifier par le propre algorithme de construction. Il est aussi facile à vérifier qu'un arbre qui ne respecte pas cette propriété n'est pas optimal, car, en supposant un nœud au niveau *l* avec une probabilité inférieure à un nœud au niveau *l*+1, la permutation de ces deux nœuds n'altère pas les coûts des deux nœuds aux quelles ils appartiennent et ni la probabilité du nœud

plus supérieur, mais la permutation réduit la probabilité du niveau *l*+1 et, par conséquence, la permutation réduit le coût total.

Pour trouver l'arbre optimal pour un arbre incomplet, une première solution est construite avec l'algorithme de Huffman. Ensuite, les nœuds seront permutés pour essayer de réduire le coût total. Pour démontrer la procédure, l'exemple général de la Figure V.19 est étudié. L'arbre de la Figure V.19 est construit à partir de l'ensemble ordonné $X=\{P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8\}$.

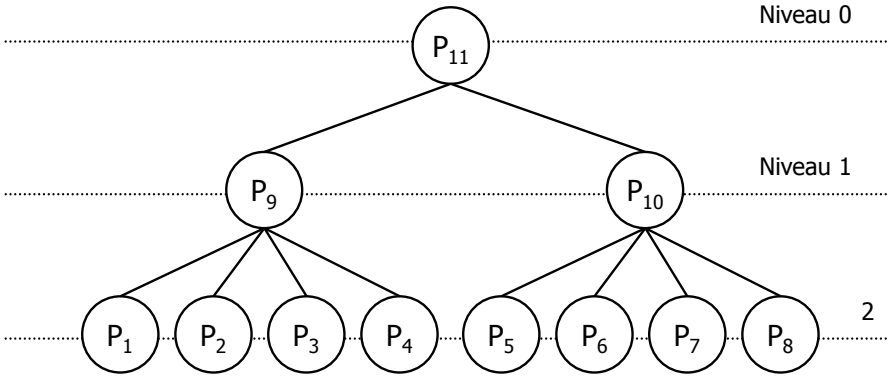


Figure V.19 - Solution initiale de l'algorithme de Huffman pour un arbre incomplet.

Par la construction de l'arbre, les nœuds P_9, P_{10} et P_{11} sont les nœuds internes créés par l'algorithme, et il est possible de définir que :

$$Pi \leq Pj \forall i < j \tag{V-6}$$

Il reste deux places au nœud P_{11}, donc, il est envisageable de monter un des huit nœuds du niveau 2 au niveau 1 de l'arbre. À ce moment, la seule possibilité est le nœud P_8, puisque, si un autre nœud est élevé au niveau 1, ce nœud pourra être permuté avec le nœud P_8 par la suite en réduisant le coût.

Il reste à trouver quelle est la condition permettant au nœud P_8 de monter. Le coût total actuel de l'arbre est donné par :

$$E_0 = P_{11} \cdot C(2) + P_{10} \cdot C(4) + P_9 \cdot C(4) \tag{V-7}$$

où $C(n)$ est la fonction coût à n entrées de l'équation V-4. Si le nœud P_8 monte d'un niveau, le nouveau coût total est égal à :

$$E_1 = P_{11} \cdot C(3) + (P_{10} - P_8) \cdot C(3) + P_9 \cdot C(4) \tag{V-8}$$

Pour effectuer cette altération dans l'arbre, il faut que le nouveau coût soit inférieur au coût de la solution initiale, ainsi :

$$\begin{aligned} & E_0 - E_1 > 0 \\ & \Rightarrow P_{11} \cdot C(2) + P_{10} \cdot C(4) + P_9 \cdot C(4) - P_{11} \cdot C(3) - (P_{10} - P_8) \cdot C(3) - P_9 \cdot C(4) > 0 \\ & \Rightarrow P_{11} \cdot \big(C(2) - C(3)\big) + P_{10} \cdot \big(C(4) - C(3)\big) + P_8 \cdot C(3) > 0 \end{aligned} \quad \text{(V-9)}$$

$$\Rightarrow -\alpha \cdot P_{11} + \alpha \cdot P_{10} + P_8 \cdot C(3) > 0 \quad \text{(V-10)}$$

$$\Rightarrow P_8 > \frac{\alpha \cdot (P_{11} - P_{10})}{C(3)} \quad \text{(V-11)}$$

À partir des équations V-9, V-10 et V-11 certaines constatations peuvent être faites. Comme le but est de minimiser le coût total, l'intérêt est de maximiser l'équation V-10 et, comme P_{11} est constante, P_{10} et P_8 sont les nœuds qui maximisent cette équation. De façon inverse et comme il a été attendu, si P_8 ne satisfait pas l'équation V-11, aucun autre nœud ne peut pas satisfaire la condition de réduction de coût. Il est possible de généraliser la condition de monter un nœud de niveau par :

$$P_x > \frac{\alpha \cdot (P_K - P_J)}{C(n-1)} \quad \text{(V-12)}$$

où P_x est le plus grand fils de P_J, P_J est fils de P_K et n est le nombre initial de fils de P_J avant le changement.

Si le changement ne peut pas s'effectuer, la solution initiale est la solution optimale. La Figure V.20 montre le graphe d'état de l'arbre et les arcs contiennent les conditions pour changer d'état. Si le nœud monte d'un niveau, l'arbre de l'état 1 de la Figure V.20 est obtenu et $P'_{10}=P_{10} - P_8$.

Maintenant, il reste une place disponible sur le nœud P_{11} et une place sur le nœud P'_{10}. En supposant qu'un nœud Px fils de P_9 essaie de prendre place sur le nœud P'_{10}, de façon analogue le coût doit être réduit. Le coût de l'arbre de l'état 1 de la Figure V.20 est donné par l'équation V-8 et le coût après le déplacement d'un fils de P_9 vers P'_{10} est exprimé par :

$$E_{P_9 \to P'_{10}} = P_{11} \cdot C(3) + (P_{10} - P_8 + P_x) \cdot C(4) + (P_9 - P_x) \cdot C(3) \quad \text{(V-13)}$$

En réalisant la soustraction et après les mêmes manipulations effectuées précédemment, la condition pour transférer un nœud de P_9 vers P'_{10} est donné par :

$$P_x < P_9 - P'_{10} \quad \text{(V-14)}$$

Comme toutes les probabilités sont positives, si la probabilité de P'_{10} est encore supérieure à la probabilité de P_9, la différence du côté droit de l'équation V-14 est négative et aucun fils de P_9 ne peut être déplacé sans augmenter le coût total. C'est-à-dire, il n'est pas concevable d'augmenter le coût et encore plus la probabilité d'un nœud possédant déjà une probabilité supérieure pour un même niveau. Inversement, les places libres doivent rester dans les nœuds plus probables sur un même niveau.

Maintenant, si P'_{10} est inférieur à P_9 à cause du déplacement du nœud P_8, il faut que $P'_{10}+P_x < P_9$ soit satisfaite. Cependant, comme les fils de P'_{10} ont des probabilités supérieures ou égales aux fils de P_9, cette condition ne peut pas être satisfaite non plus, car P'_{10} augmenté de n'importe quelle fils de P_9 sera toujours plus grand que P_9. Ainsi, après le déplacement de P_8 la seule alternative qui se présente est d'essayer de monter à nouveau un autre nœud, vu que les déplacements au même niveau ne sont pas encore possibles.

L'étape suivante est de tester si P_7 et/ou P_4 satisfont la condition de l'équation V-12. Si les deux nœuds peuvent monter d'un niveau, le nœud provocant la plus grande réduction de coût est choisi. Ainsi, l'état 2 ou l'état 3 du graphe de la Figure V.20 est atteint.

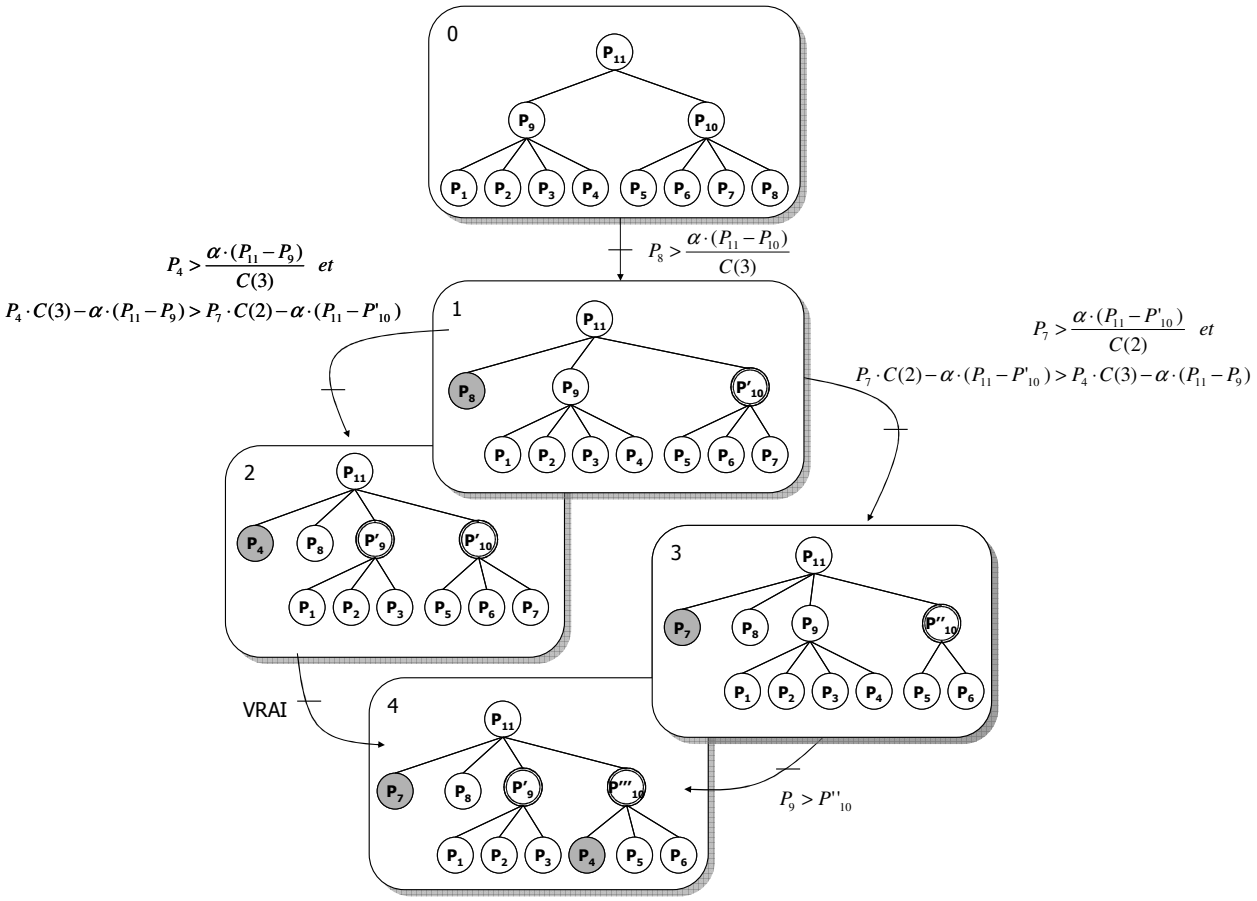


Figure V.20 – Graphe d'états de l'arbre.

Dans le cas où le nœud P_4 s'avère plus avantageux, le nœud P_4 monte d'un niveau et, par la propriété de l'arbre, il est possible, par la suite, de commuter le nœud P_4 avec le nœud P_7 avec une réduction du coût total. Ainsi, il est toujours possible d'atteindre l'état 4 à partir de l'état 2. Par induction, le nœud P_4 ne doit pas revenir à *P9*, car si l'état 4 est atteint l'état 4 offre une réduction de coût vis-à-vis de l'arbre de l'état 2 et l'état 2 a été choisi contre l'état 3.

Par contre, si il est préférable de monter le nœud P_7, l'état 3 est atteint et le nœud P''_{10} possède maintenant deux places libres. Le coût de l'arbre de l'état 3 est égal à :

$$E_3 = P_{11} \cdot C(4) + P''_{10} \cdot C(2) + P_9 \cdot C(4) \tag{V-15}$$

Pour qu'un nœud *Px* fils de P_9 puisse prendre place sur le nœud P''_{10}, le coût doit être réduit. Si *Px* est déplacé vers P''_{10}, le nouveau coût total est exprimé par :

$$\begin{aligned} E_{P_9 \to P''_{10}} &= P_{11} \cdot C(4) + (P''_{10} + P_x) \cdot C(3) + (P_9 - P_x) \cdot C(3) \\ &= P_{11} \cdot C(4) + (P''_{10} + P_9) \cdot C(3) \end{aligned} \tag{V-16}$$

et le nouveau coût ne dépend pas de la probabilité du fils de P_9 qui est déplacé. Le coût total obtenu par l'équation V-16 doit être inférieur au coût de l'état 3 donné par l'équation V-15. Ainsi la condition pour déplacer un fils de P_9 peut être calculée par :

$$\begin{aligned} E_3 - E_{P_9 \to P''_{10}} &> 0 \\ P_{11} \cdot C(4) + P''_{10} \cdot C(2) + P_9 \cdot C(4) - P_{11} \cdot C(4) - (P''_{10} + P_9) \cdot C(3) &> 0 \\ P_9 - P''_{10} &> 0 \\ P_9 &> P''_{10} \end{aligned} \tag{V-17}$$

Donc, si la probabilité du nœud P''_{10} est inférieure au nœud P_9 n'importe quelle fils de *P9* peut être déplacé vers P''_{10} et en déplaçant P_4, l'état 4 est atteint. Dans le graphe de la Figure V.20 les nœuds avec double cercles sont les nœuds qui ont été modifiés pendant le processus de recherche de l'arbre optimal. Il est clair que si l'arbre est plus complexe, l'algorithme recommence pour les nœuds modifiés avec ces nœuds comme racine.

La complexité de l'algorithme est proportionnelle au nombre de nœuds de l'arbre vu que seulement trois possibilités d'arrangement sont possibles pour chaque nœud interne. En plus, si l'arbre contient seulement une place libre, il existe une seule combinaison pour chaque nœud interne susceptible de réduire le coût total. Il est important souligner que, intuitivement, les places libres doivent rester dans les nœuds plus probables, c'est-à-dire, qu'il est souhaitable d'atténuer l'entrance des portes qui commutent le plus souvent.

Ainsi, la première solution de l'algorithme tend à la solution optimale car les deux places libres sont placées dans le nœud le plus probable, i.e., la racine de l'arbre. De façon

inverse, les places libres arriveront difficilement à parcourir tous les niveaux de l'arbre, car, plus les places libres avancent plus la première condition du graphe d'état devient difficile à satisfaire.

Les portes logiques OU contenant plusieurs entrées non exclusives peuvent aussi être décomposées par l'algorithme proposé ici. Par contre, comme plusieurs entrées peuvent commuter pour la même opération, les probabilités doivent être recalculées.

L'idée est de calculer la probabilité d'une entrée de commuter sans que les entrées possédant des probabilités supérieures ne commutent. Cela peut être obtenu par l'équation suivante :

$$P_i < P_j \forall 0 < i < j \leq m$$
$$P_i^* = P_i \cdot \prod_{j=i+1}^{m} (1 - P_j) \qquad \text{(V-18)}$$

ou m est le nombre d'entrées, Pi est la probabilité initiale de l'entrée i et P_i^* est la nouvelle probabilité de l'entrée i. L'ensemble X^* est utilisé pour la construction de l'arbre des portes logiques. Ainsi, les entrées les plus probables sont favorisées par rapport aux entrées moins probables. En plus, le retard est déterminé par la première entrée qui commute, ainsi, si les entrées avec le plus de chance de commuter sont plus proches de la racine, le retard moyen est aussi réduit.

Pour les portes de Müller, toutes les entrées doivent être présentes pour que la sortie de la porte commute. En conséquence, les probabilités de commutation n'altèrent pas la consommation de l'arbre vu que toutes les portes commutent pendant une opération. Cependant, le retard moyen des entrées peut être utilisé pour la construction de l'arbre. Dans ce cas, les entrées les plus lentes doivent être mises le plus proche de la racine de l'arbre pour minimiser le retard moyen de la porte logique décomposée. Donc, l'algorithme proposé ici peut être utilisé pour décomposer les portes de Müller et les portes ET et les retards, ou temps d'arriver, des signaux d'entrées sont employés pour la construction de l'arbre.

V.3.3. Résultats

L'algorithme proposé a été appliqué pour la décomposition de portes OU avec 25 et 24 entrées. Pour chaque porte OU, plusieurs ensemble de probabilités ont été testés avec des entrances maximales égales à 4 et 3. Par la suite, la consommation d'énergie moyenne de la solution proposée par l'algorithme a été comparée avec la consommation d'énergie d'un arbre équilibré. Un arbre équilibré est la solution usuelle des outils de synthèse et la solution

normalement utilisé par les concepteurs des circuits intégrés. Le Tableau V.2 présente les ensembles de probabilités qui ont testés et la Figure V.21 présente la compilation de ces résultats.

Tableau V.2 – Les ensembles de probabilités d'entrées testés avec l'algorithme.

Ensemble	nombre d'entrées	ensemble testé nombre d'entrées × probabilité
A	25	25×0,04
B	25	1×0,5 ; 1×0,04 ; 23×0,02
C	25	1×0,5 ; 2×0,14 ; 22×0,01
D	25	2×0,3 ; 1×0,114 ; 2×0,1 ; 3×0,018 ; 17×0,002
E	24	4×0,05 ; 20×0,04
F	24	1×0,5 ; 1×0,06 ; 22×0,02
G	24	1×0,5 ; 1×0,15 ; 1×0,14 ; 21×0,01
H	24	2×0,3 ; 1×0,16 ; 2×0,1 ; 3×0,018 ; 16×0,002

L'axe des abscisses correspond à l'ensemble, au nombre d'entrées de la porte et l'entrance maximale choisie et l'axe des ordonnées correspond à la consommation d'énergie de la solution proposée par l'algorithme normalisée par la consommation de la solution de l'arbre équilibré. Pour chaque valeur du nombre d'entrées et de l'entrance maximale, quatre ensembles de probabilités ont été évalués.

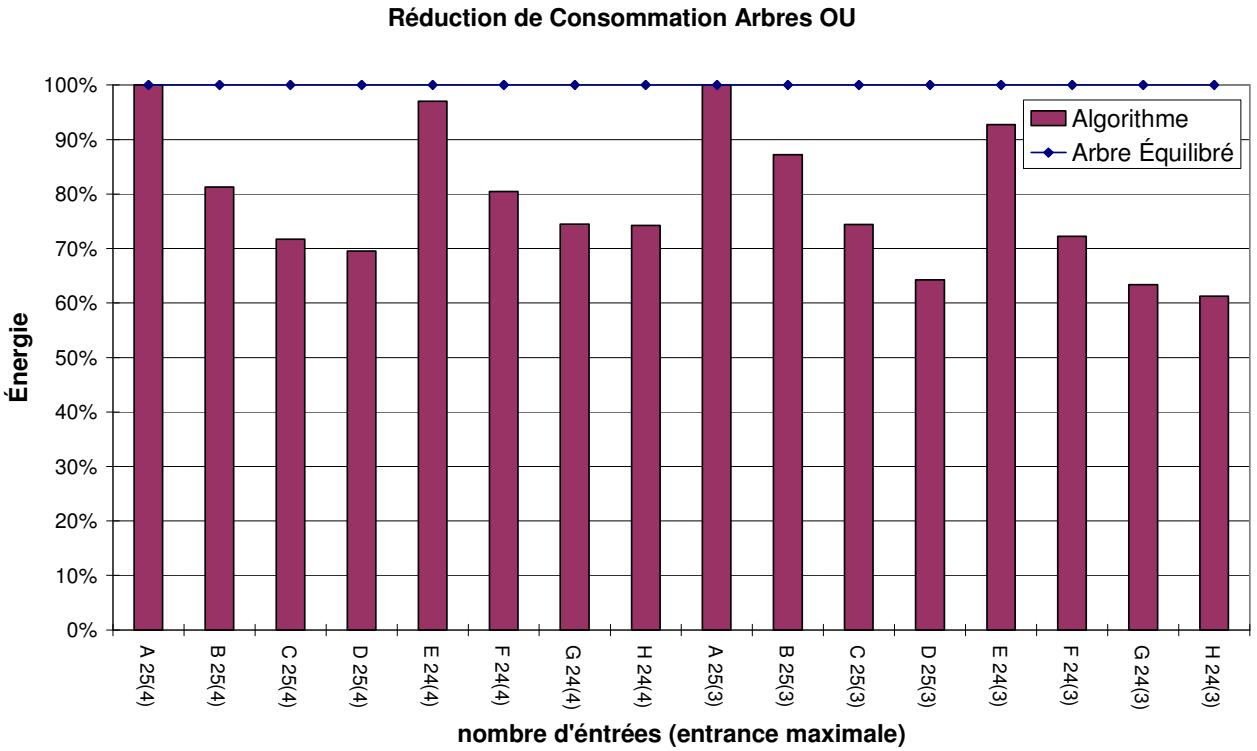


Figure V.21 – Comparaison de l'algorithme de décomposition versus un arbre équilibré.

Comme attendu, l'algorithme proposé offre toujours une réduction de consommation pouvant aller de 10% jusqu'à 40% dans les ensembles étudiés. La consommation est égale à la consommation de l'arbre équilibré seulement dans les cas où l'arbre équilibré est la solution, c'est-à-dire, lorsque toutes les entrées possèdent la même probabilité de commuter.

La consommation d'énergie de la solution initiale donnée par l'algorithme de Huffman a aussi été comparée contre l'algorithme optimisé proposé ici. La Figure V.22 montre les résultats de l'algorithme optimisé contre l'algorithme de Huffman. Dans la Figure V.22 seulement les ensembles qui contiennent des places libres sont présentés, car seulement pour ces exemples l'optimisation est utile.

Comme il a été discuté, la réduction face à l'algorithme de Huffman est moins significative, puisque la solution de l'algorithme de Huffman est « presque » optimale. Cependant, pour certains cas, l'optimisation ici proposée permet une amélioration des résultats et, de plus, la solution optimale est rapidement obtenue à partir de l'arbre initial construit avec l'algorithme de Huffman.

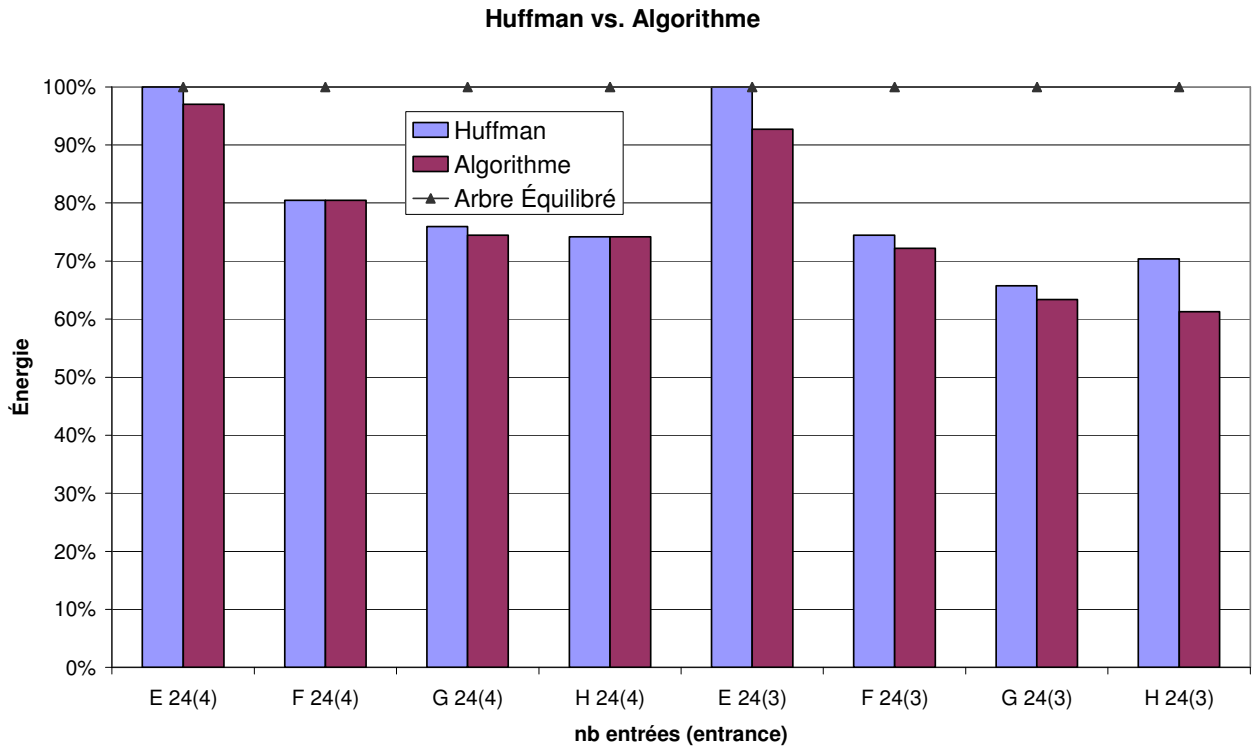


Figure V.22 – Comparaison de l'algorithme versus la solution initiale de Huffman.

En conséquence, l'algorithme proposé peut être facilement employé dans les outils de conception asynchrones pour décomposer aussi bien les portes du chemin de données que les portes du chemin des acquittements et, ainsi, obtenir des circuits moins gourmant en énergie et plus performants en temps.

V.4. Références

[V.1] Jens Sparsø and Steve Furber, editors. Principles of Asynchronous Circuit Design: A Systems Perspective. Kluwer Academic Publishers, 2001.

[V.2] Charles L. Seitz. System timing. In Carver A. Mead and Lynn A. Conway, editors, Introduction to VLSI Systems, chapter 7. Addison-Wesley, 1980.

[V.3] L. G. Heller, et al. Cascode voltage switch logic : a differencial CMOS logic family. In IEEE International Solid State Circuits Conference, 1984.

[V.4] A. Martin. Programming in VLSI: from communicating processes to delay-insensitive circuits. In C.A.R. Hoare, editor, Developments in Concurrency and Communication. Addison-Wesley, 1990.

[V.5] A. Martin. Asynchronous datapaths and the design of an asynchronous adder. Formal Methods in System Design, 1(1):117-137, July, 1992.

[V.6] T. Meng. Synchronization Design for Digital Systems. Kluwer Academic Publisher, 1991.

[V.7] C. Nielsen. Evaluating of functional blocks designs. Technical Report ID-TR:1994-135, Dept. of Computer Science, Technical University of Denmark, 1994.

[V.8] E. Sentovich, et al. SIS: a system for sequential circuits synthesis. Technical Report UCB/ERL M92/41, UC Berkeley, 1992.

[V.9] Jens Sparsø and Jørgen Staunstrup. Delay-insensitive multi-ring structures. Integration, the VLSI journal, 15(3):313-340, October 1993.

[V.10] David E. Muller and W. S. Bartky. A theory of asynchronous circuits. In Proceedings of an International Symposium on the Theory of Switching, pages 204-243. Harvard University Press, April 1959.

[V.11] David E. Muller. Asynchronous logics and application to information processing. In Symposium on the Application of Switching Theory to Space Technology, pages 289-297. Stanford University Press, 1962.

[V.12] Karl M. Fant and Scott A. Brandt. NULL conventional logic: A complete and consistent logic for asynchronous digital circuit synthesis. In International Conference on Application-specific Systems, Architectures, and Processors, pages 261-273, 1996.

[V.13] Ilana David, Ran Ginosar, and Michael Yoeli. An efficient implementation of boolean functions as self-timed circuits. IEEE Transactions on Computers, 41(1):2-11, January 1992.

[V.14] Michiel Ligthart, Karl Fant, Ross Smith, Alexander Taubin, and Alex Kondratyev. Asynchronous design using commercial HDL synthesis tools. In Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems, pages 114-125. IEEE Computer Society Press, April 2000.

[V.15] Alex Kondratyev and Kelvin Lwin. Design of asynchronous circuits by synchronous CAD tools. In Proc. ACM/IEEE Design Automation Conference, June 2002.

[V.16] Alex Kondratyev and Kelvin Lwin. Design of asynchronous circuits using synchronous CAD tools. IEEE Design & Test of Computers, 19(4):107-117, 2002.

[V.17] Alex Branover, Rakefet Kol, and Ran Ginosar. Asynchronous design by conversion: Converting synchronous circuits into asynchronous ones. In Proc. Design, Automation and Test in Europe (DATE), pages 870-875, February 2004.

[V.18] P. Maurine, J.B. Rigaud, F. Bouesse, G. Sicard, M. Renaudin, "Static Implementation of QDI asynchronous primitives", PATMOS'03, Torino, Italy, September 2003, pp.181-191.

[V.19] P. Maurine, J.B. Rigaud, F. Bouesse, G. Sicard, M. Renaudin, "TAL: une bibliothèque de cellules pour le design de circuits asynchrones QDI", FTFC'03, Paris, France, May 2003.

[V.20] D. Huffman. A method for the construction of minimum redundancy codes. In Proc IRE, 40(9):1098-101, 1952.

Chapitre VI

Additionneurs

L'addition est l'opération la plus courante en chemins de données numériques. En plus, l'addition est utilisée comme bloc de base à la réalisation de toutes les autres opérations arithmétiques. Dans les systèmes numériques actuels, l'addition est employée extensivement soit explicitement dans les calculs des chemins de données soit implicitement dans les parties de contrôle d'exécution et d'indexation de données.

De ce fait, une étude des chemins de données asynchrones doit impérativement aborder la réalisation d'une telle opération. Ce chapitre présente une méthodologie de généralisation et génération automatique d'additionneurs « *N-Rails* » où additionneurs avec les entrées codées en *1-parmi-N*.

VI.1. Représentation de Nombres Entiers

Avant de continuer, il est important de formaliser la représentation de nombres entiers. Évidemment, il existe d'innombrables façons de coder un nombre entier, cependant cette thèse s'intéresse particulièrement aux codages QDI *1-parmi-N* qui sont les codages supportés par l'outil de conception TAST.

Un digit codé en *1-parmi-N* peut représenter N valeurs distinctes entre *0* et N–1. Comme il a été discuté précédemment, N fils sont nécessaires pour implémenter le codage QDI permettant la détection de l'état invalide. Ainsi, d digits *1-parmi-N* peuvent représenter N^d valeurs distinctes. Inversement, une valeur entière v requiert d digits dans la base N, où

$$d = \lceil \log_N (v) \rceil \qquad \text{(VI-1)}$$

et $d \times N$ fils sont requis. En conséquence, le codage choisi défini la base dans laquelle les valeurs seront codées. Le Tableau VI.1 montre l'ensemble des valeurs qui peuvent être codées en utilisant le codage double rail (base binaire) avec trois digits et le codage *1-parmi-3* (base 3) avec deux digits. Dans le tableau, les colonnes « valeurs » expriment les valeurs entières dans chaque base et les colonnes « codage » présentent les valeurs pour chaque fil utilisé dans le codage avec leurs digits séparées par des points. Si les fils de chaque digit sont indexés de droite à gauche, il est facile de comprendre que la valeur de chaque digit identifie le fil actif pendant la transmission d'une donnée.

Tableau VI.1 – Ensemble des valeurs entières codées en binaire et *1-parmi-3*.

valeur/état	binaire		base 3	
	valeur	codage double rail	Valeur	codage *1-parmi-3*
Invalide	---	00.00.00	--	000.000
0	000	01.01.01	00	001.001
1	001	01.01.10	01	001.010
2	010	01.10.01	02	001.100
3	011	01.10.10	10	010.001
4	100	10.01.01	11	010.010
5	101	10.01.10	12	010.100
6	110	10.10.01	20	100.001
7	111	10.10.10	21	100.010
8			22	100.100

Comme pour les circuits synchrones, les valeurs entières négatives peuvent aussi être codées. Pour ce faire, les valeurs négatives sont codées en utilisant le complément à la base, comme pour la base binaire. Ainsi, les additionneurs peuvent traiter des valeurs signées et non signées sans aucune distinction.

Cependant, les valeurs signées avec *d* digits sont dépendantes de la base et l'ensemble de valeurs négatives codées est différent dans les bases pairs et impaires. Tandis que, une base paire *N* avec *d* digits permet de coder des valeurs entre $-N^d/2$ et $\left(N^d/2\right)-1$, une base impaire *N* avec *d* digits permet de coder des valeurs entre $-\left(N^d-1\right)/2$ et $\left(N^d-1\right)/2$. La Figure VI.1 montre une représentation des ensembles de valeurs qui peuvent être codées avec une base *N* et *d* digits.

Comme exemple, le Tableau VI.2 présente l'ensemble des valeurs signées codées avec trois digits dans la base binaire et deux digits dans la base *1-pami-3*. À partir d'ici, la référence à la base binaire signifie un codage double rail.

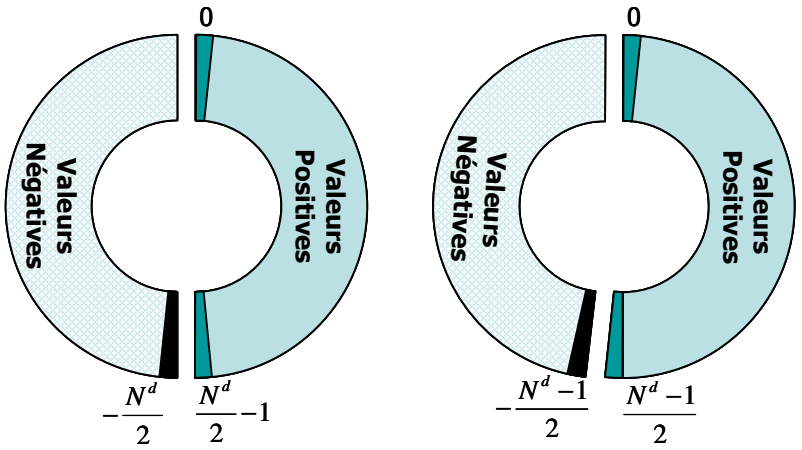


Figure VI.1 – Ensemble de valeurs codées avec une base *N* et *d* digits.

À ce point, plusieurs facteurs deviennent apparents. Deux bases ne couvrent forcement pas le même ensemble de valeurs, donc, la conception des interfaces de conversions entre deux bases peut ne pas être trivial et ces interfaces peuvent être coûteuses en termes de complexité. Cet caractéristique souligne donc l'intérêt de concevoir les opérateurs d'un chemin de donnée directement dans la base dans laquelle les données sont représentées en évitant ainsi quelque interface et complexité additionnelle. Clairement aussi, des interfaces additionnelles entraîneront des possibles augmentation de consommation et retard dans les chemins de données.

Tableau VI.2 – Ensemble des valeurs entières signées codées en binaire et *1-parmi-3*.

Valeur/état	binaire		base 3	
	valeur	codage double rail	valeur	codage *1-parmi-3*
invalide	---	00.00.00	--	000.000
0	000	01.01.01	00	001.001
1	**001**	**01.01.10**	**01**	**001.010**
2	010	01.10.01	02	001.100
3	011	01.10.10	10	010.001
4			11	010.010
-4	100	10.01.01	12	010.100
-3	101	10.01.10	20	100.001
-2	110	10.10.01	21	100.010
-1	**111**	**10.10.10**	**22**	**100.100**

En plus, dans les additionneurs binaires synchrones, un soustracteur est obtenu par l'addition du premier opérande plus le deuxième opérande inversé plus 1. En fait, l'inversion d'un opérande en base binaire permet d'obtenir le complément à 1, ou, de façon équivalente, le complément à *base*-1. En conséquence, l'addition du deuxième opérande avec 1 permet alors

d'obtenir le complément à 2, ou le complément à la base. De façon équivalente, la même opération peut être réalisée pour n'importe quelle base, c'est-à-dire, qu'un soustracteur peut être obtenu par l'addition du premier opérande plus le deuxième opérande complémenté à *base*-1 plus 1. A partir d'ici, la notation $\overline{A}$ sera employée pour exprimer le complément à la *base-1* de la valeur A. Ainsi :

$$\begin{aligned} -A &= \overline{A} + 1 \\ A + \overline{A} &= N^d - 1 \end{aligned} \qquad \text{(VI-2)}$$

où N est la base et d est le nombre de digits utilisés pour le codage de A. Il est important de mettre en évidence que la valeur $\overline{A}$ est facilement obtenue dans les codages *1-parmi-N* avec le simple échange des fils de chaque digit, tandis que dans les circuits synchrones des inverseurs sont requis pour chaque digit pour effectuer le complément à 1. Dans le Tableau VI.2, la valeur *-2*, qui équivaut au complément à *base*-1 de la valeur 1 (voir les lignes grises du tableau), peut être obtenue par la permutation adéquate des fils de la valeur 1 dans les deux bases. Finalement, en additionnant 1 à la valeur -2, la valeur -1 est atteinte.

Donc, un soustracteur asynchrone est conçu par l'addition du premier opérande plus le deuxième opérande avec les fils de chaque digit permutés plus *1*. De ce fait, aucun matériel supplémentaire n'est ajouté à la réalisation d'un soustracteur et, donc, la discussion dans ce chapitre est focalisée exclusivement sur les additionneurs.

VI.2. Additionneurs N-Rails Généralisés

L'addition binaire est une opération bien connue des concepteurs des circuits numériques :

$$\begin{aligned} S &= (A + B + Ci) \bmod(2) \\ Co &= (A + B + Ci) / 2 \end{aligned} \qquad \text{(VI-3)}$$

où A, B, Ci, S et Co sont des digits binaires, Ci et Co sont, respectivement, la retenue entrante et la retenue sortante et la division est entière.

De façon évidente, les équations VI-3 peuvent être généralisées en remplaçant la valeur 2 par N, où N est la base choisie pour coder les digits. Une propriété importante de l'addition est qu'indépendamment de la base dans laquelle les valeurs A et B sont exprimées, la retenue sortante de l'addition de A et B est inférieure ou égale à 1. En conséquence, si la retenue entrante est-elle aussi inférieure ou égale à 1, la retenue sortante de l'addition de *A, B* et *Ci* restera aussi inférieure à 1. Les équationsVI-6 suivantes sont la démonstration de cette

propriété. Par conséquence, les valeurs des retenues peuvent toujours être codées en double rail (base binaire) en dépit de la base choisie pour le codage des valeurs de *A* et *B*.

$$\begin{aligned} & A \le B \le N-1, Ci \le 1 \\ & (A+B+Ci) \le 2N-1 \\ & \Rightarrow Co = (A+B+Ci)/N \le (2N-1)/N \\ & \Rightarrow Co \le 1 \end{aligned} \qquad \text{(VI-4)}$$

Donc, un additionneurs complet généralisé (FA – de l'anglais « Full Adder ») est une cellule recevant deux digits (*A* et *B*) codées en *1-parmi-N* et une retenue (*Ci*) codée en double rail et produisant à sa sortie un digit (*S*) *1-parmi-N* et une retenue (Co) codée en double rail. La Figure VI.2 présente la cellule de l'additionneur complet généralisé.

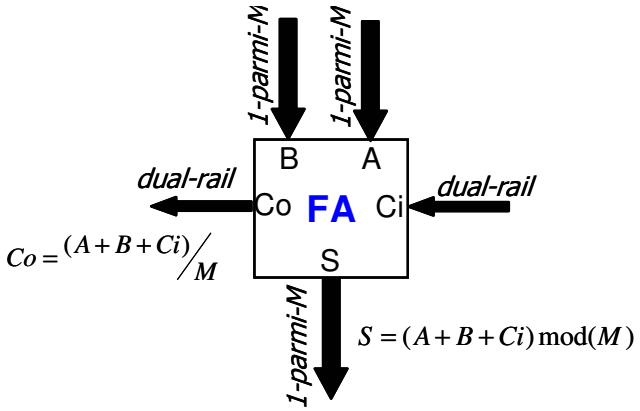


Figure VI.2 – Cellule d'additionneur complet généralisé.

Une simple implémentation par somme de produits (DIMS [VI.1]) peut maintenant être réalisée. La Figure VI.3 montre cette implémentation pour une cellule d'additionneur complet *1-parmi*-3. Comme notation, les lettres majuscules sont utilisées pour identifier les digits et les indexes pour identifier chaque fils d'un même digit. Si la variable comporte plusieurs digits, des exposants sont employés pour distinguer chaque digit d'une même variable. Si l'intérêt porte seulement sur les digits et l'information sur chaque fil peut être supprimé, les lettres minuscules avec indexes seront utilisées pour simplifier l'écriture.

Dans le circuit de la Figure VI.3, chaque porte de Müller (C-Element) implémente un minterm de la fonction. L'entrée A_0 correspond au fil d'index zéro du digit *A* et, donc, quand A_0 est active, la valeur de *A* est égale à zéro et tous les autres fils de *A* sont inactifs. Comme les entrées *A* et *B* sont codées en base *N* et *Ci* est codée en double rail, il est facile de déterminer que le nombre de minterms nécessaires à une implémentation par somme de produits est égal à :

$$\#mt = 2 \times N^2 \qquad \text{(VI-5)}$$

En plus, chaque Müller est exclusive, c'est-à-dire, qu'une seule Müller commute à la fois.

Ainsi, chaque fil de la variable *MT* est exclusif et les minterms recodent les variables d'entrée dans une base *1-parmi-T*, où *T* est le nombre de minterms.

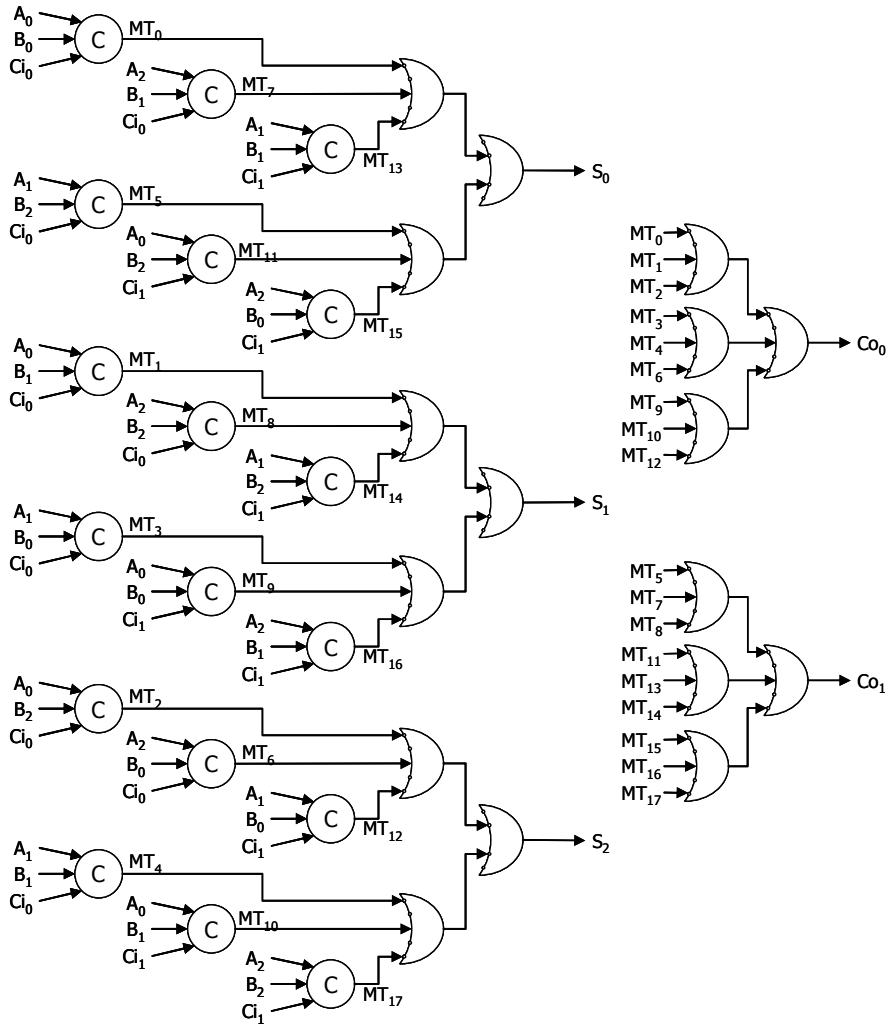


Figure VI.3 – Réalisation DIMS d'un additionneur complet *1-parmi-3*.

Pour permettre la génération automatique de la cellule « FA », les résultats de la somme et de la retenue sortante peuvent être exprimés par deux matrices carrées de calcul : une matrice pour *Ci* égal à zéro et une matrice pour *Ci* égal à 1. Dans chaque matrice, les lignes et les colonnes sont les valeurs des digits d'entrée et chaque cellule de la matrice contient les résultats pour *Co* et *S*. En étant donnée que *A* et *B* peuvent assumer *N* valeurs distinctes, chaque matrice possède donc N^2 cellules. Le Tableau VI.3 montre la matrice de calcul pour un additionneur *1-parmi-3* avec la retenue entrante égale à zéro (*Ci*=0).

Après le calcul de tous les minterms de *A* et *B*, il est possible de constater qu'une diagonale inverse de la matrice produit toujours le même résultat pour la somme. Ainsi, les

minterms appartenant à une diagonale inverse de la matrice sont utilisés pour obtenir la fonction logique pour le fil de *S* correspondant. C'est-à-dire, que la fonction logique pour chaque fil de la somme est égale à la somme logique des minterms d'une même diagonale inverse.

Tableau VI.3 – Matrice de calcul pour l'additionneur *1-parmi-3* avec retenue entrante égale à zéro (*Ci*=0).

Co/S		A		
		0	1	2
B	0	**0/0**	**0/1**	**0/2**
	1	**0/1**	**0/2**	**1/0**
	2	**0/2**	**1/0**	**1/1**

Quand la retenue entrante est égale à 1, la règle pour l'obtention des fonctions logiques est la même pour la deuxième matrice de calcul, cependant, les diagonaux inverses sont décalées à gauche, comme le montre le Tableau VI.4 pour l'additionneur *1-parmi-3*. En conséquence, la valeur de la retenue entrante peut être vue comme un contrôle pour déterminer si le résultat de la somme de *A* et *B* doit, ou non, être décalé à gauche.

Tableau VI.4 – Matrice de calcul pour l'additionneur *1-parmi-3* avec retenue entrante égale à un (*Ci*=1).

Co/S		A		
		0	1	2
B	0	**0/1**	**0/2**	**1/0**
	1	**0/2**	**1/0**	**1/1**
	2	**1/0**	**1/1**	**1/2**

Il est donc possible de séparer l'addition en deux étapes : (1) la réalisation de l'addition de *A* et *B* dans la base *N* et (2) le multiplexage du résultat avec la retenue entrante comme contrôle du multiplexeur. Le fait de séparer le calcul en deux étapes réduit le nombre de minterms nécessaires à :

$$\#mt = N^2 + 2 \times N \leq 2 \times N^2 \qquad \text{(VI-6)}$$

où N^2 est le nombre de minterms pour l'addition de *A* et *B* et $2 \times N$ est le nombre de minterms du multiplexage du résultat de *A* et *B* avec la retenue entrante.

En plus, pour les cellules au dessus de la diagonale secondaire la valeur de la retenue sortante dans les deux matrices de calcul est égale à zéro et la retenue sortante est donc indépendante de la valeur de la retenue entrante. Indépendamment aussi de la valeur de la retenue entrante, la retenue sortante est toujours égale à 1 pour toutes les cellules au dessous de la diagonale secondaire. Finalement, la valeur de la retenue entrante est copiée sur la retenue

sortante si le résultat provient d'une des cellules appartenant à la diagonale secondaire. Ainsi, les cellules au dessus de la diagonale secondaire « absorbent » (de l'anglais « Kill ») la valeur de la retenue sortante – c'est-à-dire, Co=0 – les cellules au dessous de la diagonale secondaire « Génèrent » une retenue sortante (Co=1) et les cellules appartenant à la diagonale secondaire « Propagent » la valeur de la retenue d'entrée à la retenue sortante.

Basée sur cette analyse, la matrice de calcul du Tableau VI.3 peut être réécrit en vue de traduire ce comportement, comme montre le Tableau VI.5. Cette nouvelle matrice de calcul sera appelée matrice *KPG*.

Tableau VI.5 – Matrice de calcul de KPG pour l'additionneur *1-parmi-3*.

KPG		A		
		0	1	2
B	0	**K_0**	**K_1**	**P**
	1	**K_1**	**P**	**G_0**
	2	**P**	**G_0**	**G_1**

Dans la matrice du Tableau VI.5 chaque cellule contient *K* pour les cellules « absorbant » la retenue, *P* pour les cellules « propageant » la valeur de la retenue entrante à la retenue sortante et *G* pour les cellules « générant » une retenue de sortie indépendamment de la valeur de la retenue d'entrée. Les cellules *K* et *G* comportent aussi un index pour déterminer à quelle diagonale inverse elles appartiennent. Il est important de capturer cette information vu que les cellules appartenant à une même diagonale inverse sont nécessaires au calcul du résultat de la somme. En conséquence, cette formulation permet d'utiliser la même sous-fonction logique à la fois au calcul de la retenue sortante et au calcul de la sortie *S*.

La construction de la matrice *KPG* peut être généralisée par les équations logiques suivantes :

$$\begin{aligned} K_j &= \sum_{i=0}^{j} \left(A_i \cdot B_{j-i}\right), \forall j < N-1 \\ P &= \sum_{i=0}^{N-1} \left(A_i \cdot B_{N-i-1}\right) \\ G_j &= \sum_{i=j+1}^{N-1} \left(A_i \cdot B_{N+j-i}\right), \forall j < N-1 \end{aligned} \qquad \text{(VI-7)}$$

et à partir des équations les diagonales inverses qui sont le résultat de l'addition de *A* et *B* sont données par :

$$\begin{aligned} T_j &= K_j + G_j, \forall j < N-1 \\ T_{N-1} &= P \end{aligned} \qquad \text{(VI-8)}$$

où *M* est la base dans laquelle les digits *A* et *B* sont codés.

En possession de la matrice *KPG*, il est maintenant facile de combiner ces résultats avec la retenue entrante en vue d'obtenir le résultat final de la somme et de la retenue sortante. Les fonctions logiques pour chaque fil de la sortie *S* sont exprimées par :

$$S_j = Ci_0 \cdot T_j + Ci_1 \cdot T_{(j-1) \bmod (N)}, \forall j = \{0,...,N-1\} \qquad \text{(VI-9)}$$

et les fonctions de la retenue sortante sont :

$$\begin{aligned} Co_1 &= G + P \cdot Ci_1, G = \sum_{i=0}^{M-2} G_i \\ Co_0 &= K + P \cdot Ci_0, K = \sum_{i=0}^{M-2} K_i \end{aligned} \qquad \text{(VI-10)}$$

Ainsi, cet ensemble d'équations permet de concevoir automatiquement un additionneur complet pour n'importe quelle codage *1-parmi-N* des digits d'entrée. La Figure VI.4 présente le résultat de la génération d'un additionneur complet *1-parmi-3* avec cette méthodologie.

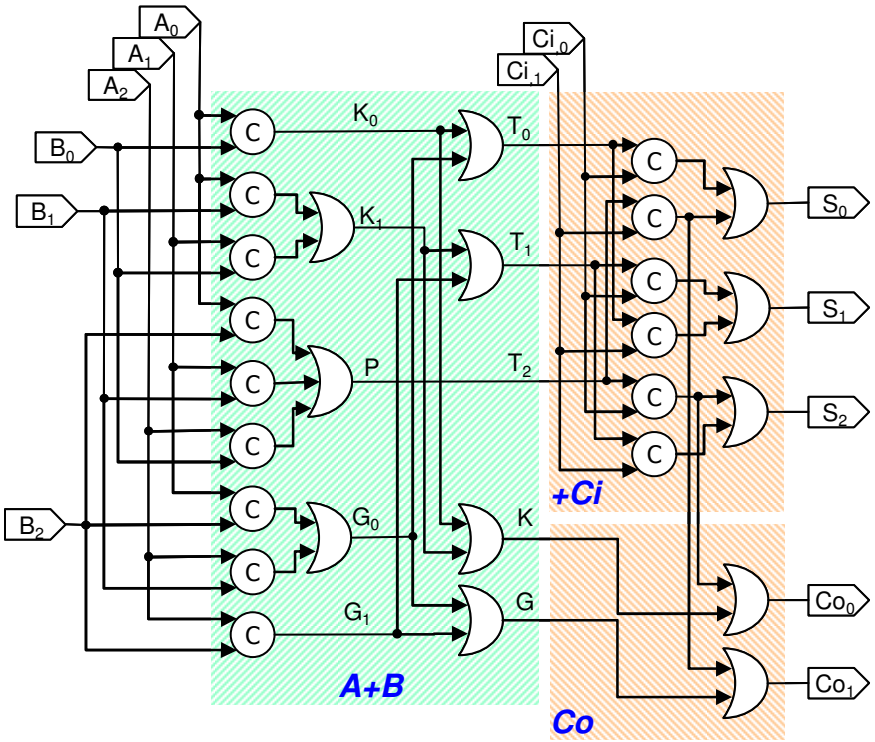


Figure VI.4 – Résultat de génération d'un additionneur complet *1-parmi-3*.

Dans le circuit de la Figure VI.4, il est possible d'identifier les deux étapes de calcul : (1) la première, dénommée « $A+B$ » dans la figure, responsable du calcul de la matrice *KPG* et les diagonales inverses de la matrice et (2) la deuxième étape, dénommée « $+Ci$ » et « Co », responsable de combiner le résultat précédent avec la retenue entrante. Il est aussi possible de vérifier que, par exemple, la sous-fonction logique de calcul de « K_1 » est utilisée pour le calcul de « T_1 » et pour le calcul de « K ».

Évidemment, un demi additionneur sans retenue d'entrée (HA, de l'anglais « Half Adder ») peut être obtenu directement à partir des équations VI-7. La somme et la retenue sortante pour un demi additionneur sont définies par :

$$\begin{aligned} S_j &= K_j + G_j, \forall j < N-1 \\ S_{N-1} &= P \\ Co_1 &= \sum_{i=0}^{N-2} G_i \\ Co_0 &= \left(\sum_{i=0}^{N-2} K_i \right) + P \end{aligned} \qquad \text{(VI-11)}$$

VI.2.1. Additionneurs à Retenue Propagée

Avec les cellules généralisées d'un additionneur complet et la cellule d'un demi additionneur, il est maintenant possible de concevoir l'implémentation d'un additionneur à plusieurs digits.

L'additionneur à retenue propagée est la plus simple structure d'addition. Cet additionneur consiste à ranger proprement une série de cellules d'additionneur d'un digit. La Figure VI.5 présente un additionneur généralisé à retenue propagée avec huit digits.

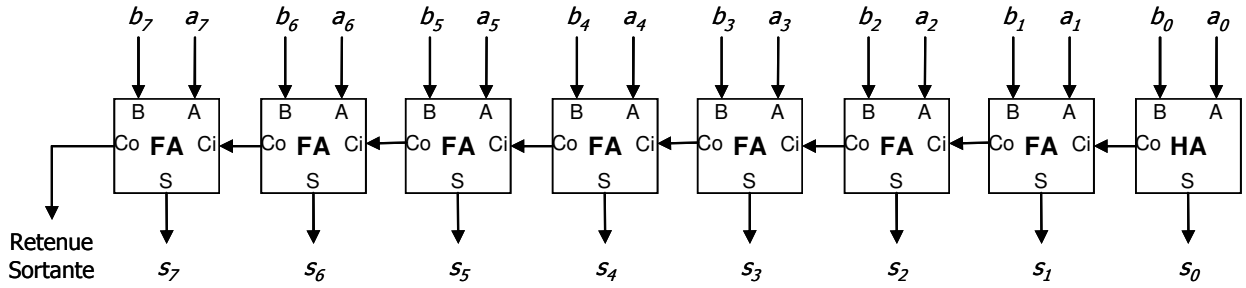


Figure VI.5 – Additionneur généralisé à retenue propagée avec huit digits.

Malheureusement, cet additionneur facile à concevoir présente un retard critique qui est une fonction linéaire du nombre de digits vu que le chemin de calcul de la retenue sortante

traverse toutes les cellules de l'additionneur. La littérature propose d'autres solutions pour réduire le chemin critique des additionneurs.

VI.2.2. Additionneurs à Préfixe Parallèle

L'addition de deux digits *A* et *B* avec la retenue entrante, si cette dernière est connue, peut s'effectuer à temps constant par la cellule généralisée d'un additionneur complet présentée dans les sections précédentes. Donc, le principal problème dans la conception d'additionneurs rapides est le calcul des retenues entrantes à chaque niveau. L'objectif doit être de briser la lente chaîne de calcul de la retenue, nécessaire au calcul de chaque digit de la somme.

Du point de vue du calcul de la propagation de la retenue et de la conception d'un réseau de retenues, les valeurs des opérandes à une position donnée *i* ne sont pas importantes. L'important pour le calcul de la retenue est de savoir si, pour une position donnée, une retenue est générée, propagée ou absorbée.

Dans la plupart des additionneurs à anticipation de la retenue (de l'anglais « carry-lookahead adders ») synchrones, la technique consiste à coder l'information de génération, propagation ou absorption de la retenue par deux bits – vu qu'il existe trois valeurs possibles – dans un préfixe et de combiner ensuite ces préfixes de façon à obtenir la retenue de chaque digit. Ici, seulement les structures de calcul de préfixes en parallèle par des arbres seront étudiées.

Les structures consistant à regrouper directement jusqu'à quatre préfixes ne sont pas exploitées ici. Ces structures dans les circuits synchrones ne peuvent regrouper plus que quatre préfixes pour raisons électriques qui limitent le nombre maximum d'entrées d'une porte logique. Inévitablement, les circuits asynchrones souffrent des mêmes limitations électriques et, en plus, il faudrait synchroniser les préfixes à regrouper dans un circuit asynchrone conduisant à une sensible complexité. Les solutions existantes, pour les additionneurs synchrones avec plusieurs digits, consistent soit à augmenter la base dans laquelle l'addition est réalisée soit à réaliser l'addition avec plusieurs niveaux d'additionneurs à complexité limitée de calcul de la retenue.

Pour les circuits asynchrones, la première solution d'augmenter la base consisterait à modifier le codage des valeurs d'entrée. La deuxième solution qui consiste à regrouper plusieurs additionneurs implique d'innombrables combinaisons possibles en fonction de la base et le nombre de digits de chaque additionneur. Comme le but recherché est la généralisation pour différentes bases des additionneurs y compris les additionneurs rapides, ces architectures ne sont pas abordées au cours de cette thèse. Cependant, les structures présentes

dans les additionneurs à préfixe parallèle peuvent être combinées en vue d'obtenir des architectures d'additionneurs hiérarchiques.

Pour effectuer la généralisation des additionneurs rapides, la première étape consiste à séparer le calcul de la somme du calcul des retenues. La Figure VI.6 montre la structure générale des additionneurs rapides à préfixe parallèle. La structure de la Figure VI.6 comporte trois étages :

- ➔ un étage d'entrée responsable du calcul de la somme de chaque digit a_i et b_i ;
- ➔ un étage de calcul des préfixes ou arbre de calcul des retenues ;
- ➔ et un étage de sortie pour additionneur le résultat de la somme provenant du premier étage avec la retenue correspondante.

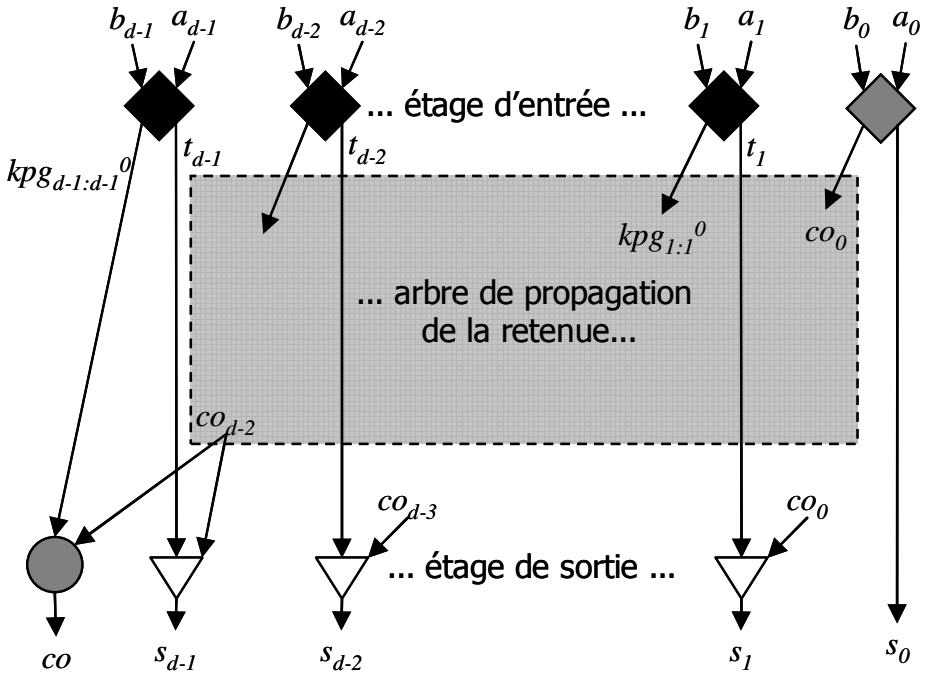


Figure VI.6 – Structure générale de base d'additionneurs à préfixe parallèle.

VI.2.2.1.Le Préfixe « *kpg* »

Avant de continuer, le préfixe « *kpg* » doit être défini. De façon générale, pour une addition en base *N* dans une position *i*, il existe trois possibilités :

7. générer une retenue, si, et seulement si, $a_i + b_i \geq N$;
8. propager la retenue de la position précédente, si, et seulement si, $a_i + b_i = N - 1$;
9. absorber, ou tuer la propagation de la retenue, si, et seulement si, $a_i + b_i < N - 1$.

Comme il a déjà été dit, dans les additionneurs synchrones, ces trois possibilités sont codées avec deux bits et donc un simple codage asynchrone double rail avec deux digits nécessitant 4 fils est suffisant pour encoder cette information. Cependant, avec une observation plus minutieuse, il est possible de constater que ces trois possibilités sont exclusives et que cette information peut être codée avec un codage asynchrone *1-parmi-3* et en utilisant seulement 3 fils. En plus, ces trois situations correspondent aux trois régions *K*, *P* et *G* de la matrice de calcul du Tableau VI.5. Ainsi le préfixe « *kpg* » peut être facilement obtenu par les équations généralisées présentées précédemment.

En conséquence, le préfixe *kpg* est défini par :

$$\begin{aligned} kpg &= (K, P, G) \\ kpg &= 0 = (1,0,0) \forall a + b < N - 1 \\ kpg &= 1 = (0,1,0) \forall a + b = N - 1 \\ kpg &= 2 = (0,0,1) \forall a + b > N - 1 \end{aligned} \qquad \text{(VI-12)}$$

et il est donc maintenant possible de généraliser la conception de chacun des étages des additionneurs rapides.

VI.2.2.2.L'Étage d'Entrée

L'étage d'entrée consiste en un opérateur qui doit calculer le résultat de la somme de deux digits *a* et *b* ainsi que le préfixe *kpg* pour une position *i* donnée. Le résultat de la somme est calculé dans la même base *N* des digits *a* et *b* et *kpg* est codée en *1-parmi-3*. Cette opération est représentée par les losanges noirs dans la Figure VI.6.

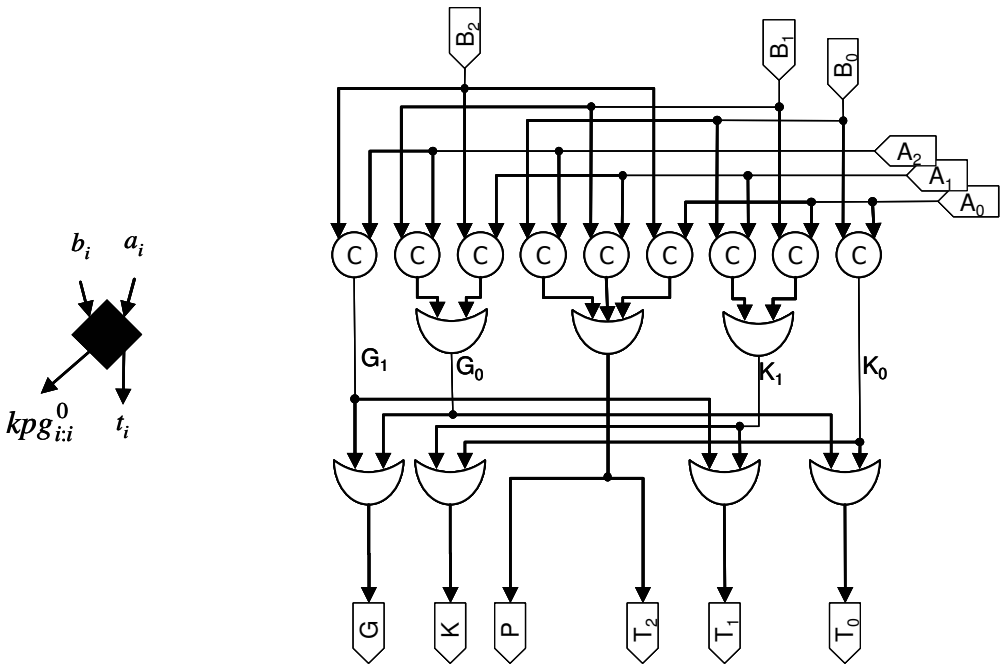


Figure VI.7 – L'opérateur diamant noir et son implémentation *1-parmi-3*.

Il est facile d'apercevoir que cette opération est la même que le premier étage de l'additionneur complet généralisé précédemment – région nommée « $A+B$ » dans la Figure VI.4. Donc les diamants noirs peuvent être généralisés à partir des équations VI-7 et VI-8.

La Figure VI.7 présent l'opérateur diamant noir de l'étage d'entrée et son implémentation pour le codage *1-parmi-3* des digits d'entrées.

Évidemment, si l'additionneur ne reçoit pas de retenue d'entrée, l'opérateur diamant du digit moins significatif peut être simplifié vu qu'il n'y a pas de retenue à propager. Il faut donc juste savoir si une retenue est générée ou pas au premier digit, ou, de façon équivalente, savoir si la retenue du digit le moins significatif est égale à zéro ou égale 1. Ainsi, si l'additionneur est conçu sans une retenue entrante, l'opérateur diamant du digit le moins significatif est équivalent à un demi additionneur. Cet opérateur est indiqué par un diamant gris dans le schéma de la Figure VI.6.

VI.2.2.3.L'Étage de Sortie

L'étage de sortie a pour rôle d'additionner le résultat de la somme produit par l'étage d'entrée avec la retenue de la position précédente provenant de l'arbre de calcul des retenues. Cette opération est représentée par les triangles dans le graphe de la Figure VI.6.

À nouveau ici, sans grand effort, il est facile de se rendre compte que l'opération de l'étage de sortie correspond à la deuxième étape de l'additionneur complet généralisé – région nommée « $+Ci$ » dans la Figure VI.4. Donc, les triangles peuvent être conçus de façon générale à partir de l'équation VI-9. La Figure VI.8 montre l'opérateur de l'étage de sortie et son implémentation pour un codage *1-parmi-3* des digits d'entrée.

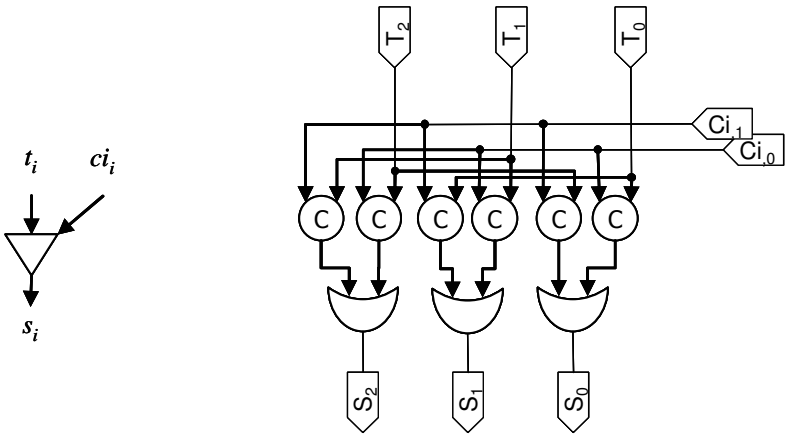


Figure VI.8 – L'opérateur triangle et son implémentation *1-parmi-3*.

VI.2.2.4.L'Arbre de Calcul de la Retenue

L'arbre de calcul de la retenue consiste à obtenir la retenue d'une position *i* en regroupant les préfixes *kpg* des positions précédentes. Les préfixes *kpg* sont combinés deux à deux et la seule règle de construction est que la retenue d'un rang *i* soit reliée à tous les préfixes de rang inférieur à *i* par un arbre binaire planaire. Cela permet d'entrelacer les arbres des digits de plusieurs façons différentes en fonction du nombre de préfixes d'entrée (nombre de digits d'entrée) et du nombre souhaité de niveaux de l'arbre. Bien sur, comme les préfixes sont combinées par un arbre binaire le nombre minimum de niveaux nécessaires est égal $\log_2(d)$ et le nombre maximum des niveaux est égal à $d-1$, où d est le nombre de digits d'entrée.

La Figure VI.9 présente trois façons de combiner les préfixes d'entrée en vue d'obtenir les huit retenues pour les huit préfixes d'entrée. Le premier arbre avec le nombre maximum de niveaux, est équivalent à une simple propagation de la retenue. Le troisième arbre, avec le nombre minimum de niveaux, est la structure proposée par Sklansky[VI.2].

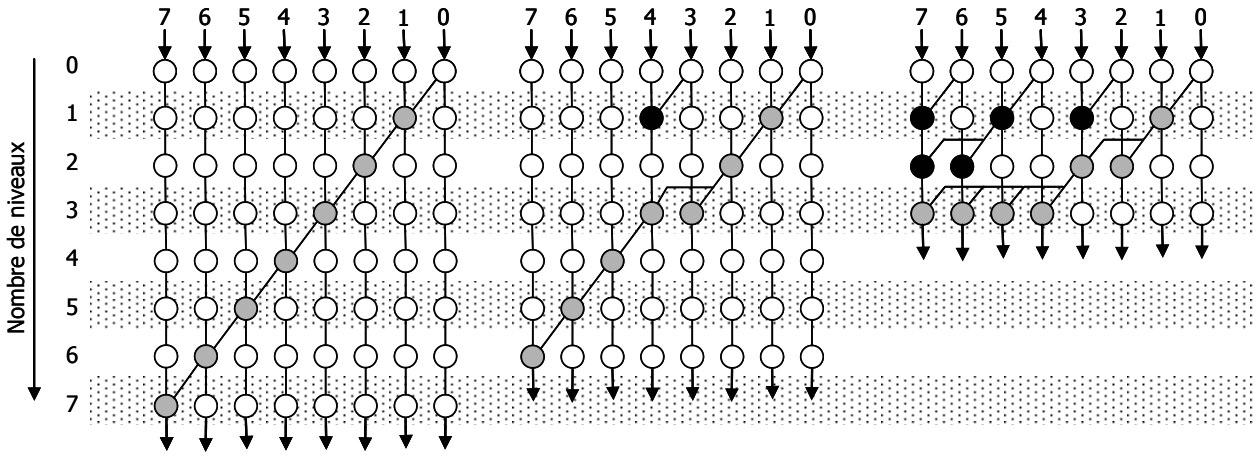


Figure VI.9 – Trois possibilités de combinaison des préfixes d'entrée.

Pour la construction de ces arbres, deux opérateurs sont nécessaires : l'opérateur *dot* (cercles noirs) et l'opérateur *dot simplifié* (cercle grisé). Les cercles blancs sont présentés simplement pour faciliter la compréhension et le préfixe *kpg* du niveau inférieur est retransmis au prochain niveau par les cercles blancs. Comme notation et pour expliquer la fonction réalisée des opérateurs *dot*, un préfixe sera noté par $kpg_{i:k}^{l}$. Un préfixe $kpg_{i:k}^{l}$ signifie l'en semble des préfixees de la position *i* jusqu'à la position *k* au niveau *l* de l'arbre. Si $kpg_{i:k}^{l}$ est égal à zéro cela implique que la retenue est absorbée de la position *i* jusqu'à la position *k*. De

façon analogue, si $kpg^{l}_{i:k}$ est égal à 1 cela implique que la retenue est propagée de la position *i* jusqu'à *k*.

En accord avec cette notation, les préfixes initiaux de l'arbre calculés par l'étage d'entrée sont de la forme $kpg^{0}_{i:i}$. L'opérateur *dot* est l'implémentation asynchrone proposé par cette thèse de la cellule de Brent-Kung [VI.3].

Pour obtenir un nouveau préfixe *kpg* l'opérateur *dot* peut être appliqué à deux autres préfixes pour réaliser la fonction suivante :

$$\begin{aligned} kpg^{l}_{i:k} &= (K^{l}_{i:k}, P^{l}_{i:k}, G^{l}_{i:k}) \\ kpg^{l}_{i:k} &= kpg^{l-1}_{i:j+1} \bullet kpg^{l-1}_{j:k} \\ &= (K^{l-1}_{i:j+1}, +P^{l-1}_{i:j+1}K^{l-1}_{j:k}, P^{l-1}_{i:j+1}P^{l-1}_{j:k}, G^{l-1}_{i:j+1}, +P^{l-1}_{i:j+1}G^{l-1}_{j:k}) \end{aligned} \qquad \text{(VI-13)}$$

et, évidemment, cet opérateur est toujours le même, indépendamment de la base choisie pour le codage des entrées *a* et *b*, car les préfixes *kpg* sont toujours codés en *1-parmi-3*. La Figure VI.10 présente l'opérateur *dot* – cercles noirs de l'arbre – et son implémentation.

Le préfixe le moins significatif d'entrée de l'arbre, notée par $kpg^{0}_{0:0}$, ne doit pas contenir de signal « propager » vu qu'il n'y a pas de retenue précédente à cet étage. Ainsi, le signal « propager » est additionné au signal « kill » et le résultat correspond à la retenue d'entrée de l'arbre de calcul de retenues. Donc,

$$ci_1 = co_0 = kpg^{0}_{0:0} \qquad \text{(VI-14)}$$

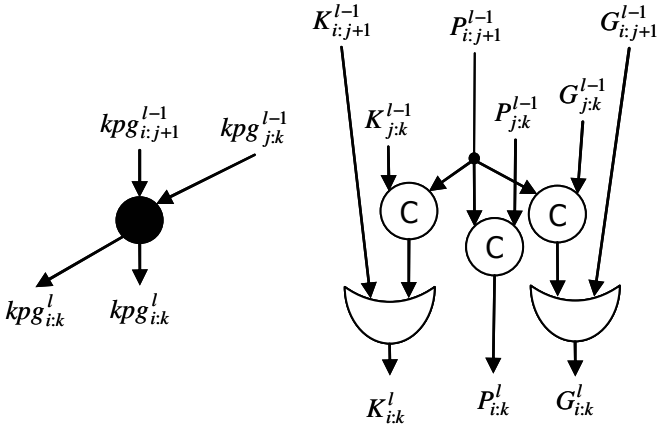


Figure VI.10 – L'opérateur *dot* et son implémentation.

De façon générale, à un niveau *m* de l'arbre de calcul, si tous les préfixes de la position zéro jusqu'à la position *i* sont assemblés et comme il n'y a pas de retenue précédente au niveau zéro à être propagée, le signal $P^{m}_{i:0}$ est toujours égal à zéro vu qu'une retenue est générée ou absorbée par le niveau *i*. Ainsi, la retenue de sortie d'un étage *i* peut être définie par :

$$\begin{aligned} ci_{i+1} &= co_i = kpg^m_{i:0} \\ co_i &= (Co0_i, Co1_i) = (K^m_{i:0}, G^m_{i:0}) \end{aligned} \qquad \text{(VI-15)}$$

Ainsi, les préfixes *kpg* de forme $kpg^m_{i:0}$ peuvent être codés en double rail pour simplifier l'arbre de calcul. Comme le démontre les équations VI-16, une cellule *dot* produisant une retenue de sortie doit recevoir une retenue à son entrée droite. Il faut souligner que l'opération réalisée par l'opérateur *dot* est distributive, mais elle n'est pas commutative. Ainsi, les cellules *dot* qui produisent les retenues de sortie de chaque position *i* peuvent être simplifiées.

$$\begin{aligned} co_i = kpg^m_{i:o} &= (K^l_{i:0}, G^l_{i:0}) \\ kpg^m_{i:0} &= kpg^{l-1}_{i:j+1} \bullet kpg^{l-1}_{j:0} \\ &= kpg^{l-1}_{i:j+1} \bullet co_j \\ &= (K^{l-1}_{i:j+1}, +P^{l-1}_{i:j+1} Co0_j, G^{l-1}_{i:j+1}, +P^{l-1}_{i:j+1} Co1_j) \end{aligned} \qquad \text{(VI-16)}$$

L'opérateur *dot* simplifié correspond aux cercles gris dans les arbres de la Figure VI.9 et cet opérateur réalise donc la fonction présentée par les équations VI-16. La Figure VI.11 présente l'opérateur *dot simplifié* et son implémentation asynchrone.

En plus, comme montré Figure VI.6, une cellule *dot simplifié* est employée dans l'étage de sortie pour le calcul de la retenue sortante de l'additionneur. Cette cellule remplace la dernière colonne de l'arbre de calcul. Tandis que toutes les sorties doivent être calculées avant d'émettre les signaux d'acquittements, il n'est pas nécessaire d'accélérer le calcul de la retenue sortante. La retenue sortante peut donc être calculé en même temps que le dernier digit de résultat de la somme.

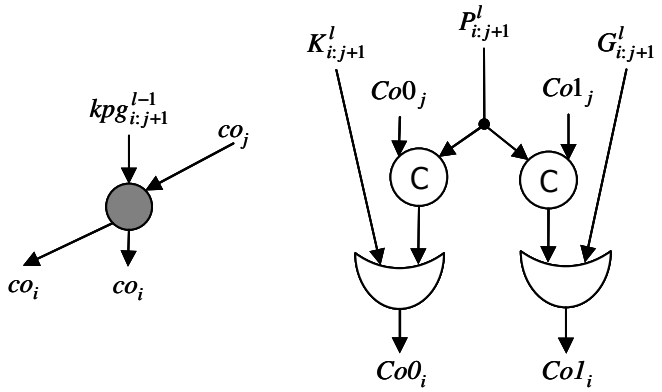
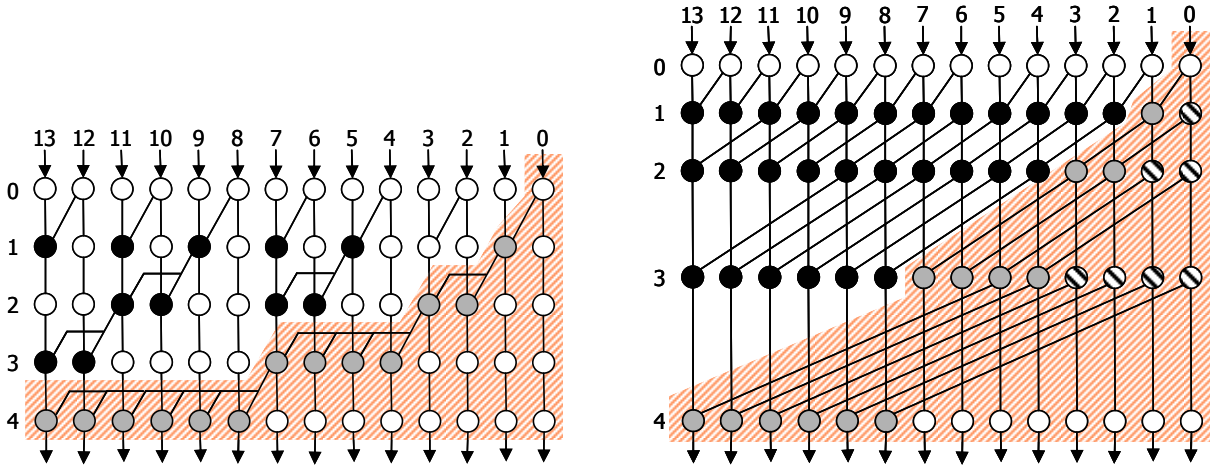


Figure VI.11 – L'opérateur *dot simplifié* et son implémentation.

Les opérateurs *dot* et *dot simplifié* peuvent donc être combinés de plusieurs façons en vue d'obtenir un arbre de calcul pour chaque retenue. Comme montre la Figure VI.9, accélérer le calcul des retenues augmente la complexité de l'arbre.

Dans le cadre de cette thèse, seulement les additionneurs de Sklansky ont été générés en raison de la régularité de l'arbre de calcul. La Figure VI.12.a montre l'arbre de calcul d'un additionneur de Sklansky à 14 digits. La région rayée correspond aux signaux double rail. Cependant, avec les opérateurs *dot* proposés ici, toutes les autres structures d'additionneurs peuvent être construites, comme les additionneurs de Brent-Kung [VI.3] et de Kogge-Stone [VI.4]. La Figure VI.12.b présente la structure proposée par Kogge-Stone que vise à limiter la sortance des opérateurs *dot*.



(a) (b)

Figure VI.12 – Deux arbres de calcul des retenue : (a) la structure de Sklansky et (b) la structure de Kogge-Stone.

Dans l'architecture de Kogge-Stone [VI.4], au prix d'un grand nombre d'opérateurs *dot*, la sortance de chaque opérateur est limitée à 2. Pour se faire, des « buffers » sont ajoutés à l'arbre de calcul – cercles rayés dans la Figure VI.12.b – afin de limiter la sortance des portes logiques.

Finalement, si l'additionneur doit être conçu pour recevoir une retenue d'entrée, les préfixes générés par l'étage d'entrée sont décalés à gauche de façon a ce que le préfixe $kpg_{0:0}^{0}$ soit la retenue d'entrée. Par conséquent, le fait d'ajouter la retenue d'entrée ajoute une colonne à l'arbre de calcul des retenues. La Figure VI.13 montre la structure générale pour les additionneurs à préfixe parallèle en présence d'une retenue entrante.

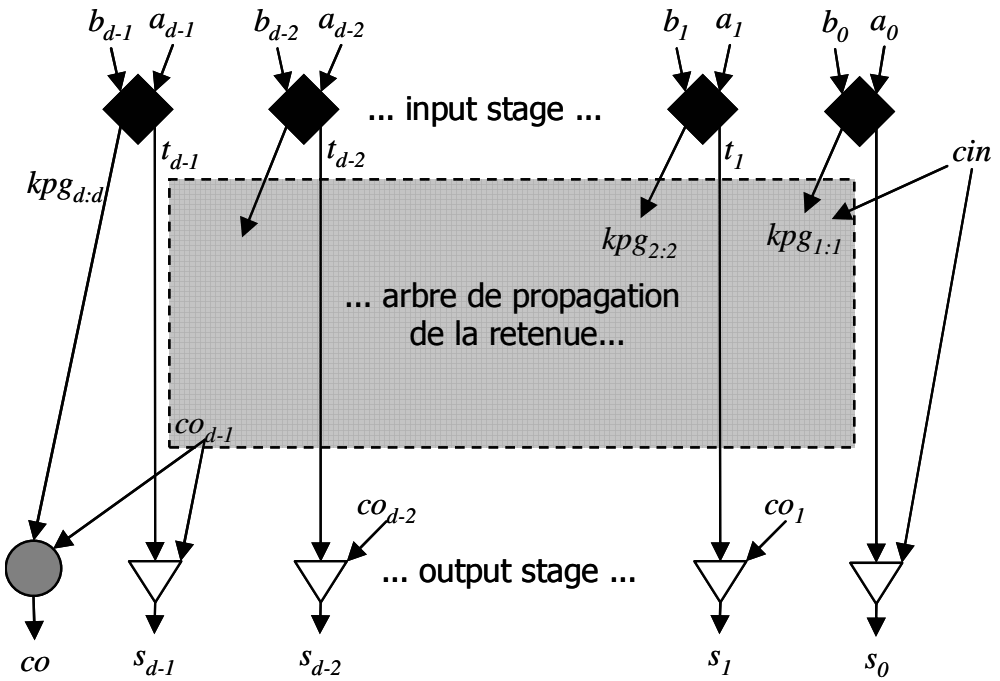


Figure VI.13 – Structure générale de base d'additionneurs à préfixe parallèle avec retenue entrante.

VI.2.3. Résultats

L'additionneur à retenue propagée et l'additionneur à préfixe parallèle de Sklansky ont été générés pour différentes bases et longueur (nombre de digits). La génération de ces additionneurs a été automatisée et intégrée dans l'outil de conception TAST.

Pour chacun des additionneurs l'énergie moyenne consommée, le retard critique et la complexité ont été évalués en utilisant la méthode proposée au chapitre IV de cette thèse. Pour comparer deux additionneurs conçus en bases distinctes, le nombre de digits ne suffit pas parce qu'un additionneur avec 4 digits en base binaire est équivalent à un additionneur avec 2 digits en base 4 (*1-parmi-4*). Donc la comparaison est faite par rapport à la plage de valeurs que l'additionneur peut représenter à sa sortie, c'est-à-dire, la base élevée au nombre de digits. Pour le calcul de retard et de consommation, une charge équivalente à 8 inverseurs de référence à été ajoutée à chaque fil de sortie.

Enfin, la complexité est mesurée par le nombre de transistors, le retard est mesuré en nombre d'inverseurs de référence à traverser et la consommation d'énergie est donnée par le nombre d'entrée des inverseurs de référence à piloter. Un retard de 100 inverseurs est le retard équivalent à une chaîne de 100 inverseurs. De façon analogue, une consommation de 100 inverseurs, est la consommation d'un nœud qui pilote 100 inverseurs.

VI.2.3.1.Additionneurs à Retenue Propagée

La Figure VI.14 montre les résultats d'estimation de la complexité par les additionneurs à retenue propagée. Comme attendu, la complexité des additionneurs à retenue propagée augmente avec le nombre de digits et aussi avec la base. L'augmentation de la base, même si elle conduit à une réduction du nombre de digits, cause une augmentation de la complexité parce que le nombre de minterms pour chaque cellule croît au carré de la base. Donc, la réduction du nombre de digits ne compense pas l'augmentation de la complexité de chaque cellule.

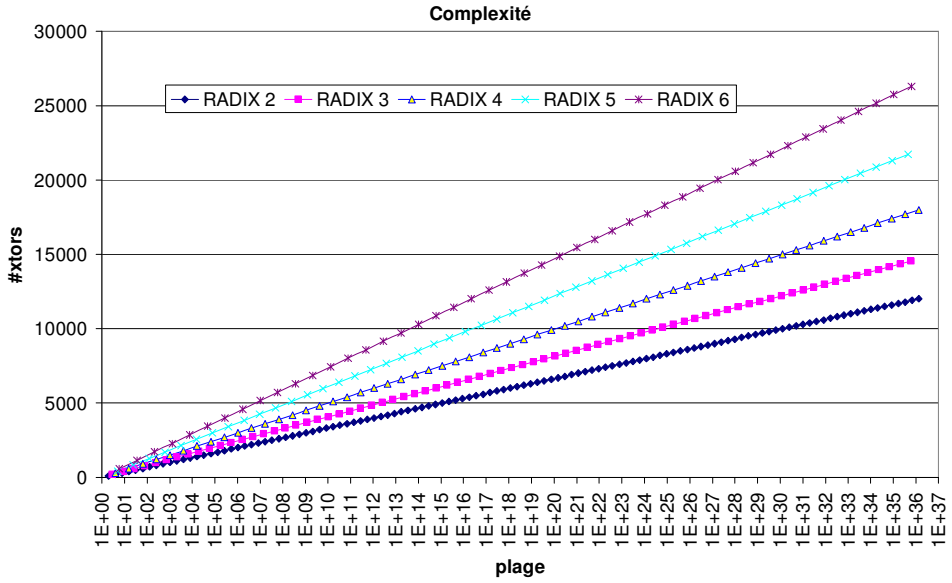


Figure VI.14 – Estimation de surface pour les additionneurs à retenue propagée.

Cependant, la réduction du nombre de digits et l'augmentation de la base conduisent à une réduction du retard critique, vu que moins de digits sont engagés dans le calcul et une réduction de l'énergie moyenne consommée par l'additionneur vu que plus d'information est porté par une simple transition. Ces conclusions peuvent être vérifiées par les résultats présentés dans la Figure VI.15 et dans la Figure VI.16.

La Figure VI.15 présente le retard critique de l'additionneur à retenue propagée pour plusieurs bases et longueurs. La Figure VI.16 montre les résultats de consommation d'énergie pour les mêmes additionneurs.

Comme il a été aussi attendu, les comportements de la complexité, de la consommation et du retard sont linéaires en fonction du nombre de digits pour chaque base en raison de la construction des additionneurs à retenue propagée.

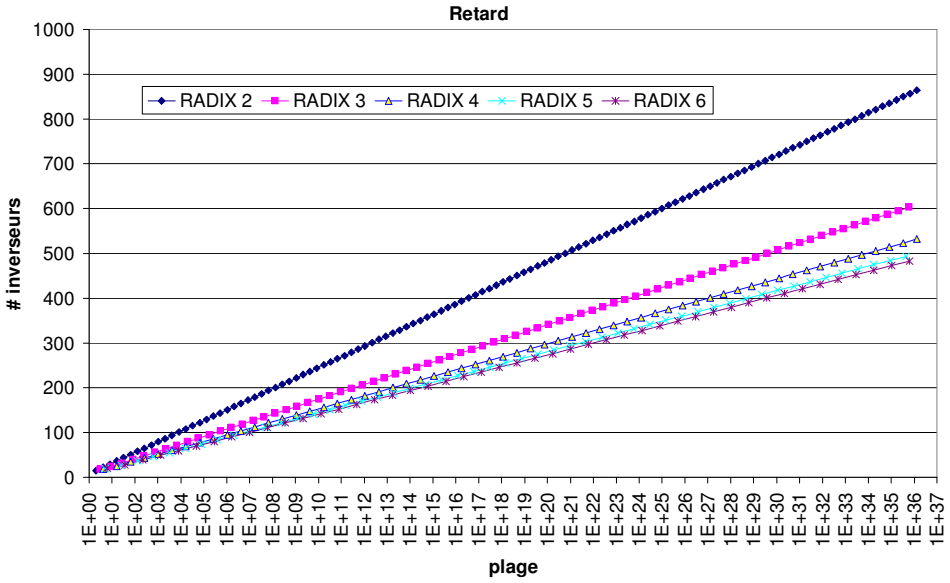


Figure VI.15 – Estimation du retard pour les additionneurs à retenue propagée.

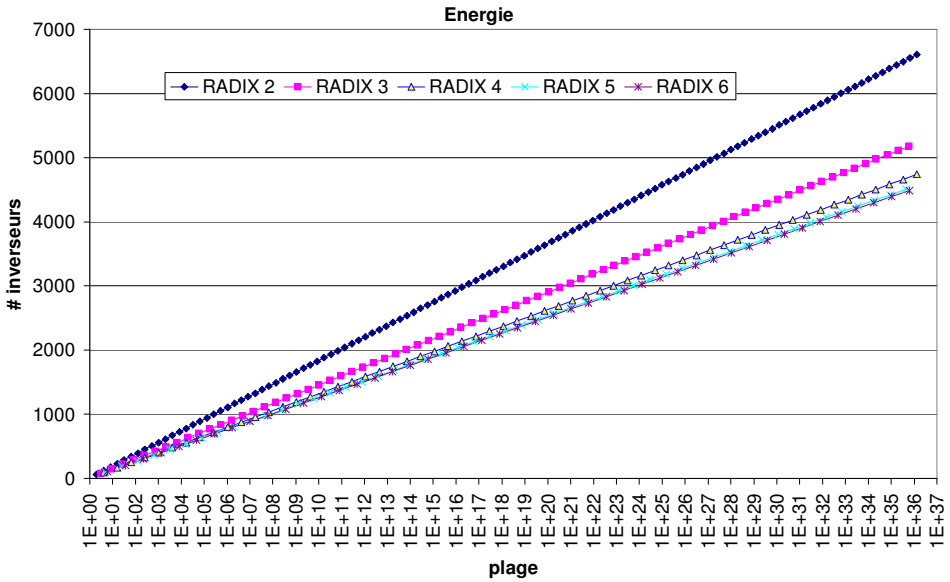


Figure VI.16 – Estimation d'énergie moyenne consommée par les additionneurs à retenue propagée.

Cependant, il a été possible d'observer que deux additionneurs équivalents en deux bases indépendamment du nombre de digits présentent un rapport constant de retard, consommation et complexité. Ces résultats ont été normalisés en fonction de l'additionneur à retenue propagée double rail et ces résultats sont présentés dans le graphique de la Figure VI.17. Comme par exemple, les résultats montrent qu'un additionneur *1-parmi-4* est 50% plus complexe que sa version équivalente double rail, cependant cet additionneur est 39% plus rapide et 28% moins consommant. En plus, si la tension d'alimentation de l'additionneur *1-parmi-4* pouvait être réduite de 39% de façon à obtenir le même retard que l'additionneur double rail, cela impliquerait une réduction d'environ 63% de l'énergie consommée par l'additionneur *1-parmi-4* conforme à la métrique Et^2. C'est-à-dire, que l'additionneur

1-parmi-4 avec le même retard consomme seulement 26,8% que la version équivalente en double rail.

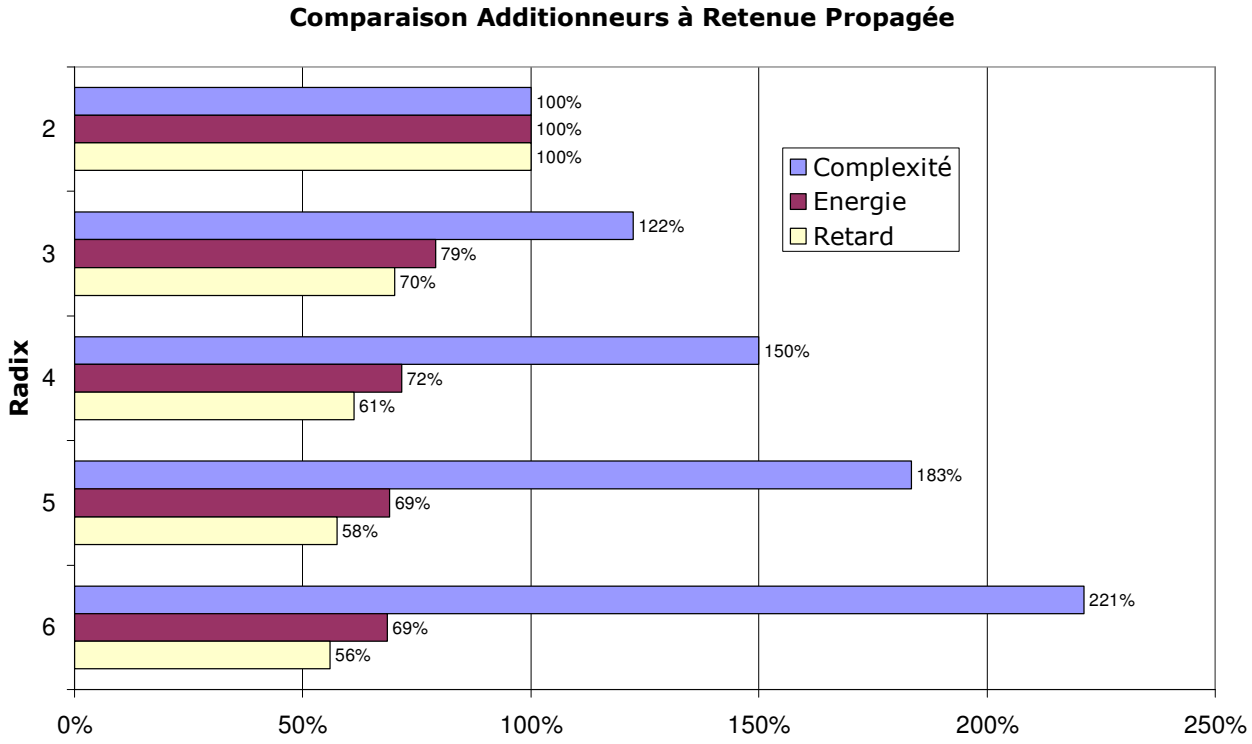


Figure VI.17 – Comparaison des additionneurs à retenue propagée.

La version double rail de la cellule de l'additionneur complet proposée par cette thèse contient 100 transistors. Il est important de souligner que cette version est basée sur de portes logiques Müller et portes logique OU et elle ne contient aucun type d'optimisation – ni même les transformations booléennes pour les logiques à deux niveaux. Malgré cette conception simplifiée, l'additionneur double rail optimisé présenté en [VI.5] compte 118 transistors. Les méthodologies de conception asynchrones présentées en [VI.5] et [VI.6] optimisent les circuits quand la totalité de minterms n'est pas requise par le circuit. Cependant, un additionneur doit implémenter tous les minterms en raison des caractéristiques de l'addition et ces méthodologies conduisent à une surcharge de la logique nécessaire à l'implémentation. En plus, ces méthodologies sont difficilement généralisées pour tous les codages *1-parmi-N*.

VI.2.3.2.Additionneurs à Préfixe Parallèle

La Figure VI.18 montre les résultats de complexité pour les additionneurs de Sklansky pour différentes bases et nombre de digits. Évidemment, la complexité des additionneurs augmente avec l'augmentation du nombre de digits. Toutefois, l'augmentation de la base n'implique pas forcement une augmentation de complexité comme pour les additionneurs à

propagation de la retenue. La Figure VI.19 présente un zoom autour des valeurs correspondantes aux additionneurs équivalents aux additionneurs 64 digits double rail. Les résultats montrent qu'au-delà d'un certain nombre de digits, l'augmentation de la base peut conduire à une réduction de complexité.

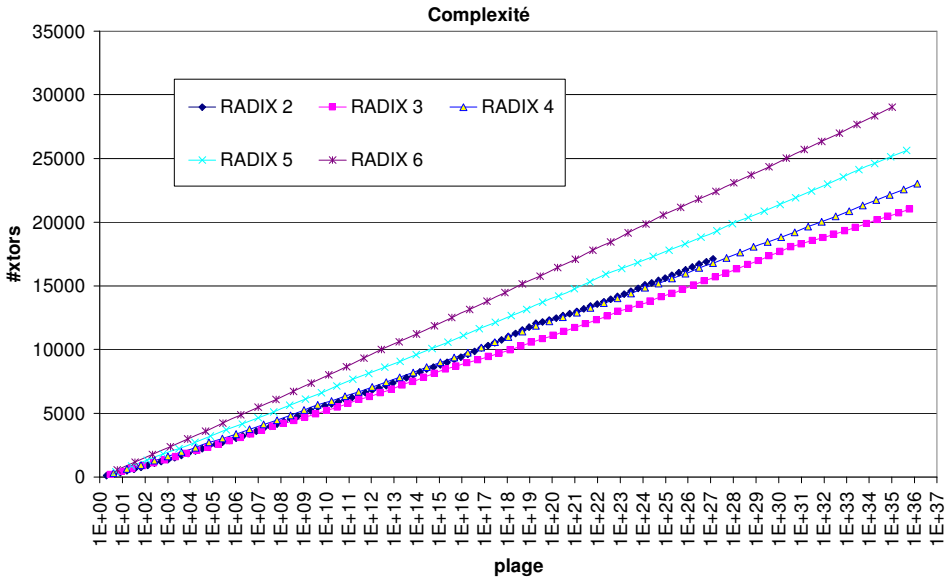


Figure VI.18 – Estimation de surface pour les additionneurs de Sklansky.

Ce comportement est facilement explicable. L'augmentation de la base implique une augmentation de la complexité des cellules des étages d'entrée et sortie des additionneurs à préfixe parallèle. Cependant, la complexité de l'arbre de calcul des retenues dépend exclusivement du nombre de digits. Ainsi, à partir d'un certain nombre de digits, l'augmentation de la base et la réduction du nombre de digits réduisent la complexité parce que la réduction de la complexité de l'arbre de calcul des retenues compense l'augmentation de complexité des étages d'entrée et de sortie.

La Figure VI.20 présente les résultats du retard critique pour les différents additionneurs de Sklansky. Dans la Figure VI.20, les retards ne sont pas linéaires avec le nombre de digits. Ce résultat est attendu en raison de la façon de construire l'arbre de calcul. On peut aussi observer que le changement de base ne modifie pas considérablement le retard. Comme le nombre de niveaux des arbres de Sklansky est donné par $\log_2(d)$, où d est le nombre de digits, le fait de multiplier par deux la base et, en conséquence, diviser par deux le nombre de digits réduit simplement d'un niveaux l'arbre de calcul.

La Figure VI.21 montre les résultats d'estimation de l'énergie moyenne consommée par les additionneurs à préfixe parallèle de Sklansky. Pour les résultats de la Figure VI.21, il est

clair que les changements de bases altèrent largement l'énergie consommée. Comme l'augmentation de la base diminue fortement le nombre de portes commutant aux étages d'entrée et sortie et, en plus, réduit la complexité de l'arbre de calcul des retenues, la consommation moyenne décroit drastiquement avec l'augmentation de la base.

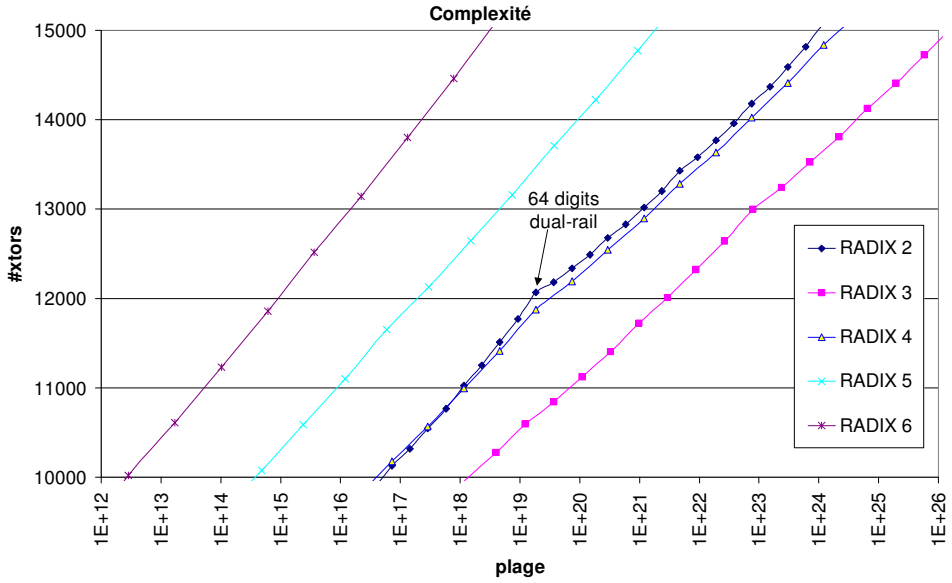


Figure VI.19 – Résultats de surface autour de 64 digits double rail.

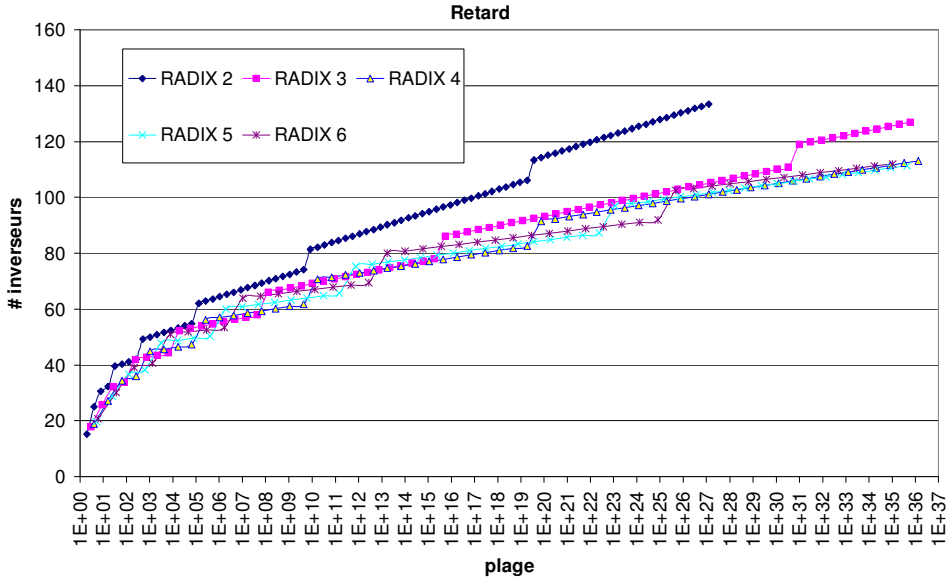


Figure VI.20 – Estimation du retard pour les additionneurs de Sklansky.

Les résultats pour les additionneurs à préfixe parallèle ne peuvent pas être généralisés à tous les nombres de digits comme cela a été le cas pour les additionneurs à propagation de la retenue. Ainsi, pour permettre au lecteur d'avoir une idée plus précise de l'impact de la base sur les additionneurs asynchrones, une comparaison a été faite entre plusieurs bases avec un additionneur de Sklansky à 64 digits double rail. Les additionneurs équivalents à un

additionneur 64 digits double rail ont été générés et leurs retard, consommation d'énergie et complexité ont été estimés.

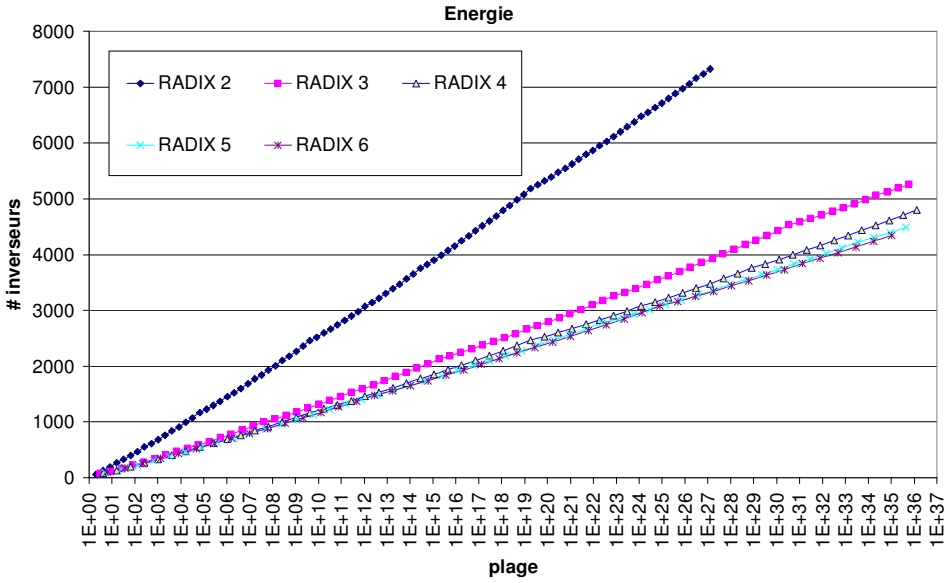


Figure VI.21 – Estimation de l'énergie moyenne consommé par les additionneurs de Sklansky.

La Figure VI.22 montre les résultats de cette comparaison. Cette comparaison démontre que le même additionneur réalisé en base *1-parmi-3* ou *1-parmi-4* est plus rapide et, en plus, est moins consommant et moins complexe. Pour les autres bases, les additionneurs présentent une réduction de consommation et du retard sans grande pénalité sur la complexité.

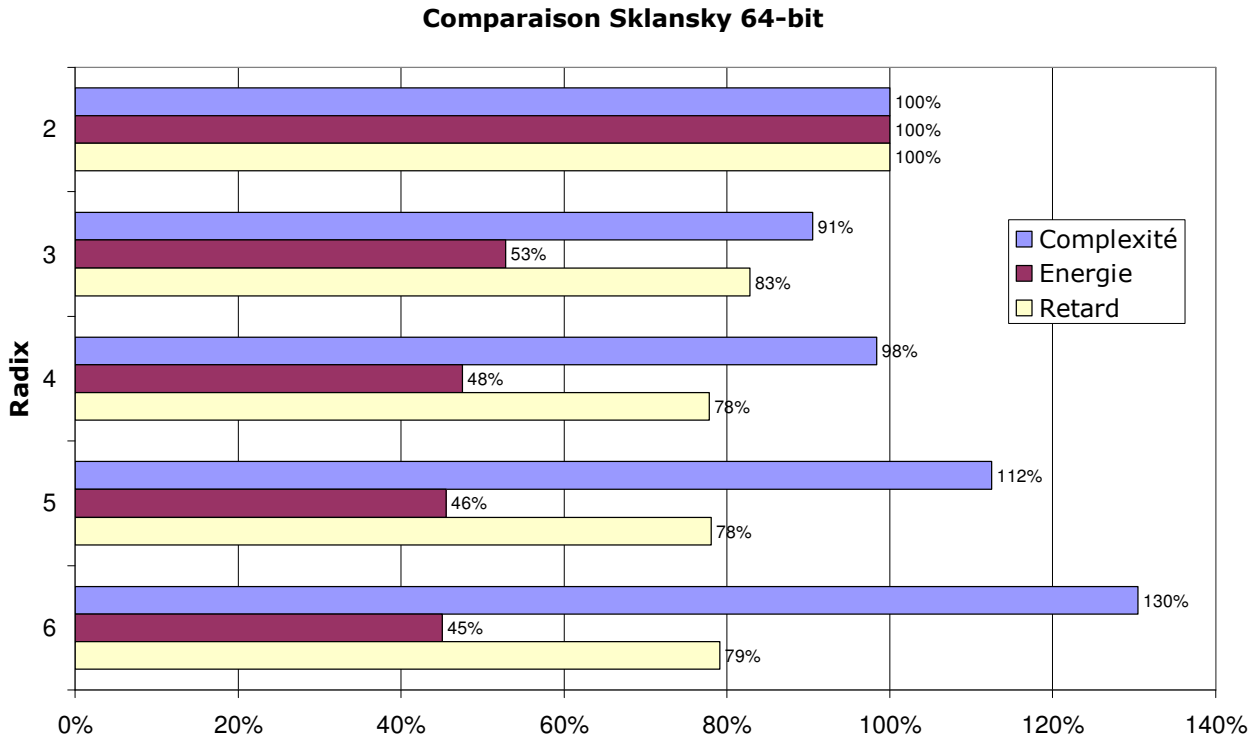


Figure VI.22 – Résultats pour différentes bases des additionneurs de Sklansky équivalent à 64 digits double rail.

Conformément à ces résultats et en utilisant la métrique Et^2, un additionneur 32 digits *1-parmi-4* avec la même performance que son équivalent 64 digits double rail consomme seulement 29% de l'énergie et il présente la même complexité que son équivalent double rail. Ces résultats montrent l'impact du choix de la base sur la performance des chemins de données asynchrones et la pertinence d'offrir des types de données multi-bases dans les langages et outils pour la synthèse.

VI.3. Conclusions

Ce chapitre a présenté une méthodologie de conception des additionneurs asynchrones QDI avec les codages *1-parmi-N*. La méthode est complètement automatisée et intégrée dans l'outil TAST. La méthode présentée ici permet la génération des additionneurs, et aussi bien de soustracteur, à propagation de retenue et des additionneurs rapides à préfixe parallèle pour toutes les bases. Ainsi, les opérateurs générés peuvent être insérés dans des chemins de données asynchrones *1-parmi-N* sans la nécessité d'interfaces de conversion de base. En plus, les additionneurs générés présentent une réduction de surface comparer à d'autres méthodes présentées dans la littérature [VI.5][VI.6]. L'utilisation d'une bibliothèque de cellules asynchrones contenant de cellules complexes peut encore améliorer les résultats de la méthodologie présentée ici.

Les résultats de cette étude ont aussi démontré que la base binaire, codage double rail asynchrone, n'est pas toujours la meilleure solution de représentation des valeurs pour la conception d'additionneurs rapides [VI.7][VI.8][VI.9]. L'augmentation de la base et la réduction du nombre de digits peuvent conduire à une réduction de la complexité et de l'énergie consommée et, en plus, d'une diminution du retard critique. Ainsi, la surface, l'énergie et la vitesse ne sont pas toujours des contraintes contradictoires dans les circuits asynchrones.

VI.4. Références

[VI.1] J. Sparsø, J. Staunstrup, M. Dantzer-Sørensen. Design of delay insensitive circuits using multi-ring structures. In: Proceedings of EURO–DAC, Hamburg, Germany, 1992. pp. 15-20.

[VI.2] J. Sklansky. Conditional-sum addition logic, IRE Trans. Electronic Computers, 9(2):226-231, 1960.

[VI.3] R. Brent, H. Kung. A Regular Layout for Parallel Adders. IEEE Transactions on Computers, vol. C-31, no. 3: 260-264. March, 1982.

[VI.4] P. Kogge, H. Stone. A Parallel Algorithm for the Efficient Solution of General Class of Recurrence Equation. IEEE Transactions on Computers, vol. 22, no. 8: 783-791. August, 1973.

[VI.5] A. Kondratyev, K. Lwin. Design of asynchronous circuits using synchronous CAD tools. IEEE Design and Test of Computers, vol. 19, no. 4:107-117. July, 2002.

[VI.6] I. David, R.Ginosar, M. Yoeli. An efficient implementation of Boolean functions as self-timed circuits. IEEE Transactions on Computers, vol. 41, no. 1:2-11. January, 1992.

[VI.7] Fragoso J., Sicard G., Renaudin M., Power/Area trade-offs in 1-of-M parallel-prefix asynchronous adders. In Proc. PATMOS'03 – 13th International Workshop on Power and Timing Modeling, Optmization and Simulation. Torino, Italy, Septembre, 2003.

[VI.8] Fragoso J., Sicard G., Renaudin M., Automatic Generation of 1-of-M QDI Asynchronous Adders. In Proc. SBCCI'03 – 16th Symposium on Integrated Circuits and System Design. São Paulo, Brazil, Septembre, 2003.

[VI.9] Fragoso J., SICARD G., Renaudin M., Generalized 1-of-M QDI Asynchronous Adders. In: 3RD ACID-WG WORKSHOP, Heraklion, 2003.

Chapitre VII

Multiplieurs

La multiplication est une autre opération largement employée dans les systèmes numériques actuels. Avec l'augmentation de la capacité d'intégration de complexité dans une même puce, les systèmes numériques qui utilisent intensivement la multiplication normalement optent pour une implémentation physique de la multiplication dans leurs chemins de données. Ainsi, cette thèse s'est aussi intéressée à la réalisation de la multiplication en plusieurs bases avec les différents types de codage de données. Même si seulement les multiplieurs à matrice de calcul sont étudiés au cours de ce chapitre, cet étude offre une vision généralisée de la multiplication qui permettra, par la suite, de concevoir d'autres architectures de multiplieurs.

VII.1. Multiplication de Nombres Entiers Non Signés

La valeur d'un nombre entier représenté avec *d* digits dans une base *N* est donné par :

$$A = \sum_{i=0}^{d-1} a_i N^i \qquad \text{(VII-1)}$$

Ainsi, la multiplication de deux valeurs *A* et *B*, non signées, avec *d* digits peut être exprimée par :

$$P = AxB = \sum_{i=0}^{d-1} \sum_{j=0}^{d-1} a_i b_j N^{i+j} \qquad \text{(VII-2)}$$

Cette équation montre que chaque digit de *A* doit être multiplié par tous les autres digits de *B* et le résultat de chaque multiplication doit ensuite être additionné en respectant la position

du nouveau digit. De façon analogue, la multiplication de deux digits restitue un produit partial et le résultat de la multiplication est la somme de tous les produits partiels.

La Figure VII.1 montre le plus simple multiplieur binaire de 4 digits [VII.1]. Ce multiplieur est composé d'une matrice de calcul et chaque ligne de la matrice calcule un produit partiel qui est ensuite additionné au produit partiel de la ligne précédente. L'opération de multiplication de deux digits binaires – cellules *PP* de la Figure VII.1 – est équivalente à l'opération « ET » à deux entrées.

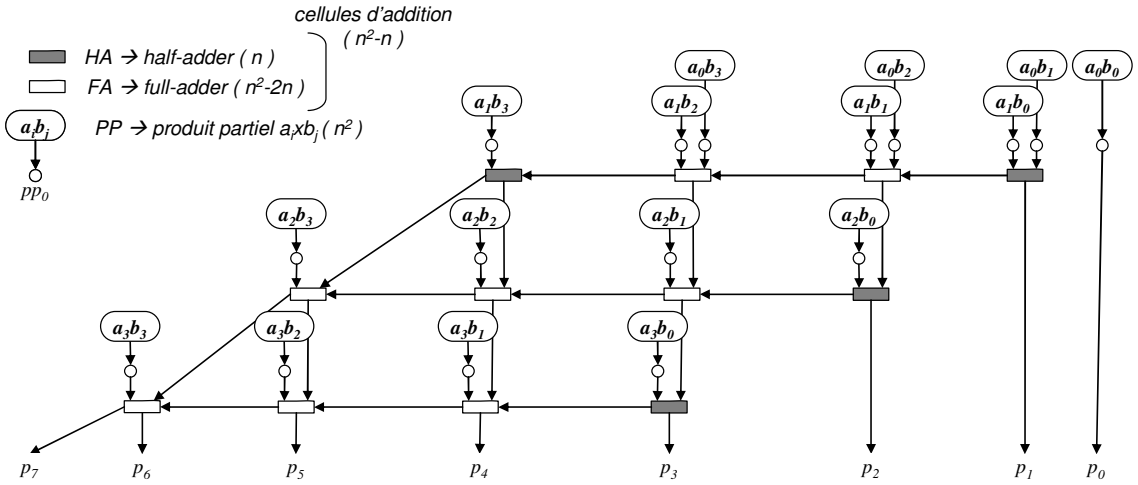


Figure VII.1 – Matrice de multiplication binaire sans signe.

Cependant, cette matrice de multiplication ne peut pas être employée pour la multiplication dans les autres bases car le produit d'un digit binaire par un autre digit binaire produit seulement un digit binaire. Comme cette propriété de la base binaire n'est pas valable dans les autres bases, la matrice de la Figure VII.1 ne peut pas être utilisée pour la multiplication dans une base *N* quelconque.

Par exemple, la multiplication d'un digit codé en base 4 – avec le codage *1-parmi-4* – multiplié par un autre digit codé en base 4 produit deux digits en base 4. De façon générale, si la base est supérieure à 2, la multiplication de deux digits produira deux autres digits dans la même base. Ainsi, la multiplication de deux digits peut être exprimée par :

$$
\begin{aligned}
& a_i b_j N^{i+j} = (pp_1) N^{i+j+1} + (pp_0) N^{i+j} \\
& (pp_1) = \left\lfloor \frac{a_i b_j}{N} \right\rfloor \\
& (pp_0) = (a_i b_j) \bmod (N)
\end{aligned}
\qquad \text{(VII-3)}
$$

À partir de ces équations, il est facile d'extraire l'équation pour chaque fil de sortie des deux digits des cellules du produit partiel et ainsi généraliser et automatiser la conception de ces cellules. Pour la matrice d'addition, deux matrices similaires à la matrice binaire sont nécessaires pour réaliser la somme de tous les produits partiels. Les résultats de ces deux matrices doivent être additionnés pour produire le résultat final de la multiplication.

La Figure VII.2 montre la matrice généralisée pour la multiplication de deux nombres entiers sans signe en base *N*. Les cellules de l'additionneur complet (FA) et les cellules des demi additionneurs sont les mêmes cellules généralisées que dans le chapitre VI.

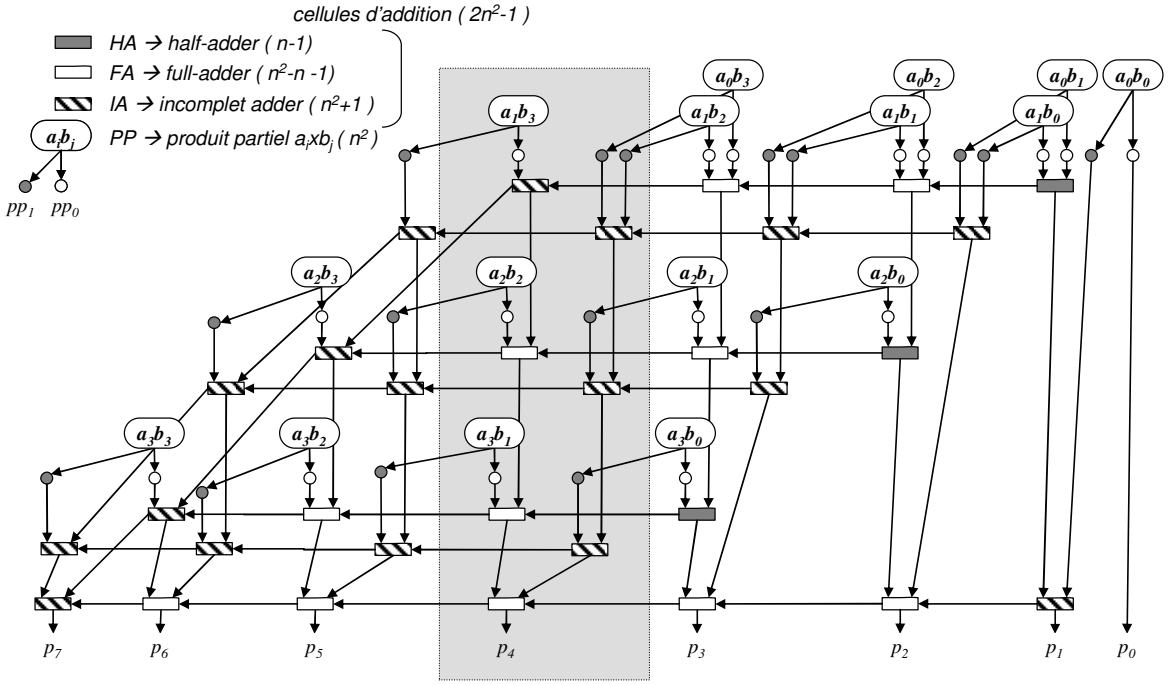


Figure VII.2 – Matrice de multiplication sans signe généralisée.

La généralisation de la multiplication sans signe permet de comprendre certaines propriétés de la multiplication dans une base *N*. Tout d'abord, le produit de deux digits donne deux autres digits : pp_0 et pp_1. Un digit en base *N* peut coder des valeurs comprises entre 0 et *N-1*, cependant, le digit le plus significatif – digit pp_1 – résultant du produit de deux autres digits ne vaut jamais *N-1*. Ainsi, la valeur du digit pp_1 est comprise entre 0 et *N-2* et cette valeur peut être codée avec un fil de moins.

Deuxièmement, certaines cellules d'addition présentes dans la matrice de calcul peuvent être simplifiées – cellules *IA* (de l'anglais « incomplete adder ») de la Figure VII.2. Ces cellules ici appelées additionneurs incomplets ne sont pas des additionneurs en base *N-1*. Même si, pour certains cas, les deux valeurs de l'addition sont codées en base *N-1*, l'addition doit être toujours effectuée en base *N* pour assurer le calcul correct de chaque ligne de produit partiel. Ainsi, toutes les cellules d'addition dans la matrice de multiplication effectuent une

somme en base *N* et les cellules incomplètes sont aussi des additionneurs en base *N* mais dont certaines des entrées ne prennent jamais toutes les valeurs possibles pour un digit codé en base *N*.

Il est clair que généraliser les cellules incomplètes d'addition pour toutes les bases et pour toutes les combinaisons possibles des valeurs d'entrées n'est pas envisageable. Ainsi, l'étude de la multiplication de nombres non signés montre la nécessité de soit générer chaque cellule d'addition pour chaque condition d'utilisation, soit de générer toute la matrice de calcul et, ensuite, effectuer une étape de simplification pour supprimer les portes logiques qui ne commuteront jamais. Dans le contexte de cette thèse, la génération de matrices de multiplication non signées a été automatisée et l'option d'effectuer une étape de simplification après génération a été retenue.

VII.2. Multiplication de Nombres Entiers Signés

La multiplication de nombres binaires signés est aussi une opération bien connue des concepteurs de circuits intégrés. La base binaire offre une représentation simple de nombres signés en utilisant le complément à la base. Une valeur entière *A* codée avec *d* digits binaires est exprimée par :

$$A = -a_{d-1}2^{d-1} + \sum_{i=0}^{d-2} a_i 2^i \qquad \text{(VII-4)}$$

où le digit le plus significatif code le signe de la valeur. De façon analogue, toutes les valeurs binaires commençant par 0 ne sont pas négatives et toutes les valeurs commençant par 1 sont des valeurs négatives. Ainsi, avant de continuer, il faut déterminer une représentation équivalente dans toutes les bases pour pouvoir généraliser la multiplication de nombres signés.

Pour faciliter la compréhension de la multiplication de nombres signés, une notation redondante a été choisie pour l'écriture de nombres signés dans toutes les bases et l'impact de cette notation sera discuté au cours de ce chapitre. Ainsi, en ajoutant un digit de signe supplémentaire, la valeur d'un nombre signé avec *d* digits en base *N* peut être écrit par :

$$A = -a_d N^d + \sum_{i=0}^{d-1} a_i N^i \qquad \text{(VII-5)}$$

où a_d est égal à 0 pour les valeurs non négatives et a_d est égal à 1 pour les valeurs négatives.

Il est clair que le nombre signé est codé en utilisant le complément à la base comme dans le chapitre VI. Pour les bases paires, il est suffisant de tester le digit le plus significatif

pour détecter le signe, mais pour les bases impaires, il faut tester tous les digits pour savoir si la valeur est négative ou non. Comme, il est possible de détecter le signe par la valeur elle-même le digit de signe est redondant, mais cette notation permet de généraliser l'algorithme de multiplication pour les nombres signés.

À partir de cette notation, il est maintenant possible d'effectuer la multiplication de deux nombres signés dans une base *N*.

$$A = -a_d \times N^d + \sum_{i=0}^{d-1} a_i \times N^i \qquad B = -b_d \times N^d + \sum_{j=0}^{d-1} b_j \times N^j$$

$$A = \left(-N \times a_d + a_{d-1}\right) \times N^{d-1} + \sum_{i=0}^{d-2} a_i \times N^i \qquad B = \left(-N \times b_d + b_{d-1}\right) \times N^{d-1} + \sum_{j=0}^{d-2} b_j \times N^j$$

$$\begin{aligned} P &= A \times B \\ &= a_d \times b_d \times N^{2d} + \left(-N \times a_d \times b_{d-1} - N \times a_{d-1} \times b_d + a_{d-1} \times b_{d-1}\right) \times N^{2d-2} \\ &+ \left(-N \times a_d + a_{d-1}\right) \times \sum_{j=0}^{d-2} b_j \times N^{j+d-1} + \left(-N \times b_d + b_{d-1}\right) \times \sum_{i=0}^{d-2} a_i \times N^{i+d-1} \\ &+ \sum_{i=0}^{d-2} \sum_{j=0}^{d-2} a_i \times b_j \times N^{i+j} \end{aligned} \quad (\text{VII-6})$$

Ces équations permettent d'identifier trois régions de calcul des produits partiels :

- ➔ la région non signée, ou indépendante des signes de *A* et *B* : cette région est équivalente à un multiplieur non signé avec *d-1* digits ;
- ➔ la région signée/non signée : deux parties de la multiplication dépendante de la valeur du signe d'une des deux variables ;
- ➔ la région signée : zone de multiplication dépendante des deux valeurs de signe.

La Figure VII.3 présente les trois régions de multiplication d'un multiplieur signé généralisé.

VII.2.1. La Région Signée / Non Signée

La région signée/non signée d'un multiplieur généralisé réalise la multiplication du digit le plus significatif et du digit de signe d'une variable par tous les digits, sauf le digit le plus significatif, de l'autre variable. Cette région doit calculer l'expression :

$$\left(-N \times a_d + a_{d-1}\right) \times \sum_{j=0}^{d-2} b_j \times N^{j+d-1} \qquad \text{(VII-7)}$$

En utilisant les propriétés des valeurs codées en complément à la base, deux solutions sont possibles pour cette expression. La première solution consiste à toujours réaliser le complément à la base-1 et ensuite additionner 1 pour produire la valeur correcte. Ainsi :

$$\left(-N\times a_d + a_{d-1}\right)\times\sum_{j=0}^{d-2} b_j \times N^{j+d-1} = \left(\sum_{j=0}^{d-2}\left(N\times\overline{a_d}\times b_j + a_{d-1}\times b_j\right)\times N^{j+d-1}\right) + 1\times N^d \quad \text{(VII-8)}$$

et cette solution correspond à la solution usuelle binaire.

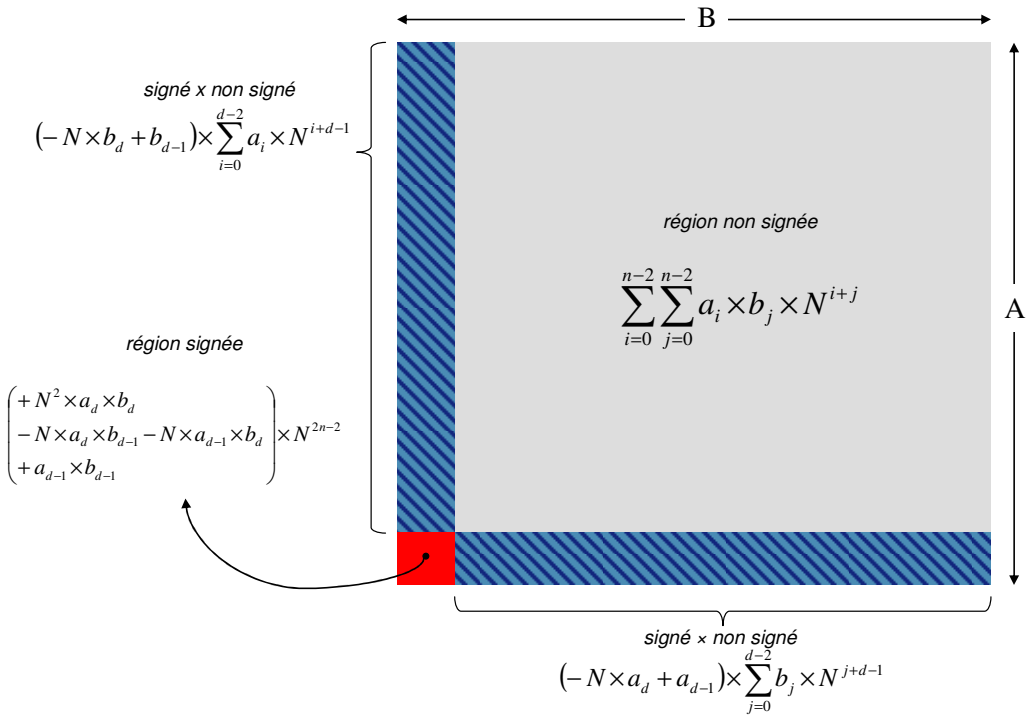


Figure VII.3 – Les trois régions de multiplication d'un multiplieur signé généralisé.

Une autre solution possible est de réaliser le complément à la base seulement si le digit de signe est égal à 1, c'est-à-dire, le complément à la base est effectué seulement si la valeur est négative. Ainsi, l'expression de la région signée/non signée peut être exprimée par :

$$\left(-N\times a_d + a_{d-1}\right)\times\sum_{j=0}^{d-2} b_j \times N^{j+d-1} = \left(\sum_{j=0}^{d-2}\left(N\times a_d\times\overline{b_j} + a_{d-1}\times b_j\right)\times N^{j+d-1}\right) + a_d\times N^d \quad \text{(VII-9)}$$

Il est facile de vérifier que ces deux solutions sont équivalentes. En effet, la deuxième solution correspond à l'optimisation proposée par Baugh et Wooley [VII.2] pour les multiplieurs binaires. L'optimisation de Baugh et Wooley évite le complément à la base des nombres positifs et, par conséquence, cette optimisation évite le long chemin de retenue pour retransformer la valeur du produit partiel à nouveau en une valeur positive.

En regardant les deux solutions possibles pour la région signée/non signée, le lecteur doit se demander pourquoi le digit le plus significatif a_{d-1} a été gardé pour cette région, car le produit de ce digit par b_j ne dépend pas de la valeur du digit de signe. En effet, le calcul de ce

produit partiel pourrait s'effectuer en deux étapes. Cependant, la multiplication de a_{d-1} pour b_j produit deux digits et la deuxième étape consisterait à additionneur le deuxième digit au résultat du produit de la valeur du signe par b_j. En conséquence, la cellule de produit partiel de a_{d-1} pour b_j peut prendre en compte la valeur du signe pour produire directement la valeur du produit partiel et éviter ainsi une addition supplémentaire. En plus, pour les bases paires, le signe peut être détecté directement à partir de la valeur de a_{d-1}. Ainsi, les cellules de produit partiel signé/non signé dans les bases paires n'ont pas besoin d'un digit de signe explicite pour la réalisation du calcul du produit partiel.

Si la deuxième solution est choisie, les équations suivantes généralisent le calcul de deux digits de résultat d'une cellule de produit partiel signé/non signé.

$$\begin{aligned}(pp_1) &= \left\lfloor \frac{a_d(N-1-b_j)N + a_i b_j}{N} \right\rfloor \\ (pp_0) &= \left(a_d(N-1-b_j)N + a_i b_j\right) \bmod(N)\end{aligned} \tag{VII-10}$$

VII.2.2. La Région Signée

La région signée est la région responsable de la multiplication des deux digits les plus significatifs de chaque variable et les deux digits de signe. Cette région calcule l'expression suivante :

$$\left(-N \times a_d \times b_{d-1} - N \times a_{d-1} \times b_d + a_{d-1} \times b_{d-1}\right) \times N^{2d-2} \tag{VII-11}$$

De façon analogue à la région signée/non signée, deux solutions sont possibles pour l'implémentation de cette multiplication. La première solution complémente toujours *A* et *B* et la deuxième réalise le complément en fonction des valeurs de signe de *A* et *B*. Les deux équations suivantes montrent les deux solutions envisageables.

$$\begin{aligned}&\left(-N \times a_d \times b_{d-1} - N \times a_{d-1} \times b_d + a_{d-1} \times b_{d-1}\right) \times N^{2d-2} \\ &= \left(N \times \left(\overline{a_d \times b_{d-1}} + \overline{a_{d-1} \times b_d} + 2\right) + a_{d-1} \times b_{d-1}\right) \times N^{2d-2}\end{aligned} \tag{VII-12}$$

$$\begin{aligned}&\left(-N \times a_d \times b_{d-1} - N \times a_{d-1} \times b_d + a_{d-1} \times b_{d-1}\right) \times N^{2d-2} \\ &= \left(N \times \left(a_d \times \overline{b_{d-1}} + b_d \times \overline{a_{d-1}} + a_d + b_d\right) + a_{d-1} \times b_{d-1}\right) \times N^{2d-2}\end{aligned} \tag{VII-13}$$

Évidemment, la solution choisie pour l'implémentation de la région signée doit être la même que la solution choisie pour l'implémentation de la région signée/non signée. En choisissant la même solution, les retenues provenant de la région signée/non signée peuvent être utilisées pour la correction de la valeur sortante. Ainsi, il est suffisant de réaliser le

complément à la base dans la région signée/non signée et propager les retenues a la région signée.

Si la deuxième solution est choisie pour les deux régions, les équations suivantes généralisent la construction de la cellule de produit partiel de la région signée pour les deux digits de résultat.

$$(pp_1) = \left\lfloor \frac{a_d(N-1-b_{d-1})N + b_d(N-1-a_{d-1})N + a_{d-1}b_{d-1}}{N} \right\rfloor \qquad \text{(VII-14)}$$
$$(pp_0) = \left(a_d(N-1-b_{d-1})N + b_d(N-1-a_{d-1})N + a_{d-1}b_{d-1}\right) \bmod (N)$$

Le terme $a_d \times b_d \times N^{2d}$ n'a été pas pris en compte, car il est simple de comprendre que ce terme correspond à la valeur du signe sortant du résultat de la multiplication. De façon générale, ce terme n'a pas besoin d'être conçu, parce que le signe peut être détecté par le résultat de la multiplication. Cependant, si le multiplieur doit produire une valeur de signe explicite, ce terme est simplement la fonction XOR des deux signes entrants et les retenues sortantes provenant de la matrice de calcul peuvent être ignorées.

La Figure VII.4 présente la matrice de multiplication signée généralisée. Cette matrice utilise la deuxième solution et, en conséquence, les valeurs des signes de *A* et *B* sont utilisées pour la construction correcte de la valeur de sortie. Si, la première solution avait été choisie, deux valeurs constantes égales à 1 remplaceraient les valeurs de signes de *A* et *B*.

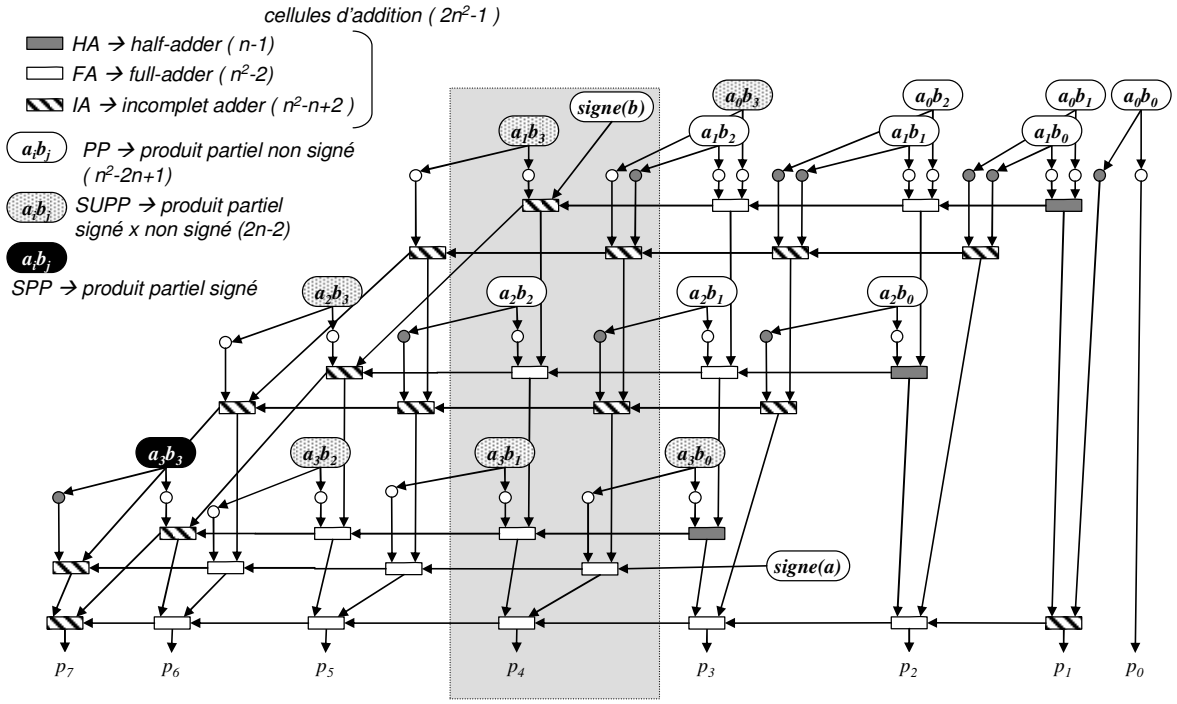


Figure VII.4 - Matrice de multiplication signée généralisée.

Les cellules *SUPP* et *SPP* de la Figure VII.4 sont, respectivement, les cellules de la région signée/non signée et de la région signée. Pour des raisons de simplicité, les valeurs des signes de *A* et *B* ne sont pas connectées au cellules *SUPP* et *SPP*. De toute façon, ces cellules dans les multiplieurs en base paire n'ont pas besoin de la valeur de signe pour le calcul du produit partiel.

Cependant, il est possible de concevoir, même pour les multiplieurs en base paire, les cellules *SUPP* et *SPP* avec un signal explicite de signe. Cela permet d'utiliser un multiplieur signé pour réaliser une multiplication non signée avec tous les digits simplement en gardant les valeurs des signes égales à zéro.

VII.3. Résultats

La génération des multiplieurs pour toutes les bases a été automatisée. Cette génération utilise les additionneurs généralisés du chapitre VI et les cellules de calcul ont été généralisées à partir des équations présentées au cours de ce chapitre. En plus, les cellules de produit partiel ont été conçues en utilisant une implémentation DIMS et ces cellules ont été optimisées avec les recommandations présentées dans le chapitre V. La Figure VII.5 montre le résultat de la génération d'une cellule de produit partiel non signé en utilisant le codage *1-parmi-4*. La cellule est un bloc logique faiblement indiqué.

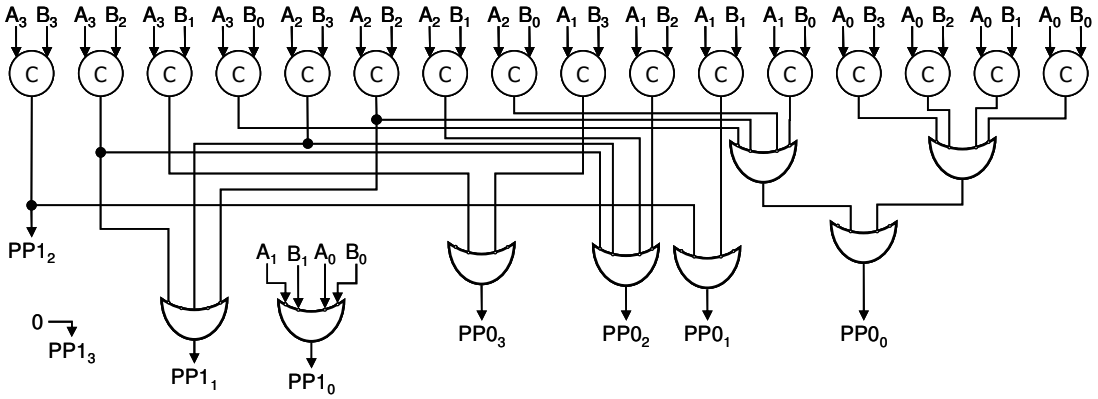


Figure VII.5 – Cellule de produit partiel non signé avec le codage *1-parmi-4*.

Plusieurs multiplieurs ont été générés et évalués avec la méthode proposée par cette thèse. Les multiplieurs en base impaire ont été conçus avec des signaux de signe entrants explicites et les multiplieurs en base paire ont été conçus sans l'utilisation des signaux de signe explicite. Ainsi, les résultats ne prennent pas en compte la détection de la valeur de signe.

La Figure VII.6 montre les résultats de complexité pour plusieurs multiplieurs signés dans différentes bases. Comme il a été fait pour les additionneurs, l'axe des abscisses

représente la plage des valeurs qui peuvent être représentées pour chaque entrée du multiplieur. De cette façon, il est possible de comparer deux multiplieurs équivalents.

L'augmentation de la base réduit le nombre de digits nécessaires au calcul de la multiplication. Cependant, chaque cellule de multiplication est plus complexe et, en plus, chaque cellule produit deux digits de sorties. Les cellules d'addition deviennent elles aussi plus complexes avec l'augmentation de la base. En conséquence, comme le montre les résultats de la Figure VII.6, la réduction du nombre de digits ne suffit pas à compenser l'augmentation de la complexité des cellules de base. Ainsi, l'augmentation de la base provoque une augmentation de la complexité des multiplieurs.

Cependant, les bases paires sont plus avantageuses que les bases impaires. Dans le graphique de la Figure VII.6, la base 3 présente une complexité supérieure à la base 4. Ce comportement est du au fait que les cellules de produit partiel de la région signée/non signée et de la région signée nécessitent une signal explicit de signe. Ainsi, ces cellules sont plus complexes que les cellules équivalentes en base *4* et, par conséquence, le multiplieur en base *3* est plus complexe que son équivalent en base *4*.

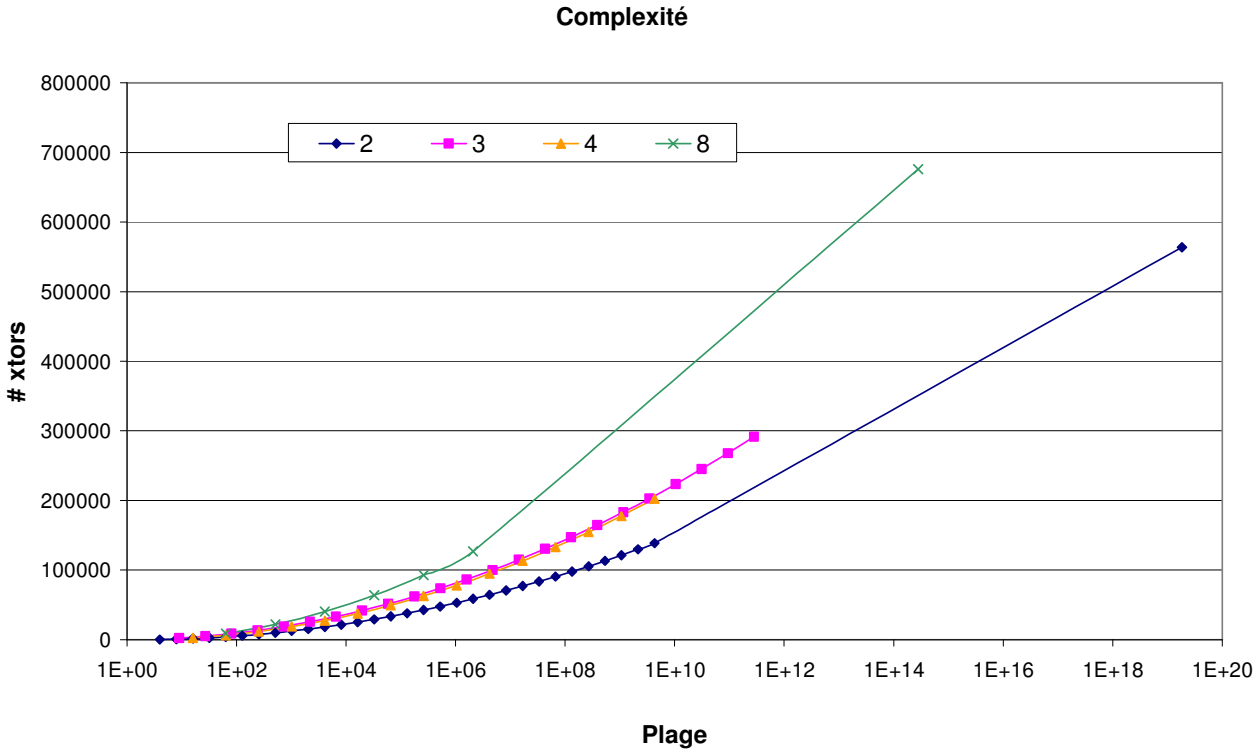


Figure VII.6 – Résultats de complexité pour les multiplieurs signés en plusieurs bases.

La Figure VII.7 présente les résultats de consommation d'énergie moyenne pour les différents multiplieurs. Les résultats montrent que l'augmentation de la base pour les

multiplieurs avec peu de digits d'entrées semble réduire la consommation. Ce comportement est une conséquence directe du fait que l'augmentation de la base réduit le nombre de transitions nécessaires pour coder la même information. Cependant, avec l'augmentation de la complexité des cellules de base, la charge de chaque nœud est plus importante. Ainsi, pour un nombre élevé de digits d'entrée, l'augmentation de la base doit être significative pour que la réduction du nombre de transitions puisse compenser l'augmentation de la complexité et de la charge de chaque nœud.

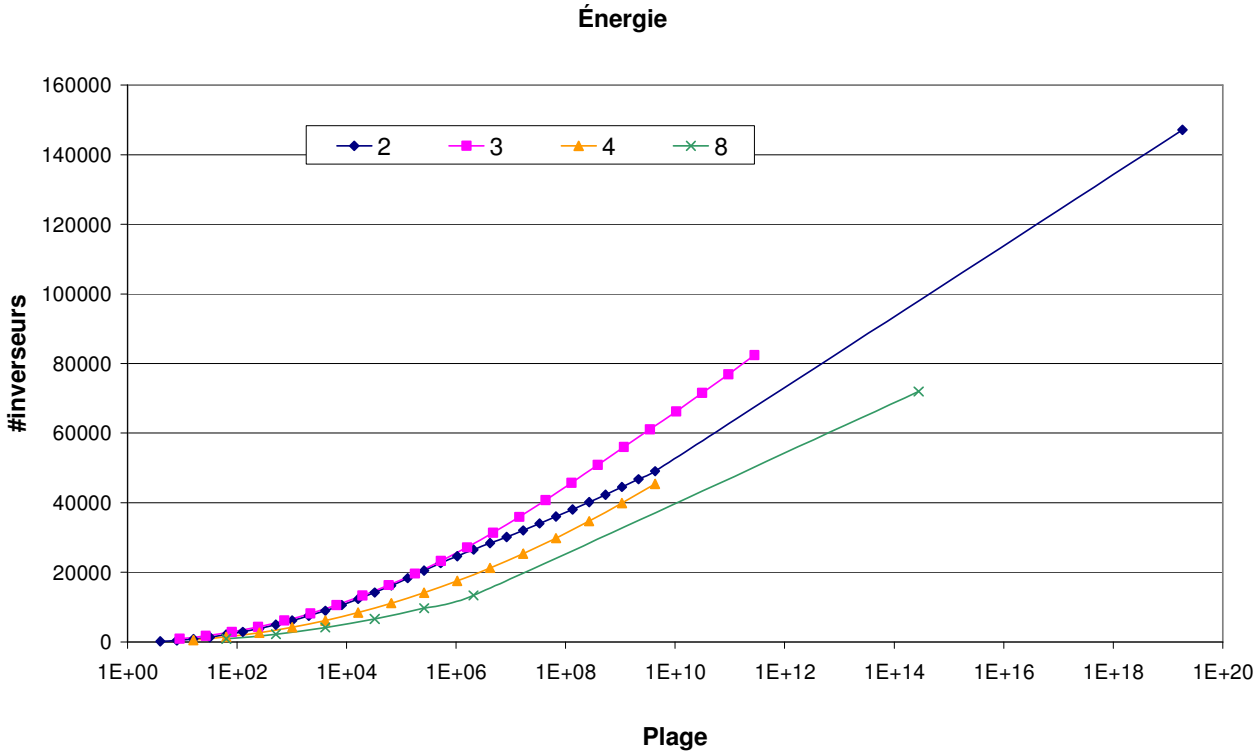


Figure VII.7 – Résultats de consommation d'énergie pour les multiplieurs signés en plusieurs bases.

La Figure VII.8 montre les résultats de retard pour plusieurs multiplieurs signés avec différents codages de données. L'augmentation de la base implique une réduction du retard critique des multiplieurs. Il est vrai que les multiplieurs en matrice étudiés au cours de ce chapitre sont les plus lents multiplieurs et, par conséquence, la réduction du retard critique est très peu sensible à l'augmentation de la base, c'est-à-dire, ils sont très peu sensibles à la réduction du nombre de digits.

La grande réduction par rapport à la base 2 est du à l'insertion d'un parallélisme dans le calcul de la matrice de calcul. La matrice généralisée présente deux matrices parallèles pour le calcul des produits partiels et ce parallélisme n'est pas implémenté dans la base binaire.

Les multiplieurs non signés évalués ont présentés des résultats similaires aux résultats des multiplieurs signés, simplement avec une légère réduction de complexité pour les multiplieurs non signés.

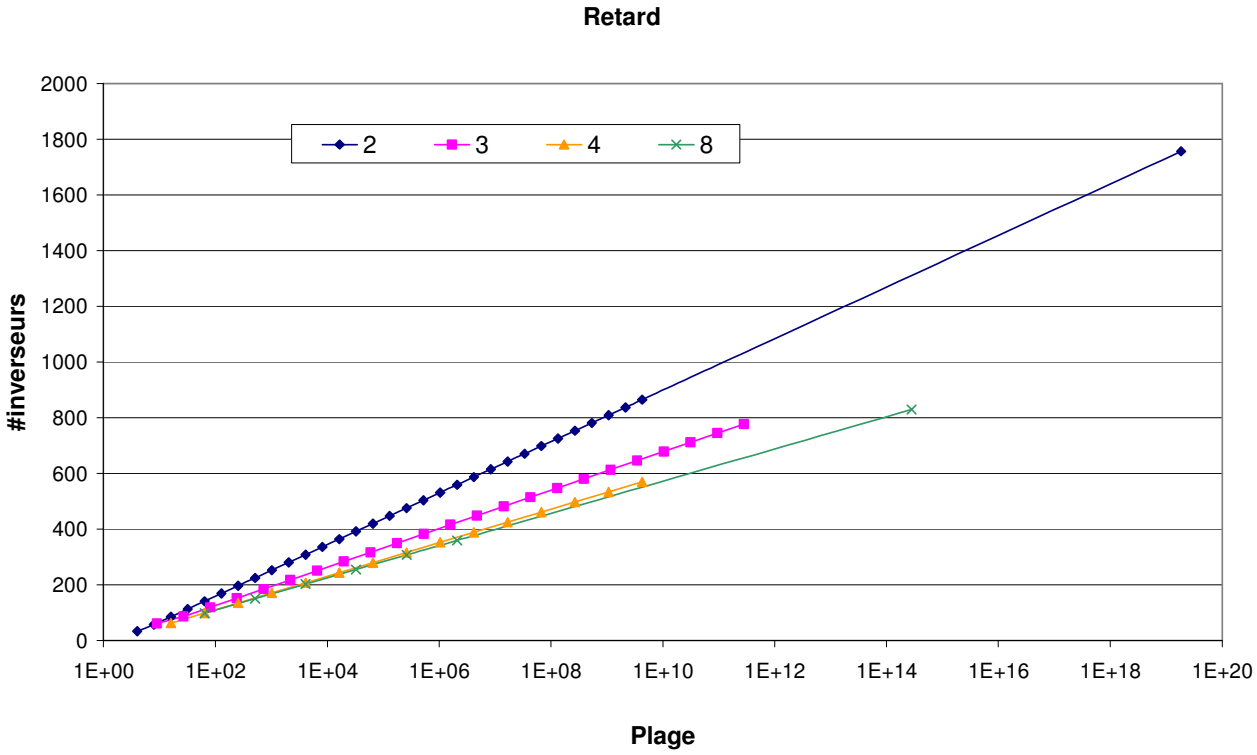


Figure VII.8 – Retard pour les différents multiplieurs signés en plusieurs bases.

En résumé, l'augmentation de la base est plus ou moins intéressante en fonction du nombre des digits à implémenter. La Figure VII.9 compare les différentes implémentations d'un multiplieur à 32 digits double rail pour les différentes bases. Dans ce cas, l'augmentation de la base fournit des implémentations plus performantes que l'implémentation double rail malgré l'augmentation de complexité.

À partir des résultats de la Figure VII.9, le multiplieur 16-digit en base *4* consomme 7,5% de moins et est 24,6% plus rapide que son équivalent double rail. En accord avec la métrique Et^2, si les deux multiplieurs – double rail et *1-parmi-4* – présentaient le même retard, la version généralisée *1-parmi-4* proposée consommerait seulement 40% de l'énergie consommée par la version double rail. Il est vrai que cette comparaison n'est pas très honnête, car la version généralisée *1-parmi-4* possède un certain degré de parallélisme qui n'est pas implémenté par la version double rail.

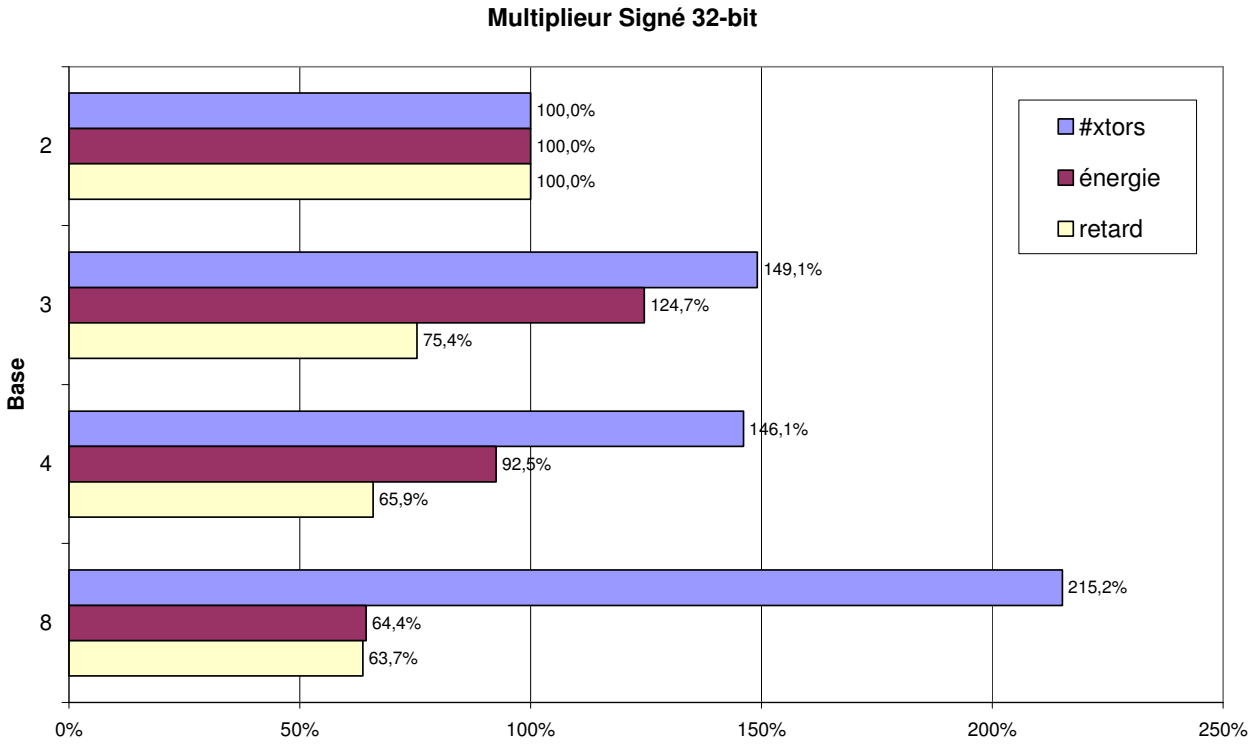


Figure VII.9 – Résultats pour différentes bases des multiplieurs équivalents à 32 digits double rail.

VII.4. Conclusions

Ce chapitre a montré comment la multiplication peut être généralisée pour les différents types de codages *1-parmi-N* insensibles aux délais. Les algorithmes de multiplication signée et non signée ont été étudiés. Cette généralisation a permis l'automatisation de la conception de multiplieurs pour toutes les bases.

Les résultats pour les différents multiplieurs évalués ont démontré que en fonction du nombre de digits à implémenter, l'augmentation de la base peut conduire à une réduction de la consommation et à une augmentation de la vitesse. Ces résultats laissent à croire que les codages en bases plus grandes sont intéressants pour les multiplieurs avec peu de digits d'entrées ou pour de multiplieurs interactifs – la multiplication est décomposée en multiplications et additions successives plus simples – où le nombre de digits reste limité.

Même si les multiplieurs en matrice sont les plus lents multiplieurs de la littérature, ces multiplieurs présentent une structure très régulière et les connexions entre les différentes cellules sont très limitées – connexions seulement entre cellules adjacentes. En conséquence, ce type de multiplieurs offre une implémentation VLSI efficace et ces multiplieurs peuvent facilement être « pipelinés ».

En plus, la généralisation ici présentée a ouvert le chemin pour l'étude d'autres architectures de multiplieurs qui sont aujourd'hui en cours de développement au sein du groupe CIS. À partir des cellules de calcul de produit partiel – non signé, signé/non signé, signé – proposées et généralisées au cours de cette thèse, il est maintenant possible de concevoir des architectures parallèles de multiplication, comme les arbres de multiplication de Wallace [VII.3], les arbres de multiplication de Dadda [VII.4]. Les arbres de Wallace et Dadda sont facilement implémentés en base binaire car tous les digits résultants des produits partiels, et des sommes successives, sont toujours binaires. Cependant, cette thèse a démontré certaines propriétés de la multiplication et il est clair que la généralisation de la construction des arbres de calcul pour la réduction de produit partiels doit impérativement passer par une étude minutieuse des signaux provenant des cellules de produit partiel. Avec la compréhension de l'algorithme généralisé de multiplication acquise au cours de ce travail, il est maintenant aussi envisageable d'étudier d'autres algorithmes de multiplication, comme par exemple l'algorithme de multiplication de Booth [VII.5].

VII.5. Références

[VII.1] B. Parhami, "Computer arithmetic: algorithms and hardware designs". Oxford University Press, New York, 2000. ISBN:0-19-512583-5.

[VII.2] C. R. Baugh, B. A. Wooley, "A two's complement parallel array multiplication algorithm", IEEE Transactions on Computers, 22:1045-1047, December 1973.

[VII.3] C. S. Wallace, "A suggestion for a fast multiplier", IEEE Transactions on Electronics Computers, 13:14-17, 1964.

[VII.4] L. Dadda, "Some schemes for parallel multipliers", Alta Frequenza, 34:349-356, 1965.

[VII.5] A. D. Booth, "A signed binary multiplication technique", Quarterly J. Mechanics and Applied Mathematics, vol. 4, pt. 2, pp. 236-240, June 1951.

Chapitre VIII

Conclusions et Perspectives

Les systèmes numériques actuels sont confrontés à une demande croissante de performances et, en même temps, de réduction de consommation d'énergie. En plus, avec l'augmentation de l'intégration qui permet la réalisation de systèmes de plus en plus complexes sur une même puce, la conception de tels systèmes est accomplie par l'assemblage de différents modules plus simple préalablement conçus. Ainsi, la nécessité de modularité, la réutilisation de modules existants et les contraintes de performance et consommation d'énergie sont aujourd'hui des obstacles majeurs à la conception de systèmes numériques notamment par la présence d'une horloge globale. L'utilisation des circuits asynchrones apparaît comme la réponse naturelle aux actuels problèmes auxquels sont confrontés les concepteurs.

Ainsi, cette thèse apporte une contribution à l'automatisation de la conception de chemins de données en logique asynchrone QDI, car le manque des outils de CAO adaptés à la conception de tels circuits est la principale faiblesse pour la large adoption de la solution asynchrone.

Tout d'abord, le chapitre II a présenté les caractéristiques des circuits asynchrones, leurs modes de fonctionnement et les potentialités des tels circuits face à leurs équivalents synchrones. Ce chapitre a aussi présenté une classification pour les circuits asynchrones en fonction de la complexité de réalisation et leur robustesse en fonction des hypothèses temporelles – retards sur les fils et portes logiques – faites au moment de la conception. Entre les classes de circuits asynchrones présentées au cours du chapitre II, il est possible de comprendre l'intérêt tout particulier de la classe de circuits QDI (quasi insensibles aux délais) qui est la cible de cette thèse. L'intérêt dans les circuits QDI provient principalement de la

robustesse de cette classe de circuits et de la possibilité de leur réalisation en utilisant les bibliothèques de cellules standard qui sont utilisées par les outils de conception synchrone.

Ensuite, le chapitre III a succinctement présenté l'environnement de conception TAST. L'environnement TAST est le résultat de l'effort du groupe CIS pour offrir un ensemble d'outils appropriés à la conception asynchrone. Ce chapitre a permis de mieux situer le contexte et les objectifs du travail développé au cours de cette thèse.

La première contribution de cette thèse est présentée dans le chapitre IV. Avant de démarrer l'étude des chemins de données asynchrones, il a été nécessaire d'établir une métrique capable de comparer différentes implémentations possibles. Un modèle d'estimation rapide de complexité, retard et consommation d'énergie a été développé. Le modèle proposé est indépendant de la technologie cible utilisée, cependant, la précision du modèle peut être ajustée avec les possibles variations particulières à une technologie de fabrication. En plus, il est possible de prendre en charge les courants de fuite (« leakage ») qui sont de plus en plus importants en technologies inférieures à 100nm. Avec la connaissance de la probabilité de commutation de chaque porte, il est possible d'estimer le nombre moyen de transistors commutant par cycle d'opération. Il est donc possible d'effectuer une correction de la valeur de l'énergie moyenne consommée en vue ajoutant l'énergie résultante des courants de fuite.

Le modèle proposé a été validé par comparaison avec les résultats de simulation électriques pour différents circuits. Le modèle s'est montré capable de comparer de façon équitable plusieurs implémentations de circuits asynchrones.

En possession d'un outil capable d'estimer rapidement le retard et la consommation d'énergie, il est maintenant possible de combiner ces valeurs dans une même métrique Et^n de qualité du circuit. Il est aussi possible de réaliser des modifications dans un circuit donné et rapidement évaluer l'impact des modifications dans la performance du circuit. Ainsi, avec le modèle d'estimation proposé par cette thèse, il est maintenant possible de choisir entre plusieurs implémentations et/ou entre plusieurs changements possibles pour atteindre les contraintes imposées par une spécification.

Le chapitre V montre les résultats de l'étude faite sur plusieurs styles de circuits QDI existants dans la littérature. Plusieurs circuits ont été implémentés avec les différents styles et ces circuits ont été évalués avec le modèle proposé par cette thèse. Cette étude a permis de comprendre l'intérêt de chaque style et de déterminer les conditions favorables à chaque possible implémentation. Ces résultats permettront d'améliorer les actuels moteurs de synthèse de l'environnement TAST. Même si les circuits QDI peuvent être réalisés avec les

bibliothèques de cellules standard, l'étude des différents styles de circuits a démontré que les résultats obtenus par les outils de synthèse asynchrone peuvent être encore améliorés par l'utilisation de bibliothèques de cellules dédiées à la conception de circuits asynchrones.

Un algorithme de décomposition de portes logiques a aussi été proposé. L'algorithme de décomposition proposé dans le chapitre V utilise les résultats du modèle d'estimation pour décomposer des portes à plusieurs entrées. L'algorithme de décomposition est optimal et algorithme permet d'obtenir des arbres de portes logiques qui consomment moins et qui sont plus performants que les arbres normalement utilisés par les concepteurs de systèmes numériques.

Une autre contribution de cette thèse est présentée dans le chapitre VI : une méthodologie de conception des additionneurs asynchrones QDI avec les codages *1-parmi-N*. La méthode proposée permet la génération d'additionneurs, aussi bien que de soustracteurs, à propagation de retenue, et des additionneurs rapides à préfixe parallèle pour toutes les bases. Ainsi, les opérateurs générés peuvent être insérés dans des chemins de données asynchrones *1-parmi-N* sans la nécessité d'interfaces de conversion de base. En plus, les additionneurs générés présentent une réduction de surface face à d'autres méthodes présentées dans la littérature.

Les résultats de l'étude de la conception des additionneurs a démontré que la base binaire, codage double rail asynchrone, n'est pas toujours la meilleure solution de représentation des valeurs pour la conception d'additionneurs rapides. L'augmentation de la base et la réduction du nombre de digits peuvent conduire à une réduction de la complexité et de l'énergie consommée et, en plus, à une diminution du retard critique. Ainsi, la surface, l'énergie et la vitesse ne sont pas toujours des contraintes contradictoires pour les circuits asynchrones.

Finalement, le chapitre VII montre la dernière contribution de ce travail de thèse. Ce chapitre a présenté une vision généralisée de la multiplication pour les différents types de codages *1-parmi-N* insensibles aux délais. Les algorithmes de multiplication signée et non signée ont été étudiés et cette étude a permis l'automatisation de la conception de multiplieurs pour toutes les bases. Les résultats pour les différents multiplieurs évalués ont démontrés que en fonction du nombre de digits à implémenter, l'augmentation de la base, comme pour les additionneurs, peut conduire à une réduction de la consommation et à une augmentation de la vitesse.

Les méthodes de génération d'additionneurs et de multiplieurs ont été complètement automatisées et ce module de génération d'opérateurs arithmétiques a été intégré dans l'environnement TAST.

Comme suite de cette thèse plusieurs chemins se sont ouverts. Le modèle proposé peut être raffiné pour de augmenter la précision des résultats obtenus par l'outil d'estimation comme par exemple par l'insertion de l'estimation de l'énergie de « leakage » qui devient de plus en plus importante pour les nouvelles technologies de fabrication de circuits intégrés.

L'étude des algorithmes de multiplication n'est qu'à son début. Avec les résultats de cette étude d'autres architectures de multiplieurs peuvent être développées. À partir des cellules de calcul de produit partiel – non signé, signé/non signé, signé – proposées et généralisées au cours de cette thèse, il est maintenant possible de concevoir de nombreuses architectures parallèles de multiplication et il est aussi envisageable d'étudier la possibilité de généralisation d'autres algorithmes de multiplication.

En plus, avec l'expertise acquise au cours de cette thèse, les autres codages de données insensibles aux délais peuvent être considérés, comme par exemple les codages du type *m-parmi-n*. En plus, il est possible d'envisager la conception d'opérateurs arithmétiques à largeur variable, c'est-à-dire, d'utiliser le codage de données pour détecter le nombre de digits significatifs de chaque variable et, ainsi, activer seulement la partie de l'opérateur nécessaire au calcul en cours. Ces techniques pourraient encore être utilisées pour produire des circuits asynchrones plus performants en termes de vitesse de calcul et de consommation d'énergie.

Bibliographie

[ABR 01] A. Abrial, J. Bouvier, M. Renaudin, P. Senn and P. Vivet, "A New Contactless Smart Card IC using On-Chip Antenna and Asynchronous Microcontroller", Journal of Solid-State Circuits, Vol. 36, 2001, pp. 1101-1107.

[BAU 73] C. R. Baugh, B. A. Wooley, “A two’s complement parallel array multiplication algorithm”, IEEE Transactions on Computers, 22:1045-1047, December 1973.

[BER 92] Kees van Berkel. Beware the isochronic fork. Integration, the VLSI journal, 13(2):103-128, June 1992.

[BIS 98] L. Bisdounis, S. Nikolaidis, O. Koufopavlou. Propagation Delay and Short-Circuit Power Dissipation Modeling of the CMOS Inverter. IEEE Transaction on Circuits and Systems-I: Fundamental Theory and Applications, 45(3). March 1998.

[BOO 51] A. D. Booth, “A signed binary multiplication technique”, Quarterly J. Mechanics and Applied Mathematics, vol. 4, pt. 2, pp. 236-240, June 1951.

[BOR 03] D. Borrione, M. Boubekeur, L. Mounier, M. Renaudin, A. Sirianni "Modeling CHP descriptions in Labeled Transitions Systems for an efficient formal validation of asynchronous circuit specifications". In Proc FDL'03, Frankfort, Germany, 23-26 September 2003.

[BOU 05] G.F. Bouesse, M. Renaudin, A. Witon, F. Germain, “A Clock-less low-voltage AES crypto-processor”, European Solid State CIRcuits Conference, ESSCIRC 2005, Grenoble, September 12-16, 2005.

[BRA 04] Alex Branover, Rakefet Kol, and Ran Ginosar. Asynchronous design by conversion: Converting synchronous circuits into asynchronous ones. In Proc. Design, Automation and Test in Europe (DATE), pages 870-875, February 2004.

[BRE 82] R. Brent, H. Kung. A Regular Layout for Parallel Adders. IEEE Transactions on Computers, vol. C-31, no. 3: 260-264. March, 1982.

[BRZ 89] J. A. Brzozowski and J. C. Ebergen. Recent developments in the design of asynchronous circuits. In J. Csirik, J. Demetrovics, and F. Gécseg, editors, Fundamentals of Computation Theory, FCT'89, volume 380 of Lecture Notes in Computer Science, pages 78-94, FCT'89, Szeged, Hungary, 1989. Springer-Verlag.

[BUR 92] R. Burch, F. Najm, P. Yang, T. Trick. McPower: A Monte Carlo approach to power estimation. IEEE/ACM Int. Conf. Computer-Aided Design. Santa Clara, CA. November 8-12, 1992. pp 90-97.

[CHA 73] T. J. Chaney, C. E. Molnar, “Anomaloues behavioural of synchronizer and arbiter circuits”, IEEE Transaction on Computers, C-22(4):421-422, April, 1973.

[CUM 94] U.V. Cummings, A.M. Lines, A.J. Martin, "An Asynchronous Pipelined Lattice Structure Filter", in Proceedings of the International Symposium on Advanced Research in Asynchronous Circuits and Systems, Salt Lake City, Utah, pp. 126-133, November 3-5, 1994.

[DAD 65] L. Dadda, “Some schemes for parallel multipliers”, Alta Frequenza, 34:349-356, 1965.

[DAV 92] Ilana David, Ran Ginosar, and Michael Yoeli. An efficient implementation of boolean functions as self-timed circuits. IEEE Transactions on Computers, 41(1):2-11, January 1992.

[DEA 94] M. Dean, D.Dill, M. Horowitz, "Self timed logic using current sensing completion detection", Journal of VLSI signal processing, N°7, pp 7 - 17, Feb.1994.

[DIN 03] A. Dinh-Duc. "Synthèse Automatique de Circuits Asynchrones QDI". PhD Thesis. INP de Grenoble, 2003.

[DIN 02] A. Dinh-Duc, L. Fesquet, M. Renaudin, "Synthesis of QDI Asynchronous Circuits from DTL-style Petri-Net" IWLS-02, 11th IEEE/ACM Internat. Workshop on Logic & Synthesis, New Orleans, Louisiana, June 4-7, 2002.

[FAN 96] Karl M. Fant and Scott A. Brandt. NULL conventional logic: A complete and consistent logic for asynchronous digital circuit synthesis. In International Conference on Application-specific Systems, Architectures, and Processors, pages 261-273, 1996.

[FRA 05] J. Fragoso, G. Sicard, M. Renaudin, “Comparaison Rapide de Performance de Circuits Asynchrones QDI”, in Recueil de Communications de FTFC’2005, Mai, 2005.

[FRA 03a] Fragoso J., Sicard G., Renaudin M., Power/Area trade-offs in 1-of-M parallel-prefix asynchronous adders. In Proc. PATMOS’03 – 13th International Workshop on Power and Timing Modeling, Optmization and Simulation. Torino, Italy, Septembre, 2003.

[FRAG03b] Fragoso J., Sicard G., Renaudin M., Automatic Generation of 1-of-M QDI Asynchronous Adders. In Proc. SBCCI'03 – 16th Symposium on Integrated Circuits and System Design. São Paulo, Brazil, Septembre, 2003.

[FRA 03c] Fragoso J., SICARD G., Renaudin M., Generalized 1-of-M QDI Asynchronous Adders. In: 3RD ACID-WG WORKSHOP, Heraklion, 2003.

[FUR 99] Stephen B. Furber, James D. Garside, Peter Riocreux, Steven Temple, Paul Day, Jianwei Liu, and Nigel C. Paver. AMULET2e: An asynchronous embedded controller. Proceedings of the IEEE, 87(2):243-256, February 1999.

[HAS 95] B. El Hassan, A. Guyot, M. Renaudin et V. Levering , "New self timed ring and their application to division and square root extraction", ESSCIRC'95, Lille, France, Sept. 1995.

[HAU 95] Scott Hauck. Asynchronous design methodologies: An overview. Proceedings of the IEEE, 83(1):69-93, January 1995.

[HED 87] N. Hedenstierna, K. O. Jeppson, CMOS Circuit Speed and Buffer Optimization. IEEE Transactions on Computer-Aided Design, CAS-6(2), March 1987.

[HEL 84] L. G. Heller, et al. Cascode voltage switch logic : a differencial CMOS logic family. In IEEE International Solid State Circuits Conference, 1984.

[HUF 64] D. A. Huffman. The synthesis of sequential switching circuits. In E. F. Moore, editor, Sequential Machines: Selected Papers. Addison-Wesley, 1964.

[HUF 52] D. Huffman. A method for the construction of minimum redundancy codes. In Proc IRE, 40(9):1098-101, 1952.

[HUI 90] C. M. Huizer. Power dissipation analysis of CMOS VLSI circuits by means of switch-level simulation. IEEE Europ. Solid-State Computer Conf. Grenoble, France, 1990. pp 61-64.

[KAR 04] S. Karandikar, S. Sapatnekar, "Fast Comparison of Circuit Implementations", in Proceedings of DATE'04, March 2004.

[KES 99] Joep Kessels and Paul Marston. Designing asynchronous standby circuits for a low-power pager. Proceedings of the IEEE, 87(2):257-267, February 1999.

[KOG 73] P. Kogge, H. Stone. A Parallel Algorithm for the Efficient Solution of General Class of Recurrence Equation. IEEE Transactions on Computers, vol. 22, no. 8: 783-791. August, 1973.

[KON 02a] Alex Kondratyev and Kelvin Lwin. Design of asynchronous circuits by synchronous CAD tools. In Proc. ACM/IEEE Design Automation Conference, June 2002.

[KON 02b] Alex Kondratyev and Kelvin Lwin. Design of asynchronous circuits using synchronous CAD tools. IEEE Design & Test of Computers, 19(4):107-117, 2002.

[LIG 00] Michiel Ligthart, Karl Fant, Ross Smith, Alexander Taubin, and Alex Kondratyev. Asynchronous design using commercial HDL synthesis tools. In Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems, pages 114-125. IEEE Computer Society Press, April 2000.

[MAR 01] A. Martin, “Towards an Energy Complexity of Computation”, Information Pocessing Letters, 77, 2001.

[MAR 98] A. Martin, “Remarks on Low-power Advantages of Asynchronous Circuits”, ESSCIRC’98: Low-power Systems on a Chip. September, 1998.

[MAR 97] A. J. Martin, A. Lines, R. Manohar, M. Nyström, P. I. Pénzes, R. Southworth, U. Cummings. The design of an asynchronous MIPS R3000 microprocessor. Proceedings of the 17th Conf. on Advanced Research in VLSI. IEEE Computer Society Press, 1997. pp 164-181.

[MAR 93] A.J. Martin, “Synthesis of Asynchronous VLSI Circuits”, Internal Report, Caltech-CS_TR-93-28, California Institute of Technology, Pasadena, 1993.

[MAR 92] A. Martin. Asynchronous datapaths and the design of an asynchronous adder. Formal Methods in System Design, 1(1):117-137, July, 1992.

[MAR 90a] Alain 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.

[MAR 90b] Alain J. Martin. Programming in VLSI: From communicating processes to delay-insensitive circuits. In C. A. R. Hoare, editor, Developments in Concurrency and Communication, UT Year of Programming Series, pages 1-64. Addison-Wesley, 1990.

[MAR 86] Alain J. Martin. Compiling communicating processes into delay-insensitive VLSI circuits. Distributed Computing, 1(4):226-234, 1986.

[MAU 03a] P. Maurine, J.B. Rigaud, F. Bouesse, G. Sicard, M. Renaudin, "Static Implementation of QDI asynchronous primitives", PATMOS'03, Torino, Italy, September 2003, pp.181-191.

[MAU 03b] P. Maurine, J.B. Rigaud, F. Bouesse, G. Sicard, M. Renaudin, "TAL: une bibliothèque de cellules pour le design de circuits asynchrones QDI", FTFC'03, Paris, France, May 2003.

[MEA 80] C. Mead, L. Conway. Introduction to VLSI Systems. Addison Wesley, 1980.

[MEN 91] T. Meng. Synchronization Design for Digital Systems. Kluwer Academic Publisher, 1991.

[MIL 65] R. E. Miller. Sequential Circuits and Machines, volume 2 of Switching Theory. John Wiley & Sons, 1965.

[MYE 01] Chris Myers. Asynchronous Circuit Design. John Wiley & Sons, 2001.

[MUL 62] David E. Muller. Asynchronous logics and application to information processing. In Symposium on the Application of Switching Theory to Space Technology, pages 289-297. Stanford University Press, 1962.

[MUL 59] David E. Muller and W. S. Bartky. A theory of asynchronous circuits. In Proceedings of an International Symposium on the Theory of Switching, pages 204-243. Harvard University Press, April 1959.

[NAJ 88] F. Najm, R. Burch, P. Yang, I. Hajj. CREST – A current estimation tool for CMOS circuits. IEEE Int. Conf. Computer-Aided Design. Santa Clara, CA. November 7-10, 1988. pp 204-207.

[NIE 99] Lars S. Nielsen and Jens Sparsø. Designing asynchronous circuits for low-power: An IFIR filter bank for a digital hearing aid. Proceedings of the IEEE, 87(2):268-281, February 1999.

[NIE 94] C. Nielsen. Evaluating of functional blocks designs. Technical Report ID-TR:1994-135, Dept. of Computer Science, Technical University of Denmark, 1994.

[OU 04] E. Ou, M. Nyström, "Energy-Delay Tradeoffs Involving Voltage Scaling in Synchronous and Asynchronous Circuits", Fourth ACiD-WG Workshop, Turku, Finland, 2004.

[PAR 00] B. Parhami, "Computer arithmetic: algorithms and hardware designs". Oxford University Press, New York, 2000. ISBN:0-19-512583-5.

[PEN 02a] P. I. Pénzes, A. J. Martin. An energy estimation method for asynchronous microprocessor. Design Automation and Test in Europe. March 4-8, Paris, France, 2002.

[PEN 02b] P. Pénzes, "Energy-Delay Complexity of Asynchronous Circuits", Thesis, Caltech, Pasadena, California, USA. 2002.

[REN 02] M. Renaudin, J.B. Rigaud, A. Dinh-Duc, A. Rezzag, A. Sirianni, J. Fragoso : "TAST CAD Tools", ASYNC'02 TUTORIAL, ISRN: TIMA--RR-02/04/01—FR, 2002.

[REN 00] M. Renaudin. Asynchronous circuits and systems: a promising design alternative. Journal of microelectronic engineering, 54: 133-149, 2000.

[REN 98] M. Renaudin, P. Vivet, F. Robin, "ASPRO-216 : a standard-cell Q.D.I. 16-bit RISC asynchronous microprocessor", Proc. of the Fourth International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC'98), San Diego - USA, 1998, p. 22-31.

[REZ 04] A. Rezzag. "Synthèse Logique de Circuits Asynchrones Micropipeline". PhD Thesis. INP de Grenoble, 2004.

[RIG 02] J-B. Rigaud, "Spécification de bibliothèques pour la synthèse de circuits asynchrones", PhD Thesis, INP Grenoble, 2002.

[SAK 90] T. Sakurai, A. R. Newton. Alpha-Power Law MOSFET Model and its Applications to CMOS Inverter Delay and Other Formulas. IEEE Journal of Solid-State Circuits, 25(2), April 1990.

[SEI 80] Charles L. Seitz. System timing. In Carver A. Mead and Lynn A. Conway, editors, Introduction to VLSI Systems, chapter 7. Addison-Wesley, 1980.

[SEN 92] E. Sentovich, et al. SIS: a system for sequential circuits synthesis. Technical Report UCB/ERL M92/41, UC Berkeley, 1992.

[SPA 01] Jens Sparsø and Steve Furber, editors. Principles of Asynchronous Circuit Design: A Systems Perspective. Kluwer Academic Publishers, 2001.

[SPA 93] Jens Sparsø and Jørgen Staunstrup. Delay-insensitive multi-ring structures. Integration, the VLSI journal, 15(3):313-340, October 1993.

[SPA 92] J. Sparsø, J. Staunstrup, M. Dantzer-Sørensen. Design of delay insensitive circuits using multi-ring structures. In: Proceedings of EURO–DAC, Hamburg, Germany, 1992. pp. 15-20.

[SKL 60] J. Sklansky. Conditional-sum addition logic, IRE Trans. Electronic Computers, 9(2):226-231, 1960.

[SUT 99] I. Sutherland, B. Sproull, D. Harris. Logical effort: designing fast CMOS circuits. Morkan Kaufmann Publishers, Inc. San Francisco, CA, 1999. ISBN 1-55860-557-6. 240 pp.

[SUT 89] I.E. Sutherland, “Micropipelines”, Communication of the ACM, Volume 32, N°6, June 1989.

[TIJ 86] Jan Tijmen Udding. A formal model for defining and classifying delay-insensitive circuits. Distributed Computing, 1(4):197-204, 1986.

[TSU 93] C. Y. Tsui, M. Pedran, A. M. Despain. Efficient estimation of synamic power consumption under a real delay model. IEEE Int. Conf. Computer-Aided Design. Santa Clara, CA. November 7-11, 1993. pp 224-228.

[UNG 71] S. H. Unger. Asynchronous sequential switching circuits with unrestricted input changes. IEEE Transactions on Computers, 20(12):1437-1444, December 1971.

[UNG 69] S. H. Unger. Asynchronous Sequential Switching Circuits. Wiley-Interscience, John Wiley & Sons, Inc., New York, 1969.

[VIV 01] P. Vivet, "Une Méthodologie de Conception de Circuits Intégrés Quasi-Insensibles aux Délais: Application à l'Etude et à la Réalisation d'un Processeur RISC 16-bits Asynchrone", Thèse de Doctorat, Institut National Polytechnique de Grenoble, 2001.

[VEM 94] S. R. Vemuru, N. Scheinberg. Short-Circuit Power Dissipation Estimation for CMOS Logic Gates. IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, 41(11). November 1994.

[WAL 64] C. S. Wallace, “A suggestion for a fast multiplier”, IEEE Transactions on Electronics Computers, 13:14-17, 1964.

[WES 94] N. H. West, K. Eshraghian. Principles of CMOS VLSI Design. Addison Wesley, 1994.

[WES 67] Wesley A. Clark. Macromodular computer systems. In AFIPS Conference Proceedings: 1967 Spring Joint Computer Conference, volume 30, pages 335-336, Atlantic City, NJ, 1967. Academic Press.

[WIL 94] T.E. Williams, "Performance of iterative computation in self timed rings", Journal of VLSI signal processing, N°7, pp 17 - 31, Feb.1994.

[WIL 91] T.E. Williams, "Self timed rings and their application to division", Ph.D dissertation, Stanford university, May 1991.

[YUN 96] K. Y. Yun, P. A. Beerel, and J. Arceo. “High-performance asynchronous pipeline circuits”. In Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems. IEEE Computer Society Press, March 1996.